ANÁLISE DISCRIMINANTE CLÁSSICA E DE NÚCLEO: AVALIAÇÕES E ALGUMAS CONTRIBUIÇÕES RELATIVAS AOS MÉTODOS BOOSTING E BOOTSTRAP

MARCELO RODRIGO PORTELA FERREIRA

Orientador: Prof. Dr. Getúlio José Amorim do Amaral

Co-orientador: Prof. Dr. Francisco Louzada–Neto

Área de Concentração: Estatística Aplicada

Dissertação submetida como requerimento parcial para obtenção do grau de Mestre em Estatística pela Universidade Federal de Pernambuco

Recife, fevereiro de 2007

Ferreira, Marcelo Rodrigo Portela
Análise discriminante clássica e de núcleo : avaliações e algumas contribuições relativas aos métodos Boosting e Bootstrap / Marcelo Rodrigo Portela Ferreira – Recife : O autor, 2007.
xv, 94 folhas: il., fig., tab.

Dissertação (mestrado) – Universidade Federal de Pernambuco. CCEN. Estatística, 2007.

Inclui bibliografia e apêndices.

1. Reconhecimento de padrão. Título.

006.4 CDD (22.ed.) MEI2007-007

Universidade Federal de Pernambuco
Pós-Graduação em Estatística

08 de fevereiro de 2007
(data)

Nós recomendamos que a dissertação de mestrado de autoria de

Marcelo Rodrigo Portela Ferreira

intitulada

"Análise discriminante clássica e de núcleo: avaliações e algumas contribuições relativas aos métodos Boosting e Bootstrap"

seja aceita como cumprimento parcial dos requerimentos para o grau de Mestre em Estatística.

Coordenador da Pós-Graduação em Estatística

Banca Examinadora:

Getúlio José Amorim do Amaral orientador

Lúcia Pereira Barroso (USP)

Borko Darko Stosic (UFRPE)

Este documento será anexado à versão final da dissertação.

Este trabalho é carinhosamente dedicado

aos meus pais, Arlindo e Edileuza.

Agradecimentos

À Natureza.

Aos meus pais, Arlindo e Edileuza, pelo amor incondicional, carinho, dedicação, confiança, paciência, amizade e por terem me ensinado os valores e princípios que até hoje norteiam a formação do meu caráter.

Às minhas irmãs, Maristela e Mariana, pelo amor incondicional, carinho, confiança e amizade.

À minha avó, Helena, pelo amor, carinho, confiança e apoio que sempre meu deu.

Às minhas tias, Iracema, Iracy, Maria José (Mazé), Edlenúzia e Edilúzia pela confiança, carinho e apoio durante todo esse tempo.

Ao professor Getúlio José Amorim do Amaral, pela orientação.

Ao professor Francisco Louzada–Neto, pela co-orientação.

Aos meus primos, Tiago, Zé Carlos e André, pela convivência, amizade e confiança.

Aos meus amigos de Palmares, Fernando (Cocada), Rodrigo (Negão), André (Dé) e Rallison, pela convivência, amizade e confiança.

Aos meus amigos do mestrado, Hemílio, Leonardo, Geraldo, Ênio, Larissa, Rejane, Katya, Juliana, Lídia, Lilian, Abraão, Valmir, Fábio Veríssimo, Raphael e Cecílio, pela amizade, carinho e pelos momentos de alegria compartilhados.

Aos meus amigos do "ano da tese", Carlos Gadelha, Tiago Sales, Renata Nunes, Themis, Artur e Luz Milena.

Aos meus amigos de turma que não seguiram comigo no mestrado, Luis Medeiros, Carlos Francisco, Cristiano, Oscar e Lucas.

Aos meus amigos da graduação, Romero, Demetrius, Pedro, Renato, Tiago "o Chico", Robinho, Manu, Claudyvan, Vanessa, Juliana, Tadeu, Maíra, Leila e Luciano.

À Valéria Bittencourt, pelo enorme carinho, paciência e amizade com que sempre tratou a mim e aos demais alunos do mestrado.

Aos professores do Departamento de Estatística, em especial aos professores Maria Cristina Falcão Raposo, Klaus Leite Pinto Vasconcellos e Francisco Cribari–Neto.

À banca examinadora, pelas valiosas críticas e sugestões.

À CAPES, pelo apoio financeiro.

Se cada dia cai

Se cada dia cai, dentro de cada noite,
há um poço
onde a claridade está presa.

Há que sentar-se na beira
do poço da sombra
e pescar luz caída
com paciência.

Pablo Neruda.

Resumo

Desde que tecnologia da informação tornou-se essencial para muitas atividades da vida moderna e grandes conjuntos de dados surgiram junto com ela, mineração de dados tornou-se uma das mais importantes áreas de pesquisa na ciência estatística. Apesar de existirem muitos campos relacionados a mineração de dados, a tarefa de classificação ainda figura como uma das mais comuns na literatura estatística.

Esta dissertação faz uma revisão de dois métodos clássicos de classificação, análise discriminante linear e quadrática, e um método não-paramétrico, a análise discriminante de núcleo. Experimentos de simulação e conjuntos de dados reais são utilizados para avaliar e comparar os três métodos de classificação.

Também apresenta algumas contribuições relacionadas aos métodos boosting e bootstrap no contexto de classificação. A primeira contribuição trata-se de uma nova formulação para o método boosting em análise discriminante linear. Os resultados numéricos mostram que esta nova formulação tem desempenho similar à formulação usual. Entretanto, a nova formulação do método boosting é conceitualmente mais adequada.

Dois métodos bootstrap para problemas de classificação são introduzidos e avaliados. O primeiro método bootstrap é utilizado para obter uma fronteira de classificação. O conceito de fronteira de classificação pode ser entendido como uma região onde é difícil alocar uma observação para uma das populações existentes.

O segundo método bootstrap é um intervalo de confiança para a taxa de erro de classificação. Intervalos de confiança podem ser utilizados para comparar dois ou mais métodos de classificação na estrutura de inferência.

Palavras-chave: Classificação; Análise Discriminante Linear; Análise Discriminante Quadrática; Análise Discriminante de Núcleo; Boosting; Bootstrap; Fronteira de Classificação; Intervalos de Confiança Bootstrap.

Abstract

Since information technology is essential for many activities of modern life and massive data sets come with it, data mining has become one of the most important research topics in statistical science. Even thought there are many fields related to data mining, the task of classification still remain as one of the most common in statistical literature.

This dissertation reviews two classical classifications methods, linear and quadratic discriminant analysis, and a nonparametric method, the kernel discriminant analysis. Simulations experiments and real data sets are used to compare the three classification methods.

It also presents some contributions which are related to boosting and bootstrap methods in the classification context. The first contribution is a new formulation to the boosting method in linear discriminant analysis. The numerical results show that this new formulation has similar performance to the previous one. However, the new boosting formulation is more appropriate from the conceptual point of view.

Two bootstrap methods for classification problems are introduced and evaluated. The first bootstrap method is used to obtain a classification frontier. The concept of classification frontier can be understood as a region where it is difficult to assign one observation to one of some relevant populations.

The second bootstrap method is a confidence interval for the classification error rate. Confidence intervals can be used to compare two or more classification methods in the inference framework.

Keywords: Classification; Linear Discriminant Analysis; Quadratic Discriminant Analysis; Kernel Discriminant Analysis; Boosting; Bootstrap; Classification Frontier; Bootstrap Confidence Intervals.

Índice

Lista de Figuras

Lista de Tabelas

CAPÍTULO 1

Introdução

1.1 Introdução

Desde a infância somos capazes de reconhecer dígitos e letras. Caracteres pequenos ou grandes, manuscritos ou impressos, escritos de maneira desordenada ou até mesmo parcialmente ocultos — todos são facilmente reconhecidos. Esta habilidade de reconhecer ou classificar padrões faz com que nós humanos sejamos os melhores classificadores existentes, ainda que não conheçamos totalmente o mecanismo através do qual reconhecemos padrões.

O largo espectro de aplicações e o constante aumento da capacidade de processamento dos computadores digitais têm motivado pesquisadores a tentar reproduzir tal habilidade em máquinas e programá-las para executar tarefas cada vez mais complexas. De fato, procedimentos de classificação têm ajudado bancos em aplicações de *credit scoring*, médicos em diagnósticos preliminares de doenças com o propósito de selecionar tratamentos imediatos enquanto se aguarda resultados definitivos de exames, pilotos em manobras de aterrisagem, etc.

Um procedimento de classificação ou simplesmente classificador é algum método formal capaz de decidir, com base em informações fornecidas, a que grupo ou população um determinado objeto pertence. De uma maneira geral, estamos interessados em construir, com base em uma amostra de dados, uma regra de classificação e utilizá-la para classificar novos objetos.

Em estatística, os procedimentos que estão entre os mais utilizados e difundidos na literatura

de classificação de padrões são a análise discriminante linear, a análise discriminante quadrática e a regressão logística, sendo a análise discriminante linear um dos métodos mais antigos de discriminação [Lachenbruch (1975); Mardia et al. (1979); Johnson & Wichern (1998); Anderson (2003)]. Estes métodos, cujos fundamentos estão estabelecidos na literatura há bastante tempo, têm baixo custo computacional e, em geral, produzem resultados satisfatórios. Contudo, outros métodos como redes neurais artificiais, árvores de classificação e regressão e análise discriminante de núcleo têm recebido bastante atenção dos pesquisadores nos últimos anos; ver, por exemplo, Friedman et al. (2000*b*).

Recentemente, em uma área de pesquisa formada por pesquisadores da área de computação e denominada *aprendizado automático* (machine learning), muitos pesquisadores têm se esforçado na busca por métodos de classificação automáticos, guiados pelos dados, e métodos que combinem as potencialidades de vários classificadores, tais como o método *boosting* [Schapire (1990)] e o método *bagging* [Breiman (1996)].

Nosso interesse neste trabalho, é estudar um método não paramétrico de classificação, denominado análise discriminante de núcleo, que é baseado na estimação não-paramétrica das funções de densidade de probabilidade das populações através de funções núcleo, e comparar este método com dois métodos clássicos de discriminação, bastante utilizados em estatítica: análise discriminante linear e análise discriminante quadrática. Dessa forma, esperamos contribuir na divulgação e no desenvolvimento deste método de classificação. Fix & Hodges (1951) apresentaram este método não paramétrico de discriminação pela primeira vez e avaliaram as propriedades assintóticas dos estimadores de densidades, no contexto de discriminação entre duas populações.

Adicionalmente, estamos interessados em explorar o método boosting, aplicado a análise discriminante linear, com o objetivo de propor um método que seja conceitualmente mais apropriado. Assim, uma forma alternativa de atribuição de pesos às observações no método boosting é proposta. Além disso, exploramos o método bootstrap como forma de fornecer melhores estimativas para as taxas de erro dos classificadores através de uma versão corrigida do estimador bootstrap para a taxa de erro, o estimador bootstrap .632 [Efron (1983)] e construção de intervalos de confiança bootstrap para a taxa de erro de um método de classificação qualquer, além

de utilizar o método bootstrap na construção de um mecanismo de auxílio à tomada de decisões pelo usuário final de um método de classificação.

Na próxima seção, temos o objetivo de apresentar o sistema de classificação básico e introduzir a notação necessária para definir um procedimento geral de classificação.

1.2 O Sistema de Classificação Básico

Um sistema de classificação de padrões pode ser definido em várias etapas. A figura 1.1 ilustra estas etapas de maneira simplificada. O sistema se inicia com a coleta dos dados através de algum tipo de sensor capaz de fornecer uma representação o padrão por meio de medidas. Por exemplo, em reconhecimento de assinaturas, esse sensor pode ser uma caneta especial capaz de fornecer informações tais como pressão da escrita, tempo de escrita, dentre outras. Em diagnóstico médico preliminar a aquisição dos dados pode ser feita através da descrição dos sintomas dos pacientes e através de medidas tais como pressão arterial, peso e níveis de determinadas substâncias na corrente sangüínea (por exemplo, nível de glicose).

Em uma etapa intermediária as características mais relevantes dos padrões são selecionadas através de algum método, possivelmente automático. Estas serão então utilizadas na definição da regra de classificação, que nos permitirá tomar decisões sobre os grupos aos quais os padrões pertencem.

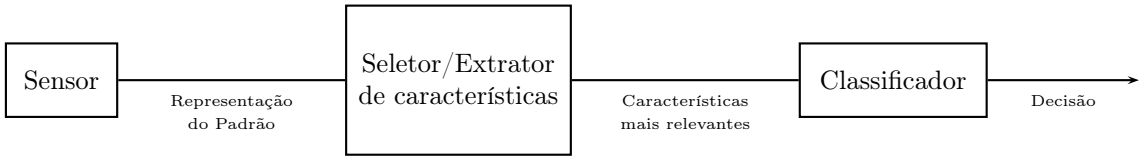

Figura 1.1: Sistema Básico de Classificação de Padrões.

Nesta dissertação estaremos preocupados apenas com a última etapa desse sistema, a construção do classificador, além da avaliação do desempenho deste.

Usaremos o termo *padrão* para denotar o vetor de dados p-dimensional $\boldsymbol{x} = (x_1, \ldots, x_p)'$

de medidas, cujos componentes x_i são medidas de características de um objeto. Desse modo, as características são as variáveis especificadas pelo investigador e julgadas importantes para a classificação. Considere que existem g populações ou grupos, denotados por $\Pi_1, \ldots, \Pi_g$, $g \geq 2$, e associada com cada padrão $\boldsymbol{x}$ existe uma variável categorizada y que denota a população ou grupo ao qual $\boldsymbol{x}$ pertence; isto é, se $y = i$, então o padrão pertence a Π_i, $i \in \mathcal{C} = \{1, \ldots, g\}$. Além disso, o espaço de todos os valores possíveis de $\boldsymbol{x}$ é designado por $\mathcal{X}$.

Para definir um procedimento de classificação, considere que dispomos de uma amostra de treinamento $T = \{\boldsymbol{x}_j, y_j\}_{j=1}^{n}$, onde n denota o tamanho da amostra. Um procedimento de classificação consiste então em obter uma regra $F : \mathcal{X} \to \mathcal{C}$ a partir da amostra T, tal que para cada $\boldsymbol{x} \in \mathcal{X}$, $F(\boldsymbol{x})$ designa uma classe em $\mathcal{C}$. É desejável que o classificador F seja capaz de tomar sólidas e razoáveis decisões a respeito das populações dos padrões.

1.3 Organização da Dissertação

Além deste capítulo introdutório, esta dissertação é composta por mais seis capítulos. No capítulo 2 nos concentramos nos métodos clássicos de discriminação entre duas ou mais populações. Apresentamos uma revisão geral sobre os métodos da análise discriminante linear e análise discriminante quadrática e discutimos suas principais características. No capítulo 3 abordamos os aspectos gerais do método não-paramétrico de estimação de funções de densidade de probabilidade através do método de núcleo e o método de discriminação baseado na estimação não-paramétrica das funções de densidade de probabilidade das populações, proposto por Fix & Hodges (1951). No capítulo 4 apresentamos as formas mais comumente utilizadas para avaliar o desempenho de um método de classificação: Treinamento-e-Teste, Cross-validation e Bootstrap. No capítulo 5 apresentamos um estudo comparativo entre os métodos de classificação clássicos e o método discriminante de núcleo. No capítulo 6 apresentamos algumas contribuições desenvolvidas ao longo desta dissertação. Nesse capítulo apresentamos uma forma alternativa de atribuição de pesos às observações no método boosting tendo como classificador base a função discriminante linear de Fisher. Também apresentamos um algoritmo, baseado no método bootstrap, para construção de uma fronteira de classificação entre duas populações, em análise discriminante linear.

Além disso, propomos uma metodolodia para construção de intervalos de confiança para a taxa de erro de um método de classificação qualquer, também baseada na técnica de bootstrap. Por fim, no capítulo 7, apresentamos as conclusões extraídas do presente trabalho e apontamos direções para trabalhos futuros.

1.4 Plataforma Computacional

O ambiente de programação, análise de dados e gráficos R [R Development Core Team (2006)] e o sistema tipográfico LaTeX [Lamport (1994)] constituem a plataforma computacional utilizada no desenvolvimento desta dissertação.

R

Todos os resultados numéricos e todos gráficos apresentados nesta dissertação de mestrado foram obtidos utilizando o ambiente de programação, análise de dados e gráficos R em sua versão 2.3.1 para sistema operacional Microsoft Windows, que se encontra disponível gratuitamente através do site `http://www.R-project.org`.

Trata-se de uma linguagem caracterizada pelo compromisso entre a flexibilidade oferecida por algumas linguagens compiladas (como `C`, `C++` e `FORTRAN`) e a conveniência dos tradicionais pacotes estatísticos. Foi criada por Ross Ihaka e Robert Gentleman na Universidade de Auckland com o objetivo de produzir um ambiente de programação parecido com S, uma linguagem desenvolvida no AT&T Bell Laboratories, mas sem sofrer dos mesmos problemas de demanda de memória e desempenho. Maiores detalhes sobre esta linguagem de programação podem ser encontrados em Cribari Neto & Zarkos (1999).

LaTeX

A presente dissertação de mestrado foi digitada utilizando o sistema de tipografia LaTeX desenvolvido por Leslie Lamport em 1985, que consiste em uma série de macros ou rotinas do sistema TeX (criado por Donald Knuth na Universidade de Stanford) que facilitam o desenvolvimento da edição do texto. LaTeX possui comandos muito cômodos e elegantes para a criação de tabelas, índices, bibliografia, referências cruzadas, etc., permitindo ao usuário

concentrar-se na estrutura do documento em vez de detalhes puramente TEX-nicos. Detalhes sobre o sistema de tipografia LATEX podem ser encontrados em Lamport (1994) ou através do site `http://www.tex.ac.uk/CTAN/latex`.

CAPÍTULO 2

Análise Discriminante Clássica

2.1 Introdução

Análise discriminante é, talvez, a técnica estatística de classificação mais antiga e mais difundida na literatura [Lachenbruch (1975); Mardia et al. (1979); Johnson & Wichern (1998); Anderson (2003)].

Considere g populações ou grupos $\Pi_1, \ldots, \Pi_g$, $g \geq 2$. Suponha que associada com cada população Π_j há uma função densidade de probabilidade (fdp) $f_j(\boldsymbol{x})$ em $\mathbb{R}^p$, tal que se um indivíduo pertence à população Π_j, então ele possui fdp $f_j(\boldsymbol{x})$. O objetivo da análise discriminante é alocar um indivíduo para uma dessas g populações com base em suas medidas $\boldsymbol{x}$. Desejamos definir uma regra que nos permita dizer a que população Π_j, $j = 1, \ldots, g$, um determinado indivíduo é mais provável pertencer.

Uma regra discriminante d corresponde a uma divisão do $\mathbb{R}^p$ em regiões disjuntas $R_1, \ldots, R_g$ ($\bigcup R_j = \mathbb{R}^p$). A regra d é definida por

$$\text{alocar } \boldsymbol{x} \text{ para } \Pi_j \text{ se } \boldsymbol{x} \in R_j, \tag{2.1}$$

para $j = 1, \ldots, g$. A discriminação será mais precisa se Π_j tem sua probabilidade concentrada em R_j, para cada j.

Normalmente não temos informação a priori a respeito de qual população um indivíduo é

mais provável pertencer. Entretanto, se tal informação estiver disponível, podemos incorporá-la em um enfoque bayesiano.

A situação onde as funções densidade de probabilidade (fdp's) são completamente conhecidas é a mais simples de analisar teoricamente, embora seja a menos realista na prática. Examinaremos esta situação na seção 2.2.

Em muitos casos conhecemos a forma da fdp para cada população, mas há parâmetros que precisam ser estimados. A estimação é baseada em uma matriz de dados amostrais $X(n \times p)$ cujas linhas (observações) são particionadas em g populações,

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_1 \\ \mathbf{X}_2 \\ \vdots \\ \mathbf{X}_g \end{bmatrix}.$$

A matrix $\mathbf{X}_j$ $(n_j \times p)$ representa a amostra dos n_j indivíduos da população Π_j. Esta situação é uma variação da primeira e será examinada na seção 2.3.

Finalmente, há um enfoque empírico, dado por Fisher (1936), em que não assumimos nenhuma forma particular para as populações, mas simplesmente procuramos por uma regra "sensível" que nos permita discriminar entre elas. Esse caso será abordado na seção 2.4.

2.2 Discriminação quando as distribuições são completamente conhecidas

O caso em que as funções de densidade de probabilidade dos grupos são completamente conhecidas é o mais fácil de análisar, contudo, essa situação quase nunca acontece na prática. Nesta seção faremos uma revisão sobre os métodos de discriminação quando conhecemos as formas das distribuições associadas com cada população.

2.2.1 Discriminação por Máxima Verossimilhança

Considere a situação onde as distribuições exatas das populações são conhecidas. É claro que isto é extremamente raro, embora seja possível estimar as distribuições com bastante precisão, dispondo de amostras grandes o suficiente.

Nosso ponto de partida é uma regra, intuitivamente plausível, denominada *regra da máxima verossimilhança*, definida a seguir.

Definição 2.2.1. *A regra discriminante da máxima verossimilhança para alocar uma observação* $\boldsymbol{x}$ *para uma das populações* $\Pi_1, \ldots, \Pi_g$ *é alocar* $\boldsymbol{x}$ *para a população que dá a maior verossimilhança para* $\boldsymbol{x}$*.*

Isto é, a regra da máxima verossimilhança nos diz que devemos alocar $\boldsymbol{x}$ para Π_j, onde

$$f_j(\boldsymbol{x}) = \max_i f_i(\boldsymbol{x}). \tag{2.2}$$

Por simplicidade devemos assumir que o caso em que várias verossimilhanças tomam o mesmo valor máximo pode ser negligenciado. Formalmente nós postulamos que

$$P\{f_i(\boldsymbol{x}) = f_k(\boldsymbol{x}) \text{ para algum } i \neq k \mid \Pi_j\} = 0,$$

para todo $j = 1, \ldots, g$, tal que a forma da regra de alocação nestes casos não tem importância prática.

2.2.2 Discriminação pela Regra de Bayes

Em certas situações faz sentido supor que as várias populações têm *probabilidades a priori*. Por exemplo, em problemas relacionados à insolvência de dívidas podemos considerar clientes adimplentes mais prováveis que clientes inadimplentes. Essa informação pode ser incorporada na análise através de uma regra discriminante bayesiana. Por simplicidade, vamos supor que todas as probabilidades a priori π_j são estritamente positivas, $j = 1, \ldots, g$. A *regra discriminante de Bayes* é definida a seguir.

Definição 2.2.2. *Se as populações* $\Pi_1, \ldots, \Pi_g$ *têm probabilidades a priori* $(\pi_1, \ldots, \pi_g) = \boldsymbol{\pi}'$*, então a regra discriminante de Bayes (com respeito a* $\boldsymbol{\pi}$*) aloca uma observação* $\boldsymbol{x}$ *para a população para a qual*

$$\pi_j f_j(\boldsymbol{x}) \tag{2.3}$$

é maximizada.

A função (2.3) pode ser considerada como proporcional à verossimilhança a posteriori de Π_j dado $\boldsymbol{x}$. Note que a regra da máxima verossimilhança é um caso particular da regra de Bayes quando todas as probabilidades a priori são iguais.

2.3 Discriminação quando as distribuições são conhecidas, exceto para os valores de um ou mais parâmetros

Nesta seção, abordamos a situação em que as formas das distribuições das populações são conhecidas, mas há parâmetros que são desconhecidos e precisam ser estimados a partir da amostra de dados. Na subseção seguinte apresentamos o método da máxima verossimilhança amostral, bastante útil nessas situações.

2.3.1 Regra da máxima verossimilhança amostral

A *regra discriminante da máxima verossimilhança amostral* é útil quando conhecemos as formas das distribuições das populações $\Pi_1, \ldots, \Pi_g$, mas um ou mais parâmetros são desconhecidos e devem ser estimados com base em uma matrix de dados $\mathbf{X}$ $(n \times p)$. Supomos que as linhas de $\mathbf{X}$ estão particionadas em g grupos, $\mathbf{X}' = (\mathbf{X}_1', \ldots, \mathbf{X}_g')$ e que $\mathbf{X}_i'$ contem n_i observações da população Π_i.

Definição 2.3.1. *A regra discriminante da máxima verossimilhança amostral para alocar uma observação $\boldsymbol{x}$ para uma das populações $\Pi_1, \ldots, \Pi_g$ é alocar $\boldsymbol{x}$ para a população que dá a maior verossimilhança amostral para $\boldsymbol{x}$.*

Isto é, a regra da máxima verossimilhança amostral nos diz que devemos alocar $\boldsymbol{x}$ para Π_j, onde

$$\tilde{f}_j(\boldsymbol{x}) = \max_i \tilde{f}_i(\boldsymbol{x}). \tag{2.4}$$

Por exemplo, suponha que os grupos são amostras de distribuições normais multivariadas com diferentes vetores de médias e mesma matriz de covariâncias. Sejam $\bar{\boldsymbol{x}}_i$ e $\mathbf{S}_i$ a média amostral e a matriz de covariâncias amostral do i-ésimo grupo. Então, estimadores não-viesados de $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_g$ e $\boldsymbol{\Sigma}$ são $\bar{\boldsymbol{x}}_1, \ldots, \bar{\boldsymbol{x}}_g$ e $\mathbf{S} = \sum_i (n_i - 1)\mathbf{S}_i/(n - g)$. A regra da máxima verossimilhança amostral é

então obtida utilizando essas estimativas nos lugares dos parâmetros desconhecidos das verossimilhanças dos grupos. Ou seja, a i-ésima verossimilhança fica escrita na forma

$$\tilde{f}_i(\boldsymbol{x}) = (2\pi)^{-p/2} \left|\mathbf{S}\right|^{-1/2} \exp\left\{-\frac{1}{2}(\boldsymbol{x} - \bar{\boldsymbol{x}}_i)'\mathbf{S}^{-1}(\boldsymbol{x} - \bar{\boldsymbol{x}}_i)\right\},$$

e alocamos $\boldsymbol{x}$ para a população que dá a maior verossimilhança amostral para $\boldsymbol{x}$. Ou seja, a regra discriminante, equação (2.4), se reduz, após tomarmos logarítmos de f_j, a (a constante $-\frac{p}{2}\log(\pi) - \log\left|\mathbf{S}\right|^{-1/2}$ pode ser ignorada já que é a mesma para todas as populações)

$$\text{alocar } \boldsymbol{x} \text{ para } \Pi_j, \text{ se } \Pi_j = \underset{i\in\{1,\ldots,g\}}{\operatorname{argmax}} \left\{-\frac{1}{2}(\boldsymbol{x} - \bar{\boldsymbol{x}}_i)'\mathbf{S}^{-1}(\boldsymbol{x} - \bar{\boldsymbol{x}}_i)\right\}. \tag{2.5}$$

Podemos ainda considerar um enfoque bayesiano se as probabilidades a priori, $(\pi_1, \ldots, \pi_g) = \boldsymbol{\pi}'$, das populações $\Pi_1, \ldots, \Pi_g$ estão disponíveis. Dessa forma, a regra da máxima verossimilhança amostral diz que devemos alocar $\boldsymbol{x}$ para Π_j, onde

$$\pi_j \tilde{f}_j(\boldsymbol{x}) = \max_i \pi_i \tilde{f}_i(\boldsymbol{x}). \tag{2.6}$$

Assim, no caso em que os grupos são amostras de distribuições normais multivariadas com diferentes vetores de médias e mesma matriz de covariâncias, (2.5) fica escrita como

$$\text{alocar } \boldsymbol{x} \text{ para } \Pi_j, \text{ se } \Pi_j = \underset{i\in\{1,\ldots,g\}}{\operatorname{argmax}} \left\{\log(\pi_i) - \frac{1}{2}(\boldsymbol{x} - \bar{\boldsymbol{x}}_i)'\mathbf{S}^{-1}(\boldsymbol{x} - \bar{\boldsymbol{x}}_i)\right\}. \tag{2.7}$$

A regra (2.7) é denominada *regra discriminante normal linear*.

