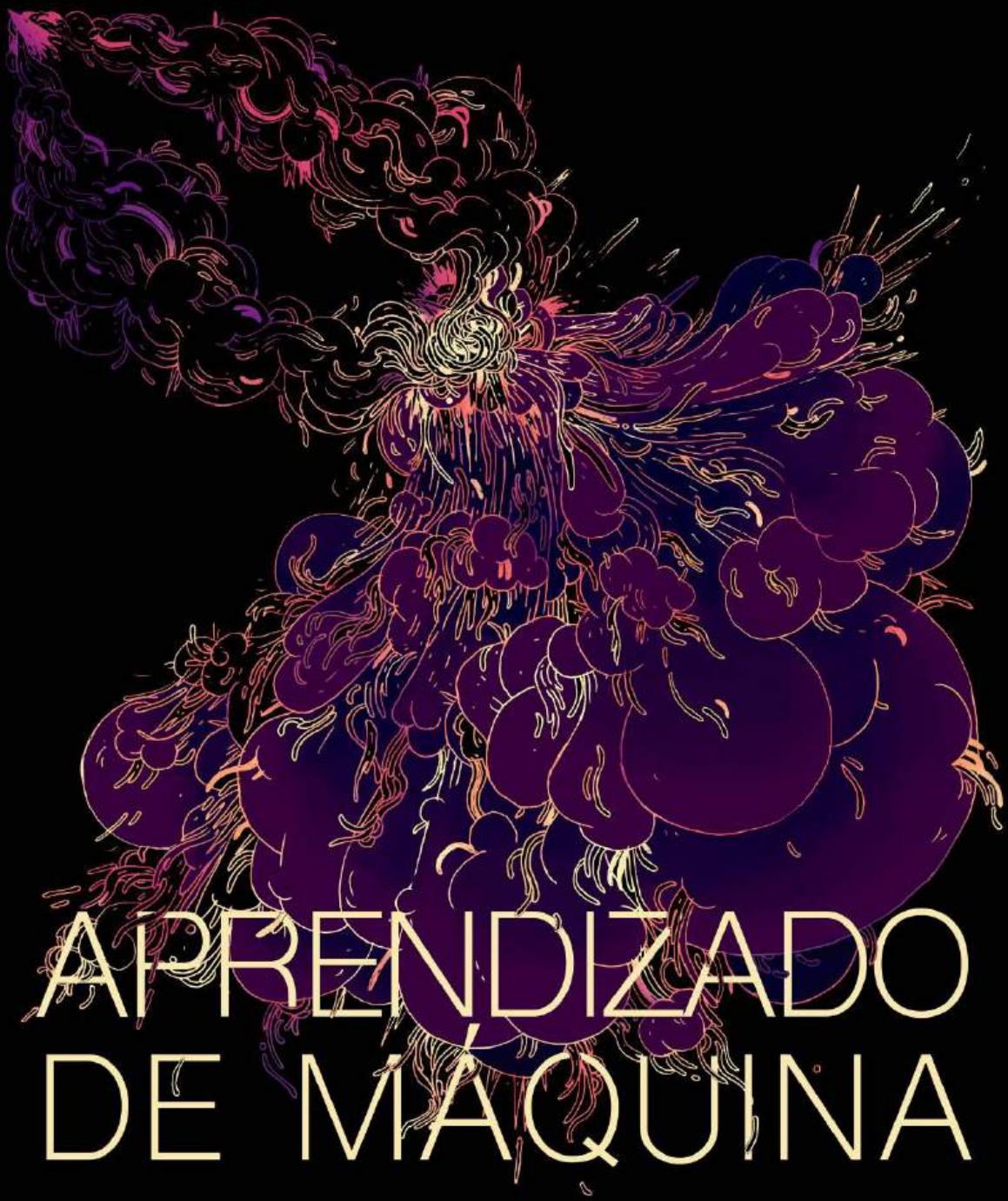


APRENDIZADO DE MÁQUINA

UMA ABORDAGEM ESTATÍSTICA

RAFAEL IZBICKI
TIAGO MENDONÇA DOS SANTOS

Aprendizado de máquina:
uma abordagem estatística

Título: Aprendizado de máquina: uma abordagem estatística, 1ª edição.

Autores: Rafael Izbicki e Tiago Mendonça dos Santos.

Capa: Leonardo M Borges & Kaori Nagata

Dados Internacionais de Catalogação na Publicação (CIP)
(Câmara Brasileira do Livro, SP, Brasil)

Izbicki, Rafael
Aprendizado de máquina : uma abordagem estatística [livro eletrônico] / Rafael Izbicki, Tiago Mendonça dos Santos. -- São Carlos, SP : Rafael Izbicki, 2020.
PDF

Bibliografia.
ISBN 978-65-00-02410-4

1. Aprendizado do computador 2. Estatística 3. Inteligência artificial 4. Probabilidades I. Santos, Tiago Mendonça dos. II. Título.

20-36124 CDD-519.5

Índices para catálogo sistemático:

1. Aprendizado de máquina : Estatística : Matemática 519.5

Cibele Maria Dias - Bibliotecária - CRB-8/9427

Rafael Izbicki

&

Tiago Mendonça dos Santos

Aprendizado de máquina: uma abordagem estatística

Aos nossos avós

Ita Zajdens
Leon Izbicki
Rebecca Gassul Izbicki

Hely Mendonça Malheiro
Therezinha Giacomin Malheiro
Maria Anita dos Santos

Sumário

.

.

Prefácio

Aprendizado de máquina (AM) nasceu na década de 60 como um campo da inteligência artificial que tinha o objetivo de aprender padrões com base em dados. Originalmente, as aplicações de AM eram de cunho estritamente computacional. Contudo, desde o final dos anos 90, essa área expandiu seus horizontes e começou a se estabelecer como um campo por si mesma. Em particular, as aplicações de AM começaram a ter muitas intersecções com as de estatística.

A comunidade de AM é bastante interdisciplinar e utiliza ideias desenvolvidas em diversas áreas, sendo a estatística uma delas. Enquanto que até os anos 90 métodos criados pela comunidade estatística rapidamente começavam a ser incorporados em AM, mais recentemente o fortalecimento de AM fez com que a direção oposta começasse a ficar cada vez mais comum: métodos desenvolvidos por AM começaram a ser usados em estatística.

O objetivo deste livro é introduzir ideias de AM sob uma ótica estatística, quebrando barreiras entre essas duas áreas. Pretendemos, assim, contribuir para diminuir alguns preconceitos existentes, de modo que o melhor de cada campo possa ser aproveitado.

Este livro. Este livro é fruto de diversos cursos que demos ao longo dos últimos anos. Ele surgiu no formato de notas de aulas, que foram criadas em função da necessidade que vimos em ter um material que fizesse a ponte entre estatística e AM. Essas notas de aula foram feitas tanto para cursos de graduação quanto para cursos de pós-graduação. Assim, há capítulos mais práticos e capítulos com mais teoria matemática. Nosso objetivo é construir uma base para que o leitor consiga entender os paradigmas mais importantes desta área. O propósito deste livro não é fornecer todos os detalhes e variações que existem sobre os métodos estudados, mas sim introduzir a essência de cada um deles.

.

Inicialmente, estudamos o problema de aprendizado supervisionado, que consiste em aprender a fazer predições a partir de um conjunto de dados em que os rótulos (ou seja, os valores da variável resposta) são observados. Tratamos tanto do problema de regressão (Parte I do livro) quanto do problema de classificação (Parte II do livro). Em seguida, estudamos o problema de aprendizado não supervisionado (Parte III), que consiste em aprender mais sobre a estrutura dos dados na ausência de rótulos. As técnicas aqui estudadas englobam métodos de transformação de variáveis e redução de dimensionalidade, análise de agrupamento, análise de associação e sistemas de recomendação. No Apêndice (Parte IV), fazemos uma breve introdução sobre como manipular imagens e textos, além de como armazenar matrizes esparsas eficientemente.

Recomendamos que a leitura do Capítulo 1 seja feita antes dos demais capítulos, pois ele apresenta a nomenclatura e principais paradigmas abordados no restante do livro. De forma geral, o entendimento da Parte II (métodos de classificação) é bastante facilitado pela leitura da Parte I (métodos de regressão).

Requisitos. Assumimos que o leitor é familiarizado com fundamentos de probabilidade e inferência estatística. Uma ótima introdução ao tema no nível desejado é feita por DeGroot e Schervish (2012).

Códigos em R. Ao longo do livro mostramos códigos em R que podem ser utilizados para aplicar os métodos apresentados. De modo algum as funções aqui apresentadas constituem uma lista exaustiva com a melhor maneira de fazer essas análises. Isso seria impossível: a comunidade R é muito dinâmica, sendo criados novos pacotes todos os dias. As funções utilizadas servem de guia para iniciar o leitor em cada método e permitir que ele reproduza os exemplos apresentados.

Teoria. Neste livro, incluímos alguns resultados teóricos sobre os problemas e métodos investigados. A teoria é uma ferramenta poderosa para se investigar um problema. Quando bem feita, ela é capaz de trazer insights diferentes sobre, por exemplo, quando um método deve funcionar bem e até mesmo sobre como melhorá-lo. Desta forma, a teoria não é um fim, mas sim uma forma diferente de se estudar um problema. Esta perspectiva se adiciona a, por exemplo, aplicações e estudos de simulação, para fornecer um olhar mais amplo sobre cada método. As seções de teoria muitas vezes requerem uma matemática um pouco mais avançada para permitir seu amplo entendimento. Contudo, o leitor que deseja ter apenas um contato inicial

com AM pode pular esses trechos sem grandes prejuízos para o entendimento do restante do conteúdo apresentado.

Bancos de dados. Em http://www.rizbicki.ufscar.br/ame disponibilizamos material suplementar que pode ser utilizado pelo leitor, incluindo os bancos de dados usados nos exemplos deste livro. Atualizações do livro também estarão disponíveis nesta página.

Agradecimentos. Muitos alunos, orientandos e colegas fizeram sugestões que contribuíram para melhorar a qualidade deste trabalho Agradecemos profundamente as sugestões de Afonso Fernandes Vaz, Afrânio M. C. Vieira, Ana Paula Jorge do Espirito Santo, Danilo Tadeschi, Davi Keglevich Neiva, Deborah Bassi Stern, Gilson Shimizu, João Marcos Alves Matos, Lucas Leite Cavalaro, Lucas Pereira Lopes, Luiz Gabriel Fernandes Cotrim, João Carlos Poloniato Ferreira, João Flávio Andrade Silva, Juliana Maia, Julio M. Stern, Luciana Morita Ishihara, Luís Ernesto Salasar, Marcela Musetti, Marcia Maria Barbosa da Silva, Marco Henrique de Almeida Inacio, Maurício Najjar da Silveira, Michel Helcias Montoril, Paula Ianishi, Rafael Bassi Stern, Sarah Izbicki, Taís Roberta Ribeiro, Victor Cândido Reis e Victor Vinicius Fernandes. Um agradecimento especial vai para Leonardo M. Borges e Kaori Nagata, que fizeram uma capa incrível e deram valiosas sugestões que deixaram o livro mais bem formatado e agradável de ser lido, assim como Lea Veras, que deu todo apoio necessário para conseguirmos concluir esse trabalho. Finalmente, agradecemos à CAPES, CNPq, FAPESP, Insper e UFSCar por tornarem esse trabalho possível.

Erros. Ainda que esta versão do livro tenha sido bastante revisada, ela certamente ainda tem diversos erros. Caso você encontre erros (sejam eles de português ou de matemática) ou tenha outras sugestões, por favor envie um email para rafaelizbicki (em) gmail.com.

.

Parte I

Regressão

Capítulo 1

Introdução

> Your assumptions are your windows on the world. Scrub them off every once in a while, or the light won't come in.
>
> Isaac Asimov

Métodos de regressão surgiram há mais de dois séculos com Legendre (1805) e Gauss (1809), que exploraram o método dos mínimos quadrados com o objetivo de prever órbitas ao redor do Sol. Hoje em dia, o problema de estimação de uma função de regressão possui papel central em estatística.

Apesar de as primeiras técnicas para solucionar esse problema datarem de ao menos 200 anos, os avanços computacionais recentes permitiram que novas metodologias fossem exploradas. Em particular, com a capacidade cada vez maior de armazenamento de dados, métodos com menos suposições sobre o verdadeiro estado da natureza ganham cada vez mais espaço. Com isso, vários desafios surgiram: por exemplo, métodos tradicionais não são capazes de lidar de forma satisfatória com bancos de dados em que há mais covariáveis que observações, uma situação muito comum nos dias de hoje. Similarmente, são frequentes as aplicações em que cada observação consiste em uma imagem ou um documento de texto, objetos complexos que levam a análises que requerem metodologias mais elaboradas. Aqui apresentamos diversos avanços recentes na área de regressão sob uma ótica preditivista, assim como uma revisão de modelos tradicionais sob esta mesma perspectiva.

De modo geral, o objetivo de um modelo de regressão é determinar a relação

entre uma variável aleatória $Y \in \mathbb{R}$ e um vetor $\mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$. Mais especificamente, busca-se estimar a função de regressão

$$r(\mathbf{x}) := \mathbb{E}[Y|\mathbf{X} = \mathbf{x}]$$

como forma de descrever tal relação. Quando Y é uma variável quantitativa, temos um problema de *regressão* (Parte I do livro). Para as situações em que Y é qualitativa temos um problema de *classificação* (Parte II desse livro).

1.1 Notação e Suposições

> Independence is happiness.
>
> Susan B. Anthony

A variável Y frequentemente recebe o nome de variável resposta, variável dependente ou rótulo (*label* em inglês). Já as observações contidas em $\mathbf{x} = (x_1, \ldots, x_d)$ são, em geral, chamadas de variáveis explicativas, variáveis independentes, características, atributos (*features* em inglês), preditores ou covariáveis. O objetivo da Parte I deste livro é descrever algumas técnicas para estimar (ou *treinar*, como dito na literatura de aprendizado de máquina) $r(\mathbf{x})$.

Vamos assumir em todo o livro, a menos que seja dito o contrário, que a estimação será feita com base em uma amostra de observações i.i.d. (independentes e identicamente distribuídas) $(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n) \sim (\mathbf{X}, Y)$. A Seção 9.3 lida com algumas situações em que essa suposição não é razoável. Denotamos por $x_{i,j}$ o valor da j-ésima covariável na i-ésima amostra, ou seja, $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,d})$, $i = 1, \ldots, n$ (veja a notação utilizada na Tabela 1.1).

Tabela 1.1: Notação utilizada no livro para as variáveis envolvidas em um problema de regressão.

Resposta	Covariáveis			
Y_1	$X_{1,1}$	$\ldots$	$X_{1,d}$	$(= \mathbf{X}_1)$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	
Y_n	$X_{n,1}$	$\ldots$	$X_{n,d}$	$(= \mathbf{X}_n)$

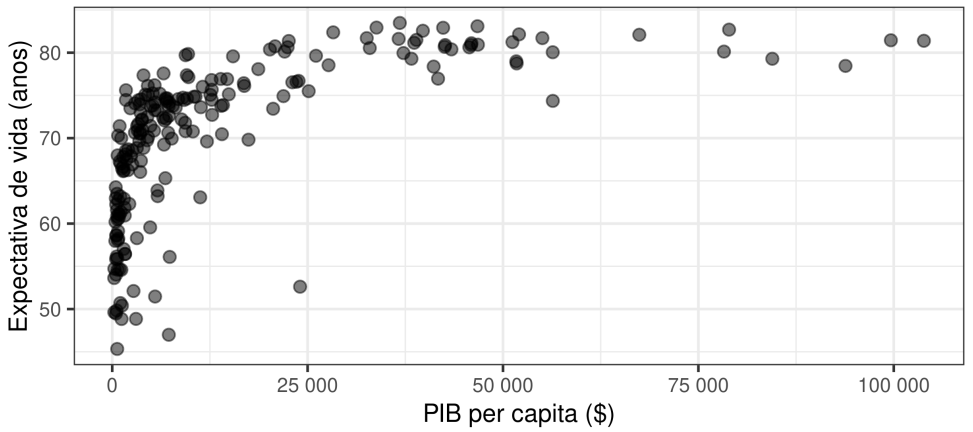


Figura 1.1: PIB per capita e expectativa de vida de 211 países.

1.2 Predição versus Inferência: Por que estimar $r(\mathbf{x})$?

> Either you run from it (the past) or you learn from it!
>
> Rafiki, The Lion King

A seguir veremos alguns exemplos reais nos quais a estimação de $r(\mathbf{x})$ possui papel central.

Exemplo 1.1. [Expectativa de vida e PIB per Capita] A Figura 1.1 mostra o PIB per Capita e Expectativa de vida em 211 países em 2012[1]. Uma pergunta de interesse é como podemos usar esses dados para estabelecer uma relação entre essas duas variáveis. Esta relação pode ser utilizada para estimar a expectativa de vida de países cujo PIB per Capita é conhecido, mas a esperança não o é. Para tanto, pode-se estimar $\mathbb{E}[Y|\mathbf{x}]$, em que Y é a expectativa de vida em um dado país e $\mathbf{x}$ é seu PIB per Capita. Uma estimativa de $\mathbb{E}[Y|\mathbf{x}]$ também pode ser utilizada para testar a hipótese de que não há relação entre essas duas variáveis.

□

[1] Dados obtidos em http://data.worldbank.org/

Exemplo 1.2. [Distância de uma galáxia até a Terra] Em cosmologia, uma variável de extrema importância é o *desvio para o vermelho* (*redshift*, em inglês) de uma galáxia, que quantifica o quão longe este objeto encontra-se da Terra. A *espectroscopia* permite que o desvio para o vermelho seja determinado com grande acurácia, contudo ela é extremamente cara e demanda muito tempo para ser feita. Assim, um grande interesse na atualidade é como estimar essa quantidade utilizando-se apenas imagens das galáxias ao invés da espectroscopia (Dalmasso et al., 2020; Schmidt et al., 2020) (veja a Figura 1.2 para um exemplo). Para tanto, pode-se usar uma amostra $(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)$, em que $\mathbf{X}_i$ é a imagem da i-ésima galáxia e Y_i seu respectivo desvio para o vermelho obtido via espectroscopia, para estimar a função de regressão $r(\mathbf{x})$. A partir dessa função é possível predizer as respostas em imagens de novas galáxias cujas distâncias até a Terra são desconhecidas. Veja o Apêndice A.1 para uma breve exposição sobre como trabalhar com imagens.

□

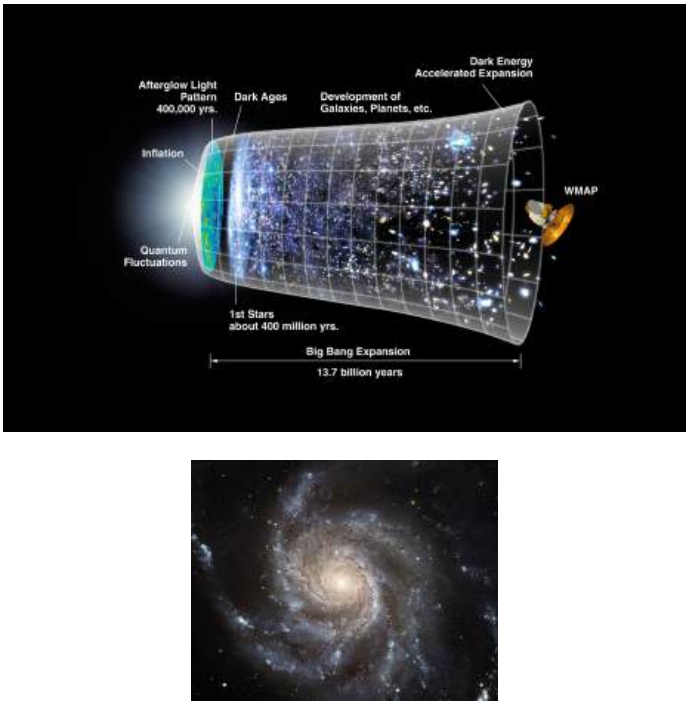


Figura 1.2: Acima: representação gráfica do universo e sua expansão. Abaixo: imagem de uma galáxia (crédito das imagens: ESA/Hubble).

Exemplo 1.3. [Isomap face data] Neste conjunto de dados, proveniente de Tenenbaum et al. (2000), o objetivo é estimar a direção para a qual uma pessoa está olhando (a resposta y) com base em uma imagem desta pessoa (covariáveis $\mathbf{x}$). Uma forma de se fazer isso é estimando a função de regressão $r(\mathbf{x})$ com base em uma amostra. Uma vez ajustada, $r(\mathbf{x})$ pode ser utilizada para predizer a direção em novas imagens definidas por covariáveis $\mathbf{x}$.

Figura 1.3: *Isomap face data*: cada observação consiste na imagem de um indivíduo que compõe um objeto $\mathbf{x}$ juntamente com a direção para a qual ele está olhando (y).

□

Exemplo 1.4. [Predição do ano de lançamento de músicas] Os dados do YearPredictionMSD[2] contém informações de diversas covariáveis sobre certas músicas do banco *Million Song Dataset* (por exemplo, informações sobre timbre, o "quão energética" é cada uma das músicas, o "quão dançáveis" elas são etc), assim como o ano de lançamento de cada uma delas. Com esse banco de dados, é possível utilizar uma estimativa de $r(\mathbf{x})$ (em que $\mathbf{x}$ são as covariáveis medidas e Y é o ano de lançamento de uma dada música) para (i) predizer o ano de lançamento de músicas com base apenas nas covariáveis e (ii) entender quais covariáveis estão relacionadas ao ano de lançamento e como se dá essa relação (por exemplo, nos anos 70 as músicas eram mais "dançáveis"?).

□

Os objetivos desses exemplos, resumidamente, podem ser divididos em duas classes:

[2] https://archive.ics.uci.edu/ml/datasets/YearPredictionMSD.

- **Objetivo inferencial**: Quais preditores são importantes? Qual a relação entre cada preditor e a variável resposta? Qual o efeito da mudança de valor de um dos preditores na variável resposta?

- **Objetivo preditivo**: Como podemos criar uma função

$$g : \mathbb{R}^d \longrightarrow \mathbb{R}$$

que tenha bom poder preditivo? Isto é, como criar g tal que, dadas *novas* observações i.i.d. $(\mathbf{X}_{n+1}, Y_{n+1}), \ldots, (\mathbf{X}_{n+m}, Y_{n+m})$, tenhamos

$$g(\mathbf{x}_{n+1}) \approx y_{n+1}, \ldots, g(\mathbf{x}_{n+m}) \approx y_{n+m}?$$

Enquanto em alguns dos problemas o objetivo é claramente inferencial ou preditivo, em outros temos uma mistura de ambos. Por exemplo, no Exemplo 1.3 temos um problema claramente preditivo: a relação entre cada pixel da imagem e a resposta em geral não é de interesse imediato. Já no Exemplo 1.4 temos um objetivo misto: desejamos criar uma função para descobrir o ano de lançamento de músicas com base em suas covariáveis e ao mesmo tempo também queremos entender como o perfil de músicas mudou ao longo dos anos.

1.3 As Duas Culturas

> We never look beyond our assumptions and what's worse, we have given up trying to meet others; we just meet ourselves.
>
> Muriel Barbery

Breiman (2001a) argumenta que existem duas culturas no uso de modelos estatísticos (em especial modelos de regressão). Grosso modo, a primeira cultura, chamada de *data modeling culture* por Breiman, é a que domina a comunidade estatística. Em geral, nela se assume que o modelo utilizado para $r(\mathbf{x})$ – por exemplo, $r(\mathbf{x}) = \beta_0 + \sum_{i=1}^{d} \beta_i x_i$ – é correto. Isso ocorre pois o principal objetivo está na interpretação dos parâmetros envolvidos no modelo (nesse caso, β_i's); em particular há interesse em testes de hipóteses e intervalos de confiança para esses parâmetros. Sob

essa abordagem, testar se as suposições do modelo (por exemplo, normalidade dos erros, linearidade, homocedasticidade etc) são válidas é de fundamental importância. Ainda que predição muitas vezes faça parte dos objetivos, o foco em geral está na inferência.

A segunda cultura, chamada de *algorithmic modeling culture* por Breiman, é a que domina a comunidade de aprendizado de máquina (*machine learning*). Neste meio, o principal objetivo é a predição de novas observações. Não se assume que o modelo utilizado para os dados é correto; o modelo é utilizado apenas para criar bons algoritmos para prever bem novas observações. Muitas vezes não há nenhum modelo probabilístico explícito por trás dos algoritmos utilizados.

Observação 1.1. Mesmo que o objetivo primordial de uma problema de predição seja obter um bom poder preditivo, a interpretabilidade do modelo final também é importante por diversas razões. Na Seção 6.1 discutimos algumas dessas razões, assim como formas de interpretar as predições dadas por modelos complexos.

□

O foco deste livro está na segunda abordagem. Buscaremos estimar $r(\mathbf{x})$ sem assumir que os modelos contenham a verdadeira função de regressão.

Apesar dessa divisão de culturas, Breiman foi um estatístico que fez um grande trabalho para unir a área de estatística com aprendizado de máquina. Devido à grande importância desse estatístico nessa tarefa, um prêmio concedido em sua homenagem foi criado pela *American Statistical Association*. Essa união entre as áreas é mutuamente benéfica e, portanto, esse livro visa unir as duas culturas.

1.4 A Função de Risco

O primeiro passo para construir boas funções de predição é criar um critério para medir o desempenho de uma dada função de predição $g : \mathbb{R}^d \longrightarrow \mathbb{R}$. Em um contexto de regressão, faremos isso através de seu risco quadrático, embora essa não seja a única opção:

$$R_{pred}(g) = \mathbb{E}\left[(Y - g(\mathbf{X}))^2\right],$$

em que $(\mathbf{X}, Y)$ é uma nova observação que não foi usada para estimar g. Quanto menor o risco, melhor é a função de predição g. Mais detalhes sobre essa esperança serão apresentados na Observação 1.3.

Observação 1.2. A função $L(g(\mathbf{X}); Y) = (Y - g(\mathbf{X}))^2$ é chamada de *função de perda quadrática*. Outras funções de perda podem ser utilizadas, como por exemplo a *função de perda absoluta*; $L(g(\mathbf{X}); Y) = |Y - g(\mathbf{X})|$. Em geral, o risco é definido como a esperança de uma função de perda.

□

Quando medimos a performance de um estimador com base em seu risco quadrático, criar uma boa função de predição $g : \mathbb{R}^d \longrightarrow \mathbb{R}$ equivale a encontrar um bom estimador para a função de regressão $r(\mathbf{x})$. Assim, responder à pergunta do porquê estimar a função de regressão é, nesse sentido, o melhor caminho para se criar uma função para predizer novas observações Y com base em covariáveis observadas $\mathbf{x}$. Isso é mostrado no seguinte teorema:

Teorema 1. *Suponha que definimos o risco de uma função de predição $g : \mathbb{R}^d \longrightarrow \mathbb{R}$ via perda quadrática: $R_{pred}(g) = \mathbb{E}\left[(Y - g(\mathbf{X}))^2\right]$, em que $(\mathbf{X}, Y)$ é uma nova observação que não foi usada para estimar g. Suponhamos também que medimos o risco de um estimador da função de regressão via perda quadrática: $R_{reg}(g) = \mathbb{E}\left[(r(\mathbf{X}) - g(\mathbf{X}))^2\right]$. Então*

$$R_{pred}(g) = R_{reg}(g) + \mathbb{E}[\mathbb{V}[Y|\mathbf{X}]].$$

Demonstração.

$$\begin{aligned}
\mathbb{E}\left[(Y - g(\mathbf{X}))^2\right] &= \mathbb{E}\left[(Y - r(\mathbf{X}) + r(\mathbf{X}) - g(\mathbf{X}))^2\right] \\
&= \mathbb{E}\left[(r(\mathbf{X}) - g(\mathbf{X}))^2\right] + \mathbb{E}[(Y - r(\mathbf{X}))^2] + 2\mathbb{E}[(r(\mathbf{X}) - g(\mathbf{X}))(Y - r(\mathbf{X}))] \\
&= \mathbb{E}\left[(r(\mathbf{X}) - g(\mathbf{X}))^2\right] + \mathbb{E}[\mathbb{V}[Y|\mathbf{X}]],
\end{aligned}$$

em que o último passo segue do fato que

$$\mathbb{E}[(r(\mathbf{X}) - g(\mathbf{X}))(Y - r(\mathbf{X}))] = \mathbb{E}[\mathbb{E}[(r(\mathbf{X}) - g(\mathbf{X}))(Y - r(\mathbf{X}))|\mathbf{X}]] = 0$$

e

$$\mathbb{E}[(Y - r(\mathbf{X}))^2] = \mathbb{E}[\mathbb{E}[(Y - r(\mathbf{X}))^2|\mathbf{X}]] = \mathbb{E}[\mathbb{V}[Y|\mathbf{X}]],$$

pois $r(\mathbf{X}) = \mathbb{E}[Y|\mathbf{X}]$.

□

Assim, visto que o termo $\mathbb{E}[\mathbb{V}[Y|\mathbf{X}]]$[3] não depende de $g(\mathbf{x})$, estimar bem a função de regressão $r(\mathbf{x})$ é fundamental para se criar uma boa função de predição sob a ótica do risco quadrático, já que *a melhor função de predição para Y é a função de regressão $r(\mathbf{x})$*:

$$\arg\min_g R_{pred}(g) = \arg\min_g R_{reg}(g) = r(\mathbf{x}).$$

Observação 1.3. A função g no Teorema 1 pode ser entendida tanto como sendo fixa (neste caso $R(g)$ é chamado de *risco condicional*) quanto como sendo aleatória (função dos dados). Neste último caso, a esperança é tomada com relação à observação $(\mathbf{X}, Y)$ e à amostra $(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)$, e o risco é chamado de *risco esperado*. Em outras palavras, o risco esperado é a esperança do risco condicional sob todas as possíveis amostras usadas para criar g. Assim, ele pode ser entendido como uma garantia sobre o processo de criação de g ao invés de uma garantia sobre a particular função g criada para um dado problema. Por exemplo, se o risco esperado de uma regressão linear vale C em um dado problema, isso implica que, em média, uma regressão linear para dados que seguem a mesma distribuição desse conjunto e com o mesmo tamanho amostral possui risco condicional C. Dependendo do contexto, utilizaremos uma ou outra definição.

□

Daqui em diante, denotaremos por R o risco preditivo R_{pred}.

Em um contexto de predição, a definição de risco condicional (isto é, considerando g fixo; veja a Observação 1.3) possui grande apelo frequentista. Digamos que observamos um novo conjunto, $(\mathbf{X}_{n+1}, Y_{n+1}), \ldots, (\mathbf{X}_{n+m}, Y_{n+m})$, i.i.d à amostra observada. Então, pela lei dos grandes números, sabemos que, se m é grande,

$$\frac{1}{m}\sum_{i=1}^{m}(Y_{n+i} - g(\mathbf{X}_{n+i}))^2 \approx R(g) := \mathbb{E}\left[(Y - g(\mathbf{X}))^2\right].$$

Em outras palavras, se $R(g)$ possui valor *baixo*, então

$$g(\mathbf{x}_{n+1}) \approx y_{n+1}, \ldots, g(\mathbf{x}_{n+m}) \approx y_{n+m},$$

e, portanto, teremos boas predições em novas observações. Note que essa conclusão

[3] Para uma interpretação deste termo, veja a Seção 1.5.2.

vale para qualquer função de perda L utilizada para definir o risco, uma vez que

$$\frac{1}{m}\sum_{i=1}^{m} L(g(\mathbf{X}_{n+i}); Y_{n+i})) \approx R(g) := \mathbb{E}\left[L(g(\mathbf{X}); Y))\right].$$

O objetivo dos métodos de regressão sob a ótica preditivista é, portanto, fornecer, em diversos contextos, métodos que apresentem bons estimadores de $r(\mathbf{x})$, isto é, estimadores com risco baixo.

1.5 Seleção de Modelos: Super e Sub-Ajuste

> The best preparation for tomorrow is doing your best today.
>
> H. Jackson Brown Jr

Em problemas práticos, é comum ajustar vários modelos para a função de regressão $r(\mathbf{x})$ e buscar qual deles possui maior poder preditivo, isto é, qual possui menor risco. Isto é exemplificado a seguir.

Exemplo 1.5. [Expectativa de vida e PIB per Capita] Revisitamos aqui o Exemplo 1.1. A Figura 1.4 mostra o ajuste de três modelos distintos para a funçaõ de predição:

$$g(x) = \beta_0 + \sum_{i=1}^{p} \beta_i x^i, \text{ para } p \in \{1,4,50\}.$$

Em outras palavras, ajustamos três regressões polinomiais: uma de 1^o grau, uma de 4^o grau e uma de 50^o grau. Os ajustes foram feitos (ou seja, β_i foram estimados) por meio do método dos mínimos quadrados (Capítulo 2). Enquanto o modelo de 1^o grau é simplista demais para os dados, o modelo com $p = 50$ (ou seja, do 50^o grau) é extremamente complexo e parece fornecer uma função g que não produzirá boas predições em novas observações. O modelo com $p = 4$ parece ser o mais razoável nesse caso. Dizemos que o modelo com $p = 1$ sofre de *sub-ajuste*, ou *underfitting* (não é suficiente para explicar bem os dados), enquanto o modelo com $p = 50$ sofre de *super-ajuste*, ou *overfitting* (ele se ajusta demais a esta amostra específica, mas possui baixo poder de generalização). O objetivo desta seção é descrever um método para escolher o modelo intermediário entre sub-ajuste e super-ajuste. Nesse exemplo, queremos um método que escolha $p = 4$ automaticamente.

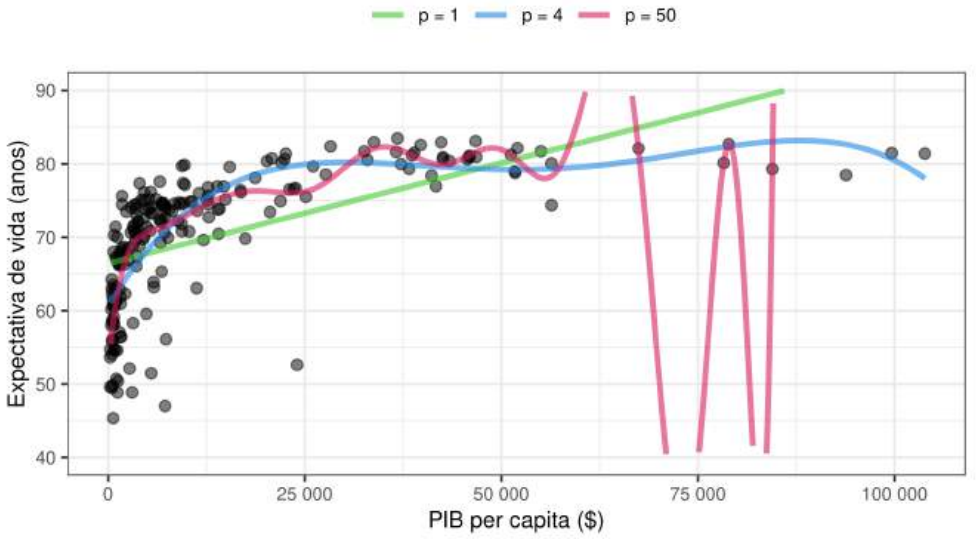


Figura 1.4: Comparação dos modelos $g(x) = \beta_0 + \sum_{i=1}^{p} \beta_i x^i$, para $p \in \{1,4,50\}$ nos dados de esperança de vida (Y) versus PIB per Capita (X).

□

Observação 1.4. Um modelo pode interpolar os dados e, ainda assim, não apresentar super-ajuste (Belkin et al., 2019b). Em outras palavras, mesmo que ocorra interpolação, uma função de predição pode ter bom poder preditivo. Esse fenômeno é frequentemente observado em redes neurais (Seção 4.11). Veja, por exemplo, Belkin et al. (2019a) e Belkin et al. (2018) para uma discussão mais aprofundada.

□

O objetivo de um método de seleção de modelos é selecionar uma boa função g. Nesse caso, utilizaremos o critério do risco quadrático para avaliar a qualidade da função. Logo, queremos escolher uma função g dentro de uma classe de candidatos $\mathbb{G}$ que possua bom poder preditivo (baixo risco quadrático). Dessa forma, queremos evitar modelos que sofram de sub ou super-ajuste. Uma vez que $R(g)$ é desconhecido, é necessário estimá-lo para avaliar a função $g \in \mathbb{G}$. Na próxima seção apresentaremos métodos para estimar esse risco.

1.5.1 *Data Splitting* e Validação Cruzada

O *risco observado* (também chamado de erro quadrático médio no conjunto de treinamento), definido por

$$EQM(g) := \frac{1}{n}\sum_{i=1}^{n}(Y_i - g(\mathbf{X}_i))^2 \tag{1.1}$$

é um estimador muito otimista do real risco. Se usado para fazer seleção de modelos, ele leva ao super-ajuste, um ajuste perfeito dos dados. Isto ocorre pois g foi escolhida de modo a ajustar bem $(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)$.

Uma maneira de solucionar este problema é dividir o conjunto de dados em duas partes, *treinamento* e *validação*:

$$\overbrace{(\mathbf{X}_1, Y_1), (\mathbf{X}_2, Y_2), \ldots, (\mathbf{X}_s, Y_s)}^{\text{Treinamento (por exemplo, 70\%)}} \quad , \quad \overbrace{(\mathbf{X}_{s+1}, Y_{s+1}), \ldots, (\mathbf{X}_n, Y_n)}^{\text{Validação (por exemplo, 30\%)}}.$$

Usamos o conjunto de treinamento exclusivamente para estimar g (por exemplo, estimar os coeficientes da regressão linear) e o conjunto de validação *apenas para estimar* $R(g)$ via

$$\hat{R}(g) = \frac{1}{n-s}\sum_{i=s+1}^{n}(Y_i - g(\mathbf{X}_i))^2 := \widehat{R}(g), \tag{1.2}$$

isto é, avaliamos o erro quadrático médio no conjunto de validação.

Uma boa prática para escolher quais amostras serão utilizadas para compor o conjunto de treinamento e quais serão utilizadas para compor o conjunto de validação é fazê-lo **aleatoriamente**. Dessa forma, utiliza-se um gerador de números aleatórios para escolher quais amostras serão usadas para o treinamento e quais serão usadas para a validação. Esse procedimento evita problemas quando o banco de dados está previamente ordenado de acordo com alguma covariável (por exemplo, quem coletou o banco pode ter ordenado as observações em função de alguma variável).

Como o conjunto de validação não foi usado para estimar os parâmetros de g, o estimador da Equação 1.2 é consistente pela lei dos grandes números.

Para uma função de perda L qualquer, o risco pode ser aproximado por

$$R(g) \approx \frac{1}{n-s} \sum_{i=s+1}^{n} L(g(\mathbf{X}_i); Y_i) := \widehat{R}(g).$$

O procedimento de dividir os dados em dois e utilizar uma parte para estimar o risco é chamado de *data splitting*. Uma variação deste método é a *validação cruzada*, que faz uso de toda a amostra. Por exemplo, no *leave-one-out cross validation* (LOOCV) (Stone, 1974), o estimador usado é dado por

$$\hat{R}(g) = \frac{1}{n} \sum_{i=1}^{n} (Y_i - g_{-i}(\mathbf{X}_i))^2,$$

em que g_{-i} é ajustado utilizando-se todas as observações exceto a i-ésima delas, ou seja, utilizando-se

$$(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_{i-1}, Y_{i-1}), (\mathbf{X}_{i+1}, Y_{i+1}), \ldots, (\mathbf{X}_n, Y_n).$$

Alternativamente, pode-se usar o *k-fold cross validation*. Nessa abordagem divide-se os dados aleatoriamente em *k-folds* (lotes) disjuntos com aproximadamente o mesmo tamanho, como mostra a Figura 1.5 para um cenário com 20 observações separadas em 5 lotes. Sejam $L_1, \ldots, L_k \subset \{1, \ldots, n\}$ os índices associados a cada um dos lotes. A ideia do *k-fold cross validation* é criar k estimadores da função de regressão, $\hat{g}_{-1}, \ldots, \hat{g}_{-k}$, em que $\hat{g}_{-j}$ é criado usando todas as observações do banco menos aquelas do lote L_j. O estimador do risco é dado por

$$\hat{R}(g) = \frac{1}{n} \sum_{j=1}^{k} \sum_{i \in L_j} (Y_i - g_{-j}(\mathbf{X}_i))^2.$$

Note que quando $k = n$, recuperamos o LOOCV. Para mais detalhes, veja Wasserman (2006).

Observação 1.5. Segue, da lei dos grandes números, que a estimação do risco baseada na divisão treinamento versus validação fornece um estimador consistente para o *erro condicional*. Por outro lado, o *leave-one-out cross validation* é aproximadamente não-viesado para o *erro esperado* (o risco descrito na Observação 1.3). De fato, se $R(g)$

é o erro esperado do método de construção de g, então

$$\mathbb{E}\left[\frac{1}{n+1}\sum_{i=1}^{n+1}(Y_i - g_{-i}(\mathbf{X}_i))^2\right] = \frac{1}{n+1}\sum_{i=1}^{n+1}\mathbb{E}\left[(Y_i - g_{-i}(\mathbf{X}_i))^2\right]$$
$$= \frac{1}{n+1}\sum_{i=1}^{n+1} R(g) = R(g),$$

em que a segunda igualdade segue do fato que g_{-i} tem a mesma distribuição de g. Assim, o estimador *leave-one-out cross validation*, calculado em um conjunto com $n+1$ observações, é não viesado para o erro esperado do estimador obtido com n amostras.

□

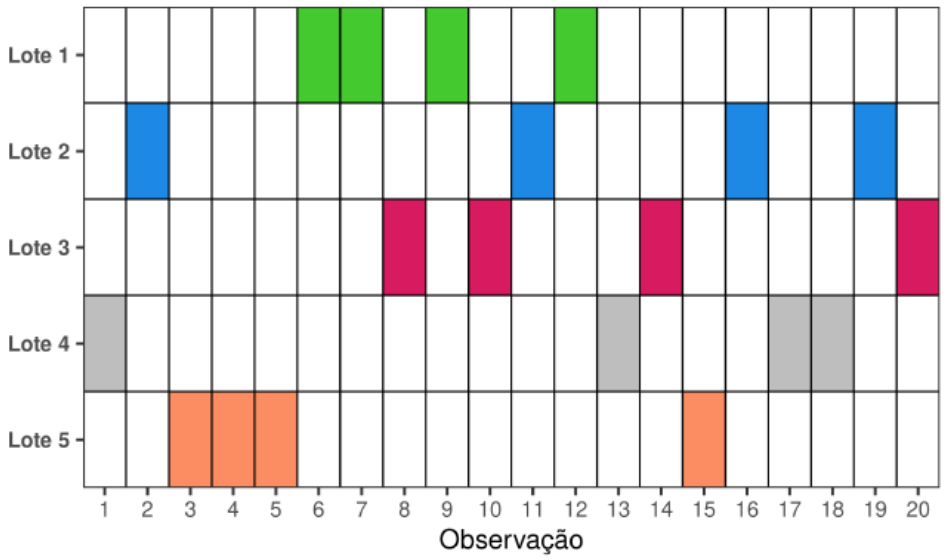


Figura 1.5: Esquema de validação cruzada para uma situação com 20 observações e 5 lotes.

Utilizando a validação cruzada para estimar $R(g)$ para cada $g \in \mathbb{G}$ nos permite escolher g com menor risco estimado.

Treino versus Validação versus Teste. Do mesmo modo que o EQM no conjunto de treinamento é muito otimista pois cada $g \in \mathbb{G}$ é escolhida de modo a (aproximada-

mente) minimizá-lo, o EQM no conjunto de validação *avaliado na g com menor EQM no conjunto de validação* também é otimista, principalmente se muitos métodos de predição forem avaliados no conjunto de validação. Isso pode levar a um super-ajuste ao conjunto de validação, isto é, pode levar à escolha de uma função g que tem bom desempenho no conjunto de validação, mas não em novas observações. Uma forma de contornar esse problema é dividir o conjunto original em três partes: treinamento, validação e teste. Os conjuntos de treinamento e validação são usados como descrito anteriormente. Já o conjunto de teste é usado para estimar o erro do melhor estimador da regressão determinado com a utilização do conjunto de validação. Assim, poderemos saber se o risco da melhor g encontrada de fato é pequeno.

Diversas variações dessa ideia podem ser usadas. Por exemplo, digamos que queremos comparar uma regressão polinomial com um lasso (Seção 3.3). Um procedimento razoável é: (i) escolher o melhor grau do polinômio e o *tuning parameter* λ do lasso usando *data-splitting* dentro do treinamento (isto é, dividindo o treinamento em duas partes: treinamento e validação) e (ii) comparar o desempenho da melhor regressão polinomial com o melhor lasso (segundo o critério (i)) no conjunto de teste. Assim, o teste será usado para comparar apenas dois modelos. Alternativamente, podemos trocar (i) por uma validação cruzada dentro do treino. De forma geral, idealmente usamos uma das divisões (a validação) para escolher os *tuning parameters* de cada um dos métodos e a outra (o teste) para comparar a melhor versão de cada método. Veja a Seção 7.2 para mais detalhes no contexto de classificação.

Intervalos de confiança para o risco. Utilizando o conjunto de teste, podemos também fazer um intervalo de confiança para o risco. Denote por $(\tilde{\mathbf{X}}_1, \tilde{Y}_1), \ldots, (\tilde{\mathbf{X}}_m, \tilde{Y}_m)$ os elementos do conjunto de teste (ou seja, não foram utilizados para treinamento ou validação do modelo). Um estimador não viesado para o risco de g (que foi estimada com base nos conjuntos de treinamento e validação) é dado por

$$\widehat{R}(g) = \frac{1}{m}\sum_{k=1}^{m} \underbrace{\left(\tilde{Y}_k - g(\tilde{\mathbf{X}}_k)\right)^2}_{W_k}.$$

Como $\widehat{R}(g)$ é uma média de variáveis i.i.d., o Teorema do Limite Central implica que

$$\widehat{R}(g) \approx \text{Normal}\left(R(g), \frac{1}{m}\mathbb{V}\left[W_1\right]\right).$$

Como $W_1, \ldots, W_m$ são i.i.d.'s, podemos estimar $\mathbb{V}[W_1]$ com

$$\widehat{\sigma^2} = \frac{1}{m}\sum_{k=1}^{m}\left(W_k - \overline{W}\right)^2,$$

em que $\overline{W} = \frac{1}{m}\sum_{k=1}^{m} W_k$. Assim, um intervalo de confiança (IC) aproximado de 95% para $R(g)$ é dado por

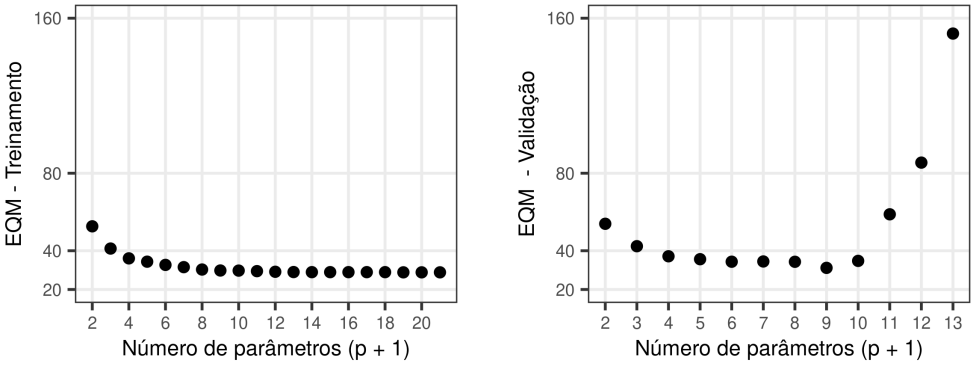
$$\widehat{R}(g) \pm 2\sqrt{\frac{1}{m}\widehat{\sigma^2}}. \tag{1.3}$$

A Equação 1.3 também nos dá um insight sobre como escolher o tamanho da divisão entre treino e validação. Podemos escolher o menor m tal que o tamanho do intervalo de confiança para o risco seja tão pequeno quanto queremos. Essa ideia é especialmente interessante se o tamanho amostra é grande. Neste caso, o tamanho do conjunto de validação pode ser bastante menor que o do conjunto de treino, uma vez que estimar o risco é uma tarefa muito mais fácil que criar uma função de predição.

Exemplo 1.6. [Expectativa de vida e PIB per Capita] Revisitamos os Exemplos 1.1 e 1.5. Aqui nosso objetivo é selecionar o melhor estimador dentro da classe

$$\mathbb{G} = \left\{ g(x) : g(x) = \beta_0 + \sum_{i=1}^{p} \beta_i x^i, \text{ para } p \in \{1,2,\ldots,50\} \right\}.$$

A Figura 1.6a mostra o erro quadrático médio *no conjunto de treinamento* para cada $g \in \mathbb{G}$. Observamos que, quanto maior o número de parâmetros, menor o erro quadrático médio no conjunto de treinamento. Por outro lado, a Figura 1.6b mostra o risco estimado para cada $g \in \mathbb{G}$ via EQM no conjunto de validação; como os riscos para $p > 11$ são demasiadamente altos, mostramos apenas o comportamento para $p \leq 11$. Vemos que o menor erro no conjunto de treinamento pode não levar ao menor erro no conjunto de validação. Isso acontece devido ao superajuste no conjunto de treinamento. Os gráficos da Figura 1.7 mostram que, de fato, minimizar o erro no conjunto de validação (gráfico da direita) leva a um melhor ajuste do que minimizar o erro no conjunto de treinamento (gráfico da esquerda). Assim, o *data splitting* nos leva à escolha de um modelo adequado para predizer novas observações.



(a) (b)

Figura 1.6: Comparação dos modelos $g(x) = \beta_0 + \sum_{i=1}^{p} \beta_i x^i$, para $p \in \{1,2,\ldots,50\}$ nos dados de esperança de vida (Y) versus PIB per Capita (X). EQM do conjunto de treinamento (esquerda) e EQM do conjunto de validação (direita). A primeira forma de fazer a seleção de modelos leva ao superajuste.

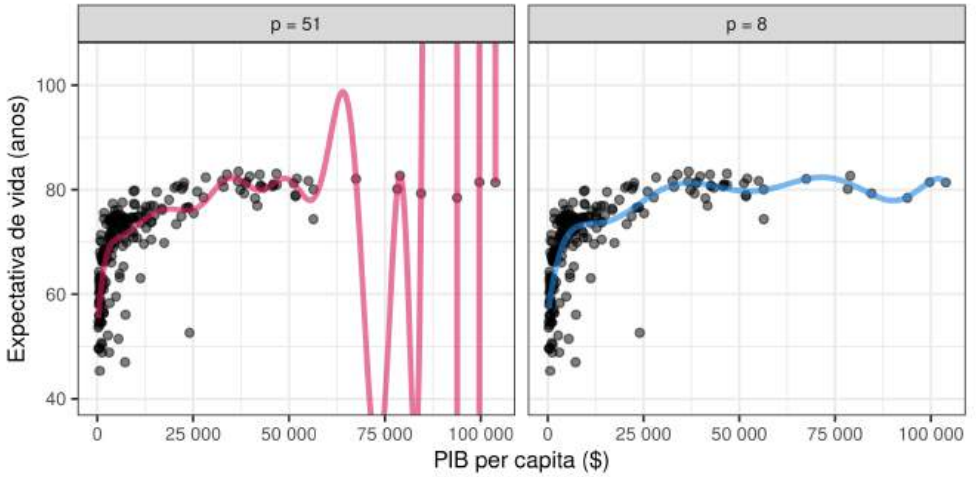


Figura 1.7: Comparação dos modelos ajustados segundo melhor EQM do conjunto de treinamento (esquerda) e segundo EQM do conjunto de validação (direita).

□

Exemplo 1.7. [A especificação do modelo estar correta não garante melhor poder

preditivo.] Dentro da cultura do *data modeling*, é comum checar se as suposições feitas por um modelo estão corretas. Contudo, a especificação do modelo estar correta não garante melhor poder preditivo. Isto é, um modelo especificado incorretamente em relação ao modelo que gerou os dados pode ter melhor desempenho. Nesta seção isso é ilustrado via um exemplo simulado. Os dados são gerados de modo que $Y|\mathbf{x} \sim N(\boldsymbol{\beta}^\intercal \mathbf{x}, 1)$, com $\boldsymbol{\beta} = (0,3,2,0.2,0.1,0.1)$ e $\mathbf{x} = (1, x, x^2, \ldots, x^5)$. Portanto, a regressão real é um polinômio de quinto grau. Ajustamos então um polinômio de quinto grau e um polinômio de segundo grau.

A Figura 1.8 indica que, apesar de o modelo correto ser um polinômio de quinto grau, um ajuste de segundo grau leva a melhores resultados. Isso ocorre pois o modelo mais simples possui variância muito menor (ainda que ele seja viesado) e, assim, evita o *overfitting*.

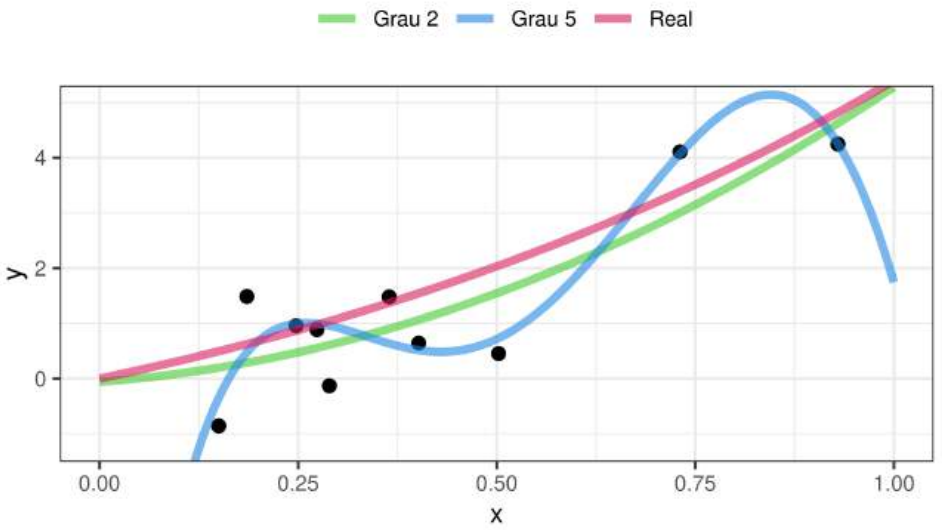


Figura 1.8: Neste exemplo, apesar de o modelo correto ser um polinômio de quinto grau, um ajuste de segundo grau levou a um resultado melhor.

A Tabela 1.2 mostra os erros estimados ao replicar o experimento acima para 1000 conjuntos de treinamento diferentes. O erro esperado do modelo com 2 graus, de fato, é menor do que o erro esperado do modelo especificado corretamente (isto é, com 5 graus). Além disso, para a grande maioria dos conjuntos de treinamento, o risco condicional do modelo mais simples é menor. Essa proporção observada foi de

97,8%.

Tabela 1.2: Risco de cada um dos modelos ajustados.

	2 graus	**5 graus**
Risco	1.7	22354.08

□

1.5.1.1 Penalização: uma alternativa

> Life is sometimes like knockout football; if you have no goal, invite penalty!
>
> Abhijit Kar Gupta

Uma forma alternativa de se estimar o risco de um certo modelo g é utilizando uma *medida de penalização* ou *complexidade*.

Quanto mais parâmetros no modelo, mais o erro quadrático médio observado, $EQM(g)$ (Eq. 1.1), subestima $R(g)$, isto é, maior a diferença entre $EQM(g)$ e $R(g)$. A ideia por trás de métodos de penalização é criar uma medida de complexidade para g, $\mathcal{P}(g)$, que é utilizada para corrigir essa diferença. Por exemplo, $\mathcal{P}(g)$ pode ser o número de parâmetros que há no modelo. Podemos então compensar o quão subestimado $R(g)$ é adicionando estas duas quantidades:

$$R(g) \approx EQM(g) + \mathcal{P}(g).$$

Existem diversas funções de penalização, com diferentes motivações teóricas. Dois exemplos populares são:

- $\mathcal{P}(g) = \frac{2}{n}p\widehat{\sigma}^2$ (conhecido como AIC);
- $\mathcal{P}(g) = \frac{\log(n)}{n}p\widehat{\sigma}^2$ (conhecido como BIC).

Aqui, p é o número de parâmetros de g e $\widehat{\sigma}^2$ é uma estimativa de $\mathbb{V}[Y|\mathbf{x}]$.

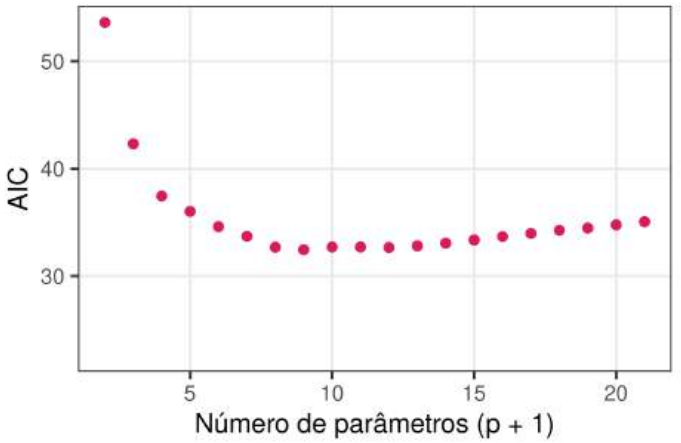


Figura 1.9: Comparação via AIC dos modelos $g(x) = \beta_0 + \sum_{i=1}^{p} \beta_i x^i$, para $p \in \{1,2,\ldots,50\}$ nos dados de esperança de vida versus PIB per Capita.

Note que, se g tem muitos parâmetros, $EQM(g)$ é em geral muito baixo (devido ao super-ajuste), mas em compensação $\mathcal{P}(g)$ é alto. Analogamente, se g tem poucos parâmetros, $EQM(g)$ é em geral muito alto (devido ao sub-ajuste), mas em compensação $\mathcal{P}(g)$ é baixo. Assim, espera-se que $EQM(g) + \mathcal{P}(g)$ seja uma boa estimativa para $R(g)$.

A Figura 1.9 ilustra o uso do AIC para seleção de modelos no Exemplo 1.6. Note como o comportamento deste gráfico é similar ao da Figura 1.6b. Em particular, os melhores valores encontrados para p em ambos são muito próximos.

1.5.2 Balanço entre Viés e Variância

> Simplicity is an exact medium between too little and too much.
>
> Sir Joshua Reynolds

Um grande apelo para o uso do risco quadrático é sua grande interpretabilidade: o risco quadrático (condicional no novo $\mathbf{x}$ observado) pode ser decomposto como

$$\mathbb{E}\left[(Y - \widehat{g}(\mathbf{X}))^2 | \mathbf{X} = \mathbf{x}\right] = \mathbb{V}[Y|\mathbf{X} = \mathbf{x}] + (r(\mathbf{x}) - \mathbb{E}[\widehat{g}(\mathbf{x})])^2 + \mathbb{V}[\widehat{g}(\mathbf{x})].$$

Note que estamos trabalhando com o erro esperado (veja a Observação 1.3), em que a aleatoriedade está tanto em Y quanto na amostra de treinamento e, por essa razão, estamos utilizando a notação $\widehat{g}$ para enfatizar que g é função dos dados.

Assim, o risco pode ser decomposto em três termos:

- $\mathbb{V}[Y|\mathbf{X} = \mathbf{x}]$ é a variância intrínseca da variável resposta, que não depende da função $\widehat{g}$ escolhida e, assim, não pode ser reduzida;
- $(r(\mathbf{x}) - \mathbb{E}[\widehat{g}(\mathbf{x})])^2$ é o quadrado do viés do estimador $\widehat{g}$ e
- $\mathbb{V}[\widehat{g}(\mathbf{x})]$ é a variância do estimador $\widehat{g}$.

É importante observar que os valores dos dois últimos itens podem ser reduzidos se escolhermos $\widehat{g}$ adequado.

Grosso modo, modelos com muitos parâmetros possuem viés relativamente baixo, mas variância alta, já que é necessário estimar todos eles. Já modelos com poucos parâmetros possuem variância baixa, mas viés muito alto, já que são demasiado simplistas para descrever o modelo gerador dos dados. Assim, com a finalidade de obter um bom poder preditivo, deve-se escolher um número de parâmetros nem tão alto, nem tão baixo. A Figura 1.10 mostra qualitativamente o balanço (também chamado de *tradeoff*) entre viés e variância.

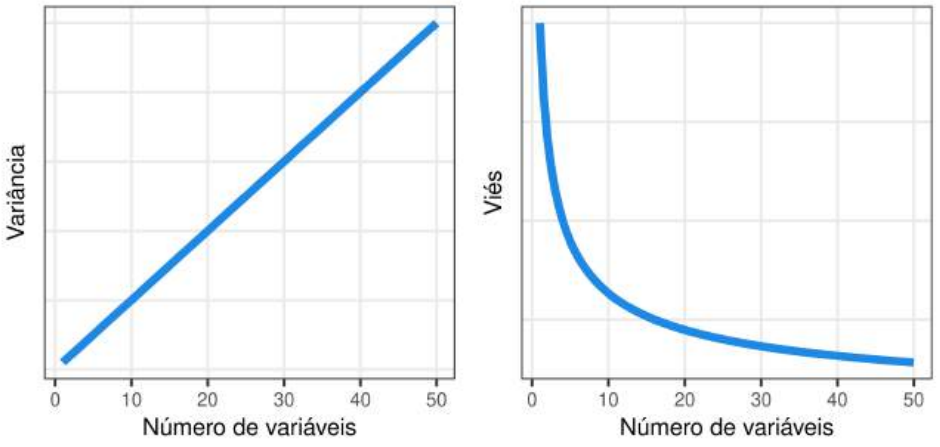


Figura 1.10: O balanço entre viés e variância.

Este *tradeoff* é justamente o que ocorre no Exemplo 1.5: enquanto $p = 50$ induz um modelo com viés relativamente baixo, mas variância alta (há muitos parâmetros para serem estimados); $p = 1$ leva a um viés extremamente alto, mas variância muito baixa. Ao se selecionar o melhor modelo utilizando, por exemplo, *data splitting* (veja Figura 1.6b) estamos justamente buscando a melhor combinação viés-variância de modo a obter um modelo com um risco baixo.

Note que, enquanto em inferência paramétrica tradicionalmente buscam-se estimadores não viesados para os parâmetros de interesse, em inferência preditiva é comum abrir mão de se ter um estimador viesado para, em troca, conseguir-se uma variância menor e, assim, um risco menor.

1.6 Tuning Parameters

A função do parâmetro p (grau do polinômio) no Exemplo 1.6 é controlar o balanço entre viés e variância. O valor ótimo de p depende de n e de $r(\mathbf{x})$. O parâmetro p é chamado de *tuning parameter* (isto é, um parâmetro de sintonização).

Vários dos métodos de regressão possuem um ou mais *tuning parameters*. Nesse livro, como feito no Exemplo 1.6, sempre iremos escolhê-los via validação cruzada ou *data splitting*, ainda que essas não sejam as únicas maneiras de fazer essa seleção (veja por exemplo Wasserman 2006).

1.7 Resumo

Nesse capítulo vimos que há ao menos dois motivos para se fazer uma análise de regressão: o *motivo inferencial*, no qual estamos interessados em tirar conclusões sobre a população à qual os dados pertencem, e o *motivo preditivista*, no qual estamos interessados em ter boas predições.

Vimos também que existem duas culturas em análise de dados: *data modeling culture* e *algorithmic modeling culture*. Enquanto na primeira há uma preocupação em se testar suposições dos modelos utilizados, na segunda o foco está em se obter modelos com bom poder preditivo.

Mostramos também que nosso objetivo na Parte I deste livro é mostrar diversos métodos que permitam estimar funções $g(\mathbf{x})$ com risco $R(g) = \mathbb{E}\left[(Y - g(\mathbf{X}))^2\right]$ baixo. Vimos que encontrar g com risco baixo equivale a encontrar uma boa estima-

tiva da função de regressão. Vimos também que o risco pode ser decomposto em viés e variância. Modelos complexos possuem variância alta, mas viés baixo, enquanto modelos simples possuem variância baixa, mas viés alto. Nosso objetivo é encontrar a complexidade que equilibre este balanço, de modo a termos um risco baixo.

Finalmente, estudamos dois métodos para estimar o risco de uma dada função g: a validação cruzada/*data splitting* e a penalização. Estes métodos podem ser utilizados para fazer seleção de modelos e, assim, equilibrar o balanço entre viés e variância.

Capítulo 2

Métodos Paramétricos

> Nature is pleased with simplicity.
> And nature is no dummy.
>
> Sir Isaac Newton

Métodos paramétricos assumem que a função de regressão pode ser parametrizada com um número finito de parâmetros. Neste capítulo iremos nos restringir apenas a modelos de regressão linear. Assumimos que a maior parte do conteúdo apresentado aqui é uma revisão de conceitos já conhecidos, mas sob uma perspectiva possivelmente diferente.

A regressão linear utiliza uma forma linear para a função de predição, ou seja, a função de predição usada pode ser escrita como

$$g(\mathbf{x}) = \boldsymbol{\beta}^\mathsf{T}\mathbf{x} = \beta_0 x_0 + \beta_1 x_1 + \ldots + \beta_d x_d, \tag{2.1}$$

em que adotamos a convenção $x_0 \equiv 1$, e onde $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_d)$. Note que x_i não é necessariamente a i-ésima variável original; podemos criar novas covariáveis que são funções das originais (ex: $x_i^2, x_i x_j$ etc; veja também a Seção 4.1 e Exemplo 1.5).

2.1 O Método dos Mínimos Quadrados

Uma forma de estimar os coeficientes $\boldsymbol{\beta}$ da regressão linear é utilizando o método dos mínimos quadrados, isto é,

$$\widehat{\boldsymbol{\beta}} = \arg\min_{\beta} \sum_{i=1}^{n} (Y_i - \beta_0 - \beta_1 x_{i,1} - \ldots - \beta_d x_{i,d})^2. \tag{2.2}$$

A solução para este problema é dada por

$$\widehat{\boldsymbol{\beta}} = (\widehat{\beta}_0, \widehat{\beta}_1, \ldots, \widehat{\beta}_d) = (\mathbb{X}^\intercal \mathbb{X})^{-1} \mathbb{X}^\intercal \mathbb{Y}, \tag{2.3}$$

em que

$$\mathbb{X} = \begin{pmatrix} X_{1,0} & \ldots & X_{1,d} \\ \vdots & \ddots & \vdots \\ X_{n,0} & \ldots & X_{n,d} \end{pmatrix}$$

e $\mathbb{Y} = (Y_1, \ldots, Y_n)$ (aqui usamos a convenção de que um vetor, quando utilizado como matriz, é um vetor *coluna*).

Assim, a função de regressão é estimada por

$$g(\mathbf{x}) = \widehat{\boldsymbol{\beta}}^\intercal \mathbf{x}.$$

Boa parte da literatura estatística é voltada para a justificativa do método dos mínimos quadrados sob um ponto de vista de estimadores de máxima verossimilhança, assim como para testes de aderência e métodos para a construção de intervalos de confiança para os parâmetros β_i. Indicamos para o leitor interessado nestes aspectos os livros Morettin e Singer (2019) e Neter et al. (1996).

