
Universidade de Pernambuco
Departamento de Sistemas e Computação (DSC)

Pós-graduação em Engenharia da Computação

# Métodos para detecção de novidades em problemas de classificação e em séries temporais baseados no algoritmo do vizinho mais próximo com minimização do risco estrutural

George Gomes Cabral

Dissertação de Mestrado

Recife
julho de 2008

Universidade de Pernambuco
Departamento de Sistemas e Computação (DSC)

George Gomes Cabral

# Métodos para detecção de novidades em problemas de classificação e em séries temporais baseados no algoritmo do vizinho mais próximo com minimização do risco estrutural

*Trabalho apresentado ao Programa de Pós-graduação em Engenharia da Computação do Departamento de Sistemas e Computação (DSC) da Universidade de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.*

Orientador: *Prof. Dr. Adriano Lorena Inácio de Oliveira*

Recife
julho de 2008

*<DIGITE A DEDICATÒRIA AQUI>*

# Agradecimentos

<DIGITE OS AGRADECIMENTOS AQUI>

*Se avexe não*
*Toda caminhada começa*
*No primeiro passo*
*A natureza não tem pressa*
*Segue seu compasso*
*Inexorávelmente chega lá...*
—ACCIOLY NETO (A Natureza das Coisas)

# Resumo

Na classificação com exemplos de uma única classe (*one-class classification*), um objeto novidade pode ser definido como um objeto sem nenhuma relação conhecida com qualquer dos objetos apresentados ao classificador durante a fase de treinamento. Objetos da classe novidade, na prática, podem se localizar em diversos pontos diferentes no espaço, sendo a coleta de objetos dessa classe difícil, ou até mesmo, impossível. Essa dificuldade na obtenção de objetos da classe novidade motiva a criação de métodos capazes de identificar novidades que sejam treinados apenas a partir de objetos considerados da classe normal. Exemplos de aplicações desses métodos são: detecção de fraudes em folhas de pagamento, detecção de intrusão em redes de computadores e detecção de anomalias clínicas em medicina.

Essa dissertação propõe a criação de novos métodos baseados na classificação com exemplos de uma única classe para a aplicação de detecção de novidades em dados atemporais (problemas de classificação) e temporais (séries temporais). O primeiro método, *k*NNDDSRM (*k-Nearest Neighbor Data Description with Structural Risk Minimization*), pode ser aplicado tanto a dados temporais quanto atemporais. Uma versão mais eficiente (fazendo uso de funções de kernel) também foi desenvolvida a fim de alcançar descrições mais flexíveis dos objetos da classe normal. O conceito SRM (*Structural Risk Minimization* - Minimização do Erro Estrutural) foi usado nos métodos no intuito de reduzir o tamanho do conjunto de treinamento sem comprometer o poder de generalização do classificador. Foram realizados experimentos, para detecção de novidades em séries temporais e dados atemporais. O resultados obtidos mostram que o método proposto obteve desempenhos semelhantes à técnica considerada mais sofisticada para detecção de novidades utilizando a abordagem *one-class*, SVDD.

A detecção de novidades em séries temporais merece uma maior atenção. Novidades em séries temporais carregam a característica não intuitiva de não necessariamente se apresentarem de maneira dispersa ou fora da descrição normal dos dados. Nesses casos, classificadores como o kernel *k*NNDDSRM e SVDD, costumam obter resultados pobres. Pensando nisso, foi desenvolvido um segundo método para detecção de novidades apenas em séries temporais baseado na transição de estados na série temporal. Esse método faz uso de técnicas de *clustering* para descoberta de estados na série e também da técnica NNSRM para a obtenção de um conjunto de referências mínimo. Experimentos realizados em três séries temporais com a característica citada acima atestam a eficiência do método.

**Palavras-chave:** Classificação com Exemplos de uma Única Classe, Vizinho mais Próximo, detecção de novidades, Redução de Protótipos e Minimização do Erro Estrutural

# Abstract

In one-class classification, a novelty object may be defined as an object that does not have any known relationship with the objects available to the classifier during the training phase. Objects belonging to the novel class, in pratice, may occur in several different locations in the input space. Therefore, collecting such objects is a hard, or even, impossible task. This difficulty motivates the investigation of methods capable of identifying novelties trained only from objects considered of the normal class. Examples of pratical applications of such methods are: fraud detection in payroll, intrusion detection in computer networks and clinical anomaly detection in medicine.

This work proposes and investigates novel methods based on the principle of one-class classification aiming to detect novelties in both classification problems as well as in time series. The first method, referred to as *k*NNDDSRM (*k*-Nearest Neighbor Data Description with Structural Risk Minimization), can be applied to both problems, that classification problems and novelty detection in time series. We have also developed another method, the kernel *k*NNDDSRM, which uses kernel functions aiming to produce more flexible descriptions of the object from the normal class. The principle of SRM (Structural Risk Minimizations) was also employed to develod these methods, aiming to reduce the size of the training set without prejudicing the generalization power of the classifier. Experiments were carried out considering both novelty detection in time series and in classification problems. The results reported in this work show that the proposed method has achieved similar performances to the technique considered more sophisticated for novelty detection using the one-class classification approach, SVDD (Support Vector Data Description).

Novelty detection applied to time series data requires a greater attention. Novelties in time series carries the non intuitive characteristic of not necessarily occuring in the sparse space or out of the normal description of the data. In these cases, classifiers such as the kernel *k*NNDDSRM and SVDD may obtain poor results for some time series. To tackle such time series, we developed another method specifically designed for novelty detection in time series data. The proposed method is based on state transition in the time series. This methods uses clustering techniques for discovering states in the time series and also uses the NNSRM technique aiming to obtain a minimum reference set. Experiments carried out in three time series with the characteristic cited above attests the efficiency of the method.

**Keywords:** one-class classification, Nearest Neighbor, novelty detection, prototype reduction and Structural Risk Minimization

# Sumário


**1 Introdução** **1**
1.1 Motivação 2
1.2 Estado da Arte 4
1.3 Objetivos 7
1.4 Organização da dissertação 8

**2 Redução de Protótipos e Minimização do Risco Estrutural** **10**
2.1 Redução de Protótipos 10
2.1.1 Características de Métodos para Redução de Protótipos 11
2.1.1.1 Representação 11
2.1.1.2 Direção de Busca 12
2.1.1.3 Instâncias na Borda x Instâncias Centrais 13
2.1.1.4 Função Distância 14
2.1.1.5 Votação 15
2.1.1.6 Estratégias de Avaliação 15
2.2 Minimização do Risco Estrutural - SRM 16
2.3 Técnicas para Redução Protótipos 17
2.3.1 Condensed Nearest Neighbor 17
2.3.2 Prototypes for Nearest Neighbor - PNN 19
2.3.3 Nearest Neighbor with Structural Risk Minimization - NNSRM 20
2.3.4 Adaptive Recursive Partitioning Prototype Reduction Scheme - ARP_PRS 22
2.4 Considerações Finais 23

**3 Classificação com Exemplos de uma Única Classe** **25**
3.1 Introdução 25
3.2 Kernel Trick 26
3.3 Métodos para Classificação com Exemplos de Única Classe 27
3.3.1 *k*NNDD 28
3.3.2 SVDD - *Support Vector Data Description* 30
3.3.3 One-class SVM 33
3.3.4 kMeans Data Description 34
3.3.5 Parzen Data Description 35
3.3.6 Aprendizado de regras e estados para detecção de novidades em séries temporais 36
3.4 Métodos para Avaliação de Desempenho 37

3.5 Pré-processamento de Séries Temporais e Detecção de novidades com Exemplos de uma Única Classe para Séries Temporais 41
3.6 Considerações Finais 42

**4 Método Propostos para Detecção de Novidades 44**
4.1 NNDDSRM e suas Extensões 44
4.1.1 NNDDSRM 44
4.1.2 *k*NNDDSRM 46
4.1.3 kernel *k*NNDDSRM 47
4.1.4 Complexidade Computacional do NNDDSRM 50
4.1.4.1 Pior caso 50
4.1.4.2 Melhor caso 51
4.2 Detecção de Novidades Baseada na Transição entre Estados da Série 52
4.3 Considerações Finais 56

**5 Experimentos 58**
5.1 Exemplo de custo computacional do kernel *k*NNDDSRM em um Problema Real 59
5.2 Experimentos para Detecção de Novidades em Bases de Dados Atemporais 59
5.2.1 Experimentos Realizados com a Base Distribuição Gaussiana 60
5.2.2 Experimentos Realizados com a Base Banana 62
5.2.3 Experimentos com a base Biomed 65
5.2.4 Experimentos com a base Wisconsin Breast Cancer 67
5.2.5 Experimentos com a base Diabetes 68
5.2.6 Experimentos com a base Soybean 70
5.2.6.1 Classe 16 71
5.2.6.2 Classe 17 71
5.2.7 Experimentos com a Base Iris 72
5.2.7.1 Experimentos com a classe Iris Setosa como novidade 73
5.2.7.2 Experimentos com a classe Iris Versicolour como novidade 74
5.2.7.3 Experimentos com a classe Iris Virgínica como novidade 75
5.3 Influência dos Parâmetros Ajustáveis do Método kernel *k*NNDDSRM 76
5.3.1 Análise do Parâmetro *k* 77
5.3.2 Análise do Parâmetro $\sigma$ e *d* no kernel *k*NNDDSRM 78
5.3.3 Análise do Parâmetro Função de Kernel 78
5.3.4 Análise do Parâmetro *fracrej* 79
5.4 Detecção de novidades em Séries Temporais 80
5.4.1 Comparação dos Resultados Obtidos pelos Métodos *k*NNDDSRM e kernel *k*NNDDSRM com os Resultados dos Métodos One-class SVM e *k*NNDD 81
5.4.1.1 Experimentos com a série Respiração 82
5.4.1.2 Experimentos com a série Space Shuttle Marotta Valve Series 84
5.4.1.3 Experimentos com a série Seno 85
5.4.2 Experimentos com bases codificadas em múltiplas dimensões 87
5.4.2.1 Seno 88

5.4.2.2 Laser 89
5.4.3 Detecção de Novidades em Séries Temporais - Método baseado na Transição Entre Estados 89
5.4.3.1 Resultados Obtidos 90
5.5 Considerações Finais 91

**6 Conclusão 93**
6.1 Contribuições 94
6.2 Propostas para Trabalhos Futuros 95
6.2.1 Proposta 1 95
6.2.2 Proposta 2 95
6.2.3 Proposta 3 97
6.3 Considerações Finais 97

# Lista de Figuras


1.1 Ilustração da classificação com exemplos de múltiplas classes e de uma única classe. 2
1.2 Exemplo de novidade em série temporal 3
1.3 Exemplo de série discretizada pelo método SAX 7

2.1 Eliminação de objetos do conjunto de treinamento fora da região de conflito 11
2.2 Exemplo de fronteira entre as classes para o caso da escolha de pontos centrais para a discriminação entre as classes 14
2.3 Exemplos de funções de distância entre vetores de dados 15
2.4 Minimização do erro estrutural 18
2.5 Pseudo código do *Condensed Nearest Neighbor* 19
2.6 Pseudo código do *Prototypes for Nearest Neighbor* 20
2.7 Pseudo código do *Nearest Neighbor with Structural Risk Minimization* (NNSRM) para um problema com apenas 2 classes. 21
2.8 Pseudo código do *Nearest Neighbor with Structural Risk Minimization* (NNSRM) para um problema com múltiplas classes. 21
2.9 O conjunto de treinamento completo e seus subconjuntos aleatóriamente divididos, onde os objetos são representados por "."e "*". (a) Conjunto de treinamento completo contendo 200 objetos. (b) Subconjunto 1 contendo 100 objetos. (c) Subconjunto 2 contendo 100 objetos. (d) Diferença entre as fronteiras alcançadas pelo subconjunto 1 (linha pontilhada) e 2 (linha sólida). 23
2.10 Pseudo código do *Adaptive Recursive Partitioning Prototype Reduction Scheme* 24

3.1 O mapeamento para um espaço de características pode simplificar a tarefa de classificação 27
3.2 NNDD: Exemplo de tomada de decisão. 29
3.3 *k*NNDD: Exemplo de tomada de decisão. 30
3.4 Hiperesfera contendo os dados normais, descrita pelo centro $a$ e pelo raio $R$. Três objetos se encontram no limite da descrição; esses objetos são chamados de vetores de suporte. Um objeto $x_i$ está fora da descrição pois tem $\xi_i > 0$. 31
3.5 Abordagem para detecção de novidades baseada na transição de estados proposta em [SC05] 37
3.6 Curva ROC perfeita, $AUC = 1$. 38
3.7 Curvas ROC representativas de três diferentes graus de capacidade de discriminação. 39
3.8 Sobreposição de duas distribuições hipotéticas. 40

3.9 Exemplo de série temporal que apresenta duas regiões da classe novidade (regiões na cor vermelha). 42
3.10 Série temporal com trecho detectado como novidade. 43

4.1 Exemplo de tomada de decisão realizada pelo classificador NNDDSRM. 46
4.2 Exemplo de ocorrência de novidade em séries temporais onde a série permanece por um tempo prolongado em um mesmo estado 53
4.3 Diagrama da fase de treinamento realizada pela abordagem proposta para detecção de novidades baseada em estados 54
4.4 Ilustração dos vários estados contidos em uma série temporal 54
4.5 Diagrama da fase de teste realizada pela abordagem proposta para detecção de novidades baseada em estados 55

5.1 Custo computacional de aplicações dos métodos kernel *k*NNDDSRM e one-class SVM. 60
5.2 Distribuições Gaussianas Geradas para o Treinamento e Teste. 61
5.3 Descrições geradas pelos métodos: a) kmeans_dd; b) SVDD; c) *k*NNDDSRM; e d) kernel *k*NNDDSRM. 63
5.4 Curvas ROC obtidas para a base de Distribuições Gaussianas. 64
5.5 Conjuntos de treinamento e teste gerados em formato de banana. 64
5.6 Curvas ROC geradas para a base de dados banana. 65
5.7 Descrições geradas pelos métodos: a) kmeans_dd; b) SVDD; c) *k*NNDDSRM; e d) kernel *k*NNDDSRM. 66
5.8 Curvas ROC geradas para a base de dados biomed. 67
5.9 Curvas ROC geradas para a base de dados Wisconsin Breast Cancer. 69
5.10 Curvas ROC geradas para a base de dados Diabetes. 70
5.11 Curvas ROC geradas para a base de dados Soybean com a classe 16 como novidade. 72
5.12 Curvas ROC geradas para a base de dados Soybean com a classe 17 como novidade. 73
5.13 Curvas ROC geradas para a base de dados Iris com a classe Iris Setosa como novidade. 74
5.14 Curvas ROC geradas para a base de dados Iris com a classe Iris Versicolour como novidade. 75
5.15 Curvas ROC geradas para a base de dados Iris Virgínica como novidade. 76
5.16 Influência do parâmetro *k* na classificação. a) k = 1; b) k = 3. 78
5.17 Influência do parâmetro $\sigma$ na classificação. 79
5.18 Variação da AUC com diferentes funções de kernel. 80
5.19 Influência do parâmetro $fracrej$ na classificação. A figura da esquerda tem *fracrej* = a 5%, enquanto que a da direita tem *fracrej* = 25%. 80
5.20 Base de dados Respiração de um paciente original. 82
5.21 Resultados dos testes com a base Respiração usando os $\theta$ da tabela 5.12. 83
5.22 Série temporal Space Shuttle Marotta Valve original. 84

5.23 Resultados dos testes com a base Space Shuttle Marotta Valve usando os $\theta$ da tabela 5.13. 85
5.24 Série temporal Seno (Conjunto de teste). 86
5.25 Resultados dos testes com a base Seno usando os $\theta$ da tabela 5.14. 87
5.26 Exemplo de detecção de novidade utilizando várias janelas simultaneamente. 88
5.27 Resultado obtido do do teste para a série Seno utilizando várias janelas simultaneamente. 88
5.28 Resultado obtido do do teste para a série Laser utilizando várias janelas simultaneamente. 89
5.29 Resultado obtido para a detecção de novidades baseada em estados aplicada ao conjunto de teste da série $ECG_1$ 90
5.30 Resultado obtido para a detecção de novidades baseada em estados aplicada ao conjunto de teste da série $ECG_2$ 91
5.31 Resultado obtido para a detecção de novidades baseada em estados aplicada ao conjunto de teste da série Space Shuttle Marotta Valve 91

6.1 a) Aplicação do kernel *k*NNDDSRM a uma distribuição de dados formada por agrupamentos dispersos entre si; b)Esquema onde um pré-processamento, visando a descoberta de *clusters* de dados, é realizada com a posterior aplicação do kernel *k*NNDDSRM em cada *cluster* encontrado 96

# Lista de Tabelas


5.1 Parâmetros utilizados pelos métodos da seção 5.1. 58
5.2 Melhores resultados obtidos por cada método aplicado à base de distribuições gaussianas. 61
5.3 Melhores resultados obtidos por cada método aplicado à base banana. 65
5.4 Melhores resultados obtidos por cada método aplicado à base biomed. 67
5.5 Melhores resultados obtidos por cada método aplicado à base Wisconsin Breast Cancer. 68
5.6 Melhores resultados obtidos por cada método aplicado à base Diabetes. 70
5.7 Melhores resultados obtidos por cada método aplicado à base Soybean com a classe 16 como novidade. 71
5.8 Melhores resultados obtidos por cada método aplicado à base Soybean com a classe 17 como novidade. 72
5.9 Melhores resultados obtidos por cada método aplicado à base Iris com a classe 1 como novidade. 74
5.10 Melhores resultados obtidos por cada método aplicado à base Iris com a classe 2 como novidade. 75
5.11 Melhores resultados obtidos por cada método aplicado à base Iris com a classe Virginica como novidade. 76
5.12 Resultados obtidos para a série temporal Respiração 83
5.13 Resultados obtidos para a série temporal Space Shuttle Marota Valve Series 84
5.14 Resultados obtidos para a série temporal Seno 86
5.15 *TN*, *TP*, *FN* and *FP* ocorrências para os métodos aplicados à série temporal Seno 86
5.16 Parâmetros utilizados nos experimentos para detecção de novidades baseada na transição de estados. 90

CAPÍTULO 1

# Introdução

Em muitos problemas de classificação não existem regras explicitas para distinção entre objetos, porém exemplos desses objetos podem ser obtidos facilmente. Em reconhecimento de padrões, ou aprendizado de máquinas, tenta-se inferir um classificador a partir de um conjunto limitado de amostras (exemplos) de treinamento. A meta é obter classificadores capazes de predizer a classe de objetos desconhecidos durante a fase de treinamento.

Na classificação de padrões, um objeto novidade pode ser definido como um objeto sem nenhuma relação conhecida com qualquer dos objetos apresentados ao classificador na fase de treinamento. Como exemplo, suponha um classificador que diferencie maçãs de pêras. Caso a esse classificador seja apresentado um objeto diferente de maçã ou pera, espera-se que esse objeto seja classificado como uma novidade. Um dos problemas dos diversos métodos para classificação investigados é que eles necessitam de amostras artificiais para a representação da classe novidade (*outlier*) durante a fase de treinamento. Já se mostrou que, em problemas de classificação convencionais, a utilização de tais amostras não é eficiente, pois é difícil selecionar as amostras mais adequadas para representar o que seriam novidades [Oli04]. Portanto, é importante investigar métodos de treinamento que não necessitem de amostras da classe novidade (amostras negativas) durante o treinamento. Este é o caso da classificação com exemplos de uma única classe (*one-class classifiers*) [SEK05][Seo07][MKH93][MP03].

A figura 1.1, retirada de [Tax01], ilustra o exemplo já citado das maçãs e pêras. Nela um classificador convencional e um classificador para exemplos de uma única classe são aplicados a um conjunto de dados contendo maçãs e pêras, representadas por dois atributos (peso e largura dos frutos). A linha sólida representa o classificador convencional que distingue entre pêras e maçãs, enquanto que a linha pontilhada seria gerada por um classificador *one-class*, também chamado de método de descrição de dados (*data description*). Essa descrição pode detectar de forma correta a maçã representando o objeto novidade no canto inferior direito da imagem, enquanto que o classificador convencional simplesmente a classificaria como uma pêra.

Dessa forma, podemos resumir o problema da classificação com exemplos de uma única classe em algumas de suas características mais relevantes.

- Apenas informação sobre a classe normal é disponível na fase de treinamento (nenhum conhecimento a-priori da classe novidade existe);
- A fronteira entre as duas classes deve ser estimada a partir de amostras apenas da classe normal;
- Objetivo: definir a borda da descrição dos dados da classe normal (para aceitar o máximo possível de objetos da classe normal maximizando também a detecção de objetos novidade).

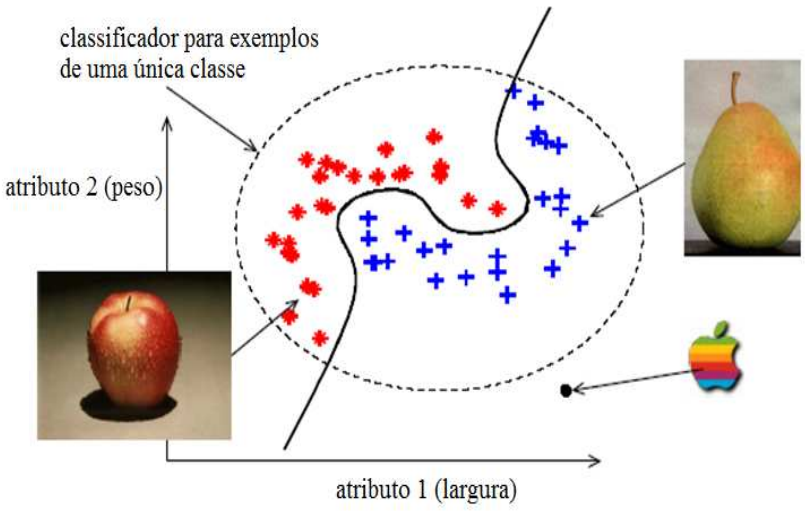


**Figura 1.1** Ilustração da classificação com exemplos de múltiplas classes e de uma única classe.

Séries temporais podem estar associadas a qualquer área do conhecimento humano. Exemplos de séries temporais incluem eletrocardiogramas, índices da Bolsa de Valores, índice pluviométrico de uma região, etc. Novidades em séries temporais podem ser, informalmente, definidas como valores ou seqüencia de valores inesperados. Em séries temporais, novidades não necessariamente ocorrem em pontos com valores de alta ou baixa magnitude; novidades podem ocorrer, também, em pontos com magnitudes consideradas dentro do intervalo normal de valores. A figura 1.2 ilustra um caso onde a novidade ocorre em um ponto fora do intervalo normal de magnitudes e também gera um comportamento anormal dentro do intervalo normal de magnitudes. O retângulo vermelho delimita a região considerada novidade. Exemplos de aplicações de detecções de novidades em séries temporais são detecção de falhas em máquinas e detecção de fraudes em sistemas financeiros [DF95] [OeABeALMS03].

Esta dissertação introduz dois novos métodos, e algumas variações, para detecção de novidades e suas aplicações a dados temporais (séries temporais) e atemporais (problemas de classificação convencionais). O primeiro método proposto baseia-se, principalmente, no método Nearest Neighbor Data Description [Tax01] (NNDD) e no conceito de minimização do erro estrutural [VC74] (SRM). O método será referenciado ao longo deste documento como *k*NNDDSRM (*k*-Nearest Neighbor Data Description with Structural Risk Minimization). O segundo método proposto se aplica apenas à detecção de novidades em séries temporais e se baseia na transição de estados da série temporal e na minimização do erro estrutural.

## 1.1 Motivação

A abordagem tomada na detecção de novidades com classificação com exemplos de uma única classe é oposta à abordagem tomada na técnicas de detecção de assinatura (*signature detection*). Em detecção de assinatura, informações sobre objetos da classe novidade são explicitadas já durante a fase de treinamento; o classificador simplesmente tenta detectar a ocorrência desses

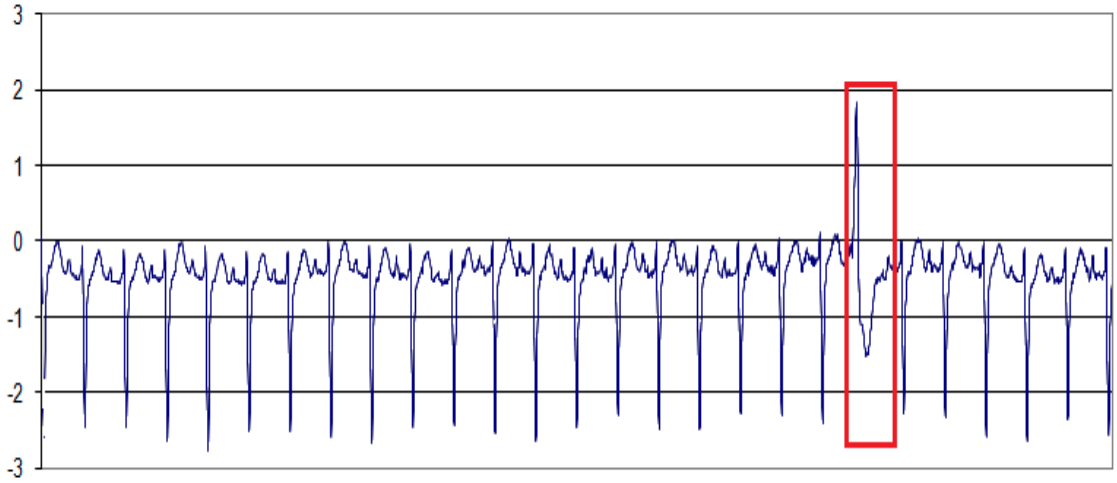


**Figura 1.2** Exemplo de novidade em série temporal

objetos durante o uso real do mesmo. Alarmes falsos (objetos da classe normal detectados como novidades) são raros no uso de detecção de assinatura pelo fato de o algoritmo ter sido programado para saber exatamente em quais condições objetos da classe novidade se enquadram. Contudo, detecção de assinatura é incapaz de detectar novos tipos de novidades, ou seja, tipos de novidades que não foram explicitadas na fase de treinamento. Apesar de a detecção de novidades com classificação com exemplos de uma única classe produzir mais alarmes falsos do que sistemas com detecção de assinatura, ela tem a significante vantagem de ser capaz de detectar novidades com características desconhecidas. Sistemas de anti-virus usam detecção de assinatura para a detecção de códigos maliciosos. Esses sistemas são confiáveis na detecção de vírus conhecidos, existe uma baixa incidência de falsos alarmes, porém, eles não são capazes de detectar novos tipos de vírus.

O trabalho reportado nessa dissertação tem como principal motivação o desenvolvimento de um métodos aplicáveis à detecção de novidades quando objetos da classe novidade são desconhecidos. Aplicações com essas características são incontáveis, pois no mundo real a "classe novidade"é quase sempre desconhecida. Exemplos de aplicações no mundo real são:

**Detecção de fraudes em folhas de pagamento** Essa é uma aplicação muito importante pois despesas com funcionários representam uma porção significante do orçamento de uma organização, seja ela pública ou privada. Como exemplo, o governo brasileiro direciona em torno de 50% de sua receita a gastos com pessoal. Esse é um cenário onde um sistema de auditoria de gastos seria de grande importância.

**Detecção de fraudes em operações no comércio eletrônico** Em sistemas tecnológicos, atividades fraudulentas têm ocorrido em diversas aplicações, tais como comunicação móvel, *on-line banking* e comércio eletrônico. Segundo a empresa norte-americana eMarketer, as fraudes virtuais fizeram com que comerciantes americanos perdessem, em 2004, US$ 2,6 bilhões. O montante representa 1,8% do total das vendas realizadas através do comér-

cio eletrônico naquele país e está relacionado ao medo que os internautas têm de realizar transações on-line. Alguns trabalhos da literatura têm reportado a preocupação de empresários com o crescimento de fraudes em operações de comércio eletrônico, por exemplo, [Sec00].

**Detecção de intrusão** O processo de detecção de intrusão se caracteriza por identificar e relatar atividades maliciosas agindo em computadores e/ou recursos da rede. Para que o sistema seja apto a determinar o que é um ataque no trafego de dados, ele deve ser previamente ensinado a reconhecer atividades normais do sistema.

**Detecção de anomalias clínicas** A detecção de anomalias clínicas consiste na identificação de eventos raros (anomalias) no contexto de uma população de pacientes. Exemplos de detecção de anomalias são: detecção de câncer de mama, detecção do mau funcionamento cardíaco, detecção de diabetes, etc.

## 1.2 Estado da Arte

Classificação com exemplos de uma única classe (*one-class classification*), como a própria descrição já insinua, é essencialmente classificação de padrões. O problema da classificação de padrões tem como estado da arte a técnica já bastante consolidada SVM (*Support Vector Machine*). O fato de SVMs serem o estado da arte na classificação de padrões, porém, não significa que o método será o de melhor desempenho na sua aplicação a todos os problemas. Vários métodos alcançam desempenhos similares e até mesmo, com certa freqüência, o superam nos critérios para desempenho adotados na aplicação a determinados problemas. Um grande diferencial do método SVM, em relação aos diversos métodos existentes para classificação de padrões, é seu alto grau de formalismo matemático que possibilita a criação de um classificador fundamentada em bases sólidas da teoria matemática. Métodos como redes neurais, também chegam a soluções bastante satisfatórias, porém, de forma obscura. O classificador pode ser visto como uma caixa preta. Outros métodos também convergem a soluções satisfatórias utilizando-se de heurísticas e conhecimento a-priori dos dados.

O conceito de classificação com exemplos de uma única classe está, por definição, fortemente alinhado com a tarefa de detecção de novidades. Na tarefa de detecção de novidades a fase mais penosa é, geralmente, a coleta de dados para representação da classe novidade (vale salientar que a detecção de novidades não necessariamente é classificação com exemplos de uma única classe, exemplos da classe novidade podem existir na detecção de novidades). A classe novidade pode ser representada por uma gama infinita de objetos os mais variados possíveis. Nada mais intuitivo do que gerar uma descrição dos dados disponíveis sobre objetos normais e inferir que um dado objeto de teste não é normal pois se encontra fora da superfície de descrição dos dados normais.

Três temas principais foram pesquisados com maior ênfase para o desenvolvimento deste trabalho: classificações com exemplos de uma única classe, detecção de novidades e redução de protótipos. Uma maior atenção deve ser dispensada ao problema da detecção de novidades aplicada tanto ao problema da detecção de novidades em séries temporais quanto em dados

atemporais.

Detecção de novidades aplicada ao problema da classificação de padrões é uma área relativamente madura, muitos trabalhos já foram desenvolvidos nessa área [Zhi07] [ABPF07] [FMW08] e as aplicações são as mais diversas. O termo *one-class classification* foi proposto por Moya et. al. [MKH93]. Esse conceito também é conhecido como *outlier detection*, *data description* e *concept learning*. Alguns autores tratam a detecção de novidades utilizando classificação com exemplos de uma única classe sem explicitamente referenciar o termo *one-class classification*. Esse é o caso de Markou e Singh [MS03b] [MS03a]. Markou e Singh [MS03b] redigiram uma revisão sobre a detecção de novidades com a utilização de métodos estatísticos. Abordagens não paramétricas, como o SVDD (*Support Vector Data Description*)[Tax01] e a estimativa Parzen de densidade, são citadas dentre outras abordagens. Em [MS03b], vê-se que métodos estatísticos para detecção de novidades consistem, basicamente, na modelagem da distribuição dos dados e posterior estimativa da probabilidade dos dados de teste pertencerem ou não a essa distribuição. Em [MS03a], uma nova revisão foi elaborada, porém, agora baseada em modelos de redes neurais. Em [MS03a], abordagens utilizando redes neurais MLP, RBF, SOM, ART, etc. são discutidas. Como resultado deste estudo comparativo, algumas conclusões são obtidas sobre quais técnicas são mais indicadas para quais tipos de dados. Fraquezas de abordagens conhecidas também são comentadas nesse trabalho.

O SVDD (*Support Vector Data Description*) [Tax01] é, atualmente, o método para classificação com exemplos de uma única classe mais sofisticado. Ele herda todo o formalismo matemático das máquinas de vetores de suporte. No caso mais simples (utilizando o simples produto interno ao invés de algum kernel não linear), uma hiperesfera é computada (ver figura 3.4) contendo todos os objetos. No intuito de minimizar as chances de aceitação de novidades, o volume dessa hiperesfera é diminuído. A sensibilidade a novidades pode ser controlada de maneira flexível com o ajuste do parâmetro *fracrej* que representa um limite superior no número de amostras de treinamento a serem classificadas como novidade. Assim como nas SVMs, o SVDD também suporta o uso de várias funções de kernel, o que possibilita a criação de superfícies mais flexíveis que uma simples hiperesfera. Scholkopf et. al [SWS+99], introduziram um método similar ao SVDD, o $\nu$-SVM, também para classificação com exemplos de uma única classe. Diferentemente do SVDD, o $\nu$-SVM está interessado no hiperplano ótimo, com máxima distância da origem, ou segunda classe, no espaço de características (*feature space*) tal que uma fração pré-definida das amostras de treinamento será separada dos dados normais por esse plano. As duas abordagens são equivalentes para funções de kernel não lineares, como a função de base radial (RBF) [Hof07].

Na detecção de novidades em séries temporais, poucos trabalhos (explicitamente rotulados como classificação com exemplos de uma única classe) e com resultados ainda, relativamente, inexpressivos estão disponíveis. A detecção de novidades em séries temporais é um problema bastante complexo e específico de cada caso, assim como em dados atemporais, não existe uma solução satisfatória para todos os casos. Séries temporais têm comportamentos bastantes distintos, dependendo do sistema do qual os dados foram coletados. Por esse motivo, é muito mais intuitivo o desenvolvimento de métodos específicos para detecção de novidades em séries temporais com comportamentos singulares. Isso não significa que métodos como $\nu$-SVM, SVDD e NNDD (*Nearest Neighbor Data Description*) [Tax01] não se apliquem ao problema.

Yonggui Dong et al. [YDJ06] propuseram um método para detecção de novidades em séries temporais que faz uso da similaridade entre as amostras aplicando a função de similaridade cosseno (*cosine similarity*), que consiste na medição da similaridade entre dois vetores pela observação do cosseno do ângulo entre eles. Nesse trabalho, Yonggui et.al. focaram no desenvolvimento de um novo método baseado na capacidade do sistema imunológico humano de reconhecer antígenos invasores e classificá-los como próprio (*self*) ou impróprio (*non self*). Sistemas imunológicos naturais são capazes de detectar a infecção por qualquer antígeno (substância estranha que induz uma resposta imune por causar uma produção de anticorpos e ou linfócitos). Dasgupta e Forrest [DF96], desenvolveram uma abordagem, assim como em [YDJ06], baseada no sistema imunológico humano. Nela, a série temporal é analisada e um conjunto de amostras normais, *S*, é selecionado para a representação da série. Essa abordagem não pode ser totalmente inserida na categoria de classificação com exemplos de uma única classe pois após a criação do conjunto *S*, um conjunto de detectores (exemplos da classe novidade) é gerado de forma artificial.

Outro trabalho voltado para detecção de novidades em series temporais foi desenvolvido por Oliveira et al. [OdLM06]. Oliveira et. al. criaram, a partir de um conjunto de treinamento com apenas amostras normais, intervalos de confiança robustos a fim de definir os limites inferior e superior para os valores da série temporal em um dado ponto da série. Detecção de novidades em series temporais ainda está longe de ser um problema com uma solução razoável. Na literatura existe um grande número de trabalhos publicados [MP03] [OdLM06] [YDJ06] [ONM03] [SS00], porém, esse é um problema que pode se apresentar de diversas maneiras específicas. Uma solução aplicável a todos os possíveis casos (diferentes tipos de séries temporais), parece ser utópica.

Eamonn Keogh et al. desenvolveram um esquema para detecção de novidades denominado HOTSAX [KLF05], baseado no SAX (*Symbolic Aggregate approXimation*) [LKWL07]. Esse sistema consiste em uma representação simbólica de séries temporais que permite a redução na dimensionalidade do conjunto de treinamento e torna viável a indexação do mesmo. No SAX, uma série temporal é discretizada e convertida em uma string onde cada token representa um determinado trecho da série e cada nova ocorrência de trechos similares na série deve ser associada ao mesmo token. A figura 1.3 mostra um exemplo de codificação do SAX. Keogh et al. [KLF05], propuseram um algoritmo capaz de encontrar eficientemente novidades em séries temporais pelo cálculo da similaridade entre os trechos da série, usando como entrada a série discretizada.

Uma outra abordagem para detecção de novidades em séries temporais, adotada por Salvador e Chan [SC05], se baseia no aprendizado de estados e regras da série. Em [SC05], a similaridade entre os registros da série temporal foi obtida através de um algoritmo construtivo de *clustering*. Para a descoberta de *clusters* (estados) na série foi desenvolvido um algoritmo, chamado Gecko, capaz de dividir a série em um número de *clusters* comparável ao número estimado em uma análise feita por um especialista humano. Feito isso, a lógica de transição entre os estados é construída e qualquer desvio significante dessa lógica de transição entre os estados é considerado uma novidade. Essa abordagem foi avaliada com êxito na série Space Shuttle Marota Valve [SC05], usada também em [KLF05].

Um grande número de esquemas para redução de protótipos para classificadores do tipo

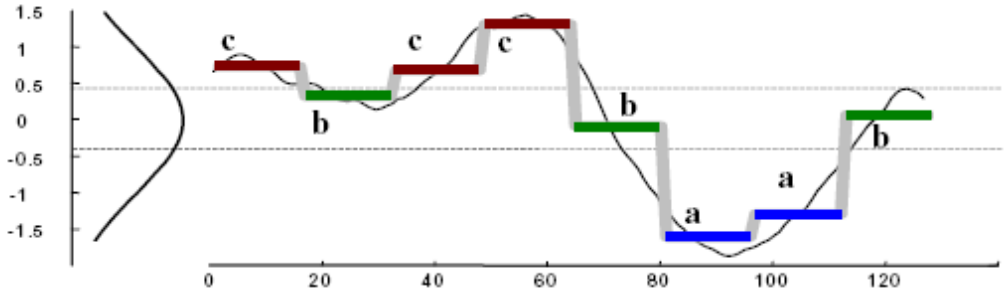


**Figura 1.3** Exemplo de série discretizada pelo método SAX

NN (*Nearest Neighbor*) está disponível na literatura. A redução de protótipos não só diminui o tempo de execução do modelo gerado como também pode diminuir o erro final associado ao classificador. Wilson e Martinez [WM00] fizeram um estudo sobre técnicas para redução de protótipos em métodos baseados na distância entre os objetos. Neste estudo foram investigadas 10 técnicas já existentes e foram introduzidas por eles 6 novas técnicas, chamadas DROP1 - DROP5 e DEL. Dentre os algoritmos que resultaram em uma redução significativa do espaço de hipóteses, os algoritmos DROP obtiveram a média de generalização mais alta nos experimentos.

Karacali e Krim [KK03], propuseram o método NNSRM (*Nearest Neighbor Structural Risk Minimization*), método baseado no conceito de SRM (*Structural Risk Minimization*), para redução de protótipos. Posteriormente, Karaçali et al. [KRS04] publicaram em 2004 uma análise comparativa entre o método NNSRM, fazendo uso de funções de kernel, e o método SVM. Os resultados desse trabalho atestaram a eficiência do princípio SRM aplicado à regra do vizinho mais próximo (*Nearest Neighbor*). Para as diferentes bases de dados utilizadas, e diferentes funções de kernel, o NNSRM utilizando kernel teve desempenhos de classificação melhores que o SVM e com tempo de execução menor devido à não necessidade da fase de otimização, que é responsável pela busca dos vetores de suporte no método SVM.

Wang e Megalooikonomou [WM08] propuseram uma técnica de redução de protótipos para análise de séries temporais. A redução de protótipos foi baseada no uso de Vector Quantization. O método proposto, chamado de PVQA (*Piecewise vector quantized approximation*), permite que uma série temporal de tamanho arbitrário $n$ seja representada por uma série temporal de tamanho $w$ onde $w \leq n$. A abordagem adotada em [WM08] é similar à adotada por [KLF05]. Primeiramente um *codebook* é gerado a partir das amostras de treinamento e a série temporal original é então codificada de acordo com as palavras contidas no *codebook* e reconstruída.

## 1.3 Objetivos

Existem diversas abordagens que podem ser utilizadas na detecção de novidades, porém, grande parte das abordagens existentes assume que novidades têm características conhecidas durante a fase de treinamento. Essas abordagens têm como característica a necessidade de objetos da classe novidade durante o treinamento do classificador. Essa necessidade consiste em uma grande limitação dessas abordagens, pois novidades, geralmente, não se encontram em um uni-

verso capaz de ser descrito. Pensando nessa limitação, esse trabalho tem como objetivo, mais geral, implementar a detecção de novidades, de forma eficiente, em dados temporais (séries temporais) e atemporais (problema da classificação) fazendo uso do conceito de classificação com exemplos de uma única classe implementando a redução de protótipos. Como objetivos específicos podemos citar:

1. Desenvolvimento de novos métodos para detecção de novidades utilizando os conceitos de classificação com exemplos de uma única classe baseado no NNDD [Tax01] e adaptando o NNSRM [KK03] para o paradigma *one-class*. Deseja-se que esses métodos sejam competitivos com métodos similares (considerados estado da arte nessa área de pesquisa). Os resultados obtidos pelos métodos propostos neste trabalho devem possibilitar a comparação desses métodos com métodos semelhantes;

2. Aplicação de conceitos da redução de protótipos [WM00] nos métodos desenvolvidos. Essa aplicação pode ser bastante benéfica a métodos para detecção de novidades com exemplos de uma única classe pelo fato da eliminação da redundância e a atenuação do efeito de ruídos no conjunto de treinamento; e

3. Realização de experimentos para a análise dos resultados alcançados e validação dos métodos propostos. Nessa fase será necessária a implementação de outros métodos para classificação com exemplos de uma única classe para a comparação de resultados. A aplicação desses métodos será analisada tanto na detecção de novidades em dados atemporais quanto em séries temporais.