Se relaxarmos a suposição de que as g populações têm mesma matriz de covariância podemos definir uma *regra discriminante normal quadrática*, ou seja

$$\text{alocar } \boldsymbol{x} \text{ para } \Pi_j, \text{ se } \Pi_j = \underset{i\in\{1,\ldots,g\}}{\operatorname{argmax}} \left\{\log(\pi_i) - \frac{1}{2}\log\left|\mathbf{S}_i\right| - \frac{1}{2}(\boldsymbol{x} - \bar{\boldsymbol{x}}_i)'\mathbf{S}_i^{-1}(\boldsymbol{x} - \bar{\boldsymbol{x}}_i)\right\}, \tag{2.8}$$

onde $\mathbf{S}_i$ é a matriz de covariância amostral da i-ésima população.

De uma maneira geral, estimamos os valores dos parâmetros desconhecidos através dos dados amostrais e utilizamos essas estimativas para obter as verossimilhanças amostrais.

Como no caso em que conhecíamos completamente as distribuições das populações, neste também podemos negligenciar os casos em que duas ou mais verossimilhanças tomam o mesmo valor máximo.

2.4 Discriminação quando as distribuições são completamente desconhecidas

Nesta seção, apresentamos a abordagem dada por Fisher (1936) ao problema de discriminação, quando as formas das distribuições das populações são totalmente desconhecidas.

2.4.1 Função discriminante linear de Fisher

Fisher (1936) deu ao problema de discriminação entre g populações o seguinte enfoque: encontrar uma função linear $\boldsymbol{a}'\boldsymbol{x}$ que maximize a razão entre a soma dos quadrados entre grupos e a soma dos quadrados dentro dos grupos, isto é, seja

$$\boldsymbol{z} = \mathbf{X}\boldsymbol{a} = \begin{bmatrix} \mathbf{X}_1\boldsymbol{a} \\ \mathbf{X}_2\boldsymbol{a} \\ \vdots \\ \mathbf{X}_g\boldsymbol{a} \end{bmatrix} = \begin{bmatrix} \boldsymbol{z}_1 \\ \boldsymbol{z}_2 \\ \vdots \\ \boldsymbol{z}_g \end{bmatrix}$$

uma combinação linear das colunas de $\mathbf{X}$. Então $\boldsymbol{z}$ tem soma dos quadrados total

$$\boldsymbol{z}'\mathbf{V}\boldsymbol{z} = \boldsymbol{a}'\mathbf{X}'\mathbf{V}\mathbf{X}\boldsymbol{a} = \boldsymbol{a}'\mathbf{T}\boldsymbol{a},$$

que pode ser particionada como a soma dos quadrados dentro dos grupos

$$\sum_i \boldsymbol{z}_i'\mathbf{V}_i\boldsymbol{z}_i = \sum_i \boldsymbol{a}'\mathbf{X}_i'\mathbf{V}_i\mathbf{X}_i\boldsymbol{a} = \boldsymbol{a}'\mathbf{S}\boldsymbol{a},$$

mais a soma dos quadrados entre grupos

$$\sum_i n_i(\bar{z}_i - \bar{z})^2 = \sum_i n_i\{\boldsymbol{a}'(\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}})\}^2 = \boldsymbol{a}'\mathbf{B}\boldsymbol{a},$$

onde $\bar{\boldsymbol{z}}_i$ é a média do i-ésimo sub-vetor $\boldsymbol{z}_i$ de $\boldsymbol{z}$ e $\mathbf{V}_i$ $(n_i \times n_i)$ é a matrix centro.

O critério de Fisher é bastante atraente porque é mais fácil distinguir um grupo do outro se a soma dos quadrados entre grupos para $\boldsymbol{z}$ é grande com relação à soma dos quadrados dentro dos grupos. A razão entre a soma dos quadrados dentro e entre os grupos é dada por

$$\frac{\boldsymbol{a}'\mathbf{B}\boldsymbol{a}}{\boldsymbol{a}'\mathbf{S}\boldsymbol{a}}. \tag{2.9}$$

Se $\boldsymbol{a}$ é o vetor que maximiza a expressão (2.9) nós chamamos a função linear $\boldsymbol{a}'\boldsymbol{x}$ de *função discriminante linear de Fisher*. Note que $\boldsymbol{a}$ pode sofrer mudança de escala sem que (2.9) sofra alteração.

Pode-se provar que o vetor $\boldsymbol{a}$ em (2.9) é o autovetor associado ao maior autovalor de $\mathbf{S}^{-1}\mathbf{B}$; ver, por exemplo, Johnson & Wichern (1998, pp. 685–686).

Uma vez que a função discriminante linear foi calculada, uma observação $\boldsymbol{x}$ pode ser alocada para uma das g populações com base em seu "escore discriminante" $\boldsymbol{a}'\boldsymbol{x}$. A média amostral $\bar{\boldsymbol{x}}_i$ tem escore $\boldsymbol{a}'\bar{\boldsymbol{x}}_i = \bar{z}_i$. Então, $\boldsymbol{x}$ é alocado para a população cujo escore médio é próximo de $\boldsymbol{a}'\boldsymbol{x}$; isto é, alocar $\boldsymbol{x}$ para Π_j se

$$\left|\boldsymbol{a}'\boldsymbol{x} - \boldsymbol{a}'\bar{\boldsymbol{x}}_j\right| < \left|\boldsymbol{a}'\boldsymbol{x} - \boldsymbol{a}'\bar{\boldsymbol{x}}_i\right|, \qquad \forall i \neq j.$$

A função discriminante linear de Fisher é muito importante no caso especial em que $g = 2$ grupos. Nesse caso, $\mathbf{B}$ tem posto 1 e pode ser escrita como

$$\mathbf{B} = \left(\frac{n_1 n_2}{n}\right) \boldsymbol{d}'\boldsymbol{d},$$

em que $\boldsymbol{d} = \bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2$. Desse modo, $\mathbf{S}^{-1}\mathbf{B}$ tem somente um autovalor não-nulo, o qual pode ser encontrado explicitamente. Este autovalor é igual a

$$\mathrm{tr}(\mathbf{S}^{-1}\mathbf{B}) = \left(\frac{n_1 n_2}{n}\right) \boldsymbol{d}'\mathbf{S}^{-1}\boldsymbol{d}$$

e o autovetor correspondente é $\boldsymbol{a} = \mathbf{S}^{-1}\boldsymbol{d}$. Então a regra discriminante para alocar uma observação $\boldsymbol{x}$ a uma das populações torna-se

$$\text{alocar } \boldsymbol{x} \text{ para } \Pi_1, \text{ se } \boldsymbol{d}'\mathbf{S}^{-1}\left\{\boldsymbol{x} - \frac{1}{2}(\bar{\boldsymbol{x}}_1 + \bar{\boldsymbol{x}}_2)\right\} > 0 \tag{2.10}$$

alocar $\boldsymbol{x}$ para Π_2, caso contrário.

Como $\boldsymbol{d} = \bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2$, podemos ainda reescrever a regra (2.10) como

$$\text{alocar } \boldsymbol{x} \text{ para } \Pi_1, \text{ se } z = (\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)'\mathbf{S}^{-1}\boldsymbol{x} \geq \widehat{m} = \frac{1}{2}(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)'\mathbf{S}^{-1}(\bar{\boldsymbol{x}}_1 + \bar{\boldsymbol{x}}_2) \tag{2.11}$$

alocar $\boldsymbol{x}$ para Π_2, caso contrário.

A regra discriminante de Fisher é equivalente à regra da máxima verossimilhança amostral no caso em que as populações são amostras de distribuições normais multivariadas com diferentes vetores de médias e mesma matriz de covariâncias, assumindo que as probabilidades a priori das populações são iguais.

CAPÍTULO 3

Análise Discriminante de Núcleo

3.1 Introdução

No capítulo anterior discutimos o problema da discriminação entre g grupos ou populações através da análise discriminante clássica. No caso em que conhecemos as funções densidade de probabilidade de cada população, $\Pi_1, \ldots, \Pi_g$, $g \geq 2$, e suas probabilidades a priori $\pi_1, \ldots, \pi_g$, alocamos uma observação $\boldsymbol{x}$ para a população para a qual a quantidade $\pi_j f_j(\boldsymbol{x})$, $j = 1, \ldots, g$, é maximizada.

Especificamente, no caso em que não conhecemos as formas paramétricas das funções densidade de probabilidade das populações, abordamos o método proposto por Fisher (1936), que tenta encontrar uma combinação linear entre as variáveis, $\boldsymbol{a}'\boldsymbol{x}$, que forneça a máxima separabilidade entre as populações.

Neste capítulo trataremos do caso em que não conhecemos as formas paramétricas das funções densidade de probabilidade das populações através do método proposto por Fix & Hodges (1951), baseado na estimação das funções densidade de probabilidade de cada população através de uma classe de estimadores de densidades denominada estimadores de núcleo. Nesse caso, alocamos uma observação $\boldsymbol{x}$ para a população para a qual

$$\hat{\pi}_j \hat{f}_j(\boldsymbol{x}) \tag{3.1}$$

é maximizada, sendo $\hat{f}_j(\boldsymbol{x})$ a estimativa da densidade pelo método de núcleo, correspondente à j-ésima população e $\hat{\pi}_j$ a estimativa da probabilidade a priori da j-ésima população.

Na seção 3.2 abordaremos os principais aspectos da estimação de densidades através do método de núcleo, tais como os critérios de erro utilizados, os métodos de escolha dos parâmetros de suavização tanto no caso de estimação de densidades univariadas quanto no caso de estimação de densidades multivariadas e a escolha da função núcleo apropriada.

Na seção 3.3 descreveremos o método da análise discriminante de núcleo proposto por Fix & Hodges (1951), baseado na estimação das densidades dos g grupos.

3.2 Estimação de Densidades através do Método de Núcleo

A *função densidade de probabilidade* é um conceito fundamental em estatística. Considere alguma quantidade aleatória X com função densidade de probabilidade f, onde f é uma função não negativa e $\int_{-\infty}^{+\infty} f(x)dx = 1$. Especificar a função densidade de X nos fornece uma descrição natural da sua distribuição, e permite que probabilidades associadas a X possam ser encontradas através da relação

$$P(a < X < b) = \int_a^b f(x)dx \qquad \text{para todo } a < b. \tag{3.2}$$

O método de núcleo tem sido um dos procedimentos mais utilizados em estimação não-paramétrica de curvas. Existe um considerável número de situações onde essa metodologia tem se mostrado bastante eficaz. Sua teoria está bem desenvolvida e tem ajudado a entender muitas características no campo não-paramétrico.

Estimação de funções densidade de probabilidade pode ser considerada uma das mais simples situações onde o método de núcleo pode ser inserido. Historicamente, a fim de reduzir o custo computacional relacionado com a estimação de densidades, uma forma funcional ou paramétrica é imposta aos dados. Esta forma paramétrica pode ser bastante subjetiva, contudo, na maioria das situações, sua imposição permite simplificar bastante o problema, pois tudo que resta é estimar os parâmetros através da amostra de dados. Estes parâmetros estimados mais a forma paramétrica imposta fornecem um estimador paramétrico de densidades. Os estimadores

paramétricos de densidade mais comuns são os estimadores de máxima verossimilhança, que são úteis em um grande número de situações. Entretanto, existem ainda situações onde esses estimadores paramétricos não são aplicáveis e nesses casos o uso de estimadores não-paramétricos de densidades é apropriado.

Certamente, o histograma é o método não-paramétrico mais antigo e mais utilizado de estimação de densidades. É importante ter em mente que o histograma é uma técnica de suavização de dados usada para estimar uma densidade desconhecida e dessa forma deve ser analisado. Dada uma origem x_0 e um comprimento de intervalo h, definimos as "caixas" do histograma como sendo os intervalos $[x_0+(r-1)h, x_0+rh)$ para valores inteiros positivos e negativos de r. Os intervalos devem ser escolhidos fechados à esquerda e abertos à direita por definição. Empiricamente a idéia e contar o número de observações que estão contidas em cada intervalo. Sem perda de generalidade, considere o intervalo $[-h/2, h/2)$. A probabilidade de uma observação pertencer ao intervalo $[-h/2, h/2)$ é dada por

$$P(X \in [-h/2, h/2)) = \int_{-h/2}^{h/2} f(x)dx,$$

onde f é a densidade de X.

Um estimador natural da densidade f pode ser pensado como o número de observações que estão contidas em cada intervalo divido pelo número total de observações. Ou seja, dado um conjunto de observações $X_1, X_2, \ldots, X_n$, temos

$$P(X \in [-h/2, h/2)) \approx \frac{1}{n}\#\{X_i \in [-h/2, h/2)\}.$$

Ou seja,

$$P(X \in [-h/2, h/2)) = \int_{-h/2}^{h/2} f(x)dx.$$

Dessa forma, uma estimativa para f seria

$$\hat{f}_h(x) = \frac{1}{nh}\#\{X_i \in [-h/2, h/2)\}, \tag{3.3}$$

para todo $x \in [-h/2, h/2)$.

Formalmente, suponha que observamos $X_1, \ldots, X_n$, independentes e identicamente distribuídas (i.i.d.), com densidade desconhecida f. Seja k o número de intervalos de comprimento h

e defina $C_r = [x_0 + (r-1)h, x_0 + rh)$, $r = 1, \ldots, k$. Tome $n_r = \sum_{i=1}^{n} I(X_i \in C_r)$, tal que, $\sum_{r=1}^{k} n_r = n$. Dessa forma,

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{r=1}^{k} n_r I(x \in C_r), \tag{3.4}$$

onde a função $I(x \in A)$ é dada por

$$I(x \in A) = \begin{cases} 1 & \text{se } x \in A \\ 0 & \text{caso contrário.} \end{cases}$$

Vale notar que a estimativa $\hat{f}_h(x)$ depende fortemente da escolha do parâmetro de suavização h. Variando o valor de h obtemos diferentes formas de $\hat{f}_h(x)$. Por exemplo, quando aumentamos o valor de h temos intervalos maiores e o histograma será uma representação mais suave. Nos extremos, digamos, quando $h \to 0$, o histograma torna-se uma representação muito ruidosa dos dados. Na situação oposta, quando $h \to \infty$, o histograma torna-se uma representação muito suave dos dados.

Podemos pensar em um estimador de densidades mais geral, baseado na idéia do histograma. Para isso, considere a seguinte função peso

$$w(x) = \begin{cases} \frac{1}{2} & \text{se } |x| < 1 \\ 0 & \text{caso contrário.} \end{cases} \tag{3.5}$$

Então, é fácil ver que uma estimativa para f neste caso é dada por

$$\hat{f}(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{h} w\left(\frac{x - X_i}{h}\right) \tag{3.6}$$

A partir de (3.5) podemos notar que o estimador (3.6), amplamente conhecido como estimador "ingênuo" (ver Silverman (1986, pp. 11-13) para maiores detalhes sobre este tipo de estimador), é construído colocando-se uma "caixa" de largura $2h$ e altura $(2nh)^{-1}$ em cada observação e então somando para obter a estimativa $\hat{f}$. Não é difícil notar que $\hat{f}$ não é uma função contínua e tem derivada nula em todos os pontos exceto nos pontos de salto $X \pm h$.

Podemos generalizar o estimador (3.6) de modo a superar algumas das dificuldades relacionadas a este tipo de estimador de densidades. O estimador de densidades baseado em uma função núcleo é obtido substituindo a função peso w por uma função núcleo não-negativa K que satisfaça a condição

$$\int_{-\infty}^{+\infty} K(x)dx = 1 \tag{3.7}$$

Usualmente, mas não sempre, K será uma função densidade de probabilidade simétrica, como, por exemplo, a função densidade de probabilidade normal. Neste caso particular, $\hat{f}$ será uma curva suave com derivadas de todas as ordens.

O estimador de núcleo no caso univariado, para uma amostra aleatória $X_1, \ldots, X_n$ retirada de uma distribuição com densidade comum f, pode ser definido como

$$\hat{f}(x; h) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right), \tag{3.8}$$

onde h é o parâmetro de suavização, positivo e não-aleatório, também chamado por alguns autores de largura da janela ou largura de banda, e K é a função núcleo. Exemplos de funções núcleo univariadas são dados na tabela 3.1. Uma função núcleo com massa de probabilidade n^{-1} é colocada em cada observação. Estas são então somadas para fornecer a curva composta. O grau de suavização é controlado pelo parâmetro de suavização h. Dessa forma, uma tarefa crucial em estimação de densidades é a escolha desse parâmetro.

Tabela 3.1: Funções núcleo comumente utilizadas com dados univariados.

Função Núcleo	Forma analítica, $K(x)$
Retangular	$\frac{1}{2}$ para $\lvert x\rvert < 1$, 0 caso contrário
Triangular	$1 - \lvert x\rvert$ para $\lvert x\rvert < 1$, 0 caso contrário
Biweight	$\frac{15}{16}(1 - x^2)^2$ para $\lvert x\rvert < 1$, 0 caso contrário
Normal	$\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right)$
Epanechnikov	$\frac{3}{4}\left(1 - x^2/5\right)/\sqrt{5}$ para $\lvert x\rvert < \sqrt{5}$, 0 caso contrário

A extensão para dados multivariados é direta, com o estimador de densidades p-dimensional, para uma amostra aleatória $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_n$ retirada de uma densidade comum f, definido por

$$\hat{f}(\boldsymbol{x}) = \frac{1}{nh^p} \sum_{i=1}^{n} K\left(\frac{1}{h}(\boldsymbol{x} - \boldsymbol{X}_i)\right), \tag{3.9}$$

onde $\boldsymbol{x} = (x_1, x_2, \ldots, x_p)'$ e $\boldsymbol{X}_i = (X_{i1}, X_{i2}, \ldots, X_{ip})'$, $i = 1, 2, \ldots, n$. A função núcleo multivariada $K(\boldsymbol{x})$ é agora uma função definida no espaço p-dimensional, satisfazendo

$$\int_{\mathbb{R}^p} K(\boldsymbol{x}) d\boldsymbol{x} = 1, \tag{3.10}$$

e h é o parâmetro de suavização.

Usualmente K será uma função densidade de probabilidade unimodal radialmente simétrica, como, por exemplo, a distribuição normal padrão multivariada

$$K(\boldsymbol{x}) = (2\pi)^{-p/2} \exp\left(-\frac{1}{2}\boldsymbol{x}'\boldsymbol{x}\right),$$

ou a função núcleo Epanechnikov

$$K(\boldsymbol{x}) = \begin{cases} \frac{(1-\boldsymbol{x}'\boldsymbol{x})(p+2)}{2c_p} & \text{para } |\boldsymbol{x}| < 1 \\ 0 & \text{caso contrário,} \end{cases}$$

onde $c_p = \frac{\pi^{p/2}}{\Gamma((p/2)+1)}$ é o volume de uma esfera unitária p-dimensional.

Uma forma da estimativa da função de densidade de probabilidade comumente utilizada é a soma do produto de funções núcleo (contudo, isto não implica independência das variáveis)

$$\hat{f}(\boldsymbol{x}) = \frac{1}{n} \frac{1}{h_1 \cdots h_p} \sum_{i=1}^{n} \prod_{j=1}^{p} K_j \left(\frac{[\boldsymbol{x} - \boldsymbol{X}_i]_j}{h_j} \right), \tag{3.11}$$

onde existem diferentes parâmetros de suavização associados com cada variável. Podemos assumir alguma das formas univariadas apresentadas na tabela 3.1 para os K_j, $j = 1, \ldots, p$. Usualmente a mesma forma é assumida para todos os K_j.

De uma maneira mais geral, podemos tomar

$$\hat{f}(\boldsymbol{x}) = \hat{f}(\boldsymbol{x}; \mathbf{H}) = \frac{1}{n} \sum_{i=1}^{n} K_{\mathbf{H}}(\boldsymbol{x} - \boldsymbol{X_i}), \tag{3.12}$$

onde $K_{\mathbf{H}} = |\mathbf{H}|^{-1/2} K(\mathbf{H}^{-1/2}\boldsymbol{x})$ é a função núcleo transformada e $\mathbf{H}$ é uma matrix não-aleatória, simétrica, positiva-definida, denominada matriz de largura de banda. No contexto de classificação, usualmente toma-se $\mathbf{H} = h_k^2 \hat{\boldsymbol{\Sigma}}_k$ para o grupo Π_k, onde h_k é uma fator de escala para o grupo Π_k e $\hat{\boldsymbol{\Sigma}}_k$ é a matriz de covariância amostral do k-ésimo grupo. Entretanto, existem formas mais sofisticadas de escolha do parâmetro de suavização, baseados em métodos como mínimos

quadrados por validação cruzada e verossimilhança por validação cruzada. Duong (2004) apresenta um extensivo estudo sobre as propriedades e o desempenho desses métodos de seleção de matrizes de largura de banda em problemas de estimação de densidades e de classificação.

3.2.1 Critérios de Erro

Várias medidas de discrepância entre o estimador de densidades $\hat{f}$ e a densidade verdadeira f têm sido estudadas. Quando consideramos estimação em um único ponto uma medida natural é o *erro quadrático médio* (EQM), definido por

$$\mathrm{EQM}_{\boldsymbol{x}}(\hat{f}(\cdot;\mathbf{H})) = E\{\hat{f}(\boldsymbol{x};\mathbf{H}) - f(\boldsymbol{x})\}^2. \tag{3.13}$$

Utilizando propriedades elementares da esperança e da variância podemos escrever (3.13) como

$$\mathrm{EQM}_{\boldsymbol{x}}(\hat{f}(\cdot;\mathbf{H})) = \{E[\hat{f}(\boldsymbol{x};\mathbf{H})] - f(\boldsymbol{x})\}^2 + Var[\hat{f}(\boldsymbol{x};\mathbf{H})], \tag{3.14}$$

que é a soma do quadrado do viés e da variância de $\hat{f}$ em $\boldsymbol{x}$. Veremos que, como em muitos ramos da estatística, existe um compromisso entre viés e variância em (3.14); o viés pode ser reduzido com o custo de aumentarmos a variância. Em contrapartida, não podemos reduzir a variância sem com isso aumentar o viés.

Um critério de erro global comum é o *erro quadrático integrado* (EQI) que é a distância quadrada integrada entre uma estimativa $\hat{f}$ e a verdadeira densidade f, ou seja,

$$\mathrm{EQI}(\hat{f}(\cdot;\mathbf{H})) = \int_{\mathbb{R}^p} [\hat{f}(\boldsymbol{x};\mathbf{H}) - f(\boldsymbol{x})]^2 d\boldsymbol{x}. \tag{3.15}$$

Entretanto, o EQI é uma variável aleatória e predizer seu valor não é simples. Uma alternativa ao EQI é o *erro quadrático integrado médio* (EQIM) [Rosenblatt (1956)], definido como

$$\mathrm{EQIM}(\hat{f}(\cdot;\mathbf{H})) = E\{\mathrm{EQI}(\hat{f}(\boldsymbol{x};\mathbf{H}))\} = E\left\{\int_{\mathbb{R}^p} [\hat{f}(\boldsymbol{x};\mathbf{H}) - f(\boldsymbol{x})]^2 d\boldsymbol{x}\right\}. \tag{3.16}$$

O EQIM é a forma mais largamente utilizada para medir a acurácia global de $\hat{f}$ como um estimador de f.

Embora existam outras medidas globais de discrepância que podem ser mais apropriadas, o EQIM é, sem dúvidas, a medida global cuja forma é a mais tratável, sendo então bastante válido considerá-lo como medida de discrepância [Silverman (1986, pp. 31–32)].

Vale notar que, como o integrando em (3.16) é não-negativo, a ordem de integração e da esperança pode ser invertida para obtermos formas alternativas, ou seja,

$$\begin{aligned} \mathrm{EQIM}(\hat{f}(\cdot;\mathbf{H})) &= \int_{\mathbb{R}^p} E\{\hat{f}(\boldsymbol{x};\mathbf{H}) - f(\boldsymbol{x})\}^2 d\boldsymbol{x} \\ &= \int_{\mathbb{R}^p} \mathrm{EQM}[\hat{f}(\boldsymbol{x};\mathbf{H})] d\boldsymbol{x}. \end{aligned} \tag{3.17}$$

A expressão (3.17) pode ainda ser escrita como

$$\mathrm{EQIM}(\hat{f}(\cdot;\mathbf{H})) = \int_{\mathbb{R}^p} \{E[\hat{f}(\boldsymbol{x};\mathbf{H})] - f(\boldsymbol{x})\}^2 d\boldsymbol{x} + \int_{\mathbb{R}^p} Var[\hat{f}(\boldsymbol{x};\mathbf{H})] d\boldsymbol{x}, \tag{3.18}$$

que é o EQIM escrito como a soma da integral do quadrado do viés e da integral da variância.

3.2.2 Escolha do Parâmetro de Suavização

O desempenho de um estimador de densidades pode ser medido através de algum dos critérios de erro descritos na subseção 3.2.1. O método de seleção do parâmetro de suavização desempenha um papel fundamental na determinação do desempenho do estimador de densidades pelo método de núcleo. Desse modo, desejamos selecionar parâmetros de suavização que forneçam desempenho ótimo. Assim, podemos estar interessados em encontrar $\mathbf{H}$ tal que o EQIM seja minimizado, ou seja,

$$\mathbf{H}_{\mathrm{EQIM}} = \underset{\mathbf{H}\in\mathcal{H}}{\operatorname{argmin}}\, \mathrm{EQIM}(\hat{f}(\cdot;\mathbf{H})), \tag{3.19}$$

onde $\mathcal{H}$ é o espaço das matrizes simétricas, positivas-definidas de dimensão $(p \times p)$. Contudo, o EQIM apresenta forma fechada apenas se f é uma mistura de distribuições normais e K é a função núcleo normal, e dessa forma, encontrar $\mathbf{H}_{\mathrm{EQIM}}$ é, em geral, extremamente difícil [Wand & Jones (1995)].