Em geral, assumir que a verdadeira regressão é, de fato, linear é uma suposição muito forte. Existe, contudo, uma literatura substancial que tenta justificar o método dos mínimos quadrados para estimar os coeficientes mesmo quando a regressão real $r(\mathbf{x})$ não satisfaz essa suposição. Na seção que segue mostramos uma dessas ideias.

2.1.1 Mínimos Quadrados quando a Suposição de Linearidade Falha

> Being entirely honest with oneself is a good exercise.
>
> Sigmund Freud

A suposição de que a regressão real $r(\mathbf{x})$ é linear muitas vezes não é válida. Contudo, mesmo quando esta suposição falha, frequentemente existe um vetor $\boldsymbol{\beta}_*$ tal que $g_{\boldsymbol{\beta}_*}(\mathbf{x}) = \boldsymbol{\beta}_*^\intercal \mathbf{x}$ tem bom poder preditivo. Neste caso, o método dos mínimos quadrados tende a produzir estimadores com baixo risco. Isto ocorre pois $\widehat{\boldsymbol{\beta}}$ converge para o melhor preditor linear,

$$\boldsymbol{\beta}_* = \arg\min_{\boldsymbol{\beta}} R(g_{\boldsymbol{\beta}}) = \arg\min_{\boldsymbol{\beta}} \mathbb{E}\left[(Y - \boldsymbol{\beta}^\intercal \mathbf{X})^2\right], \tag{2.4}$$

em que $(\mathbf{X}, Y)$ é uma nova observação, mesmo quando a verdadeira regressão $r(\mathbf{x})$ não é linear. Além disso, o risco associado a $\widehat{\boldsymbol{\beta}}$ converge para o risco associado a $\boldsymbol{\beta}_*$.[1] Isto é mostrado no teorema a seguir.

Teorema 2. *Seja $\boldsymbol{\beta}_*$ o melhor preditor linear (Equação 2.4), e $\widehat{\boldsymbol{\beta}}$ o estimador de mínimos quadrados (Equação 2.3). Assuma que $\Sigma = \mathbb{E}[\mathbf{X}\mathbf{X}^\intercal]$ está bem definida. Então*

$$\widehat{\boldsymbol{\beta}} \xrightarrow[n\longrightarrow\infty]{P} \boldsymbol{\beta}_* \text{ e } R\left(g_{\widehat{\boldsymbol{\beta}}}\right) \xrightarrow[n\longrightarrow\infty]{P} R\left(g_{\boldsymbol{\beta}_*}\right)$$

Demonstração. Primeiramente, notamos que se minimizamos $\mathbb{E}\left[(Y - \boldsymbol{\beta}^\intercal \mathbf{X})^2\right]$, obtemos que

$$\boldsymbol{\beta}_* = \Sigma^{-1}\alpha,$$

em que $\alpha = (\mathbb{E}[YX_0], \ldots, \mathbb{E}[YX_d])$. Também notamos que podemos reescrever $\widehat{\boldsymbol{\beta}}$ como

$$\widehat{\boldsymbol{\beta}} = \widehat{\Sigma}^{-1}\widehat{\alpha},$$

em que $\widehat{\Sigma} = n^{-1}\sum_{i=1}^n \mathbf{X}_i\mathbf{X}_i^\intercal$ e $\widehat{\alpha} = (\widehat{\alpha}_0, \ldots, \widehat{\alpha}_d)$, com $\widehat{\alpha}_j = n^{-1}\sum_{i=1}^n Y_i X_{i,j}$. Pela lei fraca dos grandes números, temos que

$$\widehat{\Sigma} \xrightarrow[n\longrightarrow\infty]{P} \Sigma$$

[1] $\boldsymbol{\beta}_*$ é chamado de *oráculo*.

e

$$\widehat{\alpha} \xrightarrow[n\longrightarrow\infty]{P} \alpha,$$

de modo que, pelo Teorema de Mann-Wald,

$$\widehat{\boldsymbol{\beta}} \xrightarrow[n\longrightarrow\infty]{P} \boldsymbol{\beta}_*$$

e, assim, novamente pelo Teorema de Mann-Wald,

$$R(g_{\widehat{\boldsymbol{\beta}}}) \xrightarrow[n\longrightarrow\infty]{P} R\left(g_{\boldsymbol{\beta}_*}\right)$$

□

O teorema anterior mostra, portanto, que $\widehat{\boldsymbol{\beta}}$ converge para o melhor preditor linear, $\boldsymbol{\beta}_*$. Pode-se também derivar a taxa desta convergência, isto é, o quão rápido ela ocorre; veja por exemplo Györfi et al. (2006).

2.1.2 Regressão Linear no R

Seja `dados` é um data frame com o banco de dados, o estimador de mínimos quadrados considerando duas preditoras (`explicativa_1` e `explicativa_2`) pode ser calculado utilizando

```
ajuste <- lm(resposta ~ explicativa_1 + explicativa_2,
                data = dados)
```

O padrão dessa função considera a inclusão de um intercepto. Para ajustar a regressão com todas as covariáveis no banco de dados, é possível utilizar o atalho

```
ajuste <- lm(resposta ~ ., data = dados)
```

Uma vez que o modelo foi ajustado, é possível fazer previsões para um novo banco de dados (`novos_dados`) com a função `predict`:

```
previsoes <- predict(ajuste, newdata = novos_dados)
```

Note que `novos_dados` deve ser um dataframe com o mesmo formato (incluindo nomes das variáveis) que `dados`.

2.2 Resumo

Vimos neste capítulo que uma regressão linear assume que a função de regressão pode ser escrita como uma expansão da forma $g(\mathbf{x}) = \boldsymbol{\beta}^\intercal \mathbf{x} = \beta_0 + \beta_1 x_1 + \ldots + \beta_d x_d$. O vetor $\mathbf{x}$ pode representar as covariáveis originais ou transformações dessas. Estudamos o método dos mínimos quadrados, uma forma de se estimar o vetor $\boldsymbol{\beta}$, e mostramos que, mesmo quando a verdadeira regressão não é linear, esse estimador converge para $\boldsymbol{\beta}_*$, o melhor preditor linear de y com base em $\mathbf{x}$.

Infelizmente, quando há muitas covariáveis, o método dos mínimos quadrados não leva a bons resultados devido ao super-ajuste e à variância extremamente alta (Seção 1.5). Quando $d > n$, o estimador não está bem definido, pois a matriz $\mathbb{X}^\intercal\mathbb{X}$ utilizada para estimar $\boldsymbol{\beta}$ (Eq. 2.3) não é invertível. No próximo capítulo mostramos algumas abordagens para contornar esse problema.

Capítulo 3

Métodos Paramétricos em Altas Dimensões

O estimador de mínimos quadradados, quando há muitas covariáveis (isto é, d é grande), possui baixo desempenho devido ao super-ajuste. Há muitos parâmetros a serem estimados e, portanto, a função de regressão estimada possui baixo poder preditivo. Em outras palavras, a variância do estimador resultante é alta pois muitos parâmetros devem ser estimados[1]. Neste capítulo investigamos alguns métodos que podem ser usados para contornar esse problema.

3.1 Busca pelo Melhor Subconjunto de Covariáveis

Uma solução para esse problema consiste em retirar algumas variáveis da regressão de modo a diminuir a variância da função de predição estimada. Em outras palavras, essa solução consiste em aumentar o viés do estimador da regressão real $r(\mathbf{x})$ e, em troca, esperar que isso resulte em uma diminuição substancial de sua variância. Caso algumas covariáveis observadas não estão associadas (ou têm pouca associação) à resposta Y, o aumento do viés será pequeno, de modo que o poder preditivo irá aumentar.

Uma maneira de retirar algumas variáveis da regressão é buscar a estimativa

[1] Se $d > n$, o método de mínimos quadrados nem pode ser implementado, uma vez que $\mathbb{X}^\intercal\mathbb{X}$ não é invertível!

dada por

$$\widehat{\boldsymbol{\beta}}_{L_0} = \arg \min_{\beta_0 \in \mathbb{R}, \boldsymbol{\beta} \in \mathbb{R}^d} \sum_{k=1}^{n} \left(y_k - \beta_0 - \sum_{i=1}^{d} \beta_i x_{k,i} \right)^2 + \lambda \sum_{i=1}^{d} \mathbb{I}\left(\beta_i \neq 0\right). \tag{3.1}$$

A penalização $\sum_{i=1}^{d} \mathbb{I}\left(\beta_i \neq 0\right)$ induz modelos com poucas covariáveis, em particular se λ é alto. Por exemplo, quando $\lambda \longrightarrow \infty$, a solução de 3.1 é $\widehat{\boldsymbol{\beta}}_{L_0} \equiv (\bar{y}, \mathbf{0})$, ou seja, um vetor de zeros para os coeficientes angulares (isto é, a penalização alta descarta todas as covariáveis do modelo). É importante notar que não há penalização para β_0.

No outro extremo, quando $\lambda = 0$, temos que $\widehat{\boldsymbol{\beta}}_{L_0}$ é o estimador de mínimos quadrados (isto é, nenhuma variável é descartada sem penalização). Quando $\lambda = \frac{2}{n}\widehat{\sigma}^2$, encontrar a solução da Equação 3.1 equivale a uma busca entre os 2^d modelos na classe

$$\begin{aligned}
\mathbb{G} = \{ & g(\mathbf{x}) = \widehat{\beta}_0, \\
& g(\mathbf{x}) = \widehat{\beta}_0 + \widehat{\beta}_1 x_1, \\
& g(\mathbf{x}) = \widehat{\beta}_0 + \widehat{\beta}_2 x_2, \\
& \dots \\
& g(\mathbf{x}) = \widehat{\beta}_0 + \widehat{\beta}_d x_d, \\
& g(\mathbf{x}) = \widehat{\beta}_0 + \widehat{\beta}_1 x_1 + \widehat{\beta}_2 x_2, \\
& g(\mathbf{x}) = \widehat{\beta}_0 + \widehat{\beta}_1 x_1 + \widehat{\beta}_3 x_3, \\
& \dots \\
& g(\mathbf{x}) = \widehat{\beta}_0 + \widehat{\beta}_1 x_1 + \widehat{\beta}_2 x_2 + \ldots + \widehat{\beta}_d x_d \},
\end{aligned}$$

em que se utiliza o critério AIC para determinar o melhor modelo. Em outras palavras, quando $\lambda = \frac{2}{n}\widehat{\sigma^2}$, encontrar a solução da Equação 3.1 é equivalente a ajustar cada um dos 2^d modelos via mínimos quadrados e estimar o risco $R(g)$ para cada um deles via AIC (Seção 1.5.1.1). O modelo escolhido é aquele com menor risco estimado.

De um ponto de vista teórico, estimar $\boldsymbol{\beta}$ com base na Equação 3.1 resolve o problema, isto é, conseguimos descartar várias das variáveis e, assim, obter um estimador com variância (e erro preditivo) menor. Em contrapartida, há um aumento no viés do estimador, já que nem todas as variáveis são utilizadas. Como explicado na

Seção 1.5.2, quando o aumento no viés é pequeno se comparado à diminuição na variância, o modelo resultante tem poder preditivo maior. Contudo, do ponto de vista prático, resolver a Equação 3.1 é computacionalmente trabalhoso quando há muitas covariáveis, uma vez que há 2^d modelos para serem ajustados. A Figura 3.1 ilustra como o número de modelos cresce à medida que d aumenta.

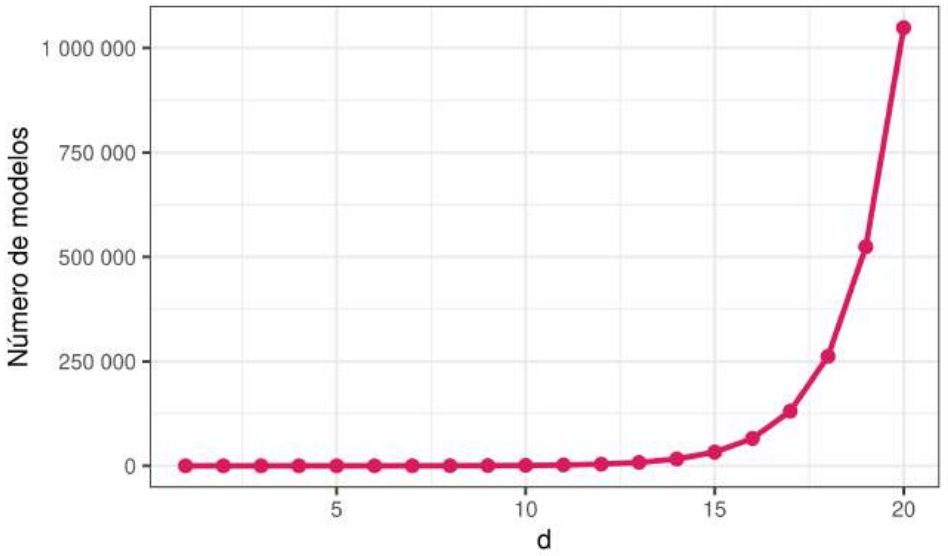


Figura 3.1: Número de subconjuntos de d covariáveis como função de d.

Uma vez que esse método é impráticavel em diversas situações, é interessante utilizar alguma heurística que reduza o número de modelos que devem ser ajustados e, ainda assim, apresentem um risco pequeno. Isto é, ao invés de resolver a Equação 3.1, busca-se por uma aproximação desta solução. Um desses métodos é descrito na próxima seção.

3.2 Regressão Stepwise

Uma série de algoritmos que consistem em *heurísticas* que consideram um número reduzido de modelos (em relação aos 2^d do melhor subconjunto) é chamada de *stepwise regression* (James et al., 2013). Um exemplo desta heurística é o *forward stepwise*, um algoritmo sequencial no qual a cada passo apenas uma variável é adici-

onada:

1. Para $j = 1,\ldots,d$, ajuste a regressão de Y na j-ésima variável X_j. Seja $\widehat{R}(g_j)$ o risco estimado desta função (usando AIC ou validação cruzada). Defina
$$\widehat{j} = \arg\min_j \widehat{R}(g_j) \qquad \text{e} \qquad S = \{\widehat{j}\}.$$

2. Para cada $j \in S^c$, ajuste a regressão de $Y = \beta_j X_j + \sum_{s\in S} \beta_s X_s + \epsilon$, e seja $\widehat{R}(g_j)$ o risco estimado desta função (usando AIC ou validação cruzada). Defina
$$\widehat{j} = \arg\min_{j\in S^c} \widehat{R}(g_j) \qquad \text{e atualize} \qquad S \longleftarrow S \cup \widehat{j}.$$

3. Repita o passo anterior até que todas as variáveis estejam em S ou até que não seja possível mais ajustar a regressão

4. Selecione o modelo com menor risco estimado.

Note que, no *forward stepwise*, ao invés de buscarmos um entre 2^d modelos, é necessário investigar somente $1 + d(d+1)/2$ modelos. A Figura 3.2 ilustra a diferença na magnitude do números de modelos ajustados em cada método.

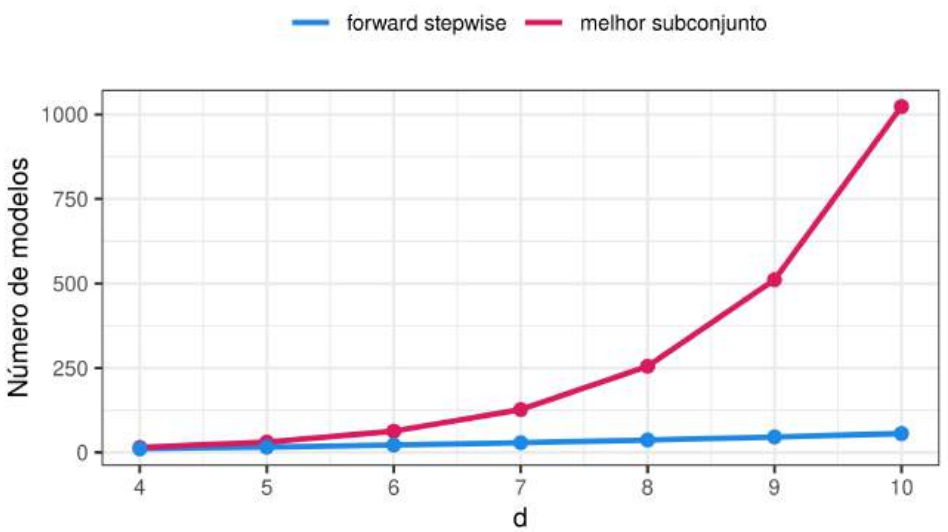


Figura 3.2: Número de modelos utilizando abordagem de melhor subconjunto e *forward stepwise* como função do número d.

Uma segunda maneira de encontrar um estimador linear da regressão com bom risco quando há muitas covariáveis é o *lasso*. Descrevemos esse método a seguir.

3.3 Lasso

> The penalty may be removed, the crime is eternal.
>
> Ovid

O lasso, desenvolvido por Tibshirani (1996), tem como finalidade encontrar um estimador de uma regressão linear que possui risco menor que o de mínimos quadrados. O lasso possui duas grandes vantagens em relação à heurísticas *stepwise*: sua solução é mais rápida e possui mais garantias teóricas.

Assim como no método baseado na Equação 3.1, o lasso consiste em encontrar uma solução $\boldsymbol{\beta}$ que minimize a soma de seu erro quadrático médio com uma medida de complexidade de $\boldsymbol{\beta}$. Novamente, a ideia é reduzir a variância do estimador de mínimos quadrados. Contudo, ao invés de medir a complexidade de um modelo como uma função do número de parâmetros (isto é, $\sum_{i=1}^{d} \mathbb{I}(\beta_i \neq 0)$, a norma L_0), no lasso essa complexidade é medida pela norma L_1 desse vetor, $\sum_{j=1}^{d} |\beta_j|$. Assim como a norma L_0, a norma L_1 é menor para modelos com muitos coeficientes iguais a zero. Contudo, ao contrário da norma L_0, a norma L_1 captura a ideia de que uma pequena mudança nos valores de $\boldsymbol{\beta}$ não altera demasiadamente a complexidade do modelo resultante (já que suas predições serão praticamente as mesmas). Isso fica evidente na Tabela 3.1: as duas primeiras linhas mostram coeficentes que levam a modelos preditivos parecidos, contudo, enquanto a norma L_1 dos dois modelos é próxima, a norma L_0 é muito diferente. Analogamente, os modelos da linha dois e três levam a predições muito diferentes (por exemplo, uma pequena alteração em x_i resultará em grande alteração na predição dada pelo segundo o modelo da terceira, mas não no modelo da segunda linha), mas a norma L_0 dois dois é a mesma.

Formalmente, no lasso, buscamos por

$$\widehat{\boldsymbol{\beta}}_{L_1,\lambda} = \arg\min_{\boldsymbol{\beta}} \sum_{k=1}^{n} \left(y_k - \beta_0 - \sum_{j=1}^{d} \beta_j x_{k,j} \right)^2 + \lambda \sum_{j=1}^{d} |\beta_j|, \tag{3.2}$$

em que L_1 indica o fato de que estamos medindo a esparsidade de um vetor $\boldsymbol{\beta}$ usando

sua norma em L_1, $||\boldsymbol{\beta}||_{L_1} = \sum_{j=1}^{d} |\beta_j|$.

Tabela 3.1: Diferentes medidas de esparsidade/penalização.

β	Penalidade		
	$\sum_{j=1}^{d} I(\beta_j \neq 0)$	$\sum_{j=1}^{d} \lvert\beta_j\rvert$	$\sum_{j=1}^{d} \beta_j^2$
(0, 0, 0, 0, 0, 0, 0, 0, 0, 0)	0	0	0.0
(0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1)	10	1	0.1
(1, 1, 1,1, 1, 1, 1, 1, 1,1)	10	10	10.0

Note que cada valor do parâmetro λ (*tuning parameter*) leva a um conjunto de coeficientes estimados $\widehat{\boldsymbol{\beta}}_{L_1,\lambda}$ diferentes. Assim como ocorre quando usamos a penalização da Equação 3.1, quando $\lambda = 0$, minimizar a Eq. 3.2 é equivalente a minimizar a Eq. 2.2, ou seja, o lasso torna-se idêntico ao estimador de mínimos quadrados, com todos coeficientes diferentes de zero. Por outro lado, quando λ é grande,

$$\sum_{k=1}^{n} \left(y_k - \beta_0 - \sum_{j=1}^{d} \beta_j x_{k,j} \right)^2 + \lambda \sum_{j=1}^{d} |\beta_j| \;\approx\; \lambda \sum_{j=1}^{d} |\beta_j|,$$

e portanto o estimador dos coeficientes é tal que $\widehat{\beta}_1 = 0, \ldots, \widehat{\beta}_d = 0$. Assim, para λ grande, o estimador dado pelo lasso tem variância próxima a zero, mas um viés muito alto. Desta maneira, adicionar a penalização $\lambda \sum_{j=1}^{d} |\beta_j|$ também induz estimadores com variância menor que aqueles dados pelo método de mínimos quadrados.

A escolha de λ é em geral feita pela validação cruzada. Isto é, estimamos $R(g_\lambda)$ para cada λ de interesse[2] e, posteriormente, selecionamos λ que leva ao melhor modelo selecionado, como mostra a Figura 3.3.

Além da penalização L_1 capturar a complexidade de forma que muitas vezes é mais razoável que a L_0, o lasso também possui duas outras características positivas:

- É extremamente rápido calcular $\widehat{\boldsymbol{\beta}}_{L_1,\lambda}$ para todos os λ simultaneamente. Diversos algoritmos foram desenvolvidos nos últimos anos para fazer esta tarefa, sendo o LARS um dos primeiros desses algoritmos. Para mais detalhes, veja Friedman et al. (2010). Em particular, por ser um problema convexo, resolver o

[2] A cada λ corresponde um $\widehat{\boldsymbol{\beta}}_{L_1,\lambda}$, que por sua vez corresponde uma função de predição $g(\mathbf{x})$.

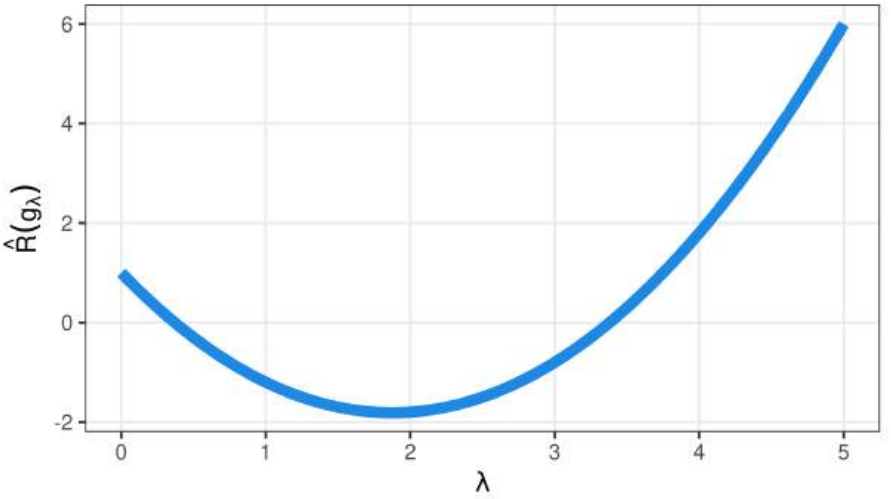


Figura 3.3: Processo de escolha de λ via validação cruzada. O risco para cada valor de λ é estimado usando-se validação cruzada.

problema de otimização do lasso é muito mais rápido que buscar pelo melhor subconjunto de covariáveis como na Equação 3.1.

- A solução induzida pela Equação 3.2 possui muitos zeros (isto é, o vetor $\widehat{\beta}_{L_1,\lambda}$ é esparso). Assim, o modelo resultante é de fácil interpretação. Note que o fato de $\widehat{\beta}_{L_1,\lambda}$ ser esparso não é trivial (ao contrário do que ocorre com a penalização $\sum_{j=1}^{d} \mathbb{I}(\beta_j \neq 0)$). Para mais detalhes sobre esse aspecto veja, por exemplo, Hastie et al. 2001.

3.3.1 Garantias Teóricas

O lasso apresenta algumas garantias teóricas interessantes. Mesmo quando a regressão não é, de fato linear, o estimador converge para o oráculo esparso. Mais especificamente, temos o seguinte teorema:

Teorema 3. *(Greenshtein, Ritov et al., 2004) Seja*

$$\beta_*^L = \arg\min_{\beta} \mathbb{E}[(Y - \beta^\intercal \mathbf{X})^2] \text{ sujeito a } ||\beta||_1 \leq L$$

o melhor preditor linear esparso com norma menor ou igual a L. Se $(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)$ *são*

i.i.d.'s e $|Y|, |\mathbf{X}_1|, \ldots, |\mathbf{X}_n| \leq B$ *para algum* $B > 0$*, então*

$$\widehat{\boldsymbol{\beta}}_L = \arg\min_{\boldsymbol{\beta}} \frac{1}{n} \sum_{i=1}^{n} (Y_i - \boldsymbol{\beta}^\intercal \mathbf{X}_i)^2 \text{ sujeito a } ||\boldsymbol{\beta}||_1 \leq L$$

(veja na Equação 3.5 que este é justamente o estimador dado pelo lasso) é tal que, com probabilidade ao menos $1 - \delta$*,*

$$R(g_{\widehat{\boldsymbol{\beta}}_L}) - R(g_{\boldsymbol{\beta}_*^L}) \leq \sqrt{\frac{16(L+1)^4 B^2}{n} \log\left(\frac{\sqrt{2}d}{\sqrt{\delta}}\right)}.$$

Note que, quanto menor o valor de L no Teorema 3, mais próximo o risco do estimador do lasso, $\widehat{\boldsymbol{\beta}}_L$, fica do risco do oráculo $\boldsymbol{\beta}_*^L$. Ou seja, mais fácil é se recuperar o melhor $\boldsymbol{\beta}$ com norma menor que L. Por outro lado, quanto menor o valor de L, pior é esse oráculo; aumentar L necessariamente melhora o oráculo, pois a minimização do risco (teórico) é feita em um espaço maior .

3.4 Regressão Ridge

Uma alternativa ao lasso (que na realidade é anterior a ele) é a regressão ridge (Hoerl e Kennard, 1970). A ideia é novamente buscar o modelo que minimize a soma do erro quadrático médio de g com uma medida de sua complexidade. Na regressão ridge isso é feito encontrando-se o estimador dado por

$$\widehat{\boldsymbol{\beta}}_{L_2,\lambda} = \arg\min_{\boldsymbol{\beta}} \sum_{k=1}^{n} \left(y_k - \beta_0 - \sum_{j=1}^{d} \beta_j x_{k,j}\right)^2 + \lambda \sum_{j=1}^{d} \beta_j^2, \tag{3.3}$$

em que L_2 indica o fato de que estamos medindo a complexidade de um vetor $\boldsymbol{\beta}$ usando sua norma em L_2, $||\boldsymbol{\beta}||_{L_2}^2 = \sum_{j=1}^{d} \beta_j^2$. Veja novamente a Tabela 3.1 para uma comparação entre as diferentes normas estudadas.

Ao contrário do lasso, a regressão ridge possui solução analítica, dada por

$$\widehat{\boldsymbol{\beta}}_{L_2,\lambda} = (\widehat{\beta}_0, \widehat{\beta}_1, \ldots, \widehat{\beta}_d) = (\mathbb{X}^\intercal \mathbb{X} + \lambda \mathbb{I}_0)^{-1} \mathbb{X}^\intercal \mathbb{Y}, \tag{3.4}$$

em que $\mathbb{I}_0$ é uma matriz identidade $(d+1) \times (d+1)$ modificada de modo que $\mathbb{I}_0(1,1) = 0$, ou seja, o elemento da primeira linha e primeira coluna é igual a zero.

Compare essa solução com o estimador de mínimos quadrados (Equação 2.3). Apesar de não introduzir soluções com zeros como o lasso, a regressão ridge também diminui a variância dos estimadores da regressão pois *encolhe* (em inglês, *shrinks*) os coeficientes β estimados pela regressão linear. Por exemplo, no caso em que as covariáveis originais são ortogonais (isto é, $\mathbb{X}^\intercal\mathbb{X} = \mathbb{I}$), temos $\widehat{\boldsymbol{\beta}}_{L_2,\lambda} = \widehat{\beta}/(1+\lambda)$, em que $\widehat{\boldsymbol{\beta}}$ é o estimador de mínimos quadrados. Assim,

$$\mathbb{V}\left[\widehat{\beta}_{i,L_2,\lambda}\right] = \frac{\mathbb{V}\left[\widehat{\beta}_i\right]}{(1+\lambda)^2}.$$

Evidentemente, como no lasso, apesar da variância da regressão ridge ser menor, seu viés é maior. Assim, λ deve ser escolhido de modo a controlar o balanço viés-variância. Novamente, isso pode ser feito via validação cruzada.

Na Seção 4.6.3, estudamos uma extensão não linear da regressão ridge, a kernel ridge regression.

3.5 Formulação Alternativa

O lasso possui uma formulação alternativa. Pode-se mostrar que, para cada $\lambda \geq 0$, existe $B \geq 0$ tal que a Equação 3.2 é equivalente a

$$\arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_1 - \ldots - \beta_d x_d)^2 \text{ sujeito a } \sum_{j=1}^{d}|\beta_j| \leq B. \tag{3.5}$$

Analogamente, as Equações 3.1 e 3.3 (melhor subconjunto e regressão ridge, respectivamente) podem ser reexpressas como

$$\arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_1 - \ldots - \beta_d x_d)^2 \text{ sujeito a } \sum_{j=1}^{d}\mathbb{I}(\beta_j \neq 0) \leq B_1$$

e

$$\arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_1 - \ldots - \beta_d x_d)^2 \text{ sujeito a } \sum_{j=1}^{d}(\beta_j)^2 \leq B_2,$$

para algum B_1 e B_2. Isso evidencia mais uma vez que, em todas as abordagens, a penalização favorece coeficientes "pequenos" quando comparados à solução de mínimos quadrados. Contudo, enquanto lasso e a penalização por AIC fazem seleção

de variáveis (zeram alguns coeficientes), o mesmo não ocorre com a regressão ridge.

3.6 Interpretação Bayesiana

> A penalty is a cowardly way to score.
>
> Edson Arantes do Nascimento (Pelé)

Tanto a regressão ridge quanto o lasso admitem uma interpretação sob o ponto de vista Bayesiano: os estimadores de $\boldsymbol{\beta}$, em ambos os procedimentos, podem ser escritos como sendo a moda da distribuição a posteriori de $\boldsymbol{\beta}$ para uma dada distribuição a priori e verossimilhança. Mais especificamente, se assumimos que

$$Y_i|\mathbf{x}_i, \boldsymbol{\beta} \sim N(\boldsymbol{\beta}^\intercal \mathbf{x}_i; \sigma^2 \mathbb{I}_d),$$

em que $\mathbb{I}_d$ é a matriz identidade de ordem d, as variáveis $Y_1, \ldots, Y_n$ são independentes e σ^2 conhecido, então:

- Se $\boldsymbol{\beta} \sim N(\mathbf{0}, \sigma^2_{\beta} \mathbb{I}_d)$, então a moda da distribuição a posteriori de $\boldsymbol{\beta}$, dado o conjunto de dados $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, é exatamente o estimador de $\boldsymbol{\beta}$ dado pela regressão ridge (Eq. 3.3) com $\lambda = \sigma^2 / \sigma^2_{\beta}$;

- Se $\beta_1, \ldots, \beta_d$ i.i.d. $\sim$ Laplace$(\mathbf{0}, \tau_{\beta})$,[3] a moda da distribuição a posteriori de $\boldsymbol{\beta}$, dado o conjunto de dados $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, é exatamente o estimador de $\boldsymbol{\beta}$ dado pelo lasso (Eq. 3.2) com $\lambda = \sigma^2 \tau_{\beta}$.

Note que, sob a abordagem Bayesiana, o *tuning parameter* λ é definido pela distribuição a priori. Quanto menor a dispersão da priori, maior o valor de λ e, consequentemente, mais próximo de zero estarão as estimativas a posteriori. Assim, a abordagem Bayesiana evita o *overfitting*, por mais que esse não seja necessariamente seu objetivo primordial. O interessante é que isso ocorre justamente porque a priori centrada em $\mathbf{0}$ reflete a crença de que os parâmetros relativos à maioria das covariáveis, em geral, devem ser pequenos.Isso leva a uma interpretação filosófica

[3] Aqui, τ_{β} é um parâmetro de precisão.

interessante: escolher λ por validação cruzada equivale a escolher a distribuição a priori que leva a um maior poder preditivo. Em outras palavras, usamos a maquinaria Bayesiana apenas para motivar estimadores com uma performance (preditiva) melhor que a dos mínimos quadrados.

Veja mais sobre a relação entre estimadores de Bayes e os estimadores descritos neste capítulo em Bishop (2006) e Park e Casella (2008).

3.7 Regressão ridge e lasso no R

No R, ambos os métodos podem ser implementados via a biblioteca `"glmnet"`. Se `x` é a matriz com as covariáveis e `y` é a matriz com os valores da variável resposta, a validação cruzada para o lasso pode ser feita utilizando o seguinte comando:

```
ajuste <- cv.glmnet(x, y, alpha = 1)
```

A regressão ridge pode ser ajustada utilizando essa mesma função, mas alterando o argumento `alpha = 0` para `alpha = 1`.

3.8 Exemplos

Exemplo 3.1. Geramos $n = 500$ observações i.i.d. segundo

$$Y_k = 3X_{k,1} - 2X_{k,2} + X_{k,3} - 3X_{k,4} + X_{k,5} + \sum_{i=6}^{20} 0X_{k,i} + \epsilon_k,$$

com $\epsilon_k \sim N(0, 0.5^2)$ e $X_{k,i} \sim N(0,1)$, $i = 1, \ldots, 20$ independentes.

A Tabela 3.2 mostra os resultados encontrados. Pode-se observar que o método dos mínimos quadrados com todas as variáveis foi rápido de ser ajustado, mas tem poder preditivo muito baixo (o risco estimado é alto). Por outro lado, todos os métodos de seleção de variáveis vistos nesse texto possuem poder preditivo consideravelmente melhor para este exemplo.

A heurística do *forward stepwise* fornece os mesmos resultados que a da busca pelo mínimo da Equação 3.1 (critério AIC), com a vantagem de levar um tempo substancialmente menor. Por outro lado, o lasso forneceu um risco ainda menor em um intervalo de tempo ainda mais curto; em particular as variáveis selecionadas foram

as que realmente estão associadas a Y. A regressão ridge foi substancialmente melhor que o método de mínimos quadrados com todas as covariáveis, mas levemente pior que os demais métodos. Contudo, o tempo de execução desse método foi menor do que o tempo observado para *forward stepwise*.

Tabela 3.2: Resultados dos métodos de seleção de variáveis no exemplo da Seção 3.8.

Método	**Variáveis Selecionadas**	**Tempo de ajuste**	**Risco Estimado[a]**
Mín. Quadrados	Todas	0.002 segundos	14.63 (0.02)
Melhor AIC*	$x_1,...,x_5,x_{10},x_{12},x_{13},x_{19},x_{20}$	1 hora e 20 minutos	0.30 (0.02)
Forward Step.	$x_1,...,x_5,x_{10},x_{12},x_{13},x_{19},x_{20}$	0.46 segundos	0.30 (0.02)
Ridge	Todas	0.16 segundos	0.33 (0.03)
Lasso	$x_1,...,x_5$	0.08 segundos	0.25 (0.02)

[a] Risco Estimado (Erro Padrão)
[*] Busca pelo melhor subconjunto

□

Exemplo 3.2 (Câncer de Próstata). Esses dados são provenientes de um estudo publicado em Stamey et al. (1989) e estudados por James et al. (2013). Nesse trabalho os autores investigaram a associação entre o nível de antígeno especifico de próstata (PSA) e algumas medidas clínicas. Foram observados dados de 97 homens que estavam para fazer prostatectomia radical. O objetivo era prever o logaritmo da PSA (lpsa) com base nas demais variáveis. A seguir são apresentadas as variáveis para algumas observações do banco de dados.

Tabela 3.3: Banco de dados de Câncer de Próstata.

lcavol	**lweight**	**age**	**lbph**	**svi**	**lcp**	**gleason**	**pgg45**	**lpsa**	**train**
0.75	3.43	62	-1.386	0	-1.39	6	0	0.37	TRUE
0.46	3.76	49	1.423	0	-1.39	6	0	2.79	TRUE
-1.35	3.60	63	1.267	0	-1.39	6	0	1.27	TRUE
1.31	4.12	64	2.171	0	-1.39	7	5	2.09	FALSE
2.88	3.77	68	1.558	1	1.56	7	80	5.48	TRUE
0.69	3.54	58	1.537	0	-1.39	6	0	0.85	TRUE

Utilizamos a separação dos dados em treinamento e validação definida no próprio banco de dados (coluna *train*) para aplicação das técnicas de mínimos quadrados, ridge e lasso.

```
set.seed(1)

dados <- read.table("../dados/prostate.data")

# conjunto de treinamento
tr <- dados$train

X_tr <- dados[tr, 1:8] %>% as.matrix()
y_tr <- dados$lpsa[tr]

# conjunto de validação
X_val <- dados[-tr, 1:8] %>% as.matrix()
y_val <- dados$lpsa[-tr]
```

A seguir iremos ajustar o modelo de mínimos quadrados, ridge e lasso.

```
# mínimos quadrados
ajuste_mq <- glmnet(X_tr, y_tr, alpha = 0, lambda = 0)
predito_mq <- predict(ajuste_mq, newx = X_val)

# regressão ridge
cv_ridge <- cv.glmnet(X_tr, y_tr, alpha = 0)
ajuste_ridge <- glmnet(X_tr, y_tr, alpha = 0)
predito_ridge <- predict(ajuste_ridge,
                         s = cv_ridge$lambda.1se,
                         newx = X_val)

# regressão lasso
cv_lasso <- cv.glmnet(X_tr, y_tr, alpha = 1)
ajuste_lasso <- glmnet(X_tr, y_tr, alpha = 1)
predito_lasso <- predict(ajuste_lasso,
                         s = cv_lasso$lambda.1se,
```

```
                newx = X_val)
```

As Figuras 3.4 e 3.5 ilustram o comportamento dos coeficientes estimados de acordo com λ para a regressão ridge e o lasso.

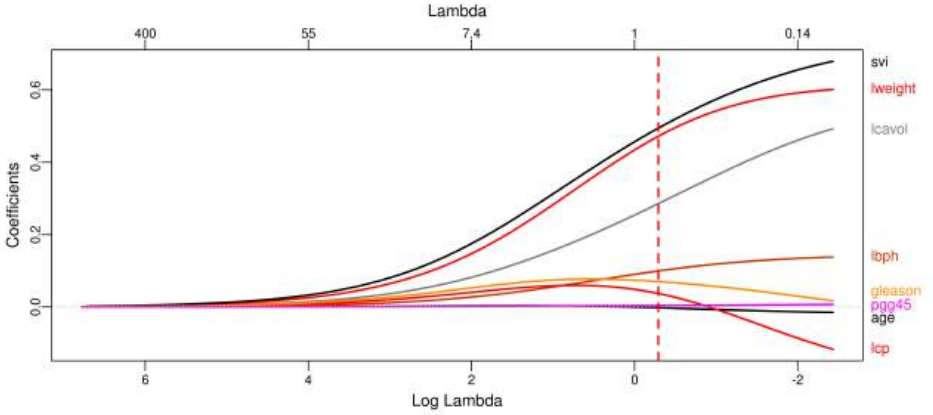


Figura 3.4: Coeficientes estimados via regressão ridge em função de λ. A reta vertical indica o valor escolhido via validação cruzada.

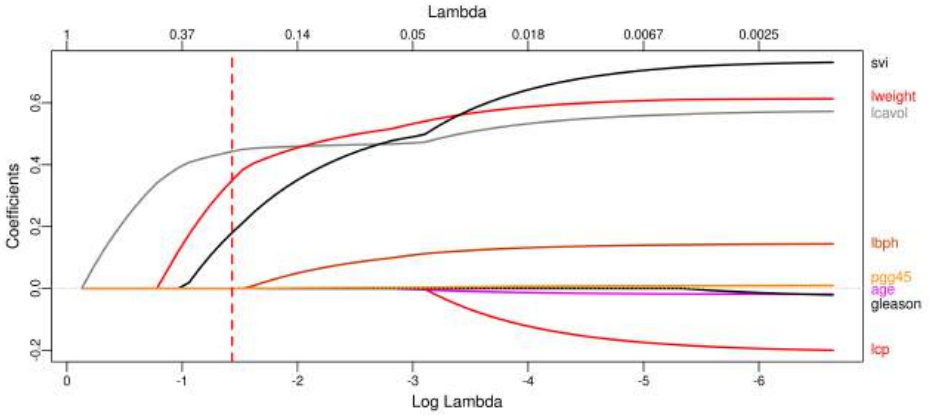


Figura 3.5: Coeficientes estimados via regressão lasso em função de λ. A reta vertical indica o valor escolhido via validação cruzada.

A Tabela 3.4 apresenta os coeficientes estimados com a utilização das três técnicas. Note a redução dos coeficientes obtidos por mínimos quadrados com estimação dos métodos ridge e lasso. Ainda, note que excluímos algumas variáveis com a utilização do lasso.

Tabela 3.4: Coeficientes estimados para os dados da próstata considerando mínimos quadrados, ridge e lasso.

Coeficiente	**Mínimos Quadrados**	**Ridge**	**Lasso**
Intercepto	0.428	0.096	0.164
lcavol	0.577	0.493	0.544
lweight	0.614	0.601	0.596
age	-0.019	-0.015	-0.015
lbph	0.145	0.138	0.135
svi	0.737	0.679	0.669
lcp	-0.206	-0.117	-0.145
gleason	-0.029	0.017	-
pgg45	0.009	0.007	0.007

A Tabela 3.5 apresenta o desempenho preditivo dos modelos obtidos com os dados de treinamento quando aplicados aos dados de validação dos 30 indivíduos. Levando em conta o erro padrão dos estimadores (que são grandes por conta do tamanho pequeno do conjunto de validação), não é possível afirmar qual modelo teve desempenho preditivo melhor. Contudo, o lasso possui a vantagem de não utilizar algumas covariáveis.

Tabela 3.5: Desempenho preditivo (erro quadrático médico e seu erro padrão) em relação ao conjunto de validação dos modelos de mínimos quadrados, ridge e lasso.

Modelo	**EQM**	**Erro Padrão**
Mínimos Quadrados	0.455	0.078
Ridge	0.516	0.079
Lasso	0.560	0.088

□

Exemplo 3.3 (Amazon Fine Food Reviews). Nesse exemplo, a partir dos textos das resenhas, mostramos como fazer predições de notas dadas na Amazon. Para isso, selecionaremos um subconjunto do banco de dados Amazon Fine Food Reviews. Esse banco conta com aproximadamente meio milhão de resenhas. Para ilustração das técnicas apresentadas anteriormente, separamos 20.000 observações. Dessas, 10.000 foram utilizadas para treinamento dos modelos e 10.000 para validação.

```
dados <- fread("../dados/Reviews.csv", header = TRUE)
```

```
# seleciona 20.000 observações
set.seed(1)

selecao <- sample(nrow(dados), 20000)
dados   <- dados[selecao,]

# cria a matriz de preditoras

corp <- VCorpus(VectorSource(dados$Text))

dtm <- DocumentTermMatrix(corp,
        control = list(tolower               = TRUE,
                       stemming              = FALSE,
                       removeNumbers         = TRUE,
                       removePunctuation     = TRUE,
                       removeStripwhitespace = TRUE,
                       weighting             = weightTf))

dtmMatrix <- sparseMatrix(i = dtm$i, j = dtm$j, x = dtm$v,
                dimnames = list(NULL, dtm$dimnames[[2]]),
                dims = c(dtm$nrow, dtm$ncol))

dim(dtmMatrix)

## [1] 20000 36841

# indica observações de treinamento
```

```
tr <- sample.int(length(selecao), 10000, replace = FALSE)
```

Assim, consideraremos nesse exemplo um total de 17.072 termos. É importante notar que não é necessário converter a matriz de dados para o formato usual com todos os elementos preenchidos. Ela pode ser utilizada de forma esparsa (veja a Seção A.3 para mais detalhes) e, assim, economizar memória do computador. Com a utilização do formato esparso, é possível trabalhar com o conjunto completo de meio milhão de resenhas.

A seguir iremos ajustar o modelo de mínimos quadrados, ridge e lasso.

```
# mínimos quadrados
ajuste_mq <- glmnet(dtmMatrix[tr,],
                    dados$Score[tr],
                    alpha = 0, lambda = 0)

predito_mq <- predict(ajuste_mq, newx = dtmMatrix[-tr,])

# ridge
ridge <- cv.glmnet(dtmMatrix[tr,], dados$Score[tr], alpha = 0)

predito_ridge <- predict(ridge,
                         s = ridge$lambda.min,
                         newx = dtmMatrix[-tr,])

# lasso
lasso <- cv.glmnet(dtmMatrix[tr,], dados$Score[tr], alpha = 1)

predito_lasso <- predict(lasso,
                         s = lasso$lambda.min,
                         newx = dtmMatrix[-tr,])
```

Comparando os erros quadráticos médios (Tabela 3.6), notamos que o lasso apresentou o melhor desempenho preditivo, seguida da regressão ridge. Já a regressão com mínimos quadrados apresentou um desempenho bastante inferior à essas duas técnicas. Além disso, o lasso seleciona apenas 558 variáveis nesse caso.

Tabela 3.6: Desempenho preditivo (erro quadrático médico e seu erro padrão) em relação ao conjunto de validação dos modelos de mínimos quadrados, ridge e lasso.

Modelo	**EQM**	**Erro Padrão**
Mínimos Quadrados	7.675	0.234
Ridge	1.205	0.024
Lasso	1.136	0.022

As Figuras 3.6 e 3.7 mostram quais palavras apresentam maior importância para a previsão. Consideramos os valores positivos e negativos dos coeficientes estimados ordenados. Para esse exemplo, selecionamos os 20 maiores de cada sentido para a regressão ridge e lasso.

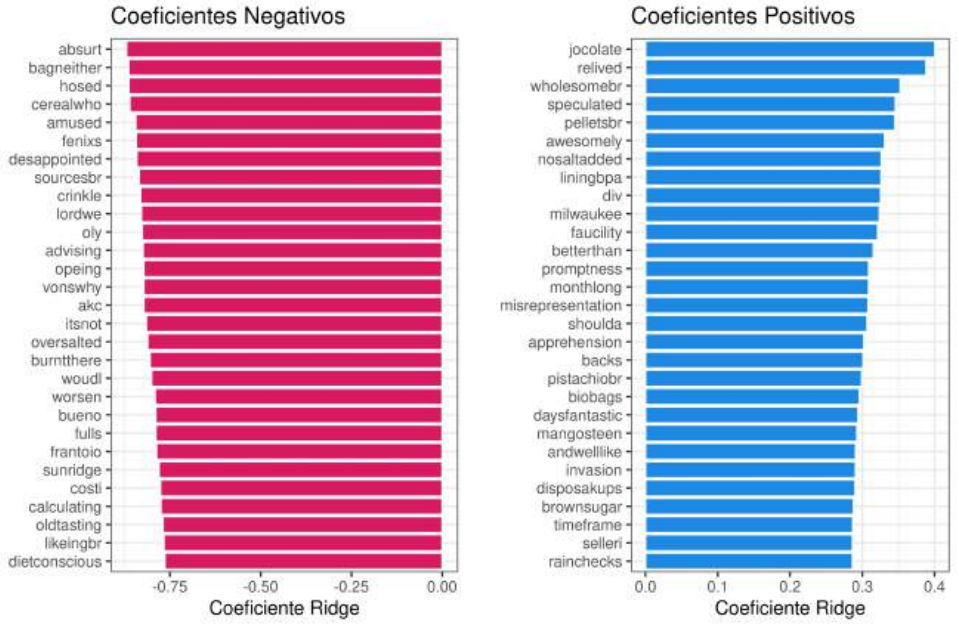


Figura 3.6: Coeficientes com os 20 maiores valores da regressão ridge.

□

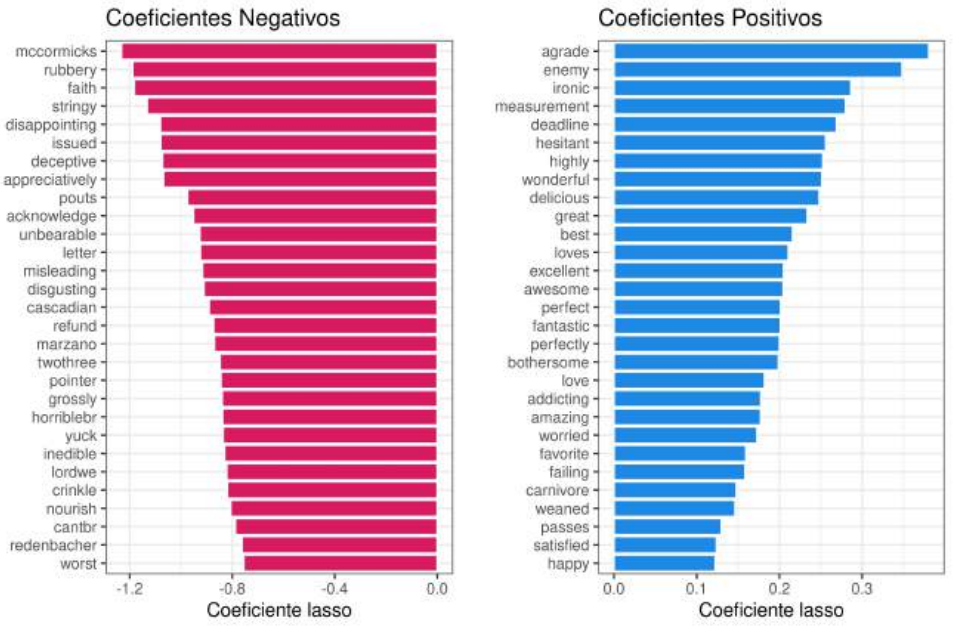


Figura 3.7: Coeficientes com os 20 maiores valores da regressão lasso.

3.9 Resumo

Neste capítulo vimos que, mesmo quando há muitas covariáveis em relação ao tamanho amostral, muitas vezes é possível criar bons estimadores de $r(\mathbf{x})$ com base em uma regressão linear. A chave para isso é usar o fato de que, frequentemente, muitas variáveis têm pouca influência na resposta. Assim, adicionando um termo de penalização à soma de quadrados (Eq. 3.1), criamos estimadores dos coeficientes β que encorajam *esparsidade*, isto é, muitos zeros na solução. Desta forma, criamos estimadores com uma variância menor. A esperança é que, se o aumento no viés por retirar essas variáveis não é muito grande, o risco também diminui e, assim, conseguimos um estimador melhor para $r(\mathbf{x})$ do que aquele obtido pelo método dos mínimos quadrados (ver Figura 3.8).

Vimos três formas de se fazer a penalização:

- **Penalidade $\mathbf{L_0}$**: $\sum_{i=1}^{d} \mathbb{I}(\beta_i \neq 0)$. Encontrar o mínimo da função objetivo consome um tempo computacional extremamente alto, pois deve-se buscar o melhor entre os 2^d subconjuntos de covariáveis. Assim, é necessário usar uma

heurística como *stepwise* para aproximar este valor.

- **Penalidade L_1**: $\sum_{i=1}^{d} |\beta_i|$. Trata-se do *lasso*. Esta penalização é rápida de ser calculada e induz esparsidade em β. Frequentemente ela leva a um estimador com bom poder preditivo.
- **Penalidade L_2**: $\sum_{i=1}^{d} \beta_i^2$. Trata-se da *regressão ridge*. Esta penalização não induz zeros em β, mas reduz a variância do estimador da regressão pois "encolhe" o estimador de mínimos quadrados.

Métodos paramétricos muitas vezes impõem muitas limitações para $r(\mathbf{x})$: por exemplo, nem sempre o melhor estimador linear é um bom estimador para $r(\mathbf{x})$. No próximo capítulo iremos introduzir métodos não paramétricos para $r(\mathbf{x})$, que são modelos mais flexíveis que métodos paramétricos. Em linhas gerais, métodos não paramétricos diminuem o viés e aumentam a variância de métodos paramétricos, como indica a Figura 3.8.

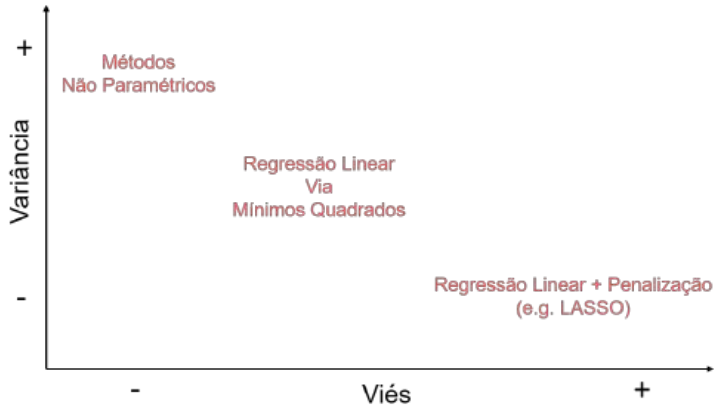


Figura 3.8: Relação entre métodos não paramétricos, sem penalização e métodos paramétricos com penalização.

Capítulo 4

Métodos Não Paramétricos

> With four parameters I can fit an elephant, and with five I can make him wiggle his trunk.
>
> John von Neumann

Para amostras pequenas, métodos paramétricos costumam levar a bons resultados: impondo-se um modelo com poucos parâmetros, é possível criar estimadores com baixa variância para situações com tamanho amostral (n) pequeno. Contudo, quando n é grande, muitas vezes é benéfico aumentar o número de parâmetros do modelo, de modo que tenhamos um viés consideravelmente menor às custas de uma variância um pouco maior. É exatamente por isso que métodos não paramétricos ganham importância: informalmente, um modelo não paramétrico é um modelo que tem *infinitos parâmetros*. Neste capítulo exploramos alguns destes modelos.

4.1 Séries Ortogonais

Métodos baseados em séries ortogonais são bastante antigos (Chentsov, 1962) e baseiam-se em uma ideia bastante simples: expandir a função de regressão em uma base ortogonal. Para facilitar a discussão, vamos assumir por hora que $\mathbf{x} \in [0,1]$ e a densidade de $\mathbf{X}$ é uniform, isto é, $f(\mathbf{x}) = 1$.

O ponto inicial é escolher $(\psi_j(\mathbf{x}))_{j\in\mathbb{N}}$, uma base ortonormal[1] para o conjunto de

[1] Isto é, $\int_0^1 \psi_i(\mathbf{x})\psi_j(\mathbf{x}) = \mathbb{I}(i=j)$.

funções

$$L^2([0,1]) := \left\{ f : [0,1] \longrightarrow \mathbb{R} : \int_0^1 f^2(\mathbf{x}) d\mathbf{x} < \infty \right\}.$$

Um exemplo de $(\psi_j(\mathbf{x}))_{j\in\mathbb{N}}$ é a *base de Fourier*:

$$\psi_0(\mathbf{x}) = 1; \qquad \psi_{2j-1}(\mathbf{x}) = \sqrt{2}\sin(2\pi j\mathbf{x}),\ j \in \mathbb{N}; \qquad \psi_{2j}(\mathbf{x}) = \sqrt{2}\cos(2\pi j\mathbf{x}),\ j \in \mathbb{N}$$

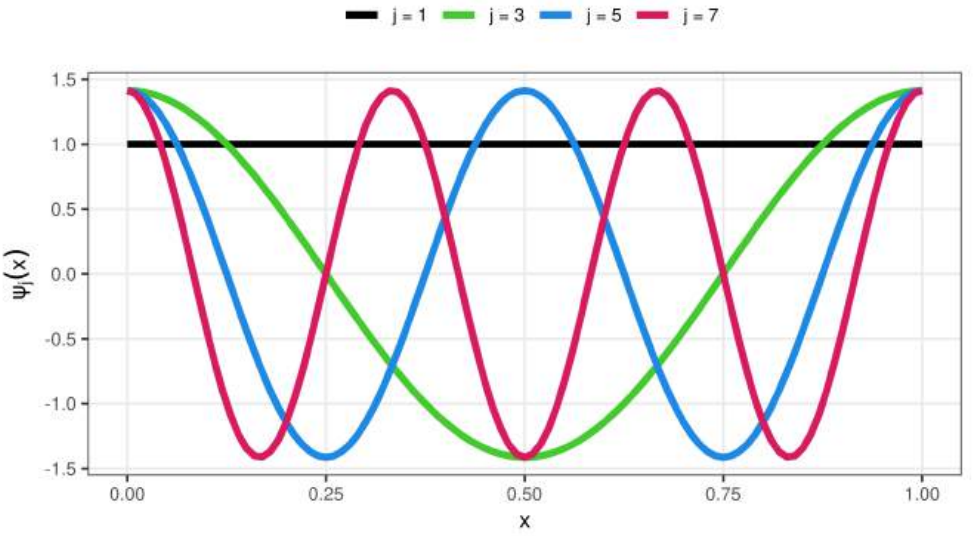


Figura 4.1: Alguns elementos da base de Fourier. Os primeiros termos são mais suaves que os termos de ordem mais alta. Qualquer função integrável em L^2 pode ser escrita como combinação linear dos elementos da base de Fourier.

A ideia central de métodos baseados em séries é que, se a função de regressão $r(\mathbf{x})$ pertence a $L^2([0,1])$, então $r(\mathbf{x})$ pode ser representada como

$$r(\mathbf{x}) = \sum_{j\in\mathbb{N}} \beta_j \psi_j(\mathbf{x}),$$

em que, devido à ortonormalidade da base $(\psi_j(\mathbf{x}))_{j\in\mathbb{N}}$, temos que os coeficientes da expansão admitem uma forma muito simples:

$$\beta_j = \int r(\mathbf{x})\psi_j(\mathbf{x})d\mathbf{x} = \int \mathbb{E}[Y|\mathbf{x}]\psi_j(\mathbf{x})f(\mathbf{x})d\mathbf{x} = \int \mathbb{E}[Y\psi_j(\mathbf{x})|\mathbf{x}]f(\mathbf{x})d\mathbf{x} = \mathbb{E}[Y\psi_j(\mathbf{X})].$$

Um estimador não viesado para os *coeficientes da expansão* β_j é dado por

$$\widehat{\beta}_j = \frac{1}{n}\sum_{i=1}^{n} Y_i \psi_j(\mathbf{X}_i).$$

Assim, o estimador baseado em séries tem a seguinte forma:

$$g(\mathbf{x}) = \sum_{j=0}^{J} \widehat{\beta}_j \psi_j(\mathbf{x}). \tag{4.1}$$

O parâmetro J é um *tuning parameter* que controla o balanço viés-variância. Valores altos de J levam a um viés baixo, mas variância alta.

Observação 4.1. Se $r(\mathbf{x})$ pertence a $L^2([0,1])$, sabemos que $\beta_j \longrightarrow 0$ quando $j \longrightarrow \infty$, o que justifica o truncamento da Equação 4.1.

□

Note que o número de parâmetros, $(\beta_j)_{j\in\mathbb{N}}$, é infinito, por isso o método de séries é considerado um método não paramétrico.

Se o número de preditoras for maior do que 1 ($d > 1$), construir uma base $\left\{\psi_j(\mathbf{x})\right\}_j$ é uma tarefa menos simples. Tipicamente, isso é feito usando-se *d produtos tensoriais* para cada uma das coordenadas de $\mathbf{x}$. Por exemplo, se $d = 2$, a base dada por produtos tensoriais é

$$\left\{\psi_{i,j}(\mathbf{x}) = \psi_i(x_1)\psi_j(x_2) : i, j \in \mathbb{N}\right\},$$

em que $\mathbf{x} = (x_1, x_2)$, e $\{\psi_i(\mathbf{x}_1)\}_i$ e $\left\{\psi_j(\mathbf{x}_2)\right\}_j$ são bases para $\mathfrak{L}^2(\mathbb{R})$. Esta é uma base para $\mathfrak{L}^2(\mathbb{R}^2)$ (Wasserman, 2006).

A abordagem de séries ortogonais é essencialmente a mesma que aquela da regressão linear com um número variável de covariáveis, com a exceção de que o estimador de cada coeficiente β_i é mais rapidamente calculado devido à ortogonalidade das funções de base.

Para deixar essa conexão mais clara, considere o exemplo a seguir. Pela expansão de Taylor, sabemos que qualquer função analítica (Krantz e Parks, 2002) pode ser expressa como

$$r(x) = \sum_{i\geq 0} \beta_i x^i.$$

Essa é justamente a família de modelos usada no Exemplo 1.6. Assim, o modelo ajustado naquele exemplo foi um modelo não paramétrico, embora os coeficientes da expansão tenham sido ajustados via regressão linear. O que torna o modelo não paramétrico é justamente o fato do número de coeficientes usados, J (que naquele exemplo é denotado por p) ser um *tuning parameter* que pode assumir qualquer valor em $\{0,1,\dots\}$. Assim, há um número infinito de parâmetros[2][3].

Para outras abordagens de estimação com séries ortogonais, veja Beran (2000), Efromovich (1999) e Wasserman (2006). Em particular, tais referências discutem o uso de outras bases além da base de Fourier, como, por exemplo, wavelets. Veja também a Seção 4.13 para um pouco de teoria sobre estimadores baseados em séries.

O método de séries pode ser implementado usando-se uma base polinomial via função `lm`, descrita no Capítulo 2:

```
ajuste <- lm(y ~ poly(as.matrix(covariaveis), degree = 5))
```

A função `poly` calcula automaticamente todos os polinômios (neste caso com grau até 5) em $\mathbb{R}^d$.

4.2 Splines

Vamos assumir novamente que $d = 1$. Assim como métodos baseados em séries ortogonais, em splines aproximamos $r(\mathbf{x})$ como uma combinação linear $\sum_{i=1}^{I} \beta_i f_i(\mathbf{x})$. Contudo, ao invés de usar uma base ortonormal, utiliza-se uma base para um espaço de *splines*.

Para descrevermos o que é um spline, fixe $t_1 < \dots < t_p$. Chamamos tais pontos de *nós*. Um spline de k-ésima ordem é uma função polinomial por partes de grau k que é contínua e tem derivadas de ordem $1,\dots,k-1$ contínuas em seus nós. Como um spline é bem mais suave que uma função que é polinomial por partes (mas sem as restrições de continuidade), ele consegue aproximar de forma mais eficiente funções contínuas[4].

[2] Note que quando escolhemos $J = J_0$, estamos estimando $\beta_j = 0, \forall j > J_0$.

[3] Essa base de polinômios não é ortonormal, mas ela pode ser convertida em uma base ortonormal via o processo de ortogonalização de Gram-Schmidt. A base resultante é chamada de base de Hermite (Efromovich, 1999).

[4] Por exemplo, se r é uma função duas vezes diferenciável em $[0,1]$, então existe um spline cúbico f com nós em $t_1 < \dots < t_p$ tal que $\sup_{\mathbf{x}\in[0,1]} |r(\mathbf{x}) - f(\mathbf{x})| \leq O\left(\frac{1}{p}\sqrt{\int_0^1 r''(\mathbf{x})^2 d\mathbf{x}}\right)$ (De Boor et al., 1978).

Há diversas formas de se definir uma base para tal espaço. Uma delas é através das seguintes funções:

$$f_1(x) = 1;\ f_2(x) = x; \ldots; f_{k+1}(x) = x^k,$$
$$f_{k+1+j}(x) = (x - t_j)_+^k,\ j = 1, \ldots, p$$

Dessa forma, com essa base, $I = k + 1 + p$. Há, contudo, muitas outras formas de definir uma base para este espaço. Existem também muitas variações de splines, como natural splines ou smoothing splines (veja, por exemplo, Hastie et al. 2001 e a Seção 4.6.3.1).

Fixados os nós, pode-se então utilizar o método de mínimos quadrados para estimar os valores dos coeficientes $\beta_1, \ldots, \beta_I$. No R, splines podem ser usados via a biblioteca `splines`:

```
fit <- lm(y ~ bs(x, degree = 3), data = dados)
predito <- predict(fit, newdata = list(x = x.grid), se = TRUE)
```

Este código ajusta *B*-splines. Utilizando-se `ns`, pode-se ajustar natural splines da mesma maneira. Para smoothing splines, pode-se usar

```
fit <- smooth.spline(x, y, df = 3)
```

4.3 k Vizinhos Mais Próximos

> We're neighbors and we're going to pull together.
>
> Rick Perry

O método dos k vizinhos mais próximos (em inglês, *k-nearest neighbours*, KNN) (Benedetti, 1977; Stone, 1977) é um dos mais populares na comunidade de aprendizado de máquina. Ele tem como base estimar a função de regressão $r(\mathbf{x})$ para uma dada configuração das covariáveis $\mathbf{x}$ com base nas respostas Y dos k-vizinhos mais próximos a $\mathbf{x}$. Formalmente, definimos

$$g(\mathbf{x}) = \frac{1}{k} \sum_{i \in \mathcal{N}_\mathbf{x}} y_i, \tag{4.2}$$

em que $\mathcal{N}_{\mathbf{x}}$ é o conjunto das k observações mais próximas de $\mathbf{x}$, isto é,

$$\mathcal{N}_{\mathbf{x}} = \left\{ i \in \{1, \ldots, n\} : d(\mathbf{x}_i, \mathbf{x}) \leq d_{\mathbf{x}}^k \right\}$$

e $d_{\mathbf{x}}^k$ é a distância do k-ésimo vizinho mais próximo de $\mathbf{x}$ a $\mathbf{x}$. Em palavras, a função de regressão avaliada em $\mathbf{x}$ é estimada utilizando-se uma *média local* das respostas dos k vizinhos mais próximos a $\mathbf{x}$ no espaço das covariáveis.

O *tuning parameter* k pode ser escolhido por validação cruzada. Um valor alto de k leva a um modelo muito simples (uma constante quando $k \longrightarrow \infty$) e, assim um viés alto, mas uma variância baixa. Por sua vez, um valor baixo para k leva a um estimador com variância alta, mas viés baixo. Veja a Figura 4.2 para uma ilustração.

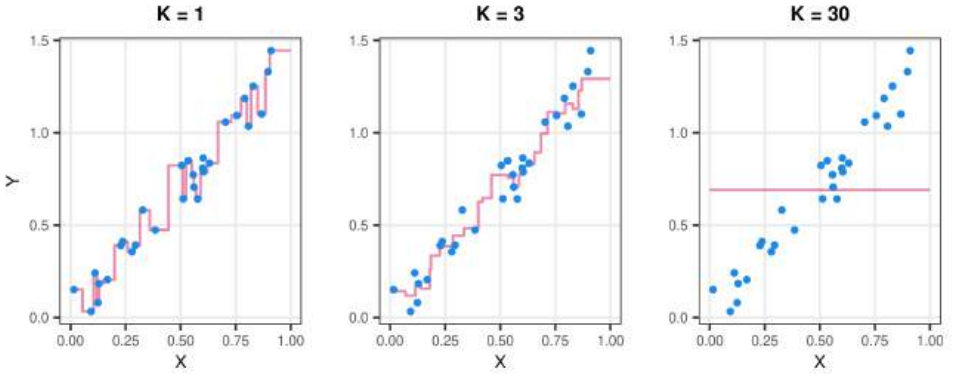


Figura 4.2: Influência da escolha de k no estimador dos k vizinhos mais próximos.

