

# Aprendizagem de Máquina

Reconhecimento de Padrões — Redes Neurais — Inteligência Artificial

**Thomas Walter Rauber**

*Última atualização desta versão: 20 de outubro de 2021, 11:58:27*

Copyright © 2020 Thomas Walter Rauber

PUBLISHED BY PUBLISHER

BOOK-WEBSITE.COM

*First printing, March 2020*

# Sumário

<b>1</b>	<b>Flores</b>	<b>9</b>
<b>1.1</b>	<b>Iris Data</b>	<b>9</b>
1.1.1	Características ou Atributos	10
1.1.2	Classes	11
1.1.3	Vetores	11
1.1.4	Visualização do Espaço das Características	15
1.1.5	Padrões	16
<b>1.2</b>	<b>Variáveis Aleatórias</b>	<b>16</b>
1.2.1	Variáveis Aleatórias Contínuas	17
1.2.2	Variáveis Aleatórias Discretas	19
1.2.3	Valor Esperado, ou Média de uma Variável Aleatória	19
1.2.4	Momentos de uma Variável Aleatória	20
<b>1.3</b>	<b>Estimativa de Parâmetros de Variáveis Aleatórias</b>	<b>21</b>
1.3.1	Estimando a Média	21
1.3.2	Estimando a Variância	22
1.3.3	Geração de Variáveis Aleatórias	22
<b>1.4</b>	<b>Variáveis Aleatórias Multidimensionais</b>	<b>23</b>
1.4.1	Covariância e Correlação	25
1.4.2	Densidade Gaussiana Multidimensional	28
<b>2</b>	<b>Régressão e Classificação Linear</b>	<b>31</b>
<b>2.1</b>	<b>Modelos Lineares</b>	<b>31</b>
2.1.1	Um Exemplo Mínimo	31
2.1.2	Aprendizagem Supervisionada	32
2.1.3	Régressão Linear Simples	33

<b>2.2</b>	<b>Regressão Linear</b>	<b>53</b>
2.2.1	Máquina Linear . . . . .	53
2.2.2	Máquina Linear como Rede Neural Artificial . . . . .	54
2.2.3	Treinamento Determinístico de Máquina Linear . . . . .	54
<b>2.3</b>	<b>Classificação Linear por Máquina Linear</b>	<b>58</b>
2.3.1	Score, Funções Discriminativas, Regiões e Fronteiras de Classes . . . . .	59
2.3.2	Região de Decisão Convexa . . . . .	65
<b>2.4</b>	<b>Classificador Linear Binário</b>	<b>65</b>
2.4.1	Modelo do Classificador Linear Binário . . . . .	66
2.4.2	Orientação do Hiperplano e Distâncias . . . . .	69
2.4.3	Aprendizado do Classificador Linear Binário . . . . .	71
2.4.4	Coordenadas Homogêneas para Representação Aumentada dos Padrões e dos Parâmetros . . . . .	72
2.4.5	Consideração Crítica da Rotulação do Classificador Linear Binário . . . . .	75
<b>3</b>	<b>Descida de Gradiente: Otimização de Funções Não Lineares . . . . .</b>	<b>79</b>
<b>3.1</b>	<b>Procurando o Mínimo</b>	<b>79</b>
3.1.1	Estratégia Básica de Descida de Gradiente em uma Dimensão . . . . .	81
3.1.2	Algoritmo Iterativo de Otimização . . . . .	83
<b>3.2</b>	<b>Descida de Gradiente em Função Vetorial</b>	<b>85</b>
3.2.1	Descida de Gradiente Estocástica . . . . .	87
3.2.2	Regressão Linear . . . . .	89
3.2.3	Generalização de Regressão Linear Múltipla . . . . .	92
<b>3.3</b>	<b>Descida de Gradiente para Funções Não Lineares</b>	<b>94</b>
3.3.1	Funções Quadráticas . . . . .	95
3.3.2	Minimização de Funções Quadráticas . . . . .	96
3.3.3	Minimização Iterativa de Funções Quadráticas . . . . .	98
3.3.4	Descida pelo Gradiente Conjugado . . . . .	105
3.3.5	Gradiente Conjugado Não Linear . . . . .	109
<b>3.4</b>	<b>Heurísticas de Primeira Ordem</b>	<b>111</b>
3.4.1	Taxa de Aprendizagem e Demais Hiperparâmetros Constantes . . . . .	111
3.4.2	Taxa de Aprendizagem e Demais Hiperparâmetros Variáveis . . . . .	113
<b>3.5</b>	<b>Perceptron</b>	<b>116</b>
3.5.1	Conceitos Históricos do Perceptron . . . . .	116
3.5.2	Perceptron Elementar $\alpha$ . . . . .	118
3.5.3	Algoritmo de Aprendizagem do Perceptron Elementar $\alpha$ . . . . .	119
3.5.4	Convergência do Hiperplano . . . . .	123
<b>4</b>	<b>Extração de Características . . . . .</b>	<b>125</b>
<b>4.1</b>	<b>Introdução</b>	<b>125</b>
<b>4.2</b>	<b>Extração Linear de Características</b>	<b>127</b>
4.2.1	Análise de Componentes Principais . . . . .	127
<b>4.3</b>	<b>Extração Não Linear de Características</b>	<b>138</b>
4.3.1	Máquina Extrema de Aprendizado, <i>Extreme Learning Machine, ELM</i> . . . . .	138
4.3.2	Modelos Lineares Generalizados . . . . .	146

<b>5</b>	<b>Redes Neurais Multicamadas Adiante (Feedforward) .....</b>	<b>149</b>
5.1	O Perceptron Multicamada e o Algoritmo de Retropropagação de Erro	149
5.1.1	Modelo do Perceptron Multicamada .....	149
5.1.2	Função de Perda do Erro Quadrático .....	156
5.1.3	Ajuste de Pesos por Descida de Gradiente .....	157
5.1.4	Retropropagação de Erro .....	163
5.1.5	Funções de Perda e seus Gradientes .....	171
5.1.6	Funções de Ativação .....	171
5.2	<b>Rede Convolucional, CNN</b>	<b>171</b>
5.2.1	Modelo .....	171
<b>6</b>	<b>Redes Neurais Baseadas em Energia .....</b>	<b>173</b>
6.1	<b>Rede de Hopfield</b>	<b>173</b>
6.1.1	Modelo da Rede de Hopfield .....	173
6.1.2	Memória Auto-Associativa .....	175
6.1.3	Modelo de Bacias de Atração .....	178
6.1.4	Capacidade de Armazenamento .....	181
6.1.5	Estados Espúrios .....	182
6.2	<b>Máquina Restrita de Boltzmann <i>Restricted Boltzmann Machine, RBM</i></b>	<b>185</b>
6.2.1	Modelo RBM .....	185
<b>7</b>	<b>Classificação Paramétrica .....</b>	<b>187</b>
7.1	<b>Regra de Bayes</b>	<b>187</b>
7.2	<b>Análise Discriminativa Quadrática</b>	<b>192</b>
7.2.1	Classificador de Erro Mínimo .....	192
7.2.2	Classificador de Verossimilhança Máxima .....	194
7.2.3	Classificador de Risco Mínimo .....	194
7.2.4	Análise Discriminativa Quadrática Multidimensional .....	197
7.3	<b>Naïve Bayes</b>	<b>200</b>
7.4	<b>Modelo Mistura de Gaussianas (<i>Gaussian Mixture Model, GMM</i>)</b>	<b>202</b>
7.5	<b>Maximização da Verossimilhança</b>	<b>205</b>
7.5.1	Verossimilhança ( <i>Likelihood</i> ) .....	205
7.5.2	Verossimilhança para uma Amostra $x_1$ .....	205
7.5.3	Verossimilhança para mais que uma Amostra .....	207
7.5.4	Maximização Determinística da Verossimilhança .....	208
7.5.5	Maximização Iterativa da Verossimilhança, Algoritmo EM-GMM Esperança-Maximização (EM) para Mistura de Gaussianas GMM .....	211
<b>8</b>	<b>Classificação Não-Paramétrica, Vizinho-Mais-Próximo, Árvores de Decisão e Regressão .....</b>	<b>219</b>
8.1	<b>K-Vizinhos-Mais-Próximos</b>	<b>219</b>
8.1.1	1-Vizinho-Mais-Próximo .....	219
8.1.2	Classificação de K-Vizinhos-Mais-Próximos .....	220
8.1.3	Aproximação de Probabilidade a Posteriori Bayesiana pelo K-Vizinhos-Mais-Próximos	221
8.1.4	A Inequação de Cover & Hart .....	223
8.1.5	Métricas de Distância .....	224

8.1.6	Varições do Classificador K-Vizinhos-Mais-Próximos . . . . .	224
<b>8.2</b>	<b>K-Vizinhos-Mais-Próximos com Kernel</b>	<b>225</b>
<b>8.3</b>	<b>Árvores de Decisão</b>	<b>229</b>
8.3.1	Árvores de Decisão com Características Simbólicas . . . . .	230
8.3.2	Relação entre Entropia, Informação Mútua e Divergência Kullback–Leibler . . . . .	241
8.3.3	Limitações na Construção de Árvores de Decisão . . . . .	245
<b>8.4</b>	<b>Árvores de Regressão</b>	<b>245</b>
<b>8.5</b>	<b>Ensembles de Árvores de Decisão e Regressão, Floresta</b>	<b>252</b>
8.5.1	Divisão de Dados de Treinamento em Conjuntos Aleatórios . . . . .	253
8.5.2	Geração da Floresta e Fusão de Saídas das Árvores . . . . .	253
<b>9</b>	<b>Avaliação de Desempenho</b> . . . . .	<b>259</b>
<b>9.1</b>	<b>Introdução</b>	<b>259</b>
9.1.1	Experimentos Ingênuos . . . . .	259
9.1.2	Diagrama de Caixa ( <i>Boxplot</i> ) . . . . .	261
<b>9.2</b>	<b>Critérios de Desempenho para Classificadores</b>	<b>265</b>
<b>9.3</b>	<b>Curvas de Desempenho, ROC e Confiabilidade Positiva–Sensitividade (<i>Precision–Recall</i>)</b>	<b>271</b>
9.3.1	O espaço FPR–TPR e a curva ROC . . . . .	276
9.3.2	O espaço PPV–TPR <i>Precision–Recall</i> . . . . .	278
<b>9.4</b>	<b>Critérios de Desempenho para Regressores</b>	<b>278</b>
<b>9.5</b>	<b>Estimativa de Desempenho de Modelo</b>	<b>281</b>
9.5.1	Técnicas de Divisão de Dados na Validação . . . . .	282
9.5.2	Dados Sintéticos como Verdade Fundamental . . . . .	289
9.5.3	Experimentos com Dados Sintéticos . . . . .	290
<b>9.6</b>	<b>Validação Cruzada e Seleção de Modelo</b>	<b>291</b>
9.6.1	Melhores hiperparâmetros por um único ciclo externo . . . . .	292
9.6.2	Melhores hiperparâmetros diferentes por ciclos externos diferentes . . . . .	295
9.6.3	Experimentos de Validação Cruzada com Dados Sintéticos . . . . .	296
<b>10</b>	<b>Seleção de Características</b> . . . . .	<b>299</b>
<b>10.1</b>	<b>Introdução</b>	<b>299</b>
<b>10.2</b>	<b>Critérios de Seleção</b>	<b>302</b>
<b>10.3</b>	<b>Estratégias de Busca</b>	<b>307</b>
10.3.1	Busca Univariável: Melhores Características . . . . .	307
10.3.2	Seleção Sequencial Adiante, <i>Sequential Forward Selection</i> , SFS . . . . .	307
10.3.3	Seleção Sequencial Para Trás, <i>Sequential Backward Selection</i> , SBS . . . . .	309
10.3.4	Seleção Sequencial Adiante Flutuante, <i>Sequential Forward Floating Selection</i> , SFFS	311
<b>11</b>	<b>Regularização</b> . . . . .	<b>317</b>
<b>12</b>	<b>Ferramentas de Análise Estatística e Visual</b> . . . . .	<b>319</b>
<b>12.1</b>	<b>Testes de Hipóteses</b>	<b>319</b>

<b>12.2 Testes Estatísticos Comparativos entre Modelos</b>	<b>319</b>
12.2.1 Teste de Friedman . . . . .	319
12.2.2 Teste de Nemenyi . . . . .	319
<b>12.3 Visualização de Dados</b>	<b>319</b>
12.3.1 Gráfico de Sammon . . . . .	319
12.3.2 Embutido Estocástico de Distribuição t-Student de Vizinhos, <i>T-distributed Stochastic Neighbor Embedding</i> , t-SNE . . . . .	319
<b>13 To Do . . . . .</b>	<b>321</b>
<b>Index . . . . .</b>	<b>331</b>



# 1. Flores

Este livro<sup>1</sup> tenta explicar um assunto complexo da maneira mais fácil possível. Sempre quando tenho a impressão que o leitor eventualmente precisa informações novas sobre matemática, ou refrescar o que já viu, colocarei a teoria junto. Isso vai aumentar o volume do texto, mas poupa o leitor de procurar em outra fonte. Leitores com os fundamentos já firmes, podem simplesmente ignorar essas passagens que às vezes possam parecer demasiado explícitas. A intenção é ensinar a teoria e uma caixa de ferramentas computacionais que permitem criar sistemas que exibem o que o ser humano chama de inteligência. Pessoalmente creio que nunca será possível desvendar os segredos como o mundo biológico armazena e processa informação. Se chegar nesse ponto, seremos obsoletos, pois seria possível criar uma ponte perfeita entre o mundo digital e o mundo analógico. Seria possível fazer uma cópia de nós mesmos, armazenar para sempre, e posteriormente transferir para um clone. Vamos nos limitar ao que é possível. Mesmo assim sistemas computacionais impressionam jogando xadrez ou Go melhor que os campeões humanos, reconhecendo faces em um simples aparelho ubíquo que nos acompanha dia e noite, ou controlando veículos autônomos. Não adianta, é matemática. Qualquer apresentação sobre inteligência artificial, aprendizagem de máquina, redes neurais torna-se mera conversa, se não tocar no mundo numérico. Mão a obra.

## 1.1 Iris Data

Durante o ensino das disciplinas relacionadas aos assuntos de aprendizado de máquina sempre usei um conjunto de dados que se tornou o padrão dos conjuntos de dados para apresentar diversas técnicas relacionadas à classificação. Trata-se de um conjunto de flores Iris [61]<sup>2</sup>, colecionados em 1935 pelo biólogo E. Anderson e usado em 1936 pelo estatístico e biólogo R. A. Fisher [36]. É

<sup>1</sup>Se o leitor tiver acesso a este texto em forma eletrônica, recomendo abrir o livro duas vezes. Isso se justifica pela grande quantidade de referências cruzadas de fórmulas, definições e figuras. Aparecerão frequentemente referências do tipo “... como já mostrado na figura tal, na página tal ...”. Com duas janelas simultâneas, não será necessário pular para frente e para trás, mantendo um fluxo de leitura melhor.

<sup>2</sup>Embora seja às vezes uma fonte cientificamente menos elaborada, o Wikipédia, para alguns assuntos dispõe boas explicações e pode ser usado sem restrição como material didático profissional. Frequentemente a língua disponível é inglês, ou o material é mais extenso em inglês.

uma abstração do mundo real de três espécies de Iris, reduzidos a quatro características numéricas, as larguras e comprimentos da sépala e pétala, com 50 amostras de cada uma das classes. Tem-se uma representação numérica de 150 amostras, usando as quatro características e o conhecimento a qual das três classes cada uma das flores pertence. A partir dessa representação pode-se usar um computador digital para processar esses dados. Vale a pena apresentar em pormenor esse conjunto de dados e posteriormente sempre fazer referência quando é apresentado um assunto novo. Por exemplo, em uma tarefa de classificação pode-se escolher duas das três classes, dividir todas as 150 amostras em um conjunto de treinamento e um conjunto de teste, ou querer saber quais das quatro características são as mais discriminativas para distinguir uma espécie da outra.

### 1.1.1 Características ou Atributos

Temos então quatro características que descrevem uma flor, chamadas alternativamente de atributos. Para ser correto teríamos que distinguir entre a característica, por exemplo o comprimento da pétala e do valor da característica, por exemplo 5.1 cm. Então teria que usar símbolos diferentes, por exemplo  $\xi$  (“csi”) para falar da característica e  $x$  para falar do seu valor. Para facilitar, chamaremos ambos  $x$ . Como existem no total quatro características, temos então  $x_1, x_2, x_3, x_4$ . Isso são quatro números reais. Um número só é uma grandeza escalar. Uma flor possui essas quatro características simultaneamente, então somos obrigados a considerar tudo ao mesmo tempo. Isso chama-se um problema multidimensional. Isso também implica que temos que trabalhar com variáveis que são multidimensionais. Se agrupamos todos as quatro características em uma única variável  $\mathbf{x}$  multidimensional, uma flor descreve-se por um *vetor*<sup>3</sup> de características, cada uma sendo um escalar

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}.$$

A primeira iris do conjunto de Fisher descreve-se pelo vetor

$$\mathbf{x} = \begin{bmatrix} 5.1 \\ 3.5 \\ 1.4 \\ 0.2 \end{bmatrix}.$$

A unidade (cm) não tem relevância, desde que sejam evitadas mistura de ordens de grandeza diferentes, por exemplo, metros e centímetros. Já reparou que o vetor  $\mathbf{x}$  está escrito em negrito e o escalar  $x$  não está. Certamente dá para ver que essa escrita vertical do vetor  $\mathbf{x}$  gasta muito espaço. Mas temos que assegurar, especialmente no contexto da álgebra linear que não haja inconsistências na multiplicação de vetores com matrizes. Uma maneira para economizar espaço é sempre escrever horizontalmente  $[x_1 \ x_2 \ x_3 \ x_4]^T$  e eventualmente usar o operador da transposição “ $T$ ” que espelha uma matriz na diagonal. Na verdade, um vetor é um caso especial de uma matriz  $n \times m$  com  $n$  linhas e  $m$  colunas, ou com uma única coluna, sendo no caso da flor uma matriz  $4 \times 1$  ou com uma única linha, uma matriz  $1 \times 4$ , se transposto.

Quando se acessa um conjunto de dados, ele normalmente representa cada amostra  $\mathbf{x}$  em uma linha. Então todas as flores formam 150 linhas, em que cada linha tem quatro colunas. Essa representação chama-se a matriz de dados  $X$ , no nosso caso ela tem dimensão  $n \times m = 150 \times 4$ .

---

<sup>3</sup>Um vetor desempenha um papel fundamental neste livro. Precisamos mais tarde estudar algumas propriedades importantes, como a norma de um vetor e o gradiente de um vetor.

Podemos então descrever todo o conjunto de flores como

$$X = \begin{bmatrix} 5.1 & 3.5 & 1.4 & 0.2 \\ 4.9 & 3.0 & 1.4 & 0.2 \\ \vdots & \vdots & \vdots & \vdots \\ 5.9 & 3.0 & 5.1 & 1.8 \end{bmatrix}. \quad (1.1)$$

A primeira linha é a primeira flor  $\mathbf{x}_1^T$  transposta e a última linha é a última flor  $\mathbf{x}_{150}^T$  transposta. Todos os valores das quatro características das 150 amostras podem ser consultadas em [61]. Nessa página também podem ser vistos as visualizações de todas as combinações possíveis de duas características. É recomendável investir um pouco tempo com esse conjunto de flores, pois será usado constantemente neste livro.

### 1.1.2 Classes

Se a única informação disponível sobre o nosso universo de flores fossem as características, somente seria possível fazer afirmações sobre a relação das flores entre si. Ou seja, sobre a estrutura dos dados<sup>4</sup>. Porém, existe mais informação, nomeadamente a qual categoria, ou classe, cada flor pertence. Existem três espécies no nosso conjunto, *Iris setosa*, *Iris versicolor* e *Iris virginica*. Precisamos de um *rótulo*  $y$  para cada classe. Se existem somente duas classes, costuma-se rotular a primeira classe como  $y = 1$  e a segunda classe como  $y = -1$ . Adicionalmente, uma das classes então é a classe positiva e a outra classe é a negativa. Essa divisão em duas classes torna possível a definição de uma *matriz de confusão* e critérios de desempenho como resultados. Voltaremos mais tarde a essa matriz.

Para o caso de mais que duas classes, simplesmente enumerar as classes como  $1, 2, \dots$  não é uma boa ideia. Isso ia afastar a terceira classe da primeira mais que a segunda classe da primeira. Se um classificador se baseia em distâncias, a simples enumeração introduz distâncias injustas. Cada classe deve ter a mesma distância da outra. Um esquema que garante essa equidistância é a representação do rótulo da classe como um vetor binário. Então “setosa” é representado por  $[1 \ 0 \ 0]^T$ , “versicolor” por  $[0 \ 1 \ 0]^T$  e “virginica” por  $[0 \ 0 \ 1]^T$ . A distância Euclidiana entre os rótulos é igualmente  $\sqrt{2}$ .

### 1.1.3 Vetores

Vamos neste ponto inserir alguma teoria essencial sobre vetores. Estamos em princípio sempre trabalhando no *espaço euclidiano*, um espaço vetorial de números reais com um produto escalar. Um cálculo extremamente importante e frequente é este *produto escalar* ou *produto interno* de dois vetores. Existem três maneiras de escrever o produto, usando o símbolo “.”, usando os *chevrons* “⟨..⟩”, ou, considerando o primeiro vetor, transposto, como matriz  $1 \times m$ , e o segundo vetor como matriz  $m \times 1$  como produto de duas matrizes. Lembre-se que um vetor é somente um caso especial de uma matriz, ou uma das suas linhas, ou uma das suas colunas.

**Definição 1.1.1 — Produto Escalar (Interno) de dois Vetores.**

$$\mathbf{a} \cdot \mathbf{b} = \langle \mathbf{a}, \mathbf{b} \rangle = \mathbf{a}^T \mathbf{b} = \sum_{j=1}^m a_j b_j = \|\mathbf{a}\| \|\mathbf{b}\| \cos \theta, \quad (1.2)$$

onde  $\theta$  é o ângulo entre os dois vetores. Dessa maneira podemos constatar que o produto escalar de um vetor com ele próprio é equivalente ao quadrado da sua norma quadrática.

---

<sup>4</sup>Na figura 1.2, todos os pontos teriam a mesma cor, e não existiriam os rótulos das classes “setosa”, “versicolor” e “virginica”.

**Definição 1.1.2 — Ortogonalidade de dois Vetores.** Dois vetores  $\mathbf{a}$  e  $\mathbf{b}$  são ortogonais um ao outro, se o seu produto escalar é zero.

$$\mathbf{a} \perp \mathbf{b} \iff \mathbf{a} \cdot \mathbf{b} = 0. \quad (1.3)$$

**Corolário 1.1.1 — Relação Norma Quadrática com Produto Escalar de Vetor com ele Próprio.**

$$\mathbf{x} \cdot \mathbf{x} = \mathbf{x}^T \mathbf{x} = \|\mathbf{x}\|^2 = \sum_{j=1}^m x_j^2 = x_1^2 + \dots + x_m^2. \quad (1.4)$$

**Definição 1.1.3 — Vetor Unitário.**

$$\hat{\mathbf{x}} = \frac{\mathbf{x}}{\|\mathbf{x}\|} \quad (1.5)$$

O vetor unitário<sup>a</sup>  $\hat{\mathbf{x}}$  tem a mesma direção que o vetor original  $\mathbf{x}$  e tem módulo um, ou seja  $\|\hat{\mathbf{x}}\| = 1$ .

<sup>a</sup>Nesta definição o símbolo do chapéu é usado para denominar um vetor unitário.

■ **Exemplo 1.1** Em um sistema tridimensional de coordenadas cartesianas temos três vetores unitários que formam a base do sistema, ou seja, os eixos. Eles são  $\mathbf{e}_1 = [1 \ 0 \ 0]^T$ ,  $\mathbf{e}_2 = [0 \ 1 \ 0]^T$  e  $\mathbf{e}_3 = [0 \ 0 \ 1]^T$ . ■

**Corolário 1.1.2 — Projeção de Vetor Paralelo a Vetor Unitário.** Seja  $\mathbf{v}$  um vetor e seja  $\hat{\mathbf{w}}$  um vetor unitário, e sejam  $\mathbf{v} \parallel \hat{\mathbf{w}}$ , isto é, os dois vetores são paralelos. O paralelismo implica que um vetor é um múltiplo do outro

$$\mathbf{v} \parallel \hat{\mathbf{w}} \iff \mathbf{v} = \alpha \hat{\mathbf{w}}, \quad \alpha \in \mathbb{R} \setminus \{0\}. \quad (1.6)$$

Se o múltiplo  $\alpha$  é positivo, os dois vetores apontam na mesma direção “ $\rightarrow, \rightarrow$ ”, e o ângulo entre eles é  $\angle(\mathbf{v}, \hat{\mathbf{w}}) = 0$ . Se  $\alpha$  é negativo, os dois vetores apontam na direção oposta “ $\rightarrow, \leftarrow$ ”, e o ângulo entre eles é  $\angle(\mathbf{v}, \hat{\mathbf{w}}) = \pi = 180^\circ$ . Então a projeção de  $\mathbf{v}$  em cima de  $\hat{\mathbf{w}}$  é a norma de  $\mathbf{v}$ , multiplicado pelo sinal do múltiplo  $\alpha$

$$\mathbf{v} \cdot \hat{\mathbf{w}} = \text{sgn}(\alpha) \|\mathbf{v}\| = \pm \|\mathbf{v}\|. \quad (1.7)$$

A prova é trivial com o ângulo zero, ou  $\pi$ , entre dois vetores paralelos e a norma do vetor unitário igual a um. Usando a definição 1.1.1 do produto escalar, temos

*Demonstração.*

$$\mathbf{v} \cdot \hat{\mathbf{w}} = \|\mathbf{v}\| \|\hat{\mathbf{w}}\| \cos \theta = \|\mathbf{v}\| \cdot 1 \cdot \pm 1 = \pm \|\mathbf{v}\|. \quad \blacksquare$$

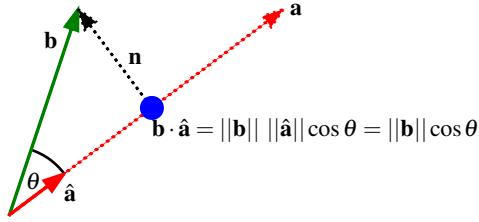


Figura 1.1: O produto escalar de dois vetores.

Na (eq. 1.7) foi usada a função do sinal que formalmente é

$$\text{sgn}(x) = \begin{cases} -1, & \text{se } x < 0 \\ 0, & \text{se } x = 0 \\ 1, & \text{se } x > 0 \end{cases} \quad (1.8)$$

Esta função tem um papel importante no contexto do aprendizado de máquina, por exemplo, pode ser usada como função de ativação não linear de um neurônio artificial.

**Proposição 1.1.3** O produto escalar tem dois papéis principais em aprendizado de máquina. Primeiramente, ele mede um semelhança entre dois objetos multidimensionais em forma de vetores. Se os dois vetores forem paralelos um ao outro, eles são semelhantes. Neste caso o produto escalar tem um valor máximo, pois o ângulo  $\theta$  entre eles é zero, então o seu cosseno é um, assim  $\cos 0 = \cos 0^\circ = 1$ . Se os dois vetores formarem um ângulo reto entre eles, ou seja, eles são ortogonais, eles são diferentes ao máximo. Neste caso o produto escalar é nulo, pois  $\cos \frac{\pi}{2} = \cos 90^\circ = 0$ .

A segunda função do produto interno é projetar um vetor em cima do outro. O resultado é um escalar, como o nome do produto já sugere, assim  $\mathbf{a} \cdot \mathbf{b} \in \mathbb{R}$ .

Um conceito fundamental na distinção de objetos é a distância. Se os nossos objetos forem descritos como vetores do mesmo espaço vetorial, podemos medir quão longe um objeto fica do outro. A Distância Euclidiana entre dois desses objetos é a mais importante.

**Definição 1.1.4 — Distância Euclidiana entre dois Vectors.**

$$D_E(\mathbf{a}, \mathbf{b}) = \sqrt{\sum_{j=1}^m (a_j - b_j)^2} = [(a_1 - b_1)^2 + \dots + (a_m - b_m)^2]^{1/2}, \quad (1.9)$$

onde  $\mathbf{a} = [a_1 \ a_2 \ \dots \ a_m]^T$  e  $\mathbf{b} = [b_1 \ b_2 \ \dots \ b_m]^T$  são dois vetores de dimensão  $m$ .

Essa é a distância mais importante entre dois vetores e é usada por padrão.

■ **Exemplo 1.2** A distância Euclidiana entre a primeira e segunda flor em um espaço 4-dimensional é então

$D_E(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{(5.1 - 4.9)^2 + (3.5 - 3.0)^2 + (1.4 - 1.4)^2 + (0.2 - 0.2)^2} = 0.53852$ , considerando a matriz de dados das flores (eq. 1.1). ■

Fortemente relacionada com a distância Euclideana é a *norma quadrática* ou *norma Euclidiana* ou *comprimento* ou *módulo* de um vetor  $\mathbf{x} \in \mathbb{R}^m$ .

**Definição 1.1.5 — Norma Quadrática de um Vetor  $\mathbf{x}$ , ou norma- $\ell_2$ .**

$$\|\mathbf{x}\|_2 = \sqrt{\sum_{j=1}^m x_j^2}. \quad (1.10)$$

Sendo a mais usada, em geral pode-se omitir o atributo “2” para definir a norma, ou seja,  $\|\mathbf{x}\|_2 = \|\mathbf{x}\|$ . Para facilitar a escrita, muitas vezes se usa o quadrado da norma em vez da própria norma, pois isso evita a necessidade da raiz quadrada, ou seja,  $\|\mathbf{x}\|_2^2 = \sum_{j=1}^m x_j^2$ . Consequentemente, usando o quadrado dos dois lados da equação, a distância Euclidiana é equivalente a norma do vetor diferença.

**Corolário 1.1.4 — Relação Distância Euclidiana Norma Quadrática.**

$$\begin{aligned} D_E^2(\mathbf{a}, \mathbf{b}) &= \|\mathbf{a} - \mathbf{b}\|^2 = \mathbf{a} \cdot \mathbf{a} - 2\mathbf{a} \cdot \mathbf{b} + \mathbf{b} \cdot \mathbf{b} \\ &= a_1^2 + \dots + a_m^2 - 2a_1b_1 - \dots - 2a_mb_m + b_1^2 + \dots + b_m^2 \\ &= a_1^2 - 2a_1b_1 + b_1^2 + \dots + a_m^2 - 2a_mb_m + b_m^2 \\ &= (a_1 - b_1)^2 + \dots + (a_m - b_m)^2. \end{aligned} \quad (1.11)$$

A generalização da norma quadrática é a norma- $\ell_p$ ,

$$\|\mathbf{x}\|_p = \left( \sum_{j=1}^m |x_j|^p \right)^{\frac{1}{p}}, \quad (1.12)$$

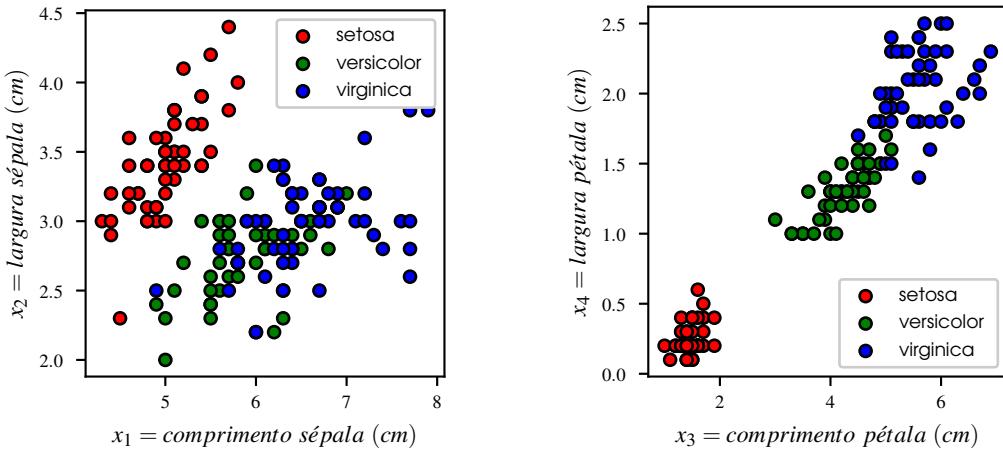
com os casos mais importantes da norma- $\ell_1$ ,

$$\|\mathbf{x}\|_1 = \sum_{j=1}^m |x_j| \quad (1.13)$$

e a norma- $\ell_\infty$

$$\|\mathbf{x}\|_\infty = \max_{j=1}^m (|x_j|). \quad (1.14)$$

■ **Exemplo 1.3** A figura 1.1 mostra o efeito do produto escalar entre dois vetores  $\mathbf{a} = \begin{bmatrix} 4 \\ 3 \end{bmatrix}$  e  $\mathbf{b} = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$ . Não vamos entrar demasiadamente nas definições axiomáticas de vetores. Basta saber que se pode somar dois vetores, multiplicar por um escalar, calcular o produto interno e diádico (veja mais tarde). Vetores são caracterizados pela sua direção, ou seja, a relação dos seus  $n$  componentes e seu módulo (eq. 1.10). O vetor de velocidade de um avião é um vetor tridimensional  $\mathbf{v}(t) = [v_x(t) \ v_y(t) \ v_z(t)]^\top$  e a velocidade do avião corresponde ao módulo  $\|\mathbf{v}(t)\|$  deste vetor. A norma quadrática do exemplo acima  $\mathbf{a}$  é calculada como  $\|\mathbf{a}\| = \sqrt{4^2 + 3^2} = \sqrt{16 + 9} = \sqrt{25} = 5$ . Consequentemente o vetor unitário é  $\hat{\mathbf{a}} = \frac{1}{5}\mathbf{a} = \begin{bmatrix} 4/5 \\ 3/5 \end{bmatrix}$ . O produto escalar (eq. 1.2) dos dois vetores é  $\mathbf{a} \cdot \mathbf{b} = 4 \times 1 + 3 \times 3 = 13$ . O produto escalar de  $\mathbf{b}$  com o vetor unitário  $\hat{\mathbf{a}}$  é  $\hat{\mathbf{a}} \cdot \mathbf{b} = 13/5 = 2.6$ . Essa é exatamente a distância que temos que percorrer ao longo de  $\mathbf{a}$  para chegar no ponto da projeção ortogonal de  $\mathbf{b}$  em cima de  $\mathbf{a}$ . A projeção ortogonal define o *vetor normal*  $\mathbf{n}$  em relação a  $\mathbf{a}$ . O produto escalar de dois vetores que são ortogonais é zero. O vetor  $\mathbf{n}$  pode ser facilmente calculado



(a) Selecionando as duas características  $x_1$  e  $x_2$  é desfavorável para discernir as três classes.

(b) Selecionando as duas características  $x_3$  e  $x_4$  é melhor para discernir as três classes.

Figura 1.2: Duas características definem um espaço bidimensional visualizável em um sistema de coordenadas cartesiano.

como diferença entre  $\mathbf{b}$  e uma versão escalada do vetor unitário  $\hat{\mathbf{a}}$ . O multiplicador é exatamente o escalar gerado pelo produto de  $\hat{\mathbf{a}}$  e  $\mathbf{b}$ . Então  $\mathbf{n} = \mathbf{b} - (\mathbf{b} \cdot \hat{\mathbf{a}})\hat{\mathbf{a}}$ , em números  $\mathbf{n} = \begin{bmatrix} 1 \\ 3 \end{bmatrix} - 13/5 \begin{bmatrix} 4/5 \\ 3/5 \end{bmatrix} = 1/25 \begin{bmatrix} -27 \\ 36 \end{bmatrix} = \begin{bmatrix} -1.08 \\ 1.44 \end{bmatrix}$ .

O conceito do vetor normal é capaz de definir um classificador binário, dividindo o espaço vetorial em duas partes complementares. Precisamos somente  $m + 1$  números para definir esse *hiperplano* em um espaço de  $m$  dimensões. Novamente, paciência, os conceitos serão explicados no contexto de *aprendizagem supervisionada* usando classificadores lineares.

#### 1.1.4 Visualização do Espaço das Características

Rapidamente chegamos no limite se quisermos visualizar uma flor, onde cada característica é um eixo em um sistema de coordenadas cartesiano [102]. Não é possível mostrar um sistema de quatro coordenadas diretamente. Consequentemente temos que limitar a quantidade de características para duas ou três. Essa redução do número de características chama-se *seleção de características* e merece um capítulo próprio. Para já, fazemos a seleção manualmente. Descartamos duas das quatro características para ficarmos com duas. Na matriz de dados (eq. 1.1) isso significa, eliminar duas colunas, restando uma matriz de dimensão  $150 \times 2$ . Agora é possível visualizar todas as flores em um gráfico bidimensional. A figura 1.2 mostra dois gráficos. Do lado esquerdo, na figura 1.2a, o sistema de coordenadas cartesiano tem como eixo  $x$  a característica  $x_1$ , comprimento da sépala e, como eixo  $y$  a característica  $x_2$ , largura da sépala. Do lado direito, na figura 1.2b, a característica  $x_3$ , comprimento da pétala como eixo  $x$ , e a característica  $x_4$ , largura da pétala, como eixo  $y$ .

Contando os pontinhos de cada classe, percebe-se que são menos que 50 o que corresponda ao número de padrões de cada uma das três classes. A razão é que pela desconsideração de duas das quatro características, surgiram padrões com o mesmo valor em todas as características. Assim, em uma única posição do gráfico várias flores foram sobrepostas uma em cima da outra, possivelmente de classes diferentes e somente a última classe aparece.

A comparação dos dois gráficos ensina alguns fatos fundamentais sobre o poder discriminativo de um conjunto de características:

1. Percebe-se claramente que, para fins de classificação, o par de características  $x_1$  e  $x_2$  da figura 1.2a, é pior que o par de características  $x_3$  e  $x_4$  da figura 1.2b, portanto existem algumas características melhores que outras;
2. Mesmo com as características melhores, as duas classes versicolor e virginica parecem ter uma região no espaço de características onde algumas flores se sobrepõem. Isso significa que podem-se esperar erros em um classificador que trabalha com estas características. Existe incerteza.

O objetivo não será eliminar essa incerteza, pois isso é impossível com este conjunto de dados. Isso também é verdade com um reconhecedor de pedestres em um contexto de um sistema de veículo autônomo. O objetivo é *minimizar* a incerteza. Sistemas baseados em aprendizagem de máquina sempre têm que conviver com essa incerteza, sempre um programador tem que definir um “número mágico”, acima do qual vale uma situação e abaixo do qual vale outra. Nunca haverá perfeição em sistemas inteligentes.

### 1.1.5 Padrões

O quê são as 150 flores no conjunto de dados Iris? São os *padrões* que normalmente são usados para aprender o objetivo do problema em questão. Neste caso é a classificação das três espécies. Como os dados foram coletadas em 1935, é pouco provável que teremos mais padrões da mesma fonte a nossa disposição. Temos que *estimar* todos os critérios de qualidade do sistema aprendido, por exemplo quantos erros o classificador aprendido vai cometer. Se existem somente 150 padrões, temos que nos basear nesses dados, por exemplo usando uma técnica de *validação cruzada*. De fato, cada característica de uma flor equivale a uma *variável aleatória*. Como esse assunto é fundamental, segue uma breve introdução para o caso especial de uma variável aleatória contínua, ou seja, uma variável que pode assumir um valor numérico qualquer.

## 1.2 Variáveis Aleatórias

A voltagem de uma tomada é uma variável aleatória  $x$ . Na maioria dos casos no Brasil afirma-se que é 127 V. Mas, se a gente mede, nunca dá exatamente esse valor. Existe ruído, e esse ruído faz que o valor varie aleatoriamente. O valor 127 V é o *valor esperado*  $\mathbb{E}[x]$ . O sinônimo de valor esperado é a *média*  $\mu$  “miú” da variável aleatória. Ninguém sabe o valor esperado exato da voltagem de uma tomada de eletricidade. É fruto de um processo caótico. Consumo, tempo, até a radiação do sol influenciam esse valor, no sentido de um efeito borboleta [30]. Consequentemente 127 V na verdade é um valor aproximado do verdadeiro valor esperado. Ao contrário das flores de 1935, temos a possibilidade de adquirir amostras a vontade para fazer afirmações sobre a voltagem. Como é que se poderia tentar obter o valor esperado? O mais óbvio é adquirir uma grande quantidade de  $n$  amostras da voltagem, e calcular a *média aritmética*. Isso realmente é o caminho. O valor esperado estimado é a média de um *conjunto de treinamento*, ou seja, somar todos os valores disponíveis e dividir a soma pela quantidade dos valores disponíveis. Posteriormente vamos expandir esses conceitos para definir variância, covariância e densidade probabilística. Consequentemente, no nosso exemplo preferido, a largura da sépala e todos as outras características são variáveis aleatórias. É um número, um escalar. Agrupando todas as características, teremos uma variável aleatória multidimensional, um vetor. Um escalar ou um vetor obedecem a uma *distribuição de probabilidade*.

Todas as aplicações relacionadas com aprendizagem de máquina representam o seu universo com a ajuda de variáveis aleatórias. Um programa que tenta reconhecer uma face humana em uma imagem digital tem como informação original um conjunto de pontos da imagem. A menor unidade da imagem é o *píxel*, que normalmente é representada por uma tripla de valores, por exemplo no modelo RGB. Cada valor é uma dessas variáveis cujo valor obedece um comportamento

estatístico. O conjunto todo desses píxeis descrevem algum objeto ou situação. Como a imagem vai ser processada depende do objetivo final, por exemplo, localizar onde na imagem aparece uma face. Outro exemplo é um processo industrial onde variáveis são usadas para controlar o processo ou para medir o estado do processo, por exemplo a pressão e a temperatura. E também as quatro características das nossas flores são variáveis aleatórias. A vantagem desses comprimentos e larguras das sépalas e pétalas é que podem ser usados diretamente em uma tarefa de classificação. No exemplo do reconhecimento de faces em uma imagem, isso não é possível. Somente após uma sequência de outros processamentos, criando novas variáveis, essas novas características são apropriadas para fazer a classificação.

As definições a seguir tentam formalizar os conceitos suficientemente, sem serem exageradamente formais. Embora tenha-se apresentado até agora a variável aleatória contínua, vamos ver também a variável aleatória discreta.

### 1.2.1 Variáveis Aleatórias Contínuas

Uma variável aleatória contínua  $x$  em geral é um número real  $x \in \mathbb{R}$  e obedece a uma distribuição de probabilidade  $p(x)$  em forma de função, a sua *função densidade de probabilidade* (FDP). Além do acrônimo FDP, usaremos também a denominação mais compacta *densidade*. A densidade adicionalmente é caracterizada pelos seus *parâmetros*. Normalmente se usa o símbolo  $\theta$  (“teta”) para representar os parâmetros. Assim  $\theta$  é uma tupla que abriga todos os parâmetros necessários para definir a densidade,  $\theta = (\theta_1, \theta_2, \dots)$ . A densidade completamente definida então é

$$p(x; \theta), \quad (1.15)$$

ou seja, o argumento é a variável aleatória  $x$  e, separado por ponto e vírgula, todos os parâmetros estão em  $\theta$ . Usa-se o símbolo “~”, junto com a identificação da distribuição para expressar a natureza da variável aleatória.

A densidade mais simples é a densidade uniforme  $p(x; a, b)$ . A variável  $x$  pode ter um valor entre um limite inferior  $a$  e um limite superior  $b$ . Os limites são os parâmetros da densidade, ou seja,  $\theta = (a, b)$ . Nenhum valor dentro desse intervalo  $[a, b]$  tem preferência sobre um outro valor, ou seja, a densidade é constante no intervalo. Fora do intervalo a densidade é zero, lá nunca pode aparecer uma amostragem. A densidade uniforme é definida como

**Definição 1.2.1 — Densidade Uniforme, Variável Aleatória Unidimensional.**

$$p(x; a, b) = \begin{cases} \frac{1}{b-a}, & \text{se } x \in [a, b] \\ 0, & \text{caso contrário.} \end{cases} \quad (1.16)$$

Expressa-se o fato que a variável aleatória  $x$  segue a distribuição uniforme como

$$x \sim U(a, b). \quad (1.17)$$

A densidade mais importante no nosso contexto é a densidade normal, ou Gaussiana,  $p(x; \mu, \sigma^2)$ , ( $\sigma$  = “sigma”). Muitas variáveis aleatórias na natureza e em ambientes técnicos obedecem a esta densidade, por exemplo, a voltagem de uma tomada que já foi mencionada anteriormente. A Gaussiana tem a forma de um sino com um único máximo em  $x = \mu$ , decaindo simetricamente, quando se afasta do máximo. Frequentemente, em vez da Gaussiana, usamos o seu logaritmo natural, mostrado aqui também.

**Definição 1.2.2 — Densidade Normal, Variável Aleatória Contínua Unidimensional.**

$$p(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\} \quad (1.18)$$

$$\ln p(x; \mu, \sigma^2) = -\frac{1}{2} \left[ 2\ln\sigma + \ln 2 + \ln\pi + \frac{(x-\mu)^2}{\sigma^2} \right]. \quad (1.19)$$

Expressa-se o fato que a variável aleatória  $x$  segue a distribuição normal como

$$x \sim N(\mu, \sigma^2). \quad (1.20)$$

As densidades uniforme e normal são ilustradas na figura 1.3, usando os parâmetros  $\theta = (a, b) = (0, 1)$  para a densidade uniforme e  $\theta = (\mu, \sigma^2) = (0, 1)$  para a densidade Gaussiana.

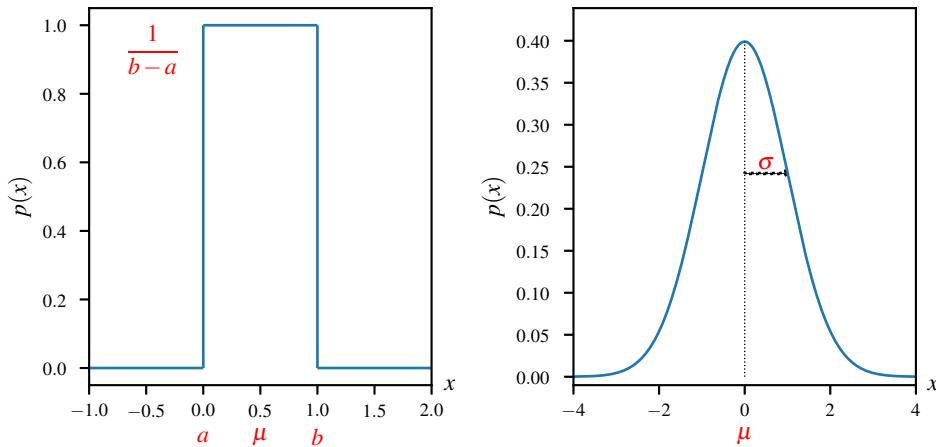


Figura 1.3: Distribuições uniforme e Gaussiana.

Vamos simplificar o problema, considerando somente uma única característica. Posteriormente estudaremos densidades multidimensionais, ou seja, onde a variável não é um escalar  $x$ , mas um vetor  $\mathbf{x}$ . Considere então uma única característica da setosa, da versicolor e da virgíñica. Qual a densidade delas? A resposta é: Não se sabe. Existem métodos de classificação que não precisam conhecer as densidades. Um exemplo muito importante é o classificador Vizinho-Mais-Próximo que pertence ao grupo dos classificadores não paramétricos e será apresentado posteriormente. Também existem métodos de classificação que precisam saber qual é a densidade e os seus parâmetros  $p(x; \theta)$ , definindo os classificadores paramétricos. Usando a regra de Bayes<sup>5</sup>, é possível calcular uma probabilidade de uma flor pertencer a uma das três classes.

Existem dois problemas fundamentais relacionados ao uso de classificadores paramétricos:

1. Pode-se *assumir* um densidade. A consequência de assumir é o perigo de errar. Se dados são amostras de uma densidade, e se um classificador assume outra densidade, temos um *erro de modelo*.

<sup>5</sup>Um exemplo da aplicação da Regra de Bayes para calcular a probabilidade que uma flor  $\mathbf{x}$  pertencer a classe setosa é  $P(\text{setosa}|\mathbf{x}) = \frac{p(\mathbf{x}|\text{setosa})P(\text{setosa})}{p(\mathbf{x})}$ , onde  $P(\text{classe}|\text{características})$  é uma probabilidade condicional,  $P(\text{classe})$  é a probabilidade a priori de encontrar uma flor de uma determinada classe, sem conhecimento prévio,  $p(\text{características}|\text{classe})$  é uma densidade condicional, e  $p(\text{características})$  é uma densidade, independente das classes. Vamos estudar os conceitos no contexto dos classificadores paramétricos.

2. Mesmo se acertamos a densidade, resta a tarefa de obter os valores dos seus parâmetros. Especialmente com poucos dados a nossa disposição para aprender, os valores nesta tarefa de *estimativa de parâmetros* podem desviar bastante dos seus valores verdadeiros.

### 1.2.2 Variáveis Aleatórias Discretas

O exemplo clássico de uma variável aleatória discreta até deu origem a palavra “aleatório”. *Alea iacta est* (o dado foi lançado, a sorte foi lançada) se refere ao processo estatístico (o dado) que produz seis valores diferentes. Também valores não numéricos, ou seja valores simbólicos são possíveis para uma variável discreta. Por exemplo, a variável  $C$  que expressa o estado do tempo poderia assumir os valores “ensolarado”, “nublado” e “chuvisco”. Podemos atribuir uma probabilidade ou um outro valor quantitativo a cada valor de uma característica. Para diferenciar uma densidade  $p$  de uma variável aleatória contínua, usa-se a letra maiúscula  $P$ . Assim, em uma tarefa de classificação,  $P(C)$  expressa, sem ter qualquer outra informação sobre um padrão, a *probabilidade a priori* da classe  $C$ . Posteriormente vamos estudar tipicamente a *probabilidade condicional*  $P(C|x)$  que expressa quanto certo um padrão  $x$  pertence a uma classe  $C$ , por exemplo,  $P(\text{setosa} | [4.9 \ 3.0 \ 1.4 \ 0.2]^\top) = 98.3\%$ .

### 1.2.3 Valor Esperado, ou Média de uma Variável Aleatória

Já vimos que aproximadamente 127 V é o valor esperado da voltagem de uma tomada, que é uma variável aleatória contínua. Vamos estudar as definições formais e alguns exemplos.

**Definição 1.2.3 — Valor Esperado, ou Média, Variável Aleatória Discreta.**

$$\mathbb{E}[x] = \mu = \sum_{i=1}^N x_i P(x_i) = x_1 P(x_1) + x_N P(x_N) + \dots + x_N P(x_N), \quad (1.21)$$

em que  $N$  é o número dos valores distintos  $x_i$  que a variável pode ter.

■ **Exemplo 1.4** O valor esperado de um dado é  $3.5 = 1 \cdot \frac{1}{6} + 2 \cdot \frac{1}{6} + 3 \cdot \frac{1}{6} + 4 \cdot \frac{1}{6} + 5 \cdot \frac{1}{6} + 6 \cdot \frac{1}{6}$ , um resultado inexistente e certamente inesperado. ■

**Definição 1.2.4 — Valor Esperado, ou Média, Variável Aleatória Contínua.**

$$\mathbb{E}[x] = \mu = \int_{-\infty}^{\infty} x p(x) dx, \quad (1.22)$$

em que  $p(x)$  é a densidade da variável para o valor  $x$  e  $dx$  é o intervalo infinitesimal em torno de  $x$ .

Essa definição vale para uma variável  $x$  unidimensional, como a largura da sépala, mas também para a variável aleatória multidimensional  $\mathbf{x}$  que representa todas as quatro características das nossas flores. Neste caso a densidade é uma função em um espaço de quatro dimensões  $p(\mathbf{x}) = p(x_1, x_2, x_3, x_4)$  e  $d\mathbf{x}$  é um *hypervolume* neste espaço. Em uma, duas e três dimensões ainda é relativamente fácil entender  $d\mathbf{x}$ , onde é um intervalo, quadrado e volume muito pequeno, respectivamente.

Não se preocupe com esta definição, pois na prática o cálculo analítico definido em (eq. 1.22) não será feito. Primeiramente, precisaria a correta definição da densidade, junto com os seus verdadeiros valores dos parâmetros. Imagine uma rede neural profunda<sup>6</sup> com milhares de pesos, que

<sup>6</sup>Uma rede neural artificial profunda é uma sequência de funções que calculam na primeira camada uma função das variáveis de entrada, na segunda camada, uma função da função anterior, e assim por diante, até calcular a saída como função dos valores da camada anterior.

são os parâmetros de uma densidade. A definição em *forma fechada*<sup>7</sup> é inacessível, e a integração teria que ser feita sobre milhares de dimensões. Para algumas densidades importantes é relativamente fácil calcular o valor esperado. Por exemplo, para a Gaussiana (eq. 1.18) temos  $\mathbb{E}[x] = \mu$  (sem prova, exercício para o leitor). Isso é um fato notável, pois o valor esperado desta densidade é simultaneamente um parâmetro que define a densidade.

■ **Exemplo 1.5** Para a densidade uniforme de uma variável (eq. 1.16) até dá para aceitar o desafio e calcular  $\mu$ . Sabendo que a densidade é constante  $p(x; a, b) = 1/(b - a)$ , e que ela é zero fora do intervalo  $[a, b]$ , e que  $a^2 - b^2 = (a + b)(a - b)$ , temos para o valor esperado (eq. 1.22)

$$\mathbb{E}[x] = \int_{\mathbb{R}} xp(x)dx = \frac{1}{b-a} \int_a^b x dx = \frac{1}{b-a} \left[ \frac{x^2}{2} \right]_a^b = \frac{a+b}{2}.$$

■

### 1.2.4 Momentos de uma Variável Aleatória

As definições dos valores esperados para os casos de uma variável aleatória discreta (eq. 1.21) e contínua (eq. 1.22) são casos especiais de primeira ordem. O conceito<sup>8</sup> geral é o *momento de ordem n*. Vamos nos limitar às variáveis aleatórias contínuas.

■ **Definição 1.2.5 — Momento de ordem  $n \in \mathbb{N}$ , Variável Aleatória Contínua.**

$$\mu_n \stackrel{\text{def}}{=} \mathbb{E}[x^n] = \int_{-\infty}^{\infty} x^n p(x)dx, \quad (1.23)$$

Se adicionalmente, da variável é subtraído a sua *média*  $\mu$  que corresponde ao valor esperado de ordem um, temos o

■ **Definição 1.2.6 — Momento central de ordem  $n \in \mathbb{N}$  em torno da média.**

$$\mu_n \stackrel{\text{def}}{=} \mathbb{E}[(x - \mathbb{E}[x])^n] = \int_{-\infty}^{\infty} (x - \mu)^n p(x)dx, \quad (1.24)$$

Podem-se considerar esses momentos como características novas. Em geral, esse processo de obter características novas a partir de variáveis originais chama-se *extração de características* e é a própria essência de aprendizado de máquina.

O momento central de ordem dois desempenha um papel muito importante em vários assuntos relacionados ao tema de aprendizagem de máquina. Define a

■ **Definição 1.2.7 — Variância.**

$$\sigma^2 = \mu_2 \stackrel{\text{def}}{=} \mathbb{E}[(x - \mathbb{E}[x])^2] = \mathbb{E}[(x - \mu)^2] = \int_{-\infty}^{\infty} (x - \mu)^2 p(x)dx, \quad (1.25)$$

A variância mede como uma variável aleatória se afasta da sua média, em média, calculando a distância  $x - \mathbb{E}[x]$ . Para garantir que a média dessas distâncias seja sempre positiva, calculase o quadrado. Repare que para calcular esses momentos seria necessário ter a densidade  $p(x)$  na mão, o que não é realista. Ainda não temos as ferramentas de *estimar* média e variância, ou outros valores estatísticos a partir de um número finito de padrões. Porém será sempre uma

<sup>7</sup>Um exemplo de uma densidade em forma fechada de uma variável unidimensional seria  $p(x; \theta_1, \theta_2) = \theta_1 e^{-\theta_2 x}$

<sup>8</sup>A ordem tem o papel de uma potência. Isso fica mais claro no exemplo da variância, pois calcula-se o quadrado. Outro exemplo é a *curtose*, que é um momento de quarta ordem. Na sua definição aparecem a quarta potência.

necessidade básica de se contentar com um conjunto que não seja infinito para treinar um sistema aprendizado de máquina. Frequentemente uma alternativa no uso da variância é o desvio padrão que simplesmente é a raiz quadrada da variância.

**Definição 1.2.8 — Desvio Padrão.**

$$\sigma \stackrel{\text{def}}{=} \sqrt{\sigma^2} \quad (1.26)$$

## 1.3 Estimativa de Parâmetros de Variáveis Aleatórias

No mundo de aprendizado de máquina, em princípio todo o conhecimento está engessado em variáveis aleatórias. Ou temos que trabalhar com a própria variável  $x$ , ou temos que trabalhar com outras variáveis aleatórias que são relacionadas com  $x$ . Por exemplo, a largura da pétala é uma variável que muda o seu valor (o próprio nome “variável” diz isso), e então tem um comportamento estatístico. Assim ela possui uma média (valor esperado, momento de primeira ordem), uma variância (momento central de segunda ordem). Todas as características formam o vetor de características de uma variável aleatória multidimensional que também possui esses momentos, por exemplo a média  $\mu$  é também um vetor que em cada componente tem a média de cada característica particular.

Outro conceito fundamental é uma *função de perda* ou *função de custo*, ou *função objetivo*, ou simplesmente *erro*, que mede em um problema de otimização a discrepância entre o que se quer e o que o sistema responde. Uma vez definida como o sistema calcula a sua saída  $\hat{y}$  para uma entrada  $x$ , é possível medir a discrepância entre a saída desejada  $y$  e a saída calculada  $\hat{y}$ . O sistema precisa definir parâmetros  $\theta$  no cálculo da sua saída. A função de perda mais conhecida é o valor esperado do quadrado da diferença (em caso de variável unidimensional), ou norma quadrática do vetor diferença, ainda dividida pela dimensão da variável (em caso de variável multidimensional). Todos esses números são variáveis aleatórias. Especialmente a função de perda é impossível ser modelada por uma densidade. A minimização dessa função é consequentemente impossível ser feita analiticamente. A minimização será feita pelo coração do aprendizado de máquina, a *descida de gradiente*. Pormenores adiante.

### 1.3.1 Estimando a Média

Sem ser especialista em estatística, todo mundo sabe estimar a média de um parâmetro de uma população. A média de idade dos estudantes de uma sala de aula é estimada somando todas as idades e dividindo pela quantidade dos alunos.

Temos que distinguir entre dois números que facilmente são confundíveis:

1. A verdadeira média  $\mu$  de uma densidade  $p(x)$  de uma variável aleatória  $x$ .
2. A média estimada  $\hat{\mu}$  de um número finito de  $n$  amostras.

Existem até densidades sem uma média definida, por exemplo a distribuição probabilística de Cauchy, mas neste livro não vamos tão longe. A diferença entre o valor verdadeiro e o estimado de um parâmetro é o *viés*, “bias” em inglês. Um bom *estimador* tenta minimizar o viés. Um estimador é uma forma fechada, ou um algoritmo que tenta obter uma estimativa de um parâmetro estatístico. Existe então uma fórmula ou um método interativo para calcular essa estimativa. No caso da média temos a forma fechada

**Definição 1.3.1 — Média Estimada de um Conjunto de Treino.**

$$\hat{\mu} = \widehat{\mathbb{E}}[x] = \bar{x} = \frac{1}{n} \sum_{i=1}^n x_i. \quad (1.27)$$

O símbolo do chapéu “ $\widehat{\phantom{x}}$ ” significa que não se sabe o valor, mas tenta-se o melhor para estimar o valor, usando a informação a disposição. No caso de aprendizado de máquina, essa informação é o conjunto de treinamento que contém um número finito de amostras. O símbolo do sobrelinhado “ $\underline{\phantom{x}}$ ” frequentemente também é usado.

■ **Exemplo 1.6** Vamos considerar a segunda característica “largura da sépala”. A média estimada desta característica, usando todos os 150 amostras independente da classe é a média aritmética da segunda coluna da matriz de dados (eq. 1.1).

$$\hat{\mu}_2 = \frac{1}{150} \sum_{i=1}^{150} X_{i,2} = (3.5 + 3.0 + \dots + 3.0) / 150 = 3.05 \dots$$

Note que o índice “2” significa a característica dois, e não a ordem do momento na definição (eq. 1.24) e (eq. 1.25) ■

### 1.3.2 Estimando a Variância

Dado um conjunto de  $n$  padrões  $x_i, i = 1, \dots, n$  a variância pode ser estimada como

**Definição 1.3.2 — Variância Estimada de um Conjunto de Treino.**

$$\hat{\sigma}^2 = \widehat{\mathbb{E}}[(x - \mu)^2] = \frac{1}{n-1} \sum_{i=1}^n (x_i - \hat{\mu})^2. \quad (1.28)$$

A estimativa da variância já usa um valor que é resultado de uma estimativa, nomeadamente a média estimada  $\hat{\mu}$ . A divisão da soma por  $(n-1)$ , em vez da esperada divisão por  $n$  tem o efeito de eliminar o viés (*bias*) da estimativa, pormenores em [8]. Esse ajuste é chamado *correção de Bessel* [7]. Na prática esse detalhe em princípio tem pouca relevância, especialmente se houverem muitos padrões de treinamento. Nesse caso, a divisão por  $(n-1)$  converge assintoticamente para a de  $n$ .

■ **Exemplo 1.7** Usando a mesma característica como no exemplo da estimativa da média, a variância estimada da largura da sépala de todas as flores é

$$\hat{\sigma}_2^2 = \frac{1}{149} \sum_{i=1}^{150} (X_{i,2} - \hat{\mu}_2)^2 = \quad (1.29)$$

$$[(3.5 - 3.05 \dots)^2 + (3.0 - 3.05 \dots)^2 + \dots + (3.0 - 3.05 \dots)^2] / 149 = 0.189979 \dots \quad (1.30)$$

Em comparação, a divisão por 150 daria 0.188712.... Note que o índice “2” significa a característica dois, e não a ordem do momento na definição (eq. 1.24) e (eq. 1.25). ■

### 1.3.3 Geração de Variáveis Aleatórias

Uma ferramenta importante em aprendizado de máquina é o gerador de números aleatórios que obedecem a uma densidade definida. É a única vez que sabemos os verdadeiros parâmetros  $\theta$  de uma distribuição, pois temos que especificar o seu valor na função que produz um número finito de amostras dessa distribuição. Frequentemente em trabalhos científicos, simulações são usadas

para ilustrar as ideias apresentadas, e essas simulações usam geradores de números aleatórios. Uma publicação de qualidade ainda por cima publica a *semente* da geração, pois isso converte a aleatoriedade dos números de certa maneira em um conjunto determinístico. Um pesquisador que queira *reproduzir* algum experimento aleatório de um outro pesquisador, com o conhecimento da semente é capaz de obter exatamente a mesma sequência de números. Isso permite a comparação justa do próprio trabalho com o dos pares na comunidade científica.

Praticamente todas as linguagens de programação disponibilizam as funções necessárias para gerar uma sequência de variáveis aleatórias. Por exemplo, na linguagem Python, usando o pacote numérico `numpy`, a chamada `numpy.random.uniform(a, b)` gera um número da densidade uniforme (eq. 1.16), e a chamada `numpy.random.normal(media, desvio_padrao)` um número da densidade Gaussiana (eq. 1.18). A chamada `numpy.random.seed(seed=66649)` define a semente do gerador com o valor 66649 que, a partir dessa chamada, gera sempre a mesma sequência de números. Porém essa sequência ainda obedece a densidade escolhida.

## 1.4 Variáveis Aleatórias Multidimensionais

O leitor já tem consciência que um padrão mais apropriadamente deve ser descrito como um vetor de variáveis aleatórias, como na descrição de uma flor por suas quatro características. A extensão das definições em uma dimensão criam novos parâmetros que não existem em uma dimensão, por exemplo a covariância. Como o vetor média de um vetor de variáveis aleatórias em cada componente simplesmente tem a média dessa componente, a definição é facilmente estendível. Vamos nos limitar a variáveis aleatórias contínuas, então uma densidade de uma variável aleatória multidimensional  $\mathbf{x}$  com todos os parâmetros  $\boldsymbol{\theta}$  que controlam o aspecto da densidade é

$$p(\mathbf{x}; \boldsymbol{\theta}). \quad (1.31)$$

**Definição 1.4.1 — Valor Esperado, Variável Aleatória Contínua Multidimensional.**

$$\mathbb{E}[\mathbf{x}] = \boldsymbol{\mu} = \int_{\mathbb{R}^m} \mathbf{x} p(\mathbf{x}) d\mathbf{x}, \quad (1.32)$$

em que  $p(\mathbf{x})$  é a densidade da variável para o valor  $\mathbf{x}$  e  $d\mathbf{x}$  é o intervalo infinitesimal em torno de  $\mathbf{x}$ .

Como já foi mencionado, esse cálculo em aplicações práticas de aprendizado de máquina não vai aparecer, ao contrário da estimativa deste valor.

**Definição 1.4.2 — Média Multidimensional Estimada de um Conjunto de Treino.**

$$\hat{\boldsymbol{\mu}} = \hat{\mathbb{E}}[\mathbf{x}] = \bar{\mathbf{x}} = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_j \\ \vdots \\ \mu_m \end{bmatrix} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i = \frac{1}{n} \begin{bmatrix} \sum_{i=1}^n X_{i,1} \\ \vdots \\ \sum_{i=1}^n X_{i,j} \\ \vdots \\ \sum_{i=1}^n X_{i,m} \end{bmatrix}, \quad (1.33)$$

onde  $X_{i,j}$  é a  $j$ -ésima característica do  $i$ -ésimo padrão e  $\mu_j$  é a média estimada (eq. 1.27) da  $j$ -ésima característica.

■ **Exemplo 1.8** Podemos calcular a média global  $\bar{\mathbf{x}}$  dos  $n = 150$  padrões de todas as três classes, ou a média específica de cada classe com  $n_k = 50$  padrões, para  $k = 1, 2, 3$ , usando a matriz de

dados  $X$  da (eq. 1.1) .

$$\begin{aligned}\bar{\mathbf{x}} &= \frac{1}{150} \begin{bmatrix} \sum_{i=1}^{150} X_{i,1} \\ \sum_{i=1}^{150} X_{i,2} \\ \sum_{i=1}^{150} X_{i,3} \\ \sum_{i=1}^{150} X_{i,4} \end{bmatrix} = \begin{bmatrix} 5.843 \\ 3.057 \\ 3.758 \\ 1.199 \end{bmatrix} & \bar{\mathbf{x}}_{\text{setosa}} &= \frac{1}{50} \begin{bmatrix} \sum_{i=1}^{50} X_{i,1} \\ \sum_{i=1}^{50} X_{i,2} \\ \sum_{i=1}^{50} X_{i,3} \\ \sum_{i=1}^{50} X_{i,4} \end{bmatrix} = \begin{bmatrix} 5.006 \\ 3.428 \\ 1.462 \\ 0.246 \end{bmatrix} \\ \bar{\mathbf{x}}_{\text{versicolor}} &= \frac{1}{50} \begin{bmatrix} \sum_{i=51}^{100} X_{i,1} \\ \sum_{i=51}^{100} X_{i,2} \\ \sum_{i=51}^{100} X_{i,3} \\ \sum_{i=51}^{100} X_{i,4} \end{bmatrix} = \begin{bmatrix} 5.936 \\ 2.770 \\ 4.260 \\ 1.326 \end{bmatrix} & \bar{\mathbf{x}}_{\text{virginica}} &= \frac{1}{50} \begin{bmatrix} \sum_{i=101}^{150} X_{i,1} \\ \sum_{i=101}^{150} X_{i,2} \\ \sum_{i=101}^{150} X_{i,3} \\ \sum_{i=101}^{150} X_{i,4} \end{bmatrix} = \begin{bmatrix} 6.588 \\ 2.974 \\ 5.552 \\ 2.026 \end{bmatrix} \end{aligned} \quad (1.34)$$

■

Vamos conhecer métodos, por exemplo, a Análise de Componentes Principais que precisam deslocar a média dos padrões para a origem do sistema de coordenadas. Dada uma matriz de dados como a das nossas flores (eq. 1.1), o procedimento de centrar os dados subtrai de cada padrão da média.

#### Definição 1.4.3 — Dados Centrados, Variável Aleatória Contínua Multidimensional.

Seja  $\mathbf{x}$  uma variável aleatória de dimensão  $m$ , e seja  $\hat{\boldsymbol{\mu}}$  a sua média estimada (eq. 1.33). Seja

$$X = \begin{bmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_i^T \\ \vdots \\ \mathbf{x}_n^T \end{bmatrix} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,j} & \cdots & x_{1,m} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i,1} & \cdots & x_{i,j} & \cdots & x_{i,m} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{n,1} & \cdots & x_{n,j} & \cdots & x_{n,m} \end{bmatrix} \quad (1.35)$$

a matriz de dados de dimensão  $n \times m$  composta por  $n$  padrões  $\mathbf{x}_i$  de dimensão  $m$ ,  $i = 1, \dots, n$ . Cada padrão transposto  $\mathbf{x}_i^T = [x_{i,1} \ \cdots \ x_{i,j} \ \cdots \ x_{i,m}]$  forma uma linha de  $X$ , veja o exemplo  $150 \times 4$  das flores na (eq. 1.1). A matriz de dados centrada é composta pelos padrões centrados, onde cada padrão  $\mathbf{x}_i$  é substituído pela diferença do padrão e sua média  $\mathbf{x}_i - \hat{\boldsymbol{\mu}}$ , ou seja,

$$X_{\text{centrada}} = \begin{bmatrix} (\mathbf{x}_1 - \hat{\boldsymbol{\mu}})^T \\ \vdots \\ (\mathbf{x}_i - \hat{\boldsymbol{\mu}})^T \\ \vdots \\ (\mathbf{x}_n - \hat{\boldsymbol{\mu}})^T \end{bmatrix} = \begin{bmatrix} (x_{1,1} - \hat{\mu}_1) & \cdots & (x_{1,j} - \hat{\mu}_j) & \cdots & (x_{1,m} - \hat{\mu}_m) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ (x_{i,1} - \hat{\mu}_1) & \cdots & (x_{i,j} - \hat{\mu}_j) & \cdots & (x_{i,m} - \hat{\mu}_m) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ (x_{n,1} - \hat{\mu}_1) & \cdots & (x_{n,j} - \hat{\mu}_j) & \cdots & (x_{n,m} - \hat{\mu}_m) \end{bmatrix} \quad (1.36)$$

O efeito<sup>9</sup> dessa operação é que a nova média  $\hat{\boldsymbol{\mu}}_{\text{centrada}}$  é um vetor zero  $\mathbf{0}$  de dimensão  $m$ , isto é, a média estimada dos padrões centrados é deslocada para a origem. Em termos geométricos esta transformação é uma *translação*.

Vamos aproveitar a definição dos dados centrados para acrescentar um outro procedimento de normalização muito importante. O intervalo  $[x_j^{\min}, x_j^{\max}]$  em que a variável  $x_j$  se movimenta, pode ser extremamente diferente para as varáveis diferentes. Considere por exemplo uma medida  $x_a$  que tem como unidades milímetros e outra variável  $x_b$  que tem como unidades quilômetros. Temos então uma diferença na ordem de grandeza de  $10^6$ . Para eliminar esse desequilíbrio na banda dos valores, podemos dividir cada valor pelo seu desvio padrão (eq. 1.26). Isso em geral é

<sup>9</sup>Isso é fácil de ver, pois  $\mathbb{E}[\mathbf{x}_{\text{centrada}}] = \mathbb{E}[\mathbf{x} - \mathbb{E}[\mathbf{x}]] = \mathbb{E}[\mathbf{x}] - \mathbb{E}[\mathbb{E}[\mathbf{x}]] = \boldsymbol{\mu} - \mathbb{E}[\boldsymbol{\mu}] = \boldsymbol{\mu} - \boldsymbol{\mu} = \mathbf{0}$ . Aqui foi usado a linearidade  $\mathbb{E}[x+y] = \mathbb{E}[x] + \mathbb{E}[y]$  do operador do valor esperado, e o fato que o valor esperado de um valor determinístico é o próprio valor. O valor esperado já não é mais uma variável aleatória, mas tornou-se um valor fixo, determinístico.

feito após centrar os dados pela (eq. 1.36). O efeito após essa medida é que cada variável  $x_j$  tem um novo desvio padrão unitário, equivalentemente uma variância unitária. Portanto, o valor de cada variável é *estandardizado*, aplicando primeiro a centralização e depois a divisão pelo desvio padrão como

$$x_j^{\text{estandardizado}} \stackrel{\text{def}}{=} \frac{x_j - \hat{\mu}_j}{\hat{\sigma}_j}. \quad (1.37)$$

Como a variância (eq. 1.25), das  $m$  diferentes variáveis pode ser extremamente diferente. Consequentemente podemos definir a matriz de dados estandardizado.

**Definição 1.4.4 — Dados Estandardizados, Variável Aleatória Contínua Multidimensional.**

$$X^{\text{estandardizada}} = \begin{bmatrix} \frac{x_{1,1} - \hat{\mu}_1}{\hat{\sigma}_1} & \dots & \frac{x_{1,j} - \hat{\mu}_j}{\hat{\sigma}_j} & \dots & \frac{x_{1,m} - \hat{\mu}_m}{\hat{\sigma}_m} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{x_{i,1} - \hat{\mu}_1}{\hat{\sigma}_1} & \dots & \frac{x_{i,j} - \hat{\mu}_j}{\hat{\sigma}_j} & \dots & \frac{x_{i,m} - \hat{\mu}_m}{\hat{\sigma}_m} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{x_{n,1} - \hat{\mu}_1}{\hat{\sigma}_1} & \dots & \frac{x_{n,j} - \hat{\mu}_j}{\hat{\sigma}_j} & \dots & \frac{x_{n,m} - \hat{\mu}_m}{\hat{\sigma}_m} \end{bmatrix} \quad (1.38)$$

■ **Exemplo 1.9** Vamos continuar com estimativa da média global  $\hat{\mu}$  de Iris na (eq. 1.34). A variância estimada (eq. 1.25) global de cada uma<sup>10</sup> das quatro características, considerando o exemplo concreto da segunda característica (eq. 1.29), resulta em  $\hat{\sigma}_1^2 = 0.6857$ ,  $\hat{\sigma}_2^2 = 0.190$ ,  $\hat{\sigma}_3^2 = 3.116$ ,

$\hat{\sigma}_4^2 = 0.5810$ . A primeira flor da segunda classe tem o padrão  $\mathbf{x}_{51} = \begin{bmatrix} x_{51,1} \\ x_{51,2} \\ x_{51,3} \\ x_{51,4} \end{bmatrix} = \begin{bmatrix} 7.0 \\ 3.2 \\ 4.7 \\ 1.4 \end{bmatrix}$ , o padrão

centrado é  $\mathbf{x}_{51}^{\text{centrado}} = \mathbf{x}_{51} - \hat{\mu} = \begin{bmatrix} x_{51,1} - \hat{\mu}_1 \\ x_{51,2} - \hat{\mu}_2 \\ x_{51,3} - \hat{\mu}_3 \\ x_{51,4} - \hat{\mu}_4 \end{bmatrix} = \begin{bmatrix} 1.157 \\ 0.147 \\ 0.942 \\ 0.201 \end{bmatrix}$ , e o estandardizado é  $\mathbf{x}_{51}^{\text{estandardizado}} =$

$$\begin{bmatrix} \frac{x_{51,1} - \hat{\mu}_1}{\hat{\sigma}_1} \\ \frac{x_{51,2} - \hat{\mu}_2}{\hat{\sigma}_2} \\ \frac{x_{51,3} - \hat{\mu}_3}{\hat{\sigma}_3} \\ \frac{x_{51,4} - \hat{\mu}_4}{\hat{\sigma}_4} \end{bmatrix} = \begin{bmatrix} 1.340 \\ 0.327 \\ 0.5336 \\ 0.2633 \end{bmatrix}.$$

■

### 1.4.1 Covariância e Correlação

Uma variável aleatória escalar  $x_j \in \mathbb{R}^1$ , ou seja, de uma única dimensão pode ser comparada com uma outra variável aleatória escalar  $x_k \in \mathbb{R}^1$ . Não é difícil de entender que, se a segunda variável tem exatamente sempre o mesmo valor da primeira, existe uma grande afinidade entre as duas. Pode-se medir essa afinidade quantitativamente pela *covariância* das duas ou pela *correlação*, que é uma versão normalizada da covariância.

**Definição 1.4.5 — Covariância entre Variáveis Aleatórias Contínuas.** Seja  $\mathbb{E}[x_j] = \mu_j$  o valor esperado (média) (eq. 1.22) de uma variável aleatória  $x_j$  e seja  $\mathbb{E}[x_k] = \mu_k$  o valor esperado (média) de uma variável aleatória  $x_k$ . Então o produto do valor esperado das duas

<sup>10</sup>Essas quatro variâncias serão reencontradas na diagonal da matriz de covariância estimada  $\hat{\Sigma}$  (“sigma maiúsculo”) da (eq. 1.46a) na pág. 27.

variáveis centradas é a sua covariância

$$\sigma_{jk} = \mathbb{E}[(x_j - \mu_j)(x_k - \mu_k)]. \quad (1.39)$$

A covariância pode ser estimada

**Definição 1.4.6 — Covariância Estimada entre Variáveis Aleatórias Contínuas de um Conjunto de Treino.**

$$\hat{\sigma}_{jk} = \frac{1}{n-1} \sum_{i=1}^n (x_{i,j} - \hat{\mu}_j)(x_{i,k} - \hat{\mu}_k), \quad (1.40)$$

onde  $\mathbf{x}_i \in \mathbb{R}^m$  é o  $i$ -ésimo padrão do conjunto de  $n$  padrões, e  $x_{i,j}$  e  $x_{i,k}$  são a  $j$ -ésima e  $k$ -ésima componente desse padrão<sup>a</sup>, respectivamente,  $j, k \in \{1, \dots, m\}$ .

<sup>a</sup>Em uma matriz de dados  $X$  seriam a  $j$ -ésima e  $k$ -ésima coluna da  $i$ -ésima linha,  $X_{i,j}$  e  $X_{i,k}$ .

A correlação<sup>11</sup> entre as duas variáveis, alias “coeficiente de correlação de Pearson”, alias “coeficiente de correlação produto–momento”, alias “ $\rho$  de Pearson” ( $\rho = “rô”$ ) é definida como

**Definição 1.4.7 — Correlação entre Variáveis Aleatórias Contínuas.**

$$\rho_{jk} \stackrel{\text{def}}{=} \text{corr}(x_j, x_k) = \frac{\sigma_{jk}}{\sigma_j \sigma_k}, \quad (1.41)$$

onde  $\sigma_j$  é o desvio padrão (eq. 1.26) de  $x_j$  e  $\sigma_k$  é o desvio padrão de  $x_k$ . A correlação pela normalização automaticamente fica limitada no intervalo

$$-1 \leq \rho_{jk} \leq 1,$$

e a correlação de uma característica com ela própria é igual a um.

Para estimar a correlação basta substituir o desvio padrão e covariância por suas estimativas.

Se considerarmos uma variável aleatória multidimensional  $\mathbf{x}$ , como uma flor que tem quatro variáveis aleatórias, podemos considerar todas as combinações possíveis de covariâncias entre as quatro características. Comparando as definições da variância (eq. 1.25) e da covariância (eq. 1.39), podemos constatar que a variância  $\sigma_j^2$  de uma variável mais nada é que a covariância  $\sigma_{jj}$  desta variável com ela própria, pois  $\sigma_j^2 = \mathbb{E}[(x_j - \mu_j)^2] = \mathbb{E}[(x_j - \mu_j)(x_j - \mu_j)]$ . Além disso, a definição da covariância e correlação revelam suas simetrias  $\sigma_{jk} = \sigma_{kj}$  e  $\rho_{jk} = \rho_{kj}$ . Dessa maneira, podemos agrupar todas as  $m \times m$  combinações possíveis na *Matriz de Covariância*. Para facilitar, usamos na definição o caso das flores  $m = 4$ .

**Definição 1.4.8 — Matriz de Covariância.**

$$\Sigma \stackrel{\text{def}}{=} \begin{bmatrix} \sigma_1^2 & \sigma_{1,2} & \sigma_{1,3} & \sigma_{1,4} \\ \sigma_{2,1} & \sigma_2^2 & \sigma_{2,3} & \sigma_{2,4} \\ \sigma_{3,1} & \sigma_{3,2} & \sigma_3^2 & \sigma_{3,4} \\ \sigma_{4,1} & \sigma_{4,2} & \sigma_{4,3} & \sigma_4^2 \end{bmatrix} \quad (1.42)$$

Formalmente, para a variável aleatória  $\mathbf{x}$  com o seu valor esperado  $\mathbb{E}[\mathbf{x}] = \boldsymbol{\mu}$ , a matriz de covariância pode ser definida ainda como o *produto diádico* do vetor centrado com ele próprio, como

<sup>11</sup>O símbolo “def” em cima de um símbolo de igualdade ou equivalência significa que está sendo feita uma definição simultaneamente.

$$\Sigma \stackrel{\text{def}}{=} \mathbb{E} \left[ (\mathbf{x} - \mathbb{E}(\mathbf{x})) (\mathbf{x} - \mathbb{E}(\mathbf{x}))^\top \right] = \mathbb{E} \left[ (\mathbf{x} - \boldsymbol{\mu}) (\mathbf{x} - \boldsymbol{\mu})^\top \right]. \quad (1.43)$$

Neste ponto, aproveitamos a definição da matriz de covariância para definir o produto diádico (*outer product* em inglês).

**Definição 1.4.9 — Produto Diádico de dois Vetores.** Dados dois vetores de coluna  $\mathbf{a}$  de dimensão  $n \times 1$  e  $\mathbf{b}$  de dimensão  $m \times 1$ , o produto diádico é a multiplicação matricial de  $\mathbf{a}$  pelo transposto de  $\mathbf{b}$ , dando origem a uma matriz de dimensão  $n \times m$

$$\mathbf{ab}^\top = \begin{bmatrix} a_1 \\ \vdots \\ a_i \\ \vdots \\ a_n \end{bmatrix} \begin{bmatrix} b_1 & \cdots & b_j & \cdots & b_m \end{bmatrix} = \begin{bmatrix} a_1 b_1 & \cdots & a_1 b_j & \cdots & a_1 b_m \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ a_i b_1 & \cdots & a_i b_j & \cdots & a_i b_m \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ a_n b_1 & \cdots & a_n b_j & \cdots & a_n b_m \end{bmatrix}. \quad (1.44)$$

A matriz de covariância é simétrica, o que facilitará certas operações do mundo da Álgebra Linear. Vamos precisar determinar os autovalores e autovetores desta matriz posteriormente na Análise das Componentes Principais. A estimativa pode ser definida explicitamente para cada par de variáveis, ou, mais compacto, em forma vetorial. Usando também para a quantidade de  $n = 150$  padrões das flores, temos

**Definição 1.4.10 — Estimativa da Matriz de Covariância.**

$$\hat{\Sigma} = \frac{1}{n-1} \sum_{i=1}^n (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^\top = \frac{1}{149} \sum_{i=1}^{150} (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^\top, \quad (1.45)$$

onde  $\mathbf{x}_i$  é a  $i$ -ésima flor e  $\boldsymbol{\mu}$  é a média global de todas as 150 flores.

Repare que a matriz de covariância não usa qualquer informação a qual classe a flor pertence. Esta definição é também um exemplo do *produto diádico* da (eq. 1.44), de dois vetores. Multiplicam-se um vetor  $m \times 1$  por um vetor  $1 \times m$ , criando uma matriz  $m \times m$ .

■ **Exemplo 1.10** Quem adivinhou que vamos estimar a matriz de covariância das flores, acertou. Além disso mostra-se a matriz de correlação  $\Gamma$  (“gama maiúsculo”) que em cada posição tem a correlação  $\rho_{jk}$  da (eq. 1.41) entre duas características.

$$\hat{\Sigma} = \begin{bmatrix} 0.6857 & -0.0424 & 1.2743 & 0.5163 \\ -0.0424 & 0.1900 & -0.3297 & -0.1216 \\ 1.2743 & -0.3297 & 3.1163 & 1.2956 \\ 0.5163 & -0.1216 & 1.2956 & 0.5810 \end{bmatrix}, \quad (1.46a)$$

$$\hat{\Gamma} = \begin{bmatrix} 1. & -0.1176 & 0.8718 & 0.8179 \\ -0.1176 & 1. & -0.4284 & -0.3661 \\ 0.8718 & -0.4284 & 1. & 0.9629 \\ 0.8179 & -0.3661 & 0.9629 & 1. \end{bmatrix}. \quad (1.46b)$$

Pode-se interpretar destas matrizes, especialmente da matriz da correlação, que, por exemplo, a largura e comprimento da pétala são fortemente correlacionadas  $\rho_{3,4} = \rho_{4,3} = 0.9629$ . Isso é também visível na figura 1.2, pois as duas características são alinhadas. O contrário pode-se afirmar com a largura e comprimento da sépala,  $\rho_{1,2} = \rho_{2,1} = -0.1176$ , visível na mesma figura, onde as amostras formam uma nuvem de pontos sem direção destacada. ■

### 1.4.2 Densidade Gaussiana Multidimensional

Já foi mencionada a importância da densidade normal, ou Gaussiana. Ela tem uma versão multidimensional. Desta vez, os parâmetros são o vetor de médias e a matriz de covariância  $\theta = (\boldsymbol{\mu}, \Sigma)$ . Na verdade, com uma variável aleatória de dimensão  $m$ , esses parâmetros têm vários subparâmetros. O vetor da média  $\boldsymbol{\mu}$  tem  $m$  médias, um para de cada componente, e a matriz de covariância por causa da sua simetria, tem  $m$  variâncias na diagonal e  $(m-1)m/2$  covariâncias acima (ou abaixo) da diagonal, totalizando  $m + \frac{(m+1)m}{2}$  parâmetros. No caso das flores temos quatro médias  $\mu_1, \mu_2, \mu_3, \mu_4$  e quatro variâncias  $\sigma_1^2, \sigma_2^2, \sigma_3^2, \sigma_4^2$  e seis covariâncias  $\sigma_{1,2}, \sigma_{1,3}, \sigma_{1,4}, \sigma_{2,3}, \sigma_{2,4}, \sigma_{3,4}$ , totalizando 14 parâmetros.

**Definição 1.4.11 — Densidade Normal, Variável Aleatória Contínua Multidimensional.**

$$\begin{aligned} p(\mathbf{x}; \boldsymbol{\mu}, \Sigma) &= \frac{1}{\sqrt{(2\pi)^m |\Sigma|}} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^\top \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right\} \\ &\stackrel{\text{def}}{=} \frac{1}{\sqrt{(2\pi)^m |\Sigma|}} \exp \left\{ -\frac{1}{2} D_M^2 \right\}, \end{aligned} \quad (1.47)$$

onde  $\boldsymbol{\mu} \in \mathbb{R}^m$  é o vetor de média,  $\Sigma \in \mathbb{R}^{m \times m}$  é a matriz de covariância e  $|\Sigma| = \det(\Sigma)$  é o determinante da matriz de covariância.

No argumento da função foi usado o quadrado da *Distância de Mahalanobis*  $D_M$  que é uma métrica muito importante [70].

**Definição 1.4.12 — Distância de Mahalanobis entre Vetor  $x$  e Vetor Média  $\mu$ .**

$$D_M(\mathbf{x}, \boldsymbol{\mu}) \stackrel{\text{def}}{=} \sqrt{(\mathbf{x} - \boldsymbol{\mu})^\top \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu})} \quad (1.48)$$

onde  $\boldsymbol{\mu}$  é o vetor de média e  $\Sigma$  é a matriz de covariância.

Em uma dimensão a Distância de Mahalanobis simplifica para o módulo da diferença entre média e o padrão escalar  $x \in \mathbb{R}$ , ainda normalizado pelo desvio padrão. Assim

$$D_M(x, \mu) = \sqrt{(x - \mu) \frac{1}{\sigma^2} (x - \mu)} = \frac{|x - \mu|}{\sigma}. \quad (1.49)$$

A distância de Mahalanobis é um tipo de distância de Euclidiana normalizada, dividindo a distância Euclidiana pelo desvio padrão da variável. O numerador seria uma distância Euclidiana em uma dimensão, reduzindo simplesmente para o valor absoluto da diferença entre os dois pontos. Comparando a Distância de Mahalanobis unidimensional com o valor estandardizado de uma característica na (eq. 1.37), podemos constatar que  $D_M(x, \mu) = |x^{\text{estandardizado}}|$ , ou seja, é o módulo do valor estandardizado. Repare que a expressão  $(\mathbf{x} - \boldsymbol{\mu})^\top \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu})$  é a multiplicação de um vetor transposto de dimensão  $1 \times m$  por uma matriz  $m \times m$ , resultando em vetor  $1 \times m$ . Em seguida é multiplicado por vetor de dimensão  $m \times 1$ , resultando finalmente em matriz  $1 \times 1$ , ou seja, em escalar. A distância de Mahalanobis é diferente da distância Euclidiana. Excepcionalmente a distância de Mahalanobis simplifica para a distância Euclidiana, se a matriz de covariância for a matriz de identidade, ou seja,

$$\Sigma = I.$$

Nesse caso a (eq. 1.48), usando o seu quadrado, resulta em

$$\begin{aligned}
 D_M^2(\mathbf{x}, \boldsymbol{\mu}) &= (\mathbf{x} - \boldsymbol{\mu})^\top \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}) \\
 &= (\mathbf{x} - \boldsymbol{\mu})^\top I^{-1} (\mathbf{x} - \boldsymbol{\mu}) \\
 &= (\mathbf{x} - \boldsymbol{\mu})^\top I (\mathbf{x} - \boldsymbol{\mu}) \\
 &= (\mathbf{x} - \boldsymbol{\mu})^\top (\mathbf{x} - \boldsymbol{\mu}) \\
 &\stackrel{1.4}{=} \|\mathbf{x} - \boldsymbol{\mu}\|^2 \\
 &\stackrel{1.11}{=} D_E^2(\mathbf{x}, \boldsymbol{\mu}).
 \end{aligned}$$

Em  $m = 2$  dimensões, um ponto no sistema de coordenadas cartesiano que tem uma distância de Mahalanobis constante do centro descreve uma trajetória de elipse, definida pela matriz de covariância. A terra tem distâncias Euclidianas sempre diferentes quando roda o sol, por causa disso existem o afélio (ponto mais próximo) e periélio (ponto mais afastado). Como a trajetória é uma elipse, a distância de Mahalanobis em relação ao centro da elipse é sempre a mesma. Porém esse centro não é o sol, pois o sol fica em um dos dois focos da elipse, de acordo com a primeira lei de Kepler. As figura 1.4 e figura 1.5 mostram dois gráficos que ilustram os conceitos da densidade Gaussiana bidimensional, mas uma vez para as últimas duas características  $x_3$  e  $x_4$  das flores. É recomendado ao leitor comparar a figura 1.2a e figura 1.2b para análise.

Na figura 1.5, em um sistema de coordenadas cartesiano de três dimensões,  $x_3$  e  $x_4$  formam o plano, onde “vivem” as nossas flores, se ignorarmos as primeiras duas características. A partir das 50 amostras de cada classe, as médias e covariâncias já foram estimadas, veja (eq. 1.34) e (eq. 1.46a). Com esses dois parâmetros, é possível definir a densidade (eq. 1.47). Assim para cada ponto  $\mathbf{x} = [x_3 \ x_4]^\top$  que neste caso tem duas dimensões, é possível calcular o valor da terceira coordenada. Com essas triplas  $(x_3, x_4, p(\mathbf{x}))$  é possível visualizar a densidade em forma de superfícies ou *wireframe*. Esses “morros”, no caso da Gaussiana bidimensional, têm o seu maior valor na posição da média  $\mathbf{x} = \boldsymbol{\mu}$ . Neste caso, a distância de Mahalanobis é zero, pois  $\mathbf{x} - \boldsymbol{\mu} = \mathbf{0}$  e o exponencial com o argumento igual a zero retorna o valor um.

Abaixo de cada uma das três densidades mostram-se no plano  $(x_3, x_4)$  as *curvas de contorno*. Essas curvas são explicitadas mais claramente na figura 1.4. São pontos para quais a densidade é constante, ou seja,  $p(\mathbf{x}; \boldsymbol{\mu}, \Sigma) = \text{constante}$ . No caso bidimensional, essas curvas formam elipses. O gráfico mostra para cada uma das três classes duas curvas de contorno, a primeira, mais próxima da média representa todos os pontos, onde a densidade tem 50% do valor máximo do centro  $\boldsymbol{\mu}$ , a segunda curva representada os ponto com 10% do valor máximo do centro  $\boldsymbol{\mu}$ . Além disso, três vetores  $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\mu}_3$  traçam o vetor de média da origem até as médias de cada classe. Se não existir variância e covariância, ou seja,  $\Sigma = I$ , as elipses simplificam para círculos. Em três dimensões temos elipsoides e esferas, e em mais que três dimensões, hiperelipsoides e hiperesferas.

Fica óbvio que a distância de Mahalanobis pode ser usada para classificar. Quanto mais próximo da média da classe, mais provável que o padrão seja dessa classe. No contexto de classificadores paramétricos voltaremos a esse tópico.

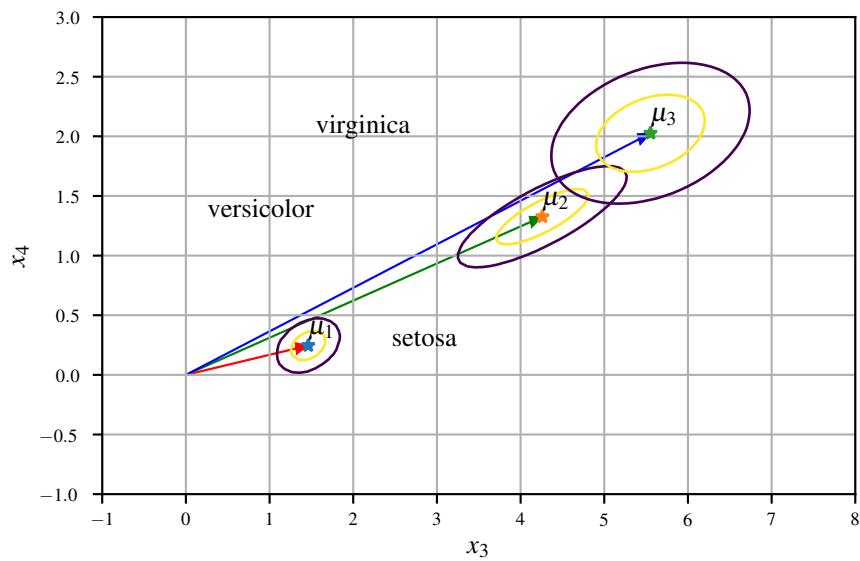


Figura 1.4: Vetores de médias e linhas de contorno para as três classes de Iris no espaço de características  $x_3$  e  $x_4$ .

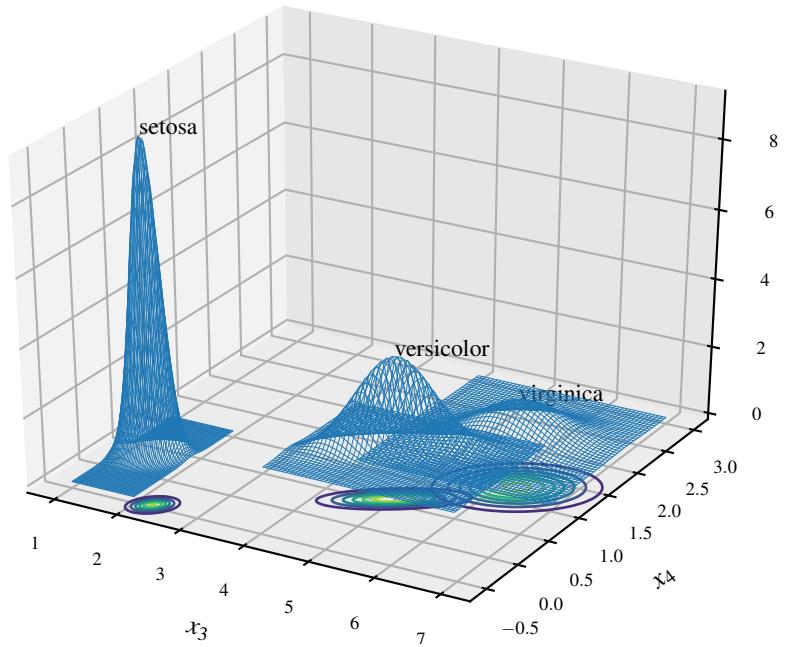


Figura 1.5: Densidades Gaussianas no espaço de características  $x_3$  e  $x_4$ .

## 2. Regressão e Classificação Linear

### 2.1 Modelos Lineares

Precisamos muito pouco, se quisermos classificar duas classes. O modelo do classificador linear binário, aquele que sabe distinguir entre uma coisa e outra em um espaço vetorial de dimensão  $m$ , precisa somente  $(m + 1)$  números reais. Vamos aprender como um classificador desses é construído, o seu modelo, ou equivalentemente, a sua *arquitetura*. A arquitetura sempre precisa de um método que treina o que pode ser modificado nessa arquitetura que são os parâmetros livres. Em uma rede neural, por exemplo, são os pesos. Esse é o algoritmo de aprendizagem. Em certos modelos, o treinamento é determinístico, usando uma forma fechada. Isso sempre é possível, se o modelo é *linear*. Precisamos calcular funções, não somente para classificar, mas também para problemas de *regressão*. Regressão é mais geral que classificação. Em um problema de regressão, a saída da função é contínua. Na classificação também, porém na classificação temos como passo final uma categorização do valor calculado em um valor discreto que indica uma das duas classes. A maneira mais fácil de dividir um valor contínuo em duas categorias é simplesmente considerar o seu sinal. Se for positivo, pertence a classe positiva, se for negativa a classe negativa.

#### 2.1.1 Um Exemplo Mínimo

Qualquer sistema de aprendizado de máquina no fundo fornece-nos uma função  $f$  que resolve o problema da melhor maneira possível. Como argumentos de entrada serve um vetor  $\mathbf{x}$ , no caso mais simples um escalar  $x$ . A saída da função pode ser um escalar  $y$  também. Se a saída tiver mais que uma dimensão, também usaremos a letra em negrito, portanto  $\mathbf{y} = \mathbf{f}(\dots)$ . Algumas coisas na função não podemos modificar. Isso é chamada a arquitetura do sistema, ou como sinônimo, o modelo. Outras coisas da função devem ser modificadas para otimizar o resultado, chamados os parâmetros livres. No contexto de variáveis aleatórias, usamos o símbolo  $\theta$  (“teta”) para denominar um ou mais parâmetros. No contexto de redes neurais, normalmente um parâmetro é chamado *peso*, ou em inglês *weight*. Como as redes neurais artificiais ocupam um parte considerável de aprendizado de máquina, usaremos a letra  $w$  para falar de um único parâmetro,  $w$ , se os parâmetros podem ser agrupados em um vetor, ou  $W$ , se faz sentido agrupar os parâmetros em forma de matriz. Um sistema de aprendizado de máquina em geral pode ser caracterizado como uma função

que recebe uma entrada multidimensional, fornece uma saída multidimensional e tem parâmetros livres ajustáveis.

**Definição 2.1.1 — Modelo ou Arquitetura de um Sistema de Aprendizagem de Máquina.**

$$\mathbf{f}(\mathbf{x}; \mathbf{w}) = \mathbf{y}, \quad (2.1)$$

onde  $\mathbf{x}$  é o vetor de entrada do modelo,  $\mathbf{w}$  são os parâmetros livres do modelo, e  $\mathbf{y}$  é o vetor de saída do modelo. A entrada  $\mathbf{x}$  são as variáveis independentes, e a saída  $\mathbf{y}$  são as variáveis dependentes.

Vamos considerar primeiramente um modelo simplificado, no sentido que tem uma única saída  $y \in \mathbb{R}$ , ou seja,  $y = f(\mathbf{x}; \mathbf{w})$ . Além disso restrinjimos o cálculo da saída para ser linear, ou seja, que contém somente termos da forma  $w_j x_j$ .

**Definição 2.1.2 — Modelo Linear.**

$$f(\mathbf{x}; \mathbf{w}, w_0) = \mathbf{w} \cdot \mathbf{x} + w_0 = \mathbf{w}^\top \mathbf{x} + w_0 = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_m x_m, \quad (2.2)$$

com  $\mathbf{x} \in \mathbb{R}^m$ ,  $\mathbf{w} \in \mathbb{R}^m$ ,  $w_0 \in \mathbb{R}$ .

Repare que tem um termo constante  $w_0 \in \mathbb{R}$  nessa soma que não depende da entrada  $\mathbf{x}$ . Essa constante é o viés (*bias*). Por isso, alternativamente, usa-se muitas vezes a letra  $b$ . Em um espaço de  $m$  dimensões, o bias é absolutamente necessário para permitir somar um valor constante ao valor calculado pela combinação linear. Para entender isso melhor, vamos nos limitar ainda mais, considerando uma função linear com uma única entrada  $x$  e uma única saída  $y$ . Se essa função não tivesse o viés, teria a forma

$$y = f(x; w) = wx. \quad (2.3)$$

É fácil reconhecer isso como uma reta no sistema de coordenadas cartesiano que passa pela origem, independente do parâmetro  $w$ , pois  $f(0; w) = w \cdot 0 = 0$ . Por outro lado, um modelo tem que ser capaz de calcular uma saída constante, diferente de zero, para uma entrada qualquer. Se formos obrigados de usar o modelo em (eq. 2.3), a única maneira de obter uma saída constante com o valor  $b$  para qualquer entrada  $x$ , seria mudar o parâmetro  $w$  para  $b/x$ , mas assim deixaria de ser um parâmetro, pois dependeria do argumento  $x$ . Então, o modelo linear mais simples, com uma entrada e uma saída tem que ter dois parâmetros  $w$  e  $b$ , e tem a forma

$$y = f(x; w, b) = wx + b. \quad (2.4)$$

Isso é reconhecível como uma reta no sistema de coordenadas cartesiano que não necessariamente tem que passar pela origem, e pode fornecer uma saída constante  $b$  para uma entrada nula  $x = 0$ .

## 2.1.2 Aprendizagem Supervisionada

Chegou a hora de aprender como mudar o que pode ser mudado em um modelo, se um conjunto de treinamento é fornecido. No caso do mais simples sistema imaginável de (eq. 2.4), o objetivo é aprender, ou seja estimar os dois parâmetros livres  $w$  e  $b$ , baseado em um conjunto de treinamento. Esse conjunto de treinamento basicamente diz “Para essa entrada, eu quero essa saída, e para aquela outra entrada eu quero aquela saída”. Existem então  $n$  padrões de entrada  $x_i$ ,  $i = 1, \dots, n$  e  $n$  padrões de saída  $y_i$ ,  $i = 1, \dots, n$ . Essas saídas  $y_i$  são as *saídas desejadas*. Quando alguma entrada  $x_i$  do conjunto de treinamento entra na função como argumento, o resultado  $\hat{y}_i$  depende dos dois parâmetros  $w$  e  $b$ . Em geral esse valor, a *saída calculada*, ou *saída explicada pelo*

modelo é diferente da saída desejada. Temos que medir essa discrepância entre os valores de saída desejados e explicados. Vamos definir uma função  $L$  que mede essa discrepância, ou custo, ou erro, ou perda. Perda em inglês é *loss*. Vamos primeiro formalizar a definição do conjunto de treinamento.

**Definição 2.1.3 — Conjunto de Treinamento de Pares de Entrada-Saída.** Seja  $\mathbf{x} \in \mathbb{R}^m$  um vetor de entrada de dimensão  $m$  e  $\mathbf{y} \in \mathbb{R}^c$  um vetor de saída de dimensão  $c$ . Então o conjunto de dados ou conjunto de treinamento é o conjunto de  $n$  desses pares  $\mathbf{x}$  e  $\mathbf{y}$

$$\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_i, \mathbf{y}_i), \dots, (\mathbf{x}_n, \mathbf{y}_n)\}. \quad (2.5)$$

Para problemas de regressão, os  $c$  valores  $y_{i,k} \in \mathbb{R}$  de um vetor de saída  $\mathbf{y}_i$  podem assumir um valor numérico real qualquer. Para problemas de classificação, existe uma maneira apropriada de codificar a qual classe um padrão pertence<sup>1</sup>. Já foi constatado que não é justo simplesmente enumerar a saída como 1, 2, 3. É melhor usar a codificação “Um-Quente”, *One-hot* em inglês. Com uma quantidade de  $c$  classes, forma-se um vetor  $\mathbf{y}$  que na  $k$ -ésima posição tem um valor de um, e zero nas posições restantes.

**Definição 2.1.4 — Codificação de Associação de Classe One-Hot.** Se o padrão  $\mathbf{x}_i$  pertence a classe  $k \in \{1, \dots, c\}$ , a saída associada é

$$\mathbf{y}_i = [y_{i,1} \ y_{i,2} \ \cdots \ y_{i,k-1} \ y_{i,k} \ y_{i,k+1} \ \cdots \ y_{i,c}]^\top = [0 \ 0 \ \cdots \ 0 \ 1 \ 0 \ \cdots \ 0]^\top \quad (2.6)$$

■ **Exemplo 2.1** Vamos considerar a classificação das flores. A codificação *One-Hot* resulta em  $[1 \ 0 \ 0]^\top$  para setosa,  $[0 \ 1 \ 0]^\top$  para versicolor e  $[0 \ 0 \ 1]^\top$  para virginica. Então o conjunto de dados de treinamento das flores é  $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_{150}, \mathbf{y}_{150})\} = \left\{ \left( [5.1 \ 3.5 \ 1.4 \ 0.2]^\top, [1 \ 0 \ 0]^\top \right), \dots, \left( [5.9 \ 3.0 \ 5.1 \ 1.8]^\top, [0 \ 0 \ 1]^\top \right) \right\}$ . ■

### 2.1.3 Regressão Linear Simples

O problema mais simples de aprendizagem supervisionada é um problema de *ajuste de curvas*. Desta vez as nossas saídas desejadas  $y_i$  e saídas explicadas  $\hat{y}_i$  são valores contínuos, ou seja,  $y_i \in \mathbb{R}$  e  $\hat{y}_i \in \mathbb{R}$ . Por isso, vamos deixar pausar o nosso exemplo preferido, as flores Iris, e vamos definir um exemplo apropriado com saídas contínuas, pois a saída desejada no caso das flores é uma variável aleatória discreta, o rótulo de classe que pode ser representada por um vetor de saída através da codificação *One-Hot* da (eq. 2.6) acima mencionada. Se a curva a ser ajustada for uma reta, temos o modelo linear simples da (eq. 2.4). Temos que providenciar um conjunto de treinamento  $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_n, \mathbf{y}_n)\}$  que para cada entrada  $x_i$  associa uma saída desejada contínua  $y_i$ ,  $i = 1, \dots, n$ . A tabela 2.1 mostra o conjunto de treinamento de  $n = 7$  padrões.

A figura 2.1 mostra todos os  $n = 7$  pontos, junto com dois possíveis resultados da aprendizagem. A curva (reta) vermelha seria uma escolha muito inapropriada, pois os pontos ficam longe dela. Na verdade, se o nosso modelo tem que ser uma reta (eq. 2.4), já que é impossível forçar que todos os pontos façam parte desta reta, vamos encontrar a melhor reta. Isso significa que temos que escolher uma reta que mais se ajusta aos pontos. Essa reta ótima é a verde, pois ela faz o possível para que todos os pontos fiquem próximos dela. A questão é como essa otimalidade do ajuste pode ser quantificada e como podemos chegar ao resultado ótimo. Sempre quando falamos de valores

<sup>1</sup>É possível associar um padrão com mais que uma classe, por exemplo, um artigo científico pode ser da classe biologia, e simultaneamente da classe informática. Essa categoria de problemas é a classificação *multi-rótulo*.

Número do padrão	$x$	$y$
1	0.4	0.7
2	0.9	1.0
3	1.5	0.8
4	2.3	0.9
5	2.9	1.4
6	3.1	2.1
7	3.7	2.4

Tabela 2.1: Conjunto de treinamento para um problema de aprendizagem supervisionada.

ótimos, o símbolo usado é o asterisco “\*”. Neste caso os valores ótimos dos dois parâmetros  $w$  e  $b$  de (eq. 2.4) são  $w^* = 0.475$  e  $b^* = 0.325$ .

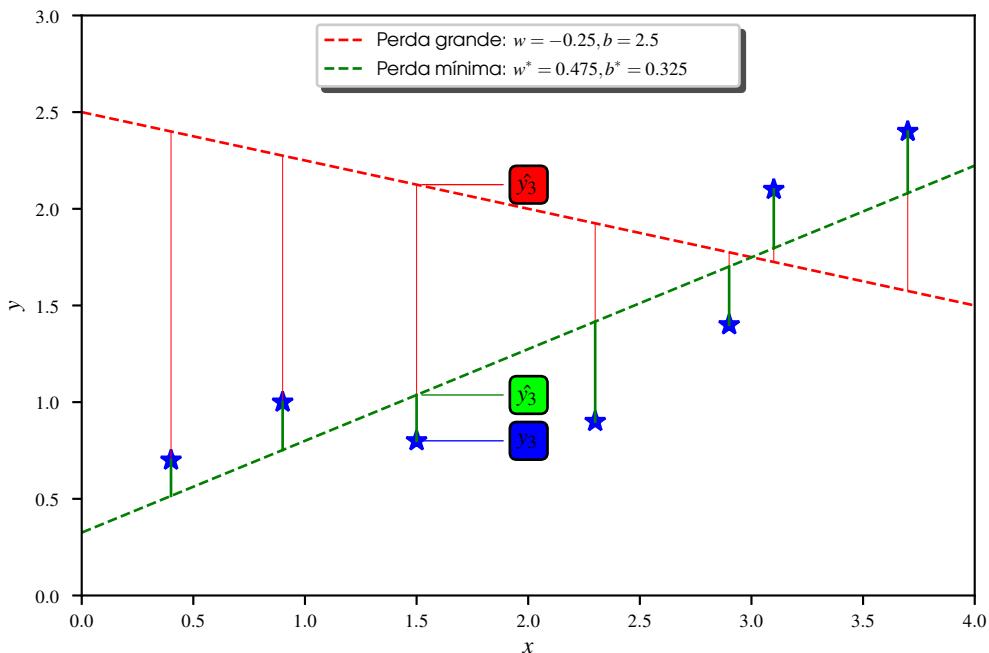


Figura 2.1: Regressão Linear, ajuste desfavorável e ajuste ótimo. Para um dos padrões, mostra-se o valor desejado  $y_i$ , e os valores  $\hat{y}_i$  explicados para o caso desfavorável (em vermelho) e o caso otimizado (em verde).

A fórmula ensinada em cálculo numérico [12, 39] é

$$w^* = \frac{\sum x_i \sum y_i - n \sum x_i y_i}{(\sum x_i)^2 - n \sum x_i^2}, \quad b^* = \frac{\sum y_i - w^* \sum x_i}{n}, \quad (2.7)$$

que nesse caso é

$$w^* = \frac{14.8 \cdot 9.3 - 7 \cdot 23.9}{14.8^2 - 7 \cdot 40.22} = 0.475, \quad b^* = \frac{9.3 - 0.475 \cdot 14.8}{7} = 0.325. \quad (2.8)$$

Qual é a origem desse cálculo? Uma primeira observação é que existe uma forma fechada para solucionar este problema. Isto significa que existe uma fórmula  $\theta^* = (w^*, b^*) = \dots$ , que permite

calcular os parâmetros diretamente, sem ter que usar um algoritmo iterativo. Falando nisso, um algoritmo iterativo tem uma estrutura típica. Ele tem que ser inicializado com um resultado adivinhado, e durante a repetição do laço principal do algoritmo, o resultado melhora cada vez mais. Pelo menos, esse é o comportamento desejado.

No caso da obtenção direta do resultado em forma fechada, temos as regras  $\theta_1^* = w^*$  e  $\theta_2^* = b^*$  na (eq. 2.7). Esse cálculo por uma equação é possível porque o modelo é linear. Se fosse um modelo não linear, em geral teríamos que usar um algoritmo para obter os parâmetros livres do modelo. Uma rede neural profunda às vezes treina um conjunto de dados enorme, realizando um modelo altamente não linear, e com muitos parâmetros livres. Para essa rede profunda não existe forma fechada. Em breve vamos também estudar o coração deste treinamento iterativo, a *descida de gradiente*.

Voltando ao cálculo dos parâmetros, podemos constatar que a (eq. 2.7) é o resultado de um problema de otimização, minimizando um critério de otimização. Temos que definir uma função de perda que quantifica a qualidade do ajuste.

**Definição 2.1.5 — Função de Perda, (Loss Function)** . Dado um resultado desejado  $y$  e um resultado explicado pelo modelo  $\hat{y}$ , a função de perda mede a discordância ou discrepância entre os dois resultados.

$$L(y, \hat{y}) = \dots \quad (2.9)$$

De propósito a definição ficou em aberto, pois na verdade a função de perda é mais que uma função, é um *funcional*. Especialmente no contexto em que a gente trabalha, entendemos um funcional como uma função que pode ter, além dos argumentos comuns, uma função como argumento. Na (eq. 2.9), o valor desejado  $y$  em aprendizagem supervisionado é fixo. O valor  $\hat{y}$  que o modelo calcula, depende da função que está sendo usada, neste nosso modelo básico  $\hat{y} = f(x; w, b) = wx + b$ . Se a função  $f$  fosse outra, o valor de  $\hat{y}$  também seria outro. Outro valor fixo é a entrada no modelo, o argumento  $x$ . Por isso, vamos omitir  $x$  como argumento na função de perda.

Então, uma vez definido o modelo (eq. 2.4), e dado o conjunto de treinamento (eq. 2.5), a única coisa que influencia a função de perda são os parâmetros  $\theta$ , podemos dizer que perda é uma função  $L(\theta)$  dos parâmetros, no nosso caso do modelo linear simples é uma função  $L(w, b)$  dos dois parâmetros  $w$  e  $b$ .

### Perda Esperada

Como no caso do valor esperado de uma variável aleatória da seção 1.2.3, a perda de um único padrão não representa a qualidade do ajuste completa. Em algoritmos iterativos até vamos ver que um único padrão já pode ser usado para o processo de aprendizagem, por exemplo na descida de gradiente estocástica (*Stochastic Gradient Descent, SGD*). Porém neste contexto, do cálculo direto dos parâmetros otimizados, precisamos julgar o modelo ajustado (neste caso a reta) em relação a todos os padrões. No exemplo concreto da regressão linear são os sete pontos. Vamos omitir a parte analítica, e logo definir a perda esperada, ou, como sinônimo o *risco empírico* [43, 112]. É a média aritmética das perdas individuais

$$L_i(\theta), \quad i = 1, \dots, n \quad (2.10)$$

relativas a cada um dos  $n$  padrões, portanto

$$L(\theta) = \frac{1}{n} \sum_{i=1}^n L_i(\theta). \quad (2.11)$$

No caso do modelo linear simples, temos dois parâmetros  $w$  e  $b$ , portanto o risco empírico especifica para

$$L(w, b) = \frac{1}{n} \sum_{i=1}^n L_i(w, b), \quad (2.12)$$

onde

$$L_i(w, b), \quad i = 1, \dots, n \quad (2.13)$$

mais uma vez é a perda individual associada a cada padrão  $(x_i, y_i)$ . Obviamente, o conjunto de parâmetros ótimos  $\boldsymbol{\theta}^*$  minimiza o risco empírico.

Na análise de funções uma tarefa muito importante é a busca de extremos. Especialmente no contexto de otimização, buscam-se principalmente os mínimos de uma função, ou seja, os valores  $x$  para qual a função  $f(x)$  tem o menor valor possível. Aproveitamos para distinguir entre o mínimo “min” de uma função e o argumento “arg min” que foi o valor  $x$  que causou a função ter esse mínimo, portanto, incluindo também o máximo e seu argumento causador

#### Definição 2.1.6 — Max e ArgMax, Min e ArgMin.

$$\begin{aligned} \max_x f(x) &\stackrel{\text{def}}{=} \{f(x) \mid \forall y : f(x) \geq f(y)\} & \min_x f(x) &\stackrel{\text{def}}{=} \{f(x) \mid \forall y : f(x) \leq f(y)\} \\ \arg \max_x f(x) &\stackrel{\text{def}}{=} \{x \mid \forall y : f(x) \geq f(y)\} & \arg \min_x f(x) &\stackrel{\text{def}}{=} \{x \mid \forall y : f(x) \leq f(y)\} \end{aligned} \quad (2.14)$$

Estas definições são bastante frequentes para identificar ou o valor extremo da função, ou qual argumento levou a esse extremo.

**Exemplo 2.2** Considere a função  $f(x) = x^2 - 1$ . Temos  $\min_x f(x) = \{-1\}$  e  $\arg \min_x f(x) = \{0\}$ . Como a função não tem limite superior, não é possível obter um valor único para  $\max_x f(x)$  e  $\arg \max_x f(x)$ . ■

Considerando o risco empírico da (eq. 2.11) que queremos minimizar, podemos então formular o parâmetro ótimo como aquele que minimiza essa função

$$\boldsymbol{\theta}^* = \arg \min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = \arg \min_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^n L_i(\boldsymbol{\theta}), \quad (2.15)$$

e no caso do modelo linear simples

$$w^*, b^* = \arg \min_{w, b} L(w, b) = \arg \min_{w, b} \frac{1}{n} \sum_{i=1}^n L_i(w, b). \quad (2.16)$$

Para obter os parâmetros ótimos no caso do modelo linear, basta calcular o gradiente da função de perda e buscar uma raiz desse gradiente, ou seja um valor que zera o gradiente. Por isso, vamos fazer uma revisão dos conceitos “derivada”, “derivada parcial” e “gradiente”.

#### Derivada da Perda Relativa aos Parâmetros

Nenhum método de aprendizado de máquina pode viver sem calcular a derivada de uma função que mede a qualidade do aprendizado, em relação aos parâmetros livres do modelo. No nosso caso temos que calcular a derivada da perda individual

$$\frac{dL_i(\boldsymbol{\theta})}{d\boldsymbol{\theta}}, \quad i = 1, \dots, n. \quad (2.17)$$

O risco (eq. 2.12) é uma soma de termos. Sabe-se que a derivada de uma soma é a soma das derivadas dos termos que formam a soma. Por isso, basta considerar a derivada de um único

termo  $L_i(\boldsymbol{\theta})$  e posteriormente somar as derivadas destes termos. Lembre-se que o modelo linear apresentado em (eq. 2.4) é o mais simples modelo de regressão que existe. Mesmo assim, ele já tem dois parâmetros,  $\boldsymbol{\theta} = (w, b)$ , ou seja, é uma função que depende de dois argumentos, logo, tem um vetor de duas componentes como argumento. Isso significa que a derivada

$$\frac{dL_i(\boldsymbol{\theta})}{d\boldsymbol{\theta}} = \frac{dL_i\left(\begin{bmatrix} w \\ b \end{bmatrix}\right)}{d\left(\begin{bmatrix} w \\ b \end{bmatrix}\right)} \quad (2.18)$$

é a derivada de uma função que depende de mais que uma variável em relação a todas essas variáveis. Revisamos aqui a definição extremamente importante do *gradiente*.

**Definição 2.1.7 — Gradiente de Função Escalar.** Por um momento vamos mudar o nome das variáveis para apresentar a teoria, pois na literatura em geral, a função chama-se  $f$  e a variável chama-se  $\mathbf{x}$ , ou frequentemente, na literatura de otimização usa-se para uma variável com mais de uma única componente o símbolo não negrito  $x$ . Não confunda esta função com a função que define o modelo linear, aquela da (eq. 2.4). Seja dada uma função vetorial  $f(\mathbf{x})$

$$f : \mathbb{R}^m \rightarrow \mathbb{R} \quad (2.19)$$

que tem como argumento de entrada um vetor  $\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_j \\ \vdots \\ x_m \end{bmatrix}$  de  $m$  números reais  $x_j$ ,  $j = 1, \dots, m$ ,

e tem como resultado um escalar, ou seja, um único número real  $f(\mathbf{x}) \in \mathbb{R}$ . O *gradiente* da função é um vetor de linha de dimensão  $m$  que em cada componente tem a *derivada parcial*  $\partial f(\mathbf{x})/\partial x_j$  da função  $f$  em relação a cada uma das variáveis  $x_j$

$$\nabla f = \text{grad } f = \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \left[ \frac{\partial f(\mathbf{x})}{\partial x_1} \quad \cdots \quad \frac{\partial f(\mathbf{x})}{\partial x_j} \quad \cdots \quad \frac{\partial f(\mathbf{x})}{\partial x_m} \right]. \quad (2.20)$$

A forma do gradiente de uma função escalar de variável vetorial como vetor de *linha* é importante para manter os cálculos matriciais consistentes, como multiplicação de vetor por matriz ou multiplicação de matriz por vetor. O símbolo “ $\nabla$ ” chama-se “nabla”. O gradiente de uma função de perda é o motor do aprendizado de máquina. O cálculo do gradiente guia o conjunto dos parâmetros livres para se deslocarem na direção em que na próxima iteração de um algoritmo iterativo a perda seja menor. Dessa maneira busca-se um mínimo dessa função. Vamos ver que essa otimização iterativa, embora não seja necessária, é possível com o nosso modelo linear simples para obter os parâmetros ótimos  $w^*$  e  $b^*$ . Também no caso do modelo linear vamos precisar do gradiente para obter a solução ótima da (eq. 2.7) em uma forma fechada.

■ **Exemplo 2.3** Como exemplo com  $m = 2$  argumentos, considere  $f(\mathbf{x}) = f\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = 3x_1^2 x_2 + x_2^3$ . O gradiente é

$$\nabla f(\mathbf{x}) = \left[ \frac{\partial f(\mathbf{x})}{\partial x_1} \quad \frac{\partial f(\mathbf{x})}{\partial x_2} \right] = [6x_1 x_2 \quad 3x_1^2 + 3x_2^2]. \quad (2.21)$$

A primeira derivada parcial  $\partial f(\mathbf{x})/\partial x_1$  considera  $x_1$  como variável e  $x_2$  como constante e a segunda  $\partial f(\mathbf{x})/\partial x_2$  considera  $x_1$  como constante e  $x_2$  como variável. ■

Se não se lembra das regras para calcular as derivadas de uma função, está na hora de refrescá-las. Mesmo se nunca teve contato com essa ferramenta matemática, precisa aprendê-la, pois sem o entendimento do gradiente, não conseguirá entender algoritmos mais sofisticados de aprendizado de máquina. Algumas regras essenciais são as derivadas de

$$\frac{dx^n}{dx} = nx^{n-1} \quad \text{um monômio (de um polinômio)} \quad (2.22a)$$

$$\frac{d[cf(x)]}{dx} = c \frac{df(x)}{dx} \quad \text{termo multiplicado por uma constante } c \quad (2.22b)$$

$$\frac{d[\sum_j f_j(x)]}{dx} = \sum_j \frac{df_j(x)}{dx} \quad \text{soma que é a soma das derivadas dos termos individuais} \quad (2.22c)$$

### Função de Perda: Primeira Tentativa

A mais óbvia das escolhas para medir a discrepância entre um valor numérico  $y_i$  e valor numérico  $\hat{y}_i$  seria a sua *diferença*

$$L_i(w, b) = y_i - \hat{y}_i = y_i - (wx_i + b). \quad (2.23)$$

O grande problema de simplesmente usar a diferença  $y_i - \hat{y}_i$  entre os valores desejados  $y_i$  e valores explicados  $\hat{y}_i$  é que em média podem ter o valor zero, pois às vezes a diferença tem um sinal positivo, e às vezes tem um sinal negativo. Graficamente podemos observar estas diferenças na figura 2.1 como as retas verticais entre o valor  $y_i$  dado pela tabela 2.1 e o valor  $\hat{y}_i = wx_i + b$  resultante, quando seguimos o ponto  $(x_i, y_i)$  até a reta, onde fica o valor explicado  $(x_i, \hat{y}_i)$ . Às vezes os valores desejados ficam acima da reta e às vezes ficam abaixo da reta. Em média estas diferenças podem se compensar de tal maneira que a soma e consequentemente a média destas diferenças seja zero. Assim, mesmo com um modelo inapropriado, longe do ótimo, o erro poderia ser nulo, o que contradiz a ideia da função de perda. A função de perda não pode ter um valor nulo ou pequeno se os pontos não se ajustam bem à reta.

Voltando a primeira tentativa de definir um termo da nossa função de perda (eq. 2.23) reconhecemos que é uma função vetorial que tem dois argumentos,  $w$  e  $b$ , e assim o gradiente seria

$$\nabla L_i(w, b) = \begin{bmatrix} \partial L_i(w, b) / \partial w \\ \partial L_i(w, b) / \partial b \end{bmatrix}^\top = \begin{bmatrix} \frac{\partial(y_i - \hat{y}_i)}{\partial w} \\ \frac{\partial(y_i - \hat{y}_i)}{\partial b} \end{bmatrix}^\top = \begin{bmatrix} \frac{\partial(y_i - wx_i - b)}{\partial w} \\ \frac{\partial(y_i - wx_i - b)}{\partial b} \end{bmatrix}^\top = - \begin{bmatrix} x_i \\ 1 \end{bmatrix}^\top. \quad (2.24)$$

Para economizar espaço, usamos aqui a transposição do vetor de linha, resultando em um vetor de coluna, ou seja, mostra-se o gradiente  $\nabla L^\top$ , que é um vetor de linha em forma vertical. Consequentemente, o gradiente da função de perda  $L(w, b)$ , definida em (eq. 2.12), como média destes termos é

$$\begin{aligned} \nabla L(w, b) &= \frac{1}{n} \sum_{i=1}^n \nabla L_i(w, b) = \begin{bmatrix} \frac{1}{n} \sum_{i=1}^n \partial L_i(w, b) / \partial w \\ \frac{1}{n} \sum_{i=1}^n \partial L_i(w, b) / \partial b \end{bmatrix}^\top \\ &= - \begin{bmatrix} \frac{1}{n} \sum_{i=1}^n x_i \\ \frac{1}{n} \sum_{i=1}^n 1 \end{bmatrix}^\top = - \begin{bmatrix} \frac{1}{n} \sum_{i=1}^n x_i \\ 1 \end{bmatrix}^\top = - \begin{bmatrix} \bar{x} \\ 1 \end{bmatrix}^\top. \end{aligned} \quad (2.25)$$

Este resultado foi somente um exercício para treinar as regras das derivadas. Não podemos usar este gradiente para achar uma solução em forma fechada, pois já foi constatado que a primeira tentativa de definir a função de perda não seria uma boa escolha, por causa da compensação mútua

das diferenças. Além disso, no resultado, os parâmetros  $w$  e  $b$  que estamos procurando foram eliminados, fazendo a expressão inutilizável para achar os valores ótimos. Para economizar um pouco de espaço, vamos usar a abreviação

$$\sum_i x_i \stackrel{\text{def}}{=} \sum_{i=1}^n x_i, \quad (2.26)$$

se ficar óbvio sobre quais elementos o somatório é aplicado.

### Função de Perda: Segunda Tentativa

Para melhorar a definição da função da perda que não tenha o defeito da primeira definição da (eq. 2.23), vamos evitar que as diferenças entre o valor desejado  $y_i$  e o explicado  $\hat{y}_i$  correm perigo de serem compensados pelo sinal da diferença  $y_i - \hat{y}_i$  que pode ser às vezes positiva e às vezes negativa. Uma solução simples é usar o *valor absoluto* ou o *módulo* da diferença em vez da própria diferença. Se tiver uma discrepância, seja negativa, ou positiva, sempre vai contribuir à função de perda. Então, a definição melhorada de um termo da função de perda é

$$L_i(w, b) = |y_i - \hat{y}_i| = |y_i - (wx_i + b)| = \begin{cases} y_i - (wx_i + b), & \text{se } y_i > \hat{y}_i \\ 0, & \text{se } y_i = \hat{y}_i \\ wx_i + b - y_i, & \text{se } y_i < \hat{y}_i, \end{cases} \quad (2.27)$$

e a perda (risco empírico) da (eq. 2.12) é

$$L(w, b) = \frac{1}{n} \sum_i |y_i - \hat{y}_i| = \frac{1}{n} \sum_i |y_i - (wx_i + b)| \stackrel{1.13}{=} \frac{1}{n} \|\mathbf{y} - \hat{\mathbf{y}}\|_1, \quad (2.28)$$

onde os vetores  $\mathbf{y}$  e  $\hat{\mathbf{y}}$  de dimensão  $n$  contêm todos os  $n$  valores desejados e explicados, respectivamente. Nesta definição, a função de perda equivale então a norma- $\ell_1$  média da diferença entre o valor desejado e explicado. Surgiu um outro problema com o valor absoluto, pois a função  $|x|$  tem uma descontinuidade em  $x = 0$ . A sua derivada é

$$\frac{d|x|}{dx} = \begin{cases} 1, & \text{se } x > 0 \\ \text{indefinido} & \text{se } x = 0 \\ -1, & \text{se } x < 0. \end{cases} \quad (2.29)$$

Em termos práticos, o caso do argumento ser zero pode ser ignorado ou atribuído zero à derivada. Dessa maneira até é possível definir a derivada da função de perda, porém temos que fazer sempre uma distinção de casos, se o valor do termo é positivo ou negativo. Para aplicar esta distinção dos dois casos, vamos decompor a soma em (eq. 2.28) em  $n_+$  termos onde o valor desejado  $y_i$  é maior que, ou igual ao valor explicado  $\hat{y}_i$ , e em  $n_-$  termos onde o desejado é menor que o explicado. Temos então  $n = n_+ + n_-$  casos. Os índices correspondentes vamos chamar  $i_+$  e  $i_-$ . Assim, podemos reformular a perda (eq. 2.28), evitando o uso do valor absoluto, como

$$\begin{aligned} L(w, b) &= \frac{1}{n_+} \sum_{i_+} (y_{i_+} - \hat{y}_{i_+}) + \frac{1}{n_-} \sum_{i_-} (\hat{y}_{i_-} - y_{i_-}) \\ &= \frac{1}{n_+} \sum_{i_+} (y_{i_+} - wx_{i_+} - b) + \frac{1}{n_-} \sum_{i_-} (wx_{i_-} + b - y_{i_-}). \end{aligned} \quad (2.30)$$

Este tipo de decomposição pode ser reencontrado posteriormente neste livro no contexto de uma técnica de regressão e classificação chamada Máquina de Vetor de Suporte (*Support Vector Machine, SVM*; *Support Vector Regression, SVR*), [27, 98].

■ **Exemplo 2.4** Considerando o exemplo ilustrativo na tabela 2.1 e na figura 2.1 para fazer regressão linear, usando o resultado ótimo (a reta verde), temos  $n = n_+ + n_- = 7 = 4 + 3$ , e os índices são  $i_+ \in \{1, 2, 6, 7\}$  (pontos acima da reta verde) e  $i_- \in \{3, 4, 5\}$  (pontos abaixo da reta verde). As  $n = 7$  distâncias  $y_i - \hat{y}_i$  são  $\{0.18496, 0.24768, -0.237056, -0.516704, -0.30144, 0.303648, 0.318912\}$ . Para a primeira tentativa (eq. 2.23) da função de perda, temos realmente  $L(w, b) = 0.0$ , o que empiricamente prova que a simples diferença é inapropriada para medir o erro entre o desejado e explicado. Para a segunda tentativa (eq. 2.27), temos  $L(w, b) = L_+(w, b) + L_-(w, b) = 0.2638 + 0.3517\bar{3} = 0.6155\bar{3}$ . ■

Como exercício, sem uso prático para obter os parâmetros otimizados, podemos calcular o gradiente da função de perda, parecidamente com (eq. 2.25), considerando a decomposição

$$\begin{aligned}\nabla L(w, b) &= \frac{1}{n_+} \sum_{i_+} \nabla L_{i_+}(w, b) + \frac{1}{n_-} \sum_{i_-} \nabla L_{i_-}(w, b) \\ &= \left[ \frac{1}{n_+} \sum_{i_+} \frac{\partial y_{i_+} - wx_{i_+} - b}{\partial w} + \frac{1}{n_-} \sum_{i_-} \frac{\partial wx_{i_-} + b - y_{i_-}}{\partial w} \right]^\top = \left[ \frac{1}{n_+} \sum_{i_+} -x_{i_+} + \frac{1}{n_-} \sum_{i_-} x_{i_-} \right]^\top \\ &= \left[ -\frac{1}{n_+} \sum_{i_+} x_{i_+} + \frac{1}{n_-} \sum_{i_-} x_{i_-} \right]^\top \quad (2.31) \\ &= \begin{bmatrix} -\bar{x}_+ + \bar{x}_- \\ 0 \end{bmatrix}, \quad (2.32)\end{aligned}$$

onde  $\bar{x}_+$  é a média aritmética das abscissas  $x$  daqueles pontos acima da reta, e  $\bar{x}_-$  é a média aritmética das abscissas  $x$  daqueles pontos abaixo da reta. Os valores concretos em relação a tabela 2.1 são  $\bar{x}_+ = 1/4(x_1 + x_2 + x_6 + x_7) = 1/4(0.4 + 0.9 + 3.1 + 3.7) = 2.025$  e  $\bar{x}_- = 1/3(x_3 + x_4 + x_5) = 1/3(1.5 + 2.3 + 2.9) = 2.2\bar{3}$ , então  $-\bar{x}_+ + \bar{x}_- = 0.208\bar{3}$ . Este gradiente, como o da primeira tentativa é inútil para determinar os parâmetros ótimos. Observe que o gradiente de  $b$  é zero independente dos dados de treinamento, assim não pode servir como base para ajustar os parâmetros livres do modelo. Isso fica claro agora na apresentação de uma função de perda conveniente, a mais usada no contexto da otimização, a do erro quadrático médio.

Contudo, vale destacar o termo do gradiente individual da função de perda  $\nabla L_i(w, b)$  no sentido da distinção de casos da (eq. 2.27). Então

$$\nabla L_i(w, b) = \begin{cases} -\begin{bmatrix} x_i \\ 1 \end{bmatrix}^\top, & \text{se } y_i > \hat{y}_i \\ \begin{bmatrix} 0 \\ 0 \end{bmatrix}^\top, & \text{se } y_i = \hat{y}_i \\ \begin{bmatrix} x_i \\ 1 \end{bmatrix}^\top, & \text{se } y_i < \hat{y}_i. \end{cases} \quad (2.33)$$

O caso da igualdade  $y_i = \hat{y}_i$  do valor esperado e explicado, em caso de valores contínuos  $y_i \in \mathbb{R}$  é muito raro.

Se o domínio for binário, ou seja,  $y_i \in \{-1, 1\}$ , a igualdade de  $y_i$  e  $\hat{y}_i$  significa que não há erro, veja (eq. 2.27). Em analogia à distinção de casos da (eq. 2.33) temos para o caso binário

$$\operatorname{sgn}(y_i - \hat{y}_i) = \begin{cases} \operatorname{sgn}(1 - -1) = \operatorname{sgn}(2) = +1, & \text{se } y_i = 1, \hat{y}_i = -1 \\ 0, & \text{se } y_i = \hat{y}_i \\ \operatorname{sgn}(-1 - 1) = \operatorname{sgn}(-2) = -1, & \text{se } y_i = -1, \hat{y}_i = 1. \end{cases} \quad (2.34)$$

Assim, combinando com a (eq. 2.34), para todos os três casos, podemos unificar o gradiente da função de perda individual da (eq. 2.33) como

$$\nabla L_i(w, b) = -\text{sgn}(y_i - \hat{y}_i) \begin{bmatrix} x_i \\ 1 \end{bmatrix}^\top, \quad (2.35)$$

pois a expressão  $\text{sgn}(y_i - \hat{y}_i)$  fornece sempre o sinal certo nos três casos da (eq. 2.33). Ao contrário do gradiente da função de perda da (eq. 2.32) que usa todos os  $n$  padrões e é sem uso prático, vamos reencontrar o gradiente individual da (eq. 2.35) quando estudaremos algoritmos iterativos para achar o mínimo de uma função de perda. O gradiente individual (eq. 2.35) da função de perda de um único padrão é o coração do algoritmo de aprendizagem do Perceptron [90], praticamente a primeira rede neural artificial usada para classificar. Já podemos antecipar que as perdas individuais relativas a cada padrão  $L_i(\boldsymbol{\theta})$  têm um papel importante em algoritmos iterativos de aprendizado de máquina. Podemos ajustar os parâmetros  $\boldsymbol{\theta}$  após calcular o gradiente da função de perda individual  $\nabla L_i(\boldsymbol{\theta})$ . Esta técnica será denominada *descida de gradiente estocástica*.

### Função de Perda do Erro Quadrático Médio: Terceira Tentativa

O Erro Quadrático Médio (EQM), em inglês *Mean Squared Error, MSE* permite obter uma solução ótima em forma fechada para um modelo linear. Os termos individuais da função de perda são definidos como os *quadrados* das diferenças entre o valor desejado  $y_i$  e o valor explicado  $\hat{y}_i$  pelo modelo linear

$$L_i(w, b) = (y_i - \hat{y}_i)^2 = (y_i - (wx_i + b))^2. \quad (2.36)$$

A função de perda<sup>2</sup>, considerando todos os  $n$  padrões então é

$$L(w, b) = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \stackrel{*}{=} \frac{1}{n} \|\mathbf{y} - \hat{\mathbf{y}}\|^2, \quad (2.37)$$

onde os vetores  $\mathbf{y}$  e  $\hat{\mathbf{y}}$  de dimensão  $n$  contêm todos os  $n$  valores desejados e explicados, respectivamente. A (eq. 1.11) na pág. 14 foi usada na última igualdade. Comparado com as primeiras duas tentativas de definir uma função de perda, o EQM tem duas vantagens. Primeiramente, as diferenças  $y_i - \hat{y}_i$  não se compensam, pois é calculado o quadrado desta diferença. Como segundo ponto positivo da EQM pode-se destacar que a derivada da função quadrática  $f(g) = g^2$  é fácil de ser calculada como  $f'(g) = 2g$  e a derivada é contínua em todo o domínio onde  $f$  for contínua. Mesmo em caso de não linearidade do modelo, o EQM é a escolha mais frequente para medir a discrepância entre o resultado desejado e explicado pelo modelo.

Neste ponto surge a necessidade de definir uma regra extremamente importante para ajustar os parâmetros, especialmente em uma sequência de operações matemáticas que constituem o modelo. Um dos casos mais típicos é o cálculo de um produto escalar, seguido por uma passagem do resultado por uma *função de ativação* que tem um escalar como entrada e um escalar como saída.

**Definição 2.1.8 — Regra da Cadeia.** Seja  $g$  uma função e  $f$  outra função. Os domínios de origem e destino das funções no nosso caso mais simples são os números reais  $\mathbb{R}$ , ou seja  $f : \mathbb{R} \rightarrow \mathbb{R}$  e  $g : \mathbb{R} \rightarrow \mathbb{R}$ . A *cadeia* de cálculo é primeiro calcular o valor  $g(x)$  da função de uma variável  $x$ , e em seguida alimentar este valor como argumento na função  $f$ , ou seja calcular

$$f(g(x)).$$

<sup>2</sup>Seria possível usar nesta definição a *soma* dos termos individuais  $L_i(w, b)$ , em vez da *média*, simplificando o cálculo um pouco. Pode-se afirmar que a solução ótima do menor valor da função de perda tem a mesma solução, seja a soma, ou seja a média dos termos.

Uma forma alternativa de expressar a cadeia, com a função mais interna à direita, é

$$f \circ g.$$

A derivada<sup>a</sup> da função  $f$  em relação ao argumento interior  $x$  usa a regra da cadeia

$$\frac{df}{dx} = \frac{df}{dg} \cdot \frac{dg}{dx}. \quad (2.38)$$

<sup>a</sup>A notação de Leibniz usa o conceito de infinitesimalmente pequeno “ $d$ ”. Então  $dx$  é uma diferença “muito muito pequena”  $dx = x_b - x_a$  entre dois valores  $x_a$  e  $x_b$ , e  $df$  é a diferença  $df = f(x_b) - f(x_a)$  entre os valores da função. O fato mais importante é que, mesmo se tratando de diferenças infinitesimais, a relação  $df/dx$  mantém um valor plausível. Além disso, o conceito se estende à dimensões mais altas. Por exemplo, no plano,  $df/dx$  significa a mudança bidimensional em um quadrado  $dx$  infinitesimalmente pequeno.

Parece que simplesmente a expressão  $df/dx$  do lado esquerdo da equação foi multiplicada pelo valor um, ou seja pelo valor  $dg/dg = 1$ . De fato, quando a complexidade dos modelos aumentar, podemos usar essa ideia para facilitar os cálculos. Uma ajuda para memorizar esta regra é denominar a função do interior  $g(x)$  como algum objeto, por exemplo “batata”. Então a derivada do quadrado de “batata” seria duas vezes “batata”, vezes a derivada interna de “batata” em relação a variável de interesse.

■ **Exemplo 2.5**  $\frac{df}{dx} = \frac{d[\sin x]^2}{dx} = 2 \sin x \cos x$ . Aqui a função  $f$  é a função externa que calcula o quadrado do seu argumento  $f(g) = g^2$ . Não interessa neste momento que  $g$  por sua vez é uma função  $g = g(x) = \sin(x)$ . Então temos o primeiro termo  $\frac{df}{dg} = 2g = 2 \sin(x)$ . O segundo termo  $\frac{dg}{dx}$  calcula a derivada interna  $\frac{dg}{dx} = \frac{d \sin(x)}{dx} = \cos(x)$ . ■

■ **Exemplo 2.6** Queremos calcular a derivada de um função, onde a variável independente  $x$  está “enterrada” mais profundamente na cadeia, neste caso primeiramente calcula-se o quadrado da variável  $x$ , depois o logaritmo natural (neperiano), e finalmente o valor cúbico.

$$\frac{d(\ln x^2)^3}{dx} = \frac{6}{x} (\ln x^2)^2.$$

Este resultado é explicável pela regra da cadeia, se considerarmos a cadeia de três funções  $f(g(h(x)))$ , com

$$f(g) \stackrel{\text{def}}{=} g^3, \quad g(h) \stackrel{\text{def}}{=} \ln h, \quad h(x) \stackrel{\text{def}}{=} x^2.$$

Temos que inserir mais um termo na derivada, assim resultando na regra

$$\frac{df}{dx} = \frac{df}{dg} \cdot \frac{dg}{dh} \cdot \frac{dh}{dx}.$$

As derivadas individuais usadas foram

$$\frac{df(g)}{dg} = 3g^2, \quad \frac{dg(h)}{dh} = \frac{1}{h}, \quad \frac{dh(x)}{dx} = 2x.$$

Então

$$\begin{aligned}
 \frac{df}{dx} &= \frac{df}{dg} \cdot \frac{dg}{dh} \cdot \frac{dh}{dx} \\
 &= \frac{dg^3}{dg} \cdot \frac{d \ln h}{dh} \cdot \frac{dx^2}{dx} \\
 &= 3g^2 \cdot \frac{1}{h} \cdot 2x \\
 &= 3(\ln h)^2 \cdot \frac{1}{x^2} \cdot 2x \\
 &= 3(\ln x^2)^2 \cdot \frac{1}{x^2} \cdot 2x = \frac{6}{x} (\ln x^2)^2.
 \end{aligned}$$

■

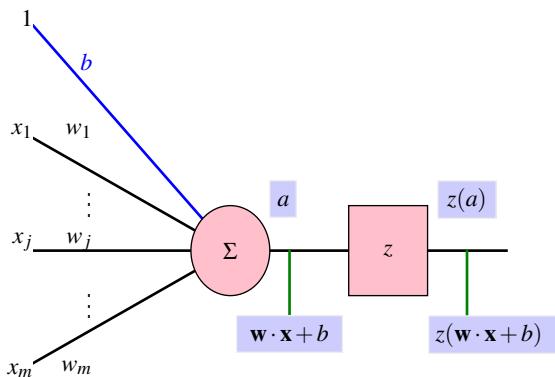


Figura 2.2: Modelo de neurônio, unidade básica de uma rede neural artificial. Cada entrada é multiplicada por um peso, somando estes termos, resultando no produto escalar do vetor de entrada  $\mathbf{x}$  com o vetor de pesos  $\mathbf{w}$ . O viés  $b$  pode ser considerado como mais um peso, multiplicado por uma entrada fixa de valor um, ou seja,  $x_0 = 1$ . Foi usado o símbolo de um somatório, o sigma maiúsculo “ $\Sigma$ ”, para expressar o fato que o cálculo básico de um neurônio é a combinação das entradas com os pesos. Após o produto escalar, mais o viés, o resultado  $\mathbf{w} \cdot \mathbf{x} + b$  passa por uma função de ativação  $z$ .

■ **Exemplo 2.7** Considere o caso exemplar em uma rede neural da figura 2.2. Um vetor de entrada  $\mathbf{x}$  e o vetor de pesos  $\mathbf{w}$  da mesma dimensão  $m$  são linearmente combinados em um produto escalar, ainda somando um viés  $b$ . Vamos chamar a função que realiza esse cálculo básico de um *neurônio* como a *ativação*  $a$ , portanto<sup>3</sup>

$$a(\mathbf{x}; \mathbf{w}, b) \stackrel{\text{def}}{=} \mathbf{w} \cdot \mathbf{x} + b = w_1 x_1 + \cdots + w_j x_j + \cdots + w_m x_m + b. \quad (2.39)$$

<sup>3</sup>De fato, a letra “d” seria também apropriada, pois  $g(\mathbf{x}; \mathbf{w}, b)$  é a *distância* Euclidiana (ainda multiplicada por  $\|\mathbf{w}\|$ ) do ponto  $\mathbf{x}$  até o hiperplano definido pelos parâmetros  $\mathbf{w}$  e  $b$ . Para não confundir com a letra “d” normalmente usada para expressar uma derivada, usa-se aqui a letra “a”.

Repare que um ponto e vírgula foi usado para dizer que  $\mathbf{x}$  tem o papel de argumento e  $\mathbf{w}$  e  $b$  têm o papel dos parâmetros. A derivada parcial<sup>4</sup> desta função em relação a qualquer um dos pesos  $w_j$  é

$$\begin{aligned}\frac{\partial a(\mathbf{x}; \mathbf{w}, b)}{\partial w_j} &= \frac{\partial (\mathbf{w} \cdot \mathbf{x} + b)}{\partial w_j} = \frac{\partial (w_1 x_1 + \dots + w_j x_j + \dots + w_m x_m + b)}{\partial w_j} \\ &= \frac{\partial w_1 x_1}{\partial w_j} + \dots + \frac{\partial w_j x_j}{\partial w_j} + \dots + \frac{\partial w_m x_m}{\partial w_j} + \frac{\partial b}{\partial w_j} \\ &= 0 + \dots + x_j + \dots + 0 + 0 \\ &= x_j,\end{aligned}\tag{2.40}$$

pois somente o termo  $w_j x_j$  depende do peso  $w_j$  em questão, e todos os restantes termos são constantes em relação a  $w_j$ . A derivada parcial do cálculo do neurônio em relação ao viés  $b$  é

$$\frac{\partial a(\mathbf{x}; \mathbf{w}, b)}{\partial b} = \frac{\partial (\mathbf{w} \cdot \mathbf{x} + b)}{\partial b} = \frac{\partial (\mathbf{w} \cdot \mathbf{x})}{\partial b} + \frac{\partial b}{\partial b} = 0 + 1 = 1.\tag{2.41}$$

Se formarmos um vetor que contém essas derivadas parciais para cada peso  $w_j$  e para o viés  $b$ , reconhecemos que é o gradiente (eq. 2.20) da função  $g$  em relação aos parâmetros  $(\mathbf{w}, b)$

$$\left[ \begin{array}{c} \frac{\partial a(\mathbf{x}; \mathbf{w}, b)}{\partial w_1} \\ \vdots \\ \frac{\partial a(\mathbf{x}; \mathbf{w}, b)}{\partial w_j} \\ \vdots \\ \frac{\partial a(\mathbf{x}; \mathbf{w}, b)}{\partial w_m} \\ \frac{\partial a(\mathbf{x}; \mathbf{w}, b)}{\partial b} \end{array} \right]^\top = \nabla g = \text{grad } g = \frac{\partial a(\mathbf{x}; \mathbf{w}, b)}{\partial (\mathbf{w}, b)} = \left[ \begin{array}{c} x_1 \\ \vdots \\ x_j \\ \vdots \\ x_m \\ 1 \end{array} \right]^\top = \left[ \begin{array}{c} \mathbf{x} \\ 1 \end{array} \right]^\top.\tag{2.42}$$

Para facilitar este cálculo da derivada, e muitos outros cálculos que seguirão, podemos considerar uma fusão dos pesos  $w_1, \dots, w_m$  com o viés  $b$  em um vetor único, ainda colocando  $b$  antes dos pesos em um vetor  $\tilde{\mathbf{w}}$  de pesos *aumentado*. Podemos assim chamar o viés como 0-ésimo peso  $b = w_0$ . Temos que criar também uma nova 0-ésima entrada  $x_0$  que será multiplicada por  $b$ . Considerando a combinação linear na (eq. 2.39), podemos facilmente identificar este valor como  $x_0 = 1$ . Criamos um vetor  $\tilde{\mathbf{x}}$  de entrada *aumentado*. Dessa forma temos

**Definição 2.1.9 — Vetor de Pesos Aumentado e Vetor de Entrada Aumentado.** Seja  $b$  o viés, e sejam  $w_j, j = 1, \dots, m$  os  $m$  pesos de uma combinação linear, e sejam  $x_j, j = 1, \dots, m$  as  $m$  entradas. Então os dois vetores  $\tilde{\mathbf{w}}$  e  $\tilde{\mathbf{x}}$  de dimensão  $(m+1)$ , onde nos vetores de coluna originais  $\mathbf{w}$  e  $\mathbf{x}$ , antes da primeira linha é criada uma nova linha com os valores  $w_0 \stackrel{\text{def}}{=} b$ , respectivamente  $x_0 \stackrel{\text{def}}{=} 1$  são os vetores aumentados de pesos e entradas.

$$\tilde{\mathbf{w}} \stackrel{\text{def}}{=} \begin{bmatrix} b \\ w_1 \\ \vdots \\ w_j \\ \vdots \\ w_m \end{bmatrix} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_j \\ \vdots \\ w_m \end{bmatrix} = \begin{bmatrix} b \\ \mathbf{w} \end{bmatrix}, \quad \tilde{\mathbf{x}} \stackrel{\text{def}}{=} \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_j \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_j \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix},\tag{2.43}$$

<sup>4</sup>Sempre quando a função depende de mais que um argumento, é melhor usar o símbolo “ $\partial$ ” da derivada parcial.

A combinação linear e seu gradiente podem ser reformulados em forma mais compacta como

$$\begin{aligned} a(\mathbf{x}; \mathbf{w}, b) &= \mathbf{w} \cdot \mathbf{x} + b = w_1 x_1 + \cdots + w_j x_j + \cdots + w_m x_m + b \\ &= w_0 x_0 + w_1 x_1 + \cdots + w_j x_j + \cdots + w_m x_m = \tilde{\mathbf{w}} \cdot \tilde{\mathbf{x}} \stackrel{\text{def}}{=} \tilde{a}(\tilde{\mathbf{x}}; \tilde{\mathbf{w}}) \\ \nabla \tilde{a} &= \text{grad } \tilde{a} = \frac{\partial \tilde{a}(\tilde{\mathbf{x}}; \tilde{\mathbf{w}})}{\partial \tilde{\mathbf{w}}} = \tilde{\mathbf{x}}^\top. \end{aligned} \quad (2.44)$$

Essa ideia da representação aumentada pode ser facilmente reconhecida na figura 2.2, onde existe uma entrada fixa  $x_0 = 1$  e o viés  $b$  está graficamente na mesma posição que os restantes pesos.