O *erro quadrático integrado médio assintótico* (EQIMA) é uma aproximação assintótica do *erro quadrático integrado médio* (EQIM). A seguinte expressão para o EQIMA foi derivada por Wand & Jones (1995, pp. 94–101) (expressões alternativas para o EQIM e para o EQIMA podem ser encontradas em Duong (2004, pp. 5–7))

$$\mathrm{EQIMA}(\hat{f}(\cdot;\mathbf{H})) = \frac{1}{n} R(K)|\mathbf{H}|^{-1/2} + \frac{1}{4}\mu_2(K)^2 \int_{\mathbb{R}^p} \mathrm{tr}^2(\mathbf{H}D^2 f(\boldsymbol{x})) d\boldsymbol{x}, \tag{3.20}$$

onde $R(v) = \int_{\mathbb{R}^p} v(\boldsymbol{x})^2$ para alguma função integrável quadrada v; $\mu_2(K)\mathbf{I}_p = \int_{\mathbb{R}^p} \boldsymbol{x}\boldsymbol{x}'K(\boldsymbol{x})$, com $\mu_2(K) < \infty$ e $\mathbf{I}_p$ é a matriz identidade de dimensão $(p \times p)$; e $D^2 f(\boldsymbol{x})$ é a matrix Hessiana de f.

Devemos então encontrar um estimador do EQIM(A), $\widehat{\text{EQIM(A)}}$, a partir dos dados disponíveis e a partir desse estimador encontrar um parâmetro de suavização $\widehat{\mathbf{H}}$ tal que

$$\widehat{\mathbf{H}} = \underset{\mathbf{H}\in\mathcal{H}}{\operatorname{argmin}}\, \widehat{\text{EQIM(A)}}. \tag{3.21}$$

A expressão (3.21) é chamada de *seletor do parâmetro de suavização.* $\widehat{\mathbf{H}}$ funciona como um substituto para $\mathbf{H}_{\text{EQIM(A)}}$. Uma revisão de vários métodos que têm sido utilizados na busca por seletores do paramêtro de suavização guiados pelos dados, baseados em vários estimadores do EQIM(A) pode ser encontrada em Duong (2004). Várias parametrizações para matrizes de largura de banda no caso de amostras bivariadas foram consideradas em Wand & Jones (1993).

3.3 Análise Discriminante de Núcleo

Muitos autores têm mostrado a utilidade e a importância de estimação de densidades em vários problemas práticos [Silverman (1986); Scott (1992); Simonoff (1996); Wand & Jones (1995)]. Como mencionamos no início deste capítulo, a intenção original de estimação de densidades [Fix & Hodges (1951)] foi propôr um enfoque não-paramétrico para a análise discriminante como uma alternativa à abordagem clássica.

A regra discriminante de núcleo pode ser definida como

Definição 3.3.1. *A regra discriminante de núcleo para alocar uma observação $\boldsymbol{x}$ para uma das populações $\Pi_1, \ldots, \Pi_g$ é alocar $\boldsymbol{x}$ para a população que maximiza $\hat{\pi}_j \hat{f}_j(\boldsymbol{x}; \mathbf{H}_j)$, $j = 1, \ldots, g$.*

Isto é, a regra discriminante de núcleo nos diz que devemos alocar $\boldsymbol{x}$ para Π_0, se

$$\Pi_0 = \underset{j\in\{1,\ldots,g\}}{\operatorname{argmax}}\, \hat{\pi}_j \hat{f}_j(\boldsymbol{x}; \mathbf{H}_j), \tag{3.22}$$

onde $\hat{f}_j(\boldsymbol{x}; \mathbf{H}_j)$ é o estimador da densidade da j-ésima população pelo método de núcleo, que é obtido a partir de (3.12).

Utilizando a regra discriminante de núcleo definida acima podemos então definir um algoritmo para análise discriminante de núcleo. Este algoritmo será apresentado na próxima subseção.

3.3.1 Algorítmo para Análise Discriminante de Núcleo

Um algoritmo para análise discriminante de núcleo pode ser descrito pelos seguintes passos:

1. Para cada amostra de treinamento $\boldsymbol{X}_j = \{\boldsymbol{X}_{j1}, \boldsymbol{X}_{j2}, \ldots, \boldsymbol{X}_{jn_j}\}$, $j = 1, \ldots, g$, calcule a estimativa da densidade

$$\hat{f}(\boldsymbol{x}; \mathbf{H}_j) = \frac{1}{n_j} \sum_{i=1}^{n_j} K_{\mathbf{H}_j}(\boldsymbol{x} - \boldsymbol{X}_{ji}),$$

   onde $\mathbf{H}_j$ deve ser escolhida através de algum método sensível.

2. Se as probabilidades a priori das classes estão disponíveis, utilize-as. Caso contrário, estime-as através das proporções amostrais no conjunto de treinamento, $\hat{\pi}_j = n_j/n$, $j = 1, \ldots, g$.

3. (a) Classifique as observações do conjunto de teste de acordo com a definição (3.3.1) (equação (3.22)).

   (b) Classifique as observações do conjunto de treinamento de acordo com a definição (3.3.1) (equação (3.22)).

4. Estime a taxa de má-classificação para os conjuntos de teste e de treinamento.

Como uma ilustração do método da análise discriminante de núcleo, considere o conjunto de dados retirado de Johnson & Wichern (1998, p. 658). Este conjunto contém medidas do diâmetro dos anéis de 100 salmões no primeiro ano de crescimento em água doce (X_1) e em água do mar (X_2) capturados em águas do estado americano do Alasca ou em águas do Canadá (50 observações para cada população).

O gráfico de dispersão apresentado na figura 3.1 nos mostra como os grupos estão distribuídos em função das variáveis consideradas. Na figura 3.2 podemos observar as regiões de classificação definidas pela regra discriminante linear, já na figura 3.3 podemos observar as regiões de classificação definidas pela regra discriminante quadrática e por fim, na figura 3.4 podemos observar as regiões de classificação definidas pela regra discriminante de núcleo. Podemos notar que esta última apresenta claramente uma maior separação dos dois grupos, porque as formas das regiões

são mais adequadas para este conjunto de dados. As taxas de erro aparentes para as regras discriminantes linear, quadrática e de núcleo foram, respectivamente, 7%, 7% e 5%.

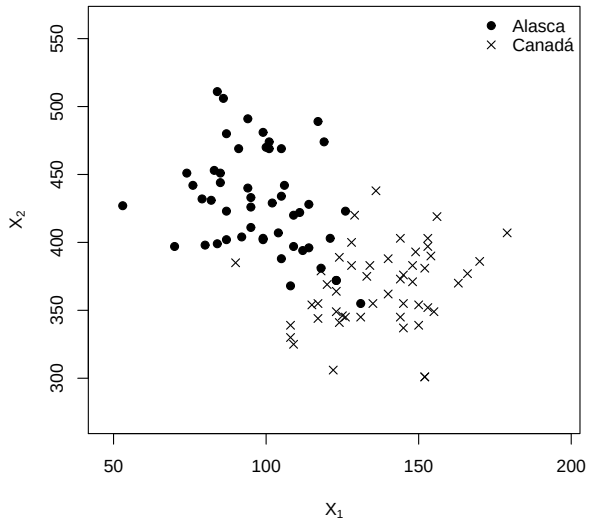

Figura 3.1: Gráfico de dispersão para o conjunto de dados de salmões.

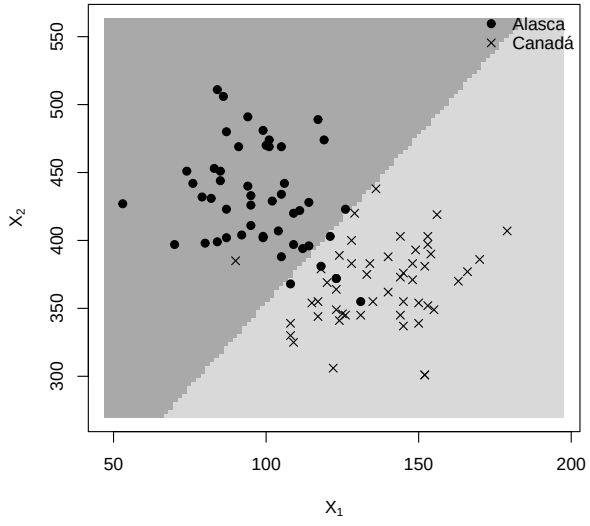

Figura 3.2: Regiões de classificação definidas pela regra discriminante linear.

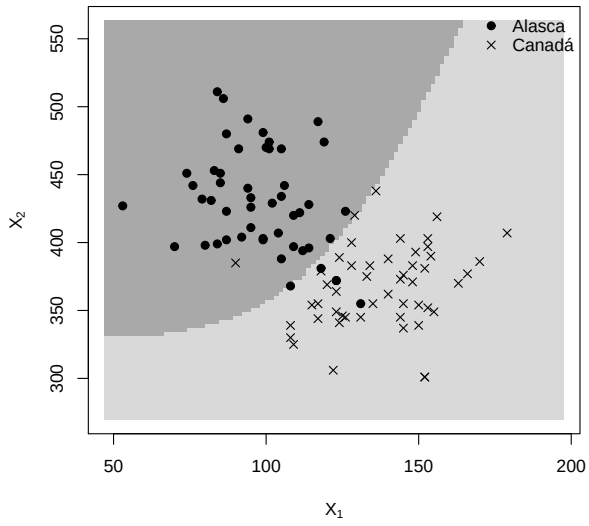

Figura 3.3: Regiões de classificação definidas pela regra discriminante quadrática.

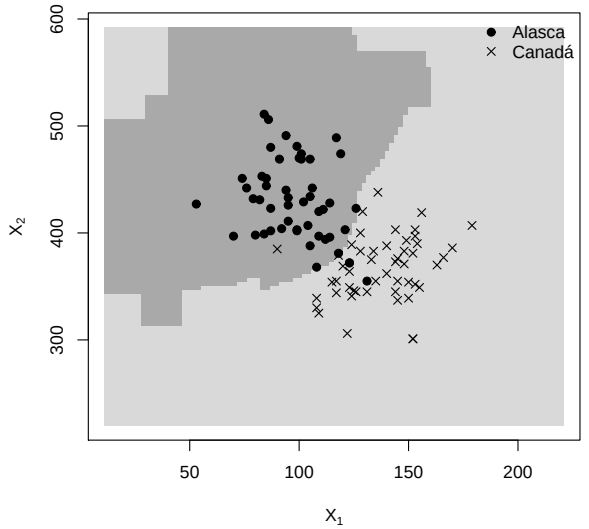

Figura 3.4: Regiões de classificação definidas pela regra discriminante de núcleo.

CAPÍTULO 4

Avaliação do Desempenho de um Classificador

4.1 Introdução

Existem muitas formas de avaliar o desempenho de um classificador e a mais comumente utilizada é a *taxa de erro* ou *taxa de má-classificação* [Webb (2002)], que representa a proporção de padrões que foram incorretamente classificados. Em adição ao cálculo da taxa de erro, podemos também, construir uma *matriz de confusão* ou *matriz de má-classificação*. O (i, j)-ésimo elemento dessa matriz é o número de padrões da classe j que foram classificados como pertencentes à classe i. Essa matriz pode ser útil na visualização de como a taxa de erro se decompõe.

Seja F o classificador construído utilizando-se o conjunto de treinamento T definida na seção 1.2. Considere a função perda $L(y, F(\boldsymbol{x}))$ chamada *função perda 0-1* e definida por

$$L(y, F(\boldsymbol{x})) = \begin{cases} 0 & \text{se } y = F(\boldsymbol{x}) \\ 1 & \text{caso contrário.} \end{cases} \tag{4.1}$$

A *taxa de erro aparente* ou *taxa de erro de treinamento*, $\overline{err}$, é obtida utilizando o conjunto de treinamento para estimar a taxa de erro, ou seja,

$$\overline{err} = n^{-1} \sum_{i=1}^{n} L(y_i, F(\boldsymbol{x}_i)). \tag{4.2}$$

Na maioria das situações a taxa de erro aparente será menor que a taxa de erro verdadeira $\text{Err} = E[L(Y, F(\boldsymbol{X}))]$, que é a probabilidade esperada de classificar incorretamente um padrão

selecionado aleatoriamente. Essa probabilidade é a taxa de erro em um conjunto de teste independente, infinitamente grande, retirado de uma população com a mesma distribuição do conjunto de treinamento. Tipicamente o classificador se adapta aos dados de treinamento, e portanto, o erro aparente ou erro de treinamento será uma estimativa otimista da taxa de erro verdadeira.

Existem algumas maneiras de obter estimativas mais realistas da taxa de erro. Em situações em que dispomos de grandes conjuntos de padrões podemos particionar, aleatoriamente, o conjunto de dados em dois subconjuntos mutuamente excludentes de tal forma que um deles seja utilizado na construção da regra de classificação (conjunto de treinamento) e o outro (conjunto de teste) na avaliação do desempenho da regra construída. Esse procedimento, usualmente denominado *treinamento-e-teste*, tem sido bastante difundido na prática e será discutido por nós na seção 4.2.

Em situações onde não dispomos de grandes conjuntos de dados outras formas de calcular a taxa de erro, como os métodos *cross-validation* e *bootstrap*, devem ser utilizadas, já que a partição do conjunto de dados em dois subconjuntos reduziria ainda mais o tamanho do conjunto de treinamento podendo acarretar em uma regra de classificação ineficiente. Os métodos cross-validation e bootstrap serão discutidos nas seções 4.3 e 4.4, respectivamente.

4.2 Treinamento-e-Teste

Este método, também denominado método *holdout*, é o mais simples e mais largamente utilizado para avaliar o desempenho de classificadores. Consiste em particionar o conjunto de padrões, de forma aleatória, em dois subconjuntos mutuamente excludentes: conjunto de treinamento e conjunto de teste. A idéia básica é que desejamos construir o classificador utilizando um conjunto de padrões cujos grupos verdadeiros são conhecidos e classificar novos padrões sem o benefício de conhecer a que grupos eles pertencem. Dessa forma, utilizamos o conjunto de treinamento para construir a regra de classificação e avaliamos seu desempenho através do conjunto de teste.

É difícil estabeler uma regra geral para escolha do número de padrões para cada um dos dois conjuntos. Comumente seleciona-se aleatoriamente entre 20% e 30% dos padrões para o conjunto

de teste e o restante dos dados é então utilizado na construção da regra de classificação.

4.3 K-fold Cross-Validation

Cross-validation vem sendo bastante utilizado por pesquisadores na estimação de erros de predição [Efron (1983); Efron & Tibshirani (1993); Stone (1974); Stone (1977)], sendo bastante útil em problemas de classificação. Este método estima diretamente a taxa de erro verdadeira, que é o erro de generalização $\text{Err} = E[L(Y, F(\boldsymbol{X}))]$ quando aplicamos o classificador F a uma amostra de teste independente com mesma distribuição que o conjunto de treinamento.

Idealmente, se dispomos de uma amostra de padrões suficientemente grande, podemos utilizar o procedimento treinamento-e-teste descrito na seção anterior para avaliar o desempenho do nosso classificador. Contudo, há muitas situações em que os dados são escassos. O método K-fold Cross-Validation utiliza uma parte dos dados disponíveis para construir o classificador e outra parte diferente dos dados para testá-lo. O procedimento consiste em particionar a amostra em K partes de tamanhos aproximadamente iguais; por exemplo, para $K = 5$, teremos o cenário ilustrado na figura 4.1.

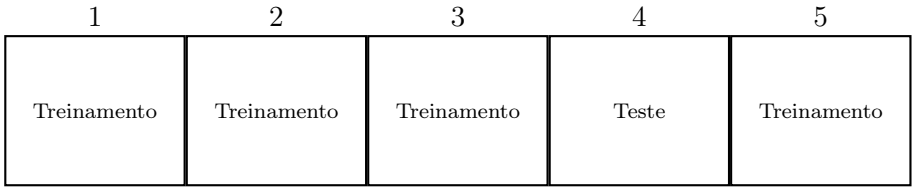

Figura 4.1: Ilustração do método K-fold Cross-Validation com $K = 5$.

Para a k-ésima parte (quarta parte na ilustração acima), nós construímos o classificador utilizando as outras $K-1$ partes, e calculamos a taxa de erro do classificador construído quando predizemos a k-ésima parte. Este procedimento é repetido para $k = 1, \ldots, K$ e a taxa de erro final será uma combinação das taxas de erro calculadas em cada uma das K partes.

De uma maneira mais detalhada, suponha que particionamos a amostra em K partes de tamanhos aproximadamente iguais. Seja $k(i)$ a parte contendo a observação i. Denote por $F^{-k(i)}$ o classificador construído com a k-ésima parte da amostra removida. Então, o estimador

cross-validation da taxa de erro é

$$CV = \frac{1}{n}\sum_{i=1}^{n} L(y_i, F^{-k(i)}(\boldsymbol{x}_i)). \tag{4.3}$$

Escolhas típicas para K são $K = 5$ ou $K = 10$. O caso em que $K = n$ é conhecido como método *leave-one-out* cross-validation [Lachenbruch & Mickey (1968)]. Neste caso particular $k(i) = i$, e para a i-ésima observação o classificador é construído usando toda a amostra exceto a observação i. Com $K = n$, CV é uma estimativa aproximadamente não-viesada da taxa de erro verdadeira, mas pode ter grande variância já que os n conjuntos de treinamento serão muito semelhantes [Friedman et al. (2000*b*)]. Além disso, o custo computacional é também considerável, requerendo n aplicações do método de classificação. Por outro lado, com valores pequenos para K ($K = 5$ por exemplo), CV têm variância pequena mas o viés pode ser um problema, dependendo de como o desempenho do método de classificação varia com o tamanho do conjunto de treinamento. De uma maneira geral, $K = 5$ ou $K = 10$ são recomendados como bons compromissos entre viés e variância.

4.4 Bootstrap

O termo *bootstrap* refere-se a uma classe de procedimentos para estimação de parâmetros em geral e taxas de erro em particular. Foi introduzido por Efron (1979) e desde então tem recebido bastante atenção dos estatísticos [Efron & Tibshirani (1993); Davison & Hinkley (1997)]. Suponha que dispomos de uma amostra de treinamento $T = \{\boldsymbol{x}_j, y_j\}_{j=1}^{n}$. A idéia básica é retirar aleatoriamente amostras com reposição, cada amostra com tamanho igual ao do conjunto de treinamento. Este procedimento é repetido B vezes (B suficientemente grande), produzindo B amostras bootstrap como ilustrado na figura 4.2.

Desejamos avaliar a acurácia estatística de uma quantidade $S(T)$ calculada em nosso conjunto de padrões. Retiramos B amostras bootstrap $T^{*(1)}, \ldots, T^{*(B)}$ cada uma com o mesmo tamanho do conjunto de treinamento original e calculamos a quantidade de interesse $S(T)$ em cada uma das B amostras. Os valores $S(T^{*(1)}), \ldots, S(T^{*(B)})$ são então utilizados para avaliar a acurácia estatística de $S(T)$. A partir desse procedimento nós podemos estimar algum aspecto

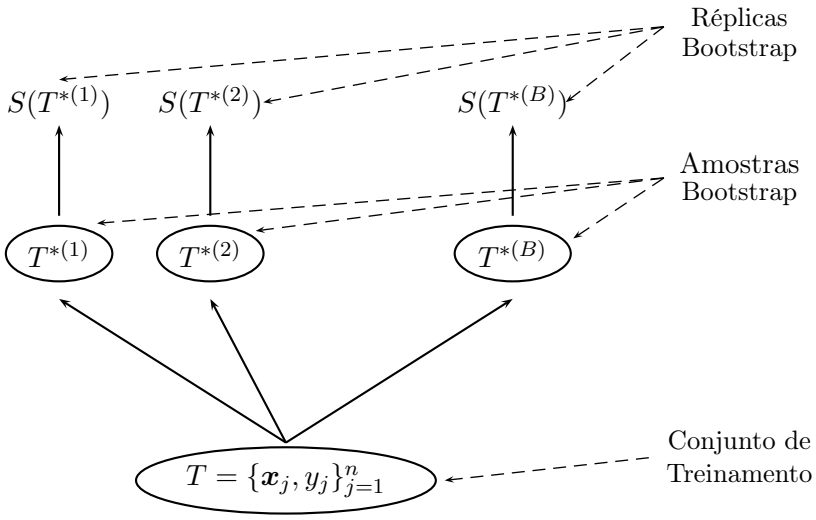

Figura 4.2: Ilustração de um procedimento Bootstrap.

da distribuição de $S(T)$, como, por exemplo, sua variância,

$$\widehat{Var}[S(T)] = \frac{1}{B-1}\sum_{b=1}^{B}(S(T^{*(b)}) - \bar{S}^{*}), \tag{4.4}$$

onde $\bar{S}^{*} = B^{-1}\sum_{b=1}^{B} S(T^{*(b)})$. Podemos interpretar (4.4) como uma estimativa de Monte Carlo da variância de $S(T)$ sob amostragem da função de distribuição empírica $\hat{G}$ para a amostra de treinamento $T = \{\boldsymbol{x}_j, y_j\}_{j=1}^{n}$.

Através do bootstrap podemos também calcular taxas de erro de procedimentos de classificação [Friedman et al. (2000*b*)]. Uma possível abordagem seria construir o classificador em um conjunto de amostras bootstrap e então verificar como ele prediz o conjunto de treinamento original. Se $F^{*(b)}$ é o valor predito em $\boldsymbol{x}_i$, a partir da regra de classificação definida na b-ésima amostra bootstrap, nossa estimativa para a taxa de erro é

$$\widehat{\mathrm{Err}}_{boot} = \frac{1}{B}\frac{1}{n}\sum_{b=1}^{B}\sum_{i=1}^{n} L(y_i, F^{*(b)}(\boldsymbol{x}_i)) \tag{4.5}$$

Contudo, podemos ver que $\widehat{\mathrm{Err}}_{boot}$ não será uma boa estimativa da taxa de erro verdadeira, já que as amostras bootstrap estão atuando como conjuntos de treinamento, enquanto que o

conjunto de treinamento original está atuando como conjunto de teste e estes dois conjuntos terão observações em comum, fazendo com que classificadores super-ajustados forneçam estimativas da taxa de erro irrealisticamente boas [Friedman et al. (2000*b*)]. Esta é a razão pela qual cross-validation explicitamente não utiliza dados coincidentes nos conjuntos de treinamento e teste.

Através de um procedimento que imita o método cross-validation, uma melhor estimativa bootstrap pode ser obtida utilizando as observações que não aparecem nas amostras bootstrap. Ou seja, para cada observação nós apenas consideramos as predições feitas a partir de amostras bootstrap que não contém aquela amostra. A estimativa *bootstrap leave-one-out* da taxa de erro é definida por

$$\widehat{\text{Err}}^{(1)} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{|C^{-i}|}\sum_{b\in C^{-i}} L(y_i, F^{*(b)}(\boldsymbol{x}_i)), \tag{4.6}$$

onde C^{-i} é o conjunto de índices das amostras bootstrap b que não contêm a observação i, e $\left|C^{-i}\right|$ é o número de tais amostras. No cálculo de $\widehat{\text{Err}}^{(1)}$ nós temos que escolher um número de amostras bootstrap suficientemente grande para assegurar que todos os $\left|C^{-i}\right|$ sejam maiores que zero, ou então nós podemos apenas deixar de fora os termos em (4.6) correspondentes aos $\left|C^{-i}\right|$ que são zero. Este estimador soluciona o problema do super-ajustamento sofrido por $\widehat{\text{Err}}_{boot}$, contudo será ainda uma estimativa viesada (para cima) da taxa de erro verdadeira.

O estimador ".632" proposto por Efron (1983) tenta corrigir esse viés. Este estimador é definido como

$$\widehat{\text{Err}}^{(.632)} = 0,368 \times \overline{err} + 0,632 \times \widehat{\text{Err}}^{(1)}. \tag{4.7}$$

Intuitivamente o estimador bootstrap .632 tenta trazer o valor da estimativa bootstrap leave-one-out para próximo do valor do erro aparente, reduzindo assim seu viés. A constante .632 está relacionada com o valor aproximado da probabilidade da i-ésima observação pertencer à b-ésima amostra bootstrap de tamanho n, ou seja,

$$\begin{aligned} P\{i\text{-ésima observação} \in b\text{-ésima amostra bootstrap}\} &= 1 - \left(1 - \frac{1}{n}\right)^n \\ &\approx 1 - e^{-1} \\ &= 0,632. \end{aligned}$$

CAPÍTULO 5

Comparação entre Análise Discriminante Clássica e de Núcleo

5.1 Introdução

Neste capítulo temos o objetivo de comparar a análise discriminante de núcleo com os métodos de classificação clássicos: análise discriminante linear e análise discriminante quadrática. Na seção 5.2 apresentamos um estudo, realizado através de simulações de Monte Carlo, cujo objetivo foi avaliar e comparar, em algumas situações de interesse, o desempenho do método discriminante de núcleo com os desempenhos dos métodos clássicos de discriminação. Neste estudo de simulação, os métodos de classificação foram comparados através dos valores das taxas de erro obtidas pelo procedimento Treinamento-e-Teste, descrito na seção 4.2. Na seção 5.3 apresentamos uma aplicação dos três métodos em estudo a dois conjuntos de dados reais. Para o primeiro conjunto de dados as taxas de erro foram calculadas através do método Treinamento-e-Teste e através do estimador bootstrap .632. Já para o segundo conjunto de dados, as taxas de erro foram obtidas utilizando-se o estimador bootstrap .632.

5.2 Estudos de Simulação

Conduzimos um estudo de simulação com a finalidade de comparar o desempenho, através de taxas erro de classificação, dos seguintes métodos de classificação:

- Análise Discriminante Linear (rotulada Discriminante Linear);

- Análise Discriminante Quadrática (rotulada Discriminante Quadrática);
- Análise Discriminante de Núcleo (rotulada Discriminante de Núcleo).

Estes classificadores foram avaliados em 6 cenários de simulação distintos (A, B, C, D, E e F), baseados em distribuições normais bivariadas e em misturas de distribuições normais bivariadas, considerando o caso de classificação com 2 populações ou grupos. Foram utilizadas 1000 réplicas de Monte Carlo e amostras de treinamento de tamanhos 50, 100, 500 e 1000. Em cada cenário considerado os classificadores construídos em cada amostra de treinamento tiveram seu desempenho avaliado em um conjunto de teste fixo, independente, composto por 1000 observações. Na tabela 5.1 a seguir, estão apresentados os cenários de simulação considerados neste estudo. As curvas de nível das densidades consideradas nestes cenários podem ser visualizadas na figura 5.1.

No cenário A as densidades das populações são claramente distintas e é esperado que os três métodos de classificação considerados neste estudo apresentem desempenho satisfatório nesta situação. No cenário B as densidades das populações são semelhantes, contudo, há uma fronteira razoávelmente bem definida entre as duas populações e, dessa forma, é esperado que os três classificadores forneçam bons resultados também nesta situação. No cenário C há uma região na qual não é possível identificar com precisão a que população determinada observação é mais provável pertencer. Nesse contexto, espera-se que o método discriminante de núcleo apresente melhores resultados que a regra discriminante linear e regra discriminante quadrática. No cenário D temos um par de distribuições normais bimodais. Nos cenários E e F temos uma mistura de densidades normais resultando em um densidade normal bimodal e uma densidade normal bastante dispersa. Para esses três últimos cenários, D, E, e F, espera-se que as regras discriminantes linear e quadrática tenham desempenhos pobres, pois é difícil definir regiões de classificação para esses casos, baseadas em funções lineares ou quadráticas. Espera-se também que, para estes cenários, o método discriminante de núcleo forneça bons resultados.