No R, o método dos vizinhos mais próximos pode ser implementando utilizando o pacote `FNN`. A função `knn.reg` faz a predição para novas observações com covariáveis `x_teste` (uma observação por linha) utilizando um conjunto de treinamento com covariáveis `x_treinamento`, respostas `y_treinamento` e `k` vizinhos da seguinte forma:

```
library(FNN)

ajuste <- knn.reg(train = x_treinamento, test = x_teste,
                  y = y_treinamento, k = k)
```

```
predito <- ajuste$pred
```

4.4 Nadaraya-Watson

O método de Nadaraya-Watson, criado por Nadaraya (1964) e Watson (1964), é uma variação do método dos k-vizinhos mais próximos bastante difundida na comunidade de estatística. Este método consiste em estimar a função de regressão em um dado ponto $\mathbf{x}$ utilizando-se uma média ponderada das observações do conjunto de treinamento:

$$g(\mathbf{x}) = \sum_{i=1}^{n} w_i(\mathbf{x}) y_i,$$

em que $w_i(\mathbf{x})$ é um peso atribuído à i-ésima observação, e que mede o quão similar $\mathbf{x}_i$ é a $\mathbf{x}$. Mais especificamente, $w_i(\mathbf{x})$ tem a forma

$$w_i(\mathbf{x}) = \frac{K(\mathbf{x}, \mathbf{x}_i)}{\sum_{j=1}^{n} K(\mathbf{x}, \mathbf{x}_j)},$$

em que $K(\mathbf{x}, \mathbf{x}_i)$ é um *kernel de suavização* usado para medir a similaridade entre as observações. Algumas escolhas populares para $K(\mathbf{x}, \mathbf{x}_i)$ são:

- **kernel uniforme:** $K(\mathbf{x}, \mathbf{x}_i) = \mathbb{I}(d(\mathbf{x}, \mathbf{x}_i) \leq h)$
- **kernel gaussiano:** $K(\mathbf{x}, \mathbf{x}_i) = (\sqrt{2\pi h^2})^{-1} \exp\left\{ -\frac{d^2(\mathbf{x}, \mathbf{x}_i)}{2h^2} \right\}$
- **kernel triangular:** $K(\mathbf{x}, \mathbf{x}_i) = (1 - d(\mathbf{x}, \mathbf{x}_i)/h)\mathbb{I}(d(\mathbf{x}, \mathbf{x}_i) \leq h)$
- **kernel de Epanechnikov:** $K(\mathbf{x}, \mathbf{x}_i) = (1 - d^2(\mathbf{x}, \mathbf{x}_i)/h^2)\mathbb{I}(d(\mathbf{x}, \mathbf{x}_i) \leq h)$

Enquanto o kernel uniforme atribui o mesmo peso para cada observação a uma distância menor que h de $\mathbf{x}$ e peso zero para observações a uma distância maior que h (que é um *tuning parameter*), os kernels triangulares e de Epanechnikov atribuem pesos diferentes de acordo com a distância até $\mathbf{x}$; observações mais próximas recebem peso maior. O kernel gaussiano, por sua vez, atribui peso positivo para *todas* as observações do conjunto de treinamento (novamente, observações mais próximas recebem peso maior). Outros exemplos de kernels podem ser encontrados em https://bit.ly/32OSvUl.

Note que um valor alto do *tuning parameter* h leva um estimador da função de regressão com variância baixa e viés alto, já que o mesmo peso é atribuído para cada amostra $\mathbf{x}_i$ neste caso. Por sua vez, h baixo leva a uma estimador com variância alta, mas viés baixo. Veja a Figura 4.3 para um exemplo com o kernel uniforme.

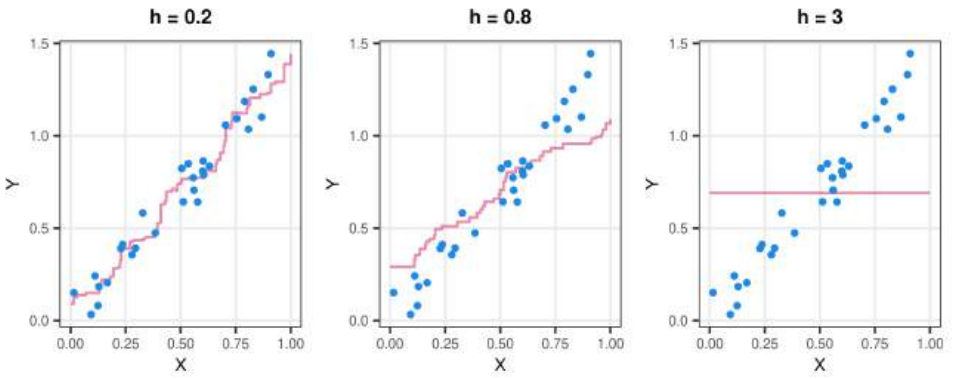


Figura 4.3: Influência da escolha de h no estimador de Nadaraya-Watson. Aqui o kernel utilizado foi o kernel uniforme.

Na prática, muitas vezes é observado que a escolha do kernel não influencia muito os resultados, enquanto a escolha dos *tuning parameters* associados a ele sim. Veja um exemplo de comparação de dois kernels na Figura 4.4.

Pode-se verificar que o estimador de Nadaraya Watson em um dado $\mathbf{x}$ fixo é justamente o ponto β_0 que minimiza

$$\sum_{i=1}^{n} w_i(\mathbf{x}) \left(Y_i - \beta_0\right)^2,$$

isto é,

$$g(\mathbf{x}) := \widehat{\beta}_0 = \arg\min_{\beta_0} \sum_{i=1}^{n} w_i(\mathbf{x}) \left(Y_i - \beta_0\right)^2, \tag{4.3}$$

Assim, esse estimador pode ser entendido como um estimador de mínimos quadrados *ponderado*; no qual consideramos uma função de regressão que contém somente o intercepto, $g(\mathbf{x}) = \widehat{\beta}_0$. Note que $\mathbf{x}$ está fixo; para cada $\mathbf{x}$ temos um valor de $\widehat{\beta}_0$ diferente, isto é, uma outra regressão linear. Esta motivação para o estimador de

Nadaraya-Watson leva a uma extensão deste método conhecida como regressão polinomial local, que descrevemos na Seção 4.5.

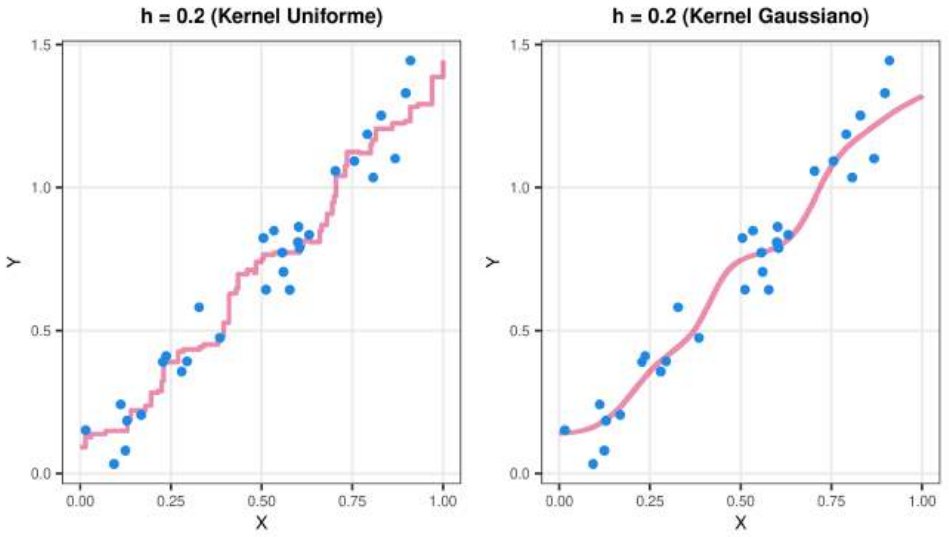


Figura 4.4: Influência da escolha do kernel no estimador de Nadaraya-Watson.

Uma forma de se ajustar o estimador de Nadaraya-Watson é utilizando o pacote `locfit`. Pode-se mudar o kernel utilizado via o argumento `kern`.

```
ajuste <- locfit(y ~ x_1 + x_2, alpha = c(0, 0.3),
                 deg = 0, data = dados)

predito <- predict(ajuste, newdata = novos_dados)
```

4.5 Regressão Polinomial Local

A regressão polinomial local (Cleveland, 1979; Fan, 1993; Fan e Gijbels, 1996; Stone, 1977) é uma extensão do estimador de Nadaraya-Watson. Suponha, por simplicidade, que há apenas $d = 1$ covariável. A ideia central é que, ao invés de buscar um estimador baseado no método de mínimos quadrados ponderados com apenas

um intercepto como na Equação 4.3, utilizamos um polinômio de grau p:

$$g(\mathbf{x}) := \widehat{\beta}_0 + \sum_{j=1}^{p} \widehat{\beta}_j \mathbf{x}^j,$$

em que $\widehat{\beta}_0, \ldots, \widehat{\beta}_p$ são dados por

$$\arg\min_{\beta_0,\ldots,\beta_p} \sum_{i=1}^{n} w_i(\mathbf{x}) \left(Y_i - \beta_0 - \sum_{j=1}^{p} \beta_j \mathbf{x}_i^j \right)^2.$$

Aqui, novamente, $w_i(\mathbf{x}) = \frac{K(\mathbf{x},\mathbf{x}_i)}{\sum_{j=1}^{n} K(\mathbf{x},\mathbf{x}_j)}$. Note que, para cada $\mathbf{x}$ de interesse, as estimativas dos coeficientes $\beta_0, \ldots, \beta_p$ são diferentes, ou seja esse estimador consiste em usar o método de mínimos quadrados *localmente*. A solução para tal problema é dada por

$$(\widehat{\beta}_0 \ \ldots \ \widehat{\beta}_p) = (B^\intercal \Omega B)^{-1} B^\intercal \Omega y,$$

em que B é uma matriz $n \times (p+1)$ cuja i-ésima linha é dada por $(1, \mathbf{x}_i, \ldots, \mathbf{x}_i^p)$ e Ω é uma matriz diagonal $n \times n$ cujo i-ésimo elemento da diagonal é $w_i(\mathbf{x})$.

Uma vantagem de se utilizar polinômios ao invés de apenas um intercepto é que isso induz estimadores com vieses menores (em particular, o viés em regiões próximas ao limite do domínio dos dados; veja, por exemplo, Wasserman 2006). Por outro lado, a variância de tais estimadores é maior, de modo que é importante selecionar o grau do polinômio a ser utilizado adequadamente. Para regressão polinomial local com mais de uma covariável (ou seja, $d > 1$) remetemos o leitor a Fan e Gijbels (1996).

No R, polinômios locais também podem ser ajustados no R usando o pacote `locfit`, como o estimador de Nadaraya-Watson (Seção 4.4). Para isso, basta modificar o valor do argumento `deg`, que representa o grau do polinômio a ser utilizado. Note que 0 corresponde ao estimador de Nadaraya-Watson.

4.6 Métodos baseados em RKHSs

Métodos de estimação da função de regressão com base em RKHSs (*Reproducing Kernel Hilbert Spaces*, Aronszajn 1950) (Hastie et al., 2001; Kimeldorf e Wahba, 1970; Nosedal-Sanchez et al., 2012) são uma família de métodos bastante geral. A ideia

central envolve a definição de uma função objetivo para quantificar a qualidade de uma função de predição e, posteriormente, busca-se a função com melhor ajuste em um espaço de funções $\mathcal{H}$. Assim, busca-se pela solução de

$$\arg\min_{g\in\mathcal{H}} \sum_{k=1}^{n} L(g(\mathbf{x}_k), y_k) + \mathcal{P}(g),$$

em que L é uma função de perda arbitrária, $\mathcal{P}$ é uma medida de complexidade de g e $\mathcal{H}$ é um espaço de funções. Para um espaço $\mathcal{H}$ arbitrário, a solução para esse problema é difícil, uma vez que se trata de um problema de otimização sobre um espaço de funções. RKHSs permitem descrever uma grande família de espaços $\mathcal{H}$ (chamadas de Reproducing Kernel Hilbert Spaces) de modo que a solução do problema de otimização seja relativamente simples de ser implementada.

A fim de motivar esta família, consideraremos uma pequena modificação do problema de minimização proposto na regressão ridge (Seção 3.4) dada por:

$$\widehat{\beta} = \arg\min_{\beta} \sum_{k=1}^{n} (y_k - \beta_0 - \beta_1 x_{k,1} - \ldots - \beta_d x_{k,d})^2 + \lambda \sum_{i=0}^{d} \beta_i^2, \tag{4.4}$$

em que β_0 também é penalizado para simplificar a notação do que será exposto. Denotando a i-ésima variável de $\mathbf{x}$ por $\phi_i(\mathbf{x})$ (considerando a convenção de que $\phi_0(\mathbf{x}) \equiv x_0 \equiv 1$), esse problema pode ser reformulado como

$$\widehat{r}(\mathbf{x}) = \arg\min_{g\in\mathcal{H}} \sum_{k=1}^{n} (y_k - g(\mathbf{x}_k))^2 + \lambda ||g||_{\mathcal{H}}^2, \tag{4.5}$$

em que

$$\mathcal{H} := \left\{ g \in L^2(\mathcal{X}) : \text{ existem } (c_i)_{i=0}^{d} \text{ com } \sum_{i=0}^{d} c_i^2 < \infty \text{ tais que } g(\mathbf{x}) = \sum_{i=0}^{d} c_i \phi_i(\mathbf{x}) \right\} \tag{4.6}$$

e

$$||g||_{\mathcal{H}}^2 := \sum_{i=0}^{d} c_i^2.$$

Enquanto que a Equação 4.4 consistem em um problema de otimização sobre vetores de $\mathbb{R}^{d+1}$, a Equação 4.5 consistem em um problema de otimização sobre *funções*. Note que enquanto o termo $\sum_{k=1}^{n} (y_k - g(\mathbf{x}_k))^2$ mede a bondade de ajuste de g,

o termo $||g||^2_{\mathcal{H}}$ mede a *suavidade* de g: se $||g||^2_{\mathcal{H}}$ é baixo, temos g suave, caso contrário g oscila muito. Em outras palavras, $||g||^2_{\mathcal{H}}$ alto implica que uma mudança pequena em $\mathbf{x}$ acarreta em mudança grande em $g(\mathbf{x})$.

Métodos de regressão com base em penalização em RKHS são uma generalização da função objetivo da Equação 4.5. Primeiramente, eles permite que a bondade do ajuste possa ser medida com outras funções além do erro quadrático. Em segundo lugar, ao invés de usarem apenas este espaço específico $\mathcal{H}$ (funções lineares), métodos baseados em RKHS consideram espaços mais gerais $\mathcal{H}$ em que a Equação 4.5 ainda pode ser resolvida facilmente. Isso permite criar uma classe mais ampla de estimadores com a possibilidade de bom desempenho em uma grande variedade de situações. Esses espaços são justamente os *Reproducing Kernel Hilbert Spaces*.

Mais especificamente, buscaremos resolver o problema

$$\arg\min_{g \in \mathcal{H}} \sum_{k=1}^{n} L(g(\mathbf{x}_k), y_k) + \lambda ||g||^2_{\mathcal{H}},$$

em que $\mathcal{H}$ é um RKHS e L é uma função adequada para o problema em questão. Veremos que a norma $||g||_{\mathcal{H}}$ reflete a suavidade das funções em $\mathcal{H}$ e que cada espaço contém uma noção de suavidade diferente. Assim, dado um novo problema, um usuário pode desenhar sua função de perda L e escolher um espaço que julgar adequado para criar a sua função de predição g. Veremos que *smoothing splines*, *support vector regression* e *kernel ridge regression* são casos particulares dessa abordagem com escolhas particulares de L e/ou $\mathcal{H}$.

4.6.1 A matemática do RKHS

Seja $\mathcal{X}$ o espaço das covariáveis (*features*), que pode ser mais geral que $\mathbb{R}^d$ que consideramos até aqui. Nesta seção veremos que um RKHS $\mathcal{H}$ é essencialmente um subespaço de $L^2(\mathcal{X})$ que contém funções *suaves*. Em particular, a norma $||g||^2_{\mathcal{H}}$ é uma medida de suavidade da função g. Uma quantidade fundamental na definição de um RKHS é um *Kernel de Mercer*:[5]

Definição 1. *Seja $K(\mathbf{x}_a, \mathbf{x}_b)$ uma função cujo domínio é $\mathcal{X} \times \mathcal{X}$[6]. Dizemos que K é um Kernel de Mercer se ele sastisfaz às seguintes condições:*

[5] Um Kernel de Mercer não é um kernel suavizador como aquele usado no estimador de Nadaraya-Watson.

[6] Para métodos de penalização em RKHS, o espaço amostral $\mathcal{X}$ pode ser um espaço qualquer, isto é, não precisa ser necessariamente $\mathbb{R}^d$.

- ***Simétrico***: $K(\mathbf{x}_a, \mathbf{x}_b) = K(\mathbf{x}_b, \mathbf{x}_a)$ *para todo* $\mathbf{x}_a, \mathbf{x}_b \in \mathcal{X}$
- ***Positivo Semi-Definido***: *a matriz* $\left[K\left(\mathbf{x}_i, \mathbf{x}_j\right)\right]_{i,j=1}^{n}$ *é positiva semi-definida para todo* $n \in \mathbb{N}$ *e para todo* $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{X}$.

Exemplo 4.1. Alguns kernels de Mercer comuns são:

- **kernel polinomial:** $K(\mathbf{x}_i, \mathbf{x}_l) = (1 + \langle \mathbf{x}_i, \mathbf{x}_l \rangle)^d$
- **kernel gaussiano:** $K(\mathbf{x}_i, \mathbf{x}_l) = \exp\left\{-\frac{d^2(\mathbf{x}_i, \mathbf{x}_l)}{2h^2}\right\}$

Veja mais exemplos em Scholkopf e Smola (2002).

□

No contexto de regressão, $K(\mathbf{x}_a, \mathbf{x}_b)$ representa uma maneira de se medir a similaridade entre dois vetores de covariáveis $\mathbf{x}_a$ e $\mathbf{x}_b$.

A seguinte decomposição de um kernel é essencial para a construção de RKHSs:

Teorema 4. ***[Teorema de Mercer]*** *Todo Kernel de Mercer K pode ser decomposto como*

$$K(\mathbf{x}_a, \mathbf{x}_b) = \sum_{i \geq 0} \gamma_i \phi_i(\mathbf{x}_a) \phi_i(\mathbf{x}_b),$$

em que $\sum_{i \geq 0} \gamma_i^2 < \infty$ *e* $\phi_0, \phi_1, \ldots$ *é um conjunto de funções.*

Um exemplo trivial de aplicação do Teorema de Mercer é dado no início da Seção 4.6: o kernel implícito na regressão ridge: $K(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}_a, \mathbf{x}_b \rangle = \sum_{i=0}^{d} x_{a,i} x_{b,i}$, o produto interno das covariáveis. Nesse caso, $\gamma_i = 1$ e $\phi_i(\mathbf{x}) = x_i$, $i = 1, \ldots, d$.

Dado um kernel K, seu RKHS associado é definido da seguinte maneira:

Definição 2. *Seja K um kernel, e sejam* ϕ_i *e* γ_i, $i \geq 0$, *como no Teorema 4. Considere o espaço de funções*

$$\mathcal{H}_K = \left\{ g \in L^2(\mathcal{X}) : \text{ existem } (c_i)_{i \geq 0} \text{ com } \sum_{i \geq 1} \frac{c_i^2}{\gamma_i} < \infty \text{ tais que } g(\mathbf{x}) = \sum_{i \geq 1} c_i \phi_i(\mathbf{x}) \right\}$$

Dizemos que $\mathcal{H}_K$ *é o* Reproducing Kernel Hilbert Space *(RKHS) associado ao kernel K, em que a norma de uma função* $g(\mathbf{x}) = \sum_{i \geq 0} c_i \phi_i(\mathbf{x})$ *é definida por* $||g||_{\mathcal{H}_K}^2 := \sum_{i \geq 0} c_i^2 / \gamma_i$.

A norma $||g||_{\mathcal{H}_K}$ captura a suavidade de uma função g. Isso ocorre pois (i) a condição $\sum_{i\geq 0}\gamma_i^2 < \infty$ do Teorema de Mercer implica que $\gamma_i \longrightarrow 0$ e (ii) tipicamente as funções ϕ_i's ficam menos suaves à medida que $i \longrightarrow \infty$. Assim, $||g||^2_{\mathcal{H}_K} := \sum_{i\geq 0} c_i^2/\gamma_i$ é pequeno quando c_i é extremamente pequeno para i grande. De fato, para c_i^2/γ_i ser baixo quando i é grande, c_i deve ser extremamente pequeno, pois $\gamma_i \approx 0$.

Embora a decomposição do Teorema 4 em geral não seja única, cada kernel K define um único RKHS, de modo que o espaço da Definição 2 de fato está bem definido:

Teorema 5. *A um kernel K corresponde um único RKHS $\mathcal{H}_K$.*

4.6.2 Solução para o problema de otimização

O problema de estimação de uma função de regressão via RKHSs consiste em encontrar a solução para uma generalização da Equação 4.5. Mais especificamente, buscamos encontrar

$$\arg\min_{g\in\mathcal{H}_K} \sum_{k=1}^{n} L(g(\mathbf{x}_k), y_k) + \lambda||g||^2_{\mathcal{H}_K}, \tag{4.7}$$

em que $\mathcal{H}_K$ é um RKHS arbitrário e $L(g(\mathbf{x}_k), y_k)$ é uma função de perda arbitrária, ambos definidos pelo usuário do método. O termo λ é um *tuning parameter* que determina o balanço entre viés e variância, como no lasso. Note que λ grande leva a funções mais suaves (pois $||g||^2_{\mathcal{H}_K}$ é menor), enquanto λ pequeno leva a funções que se ajustam melhor ao conjunto de treinamento.

O seguinte teorema, frequentemente atribuído a Kimeldorf e Wahba (1971), é de fundamental importância para resolução do problema de minimização da Equação 4.7:

Teorema 6. ***[Teorema da Representação]*** *Seja K um kernel de Mercer correspondente ao RKHS $\mathcal{H}_K$. Considere um conjunto de treinamento $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ e uma função de perda arbitrária L. Então a solução de*

$$\arg\min_{g\in\mathcal{H}_K} \sum_{k=1}^{n} L(g(\mathbf{x}_k), y_k) + \lambda||g||^2_{\mathcal{H}_K} \tag{4.8}$$

existe, é única e tem a forma

$$g(\mathbf{x}) = \sum_{k=1}^{n} \alpha_k K(\mathbf{x}_k, \mathbf{x}).$$

Este resultado converte o problema de busca pelo mínimo na classe $\mathcal{H}_K$ em um problema de otimização de um número finito de coeficientes $\alpha_1, \ldots, \alpha_n$. Esse novo problema de otimização pode ser resolvido usando técnicas usuais de cálculo ou cálculo numérico.

O Teorema de Mercer (Teorema 4) também ajuda a simplificar o problema de minimizar a função objetivo, pois mostra que a fórmula para a norma de g do Teorema 6 pode ser simplificada:

$$\sum_{k=1}^{n} \alpha_k K(\mathbf{x}_k, \mathbf{x}) = \sum_{k=1}^{n} \alpha_k \sum_{i \geq 0} \gamma_i \phi_i(\mathbf{x}_k) \phi_i(\mathbf{x}) = \sum_{i \geq 0} \gamma_i \left(\sum_{k=1}^{n} \alpha_k \phi_i(\mathbf{x}_k) \right) \phi_i(\mathbf{x}).$$

Assim, para $g(\mathbf{x}) = \sum_{k=1}^{n} \alpha_k K(\mathbf{x}_k, \mathbf{x})$, temos

$$\begin{aligned} ||g||_{\mathcal{H}_K}^2 &= \sum_{i \geq 0} \frac{\left(\gamma_i \left(\sum_{k=1}^{n} \alpha_k \phi_i(\mathbf{x}_k)\right)\right)^2}{\gamma_i} = \sum_{i \geq 0} \gamma_i \left(\sum_{1 \leq j,k \leq n} \alpha_j \alpha_k \phi_i(\mathbf{x}_j) \phi_i(\mathbf{x}_k) \right) \\ &= \sum_{1 \leq j,k \leq n} \alpha_j \alpha_k \left(\sum_{i \geq 0} \gamma_i \phi_i(\mathbf{x}_j) \phi_i(\mathbf{x}_k) \right) = \sum_{1 \leq j,k \leq n} \alpha_j \alpha_k K(\mathbf{x}_j, \mathbf{x}_k). \end{aligned} \tag{4.9}$$

Essa decomposição facilita o cálculo da solução apresentada pelo Teorema 6: a solução para o problema de otimização sobre funções da Equação 4.8 é dada pela solução do seguinte problema de otimização sobre vetores:

$$\arg\min_{\alpha_1, \ldots, \alpha_n} \sum_{k=1}^{n} L\left(\sum_{i=1}^{n} \alpha_i K(\mathbf{x}_i, \mathbf{x}), (\mathbf{x}_k, y_k) \right) + \lambda \sum_{1 \leq j,k \leq n} \alpha_j \alpha_k K(\mathbf{x}_j, \mathbf{x}_k).$$

Na sequência, apresentamos alguns exemplos do cálculo desta solução em problemas particulares.

4.6.3 Exemplo 1: Kernel Ridge Regression

Quando a função de perda da Equação 4.8 é a perda quadrática, $L(g(\mathbf{x}_k), y_i) = (y_i - g(\mathbf{x}_i))^2$, o estimador obtido minimizando-se a Equação 4.7 é chamado de *kernel ridge regression*. Isso ocorre pois, se $\mathcal{H}_K$ é o espaço descrito na Equação 4.6 (isto é, $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$), o problema da Equação 4.7 equivale ao problema da regressão ridge. Assim, para esta função de perda, o resultado é uma generalização da regressão ridge, já que outros kernel's podem ser usados.

Para esta função de perda, o Teorema da Representação e a Equação 4.9 mostram que a solução de

$$\arg\min_{g\in\mathcal{H}_K}\sum_{j=1}^{n}(y_j-g(\mathbf{x}_j))^2+\lambda||g||^2_{\mathcal{H}_K}$$

é dada por $\widehat{g}(\mathbf{x})=\sum_{k=1}^{n}\widehat{\alpha}_k K(\mathbf{x}_k,\mathbf{x})$, em que $\widehat{\alpha}_k$'s são obtidos via

$$\arg\min_{\alpha_1,\ldots,\alpha_n}\sum_{j=1}^{n}\left(y_j-\sum_{k=1}^{n}\alpha_k K(\mathbf{x}_k,\mathbf{x}_j)\right)^2+\lambda\sum_{1\leq j,k\leq n}\alpha_j\alpha_k K(\mathbf{x}_j,\mathbf{x}_k).$$

Matricialmente, este problema pode ser reescrito como

$$\arg\min_{\boldsymbol{\alpha}}(\mathbf{y}-\mathbb{K}\boldsymbol{\alpha})^\intercal(\mathbf{y}-\mathbb{K}\boldsymbol{\alpha})+\lambda\boldsymbol{\alpha}^\intercal\mathbb{K}\boldsymbol{\alpha}, \tag{4.10}$$

em que $\boldsymbol{\alpha}=(\alpha_1,\ldots,\alpha_n)^\intercal$, $\mathbf{y}=(y_1,\ldots,y_n)^\intercal$ e

$$\mathbb{K}=\begin{bmatrix}K(\mathbf{x}_1,\mathbf{x}_1) & K(\mathbf{x}_1,\mathbf{x}_2) & \cdots & K(\mathbf{x}_1,\mathbf{x}_n)\\ K(\mathbf{x}_2,\mathbf{x}_1) & K(\mathbf{x}_2,\mathbf{x}_2) & \cdots & K(\mathbf{x}_2,\mathbf{x}_n)\\ \vdots & \vdots & \ddots & \vdots\\ K(\mathbf{x}_n,\mathbf{x}_1) & K(\mathbf{x}_n,\mathbf{x}_2) & \cdots & K(\mathbf{x}_n,\mathbf{x}_n)\end{bmatrix} \tag{4.11}$$

(a matriz $\mathbb{K}$ é chamada de matriz de Gram). A solução para a Equação 4.10 é dada por

$$\widehat{\boldsymbol{\alpha}}=(\mathbb{K}+\lambda\mathbb{I})^{-1}\mathbf{y}, \tag{4.12}$$

de modo que o estimador dado pela *kernel ridge regression* é

$$\widehat{g}(\mathbf{x})=\widehat{\boldsymbol{\alpha}}^\intercal\mathbf{k}=\mathbf{y}^\intercal(\mathbb{K}+\lambda\mathbb{I})^{-1}\mathbf{k}, \tag{4.13}$$

com $\mathbf{k}=(K(\mathbf{x}_1,\mathbf{x}),\ldots,K(\mathbf{x}_n,\mathbf{x}))$.

4.6.3.1 Smoothing Splines

Quando $\mathbf{x}\in[0,1]$, um caso particular da *kernel ridge regression* é o de *smoothing splines*, em que se busca pela função g que minimiza

$$\sum_{k=1}^{n} (y_k - g(\mathbf{x}_k))^2 + \lambda \int_0^1 ||g''(\mathbf{x})||^2 d\mathbf{x}. \tag{4.14}$$

O termo $\int_0^1 ||g''(\mathbf{x})||^2 d\mathbf{x}$ também é uma forma de se medir o quão suave g é (veja mais sobre isso na Seção 4.13); essa quantidade corresponde à norma de um RKHS específico, veja Nosedal-Sanchez et al. (2012), Pearce e Wand (2006) e Wahba (1990).[7]

Pode-se mostrar que a solução para esse problema corresponde a uma expansão de $g(\mathbf{x})$ em uma base formada por splines naturais cúbicos com nós no conjunto de treinamento (Wahba, 1990). Contudo, ao contrário da metodologia apresentada na Seção 4.2, aqui os coeficientes da expansão são penalizados de modo a se obter soluções mais suaves. É justamente por isso que esse método é chamado de *smoothing splines*.

4.6.3.2 O Truque do Kernel

O estimador do *kernel ridge regression* também pode ser motivado sem a evocação direta de *Reproducing Kernel Hilbert Spaces*. Para tanto, começamos reescrevendo o estimador dado pela *regressão ridge*. Vamos assumir que a variável resposta foi redefinida segundo

$$Y_i \leftarrow Y_i - \bar{Y},$$

de modo que não colocaremos um intercepto no modelo. Neste caso, a regressão ridge busca por

$$\widehat{\boldsymbol{\beta}} = \arg\min_{\beta} \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{d} \beta_j x_{i,j} \right)^2 + \lambda \sum_{j=1}^{d} \beta_j^2,$$

e tem como solução

$$\widehat{\boldsymbol{\beta}} = (\widehat{\beta}_0, \widehat{\beta}_1, \ldots, \widehat{\beta}_d) = (\mathbb{X}^\intercal \mathbb{X} + \lambda \mathbb{I})^{-1} \mathbb{X}^\intercal Y.$$

Com um pouco de álgebra, pode-se mostrar que $\widehat{\boldsymbol{\beta}}$ pode ser escrito como $\mathbb{X}^\intercal(\mathbb{X}\mathbb{X}^\intercal +$

[7] Tecnicamente, neste contexto é necessário uma versão um pouco mais geral do Teorema da Representação, veja Nosedal-Sanchez et al. (2012) e Wahba (1990).

$\lambda\mathbb{I})^{-1}Y$, de modo que o estimador da regressão é dado por

$$g(\mathbf{x}) = Y^\intercal(\mathbb{X}\mathbb{X}^\intercal + \lambda\mathbb{I})^{-1}\mathbb{X}\mathbf{x} = Y^\intercal(\mathbb{K} + \lambda\mathbb{I})^{-1}\mathbf{k}$$

em que

$$\mathbf{k} = (\langle \mathbf{x}_1, \mathbf{x}\rangle, \ldots, \langle \mathbf{x}_n, \mathbf{x}\rangle)^\intercal$$

e

$$\mathbb{K} = \begin{bmatrix} \langle \mathbf{x}_1, \mathbf{x}_1\rangle & \langle \mathbf{x}_1, \mathbf{x}_2\rangle & \cdots & \langle \mathbf{x}_1, \mathbf{x}_n\rangle \\ \langle \mathbf{x}_2, \mathbf{x}_1\rangle & \langle \mathbf{x}_2, \mathbf{x}_2\rangle & \cdots & \langle \mathbf{x}_2, \mathbf{x}_n\rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle \mathbf{x}_n, \mathbf{x}_1\rangle & \langle \mathbf{x}_n, \mathbf{x}_2\rangle & \cdots & \langle \mathbf{x}_n, \mathbf{x}_n\rangle \end{bmatrix}.$$

Assim, para calcular $g(\mathbf{x})$, é suficiente sabermos os produtos internos entre todos os pares de observação, isto é, $\langle \mathbf{x}_i, \mathbf{x}_l\rangle = \sum_{k=1}^{d} \mathbf{x}_{i,k}\mathbf{x}_{l,k}$. O truque do kernel se utiliza desse fato para construir estimadores não lineares. Mais especificamente, suponha que estejamos interessados em uma *transformação das covariáveis originais*, da mesma maneira que discutido na Seção 2.1. Por exemplo, suponhamos que temos duas covariáveis, (x_1, x_2), e que queremos estimar uma regressão que não é linear no espaço original, mas sim que tenha a forma

$$g(\mathbf{x}) = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 + \beta_3 x_2 + \beta_4 x_2^2 + \beta_5 x_1 x_2. \tag{4.15}$$

Para isso, podemos usar o estimador da regressão ridge com novas covariáveis

$$\mathbf{w} = (1, \sqrt{2}x_1, x_1^2, \sqrt{2}x_2, x_2^2, \sqrt{2}x_1x_2).$$

Como mostrado anteriormente, para calcular este estimador, somente é necessário que saibamos os produtos internos das observações em relação às *novas covariáveis*, isto é,

$$\langle \mathbf{w}_i, \mathbf{w}_l\rangle = 1 + 2x_{i,1}x_{l,1} + x_{i,1}^2x_{l,1}^2 + 2x_{i,2}x_{l,2} + x_{i,2}^2x_{l,2}^2 + 2x_{i,1}x_{i,2}x_{l,1}x_{l,2}. \tag{4.16}$$

Mais precisamente, o estimador é dado por

$$\widehat{r}(\mathbf{x}) = Y^\intercal(\mathbb{K} + \lambda\mathbb{I})^{-1}\mathbf{k} \tag{4.17}$$

em que $\mathbf{k} = (K(\mathbf{x}_1, \mathbf{x}), \ldots, K(\mathbf{x}_n, \mathbf{x}))$,

$$\mathbb{K} = \begin{bmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & K(\mathbf{x}_1, \mathbf{x}_2) & \cdots & K(\mathbf{x}_1, \mathbf{x}_n) \\ K(\mathbf{x}_2, \mathbf{x}_1) & K(\mathbf{x}_2, \mathbf{x}_2) & \cdots & K(\mathbf{x}_2, \mathbf{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ K(\mathbf{x}_n, \mathbf{x}_1) & K(\mathbf{x}_n, \mathbf{x}_2) & \cdots & K(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix} \quad \text{e} \quad K(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{w}_i, \mathbf{w}_j \rangle.$$

Note que podemos calcular $\mathbf{w}$ com $\mathbf{x}$, por isso utilizamos a notação $K(\mathbf{x}_i, \mathbf{x}_j)$ para representar o produto interno no espaço transformado. O motivo de usarmos a letra K ficará claro na sequência.

O truque, chamado de *truque do kernel*, consiste em notar que, para algumas transformações $\mathbf{w}(\mathbf{x})$, é possível calcular o produto interno $\langle \mathbf{w}_i, \mathbf{w}_l \rangle$ facilmente, em particular não é necessário calcular as novas covariáveis em cada amostra. Isso ocorre na transformação utilizada no exemplo: o produto interno da Equação 4.16 pode ser desenvolvido como

$$K(\mathbf{x}_i, \mathbf{x}_l) = \langle \mathbf{w}_i, \mathbf{w}_l \rangle = (1 + x_{i,1}x_{l,1} + x_{i,2}x_{l,2})^2 = (1 + \langle \mathbf{x}_i, \mathbf{x}_l \rangle)^2.$$

Assim, para calcular $\langle \mathbf{w}_i, \mathbf{w}_l \rangle$, precisamos apenas saber calcular $\langle \mathbf{x}_i, \mathbf{x}_l \rangle$, isto é, não é necessário que se calcule em nenhum momento $\mathbf{w}_i$. Ou seja, é possível calcular um estimador do tipo da Equação 4.15 sem nunca calcular x_1^2, x_1x_2 etc! Para os casos em que d é grande, isso representa uma grande economia de memória e tempo computacional. Por exemplo, se $d = 10^6$ e estamos interessados em todos os produtos dois a dois entre as covariáveis, a abordagem ingênua necessitaria de um armazenamento de uma matriz com $(10^{12})/2$ colunas, enquanto a Equação 4.17 necessita apenas de uma matriz $n \times n$ além da matriz $\mathbb{X}$ original.

Na prática, para se usar o truque do kernel, começamos especificando diretamente um kernel $K(\mathbf{x}_i, \mathbf{x}_j)$ que corresponde a um produto interno em um espaço transformado. Qualquer kernel de Mercer pode ser utilizado, pois o Teorema de Mercer (Teorema 4) garante que tais kernels correspondem a um produto interno no espaço (potencialmente infinito) dado por $(\sqrt{\gamma}_1\phi_1(\mathbf{x}), \sqrt{\gamma}_2\phi_2(\mathbf{x}), \ldots)$. Uma vez que o kernel está escolhido, podemos calcular o estimador da regressão ridge para este kernel.

Note que esse truque permite que trabalhemos com a regressão ridge em espaços com infinitas covariáveis, o que não seria possível com a formulação original da re-

gressão ridge. Isto ocorre justamente porque nunca é necessário calcular essas novas covariáveis, mas apenas produtos internos neste espaço. Este é o caso, por exemplo, do kernel gaussiano: a transformação correspondente a ele é de fato infinita.

Veja que o estimador da Equação 4.17 – ainda que seja motivado por considerações bastante diferentes – é o mesmo que o estimador discutido anteriormente (Equação 4.13).

Observação 4.2. Note que o kernel linear $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$ corresponde à regressão ridge nas covariáveis originais. Nesse caso, usar a Equação 4.17 ao invés de 3.4 para resolver o problema não é uma vantagem: se $n > d$, ela leva a um tempo computacional maior para implementação, uma vez que é necessário inverter uma matriz $n \times n$ ao invés de uma matriz $d \times d$ como na formulação original. A grande vantagem dessa abordagem é permitir transformações não lineares possivelmente infinitas.

□

Finalmente, note que o truque do kernel pode ser utilizado em qualquer método que dependa das covariáveis originais através do cálculo produtos internos. Ou seja, sempre que um método de estimação da regressão puder ser calculado sem o conhecimento de $\mathbf{x}_1, \ldots, \mathbf{x}_n$, mas apenas com o conhecimento dos produtos internos entre esses pares, o truque do kernel pode ser utilizado.

4.6.3.3 Implementação eficiente da kernel ridge regression

Para implementar a *kernel ridge regression* é necessário calcular

$$(\mathbb{K} + \lambda \mathbb{I})^{-1}$$

na Equação 4.12. Uma implementação ingênua deste método é computacionalmente lenta, pois é necessário calcular a inversa de $\mathbb{K} + \lambda \mathbb{I}$ para cada λ de interesse. Uma maneira de torná-la mais rápida é usar a decomposição SVD da matriz $\mathbb{K}$:

$$\mathbb{K} = UDU^\intercal,$$

em que D é diagonal e $UU^\intercal = \mathbb{I}$. Note que essa decomposição não depende de λ. Além disso,

$$(\mathbb{K} + \lambda \mathbb{I})^{-1} = (UDU^\intercal + \lambda \mathbb{I})^{-1} = (U(D + \lambda \mathbb{I})U^\intercal)^{-1} = (U^{-1})^\intercal (D + \lambda \mathbb{I})^{-1} U^{-1}.$$

Logo, para calcular $(\mathbb{K} + \lambda\mathbb{I})^{-1}$, uma vez que a decomposição foi feita, é necessário apenas inverter uma matriz diagonal, o que é extramamente rápido. Assim, $(\mathbb{K} + \lambda\mathbb{I})^{-1}$, utilizando-se essa decomposição, pode ser calculado para diversos λ's eficientemente.

4.6.4 Exemplo 2: Support Vector Regression Machine

Para criar estimadores da função de regressão, Support Vector Regression Machines (Drucker et al., 1997) utilizam uma função de perda diferente da quadrática. Mais especificamente, seja $\epsilon > 0$ um número fixo e K um kernel de Mercer. A função de predição g dada pela support vector regression (SVR) é aquela função que minimiza

$$\arg\min_{g \in \mathcal{H}_K} \sum_{k=1}^{n} L(g(\mathbf{x}_k), y_k) + \lambda ||g||^2_{\mathcal{H}_K}, \tag{4.18}$$

em que $L(g(\mathbf{x}_k), y_k) = (|y_k - g(\mathbf{x}_k)| - \epsilon)_+$. Segundo essa função de perda, a distância entre a predição $g(\mathbf{x}_k)$ e a observação y_k ser menor que ϵ é suficiente para que não haja nenhum custo no erro. Por outro lado, se a distância é maior que ϵ, a perda é dada por $|y_k - g(\mathbf{x}_k)| - \epsilon$. Por esse motivo, essa perda é chamada de ϵ-insensível.

Pode-se utilizar o Teorema da Representação (Teorema 6) para simplificar o problema dado pela Equação 4.18 de forma a torná-lo um problema finito-dimensional. Contudo, ao contrário do que ocorre com a kernel ridge regression (Seção 4.6.3), a solução para tal problema não é analítica, e ,portanto, requer o uso de métodos numéricos de otimização.

Para ajustar a support vector regression machines no R, pode-se utilizar o pacote `e1071`.

```
ajuste <- svm(x = x_treinamento,
              y = t_treinamento,
              type = "eps-regression", kernel ="radial",
              gamma = gamma)

predito <- predict(ajuste, newdata = novos_dados)
```

O valor de ϵ pode ser definido pelo usuário do método e é, em geral, escolhido via validação cruzada.

4.7 Modelos Aditivos

Um problema de diversos dos métodos não paramétricos é sua dificuldade de interpretabilidade quando comparado com métodos paramétricos. Modelos aditivos (*additive models*, Hastie e Tibshirani 1986) são uma forma de encontrar um balanço entre interpretabilidade e flexibilidade. Eles consistem em uma generalização de uma regressão linear de modo a permitir uma relação não linear entre cada covariável x_j e a resposta y. A ideia básica é trocar o termo $\beta_j x_j$ na equação da regressão linear (Eq. 2.1) por $g_j(x_j)$, em que $g_j(x_j)$ são funções *univariadas* suaves que são estimadas com base nos dados. Assim, um modelo aditivo pode ser escrito como

$$g(\mathbf{x}) = \alpha + \sum_{j=1}^{d} g_j(x_j).$$

Note que este modelo não é identificável, já que podemos adicionar qualquer constante α e retirar a mesma constante de qualquer g_j, obtendo assim a mesma solução. Uma forma de evitar isso é restringir que nossos estimadores sejam tais que $\widehat{\alpha} = \overline{Y}$, e forçar $\sum_{i=1}^{n} \widehat{g}_j(X_{i,j}) = 0$. Este modelo é chamado de aditivo pois supõe-se que a função de regressão pode ser decomposta em termos de uma soma de funções. Assim, esse método impõe uma estrutura mais restritiva do que a observada, por exemplo, no estimador de Nadaraya-Watson. Sua grande vantagem é que ele leva a uma regressão estimada que é mais fácil de ser interpretada.

Modelos aditivos, em geral, são ajustados utilizando-se o algoritmo *backfitting*:

1. Inicialize $\widehat{\alpha} = \overline{Y}$, e crie estimativas iniciais $\widehat{g}_1, \ldots, \widehat{g}_d$.

2. Faça, até obter convergência:
   Para $j = 1, \ldots, d$

   (a) Calcule $\widetilde{Y}_i = Y_i - \widehat{\alpha} - \sum_{k \neq j} \widehat{g}_k(x_{i,k})$

   (b) Estime a regressão de $(\widetilde{Y}_i)_i$ em $(x_{j,i})_i$, $\widehat{g}_j$. Aqui pode-se utilizar qualquer método de regressão univariada; paramétrico ou não paramétrico.

   (c) Defina $\widehat{g}_j(\mathbf{x}) = \widehat{g}_j(\mathbf{x}) - \frac{1}{n}\sum_{i=1}^{n} \widehat{g}_j(x_{i,j})$

A seguir mostramos um exemplo das vantagens deste método.

Exemplo 4.2. Neste exemplo ajustamos um modelo aditivo para dados do Programa das Nações Unidas para o Desenvolvimento sobre o IDH de municípios do Brasil em 1991, 2000 e 2010.[8] Cada amostra aqui representa dados sobre um município em um dado ano. A variável resposta utilizada é o IDH de cada município, enquanto as variáveis explicativas são:

- O percentual da população de 11 a 13 anos de idade frequentando os anos finais do fundamental ou que já concluiu o fundamental no município,
- A renda per capita média do município,
- O ano em que os dados foram coletados.

Ainda que a última variável mencionada seja categorizada, é possível utilizar método aditivos para esse caso. Para as variáveis contínuas, optou-se por utilizar smoothing splines com 4 graus de liberdade.

A Figura 4.5 ilustra as três funções estimadas $r_j(x_j)$, juntamente com erros padrão sobre elas. Com ela, é possível identificar não linearidades nas relações entre as variáveis explicativas e a variável resposta e ter uma visão global de como cada covariável está associada à resposta.

□

Para implementar modelos aditivos, pode-se usar o pacote `gam`.

```
library(gam)

ajuste <- gam(resposta ~ s(explicativa_1, 4) +
                         s(explicativa_2, 4),
              data = dados)
```

A função `s()` é parte do pacote `gam` e é aplicada para indicar que a utilização de smoothing splines com 4 graus de liberdade. Predições podem ser feitas com a função `predict`.

[8] http://www.atlasbrasil.org.br/.

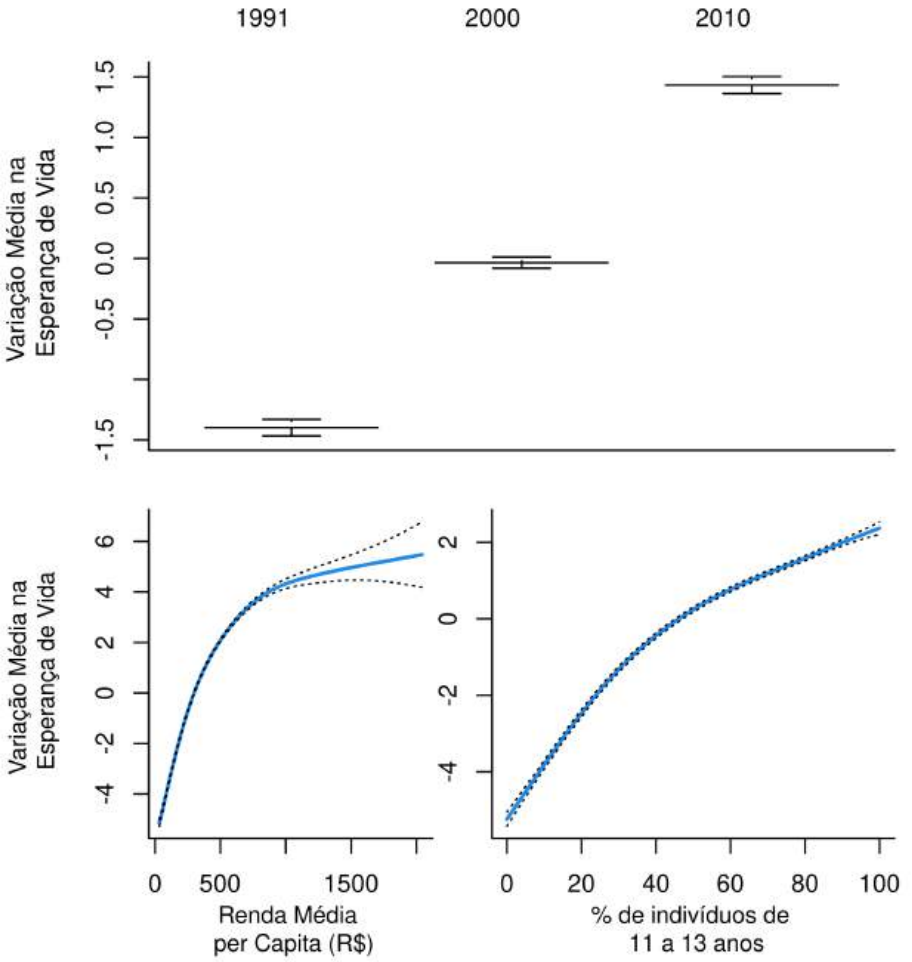


Figura 4.5: Ajuste dado pelo modelo aditivo para os dados de IDH municipais no Brasil

4.8 Árvores de Regressão

Árvores de regressão consistem em uma metodologia não paramétrica que leva a resultados extremamente interpretáveis. Uma árvore é construída por particionamentos recursivos no espaço das covariáveis. Cada particionamento recebe o nome de nó e cada resultado final recebe o nome de folha; veja a Figura 4.6.

A utilização da árvore para prever uma nova observação é feita do seguinte modo: começando pelo topo, verificamos se a condição descrita no topo (primeiro nó) é satisfeita. Caso seja, seguimos a esquerda. Caso contrário, seguimos a direita. Assim prosseguimos até atingir uma folha. No caso ilustrativo da Figura 4.6, se a condição 1 for satisfeita, a predição é dada por F1. Caso não seja satisfeita, seguimos a direita e assim, encontramos outra condição. Caso a mesma seja satisfeita, a observação é prevista como F2 e, caso contrário, é prevista como F3.

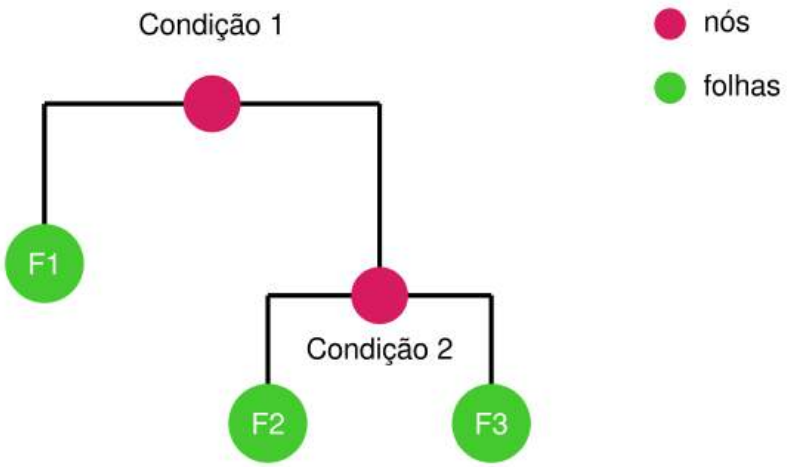


Figura 4.6: Exemplo de estrutura de uma árvore de regressão.

A Figura 4.7 ilustra um exemplo concreto de uma árvore de regressão, construída com o objetivo de prever o salário de um indivíduo dadas covariáveis como idade e anos de estudo. Note como é fácil usar este objeto para entender a relação entre as

variáveis explicativas e a variável resposta, ao contrário do que ocorre com a maior parte dos métodos não paramétricos.

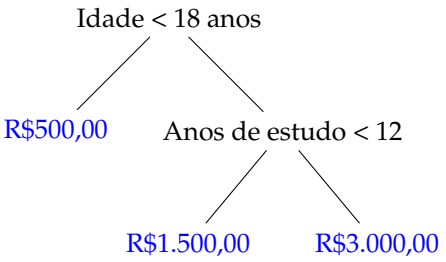


Figura 4.7: Exemplo de árvore da regressão do salário de um indivíduo dadas as covariáveis idade e anos de estudo.

Formalmente, uma árvore cria uma partição do espaço das covariáveis em regiões distintas e disjuntas: $R_1, \ldots, R_j$. A predição para a resposta Y de uma observação com covariáveis $\mathbf{x}$ que estão em R_k é dada por

$$g(\mathbf{x}) = \frac{1}{|\{i : \mathbf{x}_i \in R_k\}|} \sum_{i:\mathbf{x}_i \in R_k} y_i. \tag{4.19}$$

Isto é, para prever o valor da resposta de $\mathbf{x}$, observamos a região a qual a observação $\mathbf{x}$ pertence e, então, calculamos a média dos valores da variável resposta das amostras do conjunto de treinamento pertencentes àquela mesma região.

A criação da estrutura de uma árvore de regressão é feita através de duas grandes etapas: (i) a criação de uma árvore completa e complexa e (ii) a poda dessa árvore, com a finalidade de evitar o super ajuste.

No passo (i), busca-se criar uma árvore que leve a partições "puras", isto é, partições nas quais os valores de Y nas observações do conjunto de treinamento em cada uma das folhas sejam homogêneos. Para tanto, avalia-se o quão razoável uma dada árvore T é através de seu erro quadrático médio,

$$\mathcal{P}(T) = \sum_R \sum_{i:\mathbf{x}_i \in R} \frac{(y_i - \widehat{y}_R)^2}{n},$$

em que $\widehat{y}_R$ é o valor predito para a resposta de uma observação pertencente à região R. Encontrar T que minimize $\mathcal{P}(T)$ é computacionalmente inviável. Assim, utiliza-se uma heurística para encontrar uma árvore com erro quadrático médio baixo que

consiste na criação de divisões binárias recursivas, como mostrado na Figura 4.8. Inicialmente, o algoritmo particiona o espaço de covariáveis em duas regiões distintas (Figura 4.8 (a)). Para escolher essa partição, busca-se, dentre todas as covariáveis x_i e cortes t_1, aquela combinação que leva a uma partição (R_1, R_2) com predições de menor erro quadrático:

$$\sum_{i:\mathbf{x}_i \in R_1}^{n} (y_i - \widehat{y}_{R_1})^2 + \sum_{i:\mathbf{x}_i \in R_2}^{n} (y_i - \widehat{y}_{R_2})^2, \tag{4.20}$$

em que $\widehat{y}_{R_k}$ é a predição fornecida para a região R_k (veja Eq. 4.19). Assim, define-se

$$R_1 = \{\mathbf{x} : x_i < t_1\} \text{ e } R_2 = \{\mathbf{x} : x_i \geq t_1\},$$

em que x_i é a variavel escolhida e t_1 é o corte definido.

Uma vez estabelecidas tais regiões, o nó inicial da árvore é então fixado. No próximo passo busca-se particionar R_1 ou R_2 em regiões menores (Figura 4.8 (b)). Para escolher a nova divisão, a mesma estratégia é utilizada: busca-se, dentre todas as covariáveis x_i e cortes t_2, aquela combinação que leva a uma partição com menor erro quadrático. Note que agora também é necessário escolher qual região deve ser particionada: R_1 ou R_2. Assuma que R_1 foi a região escolhida, juntamente com a covariável x_j e o corte t_2. Chamemos a partição de R_1 de $\{R_{1,1}, R_{1,2}\}$, como mostra a Figura 4.8 (b). Assim,

$$R_{1,1} = \{\mathbf{x} : x_i < t_1, x_j < t_2\},\ R_{1,2} = \{\mathbf{x} : x_i < t_1, x_j \geq t_2\} \text{ e } R_2 = \{\mathbf{x} : x_i \geq t_1\}.$$

O procedimento continua recursivamente (veja Figuras 4.8 (c) e (d)), até que cheguemos a uma árvore com poucas observações em cada uma das folhas (por exemplo, o processo é interrompido quando todas as folhas têm menos de cinco observações).

A árvore criada utilizando-se esse processo produz bons resultados para o conjunto de treinamento, mas é muito provável que ocorra o super-ajuste. Isso gera uma performance preditiva ruim em novas observações. Assim, prossegue-se para o passo (ii), que é chamado de poda. O objetivo dessa etapa é tornar a árvore de regressão menor e menos complexa, de modo a diminuir a variância desse estimador. Nessa etapa do processo cada nó é retirado, um por vez, e observa-se como o erro de predição varia no conjunto de validação. Com base nisso, decide-se quais nós

permanecerão na árvore.

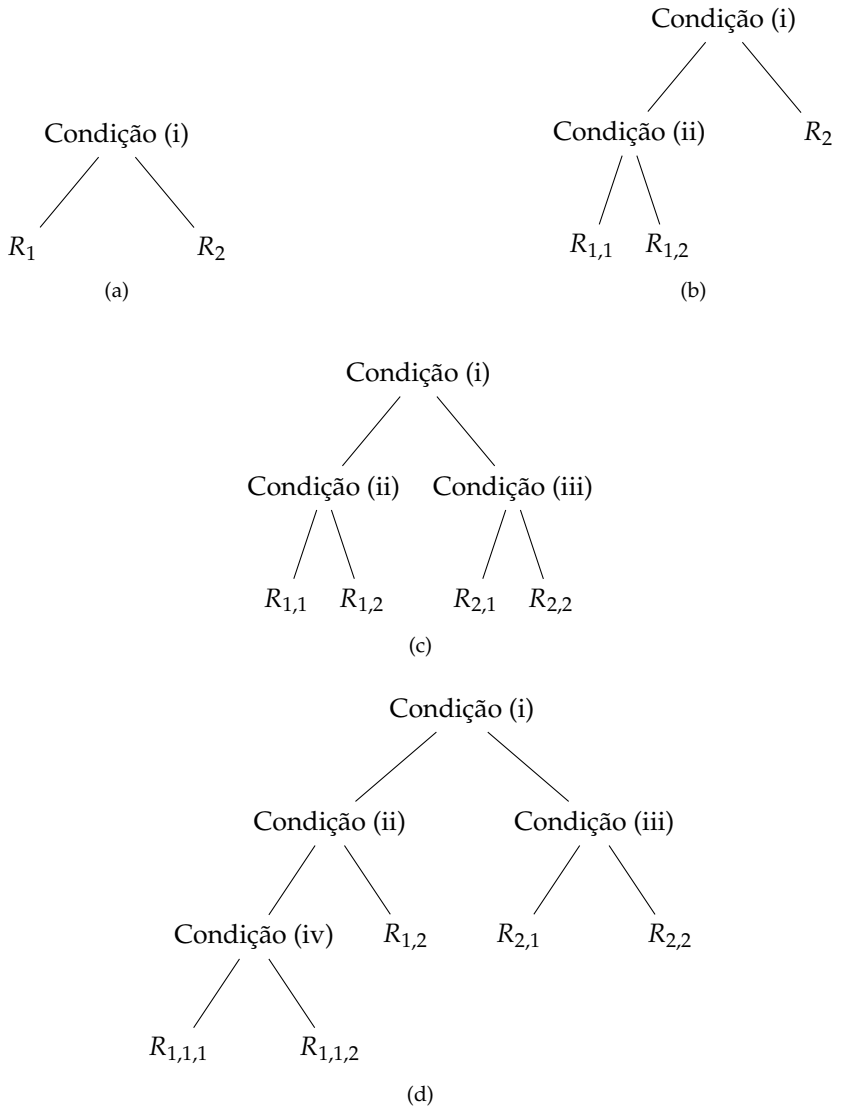


Figura 4.8: Processo de crescimento de uma árvore de regressão.

Para ajustar uma árvore no R, pode-se utilizar o pacote `rpart`. O pacote `rpart.plot` é utilizado para produzir visualizações das árvores de regressão definidas.

```
library(rpart); library(rpart.plot); data("mtcars")

fit <- rpart(mpg ~ ., method = "anova", data = mtcars)
melhor_cp <- fit$cptable[which.min(fit$cptable[, "xerror"]),
                         "CP"]
pfit <- rpart::prune(fit, cp = melhor_cp)
rpart.plot(pfit, type = 4, extra = 1)
```

A Figura 4.9 ilustra a árvore gerada pelo código acima no R. O tamanho de cada ramo na árvore gerada pode ser definido proporcionalmente à diminuição do erro quadrático médio que ocorreu quando a respectiva partição foi criada. Assim, ramos grandes indicam uma importância maior da covariável na predição da variável resposta.

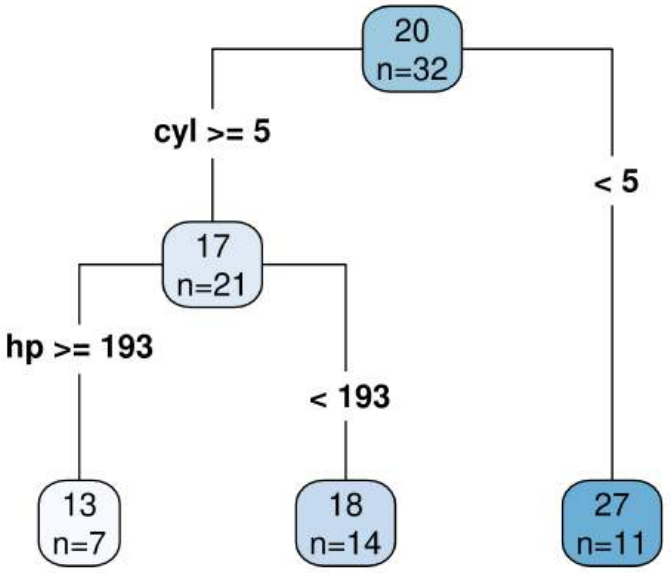


Figura 4.9: Árvore de regressão já podada.

Além de serem facilmente interpretáveis, árvores têm a vantagem de lidar trivialmente com covariáveis discretas. Por exemplo, se X_1 representa a cor de uma

flor, pode-se utilizar critérios de divisão como $X_1 \in \{\text{verde}, \text{vermelho}\}$ nos nós da árvore gerada. É importante ressaltar que isso não acontece quando se cria variáveis *dummies* para variáveis qualitativas. Além disso, a maneira como árvores são construídas faz com que covariáveis irrelevantes sejam descartadas. Assim, a seleção de variáveis é feita automaticamente. Finalmente, note que a estrutra de uma árvore naturalmente lida com interações entre variáveis. Assim, ao contrário de modelos lineares, não é necessário incluir termos de interação adicionais.

Na próxima seção serão apresentados métodos para combinar diversas árvores para melhorar o desempenho preditivo desse modelo.

4.9 Bagging e Florestas Aleatórias

> A forest of these trees is a spectacle too much for one man to see.
>
> David Douglas

Uma característica interessante das árvores de regressão é que são extremamente interpretáveis. No entanto, costumam apresentar baixo poder preditivo quando comparadas aos demais estimadores. Bagging e florestas aleatórias (Breiman, 2001b) são métodos que contornam essa limitação combinando diversas árvores para fazer uma predição para um mesmo problema.

Para motivar esta abordagem, imagine que, em um contexto de regressão, temos duas funções de predição para Y, $g_1(\mathbf{x})$ e $g_2(\mathbf{x})$. Os riscos dessas (condicionais em $\mathbf{x}$, mas não na amostra de treinamento) são dados, respectivamente, por

$$\mathbb{E}\left[(Y - g_1(\mathbf{x}))^2 | \mathbf{x}\right] \quad \text{e} \quad \mathbb{E}\left[(Y - g_2(\mathbf{x}))^2 | \mathbf{x}\right].$$

Considere agora o estimador combinado $g(\mathbf{x}) = (g_1(\mathbf{x}) + g_2(\mathbf{x}))/2$. Pelo resultado da Seção 1.5.2, temos que

$$\begin{aligned}
&\mathbb{E}\left[(Y - g(\mathbf{x}))^2 | \mathbf{x}\right] = \\
&= \mathbb{V}[Y|\mathbf{x}] + \frac{1}{4}\left(\mathbb{V}[g_1(\mathbf{x}) + g_2(\mathbf{x})|\mathbf{x}]\right) + \left(\mathbb{E}[Y|\mathbf{x}] - \frac{\mathbb{E}[g_1(\mathbf{x})|\mathbf{x}] + \mathbb{E}[g_2(\mathbf{x})|\mathbf{x}]}{2}\right)^2.
\end{aligned}$$

Assim, se $g_1(\mathbf{x})$ e $g_2(\mathbf{x})$ são não correlacionados ($\text{Cor}(g_1(\mathbf{x}), g_2(\mathbf{x})|\mathbf{x}) = 0$), não

viesados ($\mathbb{E}[g_1(\mathbf{x})|\mathbf{x}] = \mathbb{E}[g_2(\mathbf{x})|\mathbf{x}] = r(\mathbf{x})$) e têm mesma variância ($\mathbb{V}[g_1(\mathbf{x})|\mathbf{x}] = \mathbb{V}[g_2(\mathbf{x})|\mathbf{x}]$),

$$\mathbb{E}\left[(Y - g(\mathbf{x}))^2|\mathbf{x}\right] = \mathbb{V}[Y|\mathbf{x}] + \frac{1}{2}\mathbb{V}[g_i(\mathbf{x})|\mathbf{x}] \leq \mathbb{E}\left[(Y - g_i(\mathbf{x}))^2|\mathbf{x}\right] \tag{4.21}$$

com $i = 1,2$. Assim, é melhor se utilizar o estimador combinado $g(\mathbf{x})$ do que usar $g_1(\mathbf{x})$ ou $g_2(\mathbf{x})$ separadamente. Embora se considere apenas dois estimadores da função de regressão nesse exemplo, a conclusão permanece válida quando se combina um número B qualquer de estimadores.

Os métodos de *bagging* e florestas aleatórias se valem dessa ideia para melhorar as predições dadas por árvores. Ambas as abordagens consistem em criar B árvores distintas e combinar seus resultados para melhorar o poder preditivo em relação a uma árvore individual. Para criar as B árvores distintas, o *bagging* utiliza B amostras bootstrap da amostra original[9]. Para cada uma dessas amostras, cria-se uma árvore utilizando as técnicas descritas na Seção 4.8. Note, contudo, que assumimos na derivação da Equação 4.21 que os estimadores são não viesados. Assim, para criar árvores próximas de não-viesadas, não podamos as árvores criadas, isto é, utilizamos apenas o passo (i) descrito naquela seção.

Seja $g_b(\mathbf{x})$ a função de predição obtida segundo a b-ésima árvore. A função de predição dada pelo *bagging* é dada por

$$g(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B} g_b(\mathbf{x}).$$

Apesar do método *bagging* retornar preditores que não são tão fáceis de se interpretar quanto árvores, ele permite a criação de uma medida de importância para cada covariável. Essa medida de importância é baseada na redução da soma de quadrados dos resíduos (RSS - *residual sum of squares*) de cada divisão. Para ilustrar o cálculo dessa quantidade, considere a situação da Figura 4.10, na qual dividimos o nó pai em dois grupos: esq (esquerda) e dir (direita).