Em um neurônio artificial, após o cálculo do produto escalar entre pesos  $\mathbf{w}$  e entradas  $\mathbf{x}$ , e a soma do viés, o próximo passo é fornecer esse valor  $\tilde{\mathbf{w}} \cdot \tilde{\mathbf{x}} = \mathbf{w} \cdot \mathbf{x} + b$  como entrada em uma função de ativação  $z$ . Essa função é unidimensional, ela recebe um escalar como entrada e calcula um escalar como saída. Estamos ainda no contexto de um modelo linear, mas em geral uma função de ativação pode ser não linear. Uma função de ativação importante é a função sigmóide logística

$$z(a) = \frac{1}{1 + e^{-a}}. \quad (2.45)$$

A derivada desta função

$$\frac{dz(a)}{da} = z'(a) \quad (2.46)$$

é a derivada unidimensional. Uma particularidade que no momento não é relevante é que a derivada desta função é diretamente expressável pela própria função como

$$z'(a) = z(a)[1 - z(a)]. \quad (2.47)$$

Então, se existe interesse como a saída do neurônio é influenciada por um peso particular  $w_j$  ou viés, temos que calcular a derivada através da regra da cadeia como

$$\frac{\partial z}{\partial w_j} = \frac{dz}{dg} \cdot \frac{\partial g}{\partial w_j} = z'(g)x_j, \quad \frac{\partial z}{\partial b} = \frac{dz}{dg} \cdot \frac{\partial g}{\partial b} = z'(g) \cdot 1,$$

onde foi usado o resultado da (eq. 2.40), ou usando todo o vetor de pesos  $\mathbf{w}$ , usado o resultado da (eq. 2.42), como

$$\frac{dz}{d\mathbf{w}} = \frac{dz}{dg} \cdot \frac{dg}{d\mathbf{w}} = z'(g)\mathbf{x}^\top.$$

Usando a representação dos vetores aumentados, unificando os pesos e o viés, temos

$$\frac{dz}{d\tilde{\mathbf{w}}} = \frac{dz}{d\tilde{a}} \cdot \frac{d\tilde{a}}{d\tilde{\mathbf{w}}} = z'(\tilde{a})\tilde{\mathbf{x}}^\top.$$

Repare o uso da derivada parcial, se o interesse é na dependência de um único peso, ou a derivada vetorial, se o interesse é na dependência do vetor de pesos inteiro (somente  $\mathbf{w}$ , ou aumentado  $\tilde{\mathbf{w}}$ ). Esse exemplo do cálculo da derivada de um peso que fica mais aninhado no interior de uma cadeia de operações, é uma peça chave no algoritmo de *retropropagação de erro, error backpropagation*. Revemos esse cálculo no contexto do *perceptron multicamada*, uma das arquiteturas principais de redes neurais artificiais. ■

Talvez o leitor ache a quantidade e profundidade dos exemplos exagerado. Porém já foi mencionado que a regra da cadeia tem um papel fundamental no treinamento de redes neurais com

camadas ocultas que precisam analisar como a discrepância entre os valores desejados e explicados pela rede dependem dos pesos da rede.

Vamos então mais uma vez calcular o gradiente da função de perda  $L(w, b)$  definida na (eq. 2.37), necessitando agora a regra da cadeia.

$$\begin{aligned}\nabla L(w, b) &= \frac{1}{n} \sum_i \nabla L_i(w, b) = \left[ \frac{1}{n} \sum_i \frac{\partial L_i(w, b)}{\partial w}, \frac{1}{n} \sum_i \frac{\partial L_i(w, b)}{\partial b} \right]^\top = \frac{1}{n} \left[ \sum_i \frac{\partial (y_i - wx_i - b)^2}{\partial w}, \sum_i \frac{\partial (y_i - wx_i - b)^2}{\partial b} \right]^\top \\ &= \frac{1}{n} \left[ \sum_i 2(y_i - wx_i - b)(-x_i) \right]^\top = \frac{2}{n} \left[ (\sum_i wx_i^2 + \sum_i bx_i - \sum_i x_i y_i) \right]^\top \\ &= \frac{2}{n} \left[ \begin{matrix} (w \sum_i x_i^2 + b \sum_i x_i - \sum_i x_i y_i) \\ (w \sum_i x_i + b n - \sum_i y_i) \end{matrix} \right].\end{aligned}\quad (2.48)$$

Deste resultado ainda podemos reconhecer o gradiente da função de perda individual de cada padrão como

$$\nabla L_i(w, b) = \begin{bmatrix} -2(y_i - wx_i - b)x_i \\ -2(y_i - wx_i - b) \end{bmatrix}. \quad (2.49)$$

Como já mencionado este gradiente pode ser usado em algoritmos iterativos que ajustam os parâmetros após a apresentação de um padrão individual.

Falta um último passo para conectar este gradiente com a solução do problema de regressão linear simples da (eq. 2.7).

### Solução Ótima de Modelo Linear: Zero Gradiente

Para determinar o extremo (mínimo ou máximo) de uma função  $f(x)$ , o caminho normalmente é zerar sua derivada,  $f'(x) = 0$ , e resolver esta equação segundo a variável escalar  $x$ , ou seja determinar a raiz da derivada. Isso é uma *condição necessária* de primeira ordem para ter um extremo em uma vizinhança da solução [78]. No caso de um modelo linear, a condição necessária é simultaneamente uma *condição satisfatória*, pois a função de perda quadrática é uma função *estritamente convexa*. Essas propriedades serão explicadas na apresentação de métodos iterativos de minimização de funções vetoriais quadráticas em um instante posterior. Como exemplo, no caso da função  $f(x) = x^2 - 1$ , zerar a derivada significa estabelecer a equação  $f'(x) = 2x = 0$  cuja solução é a raiz  $x^* = 0$ . Neste caso temos o *mínimo* em  $x^* = 0$ . A derivada da função  $\tilde{f}(x) = -x^2 - 1$ , ou seja,  $f(x)$  espelhada em  $y = -1$ , tem a mesma propriedade da derivada zerada em  $x = 0$ , porém é um *máximo*. Para distinguir entre máximo e mínimo, precisamos da segunda derivada. Estamos principalmente interessados no menor valor de função, pois o foco é uma função de perda que tem que ser minimizada. Então vamos nos restringir a este caso.

Para uma função vetorial, aquela que tem um argumento  $\mathbf{x}$  multidimensional, a *condição necessária* é a mesma, zerar a derivada que neste caso é o gradiente  $\nabla f$  da função  $f(\mathbf{x})$ .

**Definição 2.1.10 — Condição Necessária para o Mínimo de Função Vetorial.** O vetor  $\mathbf{x}^*$  minimiza a função vetorial  $f(\mathbf{x})$  em uma vizinhança aberta e suficientemente pequena, se

$$\nabla f(\mathbf{x}^*) = \mathbf{0}^\top = [0 \quad \cdots \quad 0 \quad \cdots \quad 0] \quad (2.50)$$

ou seja, o valor ótimo  $\mathbf{x}^*$  é o argumento da função que minimiza a função  $f(\mathbf{x})$

$$\mathbf{x}^* = \arg \min_{\mathbf{x}} f(\mathbf{x}). \quad (2.51)$$

Zerar o gradiente então significa montar um *sistema de equações*. Para cada derivada parcial, temos a  $m$ -ésima equação  $0 = \partial f(\mathbf{x})/\partial x_j$ .

Como exemplo, zerar o gradiente da função  $f(\mathbf{x}) = f\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = 3x_1^2x_2 + x_2^3$  na (eq. 2.21), implica

$$\nabla f(\mathbf{x}) = [\partial f(\mathbf{x})/\partial x_1 \quad \partial f(\mathbf{x})/\partial x_2] = [6x_1x_2 \quad 3x_1^2 + 3x_2^2] = [0 \quad 0],$$

com a única solução

$$\mathbf{x}^* = [0 \quad 0]^\top.$$

A primeira restrição  $6x_1x_2 = 0$  significa que ou  $x_1$ , ou  $x_2$  seja zero, e, junto com a segunda  $x_1^2 + x_2^2 = 0$  não permite outra solução que  $\mathbf{x}^* = [0 \quad 0]^\top$ .

Observando as duas equações  $0 = \partial L(w, b)/\partial w$  e  $0 = \partial L(w, b)/\partial b$  na (eq. 2.48), podemos ver que elas são lineares em relação às duas variáveis  $w$  e  $b$ . Isso significa que temos um sistema de duas equações lineares. Pode ser facilmente resolvido, usando umas das técnicas consolidadas para cálculo numérico [12, 39], neste caso, por exemplo, a eliminação de Gauss e resolução retroativa de um sistema triangular. Multiplicando ainda os dois lados das duas equações  $0 = \partial L(w, b)/\partial w$  e  $0 = \partial L(w, b)/\partial b$  da (eq. 2.48) por  $\frac{n}{2}$ , e movendo os últimos termos  $\sum_i x_i y_i$  e  $\sum_i y_i$  para o lado direito das equações, temos o sistema  $2 \times 2$

$$\begin{bmatrix} \sum_i x_i & \sum_i x_i^2 \\ n & \sum_i x_i \end{bmatrix} \begin{bmatrix} w \\ b \end{bmatrix} = \begin{bmatrix} \sum_i x_i y_i \\ \sum_i y_i \end{bmatrix}. \quad (2.52)$$

Para verificar a equivalência deste sistema com a (eq. 2.48) (ignorando o fator  $2/n$ ), multiplique a matriz pelo vetor, subtraia o vetor da direita e compare com as expressões das duas equações em (eq. 2.48). A solução deste sistema linear é exatamente o resultado apresentado anteriormente na (eq. 2.7). Podemos resumir que existe uma forma fechada para obter a solução ótima de um modelo linear, se a função de perda é o erro quadrático médio (EQM). Não precisamos um algoritmo iterativo para calcular os parâmetros ótimos  $\boldsymbol{\theta}^*$ . Basta montar um sistema linear e resolvê-lo pelas técnicas tradicionais de cálculo numérico.

Essa estratégia será seguida no treinamento de um classificador linear e em todas as outras arquiteturas de regressão ou classificação que tenham uma parte final linear. Um exemplo interessante é o treinamento da última camada da Máquina Extrema de Aprendizado, *Extreme Learning Machine, ELM* um tipo de rede neural que, embora tenha uma estrutura simples, alcança resultados excelentes.

Já foi constatado na (eq. 2.8) que os valores ótimos dos dois parâmetros que minimizam a função de perda têm os valores  $\boldsymbol{\theta}^* = (w^*, b^*) = (0.475, 0.325)$ . A figura 2.3 mostra a função de perda do erro quadrático médio variando  $w$  e  $b$  na vizinhança do valor ótimo. Do lado esquerdo, o eixo  $z$  é a função dos dois parâmetros. O ponto vermelho marca o valor ótimo  $\boldsymbol{\theta}^*$ . Do lado direito, o valor do segundo parâmetro é fixado em  $b^* = 0.325$ , e somente o parâmetro  $w$  é variado. Fixando um dos parâmetros, vira uma parábola. Nos gráficos fica clara a propriedade da convexidade estrita da função de perda. Em uma analogia física, uma bola colocada em qualquer ponto da função, rolaría *sempre* para o mínimo  $(w^*, b^*)$ . Não existem mínimos locais da função, somente um mínimo global. Uma função não estritamente convexa pode ter vários mínimos globais com o mesmo valor. Na analogia da bola, existe uma calha onde a bola pode ficar em qualquer lugar.

### Minimização via Pseudoinversa da Matriz de Dados

Usamos a minimização do erro quadrático médio como critério para calcular os mais apropriados valores dos parâmetros livres do modelo linear. Existe uma alternativa para obter os parâmetros

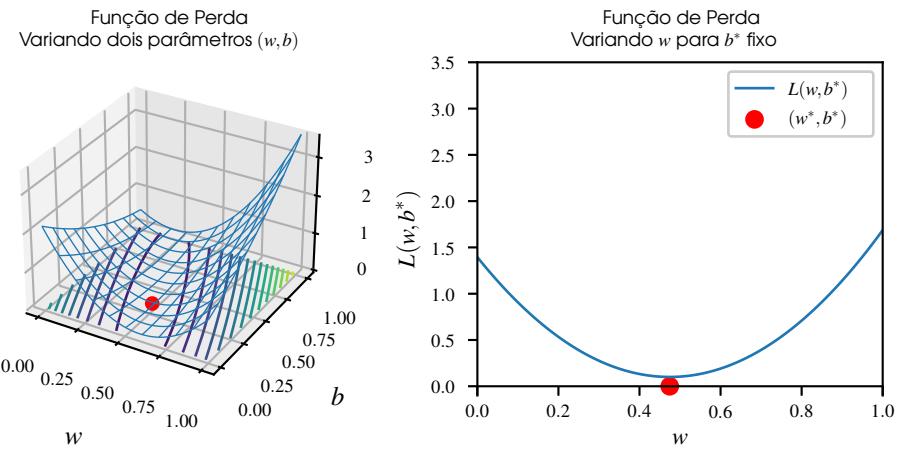


Figura 2.3: Função de perda do erro quadrático médio em um problema de regressão linear. A função é estritamente convexa e tem somente um mínimo.

ótimos  $\theta = (w, b)$  que envolve a combinação de álgebra linear com derivadas multidimensionais. Permite uma formulação compacta dos métodos de treinamento em um problema de regressão ou classificação linear. Por exemplo em uma arquitetura de classificador chamada *Máquina Linear*, ou na já mencionada *Extreme Learning Machine*, a obtenção de uma matriz  $W$  que abriga todos os pesos e vieses é uma equação em forma fechada que envolve somente a matriz de dados  $X$  e as saídas desejadas  $Y$ . Precisamos ainda algumas regras [24].

**Definição 2.1.11 — Derivadas Multidimensionais.** Seja  $\mathbf{x}$  uma variável de dimensão  $m$  em forma de vetor de coluna, seja  $A$  uma matriz que não é uma função de  $\mathbf{x}$ , e sejam  $\mathbf{u}(\mathbf{x})$  e  $\mathbf{v}(\mathbf{x})$  funções de  $\mathbf{x}$ . Essas funções podem, no caso mais simples, ser funções escalares  $u : \mathbb{R}^m \rightarrow \mathbb{R}$  e  $v : \mathbb{R}^m \rightarrow \mathbb{R}$  com um contradomínio de uma dimensão, ou em geral podem ser funções com um contradomínio multidimensional. Estamos interessados na derivada parcial de uma expressão envolvendo  $\mathbf{x}$ ,  $A$ ,  $\mathbf{u}$  e  $\mathbf{v}$ . Então

$$\frac{\partial}{\partial \mathbf{x}} (\mathbf{u}^\top A \mathbf{v}) = \mathbf{u}^\top A \frac{\partial \mathbf{v}}{\partial \mathbf{x}} + \mathbf{v}^\top A^\top \frac{\partial \mathbf{u}}{\partial \mathbf{x}}. \quad (2.53a)$$

Esta definição é bem geral e pode ser especializada em vários casos mais simples<sup>a</sup>. O primeiro caso especial é quando  $\mathbf{u}$  não é uma função de  $\mathbf{x}$  e  $\mathbf{v}$  é a função de identidade  $\mathbf{v}(\mathbf{x}) = \mathbf{x}$ . Então as derivadas simplificam para  $\partial \mathbf{u} / \partial \mathbf{x} = 0$  e  $\partial \mathbf{v} / \partial \mathbf{x} = \partial \mathbf{x} / \partial \mathbf{x} = 1$ . Então

$$\frac{\partial (\mathbf{u}^\top A \mathbf{x})}{\partial \mathbf{x}} = \mathbf{u}^\top A \frac{\partial \mathbf{x}}{\partial \mathbf{x}} + \mathbf{v}^\top A^\top \frac{\partial \mathbf{u}}{\partial \mathbf{x}} = \mathbf{u}^\top A, \quad \text{se } \mathbf{u} \neq \mathbf{u}(\mathbf{x}), \mathbf{v}(\mathbf{x}) = \mathbf{x}. \quad (2.53b)$$

Podemos ignorar  $\mathbf{u}$  completamente, se  $\mathbf{u}(\mathbf{x}) = I$ , ou seja  $\mathbf{u}$  sempre retorna a matriz de identidade

$I$  que não é uma função de  $\mathbf{x}$ , com  $I = \text{diag}(1, 1, \dots, 1) = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix}$ . Então o primeiro

caso simplifica mais ainda para

$$\frac{\partial(A\mathbf{x})}{\partial \mathbf{x}} = IA \frac{\partial \mathbf{x}}{\partial \mathbf{x}} + 0 = A, \quad \text{se } \mathbf{u}(\mathbf{x}) = I, \mathbf{v}(\mathbf{x}) = \mathbf{x}. \quad (2.53c)$$

Em um terceiro caso importante,  $\mathbf{u}$  e  $\mathbf{v}$  são funções de identidade. Temos

$$\frac{\partial(\mathbf{x}^T A \mathbf{x})}{\partial \mathbf{x}} = \mathbf{x}^T A \frac{\partial \mathbf{x}}{\partial \mathbf{x}} + \mathbf{x}^T A^T \frac{\partial \mathbf{x}}{\partial \mathbf{x}} = \mathbf{x}^T (A + A^T) \quad \text{se } \mathbf{u}(\mathbf{x}) = \mathbf{x}, \mathbf{v}(\mathbf{x}) = \mathbf{x}. \quad (2.53d)$$

Se a matriz  $A$  for uma matriz simétrica, como, por exemplo, uma matriz de covariância (eq. 1.43) na pág. 27, ela é idêntica a sua transposta, ou seja  $A = A^T$ . Nesse caso temos

$$\frac{\partial(\mathbf{x}^T A \mathbf{x})}{\partial \mathbf{x}} = \mathbf{x}^T (A + A^T) = 2\mathbf{x}^T A \quad \text{se } \mathbf{u}(\mathbf{x}) = \mathbf{x}, \mathbf{v}(\mathbf{x}) = \mathbf{x}, A = A^T. \quad (2.53e)$$

Outro caso importante acontece quando a matriz  $A$  simplifica para a matriz de identidade, ou seja,  $A = I$ . Além disso, lembrando que o produto escalar  $\mathbf{x} \cdot \mathbf{x} = \mathbf{x}^T \mathbf{x}$  de um vetor  $\mathbf{x}$  com ele próprio, definido na (eq. 1.4) na pág. 12, é o quadrado da norma  $\mathbf{x} \cdot \mathbf{x} = \|\mathbf{x}\|^2$ , temos

$$\frac{\partial(\mathbf{x}^T \mathbf{x})}{\partial \mathbf{x}} = \frac{\partial \|\mathbf{x}\|^2}{\partial \mathbf{x}} = 2\mathbf{x}^T. \quad \text{se } \mathbf{u}(\mathbf{x}) = \mathbf{x}, \mathbf{v}(\mathbf{x}) = \mathbf{x}, A = I. \quad (2.53f)$$

Um último caso importante é quando a função  $\mathbf{u}$  depende de  $\mathbf{x}$ , e a derivada é calculada em relação a essa variável interna  $\mathbf{x}$ .

$$\frac{\partial(\mathbf{u}^T \mathbf{u})}{\partial \mathbf{x}} = \mathbf{u}^T \frac{\partial \mathbf{u}}{\partial \mathbf{x}} + \mathbf{u}^T \frac{\partial \mathbf{u}}{\partial \mathbf{x}} = 2\mathbf{u}^T \frac{\partial \mathbf{u}}{\partial \mathbf{x}} \quad \text{se } \mathbf{u}(\mathbf{x}) \neq \mathbf{x}, \mathbf{v}(\mathbf{x}) = \mathbf{u}(\mathbf{x}), A = I. \quad (2.53g)$$

<sup>a</sup>A derivadas de funções com entrada e saída multidimensionais  $\frac{\partial \mathbf{u}}{\partial \mathbf{x}}$  e  $\frac{\partial \mathbf{v}}{\partial \mathbf{x}}$  são as matrizes Jacobianas, posteriormente definidas na definição 3.3.3 na pág. 98.

Especialmente a última regra (eq. 2.53g) merece uma atenção especial. Ela entra sempre em ação quando se mede a função de perda na saída de um modelo multidimensional, não somente um modelo linear, e a função de perda é o erro quadrático. Um exemplo importante é comparar a saída de uma rede neural com o valor desejado. Ambos esse valores são vetores da mesma dimensão  $c$ . No caso em que a função de perda é o erro quadrático, o vetor  $\mathbf{u}$  na (eq. 2.53g) assume o papel da diferença entre os dois vetores desejados  $\mathbf{y}$  e explicado pelo modelo da rede  $\hat{\mathbf{y}}$ , ou seja,  $\mathbf{u} = \mathbf{y} - \hat{\mathbf{y}}$ . O produto  $\mathbf{u}^T \mathbf{u}$ , ou equivalentemente a norma quadrática  $\|\mathbf{u}(\mathbf{x})\|^2$  assume o papel da função de perda. Ainda por cima, a função de perda pode ser normalizada, dividindo pela dimensão do vetor, ou seja, se  $\mathbf{u} \in \mathbb{R}^c$ , temos como perda

$$\frac{1}{c} \|\mathbf{u}(\mathbf{x})\|^2 = \frac{1}{c} \|\mathbf{y} - \hat{\mathbf{y}}\|^2.$$

Temos que calcular a derivada dessa perda em relação aos pesos da rede. A variável  $\mathbf{x}$  assume o papel destes pesos. Os pesos em geral estão afastados das saídas, de maneira que a regra da cadeia é necessária para analisar como os pesos influenciam a saída para modificá-los, assim aprendendo.

Vamos reformular a função de perda em forma matricial para chegar de uma maneira mais elegante e compacta no mesmo resultado ótimo  $\boldsymbol{\theta}^* = (w^*, b^*)$  obtido pela fórmula conhecida (eq. 2.7), ou zerando o gradiente da função de perda  $\nabla L(w, b) = \mathbf{0}^\top$  da (eq. 2.48). Lembrando do problema de regressão linear dos dados na tabela 2.1 e do modelo linear (eq. 2.4), temos  $n = 7$  valores desejados  $y_i$  e  $n = 7$  valores explicados pelo modelo linear  $\hat{y}_i = wx_i + b$ ,  $i = 1, \dots, n$ . Podemos

agrupar estes  $n$  valores em dois vetores, o desejado  $\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$  e o explicado  $\hat{\mathbf{y}} = \begin{bmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_n \end{bmatrix}$ , no caso

concreto  $\mathbf{y} = \begin{bmatrix} 0.7 \\ 1.0 \\ 0.8 \\ 0.9 \\ 1.4 \\ 2.1 \\ 2.4 \end{bmatrix}$  e  $\hat{\mathbf{y}} = \begin{bmatrix} w \cdot 0.4 + b \\ w \cdot 0.9 + b \\ w \cdot 1.5 + b \\ w \cdot 2.3 + b \\ w \cdot 2.9 + b \\ w \cdot 3.1 + b \\ w \cdot 3.7 + b \end{bmatrix}$ . Vamos utilizar a forma aumentada (eq. 2.43) dos vetores

de entrada e vetores de pesos. No momento somente temos um peso escalar e uma entrada escalar, mas na representação aumentada termos vetores bidimensionais. Então o  $i$ -ésimo valor explicado  $\hat{y}_i = wx_i + b$  é substituído pela forma aumentada equivalente  $\hat{y}_i = w_0x_0 + w_1x_i = \tilde{\mathbf{w}} \cdot \tilde{\mathbf{x}}_i$ , com as substituições conhecidas  $w_0 = b$  e  $x_0 = 1$ . Assim temos para o vetor dos  $n$  valores explicados

$$\hat{\mathbf{y}} = \begin{bmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_n \end{bmatrix} = \begin{bmatrix} w_0 \cdot 1 + w_1 x_1 \\ \vdots \\ w_0 \cdot 1 + w_1 x_n \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{w}} \cdot \tilde{\mathbf{x}}_1 \\ \vdots \\ \tilde{\mathbf{w}} \cdot \tilde{\mathbf{x}}_n \end{bmatrix} = \tilde{\mathbf{X}} \tilde{\mathbf{w}}, \quad (2.54)$$

onde a matriz  $\tilde{\mathbf{X}}$  de dimensão  $n \times 2$  tem em cada linha o vetor aumentado  $\tilde{\mathbf{x}}_i$  transposto, ou seja

$$\tilde{\mathbf{X}} \stackrel{\text{def}}{=} \begin{bmatrix} \tilde{\mathbf{x}}_1^\top \\ \vdots \\ \tilde{\mathbf{x}}_n^\top \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}.$$

No caso concreto do exemplo da tabela 2.1 temos

$$\tilde{\mathbf{X}} = \begin{bmatrix} 1 & 0.4 \\ 1 & 0.9 \\ 1 & 1.5 \\ 1 & 2.3 \\ 1 & 2.9 \\ 1 & 3.1 \\ 1 & 3.7 \end{bmatrix}. \quad (2.55)$$

Revemos a função de perda  $L(w, b) = L(\tilde{\mathbf{w}}) = \frac{1}{n} \sum_i (y_i - \hat{y}_i)^2$ , definida em (eq. 2.37) e o produto escalar de um vetor  $\mathbf{x}$  com ele próprio  $\mathbf{x} \cdot \mathbf{x} = \|\mathbf{x}\|^2 = \sum_i x_i^2$ , definido na (eq. 1.4) na pág. 12. Usando a (eq. 2.54) podemos redefinir a função de perda como

$$L(w, b) = L(\tilde{\mathbf{w}}) = \frac{1}{n} \sum_i (y_i - \hat{y}_i)^2 = \frac{1}{n} (\mathbf{y} - \hat{\mathbf{y}}) \cdot (\mathbf{y} - \hat{\mathbf{y}}) = \frac{1}{n} \|\mathbf{y} - \hat{\mathbf{y}}\|^2 = \frac{1}{n} \|\mathbf{y} - \tilde{\mathbf{X}} \tilde{\mathbf{w}}\|^2. \quad (2.56)$$

Observamos ainda que a função de perda é a média aritmética da distância Euclidiana quadrática entre os valores desejados  $\mathbf{y}$  e explicados  $\hat{\mathbf{y}}$ , definida nas (eq. 1.9) na pág. 13 e (eq. 1.11) na pág. 14, ou seja

$$L(w, b) = L(\tilde{\mathbf{w}}) = \frac{1}{n} D_E^2(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{n} \|\mathbf{y} - \hat{\mathbf{y}}\|^2 = \frac{1}{n} \|\mathbf{y} - \tilde{\mathbf{X}} \tilde{\mathbf{w}}\|^2.$$

Para obter a solução ótima  $\boldsymbol{\theta}^* = (w^*, b^*) = \tilde{\mathbf{w}}^*$ , caminhamos da mesma maneira como antes, ze-

rando o gradiente da função de perda, ou seja, a raiz da equação  $\nabla L = \mathbf{0}^\top$  é a solução. Assim

$$\begin{aligned}\nabla L &= \frac{\partial L}{\partial \tilde{\mathbf{w}}} = \frac{\partial}{\partial \tilde{\mathbf{w}}} \left( \frac{1}{n} \|\mathbf{y} - \hat{\mathbf{y}}\|^2 \right) = \frac{2}{n} (\mathbf{y} - \hat{\mathbf{y}})^\top \frac{\partial}{\partial \tilde{\mathbf{w}}} (\mathbf{y} - \hat{\mathbf{y}}) = -\frac{2}{n} (\mathbf{y} - \hat{\mathbf{y}})^\top \frac{\partial \hat{\mathbf{y}}}{\partial \tilde{\mathbf{w}}} \\ &= -\frac{2}{n} (\mathbf{y} - \hat{\mathbf{y}})^\top \frac{\partial \tilde{X} \tilde{\mathbf{w}}}{\partial \tilde{\mathbf{w}}} = -\frac{2}{n} (\mathbf{y} - \hat{\mathbf{y}})^\top \tilde{X}.\end{aligned}\quad (2.57)$$

Finalmente, zerando o gradiente da função de perda, temos

$$\nabla L = \frac{\partial L}{\partial \tilde{\mathbf{w}}} = -\frac{2}{n} (\mathbf{y} - \hat{\mathbf{y}})^\top \tilde{X} = -\frac{2}{n} (\mathbf{y} - \tilde{X} \tilde{\mathbf{w}})^\top \tilde{X} = \mathbf{0}^\top. \quad (2.58)$$

Algumas regras relacionadas ao cálculo de matrizes necessárias são

**Definição 2.1.12 — Regras de Cálculo de Matrizes: Transposição, Lei distributiva da multiplicação, Inversa.** Sejam  $A$  e  $C$  matrizes de dimensão  $k \times \ell$ ,  $B$  uma matriz de dimensão  $\ell \times m$  e  $a \in \mathbb{R}$  um escalar. Então

$$a^\top = a \quad \text{Transposta de matriz de dimensão } 1 \times 1 \quad (2.59a)$$

$$(A^\top)^\top = A \quad \text{Transposta de transposta é original} \quad (2.59b)$$

$$(AB)^\top = B^\top A^\top \quad \text{Transposta de produto é produto de transpostas em ordem inversa} \quad (2.59c)$$

$$A^\top = C^\top \Leftrightarrow A = C \quad \text{Transposta preserve equivalência da equação} \quad (2.59d)$$

$$(A^{-1})^{-1} = A \quad \text{Inversa de inversa é original} \quad (2.59e)$$

$$(AB)^{-1} = B^{-1}A^{-1} \quad \text{Inversa de produto é produto de inversa em ordem inversa} \quad (2.59f)$$

$$A^{-1} = C^{-1} \Leftrightarrow A = C \quad \text{Inversa preserve equivalência da equação} \quad (2.59g)$$

$$(A^{-1})^\top = (A^\top)^{-1} \quad \text{Transposta de inversa é inversa de transposta} \quad (2.59h)$$

$$(A + C)B = (AB) + (CA) \quad \text{Lei distributiva} \quad (2.59i)$$

As regras contendo a inversa de uma matriz supõem a existência da mesma.

Para simplificar a nomenclatura, vamos omitir a tilde da matriz de dados aumentada  $\tilde{X}$ , e do vetor aumentado dos parâmetros  $\tilde{\mathbf{w}}$ , chamando as de  $X$  e  $\mathbf{w}$ , mas lembre se que se trata da forma

aumentada (eq. 2.43). Então<sup>5</sup> temos

$$\begin{aligned}
 \nabla L = \frac{\partial L}{\partial \mathbf{w}} &= -\frac{2}{n} (\mathbf{y} - \mathbf{X}\mathbf{w})^\top \mathbf{X} = \mathbf{0}^\top \\
 \iff (\mathbf{y} - \mathbf{X}\mathbf{w})^\top \mathbf{X} &= \mathbf{0}^\top \\
 \iff \left[ (\mathbf{y} - \mathbf{X}\mathbf{w})^\top \mathbf{X} \right]^\top &= (\mathbf{0}^\top)^\top \\
 \stackrel{2.59b}{\iff} \left[ (\mathbf{y} - \mathbf{X}\mathbf{w})^\top \mathbf{X} \right]^\top &= \mathbf{0} \\
 \stackrel{2.59c}{\iff} \mathbf{X}^\top \left[ (\mathbf{y} - \mathbf{X}\mathbf{w})^\top \right]^\top &= \mathbf{0} \\
 \stackrel{2.59b}{\iff} \mathbf{X}^\top (\mathbf{y} - \mathbf{X}\mathbf{w}) &= \mathbf{0} \\
 \iff \mathbf{X}^\top \mathbf{y} &= \mathbf{X}^\top \mathbf{X}\mathbf{w}. \tag{2.60}
 \end{aligned}$$

A matriz de dados  $\mathbf{X}$  no nosso exemplo tem dimensão  $7 \times 2$ , então a sua transposta  $\mathbf{X}^\top$  tem dimensão  $2 \times 7$ . O produto  $\mathbf{X}^\top \mathbf{X}$  tem dimensão  $2 \times 2$ . Vamos supor que essa matriz quadrada não seja singular, que o seu determinante seja não nulo  $|\mathbf{X}^\top \mathbf{X}| \neq 0$ . Assim é possível calcular a sua inversa  $(\mathbf{X}^\top \mathbf{X})^{-1}$ . Multiplicando os dois lados da (eq. 2.60) por essa inversa, temos, lembrando que se trata das formas aumentadas

$$\begin{aligned}
 (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} &= (\mathbf{X}^\top \mathbf{X})^{-1} (\mathbf{X}^\top \mathbf{X}) \mathbf{w} \\
 \iff (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} &= \mathbf{w} \\
 \stackrel{\text{def}}{\iff} \mathbf{w} &= \mathbf{X}^\dagger \mathbf{y}. \tag{2.61}
 \end{aligned}$$

**Definição 2.1.13 — Pseudoinversa, Inversa de Moore-Penrose.** Dada a matriz retangular  $\mathbf{X}$  de dimensão  $n \times m$ , a pseudoinversa em relação às colunas linearmente independentes, a pseudoinversa da esquerda,  $\mathbf{X}^\dagger$  de dimensão  $m \times n$  é

$$\mathbf{X}^\dagger \stackrel{\text{def}}{=} (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top. \tag{2.62}$$

A pseudoinversa é uma generalização da inversa de uma matriz quadrada. Ela é usada para um ajuste ótimo no sentido do Erro Quadrático Médio (EQM). Essa regra para obter os parâmetros livres de um modelo linear, envolvendo os dados de entrada  $\mathbf{X}$  e saída em uma simples equação pode ser considerado o primeiro método de aprendizagem supervisionada para resolver um problema de regressão linear. Não importa, se tem mais que uma entrada e/ou mais que uma saída, a regra fornece, obviamente ajustando as dimensões para mais entradas e saídas, em uma operação simples de álgebra linear uma solução determinística, envolvendo somente multiplicações e inversão de matrizes.

■ **Exemplo 2.8** Para o exemplo ilustrativo de regressão linear simples da tabela 2.1 a matriz aumentada na (eq. 2.55) tem dimensão  $7 \times 2$ . Portanto, o produto  $\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}} = \begin{bmatrix} 7.0 & 14.8 \\ 14.8 & 40.22 \end{bmatrix}$  tem dimensão  $2 \times 2$  e a pseudoinversa tem dimensão  $2 \times 7$

<sup>5</sup>Uma referência de uma equação acima do símbolo de igualdade “=” ou equivalência “ $\iff$ ” aponta para a regra usada durante o raciocínio.

$\tilde{X}^\dagger = \begin{bmatrix} 0.5488 & 0.4304 & 0.28832 & 0.09888 & -0.0432 & -0.09056 & -0.23264 \\ -0.192 & -0.136 & -0.0688 & 0.0208 & 0.088 & 0.1104 & 0.1776 \end{bmatrix}$ . Finalmente, a solução da regressão é o vetor de parâmetros livres  $\tilde{\mathbf{w}} = \begin{bmatrix} b \\ w \end{bmatrix} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \tilde{X}^\dagger \mathbf{y} = \begin{bmatrix} 0.325 \\ 0.475 \end{bmatrix}$ , idêntico ao resultado obtido pelos caminhos anteriores. ■

## 2.2 Regressão Linear

Temos as ferramentas necessárias para definir um algoritmo que aprende um modelo linear para qualquer dimensão do vetor de entrada  $\mathbf{x}$  e qualquer dimensão do vetor de saída. Na apresentação inicial do modelo linear simples  $y = wx + b$ , a entrada era um vetor aumentado de dimensão dois com  $x_0 \stackrel{\text{def}}{=} 1$ ,  $x_1 \stackrel{\text{def}}{=} x$ , e a saída  $y_1 \stackrel{\text{def}}{=} y$  era um escalar, e tínhamos um sistema de  $n$  equações  $y_i = wx_i + b$ ,  $i = 1, \dots, n$ , representando o conjunto de treinamento. Podemos generalizar essas estruturas de entrada e saída para serem multidimensionais. Considerando o modelo simples, a forma geral da  $i$ -ésima equação seria então  $y_{i,1} = w_0x_{i,0} + w_1x_{i,1}$ , com  $w_0 \stackrel{\text{def}}{=} b$ ,  $w_1 \stackrel{\text{def}}{=} w$ .

### 2.2.1 Máquina Linear

**Definição 2.2.1 — Modelo Linear Múltiplo – Máquina Linear.** Seja  $\mathbf{x} = [1 \ x_1 \ \cdots \ x_j \ \cdots \ x_m]^\top$  um vetor de entrada aumentado de dimensão  $(m+1)$ , seja  $\mathbf{y} = [y_1 \ \cdots \ y_k \ \cdots \ y_c]^\top$  um vetor de saída de dimensão  $c$ , e seja  $W$  uma matriz de pesos de dimensão  $(m+1) \times c$  que tem como  $k$ -ésima coluna o  $k$ -ésimo vetor de pesos  $\mathbf{w}_k = [w_{0,k} \ w_{1,k} \ \cdots \ w_{j,k} \ \cdots \ w_{m,k}]^\top$  de dimensão  $(m+1)$ , ou seja,

$$W = [\mathbf{w}_1 \ \cdots \ \mathbf{w}_k \ \cdots \ \mathbf{w}_c] = \begin{bmatrix} w_{0,1} & \cdots & w_{0,k} & \cdots & w_{0,c} \\ w_{1,1} & \cdots & w_{1,k} & \cdots & w_{1,c} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ w_{j,1} & \cdots & w_{j,k} & \cdots & w_{j,c} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ w_{m,1} & \cdots & w_{m,k} & \cdots & w_{m,c} \end{bmatrix}. \quad (2.63)$$

Então o modelo linear múltiplo de  $m$  entradas e  $c$  saídas com a  $k$ -ésima saída

$$y_k = \mathbf{w}_k \cdot \mathbf{x} = \mathbf{w}_k^\top \mathbf{x} = w_{0,k}x_0 + w_{1,k}x_1 + \cdots + w_{j,k}x_j + \cdots + w_{m,k}x_m, \quad (2.64)$$

em forma matricial é definido como

$$\mathbf{y} = W^\top \mathbf{x}. \quad (2.65)$$

É a multiplicação de uma matriz  $c \times (m+1)$  por uma matriz  $(m+1) \times 1$ , resultando em uma matriz  $c \times 1$ , equivalente a um vetor de coluna de  $c$  componentes. Lembre que temos  $m$  entradas e adicionalmente o viés  $b = w_0$ , resultando em um vetor de dimensão  $(m+1)$  para a entrada e o  $k$ -ésimo vetor de pesos  $\mathbf{w}_k$ . Podemos identificar na (eq. 2.65) uma especialização da nossa definição (eq. 2.1) de um modelo de aprendizado de máquina, aqui reunindo os parâmetros livres na matriz  $W$ . Lembrando da proposição 1.1.3 na pág. 13, podemos afirmar que o  $k$ -ésimo vetor de pesos  $\mathbf{w}_k$  então mede uma afinidade ou semelhança com o vetor de entrada  $\mathbf{x}$ . Consequentemente o vetor de saída  $\mathbf{y}$  em cada uma das suas  $c$  componentes incorpora uma semelhança específica.

### 2.2.2 Máquina Linear como Rede Neural Artificial

Graficamente, acabamos de definir uma rede neural com  $(m+1)$  entradas e  $c$  saídas na figura 2.4. Em geral, este modelo pode ser usado para fazer regressão, em uma forma mais especializada também para fazer classificação. Se usarmos a Máquina Linear para classificar, cada saída então retorna uma afinidade quantitativa da  $k$ -ésima saída com a  $k$ -ésima classe. No exemplo da figura 2.4 para uma flor  $\mathbf{x}$  desconhecida, a primeira saída  $y_1$  então mede a semelhança com Iris setosa, por exemplo. A matriz de pesos  $W$  da (eq. 2.63) então concentra todo o conhecimento como distinguir uma classe de flor da outra em cada uma das suas colunas.

**Convenção:** Neste modelo da Máquina Linear existem pesos que têm dois índices, nomeadamente  $w_{j,k}$ . Vamos adotar a regra que o primeiro índice, aqui  $j$ , seja associado à origem, neste caso a  $j$ -ésima entrada  $x_j$ , e o segundo índice, aqui  $k$ , seja associado à saída, neste caso a  $k$ -ésima saída  $y_k$ . Então de agora em diante, em modelos que podem ser ilustrados em grafos, o primeiro índice dos pesos significa a origem, e o segundo índice significa o destino do processamento, ou seja,

$$w_{\text{de},\text{para}} = w_{\text{origem},\text{destino}}. \quad (2.66)$$

### 2.2.3 Treinamento Determinístico de Máquina Linear

Temos um modelo e temos um conjunto de treinamento. Então a tarefa é aprender os parâmetros que neste modelo vivem na matriz de pesos. Cada um dos  $n$  padrões do conjunto de treinamento é composto por um vetor aumentado de entrada

$$\mathbf{x}_i = [x_{i,0} \quad x_{i,1} \quad \cdots \quad x_{i,j} \quad \cdots \quad x_{i,m}]^\top$$

de dimensão  $(m+1)$  e um vetor de saída

$$\mathbf{y}_i = [y_{i,1} \quad \cdots \quad y_{i,k} \quad \cdots \quad y_{i,c}]^\top$$

de dimensão  $c$ . Podemos criar duas matrizes  $X$  e  $Y$  que em cada linha tem o  $i$ -ésimo vetor aumentado de entrada  $\mathbf{x}_i$ , respectivamente de saída desejada  $\mathbf{y}_i$ , ou seja

$$X = \begin{bmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_i^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix} = \begin{bmatrix} x_{1,0} & \cdots & x_{1,j} & \cdots & x_{1,m} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i,0} & \cdots & x_{i,j} & \cdots & x_{i,m} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{n,0} & \cdots & x_{n,j} & \cdots & x_{n,m} \end{bmatrix}, Y = \begin{bmatrix} \mathbf{y}_1^\top \\ \vdots \\ \mathbf{y}_i^\top \\ \vdots \\ \mathbf{y}_n^\top \end{bmatrix} = \begin{bmatrix} y_{1,1} & \cdots & y_{1,k} & \cdots & y_{1,c} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ y_{i,1} & \cdots & y_{i,k} & \cdots & y_{i,c} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ y_{n,1} & \cdots & y_{n,k} & \cdots & y_{n,c} \end{bmatrix}. \quad (2.67)$$

O mapeamento entrada-saída de cada um dos  $n$  pares é feita pela máquina linear (eq. 2.65) como

$$\mathbf{y}_i = W^\top \mathbf{x}_i, \quad i = 1, \dots, n. \quad (2.68)$$

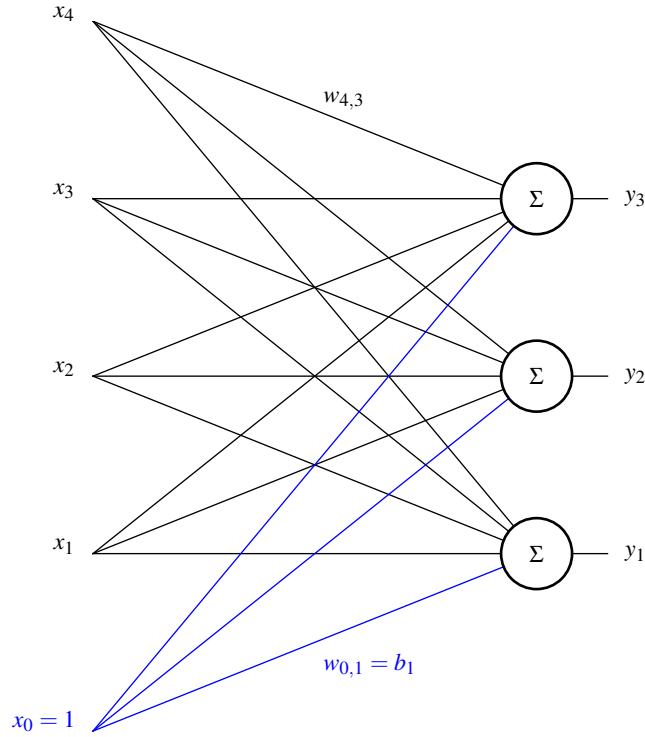


Figura 2.4: Máquina Linear. O exemplo tem  $(m+1) = (4+1) = 5$  entradas  $x_0, \dots, x_m$  e  $c = 3$  saídas  $y_1, \dots, y_c$ . A quantidade de pesos  $w_{j,k}$  é  $(m+1) \times c = (4+1) \times 3 = 15$ , armazenados em uma matriz  $W$  de dimensão  $5 \times 3$ . Os índices de entrada percorrem  $j = 0, \dots, m$ ,  $m = 4$ , e os índices de saída percorrem  $k = 1, \dots, c$ ,  $c = 3$

Dessa maneira temos

$$\begin{aligned}
 \mathbf{y}_i &= W^T \mathbf{x}_i, \quad i = 1, \dots, n \\
 \xrightleftharpoons[2.59d]{\quad} \mathbf{y}_i^T &= \left( W^T \mathbf{x}_i \right)^T, \quad i = 1, \dots, n \\
 \xrightleftharpoons[2.59c]{\quad} \mathbf{y}_i^T &= \mathbf{x}_i^T (W^T)^T, \quad i = 1, \dots, n \\
 \xrightleftharpoons[2.59b]{\quad} \mathbf{y}_i^T &= \mathbf{x}_i^T W, \quad i = 1, \dots, n \\
 \iff \begin{bmatrix} \mathbf{y}_1^T \\ \vdots \\ \mathbf{y}_n^T \end{bmatrix} &= \begin{bmatrix} \mathbf{x}_1^T W \\ \vdots \\ \mathbf{x}_n^T W \end{bmatrix} \\
 \iff \begin{bmatrix} \mathbf{y}_1^T \\ \vdots \\ \mathbf{y}_n^T \end{bmatrix} &= \begin{bmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_n^T \end{bmatrix} W \\
 \xrightleftharpoons[2.67]{\quad} Y &= XW. \tag{2.69}
 \end{aligned}$$

Isso significa que o mapeamento do conjunto de treinamento inteiro pode ser expresso em uma única equação linear. Também significa que, usando a pseudoinversa da matriz de dados, leva diretamente ao aprendizado dos pesos. Então, trocando os lados na (eq. 2.69), e, supondo que a matriz  $X^T X$  não seja singular, permitindo a sua inversão, temos

$$\begin{aligned} XW &= Y \\ \iff X^T X W &= X^T Y \\ \iff (X^T X)^{-1} (X^T X) W &= (X^T X)^{-1} X^T Y \\ \iff W &= X^{\dagger} Y \end{aligned} \tag{2.70}$$

Alcançamos o objetivo de ter uma forma fechada para aprender os pesos. Embora envolva variáveis multidimensionais, é uma fórmula extremamente simples. Os passos para aplicar a regressão linear múltipla para treinar a máquina linear de uma forma determinística são resumidos no algoritmo 1.

---

#### Algoritmo 1: Regressão Linear Múltipla para Treinamento da Máquina Linear

---

**Entrada:** Conjunto de treinamento de  $n$  pares de entrada-saída  $\mathcal{D} = \{\mathbf{x}_i, \mathbf{y}_i\}, i = 1, \dots, n$ .

Cada entrada tem  $m$  componentes em forma de vetor de coluna, ou seja  $\mathbf{x}_i \in \mathbb{R}^m$ ,  
cada saída tem  $c$  componentes em forma de vetor de coluna, ou seja  $\mathbf{y}_i \in \mathbb{R}^c$

**Resultado:** Matriz de parâmetros livres aprendidos (pesos)  $W$  de dimensão  $(m+1) \times c$

- 1 Monte a matriz de dados  $X$  de dimensão  $n \times m$ , colocando na  $i$ -ésima linha a  
 $i$ -ésima entrada em forma transposta  $\mathbf{x}_i^T$ ;
  - 2 Monte a matriz de saídas desejadas  $Y$  de dimensão  $n \times c$ , colocando na  $i$ -ésima linha a  
 $i$ -ésima saída em forma transposta  $\mathbf{y}_i^T$ ;
  - 3 Insere na matriz de dados  $X$ , antes da primeira coluna, uma coluna com o valor fixo 1 em  
cada linha, criando assim a matriz de dados aumentada  $\tilde{X} = [\mathbf{1} \ X]$  de dimensão  
 $n \times (m+1)$ ;
  - 4 Calcule a pseudoinversa  $\tilde{X}^{\dagger}$  de dimensão  $(m+1) \times n$  da matriz de dados aumentada  
conforme a (eq. 2.62);
  - 5 Multiplique a pseudoinversa  $\tilde{X}^{\dagger}$  de dimensão  $(m+1) \times n$  pela matriz de saídas desejadas  
 $Y$  de dimensão  $n \times c$ , resultando na matriz de pesos  $W$  de dimensão  $(m+1) \times c$ ,  
 $W = \tilde{X}^{\dagger} Y$
- 

■ **Exemplo 2.9** O exemplo ilustrativo para fazer regressão linear simples da tabela 2.1 e figura 2.1 tem uma variável de entrada  $x$  unidimensional, então  $m = 1$  no algoritmo 1, e uma variável de saída  $y$  unidimensional, então  $c = 1$  no algoritmo 1, definindo o modelo  $f(x; w, b) = wx + b$ , ou em forma aumentada  $f(\mathbf{x}; \mathbf{w}) = w_0 x_0 + w_1 x_1$ . Se aumentarmos a dimensão de entrada para  $(m+1) = 2$  e deixarmos a saída unidimensional, temos um problema de ajuste de pontos no plano  $xy$ , com a coordenada  $z$  como saída a um plano em um sistema de coordenadas cartesianas de três dimensões. Usando nomes de variáveis mais comuns, queremos fazer ajuste de uma função linear  $z = f(x, y)$ . Usando a nomenclatura do modelo linear, queremos fazer um ajuste  $f(\mathbf{x}; \mathbf{w}) = w_0 x_0 + w_1 x_1 + w_2 x_2$ . A tabela 2.2 e a figura 2.5 mostram um conjunto ilustrativo de dez pontos  $(x_{i,1}, x_{i,2})$  que formam a matriz de dados  $X$ , junto com o valor desejado  $y_i$ ,  $i = 1, \dots, 10$ . O resultado da regressão, aplicando o algoritmo 1 é um vetor de pesos, pois a saída  $y$  é unidimensional. O modelo ajustado é  $\mathbf{w} = [w_0 \ w_1 \ w_2]^T = [4.60 \ 2.19 \ 2.78]^T$ . ■

Número do padrão	$x_1$	$x_2$	$y$
1	1	1	8.00
2	2	3	16.66
3	3	1	14.64
4	4	4	23.93
5	2	3	18.99
6	4	4	24.76
7	6	1	19.12
8	7	7	39.56
9	6	3	28.36
10	10	10	53.69

Tabela 2.2: Conjunto de treinamento para ajuste de plano.

Ajuste de modelo linear  
 $y(x_1, x_2) = 4.60 + 2.19x_1 + 2.78x_2$

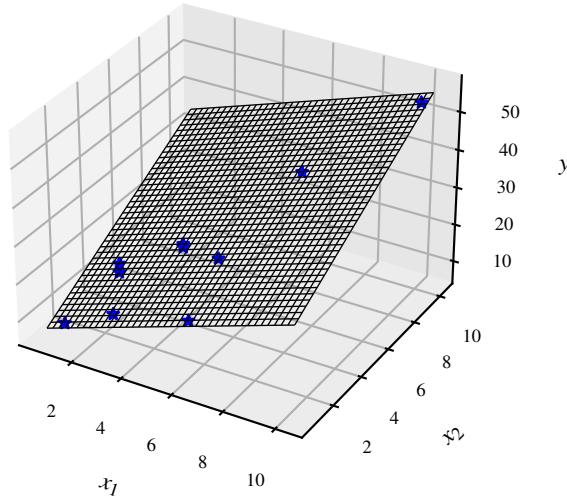


Figura 2.5: Plano ajustado, junto com pontos do conjunto de treinamento.

■ **Exemplo 2.10** Um exemplo de regressão linear múltipla com um vetor de entrada  $\mathbf{x}$  multidimensional e um vetor de saída  $\mathbf{y}$  multidimensional é o conjunto de dados<sup>6</sup> “Linnerud” [62, 107]. Os dados foram adquiridos em um ambiente de medicina de esporte. São 20 padrões com três variáveis independentes (entrada)  $x_1, x_2, x_3$  e três variáveis dependentes (saída)  $y_1, y_2, y_3$ , ou seja, as dimensões são  $m = 3$  e  $c = 3$ . As entradas são as variáveis de exercício físico “queixo”, “abdominais”, “saltos”, (*chin, sit-up, jumps*), veja a tabela 2.3, as saídas são variáveis fisiológicas “peso”, “cintura” e “batimento cardíaco”, (*weight, waist, pulse*), veja a tabela 2.3. Antes da regressão, aplica-se a estandardização (eq. 1.37) às matrizes de entradas  $X$  de dimensão  $20 \times (1+3)$  e de saída  $Y$  de dimensão  $20 \times 3$ . Após a regressão, a matriz de pesos  $W$  de dimensão  $(1+3) \times 3$  tem

<sup>6</sup>Em geral os conjuntos de dados, como “Iris” estão disponíveis nos ambientes de desenvolvimento. Usam-se aqui os conjuntos de dados da biblioteca de algoritmos de aprendizado de máquina da linguagem Python, o `scikit-learn`. O pacote `sklearn.datasets` contém, entre outros o “Iris” e “Linnerud”.

Queixo (chin)	X			Y		
	Abdominais (sit-up)	Saltos (jumps)		Peso (weight)	Cintura (waist)	Batimento cardíaco (pulse)
5	162	60		191	36	50
2	110	60		189	37	52
12	101	101		193	38	58
12	105	37		162	35	62
13	155	58		189	35	46
4	101	42		182	36	56
8	101	38		211	38	56
6	125	40		167	34	60
15	200	40		176	31	74
17	251	250		154	33	56
17	120	38		169	34	50
13	210	115		166	33	52
14	215	105		154	34	64
1	50	50		247	46	50
6	70	31		193	36	46
12	210	120		202	37	62
4	60	25		176	37	54
11	230	80		157	32	52
15	225	73		156	33	54
2	110	43		138	33	68

Tabela 2.3: Dados “Linnerud” [62, 107] para regressão múltipla. Três variáveis explicativas contínuas de entrada e três variáveis explicadas contínuas de saída.

o valor

$$W = \begin{bmatrix} 0 & 0 & 0 \\ -0.475 & -0.137 & 0.001 \\ -0.218 & -0.04 & 0.042 \\ 0.093 & 0.028 & -0.029 \end{bmatrix}$$

Devido à centralização das entradas  $\mathbf{x}$  e saídas  $\mathbf{y}$ , ou seja, devido às novas médias zeradas,  $\bar{\mathbf{x}} = \mathbf{0}$  e  $\bar{\mathbf{y}} = \mathbf{0}$ , os valores dos vieses  $w_{0,1}, w_{0,2}, w_{0,3}$  tiverem um valor numérico igual a zero, na ordem de  $10^{-16}$ . Isso não acontece em geral, se os dados não são centrados (sem prova). ■

## 2.3 Classificação Linear por Máquina Linear

Podemos usar o mecanismo da regressão linear para construir uma Máquina Linear que funciona como classificador. Isso é muito fácil, pois já temos uma maneira de codificar a qual classe pertence um padrão. Usamos a codificação *One-Hot* definida na (eq. 2.6). Com uma quantidade de  $n$  padrões e  $c$  classes, basta formar a matriz de saída  $Y$  de dimensão  $n \times c$  definida na (eq. 2.67).

■ **Exemplo 2.11** Quem sentiu falta do exemplo das flores pode ver agora a criação de um classificador usando a Máquina Linear. Já temos a matriz de dados  $X$ , veja (eq. 1.1) na pág. 11. Usando codificação *One-Hot* temos a informação sobre as três classes de Iris na matriz  $Y$  de dimensão

$150 \times 3$ , com os valores

$$Y = \begin{bmatrix} \mathbf{y}_1^\top \\ \vdots \\ \mathbf{y}_{50}^\top \\ \mathbf{y}_{51}^\top \\ \vdots \\ \mathbf{y}_{100}^\top \\ \mathbf{y}_{101}^\top \\ \vdots \\ \mathbf{y}_{150}^\top \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 1 \end{bmatrix}. \quad (2.71)$$

A matriz de pesos  $W = \tilde{X}^\dagger Y$ , aprendidos pelo algoritmo 1 é

$$W = \begin{bmatrix} w_{0,1} & w_{0,2} & w_{0,3} \\ w_{1,1} & w_{1,2} & w_{1,3} \\ w_{2,1} & w_{2,2} & w_{2,3} \\ w_{3,1} & w_{3,2} & w_{3,3} \\ w_{4,1} & w_{4,2} & w_{4,3} \end{bmatrix} = \begin{bmatrix} 0.118 & 1.577 & -0.695 \\ 0.066 & -0.020 & -0.046 \\ 0.243 & -0.446 & 0.203 \\ -0.225 & 0.221 & 0.004 \\ -0.057 & -0.494 & 0.552 \end{bmatrix}. \quad (2.72)$$

■

Lembre que a matriz de dados original  $X$  das flores de dimensão  $150 \times 4$  recebe um nova primeira coluna de valores igual a um para virar a matriz de dados aumentada  $\tilde{X}$  de dimensão  $150 \times 5$ . Podemos afirmar que com 15 números que correspondem aos pesos da matriz, e um modelo no contexto da álgebra linear, criamos um classificador multi-classe, neste caso de três classes. Ele recebe um vetor  $\mathbf{x}$  de quatro características e retorna um vetor de três números  $\mathbf{y}$  que representam numericamente a qual classe o padrão pertence. Podemos atribuir uma semântica à esta matriz dos pesos. Cada uma das três colunas é um detector de semelhança de cada uma das três espécies de iris. Lembre-se que o produto escalar de dois vetores mede a sua afinidade, veja definição 1.1.1 na pág. 11. Então, por exemplo, calcular o produto escalar  $\mathbf{w}_2 \cdot \mathbf{x}$  da segunda coluna  $\mathbf{w}_2$  da matriz (eq. 2.72) com uma flor desconhecida  $\mathbf{x}$  relata quanto forte essa flor se relaciona à segunda classe. A figura 2.6 mostra a submissão da primeira das 150 flores que é da classe setosa à Máquina Linear, porém em um espaço de características de dimensão reduzida. Usamos somente a terceira e quarta característica. Todas as 150 flores são usadas como dados de treinamento, resultando na matriz que incorpora todo o conhecimento

$$W = [\mathbf{w}_1 \quad \mathbf{w}_2 \quad \mathbf{w}_3] = \begin{bmatrix} w_{0,1} & w_{0,2} & w_{0,3} \\ w_{1,1} & w_{1,2} & w_{1,3} \\ w_{2,1} & w_{2,2} & w_{2,3} \end{bmatrix} = \begin{bmatrix} 1.266 & -0.106 & -0.160 \\ -0.251 & 0.324 & -0.073 \\ 0.010 & -0.650 & 0.640 \end{bmatrix}.$$

Submetendo a primeira setosa  $\mathbf{x} = [1 \quad 1.4 \quad 0.2]^\top$  ao modelo, resulta nos produtos escalares  $\mathbf{w}_1 \cdot \mathbf{x} = 0.92$ ,  $\mathbf{w}_2 \cdot \mathbf{x} = 0.22$ ,  $\mathbf{w}_3 \cdot \mathbf{x} = -0.13$ , evidenciando que o vetor  $\mathbf{w}_1$  responsável por detectar uma setosa resultou na maior semelhança. Lembre-se que  $x_0 = 1$  sempre.

### 2.3.1 Score, Funções Discriminativas, Regiões e Fronteiras de Classes

Uma questão em aberto é como a saída  $\mathbf{y}$  da regressão linear será transformada em uma decisão, qual é a classe do padrão. As saídas  $y_1$ ,  $y_2$  e  $y_3$  são contínuas, mas a classe é uma variável aleatória discreta que tem somente os três valores simbólicos “Iris setosa”, “Iris versicolor” e “Iris virginica”. Temos que definir uma função que retorna o rótulo da classe, dado um vetor de

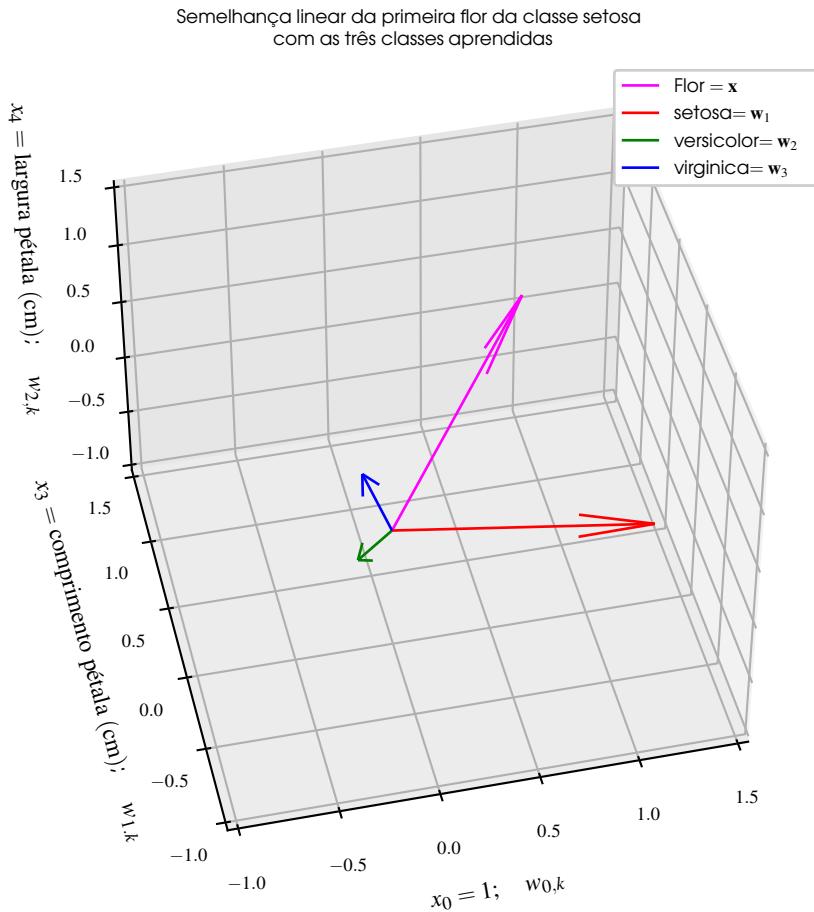


Figura 2.6: Plano ajustado, junto com pontos do conjunto de treinamento.

características. Essa função é uma *função discriminativa*. Além disso, falamos nada ainda sobre a qualidade do classificador baseado na Máquina Linear. Se este classificador mostra um bom desempenho ou não, será um assunto posterior. Antes de ilustrar como a Máquina Linear divide o espaço de características em contradomínios distintos de cada classe, precisamos de mais algumas ferramentas muito importantes.

Temos três classes de Iris. Vamos chamar Iris setosa de  $\mathcal{C}_1$ , Iris versicolor de  $\mathcal{C}_2$  e Iris virginica  $\mathcal{C}_3$ , usando uma letra caligráfica “ $\mathcal{C}$ ” que destaca o fato que se trata de uma variável simbólica, e não numérica. “Simbólico” significa que não existe uma ordem geométrica entre os valores desta variável. Quando um padrão  $\mathbf{x}_p$  pertence à classe  $\mathcal{C}_q$ , podemos escrever

$$\mathbf{x}_p \in \mathcal{C}_q, \quad (2.73)$$

em que o índice  $p$  se refere a alguma ordem dos padrões, por exemplo  $\mathbf{x}_{150}$  é a última flor, e o índice  $q$  se refere a uma classe existente, por exemplo, iris setosa seria  $\mathcal{C}_1$ . Na ocasião de classificação, um padrão qualquer  $\mathbf{x}$  não necessariamente tem algum índice, ou seja, o resultado da classificação pode ser  $\mathbf{x} \in \mathcal{C}_q$ , equivalente ao resultado da função discriminativa que será definida a seguir.

Precisamos de um mapeamento do domínio de origem do sistema de coordenadas cartesiano de quatro dimensões onde residem os padrões de quatro características: comprimento da sépala ( $x_1$ ), largura da sépala ( $x_2$ ), comprimento da pétala ( $x_3$ ) e largura da pétala ( $x_4$ ). O contradomínio é o domínio simbólico das classes. Porém, antes de tomar a decisão a qual classe pertence uma flor, temos que ter um valor numérico que nos diz quão perto um padrão fica de uma classe. Vamos

chamar essa função de avaliação a função de *score*. Ela mapeia do espaço das características para o espaço dos números reais. Isso permite associar um padrão numericamente a uma classe. Portanto, existe um *score* para cada uma das classes.

**Definição 2.3.1 — Função de Score.** Seja  $\mathcal{X} \stackrel{\text{def}}{=} \mathbb{R}^m$  o domínio dos padrões de dimensão  $m$ , e seja  $\mathcal{Y} \stackrel{\text{def}}{=} \mathbb{R}^c$  o contradomínio dos vetores de números reais de dimensão  $c$ , onde  $c$  é a quantidade de classes. Seja  $\{\mathcal{C}_1, \dots, \mathcal{C}_k, \dots, \mathcal{C}_c\}$  o conjunto de  $c$  classes. Então a função

$$\begin{aligned} y : \mathcal{X} &\rightarrow \mathcal{Y} \\ \mathbf{x} \mapsto \mathbf{y}(\mathbf{x}) &= [y_1(\mathbf{x}) \quad \cdots \quad y_k(\mathbf{x}) \quad \cdots \quad y_c(\mathbf{x})]^T \end{aligned} \quad (2.74)$$

é a função de *score*, ou abreviado, *score* das classes. O vetor de *scores*  $\mathbf{y}$  agrupa todos os  $c$  *scores* das classes  $\mathcal{C}_k$ ,  $k = 1, \dots, c$ .

■ **Exemplo 2.12** Se a máquina linear mapeia uma flor  $\mathbf{x}$  de quatro características pela (eq. 2.68), ou seja  $\mathbf{y} = W^T \mathbf{x}$ , temos como domínio  $\mathcal{X}$  de dimensão  $m = 4$  qualquer flor iris descrita por esses quatro características. Temos  $c = 3$  classes, uma para cada espécie de iris, e o contradomínio  $\mathcal{Y}$  são os vetores  $\mathbf{y}$  de dimensão  $c = 3$ . Como casos concretos considere a coluna dos scores explicados da tabela 2.4. ■

Idealmente esses *scores*  $y_k(\mathbf{x})$  seriam probabilidades com valores entre 0% e 100% para cada classe. Então uma maneira de classificar seria simplesmente escolher a classe com a maior probabilidade. Na prática é difícil obter verdadeiros valores de probabilidades, então temos que nos basear em *scores* relativos, ou seja ver como cada classe é avaliada. Naturalmente a classe escolhida é aquela que tem o maior *score* entre todas as classes possíveis. Voltaremos a esse assunto no contexto dos classificadores paramétricos, baseados na Regra de Bayes.

Existe uma maneira de transformar os *scores* das classes entre valores entre zero e um. Dessa maneira é possível *emular* probabilidades. Esses valores no intervalo  $[0, 1]$  somente em casos excepcionais coincidem com as verdadeiras probabilidades. Mesmo assim, essa normalização é útil, por exemplo, quando alguma etapa de processamento posterior assume esse intervalo. O mapeamento que consegue transformar qualquer conjunto de *scores* dos classificadores em valores  $[0, 1]$  é o *softmax*.

**Definição 2.3.2 — Função Softmax.** Seja  $\mathbf{y} = [y_1 \quad \cdots \quad y_k \quad \cdots \quad y_c]^T$  um vetor de  $c$  *scores*. Então a função

$$\begin{aligned} s : \mathbb{R}^c &\rightarrow [0, 1]^c \\ \mathbf{y} \mapsto s(\mathbf{y}) &= \frac{1}{\sum_{k=1}^c e^{y_k}} [e^{y_1} \quad \cdots \quad e^{y_k} \quad \cdots \quad e^{y_c}]^T \end{aligned} \quad (2.75)$$

é a função de softmax do vetor  $\mathbf{y}$ . A função transforma um vetor de  $c$  valores em um outro vetor de  $c$  valores, porém os valores do novo vetor não podem assumir qualquer valor, mas ficam forçados de ficar dentro de um intervalo previamente definido. A divisão pelo denominador  $\sum_{k=1}^c e^{y_k}$  age como um normalizador que garante que cada componente  $\frac{e^{y_k}}{\sum_{k=1}^c e^{y_k}}$  fique restringida ao intervalo  $[0, 1]$ .

■ **Exemplo 2.13** Até o presente ponto temos exatamente um modelo que nos permite definir um classificador. Então na Máquina Linear, o *score*  $y_k(\mathbf{x})$  da  $k$ -ésima classe  $\mathcal{C}_k$  é exatamente a  $k$ -ésima saída (eq. 2.64) e  $\mathbf{y}(\mathbf{x})$  representa o vetor de todos os *scores* do mapeamento do modelo linear  $\mathbf{y}(\mathbf{x}) = W^T \mathbf{x}$ , definido na (eq. 2.65). A Máquina Linear com o conjunto de dados Iris já foi treinada e o resultado é a matriz de pesos  $W$  na (eq. 2.72). Na figura 2.4 já tivemos exatamente a arquitetura desejada. O vetor de  $m = 4$  características, mais o viés é a entrada  $\mathbf{x} \in \mathbb{R}^{4+1}$ , e o vetor

	$\mathbf{x}_1$	$\mathbf{x}_{51}$	$\mathbf{x}_{101}$
Valor	$[5.1 \quad 3.5 \quad 1.4 \quad 0.2]^T$	$[7.0 \quad 3.2 \quad 4.7 \quad 1.4]^T$	$[6.3 \quad 3.3 \quad 6.0 \quad 2.5]^T$
Scores desejados $\mathbf{y}$	$[1 \quad 0 \quad 0]^T$	$[0 \quad 1 \quad 0]^T$	$[0 \quad 0 \quad 1]^T$
Scores explicados $\hat{\mathbf{y}}$	$[0.979 \quad 0.125 \quad -0.104]^T$	$[0.221 \quad 0.355 \quad 0.424]^T$	$[-0.156 \quad 0.068 \quad 1.088]^T$
Softmax de Scores explicados $s(\hat{\mathbf{y}})$	$[0.567 \quad 0.241 \quad 0.192]^T$	$[0.297 \quad 0.339 \quad 0.364]^T$	$[0.175 \quad 0.219 \quad 0.607]^T$

Tabela 2.4: Três padrões representativos do conjunto de treinamento, submetidos à Máquina Linear treinada, analisando os scores desejados e calculados.

de saída  $\mathbf{y} \in \mathbb{R}^3$  representa os *scores* das  $c = 3$  classes. Podemos chamar o vetor de saída desejada como vetor de *scores* desejadas. Realimentando alguma flor conhecida à Máquina Linear calcula então os *scores* explicados. Como exemplo vamos classificar a primeira flor de cada classe, os padrões  $\mathbf{x}_1$ ,  $\mathbf{x}_{51}$  e  $\mathbf{x}_{101}$  do conjunto de treinamento das 150 padrões. A tabela 2.4 mostra para cada um destes três representantes o padrão de entrada  $\mathbf{x}$ , o vetor  $\mathbf{y}$  de *scores* desejados, o vetor  $\hat{\mathbf{y}}$  de *scores* explicados pela Máquina Linear, e os *scores* explicados pós-processados  $s(\hat{\mathbf{y}})$  pela função softmax da (eq. 2.75). Podemos observar que o padrão da classe setosa  $\mathbf{x}_1$ , e o padrão da classe virginica  $\mathbf{x}_{101}$ , quando submetidos à Máquina Linear como entrada mostram nos seus respectivos *scores* valores altos,  $y_1(\mathbf{x}_1) = 0.979$  e  $y_3(\mathbf{x}_{101}) = 1.088$ . Após a passagem pelo softmax essa ordem se mantém. O representante de versicolor  $\mathbf{x}_{51}$  porém tem três *scores* relativamente parecidos. Isso indica que a Máquina Linear tem dificuldades de distinguir essa flor e atribuí-la a sua classe certa. Realmente, o score da terceira classe é maior,  $y_3(\mathbf{x}_{51}) = 0.424 > y_2(\mathbf{x}_{51}) = 0.355$ . Este classificador, em forma da Máquina Linear, que determina a classe pelo maior *score* cometerá um erro, mesmo com um padrão que faz parte do conjunto de treinamento. ■

Com a ajuda do *score*, podemos finalmente definir a função que mapeia um padrão para uma classe, via a estação intermediária do *score*. Como essa função distingue, ou discrimina, cada padrão e o associa a uma classe, será chamada de *função discriminativa*.

**Definição 2.3.3 — Função Discriminativa.** Seja  $\mathcal{X} \stackrel{\text{def}}{=} \mathbb{R}^m$  o domínio dos padrões de dimensão  $m$ , e seja  $\mathcal{Y} \stackrel{\text{def}}{=} \mathbb{R}^c$  o domínio dos *scores*. Seja  $\mathcal{C}$  o contradomínio das classes, o conjunto de  $c$  classes. Então a função

$$\begin{aligned} F : \mathcal{X} &\rightarrow \mathcal{Y} \rightarrow \mathcal{C} \\ \mathbf{x} &\mapsto \mathbf{y}(\mathbf{x}) \mapsto F(\mathbf{y}(\mathbf{x})) \end{aligned} \tag{2.76}$$

é a *função discriminativa* que mapeia o padrão  $\mathbf{x}$  para a classe  $\mathcal{C}_p$  se o score da classe  $\mathcal{C}_p$  é maior do que o score de qualquer outra classe  $\mathcal{C}_q$ , ou seja  $y_p(\mathbf{x}) > y_q(\mathbf{x})$ ,  $p, q \in \{1, \dots, c\}$ ,  $p \neq q$ , para as  $c$  *scores* de todas as classes  $\mathcal{C}_k$ ,  $k = 1, \dots, c$ .

Essa definição é bastante intuitiva. Escolhe a classe que tenha o valor mais alto de um modelo que atribui um valor quantitativo a um padrão de entrada. Vamos conhecer vários tipos de funções discriminativas. Por exemplo, supondo que as classes tenham uma densidade Gaussiana, será possível definir um classificador que calcula uma probabilidade para cada classe, usando a Regra de Bayes.

■ **Exemplo 2.14** Podemos continuar o exemplo 2.13, decidindo a classe pela função discriminativa

tiva. Considerando a tabela 2.4, temos  $F(\hat{\mathbf{y}}(\mathbf{x}_1)) = \mathcal{C}_1$ ,  $F(\hat{\mathbf{y}}(\mathbf{x}_{51})) = \mathcal{C}_3$  e  $F(\hat{\mathbf{y}}(\mathbf{x}_{101})) = \mathcal{C}_3$ . O padrão  $\mathbf{x}_{101}$  foi mal classificado, pois a sua classe verdadeira é  $\mathcal{C}_2$ . ■

Limitando a quantidade de características a duas, vamos visualizar como a Máquina Linear divide o espaço de características. Escolhemos novamente as duas características mais discriminativas, veja a figura 1.2b na pág. 15, a característica  $x_3$ , comprimento da pétala como eixo  $x$ , e a característica  $x_4$ , largura da pétala, como eixo  $y$ . Obviamente, a arquitetura da Máquina Linear muda. A quantidade de entradas é  $(m+1) = (2+1) = 3$ . A matriz de dados  $X$  (eq. 2.67) tem dimensão  $150 \times 3$ . As saídas no vetor de *scores*  $\mathbf{y}$  não ficam afetadas, pois representam as três classes, então a matriz de saídas  $Y$  é a mesma que aquela definida na (eq. 2.71). A matriz de pesos nova é

$$W = X^\dagger Y = \begin{bmatrix} w_{0,1} & w_{0,2} & w_{0,3} \\ w_{1,1} & w_{1,2} & w_{1,3} \\ w_{2,1} & w_{2,2} & w_{2,3} \end{bmatrix} = \begin{bmatrix} 1.267 & -0.106 & -0.160 \\ -0.251 & 0.324 & -0.073 \\ 0.010 & -0.650 & 0.640 \end{bmatrix}. \quad (2.77)$$

Todos os valores da tabela 2.4 mudam, exceto os *scores* desejados. A figura 2.7 mostra novamente o mesmo espaço de características que na figura 1.2b na pág. 15, porém desta vez com o resultado da classificação de cada uma das coordenadas deste espaço  $(x_3, x_4)$ . Cada coordenada desde o ponto  $(0, -1)$  no canto inferior esquerdo até o ponto oposto diagonalmente  $(8, 4)$  no canto superior direito, avançando por  $\Delta_x = 0.02$  e  $\Delta_y = 0.02$  formam o conjunto de  $251 \times 401 = 100651$  píxeis da imagem. Esses píxeis são submetidos à Máquina Linear, é os *scores* e função de softmax dos *scores* são calculados. A classe classificada é determinada pela função discriminativa pelo maior valor softmax dos *scores*. O valor do maior *score* de cada píxel é associada a uma cor. Esses valores correspondem à escala de cores. Repare que o menor valor do *score* vencedor não pode ser menor que  $\frac{1}{3}$ , pois nesse caso existiria um *score* de outra classe maior que  $\frac{1}{3}$ . O ponto no gráfico  $(3.79, 1.20)$ , onde todos os três *scores* são iguais, na confluência das três fronteiras entre duas classes, é marcado por um asterisco amarelo. Nesse ponto os três *scores* são  $y_1 = y_2 = y_3 = \frac{1}{3}$ . O maior valor teoricamente possível de um *score* é 1.0. Nesse caso os *scores* das outras duas classes teriam o valor zero. Podemos até constatar que a classificação errada da flor  $\mathbf{x}_{101}$  já não acontece neste espaço bidimensional das duas características selecionadas. Os três padrões de teste  $\mathbf{x}_1, \mathbf{x}_{51}, \mathbf{x}_{101}$  são destacados por um marcador “x”.

Claramente se destacam os píxeis, onde uma fronteira entre duas ou três classes se forma. Para marcar esses pontos, um maior e segundo maior *score* foram comparados. Se ficarem iguais estamos em uma fronteira de decisão que divide as regiões de decisão ao qual classe o píxel será atribuído. As fronteiras foram pintadas por uma reta bicolor. As duas cores correspondem às cores das duas classes de flores que a fronteira separa. Alguns *scores* foram posicionadas em cima da fronteira. Nesta posição vale uma igualdade dos dois *scores* das duas classes.

Vamos formalizar ainda o que uma região de uma classe é uma fronteira de decisão.

**Definição 2.3.4 — Região de Decisão de uma Classe.** Seja  $\mathcal{X} \stackrel{\text{def}}{=} \mathbb{R}^m$  o domínio dos padrões de dimensão  $m$ , e seja  $F$  uma função discriminativa, definição 2.3.3. Então, todos os padrões do domínio dos padrões que foram atribuídos à  $k$ -ésima classe  $\mathcal{C}_k$ , ou seja,

$$\mathcal{R}_k \stackrel{\text{def}}{=} \{\mathbf{x} | F(\mathbf{y}(\mathbf{x})) = \mathcal{C}_k\}, \quad (2.78)$$

definem a *região de decisão* da  $k$ -ésima classe.

■ **Exemplo 2.15** Considerando os resultados da tabela 2.4, podemos afirmar que a primeira virgílica  $\mathbf{x}_{101}$  faz parte da região de decisão desta classe, ou seja  $\mathbf{x}_{101} \in \mathcal{R}_3$ , pois dado o vetor de scores  $\mathbf{y}(\mathbf{x}_{101})$ , ou o softmax destes scores tem na terceira posição o maior valor. Assim a função discriminativa escolha a classe  $\mathcal{C}_3$ , ou seja,  $F(\mathbf{y}(\mathbf{x}_{101})) = \mathcal{C}_3$ . ■

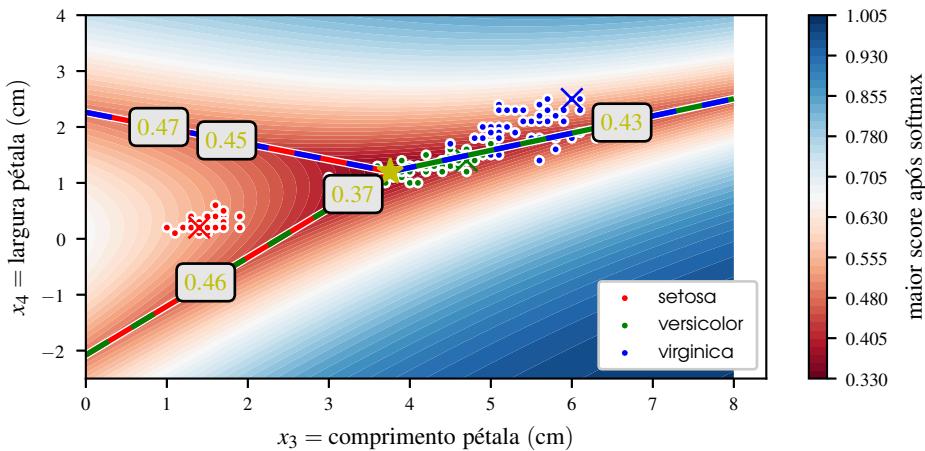


Figura 2.7: Regiões de Decisão e Fronteiras de Decisão, usando aprendizagem pela Máquina Linear. Duas características de Iris foram usadas para permitir a visualização direta. Valores negativos das características não podem ocorrer mas para o eixo  $x_4$  são mantidos para melhorar o aspecto do gráfico. Os *scores* foram normalizados para o intervalo  $[0, 1]$ , usando a função softmax. Perto das fronteiras, tem dois ou até três *scores* parecidos, mas longe das fronteiras, um dos *scores* domina e se aproxima do valor 1. Existe um ponto  $(3.79, 1.20)$ , onde os *scores* das três classes são iguais com o valor  $1/3$ .

Revendo a figura 2.7, fica bem óbvio quais píxeis no espaço das duas características formam as regiões de decisão  $\mathcal{R}_1$ ,  $\mathcal{R}_2$ ,  $\mathcal{R}_3$  de setosa, versicolor e virginica. Observe a sua forma. São polígonos convexos, virginica no norte, setosa sudoeste, versicolor sudeste. Na verdade uma região de decisão tem uma infinidade de padrões que definem a região, desejavelmente todos os padrões que verdadeiramente são de uma determinada classe, caiem na região correta. E isso deve ser verdade para todos o conjunto de treinamento, e futuros padrões que possam aparecer para serem classificados.

**Definição 2.3.5 — Fronteira de Decisão de duas Classes.** Seja  $\mathcal{X} \stackrel{\text{def}}{=} \mathbb{R}^m$  o domínio dos padrões de dimensão  $m$ , e seja  $y$  o vetor dos *scores*, definição 2.3.1. Então, todos os padrões do domínio dos padrões que para duas classes  $\mathcal{C}_p$  e  $\mathcal{C}_q$  tenham o mesmo score, ou seja

$$\mathcal{F}_{p,q} \stackrel{\text{def}}{=} \{\mathbf{x} | \exists p, q \in \{1, \dots, c\}, p \neq q : y_p(\mathbf{x}) = y_q(\mathbf{x})\}, \quad (2.79)$$

definem a *fronteira de decisão* entre as duas classes  $\mathcal{C}_p$  e  $\mathcal{C}_q$ .

Podem existir padrões  $\mathbf{x}$  que definem múltiplas fronteiras, como claramente visível no exemplo da figura 2.7, onde existe um ponto que faz parte de todas as fronteiras possíveis, setosa-versicolor, setosa-virginica, e versicolor-virginica. Ele tem<sup>7</sup> o valor  $\mathbf{x} = [x_3 \ x_4]^\top = [3.758 \ 1.120]^\top$ .

<sup>7</sup>Tendo uma fronteira  $\mathcal{F}_{p,q}$  entre as regiões  $\mathcal{R}_p$  e  $\mathcal{R}_q$ , o valor dos dois *scores* é igual em cima da fronteira, ou seja,  $y_p = \mathbf{w}_p^\top \mathbf{x} = y_q = \mathbf{w}_q^\top \mathbf{x}$ . Em duas dimensões, subtraindo as duas equações,  $y_p - y_q$ , resulta em  $(\mathbf{w}_p^\top - \mathbf{w}_q^\top) \mathbf{x} = [0 \ 0 \ 0]^\top$ . Isto é a equação de uma reta. Em analogia, na fronteira  $\mathcal{F}_{q,r}$  temos  $(\mathbf{w}_q^\top - \mathbf{w}_r^\top) \mathbf{x} = [0 \ 0 \ 0]^\top$ , outra reta. A interseção dessas duas retas (não paralelas) resulta em um único ponto  $\mathbf{x} = [x_1 \ x_2]^\top$ , o encontro das duas fronteiras. É a solução de um sistema linear  $2 \times 2$  que pode ser facilmente resolvido, por exemplo, por Eliminação de Gauss e resolução de um sistema triangular. O sistema é

$$(w_{p,1} - w_{q,1})x_1 + (w_{p,2} - w_{q,2})x_2 = -(w_{p,0} - w_{q,0}) \\ (w_{q,1} - w_{r,1})x_1 + (w_{q,2} - w_{r,2})x_2 = -(w_{q,0} - w_{r,0}).$$

### 2.3.2 Região de Decisão Convexa

Já observamos a convexidade das regiões de decisão na figura 2.7. Um círculo é convexo, uma elipse é convexa, porém esses dois objetos geométricos em duas dimensões não são lineares. Exemplos de objetos convexos lineares em duas dimensões são um retângulo, em três dimensões um paralelepípedo.

**Definição 2.3.6 — Região Convexa.** Sejam  $\mathbf{x}_a \in \mathcal{R}_p$  e  $\mathbf{x}_b \in \mathcal{R}_p$  dois padrões da mesma região de decisão  $\mathcal{R}_p$ . Seja  $\mathbf{x}(\lambda) \stackrel{\text{def}}{=} \lambda \mathbf{x}_a + (1 - \lambda) \mathbf{x}_b$  a reta parametrizada<sup>a</sup> entre os dois pontos, com o parâmetro  $\lambda \in [0, 1]$ , representando todos os pontos entre  $\mathbf{x}_a$  e  $\mathbf{x}_b$ . Seja  $y_p(\mathbf{x}) \in \mathbb{R}$  o score da região da definição 2.3.1. Então a região  $\mathcal{R}_p$  é *convexa*, se a reta parametrizada nunca sai da região, ou seja  $\mathbf{x}(\lambda) \in \mathcal{R}_p$ , satisfazendo a propriedade

$$y_p(\mathbf{x}(\lambda)) = y_p(\lambda \mathbf{x}_a + (1 - \lambda) \mathbf{x}_b) \leq \lambda y_p(\mathbf{x}_a) + (1 - \lambda) y_p(\mathbf{x}_b). \quad (2.80)$$

<sup>a</sup>Letra grega  $\lambda$  = “lambda”.

**Corolário 2.3.1** As regiões de decisão da Máquina Linear são convexas.

*Demonstração.* Pela definição da região de decisão  $\mathcal{R}_p$ , ela tem o maior *score* entre todas as regiões, ou seja  $y_p(\mathbf{x}) > y_q(\mathbf{x})$ ,  $p, q \in \{1, \dots, c\}$ ,  $p \neq q$ . Então

$$y_p(\mathbf{x}(\lambda)) = y_p(\lambda \mathbf{x}_a + (1 - \lambda) \mathbf{x}_b) \quad (2.81a)$$

$$= \lambda y_p(\mathbf{x}_a) + (1 - \lambda) y_p(\mathbf{x}_b) \quad (2.81b)$$

$$> \lambda y_q(\mathbf{x}_a) + (1 - \lambda) y_q(\mathbf{x}_b) = y_q(\mathbf{x}(\lambda)) \quad (2.81c)$$

$$\Rightarrow \forall \mathbf{x} \in [\mathbf{x}_a, \mathbf{x}_b] : y_p(\mathbf{x}) > y_q(\mathbf{x}) \quad (2.81d)$$

A prova diz que qualquer padrão no caminho entre dois padrões da região de decisão de uma classe faz parte da mesma região, assim sendo impossível passar por uma região estranha. Na transição de (eq. 2.81a) para (eq. 2.81b) foi usado o fato que a função de *score*  $y(\mathbf{x})$  da (eq. 2.74) em um modelo linear por definição é linear. Na transição de (eq. 2.81b) para (eq. 2.81c) foi usado o fato que para qualquer  $\mathbf{x} \in \mathcal{X}$  no espaço de características, o score para a região  $\mathcal{R}_p$  é o maior de todos,  $y_p(\mathbf{x}) > y_q(\mathbf{x})$ . ■

A convexidade das regiões de decisão da Máquina Linear enfatiza as limitações naturais deste modelo linear.

1. Um região de decisão é contígua. Isso é uma consequência direta da convexidade. Entre uma parte da região de decisão de uma classe e uma outra parte da mesma região, não pode se enfiar uma segunda região.
2. As fronteiras das regiões têm um caráter linear. Não é possível traçar uma fronteira que tenha um aspecto mais complexo, por exemplo uma curva quadrática.

## 2.4 Classificador Linear Binário

Uma das arquiteturas mais importantes são classificadores binários. Neste modelo existem somente duas classes, a classe positiva  $\mathcal{C}_+$  e a classe negativa  $\mathcal{C}_-$ . Por exemplo, em um contexto de um processo industrial, queremos saber, se o processo está funcionando normalmente, ou se

há um problema<sup>8</sup>. O classificador binário tem um conjunto de particularidades, comparado com o classificador multi-classe. Por exemplo, o binário precisa somente uma única saída que expressa a qual classe pertence um padrão. Podemos atribuir o valor  $-1$  a primeira classe, e o valor  $1$  a segunda classe. Além disso, existe para o classificador binário uma *matriz de confusão* que conhecemos mais adiante na seção 9.2 na pág. 265. Primeiramente vamos estudar o classificador binário linear. Este modelo parece inapropriado em geral para resolver problemas complexos de classificação, porém vamos ver que com capacidades adicionais se torna um modelo competitivo. Especialmente a Máquina de Vetor de Suporte incrementa duas qualidades adicionais ao classificador binário básico, assim o transformando em um modelo bastante otimizado. O primeiro aspecto é a maximização da margem entre as duas classes, e o segundo aspecto é a mudança do espaço de características através de *kernels*. Voltaremos ao assunto dos *kernels* posteriormente.

### 2.4.1 Modelo do Classificador Linear Binário

Usamos novamente um modelo linear da definição 2.1.2 na pág. 32 e definição 2.2.1 na pág. 53 para descrever um classificador linear binário. Embora haja uma definição parecida com a da regressão linear no contexto de ajuste de modelo, a representação do classificador binário é um conceito muito diferente. Na regressão linear existiam variáveis independentes  $\mathbf{x}$  e variáveis dependentes  $\mathbf{y}$ , e o objetivo era aprender um mapeamento funcional  $\mathbf{y} = \mathbf{f}(\mathbf{x})$ . Por outro lado, no classificador binário não existem variáveis dependentes. Vamos modelar a *fronteira* de decisão entre as duas classes. O modelo é uma equação  $f(\mathbf{x}) = 0$ . As soluções desta equação, ou seja, as raízes da função  $f(\mathbf{x})$  são a fronteira. No caso de um espaço bidimensional de características, essa fronteira é simplesmente uma reta que separa duas classes, definido<sup>9</sup> no espaço  $xy$  como  $Ax + By = C$ , ou na nossa linguagem do espaço de características como  $w_0 + w_1x_1 + w_2x_2 = 0$ . Na apresentação dos exemplos vamos aposentar temporariamente uma das três espécies de Iris, pois podemos somente trabalhar com duas classes.

No contexto da Máquina Linear já vimos que foi possível modelar uma fronteira através de uma *equação*, por exemplo a igualdade de dois *scores*,  $y_p(\mathbf{x}) = y_q(\mathbf{x})$ , ou, equivalentemente  $y_p(\mathbf{x}) - y_q(\mathbf{x}) = 0$ . Vamos pensar um pouco mais o que significa uma equação em um espaço vetorial, o mais conhecido sendo o sistema de coordenadas cartesiano de duas dimensões. Uma equação é uma *restrição*. Se temos uma única dimensão, estamos trabalhando com valores  $x \in \mathbb{R}$ . Uma equação  $x = 7$  define uma fronteira nessa escala unidimensional. Valores abaixo de sete pertencem a um semiespaço do sistema de coordenadas cartesiano de uma dimensão, por exemplo,  $-1, 2, 3, -2, 5, \dots$ , valores acima de sete, por exemplo,  $12, 9, 8, 2, 10, \dots$  pertencem ao semiespaço complementar. A fronteira define-se pela equação  $x = 7 = 0$ .

Em duas dimensões, a equação  $x^2 + y^2 - 1 < 0$  define o círculo unitário como região interior. A circunferência  $x^2 + y^2 - 1 = 0$  é a fronteira e  $x^2 + y^2 - 1 > 0$  é a região que define o exterior do círculo. Esta fronteira não é linear, ela é quadrática. Uma fronteira linear em duas dimensões é uma reta, por exemplo em duas dimensões, no espaço  $xy$ , a diagonal  $x = y$  separa o sistema de coordenadas cartesiano nos dois semiespaços acima e abaixo da diagonal. Para definir qualquer reta em 2-D, precisamos três parâmetros, por exemplo  $-2x + 3y = 4$ . Falando em termos gerais, temos no espaço bidimensional  $(x_1, x_2)$  de características três parâmetros que definem a fronteira linear como  $w_0 + w_1x_1 + w_2x_2 = 0$ . Em geral, em qualquer dimensão, a fronteira linear que separa duas classes é um *hiperplano*.

<sup>8</sup>Frequentemente, no contexto de detecção de falhas de um processo, somente existem padrões da classe positiva. Isso define um problema mais difícil, o chamado “Classificador de Classe Única”, *One-class classification*. Um método tradicional da área da estatística para tratar deste problema, é o teste de hipóteses

<sup>9</sup>No ensino básico, a definição comum de uma reta é em forma de função  $y(x) = mx + b$  que corresponde a conversão da representação paramétrica  $Ax + By = C$  em  $y(x; A, B, C) = \frac{-A}{B}x + \frac{C}{B}$ . Porém ela não é apropriada para modelar retas verticais, pois nesse caso  $B = 0$ .

Dimensão	Quantidade de parâmetros	Denominação	Definição
1	2	Ponto	$w_0 + w_1 x = 0$
2	3	Reta	$w_0 + w_1 x_1 + w_2 x_2 = 0$
3	4	Plano	$w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 = 0$
$m > 3$	$m+1$	Hiperplano	(eq. 2.82), (eq. 2.84)

Tabela 2.5: Hiperplano em várias dimensões

**Definição 2.4.1 — Hiperplano, Dicotomia.** Seja  $\mathcal{X} = \mathbb{R}^m$  um espaço de características de dimensão  $m$ . Seja  $\mathbf{w} = [w_1 \ \dots \ w_j \ \dots \ w_m]^T$  um vetor de  $m$  parâmetros (ou pesos) e seja o escalar  $w_0 = b$  o viés. A equação linear com  $(m+1)$  parâmetros  $\boldsymbol{\theta} = (w_0, w_1, \dots, w_j, \dots, w_m)$

$$d(\mathbf{x}; \mathbf{w}, b) = \mathbf{w} \cdot \mathbf{x} + b = w_1 x_1 + \dots + w_j x_j + \dots + w_m x_m + b = 0, \quad (2.82)$$

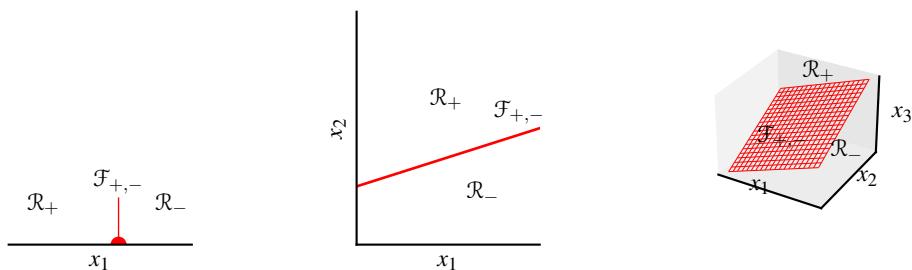
define um *hiperplano* que separa o espaço de características em dois semiespaços de características, “de um lado” e “do outro lado” do hiperplano, criando uma *dicotomia*. Vamos chamar os dois lados opostos do hiperplano  $\mathcal{R}_+$  e  $\mathcal{R}_-$ . Portanto

$$\text{padrão } \mathbf{x} \begin{cases} \text{do lado positivo do hiperplano , } \mathbf{x} \in \mathcal{R}_+, & \text{se } d(\mathbf{x}; \mathbf{w}, b) > 0 \\ \text{no hiperplano,} & \text{se } d(\mathbf{x}; \mathbf{w}, b) = 0 \\ \text{do lado negativo do hiperplano , } \mathbf{x} \in \mathcal{R}_-, & \text{se } d(\mathbf{x}; \mathbf{w}, b) < 0. \end{cases} \quad (2.83)$$

Podemos novamente usar a representação aumentada definição 2.1.9 na pág. 44 para obter uma forma mais compacta do hiperplano

$$d(\tilde{\mathbf{x}}; \tilde{\mathbf{w}}) = \tilde{\mathbf{w}} \cdot \tilde{\mathbf{x}} = \mathbf{w} \cdot \mathbf{x} + w_0 = 0. \quad (2.84)$$

A tabela 2.5 mostra a natureza do hiperplano para qualquer dimensão. A nossa percepção normalmente se esgota na dimensão três, porém matematicamente o hiperplano pode ser usado para criar um classificador em qualquer dimensão. Por exemplo, as nossas flores estão situadas em um espaço de  $m = 4$  dimensões, pois temos quatro características. Então bastam cinco parâmetros para ter um separador de duas classes, por exemplo para criar uma dicotomia de setosa e virginica. A figura 2.8 ilustra o hiperplano, as duas regiões de decisão criadas pelo hiperplano e a fronteira entre as regiões definida pelo próprio hiperplano. Vale a pena comentar a especificação dos hiper-

Figura 2.8: Hiperplanos, regiões de decisão  $\mathcal{R}$ , fronteiras de decisão  $\mathcal{F}$  em uma, duas e três dimensões.

planos. Em uma dimensão, o hiperplano é definido como  $d(x, w, b) = wx + b = 0$ . Novamente dois

parâmetros  $w = w_1$  e  $b = w_0$  são necessários para permitir que a fronteira entre as duas regiões seja diferente da origem  $x = 0$ . Se o hiperplano fosse definido por  $d(x, w) = wx = 0$ , usando um único parâmetro  $w$ , a fronteira seria sempre  $x = 0$ . A única possibilidade de definir se o lado esquerdo, ou o lado direito da origem seria a região  $\mathcal{R}_+$  da classe positiva, seria mudar o sinal de  $w$ . Em princípio não existe restrição para os valores de  $w$ , desde que seja diferente de zero. O valor de  $w = 1$  é um caso especial, em que a norma  $\|\mathbf{w}\| = |w| = 1$  do vetor degenerado para uma dimensão  $\mathbf{w} = w$  é igual a um. A fronteira nesse caso é  $\mathcal{F}_{+,-} = -b$ . O hiperplano  $wx + b = 1 \cdot x - 7 = 0$ , por exemplo, classifica todos os valores acima de sete como classe positiva, a fronteira é  $-b = -7 = 7$ , o peso é  $w = 1$ , o viés  $b = -7$ . A mesma fronteira, porém com regiões positiva e negativa trocadas, seria o hiperplano  $-2x + 14 = 0$ , ou seja, multiplicando os dois lados por  $-2$ , resultando em  $w = -2$  e  $b = 14$ . Podemos definir um hiperplano canônico que preserve a fronteira de decisão.

**Definição 2.4.2 — Hiperplano Canônico.** Seja  $d(\mathbf{x}; \mathbf{w}, b) = \mathbf{w} \cdot \mathbf{x} + b = 0$  o hiperplano da definição 2.4.1. O *hiperplano canônico* é definido como

$$\hat{d}(\mathbf{x}; \mathbf{w}, b) \stackrel{\text{def}}{=} \frac{d(\mathbf{x}; \mathbf{w}, b)}{\|\mathbf{w}\|} = \frac{\mathbf{w} \cdot \mathbf{x} + b}{\|\mathbf{w}\|} = \hat{\mathbf{w}} \cdot \mathbf{x} + \frac{b}{\|\mathbf{w}\|} = 0, \quad (2.85)$$

onde  $\|\mathbf{w}\|$  é a norma quadrática da (eq. 1.10) na pág. 14 do vetor de pesos, e  $\hat{\mathbf{w}} = \frac{\mathbf{w}}{\|\mathbf{w}\|}$  é o vetor unitário do vetor de pesos, veja (eq. 1.5) na pág. 12. O hiperplano canônico define a mesma fronteira e regiões de decisão como o hiperplano, definidos na (eq. 2.83). A prova é trivial, pois uma divisão por um valor positivo da equação do hiperplano não muda nem equação, nem inequação.

**Proposição 2.4.1** Observando a definição da função de *score* da definição 2.3.1, da função discriminativa, definição 2.3.3, da região de decisão, definição 2.3.4, e da fronteira de decisão definição 2.3.5, podemos constatar que a relação entre o valor zero, do lado direito da equação do hiperplano, e o lado esquerdo  $d(\mathbf{x}; \mathbf{w}, b)$ , ou a sua forma canônica  $\hat{d}(\mathbf{x}; \mathbf{w}, b)$  é idêntica à relação dos *scores*  $y_+$  e  $y_-$  das duas classes  $\mathcal{C}_+$  e  $\mathcal{C}_-$ , sendo

$$\begin{array}{lllll} d(\mathbf{x}; \mathbf{w}, b) > 0 & \iff & y_+(\mathbf{x}) > y_-(\mathbf{x}) & \iff & \mathbf{x} \in \mathcal{R}_+ \\ d(\mathbf{x}; \mathbf{w}, b) = 0 & \iff & y_+(\mathbf{x}) = y_-(\mathbf{x}) & \iff & \mathbf{x} \in \mathcal{F}_{+,-} \\ d(\mathbf{x}; \mathbf{w}, b) < 0 & \iff & y_+(\mathbf{x}) < y_-(\mathbf{x}) & \iff & \mathbf{x} \in \mathcal{R}_- \end{array}$$

Podemos simplificar o modelo do classificador linear binário, unificando as duas funções de *score*  $y_+$  e  $y_-$ , definindo uma única função de score  $y$  que simplesmente é a diferença das duas funções de *score* da classe positiva e negativa.

**Definição 2.4.3 — Função de Score de Classificador Binário.** Sejam  $y_+(\mathbf{x})$  e  $y_-(\mathbf{x})$  os únicos *scores* de um modelo linear múltiplo definidos na (eq. 2.64) na pág. 53. Para facilitar, usamos a representação do produto escalar sem os vetores aumentados, ou seja

$$\begin{aligned} y_+(\mathbf{x}) &= \mathbf{w}_+ \cdot \mathbf{x} + b_+ \\ y_-(\mathbf{x}) &= \mathbf{w}_- \cdot \mathbf{x} + b_- \end{aligned}$$

Então a função de *score* do classificador binário é definido como diferença dos dois *scores* da

classe positiva  $\mathcal{C}_+$  e  $\mathcal{C}_-$ . Assim

$$\begin{aligned} y(\mathbf{x}) &\stackrel{\text{def}}{=} y_+(\mathbf{x}) - y_-(\mathbf{x}) = d(\mathbf{x}; \mathbf{w}_+, b_+) - d(\mathbf{x}; \mathbf{w}_-, b_-) \\ &= \mathbf{w}_+ \cdot \mathbf{x} + b_+ - (\mathbf{w}_- \cdot \mathbf{x} + b_-) = (\mathbf{w}_+ - \mathbf{w}_-) \cdot \mathbf{x} + (b_+ - b_-) \\ &\stackrel{\text{def}}{=} \mathbf{w} \cdot \mathbf{x} + b. \end{aligned} \quad (2.86)$$

A consequência dessa simplificação é que o lado esquerdo da definição do hiperplano coincide com este *score* único e a função discriminativa é simplesmente o sinal do *score*, onde o valor  $+1$  representa a classe positiva  $\mathcal{C}_+$ , e o valor  $-1$  representa a classe negativa  $\mathcal{C}_-$ .

**Definição 2.4.4 — Função Discriminativa do Classificador Binário.** Seja  $y(\mathbf{x})$  a função de *score* do classificador linear binário. Então a função discriminativa que atribui o padrão  $\mathbf{x}$  a classe positiva ou a classe negativa é o sinal desta função de score.

$$\begin{aligned} F(\mathbf{x}) &= \operatorname{sgn}[y(\mathbf{x})] = \operatorname{sgn}[d(\mathbf{x}; \mathbf{w}, b)] \\ &= \operatorname{sgn}[\mathbf{w} \cdot \mathbf{x} + b]. \end{aligned} \quad (2.87)$$

## 2.4.2 Orientação do Hiperplano e Distâncias

Uma questão importante é como os parâmetros  $\mathbf{w}$  e  $b$  dão forma ao hiperplano. A resposta é que  $\mathbf{w}$  é o vetor normal do hiperplano e  $b$  determina a distância entre a origem do sistema de coordenadas cartesiano e o hiperplano.

**Corolário 2.4.2 — Vetor Normal do Hiperplano.** O vetor de pesos  $\mathbf{w}$  é o vetor normal do hiperplano. O vetor normal é ortogonal a qualquer vetor situado no hiperplano.

*Demonstração.* Dois vetores  $\mathbf{u}$  e  $\mathbf{v}$  são ortogonais, se o seu produto escalar é zero, ou seja  $\mathbf{u} \cdot \mathbf{v} = 0$ . Sejam  $\mathbf{x}_p$  e  $\mathbf{x}_q$  dois pontos arbitrários que fazem parte do hiperplano. Então

$$\begin{aligned} d(\mathbf{x}_p) &= d(\mathbf{x}_q) = 0 \\ \iff \mathbf{w} \cdot \mathbf{x}_p + b &= \mathbf{w} \cdot \mathbf{x}_q + b \\ \iff \mathbf{w} \cdot (\mathbf{x}_p - \mathbf{x}_q) &= 0, \end{aligned} \quad (2.88)$$

ou seja, qualquer vetor de diferença entre dois pontos do hiperplano é ortogonal ao vetor  $\mathbf{w}$  que por definição é então o vetor normal do hiperplano. ■

Neste ponto uma conta informal que envolve pontos e vetores ajuda bastante. Um ponto  $\mathbf{x}$  está sempre ligado a um sistema de coordenadas, ao contrário de um vetor que expressa um deslocamento relativo. Os dois objetos tem a mesma representação, um número multidimensional.

**Definição 2.4.5 — Contas com Pontos e Vetores.**

Ponto + Vetor = Ponto	Translação de ponto por vetor
Ponto - Ponto = Vetor	Diferença de pontos independente de sistema de coordenadas
Vetor + Vetor = Vetor	Aditividade de vetores

A definição ortodoxa de uma distância consta que ela é zero ou positiva. Porém no contexto

do classificador linear binário faz sentido atribuir uma direção, ou equivalentemente, um sinal, à distância entre um padrão  $\mathbf{x}$  e o hiperplano, pois esse sinal pode ser usado diretamente para identificar a região de decisão. A região da classe positiva  $\mathcal{R}_+$  será associada a uma direção positiva, e a região da classe negativa  $\mathcal{R}_-$  será associada a uma direção negativa.

**Definição 2.4.6 — Distância Direcionada entre um Ponto e o Hiperplano.** A direção da distância entre um padrão  $\mathbf{x}$  e o hiperplano  $d(\mathbf{x}; \mathbf{w}, b) = 0$  definido na (eq. 2.82), ou o hiperplano canônico  $\hat{d}(\mathbf{x}; \mathbf{w}, b) = 0$  definido na (eq. 2.85) é definido como sinal do produto escalar entre  $\mathbf{x}$  e  $\mathbf{w}$ , ou respectivamente  $\hat{\mathbf{w}}$ .

$$\text{Direção da distância} \stackrel{\text{def}}{=} \operatorname{sgn}(\mathbf{w} \cdot \mathbf{x}) = \operatorname{sgn}(\hat{\mathbf{w}} \cdot \mathbf{x}) = \operatorname{sgn}(\cos \angle(\mathbf{w}, \mathbf{x})) = \operatorname{sgn}(\cos \angle(\hat{\mathbf{w}}, \mathbf{x})).$$

**Teorema 2.4.3 — Distância Direcionada ao Hiperplano.** A função de *score* do hiperplano canônico é a distância Euclidiana *direcionada* de um padrão  $\mathbf{x}$  do hiperplano.

*Demonstração.* Equivalentemente, a distância entre o hiperplano  $\hat{d}(\mathbf{x}; \mathbf{w}, b) = 0$  e o padrão  $\mathbf{x}$  é a distância direcionada  $\pm \|\mathbf{x} - \mathbf{x}^{\text{proj}}\| = \operatorname{sgn}(\hat{\mathbf{w}} \cdot \mathbf{x}) \|\mathbf{x} - \mathbf{x}^{\text{proj}}\|$  entre  $\mathbf{x}$  e o ponto  $\mathbf{x}^{\text{proj}}$  que é o ponto ortogonalmente projetado em cima do hiperplano, veja a figura 2.9. Do corolário 1.1.2 na pág. 12 no contexto da definição do produto escalar sabemos que o produto escalar de um vetor com um vetor unitário paralelo é o módulo do vetor, vezes a direção. Primeiro definimos o vetor diferença  $\mathbf{x} - \mathbf{x}^{\text{proj}}$  entre o ponto  $\mathbf{x}$  e a sua versão projetada no hiperplano  $\mathbf{x}^{\text{proj}}$ . O módulo desse vetor, vezes a direção, é exatamente a distância procurada. Então

$$\operatorname{sgn}(\hat{\mathbf{w}} \cdot \mathbf{x}) \|\mathbf{x} - \mathbf{x}^{\text{proj}}\| = \hat{\mathbf{w}} \cdot [\mathbf{x} - \mathbf{x}^{\text{proj}}] \quad (2.89a)$$

$$= \hat{\mathbf{w}} \cdot \mathbf{x} - \hat{\mathbf{w}} \cdot \mathbf{x}^{\text{proj}} \quad (2.89b)$$

$$= \hat{\mathbf{w}} \cdot \mathbf{x} + \frac{b}{\|\mathbf{w}\|} - \left[ \hat{\mathbf{w}} \cdot \mathbf{x}^{\text{proj}} + \frac{b}{\|\mathbf{w}\|} \right] \quad (2.89c)$$

$$= \hat{d}(\mathbf{x}; \mathbf{w}, b) - \hat{d}(\mathbf{x}^{\text{proj}}; \mathbf{w}, b) \quad (2.89d)$$

$$= \hat{d}(\mathbf{x}; \mathbf{w}, b) \quad (2.89e)$$

A (eq. 2.89a) justifica-se pelo corolário 1.1.2 na pág. 12. A transição de (eq. 2.89d) para (eq. 2.89e) explica-se pelo fato que o ponto projetado  $\mathbf{x}^{\text{proj}}$  faz parte do hiperplano, sendo o seu *score* igual a zero. ■

Consequentemente podemos considerar o lado esquerdo  $d(\mathbf{x}; \mathbf{w}, b)$  do hiperplano como uma distância não normalizada, ou equivalentemente como uma versão escalada  $\|\mathbf{w}\| \hat{d}(\mathbf{x}; \mathbf{w}, b)$  da distância Euclidiana  $\hat{d}(\mathbf{x}; \mathbf{w}, b)$  entre um ponto  $\mathbf{x}$  e o hiperplano, ou seja

$$d(\mathbf{x}; \mathbf{w}, b) = \|\mathbf{w}\| \hat{d}(\mathbf{x}; \mathbf{w}, b). \quad (2.90)$$

Podemos ainda definir a projeção ortogonal de um ponto qualquer  $\mathbf{x}$  em cima do hiperplano.

**Definição 2.4.7 — Projeção Ortogonal no Hiperplano.** Seja o hiperplano definido pelo vetor de pesos  $\mathbf{w}$  e o viés  $b$ . A projeção ortogonal  $\mathbf{x}^{\text{proj}}$  de um ponto  $\mathbf{x}$  em cima do hiperplano é

$$\mathbf{x}^{\text{proj}} = \mathbf{x} - \hat{\mathbf{w}} \hat{d}(\mathbf{x}; \mathbf{w}, b) = \mathbf{x} - \hat{\mathbf{w}} \frac{\mathbf{w} \cdot \mathbf{x} + b}{\|\mathbf{w}\|}. \quad (2.91)$$

*Demonstração.* No teorema 2.4.3, já foi calculada a distância entre o ponto e o ponto projetado como  $\hat{d}(\mathbf{x}; \mathbf{w}, b)$ . Multiplicando o vetor normal normalizado  $\hat{\mathbf{w}}$  por essa distância é o vetor diferença entre esses dois pontos, veja a definição 2.4.5. Pelo balanço, a soma do vetor da origem até o ponto projetado  $\mathbf{x}^{\text{proj}} - \mathbf{0}$ , mais o vetor  $\hat{\mathbf{w}}\hat{d}(\mathbf{x}; \mathbf{w}, b)$  da projeção até o ponto  $\mathbf{x}$  equivale o vetor  $\mathbf{x} - \mathbf{0}$  da origem até o ponto  $\mathbf{x}$ . ■

■ **Exemplo 2.16** A figura 2.9 ilustra a situação no espaço de características de duas e três dimensões. Em duas dimensões, o vetor normal é  $\mathbf{w}_{2D} = \frac{1}{16} [3 \ 4]^T$ , e em três dimensões é  $\mathbf{w}_{3D} = \frac{1}{16} [3 \ 4 \ 12]^T$ . Em ambas as dimensões o viés é  $b = -13/16$ . As normas são  $\|\mathbf{w}_{2D}\| = \frac{1}{16} \sqrt{9 + 16} = 5/16$  e  $\|\mathbf{w}_{3D}\| = \frac{1}{16} \sqrt{9 + 16 + 144} = 13/16$ . Como as duas normas são menor que um, os vetores unitários  $\hat{\mathbf{w}}_{2D}$  e  $\hat{\mathbf{w}}_{3D}$  são mais compridos que os originais. No gráfico, os vetores  $\mathbf{w}$  azuis são os originais e os vermelhos  $\hat{\mathbf{w}}$  os normalizados. Em 2-D, o ponto e a sua projeção no hiperplano são  $\mathbf{x} = [5 \ 3]^T$  e  $\mathbf{x}^{\text{proj}} = [3.32 \ 0.76]^T$ . A distância de  $\mathbf{x}$  até o hiperplano é  $\hat{d}(\mathbf{x}; \mathbf{w}_{2D}, b) = (\mathbf{w}_{2D} \cdot \mathbf{x} + b) / \|\mathbf{w}_{2D}\| = \left\{ \frac{1}{16} [3 \ 4]^T \cdot [5 \ 3]^T - \frac{13}{16} \right\} / \frac{5}{16} = \frac{14}{5} = 2.8$ . Em 3-D, o ponto e a sua projeção no hiperplano são  $\mathbf{x} = [3 \ 5/2 \ 1/2]^T$  e  $\mathbf{x}^{\text{proj}} = [2.79 \ 2.22 \ -0.35]^T$ . A distância de  $\mathbf{x}$  até o hiperplano é  $\hat{d}(\mathbf{x}; \mathbf{w}_{3D}, b) = 0.92$ . ■

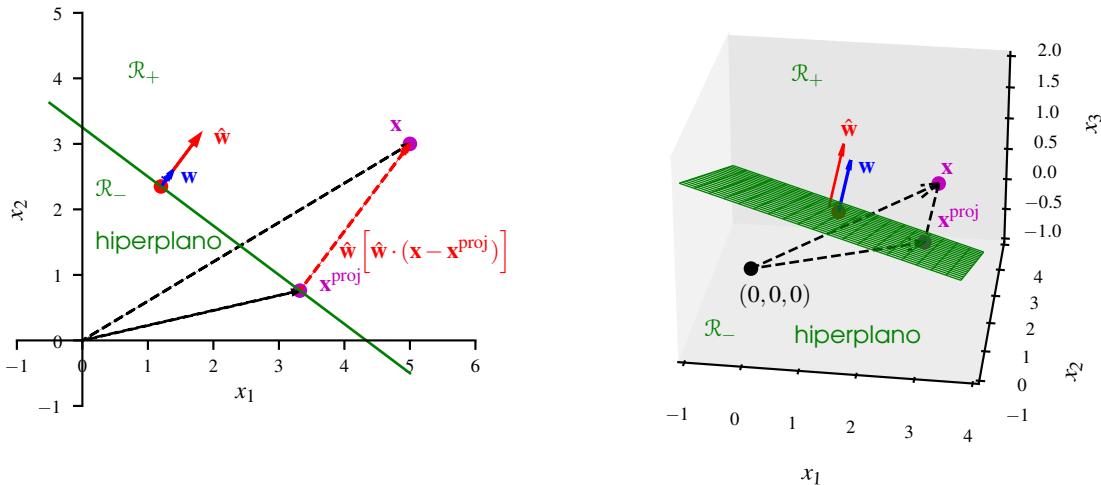


Figura 2.9: Hiperplano, vetor normal, vetor normal normalizado, ponto e ponto projetado ortogonalmente em cima do hiperplano em duas e três dimensões.

Todas as definições e observações sobre a classificação linear binária são necessárias para entender bem conceitos mais avançados. Por exemplo, a Máquina de Vetor de Suporte define como um *vetor de suporte* aquele que tem exatamente uma distância não normalizada da (eq. 2.90)  $d(\mathbf{x}; \mathbf{w}, b)$  de +1 ou -1.

### 2.4.3 Aprendizado do Classificador Linear Binário

Falta a descrição como o classificador linear binário é treinado. Podemos usar o algoritmo 1 para obter os pesos  $\mathbf{w}$  e o viés  $b$  do classificador linear binário da definição 2.4.3. Para adequar ao algoritmo, usamos a representação aumentada, incorporando o viés  $b$  como 0-ésima componente em  $\tilde{\mathbf{w}}$ . O vetor de pesos  $\tilde{\mathbf{w}}$  de dimensão  $(m+1) \times 1$  então é  $\tilde{\mathbf{w}} = \tilde{X}^\dagger \mathbf{y}$ , onde  $\tilde{X}$  é a matriz de dados de dimensão  $n \times (m+1)$ . A matriz de saídas  $Y$  no algoritmo 1 é substituída pelo vetor

de saída  $\mathbf{y} = [1 \ \cdots \ 1 \ -1 \ \cdots \ -1]^T$  de dimensão  $n \times 1$ . As primeiras  $n_+$  linhas em  $\tilde{\mathbf{X}}$  e  $\mathbf{y}$  correspondem aos padrões da classe positiva, e os últimos  $n_-$  linhas correspondem aos padrões da classe negativa, sendo a quantidade total de padrões  $n = n_+ + n_-$ .

Comparamos o resultado da aprendizagem linear binária com a regressão linear múltipla, usando o método da codificação *One-Hot* para duas classes. Temos como saída desejada um vetor bidimensional  $\mathbf{y} = [y_+ \ y_-]^T$  que para padrões da classe positiva  $\mathcal{C}_+$  tem o valor  $\mathbf{y} = [1 \ 0]^T$ , e para padrões da classe negativa  $\mathcal{C}_-$  tem o valor  $\mathbf{y} = [0 \ 1]^T$ . A matriz de pesos teria duas colunas  $W = [\mathbf{w}_+ \ \mathbf{w}_-]$ , um vetor de pesos para cada uma das classes. O modelo da regressão linear múltipla, a Máquina Linear da definição 2.2.1 na pág. 53, com somente as duas classes especializa para

$$\begin{aligned} W &= [\mathbf{w}_+ \ \mathbf{w}_-] \\ \mathbf{y} &= [y_+ \ y_-] = W^T \mathbf{x} = [\mathbf{w}_+ \ \mathbf{w}_-]^T \mathbf{x} = [\mathbf{w}_+^T \mathbf{x} \ \mathbf{w}_-^T \mathbf{x}] . \end{aligned} \quad (2.92)$$

Este *score* é um vetor de linha de dimensão  $1 \times 2$ .

■ **Exemplo 2.17** Usando somente versicolor e virginica, e somente característica  $x_3$ , comprimento da pétala e característica  $x_4$ , largura da pétala, vamos comparar os pesos aprendidos. Por um lado a Máquina Linear com duas classes, por outro lado o classificador linear binário que usa somente um *score* unidimensional. A Máquina linear aprendeu a matriz de pesos

$$W = [\mathbf{w}_+ \ \mathbf{w}_-] = \begin{bmatrix} w_{0,+} & w_{0,-} \\ w_{1,+} & w_{1,-} \\ w_{2,+} & w_{2,-} \end{bmatrix} = \begin{bmatrix} -1.582 & 2.582 \\ 0.198 & -0.198 \\ 0.663 & -0.663 \end{bmatrix} .$$

O classificador linear binário aprendeu o vetor de pesos aumentado

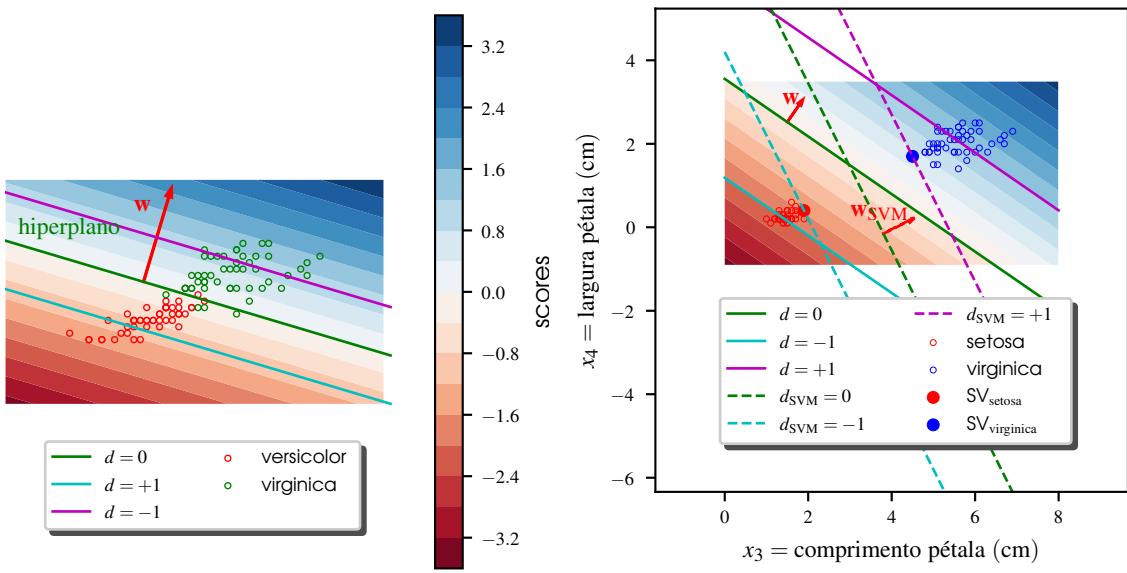
$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} -4.163 \\ 0.395 \\ 1.327 \end{bmatrix} ,$$

que é realmente a primeira coluna de  $W$  menos a segunda coluna de  $W$ , portanto o resultado é idêntico, condizendo com a definição 2.4.3. A situação dos padrões e o hiperplano  $d(\mathbf{x}; \mathbf{w}, b) = 0$  aprendido é ilustrado na figura 2.10a. Repare que as duas classes se sobrepõem no espaço das características. Além do hiperplano com  $d(\mathbf{x}; \mathbf{w}, b) = 0$ , os dois hiperplanos paralelos com  $d(\mathbf{x}; \mathbf{w}, b) \pm 1$  são visualizadas. ■

#### 2.4.4 Coordenadas Homogêneas para Representação Aumentada dos Padrões e dos Parâmetros

A figura 2.11 ilustra, porque é necessário ter um parâmetro a mais do que a dimensão dos padrões. O espaço vetorial tridimensional mostrado é definido por três variáveis. A primeira é o viés  $b$  dos parâmetros, ou pela componente  $x_0$  do padrão. Aqui começamos a enumeração da dimensão do padrão em zero para ser consistente com a definição do vetor aumentado. A segunda e terceira dimensão são definidas pelos pesos  $w_1$  e  $w_2$ , ou pelas componentes  $x_1$  e  $x_2$  do padrão, respectivamente. Podemos constatar que os padrões  $\mathbf{x}$  e os parâmetros  $\mathbf{w}$  são elementos do mesmo espaço vetorial. Isso é confirmado pelo fato que aplicamos o produto escalar entre os padrões e os parâmetros, e isso é somente possível para dois operandos do mesmo espaço.

Queremos traçar o hiperplano separador no espaço  $(x_1, x_2)$ , neste caso uma reta. Pelo problema da separação das duas classes de iris, esta reta não pode passar pela origem  $(x_1, x_2) = (0, 0)$  em 2-D, pois seria assim impossível dividir razoavelmente os 50 padrões de cada classe. Neste



(a) Hiperplano e vetor normal em classificação linear binária em duas dimensões aprendido pelo algoritmo 1. Duas classes não linearmente separáveis. A barra lateral de cores mostra o *score* da classificação binária linear, (eq. 2.86).

(b) Hiperplano aprendido por regressão linear, comparado com hiperplano aprendido pelo modelo da SVM.

Figura 2.10: Estudo comparativo de classificadores lineares binários.

caso todos os 100 padrões do problema ficariam de um dos dois lados da reta. Confirme isso na figura 2.10. Por essa razão tem que existir o terceiro parâmetro, na dimensão representada pelo eixo  $x_0$ . Este sistema de coordenadas 3-D, com os eixos  $(x_0, x_1, x_2)$  define um sistema de *coordenadas homogêneas* [17]. Podemos escolher qualquer escalar

$$h \neq 0,$$

e transformar um ponto 2-D

$$\begin{bmatrix} x_1 & x_2 \end{bmatrix}^\top$$

em um ponto equivalente 3-D

$$\begin{bmatrix} h & h \cdot x_1 & h \cdot x_2 \end{bmatrix}^\top.$$

Então neste exemplo bidimensional, podemos dizer que qualquer flor iris com duas características tem uma quantidade infinita de replicas da mesma flor em coordenadas homogêneas para valores de  $h$  diferentes. Uma escolha óbvia para o escalar é

$$h = 1.$$

Dessa maneira podemos explicar a versão aumentada de um padrão

$$\begin{bmatrix} 1 & x_1 & x_2 \end{bmatrix}^\top$$

como a sua versão em coordenadas homogêneas com o valor particular  $h = 1$ , e o valor  $x_0 = 1$  fixo. Para  $h = 1$ , e, em geral, qualquer valor de  $h$ , estamos definindo um subespaço, neste caso

um plano em 3-D, com a primeira componente fixada como  $x_0 = 1$ , onde residem os padrões originais 2-D, agora com uma componente adicional  $x_0 = 1$ . Embora seja um subespaço bidimensional, ainda assim está definido em 3-D. Podemos dizer que este plano abriga uma cópia de cada flor, originalmente residentes em um espaço 2-D. Na figura 2.11 o sistema de coordenadas 3-D definido pelos eixos  $(h \cdot x_0, h \cdot x_1, h \cdot x_2)$  com a sua origem  $(0,0,0)$  é ilustrado. O vetor  $\mathbf{w} = [b \ w_1 \ w_2]^T = [-18.25 \ 2.05 \ 4.94]^T$  é o resultado de um algoritmo de aprendizagem linear, por exemplo, do método determinístico da Máquina Linear. O conjunto de dados são os 100 flores das classes versicolor e virginica. Duas características foram usadas,  $x_3$  e  $x_4$ , que assumem o papel das coordenadas  $x_1$  e  $x_2$  na figura. O vetor aprendido, na sua versão unitária  $\mathbf{w}/\|\mathbf{w}\|$  é mostrado. Para aumentar o contraste, ele é mostrado multiplicado por 4 na seta pontilhada amarela. Esse vetor é o vetor normal de um hiperplano separador em 3-D, representado pelo plano em verde claro. Esse plano passa pela origem, pois não existe um viés em 3-D, porque só temos três parâmetros. Isto é facilmente verificável, pois  $\mathbf{w} \cdot [0 \ 0 \ 0]^T = 0$  confirma que a origem faz parte do hiperplano. Para o valor  $h = 1$  o subespaço é mostrado em forma do plano em vermelho claro, definido pelas coordenadas  $(1, x_1, x_2)$ . O vetor normal deste plano é  $\tilde{\mathbf{w}} = [1 \ 0 \ 0]^T$ , ou seja, o eixo  $x_0$ , e o viés é  $\tilde{b} = -1$ , pois qualquer ponto  $\tilde{\mathbf{x}} = [1 \ x_1 \ x_2]^T$  faz parte deste plano pela definição 2.4.1 na pág. 67, dado que  $\tilde{\mathbf{w}} \cdot \tilde{\mathbf{x}} + \tilde{b} = 0$ .

Um hiperplano separador entre duas classes neste subespaço 2-D é uma reta. Esta reta é definida pela interseção do plano 3-D e do subespaço 2-D, representado pela reta de cor magenta<sup>10</sup>. O vetor normal do hiperplano separador em 2-D, neste caso uma reta é definido pela segunda e terceira componente do vetor 3-D, ou seja  $[w_1 \ w_2]^T = [2.05 \ 4.94]^T$ , representada pela seta de cor laranja. Essa interseção *não* necessariamente passa pela origem  $(0,0)$  do subespaço 2-D que em coordenadas homogêneas é  $(1,0,0)$ , ou seja, conseguimos uma reta que não tem que passar pela origem em 2-D, capaz de separar as duas classes das flores.

Para um valor cada vez menor do escalar

$$h \rightarrow 0,$$

qualquer ponto em coordenadas homogêneas “implode” para a origem, neste caso a coordenada  $(0,0,0)$ . Este fato vale inclusive para todos os pontos que fazem parte do hiperplano separador em 2-D. A consequência disso é que podemos definir um hiperplano em 3-D que passa pela origem, e, ainda assim é capaz de separar virginica de versicolor.

Podemos ainda constatar que a definição da dicotomia das duas classes, ou seja, do hiperplano separador, não muda no sistema de coordenadas homogêneas, para qualquer valor arbitrário  $h \neq 0$ .

**Corolário 2.4.4** Seja  $\mathbf{x} = [x_1 \ \dots \ x_m]^T$  um padrão definido no espaço vetorial  $\mathbb{R}^m$ , e sejam  $\mathbf{w} = [w_1 \ \dots \ w_m]^T$  o vetor de pesos de dimensão  $m$ , e  $b \in \mathbb{R}$  o viés que definem o hiperplano separador

$$d(\mathbf{x}; \mathbf{w}, b) = \mathbf{w} \cdot \mathbf{x} + b = w_1 x_1 + \dots + w_m x_m + b = 0,$$

da definição 2.4.1 na pág. 67. O hiperplano separador, e, consequente o resultado da classifica-

<sup>10</sup>A equação  $\mathbf{w} \cdot \mathbf{x} + b = 0$  da interseção dos dois planos pode ser facilmente obtida, igualando a definição dos dois planos. O plano verde claro que passa pela origem, ou seja, não tem viés, tem a equação  $\mathbf{w} \cdot \mathbf{x} = [b \ w_1 \ w_2] \cdot [x_0 \ x_1 \ x_2] = 0$ . O plano vermelho claro tem a equação  $\tilde{\mathbf{w}} \cdot \tilde{\mathbf{x}} + \tilde{b} = 0$ . Igualando estas duas equações resulta exatamente na definição do hiperplano separador  $w_1 x_1 + w_2 x_2 + b = 0$  no subespaço 2-D.

ção é invariante para qualquer valor  $h \neq 0$ , representando o padrão  $\mathbf{x}$  na sua forma aumentada

$$\tilde{\mathbf{x}} = [h \cdot 1 \quad h \cdot x_1 \quad \cdots \quad h \cdot x_m]^T$$

e o vetor de pesos na sua forma aumentada

$$\tilde{\mathbf{w}} = [b \quad w_1 \quad \cdots \quad w_m]^T$$

no espaço das coordenadas homogêneas de dimensão  $(m + 1)$ .

*Demonstração.* O hiperplano nas coordenadas aumentados é definido por

$$\begin{aligned} \tilde{\mathbf{w}} \cdot \tilde{\mathbf{x}} &= 0 \\ \iff b h + w_1 h x_1 + \cdots + w_m h x_m &= 0 \\ \iff h [w_1 x_1 + \cdots + w_m x_m + b] &= 0 \\ \iff w_1 x_1 + \cdots + w_m x_m + b &= 0 \\ \iff \mathbf{w} \cdot \mathbf{x} + b &= d(\mathbf{x}; \mathbf{w}, b) = 0 \end{aligned}$$

■

A figura 2.12 explica novamente o hiperplano separador no espaço vetorial de dimensão  $(m + 1)$ , aqui  $m = 2$ , definido pelos  $(m + 1) = 3$  parâmetros  $b = w_0, w_1, \dots, w_2$ . O hiperplano (em verde claro) passa pela origem  $(0, 0, 0)$  do sistema de coordenadas homogêneas. Tem três subespaços 2-D (em vermelho claro), definidos pelos valores  $h = 1$ ,  $h = 2$  e  $h = 3$ . A interseção do hiperplano 3-D com cada um dos subespaços define o hiperplano separador 2-D pelos parâmetros  $b, w_1, w_2$ . Em cima de cada subespaço, os padrões das duas classes são projetados. Fica evidente que a separação das classes é idêntica para qualquer valor  $h \neq 0$ , provada no corolário 2.4.4.

#### 2.4.5 Consideração Crítica da Rotulação do Classificador Linear Binário

No contexto da aprendizagem supervisionada, a rotulação da classe positiva  $y_+ = 1$  e negativa  $y_- = -1$  pode ser chamado de informação falsa, se a classificação for baseada no *score*. Essa afirmação radical é justificável se analisarmos a definição do *score* do classificador linear binário. O teorema 2.4.3 e a rotulação se contradizem. Por um lado, o teorema diz que o score do padrão é uma distância, lembrando que se trata da distância definida na (eq. 2.90) na pág. 70. Por outro lado, o supervisor, o professor na aprendizagem supervisionada, diz que essa distância tem que ser sempre 1 ou  $-1$ . Virtualmente, todos os padrões são projetados em hiperplanos paralelos ao hiperplano separador. Na figura 2.10a, os dois hiperplanos  $d(\mathbf{x}; \mathbf{w}, b) \pm 1$  paralelos ao hiperplano separador  $d(\mathbf{x}; \mathbf{w}, b) = 0$  seriam a verdadeira localização de um padrão que tem um *score*  $y = \pm 1$ . A consequência é que o classificador aprende um hiperplano com padrões todos situados em cima dos dois hiperplanos paralelos. Obviamente essa limitação deste modelo de classificador linear binário destrói toda a informação adicional que está embutida na relação mútua dos padrões. Existem padrões que estão longe da fronteira entre as duas classes, e existem padrões que estão perto da fronteira. Todos são tratados da mesma maneira, ou seja, a tarefa é aprender os pesos e o bias de um classificador que assume que todos tenham a mesma distância, nomeadamente um, ou menos um. Em um problema de classificação, os padrões mais críticos são aqueles que ficam perto da fronteira, ou até invadem a região da outra classe. Então faz sentido atribuir uma importância maior aos padrões em zonas menos “confortáveis”.

Um modelo de classificador que eliminou esse tratamento indiscriminado da saída desejada  $y = \pm 1$  é a Máquina de Vetor de Suporte, *Support Vector Machine* (SVM). A figura 2.10b mostra

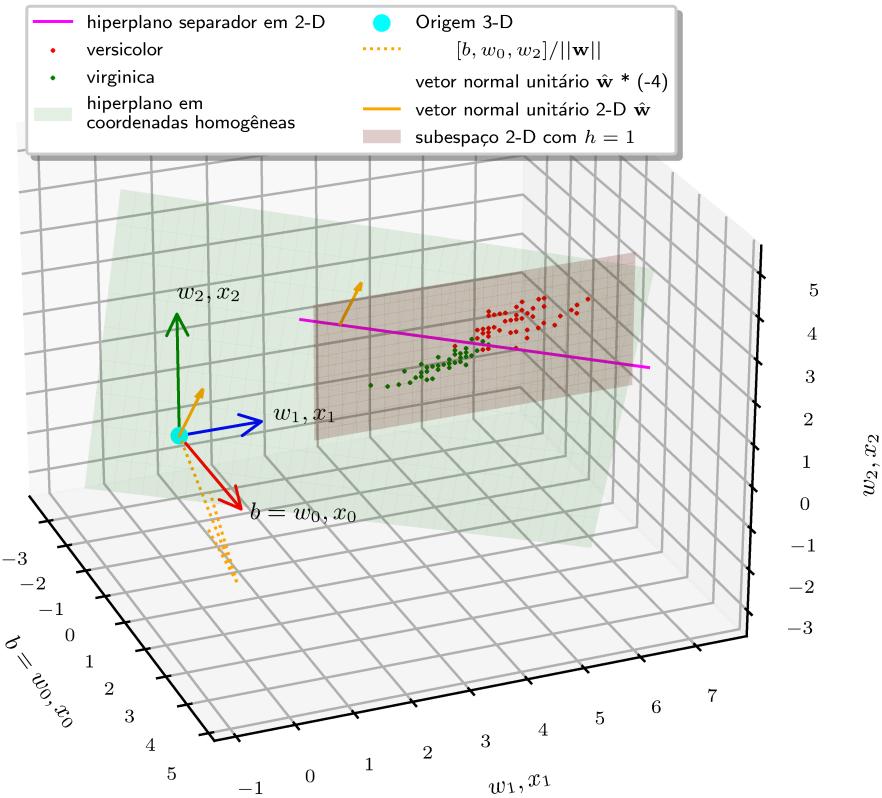


Figura 2.11: Ilustração da necessidade de ter uma dimensão adicional, em relação ao espaço de características. Temos padrões de dimensão  $d = 2$ , mas precisamos  $(d + 1) = 3$  parâmetros, nomeadamente  $b, w_1, w_2$  para formar um hiperplano separador que não passa pela origem em duas dimensões.

o resultado da aprendizagem deste modelo. Para poder visualizar a SVM diretamente no espaço de características, tem que ser usado um *kernel* linear e duas características. O modelo da SVM será estudado posteriormente, no momento consideramos somente o resultado neste caso fácil de separar duas classes de flores. Desta vez vamos usar setosa e versicolor, duas classes linearmente separáveis, considerando o conjunto de treinamento de 100 flores, os padrões  $\mathbf{x}_1$  a  $\mathbf{x}_{50}$  e  $\mathbf{x}_{101}$  a  $\mathbf{x}_{150}$ . Podemos traçar um hiperplano no meio das duas classes. Novamente, por regressão linear, aprendemos o vetor de pesos

$$\mathbf{w} = [b \quad w_1 \quad w_2]^\top = [-1.501 \quad 0.291 \quad 0.422]^\top,$$

a SVM aprendeu (estudaremos mais tarde, como)

$$\mathbf{w}_{\text{SVM}} = [b \quad w_1 \quad w_2]^\top = [-2.292 \quad 0.615 \quad 0.308]^\top.$$

A diferença principal entre regressão linear e SVM se manifesta na definição do vetor aprendido  $\mathbf{w}$ . Na regressão linear, os pesos são obtidos a partir da pseudoinversa  $\tilde{X}^\dagger$ , envolvendo *todos* os padrões do conjunto de treinamento. A SVM considera exclusivamente alguns dos padrões do conjunto de treinamento, os Vetores de Suporte (SV). No exemplo concreto, cada classe é representada por um único padrão SV, setosa por  $\mathbf{x}_{45} = [1.9 \quad 0.4]^\top$  e versicolor por  $\mathbf{x}_{107} = [4.5 \quad 1.7]^\top$ . Eles

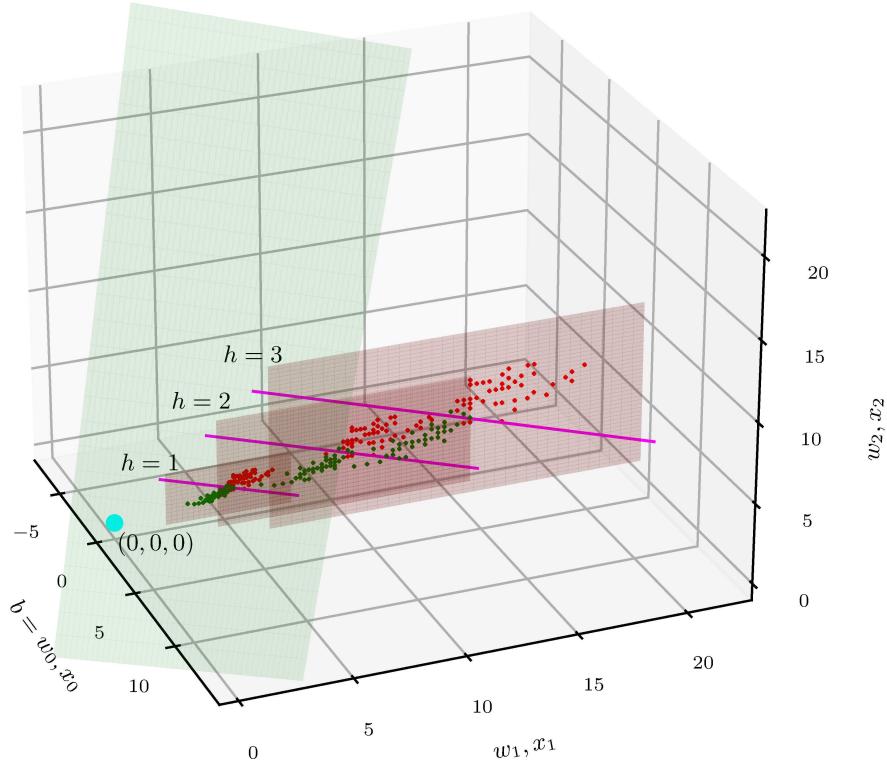


Figura 2.12: Variação do escalar  $h$  do sistema de coordenadas homogêneas. Hiperplano separador em 2-D e padrões ilustrados em cada subespaço.

são marcados na figura 2.10b como marcadores de tamanho maior. Claramente dá para ver que são os padrões mais próximos do hiperplano. E o SV da classe negativa (setosa) fica exatamente em cima do hiperplano paralelo, com uma distância  $d = -1$ , e o SV da classe positiva (virginica) fica exatamente em cima do hiperplano paralelo, com uma distância  $d = +1$ . O espaço entre esses dois hiperplanos paralelos é a *margem* que tem uma largura de  $|d_+ - d_-| = 2$ , lembrando novamente que se trata da distância não normalizada, definida na (eq. 2.90) na pág. 70. Uma propriedade da SVM é que essa margem é maximizada pelo algoritmo de aprendizado da SVM. Não é possível achar outros padrões opostos das duas classes, que uma vez declarados como SV, geram uma margem maior. Todos os outros 49 padrões de cada uma das duas classes são ignorados na definição do hiperplano, assim atribuindo uma importância extrema ao dois SVs que ficam próximos à fronteira. No caso do *kernel* linear o *score* é calculado da mesma maneira como o da regressão linear, segundo a definição 2.4.3. O vetor de pesos da SVM é uma combinação linear de todos os  $\ell$  SV presentes

$$\mathbf{w}_{\text{SVM}} = \sum_{i=1}^{\ell} y_i \alpha_i \mathbf{x}_i, \quad (2.93)$$

onde  $y_i \in \{-1, 1\}$  é o rótulo do SV,  $\alpha_i$  é o *multiplicador de Lagrange* do SV e  $\mathbf{x}_i$  é o  $i$ -ésimo SV. No caso concreto da figura 2.10b temos  $\ell = 2$ ,  $b_{\text{SVM}} = -2.292$ , e  $\mathbf{w}_{\text{SVM}} = -1 \cdot 0.237 \cdot [1.9 \ 0.4]^T + 1 \cdot 0.237 \cdot [4.5 \ 1.7]^T = [0.615 \ 0.308]^T$ . No caso geral em que o *kernel* da SVM é não linear,

será impossível representar explicitamente  $\mathbf{w}_{\text{SVM}}$ , porém a SVM não deixa de ser um simples classificador linear binário. A diferença em relação ao *kernel* linear é que o espaço de características é elevado a um novo espaço, em geral inacessível diretamente. A teoria estudaremos posteriormente.

Muitos dos modelos de regressão e classificação mais complexos que seguirão, compartilham a parte linear aqui ensinada. Isso significa que podemos reutilizar a teoria linear. Por exemplo, o treinamento da última camada de uma *Extreme Learning Machine* pode ser feito por regressão linear múltipla, definida no algoritmo 1. Vamos agora abordar o próximo tema essencial para aprendizado de máquina, a descida de gradiente.

## 3. Descida de Gradiente: Otimização de Funções

### 3.1 Procurando o Mínimo

Imagine-se com os olhos vedados em um terreno montanhoso, totalmente desconhecido, sem qualquer ajuda externa, com a missão de achar o ponto mais alto do terreno. Se você chegou a um ponto desses e não sai mais deste ponto, parabéns, já entendeu o princípio da *subida* de gradiente. Você sentiu com o seu pé, ou seja em uma vizinhança muito limitada da sua posição atual, para onde o caminho sobe, e para onde ele desce. Então, sentindo a direção da subida, você se deslocou um pouco. Depois você repete o procedimento novamente, sentir, subir, sentir, subir, até que chegou em uma posição onde era impossível achar qualquer direção de subida. Chegou no topo de uma montanha. O problema é que você não sabe, se chegou na montanha mais alta do terreno todo, ou seja, sabe que chegou em um máximo local, mas pode existir algo melhor, um máximo global. Às vezes, você teve sorte, e o terreno tinha uma forma convexa, tipo o pão de açúcar que somente tem um único cume. Nesse caso, a sua estratégia de buscar o topo mais alto garantidamente leva ao sucesso. A missão poderia ser também achar o ponto mais baixo do terreno, então seria a *descida* de gradiente. Estamos interessados em minimizar funções, aquela função de perda que mede quanto perto estamos do nosso objetivo. Neste exemplo, a dimensão do argumento da função é dois, a posição  $(x, y)$  no espaço do sistema de coordenadas cartesiano de três dimensões. A coordenada  $z$  é o valor da função na posição atual, ou seja, a altura do terreno nessa coordenada  $z = f(x, y)$ .

A figura 3.1 simboliza o conhecimento disponível para aplicar um método para achar o mínimo de uma função. No contexto de problemas de otimização, a função é geralmente chamada de *função objetivo*. Esta função é desconhecida. “Desconhecido” neste contexto significa, em geral, altamente não linear, não permitindo calcular um mínimo analiticamente. A função é conhecida no sentido que sempre é possível calcular o valor da função, se um argumento for dado como entrada.

Repare que em geral estamos falando sempre de duas funções. A função que mapeia a entrada para a saída e a função objetivo. No contexto da regressão linear, a função de mapeamento calculava  $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$  e a função objetivo era a função de perda  $L(\mathbf{w}, b)$ . Se a minimização da função de perda for baseada nos quadrados das diferenças, ela automaticamente já é uma função quadrática. Em geral, a obtenção analítica da função objetivo somente é possível, se a função é

quadrática. Já vimos na seção 2.1.3 na pág. 46 que é possível obter o mínimo de uma função quadrática, zerando o seu gradiente.

Como a teoria de otimização costuma usar a letra “f” para denominar a função, e a letra “x” para denominar o argumento, vamos novamente seguir esta nomenclatura, porém usaremos o **x** negrito para expressar a multidimensionalidade do argumento. Novamente, não confunda esta função a ser minimizada com a função que calcula, a partir de uma entrada, a saída em um sistema de aprendizado de máquina, por exemplo o mapeamento que a Máquina Linear calcula.

Portanto, para uma função altamente não linear, não é possível obter o mínimo da função  $\mathbf{x}^* = \arg \min_{\mathbf{x}} f(\mathbf{x})$  por uma expressão em forma fechada  $\mathbf{x}^* = \dots$ . Além disso, a função tem um argumento **x** de uma dimensão muito alta. Um exemplo é uma rede neural profunda. No contexto da aprendizagem supervisionada, a função a ser minimizada é a função de perda, e o argumento da função une todos os parâmetros livres da rede, os seus pesos **w**. Dependendo do problema, a dimensão do argumento pode chegar na casa dos centenas de milhares, ou mais.



Figura 3.1: A única informação disponível sobre a função a ser minimizada é o valor da função em um único ponto, e se a função está crescendo ou descendo em uma vizinhança pequena deste ponto. O peitoril da janela representa o eixo  $x$ , o domínio dos parâmetros livres da função objetivo. A parte visível da função é um único argumento  $x$ . É possível calcular o valor, e a derivada da função neste ponto particular. Na figura, a parte visível é composta por mais que um ponto particular (a curva vermelha) para ilustrar o seu comportamento na vizinhança deste ponto.

Na analogia da cortina, o acesso à função é possível para um valor concreto  $x_k$ . A cortina simboliza a nossa ignorância sobre a função, pois fora dessa pequena brecha do argumento  $x_k$ , a função é inacessível, a não ser que novamente a cortina se abre para um outro argumento concreto. Usamos o índice “k” para denominar o instante em que estamos no contexto de um algoritmo iterativo. Em cada passo  $0, 1, \dots, k-1, k, k+1, \dots$  de um algoritmo iterativo, temos o valor concreto  $x_k$  como argumento da função, temos o valor da função neste ponto concreto  $f(x_k)$ , e mais importante ainda, temos a valor da primeira derivada da função neste ponto concreto  $f'(x_k)$ , ou seja, a cortina se abre em um novo intervalo pequeno, quando fornecemos um argumento concreto  $x_k$ .

Na imagem, a função  $f(x)$  tem somente uma única dimensão, porém o eixo horizontal  $x$  simboliza o espaço vetorial multidimensional do domínio  $\mathcal{X}$ , onde residem os argumentos **x**. O peitoril seria o eixo  $x$ , e o valor concreto  $x_k$  podemos colocar, por exemplo, no meio do intervalo, onde a função (em vermelho) é visível. O valor da função  $f(x_k)$  seria então o valor encontrado na curva vermelha. A informação mais importante do que o próprio valor da função são os valores da função em uma vizinhança muito pequena. Nesse exemplo unidimensional vamos escolher o vizinho da direita  $x_k + h$ , para um valor muito pequeno  $h$ . Junto com o valor da função  $f(x_k + h)$  nesse vizinho, já temos tudo para uma aproximação da primeira derivada em  $x_k$ , ou seja

$$f'(x_k) \approx \frac{f(x_k + h) - f(x_k)}{x_k + h - x_k} = \frac{f(x_k + h) - f(x_k)}{h}. \quad (3.1)$$

Esta aproximação é equivalente à inclinação da tangente no ponto  $x_k$ , e equivalentemente pode ser

obtida pelo *polinômio de Taylor* de primeira ordem. O limite  $h \rightarrow 0$  é exatamente a definição da derivada de uma função no ponto em questão  $x_k$

$$f'(x_k) = \lim_{h \rightarrow 0} \frac{f(x_k + h) - f(x_k)}{h}. \quad (3.2)$$

A definição da derivada é muito importante para entender os conceitos de otimização. Portanto, novamente, se não se lembrar desses fundamentos de cálculo, está na hora de recuperá-las. A única informação que nos interessa é, se a função vai crescer, se o argumento  $x_k$  se desloca para a direita  $x_k + h$ , ou se o valor da função vai diminuir. Se crescer, estamos no caminho errado, se quisermos minimizar a função. Se diminuir, estamos no caminho certo. Se crescer, o sinal da aproximação da derivada na (eq. 3.1), ou da própria derivada (eq. 3.2) é positivo, se diminuir, é negativo. Não é necessária nenhuma informação adicional para traçar uma técnica que visa diminuir o valor da função, modificando seu argumento.

### 3.1.1 Estratégia Básica de Descida de Gradiente em uma Dimensão

Podemos então formular uma estratégia básica para diminuir o valor da função, dando pequenos passos. A função  $f$  tem um argumento unidimensional, ou seja,  $x \in \mathbb{R}$ .

**Definição 3.1.1 — Estratégia de Minimização de Função, Uma Dimensão.** Seja  $f(x)$  alguma função. O valor da função pode ser calculado para um argumento concreto  $x_k$ , ou seja,  $f(x_k)$  está acessível. Além do valor da função, a derivada da função para o mesmo valor concreto pode ser calculada, ou seja  $f'(x_k) = \left. \frac{\partial f(x)}{\partial x} \right|_{x_k}$  está acessível. Então a estratégia para minimizar a função é

- Aumente  $x_k$ , se  $f'(x_k)$  for negativo,
- Diminua  $x_k$ , se  $f'(x_k)$  for positivo.

É importante lembar que as mudanças do valor de  $x_k$  acontecem exclusivamente no eixo  $x$ , na analogia da cortina, no peitoril.

■ **Exemplo 3.1** Voltando a figura 3.1, vamos imaginar que no meio do peitoril seja o valor  $x_k = 1.0$ . O valor<sup>1</sup> da função é  $f(x_k) = 1.84$  e o valor da deriva da função é  $f'(x_k) = 1.92$ . A tangente colocaada neste ponto na função teria aproximadamente o dobro da inclinação da diagonal do sistema de coordenadas . A estratégia para minimizar a função manda diminuir o valor atual, ou seja, no próximo passo  $k + 1$  temos um valor concreto novo menor que o anterior,  $x_{k+1} < x_k$ . ■

Podemos então definir uma *direção*, para onde o valor atual  $x_k$  deve-se dirigir. Existem duas maneiras de definir a direção de descida, ou o valor negativo da derivada, ou simplesmente o sinal negativo da derivada. O valor negativo da derivada é a *direção não normalizada*, ela em geral pode ter um módulo diferente do valor um, ou seja  $| -f'(x_k) | \neq 1$ . O sinal negativo da derivada é a *direção normalizada*, ela sempre tem módulo um.

**Definição 3.1.2 — Direção de Descida, Uma Dimensão.** Seja  $f'(x_k)$  o valor da derivada de um valor concreto  $x_k$ . Então a direção da minimização da função é

$$p_k \stackrel{\text{def}}{=} -f'(x_k) \quad \text{Direção de descida não normalizada} \quad (3.3a)$$

$$p_k \stackrel{\text{def}}{=} -\text{sgn}(f'(x_k)) = \mp 1 \quad \text{Direção de descida normalizada} \quad (3.3b)$$

<sup>1</sup>Obviamente, o autor destas linhas conhece a função que deu origem à figura, por isso é possível fornecer os valores, porém para o leitor a função tem que ter um caráter não revelado.