Antes de apresentarmos os resultados referentes às simulações analisaremos os cenários A e D, apresentados na tabela 5.1, considerando o caso em que temos uma amostra de treinamento com 100 observações e uma amostra de teste também com 100 observações. As taxas de erro obtidas

Tabela 5.1: Distribuições consideradas em cada cenário de simulação.

Cenário	Distribuição
A	$\Pi_1 : f_1 \sim N\left(\begin{bmatrix} 1 \\ -1 \end{bmatrix}; \begin{bmatrix} \frac{4}{9} & \frac{14}{45} \\ \frac{14}{45} & \frac{4}{9} \end{bmatrix}\right); \Pi_2 : f_2 \sim N\left(\begin{bmatrix} -1 \\ 1 \end{bmatrix}; \begin{bmatrix} \frac{4}{9} & 0 \\ 0 & \frac{4}{9} \end{bmatrix}\right)$
B	$\Pi_1 : f_1 \sim N\left(\begin{bmatrix} -1 \\ 1 \end{bmatrix}; \begin{bmatrix} \frac{2}{3} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{3} \end{bmatrix}\right); \Pi_2 : f_2 \sim N\left(\begin{bmatrix} 1 \\ -\frac{1}{2} \end{bmatrix}; \begin{bmatrix} \frac{2}{3} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{3} \end{bmatrix}\right)$
C	$\Pi_1 : f_1 \sim N\left(\begin{bmatrix} -1 \\ \frac{1}{2} \end{bmatrix}; \begin{bmatrix} \frac{2}{3} & \frac{1}{5} \\ \frac{1}{5} & \frac{4}{9} \end{bmatrix}\right); \Pi_2 : f_2 \sim N\left(\begin{bmatrix} 1 \\ -\frac{1}{2} \end{bmatrix}; \begin{bmatrix} \frac{2}{3} & \frac{1}{5} \\ \frac{1}{5} & \frac{4}{9} \end{bmatrix}\right)$
D	$\Pi_1 : f_1 \sim \frac{1}{2}N\left(\begin{bmatrix} -\frac{3}{2} \\ -\frac{3}{2} \end{bmatrix}; \begin{bmatrix} \frac{4}{5} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{4}{5} \end{bmatrix}\right) + \frac{1}{2}N\left(\begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \end{bmatrix}; \begin{bmatrix} \frac{4}{5} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{4}{5} \end{bmatrix}\right);$ $\Pi_2 : f_2 \sim \frac{1}{2}N\left(\begin{bmatrix} \frac{3}{2} \\ \frac{3}{2} \end{bmatrix}; \begin{bmatrix} \frac{4}{5} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{4}{5} \end{bmatrix}\right) + \frac{1}{2}N\left(\begin{bmatrix} -\frac{1}{2} \\ -\frac{1}{2} \end{bmatrix}; \begin{bmatrix} \frac{4}{5} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{4}{5} \end{bmatrix}\right)$
E	$\Pi_1 : f_1 \sim \frac{1}{2}N\left(\begin{bmatrix} -\frac{3}{2} \\ 0 \end{bmatrix}; \begin{bmatrix} \frac{1}{12} & \frac{1}{4} \\ \frac{1}{4} & 1 \end{bmatrix}\right) + \frac{1}{2}N\left(\begin{bmatrix} \frac{3}{2} \\ 0 \end{bmatrix}; \begin{bmatrix} \frac{1}{12} & \frac{1}{4} \\ \frac{1}{4} & 1 \end{bmatrix}\right);$ $\Pi_2 : f_2 \sim N\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}; \begin{bmatrix} \frac{4}{9} & \frac{1}{5} \\ \frac{1}{5} & \frac{4}{9} \end{bmatrix}\right)$
F	$\Pi_1 : f_1 \sim \frac{1}{2}N\left(\begin{bmatrix} -\frac{3}{2} \\ 0 \end{bmatrix}; \begin{bmatrix} \frac{2}{10} & \frac{1}{4} \\ \frac{1}{4} & \frac{3}{10} \end{bmatrix}\right) + \frac{1}{2}N\left(\begin{bmatrix} \frac{3}{2} \\ 0 \end{bmatrix}; \begin{bmatrix} \frac{3}{10} & \frac{1}{4} \\ \frac{1}{4} & \frac{3}{10} \end{bmatrix}\right);$ $\Pi_2 : f_2 \sim N\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}; \begin{bmatrix} \frac{4}{5} & \frac{2}{5} \\ \frac{2}{5} & 1 \end{bmatrix}\right)$

pelas funções discriminantes linear, quadrática e de núcleo foram, respectivamente, 2%, 0% e 1%, para o cenário A. No cenário D, as taxas de erro foram de 31%, 35% e 19%, para as funções discriminantes linear, quadrática e de núcleo, respectivamente. Observando as figuras 5.2, 5.3 e 5.4 vemos que, em situações como a considerada no cenário A, as regiões de classificação obtidas através da estimação de densidades pelo método de núcleo são semelhantes àquelas obtidas através da estimação paramétrica das densidades. Dessa forma, espera-se que os três métodos forneçam resultados aproximadamente iguais. Em contrapartida, as figuras 5.5, 5.6 e 5.7 evidenciam que, em situações como a considerada no cenário D, as regiões de classificação obtidas através da estimação de densidades pelo método de núcleo são muito diferentes daquelas

obtidas através da estimação paramétrica das densidades. Podemos notar que, em situações como a apresentada no cenário D, as regiões de classificação definidas pela regra discriminante de núcleo são capazes de detectar bimodalidade nos dados, enquanto que suas contrapartes paramétricas não são.

5.2.1 Resultados das Simulações

Nesta subseção apresentamos os resultados das simulações realizadas. Na estimação das densidades dos grupos, para a obtenção das matrizes de largura de banda, $\mathbf{H}$, consideramos matrizes pertencentes à classe de todas as matrizes simétricas, positivas-definidas, ou seja,

$$\mathbf{H} = \begin{bmatrix} h_1^2 & h_{12} \\ h_{12} & h_2^2 \end{bmatrix}.$$

Em todos os cenários estas matrizes foram obtidas através do método de mínimos quadrados por validação cruzada [Duong (2004)]. A função núcleo utilizada foi a função densidade de probabilidade de uma distribuição normal bivariada.

Nas tabelas 5.2, 5.3, 5.4, 5.5, 5.6, 5.7 estão apresentadas as médias e os desvios-padrões das taxas de erro de classificação para as funções discriminantes linear, quadrática e de núcleo, para os cenários A, B, C, D, E e F, respectivamente. Em todos os cenários, vemos que as médias e os desvios-padrões das taxas de erro diminuem quando aumentamos o tamanho da amostra de treinamento.

A partir da tabela 5.2, podemos ver que os três métodos apresentam resultados semelhantes em situações como a apresentada no cenário A, com o método de núcleo apresentando desempenho um pouco pior com amostras pequenas. Podemos também notar que análise discriminante quadrática e de núcleo apresentam melhores desempenhos que a análise discriminante linear, sendo a função discriminante quadrática a que apresentou melhores resultados para todos os tamanhos de amostra considerados.

Em situações como a apresentada no cenário B, os três métodos apresentam comportamentos semelhantes aos apresentados no cenário A. Mais uma vez, o método discriminante de núcleo foi o que forneceu os piores resultados com amostras de tamanhos reduzidos, contudo, os resultados com amostras de tamanhos maiores foram muito próximos para os três métodos.

Podemos observar a partir da tabela 5.4, que em situações como a apresentada no cenário C, onde há uma região onde não é possível distinguir com precisão entre as populações, o método discriminante de núcleo forneceu os melhores desempenhos para todos os tamanhos de amostra considerados. Nesta situação, o método discriminante linear e o método discriminante de núcleo apresentaram resultados bastante próximos, tendo este último fornecido taxas de erro ligeiramente menores.

Tabela 5.2: Média e Desvio-padrão (D. P.) das taxas de erro (%) de classificação sobre as 1000 réplicas de Monte Carlo para as funções discriminantes Linear, Quadrática e de Núcleo considerando o Cenário A.

		$n = 50$	$n = 100$	$n = 500$	$n = 1000$
Discriminante Linear	Média	0,6233	0,5664	0,5033	0,5039
	D. P.	0,3017	0,2440	0,1081	0,0717
Discriminante Quadrática	Média	0,3746	0,3381	0,3117	0,3025
	D. P.	0,1790	0,0926	0,0451	0,0206
Discriminante de Núcleo	Média	1,8073	0,6713	0,3413	0,3244
	D. P.	3,6037	1,0796	0,0962	0,0655

Tabela 5.3: Média e Desvio-padrão (D. P.) das taxas de erro (%) de classificação sobre as 1000 réplicas de Monte Carlo para as funções discriminantes Linear, Quadrática e de Núcleo considerando o Cenário B.

		$n = 50$	$n = 100$	$n = 500$	$n = 1000$
Discriminante Linear	Média	1,0867	0,9642	0,9050	0,9088
	D. P.	0,2814	0,1514	0,0736	0,0706
Discriminante Quadrática	Média	1,3149	1,0702	0,9087	0,9052
	D. P.	0,4757	0,2514	0,0746	0,0687
Discriminante de Núcleo	Média	1,5593	1,2551	0,9925	0,9410
	D. P.	0,6088	0,3606	0,1576	0,1185

Na tabela 5.5 estão apresentadas as médias taxas de erro e os desvios-padrões sobre as 1000 réplicas de Monte Carlo, obtidos pelos métodos de classificação em estudo, em situações como a apresentada pelo cenário D. Neste caso, a superioridade do método de núcleo é evidente, pois este método apresentou taxas de erro muito menores que suas contrapartes linear e quadrática, para todos os tamanhos de amostra. Os valores das taxas de erro obtidas pelas regras discriminantes

linear e quadrática foram bastante próximos, tendo a regra quadrática apresentado resultados um pouco melhores.

As tabelas 5.6 e 5.7 exibem as médias das taxas de erro e os desvios-padrões, sobre as 1000 réplicas de Monte Carlo, obtidos pelos métodos de classificação em estudo, em situações como as apresentadas pelos cenários E e F, respectivamente. Novamente, o método discriminante de núcleo apresentou desempenho superior aos desempenhos dos métodos discriminantes linear e quadrática. Podemos observar que os valores das médias das taxas de erro sobre as 1000 réplicas de Monte Carlo foram menores para o método de núcleo, em todos os tamanhos de amostra considerados. Podemos ainda destacar que, tanto no cenário E quanto no cenário F, a regra discriminante quadrática apresentou resultados muito melhores que a regra discriminante linear, sendo esta última a que apresentou os piores resultados.

Tabela 5.4: Média e Desvio-padrão (D. P.) das taxas de erro (%) de classificação sobre as 1000 réplicas de Monte Carlo para as funções discriminantes Linear, Quadrática e de Núcleo considerando o Cenário C.

		$n = 50$	$n = 100$	$n = 500$	$n = 1000$
Discriminante Linear	Média	4,8170	4,6447	4,5223	4,5146
	D. P.	0,4767	0,3727	0,2348	0,1913
Discriminante Quadrática	Média	4,9875	4,7300	4,5377	4,5195
	D. P.	0,6089	0,4236	0,2559	0,2101
Discriminante de Núcleo	Média	2,5351	2,2039	1,8838	1,8341
	D. P.	1,0339	0,7230	0,3798	0,2880

Tabela 5.5: Média e Desvio-padrão (D. P.) das taxas de erro (%) de classificação sobre as 1000 réplicas de Monte Carlo para as funções discriminantes Linear, Quadrática e de Núcleo considerando o Cenário D.

		$n = 50$	$n = 100$	$n = 500$	$n = 1000$
Discriminante Linear	Média	43,3035	44,5808	46,3690	46,6073
	D. P.	2,6130	2,0645	0,5990	0,3108
Discriminante Quadrática	Média	43,1092	44,0681	46,0483	46,4462
	D. P.	2,6529	1,9639	0,7543	0,4151
Discriminante de Núcleo	Média	24,7502	19,9339	16,3670	16,1180
	D. P.	6,4371	3,3978	0,6678	0,5049

Tabela 5.6: Média e Desvio-padrão (D. P.) das taxas de erro (%) de classificação sobre as 1000 réplicas de Monte Carlo para as funções discriminantes Linear, Quadrática e de Núcleo considerando o Cenário E.

		$n = 50$	$n = 100$	$n = 500$	$n = 1000$
Discriminante Linear	Média	47,8123	48,3086	49,2980	49,5784
	D. P.	3,2952	2,4899	1,4873	1,2574
Discriminante Quadrática	Média	12,3440	10,0298	7,9060	7,7968
	D. P.	4,0999	2,4674	0,6965	0,4928
Discriminante de Núcleo	Média	10,4084	7,5744	5,9294	5,9421
	D. P.	3,8523	2,0576	0,5980	0,4642

Tabela 5.7: Média e Desvio-padrão (D. P.) das taxas de erro (%) de classificação sobre as 1000 réplicas de Monte Carlo para as funções discriminantes Linear, Quadrática e de Núcleo considerando o Cenário F.

		$n = 50$	$n = 100$	$n = 500$	$n = 1000$
Discriminante Linear	Média	48,8693	49,1778	49,6406	49,6833
	D. P.	2,4543	1,9858	1,1794	1,0829
Discriminante Quadrática	Média	23,3331	21,6190	20,0978	19,8070
	D. P.	4,6566	3,7946	2,1063	1,6127
Discriminante de Núcleo	Média	17,8189	15,4789	13,5814	13,1674
	D. P.	2,8313	1,5302	0,5391	0,3796

5.3 Aplicações

Nesta seção, temos o objetivo de comparar os desempenhos dos métodos clássicos considerados neste estudo com o desempenho do método discriminante de núcleo, em aplicações práticas.

O primeiro conjunto de dados considerado é o exemplo do salmão introduzido na seção 3.3 e contém medidas do diâmetro dos anéis de 100 salmões no primeiro ano de crescimento em água doce (X_1) e em água do mar (X_2), capturados em águas do estado americano do Alasca ou em águas do Canadá. Dessa forma, estamos lidando com um problema de classificação com duas classes, Alasca e Canadá, cada população composta por 50 observações, e duas variáveis explicativas. O desempenho dos classificadores, aplicados a este conjunto de dados, será avaliado de duas formas: através do procedimento Treinamento-e-Teste e através do estimador bootstrap .632.

O segundo conjunto de dados contém 306 observações que foram utilizadas um estudo conduzido entre 1958 e 1970 no Hospital Billings da Universidade de Chicago, sobre a sobrevivência de pacientes que se submeteram a uma cirurgia de câncer de mama. As variáveis explicativas disponíveis para este conjunto de dados eram: idade do paciente no momento da operação (X_1), o ano da operação menos 1900 (X_2) e o número de nódulos positivos detectados (X_3). As populações estão definidas por: Π_1, o paciente sobreviveu cinco anos ou mais após a cirurgia; e Π_2, o paciente morreu dentro de cinco anos depois da cirurgia. Para este conjunto de dados, os classificadores tiveram seu desempenho avaliado através do estimador bootstrap .632. Esse conjunto de dados pode ser obtido através do site `ftp://ftp.ics.uci.edu/pub/machine-learning-databases/haberman`.

5.3.1 Resultados

O conjunto de dados do salmão foi particionado, aleatoriamente, em dois subconjuntos com mesmo número de observações, ou seja, um conjunto de treinamento composto por 50 observações e um conjunto de teste também composto por 50 observações. Dessa forma, as regras discriminantes linear, quadrática e de núcleo foram obtidas a partir do conjunto de treinamento e foram utilizadas para alocar as observações do conjunto de teste a uma das duas populações. As taxas de erro obtidas pelos métodos de classificação são apresentadas na tabela 5.8. Podemos verificar que o método discriminante de núcleo foi o que apresentou menor taxa de erro.

Tabela 5.8: Taxas de Erro obtidas pelas regras discriminantes linear quadrática e de núcleo para o conjunto de teste dos dados do salmão.

Função discriminante	Taxa de Erro (%)
Linear	10
Quadrática	10
Núcleo	8

Nas figuras 5.8, 5.9 e 5.10 são apresentadas as regiões de classificação obtidas respectivamente pelas regras linear quadrática e de núcleo. Apesar de essas regiões serem bastante semelhantes, a região definida pelo método de núcleo consegue alocar com mais precisão as observações do

conjunto de teste.

Na tabela 5.9 são apresentadas as taxas de erro das funções discriminantes linear, quadrática e de núcleo, calculadas utilizando-se o estimador bootstrap .632, para os dados do salmão e dos pacientes que se submeteram a uma cirurgia para retirada de um tumor na mama. Nesta tabela, vemos que, para os dados do salmão, a função discriminante quadrática é a que apresenta a menor taxa de erro, contudo, as taxas de erro para os três métodos se apresentam bastante próximas. Para os dados dos pacientes, a regra discriminante de núcleo foi a que apresentou menor taxa de erro.

Tabela 5.9: Taxas de Erro obtidas pelas regras discriminantes linear quadrática e de núcleo calculadas utilizando o estimador bootstrap .632, para os conjuntos de dados do salmão e dos pacientes que passaram por uma cirurgia para retirada de um tumor na mama.

Função discriminante	Taxa de Erro (%)	
	Salmão	Pacientes
Linear	7,44	25,36
Quadrática	7,15	24,74
Núcleo	7,71	23,25

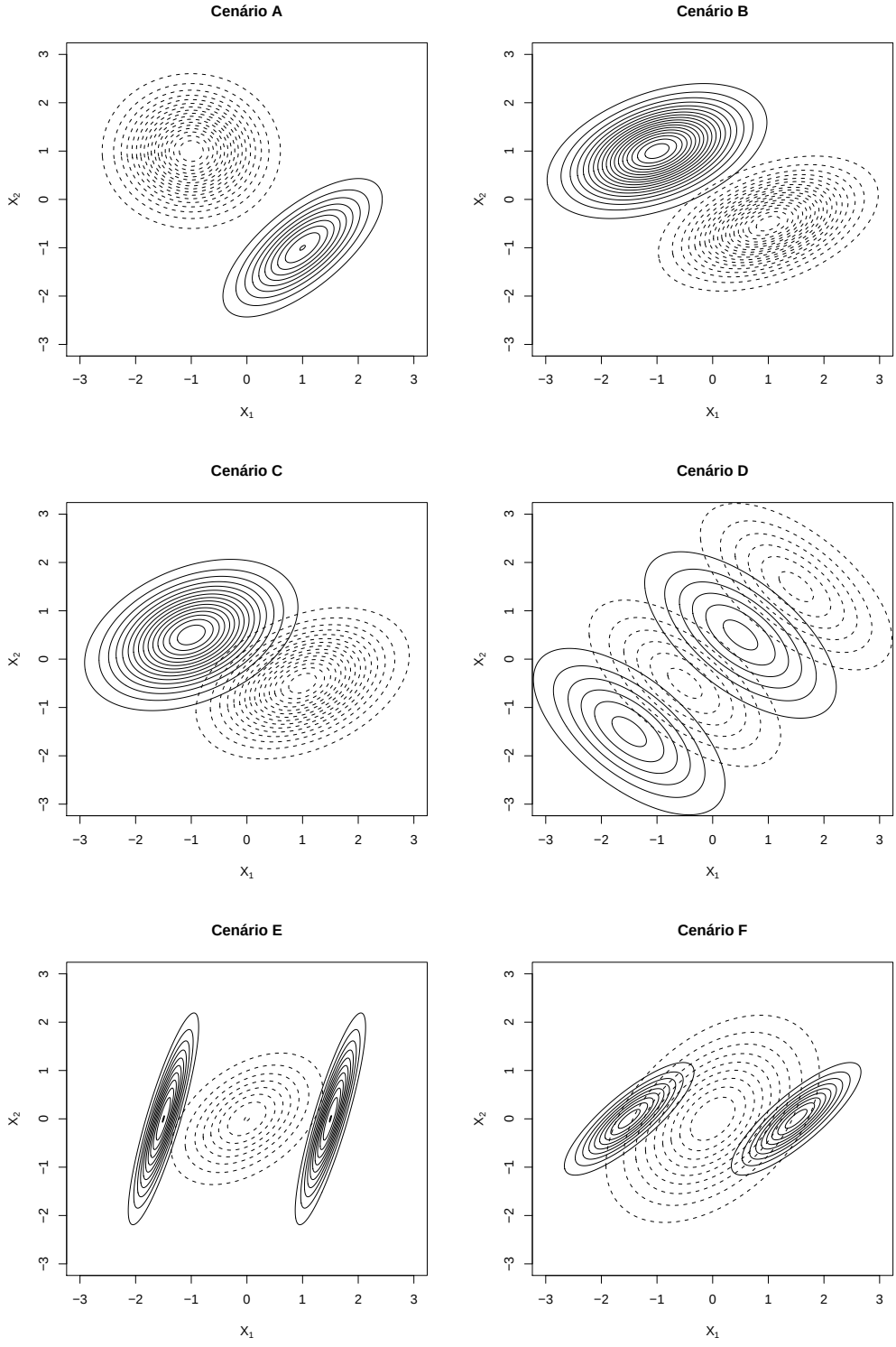

Figura 5.1: Curvas de nível das densidades consideradas no estudo de simulação: linhas sólidas – Π_1, linhas tracejadas – Π_2.

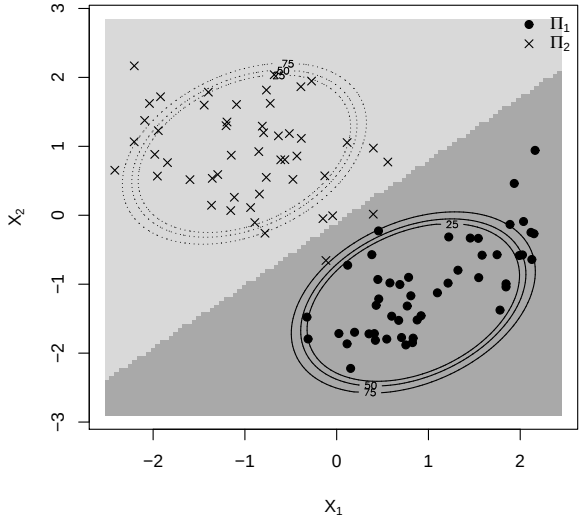

Figura 5.2: Regiões de classificação e curvas de nível das estimativas das densidades das populações para o cenário A, através da regra discriminante linear; linhas sólidas – Π_1, linhas tracejadas – Π_2.

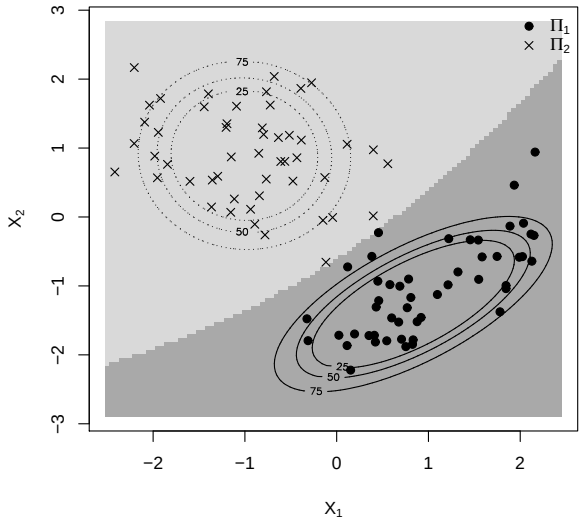

Figura 5.3: Regiões de classificação e curvas de nível das estimativas das densidades das populações para o cenário A, através da regra discriminante quadrática; linhas sólidas – Π_1, linhas tracejadas – Π_2.

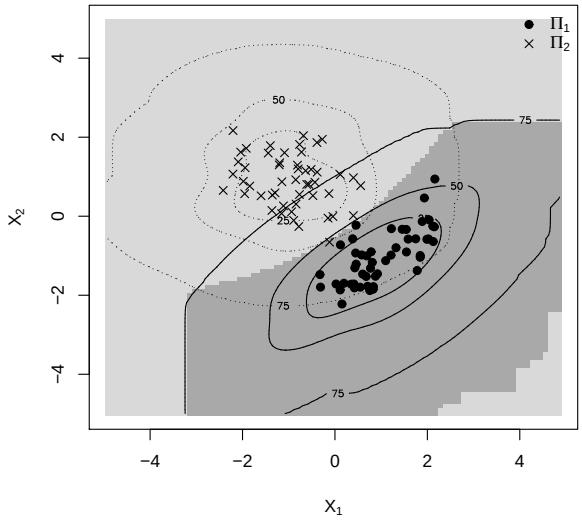

Figura 5.4: Regiões de classificação e curvas de nível das estimativas das densidades das populações para o cenário A, através da regra discriminante de núcleo; linhas sólidas – Π_1, linhas tracejadas – Π_2.

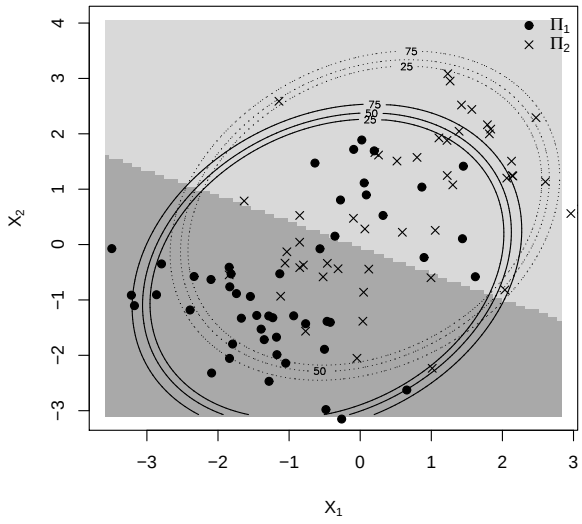

Figura 5.5: Regiões de classificação e curvas de nível das estimativas das densidades das populações para o cenário D, através da regra discriminante linear; linhas sólidas – Π_1, linhas tracejadas – Π_2.

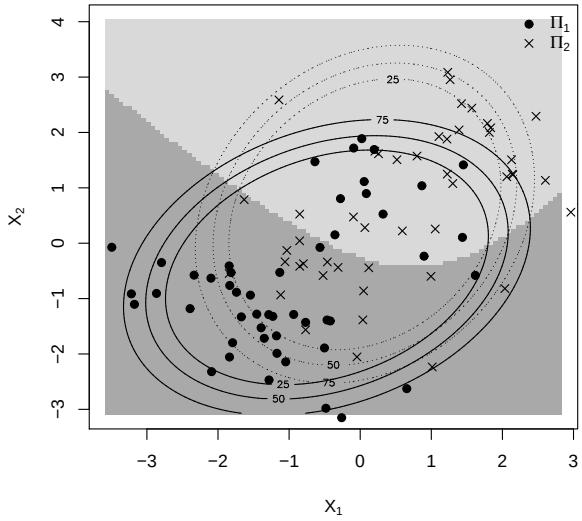

Figura 5.6: Regiões de classificação e curvas de nível das estimativas das densidades das populações para o cenário D, através da regra discriminante quadrática; linhas sólidas – Π_1, linhas tracejadas – Π_2.