A redução no RSS pela variável utilizada é dada por

[9]Uma amostra bootstrap é uma amostra aleatória com reposição de mesmo tamanho da amostra original.

$$\text{RSS}_{\text{pai}} - \text{RSS}_{\text{esq}} - \text{RSS}_{\text{dir}} =$$
$$\sum_{i \in \text{pai}} \left(y_i - \overline{y}_{\text{pai}}\right)^2 - \sum_{i \in \text{esq}} \left(y_i - \overline{y}_{\text{esq}}\right)^2 - \sum_{i \in \text{dir}} \left(y_i - \overline{y}_{\text{dir}}\right)^2.$$

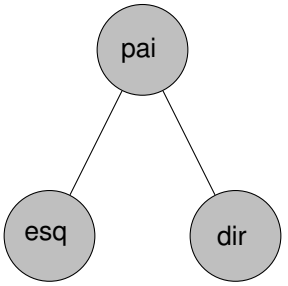


Figura 4.10: Divisão de uma árvore usada para a ilustração do cálculo do RSS.

Para calcular a medida de importância de uma variável, primeiramente calcula-se o total da redução da soma de quadrados dos resíduos (RSS - *residual sum of squares*) para todas as divisões que foram feitas com base nessa variável. Repete-se esse processo para todas as árvores e calcula-se a média dessa redução. Essa redução média é então utilizada como medida de importância para cada variável.

A Figura 4.11 ilustra a importância de cada covariável segundo dois critérios para o conjunto CO2 do R (veja exemplo na seção 4.9.1). Essa figura foi obtida com auxílio da função `varImpPlot` do pacote `randomForest` do R.

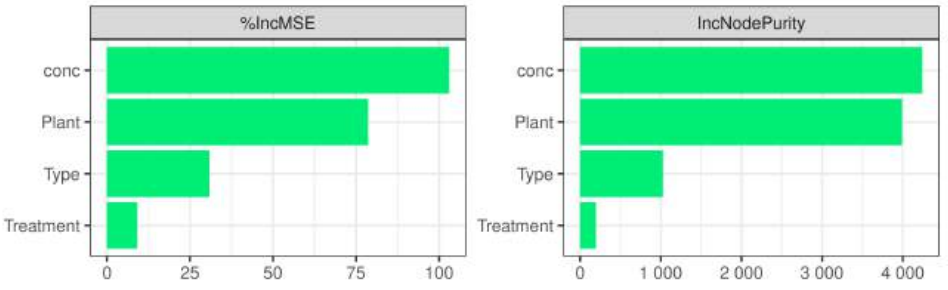


Figura 4.11: Importância da cada covariável do conjunto de dados CO2 na predição da variável uptake segundo o bagging.

4.9.1 Florestas Aleatórias

> And into the forest I go, to lose my mind and find my soul.
>
> John Muir

Na Equação 4.21, além da suposição de que os estimadores são não viesados, assumimos que são não correlacionados. Em geral, mesmo utilizando amostras bootstrap para construir cada uma das árvores, as árvores construídas tendem a dar predições parecidas. Assim, as funções $g_b(\mathbf{x})$ resultantes, em geral, apresentam alta correlação. Florestas tentam diminuir esta correlação modificando o mecanismo de criação das árvores para que essas se tornem diferentes umas das outras. Mais especificamente, ao invés de escolher qual das d covariáveis será utilizada em cada um dos nós da árvore, em cada passo só é permitido que seja escolhida uma dentre as $m < d$ covariáveis. Estas m covariáveis são escolhidas aleatoriamente dentre as covariáveis originais e, a cada nó criado, um novo subconjunto de covariáveis é sorteado.

O valor de m pode ser escolhido via validação cruzada. Resultados empíricos sugerem, de forma geral, que $m \approx d/3$ leva a uma boa performance. Note que, quando $m = d$, o método de florestas aleatórias se reduz ao *bagging*.

Esse método poder ser ajustado com o pacote `randomForest` da seguinte forma:

```
library(randomForest)

data(CO2)

ajuste <- randomForest(uptake ~ ., mtry = round(ncol(CO2)/3),
                       importance = TRUE, data = CO2)

varImpPlot(ajuste) # gráfico de importância
```

Esse método também pode ser utilizado com o pacote `ranger`, que é um pacote que apresenta uma boa performance computacional quando há um grande número de observações.

A Figura 4.12 ilustra o comportamento do risco preditivo em função de B, o número de árvores utilizadas, no conjunto de dados do CO2. A curva é bastante estável.

Em particular, não ocorre superajuste quando B cresce. Assim, de forma geral, fazer uma validação cruzada para escolher B não traz grandes ganhos (em contraste com o que ocorre com *tuning parameters* de outros métodos). Essa robustez do desempenho de florestas aleatórias à escolha de B é uma grande vantagem deste método.

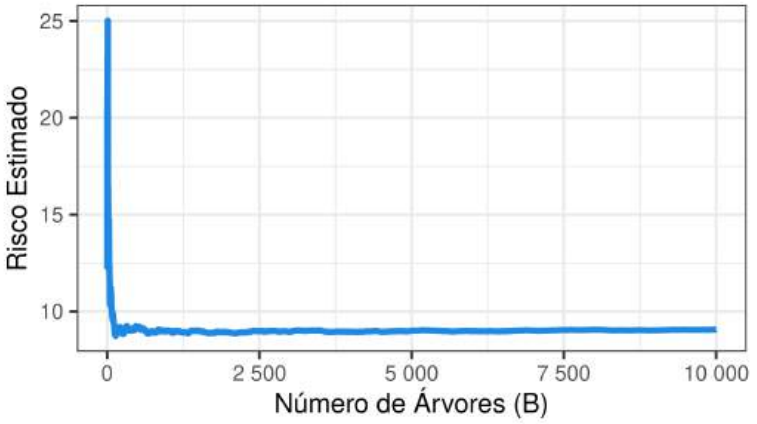


Figura 4.12: Comportamento do risco preditivo em função de B, o número de árvores utilizadas, no exemplo do CO2.

Florestas possuem um poder preditivo em geral muito maior que o de árvores. Para ilustrar essa diferença, aplicaremos ambos os métodos para o conjunto de dados simulados da Figura 4.13.

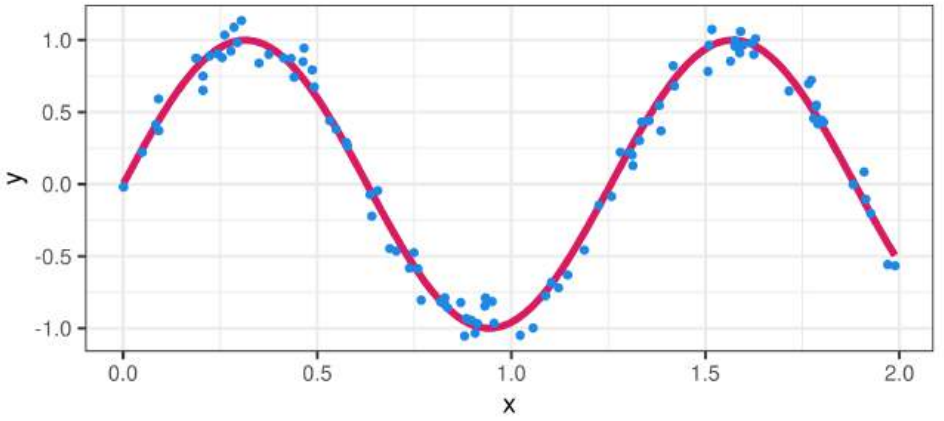


Figura 4.13: Dados simulados da regressão em vermelho.

A árvore e a floresta ajustadas para esses dados encontram-se na Figura 4.14. Note como o ajuste obtido com floresta aleatória é mais suave que o ajuste obtido com a árvore de regressão. Em particular, ele se aproxima melhor da regressão real. Também é interessante notar que as árvores utilizadas para construir a floresta (em azul) têm características muito distintas daquela usada individualmente. Isso ocorre pois essa última é podada.

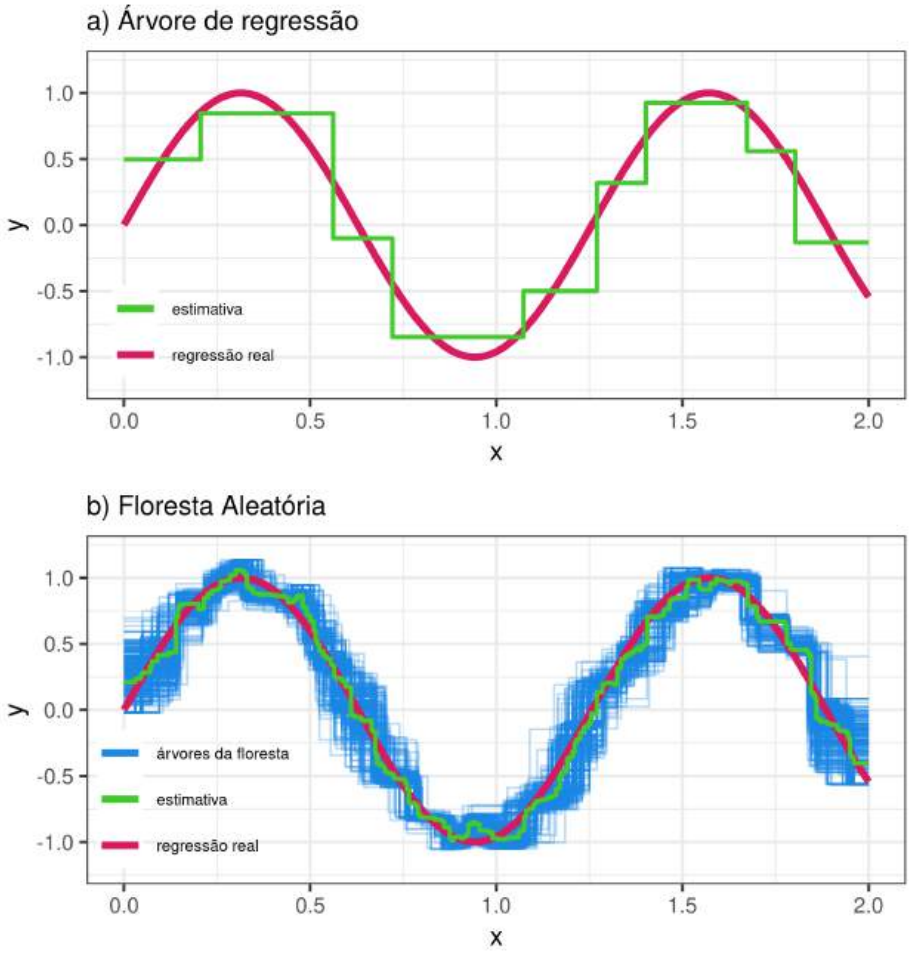


Figura 4.14: Exemplo de ajustes de árvore de regressão e floresta aleatória.

A Figura 4.15 ilustra árvores e florestas ajustadas a diferentes conjuntos de dados

vindos da mesma população que aqueles da Figura 4.13. Fica claro que árvores de regressão possuem uma variabilidade muito maior que florestas aleatórias. Isso é uma das razões que leva esse método a apresentadar um desempenho inferior em relação à floresta aleatória.

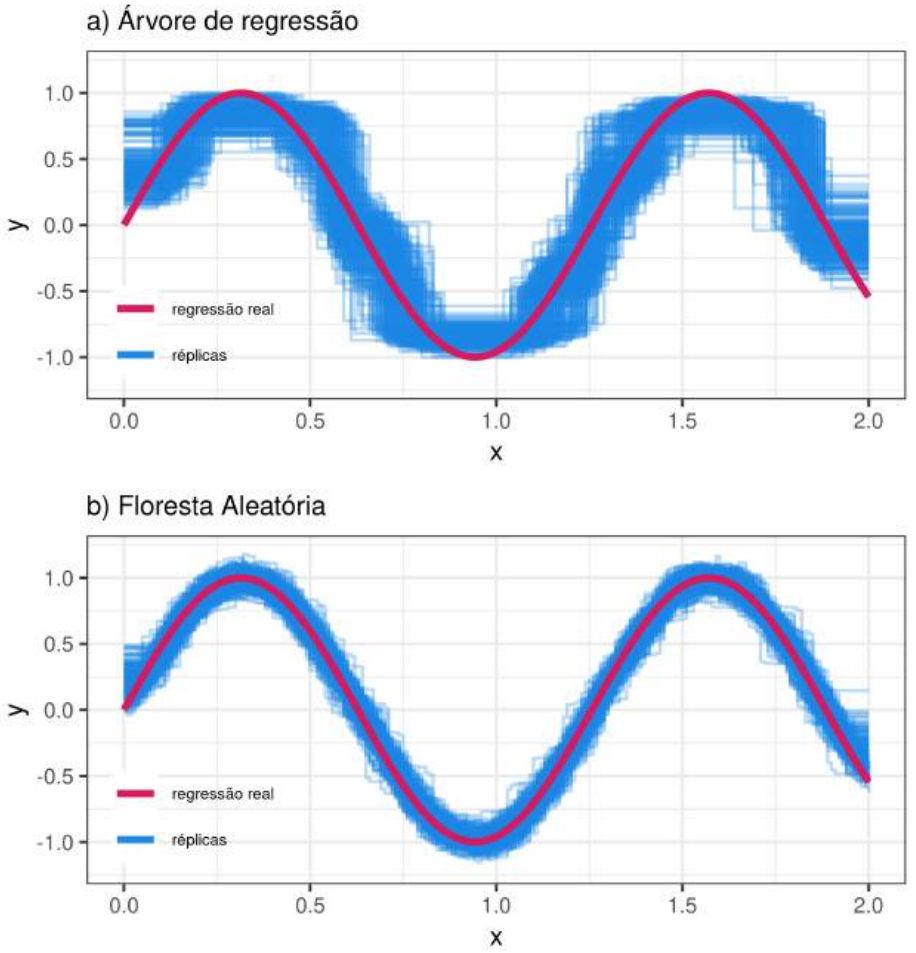


Figura 4.15: Ajustes de árvore de regressão e florestas aleatórias para diferentes conjuntos de dados vindo da mesma função de regressão.

4.10 Boosting

Assim como florestas aleatórias e o *bagging*, o *boosting* também consiste na agregação de diferentes estimadores da função de regressão. Contudo, esta combinação é feita de outra forma. Existem diversas variações e implementações de *boosting*. Abordaremos aqui a forma mais usual.

No *boosting*, o estimador $g(\mathbf{x})$ é construído incrementalmente. Inicialmente, atribuímos o valor de $g(\mathbf{x}) \equiv 0$. Este estimador possui alto viés, mas baixa variância (a saber, zero). A cada passo, atualizaremos o valor de g de modo a diminuir o viés e aumentar a variância da nova função. Isto é feito adicionando-se a g uma função que prevê os resíduos $r_i = Y_i - g(\mathbf{x}_i)$. Uma forma de se fazer isso é com a utilização de uma árvore de regressão. É importante que essa árvore tenha profundidade pequena de modo a evitar o super-ajuste. Além disso, ao invés de simplesmente adicionar essa função por completo, adicionamos ela multiplicada por λ (chamado de *learning rate*), um fator entre 0 e 1 que tem por finalidade evitar o super-ajuste. Formalmente, essa versão de *boosting* consiste no seguinte algoritmo:

1. Definimos $g(\mathbf{x}) \equiv 0$ e $r_i = y_i \ \forall i = 1, \ldots, n$.

2. Para $b = 1, \ldots, B$:

   (a) Ajustamos uma árvore com p folhas para $(\mathbf{x}_1, r_1), \ldots, (\mathbf{x}_n, r_n)$. Seja $g^b(\mathbf{x})$ sua respectiva função de predição.

   (b) Atualizamos g e os resíduos: $g(\mathbf{x}) \leftarrow g(\mathbf{x}) + \lambda g^b(\mathbf{x})$ e $r_i \leftarrow Y_i - g(\mathbf{x})$.

3. Retornamos o modelo final $g(\mathbf{x})$

Assim, os *tuning parameters* do *boosting* são B, p e λ. Tipicamente λ é pequeno (por exemplo, 0,01), $B \approx 1000$ e p é da ordem de 2 ou 4.

Para ilustrar o impacto dos hiperparâmetros desse algoritmo, considere novamente os dados apresentados na Figura 4.13. A Figura 4.16 mostra as regressões estimadas com um boosting composto por 100 árvores com apenas uma divisão cada (o que é chamado de *stump*). Cada painel ilustra o ajuste para um taxa de aprendizagem λ em $\{0.001, 0.01, 0.1, 0.2, 0.5, 1\}$. Note que λ grande leva a superajuste, enquanto que λ pequeno leva a subajuste.

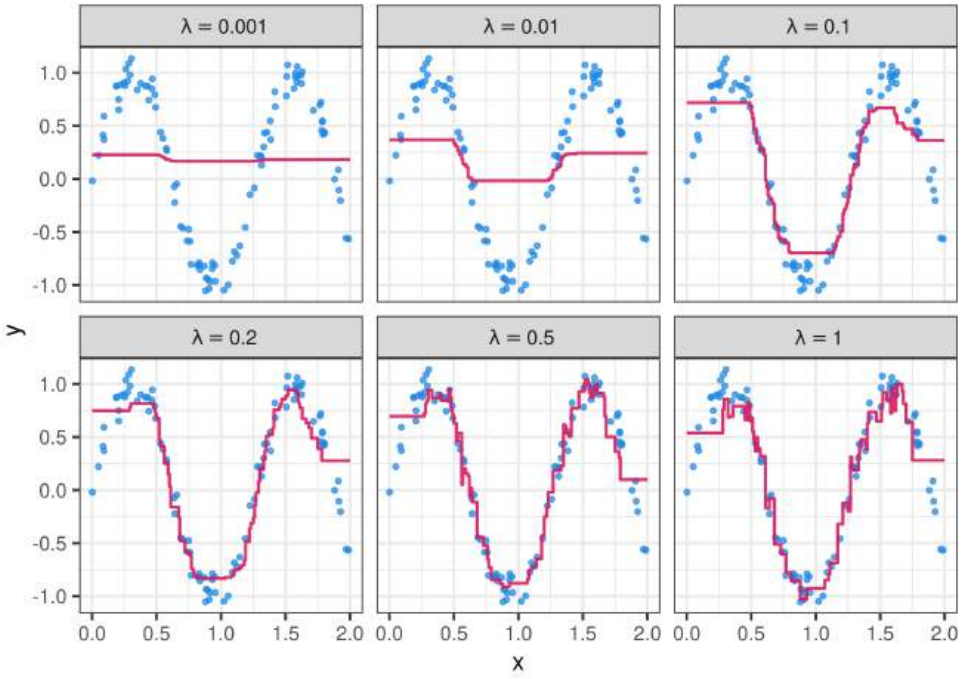


Figura 4.16: Aplicação de diversos valores de λ para boosting.

O *boosting* não necessariamente é feito com árvores, mas em geral são utilizados estimadores "fracos" (como, por exemplo, uma regressão com poucos parâmetros).

No R, o *boosting* pode ser implementando com o pacote `bst`. A Figura 4.17 ilustra como o risco estimado varia conforme o valor de B (número de iterações) utilizado. Em geral, B é escolhido via *early stopping*, isto é, pára-se de adicionar iterações que o risco estimado do método em um conjunto de validação começa a aumentar.

```
library(bst)
ajuste <- cv.bst(x = x_treinamento,
                 y = y_treinamento,
                 learner = "tree",
                 control.tree = list(maxdepth = 3),
                 ctrl = bst_control(mstop = 10000, nu = 0.02),
                 K = 2, se = FALSE)
```

```
predito <- predict(ajuste, x_teste)
```

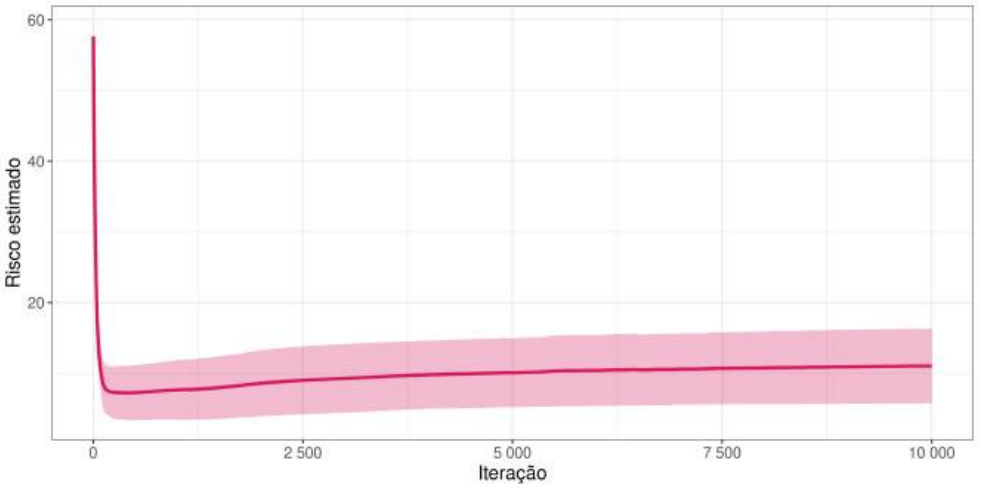


Figura 4.17: Diminuição do risco estimado por validação cruzada para a função de regressão segundo o número de iterações do algoritmo boosting.

Existem diferentes implementações e variações de boosting. Uma particular que vem ganhando bastante popularidade por seu ótimo desempenho é o XGBoost (Chen e Guestrin, 2016), que no R é implementando pelo pacote `xgboost`; veja o Exemplo 4.4 para mais detalhes.

4.11 Redes Neurais Artificiais

> A man should look for what is, and not for what he thinks should be.
>
> Albert Einstein

Redes neurais artificiais são um conceito bastante antigo em inteligência artificial (McCulloch e Pitts, 1943; Rosenblatt, 1958). Matematicamente, no contexto de regressão, trata-se de um estimador não linear de $r(\mathbf{x})$ que pode ser representado graficamente por uma estrutura como a da Figura 4.18.

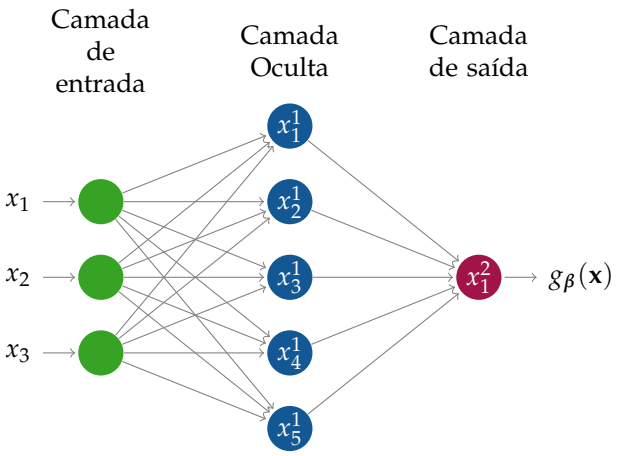


Figura 4.18: Exemplo de rede neural com uma camada oculta.

Os nós do lado esquerdo da figura representam as *entradas* da rede, isto é, cada uma das covariáveis do banco de dados (neste caso, há três delas). Os nós da segunda camada são os chamados de nós da camada oculta da rede. Cada flecha representa um peso (parâmetro) β. Cada nó nessa camada representa uma *transformação* dos nós das variáveis da camada anterior.

Para o caso dessa rede, se $\mathbf{x} = (x_1, x_2, x_3)$ é o vetor de entrada, então um dado neurônio j da camada oculta é calculado como

$$x_j^1 := f\left(\beta_{0,j}^0 + \sum_{i=1}^{3} \beta_{i,j}^0 x_i^0\right) \qquad \text{em que } x_i^0 = x_i \text{ para } i = 1,2,3,$$

em que f é uma função definida pelo usuário, chamada de *função de ativação*. O índice superescrito nessa equação denota a camada da rede. Calculados os valores de x_j^1 para todo neurônio j da camada oculta, pode-se então calcular a saída do modelo. Para esse exemplo, essa saída é dada por

$$x_1^2 := f\left(\beta_{0,j}^1 + \sum_{i=1}^{5} \beta_{i,1}^1 x_i^1\right) =: g_\beta(\mathbf{x}).$$

Esse procedimento é representado com mais detalhes na Figura 4.19.

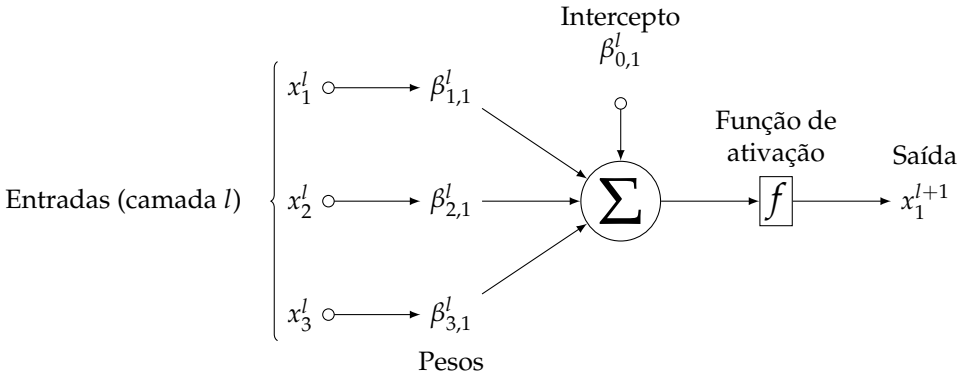


Figura 4.19: Exemplo do processamento que ocorre dentro de um neurônio situado na camada $l+1$. A saída x_1^{l+1} é dada por $f(\beta^l_{0,j} + \sum_{i=1}^{3} \beta^l_{i,1} x^l_i)$.

Algumas escolhas comuns para a função de ativação são:

- **identidade:** $f(z) = z$
- **logística:** $f(z) = \frac{1}{1+\mathbf{e}^{-z}}$
- **tangente hiperbólica:** $f(z) = \frac{\mathbf{e}^{z} - \mathbf{e}^{-}z}{\mathbf{e}^{z} + \mathbf{e}^{-z}}$
- **ReLu**(rectified linear): $f(z) = \max\{0, z\}$
- **Leaky ReLu:** $f(z) = \begin{cases} 0.01z & \text{se } z < 0 \\ z & \text{se } z \geq 0 \end{cases}$

De forma geral, uma rede neural pode ter várias camadas ocultas e um número diferente de neurônios em cada camada. Essas são escolhas feitas pelo usuário. A Figura 4.20 apresenta o esquema de uma rede neural com quatro camadas ocultas com seis neurônios cada.

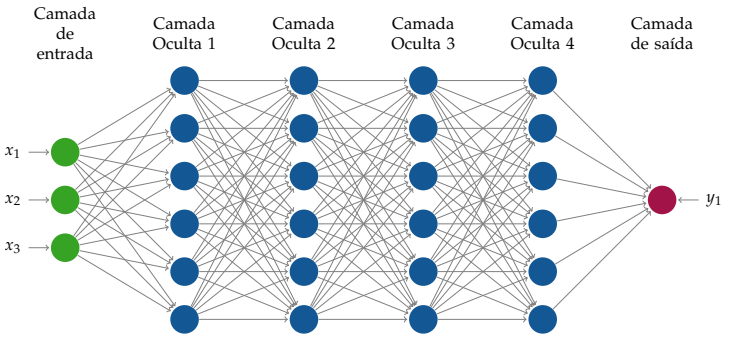


Figura 4.20: Exemplo de uma rede neural *feed forward*.

A propagação das covariáveis de entrada na rede neural é feita sequencialmente nessas camadas. Considere uma rede com H camadas ocultas, cada uma com d_h neurônios ($h = 1,\ldots,H$), seja $\beta^l_{i,j}$ o peso atribuído à conexão entre a entrada i na camada l e a saída j na camada $l+1$, $l = 0,\ldots,H$. Aqui $h = 0$ denota a camada de entrada, e $H+1$ a camada de saída. O estimador de uma rede neural artificial para a função de regressão tem a forma

$$g(\mathbf{x}) = x_1^{H+1} = f\left(\beta^H_{0,1} + \sum_{i=1}^{d_H} \beta^H_{i,1} x_i^H\right) \tag{4.22}$$

em que, para todo $l = 1,\ldots,H$ e $j = 1,\ldots,d_{l+1}$,

$$x_j^{l+1} = f\left(\beta^l_{0,j} + \sum_{i=1}^{d_l} \beta^l_{i,j} x_i^l\right).$$

É interessante notar que, se $f(z) = z$ e não há nenhuma camada oculta, uma rede neural representa uma regressão linear usual.

Uma outra forma de se representar uma rede neural se dá via notação matricial. Para $l = 0,\ldots,H$, seja

$$\mathbb{H}_l = \begin{pmatrix} \beta^l_{0,1} & \beta^l_{1,1} & \cdots & \beta^l_{d_l,1} \\ \vdots & \vdots & \ddots & \vdots \\ \beta^l_{0,d_{l+1}} & \beta^l_{1,d_{l+1}} & \cdots & \beta^l_{d_l,d_{l+1}} \end{pmatrix}.$$

Nesse caso, $d_0 = d$ é o número de neurônios na camada de entrada e d_{H+1} é o

número de neurônios na camada de saída. O estimador da Equação 4.22 pode ser escrito como a seguinte composição de funções:

$$g(\mathbf{x}) = f\left(\mathbb{H}_H \ldots \tilde{f}\left(\mathbb{H}_1 \tilde{f}\left(\mathbb{H}_0 \tilde{\mathbf{x}}\right)\right)\ldots\right),$$

em que $\tilde{\mathbf{x}} = (1, \mathbf{x})$ e $\tilde{f}(\mathbf{y}) = (1, f(\mathbf{y}))$.

A estrutura de redes neurais aqui apresentada pode ser bastante generalizada. Por exemplo, cada função f aplicada em cada camada pode ser diferente. Além disso, a saída da rede não precisa ser um único número real. Redes neurais também permitem estruturas mais complexas como retroalimentação (saídas de neurônios que voltam para entradas de camadas anteriores) e outros. Para uma análise mais aprofundada, veja Goodfellow et al. (2016).

4.11.1 Estimação: *Backpropagation*

Para estimar os coeficientes $\beta_{i,j}^l$ na Equação 4.22, deve-se especificar uma função objetivo a ser minimizada. No caso de regressão linear, utiliza-se, em geral, o erro quadrático médio,

$$EQM(g_\beta) = \frac{1}{n}\sum_{k=1}^{n}(g_\beta(\mathbf{x}_k) - y_k)^2,$$

em que a notação g_β é usada para enfatizar a dependência de g em seus parâmetros β. Pode-se também introduzir uma penalização a esse objetivo.

O vetor β que minimiza o $EQM(g_\beta)$ não possui solução analítica. Contudo, pode-se utilizar métodos numéricos para aproximá-lo. Um método que tem se destacado no contexto de redes neurais é o *backpropagation*, que nada mais é do que um método de gradiente descendente executado em uma ordem específica: primeiro atualizam-se os coeficientes da última camada, depois se atualizam os coeficientes da camada anterior a essa e assim por diante (por isso o nome *backpropagation*). Mais especificamente, cada passo do backpropagation consiste em fazer, para $l = H, \ldots, 0$, a seguinte atualização:

$$\beta_{i,j}^l \leftarrow \beta_{i,j}^l - \eta \frac{\partial EQM(g_\beta)}{\partial \beta_{i,j}^l}, \qquad \text{com} j = 1, \ldots, d_{l+1} \text{ e } i = 0, \ldots, d_l.$$

O truque é que $\frac{\partial EQM(g_\beta)}{\partial \beta_{i,j}^l}$ depende apenas dos valores dos coeficientes das cama-

das posteriores (isto é, $\beta_{i,j}^{l'}$ para $l' \geq l$) e é simples de ser calculado. Por exemplo, para uma função de ativação $f(z) = z$ na camada de saída e $j > 0$,

$$\frac{\partial EQM(g_\beta)}{\partial \beta_{j,1}^H} = \frac{\partial}{\partial \beta_{j,1}^H}\left[\frac{1}{n}\sum_{k=1}^{n}(\beta_{0,1}^H + \sum_{i=1}^{d_H}\beta_{i,1}^H x_{k,i}^H - y_k)^2\right] = \\ = \frac{2}{n}\sum_{k=1}^{n}(\beta_{0,1}^H + \sum_{i=1}^{d_H}\beta_{i,1}^H x_{k,i}^H - y_k)x_{j,1}.$$

Para mais detalhes sobre o *backpropagation* veja, por exemplo, Rojas (2013).

4.11.2 Deep Learning

Na última década, redes neurais ressurgiram como uma área ativa dentro da comunidade de aprendizado de máquina. Um dos principais fatores que promoveram esse ressurgimento foi o desenvovimento tecnológico computacional, que tornou o processo em paralelo (em particular em GPUs) barato e acessível. O método do *backpropagation* foi adaptado para se aproveitar dessa tecnologia através do gradiente descendente estocástico. Diversas variações e melhorias, tanto no processo de estimação de uma rede quanto na definição de sua arquitetura, vêm sendo desenvolvidas nos últimos anos. Para uma revisão abrangente dessa área, veja Goodfellow et al. (2016).

O pacote `keras`, utilizando diversas ferramentas desenvolvidas para deep learning, implementa redes neurais no R. A seguir apresentamos um exemplo.

Exemplo 4.3 (Bike sharing). Neste exemplo consideramos o conjunto de dados *bike sharing* (http://archive.ics.uci.edu/ml/datasets/bike+sharing+dataset). Nesse conjunto são indicadas as quantidades de bicicletas que foram emprestadas em algumas centrais a cada hora. A Figura 4.21 mostra como o risco estimado no conjunto de treinamento e de validação variam conforme o número de iterações do gradiente descendente estocástico cresce.

```
library(rsample)
library(keras)

dados <- read_csv("../dados/hour.csv")
```

```
dados <- dados %>%
  select(-instant,
         -dteday, -yr, -mnth, -casual, -registered) %>%
  mutate_at(vars(season, holiday, weekday, workingday),
            as.character)

dados <- data.frame(model.matrix(cnt ~ . -1, dados),
                    cnt = dados$cnt) %>%
  mutate_at(vars(season1:windspeed), scale)

set.seed(400)

split <- initial_split(dados, prop = 0.8)

treinamento <- training(split)
teste <- testing(split)

x_treino <- select(treinamento, -cnt) %>% as.matrix()
y_treino <- treinamento$cnt

x_teste <- select(teste, -cnt) %>% as.matrix()
y_teste <- teste$cnt

# especificação do modelo (uma rede feedforward com
# duas camadas ocultas com 8 neuronios cada)
modelo <- keras_model_sequential() %>%
  layer_dense(units = 8, activation = "relu",
              input_shape = ncol(x_treino)) %>%
  layer_dense(units = 8, activation = "relu") %>%
  layer_dense(units = 1)

# especificação da função objetivo e como ela será minimizada
modelo %>% compile(
  loss = "mse",
```

```
  optimizer = optimizer_rmsprop(),
  metrics = list("mean_absolute_error")
)


historico <- modelo %>% fit(
  x_treino,
  y_treino,
  epochs = 1200,
  validation_split = 0.2,
  verbose = FALSE
)
```

A seguir, apresentamos os riscos estimados nos conjuntos de treinamento e validação de acordo com o número de iterações.

```
plot(historico, metrics = "loss") +
  theme_bw() + theme(legend.position = "top")
```

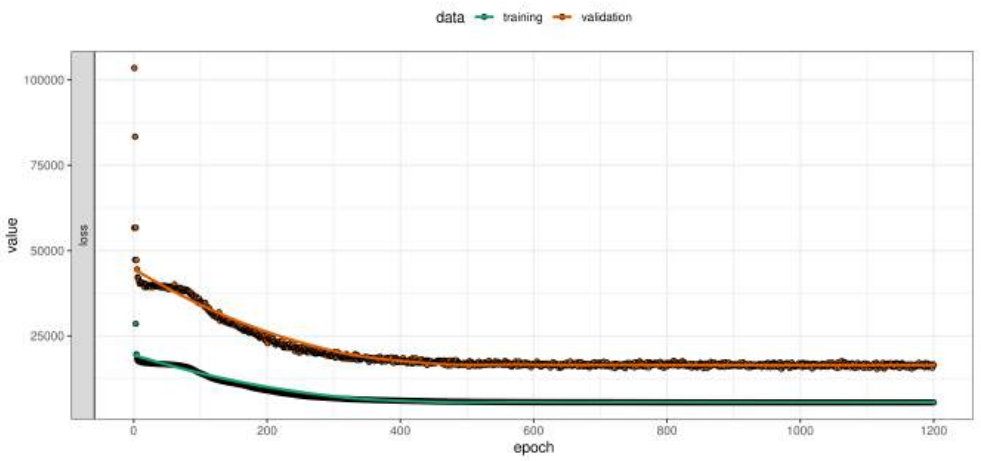


Figura 4.21: Risco estimado no conjunto de treinamento e no conjunto de validação.

```
# valor predito no conjunto de teste
predito <- modelo %>%
  predict(x_teste)
```

□

4.12 Exemplo

Exemplo 4.4 (Amazon Fine Food Reviews). Nesse exemplo retomamos o problema de fazer predições de notas dadas em resenhas da Amazon com base no conteúdo (texto) da resenha (Exemplo 3.3). Dessa vez ajustaremos alguns métodos não paramétricos.

Como alguns dos pacotes do R não permitem tabalhar com matrizes esparsas, converteremos o objeto `dtmMatrix` em uma matriz não esparsa:

```
dtmMatrix_non_sparse <- as.matrix(dtmMatrix)
```

A seguir, apresentamos os código para os métodos não paramétricos considerados.

```
# KNN
k_grid <- round(seq(1, 100, length.out = 20))
error <- vector("numeric", length(k_grid))

for (i in seq_along(k_grid)) {

  fit <- FNN::knn.reg(train = dtmMatrix_non_sparse[tr,],
                      y = dados$Score[tr],
                      k = k_grid[i])

  error[i] <- mean((fit$pred - dados$Score[tr])^2)

}

pred_knn <- FNN::knn.reg(train = dtmMatrix_non_sparse[tr,],
```

```
                          y = dados$Score[tr],
                          test = dtmMatrix_non_sparse[-tr,],
                          k = k_grid[which.min(error)])$pred

# XGBoost
xgb_model <- xgb.train(data = xgb.DMatrix(
  dtmMatrix[tr,],
  label = dados$Score[tr]),
  nrounds = 50)

pred_xgboost <- predict(xgb_model, dtmMatrix[-tr,])

# Tree
fit <- rpart(dados$Score[tr] ~ dtmMatrix_non_sparse[tr,],
             method = "anova")

melhor_cp <- fit$cptable[which.min(fit$cptable[, "xerror"]),
                         "CP"]

pfit <- rpart::prune(fit, cp = melhor_cp)

pred_arvore <- predict(pfit,
                  as.data.frame(dtmMatrix_non_sparse[-tr,]))
```

Comparando os erros quadráticos médios (Tabela 4.1), notamos que o XGBoost apresentou o melhor desempenho preditivo, comparável ao do lasso e da regressão ridge (Tabela 3.6). Note que o XGBoost possui diversos *tuning parameters* que podem ser ajustados para levar a um desempenho melhor. Em particular, recomendamos usar *early stopping* no treinamento. Como esperado, a performance preditiva da árvore foi inferior à dos demais métodos. O KNN também apresentou um desempenho ruim. O motivo disso é que, nesse exemplo, várias covariáveis são irrelevantes (ou seja, não estão associadas à nota dada pelo usuário) e o KNN não faz seleção de variáveis. Assim, todas elas têm o mesmo papel no classificador, enquanto todos os outros métodos fazem algum tipo de seleção. Voltaremos a esse assunto sob uma perspectiva teórica no Capítulo 5.

Uma vantagem das implementações do XGBoost e do Lasso no R é que elas permitem trabalhar com matrizes esparsas e são computacionalmente rápidas. Isso permite que apliquemos uma versão desses métodos em um conjunto de treinamento maior ($n = 400{,}000$). Fazendo isso, o XGBoost consegue obter um erro menor que o dos demais métodos (Tabela 4.2). Note que o ganho do XGBoost, ao se aumentar o tamanho do conjunto de treinamento, é muito maior que o do lasso. De fato, enquanto para um n pequeno ambos têm um desempenho parecido, para n grande o XGBoost é substancialmente maior. Isso ocorre porque o lasso é um método paramétrico. Assim, com um n não tão grande já se extrai o máximo que ele pode oferecer (ou seja, ele já está próximo ao oráculo linear). Por outro lado, um método não paramétrico como o XGBoost é muito mais flexível e, portanto, se beneficía mais de uma amostra maior.

Tabela 4.1: Desempenho preditivo (erro quadrático médio e seu erro padrão) em relação ao conjunto de validação dos modelos KNN, XGBoost e árvores.

Modelo	**EQM**	**EP**
KNN	1.541	0.029
XGBoost	1.117	0.021
Árvore	1.911	0.031

Tabela 4.2: Desempenho preditivo (erro quadrático médio e seu erro padrão) em relação ao conjunto de validação dos modelos XGBoost e Lasso para um conjunto de treinamento maior.

Modelo	**EQM**	**EP**
XGBoost Large	0.783	0.015
Lasso Large	0.948	0.018

□

4.13 Um Pouco de Teoria

> Experience without theory is blind, but theory without experience is mere intellectual play.
>
> Immanuel Kant

Uma maneira de estudar o desempenho de um método é avaliando sua (taxa de) convergência. Em particular, pode-se verificar sob quais circunstâncias um dado estimador converge. Com a finalidade de ilustrar como isso pode ser feito no contexto não paramétrico, investigaremos dois métodos: os k-vizinhos mais próximos (Seção 4.13.1) e séries ortogonais (Seção 4.13.2).

Para estudar a taxa de convergência de um estimador não-paramétrico, é necessário assumir que a função de regressão $r(\mathbf{x})$ real é *suave*; caso contrário ela pode ser extremamente difícil de ser estimada. Existem várias medidas de suavidade (Tsybakov, 2008); utilizaremos aqui duas noções, uma para cada uma das análises que seguem.

4.13.1 k-vizinhos Mais Próximos

Para analisar o método KNN, utilizarmos o conceito de funções Lipschitz para capturar a suavidade da função de regressão. Mais especificamente, iremos assumir que a regressão real r é L-Lipschitz, isto é, assumiremos que existe L tal que, para todo $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$,

$$|r(\mathbf{x}) - r(\mathbf{x}')| \leq L||\mathbf{x} - \mathbf{x}'||_2.$$

Iniciamos com o seguinte resultado, que decorre do Lema 6.4 e do Teorema 6.2 de Györfi et al. (2006).

Lema 1. *Se* $\mathbf{X}$ *é limitado e* $d \geq 3$, *então*

$$\mathbb{E}\left[\left(\frac{1}{k}\sum_{i \in \mathcal{N}_\mathbf{x}} ||\mathbf{X}_i - \mathbf{X}||_2\right)^2\right] \leq C\left(\frac{k}{n}\right)^{2/d}.$$

Essencialmente, o Lema 1 indica como a distância média entre as k observações mais próximas a $\mathbf{X}$ é limitada pelo tamanho amostral.

Teorema 7. *Seja $\widehat{r}_k(\mathbf{X})$ o estimador da Equação 4.2. Sob as suposições do Lema 1 e se $\sup_{\mathbf{x}} \mathbb{E}[Y|\mathbf{x}] := \sigma^2 < \infty$ e r é L-Lipschitz, então*

$$\mathbb{E}[(\widehat{r}_k(\mathbf{X}) - r(\mathbf{X}))^2] \leq K\left(\frac{k}{n}\right)^{2/d} + \frac{\sigma^2}{k},$$

em que K não depende de k nem n. Assim, se tomamos $k \asymp n^{\frac{2}{2+d}}$, obtemos

$$\mathbb{E}[(\widehat{r}_k(\mathbf{X}) - r(\mathbf{X}))^2] \leq K' n^{-\frac{2}{2+d}}$$

Demonstração. Seja $\tilde{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{x})$. Pela decomposição viés-variância (Seção 1.5.2), temos que

$$\begin{aligned}
\mathbb{E}[(\widehat{r}_k(\mathbf{x}) - r(\mathbf{x}))^2|\tilde{\mathbf{x}}] &= (\mathbb{E}[\widehat{r}_k(\mathbf{x})|\tilde{\mathbf{x}}] - r(\mathbf{x}))^2 + \mathbb{E}[(\widehat{r}_k(\mathbf{x}) - \mathbb{E}[(\widehat{r}_k(\mathbf{x})|\tilde{\mathbf{x}}])^2|\tilde{\mathbf{x}}] \\
&= \left(\mathbb{E}\left[\frac{1}{k}\sum_{i\in\mathcal{N}_{\mathbf{x}}} Y_i|\tilde{\mathbf{x}}\right] - r(\mathbf{x})\right)^2 + \mathbb{V}[\widehat{r}_k(\mathbf{x})|\tilde{\mathbf{x}}] \\
&\leq \left(\frac{1}{k}\sum_{i\in\mathcal{N}_{\mathbf{x}}} r(\mathbf{x}_i) - r(\mathbf{x})\right)^2 + \frac{\sigma^2}{k} \leq \left(\frac{L}{k}\sum_{i\in\mathcal{N}_{\mathbf{x}}} ||\mathbf{x}_i - \mathbf{x}||_2\right)^2 + \frac{\sigma^2}{k},
\end{aligned}$$

em que a última desigualdade decorre do fato de r ser L-Lipschitz.

Assim, pela lei da esperança total e Lema 1, segue que

$$\begin{aligned}
\mathbb{E}[(\widehat{r}_k(\mathbf{X}) - r(\mathbf{X}))^2] &= \mathbb{E}[\mathbb{E}[(\widehat{r}_k(\mathbf{X}) - r(\mathbf{X}))^2|\tilde{\mathbf{X}}]] \\
&\leq \mathbb{E}\left[\left(\frac{L}{k}\sum_{i\in\mathcal{N}_{\mathbf{X}}} ||\mathbf{X}_i - \mathbf{X}||_2\right)^2\right] + \frac{\sigma^2}{k} \\
&\leq C'\left(\frac{k}{n}\right)^{2/d} + \frac{\sigma^2}{k}.
\end{aligned}$$

□

Note que o Teorema 7 indica que, quando k é grande, o termo associado ao viés do estimador dado pelo KNN (isto é, $C'(k/n)^{2/d}$) é alto, enquanto o termo associado à sua variância (isto é, σ^2/k) é baixo. Para se ter um bom equilíbrio entre esses termos, deve-se tomar $k \asymp n^{\frac{2}{2+d}}$. Discutiremos essa taxa de convergência mais a fundo na Seção 5.1; em particular argumentaremos que o limitante superior mostrado no Teorema 7 não pode ser melhorado, isto é, não se pode encontrar um limitante supe-

rior menor que o encontrado.

4.13.2 Séries Ortogonais

Nessa subseção iremos derivar taxas de convergência para o método de séries ortogonais. Nos restringiremos ao caso em que $\mathbf{X} \in [0,1]$ e tem densidade uniforme neste intervalo. Novamente, para estudar a convergência do estimador, iremos também supor que a função de regressão $r(\mathbf{x})$ real é suave. No entanto, utilizaremos o conceito de derivadas para definir suavidade, uma vez que essas medem mudanças bruscas de $r(\mathbf{x})$ como função de $\mathbf{x}$. Diremos que $r(\mathbf{x})$ é suave se $r(\mathbf{x}) \in L^2([0,1])$ e

$$\int_0^1 (D^m r(\mathbf{x}))^2 d\mathbf{x} \leq K_1.$$

Quanto maior m, mais suave r é. Este conceito de suavidade está ligado a espaços de Sobolev, veja Wasserman (2006).

Um resultado fundamental mostra que, se $r(\mathbf{x})$ é suave, usar apenas os primeiros termos da expansão $\sum_{j\in\mathbb{N}} \beta_j \psi_j(\mathbf{x})$ é suficiente para aproximar $r(\mathbf{x})$ bem. Isso é formalmente descrito no lema a seguir.

Lema 2. *Se $r(\mathbf{x}) \in L^2([0,1])$ é tal que $\int_0^1 (D^m r(\mathbf{x}))^2 d\mathbf{x} \leq K_1$, então*

$$\int_0^1 \left(\sum_{j=0}^{J} \beta_j \psi_j(\mathbf{x}) - r(\mathbf{x}) \right)^2 d\mathbf{x} = \sum_{j \geq J+1} \beta_j^2 \leq \frac{K_1}{J^{2m}},$$

em que β_j's são os coeficientes de expansão de $r(\mathbf{x})$ na base de Fourier $(\psi_j)_{j\geq 0}$.

Note que, quanto maior m (a suavidade de r), menor o erro em se utilizar J termos para aproximar $r(\mathbf{x})$. É exatamente por isso que o método de séries ortogonais funciona bem: quando r é suave, um truncamento do tipo usado na Equação 4.1 leva a um viés baixo.

O resultado a seguir combina o viés obtido no Lemma 2 com a variância do estimador de modo a obter a taxa de convergência do método de séries ortogonais.

Teorema 8. *Seja $\widehat{r}_J(\mathbf{x})$ o estimador da Equação 4.1 com base na série de Fourier. Se $r \in L^2([0,1])$ é tal que $\int_0^1 (D^m r(\mathbf{x}))^2 d\mathbf{x} \leq K_1$ e $\mathbb{V}[Y] < \infty$, então o risco desse estimador satisfaz*

$$\mathbb{E}[(\widehat{r}_J(\mathbf{X}) - r(\mathbf{X}))^2] \leq K_1 \frac{1}{J^{2m}} + K_2 \frac{J}{n},$$

em que K_2 não depende de J nem n. Assim, se tomarmos $J = Cn^{\frac{1}{2m+1}}$, teremos

$$\mathbb{E}[(\widehat{r}_J(\mathbf{X}) - r(\mathbf{X}))^2] \leq Kn^{-\frac{2m}{2m+1}}.$$

Demonstração. Pela decomposição viés-variância (Seção 1.5.2) e pela lei da esperança total, temos que

$$\mathbb{E}[(\widehat{r}_J(\mathbf{X}) - r(\mathbf{X}))^2] = \mathbb{E}\left\{\mathbb{E}\left[\left(r(\mathbf{X}) - \mathbb{E}[\widehat{r}_J(\mathbf{X})|\mathbf{X}]\right)^2|\mathbf{X}\right]\right\} + \mathbb{E}\left\{\left[\mathbb{V}[\widehat{r}_J(\mathbf{X})|\mathbf{X}]\right]\right\} \quad (4.23)$$

Notando que $\mathbb{E}[\widehat{\beta}_j] = \beta_j$, temos que $\mathbb{E}[\widehat{r}_J(\mathbf{x})] = \sum_{j=0}^{J}\beta_j\psi_j(\mathbf{x})$ e, assim, o termo de viés pode ser escrito como

$$\mathbb{E}\left\{\mathbb{E}\left[\left(r(\mathbf{X}) - \mathbb{E}[\widehat{r}_J(\mathbf{X})|\mathbf{X}]\right)^2|\mathbf{X}\right]\right\} = \int_0^1\left(\sum_{j>J}\beta_j\psi_j(\mathbf{x})\right)^2 dx \leq K_1\frac{1}{J^{2m}}, \quad (4.24)$$

em que a última desigualdade segue do Lema 2.

Para desenvolver o termo da variância, note que, para todo $\mathbf{x} \in \mathbb{R}$,

$$\begin{aligned}\mathbb{V}\left[\sum_{j=0}^{J}\widehat{\beta}_j\psi_j(\mathbf{x})\right] &= \mathbb{E}\left[\left(\sum_{j=0}^{J}\widehat{\beta}_j\psi_j(\mathbf{x}) - \sum_{j=0}^{J}\beta_j\psi_j(\mathbf{x})\right)^2\right] = \mathbb{E}\left[\left(\sum_{j=0}^{J}(\widehat{\beta}_j - \beta_j)\psi_j(\mathbf{x})\right)^2\right] \\ &= \sum_{i,j<J}\mathbb{E}\left[(\widehat{\beta}_i - \beta_i)(\widehat{\beta}_j - \beta_j)\right]\psi_i(\mathbf{x})\psi_j(\mathbf{x})\end{aligned}$$

Assim, utilizando novamente a ortogonalidade da base, o termo de variância pode ser decomposto como

$$\begin{aligned}\mathbb{E}\left\{\mathbb{E}\left[\mathbb{V}[\widehat{r}_J(\mathbf{X})]|\mathbf{X}\right]\right\} &= \int_0^1\sum_{i,j<J}\mathbb{E}\left[(\widehat{\beta}_i - \beta_i)(\widehat{\beta}_j - \beta_j)\right]\psi_i(\mathbf{x})\psi_j(\mathbf{x})dx \\ &= \sum_{j=0}^{J}\mathbb{E}[(\widehat{\beta}_j - \beta_j)^2] = \sum_{j=0}^{J}\mathbb{V}[\widehat{\beta}_j] = \sum_{j=0}^{J}\frac{1}{n}\mathbb{V}[Y\psi_j(\mathbf{X})] \leq K_2\frac{J}{n},\end{aligned} \quad (4.25)$$

em que K_2 é uma constante que não depende de j e J e a última desigualdade segue do fato que $|\psi_j(\mathbf{X})| < \sqrt{2}$ e $\mathbb{V}[Y]$ é finita.

Assim, combinando as Equações 4.24 e 4.25 via Eq. 4.23, temos que

$$\mathbb{E}[(\widehat{r}_J(\mathbf{X}) - r(\mathbf{X}))^2] \leq K_1 \frac{1}{J^{2m}} + K_2 \frac{J}{n}.$$

□

O teorema anterior mostra que $n^{-\frac{2m}{2m+1}}$ é um limitante superior para a taxa de convergência do estimador baseado em séries ortogonais quando assumimos que $r(\mathbf{x})$ possui m derivadas integráveis e $\mathbf{x} \in \mathbb{R}$. Pode-se mostrar que esse limitante não pode ser melhorado (veja a Seção 5.1). Além disso, quando $\mathbf{x} \in \mathbb{R}^d$, essa taxa é dada por $n^{-\frac{2m}{2m+d}}$.[10] Essa taxa é ótima sob estas suposições, isso é, nenhum estimador pode ter taxas melhores que essa. Mais precisamente, $n^{-\frac{2m}{2m+d}}$ é a *taxa minimax* dentro dessa classe (Tsybakov, 2008). Discutiremos isso mais a fundo na Seção 5.1.

4.14 Resumo

Neste capítulo estudamos algumas classes de métodos não paramétricos que têm motivações muito distintas. Vimos que a vantagem de tais métodos é sua flexibilidade quando comparada com estimadores paramétricos: não é necessário assumir uma forma paramétrica para $r(\mathbf{x})$ para garantir a consistência de um estimador não paramétrico $\widehat{r}(\mathbf{x})$; necessita-se apenas que $r(\mathbf{x})$ seja suave (o que pode ser medido de diferentes maneiras). Essa flexibilidade, contudo, vem às custas de amostras maiores para se obter taxas de convergências razoáveis: enquanto métodos paramétricos tipicamente possuem taxas de convergência n^{-1} quando suas suposições são válidas, a taxa usual de métodos não paramétricos é $n^{-\frac{\alpha}{\alpha+d}}$ (em que α é a suavidade de r), que é muito mais lenta. Esse problema é ainda mais notório quando o número de covariáveis d é grande, assunto que será abordado no próximo capítulo.

[10] Para isso, assumimos, por exemplo, que todas as derivadas parciais de ordem m de r são L-Lipschitz (Györfi et al., 2006).

Capítulo 5

Métodos Não Paramétricos em Altas Dimensões

Como vimos no capítulo anterior, métodos não paramétricos, apesar de mais flexíveis, exigem tamanhos de amostras maiores quando comparados com métodos paramétricos. Isso ocorre pois métodos não paramétricos possuem taxas de convergência lentas. Neste capítulo verificaremos como tais métodos se comportam em altas dimensões. Veremos, em particular, que só poderemos fazer estimação não paramétrica de forma satisfatória se fizermos suposições que simplificam o problema. Duas suposições usuais que frequentemente valem na prática são esparsidade e redundância nas covariáveis.

5.1 Taxas de convergência e a maldição da dimensionalidade

A *maldição da dimensionalidade* (Bellman, 1966) é um problema enfrentado na utilização de estimadores em altas dimensões. Essencialmente, o problema é que, sem restrições adicionais, conforme a dimensão do espaço das covariáveis cresce, mais difícil fica estimar a função de regressão. Além disso, essa dificuldade cresce exponencialmente.

Mais precisamente: considere que temos d covariáveis e um tamanho amostral n. Argumentamos no Capítulo 4 que o risco de um estimador não-paramétrico $\widehat{r}^*$

frequentemente satisfaz

$$\mathbb{E}\left[(\widehat{r}^*(\mathbf{X}) - r(\mathbf{X}))^2\right] \leq \frac{K}{n^{\alpha/(\alpha+d)}}$$

para todo $r \in S$, em que K é uma constante e S é um conjunto de regressões suaves (por exemplo, com derivadas parciais de segunda ordem L-Lipschitz; o parâmetro α está ligado à noção de suavidade usada). Assim, vale que

$$\sup_{r \in S} \mathbb{E}\left[(\widehat{r}^*(\mathbf{X}) - r(\mathbf{X}))^2\right] \leq \frac{K}{n^{\alpha/(\alpha+d)}}. \tag{5.1}$$

Pode-se também mostrar que

$$\inf_{\widehat{r}} \sup_{r \in S} \mathbb{E}\left[(\widehat{r}(\mathbf{X}) - r(\mathbf{X}))^2\right] = \frac{K}{n^{4/(4+d)}},$$

isto é, $\frac{K}{n^{\alpha/(\alpha+d)}}$ é a taxa minimax para esse problema. Esse resultado, juntamente com a Equação 5.1, implica que

1. $\sup_{r \in S} \mathbb{E}\left[(\widehat{r}(\mathbf{X}) - r(\mathbf{X}))^2\right] = \frac{K}{n^{\alpha/(\alpha+d)}}$ (isto é, a desigualdade na Equação 5.1 é na realidade uma igualdade)
2. Se um estimador $\widehat{r}^*$ satisfaz a Equação 5.1 ele é ótimo em um sentido minimax, isto é, o risco mais alto que ele atinge para funções na classe S é o menor (pior) risco que poderia ser obtido por qualquer estimador. Em outras palavras,

   $$\sup_{r \in S} \mathbb{E}\left[(\widehat{r}^*(\mathbf{X}) - r(\mathbf{X}))^2\right] \leq \sup_{r \in S} \mathbb{E}\left[(\widehat{r}(\mathbf{X}) - r(\mathbf{X}))^2\right]$$

   para qualquer estimador $\widehat{r}$.

Segue da Equação 5.1 que, para conseguir obter um risco de no máximo δ, precisamos de uma amostra de tamanho ao menos

$$n \approx \left(\frac{K}{\delta}\right)^{(\alpha+d)/\alpha}.$$

Esse crescimento exponencial indica que, mesmo em dimensões moderadas, estimar satisfatoriamente uma função de regressão requer uma amostra grande. Veja na Figura 5.1 como o risco varia como função do tamanho amostral n para diferentes dimensões d e para $\alpha = 4$.

Suponha que desejamos estimar a função de regressão em um dado ponto $\mathbf{x}$. Intuitivamente, a maldição da dimensionalidade ocorre pois, quando d é grande, a probabilidade que exista *algum* elemento na amostra com covariáveis $\mathbf{X}_i$ é extremamente baixa mesmo para tamanhos de amostra moderados. Em outras palavras, *a vizinhança de* $\mathbf{x}$ *com alta probabilidade é vazia*. Logo, há pouca informação sobre $r(\mathbf{x})$. Veja mais detalhes em Wasserman (2006).

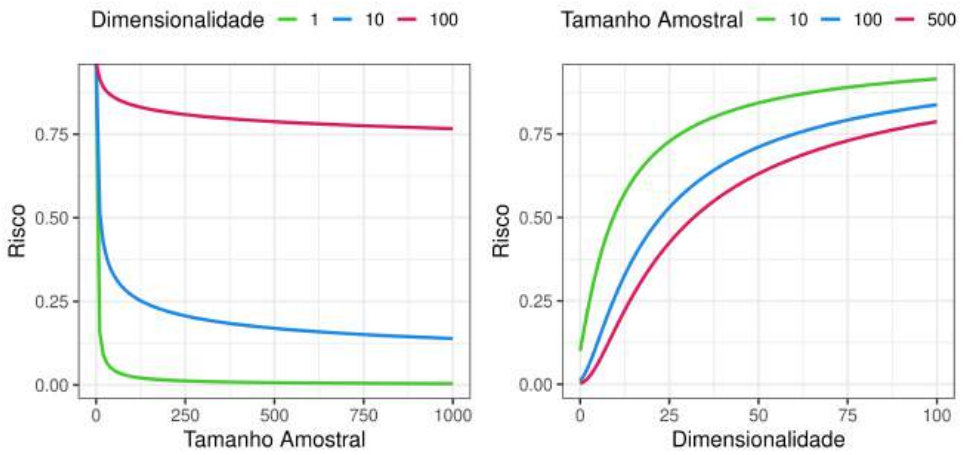


Figura 5.1: A maldição da dimensionalidade: quanto maior o número de covariáveis d, mais difícil é estimar r. Aqui tomamos $\alpha = 4$.

Assim, estimar uma função de regressão em problemas com dimensionalidade alta é um problema notoriamente difícil. O fenômeno da maldição da dimensionalidade indica que, sem suposições extras, é impossível resolver esse problema satisfatoriamente para tamanho de amostras realistas.

Nas próximas seções exxploraremos duas dessas suposições: (i) esparsidade e (ii) redundância entre as covariáveis.

5.1.1 Esparsidade

A suposição de esparsidade diz que a função de regressão $r(\mathbf{x})$ depende apenas de alguns x_i's. Isto é,

$$r(\mathbf{x}) = r\left((x_i)_{i \in S}\right),$$

em que $S \subset \{1,\ldots,d\}$. Assim, apesar de haver muitas covariáveis, poucas delas são relevantes para o problema. Sob essa hipótese, estimar $r(\mathbf{x})$ torna-se mais fácil. Esse é o caso, por exemplo, no exemplo da Amazon (4.4). Naquele caso, espera-se que várias palavras não estejam associadas à nota dada ao produto. É por esse motivo que é possível criar funções de predições razoáveis mesmo havendo uma dimensão grande no espaço das covariáveis.

5.1.2 Redundâcia

A suposição de redundância diz, intuitivamente, que as covariáveis $\mathbf{x}$ são altamente redundantes. Um exemplo trivial é o de que $x_1 = \ldots = x_d$, isto é, todas as covariáveis são iguais. Como um exemplo mais realista, considere que x_1 é a altura de um indivíduo, x_2 é seu peso e x_3 é seu índice de massa corpórea. Com quaisquer duas dentre essas três variáveis, conseguimos recuperar a terceira, de modo que essas variáveis podem ser reduzidas para apenas duas.

Outro caso em que redundância costuma ser uma suposição adequada é o de imagens. Neste contexto, cada covariável x_i é o pixel de uma figura (veja Seção A.1 para uma explicação do que são pixels). Espera-se, em geral, que pixels adjacentes tenham valores próximos, de modo que há, assim, muita redundância nas covariáveis associadas a uma imagem.

Existem diversas formas de formalizar a noção de redundância; uma muito usada é a de que $\mathbf{X}$ pertence a uma subvariedade de $\mathbb{R}^d$ com dimensão intrínseca baixa (Aswani et al., 2011). Novamente, sob essa hipótese, estimar $r(\mathbf{x})$ torna-se mais fácil.

A seguir veremos alguns métodos não paramétricos que são adequados para problemas com alta dimensionalidade.

5.2 k Vizinhos Mais Próximos e Regressão Linear Local

Tanto o estimador dos k vizinhos mais próximos (Seção 4.3) quanto a regressão linear local (Seção 4.5 para o caso $p = 1$) se adaptam automaticamente à dimensionalidade intrínseca das observações quando estas percentem a uma subvariedade de $\mathbb{R}^d$. Veja, por exemplo, Kpotufe (2011) e Bickel e Li (2007). Assim, quando se utiliza um desses estimadores, nenhuma adaptação é necessária para evitar a maldição da dimensionalidade quando x_i's são altamente redundantes. Mais precisamente, a taxa de convergência de ambos os estimadores é $\frac{K}{n^{4/(4+u)}}$, em que u é a dimensão

intrínseca da subvariedade a qual $\mathbf{X}$ pertence. Essa taxa é muito mais rápida que a vista anteriormente, $\frac{K}{n^{4/(4+d)}}$, especialmente se $u \ll d$.

5.3 Support Vector Regression

Support Vector Regression (Seção 4.6.4) com o kernel Gaussiano também têm bom desempenho sob a hipótese de redundância. Veja, por exemplo, Steinwart e Christmann (2008).

5.4 Séries Ortogonais

A abordagem de séries ortogonais apresentada na Seção 4.1 também pode ser adaptada para o cenário de dimensionalidade alta. A seguir apresentaremos a abordagem de séries espectrais.

5.4.1 Bases espectrais

Nesta seção estudamos uma extensão do método de séries ortogonais apresentado na Seção 4.1 que é mais adequada para dados com dimensionalidade alta. O estimador também consiste em uma expansão em termos de uma base ortogonal. Contudo, ao contrário do que foi estudado na Seção 4.1, aqui a base utilizada é construída com base nos dados. Em outras palavras, ao invés de utilizar, por exemplo, uma base de Fourier, utilizaremos uma base $\{\psi_j\}_{j\in\mathbb{N}}$ construída com base na amostra $\mathbf{x}_1, \ldots, \mathbf{x}_n$.

Para construir uma base espectral, começamos definindo um kernel de Mercer $K(\mathbf{x}, \mathbf{y})$ (veja Seção 4.6.1). Seja $P(\mathbf{x})$ a distribuição do vetor de covariáveis $\mathbf{X}$ e Considere o seguinte operador (Shi et al., 2009):

$$\begin{aligned} \mathbf{K} : \mathfrak{L}^2(\mathcal{X}, P) &\longrightarrow \mathfrak{L}^2(\mathcal{X}, P) \\ \mathbf{K}(g)(\mathbf{x}) &= \int_{\mathcal{X}} K(\mathbf{x}, \mathbf{y}) g(\mathbf{y}) dP(\mathbf{y}). \end{aligned} \tag{5.2}$$

O operador $\mathbf{K}$ tem uma quantidade enumerável de autofunções $\{\psi_j\}_{j\in\mathbb{N}}$ com respectivos autovalores $\lambda_1 \geq \lambda_2 \geq \ldots \geq 0$ (Minh et al., 2006).

As autofunções $\{\psi_j\}_{j\in\mathbb{N}}$ constituem uma *base espectral*. Elas formam uma base ortonormal de $\mathfrak{L}^2(\mathcal{X}, P)$ (Minh, 2010).