■ **Exemplo 3.2** No exemplo anterior temos para a direção da descida o valor  $p = -1.92$  no caso não normalizado, e  $p = -\text{sgn}(1.92) = -1$  no caso normalizado. Em ambos os casos, a estratégia é “Vá para a esquerda”, para se direcionar a um valor da função menor do que atualmente. Repare que neste caso unidimensional, direções são escalares  $p \in \mathbb{R}$ , no caso multidimensional, onde a função tem um vetor  $\mathbf{x} \in \mathbb{R}^m$  como argumento, a direção será um vetor  $\mathbf{p} \in \mathbb{R}^m$  da mesma dimensão  $m \geq 2$ . ■

Vamos somente considerar a direção não normalizada  $p_k = -f'(x_k)$  para formalizar um único passo na descida de gradiente. A questão em aberto é o tamanho do passo a ser dado, ou seja definir valor da diferença “delta”, entre o valor atual do parâmetro  $x_k$  e o próximo valor  $x_{k+1}$  como

$$\Delta x_k \stackrel{\text{def}}{=} x_{k+1} - x_k. \quad (3.4)$$

Se olharmos novamente o exemplo da figura 3.1, percebe-se que do lado esquerdo, portanto, na direção da descida, a função diminui. Porém, logo na margem esquerda da cortina, onde começa o domínio ainda desconhecido, a função parece subir de novo. Então, se o passo a ser dado for grande demais, corremos o risco de potencialmente perder um bom candidato a um mínimo. Se o passo a ser dado for curto demais, o processo de minimização da função é lento. Fomos obrigados a quantificar o tamanho do passo a ser dado para chegar do valor atual ao valor atualizado.

**Definição 3.1.3 — Taxa de Aprendizagem.** O escalar  $\alpha > 0$  (“alfa”) é o multiplicador da direção de descida de gradiente, assim controlando o tamanho do passo entre o valor atual e próximo do argumento da função objetivo. Frequentemente, usa-se a letra grega  $\eta$  (“eta”) para denominar a taxa de aprendizagem.

**Definição 3.1.4 — Descida de gradiente, Uma Dimensão.** Seja  $f'(x_k)$  o valor da derivada de uma função, dado um valor concreto  $x_k$ . Então o próximo valor  $x_{k+1}$  é obtido, somando a direção de descida  $p_k$ , ponderada pela taxa de aprendizagem. Assim

$$x_{k+1} \leftarrow x_k + \alpha [-f'(x_k)]. \quad (3.5)$$

O sinal da derivada de propósito não foi incorporado na conta para destacar o fato que a derivada negativa (oposta) é somada ao valor atual, porém na literatura, a regra aparece equivalente como

$$x_{k+1} \leftarrow x_k - \alpha f'(x_k). \quad (3.6)$$

Repare na definição 3.1.4 o símbolo da atribuição “ $\leftarrow$ ”, em vez do símbolo da igualdade “ $=$ ”. A intenção é frisar o fato que o novo valor é obtido em um contexto de um algoritmo, substituindo o valor atual pelo novo<sup>2</sup>.

■ **Exemplo 3.3** No exemplo da cortina, usando um valor típico  $\alpha = 0.2$ , com o valor inicial  $x_k = 1.0$  e o valor da derivada  $f'(x_k) = 1.92$ , o próximo valor obtido pela descida de gradiente, aplicando a (eq. 3.6), temos  $x_{k+1} \leftarrow x_k - \alpha f'(x_k) = 1.0 - 0.2 \cdot 1.92 = 0.616$ . Aplicando novamente a regra, temos  $x_{k+2} \leftarrow x_{k+1} - \alpha \cdot f'(x_{k+1}) = 0.616 - 0.2 \cdot 1.07 = 0.400$ . Podemos observar que a descida de gradiente desloca o argumento  $x$  duas vezes para esquerda. ■

Com a taxa de aprendizagem  $\alpha$  encontramos um novo animal na fauna de aprendizado de máquina. É um *hiperparâmetro*. Esses valores definem a estrutura de um modelo, ou como um modelo é treinado. Hiperparâmetros são diferentes dos parâmetros comuns de um modelo, por

<sup>2</sup>Na maioria das linguagens de programação, a atribuição é expressa pelo símbolo da igualdade, por exemplo no comando  $i = i + 1$ . Se fosse uma equação, subtraindo  $i$  dos dois lados levaria a uma contradição  $0 = 1$ . Na verdade o valor da variável  $i$  é atualizado pelo comando, incrementando o valor em um.

#	Condição	Expressão	Hiperparâmetro
1	Número máximo de iterações atingido	$k > k_{\max}$	$k_{\max}$
2	Tempo de execução ultrapassado	$t > t_{\max}$	$t_{\max}$
3	Derivada pequena	$ f'(x_k)  < g_{\min}$	$g_{\min}$
4	Valores consecutivos parecidos	$ x_{k+1} - x_k  < \Delta x_{\min}$	$\Delta x_{\min}$
5	Função objetivo pequena	$f(\dots) < f_{\min}$	$f_{\min}$
6	Divergência, valor alto	$ x_k  >= x_{\max}$	$x_{\max}$
7	Divergência, valor função alto	$ f(x_k)  >= f_{\max}$	$f_{\max}$
8	Sobreajuste ( <i>overfitting</i> )	a ser definido	a ser definido

Tabela 3.1: Critérios de parada de um algoritmo iterativo de otimização.

exemplo dos pesos de uma rede neural. Vamos conhecer um outro exemplo de um hiperparâmetro no estudo de redes neurais com mais que zero camadas ocultas. A quantidade de neurônios em camadas ocultas é um hiperparâmetro típico. Ou define-se manualmente essa quantidade, ou tenta-se obter automaticamente a quantidade mais adequada. A determinação automática chama-se *tuning*. Posteriormente estudaremos mais esse assunto.

### 3.1.2 Algoritmo Iterativo de Otimização

A busca pelo mínimo da função  $f$  em geral é um algoritmo iterativo que é inicializado com um valor  $x_0$  no instante  $k = 0$ . Na primeira iteração  $k = 1$ , o algoritmo calcula o valor  $x_1$ , supostamente melhor, somando a diferença  $\Delta x_0$  do instante  $k = 0$ . Na  $k$ -ésima iteração, o valor atual é  $x_k$ . O algoritmo termina, se uma das *condições de parada* for satisfeita. Condições de parada típicas em um algoritmo interativo são listados na tabela 3.1. Cada condição em geral tem que definir um ou mais hiperparâmetros que controlam a execução do algoritmo. Nem sempre todos esses critérios são usados. No caso mais simples, pode existir uma única condição, por exemplo, executar o algoritmo um número fixo de vezes  $k_{\max}$ . Assim, o algoritmo incondicionalmente termina após  $k_{\max}$  iterações. Podemos até considerar o valor inicial  $x_0$  como um hiperparâmetro, pois ele influencia o resultado da otimização.

Então, podemos formalizar a moldura básica de um algoritmo iterativo de otimização no algoritmo 2.

---

#### Algoritmo 2: Otimização Iterativa

---

**Entrada:** Dados, Hiperparâmetros

**Resultado:** Variável  $\mathbf{x}^*$  que supostamente minimiza função objetivo

- 1 Inicialize na iteração  $k = 0$  a variável  $\mathbf{x}$ , com o valor  $\mathbf{x}_0$ ;
  - 2 **enquanto** ainda não satisfizer um dos critérios de parada **faça**
  - 3   | calcula a próxima solução  $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \Delta \mathbf{x}_k$ ;
  - 4   |    $k \leftarrow k + 1$ ;
  - 5 **fim**
- 

Vamos especializar o algoritmo 2 para o caso da descida de gradiente de uma função com um argumento unidimensional no algoritmo 3. Além disso, usaremos apenas as condições de parada “Número máximo de iterações atingido” e “Derivada pequena”. As condições de parada são unidas em uma conjunção lógica (letra “E”). Se uma das condições for violada, o algoritmo pára.

No contexto de aprendizado de máquina, a entrada de dados no algoritmo indiretamente define a função, por exemplo, se a função for uma função de perda, o conjunto de treinamento faz parte da definição de  $f$ . A inicialização do argumento como  $x_0$  tem uma influência grande sobre

**Algoritmo 3:** Descida de Gradiente, Uma Dimensão

**Entrada :** Dados; Função a ser minimizada  $f(x)$ ; Derivada  $f'(x)$  da função; Taxa de aprendizagem  $\alpha$ ; Número máximo de iterações  $k_{\max}$ ; Limiar de módulo do valor da derivada  $g_{\min}$

**Resultado:** Variável  $x^*$  que supostamente minimiza função objetivo

- 1 Inicialize na iteração  $k = 0$  a variável  $x$ , com o valor  $x_0$ ;
- 2 **enquanto** ( $k < k_{\max}$ ) E ( $|f'(x_k)| > g_{\min}$ ) **faça**
- 3     $x_{k+1} \leftarrow x_k - \alpha f'(x_k)$  ;
- 4     $k \leftarrow k + 1$ ;
- 5 **fim**
- 6  $x^* \leftarrow x_k$

o resultado final, se a função for altamente não linear. A taxa de aprendizagem também vai influenciar fortemente o comportamento do algoritmo. Se  $\alpha$  for grande demais, eventualmente haverá oscilações fortes no valor de  $x$ , se for pequeno demais, atrasará a busca pelo valor ótimo.

Vamos por um momento levantar a cortina e revelar a função  $f(x)$  atrás em sua especificação em forma fechada, analiticamente, por razões didáticas. Lembre que, em um cenário realista, a função a ser otimizada tem um comportamento de uma caixa preta. Para uma entrada, fornece uma saída, para outra entrada, fornece outra saída. Porém, a função não permite obter o mínimo em forma fechada, devido a sua natureza altamente não linear. Até funções unidimensionais frequentemente não permitem encontrar soluções deterministicamente. Por exemplo, a busca da raiz de uma função, ou seja a solução da equação  $f(x) = 0$ , para certas funções analiticamente definidas tem que ser feita por métodos iterativos. O método de Newton-Raphson[72] é um desses métodos

■ **Exemplo 3.4** Seja a função dada como  $f(x) = x \sin(x^2) + 1$ , ilustrada na figura 3.2. A derivada da função é  $f'(x) = \sin(x^2) + 2x^2 \cos(x^2)$ . Neste ponto estou cobrando o emprego da regra da cadeia da definição 2.1.8 na pág. 42 para calcular corretamente a derivada. O leitor deve reconhecer a parte central da função na figura 3.2 como a parte visível na figura 3.1. Uma primeira análise visual revela que existem sete pontos  $x$  com derivada nula na parte visualizada  $x \in [-2, 4]$  dos quais seis são extremos. Três são máximos, entre  $[1, 2]$ ,  $[2, 3]$  e  $[3, 4]$ , e um ponto de sela em  $x = 0$ . Esses quatro pontos não nos interessam como resultado. Existem três mínimos entre  $[-2, -1]$ ,  $[2, 3]$  e  $[3, 4]$ . Na parte visualizada, o mínimo com o menor valor fica no intervalo  $[3, 4]$ , porém considerando a definição da função para valores  $x \rightarrow \pm\infty$ , temos uma quantidade infinita de mínimos, portanto, não existe um mínimo global. Dessa maneira, o algoritmo da descida de gradiente ficará sempre preso em um mínimo local, ou até no ponto de sela. Tudo depende do valor inicial  $x_0$ , da taxa de aprendizagem  $\alpha$ , e outros hiperparâmetros, como, por exemplo, o número máximo de iterações  $k_{\max}$ . ■

A tabela 3.2 mostra três simulações do algoritmo 3. Foram variados o valor inicial  $x_0$  e a taxa de aprendizagem  $\alpha$ . Com o par de valores ( $\alpha = 0.1, x_0 = -1$ ), o algoritmo convergiu para o valor  $x^* = -1.36$  que corresponde ao mínimo local do intervalo  $[-2, -1]$ . Foi atingida a condição de parada  $|f'(x_k)| < 1e-5$  após  $k = 8$  iterações. Com o par de valores ( $\alpha = 0.7, x_0 = 1.5$ ), o algoritmo divergiu, todos os valores explodiram. Com o par de valores ( $\alpha = 0.2, x_0 = 0.5$ ), o algoritmo oscilou no final entre os valores  $x = 1.628$  e  $x = -1.628$ . Os resultados com esta função unidimensional já destacam a importância de uma boa escolha da taxa de aprendizagem e do valor inicial do parâmetro a ser otimizado. Uma escolha inapropriada pode levar a divergência do algoritmo, oscilação ou iteração lenta.

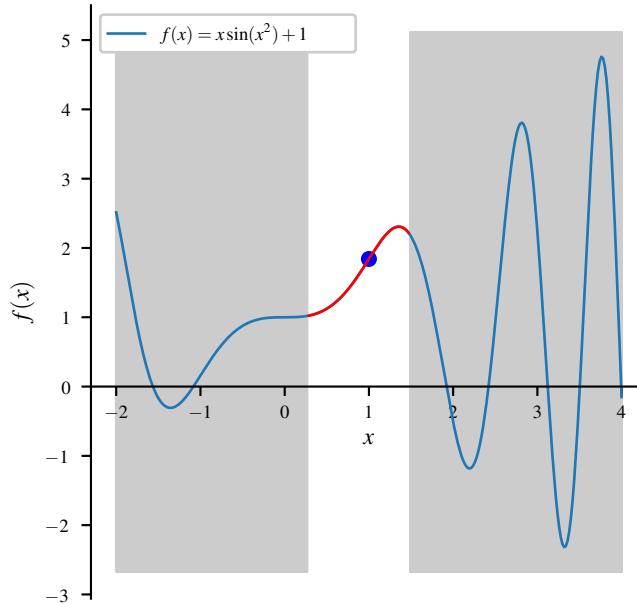


Figura 3.2: Revelação da função

## 3.2 Descida de Gradiente em Função Vetorial

Até o momento estudamos o mecanismo de otimização da descida de gradiente com uma função  $f(x)$  de um argumento escalar  $x \in \mathbb{R}$ . Em aplicações reais de aprendizado de máquina, temos um argumento  $x \in \mathbb{R}^m$ , onde a dimensão  $m$  é geralmente alta. O resultado da função ainda é um escalar, portanto vamos considerar o caso da função escalar  $f : \mathbb{R}^m \rightarrow \mathbb{R}$  e o seu gradiente  $\nabla f \in \mathbb{R}^m$  da definição 2.1.7 na pág. 37. Se quisermos manter a possibilidade de visualizar a função e seu gradiente, não podemos aumentar a dimensão  $m$  acima de dois, ou seja consideramos funções  $f(\mathbf{x}) = f\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right)$  e o seus gradientes  $\nabla f = [\partial f / \partial x_1 \quad \partial f / \partial x_2]$ . Novamente, lembre-se que as funções que devem ser minimizadas são de natureza funções de perda  $L(\mathbf{w})$  com argumentos  $\mathbf{w}$ , porém para manter a consistência com a literatura, a função é chamada  $f$ , e seu argumento  $\mathbf{x}$ .

Conceitualmente nada muda. Para minimizar a função, o parâmetro  $\mathbf{x}_k$  deve-se deslocar na direção negativa do gradiente  $\nabla f$ . Vamos primeiramente estender o conceito da direção para mais que uma dimensão, ignorando a versão normalizada.

**Definição 3.2.1 — Direção de Descida, Função Vetorial.** Seja  $\nabla f(\mathbf{x}_k)$  o valor do gradiente da função objetivo  $f$  para um argumento concreto  $\mathbf{x}_k$ . Então a direção da minimização da função é vetor na direção oposta do gradiente

$$\mathbf{p}_k = -\nabla f(\mathbf{x}_k) \tag{3.7}$$

Esta direção somente é válida para o método mais simples da descida de gradiente, o método de *Máximo Declive* que será estudado na seção 3.3.3 na pág. 99.

Agora o gradiente já não é um escalar, ele é um vetor. Mais uma vez, o deslocamento será ponderado, para evitar um movimento rápido demais, ou lento demais. Portanto, temos que definir

$\alpha = 0.1, x_0 = -1.0$					$\alpha = 0.7, x_0 = 1.5$					$\alpha = 0.2, x_0 = 0.5$				
$k$	$x$	$f(x)$	$f'(x)$	$-\nabla$	$k$	$x$	$f(x)$	$f'(x)$	$-\nabla$	$k$	$x$	$f(x)$	$f'(x)$	$-\nabla$
0	-1	0.159	1.92	$\leftarrow$	0	1.5	2.167	-2.049	$\rightarrow$	0	-0.5	0.876	0.732	$\leftarrow$
1	-1.19	-0.179	1.41	$\leftarrow$	1	2.934	3.137	-11.07	$\rightarrow$	1	-0.646	0.738	1.169	$\leftarrow$
2	-1.33	-0.305	0.247	$\leftarrow$	2	10.683	10.171	117.942	$\leftarrow$	2	-0.88	0.384	1.807	$\leftarrow$
3	-1.36	-0.308	-0.0343	$\rightarrow$	3	-71.876	-69.459	2042.673	$\leftarrow$	3	-1.242	-0.241	1.089	$\leftarrow$
4	-1.35	-0.308	0.00613	$\leftarrow$	4	-1.50E+03	-1.18E+03	2.79E+06	$\leftarrow$	4	-1.46	-0.237	-1.413	$\rightarrow$
5	-1.36	-0.308	-0.00106	$\rightarrow$	5	-1.95E+06	-1.94E+06	8.69E+11	$\leftarrow$	...	...	...	...	...
6	-1.36	-0.308	0.000185	$\leftarrow$	6	-6.08E+11	-2.40E+11	-6.80E+23	$\rightarrow$	98	-1.474	-0.216	-1.628	$\rightarrow$
7	-1.36	-0.308	-3.23E-05	$\rightarrow$	7	4.76E+23	4.70E+23	7.31E+46	$\leftarrow$	99	-1.148	-0.111	1.628	$\leftarrow$
8	-1.36	-0.308	5.63E-06	$\leftarrow$	8	-5.11E+46	3.64E+46	3.68E+93	$\leftarrow$	100	-1.474	-0.216	-1.628	$\rightarrow$

Tabela 3.2: Resultados da execução do algoritmo da descida de gradiente para hiperparâmetros diferentes. A direção do movimento do valor  $x$  está marcado com o símbolo  $-\nabla$ , ou seja, a direção oposta da derivada.

novamente o valor do hiperparâmetro  $\alpha$ , a taxa de aprendizagem. O parâmetro e o gradiente são do mesmo espaço vetorial de dimensão  $m \geq 2$ . Lembre da definição 2.1.7 na pág. 37 que o gradiente é um vetor de linha, mas o parâmetro a ser otimizado é um vetor de coluna. Então temos que transpor o gradiente na regra da descida de gradiente para conservar a consistência do cálculo matricial. Assim podemos somar o múltiplo do gradiente negativo ao valor atual do parâmetro, resultando no algoritmo 4, com a regra de atualização dos parâmetros

$$\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k)^\top. \quad (3.8)$$

O que fazer, se somente o valor da função  $f(\mathbf{x}_k)$  para um valor específico  $\mathbf{x}_k$  está disponível, mas não o seu gradiente  $\nabla f(\mathbf{x}_k)$  neste ponto? Temos que estimar o gradiente. Na (eq. 3.1) já temos um método<sup>3</sup> para estimar a derivada  $f'(x_k)$  em uma dimensão. A ideia mais simples de estender o caso unidimensional para o caso multidimensional é aplicar a estimativa para cada dimensão independentemente da outra. Seja o valor atual dos parâmetros então

$$\mathbf{x}_k = [x_{k,1} \quad \cdots \quad x_{k,j} \quad \cdots \quad x_{k,m}]^\top.$$

A  $j$ -ésima componente  $\partial f(\mathbf{x})/\partial x_j$  do gradiente

$$\nabla f(\mathbf{x}) = [\partial f(\mathbf{x})/\partial x_1 \quad \cdots \quad \partial f(\mathbf{x})/\partial x_j \quad \cdots \quad \partial f(\mathbf{x})/\partial x_m]$$

então é estimada para o ponto atual  $\mathbf{x}_k$  como

$$\frac{\partial f(\mathbf{x}_k)}{\partial x_{k,j}} \approx \frac{f(\mathbf{x}_k + \mathbf{h}_j) - f(\mathbf{x}_k)}{h}, \quad j = 1, \dots, m, \quad (3.9)$$

onde o vetor  $\mathbf{h}_j$  é nulo em todas as componentes, exceto na  $j$ -ésima componente tem o valor  $h$ , ou seja,

$$\mathbf{h}_j = [0 \quad 0 \quad \cdots \quad 0 \quad h \quad 0 \quad \cdots \quad 0]^\top.$$

Esta estimativa ignora completamente a dependência entre as componentes, mas pelo menos permite ao algoritmo de descida de gradiente de obter o vetor do gradiente necessário.

Vamos nos limitar novamente a dois critérios de parada, um número máximo de iterações e um valor mínimo da norma do gradiente.

<sup>3</sup>Uma expressão alternativa para estimar a derivada em uma dimensão é  $f'(x_k) \approx \frac{f(x_k+h) - f(x_k-h)}{x_k+h - (x_k-h)} = \frac{f(x_k+h) - f(x_k-h)}{2h}$ , afastando-se do ponto atual  $x_k$  em direções opostas.

**Algoritmo 4:** Descida de Gradiente, Mais que uma Dimensão

**Entrada:** Dados; Função a ser minimizada  $f(\mathbf{x})$ ; Gradiente  $\nabla f(\mathbf{x})$  da função; Taxa de aprendizagem  $\alpha$ ; Número máximo de iterações  $k_{\max}$ ; Limiar da norma quadrática do valor do gradiente  $g_{\min}$

**Resultado:** Variável  $\mathbf{x}^*$  que supostamente minimiza função objetivo

- 1 Inicialize na iteração  $k = 0$  a variável  $\mathbf{x}$ , com o valor  $\mathbf{x}_0$ ;
- 2 **enquanto** ( $k < k_{\max}$ )  $\mathbf{E} (||\nabla f(\mathbf{x}_k)|| > g_{\min})$  **faça**
- 3      $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k)^T$  ;
- 4      $k \leftarrow k + 1$ ;
- 5 **fim**
- 6  $\mathbf{x}^* \leftarrow \mathbf{x}_k$

Compare com o algoritmo 3 do caso de variável  $x$  unidimensional. A estrutura é idêntica. Em vez de comparar o módulo da derivada  $|f'(x_k)|$  com o limiar, agora é a norma do gradiente  $||\nabla f(\mathbf{x}_k)||$ . O limiar  $g_{\min}$  desempenha o papel de parar o algoritmo, se o comprimento do vetor do gradiente for pequeno demais. Reveja o cálculo da norma de um vetor da (eq. 1.10) na pág. 14.

**3.2.1 Descida de Gradiente Estocástica**

Lembre-se que a função a ser minimizada em um contexto de aprendizagem supervisionada é a função de perda  $L(\mathbf{w})$ , cujo argumento são os parâmetros livres  $\mathbf{w}$ . Então convém, por um momento, mudar a nomenclatura de  $f(\mathbf{x})$  para  $L(\mathbf{w})$ . Isso facilita a terminologia na aplicação da descida de gradiente ao problema da regressão linear. Consequentemente a regra de atualização da (eq. 3.8) com os nomes mudados é

$$\mathbf{w}_{k+1} \leftarrow \mathbf{w}_k - \alpha \nabla L(\mathbf{w}_k)^T. \quad (3.10)$$

A letra “f” é usada para denominar a função que mapeia a entrada  $\mathbf{x}$  para a saída  $\mathbf{y}$ , usando o conjunto de parâmetros atuais  $\mathbf{w}$ , ou seja,  $\mathbf{y} = f(\mathbf{x}; \mathbf{w})$ . Lembre-se que a função de perda  $L(\mathbf{w})$  depende do conjunto de treinamento de  $n$  padrões com entradas  $\mathbf{x}_i$  e saídas  $\mathbf{y}_i$ ,  $i = 1, \dots, n$ . O gradiente  $\nabla L(\mathbf{w}_k)$  é calculado sobre todos os  $n$  padrões na (eq. 3.10). Esta versão da descida de gradiente é denominada como descida de gradiente em lote (*batch gradient descent*).

O outro caso extremo é quando os parâmetros  $\mathbf{w}$  são atualizados no cálculo do gradiente baseado na função de perda de um único par de entrada-saída  $\mathbf{x}_i$  e  $\mathbf{y}_i$ . Esta versão é a descida de gradiente estocástica (*Stochastic Gradient Descent, SGD*), já mencionado na seção 2.1.3 na pág. 35. Quando um número reduzido de padrões é usado para calcular, temos que baralhar a sequência de apresentação ao algoritmo de descida de gradiente. Caso contrário, introduzimos uma ordem indesejada nos dados. Vamos definir então definir o algoritmo 4 para o caso estocástico, usando os nomes da função de perda.

Uma versão intermediária entre descida de gradiente em lote e estocástica é o mini-lote (*mini-batch*), em que um número pequeno  $p \ll n$  de padrões é usado para calcular o gradiente. O gradiente da função de perda é a média dos  $p$  padrões, contidos no mini-lote. A técnica do mini-lote é principalmente usada, se existe um número muito grande de padrões (*Big Data*). A versão puramente estocástica com um padrão é custosa demais, se houver muitos dados, e a descida de gradiente em lote impõe problemas de armazenamento de todos os padrões na memória. Os parâmetros  $\mathbf{w}$  são atualizados no caso do mini-lote baseado na média do gradiente do mini-lote. Vamos formalizar o conceito do conjunto de dados usado no treinamento.

**Algoritmo 5:** Descida de Gradiente Estocástica

**Entrada :** Dados:  $n$  pares de entrada  $\mathbf{x}_i$  e saída  $\mathbf{y}_i$ ,  $i = 1, \dots, n$ ; Função a ser minimizada  $L(\mathbf{w})$ ; Gradiente  $\nabla L(\mathbf{w})$  da função; Taxa de aprendizagem  $\alpha$ ; Função que mapeia a entrada para saída  $\mathbf{y} = f(\mathbf{x})$ ; Número máximo de iterações  $k_{\max}$ ; Limiar da norma quadrática do valor do gradiente  $g_{\min}$

**Resultado:** Variável  $\mathbf{w}^*$  que supostamente minimiza função objetivo

- 1 Inicialize na iteração  $k = 0$  a variável  $\mathbf{w}$ , com o valor  $\mathbf{w}_0$ ;
- 2 **enquanto** ( $k < k_{\max}$ ) E ( $\|\nabla L(\mathbf{w}_k)\| > g_{\min}$ ) **faça**
- 3    Crie uma permutação  $p$  aleatória da sequência  $1, \dots, n$ ;
- 4    **para**  $i=1$  até  $n$  **faca**
- 5      $q \leftarrow p(i)$  /\* Escolhe um índice aleatoriamente \*/;
- 6      $\hat{\mathbf{y}}_q \leftarrow f(\mathbf{x}_q; \mathbf{w}_k)$  /\* Calcule a saída \*/;
- 7      $L_q \leftarrow L(\mathbf{w}_k)$  /\* Perda do padrão \*/;
- 8      $\nabla L_q \leftarrow \partial L_q / \partial \mathbf{w}_k$  /\* Gradiente da perda do padrão \*/;
- 9      $\mathbf{w}_{k+1} \leftarrow \mathbf{w}_k - \alpha \nabla L_q^\top$  /\* Atualização de pesos \*/;
- 10    **fim**
- 11     $\nabla L(\mathbf{w}_k) \leftarrow \frac{1}{n} \sum_{q=1}^n \nabla L_q$  /\* Gradiente médio sobre os  $n$  padrões \*/;
- 12     $k \leftarrow k + 1$ ;
- 13 **fim**
- 14  $\mathbf{w}^* \leftarrow \mathbf{w}_k$

**Definição 3.2.2 — Treinamento Estocástico, Mini-lote e Lote.** Sejam dados  $p \geq 1$  padrões  $\mathbf{x}_i$ ,  $i \in \{1, \dots, n\}$  de um total de  $n$  padrões. Durante uma iteração de um algoritmo de otimização,  $p \leq n$  padrões são apresentados ao algoritmo para calcular a função de perda  $L$  e o seu gradiente  $\nabla L$ . Distinguem-se os seguintes casos

- |               |                            |
|---------------|----------------------------|
| $p = 1$ ,     | Aprendizagem Estocástica   |
| $1 < p < n$ , | Aprendizagem com Mini-lote |
| $p = n$ ,     | Aprendizagem em Lote       |

Para o caso estocástico e mini-lote, a escolha dos padrões deve ser aleatória para evitar a introdução de uma ordem não existente. Para expressar o embaralhamento da ordem dos padrões, é necessária introduzir um nível adicional dos índices, ou seja, ou conjunto de treinamento são os padrões  $\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \dots, \mathbf{x}_{i_j}, \dots, \mathbf{x}_{i_p}$ ,  $i_j \in \{1, \dots, n\}$ . Em um problema de classificação de  $c$  classes, o mini-lote deve tentar conter  $\lfloor p/c \rfloor$  padrões de cada classe<sup>a</sup>, formando um conjunto de treinamento *estratificado*. Frequentemente, na literatura o treinamento por mini-lote também faz parte da aprendizagem estocástica, pois nem todos os  $n$  padrões do conjunto de treinamento são usados simultaneamente na atualização dos parâmetros.

<sup>a</sup>O operador  $\lfloor q \rfloor$ , “piso”, (*floor*) calcula o número inteiro menor ou igual a  $q \in \mathbb{R}$ .

■ **Exemplo 3.5** Seja  $p = 6$  o tamanho do mini-lote no treinamento do conjunto iris, com  $n = 150$ . O gerador de números aleatórios escolheu os seis índices  $i_1 = 138, i_2 = 77, i_3 = 68, i_4 = 9, i_5 = 122, i_6 = 48$ , portanto o mini-lote compõe as flores  $\mathbf{x}_9$  e  $\mathbf{x}_{48}$  de setosa,  $\mathbf{x}_{68}$  e  $\mathbf{x}_{77}$  de versicolor e  $\mathbf{x}_{122}$  e  $\mathbf{x}_{138}$  de virginica. É um conjunto estratificado, pois cada classe é representada por duas flores. ■

### 3.2.2 Regressão Linear

As funções dos problemas lineares que estudamos até agora correspondem a terrenos convexos, mais precisamente<sup>4</sup> a terrenos estritamente convexos, onde existe somente um único mínimo global. Já assumimos a convexidade da função de perda baseada no erro quadrático médio na seção 2.1.3 na pág. 46, veja também a figura 2.3 na pág. 48, além de provar a convexidade das regiões de decisão da Máquina Linear na seção 2.3.2 na pág. 65. Vamos posteriormente tocar nesta propriedade, mas neste ponto afirmamos sem prova que  $L(w, b)$  do modelo linear estudado seja estritamente convexa.

Por que então estamos buscando novos métodos, se já temos o algoritmo 1 na pág. 56 que fornece em um único passo uma solução determinística? A resposta é que a Álgebra Linear infelizmente é uma ferramenta fraca demais para obter um conjunto de parâmetros  $\theta^*$  ótimos para funções não lineares.

Mas o leitor crítico, com espírito de pesquisador, vai exigir que se mostre, que para esses problemas convexos, os resultados batem. Isto quer dizer que o método determinístico, usando uma forma fechada leva ao mesmo resultado que o algoritmo iterativo da descida de gradiente. Isto significa que a solução de uma matriz de pesos  $W = \tilde{X}^\dagger Y$ , fornecida pela (eq. 2.70) na pág. 56, respectivamente o algoritmo 1 na pág. 56, teoricamente teriam que fornecer pelo menos valores muito parecidos, caso usarmos o algoritmo 4 iterativo da descida de gradiente.

Podemos aplicar o algoritmo 4 ao nosso problema elementar da tabela 2.1 na pág. 34 e figura 2.1 na pág. 34 que tem sete pares de entrada-saída. A dimensão de entrada e saída é um, porém temos dois parâmetros, portanto a descida de gradiente bidimensional é aplicável. Lembre que juntamos os dois parâmetros  $w$  e  $b$  em um vetor aumentado de parâmetros  $\tilde{w}$ . Para facilitar o entendimento do cálculo, podemos manter os dois parâmetros separados no algoritmo. Então o primeiro experimento vai obter os dois parâmetros por descida de gradiente estocástica. Lembre que os valores obtidos pela regressão linear simples pela solução determinística no exemplo 2.8 foram  $\tilde{w} = \tilde{X}^\dagger \mathbf{y} = \begin{bmatrix} b \\ w \end{bmatrix} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} 0.325 \\ 0.475 \end{bmatrix}$ .

Vamos especializar o algoritmo 5 para resolver o problema da regressão linear simples. Antes disso, é necessário focar na perda  $L_i(w, b)$  de um único padrão, e o seu gradiente  $\nabla L_i(w, b)$ . A perda individual foi abordada na (eq. 2.36) na pág. 41, e o seu gradiente foi calculado na (eq. 2.48) na pág. 46. A função de perda  $L_i(w, b)$  de um único padrão  $x_i$  e o seu gradiente  $\nabla L_i(w, b)$  foram já calculados na (eq. 2.49) na pág. 46 como

$$\begin{aligned} L_i(w, b) &= (y_i - \hat{y}_i)^2 = (y_i - (wx_i + b))^2 \\ \nabla L_i(w, b) &= \begin{bmatrix} \frac{\partial L_i(w, b)}{\partial w} \\ \frac{\partial L_i(w, b)}{\partial b} \end{bmatrix}^\top = \begin{bmatrix} \frac{\partial (y_i - \hat{y}_i)^2}{\partial w} \\ \frac{\partial (y_i - \hat{y}_i)^2}{\partial b} \end{bmatrix}^\top = \begin{bmatrix} 2(y_i - \hat{y}_i) \frac{\partial (y_i - (wx_i + b))}{\partial w} \\ 2(y_i - \hat{y}_i) \frac{\partial (y_i - (wx_i + b))}{\partial b} \end{bmatrix}^\top = \begin{bmatrix} 2(y_i - \hat{y}_i)(-x_i) \\ 2(y_i - \hat{y}_i)(-1) \end{bmatrix}^\top \\ &= -2(y_i - \hat{y}_i) \begin{bmatrix} x_i \\ 1 \end{bmatrix}. \end{aligned}$$

O multiplicador 2 pode ser simplesmente incorporado na taxa de aprendizagem para facilitar a expressão. A norma quadrática do gradiente é  $\|\nabla L_i(w, b)\| = \sqrt{(\partial L_i/\partial w)^2 + (\partial L_i/\partial b)^2}$ . Especializando o algoritmo 5, obtemos então o algoritmo 6.

A tabela 3.3 mostra o resultado do experimento comparativo entre o cálculo determinístico e pela descida de gradiente do problema ilustrativo da regressão linear simples da tabela 2.1 na pág. 34 e figura 2.1 na pág. 34. O algoritmo pára, se o número máximo de iterações  $k_{\max} = 100$  for

<sup>4</sup>Em certas aplicações pode acontecer que o sistema linear seja *subdeterminado*. Nesse caso não há um único mínimo global. Um exemplo desse tipo de aplicação é processamento de imagens, onde a quantidade de parâmetros (os pixels) em geral supera a quantidade de padrões (as imagens).

**Algoritmo 6:** Descida de Gradiente Estocástica, Regressão Linear Simples

---

**Entrada:** Dados:  $n$  pares de entrada  $x_i$  e saída  $y_i$ ,  $i = 1, \dots, n$ ; Perda individual  $L_i(w, b)$  do padrão  $x_i$  e seu gradiente  $\nabla L_i(w, b)$ ; Taxa de aprendizagem  $\alpha$ ; Função que mapeia a entrada para saída  $y = f(x; w, b) = wx + b$ ; Número máximo de iterações  $k_{\max}$ ; Limiar da norma quadrática do valor do gradiente  $g_{\min}$

**Resultado:** Variáveis  $\mathbf{w}^* = (w^*, b^*)$  que supostamente minimiza função objetivo

- 1 Inicialize na iteração  $k = 0$  as variáveis  $w$  e  $b$  aleatoriamente com os valores  $w_0$  e  $b_0$ ;
- 2 **enquanto** ( $k < k_{\max}$ )  $E(||\nabla L(\mathbf{w}_k)|| > g_{\min})$  **faça**
- 3   Crie uma permutação  $p$  aleatória da sequência  $1, \dots, n$ ;
- 4   **para**  $i=1$  até  $n$  **faça**
- 5      $q \leftarrow p(i)$  /\* Escolhe um índice aleatoriamente \*/;
- 6      $\hat{y}_q \leftarrow f(x_q; w_k, b_k)$  /\* Calcule a saída \*/;
- 7      $L_q \leftarrow L(w_k, b_k)$  /\* Perda do padrão \*/;
- 8      $\partial L_q / \partial w \leftarrow -2(y_i - \hat{y}_i)x_i$  /\* Derivada parcial em relação ao peso  $w$  \*/;
- 9      $\partial L_q / \partial b \leftarrow -2(y_i - \hat{y}_i)$  /\* Derivada parcial em relação ao viés  $b$  \*/;
- 10     $w_{k+1} \leftarrow w_k + \alpha(y_i - \hat{y}_i)x_i$  /\* Atualização do peso \*/;
- 11     $b_{k+1} \leftarrow b_k + \alpha(y_i - \hat{y}_i)$  /\* Atualização do viés \*/;
- 12   **fim**
- 13    $||\nabla L(\mathbf{w}_k)|| \leftarrow \frac{1}{n} \sum_{q=1}^n ||\nabla L_q||$  /\* Norma média do gradiente sobre os  $n$  padrões \*/;
- 14    $k \leftarrow k + 1$ ;
- 15 **fim**
- 16  $w^* \leftarrow w_k$ ;
- 17  $b^* \leftarrow b_k$

---

atingido, ou se a norma do gradiente  $||\nabla L||$  ficar abaixo do limiar  $g_{\min} = 10^{-5}$ . O resultado sugere a equipotência dos métodos determinísticos e iterativos. Lembre que existe somente sete padrões, que causa uma variância grande nos valores de  $w$  e  $b$  durante a descida de gradiente. A evolução temporal mostra que após poucas iterações, os valores ficam estáveis dentro de uma faixa pequena.

A figura 3.4 mostra a trajetória de  $(w_k, b_k)$  no espaço dos parâmetros em duas e três dimensões. O algoritmo 6 foi usado para ajustar  $w$  e  $b$  após a apresentação de cada padrão em ordem aleatória. O número máximo de iterações foi definido como  $k_{\max} = 6$ . Tendo  $n = 7$  padrões, resulta em um total de  $6 \cdot 7 = 42$  passos (setas pretas) até o valor final (ponto azul). A taxa de aprendizagem foi definida em  $\alpha = 0.07$ . Em ambos os gráficos mostram-se no plano  $(w, b)$  as curvas de contorno, onde a função de perda  $L(w, b)$  tem valores constantes. O ponto verde marca os valores iniciais  $(w_0, b_0)$  dos parâmetros. O ponto azul é a posição dos parâmetros após o número máximo de iterações. O ponto vermelho é o valor ótimo  $(w^*, b^*) = (0.475, 0.325)$  pelo método determinístico. A sequência de setas pretas é a trajetória do par de parâmetros  $(w, b)$ , desde a inicialização no instante  $k = 0$  até a última iteração  $k = k_{\max}$ . O gráfico 3-D mostra o valor da função de perda  $L(w, b)$  no eixo vertical. O gráfico 2-D adicionalmente mostra como setas amarelas o gradiente da função de perdas em pontos de uma grelha. Esse gradiente é calculado baseado no lote de todos os  $n$  padrões  $\nabla L(w, b) = \frac{1}{n} \sum_i \nabla L_i(w, b)$ , definido na (eq. 2.48) na pág. 46. Repare que o primeiro passo de  $(w_0, b_0)$  a  $(w_1, b_1)$  (seta preta saindo do ponto verde) não coincide com a direção  $-\nabla L(w_0, b_0)$  do gradiente negativo nesse ponto verde, pois é o gradiente é calculado para um padrão somente, e não como média de todos os  $n$  padrões. Mesmo com tantas iterações, não foi possível chegar no ponto ótimo  $(w^*, b^*)$ . Seriam necessárias mais iterações e/ou uma taxa de aprendizagem diferente.

A figura 3.5 mostra novamente a descida de gradiente para o mesmo problema de aprendizagem supervisionada dos dois parâmetros  $w$  e  $b$ . O número máximo de iterações foi definido como

	$w$	$b$	$L(w, b)$	$\ \nabla L(w, b)\ $
Determinístico	0.475	0.325	0.839	-
SGD	0.445	0.405	0.845	1.58

Tabela 3.3: Comparação experimental de regressão linear por método determinístico e método iterativo. Quantidade de padrões  $n = 7$ , número máximo de iterações  $k_{\max} = 100$ , limiar da norma do gradiente  $g_{\min} = 10^{-5}$ , semente do gerador aleatório  $s = 66649$ .

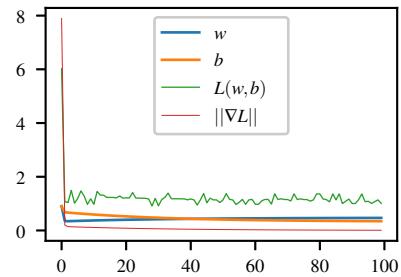


Figura 3.3: Evolução temporal durante  $k_{\max} = 100$  iterações dos valores dos parâmetros  $w$  e  $b$ , e da função de perda  $L$  e a norma da sua derivada  $\|\nabla L\|$ .

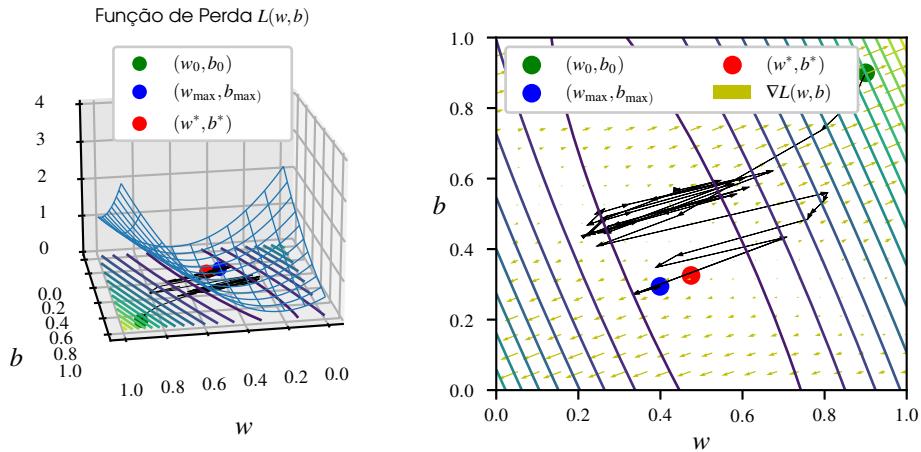
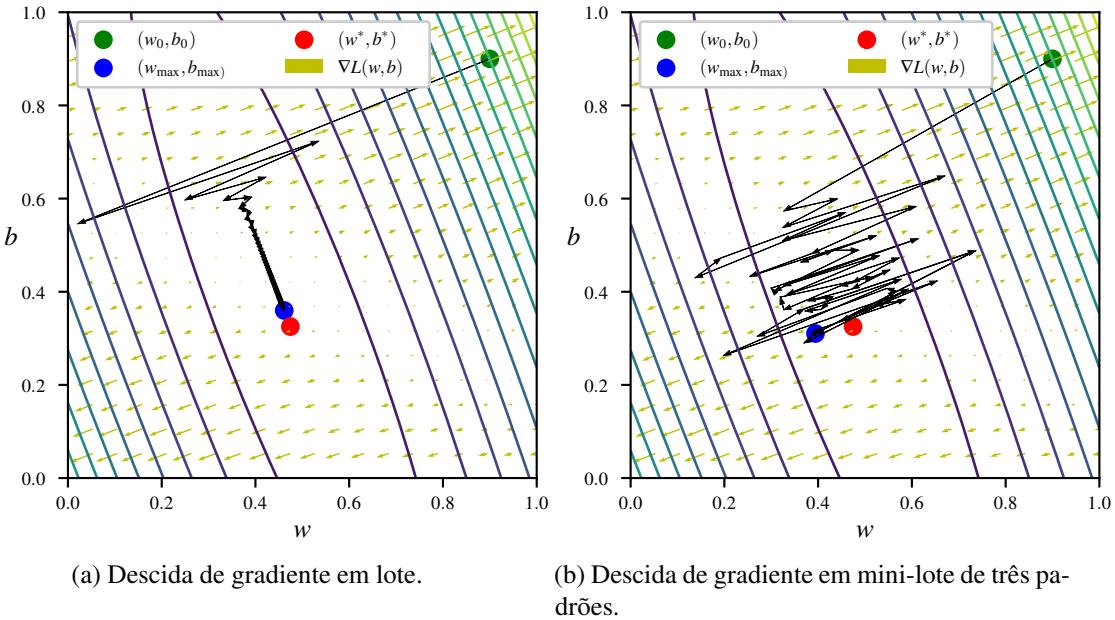


Figura 3.4: Descida de Gradiente Estocástica, Regressão Linear Simples.

$k_{\max} = 50$ , a taxa de aprendizagem como  $\alpha = 0.12$ . A diferença está no momento da atualização dos parâmetros. Na figura 3.5a, a atualização é feita baseada no gradiente da função de perda de todos os  $n = 7$  padrões. Como a atualização é feita após o lote inteiro, existem 50 passos. Repare que agora a primeira direção de descida coincide com o gradiente negativo, pois o gradiente é calculado baseado no lote inteiro. A figura 3.5b mostra o resultado da descida de gradiente, usando uma atualização por mini-lote de  $p = 3$  padrões. Como a atualização é feita após a apresentação do mini-lote, também existem 50 passos.

Uma observação da trajetória na figura 3.5a mostra um defeito essencial da estratégia de seguir cegamente o gradiente negativo. A posição ótima dos parâmetros não está nessa direção, exceto se as curvas de contorno forem concêntricas. Nesse caso, o gradiente negativo aponta sempre para o mínimo da função de perda. Retornaremos a esse assunto na apresentação do gradiente conjugado.

Podemos resumir que métodos iterativos de aprendizado de máquina baseado no princípio da descida de gradiente em geral não garantem a obtenção de um conjunto de parâmetros ótimos. Mesmo com o exemplo tão simples, o comportamento das várias versões das estratégias de descida de gradiente fornece resultados bastante variados, porém parece que a ideia leva mais cedo ou mais tarde a um resultado desejável.

Figura 3.5: Descida de Gradiente em lote e mini-lote (*batch* e *minibatch*).

### 3.2.3 Generalização de Regressão Linear Múltipla

Podemos ainda unificar todos os casos da descida de gradiente para a regressão linear múltipla. A distinção se aplica em relação à quantidade \$c\$ de saídas da função de mapeamento das entradas para as saídas, e em relação à quantidade \$p\$ de padrões que são usados para ajustar os parâmetros em um iteração do algoritmo. Reveja a definição de treinamento estocástico (\$p = 1\$), mini-lote (\$1 < p < n\$) e lote (\$p = n\$) da definição 3.2.2. É necessário estender o conceito da norma de um vetor para a norma de uma matriz e o conceito do gradiente de uma função escalar em relação a uma matriz. A introdução desta teoria vai facilitar a explicação como uma rede neural com pelo menos uma camada oculta ajusta os seus pesos. Vamos primeiro redefinir a função de perda baseada no erro quadrático entre os valores desejados e explicados pelo modelo. Para definir a função de perda baseada em uma matriz de diferença, precisamos de uma norma aplicável a uma matriz. É mais conveniente usar o quadrado de uma norma do que a própria norma.

**Definição 3.2.3 — Norma de Fröbenius (quadrática) de uma Matriz.** A norma quadrática de Fröbenius de uma matriz \$M\$ de dimensão \$p \times c\$ é

$$\|M\|_F^2 \stackrel{\text{def}}{=} \|M\|_2^2 = \sum_{j=1}^p \sum_{k=1}^c m_{j,k}^2 = \text{tr}(M^\top M), \quad (3.11)$$

onde o operador “tr” é o *traço* de uma matriz de forma quadrática, \$\text{tr}(A) \stackrel{\text{def}}{=} \sum\_i a\_{i,i}\$. Nesse cálculo, no fundo fazemos equivalentemente a *linearização* da matriz com \$p\$ linhas e \$c\$ colunas, criando o vetor \$\mathbf{v}\$ grande de dimensão \$p \cdot c\$. A \$j\$-ésima coluna da matriz é colocada entre as posições \$(j-1)p+1\$ e \$j \cdot p\$ do vetor de linearização. Depois calcula-se a norma quadrática deste vetor \$\|\mathbf{v}\|^2 = v\_1^2 + \dots + v\_{p \cdot c}^2\$, resultando no mesmo valor da norma matricial.

**Definição 3.2.4 — Função de Perda, Erro Quadrático, Caso Multidimensional.** Sejam dados \$1 \leq p \leq n\$ padrões de treinamento \$\mathbf{x}\_i\$, \$i \in \{1, \dots, n\}\$ em modalidade estocástica, mini-lote

ou lote, formando uma matriz de dados  $X$  de dimensão  $p \times (m + 1)$ . Usamos aqui a forma aumentada, onde a 0-ésima componente do  $i$ -ésimo padrão é  $x_{i,0} = 1$ ,  $i = 1, \dots, n$ , e o 0-ésimo peso de cada vetor de pesos é o viés  $w_{k,0} = b_k$ ,  $k = 1, \dots, c$ . Seja  $Y$  a matriz de saídas desejadas de dimensão  $p \times c$  em um problema de aprendizagem supervisionada, e seja  $\hat{Y}$  a matriz de saídas explicadas pelo modelo da mesma dimensão  $p \times c$ . Cada linha de  $Y - \hat{Y}$  contém, em forma transposta, a diferença entre o vetor desejado  $\mathbf{y}_{i_j}$  e o explicado  $\hat{\mathbf{y}}_{i_j} = \mathbf{f}(\mathbf{x}_{i_j})$  de um único padrão, o mapeamento em forma transposta do padrão  $\mathbf{x}_{i_j}$ , portanto a diferença entre a totalidade dos dados desejados e explicados pelo modelo é uma matriz dimensão  $p \times c$  é

$$Y - \hat{Y} = \begin{bmatrix} (\mathbf{y}_{i_1} - \hat{\mathbf{y}}_{i_1})^\top \\ \vdots \\ (\mathbf{y}_{i_j} - \hat{\mathbf{y}}_{i_j})^\top \\ \vdots \\ (\mathbf{y}_{i_p} - \hat{\mathbf{y}}_{i_p})^\top \end{bmatrix} = \begin{bmatrix} y_{i_1,1} - f_1(\mathbf{x}_{i_1}) & \cdots & y_{i_1,k} - f_k(\mathbf{x}_{i_1}) & \cdots & y_{i_1,c} - f_c(\mathbf{x}_{i_1}) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ y_{i_j,1} - f_1(\mathbf{x}_{i_j}) & \cdots & y_{i_j,k} - f_k(\mathbf{x}_{i_j}) & \cdots & y_{i_j,c} - f_c(\mathbf{x}_{i_j}) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ y_{i_p,1} - f_1(\mathbf{x}_{i_p}) & \cdots & y_{i_p,k} - f_k(\mathbf{x}_{i_p}) & \cdots & y_{i_p,c} - f_c(\mathbf{x}_{i_p}) \end{bmatrix}. \quad (3.12)$$

Então o quadrado da norma matricial, neste caso a norma de Fröbenius, da diferença dos dados desejados e explicados pelo modelo

$$L(W) = \|Y - \hat{Y}\|_F^2 \quad (3.13)$$

é a função de perda do problema de aprendizagem de um conjunto de treinamento de  $p$  padrões.

Lembre que os índices  $i_j$  das linhas foram escolhidos aleatoriamente no caso do mini-lote e no caso estocástico. Nesse último existe somente um única linha. A definição 3.2.4 é bem geral para mapeamentos que não necessariamente são lineares, ou seja, diferentes do modelo da Máquina Linear apresentada na seção 2.2.1 na pág. 53. Podemos desenhar modelos que envolvem matrizes que não são lineares, por exemplo, o modelo da rede neural do Perceptron Multicamada que conhiceremos posteriormente. Esse modelo calcula um mapeamento não linear  $\hat{Y} = Z(X, W)$  que produz uma matriz com o mesmo formato que a matriz dos dados desejados  $Y$ . O gradiente da função de perda, baseada na norma de Fröbenius no caso geral não linear, pela (eq. 2.53g) na pág. 49 é

$$\nabla L(W) = \frac{\partial \|Y - \hat{Y}\|_F^2}{\partial W} = \frac{\partial \|Y - Z(X, W)\|_F^2}{\partial W} \stackrel{2.53g}{=} -2(Y - \hat{Y})^\top \frac{\partial Z(X, W)}{\partial W}. \quad (3.14)$$

No momento, o nosso modelo é linear, então o mapeamento é a combinação linear  $Z(X, W) = XW$  dos  $p$  padrões  $X$  e dos  $(m + 1)c$  pesos  $W$ , veja a (eq. 2.67) na pág. 54 e (eq. 2.63) na pág. 53, com a diferença que temos agora  $p$  padrões, em geral em ordem aleatória. A aleatoriedade muda o índice  $i = 1, \dots, n$  em  $X$  e  $Y$  para  $i_j = 1, \dots, p$ . O gradiente da função de perda, baseada na norma de Fröbenius no caso linear, pela (eq. 2.53g) na pág. 49 é

$$\begin{aligned} \nabla L(W) &= \frac{\partial \|Y - \hat{Y}\|_F^2}{\partial W} = \frac{\partial \|Y - XW\|_F^2}{\partial W} = -2(Y - \hat{Y})^\top \frac{\partial XW}{\partial W} \\ &= -2(Y - \hat{Y})^\top X. \end{aligned} \quad (3.15)$$

Na última expressão foi calculado o gradiente do produto de duas matrizes em relação a uma matriz como se fosse uma conta unidimensional  $d(xw)/dw = x$ . Essa equação

$$\frac{\partial XW}{\partial W} = X \quad (3.16)$$

é apresentada sem prova. Ela baseia-se na identidade  $\partial(XWB)/\partial W = B^T \otimes X$ , e atribuindo à matriz  $B$  a matriz de identidade  $I$ . O operador “ $\otimes$ ” é o produto de Kronecker [81], uma generalização do *produto diádico*  $\mathbf{ab}^T$  de dois vetores de coluna  $\mathbf{a}$  e  $\mathbf{b}$ .

Dessa maneira, em uma forma muito compacta, é possível unificar todas as modalidades de treinamento, estocástico, mini-lote e lote na definição do gradiente com vários padrões, várias entradas e várias saídas. Então, a regra de atualização do algoritmo de descida de gradiente para todas essas modalidades é

$$W_{k+1} \leftarrow W_k - \alpha \nabla L(W)^T. \quad (3.17)$$

Um dos efeitos colaterais de mostrar essa forma unificada é conseguir o mesmo resultado da aprendizagem da Máquina Linear da (eq. 2.70) na pág. 56,  $W = X^T Y$  através da (eq. 3.15). Obviamente temos que usar a modalidade de treinamento em lote, em que a matriz de dados tem todos os  $n$  padrões como as linhas em forma transposta.

*Demonstração.* Zerando esse gradiente em forma transposta, temos

$$\begin{aligned} \nabla L(W)^T &= O^T \\ \Leftrightarrow \quad &\left[ (Y - \hat{Y})^T X \right]^T = O^T \\ \Leftrightarrow \quad &\left[ (Y - XW)^T X \right]^T = O^T \\ \xleftarrow[2.59b]{2.59c} \quad &X^T [(Y - XW)] = O^T \\ \Leftrightarrow \quad &X^T Y = X^T X W \\ \Leftrightarrow \quad &\left[ X^T X \right]^{-1} Y = W \\ \Leftrightarrow \quad &W = X^T Y \end{aligned} \quad (3.18)$$

■

A matriz zero  $O$  tem dimensão  $c \times (m + 1)$ .

### 3.3 Descida de Gradiente para Funções Não Lineares

As aplicações mais impressionantes de aprendizado de máquina no fundo usam o mesmo princípio na aprendizagem supervisionada, a descida de gradiente. A diferença em relação aos mapeamentos lineares até agora estudados é que a função de perda é muito complexa e requer um grande esforço computacional para procurar um mínimo. Uma boa entrada nesse contexto é o estudo de funções quadráticas. De fato, na regressão linear, a função de perda baseado no erro quadrático é uma função quadrática, seja a perda individual, perda mini-lote ou perda lote. Lembre que a função de perda para um lote de  $n$  padrões (com saída  $y$  unidimensional) foi definida como  $L(w, b) = \frac{1}{n} \sum_i (y_i - \hat{y}_i)^2$  na (eq. 2.56) na pág. 50. Considere também a figura 3.4 e a figura 3.5. Se para dois argumentos  $w$  e  $b$  a função fosse linear, em 3-D seria um plano, e as linhas de contorno seriam retas, mas esse caso pode ser descartado a priori, pois um plano não tem um ponto mínimo em um domínio sem limites. Observe que para um modelo não linear, mesmo usando a função de perda quadrática, a dependência da função de perda dos parâmetros livres do modelo não é quadrática, pois o mapeamento entrada para saída não é linear. Então, neste caso, temos uma função quadrática de uma função não linear, resultando em uma função de grau acima de dois.

Para o estudo mais sistemático da minimização de funções quadráticas vamos outra vez adequar a nomenclatura à literatura de otimização e chamar a função objetivo a ser minimizada de  $f$  e o argumento  $\mathbf{x}$ , veja a seção 3.1 na pág. 79.

### 3.3.1 Funções Quadráticas

Uma classe de funções muito importante para entender o mecanismo de descida de gradiente é a função quadrática. A função quadrática é definida como

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top A \mathbf{x} + \mathbf{b}^\top \mathbf{x} + c, \quad (3.19)$$

onde  $\mathbf{x} \in \mathbb{R}^m$  é o argumento de entrada de  $m$  componentes em forma de vetor de coluna  $m \times 1$ ,  $A$  é uma matriz simétrica  $m \times m$  com elementos reais, ou seja  $A^\top = A$ ,  $a_{i,j} \in \mathbb{R}, i, j = 1, \dots, m$ ,  $\mathbf{b}$  é um vetor de coluna  $m \times 1$ , e  $c \in \mathbb{R}$  é um escalar constante. O caso unidimensional  $m = 1$  é uma especialização, em que todos os três parâmetros  $a \in \mathbb{R}, b \in \mathbb{R}, c \in \mathbb{R}$  são escalares e a função especializa para

$$f(x) = \frac{1}{2} ax^2 + bx + c. \quad (3.20)$$

Note que em um exemplo prático de aprendizado de máquina, por causa do mapeamento altamente não linear do modelo, não temos conhecimento que a função de perda tem uma forma quadrática, nem temos acesso direto aos parâmetros, que nesse caso seriam  $A$ ,  $\mathbf{b}$ , e  $c$ . Ainda assim, o estudo da função quadrática é extremamente importante para entender as limitações na concepção de um sistema real de aprendizado de máquina.

■ **Exemplo 3.6** Considere a função quadrática bidimensional, com a variável  $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$  e os parâmetros  $A = \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix}$ ,  $\mathbf{b} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$ ,  $c = 0$

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 2 \\ 1 \end{bmatrix}^\top \mathbf{x} + 0 = x_1^2 + x_1 x_2 + 2x_1 + 2x_2^2 + x_2 \quad (3.21)$$

■

■ **Exemplo 3.7** A função de perda  $L(w, b) = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$ , baseada no erro quadrático entre o valor desejado  $y$  e o valor explicado pelo modelo  $\hat{y}$  tem a estrutura de uma função quadrática da (eq. 3.19). Se o modelo tem uma forma linear  $y(x; w, b) = wx + b$ , a função de perda realmente é quadrática<sup>5</sup>. Basta considerar a perda individual  $L_i(w, b) = (y_i - \hat{y}_i)^2$ . Temos, agrupando  $w$  e  $b$  em um vetor, e ainda dividindo por dois para facilitar a expressão

$$\begin{aligned} \frac{1}{2} L_i \left( \begin{bmatrix} w \\ b \end{bmatrix} \right) &= \frac{1}{2} (y_i - \hat{y}_i)^2 = \frac{1}{2} (y_i - wx_i - b)^2 \\ &= \frac{1}{2} x_i^2 w^2 + b^2 - x_i y_i w + x_i w b - y_i b + \frac{1}{2} y_i^2 \\ &= \frac{1}{2} \begin{bmatrix} w \\ b \end{bmatrix}^\top \begin{bmatrix} x_i^2 & x_i \\ x_i & 1 \end{bmatrix} \begin{bmatrix} w \\ b \end{bmatrix} - \begin{bmatrix} x_i y_i \\ y_i \end{bmatrix}^\top \begin{bmatrix} w \\ b \end{bmatrix} + \frac{1}{2} y_i^2. \end{aligned} \quad (3.22)$$

Comparando com a definição da função quadrática da (eq. 3.19) a identificação da matriz  $A$ , do vetor  $\mathbf{b}$  e da constante  $c$  fica óbvia. (Não confunda o vetor  $\mathbf{b}$  com o viés  $b$  da função de perda). A perda do lote dos padrões é simplesmente a média das perdas individuais. Outra observação

<sup>5</sup>Imagine um modelo não linear nos parâmetros  $w$  e  $b$ , definido como  $y(x; w, b) = wx + e^b$ . A função de perda baseado no quadro do erro, seria  $L_i(x_i; w, b) = (y_i - \hat{y}_i)^2$ . Expandido esta expressão resulta em uma função de perda não quadrática nos parâmetros  $w$  e  $b$ , definida como  $L_i(x_i; w, b) = y_i^2 - 2x_i y_i w - 2y_i e^b + w^2 x_i^2 + 2x_i w e^b + e^{2b}$ .

interessante é quando obtemos a solução ótima por zerar o gradiente. O gradiente individual da função de perda é

$$\nabla L_i \left( \begin{bmatrix} w \\ b \end{bmatrix} \right) = \begin{bmatrix} w \\ b \end{bmatrix}^\top \begin{bmatrix} x_i^2 & x_i \\ x_i & 1 \end{bmatrix} - \begin{bmatrix} x_i y_i \\ y_i \end{bmatrix} \iff \begin{bmatrix} x_i^2 & x_i \\ x_i & 1 \end{bmatrix} \begin{bmatrix} w \\ b \end{bmatrix} = \begin{bmatrix} x_i y_i \\ y_i \end{bmatrix},$$

onde foram usadas as regras de transposição (eq. 2.59c) na pág. 51 e regras de derivada multidimensional (eq. 2.53c), (eq. 2.53d) e (eq. 2.53e) na pág. 49. Se somarmos (ou calcularmos a média) chegaremos exatamente na solução bem conhecida da regressão linear, definida pelo sistema linear  $2 \times 2$  da (eq. 2.52) na pág. 47. ■

### 3.3.2 Minimização de Funções Quadráticas

A derivada de uma função quadrática unidimensional da (eq. 3.20), com os escalares  $a, b, c$  tem que ser zerada como condição necessária de um mínimo, ou seja

$$f'(x) = df(x)/dt = ax + b = 0,$$

o que leva diretamente à solução

$$x = -b/a.$$

Se adicionalmente, a segunda derivada  $f''(x) = d^2 f(x)/dt^2 = a$  de  $f(x)$  for positiva, ou seja,  $a > 0$ , a função  $f(x)$  é (estritamente) convexa, e o extremo é um mínimo, ou seja

$$x^* = \frac{-b}{a} = \arg \min_x \frac{1}{2} ax^2 + bx + c.$$

No contexto de aprendizado de máquina, a dimensão  $m$  do argumento  $\mathbf{x} \in \mathbb{R}^m$  é multidimensional, então a condição necessária para ter um mínimo da função quadrática da (eq. 3.19) é o gradiente nulo da função objetivo. Usando novamente as regras (eq. 2.53d) e (eq. 2.53e) na pág. 49, além da simetria da matriz  $A$ , temos

$$\nabla f(\mathbf{x}) = \mathbf{x}^\top A + \mathbf{b}^\top = \mathbf{0}^\top \xrightarrow{2.59c} A\mathbf{x} + \mathbf{b} = \mathbf{0}, \quad (3.23)$$

ou seja, a solução  $\mathbf{x}^*$  é a solução<sup>6</sup> de um sistema linear  $A\mathbf{x} = -\mathbf{b}$ . Realmente, a origem de otimização de funções não lineares estudou inicialmente a solução iterativa de sistemas lineares. A condição suficiente para  $\mathbf{x}^*$  ser um mínimo, além do gradiente zero, é a analogia da segunda derivada positiva em mais uma dimensão. Precisamos da definição da matriz Hessiana.

**Definição 3.3.1 — Matriz Hessiana.** Seja  $f(\mathbf{x})$  uma função de um argumento vetorial  $\mathbf{x}$  de dimensão  $m$ . A matriz Hessiana é uma matriz simétrica de dimensão  $m \times m$

$$H(f(\mathbf{x})) \stackrel{\text{def}}{=} \nabla^2 f(\mathbf{x}) \stackrel{\text{def}}{=} \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_m} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_m \partial x_1} & \frac{\partial^2 f}{\partial x_m \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_m^2} \end{bmatrix} \quad (3.24)$$

Ela tem na  $i$ -ésima linha, na  $j$ -ésima coluna a derivada parcial

$$H_{i,j} \stackrel{\text{def}}{=} \frac{\partial}{\partial x_i} \left( \frac{\partial f}{\partial x_j} \right)$$

<sup>6</sup>Na prática resolve-se o sistema linear  $A\mathbf{x} = -\mathbf{b}$  não pela multiplicação de  $-\mathbf{b}$  pela inversa  $A^{-1}$  de  $A$ , mas pelos métodos estabelecidos de cálculo numérico, por exemplo, pela eliminação de Gauss.

primeiro em relação a  $j$ -ésima variável, e depois em relação a  $i$ -ésima variável. Devida à simetria, a ordem é irrelevante.

**Exemplo 3.8** Considere a função quadrática de duas dimensões<sup>7</sup> definida no exemplo 3.6

$$f(\mathbf{x}) = x_1^2 + x_1x_2 + 2x_1 + 2x_2^2 + x_2. \quad (3.25)$$

Temos como primeira derivada, em relação à primeira variável

$$\partial f / \partial x_1 = 2x_1 + x_2 + 2.$$

Novamente a derivada desta expressão em relação à primeira variável é

$$\partial(2x_1 + x_2 + 2) / \partial x_1 = 2 = H_{1,1},$$

e relação à segunda variável é

$$\partial(2x_1 + x_2 + 2) / \partial x_2 = 1 = H_{1,2}.$$

Temos como primeira derivada, em relação à segunda variável

$$\partial f / \partial x_2 = x_1 + 4x_2 + 1.$$

Novamente a derivada desta expressão em relação à primeira variável é

$$\partial(x_1 + 4x_2 + 1) / \partial x_1 = 1 = H_{2,1},$$

e relação à segunda variável é

$$\partial(x_1 + 4x_2 + 1) / \partial x_2 = 4 = H_{2,2}.$$

Então a Hessiana é  $H = \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix}$ , o que corresponde a matriz  $A$  no exemplo 3.6, ou seja  $H = A$ . Podemos chegar diretamente ao resultado aplicando a (eq. 2.53c) na pág. 49, à (eq. 3.23) ou seja,  $\nabla^2 f(\mathbf{x}) = \nabla^2 \left[ \frac{1}{2} \mathbf{x}^\top A \mathbf{x} + \mathbf{b}^\top \mathbf{x} \right] = \nabla [A \mathbf{x} + \mathbf{b}] = A$ . ■

Se adicionalmente a Hessiana for definida positiva o extremo é um mínimo. Nesse caso, a função quadrática é uma função convexa.

**Definição 3.3.2 — Matriz Definida positiva.** Uma matriz real e simétrica  $M$  é definida positiva, se

$$\mathbf{x}^\top M \mathbf{x} > 0 \quad (3.26)$$

para qualquer vetor de coluna  $\mathbf{x}$  não nulo. A matriz é *semidefinida positiva*, se  $\mathbf{x}^\top M \mathbf{x} \geq 0$ .

Aproveitamos a ocasião para definir a matriz Jacobiana<sup>8</sup> que será necessária posteriormente.

**Definição 3.3.3 — Matriz Jacobiana.** Sejam

$$\mathbf{h}(\mathbf{x}) = \begin{bmatrix} h_1(\mathbf{x}) \\ \vdots \\ h_i(\mathbf{x}) \\ \vdots \\ h_n(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} h_1 \left( [x_1 \ \dots \ x_j \ \dots \ x_m]^\top \right) \\ \vdots \\ h_i \left( [x_1 \ \dots \ x_j \ \dots \ x_m]^\top \right) \\ \vdots \\ h_n \left( [x_1 \ \dots \ x_j \ \dots \ x_m]^\top \right) \end{bmatrix}$$

<sup>7</sup>Repare que a dimensão  $m$  do argumento  $\mathbf{x}$  da função não tem a ver com a propriedade de ser uma função quadrática.

<sup>8</sup>Vamos usar a letra “h” para denominar as derivadas na Jacobiana, para não confundir com o “f” da função objetivo

$n$  funções  $h_i(\mathbf{x})$  de um argumento vetorial  $\mathbf{x}$  de dimensão  $m$ ,  $i = 1, \dots, n$ . As funções estão organizados em um vetor de coluna de dimensão  $n$ . A matriz Jacobiana é uma matriz de dimensão  $n \times m$

$$J(\mathbf{h}(\mathbf{x})) \stackrel{\text{def}}{=} \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \dots & \frac{\partial h_1}{\partial x_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_n}{\partial x_1} & \dots & \frac{\partial h_n}{\partial x_m} \end{bmatrix}. \quad (3.27)$$

Ela tem na  $i$ -ésima linha, na  $j$ -ésima coluna a derivada parcial

$$J_{i,j} \stackrel{\text{def}}{=} \frac{\partial h_i}{\partial x_j}$$

da  $i$ -ésima função em relação à  $j$ -ésima variável  $x_j$ .

■ **Exemplo 3.9** Sejam dadas  $n = 2$  funções de  $m = 2$  variáveis

$$\mathbf{h}(\mathbf{x}) = \begin{bmatrix} h_1(\mathbf{x}) \\ h_2(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} h_1 \left( \begin{bmatrix} x_1 & x_2 \end{bmatrix}^\top \right) \\ h_2 \left( \begin{bmatrix} x_1 & x_2 \end{bmatrix}^\top \right) \end{bmatrix} = \begin{bmatrix} 2x_1 + x_2 + 2 \\ x_1 + 4x_2 + 1 \end{bmatrix}$$

A Jacobiana é  $J(\mathbf{h}(\mathbf{x})) = \begin{bmatrix} \frac{\partial(2x_1+x_2+2)}{\partial x_1} & \frac{\partial(2x_1+x_2+2)}{\partial x_2} \\ \frac{\partial(x_1+4x_2+1)}{\partial x_1} & \frac{\partial(x_1+4x_2+1)}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix}$  e coincide com a Hessiana do exemplo 3.8. De fato podemos reconhecer as funções do exemplo 3.9 como as primeiras derivadas da função quadrática, ou seja  $\mathbf{h}(\mathbf{x}) = \left[ \nabla (x_1^2 + x_1x_2 + 2x_1 + 2x_2^2 + x_2)^\top \right]$ . Lembre que o gradiente da definição 2.1.7 na pág. 37 é um vetor de linha de dimensão  $1 \times m$ . Esse resultado é generalizável, estabelecendo uma ligação<sup>9</sup> entre Jacobiana e Hessiana

$$H(f(\mathbf{x})) = J \left( \nabla f(\mathbf{x})^\top \right).$$

Nesta equação, o gradiente transposto  $\nabla f(\mathbf{x})^\top$  tem o papel das funções  $\mathbf{h}(\mathbf{x})$  na definição 3.3.3 da Jacobiana. Além disso, a quantidade  $n$  de funções coincide com a quantidade  $m$  de argumentos da função  $f$ , sendo neste caso a Jacobiana uma matriz de forma quadrada  $m \times m$ .

### 3.3.3 Minimização Iterativa de Funções Quadráticas

Embora exista uma solução determinística da minimização da função quadrática da (eq. 3.19) pela solução do sistema linear da (eq. 3.23), a minimização iterativa será estudada em seguida. A motivação disso está baseado no fato que esses métodos iterativos serão aplicados para funções em geral, ou seja, altamente não lineares. Já temos o algoritmo 4 e o algoritmo 5 para a descida de gradiente em lote ou estocástico que buscam um mínimo  $\mathbf{x}^*$  da função objetivo  $f(\mathbf{x})$ . Sendo  $\mathbf{x}_0$  o ponto inicial da busca, o algoritmo gera uma sequência que idealmente atinge o valor ótimo  $\mathbf{x}^*$  como

$$\mathbf{x}_0 \rightarrow \mathbf{x}_1 \rightarrow \dots \rightarrow \mathbf{x}_k \rightarrow \mathbf{x}_{k+1} \rightarrow \dots \rightarrow \mathbf{x}^*.$$

<sup>9</sup>Um fato notável é que Ludwig Otto Hesse (22 de abril de 1811 a 4 de agosto de 1874) era aluno de doutorado de Carl Gustav Jacob Jacobi (10 de dezembro de 1804 a 18 de fevereiro de 1851), de certa maneira o aluno definiu a segunda derivada multidimensional, o seu orientador a primeira derivada multidimensional [14].

### Descida de Gradiente (Método do Máximo Declive)

Vamos refinar o conceito da descida de gradiente. A direção de descida foi definida na definição 3.2.1 na pág. 85. Na verdade a decida de gradiente até agora estudada segue a uma direção oposta ao gradiente da função objetivo no ponto atual  $\mathbf{x}_k$  que corresponde a direção em que a função diminui *ao máximo* (declive máximo), ou seja,  $\mathbf{p}_k = -\nabla f(\mathbf{x}_k)^\top$ . Se existe um declive máximo, obviamente existem direções saindo do ponto atual com menos declive.

**Definição 3.3.4 — Derivada Direcional.** Seja  $\nabla f(\mathbf{x})$  o gradiente<sup>a</sup> de uma função  $f$  no ponto  $\mathbf{x}$ . Seja  $\mathbf{v}$  uma direção, definida por um vetor no mesmo espaço vetorial que o gradiente. Então o produto escalar do gradiente com a direção  $\mathbf{v}$  é a *derivada direcional* da função, relativa à direção

$$\nabla_{\mathbf{v}} f(\mathbf{x}) \stackrel{\text{def}}{=} \frac{\partial f(\mathbf{x})}{\partial \mathbf{v}} = \nabla f(\mathbf{x}) \cdot \mathbf{v}. \quad (3.28)$$

<sup>a</sup>Nesta definição, o gradiente é representado por um vetor de coluna.

Embora o gradiente e o vetor  $\mathbf{v}$  sejam multidimensionais, a derivada direcional é um escalar  $\nabla_{\mathbf{v}} f(\mathbf{x}) \in \mathbb{R}$ . Ela mede a variação da função  $f(\mathbf{x})$  ao longo do vetor  $\mathbf{v}$ , portanto a sua maximização identifica o vetor  $\mathbf{v}$  ao longo do qual a função cresce mais.

■ **Exemplo 3.10** Em vez de considerar a derivada direcional, vamos considerar o oposto da mesma para facilitar a visualização. Considere a figura 3.6. Mostram-se as linhas de contorno da função quadrática definida no exemplo 3.6. A solução analítica do mínimo  $\mathbf{x}^* = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$  obtém-se como solução do sistema linear  $A\mathbf{x} = -\mathbf{b}$  (condição necessária), ou seja  $\begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix} \mathbf{x} = \begin{bmatrix} -2 \\ -1 \end{bmatrix}$ , com  $f(\mathbf{x}^*) = -1$ , e confirmada pela propriedade da Hessiana  $H = A$  ser positiva definida. Se for possível expressar a Hessiana pela decomposição de Cholesky [34] como  $H = L^\top L$  é possível comprovar que  $H$  é positiva definida. Temos

$$\mathbf{x}^\top A\mathbf{x} = \mathbf{x}^\top L^\top L\mathbf{x} = (\mathbf{x}^\top L^\top)(L\mathbf{x}) \stackrel{2.59c}{=} (L\mathbf{x})^\top (L\mathbf{x}) = \|L\mathbf{x}\|^2 > 0.$$

A norma (quadrática)  $\|\cdot\|^2$  de qualquer vetor não nulo é positiva. A matriz de decomposição de Cholesky é  $L = \begin{bmatrix} \sqrt{2} & \frac{1}{2}\sqrt{2} \\ 0 & \frac{1}{2}\sqrt{14} \end{bmatrix}$ , com a propriedade  $\begin{bmatrix} \sqrt{2} & 0 \\ \frac{1}{2}\sqrt{2} & \frac{1}{2}\sqrt{14} \end{bmatrix} \begin{bmatrix} \sqrt{2} & \frac{1}{2}\sqrt{2} \\ 0 & \frac{1}{2}\sqrt{14} \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix} = H$ ,

portanto temos um mínimo da função quadrática. Vamos considerar o ponto  $\mathbf{x} = \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}$ . O valor da

função (eq. 3.21) neste ponto é  $f(\mathbf{x}) = \frac{1}{2} \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}^\top \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix} \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix} + \begin{bmatrix} 2 \\ 1 \end{bmatrix}^\top \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix} + 0 = 2.5$ . Observe que o ponto azul  $\mathbf{x}$  fica em cima da linha de contorno com este valor 2.5. O valor do gradiente no ponto  $\mathbf{x}$  é  $\nabla f(\mathbf{x})^\top = \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix} \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix} + \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 3.5 \\ 3.5 \end{bmatrix}$ . Usando uma taxa de aprendizagem de  $\alpha = 0.2$ , o gradiente negativo ponderado, o seja, o passo no algoritmo da descida de gradiente é  $\Delta\mathbf{x} = -\alpha \nabla f(\mathbf{x})^\top = -0.2 \begin{bmatrix} 3.5 \\ 3.5 \end{bmatrix} = \begin{bmatrix} -0.7 \\ -0.7 \end{bmatrix}$ , marcado por uma seta vermelha. Seja  $\mathbf{v} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$  o vetor para qual o passo é projetado arbitrariamente. Portanto a derivada direcional (negativa e ponderada) é  $-\alpha \nabla_{\mathbf{v}} f(\mathbf{x}) = -\alpha \nabla f(\mathbf{x}) \cdot \mathbf{v} = \begin{bmatrix} -0.7 \\ -0.7 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ -1 \end{bmatrix} = -0.7 \cdot 0 + (-0.7 \cdot (-1)) = 0.7$ .

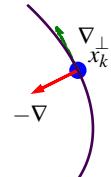
Repare mais uma vez que para o cálculo da derivada direcional, o gradiente é representado por um vetor de coluna. Em geral, pela definição 2.1.7 na pág. 37 é um vetor de linha. ■

Obviamente a derivada direcional atinge o seu maior valor, se o vetor  $\mathbf{v}$  em cima do qual o gradiente é projetado, é o próprio gradiente. Nesse caso, o ângulo entre o gradiente e o vetor de projeção  $\mathbf{v}$  é nulo, e consequentemente o seu cosseno é 1. Reveja a teoria do produto escalar da definição 1.1.1 na pág. 11 e figura 1.1 na pág. 13. Assim, a derivada direcional tem o valor  $\nabla_{\nabla f(\mathbf{x})} f(\mathbf{x}) = \nabla f(\mathbf{x}) \cdot \nabla f(\mathbf{x}) = \|\nabla f(\mathbf{x})\|^2$ .

**Corolário 3.3.1** O gradiente  $\nabla f(\mathbf{x})$  de uma função multidimensional  $f(\mathbf{x})$  aponta na direção do maior crescimento da função.

*Demonastração.* Revendo a definição do produto escalar  $\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \|\mathbf{b}\| \cos \theta$ , aplicado à derivada direcional, temos  $\nabla_{\mathbf{v}} f(\mathbf{x}) = \nabla f(\mathbf{x}) \cdot \mathbf{v} = \|\nabla f(\mathbf{x})\| \|\mathbf{v}\| \cos \theta$ . Dessa maneira, o produto escalar é maximizado se o ângulo  $\theta$  entre os dois vetores é zero, ou seja, se  $\nabla f(\mathbf{x})$  e  $\mathbf{v}$  têm a mesma direção. Consequentemente a derivada direcional da função  $f(\mathbf{x})$  é maximizada, se a direção definida é o próprio gradiente da função, ou seja  $\mathbf{v} = \nabla f(\mathbf{x})$ . ■

O exato contrário acontece com o crescimento da função ao longo das linhas de contorno em uma vizinhança muito pequena. Por definição, o contorno é uma região no espaço vetorial dos argumentos  $\mathbf{x}$ , onde o valor da função não varia. O vetor  $\nabla_{\perp}$  que é ortogonal ao gradiente  $\nabla$  (ou seu oposto  $-\nabla$ ), ou seja, a direção de descida de máximo declive é uma aproximação linear desse contorno, e então não varia em uma vizinhança pequena do ponto atual no algoritmo da descida.



**Corolário 3.3.2** A função multidimensional  $f(\mathbf{x})$  não cresce na direção  $\nabla_{\perp}$  que é ortogonal ao gradiente  $\nabla f(\mathbf{x})$ , paralela às linhas de contorno  $f(\mathbf{x}) = \text{const}$ , em uma vizinhança pequena ao longo desta direção perpendicular ao gradiente ou seja,

$$f(\mathbf{x} + \epsilon \nabla_{\perp}) = f(\mathbf{x}).$$

Isso é equivalente ao fato que o gradiente direcional  $\nabla_{\nabla_{\perp}} f(\mathbf{x})$  na direção da constância da função é zero.

*Demonastração.* Em uma vizinhança pequena, o vetor  $\mathbf{x} + \epsilon \nabla_{\perp}$  aproxima o contorno que por definição tem um valor constante da função objetivo. ■

No caso bidimensional, o contorno é uma curva, e o vetor  $\nabla_{\perp}$  é a tangente<sup>a</sup> dessa curva no ponto  $\mathbf{x}$ .

<sup>a</sup>Em um espaço de alta dimensão, o gradiente  $\nabla$  da função é o vetor normal do hiperplano tangencial e sua contrapartida ortogonal  $\nabla_{\perp}$  é um vetor dentro deste hiperplano.

A figura 3.6 revela um grande defeito de seguir a direção do maior crescimento da função. Essa direção  $-\nabla f(\mathbf{x})$ , em geral, *não* aponta para o mínimo  $\mathbf{x}^*$ . Embora o algoritmo da descida de gradiente segue para o mínimo, os passos parecem seguir uma trajetória zigue-zague. Isso fica bem nítido na figura 3.3, figura 3.5a e figura 3.5b. A consequência é que o algoritmo básico da descida de gradiente é importante para estudar a teoria de otimização de uma função objetivo, mas existem alternativas melhores. Dentro do contexto de uma função quadrática, definida na (eq. 3.19), a teoria providencia um algoritmo ótimo que chega o mais rápido no mínimo  $\mathbf{x}^*$ , a

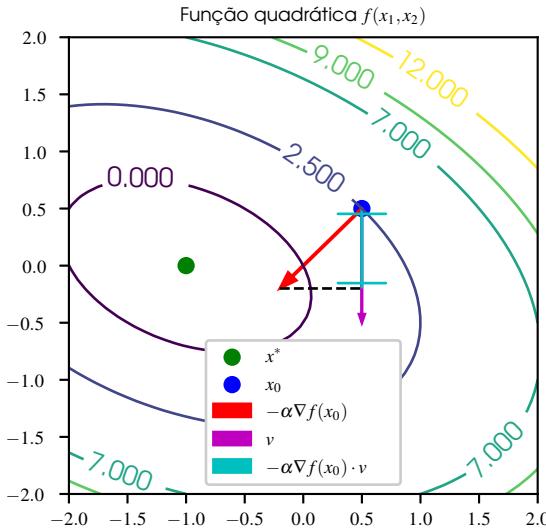


Figura 3.6: Descida de Gradiente. Derivada direcional (oposta e ponderada).

descida pelo *gradiente conjugado*.

Vamos precisar primeiro de um conceito que procura um mínimo ao longo de um caminho no espaço multidimensional, a busca linear. Embora o espaço possa ser de altíssima dimensão, a busca linear é definida por uma função unidimensional. Isso facilita encontrar um mínimo ao longo de um caminho definido por um vetor.

**Definição 3.3.5 — Busca Linear.** Seja  $f(\mathbf{x})$  uma função objetivo para qual se tenta encontrar um mínimo no ponto ótimo  $\mathbf{x}^* \in \mathbb{R}^m$ . Seja  $\mathbf{x}_k \in \mathbb{R}^m$  o ponto atual na busca, com valor da função  $f(\mathbf{x}_k) \in \mathbb{R}$ . Seja  $\mathbf{p}_k \in \mathbb{R}^m$  um vetor que define a direção de busca, ao longo do qual se procura o mínimo e seja  $\alpha \in \mathbb{R}$  um escalar que controla quão longe a busca se afasta a partir de  $\mathbf{x}_k$  em direção à  $\mathbf{p}_k$ . A busca linear tenta minimizar a função objetivo ao longo dessa direção, ou seja, minimizar a função unidimensional

$$\ell(\alpha) \stackrel{\text{def}}{=} f(\mathbf{x}_k + \alpha \mathbf{p}_k), \quad (3.29)$$

ou seja,

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k, \quad \alpha_k = \arg \min_{\alpha} f(\mathbf{x}_k + \alpha \mathbf{p}_k) = \arg \min_{\alpha} \ell(\alpha), \quad (3.30)$$

onde  $\mathbf{x}_{k+1}$  minimiza a função objetivo ao longo da direção de busca linear. O escalar que minimiza o valor da função na busca linear é denominado  $\alpha_k$ . Como este ponto será o próximo ponto em um algoritmo iterativo de descida de gradiente, ele naturalmente foi rotulado com o índice  $(k+1)$ . Em geral, a função objetivo, neste ponto  $\mathbf{x}_{k+1}$  é maior que no mínimo global  $\mathbf{x}^*$  da função.

Repare mais uma vez que a função  $\ell$  é uma função de um único argumento escalar  $\alpha$ . O ponto atual  $\mathbf{x}_k$  e a direção de busca  $\mathbf{p}_k$  são vetores invariantes na (eq. 3.29). A variável é  $\alpha$ .

No caso do algoritmo básico da descida de gradiente (máximo declive), a direção de busca coincide com o gradiente negativo, ou seja

$$\mathbf{p}_k = -\nabla f(\mathbf{x}_k)^T,$$

compare com a definição 3.2.1. Em algoritmos mais sofisticados a direção de busca em geral *não* será o gradiente oposto, ou seja  $\mathbf{p}_k \neq -\nabla f(\mathbf{x}_k)^\top$ . De qualquer maneira, a direção  $\mathbf{p}_k$  tem que ser uma direção de descida.

**Corolário 3.3.3** Em um algoritmo de descida de gradiente, uma direção de busca  $\mathbf{p}_k$  é uma direção de descida, se

$$\mathbf{p}_k \cdot \nabla f(\mathbf{x}_k) < 0. \quad (3.31)$$

*Demonstração.* A fronteira entre uma direção de subida e descida é a direção ortogonal  $\nabla_\perp$  ao gradiente  $\nabla$  do corolário 3.3.2. Qualquer direção de descida  $\mathbf{p}_k$  entre a direção ortogonal  $\nabla_\perp$  e o gradiente negativo  $-\nabla$  tem um ângulo  $\tilde{\theta}$  com  $-90^\circ < \tilde{\theta} < 90^\circ$ . Então para o ângulo complementar

$$\theta = \pi + \tilde{\theta}$$

entre a direção de descida  $\mathbf{p}_k$  e o gradiente *positivo*  $\nabla$  temos o valor  $90^\circ < \theta < 270^\circ$ , com  $\cos(\theta) < 0$ . Então para o produto escalar entre a direção e o gradiente positivo temos um valor negativo

$$\mathbf{p}_k \cdot \nabla f(\mathbf{x}_k) = \|\mathbf{p}_k\| \|\nabla f(\mathbf{x}_k)\| \cos(\theta) < 0,$$

pois a norma  $\|\mathbf{v}\|$  de qualquer vetor não nulo  $\mathbf{v}$  é positiva, e o coseno para esta faixa é negativo. ■

■ **Exemplo 3.11** A figura 3.7 mostra novamente as linhas de contorno da função quadrática do exemplo 3.6. Seja  $\mathbf{x}_k = [1/2 \ 1/2]^\top$  o ponto atual no algoritmo de descida de gradiente (ponto verde). Já calculamos no exemplo 3.10 o gradiente no ponto como  $\nabla f(\mathbf{x}_k) = [3.5 \ 3.5]$ . No algoritmo do Máximo Declive, buscamos então na direção oposta (seta vermelha)  $\mathbf{p}_k = -\nabla f(\mathbf{x}_k)^\top = [-3.5 \ -3.5]^\top$  o menor valor  $\mathbf{x}_{k+1}$  da função objetivo. Visivelmente, não podemos alcançar o mínimo global  $\mathbf{x}^* = [-1 \ 0]^\top$ , porque não fica na trajetória da busca linear. O que podemos obter como melhor ponto na direção  $\mathbf{p}_k$  é um mínimo da função  $f$  ao longo desta direção (asterisco magenta), o ponto  $\mathbf{x}_{k+1}$ . O resultado da busca linear é o valor  $\alpha_k = 1/4$ , resultando no ponto  $\mathbf{x}_{k+1} = [-3/8 \ -3/8]^\top$  que tem o valor  $f(\mathbf{x}_{k+1}) = -0.5625 > f(\mathbf{x}^*) = -1$ . ■

Como o valor ótimo  $\alpha_k = 1/4$  foi determinado? Já temos as ferramentas para calcular deterministicamente aquele argumento de uma função unidimensional que minimiza uma função quadrática. A condição necessária é uma derivada nula desta função unidimensional, ou seja

$$\ell'(\alpha) = \frac{d\ell(\alpha)}{d\alpha} = \frac{df(\mathbf{x} + \alpha\mathbf{p})}{d\alpha} = 0. \quad (3.32)$$

Pela (eq. 3.23) sabemos que o gradiente da função quadrática é  $\nabla f(\mathbf{x}) = A\mathbf{x} + \mathbf{b}$ . Para uma função

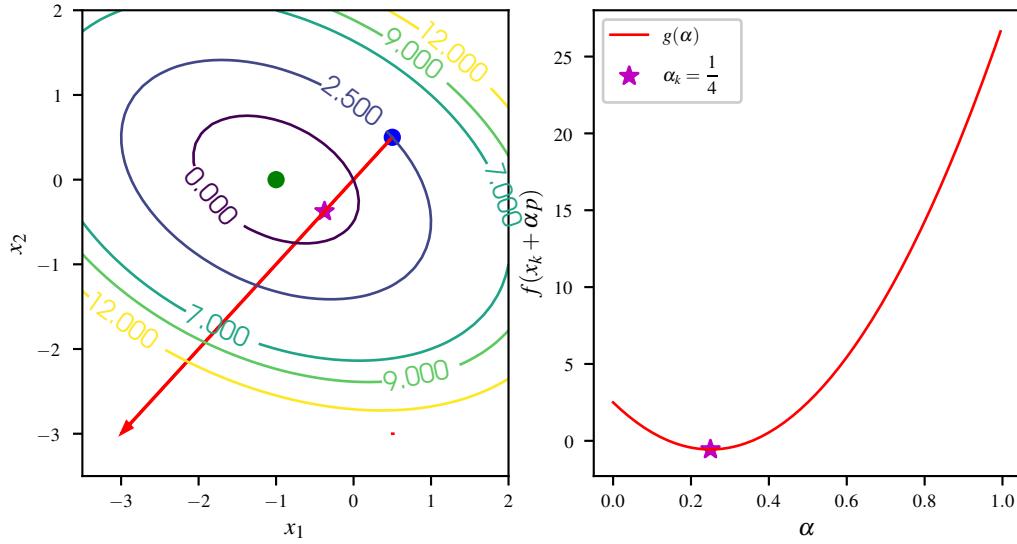


Figura 3.7: Busca Linear

objetivo quadrática  $f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T A \mathbf{x} + \mathbf{b}^T \mathbf{x} + c$  da (eq. 3.19) temos então

$$\begin{aligned}
 \ell'(\alpha) &= d \left[ \frac{1}{2} (\mathbf{x} + \alpha \mathbf{p})^T A (\mathbf{x} + \alpha \mathbf{p}) + \mathbf{b}^T (\mathbf{x} + \alpha \mathbf{p}) + c \right] / d\alpha \\
 &\stackrel{(2.59)}{=} \frac{1}{2} d \left[ (\mathbf{x}^T A + \alpha \mathbf{p}^T A) (\mathbf{x} + \alpha \mathbf{p}) \right] / d\alpha + d \left[ \mathbf{b}^T \mathbf{x} + \alpha \mathbf{b}^T \mathbf{p} \right] / d\alpha + dc/d\alpha \\
 &\stackrel{(2.59b)}{=} \frac{1}{2} d \left[ \mathbf{x}^T A \mathbf{x} + \alpha \mathbf{x}^T A \mathbf{p} + \alpha \mathbf{p}^T A \mathbf{x} + \alpha^2 \mathbf{p}^T A \mathbf{p} \right] / d\alpha + \mathbf{b}^T \mathbf{p} \\
 &= \frac{1}{2} \mathbf{x}^T A \mathbf{p} + \frac{1}{2} \mathbf{p}^T A \mathbf{x} + \alpha \mathbf{p}^T A \mathbf{p} + \mathbf{b}^T \mathbf{p} \\
 &\stackrel{(2.59c)}{=} \mathbf{x}^T A \mathbf{p} + \alpha \mathbf{p}^T A \mathbf{p} + \mathbf{b}^T \mathbf{p} \\
 &= \alpha \mathbf{p}^T A \mathbf{p} + (\mathbf{x}^T A + \mathbf{b}^T) \mathbf{p} \\
 &\stackrel{(3.23)}{=} \alpha \mathbf{p}^T A \mathbf{p} + \nabla f(\mathbf{x}) \mathbf{p}
 \end{aligned} \tag{3.33}$$

Zerando o gradiente  $\ell'(\alpha) = 0$  temos então da (eq. 3.34)

$$\alpha_k = -\frac{\nabla f(\mathbf{x}) \mathbf{p}}{\mathbf{p}^T A \mathbf{p}}. \tag{3.35}$$

Se usarmos a descida do máximo declive, a direção de descida coincide com o gradiente da função objetivo, ou seja  $\mathbf{p} = -\nabla f(\mathbf{x})^T$ , então a (eq. 3.34) especializa para

$$\alpha_k = \frac{\mathbf{p}^T \mathbf{p}}{\mathbf{p}^T A \mathbf{p}}. \tag{3.36}$$

Uma observação interessante é o caso da matriz  $A$  ser a matriz de identidade  $I$ . Nesse caso temos  $\alpha_k = 1$ . Em um espaço bidimensional, as linhas de contorno neste caso são círculos concêntricos, é o valor mínimo da função é o centro destes círculos com o valor  $\mathbf{x}^* = -\mathbf{b}$ , veja a (eq. 3.23).

**Exemplo 3.12** Vamos aplicar a estratégia de minimizar a função da (eq. 3.36) três vezes consecutivas a partir de um ponto inicial, usando a função quadrática de costume do exemplo 3.6. Temos então a sequência  $\mathbf{x}_0 \rightarrow \mathbf{x}_1 \rightarrow \mathbf{x}_2 \rightarrow \mathbf{x}_3$ , ou seja  $\mathbf{x}_0 \rightarrow (\mathbf{x}_0 + \alpha_0 \mathbf{p}_0) \rightarrow (\mathbf{x}_1 + \alpha_1 \mathbf{p}_1) \rightarrow (\mathbf{x}_2 + \alpha_2 \mathbf{p}_2)$ . Usando como direção de descida o máxímo declive  $\mathbf{p}_k = -\nabla f(\mathbf{x}_k)^\top$ , temos  $\mathbf{x}_0 \rightarrow (\mathbf{x}_0 - \alpha_0 \nabla f(\mathbf{x}_0)^\top) \rightarrow (\mathbf{x}_1 - \alpha_1 \nabla f(\mathbf{x}_1)^\top) \rightarrow (\mathbf{x}_2 - \alpha_2 \nabla f(\mathbf{x}_2)^\top)$ . Seja o valor inicial é  $\mathbf{x}_0 = [3/2 \quad 1/2]^\top$ . Então, segundo (eq. 3.36) temos

$$\begin{aligned}\nabla f(\mathbf{x}_0) &= [11/2 \quad 9/2] \\ \alpha_0 &= \frac{\mathbf{p}_0^\top \mathbf{p}_0}{\mathbf{p}_0^\top A \mathbf{p}_0} = \frac{-\nabla f(\mathbf{x}_0)(-\nabla f(\mathbf{x}_0)^\top)}{-\nabla f(\mathbf{x}_0)^\top A(-\nabla f(\mathbf{x}_0))} = \frac{\|\nabla f(\mathbf{x}_0)\|^2}{\nabla f(\mathbf{x}_0)^\top A \nabla f(\mathbf{x}_0)} \\ &= \frac{121/4 + 81/4}{[11/2 \quad 9/2] \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix} \begin{bmatrix} 11/2 \\ 9/2 \end{bmatrix}} = 0.264, \quad \mathbf{x}_1 = \begin{bmatrix} 0.046 \\ -0.690 \end{bmatrix}.\end{aligned}$$

$$\text{Semelhantemente, } \alpha_1 = 0.451, \quad \mathbf{x}_2 = \begin{bmatrix} -0.586 \\ 0.083 \end{bmatrix}, \quad \alpha_2 = 0.264, \quad \mathbf{x}_3 = \begin{bmatrix} -0.827 \\ -0.114 \end{bmatrix}$$

■

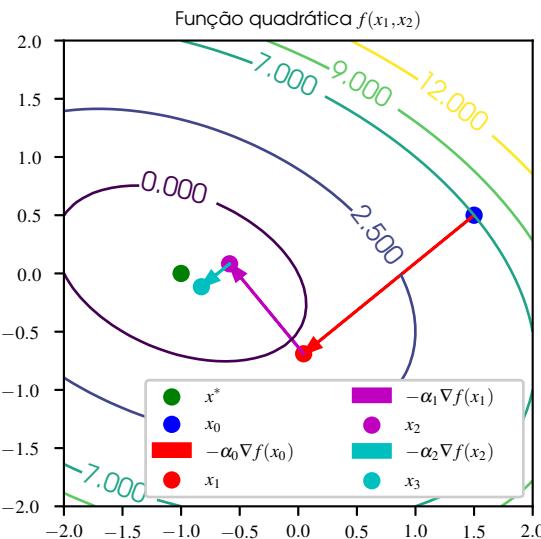


Figura 3.8: Busca Linear. Três passos consecutivos a partir do ponto inicial  $\mathbf{x}_0$  levam até o ponto  $\mathbf{x}_3$ .

A análise visual da figura 3.8 revela que após três iterações, o ponto atual  $\mathbf{x}_3$  não conseguiu alcançar o mínimo  $\mathbf{x}^*$  (ponto verde). Já constatamos que a trajetória é um caminho zigue-zague que neste caso levará aproximadamente ao mínimo após muitas iterações. Além disso podemos levantar a suspeita que duas direções de descida consecutivas são ortogonais uma a outra.

**Corolário 3.3.4** Duas direções consecutivas  $\mathbf{p}_k$  e  $\mathbf{p}_{k+1}$  no algoritmo de descida de gradiente pelo máxímo declive são ortogonais uma a outra  $\mathbf{p}_k \perp \mathbf{p}_{k+1}$ , ou seja

$$\mathbf{p}_k \cdot \mathbf{p}_{k+1} = 0. \tag{3.37}$$

*Demonstração.* Pela (eq. 3.34) temos como condição necessária  $\ell'(\alpha) = 0$  do valor ótimo de  $\alpha$ , com  $\ell(\alpha) = f(\mathbf{x} + \alpha\mathbf{p})$  e  $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$ . Para conduzir a prova, temos que usar um pequeno truque, decompondo a condição necessária  $\ell'(\alpha_k) = 0$  para o ponto atual  $\mathbf{x}_k$ , enfiando um termo multiplicativo neutro  $\partial \mathbf{x}_{k+1} / \partial \mathbf{x}_{k+1} = 1$ , usando a regra da cadeia da definição 2.1.8 na pág. 42. Então

$$\begin{aligned}\ell'(\alpha_k) &= \frac{d\ell(\alpha_k)}{d\alpha_k} = \frac{\partial \ell(\alpha_k)}{\partial \mathbf{x}_{k+1}} \cdot \frac{\partial \mathbf{x}_{k+1}}{\partial \alpha_k} \\ &= \frac{\partial f(\mathbf{x}_k + \alpha_k \mathbf{p}_k)}{\partial \mathbf{x}_{k+1}} \cdot \frac{\partial (\mathbf{x}_k + \alpha_k \mathbf{p}_k)}{\partial \alpha_k} \\ &= \frac{\partial f(\mathbf{x}_{k+1})}{\partial \mathbf{x}_{k+1}} \cdot \mathbf{p}_k = \nabla f(\mathbf{x}_{k+1})^\top \cdot \mathbf{p}_k = \mathbf{0} \cdot \mathbf{p}_k = 0,\end{aligned}$$

pois  $\nabla f(\mathbf{x}_{k+1}) = \mathbf{0}^\top$  é condição necessária para que  $\alpha_{k+1}$  seja ótimo na vizinhança de  $\mathbf{x}_{k+1}$ , [78]. ■

**Obs.:** Esta condição de ortogonalidade somente é válida para uma função quadrática, definida na (eq. 3.19), ou seja, um problema linear de otimização, se o erro quadrático é usado como função objetivo.

Podemos ainda constatar que a taxa de aprendizagem  $\alpha_k$  em cada passo do algoritmo iterativo de descida de gradiente *não* é fixa no caso estudado, mas adaptável às condições atuais do ponto  $\mathbf{x}_k$ . Esse fato vale para todos os algoritmos de aprendizado de máquina baseados na descida de gradiente que são usados em aplicações mais sofisticadas. Como, para funções altamente não lineares, não existem soluções exatas, abre-se um campo muito vasto para *heurísticas*, métodos que imprecisamente buscam a solução ótima.

### 3.3.4 Descida pelo Gradiente Conjugado

Já constatamos que a minimização da função quadrática  $\frac{1}{2}\mathbf{x}^\top A\mathbf{x} + \mathbf{b}^\top \mathbf{x} + c$  precisa zerar seu gradiente  $\nabla f(\mathbf{x})$ , o que equivale a solução do sistema linear  $A\mathbf{x} = -\mathbf{b}$ . Por essa razão o algoritmo do gradiente conjugado (*Conjugate Gradient*, “CG”) para achar o mínimo da função objetivo quadrática é chamado gradiente conjugado *linear*. De fato, as origens do CG são pesquisas relacionadas à solução de sistemas lineares de grande porte [49]. A apresentação do CG linear então parte do princípio que a matriz  $A$  e o vetor  $\mathbf{b}$  estão disponíveis. Mais tarde, o CG *não linear* para aplicações reais é apresentado. Esse método já não requer a definição da matriz e do vetor, mas gera uma aproximação dos mesmos implicitamente.

#### Gradiente Conjugado Linear

Para economizar espaço, vamos introduzir uma nomenclatura mais compacta para o gradiente no ponto atual da busca  $\mathbf{x}_k$  que merece uma definição própria.

**Definição 3.3.6 — Gradiente da Função Objetivo no Ponto Atual.**

$$\mathbf{g}_k \stackrel{\text{def}}{=} \nabla f(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_k} \tag{3.38}$$

Para simplificar, o sinal negativo do vetor  $\mathbf{b}$  será invertido, portanto, o CG linear quer resolver o sistema linear

$$A\mathbf{x} = \mathbf{b}$$

iterativamente, começando pelo valor inicial  $\mathbf{x}_0$ , e gerado a sequência

$$\mathbf{x}_0 \rightarrow \mathbf{x}_1 \rightarrow \dots \rightarrow \mathbf{x}_k \rightarrow \mathbf{x}_{k+1} \dots$$

Idealmente, a solução atual  $\mathbf{x}_k$  satisfaz a equação do sistema  $A\mathbf{x}_k = \mathbf{b}$ . Qualquer desvio pode ser quantificado pelo vetor de *resíduo* (e/ou por sua norma). Este vetor, pela (eq. 3.23), equivale ao gradiente da função quadrática  $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top A\mathbf{x} - \mathbf{b}^\top \mathbf{x} + c$  (com sinal de  $\mathbf{b}$  invertido), reveja as regras da derivada multidimensional, (eq. 2.53) na pág. 48.

**Definição 3.3.7 — Resíduo de Sistema Linear.**

$$\mathbf{r}_k = A\mathbf{x}_k - \mathbf{b} = \nabla f(\mathbf{x})^\top |_{\mathbf{x}=\mathbf{x}_k} = \mathbf{g}_k^\top. \quad (3.39)$$

Já sabemos que dois vetores  $\mathbf{p}_i$  e  $\mathbf{p}_j$  são ortogonais, se o seu produto escalar é zero, ou seja,  $\mathbf{p}_i \cdot \mathbf{p}_j = 0$ . Inserindo a matriz de identidade  $I$  nesse produto, temos  $\mathbf{p}_i \cdot I \cdot \mathbf{p}_j = 0$ . Agora podemos dizer que os dois vetores são *conjugados* em relação à  $I$ . Esse fato é generalizável para qualquer matriz, desde que ela seja simétrica e positiva definida.

**Definição 3.3.8 — Vetores Conjugados.** Dois vetores  $\mathbf{u}$  e  $\mathbf{v}$  não nulos são conjugados em relação à matriz simétrica e positiva definida  $A$ , se

$$\mathbf{u} \cdot A \cdot \mathbf{v} = 0 \iff \mathbf{u}^\top A \mathbf{v} = 0. \quad (3.40)$$

Um resultado colateral é que  $A\mathbf{u}$  é ortogonal a  $\mathbf{v}$ , e que  $A\mathbf{v}$  é ortogonal a  $\mathbf{u}$ , reveja as regras (eq. 2.59) na pág. 51.

A descida de gradiente pelo máximo declive gera direções que são mutuamente ortogonais, veja o corolário 3.3.4, e a figura 3.8. O resultado foi um caminho zigue-zague que somente com muito atraso leva ao mínimo da função, para ser exato, leva 38 iterações no exemplo 3.12 para que  $\mathbf{x}_k = \mathbf{x}_{38}$  seja numericamente idêntico ao mínimo  $\mathbf{x}^*$ , mesmo com o passo  $\alpha_k$  otimizado. A descida de gradiente pelo gradiente conjugado vai diretamente ao mínimo, um passo por dimensão no máximo. Isso significa que no exemplo ilustrativo com dimensão  $m = 2$ , o CG em dois passos chega em  $\mathbf{x}^*$ . As direções de descida pelo CG linear formam uma sequência de vetores  $\{\mathbf{p}_0, \mathbf{p}_1, \dots, \mathbf{p}_k, \mathbf{p}_{k+1}, \dots\}$  que são mutuamente conjugados um ao outro em relação à  $A$ , ou seja,

$$\mathbf{p}_i^\top A \mathbf{p}_j = 0, \quad \forall i, j, \quad i \neq j. \quad (3.41)$$

Essa afirmação será sem prova para não aumentar o escopo do texto exageradamente. Pormenores podem ser consultados em [78]. Além disso, na mesma referência, encontra-se a prova da ortogonalidade do resíduo atual com todas as direções de descida anteriores.

$$\mathbf{r}_k^\top \mathbf{p}_i = 0, \quad i = 0, \dots, k-1 \quad (3.42)$$

Para essa prova, usa-se o princípio da indução matemática [60], começando pelo fato que a primeira direção é o gradiente negativo,  $\mathbf{p}_0 = -\mathbf{g}_0^\top$  e a hipótese de indução  $\mathbf{r}_{k-1}^\top \mathbf{p}_i = 0, i = 0, \dots, k-2$ .

**Definição 3.3.9 — Gradiente Conjugado Linear Determinístico.** Seja dado o ponto inicial  $\mathbf{x}_0 \in \mathbb{R}^m$  em um espaço vetorial de dimensão  $m$ , e sejam dadas  $m$  direções conjugadas  $\{\mathbf{p}_0, \mathbf{p}_1, \dots, \mathbf{p}_{m-1}\}$ , ou seja,  $\mathbf{p}_i^\top A \mathbf{p}_j = 0, i, j = 0, \dots, m-1, i \neq j$ . Então o algoritmo do gradiente conjugado gera uma sequência determinística  $\{\mathbf{x}_0 \rightarrow \dots \rightarrow \mathbf{x}_k \rightarrow \mathbf{x}_{k+1} \rightarrow \dots \rightarrow \mathbf{x}_m\}$ , com

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k, \quad (3.43)$$

que em  $m$  passos atinge o mínimo  $\mathbf{x}^*$  da função quadrática  $f(\mathbf{x})$ . O valor  $\alpha_k$  é o valor ótimo da busca linear da (eq. 3.35)

$$\alpha_k = -\frac{\mathbf{g}_k^\top \mathbf{p}_k}{\mathbf{p}_k^\top A \mathbf{p}_k}. \quad (3.44)$$

A direção de busca atual  $\mathbf{p}_k$  é uma combinação linear do resíduo atual negativo  $-\mathbf{r}_k$  da (eq. 3.39) (equivalentemente do gradiente negativo  $-\mathbf{g}_k$ ), e da direção de busca anterior  $\mathbf{p}_{k-1}$ . Uma exceção é a primeira direção de busca  $\mathbf{p}_0$  que é iniciada pelo gradiente negativo  $-\mathbf{g}_0$  no ponto inicial. Então

$$\mathbf{p}_k = \begin{cases} -\mathbf{r}_k + \beta_k \mathbf{p}_{k-1}, & \text{se } k \geq 1 \\ -\mathbf{g}_0, & \text{se } k = 0 \end{cases}. \quad (3.45)$$

O escalar  $\beta_k$  controla a contribuição da direção anterior  $\mathbf{p}_{k-1}$  a direção atual  $\mathbf{p}_k$ . Lembre que o gradiente  $\mathbf{g}$  é um vetor de linha  $1 \times m$  e  $\mathbf{p}$  é um vetor de coluna  $m \times 1$ .

O escalar  $\beta_k$  pode ser obtido da seguinte maneira: Pré-multiplicando na definição 3.3.9 da direção atual  $\mathbf{p}_k$  os dois lados da equação pela expressão  $\mathbf{p}_{k-1}^\top A$ , temos

$$\mathbf{p}_{k-1}^\top A \mathbf{p}_k = \mathbf{p}_{k-1}^\top A \mathbf{r}_k + \beta_k \mathbf{p}_{k-1}^\top A \mathbf{p}_{k-1}.$$

Com a proposição que todas as direções são mutuamente conjugados, temos do lado esquerdo da equação  $\mathbf{p}_{k-1}^\top A \mathbf{p}_k = 0$ . Então, resolvendo para a incógnita  $\beta_k$ , temos

$$\beta_k = \frac{\mathbf{r}_k^\top A \mathbf{p}_{k-1}}{\mathbf{p}_{k-1}^\top A \mathbf{p}_{k-1}}. \quad (3.46)$$

**■ Exemplo 3.13** Vamos comparar as trajetórias do gradiente conjugado e do máximo declive. No exemplo 3.12 já simulamos a descida de gradiente simples pelo gradiente negativo, partindo do ponto  $\mathbf{x}_0 = [3/2 \ 1/2]^\top$ , passando por  $\mathbf{x}_1 = [0.046 \ -0.690]^\top$ , e  $\mathbf{x}_2 = [-0.586 \ 0.083]^\top$ . Usando o CG, temos pela (eq. 3.43), (eq. 3.44) e (eq. 3.46)

Calculando  $\mathbf{x}_1$ :

$$\mathbf{p}_0 = -\mathbf{g}_0^\top = -[11/2 \ 9/2]^\top, \quad \alpha_0 = -\frac{\mathbf{g}_0^\top \mathbf{p}_0}{\mathbf{p}_0^\top A \mathbf{p}_0} = \frac{\mathbf{g}_0^\top \mathbf{g}_0}{\mathbf{g}_0^\top A \mathbf{g}_0} = \frac{\|\mathbf{g}_0\|^2}{\mathbf{g}_0^\top A \mathbf{g}_0} = \frac{50.5}{191} = 0.264$$

$$\mathbf{x}_1 = \mathbf{x}_0 + \alpha_0 \mathbf{p}_0 = [3/2 \ 1/2]^\top - 0.264 [11/2 \ 9/2]^\top = [0.046 \ -0.690]^\top$$

Calculando  $\mathbf{x}_2$ :

$$\mathbf{r}_1 = \mathbf{g}_1^\top = [1.402 \ -1.713]^\top, \quad \alpha_1 = -\frac{\mathbf{g}_1^\top \mathbf{p}_1}{\mathbf{p}_1^\top A \mathbf{p}_1} = \frac{4.901}{9.070} = 0.540$$

$$\beta_1 = \frac{\mathbf{r}_1^\top A \mathbf{p}_0}{\mathbf{p}_0^\top A \mathbf{p}_0} = 0.097$$

$$\mathbf{p}_1 = -\mathbf{r}_1 + \beta_1 \mathbf{p}_0 = [-1.402 \ 1.713]^\top + 0.097 [11/2 \ 9/2]^\top = [-1.936 \ 1.277]^\top$$

$$\mathbf{x}_2 = \mathbf{x}_1 + \alpha_1 \mathbf{p}_1 = [0.046 \ -0.690]^\top + 0.540 [-1.936 \ 1.277]^\top = [-1 \ 0]^\top.$$

O resultado sugere numericamente que o CG chegou no ponto ótimo  $\mathbf{x}^*$  após duas iterações. ■

A figura 3.9 mostra a comparação do método pelo máximo declive (“MD”) e do gradiente conjugado (“CG”). Iniciando a busca no ponto  $\mathbf{x}_0 = [3/2 \quad 1/2]^\top$  (ponto azul), os dois métodos chegam no mesmo ponto  $\mathbf{x}_1^{\text{MD}} = \mathbf{x}_1^{\text{CG}} = \mathbf{x}_0 + \alpha_0 \mathbf{p}_0$ , pois no passo inicial a direção de descida anterior ainda não está disponível. O MD chega no segundo  $\mathbf{x}_2^{\text{MD}}$  e terceiro ponto  $\mathbf{x}_3^{\text{MD}}$  pelas regras da busca linear (eq. 3.30) e (eq. 3.36), onde duas direções são ortogonais. Já que o CG no segundo passo chega no ponto ótimo  $\mathbf{x}^*$ . A segunda direção  $\mathbf{p}_1 = -\mathbf{r}_1 + \beta_1 \mathbf{p}_0$  é a combinação linear do gradiente negativo  $-\mathbf{g}_1 = -\mathbf{r}_1$  no ponto  $\mathbf{x}_1$  e da direção anterior  $\mathbf{p}_0$ .

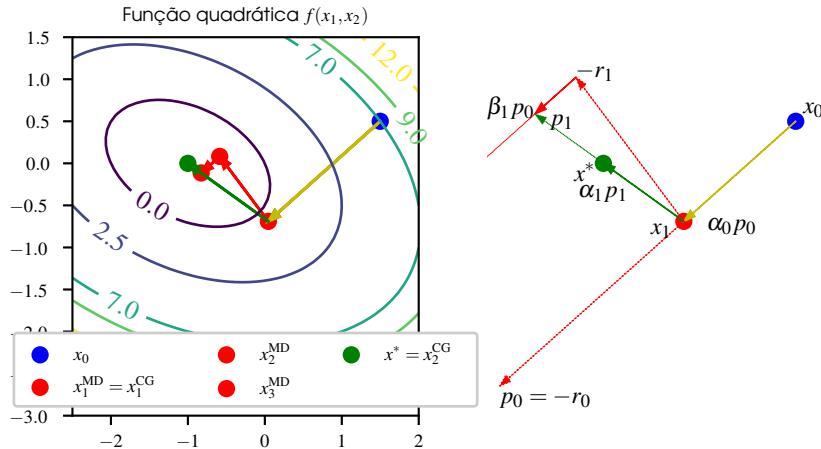


Figura 3.9: Comparação da descida de gradiente pelo método do máximo declive (“MD”) e do gradiente conjugado (“CG”).

Antes de formalizar o algoritmo de CG, vamos ainda otimizar algumas expressões. Precisamos ainda de uma identidade que pode ser obtido por subtrair dois resíduos (eq. 3.30) consecutivos, e da definição da atualização do ponto atual (eq. 3.39). Temos

$$\begin{aligned} \mathbf{r}_{k+1} - \mathbf{r}_k &\stackrel{3.39}{=} A\mathbf{x}_{k+1} - \mathbf{b} - (A\mathbf{x}_k - \mathbf{b}) \\ &= A(\mathbf{x}_{k+1} - \mathbf{x}_k) \stackrel{3.30}{=} A(\mathbf{x}_k + \alpha_k \mathbf{p}_k - \mathbf{x}_k) = \alpha_k A \mathbf{p}_k \\ \iff \mathbf{r}_{k+1} &= \mathbf{r}_k + \alpha_k A \mathbf{p}_k. \end{aligned} \quad (3.47)$$

Temos

$$\begin{aligned} -\mathbf{r}_k^\top \mathbf{p}_k &\stackrel{3.45}{=} -\mathbf{r}_k^\top [-\mathbf{r}_k + \beta_k \mathbf{p}_{k-1}] = \mathbf{r}_k^\top \mathbf{r}_k - \beta_k \mathbf{r}_k^\top \mathbf{p}_{k-1} \\ &\stackrel{3.42}{=} \mathbf{r}_k^\top \mathbf{r}_k \end{aligned} \quad (3.48)$$

Considerando a definição do passo ótimo na busca linear (eq. 3.35) e do residual (eq. 3.39), junto com a identidade (eq. 3.48), podemos reformular o passo na busca linear como

$$\alpha_k = \frac{\mathbf{r}_k^\top \mathbf{r}_k}{\mathbf{p}_k^\top A \mathbf{p}_k}. \quad (3.49)$$

A próxima reformulação é do coeficiente  $\beta$  que é usado para quantificar a contribuição da direção anterior na direção atual, veja a (eq. 3.45). Temos de (eq. 3.47), sendo  $\alpha_k$  um escalar,

$$A \mathbf{p}_k = \frac{\mathbf{r}_{k+1} - \mathbf{r}_k}{\alpha_k}. \quad (3.50)$$

Incrementando o índice de  $k$  para  $k + 1$  em (eq. 3.46), temos

$$\begin{aligned}
 \beta_{k+1} &= \frac{\mathbf{r}_{k+1}^\top A \mathbf{p}_k}{\mathbf{p}_k^\top A \mathbf{p}_k} \stackrel{3.50}{=} \frac{\frac{1}{\alpha_k} \mathbf{r}_{k+1}^\top [\mathbf{r}_{k+1} - \mathbf{r}_k]}{\frac{1}{\alpha_k} \mathbf{p}_k^\top [\mathbf{r}_{k+1} - \mathbf{r}_k]} \\
 &= \frac{\mathbf{r}_{k+1}^\top \mathbf{r}_{k+1} - \mathbf{r}_{k+1}^\top \mathbf{r}_k}{\mathbf{p}_k^\top \mathbf{r}_{k+1} - \mathbf{p}_k^\top \mathbf{r}_k} \stackrel{2.59C}{=} \frac{\mathbf{r}_{k+1}^\top \mathbf{r}_{k+1} - \mathbf{r}_{k+1}^\top \mathbf{r}_k}{\mathbf{r}_{k+1}^\top \mathbf{p}_k + \mathbf{r}_k^\top \mathbf{r}_k} \stackrel{3.37}{=} \frac{\mathbf{r}_{k+1}^\top \mathbf{r}_{k+1} - 0}{0 + \mathbf{r}_k^\top \mathbf{r}_k} \\
 &= \frac{\mathbf{r}_{k+1}^\top \mathbf{r}_{k+1}}{\mathbf{r}_k^\top \mathbf{r}_k}.
 \end{aligned} \tag{3.51}$$

Podemos formular o algoritmo do gradiente conjugado na sua forma padrão.

---

#### Algoritmo 7: Descida pelo Gradiente Conjugado

---

**Entrada:** Dados: Matriz  $A$  simétrica  $m \times m$ , definida positiva; vetor de coluna  $\mathbf{b}$  de dimensão  $m \times 1$ ; valor inicial  $\mathbf{x}_0$  de busca; número máximo de iterações  $k_{\max}$ ;

**Resultado:** Ponto  $\mathbf{x}^*$

```

1  $k \leftarrow 0$ ;
2  $\mathbf{r}_0 \leftarrow A\mathbf{x}_0 - \mathbf{b}$  /* Resíduo inicial */;
3  $\mathbf{p}_0 \leftarrow -\mathbf{r}_0$  /* Direção de descida inicial */;
4 enquanto ( $k < k_{\max}$ ) E ( $\|\mathbf{r}_k\| > 0$ ) faça
5    $\alpha_k \leftarrow \frac{\mathbf{r}_k^\top \mathbf{r}_k}{\mathbf{p}_k^\top A \mathbf{p}_k}$  /* (eq. 3.49)* */;
6    $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha_k \mathbf{p}_k$  /* (eq. 3.43)* */;
7    $\mathbf{r}_{k+1} \leftarrow \mathbf{r}_k + \alpha_k A \mathbf{p}_k$  /* (eq. 3.47)* */;
8   se ( $\|\mathbf{r}_{k+1}\| > 0$ ) então
9      $\beta_{k+1} \leftarrow \frac{\mathbf{r}_{k+1}^\top \mathbf{r}_{k+1}}{\mathbf{r}_k^\top \mathbf{r}_k}$  /* (eq. 3.51)* */;
10     $\mathbf{p}_{k+1} \leftarrow -\mathbf{r}_k + \beta_{k+1} \mathbf{p}_k$  /* (eq. 3.45)* */;
11     $k \leftarrow k + 1$ ;
12  fim
13 fim
14  $\mathbf{x}^* \leftarrow \mathbf{x}_{k+1}$ 

```

---

### 3.3.5 Gradiente Conjugado Não Linear

E se não tivéssemos a matriz  $A$  e o vetor  $\mathbf{b}$  de uma função quadrática  $\frac{1}{2}\mathbf{x}^\top A\mathbf{x} + \mathbf{b}^\top \mathbf{x} + c$ ? Esse é o cenário realista de uma aplicação séria de aprendizado de máquina. A seção anterior serve para apresentar a teoria do gradiente conjugado, porém não tem utilidade prática em geral. Na realidade a função  $f(\mathbf{x})$  a ser minimizada é altamente não linear. Temos que sair do domínio de problemas de otimização que permitem uma solução bem traçada. Agora vale tudo para a natureza não linear da função a ser minimizada. Pode se afirmar que não existe mais um algoritmo “Faz Tudo” aplicável a qualquer problema. Chegou a hora das heurísticas que apresentam soluções plausíveis. É claro que no estudo do gradiente conjugado surgiu a questão como estender o método a funções em geral. O que mudou em relação ao algoritmo do CG da função quadrática é

1. Em geral existe somente um mínimo global que não pode ser descoberto deterministicamente;
2. Existem vários mínimos locais onde o algoritmo pode ficar preso;
3. A busca linear a partir do ponto atual  $\mathbf{x}_k$  não dispõe mais de um passo  $\alpha_k$  ótimo que possa ser calculado deterministicamente, com na (eq. 3.44) na pág. 107;

4. A contribuição  $\beta_k$  da direção de busca anterior  $\mathbf{p}_{k-1}$  não dispõe mais do cálculo ótimo da (eq. 3.46) na pág. 107;
5. Durante o processo de otimização para achar um mínimo da função, a direção atual calculada  $\mathbf{p}_k$  até pode apontar na direção de *subida* de função;

A consequência dessas dificuldades é um aumento da complexidade do algoritmo. Se a direção atual aponta para onde a função vai aumentar, recomenda-se uma reinicialização da busca, atribuindo zero ao parâmetro  $\beta_k$  atual. Isso pode ser feito também periodicamente, após um certo número de iterações. A busca linear tem que ser por si só um sub-algoritmo para achar o mínimo ao longo da direção atual. Uma técnica é tentar as raízes da derivada da função unidimensional associada à busca linear. Pré-condicionamento pode melhorar o comportamento de convergência do método do CG. Apresentam-se aqui três modificações clássicas do parâmetro  $\beta$ . A análise de cada uma dessas heurísticas em termos de taxa de convergência, pontos fracos e comparações experimentais iria expandir demais este texto. Para um excelente estudo de técnicas do CG não linear aponta-se para [78].

$$\beta_{k+1}^{\text{HS}} = \frac{\mathbf{g}_{k+1}(\mathbf{g}_{k+1}^\top - \mathbf{g}_k^\top)}{(\mathbf{g}_{k+1} - \mathbf{g}_k) \mathbf{p}_k} \quad \text{Hestenes-Stiefel [49]} \quad (3.52\text{a})$$

$$\beta_{k+1}^{\text{FR}} = \frac{\mathbf{g}_{k+1}^\top \mathbf{g}_{k+1}}{\mathbf{g}_k^\top \mathbf{g}_k} = \frac{\|\mathbf{g}_{k+1}\|^2}{\|\mathbf{g}_k\|^2} \quad \text{Fletcher-Reeves [37]} \quad (3.52\text{b})$$

$$\beta_{k+1}^{\text{PR}} = \frac{\mathbf{g}_{k+1}(\mathbf{g}_{k+1}^\top - \mathbf{g}_k^\top)}{\mathbf{g}_k^\top \mathbf{g}_k} \quad \text{Polak-Ribière [80]} \quad (3.52\text{c})$$

Se a função objetivo for a função quadrática  $\frac{1}{2}\mathbf{x}^\top \mathbf{A}\mathbf{x} + \mathbf{b}^\top \mathbf{x} + c$ , os métodos de Fletcher-Reeves e Polak-Ribière coincidem, pois segundo o corolário 3.3.4, nesse caso dois gradientes consecutivos são ortogonais, ou seja,  $\mathbf{g}_{k+1}\mathbf{g}_{k+1}^\top = 0$ .

A técnica do gradiente conjugado para minimizar uma função não linear é apresentado no algoritmo 8. A função “Busca\_Linear” no algoritmo 8 tem que fornecer o tamanho do passo  $\alpha_k$  que

---

**Algoritmo 8:** Descida pelo Gradiente Conjugado, Função Objetivo Não Linear

---

**Entrada :** Dados: função escalar com argumento vetorial  $f(\mathbf{x})$ ; função  $\mathbf{g}(\mathbf{x})$  para calcular ou estimar o gradiente da função no ponto atual  $\mathbf{x}_k$ ; valor inicial  $\mathbf{x}_0$  de busca; número máximo de iterações  $k_{\max}$ ;

**Resultado:** Ponto  $\mathbf{x}^*$

```

1  $k \leftarrow 0$ ;
2  $\mathbf{g}_0 \leftarrow \mathbf{g}(\mathbf{x}_0)$  /* Resíduo inicial */;
3  $\mathbf{p}_0 \leftarrow -\mathbf{g}_0^\top$  /* Direção de descida inicial */;
4 enquanto ( $k < k_{\max}$ ) E ( $\|\mathbf{g}_k\| > 0$ ) faça
5    $\alpha_k \leftarrow \text{Busca\_Linear}(\mathbf{p}_k)$  /* (eq. 3.49)* */;
6    $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha_k \mathbf{p}_k$  ;
7   se ( $\|\mathbf{g}_{k+1}\| > 0$ ) então
8      $\beta_{k+1} \leftarrow$  (eq. 3.52a) ou (eq. 3.52b) ou (eq. 3.52c) ;
9      $\mathbf{p}_{k+1} \leftarrow -\mathbf{g}_{k+1} + \beta_{k+1} \mathbf{p}_k$  /* (eq. 3.45)* */;
10     $k \leftarrow k + 1$ ;
11  fim
12 fim
13  $\mathbf{x}^* \leftarrow \mathbf{x}_{k+1}$ 

```

---

garante que a direção de  $\mathbf{p}_k$  realmente desce, ou seja, a condição  $\mathbf{p}_k^\top \nabla f(\mathbf{x}) < 0$  conforme o corolário 3.3.3. A busca linear ramifica para um teoria própria, por isso não será mais pormenorizada aqui. Mais uma vez aponta-se para [78] para aprofundar as técnicas e considerações teóricas da busca linear para funções não lineares.

### 3.4 Heurísticas de Primeira Ordem

Os métodos contemporâneos de aprendizagem profunda [42] são baseadas unicamente na primeira derivada da função objetivo, ou seja no gradiente. A razão é que o uso de técnicas de segunda ordem, que usam a matriz Hessiana ou sua aproximação são computacionalmente proibitivos no caso em que existem muitos padrões de treinamento e muitas características que descrevem um padrão. Normalmente, as heurísticas aqui compiladas se enquadram nas redes neurais artificiais, porém a teoria é apresentável junto com técnicas simples de descida de gradiente. Primeiro, lembramos a nomenclatura do algoritmo 4 da descida de gradiente.

$\mathbf{x}_0 \rightarrow \mathbf{x}_1 \rightarrow \dots \rightarrow \mathbf{x}_k \rightarrow \mathbf{x}_{k+1} \dots$	Sequência de valores gerados
$\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta \mathbf{x}_k$	Diferença $\Delta \mathbf{x}_k$ entre dois valores consecutivos
$\mathbf{g}_k \stackrel{\text{def}}{=} \nabla f(\mathbf{x}) _{\mathbf{x}=\mathbf{x}_k}$	Gradiente da função objetivo no ponto atual $\mathbf{x}_k$ (eq. 2.20)
$g_{k,j} \stackrel{\text{def}}{=} \frac{\partial f(\mathbf{x})}{\partial x_j} _{\mathbf{x}=\mathbf{x}_k}$	$j$ -ésima componente do gradiente da função objetivo no ponto atual $\mathbf{x}_k$
$\mathbf{g}(\tilde{\mathbf{x}}) \stackrel{\text{def}}{=} \nabla f(\mathbf{x}) _{\mathbf{x}=\tilde{\mathbf{x}}}$	Gradiente da função objetivo em um ponto qualquer $\tilde{\mathbf{x}}$

Para facilitar o formalismo, a transposição dos gradientes  $\mathbf{g}^\top$  é omitida. Normalmente isso tem que ser feito, pois os pontos  $\mathbf{x}_k$  no trajeto da descida de gradiente são vetores de coluna e o gradiente é um vetor de linha. Vamos distinguir dois grupos de métodos. O primeiro grupo *não* muda a taxa de aprendizagem  $\alpha$  durante as iterações do algoritmo da descida de gradiente, ou seja

$$\alpha = \text{const.} \quad (3.54)$$

O segundo grupo muda a taxa de aprendizagem em cada uma das  $k$  iterações, ou seja,

$$\alpha_k \neq \text{const.}, \quad k = 0, 1, \dots \quad (3.55)$$

#### 3.4.1 Taxa de Aprendizagem e Demais Hiperparâmetros Constantes

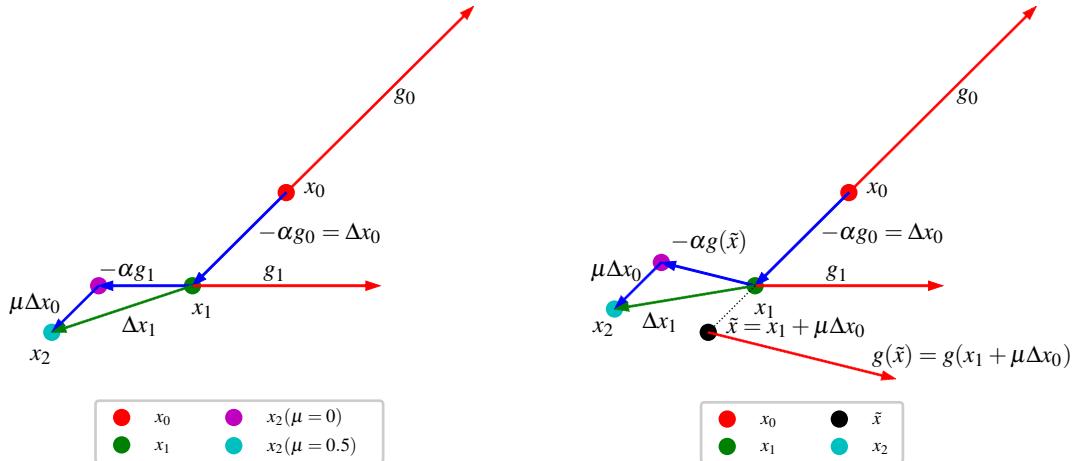
Os métodos que usam uma taxa de aprendizagem constante correm o perigo de andar rápido demais ou lento demais para procurar o ponto ótimo. Rápido demais significa que o próximo ponto  $\mathbf{x}_{k+1}$  causa uma divergência do algoritmo, ou seja, o gradiente cresce constantemente até atingir um valor numericamente infinito, invalidando o processo de otimização. Lento demais significa que a distância  $\mathbf{x}_{k+1} - \mathbf{x}_k$  entre dois valores consecutivos é muito pequena e poderia ser maior sem colocar em risco as propriedades de convergência do algoritmo.

##### Máximo Declive

O método mais simples ajusta o ponto pelo gradiente oposto, ponderando ainda pela taxa de aprendizagem  $\alpha$ .

$$\Delta \mathbf{x}_k = -\alpha \mathbf{g}_k \quad (3.56)$$

Já se ilustrou no exemplo da regressão linear simples que seguir cegamente a direção oposta do gradiente não é uma boa estratégia, veja a seção 3.2.2 na pág. 89.



(a) Descida de gradiente usando momento.

(b) Descida de gradiente usando momento de Nesterov.

Figura 3.10: Descida de gradiente com momento.

### Momento

Este método memoriza a diferença  $\Delta\mathbf{x}_k$  que foi dada na transição entre dois pontos consecutivos. Um hiperparâmetro adicional  $\mu$  tem que ser definido.

$$\Delta\mathbf{x}_k = -\alpha\mathbf{g}_k + \mu\Delta\mathbf{x}_{k-1} \quad (3.57)$$

No primeiro passo de  $\mathbf{x}_0$  para  $\mathbf{x}_1$ , o termo do momento é inicializado com zero, ou seja,  $\Delta\mathbf{x}_{-1} = \mathbf{0}$ . O efeito do termo adicional é uma aceleração do movimento em caso que dois gradientes têm uma certa direção semelhante. A figura 3.10a compara o trajeto  $\mathbf{x}_0 \rightarrow \mathbf{x}_1 \rightarrow \mathbf{x}_2$  com e sem o uso de um termo de momento na (eq. 3.56). No primeiro passo o momento é zero, portanto do ponto inicial  $\mathbf{x}_0$  anda-se na direção oposta do gradiente  $\mathbf{g}_0$  por  $\alpha$ , ou seja  $\Delta\mathbf{x}_0 = -\alpha\mathbf{g}_0$ . Se o segundo passo não usasse o termo do momento, ou seja,  $\mu = 0$ , o ponto seria  $\mathbf{x}_2 = \mathbf{x}_1 - \alpha\mathbf{g}_1$ . Com o termo do momento a diferença  $\Delta\mathbf{x}_1$  é composta pelo caminho sem momento, e o momento anterior  $\mu\Delta\mathbf{x}_0$ , portanto  $\mathbf{x}_2 = \mathbf{x}_1 - \alpha\mathbf{g}_1 + \mu\Delta\mathbf{x}_0$ .

### Momento de Nesterov

A diferença entre o uso do momento da (eq. 3.57) e o momento de Nesterov [75, 76, 77, 106] é em que posição o gradiente é calculado. Enquanto o momento convencional avalia o gradiente na posição  $\mathbf{x}_k$ , o momento de Nesterov tenta fazer uma previsão do comportamento do gradiente no próximo passo, calculando o gradiente em um ponto auxiliar  $\tilde{\mathbf{x}}$ . Este ponto é alcançado a partir do ponto atual  $\mathbf{x}_k$ , somando o momento convencional anterior, ou seja  $\tilde{\mathbf{x}} = \mathbf{x}_k + \mu\Delta\mathbf{x}_{k-1}$ . Neste ponto, calcula-se o gradiente da função objetivo e a contribuição ponderada do valor negativo deste gradiente, ou seja, o vetor  $-\alpha\mathbf{g}(\tilde{\mathbf{x}})$ . Este vetor é somado ao ponto atual  $\mathbf{x}_k$ , mais o momento convencional anterior  $\mu\Delta\mathbf{x}_{k-1}$  para produzir o próximo ponto  $\mathbf{x}_{k+1}$  pela (eq. 3.58).

$$\Delta\mathbf{x}_k = -\alpha\mathbf{g}(\mathbf{x}_k + \mu\Delta\mathbf{x}_{k-1}) + \mu\Delta\mathbf{x}_{k-1} \quad (3.58)$$

A figura 3.10b mostra o trajeto de  $\mathbf{x}_0$ , via  $\mathbf{x}_1$ , até  $\mathbf{x}_2$ . A primeira transição é idêntica ao caso do momento convencional, pois não existe um momento anterior  $\Delta\mathbf{x}_{-1}$ , compare a figura 3.10a. O

que difere é a contribuição do gradiente, pois ele foi calculado em outro lugar  $\tilde{\mathbf{x}}$ . Isso provoca uma posição diferente de  $\mathbf{x}_2$ .

### 3.4.2 Taxa de Aprendizagem e Demais Hiperparâmetros Variáveis

As heurísticas a seguir ajustam os hiperparâmetros constantemente durante o processo de descida de gradiente. Ainda podemos distinguir entre métodos que tenham hiperparâmetros iguais para todos as componentes da função objetivo, e métodos que tenham hiperparâmetros específicos para cada componente.

#### Taxa de Aprendizagem com Diminuição

Uma técnica muito simples para diminuir a taxa de aprendizagem  $\alpha$  durante as iterações de atualização do parâmetros é introduzir um hiperparâmetro em forma de uma taxa de diminuição  $\gamma \geq 0$  e atualizar a taxa de aprendizagem na  $k$ -ésima iteração como

$$\alpha_{k+1} = \alpha_k \frac{1}{1 + \gamma}, \quad (3.59)$$

onde  $\alpha_0$  é a taxa de aprendizagem inicial. No caso  $\gamma = 0$  temos a taxa de aprendizagem fixa. Para  $\gamma = 1$ , ela se reduz pela metade a cada iteração. De propósito esse hiperparâmetro não é chamado “Taxa de decaimento” (*decay rate*), pois nos métodos a seguir existe mais um hiperparâmetro  $\rho$  com este nome que controla o decaimento de contribuições dos gradientes passados.

#### Adagrad

O diferencial principal da heurística Adagrad [28] é um ajuste independente para cada componente  $x_j, j = 1, \dots, m$  do espaço dos argumentos da função objetivo, ou seja, considera-se uma diferença específica  $\Delta x_{k,j}$  na atualização dos parâmetros. Em primeiro lugar, guarda-se o histórico dos gradientes  $\mathbf{g}_0, \mathbf{g}_1, \dots, \mathbf{g}_{k-1}, \mathbf{g}_k$  até o ponto atual  $\mathbf{x}_k$ . Mais precisamente, basta guardar a soma dos quadrados de cada componente  $\mathbf{g}_{\ell,j}^2$  em um vetor acumulador  $\mathbf{a}_k$  de dimensão  $m$

$$a_{k,j} = \sum_{\ell=0}^k g_{\ell,j}^2, \quad j = 1, \dots, m. \quad (3.60)$$

A atualização da  $j$ -ésima componente é simplesmente

$$a_{k+1,j} = a_{k,j} + g_{k+1,j}^2. \quad (3.61)$$

Finalmente a diferença do parâmetro por componente calcula-se como

$$\Delta x_{k,j} = -\frac{\alpha}{\sqrt{\epsilon + a_{k,j}}}, \quad j = 1, \dots, m, \quad (3.62)$$

onde  $\epsilon$  é uma constante de valor pequeno que estabiliza o histórico, caso o gradiente fique pequeno, tipicamente na ordem de  $10^{-7}$ . O processo de minimização da função objetivo  $f(\mathbf{x})$  corre o risco de desacelerar, caso as componentes do acumulador  $\mathbf{a}$  tenham valores altos, porque o denominador da (eq. 3.62) cresce na mesma proporção.

#### Adadelta

A desaceleração do processo de descida de gradiente do Adagrad deve-se ao crescimento constante do denominador, pois os termos  $g_{k+1,j}^2$  são sempre positivos, e assim o acumulador  $a_{k,j}$  cresce também, a função da raiz quadrada é uma função monótona, e  $\epsilon$  é constante. Para evitar essa eliminação assintótica do efeito de ajuste dos parâmetros, o Adadelta [117] introduz um mecanismo de esquecimento, descartando a influência de gradientes mais antigos. Um novo hiperparâmetro  $0 < \rho < 1$  (“rho”) controla quão importante os gradientes passados são no cálculo

desta heurística. Ao contrário da *soma* dos quadrados das componentes como na (eq. 3.60), a *média móvel exponencialmente ponderada*, (EWMA, *exponentially weighted moving average*) é memorizada na  $k$ -ésima iteração como

$$\overline{g_{k,j}^2} = \rho \overline{g_{k-1,j}^2} + (1 - \rho) g_{k,j}^2, \quad \overline{g_{0,j}} = 0, \quad j = 1, \dots, m. \quad (3.63)$$

Um  $\rho$  mais próximo de zero<sup>10</sup> enfatiza as contribuições  $g_{k,j}^2, g_{k-1,j}^2, g_{k-2,j}^2, \dots$  dos gradientes mais recentes. Até este ponto, a diferença do parâmetro por componente calcula-se da mesma maneira como no Adagrad, porém com as médias em vez das somas, ou seja,

$$\Delta x_{k,j} = -\frac{\alpha}{\sqrt{\varepsilon + \overline{g_{k,j}^2}}}, \quad j = 1, \dots, m, \quad (3.64)$$

Além da EWMA dos quadrados das componentes do gradiente, a EWMA dos quadrados  $\Delta x_{k,j}^2$  das próprias diferenças é memorizada como

$$\overline{\Delta x_{k,j}^2} = \rho \overline{\Delta x_{k-1,j}^2} + (1 - \rho) \Delta x_{k,j}^2, \quad \overline{\Delta x_{0,j}} = 0, \quad j = 1, \dots, m. \quad (3.65)$$

O objetivo é a compensação de uma suposta falta de correspondência em relação às unidades da variável  $\mathbf{x}$  e o gradiente  $\mathbf{g}$  da função objetivo. A diferença dos parâmetros baseada nesse ideia é finalmente definida como

$$\Delta x_{k,j} = -\frac{\sqrt{\varepsilon + \overline{\Delta x_{k-1,j}^2}}}{\sqrt{\varepsilon + \overline{g_{k,j}^2}}}, \quad j = 1, \dots, m, \quad (3.66)$$

Pormenores sobre o algoritmo podem ser encontrados em [93, 117].

### RMSProp

Motivado pela mesma necessidade de evitar a degeneração assintótica da aprendizagem dos parâmetros, este método proposto por [50] usa o mesmo mecanismo do EWMA como o Adadelta para destacar a influência dos gradientes mais recentes no processo de otimização. A regra para calcular a diferença dos parâmetros é idêntica a primeira versão do Adadelta (eq. 3.64).

### Adam

O acrônimo deriva-se de *Adaptive Moment Estimation*. O método foi proposto por [64]. Combina os conceitos de momento e do RMSProp/Adagrad. Dois hiperparâmetros  $\beta_1$  e  $\beta_2$  do intervalo  $[0, 1]$  em forma de taxas de decaimento são usados para controlar o algoritmo. Como no caso do Adagrad, Adadelta e RMSprop, a atualização de cada um dos  $m$  parâmetros é considerado independentemente. A diferença individual nos parâmetros é

$$\Delta x_{k,j} = -\frac{\alpha \widehat{m}_{k,j}}{\sqrt{\varepsilon + \widehat{v}_{k,j}}}, \quad j = 1, \dots, m, \quad (3.67)$$

onde

$$\overline{m_{k,j}} = \beta_1 \overline{m_{k-1,j}} + (1 - \beta_1) g_{k,j}, \quad \overline{m_{0,j}} = 0, \quad j = 1, \dots, m. \quad (3.68)$$

armazena os EWMA das componentes do gradiente, e

$$\overline{v_{k,j}} = \beta_2 \overline{v_{k-1,j}} + (1 - \beta_2) g_{k,j}^2, \quad \overline{v_{0,j}} = 0, \quad j = 1, \dots, m. \quad (3.69)$$

---

<sup>10</sup>Na definição comum da EWMA, o coeficiente  $\rho$  pondera o valor atual, e  $(1 - \rho)$  pondera a média móvel. O comunicado original do Adadelta [117] não faz referência ao EWMA, embora conceitualmente a proposta seja idêntica.

o quadrados dos mesmos, incorporando assim alguma informação sobre a variância desses valores. Para eliminar efeitos indesejados de um viés em direção a zero dos valores  $\overline{m_{k,j}}$  e  $\overline{v_{k,j}}$ , eles são corrigidos como

$$\widehat{m}_{k,j} = \frac{\overline{m_{k,j}}}{1 - \beta_1^k} \quad (3.70a)$$

$$\widehat{v}_{k,j} = \frac{\overline{v_{k,j}}}{1 - \beta_2^k}, \quad j = 1, \dots, m. \quad (3.70b)$$

### Adamax

O Adam no cálculo da EWMA do termo  $\widehat{v}_{k,j}$  na (eq. 3.70b) usa a norma Euclidiana (norma- $\ell_2$ ). Pode-se generalizar para a norma- $\ell_p$  como

$$\overline{v_{k,j}} = \beta_2^p \overline{v_{k-1,j}} + (1 - \beta_2^p) g_{k,j}^p.$$

Como valores de  $p$  altos causam instabilidade numérica, desconsiderem-se esses casos, porém para a norma- $\ell_\infty$  (veja a (eq. 1.14) na pág. 14), surge uma expressão simples

$$\begin{aligned} \overline{u_{k,j}} &= \beta_2^p \overline{v_{k-1,j}} + (1 - \beta_2^p) g_{k,j}^p \\ &= \max \{ \beta_2 \overline{u_{k-1,j}}, |g_{k,j}| \}, \quad \overline{u_{0,j}} = 0, \quad j = 1, \dots, m. \end{aligned} \quad (3.71)$$

Segundo os autores do Adagrad essa extensão mostra bons resultados. A diferença na atualização do Adamax é

$$\Delta x_{k,j} = -\frac{\alpha}{1 - \beta_1^k} \frac{\widehat{m}_{k,j}}{\overline{u_{k,j}}}, \quad j = 1, \dots, m. \quad (3.72)$$

### Nadam

O momento de Nesterov é uma tentativa de prever o gradiente no próximo passo da descida de gradiente. Nadam (*Nesterov-accelerated Adaptive Moment Estimation*) [26] incorpora essa ideia na técnica Adam, dando origem a diferença de parâmetros

$$\Delta x_{k,j} = -\frac{\alpha}{\sqrt{\varepsilon + \widehat{v}_{k,j}}} \left( \beta_1 \widehat{m}_{k,j} + \frac{(1 - \beta_1) g_{k,j}}{1 - \beta_1^k} \right), \quad j = 1, \dots, m, \quad (3.73)$$

onde os termos  $\widehat{m}_{k,j}$  e  $\widehat{v}_{k,j}$  têm o mesmo significado como no Adam na (eq. 3.70a) e (eq. 3.70b).

### Comparação dos Métodos

É importante lembrar que a função objetivo a ser minimizada pode ser extremamente não linear. A sua complexidade cresce em conformidade com a quantidade de parâmetros a serem otimizados. O exemplo mais apropriado é a função de perda em um problema de aprendizagem supervisionada. Se o modelo for uma rede neural artificial com muitas camadas e se a dimensão do vetor de entrada for muito alta, por exemplo, em uma aplicação de processamento de imagens, teremos uma quantidade muito grande de parâmetros. Como não linearidade é introduzida pelas funções de ativação na rede<sup>11</sup>, a função objetivo igualmente se torna não linear, e por causa da grande quantidade dos parâmetros muito complexa. Isso significa que todos as heurísticas aqui apresentadas nunca garantem um resultado ótimo, pois se baseiam principalmente no gradiente da função em um ponto do imenso espaço dos parâmetros. Isso também significa que não pode existir um método universal que mostra sempre os melhores resultados. Os experimentos feitos com esses métodos dependem dos dados usados, da escolha dos hiperparâmetros e da inicialização dos parâmetros [106]. Especialmente a escolha dos hiperparâmetros pode influenciar o resultado de um experimento, especialmente em caso de disponibilidade de poucos padrões para treinar o modelo. Analizaremos esta questão muito importante no contexto dos métodos de avaliação de desempenho.

<sup>11</sup>Estudaremos ainda as definições das funções de ativação no contexto das redes neurais.

## Software

Este livro serve para apresentar os fundamentos teóricos das heurísticas apresentadas nesta seção. Conhecendo as ideias atrás de uma implementação protege o usuário de simplesmente invocar uma função em um ambiente de programação e não saber o que essa função faz. Evite chegar a conclusões falsas do tipo “O Adam é melhor que o Nadam”. Esse perigo é real, pois existem plataformas de desenvolvimento que providenciam os métodos aqui apresentados através de uma interface simples. No momento da escrita deste texto a biblioteca de aprendizagem profunda Keras em linguagem Python disponibiliza todas as heurísticas nesta seção. Parcialmente a interface difere dos métodos originais, motivado por uma uniformização da interface. Um exemplo é a disponibilidade da taxa de diminuição (eq. 3.59) que não aparece, por exemplo, na definição teórica do Adagrad.

## 3.5 Perceptron

Após os conceitos aprendidos até agora, podemos definir um modelo prático, capaz de fazer uma classificação binária, com a fase de treinamento baseado na descida de gradiente. Frank Rosenblatt é um dos pioneiros da teoria das redes neurais artificiais. Apresentou as suas ideias sobre um *Perceptron* em uma série de publicações [90, 91, 92] nos anos 1950 e 1960. Nesses textos tentou elaborar um modelo bastante geral, composto de unidades relativamente simples de processamento de informação, as conexões entre essas unidades, e métodos de ajustar os parâmetros livres, ou seja, um algoritmo de treinamento. O modelo concreto apresentado aqui limita-se ao classificador binário, na terminologia de Rosenblatt chamado *Perceptron Elementar*  $\alpha$ . Mesmo assim, vale a pena estudar com um pouco mais de profundidade as suas propostas, pois muitas técnicas atualmente utilizadas no aprendizado de máquina são reconhecíveis nas ideias elaboradas no período inicial das redes neurais.

### 3.5.1 Conceitos Históricos do Perceptron

A teoria de Rosenblatt apresentada nas primeiras publicações tinha uma forte influência dos trabalhos até então sobre redes neurais artificiais. A analogia biológica foi uma forte motivação na definição dos modelos propostos<sup>12</sup>. Embora a sua teoria em grandes partes não seja diretamente transferível para uma arquitetura concreta, vale a pena apresentar as suas ideias, pois podemos reencontrar conceitos interessantes nos modelos de aprendizado de máquina contemporâneos.

O elemento central é o neurônio que recebe estímulos excitatórios e inibitórios através de sinapses. Se a soma desses impulsos antagonísticas ultrapassar um limiar, o neurônio dispara, passando essa atividade para elementos subsequentes. A figura 3.11 mostra o modelo original do Perceptron em uma das primeiras publicações de Rosenblatt [90]. A aplicação intencionada é um processamento ótico, por isso a origem de informação é chamada de retina. Ela representa a fonte de dados que podemos identificar em qualquer aplicação de aprendizado de máquina.

Conectados à retina são unidades sensoriais (“Unidades-S”) que respondem em forma binária (ativo ou inativo) na sua versão simples. Representam os sensores ligados à fonte de dados. Estes impulsos são propagados adiante para um conjunto de células associativas (“Unidades-A”) na área de projeção  $A_I$ . As conexões entre as unidades sensoriais e unidades associativas são muitos para

---

<sup>12</sup>Na minha opinião a discrepância entre os modelos apresentados neste texto e a realidade biológica é tão grande que as tentativas de emular a inteligência humana através de sistemas de computação não é viável. Nem um único neurônio biológico pode ser modelado na sua completa funcionalidade elétrica e química. A complexa interação dessas componentes biológicas que permite representar pensamentos e definir a inteligência humana não pode ser simplesmente transferida para um conjunto de hardware e software. Por causa disso, nenhuma tentativa é feita aqui para estabelecer uma conexão entre biologia e tecnologia. A teoria das redes neurais artificiais aqui usada não passa de algumas analogias conceptuais, por exemplo a existência de muitos elementos básicos, ligados em uma rede. O nosso escopo limita-se nos conceitos matematicamente bem descriptivos.