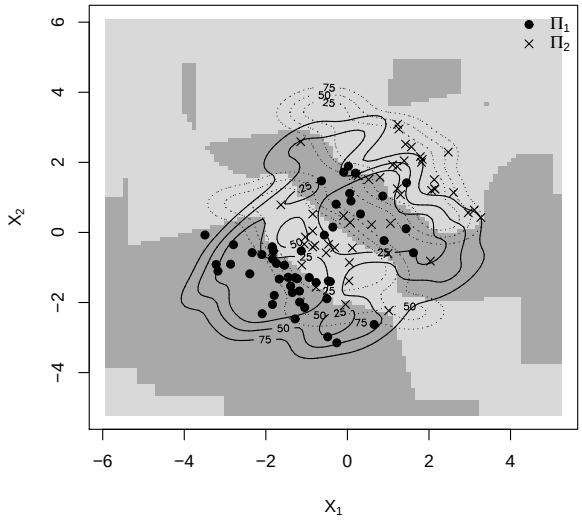

Figura 5.7: Regiões de classificação e curvas de nível das estimativas das densidades das populações para o cenário D, através da regra discriminante de núcleo; linhas sólidas – Π_1, linhas tracejadas – Π_2.

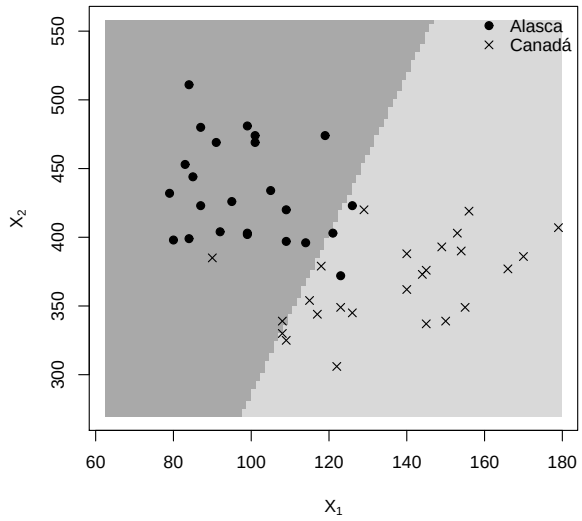

Figura 5.8: Observações do conjunto teste alocadas de acordo com as regiões de classificação definidas pela regra discriminante linear para o conjunto de treinamento dos dados do salmão.

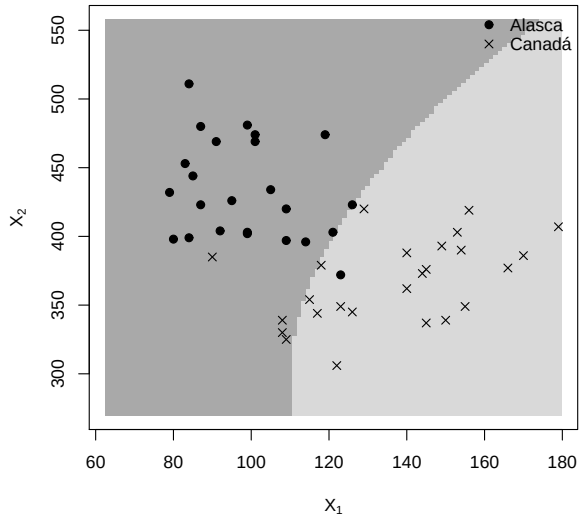

Figura 5.9: Observações do conjunto teste alocadas de acordo com as regiões de classificação definidas pela regra discriminante quadrática para o conjunto de treinamento dos dados do salmão.

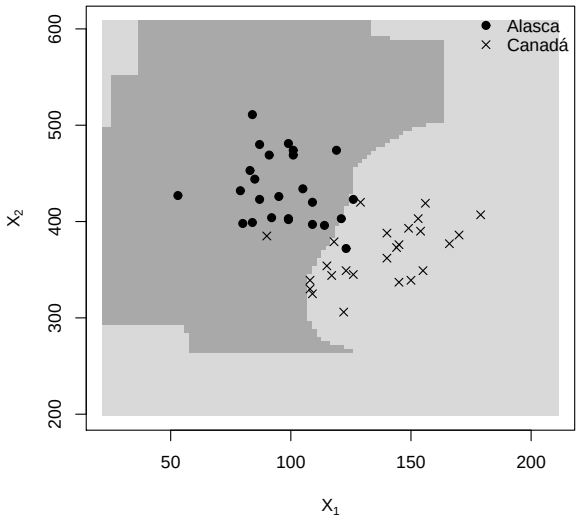

Figura 5.10: Observações do conjunto teste alocadas de acordo com as regiões de classificação definidas pela regra discriminante de núcleo para o conjunto de treinamento dos dados do salmão.

CAPÍTULO 6

Algumas Contribuições

6.1 Introdução

Neste capítulo serão apresentados alguns métodos relevantes na resolução de problemas de classificação. Estes métodos representam a aplicação de métodos já existentes na literatura em um novo contexto. Na seção 6.2 é apresentado um método que modifica a forma como os pesos das observações são incorporados no algoritmo boosting, no caso particular da análise discriminante linear de Fisher. Na seção 6.3 é introduzido o conceito de fronteira de classificação e um método para obtê-la, baseado no método bootstrap. Este conceito pode ser útil para o usuário final na resolução de problemas empresariais e em outros casos. Na seção 6.4 é introduzido um método de bootstrap para calcular um intervalo de confiança para a taxa de erro de um método de classificação qualquer, o que representa o início de um caminho para uma comparação mais formal de métodos de classificação.

6.2 Uma Nova Formulação do Algorítmo Boosting em Análise Discriminante Linear de Fisher

Nesta seção, iremos apresentar uma nova formulação do algoritmo boosting em análise discriminante linear de Fisher. Boosting faz parte de uma classe de procedimentos cujo objetivo é melhorar o desempenho de classificadores [Schapire (1990)]. Trata-se de uma técnica rela-

tivamente recente, mas que tem recebido bastante atenção de pesquisadores de várias áreas, principalmente das áreas de computação e estatística. Na subseção seguinte apresentamos o método boosting de uma maneira geral. Em seguida, apresentamos uma formulação do algoritmo boosting em análise discriminante linear de Fisher, que difere conceitualmente da formulação usual. Ao final desta seção, apresentamos um estudo de simulação com o objetivo de ilustrar o método usual e o método proposto.

6.2.1 Boosting

O ponto de partida desta seção é um procedimento denominado *boosting*, que é um método geral para melhorar o desempenho de classificadores. Este procedimento é uma das mais importantes e bem sucedidas metodologias desenvolvidas nos últimos anos na literatura de classificação de padrões e tem recebido bastante atenção dos estatísticos desde seu surgimento na literatura de aprendizado automático [Schapire (1990); Freund (1995); Freund & Schapire (1996)].

O procedimento boosting consiste em aplicar, sequencialmente, uma regra de classificação qualquer (chamada de classificador base) a versões iterativamente reponderadas da amostra de treinamento. Em cada iteração o algoritmo atribui pesos maiores às observações incorretamente classificadas na iteração anterior. A regra final de classificação é obtida através de uma combinação linear dos classificadores construídos em cada iteração, produzindo um comitê de classificação. Para muitos métodos de classificação, esta estratégia simples resulta em uma impressionante melhora de desempenho.

Uma das versões do procedimento boosting mais utilizadas e difundidas na literatura e na prática é o algoritmo *Adaboost* (**Ada**ptive **Boost**ing) [Freund & Schapire (1996)]. Friedman et al. (2000*a*) descrevem esse algoritmo de maneira semelhante a Freund & Schapire (1996), em problemas de classificação com dois grupos e o denominam Adaboost Discreto. Além disso, mostraram que o algoritmo Adaboost ajusta aproximadamente um modelo logístico aditivo, no qual o número de iterações boosting é o número de funções utilizadas na representação aditiva, ou seja, o número de funções somadas para aproximar a função estimada, o que torna o assunto atraente do ponto de vista estatístico. O algoritmo Adaboost Discreto pode ser descrito pelos

seguintes passos:

Dados: amostra de treinamento $T = \{\boldsymbol{x}_i, y_i\}_{i=1}^n$, onde $\boldsymbol{x}_i \in \mathcal{X}$ e $y_i \in \{-1, +1\}$; e o número de iterações boosting M.

1. Inicialize com pesos $w_i = 1/n$, $i = 1, \ldots, n$.

2. Repita para $m = 1, \ldots, M$:

   (a) Ajuste o classificador $F_m(\boldsymbol{x}) \in \{-1, +1\}$ utilizando os pesos w_i no conjunto de treinamento.

   (b) Calcule:

   $$\text{Err}_m = E_w \left[I_{[y_i \neq F_m(\boldsymbol{x})]} \right],$$

   $$c_m = \log[(1 - \text{Err}_m)/\text{Err}_m],$$

   onde

   $$I_{[y_i \neq F_m(\boldsymbol{x})]} = \begin{cases} 1 & \text{se } y_i \neq F_m(\boldsymbol{x}) \\ 0 & \text{caso contrário.} \end{cases}$$

   (c) Faça $w_i = w_i \exp\{c_m I_{[y_i \neq f_m(\boldsymbol{x})]}\}$, $i = 1, \ldots, n$, e renormalize para que a soma dos pesos seja igual a um, $\sum_i w_i = 1$.

3. Combine os classificadores obtidos em cada iteração através de

   $$\text{sign}\left[\sum_{m=1}^{M} c_m F_m(\boldsymbol{x})\right] = \text{sign}[G(x)].$$

Uma breve descrição deste algoritmo para o problema de classificação com dois grupos é a seguinte: temos dados de treinamento $T = \{\boldsymbol{x}_i, y_i\}_{i=1}^n$, onde, $\boldsymbol{x}_i$ é um vetor de características e $y_i = -1$ ou $+1$ representa o rótulo dos grupos. Definimos $G(\boldsymbol{x}) = \sum_{m=1}^M c_m F_m(\boldsymbol{x})$, onde $F_m(\boldsymbol{x})$ é um classificador que produz valores $+1$ ou -1 e c_m são constantes. A predição correspondente é o classificador $\text{sign}[G(\boldsymbol{x})]$. O algoritmo Adaboost constrói os classificadores $F_m(\boldsymbol{x})$ em versões ponderadas da amostra de treinamento, atribuindo maior peso às observações que foram incorretamente classificadas. Temos desse modo uma seqüência de amostras de treinamento ponderadas e o classificador final é produzido como uma combinação linear dos classificadores construídos em cada uma das amostras de treinamento ponderadas.

6.2.2 Nova Formulação do Algorítmo Boosting

Boosting vem sendo bastante explorado com árvores de classificação e tem obtido incontestável sucesso [Freund & Schapire (1996); Schapire et al. (1998); Friedman et al. (2000*a*); Bühlmann & Yu (2003)], entretando, boosting também pode ser útil com outros tipos de classificadores como, por exemplo, funções discriminantes lineares. Skurichina & Duin (2000) utilizam o procedimento boosting com funções discriminantes lineares e mostram que esse método pode ser bastante útil em algumas situações. Em seu estudo, Skurichina & Duin (2000) utilizam a forma de ponderação que é comumente utilizada na metodologia boosting, ou seja, as observações são multiplicadas pelos seus respectivos pesos.

Neste trabalho, trataremos da atribuição dos pesos às obsevações de uma forma diferente, incorporando os pesos das observações no cálculo dos vetores de médias e das matrizes de covariância dos grupos. A idéia básica é dar pesos maiores às observações cujas contribuições na variância sejam maiores, já que estas observações são as que se apresentam mais distantes do padrão médio das observações.

Considere a situação em que desejamos discriminar entre dois grupos ou populações Π_1 e Π_2. Os vetores de médias e as matrizes de covariância dos grupos podem ser expressos, respectivamente por

$$\bar{\boldsymbol{x}}_i = \frac{1}{n_i}\sum_{j=1}^{n_i} \boldsymbol{x}_{ij}, \tag{6.1}$$

e

$$\mathbf{S}_i = \frac{1}{n_i - 1}\sum_{j=1}^{n_i} (\boldsymbol{x}_{ij} - \bar{\boldsymbol{x}}_i)(\boldsymbol{x}_{ij} - \bar{\boldsymbol{x}}_i)' \tag{6.2}$$

onde $i = 1, 2$ e n_i é o número de observações da i-ésima população. Dessa forma, a matriz de covariância combinada dos grupos pode ser obtida como

$$\mathbf{S} = \frac{(n_1 - 1)\mathbf{S}_1 + (n_2 - 1)\mathbf{S}_2}{n_1 + n_2 - 2}. \tag{6.3}$$

Uma outra maneira de atribuir peso a uma observação é ponderar sua contribuição no vetor de médias e na matriz de covariância. Ou seja, podemos obter os vetores de médias ponderados

e as matrizes de covariância ponderadas dos grupos através das seguintes expressões

$$\bar{\boldsymbol{x}}_i^w = \frac{\sum_{j=1}^n w_j \boldsymbol{x}_j I_{[\boldsymbol{x}_j \in \Pi_i]}}{\sum_{j=1}^n w_j I_{[\boldsymbol{x}_j \in \Pi_i]}}, \tag{6.4}$$

e

$$\mathbf{S}_i^w = \frac{1}{n_i - 1} \sum_{j=1}^{n} (\boldsymbol{x}_{ij} - \bar{\boldsymbol{x}}_i)(\boldsymbol{x}_{ij} - \bar{\boldsymbol{x}}_i)' w_j I_{[\boldsymbol{x}_j \in \Pi_i]} \tag{6.5}$$

onde $i = 1, 2$ e n_i é o número de observações da i-ésima população. Dessa forma, a matriz de covariância combinada dos grupos pode ser da mesma forma que em (6.3), apenas substituindo os valores de S_1 e S_2 por suas versões ponderadas S_1^w e S_2^w. Esse enfoque tenta dar uma outra interpretação ao papel dos pesos no algoritmo boosting, relacionada à contribuição das observações classificadas incorretamente nas parcelas das matrizes de covariância dos grupos, ou seja, o principal objetivo do enfoque proposto é uma mudança conceitual na forma de atribuição dos pesos.

Podemos observar na literatura estatística, a preocupação com métodos que sejam mais apropriados do ponto de vista conceitual. Por exemplo, Wedderburn (1974), quando introduziu a classe de modelos de Quase-Verossimilhança, tentou estabelecer um procedimento que do ponto de vista conceitual era mais adequado em algumas aplicações dos modelos lineares generalizados.

Em relação ao caso do método boosting em análise discriminante linear de Fisher, a ponderação na matriz de covariância é mais adequada do ponto de vista conceitual. Este tipo de ponderação é semelhante àquele utilizado na análise de regressão ponderada (ver, por exemplo, Drapper & Smith (1998, pp. 108–116)).

6.2.3 Estudo de Simulação

Conduzimos um estudo de simulação com o propósito de avaliar o desempenho do método boosting quando utilizamos como classificador base a função discriminante linear de Fisher, avaliando o efeito de atribuirmos pesos diretamente às observações e atribuirmos pesos às parcelas das matrizes de covariâncias dos grupos.

Consideramos novamente o problema de classificação com duas populações. Os dados foram simulados considerando a seguinte estrutura: as observações pertencentes à população 1 foram

geradas a partir de uma distribuição normal 20-dimensional, com vetor de médias $(0, \ldots, 0)$ e matriz de covariâncias $4 \times I_{20}$, em que I_{20} é a matriz identidade de ordem 20; as observações pertencentes à população 2 foram geradas a partir de uma distribuição normal 20-dimensional, com vetor de médias $\left(\frac{1}{\sqrt{20}}, \ldots, \frac{1}{\sqrt{20}}\right)$ e matriz de covariâncias I_{20}. Essa estrutura de simulação foi considerada em Breiman (1998). No estudo consideramos amostras de tamanhos 100, 150, 200, 250, 300, 350, 400, 450 e 500 e para cada tamanho de amostra foram consideradas 1000 réplicas de Monte Carlo e 100 iterações boosting. Para cada tamanho de amostra, em cada réplica de Monte Carlo, os classificadores construídos foram testados em uma amostra de teste fixa, independente, composta por 1000 observações. Os resultados deste estudo de simulação estão apresentados na tabela 6.1 a seguir.

Tabela 6.1: Taxas de Erro para os classificadores $M1$ - Função discriminante linear de Fisher, $M2$ - Boosting tendo como classificador base a função discriminante linear de Fisher, com pesos atribuídos diretamente às observações, $M3$ - Boosting tendo como classificador base a função discriminante linear de Fisher, com pesos incorporados no cálculo dos vetores de médias e matrizes de covariâncias dos grupos.

n	Taxas de Erro (%)		
	$M1$	$M2$	$M3$
100	39,55	40,06	39,75
150	39,01	38,61	38,79
200	38,09	37,16	37,66
250	37,73	36,50	37,15
300	37,42	35,95	36,71
350	37,02	35,38	36,19
400	36,82	35,06	35,88
450	36,56	34,87	35,64
500	36,50	34,71	35,57

Podemos observar na tabela 6.1 que as taxas de erro para os três métodos considerados decresce muito lentamente com o aumento do número de observações no conjunto de treinamento. Contudo, podemos também notar que essas taxas são ligeiramente menores para os casos em que o método boosting foi utilizado. Comparando agora os desempenhos dos métodos boosting com a forma de atribuição de pesos usual e a forma de atribuição de pesos proposta, vemos que embora a forma usual forneça taxas de erro menores, a forma proposta apresenta taxas de erro bastante

próximas, o que fornece evidências que a forma como os pesos são incorporados no método boosting pode ainda ser explorada com o propósito de melhorar o desempenho do método.

6.3 Construção de uma Fronteira de Classificação entre duas populações através do método Bootstrap

Em muitas situações práticas pode ser interessante indentificar as observações situadas na fronteira entre as regiões de classificação definidas pelo método de classificação e deixar que o usuário do método decida, através de alguma forma (possivelmente subjetiva), a qual grupo tais observações pertencem.

Por exemplo, imagine a situação de um gerente de um banco que precisa decidir se deve fornecer empréstimos a seus clientes. Neste caso os clientes do banco podem ser divididos, através de algum critério adotado pelo gerente, em duas populações: possíveis adimplentes (Π_1) e possíveis inadimplentes (Π_2). Dessa forma, alguma regra de classificação razoável deverá alocar os clientes, com base em informações fornecidas tais como renda, valor do empréstimo, número de prestações, etc., a uma das duas populações, Π_1 ou Π_2. Contudo, certamente alguns clientes pertencerão a uma região em que é difícil definir com precisão a que população eles pertencem, e possivelmente serão incorretamente classificados.

Esta seção tem o objetivo de introduzir um método capaz de fornecer ao usuário um conjunto de observações que se situam em uma região de fronteira entre as regiões definidas pela regra de classificação. No caso do gerente do banco, o método irá fornecer um conjunto de clientes que estão na fronteira entre as populações de possíveis adimplentes e possíveis inadimplentes. De posse desse conjunto, o gerente poderia análisar cuidadosamente esses casos e dessa forma tomar uma melhor decisão baseada em seu conhecimento.

Essa fronteira será construída utilizando o método bootstrap apresentado na seção 4.4 do capítulo 4, para o caso da função discriminante linear de Fisher, apresentada na subseção 2.4.1 do capítulo 2, equação (2.11).

Para introduzir o método desta seção e da próxima seção, a idéia do método de bootstrap será apresentada numa forma mais ampla do que aquela da seção 4.4.

A idéia do método bootstrap é construir uma distribuição empírica para uma estatística de interesse e usar esta distribuição para construir intervalos de confiança e testar hipóteses. Se B amostras com reposição são obtidas da amostra T e seja $S(T)$ uma estatística de interesse, a distribuição empírica da estatística $S(T)$ pode ser de certa forma representada pelo histograma dos B valores da estatística $S(T)$ calculada para cada uma das B amostras bootstrap (ver Owen (2001, pp. 266–271)).

Esta distribuição empírica de bootstrap pode então ser utilizada para construir intervalos de confiança ou testes de hipóteses. O método bootstrap percentil para construção de intervalos de confiança pode ser encontrado em Efron & Tibshirani (1993, pp. 168–176).

Um novo algoritmo, baseado no método bootstrap percentil [Efron & Tibshirani (1993, pp. 168–176)] para definição dessa fronteira pode ser descrito pelos seguintes passos:

Dados: amostra de treinamento $T = \{\boldsymbol{x}_i, y_i\}_{i=1}^{n}$, onde $\boldsymbol{x}_i \in \mathcal{X}$ e $y_i \in \{1, 2\}$ tal que se $y_i = 1$ então $y_i \in \Pi_1$ e se $y_i = 2$ então $y_i \in \Pi_2$; amostra de teste $T^{\dagger} = \{\boldsymbol{x}_j, y_j\}_{j=1}^{r}$; e o número de amostras bootstrap B.

1. Gere B amostras bootstrap $T^{*(1)}, \ldots, T^{*(B)}$.

2. Obtenha as quantidades $\widehat{m}^{*(1)}, \ldots, \widehat{m}^{*(B)}$ para a amostra de treinamento, sendo $\widehat{m}^{*(b)}$, $b = 1, \ldots, B$, calculados de acordo com a expressão (2.11).

3. Ordene as quantidades $\widehat{m}^{*[1]} < \widehat{m}^{*[2]} < \ldots < \widehat{m}^{*[B-1]} < \widehat{m}^{*[B]}$.

4. Para um nível de confiança nominal $(1 - \alpha)$ obtenha os quantis de interesse $m^{*[B(\alpha)]}$ e $m^{*[B(1-\alpha)]}$, onde $[B(\alpha)]$ e $[B(1 - \alpha)]$ representam as partes inteiras de $B(\alpha)$ e $B(1 - \alpha)$ respectivamente.

5. Aloque as observações da amostra de teste de acordo com a regra:

$$\begin{aligned} &\text{alocar } \boldsymbol{x} \text{ para } \Pi_1 \text{ se} && z \geq m^{*[B(1-\alpha)]} \\ &\text{alocar } \boldsymbol{x} \text{ para } \Pi_2 \text{ se} && z \leq m^{*[B(\alpha)]} \\ &\text{alocar } \boldsymbol{x} \text{ para } \Pi_3 \text{ se} && m^{*[B(\alpha)]} \leq z \leq m^{*[B(1-\alpha)]} \end{aligned} \tag{6.6}$$

onde z é calculado de acordo com a expressão (2.11) e Π_3 é o conjunto definido como fronteira.

Note que uma possível crítica a este método é que um novo indivíduo poderá ser classificado em três populações, porém existem apenas duas populações. A população Π_3 tem como objetivo fornecer ao usuário final uma flexibilidade em relação à classificação. Dessa forma, o usuário final irá indicar se este indivíduo pertence a Π_1 ou Π_2, as duas populações que realmente existem.

Outro aspecto relevante é que este método pode ser utilizado conjuntamente com qualquer método de classificação (análise discriminante quadrática, regressão logística, análise discriminante de núcleo, etc.), porém apenas ilustraremos o caso da análise discriminante linear de Fisher, por simplicidade.

Em métodos de bootstrap para intervalos de confiança devemos nos preocupar com as propriedades da estatística utilizada [Hall (1992, pp. 83–91)]. No algoritmo acima foi utilizado o método bootstrap percentil para construir um intervalo de confiança para a regra de classificação, porém, não foi verificado se a estatística $\widehat{m}$ é assintoticamente pivotal, ou seja, se a distribuição assintótica desta estatística não depende de parâmetros desconhecidos.

Mesmo que a estatística do algoritmo desta seção não seja assintoticamente pivotal, isto não implica que o método apresentado seja inadequado. Segundo Hall (1992, pp. 83–91) o que acontece com estatísticas assintoticamente não pivotais é que o erro é de magnitude de $\mathcal{O}(n^{-1})$, onde n representa o tamanho da amostra e $\mathcal{O}(\cdot)$ é definido no apêndice A, enquanto que para estatísticas assintoticamente pivotais o erro é de magnitude $\mathcal{O}(n^{-2})$.

Exemplos de estatísticas assintoticamente pivotais podem ser encontrados em Fisher et al. (1996) e Amaral et al. (2007). Os contextos onde estas estatísticas foram desenvolvidas nestes artigos foram bastante complexos. Dessa forma, a obtenção e a prova que estas estatísticas são assintoticamente pivotais foram tarefas de grande dificuldade, por exemplo, veja Amaral et al. (2007).

Para ilustrar o método desta seção considere o seguinte exemplo, em que geramos uma amostra de treinamento composta por 50 observações, geradas de acordo como cenário C apre-

sentado na tabela 5.1 e na figura 5.1 do capítulo anterior, sendo as 25 primeiras observações pertencentes à população Π_1 e as 25 restantes pertencentes à população Π_2 e uma amostra de teste com o mesmo número de observações e a mesma estrutura da amostra de treinamento. Na tabela 6.2 são apresentadas as observações cujas populações foram preditas incorretamente pelo método de classificação de Fisher e as respectivas classes preditas pelo método da fronteira. A observação 7 pertence à população Π_1, contudo foi classificada por ambos os métodos como pertencente à população Π_2, ou seja, o método da fronteira, neste caso, não conseguiu corrigir a classificação incorreta feita pelo método de Fisher, alocando a observação 7 para a região de fronteira. Porém, a observação 7 possui um padrão de comportamento tão semelhante às observações da população 2 que não deveríamos esperar que esta observação fosse alocada para uma região de fronteira. A observação 25, pertencente à população Π_1, foi incorretamente classificada pela função discriminante linear de Fisher para a população Π_2, enquanto que, pelo método desta seção, esta observação foi classificada como pertencente a uma região de fronteira. Resultados semelhantes ocorreram para as observações 27 e 31. Retomando o exemplo de gerente de um banco, se esse conjunto de dados representasse um conjunto de clientes proponentes a empréstimos, o gerente poderia decidir, com base na sua experiência e em uma análise pormenorizada, se concederia ou não empréstimo a esses clientes classificados como pertencentes a uma região de fronteira entre as duas populações. Na figura 6.1 apresentamos o gráfico de dispersão para o conjunto de teste. Podemos notar que as observações identificadas pelo método realmente se encontram em uma região de fronteira.

Tabela 6.2: Observações classificadas incorretamente pelo método de Fisher (Fisher) e respectivas classes preditas pelo método da Fronteira de Classificação (Fronteira)

Observação	População Verdadeira	Fisher	Fronteira
7	Π_1	Π_2	Π_2
25	Π_1	Π_2	Π_3
27	Π_2	Π_1	Π_3
31	Π_2	Π_1	Π_3

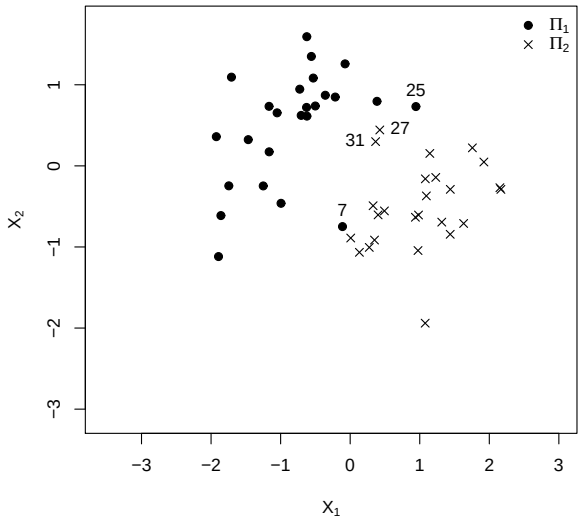

Figura 6.1: Gráfico de dispersão para o conjunto de teste considerado na ilustração.