Há duas principais razões pelas quais bases espectrais são candidatas ideais para aproximar funções suaves de $\mathbf{x}$ em altas dimensões:

1. As autofunções $\{\psi_j\}_{j\in\mathbb{N}}$ são *adaptadas à dimensão intrínseca dos dados*. Mais especificamente, quando o domínio $\mathcal{X}$ é uma subvariedade de $\mathbb{R}^d$, a base se comporta como uma base de Fourier adaptada à geometria intrínseca do dados, em que os primeiros termos são mais suaves que os termos de ordem mais alta[1]. Como um exemplo, na Figura 5.2 mostramos as autofunções do operador da Equação 5.2 quando o domínio dos dados é uma subvariedade (uma espiral) de $\mathbb{R}^2$. Compare esta figura com a Figura 4.1; a base se comporta como uma base de Fourier na direção da espiral. Segue que se $r(\mathbf{x})$ é suave em relação a esse domínio, então só precisamos de algumas poucas funções para aproximá-la bem. Esta adaptação leva a taxas de convergência que dependem apenas da dimensão intrínseca dos dados, ao invés da potencialmente maior dimensão ambiente.

2. Ao contrário das bases ortogonais tradicionais, as autofunções são ortogonais *em relação a* $P(\mathbf{x})$, a distribuição marginal dos dados, e não em relação à medida de Lebesgue (Bengio et al., 2004). Isto é,

$$\int_{\mathcal{X}} \psi_i(\mathbf{x})\psi_j(\mathbf{x})dP(\mathbf{x}) = \delta_{i,j}.$$

   Isso leva a estimadores dos coeficientes de expansão que são mais rapidamente calculados. Além disso, não há necessidade de usar produtos tensoriais em altas dimensões, que são extremamente ineficientes para serem calculados.

Como $P(\mathbf{x})$ é desconhecido, é necessário estimar $\{\psi_j\}_j$. Isso pode ser feito primeiramente calculando-se a matriz de Gram

$$\boldsymbol{G} = \begin{bmatrix} K(\mathbf{x}_1,\mathbf{x}_1) & K(\mathbf{x}_1,\mathbf{x}_2) & \cdots & K(\mathbf{x}_1,\mathbf{x}_n) \\ K(\mathbf{x}_2,\mathbf{x}_1) & K(\mathbf{x}_2,\mathbf{x}_2) & \cdots & K(\mathbf{x}_2,\mathbf{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ K(\mathbf{x}_n,\mathbf{x}_1) & K(\mathbf{x}_n,\mathbf{x}_2) & \cdots & K(\mathbf{x}_n,\mathbf{x}_n) \end{bmatrix} \tag{5.3}$$

Seja

$$\widetilde{\psi}_j := \left(\widetilde{\psi}_j(\mathbf{x}_1),\ldots,\widetilde{\psi}_j(\mathbf{x}_n)\right).$$

[1] Isso ocorre porque a distância euclidiana é localmente a mesma que a distância geodésica; veja, por exemplo, Shi et al. 2009 para uma derivação.

o j-ésimo autovetor da matrix 5.3 e $\widehat{\lambda}_j$ seu respectivo autovalor, em que ordenamos os autovetores segundo ordem decrescente de autovalores e os normalizamos de maneira que $\sum_{k=1}^{n} \widetilde{\psi}_j^2(\mathbf{x}_k) = 1$. Um estimador consistente de ψ_j é

$$\widehat{\psi}_j(\mathbf{x}) = \frac{\sqrt{n}}{\widehat{\lambda}_j} \sum_{k=1}^{n} \widetilde{\psi}_j(\mathbf{x}_k) K(\mathbf{x}, \mathbf{x}_k). \tag{5.4}$$

Este estimador é a extensão de Nyström do autovetor $\widetilde{\psi}_j$ para valores fora da amostra $\mathbf{x}$ (Bengio et al., 2004; Drineas e Mahoney, 2005).

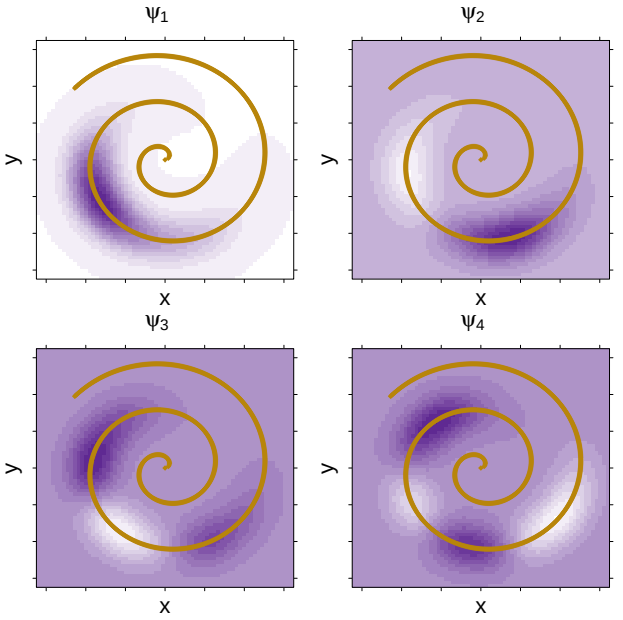


Figura 5.2: Curvas de nível das primeiras quatro autofunções do operador do kernel Gaussiano quando o domínio das covariáveis $\mathbf{x} = (x_1, x_2)$ é em uma espiral. As autofunções formam uma base de Fourier adaptada à geometria dos dados e são apropriadas para aproximar funções suaves de $\mathbf{x}$ nesse domínio. Compare esta figura com a Figura 4.1.

5.4.2 O estimador

Seja K um kernel fixo e $\{\psi_j\}_{j\in\mathbb{N}}$ a base ortonormal relacionada ao operador da Equação (5.2). A expansão da função de regressão r nessa base é dada por

$$r(\mathbf{x}) = \sum_{j\geq 1} \beta_j \psi_j(\mathbf{x}),$$

em que

$$\beta_j = \int_{\mathcal{X}} \psi_j(\mathbf{x}) r(\mathbf{x}) dP(\mathbf{x}) = \int_{\mathcal{X}} \psi_j(\mathbf{x}) \mathbb{E}[Y|\mathbf{x}] dP(\mathbf{x}) = \mathbb{E}[Y\psi_j(\mathbf{X})].$$

Note que a ortogonalidade da base espectral com relação a $P(\mathbf{x})$ é a chave para que β_j seja simplesmente $\mathbb{E}[Y\psi_j(\mathbf{X})]$. Assim, o estimador baseado em séries espectrais (Lee e Izbicki, 2016) é dado por

$$g(\mathbf{x}) = \sum_{j=1}^{J} \widehat{\beta}_j \widehat{\psi}_j(\mathbf{x}), \tag{5.5}$$

em que $\widehat{\psi}_j$'s são estimados como descrito anteriormente e

$$\widehat{\beta}_j = \frac{1}{n} \sum_{k=1}^{n} y_k \widehat{\psi}_j(\mathbf{x}_k).$$

O corte J pode ser escolhido por validação cruzada.

Lee e Izbicki (2016) provam que a taxa de covergência desse estimador depende apenas da dimensão intrínseca do espaço das covariáveis. Assim, esse estimador é um bom candidato quando há muita redundância nas covariáveis.

Exemplo 5.1 (Isomap face data). Trabalhamos aqui com os dados descritos no Exemplo 1.3, que tem por objetivo estimar a direção horizontal para a qual um indivíduo está olhando com base em sua imagem. Cada imagem aqui é representada por uma matriz 64×64, i.e., há $d = 4096$ covariáveis. Temos $n = 698$ observações. Note que, como $n < d$, não é possível implementar o método dos mínimos quadrados para esses dados. A Tabela 5.1 mostra os resultados dos ajustes desses modelos. Nesse problema, os métodos paramétricos utilizados levam a bons resultados, mas as predições obtidas pelos métodos não paramétricos tem melhor qualidade. Em particular, o método baseado em séries espectrais foi o que teve melhor desempenho.

Isso é explicado pelo fato de que, nesse exemplo, esperamos que haja uma grande redundância nas covariáveis.

Tabela 5.1: Riscos estimados e erros-padrão de alguns estimadores para o isomap face data.

Método	Séries Espectrais	KNN	NW	Lasso	Ridge
Risco Estimado	2.70 (0.70)	10.09 (1.50)	11.91 (1.91)	27.69 (6.43)	56.76 (13.48)

□

5.5 Florestas Aleatórias

A taxa de convergência de um estimador da regressão via floresta aleatória depende apenas (sob várias suposições) do número de covariáveis relevantes para predizer Y (Biau, 2012). Intuitivamente, isso ocorre pois, ao escolher qual variável será utilizada na divisão de cada árvore, apenas variáveis que diminuem o EQM são selecionadas. Assim, raramente uma variável irrelevante é utilizada.

5.6 SpAM - Modelos Aditivos Esparsos

Os modelos aditivos apresentados na Seção 4.7 são uma tentativa de diminuir a influência da maldição da dimensionalidade, uma vez que eles assumem que $r(\mathbf{x})$ pode ser decomposta em uma soma linear de funções suaves de cada uma das covariáveis. Modelos aditivos esparsos (SpAM - Sparse Additive Models; Ravikumar et al. 2009) vão um passo além: eles combinam modelos aditivos com o lasso (Seção 3.3) de modo a se obter um estimador não paramétrico que vença a maldição da dimensionalidade em alguns problemas.

Mais especificamente, em um modelo aditivo esparso, busca-se um representação da função de regressão da forma

$$g(\mathbf{x}) = \sum_{j=1}^{d} \beta_j g_j(x_j),$$

em que impõe-se que $\sum_{j=1}^{d} |\beta_j| \leq L$. Isso é, como em modelos aditivos, o SpAM assume que $r(\mathbf{x})$ pode ser decomposta em uma soma linear de funções suaves de cada uma das covariáveis, $g_j(x_j)$, mas, ao mesmo tempo, requer que algumas dessas funções sejam iguais a zero. Modelos aditivos esparsos são ajustados utilizando-se uma combinação do backfitting (Seção 4.7) com um método de solução do lasso. Assim como no lasso, o valor de L pode ser escolhido via validação cruzada. Veja detalhes técnicos em Ravikumar et al. (2009).

No R, o método SpAM pode ser implementado com o pacote `SAM`. O ajuste e predições podem ser feitos via

```
library(SAM)

ajuste <- samQL(X = x_treinamento, y = y_treinamento)
predito <- predict(ajuste, x_teste)
```

Exemplo 5.2 (Wine Quality). Aqui exemplificamos modelos aditivos esparsos para o problema de predizer a qualidade de um vinho (medida através de uma nota dada por um expert) segundo suas característica (pH, quantidade de álcool entre outras; veja mais detalhes em https://archive.ics.uci.edu/ml/datasets/Wine+Quality). As funções suaves g_j utilizadas foram splines com 3 funções de base. A Figura 5.3 mostra os resultados de tal ajuste. Assim como o caso de modelos aditivos, pode-se observar que o SpAM leva a ajustes com fácil interpretação.

5.7 Resumo

Nesse capítulo vimos que é necessário fazer suposições adicionais para que seja possível estimar bem uma função de regressão utilizando métodos não paramétricos quando há muitas covariáveis. As duas suposições investigadas foram "irrelevâncias de algumas covariáveis"e "redundância das covariáveis". Vimos que vários dos métodos estudados no Capítulo 4 automaticamente se adaptam a essas situações, no sentido de que as taxas de convergência desses modelos são muito mais rápidas se essas suposições são válidas.

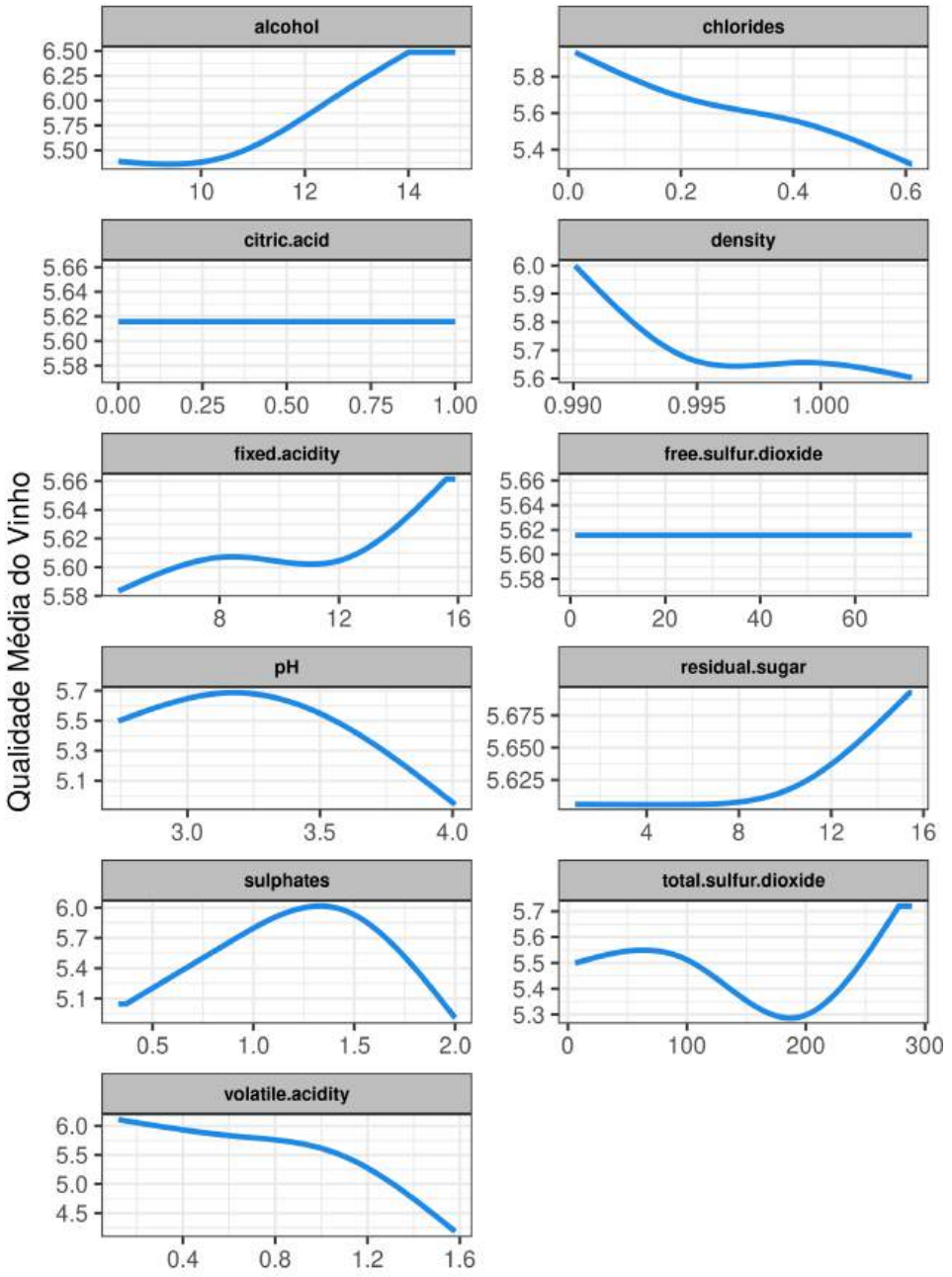


Figura 5.3: Ajuste do modelo esparso aditivo para os dados de vinho (Exemplo 5.2).

Capítulo 6

Outros Aspectos de Regressão

6.1 Interpretabilidade (ExplainableML)

> I didn't like having to explain to them, so I just shut up, smoked a cigarette, and looked at the sea.
>
> Albert Camus

Ainda que o foco da maioria dos métodos de aprendizado supervisionado seja criar bons algoritmos de predição, métodos que permitem maior interpretabilidade costumam ser mais utilizados. Isso porque trazem mais segurança para os usuários desses modelos. Além disso, métodos interpretáveis podem trazer *insights* adicionais sobre o problema abordado. Por exemplo, ao saber quais variáveis são importantes para se obter um bom poder preditivo em um dado problema, pode-se ter uma melhor ideia sobre o desempenho preditivo da função obtida para novos dados com características distintas (ou seja, dados que não são identicamente distribuídos aos dados observados no conjunto de treinamento). Também é possível diagnosticar mais facilmente viéses de seleção que poderiam passar desapercebidos.

Vimos que árvores de predição, modelos aditivos e modelos lineares esparsos são relativamente fáceis de serem interpretados enquanto que *support vector regression* e métodos baseados no truque do kernel tipicamente são mais difíceis. Existem algumas formas de tentar interpretar modelos de predição que são tidos como "caixas pretas". A seguir introduziremos três formas de fazer isso: LIME, PDP e ICE. Outros

métodos de interpretação de modelos de predição podem ser encontrados em Botari et al. (2019), Coscrato et al. (2019), Lundberg e Lee (2017), Molnar (2019) e Ribeiro et al. (2018).

Via de regra, o objetivo de *ExplainableML* é responder à pergunta "Por que esse algoritmo fornece a predição $g(\mathbf{x})$ para esse dado $\mathbf{x}$?", e não "Por que esperamos que esse $\mathbf{x}$ tenha valor predito $g(\mathbf{x})$?". Essa última é uma pergunta inferencial. É justamente confrontando a resposta do LIME com nossa opinião sobre a segunda pergunta que ganharemos ou perderemos confiança no modelo utilizado. Finalmente, notamos que alguns dos métodos descritos aqui também podem ser utilizados para a responder à segunda pergunta caso se assuma que o modelo ajustado é razoável.

6.1.1 LIME - *Local Interpretable Model-agnostic Explanations*

A ideia chave do LIME (Ribeiro et al., 2016) é tentar responder à seguinte pergunta: por que o algoritmo me forneceu a predição $g(\mathbf{x}^*)$ para a nova amostra $\mathbf{x}^*$? A resposta para essa pergunta é dada na forma de uma lista de quais variáveis foram importantes para explicar essa observação. Para tanto, o LIME aproxima a solução $g(\mathbf{x}^*)$ *localmente* (isto é, em uma vizinhança de $\mathbf{x}^*$) via uma regressão linear ajustada por lasso. Isso é feito pois, mesmo se g é não linear, uma aproximação linear pode ser razoável localmente. Para fazer essa aproximação, o procedimento do LIME consiste nos seguintes passos:

- Gerar as covariáveis de novas observações $\mathbf{x}_1^*, \ldots, \mathbf{x}_B^*$ perturbando $\mathbf{x}^*$ (por exemplo, adicionando ruído em cada dimensão de $\mathbf{x}^*$)
- Ajustar o lasso para o conjunto $(\mathbf{x}_1^*, g(\mathbf{x}_1^*)), \ldots, (\mathbf{x}_B^*, g(\mathbf{x}_B^*))$. O ajuste pode ser feito dando pesos para cada observação de acordo com a similaridade: $w_i = K(\mathbf{x}_i^*, \mathbf{x}^*)$. O valor da penalização λ do lasso é escolhido de forma que se tenha m covariáveis escolhidas (m é escolhido pelo usuário e em geral é um número pequeno para facilitar a interpretação).

Retorna-se então quais foram as m variáveis escolhidas e a magnitude dos coeficientes associados.

Exemplo 6.1 (Câncer de Próstata). Neste exemplo utilizaremos o pacote `lime` para interpretar as predições dadas pelo XGBoost para o banco de dados de câncer de próstata (Exemplo 3.2). A Figura 6.1 ilustra as explicações dadas pelo LIME para 4 observações.

```
library(xgboost)

xgb_model <- xgb.train(data = xgb.DMatrix(xtr, label = ytr),
                       nrounds = 50)

explainer <- lime(data.frame(xtr), model = xgb_model)

explanation <- lime::explain(
  data.frame(xval[sample(nrow(xval),4),]),
  explainer,
  n_features = 3)
```

```
plot_features(explanation, ncol = 2)
```

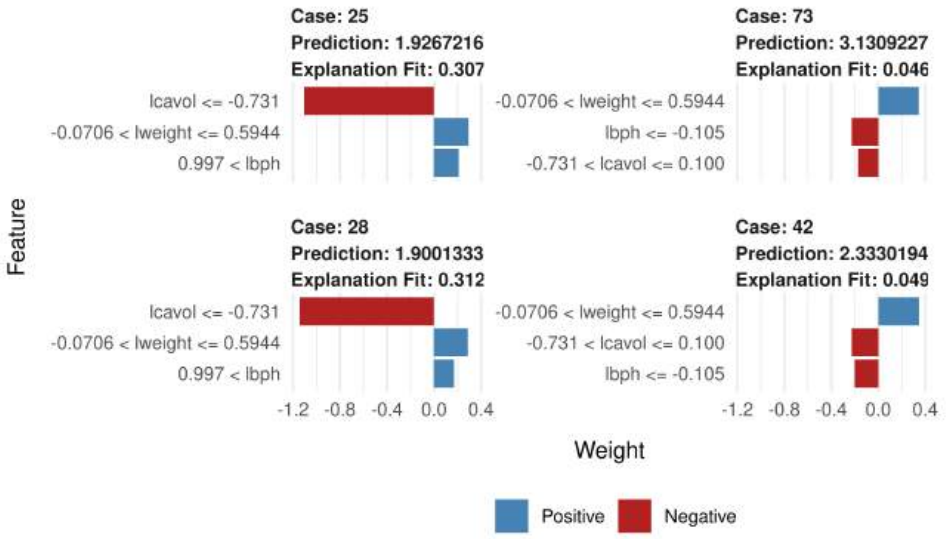


Figura 6.1: Exemplo de interpretação dado pelo LIME para o banco de dados do Câncer de Próstata.

□

6.1.2 PDP e ICE

Os gráficos de *dependência parcial* (do inglês, PDP - partial dependence plot; Friedman 2001) são uma ferramenta valiosa para visualizar como uma função de predição g depende das covariáveis. A curva do PDP para a i-ésima variável é dada por

$$h_i(x) = \frac{1}{n}\sum_{j=1}^{n} g(x_{j,1},\ldots,x_{j,i-1},x,x_{j,i+1},\ldots,x_{j,d}),$$

que é uma estimativa de $\mathbb{E}[g(X_1,\ldots,X_{i-1},x,X_{i+1},\ldots,X_d)]$. Essa curva mede a influência média da i-ésima variável no modelo preditivo g. Se $h_i(x)$ varia muito como função de x, então g é muito alterada por mudanças nessa covariável, o que indica que x_i é uma variável importante para esse modelo preditivo. O formato da curva também elucida como se dá essa relação.

Idealmente, as observações utilizadas para calcular $h_i(x)$ não devem ter sido utilizadas no treinamento do modelo, uma vez que o comportamento de g pode ser muito diferente em novas observações, especialmente se o método faz algum tipo de interpolação.

Caso g seja dado por um modelo adivo (Seção 4.7), as curvas do PDP recuperam as funções de regressão individuais $\widehat{r}_i$. Assim, a visualização dada pelo PDP é exatamente a mesma daquela apresentada na Figura 4.5. Em particular, se g é uma regressão linear, recupera-se uma reta.

Para os casos em que há uma forte interação entre as covariáveis, a influência média de uma covariável pode não ser um bom resumo de seu papel no modelo preditivo. Nessas situações, uma alternativa é a utilização do gráfico de *esperança condicional individual* (do inglês, ICE - individual conditional expectation; Goldstein et al. 2015). Esse gráfico consiste em traçar as curvas

$$h_{i,j}(x) := g(x_{j,1},\ldots,x_{j,i-1},x,x_{j,i+1},\ldots,x_{j,d})$$

para todas as observações j de interesse. Observa-se, assim, a influência da i-ésima variável para cada observação individulmente. Note que a curva do PDP é dada pela média das curvas ICE.

Exemplo 6.2 (**Câncer de Próstata**)**.** Neste exemplo utilizaremos PDP e ICE para interpretar as predições dadas pelo XGBoost para o banco de dados de câncer de próstata (Exemplo 3.2). O ajuste do modelo preditivo foi feito no Exemplo 6.1.

As curva em vermelho no painel da esquerda da Figura 6.2 mostra o PDP para a variável `lcavol`. Ela indica que, à medida que `lcavol` aumenta, as predições do modelo para a variável resposta também aumentam. Por outro lado, o painel da direita mostra que a variável `age` não é muito importante para esse modelo preditivo, pois ela se aproxima de uma reta horizontal.

Em ambos os casos, a interação entre as covariáveis do banco de dados e a variável apresentada parece baixa. O comportamento das curvas em preto é muito parecido com os das curvas em vermelho.

```
library(pdp)

predict.fun <- function(object, newdata){
  newData_x = xgb.DMatrix(data.matrix(newdata), missing = NA)
  results <- predict(object, newData_x)
  return(results)
}

pdp::partial(xgb_model,
             train = as.data.frame(xval),
             pred.var = "lcavol",
             pred.fun = predict.fun,
             ice = TRUE) %>%
  plotPartial(smooth = TRUE, alpha = 0.4,
              pdp.lwd = 4, pdp.col = "#D81B60")

pdp::partial(xgb_model,
             train = as.data.frame(xval),
             pred.var = "age",
             pred.fun = predict.fun,
             ice = TRUE) %>%
  plotPartial(smooth = TRUE,alpha = 0.4,
              pdp.lwd = 4, pdp.col = "#D81B60")
```

□

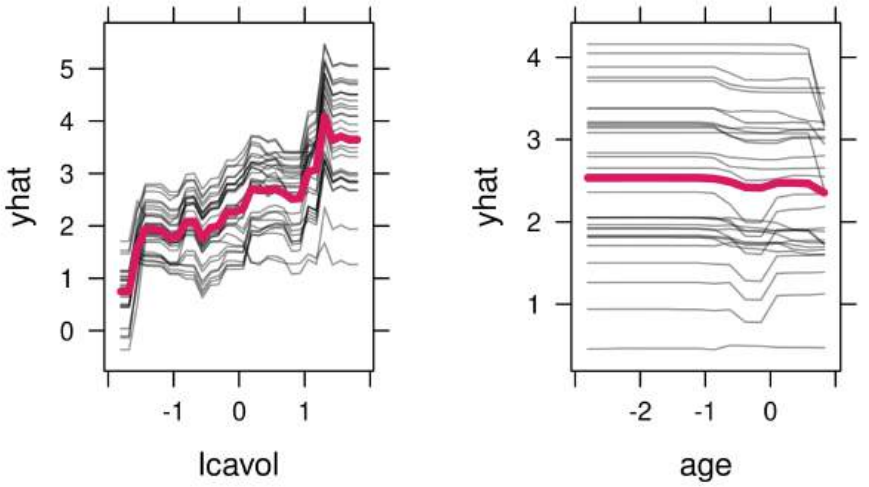


Figura 6.2: Interpretações para duas covariáveis dadas pelo ICE (curvas pretas) e PDP (curvas vermelhas).

6.2 Estimação de Densidades Condicionais

Nem sempre a função de regressão é suficiente para resumir a incerteza sobre Y quando se conhece $\mathbf{x}$. Uma alternativa é estimar não só a esperança condicional $\mathbb{E}[Y|\mathbf{x}]$, mas sim toda a densidade condicional $f(y|\mathbf{x})$. Alguns métodos não paramétricos de estimação dessa quantidade em problemas com baixa dimensionalidade podem ser encontrados em Fan et al. (1996), Hall et al. (2004), Hyndman et al. (1996), Rosenblatt (1969) e Sugiyama et al. (2010). Para métodos que funcionam em alta dimensionalidade, veja Dalmasso et al. (2020) e Izbicki e Lee (2016). Aqui descrevemos o FlexCode (Izbicki e Lee, 2017).

A ideia chave do FlexCode é que, se $\int f^2(y|\mathbf{x})dy < \infty$, então f admite a representação

$$f(y|\mathbf{x}) = \sum_{i \geq 0} \beta_i(\mathbf{x})\phi_i(y),$$

em que $(\phi_i)_{i \geq 0}$ é uma base ortonormal como a base de Fourier. Se f é suave, ela pode

então ser bem aproximada pelos primeiros termos da série:

$$f(y|\mathbf{x}) \approx \sum_{i=0}^{I} \beta_i(\mathbf{x})\phi_i(y).$$

Assim, o problema de estimar a densidade condicional pode ser reformulado como o problema de estimar os coeficientes $\beta_1(\mathbf{x}), \ldots, \beta_I(\mathbf{x})$.

Ao projetar f em ϕ_i, notamos que

$$\beta_i(\mathbf{x}) = \mathbb{E}[\phi_i(Y)|\mathbf{x}].$$

Assim, podemos estimar β_i fazendo uma regressão de $\phi_i(Y)$ em $\mathbf{x}$. O método de regressão ideal depende da estrutura do problema resolvido. Por exemplo, se esperamos uma relação esparsa entre Y e $\mathbf{x}$, podemos utilizar métodos que fazem seleção de variáveis (no sentido de não usar todas as variáveis de forma igual), como o XGBoost ou florestas aleatórias. Por outro lado, se esperamos que $\mathbf{X}$ pertença a uma subvariedade com dimensão intríseca baixa, podemos utilizar o método dos k vizinhos mais próximos.

A Figure 6.3 mostra um exemplo de densidades estimadas pelo FlexCode. Sejam $(\hat{\beta}_i(\mathbf{x}))_{i=0}^{I}$ estimativas desses coeficientes. A densidade estimada é dada por:

$$\hat{f}(y|\mathbf{x}) = \sum_{i=0}^{I} \hat{\beta}_i(\mathbf{x})\phi_i(y).$$

```
# instalação: devtools::install_github("rizbicki/FlexCode")
library(FlexCoDE)

set.seed(123)
# Divisão em treinamento/validação/teste
split_train_test <- initial_split(dados, prob = 0.95)
teste <- testing(split_train_test)
treinamento <- training(split_train_test)

split_train_validation <- initial_split(treinamento,
                                        prob = 0.7)
treinamento <- training(split_train_validation)
```

```
validacao <- testing(split_train_validation)

# Fit FlexCode with RandomForest to estimate the coefficients
fit <- fitFlexCoDE(select(treinamento, -y),
                   select(treinamento, y),
                   select(validacao, -y),
                   select(validacao, y),
                   regressionFunction =
                     regressionFunction.Forest)

p <- plot(fit,
          select(teste, -y),
          select(teste, y) %>% pull())
```

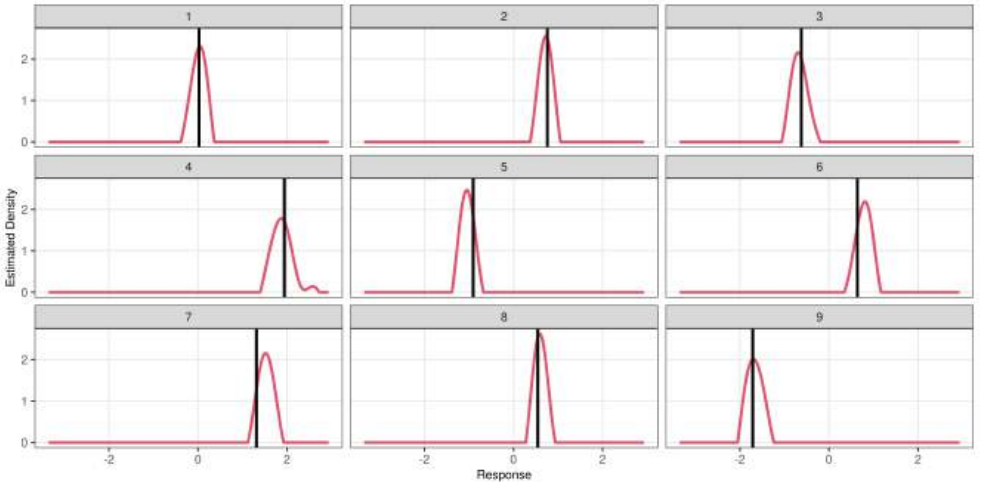


Figura 6.3: Densidade estimadas pelo FlexCode.

6.3 Inferência Conformal

> To be beyond any existing classification has always pleased me.
>
> Boyd Rice

Estimar a distribuição completa de Y condicional em $\mathbf{x}$ não é uma tarefa simples. Uma solução intermediária para fornecer predições para Y que vão além de estimativas pontuais consiste em criar uma região de predição com valores plausíveis de Y para uma amostra com covariáveis $\mathbf{x}$. Uma forma particular de se criar essas bandas, de modo a conseguir garantias de cobertura com poucas suposições, é através da *inferência conformal* (*conformal inference*) (Vovk et al., 2005; Vovk et al., 2009), que utiliza apenas a suposição de que os dados são i.i.d.'s. Mais especificamente, com esse método é possível construir bandas C tais que $\mathbb{P}(Y_{n+1} \in C(\mathbf{X}_{n+1})) = 1 - \alpha$, em que $\alpha \in (0,1)$ é um valor especificado previamente.

Uma método conformal particular é o método *split* (Lei et al., 2018; Papadopoulos et al., 2002), que possui a vantagem de ser rápido frente a outras alternativas. Nesse método, o conjunto de dados é dividido em dois: o conjunto de treinamento $\mathbb{D}' = \{(\mathbf{X}'_1, Y'_1), \ldots, (\mathbf{X}'_n, Y'_n)\}$ e o conjunto de predição $\mathbb{D} = \{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_m)\}$. Para simplificar a notação, vamos assumir que $m = n$. Treinamos então uma função $\hat{h} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, utilizando somente $\mathbb{D}'$. Uma forma usual de definir essa função é $\hat{h}(\mathbf{x}, y) = |y - \hat{r}(\mathbf{x})|$, em que $\hat{r}$ é uma estimativa da regressão obtida a partir de $\mathbb{D}'$ (Lei et al., 2018). Finalmente, $\mathbb{D}$ é utilizado para calcular $U_i := \hat{h}(\mathbf{X}_i, Y_i)$, que são chamados de *resíduos split*. Como os resíduos split são i.i.d. dados $\hat{h}$, o posto que U_{n+1} tem em $\{U_1, \ldots, U_n\}$ tem distribuição uniforme em $\{1, \ldots, n+1\}$ e, definindo $U_{\lfloor \alpha \rfloor}$ como sendo a $\lfloor n\alpha \rfloor$ estatística de ordem entre $U_1, \ldots, U_n$, obtemos que

$$\begin{aligned} \mathbb{P}\left(U_{n+1} \geq U_{\lfloor \alpha \rfloor}\right) &\geq 1 - \alpha \\ \mathbb{P}\left(\hat{h}(\mathbf{X}_{n+1}, Y_{n+1}) \geq U_{\lfloor \alpha \rfloor}\right) &\geq 1 - \alpha \\ \mathbb{P}\left(Y_{n+1} \in \left\{y : \hat{h}(\mathbf{X}_{n+1}, y) \geq U_{\lfloor \alpha \rfloor}\right\}\right) &\geq 1 - \alpha. \end{aligned}$$

Ou seja, $C(\mathbf{X}_{n+1}) = \left\{y : \hat{h}(\mathbf{X}_{n+1}, y) \geq U_{\lfloor \alpha \rfloor}\right\}$ satisfaz $\mathbb{P}(Y_{n+1} \in C(\mathbf{X}_{n+1})) \geq 1 - \alpha$.

Além de $\mathbb{P}(Y_{n+1} \in C(\mathbf{X}_{n+1}))$, outras propriedades são desejáveis. Por exemplo,

idealmente também gostaríamos que $\mathbb{P}(Y_{n+1} \in C(\mathbf{X}_{n+1})|\mathbf{X}_{n+1} = \mathbf{x}) = 1 - \alpha$ para todo $\mathbf{x} \in \mathbb{R}^d$. Contudo, é impossível criar um método que satisfaça isso, a menos que se faça fortes suposições sobre o processo gerador dos dados (Lei e Wasserman, 2014). Assim, alguns métodos de split conformal satisfazem versões mais fracas dessa propriedade. Por exemplo, o `CD-split` e o `Dist-split` (Izbicki et al., 2020a; Izbicki et al., 2020b), além de controlar $\mathbb{P}(Y_{n+1} \in C(\mathbf{X}_{n+1}))$, também controlam $\mathbb{P}(Y_{n+1} \in C(\mathbf{X}_{n+1})|\mathbf{X}_{n+1} = \mathbf{x})$ assintoticamente. Ambos os métodos utilizam densidades condicionais (Seção 6.2) para criar $\hat{h}(\mathbf{x}, y)$: enquanto o `Dist-split` utiliza $\hat{h}(\mathbf{x}, y) = \hat{F}(y|\mathbf{x})$, o `CD-split` utiliza $\hat{h}(\mathbf{x}, y) = \hat{f}(y|\mathbf{x})$. O método `CD-split` requer uma passo adicional de particionamento do espaço de covariáveis para garantir a cobertura condicional assintótica. Esses métodos podem ser calculados com o pacote `predictionBands`[1].

Exemplo 6.3 (Bike sharing). Considere novamente o conjunto de dados do Exemplo 4.3. A Figura 6.4 mostra as bandas de predição para esse número em 100 observações do conjunto de teste. Essas bandas foram criadas a partir das covariáveis disponíveis e o método *dist-split* proposto por Izbicki et al. (2020b).

```
library(rsample)
library(predictionBands)

dados <- read_csv("../dados/hour.csv") %>%
  select(-(instant:dteday),
         -(yr:mnth),
         -(casual:registered)) %>%
  mutate_at(vars(season, holiday, weekday, workingday),
            as.character)

set.seed(400)

split <- initial_split(dados, prop = .9942)

treinamento <- training(split)
teste       <- testing(split)
```

[1] https://github.com/rizbicki/predictionBands

```
X <- select(treinamento, -cnt)
y <- select(treinamento, cnt) %>% pull()

Xteste <- select(teste, -cnt)
yteste <- select(teste, cnt) %>% pull()

fit <- fit_predictionBands(X, y,
              regressionFunction.extra = list(nCores = 4),
              nIMax = 20)

bands <- predict(fit, Xteste, type = "dist")
```

```
p <- plot(bands, yteste)
```

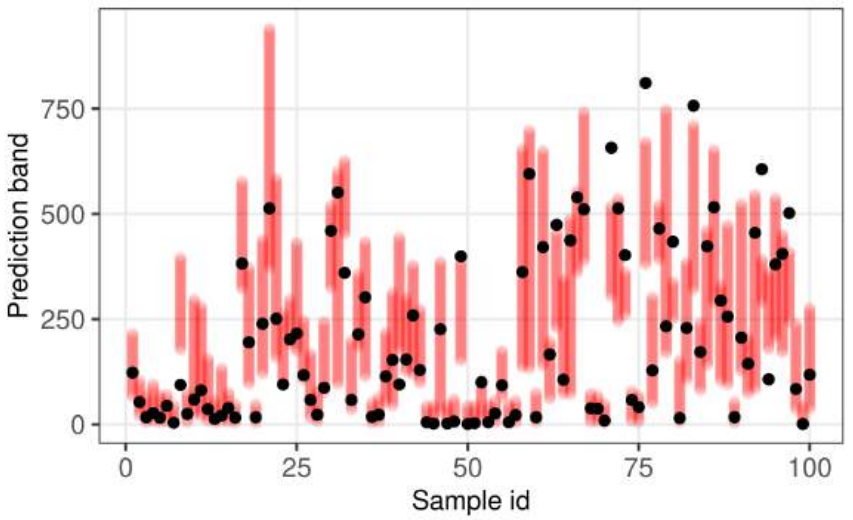


Figura 6.4: Exemplo de aplicação de inferência conformal. Os intervalos representam as bandas de predição para 100 observações do conjunto de teste. Os pontos indicam os valores reais da variável resposta.

□

Parte II

Classificação

Capítulo 7

Introdução

Neste capítulo, estudaremos outro problema central em aprendizado de máquina, o de classificação. Para o leitor que começa por esse capítulo, sugerimos a leitura do Capítulo 1 para compreensão da notação e de conceitos básicos de aprendizado supervisionado.

O problema de classificação é um problema similar ao de predição em regressão. Consideramos uma amostra com observações independentes $(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n) \sim (\mathbf{X}, Y)$ com objetivo de construir uma função $g(\mathbf{x})$ que possa ser usada para fazer bem a predição de novas observações $(\mathbf{X}_{n+1}, Y_{n+1}), \ldots, (\mathbf{X}_{n+m}, Y_{n+m})$, isto é, queremos que

$$g(\mathbf{x}_{n+1}) \approx y_{n+1}, \ldots, g(\mathbf{x}_{n+m}) \approx y_{n+m}.$$

A diferença de um problema de classificação para um problema de regressão é que no primeiro a variável resposta Y não é uma variável quantitativa, mas sim uma variável qualitativa. Por exemplo, prever se um paciente tem uma certa doença com base em variáveis clínicas $\mathbf{x}$ é um problema de classificação. Outro exemplo tradicional é o de classificar automática de dígitos escritos à mão com base em suas imagens (veja Figura 7.1 para alguns exemplos).

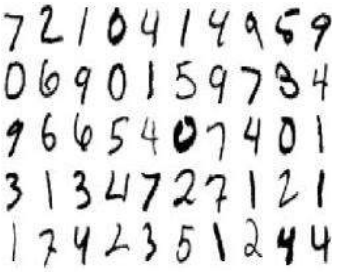


Figura 7.1: Exemplos de dígitos escritos à mão. O objetivo nesse problema é criar uma função $g(\mathbf{x})$ que consiga automaticamente predizer a qual dígito corresponde uma imagem com alta acurácia.

Neste capítulo, investigaremos como a avaliação do desempenho de uma dada função de predição $g(\mathbf{x})$ (isto é, um *classificador*) difere da avaliação estudada no contexto de regressão. No Capítulo 8 estudaremos alguns métodos de criação de classificadores com bom poder preditivo.

7.1 Função de Risco

Vamos assumir que Y possui valores em um conjunto $\mathcal{C}$ (por exemplo, $\mathcal{C}$ pode ser o conjunto $\{\text{spam, não spam}\}$). Primeiramente, notamos que a função de risco $R(g) = \mathbb{E}[(Y - g(\mathbf{X}))^2]$ estudada para o caso de regressão (Y quantitativo) não faz sentido para classificação (o que seria $Y - g(\mathbf{X})$ nesse contexto?). Dessa forma, consideraremos outra função de risco. É comum utilizar

$$R(g) := \mathbb{E}[\mathbb{I}(Y \neq g(\mathbf{X}))] = \mathbb{P}\left(Y \neq g(\mathbf{X})\right), \tag{7.1}$$

ou seja, o risco de g é agora a probabilidade de erro em um nova observação $(\mathbf{X}, Y)$. Em outras palavras, uma função de perda mais adequada para o contexto de classificação é $L(g(\mathbf{x}), Y) = \mathbb{I}(Y \neq g(\mathbf{X}))$, a chamada função de perda 0-1.

No contexto de regressão, vimos que a função que minimiza $\mathbb{E}[(Y - g(\mathbf{X}))^2]$ é dada pela função de regressão $r(\mathbf{x}) = \mathbb{E}[Y|\mathbf{X} = \mathbf{x}]$ (veja o Capítulo 1). Existe um análogo para classificação: a melhor função de classificação g, segundo a função de risco da Equação 7.1, é dada por

$$g(\mathbf{x}) = \arg\max_{d \in \mathcal{C}} \mathbb{P}(Y = d|\mathbf{x}),$$

isto é, deve-se classificar $\mathbf{x}$ como sendo daquela classe com maior probabilidade *a posteriori*. Este classificador é conhecido como classificador de Bayes. Note que, como no caso de regressão, ele é desconhecido pois a função $\mathbb{P}(Y = d|\mathbf{x})$ é desconhecida. O Teorema a seguir formaliza esta caracterização.

Teorema 9. *Suponha que definimos o risco de uma função de predição $g : \mathbb{R}^d \longrightarrow \mathcal{C}$ via perda 0-1: $R(g) = \mathbb{P}(Y \neq g(\mathbf{X}))$, em que $(\mathbf{X}, Y)$ é uma nova observação que não foi usada para estimar g. Então a função g que minimiza $R(g)$ é dada por*

$$g(\mathbf{x}) = \arg\max_{d \in \mathcal{C}} \mathbb{P}(Y = d|\mathbf{x})$$

Demonstração. Para simplificar a demonstração do teorema, vamos assumir que Y assume só dois valores, digamos, c_1 e c_2 (isto é, trata-se de um problema *binário*). Temos que

$$\begin{aligned} R(g) &:= \mathbb{E}[\mathbb{I}(Y \neq g(\mathbf{X}))] = \mathbb{P}(Y \neq g(\mathbf{X})) = \int_{\mathbb{R}^d} \mathbb{P}(Y \neq g(\mathbf{X})|\mathbf{x}) f(\mathbf{x}) d\mathbf{x} \\ &= \int_{\mathbb{R}^d} [\mathbb{I}(g(\mathbf{x}) = c_2)\mathbb{P}(Y = c_1|\mathbf{x}) + \mathbb{I}(g(\mathbf{x}) = c_1)\mathbb{P}(Y = c_2|\mathbf{x})] f(\mathbf{x}) d\mathbf{x}. \end{aligned}$$

Assim, para cada $\mathbf{x}$ fixo, o termo interno da integral é minimizado quando se escolhe $g(\mathbf{x}) = c_1$ quando

$$\mathbb{P}(Y = c_1|\mathbf{x}) \geq \mathbb{P}(Y = c_2|\mathbf{x})$$

e $g(\mathbf{x}) = c_2$ caso contrário. □

Para o caso binário, o classificador de Bayes pode ser reescrito como

$$g(\mathbf{x}) = c_1 \iff \mathbb{P}(Y = c_1|\mathbf{x}) \geq \frac{1}{2}.$$

Na Seção 9.1 discutiremos alguns casos em que utilizar um valor diferente de $1/2$ é vantajoso.

Observação 7.1. No caso de classificação binária é comum denotar os elementos de $\mathcal{C}$ por 0 e 1. Essa escolha é válida, mas arbitrária (isto é, não se deve entender que há uma ordenação entre esses elementos).

□

7.2 Estimação do Risco e Seleção de Modelos

Da mesma maneira que em regressão (Seção 1.5.1), pode-se estimar o risco de um método de classificação utilizando-se data splitting ou validação cruzada. Consideremos a primeira dessas abordagens (lembre-se que as observações devem estar em uma ordem aleatória):

$$\overbrace{(\mathbf{X}_1, Y_1), (\mathbf{X}_2, Y_2), \ldots, (\mathbf{X}_s, Y_s)}^{\text{Treinamento (por exemplo, 70\%)}} \quad , \quad \overbrace{(\mathbf{X}_{s+1}, Y_{s+1}), \ldots, (\mathbf{X}_n, Y_n)}^{\text{Validação (por exemplo, 30\%)}}.$$

Como em regressão, utilizamos o conjunto de treinamento para estimar g e o conjunto de validação *apenas para estimar* $R(g)$ via

$$R(g) \approx \frac{1}{n-s} \sum_{i=s+1}^{n} \mathbb{I}(Y_i \neq g(\mathbf{X}_i)) := \widehat{R}(g), \tag{7.2}$$

isto é, avaliamos a proporção de erros no conjunto de validação.

Assim, uma forma de selecionar um modelo g dentro de uma classe de modelos $\mathbb{G}$ consiste em utilizar validação cruzada para estimar $R(g)$ para cada $g \in \mathbb{G}$ e, então, escolher g com o menor risco estimado. O teorema a seguir mostra que esse procedimento, de fato, retorna com probabilidade alta o melhor modelo em $\mathbb{G}$.

Teorema 10. *Seja $\mathbb{G} = \{g_1, \ldots, g_N\}$ uma classe de classificadores estimados com base em um conjunto de treinamento e seja $\widehat{R}(g)$ o erro estimado de g com base no conjunto de validação (Equação 7.2). Seja g^* o modelo que minimiza o risco real $R(g)$ dentre $g \in \mathbb{G}$ e seja $\widehat{g}$ o modelo que minimiza o risco estimado $\widehat{R}(g)$ dentre $g \in \mathbb{G}$. Então, com probabilidade de no máximo ϵ ($\epsilon > 0$),*

$$|R(\widehat{g}) - R(g^*)| > 2\sqrt{\frac{1}{2(n-s)} \log \frac{2N}{\epsilon}}.$$

Demonstração. Pela desigualdade de Hoeffding, para todo $\delta > 0$ e todo $g \in \mathbb{G}$, tem-se que

$$\mathbb{P}(|\widehat{R}(g) - R(g)| > \delta) \leq 2e^{-2(n-s)\delta^2}.$$

Segue da desigualdade da união que

$$\mathbb{P}\left(\max_{g\in\mathbb{G}}|\widehat{R}(g)-R(g)|>\delta\right)=\mathbb{P}\left(\bigcup_{g\in\mathbb{G}}|\widehat{R}(g)-R(g)|>\delta\right)$$
$$\leq\sum_{g\in\mathbb{G}}\mathbb{P}\left(|\widehat{R}(g)-R(g)|>\delta\right)\leq N2e^{-2(n-s)\delta^2}.$$

Agora, tome $\delta=\sqrt{\frac{1}{2(n-s)}\log\frac{2N}{\epsilon}}$, de modo que $N2e^{-2(n-s)\delta^2}=\epsilon$. Conclua que, com probabilidade ao menos $1-\epsilon$, $|\widehat{R}(g)-R(g)|\leq\delta$ para todo $g\in\mathbb{G}$ e, assim,

$$R(\widehat{g})\leq\widehat{R}(\widehat{g})+\delta\leq\widehat{R}(g^*)+\delta\leq R(g^*)+2\delta.$$

□

O Teorema 10 mostra que, quanto maior é o número de classificadores em $\mathbb{G}$, menor é a probabilidade de recuperarmos o melhor modelo. Esse é um dos fatores que fazem com que idealmente apenas um conjunto pequeno de modelos seja comparado no conjunto de teste e o ajuste/*tuning* de cada modelo é feito utilizando validação cruzada dentro do treinamento ou particionando o treinamento em dois conjuntos: treinamento e validação. Assim, deixamos o teste para comparar apenas o melhor modelo de cada classe (por exemplo, a melhor regressão logística penalizada encontrada com o melhor KNN).

Uma extensão desse resultado para o caso em que $\mathbb{G}$ possui um número infinito de elementos será estudada na Seção 9.5, quando introduzirmos a teoria VC.

7.3 Balanço entre Viés e Variância

Assim como em regressão (Seção 1.5.2), em classificação também se enfrenta o paradigma do balanço entre viés e variância. Contudo, os detalhes são um pouco diferentes. Suponha que $\mathbb{G}$ seja uma classe de classificadores (por exemplo, todos os classificadores baseados em uma regressão logística; veja Seção 8.1.2), $R_{or}:=\inf_{g\in\mathbb{G}}R(g)$ o melhor risco (isto é, o risco do oráculo) que pode ser obtido usando-se classificadores de $\mathbb{G}$ e R_* o risco do classificador de Bayes (Teorema 9). Então, para todo $g\in\mathbb{G}$,

$$R(g)-R_*=T_1+T_2,$$

em que $T_1 = R(g) - R_{or}$ é o análogo da variância (em geral é alto se $\mathbb{G}$ possui muitos elementos – por exemplo, se há muitos parâmetros na regressão logística) e $T_2 = R_{or} - R_*$ é o análogo do viés ao quadrado (em geral é baixo se $\mathbb{G}$ possui muitos elementos). Assim, $\mathbb{G}$ não pode ser nem muito grande, nem muito pequena.

7.4 Outras medidas de desempenho

> You can't notice what isn't mentioned unless you're an expert.
>
> Daniel Dennett

Nem sempre a função de risco $R(g) := \mathbb{E}[\mathbb{I}(Y \neq g(\mathbf{X}))] = \mathbb{P}(Y \neq g(\mathbf{X}))$ traz toda informação sobre o quão razoável g é. Por exemplo, suponha que Y indica se uma pessoa tem uma certa doença rara e que, portanto, em uma amostra i.i.d., há poucos pacientes com $Y = 1$. O classificador trivial $g(\mathbf{x}) \equiv 0$ (classificar todos os pacientes como sendo saudáveis) terá risco baixo, pois $\mathbb{P}(Y \neq 0)$ é pequena, mas seu desempenho deixa a desejar: frequentemente não queremos classificar de forma errada um paciente que é doente. Em termos numéricos, considere um exemplo de uma amostra com 1.000 pacientes e apenas 10 doente. Ao classificar todos como saudáveis, o risco apresentaria um valor muito reduzido.

Na prática, para evitar esse tipo de situação, é comum avaliar o desempenho de um classificador com base em matrizes de confusão como a apresentada a seguir:

Tabela 7.1: Matriz de confusão

Valor Predito	**Valor verdadeiro**	
	$Y = 0$	$Y = 1$
$Y = 0$	VN (verdadeiro negativo)	FN (falso negativo)
$Y = 1$	FP (falso positivo)	VP (verdadeiro positivo)

Com base nessa tabela, pode-se definir várias medidas, como por exemplo:

- **Sensibilidade/Recall:** S = VP/(VP + FN) (dos pacientes doentes, quantos foram corretamente identificados?)

- **Especificidade:** E = VN/(VN + FP) (dos pacientes não doentes, quantos foram corretamente identificados?)
- **Valor preditivo positivo/Precision:** VPP = VP/(VP + FP) (dos pacientes classificados como doentes, quantos foram corretamente identificados?)
- **Valor preditivo negativo:** VPN = VN/(VN + FN) (dos pacientes classificados como não doentes, quantos foram corretamente identificados?)
- **Estatística F1:** F1 $= \frac{2}{1/S+1/VPP}$ (a média harmônica entre S e VPP)

O classificador trivial $g(\mathbf{x}) \equiv 0$ tem sensibilidade zero e especificidade um. Assim, apesar da especificidade ser alta, a sensibilidade é muito baixa. Isso indica que o classificador pode na realidade ser ruim: ele classifica incorretamente todos os pacientes doentes. Portanto, é recomendável olhar para outras medidas além do risco estimado. Isso se torna particularmente importante na presença de dados desbalanceados (isto é, caso a frequência de uma classe seja muito diferente das demais). Voltaremos para esse tópico na Seção 9.1.

As estatísticas calculadas com base na Tabela 7.1 são estimativas populacionais de certas quantidades. Por exemplo, a sensibilidade é uma estimativa de $\mathbb{P}(g(\mathbf{X}) = 1|Y = 1)$, enquanto que a especificidade é uma estimativa de $\mathbb{P}(g(\mathbf{X}) = 0|Y = 0)$. Assim, é importante calcular os valores de VP, FN, VN e FP utilizando uma amostra de teste ou validação para evitar o sobre-ajuste.

Capítulo 8

Métodos de classificação

Neste capítulo apresentamos diversos métodos que visam fornecer classificadores com bom poder preditivo.

8.1 Classificadores Plug-in

O Teorema 9 sugere uma abordagem simples para resolver um problema de predição:

1. Estimamos $\mathbb{P}(Y = c|\mathbf{x})$ para cada categoria $c \in \mathcal{C}$.

2. Tomamos então
$$g(\mathbf{x}) = \arg\max_{c \in \mathcal{C}} \widehat{\mathbb{P}}(Y = c|\mathbf{x})$$

Em particular, no caso binário, tomamos

$$g(\mathbf{x}) = 1 \iff \widehat{\mathbb{P}}(Y = 1|\mathbf{x}) > \frac{1}{2}.$$

Essa abordagem é conhecida como classificador *plug-in*, pois pluga-se o estimador da probabilidade condicional na fórmula do g ótimo. Assim, sob esta abordagem, criar um classificador se resume a estimar $\mathbb{P}(Y = c|\mathbf{x})$. Nas seguintes seções veremos algumas formas de estimar essas probabilidades. Na Seção 9.1, veremos que, muitas vezes, usar outros cortes para classificação além de $1/2$ pode levar a resultados melhores.

8.1.1 Métodos de regressão

Living is messy.

Woody Allen

Note que, para todo $c \in \mathcal{C}$,

$$\mathbb{P}(Y = c|\mathbf{x}) = \mathbb{E}[\mathbb{I}(Y = c)|\mathbf{x}].$$

Assim, pode-se utilizar qualquer modelo de regressão (como árvores, florestas ou qualquer outro método descrito na Parte I desse livro) para estimar $\mathbb{P}(Y = c|\mathbf{x})$; basta estimar a função de regressão $\mathbb{E}[Z|\mathbf{x}]$, em que $Z = \mathbb{I}(Y = c)$. Por exemplo, pode-se estimar essa probabilidade via regressão linear, isto é, assumindo-se que

$$\mathbb{P}(Y = c|\mathbf{x}) = \mathbb{E}[Z|\mathbf{x}] = \beta_0^{(c)} + \beta_1^{(c)} x_1 + \ldots + \beta_p^{(c)} x_p.$$

É possível, por exemplo, utilizar o método de mínimos quadrados, *lasso* ou outras abordagens já discutidas para estimar esses coeficientes. Ainda que as estimativas de $\mathbb{P}(Y = c|\mathbf{x})$ possam ser menores que zero ou maiores que um, essas podem ser utilizadas para definir o classificador

$$g(\mathbf{x}) = \arg\max_{c \in \mathcal{C}} \widehat{\mathbb{P}}(Y = c|\mathbf{x}).$$

Embora classificadores criados desta maneira frequentemente apresentem bons resultados, há pessoas que se sentem desconfortáveis em obter estimativas para uma probabilidade maiores que um ou menores que zero[1]. Diversos métodos que evitam isso serão apresentados nas próximas seções.

8.1.2 Regressão logística

Vamos assumir, a princípio, que Y é binária, isto é, $|\mathcal{C}| = 2$. Denotando $\mathcal{C} = \{0, 1\}$, a regressão logística apresenta a seguinte forma paramétrica:

$$\mathbb{P}(Y = 1|\mathbf{x}) = \frac{e^{\beta_0 + \sum_{i=1}^d \beta_i x_i}}{1 + e^{\beta_0 + \sum_{i=1}^d \beta_i x_i}}.$$

[1] Note, contudo, que as probabilidades estimadas dessa forma via KNN ou métodos baseados em árvores estão, por construção, necessariamente entre zero e um.

Assim como no caso de regressão, não estamos assumindo que esta relação é, de fato, válida. No entanto, esperamos que ela nos leve a um bom classificador.

Para estimar os coeficientes de uma regressão logística, podemos usar o método de máxima verossimilhança. Nesse caso, dada uma amostra i.i.d. $(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)$, a função de verossimilhança condicional nas covariáveis, é

$$L(y; (\mathbf{x}, \boldsymbol{\beta})) = \prod_{k=1}^{n} (\mathbb{P}(Y_k = 1 | \mathbf{x}_k, \boldsymbol{\beta}))^{y_k} (1 - \mathbb{P}(Y_k = 1 | \mathbf{x}_k, \boldsymbol{\beta}))^{1-y_k}$$
$$\prod_{k=1}^{n} \left(\frac{e^{\beta_0 + \sum_{i=1}^{d} \beta_i x_{k,i}}}{1 + e^{\beta_0 + \sum_{i=1}^{d} \beta_i x_{k,i}}} \right)^{y_k} \left(\frac{1}{1 + e^{\beta_0 + \sum_{i=1}^{d} \beta_i x_{k,i}}} \right)^{1-y_k}.$$

Para obter as estimativas dos coeficientes β, maximizamos $L(y; (\mathbf{x}, \beta))$. Ao contrário do estimador de mínimos quadrados de uma regressão linear (Eq. 2.2), é necessário usar algoritmos numéricos para maximizar a verossimilhança induzida pela regressão logística e, assim, chegar nas estimativas para os coeficientes $\boldsymbol{\beta}$. No R, pode-se utilizar a função `glm`:

```
glm_fit  <- glm(formula, data = dados, family = binomial)
```

Assim como no caso da regressão linear, também é possível utilizar penalização para estimar os coeficientes da regressão logística. Dessa forma, espera-se obter melhor poder preditivo ao reduzir a variância do estimador. Para mais detalhes veja Hastie et al. (2001). O pacote `glmnet` do R permite que tais modelos sejam facilmente ajustados:

```
modelo <- glmnet(X_treino, y_treino,
                 alpha = 1, family = "binomial")

# ou, para considerar validação cruzada, utilize

modelo <- cv.glmnet(X_treino, y_treino,
                 alpha = 1, family = "binomial")
```

Quando $|\mathcal{C}| > 2$ (casos com mais de duas categorias), podemos estimar, para cada $c \in \mathcal{C}$, $\mathbb{P}(Y = c|\mathbf{x})$ utilizando uma regressão logística diferente. Para tanto, basta estimar $\mathbb{P}(Z = 1|\mathbf{x})$, em que $Z = \mathbb{I}(Y = c)$. Assim, ajustamos $|\mathcal{C}|$ regressões logísticas

e então utilizamos o classificador

$$g(\mathbf{x}) = \arg\max_{c \in \mathcal{C}} \widehat{\mathbb{P}}(Y = c|\mathbf{x}).$$

Existem outros métodos que garantem que a soma das probabilidades estimadas seja um, veja por exemplo Friedman et al. (2010).

8.1.3 Bayes Ingênuo

Uma outra abordagem para estimar $\mathbb{P}(Y = c|\mathbf{x})$ consiste em usar o Teorema de Bayes. Assumindo que $\mathbf{X}$ seja um vetor de covariáveis contínuas, temos que

$$\mathbb{P}(Y = c|\mathbf{x}) = \frac{f(\mathbf{x}|Y = c)\mathbb{P}(Y = c)}{\sum_{s \in \mathcal{C}} f(\mathbf{x}|Y = s)\mathbb{P}(Y = s)}$$

Assim, pode-se obter uma estimativa de $\mathbb{P}(Y = c|\mathbf{x})$ estimando-se as probabilidades marginais $\mathbb{P}(Y = s)$ e as densidades condicionais $f(\mathbf{x}|Y = s)$ para cada $s \in \mathcal{C}$.

O termo $\mathbb{P}(Y = s)$ pode ser facilmente estimado utilizando-se as proporções amostrais de cada classe. Contudo, para estimar $f(\mathbf{x}|Y = s)$, é necessário assumir algum modelo para as covariáveis. O método Bayes ingênuo (*naive Bayes* em inglês) assume que, para todo $s \in \mathcal{C}$, $f(\mathbf{x}|Y = s)$ pode ser fatorada como

$$f(\mathbf{x}|Y = s) = f((x_1, \ldots, x_d)|Y = s) = \prod_{j=1}^{d} f(x_j|Y = s),$$

isto é, assume que as componentes de $\mathbf{x}$ são independentes condicionalmente à classe Y. Apesar dessa suposição não ser razoável em muitos problemas, ela é muito conveniente e pode levar a um bom classificador.

Podemos então estimar cada $f(x_j|Y = s)$ assumindo, por exemplo, que

$$X_j|Y = s \sim N(\mu_{j,s}, \sigma^2_{j,s}),\ j = 1, \ldots, d.$$

Em outras palavras, assumimos que cada componente do vetor $\mathbf{X}$ tem distribuição normal com parâmetros que dependem da classe e da componente em questão. Os parâmetros desse modelo podem ser estimados, por exemplo, via estimação de má-

xima verossimilhança:

$$\widehat{\mu_{j,s}} = \frac{1}{|\mathcal{C}_s|}\sum_{k\in\mathcal{C}_s} X_{j,k} \quad \widehat{\sigma^2_{j,s}} = \frac{1}{|\mathcal{C}_s|}\sum_{k\in\mathcal{C}_s}(X_{j,k}-\widehat{\mu}_{j,s})^2$$

em que $\mathcal{C}_s = \{j : Y_j = s\}$ é o conjunto de todas observações de treinamento da classe s.

Nesse caso, o estimador para a densidade condicional $f(\mathbf{x}|Y=c)$ é então dado por

$$\widehat{f}(\mathbf{x}|Y=c) = \prod_{k=1}^{d}\widehat{f}(x_k|Y=c) = \prod_{k=1}^{d}\frac{1}{\sqrt{2\pi\widehat{\sigma^2}_{k,c}}}e^{-\left(\frac{(x_k-\widehat{\mu}_{k,c})^2}{2\widehat{\sigma^2_{k,c}}}\right)}$$

Evidentemente, além da distribuição normal, outras distribuições podem ser utilizadas. Além disso, pode se utilizar diferentes métodos de estimação além do método da máxima verossimilhança. Pode-se, inclusive, utilizar métodos não-paramétricos para estimar cada uma das densidades condicionais.

Se $\mathbf{X}$ tem componentes discretas, o Teorema de Bayes afirma que

$$\mathbb{P}(Y=c|\mathbf{x}) = \frac{\mathbb{P}(\mathbf{X}=\mathbf{x}|Y=c)\mathbb{P}(Y=c)}{\sum_{s\in\mathcal{C}}\mathbb{P}(\mathbf{X}=\mathbf{x}|Y=s)\mathbb{P}(Y=s)}.$$

Nesse caso, poderíamos assumir que

$$X_j|Y=c \sim \text{Multinomial}(1,\theta_{j,c}),$$

em que $\theta_{j,c}\in\mathbb{R}^q$ é um vetor com q dimensões, o número de categorias que X_j assume. Novamente, os parâmetros dessa distribuição podem ser estimados via o método da máxima verossimilhança.

Observação 8.1. Na implementação do Bayes ingênuo para dados contínuos, calcular produtos como $\prod_{k=1}^{d}\widehat{f}(x_k|Y=c)$ é numericamente desafiador, pois cada termo é, em geral, muito próximo de zero. Dessa forma, o produto é aproximado por 0 pelo computador. Uma forma de solucionar esse problema é trabalhar com logaritmos. Para isso, note que o classificador plugin com probabilidades estimadas pelo método

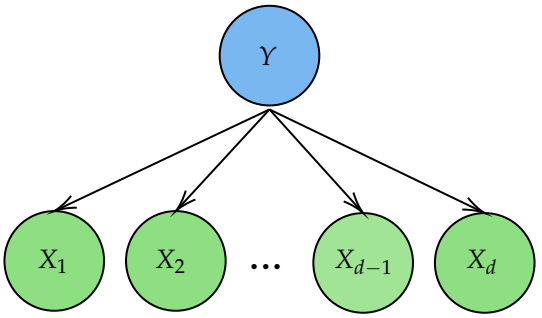


Figura 8.1: Rede Bayesiana que corresponde ao Bayes ingênuo.

Bayes ingênuo pode ser escrito como

$$\begin{aligned} g(\mathbf{x}) &= \arg\max_{c\in\mathcal{C}} \widehat{\mathbb{P}}(Y=c|\mathbf{x}) = \arg\max_{c\in\mathcal{C}} \widehat{f}(\mathbf{x}|Y=c)\widehat{\mathbb{P}}(Y=c) \\ &= \arg\max_{c\in\mathcal{C}} \left(\prod_{k=1}^{d} \widehat{f}(x_k|Y=c)\right) \widehat{\mathbb{P}}(Y=c) \\ &= \arg\max_{c\in\mathcal{C}} \sum_{k=1}^{d} \log\left(\widehat{f}(x_k|Y=c)\right) + \log\widehat{\mathbb{P}}(Y=c), \end{aligned}$$

uma vez que o logaritmo é uma função monotônica crescente. Assim, não é necessário calcular $\widehat{f}(\mathbf{x}|Y=c)$ diretamente para avaliar a decisão tomada.

□

Uma versão muito mais geral do método Bayes ingênuo é o de redes Bayesianas (Pearl, 2014). De fato, Bayes ingênuo é a rede particular mostrada na Figura 8.1. Uma grande vantagem da abordagem de redes Bayesianas é lidar facilmente com covariáveis faltantes, uma vez que essas redes modelam a distribuição conjunta $f(y,\mathbf{x})$.