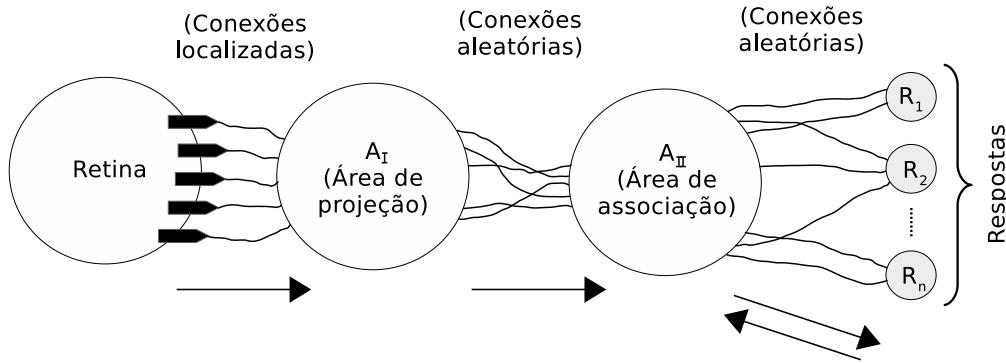


Figura 3.11: Modelo original do Perceptron de Rosenblatt de 1958 [90].

uma. Uma unidade associativa usa o conceito do neurônio com entradas excitatórias e inibitórias e um limiar. As ligações da retina para a área de projeção são determinísticas, ou seja, são definidas segundo algum critério. Dentro da área de projeção existe uma relação topológica entre as unidades. Eles formam grupos em torno de uma unidade central. Por exemplo, as conexões das unidades sensoriais da retina que são próximas levam para as unidades de projeção que são próximas também. Da área de projeção existem conexões para a área de associação  $A_{II}$ . Essas conexões são aleatórias. Finalmente, da área de associação, mais uma vez, aleatoriamente as conexões levam para as células de resposta (“Células-R”) que representam as saídas do Perceptron. No modelo simples existe somente uma única resposta. As setas na figura 3.11 mostram a direção do fluxo de informação. Da parte das respostas existe uma direção de processamento de informação oposta que representa o processo de ajuste dos parâmetros livres do modelo do Perceptron, ou seja, o mecanismo de aprendizado. Essa retroalimentação do resultado do processamento foi denominado “sistema de reforço” (*reinforcement system*). O que é muito interessante nesta proposta do Perceptron é que os conceitos apresentados fazem parte das arquiteturas atualmente utilizadas.

- Extração de características a partir de sinais físicos:** Considerando a transição da retina na figura 3.11 até a área de projeção, podemos identificar isso como um processo de extração de características a partir de uma fonte de dados inicial. Por exemplo, a retina poderia ser uma simples matriz de píxeis pretos e brancos. As conexões de dois desses píxeis para uma única unidade na área de projeção expressa um estímulo simultâneo. Se dois píxeis forem brancos, e o limiar da unidade na área de projeção for igual a um, a unidade dispara, pois um ou mais píxeis são brancos.
- Relação topológica entre unidades:** Na área de associação o modelo do Perceptron prevê uma relação de unidades do mesmo tipo, por exemplo, no exemplo anterior, se dois píxeis são próximos na retina, eles são conectados a unidades próximas na área de projeção. Este conceito podemos identificar no modelo do *Mapa Auto-organizável*, *Self-organizing Map*, *SOM* [65, 66], onde durante a fase de treinamento a vizinhança dos neurônio é relevante. O SOM será apresentado mais adiante.
- Conexões aleatórias:** Já tivemos contato com o potencial da aleatoriedade em aprendizado de máquina anteriormente. A Máquina Extrema de Aprendizado, ELM, na seção 4.3.1 na pág. 138 aumentou muito a sua capacidade de regressão ou classificação pela camada oculta que fez a extração aleatória de novas características. Também as ensembles (floresta) de Árvores de Decisão e Regressão, da seção 8.5 na pág. 252 trabalham com o conceito da aleatoriedade para formar um modelo com capacidade de generalização melhor. As conexões aleatórias entre as unidades são exatamente uma proposta para realizar uma extração não determinística de características. Podemos reconhecer em arquiteturas de redes neurais profundas esse conceito.

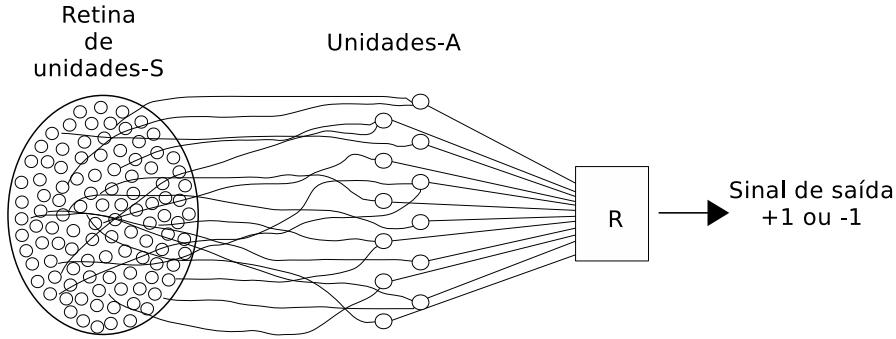


Figura 3.12: Modelo do Perceptron Simples de Rosenblatt de 1961 [92], com uma saída binária.

4. **Combinações lineares e viés em forma de limiar:** Na definição original do Perceptron não existe ainda o conceito do *peso*. Tem conexões excitatórias e inibitórias que são modelados por um valor binário, ou um, ou zero. Se a soma dessas entradas ultrapassar um limiar  $\theta$ , o neurônio dispara, e será por sua vez uma conexão excitatória para a unidade que recebe este valor. Vamos assumir que um neurônio tem  $m$  entradas  $x_j$ ,  $j = 1, \dots, m$ . Podemos modelar o fato que a  $j$ -ésima entrada é excitatória como um peso  $w_j = 1$  como multiplicador dessa entrada, e uma entrada inibitória como um peso  $w_j = 0$ . Assim o cálculo do neurônio é, considerando o modelo linear básico da (eq. 2.2) na pág. 32

$$d(\mathbf{x}; \mathbf{w}, b) = (1 + \text{sgn}(\mathbf{w} \cdot \mathbf{x} - b))/2, \quad (3.74)$$

o que resulta em um valor  $d \in \{0, 1\}$ . A única modificação é o sinal do viés  $b = -\theta$  e a mudança de saída de  $\{-1, 1\}$  para  $\{0, 1\}$ .

5. **Retropropagação do erro de uma função objetivo:** O Perceptron propõe um algoritmo iterativo para ajustar os seus parâmetros livres, chamado de *correção de erro*, (*error correction*). Pertence à categoria de aprendizagem supervisionada, pois após a apresentação de uma entrada com saída desejada, a saída do modelo é comparada com a desejada, dando origem a uma função de perda. Definiu-se como função de perda o *critério de Perceptron*.

### 3.5.2 Perceptron Elementar $\alpha$

Vamos estudar uma das propostas de Rosenblatt que pode ser usado como classificador linear binário, acompanhado por um algoritmo iterativo de aprendizagem, baseado no princípio da descida de gradiente. O *perceptron elementar  $\alpha$* , *elementary  $\alpha$ -perceptron* categorizado por Rosenblatt [92] tem somente um única unidade de resposta  $R$ , veja a figura 3.12. Ele foi definido como uma especialização do *Perceptron Simples*. Foi definida uma nomenclatura própria, porém podemos reconhecer os conceitos. Por exemplo, o conjunto de treinamento foi chamado de “mundo dos estímulos” (*stimulus world W*), com os estímulos sendo os nossos padrões, e um “reforço” (reinforcement) é o ajuste dos parâmetros livres do sistema. Os “coeficientes de generalização” (generalization coefficients) na matriz-G, *G-matrix* agem como uma espécie de pesos binários. A partir das unidades-A podemos reconhecer um classificador linear binário, intensamente estudado na seção 2.4.1 na pág. 66. A parte da retina até as unidades-A realiza a ideia da conexão aleatória que já conhecemos na Máquina Extrema de Aprendizado, ELM. A literatura mais recente que trata do Perceptron considera principalmente a parte a partir das unidades-A. Então, o cálculo realizado pelo *perceptron elementar  $\alpha$*  é exatamente a função discriminativa do classificador linear binário, da definição 2.4.4 na pág. 69, e mostrado na figura 2.2 na pág. 43, sendo a função de ativação  $z$  o sinal do *score*,

$$F(\mathbf{x}) = \text{sgn} [\mathbf{w} \cdot \mathbf{x} + b]. \quad (3.75)$$

O resultado representa a classe positiva ou negativa,  $F(\mathbf{x}) \in \{-1, 1\}$ .

### 3.5.3 Algoritmo de Aprendizagem do Perceptron Elementar $\alpha$

Resta estudar o algoritmo de aprendizagem do Perceptron elementar. Rosenblatt propôs um método capaz de aprender um classificador binário, caso exista uma separação perfeita dos padrões das duas classes. O chamado *critério de Perceptron* apresenta uma função de perda a ser minimizada durante a aprendizagem [9, 10]

$$E_P(\mathbf{w}, b) \stackrel{\text{def}}{=} - \sum_{i \in \mathcal{M}} (\mathbf{w} \cdot \mathbf{x}_i + b)y_i, \quad (3.76)$$

onde todos os índices  $i$  são de padrões que foram classificados incorretamente. Eles fazem parte do conjunto  $\mathcal{M}$ . Um Perceptron que separa perfeitamente as duas classes não tem padrões que fazem parte deste conjunto, ou seja,

$$\mathcal{M} = \emptyset.$$

Este critério é simplesmente a contagem dos padrões do conjunto de treinamento que foram classificados incorretamente, quer dizer, a cardinalidade  $|\mathcal{M}|$  do conjunto  $\mathcal{M}$ . Lembre da definição 2.4.1 na pág. 67 e definição 2.4.4 na pág. 69 que um padrão  $\mathbf{x}_i \in \mathcal{C}_+$  fica do lado positivo do hiperplano, então

$$\hat{y}_i = \text{sgn}[\mathbf{w} \cdot \mathbf{x}_i + b] = +1.$$

Então se o hiperplano definido pelos parâmetros  $\mathbf{w}$  e  $b$  classifica um padrão da classe positiva com  $y_i = 1$  incorretamente temos

$$\hat{y}_i = -1,$$

ou seja

$$\hat{y}_i y_i = \text{sgn}[\mathbf{w} \cdot \mathbf{x}_i + b] y_i = (-1) \cdot 1 = -1.$$

Em analogia, um padrão da classe negativa com  $y_i = -1$ , classificado erradamente implica

$$\hat{y}_i y_i = \text{sgn}[\mathbf{w} \cdot \mathbf{x}_i + b] y_i = 1 \cdot (-1) = -1.$$

Então a soma negativa da (eq. 3.76) é simplesmente um contador desses padrões que estão do lado errado do hiperplano.

Consequentemente, a aprendizagem do hiperplano é a minimização desta função de perda. Temos novamente a opção de aprendizagem em lote, mini-lote ou estocástica, veja a definição 3.2.2 na pág. 88. Vamos primeiro estudar o caso estocástico, ou seja, apresentar um único padrão ao modelo e ajustar os parâmetros. Obviamente somente temos que modificar os pesos e o bias, se houver um erro, ou seja  $\hat{y}_i y_i = -1$ . O critério de Perceptron da (eq. 3.76) a ser minimizado é derivável, o que permite a aplicação do princípio básico da descida de gradiente, definido na (eq. 3.10) na pág. 87

$$\begin{bmatrix} \mathbf{w}_{k+1} \\ b_{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{w}_k \\ b_k \end{bmatrix} - \alpha \nabla E_P(\mathbf{w}_k, b_k)^\top \quad (3.77)$$

$$= \begin{bmatrix} \mathbf{w}_k \\ b_k \end{bmatrix} - \alpha \begin{bmatrix} \partial[(\mathbf{w}_k \cdot \mathbf{x}_i + b)y_i] / \partial \mathbf{w}_k \\ \partial[(\mathbf{w}_k \cdot \mathbf{x}_i + b)y_i] / \partial b \end{bmatrix}, \quad (3.78)$$

ou seja,

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha \mathbf{x}_i y_i \quad (3.79)$$

$$b_{k+1} = b_k - \alpha 1 \cdot y_i. \quad (3.80)$$

Observe, que essa regra de atualização somente entra em ação em caso de classificação errada de uma padrão de treinamento.

Podemos alternativamente formular uma regra mais elegante para treinar o Perceptron. Considere as três “tentativas” de definir uma função de perda no exemplo mais básico da regressão linear, na seção 2.1.3 na pág. 33. Vamos generalizar para o classificador linear binário, onde os padrões  $\mathbf{x}$  e pesos  $\mathbf{w}$  têm mais que uma componente. Na teoria apresentada na seção 2.1.3, tinha somente uma componente  $x$  e  $w$ , mas as regras também são válidas para vetores de características  $\mathbf{x} \in \mathbb{R}^m, m > 1$ . Se considerarmos o viés como peso  $w_0$ , de fato já trabalhamos com mais que uma componente. Consideremos o caso da aprendizagem estocástica, ou seja, a função de perda *individual*

$$L_i(\mathbf{w}, b)$$

de cada padrão  $\mathbf{x}_i$  tem que ser minimizada. A primeira tentativa foi a simples diferença entre resultado esperado e calculado veja a (eq. 2.24) na pág. 38,

$$L_i(\mathbf{w}, b) = y_i - \hat{y}_i,$$

com o seu gradiente

$$\nabla L_i = - \begin{bmatrix} \mathbf{x}_i \\ 1 \end{bmatrix}^\top$$

Se aplicarmos este gradiente na regra básica da descida de gradiente (eq. 3.10) na pág. 87 no caso estocástico,

$$\begin{bmatrix} \mathbf{w}_{k+1} \\ b_{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{w}_k \\ b_k \end{bmatrix} - \alpha \nabla L_i(\mathbf{w}_k, b_k)^\top, \quad (3.81)$$

teríamos

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \alpha \mathbf{x}_i$$

$$b_{k+1} = b_k + \alpha 1.$$

Esta regra é completamente inapropriada, pois o viés  $b$  seria incrementado em cada iteração, independente da classe do padrão  $\mathbf{x}$ , e consequentemente  $b$  ia crescer infinitamente, deslocando a fronteira entre as classes positiva e negativa cada vez mais longe.

Na segunda tentativa, a função de perda foi o módulo da diferença entre resultado esperado e calculado, veja a (eq. 2.27) na pág. 39,

$$L_i(\mathbf{w}, b) = |y_i - \hat{y}_i|.$$

Na (eq. 2.35) na pág. 41 fomos capazes de unificar o gradiente desta função de perda como

$$\nabla L_i(\mathbf{w}, b) = -\text{sgn}(y_i - \hat{y}_i) \begin{bmatrix} \mathbf{x}_i \\ 1 \end{bmatrix}^\top. \quad (3.82)$$

Assim somos capazes de formular uma regra de aprendizagem do Perceptron que coincide perfeitamente com a regra na (eq. 3.80), porém para todos os padrões, não somente para os que foram classificados incorretamente. Temos

$$\begin{aligned} \begin{bmatrix} \mathbf{w}_{k+1} \\ b_{k+1} \end{bmatrix} &= \begin{bmatrix} \mathbf{w}_k \\ b_k \end{bmatrix} - \alpha \nabla L_i(\mathbf{w}_k, b_k)^\top \\ &= \begin{bmatrix} \mathbf{w}_k \\ b_k \end{bmatrix} + \alpha \operatorname{sgn}(y_i - \hat{y}_i) \begin{bmatrix} \mathbf{x}_i \\ 1 \end{bmatrix}. \end{aligned} \quad (3.83)$$

Os padrões de treinamento são apresentados ao Perceptron em ordem aleatória. Após o cálculo de saída de cada padrão, os pesos e o viés são atualizados, veja o algoritmo 5 na pág. 88 e algoritmo 6. Observe, se não houver erro em um problema de classificação binária, a diferença  $y_i - \hat{y}_i$  será zero, pois se  $y_i = \hat{y}_i$  temos ou  $1 - 1 = 0$ , ou  $-1 - (-1) = 0$ . Assim somente erros de classificação são considerados. A direção certa de descida da função é automaticamente definida pelo sinal da diferença, ou temos

$$\operatorname{sgn}(y_i - \hat{y}_i) = \operatorname{sgn}(1 - (-1)) = \operatorname{sgn}(2) = +1,$$

ou temos

$$\operatorname{sgn}(y_i - \hat{y}_i) = \operatorname{sgn}(-1 - 1) = \operatorname{sgn}(-2) = -1.$$

Ainda podemos fazer uma observação interessante, se simplificamos a taxa de aprendizagem para  $\alpha = 1$ . Neste caso, em um passo de ajuste de pesos, ou somamos o padrão diretamente ao peso

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{x}_i,$$

ou o subtraímos

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \mathbf{x}_i,$$

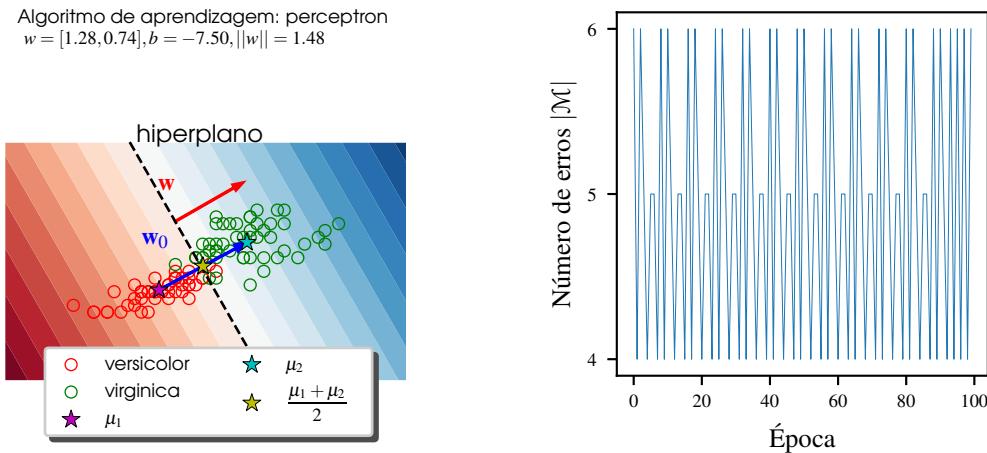
dependendo da classe esperada  $y_i = +1$ , ou  $y_i = -1$ , respectivamente. Isso mostra que no fundo, pesos e padrões têm a mesma natureza, são do mesmo espaço vetorial. Podemos concluir que a regra de atualização estocástica dos pesos e do viés definida na (eq. 3.83) é uma alternativa, contornando completamente a formulação do critério de Perceptron (eq. 3.76) necessitando exclusivamente a regra básica da descida de gradiente.

Obviamente, existe uma versão do algoritmo em lote. Neste caso, não é necessário embaralhar a ordem de apresentação dos padrões de treinamento. Todos os  $n$  padrões são apresentados, a perda individual é acumulada, e depois os pesos e o viés são atualizados. A regra de atualização que estima o valor esperado da perda pela média aritmética das perdas individuais é

$$\begin{aligned} \begin{bmatrix} \mathbf{w}_{k+1} \\ b_{k+1} \end{bmatrix} &= \begin{bmatrix} \mathbf{w}_k \\ b_k \end{bmatrix} - \alpha \frac{1}{n} \sum_{i=1}^n \nabla L_i(\mathbf{w}_k, b_k)^\top \\ &= \begin{bmatrix} \mathbf{w}_k \\ b_k \end{bmatrix} + \frac{\alpha}{n} \sum_{i=1}^n \operatorname{sgn}(y_i - \hat{y}_i) \begin{bmatrix} \mathbf{x}_i \\ 1 \end{bmatrix}. \end{aligned} \quad (3.84)$$

Uma boa inicialização dos pesos  $\mathbf{w}_0$  e do viés  $b_0$  para a iteração  $k = 0$  inicial, em caso de duas classes é escolher como normal do hiperplano a diferença entre as duas médias da classe positiva

$$\boldsymbol{\mu}_+ = \frac{1}{n_+} \sum_{i_+} \mathbf{x}_{i_+},$$



(a) Hiperplano aprendido pelo algoritmo de Perceptron, aprendizagem em lote.

(b) Evolução da quantidade de erros em dependência das épocas, aprendizagem em lote.

Figura 3.13: Algoritmo de Perceptron em lote, 100 iterações, taxa de aprendizagem  $\alpha = 0.1$ , inicialização aleatória dos pesos  $w$  e viés  $b$ .

e negativa

$$\boldsymbol{\mu}_- = \frac{1}{n_-} \sum_{i_-} \mathbf{x}_{i_-},$$

ou seja

$$\mathbf{w}_0 = \boldsymbol{\mu}_+ - \boldsymbol{\mu}_-, \quad (3.85)$$

onde os  $n_+$  padrões  $\mathbf{x}_{i_+}$  constituem a classe positiva, e os  $n_-$  padrões  $\mathbf{x}_{i_-}$  constituem a classe negativa. Isso faz sentido, se imaginarmos que temos somente essa duas médias das duas classes como os únicos dois padrões. A inicialização não funcionará bem, se as duas classes tiverem muitos padrões espalhados no espaço de características e ainda por cima estão bastante sobrepostas. Na figura 3.13 o vetor de peso inicial é mostrado. Para a inicialização do viés, o seguinte valor faz sentido

$$b_0 = -\mathbf{w}_0 \cdot \frac{\boldsymbol{\mu}_+ + \boldsymbol{\mu}_-}{2}, \quad (3.86)$$

pois o ponto intermediário

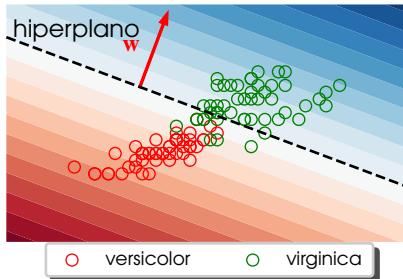
$$\bar{\boldsymbol{\mu}} \stackrel{\text{def}}{=} \frac{1}{2}(\boldsymbol{\mu}_+ + \boldsymbol{\mu}_-),$$

supostamente faz parte do hiperplano, então o seu score é zero

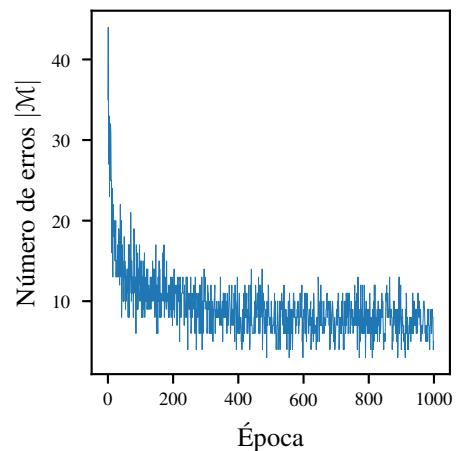
$$\mathbf{w}_0 \cdot \bar{\boldsymbol{\mu}} + b_0 = 0.$$

**Exemplo 3.14** Na figura 3.13 mostra-se o resultado do algoritmo de Perceptron na modalidade de treinamento em lote, e com a inicialização de  $w$  e  $b$  proposta na (eq. 3.85) e (eq. 3.86). A taxa de aprendizagem foi fixada em  $\alpha = 0.1$  e o número de iterações foi  $k = 100$ . Observe que logo no início a quantidade de erros  $|\mathcal{M}|$  oscila em torno de cinco. Isto sugere que a inicialização foi muito boa e o algoritmo tem pouco trabalho para convergir para uma aproximação boa do

Algoritmo de aprendizagem: perceptron  
 $w = [26.23, 70.30], b = -264.00, \|w\| = 75.03$



(a) Hiperplano aprendido pelo algoritmo de Perceptron, aprendizagem estocástica.



(b) Evolução da quantidade de erros em dependência das épocas, aprendizagem estocástica.

Figura 3.14: Algoritmo de Perceptron estocástico, 1000 iterações, taxa de aprendizagem  $\alpha = 1.0$ , inicialização não aleatória dos pesos  $w$  e viés  $b$ .

hiperplano. Na figura 3.14, o modelo do Perceptron é treinado estocasticamente, apresentando um único padrão e atualizar os parâmetros imediatamente. Os hiperparâmetros foram  $\alpha = 1.0$  e  $k = 1000$ . A inicialização de  $w$  e  $b$  foi feita aleatoriamente. O efeito é que a quantidade de erros inicialmente é grande e após as 1000 iterações converge para valores parecidos com os do algoritmo na versão treinamento em lote. ■

### 3.5.4 Convergência do Hiperplano

Se existir uma separação linear entre as amostras de duas classes, o algoritmo de Perceptron converge em um número finito de passos para um vetor de pesos  $w$  e viés  $b$  que definem um hiperplano  $w \cdot x + b = 0$  que divide todas as amostras de tal maneira que todas as amostras da classe positiva ficam do lado positivo do hiperplano e todas as amostras da classe negativa ficam do lado negativo do hiperplano. A prova é omitida aqui por razões de espaço, o leitor interessado pode consultar, por exemplo [9, 10, 48, 79, 86]. Nas últimas três dessas publicações encontram-se também gráficos didáticos que mostram a evolução do vetor de pesos (em forma aumentada) para a aprendizagem estocástica.



## 4. Extração de Características

### 4.1 Introdução

A tarefa é a seguinte. Dado a altura e o peso de uma pessoa, categoriza esse pessoa em termos de aparência corporal. Existem quatro categorias, ou equivalentemente classes, “abaixo do peso”, “normal”, “acima do peso” e “obeso”. Cada classe tem duas ou mais subclasses, não considerados aqui. Uma única variável que foi definida no século XIX por Quetelet [59] tenta definir as categorias como intervalos bem definidos.

**Definição 4.1.1 — Índice de Massa Corporal, IMC.** Dada a massa  $m$  e a altura  $h$  de uma pessoa, o IMC (*Body Mass Index*, BMI) é

$$\text{IMC} \stackrel{\text{def}}{=} \frac{m}{h^2} \quad (4.1)$$

A fronteira entre “abaixo do peso” e “normal” é  $18.5\text{kg}/\text{m}^2$ , entre “normal” e “acima do peso” é  $25.0\text{kg}/\text{m}^2$ , e entre “acima do peso” e “obeso” é  $30.0\text{kg}/\text{m}^2$ .

A figura 4.1a mostra o espaço das duas características envolvidas, o peso no eixo  $x$ , a altura no eixo  $y$ . Uma quantidade de  $n = 500$  pessoas foram simuladas, o peso de uma distribuição uniforme do intervalo  $[40\text{kg}, 140\text{kg}]$  e a altura de uma distribuição uniforme do intervalo  $[140\text{cm}, 200\text{cm}]$ , veja a definição 1.2.1 na pág. 17. Após a amostragem uniforme, o IMC foi calculado conforme a (eq. 4.1). Dados os valores  $\text{IMC}_1 = 18.5$ ,  $\text{IMC}_2 = 25.0$ ,  $\text{IMC}_3 = 30.0$  do IMC que definem a fronteira entre as categorias, as três fronteiras  $h(m, \text{IMC}_1)$ ,  $h(m, \text{IMC}_2)$ ,  $h(m, \text{IMC}_3)$ , foram desenhadas, resolvendo a (eq. 4.1) segundo a altura, dados o peso e o valor limite do IMC como,

$$h = \sqrt{\frac{m}{\text{IMC}}}.$$

Esse dependência funcional entre peso e altura é obviamente não linear. Isto implica que um classificador como a Máquina Linear cometeria erros de classificação que têm a ver com a natureza do problema.

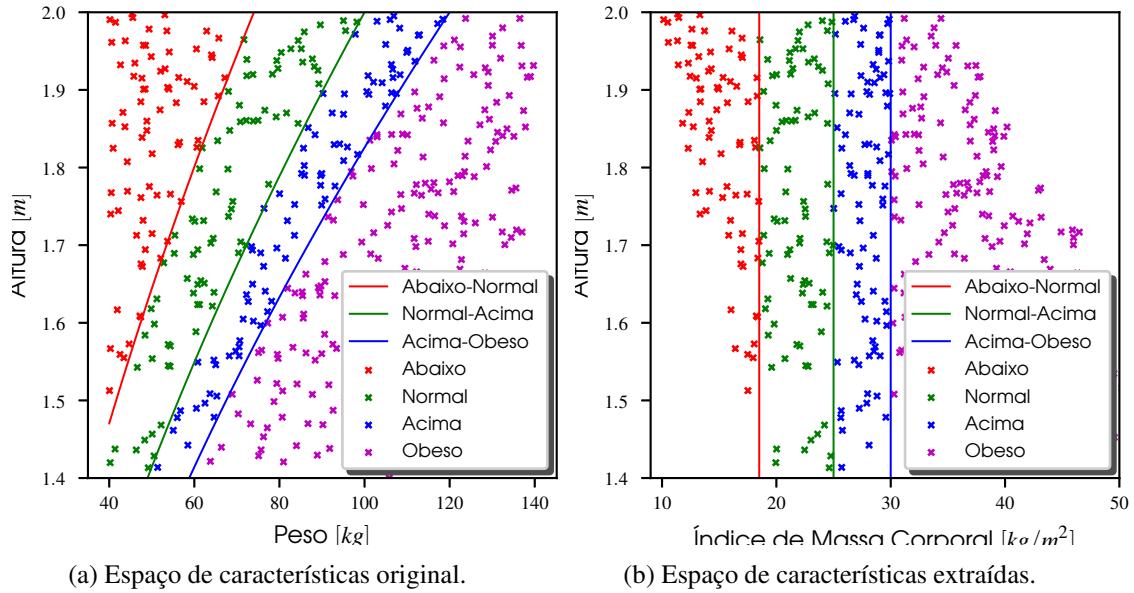


Figura 4.1: Regiões de Decisão do Índice de Massa Corporal.

A figura 4.1b justapõe o novo espaço de características criado pelo uso do IMC. O eixo  $y$  mostra ainda a altura, mas é redundante. O eixo  $y$  serve somente para esticar a ocorrência das amostras para não sobrecarregar o gráfico. O IMC no eixo  $x$  em princípio é o novo espaço de características, unidimensional. Como as fronteiras entre as categorias foram justamente definidos dentro do domínio unidimensional do IMC, elas são linhas verticais que separam sem erro as amostras. A mensagem deste exemplo em relação às características disponíveis inicialmente para classificar é

1. No espaço original  $\mathbb{R}^m$  das características  $x_1, \dots, x_m$  as regiões de decisão podem ser sofisticadas;
2. O cálculo de novas características pode criar um novo espaço de características, em que a separação de classes pode ser mais fácil;
3. A dimensão do novo espaço de características em geral tem uma dimensão diferente, comparado com a dimensão original  $m$ , podendo ser menor, igual, ou maior.

O mapeamento funcional do espaço de características para um novo espaço de características é a *extração de características*. A extração pode ser explícita, como no exemplo do IMC pela *BMI*, ou pode ser implícita. Considerando uma rede neural com várias camadas, cada uma no fundo calcula um novo conjunto de características. Vamos formalizar a extração.

**Definição 4.1.2 — Extração de Características.** Seja dado um vetor  $\mathbf{x} \in \mathbb{R}^p$  de características em um espaço vetorial de dimensão  $p$ , o mapeamento funcional  $\phi$  (“phi”)

$$\begin{aligned} \phi : \mathbb{R}^p &\rightarrow \mathbb{R}^q \\ \mathbf{x} &\mapsto \phi(\mathbf{x}) \\ \begin{bmatrix} x_1 \\ \vdots \\ x_p \end{bmatrix} &\mapsto \begin{bmatrix} \phi_1 \left( \begin{bmatrix} x_1 & \cdots & x_p \end{bmatrix}^\top \right) \\ \vdots \\ \phi_q \left( \begin{bmatrix} x_1 & \cdots & x_p \end{bmatrix}^\top \right) \end{bmatrix} \end{aligned} \tag{4.2}$$

que projeta o vetor  $\mathbf{x}$  para um novo vetor de características  $\phi(\mathbf{x})$  de dimensão  $q$  é um *extrator de características*.

■ **Exemplo 4.1** No exemplo do IMC temos um mapeamento bidimensional para unidimensional, ou seja,  $\phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \phi_1\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \phi_1\left(\begin{bmatrix} m \\ h \end{bmatrix}\right) = m/h^2$ . ■

■ **Exemplo 4.2** A Máquina Linear da definição 2.2.1 na pág. 53 é um extrator de características. Na figura 2.4 na pág. 55 uma flor é mapeada para os *scores* das três classes. A flor é representada por um vetor aumentado  $\mathbf{x}$  de dimensão  $(m+1) = p = 5$ , e mapeado para os scores  $\mathbf{y}(\mathbf{x}) = \phi(\mathbf{x})$  de dimensão  $c = q = 3$ . Esses *scores* podem ser consideradas como características extraídas, utilizáveis imediatamente como grau de pertinência a uma certa classe. Uma propriedade importante deste extrator é a sua linearidade, ou seja, pode ser realizado pela multiplicação do vetor a ser mapeado por uma matriz, como  $\mathbf{y}(\mathbf{x}) = \phi(\mathbf{x}) = W\mathbf{x}$ . ■

## 4.2 Extração Linear de Características

Uma distinção principal das técnicas de extração de características é se eles são lineares ou não. Técnicas lineares são realizáveis simplesmente pela multiplicação de uma matriz  $M$  que representa o extrator. O novo vetor de características  $\mathbf{y}$ , dado a informação original  $\mathbf{x}$  é

$$\mathbf{y} = M\mathbf{x}. \quad (4.3)$$

### 4.2.1 Análise de Componentes Principais

Uma técnica de extrema importância em diversas áreas de ciência e engenharia a Análise de Componentes Principais, (*Principal Component Analysis*, PCA). São dois estágios, primeiro a *descorrelação linear*, segundo a eliminação de componentes do vetor extraído. A PCA pertence ao grupo de métodos de aprendizagem não supervisionada. Não requer informação de classes. As amostras são unicamente consideradas em relação a sua distribuição no espaço vetorial. Neste ponto, precisamos refrescar as definições das variáveis aleatórias multidimensionais da seção 1.4 na pág. 23, matriz de dados  $X$  (eq. 1.35) na pág. 24, centrada  $X_{\text{centrada}}$  (definição 1.4.3 na pág. 24), matriz de dados estandardizada  $X_{\text{estandardizada}}$  (definição 1.4.4 na pág. 25), valor esperado  $\mathbb{E}[\mathbf{x}] = \boldsymbol{\mu}$  de uma variável aleatória (definição 1.4.1 na pág. 23), como estimar o mesmo  $\hat{\boldsymbol{\mu}}$  (eq. 1.33), a matriz de covariância  $\Sigma$  (eq. 1.42) na pág. 26, e como estimar a mesma  $\hat{\Sigma}$  (eq. 1.45).

#### Descorrelação Linear, Decomposição de Autovetores e Autovalores

O objetivo da descorrelação é achar uma transformação linear que transforma a variável multidimensional original  $\mathbf{x} \in \mathbb{R}^m$  para uma nova variável multidimensional  $\mathbf{y} \in \mathbb{R}^m$  da mesma dimensão, porém sem covariância entre as suas componentes  $y_i$  e  $y_j$ , ou seja,  $\sigma_{i,j} = 0$ ,  $i, j = 1, \dots, m, i \neq j$ . Como a transformação é linear, ela é realizável por uma pré-multiplicação de uma matriz  $A$  de dimensão  $m \times m$  como

$$\mathbf{y} = A\mathbf{x}. \quad (4.4)$$

Temos as seguintes propriedades

$$\boldsymbol{\mu}_y = A\boldsymbol{\mu}_x \quad (4.5a)$$

$$\Sigma_y = A\Sigma_x A^T, \quad (4.5b)$$

onde  $\boldsymbol{\mu}_x$  é o valor esperado da variável aleatória  $\mathbf{x}$ ,  $\boldsymbol{\mu}_y$  é o valor esperado da variável aleatória mapeada  $\mathbf{y}$ ,  $\Sigma_x$  é a matriz de covariância de  $\mathbf{x}$  e  $\Sigma_y$  é a matriz de covariância de  $\mathbf{y}$ .

*Demonstração.*

$$\boldsymbol{\mu}_y = \mathbb{E}[\mathbf{y}] = \mathbb{E}[A\mathbf{x}] = \mathbb{E}[A]\mathbb{E}[\mathbf{x}] = A\mathbb{E}[\mathbf{x}] = A\boldsymbol{\mu}_x$$

O valor esperado do produto de duas variáveis aleatórias independentes é o produto dos valores esperados das mesmas. A troca de operadores do valor esperado e da transformação linear  $\mathbb{E}[A\mathbf{x}] = A\mathbb{E}[\mathbf{x}]$  é possível, porque é transformação linear é determinística, ou seja  $\mathbb{E}[A] = A$ .

$$\begin{aligned}\Sigma_y &\stackrel{1.43}{=} \mathbb{E} \left[ (\mathbf{y} - \boldsymbol{\mu}_y)(\mathbf{y} - \boldsymbol{\mu}_y)^\top \right] \stackrel{4.50}{=} \mathbb{E} \left[ (A\mathbf{x} - A\boldsymbol{\mu}_x)(A\mathbf{x} - A\boldsymbol{\mu}_x)^\top \right] \\ &= \mathbb{E} \left[ A(\mathbf{x} - \boldsymbol{\mu}_x)(A(\mathbf{x} - \boldsymbol{\mu}_x))^\top \right] \stackrel{2.59}{=} \mathbb{E} \left[ A(\mathbf{x} - \boldsymbol{\mu}_x)(\mathbf{x} - \boldsymbol{\mu}_x)^\top A^\top \right] \\ &= \mathbb{E} \left[ A \left\{ (\mathbf{x} - \boldsymbol{\mu}_x)(\mathbf{x} - \boldsymbol{\mu}_x)^\top \right\} A^\top \right] \stackrel{1.43}{=} \mathbb{E} \left[ A\Sigma_x A^\top \right] = \mathbb{E}[A]\mathbb{E}[\Sigma_x]\mathbb{E}[A^\top] \\ &= A\Sigma_x A^\top.\end{aligned}$$

Novamente, a última igualdade justifica-se pela independência da transformação linear e da variável aleatória, pela natureza determinística da transformação linear, e que o valor esperado do valor esperado de uma variável é o valor esperado da variável. ■

A matriz  $\Phi$  de dimensão  $m \times m$  que realizará a transformação linear é composta por  $m$  vetores ortonormais como linhas ou colunas  $\boldsymbol{\phi}_i = [\phi_{i,1} \ \cdots \ \phi_{i,j} \ \cdots \ \phi_{i,m}]^\top$ ,  $i = 1, \dots, m$ .

**Definição 4.2.1 — Base Ortonormal.** Uma base ortonormal em um espaço vetorial  $\mathbb{R}^m$  é composta por  $m$  vetores  $\boldsymbol{\phi}_i \in \mathbb{R}^m$  unitários e ortogonais um ao outro, ou seja

$$\boldsymbol{\phi}_i \cdot \boldsymbol{\phi}_j = 0, \quad i, j = 1, \dots, m, i \neq j \tag{4.6a}$$

$$\|\boldsymbol{\phi}_j\| = 1 = \boldsymbol{\phi}_j \cdot \boldsymbol{\phi}_j = \boldsymbol{\phi}_j^\top \boldsymbol{\phi}_j, \quad j = 1, \dots, m \tag{4.6b}$$

Um formalismo mais compacto é

$$\boldsymbol{\phi}_i \cdot \boldsymbol{\phi}_j = \delta_{ij} = \begin{cases} 1, & \text{se } i = j \\ 0, & \text{se } i \neq j \end{cases}, \tag{4.7}$$

onde  $\delta_{ij}$  é o *Delta de Kronecker*.

Uma consequência direta é que uma matriz composta por vetores que formam uma base ortonormal

$$\Phi \stackrel{\text{def}}{=} [\boldsymbol{\phi}_1 \ \cdots \ \boldsymbol{\phi}_j \ \cdots \ \boldsymbol{\phi}_m] = \begin{bmatrix} \phi_{1,1} & \cdots & \phi_{j,1} & \cdots & \phi_{m,1} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \phi_{1,i} & \cdots & \phi_{j,i} & \cdots & \phi_{m,i} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \phi_{1,m} & \cdots & \phi_{j,m} & \cdots & \phi_{m,m} \end{bmatrix}. \tag{4.8}$$

se multiplicada pela sua transposta, produz a matriz de identidade, ou seja

$$\Phi^\top \Phi = I, \tag{4.9}$$

pois na posição  $(i, j)$  temos o escalar  $\boldsymbol{\phi}_i^\top \boldsymbol{\phi}_j = \delta_{ij}$ . Automaticamente podemos concluir que a transposta de uma matriz ortonormal simultaneamente é a sua inversa, ou seja,

$$\Phi^{-1} = \Phi^\top. \tag{4.10}$$

■ **Exemplo 4.3** O sistema de coordenadas cartesiano de duas dimensões é definido pelos dois vetores de base bidimensionais  $\mathbf{e}_1 = [1 \ 0]^\top$  e  $\mathbf{e}_2 = [0 \ 1]^\top$ . Temos  $\mathbf{e}_1 \cdot \mathbf{e}_2 = 1 \cdot 0 + 0 \cdot 1 = 0$  e  $\|\mathbf{e}_1\| = \sqrt{1^2 + 0^2} = 1$  e  $\|\mathbf{e}_2\| = \sqrt{0^2 + 1^2} = 1$ . Também os dois vetores  $\boldsymbol{\phi}_1 = [3/5 \ 4/5]^\top$  e  $\boldsymbol{\phi}_2 = [-4/5 \ 3/5]^\top$  formam uma base ortonormal. Verifique! ■

Para as considerações a seguir, assumimos que a matriz de dados  $X$  seja centrada, ou seja cada uma das  $n$  amostras  $\mathbf{x}_i$  é substituída, subtraindo a média  $\mathbf{x}_i^{\text{novo}} \leftarrow \mathbf{x}_i - \boldsymbol{\mu}_x$ ,  $i = 1, \dots, n$ . Então temos

$$\begin{aligned}\mathbb{E}[\mathbf{x}^{\text{novo}}] &= \mathbb{E}[\mathbf{x} - \boldsymbol{\mu}_x] = \mathbb{E}[\mathbf{x}] - \mathbb{E}[\boldsymbol{\mu}_x] = \boldsymbol{\mu}_x - \boldsymbol{\mu}_x \\ &= \mathbf{0}\end{aligned}\quad (4.11a)$$

$$\begin{aligned}\Sigma_x^{\text{novo}} &= \mathbb{E}[(\mathbf{x}^{\text{novo}} - \boldsymbol{\mu}_x^{\text{novo}})(\mathbf{x}^{\text{novo}} - \boldsymbol{\mu}_x^{\text{novo}})^\top] \\ &= \mathbb{E}[(\mathbf{x} - \boldsymbol{\mu}_x - \mathbf{0})(\mathbf{x} - \boldsymbol{\mu}_x - \mathbf{0})^\top] \\ &= \mathbb{E}[(\mathbf{x} - \boldsymbol{\mu}_x)(\mathbf{x} - \boldsymbol{\mu}_x^\top)] \\ &= \Sigma_x.\end{aligned}\quad (4.11b)$$

Em outras palavras, a nova média dos dados centrados é o vetor nulo, e a matriz de covariância é invariante em relação à translação dos dados, somando um vetor a cada padrão  $\mathbf{x}$ .

A descorrelação da variável aleatória  $\mathbf{x}$  é realizada pela transformação linear

$$\mathbf{y} = \Phi^\top \mathbf{x} = \begin{bmatrix} \boldsymbol{\phi}_1^\top \\ \vdots \\ \boldsymbol{\phi}_j^\top \\ \vdots \\ \boldsymbol{\phi}_m^\top \end{bmatrix} \mathbf{x} = \begin{bmatrix} \boldsymbol{\phi}_1^\top \mathbf{x} \\ \vdots \\ \boldsymbol{\phi}_j^\top \mathbf{x} \\ \vdots \\ \boldsymbol{\phi}_m^\top \mathbf{x} \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_j \\ \vdots \\ y_m \end{bmatrix}, \quad (4.12)$$

onde os vetores de dimensão  $m$ ,  $\boldsymbol{\phi}_j$ ,  $j = 1, \dots, m$  são as colunas da matriz  $\Phi$  que realiza a descorrelação, ou seja,  $\Phi = [\boldsymbol{\phi}_1 \ \dots \ \boldsymbol{\phi}_j \ \dots \ \boldsymbol{\phi}_m]$ . Os vetores  $\boldsymbol{\phi}_j$  que compõem a matriz formam uma base ortonormal. A  $j$ -ésima componente  $y_j$  do novo vetor  $\mathbf{y}$  então é o produto escalar do  $j$ -ésimo vetor da matriz ortonormal com o vetor original  $\mathbf{x}$  de características

$$y_j = \boldsymbol{\phi}_j \cdot \mathbf{x} = \boldsymbol{\phi}_j^\top \mathbf{x} \stackrel{2.59a}{=} \mathbf{x}^\top \boldsymbol{\phi}_j. \quad (4.13)$$

Enquanto, a transformação linear incorporada em  $\Phi$  é determinística, o novo vetor  $\mathbf{y}$  e cada uma das suas componentes  $y_j$  são variáveis aleatórias, dado que  $\mathbf{x}$  é uma variável aleatória. Sendo  $y_j$  uma variável aleatória unidimensional, ela tem um valor esperado  $\mu_j = \mathbb{E}[y_j]$ , (eq. 1.22) e uma variância  $\sigma_j^2 = \mathbb{E}[(y_j - \mu_j)^2]$ . Podemos calcular esses valores. Omitindo o índice  $j$ , com  $\mathbf{x}$  centrado  $\boldsymbol{\mu}_x = \mathbf{0}$ , temos

$$\mu_y = \mathbb{E}[y] = \mathbb{E}[\mathbf{x}^\top \boldsymbol{\phi}] = \mathbb{E}[\mathbf{x}]^\top \mathbb{E}[\boldsymbol{\phi}] = \mathbb{E}[\mathbf{x}]^\top \boldsymbol{\phi} = \boldsymbol{\mu}_x^\top \boldsymbol{\phi} = \mathbf{0}^\top \boldsymbol{\phi} = 0 \quad (4.14a)$$

$$\begin{aligned}\sigma_y^2 &\stackrel{1.26}{=} \mathbb{E}[(\mathbf{x}^\top \boldsymbol{\phi} - \mu_y)^2] = \mathbb{E}[(\mathbf{x}^\top \boldsymbol{\phi})^2] \stackrel{a}{=} \mathbb{E}[(\mathbf{x}^\top \boldsymbol{\phi})^\top (\mathbf{x}^\top \boldsymbol{\phi})] \\ &\stackrel{2.59c}{=} \mathbb{E}[\boldsymbol{\phi}^\top \mathbf{x} \mathbf{x}^\top \boldsymbol{\phi}] = \mathbb{E}[\boldsymbol{\phi}^\top] \mathbb{E}[\mathbf{x} \mathbf{x}^\top] \mathbb{E}[\boldsymbol{\phi}] \\ &\stackrel{1.43}{=} \boldsymbol{\phi}^\top \Sigma_x \boldsymbol{\phi}.\end{aligned}\quad (4.14b)$$

Podemos observar então que a variância da componente  $y$  da nova variável  $\mathbf{y}$  depende do vetor correspondente  $\boldsymbol{\phi}$  da transformação linear. Este vetor é o grau de liberdade para aumentar ou diminuir a variância de  $y$ . Temos que definir um critério de qualidade que guia a escolha do vetor  $\boldsymbol{\phi}$ . Vamos escolher o vetor que *maximiza* a variância da (eq. 4.14b), ou seja

$$\boldsymbol{\phi}^* = \arg \max_{\boldsymbol{\phi}} (\boldsymbol{\phi}^\top \Sigma_x \boldsymbol{\phi}). \quad (4.15)$$

Se  $\boldsymbol{\phi}$  fosse um vetor qualquer, uma solução trivial seria um vetor cujo módulo seria infinito, isto é,  $\|\boldsymbol{\phi}\| = \infty$ , dado o fato que a matriz de covariância  $\Sigma_x$  é definida positiva, ou seja  $\boldsymbol{\phi}^\top \Sigma_x \boldsymbol{\phi} > 0$ . Como  $\boldsymbol{\phi}$  é um vetor da matriz  $\Phi$ , ele é um vetor unitário, ou seja,  $\|\boldsymbol{\phi}\| = 1 = \boldsymbol{\phi}^\top \boldsymbol{\phi}$ .

Neste ponto da teoria, encontramos uma nova categoria de um problema de otimização, um *problema de otimização com restrição*. A forma genérica é

$$\mathbf{x}^* = \arg \min_{\mathbf{x}} f(\mathbf{x}) \quad \text{sujeito a} \quad \begin{cases} g_i(\mathbf{x}) = 0 & i = 1, \dots, p, \\ h_j(\mathbf{x}) \geq 0 & j = 1, \dots, q. \end{cases} \quad (4.16)$$

As funções  $g$  são as *restrições de igualdade*, as funções  $h$  são as *restrições de desigualdade*. As restrições podem ser lineares ou não lineares. Em geral, problemas de otimização com restrições são muito mais difíceis de serem resolvidas que problemas de otimização sem restrição. Tem uma vasta teoria que está fora do escopo deste texto. Vamos nos limitar aos casos necessários. Para estudos mais aprofundados recomendam-se [78], [6].

Para resolver o problema de maximizar a variância do vetor mapeado, ou equivalentemente, minimizar o seu valor negativo, temos uma única restrição de igualdade  $\|\boldsymbol{\phi}\| = 1 = \boldsymbol{\phi}^\top \boldsymbol{\phi}$ . Nos moldes do formalismo da (eq. 4.16) temos a restrição  $g_1(\boldsymbol{\phi}) = \boldsymbol{\phi}^\top \boldsymbol{\phi} - 1$ . Então procuramos um vetor unitário  $\boldsymbol{\phi}^* \in \mathbb{R}^m$  como solução do problema de otimização com restrição

$$\boldsymbol{\phi}^* = \arg \max_{\boldsymbol{\phi}} (\boldsymbol{\phi}^\top \Sigma_x \boldsymbol{\phi}) = \arg \min_{\boldsymbol{\phi}} (-\boldsymbol{\phi}^\top \Sigma_x \boldsymbol{\phi}), \quad \text{sujeito a} \quad \boldsymbol{\phi}^\top \boldsymbol{\phi} - 1 = 0. \quad (4.17)$$

Lembre que em um problema de otimização sem restrição, as soluções têm que satisfazer a condição necessária de um gradiente zerado da função objetivo, ou seja  $\nabla f(\mathbf{x}) = \mathbf{0}^\top$ . A técnica dos *Multiplicadores de Lagrange* é uma maneira elegante de incorporar as restrições de igualdade dentro da função objetivo e formular a condição necessária de uma solução também de um gradiente zerado da função associada, a *Lagrangiana*.

**Definição 4.2.2 — Multiplicadores de Lagrange, Função Lagrangiana.** Seja dada a função objetivo  $f(\mathbf{x})$  e  $p$  restrições de igualdade  $g_i(\mathbf{x}) = 0$ ,  $i = 1, \dots, p$ . A função Lagrangiana, ou, simplesmente Lagrangiana, usando  $p$  multiplicadores de Lagrange  $\lambda_i$ ,  $i = 1, \dots, p$  é definida como

$$\mathcal{L}(\mathbf{x}, \lambda_1, \dots, \lambda_p) \stackrel{\text{def}}{=} f(\mathbf{x}) + \sum_{i=1}^p \lambda_i g_i(\mathbf{x}). \quad (4.18)$$

Agrupando os multiplicadores de Lagrange em um único vetor como  $\boldsymbol{\lambda} \stackrel{\text{def}}{=} [\lambda_1 \dots \lambda_i \dots \lambda_p]^\top$ , e as restrições de igualdade em um único vetor como  $\mathbf{g}(\mathbf{x}) \stackrel{\text{def}}{=} [g_1(\mathbf{x}) \dots g_i(\mathbf{x}) \dots g_p(\mathbf{x})]^\top$ , a Lagrangiana em uma forma mais compacta é

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) \stackrel{\text{def}}{=} f(\mathbf{x}) + \boldsymbol{\lambda}^\top \mathbf{g}(\mathbf{x}). \quad (4.19)$$

Zerando o gradiente da Lagrangiana em relação a todos os parâmetros unifica a tentativa de minimizar a função objetivo, e simultaneamente satisfazer as restrições. Temos que zerar o gradiente de dimensão  $(m + p)$ , sendo  $m$  a quantidade de argumentos  $x_1, \dots, x_m$  da função objetivo, e  $p$  a quantidade de restrições  $g_1(\mathbf{x}), \dots, g_p(\mathbf{x})$ . Então

$$\mathbf{0}^\top = \nabla \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = \begin{bmatrix} \partial \mathcal{L} / \partial \mathbf{x} \\ \partial \mathcal{L} / \partial \boldsymbol{\lambda} \end{bmatrix}^\top = \begin{bmatrix} \partial \mathcal{L} / \partial x_1 \\ \vdots \\ \partial \mathcal{L} / \partial x_m \\ \partial \mathcal{L} / \partial \lambda_1 \\ \vdots \\ \partial \mathcal{L} / \partial \lambda_p \end{bmatrix}^\top \quad (4.20)$$

Voltando ao problema de maximização da variância, tendo uma restrição, a Lagrangiana de dimensão  $(m + 1)$  é

$$\mathcal{L}(\boldsymbol{\phi}, \lambda) = -\boldsymbol{\phi}^\top \Sigma_x \boldsymbol{\phi} + \lambda(\boldsymbol{\phi}^\top \boldsymbol{\phi} - 1). \quad (4.21)$$

Zerando o gradiente, temos

$$\begin{aligned} \mathbf{0}^\top &= \nabla \mathcal{L}(\boldsymbol{\phi}, \lambda) = \begin{bmatrix} \partial \left\{ -\boldsymbol{\phi}^\top \Sigma_x \boldsymbol{\phi} + \lambda(\boldsymbol{\phi}^\top \boldsymbol{\phi} - 1) \right\} / \partial \boldsymbol{\phi} \\ \partial \left\{ -\boldsymbol{\phi}^\top \Sigma_x \boldsymbol{\phi} + \lambda(\boldsymbol{\phi}^\top \boldsymbol{\phi} - 1) \right\} / \partial \lambda \end{bmatrix}^\top \\ &\stackrel{2.53e}{=} \begin{bmatrix} -2\boldsymbol{\phi}^\top \Sigma_x + 2\lambda \boldsymbol{\phi}^\top \boldsymbol{\phi} - 1 \\ \mathbf{0}^\top \quad 0 \end{bmatrix}. \end{aligned} \quad (4.22)$$

Foram usadas as regras (eq. 2.53e) e (eq. 2.53f) na pág. 49, junto com a forma simétrica da matriz de covariância  $\Sigma_x$ . A segunda coluna da (eq. 4.22) é simplesmente a restrição de o vetor  $\boldsymbol{\phi}$  ser unitário,  $\|\boldsymbol{\phi}\| = 1$ . A primeira coluna da (eq. 4.22) zerada, dividindo ainda por dois é equivalente a

$$\boldsymbol{\phi}^\top \Sigma_x = \lambda \boldsymbol{\phi}^\top.$$

Transpondo os dois lados da equação pela regra (eq. 2.59c) na pág. 51, e ainda considerando a simetria  $\Sigma_x^\top = \Sigma_x$  da matriz de covariância, temos

$$\Sigma_x \boldsymbol{\phi} = \boldsymbol{\phi} \lambda. \quad (4.23)$$

Esta equação tem a forma “matriz vezes vetor, igual ao mesmo vetor vezes escalar”, identificável como a definição de um autovetor (da direita) e autovalor [3], ou seja, o vetor que maximiza a variância, se a variável aleatória  $\mathbf{x}$  for projetada em cima deste vetor, é um autovetor da matriz de covariância desta variável aleatória. Se a matriz de dimensão  $m \times m$  no problema de autovetores e autovalores é simétrica e contém exclusivamente valores reais  $\sigma_{ij} \in \mathbb{R}$ , existem  $m$  autovalores reais  $\lambda_j \in \mathbb{R}$ ,  $j = 1, \dots, m$ , (sem prova).

Como temos  $m$  vetores  $\boldsymbol{\phi}_j$  na base ortonormal que compõem a matriz  $\Phi$  na (eq. 4.12), a questão é qual desses vetores  $\boldsymbol{\phi}_j$  é responsável pela maior variância  $\sigma_j^2$ . Temos pela (eq. 4.14b)

$$\sigma_j^2 = \boldsymbol{\phi}_j^\top \Sigma_x \boldsymbol{\phi}_j \stackrel{4.23}{=} \boldsymbol{\phi}_j^\top \lambda_j \boldsymbol{\phi}_j = \lambda_j \boldsymbol{\phi}_j^\top \boldsymbol{\phi}_j \stackrel{4.49}{=} \lambda_j. \quad (4.24)$$

Isto significa duas coisas

1. A variância  $\sigma_j^2$  de uma componente  $y_j$  da variável aleatória original  $\mathbf{x}$  projetada em cima do vetor unitário  $\boldsymbol{\phi}_j$  é equivalente ao autovalor  $\lambda_j$  correspondente da solução do problema de autovetor–autovalor  $\Sigma_x \boldsymbol{\phi}_j = \boldsymbol{\phi}_j \lambda_j$  da (eq. 4.23);
2. Em geral existem  $m$  desses autovalores  $\lambda_j$  que normalmente são diferentes um do outro. A estratégia é *ordenar* os autovalores que são o resultado da análise  $\Sigma_x \boldsymbol{\phi}_j = \boldsymbol{\phi}_j \lambda_j$ ,  $j = 1, \dots, m$ , em ordem decrescente

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_j \geq \dots \geq \lambda_m, \quad (4.25)$$

e colocar cada um dos  $m$  autovetores correspondentes na  $j$ -ésima coluna da matriz  $\Phi$  da (eq. 4.12). Podemos ainda estender a relação autovetor–autovalor  $\Sigma_x \boldsymbol{\phi} = \boldsymbol{\phi} \Lambda$  da (eq. 4.23) para todas as  $m$  componentes como

$$\Sigma_x \Phi = \Phi \Lambda, \quad (4.26)$$

onde  $\Lambda$  é uma matriz diagonal que contém os  $m$  autovalores  $\lambda_j$ , ou seja,

$$\Lambda \stackrel{\text{def}}{=} \text{diag}(\lambda_1, \dots, \lambda_m) = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 & 0 \\ 0 & \lambda_2 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & \lambda_{m-1} & 0 \\ 0 & 0 & \dots & 0 & \lambda_m \end{bmatrix}. \quad (4.27)$$

Se ainda por cima a matriz dos autovetores  $\Phi$  não for singular, ela pode ser invertida. Assim temos, multiplicando a (eq. 4.26) no lado direito pela inversa  $\Phi^{-1} = \Phi^\top$

$$\Sigma_x = \Phi \Lambda \Phi^\top, \quad (4.28)$$

considerando a (eq. 4.10). Assim se define a decomposição em autovalores e autovetores da matriz de covariância.

Vamos provar que as componentes  $y_j$  do vetor mapeado  $\mathbf{y} = \Phi^\top \mathbf{x}$  realmente não têm mais covariância, ou seja temos  $\sigma_{ij} = 0$  para os elementos fora da diagonal da matriz de covariância  $\Sigma_y$  da variável aleatória  $\mathbf{y}$ .

*Demonstração.*

$$\Sigma_y \stackrel{4.50}{=} \Phi^\top \Sigma_x \Phi \stackrel{4.26}{=} \Phi^\top \Phi \Lambda \stackrel{4.9}{=} I \Lambda = \Lambda. \quad (4.29)$$

■

Podemos constatar que a matriz de covariância  $\Sigma_y$  da variável aleatória  $\mathbf{y}$  mapeada pela matriz  $\Phi$  composta pelos autovetores da matriz de covariância  $\Sigma_x$  da variável aleatória original  $\mathbf{x}$  é uma matriz diagonal  $\Lambda$ , composta pelos autovalores de  $\Sigma_x$ .

Podemos definir um algoritmo para descorrelacionar linearmente as componentes de um vetor  $\mathbf{x}$  que representa uma variável aleatória, veja o algoritmo 9. O passo do cálculo dos autovalores e autovetores de uma matriz simétrica de valores reais pode ser feito pelo *Algoritmo de autovalores de Jacobi*, [1].

Obs.: A decomposição da matriz de covariância da (eq. 4.28) é um caso especial da decomposição em valores singulares (*Singular Value Decomposition*, SVD) que decompõe qualquer transformação linear  $A$  da (eq. 4.4) como

$$A = U S V^\top, \quad (4.30)$$

**Algoritmo 9:** Descorrelação Linear de Dados

**Entrada:** Conjunto de  $n$  padrões  $\mathbf{x}_i \in \mathbb{R}^m$  de dimensão  $m$ , organizados em uma matriz de dados  $X$  de dimensão  $n \times m$ . Cada padrão em forma transposta  $\mathbf{x}_i^\top$  é uma linha de  $X$ .

**Resultado:** Matriz de descorrelação linear  $\Phi$  de dimensão  $m \times m$ ; Eventualmente conjunto de  $n$  padrões  $\mathbf{y}_i \in \mathbb{R}^m$  de dimensão  $m$ , descorrelacionadas linearmente, organizados em uma matriz de dados  $Y$  de dimensão  $n \times m$ . Cada padrão em forma transposta  $\mathbf{y}_i^\top$  é uma linha de  $Y$ ,  $i = 1, \dots, n$ .

- 1 Estime, pela (eq. 1.33) na pág. 23, o vetor de média  $\hat{\boldsymbol{\mu}}_x$  dos dados e substitua a matriz de dados  $X$  pela sua versão centrada;
- 2 Estime, pela (eq. 1.45) na pág. 27, a matriz de covariância  $\hat{\Sigma}_x$  da (eq. 1.43) na pág. 27;
- 3 Calcule todos os  $m$  autovalores  $\lambda_j$  e os autovetores  $\boldsymbol{\phi}_j$  correspondentes da matriz de covariância  $\hat{\Sigma}_x$ ;
- 4 Ordene os autovalores em ordem decrescente, conforme a (eq. 4.25), junto com os autovetores correspondentes;
- 5 Componha a matriz de descorrelação  $\Phi$ , colocando os autovetores ordenados como colunas;
- 6 Eventualmente faça a descorrelação dos dados, mapeando a matriz de dados como  $Y = X\Phi$ ;

onde a matriz  $A$  tem dimensão  $p \times q$ , a matriz  $S = \text{diag}(\lambda_1, \dots, \lambda_r)$  contém  $r$  valores singulares,  $U$  é uma matriz de dimensão  $p \times r$  de vetores  $\mathbf{u}_i, i = 1, \dots, r$  ortonormais de dimensão  $p$  como colunas, e  $V$  é uma matriz de dimensão  $q \times r$  de vetores  $\mathbf{v}_i, i = 1, \dots, r$  ortonormais de dimensão  $q$  como colunas.

■ **Exemplo 4.4** A figura 4.2 mostra um exemplo de descorrelacionar uma variável aleatória. Como este método pertence ao grupo de técnicas de aprendizagem não supervisionado, não existem rótulos de classe. Como conjunto de dados vamos escolher exclusivamente as 50 flores da classe versicolor, ainda limitando o número de características a dois, escolhendo as primeiras duas  $x_1$ , comprimento de sépala e  $x_2$ , largura de sépala, veja a distribuição dos dados na figura 1.2a. Além disso, todas as amostras que aparecem mais que uma vez no subespaço  $(x_1, x_2)$  foram eliminados, sobrando 44 pontos  $\mathbf{x}$ . A média (eq. 1.33) e a matriz de covariância estimadas (eq. 1.45) desses pontos são

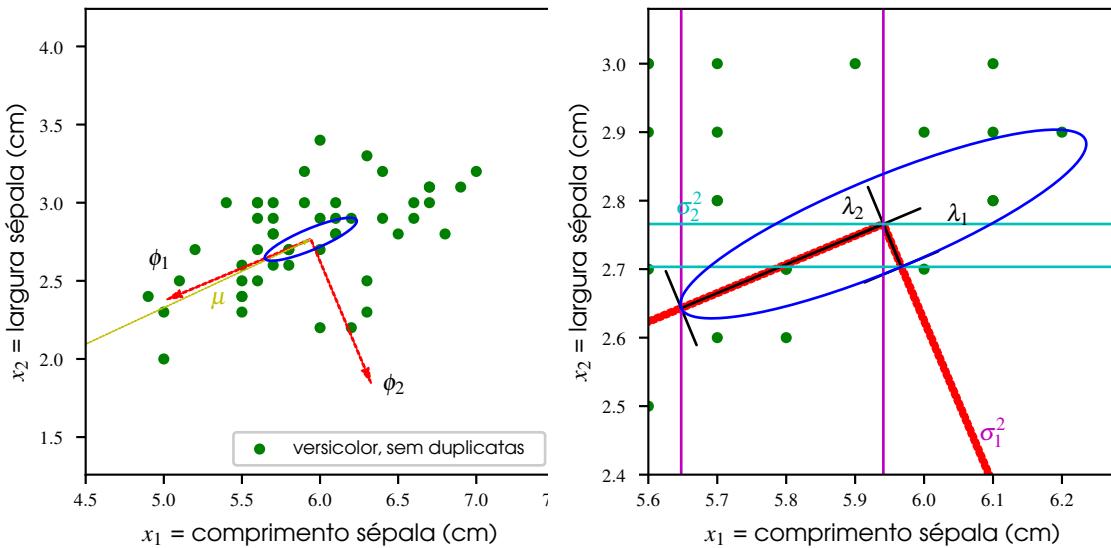
$$\hat{\boldsymbol{\mu}}_x = \begin{bmatrix} 5.941 \\ 2.766 \end{bmatrix}, \quad \hat{\Sigma}_x = \begin{bmatrix} 0.281 & 0.090 \\ 0.090 & 0.105 \end{bmatrix}.$$

As matrizes dos autovetores e autovalores são

$$\Phi = [\boldsymbol{\phi}_1 \ \boldsymbol{\phi}_2] = \begin{bmatrix} -0.922 & 0.386 \\ -0.386 & -0.922 \end{bmatrix}, \quad \Lambda = \Sigma_y = \begin{bmatrix} 0.318 & 0 \\ 0 & 0.068 \end{bmatrix}.$$

A elipse azul na figura 4.2 tem como semieixo maior o primeiro autovalor  $\lambda_1 = 0.318$  e como semieixo menor o segundo autovalor  $\lambda_2 = 0.068$ . Na figura 4.2b, as variâncias  $\sigma_1^2$  e  $\sigma_2^2$  no espaço original da variável aleatória  $\mathbf{x}$  são comparados com as variâncias  $\lambda_1$  e  $\lambda_2$  no novo espaço da variável aleatória  $\mathbf{y}$ . ■

Em duas ou três dimensões existe uma analogia interessante com a Computação Gráfica. Centrar os dados é equivalente a uma translação da variável aleatória pelo valor da média negativa. A transformação gráfica da translação é definida por somar um vetor a uma coordenada. A matriz



(a) Autovetores  $\phi_1$  e  $\phi_2$  e elipse definida pelos autovalores como semieixos.

(b) Figura aumentada, destacando as variâncias  $\sigma_1^2$  e  $\sigma_2^2$  da variável original  $x$  e os autovalores que coincidem com as variâncias da variável mapeada  $y$ .

Figura 4.2: Espaço de características da primeiras duas características de Iris. Os dados são da classe versicolor, eliminando os padrões que aparecem mais que uma vez neste espaço.

de descorrelação  $\Phi$  é uma matriz de rotação. Em duas dimensões, como no exemplo 4.4, a matriz tem a forma

$$\Phi = R_\theta \stackrel{\text{def}}{=} \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix},$$

em que  $\theta$  é o ângulo de rotação em torno do eixo  $\mathbf{e}_z = [0 \ 0 \ 1]^\top$ . Um ponto qualquer  $\mathbf{x}$  no plano definido pelos vetores  $\mathbf{e}_x = [1 \ 0 \ 0]^\top$  e  $\mathbf{e}_y = [0 \ 1 \ 0]^\top$  é rotacionado em torno de  $\mathbf{e}_z$  pelo ângulo  $\theta$  para o novo ponto  $\mathbf{x}'$  por pré-multiplicação da matriz como

$$\mathbf{x}' = \Phi \mathbf{x}.$$

Este ângulo pode ser obtido do cosseno e seno do ângulo, ainda fazendo uma distinção de casos, em qual dos quatro quadrantes o primeiro autovetor se encontra. Nas linguagens de programação usa-se a função `atan2` (ou `arctan2`). Neste caso<sup>1</sup> obtém-se  $\theta = -157.28^\circ$ . A elipse na figura 4.2 poderia então ser colocado em posição horizontal pela operação inversa, rotacionando pelo ângulo  $-\theta$ . A matriz  $\Lambda$  é uma matriz de escalamento

$$\Lambda = S_{s_x, s_y} \stackrel{\text{def}}{=} \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix},$$

que aplica um escalamento não uniforme, se os dois autovalores forem diferentes. Podemos aplicar a transformação da elipse para um círculo, escalando pelos fatores inversos, usando a matriz

$$\Lambda^{-1/2} = S_{1/\sqrt{s_x}, 1/\sqrt{s_y}} = \begin{bmatrix} 1/\sqrt{s_x} & 0 \\ 0 & 1/\sqrt{s_y} \end{bmatrix}.$$

<sup>1</sup>Usando a biblioteca `numpy` da linguagem Python, a chamada é `numpy.arctan2(sin θ, cos θ)`, onde o primeiro argumento  $-0.386$  é a segunda componente do autovetor  $\phi_1$ , e o segundo argumento  $-0.922$  é a primeira componente de  $\phi_1$ .

Observe que os autovalores são variâncias, portanto para dividir pelo desvio padrão temos que aplicar a raiz quadrada, veja a (eq. 1.37) na pág. 25.

Então, se uma variável aleatória  $\tilde{x}$  sofre um escalamento, seguindo por uma rotação para produzir a variável aleatória  $\mathbf{x}$ , a transformação inversa  $\Phi\Lambda^{-1/2}$  desfaz essas duas operações, constituindo a transformação de *branqueamento*, [63]. Essa operação é geral para qualquer dimensão. Note ainda que uma sequência de transformações lineares corresponde a uma multiplicação de matrizes, em que cada matriz representa uma transformação.

### Extração de Componentes Principais

A partir da descorrelação, a extração de componentes principais [58] é um passo trivial. Cada componente  $y_j$  da variável aleatória mapeada  $\mathbf{y}$  tem uma variância  $\lambda_j$  igual a um dos autovalores da matriz de covariância dos dados originais. A primeira componente  $y_1$  tem a maior variância, ela é a primeira componente principal. A segunda componente  $y_2$  tem a segunda a maior variância, ela é a segunda componente principal, e assim por diante.

**Definição 4.2.3 — Variação Acumulativa na Extração de Componentes Principais.** As variâncias de cada componente da variável aleatória descorrelacionada  $\mathbf{y}$  de dimensão  $m$  representam como os dados mudam o seu valor nessa dimensão projetada  $y_j$ . Com os autovetores e autovalores ordenados, podemos definir uma *variação acumulativa* começando com a primeira componente, até uma componente  $y_\ell$

$$t_\ell \stackrel{\text{def}}{=} \frac{\sum_{j=1}^\ell \lambda_j}{\sum_{j=1}^m \lambda_j}, \quad (4.31)$$

onde  $m$  é o número total de componentes. Obviamente, se todos as componentes forem usadas, temos  $t_m = 100\%$ .

Esse critério frequentemente é usado para definir a quantidade de componentes principais, a quantidade  $\ell$  de componentes usadas subsequentemente, e a quantidade  $m - \ell$  de componentes descartadas.

**Definição 4.2.4 — Extrator de Componentes Principais.** O extrator das primeiras  $\ell$  de um total de  $m$  componentes principais é uma matriz de dimensão  $m \times \ell$  composta pelas primeiras  $\ell$  colunas da matriz de descorrelacionamento linear, ou seja, pelos primeiros  $\ell$  autovetores  $\boldsymbol{\phi}_1, \dots, \boldsymbol{\phi}_\ell$ , associados aos primeiros maiores autovalores  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_\ell$

$$\Phi_\ell \stackrel{\text{def}}{=} [\boldsymbol{\phi}_1 \ \dots \ \boldsymbol{\phi}_j \ \dots \ \boldsymbol{\phi}_\ell] = \begin{bmatrix} \phi_{1,1} & \dots & \phi_{j,1} & \dots & \phi_{\ell,1} \\ \vdots & & \vdots & & \vdots \\ \phi_{1,i} & \dots & \phi_{j,i} & \dots & \phi_{\ell,i} \\ \vdots & & \vdots & & \vdots \\ \phi_{1,m} & \dots & \phi_{j,m} & \dots & \phi_{\ell,m} \end{bmatrix}. \quad (4.32)$$

A variável aleatória  $\mathbf{y}$  de dimensão  $\ell \leq m$  extraída pelo extrator das componentes principais é

$$\mathbf{y} = \Phi_\ell^\top \mathbf{x} = \begin{bmatrix} \boldsymbol{\phi}_1^\top \\ \vdots \\ \boldsymbol{\phi}_j^\top \\ \vdots \\ \boldsymbol{\phi}_\ell^\top \end{bmatrix} \mathbf{x} = \begin{bmatrix} \boldsymbol{\phi}_1^\top \mathbf{x} \\ \vdots \\ \boldsymbol{\phi}_j^\top \mathbf{x} \\ \vdots \\ \boldsymbol{\phi}_\ell^\top \mathbf{x} \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_j \\ \vdots \\ y_\ell \end{bmatrix}, \quad (4.33)$$

As restantes  $(m - \ell)$  componentes, aquelas *não* principais, formam o *residual*

$$\hat{\mathbf{y}} \stackrel{\text{def}}{=} \begin{bmatrix} \boldsymbol{\phi}_{\ell+1}^T \\ \vdots \\ \boldsymbol{\phi}_m^T \end{bmatrix} \mathbf{x} = \begin{bmatrix} \boldsymbol{\phi}_{\ell+1}^T \mathbf{x} \\ \vdots \\ \boldsymbol{\phi}_m^T \mathbf{x} \end{bmatrix} = \begin{bmatrix} y_{\ell+1} \\ \vdots \\ y_m \end{bmatrix}, \quad (4.34)$$

Podemos então separar dois subespaços, o das componentes principais  $\mathbb{R}^\ell$  e o dos residuais  $\mathbb{R}^{m-\ell}$ . Ainda podemos definir a variável aleatória *reconstruída*  $\tilde{\mathbf{x}}$  que aplica a transformação inversa do subespaço das PC de volta para o espaço original da variável aleatória  $\mathbf{x}$ , porém sem a parte das componentes residuais, ou seja

$$\tilde{\mathbf{x}} = \Phi_\ell \mathbf{y} = \Phi_\ell \Phi_\ell^T \mathbf{x}. \quad (4.35)$$

Lembre que  $\mathbf{x}$  foi centrado antes de aplicar a PCA, e eventualmente tem que ser recolocado após a reconstrução, somando a média  $\boldsymbol{\mu}$  subtraída antes de mapear para  $\mathbf{y}$ . A matriz  $\Phi_\ell \Phi_\ell^T$  tem a dimensão  $(m \times \ell) \cdot (\ell \times m) = m \times m$ , consequentemente a reconstrução  $\tilde{\mathbf{x}}$  e o original  $\mathbf{x}$  são do mesmo espaço  $\mathbb{R}^m$ .

Voltando ao exemplo 4.4, temos poucas opções para escolher a quantidade de componentes principais. Como usamos somente  $m = 2$  duas características originais, a quantidade  $\ell$  de componentes principais pode ser ou um, ou dois. Para a variabilidade da definição 4.2.3 temos  $t_1 = \lambda_1 / (\lambda_1 + \lambda_2) = 0.318 / (0.318 + 0.068) = 82.4\%$ , e para  $t_2 = (\lambda_1 + \lambda_2) / (\lambda_1 + \lambda_2) = 100\%$  obviamente. No caso de uma componente principal temos na figura 4.2 como o subespaço das componentes principais o vetor  $\boldsymbol{\phi}_1$  e como espaço residual o vetor  $\boldsymbol{\phi}_2$ .

Umas das aplicações típicas da extração de componentes principais é a visualização de dados em duas ou três dimensões, usando as primeiras duas ou três componentes. Para abreviar a nomenclatura vamos usar o acrônimo “PC” (*Principal Components*) para componentes principais e, como já antes usado, “PCA” (*Principal Components Analysis*) para análise de componentes principais. Podemos unir todas as flores Iris em uma única matriz de dados  $X$  com 150 pontos  $\mathbf{x}_i, i = 1, \dots, 150$ , sem considerar a qual classe uma flor pertence. Já estimamos a média na (eq. 1.34) na pág. 24 e a matriz de covariância na (eq. 1.46a) na pág. 27. A PCA produz as matrizes

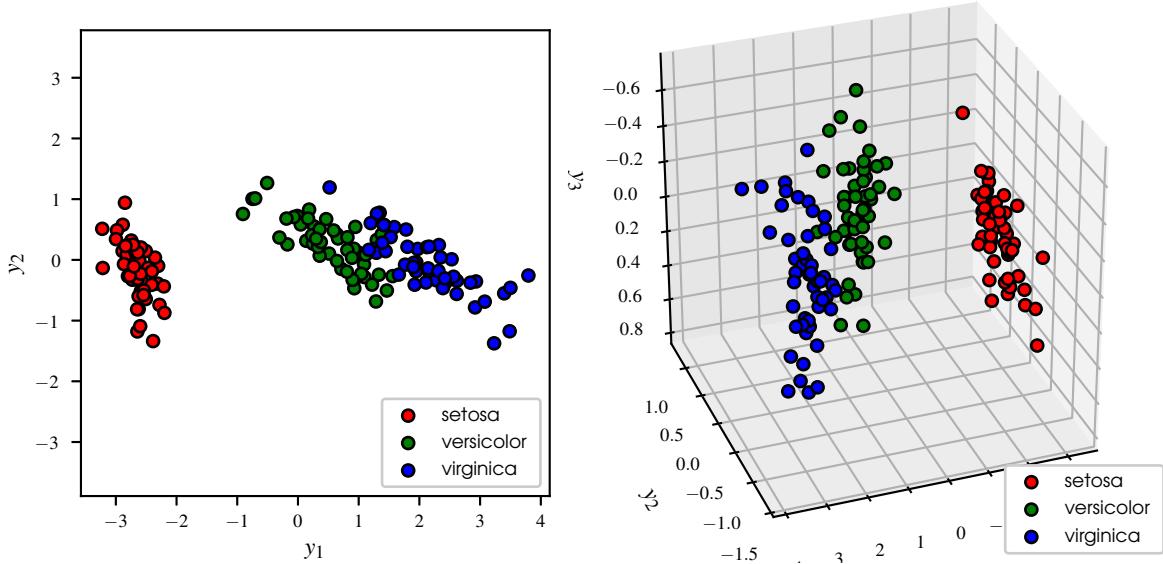
$$\Phi = \begin{bmatrix} 0.361 & -0.085 & 0.857 & 0.358 \\ -0.657 & -0.730 & 0.173 & 0.075 \\ -0.582 & 0.598 & 0.076 & 0.546 \\ 0.315 & -0.320 & -0.480 & 0.754 \end{bmatrix}, \quad \Lambda = \begin{bmatrix} 4.228 & 0 & 0 & 0 \\ 0 & 0.243 & 0 & 0 \\ 0 & 0 & 0.078 & 0 \\ 0 & 0 & 0 & 0.024 \end{bmatrix}$$

e a variação acumulativa (eq. 4.31) é  $t_1 = 92.46\%, t_2 = 97.77\%, t_3 = 99.48\%, t_4 = 100.00\%$ .

A figura 4.3 mostra a PC em duas e três dimensões. Em  $\ell = 2$  dimensões o gráfico na figura 4.3a a escala dos dois eixos é igual, revelando a maior variação na componente  $y_1$ .

A figura 4.3a revela outro fato importante sobre o efeito da extração das PC. Se considerarmos a componente  $y_2$ , eixo vertical nessa figura, podemos constatar que essa componente é completamente inapropriada para distinguir as classes. A projeção das 150 pontos em cima do eixo  $y_2$  destrói por completo o poder discriminativo dessa variável em relação a separabilidade das três classes de flores. Isto significa que o critério da escolha das PC não tem preocupação em discernir classes. Obviamente essa informação da classe não está disponível, e consequentemente será totalmente ignorada no processo da PCA.

Uma aplicação interessante na área de processamento de imagens são os *eigenfaces*, ou autofaces [101, 110] que tratam uma imagem digital como um padrão  $\mathbf{x}$ . A ideia é condensar a essência linear de um conjunto de imagens nas PC. Em termos de visualização, podemos considerar uma reconstrução (eq. 4.35) de uma face com um conjunto reduzido de PC. Considere a figura 4.4. No



(a) As primeiras duas componentes de todas as 150 flores, desconsiderando a classe.

(b) As primeiras três componentes de todas as 150 flores, desconsiderando a classe.

Figura 4.3: Subespaços das primeiras duas e três PC. A cor do ponto é mera informação da origem da flor. Não é considerado na PCA.

contexto de experimentos de reconhecimento de pessoas foram colecionados imagens de políticos da primeira década dos anos 2000, [54]. As imagens de níveis de cinza entre os valores zero e 255 inclusive, possuem uma resolução baixa de altura  $h = 62$  e largura  $w = 47$ , resultando em  $h \times w = 2914$  píxeis. Seja  $z(p, q) \in \{0, \dots, 255\}$ ,  $p = 1, \dots, h$ ,  $q = 1, \dots, w$  o valor do píxel na  $p$ -ésima linha e  $q$ -ésima coluna. A conversão entre imagem e padrão  $\mathbf{x}$  é feito via linearização da imagem. O píxel  $(p, q)$  é colocado como  $i$ -ésima componente  $x_i$ , na posição

$$i = (p - 1) \cdot w + q$$

do padrão, representado pelo vetor  $\mathbf{x} \in \mathbb{R}^{w \cdot h}$  antes do processamento. Após a geração de um vetor reconstruído  $\tilde{\mathbf{x}}$ , a componente  $\tilde{x}_i$  é recolocada na sua posição

$$(p, q) = \left( \left\lfloor \frac{i - 1}{w} \right\rfloor, i - (p - 1)w \right),$$

calculando primeiro a linha  $p$  e depois a coluna  $q$ .

A imagem na figura 4.4 mostra a classe “Gerhard Schroeder”, chanceler da Alemanha de 1998 a 2005. A primeira imagem é a média  $\mu$  de  $n = 109$  imagens dessa classe, simplesmente colocando na posição  $(p, q)$  a média dos píxeis na posição correspondente de todas as 109 imagens. As restantes imagens mostram as reconstruções  $\tilde{\mathbf{x}} = \Phi_\ell \Phi_\ell^\top \mathbf{x}$  com números crescentes  $\ell = 1, 5, 10, 50, \ell_{\max}$  de PC, sendo a última imagem a original com o máximo de  $\ell_{\max}$  componentes, onde  $\ell_{\max} = \min\{n, m\} = \min\{109, 2914\} = 109$ . Note que a quantidade máxima  $\ell_{\max}$  de PC não pode ultrapassar o mínimo da quantidade  $n$  de padrões e da dimensão  $m$  do vetor. Caso contrário, a matriz de covariância fica singular (sem prova). A imagem média, como esperado suaviza os traços da face. A variação acumulativa com somente  $\ell = 1$  uma componente já atinge  $t_1 = 24.32\%$ , já deixando perceber traços característicos da pessoa, enquanto o uso de todas as  $\ell_{\max} = 109$  componentes reconstrói a imagem original.

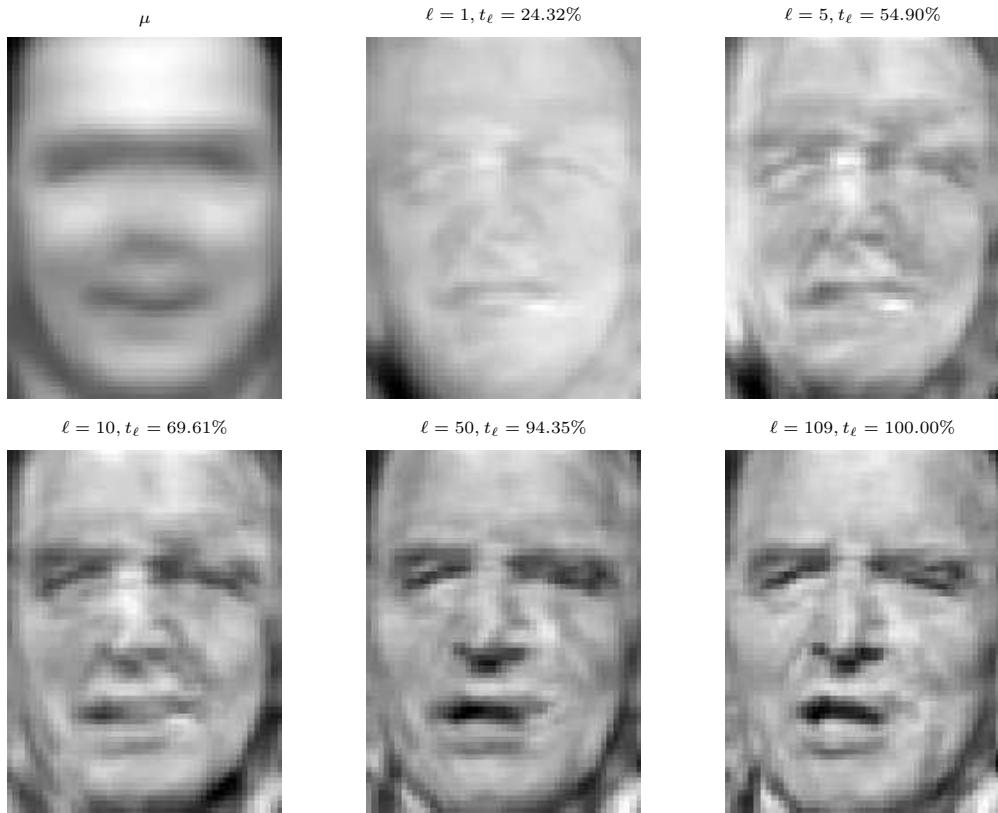


Figura 4.4: Imagem média e reconstrução de imagem, usando quantidade de PC diferentes.

### 4.3 Extração Não Linear de Características

Já vimos na introdução o exemplo do Índice de Massa Corporal como exemplo de extração de características não linear, ou seja, o vetor composto pelas novas características é obtido por uma combinação não linear das características originais. De fato, todos os métodos de aprendizado de máquina sofisticados reduzem-se a esse técnico. Uma rede neural no contexto de aprendizagem profunda, em cada camada executa uma extração de características não linear, combinando as informações da camada anterior, recombinando as características, e passando adiante. Dessa maneira, a camada é um extrator no sentido da definição 4.1.2. Apresentar todos os métodos que calculam novas características a partir de antigas características, ou diretamente de um sinal, não é possível. Vamos nos limitar a algumas arquiteturas interessantes que são de interesse acadêmico e/ou mostraram resultados julgados bons.

#### 4.3.1 Máquina Extrema de Aprendizado, *Extreme Learning Machine, ELM*

Esta arquitetura, considerada uma rede neural artificial com uma camada oculta já foi mencionada brevemente na seção 2.1.3 na pág. 46. A ideia fundamental da ELM é muito simples, tão simples que o autor do nome ELM [56, 57], Guang Bin Huang, foi bastante hostilizado, alegando plágio de ideias antigas. Não vamos entrar nesse mérito, apresentando somente a teoria. Neste ponto, recomenda-se refrescar a teoria da regressão linear baseada na Máquina Linear (ML) da seção 2.2 na pág. 53, pois a extensão para a teoria da ELM torna-se muito fácil, com a Máquina Linear dominada. Vamos acrescentar à definição da ML o conceito de mapeamento de um domínio de entrada para um domínio de saída.

**Definição 4.3.1 — Máquina Linear (revista), compare a definição 2.2.1.** Seja  $\mathcal{X} = \mathbb{R}^{m+1}$  o domínio de entrada de vetores

$$\mathbf{x} = [1 \ x_1 \ \cdots \ x_j \ \cdots \ x_m]^T$$

de dimensão aumentada  $(m+1)$ , e seja  $\mathcal{Y} = \mathbb{R}^c$  o contradomínio de saída de vetores

$$\mathbf{y} = [y_1 \ \cdots \ y_k \ \cdots \ y_c]^T$$

de dimensão  $c$ . A função vetorial  $\mathbf{y}(\mathbf{x})$  mapeia o vetor de entrada para o vetor de saída por uma função linear

$$\begin{aligned} \mathbf{y} : \mathcal{X} &\rightarrow \mathcal{Y} \\ \mathbf{x} \mapsto \mathbf{y}(\mathbf{x}) &= W^T \mathbf{x} = \left[ \mathbf{w}_1^T \mathbf{x} \ \cdots \ \mathbf{w}_k^T \mathbf{x} \ \cdots \ \mathbf{w}_c^T \mathbf{x} \right]^T \\ &= [y_1(\mathbf{x}) \ \cdots \ y_k(\mathbf{x}) \ \cdots \ y_c(\mathbf{x})]^T, \end{aligned} \quad (4.36)$$

em que a  $k$ -ésima função

$$y_k(\mathbf{x}) = \mathbf{w}_k \cdot \mathbf{x}$$

é o produto escalar da  $k$ -ésima coluna da matriz

$$W = [\mathbf{w}_1 \ \cdots \ \mathbf{w}_k \ \cdots \ \mathbf{w}_c].$$

Para definir a ELM, vamos introduzir um mapeamento intermediário  $\phi$  que primeiro mapeia a entrada  $\mathbf{x}$  para um vetor de características  $\phi(\mathbf{x})$  no sentido do extrator de características da definição 4.1.2 na pág. 127 e depois mapeia esse vetor intermediário para o vetor de saída  $\mathbf{y}$ .

**Definição 4.3.2 — Rede Neural de Alimentação Adiante com uma Camada Oculta Não Linear e Camada de Saída Linear.** Seja  $\mathcal{X} = \mathbb{R}^{m+1}$  o domínio de entrada de vetores

$$\mathbf{x} = [1 \ x_1 \ \cdots \ x_j \ \cdots \ x_m]^T$$

de dimensão aumentada  $(m+1)$ , seja  $\Phi = \mathbb{R}^H$  um domínio intermediário de vetores

$$\phi = [\phi_1 \ \cdots \ \phi_h \ \cdots \ \phi_H]^T$$

de dimensão  $H$ , e seja  $\mathcal{Y} = \mathbb{R}^c$  o domínio de saída de vetores

$$\mathbf{y} = [y_1 \ \cdots \ y_k \ \cdots \ y_c]^T$$

de dimensão  $c$ . A função vetorial  $\phi(\mathbf{x})$  mapeia o vetor de entrada para o vetor intermediário  $\phi$  por um mapeamento não linear. A função vetorial  $\mathbf{y}(\phi(\mathbf{x}))$  mapeia o vetor intermediário para

o vetor de saída por uma função linear

$$\begin{aligned}
 \phi : \mathcal{X} &\rightarrow \Phi \\
 y : \Phi &\rightarrow \mathcal{Y} \\
 \mathbf{x} &\mapsto \phi(\mathbf{x}) \mapsto y(\phi(\mathbf{x})) \\
 &= W^\top \phi(\mathbf{x}) \\
 &= \left[ \mathbf{w}_1^\top \phi(\mathbf{x}) \quad \cdots \quad \mathbf{w}_k^\top \phi(\mathbf{x}) \quad \cdots \quad \mathbf{w}_c^\top \phi(\mathbf{x}) \right]^\top \\
 &= [y_1(\phi(\mathbf{x})) \quad \cdots \quad y_k(\phi(\mathbf{x})) \quad \cdots \quad y_c(\phi(\mathbf{x}))]^\top,
 \end{aligned} \tag{4.37}$$

em que a  $k$ -ésima função

$$y_k(\phi(\mathbf{x})) = \mathbf{w}_k \cdot \phi(\mathbf{x})$$

é o produto escalar da  $k$ -ésima coluna da matriz

$$W = [\mathbf{w}_1 \quad \cdots \quad \mathbf{w}_k \quad \cdots \quad \mathbf{w}_c].$$

Falta especificar a não linearidade do mapeamento intermediário  $\phi$ . A não linearidade é obrigatória. Caso contrário, o cálculo da saída seria novamente linear, pois uma função linear de uma função linear permanece linear. Isso também é facilmente visível na multiplicação de matrizes, pois o produto de duas matrizes que incorporam o mapeamento linear, novamente é uma matriz, portanto, representa um mapeamento linear global. Nesta arquitetura da rede com alimentação adiante, normalmente o cálculo da entrada até a camada oculta é uma combinação linear  $\mathbf{w} \cdot \mathbf{x} + b$ , seguida por uma função de ativação  $z(\mathbf{w} \cdot \mathbf{x} + b)$  não linear, responsável pela introdução da não linearidade na sequência dos mapeamentos. Essa ideia já foi ilustrada na figura 2.2 na pág. 43. As funções de ativação serão apresentadas posteriormente. A mais usada é função sigmoidal logística

$$z(a) = 1/(1 + e^{-a}),$$

brevemente mencionada na (eq. 2.45) na pág. 45. A função vetorial  $\phi(\mathbf{x})$  intermediária é composta por  $H$  funções  $\phi_h(\mathbf{x})$ ,  $h = 1, \dots, H$ , calculando o vetor

$$\begin{aligned}
 \phi(\mathbf{x}) &= [\phi_1(\mathbf{x}) \quad \cdots \quad \phi_h(\mathbf{x}) \quad \cdots \quad \phi_H(\mathbf{x})]^\top \\
 &= [z(\tilde{\mathbf{w}}_1 \cdot \mathbf{x}) \quad \cdots \quad z(\tilde{\mathbf{w}}_h \cdot \mathbf{x}) \quad \cdots \quad z(\tilde{\mathbf{w}}_H \cdot \mathbf{x})]^\top.
 \end{aligned} \tag{4.38}$$

Foi sobreescrito o símbolo de tilde “~” para distinguir os pesos  $\mathbf{w}$  do mapeamento intermediário da camada oculta até a saída dos pesos  $\tilde{\mathbf{w}}$  da entrada até a camada oculta. Cada combinação linear do vetor de entrada  $\mathbf{x}$  com o  $h$ -ésimo vetor de pesos  $\tilde{\mathbf{w}}_h$  é calculado como

$$\tilde{\mathbf{w}}_h \cdot \mathbf{x} = \sum_{j=1}^m \tilde{w}_{j,h} x_j + b_h, \quad h = 1, \dots, H. \tag{4.39}$$

Como de costume, o viés  $b_h$  equivale ao 0-ésimo peso  $w_{0,h}$ . Lembre que pela convenção da (eq. 2.66) na pág. 54, o primeiro índice do peso  $w_{j,h}$  é a origem, o segundo índice é o destino do fluxo de processamento da informação. Neste caso, a origem é a  $j$ -ésima entrada  $x_j$ , o destino é o  $h$ -ésimo neurônio oculto. A figura 4.5 mostra a arquitetura resultante. A camada oculta realiza a extração de novas características  $\phi_h, h = 1, \dots, H$  de natureza não linear. Essas novas

características em seguida sujeitam-se à conhecida combinação linear da Máquina Linear.

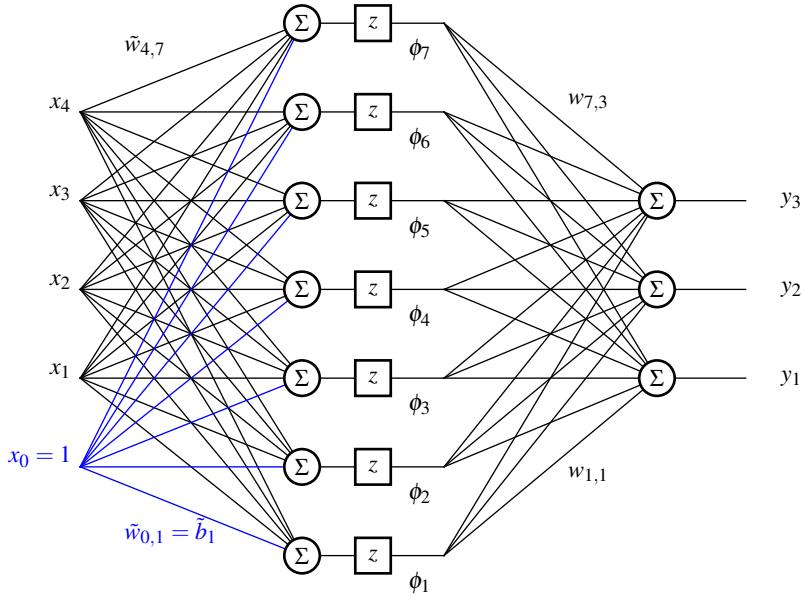


Figura 4.5: Rede neural com uma camada oculta não linear e a camada de saída linear. O exemplo tem  $(m+1) = (4+1) = 5$  entradas  $x_0, \dots, x_m$  e  $c = 3$  saídas  $y_1, \dots, y_c$ , tal como a Máquina Linear da figura 2.4 na pág. 55. Na camada oculta,  $H = 7$  neurônios foram introduzidos que primeiro calculam a combinação linear das entradas e depois passam o produto escalar por uma função de ativação não linear. A quantidade de pesos  $\tilde{w}_{j,h}$ ,  $j = 0, \dots, m$ ,  $h = 1, \dots, H$ , da entrada a camada oculta é  $(m+1) \times H = (4+1) \times 7 = 35$ , armazenados em uma matriz  $\tilde{W}$  de dimensão  $5 \times 7$ . A quantidade de pesos  $w_{h,k}$ ,  $h = 1, \dots, H$ ,  $k = 1, \dots, c$  da camada oculta até a camada de saída é  $H \times c = 7 \times 3 = 21$ , armazenados em uma matriz  $W$  de dimensão  $7 \times 3$ .

A arquitetura desta rede já é uma instanciação do Perceptron Multicamadas, *Multi-Layer Perceptron*, *MLP*, em princípio o modelo mais importante das redes neurais artificiais. Com ele é possível aproximar qualquer função não linear. O princípio de treinamento é aprendizagem supervisionada. A descida de gradiente é o método de otimização, pois não é mais possível definir uma solução determinística para obter os pesos, como no caso da Máquina Linear. O que nos falta para entender completamente o MLP, é a aplicação da regra da cadeia, descendo as camadas em ordem inversa, desde a saída até a última camada oculta (se tiver mais que uma camada oculta). Esse procedimento é a retropropagação de erro, *Error Backpropagation*. Retornaremos mais tarde ao MLP.

A definição do MLP torna a definição da ELM bastante simples. Basta definir a natureza dos pesos da entrada até a camada oculta.

#### Definição 4.3.3 — Máquina Extrema de Aprendizado, *Extreme Learning Machine*, *ELM*.

A ELM é uma rede neural de alimentação adiante com uma camada oculta com processamento não linear e camada de saída linear da definição 4.3.2, em que os pesos entre a entrada e camada oculta são aleatórios. Usando uma distribuição normal padrão com média zero e variância

unitária, temos

$$\tilde{w}_{j,h} \sim \mathcal{N}(0, 1), \quad j = 0, \dots, m, \quad h = 1, \dots, H \quad (4.40)$$

Todas os pesos da entrada até a camada oculta podem ser armazenados em uma matriz

$$\tilde{W} = [\tilde{\mathbf{w}}_1 \ \dots \ \tilde{\mathbf{w}}_h \ \dots \ \tilde{\mathbf{w}}_H] = \begin{bmatrix} \tilde{w}_{0,1} & \dots & \tilde{w}_{0,h} & \dots & \tilde{w}_{0,H} \\ \tilde{w}_{1,1} & \dots & \tilde{w}_{1,h} & \dots & \tilde{w}_{1,H} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \tilde{w}_{j,1} & \dots & \tilde{w}_{j,h} & \dots & \tilde{w}_{j,H} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \tilde{w}_{m,1} & \dots & \tilde{w}_{m,h} & \dots & \tilde{w}_{m,H} \end{bmatrix}. \quad (4.41)$$

Todas os pesos da camada oculta até a camada de saída podem ser armazenados em uma matriz

$$W = [\mathbf{w}_1 \ \dots \ \mathbf{w}_k \ \dots \ \mathbf{w}_c] = \begin{bmatrix} w_{1,1} & \dots & w_{1,k} & \dots & w_{1,c} \\ w_{2,1} & \dots & w_{2,k} & \dots & w_{2,c} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ w_{h,1} & \dots & w_{h,k} & \dots & w_{h,c} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ w_{H,1} & \dots & w_{H,k} & \dots & w_{H,c} \end{bmatrix}. \quad (4.42)$$

Esta arquitetura implica que

1. Os  $(m+1) \times H$  pesos  $\tilde{w}_{j,h}$  entre a entrada e camada oculta precisam ser inicializadas aleatoriamente;
2. Esta inicialização precisa ser feita somente uma única vez, ou seja, os pesos  $\tilde{w}_{j,h}$  ficam “congelados” durante todo o resto da aprendizagem;
3. Os  $H \times c$  pesos  $w_{h,k}, k = 1, \dots, c, h = 1, \dots, H$  da camada oculta até a camada de saída podem ser obtidos deterministicamente da mesma maneira como na Máquina Linear, usando o produto da pseudoinversa da matriz de dados mapeada para o espaço intermediário, com a matriz dos alvos, descrito na seção 2.2.3 na pág. 54;
4. A ELM pode ser usada tanto para problemas de regressão quanto para problemas de classificação.

A ideia fundamental é que a extração de características definida pela camada oculta gere novas características que melhorem a qualidade de regressão ou classificação. De fato criamos um novo espaço de características que muito provavelmente conseguiu recombinar as características existentes  $x_j$  de uma forma não linear para obter novas características  $\phi_h$  mais “poderosas”, obviamente além de características de pouco valor.

Vamos analisar o processo de treinamento do ponto de vista do conjunto de dados de treinamento. Igualmente como no caso da ML, no treinamento determinístico definido na seção 2.2.3 na pág. 54, temos  $n$  padrões de entrada  $\mathbf{x}_i$  de dimensão  $(m+1)$  e  $n$  padrões de saída  $\mathbf{y}_i$  de dimensão  $c$ . Ambos podem ser novamente armazenados em matrizes

$$X = \begin{bmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_i^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix} = \begin{bmatrix} x_{1,0} & \dots & x_{1,j} & \dots & x_{1,m} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i,0} & \dots & x_{i,j} & \dots & x_{i,m} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{n,0} & \dots & x_{n,j} & \dots & x_{n,m} \end{bmatrix}, \quad Y = \begin{bmatrix} \mathbf{y}_1^\top \\ \vdots \\ \mathbf{y}_i^\top \\ \vdots \\ \mathbf{y}_n^\top \end{bmatrix} = \begin{bmatrix} y_{1,1} & \dots & y_{1,k} & \dots & y_{1,c} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ y_{i,1} & \dots & y_{i,k} & \dots & y_{i,c} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ y_{n,1} & \dots & y_{n,k} & \dots & y_{n,c} \end{bmatrix}. \quad (4.43)$$

Na ELM porém, temos um padrão intermediário  $\phi(\mathbf{x}_i)$  para cada um dos  $n$  padrões de entrada. Além disso, esse mapeamento é feito uma única vez. Dessa maneira, podemos definir uma nova matriz  $\Phi$ , de dimensão  $n \times H$ , dos padrões intermediários

$$\Phi = \begin{bmatrix} \phi(\mathbf{x}_1)^\top \\ \vdots \\ \phi(\mathbf{x}_i)^\top \\ \vdots \\ \phi(\mathbf{x}_n)^\top \end{bmatrix} = \begin{bmatrix} \phi_{1,1} & \cdots & \phi_{1,h} & \cdots & \phi_{1,H} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \phi_{i,1} & \cdots & \phi_{i,h} & \cdots & \phi_{i,H} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \phi_{n,1} & \cdots & \phi_{n,h} & \cdots & \phi_{n,H} \end{bmatrix} = \begin{bmatrix} z(\tilde{\mathbf{w}}_1 \cdot \mathbf{x}_1) & \cdots & z(\tilde{\mathbf{w}}_h \cdot \mathbf{x}_1) & \cdots & z(\tilde{\mathbf{w}}_H \cdot \mathbf{x}_1) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ z(\tilde{\mathbf{w}}_1 \cdot \mathbf{x}_i) & \cdots & z(\tilde{\mathbf{w}}_h \cdot \mathbf{x}_i) & \cdots & z(\tilde{\mathbf{w}}_H \cdot \mathbf{x}_i) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ z(\tilde{\mathbf{w}}_1 \cdot \mathbf{x}_n) & \cdots & z(\tilde{\mathbf{w}}_h \cdot \mathbf{x}_n) & \cdots & z(\tilde{\mathbf{w}}_H \cdot \mathbf{x}_n) \end{bmatrix}. \quad (4.44)$$

A partir de agora, os pesos  $\mathbf{w}_{h,k}$  podem ser obtidos como na ML. Basta substituir a matriz de entrada  $X$  na aprendizagem de pesos pela matriz  $\Phi$ . Reveja a teoria na seção 2.2.3 na pág. 54. Temos então o mapeamento da entrada até a saída como

$$Y = \Phi W. \quad (4.45)$$

$$W = \Phi^\dagger Y, \quad (4.46)$$

onde

$$\Phi^\dagger = (\Phi^\top \Phi)^{-1} \Phi^\top$$

é a pseudoinversa da matriz de dados mapeados pela camada oculta. A matriz  $\Phi$  tem dimensão  $n \times H$ , a sua transposta  $\Phi^\top$  tem dimensão  $H \times n$ , o produto  $\Phi^\top \Phi$  e seu inverso  $(\Phi^\top \Phi)^{-1}$  têm dimensão  $H \times H$ , portanto, a pseudoinversa  $\Phi^\dagger$  têm dimensão  $H \times n$ , consequentemente  $W$  têm dimensão  $H \times c$ , pois  $Y$  tem dimensão  $n \times c$ .

Existe a possibilidade de estabilizar o cálculo da matriz de pesos da (eq. 4.42) na (eq. 4.46) por *regularização*. Um hiperparâmetro de regularização  $C > 0$  é usado na obtenção da pseudoinversa para evitar que a matriz  $\Phi^\top \Phi$  corra perigo de ficar singular, e assim impossibilitar a sua inversão. Isso pode acontecer, se o posto deste produto for menor que a quantidade  $n$  de padrões. Antes desta inversão, uma constante  $1/C$  é somada aos elementos diagonais, definindo assim a pseudoinversa regularizada

$$\Phi_C^\dagger \stackrel{\text{def}}{=} (\Phi^\top \Phi + \frac{I}{C})^{-1} \Phi^\top, \quad (4.47)$$

onde  $I$  é a matriz de identidade de dimensão  $H \times H$ . Vale a pena ainda comentar que este mecanismo aplicado ao problema mais simples da Máquina Linear, na inversão da matriz  $X^\top X$ , da matriz de dados  $X$  na (eq. 2.62) na pág. 52 é chamado regularização de Tikhonov (com  $C$  igual em todas as dimensões), *ridge regression* [43, 51, 58, 88]. A versão regularizada desta matriz é então

$$X_C^\dagger \stackrel{\text{def}}{=} (X^\top X + \frac{I}{C})^{-1} X^\top. \quad (4.48)$$

O efeito no caso de classificação é que as fronteiras que separam as regiões de decisão ficam mais suaves.

■ **Exemplo 4.5** Claro, vamos classificar as nossas flores pela ELM. Já definimos a entrada e saída certa na figura 4.5, com  $(m+1) = 5$  e  $c = 3$ . A quantidade  $H = 7$  de neurônios foi uma escolha arbitrária para manter a complexidade da figura dentro de limites razoáveis. Na prática, com a

ambição de gerar características discriminativas novas na camada oculta, vamos definir como heurística a multiplicação por dez, da quantidade de características originais de entrada, portanto com as flores temos  $H = 40$ . Como os pesos da entrada até a camada oculta são arbitrárias, espera-se uma forte dependência da maneira de definir as regiões de decisão em um problema de classificação. A semente do gerador de números aleatórios e a quantidade de neurônios são variadas na figura 4.6 na classificação de iris para as últimas duas características. A divisão do espaço de características é muito diferente quando se varia os pesos aleatórios  $\tilde{w}_{h,j}$ , porém a separação de classes permanece estável. Com um número reduzido de neurônios ocultos,  $H = 2$ , a separação de três classes não é viável. Com o aumento de  $H$ , a flexibilidade das fronteiras aumenta. Retornaremos ao assunto da flexibilidade, no contexto da *regularização*. ■

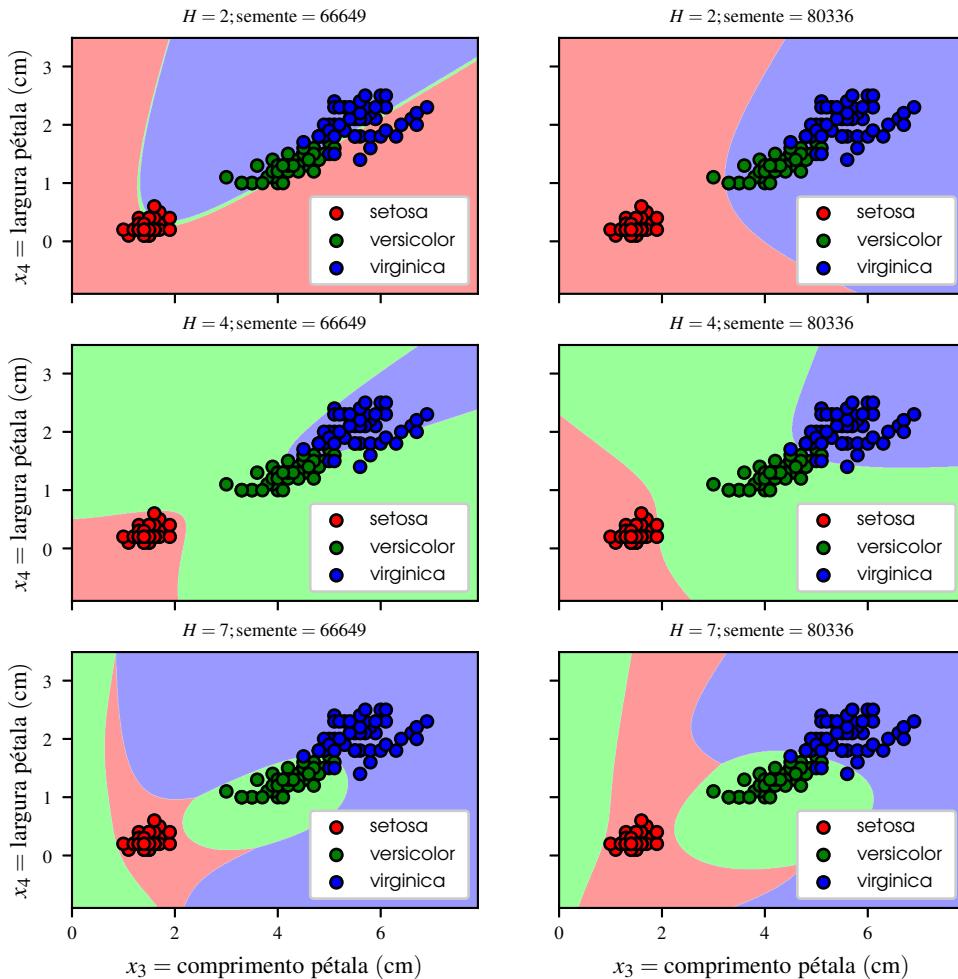


Figura 4.6: Regiões de decisão definidas pela ELM, com número de neurônios ocultos  $H$  diferentes e pesos  $\tilde{w}_{h,j}$  da entrada a camada oculta diferentes, devido à semente diferente do gerador de números aleatórios.

■ **Exemplo 4.6** A figura 4.7 mostra o efeito do hiperparâmetro de regularização para os valores

$\{0, 0.1, 1\}$  para número de neurônios ocultos  $\{8, 20, 60\}$ , fixando a semente do gerador de números aleatórios fixo 66649. A função de ativação usada é a ReLU, veja a tabela 5.1 na pág. 172. ■

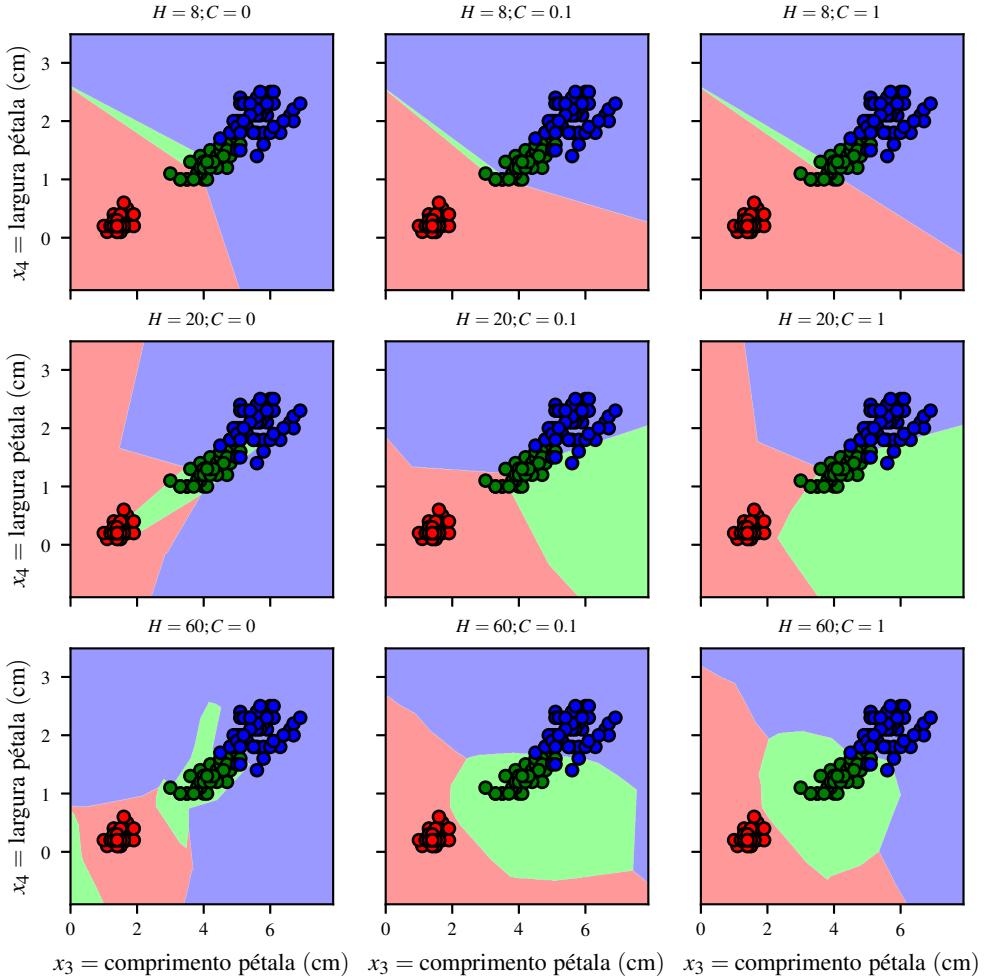


Figura 4.7: Regiões de decisão definidas pela ELM, com número de neurônios ocultos  $H$  diferentes e hiperparâmetro de regularização  $C$  diferente.

Em exemplos práticos, a ELM consta entre os classificadores com um poder grande de separabilidade de classes [55]. Neste ponto seja permitida uma especulação, porque a ELM funciona bem. O trabalho pioneiro de Cover na área de reconhecimento de padrões [19] analisa a capacidade de separar classes dos classificadores lineares. Uma *dicotomia* é a divisão em duas partes. Neste contexto significa que um conjunto de padrões é separado em duas partes, seja qual for a classe de cada um dos padrões. A quantidade de dicotomias, ou seja a atribuição arbitrária de  $n$  padrões de dimensão  $d$  em um problema de classificação binária é  $2^n$ . Por exemplo com dois padrões, eles podem ficar ambos do lado positivo do hiperplano, ambos do lado negativo do hiperplano, o primeiro do lado positivo e o segundo do lado negativo, ou vice versa, assim somando quatro casos diferentes. Um resultado importante é que a probabilidade da dimensão  $d^*$  a partir de qual um conjunto arbitrário começa a ficar linearmente separável, segue uma distribuição binomial

$$\text{Prob}[d^* = d] = \left(\frac{1}{2}\right)^{n-1} \binom{n-1}{d-1}, \quad d = 1, 2, \dots, n, \quad (4.49)$$

com valor esperado

$$\mathbb{E}[d^*] = \frac{n+1}{2}. \quad (4.50)$$

Isto significa que para um número fixo  $n$  de padrões de treinamento, se for possível aumentar a dimensão  $H$  do espaço de características até aproximadamente a metade do número de padrões, a chance aumenta de achar uma separação binária limpa entre padrões de classes diferentes. Certamente para um problema concreto de classificação, os padrões *não* são distribuídos arbitrariamente. Não obstante, a geração de  $H$  novas características arbitrárias na camada oculta pela ELM, sendo  $H$  um valor alto comparado como a dimensão original  $m$ , sugere que os padrões novos mapeados  $\phi(\mathbf{x}) \in \mathbb{R}^H$  sejam muito mais facilmente separáveis do que os padrões originais  $\mathbf{x} \in \mathbb{R}^m$  na dimensão baixa  $m$ . Além disso a geração arbitrária eventualmente favorece as condições teóricas de Cover no espaço oculto de dimensão  $H$ . Mais uma vez, estou especulando, mas plausivelmente.

### 4.3.2 Modelos Lineares Generalizados

A arquitetura da Máquina Extrema de Aprendizado é um representante de uma classe de modelos que fazem primeiro uma extração não linear de características, seguida por uma combinação linear, conforme a figura 4.5. Diversos métodos de regressão e classificação pertencem a essa categoria. O Perceptron Multicamadas (MLP) com uma camada oculta é idêntico à ELM na arquitetura, porém muito diferente no método de ajuste de pesos. O MLP treina os pesos da entrada a camada oculta pela retropropagação de erros, baseada na descida de gradiente. Como não é plausível treinar os pesos da (última) camada oculta até a saída por uma algoritmo determinístico, essa parte da rede MLP também é treinada pela descida de gradiente. A Máquina de Vetor de Suporte (SVM) também é um extrator não linear de características, seguido por uma combinação linear. Já foi mencionado na seção 2.4.5 na pág. 75 que os pesos usados na combinação linear são fruto de uma combinação não linear (com o uso de um *kernel* não linear) de alguns padrões  $\mathbf{x}_i$  do conjunto de treinamento. Outro exemplo são Redes de Base Radial (*Radial Basis Function Networks*, RBF) [73], que semelhantemente aos GMM definem alguns centros  $\mu$ . A combinação linear da distância de um padrão  $\mathbf{x}$  desses centros é a saída da rede.

Um modelo interessante é a regressão polinomial. A ideia é simplesmente criar novas características não lineares, a partir das  $m$  características originais

$$\{x_1, \dots, x_j, \dots, x_m\}$$

em forma de *monômios*

$$x_1^{d_1} \cdot x_2^{d_2} \cdots x_j^{d_j} \cdots x_m^{d_m}, \quad (4.51)$$

onde

$$0 \leq d_j \leq g_{\max}$$

é o grau da  $j$ -ésima característica, e  $g_{\max}$  é o maior grau permitido. Depois combinamos linearmente esses monômios. Equivalentemente, estamos trabalhando com um modelo linear na *base monomial*, ou seja, fizemos um mapeamento da base original no sistema de coordenadas cartesiano para uma nova base, da mesma maneira como fizemos na ELM. Considere, como exemplo, o modelo de característica original com três características distintas  $\{x_1, x_2, x_3\}$ . Usando o grau máximo dos monômios de um, ou seja

$$g_{\max} = 1,$$

temos as  $H = 8$  novas características  $\phi_h, h = 1, \dots, 8$

$$\begin{aligned}\phi_1 &= x_1^0 x_2^0 x_3^0 = 1 \cdot 1 \cdot 1 = 1, \\ \phi_2 &= x_1^0 x_2^0 x_3^1 = 1 \cdot 1 \cdot x_3 = x_3, \\ \phi_3 &= x_1^0 x_2^1 x_3^0 = 1 \cdot x_2 \cdot 1 = x_2, \\ \phi_4 &= x_1^1 x_2^0 x_3^0 = x_1 \cdot 1 \cdot 1 = x_1, \\ \phi_5 &= x_1^0 x_2^1 x_3^1 = 1 \cdot x_2 \cdot x_3 = x_2 x_3, \\ \phi_6 &= x_1^1 x_2^0 x_3^1 = x_1 \cdot 1 \cdot x_3 = x_1 x_3, \\ \phi_7 &= x_1^1 x_2^1 x_3^0 = x_1 \cdot x_2 \cdot 1 = x_1 x_2, \\ \phi_8 &= x_1^1 x_2^1 x_3^1 = x_1 x_2 x_3.\end{aligned}$$

Note que alguns das características são idênticas às existentes.

A combinação linear generalizada finalmente seria

$$y(\mathbf{x}) = \mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}) = w_1 \phi_1 + \cdots + w_h \phi_h + \cdots + w_H \phi_H \quad (4.52)$$

Com duas característica originais  $\{x_1, x_2\}$  e grau máximo  $g_{\max} = 2$  temos  $H = 8$  monômios

$$\begin{aligned}\phi_1 &= x_1^0 x_2^0 = 1 \cdot 1 = 1, \\ \phi_2 &= x_1^0 x_2^1 = 1 \cdot x_2 = x_2, \\ \phi_3 &= x_1^1 x_2^0 = x_1 \cdot 1 = x_1, \\ \phi_4 &= x_1^1 x_2^1 = x_1 x_2, \\ \phi_5 &= x_1^0 x_2^2 = 1 \cdot x_2^2 = x_2^2, \\ \phi_6 &= x_1^1 x_2^2 = x_1 x_2^2, \\ \phi_7 &= x_1^2 x_2^1 = x_1^2 x_2, \\ \phi_8 &= x_1^2 x_2^2,\end{aligned}$$

com a combinação linear generalizada

$$y(\mathbf{x}) = \mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}) = w_1 + w_2 x_2 + w_3 x_1 + w_4 x_1 x_2 + w_5 x_2^2 + w_6 x_1 x_2^2 + w_7 x_1^2 x_2 + w_8 x_1^2 x_2^2.$$

Obviamente, tendo mais que uma variável de saída  $y$ , podemos definir um vetor de pesos  $\mathbf{w}_k$  para cada uma dessas saídas. Esta arquitetura tem o sinônimo de *rede Pi–Sigma* no contexto das redes neurais artificiais [100].



## 5. Redes Neurais Multicamadas Adiante (*Feedforward*)

### 5.1 O Perceptron Multicamada e o Algoritmo de Retropropagação de Erro

Aplicações contemporâneas baseados em redes neurais artificiais são dominadas por um modelo que implementa um mapeamento funcional e ajusta os seus parâmetros livres por descida de gradiente. Já estudamos o modelo linear, aplicando regressão ou classificação. No modelo linear aplicamos a descida de gradiente para aprender os pesos como alternativa do método determinístico pela pseudoinversa dos dados. No contexto da extração de características, conhecemos já um modelo de redes neurais com uma camada oculta, a Máquina Extrema de Aprendizagem, ELM, que implementa um modelo não linear, usando uma função de ativação não linear. Os pesos da ELM da entrada até a camada oculta não são aprendidos, são simplesmente lhes atribuídos valores aleatórios. Os pesos da camada oculta para a camada de saída podem ser calculados deterministicamente, usando a pseudoinversa dos dados na camada oculta. Teoricamente poderíamos treinar esse conjunto de pesos iterativamente por descida de gradiente, porém isso não está sendo aplicado na prática no modelo da ELM.

#### 5.1.1 Modelo do Perceptron Multicamada

O modelo que vamos estudar, o Perceptron Multicamada, *Multilayer Perceptron*, MLP, na sua forma básica, tem uma arquitetura idêntica à ELM. A grande diferença é que todos os pesos no modelo são ajustados por descida de gradiente. O ajuste dos pesos da camada oculta para camada de saída é relativamente simples, e já estudamos como funciona. O ajuste dos pesos da entrada até a camada oculta foi somente proposto nos anos 1970 e 1980. Mais nada é do que a aplicação consequente da regra da cadeia para calcular a derivada da função de perda em relação a um peso que não está na última camada. A técnica de ajuste dos pesos anterior à camada oculta é a *Retropropagação de erro (error backpropagation)*. Os fundamentos matemáticos foram elaborados por Werbos na sua tese de doutorado [114], porém o potencial desse cálculo ficou mais que dez anos sem atenção, até divulgado para um público mais amplo por Rumelhart, Hinton e Williams na revista popular Nature [94], entre outras. Começou uma onda de atividades na mais diversas áreas de aplicação. Surgiram aplicações que atraíram muita atenção, por exemplo, em processamento de imagens, ou de sinais em geral. Os modelos de *aprendizagem profunda*,

*deep learning*, que no momento da elaboração deste livro dominam as atividades de pesquisa e aplicações reais, no fundo baseiam-se no mesmo princípio da retropropagação de erro, porém têm em geral mais camadas com cálculos de processamento de sinais embutidos, por exemplo, a camada convolucional. Estudaremos as redes profundas posteriormente.

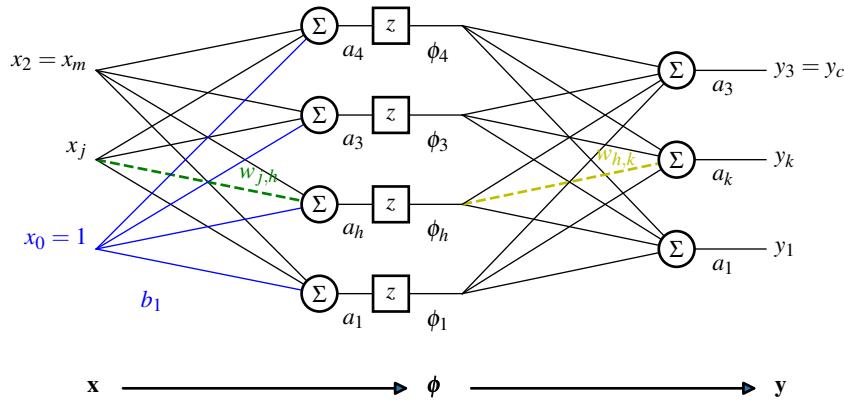


Figura 5.1: Rede neural com uma camada oculta não linear e a camada de saída linear. O exemplo tem  $(m+1) = (2+1) = 3$  entradas  $x_0, \dots, x_m$  e  $c = 3$  saídas  $y_1, \dots, y_c$ . Na camada oculta,  $H = 4$  neurônios foram introduzidos que primeiro calculam a combinação linear das entradas e depois passam o produto escalar por uma função de ativação não linear. Destacados na figura são dois pesos que representam todos os pesos da entrada para a camada oculta, e da camada oculta para a camada de saída, respectivamente.

A figura 5.1 mostra uma configuração básica do Perceptron Multicamada com uma camada oculta. A rede tem uma quantidade concreta de duas mais uma entradas, quatro neurônios ocultos, e três saídas, porém essas quantidades  $(2, 4, 3)$  serão generalizadas pelas variáveis denominadas como  $m$ ,  $H$  e  $c$ , e os seus índices enumeradores  $j = 0, \dots, m$ ,  $h = 1, \dots, H$ , e  $k = 1, \dots, c$ . A não linearidade da rede é introduzida pela função de ativação  $z$  na camada oculta. O uso de uma função de ativação na camada de saída não é necessária para estudar a teoria. Não foi usada no modelo da figura 5.1. Poderia ser colocada também como último passo de processamento, mas não muda a capacidade de cálculo do modelo. A arquitetura aqui mostrada é um *aproximador universal de funções*, ou seja, teoricamente seria capaz de modelar qualquer mapeamento funcional, seja qual for a sua complexidade. A forma da função  $\mathbf{f}$ , entrada multidimensional para saída funcional já foi discutida no modelo principal de aprendizado de máquina, veja a definição 2.1.1 na pág. 32. A definição é repetida aqui, adaptada ao modelo do MLP. Temos a função

$$\mathbf{y} = \mathbf{f}(\mathbf{x}; \boldsymbol{\theta}), \quad (5.1)$$

onde  $\mathbf{x} \in \mathbb{R}^{m+1}$  é a entrada de dimensão  $(m+1)$ ,  $\mathbf{y} \in \mathbb{R}^c$  é a saída de dimensão  $c$ , e  $\boldsymbol{\theta} \in \mathbb{R}^D$  é um conjunto que unifica todos os  $D$  parâmetros livres do modelo. No modelo MLP básico, podemos considerar todos os pesos espalhados pela rede como os parâmetros, ou seja

$$D = (m+1) \cdot H + H \cdot c. \quad (5.2)$$

Essa é a quantidade de todos os pesos da entrada até a camada oculta, junto com todos os pesos da camada oculta até a camada de saída. No exemplo concreto da figura 5.1, temos

$$D = (2+1) \cdot 4 + 4 \cdot 3 = 12 + 12 = 24$$

pesos no total. O *Teorema da aproximação universal* [20, 44, 45, 108, 111] consta que o modelo apresentado na figura 5.1 é capaz de aproximar funções contínuas em subconjuntos compactos de  $\mathbb{R}^m$ . Informalmente pode-se afirmar que, desde que a quantidade de neurônios ocultos seja suficientemente grande e os pesos sejam ajustados apropriadamente, o MLP é universalmente utilizável para tarefas de regressão e classificação, aproximando bem a função desejada.

**Dica:** A única especificação da quantidade de camadas de uma rede neural que não deixa ambiguidades, é a contagem das camadas ocultas. Alguns autores contam a entrada da rede como camada. Isso não se justifica, pois nenhum processamento é feito na entrada.

Na definição 4.3.2 na pág. 140 definiu-se pela primeira vez o modelo do MLP, no contexto da ELM. Os pesos  $\tilde{w}_{j,h}$  da entrada até a camada oculta foram qualificados com um tilde “~” para distingui-los dos pesos  $w_{h,k}$  da camada oculta até a camada de saída, veja também a figura 4.5 na pág. 141. Aqui não vamos fazer essa distinção para não sobrecarregar a nomenclatura. O peso é identificado pelas letras dos seus índices,  $w_{j,h}$  significa entrada para camada oculta,  $w_{h,k}$  significa camada oculta para camada de saída. Em caso de valores concretos dos índices, por exemplo  $w_{2,1}$  ou  $w_{3,5}$  poderiam surgir dúvidas a qual camada esse peso pertence. Se for necessário, a respectiva camada é mencionada explicitamente.

Para acompanhar a teoria, as flores servem novamente como orientação. Desta vez, na figura 5.1, temos somente duas características na entrada, ou seja, o vetor de entrada é  $\mathbf{x} = [x_1 \ x_2]^T$  na forma não aumentada, onde o viés  $b$  é tratado separadamente, ou  $\mathbf{x} = [1 \ x_1 \ x_2]^T$  na forma aumentada, onde o viés é considerado como 0-ésimo peso  $b = w_0$ , e a 0-ésima entrada tem o valor fixo  $x_0 = 1$ . Temos então

$$m + 1 = 2 + 1 = 3,$$

entradas. Usamos novamente o índice  $j$  para enumerar essas entradas, ou seja, em geral temos os índices de entrada

$$j = 0, 1, \dots, m$$

para a representação aumentada. Na camada oculta temos

$$H = 4$$

neurônios. Usamos o índice  $h$  para enumerar essas entradas, ou seja, em geral temos os índices

$$h = 1, \dots, H$$

para identificar os neurônios ocultos. É possível introduzir um viés também na camada oculta. Nesse caso a contagem de  $h$  começaria em zero. Não se aplica neste exemplo.

Vamos introduzir<sup>1</sup> uma notação útil para a combinação linear de entradas com pesos, a *ativação*, usando o símbolo  $a$ . Ocorre na rede da figura 5.1 para cada neurônio da camada oculta, ou seja, podemos definir as  $H$  ativações

$$a_h(\mathbf{x}; \mathbf{w}_h) \stackrel{\text{def}}{=} \mathbf{w}_h \cdot \mathbf{x} = \sum_{j=0}^m w_{j,h} x_j, \quad h = 1, \dots, H. \quad (5.3)$$

<sup>1</sup>A definição da *ativação* é idêntica à definição da distância não normalizada de um ponto  $x$  de um hiperplano definido pelo vetor normal  $\mathbf{w}$  e viés  $w_0$  como, usando vetores aumentados,  $d(\tilde{\mathbf{x}}; \tilde{\mathbf{w}}) = \tilde{\mathbf{w}} \cdot \tilde{\mathbf{x}} = 0$  da (eq. 2.84) na pág. 67.

Também ocorre na rede para cada neurônio da camada de saída, ou seja, podemos definir as ativações

$$a_k(\boldsymbol{\phi}; \mathbf{w}_k) \stackrel{\text{def}}{=} \mathbf{w}_k \cdot \boldsymbol{\phi} = \sum_{h=1}^H w_{h,k} \phi_h, \quad k = 1, \dots, c. \quad (5.4)$$

Após a combinação linear, normalmente esta ativação é processada pela função de ativação

$$z : \mathbb{R} \rightarrow \mathbb{R}.$$

A função de ativação tem um argumento unidimensional de entrada e a sua saída também é unidimensional. Na camada oculta, a função de ativação tem que ser obrigatoriamente não linear, caso contrário não sairíamos do mundo linear, veja posteriormente a consideração na seção 5.1.6. No modelo da figura 5.1 não tem uma função de ativação na camada de saída. Então na camada oculta, para cada um dos neurônios temos o cálculo

$$\phi_h(\mathbf{x}; \mathbf{w}_h) = z(a_h) = z(\mathbf{w}_h \cdot \mathbf{x}) = z\left(\sum_{j=0}^m w_{j,h} x_j\right), \quad h = 1, \dots, H. \quad (5.5)$$

No nosso exemplo, na camada de saída, não utilizamos uma função de ativação. Podemos também dizer que a função de ativação calcula a identidade  $z(a) = a$  da sua entrada, ou seja, na saída temos

$$y_k = z(a_k) = a_k. \quad (5.6)$$

O vetor

$$\mathbf{w}_h = [w_{0,h} \quad w_{1,h} \quad \cdots \quad w_{j,h} \quad \cdots \quad w_{m,h}]^\top \quad (5.7)$$

representa todos os pesos que entram no  $h$ -ésimo neurônio  $\phi_h$  da camada oculta. Dessa maneira podemos agrupar todos os  $H$  vetores de pesos que conectam a entrada com a camada oculta em uma matriz

$$W_{\text{entrada} \rightarrow \text{oculta}} = [\mathbf{w}_1 \quad \cdots \quad \mathbf{w}_h \quad \cdots \quad \mathbf{w}_H] = \begin{bmatrix} w_{0,1} & \cdots & w_{0,h} & \cdots & w_{0,H} \\ w_{1,1} & \cdots & w_{1,h} & \cdots & w_{1,H} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ w_{j,1} & \cdots & w_{j,h} & \cdots & w_{j,H} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ w_{m,1} & \cdots & w_{m,h} & \cdots & w_{m,H} \end{bmatrix}. \quad (5.8)$$

Para compactar a terminologia, vamos denominar esta matriz com o símbolo  $\boldsymbol{\phi}$  das características extraídas na camada oculta, ou seja

$$W_\phi \stackrel{\text{def}}{=} W_{\text{entrada} \rightarrow \text{oculta}}.$$

Então, considerando todos os pesos da entrada até a camada oculta, o cálculo feito pela camada oculta é

$$\begin{aligned} \boldsymbol{\phi}(\mathbf{x}; W_\phi) &= [\phi_1(\mathbf{x}; \mathbf{w}_1) \quad \cdots \quad \phi_h(\mathbf{x}; \mathbf{w}_h) \quad \cdots \quad \phi_H(\mathbf{x}; \mathbf{w}_H)]^\top \\ &\stackrel{5.3}{=} [z(a_1) \quad \cdots \quad z(a_h) \quad \cdots \quad z(a_H)]^\top \\ &\stackrel{5.5}{=} [z(\mathbf{w}_1 \cdot \mathbf{x}) \quad \cdots \quad z(\mathbf{w}_h \cdot \mathbf{x}) \quad \cdots \quad z(\mathbf{w}_H \cdot \mathbf{x})]^\top \\ &\stackrel{5.5}{=} \left[ z\left(\sum_{j=0}^m w_{j,1} x_j\right) \quad \cdots \quad z\left(\sum_{j=0}^m w_{j,h} x_j\right) \quad \cdots \quad z\left(\sum_{j=0}^m w_{j,H} x_j\right) \right]^\top. \end{aligned} \quad (5.9)$$

Recorde da definição 4.3.2 na pág. 140 que este cálculo  $\phi_h$  de cada um dos neurônios ocultos pode ser considerado um extração de uma nova característica, em geral não linear. Assim a camada oculta representa um novo vetor de características  $\boldsymbol{\phi}$  de dimensão  $H$  como

$$\boldsymbol{\phi} = [\phi_1 \ \phi_2 \ \cdots \ \phi_h \ \cdots \ \phi_H]^\top. \quad (5.10)$$

Para entender o princípio da retropropagação de erro, basta estudar como um peso representativo da entrada para a camada oculta é ajustado, pois o ajuste de cada peso da rede pode ser considerado separadamente. O peso responsável para a conexão entre a  $j$ -ésima entrada  $x_j$ , e o  $h$ -ésimo neurônio na camada oculta é destacado como  $w_{j,h}$  na figura 5.1. No caso concreto da figura, temos  $j = 1$  e  $h = 2$ , ligando  $x_1$  com  $\phi_2$ . O entendimento da retropropagação de erro se resume como esse peso é ajustado, baseado no gradiente da função de perda. Vamos estudar isso em breve, passo a passo. Finalmente temos

$$c = 3$$

saídas, usando o índice enumerador  $k$ . Em geral temos para os índices da saída

$$k = 1, \dots, c.$$

Então, na saída da rede, na  $k$ -ésima saída  $y_k$ , temos novamente uma combinação linear, desta vez das novas características da (eq. 5.10) com o  $k$ -ésimo vetor de pesos

$$\mathbf{w}_k = [w_{1,k} \ w_{2,k} \ \cdots \ w_{h,k} \ \cdots \ w_{H,k}]^\top \quad (5.11)$$

como

$$y_k(\boldsymbol{\phi}; \mathbf{w}_k) = \mathbf{w}_k \cdot \boldsymbol{\phi} = \sum_{h=1}^H w_{h,k} \phi_h. \quad (5.12)$$

O vetor  $\mathbf{w}_k$  representa todos os pesos que entram o  $k$ -ésimo neurônio  $y_k$  da camada de saída. Dessa maneira podemos novamente agrupar todos os  $c$  vetores de pesos que conectam a camada oculta com a camada de saída em uma matriz

$$W_{\text{oculta} \rightarrow \text{saída}} = [\mathbf{w}_1 \ \cdots \ \mathbf{w}_k \ \cdots \ \mathbf{w}_c] = \begin{bmatrix} w_{1,1} & \cdots & w_{1,k} & \cdots & w_{1,c} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ w_{h,1} & \cdots & w_{h,k} & \cdots & w_{h,c} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ w_{H,1} & \cdots & w_{H,k} & \cdots & w_{H,c} \end{bmatrix}, \quad (5.13)$$

abreviada, usando o símbolo  $y$  dos valores de saída

$$W_y \stackrel{\text{def}}{=} W_{\text{oculta} \rightarrow \text{saída}}.$$

Então, considerando todos os pesos da camada oculta até a camada de saída, o cálculo feito pela camada de saída é o mapeamento funcional que retorna um vetor  $\mathbf{y}$  de dimensão  $c$

$$\begin{aligned} \mathbf{f}(\mathbf{x}; W_\phi, W_y) &= [y_1(\mathbf{x}; W_\phi, \mathbf{w}_1) \ \cdots \ y_k(\mathbf{x}; W_\phi, \mathbf{w}_k) \ \cdots \ y_c(\mathbf{x}; W_\phi, \mathbf{w}_c)]^\top \\ &= W_y^\top \boldsymbol{\phi}(\mathbf{x}; W_\phi) \end{aligned} \quad (5.14)$$

$$\begin{aligned} &\stackrel{5.9}{=} W_y^\top [\phi_1(\mathbf{x}; \mathbf{w}_1) \ \cdots \ \phi_h(\mathbf{x}; \mathbf{w}_h) \ \cdots \ \phi_H(\mathbf{x}; \mathbf{w}_H)] \\ &\stackrel{5.9}{=} W_y^\top [z(\mathbf{w}_1 \cdot \mathbf{x}) \ \cdots \ z(\mathbf{w}_h \cdot \mathbf{x}) \ \cdots \ z(\mathbf{w}_H \cdot \mathbf{x})]^\top \end{aligned} \quad (5.14)$$

$$\stackrel{5.3}{=} W_y^\top [z(a_1) \ \cdots \ z(a_h) \ \cdots \ z(a_H)]^\top \quad (5.15)$$

Repare que o vetor de saída  $\mathbf{y}$  é resultado da multiplicação de uma matriz (transposta)  $W_y$  com um vetor  $\phi$ , portanto, a parte da camada oculta até a saída é um mapeamento linear. Se tivesse uma função de ativação na saída da rede, não seria possível definir esta multiplicação da matriz com o vetor que representa as características na camada oculta. Vamos focar na  $k$ -ésima saída  $y_k$ . O peso destacado na figura 5.1 que nos interessa é  $w_{h,k}$  entre o  $h$ -ésimo neurônio oculto  $\phi_h$ , e a  $k$ -ésima saída  $y_k$ . Expandindo a (eq. 5.12), temos o cálculo final do MLP como

$$\begin{aligned} y_k(\phi, \mathbf{w}_k) &= y_k(\mathbf{x}; W_\phi, \mathbf{w}_k) = \sum_{h=1}^H w_{h,k} \phi_h(\mathbf{x}; \mathbf{w}_h) \\ &\stackrel{5.5}{=} \sum_{h=1}^H w_{h,k} z(a_h) \end{aligned} \quad (5.16)$$

$$\stackrel{5.5}{=} \sum_{h=1}^H w_{h,k} \left\{ z \left( \sum_{j=0}^m w_{j,h} x_j \right) \right\}, \quad k = 1, \dots, c. \quad (5.17)$$

Neste cálculo, lembra-se mais uma vez que o vetor  $\mathbf{w}_h$  representa pesos da camada oculta que conectam com o  $h$ -ésimo neurônio da camada oculta, e que o vetor  $\mathbf{w}_k$  representa pesos da camada de saída que conectam com o  $k$ -ésimo neurônio da camada de saída. Na (eq. 5.17) reconhecemos os dois pesos destacados  $w_{j,h}$  e  $w_{h,k}$ . O peso  $w_{j,h}$  fica mais “enterrado” dentro da equação. Lembrando da cadeia de funções da definição 2.1.8 na pág. 42, podemos constatar que este peso  $w_{j,h}$  é usado logo no início da cadeia. Isso terá consequências quando aplicarmos a retropropagação de erro na aprendizagem deste peso. Temos que “cavar” mais para dentro da expressão no momento da aplicação da regra da cadeia, para chegar no peso. Uma observação extremamente importante igualmente relevante para esta questão, é que a  $k$ -ésima saída  $y_k$ , definida na (eq. 5.17) é influenciada por *todos* os pesos  $w_{j,h}$  da entrada a camada oculta, contidos na matriz (eq. 5.8). Por outro lado, a  $k$ -ésima saída  $y_k$  é influenciada somente pelos pesos  $w_{h,k}$  da camada oculta a camada de saída. Isto torna mais fácil a aprendizagem destes pesos  $\mathbf{w}_k$ , correspondendo à  $k$ -ésima coluna da matriz  $W_y$ . Pormenores vamos estudar no algoritmo de retropropagação de erro. No momento temos a definição do Perceptron Multicamada. Não falamos ainda nada sobre o treinamento dos seus pesos. O exemplo a seguir com valores numéricos concretos, traça o caminho de um padrão desde a entrada da rede, até a saída, aplicando o mapeamento geral  $\mathbf{y} = \mathbf{f}(\mathbf{x}; \boldsymbol{\theta})$  do aprendizado de máquina.

■ **Exemplo 5.1** Vamos usar a arquitetura da figura 5.1 para classificar três classes de iris, usando as primeiras duas características  $x_1$  e  $x_2$ . Para entender o processo de ajuste dos pesos, é suficiente analisar um único passo. No treinamento estocástico, apresentamos um único padrão  $\mathbf{x}_i$ , depois calculamos a perda individual  $L_i$ , e finalmente, baseado no gradiente dessa perda ajustamos cada peso da rede. Consideremos a primeira flor, ou seja, a primeira setosa, ignorando a terceira e quarta característica. Então temos

$$\mathbf{x}_i = [5.1 \quad 3.5]^\top,$$

sendo  $i = 1$ . Lembre que no algoritmo de aprendizagem estocástica, os padrões do conjunto de treinamento têm que ser apresentados em ordem aleatória, porém neste momento, essa necessidade não é de relevância para compreender o mecanismo que ajusta os pesos. Os pesos da rede têm que ser inicializados. Sejam as duas matrizes de pesos inicializados aleatoriamente, por exemplo pelo

método descrito em [41], como

$$W_\phi = \begin{bmatrix} b_0 & b_1 & b_2 & b_3 \\ w_{1,1} & \underline{w_{1,2}} & w_{1,3} & w_{1,4} \\ w_{2,1} & \underline{w_{2,2}} & w_{2,3} & w_{2,4} \end{bmatrix} = \begin{bmatrix} 0.060 & -0.462 & -0.392 & -0.322 \\ 0.290 & \underline{0.059} & -0.337 & -0.188 \\ -0.217 & -0.316 & 0.153 & -0.057 \end{bmatrix}$$

$$W_y = \begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} \\ w_{2,1} & w_{2,2} & w_{2,3} \\ w_{3,1} & \underline{w_{3,2}} & w_{3,3} \\ w_{4,1} & w_{4,2} & w_{4,3} \end{bmatrix} = \begin{bmatrix} -0.154 & 0.524 & 0.286 \\ 0.339 & 0.270 & -0.339 \\ 0.482 & \underline{-0.186} & 0.226 \\ -0.407 & -0.004 & 0.508 \end{bmatrix}$$

Na matriz entrada para camada oculta  $W_\phi$ , o peso destacado  $w_{j,h} = w_{1,2}$  é sublinhado. Na matriz camada oculta para camada de saída  $W_y$ , o peso destacado  $w_{h,k} = w_{3,2}$  é sublinhado. Em breve vamos focar no ajuste destes dois pesos representativos. Agora podemos processar a primeira setosa com índice  $i = 1$ , propagando adiante pela rede. A entrada é o vetor aumentado de dimensão  $(m+1) = 2+1=3$

$$\mathbf{x}_i = [x_{i,0} \ x_{i,1} \ x_{i,2}]^\top = [1 \ 5.1 \ 3.5]^\top,$$

Na camada oculta, antes da passagem pela função de ativação, temos no  $h$ -ésimo neurônio, o produto escalar da entrada como o  $h$ -ésimo vetor de peso da matriz (eq. 5.8)

$$\mathbf{w}_h \cdot \mathbf{x}_i, \quad h = 1, 2, 3, 4,$$

ou seja, os quatro valores

$$[\mathbf{w}_1 \cdot \mathbf{x}_i \ \mathbf{w}_2 \cdot \mathbf{x}_i \ \mathbf{w}_3 \cdot \mathbf{x}_i \ \mathbf{w}_4 \cdot \mathbf{x}_i]^\top = [0.719 \ -0.805 \ -1.186 \ -1.160]^\top.$$

Cada uma das  $H = 4$  componentes deste vetor passa individualmente pela função de ativação  $z$ , formando o vetor  $\phi$  das características extraídas da (eq. 5.10)

$$\begin{aligned} \phi &= [\phi_1 \ \phi_2 \ \phi_3 \ \phi_4]^\top \\ &= [z(\mathbf{w}_1 \cdot \mathbf{x}_i) \ z(\mathbf{w}_2 \cdot \mathbf{x}_i) \ z(\mathbf{w}_3 \cdot \mathbf{x}_i) \ z(\mathbf{w}_4 \cdot \mathbf{x}_i)]^\top \\ &= [z(a_1) \ z(a_2) \ z(a_3) \ z(a_4)]^\top \\ &= [0.672 \ 0.309 \ 0.234 \ 0.239]^\top. \end{aligned}$$

A função de ativação usado neste experimento é a sigmóide logística  $z(a) = 1/(1 + e^{-a})$ . Finalmente, como não temos uma função de ativação na camada de saída, o vetor de saída  $\mathbf{y}$  da (eq. 5.47) é uma combinação linear do vetor de características extraídas  $\phi$ , dada como

$$\mathbf{y} = W_y^\top \phi = [0.017 \ 0.391 \ 0.262].$$

Este vetor é a resposta  $\hat{\mathbf{y}}_i$  do modelo para a entrada  $\mathbf{x}_i$ . O valor desejado da primeira flor,  $i = 1$ , é

$$\mathbf{y}_i = [1 \ 0 \ 0]^\top.$$

Ainda vale a pena comentar que a saída contém três valores inteiros, que usa a codificação *One-Hot* da definição 2.1.4 na pág. 33. Embora seja uma saída apropriada para um problema de classificação, o modelo a trata igualmente a uma saída contínua, como no caso geral da regressão. ■

### 5.1.2 Função de Perda do Erro Quadrático

Antes de avançar para o método do ajuste dos pesos, vamos rever como a discrepância entre os valores esperados e explicados pelo modelo são definidos. No contexto da regressão linear já conhecemos o Erro Quadrático como medida apropriada. Para entender a teoria da aprendizagem dos pesos da rede, podemos nos limitar ao caso da aprendizagem estocástica que usa um único padrão para calcular a função de perda e imediatamente ajustar todos os pesos da MLP. Para completar, também vamos definir o Erro Quadrático para mais que um padrão. Para recordar das modalidades treinamento estocástico, em lote e mini-lote, reveja a definição 3.2.2 na pág. 88.

Na aprendizagem supervisionada, dado o conjunto de treinamento de  $n$  padrões, pares de entrada e saída,  $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_i, \mathbf{y}_i), \dots, (\mathbf{x}_n, \mathbf{y}_n)\}$  temos para o  $i$ -ésimo padrão

$$(\mathbf{x}_i, \mathbf{y}_i)$$

para a entrada

$$\mathbf{x}_i \in \mathbb{R}^{m+1}$$

e o valor de saída desejado

$$\mathbf{y}_i \in \mathbb{R}^c$$

e o valor de saída calculado  $\hat{\mathbf{y}}_i \in \mathbb{R}^c$  do modelo do MLP, pela função (eq. 5.15)

$$\hat{\mathbf{y}}_i = \mathbf{f}(\mathbf{x}_i; W_\phi, W_y). \quad (5.18)$$

A diferença quadrática na  $k$ -ésima saída do  $i$ -ésimo padrão é

$$(y_{i,k} - \hat{y}_{i,k})^2, \quad k = 1, \dots, c.$$

**Definição 5.1.1 — Erro Quadrático como Função de Perda.** Seja dado um problema de aprendizagem supervisionada, com o conjunto de treinamento de  $n$  padrões, pares de entrada e saída  $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_i, \mathbf{y}_i), \dots, (\mathbf{x}_n, \mathbf{y}_n)\}$ . Para o  $i$ -ésimo padrão, a entrada é  $\mathbf{x}_i \in \mathbb{R}^{m+1}$ , e saída desejada é  $\mathbf{y}_i \in \mathbb{R}^c$ . O modelo com os parâmetros livres  $\boldsymbol{\theta}$ , calcula pelo mapeamento  $\mathbf{y}(\mathbf{x}; \boldsymbol{\theta})$ , a  $i$ -ésima saída

$$\hat{\mathbf{y}}_i = \mathbf{f}(\mathbf{x}_i; \boldsymbol{\theta}). \quad (5.19)$$

O erro quadrático individual do  $i$ -ésimo padrão  $(\mathbf{x}_i, \mathbf{y}_i)$  é

$$\begin{aligned} L_i(\mathbf{x}_i; \boldsymbol{\theta}) &= \frac{1}{2} \sum_{k=1}^c (y_{i,k} - \hat{y}_{i,k})^2 \\ &\stackrel{a}{=} \frac{1}{2} \|\mathbf{y}_i - \hat{\mathbf{y}}_i\|^2. \end{aligned} \quad (5.20)$$

O multiplicador  $\frac{1}{2}$  simplifica as expressões quando o gradiente da função de perda é calculado. Eventualmente a perda ainda é normalizada pela quantidade  $c$  de saídas da rede, multiplicando por  $\frac{1}{c}$ , mas vamos omitir esta parte, sem modificar o resultado. A média sobre  $1 < p \leq n$  padrões<sup>a</sup>, sendo o conjunto de treinamento  $\mathcal{D}_p = \{(\mathbf{x}_{q_1}, \mathbf{y}_{q_1}), \dots, (\mathbf{x}_{q_\ell}, \mathbf{y}_{q_\ell}), \dots, (\mathbf{x}_{q_p}, \mathbf{y}_{q_p})\}$  constitui o erro quadrático do mini-lote ( $p < n$ ), ou lote ( $p = n$ ).

$$L(\mathcal{D}_p; \boldsymbol{\theta}) = \frac{1}{2p} \sum_{\ell=1}^p \sum_{k=1}^c (y_{q_\ell, k} - \hat{y}_{q_\ell, k})^2 = \frac{1}{2p} \sum_{\ell=1}^p \|\mathbf{y}_{q_\ell} - \hat{\mathbf{y}}_{q_\ell}\|^2. \quad (5.21)$$

<sup>a</sup>Em caso do mini-lote, o conjunto de treinamento contém um subconjunto  $\mathcal{D}_p$  de cardinalidade  $p$ , de todos os  $n$  padrões. Como o conjunto é uma seleção aleatória, não podemos enumerar  $\mathbf{x}_1, \mathbf{x}_2, \dots$ , pois isso seria a sequência ordenada dos padrões. Por isso temos que introduzir um subíndice  $q_\ell$  que reflete o fato que  $\mathbf{x}_{q_1}$  não é o primeiro padrão do conjunto completo, mas o primeiro padrão do mini-lote, por exemplo,  $\mathbf{x}_{q_1} = \mathbf{x}_7$ , expressando que  $\mathbf{x}_7$  será o primeiro padrão a ser submetido ao modelo.