Diversos métodos, que não os propostos, serão investigados nesse trabalho, dentre eles estão: SVDD e *k*NNDD [Tax01], Parzen e *k*Means. Dos métodos citados, apenas os métodos SVDD e *k*NNDD foram aplicados ao problema da detecção de novidades em séries temporais. Isso se deve ao fato de o SVDD ter um dos melhores desempenhos nos experimentos em dados atemporais. O *k*NNDD, também foi escolhido para os experimentos com séries temporais pelo fato de os métodos desenvolvidos terem relação direta com o funcionamento desse método. Para a detecção de novidades em séries temporais, a comparação de alguns resultados obtidos nesse trabalho será feita a partir de citações feitas a outros trabalhos na literatura aplicados ao mesmo problema.

## 1.4 Organização da dissertação

Esta dissertação está organizada da seguinte forma. O capítulo 2 descreve o problema da redução de protótipos. Várias características são detalhadas e discussões a respeito das conseqüências das adoções dessas características são levantadas. Ainda no capítulo 2, vários métodos para redução de protótipos são descritos abrangendo a porção considerada mais relevante da literatura atual sobre esse tema.

O capítulo 3 detalha a classificação com exemplos de uma única classe. Nesse capítulo, é descrita a técnica *kernel trick* além de vários métodos para detecção de novidades com exemplos de uma única classe. Técnicas para avaliação de desempenho desses métodos são intro-

duzidas e o pré-processamento de séries temporais visando a sua correta manipulação por esses métodos são abordados.

No capítulo 4 são introduzidos os métodos propostos para detecção de novidades nesta dissertação. É introduzido o método NNDDSRM, e suas extensões: *k*NNDDSRM e kernel *k*NNDDSRM. O desenvolvimento dos métodos é detalhado e pseudo-códigos são fornecidos, além de uma análise da complexidade do algoritmo NNDDSRM. É introduzido também um novo método para detecção de novidades em séries temporais baseado na análise de transições entre estados da série e fazendo o uso do princípio da minimização do risco estrutural a fim de reduzir o número de protótipos para a descrição dos dados considerados normais.

O capítulo 5 demonstra os resultados dos vários experimentos realizados. Um número razoável de experimentos com dados atemporais foi realizado e uma análise da influência dos parâmetros ajustáveis do método kernel *k*NNDDSRM é realizada. Resultados de experimentos para detecção de novidades em séries temporais utilizando o método kernel *k*NNDDSRM são apresentados. Resultados da aplicação do novo método baseado na transição entre os estados, para a detecção de novidades em séries temporais, são também reportados nesse capítulo.

O capítulo 6 apresenta a conclusão deste trabalho. As contribuições geradas e trabalhos futuros são expostos.

CAPÍTULO 2

# Redução de Protótipos e Minimização do Risco Estrutural

Algoritmos de aprendizado baseados em instâncias (Vizinho mais Próximo, SVM, etc.) enfrentam freqüentemente o problema da decisão de quais instâncias armazenar durante o treinamento. Armazenar uma grande quantidade de instâncias pode resultar na necessidade de um grande espaço de memória para o armazenamento e baixo desempenho na execução, assim como uma maior sensibilidade a ruídos.

Por exemplo, o algoritmo NN (*Nearest Neighbor* - Vizinho mais Próximo) original, armazena todas as instâncias de treinamento. Ele aprende muito rápido pois precisa apenas ler e armazenar o conjunto de treinamento sem a necessidade de um processamento extra, e tem uma generalização precisa para muitas aplicações. Porém, como o algoritmo do vizinho mais próximo básico armazena todas as instâncias de treinamento, dependendo do problema, pode ser necessário um grande espaço de memória para o armazenamento de instâncias. Além disso, o algoritmo deve percorrer todas as instâncias disponíveis para a classificação de um novo vetor de entradas (objeto), ou seja, ele pode se tornar lento durante a classificação. Também, como ele armazena todas as instâncias do conjunto de treinamento, instâncias que representam ruídos (instâncias com erro no vetor de entradas ou na classe de saída, ou aquelas não representativas de casos típicos) são armazenadas, o que pode prejudicar a generalização.

Nesse capítulo abordaremos também o conceito de minimização do erro estrutural, muito difundido com a popularidade de máquinas de vetores de suporte (SVM).

## 2.1 Redução de Protótipos

O conceito de redução de protótipos (*Data Reduction*) é também conhecido por vários outros termos diferentes, como, *editing*, *condensing*, *filtering*, *thinning*, etc, dependendo da finalidade da tarefa de redução de protótipos. A tarefa da classificação visa descobrir a classe de um objeto usando uma função discriminante aprendida a partir de um conjunto de instâncias presentes em um conjunto de treinamento. E esse é o que deve ser reduzido, o conjunto de treinamento. Nós tentamos representar o conjunto de treinamento completo, utilizando apenas algumas instâncias, da forma mais eficaz possível mantendo a precisão na classificação ou mesmo aumentando esta precisão. Resumindo, nós buscamos por resultados em que memória e tempo de execução sejam reduzidos enquanto a precisão do conjunto de treinamento original seja preservada ou melhore.

A figura 2.1 mostra um exemplo em que objetos no espaço bi-dimensional de duas classes distintas formam um conjunto de treinamento. Pela distribuição dos dados vê-se que apenas os

objetos mais internos já são capazes de discriminar de forma eficiente a fronteira entre as duas classes, no caso de um classificador baseado na regra do vizinho mais próximo. Como mostra a figura, apenas os objetos localizados na região de conflito entre as classes permaneceram na descrição dos dados após a redução dos protótipos.

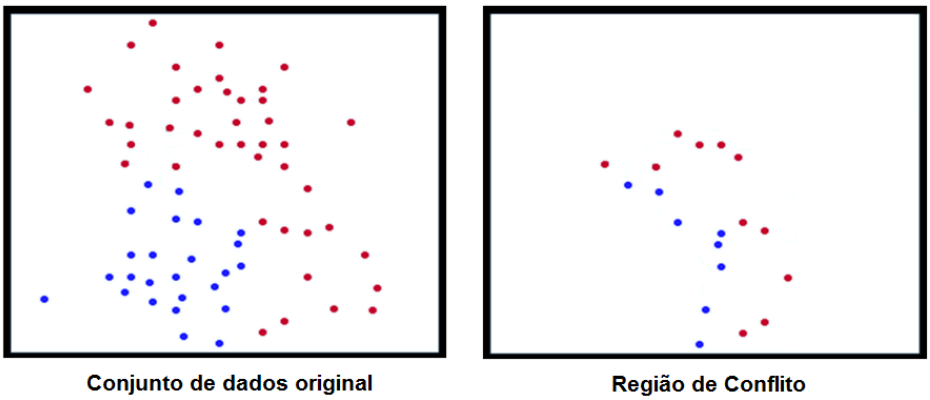


**Figura 2.1** Eliminação de objetos do conjunto de treinamento fora da região de conflito

### 2.1.1 Características de Métodos para Redução de Protótipos

Algoritmos para redução de protótipos compartilham algumas características que podem ser decisivas na escolha de uma abordagem para determinada tarefa. Os aspectos aqui descritos, de certa forma, categorizam algoritmos para redução de protótipos e devem ser levados em consideração no desenvolvimento destes. As subseções a seguir apresentam características dessa classe de algoritmos de acordo com Wilson [WM97].

#### 2.1.1.1 Representação

Uma escolha na construção de um algoritmo para redução de protótipos é a decisão entre reter um subconjunto do conjunto de treinamento original ou modificar as instâncias originais (ou até mesmo criar novas), gerando novas representações. Por exemplo, alguns algoritmos [Sal91] [WD95] usam hiperretangulos para representar uma coleção de instâncias; instâncias podem ser transformadas em regras [Dom95] [Dom96]; e protótipos podem ser usados para representar um conjunto de instâncias [Cha74], mesmo se nenhuma instância original ocorreu onde aqueles protótipos estão localizados. Estes algoritmos são chamados de *criacionais* pelo fato de poderem criar novos protótipos ou ajustarem instâncias da base de dados.

Em contrapartida, muitos algoritmos (algoritmos baseados em instâncias - *instance based algorithms*) buscam reter uma fração do conjunto de instâncias original. Um problema com o uso de instâncias em sua forma original é que pode não haver nenhuma instância localizada em um ponto que resultaria na mais precisa descrição daquele problema. Protótipos, por outro lado, podem ser construídos, de forma artificial, para que existam exatamente onde eles são necessários, caso esses locais sejam possíveis de serem determinados. Algoritmos que retêm instâncias ao invés de criá-las ou atualizá-las são chamados de algoritmos seletivos.

#### 2.1.1.2 Direção de Busca

Na busca por um subconjunto do conjunto de instâncias, a ser armazenado, existem algumas direções de busca a serem tomadas, sendo as principais a incremental, decremental e *batch*.

**Incremental** Uma busca incremental começa com um subconjunto $S$ VAZIO do conjunto de treinamento $T$, e adiciona cada instância de $T$ a $S$ se ela satisfaz alguma pré-condição. Nesse caso, a ordem de apresentação das instâncias pode ser muito importante. Em particular, as primeiras instâncias podem ter probabilidades de serem incluídas em $S$ bem diferentes caso elas fossem testadas mais tarde.
Em tais esquemas a ordem de apresentação das instâncias em $T$ ao algoritmo é geralmente aleatória. Por definição, um algoritmo incremental deve ser capaz de tratar novas instâncias à medida que elas se tornam disponíveis sem que todas elas estejam disponíveis no início da fase de aprendizagem.
Uma vantagem do esquema incremental é que se houver a disponibilidade de instâncias para adição em $S$, após o treinamento, elas podem continuar a serem adicionadas a $S$ de acordo com o mesmo critério. Outra vantagem é que algoritmos com essa característica podem ser mais rápidos na fase de aprendizado que algoritmos não incrementais, pela possibilidade do descarte de algumas instâncias enquanto outras são adicionadas.
A maior desvantagem é que algoritmos incrementais são sensíveis à ordem de apresentação das instâncias, e suas primeiras decisões são baseadas em muito pouca informação, e são suscetíveis a erro à medida que mais informação se torna disponível. Alguns algoritmos incrementais usam um pequeno número de instâncias em uma pequena fase *batch* inicial para aliviar esses problemas.
Alguns algoritmos adicionam instâncias a $S$ de forma não totalmente incremental. Eles examinam todas as instâncias disponíveis para ajudar na seleção de qual a próxima instância a adicionar. Essa abordagem pode melhorar de forma considerável o desempenho do algoritmo.

**Decremental** A busca decremental começa com $S = T$ e então procura por instâncias para remover de $S$. Novamente a ordem de apresentação das instâncias é importante, porém, diferentemente da busca incremental, todas as instâncias estão disponíveis a qualquer momento. Dessa forma, uma busca pode ser feita para a determinação de qual seria a melhor instância a se remover a cada passo do algoritmo. Exemplos de algoritmos decrementais são o SNN [Gat72], RNN [RWLI75] e o ENN [Wil72]. O RISE [Dom95] pode ser visto como decremental, porém, ao invés de apenas excluir instâncias de $S$, ele também as transforma em regras. Da mesma forma que o RISE [Dom95], o PNN (*Prototypes for Nearest Neighbor*) [Cha74] opera de forma decremental mas protótipos se fundem ao invés de serem simplesmente removidos.
Uma desvantagem na busca decremental é que ela tem um custo computacional freqüentemente maior que algoritmos incrementais. Por exemplo, na busca por um vizinho mais próximo em $T$ de uma instância $z$, $n$ cálculos devem ser feitos, sendo $n$ o número de instâncias em $T$.
De qualquer forma, se a aplicação de um algoritmo decremental puder resultar em uma

maior redução no armazenamento, então a computação extra exigida durante a aprendizagem (que é feita apenas uma vez) pode ser justificada pela redução no custo computacional no uso da aplicação após o treinamento. Melhora na precisão do algoritmo na fase de uso também pode justificar um maior custo computacional no treinamento.

**Batch** Outra forma de aplicar a redução de protótipos em um conjunto de dados é a *batch*. Dessa forma, nós decidimos se cada instância satisfaz o critério de remoção antes da remoção de alguma delas. Então, todas as instâncias que satisfazem a aquele critério são removidas de uma só vez. Por exemplo, o All k-NN rule [Tom76] opera dessa forma. Essa abordagem pode livrar o algoritmo de ter que constantemente atualizar listas de vizinhos mais próximos e outras informações quando instâncias são individualmente removidas.
Porém, também existem grandes desvantagens no processamento *batch*. Assuma que a seguinte regra seja aplicada a um conjunto de instâncias:

> Remova a instância se ela for da mesma classe que seus k vizinhos mais próximos.

Isso pode resultar no desaparecimento de clusters completos se não existirem instâncias de classes diferentes na vizinhança.

#### 2.1.1.3 Instâncias na Borda x Instâncias Centrais

Outro fator que distingue algoritmos de redução de protótipos é a escolha instâncias localizadas na borda entre as classes, instâncias localizadas no centro das distribuições das classes ou outros tipos de localização. A figura 2.1 é um bom exemplo da escolha apenas de instâncias localizadas na borda entre as classes.

A justificativa para o armazenamento de instâncias na borda é de que instâncias internas não afetam os limites de decisão tanto quanto pontos instâncias na borda, e dessa forma podem ser removidos sem que a precisão na classificação seja prejudicada.

Em contrapartida, alguns algoritmos ao invés de removerem instâncias na borda, removem pontos que são ruidosos ou que não condizem com suas vizinhanças. Dessa forma, classificadores com fronteiras entre as classes mais suaves são alcançados. Porém, esses algoritmos não removem instâncias internas, que não agregam informação ao classificador.

Pode ser necessário um grande número de instâncias para definir completamente a borda, então alguns algoritmos armazenam apenas pontos centrais no intuito de usar apenas instâncias mais típicas de uma determinada classe. Isso pode afetar de forma significativa os limites de decisão, porque estes não dependem apenas dos pontos mais típicos de uma dada classe, mas também de onde se localizam as fronteiras de outras classes.

A figura 2.2 mostra um caso em que a escolha de instâncias no centro pode levar a uma elevada taxa de erro na classificação. A linha cinza indica a fronteira entre as classes e se localiza exatamente na metade da distância entre os centros das distribuições, representados pelos objetos pretos no centro de cada distribuição dos dados das classes. Nesse caso a classe azul teve objetos erroneamente classificados pelo fato de a fronteira não levar em consideração a desproporcionalidade entre os volumes das descrições de cada classe.

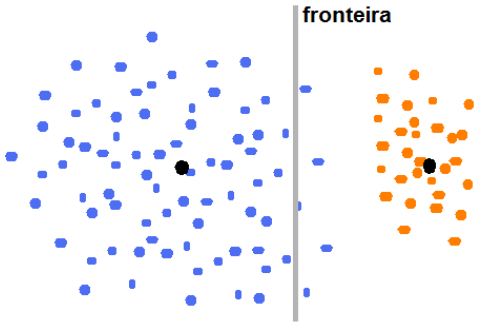


**Figura 2.2** Exemplo de fronteira entre as classes para o caso da escolha de pontos centrais para a discriminação entre as classes

#### 2.1.1.4 Função Distância

A função distância (ou seu complemento, a função de similaridade) é usada na decisão de quais instâncias se localizam mais próximas de um determinado ponto e têm uma maior importância no sistema de aprendizado.

O algoritmo do vizinho mais próximo e suas derivações, geralmente, usam variações da função de distância Euclideana, que é definida de acordo com a equação 2.1:

$$D(\overrightarrow{x}, \overrightarrow{y}) = \sqrt{\sum_{i=1}^{m}(x_i - y_1)^2} \tag{2.1}$$

onde $\overrightarrow{x}$ e $\overrightarrow{y}$ são dois vetores de entradas, $m$ é o número total de atributos (cardinalidade do vetor de entradas) e $x_i$ e $y_i$ são os valores dos atributos de entrada de índice $i$. Essa função é indicada quando todos os atributos de entrada são numéricos e seus valores variam, aproximadamente, dentro de um mesmo intervalo. Quando os atributos variam dentro de intervalos que diferem de forma significativa, pode se normalizar os vetores de entradas do conjunto de instâncias de acordo com a equação 2.2 para normalização dos valores no intervalo [0,1] e a equação 2.3 para normalização dos valores no intervalo [-1,1]:

$$Z'(t) = \frac{Z(t) - min(Z)}{max(Z) - min(Z)} \tag{2.2}$$

$$Z'(t) = \frac{Z(t) - max(Z) - min(Z)}{max(Z) - min(Z)} \tag{2.3}$$

Uma grande variedade de outras distâncias, com indicações para diferentes situações, existem. Algumas delas são: Minkowski [Bat78], Camberra, Chebychev, correlação, Manhatan, Chi-square entre outras. A figura 2.3 ilustra as equações dessas distâncias.

Algoritmos baseados em instância, normalmente lidam com valores numéricos, porém, freqüentemente não tratam valores nominais de maneira eficiente. A métrica de diferença de valores (VDM - *Value Difference Metric*) foi desenvolvida no intuito de achar valores razoáveis na distância entre atributos nominais. Wilson e Martinez [WM97] introduziram 3 novas abordagens para cálculo de distâncias envolvendo atributos nominais, numéricos ou ambos simul-

**Minkowsky**:

$$D(\vec{x},\vec{y}) = \left(\sum_{i=1}^{m}|x_i - y_i|^r\right)^{1/r}$$

**Chebychev:**

$$D(\vec{x},\vec{y}) = max_{i=1}^{m}|x_i - y_i|$$

**Camberra:**

$$D(\vec{x},\vec{y}) = \sum_{i=1}^{m}\frac{|x_i - y_i|}{|x_i + y_i|}$$

**Manhattan \ city block:**

$$D(\vec{x},\vec{y}) = \sum_{i=1}^{m}|x_i - y_i|$$

**Correlação:**

$$D(\vec{x},\vec{y}) = \frac{\Sigma_{i=1}^{m}(x_i - \mu_i)(y_i - \mu_i)}{\sqrt{\Sigma_{i=1}^{m}(x_i - \mu_i)^2 \Sigma_{i=1}^{m}(y_i - \mu_i)^2}}$$

$\mu_i$ é o valor médio do atributo *i* no conjunto treinamento.

**Chi-square:**

$$D(\vec{x},\vec{y}) = \sum_{i=1}^{m}\frac{1}{sum_i}\left(\frac{x_i}{size_x} - \frac{y_i}{size_y}\right)^2$$

$sum_i$ é o valor da soma da soma de todos os valores dos atributos *i* no conjunto de treinamento, e $size_x$ é a soma de todos os valores no vetor *x*.

**Figura 2.3** Exemplos de funções de distância entre vetores de dados

taneamente, são elas: *Heterogeneous Value Difference Metric* (HVDM), *Interpolated Value Difference Metric* (IVDM) e *Windowed Value Difference Metric* (WVDM).

#### 2.1.1.5 Votação

Outra decisão que deve ser tomada na utilização de vários algoritmos é a escolha do $k$, que é o número de vizinhos mais próximos analisados na decisão da classe de saída de um vetor de entradas.

No algoritmo básico de vizinho mais próximo, atribuir um valor maior que 1 ao $k$ diminui a sensibilidade do algoritmo ao ruído, e tende a suavizar a fronteira entre as classes. Porém, se o conjunto de treinamento foi reduzido ao ponto de haver apenas uma instância representando o que antes era um cluster, talvez $k = 1$ seja mais apropriado que $k > 1$, especialmente se as instâncias ruidosas foram removidas durante o processo de redução [WM00].

#### 2.1.1.6 Estratégias de Avaliação

Na comparação de algoritmos de redução de protótipos, existem alguns critérios que podem ser usados na medição dos pontos fortes e fracos relativos a cada algoritmo. Esses critérios serão discutidos rapidamente a seguir.

**Redução no Armazenamento** Uma das principais metas de algoritmos de redução de protótipos é a redução da necessidade de recursos para o armazenamento. Porém, dependendo

da tarefa a ser realizada, pode ser que esse critério não seja determinante na avaliação de um algoritmo.

**Aumento na Velocidade** Outra importante meta é o aumento da velocidade na classificação. Uma redução no número de instâncias armazenadas resultará numa diminuição do tempo necessário para a busca entre essas instâncias para a classificação de um vetor de entradas. Deve ser levado em consideração que representações mais complexas podem levar a uma menor velocidade na busca entre as instâncias.

**Precisão na Generalização** Um algoritmo eficiente é capaz de reduzir, de forma significativa, o conjunto de treinamento sem que haja uma redução significativa na precisão e na generalização. Em alguns casos o poder de generalização aumenta com a redução do conjunto de treinamento, como quando instâncias ruidosas são removidas ou quando as fronteiras entre as classes são suavizadas e representam de forma mais adequada os dados que as instâncias no conjunto de treinamento em sua forma original.

**Tolerância a Ruído** Algoritmos também diferem em quão bem eles trabalham na presença de ruídos. Na presença de ruídos existem dois principais problemas que podem ocorrer. O primeiro é que poucas instâncias serão removidas do conjunto de treinamento porque muitas instâncias serão necessárias para separar as instâncias das regiões onde estão localizadas as instâncias ruidosas. O segundo problema é que a precisão na generalização pode diminuir, especialmente se instâncias ruidosas são mantidas enquanto que instâncias não ruidosas são removidas. Em tais casos, o conjunto de treinamento reduzido pode ser bem menos preciso na generalização que o conjunto de treinamento original.

**Velocidade de Aprendizado** O processo de aprendizagem é executado apenas uma vez, dessa forma não é tão importante ter a fase de aprendizado rápida. Porém, se a fase de aprendizado se tornar muito lenta o algoritmo pode se tornar inviável para certas aplicações.

## 2.2 Minimização do Risco Estrutural - SRM

Minimização do erro estrutural (*Structural Risk Minimization* - SRM) [VC74] é um princípio indutivo para a seleção de modelos usado no aprendizado a partir de conjuntos de treinamento finitos. SRM fornece um *trade-off* entre a complexidade do espaço de hipóteses (conjunto de referências) e o quão bem o conjunto de referências representa o conjunto de treinamento. Como o próprio nome já diz, na minimização do erro estrutural a estrutura do modelo para classificação é modificada e não apenas parâmetros do classificador são ajustados. Um exemplo de SRM seria o corte ou adição de neurônios na camada escondida de uma rede neural. Neste caso os parâmetros não foram alterados, porém, a topologia da rede mudou pois neurônios foram extintos.

O procedimento básico utilizado pela minimização do erro estrutural é descrito a seguir:

1. Usando o conhecimento a priori do domínio dos dados, escolha uma classe de funções, como funções polinomiais de grau $n$, redes neurais tendo $n$ camadas escondidas, modelos de lógica fuzzy tendo $n$ regras, etc.

2. Divida a classe de funções em uma hierarquia de subconjuntos aninhados em ordem de complexidade crescente. Por exemplo, funções polinomiais de grau crescente.

3. Execute a minimização do erro empírico, ou seja, minimização do erro no conjunto de treinamento, (fase de treinamento). Essa etapa consiste, basicamente, na seleção de parâmetros do modelo (por exemplo, pesos de uma rede neural).

4. Selecione o modelo na série de modelos cujo a soma do risco empírico com a complexidade é mínima.

A figura 2.4 ilustra de forma simples o conceito de minimização do risco estrutural. Nela vemos três diferentes curvas: erro empírico (erro de treinamento); erro de teste; e complexidade da dimensão VC (*Vapnik-Chervonenkis Dimension*). Durante a escolha do melhor modelo, a informação do erro no teste não é conhecida e deve-se escolher o classificador com base apenas no erro empírico e na complexidade da dimensão VC. Pela figura 2.4 vemos que o melhor espaço de hipóteses é aquele que minimiza a soma da complexidade da dimensão VC com o erro empírico que, nesse caso, é o modelo $h2$. Nela, vemos ainda que uma complexidade da dimensão VC muito baixa pode causar *underfitting*, em contrapartida, uma complexidade da dimensão VC muito alta pode indicar que o modelo decorou o conjunto de treinamento, ou seja, *overfitting*.

A inequação 2.4 formaliza a idéia descrita na figura 2.4. Dado um classificador $f$ e sendo $h$ sua dimensão VC, $h$ consiste no poder de classificação de $f$ ($h$ não depende do conjunto de treinamento). Vapnik mostrou que com probabilidade $1-\eta$ a inequação 2.4 é verdadeira [Vap98].

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\frac{h(\log(2l/h)+1) - \log(\eta/4)}{l}} \tag{2.4}$$

onde $R(\alpha)$ é o verdadeiro erro para a dimensão VC $h$, $R_{emp}(\alpha)$ é o erro da dimensão $h$ para o conjunto de treinamento $\alpha$ (erro empírico) e $l$ é o número de amostras em $\alpha$.

## 2.3 Técnicas para Redução Protótipos

Como mencionado anteriormente, várias técnicas para redução de protótipos podem ser encontradas na literatura. Essa seção discute algumas técnicas tomando como base as características citadas nas seções anteriores, além de explicar o conceito de redução de protótipos recursiva.

### 2.3.1 Condensed Nearest Neighbor

Hart [Har68] criou uma das primeiras técnicas para redução de instâncias no conjunto de treinamento, chamada *Condensed Nearest Neighbor*. Seu algoritmo encontra um subconjunto $S$ do conjunto de treinamento $T$ tal que cada membro de $T$ está mais próximo de um membro de $S$ da mesma classe que de um membro de $S$ de classe diferente. Dessa maneira, o subconjunto

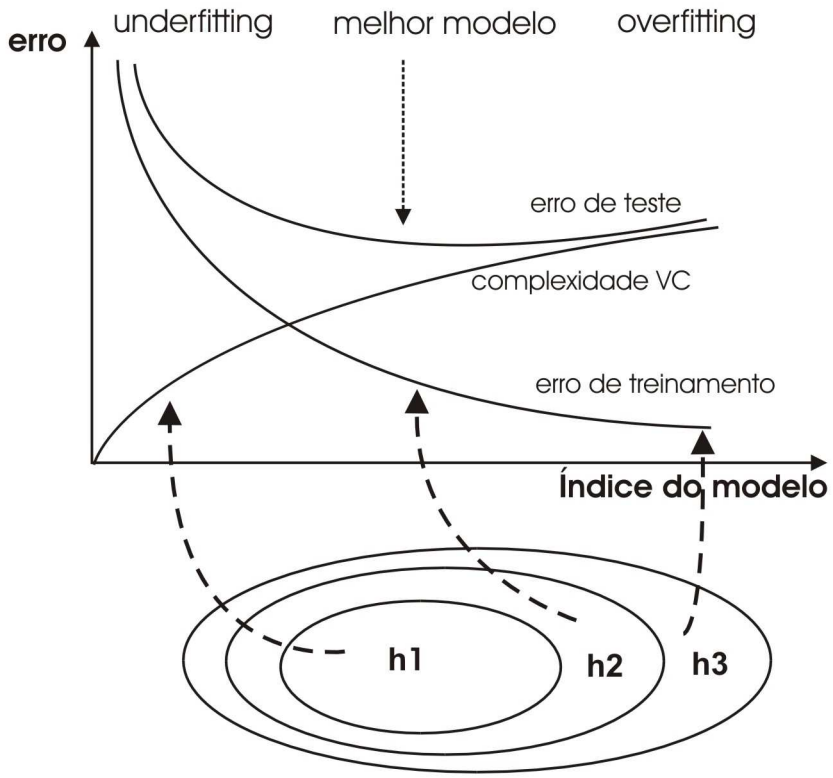


**Figura 2.4** Minimização do erro estrutural

$S$ pode ser usado para classificar cada instância em $T$ corretamente (assumindo que $T$ é consistente, ou seja, que nenhum par de instâncias em $T$ tenham vetores de entradas idênticos e pertençam a classes diferentes).

Esse algoritmo começa selecionando uma instância em $T$, de forma aleatória, e adicionando-a em $S$. Então, cada instância em $T$ é classificada usando apenas a instância em $S$. Se alguma instância em $T$ é classificada de forma incorreta, ela é adicionada a $S$, assegurando assim a sua correta classificação. Esse processo é repetido até que não hajam mais instâncias em $T$ que sejam incorretamente classificadas. Esse algoritmo garante que todas as instâncias em $T$ sejam corretamente classificadas, porém, não garante um conjunto resultante mínimo.

O algoritmo *Condensed Nearest Neighbor* tem um alto grau de sensibilidade a ruído, porque instâncias ruidosas serão, geralmente, incorretamente classificadas por seus vizinhos mais próximos e, conseqüentemente, adicionadas ao conjunto $S$. Esse fato causa pelo menos dois problemas. Primeiro, haverá uma menor redução no número de protótipos pois instâncias não ruidosas próximas às ruidosas em $S$ serão, também, adicionadas a $S$. O segundo problema é que o poder de generalização é afetado pois instâncias ruidosas estarão no conjunto $S$, porém, elas não representam a verdadeira função a ser simulada. Com a provável redução do conjunto de treinamento $T$, uma instância ruidosa em $S$ terá uma maior influência na generalização, causando mais classificações incorretas e tornando a generalização após a redução de protótipos mais pobre que a generalização utilizando-se a regra do vizinho mais próximo original.

O figura 2.5 mostra pseudo-código do funcionamento do CNN, sendo $f(p)$ o resultado da classificação de uma instância $p$ com base no conjunto $S$.

```
1. A primeira amostra é copiada, aleatoriamente, de T para S
2. para i = 0 até i = #T
        se (f(T(i)) ≠ T(i))
            adiciona T(i) a S
   fim do laço
3. se (alguma instância foi adicionada a S)  //S foi atualizado
        retorna para 2
   senão
        fim
```

**Figura 2.5** Pseudo código do *Condensed Nearest Neighbor*

Esse algoritmo claramente pode ser categorizado como seletivo, incremental e com pontos na borda.

### 2.3.2 Prototypes for Nearest Neighbor - PNN

O algoritmo PNN [Cha74] pode ser resumido da seguinte maneira: dado um conjunto de treinamento $T$, o algoritmo inicia com todos os objetos em $T$ como protótipos. Inicialmente o conjunto $A$ está vazio e o conjunto $B$ é igual a $T$. O algoritmo seleciona um objeto de forma arbitrária em $B$ e o adiciona a $A$. Após isso, os dois protótipos mais próximos em $p \subset A$ e $q \subset B$ da mesma classe são fundidos, sucessivamente, em um novo protótipo, $p*$, se a junção não degradar a generalização em $T$, onde $p*$ é o peso médio entre $p$ e $q$. Por exemplo, se $p$ e $q$ são

associados com os pesos $W_p$ e $W_q$, respectivamente, $p*$ é definido como $(W_p p + W_q q)/(W_p + W)$ e inicialmente, $p$ tem peso $(W_p + W_q)$. Quando criado, todo protótipo tem peso 1. O procedimento adotado pelo PNN é ilustrado pela figura 2.6.

1. Copia *T* para *B;*
2. Para todo $q \subset B$, $W_q$=1;
3. Seleciona um ponto em *B* e o insere em *A;*
4. MERGE = 0;
5. Enquanto #*B* > 0
   a. Encontre os protótipos *p* e *q*, de A e B, respectivamente, mais próximos;
   b. Se a classe de *p* for diferente da de *q*, então insere *q* em *A* e o remove de *B*;
   c. Senão, funde *p* de peso $W_p$, e *q* de peso $W_q$, resultando em *p**, onde $p^* = (W_p p + W_q q)/(W_p + W_q)$ . Seja o erro de classificação desse novo conjunto de protótipos $\epsilon$.
   Se $\epsilon$ aumentou, então insere *q* em *A* e remove *q* de *B*.
   Senão remove *p* e *q* de *A* e *B* e insere *p** com peso $W_p + W_q$, e MERGE++;
6. *Se MERGE = 0,* então o resultado do treinamento é *A,* e o processo termina
7. Senão copia *A* em *B* e vai para 3.

**Figura 2.6** Pseudo código do *Prototypes for Nearest Neighbor*

Esse algoritmo é um bom exemplo de algoritmo que utiliza a forma de representação criacional. Outras características dele são o uso de pontos na borda e direção de busca incremental, mesmo começando removendo elementos do conjunto *B*.

### 2.3.3 Nearest Neighbor with Structural Risk Minimization - NNSRM

O algoritmo NNSRM [KK03] é similar, em objetivo, aos algoritmos já citados anteriormente, porém, conceitualmente, o NNSRM difere das técnicas convencionais, pois ele foi criado pela aplicação explicita do princípio da minimização do erro estrutural a um classificador baseado na regra do vizinho mais próximo.

A proposta do NNSRM é encontrar o conjunto de referências (subconjunto de $T$, sendo $T$ o conjunto de treinamento) com menor cardinalidade, tal que, para todo $x_i \in T$, a função discriminante encontrada $f(x_i) = y_i$. Sendo cada objeto de $T$ da forma $(x_i, y_i)$, onde $x_i$ representa o vetor de entradas do objeto e $y_i$ representa a saída (classe) desse objeto. Busca exaustiva é claramente impraticável, pois poderia se chegar a um grande número de subconjuntos de $T$, $(2^{l_1} - 1)(2^{l_2} - 1)$ subconjuntos para um problema de apenas duas classes, testá-los no conjunto de treinamento e selecionar o de menor cardinalidade com erro de treinamento igual a zero. Ao invés disso, Karacali e Krim propuseram o seguinte algoritmo [KK03].

Considere todas as distâncias $\rho(x_i, x_j)$, tal que, $y_i = -1$ e $y_j = 1$, e seja $d_k$ um array, para

$k = 1,..,l_1 l_2$, ordenado em ordem crescente contendo essas distâncias. O algoritmo para um problema envolvendo apenas duas classes é ilustrado na figura 2.7.

1. inicializa $S = \emptyset$, k = 1;
2. enquanto $(R_{emp}(f_j) > 0)$ *faça*
   a. encontre $x_i$ e $x_j$ tal que $\rho(x_i, x_j) = d_k$ para $y_i = -1$ e $y_j = 1$
   b. se $\{i,j\} \not\subset S$, atualiza $S \leftarrow S \cup \{i,j\}$
   c. $k \leftarrow k + 1$

**Figura 2.7** Pseudo código do *Nearest Neighbor with Structural Risk Minimization* (NNSRM) para um problema com apenas 2 classes.

O procedimento proposto aumenta a inclusão de pares de objetos $\{(x_i,x_j)|y_i = -1, y_j = 1\}$ mais próximos, e se mantém atualizando o conjunto de referências, $S$, até que a classificação correta do conjunto de treinamento completo ocorra. A intuição por trás do algoritmo NNSRM é que a maioria dos erros na classificação ocorre em regiões onde objetos de classes diferentes se encontram próximos. Dado um conjunto de treinamento, essas regiões são identificadas pelos pares de objetos de classes diferentes com menores distâncias entre si, e a superfície de separação deve funcionar, primeiramente, para esses objetos situados nos limites entre as classes, ou, regiões de conflito entre as classes (ver figura 2.1).

Para a classificação com amostras de múltiplas classes, seja $m_i$ os rótulos das classes, para $i = 1..M$. Como no problema com duas classes, o classificador deve classificar todo o conjunto de treinamento de maneira correta. Seja $X_T = \{x_1^{m_1}, x_2^{m_1}, x_3^{m_1}, ..., x_{l_1}^{m_1}, x_1^{m_2}, ..., x_{l_M}^{m_M}\}$ o conjunto de treinamento, onde $l_i$ denota o número de elementos na classe $m_i$ no conjunto de treinamento, para $i = 1,..,M$. A figura 2.8 ilustra o algoritmo NNSRM para o caso de múltiplas classes.

1. inicializa $S = k^{(i,j)} = 1$, para todo par (i, j)
2. enquanto $R_{emp} > 0$ faça
   a. para todo (i, j) tal que $f_S$ classifica de forma errada um objeto de $m_i$ como um objeto de $m_j$ ou vice-versa, faça
      i. encontre os pontos $x^{mi}$ e $x^{mj}$ tal que $\rho(x^{mi}, x^{mj}) = d^{(i,j)}$
      ii. incremente $k^{(i,j)} = k^{(i,j)} + 1$
      iii. se $\{x^{mi}, x^{mj}\} \notin S$ atualiza $S = S \cup \{x^{mi}, x^{mj}\}$, senão, volta para 2.a.i

**Figura 2.8** Pseudo código do *Nearest Neighbor with Structural Risk Minimization* (NNSRM) para um problema com múltiplas classes.

O algoritmo NNSRM claramente converge, pois no pior caso todos os objetos no conjunto de treinamento são incluídos em $S$.

Em [KRS04], Karacali et. al. experimentaram a substituição da distância Euclideana utilizada no NNSRM original, pela medida de distância utilizando o kernel RBF.

$$\|z(x_i) - z(x_j)\|^2 = K(x_i, x_j) + K(x_i, x_j) - 2K(x_i, x_j) \tag{2.5}$$

onde $K$ é a função que caracteriza o *kernel*. O algoritmo em [KRS04] funciona da mesma forma que em [KK03], apenas a distância foi substituída pela função de kernel.

O NNSRM é claramente incremental e seletivo. Outra característica dele é o uso de pontos na borda.

### 2.3.4 Adaptive Recursive Partitioning Prototype Reduction Scheme - ARP_PRS

A maioria dos algoritmos para redução de protótipos disponíveis na literatura, processam os dados em sua totalidade para resultarem em um subconjunto de protótipos que serão úteis na posterior classificação baseada na regra do vizinho mais próximo. Porém, protótipos no interior de suas respectivas classes são processados por, aparentemente, nenhum motivo (no caso de pontos na borda protótipos próximos às fronteiras entre as classes são mais importantes). Esses algoritmos sofrem com um custo computacional excessivo, pelo processamento de todos os dados, que pode se tornar inaceitável no caso de conjuntos de dados volumosos.

Kim e Oommen [KO04] propuseram um mecanismo recursivo onde os dados são subdivididos recursivamente em subconjuntos menores para a filtragem de objetos internos, menos úteis ao classificador. Na seqüencia, um método para redução de protótipos (podendo ser qualquer método proposto na literatura) processa os subconjuntos menores o que resulta em subconjuntos de protótipos  um subconjunto de protótipos para cada subconjunto criado na subdivisão inicial. Os protótipos que resultam de cada subconjunto são então reunidos em um novo subconjunto e novamente processados pelo mesmo método para redução de protótipos a fim de encontrar um conjunto mais refinado de protótipos. Dessa maneira, protótipos que estão no interior do espaço de Voronoi, e não afetam à classificação, são eliminados nas chamadas consecutivas ao método para redução de protótipos. Uma conseqüência direta da eliminação de objetos redundantes nas execuções do método para redução de protótipos é a redução significativa no tempo de treinamento.

A figura 2.9 mostra um conjunto de treinamento completo e seus subconjuntos divididos aleatóriamente. Nesse caso deve se observar que a superfície de separação obtida pela execução do algoritmo para redução de protótipos no conjunto de treinamento completo (ver figura 2.9(a)) é, relativamente, similar às superficies obtidas a partir de subconjuntos com a metade do tamanho do conjunto de treinamento completo (ver figura 2.9(b) e (c)). Vale salientar que este caso é apenas um exemplo, em geral, a recursão pode ser invocada sempre que o tamanho do conjunto processado for maior que o permitido.

O algoritmo completo do ARP_PRS é ilustrado no pseudo código da figura 2.10.

A abordagem ARP_PRS é indefinida quanto às características de algoritmos para redução de protótipos citadas nesse capítulo. Estas características dependem do esquema para redução de protótipos escolhido para ser usado nessa abordagem.

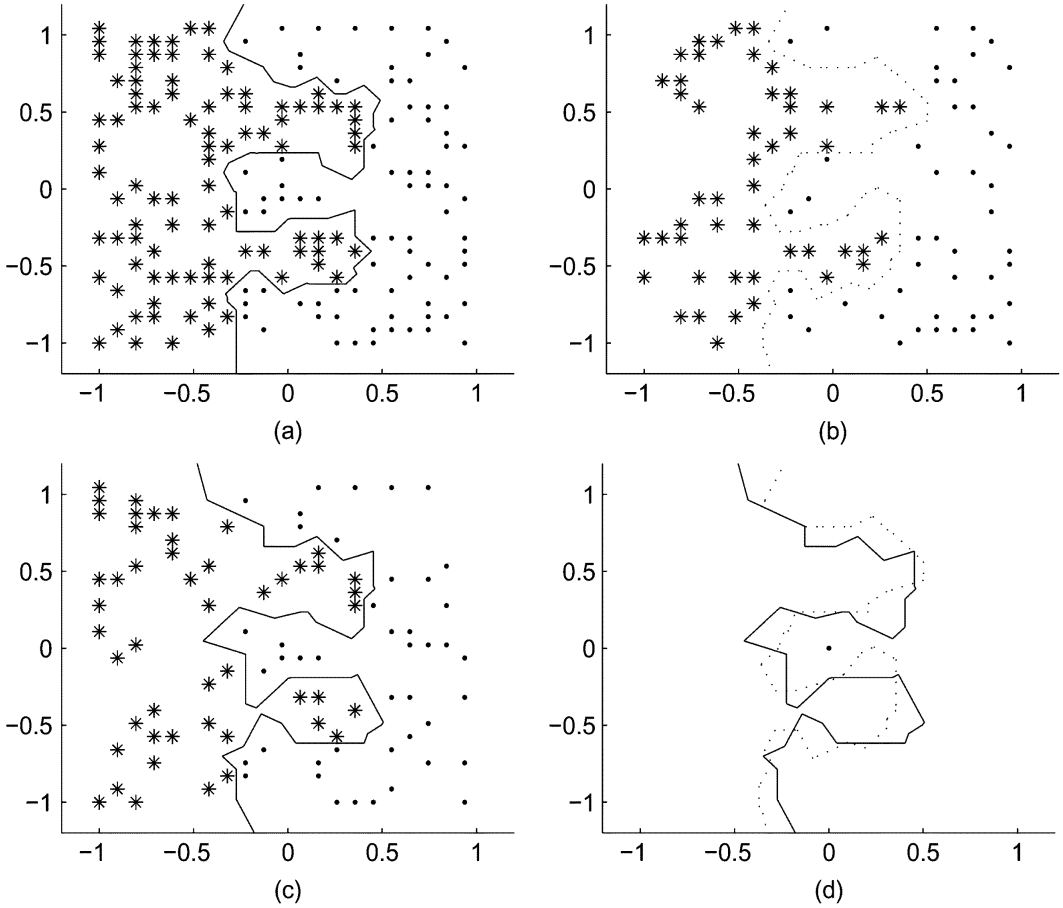


**Figura 2.9** O conjunto de treinamento completo e seus subconjuntos aleatóriamente divididos, onde os objetos são representados por "."e "*". (a) Conjunto de treinamento completo contendo 200 objetos. (b) Subconjunto 1 contendo 100 objetos. (c) Subconjunto 2 contendo 100 objetos. (d) Diferença entre as fronteiras alcançadas pelo subconjunto 1 (linha pontilhada) e 2 (linha sólida).