Para dados discretos, pode-se ajustar o método Bayes ingênuo utilizando-se a função `naiveBayes` no pacote `e1071` do R.

Exemplo 8.1 (Detecção de SPAMs). Neste exemplo, consideramos o conjunto de dados Spambase, disponível do repositório do UCI[2]. Esse conjunto contém informações relativas a $n = 4.601$ emails. Nesses emails foram medidas a frequência relativa (isto é, qual a proporção de vezes que essa palavra aparece em cada email) de

[2] https://archive.ics.uci.edu/ml/datasets/Spambase.

57 palavras, como `internet`, `free`, `credit`, `money`, `data`, `technology`, `direct` entre outras. Também foi verificado se cada email era ou não era SPAM (variável resposta). A Tabela 8.1 apresenta o erro preditivo estimado em um conjunto de teste de tamanho 600 para três classificadores.

Tabela 8.1: Riscos estimados e erros-padrão de alguns estimadores para o Exemplo 8.1.

Método	Regressão Linear	Regressão Logística	Naive Bayes
Risco Estimado	0.111	0.086	0.188
Erro Padrão	0.020	0.020	0.020

□

8.1.4 Análise Discriminante

Enquanto o método de Bayes ingênuo assume que a distribuição condicional das covariáveis pode ser fatorada como $f(\mathbf{x}|Y=c) = f(x_1,\ldots,x_d|Y=c) = \prod_{j=1}^{d} f(x_j|Y=c)$, outras suposições podem ser feitas de modo a estimar essa quantidade. Na análise discriminante, supõe-se que o vetor $\mathbf{X}$, condicional em $Y=c$, possui distribuição normal multivariada. Existem duas formas de análise discriminante mais comuns. Essas formas serão estudadas nas próximas seções.

8.1.4.1 Análise Discriminante Linear

Na análise discriminante linear assume-se que cada distribuição condicional $\mathbf{X}|Y=c$ segue uma distribuição normal multivariada. Embora essas distribuições possam ter médias diferentes, a matriz de covariância é a mesma para todas as distribuições. Assim, assume-se que

$$\mathbf{X} = (X_1,\ldots,X_d)|Y=c \sim \text{Normal}(\mu_c,\Sigma),$$

isto é,

$$f(\mathbf{x}|Y=c) = \frac{1}{\sqrt{(2\pi)^d|\Sigma|}} e^{-(\mathbf{x}-\mu_c)^\intercal \Sigma^{-1}(\mathbf{x}-\mu_c)}.$$

Pode-se estimar os parâmetros dessa distribuição utilizando-se o método da máxima verossimilhança. Com isso, obtêm-se

$$\widehat{\mu}_c = \frac{1}{|\mathcal{C}_c|}\sum_{k\in\mathcal{C}_c}\mathbf{X}_k; \quad \widehat{\Sigma} = \frac{1}{n}\sum_{c\in\mathcal{C}}\sum_{k\in\mathcal{C}_c}(\mathbf{x}_k-\widehat{\mu}_c)(\mathbf{x}_k-\widehat{\mu}_c)^\intercal$$

em que $\mathcal{C}_c = \{j = 1,\ldots,n : Y_j = c\}$.

Como no método Bayes ingênuo, utiliza-se as estimativas de $\widehat{f}(\mathbf{x}|Y=c)$ para construir o classificador plugin

$$g(\mathbf{x}) = \arg\max_{c\in\mathcal{C}}\widehat{\mathbb{P}}(Y=c|\mathbf{x}) = \arg\max_{c\in\mathcal{C}}\widehat{f}(\mathbf{x}|Y=c)\widehat{\mathbb{P}}(Y=c).$$

Para o caso binário, tem-se que a regra de classificação é $g(\mathbf{x}) = 1$ se, e só se,

$$\begin{aligned}
&\frac{\widehat{\mathbb{P}}(Y=1|\mathbf{x})}{\widehat{\mathbb{P}}(Y=0|\mathbf{x})} \geq K \iff \frac{\widehat{f}(\mathbf{x}|Y=1)\widehat{\mathbb{P}}(Y=1)}{\widehat{f}(\mathbf{x}|Y=0)\widehat{\mathbb{P}}(Y=0)} \geq K \iff \\
&\log\widehat{f}(\mathbf{x}|Y=1) - \log\widehat{f}(\mathbf{x}|Y=0) \geq \log K + \log\widehat{\mathbb{P}}(Y=0) - \log\widehat{\mathbb{P}}(Y=1) \\
&\iff -(\mathbf{x}-\widehat{\mu}_1)^\intercal\widehat{\Sigma}^{-1}(\mathbf{x}-\widehat{\mu}_1) + (\mathbf{x}-\widehat{\mu}_0)^\intercal\widehat{\Sigma}^{-1}(\mathbf{x}-\widehat{\mu}_0) \geq K' \\
&\iff +2\mathbf{x}^\intercal\widehat{\Sigma}^{-1}\widehat{\mu}_1 - \widehat{\mu}_1^\intercal\widehat{\Sigma}^{-1}\widehat{\mu}_1 - 2\mathbf{x}^\intercal\widehat{\Sigma}^{-1}\widehat{\mu}_0 + \widehat{\mu}_0^\intercal\widehat{\Sigma}^{-1}\widehat{\mu}_0 \geq K' \\
&\iff a\mathbf{x}^\intercal \geq K'',
\end{aligned}$$

em que a, K, K' e K'' são constantes[3]. Assim, a região do espaço amostral em que a regra de decisão consiste em $g(\mathbf{x}) = 1$ é um *hiperplano* de $\mathbb{R}^d$. É por esse motivo que o método recebe o nome de *análise discriminante linear*. Veja uma ilustração dessa região no Exemplo 8.2.

A análise discriminante linear pode ser ajustada no R via o pacote `MASS`:

```
library(MASS)

lda_fit <- lda(x = X_treino, grouping = y_treino)
lda_pred <- predict(lda_fit, newdata = X_novo) # previsões
lda_pred$posterior # contém as estimativas de P(Y=c|x)
```

[3] Até aqui estamos tomando $K = 1$, mas na Seção 9.1 veremos que esse corte pode ser diferente.

Assim como nos outros métodos preditivos vistos, não acreditamos necessariamente na suposição de normalidade, mas ela é apenas utilizada para obter um classificador com poder preditivo potencialmente bom.

8.1.4.2 Análise Discriminante Quadrática

A suposição feita pela análise discriminante quadrática também é de normalidade multivariada. Contudo, cada distribuição condicional pode ter sua própria matriz de covariância. Assim, assume-se que

$$\mathbf{X} = (X_1, \ldots, X_d)|Y = c \sim \text{Normal}(\mu_c, \Sigma_c),$$

isto é,

$$f(\mathbf{x}|Y = c) = \frac{1}{\sqrt{(2\pi)^d |\Sigma_c|}} e^{-(\mathbf{x}-\mu_c)^\intercal \Sigma_c^{-1} (\mathbf{x}-\mu_c)}.$$

Novamente, pode-se estimar esses parâmetros pelo método da máxima verossimilhança:

$$\widehat{\mu_c} = \frac{1}{|\mathcal{C}_c|} \sum_{k \in \mathcal{C}_c} \mathbf{X}_k; \quad \widehat{\Sigma}_c = \frac{1}{|\mathcal{C}_c|} \sum_{k \in \mathcal{C}_c} (\mathbf{x}_k - \widehat{\mu_c})(\mathbf{x}_k - \widehat{\mu_c})^\intercal$$

em que $\mathcal{C}_c = \{j = 1, \ldots, n : Y_j = c\}$.

Como na análise discriminante linear, as estimativas de $\widehat{f}(\mathbf{x}|Y = c)$ são utilizadas para construir o classificador plugin

$$g(\mathbf{x}) = \arg\max_{c \in \mathcal{C}} \widehat{\mathbb{P}}(Y = c|\mathbf{x}) = \arg\max_{c \in \mathcal{C}} \widehat{f}(\mathbf{x}|Y = c)\widehat{\mathbb{P}}(Y = c).$$

Para o caso binário, tem-se que a regra de classificação é $g(\mathbf{x}) = 1$ se, e só se,

$$\frac{\widehat{\mathbb{P}}(Y = 1|\mathbf{x})}{\widehat{\mathbb{P}}(Y = 0|\mathbf{x})} \geq K \iff \frac{\widehat{f}(\mathbf{x}|Y = 1)\widehat{\mathbb{P}}(Y = 1)}{\widehat{f}(\mathbf{x}|Y = 0)\widehat{\mathbb{P}}(Y = 0)} \geq K \iff$$

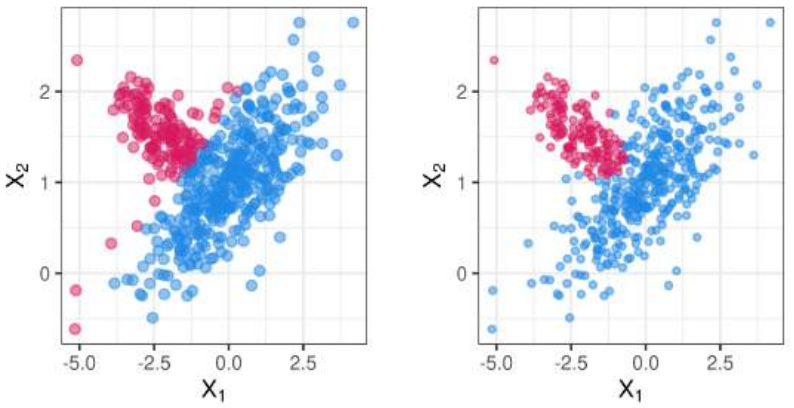
$$\log \widehat{f}(\mathbf{x}|Y = 1) - \log \widehat{f}(\mathbf{x}|Y = 0) \geq \log K + \log \widehat{\mathbb{P}}(Y = 0) - \log \widehat{\mathbb{P}}(Y = 1)$$

$$\iff -(\mathbf{x} - \widehat{\mu}_1)^\intercal \widehat{\Sigma_1}^{-1} (\mathbf{x} - \widehat{\mu}_1) + (\mathbf{x} - \widehat{\mu}_0)^\intercal \widehat{\Sigma_0}^{-1} (\mathbf{x} - \widehat{\mu}_0) \geq K'.$$

Assim, a região do espaço amostral $\mathbb{R}^d$ em que a regra de decisão é $g(\mathbf{x}) = 1$ é

dada por uma equação quadrática. É por esse motivo que o método recebe o nome de análise discriminante quadrática. Veja uma ilustração desta região no Exemplo 8.2.

Exemplo 8.2. A Figura 8.2 mostra, em um exemplo simulado, as classificações dadas pela análise discriminante linear e quadrática para o conjunto de teste apresentado na Figura 8.3. Nesse caso, a análise discriminante quadrática leva a classificações melhores.



(a) Análise Discriminante Linear (b) Análise Discriminante Quadrática

Figura 8.2: Valores preditos pela análise discriminante linear e valores preditos pela análise discriminante quadrática no Exemplo 8.2.

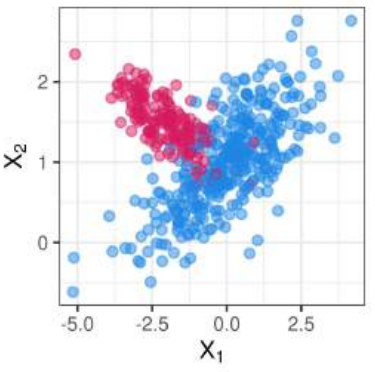


Figura 8.3: Conjunto de teste com os rótulos observados utilizados no Exemplo 8.2.

□

A análise discriminante quadrática pode ser ajustada no R via pacote `MASS`:

```
library(MASS)

qda_fit <- qda(x = X_treino, grouping = y_treino)
qda_pred <- predict(qda_fit, newdata = X_novo) # previsões
qda_pred$posterior # contém as estimativas de P(Y=c|x)
```

8.2 Support Vector Machines (SVM)

> Navego pela memória sem margens.
>
> Cecília Meireles

Support vector machines (SVM) é uma metodologia de classificação proposta por Cortes e Vapnik (1995) que leva a resultados excelentes em muitas aplicações. Essa técnica utiliza uma motivação muito diferente daquela associada aos classificadores vistos até aqui. No SVM, em nenhum momento estimamos as probabilidades $\mathbb{P}(Y = c|\mathbf{x})$; o resultado dessa técnica indica as classes estimadas de novas observações. Vamos assumir nessa seção que Y assume valores em $\mathcal{C} = \{-1, 1\}$.

Considere uma função linear

$$f(\mathbf{x}) := \beta_0 + \beta_1 x_1 + \ldots + \beta_d x_d.$$

O classificador $g(\mathbf{x})$ dado pelo SVM tem a seguinte forma:

$$\begin{cases} \text{Se } f(\mathbf{x}) < 0,\ g(\mathbf{x}) = -1 \\ \text{Se } f(\mathbf{x}) \geq 0,\ g(\mathbf{x}) = 1 \end{cases} \tag{8.1}$$

Para descrever como construir $f(\mathbf{x})$, suponha que as observações sejam linearmente separáveis, ou seja, existe um hiperplano que separa perfeitamente todas as observações do conjunto de treinamento de acordo com a classe (veja um exemplo na Figura 8.4).

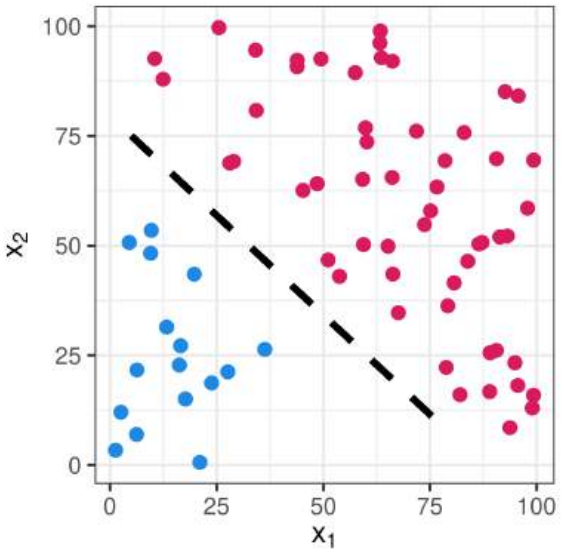


Figura 8.4: Exemplo de conjunto de dados no qual existe um hiperplano que os separa perfeitamente bem.

Nesse caso, existe $f(\mathbf{x})$ linear tal que $f(\mathbf{x}_i) < 0$ se, e só se, $y_i = -1$. Assim, podemos escrever para todo $i = 1, \ldots, n$,

$$y_i(\beta_0 + \beta_1 x_{i,1} + \ldots + \beta_d x_{i,d}) = y_i f(\mathbf{x}_i) > 0. \tag{8.2}$$

Quando existem muitos hiperplanos que separam os dados perfeitamente (ou seja, quando há várias funções f tais que a Eq. 8.2 está satisfeita), o SVM busca por aquele que tem maior margem M (ver Figura 8.5), isto é, aquele que fica "mais distante" de todos os pontos observados. Os pontos utilizados para definir as margens são chamados de vetores de suporte.

Assim, no caso de haver um hiperplano que separa perfeitamente bem os dados, buscamos o hiperplano com coeficientes $\boldsymbol{\beta}$ tais que

$$\boldsymbol{\beta} = \arg\max_{\boldsymbol{\beta}} M$$

sujeito às restrições

(1) $\sum_{i=1}^{d} \beta_i^2 = 1$ e

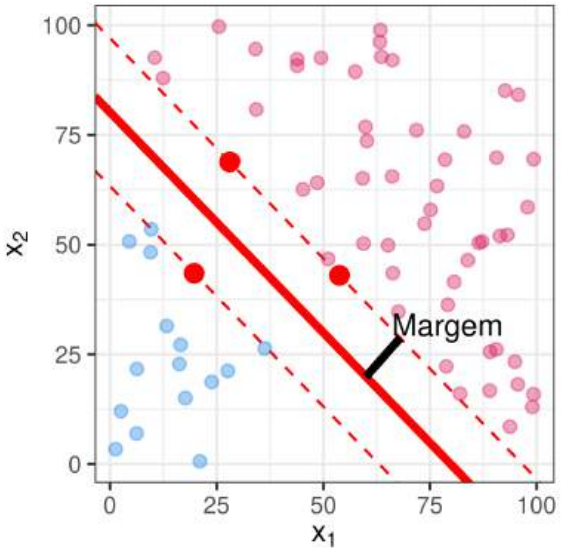


Figura 8.5: Exemplo de aplicação do SVM destacando os vetores de suporte e a margem.

(2) para todo $i = 1, \ldots, n$, $y_i f_\beta(\mathbf{x}_i) \geq M$.

A restrição (1) garante a comparabilidade dos diferentes hiperplanos em termos de sua norma e com essa restrição é possível mostrar que $|f(\mathbf{x})| = y_i f(\mathbf{x}_i)$ é a distância entre a observação e o hiperplano.

É importante notar que, além de assumir a existência de um hiperplano que separa perfeitamente os dados, essa formulação é muito sensível a pequenas mudanças nos dados. Na Figura 8.6 apresentamos as margens com os dados apresentados na Figura 8.4 alterando apenas uma observação no conjunto de dados.

Apresentamos uma solução que considera que os dados sejam linearmente separáveis. No entanto, em diversas situações isso não ocorre. Assim, a formulação utilizada pelo SVM se propõe a resolver uma generalização do problema descrito anteriormente, de modo que seja permitido que alguns dos pontos estejam do lado "errado" das margens (e eventualmente do hiperplano). Matematicamente, o SVM busca pela solução de

$$\arg\max_{\beta} M$$

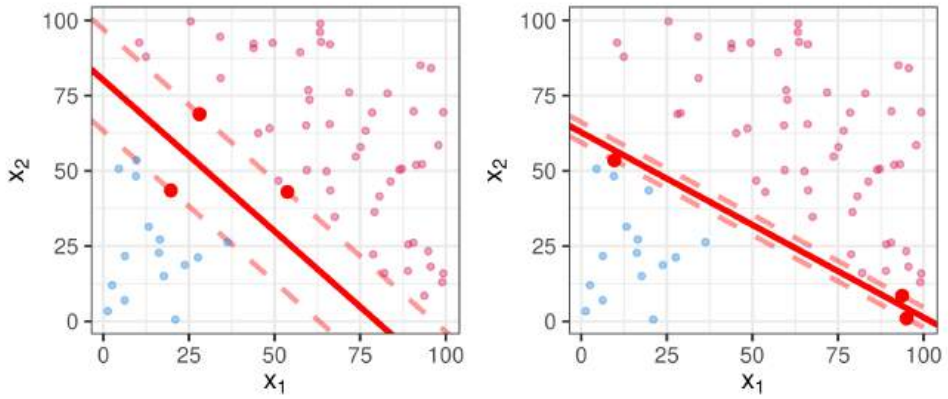


Figura 8.6: Exemplo de alteração nos vetores de suporte e margem devido a somente uma observação.

sujeito às restrições

(1) $\sum_{i=1}^{d} \beta_i^2 = 1$ e

(2) para todo $i = 1, \ldots, n$, $y_i f_\beta(\mathbf{x}_i) \geq M(1 - \epsilon_i)$, em que $\epsilon_i > 0$ e $\sum_{i=1}^{n} \epsilon_i \leq C$.

Note que ϵ_i pode ser maior que um, de modo que é permitido que $y_i f_\beta(\mathbf{x}_i)$ seja negativo, ou seja, que a i-ésima amostra fique do lado errado do hiperplano. O quão longe ela fica, contudo, é limitado por C, uma vez que ϵ_i não pode ser maior que C. Assim, C é hiperparâmetro (*tuning parameter*): quanto maior é seu valor, mais se permite que observações estejam do lado "errado" das margens.

Utilizando-se o truque do kernel (Seção 4.6.3.2), pode-se buscar por divisões mais complexas que hiperplanos. A ideia central é que a solução ótima $f(\mathbf{x})$ do SVM pode ser reescrita como

$$f(\mathbf{x}) = \beta_0 + \beta_1 x_1 + \ldots + \beta_d x_d = \beta_0 + \sum_{k=1}^{n} \alpha_k \langle \mathbf{x}, \mathbf{x}_k \rangle, \tag{8.3}$$

em que $\langle \mathbf{x}, \mathbf{x}_k \rangle = \sum_{i=1}^{d} x_i x_{k,i}$. Assim, para calcular f (isto é, os coeficientes α_k), tudo o que precisamos é do produto interno entre todas as observações. Pode-se também mostrar que, para calcular α_k, também necessita-se apenas dos produtos internos entre as observações. Podemos, portanto, trocar $\langle \mathbf{x}, \mathbf{x}_k \rangle$ por um kernel genérico $K(\mathbf{x}, \mathbf{x}_k)$ e, assim, utilizar o truque do kernel. Além disso, para todos os pontos que não são

vetores de suporte, os valores de α_i serão iguais a zero. Dessa forma, podemos reescrever a Equação 8.4 como:

$$f(\mathbf{x}) = \beta_0 + \sum_{k \in S} \alpha_k \langle \mathbf{x}, \mathbf{x}_k \rangle, \tag{8.4}$$

em que S indica o conjunto de todos os vetores de suporte.

Support Vector Machines também estão ligados a Reproducing Kernel Hilbert Spaces (Seção 4.6). Pode-se mostrar que, dado um kernel de Mercer K, o classificador g dado pelo SVM é aquele que minimiza

$$\arg\min_{g \in \mathcal{H}_K} \sum_{k=1}^{n} L(g(\mathbf{x}_k), y_k) + \lambda ||g||^2_{\mathcal{H}_K},$$

em que $L(g(\mathbf{x}_k), y_k) = (1 - y_k g(\mathbf{x}_k))_+$ (Pontil, 2003). Assim, pode-se utilizar o Teorema da Representação (Teorema 6) para encontrar os valores dos coeficientes α_k da Equação 8.4. Note que a perda L é zero quando $yg(\mathbf{x}) > 1$. No caso linear, essa restrição indica que $(\mathbf{x}, y)$ está do lado correto do hiperplano definido por g e $\mathbf{x}$ está a uma distância de ao menos $1/||g||_K$ do hiperplano. Ou seja, a perda é zero quando o exemplo é classificado corretamente de forma fácil. Note também que, no caso separável, essa função objetivo indica (como esperado) que o SVM procura, entre todos os hiperplanos separados, aquele com maior margem (isto é, maior valor de $1/||g||_K$).

O ajuste do SVM pode ser feito no R via o pacote `e1071`.

```
library(e1071)

# Kernel Linear
tune_out <- tune(svm, train.x = X_treino,
                 train.y = y_treino,
                 kernel = "linear", scale = TRUE,
                 ranges = list(cost = c(0.001,0.01,0.1,1,10)))

plot(tune_out)

melhor_modelo <- tune_out$best.model
y_pred <- predict(melhor_modelo, X_teste)
```

```
# Kernel Gaussiano
tune_out <- tune(svm, train.x = X_treino,
                 train.y = y_treino,
                 kernel = "radial", scale = TRUE,
                 ranges = list(cost = c(5,10,20),
                               gamma = c(0.001,0.01,0.05)))
plot(tune_out)

melhor_modelo <- tune_out$best.model
y_pred <- predict(melhor_modelo, X_teste)
```

8.3 Árvores de Classificação

Árvores de regressão podem ser utilizadas para resolver problemas de classificação considerando a ideia descrita na Seção 8.1.1. Alternativamente, pode-se utilizar árvores de classificação, que são o análogo de árvores de regressão (Seção 4.8), mas com a finalidade específica de se fazer classificação. Sua construção se dá de forma análoga à descrita naquela seção. Em particular, o processo de criar uma árvore grande e depois podá-la também ocorre. No entanto, é necessário fazer algumas adaptações. Primeiramente, a predição para a resposta Y de uma observação com covariáveis $\mathbf{x}$ que estão em uma região R_k não é mais dada pela média amostral das observações do conjunto de treinamento que pertecem à essa região (Equação 4.19), mas sim pela moda destas:

$$g(\mathbf{x}) = \text{moda}\{y_i : \mathbf{x}_i \in R_k\}. \tag{8.5}$$

Além disso, o critério utilizado para buscar a melhor partição em cada etapa do processo (I) também é diferente, uma vez que o erro quadrático (por exemplo, Equação 4.20) já não faz mais sentido. Um critério muito utilizado é o índice de Gini:

$$\sum_{R} \sum_{c \in \mathcal{C}} \widehat{p}_{R,c}(1 - \widehat{p}_{R,c}),$$

em que R representa uma das regiões induzidas pela árvore e $\widehat{p}_{R,c}$ é a proporção de observações classificadas como sendo da categoria c entre as que caem na região R.

Este índice é mínimo quando todas as proporções $\widehat{p}_{R,c}$ são zero ou um, indicando assim uma árvore "pura" (isto é, cada folha contém somente observações de uma única classe). Portanto, na prática, busca-se minimizar esse índice.

Exemplo 8.3. Para ilustrar uma aplicação do índice de Gini, considere a situação em que se deseja predizer se um hospital possui tomógrafo. Para isso contamos com informações sobre a presença de especialidade em cardiologia e se o hospital apresenta mais de 100 leitos. Os dados são apresentados na Tabela 8.2.

Tabela 8.2: Dados fictícios do Exemplo 8.3.

Cardiologia	**Acima de 100 leitos**	**Possui tomógrafo**
Sim	Sim	Sim
Sim	Não	Sim
Sim	Sim	Não
Sim	Não	Não
Não	Sim	Sim
Não	Sim	Sim
Não	Não	Não
Não	Não	Não
Não	Sim	Sim
Não	Sim	Sim

Assim, para determinar qual será a melhor partição (*cardiologia* ou *acima de 100 leitos*), utilizaremos o índice de Gini. Lembre-se que queremos determinar a partição que apresenta a maior "pureza", ou seja, o menor valor do índice. Dessa forma, teremos:

- $Gini(Cardiologia) = \frac{2}{4} \times \frac{2}{4} + \frac{4}{6} \times \frac{2}{6} = 0.472$
- $Gini(Leitos) = \frac{5}{6} \times \frac{1}{6} + \frac{1}{4} \times \frac{3}{4} = 0.326$

Logo, a partição que apresenta a maior "pureza" é dada pela partição da variável *leitos*.

□

Observação 8.2. À primeira vista, pode-se estranhar o fato do índice de Gini ser utilizado ao invés de, por exemplo, a proporção de erros. O motivo desta escolha é

que o índice de Gini é mais sensível a mudanças nas proporções de cada categoria nos nós.

□

Para a etapa da poda, em geral, utiliza-se a proporção de erros no conjunto de validação como estimativa do risco.

Assim como em árvores de regressão, podemos fazer árvores de classificação no R utilizando o pacote `rpart`.

```
set.seed(123)
library(rpart)

# Ajustar a árvore:
fit <- rpart(Species ~ .,
             method = "class", data = iris)

# poda:
melhor_cp <- fit$cptable[which.min(fit$cptable[, "xerror"]),
                         "CP"]

pfit <- rpart::prune(fit, cp = melhor_cp)

# plotar árvore podada
rpart.plot(pfit)
```

A Figura 8.7 ilustra a árvore gerada pelo código acima no R.

8.4 Bagging e Florestas Aleatórias

Tanto o bagging quanto florestas aleatórias, vistos no contexto de regressão na Seção 4.9, podem ser trivialmente adaptados para classificação. Para tanto, a agregação das diferentes árvores de classificação é feita através da função

$$g(\mathbf{x}) = \text{moda}\{g^b(\mathbf{x}), b = 1, \ldots, B\}$$

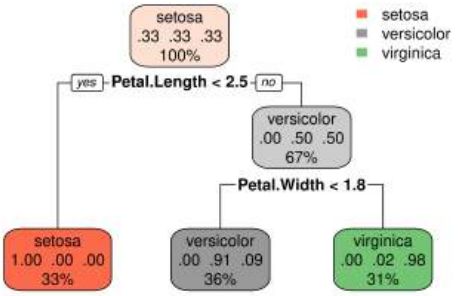


Figura 8.7: Árvore de classificação já podada.

ou seja, uma observação com covariáveis $\mathbf{x}$ é classificada por cada árvore construída e, posteriormente, a predição é dada pela categoria predita com maior frequência. No R, isso é feito pelo pacote `randomForests` ou `ranger`.

8.5 Boosting

Assim como no boosting de regressão (Seção 4.10), métodos de boosting também podem ser utilizados para agregar classificadores *fracos* de forma a construir um classificador mais poderoso. Aqui descreveremos uma forma de boosting específica, o *Adaboost.M1* (Freund e Schapire, 1995). Para tanto, assuma que $y_i \in \{-1,1\}$ e $\mathbf{x} \in R^p$. Lembre-se que, nesse contexto, utilizaremos classificadores sequenciais $g_b(\mathbf{x})$ com $b = 1,\ldots,B$. Portanto, o b-ésimo classificador dependerá diretamente do $(b-1)$-ésimo classificador.

A seguir, apresentamos o algoritmo para obter o classificador.

1. Inicialize os pesos $w_1 = \ldots = w_n = \frac{1}{n}$. Note que, inicialmente, todas as observações recebem o mesmo peso.

2. Para $b = 1,\ldots,B$:

   (a) Ajuste um classificador $g_b(\mathbf{x})$ para a amostra de treinamento utilizando os pesos $w_1,\ldots,w_n$

   (b) Calcule o erro ponderado $err_b = \frac{\sum_{i=1}^n w_i \mathbb{I}(y_i \neq g_b(\mathbf{x}_i))}{\sum_{i=1}^n w_i}$,

   (c) Calcule $\alpha_b = \log((1 - err_b)/err_b)$

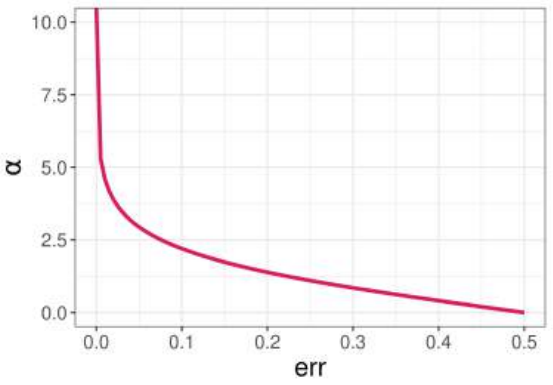


Figura 8.8: Peso do b-ésimo classificador em função do erro ponderado.

(d) Atualize $w_i \leftarrow w_i \exp\{\alpha_b \mathbb{I}(y_i \neq g_b(\mathbf{x}_i))\}$, $i = 1, \ldots, n$

3. Retorne o modelo final $g(\mathbf{x}) = \text{sinal}\left(\sum_{b=1}^{B} \alpha_b g_b(\mathbf{x})\right)$

Em geral, o passo 2a) desse método é executado utilizando-se um classificador bastante simples, como por exemplo uma árvore com 4 folhas. Já o passo 2c) apresenta o peso que o b-ésimo classificador terá no classificador combinado. Note que, quanto maior a proporção de erro do classificador, menor o peso ele terá na combinação final (Figura 8.8). Em particular, se $err_b > 50\%$, então $\alpha_b < 0$, ou seja, as predições de g_b são invertidas no classificador final. Esta inversão se dá pois o classificador $-g_b$ tem erro menor que 50% nesta situação.

A atualização dos pesos dada por $w_i \leftarrow w_i \exp(\alpha_b \mathbb{I}(y_i \neq g_b(\mathbf{x}_i)))$ tende a fazer com que observações classificadas erroneamente por g_b recebam peso maior no ajuste de g_{b+1}. Já as observações classificadas corretamente permanecem com peso w_i pois, nesse caso, $w_i \leftarrow w_i \exp(\alpha_b \mathbb{I}(y_i \neq g_b(\mathbf{x}_i))) = w_i \exp(\alpha_b \times 0) = w_i$.

Para se ajustar o adaboost no R, pode-se utilizar o pacote `gbm`.

```
library(gbm)

ajuste <- gbm.fit(x = X_treinamento,
                  y = y_treinamento,
                  n.trees = 100,
                  interaction.depth = 1,
```

```
                    distribution = "adaboost")

predito <- predict(ajuste, newdata = X_teste,
                   n.trees = ajuste$n.trees)
```

8.6 Método dos k Vizinhos Mais Próximos

O método do KNN (Seção 4.3) pode ser trivialmente adaptado para o contexto de classificação. Para tanto, pode-se definir o classificador

$$g(\mathbf{x}) = \text{moda}_{i \in \mathcal{N}_\mathbf{x}} y_i,$$

em que $\mathcal{N}_\mathbf{x}$ é o conjunto das k observações mais próximas de $\mathbf{x}$ (como definido na Seção 4.3). Assim, busca-se a classe mais frequentemente entre as observações mais próximas ao vetor de covariáveis $\mathbf{x}$ de interesse.

No R, esse método pode ser utilizado pelo pacote `FNN`.

```
library(FNN)

ajuste <- knn(train = X_treinamento,
              test = X_teste,
              cl = y_treinamento,
              k = k)

preditos <- ajuste$pred
```

8.7 Redes Neurais Artificiais

Redes neurais artificiais (Seção 4.11) também podem ser utilizadas no contexto de classificação. A estrutura da rede é essencialmente a mesma, contudo, em geral, há algumas diferenças:

- Ao invés de buscar apenas uma função de predição g_β, estima-se $|\mathcal{C}|$ funções, $g_{\beta,1}, \ldots, g_{\beta,|\mathcal{C}|}$, uma para cada categoria que a variável resposta pode assumir,

veja a Figura 8.9. A i-ésima dessas funções representa a probabilidade de Y assumir a categoria i, $i = 1, \ldots, |\mathcal{C}|$.

- Utiliza-se uma função de ativação f diferente da linear na camada de saída. Pode-se utilizar, por exemplo, a função *softmax*:

$$f_i(\mathbf{z}) = \frac{e^{z_i}}{\sum_{j=1}^{|\mathcal{C}|} e^{z_j}}.$$

  Aqui, $\mathbf{z}$ representa o vetor que contém todas as entradas na última camada e f_i é a função de ativação do i-ésimo neurônio da camada de saída. Essa função de ativação garante que as saídas estejam entre zero e um e que o vetor some 1.

- A função objetivo a ser minimizada também não é mais o EQM. Pode-se utilizar, por exemplo, a entropia cruzada, dada por

$$CE(g_{\beta,1}, \ldots, g_{\beta,|\mathcal{C}|}) = -\frac{1}{n} \sum_{k=1}^{n} \sum_{c \in \mathcal{C}} \mathbb{I}(y_k = c) \log(g_{\beta;c}(\mathbf{x}_k))$$

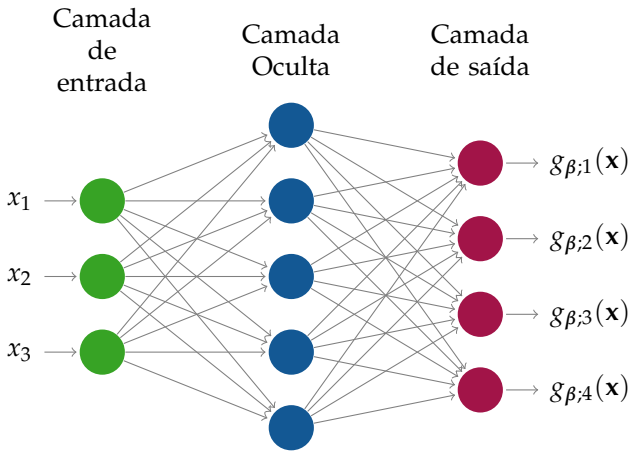


Figura 8.9: Exemplo de rede neural de classificação com uma camada oculta no caso em que Y assume quatro possíveis valores.

8.8 Exemplos

Exemplo 8.4 (Amazon Fine Food Reviews). A seguir iremos trabalhar novamente com o conjunto de dados Amazon Fine Food Reviews (apresentado no Exemplo 3.3). No entanto, nesse momento, trabalharemos com as notas categorizadas em máxima (escore igual a 5) vs demais (escore de 0 a 4).

Nesse exemplo utilizaremos 40.000 observações de treinamento e 10.000 de validação. Iremos utilizar regressão logística com máxima verossimilhança, penalização ridge e penalização lasso. Também utilizaremos árvore e floresta para a classificação.

```
library(data.table)
library(tm)
library(glmnet)
library(rpart)
library(rpart.plot)
library(xgboost)
library(ranger)
```

Primeiramente, faremos a leitura e transformação dos dados em uma matriz documento-termo (Apêndice A.2).

```
dados <- fread("../dados/Reviews.csv", header = TRUE)

set.seed(1)

# seleciona 50.000 observações
dados   <- dados[sample(nrow(dados), 50000),]

# indica observações de treinamento
tr <- sample.int(50000, 40000, replace = FALSE)

# categoriza escore
dados$Score <- ifelse(dados$Score <= 4, 0, 1)
corp <- VCorpus(VectorSource(dados$Text))

dtm <- DocumentTermMatrix(corp,
```

```
                            control =
                              list(tolower = TRUE,
                                   stemming = FALSE,
                                   removeNumbers = TRUE,
                                   removePunctuation = TRUE,
                                   removeStripwhitespace = TRUE,
                                   weighting = weightTf,
                                   bounds =
                                     list(global = c(50, Inf))))

dtmMatrix <- sparseMatrix(i = dtm$i, j = dtm$j, x = dtm$v,
                            dimnames = list(NULL,
                                            dtm$dimnames[[2]]),
                            dims = c(dtm$nrow, dtm$ncol))
```

Após a leitura dos dados, iremos ajustar os modelos de regressão logística com mínimos quadrados, ridge e lasso. Iremos atribuir a classe de escore 5 aos indivíduos do conjunto de validação que apresentarem probabilidade dessa classe maior ou igual a proporção de escore 5 no conjunto de treinamento (veja a Seção 9.1 para uma justificativa para esse critério). Note que, para essas técnicas, estamos utilizando o formato esparso para a matrix documento-termo.

```
# logística sem penalização
ajuste_glm <- glmnet(dtmMatrix[tr,], dados$Score[tr],
                     family = "binomial", alpha = 0, lambda = 0)

predito_glm <- ifelse(predict(ajuste_glm, s = 0,
                              newx = dtmMatrix[-tr,],
                              type = "response") >=
                        mean(dados$Score[tr]), 1, 0)

# logística com penalização ridge
vc_ridge <- cv.glmnet(dtmMatrix[tr,], dados$Score[tr],
                      family = "binomial", alpha = 0)
```

```
predito_ridge <- ifelse(predict(vc_ridge,
                                s = vc_ridge$lambda.1se,
                                newx = dtmMatrix[-tr,],
                                type = "response") >=
                        mean(dados$Score[tr]), 1, 0)

# logística com penalização lasso
vc_lasso <- cv.glmnet(dtmMatrix[tr,], dados$Score[tr],
                      family = "binomial", alpha = 1)

predito_lasso <- ifelse(predict(vc_lasso,
                                s = vc_lasso$lambda.1se,
                                newx = dtmMatrix[-tr,],
                                type = "response") >=
                        mean(dados$Score[tr]), 1, 0)
```

Para a árvore de classificação, será necessário fazer uma conversão para um `data.frame`.

```
dtmMatriz <- data.frame(as.matrix(dtmMatrix))

# organiza dados para o formato utilizado em rpart
dtmMatrixArvore <- cbind(Score = as.factor(dados$Score),
                         dtmMatriz)

# ajuste da arvore
arvore <- rpart(Score ~., method = "class",
                data = dtmMatrixArvore[tr,])

# poda
min_error <- which.min(arvore$cptable[,"xerror"])
melhorCp <- arvore$cptable[min_error, "CP"]

poda  <- rpart::prune(arvore, cp = melhorCp)
```

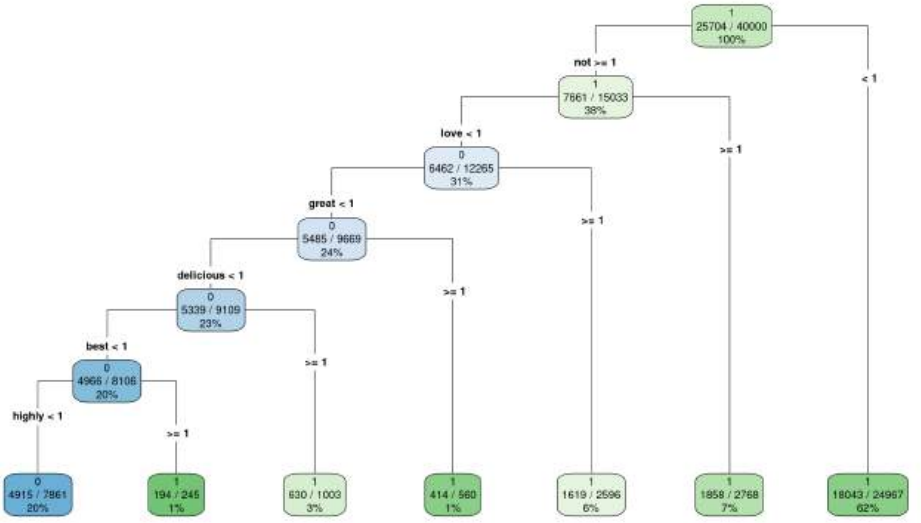


Figura 8.10: Árvore de classificação já podada para os dados da Amazon Fine Food Reviews.

```
# predito árvore
predito_arvore <- predict(poda, dtmMatrixArvore[-tr, -1],
                          type = "class")

# arvore
rpart.plot(poda, type = 4, extra = 102)
```

A Figura 8.10 ilustra a árvore podada gerada pelo código acima. Nessa figura, o primeiro número na forma geométrica indica a classe a qual o indivíduo é classificado em cada nó/folha, o número de classificações corretas em cada nó com o número de classificações em cada nó e, por fim, a porcentagem de observações em cada nó relativa ao total.

Para ajustar a floresta aleatória, utilizamos o pacote `ranger`.

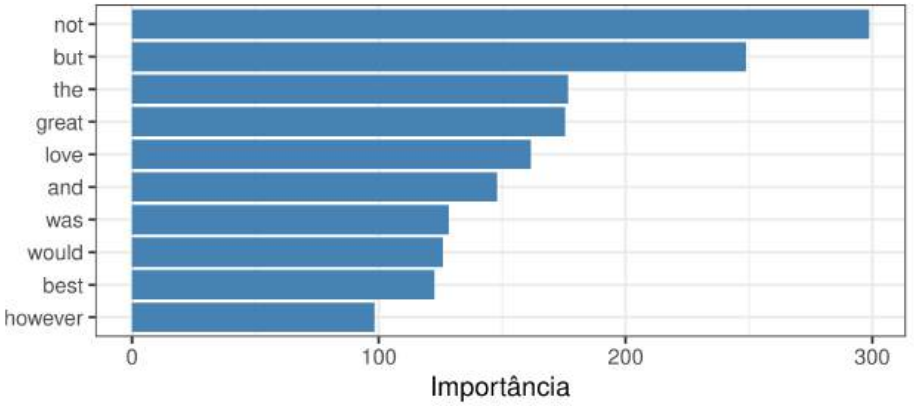


Figura 8.11: Importância obtida com floresta aleatória para os dados da Amazon Fine Food Reviews.

```
floresta <- ranger(Score ~ ., data = dtmMatrixArvore[tr,],
                   num.trees = 500,
                   importance = "impurity",
                   write.forest = TRUE,
                   verbose = FALSE)

predito_floresta <- predict(floresta,
                            data = dtmMatrixArvore[-tr, -1])
```

A Figura 8.11 mostra a importância das dez covariáveis mais importântes segundo florestas aleatórias.

Ajustaremos o boosting via o pacote `xgboost`.

```
bst <- xgboost(data = dtmMatrix[tr,],
               label = dados$Score[tr], nthread = 7,
               nround = 1000,
               objective = "binary:logistic",
               verbose = FALSE,lambda = 0.01)

predicoes   <- predict(bst, dtmMatrix[-tr,])
predito_bst <- ifelse(predicoes > mean(dados$Score[tr]), 1, 0)
```

A Tabela 8.3 mostra que regressão logística, regressão logística com penalização ridge e regressão logística com penalização lasso apresentam um desempenho muito semelhante, com uma ligeira vantagem para lasso. A árvore apresentou o pior desempenho geral e de especificidade entre todas as técnicas, embora tenha apresentado a segunda melhor sensibilidade. A floresta apresenta um desempenho geral muito próximo ao das três técnicas citadas anteriormente. No entanto, apresenta uma sensibilidade melhor e uma baixa especificidade. Já XGBoost apresentou o melhor dos resultados com respeito à porcentagem total de acerto.

Tabela 8.3: Desempenho preditivo (em porcentagem) dos métodos Mínimos Quadrados, Ridge, Lasso, Árvore e XGBoost.

Método	**Acurácia**	**Sensibilidade**	**Especificidade**
Logística MV	78.5	80.0	75.9
Logística Ridge	79.9	81.0	77.8
Logística Lasso	79.5	80.3	78.2
Árvore	68.8	88.6	34.1
Floresta Aleatória	81.1	93.6	59.3
XGBoost	81.7	84.6	76.7

□

Capítulo 9

Outros Aspectos de Classificação

9.1 Assimetria na Função de Perda, Conjuntos de Dados Desbalanceados e Outros Cortes

> Confidence is what you have before you understand the problem.
>
> Woody Allen

Como discutimos na Seção 7.4, nem sempre o risco $R(g) := \mathbb{E}[\mathbb{I}(Y \neq g(\mathbf{X}))] = \mathbb{P}(Y \neq g(\mathbf{X}))$ é razoável para avaliar o desempenho do classificador g. Isso acontece porque a função de perda $\mathbb{I}(g(\mathbf{X}) \neq Y)$ atribui perda 1 para todo tipo de erro cometido. Isso não é adequado em muitas situações. Por exemplo, erros de diferentes tipos podem ter diferentes custos. Considere o exemplo em que temos pacientes doentes ($Y = 1$) e saudáveis ($Y = 0$). Neste caso, se desejamos fazer um classificador para *screening* da doença, queremos um classificador que não classifique de forma errada pacientes que têm a enfermidade, enquanto que o erro oposto não é tão grave. Assim, a função de perda $\mathbb{I}(g(\mathbf{X}) \neq Y)$ não captura esse objetivo. De fato, se a doença é rara e as covariáveis não tem informação muito forte sobre Y, $\mathbb{P}(Y = 1|\mathbf{x})$ será sempre pequena e, portanto, o classificador de Bayes ($\mathbb{I}(\mathbb{P}(Y = 1|\mathbf{x}) \geq 0.5)$) será o clas-

sificador trivial $g(\mathbf{x}) \equiv 0$. Como métodos de classificação frequentemente buscam aproximar o classificador de Bayes, naturalmente eles também irão se aproximar do classificador trivial neste caso.

Existem diversas formas de contornar este problema. Uma abordagem comum consiste em buscar cortes diferentes de 1/2 em classificadores probabilísticos, isto é, buscar regras do tipo

$$g(\mathbf{x}) = \mathbb{I}(\widehat{\mathbb{P}}(Y = 1|\mathbf{x}) \geq K)$$

para diferentes cortes K. Uma forma de escolher K se dá com a utilização da curva ROC (*Receiver operating characteristic*), que mostra como a sensibilidade varia com a especificidade para diferentes valores de K. Para um exemplo, veja a Figura 9.1. Note que, para evitar que a sensibilidade e especificidade sejam subestimadas, esse gráfico deve ser feito com o conjunto de validação ou teste, nunca com o de treinamento.

É comum escolher K que maximize o valor de "Sensibilidade+Especificidade", embora esta não seja a única abordagem existente. Diferentes classificadores podem dar importâncias diferentes para essas medidas.

Alternativamente, pode-se definir o risco com base em outra outra função de perda que considere que erros diferentes podem ser penalizados de forma distinta. Por exemplo, pode-se tomar

$$R(g) = \mathbb{E}[(l_1\mathbb{I}(Y \neq g(\mathbf{X}) \text{ e } Y = 0)) + (l_0\mathbb{I}(Y \neq g(\mathbf{X}) \text{ e } Y = 1))] =$$

$$= l_1\mathbb{P}(Y \neq g(\mathbf{X}) \text{ e } Y = 0) + l_0\mathbb{P}(Y \neq g(\mathbf{X}) \text{ e } Y = 1),$$

em que l_1 e l_0 são pesos (custos) escolhidos para cada um dos tipos de erro. A função $g(\mathbf{x})$ que minimiza $R(g)$ é dada por

$$g(\mathbf{x}) = \mathbb{I}\left(\mathbb{P}(Y = 1|\mathbf{x}) > \frac{l_1}{l_0 + l_1}\right).$$

Essa fórmula mostra como o corte de classificadores plugin podem ser escolhidos de acordo com os pesos dos diferentes tipos de erro que podem ser cometidos[1]. Em particular, se escolhemos $l_0 = \pi_0$ (a probabilidade de uma observação pertencer à classe $Y = 0$) e $l_1 = \pi_1$ (a probabilidade de uma observação pertencer à classe $Y = 1$) em um contexto em que $\mathbb{P}(Y = 0) > \mathbb{P}(Y = 1)$, esse risco dá maior im-

[1] A ideia de atribuir custos diferentes pode também ser usada no caso em que Y pode assumir mais que duas categorias. Neste caso, o classificador ótimo tem o formato $\arg\min_{d \in C} c_d \mathbb{P}(Y = d|\mathbf{x})$, em que c_d é função dos custos especificados.

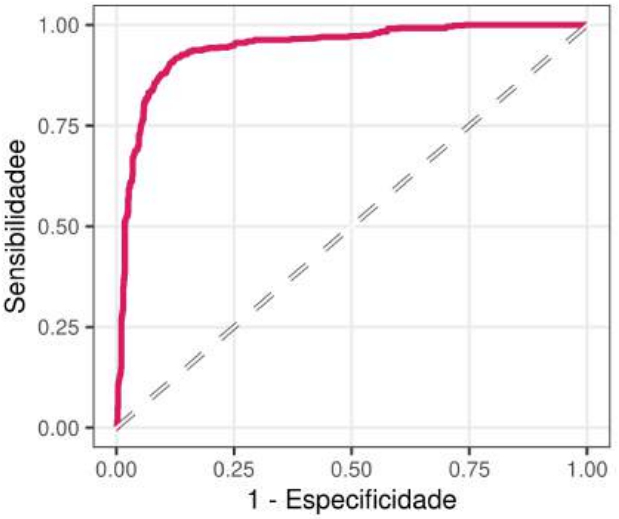


Figura 9.1: Exemplo de curva ROC da análise discriminante linear para o conjunto de dados de spams do Exemplo efex::spam

portância do erro de uma observação da classe 1 ser classificada como pertencente à classe 0 e menor importância ao erro de uma observação da classe 0 ser classificada como pertencente à classe 1. Nesse caso, a função $g(\mathbf{x})$ que minimiza $R(g)$ é dada por $g(\mathbf{x}) = \mathbb{I}(\mathbb{P}(Y = 1|\mathbf{x}) > \pi_1)$. Isso motiva o uso do classificador plugin $\mathbb{I}(\widehat{\mathbb{P}}(Y = 1|\mathbf{x}) \geq \widehat{\mathbb{P}}(Y = 1))$ em que $\widehat{\mathbb{P}}(Y = 1)$ é a proporção amostral da classe de interesse. Ou seja, ao invés de se utilizar $1/2$ como corte, utiliza-se $\widehat{\mathbb{P}}(Y = 1)$.

Para classificadores que não são baseados na estimação de $\mathbb{P}(Y = 1|\mathbf{x})$, outras abordagens são utilizadas. Por exemplo, a atribuição de pesos maiores a observações da categoria menos frequente. Esses pesos são então utilizados de modo que a contribuição de cada observação na criação do classificador seja proporcional ao peso por ela recebida. A forma como estes pesos são utilizados depende do método de classificação em questão. Também é possível alterar artificialmente a prevalência de cada categoria na amostra de treinamento com *super-sampling* ou *sub-sampling*. Mesmo que essas abordagens sejam úteis, é importante lembrar que a fonte do problema não é o desbalanceamento, mas sim os custos de classificação serem diferentes, de modo que o classificador de Bayes não é um bom classificador (já que ele é criado para minimizar apenas a probabilidade de erro).

9.2 Classificação vs Estimação de Probabilidades

> Hope is the confusion of the desire for a thing with its probability.
>
> Arthur Schopenhauer

Embora classificadores *plug-in* requeiram uma estimativa de $\mathbb{P}(Y = 1|\mathbf{x})$, uma estimativa que leva a boas classificações não é necessariamente uma estimativa razoável. Isso ocorre pois um classificador *plug-in* pode estar muito próximo ao classificador de Bayes mesmo sem que $\widehat{\mathbb{P}} \approx \mathbb{P}$: para o classificador *plug-in*, basta saber se uma probabilidade estimada é menor ou maior que o corte C. Assim, sempre que $\widehat{\mathbb{P}}(Y = 1|\mathbf{x}) - C$ tem o mesmo sinal que $\mathbb{P}(Y = 1|\mathbf{x}) - C$, o classificador *plug-in* concorda com o classificador de Bayes nesta observação.

Em muitas situações, diferente do objetivo comentado anteriormente, é importante estimar $\mathbb{P}$ bem. Por exemplo, frequentemente desejamos ordenar as observações segundo $\widehat{\mathbb{P}}$ para tomar decisões. Outra aplicação se dá inferência conformal (Seção 6.3), em que a consistência de $\widehat{\mathbb{P}}$ é necessária para criar regiões de predições pequenas. Assim, se o objetivo é estimar bem $\mathbb{P}$, outras funções de perda devem ser utilizadas como, por exemplo, a entropia cruzada (log-verossimilhança) ou o score de Brier, que são medidas *próprias* (Diniz et al., 2019; Lad, 1996). Em particular, é comum a utilização desses critérios para escolher *tuning parameters* para métodos de classificação probabilísticos, como na regressão logística com penalização.

9.3 Dataset Shift e Viés de Seleção

> I have not failed. I've just found 10,000 ways that won't work.
>
> Thomas Edison

Em uma parcela significativa de problemas de predição, a suposição básica de que as observações onde um classificador será aplicado são identicamente distribuídas às observações rotuladas utilizadas para construir g não é razoável. Esta situação é chamada de *dataset shift* (Sugiyama et al., 2017).

Muitas vezes, o *dataset shift* ocorre devido a um *viés de seleção* que faz com que os dados rotulados tenham características diferentes daqueles em que os classificadores

serão aplicados. Por exemplo, em pesquisas cosmológicas que para estimar a distância de uma galáxia até a Terra (Exemplo 1.2), é muito mais fácil observar a distância real de galáxias que são luminosas (Freeman et al., 2017; Izbicki et al., 2017) do que galáxias pouco luminosas. Isso faz com que o conjunto rotulado tenha mais galáxias luminosas que o conjunto ao qual os métodos de predição serão aplicados.

Dataset shift também pode ocorrer quando se utiliza dados de um certo domínio (por exemplo, resenhas da Amazon) para criar um classificador que será aplicado em outro domínio (por exemplo, textos do Twitter).

Para que seja possível aprender algo com o conjunto rotulado, é necessário fazer algumas suposição sobre como esse conjunto se relaciona ao conjunto em que o classificador será aplicado. Nesta seção descreveremos duas suposições: *covariate shift* (Seção 9.3.1) e *prior shift* (Seção 9.3.2). Em ambas as seções, assumiremos que o conjunto rotulado é dado por observações i.i.d.'s $(\mathbf{X}_1^L, Y_1^L), \ldots, (\mathbf{X}_{n_L}^L, Y_{n_L}^L)$, enquanto o conjunto em que a função de predição será aplicada é dado por observações i.i.d.'s $(\mathbf{X}_1^U, Y_1^U), \ldots, (\mathbf{X}_{n_U}^U, Y_{n_U}^U)$. Note que os valores de $Y_1^U, \ldots, Y_{n_U}^U$ não são observados.

9.3.1 Covariate Shift

A suposição de *covariate shift* consiste em assumir que $Y^L|\mathbf{X}^L \sim Y^U|\mathbf{X}^U$, de modo que a distribuição conjunta de $(\mathbf{X}^L, Y^L)$ difere de $(\mathbf{X}^U, Y^U)$ apenas segundo a distribuição marginal das covariáveis. Outra caracterização desta suposição é a de que o processo de escolha de que amostras serão rotuladas depende apenas das covariáveis medidas (veja Izbicki et al. 2017 para maior detalhamento).

À primeira vista, pode parecer que o *covariate shift* não é um problema para problemas de predição: como a distribuição de $Y|\mathbf{x}$ é a mesma em ambos os conjuntos de dados, então $\mathbb{P}(Y^L = 1|\mathbf{x}^L) = \mathbb{P}(Y^U = 1|\mathbf{x}^U)$. Assim, a quantidade $\mathbb{P}(Y^U = 1|\mathbf{x}^U)$, que é essencial para criar classificadores *plug-in*, é a mesma no conjunto rotulado e no conjunto de interesse. Note, contudo, que isso não implica que um estimador razoável de $\mathbb{P}(Y^L = 1|\mathbf{x}^L)$, segundo o conjunto rotulado, tenha bom desempenho no conjunto não-rotulado: frequentemente $\mathbb{P}(Y^L = 1|\mathbf{x}^L)$ estará bem estimado somente em regiões em que há bastantes dados rotulados, e essas regiões não necessariamente coincidem com regiões em que há bastantes dados não-rotulados; veja a Figura 9.2 para um exemplo no contexto de regressão. Levando em consideração apenas o conjunto rotulado, a reta azul parece ter bom desempenho, contudo isso não ocorre para o conjunto não-rotulado.

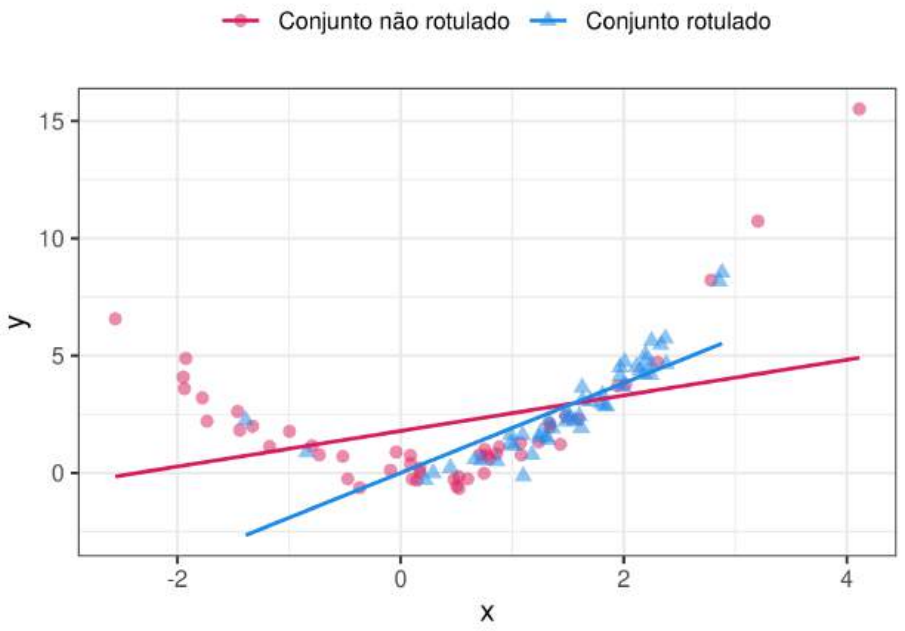


Figura 9.2: Ilustração do covariate shift em uma regressão linear: a linha azul é razoável para predizer novos dados do conjunto rotulado, mas ela está longe de ser ótima para o conjunto não-rotulado.

Sob uma perspectiva estatística, o problema ocorre porque a função de risco utilizada para construir o método de predição depende da distribuição de $\mathbf{X}$. Assim, um estimador que tem bom desempenho com respeito a $f_L(\mathbf{x})$ não necessariamente tem bom desempenho com respeito a $f_U(\mathbf{x})$.

Em cenários com *covariate shift*, enfrentamos dois problemas:

(i) como comparar métodos de predição

(ii) como construir bons métodos de predição.

Uma solução para ambos os problemas é atribuir pesos a cada observação rotulada de modo que a distribuição ponderada do conjunto rotulado seja próxima à distribuição do conjunto não-rotulado. Mais especificamente, a ideia chave para resolver (i) é notar que uma função de risco razoável é o risco *sob a distribuição do conjunto não-rotulado*, $\mathbb{E}[L(g(\mathbf{X}^U), Y^U)]$, em que L é uma função de perda (veja a Observação 1.2). Sob a suposição de *covariate shift*, este se relaciona com a distribuição

do conjunto rotulado da seguinte maneira:

$$\begin{aligned} R(g) &:= \mathbb{E}[L(g(\mathbf{X}^U), Y^U)] = \int L(g(\mathbf{x}), y) f_U(y|\mathbf{x}) f_U(\mathbf{x}) d\mathbf{x} dy = \\ &= \int L(g(\mathbf{x}), y) f_L(y|\mathbf{x}) \frac{f_U(\mathbf{x})}{f_L(\mathbf{x})} f_L(\mathbf{x}) d\mathbf{x} dy \\ &= \mathbb{E}\left[L(g(\mathbf{X}^L), Y^L) \beta(\mathbf{X}^L) \right], \end{aligned} \tag{9.1}$$

em que

$$\beta(\mathbf{x}) = \frac{f_U(\mathbf{x})}{f_L(\mathbf{x})}.$$

A função $\beta(\mathbf{x})$ é responsável por ponderar cada observação $\mathbf{x}$ de acordo com sua frequência no conjunto não-rotulado. Na prática, essa função deve ser estimada. Uma forma de se fazer isso é estimando a densidade de $\mathbf{x}$ no conjunto rotulado e no conjunto não-rotulado separadamente (pode-se utilizar, por exemplo, um estimador de densidade por kernel (Parzen, 1962)).

Um estimador alternativo para $\beta(\mathbf{x})$ consiste em criar o conjunto dados aumentado

$$(\mathbf{X}_1^L, S_1), \ldots, (\mathbf{X}_{n_L}^L, S_{n_L}), (\mathbf{X}_1^U, S_{n_L+1}), \ldots, (\mathbf{X}_{n_U}^U, S_{n_L+n_U}),$$

em que $S_i = \mathbb{I}(1 \leq i \leq n_L)$ é um indicador da amostra ser rotulada. Com base nessa amostra, estima-se então $\mathbb{P}(S = 1|\mathbf{x})$, a probabilidade de uma unidade amostral ser proveniente da amostra rotulada (utilizando, por exemplo, uma regressão logística). Como

$$\beta(\mathbf{x}) = \frac{\mathbb{P}(S=1)}{\mathbb{P}(S=0)} \frac{\mathbb{P}(S=0|\mathbf{x})}{\mathbb{P}(S=1|\mathbf{x})},$$

pode-se, então, estimar β com

$$\widehat{\beta}(\mathbf{x}) := \frac{\widehat{\mathbb{P}}(S=0)}{\widehat{\mathbb{P}}(S=1)} \frac{\widehat{\mathbb{P}}(S=1|\mathbf{x})}{\widehat{\mathbb{P}}(S=0|\mathbf{x})}.$$

Seja $\widehat{\beta}$ uma estimativa de β. A Equação 9.1 indica que uma forma de se estimar o risco de um método de predição g no conjunto não-rotulado é utilizar

$$\widehat{R}(g) = \frac{1}{\tilde{n}_L} \sum_{k=1}^{\tilde{n}_L} L(g(\tilde{\mathbf{X}}_k^L), \tilde{Y}_k^L) \widehat{\beta}(\tilde{\mathbf{X}}_k^L),$$

em que $(\tilde{\mathbf{X}}_1^L, \tilde{Y}_1^L), \ldots, (\tilde{\mathbf{X}}_{\tilde{n}_L}^L, \tilde{Y}_{\tilde{n}_L}^L)$ é uma amostra de validação do conjunto rotulado.

Isto é, com os pesos, pode-se estimar o risco de interesse utilizando-se apenas a amostra rotulada.

O estimador $\widehat{R}$ pode ser utilizado para se comparar diversos modelos. Note, contudo, que se β é muito alto para alguns pontos, o estimador do risco terá desempenho ruim, uma vez que consistirá efetivamente em avaliar a função de perda em apenas algumas observações. Em particular, $\beta(\mathbf{x})$ assume o valor zero em regiões do espaço amostral em que há apenas observações não-rotuladas, de modo que a suposição de *covariate shift* não é útil nesse caso.

Os pesos $\widehat{\beta}$ também podem ser utilizados para responder (ii), isto é, para criar funções de predição com bom poder preditivo no conjunto não-rotulado. Isso pode ser feito utilizando versões ponderadas dos métodos de predição estudados nesse livro. Por exemplo, a função `glmnet` do R permite que pesos sejam passados como um dos argumentos para o modelo a ser ajustado.

Note que a solução apresentada aqui para o problema de viés de seleção sob *covariate shift* pode ser utilizada tanto para problemas de classificação quanto para problemas de regressão, bastanto utilizar uma função de perda L adequada para cada problema.

9.3.2 Prior Shift

A suposição de *prior shift* consiste em assumir que $\mathbf{X}^L|Y^L \sim \mathbf{X}^U|Y^U$, de modo que a distribuição conjunta de $(\mathbf{X}^L, Y^L)$ difere de $(\mathbf{X}^U, Y^U)$ apenas segundo a distribuição marginal da variável resposta. Como $\mathbb{P}(Y^U = 1) \neq \mathbb{P}(Y^L = 1)$, então $\mathbb{P}(Y^U = 1|\mathbf{x}^U) \neq \mathbb{P}(Y^L = 1|\mathbf{x}^L)$, de modo que alguma correção deve ser feita para estimar $\mathbb{P}(Y^U = 1|\mathbf{x}^U)$. Pode-se mostrar que, sob a suposição de *prior shift* (Saerens et al., 2002),

$$\mathbb{P}(Y^U = 1|\mathbf{x}^U) = \frac{\frac{\mathbb{P}(Y^U=1)}{\mathbb{P}(Y^L=1)}\mathbb{P}(Y^L = 1|\mathbf{x}^L)}{\sum_{i=0}^{1}\frac{\mathbb{P}(Y^U=i)}{\mathbb{P}(Y^L=i)}\mathbb{P}(Y^L = i|\mathbf{x}^L)}.$$

Assim, para se estimar $\mathbb{P}(Y^U = 1|\mathbf{x}^U)$ basta estimar:

- $\mathbb{P}(Y^L = 1|\mathbf{x}^L)$, que pode ser feito utilizando as técnicas de classificação via estimação de probabilidades apresentadas na Parte II deste livro;
- $\mathbb{P}(Y^L = 1)$, que pode ser feito utilizando a proporção de amostras com rótulo $Y = 1$ na amostra rotulada;

- $\mathbb{P}(Y^U = 1)$, que é um termo mais complicado para ser estimado.

Em muitos problemas, uma estimativa razoável de $\mathbb{P}(Y^U = 1)$ é conhecida de antemão. Por exemplo, considere um estudo caso-controle para criar um classificador para determinar se um paciente tem determinada doença. Nesse caso, a amostra rotulada é criada de modo que o número de pacientes doentes (e não doentes) investigados seja um número pré-fixado anteriormente. Assim, a proporção de doentes na amostra é diferente da proporção de doentes na população, $\mathbb{P}(Y^U = 1)$. Contudo, $\mathbb{P}(Y^U = 1)$ é frequentemente conhecido devido a estudos prévios.