Dessa maneira, na continuação do exemplo 5.1, podemos calcular a perda individual da (eq. 5.20), em relação à primeira flor,  $i = 1$ , como

$$\begin{aligned} L_i(\mathbf{x}_i; \boldsymbol{\theta}) &= \frac{1}{2} \|\mathbf{y}_i - \hat{\mathbf{y}}_i\|^2 = \frac{1}{2} \{(y_{i,1} - \hat{y}_{i,1})^2 + (y_{i,2} - \hat{y}_{i,2})^2 + (y_{i,3} - \hat{y}_{i,3})^2\} \\ &= \frac{1}{2} \{(1 - 0.017)^2 + (0 - 0.391)^2 + (0 - 0.262)^2\} = 0.594, \end{aligned}$$

onde os parâmetros  $\boldsymbol{\theta} = (W_\phi, W_y)$  compõem todos os pesos da rede. Observe que a perda é especificamente baseado no erro quadrático. Existem alternativas para a função de perda que estudaremos posteriormente.

### 5.1.3 Ajuste de Pesos por Descida de Gradiente

Para não sobrecarregar a nomenclatura, omitindo o índice individual  $i$ , vamos simplificar as expressões da função de perda  $L_i$  para  $L$ , do  $i$ -ésimo padrão  $\mathbf{x}_i$  para  $\mathbf{x}$ , e a  $i$ -ésima saída desejada  $\mathbf{y}_i$  e calculada  $\hat{\mathbf{y}}_i$  para  $\mathbf{y}$  e  $\hat{\mathbf{y}}$  respectivamente. Consequentemente, para as  $k$ -ésimas saídas  $y_{i,k}$  e  $\hat{y}_{i,k}$  na (eq. 5.20), temos  $y_k$  e  $\hat{y}_k$ .

O princípio básico para aprender os parâmetros livres do Perceptron com mais que zero camadas ocultas é a descida de gradiente, estudada no capítulo 3. Vamos ter a necessidade de calcular a derivada parcial da função de perda em relação a todos os pesos da rede para podermos aplicar o algoritmo da descida de gradiente. É suficiente analisar o ajuste de um peso individualmente, pois o cálculo da derivada em relação a um peso é independente do cálculo da derivada em relação a um outro peso na rede. Além disso, temos termos de derivadas parciais em relação à expressões intermediárias, por exemplo, a derivada parcial da função de perda em relação a uma ativação de um neurônio. Podemos analisar a dependência da função de perda em relação a qualquer variável definida no contexto da rede. Então a derivada parcial<sup>2</sup> da função de perda em relação a alguma variável  $v$ , considerando a (eq. 5.25), no caso do treinamento estocástico, usando o padrão  $\mathbf{x}$  como entrada é

$$\frac{\partial L(\mathbf{x}; \boldsymbol{\theta})}{\partial v}, \quad (5.22)$$

onde  $\boldsymbol{\theta}$  representa todos os pesos da rede, ou simplesmente, carregando os argumentos e parâmetros implicitamente

$$\frac{\partial L}{\partial v}.$$

Posteriormente ainda vamos precisar a seguinte regra importante de derivada parcial de uma função que tem um argumento multidimensional, e, esse argumento por sua vez é novamente multidimensional, completando as regras na definição 2.1.11 na pág. 49.

<sup>2</sup>A variável  $v$  pode ser um escalar, então o resultado  $\frac{\partial L}{\partial v}$  também é um escalar, a variável também poderia ser um vetor  $\mathbf{v}$ , então o resultado  $\frac{\partial L}{\partial \mathbf{v}}$  seria o gradiente de  $L$  em relação a  $\mathbf{v}$ , que seria um vetor com a mesma dimensão que  $\mathbf{v}$ .

**Definição 5.1.2 — Derivada parcial usada na retropropagação de erro.** Seja  $\mathbf{x} \in \mathbb{R}^n$  uma variável multidimensional. Sejam

$$\mathbf{u} : \mathbb{R}^n \rightarrow \mathbb{R}^m,$$

$$\mathbf{u}(\mathbf{x}) = \begin{bmatrix} u_1(\mathbf{x}) \\ \vdots \\ u_\ell(\mathbf{x}) \\ \vdots \\ u_m(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} u_1 \left( [x_1 \ \cdots \ x_i \ \cdots \ x_n]^T \right) \\ \vdots \\ u_\ell \left( [x_1 \ \cdots \ x_i \ \cdots \ x_n]^T \right) \\ \vdots \\ u_m \left( [x_1 \ \cdots \ x_i \ \cdots \ x_n]^T \right) \end{bmatrix}$$

$m$  funções  $u_\ell(\mathbf{x})$  do argumento vetorial  $\mathbf{x}$  de dimensão  $n$ ,  $\ell = 1, \dots, m$ . Finalmente, seja

$$\begin{aligned} y &: \mathbb{R}^n \rightarrow \mathbb{R}^m \rightarrow \mathbb{R} \\ \mathbf{x} &\mapsto \mathbf{u} \mapsto y \end{aligned} \tag{5.23}$$

uma função  $y(\mathbf{u}(\mathbf{x}))$  com um argumento multidimensional  $\mathbf{u}(\mathbf{x}) \in \mathbb{R}^m$  que por sua vez depende de outro argumento multidimensional  $\mathbf{x} \in \mathbb{R}^n$ , e resultado unidimensional  $y \in \mathbb{R}$ . Então, a derivada parcial da função  $y$  em relação a uma variável  $x_i$  é

$$\frac{\partial y}{\partial x_i} = \sum_{\ell=1}^m \frac{\partial y}{\partial u_\ell} \frac{\partial u_\ell}{\partial x_i}. \tag{5.24}$$

■ **Exemplo 5.2** Sejam  $n = 2$ ,  $m = 3$ , e

$$\begin{aligned} u_1(x_1, x_2) &= x_1^2 + 2x_2, \\ u_2(x_1, x_2) &= x_1 \sin^2(x_2), \\ u_3(x_1, x_2) &= 2x_1 + x_2^3. \end{aligned}$$

Seja

$$y(\mathbf{u}) = y(u_1, u_2, u_3) = u_1 u_2 + 4u_3.$$

Então, considerando a derivada parcial de  $y$  em relação a, por exemplo,  $x_2$  temos pela definição 5.1.2

$$\begin{aligned} \frac{\partial y}{\partial x_2} &= \sum_{\ell=1}^3 \frac{\partial y}{\partial u_\ell} \frac{\partial u_\ell}{\partial x_2} \\ &= \frac{\partial y}{\partial u_1} \frac{\partial u_1}{\partial x_2} + \frac{\partial y}{\partial u_2} \frac{\partial u_2}{\partial x_2} + \frac{\partial y}{\partial u_3} \frac{\partial u_3}{\partial x_2} \\ &= u_2 \cdot 2 + u_1 \cdot 2 \sin(x_2) \cos(x_2) + 4 \cdot 3x_2^2 \\ &= 2x_1 \sin^2(x_2) + 2(x_1^2 + 2x_2)x_1 \sin(x_2) \cos(x_2) + 12x_2^2. \end{aligned}$$

Seja então  $w$  algum peso da rede. O objetivo é calcular a derivada parcial da função de perda a este peso. É suficiente considerar a derivada em relação a um único padrão para definir o caso da aprendizagem estocástica. Para os casos de usar um mini-lote e lote, basta considerar a média desta derivada parcial do caso estocástico sobre o conjunto dos padrões no mini-lote ou o conjunto

de treinamento inteiro. Resumindo, a tarefa é obter a derivada parcial da função de perda para um único padrão  $(\mathbf{x}_i, \mathbf{y}_i)$ , em relação a um peso  $w$  da rede

$$\frac{\partial L_i(\mathbf{x}_i; \boldsymbol{\theta})}{\partial w}. \quad (5.25)$$

Como já mencionado, vamos omitir o índice individual  $i$  na função de perda. Como todos os pesos  $w$  da rede são independentes entre si, a descida de gradiente, usando a taxa de aprendizagem  $\alpha$ , é simplesmente aplicada a todos os pesos independentemente como

$$w^{\text{novo}} \leftarrow w^{\text{antigo}} - \alpha \frac{\partial L(\mathbf{x}; \boldsymbol{\theta})}{\partial w} \Big|_{w^{\text{antigo}}}. \quad (5.26)$$

Este é a estratégia básica do máximo declive, veja a (eq. 3.10) na pág. 87, o algoritmo 5 na pág. 88, e a (eq. 3.56) na pág. 111. As outras heurísticas apresentadas na seção 3.4 na pág. 111 também se baseiam exclusivamente nesta derivada  $\partial L/\partial w$  da função de perda em relação a um peso.

### Ajuste de Pesos na Camada de Saída

Vamos primeiro calcular a derivada parcial da função de perda baseada no erro quadrático em relação ao peso de destaque  $w_{h,k}$  que está na conexão do  $h$ -ésimo neurônio oculto até o  $k$ -ésimo neurônio de saída. Considere a figura 5.1. Então

$$\frac{\partial L}{\partial w_{h,k}} \stackrel{5.20}{=} \frac{\partial \left\{ \frac{1}{2} \sum_{k=1}^c (y_k - \hat{y}_k)^2 \right\}}{\partial w_{h,k}} \quad (5.27a)$$

$$\stackrel{2.22b}{=} \frac{1}{2} \sum_{k=1}^c \frac{\partial (y_k - \hat{y}_k)^2}{\partial w_{h,k}} \quad (5.27b)$$

$$= \frac{1}{2} \frac{\partial (y_k - \hat{y}_k)^2}{\partial w_{h,k}} \quad (5.27c)$$

$$\stackrel{2.38}{=} (y_k - \hat{y}_k) \frac{\partial (y_k - \hat{y}_k)}{\partial w_{h,k}} \quad (5.27d)$$

$$\stackrel{2.22c}{=} (y_k - \hat{y}_k) \left\{ \frac{\partial y_k}{\partial w_{h,k}} - \frac{\partial \hat{y}_k}{\partial w_{h,k}} \right\} \quad (5.27e)$$

$$= (y_k - \hat{y}_k) \left\{ 0 - \frac{\partial (\mathbf{w}_k \cdot \boldsymbol{\phi})}{\partial w_{h,k}} \right\} \quad (5.27f)$$

$$\stackrel{5.12}{=} -(y_k - \hat{y}_k) \left\{ \frac{\partial \sum_{h=1}^H (w_{h,k} \phi_h)}{\partial w_{h,k}} \right\} \quad (5.27g)$$

$$\stackrel{2.22b}{=} -(y_k - \hat{y}_k) \sum_{h=1}^H \frac{\partial (w_{h,k} \phi_h)}{\partial w_{h,k}} \quad (5.27h)$$

$$= -(y_k - \hat{y}_k) \frac{\partial (w_{h,k} \phi_h)}{\partial w_{h,k}} \quad (5.27i)$$

Por um momento vamos assumir que na camada de saída ainda tenha uma função de ativação  $z$  após o cálculo da ativação  $a_k$ , ou seja, na (eq. 5.27h) teria o termo

$$z(w_{h,k} \phi_h),$$

em vez de

$$w_{h,k} \phi_h.$$

Dessa maneira, a derivada da função de perda em relação ao peso  $w_{h,k}$  seria modificada a partir da (eq. 5.27h) como

$$\frac{\partial L}{\partial w_{h,k}} = -(y_k - \hat{y}_k) \frac{\partial z(w_{h,k}\phi_h)}{\partial w_{h,k}} \quad (5.27j)$$

$$\stackrel{2.38}{=} -(y_k - \hat{y}_k) z'(w_{h,k}\phi_h) \frac{\partial w_{h,k}\phi_h}{\partial w_{h,k}} \quad (5.27k)$$

$$\stackrel{5.3}{=} -(y_k - \hat{y}_k) z'(a_k)\phi_h \quad (5.27l)$$

$$= \phi_h \delta_k \quad (5.27m)$$

Na última equação ainda foi introduzida a *sensibilidade*, (*sensitivity*), ou o *delta*  $\delta$  do  $k$ -ésimo neurônio de saída como

$$\delta_k \stackrel{\text{def}}{=} -z'(a_k)(y_k - \hat{y}_k). \quad (5.28)$$

Como todas as saídas desejadas  $y_k$  são constantes, qualquer derivada em relação a esta saída é nula. Na (eq. 5.27f) este cálculo é aplicado. Para entender a transição de (eq. 5.27b) para (eq. 5.27c), observe que a  $k$ -ésima saída  $y_k$  é a única saída influenciada pelo peso  $w_{h,k}$ . Dessa maneira os termos  $(y_k - \hat{y}_k)^2$  no somatório são ignorados na (eq. 5.27b) no cálculo da derivada parcial, a não ser que o índice  $k$  coincida com o índice  $k$  do peso  $w_{h,k}$ . Temos os termos nulos

$$\frac{\partial(y_1 - \hat{y}_1)^2}{\partial w_{h,k}} = 0, \dots, \frac{\partial(y_{k-1} - \hat{y}_{k-1})^2}{\partial w_{h,k}} = 0, \frac{\partial(y_{k+1} - \hat{y}_{k+1})^2}{\partial w_{h,k}} = 0, \dots, \frac{\partial(y_c - \hat{y}_c)^2}{\partial w_{h,k}} = 0.$$

Já elaboramos esse fato no exemplo na introdução ao modelo do neurônio no exemplo 2.7 na pág. 45, veja a (eq. 2.40) na pág. 44.

Em analogia, na transição de (eq. 5.27h) para a (eq. 5.27i), resta analisar a dependência  $\partial \hat{y}_k / \partial w_{h,k}$  da  $k$ -ésima saída da rede  $\hat{y}_k$  do peso  $w_{h,k}$ . A origem é o  $h$ -ésimo neurônio  $\phi_h$  da camada oculta, e o destino é esta  $k$ -ésima saída  $y_k$ . Da (eq. 5.17) temos

$$\begin{aligned} \hat{y}_k &= \sum_{h=1}^H w_{h,k}\phi_h \\ &= w_{1,k}\phi_1 + \dots + w_{h-1,k}\phi_{h-1} + w_{h,k}\phi_h + w_{h+1,k}\phi_{h+1} + \dots + w_{H,k}\phi_H. \end{aligned} \quad (5.29)$$

Observe novamente que nesta soma, o peso relevante  $w_{h,k}$  aparece somente uma única vez. Além disso, o cálculo da característica  $\phi_h$  não depende do peso  $w_{h,k}$ , dado o fato que ele, na rede, é posterior ao cálculo de  $\phi_h$ . A consequência disso é novamente uma derivada nula, se a derivada não for calculada relativamente a este peso particular, ou seja

$$\frac{\partial w_{1,k}\phi_1}{\partial w_{h,k}} = 0, \dots, \frac{\partial w_{h-1,k}\phi_{h-1}}{\partial w_{h,k}} = 0, \quad \frac{\partial w_{h+1,k}\phi_{h+1}}{\partial w_{h,k}} = 0, \dots, \frac{\partial w_{H,k}\phi_H}{\partial w_{h,k}} = 0, \quad (5.30)$$

e somente uma derivada não nula

$$\frac{\partial w_{h,k}\phi_h}{\partial w_{h,k}} \stackrel{2.22b}{=} \phi_h.$$

Podemos então concluir que

$$\frac{\partial \hat{y}_k}{\partial w_{h,k}} = \phi_h.$$

Este resultado se aplica a todos os pesos da matriz dos pesos de saída  $W_y$ , veja a (eq. 5.13), ou seja, achamos o método de treinar todos os pesos entre a camada oculta e a camada de saída. Todos esses  $H \times c$  pesos são independentes entre si. Assim podemos aplicar a regra básica da descida de gradiente da (eq. 5.26) e formular um algoritmo básico de treinamento dos pesos da camada mais externa do Perceptron Multicamada.

---

**Algoritmo 10:** Descida de gradiente estocástica, atualização dos pesos da camada oculta até a camada de saída

---

**Entrada:** Padrão  $(\mathbf{x}_i, \mathbf{y}_i)$  de um conjunto de treinamento; Taxa de aprendizagem  $\alpha$ ;

**Resultado:** Pesos atualizados

```

1 para  $h \leftarrow 1$  até  $H$  faça
    // Para todos os neurônios  $\phi_h$  da camada oculta
2     para  $k \leftarrow 1$  até  $c$  faça
        // Para todos os neurônios  $y_k$  da camada de saída
3         Calcule a derivada parcial em relação ao peso que conecta o neurônio da camada
            oculta à camada de saída  $\frac{\partial L}{\partial w_{h,k}}$  pela (eq. 5.27) ;
4         Atualize o peso  $w_{h,k}^{\text{novo}} \leftarrow w_{h,k}^{\text{antigo}} - \alpha \frac{\partial L}{\partial w_{h,k}}$  pela (eq. 5.26);
5     fim
6 fim

```

---

A única diferença da aprendizagem estocástica em relação a usar um mini-lote ou o conjunto inteiro dos padrões do conjunto de treino é que a derivada é acumulada primeiro para os padrões do conjunto, e depois eles são atualizados.

### Ajuste de Pesos na Camada Oculta

Agora vem a parte mais complexa, o ajuste de um peso  $w_{j,h}$  que está situado logo no início da rede, da entrada a camada oculta, veja a figura 5.1. Assumimos novamente uma função de ativação após

a ativação  $a_k$  na camada de saída para considerar o caso geral. Então

$$\frac{\partial L}{\partial w_{j,h}} \stackrel{5.20}{=} \frac{\partial \left\{ \frac{1}{2} \sum_{k=1}^c (y_k - \hat{y}_k)^2 \right\}}{\partial w_{j,h}} \quad (5.31a)$$

$$\stackrel{2.22b}{=} \frac{1}{2} \sum_{k=1}^c \frac{\partial (y_k - \hat{y}_k)^2}{\partial w_{j,h}} \quad (5.31b)$$

$$\stackrel{2.38}{=} \sum_{k=1}^c (y_k - \hat{y}_k) \frac{\partial (y_k - \hat{y}_k)}{\partial w_{j,h}} \quad (5.31c)$$

$$= - \sum_{k=1}^c (y_k - \hat{y}_k) \frac{\partial \hat{y}_k}{\partial w_{j,h}} \quad (5.31d)$$

$$= - \sum_{k=1}^c (y_k - \hat{y}_k) \frac{\partial z(a_k)}{\partial w_{j,h}} \quad (5.31e)$$

$$\stackrel{2.38}{=} - \sum_{k=1}^c (y_k - \hat{y}_k) z'(a_k) \frac{\partial \sum_{h=1}^H w_{h,k} \phi_h}{\partial w_{j,h}} \quad (5.31f)$$

$$\stackrel{5.3}{=} - \sum_{k=1}^c (y_k - \hat{y}_k) z'(a_k) \sum_{h=1}^H \frac{\partial w_{h,k} \phi_h}{\partial w_{j,h}} \quad (5.31g)$$

$$= - \sum_{k=1}^c (y_k - \hat{y}_k) z'(a_k) \frac{\partial w_{h,k} \phi_h}{\partial w_{j,h}} \quad (5.31h)$$

$$\stackrel{2.22a}{=} - \sum_{k=1}^c (y_k - \hat{y}_k) z'(a_k) w_{h,k} \frac{\partial \phi_h}{\partial w_{j,h}} \quad (5.31i)$$

$$\stackrel{5.5}{=} - \sum_{k=1}^c (y_k - \hat{y}_k) z'(a_k) w_{h,k} \frac{\partial z(a_h)}{\partial w_{j,h}} \quad (5.31j)$$

$$= - \sum_{k=1}^c (y_k - \hat{y}_k) z'(a_k) w_{h,k} z'(a_h) \frac{\partial a_h}{\partial w_{j,h}} \quad (5.31k)$$

$$= - \sum_{k=1}^c (y_k - \hat{y}_k) z'(a_k) w_{h,k} z'(a_h) \frac{\partial \sum_{j=0}^m w_{j,h} x_j}{\partial w_{j,h}} \quad (5.31l)$$

$$\stackrel{2.38}{=} - \sum_{k=1}^c (y_k - \hat{y}_k) z'(a_k) w_{h,k} z'(a_h) \sum_{j=0}^m \frac{\partial w_{j,h} x_j}{\partial w_{j,h}} \quad (5.31m)$$

$$= - \sum_{k=1}^c (y_k - \hat{y}_k) z'(a_k) w_{h,k} z'(a_h) \frac{\partial w_{j,h} x_j}{\partial w_{j,h}} \quad (5.31n)$$

$$\stackrel{2.22a}{=} - \sum_{k=1}^c (y_k - \hat{y}_k) z'(a_k) w_{h,k} z'(a_h) x_j \quad (5.31o)$$

$$\stackrel{5.28}{=} x_j z'(a_h) \sum_{k=1}^c w_{h,k} \delta_k \quad (5.31p)$$

$$= x_j \delta_h. \quad (5.31q)$$

Nas transições de (eq. 5.31f) para (eq. 5.31g) e (eq. 5.31m) para (eq. 5.31n) vale o mesmo princípio como na camada de saída, elaborado na (eq. 5.27k). Somente se houver uma dependência da expressão do peso, a derivada parcial não é nula. A sensibilidade da camada oculta do  $h$ -ésimo

neurônio da camada oculta foi definida como

$$\delta_h \stackrel{\text{def}}{=} z'(a_h) \sum_{k=1}^c w_{h,k} \delta_k. \quad (5.32)$$

Observe que o vetor de pesos composto pela iteração sobre o índice  $k$  *não* é um dos vetores  $\mathbf{w}_k$  definidos na (eq. 5.11) como colunas da matriz  $W_y$  de pesos da camada da (eq. 5.13). Ele é uma *linha* da matriz  $W_y$  e representa um fluxo inverso de informação que vamos precisar na definição da retropropagação de erro. Podemos denominar este vetor com destino o  $h$ -ésimo neurônio da camada oculta, e todos as  $k = 1, \dots, c$  neurônios de saída com origem como

$$\overleftarrow{\mathbf{w}}_h \stackrel{\text{def}}{=} [w_{h,1} \ \cdots \ w_{h,k} \ \cdots \ w_{h,c}], \quad (5.33)$$

onde o símbolo  $\leftarrow$  realça a direção da informação.

É interessante de observar, como o operador da derivada trabalha para chegar cada vez mais perto do seu alvo, aplicando a regra da cadeia consequentemente. Em redes mais profundas, essa sequência de passos é cada vez mais longe, quando se analisa a dependência da função de perda em relação a pesos no início do processamento, ou seja, nas primeiras camadas da rede. Na (eq. 5.32) já podemos reconhecer a natureza recursiva do mecanismo de atualização dos pesos mais profundos na rede. A relação do delta de um neurônio na (eq. 5.32) é recursivamente definido.

#### 5.1.4 Retropropagação de Erro

O modelo do Perceptron Multicamada da figura 5.1 somente tem uma camada oculta. Se tivéssemos mais camadas ocultas, os cálculos das derivadas parciais da função de perda, como o da (eq. 5.31), em relação a pesos das primeiras camadas ficariam cada vez mais complexos. Compare a influência do peso na camada de saída da (eq. 5.27h) com a influência do peso na camada oculta da (eq. 5.31o). Felizmente tem um formalismo unificador que deixa obter essas derivadas de uma forma recursiva, a *retropropagação de erro*.

Uma definição equivalente do delta podemos obter pela regra da cadeia, separando a ativação do neurônio da função de ativação do mesmo [10].

Revisando a (eq. 5.31), temos pela regra da cadeia

$$\frac{\partial L}{\partial w_{j,h}} = \frac{\partial L}{\partial a_h} \frac{\partial a_h}{\partial w_{j,h}}, \quad (5.34)$$

e o delta do  $h$ -ésimo neurônio na camada oculta é definido alternativamente à (eq. 5.32) como

$$\delta_h \stackrel{\text{def}}{=} \frac{\partial L}{\partial a_h}. \quad (5.35)$$

Considerando o segundo termo na (eq. 5.34), da transição de (eq. 5.31n) até a (eq. 5.31o), já sabemos que

$$\frac{\partial a_h}{\partial w_{j,h}} = x_j. \quad (5.36)$$

O primeiro termo da (eq. 5.34), a definição do delta da (eq. 5.32) é

$$\begin{aligned} \delta_h &= \frac{\partial L}{\partial a_h} \\ &= \sum_{k=1}^c \frac{\partial L}{\partial a_k} \frac{\partial a_k}{\partial a_h}. \end{aligned} \quad (5.37)$$

Na última equivalência aplicamos a regra introduzida na definição 5.1.2, pois a ativação  $a_k = \mathbf{w}_k \cdot \phi(\mathbf{x})$  da (eq. 5.4) é uma função que depende de um argumento multidimensional  $\phi$ , que por sua vez depende de um argumento multidimensional  $\mathbf{x}$ .

Para calcular o primeiro termo  $\frac{\partial L}{\partial a_k}$  na (eq. 5.37), podemos reciclar quase totalmente a sequência da (eq. 5.27). Somente na (eq. 5.27k) o resultado da derivada é diferente. Temos da (eq. 5.27j)

$$\frac{\partial L}{\partial a_k} = -(y_k - \hat{y}_k) \frac{\partial z(a_k)}{\partial a_k} \quad (5.38a)$$

$$= -(y_k - \hat{y}_k) z'(a_k) \frac{\partial a_k}{\partial a_k} \quad (5.38b)$$

$$= -(y_k - \hat{y}_k) z'(a_k) \quad (5.38c)$$

$$\stackrel{5.28}{=} \delta_k. \quad (5.38d)$$

Para o segundo termo  $\frac{\partial a_k}{\partial a_h}$  na (eq. 5.34) temos

$$\frac{\partial a_k}{\partial a_h} \stackrel{5.4}{=} \frac{\partial \sum_{h=1}^H w_{h,k} \phi_h}{\partial a_h} \quad (5.39a)$$

$$\stackrel{5.22b}{=} \sum_{h=1}^H \frac{\partial w_{h,k} \phi_h}{\partial a_h} \quad (5.39b)$$

$$= \frac{\partial w_{h,k} \phi_h}{\partial a_h} \quad (5.39c)$$

$$= w_{h,k} \frac{\partial \phi_h}{\partial a_h} \quad (5.39d)$$

$$= w_{h,k} \frac{\partial z(a_h)}{\partial a_h} \quad (5.39e)$$

$$= w_{h,k} z'(a_h) \frac{\partial a_h}{\partial a_h} \quad (5.39f)$$

$$= w_{h,k} z'(a_h) \quad (5.39g)$$

Juntando os resultados da (eq. 5.38) e (eq. 5.39) na (eq. 5.37), finalmente temos

$$\begin{aligned} \delta_h &= \frac{\partial L}{\partial a_h} \\ &= \sum_{k=1}^c \frac{\partial L}{\partial a_k} \frac{\partial a_k}{\partial a_h} \\ &\stackrel{5.38, 5.39}{=} \sum_{k=1}^c \delta_k w_{h,k} z'(a_h) \\ &= z'(a_h) \sum_{k=1}^c w_{h,k} \delta_k \end{aligned} \quad (5.40)$$

equivalente à definição de  $\delta_h$  da (eq. 5.32).

Podemos associar cada delta a um neurônio da camada oculta e camada de saída. O importante na definição do delta  $\delta_h$  na camada oculta é a sua dependência. Precisamos somente a sua entrada  $a_h$  e a soma ponderada dos deltas  $\delta_k$  que sofrem a influência do  $h$ -ésimo neurônio da camada oculta. Então é fácil calcular a expressão completa  $x_j \delta_h$  da (eq. 5.31p) que é necessária para aplicar a descida de gradiente.

Com somente uma camada oculta, a essência da retropropagação de erro ainda não fica bem visível, pois temos somente o delta  $\delta_k$  na saída e o delta  $\delta_h$  da camada oculta. A fórmula desses dois deltas é bastante diferente. Para remediar isso, vamos introduzir mais uma camada oculta e assim transitivamente explicar a retropropagação de erro entre duas camadas ocultas consecutivas quaisquer. Na figura 5.2, as entradas da figura 4.5 foram substituídas por neurônios de uma camada oculta, porém o símbolo da variável  $x$  foi mantido. Vamos omitir também os limites nos somatórios, e somente usar a variável de contagem. Sendo irrelevante para estudar a teoria, podemos manter o viés na nova camada oculta, ou seja, começar com  $x_0$ , ou ignorar, começando com  $x_1$ . Para retropropagar o  $\delta_h$  da camada oculta mais à direita, para a nova camada oculta anterior, temos que calcular o  $\delta_j$  para cada neurônio  $x_j$  da nova camada oculta. Pela definição da (eq. 5.35), temos

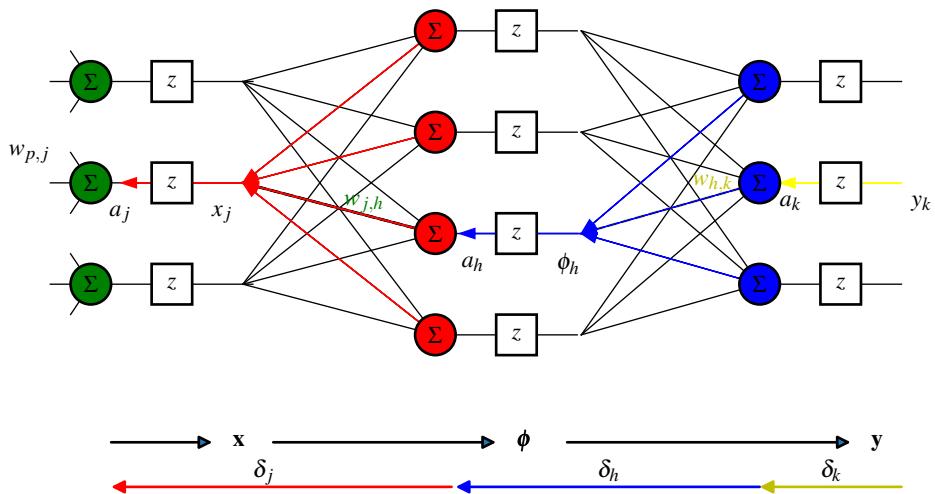


Figura 5.2: Rede neural com duas camadas ocultas não lineares e a camada de saída não linear para ilustrar a retropropagação de erro, necessária no cálculo da derivada da função de perda em relação a um peso na rede. As entradas da rede não são mostrados.

$$\begin{aligned}\delta_j &= \frac{\partial L}{\partial a_j} \\ &\stackrel{\parallel 5.24}{=} \sum_h \frac{\partial L}{\partial a_h} \frac{\partial a_h}{\partial a_j}\end{aligned}\tag{5.41a}$$

$$\stackrel{\parallel 5.35}{=} \sum_h \delta_h \frac{\partial a_h}{\partial a_j}\tag{5.41b}$$

$$\stackrel{\parallel 5.3}{=} \sum_h \delta_h \frac{\partial \sum_j w_{j,h} x_j}{\partial a_j}\tag{5.41c}$$

$$\stackrel{\parallel 2.22b}{=} \sum_h \delta_h \sum_j \frac{\partial w_{j,h} x_j}{\partial a_j}\tag{5.41d}$$

$$= \sum_h \delta_h \frac{\partial w_{j,h} x_j}{\partial a_j}\tag{5.41e}$$

$$= \sum_h \delta_h w_{j,h} \frac{\partial z(a_j)}{\partial a_j}\tag{5.41f}$$

$$\stackrel{\parallel 2.38}{=} \sum_h \delta_h w_{j,h} z'(a_j) \frac{\partial a_j}{\partial a_j}\tag{5.41g}$$

$$= z'(a_j) \sum_h w_{j,h} \delta_h.\tag{5.41h}$$

Na transição da (eq. 5.41d) para a (eq. 5.41e), o somatório some, pois a dependência é nula, se o índice  $j$  do peso  $w_{j,h}$  não coincide com o índice  $j$  da ativação  $a_j$ . Esse fato já foi várias vezes comentado, por exemplo na (eq. 5.30). Novamente temos a definição recursiva do delta  $\delta_j$  de um neurônio  $x_j$  baseada na combinação linear dos  $\delta_h$  da camada oculta subsequente, adicionalmente ainda a derivada da função de ativação da ativação  $a_j$  que entre no neurônio  $x_j$ . Seja  $p$  o índice contador do neurônios anterior à camada oculta os elementos  $x_j$ . Para ajustar pesos  $w_{p,j}$  anterior a camada oculta dos  $x_j$  basta o  $\delta_j$ , e a respectiva entrada associada ao peso  $w_{p,j}$ .

Temos finalmente um esquema recursivo da retropropagação dos deltas que permite facilmente ajustar os pesos em camadas cada vez mais profundas, do ponto de vista da saída da rede. Com o objetivo de compactar as expressões, vamos juntar os deltas  $\delta$  de cada camada em um vetor  $\boldsymbol{\delta}$ , as atividades  $a$  de cada camada igualmente em um vetor  $\mathbf{a}$ , e a passagem dessas ativações  $a$  por uma função de ativação  $z$  em um vetor  $\mathbf{z}$ . Este vetor é composto pelas passagens individuais de cada ativação da  $\ell$ -ésima camada, ou seja

$$\mathbf{z}^{(\ell)} \stackrel{\text{def}}{=} \begin{bmatrix} z(a_1^{(\ell)}) \\ z(a_2^{(\ell)}) \\ \vdots \end{bmatrix}.\tag{5.42}$$

As derivadas individuais das ativações podemos agrupar no vetor

$$\mathbf{z}'^{(\ell)} \stackrel{\text{def}}{=} \begin{bmatrix} z'(a_1^{(\ell)}) \\ z'(a_2^{(\ell)}) \\ \vdots \end{bmatrix}.\tag{5.43}$$

Já temos o agrupamento de todos os pesos entre a entrada e a primeira camada, e entre duas camadas em uma matriz  $W$  de pesos.

A figura 5.3 resume todos os fluxos de informações nos cálculos de todas as variáveis envolvidas. A rede mostrada tem duas camadas ocultas e uma camada de saída. A entrada de fato não é uma camada, mas recebe a enumeração zero. O índice da camada está marcado sobrescrito entre parênteses. Por exemplo, o vetor que contém a passagem de cada componente da ativação pela função de ativação após a primeira camada oculta é  $\mathbf{z}^{(2)}$ . Com o objetivo de uniformizar os vetores em cada posição da rede, o vetor de entrada  $\mathbf{x}$  então equivale ao vetor de ativação  $\mathbf{a}^{(0)}$ . Ele é multiplicado pela matriz que contém todos os pesos da entrada até a primeira camada oculta, para produzir o vetor de ativação  $\mathbf{a}^{(1)}$ . A matriz, como contém os pesos da camada zero até a primeira camada, é chamado  $W_{0 \rightarrow 1}$ . Em geral a matriz que contém os pesos entre a camada  $\ell$  e a próxima camada  $(\ell + 1)$  será denominada

$$W_{\ell \rightarrow (\ell+1)}.$$

A quantidade total de todas as camadas é

$$\ell_{\max},$$

composta por  $\ell_{\max} - 1$  camadas ocultas é uma camada de saída. A entrada não é considerada camada, mas recebe o índice  $\ell = 0$ . A passagem por componente pela função de ativação  $z$  produz o vetor  $\mathbf{z}^{(1)}$ . Em geral, de uma camada oculta até a próxima, ou da última camada oculta até a camada de saída, a ativação é

$$\mathbf{a}^{(\ell+1)} = W_{\ell \rightarrow (\ell+1)} \mathbf{z}^{(\ell)}. \quad (5.44)$$

Após a passagem de cada componente da última ativação pela função de ativação, a saída da rede equivale ao vetor de saída  $\hat{\mathbf{y}}$ . Neste exemplo é  $\mathbf{z}^{(3)} = \mathbf{z}^{(\ell_{\max})}$ . Junto com o vetor  $\mathbf{y}$  dos valores desejados, a função de perda  $L$  pode ser calculada.

O objetivo do algoritmo de treinamento da rede é obter a derivada parcial em relação a cada peso da rede, ou seja, calcular

$$\frac{\partial L}{\partial w_{\ell \rightarrow (\ell+1)}},$$

onde o peso  $w_{\ell \rightarrow (\ell+1)}$  representa cada peso da matriz de pesos  $W_{\ell \rightarrow (\ell+1)}$ . Para obter essa derivada parcial precisamos o vetor de deltas  $\delta^{(\ell+1)}$  e o vetor  $\mathbf{z}^{(\ell+1)}$ . Podemos compactar os cálculos dessas derivadas parciais, juntando cada componente em uma matriz. Então

$$\frac{\partial L}{\partial W_{\ell \rightarrow (\ell+1)}} = \mathbf{z}^{(\ell)} \delta^{(\ell+1)\top}. \quad (5.45)$$

A (eq. 5.31p)

$$\frac{\partial L}{w_{j,h}} = -x_j \delta_h$$

é uma especialização da (eq. 5.45) para um peso da matriz de pesos  $W_\phi$  da entrada até a camada oculta.

A definição recursiva multidimensional dos deltas podemos expressar como

$$\delta^{(\ell)} = \mathbf{z}'^{(\ell)} \odot W_{\ell \rightarrow (\ell+1)} \delta^{(\ell+1)} = \begin{bmatrix} z_1'^{(\ell)} \overleftarrow{W}_1 \delta^{(\ell+1)} \\ z_2'^{(\ell)} \overleftarrow{W}_2 \delta^{(\ell+1)} \\ \vdots \end{bmatrix}, \quad (5.46)$$

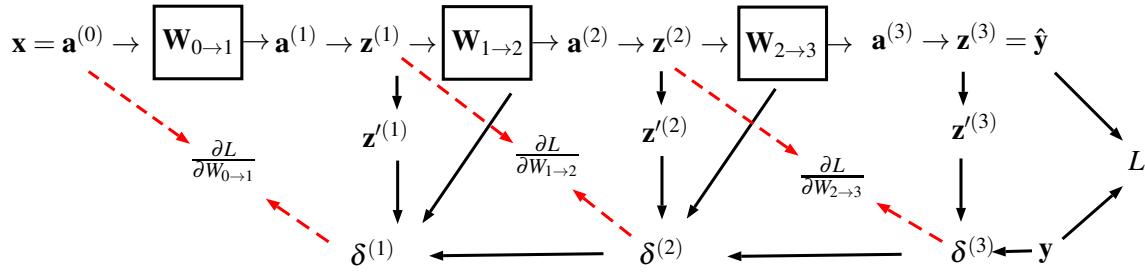


Figura 5.3: Fluxos de informação em um Perceptron Multicamada. A rede tem duas camadas ocultas.

onde o símbolo “ $\odot$ ” significa multiplicação elemento por elemento. A expressão recursiva dos deltas  $\delta_h$  da (eq. 5.32) é um exemplo de uma linha da (eq. 5.46), onde a camada  $\ell$  é a camada oculta, e a camada  $(\ell + 1)$  é a camada de saída.

Podemos finalmente formular o algoritmo 11 de retropropagação de erro, como um passo de descida de gradiente estocástica. O algoritmo calcula o gradiente da função de perda em relação a cada peso da rede. Este gradiente é a base de todas as variações de heurísticas de primeira ordem, estudadas na seção 3.4. Os passos valem para a versão estocástica, usando uma entrada individual  $x_i$  e saída  $y_i$ , mas são facilmente estendíveis para mini-lote e lote, como já mencionado diversas vezes.

---

#### Algoritmo 11: Retropropagação de Erro para ajustar pesos em Perceptron Multicamada

---

**Entrada:** Padrão  $(x, y)$  de um conjunto de treinamento; Taxa de aprendizagem  $\alpha$ ; Função de perda  $L$ ;

**Resultado:** Pesos atualizados

// Passo adiante: Propague a entrada  $x$  pelas camadas até a saída  $\hat{y}$  do modelo

- 1  $\hat{y} = f(x; \theta)$ ,  $\theta = (W_{0 \rightarrow 1}, \dots, W_{\ell \rightarrow (\ell+1)}, \dots)$  // (eq. 5.1)
  - 2 // Cálculo da função de perda, neste caso o erro quadrático
  - 3  $L \leftarrow \frac{1}{2} \|y - \hat{y}\|^2$  // (eq. 5.20)
  - 4 // Inicialização dos  $\delta$  da camada de saída
  - 5  $\delta^{(\ell_{\max})} \leftarrow -z^{(\ell_{\max})} \odot (y - \hat{y})$ ;
  - 6 // Cálculo das derivadas parciais de todos os pesos da última camada oculta até a camada de saída
  - 7  $\partial L / \partial W_{(\ell_{\max}-1) \rightarrow \ell_{\max}} \leftarrow z^{(\ell_{\max}-1)} \delta^{(\ell_{\max})^T}$  // (eq. 5.45)
  - 8 // Para todas as camadas, da última camada oculta até a entrada da rede, aplique a retropropagação dos deltas
  - 9 para  $\ell \leftarrow \ell_{\max-1}$  até 1 faça
    - 10 // Cálculo recursivo dos deltas da camada  $\ell$
    - 11  $\delta^{(\ell)} \leftarrow z^{(\ell)} \odot W_{\ell \rightarrow (\ell+1)} \delta^{(\ell+1)}$  // (eq. 5.46)
    - 12 // Cálculo das derivadas parciais
    - 13  $\partial L / \partial W_{(\ell-1) \rightarrow \ell} \leftarrow z^{(\ell-1)} \delta^{(\ell)^T}$ ;
  - 14 fim
- 

■ **Exemplo 5.3** Vamos usar a rede definida na figura 5.3 para explicitar um único passo na descida de gradiente estocástica. O problema é novamente a classificação das flores, usando duas características de entrada. A rede tem duas camadas ocultas e uma camada de saída. A primeira

camada oculta tem três neurônios, a segunda quatro neurônios, e na saída temos três neurônios, correspondendo ao número de classes. A função de ativação das camadas de saída é a sigmóide logística  $z(a) = 1/(1 + e^{-a})$ , e a função de ativação da camada de saída é a função de identidade  $z(a) = a$ . O padrão usado é novamente a primeira setosa, usando as primeiras duas características, com o vetor aumentado

$$\mathbf{x} = [x_0 \quad x_1 \quad x_2]^\top = [1 \quad 5.1 \quad 3.5]^\top.$$

Considerando a figura 5.3, este vetor  $3 \times 1$  de entrada é considerado a 0-ésima ativação  $\mathbf{x} = \mathbf{a}^{(0)}$ . A matriz  $3 \times 3$  de pesos da entrada até a primeira camada oculta com 3 neurônios ocultos foi aleatoriamente inicializada para o valor

$$W_{0 \rightarrow 1} = \begin{bmatrix} 0.167 & -0.063 & 0.065 \\ 0.318 & 0.065 & -0.370 \\ -0.206 & -0.238 & -0.347 \end{bmatrix}.$$

Assim a primeira ativação após a primeira camada oculta pode ser calculada como

$$\mathbf{a}^{(1)} = W_{0 \rightarrow 1}^\top \mathbf{a}^{(0)} = [0.901 \quad -0.502 \quad -3.098]^\top.$$

Após a passagem individual desses valores pela função de ativação temos

$$\mathbf{z}^{(1)} = [z(a_1^{(1)}) \quad z(a_2^{(1)}) \quad z(a_3^{(1)})]^\top = [0.711 \quad 0.377 \quad 0.043]^\top.$$

No cálculo do  $\delta^{(1)}$  também vamos precisar a derivada

$$\mathbf{z}'^{(1)} = [z'(a_1^{(1)}) \quad z'(a_2^{(1)}) \quad z'(a_3^{(1)})]^\top = [0.221 \quad 0.241 \quad 0.250]^\top.$$

Os pesos da primeira camada oculta a segunda camada oculta pela quantidade de neurônios de cada camada é a matriz  $3 \times 4$

$$W_{1 \rightarrow 2} = \begin{bmatrix} -0.428 & -0.363 & -0.298 & -0.154 \\ 0.524 & 0.286 & 0.339 & 0.270 \\ -0.339 & 0.482 & -0.186 & 0.226 \end{bmatrix}.$$

Então a ativação após a primeira camada oculta é

$$\mathbf{a}^{(2)} = W_{1 \rightarrow 2}^\top \mathbf{z}^{(1)} = [-0.121 \quad -0.129 \quad -0.092 \quad 0.002]^\top,$$

a passagem pela função de ativação e a respectiva derivada são

$$\mathbf{z}^{(2)} = [z(a_1^{(2)}) \quad z(a_2^{(2)}) \quad z(a_3^{(2)}) \quad z(a_4^{(2)})]^\top = [0.470 \quad 0.468 \quad 0.477 \quad 0.501]^\top.$$

$$\mathbf{z}'^{(2)} = [z'(a_1^{(2)}) \quad z'(a_2^{(2)}) \quad z'(a_3^{(2)}) \quad z'(a_4^{(2)})]^\top = [0.249 \quad 0.249 \quad 0.249 \quad 0.250]^\top.$$

Finalmente, temos a matriz de pesos  $4 \times 3$  da última camada oculta até a camada de saída

$$W_{2 \rightarrow 3} = \begin{bmatrix} -0.407 & -0.004 & 0.508 \\ -0.428 & 0.187 & -0.421 \\ 0.408 & -0.274 & -0.366 \\ 0.312 & 0.389 & 0.256 \end{bmatrix},$$

a ativação

$$\mathbf{a}^{(3)} = W_{2 \rightarrow 3}^T \mathbf{z}^{(2)} = [-0.041 \quad 0.149 \quad -0.005]^T,$$

e como a função de ativação da camada de saída é a função de identidade, temos como a saída da rede os mesmos valores.

$$\hat{\mathbf{y}} = \mathbf{z}^{(3)} = \mathbf{a}^{(3)}.$$

Este resultado encerra a fase de propagação adiante. O valor da função de perda para o padrão submetido é

$$\begin{aligned} L(\mathbf{x}; \boldsymbol{\theta}) &= \frac{1}{2} \|\mathbf{y} - \hat{\mathbf{y}}\|^2 = \frac{1}{2} \{(y_1 - \hat{y}_1)^2 + (y_2 - \hat{y}_2)^2 + (y_3 - \hat{y}_3)^2\} \\ &= \frac{1}{2} \{(1 - -0.041)^2 + (0 - 0.150)^2 + (0 - -0.005)^2\} = 1.105. \end{aligned}$$

A derivada da última ativação é

$$\mathbf{z}'^{(3)} = [z'(a_1^{(3)}) \quad z'(a_2^{(3)}) \quad z'(a_3^{(3)})]^T = [1 \quad 1 \quad 1]^T,$$

pois a derivada da função de identidade  $z(a) = a$  é  $z'(a) = 1$ .

Podemos iniciar a fase de retropropagação de erro, junto com o cálculo da derivada parcial da função de perda em relação a cada peso de uma matriz de pesos antes de cada camada. O delta inicial da última camada  $\ell_{\max} = 3$  é

$$\boldsymbol{\delta}^{(3)} = -\mathbf{z}'^{(3)} \odot (\mathbf{y} - \hat{\mathbf{y}}) = -\begin{bmatrix} z'(a_1^{(3)})(y_1 - \hat{y}_1) \\ z'(a_2^{(3)})(y_2 - \hat{y}_2) \\ z'(a_3^{(3)})(y_3 - \hat{y}_3) \end{bmatrix} = -\begin{bmatrix} 1(1 - -0.041) \\ 1(0 - 0.150) \\ 1(0 - -0.005) \end{bmatrix} = \begin{bmatrix} -1.041 \\ 0.149 \\ -0.005 \end{bmatrix}.$$

Assim podemos calcular as derivadas da função de perda em relação a todos os pesos da matriz de pesos da última camada oculta a camada de saída. Então temos

$$\partial L / \partial W_{2 \rightarrow 3} \leftarrow \mathbf{z}^{(2)} \boldsymbol{\delta}^{(3)T} = \begin{bmatrix} 0.470 \\ 0.468 \\ 0.477 \\ 0.501 \end{bmatrix} \begin{bmatrix} -1.041 \\ 0.149 \\ -0.005 \end{bmatrix}^T = \begin{bmatrix} -0.4889 & 0.0701 & -0.0023 \\ -0.4867 & 0.0698 & -0.0023 \\ -0.4963 & 0.0712 & -0.0024 \\ -0.521 & 0.0747 & -0.0025 \end{bmatrix}$$

Retropropagando o delta, temos

$$\boldsymbol{\delta}^{(2)} = \mathbf{z}'^{(2)} \odot W_{2 \rightarrow 3} \boldsymbol{\delta}^{(3)} = \begin{bmatrix} 0.249 \\ 0.249 \\ 0.249 \\ 0.250 \end{bmatrix} \odot \begin{bmatrix} -0.407 & -0.004 & 0.508 \\ -0.428 & 0.187 & -0.421 \\ 0.408 & -0.274 & -0.366 \\ 0.312 & 0.389 & 0.256 \end{bmatrix} \begin{bmatrix} -1.041 \\ 0.149 \\ -0.005 \end{bmatrix} = \begin{bmatrix} 0.105 \\ 0.118 \\ -0.116 \\ -0.067 \end{bmatrix}.$$

Assim podemos calcular as derivadas relativos à matriz de pesos da primeira a segunda camada oculta

$$\partial L / \partial W_{1 \rightarrow 2} \leftarrow \mathbf{z}^{(1)} \boldsymbol{\delta}^{(2)T} = \begin{bmatrix} 0.711 \\ 0.377 \\ 0.043 \end{bmatrix} \begin{bmatrix} 0.105 \\ 0.118 \\ -0.116 \\ -0.067 \end{bmatrix}^T = \begin{bmatrix} 0.0745 & 0.0842 & -0.0823 & -0.0476 \\ 0.0395 & 0.0446 & -0.0436 & -0.0252 \\ 0.0045 & 0.0051 & -0.0050 & -0.0029 \end{bmatrix}$$

Continuando retropropagando, temos para o delta e as derivadas

$$\boldsymbol{\delta}^{(1)} = \mathbf{z}'^{(1)} \odot W_{1 \rightarrow (2)} \boldsymbol{\delta}^{(2)} = \begin{bmatrix} 0.205 \\ 0.235 \\ 0.041 \end{bmatrix} \odot \begin{bmatrix} -0.428 & -0.363 & -0.298 & -0.154 \\ 0.524 & 0.286 & 0.339 & 0.270 \\ -0.339 & 0.482 & -0.186 & 0.226 \end{bmatrix} \begin{bmatrix} 0.105 \\ 0.118 \\ -0.116 \\ -0.067 \end{bmatrix} = \begin{bmatrix} -0.0088 \\ 0.0074 \\ 0.0016 \end{bmatrix}.$$

$$\partial L / \partial W_{0 \rightarrow 1} \leftarrow \mathbf{x} \boldsymbol{\delta}^{(1)^\top} = \begin{bmatrix} 1 \\ 5.1 \\ 3.5 \end{bmatrix} \begin{bmatrix} -0.0088 \\ 0.0074 \\ 0.0016 \end{bmatrix}^\top = \begin{bmatrix} -0.0088 & 0.0074 & 0.0012 \\ -0.0450 & 0.0377 & 0.0059 \\ -0.0309 & 0.0258 & 0.0040 \end{bmatrix}$$

A partir de agora a primeira derivada da função de perda para um padrão, em relação a um peso da rede pode ser usado para possibilitar a descida de gradiente em todas as suas variações, por exemplo aplicar as heurísticas discutidas na seção 3.4. ■

### 5.1.5 Funções de Perda e seus Gradientes

### 5.1.6 Funções de Ativação

Em geral, o poder de cálculo do MLP é conseguido pela sua não linearidade. Isto significa que em algum ponto da sequência de processamento, temos que ter uma função não linear. No modelo da figura 5.1, esse papel é assumido pela função de ativação  $z$  na camada oculta. Podemos fazer um experimento mental e omitir a função de ativação. Equivalentemente podemos usar uma função de ativação linear que na sua forma mais básica simplesmente copia a sua entrada para a sua saída, ou seja

$$z : \mathbb{R} \rightarrow \mathbb{R}, \quad z(a) = a.$$

Então teríamos um mapeamento linear da entrada até a camada oculta

$$\boldsymbol{\phi} = W_\phi^\top \mathbf{x},$$

igual à Máquina linear, e depois teríamos novamente um mapeamento linear

$$\mathbf{y} = W_y^\top \boldsymbol{\phi} = W_y^\top W_\phi^\top \mathbf{x}. \quad (5.47)$$

Poderíamos definir uma matriz  $W$  de dimensão  $(m+1) \times c$

$$W \stackrel{\text{def}}{=} W_\phi W_y,$$

que na sua versão transposta de dimensão  $c \times (m+1)$  seria

$$W^\top \stackrel{\text{def}}{=} W_y^\top W_\phi^\top,$$

aplicando o cálculo inteiro desde a entrada a saída. Então temos um cálculo final de

$$\mathbf{y} = W^\top \mathbf{x},$$

idêntico à Maquina Linear, compare com a (eq. 2.65) na pág. 53. Podemos concluir que para ter um mapeamento funcional não linear entre a entrada de um MLP e a sua saída, temos que ter uma função de ativação não linear na camada oculta, após o produto escalar do vetor de entrada  $\mathbf{x}$  e o vetor de pesos  $\mathbf{w}_h$ .

Na tabela 5.1 mostra-se uma seleção de funções de ativação, veja também [40] para uma lista mais completa.

## 5.2 Rede Convolucional, CNN

### 5.2.1 Modelo

Nome	Definição $z(a)$	Derivada $z'(a) = dz(a)/da$
Identidade	$a$	1
Sinal	$\text{sgn}(a) = \begin{cases} -1, & \text{se } a < 0 \\ 0, & \text{se } a = 0 \\ 1, & \text{se } a > 0 \end{cases}$	0
ReLU ( <i>Rectified linear unit</i> )	$\max\{0, a\} = \begin{cases} 0, & \text{se } a \leq 0 \\ a, & \text{se } a > 0 \end{cases}$	$\begin{cases} 0, & \text{se } a \leq 0 \\ 1, & \text{se } a > 0 \end{cases}$
Sigmóide logística	$\frac{1}{1+\exp(-a)}$	$z(a)[1-z(a)]$
Tangente hiperbólica	$\tanh a = \frac{\exp(a)-\exp(-a)}{\exp(a)+\exp(-a)}$	$[1-z(a)]^2$

Tabela 5.1: Funções de ativação

# 6. Redes Neurais Baseadas em Energia

## 6.1 Rede de Hopfield

A rede de Hopfield [53] faz parte de uma categoria de redes recorrentes, ou seja, que tem um comportamento dinâmico, se é apresentado um padrão  $\mathbf{x}$ . Em analogia à física, aos modelos de vidro de spin (*spin glass*), que são modelos de magnetismo com propriedades estocásticos, [48], define-se um modelo de *energia*. Existem duas fases, treinamento e consulta. O treinamento é determinístico, os pesos podem ser obtidos por uma forma fechada, semelhante ao treinamento da máquina linear, o treinamento pode ser incremental também, ou seja, um novo padrão pode ser incluído na rede já treinada. A aprendizagem da rede de Hopfield é não supervisionada. A aplicação principal desta arquitetura é a *auto-associação*. Isto significa que a rede memoriza padrões, e que na apresentação de um padrão qualquer à rede, a resposta deve ser um dos padrões memorizados. Este comportamento sugere que a rede consiga recuperar informação, até com perturbações na apresentação de um padrão conhecido.

### 6.1.1 Modelo da Rede de Hopfield

Não existem camadas nesta arquitetura de redes neurais artificiais. Um neurônio  $x$  é uma unidade binária, normalmente com os dois valores possíveis

$$x \in \{-1, 1\}. \quad (6.1)$$

A estado da rede toda pode ser representada por um vetor de dimensão  $H$

$$\mathbf{x} = [x_1 \ \dots \ x_i \ \dots \ x_j \ \dots \ x_H]^T. \quad (6.2)$$

Cada neurônio  $x_i$  está ligado a cada outro neurônio  $x_j$ , valendo as seguintes restrições:

1. A ligação entre  $x_i$  e  $x_j$  é definida por um peso contínuo

$$w_{i,j} \in \mathbb{R};$$

2. Não existe uma ligação de um neurônio com ele próprio, sendo equivalente a um peso nulo

$$w_{i,i} = 0, \quad i = 1, \dots, H; \quad (6.3)$$

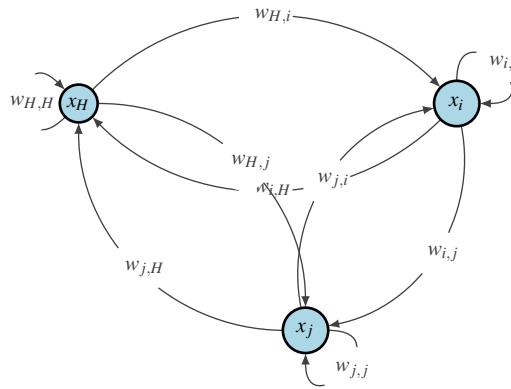


Figura 6.1: Rede de Hopfield.

3. Os pesos entre duas unidades são simétricos, ou seja,

$$w_{i,j} = w_{j,i}, \quad i, j = 1, \dots, H. \quad (6.4)$$

Todos os  $H$  pesos efluentes do neurônio  $x_i$  são agrupados no vetor  $\mathbf{w}_{i,\bullet}$ , todos os  $H$  pesos afluentes do neurônio  $x_i$  são agrupados no vetor  $\mathbf{w}_{\bullet,i}$ , todos os  $H^2$  pesos da rede são agrupados na matriz

$$\begin{aligned} W &= \left[ \mathbf{w}_{1,\bullet}^\top \quad \cdots \quad \mathbf{w}_{i,\bullet}^\top \quad \cdots \quad \mathbf{w}_{H,\bullet}^\top \right]^\top \\ &= \left[ \mathbf{w}_{\bullet,1} \quad \cdots \quad \mathbf{w}_{\bullet,i} \quad \cdots \quad \mathbf{w}_{\bullet,H} \right] \\ &= \begin{bmatrix} w_{1,1} & \cdots & w_{1,i} & \cdots & w_{1,j} & \cdots & w_{1,H} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ w_{i,1} & \cdots & w_{i,i} & \cdots & w_{i,j} & \cdots & w_{i,H} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ w_{j,1} & \cdots & w_{j,i} & \cdots & w_{j,j} & \cdots & w_{j,H} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ w_{H,1} & \cdots & w_{H,i} & \cdots & w_{H,j} & \cdots & w_{H,H} \end{bmatrix}. \end{aligned} \quad (6.5)$$

Note que a matriz é simétrica

$$W = W^\top,$$

e nula na diagonal. Consequentemente existem

$$\frac{H}{2}(H - 1) \in \mathbb{N}$$

pesos diferentes que constituem o “conhecimento” adquirido após a fase de treinamento.

A figura 6.1 mostra um esquema básico com três neurônios, todos interligados. O objetivo da rede de Hopfield é a criação de uma *memória associativa*, ou seja, apresentando um padrão  $\mathbf{x}$  à rede, ela deve responder um padrão, idealmente o padrão  $\mathbf{x}$ , se faz parte da memória. Padrões parecidos com  $\mathbf{x}$  também devem provocar a resposta do padrão memorizado  $\mathbf{x}$ . A resposta da rede é um processo dinâmico com um número sempre finito de passos que finalmente leva a uma resposta bem definida da rede.

A natureza dinâmica da rede implica que o estado da rede da (eq. 6.2) está definido para instantes de tempos discretos sequenciais

$$0, 1, \dots, \tau - 1, \tau, \tau + 1, \dots,$$

com o estado inicial

$$\mathbf{x}(0) = [x_1(0) \ \dots \ x_i(0) \ \dots \ x_j(0) \ \dots \ x_H(0)]^\top,$$

e o estado atual

$$\mathbf{x}(\tau) = [x_1(\tau) \ \dots \ x_i(\tau) \ \dots \ x_j(\tau) \ \dots \ x_H(\tau)]^\top.$$

A atualização de cada componente do estado atual pode ser feita de forma *síncrona* e de forma *assíncrona*. Na versão síncrona, uma cópia  $\tilde{\mathbf{x}}(\tau + 1)$  do estado atualizado é gerada, usando exclusivamente o estado atual  $\mathbf{x}(\tau)$ . Após a atualização de todos os neurônios, o estado atualizado é substituído pela cópia, ou seja,  $\mathbf{x}(\tau + 1) \leftarrow \tilde{\mathbf{x}}(\tau + 1)$ . A atualização síncrona é menos usada na simulação do dinamismo da rede. Além disso, podem surgir oscilações na fase de consulta, ou seja, padrões  $\mathbf{x}(\tau)$  podem ocorrer mais que uma vez na atualização dos pesos, o que implica que a rede não converge para um estado final.

Na forma *assíncrona*, um neurônio é aleatoriamente escolhido e atualizado pelo estado dos restantes neurônios, até que todos os  $H$  neurônios foram atualizados pela regra

$$x_i = \text{sgn} \left( \sum_{j=1}^H w_{i,j} x_j \right) = \text{sgn} (\mathbf{w}_{i,\bullet} \cdot \mathbf{x}). \quad (6.6)$$

Reconhecemos no cálculo do estado do neurônio  $x_i$  o esquema básico de uma combinação linear das “entradas”  $x_j$ , seguida por uma função de ativação, neste caso a função do sinal da (eq. 1.8). A diferença importante deste cálculo de  $x_i$  em relação ao uso da combinação linear de entradas, seguida por uma função de ativação usada no contexto de um classificador linear binário da (eq. 2.87) na pág. 69, é que a saída é realimentada como novo estado do  $i$ -ésimo neurônio da rede. Além disso, na (eq. 6.6) não existe viés, ou seja, comparada com a definição 2.4.4 na pág. 69, temos aqui  $b = 0$ .

Mais especificamente, a atualização assíncrona dos pesos está formalizada no algoritmo 12.

---

**Algoritmo 12:** Atualização assíncrona do estado da rede de Hopfield

---

**Entrada:** Estado atual  $\mathbf{x}(\tau)$  da rede de Hopfield

**Resultado:** Estado atualizado  $\mathbf{x}(\tau + 1)$

- 1 Crie uma permutação  $p$  aleatória da sequência de todos os neurônios  $1, \dots, H$ ;
  - 2 **para**  $i \leftarrow 1$  até  $H$  **faça**
    - 3     // Para todos os neurônios  $x_i$ , em ordem aleatória
    - 4      $q \leftarrow p(i)$  /\* Escolhe um índice aleatoriamente, sem reposição \*/;
    - 5      $x_q(\tau + 1) = \text{sgn} (\sum_{j=1}^H w_{q,j} x_j(\tau)) = \text{sgn} (\mathbf{w}_{q,\bullet} \cdot \mathbf{x}(\tau))$
  - 5 **fim**
- 

### 6.1.2 Memória Auto-Associativa

Uma das possíveis aplicações da rede de Hopfield é a capacidade de armazenar padrões e recuperar um destes padrões na fase de consulta à rede. Existe um método muito mais elementar para detectar a semelhança entre dois padrões binários  $\mathbf{a}$  e  $\mathbf{b}$  de dimensão  $H$ , a *distância de Hamming*

$$D_H(\mathbf{a}, \mathbf{b}) \stackrel{\text{def}}{=} \sum_{i=1}^H I(a_i \neq b_i), \quad (6.7)$$

onde  $I(B)$  é a função indicadora da (eq. 8.48) na pág. 254. Aquele padrão  $\mathbf{x}$  que tiver a menor distância de Hamming em relação a todos os padrões armazenados é declarado o mais parecido. No modelo da rede de Hopfield queremos obter esta resposta pelo comportamento dinâmico na fase de treinamento.

A apresentação de um padrão na rede de Hopfield  $\mathbf{x}$  como entrada à rede treinada é atribuir como estado inicial  $\mathbf{x}(\tau = 0) \leftarrow \mathbf{x}$  o padrão. Após a inicialização do padrão, o dinamismo desta rede recorrente produz uma sequência

$$\mathbf{x}(0) \rightarrow \mathbf{x}(1) \rightarrow \dots \rightarrow \mathbf{x}(\tau) \rightarrow \dots \rightarrow \mathbf{x}(\tau_{\max}) \rightarrow \mathbf{x}(\tau_{\max} + 1) = \mathbf{x}(\tau_{\max}).$$

Cada estado subsequente da rede é diferente do estado anterior, ou seja

$$\mathbf{x}(\tau) \neq \mathbf{x}(\tau - 1),$$

exceto no último passo da atualização.

**Definição 6.1.1 — Estado estável da rede de Hopfield.** O estado da rede de Hopfield é estável, se um passo de atualização pela regra (eq. 6.6) não muda mais o estado, ou seja

$$\mathbf{x}(\tau + 1) = \mathbf{x}(\tau). \quad (6.8)$$

Um conceito importante da rede de Hopfield é a energia, dado um estado atual.

**Definição 6.1.2 — Energia da rede de Hopfield.** Considerando o estado atual  $\mathbf{x} = \mathbf{x}(\tau)$  da rede, e todos os pesos da rede (eq. 6.5), a *energia* da rede de Hopfield é

$$\mathcal{H} \stackrel{\text{def}}{=} -\frac{1}{2} \sum_{i=1}^H \sum_{j=1}^H w_{i,j} x_i x_j = -\frac{1}{2} \mathbf{x}^\top W \mathbf{x} \stackrel{1.4}{=} \stackrel{2.59}{=} -\frac{1}{2} \|L^\top \mathbf{x}\|^2, \quad (6.9)$$

onde a matriz  $L$  é a matriz triangular da decomposição de Cholesky. A decomposição da matriz real e simétrica  $W$  pela matriz real triangular inferior  $L$  é possível como

$$W = LL^\top.$$

Temos para cada estado atual da rede  $\mathbf{x}(\tau)$  um valor da energia atual

$$\mathcal{H}(\tau) = -\frac{1}{2} \|L^\top \mathbf{x}(\tau)\|^2. \quad (6.10)$$

A matriz de pesos  $W$ , e consequentemente a matriz triangular  $L$  contêm somente valores finitos. O vetor do estado da rede  $\mathbf{x}(\tau)$  contém valores  $x_i \in \{-1, 1\}$ . Podemos concluir que a norma quadrática  $\|\mathbf{v}(\tau)\|^2$  do vetor

$$\mathbf{v}(\tau) \stackrel{\text{def}}{=} L^\top \mathbf{x}(\tau)$$

é também finita, ou seja,

$$\|\mathbf{v}(\tau)\|^2 < \infty,$$

e finalmente, podemos afirmar que a energia tem um limite inferior, ou seja,

$$\mathcal{H}(\tau) > -\infty. \quad (6.11)$$

Se podemos mostrar que a energia diminui de um passo  $\tau$  para o próximo  $\tau + 1$ , ou fica finalmente constante,

$$\mathcal{H}(\tau + 1) \leq \mathcal{H}(\tau),$$

fica provado que o algoritmo de atualização da rede de Hopfield converge.

Para analisar o valor da energia da rede, vamos primeiro expandir a sua definição da (eq. 6.9). Temos

$$\begin{aligned} \mathcal{H} = & -\frac{1}{2} \{ \\ & w_{1,1}x_1x_1 + w_{1,2}x_1x_2 + \cdots + w_{1,i}x_1x_i + \cdots + w_{1,H}x_1x_H \\ & + w_{2,1}x_2x_1 + w_{2,2}x_2x_2 + \cdots + w_{2,i}x_2x_i + \cdots + w_{2,H}x_2x_H \\ & \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\ & + w_{i,1}x_ix_1 + w_{i,2}x_ix_2 + \cdots + w_{i,i}x_ix_i + \cdots + w_{i,H}x_ix_H \\ & \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\ & + w_{H,1}x_Hx_1 + w_{H,2}x_Hx_2 + \cdots + w_{H,i}x_Hx_i + \cdots + w_{H,H}x_Hx_H \end{aligned} \} . \quad (6.12)$$

Todos os termos, onde o valor do  $i$ -ésimo neurônio  $x_i$  tem uma contribuição à energia foram destacados na (eq. 6.12). Essa contribuição seja denominada  $\mathcal{H}_i$ . Então

$$\begin{aligned} \mathcal{H}_i &\stackrel{\text{def}}{=} -\frac{1}{2} \left[ \sum_{j=1}^H w_{i,j}x_i x_j + \sum_{j=1}^H w_{j,i}x_j x_i - w_{i,i}x_i x_i \right] \\ &\stackrel{\text{§ 5}}{=} -\frac{1}{2} \left[ \sum_{j=1}^H w_{i,j}x_i x_j + \sum_{j=1}^H w_{i,j}x_i x_j - 0x_i^2 \right] \\ &= -\sum_{j=1}^H w_{i,j}x_i x_j. \end{aligned} \quad (6.13)$$

O termo  $w_{i,i}x_i x_i$  tem que ser subtraído uma vez, pois aparece simultaneamente na linha e coluna que envolvem  $x_i$ . De qualquer maneira, não tem influência na soma, pois  $w_{i,i} = 0$ .

Vamos analisar a mudança de energia da rede entre dois instantes de tempo, focando no  $i$ -ésimo neurônio  $x_i$ . Os valores dos restantes neurônios  $x_j$  não mudam com o tempo, por isso não são rotulado como  $x_j(\tau)$ . Então

$$\begin{aligned} \Delta\mathcal{H}_i(\tau) &\stackrel{\text{def}}{=} \mathcal{H}_i(\tau + 1) - \mathcal{H}_i(\tau) \\ &= -\sum_{j=1}^H w_{i,j}x_i(\tau + 1)x_j - -\sum_{j=1}^H w_{i,j}x_i(\tau)x_j \\ &= -[x_i(\tau + 1) - x_i(\tau)] \sum_{j=1}^H w_{i,j}x_j \\ &= \begin{cases} 0, & \text{se } x_i(\tau + 1) = x_i(\tau) \\ 2x_i(\tau) \sum_{j=1}^H w_{i,j}x_j, & \text{se } x_i(\tau + 1) \neq x_i(\tau) \end{cases}. \end{aligned} \quad (6.14)$$

No primeiro caso da (eq. 6.14), não houve mudança de estado do  $i$ -ésimo neurônio do estado atual para o próximo estado, e consequentemente não houve mudança na energia da rede toda, se desconsiderar os restantes neurônios. No segundo caso, houve uma mudança de estado

$$x_i(\tau + 1) = -x_i(\tau) \implies x_i(\tau + 1) \cdot x_i(\tau) = -1, \quad (6.15)$$

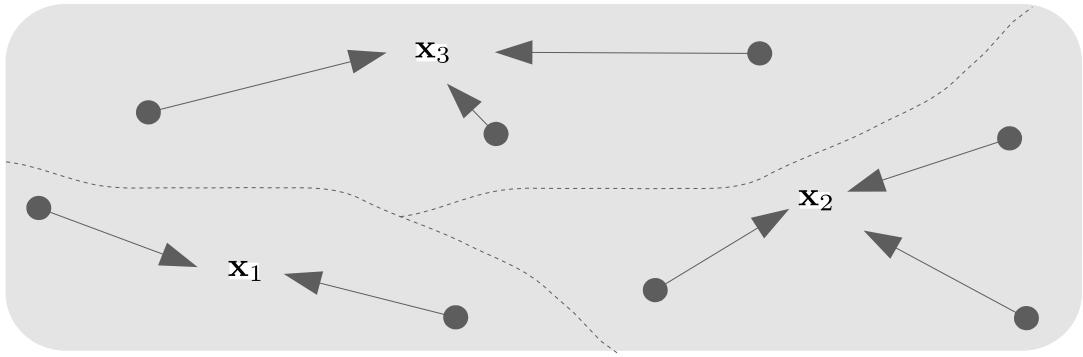


Figura 6.2: Modelo de bacias de atração.

ou seja, se  $x_i$  tinha o valor  $-1$ , mudou para  $+1$ , ou seja  $-(1 - -1) = -2 = 2x_i(\tau)$ , se tinha o valor  $+1$ , mudou para  $-1$ , ou seja, igualmente  $-(-1 - 1) = 2 = 2x_i(\tau)$ .

Estamos unicamente interessados no sinal da diferença  $\Delta\mathcal{H}_i(\tau)$ , se for negativo, fica provado que a energia diminui, se houve mudança de estado do neurônio  $x_i$ . Então, ainda considerando a regra

$$\operatorname{sgn}(a \cdot b) = \operatorname{sgn}(a) \cdot \operatorname{sgn}(b), \quad (6.16)$$

temos

$$\begin{aligned} \operatorname{sgn}(\Delta\mathcal{H}_i(\tau)) &= \operatorname{sgn}\left(2x_i(\tau) \sum_{j=1}^H w_{i,j}x_j\right) \\ &= \operatorname{sgn}(2x_i(\tau)) \cdot \operatorname{sgn}\left(\sum_{j=1}^H w_{i,j}x_j\right) \\ &\stackrel{\text{def}}{=} x_i(\tau) \cdot x_i(\tau+1) \\ &\stackrel{\text{def}}{=} -1 < 0. \end{aligned} \quad (6.17)$$

Podemos concluir que, ou a energia diminui, ou fica constante. Como existe um limite inferior, e, em caso de estabilidade dos estados não haverá mais mudança, a rede *sempre* convergirá para um estado estável  $\mathbf{x}(\tau_{\max})$ , representando a saída da rede, se apresentado com um estado inicial  $\mathbf{x}(0)$ .

### 6.1.3 Modelo de Bacias de Atração

A figura 6.2 idealiza a evolução de um estado inicial  $\mathbf{x}(\tau = 0)$  para um estado estável. Os estados iniciais são representados por círculos que em um ou mais passos convergem para um dos três estados finais possíveis, ou  $\mathbf{x}_1$ , ou  $\mathbf{x}_2$ , ou  $\mathbf{x}_3$ . Esse modelo define *bacias de atração* onde qualquer padrão inicial caminha para o estado final associado à respectiva bacia.

#### Memorização de Padrão Único

Uma análise útil preliminar é o armazenamento de um único padrão  $\mathbf{x}$ . Espera-se da rede na apresentação desse mesmo padrão, ou um padrão parecido que a rede “se lembra” de  $\mathbf{x}$ . Já pode-se antecipar que o processo do ajuste dos pesos da rede de Hopfield, para todos os casos, é determinístico, ou seja, podemos achar uma forma fechada  $w_{i,j} = \dots$  para obter o valor dos pesos.

A resposta mais elementar da rede treinada por um único padrão  $\mathbf{x}$ , na apresentação desse mesmo padrão, deve ser idempotência

$$\mathbf{x} = \mathbf{x}(0) = \mathbf{x}(\tau_{\max}), \quad (6.18)$$

ou seja, pela regra de atualização da (eq. 6.6) todos os neurônios permanecem em um estado estável

$$x_i = \operatorname{sgn} \left( \sum_{j=1}^H w_{i,j} x_j \right), \quad i = 1, \dots, H. \quad (6.19)$$

Seja a regra de treinamento dos pesos entre os neurônios  $x_i$  e  $x_j$ , em caso de memorização de um único padrão

$$w_{i,j} = \frac{1}{H} x_i x_j. \quad (6.20)$$

Independentemente, se o valor de  $x_j$  é 1 ou -1, temos sempre

$$x_j x_j = 1. \quad (6.21)$$

Então, plugando a (eq. 6.20) na (eq. 6.19), temos

$$\begin{aligned} x_i &= \operatorname{sgn} \left( \sum_{j=1}^H \frac{1}{H} x_i x_j x_j \right) \\ &\stackrel{\text{def}}{=} \operatorname{sgn} \left( \sum_{j=1}^H \frac{1}{H} x_i \right) \\ &= \operatorname{sgn} \left( \frac{1}{H} \sum_{j=1}^H x_i \right) \end{aligned} \quad (6.22)$$

$$\begin{aligned} &= \operatorname{sgn} \left( \frac{1}{H} H x_i \right) \\ &= \operatorname{sgn} (x_i) \\ &\stackrel{\text{def}}{=} x_i. \end{aligned} \quad (6.23)$$

Isto prova que a memorização de um único padrão, e a posterior apresentação do mesmo padrão o recupera perfeitamente.

Vamos assumir agora que houve uma corrupção de alguns dos neurônios, digamos de uma quantidade de  $N$  do total de  $H$  neurônios, ou seja, houve para esses  $N$  neurônios<sup>1</sup> uma inversão do seu valor binário, negativo para positivo, ou vice versa

$$x_i \leftarrow -x_i.$$

Então, a partir da (eq. 6.22) temos

$$\begin{aligned} x_i &= \operatorname{sgn} \left( \frac{1}{H} \sum_{j=1}^H x_i \right) \\ &= \operatorname{sgn} \left( \frac{1}{H} \left\{ \sum_{j=1}^{H-N} x_i + \sum_{j=1}^N -x_i \right\} \right) \\ &= \operatorname{sgn} \left( \frac{1}{H} (H - N - N) x_i \right) \\ &\stackrel{\text{def}}{=} \operatorname{sgn} \left( \frac{1}{H} \right) \operatorname{sgn} (H - 2N) \operatorname{sgn} (x_i) \\ &= 1 \cdot \operatorname{sgn} (H - 2N) x_i. \end{aligned} \quad (6.24)$$

---

<sup>1</sup>Teríamos que introduzir mais um nível de índice para não usar a mesma enumeração dos neurônios originais, do tipo  $x_{j_\ell}, \ell = 1, \dots, N$ , mas isso é omitido por razões de simplicidade.

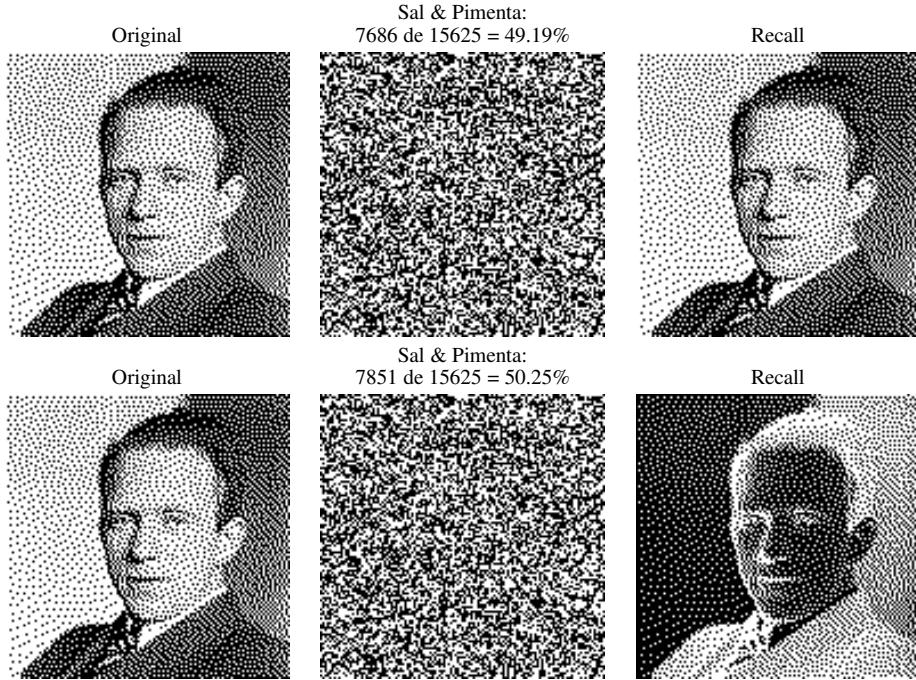


Figura 6.3: Recuperação de padrão memorizado e padrão invertido. A imagem original foi normalizada para o formato  $125 \times 125$ , e oito bits em tons de cinza e depois binarizada Apelo algoritmo de Floyd–Steinberg. Esta única imagem binária de  $H = 125^2 = 15625$  bits é memorizada na rede de Hopfield de  $H$  neurônios.

Temos

$$\text{sgn}(H - 2N) = \begin{cases} 1, & \text{se } N < H/2 \\ -1, & \text{se } N > H/2 \end{cases}.$$

Em outras palavras, se menos que a metade dos valores originais  $x_i$  forem corrompidos, o padrão fica estável. Caso contrário, todos os neurônios mudam de valor, ou seja, o padrão  $\mathbf{x}$  todo é invertido para  $-\mathbf{x}$ .

■ **Exemplo 6.1** A figura 6.3 mostra um experimento de memorização de um padrão pela rede de Hopfield. Somente um único padrão é usado para obter os pesos da rede, pela regra da (eq. 6.20). O processo de linearização de uma imagem de tamanho  $n \times n$  para um vetor  $\mathbf{x}$  de tamanho  $n^2$ , e sua inversão, é o mesmo descrito na seção 6 na pág. 135. A imagem original tem tamanho  $125 \times 125$ , mostrado o físico Werner Heisenberg, “pai” da física quântica. Os valores são binários<sup>2</sup>, assim a imagem é apropriada para ser armazenada pela rede de Hopfield. A conversão da imagem original em tons de cinza para a imagem binária foi feite pelo método de binarização de Floyd–Steinberg [38] (*Floyd–Steinberg dithering*). Na linha de cima, o original foi contaminado por ruido “Sal & Pimenta”, convertendo aleatoriamente, píxeis brancos para píxeis pretos, e vice versa. O nível de ruido fica com 49.19% abaixo de 50%, portanto a bacia de atração é a imagem original. Na linha de baixo, 50.25% dos neurônios foram invertidos, ou seja, mais que a metade dos 15625 píxeis. A consequência é que, na consulta, a imagem contaminada caminha para a bacia de atração da versão negativa da imagem. A consulta termina em ambos os casos após uma única iteração. No primeiro caso, a energia reduz de -1.548 para -7812.0, no segundo caso de 0.310 também para -7812.0. ■

<sup>2</sup>Para enxergar que as imagens binárias realmente são exclusivamente compostas de pontos pretos e brancos, faça uma amplificação na visualização desta imagem.

### Memorização de Conjunto de Padrões

Vamos considerar a memorização de  $n$  padrões<sup>3</sup> binários, cada um de dimensão  $H$ .

$$\mathcal{D} = \left\{ \mathbf{x}^{(1)}, \dots, \mathbf{x}^{(k)}, \dots, \mathbf{x}^{(n)} \right\}. \quad (6.25)$$

A regra determinística de aprendizagem do peso entre neurônio  $x_i$  e  $x_j$ , considerando a regra para um padrão na (eq. 6.20) é modificada para

$$w_{i,j} = \frac{1}{H} \sum_{k=1}^n x_i^{(k)} x_j^{(k)}. \quad (6.26)$$

Esta regra condiz vagamente com a hipótese da aprendizagem de neurônios biológicos, estabelecida por Hebb [46, 47] que reforça uma ligação entre neurônios (seus pesos), se houver uma atividade simultânea entre eles (modelado pelo valor 1)<sup>4</sup>. Note que o treinamento da rede, mesmo com um conjunto maior de padrões não precisa de um processo iterativo de aprendizagem. Basta uma forma fechada, como a (eq. 6.26) para ajustar os parâmetros livres do modelo.

**Definição 6.1.3 — Modelo de Hopfield.** Uma rede neural com neurônios binários  $x_i \in \{-1, 1\}$  que usa a regra de Hebb (eq. 6.26) para o treinamento dos pesos  $w_{ij}$ , e que usa a atualização assíncrona do algoritmo 12 para diminuir a energia definição 6.1.2 até convergir, na consulta após a apresentação de um padrão  $\mathbf{x}$  é um modelo de Hopfield.

#### 6.1.4 Capacidade de Armazenamento

Certamente, a quantidade máxima de padrões que possam ser aprendidos e recuperados corretamente, se apresentados como estado inicial, é limitada. Não é plausível que uma rede com poucas unidades binárias seja capaz de memorizar muitos vetores. Tem que existir uma relação entre a quantidade  $n$  de padrões na memória e a quantidade  $H$  de neurônios. Em [48] um estudo sobre a capacidade é apresentado, começando com o caso de padrões completamente arbitrários.

Vamos considerar a memorização de  $n$  padrões  $\mathbf{x}^{(k)}$ ,  $k = 1, \dots, n$  pela rede. Para o  $k$ -ésimo padrão  $\mathbf{x}^{(k)}$  ser estável na fase de consulta, em todos os seus  $H$  componentes binários  $x_i^{(k)}$ ,  $i = 1, \dots, H$ , pela condição de idempotência da (eq. 6.19) temos

$$x_i^{(k)} = \text{sgn} \left( \mathbf{w}_{i,\bullet} \cdot \mathbf{x}^{(k)} \right), \quad i = 1, \dots, H. \quad (6.27)$$

Definindo uma “ativação” neste contexto

$$a(\mathbf{w}; \mathbf{x}) \stackrel{\text{def}}{=} \mathbf{w} \cdot \mathbf{x}, \quad (6.28)$$

---

<sup>3</sup>Aqui usaremos uma nomenclatura ligeiramente diferente. O  $k$ -ésimo padrão é chamado  $\mathbf{x}^{(k)}$ , e o seu  $i$ -ésimo valor é  $x_i^{(k)}$ , enquanto  $x_i$  é o valor do  $i$ -ésimo neurônio da rede.

<sup>4</sup>O livro de Hebb situa-se na área de biologia, psicologia e medicina. Não contém expressões matemáticas para modelar a teoria.

podemos constatar que

$$\begin{aligned}
a_i^{(k)} &= \mathbf{w}_{i,\bullet} \cdot \mathbf{x}^{(k)} \\
&= \sum_{j=1}^H w_{i,j} x_j^{(k)} \\
&\stackrel{\text{def}}{=} \sum_{j=1}^H \left( \frac{1}{H} \sum_{\ell=1}^n x_i^{(\ell)} x_j^{(\ell)} \right) x_j^{(k)} \\
&= \sum_{j=1}^H \frac{1}{H} \left[ \sum_{\substack{\ell=1 \\ \ell \neq k}}^n x_i^{(\ell)} x_j^{(\ell)} x_j^{(k)} + \overbrace{x_i^{(k)} x_j^{(k)} x_j^{(k)}}^{\ell=k} \right] \\
&= \sum_{j=1}^H \left[ \frac{1}{H} \sum_{\substack{\ell=1 \\ \ell \neq k}}^n x_i^{(\ell)} x_j^{(\ell)} x_j^{(k)} + \frac{1}{H} x_i^{(k)} \right] \\
&= \left( \frac{1}{H} \sum_{j=1}^H \sum_{\substack{\ell=1 \\ \ell \neq k}}^n x_i^{(\ell)} x_j^{(\ell)} x_j^{(k)} \right) + H \frac{1}{H} x_i^{(k)} \\
&= x_i^{(k)} + C_i^{(k)},
\end{aligned} \tag{6.29}$$

onde o termo de diafonia (*crosstalk*) foi definido como

$$C_i^{(k)} \stackrel{\text{def}}{=} \frac{1}{H} \sum_{j=1}^H \sum_{\substack{\ell=1 \\ \ell \neq k}}^n x_i^{(\ell)} x_j^{(\ell)} x_j^{(k)}. \tag{6.30}$$

Temos então pela (eq. 6.27) estabilidade, se

$$\begin{aligned}
x_i^{(k)} &= \operatorname{sgn} a_i^{(k)} \\
&\stackrel{\text{def}}{=} \operatorname{sgn} (x_i^{(k)} + C_i^{(k)}) \\
&= \begin{cases} x_i^{(k)}, & \text{se } |C_i^{(k)}| < 1 \\ -x_i^{(k)}, & \text{caso contrário.} \end{cases}
\end{aligned} \tag{6.31}$$

Se o termo de diafonia da (eq. 6.30) for pequeno para todos os  $i = 1, \dots, H$  neurônios e todos os  $k = 1, \dots, n$  padrões, podemos esperar estabilidade para os  $n$  padrões memorizados, se apresentados na consulta.

Assumindo arbitrariedade nos valores binários dos padrões, e uma distribuição normal dos erros pela instabilidade de um bit na consulta, pode-se estabelecer uma heurística

$$n_{\max} \leq 0.138H \tag{6.32}$$

para o número máximo  $n_{\max}$  de padrões que possam ser armazenados em uma rede com  $H$  neurônios, veja [48] para uma explicação mais detalhada.

### 6.1.5 Estados Espúrios

Além das bacias de atração relacionadas aos  $n$  padrões  $\mathbf{x}^{(k)}$ ,  $k = 1, \dots, n$  memorizados existentes, existem bacias de atração que nem são padrões existentes  $\mathbf{x}^{(k)}$ ,  $k \in n$ , nem os seus inversos  $-\mathbf{x}^{(k)}$ ,

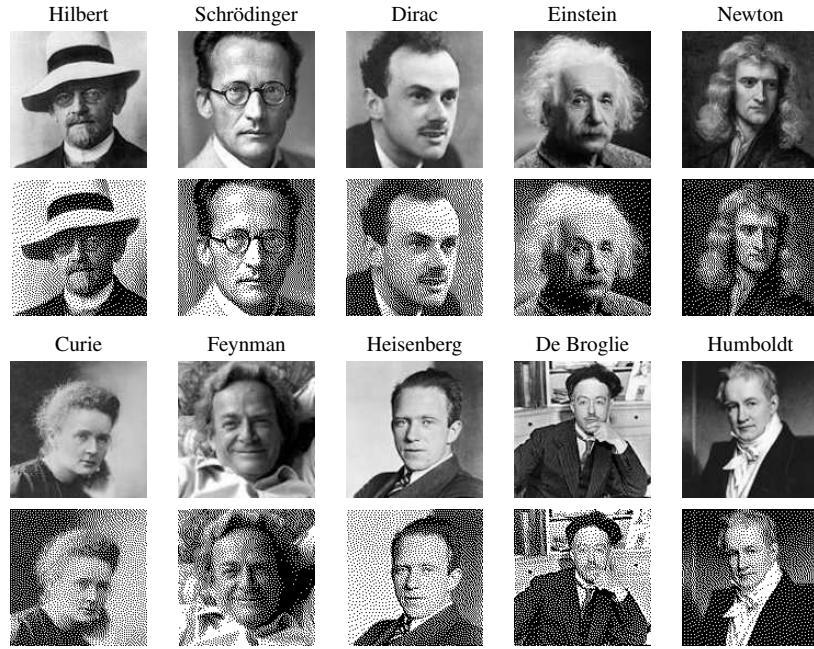


Figura 6.4: Dez físicos famosos. As imagens originais foram normalizadas para o formato  $125 \times 125$ , e oito bits em tons de cinza. Abaixo de cada imagem mostra-se a versão binarizada pelo algoritmo de Floyd–Steinberg. As dez imagens binárias de  $H = 125^2 = 15625$  bits são memorizadas na rede de Hopfield de  $H$  neurônios.

$k \in n$ . Segundo [2], existem estados estáveis da rede que são combinações lineares de (um número ímpar) de padrões armazenados, ou seja,

$$\mathbf{x}^{\text{mix}} \stackrel{\text{def}}{=} \sum_{k \in n} w_k \mathbf{x}^{(k)}, \quad w_k \in \mathbb{R}. \quad (6.33)$$

■ **Exemplo 6.2** Na figura 6.4, mostram-se  $n = 10$  imagens de físicos famosos e as versões das imagens binarizadas de  $H = 125^2 = 15625$  píxeis, respectivamente. Os píxeis linearizados em vetores, constituem os  $n$  padrões que são submetidos à rede para aprender a matriz de pesos pela regra da (eq. 6.26). A quantidade de pesos “úteis”, por causa da simetria da matriz, e da diagonal nula, é  $H/2(H - 1) = 7875$ . A figura 6.5 ilustra experimentos de recuperação, submetendo padrões que não são os padrões memorizados. São quatro experimentos. A evolução do valor da energia definição 6.1.2 pode ser acompanhado na figura 6.6. No primeiro experimento, a imagem original é uma combinação linear de um terço da imagem dos primeiros três físicos “1/3(Hilbert+Schrödinger+Dirac)”. Após cinco iterações, o estado da rede mostra uma convergência para a imagem de Hilbert, o padrão  $\mathbf{x}^{(1)}$  existente no conjunto de treinamento. No segundo experimento a imagem “1/3(Einstein+Newton+Curie)”, após  $\tau_{\max} = 8$  iterações converge para um padrão  $\mathbf{x}^{\text{mix}}$  que não faz parte dos físicos memorizados, compare com as imagens binarizadas da figura 6.4. Parece que a primeira transição já está muito perto do resultado final, visível também nos valores da energia muito próximos após a primeira transição  $\tau_0 \rightarrow \tau_1$ . O resultado parece estar sendo dominado pela imagem de Newton, porém não é idêntica. Em analogia, “1/3(Feynman+Heisenberg+De Broglie)” converge após quatro iterações para a imagem existente de Heisenberg. Finalmente no último experimento, a imagem inicial contém um décimo de cada imagem. Após quatro iterações converge para um resultado muito parecido, mas não igual ao resultado do segundo experimento. Faça uma amplificação do bitmap e compare. ■

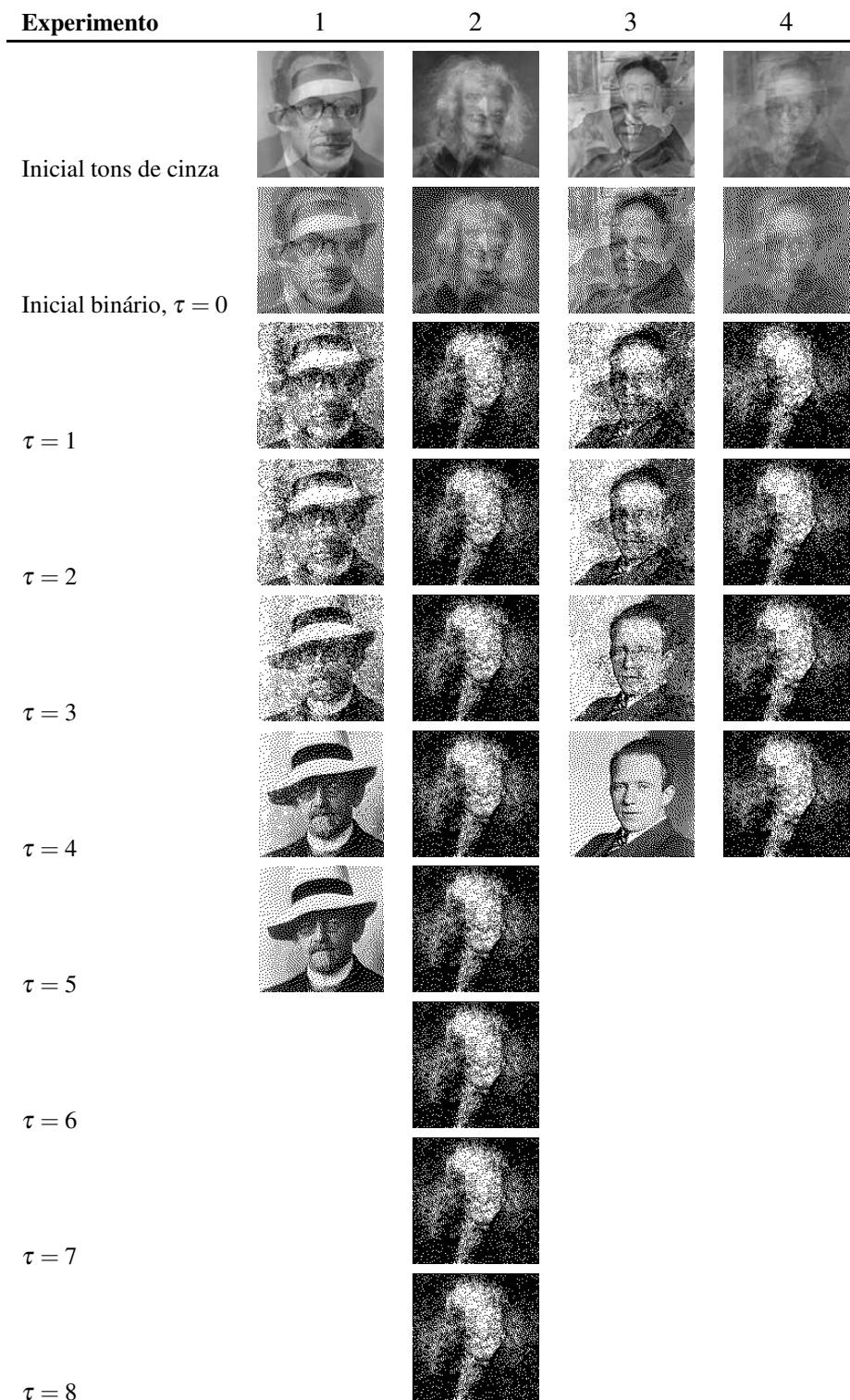


Figura 6.5: Experimentos de recuperação de estados espúrios. Uma combinação linear de padrões existentes é submetida à rede, e um padrão estável, parcialmente inexistente no conjunto de padrões memorizados é recuperado.

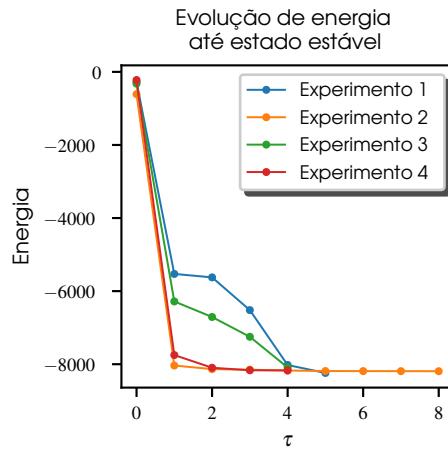


Figura 6.6: Evolução do valor de energia dos experimentos descritos no exemplo 6.2.

Podemos concluir que o modelo de Hopfield é muito interessante em relação à ideia de providenciar uma memória associativa que armazena padrões binários e permite a recuperação dos mesmos, até com perturbações dos padrões apresentados. Porém, os resultados dos experimentos não permitem associar uma certa “inteligência” a este modelo. A natureza do cálculo linear no treinamento e do cálculo linear seguido por uma função de ativação não linear, neste caso a função de sinal, definem limites da capacidade da rede de Hopfield. No entanto, do ponto de vista acadêmico, é uma rede neural artificial que se destaca das arquiteturas com fluxo de informação adiante, como, por exemplo, o Perceptron Multicamada. Na próxima seção apresenta-se um modelo mais sofisticado que também usa o conceito da consulta dinâmica e de uma função de energia.

## 6.2 Máquina Restrita de Boltzmann *Restricted Boltzmann Machine*, RBM

### 6.2.1 Modelo RBM

#### Modelo RBM

Uma das peças de mosaico para entender o modelo RBM é a amostragem de uma variável aleatória discreta que tenha uma Distribuição de Bernoulli. O domínio dos valores da variável tem somente dois valores possíveis, ou seja

$$x \in \{0, 1\},$$

“sucesso”, e “falha”. Basta um único parâmetro  $P$  para definir uma variável aleatória  $x$  que segue esta distribuição

$$x \sim \text{Bernoulli}(P), \quad (6.34)$$

onde  $P \in [0, 1]$  é a probabilidade de sucesso. Uma moeda tem  $P = \frac{1}{2}$ , onde os dois lados representam os dois valores possíveis da variável. Uma sequência de várias *amostragens* com este valor de  $P = 0.5$  produz valores, como, por exemplo,  $1, 0, 0, 1, 0, 1, 1, 1, 0, 0, \dots$ . Valores de  $P$  mais próximos de zero produzem mais valores zero que valores um, se feito repetidamente. Uma única amostra é feita da seguinte maneira:

1. Gere um número aleatório  $r$  que segue uma distribuição uniforme com valores entre zero e um, ou seja,

$$r \sim U(0, 1)$$

Reveja a distribuição uniforme da definição 1.2.1 na pág. 17.

2. Atribua o valor binário (zero ou um) à variável aleatória, simulando um único experimento

$$x \leftarrow \begin{cases} 1 & \text{se } r < p \\ 0 & \text{se } r \geq p. \end{cases} \quad (6.35)$$

A RBM é composta por vários neurônios binários, cada um representada por uma variável aleatória com distribuição de Bernoulli. Esta técnica de amostragem vai servir como base na obtenção de valores dos neurônios pelo modelo da RBM.

## 7. Classificação Paramétrica

### 7.1 Regra de Bayes

Focamos novamente em variáveis aleatórias para criar classificadores. Desta vez a distribuição dos dados é conhecida, ou, pelo menos assumida de originar de alguma distribuição. O exemplo melhor estudado é a Gaussiana, releia a definição da (eq. 1.18) na pág. 18 e figura 1.3 para uma dimensão, e a (eq. 1.47) na pág. 28 e figura 1.4 e figura 1.5 para mais que uma dimensão. Para calcular o *score* de uma classe para a tomada de decisões, precisamos ainda uma ferramenta importante.

No fundo, a regra de Bayes é baseado em *distribuição de probabilidade (e/ou densidade de probabilidade) conjunta*, escrita de duas maneiras diferentes. Vamos precisar de trabalhar com variáveis aleatórias discretas e contínuas, unidimensionais e multidimensionais. Vamos trabalhar com probabilidade, densidades de probabilidade, conjuntas e condicionadas, usando somente os axiomas e regras necessárias da teoria de probabilidade

**Definição 7.1.1 — Regra de Bayes.** Lembre que uma variável aleatória discreta  $x$  tem uma probabilidade  $0 \leq P(x) \leq 1$  e uma variável aleatória contínua  $x$  tem uma densidade de probabilidade  $0 \leq p(x)$ . Usamos o “P” maiúsculo para probabilidades e o “p” minúsculo para densidades. Uma Probabilidade/Densidade Conjunta descreve o comportamento simultâneo de duas variáveis aleatórias. Probabilidade/Densidade Condicionada fixa uma variável e descreve o comportamento da outra variável sob esta restrição. Podemos ter quatro combinações possíveis para as definições.

1. Duas variáveis aleatórias discretas
2. A primeira variável aleatória discreta e a segunda variável aleatória contínua
3. A primeira variável aleatória contínua e a segunda variável aleatória discreta
4. Duas variáveis aleatórias contínuas

O caso que os interessa mais é o segundo, em que a variável aleatória discreta é o conjunto de classes, e a variável aleatória contínua (unidimensional ou multidimensional) é o padrão. Um axioma da teoria de probabilidades consta que a densidade conjunta pode ser decomposta de

duas maneiras diferentes como

$$p(\mathcal{C}, x) = P(\mathcal{C}|x) \cdot p(x) = p(x|\mathcal{C}) \cdot P(\mathcal{C}), \quad (7.1)$$

em que temos as seguintes variáveis

$$P(\mathcal{C}|x) \quad \text{Probabilidade } a \text{ posteriori de } x \text{ pertencer a classe } \mathcal{C} \quad (7.2a)$$

$$p(x) \quad \text{Evidência: } x \text{ tem essa densidade, independente das classes} \quad (7.2b)$$

$$p(x|\mathcal{C}) \quad \text{Densidade condicional, dada uma classe particular } \mathcal{C} \quad (7.2c)$$

$$P(\mathcal{C}) \quad \text{Probabilidade } a \text{ priori da classe } \mathcal{C}, \text{ desconhecendo qualquer densidade} \quad (7.2d)$$

A evidência (eq. 7.2b) é uma densidade *marginal*, em que para todos os valores possíveis da outra variável aleatória o resultado é integrado (caso a variável marginalizada é contínua), ou somada (caso a variável marginalizada é discreta). No nosso caso, temos  $c$  classes  $\mathcal{C}_1, \dots, \mathcal{C}_c$ . Então, a evidência, marginalizada a variável aleatória das classes, temos

$$p(x) = \sum_{k=1}^c p(x|\mathcal{C}_k) \cdot P(\mathcal{C}_k). \quad (7.3)$$

Resolvendo a (eq. 7.1), considerando uma classe específica  $\mathcal{C}_k$ , temos a regra de Bayes

$$P(\mathcal{C}_k|x) = \frac{p(x|\mathcal{C}_k) \cdot P(\mathcal{C}_k)}{p(x)} = \frac{p(x|\mathcal{C}_k) \cdot P(\mathcal{C}_k)}{\sum_{k=1}^c p(x|\mathcal{C}_k) \cdot P(\mathcal{C}_k)}. \quad (7.4)$$

Essa regra em termos coloquiais pode ser expressa equivalentemente como

$$\text{Probabilidade a posteriori} = \frac{\text{verossimilhança} \cdot \text{Probabilidade a priori}}{\text{Evidência}} \quad (7.5)$$

■ **Exemplo 7.1** “O viajante anglófono”. Vamos considerar um problema que envolve duas variáveis aleatórias discretas, em que cada uma das variáveis pode ter dois valores diferentes que têm probabilidades diferentes. Temos as seguinte situação:

- Uma variável aleatória discreta  $x$  “origem” com dois valores possíveis do seu domínio  $\mathcal{X}$ , ou seja,  $x \in \mathcal{X} = \{\text{nativo}, \text{turista}\}$
- Uma variável aleatória discreta  $y$  “anglófono” com dois valores possíveis do seu domínio  $\mathcal{Y}$ , ou seja,  $y \in \mathcal{Y} = \{\text{sim}, \text{não}\}$
- 10 % dos nativos falam inglês
- 20 % das pessoas são turistas
- 50 % dos turistas falam inglês

A tarefa é calcular a probabilidade a posteriori de uma pessoa ser nativa, dado o fato que ela fala inglês, usando a regra de Bayes.

Temos que traduzir a especificação verbal em expressões apropriadas. Podemos identificar a expressão “dado o fato que” como condição de um expressão de probabilidade condicional, ou seja, é o lado direito do símbolo “|”. Neste caso é  $y = \text{sim}$ . O símbolo “|” não é por acaso parecido com o símbolo “/” da divisão, pois representa uma divisão da probabilidade pelos casos que exclusivamente são especificados pela restrição embutida em “dado o fato que”. Então temos que calcular

$$P(x = \text{nativo} | y = \text{sim}).$$

Usando a regra de Bayes na versão com duas variáveis aleatórias discretas, queremos

$$P(x = \text{nativo}|y = \text{sim}) = \frac{P(y = \text{sim}|x = \text{nativo})P(x = \text{nativo})}{P(y = \text{sim})}.$$

O denominador  $P(y = \text{sim})$  representa realmente uma evidência, nomeadamente que se ouviu uma pessoa falar inglês, independentemente do fato de ser turista ou nativo. Podemos identificar esse “ou” como uma marginalização sobre a variável  $x$  do origem da pessoa, ou seja

$$\begin{aligned} P(y = \text{sim}) &= \sum_{x \in \mathcal{X}} P(y = \text{sim}|x)P(x) \\ &= P(y = \text{sim}|x = \text{nativo})P(x = \text{nativo}) + P(y = \text{sim}|x = \text{turista})P(x = \text{turista}). \end{aligned}$$

Nessa expressão conhecemos todos os termos.  $P(y = \text{sim}|x = \text{nativo})$  é a probabilidade condicional de uma pessoa saber falar inglês, dado o fato de ela ser um nativo, especificada com o valor 10 %. Sendo o complemento de uma pessoa não ser turista, temos  $P(x = \text{nativo}) = 100\% - 20\% = 80\%$ . Então, a evidência de alguém falar inglês é

$$P(y = \text{sim}) = 0.1 \cdot 0.8 + 0.5 \cdot 0.2 = 0.18.$$

Então, finalmente temos

$$P(x = \text{nativo}|y = \text{sim}) = \frac{0.1 \cdot 0.8}{0.18} = \frac{4}{9} \approx 44.4\%.$$

Um fato notável é a quase igualdade da probabilidade a posteriori de uma pessoa ser nativa quando fala inglês com a probabilidade a posteriori de uma pessoa ser turista quando fala inglês, embora a probabilidade condicional é somente 10 %. Isso se deve ao fato do grande desequilíbrio da probabilidade a priori, ou seja tem muito menos turistas do que nativos. ■

Neste ponto convém introduzir uma maneira de estimar a probabilidade a priori de um padrão pertencer a uma classe, sem conhecimento algum sobre o valor das características.

**Definição 7.1.2 — Probabilidade A Priori Estimada de uma Classe, dado Conjunto de Treinamento.** Seja o conjunto de treinamento composto por  $n$  padrões. Temos  $c$  classes  $\mathcal{C}_k, k = 1, \dots, c$  e cada classe possui  $0 \leq n_k \leq n$  padrões que em soma correspondem à quantidade total, ou seja

$$n = \sum_{k=1}^c n_k. \quad (7.6)$$

Então a probabilidade a priori que um padrão pertence à classe  $\mathcal{C}_k$  é estimado simplesmente pela fração dos padrões de treinamento que pertencem à classe  $\mathcal{C}_k$  relativa à quantidade total  $n$  dos padrões de treinamento, ou seja,

$$\hat{P}(\mathcal{C}_k) = \frac{n_k}{n}. \quad (7.7)$$

■ **Exemplo 7.2** Um exemplo mais próximo da realidade da classificação, dado um vetor de características envolve outra vez as nossas flores. Para facilitar, vamos restringir a quantidade de características apenas para uma, a largura da pétala  $x_4$ . Desta vez, na definição 7.1.1, temos uma variável aleatória discreta (a classe), e uma variável contínua (a característica). Temos que estimar a probabilidade a posteriori que uma flor pertence a uma classe  $\mathcal{C}_k$ , dado o valor  $x$  da característica, ou seja,  $P(\mathcal{C}_k|x)$ . Temos três classes, então queremos saber a probabilidade a posteriori de cada

uma delas, ou seja,  $P(\text{setosa}|x)$ ,  $P(\text{versicolor}|x)$  e  $P(\text{virginica}|x)$ . Pela regra de Bayes (eq. 7.4) temos para  $k = 1, 2, 3$ ,

$$\begin{aligned} P(\mathcal{C}_k|x) &= \frac{p(x|\mathcal{C}_k) \cdot P(\mathcal{C}_k)}{p(x)} = \frac{p(x|\mathcal{C}_k) \cdot P(\mathcal{C}_k)}{\sum_{k=1}^c p(x|\mathcal{C}_k) \cdot P(\mathcal{C}_k)} \\ &= \frac{p(x|\mathcal{C}_k) \cdot P(\mathcal{C}_k)}{p(x|\text{setosa})P(\text{setosa}) + p(x|\text{versicolor})P(\text{versicolor}) + p(x|\text{virginica})P(\text{virginica})}. \end{aligned}$$

Neste ponto barramos em um problema essencial de classificação paramétrica. Qual é a densidade  $p$  que determina o comportamento dos valores  $x$  da característica? Ela tem a mesma forma funcional para todas as classes? Qual é a probabilidade a priori  $P(\mathcal{C}_k)$  de cada classe? A quantidade de padrões de treinamento é suficiente para obter uma estimativa fiel (sem muito viés) dos valores dos parâmetros? Assumindo uma distribuição das variáveis aleatórias que não corresponde à realidade, introduz um *erro de modelo*. Automaticamente, o desempenho de um sistema de aprendizado de máquina degradará com essa pressuposição falsa. Tendo poucos padrões de treinamento, introduz outro erro, o viés de um ou mais parâmetros. Isso são defeitos problemáticos de usar um classificador paramétrico, que podem resultar em um desempenho muito abaixo do que se pode atingir por um modelo de classificador alternativo.

Para tornar viável um classificador paramétrico, em ausência de qualquer informação da densidade das características, vamos adotar a seguinte estratégia:

1. Assume-se uma densidade Gaussiana da definição 1.2.2 na pág. 18 para cada classe. Temos que introduzir um índice ‘ $k$ ’ para cada parâmetro de cada classe  $\mathcal{C}_k$ , ou seja

$$p(x; \mu_k, \sigma_k^2) = \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left\{-\frac{(x - \mu_k)^2}{2\sigma_k^2}\right\}, \quad k = 1, 2, 3 \quad (7.8)$$

2. Usam-se a estimativa da probabilidade a priori  $\hat{P}(\mathcal{C}_k)$  da definição 7.1.2, da média  $\hat{\mu}_k$  da definição 1.3.1 na pág. 22 e da variância  $\hat{\sigma}_k^2$  da definição 1.3.2 para obter os parâmetros necessários para poder usar a regra de Bayes.

A estimativa da probabilidade a priori das três classes iris, em cada uma tem 50 exemplos, simplesmente resulta em

$$\hat{P}(\mathcal{C}_k) = \frac{n_k}{n} = \frac{50}{150} = \frac{1}{3} \approx 33.3\%, \quad k = 1, 2, 3.$$

As estimativas da médias podemos reciclar de (eq. 1.34) na pág. 24, escolhendo a característica correspondente na posição do vetor das médias. Como temos três classes diferentes, teremos três matrizes de covariância diferentes da definição 1.4.10 na pág. 27, ou seja,

$$\hat{\Sigma}_k = \frac{1}{n_k - 1} \sum_{i=1}^{n_k} (\mathbf{x}_{i,k} - \boldsymbol{\mu}_k)(\mathbf{x}_{i,k} - \boldsymbol{\mu}_k)^T = \frac{1}{49} \sum_{i=1}^{50} (\mathbf{x}_{i,k} - \boldsymbol{\mu}_k)(\mathbf{x}_{i,k} - \boldsymbol{\mu}_k)^T,$$

onde  $\mathbf{x}_{i,k}$  é a  $i$ -ésima flor da  $k$ -ésima classe e  $\boldsymbol{\mu}_k$  é a média local de todas as 50 flores da respectiva classe. Usando todas as quatro características, temos como vetores de médias e matrizes de covariância

$$\hat{\boldsymbol{\mu}}_1 = [5.006 \quad 3.428 \quad 1.462 \quad 0.246]^T, \hat{\boldsymbol{\mu}}_2 = [5.936 \quad 2.770 \quad 4.260 \quad 1.326]^T, \hat{\boldsymbol{\mu}}_3 = [6.588 \quad 2.974 \quad 5.552 \quad 2.026]^T,$$

$$\hat{\Sigma}_1 = \begin{bmatrix} 0.124 & 0.099 & 0.016 & 0.01 \\ 0.099 & 0.144 & 0.012 & 0.009 \\ 0.016 & 0.012 & 0.03 & 0.006 \\ 0.01 & 0.009 & 0.006 & 0.011 \end{bmatrix}, \hat{\Sigma}_2 = \begin{bmatrix} 0.266 & 0.085 & 0.183 & 0.056 \\ 0.085 & 0.098 & 0.083 & 0.041 \\ 0.183 & 0.083 & 0.221 & 0.073 \\ 0.056 & 0.041 & 0.073 & 0.039 \end{bmatrix}, \hat{\Sigma}_3 = \begin{bmatrix} 0.404 & 0.094 & 0.303 & 0.049 \\ 0.094 & 0.104 & 0.071 & 0.048 \\ 0.303 & 0.071 & 0.305 & 0.049 \\ 0.049 & 0.048 & 0.049 & 0.075 \end{bmatrix}$$

Como já mencionado, usamos somente a largura da pétala  $x_4$  como característica. Temos então como média a quarta componente dos vetores de média da (eq. 1.34) na pág. 24, e da posição (4,4) das matrizes de covariância as variâncias

$$\hat{\mu}_{\text{setosa},4} = 0.246, \quad \hat{\mu}_{\text{versicolor},4} = 1.326, \quad \hat{\mu}_{\text{virginica},4} = 2.026,$$

$$\hat{\sigma}_{\text{setosa},4}^2 = 0.011, \quad \hat{\sigma}_{\text{versicolor},4}^2 = 0.039, \quad \hat{\sigma}_{\text{virginica},4}^2 = 0.075,$$

A figura 7.1a mostra para a característica  $x_4$ , as três Gaussianas como densidades condicionais  $p(x|\mathcal{C}_k)$  (eq. 7.8) baseadas nas estimativas das médias e variâncias. Além disso, para cada classe, mostra-se essa densidade ponderada pela probabilidade a priori de cada classe, ou seja,  $p(x|\mathcal{C}_k) \cdot P(\mathcal{C}_k)$ . Esse produto é o numerador na regra de Bayes (eq. 7.4). Neste exemplo, cada probabilidade a priori é 1/3, portanto a função  $p(x|\mathcal{C}_k) \cdot P(\mathcal{C}_k)$  tem exatamente 1/3 da altura da Gaussiana  $p(x|\mathcal{C}_k)$ . A soma dos três produtos constitui a evidência  $p(x)$ , o denominador na regra de Bayes (eq. 7.4). De certa maneira, a evidência  $p(x)$  expressa quanto plausível é a ocorrência de um certo valor da característica em um certo ponto do domínio. Por exemplo, no valor  $x = 0.64$ , entre a classe setosa e versicolor pode-se esperar que praticamente nunca aparecerá uma flor iris com a largura da pétala com esse valor.

A figura 7.1b repete a ilustração das três Gaussianas em outra escala. Adicionalmente, a probabilidade a posteriori  $P(\mathcal{C}_k|x)$  de cada classe é mostrada, ou seja, o cálculo da regra de Bayes (eq. 7.4). Repare que o valor se mantém em torno do seu máximo  $P(\mathcal{C}_k|x) = 1$  nas regiões onde a respectiva Gaussiana domina. Quando o valor  $x$  da característica começa a migrar para a área da Gaussiana de uma outra classe, a probabilidade da classe atual cai, enquanto a da outra classe começa a subir. No caso de igualdade, ou seja, se  $P(\mathcal{C}_k|x) = P(\mathcal{C}_\ell|x)$ , definimos uma fronteira  $\mathcal{F}_{k,\ell}$ . Se a maior probabilidade a posteriori for usada como *score* no sentido de uma função discriminativa da definição 2.3.3, definiria as regiões de decisão (definição 2.3.4)  $\mathcal{R}_k$  das classes. Neste caso específico, a probabilidade de uma das três classes nas regiões das outras duas existe, mas o valor é tão ínfimo que podemos claramente distinguir as regiões uma da outra. Essa situação seria muito diferente, se as três Gaussianas tivessem uma sobreposição considerável. Repare que as fronteiras entre as regiões coincidem com os valores  $x$ , onde uma densidade  $p(x|\mathcal{C}_k)$  é igual a da classe vizinha  $p(x|\mathcal{C}_\ell)$ . Isso é verdade somente para o caso de as duas probabilidades a priori serem iguais, ou seja,  $P(\mathcal{C}_k) = P(\mathcal{C}_\ell)$ .

A figura 7.1c ilustra a influência da probabilidade a priori de cada classe. Em um cenário fictício assumimos que a versicolor, com 80% seja muito mais frequente que as outras duas espécies, em que setosa e virginica têm 10% de probabilidade a priori. As curvas e linhas pontilhadas mostrar a situação anterior, em que cada classe tinha um terço de probabilidade a priori. As curvas e linhas sólidas mostrar a situação modificada. As Gaussianas de setosa e virginica encolhem, enquanto a de versicolor cresce. As fronteiras de versicolor se deslocaram para dentro da flor “concorrente”, para esquerda em direção a setosa e para direita em direção de virginica. Fica claro que uma probabilidade a priori alta favorece sua classe, embora a densidade  $p(x|\mathcal{C})$  seja estável. Isso acontece por que o denominador na regra de Bayes é igual para todas as classes, mas o numerador é o produto da probabilidade a priori e da densidade condicional. Vamos considerar uma largura da pétala específica de  $x = 1.5$  e observar a mudança de valores para os dois casos de probabilidades iguais e diferentes acima. Já podemos descartar os valores para setosa, pois a densidade condicional para esse valor é numericamente nulo, ou seja,  $p(1.5|\text{setosa}) = 0$ . Consequentemente o numerador da regra de Bayes também é nulo, bem como a probabilidade a posteriori. A evidência em (eq. 7.3) é

$$p(1.5) = p(1.5|\text{setosa}) \cdot P(\text{setosa}) + p(1.5|\text{versicolor}) \cdot P(\text{versicolor}) + p(1.5|\text{virginica}) \cdot P(\text{virginica}).$$

Para probabilidades a priori iguais de 1/3 resulta em  $p(1.5) = 0.534$ , para  $P(\text{setosa}) = 10\%$ ,

$P(\text{versicolor}) = 80\%$  e  $P(\text{virginica}) = 10\%$  resulta em  $p(1.5) = 1.119$ . As probabilidades a posteriori são para probabilidades a priori equilibradas de  $1/3$

$$\begin{aligned}P(\text{setosa}|1.5) &= p(1.5|\text{setosa}) \cdot P(\text{setosa})/p(1.5) = 0.0 \cdot 0.33/0.534 = 0.0\%, \\P(\text{versicolor}|1.5) &= p(1.5|\text{versicolor}) \cdot P(\text{versicolor})/p(1.5) = 1.370 \cdot 0.33/0.534 = 85.51\%, \\P(\text{virginica}|1.5) &= p(1.5|\text{virginica}) \cdot P(\text{virginica})/p(1.5) = 0.232 \cdot 0.33/0.534 = 14.49\%. \end{aligned}$$

Para as probabilidades desiguais, a estimativa da probabilidade a posteriori resulta em

$$\begin{aligned}P(\text{setosa}|1.5) &= p(1.5|\text{setosa}) \cdot P(\text{setosa})/p(1.5) = 0.0 \cdot 0.1/1.119 = 0.0\%, \\P(\text{versicolor}|1.5) &= p(1.5|\text{versicolor}) \cdot P(\text{versicolor})/p(1.5) = 1.370 \cdot 0.8/1.119 = 97.93\%, \\P(\text{virginica}|1.5) &= p(1.5|\text{virginica}) \cdot P(\text{virginica})/p(1.5) = 0.232 \cdot 0.1/1.119 = 2.07\%. \end{aligned}$$

■

## 7.2 Análise Discriminativa Quadrática

Vamos construir um classificador que assume que as densidades das características sejam de natureza Gaussiana. Lembre que o cálculo de um *score* e a tomada de decisão qual classe temos, são dois assuntos distintos. Utilizando os conceitos definidos na seção 2.3.1 na pág. 59 no contexto de uma classificação linear, vamos elaborar três tipos de classificadores, ou seja, três funções discriminativas. O primeiro usa a probabilidade a posteriori como *score*, a maneira mais óbvia de tomar uma decisão. O segundo ignora a probabilidade a priori e o terceiro introduz um novo conceito, a minimização de risco.

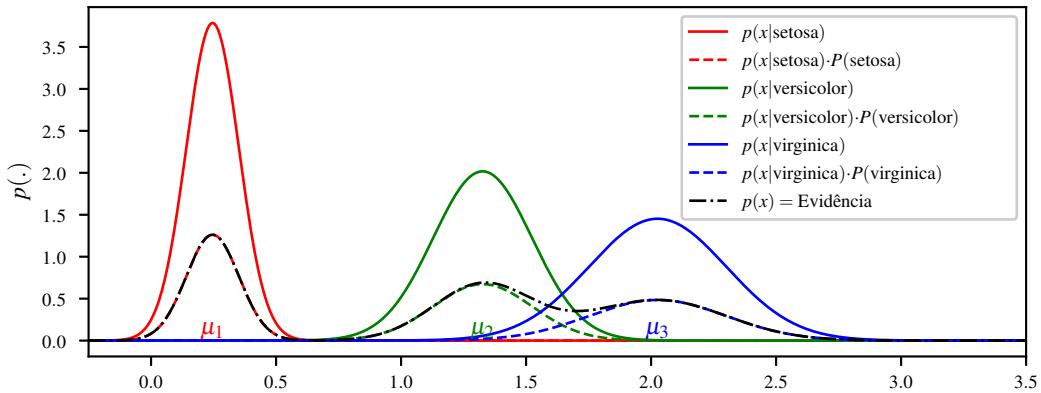
### 7.2.1 Classificador de Erro Mínimo

Uma tomada de decisão natural baseia-se na escolha daquela classe que exibiu a maior probabilidade a posteriori  $P(\mathcal{C}_k|x)$  para o certo valor  $x$ . Vamos considerar novamente a figura 7.1b. Para o valor de exemplo  $x = 1.5$ , a curva  $P(\text{versicolor}|1.5)$  mostra esse maior valor comparado com o valor menor para  $P(\text{virginica}|1.5)$ , e quase nulo para  $P(\text{setosa}|1.5)$ .

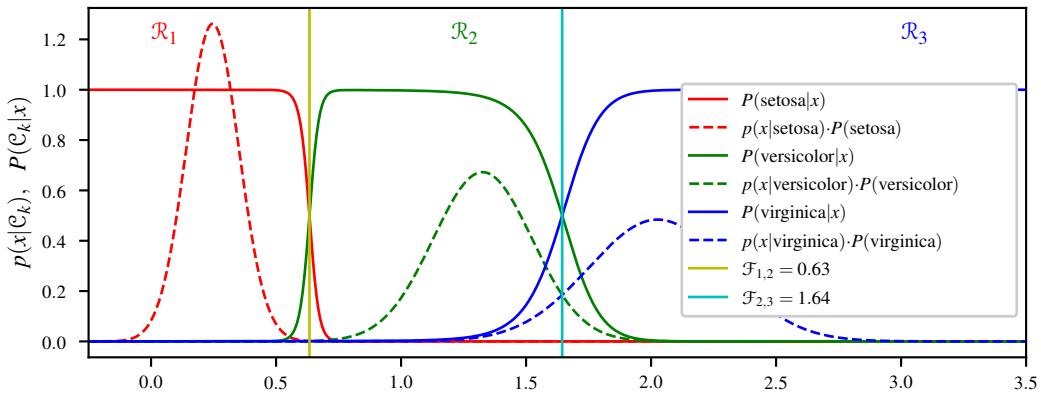
Esse estratégia é equivalente a minimizar o *erro de classificação*. Para analisarmos esta questão com mais facilidade, vamos dividir as classes em dois grupos para um certo valor  $x$ , a classe  $\mathcal{C}_{\max}$  e uma classe  $\mathcal{C}_{\text{não\_max}}$ . A classe  $\mathcal{C}_{\text{não\_max}}$  é a fusão de todas as classes complementares a  $\mathcal{C}_{\max}$ . A classe  $\mathcal{C}_{\max}$  tem a maior probabilidade a posteriori para o valor  $x$ , ou seja,

$$P(\mathcal{C}_{\max}|x) \geq P(\mathcal{C}_{\text{não\_max}}|x) \iff P(\mathcal{C}_{\max}|x) \geq P(\mathcal{C}_k), \quad k = 1, \dots, c, \quad k \neq \max$$

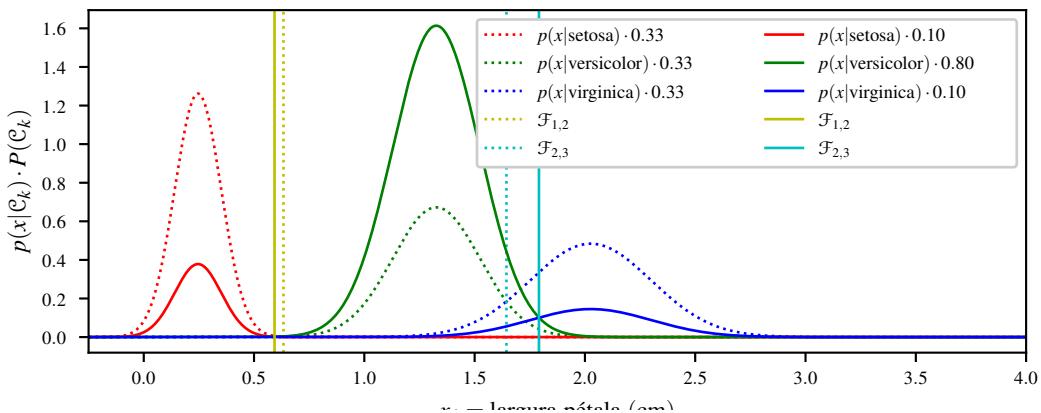
No exemplo 7.2, para o valor da característica  $x = 1.5$  temos as duas classes complementares  $\mathcal{C}_{\max} = \mathcal{C}_2 = \text{versicolor}$  e  $\mathcal{C}_{\text{não\_max}} = \{\mathcal{C}_1 \cup \mathcal{C}_3\} = \{\text{setosa} \cup \text{virginica}\}$ . Para esse valor a probabilidade a posteriori de versicolor é de 85.51%, de virginica 14.49% e para setosa 0.0%. Em outras palavras, temos 85.51% chance de decidir a classe certa, se escolhermos versicolor e  $14.49\% + 0.0\% = 14.49\%$  chance de cometer o erro. Se adotarmos essa estratégia de sempre escolher  $P(\mathcal{C}_{\max})$  para qualquer valor, automaticamente vamos sempre minimizar o erro para esse valor  $x$ . A probabilidade de cometer o erro é então somar todas as probabilidades das classes que *não* tenham o maior valor da probabilidade a posteriori para cada valor possível do domínio em questão, no caso do exemplo 7.2 a largura da pétala  $x = x_4$ . Omite-se neste ponto uma formalização mais detalhada da definição analítica da probabilidade  $P(\text{erro}|x)$ , pois não agrupa muita informação prática para construir um classificador. Para maiores detalhes, recomenda-se [86].



(a) Densidades condicionais  $p(x|\mathcal{C}_k)$  das três classes, o numerador  $p(x|\mathcal{C}_k) \cdot P(\mathcal{C}_k)$  da regra de Bayes para as três classes, e o denominador  $p(x)$  da regra de Bayes.



(b) Os numeradores  $p(x|\mathcal{C}_k) \cdot P(\mathcal{C}_k)$  da regra de Bayes para as três classes, as respectivas probabilidades a posterior  $P(\mathcal{C}_k|x)$ , e as fronteiras entre as classes, definidas pela igualdade de duas probabilidades a posterior iguais.



(c) Mudança da fronteira entre as classes, devida à probabilidades a prior desiguais das três classes.

Figura 7.1: Classificação paramétrica das três classes de Iris. Somente a última característica  $x_4$ , a largura da pétala é usada para calcular as densidades e probabilidades pela regra de Bayes.

**Definição 7.2.1 — Regra de Bayes de Classificação de Erro Mínimo.** Considere a definição 2.3.3 na pág. 62. Usando como *score* a maior probabilidade a posteriori  $P(\mathcal{C}_k|x)$ , define como função discriminativa  $F$  um classificador que minimiza o erro de classificação, ou seja,

$$F(P(\mathcal{C}_k|x)) = \mathcal{C}_k, \text{ se } P(\mathcal{C}_k|x) \geq P(\mathcal{C}_\ell|x), k \in \{1, \dots, c\}, \ell = 1, \dots, c, k \neq \ell. \quad (7.9)$$

Observado o numerador e denominador na regra de Bayes definição 7.1.1, o resultado não muda se omitirmos o denominador, a evidência  $p(x)$  que é um fator normalizador, e é igual para todas as classes. Então, para os mesmos índices  $k$  e  $\ell$ , a regra equivalente é

$$F(P(\mathcal{C}_k|x)) = \mathcal{C}_k, \text{ se } p(x|\mathcal{C}_k) \cdot P(\mathcal{C}_k) \geq p(x|\mathcal{C}_\ell) \cdot P(\mathcal{C}_\ell). \quad (7.10)$$

O logaritmo é uma função monótona que mantém a mesma relação que os seus argumentos, portanto mais uma regra equivalente é

$$F(P(\mathcal{C}_k|x)) = \mathcal{C}_k, \text{ se } \ln p(x|\mathcal{C}_k) + \ln P(\mathcal{C}_k) \geq \ln p(x|\mathcal{C}_\ell) + \ln P(\mathcal{C}_\ell). \quad (7.11)$$

### 7.2.2 Classificador de Verossimilhança Máxima

Já vimos na figura 7.1c que a probabilidade a priori tem uma influência sobre o resultado da classificação. Porém, nem sempre essa probabilidade está disponível nos cálculos, ou voluntariamente é ignorada, ou é igual para todas as classes. Se  $P(\mathcal{C})$  for omitida das regras de classificação temos a

**Definição 7.2.2 — Regra de Bayes de Classificação de Verossimilhança Máxima.** Consequentemente podemos reutilizar as regras anteriores (eq. 7.10) e (eq. 7.11) sem probabilidade a priori para obter

$$F(P(\mathcal{C}_k|x)) = \mathcal{C}_k, \text{ se } p(x|\mathcal{C}_k) \geq p(x|\mathcal{C}_\ell). \quad (7.12)$$

$$F(P(\mathcal{C}_k|x)) = \mathcal{C}_k, \text{ se } \ln p(x|\mathcal{C}_k) \geq \ln p(x|\mathcal{C}_\ell). \quad (7.13)$$

As verossimilhanças que na verdade equivalem às probabilidades condicionais, e o seu logaritmo neperiano são mostrados na figura 7.2. Observe que a fronteira entre as classes é a mesma para regra da (eq. 7.12) e (eq. 7.13).

### 7.2.3 Classificador de Risco Mínimo

Imagine uma situação de ameaça de bomba contra a reitoria da sua universidade. A polícia tem que decidir, se essa ameaça é real, ou somente fingida pelos criminosos, ou seja, tem que ser tomada uma decisão. Se a gente detecta versicolor, se na realidade temos uma virginica, a confusão não traz consequências severas, mas ignorar a existência de um perigo muito grande, mesmo pouco provável pela experiência, seria catastrófica. Surge a necessidade de introduzir um novo conceito, o *risco* e o *custo* de uma decisão. Temos então duas fases, a de classificar, e a de tomar uma decisão. Vamos associar a uma classificação um custo

$$\lambda(\mathcal{C}_\ell|\mathcal{C}_k), \quad (7.14)$$

abreviando como  $\lambda_{\ell,k}$ . A classe verdadeira é  $\mathcal{C}_k$  e a classe decidida é  $\mathcal{C}_\ell$ . Obviamente no exemplo anterior  $\lambda$  (não há bomba|há bomba) tem que ter um custo muito alto que vai ponderar a decisão

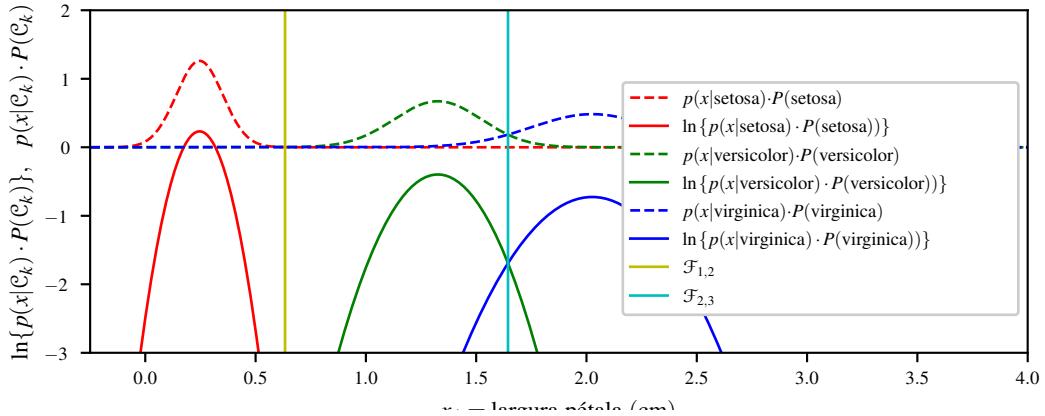


Figura 7.2: Classificador de verossimilhança máxima e seu logaritmo. As densidades são Gaussianas. As probabilidades a priori são iguais para as três classes e consequentemente serão ignoradas.

a ser tomada afinal das contas. Se a classificação for correta, normalmente o custo é zero, ou seja,  $\lambda(\mathcal{C}_k|\mathcal{C}_k) = 0$ . Todas as combinações possíveis com as classes verdadeiras e classificadas podem ser compilados em uma

**Definição 7.2.3 — Matriz de Custo em Classificador de Risco Mínimo.** Em uma tarefa de classificação, dada a classe verdadeira  $\mathcal{C}_k$  de um padrão e a classe resultante pela classificação  $\mathcal{C}_\ell$ , o custo desta decisão para todas as classes forma a matriz de custo

Classe decidida	Classe verdadeira							
	$\mathcal{C}_1$	$\mathcal{C}_2$	...	$\mathcal{C}_k$	...	$\mathcal{C}_\ell$	...	$\mathcal{C}_c$
$\mathcal{C}_1$	0	$\lambda_{1,2}$	...	$\lambda_{1,k}$	...	$\lambda_{1,\ell}$	...	$\lambda_{1,c}$
$\mathcal{C}_2$	$\lambda_{2,1}$	0	...	$\lambda_{2,k}$	...	$\lambda_{2,\ell}$	...	$\lambda_{2,c}$
$\vdots$	$\vdots$	$\vdots$	...	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\mathcal{C}_k$	$\lambda_{k,1}$	...	...	0	...	$\lambda_{k,\ell}$	...	$\lambda_{k,c}$
$\vdots$	$\vdots$	$\vdots$	...	$\vdots$	...	$\vdots$	$\vdots$	$\vdots$
$\mathcal{C}_\ell$	$\lambda_{\ell,1}$	...	...	$\lambda_{\ell,k}$	...	0	...	...
$\vdots$	$\vdots$	$\vdots$	...	$\vdots$	$\vdots$	$\vdots$	...	$\vdots$
$\mathcal{C}_c$	$\lambda_{c,1}$	...	...	$\lambda_{c,k}$	...	$\lambda_{c,\ell}$	...	0

(7.15)

■ **Exemplo 7.3** Vamos imaginar um processo industrial cuja condição operacional ou é normal, ou é falha. Esses dois estados constituem as duas classes. Um sistema de detecção de falhas tem a tarefa de processar informação sensorial, e de alguma maneira decidir, se tudo está em ordem, ou se existe uma falha. Neste exemplo simplificado vamos assumir que a falha causa a necessidade de parar a produção para consertar a causa da falha. Essa manutenção tem um certo custo, independente se o classificador detectou uma falha corretamente ou se o estado do processo é normal e mesmo assim foi detectado uma falha. Bem pior é o caso em que foi classificado que o processo opera normalmente, mas na verdade tem uma falha. Isso pode resultar em um sério dano

do equipamento, portanto tem que ter um custo alto. A matriz de custo<sup>1</sup> fictícia está em (eq. 7.16).

		Classe verdadeira		(7.16)
		Normal	Falha	
Classe classificada	Normal	0	10000	
	Falha	100	0	

**Definição 7.2.4 — Risco Condisional de Decidir Classe  $\mathcal{C}_\ell$ .** Supõe-se que para um certo padrão  $x$  uma classificação da classe  $\mathcal{C}_\ell$  seja feita, e que a classe verdadeira do padrão seja a classe  $\mathcal{C}_k$ . Então o risco condicional de tomar uma decisão é

$$R_\ell \stackrel{\text{def}}{=} \sum_{k=1}^c \lambda(\mathcal{C}_\ell|\mathcal{C}_k)P(\mathcal{C}_k|x) \quad (7.17)$$

Obs.: Se o custo de classificar errado for  $\lambda(\mathcal{C}_\ell|\mathcal{C}_k) = 1$  para  $k \neq \ell$  o classificador do risco mínimo é idêntico ao classificador do erro mínimo. Verifique!

■ **Exemplo 7.4** Considerando o exemplo 7.3 da matriz de custo do processo industrial, vamos supor que a probabilidade a posteriori da condição normal para um padrão  $x$  seja  $P(\text{Normal}|x) = 97.0\%$ , e para a falha  $P(\text{Falha}|x) = 3.0\%$ . Podemos calcular os dois riscos condicionais como

$$\begin{aligned} R_{\text{Normal}} &= \lambda(\text{Normal}|\text{Normal})P(\text{Normal}|x) + \lambda(\text{Normal}|\text{Falha})P(\text{Falha}|x) \\ &= 0 \cdot 0.97 + 10000 \cdot 0.03 = 300.0 \\ R_{\text{Falha}} &= \lambda(\text{Falha}|\text{Normal})P(\text{Normal}|x) + \lambda(\text{Falha}|\text{Falha})P(\text{Falha}|x) \\ &= 100 \cdot 0.97 + 0 \cdot 0.03 = 97.0 \end{aligned}$$

A decisão a ser tomada que constitui a menos ariscada, é classificar a existência de uma falha, embora a probabilidade é somente 3%. A razão é o alto custo de 10000 em caso de verdadeira ocorrência de uma falha, mas a classe classificada seria o estado normal do processo. ■

Relembrando a definição genérica de *score* e função discriminativa na seção 2.3.1 na pág. 59, podemos unificar os *scores* para os três classificações no contexto Bayesiano na (eq. 7.18).

**Definição 7.2.5 — Função de Score, Classificador Bayesiano.** Baseado na regra de Bayes definição 7.1.1, as definições da função de *score* da definição 2.3.1 na pág. 61 para cada classe  $\mathcal{C}_k$ ,  $k = 1, \dots, c$ , usadas na função discriminativa, definição 2.3.3 na pág. 62 são

$$y_k(x) = P(\mathcal{C}_k|x) \quad \text{Função de score para classificador de erro mínimo} \quad (7.18a)$$

$$y_k(x) = p(x|\mathcal{C}_k) \quad \text{Função de score para classificador de verossimilhança máxima} \quad (7.18b)$$

$$y_k(x) = -R_k \quad \text{Função de score para classificador de risco mínimo} \quad (7.18c)$$

<sup>1</sup>Neste contexto omitimos a classe  $\mathcal{C}_0$ , a “rejeição”, que delega a decisão para uma instância maior, por exemplo um especialista humano. Esse mecanismo entraria em ação, por exemplo, se as duas maiores probabilidades a priori estivessem muito próximo uma a outra. Se usássemos esse classe de rejeição, poderíamos definir  $c$  custos  $\lambda_{0,k}$ ,  $k = 1, \dots, c$ , que refletiriam o custo de uma rejeição para cada classe verdadeira. A nova matriz de custo  $(c+1) \times c$  teria mais uma linha antes da primeira linha.

Repare o sinal negativo no classificador de risco mínimo, pois queremos minimizar o valor do risco, mas a função de *score* em geral busca a classe com o valor máximo.

### 7.2.4 Análise Discriminativa Quadrática Multidimensional

Podemos estender os conceitos aprendidos da combinação da regra de Bayes com a distribuição Gaussiana para mais que uma dimensão  $m > 1$ . Relembrando a definição da densidade normal multidimensional na definição 1.4.11 na pág. 28, em analogia ao caso unidimensional (eq. 7.8), temos que introduzir um novo índice para cada classe  $\mathcal{C}_k$ , assim a densidade Gaussiana, específica de cada classe é

$$p(\mathbf{x}; \boldsymbol{\mu}_k, \Sigma_k) = \frac{1}{\sqrt{(2\pi)^m |\Sigma_k|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \Sigma_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k)\right). \quad (7.19)$$

Sendo a Gaussiana uma densidade de natureza exponencial, a aplicação do logaritmo traz a vantagem de simplificar os cálculos da verossimilhança  $p(\mathbf{x}|\mathcal{C}_k)$ , assim<sup>2</sup>

$$\ln p(\mathbf{x}; \boldsymbol{\mu}_k, \Sigma_k) = -\frac{m}{2} \ln(2\pi) - \frac{1}{2} \ln |\Sigma_k| - \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_k)^\top \Sigma_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k) \quad (7.20)$$

A regra de Bayes de classificação de erro mínimo, usando uma Gaussiana multidimensional especializa a regra (eq. 7.11) que comparar duas funções discriminativas  $y_k(\mathbf{x})$  e  $y_\ell(\mathbf{x})$  para

$$\begin{aligned} F(P(\mathcal{C}_k|\mathbf{x})) &= \mathcal{C}_k, \text{ se } y_k(\mathbf{x}) \geq y_\ell(\mathbf{x}) \\ &\quad -\frac{1}{2} \ln |\Sigma_k| - \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_k)^\top \Sigma_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k) + \ln P(\mathcal{C}_k) \\ &\geq -\frac{1}{2} \ln |\Sigma_\ell| - \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_\ell)^\top \Sigma_\ell^{-1} (\mathbf{x} - \boldsymbol{\mu}_\ell) + \ln P(\mathcal{C}_\ell), \end{aligned} \quad (7.21)$$

onde o termo  $-\frac{m}{2} \ln(2\pi)$  foi omitido, pois é independente da classe. O classificador de verossimilhança máxima ignora os termos da probabilidade a priori  $P(\mathcal{C}_k)$  e  $P(\mathcal{C}_\ell)$ . Multiplicando os dois lados por  $(-2)$ , temos

$$\begin{aligned} F(P(\mathcal{C}_k|\mathbf{x})) &= \mathcal{C}_k, \text{ se} \\ &\ln |\Sigma_k| + (\mathbf{x} - \boldsymbol{\mu}_k)^\top \Sigma_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k) \leq \ln |\Sigma_\ell| + (\mathbf{x} - \boldsymbol{\mu}_\ell)^\top \Sigma_\ell^{-1} (\mathbf{x} - \boldsymbol{\mu}_\ell) \\ &\iff \ln |\Sigma_k| + D_{M,k}^2 \leq \ln |\Sigma_\ell| + D_{M,\ell}^2, \end{aligned} \quad (7.22)$$

onde  $D_{M,k}^2$  e  $D_{M,\ell}^2$  são as distâncias de Mahalanobis quadráticas da definição 1.4.12 na pág. 28, específicas de cada classe, que um padrão  $\mathbf{x}$  tem do centro específico  $\boldsymbol{\mu}_k$  ou  $\boldsymbol{\mu}_\ell$ . Assim fica óbvio que o classificador associa os padrões à classe que tenha do seu centro uma distância pequena ao padrão, considerando ainda o escalamento diferenciado em cada característica, introduzido pela matriz de covariância. Reveja a figura 1.4 na pág. 30 que mostra as três densidades das classes de flores em um espaço bidimensional.

O uso de funções discriminativas introduz as regiões de decisão e fronteiras entre classes, veja, por exemplo, a divisão do espaço de característica criada pela máquina linear na figura 2.7 na pág. 64. A figura 7.3 mostra a situação para as três classes de iris. Uma variação fictícia das três probabilidades a priori desloca as fronteiras, veja também figura 7.1c. Repare a natureza quadrática das fronteiras, justificado o nome “Análise Discriminativa Quadrática”.

Vamos considerar três casos em relação à natureza da matriz de covariância, sem qualquer covariância e com variância igual em todas as dimensões, matrizes de covariância iguais para todas as classes, e covariâncias diferentes nas classes.

<sup>2</sup>Relembrando,  $\frac{1}{\sqrt{x}} = x^{-\frac{1}{2}}$ ,  $\log(a \cdot b) = \log a + \log b$ ,  $\log x^a = a \log x$ .

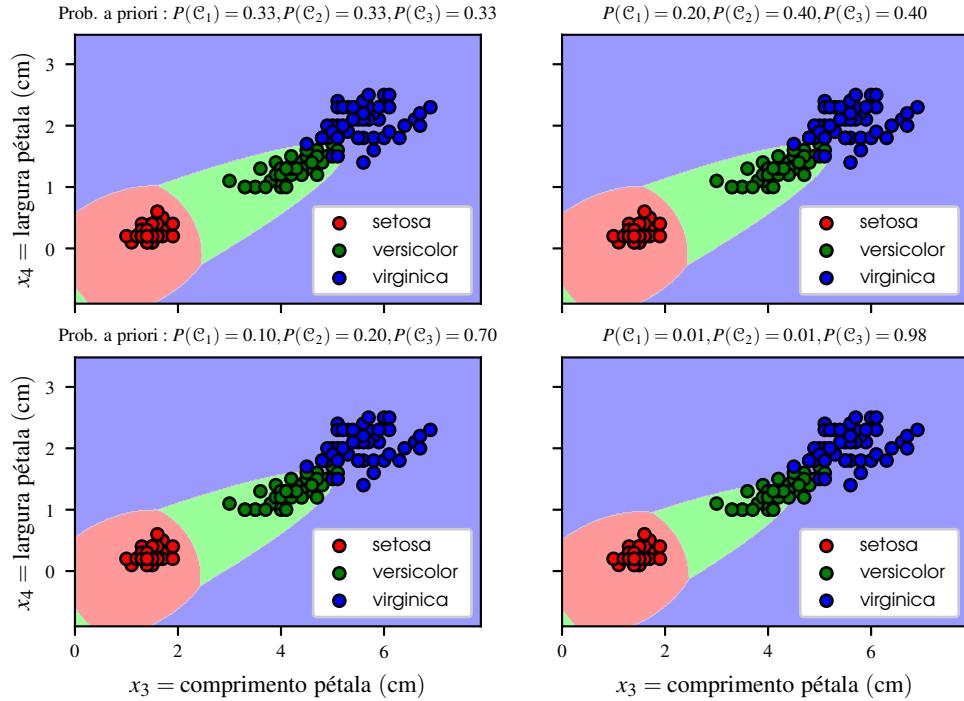


Figura 7.3: Regiões de decisão e fronteiras de decisão do classificador de Bayes de erro mínimo, assumindo distribuições Gaussianas das classes. Variações para probabilidades a priori diferentes.

### Sem Covariância, Variância Igual: Função Discriminativa Linear

A função discriminativa  $y_k(\mathbf{x})$  da  $k$ -ésima classe  $\mathcal{C}_k$  baseado no classificador do erro mínimo com densidade Gaussiana na (eq. 7.21) neste caso é uma matriz de identidade  $I$  multiplicada pela variância  $\sigma^2$  que é igual para todas as características, ou seja  $\sigma_1^2 = \sigma_2^2 = \dots = \sigma_m^2$ . Então  $\Sigma_k = \sigma^2 I$  e a  $k$ -ésima função discriminativa em (eq. 7.21) simplifica para

$$\begin{aligned} y_k(\mathbf{x}) &= -\frac{1}{2} \ln |\sigma^2 I| - \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_k)^T \left[ \frac{1}{\sigma^2} I \right] (\mathbf{x} - \boldsymbol{\mu}_k) + \ln P(\mathcal{C}_k) \\ &= -\frac{\|\mathbf{x} - \boldsymbol{\mu}_k\|^2}{2\sigma^2} + \ln P(\mathcal{C}_k). \end{aligned} \quad (7.23)$$

Neste cálculo foram usadas as regras do determinante de uma matriz diagonal de dimensão  $m \times m$ , portanto  $|\Sigma_k| = \sigma^{2m}$  e a identidade (eq. 1.4) na pág. 12,  $\mathbf{x}^T \mathbf{x} = \|\mathbf{x}\|^2$ . Além disso, o termo  $-\frac{1}{2} \ln |\sigma^2 I| = -\frac{1}{2} \ln \sigma^{2m} = -m \ln \sigma$  foi ignorado, pois ele é igual para todas as classes. Podemos expandir pela regra (eq. 1.11) na pág. 14  $\|\mathbf{a} - \mathbf{b}\|^2 = \mathbf{a} \cdot \mathbf{a} - 2\mathbf{a} \cdot \mathbf{b} + \mathbf{b} \cdot \mathbf{b}$ . Assim temos

$$\begin{aligned} y_k(\mathbf{x}) &= -\frac{1}{2\sigma^m} \left\{ \mathbf{x}^T \mathbf{x} - 2\boldsymbol{\mu}_k^T \mathbf{x} + \boldsymbol{\mu}_k^T \boldsymbol{\mu}_k \right\} \\ &\stackrel{\text{def}}{=} \mathbf{w}_k^T \mathbf{x} + w_{0,k}, \end{aligned} \quad (7.24)$$

que claramente é um modelo linear, veja definição 2.1.2 na pág. 32. Como o termo  $\mathbf{x}^T \mathbf{x}$  não é específico da classe, pode ser omitido da função discriminativa. Para a definição do vetor de pesos

e o viés foram usadas as abreviações

$$\mathbf{w}_k = \frac{1}{\sigma^2} \boldsymbol{\mu}_k$$

$$w_{0,k} = -\frac{1}{2\sigma^2} \boldsymbol{\mu}_k^\top \boldsymbol{\mu}_k + \ln P(\mathcal{C}_k).$$

As regiões de decisão desta simplificação linear são idênticas às mostradas na figura 2.7 na pág. 64.