## 2.4 Considerações Finais

Hoje em dia, um grande volume de informação é disponibilizada em meios de comunicação como a internet, por exemplo. Classificar, entender ou comprimir essa informação é uma tarefa difícil. O mesmo problema é, também, comum em conjuntos de dados extremamente volumosos como os encontrados na mineração de dados, categorização de textos, previsão de séries temporais, etc. A redução de protótipos visa a diminuição no número de vetores de amostras enquanto, simultaneamente, assegura que a redução no conjunto de dados não afetará a superfície de decisão do conjunto de treinamento original ou afetará apenas levemente.

É importante ter em mãos, durante o uso do algoritmo de redução de protótipos, algum conhecimento a priori sobre a distribuição dos dados a serem descritos. A existência ou não de ruídos no conjunto de treinamento, a existência ou não de uma fronteira bem definida entre as classes, o uso de conjuntos de dados volumosos para a descrição do problema ou a disposição dos dados no espaço de entradas são alguns dos fatores que podem ser decisivos na

**Algoritmo ARP_PRS**
**Entrada:** Conjunto de treinamento original, $T$
**Saída:** Conjunto reduzido de protótipos, $Y_{final}$
**Método:**
Chama PRS_Recursivo($T$, $Y_{final}$, $k$, $j$)
**FIM Algoritmo ARP_PRS**
**Procedimento** PRS_Recursivo(*InSet*, *OutSet*, $k$, $j$)
**Entrada:** O subconjunto do conjunto de treinamento, *InSet*, e o parâmetro $k$, que especifica o tamanho do menor conjunto para o qual o procedimento não é invocado recursivamente. *InSet* não é subdividido recursivamente se $\#InSet \leq k$. Nesse caso é chamado um método convencional para redução de protótipos. Também, a cada nível *InSet* é subdividido em $J$ subconjuntos, onde $J$ é um parâmetro definido pelo usuário

**Se** $\#InSet \leq k$
Chama PRS(*InSet*, *OutSet*)
**Retorna** *OutSet*
**Senão**
Subdivide *InSet* em $J$ mutuamente exclusivos subconjuntos $InSet_1$ ... $InSet_j$
**Para** i ← 1 **a** $J$ **faça**
Chama PRS_Recursivo($InSet_j$, $OutSet_j$, $k$, $j$)
$TempSet \leftarrow OutSet_1 \cup OutSet_2 \ldots OutSet_j$
Chama PRS_Recursivo(*TempSet*, *OutSet*, $k$, $j$)
**FIM Procedimento PRS_Recursivo**

**Figura 2.10** Pseudo código do *Adaptive Recursive Partitioning Prototype Reduction Scheme*

escolha dentre o grande número de métodos disponíveis na literatura para a tarefa da redução de protótipos.

CAPÍTULO 3

# Classificação com Exemplos de uma Única Classe

## 3.1 Introdução

Este capítulo detalha várias técnicas para classificação com exemplos de uma única classe. A detecção de novidades (*novelty detection*), também conhecida como *anomaly detection* e *data description*, consiste em uma importante tarefa com aplicação em muitas áreas do conhecimento. Por exemplo, um sistema de análise de vídeo pode ser planejado de forma a detectar o aparecimento de objetos não pertencentes a uma determinada cena ou pode ser responsável pela detecção de falhas em um equipamento de acordo com o histórico do funcionamento deste equipamento.

Vários métodos foram propostos para a resolução de problemas de classificação com exemplos de uma única classe. Três abordagens distintas podem ser identificadas: estimativa de densidade (*density estimation*), métodos baseados em fronteiras entre as classes (*boundary methods*) e métodos baseados na reconstrução dos dados (*reconstruction methods*). Tendo por base essas abordagens; diferentes métodos foram propostos.

Existem vários aspectos importantes relativos à detecção de novidades. Alguns deles são expressos de acordo com os seguintes princípios [MS03b]:

1. *Princípio da robustez e do trade-off*: a detecção de novidades deve ser capaz de maximizar, no conjunto de teste, a exclusão de amostras da classe novidade e ao mesmo tempo minimizar a exclusão de amostras da classe normal;

2. *Princípio da minimização de parâmetros*: um método para detecção de novidades deve ter o menor número possível de parâmetros ajustáveis pelo usuário;

3. *Princípio da generalização*: o sistema deve ser capaz de generalizar sem confundir a informação generalizada com novidade (classificação de dados normais como novidades) [TD98];

4. *Princípio da independência*: a detecção de novidades deve ser independente do número de atributos e classes disponíveis. O classificador deverá mostrar desempenho razoável no contexto de desbalanceamento de dados, baixo número de amostras e ruído.

5. *Princípio da adaptabilidade*: um sistema que reconhece novidades durante o teste deve ser capaz de usar essa informação para um re-treinamento [SG01].

6. *Princípio da complexidade computacional*: boa parte das aplicações de detecção de novidades são em tempo real, dessa forma, a complexidade computacional de um mecanismo de detecção de novidades deve ser o mais baixa possível.

Os métodos para classificação com exemplos de uma única classe apresentados nesse trabalho diferem na maneira de explorar características diferentes dos dados. Um aspecto importante na utilização desses métodos é a normalização dos atributos das bases. Bases de dados não normalizadas podem gerar resultados pobres, de acordo com a escolha do kernel. Nesse capítulo nós iremos apenas discutir as características de métodos para classificação com exemplos de uma única classe; nenhum experimento será reportado. Experimentos para comparação entre os métodos serão conduzidos no capítulo de experimentos (Cap. 5).

O restante deste capítulo está estruturado da seguinte forma. A seção 3.2 explica o conceito de *kernel trick* (truque do kernel), bastante difundido e utilizado na literatura. A seção 3.3 explica o funcionamento e características dos métodos para classificação com exemplos de uma única classe utilizados nessa pesquisa, também, esclarecendo as vantagens e desvantagens dos métodos. A seção 3.3 descreve vários métodos que utilizam abordagens diferentes para a resolução de problemas de classificação com exemplos de uma única classe. A seção 3.4 descreve os métodos de avaliação usados nesse trabalho, métodos esses consolidados na avaliação da classificação com exemplos de uma única classe. A seção 3.5 explica o pré-processamento realizado nas séries temporais para o uso de classificadores desse tipo no problema da detecção de novidades em séries temporais. Finalmente, a seção 3.6 expõe considerações finais sobre métodos de classificação com exemplos de uma única classe.

## 3.2 Kernel Trick

O poder computacional limitado de classificadores lineares foi primeiramente destacado por Minsky e Papert na década de 1960 [MP66]. Em geral, aplicações complexas do mundo real requerem espaços de hipóteses mais expressivos que apenas funções lineares. Múltiplas camadas de funções lineares foram propostas como uma solução para esse problema e essa abordagem resultou no desenvolvimento das redes neurais de múltiplas camadas e algoritmos de aprendizado como o *back-propagation* para treinamento dessas redes.

Representações de kernel oferecem uma solução alternativa projetando os dados em um espaço de características n-dimensional afim de aumentar o poder computacional de classificadores lineares. Outro atrativo do uso de kernels é que algoritmos de aprendizado podem ser dissociados de conceitos específicos de sua área aplicação (algoritmos para classificação podem ser construídos de uma maneira mais geral). Esses conceitos podem ser simplesmente codificados no desenvolvimento de uma função de kernel apropriada. Dessa forma, o problema da escolha de uma arquitetura para uma rede neural pode ser substituído pela escolha de um kernel confiável para uma máquina de vetor de suporte, por exemplo.

A dificuldade da tarefa de aprendizado pode variar de acordo com a maneira com que a função a ser aprendida é representada. De preferência uma representação que se adeque ao problema específico deve ser escolhida. A equação 3.1 ilustra um exemplo onde há um mapeamento do espaço de entradas $X$ para um novo espaço , $F = \{\phi(x)|x \in X\}$.

$$x = (x_1, x_2, .., x_n) \mapsto \phi(x) = (\phi_1(x), \phi_2(x), .., \phi_n(x)) \tag{3.1}$$

Os valores do vetor usado para a representação de uma amostra de dados no hiperespaço

são, geralmente, chamados de características enquanto que no espaço original de entradas são chamados de atributos. A figura 3.1 mostra um exemplo de um mapeamento de um espaço de entradas bi-dimensional para um espaço bi-dimensional de características, onde os dados não podem ser separados por uma função linear no espaço de entradas, mas podem no espaço de características. Esse exemplo mostra como tais mapeamentos podem ser feitos no espaço n-dimensional onde uma separação linear se torna possível.

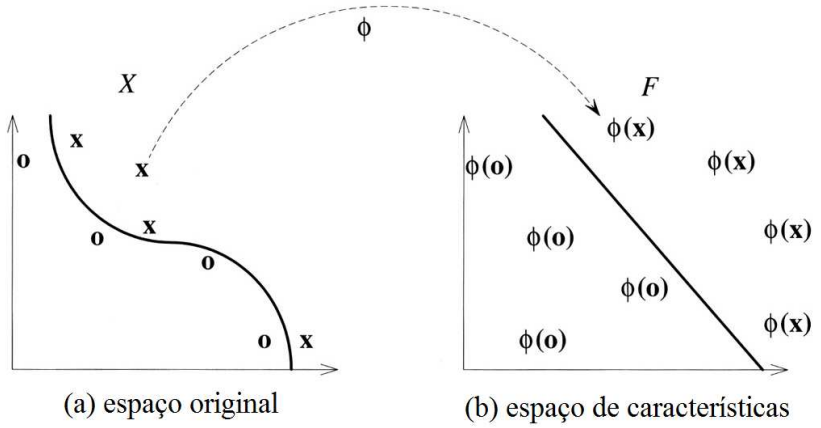


**Figura 3.1** O mapeamento para um espaço de características pode simplificar a tarefa de classificação

Para a criação de uma função de kernel deve-se primeiramente determinar quais propriedades são necessárias para assegurar que essa função é um kernel para algum espaço de características. Tais propriedades são satisfeitas se a função criada satisfaz o teorema de Mercer [Mer09]. Exemplos de funções de kernel são:

1. Linear : $K(x_i, x_j) = x_i^T x_j$
2. Polinomial : $K(x_i, x_j) = (x_i \bullet x_j + 1)^d$
3. Sigmóide : $K(x_i, x_j) = \tanh(kx_i \bullet x_j - \delta)$
4. Função de Base Radial (RBF) : $K(x_i, x_j) = e^{\left(\frac{-\|x_i - x_j\|}{\sigma^2}\right)}$

## 3.3 Métodos para Classificação com Exemplos de Única Classe

Existem basicamente três diferentes abordagens para resolução do problema de classificação, e conseqüentemente, também, da classificação com amostras de uma única classe, são elas: métodos baseados em estimativa de densidade (*density estimation*); métodos baseados em fronteiras entre as classes (*boundary methods*); e métodos baseados na reconstrução dos dados (*reconstruction methods*).

O método mais simples para a obtenção de um classificador para o problema da classificação com exemplos de uma única classe é estimar a densidade e atribuir um limiar (*threshold*)

a essa densidade. Várias distribuições em relação aos dados podem ser assumidas (como distribuição Gaussiana ou distribuição de Poison). Infelizmente, estimativa de densidade requer um grande número de amostras, para a representação do classificador. Esse grande número de amostras pode resultar em problemas de desempenho na classificação, pelo alto custo computacional das computações necessárias. No caso onde o conjunto de treinamento tende ao infinito, representando fielmente o universo de amostras do problema, classificadores baseados na estimativa de densidade teoricamente obtêm os melhores resultados.

Estimar completamente a densidade dos dados para o problema da classificação com exemplos de uma única classe pode ser uma tarefa irreal, impossível de ser realizada. Porém, apenas dados no limite da classe normal são necessários. Em métodos baseados em fronteiras entre as classes, portanto, apenas uma descrição fechada da classe normal é otimizada. Embora o volume da descrição nem sempre seja minimizado em métodos baseados em fronteiras entre as classes, a maioria dos métodos tem uma forte tendência a uma solução que obtenha um volume mínimo. A minimização do volume depende da adequação do método aos dados. Pelo fato de métodos baseados em fronteiras entre as classes se basearem fortemente em distâncias entre objetos, eles tendem a serem sensíveis à escala das características (atributos das amostras de dados no hiperespaço). Por outro lado, o número de objetos necessários para a descrição dos dados é menor que métodos baseados em estimativa de densidade. Exemplos de métodos baseados em fronteiras entre as classes são: SVDD, *one-class* SVM e NNDD.

Usando o conhecimento a priori dos dados e fazendo pressupostos em relação ao processo de reconstrução dos dados, um modelo é escolhido e adequado aos dados segundo a abordagem adotada pelos métodos baseados em reconstrução. Novos objetos podem agora ser ajustados, ou até mesmo criados, de acordo com o estado do modelo que está sendo gerado. Nesses métodos, nós assumimos que uma representação mais compacta da classe normal pode ser obtida de forma que nessa representação a contribuição do ruído é minimizada. Exemplos de métodos baseados em reconstrução são: k-means *clustering*, LVQ (*Learning Vector Quantization*), PCA (*Principal Component Analysis*) e redes SOM (*Self Organizing Maps*). Esses métodos diferem na definição dos protótipos (ou sub-espaços) e na rotina de otimização.

Nessa seção descrevemos, detalhadamente, os métodos, de classificação com exemplos de uma única classe utilizados nos experimentos para a comparação dos resultados obtidos com o nosso método proposto, kernel *k*NNDDSRM. São eles: estimador de Parzen data description (estimativa de densidade); *k*NNDD, SVDD e *one-class* SVM (métodos baseados em fronteiras entre as classes); e k-means data description (método baseado em reconstrução dos dados). Todos os métodos detalhados nessa seção, exceto o *one-class* SVM, foram implementados e os resultados foram comparados aos resultados de suas implementações no toolbox DDTools [Tax07] no intuito de verificar a corretude da implementação. A implementação se fez necessária para uma maior facilidade na análise dos resultados, como geração de gráficos, por exemplo. A linguagem de programação escolhida para essa implementação foi a linguagem Java.

### 3.3.1 *k*NNDD

O desenvolvimento do NNDD (*Nearest Neighbor Data Description*) [Tax01] foi inspirado no algoritmo do vizinho mais próximo (*NN*) [DH73], que é um método para classificação que

toma decisões baseado na distância entre o objeto a ser testado e seu vizinho mais próximo no conjunto de treinamento. O algoritmo *NN* original foi desenvolvido para classificação com múltiplas classes (*multi-class classification*). Em contraste, o *NNDD* é um método de classificação para classificação com exemplos de uma única classe.

A fase de treinamento do *NNDD* consiste apenas no armazenamento de todas as amostras do conjunto de treinamento na memória. Na fase de teste, o *NNDD* tem que fazer uma busca exaustiva considerando todas as amostras no conjunto de treinamento para encontrar o vizinho mais próximo do objeto a ser classificado. Essa é uma desvantagem de algoritmos baseados no método do vizinho mais próximo.

No NNDD, um objeto $p$ é classificado como normal (também referido como *target*) se a distância entre seu vizinho mais próximo no conjunto de treinamento, $tr$, for menor do que a distância de $tr$ a seu vizinho mais próximo no conjunto de treinamento. Caso contrário, $p$ é classificado como novidade. A equação 3.2 explica a maneira como é feita a decisão no NNDD. Na equação 3.2, *dist* é a distância Euclideana entre dois objetos.

$$f(x) = \begin{cases} \frac{\|dist(x,NN_{tr}(x))\|}{\|dist(NN_{tr}(x),NN_{tr}(NN_{tr}(x)))\|} \leq 1, & x \text{ é normal} \\ \text{caso contrário ,} & x \text{ é novidade} \end{cases} \tag{3.2}$$

A Figura 3.2 ilustra um exemplo de classificação de uma amostra $p$ usando o NNDD. Nesse exemplo $p$ é classificada como novidade porque $\frac{d_1}{d_2} > 1$, onde $d_1 = \|dist(x,NN_{tr}(x))\|$ e $d_2 = \|dist(NN_{tr}(x),NN_{tr}(NN_{tr}(x)))\|$ na eq. 3.2.

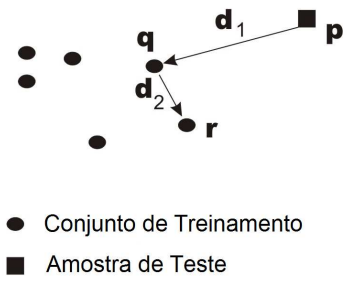


**Figura 3.2** NNDD: Exemplo de tomada de decisão.

O método $k$NNDD é similar ao NNDD original. A única diferença é que, nesse caso, a classificação não considera apenas seu vizinho mais próximo e sim todos os $k$ vizinhos mais próximos. No $k$NNDD, a equação 3.2 é calculada para cada um dos $k$ vizinhos mais próximos. O resultado, novidade ou normal, com maior ocorrência será o resultado final. Figura 3.3 mostra um exemplo bi-dimensional de classificação no $k$NNDD com o parâmetro $k = 3$; nesse exemplo a amostra de teste é classificada como novidade.

Na figura 3.3 as distâncias entre cada um dos três vizinhos mais próximos ($tr_1$, $tr_2$ e $tr_3$) à amostra de teste ($te$) são calculadas; essas distâncias são chamadas de $d_{11}$, $d_{12}$ e $d_{13}$, respectivamente. $d_{14}$, $d_{25}$ e $d_{35}$ são as distâncias de $tr_1$, $tr_2$ e $tr_3$ aos seus vizinhos mais próximos,

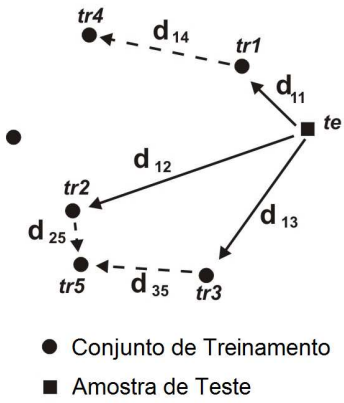


**Figura 3.3** *k*NNDD: Exemplo de tomada de decisão.

respectivamente. O NNDD original é então calculado para a amostra de teste e para $tr_1$, $tr_2$ e $tr_3$. Na figura 3.3 nós temos $d_{11} < d_{14}$, $d_{12} < d_{25}$ e $d_{13} < d_{35}$. Dessa maneira, *te* é considerada novidade pois dois de seus vizinhos mais próximos ($tr_2$ e $tr_3$) se localizam mais distantes de *te* do que de seus respectivos vizinhos mais próximos no conjunto de treinamento.

### 3.3.2 SVDD - *Support Vector Data Description*

Na técnica SVDD [Tax01], dado um conjunto de dados com N objetos $x_i, i = 1,...,N$ tenta-se encontrar a esfera de menor volume contendo todos (ou quase todos) os objetos. Quando um ou poucos objetos remotos (muito distantes) estão no conjunto de dados, uma grande esfera é obtida e os dados não serão tão bem representados. Para transpor essa limitação do modelo é permitido que algumas amostras de treinamento se localizem fora da esfera introduzindo o conceito de variáveis de folga, analogamente a máquinas de vetor de suporte para classificação com exemplos de múltiplas classes [Vap95].

A partir da equação 3.3, que representa uma esfera descrita com centro $a$ e raio R, nós minimizamos o raio onde a variável $C$ consiste no trade-off entre simplicidade (ou volume da esfera) e o número de erros (número de objetos normais do conjunto de treinamento rejeitados, ou seja, localizados fora da esfera). A figura 3.4 ilustra a hipersfera gerada pelo SVDD no caso de um kernel linear. Assim como o SVM original [Vap95], o SVDD também faz uso de variáveis de folga $\xi$. A cada objeto durante o treinamento é associada uma variável de folga, sendo que se $\xi_i > 0$, significa que o objeto $i$ está localizado fora da descrição gerada pelo classificador.

$$F(R,a,\xi) = R^2 + C\sum_i \xi_i \tag{3.3}$$

A equação 3.3 tem que ser minimizada com as seguintes restrições:

$$(x_i - a)^T (x_i - a) \leq R^2 + \xi_i \qquad \forall_i, \xi_i \geq 0 \tag{3.4}$$

Incorporando-se essas limitações na equação 3.3 construímos a equação de Lagrange:

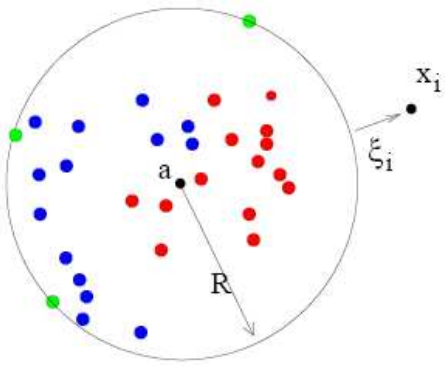


**Figura 3.4** Hiperesfera contendo os dados normais, descrita pelo centro $a$ e pelo raio $R$. Três objetos se encontram no limite da descrição; esses objetos são chamados de vetores de suporte. Um objeto $x_i$ está fora da descrição pois tem $\xi_i > 0$.

$$L(R,a,\alpha,\xi) = R^2 + C\sum_i \xi_i - \sum_i \alpha_i\{R^2 + \xi_i - (x_i^2 - 2\alpha x_i + \alpha^2)\} - \sum_i \gamma_i\xi_i \tag{3.5}$$

Com os multiplicadores de Lagrange $\alpha_i \geq 0$ e $\gamma_i \geq 0$. Igualando as derivadas parciais a zero, novas restrições são obtidas.

$$\sum_i \alpha_i = 1, \qquad a = \frac{\sum_i \alpha_i x_i}{\sum_i \alpha_i} = \sum_i \alpha_i x_i, \qquad C - \alpha_i - \gamma_i = 0 \quad \forall_i \tag{3.6}$$

Sendo $\alpha_i \geq 0$ e $\gamma_i \geq 0$ nós podemos remover as variáveis da terceira equação e usar a restrição $0 \geq \alpha_i \geq C, \forall_i$.

Reescrevendo a equação 3.5 com as restrições da equação 3.6 e maximizando com respeito a $\alpha_i$:

$$L = \sum_i \alpha_i(x_i \bullet x_i) - \sum_{i,j} \alpha_i\alpha_i(x_i \bullet x_j) \tag{3.7}$$

A segunda equação de 3.6 mostra que o centro da esfera é dado como uma combinação linear das amostras com pesos $\alpha_i$ que são obtidos pela otimização da equação 3.7. Apenas para um pequeno conjunto de objetos a condição da equação 3.4 é satisfeita, são os objetos que se encontram exatamente no limite da esfera. Esses objetos são chamados de vetores de suporte. Apenas esses objetos são necessários na descrição da esfera. O raio $R$ da esfera pode ser obtido calculando-se a distância do centro da esfera a um vetor de suporte com $\alpha_i > 0$ e $\alpha_i < C$. Objetos com $\alpha_i = C$ atingiram o limite da esfera e são considerados novidade.

Para determinar se um objeto $z$, de teste, se encontra dentro da esfera, a distância para o centro da esfera deve ser calculada. Um objeto de teste é aceito quando sua distância até o centro da esfera é menor que o raio, ou seja, $(x_i - a)^T(x_i - a) \leq R^2$. Reescrevendo essa equação em termos de vetores de suporte, objetos são aceitos quando:

$$(z \bullet z) - 2\sum_i \alpha_i (z \bullet x_i) - \sum_{i,j} \alpha_i \alpha_i (x_i \bullet x_j) \leq R^2 \tag{3.8}$$

O SVDD apresentado até agora apenas computa uma esfera ao redor dos dados no espaço original de entradas. Geralmente os dados não estão distribuídos de forma esférica, mesmo se não considerarmos os objetos mais distantes. Em geral não se pode esperar uma descrição similar a uma esfera. Como o problema pode ser resolvido completamente em termos de produtos internos entre vetores (equações 3.7 e 3.8), o método pode se tornar mais flexível, analogamente às SVMs [Vap95]. Produtos internos de objetos podem ser substituídos por funções de kernel, $K(x_i, x_j)$, quando essas funções satisfazem o teorema de Mercer. Isto implicitamente mapeia objetos em algum hiperespaço. Quando um hiperespaço é escolhido, uma melhor, e mais justa, descrição pode ser obtida. Não é necessário nenhum mapeamento explícito. Dessa forma, todos os produtos internos podem ser substituídos por uma função de kernel confiável e o problema de otimização é agora dado por :

$$L = \sum_i \alpha_i K(x_i, x_i) - \sum_{i,j} \alpha_i \alpha_i K(x_i, x_j) \tag{3.9}$$

A minimização desse erro com as *constraints*, limitações, é um problema já bastante conhecido e chamado de problema da programação quadrática. Algoritmos padrões de otimização já existem, eles são capazes de encontrar valores ótimos para os multiplicadores de Lagrange $\alpha_i$.

Com a possibilidade de se calcular uma fórmula que expresse o centro, $a$, da hiperesfera, nós podemos testar se um novo objeto $z$ é aceito pela descrição. Para isso, a distância do objeto $z$ ao centro, $a$, da hiperesfera, deve ser calculada. Um objeto de teste, $z$, é aceito, ou seja, é considerado normal, quando sua distância ao centro da hiperesfera é menor ou igual ao raio da hiperesfera:

$$\|z-a\|^2 = K(z,z) - 2\sum_i \alpha_i K(z, x_i) + \sum_{i,j} \alpha_i \alpha_i K(x_i, x_j) \leq R^2 \tag{3.10}$$

Por definição, $R^2$ é a distância quadrática do centro da hiperesfera a um de seus vetores de suporte na superfície (limite) da descrição.

$$R^2 = K(x_k, x_k) - 2\sum_i \alpha_i K(x_k, x_i) + \sum_{i,j} \alpha_i \alpha_i K(x_i, x_j) \tag{3.11}$$

para qualquer $x_k$ pertencente ao conjunto de vetores de suporte tal que $0 < \alpha_i < C$, ou seja, se encontrem no limite da descrição.

Nesse momento a função discriminante gerada pelo SVDD pode ser vista como:

$$f(z) = \begin{cases} \text{z é normal} & se \quad \|z-a\|^2 \leq R^2 \\ \text{z é novidade} & se \quad \|z-a\|^2 > R^2 \end{cases} \tag{3.12}$$

### 3.3.3 One-class SVM

Máquinas de vetores de suporte (SVMs - *Support Vector Machines*) consistem em um poderoso método de classificação baseado nos princípios da minimização do risco estrutural (SRM - *Structural Risk Minimization*). SVMs estão entre os mais sofisticados métodos supervisionados e não paramétricos existentes. Baseados nas SVMs, Schölkopf et. al. [SPST$^+$01] propuseram o método *one-class* SVM para classificação com exemplos de uma única classe. Exemplos de aplicação desse método são a classificação de documentos [MY01] e a recuperação de imagens [CZH].

Após o mapeamento, via kernel, dos objetos para uma espaço de características, o método *one-class* SVM trata a origem como o único membro da segunda classe. Como já explicado anteriormente, diferentemente do SVDD (que busca pela hipersfera de menor volume), o método *one-class* SVM, ou $\nu$-SVM, está interessado no hiperplano ótimo, com máxima distância da origem, no espaço de características, (*feature space*) tal que uma fração pré-definida das amostras de treinamento será separada dos dados normais por esse plano. A meta é maximizar a margem de separação da origem. Da mesma forma que em SVM para múltiplas classes, variáveis de folga, denotadas por $\vec{\xi}_i$ , são associadas a cada amostra de dados. Essa associação possibilita que algumas das amostras de treinamento se localizem na face do hiperplano oposta à face que representa sua verdadeira classe (ou seja, sejam classificadas como novidade) quando uma menor distância à origem é escolhida. Uma amostra de treinamento é chamada de vetor de suporte quando ela é incorretamente classificada como novidade durante o treinamento ou se localiza na descrição do hiperplano. As duas abordagens (SVDD e *one-class* SVM) são equivalentes para funções de kernel não lineares, como a função de base radial (RBF) [Hof07].

No intuito de separar os dados da origem com a margem máxima, deve-se resolver o seguinte problema quadrático:

$$\vec{min}\frac{1}{2}\|\omega\|^2 - \rho + \frac{1}{\nu\ell}\sum_{i=1}^{l}\vec{\xi}_i \tag{3.13}$$

onde $\omega$ é o vetor normal ao hiperplano de separação, $\ell$ é o número de amostras de treinamento e $\rho$ é o *offset*, sujeito a

$$(\omega \bullet \Phi(x_i)) \geq \rho - \xi_i \quad i = 1,2,...,\ell \quad \xi_i \geq 0 \tag{3.14}$$

Se $\omega$ e $\rho$ resolvem este problema, então nós encontramos uma função $f(x) = sign\,((\omega \bullet \Phi(x))) - \rho)$ tal que se $f(x) > 0$, o objeto $x$ é classificado como normal. Senão $x$ é classificado como novidade.

Se $\rho > 0$, então o parâmetro $\upsilon \in (0,1)$ é o limite superior na fração de novidades no treinamento (erro de treinamento) e, também, um limite inferior na fração de vetores de suporte. O dual desse problema é:

$$\vec{min}_\alpha \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j k(\Phi(x_i),\Phi(x_j)) \tag{3.15}$$

sujeito a

$$0 \leq \alpha_i \leq \frac{1}{\nu\ell} \quad and \quad \sum_i \alpha_i = 1 \tag{3.16}$$

e agora a função discriminante é

$$f(x) = sign\left(\sum_{i=1}^{\ell} \alpha_i k(\Phi(x_i), \Phi(z)) - \rho\right) \tag{3.17}$$

e $\rho$ pode ser calculado de acordo com

$$\rho = \sum_{i=1}^{l}\sum_{j=1}^{l} \alpha_i \alpha_j k(\Phi(x_i), \Phi(x_j)) \tag{3.18}$$

onde $0 \leq \alpha_i, \alpha_j \leq \frac{1}{\nu\ell}$.

Em nossos experimentos foi usada a biblioteca LibSVM versão 2.84. A LibSVM está disponível em http://www.csie.ntu.edu.tw/ cjlin/libsvm/.

### 3.3.4 kMeans Data Description

O *k*-means [Mac67] é um dos algoritmos não supervisionados mais simples para a resolução do problema da descoberta de agrupamentos, (*clustering*). O procedimento consiste de uma maneira simples e fácil de agrupar um dado conjunto de amostras através de um certo número de clusters (*k* clusters) fixados a priori. A idéia principal é definir *k* centróides, um para cada cluster. Esses centróides devem ser inicializados em pontos estratégicos pelo fato de que diferentes localizações causam diferentes resultados. A melhor escolha é inseri-los o mais distante possível uns dos outros. O próximo passo, no caso de classificação para várias classes, é tomar cada ponto pertencente ao conjunto de treinamento e associá-lo ao centróide mais próximo. Nesse momento é necessário re-calcular os *k* novos centróides como centros de gravidade dos clusters resultantes do passo anterior. Após o re-calculo dos *k* centróides, uma nova associação no conjunto de treinamento deve ser feita levando em conta os novos centróides. Dessa maneira gera-se um laço. Como resultado desse laço, nós podemos afirmar que os *k* centróides mudam suas localizações passo a passo até que não mais ocorram mudanças. Em outras palavras, os centróides se estabilizam. Finalmente, o algoritmo visa minimizar uma função objetivo, no nosso caso a MSE (*Mean Square Error*). O kMeans data description (kMeans_dd) [Tax01] funciona de forma similar ao kMeans para múltiplas classes, exceto pelo fato de na fase de teste um objeto poder ser associado a qualquer um dos centróides. O que discrimina um objeto normal de uma novidade é o fato de a distância de um objeto $z$, de teste, ao seu centróide mais próximo ser ou não abaixo de de um limiar, $\theta$, de aceitação.

O pseudo-código a seguir ilustra a fase de treinamento do *k*means data description:

1. Carrega o conjunto de treinamento na memória;

2. Escolhe, aleatóriamente, *k* amostras no conjunto de treinamento para serem os centróides do classificador;

3. Cria $k$ clusters onde cada elemento de um dado cluster, $C$, tem como centróide mais próximo o centróide responsável pela representação desse cluster, $C$;

4. $\forall a | a \subset Conj.Treinamento$ { // err = MSE no treinamento.
   err += distancia_euclidiana_quadratica(a, centroide(a));
   }

5. olderr = 10 * err;

6. errTol = 1e-5;

7. ENQUANTO ((olderr - err) > (errTol * err))
   ▶ Calcula-se os novos centróides (que serão o centro de gravidade de cada cluster)
   ▶ Recalculam-se as amostras pertencentes a cada novo cluster, de acordo com a proximidade com cada centróide.
   ▶ olderr = err;
   ▶ err = 0;
   ▶ $\forall a | a \subset Conj.Treinamento$ {
   err += distancia_euclidiana_quadratica(a, centroide(a));
   }
   FIM ENQUANTO

No pseudo-código acima, a função $centroide(amostra)$ retorna o centróide do cluster dessa amostra passada como parâmetro.

O teste de uma amostra é feito de acordo com a equação 3.19, onde são calculadas todas as distâncias entre os centróides, $c_i$, e a amostra, $z$, de teste, se nenhuma dessas distâncias for menor ou igual a um limiar, $\theta$, pré-definido, a amostra $z$ é considerada novidade.

$$f(z) = \begin{cases} \text{z é normal} & se \quad \sqrt{min_i(z-c_i)^2} \leq \theta \\ \text{z é novidade} & se \quad \sqrt{min_i(z-c_i)^2} > \theta \end{cases} \tag{3.19}$$

### 3.3.5 Parzen Data Description

Um importante problema no campo de reconhecimento de padrões é a estimativa da função de densidade de probabilidade $F(x)$ de um número de amostras aleatóriamente selecionadas $x_1, x_2, .., x_n$. Se pouco conhecimento *a priori* em relação a $F(x)$ for disponível, um estimador não paramétrico de $F(x)$ pode ser interessante. A estimativa Parzen consiste em uma técnica não paramétrica para estimativa de densidade. Em estatística, estimar a densidade é construir uma estimativa baseada em dados observados, com base em uma função de probabilidade de densidade não observável. Sendo $\omega_1$ e $\omega_2$ duas classes, suas probabilidades a priori $P(\omega_1)$ e $P(\omega_2)$ são conhecidas. Sendo $N$ o número total de padrões de treinamento e $N_1$ e $N_2$ os padrões de treinamento de cada classe, tem-se as seguintes estimativas para as probabilidades a-priori:

$$P(\omega_1) = \frac{N_1}{N} \qquad P(\omega_2) = \frac{N_2}{N} \tag{3.20}$$

Para utilização da regra de *Bayes* outros valores estatísticos também devem ser conhecidos. São eles as funções de densidade de probabilidade (*probability density function* - pdf), $P(x|\omega_i), i = 1,2$, que descrevem a distribuição dos vetores de características de cada classe. A estimativa de todos esses valores pode ser feita fazendo uso dos dados de treinamento. Com esses dados em mãos pode-se obter probabilidades condicionais relativas a essas classes aplicando a regra de *Bayes*:

$$P(x|\omega_i) = \frac{p(\omega_i|x) \cdot P(\omega_i)}{p(x)} \tag{3.21}$$

onde $p(x)$ é a *pdf* de x, e para cada $x$ tem-se:

$$p(x) = \sum_{i=1}^{2} p(x|\omega_i) \cdot P(\omega_i) \tag{3.22}$$

Assim, a regra de classificação de *Bayes* pode ser descrita como:

$$f(x) = \begin{cases} \text{x} \in \omega_1, & se \quad P(\omega_1|x) \geq P(\omega_2|x) \\ \text{x} \in \omega_2, & se \quad P(\omega_2|x) > P(\omega_1|x) \end{cases} \tag{3.23}$$

O algoritmo de classificação *Parzen Data Description* não necessita de uma fase de treinamento; porém, a combinação entre todas as amostras do treinamento pode tornar a fase de teste relativamente lenta.

O uso da função discriminante do *Parzen Data Description*, para uma mostra $z$ de teste, é feito de acordo com a equação 3.24:

$$f(z) = \begin{cases} \text{z é normal} & se \quad \sum_{i=1}^{n} \exp(-(z-x_i)^T h^{-2}(z-x_i)) \geq \theta \\ \text{z é novidade} & se \quad \sum_{i=1}^{n} \exp(-(z-x_i)^T h^{-2}(z-x_i)) < \theta \end{cases} \tag{3.24}$$

Na equação 3.24, $h$ é a largura do kernel que nós otimizamos utilizando o toolbox para Matlab DD_tools [Tax07], de acordo com [Kraaijveld and Duin, 1991], antes do uso do classificador.

### 3.3.6 Aprendizado de regras e estados para detecção de novidades em séries temporais

Salvador e Chan propuseram em [SC05] uma abordagem para aprendizado de estados e regras para a detecção de novidades em séries temporais. Eles partiram do pré-suposto de que a operação normal de um dispositivo pode ser caracterizada em diferentes estados temporais. Para identificar esses estados foi introduzido o algoritmo para segmentação Gecko que pode determinar um número razoável de segmentos usando o método $L$ (também proposto em [SC05]). Nessa etapa, usasse então o algoritmo para classificação RIPPER para a descrição desses estados em regras lógicas. Finalmente, um automaton de estados finitos é criado a partir da lógica de transição entre os estados.

A figura 3.5 ilustra o sistema criado em [SC05]. Nela uma série temporal é dada como entrada e uma fase de segmentação é responsável por identificar três estados diferentes. O classificador RIPPER é então usado para gerar as regras que irão formar o automaton de estados finitos.

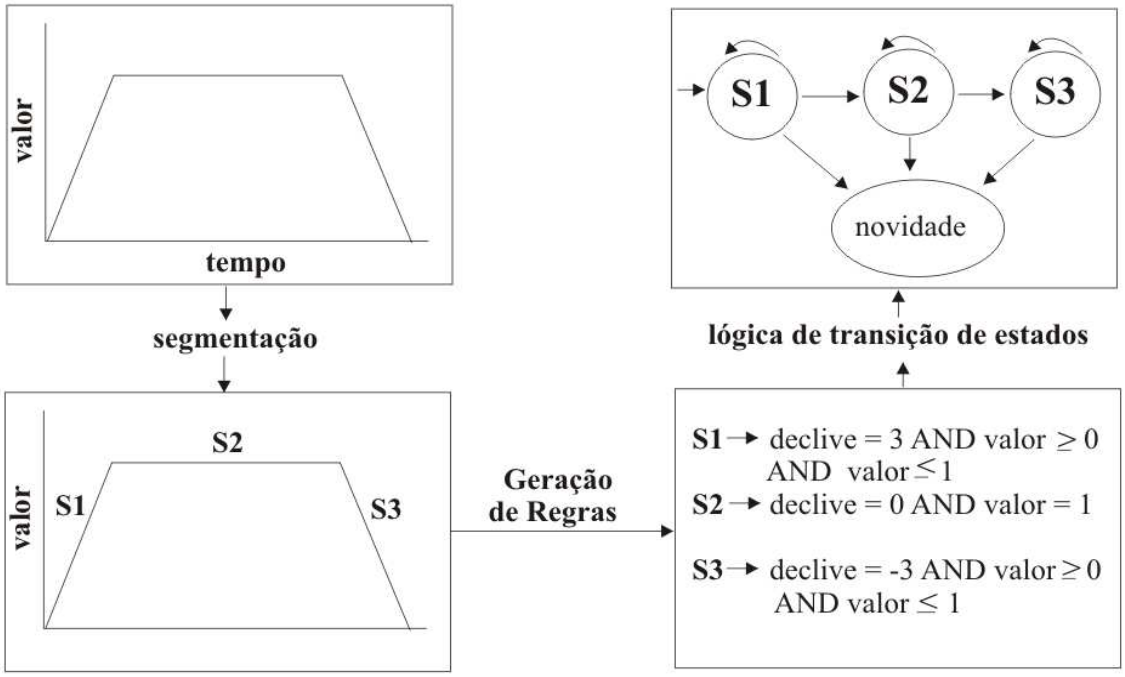


**Figura 3.5** Abordagem para detecção de novidades baseada na transição de estados proposta em [SC05]

Nesse sistema, a novidade é identificada caso ocorra uma transição, na fase de teste, que não se encontre no automaton construído.

Essa abordagem carrega um problema que é o armazenamento de todo o conjunto de treinamento para representar todos os estados. A amostra de teste deve ser comparada a todas as amostras de treinamento para que seu estado seja identificado. Após essa identificação verificamos se, caso ocorra uma transição de estados, a transição é ou não válida.

Esse sistema é um sistema para classificação com exemplos de uma única classe (para seu treinamento são fornecidos apenas dados da classe normal), porém, após a identificação dos estados ele se torna um pseudo classificador para múltiplas classes.

## 3.4 Métodos para Avaliação de Desempenho

No intuito de comparar os desempenhos dos métodos para detecção de novidades com exemplos de uma única classe abordados nesse trabalho, nós faremos uso de curvas ROC (*Receiver Operating Characteristic*) [Faw06] e de suas áreas sob as curvas (AUC - *Area Under Curve*).

Curvas ROC são técnicas para visualização e seleção de classificadores baseadas em seus desempenhos. Um exemplo de aplicação dessa técnica ocorre na teoria da detecção de sinais para mostrar o *trade-off* (compromisso) entre acertos na classe normal e erros na classe normal [Swe88] [Ega75].

Em um teste de diagnóstico existem dois tipos de erro que podem ocorrer na decisão, a escolha de um falso normal (no sentido de declarar uma pessoa enferma como sã) ou a escolha de uma falsa novidade (declarar uma pessoa sã como doente). Por exemplo, para um profissional

que tem perante si um dado diagnóstico para uma doença, ao ter que decidir, ele irá preferir uma falsa novidade a um falso normal - principalmente se a doença for contagiosa - pois este tipo de erro conduzirá a "um mal menor"em termos de diagnóstico.

A figura 3.6 mostra um exemplo de curva ROC onde o classificador consegue distinguir perfeitamente os objetos entre normal e novidade. Na prática esse caso é, na maioria das vezes, irreal pois problemas do mundo real, geralmente, não são linearmente separáveis. Na figura 3.6, *PD* significa probabilidade de detecção e *PFA* significa probabilidade de falso alarme.