Quando $\mathbb{P}(Y^U = 1)$ não é conhecida, é necessário estimá-la. Um método bastante popular para estimar essa quantidade foi desenvolvido por (Forman, 2006). Esse método consiste em notar que, se $g(\mathbf{x})$ é um classificador, então (Vaz et al., 2019)

$$\mathbb{P}(Y^U = 1) = \frac{\mathbb{P}(g(\mathbf{X}^U) = 1) - \mathbb{P}(g(\mathbf{X}^L) = 1|Y^L = 0)}{\mathbb{P}(g(\mathbf{X}^L) = 1|Y^L = 1) - \mathbb{P}(g(\mathbf{X}^L) = 1|Y^L = 0)}.$$

Todos os termos desta equação podem ser facilmente estimados utilizando o conjunto de dados disponível.

9.4 Combinando Classificadores

Combinar diversos classificadores construídos com base no mesmo conjunto de treinamento pode levar a um classificador com poder preditivo ainda maior. Bagging, florestas aleatórias e boosting (Seções 8.4 e 8.5) são algumas formas de se fazer isso. Como vimos, enquanto bagging e florestas aleatórias foram criados para combinar funções de predição aproximadamente não viesadas (mais especificamente, árvores sem poda); boosting combina classificadores fracos. Existem, contudo, outras formas de se combinar classificadores; nesta seção exploramos uma delas, o *blending* (muitas vezes chamado de *stacked ensembling*).

A ideia deste método é criar um *meta-classificador*, treinado com covariáveis dadas pelas predições dos modelos a serem combinados. Mais especificamente, divide-se o conjunto de treinamento em duas partes: treinamento (por exemplo, 90%) e ensemble set (por exemplo, 10%). Sejam $g_1, \ldots, g_B$ as funções de predição treinadas utilizando-se o conjunto de treinamento. Essas funções podem ser obtidas obtidas a partir de quaisquer métodos (por exemplo, regressão logística, SVM, florestas aleatórias etc). Denotando o ensemble set por $(\tilde{\mathbf{x}}_1, \tilde{y}_1), \ldots, (\tilde{\mathbf{x}}_{\tilde{n}}, \tilde{y}_{\tilde{n}})$, ajustamos então um

classificador para o conjunto $(\tilde{\mathbf{w}}_{\tilde{1}}, \tilde{y}_{\tilde{1}}), \ldots, (\tilde{\mathbf{w}}_{\tilde{n}}, \tilde{y}_{\tilde{n}})$, em que $\tilde{\mathbf{w}}_i = (g_1(\tilde{\mathbf{x}}_i), \ldots, g_B(\tilde{\mathbf{x}}_i))$.

Exemplo 9.1 (Detecção de SPAMs). Considere novamente o conjunto de dados `spam` do Exemplo 8.1. Ajustamos aqui os seguintes métodos: análise discriminante linear, regressão logística com penalização L1, florestas aleatórias e SVM. Para combinar os modelo, utilizamos a regressão logística.

Tabela 9.1: Coeficientes encontrados pela regressão logística do modelo ensemble. A floresta aleatória recebe o maior peso na combinação.

	Coeficientes
Intercepto	-5.269
Pred. Análise Discriminante Linear	-2.152
Pred. Logística L1	3.114
Pred. Floresta Aleatória	8.542
Pred. SVM	0.914

Tabela 9.2: Risco estimado de cada modelo individual e do modelo de ensemble.

Método	**Risco**
Análise Descriminante Linear	0.106
Regressão Logística L1	0.077
FLoresta Aleatória	0.067
SVM	0.073
Ensemble	0.061

A Tabela 9.1 mostra que quase todo o peso do modelo logístico de ensemble foi dado ao modelo de florestas aleatórias. Isso se justifica porque o desempenho de florestas aleatórias é semelhante ao do modelo combinado (Tabela 9.2).

□

Para outros métodos de ensemble, veja Breiman (1996), Coscrato et al. (2020), Dietterich (2000) e Izbicki e Musetti (2020).

9.5 Teoria do Aprendizado Estatístico

> It is a dimension as vast as space and as timeless as infinity. It is the middle ground between light and shadow, between science and superstition, and it lies between the pit of man's fears and the summit of his knowledge.
>
> The Twilight Zone

Neste seção introduzimos parte da teoria do aprendizado estatístico desenvolvida por Cherkassky et al. (1999) e Vapnik (1999), cujo objetivo é fornecer garantias sobre alguns procedimento de classificação.

Sejam $\mathbf{Z}_i = (\mathbf{X}_i, Y_i)$, $i = 1,\ldots,n$ vetores aleatórios i.i.d. e $f : \mathbb{R}^{d+1} \longrightarrow \mathbb{R}$ uma função. Defina

$$P(f) := \mathbb{E}[f(\mathbf{Z})] \qquad \text{e} \qquad P_n(f) = \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{Z}_i)$$

e seja

$$\widehat{f} = \arg\min_{f\in\mathcal{F}} P_n(f) \qquad \text{e} \qquad f^* = \arg\min_{f\in\mathcal{F}} P(f),$$

em que $\mathcal{F}$ é um conjunto de funções de interesse. O principal objetivo da teoria do aprendizado estatístico é dar garantias sobre o quão longe $P(\widehat{f})$ está de $P(f^*)$. Embora a teoria do aprendizado estatístico seja bastante geral, focaremos inicialmente no problema de classificação:

Exemplo 9.2. No contexto de classificação, podemos tomar $f(\mathbf{Z}) = \mathbb{I}(Y \neq g(\mathbf{X}))$, de modo que $P(f) = R(g)$ (isto é, é o risco de classificação) e $P_n(f) = \widehat{R}_{tr}(g) := \frac{1}{n}\sum_{i=1}^{n}\mathbb{I}(Y_i \neq g(\mathbf{X}_i))$ (isto é, é a proporção de erros que g faz no conjunto de treinamento). Uma forma de se escolher um classificador é definir

$$\widehat{g} = \arg\min_{g\in\mathcal{G}} \widehat{R}_{tr}(g),$$

em que $\mathcal{G}$ é uma classe de classificadores (por exemplo, o conjunto de todos os clas-

sificadores lineares). Essa abordagem é chamada de *minimização do erro empírico*. Assim, a teoria do aprendizado estatístico fornece garantias sobre este procedimento. Mais especificamente, essa teoria fornece garantias sobre o quão longe o risco de $\widehat{g}$ está do risco do oráculo g^*, definido por

$$g^* = \arg\min_{g \in \mathcal{G}} R(g),$$

ou seja, o melhor classificador dentro de $\mathcal{G}$ (como visto no Teorema 10). Isso decorre do fato de que $\min_{g \in \mathcal{G}} \widehat{R}_{tr}(g) = \min_{f \in \mathcal{F}} \widehat{P}_n(f)$ e $\min_{g \in \mathcal{G}} R(g) = \min_{f \in \mathcal{F}} P(f)$, em que $\mathcal{F} = \{f : f(\mathbf{Z}) = \mathbb{I}(Y \neq g(\mathbf{X})) \text{ para algum } g \in \mathcal{G}\}$. Note que uma garantia desse tipo já foi dada no Teorema 10. Contudo, foi considerado apenas o caso em que a busca é feita sob um número finito de classificadores. A teoria VC apresenta uma generalização para que a classe de funções possa ser infinita.

□

A estratégia da teoria do aprendizado estatístico para limitar o quão longe o risco de $\widehat{g}$ está do risco do oráculo g^* é limitar o quão longe $R(g)$ está de $\widehat{R}_{tr}(g)$ *para todo* $g \in \mathcal{G}$ *simultaneamente.* Uma forma de se obter este limitante se dá com o Teorema VC:[2]

Teorema 11 (Teorema VC). *Se $\mathcal{F}$ é um conjunto de funções binárias, então, para todo $\epsilon > \sqrt{2/n}$,*

$$\mathbb{P}\left(\sup_{f \in \mathcal{F}} |P_n(f) - P(f)| > \epsilon\right) \leq 4s(\mathcal{F}, 2n)e^{-n\epsilon^2/8}, \tag{9.2}$$

em que $s(\mathcal{F}, 2n)$ é o shattering number *da classe $\mathcal{F}$ para uma amostra de tamanho n. Assim, com probabilidade ao menos $1 - \delta$,*

$$\sup_{f \in \mathcal{F}} |P_n(f) - P(f)| \leq \sqrt{\frac{8}{n} \log\left(\frac{4s(\mathcal{F}, 2n)}{\delta}\right)}.$$

O *shattering number* $s(\mathcal{F}, 2n)$ é uma medida de quão complexa e expressiva é a classe de funções $\mathcal{F}$. Antes de introduzir formalmente essa quantidade, vamos discutir o motivo desse teorema, de fato, nos ajudar a atingir nosso objetivo. O fato

[2] VC é a abreviação de Vapnik-Chervonenkis, os sobrenomes dos criadores dessa teoria.

central da teoria VC é que o Teorema 11 implica que, com probabilidade ao menos $1 - 4s(\mathcal{F}, 2n)e^{-2n\epsilon^2/8}$,

$$\begin{aligned} P(\widehat{f}) - P(f^*) &= P(\widehat{f}) - P_n(\widehat{f}) + P_n(\widehat{f}) - P_n(f^*) + P_n(f^*) - P(f^*) \\ &\leq P(\widehat{f}) - P_n(\widehat{f}) + P_n(f^*) - P(f^*) \\ &\leq 2\epsilon, \end{aligned}$$

onde a primeira desigualdade segue do fato de que, por definição, $P_n(\widehat{f}) - P_n(f^*) \leq 0$ e a última desigualdade segue da Equação 9.2. Assim, se n é grande e $s(\mathcal{F}, 2n)$ cresce a uma taxa menor que $e^{-n\epsilon^2/8}$ (em breve veremos quando esta última condição ocorre), então com probabilidade alta $P(\widehat{f}) \approx P(f^*)$. No contexto de classificação (Exemplo 9.2), o Teorema 11 garante que

$$\mathbb{P}\left(\sup_{g \in \mathcal{G}} |\widehat{R}_{tr}(g) - R(g)| > \epsilon\right) \leq 4s(\mathcal{F}, 2n)e^{-n\epsilon^2/8},$$

o que implica que, com probabilidade alta, o risco do estimador encontrado via minimização de risco empírico estará próximo do risco do oráculo.

Note que, em particular, o Teorema VC garante que, se

$$4s(\mathcal{F}, 2n)e^{-2n\epsilon^2/8} \xrightarrow[n \longrightarrow \infty]{} 0, \qquad \text{então} \qquad P(\widehat{f}) \xrightarrow[n \longrightarrow \infty]{\mathbb{P}} P(f^*),$$

isto é, $P(\widehat{f})$ converge em probabilidade para $P(f^*)$.

Shattering number. Note que, se $\mathcal{F}$ contém apenas funções binárias, então $(f(\mathbf{z}_1), \ldots, f(\mathbf{z}_n)) \in \{0,1\}^n$ para todo $\mathbf{z}_1, \ldots, \mathbf{z}_n$ e $f \in \mathcal{F}$. Seja

$$\mathcal{F}(\mathbf{z}_1, \ldots, \mathbf{z}_n) = \{(f(\mathbf{z}_1), \ldots, f(\mathbf{z}_n)) : f \in \mathcal{F}\}$$

o conjunto de todos os valores que o vetor $(f(\mathbf{z}_1), \ldots, f(\mathbf{z}_n))$ pode assumir para $\mathbf{z}_1, \ldots, \mathbf{z}_n$ fixos. Se $\mathcal{F}$ é uma classe rica de funções, esperamos que $\mathcal{F}(\mathbf{z}_1, \ldots, \mathbf{z}_n)$ contenha um grande número de vetores. Com base nessa ideia, uma forma de quantificar a complexidade de uma classe $\mathcal{F}$ de funções binárias é através do número máximo de elementos que $\mathcal{F}(\mathbf{z}_1, \ldots, \mathbf{z}_n)$ pode ter levando em conta todos os possíveis valores que $\mathbf{z}_1, \ldots, \mathbf{z}_n$ podem assumir. Esta quantidade é chamada de *shattering number*. Mais

especificamente, este número é dado por

$$s(\mathcal{F},n) := \sup_{\mathbf{z}_1,\ldots,\mathbf{z}_n} |\mathcal{F}(\mathbf{z}_1,\ldots,\mathbf{z}_n)|.$$

Note que, necessariamente, $s(\mathcal{F},n) \leq 2^n$. Quando $|\mathcal{F}(\mathbf{z}_1,\ldots,\mathbf{z}_n)| = 2^n$, dizemos que os dados $\mathbf{z}_1,\ldots,\mathbf{z}_n$ são destruídos (shattered) por $\mathcal{F}$.

Exemplo 9.3. Seja $\mathcal{F} = \{f_t : t \in \mathbb{R}\}$, em que $f_t(z) = \mathbb{I}(z > t)$, $z \in \mathbb{R}$. Para quaisquer $z_1 < \ldots < z_n$, temos que

$$\mathcal{F}(z_1,\ldots,z_n) = \{(0,\ldots,0,0),(0,\ldots,0,1),(0,\ldots,1,1),\ldots,(1,\ldots,1,1)\}.$$

Logo $|\mathcal{F}(z_1,\ldots,z_n)| = n+1$. Em geral, $|\mathcal{F}(z_1,\ldots,z_n)| \leq n+1$, de modo que, neste caso, $s(\mathcal{F},n) = n+1$.

□

Uma condição necessária e suficiente para a consistência (em relação ao oráculo) do procedimento de minimização empírica do risco é que

$$\log s(\mathcal{F},n)/n \longrightarrow 0.$$

Assim, se $\mathcal{F}$ é muito complexa (e.g., $s(\mathcal{F},n) = 2^n$) não há consistência.

Dimensão VC. Outro conceito importante na Teoria VC é o de *dimensão VC*, que também é uma forma de medir a complexidade de um espaço de funções binárias. Essa medida é dada por:

$$VC(\mathcal{F}) = \sup\{n : s(\mathcal{F},n) = 2^n\}.$$

Assim, $VC(\mathcal{F})$ é o maior tamanho amostral n para o qual existe uma amostra $\mathbf{z}_1,\ldots,\mathbf{z}_n$ que é destruída por $\mathcal{F}$. Se $\mathcal{F}$ contém uma grande variedade de funções, é razoável esperar que sua dimensão VC seja grande.

Exemplo 9.4. No Exemplo 9.3, $VC(\mathcal{F}) = 1$, pois $s(\mathcal{F},1) = 2 = 2^1$, mas $s(\mathcal{F},n) = n+1 < 2^n$ para $n > 1$.

□

A dimensão VC também está relacionada com o *shattering number* via o Lema de Sauer:

Teorema 12 (Lema de Sauer). *Se* $d := VC(\mathcal{F}) < \infty$, *então*

$$s(\mathcal{F}, n) \leq \left(\frac{en}{d}\right)^d$$

para todo $n > d$. *Além disso, se a dimensão VC é infinita, então* $s(\mathcal{F}, n) = 2^n$ *para todo* n.

Esse lema é útil pois permite uma forma alternativa para o limitante do Teorema VC. Mais especificamente, se a dimensão VC d de uma classe $\mathcal{F}$ é finita, então o Teorema VC juntamente com o Lema de Sauer implica que, se $n > d$, então com probabilidade ao menos $1 - \delta$

$$\sup_{f \in \mathcal{F}} |P_n(f) - P(f)| \leq \sqrt{\frac{8}{n}\left(\log\left(\frac{4}{\delta}\right) + d\log\left(\frac{ne}{d}\right)\right)}.$$

Segue que se a dimensão VC de $\mathcal{F}$ é finita, então $P(\widehat{f})$ converge em probabilidade para $P(f^*)$. Além disso, se a dimensão VC de $\mathcal{F}$ é infinita, então $P(\widehat{f})$ não converge em probabilidade para $P(f^*)$. Ainda, a dimensão VC caracteriza completamente a consistência do método de minimização empírica do risco em uma classe de funções.

Exemplo 9.5. [Data splitting] Seja $\mathbf{Z}_1, \ldots, \mathbf{Z}_n$ o conjunto de validação em um problema de classificação e $\mathbb{G}$ é uma classe de classificadores parametrizadas por um *tuning parameter* λ. A Teoria VC garante que

$$\mathbb{P}\left(\sup_{g \in \mathcal{G}} |\widehat{R}_{val}(g) - R(g)| > \epsilon\right) \leq 4s(\mathcal{F}, 2n)e^{-n\epsilon^2/8},$$

em que $\widehat{R}_{val}$ é o risco estimado de g pelo conjunto de validação, de modo que ela fornece limitantes superiores para o quão próximo λ escolhido por data splitting está do valor ótimo de λ. Além disso, essa teoria implica que o processo de escolha do melhor λ via data splitting será consistente se, e somente se, a dimensão VC da classe $\mathcal{F} = \{f : f(\mathbf{Z}) = \mathbb{I}(Y \neq g(\mathbf{X})) \text{ para algum } g \in \mathcal{G}\}$ for finita. Para outras aplicações da Teoria VC para data splitting, veja Izbicki et al. (2019) e Izbicki e Stern (2013).

□

9.5.1 Prova do teorema VC

Inicialmente limitaremos o quão longe $P_n(f)$ está de $P(f)$ para uma função f fixa. Uma forma de fazer isso é utilizando a desigualdade de Hoeffding:

Teorema 13 (Desigualdade de Hoeffding). *Se $W_1, \ldots, W_n$ são variáveis aleatórias independentes e W_i tem suporte em (a_i, b_i) então, para todo $t > 0$,*

$$\mathbb{P}(|\bar{W}_n - \mu| > \epsilon) \leq 2e^{-2n^2\epsilon^2 / \sum_{i=1}^n (b_i - a_i)^2},$$

em que $\bar{W}_n = n^{-1}\sum_{i=1}^n W_i$ e $\mu = \mathbb{E}[\bar{W}_n]$.

Dessa desigualdade, segue imediatamente que

Corolário 1. *Para toda função binária f,*

$$\mathbb{P}(|P_n(f) - P(f)| > \epsilon) \leq 2e^{-2n\epsilon^2}$$

e, portanto, com probabilidade ao menos $1 - \delta$,

$$|P_n(f) - P(f)| \leq \sqrt{\frac{1}{2n}\log(2/\delta)}.$$

Exemplo 9.6. No contexto de classificação, o Corolário 1 implica que, para todo classificador binário g,

$$\mathbb{P}(|\widehat{R}_{tr}(g) - R(g)| > \epsilon) \leq 2e^{-2n\epsilon^2}$$

e, portanto, com probabilidade ao menos $1 - \delta$,

$$|\widehat{R}_{tr}(g) - R(g)| \leq \sqrt{\frac{1}{2n}\log(2/\delta)}.$$

□

Lema 3 (Lema da simetrização). *Para todo $\epsilon > \sqrt{2/n}$,*

$$\mathbb{P}\left(\sup_{f\in\mathcal{F}} |P_n(f) - P(f)| > \epsilon\right) \leq 2\mathbb{P}\left(\sup_{f\in\mathcal{F}} |P_n(f) - P_n'(f)| > \epsilon/2\right),$$

em que $P_n'(f) = \frac{1}{n}\sum_{i=1}^n f(\mathbf{Z}_i')$, com $\mathbf{Z}_1', \ldots, \mathbf{Z}_n'$ uma amostra de vetores aleatórios iids a $\mathbf{Z}_1, \ldots, \mathbf{Z}_n$.

Demonstração. Seja $\widehat{f} = \arg\sup_{f \in \mathcal{F}} |P_n(f) - P(f)|$. Note que, pela desigualdade triangular,

$$\epsilon < |P_n(\widehat{f}) - P(\widehat{f})| = |P_n(\widehat{f}) - P'_n(\widehat{f}) + P'_n(\widehat{f}) - P(\widehat{f})| \leq |P_n(\widehat{f}) - P'_n(\widehat{f})| + |P'_n(\widehat{f}) - P(\widehat{f})|.$$

Assim,

$$|P_n(\widehat{f}) - P(\widehat{f})| > \epsilon \text{ e } |P'_n(\widehat{f}) - P(\widehat{f})| \leq \epsilon/2 \Rightarrow |P_n(\widehat{f}) - P'_n(\widehat{f})| > \epsilon/2.$$

Logo,

$$\mathbb{I}(|P_n(\widehat{f}) - P(\widehat{f})| > \epsilon)\mathbb{I}(|P'_n(\widehat{f}) - P(\widehat{f})| \leq \epsilon/2) \leq \mathbb{I}(|P_n(\widehat{f}) - P'_n(\widehat{f})| > \epsilon/2).$$

Tomando a esperança sobre $\mathbf{Z}'_1, \ldots, \mathbf{Z}'_n$, concluímos que

$$\mathbb{I}(|P_n(\widehat{f}) - P(\widehat{f})| > \epsilon)\mathbb{P}'(|P'_n(\widehat{f}) - P(\widehat{f})| \leq \epsilon/2) \leq \mathbb{P}'(|P_n(\widehat{f}) - P'_n(\widehat{f})| > \epsilon/2). \quad (9.3)$$

Mas, pela desigualdade de Chebyshev,

$$\mathbb{P}'(|P(\widehat{f}) - P'_n(\widehat{f})| > \epsilon/2) \leq \frac{4\mathbb{V}'[\widehat{f}]}{n\epsilon^2} \leq \frac{1}{n\epsilon^2} \leq \frac{1}{2}.$$

Logo,

$$\mathbb{P}'(|P(\widehat{f}) - P'_n(\widehat{f})| \leq \epsilon/2) \geq \frac{1}{2}.$$

Desse fato e da Equação 9.3, concluímos que

$$\mathbb{I}(|P_n(\widehat{f}) - P(\widehat{f})| > \epsilon) \leq 2\mathbb{P}'(|P_n(\widehat{f}) - P'_n(\widehat{f})| > \epsilon/2).$$

Tomando a esperança sobre $\mathbf{Z}_1, \ldots, \mathbf{Z}_n$, concluímos que

$$\mathbb{P}(|P_n(\widehat{f}) - P(\widehat{f})| > \epsilon) \leq 2\mathbb{P}(|P_n(\widehat{f}) - P'_n(\widehat{f})| > \epsilon/2).$$

Assim,

$$\mathbb{P}\left(\sup_{f \in \mathcal{F}} |P_n(f) - P(f)| > \epsilon\right) \leq 2\mathbb{P}\left(\sup_{f \in \mathcal{F}} |P_n(f) - P'_n(f)| > \epsilon/2\right)$$

□

Demonstração. **[Prova do Teorema VC]** Seja

$$V = \mathcal{F}(\mathbf{z}_1, \ldots, \mathbf{z}_n, \mathbf{z}'_1, \ldots, \mathbf{z}'_n).$$

Para $v \in V$, seja

$$\mathcal{P}_n(v) - \mathcal{P}'_n(v) = \frac{1}{n}\left(\sum_{i=1}^{n} v_i - \sum_{i=n+1}^{2n} v_i\right).$$

Segue do Lema 3 que

$$\begin{aligned}
\mathbb{P}\left(\sup_{f\in\mathcal{F}} |P_n(f) - P(f)| > \epsilon\right) &\leq 2\mathbb{P}\left(\sup_{f\in\mathcal{F}} |P_n(f) - P'_n(f)| > \epsilon/2\right) \\
&= 2\mathbb{P}\left(\sup_{v\in\mathcal{F}(\mathbf{Z}_1,\ldots,\mathbf{Z}_n,\mathbf{Z}'_1,\ldots,\mathbf{Z}'_n)} |P_n(v) - P'_n(v)| > \epsilon/2\right) \\
&\leq 2s(\mathcal{F}, 2n)\mathbb{P}(|P_n(f) - P'_n(f)| > \epsilon/2) \\
&\leq 4s(\mathcal{F}, 2n)e^{-n\epsilon^2/8},
\end{aligned}$$

em que a penúltima desigualdade segue da desigualdade da união e do fato que $\mathcal{F}_{\mathbf{z}_1,\ldots,\mathbf{z}_n,\mathbf{z}'_1,\ldots,\mathbf{z}'_n}$ possui no máximo $s(\mathcal{F}, 2n)$ elementos. Já a última desiguldade segue da desigualdade de Hoeffding. □

9.6 Resumo

Vimos que um problema de classificação difere de um problema de regressão pelo fato da variável Y ser qualitativa. Uma implicação disso é que o risco quadrático, utilizado para regressão, não é adequado para o problema de classificação. Vimos que o risco 0-1, a probabilidade de classificar uma nova observação incorretamente, é mais adequado. Contudo, essa métrica deve ser complementada com outras, como a sensibilidade e a especificidade. Também é possível adaptar o risco para que cada tipo de erro seja penalizado de forma diferente. Vimos que o classificador de Bayes é o melhor classificador segundo o risco 0-1, o que motiva o uso de classificadores *plugin*, que se baseiam em estimar a probabilidade de uma observação percentecer a uma classe dadas as covariáveis associadas a ela. Contudo, nem todos os classificadores se baseiam nisso. Finalmente, vimos ferramentas para lidar com situações de predição em que os dados não são i.i.d.

Parte III

Aprendizado não supervisionado

Capítulo 10

Redução de Dimensionalidade e Mudança de Representação

Métodos de redução de dimensionalidade buscam encontrar transformações das covariáveis originais que capturam uma parte considerável das informações presentes nelas, de modo a retirar redundâncias nas covariáveis e diminuir o número dessas. Existem diversas razões para isso, como, por exemplo, facilitar a visualização dos dados e promover a criação de funções de predição com maior poder preditivo. Veja, por exemplo, a Figura 10.1, que apresenta um diagrama de dispersão de duas variáveis criadas a partir dos pixels das imagens de dígitos escritos a mão (como na Figura 7.1) utilizando a técnica de componentes principais. Cada ponto é representado pelo dígito correspondente. Note que 9's e 7's estão bastante sobrepostos, indicando uma maior similaridade entre si.

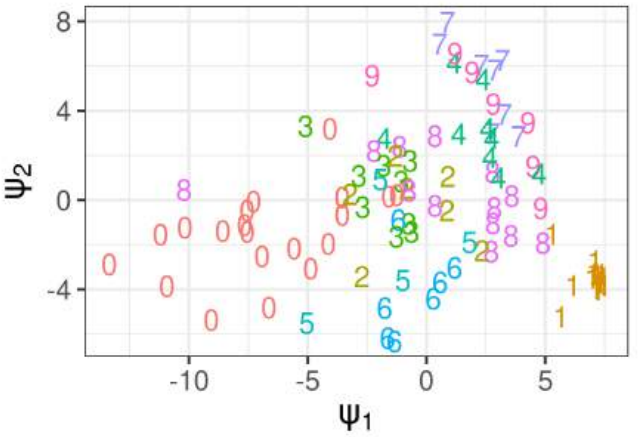


Figura 10.1: Diagrama de dispersão de duas variáveis criadas a partir dos pixels das imagens de dígitos escritos a mão (veja a Figura 7.1). Cada ponto é representado pelo dígito correspondente.

As técnicas de redução de dimensionalidade têm como objetivo a criação de um número "pequeno" de variáveis $Z_1, Z_2, \ldots$ a partir de $X_1, X_2, \ldots$ que resumam bem as informações presentes nelas. Neste capítulo estudamos diversas formas de se fazer essa redução.

De forma mais geral, veremos também como alterar a representação das covariáveis originais (isto é, fazer uma mudança de coordenadas), mesmo que isso envolva aumentar o número de variáveis. A nova representação pode ser útil para, posteriormente, utilizar técnicas de aprendizado supervisionado.

10.1 Componentes Principais (PCA - *Principal Component Analysis*)

Por simplicidade, nesta seção iremos assumir que as covariáveis estão normalizadas para ter média zero.

Na técnica de componentes principais, a informação de uma variável Z_i é medida através de sua variabilidade. Mais especificamente, ela é medida utilizando sua variância. Além disso, a redundância entre duas variáveis Z_i e Z_j é medida através de sua correlação. Finalmente, essa técnica se restringe a buscar por variáveis que sejam

combinações lineares das covariáveis originais.

Formalmente, o primeiro componente principal de $\mathbf{X}$ é a variável Z_1 que

1. é combinação linear das variáveis $\mathbf{X}$, isto é, pode ser escrito como

$$Z_1 = \phi_{11}X_1 + \ldots + \phi_{d1}X_d \tag{10.1}$$

2. tem a maior variância possível.

Assim, Z_1 é a variável que pode ser representada pela Equação 10.1, em que os coeficientes $\phi_{11},\ldots,\phi_{d1}$ são tais que Z_1 resultante possui a maior variabilidade possível. Adiciona-se também a restrição de que $\sum_{k=1}^{d}\phi_{k1}^2 = 1$, caso contrário os coeficientes ϕ_{k1} teriam valor absoluto tão grande quanto se queira e, dessa forma, poderia se aumentar a variância diretamente em função desses coeficientes.

Da mesma maneira, o segundo componente principal de $\mathbf{X}$ é a variável Z_2 que

1. é combinação linear das variáveis $\mathbf{X}$, isto é, pode ser escrito como

$$Z_2 = \phi_{12}X_1 + \ldots + \phi_{d2}X_d$$

   com a restrição: $\sum_{k=1}^{d}\phi_{k2}^2 = 1$

2. tem a maior variância possível.

3. tem correlação zero com Z_1.

Em outras palavras, o segundo componente também deve ser função linear das variáveis, deve ter a maior variância possível (para trazer o máximo de informação) e deve ter correlação zero com o componente já calculado (de modo que não haja informação redundante entre ambos).

De modo genérico, o i-ésimo componente principal de $\mathbf{X}$, $i > 1$, é a variável Z_i que

1. é combinação linear das variáveis $\mathbf{X}$, isto é, pode ser escrito como

$$Z_i = \phi_{1i}X_1 + \ldots + \phi_{di}X_d$$

   com a restrição: $\sum_{k=1}^{d}\phi_{ki}^2 = 1$

2. tem a maior variância possível.

3. tem correlação zero com $Z_1, \ldots, Z_{i-1}$.

Felizmente a solução para esse problema é simples. Ela consiste em primeiramente encontrar autovetores de $C = \mathbb{X}^\intercal \mathbb{X}$, que é matriz de variância-covariância das covariáveis (a menos de um fator multiplicativo), pois estamos assumindo que elas têm média zero. Seja U a matriz $d \times d$ em que a i-ésima coluna contém o i-ésimo autovetor de C. U é justamente a matriz de *cargas*, i.e., o seu elemento i, j é dado pelo coeficiente ótimo ϕ_{ij}. Assim, $Z = \mathbb{X}U$ é a matriz $n \times d$ com as d novas covariáveis para cada uma das n observações.

A Figura 10.2 apresenta uma ilustração dos dois primeiros componentes principais para um conjunto de dados simulados Nesse caso, o primeiro componente, z_1, representa a posição de cada ponto com relação ao eixo marcado de vermelho, que é o eixo de maior variabilidade dos dados. Note que, sabendo o valor desta posição (por exemplo, $z_1 = -1$), pode-se ter uma noção do valor tanto de x_1 quanto de x_2. Assim, de fato o primeiro componente captura o que é mais importante sobre os dados. O segundo componente, z_2 é a posição de cada ponto com relação ao eixo azul. Essa informação complementa aquela dada pelo componente z_1: com base em z_1 e z_2 é possível reconstruir x_1 e x_2. Além disso, os eixos são ortogonais entre si. Isso sempre ocorre devido ao fato que se exige que a correlação entre Z_1 e Z_2 seja zero.

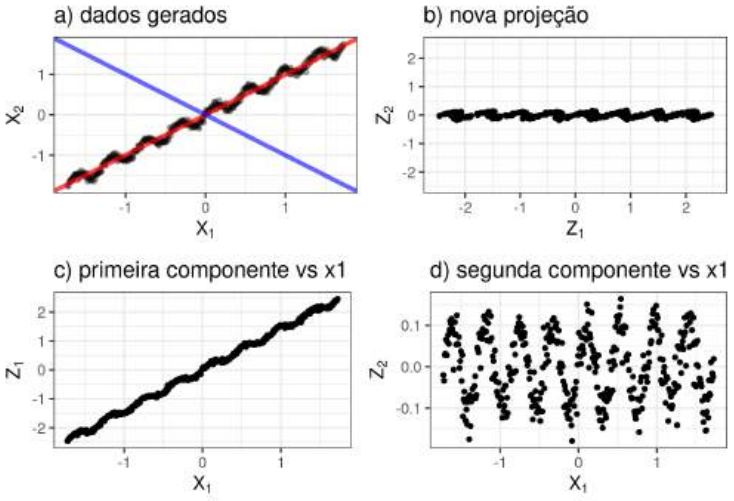


Figura 10.2: Componentes principais encontrados para um conjunto de dados simulado.

10.1.1 Interpretação alternativa: escalonamento multidimensional

Uma outra forma de motivar a utilização dos componentes principais é dada pelo escalonamento multidimensional. Suponha que deseja-se encontrar as transformações *lineares* $\mathbf{Z} = (Z_1, \ldots, Z_m)$, $m < d$ de $\mathbf{X} = (X_1, \ldots, X_d)$ tais que

$$\sum_{i,j} (||\mathbf{x}_i - \mathbf{x}_j||^2 - ||\mathbf{z}_i - \mathbf{z}_j||^2)$$

tenha o menor valor possível. Ou seja, deseja-se buscar a transformação linear das variáveis originais tal que as distâncias euclidianas entre os pares de vetores originais, $||\mathbf{x}_i - \mathbf{x}_j||$ sejam o mais próximas possíveis das distâncias euclidianas entre os pares de novos vetores, $||\mathbf{z}_i - \mathbf{z}_j||$ no sentido de minimizar

$$\sum_{i,j} ||\mathbf{x}_i - \mathbf{x}_j||^2 - ||\mathbf{z}_i - \mathbf{z}_j||^2.$$

A solução para esse problema consiste justamente em tomar $\mathbf{Z}$ como sendo o vetor dos m primeiros componentes principais de $\mathbf{x}$.

Exemplo 10.1. Considere o conjunto de dados Zip Code[1], que contém a imagem de dígitos escritos a mão.

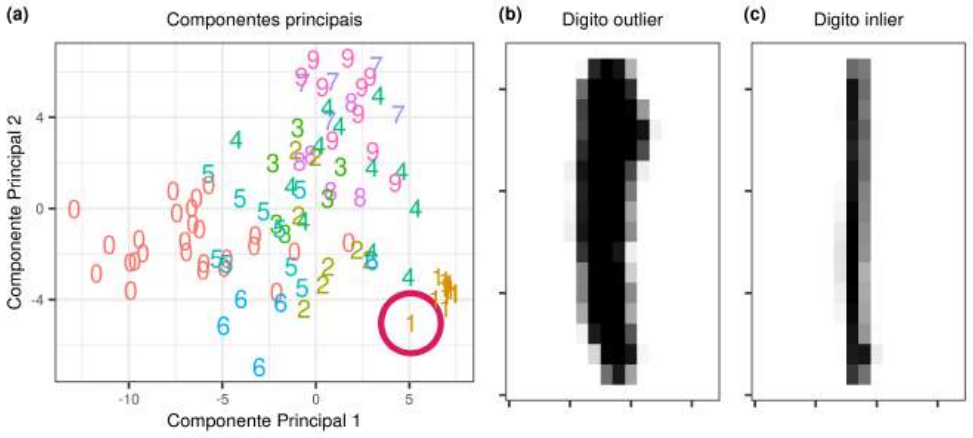


Figura 10.3: Dois primeiros componentes principais encontrados para o conjunto de dados de dígitos escritos à mão, juntamente com a imagem de um dígito *outlier* (correspondente ao ponto circulado na primeira figura) e um dígito *inlier*.

[1] http://statweb.stanford.edu/~tibs/ElemStatLearn/.

A Figura 10.3 ilustra o diagrama de dispersão dos dois primeiros componentes principais, assim como a imagem de um dígito *outlier* (correspondente ao ponto circulado na primeira figura) e um dígito *inlier*. Claramente o dígito *outlier* possui características diferentes do que é esperado para um dígito "1" escrito à mão.

□

10.1.2 Aplicação: compressão de imagens

Seja $\mathbb{X}$ a matriz $n \times d$ com os dados observados. Em PCA, fazemos a decomposição SVD da matriz de covariâncias:

$$\mathbb{X}^\intercal \mathbb{X} = U \Sigma U^\intercal,$$

em que U é uma matriz $d \times d$ cujas colunas contém os autovetores da matriz $\mathbb{X}^\intercal\mathbb{X}$ (a chamada matriz de cargas) e Σ é uma matriz diagonal com seus autovalores. Os autovetores de U são ordenados de acordo com a magnitude do respectivo autovalor de forma decrescente, ou seja, o primeiro autovetor corresponde ao maior autovalor.

A matriz $Z = \mathbb{X}^\intercal U$ (que é $n \times d$) contém os valores das d novas covariáveis avaliadas em cada uma das n observações. Além disso, os dados $\mathbb{X}$ podem ser reconstruídos calculando-se $ZU^\intercal$. Note que a i-ésima linha de $ZU^\intercal$ é dada por

$$\sum_{k=1}^{d} z_{i,k} \mathbf{u}_k,$$

em que $z_{i,k} = \mathbf{u}_k^\intercal \mathbf{x}_i$ é o valor da k-ésima nova variável para a i-ésima amostra e $\mathbf{u}_k$ é a k-ésima coluna de U. É possível mostrar que

$$\sum_{k=1}^{p} z_{i,k} \mathbf{u}_k \approx \sum_{k=1}^{d} z_{i,k} \mathbf{u}_k$$

para $p < d$, contanto que os autovalores da matriz de covariâncias decaiam rapidamente para zero.

Isso sugere aproximar $\mathbf{x}_i$ truncando $\sum_{k=1}^{p} c_{i,k} \mathbf{u}_k$ em um valor pequeno de p. A vantagem de tal aproximação é que ela reduz o espaço necessário para armazenar $\mathbf{x}_i$. Dessa forma, só é necessário armazenar o valor das p novas covariáveis $(z_{i,1}, \ldots, z_{i,p})$

e os p primeiros pesos $\mathbf{u}_1, \ldots, \mathbf{u}_p$. Matricialmente,

$$\mathbb{X} \approx Z_p U_p^\intercal,$$

em que Z_p contém as p primeiras colunas de Z e U_p contém as p primeiras colunas de U.

Nessa seção mostraremos como utilizar PCA para comprimir a seguinte imagem:

Figura 10.4: Figura que será comprimida via PCA.

A ideia central é tratar a imagem como sendo a coleção de três matrizes, uma para cada canal de cor (*red*/*green*/*blue*). Primeiramente, faremos a leitura da imagem no R. Note que o objeto *imagem* é um array $639 \times 540 \times 3$. As duas primeiras dimensões indicam o tamanho da imagem e a última dimensão indica as camadas de cor (nesse caso, RGB).

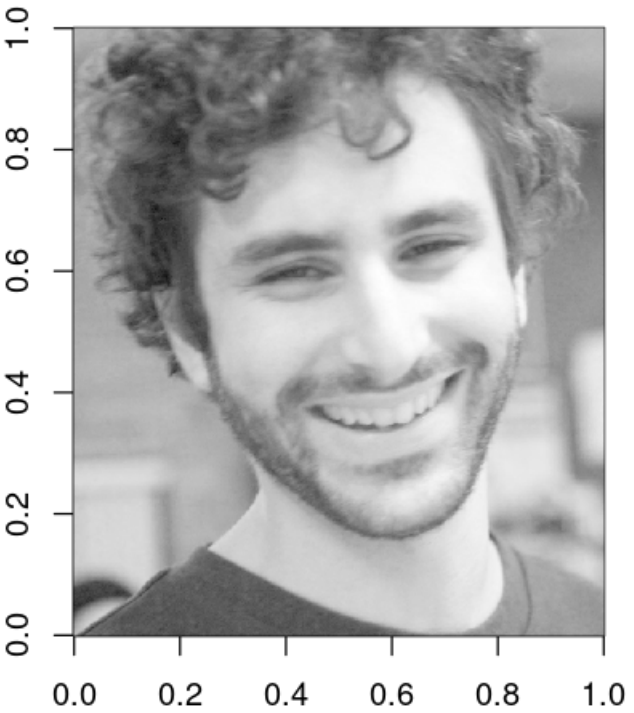
```
library(jpeg)
library(gridExtra)

imagem <- readJPEG("../Figures/izbicki.jpg", native = FALSE)

red   <- imagem[,,1]
green <- imagem[,,2]
blue  <- imagem[,,3]
```

```
rotate <- function(x) t(apply(x, 2, rev))

# canal vermelho
image(rotate(red), col = grey.colors(300, 0, 1))
```



Aplicaremos a decomposição SVD na matriz de cada canal separadamente.

```
r_svd <- svd(t(red) %*% red)
g_svd <- svd(t(green) %*% green)
b_svd <- svd(t(blue) %*% blue)
```

Com isso, iremos aplicar PCA para comprimir as imagens. Para comparar diferentes níveis de compressão, utilizaremos diferentes opções para o número de componentes utilizados.

```
ncolunas <- c(3, 5, 25, 100, 200, 300) # níveis
# de compressão
for(i in ncolunas) {
  #  (novas variaveis)
  r_projecao <- red %*% r_svd$u[, 1:i]

  #  (reconstrucao)
  r_approx <- r_projecao %*% t(r_svd$u[, 1:i])

  g_projecao <- green %*% g_svd$u[, 1:i]
  g_approx <- g_projecao %*% t(g_svd$u[, 1:i])

  b_projecao <- blue %*% b_svd$u[, 1:i]
  b_approx <- b_projecao %*% t(b_svd$u[, 1:i])

  imagemFinal <- array(NA, dim = c(nrow(red), ncol(red), 3))
  imagemFinal[,,1] <- r_approx
  imagemFinal[,,2] <- g_approx
  imagemFinal[,,3] <- b_approx

  imagemFinal <- ifelse(imagemFinal < 0, 0,
                 ifelse(imagemFinal > 1, 1, imagemFinal))

  grid.raster(imagemFinal)

  cat("Numero de componentes =", i, "\n",
      "Economia de espaço = ",
          paste0(round(100 - (prod(dim(r_projecao)) +
                  prod(dim(r_svd$u[, 1:i])))/prod(dim(red))*
                  100, 2),"%"))
}
```

(a) 3 componentes; 98,9% de compressão (b) 5 componentes; 98,3% de compressão (c) 25 componentes; 91,46% de compressão

(d) 100 componentes; 65,8% de compressão (e) 200 componentes; 31,2% de compressão (f) 300 componentes; - 2,5% de compressão

Figura 10.5: Imagens recuperadas a partir dos componentes principais

10.2 Kernel PCA (KPCA)

Se restringir a transformações lineares das variáveis originais, como feito na seção anterior, pode ser um fator muito limitante. Considere a Figura 10.6. Intuitivamente, o primeiro componente principal deveria ser capaz de distinguir os grupos vermelho e preto. No entanto, o primeiro componente, como mostrado na figura, não é capaz de fazer isso. Contudo, Kernel PCA, a técnica descrita nesta seção, é capaz de fazê-lo.

A ideia do Kernel PCA é utilizar o truque do kernel (já utilizado na Seção 4.6.3.2 no contexto de regressão) de forma a considerar transformações não lineares das covariáveis.

Lembre-se que a solução do PCA consiste em encontrar autovetores de $C = \mathbb{X}^\intercal\mathbb{X}$. O fato fundamental para usar o truque do kernel é que Z também pode ser calculado

via o cálculo dos autovetores de

$$K = \mathbb{X}\mathbb{X}^\intercal,$$

a matriz com os produtos internos entre cada par de observações. Note que essa matriz é diferente de C. Mais especificamente, tem-se que

$$Z = U_2^\intercal,$$

em que U_2 é a matriz de autovetores de K. Assim, para calcular as componentes principais, basta saber os produtos internos entre as observações.

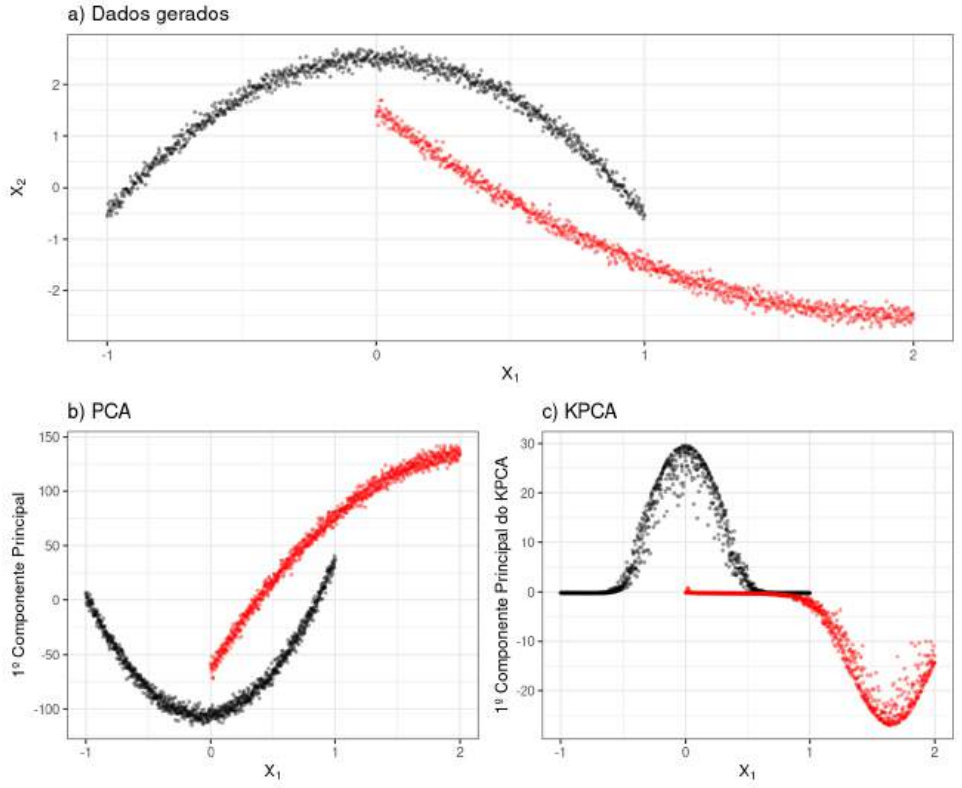


Figura 10.6: Primeiro componente da PCA e do KPCA encontrados para um conjunto de dados simulado.

O truque do kernel na análise de componentes principais consiste em utilizar

outros produtos internos (kernels) ao invés de $\langle \mathbf{x}_i, \mathbf{x}_j \rangle = \sum_{k=1}^{d} x_{i,k} x_{j,k}$. Ou seja, ao invés de calcular a autodecomposição de $K = \mathbb{X}\mathbb{X}^\intercal$, utilizamos outros kernels,

$$K = \begin{bmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & K(\mathbf{x}_1, \mathbf{x}_2) & \cdots & K(\mathbf{x}_1, \mathbf{x}_n) \\ K(\mathbf{x}_2, \mathbf{x}_1) & K(\mathbf{x}_2, \mathbf{x}_2) & \cdots & K(\mathbf{x}_2, \mathbf{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ K(\mathbf{x}_n, \mathbf{x}_1) & K(\mathbf{x}_n, \mathbf{x}_2) & \cdots & K(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix},$$

em que $K(\mathbf{x}_i, \mathbf{x}_j)$ corresponde ao produto interno de alguma transformação das covariáveis $\mathbf{x}$. Lembre-se que o truque consiste no fato de que a transformação não precisa ser calculada analiticamente (Seção 4.6.3.2).

No R, o KPCA (e, em particular, PCA) pode ser ajustado utilizando o pacote `kernlab`.

```
library(kernlab)
# kernel polinomial (note que degree=1 corresponde a
# PCA usual):
kpc <- kpca(dados, kernel = "polydot",
            kpar = list(degree = 2), features = 2)

# ou kernel gaussiano (sigma = 1/variancia):
kpc <- kpca(dados, kernel = "rbfdot",
            kpar = list(sigma = 1), features = 2)

variaveis_novos_dadoss <- predict(kpc, novos_dados)
```

Para kernels diferentes do kernel linear, o número de componentes criados pode ser maior que d. Assim, essa técnica não necessariamente faz uma redução de dimensão; ela cria um novo sistema de coordenadas para os dados.

Observação 10.1. KPCA também pode ser motivada via escalonamento multidimensional: fazer KPCA equivale a fazer escalonamento multidimensional com base na distância $d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{2(1 - K(\mathbf{x}_i, \mathbf{x}_j))}$.

□

Na Seção 11.2 estudaremos outra técnica que também utilizar um kernel de Mercer para criar uma nova representação dos dados.

10.3 Projeções Aleatórias

Uma estratégia alternativa para reduzir a dimensionalidade dos dados é o método de projeções aleatórias. Nele, cada componente z_i é criado de modo a ser uma projeção linear das covariáveis originais com coeficientes escolhidos aleatoriamente. Mais especificamente, seja S uma matriz com m linhas e d colunas cujas entradas são amostras i.i.d. de normais com média zero e variância um. Seja $s_{i,j}$ a entrada (i,j) desta matriz. Definimos a k-ésima projeção aleatória de $\mathbf{x}$ como

$$z_i = \sum_{k=1}^{d} \frac{s_{i,k}}{\sqrt{m}} x_k.$$

Seja $\mathbf{z}_i = (z_{i,1}, \ldots, z_{i,m})$ o vetor composto pelas novas variáveis. O teorema a seguir mostra que, com probabilidade alta, tal transformação preserva distâncias, i.e., $||\mathbf{x}_i - \mathbf{x}_j|| \approx ||\mathbf{z}_i - \mathbf{z}_j||$ para todos índices i e j.

Teorema 14 (Johnson-Lindenstrauss, (Johnson e Lindenstrauss, 1984)). *Fixe $\epsilon > 0$ e seja $m \geq 32 \log n / \epsilon^2$. Então, com probabilidade ao menos $1 - e^{-m\epsilon^2/16}$, vale que*

$$(1-\epsilon)||\mathbf{x}_i - \mathbf{x}_j||^2 \leq ||\mathbf{z}_i - \mathbf{z}_j||^2 \leq (1+\epsilon)||\mathbf{x}_i - \mathbf{x}_j||^2$$

para todos $i, j = 1, \ldots, n$.

Exemplo 10.2. Considere novamente o conjunto de dados Zip Code (Exemplo 10.1). As Figuras 10.7 e 10.8 ilustram as duas e três primeiras projeções aleatórias calculadas. Note como imagens referentes aos mesmos dígitos tendem a se agrupar.

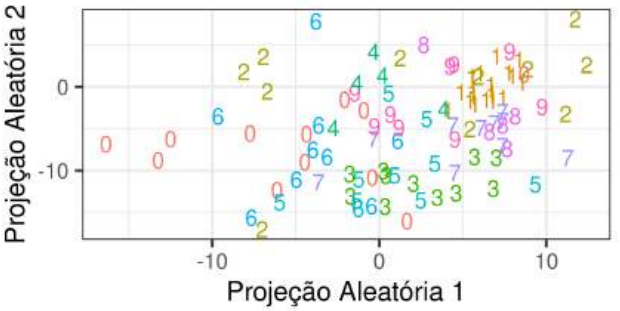


Figura 10.7: Duas primeiras projeções aleatórias calculadas no conjunto de dados de dígitos escritos à mão.

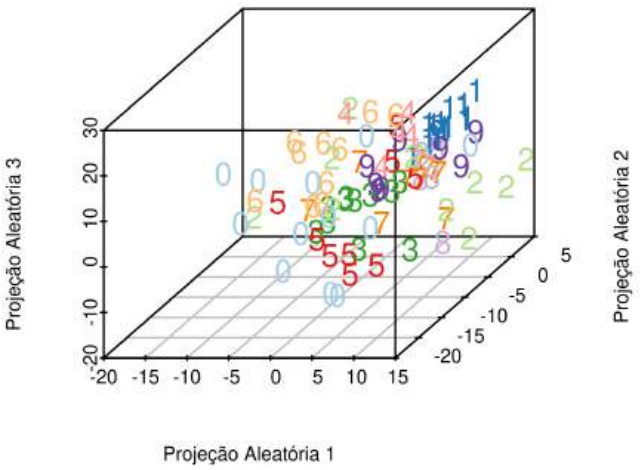


Figura 10.8: Três primeiras projeções aleatórias calculadas no conjunto de dados de dígitos escritos à mão.

□

Além das aplicações de técnicas de redução de dimensionalidade já discutidas, projeções aleatórias também podem ser utilizadas como forma de deixar diversos algoritmos mais rápidos. Isso ocorre pois esse método preserva distâncias (e produtos internos) e é muito rápido de ser calculado. Isso permite que métodos de regressão e classificação, que exigem apenas o cálculo do produto interno para serem implementados (como a regressão ridge e support vector machines), possam ser aproximados sem grandes perdas de desempenho estatístico.

10.4 Autoencoders

Redes neurais podem ser utilizadas para reduzir a dimensão dos dados. Uma maneira de fazer isso é através de autoeconders, que são redes com camada de saída

com o mesmo número de neurônios que a camada de entrada (Figura 10.9).

Em um autoencoder, a função de perda é dada por

$$\frac{1}{n}\sum_{k=1}^{n}\sum_{i=1}^{d}\left(\mathbf{x}_{k,i} - g_{\beta;i}(\mathbf{x}_k)\right)^2,$$

que incentiva que a saída da rede seja a mesma que a da entrada: como a camada oculta tem menos neurônios que a camada de entrada, a saída deve recuperar a entrada a partir de um número menor de variáveis. Assim, essa estrutura exige que as camadas ocultas consigam capturar o máximo de informação sobre $\mathbf{x}$ em um número pequeno de variáveis.

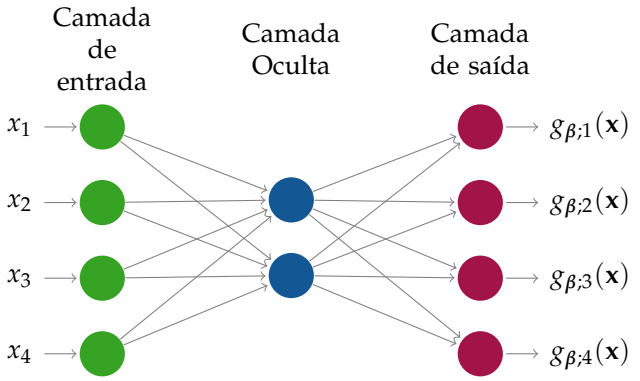


Figura 10.9: Exemplo de autoencoder.

Quando há apenas uma camada oculta, esse processo leva essencialmente aos mesmos resultados que PCA (Seção 10.1). Autoenconders, contudo, podem ter uma estrutura muito mais geral, com várias camadas ocultas e outras arquiteturas.

Exemplo 10.3. Nesse exemplo apresentamos dados processados obtidos a partir da página de dados estatísticos da Secretaria de Segurança Pública de São Paulo (www.ssp.sp.gov.br/Estatistica) para o número de ocorrências registradas no ano de 2019. Utilizaremos os seguintes indicadores:

- homicídio doloso
- nº de vítimas em homicídio doloso
- homicídio doloso por acidente de trânsito

- nº de vítimas em homicídio doloso por acidente de trânsito
- homicídio culposo por acidente de trânsito
- homicídio culposo outros
- tentativa de homicídio
- lesão corporal seguida de morte
- lesão corporal dolosa
- lesão corporal culposa por acidente de trânsito
- lesão corporal culposa - outras
- latrocínio
- nº de vítimas em latrocínio
- total de estupro
- estupro
- estupro de vulnerável
- total de roubo - outros
- roubo - outros
- roubo de veículo
- roubo a banco
- roubo de carga
- furto - outros
- furto de veículo

Utilizaremos uma rede com 23 unidades de entrada (o número de indicadores/covariáveis), três unidades na camada oculta e 23 unidades na camada de saída. Assim, o erro observado será pequeno se não houver grande perda de informação após a redução de dimensão dos dados com três unidades na camada oculta.

Para avaliar o desempenho da rede neural, faremos a reconstrução dos dados de diversos municípios separados em um conjunto de teste e avaliaremos a diferença absoluta de cada indicador. Essa diferença pode ser observada na Figura 10.10.

```
library(viridis)
library(tidyverse)
library(tidymodels)
library(keras)

dados <- read_csv2("../dados/indicadores_ssp.csv")

# processamento dos dados --------------------------------

receita_prep <- dados %>%
```

```
  select(-(municipio_cod:pop_19)) %>%
  recipe(~ .) %>%
  step_normalize(all_numeric()) %>%
  prep()

# treinamento e teste ------------------------------------
set.seed(314)

splits <- juice(receita_prep) %>%
            initial_split(0.8)

treinamento <- training(splits) %>% as.matrix()
teste <- testing(splits) %>% as.matrix()

network <- keras_model_sequential() %>%
  layer_dense(units = 3, activation = "relu",
              input_shape = 23) %>%
  layer_dense(units = 23)

network %>%
  compile(optimizer = "adam",
          loss = "mean_squared_error")

network %>%
  fit(treinamento, treinamento,
      epochs = 40,
      batch_size = 16,
      validation_split = 0.2)

preditos <- predict(network, teste)
diferenca <- data.frame(abs(teste - preditos))

tibble(id = 1:nrow(diferenca)) %>%
  bind_cols(diferenca) %>%
  pivot_longer(-id, names_to = "indicador",
```

```
              values_to = "diferenca") %>%
  ggplot(aes(id, indicador, fill = diferenca)) +
  geom_tile() +
  labs(x = "município", y = "",
       fill = "diferença entre observado e predito") +
  scale_fill_viridis(direction = -1) +
  scale_x_continuous(expand = c(0, 0)) +
  theme_minimal() +
  theme(legend.position = "top",
        text = element_text(size = 8))
```

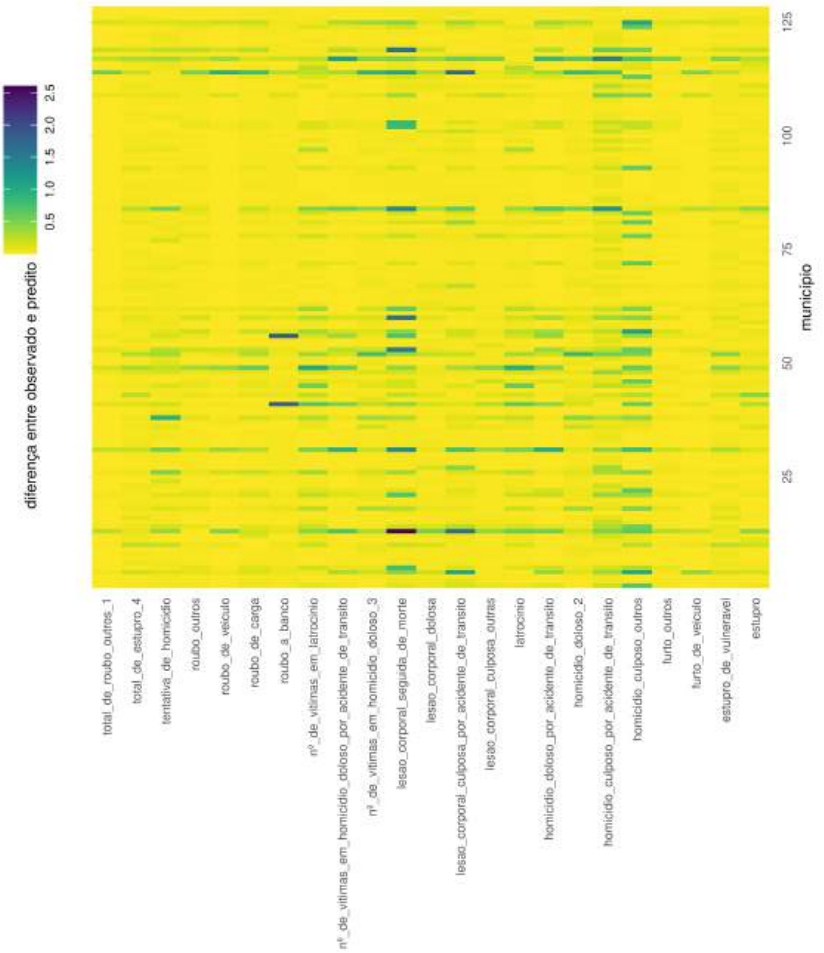


Figura 10.10: Erro de reconstrução da compressão dada pelo autoencoder em observações de um conjunto de teste.

Capítulo 11

Análise de Agrupamento

Métodos de análise de agrupamento (análise de cluster) têm por finalidade dividir a amostra em grupos. Essa divisão é feita de modo que os grupos sejam diferentes uns dos outros, mas os elementos pertencentes ao mesmo grupo sejam parecidos entre si. Esses métodos podem ser utilizados com diversas finalidades. Por exemplo, uma loja de departamento pode utilizar análise de agrupamento para segmentar seus clientes. Assim, pode decidir como deve agir em relação a cada grupo. Por exemplo, a empresa poderá enviar um tipo diferente de publicidade para cada cliente de acordo com o perfil do grupo que pertence. Outro exemplo são as resenhas de usuários mostradas em sites de compras online quando um indivíduo busca por um produto. Em geral, a empresa escolhe automaticamente duas ou três resenhas bastante distintas, mas que ao mesmo tempo são representativas das milhares de resenhas feitas sobre esse produto.

Formalmente, o objetivo de um método de clustering é criar uma partição $C_1, \ldots, C_K$ dos elementos amostrais $\{1, \ldots, n\}$. Isto é, devemos ter ao mesmo tempo

$$C_1 \bigcup C_2 \bigcup \ldots \bigcup C_K = \{1, 2 \ldots, n\}$$

e

$$C_i \bigcap C_j = \emptyset \; \forall i \neq j.$$

Neste capítulo apresentamos diversos métodos de análise de agrupamento com diferentes heurísticas para criar tais partições.

Um conceito essencial a diversos métodos de agrupamento é como medir dissi-

milaridade (ou similaridade) entre dois elementos com covariáveis $\mathbf{x}_i$ e $\mathbf{x}_j$. Existem várias medidas possíveis. A seguir, listamos alguns exemplos:

- **distância Euclidiana:** $d^2(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{d}(x_{i,k} - x_{j,k})^2$;
- **distância de Mahalanobis:** $d(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{d}|x_{i,k} - x_{j,k}|$;
- **distância cosseno:** $d(\mathbf{x}_i, \mathbf{x}_j) = 1 - \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{||\mathbf{x}_i||\,||\mathbf{x}_j||}$;
- **distância de Jaccard:** $d(\mathbf{x}_i, \mathbf{x}_j) = 1 - \frac{\sum_{k=1}^{d} x_{i,k}x_{j,k}}{\sum_{k=1}^{d} x_{i,k} + \sum_{k=1}^{d} x_{j,k} - \sum_{k=1}^{d} x_{i,k}x_{j,k}}$, em que $\mathbf{x}$ assume valores 0 e 1.

Cada distância apresentada é mais adequada em um tipo de contexto. Por exemplo, a distância cosseno é muito utilizada em aplicações de texto. Para aplicações de exemplos reais com a distância de Jaccard, veja Fernandez e F. (2020).

Na próxima seção apresentaremos um método em que é necessário fixar previamente o número de clusters.

11.1 K-Médias

O método das K-Médias assume que a medida de dissimilaridade usada é a distância Euclidiana. Para utilizá-lo, é necessário especificar de antemão o valor de K, quantos clusters se deseja. Nesse algoritmo, buscar o melhor clustering é entendido como buscar pela partição $C_1, \ldots, C_K$ da amostra tal que

$$\sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,j \in C_k} d^2(\mathbf{x}_i, \mathbf{x}_j)$$

seja o menor possível. Essa quantidade é a soma de quadrados dentro de cada cluster.

Como há K^N partições dos dados, não é possível minimizar essa função objetivo de forma exata. Um método para aproximar a solução desse problema é um algoritmo iterativo chamado Algoritmo de Lloyd. Embora não haja garantias de que a melhor solução será de fato encontrada, esse método, frequentemente, leva a bons resultados. O algoritmo consiste nos seguintes passos:

1. Escolha aleatoriamente k centróides $c_1, \ldots, c_k$.

Itere até obter convergência:

2. **Atribuição**: Defina o cluster C_j ($j = 1, \ldots, k$) como sendo

$$C_j = \{x_i : \arg\min_r d(x_i, c_r) = r\}$$

3. **Atualização**: Calcule os novos centroides utilizando os grupos que foram criados:

$$c_j \longleftarrow \frac{1}{|C_j|} \sum_{j:x_j \in C_j} x_j$$

A Figura 11.1 ilustra o algoritmo de Lloyd. Nela, os centróides são denotados por círculos vazios, enquanto que dados observados têm cor sólida. Além disso, o centróide atribuido a cada ponto é denotado por vermelho ou azul. Neste caso, o algoritmo convergiu em apenas quatro interações.

O algoritmo de Lloyd depende de escolhas iniciais feitas no passo 1. Cada escolha pode levar a um mínimo (local) diferente; veja os exemplos da Figura 11.2. Em ambos os casos o algoritmo convergiu para um mínimo que não é global.

Uma forma de contornar esse problema é utlizar o algoritmo k-médias++ (Arthur e Vassilvitskii, 2006), que é uma variação do algoritmo de Lloyd. Essa variação consiste em alterar a escolha dos pontos iniciais de forma a aumentar a chance de que o mínimo global seja encontrado. Para tanto, no k-médias++ escolhe-se C, o conjunto de pontos iniciais, da seguinte forma:

1. Escolha c_1 aleatoriamente entre $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ e defina $C = \{c_1\}$.

2. Para $j = 2, \ldots, k$:

   (a) Calcule $D(\mathbf{x}_i) = \min_{c \in C} ||\mathbf{x}_i - c||$ para cada $\mathbf{x}_i$

   (b) Escolha uma amostra $\mathbf{x}_i$ aleatoriamente entre todas as amostras observadas com probabilidade

$$p_i = \frac{D^2(\mathbf{x}_i)}{\sum_{j=1}^{n} D^2(\mathbf{x}_j)}$$

   (c) Defina c_j como sendo o ponto escolhido. Atualize

$$C \leftarrow C \cup \{c_j\}$$

A partir desses passos, o kmédias++ prossegue de forma similar ao *K*-médias.

No R, *K*-médias pode ser aplicado com a função `kmeans`. Já o *K*-médias++ pode ser utilizado via a função `kmeanspp` do pacote `LICORS`.