### Matrizes de Covariância Iguais: Função Discriminativa Linear

O próximo caso especial das Gaussianas de todas as classes é uma matriz de covariância única para todas as  $c$  classes, isto é,  $\Sigma \stackrel{\text{def}}{=} \Sigma_1 = \dots = \Sigma_c$ . Desconsiderando novamente todos os termos que não são específicos de uma classe, a função discriminativa de (eq. 7.21) especializa para

$$y_k(\mathbf{x}) = -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_k)^\top \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}_k) + \ln P(\mathcal{C}_k). \quad (7.25)$$

Para expandir o primeiro termo, podemos reciclar uma conta parecida na (eq. 7.21). Surge um termo  $\mathbf{x}^\top \Sigma^{-1} \mathbf{x}$  que é independente da classe. Novamente temos uma função discriminativa linear

$$y_k(\mathbf{x}) = \mathbf{w}_k^\top \mathbf{x} + w_{0,k},$$

onde desta vez o vetor de pesos e o viés foram definidos como

$$\mathbf{w}_k = \Sigma^{-1} \boldsymbol{\mu}_k$$

$$w_{0,k} = -\frac{1}{2} \boldsymbol{\mu}_k^\top \Sigma^{-1} \boldsymbol{\mu}_k + \ln P(\mathcal{C}_k).$$

A diferença em relação ao primeiro caso é que desta vez as Gaussianas são hiperelipsoides, enquanto no primeiro caso mais simples eram hiperesferas. De qualquer maneira, lidamos novamente como uma Máquina Linear como classificador que foi bem estudada na seção 2.2 na pág. 53

### Matrizes de Covariância Diferentes por Classe: Função Discriminativa Quadrática

Vale como função discriminativa a definição da (eq. 7.21). Lembrando da estrutura de uma função quadrática multidimensional  $\frac{1}{2} \mathbf{x}^\top A \mathbf{x} + \mathbf{b}^\top \mathbf{x} + c$  da (eq. 3.19) na pág. 95 que estudamos no contexto da descida de gradiente, podemos reconhecer que a função discriminativa tem exatamente essa forma quadrática

$$y_k(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top W_k \mathbf{x} + \mathbf{w}_k^\top \mathbf{x} + w_{0,k}, \quad (7.26)$$

onde a matriz de pesos, o vetor de pesos, e o viés foram definidos como

$$W_k = -\Sigma_k^{-1}$$

$$\mathbf{w}_k = \Sigma_k^{-1} \boldsymbol{\mu}_k$$

$$w_{0,k} = -\frac{1}{2} \boldsymbol{\mu}_k^\top \Sigma_k^{-1} \boldsymbol{\mu}_k - \frac{1}{2} \ln |\Sigma_k| + \ln P(\mathcal{C}_k).$$

A natureza quadrática da função discriminativa dá origem ao termo “Função Discriminativa Quadrática”. As fronteiras de decisão entre as classes na geometria analítica são hiperquádricas (*hyperquadrics*), por exemplo, hiperplanos, hiperesferas, hiperelipsoides, hiperparaboloides, entre outros. Em [86], tem uma excelente apresentação dos gráficos dos três casos, em uma, duas e três dimensões.

### 7.3 Naïve Bayes

Foi usado o termo<sup>3</sup> “naïve” para expressar uma simplificação das pressuposições da relação das características que constituem um padrão. A gente já sabe que uma flor iris tem uma dependência entre a pétala e sépala, assim surgindo a necessidade de fazer uma análise multivariável, em vez de simplesmente assumir que esses dois atributos da flor sejam independentes. Um classificador Bayesiano que faz exatamente essa simplificação é um classificador “naïve” Bayes. A regra de Bayes na definição 7.1.1 trabalha com densidades  $p(\mathbf{x})$ , em que a variável aleatória em geral é um vetor  $\mathbf{x} = [x_1 \ \dots \ x_m]^T$ . Explicitando esse cada uma das variáveis aleatórias unidimensionais como argumento temos uma densidade  $p(\mathbf{x}) = p(x_1, \dots, x_m)$ . A vírgula entre os argumentos tem a semântica de *conjunção*, ou seja,  $p$  é uma densidade que depende de  $x_1$  e de  $x_2$  e de  $\dots$  e de  $x_m$ , assim expressando uma densidade conjunta. Uma regra fundamental da teoria de probabilidade fala sobre a independência de duas variáveis aleatórias. Dois eventos  $A$  e  $B$  são independentes, se, e somente se a sua probabilidade conjunta iguala o produto das duas probabilidades individuais, ou seja,

$$P(A \cap B) = P(A)P(B) \iff A \text{ independente de } B.$$

O símbolo “ $\cap$ ” da conjunção pode ser abreviado pela vírgula, ou seja,  $P(A \cap B) = P(A, B)$ . A mesma regra aplica-se para probabilidades  $P$  de variáveis aleatórias discretas  $X$  e  $Y$ , e densidades  $p$  de probabilidades de variáveis aleatórias contínuas  $x$  e  $y$ , ou seja,

$$P(X, Y) = P(X)P(Y) \iff X \text{ independente de } Y \quad (7.27a)$$

$$p(x, y) = p(x)p(y) \iff x \text{ independente de } y \quad (7.27b)$$

Para mais que uma variável, independência mútua entre as variáveis contínuas então significa

$$p(\mathbf{x}) = p(x_1, \dots, x_m) = p(x_1) \cdot p(x_2) \cdot \dots \cdot p(x_m) = \prod_{j=1}^m p(x_j). \quad (7.28)$$

Para uma densidade condicional, dada uma classe específica  $\mathcal{C}_k$ , essa regra obviamente se mantém, então

$$p(\mathbf{x}|\mathcal{C}_k) = p(x_1, \dots, x_m|\mathcal{C}_k) = p(x_1|\mathcal{C}_k) \cdot p(x_2|\mathcal{C}_k) \cdot \dots \cdot p(x_m|\mathcal{C}_k) = \prod_{j=1}^m p(x_j|\mathcal{C}_k). \quad (7.29)$$

**Definição 7.3.1 — Classificador Naïve Bayes.** O classificador baseado na regra de Bayes definição 7.1.1, assumindo independência entre todas as  $m$  características  $x_j$ ,  $j = 1, \dots, m$  é o classificador Naïve Bayes, definido como

$$\begin{aligned} P(\mathcal{C}_k|\mathbf{x}) &= \frac{p(\mathbf{x}|\mathcal{C}_k) \cdot P(\mathcal{C}_k)}{p(\mathbf{x})} = \frac{p(x_1, \dots, x_m|\mathcal{C}_k) \cdot P(\mathcal{C}_k)}{p(x_1, \dots, x_m)} = \frac{p(x_1, \dots, x_m|\mathcal{C}_k) \cdot P(\mathcal{C}_k)}{\sum_{\ell=1}^c p(x_1, \dots, x_m|\mathcal{C}_\ell) \cdot P(\mathcal{C}_\ell)} \\ &= \frac{\left\{ \prod_{j=1}^m p(x_j|\mathcal{C}_k) \right\} \cdot P(\mathcal{C}_k)}{\sum_{\ell=1}^c \left\{ \prod_{j=1}^m p(x_j|\mathcal{C}_\ell) \right\} \cdot P(\mathcal{C}_\ell)} \end{aligned} \quad (7.30)$$

A ignorância da dependência é equivalente a uma ausência de covariâncias  $\sigma_{j,\ell}$  entre características  $x_j$  e  $x_\ell$ .

Vale a pena mencionar novamente que a densidade  $p(\mathbf{x})$  pode ter qualquer aparência, mas que a gente se limita ainda na densidade normal (Gaussiana). Essa pressuposição também é aplicada no exemplo a seguir.

<sup>3</sup>A etimologia dessa palavra com um trema em cima do “i” aponta para o francês “naïf” (ingênuo, despreparado) e o latim “natus” (natural, nativo).

■ **Exemplo 7.5** Em relação às flores já temos as médias e matrizes de covariância por classe calculadas na (eq. 7.31) na pág. 201. Esse valores podem ser usados para o cálculo da densidade multidimensional na (eq. 7.19), e também para o cálculo da densidade unidimensional (eq. 7.8), pois as médias das características em ambos os casos fazem parte do cálculo, e as variâncias no Naïve Bayes são os elementos na diagonal da matriz de covariância de cada classe. Vamos nos limitar novamente às duas características  $x_3$  e  $x_4$  que já serviram diversas vezes como objeto de estudo, por exemplo, a última vez na figura 7.3. Vamos comparar o classificador de Bayes com densidade Gaussiana sem e com a pressuposição de independência. Usando essas duas características, temos as médias e matrizes de covariância por classes a partir dos valores em (eq. 7.31) na pág. 201 como

$$\begin{aligned}\hat{\boldsymbol{\mu}}_1 &= \begin{bmatrix} 1.462 \\ 0.246 \end{bmatrix}, & \hat{\boldsymbol{\mu}}_2 &= \begin{bmatrix} 4.260 \\ 1.326 \end{bmatrix}, & \hat{\boldsymbol{\mu}}_3 &= \begin{bmatrix} 5.552 \\ 2.026 \end{bmatrix}, \\ \hat{\Sigma}_1 &= \begin{bmatrix} 0.030 & 0.006 \\ 0.006 & 0.011 \end{bmatrix}, & \hat{\Sigma}_2 &= \begin{bmatrix} 0.221 & 0.073 \\ 0.073 & 0.039 \end{bmatrix}, & \hat{\Sigma}_3 &= \begin{bmatrix} 0.305 & 0.049 \\ 0.049 & 0.075 \end{bmatrix}.\end{aligned}\quad (7.31)$$

Além disso, a probabilidade a priori de cada classe é  $P(\mathcal{C}_k) = 1/3, k = 1, 2, 3$ .

Uma flor  $\tilde{\mathbf{x}}$  perto da fronteira entre virginica e versicolor teria aproximadamente o valor  $\tilde{\mathbf{x}} = [x_3 \ x_4]^\top = [5.1 \ 1.6]^\top$ , veja a figura 7.3. Para o cálculo multidimensional, supondo distribuição Gaussiana definida na (eq. 7.19) temos

$$\begin{aligned}p(\tilde{\mathbf{x}}|\mathcal{C}_1) &= p(\tilde{\mathbf{x}};\boldsymbol{\mu}_1, \Sigma_1) = 0.000 \\ p(\tilde{\mathbf{x}}|\mathcal{C}_2) &= p(\tilde{\mathbf{x}};\boldsymbol{\mu}_2, \Sigma_2) = 0.561 \\ p(\tilde{\mathbf{x}}|\mathcal{C}_3) &= p(\tilde{\mathbf{x}};\boldsymbol{\mu}_3, \Sigma_3) = 0.315\end{aligned}$$

A evidência (denominador na regra de Bayes) então é

$$p(\tilde{\mathbf{x}}) = p(\tilde{\mathbf{x}}|\mathcal{C}_1) \cdot P(\mathcal{C}_1) + p(\tilde{\mathbf{x}}|\mathcal{C}_2) \cdot P(\mathcal{C}_2) + p(\tilde{\mathbf{x}}|\mathcal{C}_3) \cdot P(\mathcal{C}_3) = 0.000 \cdot \frac{1}{3} + 0.561 \cdot \frac{1}{3} + 0.315 \cdot \frac{1}{3} = 0.2919,$$

e as três probabilidades a posteriori pela regra de Bayes, (eq. 7.4)

$$\begin{aligned}P(\mathcal{C}_1|\tilde{\mathbf{x}}) &= p(\tilde{\mathbf{x}}|\mathcal{C}_1) \cdot P(\mathcal{C}_1)/p(\tilde{\mathbf{x}}) = 0.000 \\ P(\mathcal{C}_2|\tilde{\mathbf{x}}) &= p(\tilde{\mathbf{x}}|\mathcal{C}_2) \cdot P(\mathcal{C}_2)/p(\tilde{\mathbf{x}}) = 0.641 \\ P(\mathcal{C}_3|\tilde{\mathbf{x}}) &= p(\tilde{\mathbf{x}}|\mathcal{C}_3) \cdot P(\mathcal{C}_3)/p(\tilde{\mathbf{x}}) = 0.359.\end{aligned}$$

Por outro lado, pelo Naïve Bayes, supondo distribuição Gaussiana unidimensional da (eq. 7.8) e independência entre as duas características  $x_3 = 5.1$  e  $x_4 = 1.6$  presentes, temos

$$p(\tilde{\mathbf{x}}|\mathcal{C}_k) = p(x_3|\mathcal{C}_k) \cdot p(x_4|\mathcal{C}_k) = p(x_3;\mu_{k,3}, \sigma_{k,3}^2) \cdot p(x_4;\mu_{k,4}, \sigma_{k,4}^2).$$

Então para as três classes

$$\begin{aligned}p(\tilde{\mathbf{x}}|\mathcal{C}_1) &= p(5.1; 1.462, 0.030) \cdot p(1.6; 0.246, 0.011) = 0.000 \cdot 0.000 = 0.000 \\ p(\tilde{\mathbf{x}}|\mathcal{C}_2) &= p(5.1; 4.260, 0.221) \cdot p(1.6; 1.326, 0.039) = 0.168 \cdot 0.765 = 0.129 \\ p(\tilde{\mathbf{x}}|\mathcal{C}_3) &= p(5.1; 1.462, 0.030) \cdot p(1.6; 0.246, 0.011) = 0.519 \cdot 0.430 = 0.223\end{aligned}$$

A evidência então é

$$p(\tilde{\mathbf{x}}) = p(\tilde{\mathbf{x}}|\mathcal{C}_1) \cdot P(\mathcal{C}_1) + p(\tilde{\mathbf{x}}|\mathcal{C}_2) \cdot P(\mathcal{C}_2) + p(\tilde{\mathbf{x}}|\mathcal{C}_3) \cdot P(\mathcal{C}_3) = 0.000 \cdot \frac{1}{3} + 0.129 \cdot \frac{1}{3} + 0.223 \cdot \frac{1}{3} = 0.117,$$

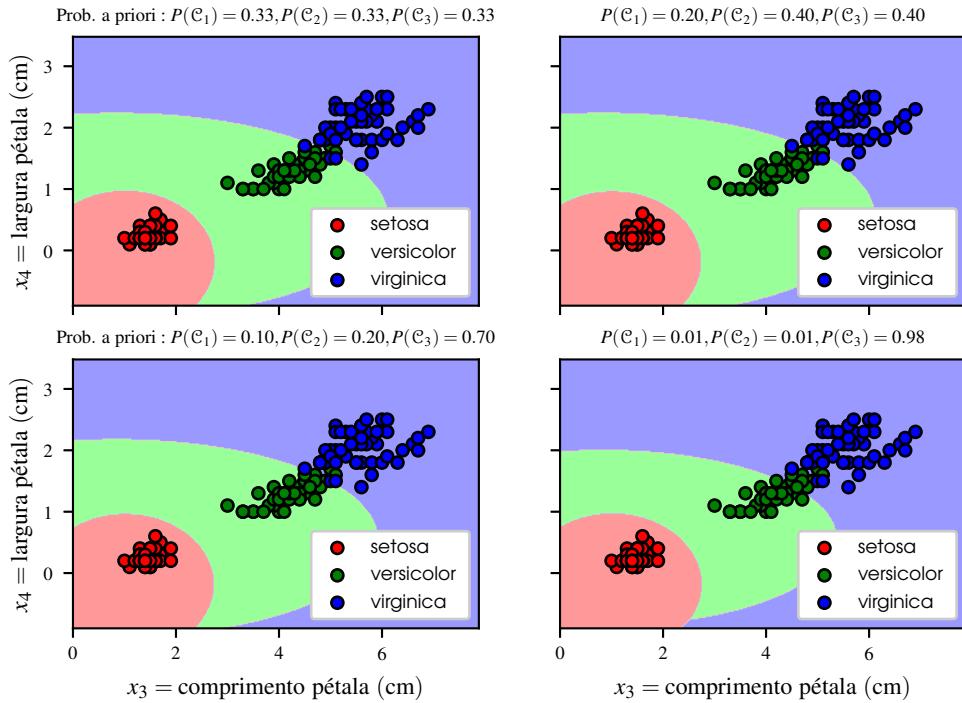


Figura 7.4: Regiões de decisão e fronteiras de decisão de classificador Naïve Bayes de erro mínimo, assumindo distribuições Gaussianas das classes, porém assumindo independência entre as características. Variações para probabilidades a priori diferentes.

e as três probabilidades a posteriori pela regra de Bayes, (eq. 7.4)

$$\begin{aligned}P(\mathcal{C}_1|\tilde{\mathbf{x}}) &= \{p(\tilde{x}_3|\mathcal{C}_1) \cdot p(\tilde{x}_4|\mathcal{C}_1)\} \cdot P(\mathcal{C}_1)/p(\tilde{\mathbf{x}}) = 0.000 \\P(\mathcal{C}_2|\tilde{\mathbf{x}}) &= \{p(\tilde{x}_3|\mathcal{C}_2) \cdot p(\tilde{x}_4|\mathcal{C}_2)\} \cdot P(\mathcal{C}_2)/p(\tilde{\mathbf{x}}) = 0.366 \\P(\mathcal{C}_3|\tilde{\mathbf{x}}) &= \{p(\tilde{x}_3|\mathcal{C}_3) \cdot p(\tilde{x}_4|\mathcal{C}_3)\} \cdot P(\mathcal{C}_3)/p(\tilde{\mathbf{x}}) = 0.634.\end{aligned}$$

Usando o classificador de erro mínimo, o Naïve Bayes decide virginica, enquanto o classificador multidimensional decide versicolor. ■

A figura 7.4 mostra as regiões de decisão do Naïve Bayes, compare com o classificador Bayesiano com Gaussiana multidimensional na figura 7.3. Repare que os eixos maior e menor das elipses são paralelos aos eixos horizontais e verticais  $x_3$  e  $x_4$ , expressando o fato que não existe covariância entre as características. Ao contrário, as elipses na figura 7.3 sofreram uma rotação em relação ao sistema de coordenadas original.

#### 7.4 Modelo Mistura de Gaussianas (Gaussian Mixture Model, GMM)

Novamente vamos lembrar que os modelos de classificadores paramétricos até agora estudados, usaram densidades  $p(\mathbf{x})$  de natureza Gaussiana. Em geral isso não necessariamente o caso. Um modelo de mistura define uma distribuição de uma variável aleatória como soma ponderada de mais que uma componente. Cada componente é de uma distribuição bem definida, até agora vimos, por exemplo, a normal e a uniforme, considere a figura 1.3 na pág. 18. O modelo apresentado a seguir é um modelo mistura, limitando-se à Gaussianas como componentes. Considere novamente

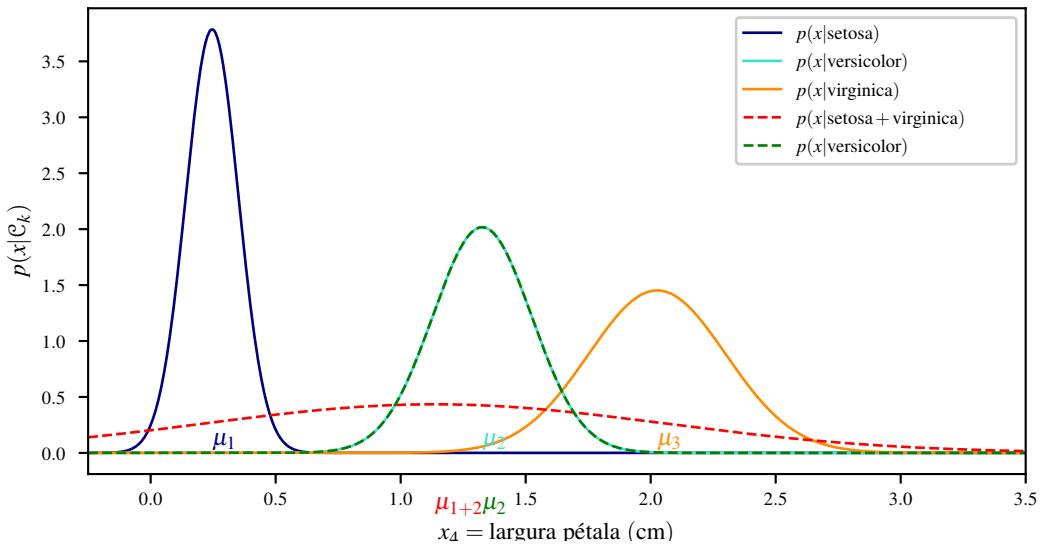


Figura 7.5: Fusão das duas classes setosa e virginica no espaço de características unidimensional da largura da pétala  $x_4$ . A modelagem usando uma única densidade Gaussiana é inapropriada, pois a densidade agora tem duas regiões com as amostras das antigas duas classes.

a Gaussiana unidimensional na figura 7.1. Fica claro que as amostras de uma classe pertence a regiões contíguas, ou seja, não pode acontecer que algumas flores da mesma classe  $\mathcal{C}_k$  tenham valores que se movimentam em torno de um valor  $\mu_k^a$  e outros valores que se movimentam em torno de um valor  $\mu_k^b$  que fica longe do primeiro “centro”  $\mu_k^a$ . Imagina agora o caso fictício que setosa e virginica decidiram que já não são classes diferentes, que os 50 amostras de cada classe se juntam para formar 100 novos exemplos da nova classe “setosa+virginica”. Vamos considerar somente uma característica, a largura da pétala  $x_4$ . Agora as 50 amostras da versicolor ficam cercados no meio de duas *modas* da outra classe, formada pela fusão das antigas classes setosa e virginica. A densidade *unimodal* de antes para setosa e virginica virou uma densidade *bimodal* das duas classes misturadas. Segue uma definição relativamente informal.

**Definição 7.4.1 — Moda de Distribuição de Variável Aleatória, Cluster.** Dada uma variável aleatória contínua unidimensional  $x \in \mathbb{R}$ , ou multidimensional  $\mathbf{x} \in \mathbb{R}^m$ , um máximo local da densidade é uma *moda*. A moda constitui o centro de um *cluster* de amostras. Distribuições com uma moda são unimodais, com duas modas bimodais, e com mais modas multimodais.

Se aplicarmos cegamente uma modelagem Gaussiana como antes, temos uma situação ilustrada na figura 7.5. A nova média  $\mu_{1\cup 2}$  da classe fusionada fica próxima da antiga e nova média da versicolor  $\mu_2$ . Além disso, a variância da classe fusionada é muito maior que antes, pois as amostras ficaram mais espalhadas. As antigas médias agora são modas da distribuição fusionada.

Uma modelagem muito mais poderosa que a Gaussiana unimodal é usar uma Gaussiana multimodal para cada classe. Esse é a ideia fundamental do modelo mistura de Gaussianas, GMM. O preço a ser pago que agora temos muito mais parâmetros e hiperparâmetros, pois ainda estamos no contexto de modelos paramétricos. Um dos hiperparâmetros a ser determinado, ou manualmente especificado pelo usuário, ou por um método de tuning, é a quantidade de componentes  $L$  de Gaussianas que compõem uma mistura. Na sua forma básica, o GMM é um método não supervisionado, ou seja, não existem classes. Somente temos uma distribuição  $p(\mathbf{x})$  de uma variável aleatória, sem ser condicional de uma certa classe, ou seja,  $p(\mathbf{x}|\mathcal{C}_k)$ . Esse distribuição é uma combinação linear

de distribuições de uma família de distribuições bem estudadas.

**Definição 7.4.2 — Modelo Mistura.** Uma distribuição de uma variável aleatória  $\mathbf{x}$  composto pela soma ponderada de  $L$  componentes é definido como

$$p(\mathbf{x}; \boldsymbol{\theta}) \stackrel{\text{def}}{=} \sum_{\ell=1}^L \pi_\ell p(\mathbf{x}; \boldsymbol{\theta}_\ell), \quad (7.32)$$

onde  $\pi_\ell$  é o peso da  $\ell$ -ésima componente, ou *coeficiente de mistura* [9, 10], e  $p(\mathbf{x}; \boldsymbol{\theta}_\ell)$  é a distribuição, definida pelos seus parâmetros  $\boldsymbol{\theta}_\ell$ . Os pesos das  $L$  componentes têm que somar 100%, ou seja

$$\sum_{\ell=1}^L \pi_\ell = 1. \quad (7.33)$$

**Definição 7.4.3 — Modelo Mistura de Gaussianas.** Uma densidade de uma variável aleatória  $\mathbf{x}$  contínua composta pela soma ponderada de  $L$  componentes que todas têm a forma de uma Gaussiana, é definido como

$$p(\mathbf{x}; \boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_L, \Sigma_1, \dots, \Sigma_L, \pi_1, \dots, \pi_L) = \sum_{\ell=1}^L \pi_\ell p(\mathbf{x}; \boldsymbol{\mu}_\ell, \Sigma_\ell), \quad (7.34)$$

onde  $\pi_\ell$  é o peso (coeficiente de mistura) da  $\ell$ -ésima Gaussiana, e  $p(\mathbf{x}; \boldsymbol{\mu}_\ell, \Sigma_\ell)$  é a  $\ell$ -ésima Gaussiana, definida pela sua média  $\boldsymbol{\mu}_\ell$  e matriz de covariância  $\Sigma_\ell$  na (eq. 1.47) na pág. 28. Todos os parâmetros foram acolhidos em  $\boldsymbol{\theta}$ , as médias, matrizes de covariância e coeficientes de mistura.

Note que os parâmetros *não* são associados a uma classe em um contexto de classificação. A mistura de várias Gaussianas serve apenas para modelar um distribuição complexa que não se limita a uma aglomeração das amostrar em torno de um único centro. Se trabalharmos com um problema de aprendizagem supervisionada, temos mais complexidade ainda. Temos que definir uma mistura de Gaussianas para cada classe. Nesse caso temos para cada uma das  $c$  classes uma distribuição condicional da classe, ou seja

$$p(\mathbf{x}; \boldsymbol{\theta}_k | \mathcal{C}_k) = p(\mathbf{x}; \boldsymbol{\mu}_{k,1}, \dots, \boldsymbol{\mu}_{k,L_k}, \Sigma_{k,1}, \dots, \Sigma_{k,L_k} | \mathcal{C}_k) = \sum_{\ell=1}^{L_k} \pi_{k,\ell} p(\mathbf{x}; \boldsymbol{\mu}_{k,\ell}, \Sigma_{k,\ell} | \mathcal{C}_k), \quad (7.35)$$

■ **Exemplo 7.6** Vamos considerar o exemplo anterior, em que duas classes de flores se reuniram para formar uma classe única. Obviamente o cenário ótimo é o seguinte:

1. A nova classe  $\mathcal{C}_1$  “setosa+virginica” é modelada pela soma de duas Gaussianas, ou seja,  $L_1 = 2$ , usando os coeficientes de mistura específica da classe  $\pi_{1,1}, \pi_{1,2}$ . As duas médias e matrizes de covariância são  $\boldsymbol{\mu}_{1,1}, \boldsymbol{\mu}_{1,2}, \Sigma_{1,1}, \Sigma_{1,2}$ . Como temos somente uma característica, o vetor de médias e a matriz de covariância simplificam para um escalar. Então temos duas médias e variâncias  $\mu_{1,1}, \mu_{1,2}, \sigma_{1,1}^2, \sigma_{1,2}^2$ .
2. A nova classe  $\mathcal{C}_2$ , que também é a antiga classe “versicolor” é modelada por uma Gaussiana somente, ou seja,  $L_2 = 1$ , usando um coeficiente de mistura específica da classe  $\pi_{2,1} = 100\%$ . Temos somente uma média  $\mu_{2,1}$  e uma variância  $\sigma_{2,1}^2$ .

A figura 7.6 mostra o resultado do processo de aprendizagem das componentes de um modelo GMM. Cada classe tem duas componentes. Em princípio a quantidade de componentes não é conhecida, aqui foi definido que cada classe dispõe de  $L_k = 2$  componentes. A primeira classe é a fusão de duas classes antigas, e consequentemente a densidade multimodal é mais apropriado para

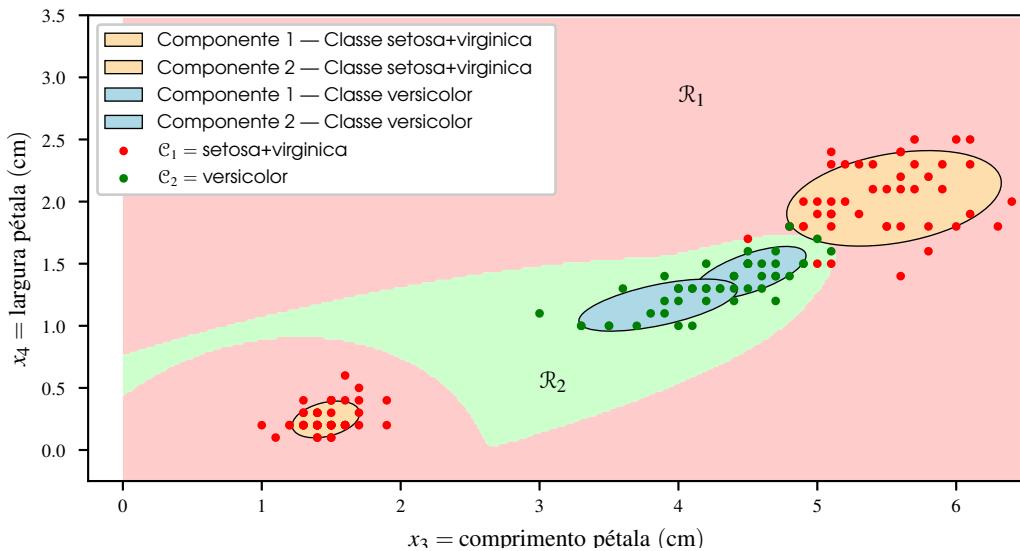


Figura 7.6: Regiões de decisão definidas por mistura de Gaussianas, específicas de cada classe. Cada elipse representa uma componente. Os dois semieixos correspondem ao maior e segundo maior autovalor, reveja a figura 4.2 na pág. 134. Compare com as regiões de decisão do classificador Bayesiano de erro mínimo na figura 7.3 na pág. 198 e do Naïve Bayes na figura 7.4 na pág. 202.

modelagem da classe fusionada que agora tem duas modas. Nada foi falado como as componentes foram aprendidos. Essa questão estudaremos a seguir. ■

## 7.5 Maximização da Verossimilhança

Já conhecemos a definição de uma distribuição probabilística. A densidade de uma variável aleatória contínua unidimensional  $x$ ,  $p(x; \boldsymbol{\theta})$ , ou multidimensional  $\mathbf{x}$ ,  $p(\mathbf{x}; \boldsymbol{\theta})$  foram definidas na seção 1.2.1 na pág. 17. A mais estudada é a densidade Gaussiana. Vamos considerar o caso unidimensional da (eq. 1.18) na pág. 18

$$p(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}. \quad (7.36)$$

Nesta equação os dois parâmetros  $\boldsymbol{\theta} = (\mu, \sigma^2)$  são invariáveis, ou seja, dado um valor  $x$ , calcule a densidade  $p(x)$ , sem “questionar”  $\mu$  e  $\sigma^2$ . Em seguida usamos a mesma função  $p$ , mas o papel o que é fixo e o que é variável muda.

### 7.5.1 Verossimilhança (Likelihood)

Agora vamos considerar a densidade de um ponto de vista de uma aprendizado de máquina, seja uma aprendizagem supervisionada, em que existem  $n$  exemplos  $x_1, \dots, x_n$  com rótulos de classes  $y_1, \dots, y_n$ , ou seja uma aprendizagem não supervisionada, onde existem somente  $n$  exemplos  $x_1, \dots, x_n$ . O valor de cada exemplo  $x_i$ ,  $i = 1, \dots, n$  é dado.

### 7.5.2 Verossimilhança para uma Amostra $x_1$

Vamos simplificar ainda mais o problema e considerar um único dado  $x_1$ . Ficando no contexto da Gaussiana sabemos que o processo aleatório foi responsável pela geração, ou seja,  $x_1$  é uma

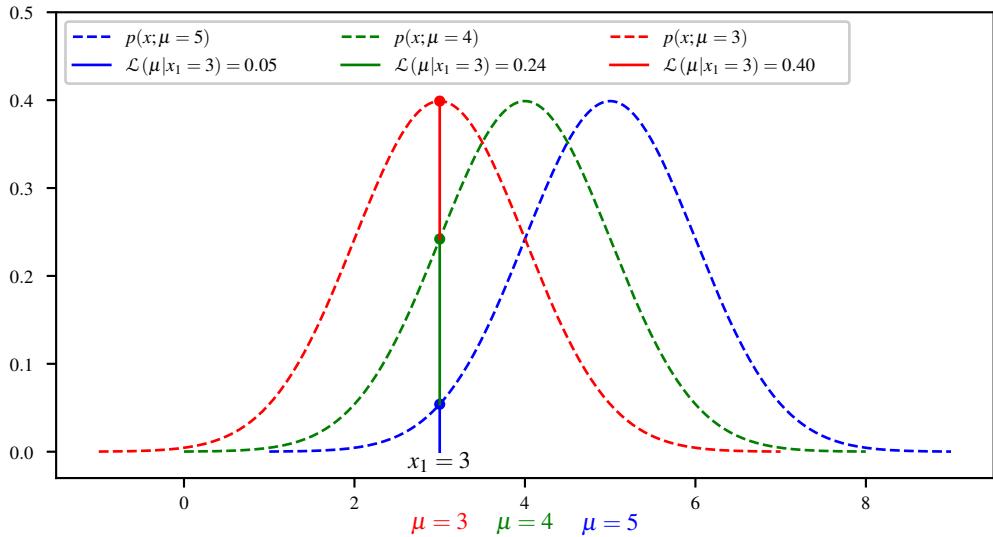


Figura 7.7: Função de verossimilhança  $\mathcal{L}(\mu|x_1)$  para um único padrão  $x_1$ . Também se mostram as três densidades Gaussianas  $p(x,\mu)$  para valores diferentes do parâmetro  $\mu$ . O outro parâmetro, a variância é deixada com o valor constante de  $\sigma^2 = 1$ .

amostra de uma densidade Gaussiana. A questão é a seguinte: Dado esse único exemplo, qual é a melhor estimativa da média e da variância que explica este valor? Para focar mais ainda nessa questão, assumimos uma variância  $\sigma^2 = 1$  constante, deixando assim ser um parâmetro. A resposta para a média é óbvia,  $\mu = x_1$ . Qualquer outra opção para o valor de  $\mu$  seria pior, porque é menos “plausível”. Por exemplo, se  $x_1 = 3$  e definimos  $\mu = 3000$  a única amostra fica muito longe da suposta média. A consequência é que a função  $p$  é praticamente nula. O nosso objetivo tem que ser a *maximização* de  $p$ . A densidade, onde a amostra  $x$  é fixo e os parâmetros  $\theta$  são variáveis é a

**Definição 7.5.1 — Função de Verossimilhança, Likelihood Function.** Com uma amostragem  $x$  fixa de uma densidade, variando os parâmetros da densidade o valor da densidade tem o significado de uma *verossimilhança*

$$\mathcal{L}(\theta|x) \stackrel{\text{def}}{=} p(x;\theta). \quad (7.37)$$

Repare que isso é idêntico à definição de uma densidade, mas agora o valor  $x$  é fixo e os parâmetros  $\theta$  podem mudar. O objetivo tem que ser a obtenção dos parâmetros que melhor explicam a amostra  $x$ .

■ **Exemplo 7.7** Vamos supor uma Gaussiana unidimensional com parâmetros  $\mu$  e  $\sigma^2 = 1$  constante, e uma única amostragem com o valor conhecido  $x_1 = 3$ . Na figura 7.7 mostram-se três escolhas diferentes para o único parâmetro restante  $\mu$ . Repare que no gráfico das três Gaussianas,  $x$  é o argumento da densidade  $p(x;\mu_j)$ ,  $j = 1, 2, 3$  para valores diferentes do parâmetro da média  $\mu$ . Por outro lado, na função de verossimilhança,  $x_1$  é um valor de uma amostra, e não muda o seu valor. ■

### 7.5.3 Verossimilhança para mais que uma Amostra

Evidentemente, a apresentação da verossimilhança com um único padrão  $x_1$  não tem nenhuma utilidade prática. No caso real de aprendizado de máquina, temos  $n$  amostras. Então a função de verossimilhança que vale para  $n$  amostras simultaneamente.

$$\mathcal{L}(\boldsymbol{\theta} | \{x_1, \dots, x_n\}) = \mathcal{L}(\boldsymbol{\theta} | \mathcal{D}), \quad (7.38)$$

onde os padrões formam o conjunto de dados de treinamento  $\mathcal{D} = \{x_i\}_{i=1}^n = \{x_1, \dots, x_n\}$ . Não consideramos rótulos de classes neste momento, ou seja, temos um problema de aprendizagem não supervisionado. Mais uma vez, como no contexto do Naïve Bayes na seção 7.3, a vírgula entre as amostras significa aqui densidade conjunta.

Uma das pressuposições essenciais em algoritmos que aprendem, é a independência na aquisição das amostras, ou seja, com as nossas 150 flores, a primeira setosa  $\mathbf{x}_1$  foi colhida independentemente da quinquagésima virginica  $\mathbf{x}_{150}$ . Na prática, frequentemente existe dependência entre os padrões adquiridos, especialmente se a obtenção foi feita em uma sequência temporal, como, por exemplo, em um processo industrial, onde medições de sensores capturam sinais que dependem de condições anteriores a aquisição. Novamente, como com o Naïve Bayes, a independência equivale o produto das densidades à densidade conjunta. Para a verossimilhança, isso significa

$$\mathcal{L}(\boldsymbol{\theta} | \{x_1, \dots, x_n\}) = \mathcal{L}(\boldsymbol{\theta} | x_1) \cdot \mathcal{L}(\boldsymbol{\theta} | x_2) \cdot \dots \cdot \mathcal{L}(\boldsymbol{\theta} | x_n) = \prod_{i=1}^n \mathcal{L}(\boldsymbol{\theta} | x_i). \quad (7.39)$$

Especialmente se a densidade tem uma natureza exponencial, com no caso da Gaussiana, convém considerar o logaritmo da verossimilhança, em vez da própria verossimilhança, pois isso simplifica as contas. Outro problema na (eq. 7.38) é a multiplicação de muitos termos que pode levar a instabilidades numéricas. Imagine um conjunto de dados com  $n = 10^6$  padrões. Então temos um milhão de termos multiplicados, o que certamente coloca em risco um resultado razoável. Como o logaritmo é, como aplicado no classificador Bayesian, uma função monótona que não interfere na ordem dos resultados, vamos trabalhar com o logaritmo da função de verossimilhança. Por padrão, o logaritmo  $\log(\cdot)$  é o logaritmo natural (neperiano)  $\ln(\cdot)$ . Assim

$$\ell(\boldsymbol{\theta} | \mathcal{D}) \stackrel{\text{def}}{=} \log \mathcal{L}(\boldsymbol{\theta} | \mathcal{D}) = \log [\mathcal{L}(\boldsymbol{\theta} | x_1) \cdot \mathcal{L}(\boldsymbol{\theta} | x_2) \cdot \dots \cdot \mathcal{L}(\boldsymbol{\theta} | x_n)] = \sum_{i=1}^n \log \mathcal{L}(\boldsymbol{\theta} | x_i). \quad (7.40)$$

■ **Exemplo 7.8** Consideremos as primeiras quatro amostras da setosa, para a primeira característica, o comprimento da sépala. Os valores<sup>4</sup> são  $x_1 = 5.1, x_2 = 4.9, x_3 = 4.7, x_4 = 4.6$ . Na figura 7.8 mostra-se o valor da densidade dos quatro padrões para duas médias diferentes. A primeira média é  $\mu^* = 4.825$ , e segunda média é  $\mu = 6.0$ . A função de verossimilhança da (eq. 7.39) para a primeira e segunda média tem os valores

$$\begin{aligned} \mathcal{L}(\mu^* | \{x_1, x_2, x_3, x_4\}) &= \mathcal{L}(\mu^* | x_1) \cdot \mathcal{L}(\mu^* | x_2) \cdot \mathcal{L}(\mu^* | x_3) \cdot \mathcal{L}(\mu^* | x_4) \\ &= 0.3841 \cdot 0.3978 \cdot 0.3959 \cdot 0.3890 = 0.0235 \\ \mathcal{L}(\mu | \{x_1, x_2, x_3, x_4\}) &= \mathcal{L}(\mu | x_1) \cdot \mathcal{L}(\mu | x_2) \cdot \mathcal{L}(\mu | x_3) \cdot \mathcal{L}(\mu | x_4) \\ &= 0.2661 \cdot 0.2179 \cdot 0.1714 \cdot 0.1497 = 0.0015. \end{aligned}$$

O logaritmo das duas verossimilhanças é  $\ell(\mu^* | \mathcal{D}) = \log \mathcal{L}(\mu^* | \mathcal{D}) = -3.750$  e  $\ell(\mu | \mathcal{D}) = \log \mathcal{L}(\mu | \mathcal{D}) = -6.512$ . Fica evidente que para a média  $\mu^*$  a verossimilhança e seu logaritmo tem um valor bem mais alto do que para o valor  $\mu$ , ou seja, o valor do parâmetro da média  $\mu^* = 4.825$  causa mais verossimilhança do que o valor do parâmetro da média  $\mu^* = 6.0$ . ■

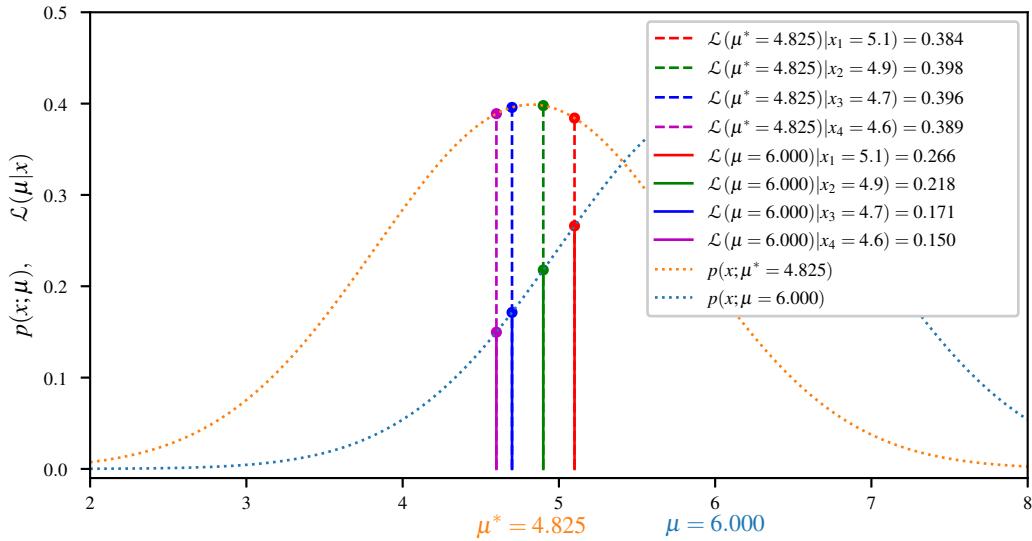


Figura 7.8: Função de verossimilhança  $\mathcal{L}(\mu|x_i)$  para quatro padrões  $x_i$ ,  $i = 1, 2, 3, 4$ , usando dois valores diferentes para o parâmetro da média.

No exemplo anterior o valor da média que rendeu uma verossimilhança maior tem o valor  $\mu^* = 4.825$ . Este valor é exatamente a média aritmética dos quatro pontos, ou seja  $\mu^* = \frac{1}{4} \sum_{i=1}^4 x_i$ . A figura 7.9 mostra a função de verossimilhança a seu logaritmo, variando o valor do parâmetro, que é a média  $\mu$ . Na média aritmética  $\bar{x} = \frac{1}{4} \sum_{i=1}^4 x_i$  as funções têm um máximo. Esse fato é plausível e tem por trás uma justificativa teórica, a maximização da verossimilhança, apresentado a seguir.

#### 7.5.4 Maximização Determinística da Verossimilhança

Para o caso da Gaussiana somos capazes de maximizar a verossimilhança analiticamente. Considerando o caso unidimensional com variância fixa, a figura 7.9 revela que a verossimilhança para um conjunto de dados fixo é uma função convexa cuja maximização deve ter uma solução relativamente fácil. Para manter uma certa consistência com a teoria de otimização até agora apresentada, em vez de maximizar a verossimilhança ou seu logaritmo, vamos minimizar o logaritmo negativo, fornecendo um resultado idêntico. Dessa maneira procuramos os parâmetros  $\boldsymbol{\theta}^*$  que maximizam a função de verossimilhança, relativo aos dados de treinamento  $\mathcal{D} = \{x_1, \dots, x_n\}$ , onde os dados obedecem a uma distribuição probabilística paramétrica com os parâmetros  $\boldsymbol{\theta}$ . Assim

$$\boldsymbol{\theta}^* = \arg \min_{\boldsymbol{\theta}} [-\ell(\boldsymbol{\theta}|\mathcal{D})], \quad (7.41)$$

onde podemos definir<sup>5</sup> uma função de perda

$$L(\boldsymbol{\theta}) \stackrel{\text{def}}{=} -\ell(\boldsymbol{\theta}|\mathcal{D}), \quad (7.42)$$

que tem que ser minimizada, compare com a (eq. 2.15) na pág. 36 que define o risco empírico. Note mais uma vez que neste contexto, não temos um problema de aprendizagem supervisionada,

<sup>4</sup>Como já mencionado, usamos  $x_1$  para a primeira amostra, ou para a primeira característica, dependendo do contexto. Aqui, o índice é o contador das 150 amostras de iris.

<sup>5</sup>Não confunda o “ $\mathcal{L}$ ” da função da verossimilhança, o seu logaritmo “ $\ell$ ” e a função de perda “ $L$ ”.

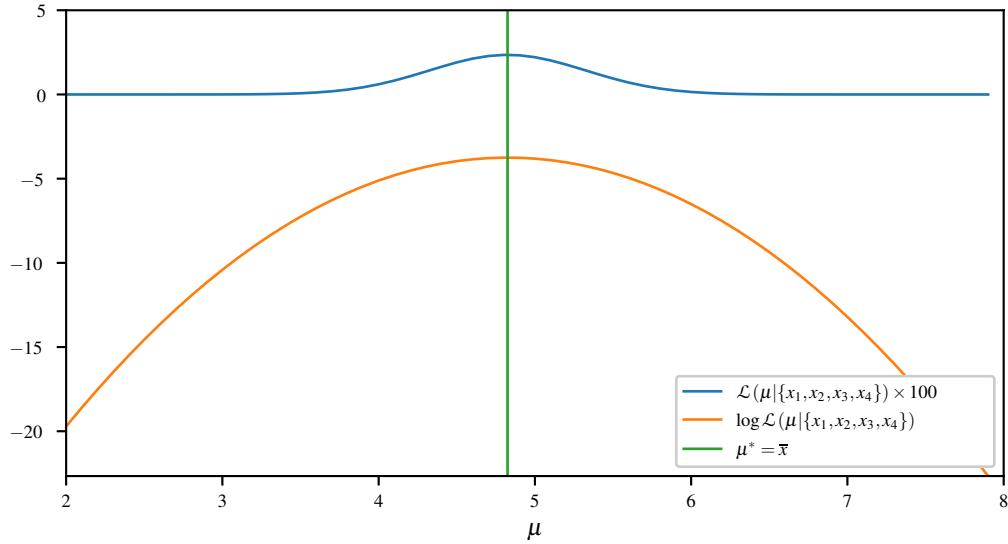


Figura 7.9: Função de verossimilhança  $\mathcal{L}(\mu|\{x_1, x_2, x_3, x_4\})$  para quatro padrões  $x_i$ ,  $i = 1, 2, 3, 4$ , variando o parâmetro da média. A verossimilhança foi multiplicada pelo fator 100 para ampliar os valores pequenos  $p(\mu|\mathcal{D})$ .

portanto não existem rótulos de classes. Além disso, a função de perda na aprendizagem supervisionada normalmente é o erro quadrático. Aqui temos a verossimilhança negativa, e o cálculo não é baseado na discrepância entre valores desejados e explicados pelo modelo.

No exemplo 7.8, temos

$$\begin{aligned} 4.825 &= \mu^* = \arg \min_{\mu} [-\ell(\mu|\{x_1, x_2, x_3, x_4\})] \\ &= \arg \min_{\mu} [-\{\log p(x_1; \mu) + \log p(x_2; \mu) + \log p(x_3; \mu) + \log p(x_4; \mu)\}] \end{aligned}$$

O logaritmo da densidade unidimensional para uma variância uniforme e fixa  $\sigma^2 = 1$ , considerando a (eq. 1.19) na pág. 18 é

$$\ln p(x; \mu) = -\frac{1}{2} [\ln 2\pi + (x - \mu)^2].$$

Assim temos

$$\begin{aligned} \mu^* &= \arg \min_{\mu} \left[ \frac{1}{2} \{ \ln 2\pi + (x_1 - \mu)^2 + \ln 2\pi + (x_2 - \mu)^2 \right. \\ &\quad \left. + \ln 2\pi + (x_3 - \mu)^2 + \ln 2\pi + (x_4 - \mu)^2 \} \right] \\ &= \arg \min_{\mu} \left[ 2 \ln 2\pi + \frac{1}{2} x_1^2 - x_1 \mu + \frac{1}{2} \mu^2 + \frac{1}{2} x_2^2 - x_2 \mu + \frac{1}{2} \mu^2 \right. \\ &\quad \left. + \frac{1}{2} x_3^2 - x_3 \mu + \frac{1}{2} \mu^2 + \frac{1}{2} x_4^2 - x_4 \mu + \frac{1}{2} \mu^2 \right] \\ &= \arg \min_{\mu} \left[ 2 \ln 2\pi + \frac{1}{2} (x_1^2 + x_2^2 + x_3^2 + x_4^2) - \mu (x_1 + x_2 + x_3 + x_4) + 2\mu^2 \right] \stackrel{\text{def}}{=} \arg \min_{\mu} L(\mu). \end{aligned}$$

Temos aqui o problema de minimização de uma função de perda  $L(\mu)$  quadrática que depende da variável  $\mu$ . Já resolvemos isto deterministicamente, zerando o gradiente. Assim zerando

$$\frac{dL(\mu)}{d\mu} = -(x_1 + x_2 + x_3 + x_4) + 4\mu,$$

temos

$$\mu = \frac{x_1 + x_2 + x_3 + x_4}{4} = \bar{x}.$$

Esse cálculo com somente quatro padrões em geral resulta no mesmo resultado, ou seja a média dos padrões maximiza a função de verossimilhança . Vamos generalizar o formalismo. A função de perda a ser minimizada na tentativa de maximizar a verossimilhança é

$$L(\boldsymbol{\theta}) = -\ell(\boldsymbol{\theta}|\mathcal{D}) = -\ln \mathcal{L}(\boldsymbol{\theta}|\mathcal{D}) = -\ln \prod_{i=1}^n \mathcal{L}(\boldsymbol{\theta}|\mathbf{x}_i) = -\sum_{i=1}^n \ln \mathcal{L}(\boldsymbol{\theta}|\mathbf{x}_i), \quad (7.43)$$

onde  $\boldsymbol{\theta}$  são todos os parâmetros do modelo paramétrico e os  $\mathbf{x}_i$  todos os padrões, em geral com mais que uma dimensão, que fazem parte do conjunto de treinamento. Um método determinístico tenta zerar o gradiente desta função de perda em relação aos parâmetros. Reveja a condição necessária para o mínimo de função vetorial na definição 2.1.10 na pág. 47 e as regras para derivadas multidimensionais na definição 2.1.11 na pág. 49. Temos para o gradiente da função de perda

$$\frac{dL(\boldsymbol{\theta})}{d\boldsymbol{\theta}} = \nabla L(\boldsymbol{\theta}) = \nabla \left( -\sum_{i=1}^n \ln \mathcal{L}(\boldsymbol{\theta}|\mathbf{x}_i) \right) = -\sum_{i=1}^n \nabla \ln \mathcal{L}(\boldsymbol{\theta}|\mathbf{x}_i). \quad (7.44)$$

Temos uma solução determinística então, se conseguimos resolver a expressão do gradiente zerado, ou seja resolver o sistema de  $p$  equações não lineares

$$\sum_{i=1}^n \nabla \ln \mathcal{L}(\boldsymbol{\theta}|\mathbf{x}_i) = \mathbf{0}^\top = \begin{bmatrix} \sum_{i=1}^n \frac{\partial \ln \mathcal{L}(\boldsymbol{\theta}|\mathbf{x}_i)}{\partial \theta_1} = 0 \\ \vdots \\ \sum_{i=1}^n \frac{\partial \ln \mathcal{L}(\boldsymbol{\theta}|\mathbf{x}_i)}{\partial \theta_p} = 0 \end{bmatrix}^\top, \quad (7.45)$$

onde  $p$  é a quantidade dos parâmetros  $\theta_1, \dots, \theta_p$  no vetor de parâmetros  $\boldsymbol{\theta}$ . Foi usada a regra que a derivada de soma é soma de derivada.

■ **Exemplo 7.9** Vamos novamente focar na Gaussiana unidimensional (eq. 7.36). O padrão  $\mathbf{x}$  é um escalar  $x$ . Temos dois parâmetros, a media  $\mu$  e a variância  $\sigma^2$ , o seja  $\boldsymbol{\theta} = (\theta_1, \theta_2) = (\mu, \sigma^2)$ . O logaritmo da função de verossimilhança é

$$\ln \mathcal{L}((\mu, \sigma^2)|x_i) = \ln p(x_i; \mu, \sigma^2) = -\frac{1}{2} \ln 2\pi\sigma - \frac{(x_i - \mu)^2}{2\sigma^2}$$

A derivada parcial do logaritmo da função de verossimilhança em relação ao primeiro parâmetro  $\mu$  que precisamos para cada componente na (eq. 7.45) é

$$\begin{aligned} \frac{\partial \ln \mathcal{L}((\mu, \sigma^2)|x_i)}{\partial \mu} &= \frac{\partial (-\frac{1}{2} \ln 2\pi\sigma)}{\partial \mu} + \frac{\partial (-\frac{1}{2\sigma^2}(x_i - \mu)^2)}{\partial \mu} = 0 - \frac{1}{\sigma^2}(x_i - \mu)(-1) \\ &= \frac{x_i - \mu}{\sigma^2}. \end{aligned}$$

Consequentemente, zerando uma componente na (eq. 7.45) equivale a

$$0 = \sum_{i=1}^n \frac{\partial \ln \mathcal{L}((\mu, \sigma^2)|x_i)}{\partial \mu} = \sum_{i=1}^n \frac{x_i - \mu}{\sigma^2} = \frac{1}{\sigma^2} \left( -n\mu + \sum_{i=1}^n x_i \right).$$

Temos que resolver esta equação segundo o parâmetro. Então o valor do parâmetro  $\mu$  que zera a expressão, ou seja que maximiza a função de verossimilhança é

$$\mu^* = \frac{1}{n} \sum_{i=1}^n x_i.$$

Este resultado geral para  $n$  padrões já foi visto no exemplo anterior com somente quatro padrões. Falta a obtenção do segundo parâmetro  $\sigma^2$ , portanto, temos que resolver

$$0 = \sum_{i=1}^n \frac{\partial \ln \mathcal{L}((\mu, \sigma^2) | x_i)}{\partial \sigma^2}.$$

O resultado [10], sem prova, é

$$\sigma^{2*} = \frac{1}{n-1} \sum_{i=1}^n (x_i - \mu^*)^2.$$

Para o caso de um padrão com mais que uma dimensão, a maximização da verossimilhança resulta nos estimadores conhecidos do vetor de média e matriz de covariância

$$\begin{aligned}\boldsymbol{\mu}^* &= \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i \\ \Sigma^* &= \frac{1}{n-1} \sum_{i=1}^n (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^\top.\end{aligned}$$

■

### 7.5.5 Maximização Iterativa da Verossimilhança, Algoritmo EM-GMM Esperança-Maximização (EM) para Mistura de Gaussianas GMM

Em geral em um modelo mistura, (eq. 7.32), não somos mais capazes de ter uma forma fechada para maximizar a verossimilhança, e obter todos os parâmetros em questão. No caso da mistura de  $L$  Gaussianas, (eq. 7.34), por exemplo, não fica claro se uma amostra qualquer  $\mathbf{x}_i$  é fruto da Gaussiana  $A$  ou da Gaussiana  $B$ , então a amostra não pode ser usada para estimar a média de uma determinada Gaussiana sem dúvida, pois não se sabe, se a amostra pertence a essa Gaussiana. Essa associação nebulosa de uma amostra a uma componente da mistura pode ser modelada pela *responsabilidade*.

**Definição 7.5.2 — Responsabilidade de Componente de Mistura na Geração de Amostra.** Dado um modelo mistura (eq. 7.32) com  $L$  componentes, com coeficientes de mistura  $\pi_\ell$ , e os restantes parâmetros  $\boldsymbol{\theta}_\ell$ , a responsabilidade na geração da  $i$ -ésima amostra  $\mathbf{x}_i$  do conjunto de treinamento é

$$r_{i,\ell} \stackrel{\text{def}}{=} \frac{\pi_\ell p(\mathbf{x}_i; \boldsymbol{\theta}_\ell)}{\sum_{\ell=1}^L \pi_\ell p(\mathbf{x}_i; \boldsymbol{\theta}_\ell)} \quad i = 1, \dots, n; \quad \ell = 1, \dots, L \quad (7.46)$$

Todas as responsabilidades podem ser armazenadas em uma matriz de dimensão  $n \times L$ . A própria responsabilidade também é considerado um parâmetro da  $\ell$ -ésima componente, porém convém separá-la dos restantes parâmetros. O denominador nesta expressão tem o mesmo papel como a evidência (eq. 7.3), na regra de Bayes (eq. 7.4). É um fator normalizador que garante que todas as responsabilidades de cada componente somam o valor unitário, marginalizado sobre as componentes. Assim

$$\sum_{\ell=1}^L r_{i,\ell} = 1. \quad (7.47)$$

Consequentemente podemos interpretar a responsabilidade como uma probabilidade. Expressa quanto provável a  $\ell$ -ésima componente participou na geração da  $i$ -ésima amostra. Marginalizando a responsabilidade sobre todas as  $n$  amostras de uma determinada componente

$$r_{\cdot,\ell} = \sum_{i=1}^n r_{i,\ell}. \quad (7.48)$$

expressa a fração de responsabilidade da  $\ell$ -ésima componente na modelagem dos dados.

Mesmo com este modelo mistura de Gaussianas, o objetivo ainda é a maximização da verosimilhança (eq. 7.38) ou seu logaritmo (eq. 7.40), ou equivalentemente, a minimização da função de perda (eq. 7.42), para encontrar o conjunto ótimo  $\boldsymbol{\theta}^*$  de parâmetros. Com o modelo definido na (eq. 7.32), a função de perda em relação aos parâmetros, (eq. 7.44) a ser minimizada é

$$\begin{aligned} L(\boldsymbol{\theta}) &= -\sum_{i=1}^n \ln \mathcal{L}(\boldsymbol{\theta} | \mathbf{x}_i) \\ &= -\sum_{i=1}^n \ln \sum_{\ell=1}^L \pi_\ell p(\mathbf{x}_i; \boldsymbol{\theta}_\ell). \end{aligned} \quad (7.49)$$

Zerando o gradiente desta função de perda, como condição necessária de um mínimo, seria a primeira tentativa de obter os parâmetros ótimos  $\boldsymbol{\theta}^*$ . No caso do modelo  $p(\mathbf{x}|\boldsymbol{\theta})$  de mistura de Gaussianas são todos os  $L$  coeficientes de mistura  $\pi_\ell$ , todos os  $L$  vetores de médias  $\boldsymbol{\mu}_\ell$ , e todas as  $L$  matrizes de covariância  $\Sigma_\ell$ . Apresentam-se os resultados para o cálculo dos parâmetros sem os passos intermediários que podem ser consultados em [10]. Zerando a derivada parcial da função de perda (eq. 7.49) em relação à  $\ell$ -ésima média resultada em

$$\begin{aligned} \frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\mu}_\ell} &= \mathbf{0}^\top \\ \implies \boldsymbol{\mu}_\ell &= \frac{1}{n_\ell} \sum_{i=1}^n r_{i,\ell} \mathbf{x}_i, \end{aligned} \quad (7.50a)$$

onde  $n_\ell$  pode ser interpretada como quantidade de amostras associadas à  $\ell$ -ésima componente da mistura, definido como

$$n_\ell \stackrel{\text{def}}{=} \sum_{i=1}^n r_{i,\ell}, \quad (7.50b)$$

o que é equivalente a fração de responsabilidade da  $\ell$ -ésima componente, definida na (eq. 7.48). Zerando a derivada parcial da função de perda em relação à  $\ell$ -ésima matriz de covariância resulta em

$$\begin{aligned} \frac{\partial L(\boldsymbol{\theta})}{\partial \Sigma_\ell} &= \mathbf{0}^\top \\ \implies \Sigma_\ell &= \frac{1}{n_\ell} \sum_{i=1}^n r_{i,\ell} (\mathbf{x}_i - \boldsymbol{\mu}_\ell) (\mathbf{x}_i - \boldsymbol{\mu}_\ell)^\top. \end{aligned} \quad (7.50c)$$

A obtenção dos coeficientes de mistura  $\pi_\ell$  envolve a restrição  $\sum_\ell \pi_\ell = 1$ . A técnica da função Lagrangiana da definição 4.2.2 na pág. 130 é usada para incorporar esta restrição. Assim a seguinte expressão da função de perda, junto com o multiplicador de Lagrange único  $\lambda$  tem que ser minimizada

$$-L(\boldsymbol{\theta}) - \lambda \left( -1 + \sum_{\ell=1}^L \pi_\ell \right),$$

o que resulta finalmente em

$$\pi_\ell = \frac{n_\ell}{n}. \quad (7.50d)$$

As regras para obter os parâmetros tem do lado esquerdo o parâmetro desejado, porém do lado direito, oculto nas responsabilidades (eq. 7.46), novamente os parâmetros. Por exemplo, expandindo a (eq. 7.50d) pela definição da responsabilidade (eq. 7.46), e fração de amostras (eq. 7.50b), temos

$$\pi_\ell = \frac{1}{n} \sum_{i=1}^n \frac{\pi_\ell p(\mathbf{x}_i; \boldsymbol{\theta}_\ell)}{\sum_{\ell=1}^L \pi_\ell p(\mathbf{x}_i; \boldsymbol{\theta}_\ell)},$$

ou seja, temos dos dois lados a incógnita  $\pi_\ell$ . Infelizmente não existe uma maneira de forma fechada, como foi possível com uma única Gaussiana no exemplo 7.9. Temos que resolver um sistema de equações não lineares acopladas de uma maneira não determinística, usando um algoritmo iterativo. O sistema de equações é composto por (eq. 7.50a), (eq. 7.50c) e (eq. 7.50d). O caminho será a inicialização de todos os parâmetros  $\boldsymbol{\theta}$  procurados no 0-ésimo passo como  $\boldsymbol{\theta}^{(0)}$ , e consecutivamente atualizar o  $(k+1)$ -ésimo conjunto de parâmetros  $\boldsymbol{\theta}^{(k+1)}$ , baseado no conjunto atual  $\boldsymbol{\theta}^{(k)}$ . Nesta nomenclatura, o índice de sequência  $k$  é posicionado como sobrescrito entre parênteses, considerando a sequência gerada  $\boldsymbol{\theta}^{(0)} \rightarrow \boldsymbol{\theta}^{(1)} \rightarrow \dots \rightarrow \boldsymbol{\theta}^{(k)} \rightarrow \boldsymbol{\theta}^{(k+1)} \rightarrow \dots$ .

A solução adotada na aprendizado de máquina é uma especialização de um método chamado Esperança-Maximização (*Expectation-Maximization, EM*), composto por duas fases que dão origem ao nome do método. Referências relevantes são [10, 22, 23, 71]. A ideia básica é definir uma variável aleatória observável e uma variável aleatória composta pelas variáveis observáveis, junto com variáveis latentes (ocultas). No caso da mistura de Gaussianas, a variável aleatória observável é a amostra  $\mathbf{x}$ , a variável aleatória latente a informação qual componente gerou a amostra. Essa informação pode ser modelada no caso de um modelo mistura através de um vetor de pertença que a dimensão igual ao número de componentes. Por exemplo  $\mathbf{z}_i = [z_{i,1} \ z_{i,2} \ \dots \ z_{i,L}] = [0 \ 1 \ 0 \ \dots \ 0]$  indica que a  $i$ -ésima amostra  $\mathbf{x}_i$  foi gerada pela segunda componente do modelo mistura. Esse vetor segue a codificação *One-Hot* da (eq. 2.6) na pág. 33, porém a associação não é a alguma classe, mas a alguma componente em um problema de aprendizagem não supervisionada. A teoria do EM como método universal envolve conceitos avançados de estatística. A aplicação ao problema da estimativa de parâmetros ótimos em um modelo GMM de mistura de Gaussianas é somente uma instanciação desse conceito. Apresenta-se aqui a sequência de passos para atualizar os parâmetros na transição  $\boldsymbol{\theta}^{(k)} \rightarrow \boldsymbol{\theta}^{(k+1)}$ . No estágio E do EM as responsabilidades (eq. 7.46) de cada componente são atualizados, usando o conjunto atual de parâmetros  $\boldsymbol{\theta}^{(k)}$ . No estágio M do EM, os restantes parâmetros, separadamente para cada componente  $\boldsymbol{\theta}_\ell^{(k+1)}$  são atualizados. Enquanto o método EM em geral não tem a estrutura bem definida de um algoritmo, também criticado na comunicação original [23], a especialização do EM para o caso da mistura de Gaussianas GMM tem.

No algoritmo 13, a estrutura do algoritmo é apresentada. Na linha 1 todos os parâmetros são inicializados. O algoritmo de aglomeração  $k$ -means, apresentado posteriormente, pode ser usado para esse fim. Temos que fixar a quantidade  $L$  de componentes e a posição inicial das  $L$  médias  $\boldsymbol{\mu}_\ell$ . O  $k$ -means converge para uma configuração, em que as médias ficam estacionárias após um determinado tempo, então pode-se determinar quais das amostras  $\mathbf{x}_i$  são mais próximas de uma média. Assim, também é possível obter uma matriz de covariância dessas amostras associadas a uma média. O  $k$ -means tem uma associação fixa de uma amostra a uma média, ao contrário do EM-GMM que faz uma associação nebulosa em termos de uma probabilidade definida pela responsabilidade. Na linha 2 usam-se critérios de semelhança para parar o algoritmo. Principalmente pode-se medir a diminuição da função de perda, ou, equivalentemente, o aumento (do logaritmo)

da verossimilhança, ou seja

$$\begin{aligned}\Delta L^{(k)} &= -\Delta \ell^{(k)} \stackrel{\text{def}}{=} L(\boldsymbol{\theta}^{(k+1)}) - L(\boldsymbol{\theta}^{(k)}) \\ &= -\sum_{i=1}^n \ln \sum_{\ell=1}^L \pi_{\ell}^{(k+1)} p(\mathbf{x}_i; \boldsymbol{\mu}_{\ell}^{(k+1)}, \Sigma_{\ell}^{(k+1)}) + \sum_{i=1}^n \ln \sum_{\ell=1}^L \pi_{\ell}^{(k)} p(\mathbf{x}_i; \boldsymbol{\mu}_{\ell}^{(k)}, \Sigma_{\ell}^{(k)}).\end{aligned}$$

Se este ganho de verossimilhança for muito pequeno, ou seja  $\Delta \ell^{(k)} < \text{limiar}$ , é provável que o algoritmo tenha收敛ido, e podemos abortar a busca por melhores parâmetros. A verossimilhança pode ser inicializada pelo valor zero. O limiar pode ser por exemplo 0.001.

---

**Algoritmo 13:** EM-GMM: Método Esperança-Maximização aplicado à Mistura de Gaussianas

---

**Entrada:** Conjunto de Dados de  $n$  padrões  $\mathcal{D} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ ; Número máximo de iterações  $k_{\max}$ ; Limiares de semelhança da verossimilhança  $\mathcal{L}(\boldsymbol{\theta})$  e limiares de semelhança dos parâmetros  $\boldsymbol{\theta}$ ;

**Resultado:** Conjunto de parâmetros otimizados  $\boldsymbol{\theta}^*$  que maximizam a verossimilhança, composto das responsabilidades, médias, matrizes de covariância e coeficientes de mistura de cada uma das  $L$  componentes;

**1 Inicialização:** Inicialize na iteração  $k = 0$  a quantidade  $L$  de componentes, os coeficientes de mistura  $\pi_{\ell}^{(0)}$ , as médias  $\boldsymbol{\mu}_{\ell}^{(0)}$  e as matrizes de covariância  $\Sigma_{\ell}^{(0)}$  de todas as componentes de mistura,  $\ell = 1, \dots, L$ ;

**2 enquanto** ( $k < k_{\max}$ ) **E duas verossimilhanças consecutivas não forem parecidas faça**

3   **Estágio E:** Calcule as responsabilidades (eq. 7.46)

4    $r_{i,\ell}^{(k)} \leftarrow \frac{\pi_{\ell}^{(k)} p(\mathbf{x}_i; \boldsymbol{\mu}_{\ell}^{(k)}, \Sigma_{\ell}^{(k)})}{\sum_{\ell=1}^L \pi_{\ell}^{(k)} p(\mathbf{x}_i; \boldsymbol{\mu}_{\ell}^{(k)}, \Sigma_{\ell}^{(k)})}$ , usando os parâmetros da Gaussiana multivariável

$$p(\mathbf{x}_i; \boldsymbol{\mu}_{\ell}^{(k)}, \Sigma_{\ell}^{(k)}) = [(2\pi)^m |\Sigma_{\ell}^{(k)}|]^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2} (\mathbf{x}_i - \boldsymbol{\mu}_{\ell}^{(k)})^\top \Sigma_{\ell}^{(k)^{-1}} (\mathbf{x}_i - \boldsymbol{\mu}_{\ell}^{(k)}) \right\};$$

5   **Estágio M:** Maximize a verossimilhança, atualizando os parâmetros

$$6 \quad n_{\ell}^{(k)} \leftarrow \sum_{i=1}^n r_{i,\ell}^{(k)}$$

$$7 \quad \boldsymbol{\mu}_{\ell}^{(k+1)} \leftarrow \frac{1}{n_{\ell}^{(k)}} \sum_{i=1}^n r_{i,\ell}^{(k)} \mathbf{x}_i$$

$$8 \quad \Sigma_{\ell}^{(k+1)} \leftarrow \frac{1}{n_{\ell}^{(k)}} \sum_{i=1}^n r_{i,\ell}^{(k)} (\mathbf{x}_i - \boldsymbol{\mu}_{\ell}^{(k+1)}) (\mathbf{x}_i - \boldsymbol{\mu}_{\ell}^{(k+1)})^\top$$

$$9 \quad \pi_{\ell}^{(k+1)} \leftarrow \frac{n_{\ell}^{(k)}}{n}$$

10   **Avaliação da verossimilhança:** Calcule a função de perda  $L(\boldsymbol{\theta})$  da (eq. 7.49)

(logaritmo negativa da verossimilhança) do modelo GMM ;

11    $k \leftarrow k + 1$ ;

12 **fim**

13  $\boldsymbol{\theta}^* \leftarrow \boldsymbol{\theta}^{(k)}$

---

■ **Exemplo 7.10** Considere um exemplo ilustrativo com uma única dimensão das amostras, ou seja,  $x_i \in \mathbb{R}$ . Seis amostras são aleatoriamente geradas, formando a matriz de dados de uma única coluna que após ordenação em ordem crescente têm os valores

$$X = [x_1 \quad \cdots \quad x_6]^\top = [0.208 \quad 0.226 \quad 0.312 \quad 0.337 \quad 0.551 \quad 0.751]^\top.$$

Em relação à inicialização, linha 1 sejam dadas as seguintes configurações: A posição das médias iniciais  $\mu_1^{(0)}$  e  $\mu_2^{(0)}$  não é inicializada pelo algoritmo de aglomeração  $k$ -means. Caso contrário, a inicialização seria “boa demais”, praticamente igual ao resultado final da posição das médias. Duas componentes formam a mistura de Gaussianas, ou seja,  $L = 2$ . Os dois coeficientes de mistura são inicializados aleatoriamente na 0-ésima iteração,  $k = 0$  como  $\pi_1^{(0)} = 0.56478$  e  $\pi_2^{(0)} = 0.43522$ . Como temos somente uma única dimensão, o vetor de média é um escalar com  $\mu_1^{(0)} = 0.403$  e  $\mu_2^{(0)} = 0.390$  e a matriz de covariância simplifica para uma variância, com  $\Sigma_1^{(0)} = \sigma_1^2 = 0.0327$  e  $\Sigma_2^{(0)} = \sigma_2^2 = 0.0437$ .

Vamos executar o primeiro estágio E do EM da linha 3, calculando as responsabilidades (eq. 7.46) que associam cada amostra às duas componentes. Eles formam uma matriz  $n \times L = 6 \times 2$ . Como valor representativo vamos explicitar somente o numerador da responsabilidade  $r_{1,2}^{(0)}$  do primeiro padrão  $x_1$  na segunda componente, ou seja,

$$\begin{aligned}\pi_2^{(0)} \cdot p(x_1; \mu_2^{(0)}, \sigma_2^2) &= 0.435 \cdot p(0.208; 0.390, 0.0437) \\ &= 0.435 \cdot \frac{1}{\sqrt{2\pi \cdot 0.0437}} \exp \left\{ -\frac{(0.208 - 0.390)^2}{2 \cdot 0.0437} \right\} \\ &= 0.435 \cdot 1.304 = 0.567.\end{aligned}$$

Dividindo ainda pelo denominador que é o fator normalizador, a responsabilidade resulta no valor  $r_{1,2}^{(0)} = 0.423$  que é a segunda coluna da primeira linha da matriz de responsabilidades

$$r_{i,\ell}^{(0)} = \begin{bmatrix} 0.577 & 0.423 \\ 0.381 & 0.619 \\ 0.580 & 0.420 \\ 0.584 & 0.416 \\ 0.847 & 0.153 \\ 0.420 & 0.580 \end{bmatrix}, \quad i = 1, \dots, 6; \quad \ell = 1, 2.$$

Pela inicialização arbitrária, as frações de responsabilidade da (eq. 7.50b) de cada componente não são valores redondos,  $n_1^{(0)} = 3.389$  e  $n_2^{(0)} = 2.611$ , veja a linha 6. Nos restantes cálculos do estágio M do EM, as médias, matrizes de covariância e coeficientes de mistura são atualizados, resultando em

$$\begin{aligned}\mu_1^{(1)} &= \frac{1}{n_1^{(0)}} \sum_{i=1}^6 r_{i,1}^{(0)} \mathbf{x}_i = 0.3932, \quad \mu_2^{(1)} = \frac{1}{n_2^{(0)}} \sum_{i=1}^6 r_{i,2}^{(0)} \mathbf{x}_i = 0.4032 \\ \sigma_1^2 &= \frac{1}{n_1^{(0)} - 1} \sum_{i=1}^6 r_{i,1}^{(0)} (\mathbf{x}_i - \mu_1^{(1)}) (\mathbf{x}_i - \mu_1^{(1)})^\top = 0.0356 \\ \sigma_2^2 &= \frac{1}{n_2^{(0)} - 1} \sum_{i=1}^6 r_{i,2}^{(0)} (\mathbf{x}_i - \mu_2^{(1)}) (\mathbf{x}_i - \mu_2^{(1)})^\top = 0.0400 \\ \pi_1^{(1)} &= \frac{n_1^{(0)}}{6} = \frac{3.389}{6} = 0.56475, \quad \pi_2^{(1)} = \frac{n_2^{(0)}}{6} = \frac{2.611}{6} = 0.43522.\end{aligned}$$

Finalmente o cálculo do logaritmo da função de verossimilhança, (eq. 7.40) na linha 10 resulta em

$$\begin{aligned}\ell(\boldsymbol{\theta}^{(1)} | \mathcal{D}) &= \ell(\boldsymbol{\theta}^{(1)} | \{x_1, \dots, x_6\}) = \sum_{i=1}^6 \ln \mathcal{L}(\boldsymbol{\theta}^{(1)} | x_i) = \sum_{i=1}^6 \ln p(x_i | \boldsymbol{\theta}^{(1)}) \\ &\stackrel{7.32}{=} \sum_{i=1}^6 \ln \left[ \pi_1^{(1)} \cdot p(x_i; \mu_1^{(1)}, \sigma_1^2) + \pi_2^{(1)} \cdot p(x_i; \mu_2^{(1)}, \sigma_2^2) \right] = 0.218.\end{aligned}$$

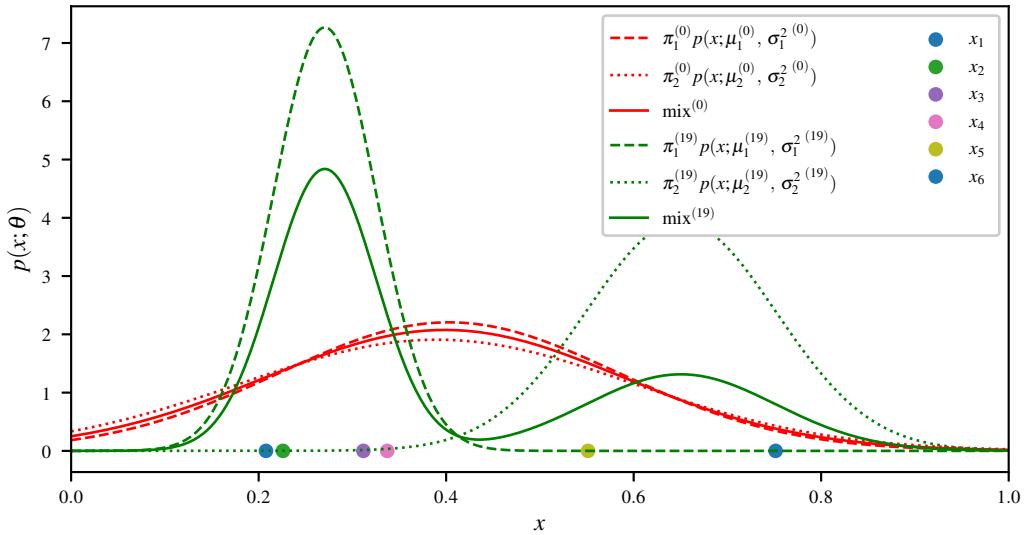


Figura 7.10: Modelo de mistura de Gaussianas antes e após a aprendizagem do modelo pelo algoritmo EM-GMM. Após a aprendizagem, as responsabilidades de cada componente em relação às seis amostras são claramente distinguíveis.

Com um limiar de semelhança igual a  $10^{-5}$ , na iteração  $k = 19$ , o módulo da diferença entre o logaritmo da penúltima e última verossimilhança é

$$\begin{aligned} |\ell(\boldsymbol{\theta}^{(19)}|\mathcal{D}) - \ell(\boldsymbol{\theta}^{(18)}|\mathcal{D})| &= |0.6478546581790326 - 0.6478605428714141| \\ &= |0.0000058847| = 5.8847 \cdot 10^{-6}, \end{aligned}$$

o que causa o fim do algoritmo. Os parâmetros finais são

$$\begin{aligned} r_{i,\ell}^{(19)} &= \begin{bmatrix} 0.999961 & 0.000039 \\ 0.999939 & 0.000061 \\ 0.998619 & 0.001381 \\ 0.995145 & 0.004855 \\ 0.000012 & 0.999988 \\ 0.0 & 1.0 \end{bmatrix} \approx \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}, \quad i = 1, \dots, 6; \ell = 1, 2 \\ \mu_1^{(19)} &= 0.270567, \quad \mu_2^{(19)} = 0.650375 \\ \sigma_1^2{}^{(19)} &= 0.003014, \quad \sigma_2^2{}^{(19)} = 0.010299 \\ \pi_1^{(19)} &= 0.665613, \quad \pi_2^{(19)} = 0.334387. \end{aligned}$$

Além disso, agora as frações de responsabilidade da (eq. 7.50b) de cada componente estão bem definidas, quase redondas, com  $n_1^{(19)} = 3.993676 \approx 4$  e  $n_2^{(19)} = 2.006324 \approx 2$ . A figura 7.10 mostra as duas misturas antes e depois da execução do algoritmo 13. De cada mistura, as duas Gaussianas que a compõem também são exibidas, porém não multiplicadas pelo seus respectivos coeficientes de mistura. ■

■ **Exemplo 7.11** A figura 7.11 mostra os vários estados na organização das componentes em um espaço bidimensional. Os dados são os  $n = 150$  flores iris, porém sem qualquer rótulo de classe. As duas características são  $x_2$ , largura da sépala e  $x_4$ , largura da pétala. Quatro componentes

são arbitrariamente geradas. Seis estágios são ilustrados, após a primeira iteração, ou seja, após  $k = 1, 2, 5, 10, 20, 100$  no algoritmo 13. Novamente, cada elipse representa uma componente. Os dois semieixos correspondem ao maior e segundo maior autovalor da matriz de covariância, reveja a figura 4.2 na pág. 134 e figura 7.6 na pág. 205. Neste exemplo duas componentes conseguem modelar relativamente bem, o que era a classes setosa, pois fica bastante separado do resto, porém constata-se uma certa redundância, pois uma componente praticamente engloba a outra. As outras duas componentes tentam modelar o resto dos dados, originalmente versicolor e virginica, porém pode-se ver um desequilíbrio em relação a quantidade de padrões associados a essas duas componentes. ■

Lembre que o modelo mistura definição 7.4.2 na pág. 204 não está limitado a uma densidade Gaussiana, nem o método Esperança-Maximização.

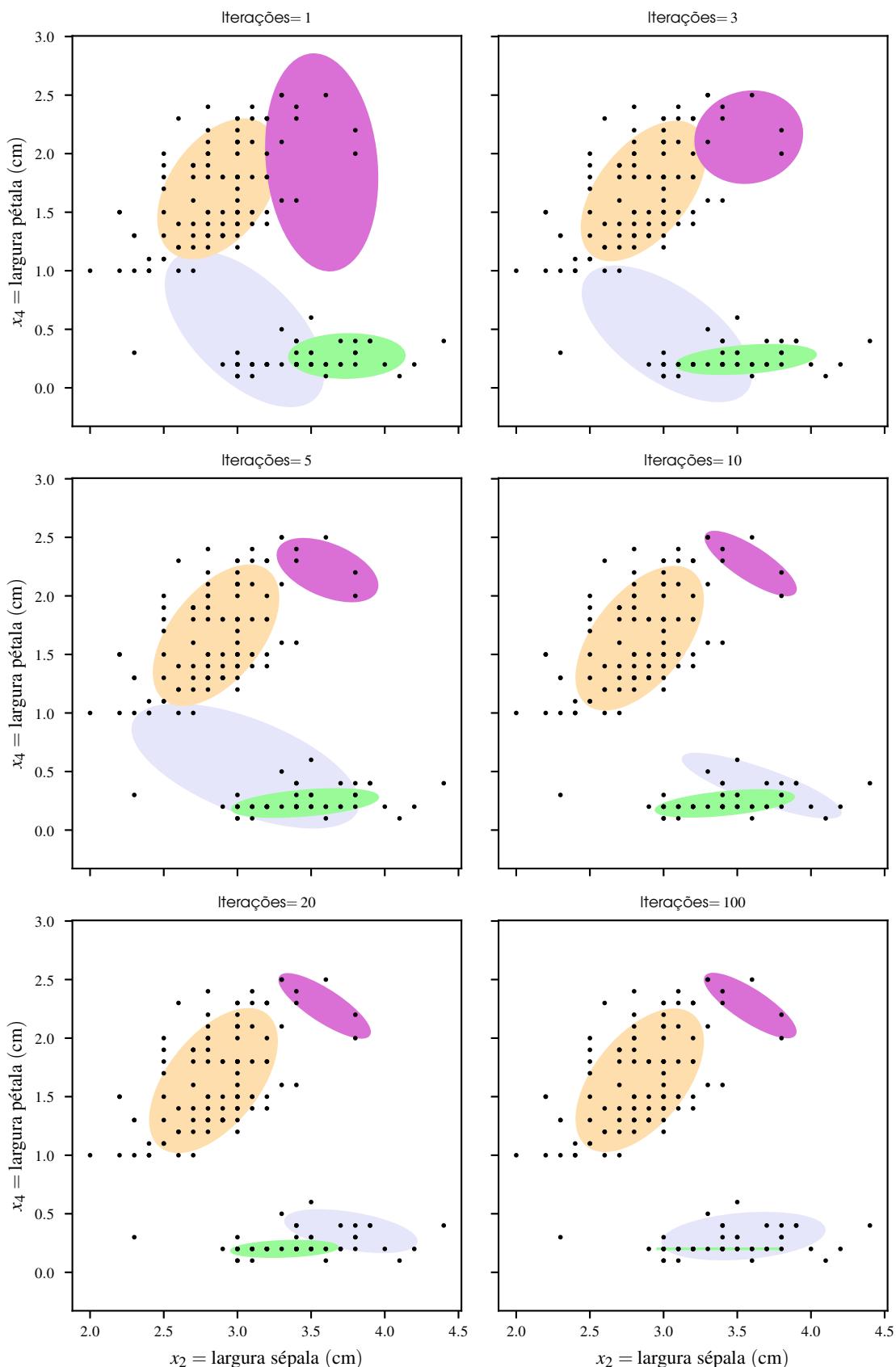


Figura 7.11: Modelo de mistura de quatro Gaussianas em duas dimensões em vários estágios da aprendizagem do modelo pelo algoritmo EM-GMM.

## 8. Classificação Não-Paramétrica, Vizinho-Mais-Próximo

A desvantagem de modelos paramétricos é a necessidade de definir uma distribuição probabilística e estimar os seus parâmetros. Isso são duas fontes de erro potenciais que podem deteriorar o desempenho do sistema consideravelmente. Se os dados não forem originais da distribuição suposta, temos um erro de modelo. Se os dados realmente são de um suposto modelo, porém a estimativa tiver um viés grande em relação aos verdadeiros parâmetros, o modelo paramétrico também não vai funcionar de uma maneira ótima. Como exemplo assume que a verdadeira distribuição dos dados seja Gaussiana, e os parâmetros corretos estejam disponíveis. Somente assim os limites teóricos da regra de Bayes podem ser atingidos, por exemplo uma taxa de erro mínima. Esses critérios de desempenho ainda vamos discutir em um capítulo próprio.

### 8.1 K-Vizinhos-Mais-Próximos

O modelo não paramétrico mais conhecido e mais simples é baseado na distância entre os padrões. Mesmo assim, esse modelo nunca pode faltar em um estudo comparativo com outras técnicas, especialmente em problemas de classificação. Isso é devido a um conjunto de propriedades desejáveis desta técnica, nomeadamente a ausência de hiperparâmetros do modelo básico 1-NN e a modelagem de qualquer distribuição com uma taxa de erro limitada. Em breve discutiremos essas propriedades. O K-Vizinhos-Mais-Próximos, (*K-Nearest-Neighbors*, KNN) [18] pode ser usado para classificar e para estimar uma densidade. Apresenta-se aqui somente o primeiro propósito.

#### 8.1.1 1-Vizinho-Mais-Próximo

Na sua forma básica, o classificador 1-NN mede a distância Euclidiana entre um padrão a ser classificado  $\mathbf{x}$  e todos os  $n$  padrões  $\mathbf{x}_i$  do conjunto de treinamento. Um deles é mais perto do padrão ainda sem classe  $\mathbf{x}$  do que os restantes ( $n - 1$ ) padrões. Consequentemente, podemos assumir que a classe conhecida do vizinho mais próximo coincide com a classe desconhecida do padrão a ser classificado  $\mathbf{x}$ . Assim, já podemos formular um método básico para classificar no algoritmo 14.

Formalmente podemos definir a função discriminativa definição 2.3.3 na pág. 62 *F* que, dado

**Algoritmo 14:** Classificação pelo 1-Vizinho-Mais-Próximo

**Entrada:** Conjunto de dados de treinamento de  $n$  padrões  $\mathcal{D} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ , Padrão desconhecido  $\mathbf{x}$

**Resultado:** Classe  $\mathcal{C}_p$  a qual pertence o padrão  $\mathbf{x}$  que tem a menor distância do padrão desconhecido  $\mathbf{x}$

**Treinamento:**

Faça nada !;

**Classificação:**

$d^* \leftarrow \infty$  /\* Inicialize a menor distância com o valor infinito \*/ ;

```

1 para  $i = 1, \dots, n$  faça
2   Meça a distância Euclidiana  $d_i \leftarrow \|\mathbf{x} - \mathbf{x}_i\|$ , com  $\mathbf{x}_i \in \mathcal{C}_p$ ;
3   se  $d_i < d^*$  então
4      $d^* \leftarrow d_i$ ;
5      $\mathcal{C}_p \leftarrow$  Classe do padrão  $\mathbf{x}_i$ ;
6   fim
7 fim

```

um padrão  $\mathbf{x}$  decide a classe  $\mathcal{C}_p$  a qual o padrão com a menor distância ao menor

$$F(\mathbf{x}) = \mathcal{C}_p, \quad q = \arg \min_{i=1}^n \|\mathbf{x} - \mathbf{x}_i\|, \quad \mathbf{x}_q \in \mathcal{C}_p. \quad (8.1)$$

Uma das propriedades notáveis deste classificador é que não existe um estágio de treinamento. O conhecimento sobre as classes dos padrões existentes e que surjam eventualmente para serem classificados, está unicamente embutido nos padrões. Dessa maneira revela-se automaticamente uma desvantagem do método, pois para determinar qual é o mais próximo, *todas* as  $n$  distâncias têm que ser calculadas. Em caso de um conjunto grande de treinamento, pode atrasar a classificação.

A figura 8.1 foca em alguns padrões de duas classes de iris no espaço das primeiras duas características. O padrão mais próximo do desconhecido fica a esquerda, um pouco acima de uma distância de  $d = 0.042$ , e pertence a classe versicolor. Consequentemente o resultado do classificador é a classe versicolor. Essa decisão corresponde a primeira linha da tabela 8.1. Note que no total, não somente as onze distâncias dos padrões visíveis na figura 8.1 são calculadas, mas todas as  $n = 150$  padrões de iris, mesmo os não visíveis, veja os invisíveis na figura 1.2a na pág. 15.

### 8.1.2 Classificação de K-Vizinhos-Mais-Próximos

Considerando mais que um dos vizinhos mais perto do desconhecido  $\mathbf{x}$  resulta em um *ensemble de classificadores* mais simples imaginável. O mais próximo dá o seu voto da sua classe, o segundo mais próximo dá o voto da sua classe que pode ser a mesma do anterior ou não, o terceiro mais próximo dá o seu voto da sua classe, e assim por diante. A quantidade de padrões que cada classe tiver mais próximo do desconhecido poderia ser considerada a função de *score* para cada classe, veja definição 2.3.1 na pág. 61. Se a decisão da votação for por maioria, temos o classificador dos K-Vizinhos-Mais-Próximos. Para  $K = 1, 2, 3, 4, 5$ , o classificador KNN, e o resultado são listados na tabela 8.1. Em caso de empate, outros critérios têm que ajudar na decisão final, por exemplo a probabilidade a priori de cada classe, ou a própria distância. Por exemplo, se dos quatro vizinhos dois pertencem a duas classes diferentes, a menor distância média de cada par, até o padrão desconhecido pode resolver o empate. Números pares de  $K$  normalmente são omitidos, pois a probabilidade de empate é mais alta.

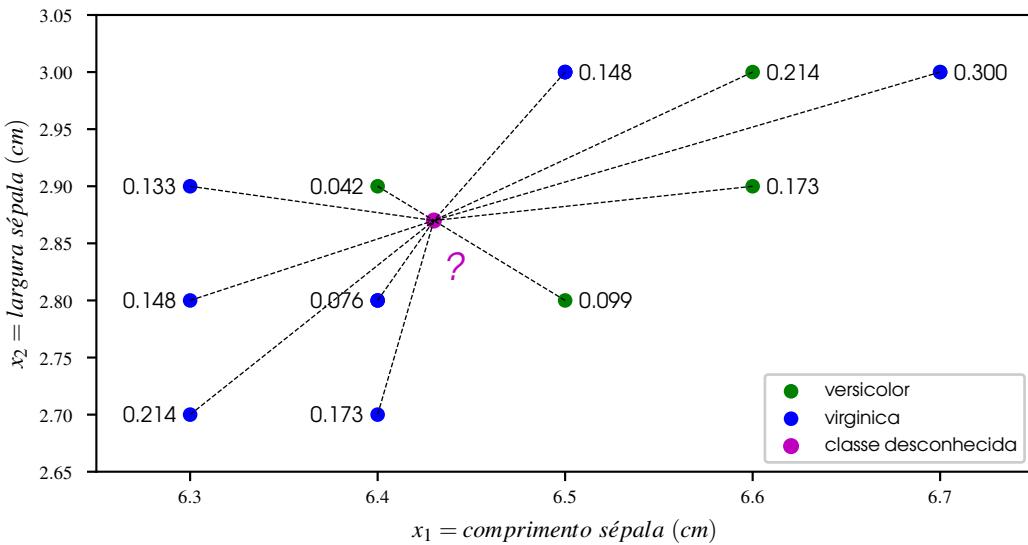


Figura 8.1: Onze padrões das duas classes de iris virgínia e versicolor com rótulo da classe e distância Euclidiana de um padrão desconhecido  $\mathbf{x} = [6.43 \quad 2.87]^\top$ .

Ordem na proximidade	Distância Euclidiana	Classe	Classificador resultante	Proporção (= Função de score ) virginica / versicolor	Situação	Resultado classificação
1	0.042	versicolor	1-NN	0 1	Maioria versicolor	versicolor
2	0.076	virginica	2-NN	1 1	Empate	indefinido
3	0.099	versicolor	3-NN	1 2	Maioria versicolor	versicolor
4	0.133	virginica	4-NN	2 2	Empate	indefinido
5	0.148	virginica	5-NN	3 2	Maioria virginica	virginica

Tabela 8.1: Classificação do padrão  $\mathbf{x} = [6.43 \quad 2.87]^\top$  pelo classificador não paramétrico do K-Vizinhos-Mais-Próximos para  $K = 1, \dots, 5$ .

A figura 8.2 mostra as regiões de decisão resultantes de um classificador 1-Vizinho-Mais-Próximo. Para dificultar a tarefa de classificar, novamente duas classes de iris foram hipoteticamente fusionadas, formando regiões não contíguas. Compare com a figura 7.6 na pág. 205, onde a mistura de Gaussianas foi usada para modelar a distribuição de dados. O classificador 1-Vizinho-Mais-Próximo consegue separar muito bem as classes, embora saiba nada sobre a distribuição relativamente complicada de cada classe. Essa propriedade de implicitamente obedecer a qualquer densidade torna esse classificador simples uma ferramenta importante. Veja que especialmente na região de sobreposição dos padrões, a fronteira é muito bem aproximada.

### 8.1.3 Aproximação de Probabilidade a Posteriori Bayesiana pelo K-Vizinhos-Mais-Próximos

Nesta seção fica claro, por que o KNN consegue aproximadamente modelar qualquer densidade. Na estatística um *histograma* faz uma contagem de amostras dentro de uma *barra*. A variável aleatória contínua  $x$  em consideração no caso básico é unidimensional, ou seja  $x \in \mathbb{R}$ . Essa variável está sujeita a uma distribuição probabilística, expressa na sua densidade  $p(x)$ . Essa densidade pode assumir qualquer forma, não somente as funções bem comportadas, como a Gaussiana, por exemplo. Uma barra é um pequeno intervalo  $V = [a, b]$  onde se faz a contagem quantas amostras caem nesta barra. Por exemplo, na figura 7.10 na pág. 216 temos quatro amostras no intervalo  $V = [0.2, 0.4]$ . Seja  $n$  a quantidade total de amostras, e  $K_j$  a quantidade de amostras que caem na

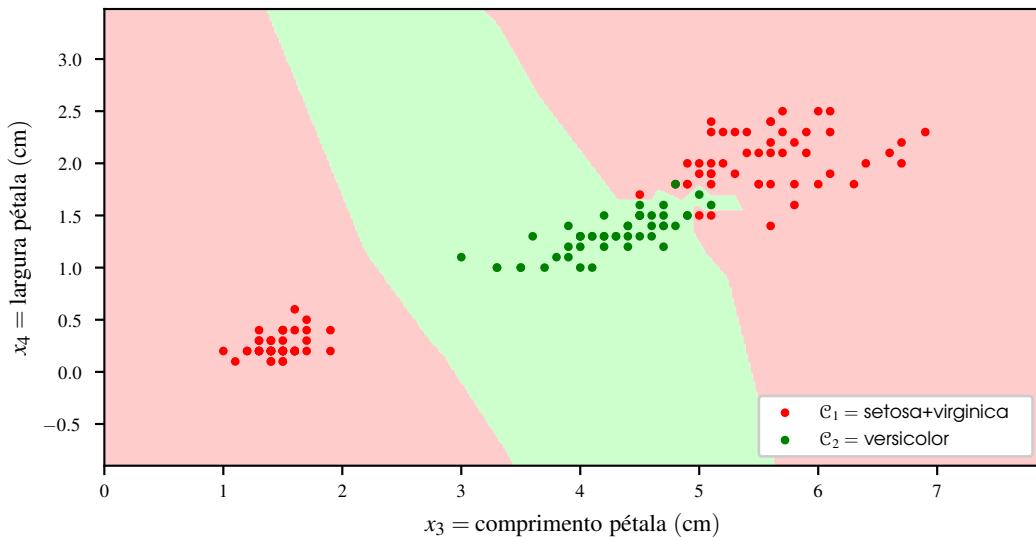


Figura 8.2: Regiões de decisão definidos pelo 1-Vizinho-Mais-Próximo, onde duas classes de iris foram fusionadas para dificultar a classificação.

$j$ -ésima barra  $V_j$  de um total de  $B$  barras da mesma largura. Por exemplo, novamente na figura 7.10, se definimos  $n = 3$  intervalos  $V_1 = [0.2, 0.4]$ ,  $V_2 = [0.4, 0.6]$ ,  $V_3 = [0.6, 0.8]$ , temos  $K_1 = 4$ ,  $K_2 = 1$ ,  $K_3 = 1$ . De fato,  $K_j$  é uma função que faz a contagem das amostras das respectiva barra. Então a seguinte condição é satisfeita.

$$n = \sum_{j=1}^B K_j \iff 1 = \sum_{j=1}^B \frac{K_j}{n}. \quad (8.2)$$

No exemplo, temos  $1 = 4/6 + 1/6 + 1/6$ . Assim fica plausível que a fração de amostras que caem no intervalo  $K_j$  são uma aproximação da probabilidade que qualquer amostra cai nesse intervalo, dado o fato que as probabilidades que são mutuamente exclusivas, somam o valor 1. Assim

$$P(x \in V_j) \approx \frac{K_j}{n}. \quad (8.3)$$

Se especializarmos essa probabilidade para cada classe em um problema de classificação, temos para essa probabilidade, condicionada ao fato que  $x$  faz parte da classe  $\mathcal{C}_k$ , a aproximação

$$P(x \in V_j | \mathcal{C}_k) \approx \frac{K_{j,k}}{n_k}, \quad (8.4)$$

onde  $K_{j,k}$  é a quantidade de amostras da  $k$ -ésima classe  $\mathcal{C}_k$  que caem na barra  $V_j$  e  $n_k$  é a quantidade total de amostras que pertencem à classe  $\mathcal{C}_k$ . Estamos interessados na probabilidade a posteriori que uma amostra pertence a  $k$ -ésima classe, dado o fato que ela faz parte a  $j$ -ésima barra  $V_j$ . Pela regra de Bayes, (eq. 7.4) na pág. 188 temos

$$P(\mathcal{C}_k | x \in V_j) = \frac{p(x | \mathcal{C}_k)P(\mathcal{C}_k)}{p(x)}. \quad (8.5)$$

Já sabemos que a probabilidade a priori de cada classe pode ser simplesmente ser estimado pela fração das amostras da classe em relação ao número total de amostras como

$$P(\mathcal{C}_k) = \frac{n_k}{n}. \quad (8.6)$$

Falta obter uma estimativa para a densidade  $p(x)$ . A definição formal<sup>1</sup> da probabilidade que um padrão cai dentro do intervalo é

$$P(x \in V_j) = \int_{V_j} p(\xi) d\xi \approx p(x)V_j, \quad (8.7)$$

ou seja

$$p(x) \approx \frac{P(x \in V_j)}{V_j}. \quad (8.8)$$

A densidade específica da classe então é

$$p(x|\mathcal{C}_k) \approx \frac{P(x \in V_j|\mathcal{C}_k)}{V_j}. \quad (8.9)$$

onde  $x$  é um ponto qualquer no interior da barra. Repare que aqui foi usada a regra mais simples possível de aproximar uma integral por um retângulo da largura da barra vezes a altura da barra, onde a altura é a densidade  $p(x)$  de algum ponto no interior da barra. Inserindo a (eq. 8.9) e a (eq. 8.8) na regra de Bayes (eq. 8.5), temos

$$P(\mathcal{C}_k|x \in V_j) = \frac{P(x \in V_j|\mathcal{C}_k)P(\mathcal{C}_k)}{V_j} \frac{V_j}{P(x \in V_j)} = \frac{P(x \in V_j|\mathcal{C}_k)P(\mathcal{C}_k)}{P(x \in V_j)}. \quad (8.10)$$

Inserindo a (eq. 8.3) e (eq. 8.4) na (eq. 8.10), temos finalmente

$$P(\mathcal{C}_k|x \in V_j) = \frac{\frac{K_{j,k}}{n_k} \frac{n_k}{n}}{\frac{K_j}{n}} = \frac{K_{j,k}}{K_j}. \quad (8.11)$$

Em outras palavras, a probabilidade de um padrão  $x$  que fica dentro da região  $V_j$  pertencer a uma classe  $\mathcal{C}_k$ , pode ser estimado pela fração da quantidade de amostras que pertencem a  $k$ -ésima classe em relação a quantidade total de amostras que se encontram na região. Essa quantidade total são exatamente os  $K$  vizinhos mais próximos da amostra  $x$ . A região  $V_j$  é definida pelo padrão a ser classificado  $x$ . Se usarmos  $K = 5$  vizinhos, a região em torno do padrão  $x$  com classe incógnita tem que expandido até abrigar cinco padrões. No caso multidimensional temos que aumentar o raio da hiperesfera, em duas dimensões, o raio do círculo com  $x$  como centro.

Obviamente neste raciocínio foram usadas algumas aproximações e simplificações, mas mesmo assim, o classificador K-Vizinhos-Mais-Próximos justifica seu poder pela forte relação com a regra de Bayes, sem definir por modelo e seus parâmetros a densidade que rege os dados. Por isso, é denominado um classificador não paramétrico.

#### 8.1.4 A Inequação de Cover & Hart

Uma outra observação sobre a taxa de erro que um classificador KNN comete é a inequação de Cover e Hart [18], comentada em [25]. A taxa de erro ainda temos que definir formalmente, mas é fácil de entender esse conceito. Se de 500 padrões apresentados a um classificador 100 forem classificados erradamente, a taxa de erro é  $100/500 = 20\%$ .

**Definição 8.1.1 — Inequação de Cover e Hart.** Dado um número de amostras teoricamente infinita  $n \rightarrow \infty$ , em um problema de classificação binária, ou seja, com duas classes, a taxa de erro  $R$  cometida pelo classificador 1-Vizinho-Mais-Próximo é limitado pelos seguintes limites

$$R^* \leq R \leq 2R^*(1 - R^*), \quad (8.12)$$

<sup>1</sup>Foi usada uma variável auxiliar  $\xi$  na integração, pois não pode ser usado o mesmo símbolo  $x$ .

onde  $0 \leq R^* \leq \frac{1}{2}$  é a probabilidade de erro de Bayes.

A probabilidade de erro de Bayes é o limite teórico abaixo do qual não é possível chegar. Posteriormente, na apresentação de critérios de desempenho de um classificador, retornaremos a esse assunto. A importante lição desta inequação é que o desempenho desta classificador não ultrapassa duas vezes o limite teórico. Assim torna-se possível fazer afirmações sobre a qualidade de dados. Se, com muitas amostras, o 1-NN estima 30 % de erro, não podemos esperar uma taxa de erro perto de zero, seja qual for o classificador usado.

### 8.1.5 Métricas de Distância

Neste ponto podemos considerar alternativas da maneira como a distância entre os padrões é calculada. Na grande maioria dos casos, a distância Euclidiana é usada para medir a proximidade entre dois padrões. Existem outras métricas que eventualmente sejam mais apropriadas, dependendo do problema em consideração. Métrica é uma definição de função de distância. Uma distância mais geral é a

**Definição 8.1.2 — Distância de Minkowski de Ordem  $p$ .** Dados dois vetores  $\mathbf{a}$  e  $\mathbf{b}$  de dimensão  $m$ , a distância de Minkowski de ordem  $p \in \mathbb{N}$  entre eles é

$$D_M^p(\mathbf{a}, \mathbf{b}) = \left( \sum_{j=1}^m |a_j - b_j|^p \right)^{\frac{1}{p}}. \quad (8.13)$$

A distância de Minkowski é equivalente à norma norma- $\ell_p$ , definida na (eq. 1.12) na pág. 14, da diferença dos dois vetores, ou seja,

$$D_M^p(\mathbf{a}, \mathbf{b}) = \|\mathbf{a} - \mathbf{b}\|_p. \quad (8.14)$$

Para  $p = 1$ , a distância de Minkowski especializa para a métrica do táxi ou distância de Manhattan ou distância de *City Block*<sup>2</sup>

$$D_M^1(\mathbf{a}, \mathbf{b}) = D_C(\mathbf{a}, \mathbf{b}) = \|\mathbf{a} - \mathbf{b}\|_1 = \sum_{j=1}^m |a_j - b_j|. \quad (8.15)$$

Para  $p = 2$ , a distância de Minkowski especializa para a distância Euclidiana

$$D_M^2(\mathbf{a}, \mathbf{b}) = D_E(\mathbf{a}, \mathbf{b}) = \|\mathbf{a} - \mathbf{b}\|_2. \quad (8.16)$$

### 8.1.6 Variações do Classificador K-Vizinhos-Mais-Próximos

A comparação de um padrão com todos os padrões do conjunto de treinamento para obter a lista dos mais próximos é computacionalmente cara. Ela cresce linearmente com a quantidade  $n$  de amostras de treinamento. Vale a pena mencionar algumas técnicas de pré-processamento que diminuem a carga computacional na fase de classificação, obviamente aumentando a carga na fase de treinamento. Estruturas de dados são formados que aceleram o processo de medir distâncias. Em seguida, porém vamos estudar um conceito que não otimiza a organização dos dados, mas realiza uma extração de características implicitamente, mapeando os padrões para um outro espaço vetorial, antes de medir a distância entre eles.

<sup>2</sup>A analogia origina no trajeto que um veículo ou pedestre tem que percorrer em duas dimensões, ao longo do caminho paralelos aos eixos do sistema de coordenadas, pois é impossível passar diagonalmente pelos prédios.

## 8.2 K-Vizinhos-Mais-Próximos com Kernel

Neste ponto do texto vamos introduzir um conceito poderoso de extração de características, o do *kernel* que pode ser aplicado não somente com o classificador K-Vizinhos-Mais-Próximos, mas também com outras arquiteturas, em especial a Máquina de Vetor de Suporte. É igualmente aplicável à análise de componentes principais, dando origem a PCA com *kernel*, a KPCA.

O KNN na sua forma usa a distância Euclidiana  $\|\mathbf{x} - \mathbf{x}_i\|$  entre um padrão a ser classificado  $\mathbf{x}$  e um padrão  $\mathbf{x}_i$  do conjunto de treinamento. A versão do KNN com *kernel* [69, 115] calcula a distância Euclidiana em um novo espaço de características, em vez de medir a distância no espaço original de características. Lembra-se aqui o corolário 1.1.4 na pág. 14 que relaciona a distância Euclidiana entre dois vetores  $\mathbf{x}$  e  $\mathbf{y}$  com uma expressão que envolve somente termos de produtos escalares

$$D_E^2(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|^2 = \mathbf{x} \cdot \mathbf{x} - 2\mathbf{x} \cdot \mathbf{y} + \mathbf{y} \cdot \mathbf{y}$$

no espaço vetorial, onde o produto escalar é definido. Foi usado aqui o quadrado da distância Euclidiana, em vez da distância Euclidiana para facilitar a expressão.

Vamos definir um novo espaço  $\mathcal{H}$ , o espaço de Hilbert de Reprodução com Núcleo, (*Reproducing Kernel Hilbert Space*, RKHS). A teoria associada para usar essa técnica leva a assuntos de matemática mais avançadas [98], porém para poder usar o método na prática no contexto de aprendizado de máquina, não precisamos nos envolver muito profundamente. Na literatura mais aprofundada também encontra-se também a explicação por que o espaço é de *reprodução*.

Vamos primeiro definir dois conceitos entrelaçados, um mapa, e um núcleo (*kernel*). O mapa é um extrator de características, já estudado na definição 4.1.2 na pág. 127. Um extrator calcula um vetor  $\phi(\mathbf{x})$  a partir da entrada do vetor de características originais  $\mathbf{x}$ . A diferença principal deste mapa como extrator em relação a um extrator de características já conhecido anteriormente, é que o resultado da extração já não está diretamente acessível em forma de vetor  $\phi(\mathbf{x})$ . O que ainda pode ser feito, é calcular um produto escalar de dois vetores mapeados, ou seja, embora os mapas  $\phi(\mathbf{x})$  e  $\phi(\mathbf{y})$  de dois vetores  $\mathbf{x}$  e  $\mathbf{y}$  não estejam acessíveis, o seu produto escalar  $\phi(\mathbf{x}) \cdot \phi(\mathbf{y})$  está. O *kernel* pode ser considerado um produto escalar no novo espaço  $\mathcal{H}$ . O conceito do RKHS vai além do contexto do classificador K-Vizinhos-Mais-Próximos. Posteriormente vamos reencontrar o *kernel* na extração de componentes principais com *kernel*, na Máquina de Vetor de Suporte, entre outros.

**Definição 8.2.1 — Mapa do Espaço Vetorial  $\mathcal{X}$  para o espaço de Hilbert de Reprodução com Núcleo  $\mathcal{H}$ .** O espaço vetorial considerado é o espaço Euclidiano  $\mathcal{X} = \mathbb{R}^m$  de dimensão  $m$ . Seja dado um vetor no espaço de características original  $\mathbf{x} \in \mathbb{R}^m$ . O mapa

$$\begin{aligned} \phi : \mathcal{X} &\rightarrow \mathcal{H} \\ \mathbf{x} &\mapsto \phi(\mathbf{x}) \end{aligned} \tag{8.17}$$

calcula um novo padrão  $\phi(\mathbf{x})$  no espaço  $\mathcal{H}$ .

Em geral, a definição explícita do mapeamento não nos interessa, e nem é possível especificar essa definição. O importante é a possibilidade de calcular um produto escalar entre dois padrões mapeados.

**Definição 8.2.2 — Núcleo (*kernel*).** Seja  $\phi$  o mapa da definição 8.2.1 que calcula um vetor  $\phi(\mathbf{x}) \in \mathcal{H}$  a partir do vetor  $\mathbf{x} \in \mathbb{R}^m$  e um outro vetor  $\phi(\mathbf{y}) \in \mathcal{H}$  a partir do vetor  $\mathbf{y} \in \mathbb{R}^m$ . Então

o núcleo (*kernel*) é o produto escalar no novo espaço  $\mathcal{H}$  dos dois vetores mapeados

$$k : \langle \cdot, \cdot \rangle : \mathcal{X} \times \mathcal{X} \rightarrow \mathcal{H} \times \mathcal{H} \rightarrow \mathbb{R}$$

$$(\mathbf{x}, \mathbf{y}) \mapsto (\phi(\mathbf{x}), \phi(\mathbf{y})) \mapsto \phi(\mathbf{x}) \cdot \phi(\mathbf{y}) \stackrel{\text{def}}{=} k(\mathbf{x}, \mathbf{y}). \quad (8.18)$$

■ **Exemplo 8.1** Vamos tentar providenciar um exemplo mínimo que ilustra o cálculo feito. Ao contrário do caso geral, neste exemplo simples, o mapa tem uma representação concreta. Isto significa que podemos fornecer uma expressão para o mapeamento  $\phi(\mathbf{x})$  do padrão original  $\mathbf{x}$  para o espaço RKHS que neste caso é fácil de se entendido, e, até ser visualizado. O espaço original de características seja o plano  $\mathbb{R}^2$ . O novo espaço de características, o espaço de Hilbert de Reprodução com Núcleo seja o espaço 3-D, ou seja,  $\mathcal{H} = \mathbb{R}^3$ . O mapa é definido explicitamente como um extrator de características comum como

$$\phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \mapsto \phi(\mathbf{x}) = \begin{bmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{bmatrix}. \quad (8.19)$$

Este mapa equivale a um extrator de características já estudado, definido na definição 4.1.2 na pág. 127. É uma combinação quadrática das componentes  $x_1$  e  $x_2$  do vetor de características  $\mathbf{x}$ . Podemos facilmente mostrar que de fato o *kernel* associado a esse mapa, para dois padrões  $\mathbf{x}$  e  $\mathbf{y}$  do espaço de características original satisfaz

$$k(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{y}).$$

*Demonstração.*

$$\begin{aligned} k(\mathbf{x}, \mathbf{y}) &= k\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}\right) \\ &= \phi(\mathbf{x}) \cdot \phi(\mathbf{y}) = \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) \cdot \phi\left(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}\right) \\ &= \begin{bmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{bmatrix} \cdot \begin{bmatrix} y_1^2 \\ y_2^2 \\ \sqrt{2}y_1y_2 \end{bmatrix} \\ &\stackrel{?}{=} x_1^2y_1^2 + x_2^2y_2^2 + \sqrt{2}x_1x_2\sqrt{2}y_1y_2 \\ &= x_1^2y_1^2 + 2x_1x_2y_1y_2 + x_2^2y_2^2 \\ &= (x_1y_1 + x_2y_2)^2 \\ &= \left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \cdot \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}\right)^2 \\ &= (\mathbf{x} \cdot \mathbf{y})^2. \end{aligned}$$

No exemplo anterior temos definido então um *kernel*

$$k(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})^2,$$

Kernel	Definição $k(\mathbf{x}, \mathbf{y})$	Parâmetros
Linear	$\mathbf{x} \cdot \mathbf{y}$	
Polinomial	$(\mathbf{x} \cdot \mathbf{y} + c)^p$	$c = 0$ : homogêneo $c > 0$ : não homogêneo grau $p \in \mathbb{N}$
Função de Base Radial ( <i>Radial Basis Function</i> , RBF)	$\exp(-\gamma \ \mathbf{x} - \mathbf{y}\ ^2)$	$\gamma = \frac{1}{2\sigma^2}, \sigma \in \mathbb{R} \setminus \{0\}$
Tangente hiperbólica	$\tanh(\gamma \mathbf{x} \cdot \mathbf{y} + c)$	$\gamma \in \mathbb{R} \setminus \{0\}, c \in \mathbb{R}$
Sigmóide logística	$1 / [1 + \exp(-\gamma \mathbf{x} \cdot \mathbf{y} + c)]$	$\gamma \in \mathbb{R} \setminus \{0\}, c \in \mathbb{R}$
Logaritmo [95]	$-\log(1 + \ \mathbf{x} - \mathbf{y}\ ^\beta)$	$0 < \beta \leq 2$
Cauchy [4]	$1 / \left[1 + \frac{\ \mathbf{x} - \mathbf{y}\ ^2}{\sigma}\right]$	$\sigma \in \mathbb{R} \setminus \{0\}$
t-Student generalizado	$1 / [1 + \ \mathbf{x} - \mathbf{y}\ ^\sigma]$	$\sigma \in \mathbb{R} \setminus \{0\}$
Sinc [74, 105]	$\prod_{j=1}^d \frac{\sin(\sigma(x_j - y_j))}{x_j - y_j}$	$\sigma \in \mathbb{R} \setminus \{0\}$

Tabela 8.2: Compilação de *kernels* aplicáveis em aprendizado de máquina .

que calcula o quadrado do produto escalar entre os dois argumentos de entrada. Por outro lado, implicitamente, cada um dos dois padrões  $\mathbf{x}$  e  $\mathbf{y}$  é mapeado para o novo espaço  $\mathcal{H}$  através do mapa (eq. 8.19), ou seja, para  $\phi(\mathbf{x})$  e  $\phi(\mathbf{y})$ , e depois o produto escalar dos dois padrões mapeados é calculado como  $\phi(\mathbf{x}) \cdot \phi(\mathbf{y})$ . Os dois cálculos são equivalentes. Isto significa que, tendo a definição de um *kernel*, o cálculo  $k(\mathbf{x}, \mathbf{y})$  permite indiretamente trabalhar no espaço novo. Note que neste exemplo simples, o espaço novo é simplesmente o  $\mathbb{R}^3$  que nos permite ainda ver explicitamente os padrões mapeados. Com o *kernel* RBF, por exemplo, isso já não é possível. A sua dimensão é infinita, o que pode parecer bastante abstrato. Os pormenores matemáticos são omitidos neste momento. A tabela 8.2 apresenta uma lista de *kernels* usáveis em diversos métodos de aprendizado de máquina, por exemplo, o K-Vizinhos-Mais-Próximos com *kernel*, a SVM ou o PCA com *kernel*. Essa lista não é exaustiva. Qualquer método em aprendizado de máquina que consegue obter resultados usando um produto escalar, pode se beneficiar do uso de *kernels*, potencialmente melhorando o seu desempenho. Os primeiros três *kernels* na tabela 8.2 são os mais utilizados em aprendizado de máquina, nomeadamente o linear, o polinomial e o RBF [52]. Mais *kernels* podem ser encontrados, por exemplo, em [89].

Uma das propriedades desejáveis de um *kernel* é que ele seja positivo definido. A definição é mais fácil feita através da

**Definição 8.2.3 — Matriz de Gram.** Dado um conjunto de treinamento de  $n$  padrões,  $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^n = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ , a matriz de dimensão  $n \times n$  que tem na  $i$ -ésima linha e  $j$ -ésima coluna,

$i, j = 1 \dots, n$ , o valor  $K_{ij} \stackrel{\text{def}}{=} k(\mathbf{x}_i, \mathbf{x}_j)$ , é a matriz de Gram

$$K = \begin{bmatrix} K_{11} & \cdots & K_{1j} & \cdots & K_{1n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ K_{i1} & \cdots & K_{ij} & \cdots & K_{in} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ K_{n1} & \cdots & K_{nj} & \cdots & K_{nn} \end{bmatrix}. \quad (8.20)$$

Esta matriz tem na  $i$ -ésima linha e  $j$ -ésima coluna um produto escalar que é um número real (em geral complexo). No caso mais simples do *kernel* linear  $k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$ , a matriz de Gram está definida nos moldes da Álgebra Linear comum. Se um *kernel* diferente do linear for usado, temos um valor real que é o produto escalar de dois padrões mapeados para o espaço novo, o RKHS.

**Definição 8.2.4 — Kernel Positivo Definido.** O *kernel*  $k$  é positivo definido, se a sua matriz de Gram for positiva definida. A propriedade da matriz ser definida positiva já encontramos uma vez no contexto das funções quadráticas na definição 3.3.2 na pág. 97. O escalar  $\mathbf{x}^\top K \mathbf{x}$  tem que ser positivo para qualquer vetor  $\mathbf{x}$  não nulo.

Como consequência temos duas propriedades [98] do *kernel*, ser positivo na diagonal da sua matriz de Gram para qualquer padrão  $\mathbf{x}$

$$k(\mathbf{x}, \mathbf{x}) \geq 0, \quad (8.21a)$$

e ser simétrico

$$k(\mathbf{x}, \mathbf{y}) = k(\mathbf{y}, \mathbf{x}) \quad (8.21b)$$

Podemos voltar agora ao problema do K-Vizinhos-Mais-Próximos, porém desta vez mediremos as distâncias no novo espaço vetorial que nos se abriu, através do uso de um *kernel* [115]. Em vez de medir a distância Euclidiana  $D_E^2(\mathbf{x}, \mathbf{y})$  entre dois padrões  $\mathbf{x}$  e  $\mathbf{y}$ , a distância Euclidiana é medida entre os seus pares mapeados  $\phi(\mathbf{x})$  e  $\phi(\mathbf{y})$ , ou seja, calculamos  $D_E^2(\phi(\mathbf{x}), \phi(\mathbf{y}))$ , mesmo sem saber diretamente como os padrões mapeados se relacionam. Podemos justapor as distâncias nos dois espaços, no espaço vetorial comum  $\mathbb{R}^m$  de dimensão  $m$  e no espaço mapeado RKHS  $\mathcal{H}$ . Usamos novamente o quadrado das distâncias dos dois lados das equações para simplificar as expressões.

Distância Euclidiana no Espaço Vetorial Original  $\mathbb{R}^m$

$$\begin{aligned} D_E^2(\mathbf{x}, \mathbf{y}) &\stackrel{1.1.1}{=} ||\mathbf{x} - \mathbf{y}||^2 \\ &\stackrel{1.1.1}{=} \mathbf{x} \cdot \mathbf{x} - 2\mathbf{x} \cdot \mathbf{y} + \mathbf{y} \cdot \mathbf{y} \end{aligned} \quad (8.22a)$$

Distância Euclidiana no Espaço Vetorial Mapeado  $\mathcal{H}$

$$\begin{aligned} D_E^2(\phi(\mathbf{x}), \phi(\mathbf{y})) &\stackrel{1.1.1}{=} ||\phi(\mathbf{x}) - \phi(\mathbf{y})||^2 \\ &\stackrel{1.1.1}{=} \phi(\mathbf{x}) \cdot \phi(\mathbf{x}) - 2\phi(\mathbf{x}) \cdot \phi(\mathbf{y}) + \phi(\mathbf{y}) \cdot \phi(\mathbf{y}) \\ &\stackrel{1.8}{=} k(\mathbf{x}, \mathbf{x}) - 2k(\mathbf{x}, \mathbf{y}) + k(\mathbf{y}, \mathbf{y}). \end{aligned} \quad (8.22b)$$

Para o *kernel* linear da tabela 8.2, a (eq. 8.22a) e (eq. 8.22b) são idênticas. Para o *kernel* RBF, por exemplo, a (eq. 8.22b) instancia para

$$D_E(\phi(\mathbf{x}), \phi(\mathbf{y})) = \sqrt{2 - 2 \exp\{-||\mathbf{x} - \mathbf{y}||^2/(2\sigma^2)\}},$$

pois os dois termos diagonais  $k(\mathbf{x}, \mathbf{x})$  e  $k(\mathbf{y}, \mathbf{y})$  reduzem para o valor 1. A consequência desse incremento conceitual é que no novo espaço, a separabilidade das classes possa ser melhor. Então ganharíamos com o uso do kernel.

Um outro benefício potencial do uso do *kernel* poderia ser a separabilidade linear de classes no novo espaço de padrões que antes no espaço vetorial convencional não apresentavam essa facilidade. Esse raciocínio é exatamente uma das ideias da Máquina de Vetor de Suporte. Considerando um problema de classificação binária, o uso do kernel pode eventualmente mapear todos os padrões de uma classe de um lado do hiperplano e todos os padrões da outra classe do outro lado do hiperplano. Mesmo, se isso não for possível, o mapeamento pode ter melhorado a separabilidade. Voltaremos a esse assunto na apresentação da SVM [13].

### 8.3 Árvores de Decisão

A classificação de um padrão desconhecido pode ser realizado por uma sequência de perguntas relativas às características do padrão. No caso mais simples, essas perguntas estão relacionadas a uma única característica, e a resposta é binária. Um exemplo no nosso exemplo de iris poderia ser a seguinte pergunta sobre a terceira característica: “O comprimento da pétala é menor ou igual a 4.95 cm?”. Essa pergunta constitui um *nó* da árvore. Antes já foram feitas perguntas de mesma natureza, ou seja, em um nó anterior poderia ter sido feito uma pergunta semelhante, por exemplo “A largura da pétala é menor ou igual a 1.75 cm?”. Cada nó da árvore então tem dois *filhos* que representam “Sim” ou “Não”. Seguindo os caminhos dos “Sim” e “Não” desde a *raiz* da árvore, chega-se a uma *folha* da árvore que representa uma classificação realizada, por exemplo, da classe virginica.

Vamos começar com um exemplo motivador, ilustrado na figura 8.3. É a árvore<sup>3</sup> de classificação do nosso conjunto de flores. Os dois nós mencionados podem ser vistos no meio da árvore. Cada nó é uma caixa com informações que guiam o processo de decisão. Quando o nó contiver na primeira linha o nome de uma característica, então essa característica é usada para montar a pergunta. Isto significa que essa característica ganhou das outras características em uma competição relativa à importância de dividir o conjunto de padrões conforme às classes desses padrões. As perguntas nos nós baseiam-se em um limiar do valor da característica. Por exemplo, a raiz divide o conjunto completo de flores, perguntando, se a largura da pétala tem um valor abaixo ou igual, ou acima de 0.8 cm. Dependendo da resposta dessa pergunta o conjunto de padrões que chegaram até esse nó é dividido em duas metades. O conjunto de padrões que está associado ao nó está quantificado na informação “amostras”. Obviamente temos todos as 150 flores na raiz. O campo “valor” expressa a proporção de cada classe. Na raiz temos 50 padrões de cada classe. O filho esquerdo da raiz contém todos as setosas e nenhuma outra espécie, portanto, não precisamos mais seguir esse caminho. O nó é uma folha. O filho direito porém contém as restantes 100 versicolor e virginica. Então, a quantidade de padrões a serem divididos diminui com cada pergunta respondida. Uma parte dos padrões diz respeito ao ramo direito, outra parte dos padrões diz respeito ao ramo esquerdo. O campo “classe” mostra qual classe tem a maioria de amostras restantes no nó. Em caso de empate a primeira classe na lista é arbitrariamente escolhida. O campo “gini” mostra um valor numérico que decide como o processo de ramificação da árvore é feita. Vamos estudar esse critério entre outros em breve.

<sup>3</sup>Foi usada a classe `sklearn.tree.DecisionTreeClassifier` da biblioteca de aprendizado de máquina `scikit-learn` para inferir a árvore e gerar o gráfico.

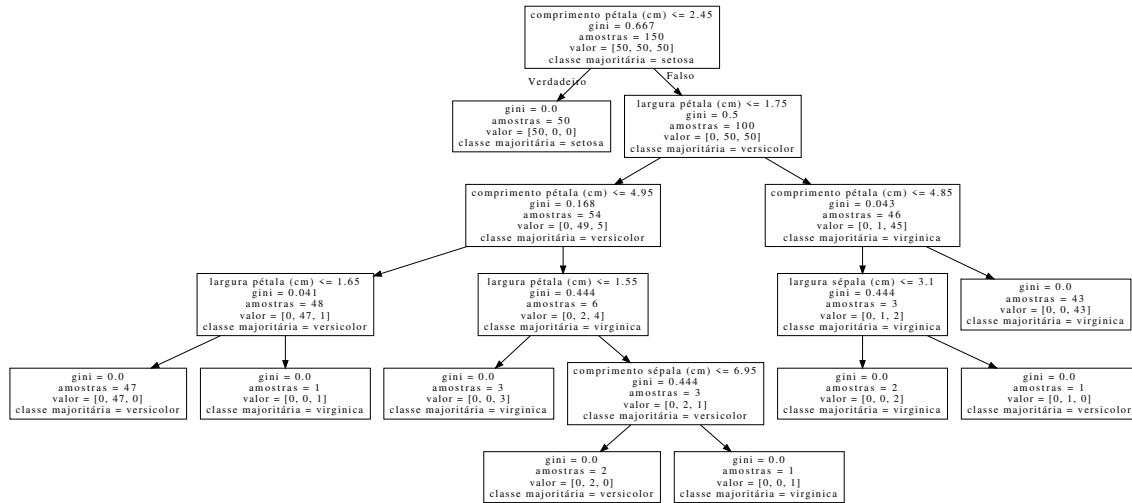


Figura 8.3: Árvore de classificação das flores iris gerada pelo software `scikit-learn`. Em caso de empate de amostras por classe, a classe majoritária é a primeira da lista.

### 8.3.1 Árvores de Decisão com Características Simbólicas

No exemplo da árvore para classificar as flores vimos que as perguntas se basearam em limiares de uma única característica, que por natureza tem valores contínuos. A origem dessas árvores tradicionalmente usou preferencialmente características de natureza *nominal* ou *simbólica*. Na comunidade de pesquisadores de árvores de decisão as características tem o sinônimo de *atributos*. Esse nome usaremos preferencialmente no contexto de uma árvore de decisão. Chamaremos o  $j$ -ésimo valor do atributo novamente de  $x_j$ , como no caso de uma característica contínua. Pelo contexto, o leitor reconhecerá a natureza discreta ou contínua. O  $j$ -ésimo valor  $x_j$  de uma característica contínua em geral é um número real, ou seja,  $x_j \in \mathbb{R}$ . Então todas as componentes de um vetor de características contínuas vêm do mesmo domínio  $\mathbb{R}$ . Em comparação, o valor  $x_j$  de cada atributo nominal tem um domínio próprio  $\mathcal{X}_j$  de valores, com uma cardinalidade<sup>4</sup>  $q = |\mathcal{X}_j|$  em geral diferente, ou seja,

$$x_j \in \mathcal{X}_j = \{v_{j_1}, v_{j_2}, \dots, v_{j_r}, \dots, v_{j_q}\}. \quad (8.23)$$

Para ser correto, tivemos que introduzir um subíndice  $j_r$  para cada atributo, pois a quantidade de valores possíveis do  $j$ -ésimo atributo pode variar.

■ **Exemplo 8.2** Citamos aqui<sup>5</sup> o exemplo do trabalho importante de Quinlan sobre árvores de decisão [85]. Quatro atributos (características)  $x_j, j = 1, 2, 3, 4$  descrevem cada um dos 14 padrões, “Previsão de tempo”, “Temperatura”, “Umidade” e “Vento”. Temos então

$$\begin{aligned} \text{Previsão de tempo} &= x_1 \in \mathcal{X}_1 = \{v_{1_1}, v_{1_2}, v_{1_3}\} = \{\text{Ensolarado, Nublado, Chuva}\}, |\mathcal{X}_1| = 3, \\ \text{Temperatura} &= x_2 \in \mathcal{X}_2 = \{v_{2_1}, v_{2_2}, v_{2_3}\} = \{\text{Quente, Amena, Moderada}\}, |\mathcal{X}_2| = 3, \\ \text{Umidade} &= x_3 \in \mathcal{X}_3 = \{v_{3_1}, v_{3_2}\} = \{\text{Alta, Normal}\}, |\mathcal{X}_3| = 2, \\ \text{Vento} &= x_4 \in \mathcal{X}_4 = \{v_{4_1}, v_{4_2}\} = \{\text{Fraco, Forte}\}, |\mathcal{X}_4| = 2, \end{aligned}$$

<sup>4</sup>Relembrando: cardinalidade = quantidade de elementos de um conjunto

<sup>5</sup>No trabalho original [85], tem um atributo “Ventoso” com um domínio binário {Verdadeiro, Falso}, em vez de um atributo “Vento” com o domínio {Fraco, Forte}.

Número de padrão	Atributos					Classe	
	Previsão de tempo		Temperatura		Umidade	Vento	
		Nominal	Numérico	Nominal	Numérico		
1	Ensolarado	Quente	85	Alta	85	Fraco	Não
2	Ensolarado	Quente	80	Alta	90	Forte	Não
3	Nublado	Quente	83	Alta	78	Fraco	Sim
4	Chuva	Amena	70	Alta	96	Fraco	Sim
5	Chuva	Moderada	68	Normal	80	Fraco	Sim
6	Chuva	Moderada	65	Normal	70	Forte	Não
7	Nublado	Moderada	64	Normal	65	Forte	Sim
8	Ensolarado	Amena	72	Alta	95	Fraco	Não
9	Ensolarado	Moderada	69	Normal	70	Fraco	Sim
10	Chuva	Amena	75	Normal	80	Fraco	Sim
11	Ensolarado	Amena	75	Normal	70	Forte	Sim
12	Nublado	Amena	72	Alta	90	Forte	Sim
13	Nublado	Quente	81	Normal	75	Fraco	Sim
14	Chuva	Amena	71	Alta	80	Forte	Não

Tabela 8.3: Conjunto de treinamento ilustrativo para gerar uma árvore de classificação.

Vamos estudar a construção de uma árvore, onde cada padrão é descrito exclusivamente por características não numéricas. Afinal, a árvore somente pode trabalhar com esse tipo de características. Se a característica não for simbólica, ela tem que ser transformada nessa forma. Repare que nas flores um limiar automaticamente cria uma característica simbólica com dois valores, nomeadamente “abaixo” e “acima”. Um ponto de entrada para estudar as árvores de decisão são as arquiteturas ID3 [85] e C4.5 [84]. A tabela 8.3 mostra um conjunto de treinamento ilustrativo de [85]. A tarefa é o aprendizado da possibilidade de uma atividade não especificada, por exemplo, podemos imaginar jogar tênis ou caminhar. Existem duas classes possíveis, “Sim” e “Não”. Todos os atributos na referência original [85] são de natureza nominal. Isto significa que não existe nenhuma afinidade geométrica entre os padrões que pudesse permitir o uso de uma métrica numérica, por exemplo, a distância Euclidiana, para montar um classificador do tipo K-Vizinhos-Mais-Próximos, por exemplo. O significado e os domínios dos quatro atributos já foram explicados na exemplo 8.2. Além dos valores nominais dos atributos “Temperatura” e “Umidade”, na tabela 8.3 foi tentado atribuir um valor numérico correspondente, em graus Fahrenheit e porcentagem, respectivamente. O objetivo desse acréscimo em relação à tabela original, é ilustrar posteriormente como se converte uma característica numérica em um atributo nominal binário.

### Tabelas de Contingência

Em cada nó da árvore temos que julgar qual característica é usada para dividir os padrões que restaram neste nó, veja a primeira linha de texto dos nós que não são folhas na figura 8.3. Um esquema de representação ideal para calcular estimativas de probabilidades necessárias é a *tabela de contingência* (CT) que relaciona duas variáveis aleatórias discretas. Além das variáveis serem discretas, elas são nominais, ou seja, sem qualquer relação de proximidade. No contexto de classificação, a CT é uma matriz que diz respeito a um atributo particular. No exemplo da tabela 8.3, somente considerando a parte nominal, temos então quatro tabelas de contingência, uma para cada atributo. Com o avanço do crescimento da árvore, o conjunto de padrões muda, e, consequentemente a CT. Os exemplos vão esclarecer isso mais adiante. Os critérios que são usados para guiar o crescimento da árvore podem ser apresentadas sem estudar a teoria da CT, porém o entendimento dos critérios fica mais plausível. A estrutura da tabela de contingência é mostrada exemplarmente na

ATRIBUTO	CLASSE		TOTAL	ATRIBUTO	CLASSE		TOTAL
Previsão de tempo	Sim	Não		Temperatura	Sim	Não	
Ensolarado	2	3	5	Quente	2	2	4
	4	0	4		4	2	6
	3	2	5		3	1	4
	9	5	14		9	5	14

ATRIBUTO	CLASSE		TOTAL	ATRIBUTO	CLASSE		TOTAL
Umidade	Sim	Não		Vento	Sim	Não	
Alta	3	4	7	Fraco	6	2	8
	6	1	7		3	3	6
	9	5	14		9	5	14

Tabela 8.4: Tabela de contingência para cada atributo da tabela 8.3.

tabela 8.4 para todos os quatro atributos nominais do exemplo da tabela 8.3. As linhas representam todos os valores possíveis do atributo, as colunas as classes. A quantidade de linhas depende aos valores possíveis do domínio particular de cada atributo, veja o exemplo 8.2. Cada posição da matriz interior na CT faz a contagem das ocorrências de cada valor por classe, por exemplo, o atributo “Umidade” tem quatro ocorrências do valor “Alta” para a classe “Não”. A última coluna contém o valor marginal de cada valor do atributo, independente da classe, por exemplo umidade “Alta” aparece sete vezes, marginalizando sobre todas as duas classes. A última linha é o valor marginal de cada classe, independente do valor do atributo, por exemplo para a classe “Sim” no atributo “Umidade” temos nove ocorrências. Finalmente, o valor marginal total na posição inferior direita é idêntico ao número  $n$  de padrões. Novamente deve-se observar que esse número  $n$  depende de cada nó da árvore. No exemplo de iris da figura 8.3, temos  $n = 150$  na raiz, dando origem a um conjunto de tabelas de contingência (veremos as discretização de características contínuas posteriormente). Após a primeira bifurcação da raiz, o ramo esquerdo vira uma folha. O ramo da direita carrega agora  $n = 100$  padrões, dando origem a um novo conjunto de CT.

Em geral para um atributo, a tabela de contingência tem a estrutura ilustrada na tabela 8.5. Faça o casamento entre esta estrutura geral e as quatro instâncias na tabela 8.4 como exercício para entender melhor o significado de cada campo. Em cada célula da matriz tem a contagem quanto frequente um determinado valor  $v_r$  do atributo  $x$  ocorre para a classe  $C_k$ . Descartamos aqui o primeiro índice  $j$  do atributo para simplificar o aspecto das variáveis. Por exemplo, o segundo valor possível do primeiro atributo  $x_j = x_1$ , com  $j = 1$ , deveria se chamar  $v_{12}$ , mas o chamamos simplesmente  $v_2$ . Pelo contexto, espera-se do leitor a correta identificação, de qual atributo se trata. Podemos então identificar as seguintes variáveis aleatórias nominais na tabela de contingência

		Classes						Total de linhas $\Sigma$
		$C_1$	$C_2$	$\dots$	$C_k$	$\dots$	$C_c$	
Valores do atributo $x_j$	$v_1$	$n_{1,1}$	$n_{1,2}$	$\dots$	$n_{1,k}$	$\dots$	$n_{1,c}$	$n_{1,\bullet}$
	$v_2$	$n_{2,1}$	$n_{2,2}$	$\dots$	$n_{2,k}$	$\dots$	$n_{2,c}$	$n_{2,\bullet}$
	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
	$v_r$	$n_{r,1}$	$n_{r,2}$	$\dots$	$n_{r,k}$	$\dots$	$n_{r,c}$	$n_{r,\bullet}$
	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
	$v_q$	$n_{q,1}$	$n_{q,2}$	$\dots$	$n_{q,k}$	$\dots$	$n_{q,c}$	$n_{c,\bullet}$
Total de colunas $\Sigma$		$n_{\bullet,1}$	$n_{\bullet,2}$	$\dots$	$n_{\bullet,k}$	$\dots$	$n_{\bullet,c}$	$n$

Tabela 8.5: Tabela de contingência de contagem de ocorrências de cada valor do atributo para cada classe.

tabela 8.5 para cada atributo  $x_j, j = 1, \dots, m$ .

$n_{r,k}$  Quantidade de ocorrências do valor  $v_r$  que pertencem a classe  $C_k$

(8.24a)

$n_{r,\bullet} = \sum_{k=1}^c n_{r,k}$  Quantidade de ocorrências do valor  $v_r$ , marginalizado sobre todas as classes  
(8.24b)

$n_{\bullet,k} = \sum_{r=1}^q n_{r,k}$  Quantidade de ocorrências que pertencem a classe  $C_k$ , marginalizado sobre todos os valores  
(8.24c)

$n = \sum_{k=1}^c n_{\bullet,k} = \sum_{r=1}^q n_{r,\bullet} = \sum_{r=1}^q \sum_{k=1}^c n_{r,k}$  Quantidade de ocorrências do valor  $v_r$  que pertencem a classe  $C_k$ , marginalizado sobre todas os valores e todas as classes  
(8.24d)

Partindo das quantidades armazenadas na tabela 8.5, facilmente podemos estimar as probabilidades  $\hat{P}$  necessárias para calcular os critérios que guiam a geração da árvore. Omite-se aqui o símbolo de estimar “ $\wedge$ ” para facilitar a nomenclatura. Lembre que estamos trabalhando aqui com uma variável discreta. A probabilidade pode ser simplesmente estimado pela proporção de um subconjunto, por exemplo, a probabilidade de ter um número par, jogando um dado, é  $P(\text{par}) = 3/6 = 50\%$ .

Na tabela 8.6, podemos identificar as seguintes probabilidades estimadas para cada atributo

		Classes						Marginais das classes $\Sigma$
		$C_1$	$C_2$	...	$C_k$	...	$C_c$	
Valores dos atributo $x_j$	$v_1$	$P(C_1, v_1)$	$P(C_2, v_1)$	...	$P(C_k, v_1)$	...	$P(C_c, v_1)$	$P(v_1)$
	$v_2$	$P(C_1, v_2)$	$P(C_2, v_2)$	...	$P(C_k, v_2)$	...	$P(C_c, v_2)$	$P(v_2)$
	$\vdots$	$\vdots$	$\vdots$	..	$\vdots$	..	$\vdots$	$\vdots$
	$v_r$	$P(C_1, v_r)$	$P(C_2, v_r)$	...	$P(C_k, v_r)$	...	$P(C_c, v_r)$	$P(v_r)$
	$\vdots$	$\vdots$	$\vdots$	..	$\vdots$	..	$\vdots$	$\vdots$
	$v_q$	$P(C_1, v_q)$	$P(C_2, v_q)$	...	$P(C_k, v_q)$	...	$P(C_c, v_q)$	$P(v_q)$
Marginais dos valores do atributo $\Sigma$		$P(C_1)$	$P(C_2)$	...	$P(C_k)$	...	$P(C_c)$	100%

Tabela 8.6: Tabela de contingência de probabilidades de cada valor do atributo para cada classe.

$x_j, j = 1, \dots, m$ .

$$P(C_k, v_r) = \frac{n_{r,k}}{n} \quad \begin{array}{l} \text{Probabilidade conjunta que atributo} \\ \text{tenha o valor } v_r \text{ e padrão pertença} \\ \text{simultaneamente à classe } C_k \end{array} \quad (8.25a)$$

$$P(v_r) = \frac{n_{r,\bullet}}{n} = \frac{1}{n} \sum_{k=1}^c n_{r,k} = \sum_{k=1}^c P(C_k, v_r) \quad \begin{array}{l} \text{Probabilidade do valor } v_r, \\ \text{marginalizado sobre todas as classes} \end{array} \quad (8.25b)$$

$$P(C_k) = \frac{n_{\bullet,k}}{n} = \frac{1}{n} \sum_{r=1}^q n_{r,k} = \sum_{r=1}^q P(C_k, v_r) \quad \begin{array}{l} \text{Probabilidade que padrão pertença} \\ \text{à classe } C_k, \text{ marginalizado sobre} \\ \text{todas as valores} \end{array} \quad (8.25c)$$

$$100\% = \sum_{k=1}^c P(C_k) = \sum_{r=1}^q P(v_r) = \sum_{r=1}^q \sum_{k=1}^c P(C_k, v_r) \quad \begin{array}{l} \text{Probabilidade que atributo tenha} \\ \text{qualquer valor e padrão pertença} \\ \text{a qualquer classe} \end{array} \quad (8.25d)$$

### Ganho de Entropia como Critério de Separação de Subconjuntos de Padrões

Agora temos todas as probabilidades para definir os vários critérios que decidem quais ramos um nó da árvore terá. Estamos no meio da árvore, com um subconjunto sobrando que tem padrões de mais que uma classe. Temos que decidir qual atributo serve para definir essa ramificação (bifurcação em caso de árvore binária). No nó raiz da árvore da figura 8.3, um limiar apropriado da largura da pétala conseguiu separar todas as 50 amostras da classe  $C_1$  setosa dos restantes 100 amostras. Como o ramo da esquerda ficou completamente homogêneo com uma única classe, não precisamos mais seguir esse caminho. As proporções [50, 50, 50] dos padrões na raiz foram transformadas na proporção [50, 0, 0] no ramo esquerdo, e [0, 50, 50] no ramo direito. Então podemos suspeitar que a divisão no ramo esquerdo que mostra uma forte vantagem de 50 amostras para setosa e zero para as outras duas classes é uma divisão favorável.

Vamos começar com um clássico da teoria de informação, a *entropia*, conceito criado pelo trabalho pioneiro de Shannon [99].

**Definição 8.3.1 — Entropia.** Dada uma variável aleatória discreta  $x$  com o domínio finito de  $N$  valores distintos  $\{x_1, x_2, \dots, x_\ell, \dots, x_N\}$ , cada um com a probabilidade  $P(x_\ell)$ , a *entropia* de  $x$  é definida como

$$H(x) \stackrel{\text{def}}{=} - \sum_{\ell=1}^N P(x_\ell) \log_2 P(x_\ell) \stackrel{[2]}{=} \mathbb{E} \left[ \frac{1}{\log_2 P(x_\ell)} \right]. \quad (8.26)$$

A base binária do logaritmo indica que se trata da *entropia de informação*, cuja unidade é expressa em “bits”. A última igualdade<sup>a</sup> explica-se pela definição do valor esperado  $\mathbb{E}[x]$  de uma variável aleatória discreta na definição 1.2.3 na pág. 19. O argumento do operador  $\mathbb{E}$  é proporcional ao logaritmo inverso de uma probabilidade. Quanto menos provável, melhor, mais informação o valor da variável aleatória tem. Pode-se considerar então o argumento como uma *auto-informação*.

<sup>a</sup>Relembrando,  $-\log y = \log \frac{1}{y}$ .

■ **Exemplo 8.3** Considerando o exemplo da tabela 8.3 e a tabela de contingência do atributo “Vento” na tabela 8.4, a probabilidade marginal, ignorando a classe, é para os dois valores possíveis “Fraco” e “Forte”  $P(\text{Fraco}) = 8/14$  e  $P(\text{Forte}) = 6/14$ . Então a entropia<sup>6</sup> da variável aleatória “Vento” é

$$\begin{aligned} H(\text{Vento}) &= -P(\text{Fraco}) \log_2 P(\text{Fraco}) - P(\text{Forte}) \log_2 P(\text{Forte}) \\ &= -\frac{8}{14} \log_2 \frac{8}{14} - \frac{6}{14} \log_2 \frac{6}{14} = -\frac{8}{14} * (-0.807) - \frac{6}{14} * (-1.222) = 0.985. \end{aligned}$$

Estamos observando uma entropia perto do valor máximo 1. Se tivéssemos duas probabilidades bastante desequilibrados, por exemplo 1% e 99%, a entropia chega perto de zero. A entropia então penaliza se houver muita igualdade nas probabilidades por um valor alto. Para ramificar o nó da árvore estamos mais interessados na entropia da classe do que na entropia de um atributo, pois é importante, como o atributo consegue dividir os padrões em relação às classes. Focamos então na variável aleatória discreta  $\mathcal{C}$  que tem um domínio de  $c$  diferentes valores

$$\mathcal{C} \in \{\mathcal{C}_1, \dots, \mathcal{C}_k, \dots, \mathcal{C}_c\}. \quad (8.27)$$

Considerando novamente o exemplo da tabela 8.3 e a tabela de contingência do atributo “Vento” na tabela 8.4, para a variável aleatória da classe  $\mathcal{C} \in \{\mathcal{C}_1, \mathcal{C}_2\} = \{\text{Sim}, \text{Não}\}$ , temos para todos os quatro atributos os valores idênticos das probabilidades, marginalizando sobre os valores dos atributos  $P(\text{Sim}) = 9/14$  e  $P(\text{Não}) = 5/14$  e então para a entropia

$$\begin{aligned} H(\mathcal{C}) &= -P(\text{Sim}) \log_2 P(\text{Sim}) - P(\text{Não}) \log_2 P(\text{Não}) \\ &= -\frac{9}{14} \log_2 \frac{9}{14} - \frac{5}{14} \log_2 \frac{5}{14} = -\frac{9}{14} * (-0.637) - \frac{5}{14} * (-1.485) = 0.940. \end{aligned}$$

Para a variável aleatória “Classe” então temos a

**Definição 8.3.2 — Entropia a priori de Classe.** Dada a variável aleatória nominal “Classe”,  $\mathcal{C}$  com um domínio de  $c$  valores discretos  $\{\mathcal{C}_1, \dots, \mathcal{C}_k, \dots, \mathcal{C}_c\}$  possíveis, cada valor com uma

<sup>6</sup>Em todos os cálculos numéricos omitiremos a unidade “bits”.

probabilidade  $P(\mathcal{C}_k)$ , a entropia a priori da variável aleatória  $\mathcal{C}$  é

$$H(\mathcal{C}) = - \sum_{k=1}^c P(\mathcal{C}_k) \log_2 P(\mathcal{C}_k) = \mathbb{E} \left[ \frac{1}{\log_2 P(\mathcal{C}_k)} \right]. \quad (8.28)$$

Podemos chamar esse entropia uma *entropia a priori*, pois considera somente a probabilidades a priori das classes  $P(\mathcal{C}_k)$ , sem saber nada sobre as probabilidades das classes condicionais em relação às probabilidades dos valores de um determinado atributo. A questão importante agora é como a entropia desta variável aleatória “Classe” se comporta, se houver uma separação dos padrões conforme os valores de um determinado atributo, ou seja, como será a entropia dessa variável “Classe”, se considerarmos somente um subconjunto de padrões em um ramo da árvore que foi selecionado por um determinado valor de um determinado atributo? Por exemplo, se considerarmos o atributo “Vento” em algum nó, teremos uma bifurcação, relativa aos valores “Forte” e “Fraco”, cada ramo restringindo o conjunto de padrões que tenham o respectivo valor. Considerando um determinado atributo e um determinado valor do domínio deste atributo então podemos definir

**Definição 8.3.3 — Entropia de Ramo.** Dada a variável aleatória nominal “Classe”,  $\mathcal{C}$ , e dado um atributo  $x$  e um valor específico  $v_r$  do domínio finito de  $q$  valores distintos  $\{v_1, v_2, \dots, v_r, \dots, v_q\}$  deste atributo  $x$ , e a sua probabilidade específica  $P(v_r)$  de ter esse valor, a *entropia do ramo* da variável aleatória “Classe” com o valor  $v_r$  do atributo é definida como

$$H(\mathcal{C}|v_r) \stackrel{\text{def}}{=} - \sum_{k=1}^c P(\mathcal{C}_k|v_r) \log_2 P(\mathcal{C}_k|v_r) = \mathbb{E} \left[ \frac{1}{\log_2 P(\mathcal{C}_k|v_r)} \right]. \quad (8.29)$$

Usou-se uma probabilidade condicional  $P(\mathcal{C}_k|v_r)$  da  $k$ -ésima classe, dado o valor concreto  $v_r$  do atributo. Pelas regras elementares de probabilidade já vistos no contexto da regra de Bayes na (eq. 7.1) na pág. 188, válido também para uma variável aleatória nominal, temos para a relação entre as probabilidades a priori, conjunta e condicional

$$\begin{aligned} P(\mathcal{C}_k, v_r) &= P(\mathcal{C}_k|v_r)P(v_r) = P(v_r|\mathcal{C}_k)P(\mathcal{C}_k) \\ \implies P(\mathcal{C}_k|v_r) &= \frac{P(\mathcal{C}_k, v_r)}{P(v_r)}. \end{aligned} \quad (8.30)$$

As probabilidades da (eq. 8.30) podem ser imediatamente extraídas da tabela de contingência tabela 8.6.

■ **Exemplo 8.4** Considerando o exemplo concreto do atributo “Previsão de tempo” na tabela 8.4, temos para o valor  $v_1 = \text{Ensolarado}$

$$\begin{aligned} P(\mathcal{C}_1|v_1) &= P(\text{Sim}|\text{Ensolarado}) = \frac{P(\mathcal{C}_1, v_1)}{P(v_1)} = \frac{2}{5} \\ P(\mathcal{C}_2|v_1) &= P(\text{Não}|\text{Ensolarado}) = \frac{P(\mathcal{C}_2, v_1)}{P(v_1)} = \frac{3}{5}. \end{aligned}$$

Então a entropia do ramo pela (eq. 8.29) é

$$\begin{aligned} H(\mathcal{C}|v_1) &= H(\mathcal{C}|\text{Ensolarado}) = -P(\mathcal{C}_1|v_1)\log_2 P(\mathcal{C}_1|v_1) - P(\mathcal{C}_2|v_1)\log_2 P(\mathcal{C}_2|v_1) \\ &= -\frac{2}{5}\log_2 \frac{2}{5} - \frac{3}{5}\log_2 \frac{3}{5} = 0.971. \end{aligned}$$

Semelhantemente, para os outros dois valores possíveis do atributo temos

$$\begin{aligned} H(\mathcal{C}|v_2) &= H(\mathcal{C}|\text{Nublado}) = -P(\mathcal{C}_1|v_2)\log_2 P(\mathcal{C}_1|v_2) - P(\mathcal{C}_2|v_2)\log_2 P(\mathcal{C}_2|v_2) \\ &= -\frac{4}{4}\log_2 \frac{4}{4} - \frac{0}{4}\log_2 \frac{0}{4} = -0.0 - 0.0 = 0.0. \end{aligned}$$

Neste cálculo usou-se a convenção  $0\log 0 = 0$ .

$$\begin{aligned} H(\mathcal{C}|v_3) &= H(\mathcal{C}|\text{Chuva}) = -P(\mathcal{C}_1|v_3)\log_2 P(\mathcal{C}_1|v_3) - P(\mathcal{C}_2|v_3)\log_2 P(\mathcal{C}_2|v_3) \\ &= -\frac{3}{5}\log_2 \frac{3}{5} - \frac{2}{5}\log_2 \frac{2}{5} = 0.971. \end{aligned}$$

■

Para concluir a avaliação da ramificação de um nó da árvore de decisão, temos que juntar o resultado de todos os ramos de uma forma já conhecida, pelo valor esperado da entropia dos ramos, ou equivalentemente, a soma ponderada pelas probabilidades de cada ramo.