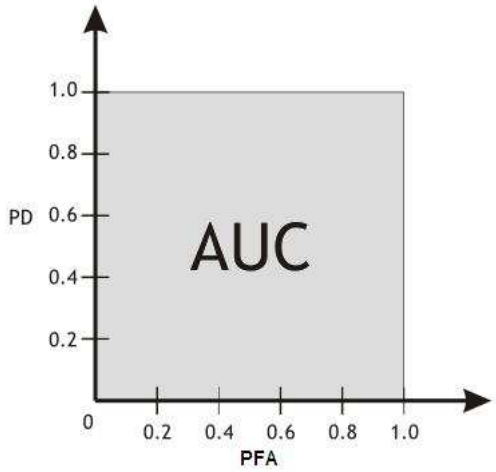


**Figura 3.6** Curva ROC perfeita, $AUC = 1$.

Para a obtenção dos pontos da curva ROC, quatro possíveis situações na classificação devem ser levadas em consideração:

1. True Positive (TP) - quando uma amostra da classe normal é corretamente classificada como normal;

2. True Negative (TN) - quando uma amostra da classe novidade é corretamente classificada como novidade;

3. False Positive (FP) - quando uma amostra da classe novidade é incorretamente classificada como normal; e

4. False Negative (FN) - quando uma amostra da classe normal é incorretamente classificada como novidade.

Levando em conta as quatro possíveis situações acima a equação 3.25 fornece os valores para cálculo dos pontos da ROC. Na equação 3.25, *P* representa o número de amostras da classe positiva enquanto que *N* representa o número de amostras da classe negativa.

$$PD = \frac{TP}{P} = \frac{TP}{TP+FN} \qquad PFA = \frac{FP}{N} = \frac{FP}{TN+FP} \tag{3.25}$$

Com esses valores podemos ainda definir a probabilidade de um padrão classificado como positivo ser realmente desta classe (*precision*), e a probabilidade de se classificar corretamente os padrões considerando as duas classes (*accuracy*). A métrica *precision* trata apenas a capacidade de detecção dos padrões da classe positiva. *Accuracy*, por sua vez, avalia de modo geral a capacidade de detecção e exclusão de um classificador. A equação 3.26 ilustra o cálculo dessas duas métricas.

$$precision = \frac{TP}{TP+FP} \qquad accuracy = \frac{TP+TN}{P+N} \tag{3.26}$$

A figura 3.7 consiste em três curvas ROC geradas de maneira hipotética. Segundo essa figura, três diferentes níveis de discriminação de um problema foram alcançados. Classificadores com melhor desempenho produzem curvas mais próximas à curva ideal da figura 3.6. A diagonal mostra o caso em que o poder de discriminação se iguala ao acaso, ou seja, o discriminante (levando em conta os atributos fornecidos para a resolução do problema) é aleatório.

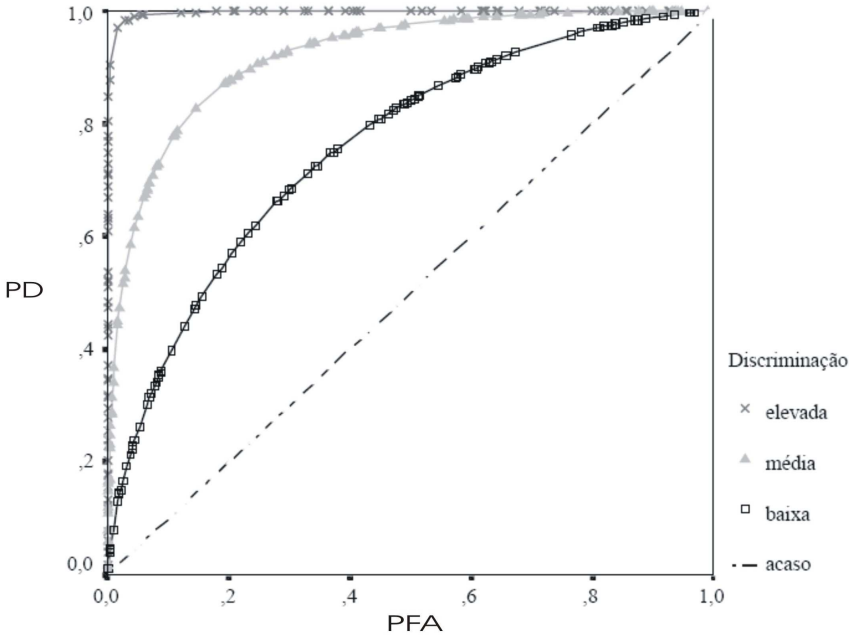


**Figura 3.7** Curvas ROC representativas de três diferentes graus de capacidade de discriminação.

Classificadores para classificação com exemplos de uma única classe compartilham o modo de operação e podem ser descritos, em alto nível, como uma função $F(x)$, tal que se $F(x) < \theta$ o objeto $x$ é dito ser uma novidade, senão o objeto é classificado como normal. De acordo com essa representação, podemos dizer que classificadores dessa natureza funcionam baseados em limiares (*threshold*) de operação, $\theta$. Nesse trabalho iremos nos referir a esses limiares como pontos de operação. Logo, um classificador pode ser representado como um modelo $F(x)$, gerado pelo treinamento de determinado algoritmo, mais um ponto de operação. Diferentes pontos de operação resultam em diferentes desempenhos na classificação. Alterando um ponto de operação o classificador pode aumentar sua tolerância a novidades ou ter uma maior taxa de detecção de novidades.

A figura 3.8 ilustra um exemplo de classificador onde um determinado ponto de operação foi escolhido de forma a melhorar a taxa de classificações normais em detrimento à detecção de objetos novidades.

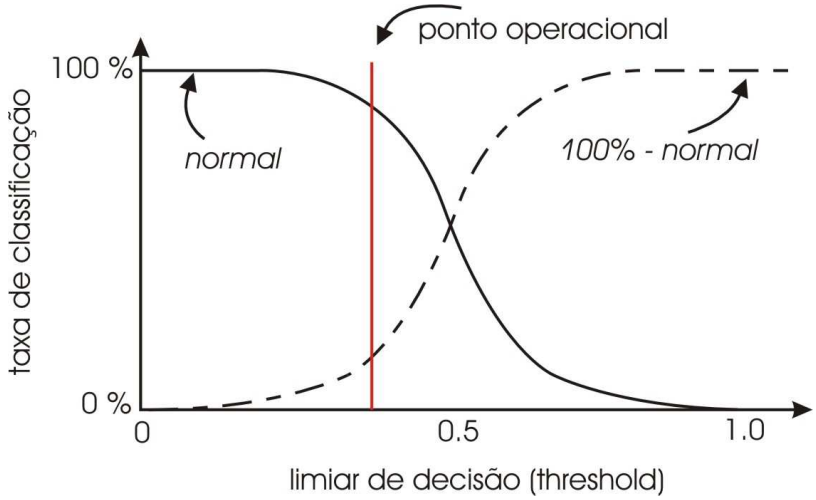


**Figura 3.8** Sobreposição de duas distribuições hipotéticas.

Para a geração da curva ROC duas abordagens são possíveis. A primeira consiste na variação dos pontos de operação para a obtenção das coordenadas da curva e para cada variação a execução de um teste, de maneira que se possa calcular a probabilidade de detecção (*PD*) e a probabilidade de falso alarme (*PFA*). Um problema dessa abordagem é o não conhecimento dos pontos de operação que contem valores de modo que as taxas *PD* e *PFA* se alterem. Nesse caso podemos gerar curvas que não contem a maior AUC possível para determinado classificador e base de dados. Outro problema consiste na necessidade de tantos testes quanto forem os números de pontos da ROC, o que faz com que a criação da curva seja uma tarefa bastante custosa. Uma segunda abordagem, mais eficiente, para a criação de curvas ROC foi introduzida em [Faw06]. Nessa abordagem, para cada objeto, $x$, testado pelo classificador é encontrada uma saída denominada *score*, onde $F(x) = score_x$. Nessa abordagem apenas um teste é necessário pois a posterior execução de um algoritmo, ilustrado a seguir, gera a ROC com maior AUC possível para aquele classificador aplicado à base de dados de teste. O pseudo-código a seguir mostra a geração dos pontos da ROC seguindo o método descrito em [Faw06].

1. **Entradas**: L, o conjunto de amostras de teste; $f(i)$, a estimativa probabilística de que a amostra i é da classe normal; P e N, os números de amostras da classe normal e da classe novidade, respectivamente.

2. **Saída**: R, uma lista de coordenadas da ROC ordenada em ordem crescente da taxa de falsos positivos.

3. **Requer**: P > 0 e N > 0.

4. $L_{ordenado} \leftarrow$ L ordenado de forma decrescente em relação aos *scores*

5. FP $\leftarrow$ TP $\leftarrow$ 0
6. R = $\langle\rangle$
7. $f_{prev} \leftarrow -\infty$
8. $i = 1$
9. ► **ENQUANTO** i $\leq \#L\{$
   ► **SE** $f(i) \neq f_{prev}$ **ENTAO**
   ► **INSERE** $\left(\frac{FP}{N}, \frac{TP}{P}\right)$ **EM R**
   ► $f_{prev} \leftarrow f(i)$
   ► }
   ► **SE** $L_{ordenado}[i]$ é uma amostra da classe normal ENTAO{
   ► TP $\leftarrow$ TP + 1
   ► }**SENAO** {
   ► FP $\leftarrow$ FP + 1
   ► }
   ► **FIM DO ENQUANTO**
   ► **INSERE** $\left(\frac{FP}{N}, \frac{TP}{P}\right)$ **EM R** //coordenada (1,1)
10. **FIM**

Nesta dissertação, a comparação entre os métodos será feita, também, em termos de armazenamento de protótipos. Computadores oferecem maior poder computacional e de armazenamento todos os anos, métodos que requerem vários minutos para a avaliação de apenas uma amostra de teste podem ser inúteis na prática. Como o treinamento é, na maioria das vezes, realizado *off-line*, os custos do treinamento geralmente não são levados em consideração. Problemas específicos podem aceitar uma pequena queda no desempenho do treinamento, contanto que também ocorra a diminuição do modelo para representação do problema.

A avaliação de desempenho na detecção de novidades em séries temporais ocorre de forma mais prática. Em séries temporais nós tentamos encontrar um ponto operacional onde ocorra no mínimo uma detecção de novidade da forma *true-negative* em cada região contendo amostras da classe novidade de maneira que ocorram, também, o mínimo de falsas novidades (amostras da classe normal classificadas como novidades). A figura 3.9 mostra um exemplo de série temporal com essa característica. Nessa série, nós, visualmente, pré-definimos as novidades por volta dos pontos 209 e 313 do gráfico.

## 3.5 Pré-processamento de Séries Temporais e Detecção de novidades com Exemplos de uma Única Classe para Séries Temporais

De acordo com o que foi visto até agora, os métodos apresentados podem apenas ser aplicados a um conjunto de vetores, e, dessa maneira, estes métodos não são diretamente aplicáveis a dados de séries temporais. A fim de resolver este problema, nós temos que, primeiramente,

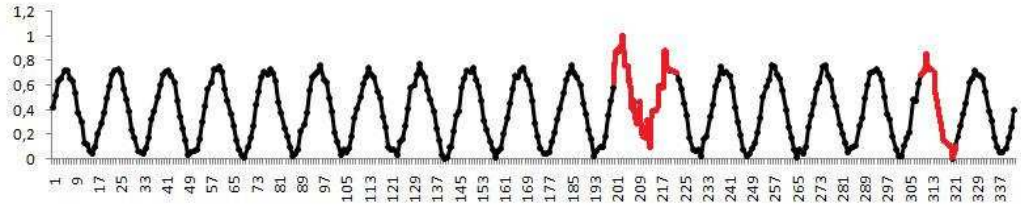


**Figura 3.9** Exemplo de série temporal que apresenta duas regiões da classe novidade (regiões na cor vermelha).

encontrar uma maneira de representar a série temporal como um conjunto de vetores. Uma solução para este problema é o uso de janelas deslizantes [MP03].

A conversão usando janelas deslizantes ocorre da seguinte maneira: dada uma série temporal $x(t), t = 1...N$ , essa série pode ser decomposta de seu espaço *1*-dimensional original em um espaço *w*dimensional, onde $w$ é chamado de janela de tempo da série temporal, usando o processo de atraso no tempo de acordo com a equação 3.27.

$$x_w(t) = [\ x(t-w+1) \quad x(t-w+2) \quad .. \quad x(t)\ ] \tag{3.27}$$

Resumindo, uma série temporal $x(t)$ pode ser convertida em um conjunto de vetores $T_w(N) = \{x_w(t), t = w..N\}$.

Pela nossa representação, $T_w(n)$ representa a $x(n+w)$ amostra na série temporal original, com $n$ variando de $w$ até a quantidade total de registros da série temporal original, ou seja, $N$. Note que, depois do processo de conversão, o conjunto de treinamento contém $N-w+1$ elementos.

Depois de realizada a conversão, deve-se aplicar o operador de normalização à série temporal no intuito de obter resultados válidos no uso de classificadores com funções de kernel sensíveis à magnitude de valores dos atributos da base.

As representações normal e novidade ocorrem da seguinte maneira: um registro,*x(t)*, na série temporal original é classificado como novidade se um dos elementos no intervalo *x(t-w)* a *x(t)* for classificado como novidade; senão, *x(t)* é classificado como normal.

A figura 3.10 ilustra um exemplo onde uma série temporal contendo, originalmente, 2060 amostras foi convertida em um conjunto de treinamento com 2000 amostras utilizando uma janela de tempo de 60 amostras. Para esse exemplo a amostra de índice 1310, após conversão, foi classificada como novidade o que fez com que as 59 amostras anteriores fossem também classificadas como novidade. Nesse exemplo temos o intervalo de pontos 1370 a 1430, na série original, classificado como novidade.

## 3.6 Considerações Finais

Métodos para classificação com exemplos de uma única classe se baseiam em três diferentes abordagens: estimativa de densidade, estimativa de fronteira entre as classes e reconstrução dos dados. A estimativa de densidade retorna a mais completa descrição dos dados, mas pode

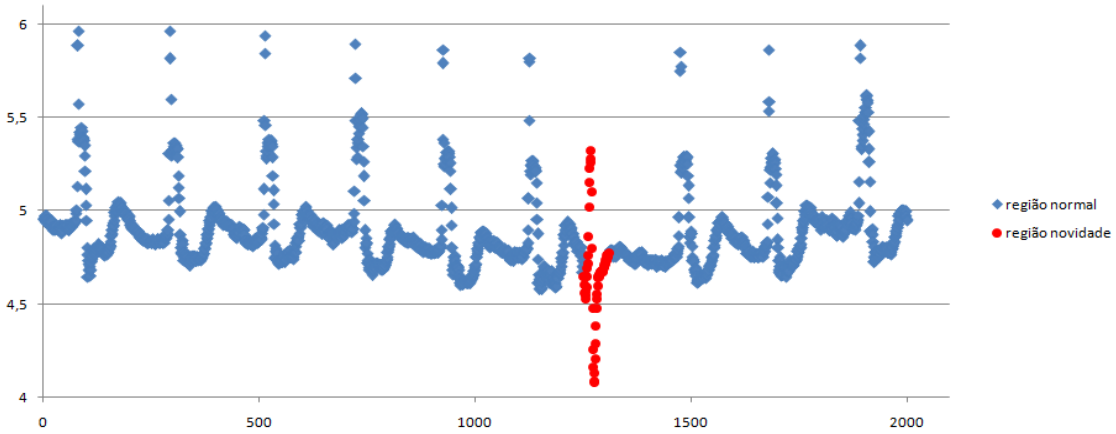


**Figura 3.10** Série temporal com trecho detectado como novidade.

requerer um grande armazenamento de dados. Para uma menor descrição dos dados um método que estima a fronteira entre as classes pode ser mais adequado. O método de reconstrução, por sua vez, tem a habilidade de incorporar conhecimento a-priori (extra) do problema.

A maioria dos métodos para classificação, baseados em exemplos de uma única classe, utiliza a abordagem de modelar distribuições de dados normais e assim estimar a probabilidade de uma amostra de teste pertencer a essa distribuição. Em classificadores desse tipo, não é preciso especificar ou fazer suposições relativas à natureza dos dados. Em contrapartida, o montante de dados no treinamento e a qualidade dos mesmos se tornam cruciais para a criação de um modelo de bom desempenho.

No próximo capítulo introduziremos os métodos *kNNDDSRM* e kernel *kNNDDSRM*, propostos nesta dissertação, e analisaremos os aspectos de classificação com exemplos de uma única classe em relação a esses classificadores.

CAPÍTULO 4

# Método Propostos para Detecção de Novidades

Neste trabalho foram desenvolvidas duas abordagens diferentes para o problema da detecção de novidades, ambas se encaixam no paradigma de classificação com exemplos de uma única classe e se baseiam no funcionamento do NNSRM (classificador pra múltiplas classes apresentado na seção 2.3.3). A primeira, denominada kernel *k*NNDDSRM, se aplica tanto ao problema da detecção de novidades em dados atemporais como em séries temporais. A segunda abordagem é específica para detecção de novidades em séries temporais e funciona, basicamente, monitorando a transição de estados na série temporal. Nosso objetivo foi desenvolver métodos que possuam complexidade reduzida na fase de teste através da redução de protótipos armazenados.

## 4.1 NNDDSRM e suas Extensões

Nesta seção, introduzimos o método NNDDSRM e evoluímos a análise dessa abordagem até a apresentação dos métodos *k*NNDDSRM e kernel *k*NNDDSRM, sendo este último o mais sofisticado. Em todos os casos, o produto da execução dos métodos é a descrição da classe normal, porém, cada método resulta em uma descrição diferente. No caso mais simples uma hiperesfera contendo todos os dados da classe normal é computada. A fim de minimizar a chance de aceitação de novidades, o volume dessa hiperesfera é minimizado.

O método kernel *k*NNDDSRM possui a habilidade de mapear os dados em uma nova dimensão sem o aumento excessivo do custo computacional. Com esse mapeamento descrições mais flexíveis podem ser obtidas. Será mostrado como a sensibilidade a objetos da classe novidade pode ser controlada de maneira flexível na fase de treinamento.

### 4.1.1 NNDDSRM

O método NNDDSRM tem como principal contribuição a redução do número de protótipos que precisam ser armazenados para a representação dos dados em relação ao NNDD. Essa redução produz pelo menos duas melhorias: a primeira é a redução no tempo de busca por vizinhos durante a classificação e a segunda é a redução na memória necessária para o armazenamento do conjunto de referências.

O NNDD armazena todas as amostras como protótipos para o uso do classificador. Muitas dessas amostras são redundantes, ou seja, elas armazenam informação já disponível por outras amostras já presentes no conjunto de referências. Dessa forma, nosso método (NNDDSRM) visa a remoção de tais amostras do modelo do classificador.

O NNDDSRM se baseia no NNSRM, algoritmo para classificação com múltiplas classes baseado no NN e no conceito SRM (ver seção 2.2 e referências [VC74] [KK03]). A idéia do NNSRM, para o caso de classificação com apenas duas classes, é incluir no conjunto de referências apenas as amostras de treinamento que se situam na fronteira entre as classes. As amostras de treinamento são incluídas no conjunto de referências até o erro de treinamento se tornar zero. A função de distância utilizada no NNDDSRM é a distância Eclideana.

O primeiro passo do NNDDSRM consiste em computar o centro de massa (*cm*) do conjunto de treinamento, como mostra a equação 4.1:

$$cm = \left( \sum_{i=1}^{n} a_{i1}, \sum_{i=1}^{n} a_{i2}, \sum_{i=1}^{n} a_{i3}, \quad .. \quad , \sum_{i=1}^{n} a_{ij} \right) \tag{4.1}$$

onde $n$ é o número de amostras no conjunto de treinamento, $j$ o número de atributo de cada amostra e $a_{ij}$ o atributo $j$ da amostra $i$.

Na fase de treinamento, o NNDDSRM irá computar dois diferentes conjuntos de atributos, chamados, conjunto de rejeitados (*CR*) e conjunto de protótipos (*CP*). *CR* contém as amostras do conjunto de treinamento mais distantes do centro de massa como mostra a figura 4.1. A idéia principal é que uma fração do conjunto de treinamento (*fracrej*) seja considerada da classe novidade. Em nosso método, essas amostras são incluídas no conjunto *CR*. O conjunto *CP* é responsável por armazenar os protótipos que delimitam a região de amostras da classe normal (ver figura 4.1). As amostras mais internas, ou seja, aquelas mais próximas ao centro de massa não serão incluídas em *CP*. O número de amostras a ser armazenado em CP é determinado como no NNSRM, amostras de treinamento são incluídas em *CP* até que o erro de treinamento seja igual a zero.

Após o treinamento nós temos dois conjuntos de amostras, *CR* e *CP*. Ambos são usados na fase de teste do algoritmo, portanto, o número total de amostras armazenadas pelo classificador é o número de amostras contidas em *CP* somado ao número de amostras contidas em *CR*.

A figura 4.1 mostra um exemplo de classificação em que nós temos duas amostras, $p_1$ e $p_2$. Neste exemplo, $\frac{d_{11}}{d_{12}} < 1$ e dessa forma $p_1$ é classificado como normal. Facilmente, podemos também observar que $p_2$ será considerada novidade pelo nosso algoritmo.

O pseudo código a seguir mostra a fase de treinamento do NNDDSRM.

1. carrega o conjunto de treinamento (*TS*) na memória
2. computa o centro de massa (*cm*) de *TS*
3. calcula a distância entre *cm* e cada amostra em *TS*
4. ordena *TS* em ordem crescente de distâncias ao centro de massa
5. remove as *fracrej*% amostras do conjunto de treinamento e adiciona no conjunto *CR*
6. remove as duas amostras, em *TS*, mais distantes de *cm* e insere em *CP*
7. PARA TODO padrão de treinamento ($p$)

   $d_1$ = menor_distancia(dist($p$,$q$) $\mid$ $q \in RS$)

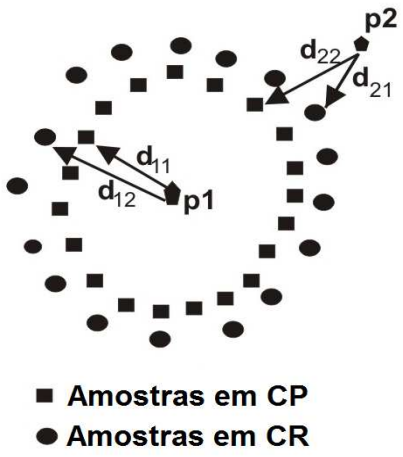


**Figura 4.1** Exemplo de tomada de decisão realizada pelo classificador NNDDSRM.

$d_2$ = menor_distancia(dist($p$,$q$) | $q \in PS$)

IF $(d_1/d_2) < 1$

erro++

8. SE erro $> 0$

//remove os dois padrões em *TS* mais distantes do centro de massa e insere em *PS*, erro $= 0$

//retorna a (7)

SENAO

//fim

A fase de teste consiste na escolha de um ponto de operação apropriado, de acordo com a seção 3.4, ou seja, um ponto que maximize a aceitação de objetos normais minimizando a aceitação de objetos novidades. Dessa forma, a equação 4.2 ilustra a função discriminante do algoritmo para um um objeto $z$ de teste.

$$f(z) = \begin{cases} \text{z é normal} & se \quad \frac{menor\_distancia(dist(z,q)|q \in RS}{menor\_distancia(dist(z,q)|q \in PS} \geq \theta \\ \text{z é novidade} & caso \quad contrario \end{cases} \tag{4.2}$$

### 4.1.2 *k*NNDDSRM

Uma extensão do NNDDSRM, o *k*NNDDSRM [COC08], envolvendo os $k$ vizinhos mais próximos, do padrão $z$ de teste, nos conjuntos *CP* e *CR* também foi desenvolvida. O treinamento do kNNDDSRM é idêntico ao treinamento do NNDDSRM original. A abordagem *k*NNDDSRM testa cada uma das $k$ mais próximas amostras, em *CP* e *CR*, a um padrão, $z$, de teste. A distância

da primeira amostra mais próxima em *CP* é comparada à distância da primeira amostra mais próxima em *CR* e a comparação é repetida para as próximas $k-1$ amostras em *CP* e *CR*. O seguinte pseudo-código mostra como o algoritmo toma uma decisão em relação a uma amostra *p* de teste a ser classificada:

1. *k*CR //conjunto com as *k* amostras mais próximas a *p* em *CR*, ordem decrescente
   *k*CP //conjunto com as *k* amostras mais próximas a *p* em *CP*, ordem decrescente
   NOVIDADES = 0 // número de classificações como novidade
   NORMAIS = 0 // número de classificações como normal

2. FOR( i = 1 to i <= k)
   d1 = distance(p,kRS[i])
   d2 = distance(p,kPS[i])
   SE (d1/d2) < $\theta$
   NOVIDADES++;
   SENAO
   NORMAIS++;
   END FOR

3. SE (NOVIDADES > NORMAIS)
   retorna NOVIDADE
   SENAO
   retorna NORMAL

### 4.1.3 kernel *k*NNDDSRM

A hiperesfera gerada pelo *k*NNDDSRM é um modelo muito rígido para descrição dos dados. Em geral, não pode ser esperado que essa descrição se adapte bem a diversos tipos de conjuntos de dados. Se nós pudermos mapear os dados em uma nova representação, pode ser que nós obtenhamos uma descrição que melhor caracterize a verdadeira fronteira entre a classe normal e a classe novidade.

A técnica de substituir o produto interno, no nosso caso a distância Euclidiana, pela aplicação de um kernel, $K(x_i, x_j)$, foi introduzida por Vapnik [Vap98] e é chamada de "kernel trick"(ver seção 3.2). Essa técnica é utilizada em classificadores que utilizam vetores de suporte quando as classes não são linearmente separáveis. Nesse caso, os dados são mapeados a outro espaço de características onde, a partir de então, passam a ser linearmente separáveis. Outra vantagem do kernel trick é que a introdução do conceito de kernel não adiciona custo computacional extra excessivamente, o único custo extra é a computação da função de kernel $K(x_i, x_j)$.

A principal característica do kernel *k*NNDDSRM [CO08] continua sendo a redução no número de protótipos, porém, além das duas melhorias alcançadas com o *k*NNDDSRM sem kernel (redução no tempo de busca por vizinhos mais próximos e redução no espaço de memória para armazenamento de dados) o kernel *k*NNDDSRM alcança também um melhor desempenho no problema da detecção de novidades.

O primeiro passo do kernel *k*NNDDSRM consiste na computação da matriz *nxn*, onde *n* é o número de padrões de entrada no conjunto de treinamento, com os resultados da função de kernel escolhida aplicada a cada par de padrões padrões, $(x_i, y_j)$. Após a computação da matriz, nós computamos um array que armazena a soma, $S_i$, de cada linha na matriz, como mostra a equação 4.3. Após a obtenção do array contendo as somas de cada linha da matriz ele deve ser ordenado em ordem ascendente.

$$\begin{array}{rcl} \sum_{i=1}^{\ell} k(x_i, x_1) & = & s_1 \\ \sum_{i=1}^{\ell} k(x_i, x_2) & = & s_2 \\ \sum_{i=1}^{\ell} k(x_i, x_3) & = & s_3 \\ \cdots & \cdots & \cdots \\ \sum_{i=1}^{\ell} k(x_i, x_l) & = & s_l \end{array} \tag{4.3}$$

A fase de treinamento do kernel *k*NNDDSRM é idêntica à fase de treinamento do NNDDSRM. O kernel *k*NNDDSRM irá computar os conjuntos *CP* e *CR*, porém nesse caso, *CR* irá conter as *fracrej%* amostras com menor saída da função de kernel $K(x_i, x_j)$, sendo $1 \leq i, j \leq n$ para *n* igual ao tamanho do conjunto de treinamento. Em contrapartida, *CP* irá conter as amostras $a_i$ tal que as somas $S_i$ sejam os menores valores do vetor *S* após a remoção das $fracrej\%$ amostras do conjunto de treinamento. O número de amostras em *CP* é determinado da mesmo forma que no NNDDSRM, ou seja, de forma que o erro de treinamento seja zero.

Após o treinamento, o kernel *k*NNDDSRM irá conter, assim como o NNDDSRM, apenas os conjuntos *CR* e *CP* que mais tarde serão usados na fase de teste.

Seja $\phi : X \rightarrow H$ o mapeamento com o auxílio de um kernel que transforma a amostra de treinamento de um espaço *X* para um outro espaço *H*. O pseudo-código a seguir ilustra como acontece a fase de treinamento do kernel *k*NNDDSRM.

1. Carrega os dados do conjunto de treinamento (*TS*)
2. Computa o array (*S*) contendo todas as somas da função de kernel, $K(x_i, x_j)$, entre cada amostra de entrada e o resto das amostras contidas no conjunto *TS*
3. Ordena *TS* em ordem ascendente, de acordo com *S*.
4. Remove *fracrej%* das amostras a partir do começo do conjunto *TS* e as adiciona a *CR*
5. Remove as duas primeira amostras em *TS* e as adiciona a *CP*.
6. PARA TODAS amostras de treinamento (*p*)

   d1 = max(K($\Phi(p)$,$\Phi(q)$) $\mid q \in CR$)

   d2 = max(K($\Phi(p)$,$\Phi(j)$) $\mid j \in CP$)

   SE (d2/d1) $< 1$

   erro++

7. SE erro $> 0$

   //Remove as 2 primeiras amostras de *TS*, as adiciona em *PS*, erro = 0

   //retorna a (7)

   SENAO

   //FIM

O teste de um protótipo $p$, é executado usando o seguinte pseudo-código:

r1 = max(K($\Phi(p)$,$\Phi(q)$) | $q \in CR$)
r2 = max(K($\Phi(p)$,$\Phi(j)$) | $j \in CP$)
if (r2/r1) $< \theta$
    return NOVIDADE
else
    return NORMAL

A abordagem kernel *k*NNDDSRM consiste basicamente de uma extensão do kernel 1NNDDSRM envolvendo os $k$ membros de *CP* e *CR* com maiores saídas da função de kernel para um dado objeto, $z$, de teste.

O pseudo-código a seguir mostra como o algoritmo kernel *k*NNDDSRM toma uma decisão em relação a uma amostra $z$ a ser classificada:

1. kCR //conjunto com os $k$ protótipos com maiores saídas de kernel em relação à amostra

   //$z$ em *CR*, ordem ascendente

2. kPS //conjunto com os $k$ protótipos com maiores saídas de kernel em relação à amostra

   //$z$ em *CP*, ordem ascendente

   NOVIDADES = 0 // número de amostras classificadas como novidade

   NORMAIS = 0 // número de amostras classificadas como normal

3. FOR ( i = 1 a i $\leq k$)

   d1 = K($\Phi(z)$,$\Phi(kRS[i])$)

   d2 = K($\Phi(z)$,$\Phi(kPS[i])$)

   SE (d1/d2) $\leq \theta$

   NORMAIS++;

   SENAO

   NOVIDADES++;

   END FOR

4. SE(NOVIDADES $>$ NORMAIS)

   //a amostra $z$ é classificada como novidade

   SENAO

   //a amostra $z$ é classificada como normal

### 4.1.4 Complexidade Computacional do NNDDSRM

Em métodos para classificação de dados a complexidade computacional (custo computacional) está diretamente ligada aos dados a serem tratados. No caso do NNDDSRM se não existe relação entre os dados ou a relação é muito confusa, o método pode adicionar toda a base de dados, após a criação do conjunto *CR*, ao conjunto *CP* tornando a execução do algoritmo bastante lenta. Por outro lado, se o problema for de fácil representação o algoritmo pode convergir já nas primeiras iterações; este é o caso mais comum.

#### 4.1.4.1 Pior caso

Seja o conjunto de treinamento $X$, onde $x_i \in R^n$, para $i = 1,..,l$, sendo $l$ o número de amostras no conjunto de treinamento. A inicialização do algoritmo requer a computação do centro de massa e o posterior cálculo e ordenação das distâncias das amostras de treinamento ao centro de massa encontrado (ver algoritmo NNDDSRM). A ordem de complexidade dessas duas operações mais a ordenação do conjunto de treinamento de acordo com a distância de cada amostra ao centro de massa é da ordem de $O(2l) + O(l * \log l)$, sendo o segundo termo a complexidade da ordenação do algoritmo quicksort.

No pior caso, a fase de treinamento se inicia com o conjunto *CR* e o *CP*, tendo o *CP* apenas duas amostras, e cada nova amostra de treinamento apresentada é corretamente classificada como normal até que a última amostra de treinamento apresentada é detectada como novidade e o teste recomeça novamente. Esse processo continua até que todo o conjunto de treinamento, após a remoção das amostras para criação do conjunto *CR*, é adicionado ao conjunto *CP*. A cada passo o algoritmo calcula a distância de todas as amostras de treinamento a todo o conjunto *CR* e *CP*. Dessa forma, a equação 4.4 indica o número de operações que o algoritmo executa no pior caso.

$$2l + l * \log l + \sum_{r=3}^{l} r(p-r) + g(p-r) \tag{4.4}$$

sendo $r$ o número de amostras em *CP*, $p$ o número de amostras de treinamento que não estão em *CR*, ou seja, *l(1-fracrej)*, e g o número de amostras em *CR*. O primeiro termo do somatório, $r(p-r)$, significa que, a cada iteração, haverão $\#CP * (p - \#CP)$ computações ou seja o número de protótipos em *CP* multiplicado pelo conjunto de treinamento menos *CP* e *CR*. O segundo termo do somatório diz que a cada iteração haverão $\#CR * (p - \#CP)$ computações.

Desenvolvendo a equação 4.4 temos:

$$2l + l * \log l + \sum_{r=3}^{l} rp - r^2 + pg - rg \tag{4.5}$$

Como $p = [l * (1 - fracrej)] - r$, temos:

$$2l + l * \log l + \sum_{r=3}^{l} rl * (1 - fracrej) - r^2 - r^2 + g * l - g * l * fracrej - g * r - r * g \tag{4.6}$$

*fracrej* e g são constantes e podem ser eliminadas

$$2l + l * \log l + \sum_{r=3}^{l} rl - 2r^2 - 2r \tag{4.7}$$

passando-se a constante $l$ para fora do somatório temos

$$2l + l * \log l + l \sum_{r=3}^{l} r - \sum_{r=3}^{l} 2r^2 - \sum_{r=3}^{l} 2r \tag{4.8}$$

eliminando os somatórios da equação 4.8, temos

$$2l + l * \log l + l * \frac{l(l+1)}{2} - 2 * \frac{l(l+1)(2l+1)}{6} - 2g * \frac{l(l+1)}{2} \tag{4.9}$$

resolvendo a equação 4.9 chegamos na equação que representa a complexidade do NNDDSRM

$$2l + l * \log l + \frac{l^3}{2} + \frac{l^2}{2} - \frac{2l^3}{3} - \frac{2l^2}{2} - \frac{2l}{6} - \frac{2gl^2}{2} - \frac{2gl}{2} \tag{4.10}$$

De acordo com a equação 4.10, vemos então que a ordem de complexidade do NNDDSRM, assim como a do *k*NNDSRM e do kernel *k*NNDDSRM, é de $O(l^3)$, no pior caso. Essa situação deixa a execução de nosso algoritmo abaixo da execução de algoritmos com a complexidade $O(\log n)$, $O(n)$, $O(n \log n)$ e $O(n^2)$. Porém, o pior caso é um caso quase impossível de se acontecer, na prática. A ordenação dos dados antes do treinamento e a remoção dos *fracrej*% mais distantes do centro de massa para criação do conjunto *CR* praticamente descarta a possibilidade de acontecimento do pior caso. Esse só aconteceria se cada amostra de treinamento, após a criação de *CR*, se encontrasse mais próxima a objetos pertencentes a CR que a objetos pertencentes a *CP*. O fato da escolha de valores baixos para $fracrej$ também ajuda é importante para o melhoramento no tempo de treinamento do NNDDSRM.

#### 4.1.4.2 Melhor caso

Para o melhor caso, a ordem de complexidade da inicialização permanece $O(2l) + O(l * \log l)$. No treinamento, a primeira amostra testada após a criação do conjunto *CR* já satisfaz o critério de parada que é a não ocorrência de erro nos dados de treinamento. Dessa forma o número de operações que o algoritmo executa no melhor caso é o seguinte:

$$2l + l * \log l + r(p - r) + g(p - r) \tag{4.11}$$

sendo $p = [l * (1 - fracrej)] - r$ e $r$, nesse caso, uma constante, então podemos representar a equação 4.13 da seguinte maneira:

$$2l + l * \log l + l - l * fracrej + g(l - l * fracrej) \tag{4.12}$$

como g e *fracrej* são constantes, a maior ordem de complexidade da equação 4.13 é

$$l \log l \tag{4.13}$$

Dessa forma, a complexidade de nosso algoritmo fica da ordem de $O(l \log l)$, ou seja, inferior ao pior caso (que é $O(l^3)$).

## 4.2 Detecção de Novidades Baseada na Transição entre Estados da Série

Subseqüências de séries temporais, para classificadores, podem ser vistas como pontos no espaço e então, métodos convencionais para detecção de novidades podem ser usados para encontrar quais desses pontos estão fora da descrição normal dos dados. Essa abordagem carrega um problema que é o fato não intuitivo de que novidades não necessariamente se encontram de maneira dispersa ou fora de uma descrição normal dos dados, no caso a série temporal. Um exemplo, dentre os inúmeros possíveis, dessa característica de séries temporais é ilustrado na figura 4.2. Nela vemos uma série com períodos bem definidos. A subseqüência da série na cor vermelha indica a novidade. A ocorrência da novidade se dá, não pelo fato de o trecho ser de formato incomum ou ser de magnitude acima ou abaixo da média; a novidade ocorre pelo fato de haver um trecho da série (em vermelho) com tempo de permanência no mesmo estado muito acima do normal. Esse é um caso em que métodos para detecção de novidades convencionais costumam falhar pois cada ponto isolado da série naquela região parece ser genuíno à distribuição normal dos dados, porém, a ocorrência seguida e prolongada desses pontos, por um tempo consideravelmente acima da média, pode caracterizar uma novidade.

Em casos como o da figura 4.2, vê-se uma série temporal com um comportamento bem definido. Pela regularidade no comportamento dessa série, vemos que é possível a descoberta de agrupamentos de objetos (*clusters*) que caracterizam estados. Salvador e Chan em [SC05] introduziram um método onde estados eram identificados na série e uma seqüencia normal de possíveis transições entre esses estados era identificada (ver mais detalhes na seção 3.3.6 desta dissertação). Qualquer transição não pertencente à essa seqüencia é caracterizada como uma novidade. Para isso, cada amostra da série temporal deve ser associada a um estado (*cluster*) contido na série e essa associação é feita através de um algoritmo de *clustering* que leva em consideração todo o conjunto de treinamento, o que pode resultar num alto custo computacional devido à grande quantidade de dados no conjunto de treinamento. O uso de um classificador com essa quantidade de dados pode demandar muitos recursos computacionais o tornando inviável na prática.

Nesta seção propomos um método para detecção de novidades em séries temporais, também baseado na análise das transições de estados da série, porém, pretendemos resolver o problema do alto custo computacional do método de Salvador e Chan [SC05]. O NNSRM se encaixa perfeitamente para a redução desse custo computacional. Após a fase de descoberta dos estados no conjunto de treinamento, o NNSRM busca remover as amostras menos significativas. Dessa forma temos um método para detecção de séries temporais baseado na transição de estados utilizando conceitos de minimização do risco estrutural. Como a fase mais custosa de nossa abordagem é a execução do NNSRM, que tem custo computacional $O(l^3)$ (no pior caso), o custo computacional dessa nova abordagem é também de $O(l^3)$, onde $l$ é a cardinalidade do conjunto de treinamento.

No método proposto, utilizamos para a identificação dos estados da série temporal o executável vCluster.exe pertencente à ferramenta CLUTO (*CLUstering TOolkit*) [Kar03]. O

vCluster.exe trata cada objeto como um vetor no hiperespaço. Existem cinco métodos possíveis para agrupamento de dados no vCluster.exe, desses, apenas dois nos interessa: *repeated bissections* e o paradigma aglomerativo. O paradigma aglomerativo é usado quando selecionamos a distância Euclideana como função de similaridade e o método *repeated bissections* quando selecionamos a função cosseno como função de similaridade. O funcionamento desta ferramenta está fora do escopo desde trabalho, informações mais detalhadas sobre o funcionamento dos métodos contidos nessa ferramenta estão disponíveis em [Kar03].

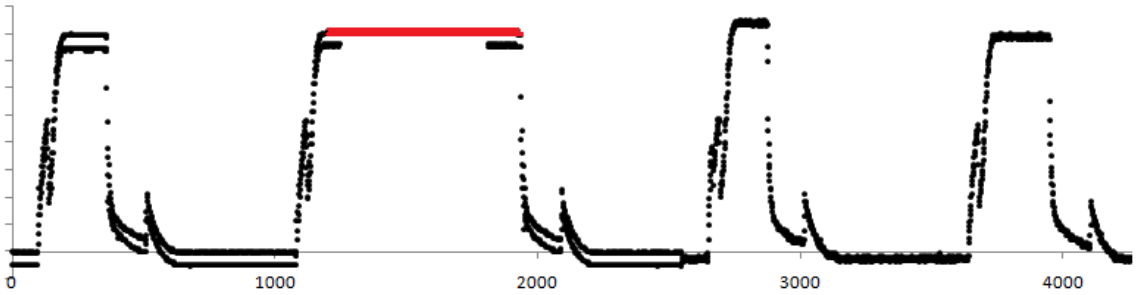


**Figura 4.2** Exemplo de ocorrência de novidade em séries temporais onde a série permanece por um tempo prolongado em um mesmo estado

O método desenvolvido é ilustrado no diagrama da figura 4.3. A entrada do sistema é uma série temporal codificada em janelas de tempo. A primeira fase consiste na busca por estados na série temporal do conjunto de treinamento. A quantidade de estados é de grande importância; uma grande quantidade de estados pode resultar num alto grau de adaptação ao conjunto de treinamento, prejudicando a generalização do classificador. Por outro lado, uma pequena quantidade de estados pode gerar um conjunto de transições com alta taxa de conexão entre elas (no pior caso, todas as transições estão conectadas) fazendo com que o sistema não detecte novidades pois todas as transições possíveis entre os estados estão presentes no conjunto de treinamento. Outro fator importante na fase da descoberta de estados é a função de similaridade utilizada pelo método de *clustering*. A figura 4.4 mostra uma série temporal dividida em estados, cada cor representando um estado distinto. Nesse exemplo a clusterização foi feita com base na distância Euclidiana como função de similaridade.