6.4 Intervalos de Confiança Bootstrap para a Taxa de Erro de Classificação

A taxa de erro de classificação é uma variável aleatória que depende da distribuição das g populações (na maioria das situações desconhecidas) e do método de classificação empregado, que pode ser visto no contexto usual da estatística como um tratamento. Analiticamente, é muito difícil (ou até mesmo impossível) encontrar a distribuição da taxa de erro de classificação.

Utilizando-se o método bootstrap é possível encontrar a distribuição empírica da taxa de erro, similarmente ao que foi feito na seção 6.3. Com a distribuição empírica da taxa de erro e com alguns resultados teóricos relevantes, é possível construir intervalos de confiança e testar a hipótese de que dois ou mais métodos de classificação são iguais.

Nesta seção, iremos descrever um novo método para determinar um intervalo de confiança para a taxa de erro. Entretanto, o desenvolvimento de testes de hipóteses será deixado como um trabalho futuro devido ao fato de sua complexidade estar além dos objetivos deste trabalho.

Um novo algoritmo, baseado no método bootstrap percentil [Efron & Tibshirani (1993,

pp. 168–176)] utilizando o estimador bootstrap .632 para estimar a taxa de erro de uma regra de classificação $F(\boldsymbol{x})$, para definição deste intervalo de confiança, pode ser descrito pelos seguintes passos:

Dados: amostra de treinamento $T = \{\boldsymbol{x}_i, y_i\}_{i=1}^{n}$, onde $\boldsymbol{x}_i \in \mathcal{X}$ e $y_i \in \{1, 2\}$ tal que se $y_i = 1$ então $y_i \in \Pi_1$ e se $y_i = 2$ então $y_i \in \Pi_2$; e o número de amostras bootstrap B.

1. Gere B amostras bootstrap $T^{*(1)}, \ldots, T^{*(B)}$.

2. Obtenha as taxas de erro, $\mathrm{Err}^{(.632)*(1)}, \ldots, \mathrm{Err}^{(.632)*(B)}$, para as B amostras bootstrap utilizando o estimador bootstrap .632 dado pela equação (4.7) para algum classificador $F(\boldsymbol{x})$ de interesse, como por exemplo, a função discriminante linear de Fisher, dada pela equação (2.11).

3. Ordene as quantidades $\widehat{\mathrm{Err}}^{(.632)*[1]} \leq \widehat{\mathrm{Err}}^{(.632)*[2]} \leq \ldots \leq \widehat{\mathrm{Err}}^{(.632)*[B-1]} \leq \widehat{\mathrm{Err}}^{(.632)*[B]}$.

4. Para um nível de confiança nominal $(1-\alpha)$ obtenha os quantis de interesse $\widehat{\mathrm{Err}}^{(.632)*[B(\alpha)]}$ e $\widehat{\mathrm{Err}}^{(.632)*[B(1-\alpha)]}$, onde $[B(\alpha)]$ e $[B(1-\alpha)]$ representam as partes inteiras de $B(\alpha)$ e $B(1-\alpha)$ respectivamente. Assim, o intervalo de confiança bootstrap de nível $(1-\alpha)$ para a taxa de erro é dado por
$$\left[\widehat{\mathrm{Err}}^{(.632)*[B(\alpha)]}; \widehat{\mathrm{Err}}^{(.632)*[B(1-\alpha)]}\right].$$

O intervalo de confiança representa uma informação mais completa do que a estimativa pontual da taxa de erro. Para um particular conjunto de dados, o usuário final poderá avaliar qual será o valor máximo e o valor mínimo do método de classificação que ele esteja utilizando, informação que não é obtida com a simples estimativa pontual da referida taxa.

Testes de hipóteses comparando diferentes métodos de classificação é outro ponto de interesse. Bons estudos sobre testes de hipóteses baseados no método bootstrap são apresentados por Fisher & Hall (1990) e Hall & Wilson (1991). Estes artigos apresentam dois critérios para a formulação de testes de hipóteses baseados no método bootstrap: usar uma estatística assintoticamente pivotal e executar a reamostragem sob a hipótese nula. No contexto de classificação, onde nosso

interesse seria comparar dois ou mais métodos de classificação, a hipótese nula seria que as taxas de erro desses métodos são iguais.

Como ilustração, considere novamente o conjunto de dados do salmão introduzido na seção 3.3.1, a função discriminante linear de Fisher dada pela equação (2.11) e a regra discriminante de núcleo dada pela definição (3.3.1). Na figura 6.2 é apresentada a distribuição empírica da taxa de erro aparente como um exemplo da aplicação do algoritmo desta seção utilizando a função discriminante de Fisher. A estimativa pontual para a taxa de erro aparente foi igual a $0,070$ e o intervalo de confiança de nível nominal 5% correspondeu a $[0,030; 0,110]$.

Observando a figura 6.3, referente à representação gráfica da distribuição empírica da taxa de erro bootstrap .632 calculada utilizando o algoritmo desta seção com a função discriminante de Fisher, podemos notar como o erro aparente pode ser viesado [Efron (1986)]. A estimativa pontual e o intervalo para o erro calculado pelo estimador bootstrap .632 foram $0,075$ e $[0,037; 0,116]$, respectivamente. Dessa forma podemos ver como a distribuição empírica sofre uma mudança de locação. Esta mudança de locação está ilustrada na figura 6.4.

Na figura 6.5 podemos observar a distribuição empírica da taxa de erro aparente obtida utilizando o algoritmo desta seção com a função discriminante de núcleo. A estimativa pontual para a taxa de erro aparente para este método foi igual a $0,050$ e o intervalo de confiança bootstrap de nível nominal 5% correspondeu a $[0,010; 0,070]$.

A distribuição empírica da taxa de erro bootstrap .632 obtida utilizando o algoritmo desta seção com a função discriminante de núcleo pode ser observada na figura 6.6. A estimativa pontual e o intervalo para o erro calculado pelo estimador bootstrap .632 foram $0,077$ e $[0,027; 0,096]$, respectivamente. Observando a figura 6.7, podemos mais uma vez observar que a distribuição empírica sofre mudança de locação, indicando que a taxa de erro aparente é uma estimativa otimista da taxa de erro verdadeira.

Através dos intervalos de confiança bootstrap obtidos podemos comparar o método discriminante de Fisher com o método discriminante de núcleo. Observando os intervalos de confiança referentes às taxas de erro bootstrap .632 podemos concluir que com o método de núcleo podemos obter taxas de erro menores do que as taxas de erro obtidas com o método discriminante de

Fisher.

O esforço computacional para o cálculo do intervalo de confiança quando o estimador bootstrap .632 é utilizado deve ser destacado. Para o exemplo do salmão, considerado nesta seção, foram utilizadas 400 amostras bootstrap e para o cálculo de estimador bootstrap .632 foram utilizadas 500 amostras utilizando-se a expressão (4.6) para cada uma das 400 amostras bootstrap. Portanto, foram geradas $400 \times 500 = 200000$ amostras bootstrap o que corresponde a um considerável esforço computacional.

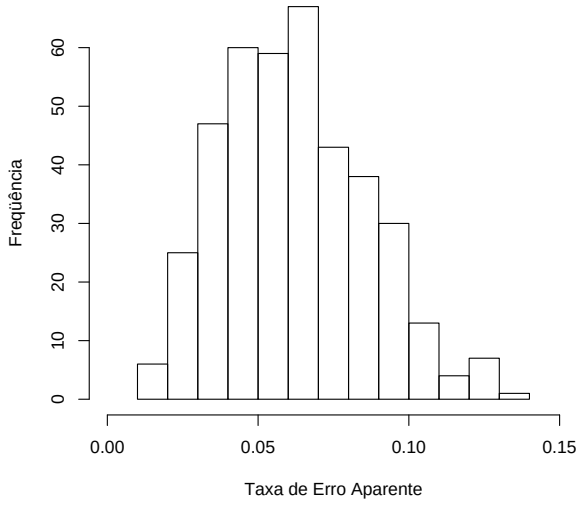

Figura 6.2: Representação gráfica da distribuição empírica da taxa de erro aparente (equação (4.2)) obtida utilizando o algoritmo desta seção com a função discriminante linear de Fisher.

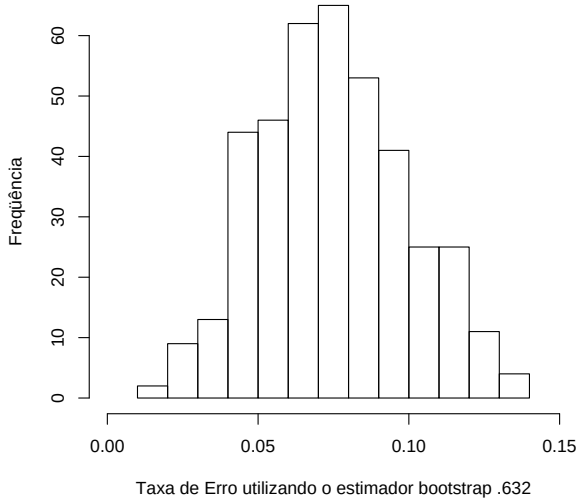

Figura 6.3: Representação gráfica da distribuição empírica da taxa de erro calculada através do estimador bootstrap .632 (equação (4.7)) obtida utilizando o algoritmo desta seção com a função discriminante linear de Fisher.

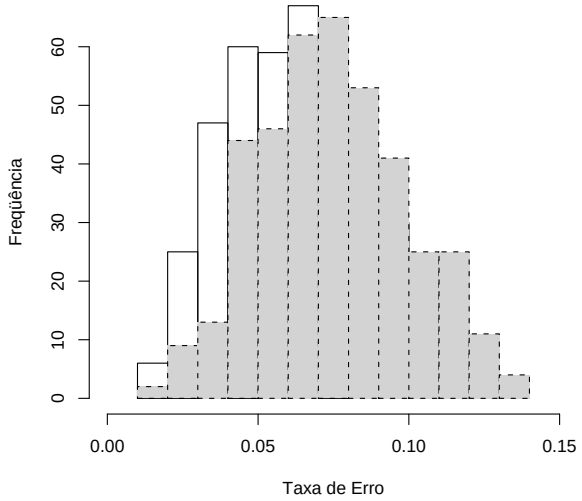

Figura 6.4: Representação gráfica da sobreposição da distribuição empírica da Taxa de Erro Aparente (Cor branca/linhas sólidas) e da distribuição empírica da Taxa de Erro bootstrap .632 (Cor cinza/linhas tracejadas) obtidas utilizando o algoritmo desta seção com a função discriminante linear de Fisher.

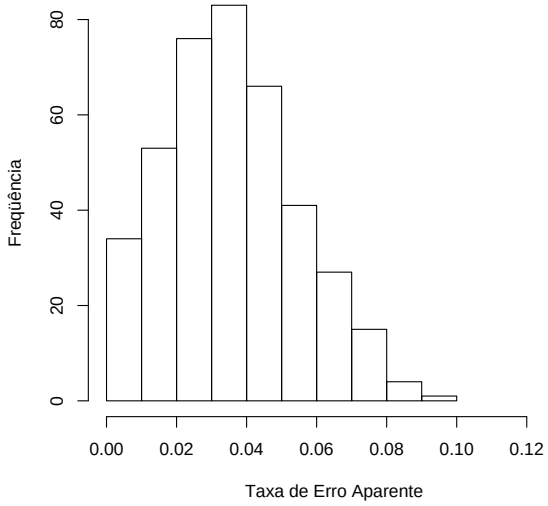

Figura 6.5: Representação gráfica da distribuição empírica da taxa de erro aparente (equação (4.2)) obtida utilizando o algoritmo desta seção com a função discriminante de núcleo.

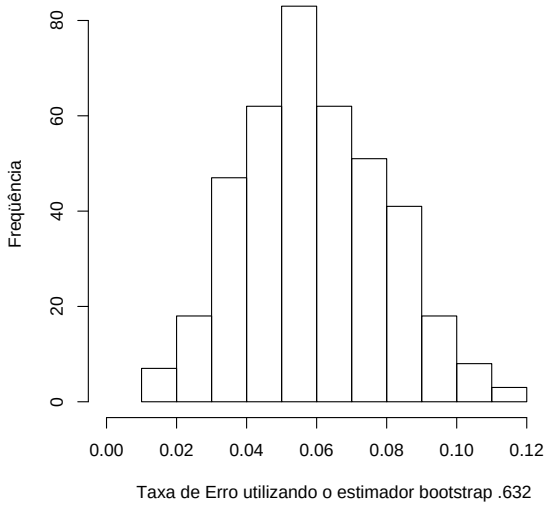

Figura 6.6: Representação gráfica da distribuição empírica da taxa de erro calculada através do estimador bootstrap .632 (equação (4.7)) obtida utilizando o algoritmo desta seção com a função discriminante de núcleo.

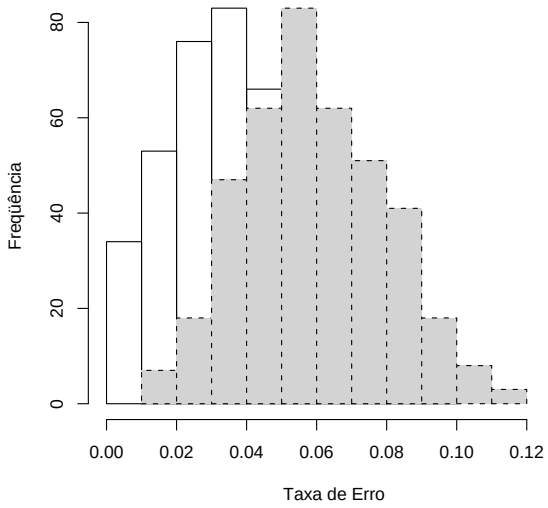

Figura 6.7: Representação gráfica da sobreposição da distribuição empírica da Taxa de Erro Aparente (Cor branca/linhas sólidas) e da distribuição empírica da Taxa de Erro bootstrap .632 (Cor cinza/linhas tracejadas) obtidas utilizando o algoritmo desta seção com a função discriminante de núcleo.

CAPÍTULO 7

Conclusões e Direções para Trabalhos Futuros

7.1 Conclusões

No desenvolvimento deste trabalho, foram apresentados dois métodos de classificação clássicos, a análise discriminante linear e a análise discriminante quadrática, e um método não-paramétrico de classificação, baseado na estimação das funções de densidade de probabilidade das populações através de funções núcleo, a análise discriminante de núcleo. Fizemos uma revisão destes métodos e apresentamos um estudo de simulação e algumas aplicações práticas, com o objetivo de comparar os desempenhos destes métodos. Nesta comparação, vimos que, em situações cuja complexidade é pequena, como nos cenários A e B, funções discriminantes lineares, quadráticas e de núcleo produzem bons resultados e comportamentos muito semelhantes. Em situações como a apresentada no cenário C, os três métodos fornecem resultados satisfatórios, entretanto, o método discriminante de núcleo foi o que apresentou melhores resultados. Por outro lado, em situações mais complexas, como os cenários D, E e F, o método discriminante de núcleo apresenta desempenho muito superior aos demais métodos. As aplicações a dados reais permitiram ilustrar a utilidade dos métodos em problemas reais do interesse de diversas áreas, como no exemplo do salmão, a classificação dos salmões como provenientes de águas do estado americano do Alasca ou de águas do Canadá, e no exemplo dos pacientes que se submeteram a uma cirurgia para retirada de um tumor na mama, a classificação dos pacientes em duas classes: pacientes

que sobreviveram após 5 anos e pacientes que faleceram dentro do intervalo de 5 anos. Estas aplicações mostram que estes métodos podem ser úteis na resolução de problemas de distintas áreas. Outro ponto relevante dessas aplicações práticas foi a utilização do estimador bootstrap .632 para estimar as taxas de erro de métodos de classificação.

O método boosting foi explorado com o objetivo de reduzir a taxa de erro de classificação obtida pela função discriminante linear de Fisher. Porém, no estudo de simulação realizado, esta redução não foi significativa. Apresentamos uma nova formulação para a questão da atribuição de pesos às observações no método boosting aplicado à análise discriminante de Fisher, relacionada a uma mudança conceitual na forma como esses pesos são incorporados no algoritmo boosting e, embora essa mudança não tenha ocasionado uma melhora no método, ela serviu para mostrar que uma mudança na forma como os pesos são atribuídos pode ser uma alternativa para um aperfeiçoamento do método.

Outro ponto abordado nesta dissertação foi a utilização do método bootstrap para definição de uma fronteira de classificação em análise discriminante de Fisher, que ajudasse o usuário final do método de classificação no processo de tomadas de decisões. No exemplo apresentado, esse novo método apresentou resultados bastante interessantes, o que nos leva a acreditar que ele pode ser explorado em problemas práticos. O método bootstrap também foi utilizado para definição de uma nova metodologia para construção de intervalos de confiança para a taxa de erro de classificação. No exemplo apresentado, ilustramos que o cálculo de intervalos de confiança para taxas de erro de um método de classificação se constitui numa informação mais completa do que a estimação pontual da taxa de erro.

7.2 Direções para Trabalhos Futuros

Essa seção tem o objetivo de apontar direções para trabalhos futuros relacionados ao conteúdo abordados nesta dissertação. Abaixo listamos algumas idéias deixadas para trabalhos futuros. Algumas dessas idéias já estão sendo trabalhadas e esperamos que possam resultar em bons trabalhos.

- Implementar e explorar o método boosting aplicado à análise discriminante de núcleo, baseando-nos nos trabalhos de Di Marzio & Taylor (2005*a*) e Di Marzio & Taylor (2005*b*).

- Avaliar outros critérios para obtenção e incorporação de pesos no algoritmo boosting em análise discriminante de Fisher, de forma a obter melhores resultados.

- Estudar as propriedades assintóticas relativas aos métodos da fronteira de classificação e do intervalo de confiança bootstrap para a taxa de erro. Além disso, tentar formular testes de hipóteses baseados no método bootstrap, para comparar dois ou mais métodos de classificação.

- Avaliar o método discriminante de núcleo utilizado com outros tipos de função núcleo e outras formas de obtenção da matriz de largura de banda, $\mathbf{H}$.

APÊNDICE A

Notação de Ordem

Suponha que a_n e b_n são seqüências de números reais. A notação

$$a_n = \mathcal{O}(b_n) \tag{A.1}$$

significa que

$$\limsup_{n\to\infty} \frac{|a_n|}{|b_n|} < \infty.$$

Por exemplo, se $a_n = \mu + n\sigma^2$, onde μ e σ são constantes, então $a_n = \mathcal{O}(n)$, pois,

$$\limsup_{n\to\infty} \frac{|\mu + n\sigma^2|}{|n|} = \sigma^2 < \infty.$$

Nesta dissertação, apenas a notação de ordem $\mathcal{O}(\cdot)$ para seqüências de variáveis reais é utilizada.

APÊNDICE B

Programas de Simulação

Neste capítulo são apresentadas as rotinas em linguagem R utilizadas para obtenção dos resultados de simulação apresentados nesta dissertação.

B.1 Comparação entre Análise Disriminante Clássica e de Núcleo

```
##################################################
#                                                #
# Programa de Simulação com misturas de normais #
# utilizado na comparação entre os métodos dis- #
# criminantes linear, quadrático e de núcleo.   #
#                                                #
##################################################

## Bibliotecas necessárias
library(MASS)
library(ks)

##################################################

## Parâmetros gerais
R <- 10000 ## Número de réplicas de Monte Carlo
N <- 50 ## Tamanho das amostras de treinamento
n <- 1000 ## Tamanho da amostra de teste

y <- sort(rep(c(-1, 1), N/2))
ytst <- sort(rep(c(-1, 1), n/2))

############## CENÁRIOS DE SIMULAÇÃO ##############

## CENÁRIO 1 ##

mus1.1 <- c(1, -1)
mus2.1 <- c(-1, 1)

sigmas1.1 <- matrix(c(4/9, 14/45, 14/45, 4/9), 2, 2)
sigmas2.1 <- matrix(c(4/9, 0, 0, 4/9), 2, 2)
```

```
set.seed(1234567)
xtst.1 <- rbind(rmvnorm.mixt(n/2, mus1.1, sigmas1.1, 1),
                rmvnorm.mixt(n/2, mus2.1, sigmas2.1, 1))

erro.lda.1 <- numeric(R)
erro.qda.1 <- numeric(R)
erro.kda.1 <- numeric(R)

## CENÁRIO 2 ##

mus1.2 <- rbind(c(-3/2, -3/2), c(1/2, 1/2))
mus2.2 <- rbind(c(3/2, 3/2), c(-1/2, -1/2))

sigmas1.2 <- rbind(matrix(c(4/5, -1/2, -1/2, 4/5), 2, 2),
                   matrix(c(4/5, -1/2, -1/2, 4/5), 2, 2))
sigmas2.2 <- rbind(matrix(c(4/5, -1/2, -1/2, 4/5), 2, 2),
                   matrix(c(4/5, -1/2, -1/2, 4/5), 2, 2))

props.2 <- c(1/2, 1/2)

set.seed(1234567)
xtst.2 <- rbind(rmvnorm.mixt(n/2, mus1.2, sigmas1.2, props.2),
                rmvnorm.mixt(n/2, mus2.2, sigmas2.2, props.2))

erro.lda.2 <- numeric(R)
erro.qda.2 <- numeric(R)
erro.kda.2 <- numeric(R)

## CENÁRIO 3 ##

mus1.3 <- rbind(c(-3/2, 0), c(3/2, 0))
mus2.3 <- c(0, 0)

sigmas1.3 <- rbind(matrix(c(3/10, 1/4, 1/4, 3/10), 2, 2),
                   matrix(c(3/10, 1/4, 1/4, 3/10), 2, 2))
sigmas2.3 <- matrix(c(4/5, 2/5, 2/5, 1), 2, 2)

props.3 <- c(1/2, 1/2)

set.seed(1234567)
xtst.3 <- rbind(rmvnorm.mixt(n/2, mus1.3, sigmas1.3, props.3),
                rmvnorm.mixt(n/2, mus2.3, sigmas2.3, 1))

erro.lda.3 <- numeric(R)
erro.qda.3 <- numeric(R)
erro.kda.3 <- numeric(R)

## CENÁRIO 4 ##

mus1.4 <- c(-1, 1/2)
mus2.4 <- c(1, -1/2)

sigmas1.4 <- matrix(c(2/3, 1/5, 1/5, 4/9), 2, 2)
sigmas2.4 <- matrix(c(2/3, 1/5, 1/5, 4/9), 2, 2)

set.seed(1234567)
xtst.4 <- rbind(rmvnorm.mixt(n/2, mus1.4, sigmas1.4, 1),
                rmvnorm.mixt(n/2, mus2.4, sigmas2.4, 1))

erro.lda.4 <- numeric(R)
erro.qda.4 <- numeric(R)
erro.kda.4 <- numeric(R)

## CENÁRIO 5 ##
```

```
mus1.5 <- c(-1, 1)
mus2.5 <- c(1, -1/2)

sigmas1.5 <- matrix(c(2/3, 1/5, 1/5, 1/3), 2, 2)
sigmas2.5 <- matrix(c(2/3, 1/5, 1/5, 1/3), 2, 2)

set.seed(1234567)
xtst.5 <- rbind(rmvnorm.mixt(n/2, mus1.5, sigmas1.5, 1),
                rmvnorm.mixt(n/2, mus2.5, sigmas2.5, 1))

erro.lda.5 <- numeric(R)
erro.qda.5 <- numeric(R)
erro.kda.5 <- numeric(R)

## CENÁRIO 6 ##

mus1.6 <- rbind(c(-3/2, 0), c(3/2, 0))
mus2.6 <- c(0, 0)

sigmas1.6 <- rbind(matrix(c(1/12, 1/4, 1/4, 1), 2, 2),
                   matrix(c(1/12, 1/4, 1/4, 1), 2, 2))
sigmas2.6 <- matrix(c(4/9, 1/5, 1/5, 4/9), 2, 2)

props.6 <- c(1/2, 1/2)

set.seed(1234567)
xtst.6 <- rbind(rmvnorm.mixt(n/2, mus1.6, sigmas1.6, props.6),
                rmvnorm.mixt(n/2, mus2.6, sigmas2.6, 1))

erro.lda.6 <- numeric(R)
erro.qda.6 <- numeric(R)
erro.kda.6 <- numeric(R)

################################################

set.seed(321)
inicio <- proc.time()[1]
## Início do laço de Monte Carlo
for (j in 1 : R) {

    ## CENÁRIO 1 ##

    x.1 <- rbind(rmvnorm.mixt(N/2, mus1.1, sigmas1.1, 1),
                 rmvnorm.mixt(N/2, mus2.1, sigmas2.1, 1))
    ## Análise discriminante linear
    fit1.1 <- pda(x.1, y, xtst.1, type = "line")
    erro.lda.1[j] <- compare(fit1.1, ytst)$error
    ## Análise discriminante quadrática
    fit2.1 <- pda(x.1, y, xtst.1, type = "quad")
    erro.qda.1[j] <- compare(fit2.1, ytst)$error
    ## Análise discriminante kernel
    H.1 <- Hkda(x.1, y, bw = "lscv")
    fit3.1 <- kda(x.1, y, H.1, xtst.1)
    erro.kda.1[j] <- compare(fit3.1, ytst)$error

    ## CENÁRIO 2 ##

    x.2 <- rbind(rmvnorm.mixt(N/2, mus1.2, sigmas1.2, props.2),
                 rmvnorm.mixt(N/2, mus2.2, sigmas2.2, props.2))
    ## Análise discriminante linear
    fit1.2 <- pda(x.2, y, xtst.2, type = "line")
    erro.lda.2[j] <- compare(fit1.2, ytst)$error
    ## Análise discriminante quadrática
    fit2.2 <- pda(x.2, y, xtst.2, type = "quad")
    erro.qda.2[j] <- compare(fit2.2, ytst)$error
```

```
## Análise discriminante kernel
H.2 <- Hkda(x.2, y, bw = "lscv")
fit3.2 <- kda(x.2, y, H.2, xtst.2)
erro.kda.2[j] <- compare(fit3.2, ytst)$error

## CENÁRIO 3 ##

x.3 <- rbind(rmvnorm.mixt(N/2, mus1.3, sigmas1.3, props.3),
             rmvnorm.mixt(N/2, mus2.3, sigmas2.3, 1))
## Análise discriminante linear
fit1.3 <- pda(x.3, y, xtst.3, type = "line")
erro.lda.3[j] <- compare(fit1.3, ytst)$error
## Análise discriminante quadrática
fit2.3 <- pda(x.3, y, xtst.3, type = "quad")
erro.qda.3[j] <- compare(fit2.3, ytst)$error
## Análise discriminante kernel
H.3 <- Hkda(x.3, y, bw = "lscv")
fit3.3 <- kda(x.3, y, H.3, xtst.3)
erro.kda.3[j] <- compare(fit3.3, ytst)$error

## CENÁRIO 4 ##

x.4 <- rbind(rmvnorm.mixt(N/2, mus1.4, sigmas1.4, 1),
             rmvnorm.mixt(N/2, mus2.4, sigmas2.4, 1))
## Análise discriminante linear
fit1.4 <- pda(x.4, y, xtst.4, type = "line")
erro.lda.4[j] <- compare(fit1.4, ytst)$error
## Análise discriminante quadrática
fit2.4 <- pda(x.4, y, xtst.4, type = "quad")
erro.qda.4[j] <- compare(fit2.4, ytst)$error
## Análise discriminante kernel
H.4 <- Hkda(x.4, y, bw = "lscv")
fit3.4 <- kda(x.4, y, H.4, xtst.5)
erro.kda.4[j] <- compare(fit3.4, ytst)$error

## CENÁRIO 5 ##

x.5 <- rbind(rmvnorm.mixt(N/2, mus1.5, sigmas1.5, 1),
             rmvnorm.mixt(N/2, mus2.5, sigmas2.5, 1))
## Análise discriminante linear
fit1.5 <- pda(x.5, y, xtst.5, type = "line")
erro.lda.5[j] <- compare(fit1.5, ytst)$error
## Análise discriminante quadrática
fit2.5 <- pda(x.5, y, xtst.5, type = "quad")
erro.qda.5[j] <- compare(fit2.5, ytst)$error
## Análise discriminante kernel
H.5 <- Hkda(x.5, y, bw = "lscv")
fit3.5 <- kda(x.5, y, H.5, xtst.5)
erro.kda.5[j] <- compare(fit3.5, ytst)$error

## CENÁRIO 6 ##

x.6 <- rbind(rmvnorm.mixt(N/2, mus1.6, sigmas1.6, props.6),
             rmvnorm.mixt(N/2, mus2.6, sigmas2.6, 1))
## Análise discriminante linear
fit1.6 <- pda(x.6, y, xtst.6, type = "line")
erro.lda.6[j] <- compare(fit1.6, ytst)$error
## Análise discriminante quadrática
fit2.6 <- pda(x.6, y, xtst.6, type = "quad")
erro.qda.6[j] <- compare(fit2.6, ytst)$error
## Análise discriminante kernel
H.6 <- Hkda(x.6, y, bw = "lscv")
fit3.6 <- kda(x.6, y, H.6, xtst.6)
erro.kda.6[j] <- compare(fit3.6, ytst)$error
```

```
}
tempo <- proc.time()[1] - inicio[1]
result.cen1 <- matrix(c(mean(erro.lda.1), mean(erro.qda.1), mean(erro.kda.1),
                        sd(erro.lda.1), sd(erro.qda.1), sd(erro.kda.1)), 3, 2)
rownames(result.cen1) <- c("LDA", "QDA", "KDA")
colnames(result.cen1) <- c("Média", "Desvio Padrão")

result.cen2 <- matrix(c(mean(erro.lda.2), mean(erro.qda.2), mean(erro.kda.2),
                        sd(erro.lda.2), sd(erro.qda.2), sd(erro.kda.2)), 3, 2)
rownames(result.cen2) <- c("LDA", "QDA", "KDA")
colnames(result.cen2) <- c("Média", "Desvio Padrão")

result.cen3 <- matrix(c(mean(erro.lda.3), mean(erro.qda.3), mean(erro.kda.3),
                        sd(erro.lda.3), sd(erro.qda.3), sd(erro.kda.3)), 3, 2)
rownames(result.cen3) <- c("LDA", "QDA", "KDA")
colnames(result.cen3) <- c("Média", "Desvio Padrão")

result.cen4 <- matrix(c(mean(erro.lda.4), mean(erro.qda.4), mean(erro.kda.4),
                        sd(erro.lda.4), sd(erro.qda.4), sd(erro.kda.4)), 3, 2)
rownames(result.cen4) <- c("LDA", "QDA", "KDA")
colnames(result.cen4) <- c("Média", "Desvio Padrão")

result.cen5 <- matrix(c(mean(erro.lda.5), mean(erro.qda.5), mean(erro.kda.5),
                        sd(erro.lda.5), sd(erro.qda.5), sd(erro.kda.5)), 3, 2)
rownames(result.cen5) <- c("LDA", "QDA", "KDA")
colnames(result.cen5) <- c("Média", "Desvio Padrão")

result.cen6 <- matrix(c(mean(erro.lda.6), mean(erro.qda.6), mean(erro.kda.6),
                        sd(erro.lda.6), sd(erro.qda.6), sd(erro.kda.6)), 3, 2)
rownames(result.cen6) <- c("LDA", "QDA", "KDA")
colnames(result.cen6) <- c("Média", "Desvio Padrão")

sink("resultados50-R10000.txt")
cat("N = ",N,"\n")
cat("\nCENÁRIO 1\n")
result.cen1
cat("\n-------------------------\n")
cat("\nCENÁRIO 2\n")
result.cen2
cat("\n-------------------------\n")
cat("\nCENÁRIO 3\n")
result.cen3
cat("\n-------------------------\n")
cat("\nCENÁRIO 4\n")
result.cen4
cat("\n-------------------------\n")
cat("\nCENÁRIO 5\n")
result.cen5
cat("\n-------------------------\n")
cat("\nCENÁRIO 6\n")
result.cen6
cat("\n-------------------------\n")
cat("\nTempo de execução: ",tempo,"\n")
sink()
```