**Definição 8.3.4 — Entropia de Atributo.** Dada a variável aleatória nominal “Classe”, dado um atributo  $x$  com todos os  $q$  valores do seu domínio  $\{v_1, v_2, \dots, v_r, \dots, v_q\}$ , e suas probabilidades específicas  $P(v_r)$ , a *entropia do atributo* da variável aleatória “Classe” é o valor esperado sobre os valores do atributo das entropias dos ramos

$$\begin{aligned} H(\mathcal{C}|x) &\stackrel{\text{def}}{=} \sum_{r=1}^q P(v_r)H(\mathcal{C}|v_r) \\ &= \sum_{r=1}^q P(v_r) \left\{ -\sum_{k=1}^c [P(\mathcal{C}_k|v_r)\log_2 P(\mathcal{C}_k|v_r)] \right\}. \end{aligned} \quad (8.31)$$

Temos agora a possibilidade de julgar qual atributo é o mais apropriado para ganhar o privilégio de definir a pergunta no ramo e definir a ramificação. Vamos comparar a entropia a priori do nó e a entropia do atributo, definindo o

**Definição 8.3.5 — Ganho de Entropia.** Dada a variável aleatória nominal “Classe”,  $\mathcal{C}$ , e a sua entropia a priori da definição 8.3.2, dado um atributo  $x$  e o seu valor esperado das entropias dos ramos  $H(\mathcal{C}|x)$ , o *ganho de entropia* é a diferença entre a entropia a priori e a entropia do atributo

$$\Delta H(\mathcal{C}) \stackrel{\text{def}}{=} H(\mathcal{C}) - H(\mathcal{C}|x). \quad (8.32)$$

A (eq. 8.32) expressa claramente uma diferença entre conhecimento antes de uma ramificação por um atributo  $x$ , e após da ramificação.

■ **Exemplo 8.5** Temos para o atributo  $x_1 = \text{Previsão de tempo}$

$$\begin{aligned} \Delta H(\mathcal{C}) &= H(\mathcal{C}) - H(\mathcal{C}|x_1) = H(\mathcal{C}) - H(\mathcal{C}|\text{Previsão de tempo}) \\ &= H(\mathcal{C}) - \sum_{r=1}^3 P(v_r)H(\mathcal{C}|v_r) \\ &= H(\mathcal{C}) - \{P(\text{Ensolarado})H(\mathcal{C}|\text{Ensolarado}) \\ &\quad + P(\text{Nublado})H(\mathcal{C}|\text{Nublado}) \\ &\quad + P(\text{Chuva})H(\mathcal{C}|\text{Chuva})\} \\ &= 0.94029 - (5/14 * 0.971 + 5/14 * 0.971 + 4/14 * 0.000) = 0.247 \end{aligned}$$

Semelhantemente, temos para os outros três atributos

$$\begin{aligned}\Delta H(\mathcal{C}) &= H(\mathcal{C}) - H(\mathcal{C}|x_2) = H(\mathcal{C}) - H(\mathcal{C}|\text{Temperatura}) \\ &= 0.94029 - (6/14 * 0.918 + 4/14 * 0.811 + 4/14 * 1.000) = 0.029 \\ \Delta H(\mathcal{C}) &= H(\mathcal{C}) - H(\mathcal{C}|x_3) = H(\mathcal{C}) - H(\mathcal{C}|\text{Umidade}) \\ &0.94029 - (7/14 * 0.985 + 7/14 * 0.592) = 0.152 \\ \Delta H(\mathcal{C}) &= H(\mathcal{C}) - H(\mathcal{C}|x_4) = H(\mathcal{C}) - H(\mathcal{C}|\text{Vento}) \\ &0.94029 - (8/14 * 0.811 + 6/14 * 1.000) = 0.048\end{aligned}$$

O vencedor é  $x_1$  = “Previsão de tempo” com o maior valor do ganho de entropia  $\Delta H(\mathcal{C}) = 0.247$ . Esse atributo tem agora o direito de definir as ramificações do nó. O nó neste exemplo é a raiz. Temos três valores ( $v_1$  = Ensolarado,  $v_2$  = Nublado,  $v_3$  = Chuva) possíveis, então a ramificação gera três ramos, visível no segundo nível da figura 8.4. Cada ramo filtra o conjunto de padrões cujo valor do atributo vencedor coincide com o valor do atributo do ramo. Por exemplo, o valor “Chuva” aparece cinco vezes. Esse subconjunto é recursivamente analisado. Se o critério de um nó ser uma folha for satisfeita, o crescimento do ramo terminou. O critério no caso mais simples é a homogeneidade dos rótulos de classe, ou seja, o conjunto filtrado pertence todo à mesma classe. Isso acontece com o subconjunto de quatro amostras do valor “Nublado” que são todos da classe “Sim”. Os nós que não são declarados como folhas vão ser analisados da mesma maneira como todos os conjuntos. No caso do valor “Chuva” o conjunto tem agora  $n = 5$  amostras. Nesse conjunto, o atributo vencedor é “Vento” que gera dois filhos homogêneos que então serão folhas da árvore. A mesma coisa acontece com os cinco amostras do valor “Ensolarado” do nó raiz da árvore, que gera por sua vez duas folhas. ■

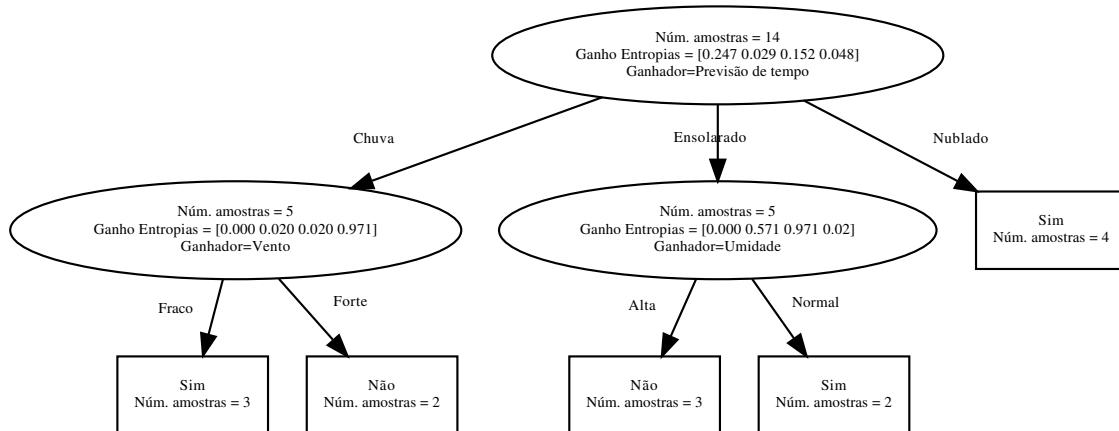


Figura 8.4: Árvore de classificação do exemplo de atividades [85].

A forma básica da geração recursiva da árvore está resumida no algoritmo 15. Cada nó determina seu atributo vencedor, a não ser que o nó já seja uma folha. Uma folha em uma árvore de classificação pode ser definida, se todos os padrões restantes pertencem à mesma classe. Existem outros critérios de impedir a ramificação, mesmo se não houver homogeneidade de classe. Especialmente o processo de *poda* (*pruning*) aborta a ramificação. A técnica mais simples é declarar um nó como um folha pela maioria. A classe com a maior quantidade de amostras no nó então é declarada a classe classificada. Obviamente isso introduz um erro na classificação para um padrão desconhecido. O importante porém, é não aumentar o erro em termos estatísticos. Posteriormente vamos estudar os métodos de medir o desempenho de um sistema, por exemplo, como estimar a taxa de erro de um classificador, neste caso, uma árvore de decisão.

---

**Algoritmo 15:** Geração recursiva de árvore de decisão

---

**Entrada:** Conjunto de dados de treinamento  $\mathcal{D}$  de  $n$  padrões; cada padrão com  $m$  atributos; Critério  $J(\mathcal{D})$  de ramificação, aqui, por exemplo, ganho de entropia  $\Delta H$

**Resultado:** Árvore de decisão  $\mathcal{A}$

```

1 Algoritmo Árvore( $\mathcal{D}, J$ )
2    $\mathcal{A} \leftarrow$  raiz da árvore vazia
3    $\mathcal{A} \leftarrow$  geraÁrvore( $\mathcal{A}, \mathcal{D}, J$ )
4   retorna  $\mathcal{A}$ 

```

```

1 Função geraÁrvore( $\mathcal{A}, \mathcal{D}, J$ )
2   Calcule se nó é folha
3   se nó for folha então
4     retorna folha
5   senão
6     Calcule a entropia  $H(\mathcal{C})$  do conjunto todo  $\mathcal{D}$ , (eq. 8.28)
7     para  $j \leftarrow 1$  até  $m$  faz
8       //Para todos os atributos  $x_j$ 
9       para  $r \leftarrow 1$  até  $q$  faz
10      //Para todos os valores  $v_r$  do atributo
11      Calcule a entropia  $H(\mathcal{C}|v_r)$  do ramo, (eq. 8.29)
12      fim
13      Calcule a entropia  $H(\mathcal{C}|x_j)$  do atributo, (eq. 8.31)
14      Calcule o ganho de entropia  $\Delta H(\mathcal{C}|x_j)$ , (eq. 8.32)
15    fim
16    Determine o vencedor, o atributo com o maior ganho de entropia e o coloque no
17    nó atual  $\mathcal{A}$ 
18     $j^* \leftarrow \arg \max_j \Delta H(\mathcal{C}|x_j)$ 
19    para  $r \leftarrow 1$  até  $q$  faz
20      //Para todos os valores  $v_r$  do atributo vencedor  $x_{j^*}$ 
21       $\mathcal{D}_r \leftarrow$  filtre todos os  $n_r$  padrões com o valor  $v_r$ 
22       $\mathcal{A}_r \leftarrow$  geraÁrvore( $\mathcal{A}, \mathcal{D}_r, J$ ) //Ramo, Subárvore
23      Coloca subárvore  $\mathcal{A}_r$  como ramo do nó superior  $\mathcal{A}$ 
24    fim
25    retorna  $\mathcal{A}$ 
26
27  fim
28  retorna  $\mathcal{A}$ 

```

---

### Critérios Alternativos de Separação de Subconjuntos de Padrões

Até o momento consideramos somente o ganho de entropia definido na (eq. 8.32) para guiar a ramificação no processo da geração recursiva da árvore. Obviamente existem alternativas de métricas quantitativas que decidem qual atributo ganha o direito de definir os ramos.

**Razão de Entropia de Atributo:** Em uma tentativa de reduzir a influência de atributos  $x_j$  que têm muitos valores diferentes, ou seja, uma cardinalidade  $|\mathcal{X}_j|$  alta na (eq. 8.23), a entropia do atributo pode ser ainda normalizada pela divisão de um termo que penaliza muitos valores diferentes do domínio. O *valor intrínseco* de cada atributo pode ser definido como

$$\text{IV}(x) \stackrel{\text{def}}{=} \sum_{r=1}^q P(v_r) \log_2 P(v_r) = \mathbb{E} \left[ \frac{1}{\log_2 P(v_r)} \right]. \quad (8.33)$$

A razão de entropia do atributo da (eq. 8.31) então modifica para

$$\tilde{H}(\mathcal{C}|x) \stackrel{\text{def}}{=} \frac{H(\mathcal{C}|x)}{\text{IV}(x)}, \quad (8.34)$$

e o ganho de entropia da (eq. 8.32), considerando esta normalização, muda para

$$\Delta\tilde{H}(\mathcal{C}) \stackrel{\text{def}}{=} H(\mathcal{C}) - \tilde{H}(\mathcal{C}|x). \quad (8.35)$$

Já calculamos o valor intrínseco IV(Vento) do atributo “Vento” no exemplo 8.3.

**Índice Gini de Diversidade (Impuridade):** Se todos os  $n$  padrões das  $c$  classes forem divididos igualmente por um atributo, ou seja,  $P(\mathcal{C}_k) = \frac{c}{n}$ , para todas as classes, seria muito desfavorável para tomar uma decisão. Se a uma única classe forem atribuídos todos os  $n$  amostras, seria o caso ideal. Então um critério que valoriza um desequilíbrio das classes e penaliza a igualdade das classes, atende os requisitos para ramificar. O índice Gini de impuridade foi proposto em [11]. Em analogia à entropia, podemos definir

$$i(\mathcal{C}) \stackrel{\text{def}}{=} \sum_{k=1}^c \sum_{\substack{\ell=1 \\ \ell \neq k}}^c P(\mathcal{C}_k)P(\mathcal{C}_\ell) = 1 - \sum_{k=1}^c [P(\mathcal{C}_k)]^2 \quad \begin{array}{l} \text{Impuridade a priori das classes} \\ \text{compare com (eq. 8.28)} \end{array} \quad (8.36a)$$

$$i(\mathcal{C}|v_r) \stackrel{\text{def}}{=} \sum_{k=1}^c [1 - P(\mathcal{C}_k|v_r)]^2 \quad \begin{array}{l} \text{Impuridade de ramo} \\ \text{compare com (eq. 8.29)} \end{array} \quad (8.36b)$$

$$i(\mathcal{C}|x) \stackrel{\text{def}}{=} \sum_{r=1}^q P(v_r) \sum_{k=1}^c [1 - P(\mathcal{C}_k|v_r)]^2 \quad \begin{array}{l} \text{Impuridade do atributo} \\ \text{compare com (eq. 8.31)} \end{array} \quad (8.36c)$$

$$\Delta i(\mathcal{C}) \stackrel{\text{def}}{=} i(\mathcal{C}) - i(\mathcal{C}|x) \quad \begin{array}{l} \text{Ganho de Impuridade} \\ \text{compare com (eq. 8.32)} \end{array} \quad (8.36d)$$

$$(8.36e)$$

■ **Exemplo 8.6** Em analogia ao cálculo do ganho da entropia, temos o ganho do índice Gini de

diversidade para o atributo  $x_1 = \text{Previsão de tempo}$

$$\begin{aligned}
\Delta i(\mathcal{C}) &= i(\mathcal{C}) - i(\mathcal{C}|x_1) = i(\mathcal{C}) - i(\mathcal{C}|\text{Previsão de tempo}) \\
&= \left(1 - \sum_{k=1}^c [P(\mathcal{C}_k)]^2\right) - \sum_{r=1}^3 P(v_r) i(\mathcal{C}|v_r) \\
&= \left(1 - [P(\text{Sim})]^2 - [P(\text{Não})]^2\right) \\
&\quad - P(\text{Ensolarado}) \cdot i(\mathcal{C}|\text{Ensolarado}) \\
&\quad - P(\text{Nublado}) \cdot i(\mathcal{C}|\text{Nublado}) \\
&\quad - P(\text{Chuva}) \cdot i(\mathcal{C}|\text{Chuva}) \\
&= \left(1 - \left[\frac{9}{14}\right]^2 - \left[\frac{5}{14}\right]^2\right) \\
&\quad - \frac{5}{14} \left\{1 - \left[\frac{3}{5}\right]^2 - \left[\frac{2}{5}\right]^2\right\} - \frac{4}{14} \left\{1 - \left[\frac{4}{4}\right]^2 - \left[\frac{0}{4}\right]^2\right\} - \frac{5}{14} \left\{1 - \left[\frac{2}{5}\right]^2 - \left[\frac{3}{5}\right]^2\right\} \\
&= \frac{90}{196} - \frac{5}{14} \cdot \frac{12}{25} - \frac{4}{14} \cdot 0 - \frac{5}{14} \cdot \frac{12}{25} = \frac{45}{98} - \frac{6}{35} - \frac{6}{35} = \frac{57}{490} = 0.116.
\end{aligned}$$

Calcule como exercício para os restantes três atributos o ganho do índice Gini de diversidade. ■

### Discretização de Características Contínuas

Já observamos na geração de uma árvore de decisão para as flores iris no figura 8.3 que a ramificação foi feita através de uma bifurcação, definindo um limiar e criando um valor nominal “abaixo ou igual” e “acima”. Dessa maneira, a geração recursiva da árvore segue o mesmo esquema ilustrado no algoritmo 15. A questão em aberto é o valor do limiar. Na raiz da árvore na figura 8.3, o vencedor, usando o critério ganho do índice Gini foi o atributo criado

$$x_j \stackrel{\text{def}}{=} \text{“Relação com o limiar } 0.8 \text{ da largura da pétala”},$$

com dois valores nominais possíveis “abaixo ou igual” e “acima”. Como estamos considerando uma única dimensão, ou seja, uma única característica do conjunto das quatro características das flores, a tarefa é bastante simples. Imaginemos novamente que o processo da geração da árvore esteja em um nó qualquer com  $n$  amostras. Note que esse valor é a quantidade das amostras que foram filtradas até esse nó, no exemplo das flores  $n = 150$  somente na raiz. Para uma característica  $x \in \mathbb{R}$  original temos então  $n$  valores contínuos. O esquema de discretização de uma característica  $x \in \mathbb{R}$ , na sua versão mais simples está explicado no algoritmo 16

■ **Exemplo 8.7** Na tabela 8.3, a temperatura, além dos valores nominais, tem alternativamente 14 valores contínuos  $[85, 80, 83, 70, 68, 65, 64, 72, 69, 75, 75, 72, 81, 71]$ . São valores inteiros, porém o importante é a existência de uma ordem imposta pela métrica da diferença aritmética entre dois valores. Após eliminar dois valores duplicados e ordenar em ordem crescente, temos a sequência de doze valores  $[64, 65, 68, 69, 70, 71, 72, 75, 80, 81, 83, 85]$ . Os limiares definidos pelo meio entre dois valores consecutivos são  $[64.5, 66.5, 68.5, 69.5, 70.5, 71.5, 73.5, 77.5, 80.5, 82, 84]$ . Os ganhos de entropia, como exemplo do critério de ramificação, pela (eq. 8.31) e (eq. 8.32) são  $[0.0477, 0.0103, 0.0005, 0.015, 0.0453, 0.0013, 0.0013, 0.0251, 0.0005, 0.0103, 0.1134]$ . Dessa maneira, o vencedor é  $\mu_{11} = 84$  entre  $v_{11} = 83$  e  $v_{12} = 85$  com o valor do critério  $J_j = J(\mathcal{C}|x_{r^*}) = J(\mathcal{C}|x_{11}) = \frac{11}{12} \cdot J(\mathcal{C}| \leq) + \frac{1}{12} \cdot J(\mathcal{C}| >)$ . ■

### 8.3.2 Relação entre Entropia, Informação Mútua e Divergência Kullback–Leibler

O critério de ramificação de um nó da árvore de decisão tem como objetivo de otimizar a separação das classes, dado um atributo. Estudamos o ganho de entropia ou índice Gini de diversidade. A

---

**Algoritmo 16:** Discretização de características contínuas para atributos nominais

---

**Entrada:** Conjunto de  $n$  valores  $\tilde{x}_{j_1}, \dots, \tilde{x}_{j_n}$  de uma característica contínua  $\tilde{x}_j \in \mathbb{R}$  de  $n$  padrões em um nó da árvore de decisão ; um critério  $J$  de ramificação específico do valor  $v$  de um atributo nominal, por exemplo a entropia  $H(\mathcal{C}|v)$  do ramo, (eq. 8.29), ou a impuridade Gini do ramo  $i(\mathcal{C}|v)$ , (eq. 8.36b), ou outros critérios.

**Resultado:** Critério  $J_j$  do atributo nominal discretizado  $x_j$  definido por um limiar  $\tau_j$

- 1 Dos  $n$  valores, determine os  $q \leq n$  valores únicos, ou seja, não há mais valores repetidos ;
  - 2 Ordene o conjunto dos  $q$  valores únicos para que haja uma sequência  $\tilde{x}_1 < \tilde{x}_2 < \dots < \tilde{x}_{r-1} < \tilde{x}_r < \tilde{x}_{r+1} < \dots < \tilde{x}_q$  ;
  - 3 **para**  $r \leftarrow 1$  até  $q - 1$  **faça**
    - //Para todos os valores  $v_r$  únicos do atributo
    - 4  $\mu_r \leftarrow \frac{\tilde{x}_r + \tilde{x}_{r+1}}{2}$  //Valor médio do intervalo de dois valores consecutivos
    - 5 Divida os  $n$  valores em duas metades, conforme o limiar  $\mu_r$ ;
    - 6 Defina o atributo novo  $x_r$  com o domínio binário de valores  $v_r \in \{\leq, >\}$  ;
    - 7 Conte a quantidade  $n_{\leq}$  dentro dos  $q$  valores abaixo do limiar ou igual ;
    - 8 Conte a quantidade  $n_{>}$  dentro dos  $q$  valores acima do limiar ;
    - 9 Calcule o critério  $J(\mathcal{C}|\leq)$  do ramo inferior ;
    - 10 Calcule o critério  $J(\mathcal{C}|>)$  do ramo superior ;
    - 11 Calcule o critério  $J(\mathcal{C}|x_r)$  do atributo candidato, veja (eq. 8.31) e (eq. 8.36c)
  - 12  $J(\mathcal{C}|x_r) \leftarrow \frac{n_{\leq}}{q} \cdot J(\mathcal{C}|\leq) + \frac{n_{>}}{q} \cdot J(\mathcal{C}|>)$
  - 13 **fim**
  - 14 Determine o índice  $r^*$  que maximizou o critério  $r^* \leftarrow \arg \max_r J(\mathcal{C}|x_r)$  ;
  - 15  $J_j \leftarrow J(\mathcal{C}|x_{r^*})$
-

justificativa atrás desses critérios é um conceito mais geral, a *dependência probabilística*. Mede-se uma diferença de informação sobre as classes, antes e depois saber mais sobre um atributo. “Antes” significa considerar a distribuição das classes independentemente da distribuição dos valores do atributo, ou seja, a probabilidade a priori  $P(\mathcal{C})$  da variável aleatória da classe. “Após” significa incluir o conhecimento sobre a probabilidade dos valores do atributo  $x$ , ou seja, a probabilidade *condicional* da classe  $P(\mathcal{C}|x)$ , dado o atributo. Relembre as definições do contexto da regra de Bayes definição 7.1.1 na pág. 188.

Temos que definir alguma métrica que quantifica essa dependência probabilística. Usamos o símbolo  $\Delta$  (“Delta maiúsculo”) para denominar essa métrica, expressando que se trata de uma diferença ou **discrepância**. Os argumentos desta métrica são a probabilidade a priori da classe  $P(\mathcal{C})$  e a probabilidade condicional da classe, dado o atributo  $P(\mathcal{C}|x)$ . A probabilidade condicional ainda carrega implicitamente a probabilidade a priori do atributo  $P(x)$ , pois, através da (eq. 7.1) na pág. 188 temos a probabilidade conjunta

$$P(\mathcal{C}, x) = P(\mathcal{C}|x) \cdot P(x), \quad (8.37)$$

necessitada nas contas. Alternativamente, podemos considerar a distância entre a probabilidade conjunta  $P(\mathcal{C}, x)$  e o produto  $P(\mathcal{C}) \cdot P(x)$  das duas probabilidades marginais  $P(\mathcal{C})$  e  $P(x)$ . Consequentemente, por rigor matemático temos que definir um *funcional*, que é “função de uma função”, pois os argumentos são probabilidades que, por sua vez, são funções<sup>7</sup>

**Definição 8.3.6 — Dependência Probabilística como Critério de Ramificação em Árvore de Decisão.** Dada a variável aleatória discreta  $\mathcal{C}$  que expressa a associação de um padrão a uma classe e a variável aleatória discreta  $x$  de um atributo nominal, a dependência probabilística entre a probabilidade a priori  $P(\mathcal{C})$  da classe e a probabilidade condicional da classe  $P(\mathcal{C}|x)$ , dado o atributo  $x$  com probabilidade a priori  $P(x)$  dos seus valores, é o funcional

$$\Delta(P(\mathcal{C}), P(\mathcal{C}|x)) = \Delta(P(\mathcal{C}, x), P(\mathcal{C}) \cdot P(x)) \quad (8.38)$$

$$= \begin{cases} 0, & \text{se } \mathcal{C} \text{ for independente de } x \\ > 0, & \text{caso contrário,} \end{cases} \quad (8.39)$$

Existem várias métricas para quantificar a dependência probabilística. Todos têm que ter um valor zero, se não existir dependência, equivalente a

$$P(\mathcal{C}, x) = P(\mathcal{C}) \cdot P(x) \iff P(\mathcal{C}|x) = P(\mathcal{C}).$$

Se existir dependência, o valor é positivo, porém o seu máximo não é necessariamente bem definido, como, por exemplo pelo valor 1.

A *Informação Mútua* é um conceito clássico da teoria de informação de Shannon [99]. A sua definição se encaixa nos moldes de uma dependência probabilística. Na seguinte definição, já usamos para o nome das variáveis e os seus domínios o exemplo da classificação por um atributo.

**Definição 8.3.7 — Informação Mútua, Variável Aleatória Discreta.** Sejam dadas duas variáveis aleatórias discretas  $\mathcal{C}$  e  $x$  com valores dos seus respectivos domínios  $\mathcal{C} \in \{\mathcal{C}_1, \dots, \mathcal{C}_k, \dots, \mathcal{C}_c\}$  e  $x \in \{v_1, \dots, v_r, \dots, v_q\}$  e sejam dadas as probabilidades conjunta  $P(\mathcal{C}, x)$ , e marginais sobre as classes  $P(\mathcal{C})$  e os valores do atributo  $P(x)$ , veja a tabela 8.6. A informação mútua das duas

<sup>7</sup>Para ser mais rigoroso ainda, teríamos que mudar o nome de cada função usada. Por exemplo, a função  $P(\mathcal{C})$  que determina a distribuição da classe não é a mesma função que determina a distribuição do atributo. Lembre que no exemplo das duas densidades da figura 1.3 na pág. 18, a Gaussiana e a uniforme ambas eram chamadas de  $p(x)$ , mas teriam que ser chamadas  $p_1(x)$  e  $p_2(x)$ , por exemplo. Tivemos  $p(0) = 1/\sqrt{2\pi}$  para a Gaussiana e  $p(0) = 1$  para a densidade uniforme, então somente pelo cálculo  $p(x)$  não sabemos de que função estamos falando.

variáveis é

$$I(\mathcal{C};x) \stackrel{\text{def}}{=} \sum_{k=1}^c \sum_{r=1}^q P(\mathcal{C}_k, v_r) \log \frac{P(\mathcal{C}_k, v_r)}{P(\mathcal{C}_k) \cdot P(v_r)} \quad (8.40)$$

**Corolário 8.3.1 — Equivalência do Ganho de Entropia e da Informação Mútua.** Usando a base 2 no logaritmo na definição 8.3.7, a informação mútua e o ganho de entropia da definição 8.3.5 são idênticos.

*Demonstração.* <sup>a</sup>

$$\begin{aligned} \Delta H(\mathcal{C}) &\stackrel{8.32}{=} H(\mathcal{C}) - H(\mathcal{C}|x) \\ &\stackrel{8.31}{=} - \sum_{k=1}^c P(\mathcal{C}_k) \log_2 P(\mathcal{C}_k) - \sum_{r=1}^q P(v_r) H(\mathcal{C}|v_r) \\ &\stackrel{8.29}{=} - \sum_{k=1}^c \left\{ \sum_{r=1}^q P(\mathcal{C}_k, v_r) \right\} \log_2 P(\mathcal{C}_k) - \sum_{r=1}^q P(v_r) \left\{ - \sum_{k=1}^c P(\mathcal{C}_k|v_r) \log_2 P(\mathcal{C}_k|v_r) \right\} \\ &\stackrel{8.37}{=} - \sum_{k=1}^c \sum_{r=1}^q P(\mathcal{C}_k, v_r) \log_2 P(\mathcal{C}_k) + \sum_{k=1}^c \sum_{r=1}^q P(v_r) \frac{P(\mathcal{C}_k, v_r)}{P(v_r)} \log_2 \frac{P(\mathcal{C}_k, v_r)}{P(v_r)} \\ &= \sum_{k=1}^c \sum_{r=1}^q P(\mathcal{C}_k, v_r) \{ \log_2 P(\mathcal{C}_k, v_r) - \log_2 P(\mathcal{C}_k) - \log_2 P(v_r) \} \\ &= \sum_{k=1}^c \sum_{r=1}^q P(\mathcal{C}_k, v_r) \log \frac{P(\mathcal{C}_k, v_r)}{P(\mathcal{C}_k) \cdot P(v_r)} \end{aligned} \quad (8.41)$$

■

<sup>a</sup>Relembrando,  $\log(a/b) = \log a - \log b$ .

Mais um conceito frequentemente usado em aprendizado de máquina , mais uma vez usando os nomes do contexto da classificação, é a

**Definição 8.3.8 — Divergência Kullback–Leibler.** Sejam dadas duas variáveis aleatórias discretas  $\mathcal{C}$  e  $x$  com valores dos seus respectivos domínios  $\mathcal{C} \in \{\mathcal{C}_1, \dots, \mathcal{C}_k, \dots, \mathcal{C}_c\}$  e  $x \in \{v_1, \dots, v_r, \dots, v_q\}$  e sejam dadas as probabilidades conjunta  $P(\mathcal{C}, x)$ , e marginais sobre as classes  $P(\mathcal{C})$  e os valores do atributo  $P(x)$ , veja a tabela 8.6. A divergência Kullback–Leibler entre a probabilidade conjunta e o produto das probabilidades marginais é idêntica à informação mútua da definição 8.3.7

$$D_{\text{KL}}(P(\mathcal{C}, x) || P(\mathcal{C}_k) \cdot P(v_r)) \stackrel{\text{def}}{=} I(\mathcal{C}; x). \quad (8.42)$$

Repare que a divergência Kullback–Leibler não é simétrica, ou seja

$$D_{\text{KL}}(x||y) \neq D_{\text{KL}}(y||x),$$

portanto não satisfaz o pré-requisito de ser uma *distância*.

Um último comentário em relação ao critérios de ramificação de árvores de decisão, é que a teoria da análise entre um par de variáveis aleatórias independentes e dependentes é muito antiga. No

contexto de estatística já foram definidos critérios que medem a associação entre dois atributos e duas classes, baseados em tabelas de contingência [116]. Retornaremos a esse assunto no contexto de *seleção de características*.

### 8.3.3 Limitações na Construção de Árvores de Decisão

Considere o sistema de coordenadas cartesiano de duas características,  $x_1$  e  $x_2$  e duas classes  $\mathcal{C}_1$  e  $\mathcal{C}_2$ . Seja a fronteira que separa as duas classes a diagonal do plano, definido<sup>8</sup> pela equação  $x_1 = x_2$ . Isto significa que todos os padrões que pertencem a uma classe ficam acima da diagonal e os padrões que pertencem a outra classe ficam abaixo da diagonal (ou em cima da reta). Podemos manualmente costurar uma árvore de decisão que tem um único nó que separa perfeitamente as classes, veja a figura 8.5.

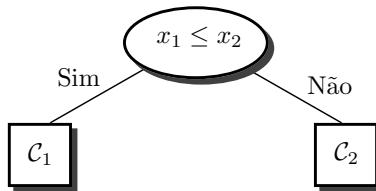


Figura 8.5: Árvore de decisão perfeita, manualmente concebida.

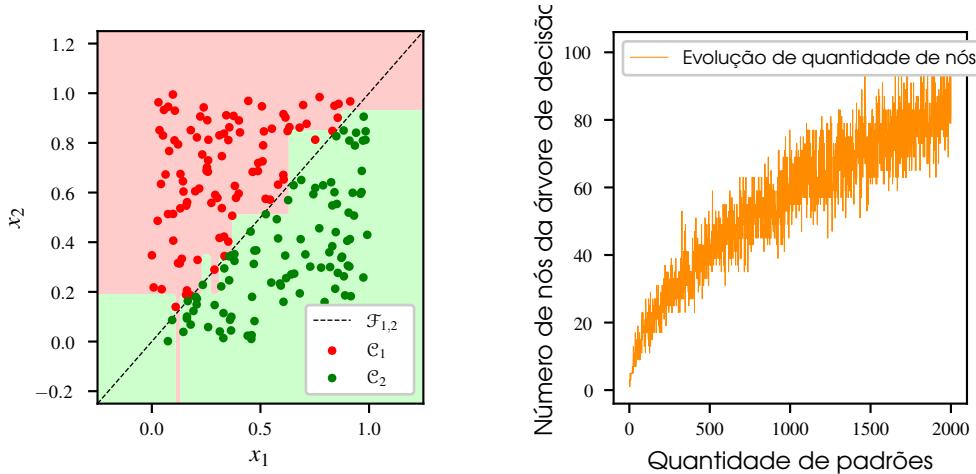
A figura 8.6 mostra, o que acontece se um conjunto finito de padrões de treinamento for sujeito a um algoritmo de geração de uma árvore de decisão. Na figura 8.6a,  $n = 200$  padrões foram gerados arbitrariamente por uma densidade uniforme nas duas dimensões  $x_1$  e  $x_2$ . Se um padrão caiu acima da diagonal, ele foi rotulado como a primeira classe, abaixo da diagonal, ou em cima dela, foi rotulado como a segunda classe. Reconhece-se claramente que a fronteira é aproximada por segmentos paralelos aos eixos  $x_1$  e  $x_2$ . Este fato se deve a incapacidade do algoritmo de reconhecer relações multidimensionais entre os atributos. Em cada ramificação, um único atributo é usado para definir como o conjunto atual dos padrões é dividido. Por exemplo, usando o ganho de entropia como critério na (eq. 8.32), a (eq. 8.31) considera um único  $x$  é analisado. Em um atributo nominal que originalmente era uma característica contínua, como neste exemplo bidimensional, a visão limitada do algoritmo enxerga somente um dos eixos,  $x_1$  ou  $x_2$ , mas nunca as duas características em conjunto. A consequência é que após a geração de uma, neste caso, bifurcação, o conhecimento anterior da análise do atributo se perdeu, e uma nova análise de um único atributo tem que se inicializar.

Uma das justificativas de usar uma árvore de decisão no processo de classificação é a possibilidade para um ser humano de entender as ramificações. Este simples exemplo mostra que essa vantagem vale somente em casos específicos, com poucos exemplos. Se árvores de decisão forem usados com quantidades muito grandes de dados, em “Big Data”, a vantagem da explicabilidade dos resultados é bastante questionável. Neste exemplo ilustrativo, um análise de componentes principais como pré-processamento teria criado uma nova característica  $y$  que teria produzido uma árvore simples, porém esse pré-processamento não é automático e teria que ser introduzido manualmente.

## 8.4 Árvores de Regressão

A estrutura de uma árvore não se limita à criação de um classificador que tem como resultado o valor nominal da classe. Podemos criar uma *árvore de regressão*. Para a união dos dois conceitos,

<sup>8</sup>Em termos de um classificador linear em 2-D, usando a (eq. 2.82) na pág. 67,  $w_1x_1 + w_2x_2 + b = 0$ , temos para os valores dos parâmetros,  $w_1 = 1, w_2 = -1, b = 0$ .



(a) Dispersão de  $n = 200$  padrões do conjunto de treinamento. As regiões de decisão geradas divergem da fronteira  $x_1 - x_2 \geq 0$  analiticamente definida

(b) Crescimento da quantidade de nós gerados em função da quantidade de padrões no conjunto de treinamento de no máximo  $n = 2000$ .

Figura 8.6: Limitação na geração de uma árvore de decisão, devida a análise unidimensional na ramificação dos nós.

$x_i$	0.4	0.9	1.5	2.3	2.9	3.1	3.7
$y_i$	0.7	1.0	0.8	0.9	1.4	2.1	2.4

Tabela 8.7: Conjunto de dados de treinamento mínimo para ilustrar regressão.

frequentemente usa-se o acrônimo CART (*Classification and Regression Trees*), baseado no trabalho pioneiro de Breiman et al. [11]. Já temos praticamente toda a teoria necessária para definir uma árvore de regressão. As diferenças principais são o critério de ramificação e o resultado da regressão, dado um valor contínuo qualquer. Considere novamente o conjunto de treinamento  $\mathcal{D}$  do exemplo ilustrativo mínimo da figura 2.1 na pág. 34, e da tabela 2.1 na pág. 34, repetidos aqui. São  $n = 7$  pares  $(x_i, y_i)$ ,  $i = 1, \dots, n$ . Dado o valor de entrada  $x_i$ , o conjunto de treinamento tem um valor esperado  $y_i$ . A tarefa é a aprendizagem da função desconhecida

$$y = f(x),$$

que fornece para uma entrada qualquer  $x$  o valor de saída estimado  $\hat{y}$ . O modelo linear  $y = wx + b$  definido na (eq. 2.4) na pág. 32 é um modelo paramétrico com os parâmetros  $w$  e  $b$ . Uma árvore de regressão na versão mais simples é um modelo paramétrico, linear por partes. O caso extremo modela cada par do conjunto de treinamento  $(x_i, y_i)$  como uma função linear própria

$$y_i = 1 \cdot x_i + 0 = x_i.$$

Isso acontece, se a árvore de regressão é gerada até que todos os nós sejam folhas com um único valor de saída. Esse valor é exatamente um dos valores de saída  $y_i$  do conjunto de treinamento. No caso do exemplo ilustrativo mínimo teríamos então uma árvore com seta folhas, cada uma com o valor de saída. Em geral, uma árvore de regressão pode fazer uma regressão múltipla, ou seja, com entradas e saídas multidimensionais

$$\mathbf{y} = \mathbf{f}(\mathbf{x}),$$

veja a figura 2.4 que mostra a multidimensionalidade da Máquina Linear. No caso da árvore de regressão porém a regressão não necessariamente é linear, a função  $\mathbf{f}(\mathbf{x})$  pode representar em princípio qualquer mapeamento.

A construção da árvore de regressão segue o esquema genérico do algoritmo 15. O procedimento da discretização de características do algoritmo 16 pode ser usado também para definir um limiar que gera uma bifurcação no nó atual. Bifurcação, pois teremos uma árvore binária com dois filhos em cada um dos nós, exceto nos nós terminais. A diferença está no critério de ramificação. No algoritmo 15, para uma árvore de decisão, usa-se como representante dos critérios o ganho de entropia, mas lembre-se que podemos usar outros critérios. No caso de uma árvore de regressão, temos que escolher um critério apropriado para características contínuas. No contexto das funções de perda já conhecemos o Erro Quadrático Médio (EQM), (*Mean Squared Error, MSE*) que mede uma aproximação do valor esperado entre  $n$  saídas esperadas. como

$$\text{EQM} = \frac{1}{n} \sum_{i=1}^n \|\mathbf{y}_i - \hat{\mathbf{y}}_i\|^2 = \frac{1}{n} \sum_{i=1}^n \|\mathbf{y}_i - \mathbf{f}(\mathbf{x}_i)\|^2.$$

Não podemos usar esse critério diretamente na definição da bifurcação. O raciocínio é que temos dois ramos, cada um com um valor esperado próprio. Podemos calcular o EQM dos valores do ramo em relação ao seu respectivo valor esperado.

**Definição 8.4.1 — Critério de Bifurcação de Árvore de Regressão.** Seja dado, no nó atual da árvore, um conjunto de treinamento  $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n = \{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_n, \mathbf{y}_n)\}$ , composto por  $n$  pares de padrões de entrada  $\mathbf{x}_i \in \mathbb{R}^m$  de dimensão  $m$  e  $\mathbf{y}_i \in \mathbb{R}^c$  de dimensão  $c$ . Para cada componente  $x_j$  do vetor  $\mathbf{x}$  de entrada,  $j = 1, \dots, m$  podemos calcular o critério de bifurcação, portanto considera-se somente a  $j$ -ésima componente de cada padrão de entrada, dando origem ao conjunto de treinamento  $\mathcal{D}_j = \{(x_{i,1}, y_1), \dots, (x_{i,q}, y_q)\}$ , composto pelos  $q$  valores únicos dos  $n$  valores, já ordenado pelo valor, portanto  $x_{i,1} < \dots < x_{i,q}$ . Use o algoritmo 16 para determinar o limiar  $\mu_r$  que divide os  $n$  pares em dois ramos, modificando os cálculos do critério do ramo inferior e superior na linha 9 e linha 10 como

$$J(\mathcal{C} | \leq) = \frac{1}{r} \sum_{i=1}^r \|\mathbf{y}_i - \bar{\mathbf{y}}_{\text{esq}}\|^2, \quad (8.43)$$

$$J(\mathcal{C} | >) = \frac{1}{q-r} \sum_{i=r+1}^q \|\mathbf{y}_i - \bar{\mathbf{y}}_{\text{dir}}\|^2, \quad (8.44)$$

onde

$$\bar{\mathbf{y}}_{\text{esq}} \stackrel{\text{def}}{=} \frac{1}{r} \sum_{i=1}^r \mathbf{y}_i$$

é a média (valor esperado aproximado) dos valores do ramo esquerdo, e

$$\bar{\mathbf{y}}_{\text{dir}} \stackrel{\text{def}}{=} \frac{1}{q-r} \sum_{i=r+1}^q \mathbf{y}_i$$

é a média (valor esperado aproximado) dos valores do ramo direito. Uma alternativa do Erro Quadrático Médio (EQM) relativo ao termo

$$\|\mathbf{y}_i - \bar{\mathbf{y}}\|^2$$

na (eq. 8.43) e na (eq. 8.44), é o Erro Absoluto Médio (EAM), (*Mean Absolute Error, MAE*) substituindo pelo termo

$$||\mathbf{y}_i - \bar{\mathbf{y}}||.$$

Note que a variável aleatória nominal  $\mathcal{C}$  é implicitamente definida pelo limiar. Os  $r$  elementos da esquerda formam a classe  $\mathcal{C}_1 \stackrel{\text{def}}{=} \mathcal{C}_{\text{esq}}$  e os  $q - r$  elementos da direita a classe  $\mathcal{C}_2 \stackrel{\text{def}}{=} \mathcal{C}_{\text{dir}}$ .

Se deixarmos desenvolver a ramificação da árvore normalmente, teremos como folhas exatamente todos os valores  $\mathbf{y}_i$  do conjunto inicial de treinamento. No caso da árvore de decisão tivemos como saída um valor nominal, o rótulo de uma classe. No caso da árvore de regressão temos, dado uma entrada  $\mathbf{x}$ , uma saída contínua unidimensional  $\hat{y} = f(\mathbf{x})$  ou multidimensional  $\hat{\mathbf{y}} = \mathbf{f}(\mathbf{x})$ . No exemplo ilustrativo da tabela 8.7 tivemos somente uma única dimensão de entrada, e única dimensão de saída, portanto o vetor que representa as características do padrão tem somente uma componente  $\mathbf{x} = [x_1] = x$  e a saída também tem somente uma componente  $\mathbf{y} = [y_1] = y$ . A figura 8.7 mostra a árvore de regressão gerada pelo critério de bifurcação da definição 8.4.1 no algoritmo 15, usando o conjunto da tabela 8.7, junto com o resultado da regressão. Como temos uma única característica,  $x$  sempre aparece nos nós. A árvore foi completamente gerada com todos os sete valores de saída  $y_i$  como folhas. Então no resultado da regressão na figura 8.7b, todos os valores desejados  $y_i$  coincidem com os valores explicados  $\hat{y}_i$ . Além disso, na figura 8.7b, mostram-se todas as saídas  $\hat{y} = f(x)$  para qualquer entrada  $x \in [x_{\min}, x_{\max}]$  no intervalo entre o menor e maior valor  $x_i$  do conjunto de treinamento. Repare que as saídas  $\hat{y}$  têm o mesmo valor das folhas. O início e fim da coincidência de  $\hat{y}$  com uma saída  $y_i$  no conjunto de treinamento é sempre no meio

$$\mu_i = \frac{x_i + x_{i+1}}{2}$$

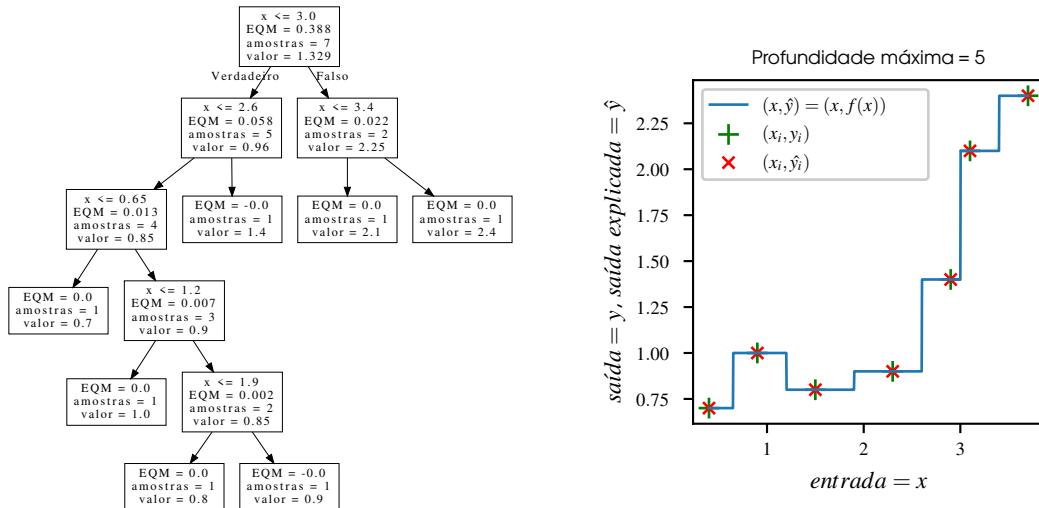
entre dois valores consecutivos  $x_i$  e  $x_{i+1}$ . Nesse sentido o valor da regressão é determinado pelo algoritmo do 1-Vizinho-Mais-Próximo que fornece a saída  $y_i$  existente no conjunto de treinamento, procurando a entrada correspondente mais próxima  $x_i$  da entrada arbitrária  $x$ .

Se decidirmos não levar a geração da árvore de regressão até o fim, estamos aplicado a técnica da *poda* (*pruning*) da árvore. O poda é um método para evitar o sobreajuste (*overfitting*). Lembre da figura 2.1 na pág. 34 que a regressão, que nesse caso era uma regressão linear, aprende o modelo, nesse caso uma reta. Ao contrário, no modelo, implicitamente oculto na árvore de regressão da figura 8.7b, o ruído nos padrões persiste, devido à saída idêntica dos valores esperados  $y_i$  e explicados  $\hat{y}_i$ . Posteriormente, em outros modelos de regressão e classificação, veremos que um bom modelo no aprendizado de máquina tem que evitar esse tipo de sobreajuste. Então a poda não retorna exatamente um dos elementos no conjunto de saída, mas algum valor *suavizado*. Vamos definir o cálculo do valor de saída da regressão.

**Definição 8.4.2 — Predição de Árvore de Regressão.** Dada uma árvore de regressão gerada, dado um valor de entrada  $\mathbf{x}$ , após seguindo as bifurcações até um nó final com o conjunto de  $n$  padrões  $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \dots, (\mathbf{x}_n, \mathbf{y}_n)\}$  neste nó, a predição  $\hat{y} = f(\mathbf{x})$  do modelo é

$$\mathbf{f}(\mathbf{x}) = \begin{cases} \mathbf{y}_1, & \text{se } n = 1, \mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1)\} \quad (\text{padrão único no nó}) \\ \bar{\mathbf{y}} = \frac{1}{n} \sum_{i=1}^n \mathbf{y}_i, & \text{se } n > 1 \quad (\text{mais que um padrão no nó}) \end{cases} \quad (8.45)$$

Chame-se novamente atenção pelo fato que a quantidade  $n$  dos padrões em cada nó difere da quantidade inicial do conjunto inteiro de dados. Pelas bifurcações, os conjuntos em cada nó são subdivididos, até um nó final que tem um ou mais padrões, sem poda ou com poda, respectivamente.



(a) Árvore de regressão gerada com profundidade máxima.

(b) Entradas e saídas de uma árvore de regressão. Saídas originais  $y_i$  do conjunto de treinamento, dados ajustados  $\hat{y}_i$  do conjunto de treinamento, e saídas de todos os valores do intervalo  $[x_{\min}, x_{\max}] = [0.4, 3.7]$ Figura 8.7: Conjunto de treinamento e árvore de regressão de profundidade máxima. Coincidência  $y_i = \hat{y}_i$ , pois cada saída individual é representada por uma folha da árvore de regressão.

Um dos critérios de poda é a *profundidade máxima*, ou seja, a quantidade máxima de nós a serem percorridos até chegar até uma folha. No caso do exemplo simples na figura 8.7 esse valor é  $p_{\max} = 5$ . Se um nó atingiu a profundidade máxima sem ser uma folha, ele carrega um conjunto de  $n > 1$  de amostras, resultado da divisão recursiva do conjunto de treinamento original.

**■ Exemplo 8.8** A figura 8.8 mostra a árvore de regressão gerada com a profundidade máxima de  $p_{\max} = 1$ , usando o conjunto ilustrativo da tabela 8.7. Para esta árvore da figura 8.8 temos a seguinte situação para calcular a predição: O conjunto de treinamento de originalmente  $n = 7$  padrões foi subdividido nos dois filhos em  $n = 5$  e  $n = 2$  padrões respectivamente. Em relação à predição de um valor qualquer  $x$ , considerando a regra na definição 8.4.2, temos para o filho esquerdo, em caso de  $x \leq 3.0$ , o resultado  $\hat{y} = f(x) = \bar{y} = \frac{1}{5}(0.7 + 1.0 + 0.8 + 0.9 + 1.4) = 0.96$ . Para o filho direito, temos  $\hat{y} = f(x) = \bar{y} = \frac{1}{2}(2.1 + 2.4) = 2.25$ . Na figura 8.8 reconhece-se claramente que existem somente dois valores de saída permitidos, e que todos os valores explicados  $\hat{y}_i$  diferem das saídas dos padrões existentes  $y_i$ , ao contrário da árvore de regressão completamente desenvolvida da figura 8.7, onde  $\hat{y}_i = y_i$ . Na figura 8.9, figura 8.10, figura 8.11 mostram-se os resultados para as profundidades máximas de dois, três e quatro, respectivamente. ■

Vamos aplicar uma regressão com entrada  $\mathbf{x}$  bidimensional e saída  $y$  unidimensional. Reaproveita-se o exemplo da regressão linear múltipla do exemplo 2.9 na pág. 56 e da tabela 2.2, composto por dez padrões de entrada bidimensional  $\mathbf{x} = [x_1 \ x_2]^T$  e saída unidimensional  $y$ . Na figura 8.12 mostra-se o resultado da geração de uma árvore de regressão baseada nesse conjunto de dados. Como estratégia de poda, a profundidade máxima da árvore foi fixada em dois. Repare que ambos as características  $x_1$  e  $x_2$  aparecem nos nós. No gráfico, reconhece-se claramente os quatro níveis diferentes que correspondem às quatro folhas da árvore. Consequentemente para qualquer entrada  $\mathbf{x}$  bidimensional, as quatro saídas possíveis se limitam aos valores 14.572, 24.042, 39.56 e 53.69.

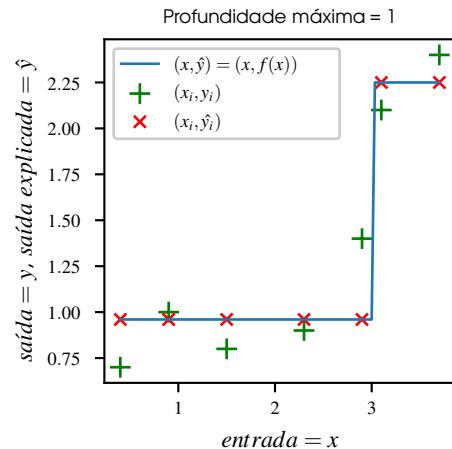
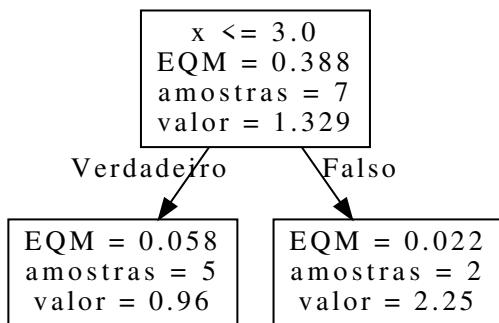


Figura 8.8: Árvore de regressão com profundidade máxima  $p_{\max} = 1$ .

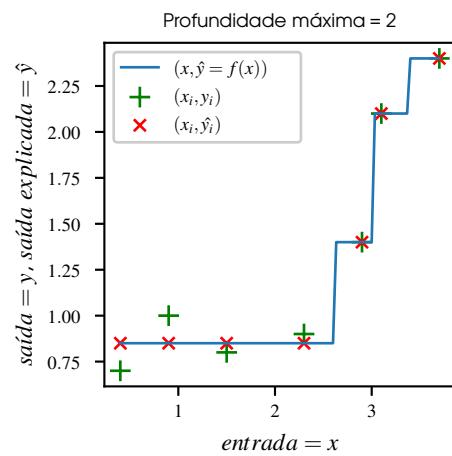
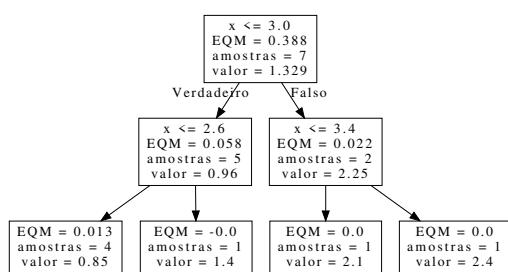
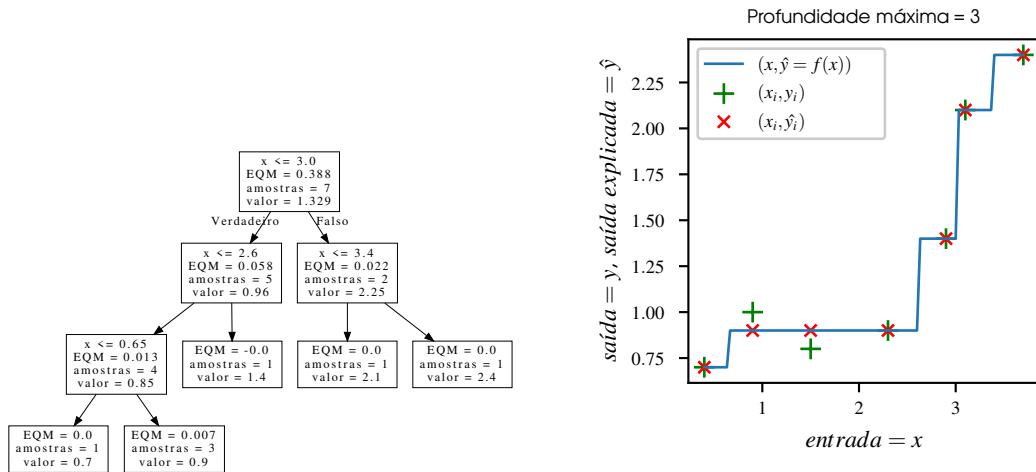
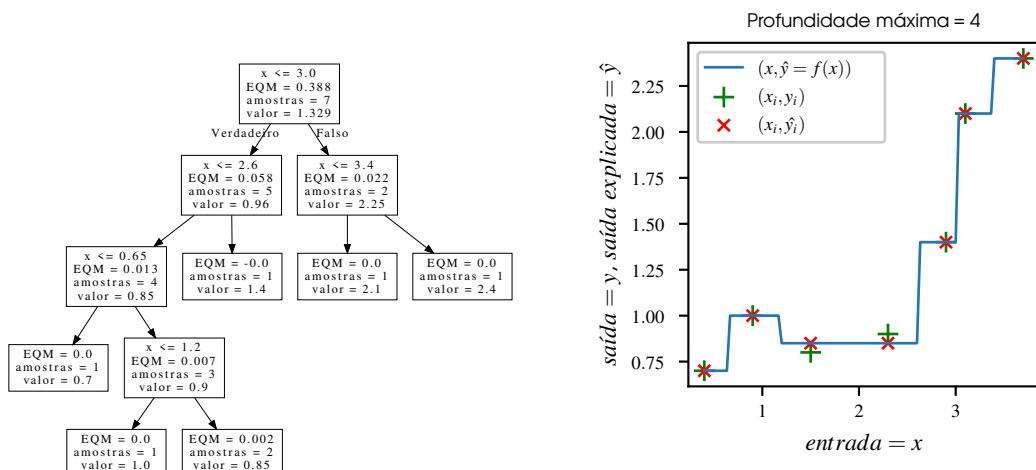


Figura 8.9: Árvore de regressão com profundidade máxima  $p_{\max} = 2$ .

Figura 8.10: Árvore de regressão com profundidade máxima  $p_{\max} = 3$ .Figura 8.11: Árvore de regressão com profundidade máxima  $p_{\max} = 4$ .

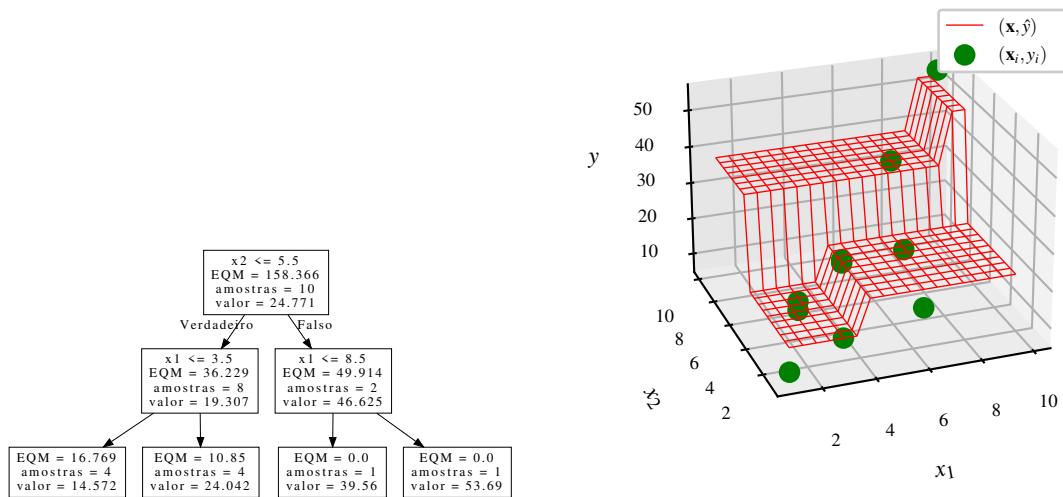


Figura 8.12: Árvore de regressão múltipla de entrada de dimensão dois e saída unidimensional, com profundidade máxima  $p_{\max} = 2$ . Dados originados da tabela 2.2 na pág. 57.

## 8.5 Ensembles de Árvores de Decisão e Regressão, Floresta

Neste ponto do texto vamos introduzir o conceito do *ensemble*. Um ensemble de árvores de decisão ou regressão chama-se *floresta*. Frequentemente, em caso de árvores de decisão, em vez de ensemble, usa-se a denominação de comitê (*committee*), pois a decisão final na classificação é feita por uma votação. Cada uma das árvores é treinada por uma porção do conjunto total de treinamento de  $n$  amostras de uma maneira arbitrária. Essas partes não necessariamente são mutuamente exclusivas. Isto significa que algumas porções do conjunto de treinamento possam se repetir nas árvores individuais. Neste caso o ensemble de árvores é uma *floresta aleatória*, (*random forest*, RF)<sup>9</sup>. A ideia básica do ensemble é dividir o conjunto de treinamento em várias partes. Além da divisão arbitrária dos padrões, em cada árvore, um subconjunto dos  $m$  atributos pode ser usado, aumentando ainda mais a divisão arbitrária da informação disponível. Vale ainda mencionar que árvores não são os únicos métodos de regressão e classificação que possam formar ensembles. Pode-se formar um ensemble de várias máquinas lineares, classificadores K-Vizinhos-Mais-Próximos, ou regressores baseados na *Extreme Learning Machine*. Cada membro desse ensemble fornece uma predição (regressão ou classificação), se apresentado com uma entrada desconhecida  $\mathbf{x}$ . A fusão de todos os resultados foi denominado *bootstrap aggregating*, *bagging* pelo autor, L. Breiman [67]. O objetivo global é melhorar o desempenho do modelo completo. Os critérios de desempenho melhoraram e, eventualmente a sua variância diminui, comparado com um modelo composto por um único classificador ou regressor.

Voltando a uma vantagem de árvores, a interpretabilidade dos resultados por um ser humano, podemos dizer que a floresta elimina essa propriedade. É muito pouco provável atribuir alguma semântica explicável a uma árvore da floresta que foi gerada por uma parte dos dados de treinamento, e que desvia de outra árvore no ensemble. A motivação principal de empregar o ensemble em vez de um regressor ou classificador único, é a eventual melhoria no desempenho.

<sup>9</sup>O autor do método, Leo Breiman, registrou como marca registrada os termos Random Forests™, RF™, Random Forests™, RandomForest™ e Random Forest™.

### 8.5.1 Divisão de Dados de Treinamento em Conjuntos Aleatórios

Primeiro vamos estudar como as amostras em cada uma das árvores da floresta são aleatoriamente escolhidos. O ideal seria uma fonte de dados inesgotável que possa fornecer um fluxo contínuo de conjuntos de dados, sujeitos à mesma distribuição probabilística. Esses conjuntos poderiam então ser alimentados a cada um dos modelos. Essa situação ideal em princípio é irrealística, o caso normal é um único conjunto de dados, veja o conjunto de flores. Naturalmente podemos pensar em uma divisão aleatória de 50% do total de  $n$  amostras para treinar uma árvore particular. O método padrão para a divisão de amostras é o *bootstrapping*.

**Definição 8.5.1 — Bootstrapping.** Uma escolha aleatória de  $n$  elementos de uma urna, com reposição, estatisticamente, para  $n \rightarrow \infty$ , escolha

$$n' = n \left(1 - \frac{1}{e}\right) = n \cdot 0.63212055\dots \quad (8.46)$$

elementos diferentes. Dessa maneira um conjunto de  $n$  elementos pode ser dividido em duas partes não sobrepostas, com  $n'$  elementos escolhidos uma ou mais vezes, e  $(n - n')$  elementos nunca escolhidos, respectivamente.

■ **Exemplo 8.9** Considerando o exemplo das flores, temos  $n = 150$  amostras com os índices  $1, 2, \dots, 150$ . Os índices das  $n' = 97$  amostras escolhidas<sup>10</sup> pelo método do bootstrap, somente mostrando uma parte, são  $48, 118, 68, 104, \dots, 87, 122, 110$ . Alguns índices foram escolhidos várias vezes, por exemplo, 128 quatro vezes e 132 duas vezes. Dos  $n = 150$  índices, nunca foram escolhidos então  $53 = 150 - 97$  amostras, por exemplo os índices 130, 6, 135. A proporção das amostras escolhidas foi  $n'/n = 97/150 = 0.64\bar{6}$ , o que desvia do limite teórico  $(1 - 1/e)$  em 0.0145461.... ■

Posteriormente, veremos que o *bootstrapping* é um dos métodos para dividir um conjunto de padrões em uma parte para treinar e outra parte para testar um modelo, por exemplo um classificador.

### 8.5.2 Geração da Floresta e Fusão de Saídas das Árvores

Um dos hiperparâmetros de uma RF é a quantidade  $T$  de árvores. Considere a figura 8.13. Cada uma das árvores é treinada por um conjunto de dados, composto por aproximadamente 67% dos  $n$  exemplos do conjunto de treinamento total, usando bootstrap. Após o treinamento de cada uma das árvores, a floresta está treinada, o ensemble está pronto para fazer uma regressão ou classificação. Uma amostra desconhecida  $\mathbf{x}$  é alimentada a cada uma das  $T$  árvores, fornecendo  $T$  respostas  $\mathbf{y}_t$ ,  $t = 1, \dots, T$ , possivelmente divergentes. A resposta pode ser  $T$  valores contínuos, em caso de árvore de regressão, ou  $T$  valores nominais, em caso de árvore de decisão. A fusão dos  $T$  resultados é [67] o *bagging*.

**Definição 8.5.2 — Bagging, Bootstrap Aggregation.** Dado um conjunto de treinamento de  $n$  padrões, em um problema de aprendizagem supervisionada, composto por  $n$  pares de entradas e saídas  $\mathcal{D} = (\mathbf{x}_i, \mathbf{y}_i)$ ,  $i = 1, \dots, n$ , dado a quantidade de  $T$  modelos de regressão ou classificação, dado que cada um dos modelos foi treinado por um subconjunto de dados de treinamento  $\mathcal{D}_t$ ,  $t = 1, \dots, T$ , obtido por bootstrapping da definição 8.5.1, e, dado uma entrada desconhecida  $\mathbf{x}$

<sup>10</sup>Foi usada a chamada `sklearn.utils.resample(numpy.linspace(1, 150, 150), n_samples=150, random_state=0)` da biblioteca de aprendizado de máquina `scikit-learn` para gerar as amostras de bootstrap.

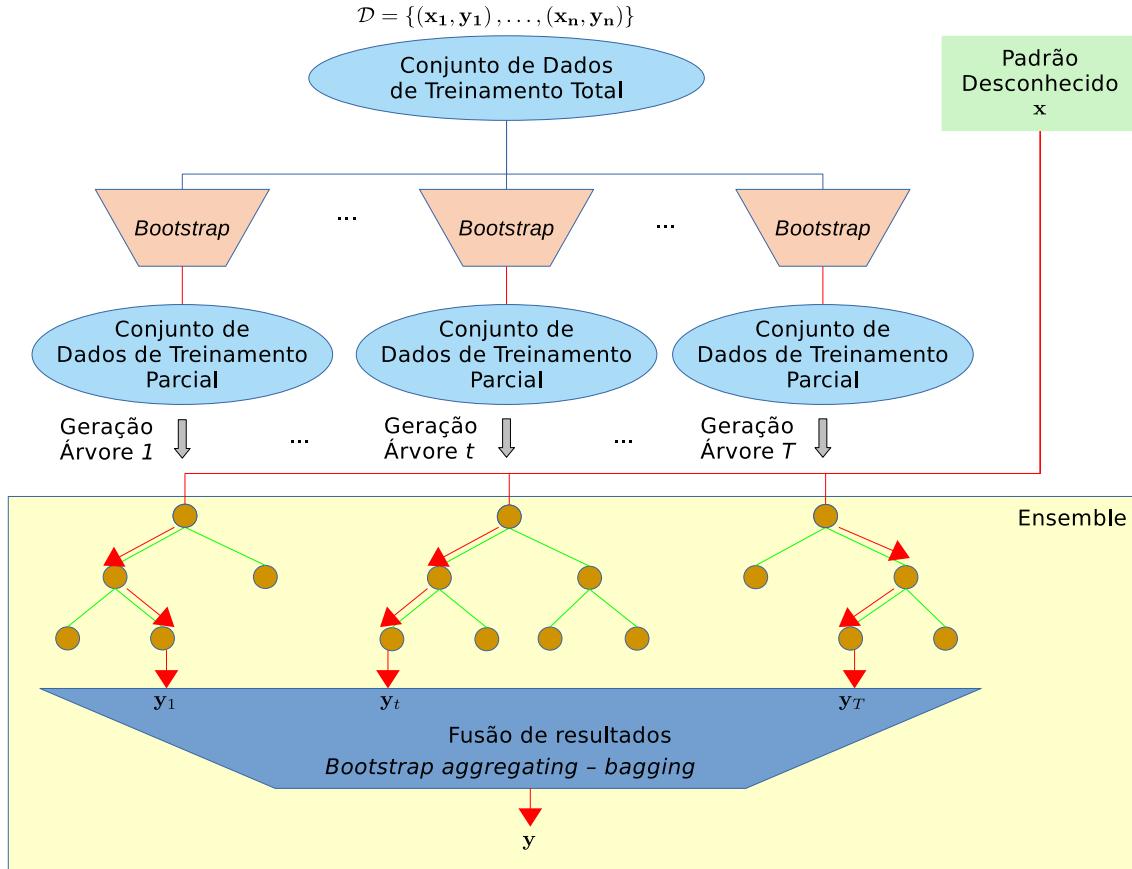


Figura 8.13: Floresta de Árvores de Decisão ou Regressão

e todas as  $T$  saídas  $\mathbf{y}_t$  do ensemble, o resultado do *bagging* é

$$\mathbf{f}(\mathbf{x}) = \begin{cases} \mathcal{C}_p, & p = \arg \max_{k \in \{1, \dots, c\}} \sum_{t=1}^T I(y_t(\mathbf{x}) = \mathcal{C}_k), \\ \bar{\mathbf{y}} = \frac{1}{T} \sum_{t=1}^T \mathbf{y}_t, & \text{se ensemble de classificação} \\ & \text{se ensemble de regressão,} \end{cases} \quad (8.47)$$

onde

$$I(B) = \begin{cases} 1 & \text{se } B \text{ verdadeiro} \\ 0 & \text{caso contrário} \end{cases} \quad (8.48)$$

é a função indicadora, se o  $t$ -ésimo modelo deu como resposta a  $k$ -ésima classe  $\mathcal{C}_k$ . A agregação de classificação representa uma votação majoritária. A regra da (eq. 8.47) em princípio somente faz uma contagem qual classe foi classificada com a maior frequência. A  $t$ -ésima saída  $y_t$  retorna o rótulo de uma classe. A agregação de regressão calcula a média das  $T$  saídas. Essas saídas podem ser multidimensionais, mas como o cálculo da média é independente para cada componente, a média  $\bar{\mathbf{y}}$  tem a mesma dimensão que as saídas individuais  $\mathbf{y}_t \in \mathbb{R}^m$ , ou seja  $\bar{\mathbf{y}} \in \mathbb{R}^m$ .

Em [43], considerações teóricas avançadas sobre o bagging de árvores são elaboradas.

■ **Exemplo 8.10** A figura 8.14 mostra o ensemble de três árvores de decisão geradas, com pro-

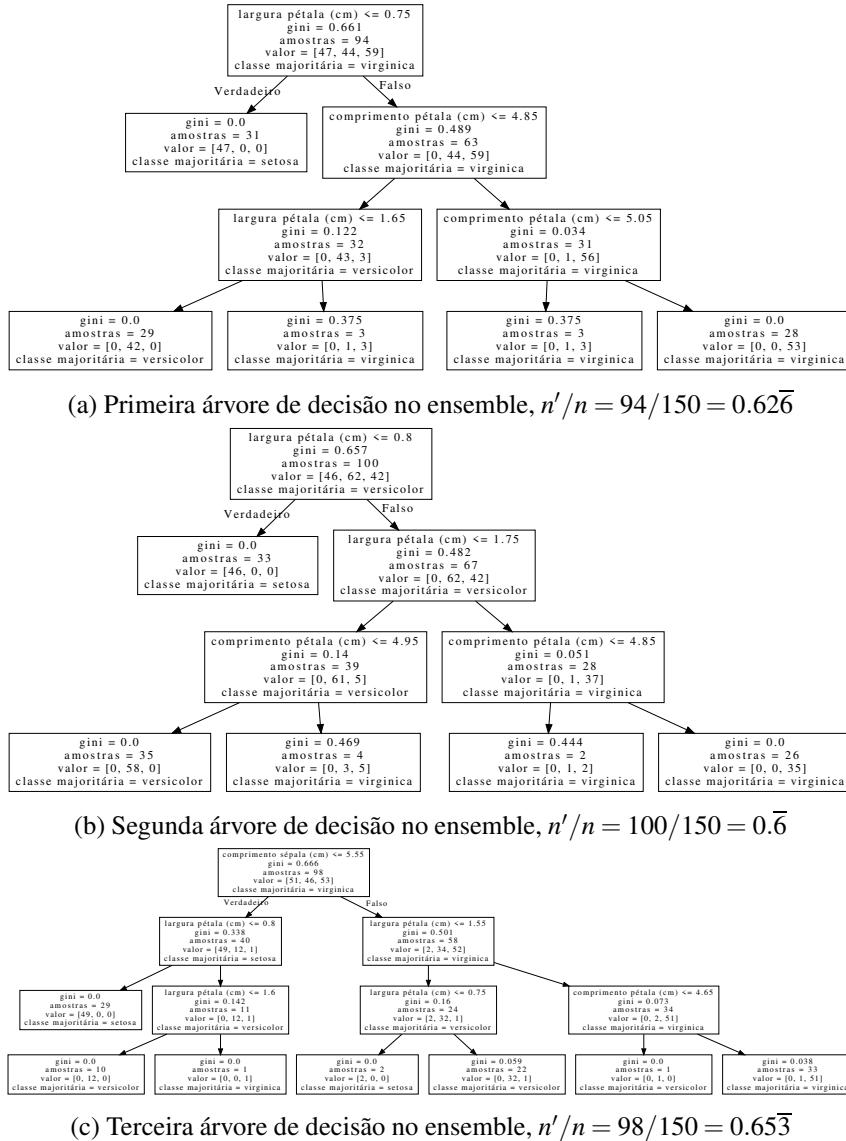


Figura 8.14: Três árvores de Decisão da Floresta, usando dados de iris

fundidade máxima de três,<sup>11</sup> para todos  $n = 150$  padrões de iris. Em seguida a flor  $\mathbf{x}_{134} = [6.3 \ 2.8 \ 5.1 \ 1.5]^T$  que pertence à classe virginica é submetida à floresta de árvores de decisão como amostra de teste. Recorde que a sequência de características é comprimento da sépala, largura da sépala, comprimento da pétala, e largura da pétala. Na primeira árvore, como a largura da sépala com o valor 2.8 é maior que o limiar 0.75, o fluxo de decisão caminha para o filho da direita, depois novamente para direita, pois  $5.1 > 4.85$ , e finalmente para direita, pois  $5.1 > 5.05$ , que uma folha com a única classe virginica. Em analogia a decisão da segunda árvore resulta em virginica de novo, enquanto que a terceira árvore decide versicolor. Pela regra do *bagging* da definição 8.5.2 para classificação, o voto de maioria decide  $C_3$ , pois  $3 = \arg \max_{k \in \{1,2,3\}} [I(y_1(\mathbf{x}_{134}) = C_3) + I(y_2(\mathbf{x}_{134}) = C_3) + I(y_3(\mathbf{x}_{134}) = C_3)]$ . ■

<sup>11</sup> Invocação de `sklearn.ensemble.RandomForestClassifier(n_estimators=3, max_depth=3, random_state=0)` da biblioteca de aprendizado de máquina scikit-learn para induzir a floresta de árvores de decisão.

■ **Exemplo 8.11** A figura 8.15 mostra o ensemble de duas árvores de regressão, usando o conjunto de dados “Linnerud” [62, 107], apresentados explicitamente na tabela 2.3 na pág. 58, no contexto da regressão linear múltipla. Duas árvores de regressão compõem o ensemble. A profundidade máxima foi definida em dois.<sup>12</sup> Como exemplo de predição, considere o último padrão com a entrada  $\mathbf{x} = [\text{Queixo, Abdominais, Saltos}]^T = [2 \ 110 \ 43]^T$  e saída esperada  $\mathbf{y} = [\text{Peso, Cintura, Batimento cardíaco}]^T = [138 \ 33 \ 68]^T$ . Na primeira árvore de regressão, como temos  $\text{Abdominais} \leq 212.5$ , o filho esquerdo é escolhido. Depois, novamente para esquerda, pois  $\text{Saltos} \leq 50.0$ . O resultado é a folha com  $\mathbf{y}_1(\mathbf{x}) = [172.6 \ 35.0 \ 60.0]$  que representa a média esperada de três valores. Essa folha não é composta por uma saída única, pois a árvore foi podada com a profundidade máxima de dois. Em analogia a segunda árvore fornece a saída  $\mathbf{y}_2(\mathbf{x}) = [179.25 \ 35.0 \ 56.3]$ . Então o resultado final, pela regra do *bagging* da definição 8.5.2 para regressão, a média das duas árvores de regressão é  $\bar{\mathbf{y}} = (\mathbf{y}_1(\mathbf{x}) + \mathbf{y}_2(\mathbf{x}))/2 = [175.96 \ 35.0 \ 58.16]$ .

---

<sup>12</sup>Invocação de `sklearn.ensemble.RandomForestRegressor(n_estimators=2, max_depth=2, random_state=0)` da biblioteca de aprendizado de máquina scikit-learn para induzir a floresta de árvores de regressão.

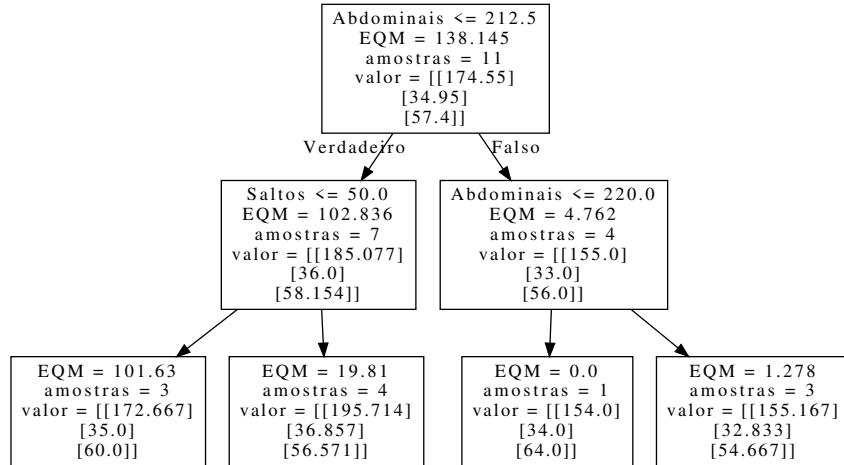
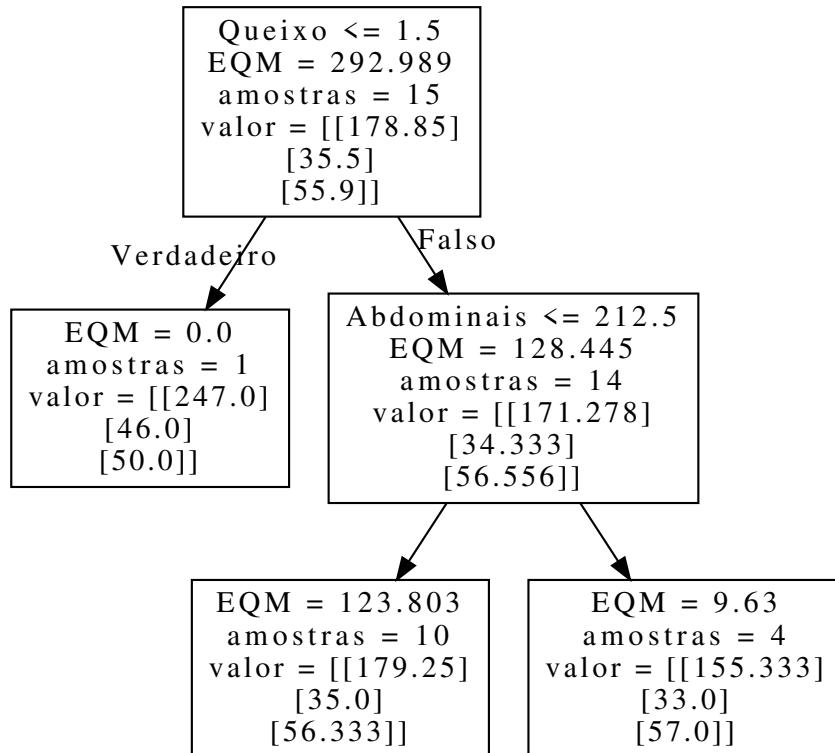
(a) Primeira árvore de regressão no ensemble,  $n'/n = 11/20 = 0.55$ (b) Segunda árvore de regressão no ensemble,  $n'/n = 15/20 = 0.75$ 

Figura 8.15: Duas árvores de Regressão da Floresta, usando dados de “Linnerud”



## 9. Avaliação de Desempenho

### 9.1 Introdução

Já definimos vários modelos de regressão e classificação. O classificador K-Vizinhos-Mais-Próximos ou o regressor feito por uma floresta aleatória são exemplos. Já estudamos como esses modelos são treinados baseados em um conjunto de treinamento. Uma questão que urgentemente tem que ser respondida é sobre a qualidade deste modelo. Será que classifica bem? Como podemos medir o desempenho de uma Máquina Linear em relação aos dados que temos a nossa disposição? As flores que nos servem como um guia nesse caminho dos métodos são um conjunto de 150 setosas, versicolores e virgínicas. Não temos mais que isso. Então todas as pronunciamentos sobre o desempenho de um modelo tem que se limitar a esse conjunto. O ideal seria uma fonte de exemplos novos inesgotável. Assim a “Lei dos grandes números” estaria a nosso favor. Poderíamos repetir a vontade um teste do modelo com dados sempre renovados, e teríamos uma estimativa de algum parâmetro de desempenho que se aproximaria bem ao valor esperado verdadeiro. Essa situação não se aplica à grande maioria dos casos. Às vezes temos já uma divisão de dados previamente estabelecida, um conjunto de treinamento, um conjunto de teste, e eventualmente mais conjuntos de validação que possam ajudar na concepção de fazer um modelo ótimo.

Temos que definir critérios de qualidade de um modelo, como, por exemplo, a taxa de erro estimada de um classificador. Além disso, temos que definir como vamos chegar a esta estimativa. Uma quantidade imensurável de publicações científicas unicamente apresenta como resultado algum número que supostamente descreve a qualidade do sistema desenvolvido. Vamos descobrir que esses números podem desviar bastante da realidade, se a maneira de estimar o desempenho tem certos vícios. Por exemplo, um número pequeno de padrões causa uma variação muito grande dos resultados de um experimento, se esse for feito repetitivamente. Modelos muito flexíveis ao ruído contido nos dados também podem causar uma impressão falsa, uma espécie de superotimismo.

#### 9.1.1 Experimentos Ingênuos

Vamos simplesmente fazer alguns experimentos de classificação e regressão e ver as consequências em relação à mudança das variáveis que podemos modificar. Temos 150 flores, um único conjunto. Para testar um classificador, temos que ter um conjunto de teste. Ou esse conjunto de teste é

completamente diferente do conjunto de treinamento, ou parcialmente diferente, ou até idêntico. A ideia mais óbvia que surge é dividir um conjunto único em uma parte para treinamento e uma parte complementar para teste. A primeira questão quantos dos 150 exemplos serão treinamento e quantos serão teste. Mais uma vez, pensando em uma solução fácil, vamos dividir em 50% de treinamento e 50% de teste, que são 75 padrões para cada um dos dois conjuntos. A próxima questão é sobre a aleatoriedade dessa divisão. Se a divisão for determinística, teremos sempre o mesmo resultado. Uma divisão determinística poderia ser os padrões ímpares como treinamento e os pares como teste. Outra maneira poderia ser os primeiros 75 padrões como treinamento e os últimos 75 padrões como teste. Neste caso pode surgir um problema, o desequilíbrio dos padrões de cada classe, se, como é de costume os primeiros 50 são setosa. O conjunto de treinamento teria todos as setosas e o de teste nenhuma. Uma divisão que respeita a proporção de cada classe é *estratificada*.

Preliminarmente vamos definir informalmente o critério de qualidade mais óbvio de um classificador, a *acurácia* como

$$\text{Acurácia} \stackrel{\text{def}}{=} \frac{\text{Quantidade de padrões do conjunto de teste corretamente classificados}}{\text{Número total de padrões do conjunto de teste}} \quad (9.1)$$

Para os experimentos de classificação, como é de costume, as flores servirão como conjunto de dados. A separação do conjunto total de 150 em dados de treinamento e dados de teste é feita arbitrariamente em 50% para o conjunto de treinamento e 50% para o conjunto de teste. Não se considere uma separação estratificada, simplesmente uma divisão arbitrária em 75 padrões para cada conjunto.

**Dica:** Um bom experimento científico é reproduzível. Isto quer dizer que qualquer outro pesquisador que queira reproduzir seus resultados publicados tem que ter a totalidade de informações disponíveis. Frequentemente, em publicações de métodos de classificação, o único parâmetro de qualidade apresentado é a acurácia nos testes. Na submissão do trabalho para ser publicado em conferência ou revista de qualidade, um bom revisor tem o direito de rejeitar a publicação, se a descrição dos experimentos for incompleta. Por exemplo, neste experimento de classificação, foi descrita a separação de dados em 50% para treinamento e 50% de uma forma *arbitrária*. Normalmente, um gerador de números aleatórios é usado para fazer esse trabalho, porém uma informação crucial que falta para assegurar a reproduzibilidade do seu experimento é a publicação também da *semente* do gerador. Com a semente do gerador que normalmente usa um algoritmo padrão para gerar a sequência dos valores aleatórios, torna-se possível reproduzir exatamente a mesma sequência.

■ **Exemplo 9.1** Para os três modelos de classificadores, Máquina Linear, 1-Vizinho-Mais-Próximo e árvore de decisão as 150 flores são arbitrariamente divididos em dois grupos<sup>1</sup> igual de treinamento e teste. Cada classificador é treinado com o conjunto de treinamento de 75 padrões. Após o treinamento, ambos os conjuntos (treinamento e teste) são submetidos ao classificador. Espera-se um resultado pior para o conjunto de teste, pois esse não foi usado no treinamento, enquanto submeter o mesmo conjunto de treinamento novamente ao classificador certamente é mais fácil para o classificador. A reciclagem dos dados de treinamento como dados de teste chama-se *resubstituição*. Os resultados são resumidos na tabela 9.1. ■

<sup>1</sup>Semente do  $i$ -ésimo experimento para dividir o conjunto total de 150 flores em 75 de treinamento e 75 de teste foi definido como  $s = 10(i - 1)$ ,  $i = 1, \dots, 10$ . Invocação de `sklearn.model_selection.train_test_split(X, y, test_size=0.5, random_state=10*(i-1))` da biblioteca de aprendizado de máquina `scikit-learn`, onde  $X$  e  $y$  são a matriz de dados e os rótulos das 150 flores, respectivamente.

Tabela 9.1: Resultados de dez repetições de experimentos de classificação com três modelos diferentes. Todos modelos treinados com conjunto de treinamento. Classificação do mesmo conjunto de treinamento e classificação do conjunto complementar de teste.

Experimento	Modelo					
	Máquina Linear		1-Vizinho-Mais-Próximo		Árvore de Decisão	
	Treino	Teste	Treino	Teste	Treino	Teste
1	93.33	77.33	100.00	93.33	100.00	96.00
2	88.00	78.67	100.00	93.33	100.00	90.67
3	85.33	78.67	100.00	96.00	100.00	94.67
4	84.00	86.67	100.00	96.00	100.00	93.33
5	85.33	84.00	100.00	93.33	100.00	90.67
6	85.33	80.00	100.00	92.00	100.00	93.33
7	80.00	80.00	100.00	96.00	100.00	96.00
8	85.33	82.67	100.00	94.67	100.00	94.67
9	81.33	89.33	100.00	96.00	100.00	94.67
10	88.00	82.67	100.00	96.00	100.00	94.67

Já podemos observar vários fatos que ajudam no entendimento dos resultados de desempenho de um modelo.

1. Os valores variam de um modelo para outro;
2. Os valores variam para o mesmo modelo de um experimento para outro;
3. Classificar os mesmos padrões que foram usados para treinar em geral tem resultados superiores, comparado com a classificação de padrões que não foram vistos durante o treinamento, chegando até 100% para o 1-NN e a árvore;

A conclusão deste simples experimento é

1. Nunca se pode usar os mesmos dados para treinar um modelo e posteriormente testar um modelo (Superotimismo);
2. Apresentar um resultado baseado em um único experimento, com divisão de dados simples, é cientificamente inaceitável;
3. Em caso de ausência de divisão explícita de dados em treinamento e teste, temos que ter métodos mais confiáveis para avaliar o desempenho de um sistema (Validação Cruzada);
4. Temos que ter ferramentas de análise de resultados, por exemplo um resultado final como fusão de uma série de experimentos, a comparação estatística de modelos diferentes, ou a visualização da variabilidade dos resultados;
5. A capacidade de generalização de um modelo não é suficiente para um modelo simples, como a Máquina Linear, se os dados obedecem a uma distribuição probabilística mais complexa (subajuste, *underfitting*);
6. A capacidade de generalização de um modelo é demais para um modelo altamente flexível como o 1-NN e a árvore (sobreajuste, *overfitting*);

### 9.1.2 Diagrama de Caixa (Boxplot)

Uma ferramenta de visualização simples e eficiente da variabilidade de um conjunto de valores de uma variável aleatória contínua é o diagrama de caixa. Sinônimos são *boxplot*, *box-and-whisker diagram*, ou diagrama de extremos e quartis. É um método da estatística descritiva, não paramétrico que não supõe qualquer distribuição probabilística dos dados. A entrada do método é uma sequência ordenada, e a saída são os parâmetros associados ao diagrama de caixa, além da apresentação visual da mesma. Primeiramente algumas definições que facilitam o entendimento.

**Definição 9.1.1 — Quartil, Mediana.** Dado um conjunto de  $k$  valores numéricos e sua sequência ordenada em ordem crescente

$$s = \{x_1 \leq x_2 \leq \dots \leq x_{\ell-1} \leq x_\ell \leq x_{\ell+1} \leq \dots \leq x_k\}, \quad (9.2)$$

a *mediana* da sequência é o valor no meio da sequência, se a quantidade  $k$  de valores for ímpar, e a média aritmética dos dois valores no meio da sequência, se a quantidade  $k$  de valores for par, ou seja,

$$Q_2 = \text{mediana}(s) \stackrel{\text{def}}{=} \begin{cases} x_\ell, & \ell = \frac{k+1}{2} \text{ se } k \text{ ímpar} \\ (x_\ell + x_{\ell+1})/2, & \ell = \frac{k}{2} \text{ se } k \text{ par.} \end{cases} \quad (9.3)$$

A mediana é equivalente ao *segundo quartil*  $Q_2$ , o que constitui uma fronteira que separa os primeiros dois quartos dos dados ordenadas  $s$  dos segundos dois quartos dos dados ordenadas  $s$ . Isto significa que dividimos a sequência  $s$  em quarto partes iguais. As quatro partes são separadas por três fronteiras, os quartis  $Q_1, Q_2, Q_3$ . Não há consenso sobre a exata posição das fronteiras  $Q_1$  e  $Q_3$  que dividem os primeiros 25% e primeiros 75% dos dados do resto. Adotamos aqui o posicionamento definido pela função `numpy.percentile` da biblioteca numérica da linguagem de programação Python, usado pelos módulos superiores de visualização, como `matplotlib.pyplot.boxplot` da biblioteca de visualização. No caso padrão, a posição contínua  $q \in \mathbb{R}$  é determinada por uma interpolação linear

$$q = 1 + \frac{p(k-1)}{100\%}, \quad (9.4)$$

onde  $k$  é a quantidade dos elementos da sequência

$$s = \{x_1 \leq \dots \leq x_{j-1} \leq x_j \leq x_{j+1} \leq \dots \leq x_k\},$$

e

$$p \in [0\%, 100\%]$$

é o *percentil*. Se uma posição inteira for exigida, tem que obtida por arredondamento. Esse pós-processamento não é aplicado nos cálculos aqui apresentados. Em geral, essa posição  $q$  cai no intervalo de dois elementos consecutivos  $x_j \leq x_{j+1}$  na sequência  $s$ . O valor do quartil, tendo a sua posição, finalmente é novamente obtido, por padrão, por interpolação linear

$$Q = x_j + \alpha(x_{j+1} - x_j), \quad (9.5)$$

onde

$$j = \lfloor q \rfloor$$

é o piso da posição, e

$$\alpha = q - j = q - \lfloor q \rfloor$$

é a fração do intervalo entre as duas posições consecutivas,  $\alpha \in [0, 1]$ .

Repare que a posição  $q$  para a mediana  $p = 50\%$  na (eq. 9.3), pela (eq. 9.4) instancia exatamente para  $(k+1)/2$ , que para  $k$  par é a posição no meio dos elementos centrais com os índices  $j = \ell = k/2$  e  $j+1 = \ell+1 = k/2+1$ , e, para  $k$  ímpar é a posição  $j = \ell = (k+1)/2$  inteira exatamente no meio da sequência.

- **Exemplo 9.2** Nas dez taxas de erro da tabela 9.1, para a Máquina Linear, e a classificação dos próprios dados de treinamento, temos os dez valores que dão origem a sequência ordenada

$$\begin{aligned} s &= \{x_1 \leq x_2 \leq x_3 \leq x_4 \leq x_5 \leq x_6 \leq x_7 \leq x_8 \leq x_9 \leq x_{10}\} \\ &= \{80 \leq 81\frac{1}{3} \leq 84 \leq 85\frac{1}{3} \leq 85\frac{1}{3} \leq 85\frac{1}{3} \leq 85\frac{1}{3} \leq 88 \leq 88 \leq 93\frac{1}{3}\}. \end{aligned}$$

Como a quantidade de valores é par, temos para a posição da mediana

$$q = 1 + 50\%(10 - 1)/100\% = 5.5.$$

Assim os índices são  $\ell = 10/2 = 5$  e  $\ell + 1 = 10/2 + 1 = 6$ . A mediana então é

$$Q_2 = (x_5 + x_6)/2 = (85.33 + 85.33)/2 = 85.33.$$

A posição do primeiro quartil  $Q_1$  que é equivalente a um percentil de 25%, usando a (eq. 9.4), é

$$q = 1 + 25/100(10 - 1) = 3\frac{1}{4}.$$

O primeiro índice em consideração na sequência é

$$j = \lfloor q \rfloor = \left\lfloor 3\frac{1}{4} \right\rfloor = 3.$$

Então a interpolação com  $\alpha = 3\frac{1}{4} - 3 = \frac{1}{4}$  entre os dois elementos  $x_3 = 84.00$  e  $x_{3+1} = x_4 = 85\frac{1}{3}$  é

$$Q_1 = x_3 + \alpha(x_4 - x_3) = 84 + \frac{1}{4}(85\frac{1}{3} - 84) = 84\frac{1}{3}.$$

Em analogia, temos para o terceiro quartil

$$Q_3 = 87\frac{1}{3}.$$

■

**Definição 9.1.2 — Diagrama de Caixa (Boxplot).** Dados os três quartis  $Q_1, Q_2, Q_3$  de uma sequência ordenada  $s = \{x_1 \leq \dots \leq x_k\}$  de  $k$  valores, o *intervalo interquartil* é definido como diferença entre o terceiro e primeiro quartil como

$$\text{IIQ} \stackrel{\text{def}}{=} Q_3 - Q_1. \quad (9.6)$$

Dada uma constante  $c$  que por padrão tem o valor  $c = 1.5$ , o limite inferior e limite superior são definidos como

$$\text{LI} \stackrel{\text{def}}{=} Q_1 - c \cdot \text{IIQ} = (c+1)Q_1 - cQ_3 \quad (9.7)$$

$$\text{LS} \stackrel{\text{def}}{=} Q_3 + c \cdot \text{IIQ} = (c+1)Q_3 - cQ_1. \quad (9.8)$$

Valores  $x_j \in s$  da sequência são considerados dados discrepantes (*outliers*), se forem menor que o limite inferior ou maior que o limite superior, ou seja

$$x_j \in s \quad \text{é valor discrepante ou (\textit{outlier}) se} \quad x_j < \text{LI} \quad \vee \quad x_j > \text{LS}. \quad (9.9)$$

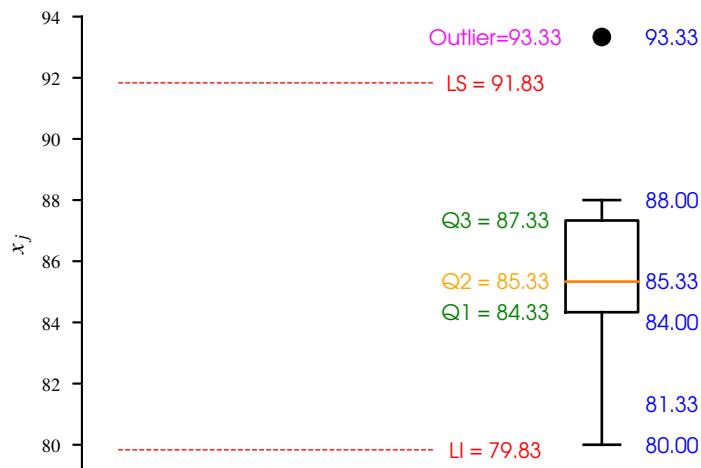


Figura 9.1: Estrutura de um diagrama de caixa (*boxplot*). Dados da primeira coluna (dados de treinamento) da Máquina Linear da tabela 9.1, veja também o exemplo 9.2. Os valores da coluna direita são todos os valores distintos da sequência  $s$  de valores.

O diagrama de caixa, ou (*boxplot*) é uma representação gráfica não paramétrica da variabilidade de um conjunto de dados, em forma horizontal ou vertical. Na versão horizontal temos

1. O intervalo interquartil IIQ como retângulo;
2. A mediana como reta vertical dentro do retângulo do IIQ;
3. O fio de bigode ou *whisker* inferior como reta horizontal entre o valor  $x_j \geq LI$  e o primeiro quartil  $Q_1$ , limitado por uma pequena reta vertical no início; o ponto  $x_j$  é o ponto existente da sequência mais próximo e acima do limite inferior (eventualmente coincide com o limite inferior);
4. O fio de bigode superior como reta horizontal entre o terceiro quartil  $Q_3$  e o valor  $x_j \leq LS$ , limitado por uma pequena reta vertical no final; o ponto  $x_j$  é o ponto existente da sequência mais próximo e abaixo do limite superior (eventualmente coincide com o limite superior);
5. Os valores discrepantes (*outliers*) como circunferências na posição do seu valor no eixo horizontal.

■ **Exemplo 9.3** A figura 9.1 mostra um *boxplot* com os dados do exemplo 9.2. Temos para o intervalo interquartil o valor

$$IIQ = Q_3 - Q_1 = 87\frac{1}{3} - 84\frac{1}{3} = 3.$$

Consequentemente para o limite inferior e superior, com  $c = \frac{3}{2}$ , temos

$$LI = Q_1 - c \cdot IIQ = 84\frac{1}{3} - \frac{3}{2} \cdot 3 = 79\frac{5}{6}$$

e

$$LS = Q_3 + c \cdot IIQ = 87\frac{1}{3} + \frac{3}{2} \cdot 3 = 91\frac{5}{6}$$

O ponto  $x_1 = 80.0 > LI = 79\frac{5}{6}$  é o início do fio do bigode inferior que vai até  $Q_1$ . O ponto  $x_9 = 88.0 < LS = 91\frac{5}{6}$  é o final do fio de bigode superior que começa em  $Q_3$ . ■

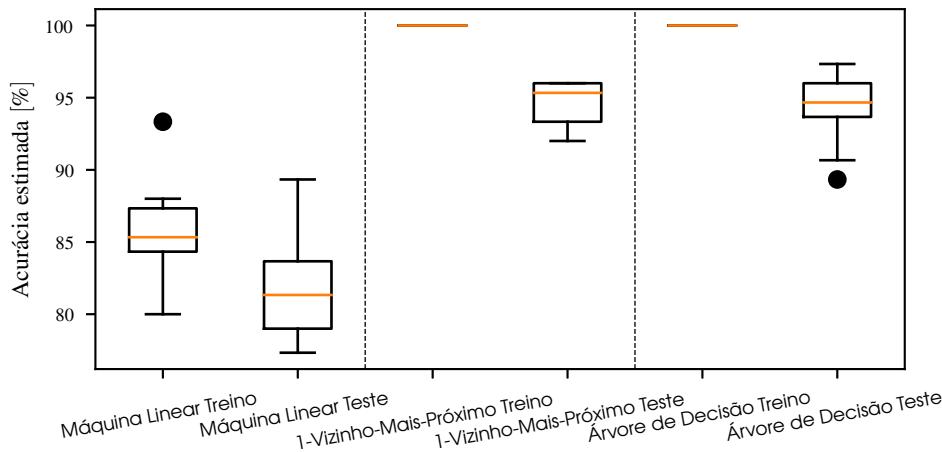


Figura 9.2: Diagramas de caixa (*boxplot*) dos experimentos de classificação para três classificadores diferentes.

A figura 9.2 mostra todos os seis diagramas de caixa dos experimentos simples de classificação resumidos na tabela 9.1. Os gráficos ilustram bem o desempenho de cada situação. Mostra que o resultado de reutilizar os dados de treinamento na classificação é sempre melhor que classificar dados não vistos antes. Mostra também diagramas de caixa degeneradas para o uso de dados de treinamento na classificação para o 1-NN e árvore. Esse resultado sempre 100% de acurácia estimada enfatiza que os últimos dois modelos se ajustam demais aos dados de treinamento. Com o 1-Vizinho-Mais-Próximo é óbvio, pois o vizinho mais próximo de um padrão de treinamento é ele próprio. A única possibilidade de aparecer um erro seria o caso de dois padrões  $\mathbf{x}_i = \mathbf{x}_j$  que tenham rótulos distintos, o que não é o caso no dados iris com quatro características. Igualmente uma árvore de decisão com profundidade máxima, tem nas folhas um único padrão de treinamento. Um padrão de treinamento idêntico acha o caminho automaticamente até essa folha, causando nenhum erro no reuso dos dados de treinamento.

## 9.2 Critérios de Desempenho para Classificadores

A acurácia de um classificador é o critério de qualidade de desempenho mais intuitivo. Um bom classificador comete poucos erros. Vamos estender o conceito da acurácia, pois nem sempre a maior importância é dada ao fato quantos padrões foram classificados corretamente. Às vezes é mais importante destacar que um classificador binário consegue reconhecer o máximo de padrões de uma classe, pagando o preço que a classe complementar seja reconhecida com menos frequência. Em um problema de aprendizagem supervisionada, temos  $n$  padrões de teste  $\mathbf{x}_i$ ,  $i = 1, \dots, n$ , cada um com um rótulo de classe  $\mathcal{C}_k$ ,  $k = 1, \dots, c$  esperado, de um total de  $c$  classes. Em um experimento, a classe explicada (predita, calculada) pode coincidir com a classe esperada, ou não. Podemos construir uma tabela de contingência quadrática que na posição  $(k, k')$  conta a quantidade  $n_{k,k'}$  de vezes que a classe esperada foi  $\mathcal{C}_k$ , e a classe explicada foi  $\mathcal{C}_{k'}$ , com  $k, k' \in \{1, \dots, c\}$ . Vamos chamar esse esquema de dimensão  $c \times c$  uma *tabela de confusão*. No caso especial de  $c = 2$  classes vamos chamar esse esquema de dimensão  $2 \times 2$  uma *matriz de confusão*. Não há consenso sobre essa denominação, frequentemente tabelas com mais que duas classes também são chamadas de matrizes de confusão. A tabela 9.2 ilustra uma tabela de confusão geral de  $c$  classes.

Voltando ao experimento simples de classificação de iris da tabela 9.1, a tabela 9.3 mostra a tabela de confusão do primeiro experimento com o modelo da Máquina Linear, onde a classifica-

Tabela 9.2: Tabela de confusão de experimento de classificação de um classificador com  $c$  classes

		Classe verdadeira						
		$\mathcal{C}_1$	$\dots$	$\mathcal{C}_k$	$\dots$	$\mathcal{C}_{k'}$	$\dots$	$\mathcal{C}_c$
Classe explicada	$\mathcal{C}_1$	$n_{1,1}$	$\dots$	$n_{1,k}$	$\dots$	$n_{1,k'}$	$\dots$	$n_{1,c}$
	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
	$\mathcal{C}_k$	$n_{k,1}$	$\dots$	$n_{k,k}$	$\dots$	$n_{k,k'}$	$\dots$	$n_{k,c}$
	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
	$\mathcal{C}_{k'}$	$n_{k',1}$	$\dots$	$n_{k',k}$	$\dots$	$n_{k',k'}$	$\dots$	$n_{k',c}$
	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
	$\mathcal{C}_c$	$n_{c,1}$	$\dots$	$n_{c,k}$	$\dots$	$n_{c,k'}$	$\dots$	$n_{c,c}$

Tabela 9.3: Tabela de confusão de experimento de classificação, usando o modelo da Máquina Linear, conjunto de iris, divisão treinamento–teste de 50% – 50%, classificando o conjunto de teste.

		Classe verdadeira		
		<i>Setosa</i>	<i>Versicolor</i>	<i>Virginica</i>
Classe explicada	<i>Setosa</i>	21	0	0
	<i>Versicolor</i>	0	18	5
	<i>Virginica</i>	0	12	19

ção dos 75 padrões de teste resultou em uma acurácia de 77.33%. Por exemplo em doze das 75 classificações, versicolor era esperado, mas virginica foi classificada pela Máquina Linear. Neste gráfico, a classe verdadeira é representada nas  $c$  colunas, a classe explicada nas  $c$  linhas. Essa convenção não é universal, por exemplo na biblioteca de aprendizado de máquina `scikit-learn`, a tabela é mostrada em forma transposta, com as linhas como classe explicada e as colunas como classe verdadeira. Considere o caso geral da tabela 9.2 e o caso particular de iris da tabela 9.3. Note que a soma dos valores da  $k$ -ésima coluna é sempre a quantidade  $n_k$  de padrões da classe  $\mathcal{C}_k$ , ou seja

$$n_k = \sum_{i=1}^c n_{i,k}. \quad (9.10)$$

Por outro lado, a soma da  $k$ -ésima linha representa a quantidade de padrões que foram classificados como classe  $\mathcal{C}_k$ . Esta quantidade pode assumir qualquer valor entre zero e a quantidade total  $n$  de padrões, ou seja

$$\left( \sum_{j=1}^c n_{k,j} \right) \in [0, n]. \quad (9.11)$$

No caso de iris temos na soma das colunas,  $n = n_1 + n_2 + n_3 = 21 + 30 + 24 = 75$  padrões de teste, onde cada coluna representa exatamente a quantidade de padrões de teste 21, 30, 24 desta classe. As somas das três colunas são 21, 23, 31, diferentes de  $n_1, n_2, n_3$ .

No momento restringimo-nos a um classificador binário, ou seja, que tem somente padrões com dois rótulos distintos. A matriz de confusão então tem o formato de uma tabela de contingência  $2 \times 2$ . Por conveniência os nomes da duas classes são a classe positiva e a classe negativa. Isso

atribui frequentemente uma semântica útil ao problema de classificação. Por exemplo, para detectar uma falha em um sistema, a detecção da falha claramente seria a classe positiva, um alarme falso seria um falso positivo. Em um contexto de um diagnóstico médico, detectando uma doença, embora não seja um fato positivo, seria a classe positiva.

A definição dos critérios de desempenho de um classificador são calculados a partir de  $n$  padrões de *teste* em um contexto de aprendizagem. Isto significa que temos  $n$  rótulos esperados no conjunto de teste

$$\mathcal{T} = \{y_1, \dots, y_i, \dots, y_n\}. \quad (9.12)$$

Como temos somente duas classes, não precisamos considerar uma saída  $\mathbf{y}_i$  multidimensional. É útil escolher para a classe positiva o valor 1 e para a classe negativa o valor  $-1$ , ou seja  $y_i \in \{-1, 1\}$ . Os resultados de classificação de cada um dos padrões de teste  $\mathbf{x}_i$  são os rótulos explicados pelo modelo  $f$ . São estimativas

$$\hat{y}_i = f(\mathbf{x}_i) \quad (9.13)$$

da verdadeira classe com rótulo  $y_i$ . Em caso de erro temos  $y_i \neq \hat{y}_i$ . Já tocamos neste assunto no contexto do classificador linear binário na seção 2.4 na pág. 65, só que agora o modelo do classificador já não é mais limitado a uma arquitetura linear. A quantidade total de padrões do conjunto de teste é decomposto em na quantidade  $P$  do padrões da classe positiva e na quantidade  $N$  do padrões da classe negativa com

$$n = P + N. \quad (9.14)$$

Dependendo do resultado de classificação de cada um dos padrões podemos categorizar quatro casos diferentes com denominações próprias

$y_i = 1, \hat{y}_i = 1$	Verdadeiro positivo, <i>true positive</i> , TP
$y_i = -1, \hat{y}_i = 1$	Falso positivo, Erro do tipo I, “alarme falso”, <i>false positive</i> , FP
$y_i = 1, \hat{y}_i = -1$	Falso negativo, Erro do tipo II, “perda”, <i>false negative</i> , FN
$y_i = -1, \hat{y}_i = -1$	Verdadeiro negativo, <i>true negative</i> , TN

(9.15)

Na verdade, os acrônimos TP, FN, FP, TN, na maioria dos casos, significam a quantidade dos resultados de classificação de todos os  $n = TP + FN + FP + TN$  padrões do conjunto de teste, que caiem em cada uma das categorias, ou seja,

$$\begin{aligned} TP &= \sum_{i=1}^n I(y_i = 1, \hat{y}_i = 1), \\ FP &= \sum_{i=1}^n I(y_i = -1, \hat{y}_i = 1), \\ FN &= \sum_{i=1}^n I(y_i = 1, \hat{y}_i = -1), \\ TN &= \sum_{i=1}^n I(y_i = -1, \hat{y}_i = -1), \end{aligned} \quad (9.16)$$

Tabela 9.4: Matriz de confusão de experimento de classificação de um classificador binário

		Classe verdadeira	
		Positiva	Negativa
Classe explicada	Positiva	TP	FP
	Negativa	FN	TN

onde  $I(B)$  é a função indicadora, já vista na (eq. 8.48) na pág. 254, e que implementa um contador de predicados verdadeiros. Esses quatro quantidades compõem a matriz de confusão, mostrado<sup>2</sup> na tabela 9.4. Temos a ainda como soma dos verdadeiros e falsos das duas classes

$$\begin{aligned} P &= TP + FN, \\ N &= TN + FP. \end{aligned} \tag{9.17}$$

A partir da matriz de confusão podemos compilar critérios de qualidade de desempenho. A lista na tabela 9.5 não é exaustiva, mas cobre os mais importantes critérios, encontrados na literatura.

Em um problema de classificação multi-classe temos mais que duas classes. Nesse caso, os critérios de desempenho de um problema de classificação binária podem ser estendidos [103]. Uma das  $c$  classes  $\mathcal{C}_k$  é declarada como classe positiva, e os restantes ( $c - 1$ ) classes  $\mathcal{C}_{k' \neq k}$  são mescladas para formarem a classe negativa. No contexto do modelo de mistura de Gaussianas já vimos a fusão das setosas e virgínicas, reveja a figura 7.6 na pág. 205. Podemos ajustar os critérios da classificação binária ao caso de multi-classe, introduzindo um índice. As quatro categorias, verdadeiros positivos, falsos positivos, verdadeiros negativos e falsos negativos, da  $k$ -ésima classe são

$$\begin{aligned} TP_k &= \sum_{i=1}^n I(F(\mathbf{y}(\mathbf{x}_i) = \mathcal{C}_k), \mathbf{x}_i \in \mathcal{C}_k), \\ FP_k &= \sum_{i=1}^n I(F(\mathbf{y}(\mathbf{x}_i) = \mathcal{C}_k), \mathbf{x}_i \notin \mathcal{C}_k), \\ FN_k &= \sum_{i=1}^n I(F(\mathbf{y}(\mathbf{x}_i) \neq \mathcal{C}_k), \mathbf{x}_i \in \mathcal{C}_k), \\ TN_k &= \sum_{i=1}^n I(F(\mathbf{y}(\mathbf{x}_i) \neq \mathcal{C}_k), \mathbf{x}_i \notin \mathcal{C}_k), \end{aligned} \tag{9.18}$$

onde  $F(\mathbf{y}(\mathbf{x}))$  é a função discriminativa, definida na definição 2.3.3 na pág. 62 que retorna uma classe, e  $I$  é novamente a função indicadora. Nesta definição, a variável  $\mathcal{C}_k$  foi usada como o rótulo da  $k$ -ésima classe, e também representando o conjunto dos padrões que pertencem a essa classe. Se  $M$  for uma tabela de confusão definida na tabela 9.2, podemos definir os quatro grupos de cada classe alternativamente. Podemos ainda definir como *suporte* de cada classe  $\mathcal{C}_k$  a quantidade  $n_k$

---

<sup>2</sup>Usa-se aqui a organização da matriz, seguindo [35], onde a classe verdadeira representa uma coluna e a classe explicada pelo modelo representa uma linha. Na literatura, por exemplo, em [16], frequentemente encontra-se uma versão transposta, onde a classe verdadeira representa uma linha.

Tabela 9.5: Critérios de desempenho baseados na matriz de confusão de um problema de classificação binária

Denominação	Definição	Significado
Acurácia ( <i>accuracy</i> )	$ACC = \frac{TP+TN}{P+N} = \frac{TP+TN}{TP+FP+TN+FN}$	Quantidade de classificações corretas em relação a quantidade total de padrões
Sensitividade, Sensibilidade, Revocação ( <i>sensitivity</i> , <i>recall</i> , <i>hit rate</i> , <i>true positive rate</i> , TPR)	$TPR = \frac{TP}{P} = \frac{TP}{TP+FN}$	Proporção dos corretamente classificados como positivos dentro de todos os positivos
Taxa de falsos positivos ( <i>fall-out rate</i> , <i>false positive rate</i> , <i>false alarm rate</i> , FPR)	$FPR = \frac{FP}{N} = \frac{FP}{FP+TN}$	Proporção dos incorretamente classificados como positivos dentro de todos os negativos, “Alarme Falso”
Especificidade ( <i>specificity</i> , <i>selectivity</i> , <i>true negative rate</i> , TNR)	$TNR = \frac{TN}{N} = \frac{TN}{TN+FP}$	Proporção dos corretamente classificados como negativos dentro de todos os negativos
Confiabilidade Positiva, Precisão, Valor Preditivo Positivo ( <i>precision</i> , <i>positive predictive value</i> , PPV)	$PPV = \frac{TP}{TP+FP}$	Proporção dos corretamente classificados dentro de todos classificados como positivos
<i>F-Measure</i>	$F_\beta = \frac{(1+\beta^2)PPV \cdot TPR}{\beta^2 PPV + TPR}$	Ponderação entre Confiabilidade Positiva e Sensitividade, hiperparâmetro $\beta \in [0, 1]$
<i>F1-Measure</i>	$F1 = \frac{2}{\frac{1}{PPV} + \frac{1}{TPR}} = \frac{2 \cdot PPV \cdot TPR}{PPV + TPR}$	Média Harmônica entre Confiabilidade Positiva e Sensitividade, <i>F-Measure</i> com $\beta = 1$

Tabela 9.6: Critérios de desempenho baseados na matriz de confusão de um problema de classificação multi-classe

Denominação	Definição
Acurácia média por classe	$\frac{1}{c} \sum_{k=1}^c \frac{\text{TP}_k + \text{TN}_k}{\text{TP}_k + \text{FP}_k + \text{TN}_k + \text{FN}_k}$
Confiabilidade micro	$\text{PPV}_\mu = \frac{\sum_{k=1}^c \text{TP}_k}{\sum_{k=1}^c (\text{TP}_k + \text{FP}_k)}$
Sensitividade micro	$\text{TPR}_\mu = \frac{\sum_{k=1}^c \text{TP}_k}{\sum_{k=1}^c (\text{TP}_k + \text{FN}_k)}$
F-Measure micro	$\text{F}_\mu = \frac{(1+\beta^2)\text{PPV}_\mu \cdot \text{TPR}_\mu}{\beta^2 \text{PPV}_\mu + \text{TPR}_\mu}$
Confiabilidade macro	$\text{PPV}_M = \frac{1}{c} \sum_{k=1}^c \frac{\text{TP}_k}{\text{TP}_k + \text{FP}_k}$
Sensitividade macro	$\text{TPR}_M = \frac{1}{c} \sum_{k=1}^c \frac{\text{TP}_k}{\text{TP}_k + \text{FN}_k}$
F-Measure macro	$\text{F}_M = \frac{(1+\beta^2)\text{PPV}_M \cdot \text{TPR}_M}{\beta^2 \text{PPV}_M + \text{TPR}_M}$

dos padrões de cada classe, usados no experimento.

$$\begin{aligned}
 \text{TP}_k &= M_{k,k}, \\
 \text{FP}_k &= \sum_{j=1, j \neq k}^c M_{k,j}, \\
 \text{FN}_k &= \sum_{i=1, i \neq k}^c M_{i,k}, \\
 \text{TN}_k &= \sum_{\ell=1, \ell \neq k}^c M_{\ell,\ell}, \\
 \text{suporte}_k &= n_k = \sum_{i=1}^c M_{i,k},
 \end{aligned} \tag{9.19}$$

A acurácia global é

$$\text{Acurácia} = \frac{\sum_{k=1}^c \text{TP}_k}{\sum_{k=1}^c n_k} = \frac{1}{n} \sum_{k=1}^c \text{TP}_k. \tag{9.20}$$

Adicionalmente podemos definir os critérios no nível micro  $\mu$  e no nível macro  $M$ . Os critérios multi-classe estão listados na tabela 9.6.

■ **Exemplo 9.4** Na tabela 9.7, usando os resultados do experimento simples da tabela 9.3, todos os critérios de desempenho para classificação multi-classe da tabela 9.6 são ilustrados. ■

Tabela 9.7: Critérios de desempenho de classificação globais e por classe

Classe	TP	FP	FN	TN	Confiabilidade	Sensitividade	F1-Measure	Acurácia média por classe
Setosa	21	0	0	37	1.0	1.0	1.0	100% 21
Versicolor	18	5	12	40	0.783	0.600	0.679	77.33% 30
Virginica	19	12	5	39	0.613	0.792	0.691	77.33% 24
Acurácia global						77.33%		75
Confiabilidade macro						0.799		75
Sensitividade macro						0.797		75
F1-Measure macro						0.798		75
Confiabilidade micro						0.773		75
Sensitividade micro						0.773		75
F1-Measure micro						0.773		75

### 9.3 Curvas de Desempenho, ROC e Confiabilidade Positiva–Sensitividade (*Precision–Recall*)

Podemos definir espaços bidimensionais que revelam informações relevantes sobre o desempenho de uma classificador. Por exemplo, o espaço FPR–TPR, Taxa de falsos positivos como eixo  $x$  – Sensitividade como eixo  $y$ , permite, para um classificador binário com um *score* contínuo, junto com um limiar, de traçar uma curva que captura relações interessantes sobre o desempenho do classificador, a curva ROC. Outro espaço<sup>3</sup> que será estudado é o TPR–PPV, Sensitividade como eixo  $x$  – Confiabilidade Positiva como eixo  $y$ . Vamos repetir os experimentos simples que resultaram na tabela 9.1, onde uma Máquina Linear, um 1-Vizinho-Mais-Próximo, e uma árvore de decisão foram comparados. Porém, desta vez vamos omitir a classe setosa para ter um problema de classificação binária somente com as classes versicolor e virginica. Adicionalmente, um classificador “cego” é usado, ou seja, um classificador arbitrário que simplesmente adivinha uma classe na sua decisão. Neste teste, todas as classes são arbitrariamente escolhidas pelo classificador cego, independente da sua probabilidade a priori estimada.

Uma observação geral muito importante neste contexto é sobre o valor absoluto da acurácia de um classificador. Imagine um processo industrial, onde um em mil condições de um processo é uma falha. Um classificador que tem uma acurácia de 99.9% sobre essa condição do processo, não tem utilidade nenhuma, pois não decide melhor que o classificador que se baseia puramente na probabilidade a priori.

#### Um experimento – Um ponto no espaço

A tabela 9.8 mostra o resultado de um único experimento com estes quatro classificadores. Uma simples divisão arbitrária de 50%–50% em um conjunto de treinamento e um conjunto de teste foi aplicada, os classificadores treinados, e os critérios de desempenhos foram registrados. Para este experimento, as coordenadas bidimensionais dos quatro classificadores são mostrados na figura 9.3. A figura 9.3a relaciona os dois critérios FRP como eixo  $x$  e TPR como eixo  $y$ . A figura 9.3b relaciona os dois critérios TPR como eixo  $x$  e PPV como eixo  $y$ .

<sup>3</sup>Em inglês esse espaço é denominado *Precision–Recall*, que corresponde às coordenadas  $(y,x)$ , em vez de  $(x,y)$ .

Tabela 9.8: Critérios de desempenho da tabela 9.5 para um único experimento de quatro classificadores diferentes

	Modelo			
	Máquina Linear	1-Vizinho-Mais-Próximo	Árvore de Decisão	Classificador Arbitrário
Matriz de Confusão	$\begin{bmatrix} 23 & 2 \\ 1 & 24 \end{bmatrix}$	$\begin{bmatrix} 22 & 3 \\ 0 & 25 \end{bmatrix}$	$\begin{bmatrix} 22 & 3 \\ 2 & 23 \end{bmatrix}$	$\begin{bmatrix} 13 & 12 \\ 11 & 14 \end{bmatrix}$
<b>Critério</b>				
TP	23	22	22	13
FP	2	3	3	12
FN	1	0	2	11
TN	24	25	23	14
ACC	0.94	0.94	0.90	0.54
TPR	0.96	1.00	0.92	0.54
FPR	0.08	0.11	0.12	0.46
TNR	0.92	0.89	0.88	0.54
PPV	0.92	0.88	0.88	0.52
F1	0.94	0.94	0.90	0.53

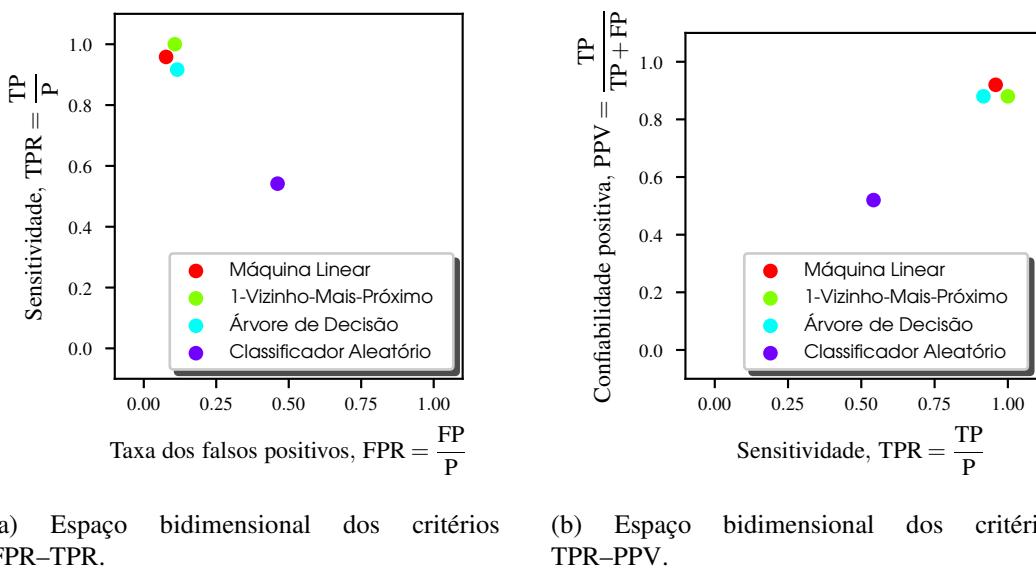


Figura 9.3: Espaço de visualização de experimentos de classificação binária, baseados na matriz de confusão. Cada ponto representa o resultado de um experimento de classificação, usando como coordenada dois dos critérios de desempenho.

Cada experimento de classificação binária resulta em um único ponto nos espaços apresentados. A causa disso é o valor fixo da matriz de confusão.

### Um ponto no espaço FPR–TPR

Algumas coordenadas e regiões no espaço FPR–TPR da figura 9.3a são de interesse.

1. O ponto do canto inferior esquerdo com  $(\text{FPR-TPR}) = (0,0)$  representa um classificador que nunca erra em relação aos padrões da classe positiva, porém também não é capaz de reconhecer um único padrão positivo;
2. Do lado oposto, no canto superior direito em  $(1,1)$  temos um classificador que decide cada padrão como positivo, cometendo simultaneamente o erro de dizer que qualquer padrão seja positivo, mesmo sendo a sua verdadeira classe a negativa;
3. O ponto mais desejável é  $(0,1)$ , pois esse classificador não comete erro nenhum, a matriz de confusão tem forma diagonal, pois não há falsos positivos ou falsos negativos. Nesse sentido, um classificador que gera uma coordenada na área noroeste é melhor do que um classificador que gera uma coordenada mais próximo da diagonal;
4. Um classificador com uma coordenada mais a esquerda pode ser rotulado como mais “conservador”, pois somente com uma evidência forte para a classe positiva, um padrão é classificado como tal, evitando assim muitos falsos positivos;
5. Outro oposto são classificadores “liberais”, que com pouca evidência dizem que o padrão pertence à classe positiva, pagando o preço de gerar mais falsos positivos;
6. Qualquer coordenada com igualdade das duas taxas, ou seja,  $\text{FPR} = \text{TPR}$  representa uma decisão arbitrária.

*Demonstração.* Temos

$$\begin{aligned}\text{TPR} = \text{FPR} &\iff \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{FP}}{\text{FP} + \text{TN}} \\ &\iff \text{TP} \cdot \text{FP} + \text{TP} \cdot \text{TN} = \text{TP} \cdot \text{FP} + \text{FP} \cdot \text{FN} \\ &\iff \text{TP} \cdot \text{TN} = \text{FP} \cdot \text{FN},\end{aligned}$$

ou seja, a relação dos verdadeiros “T” e falsos “F” em uma classificação é balanceada, pois os dois lados da equação são o produto da diagonal da matriz de confusão que contém os acertos, e produto dos elementos fora da diagonal que contém os erros; ■

7. A coordenada do experimento de classificação não depende da probabilidade a priori das classes. Essa propriedade do espaço FPR–TPR é muito importante, pois permite fazer afirmações sobre o desempenho de um classificador, sem ter que considerar a probabilidade a priori. Como já foi mencionado, uma classificação de 99.9% de acurácia é arbitrária, se a probabilidade a priori de uma das duas classes for 0.1%. Vamos mostrar a independência da probabilidade a priori em seguida.

*Demonstração.* Na matriz de confusão tabela 9.4, a soma da primeira coluna, os verdadeiros positivos e falsos negativos são todos os  $\text{TP} + \text{FN} = n_{\text{pos}}$  padrões positivos. Essas duas quantidades são então individualmente proporcionais à probabilidade a priori da classe positiva  $P_{\text{pos}}$ , pois essa probabilidade a priori pode ser estimada como proporção dos padrões positivos em relação à todos os  $n$  padrões como

$$\hat{P}_{\text{pos}} = n_{\text{pos}}/n.$$

Então temos

$$\text{TP} \propto n_{\text{pos}}/n, \quad \text{FN} \propto n_{\text{pos}}/n,$$

ou, equivalente para duas constantes  $a$  e  $b$

$$TP = a \cdot n_{\text{pos}}, \quad FN = b \cdot n_{\text{pos}}.$$

Em analogia, para a segunda coluna  $FP + TN = n_{\text{neg}}$  da matriz de confusão, e constantes  $c$  e  $d$ , temos

$$FP = c \cdot n_{\text{neg}}, \quad TN = d \cdot n_{\text{neg}}.$$

Considerando as definições de TPR e FPR da tabela 9.5, a relação dos verdadeiros positivos e falsos positivos então é

$$\frac{\text{TPR}}{\text{FPR}} = \frac{\text{TP}}{\text{P}} \cdot \frac{\text{N}}{\text{FP}} = \frac{\text{TP}(\text{FP} + \text{TN})}{(\text{TP} + \text{FN})\text{FP}} = \frac{a \cdot n_{\text{pos}} \cdot n_{\text{neg}}(c + d)}{(a + b)n_{\text{pos}} \cdot n_{\text{neg}} \cdot c} = \frac{a(c + d)}{(a + b)c} = \text{const},$$

portanto um intercâmbio entre a classe positiva e negativa que muda as quantidades  $n_{\text{pos}}$  e  $n_{\text{neg}}$ , não afeta a relação dos verdadeiros positivos e falsos positivos. ■

### Um ponto no espaço TPR–PPV

Uma visão alternativa em um espaço bidimensional é a sensitividade (TPR) como eixo  $x$ , e a confiabilidade positiva (PPV) como eixo  $y$ . De uma forma semelhante, podemos fazer algumas afirmações sobre o espaço TPR–PPV da figura 9.3b. Em [96], na análise de um problema de classificação binária, é dada preferência a este espaço em caso de desbalanceamento das duas classes. Desbalanceamento significa que, ou a classe positiva, ou a classe negativa tem uma probabilidade a priori muito maior que a classe complementar. É questionável que o outro espaço FPR–TPR seja menos apropriado para este caso de desbalanceamento, pois já foi comprovado que a coordenada do experimento de classificação não sofre influência da probabilidade a priori, veja a seção 9.3. Há, de fato uma função bijetiva entre os dois espaços, ou seja, um mapeamento não linear, um para um, entre o espaço FPR–TPR e o espaço TPR–PPV. A TPR é o eixo- $y$  do primeiro espaço é o eixo- $x$  do segundo espaço. Então temos que encontrar, para os eixos que são diferentes, um mapeamento bijetivo, então entre os eixos FPR e PPV. Temos que definir uma função e a sua inversa que mapeiam uma matriz de confusão de FPR para PPV e, pelo caminho inverso de PPV para FPR. Em [21] já foi provada a existência de deste mapeamento. Aqui apresentam-se explicitamente<sup>4</sup> a função e sua inversa.

$$\text{PPV}(\text{FPR}) = \text{PPV}(\text{FP}, \text{TN}, \text{TP}) = \left\{ \left[ \left( \left\{ \frac{\text{FP}}{\text{FP} + \text{TN}} \right\}^{-1} - 1 \right) \frac{\text{TP}}{\text{TN}} \right]^{-1} + 1 \right\}^{-1} \quad (9.21\text{a})$$

$$\text{FPR}(\text{PPV}) = \text{FPR}(\text{FP}, \text{TN}, \text{TP}) = \left\{ \left[ \left( \left\{ \frac{\text{TP}}{\text{TP} + \text{FP}} \right\}^{-1} - 1 \right) \frac{\text{TP}}{\text{TN}} \right]^{-1} + 1 \right\}^{-1} \quad (9.21\text{b})$$

Esta equivalência significa informalmente que não podemos ganhar mais informação mudando o espaço, mas podemos tirar conclusões diferentes pela visualização diferente.

No espaço FPR–TPR, um classificador binário “cego”, que toma decisões arbitrárias é associado à diagonal que se estende do canto inferior esquerdo ao canto superior direito, independente da probabilidade a priori das duas classes. Qual é a estrutura geométrica associada a um classificador arbitrário no espaço TPR–PPV? Considerando a matriz de confusão da tabela 9.4, a primeira linha representa todos os padrões que foram classificados como positivos. A confiabilidade positiva (PPV) é a relação do primeiro elemento desta linha com a soma da linha toda, ou seja  $\text{TP}/(\text{TP}+\text{FP})$ .

<sup>4</sup>Para verificar a equivalência, bastam regras simples de álgebra, como  $(a/(a+b))^{-1} - 1 = (a+b)/a - 1 = b/a$ .

Tabela 9.9: Vinte *scores* de padrões e suas classes esperadas de [35]. Classe positiva: p, classe negativa: n.

Nº	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
y	.9	.8	.7	.6	.55	.54	.53	.52	.51	.505	.4	.39	.38	.37	.36	.35	.34	.33	.3	.1
C	p	p	n	p	p	p	n	n	p	n	p	n	p	n	n	p	n	p	n	

Se fizermos uma classificação arbitrária, a proporção dos classificados como positivos será também a proporção dos padrões positivos entre todos os padrões, ou seja, temos um valor constante para PPV que depende somente das probabilidades a priori da classes. Assim temos

$$\text{PPV} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\text{P}}{\text{P} + \text{N}} = \text{const.} \quad (9.22)$$

No espaço TPR–PPV temos então uma reta horizontal com  $\text{PPV} \geq \text{P}/(\text{P}+\text{N})$  que representa para qualquer valor de TPR uma “linha de base” (*baseline*) abaixo de qual nenhum desempenho do classificador pode cair, mesmo sendo um classificador arbitrário.

### Mudando a decisão – Uma curva no espaço

Para classificadores que baseiam as suas decisões em *scores*, podemos gerar mais que um ponto, com a introdução de um limiar. Lembramos aqui a definição 2.3.1 na pág. 61, que para cada uma das  $c$  classes  $\mathcal{C}_k$  fornece um *score* contínuo

$$y_k(\mathbf{x}) \in \mathbb{R}.$$

Com duas classes, podemos definir um *score* único para as duas classes, a classe positiva  $\mathcal{C}_{\text{pos}}$  e a classe negativa  $\mathcal{C}_{\text{neg}}$ , simplesmente pela diferença

$$y(\mathbf{x}) \stackrel{\text{def}}{=} y_{\text{pos}}(\mathbf{x}) - y_{\text{neg}}(\mathbf{x}). \quad (9.23)$$

A origem dos *scores* não é muito importante. Pode ser uma probabilidade, como no classificador baseado na regra de Bayes, ou pode ser a saída de uma Máquina Linear. O score até pode ser forçado para o intervalo  $[0, 1]$  pela função softmax, reveja a (eq. 2.75) na pág. 61. O importante é que cada padrão  $\mathbf{x}_i$  fornece um *score*  $y(\mathbf{x}_i) \in \mathbb{R}$  que representa um valor intermediário contínuo, antes de chegar na fase de atribuição à classe pela função discriminativa.

Considere um experimento com  $n$  padrões de teste  $\mathbf{x}_i$  e os  $n$  *scores*  $y(\mathbf{x}_i)$  da (eq. 9.23), fornecidos pelo classificador binário,  $i = 1, \dots, n$ . Vamos ordenar os  $n$  *scores* em ordem decrescente

$$\{y_1 \geq y_2 \geq \dots \geq y_\ell \geq \dots \geq y_n\}. \quad (9.24)$$

Note que o índice  $\ell$  é diferente do índice  $i$ , pois a sequência dos padrões  $\mathbf{x}_1, \mathbf{x}_2, \dots$  em geral não coincide com a sequência dos *scores*  $y_1, y_2, \dots$ . A tabela 9.9 reproduz o exemplo didático de [35]. São os *scores* ordenados  $y$  de vinte padrões, junto com a classe esperada  $C$  de cada um dos padrões associados ao seu *score*. Cada uma das duas classes tem dez padrões associados. Podemos constatar que o maior *score* é  $y = 0.9$  e o menor *score* é  $y = 0.1$ . Introduz-se agora um limiar  $\tau$  contínuo, com

$$\infty > \tau > -\infty,$$

livremente variável que determina, juntamente com o *score* do padrão, se a classe explicada  $\hat{C}$  é a positiva, ou seja

$$\hat{C} = \begin{cases} \text{positiva,} & \text{se } y \geq \tau \\ \text{negativa,} & \text{se } y < \tau. \end{cases} \quad (9.25)$$

Tabela 9.10: Vinte e um limiares diferentes, mudando os valores da matriz de confusão. Dados de [35], PPV=0 definido para TP=0 e FP=0.

$\tau$	$\infty$	.9	.8	.7	.6	.55	.54	.53	.52	.51	.505	.4	.39	.38	.37	.36	.35	.34	.33	.30	.1
TP	0	1	2	2	3	4	5	5	5	6	6	7	7	8	8	8	9	9	10	10	
FP	0	0	0	1	1	1	1	2	3	3	4	4	5	5	6	7	8	8	9	9	10
FN	10	9	8	8	7	6	5	5	5	4	4	3	3	2	2	2	1	1	0	0	
TN	10	10	10	9	9	9	9	8	7	7	6	6	5	5	4	3	2	2	1	1	0
FPR	0.00	0.00	0.00	0.10	0.10	0.10	0.10	0.20	0.30	0.30	0.40	0.40	0.50	0.50	0.60	0.70	0.80	0.80	0.90	0.90	1.00
TPR	0.00	0.10	0.20	0.20	0.30	0.40	0.50	0.50	0.50	0.60	0.60	0.70	0.70	0.80	0.80	0.80	0.90	0.90	1.00	1.00	
PPV	0.00	1.00	1.00	0.67	0.75	0.80	0.83	0.71	0.62	0.67	0.60	0.64	0.58	0.62	0.57	0.53	0.50	0.53	0.50	0.53	0.50

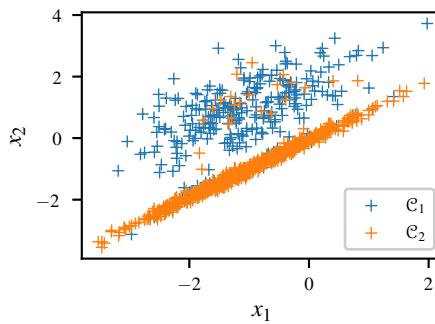


Figura 9.4: Conjunto sintético de duas classes para testes de critério de desempenho..

Um erro é definido em caso de discrepância entre classe esperada  $\mathcal{C}$  e classe explicada  $\hat{\mathcal{C}}$ , ou seja

$$\text{Erro} \iff \hat{\mathcal{C}} \neq \mathcal{C}.$$

A consequência da variação do limiar  $\tau$  é uma matriz de confusão dinâmica. Se o limiar começar a flutuar, as porções de verdadeiros positivos, falsos negativos, falsos positivos e verdadeiros negativos começam a ter um intercâmbio entre os grupos. A tabela 9.10 mostra para cada limiar diferente os valores da matriz de confusão e os critérios de desempenho resultantes. Se o limiar for, por exemplo,  $\tau > 0.9$ , nenhum padrão esperado da classe positiva será explicado como sendo da classe positiva, então a quantidade dos verdadeiros positivos é zero. Todos os dez positivos esperados viram dez falsos negativos. O outro extremo é um limiar  $\tau \leq 0.1$  que classifica corretamente todos os dez padrões da classe positiva. O preço a ser pago é que todos os dez negativos esperados também são classificados como falsos positivos.

Antes de apresentar dois espaços diferentes que são compostos por dois critérios de desempenho, um novo conjunto de dados que será testado nesses espaços é inserido aqui. A figura 9.4 mostra um conjunto sintético de amostras de duas classes que foi gerado<sup>5</sup> arbitrariamente.

### 9.3.1 O espaço FPR–TPR e a curva ROC

Cada experimento de um classificador binário fornece o conjunto  $\{TP, FP, FN, TN\}$  dos quatro valores da matriz de confusão. Derivado desse conjunto, podemos calcular a taxa de falsos positivos  $FPR = FP/N$  e a sensitividade  $TPR = TP/P$ . Uma introdução excelente à análise de desempenho focado neste espaço FPR–TPR foi elaborada por Fawcett [35].

<sup>5</sup>Invocação de `X, y = sklearn.datasets.make_classification(n_samples=1000, n_features=2, n_redundant=0, n_clusters_per_class=1, weights=(0.25, 0.75), flip_y=0.20)` da biblioteca de aprendizado de máquina `scikit-learn`. A probabilidade a priori da classe positiva é 75% (segundo valor no parâmetro “weights”). Adicionalmente pelo parâmetro “flip\_y” alguns padrões mudam de classe, o que resulta em uma probabilidade a priori da classe positiva de 69%, e 31% da classe negativa.

### A curva ROC

Já constatamos que uma matriz de confusão dá origem a uma única coordenada, veja a figura 9.3a. Agora vamos modificar o resultado da classificações pela variação do limiar. Para simplificar, vamos usar um classificador geral que fornece para as duas classes um *score*  $y_{\text{pos}}(\mathbf{x})$  e  $y_{\text{neg}}(\mathbf{x})$ . Porém desta vez, vamos generalizar a regra de classificação  $y(\mathbf{x}) = y_{\text{pos}}(\mathbf{x}) - y_{\text{neg}}(\mathbf{x}) \geq 0$  da (eq. 9.23), introduzindo um limiar  $\tau \in [-\infty, \infty]$ . Assim

$$y(\mathbf{x}) = y_{\text{pos}}(\mathbf{x}) - y_{\text{neg}}(\mathbf{x}) \geq \tau. \quad (9.26)$$

Para  $\tau = 0$  temos o caso convencional da (eq. 9.23). Se esse limiar for, por exemplo  $\tau = -\infty$ , a classe positiva sempre ganha. Não existiriam falsos negativos, nem verdadeiros negativos. A linha da classe negativa explicada na matriz de confusão estaria nula.

Na figura 9.5 podemos observar o resultado da variação do limiar da tabela 9.10 no espaço FPR–TPR. O traço gerado pela variação do limiar é a Curva Característica de Operação do Receptor (Curva COR), ou *Receiver Operating Characteristic*, ROC<sup>6</sup>. Mais precisamente é uma sequência de retas verticais e horizontais que dizem respeito aos valores de TPR e FPR na transição do limiar. Quando mudar a TPR, mas não a FPR, a reta é horizontal, quando a FPR mudar, mas não a TPR, a reta é vertical. Quanto mais padrões tiver, a sequência de retas aproxima-se a uma curva mais suave. Quanto mais essa curva se direciona ao canto superior esquerdo, melhor é o desempenho do classificador binário. O melhor classificador tem uma curva ROC composta por duas retas, primeiro uma reta vertical de  $(0, 0)$  a  $(0, 1)$ , depois uma reta horizontal de  $(0, 1)$  a  $(1, 1)$ . Como uma igualdade de TPR e FPR representa uma decisão arbitrária, a diagonal no espaço ROC, de  $(0, 0)$  ao ponto  $(1, 1)$  representa a pior curva ROC possível.

### A área abaixo da curva ROC, AUC-ROC

Já foi constatado que o pior classificador produz como curva ROC uma diagonal no quadrado unitário  $[0, 1] \times [0, 1]$ . Isso significa que a metade da área do quadrado com o valor de 0.5 representa um limite inferior do desempenho. O outro extremo, a curva ROC que passa pelas coordenadas  $(0, 0)$  a  $(0, 1)$ , e depois a  $(1, 1)$  engloba o quadrado inteiro, portanto uma área com o valor máximo de 1.0.

**Definição 9.3.1 — Área abaixo da curva ROC.** Dado um classificador binário que produz um *score* e um limiar variável que produz a curva ROC, a área abaixo da curva ROC, *Area under the ROC curve*, AUC-ROC  $\in [0, 1]$  é um critério de desempenho do classificador.

$$\text{AUC-ROC} \stackrel{\text{def}}{=} \int_0^1 \text{TPR} \cdot d\text{FPR}. \quad (9.27)$$

A AUC-ROC na figura 9.5 tem um valor de 0.68. Na prática não se usa a definição do cálculo da (eq. 9.27), mas um algoritmo que soma os trapézios formados abaixo da curva, veja o texto didático de [35]. Nessa apresentação também são discutidos a envoltória convexa (*convex hull*) de várias curvas ROC, a interpolação de vários classificadores, retas de desempenho isométricas, e a extensão do espaço FPR–TPR para mais que duas classes. Esses tópicos não são abordados aqui. A figura 9.6 mostra as curvas ROC e a AUC-ROC do conjunto sintético da figura 9.4. Cinco classificadores diferentes até agora estudados que são capazes de produzir um *score* são comparados, um classificador aleatório que decide a classe exclusivamente pela probabilidade a priori, a Máquina Linear da seção 2.2.1 na pág. 53, o classificador Bayesiano de Erro Mínimo, supondo distribuição Gaussiana definido na (eq. 7.21) na pág. 197, o Naïve Bayes da seção 7.3

<sup>6</sup>O acrônimo ROC tem origens históricos. A *Receiver Operating Characteristic* servia para analisar sinais de radar. O objetivo na área militar é identificar aviões hostis, medindo a capacidade de um receptor (*receiver*) de tomar decisões.

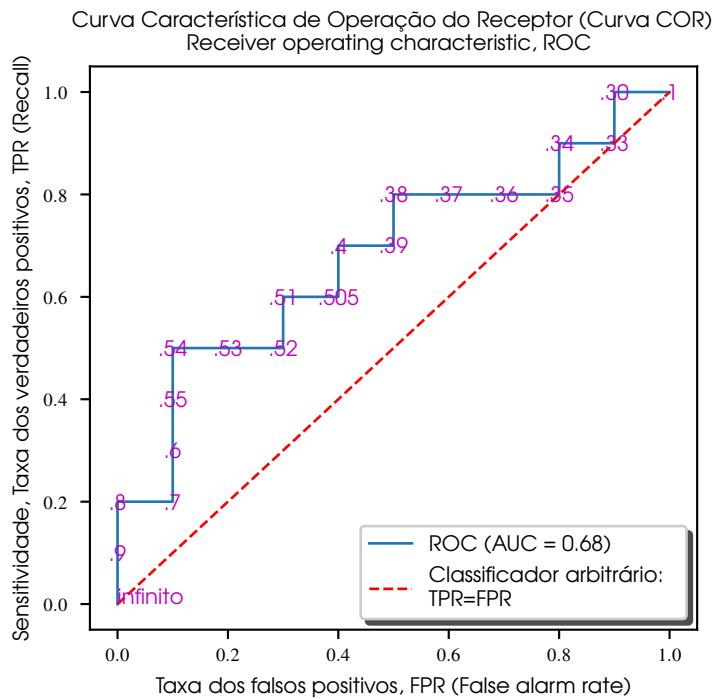


Figura 9.5: A curva ROC gerada pelos dados da tabela 9.10.

na pág. 200, e a Máquina Máquina Extrema de Aprendizado, Extreme Learning Machine, ELM da seção 4.3.1 na pág. 138. Exceto o classificador aleatório, os outros modelos mostram um desempenho em torno de 0.85 para a AUC-ROC. O Naïve Bayes ainda tem um desempenho um pouco pior, enquanto o classificador arbitrário se orienta à diagonal com uma AUC-ROC em torno do esperado 50%.

### 9.3.2 O espaço PPV-TPR Precision–Recall

Na figura 9.7 podemos observar o resultado da variação do limiar da tabela 9.10 no espaço *Precision–Recall*. Embora o conteúdo de informação dos dois espaços seja o mesmo, o aspecto da curva é bastante diferente, lembrando de uma serra. Neste exemplo, a probabilidade a priori das duas classes é idêntica, com o valor de 50%. Assim a linha de base do classificador passa por  $PPV = P/(P+N) = 50\%/(50\% + 50\%) = 0.5$ .

A figura 9.8 usa novamente o conjunto de dados sintéticos da figura 9.4, porém desta vez o desempenho dos cinco classificadores é mostrado no espaço *precision–recall*, PPV–TPR. O comportamento das curvas condiz com os resultados do espaço ROC, três classificadores com bons resultados, o Naïve Bayes com desempenho razoável, e o arbitrário perto do limite, que neste caso é 0.69 dada a relação das duas probabilidades a priori de 69% da primeira classe e 31% da segunda classe.

## 9.4 Critérios de Desempenho para Regressores

Os critérios de desempenho de um classificador são baseados na matriz de confusão. A origem de cada verdadeiro ou falso positivo ou negativo é uma discrepância nítida, “é, ou não é a classe esperada”. No caso de um problema de regressão a saída é contínua, ou um único valor  $y$  de saída, ou um vetor  $y$  de vários valores contínuos. Na (eq. 2.9) na pág. 35 conhecemos a função de perda responsável por medir a discrepância entre um valor ou vetor esperado e um valor ou vetor

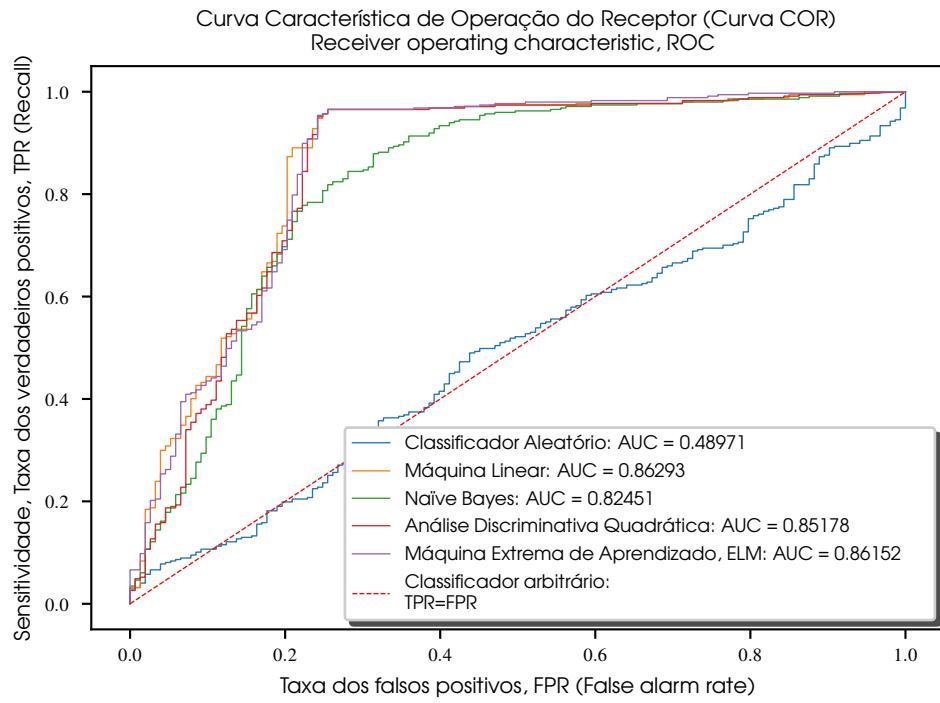


Figura 9.6: As curvas de desempenho de cinco classificadores no espaço FPR–TPR.

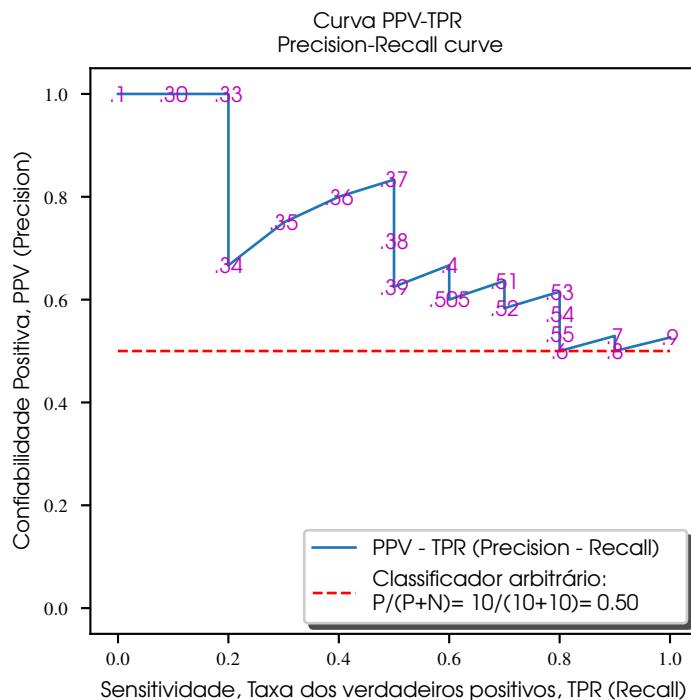


Figura 9.7: A curva no espaço PPV–TPR gerada pelos dados da tabela 9.10.

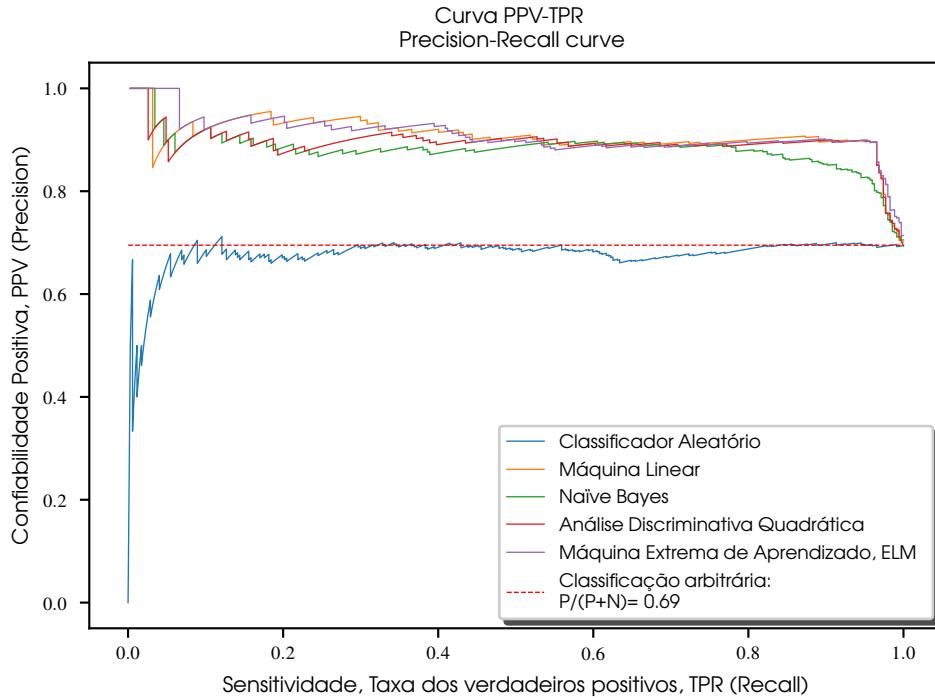


Figura 9.8: As curvas de desempenho de cinco classificadores no espaço TPR–PPV.

explicado de um modelo. Por exemplo no modelo linear simples, a discrepância entre o valor explicado pelo modelo

$$\hat{y} = f(x; \boldsymbol{\theta}) = f(x; w, b) = wx + b,$$

e o valor  $y$  esperado foi medido pelo erro quadrático

$$EQ = (y - \hat{y})^2,$$

e considerando todos os  $n$  padrões, foi definido o Erro Quadrático Médio (EQM), *Mean Squared Error, MSE* a função de perda preferida foi o erro quadrático médio

$$EQM = MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2,$$

também válido para saídas multidimensionais  $\mathbf{y}_i \in \mathbb{R}^m$ , como na figura 2.4 na pág. 55 para a Máquina Linear, ou na figura 4.5 na pág. 141 para uma rede neural multicamada

$$EQM = MSE = \frac{1}{n} \sum_{i=1}^n \|\mathbf{y}_i - \hat{\mathbf{y}}_i\|^2, \stackrel{?}{=} \frac{1}{n} \sum_{i=1}^n \sum_{j=1}^m (y_{ij} - \hat{y}_{ij})^2$$

Na tabela 9.11, alguns critérios de desempenho de um regressor são apresentados. Existe um modelo, que produz uma saída para  $n$  padrões de entrada de um conjunto. Esse conjunto pode ser o conjunto de treinamento, onde se tenta minimizar o critério de desempenho, ou um conjunto de teste que mede o desempenho do modelo após de ter sido treinado. Pode ser também um conjunto de padrões usado para guiar o aprendizado do modelo, por exemplo um conjunto de validação que decide quando parar o treinamento em um modelo que é ajustado iterativamente. A matriz  $Y$  de

Tabela 9.11: Critérios de desempenho de regressão

Acrônimo	Denominação	Definição
EQM	Erro Quadrático Médio ( <i>Mean Squared Error</i> , MSE)	$\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$
EQM	Erro Quadrático Médio, saída multidimensional	$\frac{1}{n} \sum_{i=1}^n \sum_{j=1}^m (y_{ij} - \hat{y}_{ij})^2 \stackrel{[3.11]}{=} \frac{1}{n} \ Y - \hat{Y}\ _F^2$
REQM	Raiz do Erro Quadrático Médio ( <i>Root Mean Squared Error</i> , RMSE)	$\sqrt{\text{EQM}}$ , unidimensional ou multidimensional
EAM	Erro Absoluto Médio ( <i>Mean Absolute Error</i> , MAE)	$\frac{1}{n} \sum_{i=1}^n  y_i - \hat{y}_i $
EAM	Erro Absoluto Médio, saída multidimensional	$\frac{1}{n} \sum_{i=1}^n \ \mathbf{y}_i - \hat{\mathbf{y}}_i\ $
ERQM	Raiz do Erro Relativo Quadrático ( <i>Root Relative Squared Error</i> , RRSE)	$\left( \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2} \right)^{1/2}, \bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$
ERQM	Raiz do Erro Relativo Quadrático, saída multidimensional	$\left( \frac{\sum_{i=1}^n \ \mathbf{y}_i - \hat{\mathbf{y}}_i\ ^2}{\sum_{i=1}^n \ \mathbf{y}_i - \bar{\mathbf{y}}\ ^2} \right)^{1/2}, \bar{\mathbf{y}} = \frac{1}{n} \sum_{i=1}^n \mathbf{y}_i$

dimensão  $n \times m$  tem na sua  $i$ -ésima linha a saída esperada  $\mathbf{y}_i^\top$  transposta do  $i$ -ésima padrão, e  $\hat{Y}$  a saída explicada pelo modelo  $\hat{\mathbf{y}}_i^\top$  transposta.