Após a execução do algoritmo de *clustering*, o conjunto de treinamento já pode ser visto como um conjunto com múltiplas classes, sendo cada estado uma classe diferente. Durante a transição entre os estados ocorre uma região de ruído, onde fica difícil definir qual estado ocorre nessa região. Para eliminar essa região de ruído é realizado um pré-processamento onde se utiliza o parâmetro janela de estabilização do estado, representado pela letra $\boldsymbol{b}$. A mudança de estados só ocorre quando houver $b$ ocorrências consecutivas de um novo estado. O valor de $b$ deve ser grande o bastante para converter regiões ruidosas em regiões com estados bem definidos, porém, o seu aumento excessivo pode causar uma alta taxa de eliminação de estados da série.

A ocorrência de um estado por grande intervalo de tempo como já citado anteriormente, também pode caracterizar uma novidade. Dessa forma, é necessário obter o intervalo máximo de ocorrências de cada estado a fim de verificar a estagnação, ou não, de uma dada série de

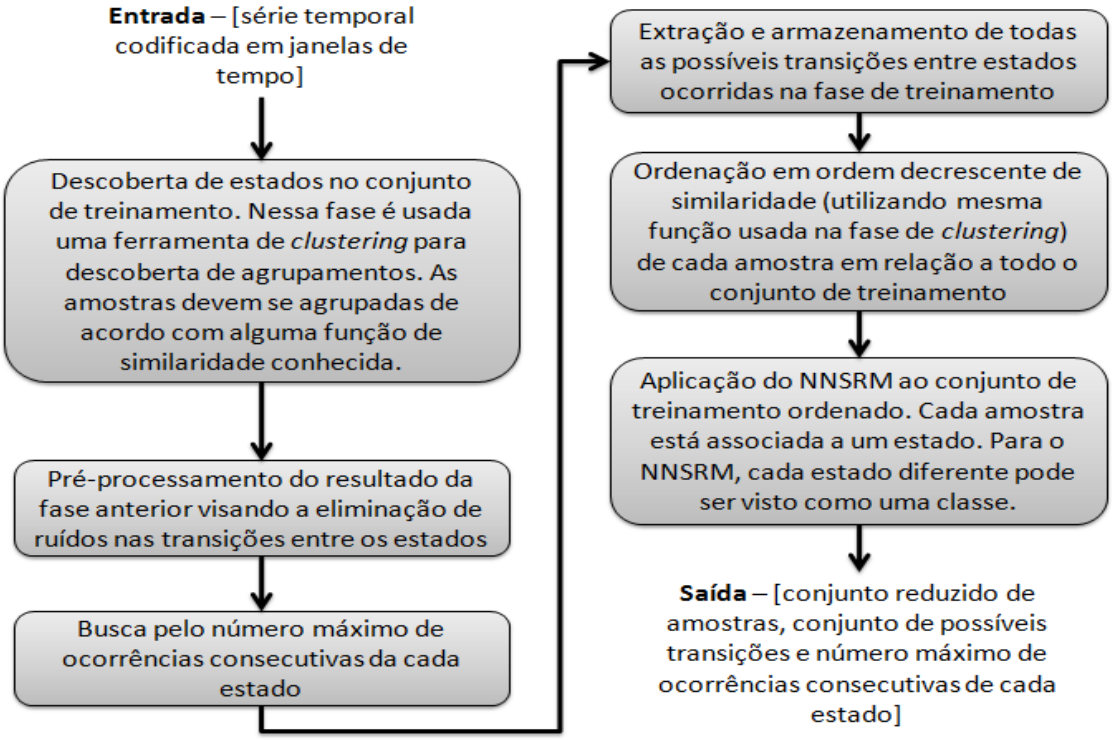


**Figura 4.3** Diagrama da fase de treinamento realizada pela abordagem proposta para detecção de novidades baseada em estados

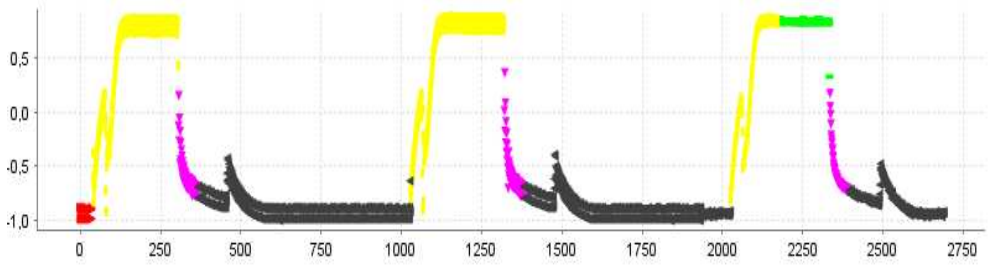


**Figura 4.4** Ilustração dos vários estados contidos em uma série temporal

teste em um determinado estado. Após a busca pelos intervalos máximos de ocorrências de cada estado no conjunto de treinamento, o sistema para detecção de novidades executa a fase de busca e armazenamento das transições entre os estados. A partir desse armazenamento pode se criar um autômato de estados finitos responsável por representar o comportamento da série.

O algoritmo NNSRM, já detalhado neste trabalho (seção 2.3.3), exige uma ordenação do conjunto de treinamento a fim da escolha dos elementos mais aptos à representação do problema. No nosso caso, essa ordenação é feita na ordem decrescente de similaridade entre as amostras. A função de similaridade usada no sistema deve ser a mesma utilizada na fase de clustering pela ferramenta CLUTO, exceto no caso em que a distância Euclideana foi usada no CLUTO. Nesse caso o sistema para detecção de novidades deve usar a função RBF. Após essa

ordenação, o conjunto resultante da execução do NNSRM, o conjunto de transições e o número máximo de ocorrências consecutivas de cada estado constituem o classificador.

A fase de teste da abordagem desenvolvida faz uso de basicamente duas verificações: se a transição entre estados existe na tabela de transições e se a permanência em um dado estado não é excessiva. No caso de uma dada transição ser classificada como novidade, as $w-1$ amostras anteriores à primeira amostra novidade serão também classificadas como novidade (sendo $w$ a janela de tempo em que a série temporal foi codificada). A figura 4.5 mostra o diagrama da fase de teste.

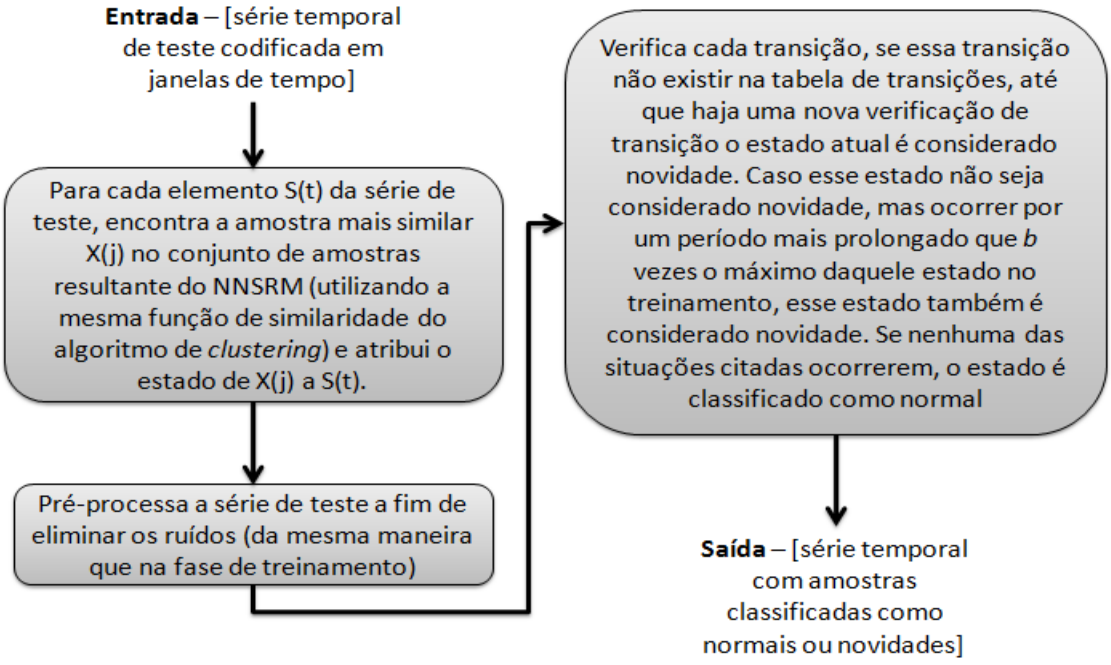


**Figura 4.5** Diagrama da fase de teste realizada pela abordagem proposta para detecção de novidades baseada em estados

A lista a seguir explica o papel de cada parâmetro utilizado pelo sistema:

**Janela ($w$)** O valor atribuído à janela de codificação da série deve ser escolhido de modo que represente de forma eficiente um trecho significativo da série (ou de um período da série) sem que seja grande o bastante a ponto de resultar em um custo computacional tão alto (na fase de treinamento) que inviabilize o uso do detector de novidades. Uma boa escolha em geral para esse parâmetro é em torno de 20% do tamanho de um período da série.

**Memória** A memória serve para indicar se uma transição de estados (na fase de teste) é ou não conhecida. A atribuição do valor 2 a esse parâmetro faz com que o sistema verifique o último e penúltimo estado no conjunto de teste, ou seja, verifique apenas se existe a transição entre o último e penúltimo estado. Caso o valor escolhido seja 3, o sistema irá verificar se existe o caminho entre o ante-penúltimo, penúltimo e último estado. Ou seja, as duas últimas transições. E assim por diante.

**Número de Estados** O número de clusters deve ser grande o bastante para que o sistema seja capaz de identificar estados com características bem definidas, e pequeno o bastante para que não se decore o comportamento da série temporal (*overfitting*).

**Função de Similaridade** A função de similaridade deve ser a mesma usada na fase da descoberta de agrupamentos (clustering) realizada pela ferramenta CLUTO [Kar03]. No uso da distância Euclideana pelo CLUTO utilizamos a função RBF. Em geral, o agrupamento de acordo com a distância Euclideana é suficiente, porém, em certos casos a função cosseno tem uma vantagem que é a não sensibilidade Às mudanças de níveis da série (apenas o cosseno entre vetores (amostras) é levado em consideração).

**Janela de Estabilização do Estado (*b*)** Esse parâmetro serve para eliminar áreas onde a transição entre estados ocorre com alta freqüência. Essas áreas podem significar ruídos ou a redundância de estados (diferentes estados para caracterizar regiões com alto grau de similaridade; ao ponto de poderem ser representadas apenas por um único estado). Um alto valor para esse parâmetro pode resultar na eliminação de estados enquanto que um valor muito baixo pode não eliminar áreas ruidosas.

**Limite de Estagnação no Estado** Pode ocorrer o caso em que uma série permaneça em um determinado estado por tempo indefinido. Nesse caso, é necessário estipular um tempo máximo de permanência da série em um dado estado. Esse parâmetro indica o fator pelo qual o número máximo de ocorrências de um dado estado na fase de treinamento será multiplicado. Dessa forma, se esse dado estado ocorrer, consecutivamente, mais vezes que esse produto o sistema irá identificar essa estagnação como novidade. Exemplo: para uma tolerância de 10% a mais em relação ao maior intervalo de ocorrência em um dado estado devemos usar o valor 1.1.

## 4.3 Considerações Finais

Neste capítulo introduzimos o método NNDDSRM (*Nearest Neighbor Data Description with Structural Risk Minimization*), o extendemos para o *k*NNDDSRM e, visando obter descrições mais flexíveis do conjunto normal de dados, chegamos ao kernel *k*NNDDSRM. A principal característica do método NNDDSRM e a redução de protótipos que causa pelo menos duas melhorias: a diminuição da necessidade de espaço para armazenamento de protótipos e aumento na velocidade de busca por vizinhos mais próximos na fase de teste. O *k*NNDDSRM considera os *k* vizinhos mais próximos, tanto no conjunto *CP* quanto no *CR*. A consideração dos *k* vizinhos mais próximos torna o algoritmo mais tolerante à presença de ruídos na fase de teste. O kernel *k*NNDDSRM consegue, através do uso de kernel, mapear os dados de entrada em um hiperespaço onde os dados são linearmente separáveis (ver seção 3.2) melhorando de forma significativa o desempenho do método na classificação.

Na detecção de novidades nem sempre novidades se encontram localizadas em pontos isolados na distribuição dos dados; novidades podem acontecer também em regiões no interior da distribuição do conjunto normal de treinamento. Métodos que criam uma descrição fechada dos dados normais, geralmente, não se comportam de maneira adequada nesses casos. Visando

solucionar esse problema, foi apresentado nesse capítulo um novo método para detecção de novidades em séries temporais baseado na transição entre estados da série fazendo uso do conceito de minimização do erro estrutural (ver seção 2.2) e baseado no método proposto por Salvador e Chan [SC05].

CAPÍTULO 5

# Experimentos

Neste capítulo, são reportados experimentos para detecção de novidades tanto em séries temporais quanto em dados atemporais, como, por exemplo, a detecção de câncer de mama na base de dados *winsconsin breast cancer* [AN07]. Este capítulo foi dividido em cinco seções: Exemplo de custo computacional em um problema real; detecção de novidades em dados atemporais; análise dos parâmetros ajustáveis do kernel *k*NNDDSRM; a detecção de novidades em séries temporais; e considerações finais.

Na primeira seção do capítulo o custo computacional (medido através do tempo de execução do treinamento) do método kernel *k*NNNDSRM é comparado ao custo do one-class SVM. Na segunda seção do capítulo, 5.2, os métodos para classificação com exemplos de uma única classe, detalhados no capítulo 3, mais os métodos kNNDDSRM e kernel kNNDDSRM, propostos nesta dissertação, (ambos fazendo uso de conceitos de redução de protótipos, ver capítulo 2) são aplicadas a várias bases de dados *benchmark* de problemas artificiais e do mundo real com diferentes níveis de complexidade. A escolha dos parâmetros foi feita de forma empírica visando encontrar o conjunto de parâmetros, contidos em um espaço pré-definido de possíveis valores, que otimizasse o desempenho de cada classificador (resultasse em uma maior AUC). A tabela 5.1 mostra os valores escolhidos para serem usados como parâmetros nos experimentos da seção 5.2. O parâmetro $h$ do Parzen_dd foi conseguido através da execução de uma rotina do toolbox para Matlab DDTolls [Tax07]. Para os métodos que baseados em kernel (kernel *k*NNDDSRM e SVDD), os experimentos dessa seção utilizaram apenas o kernel RBF. Isso se deve ao fato de que esse é o kernel mais usado na literatura e responsável pelos melhores desempenhos na prática. Por fim, a otimização quadrática, necessário ao SVDD, foi executada com o auxílio do Matlab.

A seção 5.3 faz uma análise sobre os efeitos dos parâmetros ajustáveis no método kernel *k*NNDDSRM.

Na quarta seção deste capítulo, seção 5.4, são mostrados os resultados obtidos para a detecção de novidades em séries temporais. Nesses experimentos foram utilizados os métodos

**Tabela 5.1** Parâmetros utilizados pelos métodos da seção 5.1.

| Parâmetro | Valores Possíveis |
|---|---|
| $\sigma$ (SVDD) | 5, 10, 15, 20 |
| $\sigma$ (kernel *k*NNDDSRM) | 0.5, 1.0, 1.5, 2.0 |
| $fracrej$ | 1 .. 40 |
| $k$ | 1 .. 40 |
| função de kernel | RBF |

*k*NNDDSRM, kernel *k*NNDDSRM, *k*NNDD, One-class SVM e o método para detecção de novidades baseado na transição entre estados utilizando NNSRM. O fato da troca do SVDD pelo One-class SVM se deve ao fato de a otimização quadrática, realizada pelo Matlab e utilizada pelo SVDD, não ser realizada de forma eficiente para bases com um número relativamente grande de amostras, como as usadas nos experimentos com séries temporais. O *one-class* SVM associado ao kernel RBF é conceitualmente idêntico ao SVDD. Três diferentes metodologias foram usadas para a detecção de novidades em séries temporais. A primeira é idêntica à detecção de novidades em dados atemporais, porém, quando uma novidade é detectada as $w$ (sendo $w$ a janela de tempo em que a série foi codificada) últimas amostras são também consideradas novidades. A segunda metodologia utiliza múltiplas dimensões, ou seja, a série é codificada em várias janelas diferentes e o classificador é treinado e testado para cada uma dessas janelas. Caso a mesma amostra seja detectada como novidade para todas as janelas utilizadas ela é, finalmente, dita ser uma novidade. A última metodologia utiliza a análise das transições entre estados na série temporal. A metodologia utilizada para utilização de parâmetros nos experimentos realizados nessa seção foi a escolha de forma empírica dos parâmetros, estando os valores de cada parâmetro dentro de um conjunto finito de possíveis valores.

Na seção 5.5 nós apresentaremos considerações finais sobre o resultados obtidos nos experimentos realizados.

## 5.1 Exemplo de custo computacional do kernel *k*NNDDSRM em um Problema Real

A figura 5.1 mostra uma análise feita levando-se em consideração três diferentes conjuntos de treinamento. As bases escolhidas foram a biomed, Wisconsin Breast Cancer e distribuições gaussianas, com 80, 184 e 300 amostras de treinamento respectivamente. Escolhemos o one-class SVM (para essa análise foi utilizada a ferramenta LibSVM, já citada na seção 3.3.3) para comparação dos tempos de execução dos algoritmos. De acordo com a figura 5.1, vemos que mesmo com um custo computacional relativamente alto no pior caso, o kernel *k*NNDDSRM se comporta muito bem na prática, já que, na prática, o melhor caso é consideravelmente mais provável de acontecer que o pior caso. Pela análise realizada vê-se que o one-class SVM, que faz uso de otimização quadrática, gasta um tempo significantemente maior durante o treinamento na aplicação às três bases testadas.

## 5.2 Experimentos para Detecção de Novidades em Bases de Dados Atemporais

Foram realizados experimentos com seis diferentes bases de dados atemporais, sendo que desse total, duas delas são artificiais, as bases banana [Tax01] e distribuição gaussiana [CO08]. As bases artificiais são ambas bi-dimensionais o que as torna de maior valia na análise do funcionamento dos classificadores, pois é possível avaliar os resultados visivelmente. Das bases restantes, duas delas são para problemas na classificação de múltiplas classes e nesses casos,

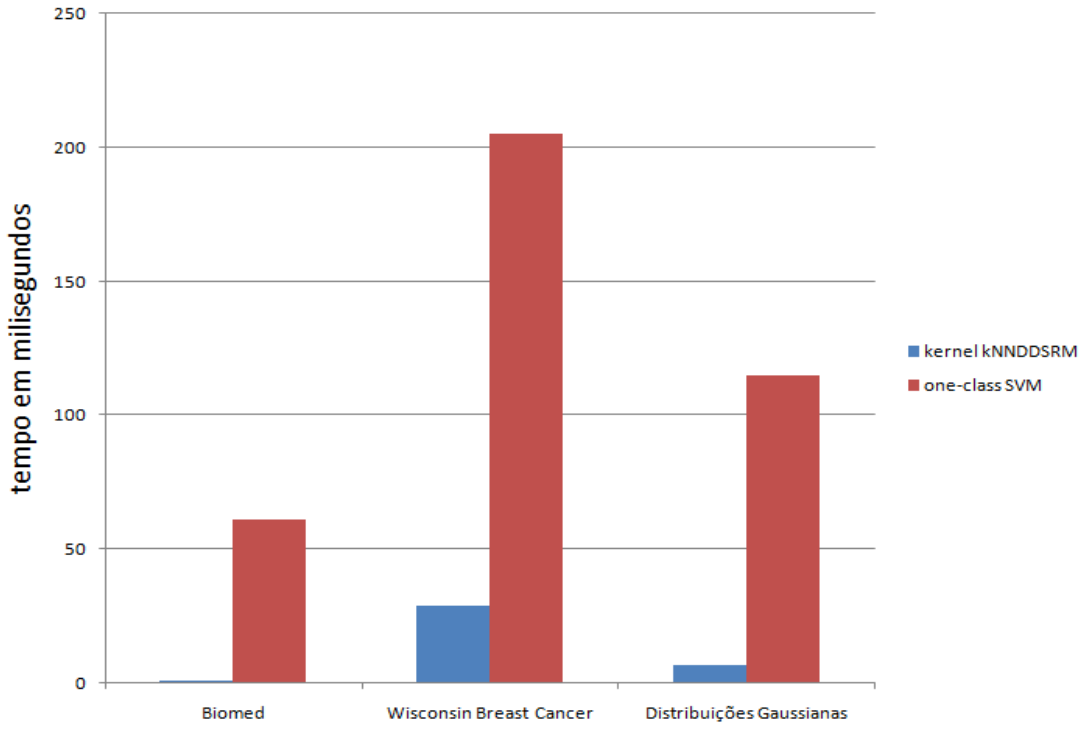


**Figura 5.1** Custo computacional de aplicações dos métodos kernel *k*NNDDSRM e one-class SVM.

uma das classes foi eleita novidade em cada experimento realizado. As bases com múltiplas classes são a Soybean e a Iris, ambas disponíveis no repositório UCI [AN07]. As outras duas bases são a Wisconsin Breast Cancer [AN07] e a Biomed, obtida em (http://lib.stat.cmu.edu/).

### 5.2.1 Experimentos Realizados com a Base Distribuição Gaussiana

Para o primeiro experimento uma base dados artificial foi gerada. Essa base de dados foi, também usada por Cao et al. [CLC03]. Essa base consiste de 600 amostras de dados bi-dimensionais geradas a partir de duas distribuições gaussianas. O conjunto de treinamento contém 300 amostras representando a classe normal e o conjunto de teste consiste de 150 amostras representando a classe normal mais 150 amostras representando a classe novidade. As amostras da classe normal foram geradas por uma distribuição gaussiana com vetor de média 0 e matriz de covariância com ambas as entradas de valor 4; as amostras pertencentes à classe novidade foram geradas por uma distribuição gaussiana com vetor de média com as duas entradas 4 e matriz de covariância, também, com ambas as entradas de valor 4. A figura 5.2 mostra a distribuição dos dados da base de dados da distribuição gaussiana.

A base gaussiana é uma base de discriminação entre as classes novidade e normal relativamente fácil, porém existe uma certa fração de dados das duas classes que se sobrepõem tornando esta base de dados não linearmente separável.

A tabela 5.2 mostra os melhores resultados obtidos, de acordo com o conjunto pré-definido de parâmetros possíveis para cada método, para essa base de dados. De acordo com a tabela

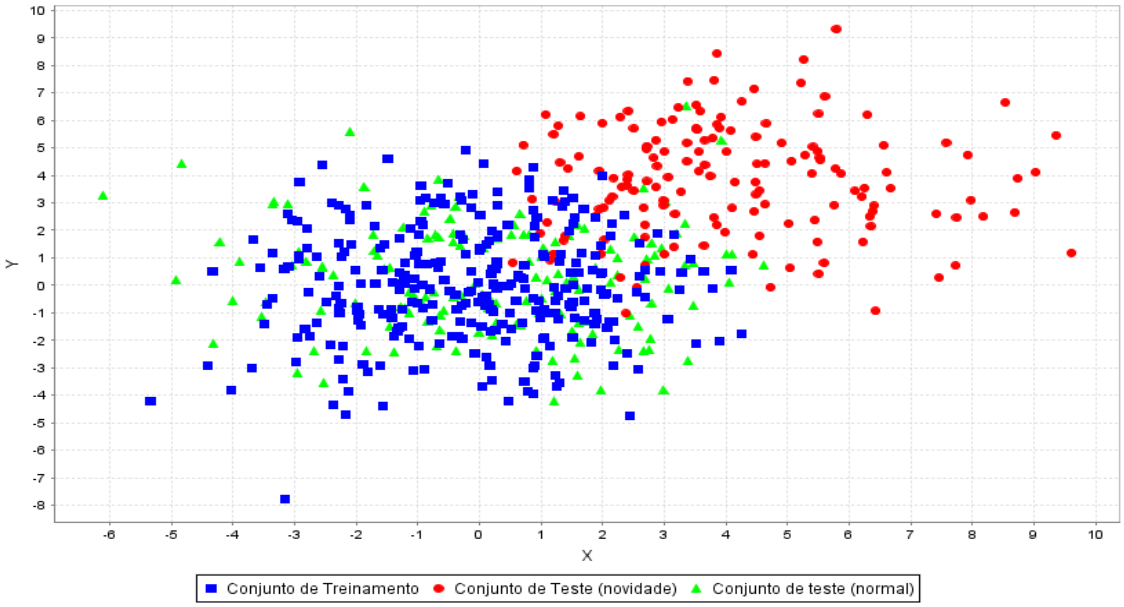


**Figura 5.2** Distribuições Gaussianas Geradas para o Treinamento e Teste.

**Tabela 5.2** Melhores resultados obtidos por cada método aplicado à base de distribuições gaussianas.

| Método | Quantidade de Protótipos | AUC |
|---|---|---|
| *k*NNDDSRM (fracrej = 7, k = 2) | 47 | 0.8362 |
| kernel *k*NNDDSRM (fracrej = 17, k = 2, $\sigma$ = 1.0) | 86 | **0.9144** |
| parzen_dd (h = 0.8818) | 300 | 0.9124 |
| kmeans_dd (k = 2) | 2 | 0.9039 |
| SVDD (fracrej = 8, $\sigma$ = 10) | 24 | 0.8940 |
| *k*NNDD (k = 2) | 300 | 0.8958 |

5.2 vê-se um melhor desempenho do método kernel *k*NNDDSRM em relação aos outros métodos aplicados. Porém, dependendo da finalidade da aplicação pode-se escolher outros métodos pois nesse caso as AUCs têm valores bastante próximos e alguns métodos necessitam de quantidades significantemente menores de protótipos para a descrição dos dados que os outros. É o caso do k-means que necessita de apenas dois protótipos para a descrição dos dados enquanto que o *k*NNDD necessita de 300 protótipos. O parzen_dd obteve a segunda maior AUC, com valor muito próximo da AUC obtida pelo kernel *k*NNDDSRM, porém, necessita do armazenamento do conjunto de treinamento completo. De acordo com a tabela 5.2 vê-se ainda uma grande diferença nos desempenhos entre os métodos *k*NNDDSRM e kernel *k*NNDDSRM, fato justificado pela maior flexibilidade na descrição dos dados com o uso de funções de kernel.

Vale ressaltar que para os experimentos com a base de distribuição gaussianas foi usado o kernel RBF para os métodos kernel kNNDDSRM e SVDD.

A figura 5.3 mostra as descrições geradas por cada método aplicado à base gerada pelas distribuições gaussianas. A descrição 5.3 a) foi gerada pelo método kmeans_dd com o parâmetro $k = 2$ (número de médias), nela podemos ver a localização dos dois protótipos necessários

para a descrição dos dados. A figura mostra ainda que houve uma convergência das médias. A descrição 5.3 b) foi gerada pelo método SVDD. Nela podemos ver a localização dos vetores de suporte localizados na superfície da descrição. Nenhum vetor de suporte fora da descrição (erro no treinamento) foi gerado pelo método. A descrição 5.3 c) foi gerada pelo método *k*NNDDSRM sem kernel. Nela visualizamos uma descrição mais justa do conjunto de treinamento onde os protótipos mais distantes do centro de massa da base de treinamento são automaticamente rejeitados pela descrição. A descrição 5.3 d) foi gerada pelo método kernel *k*NNDDSRM com o auxílio do kernel RBF. Nessa descrição vê-se que foi alcançado um resultado similar ao *k*NNDDSRM sem kernel, porém, o desempenho do método com o uso de kernel foi significativamente melhor. Os métodos Parzen_dd e kNNDD não possuem uma fase de treinamento sendo necessário apenas o armazenamento do conjunto de treinamento completo na memória. Dessa forma não há a necessidade de apresentar as descrições geradas por esses métodos.

A figura 5.4 mostra as curvas ROC geradas pelos métodos para a distribuição gaussiana de acordo com os parâmetros descritos na tabela 5.2. Para esse problema as curvas ROC têm comportamento bastante similares, porém, cada uma possuindo particularidades que podem ser decisivas na escolha do classificador. Analisando as curva ROC vemos que para uma aplicação que exija um alto grau de acerto na classe positiva obtendo também uma baixa taxa de falsos positivos (não detecção das novidades) deve-se escolher dentre os métodos kernel *k*NNDDSRM, Parzen_dd e kmeans_dd. Vemos também que para uma alta taxa de acertos na classe normal (verdadeiros positivos) com uma certa tolerância a erros na classe novidade (falsos positivos), o *k*NNDD é uma boa escolha. Se para a escolha do classificador for de grande importância o baixo uso de recursos computacionais, o kmeans_dd aparece como uma alternativa atraente, seguido pelo SVDD e *k*NNDDSRM.

### 5.2.2 Experimentos Realizados com a Base Banana

A base artificial banana é uma base bi-dimensional em formato de banana. Para nossos experimentos geramos 80 amostras da classe normal para o treinamento e mais 160 para o teste, sendo 80 da classe novidade e 80 da classe normal. Essa base foi também usada em [Tax01] [CO08] [Rät98] e [ROM98].

A figura 5.5 mostra o gráfico de distribuição dos conjuntos de treinamento e teste da base banana. Nessa figura podemos ver que existe uma separação bem definida entre as classes normal e novidade, mas vemos também que uma descrição rígida, como a criada pelo *k*NNDDSRM sem kernel, pode não representar bem a classe normal. Essa base, claramente, mostra a necessidade do uso de descrições mais flexíveis que uma simples esfera, por exemplo.

A tabela 5.3 mostra os valores das AUCs obtidas pelas curvas ROCs dos métodos aplicados à base de dados banana. Para essa base o método SVDD obteve um resultado próximo do ótimo. Com uma AUC de 0.9975 o SVDD se aproximou da discriminação perfeita entre os dados da classe normal e da classe novidade. Os métodos *k*NNDD e kmeans_dd também obtiveram desempenhos interessantes (aproximando-se também do ótimo), porém, com um rendimento um pouco abaixo do SVDD. Os métodos kernel *k*NNDDSRM e *k*NNDDSRM obtiveram resultados um pouco abaixo das três maiores AUCs, porém, alcançaram um resultado, ainda, bastante satisfatório, ficando o kernel *k*NNDDSRM com uma AUC apenas 0.0297 abaixo da

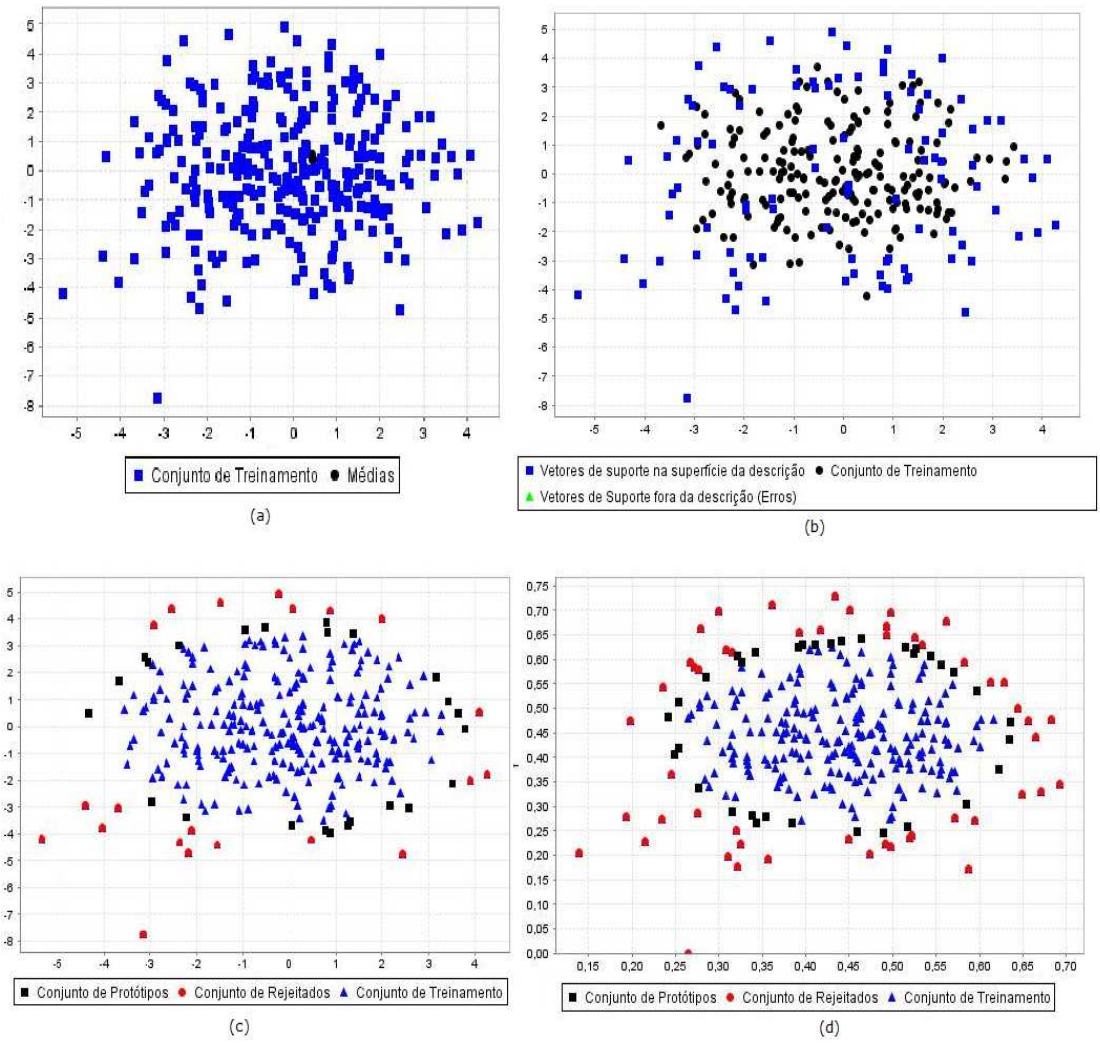


**Figura 5.3** Descrições geradas pelos métodos: a) kmeans_dd; b) SVDD; c) *k*NNDDSRM; e d) kernel *k*NNDDSRM.

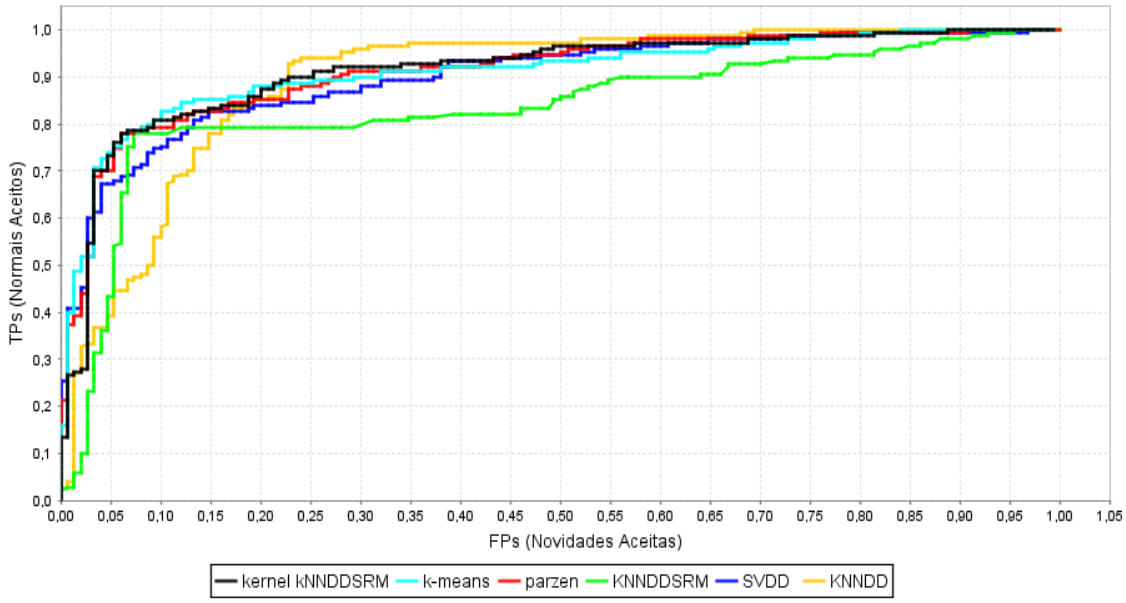


**Figura 5.4** Curvas ROC obtidas para a base de Distribuições Gaussianas.

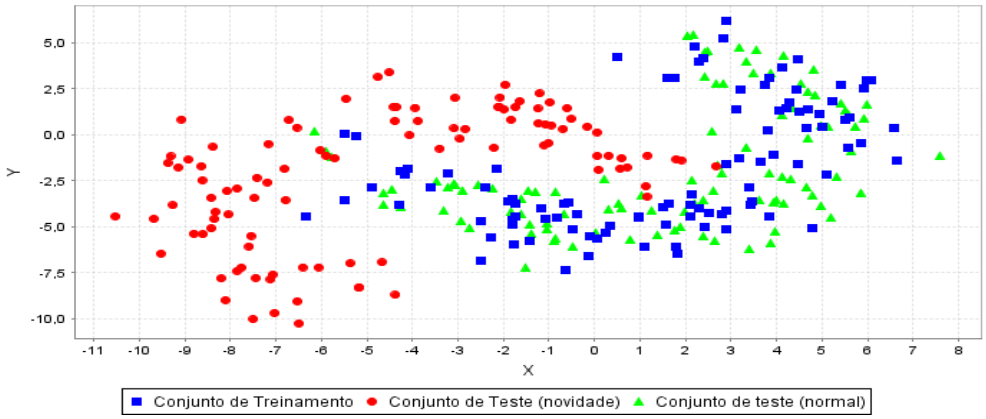


**Figura 5.5** Conjuntos de treinamento e teste gerados em formato de banana.

AUC obtida pelo SVDD. O método Parzen_dd alcançou uma AUC de apenas 0.7827 ficando com um desempenho significativamente abaixo da média dos demais métodos.

A figura 5.6 ilustra as curvas ROC geradas para a base banana. Pelo valor quase ótimo da AUC alcançada pelo SVDD, o *tradeoff* deixa de ser uma limitação pois a ocorrência de classificações normais corretamente classificadas como tais (*verdadeiros positivos*) chega a 100% com um valor ainda baixo de *falsos positivos*, padrões da classe novidade classificados como normais. Pela figura 5.6 vê-se ainda que o SVDD foi superado pelo kmeans_dd em um pequeno trecho imediatamente no início da curva ROC.

A figura 5.7 apresenta os modelos de melhor desempenho gerados por cada método para os dados da base banana. O modelo gerado pelo kmeans_dd (figura 5.7 a)) novamente obteve médias espacialmente próximas. Para essa base, o SVDD gerou o modelo que melhor repre-

**Tabela 5.3** Melhores resultados obtidos por cada método aplicado à base banana.

| Método | Quantidade de Protótipos | AUC |
|---|---|---|
| *k*NNDDSRM (fracrej = 5, k = 2) | 18 | 0.9580 |
| kernel *k*NNDDSRM (fracrej = 17, k = 1, $\sigma$ = 0.5) | 19 | 0.9678 |
| parzen_dd (h = 0.9089) | 80 | 0.7827 |
| kmeans_dd (k = 7) | 7 | 0.9836 |
| SVDD (fracrej = 5, $\sigma$ = 5) | 9 | **0.9975** |
| *k*NNDD (k = 1) | 80 | 0.9931 |

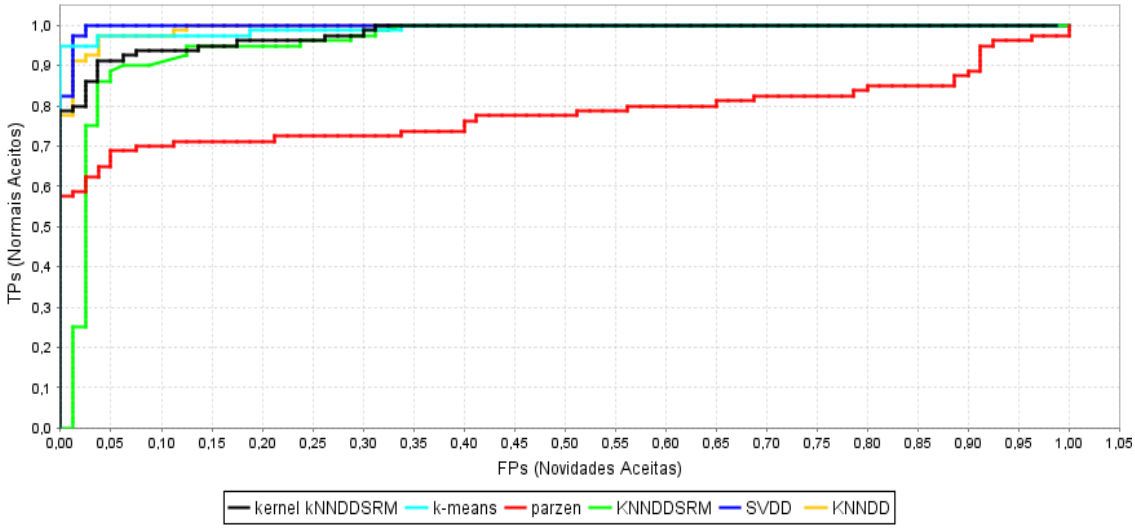


**Figura 5.6** Curvas ROC geradas para a base de dados banana.

sentou os dados, com vetores de suporte localizados na região mais convexa da distribuição dos dados. O *k*NNDDSRM, como mostra a figura 5.7 c), elegeu as 5% amostras de treinamento mais distantes do centro de massa da distribuição para a formação do conjunto de amostras rejeitadas. Finalmente, o kernel *k*NNDDSRM, escolheu as 17% amostras de treinamento em regiões de menor densidade para compor o conjunto de amostras rejeitadas.