B.2 Boosting em Análise Discriminante Linear de Fisher

```
##########################################
#                                        #
# Implementa a função discriminante linear #
# de Fisher para o caso de dois grupos     #
#                                        #
##########################################
flda <- function(xtrain, group)
```

```
{
    train <- cbind(xtrain, group)
    x1 <- as.matrix(train[group == -1, 1 : (dim(train)[2] - 1)])
    x2 <- as.matrix(train[group == 1, 1 : (dim(train)[2] - 1)])
    mean_x1 <- apply(x1, 2, mean)
    mean_x2 <- apply(x2, 2, mean)
    S1 <- cov(x1)
    S2 <- cov(x2)
    Sc <- (((dim(x1)[1] - 1) * S1) + ((dim(x2)[1] - 1) * S2)) /
          (dim(x1)[1] + dim(x2)[1] - 2)
    l_hat <- t(mean_x1 - mean_x2) %*% solve(Sc)
    m_hat <- .5 * t(mean_x1 - mean_x2) %*% solve(Sc) %*% (mean_x1 + mean_x2)
    means <- rbind(mean_x1, mean_x2)
    rownames(means) <- list("group 1 ", "group 2 ")
    result <- list(means, S1, S2, Sc, l_hat, m_hat)
    names(result) <- list("means", "s1", "s2", "sc", "coef", "threshold")
    result
}

##########################################
#                                        #
# Implementa a função discriminante linear #
# de Fisher para o caso de dois grupos com #
# pesos incorporados no calculo da média e #
# da covariância                         #
#                                        #
##########################################

wflda <- function(xtrain, group, w)
{
    train <- cbind(xtrain, group)
    x1 <- as.matrix(train[group == -1, 1 : (dim(train)[2] - 1)])
    x2 <- as.matrix(train[group == 1, 1 : (dim(train)[2] - 1)])
    w1 <- w[1 : (dim(x1)[1])]
    w2 <- w[((dim(x1)[1]) + 1) : ((dim(x1)[1]) + (dim(x2)[1]))]
    mean_x1 <- apply((w1 * x1), 2, sum) / sum(w1)
    mean_x2 <- apply((w2 * x2), 2, sum) / sum(w2)
    S1 <- matrix(0, dim(x1)[2], dim(x1)[2])
    for(i in 1 : (dim(x1)[1]))
    S1 <- S1 + w1[i] * (x1[i,] - mean_x1) %*% t(x1[i,] - mean_x1)
    S1 <- S1 / ((dim(x1)[1]) - 1)
    S2 <- matrix(0, dim(x2)[2], dim(x2)[2])
    for(j in 1 : (dim(x2)[1]))
    S2 <- S2 + w2[j] * (x2[j,] - mean_x2) %*% t(x2[j,] - mean_x2)
    S2 <- S2 / ((dim(x2)[1]) - 1)
    Sc <- (((dim(x1)[1] - 1) * S1) + ((dim(x2)[1] - 1) * S2)) /
          (dim(x1)[1] + dim(x2)[1] - 2)
    l_hat <- t(mean_x1 - mean_x2) %*% solve(Sc)
    m_hat <- .5 * t(mean_x1 - mean_x2) %*% solve(Sc) %*% (mean_x1 + mean_x2)
    means <- rbind(mean_x1, mean_x2)
    rownames(means) <- list("group 1 ", "group 2 ")
    result <- list(means, S1, S2, Sc, l_hat, m_hat)
    names(result) <- list("means", "s1", "s2", "sc", "coef", "threshold")
    result
}

################################################
#                                              #
# Predição para a função discriminante linear #
# de Fisher para o caso de dois grupos         #
#                                              #
################################################

predict.lda <- function(fit, newdata)
{
```

```
    x <- newdata
    n <- dim(x)[1]
    coef <- fit$coef
    threshold <- fit$threshold
    y_hat <- numeric(n)
    class <- numeric(n)
    for(t in 1 : n)
    {
        y_hat[t] <- coef %*% x[t,]
        if(y_hat[t] >= threshold)
            class[t] <- -1
        else
            class[t] <- 1
    }
    class
}

################################################
#                                              #
# Predição para o boosting em análise          #
# discriminante linear de Fisher para o caso   #
# de dois grupos                               #
#                                              #
################################################

predict.boostlda <- function(fit, newdata)
{
    x <- newdata
    n <- dim(x)[1]
    p <- dim(x)[2]
    coefs <- fit$coefs
    thresholds <- fit$thresholds
    boost.weights <- fit$const
    M <- length(thresholds)
    y_hat <- matrix(0, n, M)
    class <- matrix(0, n, M)
    for (i in 1 : M) {
        for (j in 1 : n) {
            y_hat[j, i] <- coefs[i, ] %*% x[j, ]
            if (y_hat[j, i] >= thresholds[i])
                class[j, i] <- -1
            else
                class[j, i] <- 1
        }
    }
    class.final <- sign(t(t(boost.weights) %*% t(class)))
    result <- list(class, class.final)
    names(result) <- list("class", "class.final")
    result
}

############################################
#                                          #
# Boosting em análise discriminante linear #
# de Fisher para o caso de dois grupos     #
#                                          #
############################################

BoostLDA <- function(xtrn, ytrn, M)
{
    N <- dim(xtrn)[1]
    p <- dim(xtrn)[2]
    w <- rep((1 / N), N)
    f.trn <- matrix(0, N, M)
    F.trn <- numeric(N)
```

```
    I <- matrix(0, N, M)
    error <- numeric(M)
    c <- numeric(M)
    coefs <- matrix(0, M, p)
    thresholds <- numeric(M)
    for (m in 1 : M) {
        xweighted <- w * xtrn
        classifier <- flda(xweighted, ytrn)
        coefs[m, ] <- classifier$coef
        thresholds[m] <- classifier$threshold
        f.trn[, m] <- predict.lda(classifier, xtrn)
        I[, m] <- (f.trn[, m] != ytrn)
        error[m] <- w %*% I[, m]
        c[m] <- .5 * log((1 - error[m]) / error[m])
        w <- w * exp(-c[m] * ytrn * f.trn[, m])
        w <- pmax(w / sum(w), 1e-24)
        F.trn <- F.trn + (c[m] * f.trn[, m])
    }
    class.trn <- sign(F.trn)
    result <- list(coefs, thresholds, error, w, c, class.trn,
                   error.trn = apply(I, 2, mean))
    names(result) <- list("coefs", "thresholds", "error",
                     "weights", "const", "class.trn", "error.trn")
    result
}

##############################################
#                                            #
# Boosting em análise discriminante linear   #
# de Fisher ponderada para o caso de dois    #
# grupos                                     #
#                                            #
##############################################

BoostWLDA <- function(xtrn, ytrn, M)
{
    N <- dim(xtrn)[1]
    p <- dim(xtrn)[2]
    w <- rep((1 / N), N)
    f.trn <- matrix(0, N, M)
    F.trn <- numeric(N)
    I <- matrix(0, N, M)
    error <- numeric(M)
    c <- numeric(M)
    coefs <- matrix(0, M, p)
    thresholds <- numeric(M)
    for (m in 1 : M) {
        classifier <- wflda(xtrn, ytrn, w)
        coefs[m, ] <- classifier$coef
        thresholds[m] <- classifier$threshold
        f.trn[, m] <- predict.lda(classifier, xtrn)
        I[, m] <- (f.trn[, m] != ytrn)
        error[m] <- w %*% I[, m]
        c[m] <- .5 * log((1 - error[m]) / error[m])
        w <- w * exp(-c[m] * ytrn * f.trn[, m])
        w <- pmax(w / sum(w), 1e-24)
        F.trn <- F.trn + (c[m] * f.trn[, m])
    }
    class.trn <- sign(F.trn)
    result <- list(coefs, thresholds, error, w, c,
                   class.trn, error.trn = apply(I, 2, mean))
    names(result) <- list("coefs", "thresholds", "error",
                     "weights", "const", "class.trn", "error.trn")
    result
}
```

```
###################################
#                                 #
# Simulação Análise Discriminante #
# (Seção Boosting)                #
#                                 #
###################################

library(mvtnorm)

R <- 1000    ## Número de Réplicas de Monte Carlo
N <- 10      ## Tamanho de amostra para o conjunto de treinamento
n <- 1000    ## Tamanho de amostra para o conjunto de teste
p <- 20      ## Número de variáveis preditoras

set.seed(123456789)

## Geração do conjunto de teste e das variáveis indicadoras
## da classe para os conjuntos de treinamento e teste
a <- 1 / sqrt(p)
x.tst <- rbind(rmvnorm(n/2, rep(0, p), 4 * diag(1, p, p)),
               rmvnorm(n/2, rep(a, p)))
y.tst <- sort(rep(c(-1, 1), n/2))
y.trn <- sort(rep(c(-1, 1), N/2))

z <- rep(1 / N, N) ## Vetor de pesos (WLDA)

## Variáveis utilizadas para armazenar resultados
## obtidos no laço de Monte Carlo

## Conjunto de Treinamento
N11.trn.1 <- N12.trn.1 <- N21.trn.1 <- N22.trn.1 <- 0

## Conjunto de Teste
N11.tst.1 <- N12.tst.1 <- N21.tst.1 <- N22.tst.1 <- 0

## Início do laço de Monte Carlo
for (j in 1 : R)
{
    x.trn <- rbind(rmvnorm(N/2, rep(0, p), 4 * diag(1, p, p)),
                   rmvnorm(N/2, rep(a, p)))
    fit1 <- flda(x.trn, y.trn)
    pred1.trn <- predict.lda(fit1, x.trn)
    tab1.trn <- table(y.trn, pred1.trn)
    N11.trn.1 <- N11.trn.1 + tab1.trn[1, 1]
    N12.trn.1 <- N12.trn.1 + tab1.trn[1, 2]
    N21.trn.1 <- N21.trn.1 + tab1.trn[2, 1]
    N22.trn.1 <- N22.trn.1 + tab1.trn[2, 2]
    pred1.tst <- predict.lda(fit1, x.tst)
    tab1.tst <- table(y.tst, pred1.tst)
    N11.tst.1 <- N11.tst.1 + tab1.tst[1, 1]
    N12.tst.1 <- N12.tst.1 + tab1.tst[1, 2]
    N21.tst.1 <- N21.tst.1 + tab1.tst[2, 1]
    N22.tst.1 <- N22.tst.1 + tab1.tst[2, 2]
}

matrix1.trn <-  matrix(0, 2, 2)
rownames(matrix1.trn) <- c("True -1","True  1")
colnames(matrix1.trn) <- c("Pred. -1","Pred. 1")
matrix1.trn <- as.table(matrix1.trn)
matrix1.trn[1, 1] <- N11.trn.1
matrix1.trn[1, 2] <- N12.trn.1
matrix1.trn[2, 1] <- N21.trn.1
matrix1.trn[2, 2] <- N22.trn.1
```

```
erro1.trn <- sum(matrix1.trn[row(matrix1.trn) !=
                 col(matrix1.trn)])/sum(matrix1.trn)

matrix1.tst <-  matrix(0, 2, 2)
rownames(matrix1.tst) <- c("True -1","True  1")
colnames(matrix1.tst) <- c("Pred. -1","Pred. 1")
matrix1.tst <- as.table(matrix1.tst)
matrix1.tst[1, 1] <- N11.tst.1
matrix1.tst[1, 2] <- N12.tst.1
matrix1.tst[2, 1] <- N21.tst.1
matrix1.tst[2, 2] <- N22.tst.1

erro1.tst <- sum(matrix1.tst[row(matrix1.tst) !=
                 col(matrix1.tst)])/sum(matrix1.tst)

sink("resultados (LDA).txt")

print("Matriz de Confusão - FLDA - TRN:")
matrix1.trn
print("Taxa de Erro:")
erro1.trn
print("--------------------------")
print("--------------------------")
print("Matriz de Confusão - FLDA - TST:")
matrix1.tst
print("Taxa de Erro:")
erro1.tst
print("--------------------------")

sink()

####################################
#                                  #
# Simulação Boosting em Análise    #
# Discriminante de Fisher Ponderada #
# (Seção Boosting)                 #
#                                  #
####################################

library(mvtnorm)

R <- 100     ## Número de Réplicas de Monte Carlo
N <- 350     ## Tamanho de amostra para o conjunto de treinamento
n <- 1000    ## Tamanho de amostra para o conjunto de teste
p <- 20      ## Número de variáveis preditoras
M <- 100

set.seed(123456789)

## Geração do conjunto de teste e das variáveis indicadoras
## da classe para os conjuntos de treinamento e teste
a <- 1 / sqrt(p)
x.tst <- rbind(rmvnorm(n/2, rep(0, p), 4 * diag(1, p, p)),
               rmvnorm(n/2, rep(a, p)))
y.tst <- sort(rep(c(-1, 1), n/2))
y.trn <- sort(rep(c(-1, 1), N/2))

## Variáveis utilizadas para armazenar resultados
## obtidos no laço de Monte Carlo

## Conjunto de Treinamento
N11.trn.1 <- N12.trn.1 <- N21.trn.1 <- N22.trn.1 <- 0

## Conjunto de Teste
N11.tst.1 <- N12.tst.1 <- N21.tst.1 <- N22.tst.1 <- 0
```

```
## Início do laço de Monte Carlo
for (j in 1 : R)
{
    x.trn <- rbind(rmvnorm(N/2, rep(0, p), 4 * diag(1, p, p)),
                   rmvnorm(N/2, rep(a, p)))
    fit1 <- BoostLDA(x.trn, y.trn, M)
    pred1.trn <- predict.boostlda(fit1, x.trn)$class.final
    tab1.trn <- table(y.trn, pred1.trn)
    N11.trn.1 <- N11.trn.1 + tab1.trn[1, 1]
    N12.trn.1 <- N12.trn.1 + tab1.trn[1, 2]
    N21.trn.1 <- N21.trn.1 + tab1.trn[2, 1]
    N22.trn.1 <- N22.trn.1 + tab1.trn[2, 2]
    pred1.tst <- predict.boostlda(fit1, x.tst)$class.final
    tab1.tst <- table(y.tst, pred1.tst)
    N11.tst.1 <- N11.tst.1 + tab1.tst[1, 1]
    N12.tst.1 <- N12.tst.1 + tab1.tst[1, 2]
    N21.tst.1 <- N21.tst.1 + tab1.tst[2, 1]
    N22.tst.1 <- N22.tst.1 + tab1.tst[2, 2]
}

matrix1.trn <-  matrix(0, 2, 2)
rownames(matrix1.trn) <- c("True -1","True  1")
colnames(matrix1.trn) <- c("Pred. -1","Pred. 1")
matrix1.trn <- as.table(matrix1.trn)
matrix1.trn[1, 1] <- N11.trn.1
matrix1.trn[1, 2] <- N12.trn.1
matrix1.trn[2, 1] <- N21.trn.1
matrix1.trn[2, 2] <- N22.trn.1

erro1.trn <- sum(matrix1.trn[row(matrix1.trn) !=
                 col(matrix1.trn)])/sum(matrix1.trn)

matrix1.tst <-  matrix(0, 2, 2)
rownames(matrix1.tst) <- c("True -1","True  1")
colnames(matrix1.tst) <- c("Pred. -1","Pred. 1")
matrix1.tst <- as.table(matrix1.tst)
matrix1.tst[1, 1] <- N11.tst.1
matrix1.tst[1, 2] <- N12.tst.1
matrix1.tst[2, 1] <- N21.tst.1
matrix1.tst[2, 2] <- N22.tst.1

erro1.tst <- sum(matrix1.tst[row(matrix1.tst) !=
                 col(matrix1.tst)])/sum(matrix1.tst)

sink("resultados (BoostLDA)(N = ).txt")

print("Matriz de Confusão:")
matrix1.trn
print("Taxa de Erro:")
erro1.trn
print("--------------------------")
print("--------------------------")
print("Matriz de Confusão:")
matrix1.tst
print("Taxa de Erro:")
erro1.tst

sink()

#####################################
#                                   #
# Simulação Boosting em Análise     #
# Discriminante de Fisher Ponderada #
# (Seção Boosting)                  #
```

```
#                                    #
######################################

library(mvtnorm)

R <- 100     ## Número de Réplicas de Monte Carlo
N <- 200     ## Tamanho de amostra para o conjunto de treinamento
n <- 1000    ## Tamanho de amostra para o conjunto de teste
p <- 20      ## Número de variáveis preditoras
M <- 100

set.seed(123456789)

## Geração do conjunto de teste e das variáveis indicadoras
## da classe para os conjuntos de treinamento e teste
a <- 1 / sqrt(p)
x.tst <- rbind(rmvnorm(n/2, rep(0, p), 4 * diag(1, p, p)),
         rmvnorm(n/2, rep(a, p)))
y.tst <- sort(rep(c(-1, 1), n/2))
y.trn <- sort(rep(c(-1, 1), N/2))

## Variáveis utilizadas para armazenar resultados
## obtidos no laço de Monte Carlo

## Conjunto de Treinamento
N11.trn.1 <- N12.trn.1 <- N21.trn.1 <- N22.trn.1 <- 0

## Conjunto de Teste
N11.tst.1 <- N12.tst.1 <- N21.tst.1 <- N22.tst.1 <- 0

## Início do laço de Monte Carlo
for (j in 1 : R)
{
    x.trn <- rbind(rmvnorm(N/2, rep(0, p), 4 * diag(1, p, p)),
                   rmvnorm(N/2, rep(a, p)))
    fit1 <- BoostWLDA(x.trn, y.trn, M)
    pred1.trn <- predict.boostlda(fit1, x.trn)$class.final
    tab1.trn <- table(y.trn, pred1.trn)
    N11.trn.1 <- N11.trn.1 + tab1.trn[1, 1]
    N12.trn.1 <- N12.trn.1 + tab1.trn[1, 2]
    N21.trn.1 <- N21.trn.1 + tab1.trn[2, 1]
    N22.trn.1 <- N22.trn.1 + tab1.trn[2, 2]
    pred1.tst <- predict.boostlda(fit1, x.tst)$class.final
    tab1.tst <- table(y.tst, pred1.tst)
    N11.tst.1 <- N11.tst.1 + tab1.tst[1, 1]
    N12.tst.1 <- N12.tst.1 + tab1.tst[1, 2]
    N21.tst.1 <- N21.tst.1 + tab1.tst[2, 1]
    N22.tst.1 <- N22.tst.1 + tab1.tst[2, 2]
}

matrix1.trn <-  matrix(0, 2, 2)
rownames(matrix1.trn) <- c("True -1","True  1")
colnames(matrix1.trn) <- c("Pred. -1","Pred. 1")
matrix1.trn <- as.table(matrix1.trn)
matrix1.trn[1, 1] <- N11.trn.1
matrix1.trn[1, 2] <- N12.trn.1
matrix1.trn[2, 1] <- N21.trn.1
matrix1.trn[2, 2] <- N22.trn.1

erro1.trn <- sum(matrix1.trn[row(matrix1.trn) !=
                 col(matrix1.trn)])/sum(matrix1.trn)

matrix1.tst <-  matrix(0, 2, 2)
rownames(matrix1.tst) <- c("True -1","True  1")
colnames(matrix1.tst) <- c("Pred. -1","Pred. 1")
```

```
matrix1.tst <- as.table(matrix1.tst)
matrix1.tst[1, 1] <- N11.tst.1
matrix1.tst[1, 2] <- N12.tst.1
matrix1.tst[2, 1] <- N21.tst.1
matrix1.tst[2, 2] <- N22.tst.1

erro1.tst <- sum(matrix1.tst[row(matrix1.tst) !=
                 col(matrix1.tst)])/sum(matrix1.tst)

sink("resultados (BoostWLDA)(N = 200).txt")

print("Matriz de Confusão:")
matrix1.trn
print("Taxa de Erro:")
erro1.trn
print("--------------------------")
print("--------------------------")
print("Matriz de Confusão:")
matrix1.tst
print("Taxa de Erro:")
erro1.tst

sink()
```

B.3 Fronteira de Classificação

```
rm(list=ls(all=TRUE))
library(ks)
library(lattice)

#####################################################################

flda <- function(xtrain, group)
# Implementa a função discriminante linear de Fisher para o caso de dois grupos
# xtrain = padrões de entrada para o conjunto de treinamento (objeto tipo matrix)
# group = rótulos dos padrões de entrada codificados como -1 ou +1
{
    train <- cbind(xtrain, group)
    x1 <- as.matrix(train[group == 1, 1 : (dim(train)[2] - 1)])
    x2 <- as.matrix(train[group == 2, 1 : (dim(train)[2] - 1)])
    mean_x1 <- apply(x1, 2, mean)
    mean_x2 <- apply(x2, 2, mean)
    S1 <- cov(x1)
    S2 <- cov(x2)
    Sc <- (((dim(x1)[1] - 1) * S1) + ((dim(x2)[1] - 1) * S2)) / (dim(x1)[1] + dim(x2)[1] - 2)
    l_hat <- t(mean_x1 - mean_x2) %*% solve(Sc)
    m_hat <- .5 * t(mean_x1 - mean_x2) %*% solve(Sc) %*% (mean_x1 + mean_x2)
    means <- rbind(mean_x1, mean_x2)
    rownames(means) <- list("group 1 ", "group 2 ")
    result <- list(means, S1, S2, Sc, l_hat, m_hat)
    names(result) <- list("means", "s1", "s2", "sc", "coef", "threshold")
    result
}

#####################################################################

predict.lda <- function(fit, newdata)
# Implementa um módulo de predição para a função discriminante linear de Fisher
# fit = objeto do tipo "flda" ou "wflda" (funções flda.R ou wflda.R)
# newdata = padrões cujas classes queremos predizer (objeto tipo matrix)
{
    x <- newdata
    n <- dim(x)[1]
    coef <- fit$coef
```

```
    threshold <- fit$threshold
    y_hat <- numeric(n)
    class <- numeric(n)
    for(t in 1 : n)
    {
        y_hat[t] <- coef %*% x[t,]
        if(y_hat[t] >= threshold)
            class[t] <- 1
        else
            class[t] <- 2
    }
    result <- list(class, y_hat)
    names(result) <- list("class", "y0")
    result
}

################################################################

B <- 400         ## Números de amostras Bootstrap
N <- 50          ## Tamanho de amostra para o conjunto de treinamento
n <- 50          ## Tamanho da amostra para o conjunto de teste
alpha <- 0.05    ## Nível nominal de significância

#########################################

## Geração dos conjunto de treinamento e de teste e das variáveis indicadoras das populações

#########################################

y.trn <- sort(rep(c(1, 2), N/2))
y.tst <- sort(rep(c(1, 2), n/2))

mus1.2 <- c(-1, 1/2)
mus2.2 <- c(1, -1/2)

sigmas1.2 <- matrix(c(2/3, 1/5, 1/5, 4/9), 2, 2)
sigmas2.2 <- matrix(c(2/3, 1/5, 1/5, 4/9), 2, 2)

set.seed(321)
x.tst <- rbind(rmvnorm.mixt(n/2, mus1.2, sigmas1.2, 1), rmvnorm.mixt(n/2, mus2.2, sigmas2.2, 1))
x.trn <- rbind(rmvnorm.mixt(N/2, mus1.2, sigmas1.2, 1), rmvnorm.mixt(N/2, mus2.2, sigmas2.2, 1))

#########################################
## Programa Principal

fit0 <- flda(x.trn, y.trn)
pred0 <- predict.lda(fit0, x.tst)
yhat0 <- pred0$y0

mb <- numeric(B)

## Início do laço de Bootstrap
for (b in 1 : B) {
    ind.b <- sample(1 : N, N, replace = T)
    x.trn.b <- x.trn[ind.b, ]
    y.trn.b <- y.trn[ind.b]
    fit1 <- flda(x.trn.b, y.trn.b)
    mb[b] <- fit1$threshold
}

sorted.mb <- sort(mb)
LI <- sorted.mb[alpha*B]
LS <- sorted.mb[(1-alpha)*B]

pop <- numeric(n)
```

```
for (i in 1 : n) {
    if (yhat0[i] >= LS)
        pop[i] <- 1
    if (yhat0[i] <= LI)
        pop[i] <- 2
    if (yhat0[i] < LS && yhat0[i] > LI)
        pop[i] <- 3
}

## Comparando as classes veradeiras com as preditas pelos 2 métodos
cbind(1:n,y.tst,pred0$class,pop)

#####################################################################

f <- pda.pde(x.trn, y.trn, type = "line")
plot(f, x.tst, y.tst, col = c("white", "white"), cont = 0, pch = c(16, 4), ptcol = c("black", "black"),
     xlab = expression(X[1]), ylab = expression(X[2]))
identify(x.tst, labels = 1:n)
legend("topright", legend = c(expression(Pi[1]), expression(Pi[2])), pch = c(16, 4), bty = "n")
```

APÊNDICE C

Aplicações

Neste capítulo apresentamos as rotinas em linguagem R utilizadas para obtenção dos resultados das aplicações a dados reais apresentadas nesta dissertação.