Uma diferença notável entre um critério de desempenho de classificação, por exemplo, a acurácia, e um critério de regressão, por exemplo a EQM é que a EQM pode ser usada diretamente como uma função de perda (*loss function*)

$$L(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 = \frac{1}{n} \sum_{i=1}^n (y_i - f(x_i; \boldsymbol{\theta}))^2.$$

Dessa maneira, usando o gradiente desse critério de desempenho  $\nabla L(\boldsymbol{\theta})$ , é possível embutir o critério de desempenho diretamente em um algoritmo de otimização iterativo de aprendizado de máquina. Isso foi feito, por exemplo, no algoritmo 5 na pág. 88, descida de gradiente estocástico, onde os parâmetros  $\boldsymbol{\theta}$  foram denominados como  $w$ . Retornaremos aos critérios posteriormente no contexto de algoritmos iterativos.

## 9.5 Estimativa de Desempenho de Modelo

Já tocamos brevemente no assunto da estimativa do desempenho de um classificador, fazendo uma única divisão do conjunto de dados em um conjunto de treinamento e um conjunto de teste. Na seção 9.1.1 dividimos os 150 padrões de flores em 75 para treinamento e 75 para teste. Esta divisão é simplista demais para permitir a obtenção de uma medida confiável, por exemplo, uma acurácia estimada de uma árvore de decisão, ou o erro quadrático médio de uma Máquina Linear na modalidade de regressão. Além disso, a divisão do conjunto de dados pode ser em mais que dois conjuntos. Por exemplo, em um algoritmo de descida de gradiente podemos reservar uma parte dos dados para testar o modelo final. Para estimar corretamente os critérios de desempenho, nunca mais se pode tocar neste conjunto de teste, até a finalização do treinamento do modelo. Os restantes dados são novamente divididos em partes diferentes. Uma porção poderia ser a parte de treinamento para ajustar os parâmetros livres do modelo, por exemplo, os pesos de uma Máquina

Linear, ou a estrutura de uma árvore de regressão. Uma outra porção poderia ser o conjunto de validação que serve para guiar o processo de aprendizagem, ou para escolher os melhores hiperparâmetros.

**Aviso:** Uma alerta preliminar seja dada aqui. Mesmo com o método mais sofisticado de estimativa de desempenho, o valor obtido em geral desvia do valor verdadeiro, ou seja, existe um viés. Somente se tivéssemos um número infinito de padrões, seria possível aproximar com relativa segurança um valor confiável para o critério de desempenho.

Inicialmente, vamos limitar o estudo dos métodos que dividem o conjunto de dados em somente duas partes, o conjunto de treinamento, e o conjunto de teste. Depois vamos definir métodos mais sofisticados que possam ser usados para ajustar o modelo, e ao mesmo tempo, fazer afirmações sobre o desempenho. A figura 9.9 ilustra um procedimento muito simples para obter o valor de desempenho de um modelo de aprendizado de máquina. Um conjunto de  $n$  padrões é usado em um problema de aprendizagem supervisionada. Então cada padrão de entrada  $x_i$  é acompanhado de uma saída  $y_i$  esperada,  $i = 1, \dots, n$ . O conjunto total de  $n$  padrões é dividido em duas partes, o conjunto de treinamento e o conjunto de teste. A separação tem a intenção de ocultar a informação contida no conjunto de teste durante o treinamento do modelo. O conjunto de treinamento é alimentado ao algoritmo de aprendizado de máquina que ajusta os parâmetros livres do modelo. A natureza da aprendizagem pode ser determinística, como no caso de uma árvore de decisão ou regressão, ou pode ser iterativo, como no caso de uma rede neural artificial multicamada, treinada por descida de gradiente. Após o treinamento, o modelo pode ser alimentado pelo conjunto de teste, permitindo a obtenção de uma estimativa de desempenho. No exemplo 9.1 fizemos a divisão em 50% dados para treinamento e 50% para teste. O problema foi classificar as espécies de iris, e comparamos três modelos diferentes, Máquina Linear, 1-Vizinho-Mais-Próximo e árvore de decisão. Além do conjunto de teste, até o conjunto de treino foi realimentado no modelo. Os resultados da acurácia estimada foram compilados na tabela 9.1. Podemos dizer que um algoritmo de aprendizado de máquina tem o objetivo de *generalizar*, ou seja, aprender bem a função de mapeamento das entradas para as saídas do modelo.

### 9.5.1 Técnicas de Divisão de Dados na Validação

Em seguida, as técnicas mais importantes para a divisão do conjunto de dados em treinamento e teste e a obtenção da estimativa de desempenho são apresentadas. Primeiro, apresenta-se uma definição importante na divisão de dados em um problema de classificação.

**Definição 9.5.1 — Divisão de Conjunto de Dados Estratificada.** Dada uma partição de  $n$  amostras, pertencendo a  $c$  classes, em partes possivelmente de tamanhos diferentes, a divisão é *estratificada*, se a proporção de cada uma das  $c$  classes em relação ao tamanho de cada conjunto seja aproximadamente igual.

■ **Exemplo 9.5** Vamos imaginar que as  $n = 150$  flores de  $c = 3$  classes sejam divididas em dois conjuntos, o primeiro com 100 amostras e o segundo com 50 amostras. Então o primeiro conjunto com 33, 33, e 34 amostras de cada uma das classes, e o segundo conjunto com 16, 17 e 17 amostras de cada uma das classes, é uma divisão estratificada. ■

#### Resubstituição

Inapropriado para obter números realísticos, mas valendo a pena incluir aqui é a *resubstituição*. Neste procedimento, não existe a divisão do conjunto total de  $n$  amostras. O mesmo conjunto de  $n$  amostras é usado para treinar o modelo. Depois o mesmo conjunto de  $n$  amostras é submetido ao modelo treinado, veja a figura 9.10. Reveja novamente a tabela 9.1 na pág. 261, onde três classificadores são comparados. O conjunto de treinamento é usado para treinar o modelo, mas também

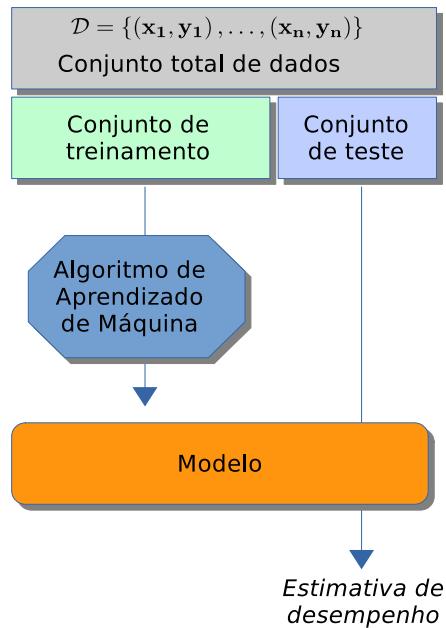


Figura 9.9: Estrutura básica de modelo de aprendizado de máquina.

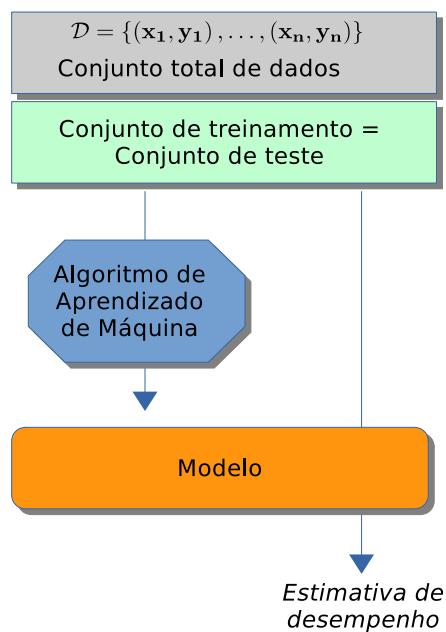


Figura 9.10: Resubstituição de dados de treinamento como dados de teste.

realimentado novamente no modelo, dizendo respeito às colunas “Treino”. A Máquina Linear, mesmo já conhecendo toda a informação do conjunto de treinamento, não consegue evitar erros de classificação. Por outro lado, o 1-Vizinho-Mais-Próximo e a árvore de decisão não cometem erros na resubstituição. Essas propriedades chamamos de subajuste (*underfitting*) e sobreajuste (*overfitting*) nos comentários do exemplo 9.1 na pág. 260. O 1-NN e a árvore têm a capacidade de se ajustar perfeitamente aos dados de treinamento. Mas isso não significa que para novos dados, não vistos antes pelo modelo, o ajuste é apropriado, ou seja, o modelo não consegue *generalizar* bem para a verdadeira distribuição dos dados. Os assuntos de subajuste e sobreajuste serão elaborados posteriormente. De qualquer maneira, usar a resubstituição para estimar o desempenho de um modelo fornece cifras muito longe dos valores verdadeiros, otimistas demais. A resubstituição não pode servir como base para estimar o desempenho de um modelo de regressão ou classificação.

### Divisão Única, *Holdout*

A já apresentada figura 9.9 mostra exatamente o esquema de um divisão dos  $n$  padrões em  $\ell$  padrões de treinamento de  $(n - \ell)$  padrões de teste. Geralmente a porção dos padrões de teste é especificada em valores absolutos, neste caso  $(n - \ell)$ , ou em porcentagem  $(n - \ell)/n\%$ , com um valor comum de 30%. Vamos denominar esta divisão, feita uma única vez como *holdout*. O perigo desta maneira de estimar o desempenho por uma única divisão é ilustrado na figura 9.11. Imagine uma urna com  $N = 20$  bolas,  $K = 10$  pretas e  $N - K = 10$  vermelhas. As pretas representam a classe positiva, as vermelhas a classe negativa. Vamos escolher  $n \leq N$  bolas da urna, *sem reposição*. A função massa de probabilidade<sup>7</sup> que  $k$  dessas  $n$  bolas escolhidas sejam bolas pretas segue uma distribuição hipergeométrica<sup>8</sup>

$$P(k = \text{preta}) = \frac{\binom{K}{k} \binom{N-K}{n-k}}{\binom{N}{n}}, \quad (9.28)$$

com o valor esperado

$$\mathbb{E}[P(k = \text{preta})] = n \frac{K}{N}. \quad (9.29)$$

Vamos definir um problema de classificação binária de bolas, onde as duas classes tenham a mesma probabilidade a priori. A classe da bola seja simplesmente a sua cor. O classificador seja definido pela sua função discriminativa. Ele decide pela cor da bola, dado uma bola do conjunto de teste. Se for preta, é classificada como preta (= positiva), se for vermelha, é classificada como vermelha (= negativa). Isto implica que o classificador maximiza a acurácia verdadeira de 50%, se tivéssemos um número infinito de amostras. Se o nosso conjunto de teste tiver 10 bolas, a acurácia então é simplesmente estimada pela proporção das bolas positivas. Por exemplo, se o conjunto de teste com 10 bolas tiver oito positivas e duas negativas, a acurácia estimada é de 80%. Se repetíssemos a seleção dos 10 padrões de teste um número infinito de vezes, o valor esperado das amostras da classe positiva no conjunto de teste, pela (eq. 9.29) seria

$$\mathbb{E}[P(k = \text{positiva})] = 10 \frac{10}{20} = 5.$$

<sup>7</sup>Sendo uma variável aleatória discreta, a função massa de probabilidade expressa diretamente um valor de uma probabilidade  $0 \leq P \leq 1$ .

<sup>8</sup>Na definição da distribuição hipergeométrica, foram usados os nomes usuais na literatura, sendo, por exemplo  $N$  o número total de amostras e  $n$  os do “sucesso”. Por outro lado, a variável  $n$  no nosso contexto costuma denominar a cardinalidade do conjunto de dados em consideração.

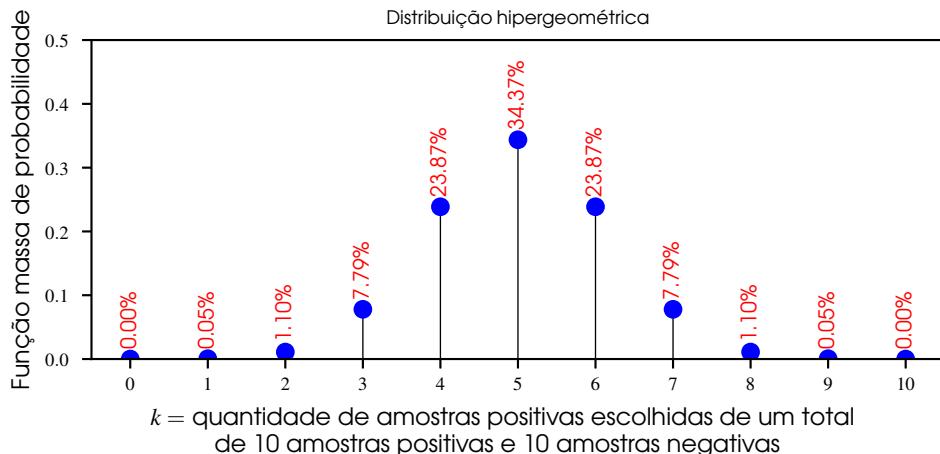


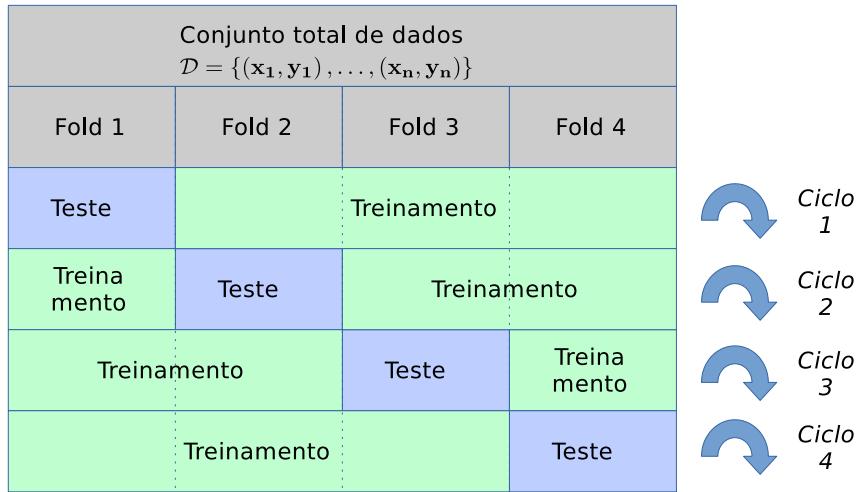
Figura 9.11: Probabilidade da quantidade de padrões de uma classe dentro do de conjunto de treinamento

Isto implica que o valor esperado da acurácia é de 50%, pois na média temos cinco bolas positivas no conjunto de teste de 10 bolas.

Lembre-se que o *holdout* faz uma *única* divisão do conjunto total em treinamento e teste. A única possibilidade que a verdadeira acurácia de 50% coincide com a acurácia estimada, em um *único* experimento com 10 bolas no conjunto de teste, acontece, se exatamente cinco bolas do conjunto de teste são bolas positivas, e isso, pela figura 9.11, acontece somente em 34.37% das vezes. Isso significa equivalentemente que com uma probabilidade de 65.65%, a nossa estimativa da acurácia pelo *holdout* tem um viés. E este viés pode ser grande, por exemplo, 30% ou 70% em vez dos verdadeiros 50% com uma probabilidade de 7.79%, visível nos valores  $k = 3$ , respectivamente  $k = 7$  na figura 9.11. Concluindo, podemos afirmar que o *holdout* é primitivo demais para obter uma estimativa confiável. Isto somente não acontece, se o conjunto de teste representa bem a distribuição dos dados, e se muitos padrões estiverem disponíveis para treinar e testar um modelo. Especialmente em um estudo comparativo de vários classificadores, usar o *holdout* para obter as estimativas seria muito pouco confiável para fazer afirmações que um modelo seja melhor que outro modelo.

### Divisão Única Repetida, Holdout Repetido

Para remediar esse problema, temos que fazer mais experimentos, ou seja, repetir a divisão do conjunto em treinamento e teste, a geração do modelo e a estimativa do desempenho. Uma solução relativamente simples é fazer uma sequência de experimentos, obter uma estimativa de desempenho para cada experimento, e depois a média aritmética de todos os experimentos é o critério de desempenho estimado final. Obviamente, a divisão do conjunto de  $n$  padrões em treinamento e teste tem que variar de experimento para experimento. Considerando a distribuição hipergeométrica, pela definição do valor esperado na (eq. 9.29), a estratégia da repetição direciona a proporção dos padrões de cada classe para um valor mais representativo. Mas isso não garante que a estimativa de desempenho não tenha um viés por outra razão, um modelo inapropriado, uma quantidade muito pequena de padrões, ou problemas menos óbvios associados à tarefa de aprendizado de máquina em questão, por exemplo, dependência entre conjuntos de treinamento e teste. Não podemos esquecer que as técnicas de estimativa de desempenho devem fazer afirmações principalmente sobre o comportamento *futuro* do modelo, e em princípio, dados futuros serão disponíveis somente no futuro e podem desviar do conjunto presente disponível de  $n$  padrões. E esses dados devem ser da mesma distribuição que os dados que foram usados para treinar o modelo, mas têm que ser inde-

Figura 9.12: Divisão de dados em  $k = 4$  folds.

pendentes<sup>9</sup>. Como exemplo de um *holdout* repetido considere a primeira coluna da tabela 9.1, cuja média é 85.60% o que corresponde ao *holdout* dez vezes repetido com uma divisão entre duas partes iguais para treinamento e teste. Uma vantagem do *holdout* é a baixa complexidade computacional, pois um único ciclo de treinamento é necessário.

### Validação Cruzada $k$ -fold

A figura 9.12 mostra como  $n$  padrões são subdivididos em  $k$  partes, se for possível, exatamente iguais, e em caso de classificação, ainda em proporções estratificadas. Um método popular para estimar o desempenho é o  $k$ -fold. O conjunto é dividido nos *folds* que são partes do conjunto total de  $n$  padrões que não se sobrepõem. Para cada *fold*, um experimento com treinamento do modelo e teste posterior é realizado, aqui denominado de *ciclo*. Em cada ciclo um dos  $k$  *folds* assume o papel do conjunto de teste, os restantes ( $k - 1$ ) *folds* assumem o papel do conjunto de treinamento. O ajuste do modelo é feito, o *fold* omitido do treinamento é alimentado ao modelo, e o desempenho é registrado. Esses ciclos são repetidos  $k$  vezes. A consequência implícita dessa estratégia é que todos os padrões fazem parte dos conjuntos de treinamento e teste, porém não simultaneamente como no caso da resubstituição. A média dos desempenhos dos  $k$  ciclos é o desempenho final<sup>10</sup>. No algoritmo 17 os passos são explicitados.

### Validação Cruzada *leave-one-out*

Temos  $n$  padrões no total. Podemos levar o  $k$ -fold ao extremo, e igualar a quantidade de *folds* ao número total de padrões. Então simplesmente temos que definir  $k = n$  no algoritmo 17. Isto significa que em cada ciclo de treinamento e teste, o conjunto de teste é composto por um único padrão. O problema principal, embora o valor obtido do critério de desempenho seja mais confiável, é o alto custo computacional. Imagine uma rede neural bastante complexa, e um número muito grande de exemplos de treinamento, com muitas características na entrada e muitas saídas na rede. Essa constelação seria computacionalmente inviável para obter uma estimativa de desempenho pelo *leave-one-out*.

### Estimativa de Desempenho por *Bootstrapping*

Já tivemos contato com a técnica de seleção arbitrária pelo *bootstrapping* da definição 8.5.1 na pág. 253 no contexto da floresta de árvores de decisão ou regressão. Aproximadamente 63.21%

<sup>9</sup>Em inglês, usa-se a sigla “iid.”, *Independent and identically distributed random variables*.

<sup>10</sup>Alternativamente, todos os resultados de cada uma das  $n$  avaliações individuais de cada padrão podem ser concatenados e a média final de  $n$  avaliações pode ser calculado, chegando no mesmo resultado.

**Algoritmo 17:** Estimativa de desempenho pelo método  $k$ -fold

**Entrada:** Conjunto de dados de treinamento de  $n$  padrões  $\mathcal{D} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ ; número  $k$  de  $folds$ ; modelo de aprendizado de máquina

**Resultado:** Estimativa de desempenho do modelo de aprendizado de máquina

**Divisão de dados:**

Divida aleatoriamente os  $n$  padrões em  $k$  partes não sobrepostas de (aproximadamente)  $k/n$  padrões, veja a figura 9.12;

**Estimativa de desempenho:**

**para**  $\ell = 1, \dots, k$  **faça**

- 1 Defina o  $\ell$ -ésimo *fold* como conjunto de teste;
- 2 Defina a união de todos os restantes  $(k - 1)$  *folds* como conjunto de treinamento;
- 3 Treine o modelo, usando o conjunto de treinamento;
- 4 Teste o modelo, usando o conjunto de teste e memorize o  $k$ -ésimo desempenho  $p_k$  do experimento;
- 5 **fim**
- 6  $p \leftarrow \overline{p_k}$  /\* Desempenho final é média dos  $k$  *folds* \*/ ;

de um conjunto é escolhido quando se aplica reposição dos elementos já escolhidos. Este método pode obter do conjunto completo, uma amostra representativa, supostamente com as mesmas propriedades estatísticas. Junto com uma correção baseada nas propriedades estatísticas dos desempenhos dessas repetições, é possível obter uma estimativa final do desempenho. Podemos já antecipar que este caminho é mais difícil de implementar, quando comparado com técnicas mais convencionais, como, por exemplo, a validação cruzada por  $k$ -fold, fornecendo resultados parecidas. A ideia original do método do *bootstrap* [31, 32, 33] é a estimativa de um parâmetro estatístico, baseada na amostragem com reposição de  $n$  valores de uma distribuição. O princípio do *bootstrap* postula que a distribuição da amostragem seja semelhante à distribuição da verdadeira distribuição da variável aleatória, e que uma estatística calculada da amostragem tenha a mesma variação de uma estatística calculada da distribuição original. Apresenta-se aqui a aplicação deste princípio à estimativa de desempenho. A nomenclatura de [43], sec. 7.11 é replicada aqui. Para entender o método profundamente, os autores alertem que envolve um teoria não trivial. Na prática preferimos métodos de estimativa de desempenho mais simples, como  $k$ -fold, que fornecem valores confiáveis, porém com menos complexidade. Será experimentalmente confirmado em breve.

Sejam gerados  $B$  conjuntos de dados  $\mathcal{D}_b$ , cada um pelo método de *bootstrap*, (eq. 8.46) na pág. 253, a partir de um conjunto de treinamento de  $n$  padrões

$$\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_i, \mathbf{y}_i), \dots, (\mathbf{x}_n, \mathbf{y}_n)\}.$$

Cada conjunto de *bootstrap*  $\mathcal{D}_b$ ,  $b = 1, \dots, B$  contém  $|\mathcal{D}_b| = n_b < n$  elementos não repetidos escolhidos pela seleção aleatória com reposição, aproximadamente  $n_b = n \cdot 61.21\%$ . Uma tentativa certamente otimista demais seria treinar o modelo com o conjunto original  $\mathcal{D}$  e testar com o conjunto de *bootstrap*  $\mathcal{D}_b$ . O viés otimista se deve à sobreposição dos dois conjuntos, ou seja, usariam duas vezes a mesma informação. O problema é conhecido pelo caso extremo da resubstituição da seção 9.5.1, onde todos os  $n$  padrões de teste já foram usados na fase de treinamento.

A função de mapeamento do modelo seja chamada aqui  $\mathbf{f}$  que para a entrada do  $i$ -ésimo padrão  $\mathbf{x}_i$  produz a  $i$ -ésima saída explicada  $\hat{\mathbf{y}}_i$ , em geral diferente do valor desejado  $\mathbf{y}_i$ . A função de perda<sup>11</sup>

<sup>11</sup>No contexto de modelos lineares no capítulo 2, e descida de gradiente no capítulo 3, a função de perda  $L$  foi

seja definida como

$$L(\mathbf{y}_i, \hat{\mathbf{y}}_i) = L(\mathbf{y}_i, \mathbf{f}(\mathbf{x}_i)).$$

Lembre que a função mais usada é o Erro Quadrático (EQ)  $L(\mathbf{y}, \hat{\mathbf{y}}) = \|\mathbf{y} - \hat{\mathbf{y}}\|^2$ . Caso todos os  $n$  padrões do conjunto completo  $\mathcal{D}$  forem usados para treinar o modelo, a função de mapeamento mantém o nome  $\mathbf{f}$ , caso somente o conjunto reduzido de *bootstrap*  $\mathcal{D}_b$  for usado para treinar o modelo, a função de mapeamento será chamada de  $\mathbf{f}^{*b}$ . O *erro de resubstituição* da seção 9.5.1, onde o conjunto total é usado para treinamento e posteriormente usado para teste então é

$$\overline{\text{err}} \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n L(\mathbf{y}_i, \mathbf{f}(\mathbf{x}_i)). \quad (9.30)$$

Já vimos que essa estimativa desvia completamente de valores realísticos na direção de otimismo, e é inapropriado na prática. O erro que usa um único conjunto de *bootstrap*, ou seja,  $B = 1$ , é definido como

$$\widehat{\text{Err}}_{\text{boot}} \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n L(\mathbf{y}_i, \mathbf{f}^{*b}(\mathbf{x}_i)). \quad (9.31)$$

A estimativa melhora em relação à resubstituição, porém por causa do uso sobreposto em treinamento e teste ainda há um viés otimista. Calculado a média sobre  $B$  conjuntos de *bootstrap* temos

$$\widehat{\text{Err}}_{\text{boot}} \stackrel{\text{def}}{=} \frac{1}{B} \frac{1}{n} \sum_{b=1}^B \sum_{i=1}^n L(\mathbf{y}_i, \mathbf{f}^{*b}(\mathbf{x}_i)). \quad (9.32)$$

Neste caso, o modelo é treinado somente com os padrões escolhidos pelo *bootstrap*, porém todos esses padrões, além dos restantes, aparecem no conjunto de teste. Novamente existe sobreposição de informação que retorna um número otimista demais para o desempenho do modelo.

A moldura algorítmica usada para obter a estimativa de desempenho no contexto do *bootstrap* é o *leave-one-out*, imitando a validação cruzada onde cada padrão é usado uma única vez para testar, enquanto os restantes padrões formam o conjunto de treinamento. Como o conjunto de *bootstrap*  $\mathcal{D}_b$  eventualmente contém esse mesmo  $i$ -ésimo padrão, ele será omitido no teste. Isso significa que vamos ignorar todos os conjuntos  $\mathcal{D}_b$  como conjunto de treinamento, se o padrão  $\mathbf{x}_i$  estiver nesse conjunto. Assim podemos definir o erro melhorado para um conjunto de *bootstrap*  $\mathcal{D}_b$

$$\widehat{\text{Err}}^{(1)} \stackrel{\text{def}}{=} \frac{1}{B} \sum_{b=1}^B \frac{1}{n-b} \sum_{\substack{i=1 \\ i \notin \mathcal{D}_b}}^n L(\mathbf{y}_i, \mathbf{f}^{*b}(\mathbf{x}_i)). \quad (9.33)$$

Entendemos aqui  $\mathcal{D}_b$  como os índices dos padrões que estão no conjunto de *bootstrap*. A quantidade  $n-b$  representa quantas vezes os índices  $i$  não apareceram nos índices do conjunto  $\mathcal{D}_b$ .

■ **Exemplo 9.6** Imagine um conjunto ilustrativo de apenas cinco padrões, então o conjunto de índices seria  $\mathcal{D} = \{1, 2, 3, 4, 5\}$ . Vamos gerar  $B = 3$  conjuntos de *bootstrap*. Após a eliminação dos índices duplicados devido à reposição sejam eles  $\mathcal{D}_1 = \{2, 3, 5\}$ ,  $\mathcal{D}_2 = \{2, 5\}$  e  $\mathcal{D}_3 = \{1, 2, 4\}$ . Então teríamos para  $b = 1$ , o valor  $n-1 = 2$ , pois os índices 1 e 4 não aparecem em  $\mathcal{D}_1$ ,  $n-2 = 3$ , pois os índices 1, 3 e 4 não aparecem em  $\mathcal{D}_2$ , e, finalmente,  $n-3 = 2$ , pois  $3 \notin \mathcal{D}_3$  e  $5 \notin \mathcal{D}_3$ . ■

---

definida com os parâmetros livres como argumentos. Aqui usamos o valor esperado  $\mathbf{y}$  e o valor explicado pelo modelo  $\hat{\mathbf{y}}$  como parâmetros. Os parâmetros livres  $\boldsymbol{\theta}$  dependem do modelo usado, e estão implicitamente no cálculo da função  $\mathbf{f}$  do modelo.

A estimativa da (eq. 9.33) condiz com a separação de informação entre treinamento e teste, usado no *leave-one-out*, porém o padrão é somente testado, se ele não aparece no conjunto de *bootstrap* que foi usado para treinar o modelo.

Mesmo com essa melhoria, ainda existe um viés que se deve ao número reduzido de padrões no conjunto de treinamento de aproximadamente 63.2% do número total  $n$  de padrões. O estimador “.632” tenta remediar esse viés, ponderando entre o erro de resubstituição e o erro simulado *leave-one-out*, definido como

$$\widehat{\text{Err}}^{(.632)} \stackrel{\text{def}}{=} .368 \cdot \overline{\text{err}} + .632 \cdot \widehat{\text{Err}}^{(1)}. \quad (9.34)$$

A teoria atrás desse raciocínio não é trivial, segundo [43]. Não nos aprofundamos mais nesse caminho. Mais uma melhoria é proposta na forma do estimador “.632+”, definido como

$$\widehat{\text{Err}}^{(.632+)} \stackrel{\text{def}}{=} (1 - \hat{w}) \cdot \overline{\text{err}} + \hat{w} \cdot \widehat{\text{Err}}^{(1)}, \quad (9.35)$$

onde a ponderação melhorada entre erro de resubstituição e erro simulado *leave-one-out* é definido como

$$\hat{w} = \frac{.632}{1 - .368\hat{R}}, \quad (9.36)$$

e a quantidade  $\hat{R}$  na (eq. 9.37) representa a taxa relativa de sobreajuste (*relative overfitting rate*), calculada como

$$\hat{R} = \frac{\widehat{\text{Err}}^{(1)} - \overline{\text{err}}}{\hat{\gamma} - \overline{\text{err}}}, \quad (9.37)$$

A estatística  $\gamma$  é o “erro sem informação”, definida como

$$\hat{\gamma} = \frac{1}{n^2} \sum_{i=1}^n \sum_{i'=1}^n L(\mathbf{y}_i, \mathbf{f}(\mathbf{x}_{i'})). \quad (9.38)$$

Elá submete qualquer entrada  $\mathbf{x}_{i'}$ , e compara a saída correspondente  $\mathbf{f}(\mathbf{x}_{i'})$  com uma saída específica  $\mathbf{y}_i$ . Essa classificação age, como se as entradas e saídas fossem independentes. O cálculo tem que ser feito uma única vez para um conjunto de dados  $\mathcal{D}$  e um modelo representado pela função de mapeamento  $\mathbf{f}$ .

### 9.5.2 Dados Sintéticos como Verdade Fundamental

As técnicas apresentadas para estimar desempenhos até agora estudados, permitem nos dar uma ideia da qualidade de um modelo, ou a dificuldade de classificar corretamente um problema, ou avaliar uma regressão. Podemos dizer que, por exemplo, usando uma árvore de classificação e o *leave-one-out* para estimar o erro, o conjunto de iris vai nos permitir<sup>12</sup> obter em torno de 94.0% de acurácia. Porém, não temos a chamada *ground truth*, a verdadeira informação sobre a distribuição probabilística das flores. Essa situação muda se nos mesmos somos “donos” dos dados, ou seja, definimos uma distribuição e dispararmos um gerador de números aleatórios para produzir uma quantidade finita de amostras dessa distribuição. Podemos, por exemplo gerar 100 padrões de uma distribuição Gaussiana, fornecendo os parâmetros da distribuição, neste caso o vetor de médias e a matriz de covariância. Definindo uma segunda Gaussiana e atribuindo essa

<sup>12</sup>Como exercício, verifique o valor de 94.0% de acurácia estimada pela biblioteca de aprendizado de máquina `scikit-learn`, usando as classes `sklearn.tree.DecisionTreeClassifier` para definir o classificador, `sklearn.model_selection.LeaveOneOut` para definir o método de validação cruzada, e a função `sklearn.datasets.load_iris()` para obter os dados das flores.

segunda distribuição a uma outra classe, torne-se possível calcular analiticamente a taxa de erro, o limite Bayesiano verdadeiro [29], além de qual qualquer resultado é otimista demais. No trabalho pioneiro de Fukunaga [63] alguns conjuntos de dados sintéticos com distribuição Gaussiana são definidas com duas classes e a taxa de erro de Bayes é informada. Com o conhecimento do limite teórico, podemos ganhar uma visão interessante sobre o nosso modelo, por exemplo afirmar com certeza que certas taxas de desempenho não sejam realísticas, ou que tal método de avaliação seja melhor que outro.

Três conjuntos de pares de Gaussianas são propostas em um problema de classificação binária. A dimensão do espaço de características é oito, ou seja, qualquer padrão é um ponto no espaço  $\mathbf{x} \in \mathbb{R}^8$ . As duas classes  $\mathcal{C}_a$  e  $\mathcal{C}_b$  de cada problema são inequivocamente definidas pelos seus vetores de médias  $\boldsymbol{\mu}_a$ ,  $\boldsymbol{\mu}_b$  e suas matrizes de covariância  $\boldsymbol{\Sigma}_a$ ,  $\boldsymbol{\Sigma}_b$ , respectivamente. Não existe covariância nos conjuntos definidos por Fukunaga, ou seja, os elementos fora da diagonal  $\sigma_{ij}$ ,  $i, j = 1, \dots, 8$  em ambas as matrizes de covariância  $\boldsymbol{\Sigma}_a$  e  $\boldsymbol{\Sigma}_b$  são zero. Isto é justificável, pois qualquer par de matrizes de covariância com elementos não nulos fora da diagonal podem ser simultaneamente diagonalizadas por uma transformação linear. Além disso, o vetor de média não nulo pode ser transladado para a origem do sistema de coordenadas sem prejudicar o caso geral, definido por vetores de média não nulos e matrizes de covariância não nulas fora da diagonal [63]. Assim, somente 32 parâmetros têm que ser especificados para definir as duas funções de densidade de probabilidade, dois vetores de média de dimensão oito, e duas vezes oito variâncias nas diagonais das matrizes de covariância. As duas probabilidades a priori idênticas  $P(\mathcal{C}_a) = P(\mathcal{C}_b) = 50\%$  das duas classes podem ser simulados por uma quantidade  $n'$  igual de amostras para cada classe. Para garantir reproduzibilidade por outros pesquisadores, na geração aleatória de  $n'$  amostras por classes, ou seja, um total de  $n = n' * 2$  amostras, a semente do gerador de números aleatórios<sup>13</sup> tem que ser fornecida, como já mencionado na seção 1.3.3 na pág. 22. A tabela 9.12 mostra a especificação dos parâmetros<sup>14</sup> de cada um dos três conjuntos de dados por Fukunaga. Adicionalmente, o chamado conjunto “null” de [113] é incluído na tabela, onde as duas Gaussianas têm média zero e como matriz de covariância a matriz de identidade. Dessa maneira as duas distribuições se sobrepõem perfeitamente, dando origem a uma taxa de erro de classificação de 50%, pois a decisão a qual classe um padrão pertence é equivalente a jogar uma moeda e verificar cara ou coroa.

### 9.5.3 Experimentos com Dados Sintéticos

A tabela 9.13 mostra um experimento simples para estimar o erro, ou complementarmente, a acurácia de um classificador com um dos conjuntos da tabela 9.12, o “I-Λ” que tem um limite teórico de 98.1% para a acurácia. A quantidade de amostras geradas para cada uma das duas classes é 1000, portanto somando  $n = 2000$  amostras no total. Foram comparados a resubstituição, a validação cruzada com  $K = 5$ , o *leave-one-out*, e as estimativas obtidas pelas técnicas estatísticas baseado no *bootstrap* com  $B = 100$  repetições de seleção por reposição. Além da acurácia, o custo computacional<sup>15</sup> de cada técnica é registrada. O teste revela que o custo computacional das técnicas de *bootstrap* são muito altas. Embora tenha que ser feito somente uma única vez para um conjunto de treinamento e um modelo de classificador, o cálculo do parâmetro  $\hat{\gamma}$  da (eq. 9.38) é muito caro. A complexidade de uma única rodada de *bootstrap* é igual ao *leave-one-out*. Essa complexidade então tem que ser multiplicada ainda pela quantidade  $B$  das repetições. Pelo menos

<sup>13</sup>A semente é definida novamente com o valor 66649. A classe de Python `numpy.random` com o método `numpy.random.normal` foi usado para gerar as amostras da distribuição Gaussiana, por exemplo, a invocação de `numpy.random.normal(loc=numpy.zeros(8), scale=numpy.sqrt(numpy.diag(4*numpy.eye(8))), size=(100,8))` gera  $n = 100$  amostras da segunda classe  $\mathcal{C}_b$  do conjunto de dados “I-4I” da tabela 9.12, com vetor de médias zero  $\boldsymbol{\mu} = \mathbf{0}$  e variâncias  $\sigma_i^2 = 4.0$  na diagonal,  $i = 1, \dots, 8$ .

<sup>14</sup>Relembrando que o operador “`diag`” com  $m$  argumentos especifica uma matriz quadrada de dimensão  $m \times m$ , onde os argumentos formam a diagonal.

<sup>15</sup>Tempo de CPU contemporânea convencional à confecção deste teste com 3.6 GHz, em segundos.

Tabela 9.12: Parâmetros dos conjuntos de dados de distribuição Gaussiana de [63], além do conjunto “null” de [113], informando o limite teórico da acurácia atingível em um problema de classificação binária.

Conjunto	Classe	Vetor de Médias $\mu$	Matriz de Covariância $\Sigma$	Acurácia Bayesiana
“I-I”	$C_a$	[0,0,0,0,0,0,0]	diag(1,1,1,1,1,1,1)	
	$C_b$	[2.56,0,0,0,0,0,0]	diag(1,1,1,1,1,1,1)	90.0%
“I-4I”	$C_a$	[0,0,0,0,0,0,0]	diag(1,1,1,1,1,1,1)	$\approx 91.0\%$
	$C_b$	[0,0,0,0,0,0,0]	diag(4,4,4,4,4,4,4)	
“I- $\Lambda$ ”	$C_a$	[0,0,0,0,0,0,0]	diag(1,1,1,1,1,1,1)	
	$C_b$	[3.86,3.10,0.84,0.84, 1.64,1.08,0.26,0.01]	diag(8.41,12.06,0.12,0.22, 1.49,1.77,0.35,2.73)	$\approx 98.1\%$
“null”	$C_a$	[0,0,0,0,0,0,0]	diag(1,1,1,1,1,1,1)	
	$C_b$	[0,0,0,0,0,0,0]	diag(1,1,1,1,1,1,1)	50.0%

Tabela 9.13: Acurácia estimada para o conjunto de dados “I- $\Lambda$ ” da tabela 9.12, usando o classificador da seção 7.2.4 na pág. 197, Análise Discriminativa Quadrática. A acurácia teórica de Bayes é 98.1%.

	Resubs- tituição	5-fold	Leave- one-out	Bootstrap				
				$\hat{\gamma}$	$\widehat{Err}_{boot}$	$\widehat{Err}^{(1)}$	$\widehat{Err}^{(.632)}$	$\widehat{Err}^{(.632+)}$
Acurácia	98.30%	98.25%	98.30%	50%	98.3525%	98.3421%	98.3266%	98.3266%
Tempo CPU	0.004	0.013	2.003	355.40			139.298	

neste experimento os resultados são bastante parecidos, o que não justificaria o uso do *bootstrap*, pois uma validação cruzada de  $k$ -fold fornece resultados equivalentes. Podemos ainda observar que as acuráncias estimadas ultrapassam o limite teórico de 98.1% com todos os métodos, exceto para o  $\hat{\gamma}$  que representa uma classificação sem informação. Esse viés se deve ao número finito de amostras geradas. Em um experimento adicional veremos a extrema influência da quantidade de amostras no conjunto de treinamento, e o perigo de falsas conclusões sobre o desempenho de um modelo, se a quantidade de amostras é limitada e pequena.

## 9.6 Validação Cruzada e Seleção de Modelo

Além de estimar o desempenho de um modelo, as técnicas de avaliação, como por exemplo, o  $k$ -fold, têm outras funções muito importantes. São elas

1. Ajuste dos hiperparâmetros do modelo;
2. Análise estatística do resultados para fins de comparação de modelos.

Esses dois assuntos podem ser combinados, por exemplo, podemos conceber algoritmos que tentam achar os melhores hiperparâmetros e simultaneamente produzir um conjunto de valores de desempenho que possam ser analisados em um teste estatístico. Vamos primeiro considerar o assunto do ajuste dos hiperparâmetros. Algum modelo de regressão ou classificação normalmente tem parâmetros e hiperparâmetros. O método de treinamento serve para ajustar os parâmetros.

Uma obrigação adicional é a determinação dos outros graus de liberdade do modelo<sup>16</sup>. Como exemplo, considere o classificador K-Vizinhos-Mais-Próximos. Se não usarmos nenhuma técnica mais sofisticada de otimizar a organização dos dados, o treinamento é trivial, não temos que fazer nada. Então essa parte de aprendizagem estamos economizando. Por outro lado, podemos ainda escolher o número de vizinhos, além de outros hiperparâmetros. Será que um, três, cinco, ou mais vizinhos é o melhor hiperparâmetro para ter o melhor classificador possível para o problema em consideração?

**Aviso:** Existe um perigo não negligenciável que a estimativa seja otimista demais, se forçarmos o modelo nessa direção, por exemplo, por repetições do mesmo experimento, mudando os hiperparâmetros do modelo até observar o melhor resultado. Nesse caso usariamos a mesma informação mais que uma vez, violando o princípio da independência estatística.

Para o classificador do K-Vizinhos-Mais-Próximos, por exemplo, podemos imaginar o método de *leave-one-out* para obter a estimativa da acurácia de um conjunto de dados, variando o número de vizinhos, mas não variando a validação cruzada. No fundo estamos usando informação duas vezes, e isso muito provavelmente nos levará a uma estimativa da acurácia verdadeira que é alta demais.

Para integrar a validação cruzada e a seleção do modelo, propõe-se uma estrutura aninhada (*nested*) com laços interiores e laços externos que tenta evitar o uso duplicado de informação para evitar um viés dos valores estimados em relação aos valores verdadeiros. Assim os resultados são mais perto da realidade. Na figura 9.13 a ideia da validação cruzada aninhada é ilustrada. O termo “Seleção de Modelo” neste contexto significa escolher um hiperparâmetro. A parte externa tem a mesma função como antes, uma validação cruzada  $k$ -fold que é responsável de dividir o conjunto completo de  $n$  padrões em  $k$  partes. Em cada um dos  $k$  ciclos executados, a  $k$ -ésima parte fica reservada para testar um modelo treinado. A diferença em relação à técnica convencional é que os restantes  $(k - 1)$  partes não são usadas para treinar o modelo e depois imediatamente submeter a  $k$ -ésima para testar. O conjunto das  $(k - 1)$  partes é chamado “Treinamento e Validação” na figura 9.13. Esta parte é submetida à tarefa de *tuning*<sup>17</sup>. O *tuning* é responsável pela busca do melhor modelo. Neste caso específico, podemos modificar hiperparâmetros. Por exemplo, podemos procurar em um classificador K-Vizinhos-Mais-Próximos qual é o melhor número de vizinhos. É muito importante notar que cada vez que utilizados o bloco do *tuning*, um outro melhor hiperparâmetro pode sair. Esse fato se explica pelo conjunto de dados submetido ao *tuning* que em geral muda com cada conjunto de treinamento e validação recebido dos ciclos externos. No algoritmo 18 a validação aninhada é explicitada.

### 9.6.1 Melhores hiperparâmetros por um único ciclo externo

A única maneira de definir qual é “o melhor” conjunto de hiperparâmetros é limitar a quantidade de ciclos externos a um **único**. Por exemplo, na figura 9.13, o ciclo externo 1, submetido ao *tuning* fornece um **único** conjunto de hiperparâmetros. O preço a ser pago é uma significância estatística reduzida [15]. Será que este conjunto de hiperparâmetros obtido por um único *tuning* é realmente o melhor? As afirmações sobre o desempenho do modelo com este conjunto otimizado de hiperparâmetros obviamente podem sofrer de um viés, pois é um único ciclo externo.

■ **Exemplo 9.7** Seja o problema de aprendizagem supervisando a classificação das nossas  $n = 150$  flores, e seja o modelo o classificador uma árvore de decisão definida na seção 8.3 na pág. 229. O

<sup>16</sup>Frequentemente em publicações, experimentos computacionais são apresentados, onde os hiperparâmetros por padrão (*default*) de um software específico foram usados, por exemplo, a profundidade máxima de uma árvore, ou o tipo de *kernel* associado a uma distância. Essa prática é perigosa, pois pode levar a conclusões erradas.

<sup>17</sup>pronunciado “t-i-u-n-i-n-g”

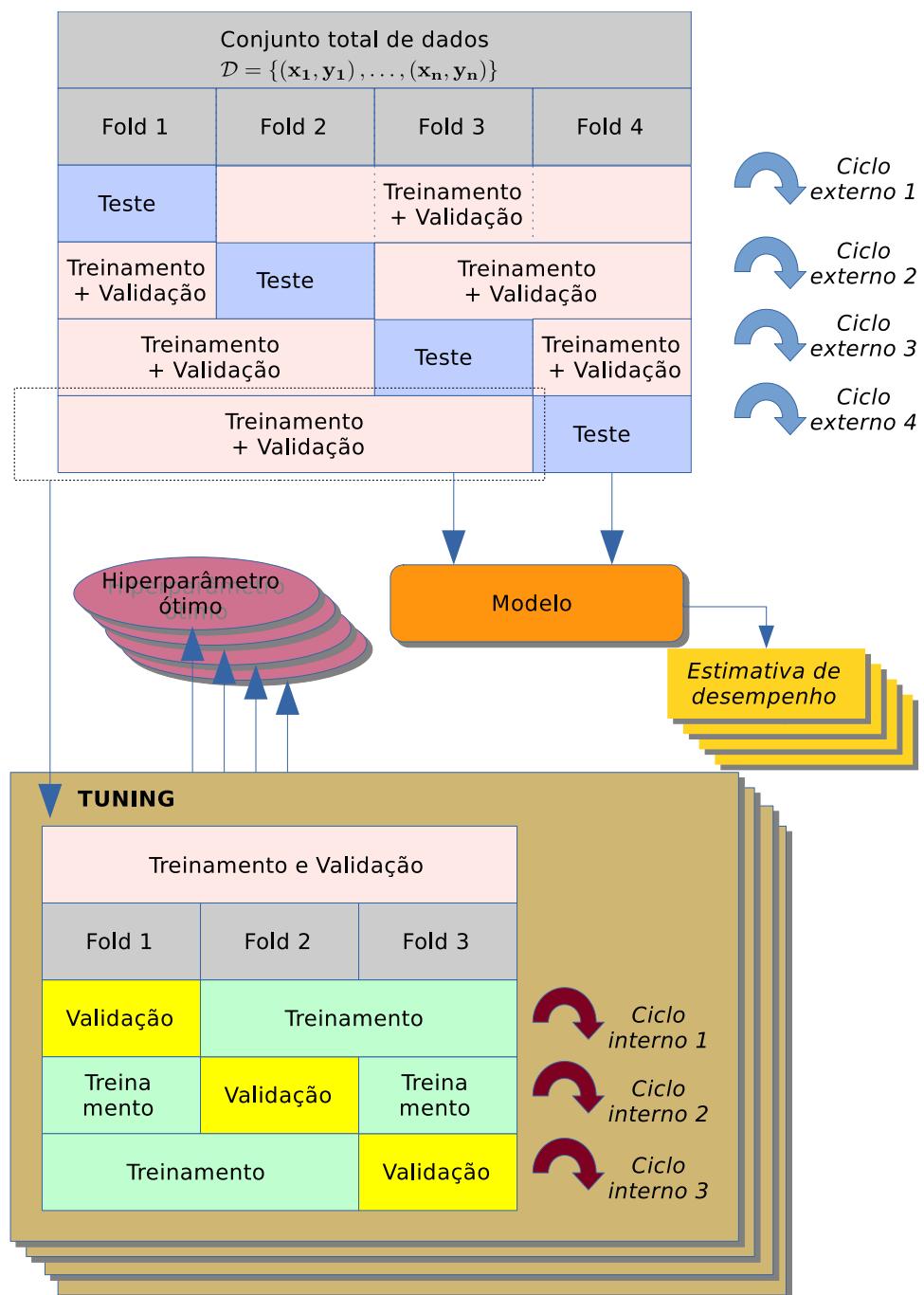


Figura 9.13: Validação cruzada aninhada para selecionar modelo.

**Algoritmo 18:** Validação cruzada aninhada.

---

```

1 Function desempenho  $\leftarrow$  DesempenhoModelo( $\mathcal{D}$ ,  $K$ ,  $L$ )
   Entrada:  $n$  padrões rotulados  $\mathcal{D}$  de  $c$  classes, número de folds  $K$  (ciclos externos),  $L$ 
      (ciclos internos)
   Saída: Faça  $K$  folds estratificados. Um fold é fixado como teste, os folds restantes
      são usados para treinamento e tuning. Armazene o critério de desempenho
      para cada ciclo externo para processamento posterior, por exemplo, o cálculo
      da média aritmética.
2   // Gere  $K$  folds estratificados
3    $f_k, k = 1, \dots, K;$ 
4   para  $k = 1$  até  $K$  faça // ciclo externo
5     // conjunto de treinamento do  $k$ -ésimo fold
6      $\mathcal{T} \leftarrow \{f_1 \cup \dots \cup f_K\} \setminus f_k;$ 
7     // conjunto de teste do  $k$ -ésimo fold
8      $\mathcal{V} \leftarrow f_{r,k};$ 
9      $\mathcal{P}_k^* \leftarrow \text{Tuning}(\mathcal{T}, L);$ 
10     $\mathcal{C} \leftarrow \text{treinamentoModelo}(\mathcal{T}, \mathcal{P}_k^*);$ 
11     $\text{crit} \leftarrow \text{testeModelo}(\mathcal{C}, \mathcal{V});$ 
12     $\text{Desempenho}(k) \leftarrow \text{crit}$ 
13  fim
14   $\text{desempenhofinal} \leftarrow \overline{\text{Desempenho}(k)}, \quad k = 1, \dots, K;$ 
15 end
16
17 Function  $\mathcal{P}^* \leftarrow \text{Tuning}(\tilde{\mathcal{D}}, L)$ 
   Entrada: Conjunto de dados  $\tilde{\mathcal{D}}$ , número  $L$  de folds de tuning
   Saída: Conjunto de hiperparâmetros ótimos  $\mathcal{P}^*$  do modelo
18   Initialize critério ótimo  $\text{maxcrit} \leftarrow 0;$ 
19   // Gere  $L$  folds estratificados
20    $f_\ell, \ell = 1, \dots, L;$ 
21   repita // Busca (exaustiva) pelo melhor conjunto de hiperparâmetros  $\mathcal{P}^*$ 
22     gere conjunto de candidatos de hiperparâmetros  $\mathcal{P}$ ;
23     para  $\ell = 1$  até  $L$  faça // todos os ciclos internos
24       // conjunto de treinamento do  $k$ -ésimo fold
25        $\mathcal{T} \leftarrow \{f_1 \cup \dots \cup f_L\} \setminus f_\ell;$ 
26       // conjunto de teste do  $k$ -ésimo ciclo interno
27        $\mathcal{V} \leftarrow f_\ell;$ 
28        $\mathcal{C} \leftarrow \text{treinamentoModelo}(\mathcal{T}, \mathcal{P});$ 
29        $\text{crit}_\ell \leftarrow \text{treinamentoModelo}(\mathcal{C}, \mathcal{V});$ 
30     fim
31      $\text{crit} \leftarrow \overline{\text{mean}}(\text{crit}_\ell);$ 
32     se  $\text{crit} > \text{maxcrit}$  então
33        $\text{maxcrit} \leftarrow \text{crit}; \mathcal{P}^* \leftarrow \mathcal{P}$ 
34     fim
35   até busca finalizada;
36 end
```

---

critério de desempenho seja a acurácia estimada. Vamos supor que o único hiperparâmetro a ser ajustado é a profundidade da árvore. Então, o conjunto de hiperparâmetros é composto por um hiperparâmetro somente. Podemos escolher a profundidade entre os valores 1, 2, 3, 4, 5 e 6, por exemplo. Todos as possibilidades de escolha constituem o *espaço de busca* do hiperparâmetro, que neste caso tem, seis combinações possíveis. Se tivéssemos um segundo hiperparâmetro a ser variado, teríamos o produto Cartesiano de primeiro e segundo hiperparâmetro. Por exemplo, se variarmos o critério de bifurcação também, teríamos doze combinações, as seis possíveis profundidades e os dois possíveis critérios “ganho de impureza gini” e “ganho de entropia”, veja a (eq. 8.32) na pág. 237 e a (eq. 8.36d).

No gráfico da figura 9.13, o bloco externo tem  $k = 4$  folds. Então teremos 37 ou 38 padrões em cada fold, somando 150. No primeiro ciclo externo um quarto é reservado para testar. Os restantes três quartos, somando 112 padrões<sup>18</sup> são submetidos ao *tuning*. Dentro do bloco de *tuning*, os 112 padrões passados são divididos em três folds, ou seja, em três grupos de 38, 37, e 37 padrões. Note que a quantidade de folds não é necessariamente igual à quantidade do bloco externo, menos um, ou seja neste caso  $4 - 1 = 3$ . No primeiro, segundo, e terceiro ciclo interno do *tuning*, 74, 75, e 75 padrões são utilizados para treinar (grupo “Treinamento”) a árvore, e 38, 37, e 37 padrões do mesmo ciclo são usados para obter um estimativa de desempenho (grupo “Validação”). Após os três ciclos internos, temos a estimativa final como média dos três ciclos internos. Esta estimativa diz respeito a uma única combinação dos valores possíveis dos hiperparâmetros, a profundidade de valor 1. Lembrando que podemos somente variar a profundidade da árvore, a primeira estimativa com profundidade 1 retorna como média dos três ciclos internos uma acurácia estimada de 66.96%.

Temos que repetir esses três ciclos internos para todas as combinações dos valores dos hiperparâmetros, neste caso para as possíveis profundidades 2, 3, 4, 5, 6. As acurácia estimados para esses combinações do hiperparâmetro foram 91.96%, 92.86%, 94.64%, 91.96%, 92.86%. Então o hiperparâmetro ótimo do primeiro ciclo **externo** foi a profundidade com o valor 4, pois deu o melhor resultado de 94.64% acurácia estimada.

Se o nosso objetivo for achar “o melhor” conjunto de hiperparâmetros, teríamos que parar neste ponto. O resultado seria: “A melhor profundidade da árvore tem quatro níveis. Tendo este hiperparâmetro ótimo na mão, o fold de teste de 38 padrões é submetido a uma árvore treinada com o conjunto “Treinamento+Validação” do primeiro cinclo externo. É justificável usar a parte de “Validação” também no treinamento, pois o conjunto de teste nunca foi usado durante a fase de *tuning*. O resultado foi uma acurácia estimada de 92.11%. Repare que este resultado é pior que o resultado obtido pelo hiperparâmetro ótimo de profundidade 4 no bloco do *tuning*. ■

### 9.6.2 Melhores hiperparâmetros diferentes por ciclos externos diferentes

Como já foi apontado, a repetição de vários ciclos externos pode fornecer vários conjuntos de hiperparâmetros diferentes. Temos que nos despedir da ideia de achar “o melhor”. Ganhamos mais segurança estatística com vários ciclos externos, por exemplo podemos obter estimativas de desempenhos para cada ciclo externo e escolher a média desses valores como resultado final.

■ **Exemplo 9.8** Temos que percorrer todos os estágios descritos até agora no exemplo 9.7 para os ciclos externos 2, 3, e 4, ou seja repetir o *tuning* para cada um desses ciclos, novamente executando os ciclos internos para todas as combinações dos hiperparâmetros possíveis. Fica óbvio, para evitar uma explosão combinatória, que não podemos simplesmente aumentar as combinações dos hiperparâmetros, a quantidade de folds, e o modelo de regressão ou classificação. Certamente a sintonia de todos esses graus de liberdade depende do problema analisado e seus dados fornecidos. As estimativas dos ciclos externos 2, 3, e 4 foram 92.11%, 91.89%, e 97.30%, respectivamente,

<sup>18</sup>Os resultados descritos neste exemplo foram implementados no `sklearn`.

com profundidades da árvore 2, 6, e 4 respectivamente. Observe mais uma vez que não existe um único hiperparâmetro ótimo, pois nos quatro ciclos externos, as profundidades foram 4, 2, 6, 4.

O resultado final da estimativa do desempenho é a média aritmética dos quatro ciclos externos, ou seja  $(92.11\% + 92.11\% + 91.89\% + 97.30\%)/4 = 93.35\%$ . Em comparação, a estimativa pelo  $k$ -fold convencional da figura 9.12 resultou na média  $(94.74\% + 92.11\% + 91.89\% + 97.30\%)/4 = 94.01\%$ , o que é ligeiramente mais otimista. Para obter este último resultado, foram exaustivamente testadas todas as combinações dos hiperparâmetros sempre para a mesma divisão  $k$ -fold. Chegou-se à profundidade 2 que deu esse melhor resultado 94.01%. O perigo de otimizar manualmente o hiperparâmetro é exatamente esse, escolhendo somente aquele valor (profundidade 2) do hiperparâmetro que exibiu o melhor resultado (acurácia estimada de 94.01%) no  $k$ -fold convencional. ■

### 9.6.3 Experimentos de Validação Cruzada com Dados Sintéticos

Neste experimento a validação cruzada convencional e a validação cruzada aninhada são comparadas, usando os dados sintéticos descritos na tabela 9.12. A grande vantagem de usar os dados sintéticos é o conhecimento dos limites teóricos de acurácia. Podemos gerar uma quantidade  $n$  de amostras para cada uma das duas classes, definidas pelas duas distribuições Gaussianas, definir um modelo de classificador, e avaliar esse classificador por um dos métodos estudados de estimativa de desempenho. Neste experimento, vamos somente comparar a validação cruzada  $k$ -fold convencional da seção 9.5.1 e a validação cruzada  $k$ -fold aninhada seção 9.6. O modelo de classificador a ser usado é a Análise Discriminativa Quadrática (QDA) da seção 9.6, com a função discriminativa da (eq. 7.26) na pág. 199. Como cada classe tem  $n$  padrões, a probabilidade a priori de cada classe é 50%, e pode ser ignorada no função discriminativa, resultando na Regra de Bayes de Classificação de Verossimilhança Máxima da (eq. 7.22) na pág. 197. O modelo QDA por natureza é ótimo, pois cada classe forma uma Gaussiana. A figura 9.14 mostra para cada um dos quatro conjunto de dados sintéticos da tabela 9.12 a evolução da acurácia estimada para o  $k$ -fold convencional e  $k$ -fold aninhada. As acurárias apresentadas são a média de dez repetições do algoritmo 18. Como conhecemos o limite teórico da acurácia, baseado na regra de Bayes, podemos relacionar as acurárias estimados com esse limite. O valores de desempenho estimados sugerem os seguintes fatos:

1. Para um número pequeno  $n$  de amostras, em um experimento de aprendizagem supervisionada, a estimativa em geral é otimista demais, independente do método de validação cruzada;
2. Para um número grande de amostras, técnicas simples e mais sofisticadas fornecem o mesmo resultado em termos assintóticos;
3. Para um número pequeno  $n$  de amostras, a validação aninhada retorna valores mais realistas do que métodos de validação cruzada convencional, como, por exemplo, o  $k$ -fold, pois ultrapassa o limite teórico de Bayes com menos frequência.

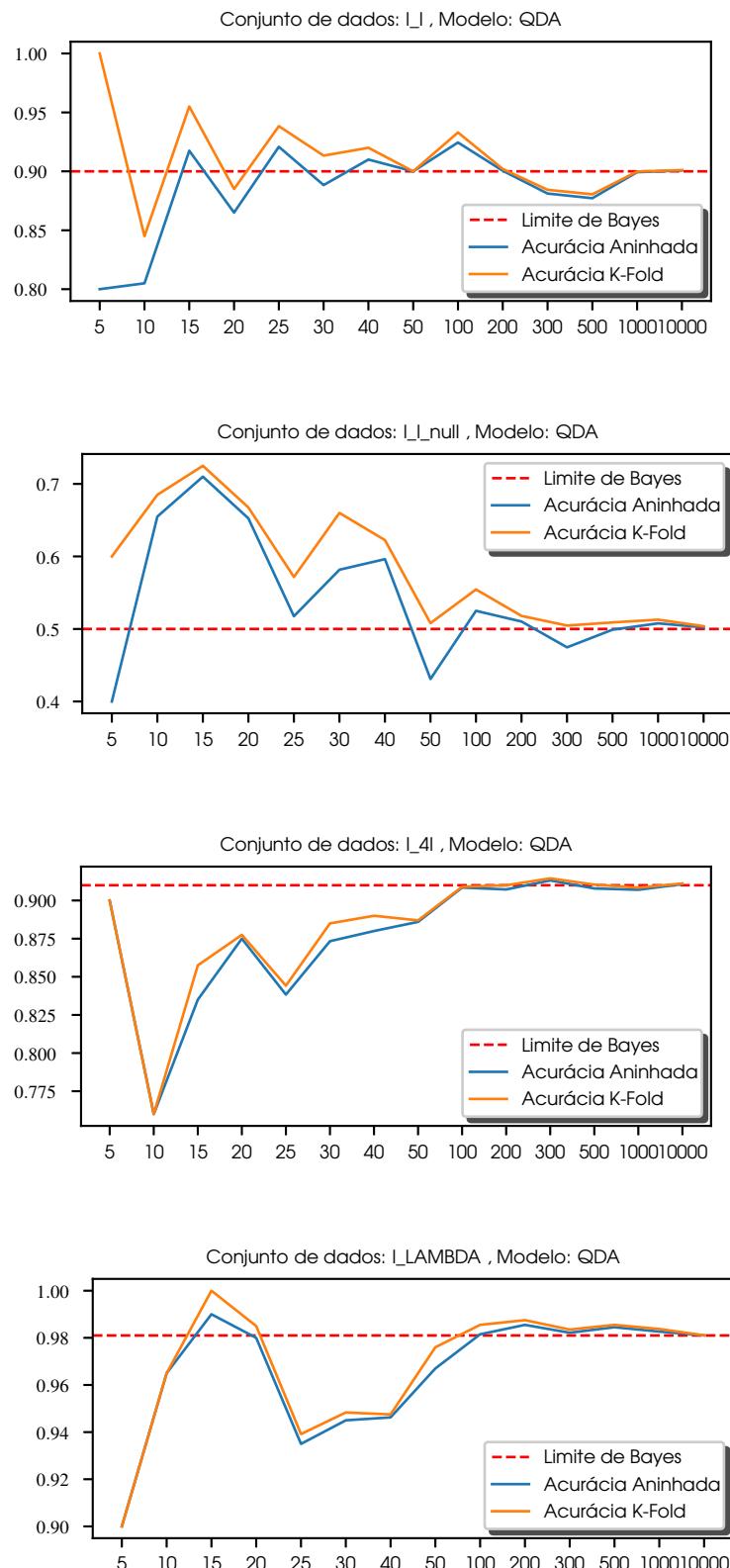


Figura 9.14: Comparação entre validação cruzada  $k$ -fold convencional da seção 9.5.1 e validação cruzada  $k$ -fold aninhada, descrita na seção 9.6. O eixo  $x$  é o número de padrões geradas por classe, e o eixo  $y$  é a acurácia estimada.



# 10. Seleção de Características

## 10.1 Introdução

Uma alerta importante no início deste assunto muito relevante na área de aprendizado de máquina. Quais e quantas características são usadas para treinar um modelo no fundo é um hiperparâmetro do modelo. Dessa maneira não podemos usar os mesmos dados para selecionar características, e posteriormente treinar o modelo com esses dados. Caso contrário, vamos caminhar novamente para um sobreajuste. Nas técnicas de validação cruzada, por exemplo,  $k$ -fold, vimos que está errado ajustar os hiperparâmetros com os mesmos folds que são usados para estimar o desempenho. Embora seja um erro óbvio, até pode-se afirmar que a maioria de publicações em conferências e revistas de alta qualificação não respeitam esse princípio, Citando [43], sec. 7.10.2:

Em geral, em um procedimento de modelagem de múltiplos estágios, a validação cruzada tem que ser aplicada à sequência completa dos passos de modelagem. Em particular, amostras têm que ser 'deixados fora' antes da aplicação de qualquer passos de seleção ou filtragem. ...

Enquanto este ponto pode parecer óbvio para o leitor, vimos esta tolice muitas vezes em publicações de revistas de altos estratos de qualidade.

A consequência dessa constatação é que não podemos considerar a seleção de características como um passo isolado que possa ser aplicado separadamente do modelo.

Vamos definir primeiro a ideia básica da seleção de características, elaborando a diferença entre *extração* de características e *seleção* de características. A técnica de extração de características já encontramos em vários exemplos ao longo deste texto, principalmente no capítulo 4. A descorrelação linear, respectivamente a Análise de Componentes Principais (PCA) é uma extração linear. o vetor de característica originais  $\mathbf{x}$  é transformado em um outro vetor de características  $\mathbf{y}$  através de um mapeamento linear como

$$\mathbf{y} = \Phi^T \mathbf{x},$$

veja a (eq. 4.33) na pág. 135. Repare que nenhuma das componentes  $y_j$  novas que compõem o vetor  $\mathbf{y}$  estão entre as características originais  $x_j$  que compõem o vetor  $\mathbf{x}$ , a não ser que o extrator

$\Phi$  seja a matriz de identidade. A mesma coisa acontece, considerando uma camada oculta de uma rede neural, veja a figura 4.5 na pág. 141. Cada componente calculada  $\phi$  em uma camada oculta constitui uma característica extraída nova, nunca vista antes no vetor de entrada original. A diferença fundamental entre PCA e a rede neural é a não linearidade da extração da rede. Em geral, a nova característica da rede é uma combinação da camada anterior  $\mathbf{x}$  com os pesos  $\mathbf{w}$  da camada, seguido pela passagem por uma função de ativação não linear  $z$  como

$$\phi = z(\mathbf{w} \cdot \mathbf{x}).$$

Lembre também do exemplo introdutório de extração de características, o índice de massa corporal do exemplo 4.1 na pág. 127, uma extração de características não linear. Duas características originais  $x_1$ , sendo o peso, e  $x_2$ , sendo a altura da pessoa, não se encontram no conjunto das características extraídas que neste caso é somente uma  $y_1 = x_1/x_2^2$ .

A natureza da *seleção* de características é tudo ou nada, ou uso uma característica  $x_j$  em um modelo, ou a descarto. No exemplo do índice da massa corporal, teríamos somente duas possibilidades, ou selecionamos o peso  $x_1$  ou a altura  $x_2$ . Já constatamos que nenhuma das duas por si só é apropriada para classificar bem entre as várias classes de pessoas. No caso das nossas flores, a seleção de características significa que em todos os gráficos bidimensionais que mostram somente uma das quatro características nos dois eixos, por exemplo, a figura 7.11 na pág. 218, fizemos uma seleção manual de duas de um total de quatro características. Duas foram usadas, as outras duas foram descartadas. Ao contrário da extração de características, nenhuma característica  $x_i$  foi combinada com outra e transformada em uma característica nova. Ou  $x_i$  é usado no modelo, ou não.

A figura 10.1 mostra a diferença entre seleção de características e extração de características. Na seleção, o conjunto<sup>1</sup> de  $D$  características

$$\mathcal{X} \stackrel{\text{def}}{=} \{x_1, \dots, x_D\} \quad (10.1)$$

é reduzido para um conjunto de  $d$  características

$$\mathcal{X}_d \stackrel{\text{def}}{=} \{x'_1, \dots, x'_d\}, \quad (10.2)$$

portanto o conjunto selecionado é um subconjunto do conjunto original de características

$$\mathcal{X}_d \subset \mathcal{X}.$$

Uma característica selecionada  $x'_j \in \mathcal{X}_d$  é uma das características do conjunto original, ou seja  $x'_j \in \mathcal{X}$ . Normalmente, a quantidade  $d$  das características selecionadas é bem menor que a cardinalidade  $D$  do conjunto original, ou seja

$$d \ll D.$$

■ **Exemplo 10.1** No caso das flores Iris, temos quatro características, ou seja o conjunto total das características disponíveis é

$$\mathcal{X} = \{x_1, x_2, x_3, x_4\}.$$

Podemos analisar exaustivamente todos os casos de seleção. Temos para a quantidade  $d$  de características selecionadas três possibilidades. Para *uma* característica selecionada temos quatro opções, então

$$d = 1 : \quad \mathcal{X}_1 = \{x_1\}, \quad \mathcal{X}_1 = \{x_2\}, \quad \mathcal{X}_1 = \{x_3\}, \quad \mathcal{X}_1 = \{x_4\}.$$

---

<sup>1</sup>Na verdade não estamos trabalhando com um vetor de valores, mas com um conjunto de características. Já falamos sobre essa simplificação logo no inicio, na seção 1.1.1 na pág. 10.

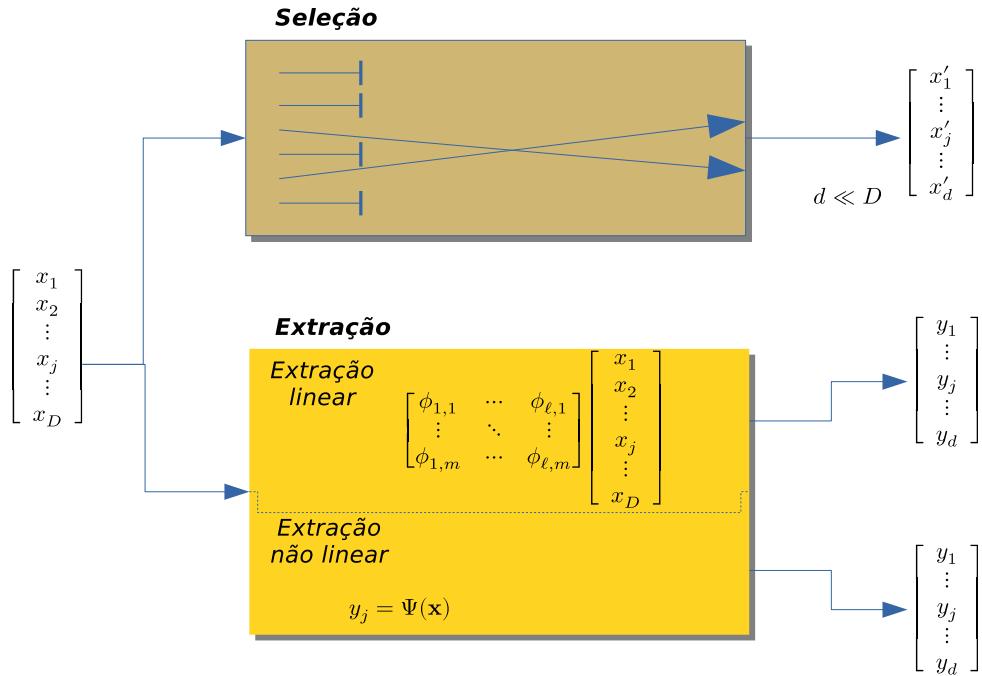


Figura 10.1: Diferença entre seleção de características e extração de características. Na seleção, o símbolo “ $\cancel{-}$ ” significa que uma característica pode ser descartada na seleção. O cruzamento das cetas na seleção significa que eventualmente uma ordem de qualidade da seleção possa surgir, ou seja, uma característica ou um conjunto pode ser mais apropriado em relação a um *critério de seleção*.

Para *duas* características selecionadas temos seis opções

$$d = 2 : \quad \mathcal{X}_2 = \{x_1, x_2\}, \quad \mathcal{X}_2 = \{x_1, x_3\}, \quad \mathcal{X}_2 = \{x_1, x_4\}, \\ \mathcal{X}_2 = \{x_2, x_3\}, \quad \mathcal{X}_2 = \{x_2, x_4\}, \quad \mathcal{X}_2 = \{x_3, x_4\}.$$

Finalmente, para *três* características selecionadas temos quatro opções

$$d = 3 : \quad \mathcal{X}_3 = \{x_1, x_2, x_3\}, \quad \mathcal{X}_3 = \{x_1, x_2, x_4\}, \\ \mathcal{X}_3 = \{x_1, x_3, x_4\}, \quad \mathcal{X}_3 = \{x_2, x_3, x_4\}.$$

Usando todas as  $d = 4$  características não seria um seleção, ou seria uma seleção trivial. Nenhuma característica selecionada também não é uma opção, portanto temos a restrição

$$0 < d < D.$$

■

Podemos constatar que a seleção exaustiva de  $d$  de  $D$  características pelas de regras de probabilidade corresponde à amostragem de  $d$  de  $D$  bolas de um urna, sem reposição, o que é calculado<sup>2</sup> pelo coeficiente binomial

$$\binom{D}{d} = \frac{D!}{d!(D-d)!}. \quad (10.3)$$

<sup>2</sup>Relembmando,  $n! = 1 \cdot 2 \cdot \dots \cdot (n-1) \cdot n$  é o factorial de  $n \in \mathbb{N}$ .

Para o exemplo da Iris temos

$$\binom{4}{1} = \frac{4!}{1!3!} = \frac{1 \cdot 2 \cdot 3 \cdot 4}{1 \cdot 1 \cdot 2 \cdot 3} = 4, \quad \binom{4}{2} = \frac{4!}{2!2!} = 6, \quad \binom{4}{3} = \frac{4!}{3!1!} = 4.$$

A busca exaustiva não faz sentido por duas razões. Primeiro, se a quantidade de características  $D$  for grande, a explosão combinatória inviabilizaria uma análise de qualquer possibilidade de seleção. Por exemplo selecionar dez de cem características, analisando todas as possibilidades requer 17310309456440 testes. Segundo, uma seleção tão “forçada”, como a exaustiva traria o perigo de sobreajuste (*overfitting*). Lembre que  $d$  e  $\mathcal{X}_d$  são hiperparâmetros do modelo, e, consequentemente a escolha desses hiperparâmetros tem que se submeter aos mesmos procedimentos discutidos na validação cruzada para estimar o desempenho do modelo. A busca exaustiva é um exemplo de uma estratégia de busca, um tópico extremamente importante na área de otimização. Porém, para entender melhor a natureza da seleção de características, vamos abordar brevemente como dizer se um conjunto de *candidatos* para seleção é melhor que outro conjunto de candidatos. Posteriormente o assunto do critério de seleção será aprofundado.

## 10.2 Critérios de Seleção

No exemplo das flores vimos que um conjunto de  $d$  características que são selecionados da totalidade de  $D$  características, pode variar na sua cardinalidade  $d$ , uma, duas, ou três. Então vamos definir um novo índice “k” que conta a quantidade de candidatos para seleção. O valor desse índice varia entre zero e a quantidade máxima de características inclusive, ou seja

$$0 < k \leq D.$$

Essa flexibilidade da quantidade de candidatos é importante, pois a busca tem que ser capaz de avaliar diferentes conjuntos de cardinalidades diferentes. Por exemplo, veremos uma busca sequencial que primeiro testa todos os conjuntos de cardinalidade um, depois alguns conjuntos de cardinalidade dois, até chegar no número desejado  $d$  de características. Consequentemente, temos que ter uma função que avalia a qualidade desses conjuntos de candidatos, independente da cardinalidade. Usaremos a letra “J” para expressar um escalar numérico que julga a qualidade de um conjunto de  $k$  candidatos.

Para distinguir bem os candidatos de todas as  $D$  características  $\mathcal{X}$  disponíveis, vamos modificar a nomenclatura. O conjunto total das  $D$  características é renomeado  $\mathcal{F}$  (“features”), e o conjunto de  $k$  características que estão atualmente no conjunto selecionado permanece como  $\mathcal{X}_k$ , porém, para simplificar, uma das características da seleção é renomeada de  $x'$  para  $x$ . Então temos

$$\mathcal{F} = \{f_1, \dots, f_i, \dots, f_D\} \quad \text{Conjunto de todas as } D \text{ características} \quad (10.4a)$$

$$\mathcal{X}_k = \{x_1, \dots, x_i, \dots, x_k\} \quad \text{Conjunto de } k \text{ características selecionadas,} \\ x_i \in \mathcal{F}, \quad i = 1, \dots, k, \quad \mathcal{X}_k \subset \mathcal{F}, \quad \mathcal{X}_D = \mathcal{F} \quad (10.4b)$$

Um candidato faz parte do conjunto  $\mathcal{F}$  de todos as características, mas ainda não faz parte da seleção  $\mathcal{X}_k$ , ou seja

$$f_i \in \mathcal{F} \setminus \mathcal{X}_k \quad \text{Candidato para inclusão na seleção} \quad (10.5)$$

O critério de seleção retorna um escalar que julga a qualidade do conjunto dos candidatos.

$$J(\mathcal{X}_k) \in \mathbb{R} \quad \text{Critério de Seleção} \quad (10.6)$$

Alguns critérios de seleção são capazes de somente avaliar uma única característica, ou seja, calcular

$$J(\mathcal{X}_1) = J(\{x_i\}), \quad x_i \in \mathcal{F}, \quad (10.7)$$

outros podem avaliar todos as cardinalidades  $1 \leq k \leq D$  possíveis para o conjunto de todos as  $D$  características disponíveis, ou seja, calcular

$$J(\mathcal{X}_k) = J(\{x_1, \dots, x_i, \dots, x_k\}), \quad x_i \in \mathcal{F}, \quad (10.8)$$

■ **Exemplo 10.2** No caso das flores, supondo que o critério pode ser calculado para todas as cardinalidades possíveis, temos então para as cardinalidades  $k = 1, 2, 3, 4$  a possibilidade de calcular o critério

$$\begin{aligned} k = 1 : & J(\{x_1\}), \dots, J(\{x_4\}) \\ k = 2 : & J(\{x_1, x_2\}), \dots, J(\{x_3, x_4\}) \\ k = 3 : & J(\{x_1, x_2, x_3\}), \dots, J(\{x_2, x_3, x_4\}) \\ k = 4 : & J(\{x_1, x_2, x_3, x_4\}) \end{aligned}$$

■

O critério de seleção guiará o processo de busca pelo melhor conjunto. Queremos achar portanto o conjunto  $\mathcal{B}_k$  (“B” = best) de  $k$  candidatos que é melhor que qualquer outro conjunto da mesma cardinalidade. Então

$$\mathcal{B}_k = \arg \max_{\mathcal{X}_k} J(\mathcal{X}_k). \quad (10.9)$$

O resultado final da seleção é o melhor conjunto que contém a quantidade desejada de  $d$  características, ou seja

$$\mathcal{B}_d = \arg \max_{\mathcal{X}_d} J(\mathcal{X}_d). \quad (10.10)$$

Uma ideia que o leitor deve ter é que aquele conjunto de características é um bom conjunto de características, se o modelo que usa esse conjunto tenha um desempenho bom. Logo uma estratégia plausível na seleção é dar preferência àquele conjunto que tenha para um certo número de candidatos o maior valor. Se a intenção é criar um classificador, a acurácia deste modelo seria um critério apropriado. Como não temos, em geral, como calcular a acurácia analiticamente, somos obrigados a usar as técnicas descritas no capítulo 9. Podemos, por exemplo, usar a validação  $k$ -fold para estimar a acurácia de uma Máquina Linear, se esse for o modelo.

■ **Exemplo 10.3** Vamos fazer experimentos de seleção de características com todas as cardinalidades possíveis das flores. Isto significa que vamos fazer uma busca exaustiva para uma, duas e três características, e para completar o conhecimento sobre o desempenho ainda vamos usar todas as quatro características. O modelo a ser usado para avaliar o desempenho seja o classificador 1-Vizinho-Mais-Próximo, a técnica de validação cruzada para estimar o critério de seleção seja 10-fold, e o próprio critério de seleção seja a acurácia. A tabela 10.1 mostra o resultado. Simplificando a escrita, vamos trocar ocasionalmente a designação da característica de  $\{x_i\}$  para simplesmente  $\{i\}$ . Se a nossa intenção for escolher uma única característica, a melhor das quatro, seria a quarta  $\{4\}$ , a largura da pétala, pois a acurácia média estimada de dez folds do classificador foi 91.33%. Essa acurácia estimada é melhor que 52.67%, correspondendo ao modelo que usa somente a característica  $\{1\}$ , ou somente a característica  $\{2\}$ . O classificador que só usa  $\{3\}$

Tabela 10.1: Seleção de características para todas as cardinalidades possíveis das quatro características do conjunto de flores Iris. O critério de seleção é a acurácia estimada de dez folds, usando o classificador 1-Vizinho-Mais-Próximo . O desvio padrão diz respeito às dez acurácias estimadas em cada ciclo. Usa-se a forma mais compacta para expressão a característica,  $j$  em vez de  $x_j$ .

Cardinalidade	Candidatos $\mathcal{Z}$	Acurácia estimada 10-fold [%]	Desvio padrão acurácia
1	{1}	52.67	0.1698
1	{2}	52.67	0.1281
1	{3}	88.67	0.0846
1	{4}	91.33	0.0670
2	{1, 2}	69.33	0.1123
2	{1, 3}	93.33	0.0516
2	{1, 4}	93.33	0.0516
2	{2, 3}	91.33	0.0600
2	{2, 4}	92.00	0.0653
2	{3, 4}	96.67	0.0333
3	{1, 2, 3}	94.00	0.0814
3	{1, 2, 4}	95.33	0.0427
3	{1, 3, 4}	95.33	0.0427
3	{2, 3, 4}	96.00	0.0327
4	{1, 2, 3, 4}	96.00	0.0533

conseguiu o segundo melhor resultado com 88.67%. Usando a cardinalidade dois, o conjunto  $\{3, 4\}$ , comprimento e largura da pétala, ganha com 96.67%, exatamente a situação mostrada na figura 1.2b na pág. 15. Além disso, com as cardinalidades três e quatro, não podemos atingir acurácias maiores. A conclusão deste experimento poderia ser que as características  $\{3, 4\}$ , são as mais discriminativas e  $\{1, 2\}$  são redundantes. Reduzimos a dimensionalidade do nosso modelo pela metade, pois a quantidade de características diminui de  $D = 4$  para  $d = 2$ , e, em consequência, a complexidade computacional do modelo de aprendizado de máquina reduz também. ■

Este pequeno experimento faz parte da categoria de um *wrapper*<sup>3</sup>, um método de seleção de características que treina um modelo, estima o desempenho e seleciona características simultaneamente. Os problemas do *wrapper* são o alto custo computacional, pois para cada conjunto de candidatos  $\mathcal{X}_k$  a serem selecionados, o modelo tem que ser treinado. Para modelos complexos com muitos parâmetros e hiperparâmetros isso é computacionalmente inviável. Além disso existe o perigo do sobreajuste, pois estamos forçando o modelo para um conjunto específico que talvez seja o melhor somente para um experimento particular. Neste ponto seja mais uma vez relembrado que não é permitido obter o resultado otimizado, por exemplo, da tabela 10.1, e depois os mesmos dados para treinar o modelo. De fato, nem precisamos mais fazer isso, pois o *wrapper* já nos forneceu a acurácia estimada.

Uma outra categoria de métodos de seleção são os *filtros*, *filters* em inglês. O modelo a ser usado é definido somente após da filtragem das características.

■ **Exemplo 10.4** Considere um problema de regressão, onde temos  $m$  características de entrada contínuas  $x_j$ ,  $j = 1, \dots, m$  (chamadas as variáveis explicativas, ou independentes, no contexto da regressão), e uma variável de saída  $y$  contínua (variável explicada, ou dependente). Vamos aplicar uma análise univariável quais são as características mais apropriadas para serem usadas na regressão. Isto significa que a relação de cada uma das características disponíveis com a característica de saída é analisada, ou seja, o par  $(x_j, y)$ . Note que a análise individual é uma simplificação que ignora as dependências entre as características, da mesma maneira como foi assumido no modelo do Naïve Bayes da seção 7.3 na pág. 200.

<sup>3</sup>“To wrap” em inglês embrulhar, ou seja, uma técnica que usa um modelo específico, aqui o K-Vizinhos-Mais-Próximos, para produzir o critério de seleção.

Obviamente características  $x$  que tenham uma “ligação” forte com a variável explicada  $y$  são melhores que características que não têm “pouco em comum” com a saída. Na seção 1.4.1 na pág. 25 já foi apresentada uma medida que expressa a interdependência entre duas variáveis aleatórias contínuas, a correlação na (eq. 1.41) na pág. 26

$$\rho_{xy} = \frac{\sigma_{xy}}{\sigma_x \sigma_y},$$

onde  $\sigma_{xy}$  é a covariância entre as variáveis, e  $\sigma_x$  e  $\sigma_y$  são os dois desvios padrões. Como primeira ideia poderíamos definir como critério de seleção, o módulo (ou quadrado) da correlação, dado o intervalo dos valores possíveis  $-1 \leq \rho_{xy} \leq 1$ , sendo -1 ou 1 o maior valor possível de correlação, e zero o menor valor. Um critério mais robusto é fazer um teste de hipóteses [109], uma ferramenta essencial na estatística para tomar decisões. Como informação adicional, esse teste, em geral produz um valor quantitativo que expressa quanto certo a decisão pode ser tomada.

Supondo que a *estatística de teste* segue uma distribuição  $t$  de Student, o valor da estatística e o *valor p* correspondente são critérios apropriados para julgar a interdependência entre característica e variável explicada. A estatística  $t$  para determinar sobre a independência entre  $x$  e  $y$  é definida [68] como

$$t = \hat{\rho} \sqrt{\frac{n-2}{1-\hat{\rho}^2}}, \quad (10.11)$$

onde  $n$  é a quantidade de amostras disponíveis, e  $\hat{\rho} = \widehat{\rho}_{xy}$  é a estimativa da correlação entre  $x$  e  $y$  obtida pelas mesmas amostras

$$\begin{aligned} \widehat{\rho}_{xy} &= \frac{\widehat{\sigma}_{xy}}{\widehat{\sigma}_x \widehat{\sigma}_y} = \frac{\frac{1}{n-1} \sum_{i=1}^n (x_i - \widehat{\mu}_x)(y_i - \widehat{\mu}_y)}{\left[ \frac{1}{n-1} \sum_{i=1}^n (x_i - \widehat{\mu}_x)^2 \right]^{\frac{1}{2}} \left[ \frac{1}{n-1} \sum_{i=1}^n (y_i - \widehat{\mu}_y)^2 \right]^{\frac{1}{2}}} \\ \widehat{\mu}_x &= \frac{1}{n} \sum_{i=1}^n x_i, \\ \widehat{\mu}_y &= \frac{1}{n} \sum_{i=1}^n y_i, \end{aligned} \quad (10.12)$$

onde  $\widehat{\mu}_x$  e  $\widehat{\mu}_y$  são as médias estimadas da variável dependente e independente, respectivamente, veja (eq. 1.27) na pág. 22 e (eq. 1.40) na pág. 26. O teste de hipóteses com a hipótese nula  $H_0$  e a hipótese alternativa  $H_1$  pode ser formulado assim

$$H_0 : \rho = 0 \quad (10.13)$$

$$H_1 : \rho \neq 0. \quad (10.14)$$

O valor  $p$  unicaudal da estatística expressa o grau de confiança que a hipótese é rejeitada. A rejeição da hipótese é equivalente com a afirmação que a correlação não é nula. Normalmente se usa um valor de  $p = 0.05$  para dizer que a rejeição é significante. Valores menores equivalem à certezas ainda maiores que a correlação é alta. A tabela 10.2 mostra o resultado de experimentos com a base de dados “Boston Housing” [5], [83]. O preço de imóveis, a variável explicada  $y$ , é descrita por 13 variáveis explicativas  $x_j$ ,  $j = 1, \dots, 13$ . O objetivo do experimento é estabelecer uma ordem para filtrar as variáveis mais importantes. Os três critérios usados são o valor absoluto  $|\rho_{xy}|$  da correlação, a estatística  $t$  da (eq. 10.11), e o valor  $p$ , usado para determinar a significância do valor da estatística. Nota-se que a ordem dessas três alternativas fornece o mesmo resultado. Se tivéssemos que escolher os melhores três desses 13 características, seria o conjunto  $\{13, 6, 11\}$ ,

Tabela 10.2: Resultados de dez repetições de experimentos de classificação com três modelos diferentes. Todos modelos treinados com conjunto de treinamento. Classificação do mesmo conjunto de treinamento e classificação do conjunto complementar de teste.

Variável	Acrônimo	$\rho_{xy}$	Ordem	$t$	Ordem	$p$	Ordem
1	CRIM	-0.388	7	9.460	7	5.870e-20	7
2	ZN	0.360	10	8.675	10	2.857e-17	10
3	INDUS	-0.484	4	12.408	4	2.450e-31	4
4	CHAS	0.175	13	3.996	13	3.695e-05	13
5	NOX	-0.427	6	10.611	6	3.533e-24	6
6	RM	0.695	2	21.722	2	1.244e-74	2
7	AGE	-0.377	9	9.137	9	7.850e-19	9
8	DIS	0.250	12	5.795	12	6.033e-09	12
9	RAD	-0.382	8	9.269	8	2.733e-19	8
10	TAX	-0.469	5	11.906	5	2.819e-29	5
11	PTRATIO	-0.508	3	13.233	3	8.048e-35	3
12	B	0.333	11	7.941	11	6.591e-15	11
13	LSTAT	-0.738	1	24.528	1	2.541e-88	1

o que corresponde às características “porcentagem de faixa baixa de população”, “quantidade média de cômodos por habitação”, e “relação professor–aluno por cidade”. Chama-se novamente atenção ao fato que a relação *entre* as características foi completamente ignorada nesta análise univariável. A categoria do *filtro* na seleção de características em geral pode também usar uma análise multivariável. ■

A terceira categoria de seletores de características são os métodos *embutidos*, *embedded*, em inglês, que incorporam a seleção de características dentro do processo de aprendizagem de um modelo específico. Já conhecemos uma seleção embutida, nomeadamente a escolha do atributo de um nó de uma árvore de decisão ou regressão. Enquanto a árvore está sendo construída, em cada um dos nós temos que selecionar uma característica, nomeadamente aquela que maximiza, para árvores de decisão, o critério do Ganho de Entropia (ou equivalentemente Informação Mútua, ou equivalente Divergência Kullback–Leibler) da seção 8.3.1 na pág. 234, ou o Índice Gini de Diversidade (Impuridade) da (eq. 8.36d) na pág. 240, ou o Critério de Bifurcação da definição 8.4.1 na pág. 248 para árvores de regressão. A *regularização* pode ser também caracterizada como um método embutido. Essa ferramenta do aprendizado de máquina muito importante será estudada posteriormente.

Cada uma das categorias das estratégias para selecionar características tem as suas vantagens e desvantagens. Um *wrapper* é computacionalmente muito caro e sujeito a sobreajuste (*overfitting*), especialmente no caso de ter poucas amostras para treinamento a disposição. Por outro lado, a interdependência entre as características é explorada melhor por um *wrapper*. O filtro é rápido, especialmente no caso da análise unidimensional entre uma única característica e a variável dependente. Também é menos flexível em relação à capacidade de se “curvar” aos dados, sendo assim menos sujeito ao sobreajuste. Para os métodos embutidos, uma afirmação geral é difícil, pois os métodos são muito diferentes em relação como a seleção de características é amarrada no treinamento do modelo.

### 10.3 Estratégias de Busca

Já constatamos que a busca pelo melhor subconjunto de características a partir de um conjunto de características disponíveis, tem que ser feito através de um algoritmo de busca. Eventualmente esse algoritmo é muito simples, e pode ser formulado em poucos comandos. O “melhor” no “melhor conjunto” é definido pelo critério de seleção que foi brevemente abordado. Após o estudo dos métodos de busca retornaremos ao assunto dos critérios de seleção.

#### 10.3.1 Busca Univariável: Melhores Características

Este é a técnica mais simples. Assume-se independência entre as  $D$  características que estão à disposição na escolha do melhor subconjunto de  $d$  características. O algoritmo simplesmente faz uma análise individual para cada característica, ou seja, analisa a relação da  $j$ -ésima característica  $x_j$  com a saída do modelo, por exemplo, em um problema de regressão com a variável dependente ( $y$  em caso de saída unidimensional, e  $\mathbf{y}$  em caso de saída multidimensional). No exemplo 10.1, para  $d = 1$  temos a análise das quatro características de Iris, e na tabela 10.1, para a cardinalidade 1, o erro estimado de um classificador K-Vizinhos-Mais-Próximos pode ser vista nas primeiras quatro linhas. Também na geração de uma árvore de classificação ou regressão aplicamos este tipo de busca unidimensional. Na algoritmo 15 na pág. 239, na linha 9, está definida a bifurcação da árvores de decisão, podemos observar um outro exemplo de seleção univariável. Para cada atributo disponível é validada a capacidade de agrupar as amostras restantes. O critério de seleção neste caso é o ganho de entropia, ou ganho de impuridade. O algoritmo 19 ilustra os passos necessários na análise univariável. A grande vantagem deste algoritmo é a sua complexidade linear, pois cada

---

#### Algoritmo 19: Busca Univariável: Melhores Características

---

**Entrada:** Conjunto de  $D$  características  $\mathcal{F} = \{f_1, \dots, f_D\}$ ;

**Resultado:** Conjunto de  $d$  características selecionadas  $\mathcal{B}_d = \{x_1, \dots, x_d\}$

- 1 **para**  $i = 1, \dots, D$  **fça**
  - 2   |  Calcule o critério de seleção  $J(\{f_i\})$  do candidato  $f_i \in \mathcal{F}$  ;
  - 3 **fim**
  - 4 Ordene os  $D$  candidatos para seleção segundo o critério, em ordem decrescente  

$$J(\{x_1\}) \geq \dots \geq J(\{x_i\}) \geq \dots \geq J(\{x_D\}), \quad x_i \in \mathcal{F};$$
  - 5 Escolhe os primeiros  $d$  características desta sequência ordenada  $\mathcal{B}_d \leftarrow \{x_1, \dots, x_d\}$ ;
- 

um dos candidatos é analisado individualmente uma única vez. Essa vantagem ainda aumenta, se a busca individual é aplicada na modalidade de um *filtro*. Um exemplo poderia ser em um problema de regressão, o cálculo do coeficiente de correlação, explicitado no exemplo 10.4. Obviamente a análise univariável não considera nenhuma relação entre as características, o que destrói informação no processo de treinamento do modelo. Por outro lado, a combinação de seleções univariáveis repetido no contexto de uma árvore ou floresta, parece introduzir implicitamente dependências multidimensionais, pois os desempenhos desses modelos frequentemente ficam entre os melhores. Um exemplo ilustrativo como “enganar” este tipo de busca unidimensional é simplesmente criar uma réplica da melhor característica. O algoritmo ia selecionar como segunda melhor característica essa cópia da melhor característica, embora nenhuma informação nova foi adicionada, pois o valor agregado para o modelo seria nulo.

#### 10.3.2 Seleção Sequencial Adiante, *Sequential Forward Selection*, SFS

Esta técnica é um bom compromisso entre complexidade de busca e análise multivariável. A ideia é incluir um novo candidato e avaliar esse candidato, *junto* com as características que já fazem parte do conjunto atualmente selecionado.

Vamos introduzir mais uns conceitos que ajudam a formalizar os algoritmos de busca sequencial. A *significância* [82, 97, 104] individual de uma única característica, ou de um conjunto de características são definidos como

**Definição 10.3.1 — Significância de uma única característica.**

$$S_0(\{f_i\}) \stackrel{\text{def}}{=} J(\{f_i\}) \quad (10.15)$$

**Definição 10.3.2 — Significância de uma característica  $y_i$  em conjunto com todas as características do conjunto  $\mathcal{Y}$ .**

$$S(y_i, \mathcal{Y}) \quad (10.16)$$

**Definição 10.3.3 — Significância na remoção da característica  $x_i$  da seleção  $\mathcal{X}_k$ .**

$$S^-(x_i, \mathcal{X}_k) \stackrel{\text{def}}{=} J(\mathcal{X}_k) - J(\mathcal{X}_k \setminus \{x_i\}), \quad x_i \in \mathcal{X}_k \quad (10.17)$$

**Definição 10.3.4 — Significância da inclusão do candidato  $f_i$  na seleção  $\mathcal{X}_k$ .**

$$S^+(f_i, \mathcal{X}_k) \stackrel{\text{def}}{=} J(\mathcal{X}_k \cup \{f_i\}) - J(\mathcal{X}_k), \quad f_i \in \mathcal{F} \setminus \mathcal{X}_k \quad (10.18)$$

Na definição 10.3.1, a significância individual é equivalente ao critério de seleção  $J$  de uma única característica, definido na (eq. 10.7), e usado no algoritmo 19. A definição 10.3.2 é uma moldura para considerar uma característica em relação a um conjunto existente, ou removendo, ou incluindo. A definição 10.3.3 mede o efeito de excluir alguma característica  $x_i$  de uma seleção existente  $\mathcal{X}_k$  de  $k$  características. O critério de seleção  $J$  é calculado antes e após da remoção. A remoção pode diminuir ou aumentar o critério. Pode acontecer, por exemplo, que a eliminação de uma característica redundante aumenta o poder discriminativo da seleção existente. Neste caso, a significância na remoção seria negativa. A análise do efeito da remoção de  $x_i$  permite calcular um valor numérico para uma determinada característica, e consequentemente estabelecer uma ordem. A remoção de uma característica “boa” vai provocar uma diferença positiva grande entre o critério da seleção atual. A remoção de uma característica pouco importante terá pouco efeito, ou seja uma significância positiva pequena, ou até negativa. Por outro lado, na inclusão, na definição 10.3.4, podemos medir o efeito para cada candidato, e assim também estabelecer uma ordem que permite identificar ótimos candidatos para a inclusão. Na inclusão de um único candidato  $f_i$  procura-se obviamente aquele candidato  $f^* \in \mathcal{F} \setminus \mathcal{X}_k$  que aumenta o critério de seleção ao máximo, ou seja, procura-se

$$f^* \stackrel{\text{def}}{=} \arg \max_{f_i} S^+(f_i, \mathcal{X}_k).$$

Usando a definição da significância, podemos facilmente formular a busca da Seleção Sequencial Adiante, SFS. O algoritmo 20 mostra a ideia básica.

Inicialmente nenhuma característica é selecionada, então na linha 2, a seleção  $\mathcal{X}_0$  contém nenhuma característica e o conjunto  $\mathcal{F}$  dos candidatos ainda disponíveis inicialmente para seleção, contém todas as  $D$  características. O laço principal sequencialmente junta  $d$  características, indivi-

**Algoritmo 20:** Busca Multivariável: SFS

---

**Entrada:** Conjunto de  $D$  características  $\mathcal{F} = \{f_1, \dots, f_D\}$ ;

**Resultado:** Conjunto de  $d$  características selecionadas  $\mathcal{B}_d = \{x_1, \dots, x_d\}$

```

1  $k \leftarrow 0$  /* Quantidade de características selecionados atualmente */ ;
2  $\mathcal{X}_k \leftarrow \emptyset$  /* Conjunto de candidatos selecionados atualmente */ ;
3 enquanto ( $k < d$ ) faz
4    $x_{k+1} \leftarrow \arg \max_{f_i \in \mathcal{F} \setminus \mathcal{X}_k} S^+(f_i, \mathcal{X}_k)$  /* candidato que maximiza significância de inclusão */ ;
5    $\mathcal{X}_{k+1} \leftarrow \mathcal{X}_k \cup \{x_{k+1}\}$  /* Inclua melhor candidato na seleção */ ;
6    $k \leftarrow k + 1$  /* próxima inclusão */ ;
7 fim
8  $\mathcal{B}_d \leftarrow \mathcal{X}_k$  /* Seleção final */ ;

```

---

dualmente ao conjunto das características selecionadas  $\mathcal{X}_k$ . Em cada iteração deste laço, o melhor candidato  $x_{k+1}$  ainda disponível em  $\mathcal{F} \setminus \mathcal{X}_k$  é determinado (linha 4), e assim implicitamente eliminado do conjunto dos candidatos  $\mathcal{F}$ , e incluído no conjunto da seleção  $\mathcal{X}_k$  (linha 6). A detecção do melhor candidato  $x_{k+1}$  para inclusão na linha 4 é feita em uma iteração sobre todos os  $D - k$  candidatos  $f_i$  ainda disponíveis em  $\mathcal{F} \setminus \mathcal{X}_k$ , calculado a significância de inclusão da definição 10.3.4. Finalmente, a seleção ótima  $\mathcal{B}_d$  equivale à última seleção atual na linha 8.

■ **Exemplo 10.5** Podemos reutilizar a seleção das características das flores do exemplo 10.3, aplicando a busca SFS do algoritmo 20. A simulação é apresentada passo a passo na tabela 10.3<sup>4</sup>. A busca termina com o conjunto selecionado das melhores três características  $\mathcal{B}_3 = \{4, 3, 2\}$ , segundo o método SFS. Obviamente, em caso de uso de outro critério de seleção e/ou outro modelo de classificador, o resultado em geral seria diferente. Na seleção da primeira característica, {4} ganha com uma significância de 91.3%, assumindo que a significância (e critério de seleção) de um conjunto vazio seja zero, ou seja,  $J(\emptyset) = 0$ . Na inclusão da segunda característica, {3} ganha com uma significância de 5.4%,

$$\begin{aligned} S^+(\{3\}, \mathcal{X}_1) &= J(\mathcal{X}_1 \cup \{3\}) - J(\mathcal{X}_1) \\ &= J(\{4\} \cup \{3\}) - J(\{4\}) = 96.7\% - 91.3\% = 5.4\%. \end{aligned}$$

Finalmente, na inclusão da terceira característica, {2} ganha com uma significância negativa de  $-0.7\%$ . É notável que neste ponto do algoritmo, o critério de seleção diminui, porém por causa da quantidade pré-definida de  $d = 3$  características, mais um passo tem que ser executado obrigatoriamente. ■

**10.3.3 Seleção Sequencial Para Trás, Sequential Backward Selection, SBS**

Um mecanismo de busca parecido com a SFS é a análise de um conjunto existente, e testar os candidatos restantes em relação a este conjunto. Porém desta vez, estamos *excluindo* os candidatos, em vez de incluindo, como no caso do SFS. A ideia principal é começar com todas as  $D$  características, considerá-los como o conjunto selecionado. Em cada passo sequencial, um único candidato é eliminado, nomeadamente aquele que resultou no melhor valor do critério de seleção com a sua ausência do conjunto selecionado, ou, equivalentemente com a maior significância de remoção da definição 10.3.3. Pode parecer contraditório que a eliminação de características melhora o desempenho, mas de fato, em geral seleção de características tem dois benefícios principais:

---

<sup>4</sup>Foi usada uma versão modificada do software MLxtend [87] para realizar os experimentos dos métodos SFS, SBS, SFFS, SBFS, e busca exaustiva.

Tabela 10.3: Simulação de dos comandos da seleção de características para o conjunto de flores Iris. Três das quatro características são selecionadas pela busca SFS, ou seja,  $d = 3$ ,  $D = 4$ . O critério de seleção é a acurácia estimada de dez folds, usando o classificador 1-Vizinho-Mais-Próximo

Comando no algoritmo 20	$k$	$\mathcal{X}_k$	$\mathcal{F} \setminus \mathcal{X}_k$	$i$	$f_i$	$J(\mathcal{X}_k \cup \{f_i\})$	$S^+(f_i, \mathcal{X}_k)$	$x_{k+1}$
linha 1	0	—	—	—	—	—	—	—
linha 2	0	$\emptyset$	{1, 2, 3, 4}	—	—	—	—	—
linha 4	0	$\emptyset$	{1, 2, 3, 4}	1	1	52.7%	52.7%	—
linha 4	0	$\emptyset$	{1, 2, 3, 4}	2	2	52.7%	52.7%	—
linha 4	0	$\emptyset$	{1, 2, 3, 4}	3	3	88.7%	88.7%	—
linha 4	0	$\emptyset$	{1, 2, 3, 4}	4	4	91.3%	91.3%	—
linha 4	0	$\emptyset$	{1, 2, 3, 4}	—	—	—	—	4
linha 5	0	{4}	{1, 2, 3}	—	—	—	—	4
linha 6	1	{4}	{1, 2, 3}	—	—	—	—	—
linha 4	1	{4}	{1, 2, 3}	1	1	92.7%	1.4%	—
linha 4	1	{4}	{1, 2, 3}	2	2	91.3%	0.0%	—
linha 4	1	{4}	{1, 2, 3}	3	3	96.7%	5.4%	—
linha 4	1	{4}	{1, 2, 3}	—	—	—	—	3
linha 5	1	{3,4}	{1, 2}	—	—	—	—	3
linha 6	2	{3,4}	{1, 2}	—	—	—	—	—
linha 4	2	{3,4}	{1, 2}	1	1	95.3%	-1.4%	—
linha 4	2	{3,4}	{1, 2}	2	2	96.0%	-0.7%	—
linha 4	2	{3,4}	{1, 2}	—	—	—	—	2
linha 5	2	{2,3,4}	{1}	—	—	—	—	2
linha 6	3	{2,3,4}	{1}	—	—	—	—	—
linha 8	—	$\mathcal{B}_3 \leftarrow \{2,3,4\}$	—	—	—	—	—	—

1. Redução da dimensionalidade da entrada no modelo de  $D$  para  $d$ ;
2. Melhoria no desempenho do modelo.

Se conseguirmos reduzir a quantidade de variáveis independentes na entrada do modelo, e simultaneamente ainda aumentar os indicadores de desempenho, a seleção cumpriu a sua missão.

---

**Algoritmo 21:** Busca Multivariável: SBS

---

**Entrada:** Conjunto de  $D$  características  $\mathcal{F} = \{f_1, \dots, f_D\}$ ;  
**Resultado:** Conjunto de  $d$  características selecionadas  $\mathcal{B}_d = \{x_1, \dots, x_d\}$

```

1  $k \leftarrow D$  /* Quantidade de características selecionados atualmente */ ;
2  $\mathcal{X}_k \leftarrow \mathcal{F}$  /* Conjunto de candidatos selecionados atualmente */ ;
3 enquanto ( $k > d$ ) faz
4    $x_{k-1} \leftarrow \arg \min_{x_i \in \mathcal{X}_k} S^-(x_i, \mathcal{X}_k)$  /* candidato que minimiza significância de remoção */ ;
5    $\mathcal{X}_{k-1} \leftarrow \mathcal{X}_k \setminus \{x_{k-1}\}$  /* Exclua pior candidato da seleção atual */ ;
6    $k \leftarrow k - 1$  /* próxima remoção */ ;
7 fim
8  $\mathcal{B}_d \leftarrow \mathcal{X}_k$  /* Seleção final */ ;

```

---

Novamente, uma simulação é apresentada passo a passo do algoritmo 21 na tabela 10.4. Desta vez, somente uma característica é selecionada para executar pelo menos três passos de eliminação, ou seja,  $d = 1$ . No início todas as quatro características fazem parte do conjunto selecionado. Na primeira eliminação, a característica 2 sai do conjunto, pois com 95.3%, o conjunto restante  $\{1, 3, 4\}$  mostra o melhor desempenho. Alternativamente, a característica 1 poderia sair com o mesmo resultado. Na segunda eliminação, a característica 1 sai com o conjunto restante  $\{3, 4\}$  exibindo o melhor desempenho com 96.7%. Esse conjunto também é selecionado no SFS, quando a *inclusão* de características é feita. Finalmente, a terceira eliminada é a característica 3. A busca termina com o conjunto selecionado da melhor característica  $\mathcal{B}_1 = \{4\}$ , segundo o método SBS.

Uma observação que vale a pena mencionar é a diferença do valor do critério de seleção, comparando SFS na tabela 10.3 e SBS na tabela 10.4 quando o conjunto de candidatos é composto unicamente pela característica  $\{4\}$ . Na busca SFS o critério tem o valor 91.3% e na busca SBS o critério tem o valor 90.0%. Estes valores diferentes se explicam pelo comportamento estocástico da divisão do conjunto de dados pela validação cruzada  $k$ -fold. No experimento, SFS e SBS são inicializados com a mesma semente 66649, porém a sequência dos números aleatórios é diferente quando um única característica é usada, devido ao momento diferente da geração do conjunto de candidatos. No SFS acontece logo no primeiro laço exterior, com  $k = 1$ , e no SBS somente na terceira iteração  $k = 3$  do laço exterior.

#### 10.3.4 Seleção Sequencial Adiante Flutuante, *Sequential Forward Floating Selection*, SFFS

SFFS acrescenta um característica por vez no conjunto selecionado, até atingir o número desejado  $d$  do total de  $D$ . É um fato que, uma vez incluído no conjunto selecionado, a característica nunca mais pode ser “expulso do clube”, mesmo se na seleção uma característica que ficou pior que a incluída, no futuro, agora com o conjunto selecionado atual, combinaria melhor. Imagine um total de dez características, ou seja,  $D = 10$ , e uma seleção de  $d = 5$ . Seja a sequencia após duas selecionadas  $\{7, 4\}$ . Na terceira seleção, segundo o critério de seleção, a característica 3 seria a melhor opção, então temos  $\{7, 4, 3\}$ . A segunda melhor opção seja a característica 1 que desta vez ficou fora da seleção. Após mais duas inclusões seja o conjunto final  $\{7, 4, 3, 9, 2\}$ . É perfeitamente possível que um conjunto contendo a característica 1, mas sem a característica 3, ou seja, por exemplo  $\{7, 8, 1, 4, 9\}$  tenha um desempenho superior do que o final selecionado pelo SFS, mas a eliminação da característica 3 não está prevista. Em [82, 104] foram propostas modificações

Tabela 10.4: Simulação de dos comandos da seleção de características para o conjunto de flores Iris. Uma das quatro características são selecionadas pela busca SBS, ou seja,  $d = 1, D = 4$ . O critério de seleção é a acurácia estimada de dez folds, usando o classificador 1-Vizinho-Mais-Próximo

Comando no algoritmo 21	$k$	$\mathcal{X}_k$	$i$	$x_i$	$J(\mathcal{X}_k \setminus \{x_i\})$	$S^-(x_i, \mathcal{X}_k)$	$x_{k-1}$
linha 1	4	—	—	—	—	—	—
linha 2	4	{1, 2, 3, 4}	—	—	96.0%	—	—
linha 4	4	{1, 2, 3, 4}	1	1	95.3%	0.7%	—
linha 4	4	{1, 2, 3, 4}	2	2	95.3%	0.7%	—
linha 4	4	{1, 2, 3, 4}	3	3	94.0%	2.0%	—
linha 4	4	{1, 2, 3, 4}	4	4	93.3%	2.7%	—
linha 4	4	{1, 2, 3, 4}	—	—	—	—	2
linha 5	4	{1, 3, 4}	—	—	—	—	2
linha 6	3	{1, 3, 4}	—	—	95.3%	—	—
linha 4	3	{1, 3, 4}	1	1	96.7%	-1.4%	—
linha 4	3	{1, 3, 4}	2	3	93.3%	2.0%	—
linha 4	3	{1, 3, 4}	3	4	93.3%	2.0%	—
linha 4	3	{1, 3, 4}	—	—	—	—	1
linha 5	3	{3, 4}	—	—	—	—	1
linha 6	2	{3, 4}	—	—	96.7%	—	—
linha 4	2	{3, 4}	1	3	90.0%	-6.7%	—
linha 4	2	{3, 4}	1	4	88.7%	-8.0%	—
linha 5	2	{3, 4}	—	—	—	—	4
linha 5	2	{3}	—	—	—	—	4
linha 6	1	{3}	—	—	90.0%	—	—
linha 8	—	$\mathcal{B}_1 \leftarrow \{3\}$	—	—	—	—	—

dos algoritmos sequenciais, que permitem uma revisão dos já incluídos (no caso do SFS), ou dos já eliminados (no caso do SBS). A ideia principal é inverter o sentido de inclusão ou exclusão, sempre que melhorar o desempenho, um movimento análogo ao “acordeão”, onde alternadamente entra e sai ar. Então após a inclusão de uma característica, temos que controlar, se uma eventual remoção de uma característica já incluída anteriormente pode melhorar o resultado. O mesmo vale no caminho inverso, após a remoção temos que sondar, se a inclusão de uma característica possa melhorar o resultado.

Apresentam-se aqui os algoritmos flutuantes que restringem a inclusão e exclusão a uma única característica, introduzidos em [82]. Posteriormente, em [104], a inclusão e exclusão foram generalizadas, de tal maneira que várias características podem ser incluídas ou removidas simultaneamente.

O algoritmo 22 mostra a inclusão sequencial adiante no sentido do algoritmo 20, porém desta vez, permitindo a remoção *condicional* de uma característica, caso melhore o critério de seleção. Até a inclusão das primeiras duas características, o método é idêntico ao SFS, até a linha 7. O laço principal (linha 8) inclui e remove, até atingir a quantidade desejada de  $d$  características selecionadas. Na linha 9 detecta-se o candidato com a maior significância da inclusão (eq. 10.18) e na linha 10, uma inclusão convencional do SFS desse melhor candidato é feita. A linha 11, a pior característica na seleção atual é detectada, usando o candidato com a maior significância na remoção (eq. 10.17). Se esse pior candidato *não* coincidir com a última inclusão da linha 10, a remoção condicional entra em ação pelo SBS, enquanto o critério de seleção melhorar pela remoção (linha 14). Essa remoção tem que parar também, se somente duas características permanecerem. Se o pior candidato foi o último incluído, não faz sentido removê-lo novamente. Então o algoritmo continua normalmente como SFS convencional. O conjunto selecionado durante a remoção fica armazenado em um conjunto auxiliar  $\hat{\mathcal{X}}_k$ , que é transferido para a seleção  $\mathcal{X}_k$  na linha 19 ou linha 21.

■ **Exemplo 10.6** Vamos comparar dois algoritmos de seleção sequencial, SFS do algoritmo 20 e SFFS do algoritmo 22. Como conjunto de dados usamos novamente a base de dados “Boston Housing” [5], [83], que descreve o preço de imóveis por 13 variáveis. A variável explicada é contínua, portanto, temos um problema de regressão múltipla, com uma única saída. Como critério de seleção, temos que usar um critério que aceite mais que um argumento para as variáveis independentes, por exemplo, não podemos usar a correlação, como no exemplo 10.4. Como critério de seleção, usamos o valor negativo do erro quadrático médio (EQM) tabela 9.11 na pág. 281

$$J(\mathcal{X}) = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 = \frac{1}{n} \sum_{i=1}^n (y_i - f(\mathbf{x}_i))^2,$$

onde para o modelo  $f(\mathbf{x})$  usaremos a Máquina Extrema de Aprendizado, *Extreme Learning Machine*, ELM, da definição 4.3.3 na pág. 142, na modalidade regressão. Para estimar o critério, uma validação cruzada  $k$ -fold com  $K = 10$  é usada<sup>5</sup>. Note que essa validação cruzada pode fornecer diferenças com o mesmo subconjunto de  $k$  características selecionadas  $\mathcal{X}_k$ , se for rodado mais que uma vez, pois a composição dos folds muda com cada nova estimativa do critério. Por exemplo, no SFFS, em algum instante, os candidatos são  $\mathcal{X}_3 = \{2, 4, 7\}$ , após a inclusão de  $\{4\}$ . Depois, o conjunto  $\mathcal{X}_3 = \{2, 4, 7\}$  é resultado da remoção de  $\{6\}$  de  $\mathcal{X}_4 = \{2, 4, 6, 7\}$ . Os folds antes e depois certamente são outros, assim provocando uma diferença no critério. Vamos selecionar então seis das 13 características, uma vez pelo SFS e depois pelo SFFS.

A sequência selecionada pelo SFS, algoritmo 20, não será detalhada. Ela é

$$\emptyset \rightarrow \{9\} \rightarrow \{9, 4\} \rightarrow \{9, 4, 10\} \rightarrow \{9, 4, 10, 6\} \rightarrow \{9, 4, 10, 6, 13\} \rightarrow \{9, 4, 10, 6, 13, 11\}.$$

---

<sup>5</sup>Usa-se aqui a letra “K” para denominar a quantidade de folds, para não confundir com o contador “k” das características selecionadas

**Algoritmo 22:** Busca Multivariável: SFFS

---

**Entrada :** Conjunto de  $D$  características  $\mathcal{F} = \{f_1, \dots, f_D\}$ ;

**Resultado:** Seleção de  $d$  características  $\mathcal{B}_d = \{x_1, \dots, x_d\}$

- 1  $k \leftarrow 0$  /\* Quantidade de características selecionados atualmente \*/ ;
- 2  $\mathcal{X}_k \leftarrow \emptyset$  /\* Conjunto de candidatos selecionados atualmente \*/ ;
- 3  $x_1 \leftarrow \arg \max_{f_i \in \mathcal{F}} S^+(f_i, \emptyset)$  /\* Melhor característica \*/;
- 4  $\mathcal{X}_1 \leftarrow \{x_1\}$  /\* Inclua candidato que maximiza significância de inclusão \*/ ;
- 5  $x_2 \leftarrow \arg \max_{f_i \in \mathcal{F} \setminus \{x_1\}} S^+(f_i, \{x_1\})$  ;
- 6  $\mathcal{X}_2 \leftarrow \{x_1\} \cup \{x_2\}$  /\* Inclua melhor candidato, exceto já selecionado \*/;
- 7  $k \leftarrow 2$  /\* Duas características é selecionadas pelo SFS \*/ ;
- 8 **enquanto** ( $k < d$ ) **faça**
- 9      $x_{k+1} \leftarrow \arg \max_{f_i \in \mathcal{F} \setminus \mathcal{X}_k} S^+(f_i, \mathcal{X}_k)$  /\* Melhor candidato \*/ ;
- 10     $\mathcal{X}_{k+1} \leftarrow \mathcal{X}_k \cup \{x_{k+1}\}$  /\* Inclua melhor candidato na seleção \*/ ;
- 11     $x_w \leftarrow \arg \min_{x_i \in \mathcal{X}_k} S^-(x_i, \hat{\mathcal{X}}_k)$  /\* Pior característica na seleção atual \*/ ;
- 12    /\* Remoção condicional \*/ ;
- 13    **se**  $x_w \neq x_{k+1}$  **então**
- 14       $\hat{\mathcal{X}}_k \leftarrow \mathcal{X}_k$  ;
- 15      **enquanto**  $J(\hat{\mathcal{X}}_{k+1} \setminus \{x_w\}) > J(\hat{\mathcal{X}}_k)$  E  $k > 2$  **faça**
- 16        /\* Exclua até critério de seleção não melhorar mais \*/ ;
- 17         $\hat{\mathcal{X}}_k \leftarrow \hat{\mathcal{X}}_{k+1} \setminus \{x_w\}$  ;
- 18         $x_w \leftarrow \arg \min_{x_i \in \hat{\mathcal{X}}_{k+1}} S^-(x_i, \hat{\mathcal{X}}_{k+1})$  /\* Pior característica na seleção \*/ ;
- 19         $k \leftarrow k - 1$  ;
- 20      **fim**
- 21       $\mathcal{X}_{k+1} \leftarrow \hat{\mathcal{X}}_{k+1}$  ;
- 22    **senão**
- 23      /\* Pior na seleção foi o recém-incluído: Aplique SFS comum \*/ ;
- 24       $\mathcal{X}_{k+1} \leftarrow \hat{\mathcal{X}}_{k+1}$  ;
- 25    **fim**
- 26     $k \leftarrow k + 1$  /\* próxima inclusão \*/ ;
- 27 **fim**
- 28  $\mathcal{B}_d \leftarrow \mathcal{X}_k$  /\* Seleção final \*/ ;

---

Na tabela 10.5, o SFFS, algoritmo 22, inicializa até  $k = 2$  identicamente ao SFS, então

$$\emptyset \rightarrow \{9\} \rightarrow \{9, 4\}$$

até linha **10**. No próximo passo, linha **11**, a pior característica da seleção atual é identificada como

$$x_w = \{9\}.$$

Como a quantidade de característica selecionadas não pode ser menor que duas,  $\{9\}$  fica na seleção. ■

O algoritmo SBFS funciona em analogia ao SFFS. Não é mais detalhado aqui, consulte as publicações originais [82, 104].

Tabela 10.5: Resumo dos passos mais importantes de uma simulação de seleção de características pelo algoritmo 22, SFFS. Problema de regressão com os dados do “Boston Housing”, selecionado seis de 13 características. O modelo é a ELM em modo regressão, o critério de seleção é o erro quadrático médio, e a validação cruzada do desempenho do modelo é 10-fold.

Comando no algoritmo 22	$k$	$\mathcal{X}_k$	$x_{k+1}$	$x_w$	$x_w \neq x_{k+1}$	$\hat{\mathcal{X}}_k$	$J(\mathcal{X}_k \setminus \{x_w\})$	$>$ $J(\mathcal{X}_{k-1})$
linha 2	0	$\emptyset$	—	—	—	—	—	—
linha 4	1	{8}	8	—	—	—	—	—
linha 6	1	{3,8}	3	—	—	—	—	—
linha 9	2	{3,8}	9	—	—	—	—	—
linha 10	2	{3,8,9}	9	—	—	—	—	—
linha 11	2	{3,8,9}	9	3	Verdad.	—	—	—
linha 13	2	{3,8,9}	9	3	Verdad.	{3,8,9}	—	—
linha 14	2	{3,8,9}	9	3	Verdad.	{3,8,9}	Falso	—
linha 23	3	{3,8,9}	—	—	—	—	—	—
linha 9	3	{3,8,9}	5	—	—	—	—	—
linha 10	3	{3,5,8,9}	5	—	—	—	—	—
linha 11	3	{3,5,8,9}	5	8	Verdad.	—	—	—
linha 13	3	{3,5,8,9}	5	8	Verdad.	{3,5,8,9}	—	—
linha 14	3	{3,5,8,9}	5	8	Verdad.	{3,5,8,9}	Verdad.	—
linha 15	3	—	—	—	—	{3,5,9}	—	—
linha 16	3	—	3	—	—	{3,5,9}	—	—
linha 17	2	—	—	—	—	{3,5,9}	—	—
linha 19	2	{3,5,9}	—	—	—	{3,5,9}	—	—
linha 23	3	{3,5,9}	—	—	—	—	—	—
linha 9	3	{3,5,9}	8	—	—	—	—	—
linha 10	3	{3,5,8,9}	8	—	—	—	—	—
linha 11	3	{3,5,8,9}	8	3	Verdad.	—	—	—
linha 13	3	{3,5,8,9}	8	3	Verdad.	{3,5,8,9}	—	—
linha 14	3	{3,5,8,9}	8	3	Verdad.	{3,5,8,9}	Falso	—
linha 23	4	{3,5,8,9}	—	—	—	—	—	—
linha 9	4	{3,5,8,9}	4	—	—	—	—	—
linha 10	4	{3,4,5,8,9}	4	—	—	—	—	—
linha 11	4	{3,4,5,8,9}	4	3	Verdad.	—	—	—
linha 13	4	{3,4,5,8,9}	4	3	Verdad.	{3,4,5,8,9}	—	—
linha 14	4	{3,4,5,8,9}	4	3	Verdad.	{3,4,5,8,9}	Falso	—
linha 23	5	{3,4,5,8,9}	—	—	—	—	—	—
linha 9	5	{3,4,5,8,9}	10	—	—	—	—	—
linha 10	5	{3,4,5,8,9,10}	10	—	—	—	—	—
linha 11	5	{3,4,5,8,9,10}	10	4	Verdad.	—	—	—
linha 13	5	{3,4,5,8,9,10}	10	4	Verdad.	{3,4,5,8,9,10}	—	—
linha 14	5	{3,4,5,8,9,10}	10	4	Verdad.	{3,4,5,8,9,10}	Falso	—
linha 23	6	{3,4,5,8,9,10}	—	—	—	—	—	—
linha 25	—	$\mathcal{B}_6 \leftarrow \{3,4,5,8,9,10\}$	—	—	—	—	—	—

## **11. Regularização**





## 12. Ferramentas de Análise Estatística e Visual

12.1 Testes de Hipóteses

12.2 Testes Estatísticos Comparativos entre Modelos

12.2.1 Teste de Friedman

12.2.2 Teste de Nemenyi

12.3 Visualização de Dados

12.3.1 Gráfico de Sammon

12.3.2 Embutido Estocástico de Distribuição t-Student de Vizinhos, *T-distributed Stochastic Neighbor Embedding*, t-SNE



## **13. To Do**

Boosting

Gradient Boosting

MLP

RBM

Não supervisionado



## Bibliografia

- [1] Algoritmo de autovalores de Jacobi. *Algoritmo de autovalores de Jacobi — Wikipédia, a encyclopédia livre*. [Online; accessed 29-September-2020]. 2020. URL: [https://en.wikipedia.org/wiki/Algoritmo\\_de\\_autovalores\\_de\\_Jacobi](https://en.wikipedia.org/wiki/Algoritmo_de_autovalores_de_Jacobi) (ver página 132).
- [2] Daniel J Amit, Hanoch Gutfreund e Haim Sompolinsky. “Spin-glass models of neural networks”. Em: *Physical Review A* 32.2 (1985), página 1007 (ver página 183).
- [3] Autovalores e autovetores. *Autovalores e autovetores — Wikipédia, a encyclopédia livre*. [Online; accessed 29-September-2020]. 2020. URL: [https://pt.wikipedia.org/wiki/Autovalores\\_e\\_autovetores](https://pt.wikipedia.org/wiki/Autovalores_e_autovetores) (ver página 131).
- [4] Jayanta Basak. “A least square kernel machine with box constraints”. Em: *ICPR*. 2008, páginas 1–4 (ver página 227).
- [5] David A Belsley, Edwin Kuh e Roy E Welsch. *Regression diagnostics: Identifying influential data and sources of collinearity*. Volume 571. John Wiley & Sons, 2005 (ver páginas 305, 313).
- [6] Dimitri P Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014 (ver página 130).
- [7] Bessel's correction. *Bessel's correction — Wikipédia, A encyclopédia livre*. [Online; accessed 29-September-2020]. 2020. URL: [https://en.m.wikipedia.org/wiki/Bessel%27s\\_correction](https://en.m.wikipedia.org/wiki/Bessel%27s_correction) (ver página 22).
- [8] Bias of an estimator. *Bias of an estimator — Wikipédia, A encyclopédia livre*. [Online; accessed 29-September-2020]. 2020. URL: [https://en.wikipedia.org/wiki/Bias\\_of\\_an\\_estimator](https://en.wikipedia.org/wiki/Bias_of_an_estimator) (ver página 22).
- [9] Christopher M Bishop et al. *Neural networks for pattern recognition*. Oxford university press, 1995 (ver páginas 119, 123, 204).
- [10] Christopher M Bishop. *Pattern recognition and machine learning*. Springer, 2006 (ver páginas 119, 123, 163, 204, 211–213).

- [11] L. Breiman et al. *Classification and Regression Trees*. The Wadsworth and Brooks-Cole statistics-probability series. Taylor & Francis, 1984. ISBN: 9780412048418 (ver páginas 240, 246).
- [12] R. Burian e A.C. de Lima. *Cálculo numérico*. Fundamentos de informática. LTC, 2007. ISBN: 9788521615620 (ver páginas 34, 47).
- [13] C. Cortes e V. Vapnik. “Support-vector network”. Em: *Machine Learning* 20 (1995) (ver página 229).
- [14] Carl Gustav Jacob Jacobi. *Carl Gustav Jacob Jacobi — Wikipedia, The Free Encyclopedia*. [Online; accessed 29-September-2020]. 2020. URL: [https://en.wikipedia.org/wiki/Carl\\_Gustav\\_Jacobi](https://en.wikipedia.org/wiki/Carl_Gustav_Jacobi) (ver página 98).
- [15] Gavin C Cawley e Nicola LC Talbot. “On over-fitting in model selection and subsequent selection bias in performance evaluation”. Em: *The Journal of Machine Learning Research* 11 (2010), páginas 2079–2107 (ver página 292).
- [16] Confusion Matrix. *Confusion matrix — Wikipédia, a encyclopédia livre*. [Online; accessed 29-September-2020]. 2020. URL: [https://en.wikipedia.org/wiki/Confusion\\_matrix](https://en.wikipedia.org/wiki/Confusion_matrix) (ver página 268).
- [17] Coordenadas homogêneas. *Coordenadas homogêneas — Wikipédia, a encyclopédia livre*. [Online; accessed 29-September-2020]. 2020. URL: [https://pt.wikipedia.org/wiki/Coordenadas\\_homog%C3%A9neas](https://pt.wikipedia.org/wiki/Coordenadas_homog%C3%A9neas) (ver página 73).
- [18] T. Cover e P. Hart. “Nearest neighbor pattern classification”. Em: *Information Theory, IEEE Transactions on* 13.1 (jan. de 1967), páginas 21–27. ISSN: 0018-9448. DOI: 10.1109/tit.1967.1053997 (ver páginas 219, 223).
- [19] Thomas M Cover. “Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition”. Em: *IEEE transactions on electronic computers* 3 (1965), páginas 326–334 (ver página 145).
- [20] Balázs Csanád Csáji et al. “Approximation with artificial neural networks”. Em: *Faculty of Sciences, Eötvös Loránd University, Hungary* 24.48 (2001), página 7 (ver página 151).
- [21] Jesse Davis e Mark Goadrich. “The relationship between Precision-Recall and ROC curves”. Em: *Proceedings of the 23rd international conference on Machine learning*. 2006, páginas 233–240 (ver página 274).
- [22] Marc Peter Deisenroth, A Aldo Faisal e Cheng Soon Ong. *Mathematics for machine learning*. Cambridge University Press, 2020 (ver página 213).
- [23] Arthur P Dempster, Nan M Laird e Donald B Rubin. “Maximum likelihood from incomplete data via the EM algorithm”. Em: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1 (1977), páginas 1–22 (ver página 213).
- [24] Derivadas com Vetores. *Derivadas com Vetores — Wikipedia, The Free Encyclopedia*. [Online; accessed 29-September-2020]. 2020. URL: [https://en.wikipedia.org/wiki/Matrix\\_calculus#Derivadas\\_com\\_vetores](https://en.wikipedia.org/wiki/Matrix_calculus#Derivadas_com_vetores) (ver página 48).
- [25] Luc Devroye. “On the inequality of Cover and Hart in nearest neighbor discrimination”. Em: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 1 (1981), páginas 75–78 (ver página 223).
- [26] Timothy Dozat. “Incorporating nesterov momentum into adam”. Em: (2016) (ver página 115).
- [27] Harris Drucker et al. “Support vector regression machines”. Em: *Advances in neural information processing systems*. 1997, páginas 155–161 (ver página 39).

- [28] John Duchi, Elad Hazan e Yoram Singer. “Adaptive subgradient methods for online learning and stochastic optimization”. Em: *Journal of Machine Learning Research* 12.Jul (2011), páginas 2121–2159 (ver página 113).
- [29] Richard O Duda, Peter E Hart e David G Stork. *Pattern classification*. John Wiley & Sons, 2012 (ver página 290).
- [30] Efeito borboleta. *Sistema de coordenadas cartesiano — Wikipédia, A encyclopédia libre*. [Online; accessed 29-September-2020]. 2020. URL: [https://pt.wikipedia.org/wiki/Efeito\\_borboleta](https://pt.wikipedia.org/wiki/Efeito_borboleta) (ver página 16).
- [31] Bradley Efron. “Bootstrap methods: another look at the jackknife”. Em: *Breakthroughs in statistics*. Springer, 1992, páginas 569–593 (ver página 287).
- [32] Bradley Efron e Robert Tibshirani. “Improvements on cross-validation: the 632+ bootstrap method”. Em: *Journal of the American Statistical Association* 92.438 (1997), páginas 548–560 (ver página 287).
- [33] Bradley Efron e Robert J Tibshirani. *An introduction to the bootstrap*. CRC press, 1994 (ver página 287).
- [34] Fatoração de Cholesky. *Fatoração de Cholesky — Wikipédia, a encyclopédia libre*. [Online; accessed 29-September-2020]. 2020. URL: [https://pt.wikipedia.org/wiki/Fatora%C3%A7%C3%A3o\\_de\\_Cholesky](https://pt.wikipedia.org/wiki/Fatora%C3%A7%C3%A3o_de_Cholesky) (ver página 99).
- [35] Tom Fawcett. “An introduction to ROC analysis”. Em: *Pattern recognition letters* 27.8 (2006), páginas 861–874 (ver páginas 268, 275–277).
- [36] Ronald A Fisher. “The use of multiple measurements in taxonomic problems”. Em: *Annals of eugenics* 7.2 (1936), páginas 179–188 (ver página 9).
- [37] Roger Fletcher. “An optimal positive definite update for sparse Hessian matrices”. Em: *SIAM Journal on Optimization* 5.1 (1995), páginas 192–218 (ver página 110).
- [38] Floyd–Steinberg dithering. *Floyd–Steinberg dithering — Wikipédia, a encyclopédia libre*. [Online; accessed 29-September-2020]. 2021. URL: [https://en.wikipedia.org/wiki/Floyd%E2%80%93Steinberg\\_dithering](https://en.wikipedia.org/wiki/Floyd%E2%80%93Steinberg_dithering) (ver página 180).
- [39] filho Frederico Ferreira Campos. *Algoritmos Numéricos - Uma Abordagem Moderna de Cálculo Numérico*. Fundamentos de informática. LTC, 2018. ISBN: 9788521635550 (ver páginas 34, 47).
- [40] Funções de ativação. *Activation functions — Wikipédia, a encyclopédia libre*. [Online; accessed 29-September-2020]. 2020. URL: [https://en.wikipedia.org/wiki/Activation\\_function](https://en.wikipedia.org/wiki/Activation_function) (ver página 171).
- [41] Xavier Glorot e Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks”. Em: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop e Conference Proceedings. 2010, páginas 249–256 (ver página 155).
- [42] Ian Goodfellow, Yoshua Bengio e Aaron Courville. *Deep learning*. MIT press, 2016 (ver página 111).
- [43] Trevor Hastie, Robert Tibshirani e Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Science & Business Media, 2009. URL: <http://www-stat.stanford.edu/~hastie/Papers/ESLII.pdf> (ver páginas 35, 143, 254, 287, 289, 299).
- [44] Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice–Hall, Inc., 2007 (ver página 151).

- [45] Simon Haykin. *Redes neurais: princípios e prática*. Bookman Editora, 2007 (ver página 151).
- [46] D.O. Hebb. *The Organisation of Behaviour: A Neuropsychological Theory*. London, 1949. URL: [https://books.google.de/books?id=%5C\\_IekzQEACAAJ](https://books.google.de/books?id=%5C_IekzQEACAAJ) (ver página 181).
- [47] Donald Olding Hebb. *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005 (ver página 181).
- [48] John Hertz et al. “Introduction to the theory of neural computation”. Em: *PhT* 44.12 (1991), página 70 (ver páginas 123, 173, 181, 182).
- [49] Magnus Rudolph Hestenes e Eduard Stiefel. *Methods of conjugate gradients for solving linear systems*. Volume 49. 1. NBS Washington, DC, 1952 (ver páginas 105, 110).
- [50] Geoffrey Hinton, Nitish Srivastava e Kevin Swersky. “Neural networks for machine learning”. Em: *Coursera, video lectures* 264 (2012) (ver página 114).
- [51] Arthur E Hoerl e Robert W Kennard. “Ridge regression: Biased estimation for nonorthogonal problems”. Em: *Technometrics* 12.1 (1970), páginas 55–67 (ver página 143).
- [52] Thomas Hofmann, Bernhard Schölkopf e Alexander J. Smola. “Kernel Methods in Machine Learning”. Em: *Annals of Statistics* 36 (2008), páginas 1171–1220 (ver página 227).
- [53] John J Hopfield. “Neural networks and physical systems with emergent collective computational abilities”. Em: *Proceedings of the national academy of sciences* 79.8 (1982), páginas 2554–2558 (ver página 173).
- [54] Gary B Huang et al. “Labeled faces in the wild: A database for studying face recognition in unconstrained environments”. Em: 2008 (ver página 137).
- [55] Guang-Bin Huang, Dian Hui Wang e Yuan Lan. “Extreme learning machines: a survey”. Em: *International journal of machine learning and cybernetics* 2.2 (2011), páginas 107–122 (ver página 145).
- [56] Guang-Bin Huang, Qin-Yu Zhu e Chee-Kheong Siew. “Extreme learning machine: a new learning scheme of feedforward neural networks”. Em: *2004 IEEE international joint conference on neural networks (IEEE Cat. No. 04CH37541)*. Volume 2. IEEE. 2004, páginas 985–990 (ver página 138).
- [57] Guang-Bin Huang, Qin-Yu Zhu e Chee-Kheong Siew. “Extreme learning machine: theory and applications”. Em: *Neurocomputing* 70.1-3 (2006), páginas 489–501 (ver página 138).
- [58] I. T. Jolliffe. *Principal Component Analysis, Second Edition*. Springer, 2002 (ver páginas 135, 143).
- [59] Índice de massa corporal. *Índice de massa corporal — Wikipédia, a encyclopédia livre*. [Online; accessed 29-September-2020]. 2020. URL: [https://pt.wikipedia.org/wiki/%C3%8Dndice\\_de\\_massa\\_corporal](https://pt.wikipedia.org/wiki/%C3%8Dndice_de_massa_corporal) (ver página 125).
- [60] Indução Matemática. *Indução Matemática — Wikipedia, The Free Encyclopedia*. [Online; accessed 29-September-2020]. 2020. URL: [https://pt.wikipedia.org/wiki/Indu%C3%A7%C3%A3o\\_matem%C3%A1tica](https://pt.wikipedia.org/wiki/Indu%C3%A7%C3%A3o_matem%C3%A1tica) (ver página 106).
- [61] Iris flower data set. *Iris flower data set — Wikipedia, The Free Encyclopedia*. [Online; accessed 29-September-2020]. 2020. URL: [https://en.wikipedia.org/wiki/Iris\\_flower\\_data\\_set](https://en.wikipedia.org/wiki/Iris_flower_data_set) (ver páginas 9, 11).
- [62] J.E. Jackson. *A User’s Guide to Principal Components*. Wiley Series in Probability and Statistics. Wiley, 2005. ISBN: 9780471725329. URL: <https://books.google.com.br/books?id=f9s6g6c> (ver páginas 57, 58, 256).

- [63] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. 2nd. San Diego: Academic Press, 1990 (ver páginas 135, 290, 291).
- [64] Diederik P Kingma e Jimmy Ba. “Adam: A method for stochastic optimization”. Em: *arXiv preprint arXiv:1412.6980* (2014) (ver página 114).
- [65] Teuvo Kohonen. “Self-organized formation of topologically correct feature maps”. Em: *Biological cybernetics* 43.1 (1982), páginas 59–69 (ver página 117).
- [66] Teuvo Kohonen. “The self-organizing map”. Em: *Proceedings of the IEEE* 78.9 (1990), páginas 1464–1480 (ver página 117).
- [67] L. Breiman. “Bagging Predictors”. Em: *Machine Learning* 24(2) (1996), páginas 123–140 (ver páginas 252, 253).
- [68] Chap T Le e Lynn E Eberly. *Introductory biostatistics*. John Wiley & Sons, 2016 (ver página 305).
- [69] M. Aizerman, E. Braverman e L. Rozonoer. “Extrapolative Problems in Automatic Control and the Method of Potential Functions”. Em: *Am. Math. Soc. Transl.* 87 (1970), páginas 281–303 (ver página 225).
- [70] Prasanta Chandra Mahalanobis. “On the generalized distance in statistics”. Em: National Institute of Science of India. 1936 (ver página 28).
- [71] GJ McLachlan e T Krishnan. *The EM algorithm and Extensions*. Wiley, 1997 (ver página 213).
- [72] Método de Newton–Raphson. *Método de Newton–Raphson — Wikipédia, a encyclopédia livre*. [Online; accessed 29-September-2020]. 2020. URL: [https://pt.wikipedia.org/wiki/M%C3%A9todo\\_de\\_Newton-Raphson](https://pt.wikipedia.org/wiki/M%C3%A9todo_de_Newton-Raphson) (ver página 84).
- [73] John Moody e Christian J Darken. “Fast learning in networks of locally-tuned processing units”. Em: *Neural computation* 1.2 (1989), páginas 281–294 (ver página 146).
- [74] James D.B. Nelson et al. “A signal theory approach to support vector classification: The sinc kernel”. Em: *Neural Networks* 22.1 (2009), páginas 49–57. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2008.09.016. URL: <http://www.sciencedirect.com/science/article/pii/S089360800800161> (ver página 227).
- [75] Y Nesterov. “A method of solving a convex programming problem with convergence rate  $O(\frac{1}{k^2}) O(1/k)$ ”. Em: *Soviet Math. Dokl.* Volume 27. 1983 (ver página 112).
- [76] Yurii Nesterov. “Introductory lectures on convex programming volume i: Basic course”. Em: *Lecture notes* 3.4 (1998), página 5 (ver página 112).
- [77] Yurii Nesterov. *Lectures on convex optimization*. Volume 137. Springer, 2018 (ver página 112).
- [78] Jorge Nocedal e Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006 (ver páginas 46, 105, 106, 110, 111, 130).
- [79] Albert B Novikoff. *On convergence proofs for perceptrons*. Relatório técnico. STANFORD RESEARCH INST MENLO PARK CA, 1963 (ver página 123).
- [80] Elijah Polak e Gerard Ribiere. “Note sur la convergence de méthodes de directions conjuguées”. Em: *ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique* 3.R1 (1969), páginas 35–43 (ver página 110).
- [81] Produto de Kronecker. *Produto de Kronecker — Wikipedia, The Free Encyclopedia*. [Online; accessed 29-September-2020]. 2020. URL: [https://en.wikipedia.org/wiki/Kronecker\\_product](https://en.wikipedia.org/wiki/Kronecker_product) (ver página 94).

- [82] Pavel Pudil, Jana Novovičová e Josef Kittler. “Floating search methods in feature selection”. Em: *Pattern recognition letters* 15.11 (1994), páginas 1119–1125 (ver páginas 308, 311, 313, 315).
- [83] J Ross Quinlan. “Combining instance-based and model-based learning”. Em: *Proceedings of the tenth international conference on machine learning*. 1993, páginas 236–243 (ver páginas 305, 313).
- [84] J Ross Quinlan. *C4.5: programs for machine learning*. Elsevier, 2014 (ver página 231).
- [85] J. Ross Quinlan. “Induction of decision trees”. Em: *Machine learning* 1.1 (1986), páginas 81–106 (ver páginas 230, 231, 238).
- [86] R. O. Duda, P. E. Hart e D. G. Stork. *Pattern Classification*. 2nd. New York: John Wiley e Sons, 2001 (ver páginas 123, 192, 199).
- [87] Sebastian Raschka. “MLxtend: Providing machine learning and data science utilities and extensions to Python’s scientific computing stack”. Em: *The Journal of Open Source Software* 3.24 (abr. de 2018). DOI: 10.21105/joss.00638. URL: <http://joss.theoj.org/papers/10.21105/joss.00638> (ver página 309).
- [88] Regularização de Tikhonov. *Tikhonov regularization — Wikipédia, a encyclopédia livre*. [Online; accessed 29-September-2020]. 2020. URL: [https://en.wikipedia.org/wiki/Tikhonov\\_regularization](https://en.wikipedia.org/wiki/Tikhonov_regularization) (ver página 143).
- [89] J.L. Rojo-Alvarez et al. *Digital Signal Processing with Kernel Methods*. Wiley - IEEE. Wiley, 2018. ISBN: 9781118611791. URL: <https://books.google.com.br/books?id=-41FDwAAQBAJ> (ver página 227).
- [90] Frank Rosenblatt. “The perceptron: a probabilistic model for information storage and organization in the brain.” Em: *Psychological review* 65.6 (1958), página 386 (ver páginas 41, 116, 117).
- [91] Frank Rosenblatt. “On the convergence of reinforcement procedures in simple perceptrons”. Em: *Cornell Aeronautical Laboratory Report VG-1196-G-4, Buffalo, NY* (1960), página 72 (ver página 116).
- [92] Frank Rosenblatt. *Principles of neurodynamics. perceptrons and the theory of brain mechanisms*. Relatório técnico. Cornell Aeronautical Lab Inc Buffalo NY, 1961 (ver páginas 116, 118).
- [93] Sebastian Ruder. “An overview of gradient descent optimization algorithms”. Em: *arXiv preprint arXiv:1609.04747* (2016) (ver página 114).
- [94] David E Rumelhart, Geoffrey E Hinton e Ronald J Williams. “Learning representations by back-propagating errors”. Em: *Nature* 323.6088 (1986), páginas 533–536 (ver página 149).
- [95] S. Boughorbel, J-P. Tarel e N. Boujemaa. “Conditionally Positive Definite Kernels for SVM Based Image Recognition”. Em: *IEEE International Conference on Multimedia and Expo (ICME’05)*. Jul. de 2005 (ver página 227).
- [96] Takaya Saito e Marc Rehmsmeier. “The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets”. Em: *PloS one* 10.3 (2015) (ver página 274).
- [97] J. Schenk, M. Kaiser e G. Rigoll. “Selecting Features in On-Line Handwritten Whiteboard Note Recognition: SFS or SFFS?” Em: *2009 10th International Conference on Document Analysis and Recognition* (2009), páginas 1251–1254 (ver página 308).
- [98] Bernhard Schölkopf e Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001 (ver páginas 39, 225, 228).

- [99] Claude E Shannon. “A note on the concept of entropy”. Em: *Bell System Tech. J* 27.3 (1948), páginas 379–423 (ver páginas 234, 243).
- [100] Yoan Shin e Joydeep Ghosh. “The pi-sigma network: An efficient higher-order neural network for pattern classification and function approximation”. Em: *IJCNN-91-Seattle international joint conference on neural networks*. Volume 1. IEEE. 1991, páginas 13–18 (ver página 147).
- [101] L. Sirovich e M. Kirby. “Low-dimensional procedure for the characterization of human faces”. Em: *J. Opt. Soc. Am. A* 4.3 (mar. de 1987), páginas 519–524. DOI: 10.1364/JOSAA.4.000519. URL: <http://josaa.osa.org/abstract.cfm?URI=josaa-4-3-519> (ver página 136).
- [102] Sistema de coordenadas cartesiano. *Sistema de coordenadas cartesiano — Wikipédia, A encyclopédia livre*. [Online; accessed 29-September-2020]. 2020. URL: [https://pt.wikipedia.org/wiki/Sistema\\_de\\_coordenadas\\_cartesiano](https://pt.wikipedia.org/wiki/Sistema_de_coordenadas_cartesiano) (ver página 15).
- [103] Marina Sokolova e Guy Lapalme. “A systematic analysis of performance measures for classification tasks”. Em: *Information processing & management* 45.4 (2009), páginas 427–437 (ver página 268).
- [104] Petr Somol et al. “Adaptive floating search methods in feature selection”. Em: *Pattern recognition letters* 20.11-13 (1999), páginas 1157–1163 (ver páginas 308, 311, 313, 315).
- [105] Bharath K. Sriperumbudur et al. “Hilbert Space Embeddings and Metrics on Probability Measures”. Em: *Journal of Machine Learning Research* 11 (2010), páginas 1517–1561 (ver página 227).
- [106] Ilya Sutskever et al. “On the importance of initialization and momentum in deep learning”. Em: *International conference on machine learning*. 2013, páginas 1139–1147 (ver páginas 112, 115).
- [107] M. Tenenhaus. *La régression PLS: Théorie et pratique*. Editions Technip, 1998. ISBN: 9782710807353 (ver páginas 57, 58, 256).
- [108] Teorema da aproximação universal. *Teorema da aproximação universal — Wikipédia, a encyclopédia livre*. [Online; accessed 29-September-2020]. 2020. URL: [https://pt.wikipedia.org/wiki/Teorema\\_da\\_aproxima%C3%A7%C3%A3o\\_universal](https://pt.wikipedia.org/wiki/Teorema_da_aproxima%C3%A7%C3%A3o_universal) (ver página 151).
- [109] Testes de hipóteses. *Testes de hipóteses — Wikipédia, A encyclopédia livre*. [Online; accessed 29-September-2020]. 2020. URL: [https://pt.wikipedia.org/wiki/Testes\\_de\\_hip%C3%B3teses](https://pt.wikipedia.org/wiki/Testes_de_hip%C3%B3teses) (ver página 305).
- [110] Matthew Turk e Alex Pentland. “Eigenfaces for Recognition”. Em: *Journal of Cognitive Neuroscience* 3.1 (1991). PMID: 23964806, páginas 71–86. DOI: 10.1162/jocn.1991.3.1.71 (ver página 136).
- [111] Universal approximation theorem. *Universal approximation theorem — Wikipédia, a encyclopédia livre*. [Online; accessed 29-September-2020]. 2020. URL: [https://en.wikipedia.org/wiki/Universal\\_approximation\\_theorem](https://en.wikipedia.org/wiki/Universal_approximation_theorem) (ver página 151).
- [112] V. Vapnik. *The Nature of Statistical Learning Theory*. Information Science and Statistics. Springer New York, 2013. ISBN: 9781475732641. URL: <https://books.google.com.br/books?id=EqgAAQ> (ver página 35).
- [113] Sudhir Varma e Richard Simon. “Bias in error estimation when using cross-validation for model selection”. Em: *BMC bioinformatics* 7.1 (2006), página 91 (ver páginas 290, 291).
- [114] Paul Werbos. “Beyond regression: new tools for prediction and analysis in the behavioral sciences”. Em: *Ph. D. dissertation, Harvard University* (1974) (ver página 149).

- [115] Kai Yu, Liang Ji e Xuegong Zhang. “Kernel Nearest-Neighbor Algorithm”. Em: *Neural Processing Letters* 15 (2 2002). 10.1023/A:1015244902967, páginas 147–156. ISSN: 1370-4621 (ver páginas 225, 228).
- [116] G Udny Yule. “On the methods of measuring association between two attributes”. Em: *Journal of the Royal Statistical Society* 75.6 (1912), páginas 579–652 (ver página 245).
- [117] Matthew D Zeiler. “ADADELTA: an adaptive learning rate method”. Em: *arXiv preprint arXiv:1212.5701* (2012) (ver páginas 113, 114).

## **Índice Remissivo**

ensemble, 252

mediana, 263

percentil, 262

quartil, 263