### 5.2.3 Experimentos com a base Biomed

A base de dados Biomed, disponível no repositório Statlib, também foi considerada em nossos experimentos. Essa base de dados também foi usada em Cao et al. [CLC03]. A base Biomed foi criada em um estudo para o desenvolvimento de métodos para a identificação de portadores de uma rara desordem genética. Quatro medições, m1, m2, m3 e m4 foram feitas em amostras de sangue. Pela natureza rara da doença, existiam apenas poucos portadores da doença para a disponibilização de dados da classe doença, novidade. A base de dados contém 134 amostras da classe normal (voluntários não portadores da doença) e 75 amostras da classe novidade (voluntários portadores da doença). O conjunto de treinamento foi formado por 80 amostras da classe normal sendo o conjunto de teste formado pelo restante das amostras.

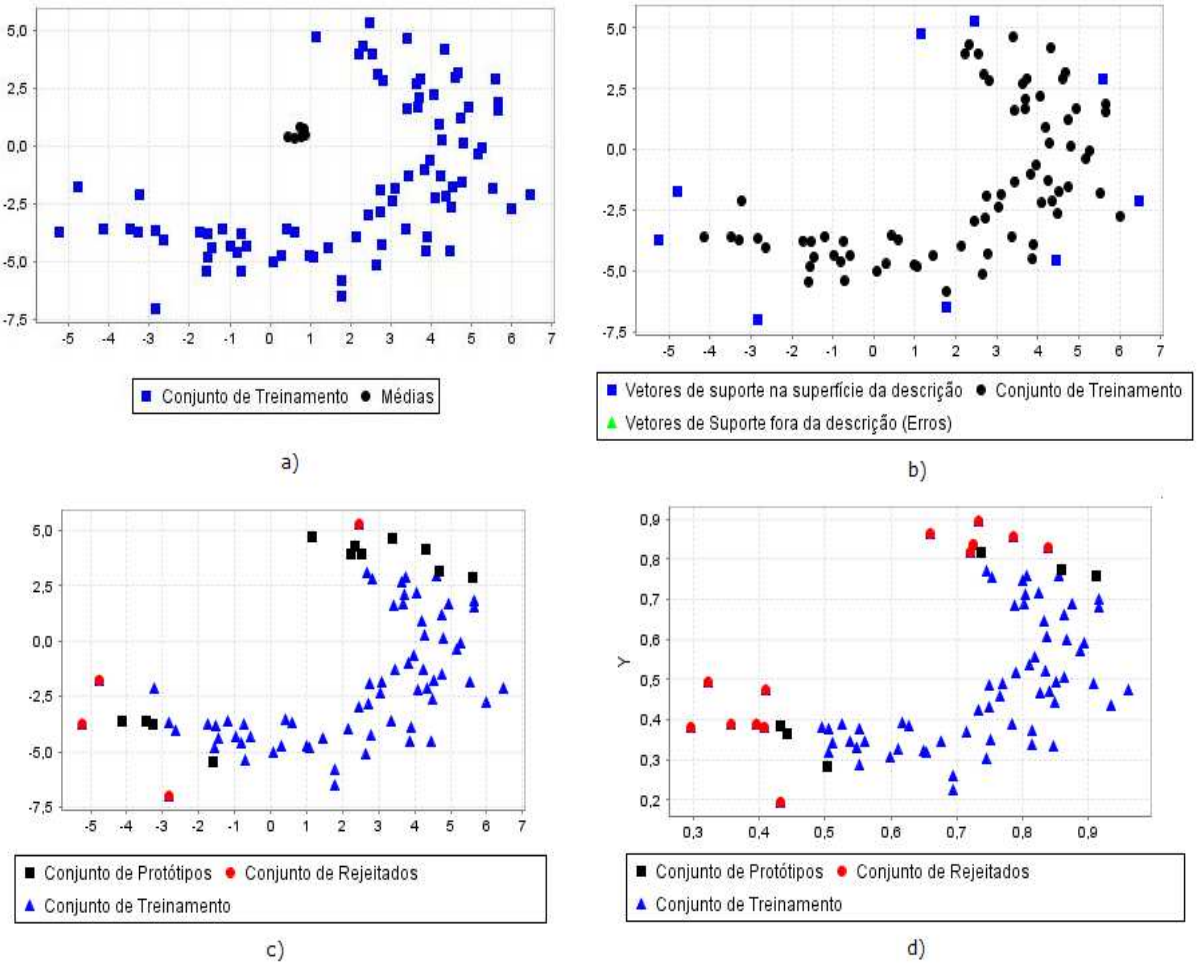


**Figura 5.7** Descrições geradas pelos métodos: a) kmeans_dd; b) SVDD; c) *k*NNDDSRM; e d) kernel *k*NNDDSRM.

De acordo com a tabela 5.4, o kernel *k*NNDDSRM obteve o melhor resultado dentre os métodos aplicados à essa base. Em número de protótipos o kernel *k*NNDDSRM foi superado apenas pelo kmeans_dd que armazenou apenas 16 protótipos. Para os experimentos com a base Biomed a surpresa foi o baixo desempenho do método SVDD que, mesmo com o armazenamento do conjunto de treinamento na íntegra, obteve melhor desempenho apenas que o método parzen_dd.

A figura 5.8 mostra as curvas ROC geradas para a base Biomed. Nela vê-se que dois métodos conseguem uma fração de *verdadeiros positivos* razoável frente à fração de *falsos positivos*, os métodos kernel *k*NNDDSRM e SVDD. Exceto pela região inicial das curvas ROC, onde se destacam o SVDD e kernel *k*NNDDSRM, os métodos se comportam de maneiras idênticas, alcançando uma alta taxa de *verdadeiros positivos* a custo de uma, também, alta taxa de *falsos positivos*.

**Tabela 5.4** Melhores resultados obtidos por cada método aplicado à base biomed.

| Método | Quantidade de Protótipos | AUC |
|---|---|---|
| *k*NNDDSRM (fracrej = 12, k = 2) | 23 | 0.8742 |
| kernel *k*NNDDSRM (fracrej = 13, k = 4, $\sigma$ = 0.5) | 21 | **0.9080** |
| parzen_dd (h = 104079) | 80 | 0.8701 |
| kmeans_dd (k = 16) | 16 | 0.8906 |
| SVDD (fracrej = 5, $\sigma$ = 5) | 80 | 0.8725 |
| *k*NNDD (k = 1) | 80 | 0.8818 |

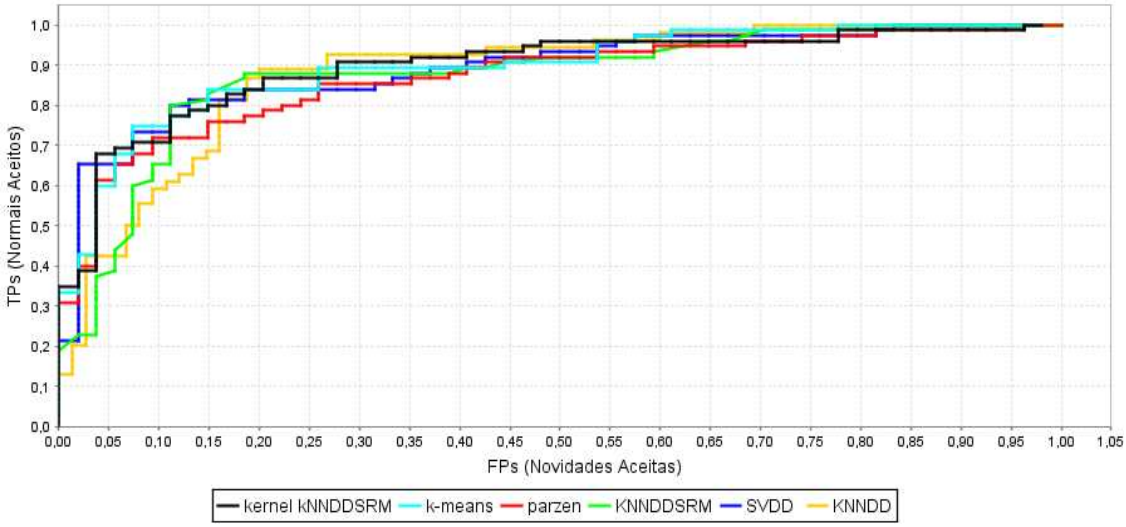


**Figura 5.8** Curvas ROC geradas para a base de dados biomed.

### 5.2.4 Experimentos com a base Wisconsin Breast Cancer

O conjunto de dados Wisconsin Breast Cancer consiste de 699 amostras. Cada amostra é composta por nove atributos de entrada mais o atributo de saída, classe benigna ou maligna. Nós selecionamos a classe benigna para representar a classe normal e dessa forma o conjunto de treinamento contém 184 amostras da classe normal e o conjunto de teste contém 239 amostras da classe novidade mais 260 amostras da classe normal. Essa base de dados também foi usada em vários outros trabalhos de mineração de dados como [Bro04] [BDM02] e [KLL01].

A seguinte lista descreve os atributos da base de dados Wisconsin Breast Cancer:

1. Espessura do tumor - real de 1 a 10;
2. Uniformidade do tamanho da célula - real de 1 a 10;
3. Uniformidade do formato da célula - real de 1 a 10;
4. Adesão marginal - real de 1 a 10;
5. Tamanho de uma célula epitelial - real de 1 a 10;

6. Núcleos expostos - real de 1 a 10;
7. Cromatina branda - real de 1 a 10;
8. Normalidade do nucléolo - real de 1 a 10;
9. Mitose - real de 1 a 10; e
10. Classe - 2 para benigno e 4 para maligno.

A tabela 5.5 mostra que que para a base Wisconsin Breast Cancer o método kernel *k*NNDDSRM teve um desempenho ligeiramente superior aos melhores parzen_dd, kmeans_dd, e kNNDD, que também obtiveram bons resultados para essa base. Dentre os melhores resultados, um método que, dependendo da aplicação, pode ter o uso mais aconselhado é o kmeans_dd pelo fato de serem necessários apenas 12 protótipos para a descrição dos dados. O kernel *k*NNDDSRM precisou de 79 protótipos, ficando atrás apenas do kmeans_dd e do SVDD, em relação ao número de protótipos. A surpresa entre os métodos no teste com a base Wisconsin Breast Cancer ficou por conta do SVDD que obteve um desempenho relativamente abaixo da média dos métodos, ficando com uma AUC 0.7735 abaixo do *k*NNDDSRM, que ficou com o segundo pior desempenho.

**Tabela 5.5** Melhores resultados obtidos por cada método aplicado à base Wisconsin Breast Cancer.

| Método | Quantidade de Protótipos | AUC |
|---|---|---|
| *k*NNDDSRM (fracrej = 14, k = 1) | 95 | 0.9396 |
| kernel *k*NNDDSRM (fracrej = 13, k = 3, $\sigma$ = 1.5) | 79 | **0.9984** |
| parzen_dd (h = 0.8) | 184 | 0.9975 |
| kmeans_dd (k = 12) | 12 | 0.9851 |
| SVDD (fracrej = 7, $\sigma$ = 5) | 50 | 0.7735 |
| *k*NNDD (k = 1) | 184 | 0.9973 |

As curvas ROC geradas para a base Wisconsin Breast Cancer, figura 5.9, mostram que os três melhores métodos (kernel *k*NNDDSRM, kNNDD e Parzen_dd) iniciam as suas curvas em pontos já com uma alta taxa de *verdadeiros positivos* e baixa taxa de falsos positivos, o que significa um bom desempenho na discriminação das classes novidade e normal. Qualquer dos três métodos é uma boa escolha para a criação de um mecanismo de detecção de novidades. A figura 5.9 ratifica também o baixo desempenho do método SVDD, que teve desempenho bem abaixo dos resultados dos métodos kmeans_dd e kNNDDSRM, que também não obtiveram bons resultados.

### 5.2.5 Experimentos com a base Diabetes

Esta base de dados possui como título original "Pima Indians Diabetes Database"e é de propriedade do National Institute of Diabetes and Digestive and Kidney Diseases. Cada amostra consiste em medições realizadas em pacientes do sexo feminino com mais de 21 anos, residentes em Phoenix, Arizona, EUA a fim de detectar a ocorrência ou não de diabetes. Das 768

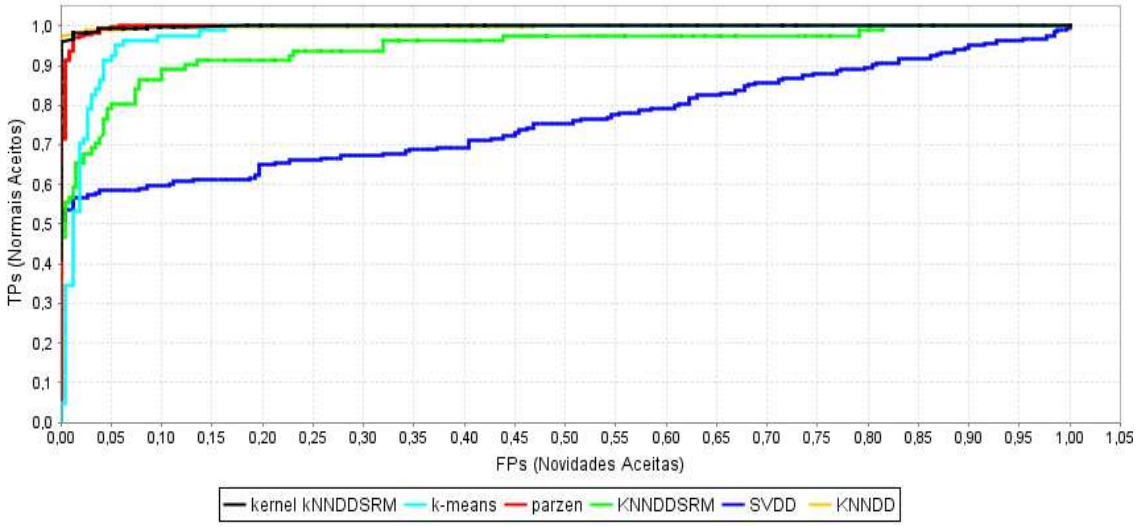


**Figura 5.9** Curvas ROC geradas para a base de dados Wisconsin Breast Cancer.

amostras disponíveis na base, foram separadas 250 amostras da classe normal (pacientes sem diabetes) para o treinamento, 250 amostras da classe normal para o teste e mais 268 amostras da classe novidade (ocorrência de diabetes) para o teste.

Cada registro contem 8 atributos de entradas mais a classe. Os atributos representam as seguintes informações:

1. Número de vezes que ficou grávida;
2. Concentração de glicose no Plasma em teste de tolerância de glicose oral;
3. Pressão sangüínea Diastólica (mm Hg);
4. Dobras na pele do tríceps (mm);
5. Aplicações de 2 horas de insulina de soro (mu U/ml);
6. Índice de massa corpórea (peso em kg/(altura em m2);
7. Função de genealogia de diabete;
8. Idade (anos); e
9. Classe: 0 - Saudável, 1 - Diabetes.

A tabela 5.6 mostra as melhores AUCs obtidas para a base Diabetes. Para essa base o kernel *k*NNDDSRM obteve a maior AUC seguida do método parzen_dd. Os resultados mostram a proximidade entre os desempenhos. A melhor AUC foi apenas 0.0157 melhor que a segunda melhor AUC e 0.0236 melhor que a terceira melhor AUC. Os métodos kNNDD e *k*NNDDSRM obtiveram as mais baixas AUCs, respectivamente. As AUCs alcançadas nesse experimento indicam um baixo desempenho em todos os testes realizados.

**Tabela 5.6** Melhores resultados obtidos por cada método aplicado à base Diabetes.

| Método | Quantidade de Protótipos | AUC |
|---|---|---|
| *k*NNDDSRM (fracrej = 39, k = 3) | 170 | 0.6554 |
| kernel *k*NNDDSRM (fracrej = 22, k = 4, $\sigma$ = 1.5) | 138 | **0.7185** |
| parzen_dd (h = 2.6492) | 250 | 0.7028 |
| kmeans_dd (k = 1) | 1 | 0.6949 |
| SVDD (fracrej = 16, $\sigma$ = 5) | 40 | 0.6911 |
| *k*NNDD (k = 1) | 250 | 0.6580 |

A figura 5.10 mostra as curvas ROC geradas para a base Diabetes. Inicialmente, a curva do método kernel *k*NNDDSRM mostra um rendimento bastante superior a todas as outras curvas, porém, ainda abaixo do que pode ser considerado um bom rendimento. As curvas se sobrepõem em grande parte de seus trajetos, mostrando a similaridade de desempenho dos métodos aplicados à base Diabetes.

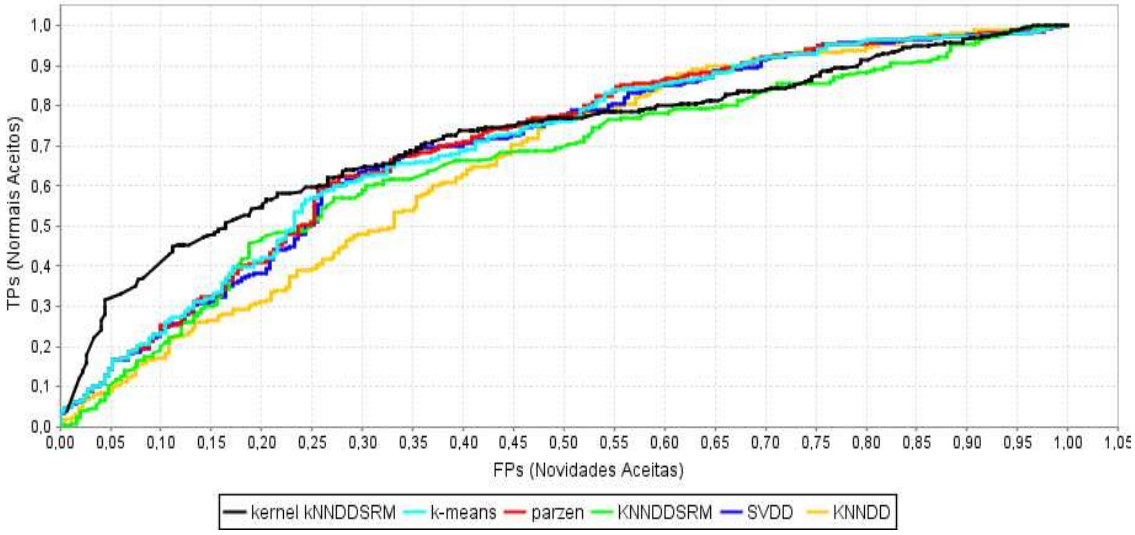


**Figura 5.10** Curvas ROC geradas para a base de dados Diabetes.

### 5.2.6 Experimentos com a base Soybean

A base de dados Soybean é uma base benchmark amplamente usada na mineração de dados. Essa base contém 19 classes, diferentemente de todas as outras bases usadas até o momento que têm apenas 2 classes e basta eleger uma das classes como novidade (as demais classes passam a fazer parte de uma única classe, a considerada normal). Após a escolha da classe novidade, todas as amostras dessa classe são retiradas do conjunto de treinamento. Na base de dados soybean, cada amostra é formada por 35 atributos, numéricos e nominais, que quando convertidos para atributos apenas numéricos resultam em 82 atributos.

Para ocaso da base Soybean, dois experimentos foram realizados, com classes diferentes para representar a novidade. No primeiro caso a classe 16 foi escolhida para representar a

novidade. A classe 16 tem 88 ocorrências no conjunto de teste que nesse caso fica com 231 amostras deixando o conjunto de treinamento com 452 amostras. No segundo conjunto de experimentos escolhemos a classe 17 para representar novidades. A classe 17 tem 91 ocorrências no conjunto de teste que nesse caso fica com 241 amostras deixando o conjunto de treinamento com 442 amostras.

#### 5.2.6.1 Classe 16

De acordo com a tabela 5.7, foram obtidos resultados bastante satisfatórios com a classe 16 como novidade da base Soybean. Os métodos parzen_dd e *k*NNDD obtiveram os melhores resultados seguidos do método SVDD, que ficou com uma AUC apenas 0.0025 abaixo das melhores AUCs obtidas. O kmeans_dd obteve uma AUC um pouco abaixo da alcançada pelo SVDD, porém com um número consideravelmente menor de protótipos necessários para a descrição, apenas 32 protótipos, contra 423 utilizados no SVDD. Os métodos kernel *k*NNDDSRM e *k*NNDDSRM obtiveram AUCs abaixo da média dos demais métodos, mas ainda razoáveis, sendo para essa base, o kernel *k*NNDDSRM o método com menor necessidade de armazenamento de protótipos para a descrição, apenas 27.

**Tabela 5.7** Melhores resultados obtidos por cada método aplicado à base Soybean com a classe 16 como novidade.

| Método | Quantidade de Protótipos | AUC |
|---|---|---|
| *k*NNDDSRM (fracrej = 12, k = 4) | 140 | 0.8841 |
| kernel *k*NNDDSRM (fracrej = 5, k = 1, $\sigma$ = 2) | 27 | 0.9138 |
| parzen_dd (h = 0.1238) | 452 | **0.9947** |
| kmeans_dd (k = 32) | 32 | 0.9877 |
| SVDD (fracrej = 5, $\sigma$ = 5) | 423 | 0.9922 |
| *k*NNDD (k = 1) | 452 | **0.9947** |

As curvas ROC apresentadas na figura 5.11 mostram que na escolha de um bom ponto operacional (limiar de operação do classificador) o *k*NNDD e o parzen_dd são mais indicados que qualquer outro método, dentre os seis utilizados. o *k*NNDD consegue obter uma taxa por volta de 96% de verdadeiros positivos com 0% de falsos positivos enquanto que o parzen_dd consegue uma taxa de 100% de verdadeiros positivos com uma baixa taxa de falsos positivos, entre 3% e 4%. As curva ROC gerada pelo método kernel *k*NNDDSRM se mostra abaixo das três melhores curvas durante todo o trajeto das curvas. A curva gerada pelo *k*NNDDSRM começa a obter uma taxa razoável de verdadeiros positivos já com uma taxa relativamente elevada de falsos positivos o que inviabiliza o seu uso diante das outras alternativas.

#### 5.2.6.2 Classe 17

A tabela 5.8 apresenta os resultados da base Soybean com as amostras da classe 17 como novidades. Para esses experimentos, a maior AUC foi obtida pelo kernel *k*NNDDSRM, $AUC = 0.7527$. Esse problema se mostrou de bastante difícil solução, tendo em vista os resultados obtidos por todos os métodos. O métodos SVDD, parzen_dd e *k*NNDD tiveram desempen-

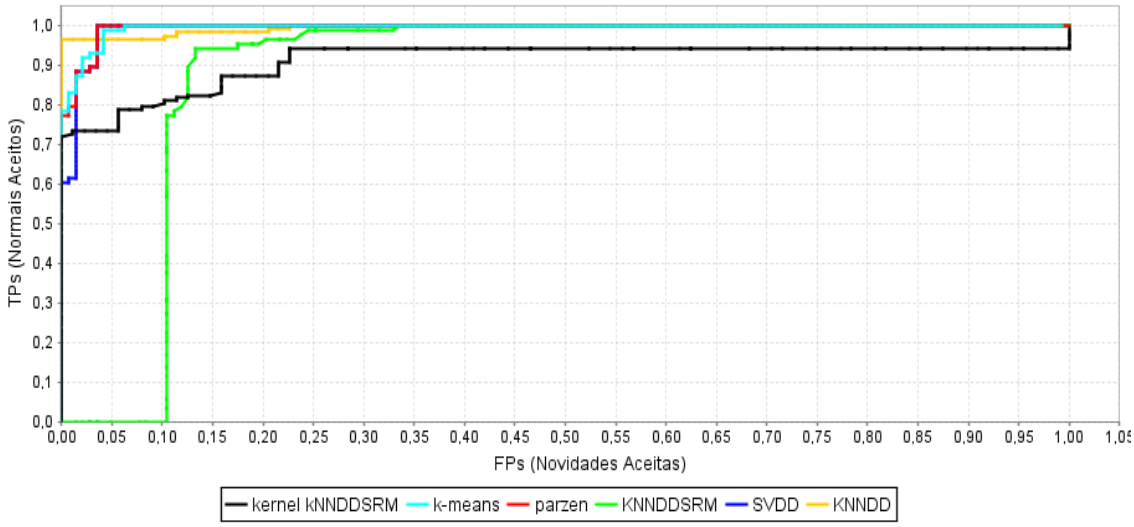


**Figura 5.11** Curvas ROC geradas para a base de dados Soybean com a classe 16 como novidade.

hos similares, porém, consideravelmente inferiores ao melhor resultado. O kmeans_dd obteve o pior desempenho, para um $k = 32$, ficando 0.2294 abaixo do kernel *k*NNDDSRM e bem próximo de um resultado aleatório, ao acaso, que seria uma AUC de 0.5.

**Tabela 5.8** Melhores resultados obtidos por cada método aplicado à base Soybean com a classe 17 como novidade.

| Método | Quantidade de Protótipos | AUC |
|---|---|---|
| *k*NNDDSRM (fracrej = 19, k = 5) | 159 | 0.5943 |
| kernel *k*NNDDSRM (fracrej = 7, k = 4, $\sigma$ = 1.5) | 138 | **0.7527** |
| parzen_dd (h = 0.1230) | 442 | 0.6676 |
| kmeans_dd (k = 32) | 32 | 0.5233 |
| SVDD (fracrej = 5, $\sigma$ = 20) | 417 | 0.6687 |
| *k*NNDD (k = 1) | 442 | 0.6648 |

Analisando as curvas ROC dos métodos SVDD e parzen_dd, mostradas na figura 5.12, vê-se que as curvas se comportam de forma similar durante todo o comprimento das mesmas. Essa similaridade indica um alto grau de similaridade no desempenho dos métodos para a escolha de pontos operacionais, dos classificadores, localizados em coordenadas próximas no gráfico da curva ROC. Como visto na tabela 5.8, o método kernel *k*NNDDSRM é o que obtém o melhor trade-off *verdadeiros positivos* x *falsos positivos*, porém ainda muito aquém do que se considera um bom desempenho.

### 5.2.7 Experimentos com a Base Iris

A base de dados Iris foi introduzida por R.A. Fisher [Fis36] e consiste na classificação de exemplos em três tipos de flores Iris, da família das Iridáceas: Iris Setosa, Iris Versicolour e Iris Virginica. Cada amostra é representada por quatro diferente atributos:

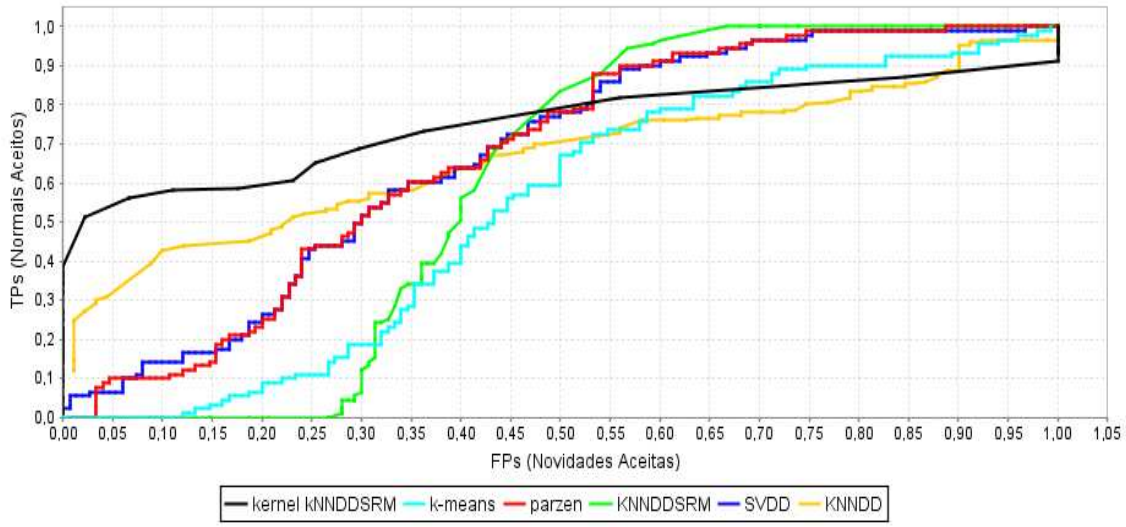


**Figura 5.12** Curvas ROC geradas para a base de dados Soybean com a classe 17 como novidade.

1. comprimento da sépala;
2. largura da sépala;
3. comprimento da pétala; e
4. largura da pétala.

Essa base de dados consiste de 150 amostras, sendo 50 de cada classe. O particionamento do conjunto de treinamento foi feito de forma que quando um classe fosse eleita novidade, todas as amostras dessa classe seriam colocadas no conjunto de teste e as outras classes seriam divididas em partes iguais para o conjunto de teste e treinamento. Dessa forma temos sempre 50 amostras no conjunto de treinamento e 100 amostras para teste. Uma peculiaridade dessa base, pouco comum em amostras do mundo real, é o fato de a classe Iris Setosa ser linearmente separável das demais.

#### 5.2.7.1 Experimentos com a classe Iris Setosa como novidade

Analisando a tabela 5.9 vemos que todos os métodos utilizados obtiveram AUCs elevadas para esse problema. O parzen_dd e o *k*NNDD alcançaram a maior AUC possível indicando uma discriminação perfeita da classe Iris Setosa em relação às outras duas classes. Em termos de AUC o segundo melhor resultado ficou com o *k*NNDDSRM, que diferentemente dos métodos que obtiveram AUC maior, precisou armazenar apenas 11 das 50 amostras no conjunto de treinamento. Os métodos com menor AUC obtiveram ainda uma AUC bastante elevada em relação à maioria dos problemas abordados até o momento. O kmeans_dd conseguiu uma AUC bastante elevada, 0.98, com o uso de apenas 1 protótipo. O SVDD e o kernel *k*NNDDSRM necessitaram do armazenamento de 4 e 17 amostras, respectivamente.

Pela figura 5.13 vemos que apenas a curva ROC do método *k*NNDDSRM não inicia seu percurso da coordenada (0,1) do gráfico. Qualquer dos cinco métodos, que não o *k*NNDDSRM,

**Tabela 5.9** Melhores resultados obtidos por cada método aplicado à base Iris com a classe 1 como novidade.

| Método | Quantidade de Protótipos | AUC |
|---|---|---|
| *k*NNDDSRM (fracrej = 10, k = 5) | 11 | 0.9968 |
| kernel *k*NNDDSRM (fracrej = 20, k = 2, $\sigma$ = 0.5) | 17 | 0.9800 |
| parzen_dd (h = 0.2805) | 50 | **1.0** |
| kmeans_dd (k = 1) | 1 | 0.9800 |
| SVDD (fracrej = 5, $\sigma$ = 5) | 4 | 0.9800 |
| *k*NNDD (k = 1) | 50 | **1.0** |

consegue uma taxa de acertos na classe normal de 100% com nenhuma aceitação de objetos da classe negativa, falsos positivos. Como o objetivo da aplicação destes métodos é a detecção de novidades, qualquer dos seis métodos consiste em uma boa escolha pra essa aplicação, sendo que apenas o *k*NNDDSRM não é um escolha ótima. O fato de apenas o parzen_dd e o *k*NNDD alcançarem AUC = 1 não é decisivo na escolha de um bom detector já que mais importante que uma AUC = 1 é o fato de a curva ROC começar da coordenada (0,1), indicando uma separação perfeita entre a classe normal e novidade.

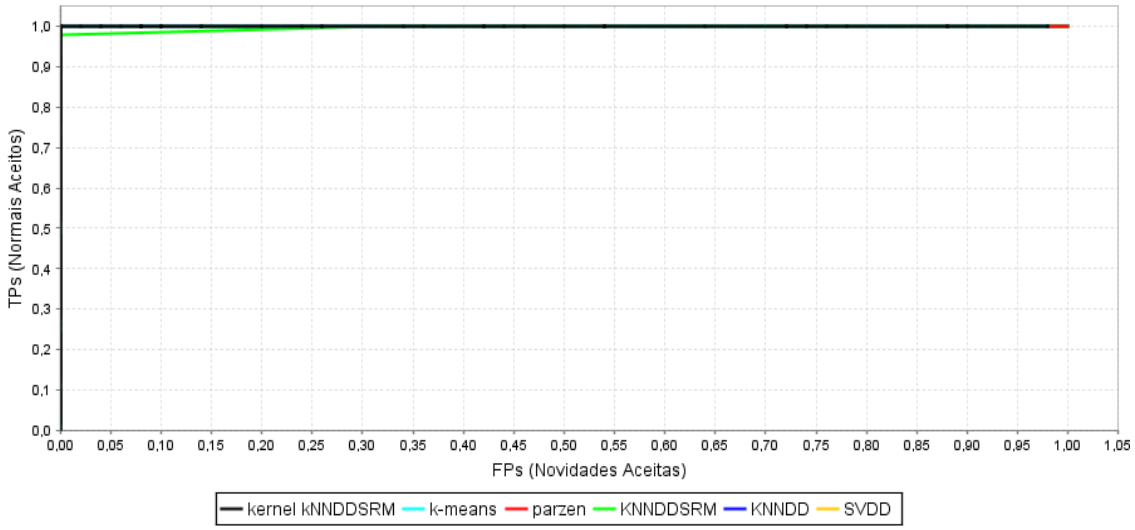


**Figura 5.13** Curvas ROC geradas para a base de dados Iris com a classe Iris Setosa como novidade.

#### 5.2.7.2 Experimentos com a classe Iris Versicolour como novidade

A tabela 5.10 mostra o melhor desempenho obtido pelo método parzen_dd, 0.9852, seguido pelo método *k*NNDD, 0.9808. O método kernel *k*NNDDSRM também alcançou um resultado satisfatório, ficando 0.0199 abaixo do parzen_dd, porém, armazenando apenas 16 protótipos para a descrição dos dados. Nos experimentos com a classe Iris Versicolour como novidade, a surpresa ficou por conta do desempenho do método SVDD que obteve uma AUC irrisória, apenas 0.0896.

**Tabela 5.10** Melhores resultados obtidos por cada método aplicado à base Iris com a classe 2 como novidade.

| Método | Quantidade de Protótipos | AUC |
|---|---|---|
| *k*NNDDSRM (fracrej = 16, k = 3) | 17 | 0.9312 |
| kernel *k*NNDDSRM (fracrej = 16, k = 2, $\sigma$ = 2.0) | 16 | 0.9653 |
| parzen_dd (h = 0.2635) | 50 | **0.9852** |
| kmeans_dd (k = 4) | 4 | 0.9196 |
| SVDD (fracrej = 8, $\sigma$ = 5) | 4 | 0.0896 |
| *k*NNDD (k = 1) | 50 | 0.9808 |

A figura 5.14 mostra as curvas ROC geradas pelos métodos para a classe Iris Versicolour. Inicialmente, a curva gerada pelo método parzen_dd se destaca sobre as outras conseguindo uma alta taxa de verdadeiros positivos e baixa taxa de falsos positivos, porém, quando o ponto operacional é escolhido de maneira que a taxa de falsos positivos seja maior que 5%, de acordo com a curva ROC apresentada na figura 5.14, vemos que o *k*NNDDSRM consegue um melhor desempenho (aumentando consideravelmente a taxa de verdadeiros positivos). O kernel *k*NNDDSRM obtém um rendimento razoável, segundo sua ROC, porém sempre abaixo das melhores curvas. A curva gerada pelo SVDD é bem atípica e mostra um método que se comportou muito pior que o acaso.

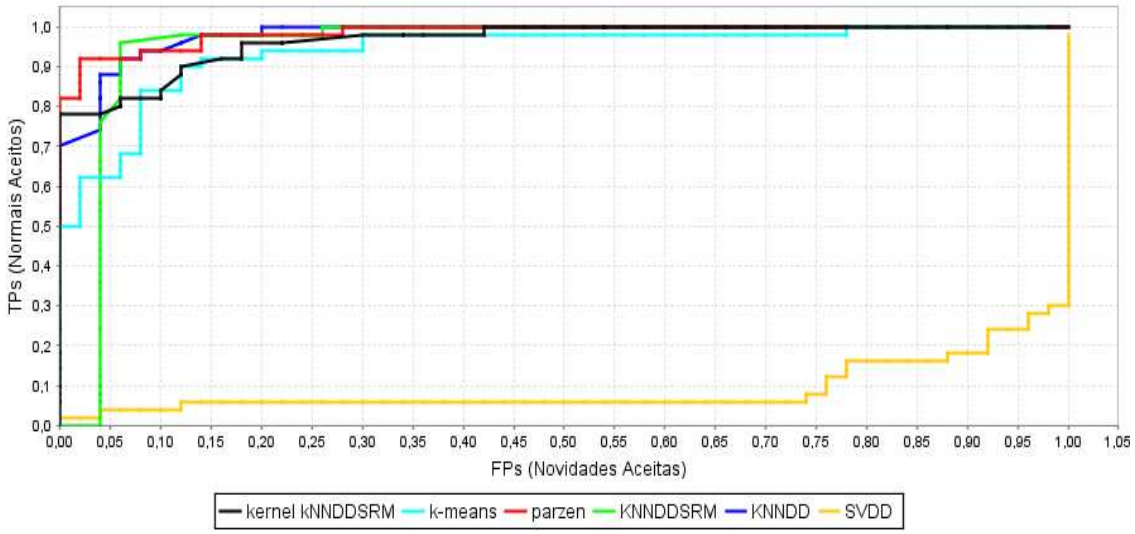


**Figura 5.14** Curvas ROC geradas para a base de dados Iris com a classe Iris Versicolour como novidade.

#### 5.2.7.3 Experimentos com a classe Iris Virgínica como novidade

A tabela 5.11 mostra que o método kernel *k*NNDDSRM obteve a maior AUC, usando apenas 19 amostras da base de treinamento como protótipos, seguido pelo método parzen_dd, que necessitou da base de treinamento completa. Além dos métodos que resultaram nas duas maiores AUCs (kernel *k*NNDDSRM e parzen_dd), os outros métodos obtiveram AUCs na casa de 0.96,

o que não indica, diretamente, um baixo rendimento do classificador perante as maiores AUCs.

**Tabela 5.11** Melhores resultados obtidos por cada método aplicado à base Iris com a classe Virginica como novidade.

| Método | Quantidade de Protótipos | AUC |
|---|---|---|
| *k*NNDDSRM (fracrej = 12, k = 2) | 21 | 0.9636 |
| kernel *k*NNDDSRM (fracrej = 16, k = 2, $\sigma$ = 1.5) | 19 | **0.9830** |
| parzen_dd (h = 0.2215) | 50 | 0.9820 |
| kmeans_dd (k = 1) | 1 | 0.9692 |
| SVDD (fracrej = 14, $\sigma$ = 10) | 8 | 0.9660 |
| *k*NNDD (k = 1) | 50 | 0.9672 |

Segundo a figura 5.15 e o critério de avaliação que visa obter o maior número de verdadeiros positivos com o menor número de falsos positivos, vemos que os métodos parzen_dd, SVDD e kmenas_dd, foram os que melhor se ajustaram a esse critério. O método kernel *k*NNDDSRM, mesmo obtendo a maior AUC, em sua aplicação não teria um rendimento melhor que um desses três métodos citados. Isso acontece pelo fato de em alguns casos a curva ROC não ser completa até a coordenada (1,1), isso faz com que a área abaixo da curva seja menor. Por esse fato a escolha do classificador deve ser feita após a análise da curva ROC, e não apenas de sua AUC.

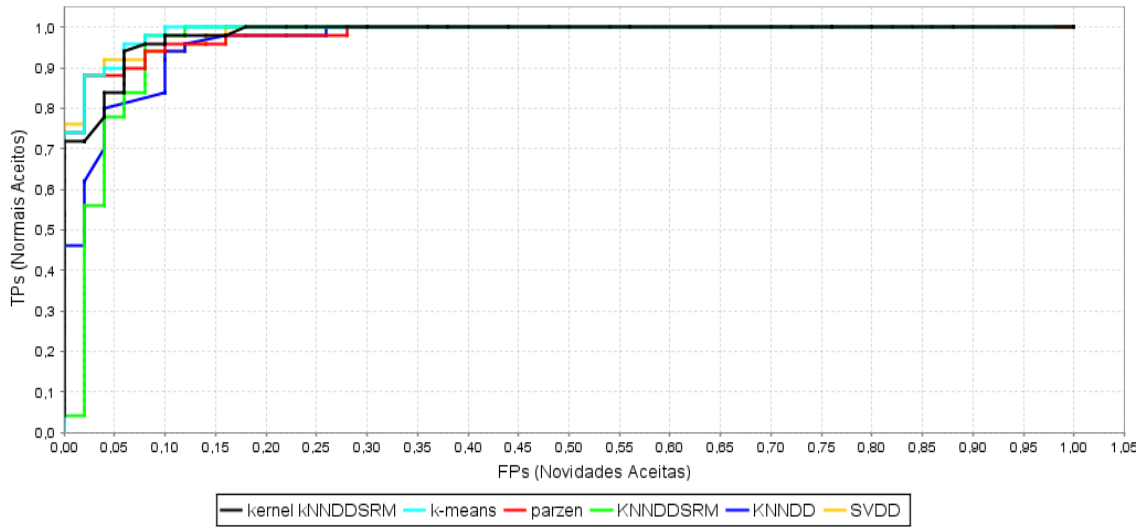


**Figura 5.15** Curvas ROC geradas para a base de dados Iris Virgínica como novidade.

## 5.3 Influência dos Parâmetros Ajustáveis do Método kernel *k*NNDDSRM

Dois tipos de parâmetros são comuns a todos os métodos para classificação com exemplos de uma única classe. O primeiro tipo de parâmetro é otimizado pela rotina de otimização

de cada método durante a minimização de uma função de erro especifica do método. Esse parâmetros incluem, por exemplo, a localização dos protótipos no métodos k-means ou a escolha dos vetores de suporte no SVDD. Pela otimização automática, a escolha desse tipo de parâmetro não importa ao usuário. O segundo tipo de parâmetros são chamados de "parâmetros mágicos"[Tax01] que devem ser ajustados antes do treinamento do classificador pelo usuário. Exemplos do segundo tipo de parâmetro são o $k$ do k-means e o $\sigma$ do kernel RBF.

Diferentemente do primeiro tipo de parâmetros os valores dos parâmetros mágicos devem ser ajustados pelo usuário e escolhas erradas podem ter uma grande influência no desempenho dos métodos. De preferência, esses valores devem ser ajustados de acordo com os dados. Por exemplo, quando um número insuficiente de protótipos é usado no k-means, ele não pode gerar uma descrição satisfatória e a aproximação dos dados será feita de forma grosseira resultando em um baixo desempenho na classificação. Por outro lado, o uso de muitos protótipos no k-means pode resultar em overfitting - quando o método decora o conjunto de treinamento.