C.1 Treinamento-e-Teste

```
##############################################
#                                            #
# Exemplo com o conjunto de dados do salmão #
# utilizado no capítulo 3                    #
#                                            #
##############################################

library(ks)
library(MASS)

data <- read.table("salmon.dat")
salmon <- data[, 2:3]
salmon.gr <- data[, 1]

salmon.lda <- pda(salmon, salmon.gr, salmon, type = "line")
salmon.qda <- pda(salmon, salmon.gr, salmon, type = "quad")
H <- Hkda(salmon, salmon.gr)
salmon.kda <- kda(salmon, salmon.gr, H, salmon)

compare(salmon.lda, salmon.gr)
compare(salmon.qda, salmon.gr)
compare(salmon.kda, salmon.gr)

f.lda <- pda.pde(salmon, salmon.gr, type = "line")
f.qda <- pda.pde(salmon, salmon.gr, type = "quad")
f.kda <- kda.kde(salmon, salmon.gr, H)

plot(f.kda, col = c("darkgray", "gray85"), cont = 0, pch = c(16,4),
     ptcol = c("black", "black"), xlab = expression(X[1]), ylab = expression(X[2]))
legend("topright", legend = c("Alasca", "Canadá"), pch = c(16, 4), bty = "n")
x11()
plot(f.lda, col = c("darkgray", "gray85"), cont = 0, pch = c(16,4),
```

```
     ptcol = c("black", "black"), xlab = expression(X[1]), ylab = expression(X[2]))
legend("topright", legend = c("Alasca", "Canadá"), pch = c(16, 4), bty = "n")
x11()
plot(f.qda, col = c("darkgray", "gray85"), cont = 0, pch = c(16,4),
     ptcol = c("black", "black"), xlab = expression(X[1]), ylab = expression(X[2]))
legend("topright", legend = c("Alasca", "Canadá"), pch = c(16, 4), bty = "n")
x11()
plot(f.lda, col = c("white", "white"), cont = 0, pch = c(16,4),
     ptcol = c("black", "black"), xlab = expression(X[1]), ylab = expression(X[2]))
legend("topright", legend = c("Alasca", "Canadá"), pch = c(16, 4), bty = "n")

#######################################
#                                     #
# Rotina utilizada para aplicação dos #
# método aos dados do salmão.         #
#                                     #
#######################################

library(MASS)
library(ks)
library(caTools)

###############################################
############  TREINAMENTO-E-TESTE  ############
###############################################

############## SALMAO ##############

salmao <- read.table("salmon.dat")
X <- salmao[, 2:3]
Y <- salmao[, 1]

msk <- sample.split(Y, 1/2)
x.trn <- X[msk,]
x.tst <- X[!msk,]
y.trn <- Y[msk]
y.tst <- Y[!msk]

salmon.lda <- pda(x.trn, y.trn, x.tst, type = "line")
salmon.qda <- pda(x.trn, y.trn, x.tst, type = "quad")
H <- Hkda(x.trn, y.trn)
salmon.kda <- kda(x.trn, y.trn, H, x.tst)

compare(salmon.lda, y.tst)
compare(salmon.qda, y.tst)
compare(salmon.kda, y.tst)

f.lda <- pda.pde(x.trn, y.trn, type = "line")
f.qda <- pda.pde(x.trn, y.trn, type = "quad")
f.kda <- kda.kde(x.trn, y.trn, H)

plot(f.lda, x.tst, y.tst, col = c("darkgray", "gray85"), cont = 0,
     pch = c(16,4), lty = c(1, 3), ptcol = c("black", "black"),
     xlab = expression(X[1]), ylab = expression(X[2]))
legend("topright", legend = c("Alasca", "Canadá"), pch = c(16, 4), bty = "n")
x11()
plot(f.qda, x.tst, y.tst, col = c("darkgray", "gray85"), cont = 0,
     pch = c(16,4), lty = c(1, 3), ptcol = c("black", "black"),
     xlab = expression(X[1]), ylab = expression(X[2]))
legend("topright", legend = c("Alasca", "Canadá"), pch = c(16, 4), bty = "n")
x11()
plot(f.kda, x.tst, y.tst, col = c("darkgray", "gray85"), cont = 0,
     pch = c(16,4), lty = c(1, 3), ptcol = c("black", "black"),
     xlab = expression(X[1]), ylab = expression(X[2]))
legend("topright", legend = c("Alasca", "Canadá"), pch = c(16, 4), bty = "n")
```

C.2 Bootstrap

```
################################################
# Função que calcula o erro de predição de uma #
# função discriminante linear, através do      #
# estimador bootstrap .632.                    #
# Esta função é baseada na função bootpred do  #
# pacote bootstrap.                            #
################################################

boot.632.lda <- function(x, y, nboot) {
    x <- as.matrix(x)
    n <- length(y)
    saveii <- NULL
    priori <- as.vector(table(y) / n)
    yhat0 <- pda(x, y, x, priori, type = "line")
    err.meas <- function(y, yhat) {1 * (yhat != y)}
    app.err <- mean(err.meas(y, yhat0))
    err1 <- matrix(0, nrow = nboot, ncol = n)
    err2 <- numeric(nboot)
    for (b in 1 : nboot) {
        ii <- sample(1 : n, replace = TRUE)
        saveii <- cbind(saveii, ii)
        yhat1 <- pda(x[ii, ], y[ii], x[ii, ], priori, type = "line")
        yhat2 <- pda(x[ii, ], y[ii], x, priori, type = "line")
        err1[b, ] <- err.meas(y, yhat2)
        err2[b] <- mean(err.meas(y[ii], yhat1))
    }
    optim <- mean(apply(err1, 1, mean) - err2)
    junk <- function(x, i) {sum(x == i)}
    e0 <- 0
    for (i in 1 : n) {
        o <- apply(saveii, 2, junk, i)
        if (sum(o == 0) == 0)
            cat("increase nboot for computation of the .632 estimator",
                fill = TRUE)
        e0 <- e0 + (1 / n) * (sum(err1[o == 0, i]) / sum(o == 0))
    }
    err.632 <- 0.368 * app.err + 0.632 * e0
    result <- list(app.err, optim, err.632)
    names(result) <- list("app.err", "optim", "err.632")
    return(result)
}

################################################
# Função que calcula o erro de predição    #
# de uma função discriminante quadrática,  #
# através do estimador bootstrap .632.     #
# Esta função é baseada na função bootpred #
# do pacote bootstrap.                     #
################################################

boot.632.qda <- function(x, y, nboot) {
    x <- as.matrix(x)
    n <- length(y)
    saveii <- NULL
    priori <- as.vector(table(y) / n)
    yhat0 <- pda(x, y, x, priori, type = "quad")
    err.meas <- function(y, yhat) {1 * (yhat != y)}
    app.err <- mean(err.meas(y, yhat0))
    err1 <- matrix(0, nrow = nboot, ncol = n)
    err2 <- numeric(nboot)
    for (b in 1 : nboot) {
        ii <- sample(1 : n, replace = TRUE)
        saveii <- cbind(saveii, ii)
```

```
        yhat1 <- pda(x[ii, ], y[ii], x[ii, ], priori, type = "quad")
        yhat2 <- pda(x[ii, ], y[ii], x, priori, type = "quad")
        err1[b, ] <- err.meas(y, yhat2)
        err2[b] <- mean(err.meas(y[ii], yhat1))
    }
    optim <- mean(apply(err1, 1, mean) - err2)
    junk <- function(x, i) {sum(x == i)}
    e0 <- 0
    for (i in 1 : n) {
        o <- apply(saveii, 2, junk, i)
        if (sum(o == 0) == 0)
            cat("increase nboot for computation of the .632 estimator",
                fill = TRUE)
        e0 <- e0 + (1 / n) * (sum(err1[o == 0, i]) / sum(o == 0))
    }
    err.632 <- 0.368 * app.err + 0.632 * e0
    result <- list(app.err, optim, err.632)
    names(result) <- list("app.err", "optim", "err.632")
    return(result)
}


################################################
# Função que calcula o erro de predição de um #
# classificador baseado em funções núcleo,    #
# através do estimador bootstrap .632. Esta   #
# função é baseada na função bootpred do      #
# pacote bootstrap.                           #
################################################

boot.632.kda <- function(x, y, nboot) {
    x <- as.matrix(x)
    n <- length(y)
    saveii <- NULL
    priori <- as.vector(table(y) / n)
    H0 <- Hkda(x, y)
    yhat0 <- kda(x, y, H0, x, priori)
    err.meas <- function(y, yhat) {1 * (yhat != y)}
    app.err <- mean(err.meas(y, yhat0))
    err1 <- matrix(0, nrow = nboot, ncol = n)
    err2 <- numeric(nboot)
    for (b in 1 : nboot) {
        ii <- sample(1 : n, replace = TRUE)
        saveii <- cbind(saveii, ii)
        Hb <- Hkda(x[ii, ], y[ii])
        yhat1 <- kda(x[ii, ], y[ii], Hb, x[ii, ], priori)
        yhat2 <- kda(x[ii, ], y[ii], Hb, x, priori)
        err1[b, ] <- err.meas(y, yhat2)
        err2[b] <- mean(err.meas(y[ii], yhat1))
    }
    optim <- mean(apply(err1, 1, mean) - err2)
    junk <- function(x, i) {sum(x == i)}
    e0 <- 0
    for (i in 1 : n) {
        o <- apply(saveii, 2, junk, i)
        if (sum(o == 0) == 0)
            cat("increase nboot for computation of the .632 estimator",
                fill = TRUE)
        e0 <- e0 + (1 / n) * (sum(err1[o == 0, i]) / sum(o == 0))
    }
    err.632 <- 0.368 * app.err + 0.632 * e0
    result <- list(app.err, optim, err.632)
    names(result) <- list("app.err", "optim", "err.632")
    return(result)
}
```

```
####################################
#                                  #
# Função utilizada para calculo do #
# utilizando o estimador .632      #
#                                  #
####################################

library(MASS)
library(ks)

########################################
###########  BOOTSTRAP .632  ###########
########################################

source("bootstrap .632 (lda).R")
source("bootstrap .632 (qda).R")
source("bootstrap .632 (kda).R")

B <- 500

##############  SALMÃO  ##############

salmao <- read.table("salmon.dat")
Y <- salmao[, 1]
X <- salmao[, 2:3]
set.seed(321)
salmao.err.lda <- boot.632.lda(X, Y, B)
set.seed(321)
salmao.err.qda <- boot.632.qda(X, Y, B)
set.seed(321)
salmao.err.kda <- boot.632.kda(X, Y, B)

##############  HABERMAN  ##############

haberman <- read.table("haberman.dat", sep = ",")
Y <- haberman[, 4]
X <- haberman[, 1:3]
set.seed(321)
haberman.err.lda <- boot.632.lda(X, Y, B)
set.seed(321)
haberman.err.qda <- boot.632.qda(X, Y, B)
set.seed(321)
haberman.err.kda <- boot.632.kda(X, Y, B)

##########  IMPRESSÃO DOS RESULTADOS  ##########

sink("resultados-bootstrap.txt")
cat("\nSALMÃO\n")
cat("\nAnálise Discriminante Linear\n")
salmao.err.lda
cat("\nAnálise Discriminante Quadrática\n")
salmao.err.qda
cat("\nAnálise Discriminante de Núcleo\n")
salmao.err.kda
cat("-----------------------------------")
cat("\nHABERMAN\n")
cat("\nAnálise Discriminante Linear\n")
haberman.err.lda
cat("\nAnálise Discriminante Quadrática\n")
haberman.err.qda
cat("\nAnálise Discriminante de Núcleo\n")
haberman.err.kda
cat("-----------------------------------")
sink()
```

APÊNDICE D

Geração dos Gráficos

Neste capítulo apresentamos as rotinas em linguagem R utilizadas para geração dos gráficos apresentados nesta dissertação.

```
####################################
#                                  #
# Geração dos gráficos dos cenários #
# apresentados no capítulo 5.      #
#                                  #
####################################

rm(list=ls(all=TRUE))
library(ks)

x <- y <- seq(-5, 5, length = 1000)

normal.bi <- function(x, y, mx, my, sx, sy, sxy) {
    x <- cbind(x, y)
    mus <- c(mx, my)
    sigmas <- matrix(c(sx, sxy, sxy, sy), 2, 2)
    z <- dmvnorm(x, mus, sigmas)
}

normal.bi.mixt <- function(x, y, mx1, my1, sx1, sy1, sxy1,
                           mx2, my2, sx2, sy2, sxy2, props) {
    x <- cbind(x, y)
    mus <- rbind(c(mx1, my1), c(mx2, my2))
    sigmas <- rbind(matrix(c(sx1, sxy1, sxy1, sy1), 2, 2),
                    matrix(c(sx2, sxy2, sxy2, sy2), 2, 2))
    z <- dmvnorm.mixt(x, mus, sigmas, props)
}

##### CENÁRIO 1 #####

z <- outer(x, y, normal.bi, mx = 1, my = -1, sx = 4/9,
           sy = 4/9, sxy = 14/45)
contour(x, y, z, nlevels = 15, drawlabels = FALSE,
        xlim = c(-3, 3), ylim = c(-3, 3), main = "Cenário A",
        xlab = expression(X[1]), ylab = expression(X[2]))
```

```
z <- outer(x, y, normal.bi, mx = -1, my = 1, sx = 4/9,
           sy = 4/9, sxy = 0)
contour(x, y, z, nlevels = 15, drawlabels = FALSE,
        lty = "dashed", xlim = c(-3, 3), ylim = c(-3, 3), add = T)

x11()
##### CENÁRIO 2 #####

z <- outer(x, y, normal.bi.mixt, mx1 = -3/2, my1 = -3/2,
           sx1 = 4/5, sy1 = 4/5, sxy1 = -1/2, mx2 = 1/2, my2 = 1/2,
           sx2 = 4/5, sy2 = 4/5, sxy2 = -1/2, props = c(1/2, 1/2))
contour(x, y, z, nlevels = 8, drawlabels = FALSE,
        xlim = c(-3, 3), ylim = c(-3, 3), main = "Cenário D",
        xlab = expression(X[1]), ylab = expression(X[2]))

z <- outer(x, y, normal.bi.mixt, mx1 = 3/2, my1 = 3/2,
           sx1 = 4/5, sy1 = 4/5, sxy1 = -1/2, mx2 = -1/2, my2 = -1/2,
           sx2 = 4/5, sy2 = 4/5, sxy2 = -1/2, props = c(1/2, 1/2))
contour(x, y, z, nlevels = 8, drawlabels = FALSE, lty = "dashed",
        xlim = c(-3, 3), ylim = c(-3, 3), add = T)

x11()
##### CENÁRIO 3 #####

z <- outer(x, y, normal.bi.mixt, mx1 = -3/2, my1 = 0,
           sx1 = 3/10, sy1 = 3/10, sxy1 = 1/4, mx2 = 3/2, my2 = 0,
           sx2 = 3/10, sy2 = 3/10, sxy2 = 1/4, props = c(1/2, 1/2))
contour(x, y, z, nlevels = 10, drawlabels = FALSE,
        xlim = c(-3, 3), ylim = c(-3, 3), main = "Cenário F",
        xlab = expression(X[1]), ylab = expression(X[2]))

z <- outer(x, y, normal.bi, mx = 0, my = 0, sx = 4/5, sy = 1, sxy = 2/5)
contour(x, y, z, nlevels = 10, drawlabels = FALSE, lty = "dashed",
        xlim = c(-3, 3), ylim = c(-3, 3), add = T)

x11()
##### CENÁRIO 4 #####

z <- outer(x, y, normal.bi, mx = -1, my = 1/2,
           sx = 2/3, sy = 4/9, sxy = 1/5)
contour(x, y, z, nlevels = 15, drawlabels = FALSE,
        xlim = c(-3, 3), ylim = c(-3, 3), main = "Cenário C",
        xlab = expression(X[1]), ylab = expression(X[2]))

z <- outer(x, y, normal.bi, mx = 1, my = -1/2,
           sx = 2/3, sy = 4/9, sxy = 1/5)
contour(x, y, z, nlevels = 15, drawlabels = FALSE, lty = "dashed",
        xlim = c(-3, 3), ylim = c(-3, 3), add = T)

x11()
##### CENÁRIO 5 #####

z <- outer(x, y, normal.bi, mx = -1, my = 1,
           sx = 2/3, sy = 1/3, sxy = 1/5)
contour(x, y, z, nlevels = 15, drawlabels = FALSE,
        xlim = c(-3, 3), ylim = c(-3, 3), main = "Cenário B",
        xlab = expression(X[1]), ylab = expression(X[2]))

z <- outer(x, y, normal.bi, mx = 1, my = -1/2,
           sx = 2/3, sy = 1/3, sxy = 1/5)
contour(x, y, z, nlevels = 15, drawlabels = FALSE, lty = "dashed",
        xlim = c(-3, 3), ylim = c(-3, 3), add = T)

x11()
```

```
##### CENÁRIO 6 #####

z <- outer(x, y, normal.bi.mixt, mx1 = -3/2, my1 = 0,
           sx1 = 1/12, sy1 = 1, sxy1 = 1/4,
           mx2 = 3/2, my2 = 0, sx2 = 1/12, sy2 = 1,
           sxy2 = 1/4, props = c(1/2, 1/2))
contour(x, y, z, nlevels = 10, drawlabels = FALSE,
        xlim = c(-3, 3), ylim = c(-3, 3), main = "Cenário E",
        xlab = expression(X[1]), ylab = expression(X[2]))

z <- outer(x, y, normal.bi, mx = 0, my = 0, sx = 4/9, sy = 4/9, sxy = 1/5)
contour(x, y, z, nlevels = 10, drawlabels = FALSE, lty = "dashed",
        xlim = c(-3, 3), ylim = c(-3, 3), add = T)
```

Referências Bibliográficas

Amaral, G. J. A., Dryden, I. L. & Wood, A. T. A. (2007), 'Pivotal bootstrap methods for k-sample problems in directional statistics and shape analysis', *Journal of the American Statistical Association* . Artigo aceito para publicação.

Anderson, T. W. (2003), *An introduction to multivariate statistical analysis*, 3rd. edn, Wiley-Interscience, New Jersey.

Bühlmann, P. & Yu, B. (2003), 'Boosting with the ℓ_2-loss: Regression and classification', *Journal of the American Statistical Association* **98**, 324–339.

Breiman, L. (1996), 'Bagging predictors', *Machine Learning* **24**, 123–140.

Breiman, L. (1998), 'Arcing classifiers', *The Annals of Statistics* **26**, 801–849.

Cribari Neto, F. & Zarkos, S. G. (1999), 'R: Yet another econometric programming environment', *Journal of Applied Econometrics* **14**, 319–329.

Davison, A. C. & Hinkley, D. V. (1997), *Bootstrap Methods and Their Application*, Cambridge University Press, New York.

Di Marzio, M. & Taylor, C. C. (2005*a*), 'Kernel density classification and boosting: an l_2 analysis', *Statistics and Computing* **15**, 113–123.

Di Marzio, M. & Taylor, C. C. (2005*b*), 'On boosting kernel density methods for multivariate data : density estimation and classification', *Statistical Methods & Applications* **14**, 163–178.

Drapper, N. R. & Smith, H. (1998), *Applied Regression Analysis*, 3rd. edn, John Wiley & Sons, New York.

Duong, T. (2004), Bandwidth selectors for multivariate kernel density estimation, PhD thesis, University of Western Australia, School of Mathematics and Statistics.

Efron, B. (1979), 'Bootstrap methods: another look at the jackknife', *Annals of Statistics* **7**, 1–26.

Efron, B. (1983), 'Estimating the error rate of a prediction rule: improvements on cross-validation', *Journal of the American Statistical Association* **78**, 316–331.

Efron, B. (1986), 'How biased is the apparent error rate of a prediction rule?', *Journal of the American Statistical Association* **81**, 461–470.

Efron, B. & Tibshirani, R. (1993), *An Introduction to the Bootstrap*, Chapman & Hall, New York.

Fisher, N. I. & Hall, P. (1990), 'On bootstrap hypothesis testing', *Australian Journal of Statistics* **32**, 177–190.

Fisher, N. I., Hall, P., Jing, B.-Y. & Wood, A. T. A. (1996), 'Improved pivotal methods for constructing confidence regions with directional data', *Journal of the American Statistical Association* **91**, 1062–1070.

Fisher, R. A. (1936), 'The use of multiple measurements in taxonomic problems', *Annals of Eugenics* **7**, 179–188.

Fix, E. & Hodges, J. L. (1951), Discriminatory analysis — nonparametric discrimination: consistency properties, Technical report, US Air Force School of Aviation Medicine, Randolph Field, Texas.

Freund, Y. (1995), 'Boosting a weak learning algorithm by majority', *Information and Computation* **121**, 256–285.

Freund, Y. & Schapire, R. E. (1996), Experiments with a new boosting algorithm, *in* 'Machine Learning: Proceedings of the Thirteenth International Conference', pp. 148–156.

Friedman, J., Hastie, T. & Tibshirani, R. (2000*a*), 'Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)', *The Annals of Statistics* **38**, 337–374.

Friedman, J., Hastie, T. & Tibshirani, R. (2000*b*), *The Elements of Statistical Learning*, Springer, New York.

Hall, P. (1992), *The Bootstrap and Edgeworth Expansion*, Springer-Verlag, New York.

Hall, P. & Wilson, S. R. (1991), 'Two guidelines for bootstrap hypothesis testing', *Biometrics* **47**, 757–762.

Johnson, R. A. & Wichern, D. W. (1998), *Applied multivariate statistical analysis*, 4th. edn, Prentice-Hall, New York.

Lachenbruch, P. A. (1975), *Discriminant Analysis*, Hafner Press, New York.

Lachenbruch, P. A. & Mickey, M. R. (1968), 'Estimation of error rates in discriminant analysis', *Technometrics* **10**, 1–11.

Lamport, L. (1994), *A Document Preparation System*, 2nd. edn, Addison-Wesley, Massachusetts.

Mardia, K. V., Kent, J. T. & Bibby, J. M. (1979), *Multivariate Analysis*, Academic Press, London.

Owen, A. B. (2001), *Empirical Likelihood*, Chapman & Hall/CRC, New York.

R Development Core Team (2006), *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0. *http://www.R-project.org

Rosenblatt, M. (1956), 'Remarks on some nonparametrics estimates of a density function', *The Annals of Mathematical Statistics.* **27**, 832–837.

Schapire, R. E. (1990), 'The strength of weak learnability', *Machine Learning* **5**, 197–227.

Schapire, R. E., Freund, Y., Bartlett, P. & Lee, W. S. (1998), 'Boosting the margin: A new explanation for the effectiveness of voting methods', *The Annals of Statistics* **26**, 1651–1686.

Scott, D. W. (1992), *Multivariate Density Estimation: Theory, Practice, and Visualization*, John Wiley & Sons, New York.

Silverman, B. W. (1986), *Density Estimation for Statistics and Data Analysis*, Chapman & Hall, London.

Simonoff, J. S. (1996), *Smoothing Methods in Statistics*, Springer-Verlag, New York.

Skurichina, M. & Duin, R. P. (2000), Boosting in linear discriminant analysis, *in* J. Kittler & F. Roli, eds, 'Multiple Classifier Systems: Proceedings of the First International Workshop', Cagliari, Italy, pp. 190–199.

Stone, M. (1974), 'Cross-validatory choice and assessment of statistical predictions', *Journal of the Royal Statistical Society* **36**, 111–147.

Stone, M. (1977), 'An asymptotic equivalence of choice of model by cross-validation and akaike's criterion', *Journal of the Royal Statistical Society* **39**, 44–47.

Wand, M. P. & Jones, M. C. (1993), 'Comparison of smoothing parametrizations in bivariate kernel density estimation', *Journal of the American Statistical Association* **88**, 520–528.

Wand, M. P. & Jones, M. C. (1995), *Kernel Smoothing*, Chapman & Hall, London.

Webb, A. (2002), *Statistical Pattern Recognition*, 2nd. edn, John Wiley & Sons, London.

Wedderburn, R. W. M. (1974), 'Quasi-likelihood function, generalized linear models and the gauss-newton method', *Biometrika* **61**, 439–447.

Livros Grátis

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

Milhares de Livros para Download:

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