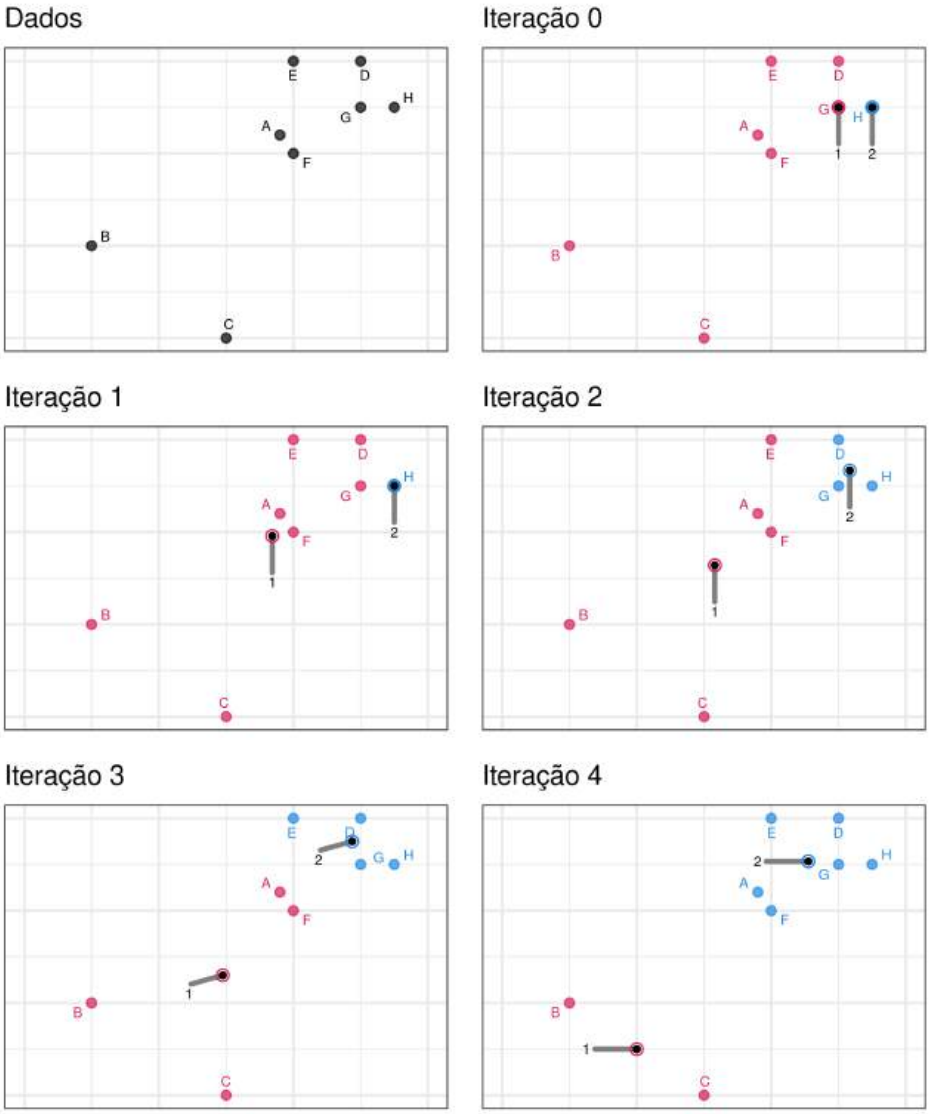


Figura 11.1: Algoritmo de Lloyd.

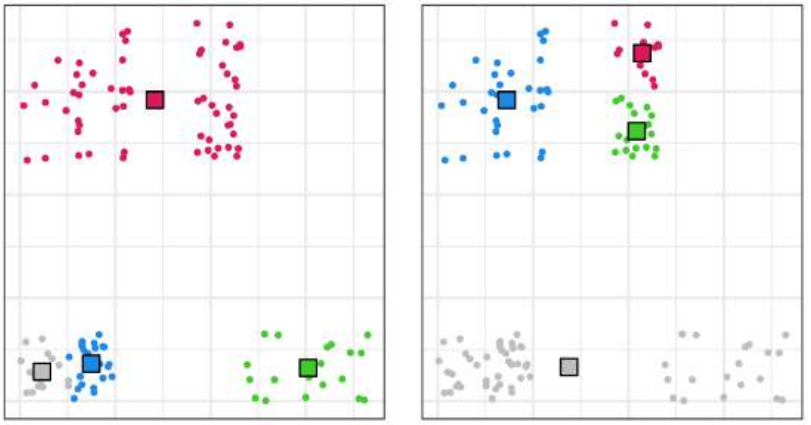


Figura 11.2: Exemplos em que o *K*-médias com Algoritmo de Lloyd não converge para o mínimo global.

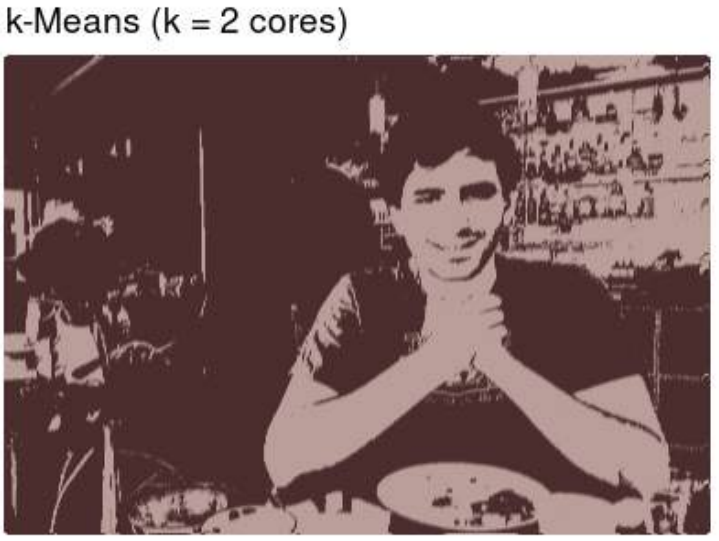
Exemplo 11.1. Neste exemplo iremos utilizar o *K*-médias para fazer um agrupamento de pixels de uma imagem. O agrupamento será feito de acordo com as cores do pixel. Cada pixel passará então a ser representado pelo centróide do cluster. Assim, o número de clusters será o número de cores de cada imagem.

```
library(jpeg)
imagem <- readJPEG("../Figures/izbicki2.jpg")
# organiza a imagem em data frame
imagemRGB <- tibble(
  x = rep(1:dim(imagem)[2], each = dim(imagem)[1]),
  y = rep(dim(imagem)[1]:1, dim(imagem)[2]),
  R = as.vector(imagem[,, 1]),
  G = as.vector(imagem[,, 2]),
  B = as.vector(imagem[,, 3]))

k <- 2   # especifique o valor de k
kMeans <- kmeans(imagemRGB[, c("R", "G", "B")], centers = k)

imagemRGB %>%
```

```
  mutate(kColours = rgb(kMeans$centers[kMeans$cluster,])) %>%
  ggplot(aes(x = x, y = y, color = I(kColours))) +
  geom_point(show.legend = FALSE) +
  labs(title = paste("k-Means (k =", k, "cores)")) +
  theme_minimal()
```



k-Means (k = 10 cores)

k-Means (k = 200 cores)

□

11.2 Agrupamento Espectral

Métodos de agrupamento espectral são uma classe de algorítmos que fazem agrupamentos a partir de autovetores. Nessa seção introduziremos um método específico

de agrupamento espectral, os mapas de difusão (Coifman e Lafon, 2006; Nadler et al., 2006).

Seja K um kernel de Mercer (Definição 1) e $\mathbb{K}$ a matriz de Gram associada (Equação 4.11). A ideia chave de mapas de difusão é construir um passeio aleatório nos dados a partir do qual é definida uma métrica entre as observações do banco. Veremos que, em algumas situações, essa distância captura melhor a conectividade entre esses pontos que a distância euclidiana.

O passeio aleatório utilizado é dado por uma cadeia de Markov no espaço de estados $\mathcal{A} := \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ que tem matriz de transição

$$\mathbb{M} := \mathbb{D}^{-1}\mathbb{K},$$

em que $\mathbb{D}$ é a matriz diagonal cuja entrada (i,i) é dada por $\sum_{j=1}^{n} K(\mathbf{x}_i, \mathbf{x}_j)$. Seja $p_t(\mathbf{x}_i|\mathbf{x}_j)$ a probabilidade de o passeio visitar o estado $\mathbf{x}_i$ daqui a t passos dado que ele está atualmente no estado $\mathbf{x}_j$. A medida estacionária dessa cadeia é dada por

$$s(\mathbf{x}_i) := \frac{\sum_{j=1}^{n} K(\mathbf{x}_i, \mathbf{x}_j)}{\sum_{i=1}^{n}\sum_{j=1}^{n} K(\mathbf{x}_i, \mathbf{x}_j)}.$$

A *distância de difusão no tempo t* entre $\mathbf{x}_i$ e $\mathbf{x}_j$ é dada por

$$D_t(\mathbf{x}_i, \mathbf{x}_j) = \sum_{\mathbf{a}\in\mathcal{A}} \frac{\left(p_t(\mathbf{a}|\mathbf{x}_i) - p_t(\mathbf{a}|\mathbf{x}_j)\right)^2}{s(\mathbf{a})}. \tag{11.1}$$

Dois pontos serão próximos segundo D_t quando houver muitos pontos que são facilmente acessíveis saindo de $\mathbf{x}_i$ ou de $\mathbf{x}_j$. Ou seja, $\mathbf{x}_i$ e $\mathbf{x}_j$ são próximos se a cadeia consegue facilmente transitar entre esses pontos em um dado tempo t. Quanto maior t, mais próximos os pontos se tornam.

A Figura 11.3 mostra como a distância de difusão difere da distância euclidiana em dois exemplos em que os dados foram simulados de forma a terem como domínio dois círculos concêntricos. Em ambos os casos, mostramos os 75 pontos mais próximos ao ponto marcado com um círculo preto. A distância de difusão tende a considerar como próximos pontos que estão no mesmo círculo, enquanto a distância euclidiana não leva em conta a geometria local.

Uma forma de usar mapas de difusão para fazer agrupamento é utilizando o algorítmo k-médias. Para isso, inicialmente, transformaremos os dados para um novo

espaço em que a distância euclidiana corresponde à distância de difusão no espaço original, de modo que o k-médias possa ser implementado.

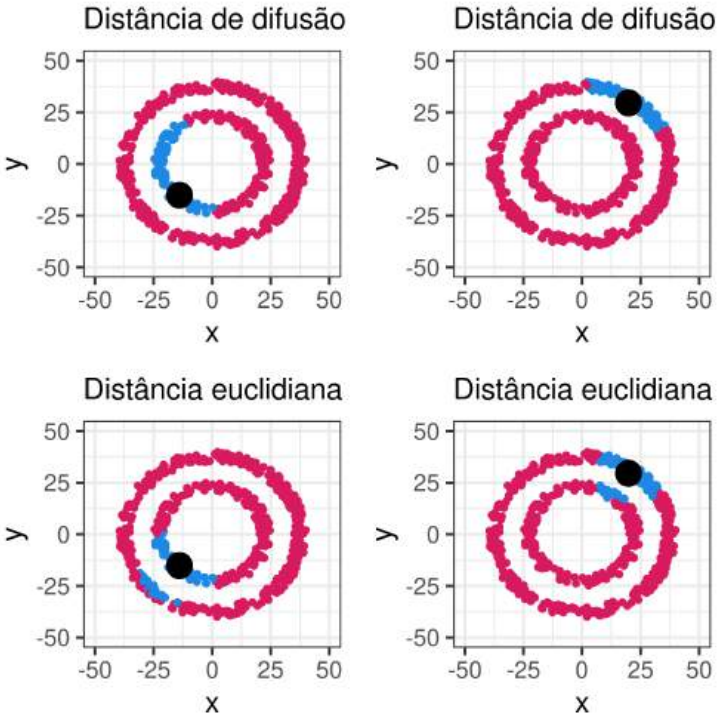


Figura 11.3: Em azul são mostradas as 75 observações mais próximas ao ponto preto segundo a distância de difusão (acima) e segundo a distância euclidiana (abaixo) .

Para isso, denote por $\lambda_0, \ldots, \lambda_n$ os autovalores de $\mathbb{M}$ ordenados de forma que $\lambda_0 \geq \ldots \geq \lambda_n$. Sejam $\psi_0, \ldots, \psi_n \in \mathbb{R}^n$ os autovetores à direta correspondentes. Pode-se mostrar que

$$D_t^2(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{n} \lambda_k^{2t} \left(\psi_k(\mathbf{x}_i) - \psi_k(\mathbf{x}_j)\right)^2 = ||\Psi(\mathbf{x}_i) - \Psi(\mathbf{x}_j)||^2, \tag{11.2}$$

em que

$$\Psi(\mathbf{x}_i) := (\lambda_1^t \psi_1(\mathbf{x}_i), \ldots, \lambda_n^t \psi_n(\mathbf{x}_i)).$$

A transformação Ψ é chamada de *mapa (ou transformação) de difusão*. Nesse novo

espaço, a distância euclidiana pode ser mais apropriada que no espaço original. De fato, a Equação 11.2 mostra que ela corresponde à distância de difusão no espaço original.

O mapa de difusão também pode ser utilizado como uma técnica de mudança de representação dos dados, como estudado no Capítulo 10. De fato, o Teorema de Eckart-Young-Mirsky mostra que os primeiros componentes do mapa de difusão são os mais importantes para aproximar a distância de difusão:

$$D_t^2(\mathbf{x}_i, \mathbf{x}_j) \approx \sum_{k=1}^{K} \lambda_k^{2t} \left(\psi_k(\mathbf{x}_i) - \psi_k(\mathbf{x}_j)\right)^2.$$

Isso motiva, por exemplo, a utilização das primeiras coordenadas desse mapa para fazer visualização de dados.

Para ajustar mapas de difusão no R, podemos usar o seguinte código:

```
# distâncias Euclidianas:
distances <- fields::rdist(dados,dados)
# tamanho da banda
eps <- quantile(distances^2, probs = 0.02)
# kernel (gaussiano):
L <- exp(-distances^2/eps)
D_inverse <- diag(1/rowSums(L))
# matrix de Markov
M <- D_inverse %*% L

# autovetores da direita:
eigen_decomp <- eigen(M)
right_eigen_vectors <- Re(eigen_decomp$vectors)

# autovetores da esquerda:
eigen_decomp <- eigen(t(M))
left_eigen_vectors <- Re(eigen_decomp$vectors)

# autovalores
values <- Re(eigen_decomp$values)
```

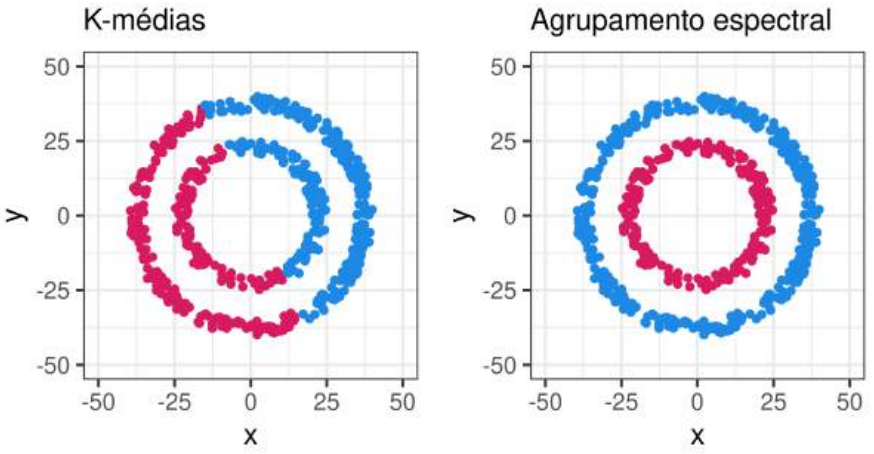


Figura 11.4: Clusters encontrados pelo k-médias (esquerda) e pelo agrupamento espectral via mapas de difusão (direita).

```
# calcular o mapa (transformação) de difusão:
mapa <- sweep(right_eigen_vectors, MARGIN = 2, values^50, '*')
distancias_difusao <- fields::rdist(mapa, mapa)
```

A Figura 11.4 mostra os clusters encontrados pelo k-médias usual e pelo k-médias aplicado no mapa de difusão. Esses agrupamentos foram obtidos via

```
library(LICORS)

cl <- kmeanspp(data = dados, k = 2) # K-médias usual
cl_diff <- kmeanspp(data = mapa, k = 2) # K-médias espectral
```

11.3 Métodos Hierárquicos

Um problema do método K-médias é que K deve ser especificado de antemão. Métodos hierárquicos são uma alternativa ao K-médias que evitam isso. Esses métodos se baseiam na aplicação do seguinte algoritmo (ou uma variação dele):

1. Atribua cada observação a um cluster diferente. Calcule cada uma das $\binom{n}{2}$ distâncias entre esses clusters.

2. Para $i = n, n-1, \ldots, 2$:

   (a) Procure entre todos os pares formados por dois dos i clusters aqueles mais parecidos. Junte esses dois clusters em um só. A dissimilaridade entre esses dois clusters indica a altura do dendrograma em que a junção será feita.

   (b) Calcule cada uma das distâncias entre os novos $i-1$ clusters.

A Figura 11.5 ilustra esse procedimento passo a passo para um conjunto de dados simulado.

Para fazer o agrupamento é necessário definir a distância entre dois *clusters*. Há várias formas de se definir essas distâncias chamadas de *linkages*. Algumas formas são:

- **Complete:** a maior das distâncias entre todos os pares de observações pertencentes aos dois clusters.
- **Single:** a menor das distâncias entre todos os pares de observações pertencentes aos dois clusters.
- **Average:** a média das distâncias entre todos os pares de observações pertencentes aos dois clusters.
- **Centroid:** a distância entre os centroides dos dois clusters.

No R, clustering hierárquico pode ser ajustado com a função `hclust`, os agrupamentos obtidos com a função `cutreet` e a visualização com a função `plot`.

```
dendrograma <- hclust(dist(dados), method = "ave")

plot(dendrograma)

# particionar os dados em 3 grupos:
clusters <- cutree(dendrograma, k = 3)
```

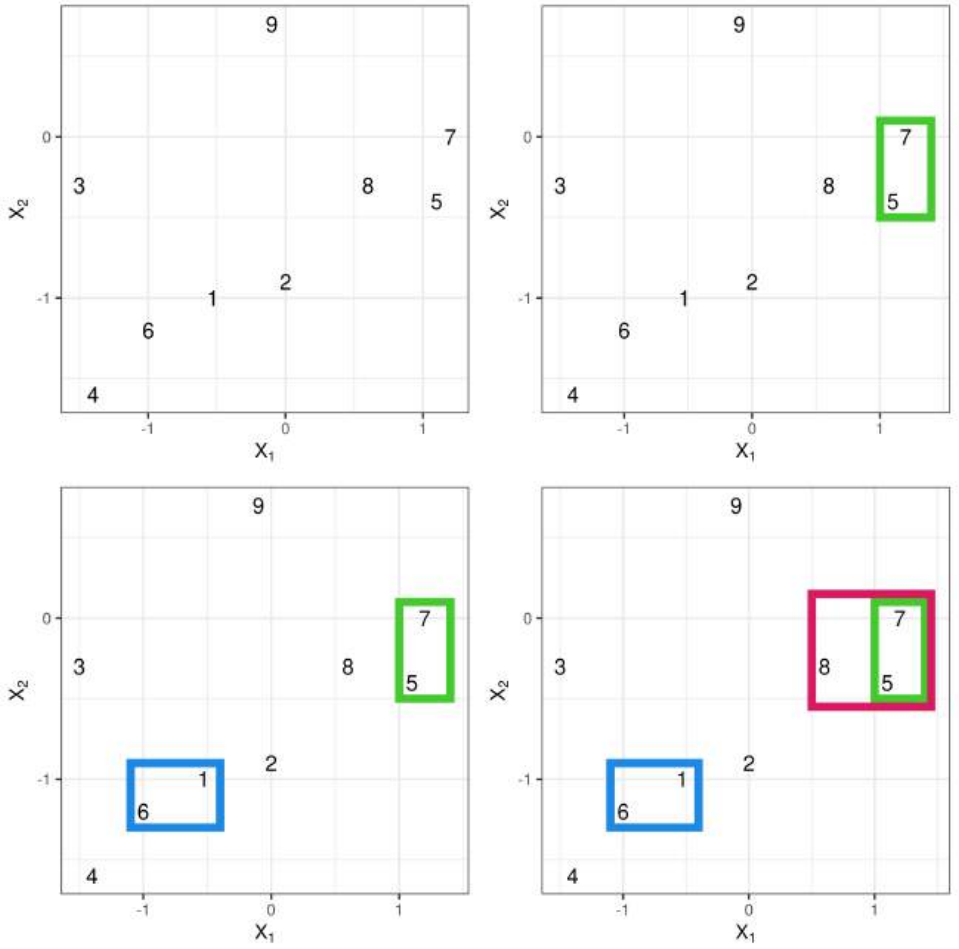


Figura 11.5: Exemplificação da construção do clustering hierárquico.

Uma alternativa para visualização pode ser obtida com a função `fviz_dend` do pacote factoextra (Kassambara e Mundt, 2020). Esse pacote é uma excelente opção para rotinas relacionadas a agrupamento, estatística multivariada e visualização, como mostra a Figura 11.6.

```
fviz_dend(dendrograma, k = 3)
```

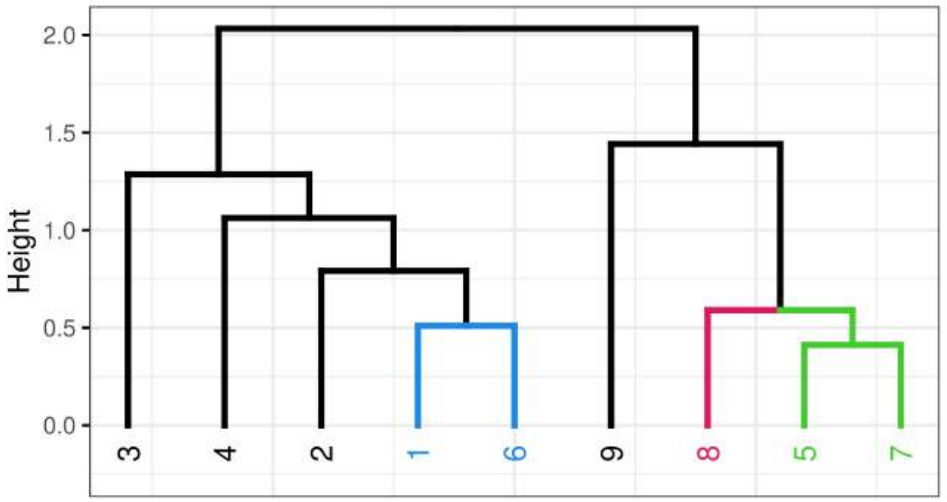


Figura 11.6: Dendrograma obtido a partir do processo acima.

Capítulo 12

Regras de Associação

Imagine que temos um banco de dados em que cada linha representa a ida de uma pessoa a um supermercado e cada coluna indica se ela comprou ou não determinado produto, como mostra a Tabela 12.1.

Tabela 12.1: Exemplo de banco de dados utilizado para criar regras de associação.

	Produto			
Compra	**Leite**	**Pão**	**Cerveja**	**Refrigerante**
1	0	1	1	0
2	1	1	0	0
3	0	0	1	1

O objetivo de regras de associação (*market basket*) é descobrir regras que ocorram com alta frequência no banco de dados. Por exemplo,

- "Quem compra leite em geral também compra pão”
- "Quem compra cerveja e refrigerante em geral também compra carne”.

Com base nessas regras, o mercado pode, por exemplo, decidir a organização dos produtos de forma a maximizar suas vendas ou melhorar a experiência de seus clientes. Outro exemplo de aplicação de regras de associação se dá no contexto de sites de vendas. Os administradores dos sites podem decidir, com base nessas regras, quais produtos oferecer a um usuário com base no histórico de compras.

Note que, ainda que a tarefa abordada por regras de associação pareça simples (a princípio basta investigar diversas tabelas de contingência), a chave do problema

.

é como fazer isso de forma eficiente. Os bancos de dados obtidos nessas aplicações, geralmente, contém muitas covariáveis e, consequentemente, o número de tabelas a serem investigadas é extremamente grande.

Nesta seção, vamos assumir que cada variável X_i do banco é binária (indicadora do produto i ter sido comprado). A fim de tornar o problema de regras de associação factível, nos restringiremos a regras que envolvam subconjuntos S das d variáveis (por exemplo, $S = \{2, 4, 10\}$) tais que

$$\mathbb{P}\left(\bigcap_{i \in S} X_i = 1\right) \tag{12.1}$$

é alto. Assim, nos restringiremos a regras que envolvam combinações de produtos que são frequentemente comprados.

A probabilidade da Eq. 12.1 é chamada de suporte do conjunto de itens S e é estimada utilizando-se $\frac{1}{n}\sum_{k=1}^{n} \mathbb{I}(X_{k,i} = 1 \ \forall i \in S)$. Busca-se, portanto, entre todos os subconjuntos de itens S, aqueles que tenham suporte maior ou igual a um corte predefinido t.

Para tornar a busca por esses conjuntos eficiente, pode-se utilizar o algoritmo *Apriori* (Agrawal et al., 1993). Para tanto, esse algoritmo explora o fato de que se $S_0 \subseteq S_1$, então o suporte de S_1 é necessariamente menor ou igual ao suporte de S_0. Isso limita o número de subconjuntos que devem ser investigados.

Após encontrar todos os subconjuntos S com suporte alto, o algoritmo *Apriori* busca por regras do tipo

$$A \Rightarrow B,$$

em que A é chamado de antecedente e B de consequente. Dada uma regra deste tipo, definem-se duas estatísticas que são úteis para descobrir regras de associação interessantes:

- **Confiança:** uma estimativa de $P(B|A)$ (entre os usuários que compraram A, quantos compraram B?)
- **Lift/Levantamento:** Uma estimativa de $\frac{P(B|A)}{P(B)}$ (quanto o usuário ter comprado A aumenta a probabilidade dele comprar B?)

Assim, buscamos, entre todas as regras que têm suporte maior que t prefixado, aquelas que têm levantamento ou confiança alta. Essa busca é factível, pois, em geral, há poucas regras com suporte alto.

Regras de associação podem ser implementadas no R com o pacote `arules`.

```
library(arules)
library(arulesViz)

# carregar o conjunto de dadis
data(Groceries)

# encontrar Regras com suporte ao menos 0.005 e confianca 0.5:
# obs: maxlen = 3, no máximo 3 itens
regras <- apriori(Groceries,
                  parameter = list(supp = 0.005,
                                   conf = 0.5, maxlen = 3))

## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.5    0.1    1 none FALSE            TRUE       5   0.005      1
##  maxlen target  ext
##       3  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 49
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [120 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 done [0.00s].
## writing ... [99 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

.

```
arules::inspect(regras[1:5])
```

lhs		rhs	support	confidence	coverage	lift	count
baking powder	=>	whole milk	0.01	0.52	0.02	2.05	91
other vegetables,oil	=>	whole milk	0.01	0.51	0.01	2.00	50
root vegetables,onions	=>	other vegetables	0.01	0.60	0.01	3.11	56
onions,whole milk	=>	other vegetables	0.01	0.55	0.01	2.82	65
other vegetables,hygiene articles	=>	whole milk	0.01	0.54	0.01	2.12	51

```
# reoderdenar por confianca:
regras <- sort(regras, by = "confidence", decreasing = TRUE)
arules::inspect(regras[1:5])
```

lhs		rhs	support	confidence	coverage	lift	count
butter,whipped/sour cream	=>	whole milk	0.01	0.66	0.01	2.58	66
pip fruit,whipped/sour cream	=>	whole milk	0.01	0.65	0.01	2.54	59
butter,yogurt	=>	whole milk	0.01	0.64	0.01	2.50	92
root vegetables,butter	=>	whole milk	0.01	0.64	0.01	2.50	81
tropical fruit,curd	=>	whole milk	0.01	0.63	0.01	2.48	64

```
# reoderdenar por lift:
regras = sort(regras, by = "lift", decreasing = TRUE)
arules::inspect(regras[1:5])
```

lhs		rhs	support	confidence	coverage	lift	count
tropical fruit,curd	=>	yogurt	0.01	0.51	0.01	3.69	52
pip fruit,whipped/sour cream	=>	other vegetables	0.01	0.60	0.01	3.12	55
root vegetables,onions	=>	other vegetables	0.01	0.60	0.01	3.11	56
citrus fruit,root vegetables	=>	other vegetables	0.01	0.59	0.02	3.03	102
tropical fruit,root vegetables	=>	other vegetables	0.01	0.58	0.02	3.02	121

```
# buscar regras que envolvam cerveja no lado direto (rhs):
regras <- apriori(data = Groceries,
                  parameter = list(supp = 0.001, conf = 0.08),
                  appearance = list(default = "lhs",
                                    rhs = "bottled beer"),
```

```
                    control = list(verbose = FALSE))

regras <- sort(regras, by = "lift", decreasing = TRUE)
arules::inspect(regras[1:5])
```

lhs		rhs	support	confidence	coverage	lift	count
liquor,red/blush wine	=>	bottled beer	0	0.90	0.00	11.24	19
soda,liquor	=>	bottled beer	0	0.57	0.00	7.10	12
liquor	=>	bottled beer	0	0.42	0.01	5.24	46
herbs,bottled water	=>	bottled beer	0	0.40	0.00	4.97	12
whole milk,soups	=>	bottled beer	0	0.38	0.00	4.71	11

```
# buscar regras que involvam cerveja no lado esquedo (lhs):
regras <- apriori(data = Groceries,
                  parameter = list(supp = 0.001, conf = 0.15,
                                   minlen = 2),
                  appearance = list(default = "rhs",
                                    lhs = "bottled beer"),
                  control = list(verbose = FALSE))

arules::inspect(regras)
```

lhs		rhs	support	confidence	coverage	lift	count
bottled beer	=>	bottled water	0.02	0.20	0.08	1.77	155
bottled beer	=>	soda	0.02	0.21	0.08	1.21	167
bottled beer	=>	rolls/buns	0.01	0.17	0.08	0.92	134
bottled beer	=>	other vegetables	0.02	0.20	0.08	1.04	159
bottled beer	=>	whole milk	0.02	0.25	0.08	0.99	201

Para visualizar interativamente as regras, pode-se utilizar

```
regras <- apriori(Groceries,
                  parameter = list(supp = 0.001, conf = 0.8))

plot(regras[1:20], method = "graph",
     interactive = TRUE, shading = NA)
```

.

Em geral, utilizam-se matrizes esparsas para representar os dados binários, pois a matriz de dados contém poucas entradas "1". Dessa forma, a representação esparsa leva a uma grande economia de memória. Veja mais detalhes na Seção A.3.

Capítulo 13

Sistemas de Recomendação

No Capítulo 12 vimos que regras de associação são uma maneira de indicar um produto a um usuário a partir do seu histórico de compra. Neste capítulo, estudaremos algumas técnicas de recomendação que funcionam em contextos mais gerais. Assuma que cada usuário pode atribuir uma nota (*avaliação*) para cada produto. Por exemplo, no Netflix, podemos avaliar cada filme como 1, 2, 3, 4 ou 5 estrelas.

De um ponto de vista formal, considere os conjuntos

$$\mathcal{U} = \{u_1, \ldots, u_m\} \qquad \text{e} \qquad \mathcal{I} = \{i_1, \ldots, i_n\},$$

em que $\mathcal{U}$ indica o conjunto de usuários e $\mathcal{I}$ indica o conjunto de produtos (*itens*). Seja $R_{j,k}$ a avaliação dada pelo usuário j ao produto k, como mostrado na Tabela 13.1.

Tabela 13.1: Estrutura dos dados para um problema sistemas de recomendação.

Usuário/Produto	$\mathbf{i_1}$	$\mathbf{i_2}$	$\ldots$	$\mathbf{i_n}$
u_1	$R_{1,1}$	$R_{1,2}$	$\ldots$	$R_{1,n}$
u_2	$R_{2,1}$	$R_{2,2}$	$\ldots$	$R_{2,n}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
u_m	$R_{m,1}$	$R_{m,2}$	$\ldots$	$R_{m,n}$

Normalmente diversas entradas dessa matriz são desconhecidas, pois muitos usuários nunca atribuíram notas para uma grande quantidade de produtos. Uma questão central em um problema de recomendação consiste em como imputar essas notas do modo a descobrir, para cada usuário, produtos não avaliados por ele que provavelmente o agradarão.

Existem dois principais grupos de sistemas de recomendação:

- **Sistemas baseados no conteúdo.** Esses sistemas são baseados em características dos produtos que o usuário gosta (por exemplo, diretores dos filmes, gêneros etc) e, com base nesses produtos, busca-se descobrir outros produtos com as mesmas características.
- **Sistemas baseados em filtros colaborativos**. Esses sistemas fornecem recomendações baseadas somente na matriz de notas. Utilizando essa matriz, procura-se por padrões (por exemplo, usuários parecidos com o usuário de interesse). A suposição básica de filtros colaborativos é que usuários que concordaram no passado irão concordar no futuro.

Sistemas baseados no conteúdo, em geral, são obtidos através de métodos de regressão já estudados anteriormente. Assim, focaremos aqui em filtros colaborativos.

Observação 13.1 (Normalização das notas)**.** É comum se observar um viés nas notas. Alguns usuários tendem a dar notas muito altas para todos os produtos; outros tendem a dar notas muito baixas. Para tentar levar isso em consideração, é comum normalizar as notas de cada usuário antes de aplicar os métodos descritos nesse capítulo. Uma forma de se fazer isso é trabalhar com as avaliações dadas por

$$\widetilde{R}_{j,k} = R_{j,k} - \bar{R}_j,$$

em que $\bar{R}_j$ é a média das notas dadas pelo usuário j.

□

Nas Seções 13.1-13.3 apresentaremos alguns filtros colaborativos comumente utilizados enquanto na Seção 13.4 discutiremos como avaliar a performance desses métodos, bem como fazer seleção de modelos.

13.1 Filtro Colaborativo Baseado no Usuário

Filtros colaborativos baseados no usuário funcionam de forma parecida com o método KNN (Seções 4.3 e 8.6). Inicialmente, define-se uma medida de similaridade entre os usuários que avalia a similaridade entre as notas por eles atribuídas. Uma medida usual é a correlação de Pearson:

$$\text{sim}(u_a, u_b) = \text{cor}(R^*_{u_a,\cdot}, R^*_{u_b,\cdot}),$$

em que $R^*_{u_a,\cdot}$ é o vetor de notas atribuídas pelo usuário u_a aos produtos avaliados por *ambos* u_a e u_b.

Seja $k \geq 1$ fixo. Para estimar $R_{j,l}$, a nota que o usuário j daria para o produto l caso o avaliasse, buscam-se os k usuários mais parecidos a j (isto é, os usuários com maior similaridade com j) *que avaliaram o produto* l. Calcula-se, então, as médias das notas dadas por esses k usuários ao produto l

$$\widehat{R}_{j,l} = \frac{1}{k} \sum_{s \in \mathcal{V}_j} R_{s,l},$$

em que $\mathcal{V}_j$ é o conjunto dos k usuários mais similares ao usuário j.

O valor de k pode ser escolhido com a utilização de técnicas abordadas na Seção 13.4.

13.2 Filtro Colaborativo Baseado no Produto

Filtros colaborativos baseados no produto são muito parecidos com filtros colaborativos baseados no usuário (Seção 13.1). A diferença básica entre ambos é que, para se estimar $R_{j,l}$, a nota que o usuário j daria para o produto l caso o avaliasse, buscam-se agora os k *produtos* mais parecidos ao item l (e não os k *usuários* mais parecidos com j).

Para medir a similaridade entre as notas atribuídas por cada um dos usuários ao produto l com as notas atribuídas por cada um dos usuários aos outros produtos, pode-se, novamente, utilizar a correlação de Pearson. Dessa vez, definida como:

$$\text{sim}(i_a, i_b) = \text{cor}(R^*_{\cdot,i_a}, R^*_{\cdot,i_b}),$$

em que $R^*_{\cdot,i_a}$ é o vetor de notas atribuídas para o produto i_a por cada um dos usuários que avaliaram i_a e i_b.

A predição para $R_{j,l}$ é então dada pelas médias das notas dos k produtos mais parecidos a l:

$$\widehat{R}_{j,l} = \frac{1}{k} \sum_{s \in \mathcal{V}_l} R_{j,s},$$

em que $\mathcal{V}_l$ é o conjunto dos k produtos mais similares ao produto l. Novamente, o valor de k pode ser escolhido com a utilização de técnicas abordadas na Seção 13.4.

13.3 FunkSVD

Neste método, desenvolvido por Simon Funk em https://sifter.org/simon/journal/20061211.html, cada usuário j é descrito por um vetor de variáveis latentes $\mathbf{u}_j$ e cada produto l é descrito por um vetor de variáveis latentes $\mathbf{v}_l$. A ideia é modelar cada nota utilizando a seguinte aproximação:

$$R_{j,l} \approx \mathbf{u}_j^\intercal \mathbf{v}_l.$$

Os vetores latentes são estimados de forma a minimizar

$$\sum_j \sum_l \left(R_{j,l} - \mathbf{u}_j^\intercal \mathbf{v}_l \right)^2,$$

em que a somatória é feita sob todos os índices referentes às avaliações feitas por cada usuário. Essa minimização é feita utilizando-se um método de gradiente descendente estocástico, o que torna o processo rápido e eficiente em termos de utilização de memória.

13.4 Seleção de Modelos

Para comparar modelos de sistemas de recomendação, utilizamos uma variação de data splitting (Seção 1.5.1). Essa técnica permite também escolher parâmetros dos modelos utilizados (por exemplo, k no filtro colaborativo). Primeiramente, separamos os usuários em dois conjuntos: treinamento e validação. Então, consideramos que algumas das notas dadas por alguns dos usuários do conjunto de validação são desconhecidas e verificamos a capacidade de previsão dos algoritmos utilizados para essas notas.

Seja $\mathcal{K}$ o conjunto de todos os pares usuário/produto para o qual a nota real $R_{i,j}$ é conhecida, mas não foi utilizada para treinar o modelo. Diversas medidas da capacidade preditiva podem ser utilizadas, dentre as quais

- **MSE:** $\frac{\sum_{(i,j)\in\mathcal{K}}(R_{i,j}-\widehat{R}_{i,j})^2}{|\mathcal{K}|}$
- **RMSE:** $\sqrt{EQM}$
- **MAE:** $\frac{\sum_{(i,j)\in\mathcal{K}}|R_{i,j}-\widehat{R}_{i,j}|}{|\mathcal{K}|}$.

Essas medidas avaliam o desempenho preditivo dos modelos e podem ser utilizadas para fazer seleção de modelos. Contudo, podemos ter interesse apenas em medir se o método está fornecendo boas recomendações, e não se as notas individuais são preditas de forma satisfatória. Para tanto, outras medidas podem ser utilizadas. Assuma que recomendamos para um dado usuário os N produtos com maior valor predito para a avaliação. Assuma também que definimos o que é uma avaliação boa e o que é uma avaliação ruim (por exemplo, uma avaliação boa é uma nota maior que 3).

Com base nas N recomendações feitas para cada usuário do conjunto de validação (isto é, os N produtos com maior nota predita), podemos construir a matriz de confusão apresentada na Tabela 13.2.

Tabela 13.2: Matriz de confusão para produtos recomendados.

Valor Verdadeiro	**Valor Predito**	
	Ruim	Bom
Ruim	a	b
Bom	c	d

A partir dessa tabela diversas medidas podem ser utilizadas para fazer seleção de modelos. Entre elas destacamos:

- **Acurácia:** $\frac{a+d}{a+b+c+d}$
- **VPP/*Precision*:** $\frac{d}{b+d}$
- **Sensibilidade/*Recall*:** $\frac{d}{c+d}$

Outra medidas de desempenho definidas na Seção 7.4 no contexto de classificação podem também ser utilizadas. A Figura 13.1 ilustra algumas dessas medidas para três sistemas de recomendação aplicados ao banco de dados `MovieLense`. Esses sistemas foram treinados com a utilização do pacote `recommenderlab`:

```
library("recommenderlab", verbose = FALSE)

# dados de filmes
data(MovieLense)

# listar algoritmos utilizados atribuindo nomes
metodos <- list(
  "item-based CF 1" = list(name = "IBCF", param = list(k = 5)),
  "user-based CF 1" = list(name = "UBCF", param = list(nn = 5)),
  "funkSVD" = list(name = "SVDF"))

# definir o esquema do data splitting
esquema1 <- evaluationScheme(MovieLense,
                             method = "split",
                             train = 0.8,
                             given = 15)
# given: nº de avaliações já feitas
# por cada usuário do conjunto de teste
```

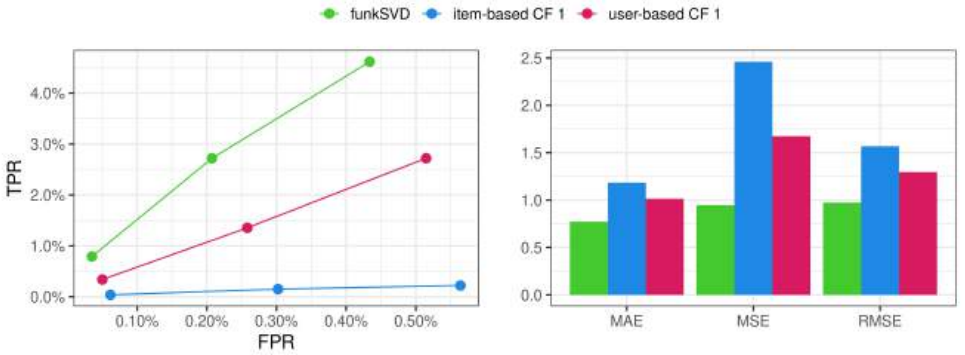


Figura 13.1: Valor verdadeiro positivo vs valor verdadeiro negativo para diferentes métodos e N = 1, 5 e 10.

```
# parte 1: avaliar o quao bom sao as notas atribuidas
results1 <- recommenderlab::evaluate(esquema1,
                                     metodos,
                                     type = "ratings",
                                     progress = FALSE)

# parte 2: avaliar o quao bom sao as recomendacoes top-N
esquema2 <- evaluationScheme(MovieLense,
                             method = "split",
                             train = 0.8,
                             given = 10,
                             goodRating = 3)

# goodRating: a partir de que nota consideramos que temos
#             uma boa predicao?

results2 <- recommenderlab::evaluate(esquema2, metodos,
                             n = c(1, 5, 10),
                             progress = FALSE)

# n: quantos produtos estamos recomendando
```

Parte IV

Apêndice

Apêndice A

Apêndice

A.1 Imagens

Este capítulo descreve brevemente como manipular imagens que têm formato do tipo *raster*, como por exemplo JPEG e PNG. Um formato é dito do tipo *raster* quando ele representa uma dada imagem utilizando uma ou mais matrizes que contenham informações sobre os pixels da figura. Para entender esse conceito, vamos começar com uma ideia simples: considere a matriz binária

$$M = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

A ideia chave é que podemos associar a essa matriz a imagem

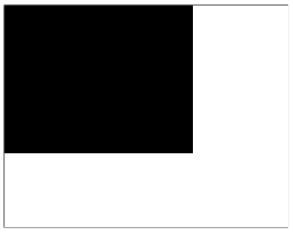


Figura A.1: Imagem correspondente à matriz M.

Nessa representação uma entrada 1 na matriz simboliza que o pixel correspondente a esse

elemento na imagem tem a cor preto enquanto o valor 0 simboliza um pixel de cor branca. A partir dessa ideia podemos criar imagens como

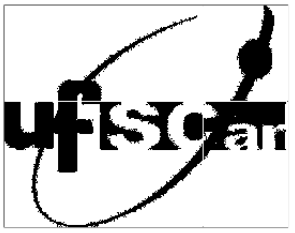


Figura A.2: Imagem feita utilizando-se apenas uma matriz binária.

Note que, quanto mais pixels em uma imagem (i.e., quanto mais entradas há em uma matriz), maior a resolução da imagem.

Essa ideia pode ser sofisticada. Ao invés de utilizar apenas 0 (branco) e 1 (preto), podemos utilizar qualquer número entre 0 e 1 para denotar uma intensidade de cinza. Com isso, podemos fazer imagens como

Figura A.3: Imagem feita utilizando-se uma matriz cujas entradas representam o tom de cinza a ser utilizado.

A partir desse princípio podemos construir imagens coloridas. Qualquer cor pode ser descrita como uma combinação das cores vermelho, verde e azul. Assim, podemos representar uma imagem com três matrizes simultaneamente:

- A primeira indica quanto vermelho vamos ter em cada pixel (cada elemento é um número entre 0 e 1)

- A segunda indica quanto verde vamos ter em cada pixel (cada elemento é um número entre 0 e 1)
- A terceira indica quanto azul vamos ter em cada pixel (cada elemento é um número entre 0 e 1)

Com isso, podemos faze

Figura A.4: Imagem feita utilizando três matrizes cujas entradas representam a intensidade de vermelho, verde e azul utilizada em cada pixel.

Essas matriz são as chamadas de RGB channels (RGB é a sigla em inglês para vermelho, verde e azul). Cada matriz é chamada de um *canal*.

Note que utilizar o valor 0 para branco e 1 preto é apenas uma convenção. Formatos diferentes utilizam convenções diferentes (por exemplo, alguns formatos atribuem 0 a branco e 256 a preto). Em vários dos formatos também é feita uma compressão da imagem, de modo que ela ocupe menos espaço. Nesse caso, leitores destes formatos devem saber também como descomprimi-las (o que muitas vezes leva a uma pequena perda de resolução).

Para ler uma imagem no R, podemos utilizar o seguinte código.

```
m <- matrix(c(1, 1, 0, 1, 1, 0, 0, 0, 0), 3, 3)
image(m[,3:1], col = c("white", "black"))

## Exemplo do simbolo da ufscar:
library(jpeg)
imagem <- readJPEG("1024px-UFSCar.jpg") # ler a imagem
dim(imagem)
#[1]  746 1024    3 # 3 matrizes com 746*1024 pixels
image(t(imagem[746:1,, 3]),
      col = grey.colors(1000, start = 0, end = 1))
# imagem em tons de cinza (utilizando apenas a terceira matriz)
rasterImage(imagem, 0, 0, 1, 1) # imagem colorida
```

A.2 Textos

Digamos que observamos diversos documentos de texto, como, por exemplo, o texto em uma página de internet, o texto de um tweet ou de um post do facebook. Nesta seção descreveremos o método *bag-of-words*. Esse método é utilizado para criar uma representação vetorial (ou seja, criar covariáveis) para cada texto. Considere os seguintes textos, retirados de uma conta de e-email:

Texto 1: $x_1 =$ "Poderoso Estimulante Natural - Esquente sua noite na cama."

Texto 2: $x_2 =$ "Olá professor, sou aluna de Mineração de Dados."

Texto 3: $x_3 =$ "Boa tarde professor Rafael, segue contato como pedido."

Texto 4: $x_4 =$ "Aumente sua performance na cama a noite usando esse estimulante. Esquente seu relacionamento!"

A ideia do bag-of-words (literalmente, "sacola de palavras") é listar todas as palavras que aparecem nos documentos e registrar a frequência com que cada uma aparece. Esse processo leva à seguinte matriz *documento-termo*:

Tabela A.1: Exemplo de matriz documento-termo

	poderoso	**estimulante**	**natural**	**esquente**	**sua**	**noite**	**na**	...	**esse**
Texto 1	1	1	1	1	1	1	1	...	0
Texto 2	0	0	0	0	0	0	1	...	0
Texto 3	0	0	0	0	0	0	0	...	0
Texto 4	0	1	0	1	1	1	1	...	1

Nessa matriz documentos "próximos" têm uma representação mais parecida, pois utilizam palavras parecidas. Há algumas variações que podem ser feitas nessa matriz documento-termo. Por exemplo, como documentos diferentes têm tamanhos diferentes, pode-se normalizar esses vetores (por exemplo, dividindo-se as frequências absolutas pelo tamanho de cada um dos documentos. Isso torna os diferentes documentos mais comparáveis.

Outra alteração comumente feita na representação bag-of-words consiste em remover palavras muito comuns, como artigos e preposições. Essas palavras são chamadas de *stop words*. A maioria dos pacotes para trabalhar com texto (como o `tm` do R) já contém uma lista delas. Finalmente, também pode-se juntar palavras que têm mesma raiz (um processo chamado de *stemming*).

Um problema da representação da Tabela A.1 é que ela ignora completamente a ordem das palavras. Uma maneira de contornar isso é adicionar termos que são *bigramas*. Bigramas são conjuntos de duas palavras que aparecem simultaneamente, uma seguida da outra (como "poderoso estimulante" no exemplo). Cada bigrama pode também fazer parte da matriz documento-termo. Essa ideia é generalizada para n-gramas.

A.3 Matrizes esparsas

Matrizes esparsas são matrizes com muitas entradas 0. Elas ocorrem naturalmente em diversas aplicações. Por exemplo, a matriz documento-termo descrita na Seção A.2 possui, em geral, quase todas as entradas 0, uma vez que cada texto utiliza poucas palavras de todo o dicionário. Analogamente, matrizes de dados utilizadas em algoritmos de regras de associação também costumam ser esparsas: a maior parte das compras de mercado envolvem pouquíssimos produtos.

Matrizes esparsas podem ser armazenadas de forma mais eficiente. Por exemplo, considere a matriz

$$\begin{pmatrix} 0 & -5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 9 & 0 & 8 & 0 & 0 \end{pmatrix}$$

Uma representação ingênua dessa matriz requer o armazenamento de $4 \times 7 = 28$ números, um para cada entrada da matriz. Uma representação mais eficiente consiste em armazenar apenas as entradas que são diferentes de zero. Assim, temos a seguinte representação nesse caso:

Linha	**Coluna**	**Valor**
1	2	-5
2	3	10
3	7	5
4	3	9
4	5	8

Essa segunda representação, que é chamada de *representação esparsa*, exige armazenar apenas $5 \times 3 = 15$ números. Assim, ela utiliza menos memória do computador, o que permite que matrizes muito maiores possam ser armazenadas.

Tanto o pacote `tm` quando o `arules` fornecem matrizes com essa representação. Contudo, nem todos os pacotes com algoritmos de aprendizado de máquina dão suporte para matrizes com representação esparsa. Por exemplo, enquanto o pacote `glmnet` pode ajustar uma regressão com uma matriz de covariáveis nesse formato, a função `lm` requer que a matriz seja convertida para uma matriz em sua representação usual. Essa limitação é um obstáculo para que algumas dessas funções possam ser utilizadas em bancos de dados grandes (mas esparsos).

Índice Remissivo

.

Bibliografia

Agrawal, R., Imieliński, T. & Swami, A. (1993). Mining association rules between sets of items in large databases, Em *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*.

Aronszajn, N. (1950). Theory of Reproducing Kernels. *Transactions of the American Mathematical Society*, *68*(3), 337–404.

Arthur, D. & Vassilvitskii, S. (2006). *k-means++: The advantages of careful seeding* (rel. técn.). Stanford.

Aswani, A., Bickel, P., Tomlin, C. Et al. (2011). Regression on manifolds: estimation of the exterior derivative. *The Annals of Statistics*, *39*(1), 48–81.

Belkin, M., Hsu, D., Ma, S. & Mandal, S. (2019a). Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, *116*(32), 15849–15854.

Belkin, M., Hsu, D. J. & Mitra, P. (2018). Overfitting or perfect fitting? Risk bounds for classification and regression rules that interpolate, Em *Advances in neural information processing systems*.

Belkin, M., Rakhlin, A. & Tsybakov, A. B. (2019b). Does data interpolation contradict statistical optimality?, Em *The 22nd International Conference on Artificial Intelligence and Statistics*.

Bellman, R. (1966). Dynamic programming. *Science*, *153*(3731), 34–37.

Benedetti, J. K. (1977). On the nonparametric estimation of regression functions. *Journal of the Royal Statistical Society. Series B (Methodological)*, 248–253.

Bengio, Y., Delalleau, O., Roux, N. L., Paiement, J.-F., Vincent, P. & Ouimet, M. (2004). Learning eigenfunctions links spectral embedding and kernel PCA. *Neural computation*, *16*(10), 2197–2219.

Beran, R. (2000). REACT scatterplot smoothers: Superefficiency through basis economy. *Journal of the American Statistical Association*, *95*(449), 155–171.

.

Biau, G. (2012). Analysis of a Random Forests Model. *JMLR*, *13*(1), 1063–1095.

Bickel, P. J. & Li, B. (2007). Local polynomial regression on unknown manifolds. Em *Complex datasets and inverse problems* (pp. 177–186). Institute of Mathematical Statistics.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Botari, T., Izbicki, R. & de Carvalho, A. C. (2019). Local Interpretation Methods to Machine Learning Using the Domain of the Feature Space, Em *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer.

Breiman, L. (2001a). Statistical modeling: The two cultures. *Statistical Science*, *16*(3), 199–231.

Breiman, L. (1996). Stacked regressions. *Machine learning*, *24*(1), 49–64.

Breiman, L. (2001b). Random Forests. *Machine Learning*, *45*(1), 5–32.

Chen, T. & Guestrin, C. (2016). Xgboost: A scalable tree boosting system, Em *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*.

Chentsov, N. (1962). Evaluation of an unknown distribution density from observations. *DOKLADY*, *147*(1), 1559–1562.

Cherkassky, V., Shao, X., Mulier, F. M. & Vapnik, V. N. (1999). Model complexity control for regression using VC generalization bounds. *IEEE transactions on Neural Networks*, *10*(5), 1075–1089.

Cleveland, W. S. (1979). Robust locally weighted regression and smoothing scatterplots. *Journal of the American statistical association*, *74*(368), 829–836.

Coifman, R. R. & Lafon, S. (2006). Diffusion maps. *Applied and computational harmonic analysis*, *21*(1), 5–30.

Cortes, C. & Vapnik, V. (1995). Support vector machine. *Machine learning*, *20*(3), 273–297.

Coscrato, V., de Almeida Inácio, M. H. & Izbicki, R. (2020). The NN-Stacking: Feature weighted linear stacking through neural networks. *Neurocomputing*.

Coscrato, V., Inácio, M. H. d. A., Botari, T. & Izbicki, R. (2019). NLS: an accurate and yet easy-to-interpret regression method. *arXiv preprint arXiv:1910.05206*.

Dalmasso, N., Pospisil, T., Lee, A. B., Izbicki, R., Freeman, P. E. & Malz, A. I. (2020). Conditional density estimation tools in python and R with applications to photometric redshifts and likelihood-free cosmological inference. *Astronomy and Computing*, 100362.

De Boor, C., De Boor, C., Mathématicien, E.-U., De Boor, C. & De Boor, C. (1978). *A practical guide to splines* (Vol. 27). Springer-Verlag New York.

DeGroot, M. H. & Schervish, M. J. (2012). *Probability and statistics*. Pearson Education.

Dietterich, T. G. (2000). Ensemble methods in machine learning, Em *International workshop on multiple classifier systems*. Springer.

Diniz, M. A., Izbicki, R., Lopes, D. & Salasar, L. E. (2019). Comparing probabilistic predictive models applied to football. *Journal of the Operational Research Society*, *70*(5), 770–782.

Drineas, P. & Mahoney, M. W. (2005). On the Nyström method for approximating a Gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, *6*(Dec), 2153–2175.

Drucker, H., Burges, C. J., Kaufman, L., Smola, A. J. & Vapnik, V. (1997). Support vector regression machines, Em *Advances in neural information processing systems*.

Efromovich, S. (1999). *Nonparametric Curve Estimation: Methods, Theory and Application*. Springer.

Fan, J. (1993). Local Linear Regression Smoothers and Their Minimax Efficiencies. *Annals of Statistics*, *21*, 196–216.

Fan, J. & Gijbels, I. (1996). *Local polynomial modelling and its applications* (Vol. 66). Monographs on statistics; applied probability 66, CRC Press.

Fan, J., Yao, Q. & Tong, H. (1996). Estimation of conditional densities and sensitivity measures in nonlinear dynamical systems. *Biometrika*, *83*(1), 189–206.

Fernandez, P. & F., P. C. M. (2020). *Data Science, Marketing Business*. Insper. https://datascience.insper.edu.br

Forman, G. (2006). Quantifying trends accurately despite classifier error and class imbalance, Em *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*.

Freeman, P. E., Izbicki, R. & Lee, A. B. (2017). A unified framework for constructing, tuning and assessing photometric redshift density estimates in a selection bias setting. *Monthly Notices of the Royal Astronomical Society*, *468*(4), 4556–4565.

Freund, Y. & Schapire, R. E. (1995). A desicion-theoretic generalization of on-line learning and an application to boosting, Em *European conference on computational learning theory*. Springer.

Friedman, J., Hastie, T. & Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, *33*(1), 1.

.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 1189–1232.

Gauss, C. (1809). *Theoria motus corporum coelestium in sectionibus conicis solem ambientium*.

Goldstein, A., Kapelner, A., Bleich, J. & Pitkin, E. (2015). Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *Journal of Computational and Graphical Statistics*, *24*(1), 44–65.

Goodfellow, I., Bengio, Y. & Courville, A. (2016). *Deep learning*. MIT press.

Greenshtein, E., Ritov, Y. Et al. (2004). Persistence in high-dimensional linear predictor selection and the virtue of overparametrization. *Bernoulli*, *10*(6), 971–988.

Györfi, L., Kohler, M., Krzyzak, A. & Walk, H. (2006). *A distribution-free theory of nonparametric regression*. Springer Science & Business Media.

Hall, P., Racine, J. S. & Li, Q. (2004). Cross-Validation and the Estimation of Conditional Probability Densities. *Journal of the American Statistical Association*, *99*, 1015–1026.

Hastie, T. & Tibshirani, R. (1986). Generalized Linear Models (with Discussion). *Statistical Science*, *1*(3), 297–318.

Hastie, T., Tibshirani, R. & Friedman, J. (2001). *The Elements of Statistical Learning*. New York, NY, USA, Springer New York Inc.

Hoerl, A. E. & Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, *12*(1), 55–67.

Hyndman, R. J., Bashtannyk, D. M. & Grunwald, G. K. (1996). Estimating and visualizing conditional densities. *Journal of Computational & Graphical Statistics*, *5*, 315–336.

Izbicki, R. & Lee, A. B. (2016). Nonparametric Conditional Density Estimation in a High-Dimensional Regression Setting. *Journal of Computational and Graphical Statistics*, *25*(4), 1297–1316.

Izbicki, R. & Lee, A. B. (2017). Converting high-dimensional regression to high-dimensional conditional density estimation. *Electron. J. Statist.*, *11*(2), 2800–2831.

Izbicki, R., Lee, A. B. & Freeman, P. E. (2017). Photo-z estimation: An example of nonparametric conditional density estimation under selection bias. *The Annals of Applied Statistics*, *11*(2), 698–724.

Izbicki, R., Lee, A. B. & Pospisil, T. (2019). ABC–CDE: Toward approximate bayesian computation with complex high-dimensional data and limited simulations. *Journal of Computational and Graphical Statistics*, *28*(3), 481–492.

Izbicki, R. & Musetti, M. (2020). Combinando Métodos de Aprendizado Supervisionado para a Melhoria da Previsão do Redshift de Galáxias. *TEMA (São Carlos)*, *21*(1), 117.

Izbicki, R., Shimizu, G. & Stern, R. B. (2020a). CD-split: efficient conformal regions in high dimensions. *arXiv preprint arXiv:2007.12778*.

Izbicki, R., Shimizu, G. T. & Stern, R. B. (2020b). Distribution-free conditional predictive bands using density estimators. *Proceedings of Machine Learning Research (AISTATS Track)*.

Izbicki, R. & Stern, R. B. (2013). Learning with many experts: model selection and sparsity. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, *6*(6), 565–577.

James, G., Witten, D., Hastie, T. & Tibshirani, R. (2013). *An introduction to statistical learning* (Vol. 112). Springer.

Johnson, W. B. & Lindenstrauss, J. (1984). Extensions of Lipschitz mappings into a Hilbert space. *Contemporary mathematics*, *26*(189-206), 1.

Kassambara, A. & Mundt, F. (2020). *factoextra: Extract and Visualize the Results of Multivariate Data Analyses* [R package version 1.0.7]. R package version 1.0.7. https://CRAN.R-project.org/package=factoextra

Kimeldorf, G. & Wahba, G. (1971). Some results on Tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, *33*(1), 82–95.

Kimeldorf, G. S. & Wahba, G. (1970). A correspondence between Bayesian estimation on stochastic processes and smoothing by splines. *The Annals of Mathematical Statistics*, 495–502.

Kpotufe, S. (2011). k-NN regression adapts to local intrinsic dimension, Em *Advances in Neural Information Processing Systems*.

Krantz, S. G. & Parks, H. R. (2002). *A primer of real analytic functions*. Springer.

Lad, F. (1996). *Operational subjective statistical methods: A mathematical, philosophical, and historical introduction* (Vol. 315). Wiley-Interscience.

Lee, A. B. & Izbicki, R. (2016). A spectral series approach to high-dimensional nonparametric regression. *Electronic Journal of Statistics*, *10*(1), 423–463.

Legendre, A. M. (1805). *Nouvelles méthodes pour la détermination des orbites des comètes*. F. Didot.

Lei, J., G'Sell, M., Rinaldo, A., Tibshirani, R. J. & Wasserman, L. (2018). Distribution-free predictive inference for regression. *Journal of the American Statistical Association*, *113*(523), 1094–1111.

.

Lei, J. & Wasserman, L. (2014). Distribution-free prediction bands for non-parametric regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *76*(1), 71–96.

Lundberg, S. M. & Lee, S.-I. (2017). A unified approach to interpreting model predictions, Em *Advances in neural information processing systems*.

McCulloch, W. S. & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*(4), 115–133.

Minh, H. Q. (2010). Some properties of Gaussian reproducing kernel Hilbert spaces and their implications for function approximation and learning theory. *Constructive Approximation*, *32*(2), 307–338.

Minh, H. Q., Niyogi, P. & Yao, Y. (2006). Mercer's theorem, feature maps, and smoothing, Em *International Conference on Computational Learning Theory*. Springer.

Molnar, C. (2019). *Interpretable machine learning*. Lulu.com.

Morettin, P. A. & Singer, J. M. (2019). Introdução à Ciência de Dados.

Nadaraya, E. A. (1964). On estimating regression. *Theory of Probability & Its Applications*, *9*(1), 141–142.

Nadler, B., Lafon, S., Coifman, R. R. & Kevrekidis, I. G. (2006). Diffusion maps, spectral clustering and reaction coordinates of dynamical systems. *Applied and Computational Harmonic Analysis*, *21*(1), 113–127.

Neter, J., Kutner, M. H., Nachtsheim, C. J. & Wasserman, W. (1996). Applied linear statistical models.

Nosedal-Sanchez, A., Storlie, C. B., Lee, T. C. & Christensen, R. (2012). Reproducing kernel hilbert spaces for penalized regression: A tutorial. *The American Statistician*, *66*(1), 50–60.

Papadopoulos, H., Proedrou, K., Vovk, V. & Gammerman, A. (2002). Inductive confidence machines for regression, Em *European Conference on Machine Learning*. Springer.

Park, T. & Casella, G. (2008). The bayesian lasso. *Journal of the American Statistical Association*, *103*(482), 681–686.

Parzen, E. (1962). On estimation of a probability density function and mode. *The annals of mathematical statistics*, *33*(3), 1065–1076.

Pearce, N. D. & Wand, M. P. (2006). Penalized splines and reproducing kernel methods. *The american statistician*, *60*(3).

Pearl, J. (2014). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier.

Pontil, M. (2003). Learning with reproducing kernel Hilbert spaces: a guide tour. *Bulletin of the Italian Artificial Intelligence Association, AI* IA Notizie*.

Ravikumar, P., Lafferty, J., Liu, H. & Wasserman, L. (2009). Sparse additive models. *Journal of the Royal Statistical Society: Series B*, *71*(5), 1009–1030.

Ribeiro, M. T., Singh, S. & Guestrin, C. (2016). "Why should i trust you?"Explaining the predictions of any classifier, Em *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*.

Ribeiro, M. T., Singh, S. & Guestrin, C. (2018). Anchors: High-precision model-agnostic explanations, Em *Thirty-Second AAAI Conference on Artificial Intelligence*.

Rojas, R. (2013). *Neural networks: a systematic introduction*. Springer Science & Business Media.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, *65*(6), 386.

Rosenblatt, M. (1969). Conditional probability density and regression estimators. Em P. Krishnaiah (Ed.), *Multivariate Analysis II*.

Saerens, M., Latinne, P. & Decaestecker, C. (2002). Adjusting the outputs of a classifier to new a priori probabilities: a simple procedure. *Neural computation*, *14*(1), 21–41.

Schmidt, S. J., Malz, A. I., Soo, J. Y. H., Almosallam, I. A., Brescia, M., Cavuoti, S., Cohen-Tanugi, J., Connolly, A. J., DeRose, J., Freeman, P. E., Graham, M. L., Iyer, K. G., Jarvis, M. J., Kalmbach, J. B., Kovacs, E., Lee, A. B., Longo, G., Morrison, C. B., Newman, J. A., … Izbicki, R. (2020). Evaluation of probabilistic photometric redshift estimation approaches for The Rubin Observatory Legacy Survey of Space and Time (LSST) [staa2799]. *Monthly Notices of the Royal Astronomical Society*, https://academic.oup.com/mnras/advance-article-pdf/doi/10.1093/mnras/staa2799/33745738/staa2799.pdf. https://doi.org/10.1093/mnras/staa2799

Scholkopf, B. & Smola, A. (2002). Learning with kernels. *MIT Press*, 110–146.

Shi, T., Belkin, M., Yu, B. Et al. (2009). Data spectroscopy: Eigenspaces of convolution operators and clustering. *The Annals of Statistics*, *37*(6B), 3960–3984.

Stamey, T. A., Kabalin, J. N. & Ferrari, M. (1989). Prostate Specific Antigen in the Diagnosis and Treatment of Adenocarcinoma of the Prostate. II. Radiation Treated Patients. *The Journal of urology*, *141*(5), 1084–1087.

.

Steinwart, I. & Christmann, A. (2008). *Support vector machines*. Springer Science & Business Media.

Stone, C. J. (1977). Consistent nonparametric regression. *The annals of statistics*, 595–620.

Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society: Series B (Methodological)*, *36*(2), 111–133.

Sugiyama, M., Lawrence, N. D. & Schwaighofer, A. (2017). *Dataset shift in machine learning*. The MIT Press.

Sugiyama, M., Takeuchi, I., Suzuki, T., Kanamori, T., Hachiya, H. & Okanohara, D. (2010). Conditional density estimation via least-squares density ratio estimation, Em *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*.

Tenenbaum, J. B., De Silva, V. & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *science*, *290*(5500), 2319–2323.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, *58*(1), 267–288.

Tsybakov, A. B. (2008). *Introduction to nonparametric estimation*. Springer Science & Business Media.

Vapnik, V. N. (1999). An overview of statistical learning theory. *IEEE transactions on neural networks*, *10*(5), 988–999.

Vaz, A. F., Izbicki, R. & Stern, R. B. (2019). Quantification Under Prior Probability Shift: the Ratio Estimator and its Extensions. *Journal of Machine Learning Research*, *20*(79), 1–33.

Vovk, V. Et al. (2005). *Algorithmic learning in a random world*. Springer Science & Business Media.

Vovk, V., Nouretdinov, I., Gammerman, A. Et al. (2009). On-line predictive linear regression. *The Annals of Statistics*, *37*(3), 1566–1590.

Wahba, G. (1990). *Spline Models for Observational Data*. SIAM.

Wasserman, L. (2006). *All of Nonparametric Statistics* (Springer, Ed.). Springer-Verlag New York, Inc.

Watson, G. S. (1964). Smooth regression analysis. *Sankhyā: The Indian Journal of Statistics, Series A*, 359–372.

ISBN: 978-65-00-02410-4