Nessa seção iremos discutir a influência dos parâmetros $k$, $\sigma$ do kernel RBF, $d$ (grau de polinomialidade do kernel polinomial), função de kernel e *fracrej* no desempenho do método kernel kNNDDSRM.

### 5.3.1 Análise do Parâmetro *k*

A fase de teste do método kernel *k*NNDDSRM é, basicamente, a mesma do método *k*NNDDSRM. São levadas em consideração as $k$ amostras do Conjunto de Protótipos ($CP$) e as $k$ amostras do Conjunto de Rejeitados ($CR$). A figura 5.16 a) ilustra um exemplo de classificação de um exemplo de teste localizado em uma determinada região da base de dados Distribuições Gaussianas (seção 5.2.1). Nesse exemplo vemos que, de acordo com o capítulo 4, a amostra de teste (preenchida com a cor verde) é classificada como normal para um $k$ igual a 1 , pois a amostra do conjunto $CP$ está mais proxima do objeto de teste que a amostra mais proxima do conjunto $CR$.

No exemplo da figura 5.16 b), levamos em consideração 3 amostras de cada conjunto, $CR$ e $CP$, $k = 3$. Dessa forma vemos que o objeto é classificado como novidade pois há uma maior ocorrência de objetos da classe novidade na região. Sendo $CP_1$, $CP_2$ e $CP_3$ as distâncias Euclideanas das primeira, segunda e terceira amostras em $CP$ mais próximas ao objeto de teste. Sendo $CR_1$, $CR_2$ e $CR_3$ as distâncias Euclideanas das primeira, segunda e terceira amostras em $CR$ mais próximas ao objeto de teste. Como $k = 3$, temos então que $\frac{CP_1}{CR_1} \leq 1, \frac{CP_2}{CR_2} > 1$ e $\frac{CP_3}{CR_3} > 1$, o que classifica a amostra de teste como novidade (ver seção 4.1.2).

Aumentando o valor do parâmetro $k$, tornamos o classificador mais sensível à região onde a amostra de teste está inserida. A escolha do $k$ deve ser feita cuidadosamente pois à medida que esse parâmetro é aumentado um maior custo computacional é requerido pois para a execução do teste em um objeto o *k*NNDDSRM e o kernel *k*NNDDSRM requerem $k$ comparações no $CR$ e $CP$. O aumento do valor do $k$ pode levar o classificador a considerar regiões já distantes do contexto onde a amostra de teste está inserida.

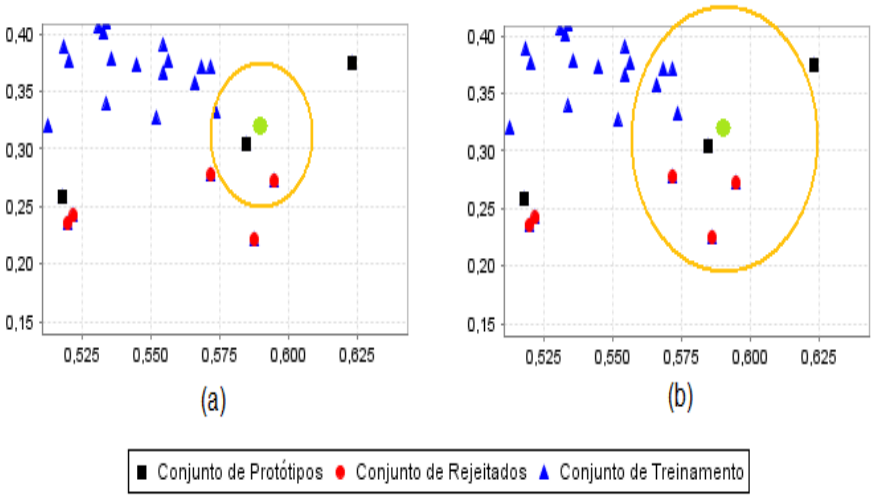


**Figura 5.16** Influência do parâmetro *k* na classificação. a) k = 1; b) k = 3.

### 5.3.2 Análise do Parâmetro $\sigma$ e *d* no kernel *k*NNDDSRM

O parâmetro $\sigma$ da função de base radial adotada, no nosso caso a função Gaussiana, consiste na largura da curva Gaussiana. O parâmetro *d* do kernel polinomial consiste no grau de polinomialidade da função polinomial. Os valores do $\sigma$ e do *d*, interferem no desempenho do kernel *k*NNDDSRM, porém, não existe uma regra clara que explique a interferência desses parâmetros no desempenho do método. Alterando-se o valor do $\sigma$ e do *d* pode-se chegar a saídas iguais das RBFs ou da função polinomial entre dois ou mais amostras no treinamento. Essa igualdade pode afetar a ordenação das amostras no passo anterior à criação dos conjunto de protótipos e conjunto de rejeitados, gerando conjuntos de rejeitados diferentes o que implica na criação de conjunto de protótipos de forma a zerar o erro no conjunto de treinamento dado o conjunto pré-definido de amostras rejeitadas.

A figura 5.17 mostra um exemplo onde duas diferentes descrições foram geradas, porém, apenas o parâmetro $\sigma$ do kernel RBF foi alterado, de maneira arbitrária. Nesse caso um conjunto de rejeitados diferente foi alcançado nas duas descrições forçando uma diferente escolha das amostras para a composição do conjunto de protótipos. Os círculos apresentados na figura superior ilustram pontos onde houve diferenças na escolha das amostras tanto em *CR* como em *CP*.

### 5.3.3 Análise do Parâmetro Função de Kernel

Diferentes funções de kernel geram diferentes descrições. A simples distância Euclidiana já se mostrou ineficiente no caso de descrições mais complexas, bases distribuição Gaussiana e Banana (ver seções 5.2.1 e 5.2.2). Nesse trabalho, a critério de comparação, nós utilizamos apenas o kernel RBF no método kernel *k*NNDDSRM. Outras funções de kernel foram testadas e três dos problemas anteriores foram escolhidos para uma análise de desempenho no uso de diferentes funções no kernel *k*NNDDSRM.

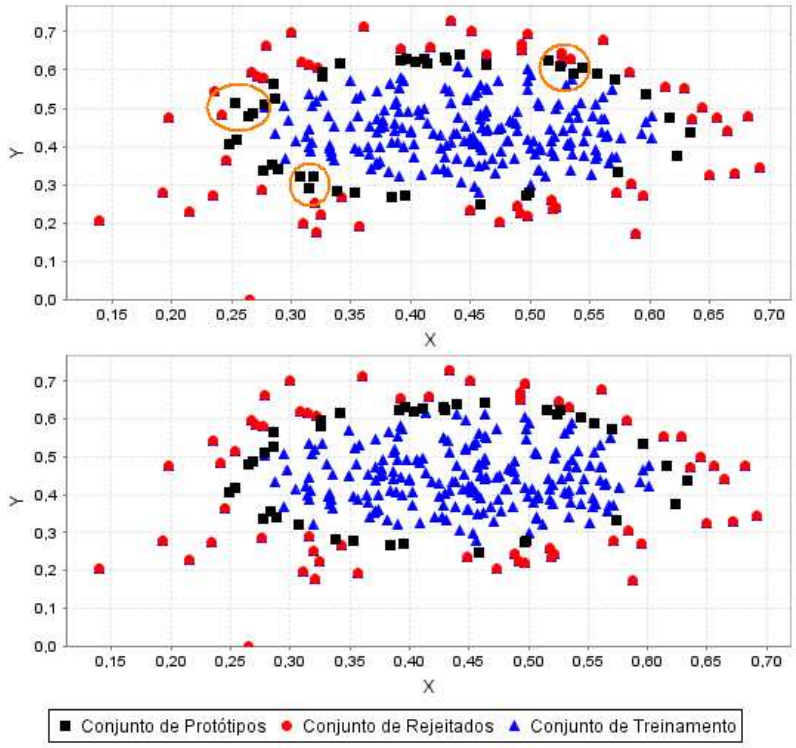


**Figura 5.17** Influência do parâmetro $\sigma$ na classificação.

A figura 5.18 mostra os desempenhos das funções de kernel RBF, polinomial e cosseno. Os experimentos com o kernel RBF tiveram o sigma variando de acordo com os valores [0.5 1.0 1.5 2.0] enquanto que o grau de polinomialidade do kernel polinomial variou de 1 a 5. O *k* e o fracrej variaram de 1 a 5 e 5 a 40 respectivamente. De acordo com a figura 5.18, o kernel RBF foi melhor em todos os experimentos, obtendo uma discreta diferença em relação ao kernel polinomial na aplicação à base Biomed, nas outras duas bases a diferença para o kernel polinomial foi um pouco maior. O kernel cosseno não obteve resultado satisfatório; seu desempenho foi consideravelmente inferior ao das outras duas funções de kernel utilizadas em todos os três experimentos realizados.

### 5.3.4 Análise do Parâmetro *fracrej*

A idéia básica do *k*NNDDSRM e kernel *k*NNDDSRM é de que as amostras do conjunto de treinamento mais distantes do centro de massa, ou em uma região de menor densidade devem ser classificadas como novidade. Assim como vários métodos de mesma natureza, apenas as amostras situadas nas regiões de conflito devem ser armazenadas. Um problema que existe nessa abordagem consiste no não conhecimento *a-priori* da região de fronteira entre a classe novidade e a classe normal, em problemas com classificação com exemplos de uma única classe não se tem nenhum conhecimento de como se apresenta a classe novidade. Dessa forma, devemos rejeitar uma pequena porcentagem de amostras do treinamento para a criação do *CR* (conjunto de rejeitados) de maneira que se descreva bem os dados normais, mas, sem englobar

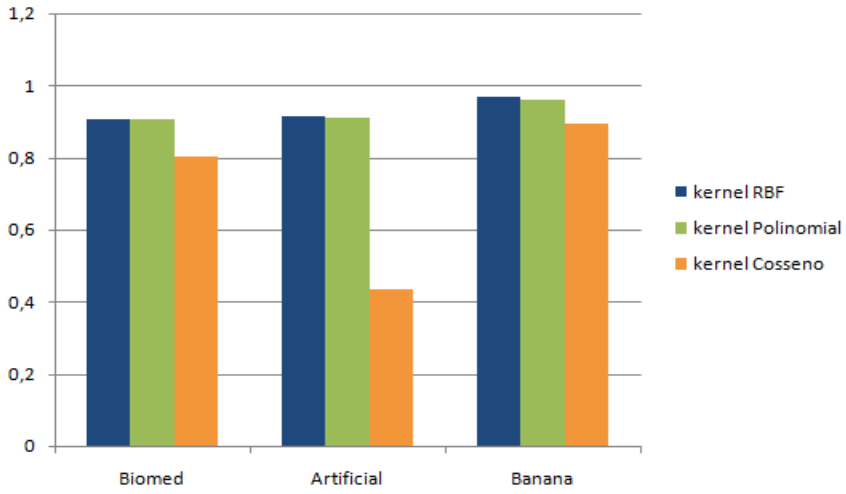


**Figura 5.18** Variação da AUC com diferentes funções de kernel.

de forma significativa elementos mais próximos ao interior da representação da classe normal.

A figura 5.19 mostra o mesmo modelo para classificação com apenas o parâmetro *fracrej* alterado. A figura da esquerda tem um menor percentual para o *fracrej* (*fracrej* = 5%) que a da direita (*fracrej* = 25%). Na figura da direita vemos que o conjunto de rejeitados armazena amostras que visualmente podem ser classificadas como normal o que pode acarretar em uma grande taxa de erro no teste do classificador.

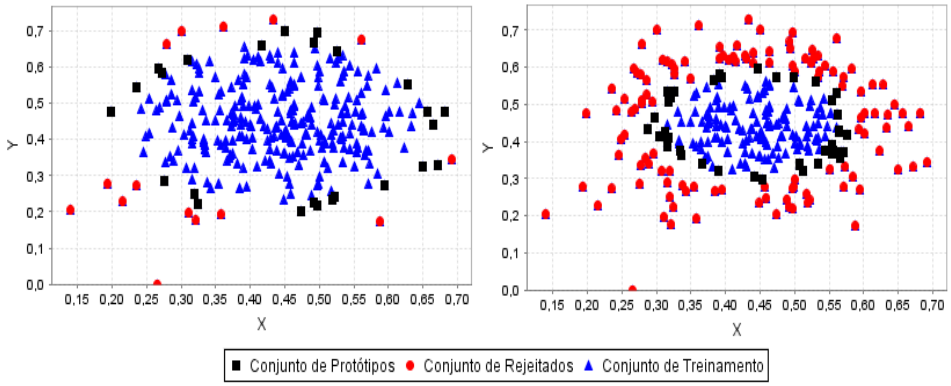


**Figura 5.19** Influência do parâmetro $fracrej$ na classificação. A figura da esquerda tem *fracrej* = a 5%, enquanto que a da direita tem *fracrej* = 25%.

## 5.4 Detecção de novidades em Séries Temporais

Foram realizados experimentos com seis séries temporais formadas tanto por dados sintéticos quanto por dados reais. Quatro das seis séries (Space Shuttle Marotta Valve Series, Respira-

tion, $ECG_1$ e $ECG_2$) foram obtidas em [Keo05], a série Laser pode ser obtida em http://www-psych.stanford.edu/ andreas/Time-Series/SantaFe.html e a série seno pode ser gerada de forma artificial. As quatro primeiras séries são representadas por um conjunto de dados considerado grande (mais de 2000 amostras para o treinamento) e que estados podem ser identificados de forma visual. As duas últimas séries são representadas por um pequeno conjunto de treinamento (menos que 1000 amostras) e não têm estados tão bem definidos quanto as quatro primeiras. Com base nas características das séries as seis bases foram divididas da seguinte forma, de acordo com as abordagens adotadas para a detecção de novidades:

- **Subseção 5.4.1**: Bases de dados Respiration, Space Shuttle Marotta Valve Series e Seno;
- **Subseção 5.4.2**: Bases de dados Seno e Laser; e
- **Subseção 5.4.3**: Bases de dados $ECG_1$, $ECG_2$ e Space Shuttle Marotta Valve Series.

Essa divisão se dá pelo fato de, no primeiro conjunto de experimentos, nenhum dos métodos utilizados terem obtido resultados satisfatórios para as bases $ECG_1$ e $ECG_2$. A série Laser foi escolhida apenas para validar os resultados dos experimentos da segunda abordagem (subseção 5.4.2) com os resultados obtidos em [MP03]. Os experimentos realizados na subseção 5.4.2 não se comportam bem em bases com um grande número de amostras, pois, nesse caso, devem ser criados vários classificadores simultaneamente; isso implica em um alto custo computacional quando essa abordagem é aplicada a bases com um grande número de amostras. A subseção 5.4.3 trata problemas onde os estados da série são bem definidos e novidades ocorrem em pontos não intuitivos da série (podendo ocorrer dentro da descrição normal dos dados). Nesse caso, foram escolhidas as séries em que os outros métodos não obtiveram bons resultados ($ECG_1$ e $ECG_2$), com exceção da série Space Shuttle Marotta Valve Series onde as abordagens do primeiro e terceiro conjunto de experimentos obtiveram bons resultados.

O primeiro conjunto de experimentos (ver subseção 5.4.1) compara o desempenho do kernel *k*NNDDSRM e do *k*NNDDSRM com os métodos one-class SVM e *k*NNDD. A subseção 5.4.1 tem como finalidade, também, demonstrar a eficiência dos métodos *k*NNDDSRM e kernel *k*NNDDSRM aplicados ao problema da detecção de novidades em séries temporais.

A subseção 5.4.2 reporta experimentos com séries codificadas em várias janelas de tempo de forma simultânea. Nessa abordagem, vários classificadores *k*NNDDSRM são criados, um para cada janela de tempo em que a série foi codificada, e os resultados de cada classificador são cruzados de forma a obter um resultado final mais robusto. Essa abordagem foi introduzida por Perkins em [MP03]. A subseção 5.4.3 apresenta os resultados obtidos pelo método para detecção de novidades baseado na transição de estados entre as séries.

### 5.4.1 Comparação dos Resultados Obtidos pelos Métodos *k*NNDDSRM e kernel *k*NNDDSRM com os Resultados dos Métodos One-class SVM e *k*NNDD

O *k*NNDDSRM e o *k*NNDD não fazem uso do kernel, já o kernel *k*NNDDSRM e One-class SVM têm o kernel RBF como padrão. O One-class SVM foi modificado de forma a utilizar a simples distância Euclidiana ao invés do kernel, no intuito de uma comparação justa com os métodos *k*NNDDSRM e *k*NNDD. Essa modificação foi feita na implementação da biblioteca

LibSVM, utilizada neste trabalho. Os métodos são comparados em termos de AUC e em termos de número de falsos alarmes ou falsos negativos (amostras da classe normal detectadas como novidade) contra o número de detecções ou verdadeiros negativos (novidades corretamente detectadas). Nossa meta é, sempre, construir um classificador com exemplos de uma única classe capaz de corretamente detectar novidades sem erroneamente classificar amostras da classe normal como novidade. Dessa maneira, o critério de avaliação de desempenho adotado nesse trabalho é o seguinte: menor número possível de ocorrências de falsos alarmes (FN) de uma forma que cada região com com ocorrências de novidades, na série temporal, tenha pelo menos uma ocorrência corretamente detectada como novidade. Os parâmetros dos classificadores foram ajustados de forma a se alcançar da melhor forma possível o critério citado (no conjunto de teste) e, também, buscando armazenar o menor número possível de amostras para a descrição dos dados. O classificador em si, só é obtido quando um limiar (threshold) apropriado é escolhido. Para nossos experimentos, os limiares para cada método utilizado (kernel *k*NNDDSRM (RBF), *k*NNDDSRM, *k*NNDD, *one-class* SVM (RBF) e *one-class* SVM (dist. Euclidiana)) em cada problema diferente, foram escolhidos apenas nos casos em que cada classificador obtive sua maior AUC para aquele determinado problema.

#### 5.4.1.1 Experimentos com a série Respiração

Na série temporal Respiração, ilustrada na figura 5.20, a novidade se apresenta a partir do registro 3015. Antes desse instante o paciente está dormindo e nesse instante ele respira profundamente e abre os olhos. Essa é uma novidade óbvia que pode ser facilmente detectada de forma visual. Nós a usamos como primeiro teste de nosso método. Nós usamos apenas uma porção da série disponível na Eamonn's Collection [Keo05]. Em nossos experimentos nós usamos uma subseqüência da série original do registro 17500 ao registro 20900 (como mostra a figura 5.20).

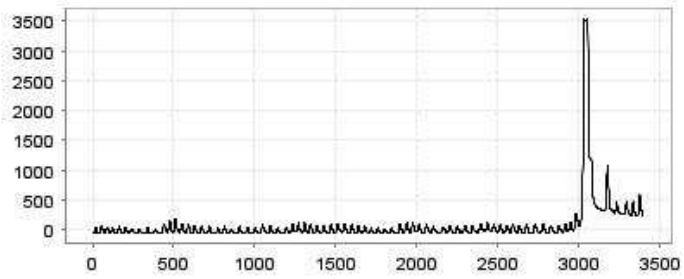


**Figura 5.20** Base de dados Respiração de um paciente original.

Nos experimentos com a série temporal Respiração, da mesma maneira que em [OdLM06], nós usamos uma janela de tempo *w=12*. O conjunto de treinamento contem 989 amostras representando a classe normal da amostra 12 até a amostra 1000. O conjunto de teste contém 847 amostras representando tanto a classe normal quanto a classe novidade.

A tabela 5.12 mostra os resultados dos métodos para a série Respiração. Para essa série temporal, todos os métodos obtiveram um bom desempenho, porém o *k*NNDDSRM precisou armazenar apenas 8.39% do total de amostras do conjunto de treinamento. Os experimentos

realizados usando apenas a distância Euclidiana resultaram em um desempenho similar aos resultados obtidos com a aplicação do kernel RBF.

**Tabela 5.12** Resultados obtidos para a série temporal Respiração

| método | *fracrej* | $k$ | $\sigma$ | $\theta$ | CT%[1] | AUC |
|---|---|---|---|---|---|---|
| kernel $k$NNDDSRM | 10 | 1 | 4 | 1.000203 | 14.05 | 0.9958 |
| $k$NNDDSRM | 6 | 1 | – | 1.000245 | 8.39 | 0.9952 |
| $k$NNDD | – | 1 | – | 0.99975 | 100 | 0.9995 |
| one-class SVM (RBF) | 22 | – | 1 | -1.7 | 21.94 | 0.9978 |
| one-class SVM (dist Euclid.) | 22 | – | – | -8.0 | 22.55 | 0.9980 |

A figura 5.21 mostra os resultados da aplicação de cada método à série Respiração. Para esta base de dados todos os métodos obtiveram resultados similares, porém o $k$NNDDSRM obteve a menor taxa de armazenamento, de acordo com a tabela 5.12. Analisando a figura 5.21 vemos que os resultados foram bastante satisfatórios; alcançamos uma alta taxa de verdadeiros negativos e verdadeiros positivos e uma baixa taxa de falsos positivos.

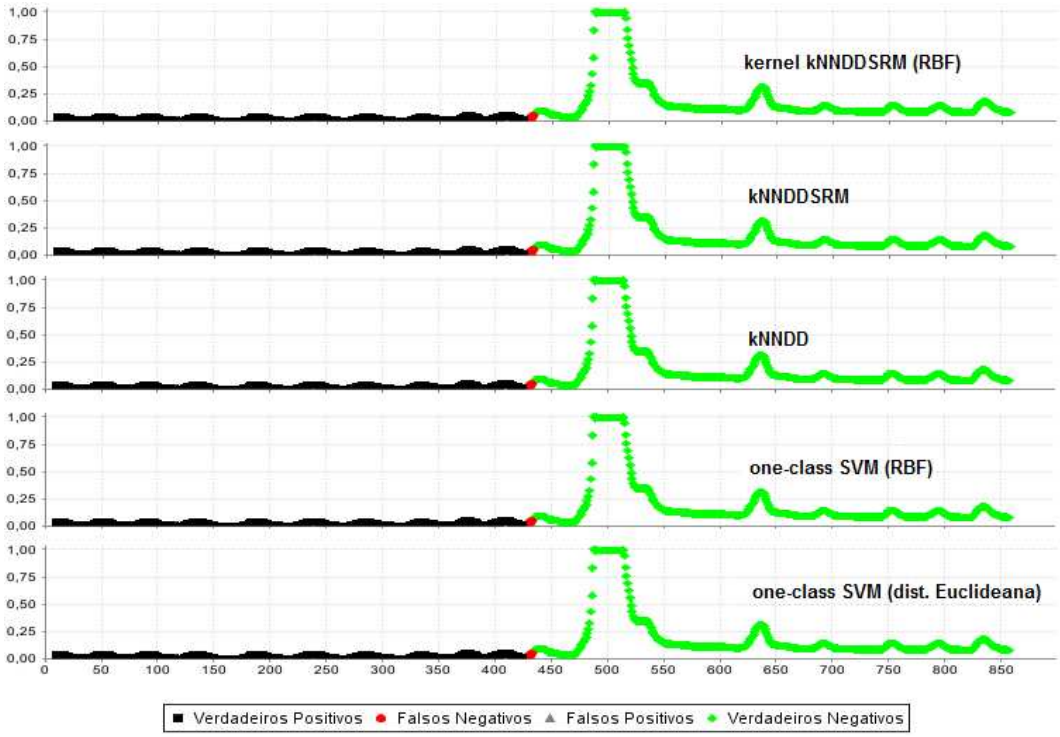


**Figura 5.21** Resultados dos testes com a base Respiração usando os $\theta$ da tabela 5.12.

[1]Percentual do total de amostras do conjunto de treinamento utilizadas como protótipos pelo classificador após a fase de treinamento

#### 5.4.1.2 Experimentos com a série Space Shuttle Marotta Valve Series

Na série temporal Space Shuttle Marotta Valve Series, ilustrada na figura 5.22, a novidade se apresenta entre a amostra 4254 e a amostra 4341. Esse é um caso em que a novidade se apresenta de forma mais sutil que na série temporal Respiração.

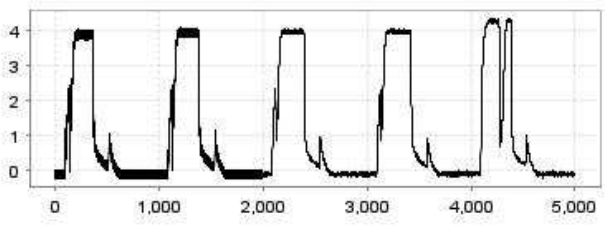


**Figura 5.22** Série temporal Space Shuttle Marotta Valve original.

Para os experimentos com a série Space Shuttle Marotta Valve Series foi usada uma janela de tempo $w = 30$, que foi selecionada empiricamente. Esse valor foi atribuído pelo fato de a série ter uma regularidade evidente e valores menores para a janela de tempo podem não obter resultados satisfatórios. O conjunto de treinamento contem 2720 amostras representando a classe normal do registro coletado de índice 30 até o registro de índice 2749. O conjunto de teste contém 2212 amostras representando tanto a classe novidade quanto a classe normal.

A tabela 5.13 mostra os melhores resultados obtidos para essa série temporal. O kernel *k*NNDDSRM alcançou a maior AUC seguido do one-class SVM, ambos usando o kernel RBF. O *k*NNDD, por sua vez, chegou a um resultado também satisfatório, porém, nem sempre uma boa AUC significa um bom desempenho. O kernel *k*NNDDSRM conseguiu a maior AUC fazendo o uso do menor número de amostras para a construção do modelo, apenas 33.45 % contra 49.92% do one-class SVM utilizando o kernel RBF. O *k*NNDDSRM e o one-class SVM utilizando apenas a distância Euclidiana não obtiveram AUCs satisfatórios, comparando-se com os três melhores resultados.

**Tabela 5.13** Resultados obtidos para a série temporal Space Shuttle Marota Valve Series

| método | *fracrej* | $k$ | $\sigma$ | $\theta$ | CT%[2] | AUC |
|---|---|---|---|---|---|---|
| kernel *k*NNDDSRM | 15 | 1 | 1 | 0.0044 | 33.45 | 0.9577 |
| *k*NNDDSRM | 26 | 1 | – | 0.9782 | 26.80 | 0.7820 |
| *k*NNDD | – | 1 | – | 0.9935 | 100 | 0.9407 |
| one-class SVM (RBF) | 25 | – | 1 | -109 | 49.92 | 0.9561 |
| one-class SVM (dist Euclid.) | 67 | – | – | -1947.14 | 67.09 | 0.8130 |

A figura 5.23 mostra os resultados obtidos para a série Space Shuttle Marotta Valve Series. Os limiares usados são mostrados na tabela 5.13. Na figura 5.23 vemos um desempenho bastante similar dos métodos kernel kNNDDSRM e one-class SVM, no uso do kernel RBF, sendo que o kernel *k*NNDDSRM precisou de 16.47% menos armazenamento de amostras de treinamento que o one-class SVM. Falsos alarmes ocorreram imediatamente antes da correta

[2]Percentual do total de amostras do conjunto de treinamento usado pelo classificador

detecção de novidades, diferentemente dos outros experimentos que obtiveram uma alta taxa de falsos negativos após o ajuste do classificador para que houvesse a incidência mínima de verdadeiros positivos.

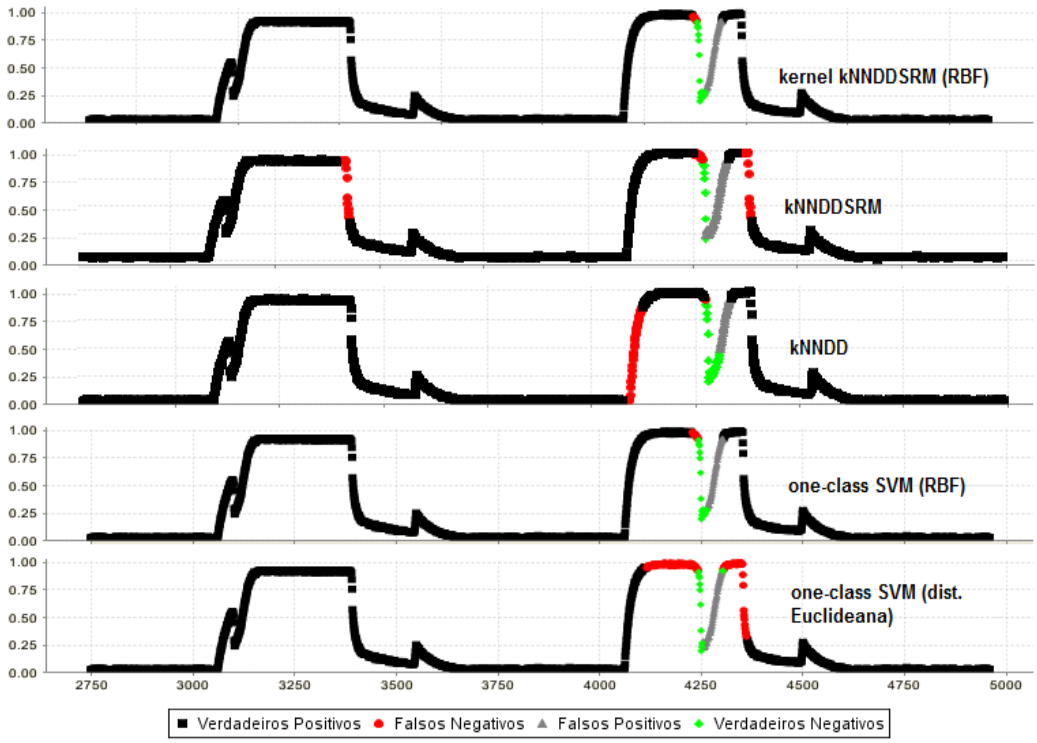


**Figura 5.23** Resultados dos testes com a base Space Shuttle Marotta Valve usando os $\theta$ da tabela 5.13.

#### 5.4.1.3 Experimentos com a série Seno

Duas séries temporais artificiais, $x_1(t)$ e $x_2(t)$ foram geradas, ambas contendo 360 registros. $x_1(t)$ é um sinal senoidal com um leve ruído aditivo enquanto que $x_2(t)$ é o mesmo sinal que $x_1(t)$ exceto que nesse caso, são adicionados dois segmentos com um maior ruído aditivo, o primeiro do registro 200 ao 220 e o segundo, mais discreto, do registro 300 ao 320 (ver figura 5.24). O ruído aditivo foi gerado por um componente aleatório variando de 0 a 1. Para $x_1(t)$ esse componente aleatório foi dividido por 4, o fator de suavização. O primeiro segmento representando a classe novidade na série temporal de teste, $x_2(t)$, não foi suavizado enquanto que o segundo segmento, mais discreto, teve um fator de suavização 2. Essa série temporal também foi usada em [MP03]. A figura 5.24 mostra $x_2(t)$, a série seno gerada para o teste dos métodos. $x_1(t)$ se comporta da mesma maneira que $x_2(t)$ exceto pelos segmentos de maior ruído aditivo.

Para a série seno, o *k*NNDDSRM alcançou o melhor desempenho em termos de menor número de ocorrências de falsos alarmes, mesmo com a menor AUC entre todos os experimentos para essa série. Em contrapartida, os métodos *k*NNDD and one-class SVM obtiveram as maiores AUCs (com menor armazenamento de amostras de treinamento, no caso do one-class

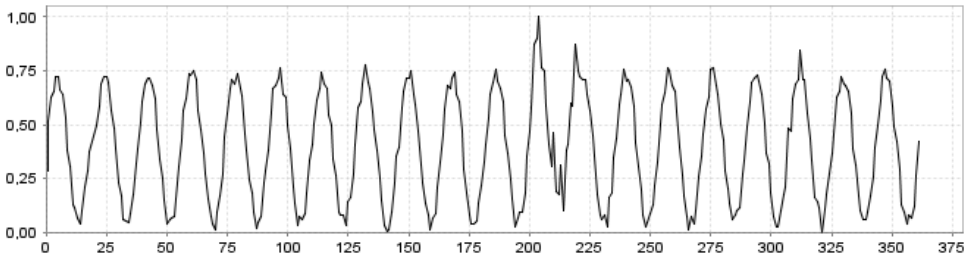


**Figura 5.24** Série temporal Seno (Conjunto de teste).

SVM) mas também com esses métodos houve uma maior incidência de falsos alarmes, como mostram as tabelas 5.14 e 5.15. Dependendo da finalidade da aplicação, um modelo mais sensitivo, mais propício a incidência de falsos negativos, ou falsos alarmes, pode ser de pouca utilidade. A tabela 5.15 mostra o número de ocorrências de cada possível estado para cada método aplicado à série seno. Analisando a tabela 5.15 vê-se um grande número de ocorrências verdadeiras positivas para o método kernel *k*NNDDSRM, também, alcançando uma taxa significante de verdadeiros negativos para as regiões pré-estabelecidas como novidades.

**Tabela 5.14** Resultados obtidos para a série temporal Seno

| método | *fracrej* | $k$ | $\sigma$ | $\theta$ | CT%[2] | AUC |
|---|---|---|---|---|---|---|
| kernel *k*NNDDSRM | 31 | 1 | 4 | 0.999 | 98.00 | 0.7338 |
| *k*NNDDSRM | 19 | 1 | – | 1.0003 | 94.75 | 0.7547 |
| *k*NNDD | – | 1 | – | 0.9935 | 100 | 0.8126 |
| one-class SVM (RBF) | 0.2 | – | 1 | -1 | 40.81 | 0.8004 |
| one-class SVM (dist Euclid.) | 0.1 | – | – | -7.6 | 13.11 | 0.8043 |

**Tabela 5.15** *TN*, *TP*, *FN* and *FP* ocorrências para os métodos aplicados à série temporal Seno

| Método | TP | TN | FP | FN |
|---|---|---|---|---|
| kernel *k*NNDDSRM (RBF) | 283 | 27 | 18 | 15 |
| *k*NNDDSRM (dist. Euclideana) | 263 | 21 | 21 | 38 |
| *k*NNDD | 280 | 21 | 21 | 21 |
| One-class SVM (RBF) | 270 | 40 | 2 | 31 |
| One-class SVM (dist. Euclideana) | 301 | 18 | 24 | 0 |

A figura 5.25 mostra os resultados obtidos de cada método para a série seno. Neste experimento vemos um desempenho mais pobre dos métodos com o uso apenas da distância Euclidiana. O one-class SVM não conseguiu detectar a segunda região da classe novidade, enquanto que o *k*NNDDSRM e o *k*NNDD o fizeram, porém, com um grande número de falsos alarmes. Os melhores desempenhos ficaram por conta do kernel *k*NNDDSRM e do one-class SVM, ambos com o kernel RBF. Para esses dois métodos, vemos que as duas regiões da classe novidade foram detectadas e falsos alarmes ocorreram apenas próximos às regiões com um ruído aditivo mais elevado. O kernel *k*NNDDSRM com kernel RBF, sobressaiu-se sobre o one-class SVM

em número de falsos alarmes, como mostra a figura 5.25 e a tabela 5.15. Para essa base de dados, mesmo obtendo AUCs similares, o uso do kernel RBF é visivelmente mais confiável.

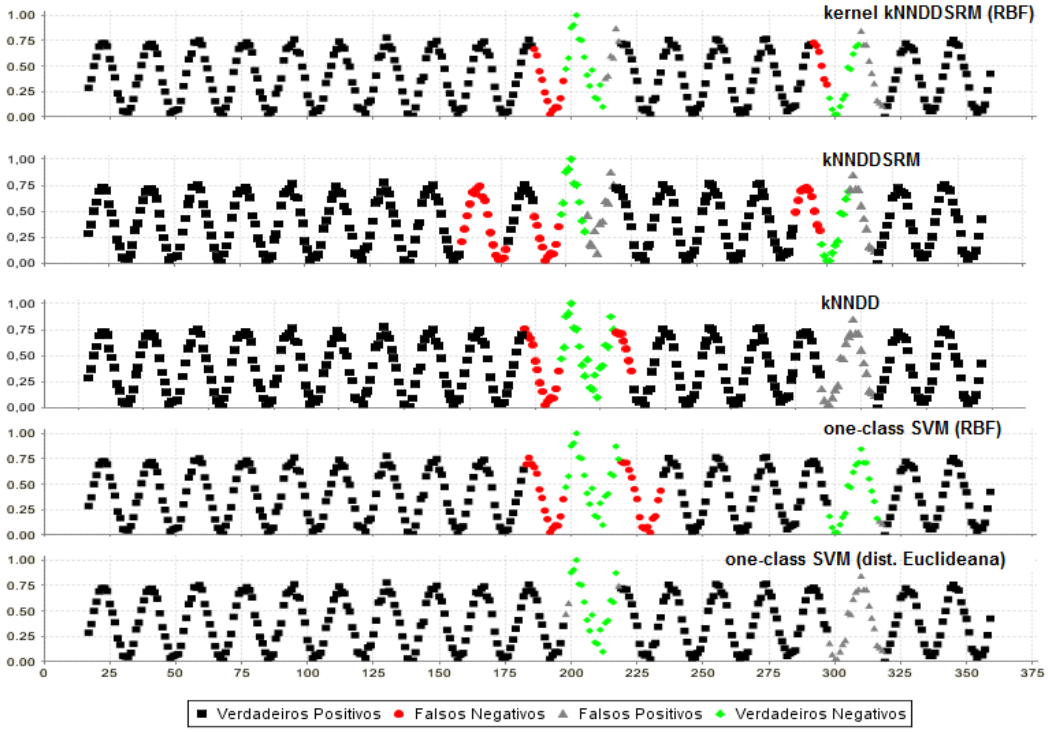


**Figura 5.25** Resultados dos testes com a base Seno usando os $\theta$ da tabela 5.14.

### 5.4.2 Experimentos com bases codificadas em múltiplas dimensões

Outra forma de experimento utilizada na detecção novidades em séries temporais foi o uso de várias janelas simultaneamente durante a classificação [MP03]. Essa abordagem funciona como se vários classificadores, cada um treinado com a mesma série codificada com diferentes janelas, fossem testados na mesma base. Dessa forma, a saída final de uma dada amostra $x(t)$ no tempo consiste no produtório da saída dos classificadores para a mesma amostra no tempo $t$. A figura 5.26 ilustra como acontece a detecção ou aceitação de uma amostra de teste na série temporal. Nela, quatro diferentes janelas são testadas e em apenas dois casos todos os classificadores detectaram simultaneamente uma dada amostra como novidade.

Essa abordagem agrega robustez à classificação, porém, requer um maior poder computacional e pode se tornar inviável para bases de dados com um conjunto de treinamento volumoso. Duas série temporais foram experimentadas com essa abordagem: a série seno e a série, laser, da competição do Instituto Santa Fé [WG94].

| w=3 | w=4 | w=5 | w=6 | | |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | = | normal |
| 1 | 0 | 0 | 1 | = | normal |
| 1 | 1 | 1 | 1 | = | novidade |
| 0 | 1 | 1 | 1 | = | normal |
| 0 | 1 | 1 | 1 | = | normal |
| 0 | 1 | 1 | 1 | = | normal |
| 0 | 0 | 1 | 0 | = | normal |
| 1 | 0 | 0 | 0 | = | normal |
| 1 | 0 | 0 | 1 | = | normal |
| 1 | 1 | 1 | 1 | = | novidade |
| 0 | 1 | 1 | 1 | = | normal |
| 0 | 1 | 1 | 1 | = | normal |
| 0 | 1 | 1 | - | = | - |
| 0 | 0 | - | - | = | - |
| 0 | - | - | - | = | - |

**Figura 5.26** Exemplo de detecção de novidade utilizando várias janelas simultaneamente.

#### 5.4.2.1 Seno

Duas séries temporais artificiais, $x_1(t)$ e $x_2(t)$ foram geradas sendo a de treinamento idêntica à gerada na seção 5.4.1.3 (porém, ambas com 1200 registros) e a de teste contendo apenas 1 pequeno segmento com maior ruído aditivo, equivalente ao segmento com ruído mais discreto da série de teste da seção 5.4.1.3. A geração dessa nova série seno foi necessária para uma comparação justa com o resultado obtido em Junshui Ma e Simon Perkins [MP03].

Para esse experimento comparamos o nosso resultado ao obtido em Junshui Ma e Simon Perkins [MP03] para essa mesma série. Para esse problema o *k*NNDDSRM precisou armazenar 500 (10.5% do total de amostras analisadas) amostras de treinamento, sendo essas amostras codificadas em várias janelas de tempo diferentes. [7, 9, 11, 13] foram as 4 janelas de tempo utilizadas para se obter esse resultado com o parâmetro *fracrej* a 2%. O resultado obtido para essa série temporal, mostrado na figura 5.27, condiz exatamente com o obtido Junshui Ma and Simon Perkins [MP03] utilizando one-class SVM e conjunto de janelas [3, 5, 7, 9, 11, 13, 15, 17, 19], kernel RBF e parâmetro $\nu = 0.2$. Na figura 5.27 a novidade acontece por volta do registro de número 600, um alarme (no formato de um quadrado) está localizado nesse ponto.

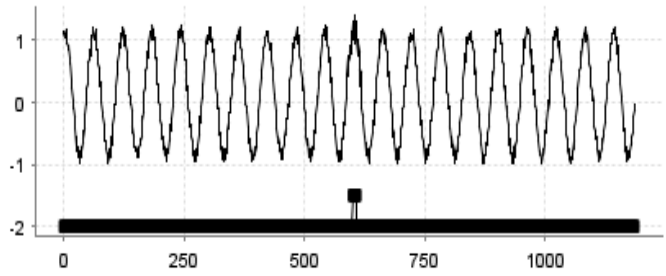


**Figura 5.27** Resultado obtido do do teste para a série Seno utilizando várias janelas simultaneamente.

#### 5.4.2.2 Laser

A série Laser foi coletada a partir das saídas de um *Far-infrared laser*, capaz de penetrar tecidos humanos a uma profundidade de até 10 milímetros, em estado caótico. A base contém 1000 medições. As bases de dados foram criadas de forma que as duzentas primeiras medições fossem usadas para o treinamento e o restante usado para o teste.

Para essa base usamos as janelas [5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25], diferentemente de [MP03] que utilizou as janelas [3, 5, 7, 9, 11, 13, 15, 17, 19]. O nosso método, diferente do método proposto em [MP03], necessitou de um maior número de janelas para conseguir detectar de forma correta as novidades nessa série temporal. Em Junshui Ma e Simon Perkins [MP03], a one-class SVM teve ainda $\nu = 0.2$ e kernel RBF. O *fracrej* utilizado foi de 2% com função RBF e $\sigma = 2$. A descrição final selecionou com 385 amostras dentre os vários conjuntos de treinamento codificados com as janelas de tempo, *w*, citadas.

A figura 5.28 mostra que duas regiões de maior incidência da classe novidade foram identificadas pelo algoritmo. Essas regiões condizem exatamente com as regiões encontradas em [MP03]. Assim como na figura 5.27, as novidades são representadas por uma série logo abaixo da série de teste, onde pontos de valor -2 nessa série significam amostras da classe normal corretamente aceitas e pontos de valor -1.5 significam as novidades detectadas.

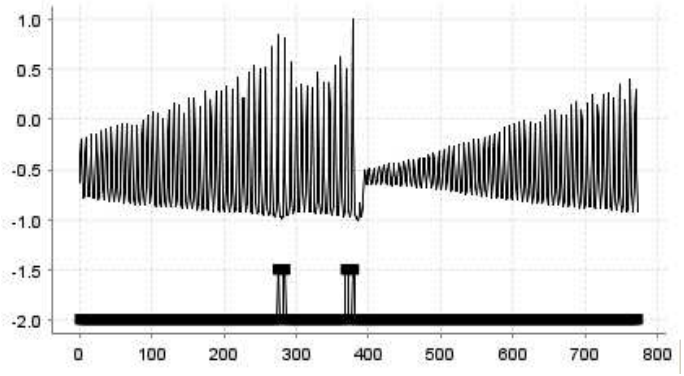


**Figura 5.28** Resultado obtido do do teste para a série Laser utilizando várias janelas simultaneamente.

### 5.4.3 Detecção de Novidades em Séries Temporais - Método baseado na Transição Entre Estados

Para a geração de um bom sistema para detecção de novidades em séries temporais, o número de clusters (estados) da série deve ser escolhido de acordo com características específicas de cada série. A janela em que cada série foi codificada com base no *trade-off* entre custo computacional no treinamento e eficiência na representação da série. Tornar a janela muito grande pode resultar em perda de desempenho na execução, caso ela seja muito pequena pode ser que ela não represente de forma adequada o comportamento da série temporal original. Para esse trabalho, o número de estados iniciais de cada série foi escolhido com base eu uma análise visual da série. Outra escolha importante é a função de similaridade utilizada na descoberta de estados pela ferramenta de *clustering*. A ferramenta CLUTO oferece quatro possíveis funções

de similaridade: cosseno, inverso da distância Euclidiana, coeficiente de correlação entre os objetos e uma extensão do coeficiente de Jaccard. Dessas, apenas as funções cosseno e inversa da distância Euclideana foram testadas. O valor do parâmetro $\boldsymbol{b}$, janela de estabilização do estado, deve ser escolhido com cuidado pois na ocorrência de clusters (estados) pequenos, esses podem ser removidos, na escolha de valores altos. A memória, utilizada na verificação de transições não se mostrou um parâmetro de difícil atribuição, o valor 3 foi eficiente para todas os experimentos realizados. Por fim, o parâmetro Limite de Estagnação no Estado também não se mostrou um parâmetro complicado de se atribuir um valor, o valor 1.5 foi suficiente para todos os experimentos. A seção 4.2 explica o papel de cada parâmetro utilizado nesta seção de forma mais detalhada.

Duas séries representando eletrocardiogramas (obtidas em [Keo05]), além da série Space Shuttle Marotta Valve Series, já usada no experimento com o método kernel *k*NNDDSRM, foram testadas. A tabela 5.16 mostra os parâmetros utilizados em cada experimento.

**Tabela 5.16** Parâmetros utilizados nos experimentos para detecção de novidades baseada na transição de estados.

| Série | $w$ | memória | número de estados (clusters) | função de similaridade | $b$ | limite de estagnação no estado |
|---|---|---|---|---|---|---|
| $ECG_1$ | 50 | 3 | 10 | RBF | 5 | 1.5 |
| $ECG_2$ | 60 | 3 | 10 | Cosseno | 4 | 1.5 |
| Marotta | 60 | 3 | 10 | RBF | 5 | 1.5 |

#### 5.4.3.1 Resultados Obtidos

A figura 5.29 mostra o resultado obtido para a série $ECG_1$. O conjunto de treinamento dessa série contém 1750 amostras codificadas com janela 50, como mostra a tabela 5.16. Para esse experimento, nossa abordagem conseguiu reduzir o conjunto original de treinamento em 91.83%, foram armazenados apenas 8.17% do total de amostras. Na figura 5.29, vê-se que a novidade é visivelmente detectada e o algoritmo detectou de forma correta a região.

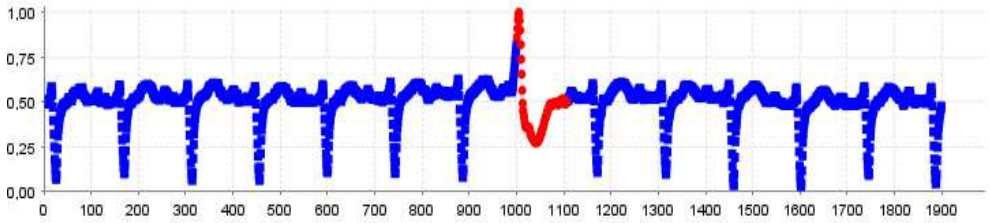


**Figura 5.29** Resultado obtido para a detecção de novidades baseada em estados aplicada ao conjunto de teste da série $ECG_1$

A figura 5.30 mostra o resultado obtido para a série $ECG_2$. Para essa base, a função de similaridade RBF não obteve bons resultados, a função cosseno foi escolhida e obteve um resultado que condiz com a avaliação visual da série. O conjunto de treinamento dessa série contém 2440 amostras codificadas com janela 60. Para esse experimento, nossa abordagem

conseguiu reduzir o conjunto original de treinamento em 86.27%, foram armazenados apenas 13.73% do total de amostras. Na figura 5.30, o segundo trecho detectado como novidade (após o pequeno trecho azul à direita da primeira novidade), foi assim classificado não pelo fato da ocorrência de transições entre estados desconhecidas, mas pela estagnação da série por um tempo maior que 1.5 vezes o maior número de repetições de amostras, no treinamento, naquele mesmo estado.

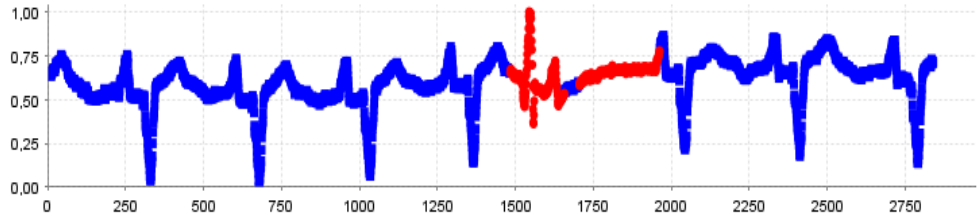


**Figura 5.30** Resultado obtido para a detecção de novidades baseada em estados aplicada ao conjunto de teste da série $ECG_2$

A figura 5.31 mostra o resultado obtido para a série Space Shuttle Marotta Valve. O conjunto de treinamento dessa série contém 2690 amostras codificadas com janela 60. Para esse experimento, nossa abordagem conseguiu reduzir o conjunto original de treinamento em 91.12%, foram armazenados apenas 8.88% do total de amostras. Na figura 5.31, vê-se que, diferentemente dos experimentos anteriores para essa mesma base (ver figura 5.23), a região novidade foi totalmente detectada como tal, porém, uma parte da classe normal também foi detectada como novidade. Essa detecção da classe normal como novidade pode não ser prejudicial ao resultado pois ela acontece de forma seguida a uma correta detecção da classe novidade. Uma detecção incorreta de objetos novidades (alarmes falsos - falsos negativos) em pontos distantes de verdadeiras novidades podem resultar em maiores transtornos.

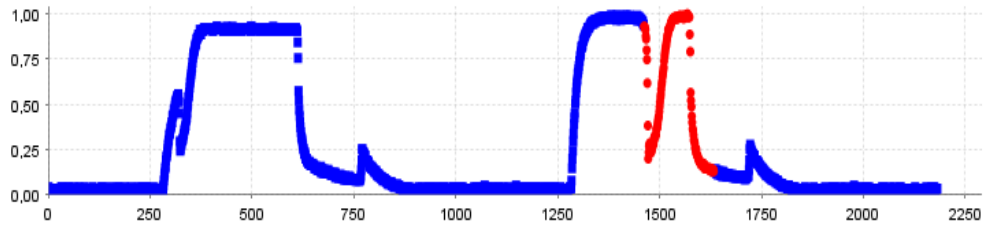


**Figura 5.31** Resultado obtido para a detecção de novidades baseada em estados aplicada ao conjunto de teste da série Space Shuttle Marotta Valve

## 5.5 Considerações Finais

Nesta seção foram reportados experimentos para a detecção de novidades em dois diferentes contextos: bases de dados atemporais e bases temporais (séries temporais). Sete bases de dados atemporais foram utilizadas na validação dos resultados do método kernel *k*NNDDSRM, sendo

que dessas sete, duas são bases com múltiplas classes. Durante os experimentos, analisando os resultados, vemos que o método kernel *k*NNDDSRM consegue resultados similares aos mais conhecidos métodos para classificação com exemplos de uma única classe, obtendo uma maior AUC em 6 dos 10 experimentos realizados quando aplicado o kernel RBF.

Segundo a análise realizada sobre os parâmetros ajustáveis do método kernel *k*NNDDSRM, concluímos que o uso do parâmetro $k \neq 1$ pode ser prejudicial em algumas bases, a medida que seu valor é incrementado, a partir de um certo ponto, o classificador começa a perder em precisão. Vimos também que os parâmetros $\sigma$ e $d$ influenciam de maneira discreta na construção da descrição dos dados. O uso do kernel se mostrou bastante benéfico, porém, das quatro diferentes situações testadas (simples distância Euclideana, kernel RBF, kernel polinomial e Cosseno) o uso de kernel RBF e polinomial se sobressaiu significantemente sobre as outras duas opções. Assim como o $k$, o *fracrej* é, também, um parâmetro com forte dependência em relação à base de dados. Um valor alto para esse parâmetro pode resultar numa grande tendência para a classificação, apenas, de objetos da classe novidade.

Foram realizados 8 experimentos com seis séries temporais diferentes. Para os experimentos com o kernel *k*NNDDSRM aplicado a séries temporais, além da AUC utilizamos, como critério de avaliação, os números de *verdadeiros positivos*, *verdadeiros negativos*, *falsos positivos* e *falsos negativos*. Um bom classificador é um classificador que alcança o maior número possível de *verdadeiros positivos* com o menor número possível de *falsos negativos*, havendo pelo menos uma ocorrência *verdadeira negativa*. Pela combinação desses critérios vimos um melhor desempenho dos métodos kernel *k*NNDDSRM e one-class SVM, ambos utilizando o kernel RBF. Na série seno o kernel *k*NNDDSRM se sobressaiu discretamente sobre o one-class SVM obtendo um menor número de *falsos negativos*, ocorrência mais prejudicial no problema da detecção de novidades em séries temporais.

A subseção 5.4.2 aborda o uso de várias dimensões (janelas de tempo) diferentes simultaneamente durante o treinamento e teste do classificador. Essa abordagem agrega robustez à classificação, porém, requer um maior poder computacional pois resulta em descrições com maior armazenamento de amostras de treinamento.

Como já citado anteriormente, a abordagem utilizada por classificadores para exemplos de uma única classe convencionais carrega um problema que é o fato não intuitivo de que novidades não necessariamente se encontram de maneira dispersa ou fora de uma descrição normal dos dados, no caso a série temporal. Dessa forma, realizamos experimentos com nossa nova abordagem baseada na transição entre estados, introduzida neste trabalho (seção 4.2), em três séries temporais. Os resultados mostram que, com o uso dessa abordagem, o conjunto de treinamento teve reduções na ordem de 90%. Além disso, o método detectou de forma acertada as novidades ocorridas nos três casos.

CAPÍTULO 6

# Conclusão

Essa dissertação introduziu dois novos métodos para detecção de novidades com exemplos de uma única classe (*one-class classification*): kernel *k*NNDDSRM e o método baseado na transição entre estados para detecção de novidades em séries temporais. Aplicações dos métodos no problema da detecção de novidades em dados atemporais e temporais (séries temporais) foram demonstradas. Vários experimentos com técnicas para classificação com exemplos de uma única classe que aplicam, ou não, conceitos de redução de protótipos foram realizados a fim de atestar a eficiência do método kernel *k*NNDDSRM.

Algoritmos para detecção de novidades podem ser aplicados a muitos problemas do mundo real em várias diferentes áreas do conhecimento, como, detecção de falhas em máquinas, diagnósticos médicos e detecção de fraudes em sistemas financeiros. A detecção de novidades encontra grande aplicabilidade tanto em problemas com dados atemporais quanto em problemas com dados temporais (séries temporais). Análise de imagens médicas para diagnóstico patológicos é um exemplo de aplicação com dados atemporais e detecção de mau funcionamento de equipamentos pode ser um exemplo de aplicação para a detecção de novidades em séries temporais.

Métodos baseados na regra do vizinho mais próximo têm como características a necessidade de um grande armazenamento de dados e posterior computação desse grande montante de dados; características que podem ser indesejáveis em aplicações práticas. Nossa suposição é que a redução no número de protótipos para a classificação ajudará na lida com essas características indesejáveis. Existem casos em que essa redução de protótipos, contudo, não é alcançada de forma simples, alguns problemas não apresentam dados com um nível suficiente de correlação. Outro problema na redução de protótipos é a existência de ruídos, o algoritmo pode ser obrigado a armazenar mais protótipos que o necessário. Dentre os métodos testados, analisando em termos de redução de objetos no conjunto de treinamento, o método kmeans_dd (kmeans data description) obteve as maiores taxas de redução de protótipos, poucos centros foram necessários para descrever os problemas tratados. O método kernel *k*NNDDSRM obteve taxas de redução de protótipos comparáveis ao SVDD, método, teoricamente, mais sofisticado atualmente.

Em classificação com exemplos de uma única classe alguns métodos são mais indicados para determinadas situações. Para uma boa amostragem dos dados, assumimos que a distribuição do conjunto de treinamento se assemelha à distribuição real dos dados e nesse caso métodos baseados em estimativa de densidade, como o parzen, teoricamente, têm um melhor desempenho. Para um conjunto de treinamento com poucas amostras, suposições devem ser feitas (pelo método) em relação aos dados para que se alcance uma boa solução, a obtenção de um subconjunto representativo se torna bastante difícil. Nesses casos, métodos baseados em

fronteiras entre os dados têm um melhor desempenho. Métodos como kernel *k*NNDDSRM, SVDD e *k*NNDD, teoricamente são mais aplicáveis. A medida utilizada para a avaliação de desempenho dos métodos foi a AUC (*Area Under Curve*). Nos experimentos realizados o kernel *k*NNDDSRM obteve AUCs comparáveis aos melhores métodos, quando não, as melhores AUCs.

Séries temporais podem se apresentar com características (comportamentos) bastante diversas. Para se aplicar um classificador para exemplos de uma única classe a uma determinada, série deve-se conhecer o comportamento normal da série. Classificadores nem sempre são eficientes na descoberta de novidades. Neste trabalho, além da aplicação de classificadores convencionais, como o kernel *k*NNDDSRM e o One-class SVM, ao problema da detecção de novidades em séries temporais, foi desenvolvida uma nova abordagem baseada na transição entre estados da série temporal. Esse método se mostrou, de acordo com os resultados, bastante eficaz na resolução de problemas onde classificadores convencionais, geralmente, retornam resultados pobres.

## 6.1 Contribuições

As principais contribuições desse trabalho estão listadas abaixo:

1. Criação de um novo método (kernel *k*NNDDSRM) baseado em princípios do conceito de minimização do erro estrutural capaz de obter resultados similares ao SVDD (atualmente, método mais sofisticado para classificação com exemplos de uma única classe) sem a necessidade da resolução da otimização quadrática, utilizada em métodos baseados em máquinas de vetores de suporte. O método pode ser aplicado tanto a detecção de novidades em series temporais quanto a dados atemporais.

2. Criação de um novo método, exclusivamente para detecção de novidades em séries temporais, baseado na análise de transições entre os estados da série, assim como em Salvador e Chan [SC05], porém, fazendo uso do conceito de minimização do risco estrutural para redução do conjunto de treinamento. Experimentos com essa abordagem demonstraram a eficiência do método em problemas onde novidades ocorrem de forma não intuitiva.

Classificação com exemplos de uma única classe ainda é um tema relativamente pouco pesquisado. Na literatura, dos trabalhos sobre classificação com exemplos de uma única classe, poucos explicitamente se intitulam nessa categoria. Em muitos casos a detecção de novidades é confundida com classificação com exemplos de uma única classe. Essa é uma área ainda pouco explorada, onde, muitos dos trabalhos giram em torno de técnicas como SVDD e one-class SVM, portanto, a criação e investigação de novas técnicas é imprescindível ao avanço nas pesquisas nessa área.

Durante esta dissertação publicamos três artigos, sendo que um deles foi publicado em um periódico internacional [COC08] e dois deles foram publicados em conferencias internacionais [COC07] [CO08].

## 6.2 Propostas para Trabalhos Futuros

De acordo com o trabalho apresentado até este momento, alguns trabalhos futuros e melhorias podem ser propostas. Nessa seção nós propomos três trabalhos futuros.

### 6.2.1 Proposta 1

Bases de dados podem se apresentar distribuídas de diversas maneiras. Exemplos são as bases banana e artificial que têm formato de banana e círculo respectivamente. Em caso de classificadores como o kernel *k*NNDDSRM e SVDD, como já discutido nesse documento, descrições fechadas do problema são criadas e objetos que se localizem fora dessas descrições serão rejeitados. Em certos casos a distribuição pode ter um formato fora do comum que pode ser melhor representado por agrupamentos de dados que juntos caracterizem a distribuição normal. Nesses casos, seria desejável um pré-processamento do conjunto de treinamento de forma a identificar os possíveis agrupamentos de dados agindo de forma isolada em cada agrupamento por vez. Caso não exista conhecimento a-priori dos dados, caso mais comum, essa identificação de agrupamentos deve ser feita de forma não supervisionada, pois não se sabe com antecedência o número de agrupamentos existente na distribuição. Existem vários métodos para agrupamentos propostos na literatura, por exemplo o Chameleon [KHN99], que consiste num algoritmo para agrupamento de dados de forma hierárquica.

A figura 6.1 ilustra de forma gráfica o problema descrito anteriormente.

Na figura 6.1, duas representações diferentes do mesmo problema são ilustradas. Nela uma base de dados, visivelmente, representada por três agrupamentos de dados sem intersecção entre eles é simulada. A base consiste da distribuição Gaussiana, utilizada no capítulo de experimentos, mais sua replicação em outros dois pontos distintos do espaço bi-dimensional. Nesse caso, o tratamento da base de dados como uma distribuição uniforme pode levar a descrições imprecisas. Na figura 6.1 (a) é mostrada a descrição gerada pelo método kernel *k*NNDDSRM, como visto anteriormente, o kernel *k*NNDDSRM rejeita as *fracrej* % amostras localizadas em pontos menos densos do conjunto de treinamento. Nesse caso, vê-se que uma descrição pouco condizente com a realidade dos dados é gerada. Em contrapartida, na figura 6.1 (b), um algoritmo de agrupamento de dados foi utilizado e para cada agrupamento encontrado o método kernel *k*NNDDSRM foi aplicado gerando uma descrição mais justa e condizente com a verdadeira distribuição da classe normal.

### 6.2.2 Proposta 2

Tax quando desenvolveu o SVDD [Tax01], o habilitou para a incorporação de novidades quando essas forem disponíveis. O problema da otimização quadrática foi reformulado de maneira a utilizar as novidades conhecidas durante o treinamento na busca pelos vetores de suporte, alcançando assim uma descrição mais precisa da classe normal.

O método proposto nesse trabalho é genuinamente para classificação com exemplos de uma única classe. Dados da classe novidade, mesmo que disponíveis não seriam aproveitados. Na detecção de novidades, como já comentado, novidades nem sempre estão disponíveis. Porém, quando esses dados estiverem disponíveis, a sua incorporação na descrição é de extrema im-

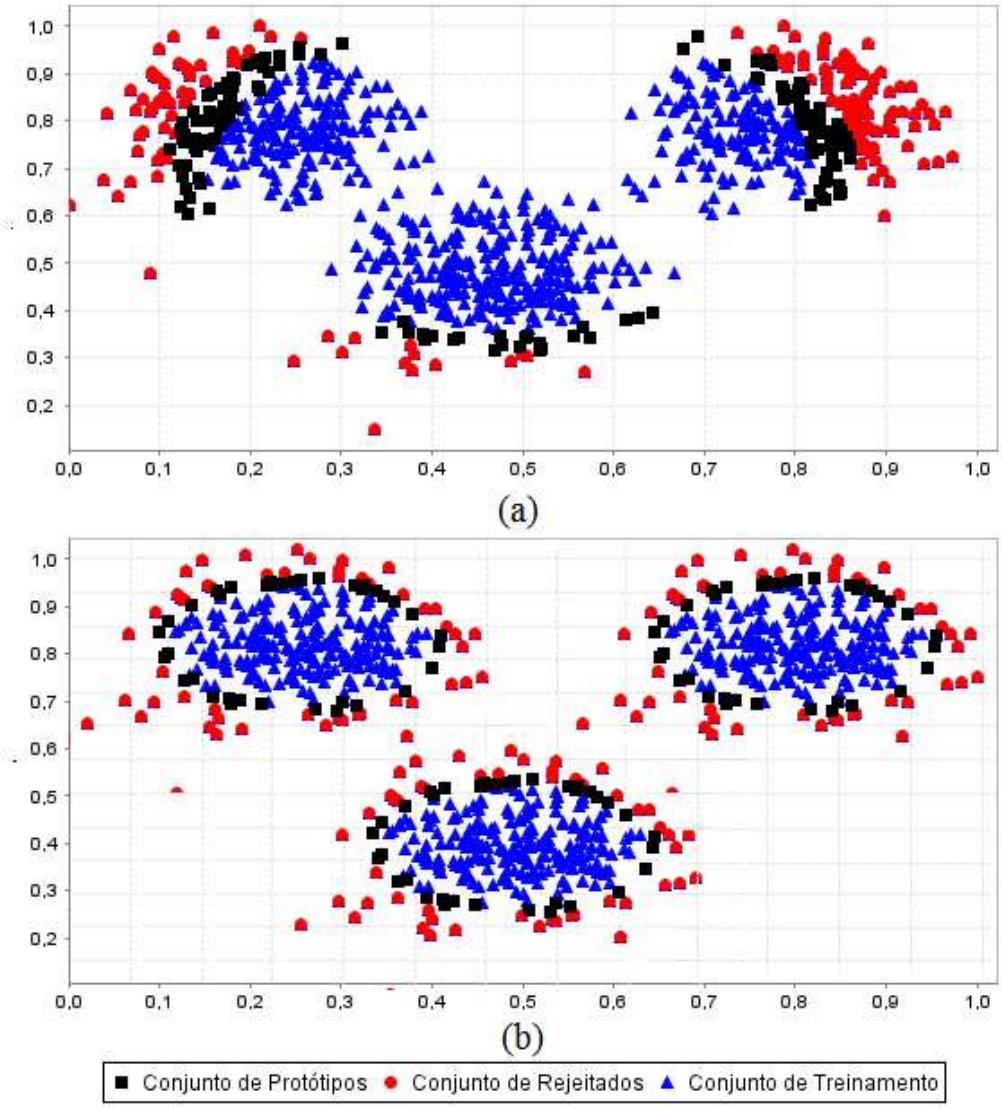


**Figura 6.1** a) Aplicação do kernel *k*NNDDSRM a uma distribuição de dados formada por agrupamentos dispersos entre si; b)Esquema onde um pré-processamento, visando a descoberta de *clusters* de dados, é realizada com a posterior aplicação do kernel *k*NNDDSRM em cada *cluster* encontrado

portância devido à relevância desses dados. O sistema passaria a ter características de um sistema para classificação com duas classes, porém, a descrição fechada da classe normal ainda seria executada. Nesse ponto o sistema se diferencia da classificação com duas classes.

No caso do kernel *k*NNDDSRM, os objetos mais representativos, senão todos (no caso de uma pequena amostragem), da classe novidade, seriam automaticamente adicionados no conjunto de rejeitados e o conjunto de protótipos seria formado por objetos de forma a isolar o restante do conjunto de treinamento dos objetos da classe novidade.

### 6.2.3 Proposta 3

Nesse trabalho nós focamos na busca por um bom classificador para o problema da classificação com exemplos de uma única classe. De acordo com o tipo dos dados (tamanho das amostras, distribuição dos dados e o quão bem a verdadeira distribuição é representada), a descrição dos dados mais adequada deve ser encontrada. Infelizmente classificadores dificilmente se adequam aos dados da melhor forma possível. O uso apenas do melhor classificador, descartando classificadores com piores desempenhos, pode levar à perda de informações preciosas [Wol92]. Para uma melhoria no desempenho, classificadores diferentes (que podem diferir em complexidade ou no algoritmo de treinamento) podem ser combinados. Essa combinação pode não apenas melhorar o desempenho, mas também aumentar a robustez da classificação [SS95]. É possível ainda, nessa combinação, se incluir conhecimento a-priori usando-se regras específicas nessa combinação.

Em trabalhos futuros a combinação de classificadores pode ser uma boa alternativa para problemas cujos métodos utilizados não obtiveram um resultado razoável. O método proposto, kernel *k*NNDDSRM, tem um comportamento bem definido que pode ser estudado para a obtenção da melhor combinação para um determinado problema.

## 6.3 Considerações Finais

O trabalho apresentado nessa dissertação estudou o problema da detecção de novidades, utilizando o conceito de classificação com exemplos de uma única classe, aplicada aos problemas com dados atemporais e a séries temporais. Esperamos que os métodos propostos nesse trabalho sejam de utilidade prática em problemas do mundo real como detecção de fraudes em sistemas de pagamento, detecção de falhas em máquinas e identificação de objetos não pertencentes à cena na análise de imagens.

Vários métodos que implementam a redução de protótipos e a classificação com exemplos de uma única classe, ou ambos, foram propostos e investigados. Características desses métodos foram evidenciadas tornando possível a escolha do método mais indicado para um problema específico. Vários experimentos com dados atemporais foram realizados e foi visto que não houve um método de desempenho superior para todos os casos (em relação às comparações realizadas com o kernel *k*NNDDSRM), porém, houve métodos que obtiveram bons desempenhos , em termos de AUC e taxa de redução de protótipos, regularmente. Dentre esses métodos está o kernel *k*NNDDSRM. Para o problema de séries temporais, foram testados apenas três métodos (kernel *k*NNDDSRM, *one-class* SVM e *k*NNDD). Os métodos kernel *k*NNDDSM e *one-class* SVM obtiveram resultados similares para todos os experimentos realizados. O método *k*NNDD não obteve resultados satisfatórios.

Na detecção de novidades em séries temporais, em casos onde os métodos kernel *k*NNDDSRM, *k*NNDDSRM, *k*NNDD e One-class SVM não foram eficientes, o nosso método baseado na transição de estados da série, com o uso de NNSRM, se mostrou bastante eficiente.

# Referências Bibliográficas


[ABPF07] J. A. S. Almeida, L. M. S. Barbosa, A. A. C. C. Pais, and S. J. Formosinho, *Improving hierarchical cluster analysis: A new method with outlier detection and automatic clustering*, Chemometrics and Intelligent Laboratory Systems **87** (2007), no. 2, 208–217.

[AN07] A. Asuncion and D.J. Newman, *UCI machine learning repository*, 2007, Avaiable at: http://www.ics.uci.edu/∼mlearn/MLRepository.html.

[Ang07] F. Angiulli, *Condensed nearest neighbor data domain description*, IEEE Trans. Pattern Analysis and Machine Intelligence **29** (2007), no. 10, 1746–1758.

[Bat78] B. G. Batchelor, *Pattern recognition: Ideas in practice*, Plenum, 1978.

[BDM02] Kristin P. Bennett, Ayhan Demiriz, and Richard Maclin, *Exploiting unlabeled data in ensemble methods*, Proc. of Int. Conf. on Knowledge Discovery in Data-Mining (KDD), ACM, 2002, pp. 289–296.

[Bel61] R. Bellman, *Adaptive control processes: a guided tour*, Princeton University, Princeton, 1961.

[Bro04] G. Brown, *Diversity in neural network ensembles*, Ph.D. thesis, School of Computer Science, University of Birmingham, 2004.

[Cha74] C. L. Chang, *Finding prototypes for nearest neighbor classifiers*, IEEE Transactions on Computers **23** (1974), 1179–1184.

[CLC03] L. J. Cao, H. P. Lee, and W. K. Chong, *Modified support vector novelty detector using training data with outliers*, Pattern Recognition Letters **24** (2003), no. 14, 2479–2487.

[CO08] George G. Cabral and Adriano L. I. Oliveira, *Novelty detection in time series using a novel method for one-class classification based on the nearest neighbor data description and structural risk minimization*, Proc. of IFIP WCC'2008, World Computer Congress, 2008, (Aceito para Publicação).

[COC07] George G. Cabral, Adriano L. I. Oliveira, and Carlos B. G. Cahú, *A novel method for one-class classification based on the nearest neighbor rule and*

*structural risk minimization*, Proc. IJCNN'2007, International Joint Conference on Neural Networks, 2007, pp. 1976–1981.

[COC08] George G. Cabral, Adriano L. I. Oliveira, and Carlos B. G. Cahú, *Combining nearest neighbor data description and structural risk minimization for one-class classification*, Neural Computing & Applications (2008), Accepted for publication.

[CZH] Y. Chen, X. Zhou, and T. Huang, *One-class svm for learning in image retrieval*.

[DF95] Dipankar Dasgupta and Stephanie Forrest, *Tool breakage detection in milling operations using a negative-selection algorithm.*, Tech. report, 1995.

[DF96] Dipankar Dasgupta and Stephanie Forrest, *Novelty detection in time series data using ideas from immunology*, Web doc, 1996.

[DH73] R. O. Duda and P. E. Hart, *Pattern classification and scene analysis*, Wiley, 1973.

[Dom95] Pedro Domingos, *Rule induction and instance-based learning: A unified approach*, IJCAI, 1995, pp. 1226–1232.

[Dom96] Pedro Domingos, *Unifying instance-based and rule-based induction*, Machine Learning **24** (1996), no. 2, 141–168.

[Ega75] James P. Egan, *Signal detection theory and ROC-analysis*, Academic Press, New York, 1975, ID: 1499787.

[Faw06] T. Fawcett, *An introduction to ROC analysis*, Pattern Recognition Letters **27** (2006), no. 8, 861–874.

[Fis36] R. A. Fisher, *The use of multiple measurements in taxonomic problems*, Annals of Eugenics **7** (1936), no. 2, 179–188.

[FMW08] Peter Filzmoser, Ricardo Maronna, and Mark Werner, *Outlier identification in high dimensions*, Computational Statistics & Data Analysis **52** (2008), no. 3, 1694–1711.

[Gat72] G. W. Gates, *The reduced nearest neighbor rule*, IEEE Trans. Information Theory **18** (1972), no. 5, 431–433.

[Har68] P. E. Hart, *The condensed nearest neighbor rule*, IEEE Transactions on Information Theory **114** (1968), 515–516.

[Hof07] H. Hoffmann, *Kernel PCA for novelty detection*, Pattern Recognition **40** (2007), no. 3, 863–874.

[Kar03] George Karypis, *Cluto - a clustering toolkit*, Tech. Report 02-017, University of Minnesota, Department of Computer Science, nov 2003.

[Keo05] E. Keogh, 2005, http://www.cs.ucr.edu/ eamonn/discords.

[KHN99] George Karypis, Eui-Hong (Sam) Han, and Vipin Kumar NEWS, *Chameleon: Hierarchical clustering using dynamic modeling*, Computer **32** (1999), no. 8, 68–75.

[KK03] B. Karacali and H. Krim, *Fast minimization of structural risk by nearest neighbor rule*, IEEE Trans. on Neural Networks **14** (2003), 127–137.

[KLF05] Eamonn J. Keogh, Jessica Lin, and Ada Wai-Chee Fu, *Hot sax: Efficiently finding the most unusual time series subsequence*, Proc. of Int. Conf. on Data Mining, 2005, pp. 226–233.

[KLL01] Kegl, Linder, and Lugosi, *Data-dependent margin-based generalization bounds for classification*, COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers, 2001, pp. 73–98.

[KO04] Sang-Woon Kim and B. John Oommen, *Enhancing prototype reduction schemes with recursion: a method applicable for large data sets*, IEEE Transactions On Systems, Man, and Cybernetics, Part B **34** (2004), no. 3, 1384–1397.

[KRS04] B. Karacali, R. Ramanath, and W. E. Snyder, *A comparative analysis of structural risk minimization by support vector machines and nearest neighbor rule*, Pattern Recognition Letters **25** (2004), no. 1, 63–71.

[LKWL07] Jessica Lin, Eamonn J. Keogh, Li Wei, and Stefano Lonardi, *Experiencing SAX: a novel symbolic representation of time series*, Data Min. Knowl. Discov **15** (2007), no. 2, 107–144.

[Mac67] J. MacQueen, *Some methods for classification and analysis of multivariate observations*, Proc. of the 5th Berkeley Symp. on Mathematics Statistics and Probability (L. M. LeCam and J. Neyman, eds.), vol. 1, 1967, pp. 281–298.

[Mer09] John Mercer, *Functions of positive and negative type and their connection with the theory of integral equations*, Philos. Trans. Roy. Soc. London **209** (1909), 415–446.

[MKH93] Mary M. Moya, Mark W. Koch, and L. D. Hostetler, *One-class classifier networks for target recognition applications*, Proc. of WCNN'93, World Congress on Neural Networks (Hillsdale, NJ), vol. 3, INNS, Lawrence Erlbaum, 1993, pp. 797–801.

[MP66] Marvin Minsky and Seymour Papert, *Unrecognizable sets of numbers*, Journal of the ACM **13** (1966), no. 2, 281–286.

[MP03] J. Ma and S. Perkins, *Time-series novelty detection using one-class support vector machines*, IJCNN'2003, Proc. of International Joint Conference on Neural Networks, 2003, pp. Vol. 3, Pages 1741 – 1745.

[MS03a] M. Markou and S. Singh, *Novelty detection: a review-part 2: neural network based approaches*, Signal Processing **83** (2003), 2499–521.

[MS03b] Markos Markou and Sameer Singh, *Novelty detection: a review - part 1: statistical approaches*, Signal Processing **83** (2003), no. 12, 2481–2497.

[MY01] Larry M. Manevitz and Malik Yousef, *One-class SVMs for document classification*, Journal of Machine Learning Research **2** (2001), 139–154.

[NS93] Morton Nadler and Eric P. Smith, *Pattern recognition engineering*, Wiley-Interscience, May 1993.

[OdLM06] Adriano L. I. Oliveira and Silvio Romero de Lemos Meira, *Detecting novelties in time series through neural networks forecasting with robust confidence intervals*, Neurocomputing **70** (2006), no. 1-3, 79–92.

[OeABeALMS03] Adriano L. I. Oliveira and Gabriel Azevedo e Adelia Barros e Andre L. M. Santos, *Sistema de suporte a auditoria de folhas de pagamento baseado em redes neurais* (por; eng).

[Oli04] A. L. I. Oliveira, *Neural networks forecasting and classification-based techniques for neural networks forecasting and classification-based techniques for novelty detection in time series*, Ph.D. thesis, Federal University of Pernambuco, 2004.

[ONM03] Adriano L. I. Oliveira, Fernando B. L. N., and Silvio R. L. Meira, *Novelty detection for short time series with neural networks*, Proc. of Int. Hybrid Int. Systems, 2003, pp. 66–76.

[Rät98] G. Rätsch, *Ensemble learning methods for classification*, Diploma thesis (in german).

[ROM98] G. Rätsch, T. Onoda, and K.-R. Müller, *Soft margins for AdaBoost*, Tech. Report NC-TR-1998-021, Department of Computer Science, Royal Holloway, University of London, Egham, UK, August 1998, Submitted to Machine Learning.

[RWLI75] G. L. Ritter, H. B. Woodruff, S. R. Lowry, and T. L. Isenhour, *An algorithm for a selective nearest neighbor decision rule*, IEEE Transactions on Information Theory **21** (1975), 665–669.

[Sal91] Salzberg, *A nearest hyperrectangle learning method*, Machine Learning **6** (1991), 251–276.

[SC05] Stan Salvador and Philip Chan, *Learning states and rules for detecting anomalies in time series*, Appl. Intell **23** (2005), no. 3, 241–255.

[Sec00] Network Security, *43% of credit card fraud not reported*, Network Security (2000), no. 10, 4.

[SEK05] Hyun Joon Shin, Dong-Hwan Eom, and Sung-Shick Kim, *One-class support vector machines: an application in machine fault detection and classification*, Comput. Ind. Eng. **48** (2005), no. 2, 395–408.

[Seo07] Kwang-Kyu Seo, *An application of one-class support vector machines in content-based image retrieval*, Expert Syst. Appl **33** (2007), no. 2, 491–498.

[SG01] Rob Saunders and S. Gero, *The importance of being emergent*, Proc. of Artificial Intelligence in Design (2001).

[SPST+01] Bernhard Schölkopf, John C. Platt, John Shawe-Taylor, Alex J. Smola, and Robert C. Williamson, *Estimating the support of a high-dimensional distribution*, Neural Computation **13** (2001), no. 7, 1443–1471.

[SS95] A J. C. Sharkey and Noel E. Sharkey, *How to improve the reliability of artificial neural networks*, October 16 1995, Research Report CS-95-11, Department of Computer Science, University of Shefield.

[SS00] Zbigniew R. Struzik and Arno P. J. M. Siebes, *Outlier detection and localisation with wavelet based multifractal formalism*, 1161, Centrum voor Wiskunde en Informatica (CWI), ISSN 1386-3681, Febuary 29 2000, INS (Information Systems (INS)), p. 18.

[Swe88] J. A. Swets, *Measuring the Accuracy of Diagnostic Systems*, Science **240** (1988), 1285–1293.

[SWS+99] Bernhard Schölkopf, Robert C. Williamson, Alex J. Smola, John Shawe-Taylor, and John C. Platt, *Support vector method for novelty detection*, NIPS (Sara A. Solla, Todd K. Leen, and Klaus-Robert Müller, eds.), The MIT Press, 1999, pp. 582–588.

[Tax01] D. M. J. Tax, *One-class classification concept-learning in the absence of counter-examples*, Ph.D. thesis, Technische Universiteit Delft, 2001.

[Tax07] D.M.J. Tax, *Ddtools, the data description toolbox for matlab*, July 2007, version 1.6.1.

[TD98] D. M. J. Tax and R. P. W. Duin, *Outlier detection using classifier instability*, Lecture Notes in Computer Science **1451** (1998), 593–601.

[TD99] David M. J. Tax and Robert P. W. Duin, *Data domain description using support vectors*, Proc. of European Symposium on Artificial Neural Networks, 1999, pp. 251–256.

[Tom76] I. Tomek, *An experiment with the edited nearest-neighbor rule*, IEEE Transactions on Systems, Man and Cybernetics **6** (1976), 448–452.

[Vap95] V. N. Vapnik, *The nature of statistical learning theory*, Springer, 1995.

[Vap98] V. Vapnik, *Statistical learning theory*, Wiley, 1998.

[VC74] Vladimir N. Vapnik and Aleksei Ja. Chervonenkis, *Ordered risk minimization (i and ii)*, Automation and Remote Control **34** (1974), 1226–1235 and 1403–1412.

[WD95] Wettchereck and Dietterich, *An experimental comparison of the nearest-neighbor and nearest-hyperrectangle algorithms*, Machine Learning **19** (1995), 5–27.

[WG94] Andreas S. Weigend and Neil A. Gerschenfeld, *Time-series prediction: Forecastind the future and understanding the past.*, Addison-Wesley, 1994.

[Wil72] D. L. Wilson, *Asymptotic properties of nearest neighbor rules using edited data*, IEEE Trans. Systems, Man and Cybernetics **2** (1972), 408–421.

[WM97] D. Randall Wilson and Tony R. Martinez, *Improved heterogeneous distance functions*, J. Artif. Intell. Res. (JAIR) **6** (1997), 1–34.

[WM00] D. Randall Wilson and Tony R. Martinez, *Reduction techniques for instance-based learning algorithms*, Machine Learning **38** (2000), no. 3, 257.

[WM08] Qiang Wang and Vasileios Megalooikonomou, *A dimensionality reduction technique for efficient time series similarity analysis*, Inf. Syst **33** (2008), no. 1, 115–132.

[Wol92] D. Wolpert, *Stacked generalization*, Neural Networks **5** (1992), no. 2, 241–260.

[YDJ06] Zhaoyan Sun Yonggui Dong and Huibo Jia, *A cosine similarity-based negative selection algorithm for time series novelty detection*, Mechanical Systems and Signal Processing **20** (2006), 1461–1472.

[Zhi07] S. I. Zhilin, *Simple method for outlier detection in fitting experimental data under interval error*, Chemometrics and Intelligent Laboratory Systems **88** (2007), no. 1, 60–68.

# Livros Grátis

( http://www.livrosgratis.com.br )

Milhares de Livros para Download:


Baixar livros de Administração
Baixar livros de Agronomia
Baixar livros de Arquitetura
Baixar livros de Artes
Baixar livros de Astronomia
Baixar livros de Biologia Geral
Baixar livros de Ciência da Computação
Baixar livros de Ciência da Informação
Baixar livros de Ciência Política
Baixar livros de Ciências da Saúde
Baixar livros de Comunicação
Baixar livros do Conselho Nacional de Educação - CNE
Baixar livros de Defesa civil
Baixar livros de Direito
Baixar livros de Direitos humanos
Baixar livros de Economia
Baixar livros de Economia Doméstica
Baixar livros de Educação
Baixar livros de Educação - Trânsito
Baixar livros de Educação Física
Baixar livros de Engenharia Aeroespacial
Baixar livros de Farmácia
Baixar livros de Filosofia
Baixar livros de Física
Baixar livros de Geociências
Baixar livros de Geografia
Baixar livros de História
Baixar livros de Línguas

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo