

MAE 5905: Introdução à Ciência de Dados

Pedro A. Morettin

Instituto de Matemática e Estatística
Universidade de São Paulo
pam@ime.usp.br
<http://www.ime.usp.br/~pam>

Aula 11

12 de maio de 2023

Sumário

1 Classificação por meio de árvores

2 Bagging

3 Boosting

4 Florestas Aleatórias

5 Árvores para regressão

Preliminares I

- Modelos baseados em árvores foram desenvolvidos por Leo Breiman e associados e são bastante utilizados tanto para classificação quanto para previsão.
- Esses modelos são baseados numa segmentação do espaço gerado pelas variáveis explicativas (preditoras) em algumas regiões em que ou a moda (no caso de variáveis respostas categorizadas) ou a média (no caso de variáveis contínuas) é utilizada como predição.
- A definição dessas regiões é baseada em alguma medida de erro de previsão (ou de classificação). Em geral, as árvores são construídas a partir de um conjunto de **observações de treinamento** e testadas em um conjunto de **observações de teste**
- Os modelos de árvores de decisão são conceitualmente e computacionalmente simples e bastante populares em função de sua interpretabilidade, apesar de serem menos precisos que outros modelos mais complexos.

Preliminares II

- Generalizações dos modelos originais, conhecidos como florestas aleatórias (*random forests*) costumam apresentar grande precisão, mesmo quando comparados com modelos lineares, porém pecam pela dificuldade de interpretação. A referência básica para esse tópico é Breiman et al. (1984). O texto de Hastie and Tibshirani (1990) também contém resultados sobre esse tópico.
- Para predizer o valor de uma variável resposta Y (no caso de variáveis contínuas) ou classificar as observações em uma de suas categorias (no caso de variáveis categorizadas) a partir de um conjunto de variáveis preditoras X_1, \dots, X_p , o algoritmo usado na construção de árvores de decisão consiste essencialmente na determinação das regiões (retângulos mutuamente exclusivos) em que o espaço das variáveis preditoras é particionado. A metodologia desenvolvida em Breiman et al. (1994), e designada **CART** (de *Classification And Regression Trees*) é baseada na seguinte estratégia:
 - (a) Considere uma partição do espaço das variáveis preditoras (conjuntos dos possíveis valores de X_1, \dots, X_p) em M regiões, R_1, \dots, R_M .

Preliminares III

- (b) Para cada observação pertencente a R_j , o previsor (ou categoria) de Y (que designaremos \widehat{Y}_{R_j}) será a moda (no caso discreto), a média (no caso contínuo) ou a porcentagem (no caso categorizado) dos valores de Y correspondentes àqueles com valores de X_1, \dots, X_p em R_j .
- Embora a partição do espaço das variáveis preditoras seja arbitrária, usualmente ela é composta por retângulos p -dimensionais que devem ser construídos de modo a minimizar alguma medida de erro de previsão ou de classificação (que explicitaremos posteriormente).
 - Como esse procedimento geralmente não é computacionalmente factível dado o número de partições possíveis, mesmo com p moderado, usa-se uma **divisão binária recursiva** (*recursive binary splitting, RBS*) que é uma abordagem **top-down and greedy**, segundo James et al. (2013).
 - Dado o vetor de variáveis preditoras $\mathbf{X} = (X_1, \dots, X_p)^\top$, o algoritmo consiste dos passos:
 - (i) Selecione uma variável preditora X_j e um limiar (ou ponto de corte) t , de modo que a divisão do espaço das variáveis preditoras nas regiões $\{\mathbf{X} : X_j < t\}$ e $\{\mathbf{X} : X_j \geq t\}$ corresponda ao menor erro de previsão (ou de classificação).

Preliminares IV

- (ii) Para todos os pares (j, t) , considere as regiões

$$R_1(j, t) = \{\mathbf{X} : X_j < t\}, \quad R_2(j, t) = \{\mathbf{X} : X_j \geq t\}$$

e encontre o par (j, s) que minimize o erro de predição (ou de classificação) adotado.

- (iii) Repita o procedimento, agora dividindo uma das duas regiões encontradas, obtendo três regiões; depois divida cada uma dessas três regiões minimizando o erro de predição (ou de classificação);
 - (iv) Continue o processo até que algum critério de parada seja satisfeito.
- Um possível critério de parada consiste na obtenção de um número mínimo fixado de observações em cada região.

Árvores para Classificação

- Quando a variável resposta Y é categorizada, o objetivo é identificar a classe mais provável (**classe modal**) associada aos valores $\mathbf{X} = (X_1, \dots, X_p)^\top$ das variáveis preditoras.
- Neste caso, uma medida de erro de classificação, comumente denominada **taxa de erros de classificação** (TEC) é a proporção de observações do conjunto de treinamento que não pertencem à classe modal.
- Outras medidas de erro de classificação, como entropia ou índice de Gini também podem ser usadas.
- Admitamos que a variável resposta tenha K classes e que o espaço de variáveis preditoras seja particionado em M regiões. Designando por \hat{p}_{mk} , a proporção de observações de treinamento da m -ésima região, $m = 1, \dots, M$, pertencentes à k -ésima classe $k = 1, \dots, K$, a taxa de erros de classificação dos elementos pertencentes à m -ésima região é

$$TEC_m = 1 - \max_k(\hat{p}_{mk}).$$

Árvores para Classificação

- Como alternativa para as taxas de erros de classificação, pode-se usar o **índice de Gini** definido como

$$G_m = \sum_{k=1}^K \hat{p}_{mk} (1 - \hat{p}_{mk}),$$

que, essencialmente, corresponde à soma das variâncias das proporções de classificação em cada classe. Quando o valor de \hat{p}_{mk} para um dado nó m estiver próximo de 1 para uma dada categoria k e estiver próximo de zero para as demais categorias, o índice de Gini correspondente estará próximo de zero, indicando que para esse nó, uma das K categorias concentrará uma grande proporção dos elementos do conjunto de dados; poucos deles serão classificadas nas demais $K - 1$ categorias. Quanto mais concentrados em uma categoria forem as classificações em um dado nó, tanto maior será o seu grau de **pureza**.

- Outra medida utilizada com o mesmo propósito e que tem características similares àquelas do coeficiente de Gini é a **entropia cruzada**, definida como

$$ET_m = \sum_{k=1}^K \hat{p}_{mk} \log(\hat{p}_{mk}).$$

Árvores–Exemplo 1

- Vários pacotes, `tree`, `partykit`, `rpart`, `caret` podem ser utilizados com esse propósito. Cada um desses pacotes é regido por parâmetros que controlam diferentes aspectos da construção das árvores. Consultar os manuais correspondentes para nortear uma seleção adequada a problemas específicos.
- **Exemplo 1.** Consideremos novamente os dados analisados no Exemplo 9.2, disponíveis no arquivo `tipofacial`, extraídos de um estudo cujo objetivo era avaliar se duas ou mais medidas ortodônticas poderiam ser utilizadas para classificar indivíduos segundo o tipo facial (braquicéfalo, mesocéfalo ou dolicocéfalo).
Como no Exemplo 9.2, para efeitos didáticos consideramos apenas duas variáveis preditoras, correspondentes a duas distâncias de importância ortodôntica, nomeadamente, a altura facial (`altfac`) e a profundidade facial (`proffac`).

Árvores–Exemplo 1

- Os comandos do pacote `partykit` (com os parâmetros `default`) para a construção da árvore de classificação e do gráfico correspondente seguem juntamente com os resultados

```
facetree <- ctree(grupo ~ altfac + proffac, data=face)
facetree
```

Model formula:

```
grupo ~ altfac + proffac
```

Fitted party:

```
[1] root
|   [2] altfac <= -0.1: dolico (n = 31, err = 9.7%)
|   [3] altfac > -0.1
|   |   [4] altfac <= 0.8: meso (n = 28, err = 14.3%)
|   |   [5] altfac > 0.8
|   |   |   [6] proffac <= -0.6: meso (n = 17, err = 41.2%)
|   |   |   [7] proffac > -0.6: braq (n = 25, err = 0.0%)
```

```
Number of inner nodes:      3
```

```
Number of terminal nodes: 4
```

```
> plot(facetree)
```

Árvores–Exemplo 1

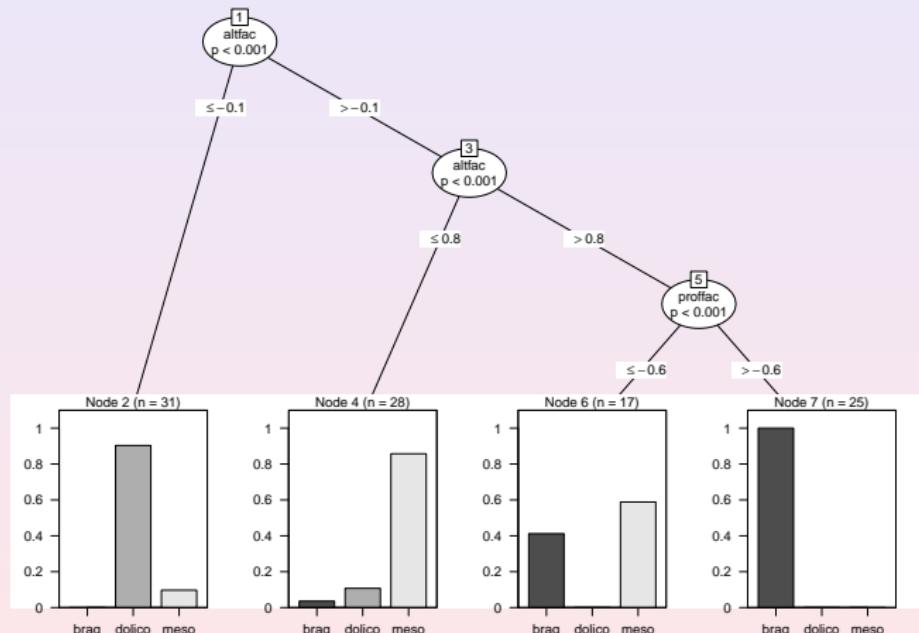


Figura 1: Árvore de decisão para os dados do Exemplo 1.

Árvores—Exemplo 1

- Na Figura 1, os símbolos ovais, que indicam divisões no espaço das variáveis preditoras são chamados de **nós internos** e os retângulos, que indicam as divisões finais são conhecidos por **nós terminais ou folhas** da árvore.
- Neste exemplo, temos 3 nós internos e 4 nós terminais. Os segmentos que unem os nós são os **galhos** da árvore.
- As barras em cada nó terminal indicam a frequência relativa com que as observações que satisfazem as restrições definidoras de cada galho são classificadas nas categorias da variável resposta.
- As regiões em que o espaço das variáveis preditoras foi particionado estão indicadas na Figura 2.

Árvores–Exemplo 1

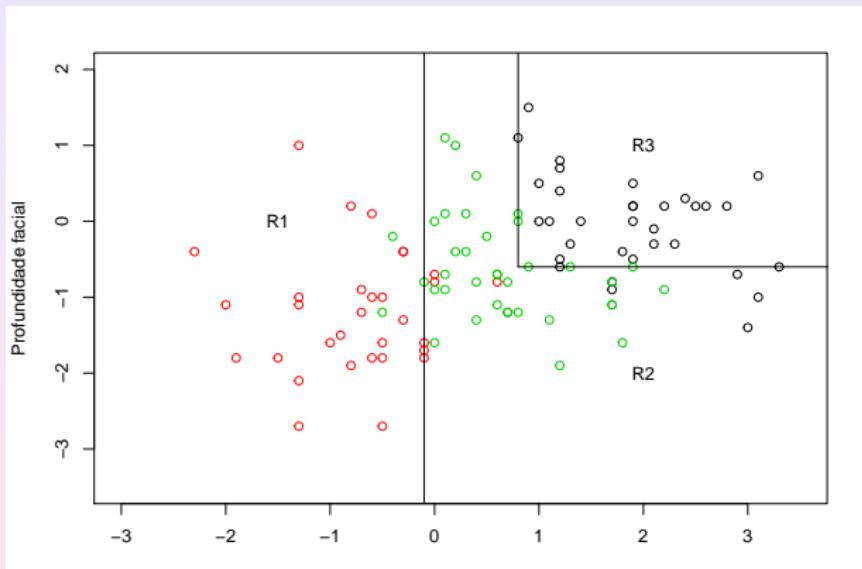


Figura 2: Regiões em que o espaço das variáveis preditoras do Exemplo 1 está particionado (círculos vermelhos = dolicocéfalos, verdes = mesocéfalos, pretos = braquicéfalos).

Árvores–Exemplo 1

- A variável preditora principal e o correspondente ponto de corte que minimiza a taxa de erros de classificação são **altfac** e $t = -0,1$, com $TEC = 9,7\%$.
- Para observações com valores $altfac \leq -0,1$ (região R_1), classificamos o indivíduo como dolicocéfalo.
- Para valores de $altfac > -0,1$, a classificação depende do valor de **proffac**. Nesse caso, se $altfac$ estiver entre $-0,1 \leq 0,8$, (região R_2), classificamos o indivíduo como mesocéfalo com $TEC = 14,3\%$.
- Se, por outro lado, $altfac > 0,8$ e $proffac \leq -0,6$, também classificamos o indivíduo como mesocéfalo (região R_2), com $TEC = 41,2\%$.
- Agora, se $altfac > 0,8$ e $proffac > -0,6$, o indivíduo deve ser classificado como braquicéfalo (região R_3), com $TEC = 0,0\%$.

Árvores–Exemplo 1

- Uma tabela com as classificações originais e preditas por meio da árvore de classificação é obtida por meio do comando `predict`

```
table(predict(facetree), face$grupo)
```

	braq	dolico	meso
braq	25	0	0
dolico	0	28	3
meso	8	3	34

- A acurácia do classificador é de 86% e $TEC = 14\%$.

Árvores–Exemplo 2

- Um dos problemas associados à construção de árvores de decisão está relacionado com o **sobreajuste** (*overfitting*).
- Se não impusermos uma regra de parada para a construção dos nós, o processo é de tal forma flexível que o resultado final pode ter tantos nós terminais quantas forem as observações, gerando uma árvore em que cada observação é classificada perfeitamente.
- Para contornar esse problema, pode-se considerar o procedimento conhecido como **poda**, que engloba técnicas para limitar o número de nós terminais das árvores.
- A ideia que fundamenta essas técnicas está na construção de árvores com menos nós e, consequentemente, com menor variância e interpretabilidade. O preço a pagar é um pequeno aumento no viés.

Árvores–Exemplo 2

- **Exemplo 2.** Consideremos agora os dados do arquivo **coronarias** provenientes de um estudo cujo objetivo era avaliar fatores prognósticos de lesão obstrutiva coronariana (L03) com categorias $1 : \geq 50\%$ ou $0 : < 50\%$ em 1500 pacientes.
- Embora tenham sido observadas cerca de 70 variáveis preditoras, aqui trabalharemos com **SEXO** (0=fem, 1=masc), **DIAB** (diabetes: 0=não, 1=sim), **IMC** (índice de massa corpórea), **IDADE1** (idade), **TRIG** (concentração de triglicérides) e **GLIC** (concentração de glicose).
- Com propósito didático eliminamos casos em que havia dados omissos em alguma dessas variáveis, de forma que 1034 pacientes foram considerados na análise.
- Os comandos do pacote **rpart** para a construção da árvore de classificação por meio de validação cruzada com os resultados correspondentes e o gráfico associado, disposto na Figura 3 seguem:

Árvores–Exemplo 2

```
lesaoobs <- rpart(formula = L03 ~ IDADE1 + SEXO + GLIC + DIAB +
                     IMC + TRIG, method="class", data = coronarias3,
                     xval = 20, minsplit=10, cp=0.005)
printcp(lesaoobs)
```

Variables actually used in tree construction:

```
[1] GLIC IDADE1 IMC SEXO TRIG
```

Root node error: 331/1034= 0.32012

n= 1034

Árvores–Exemplo 2

	CP	nsplit	rel error	xerror	xstd
1	0.0453172	0	1.00000	1.00000	0.045321
2	0.0392749	3	0.85801	0.97281	0.044986
3	0.0135952	4	0.81873	0.88218	0.043733
4	0.0090634	6	0.79154	0.87915	0.043687
5	0.0075529	7	0.78248	0.88822	0.043823
6	0.0060423	11	0.75227	0.92749	0.044386
7	0.0050000	13	0.74018	0.97885	0.045062

- A função `rpart()` tem um procedimento de VC embutido, ou seja, usa o CTr para construir a árvore e outro (CTeste) para avaliar a taxa de erros de previsão, repetindo o processo várias vezes. Cada linha da tabela representa um nível diferente da árvore. O erro de classificação obtido por validação cruzada tende a aumentar, pelo menos após o nível ótimo.
- A taxa de erro no **nó raiz** (root node error) corresponde à decisão obtida quando todas as observações do conjunto de dados são classificados na categoria LO3> 50%.
- O erro relativo (**rel error**) mede o erro relativo de classificação dos dados de treinamento obtido por intermédio da árvore. O termo rotulado **xerror** mede o erro relativo de classificação dos elementos do conjunto teste por intermédio da árvore construída com os dados do CTr, e o correspondente erro padrão é rotulado **xstd**.

Árvores–Exemplo 2

- O produto **root node error** \times **rel error** corresponde ao valor absoluto da taxa de erros obtida no CTr ($0,263=0,320 \times 0,740$ para a árvore com 13 subdivisões). O produto **root node error** \times **xerror** corresponde ao valor absoluto da taxa de erros obtida no CTeste e corresponde a uma medida mais objetiva da acurácia da previsão ($0,313=0,320 \times 0,979$ para a árvore com 13 subdivisões).
- A árvore gerada obtida por intermédio do comando

```
> rpart.plot(lesaoobs, clip.right.labs = TRUE, under = FALSE,  
             extra = 101, type=4)
```

está disposto na Figura 3.

Árvores–Exemplo 2

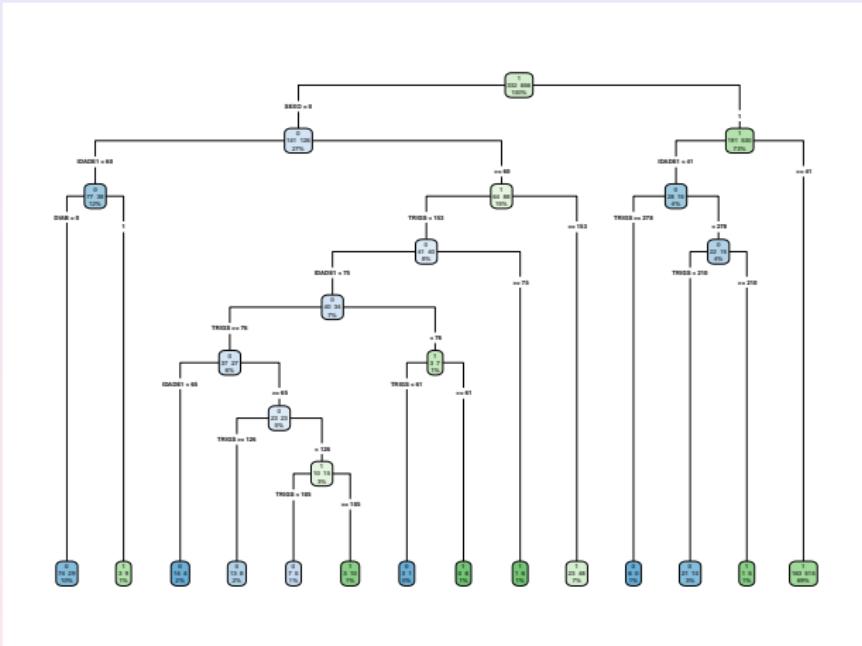


Figura 3: Árvore de decisão para os dados do Exemplo 2.

Árvores–Exemplo 2

- A tabela com as classificações originais e preditas é obtida por meio do comando

```
table(coronarias\$L03, predict(lesaoobs, type="class"))  
0 1  
0 145 186  
1 59 644
```

e indica um erro de classificação de $23,4\% = (186 + 59)/1034$.

- O parâmetro CP (**complexity parameter**) serve para controlar o tamanho da árvore e corresponde ao menor incremento no custo do modelo necessário para a adição de uma nova variável.
- Um gráfico com a variação desse parâmetro com o número de nós, obtido com o comando `plotcp()` pode ser visto na Figura 4.

Árvores–Exemplo 2

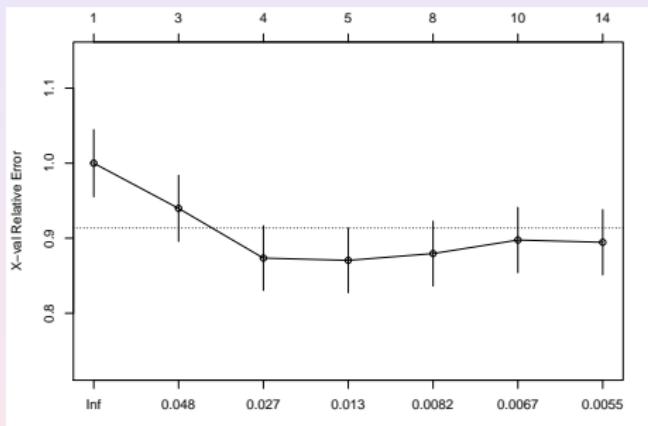


Figura 4: Gráfico CP para o ajuste da árvore aos dados do Exemplo 2.

Nesse gráfico procura-se o nível para o qual o erro relativo obtido por VC é mínimo. Para o exemplo, esse nível é 4, sugerindo que a árvore obtida no exemplo deve ser podada.

Árvores–Exemplo 3

- Vejamos, agora, ver um exemplo usando o pacote `tree`.
- **Exemplo 3.** Vamos considerar os dados de crianças que foram submetidas a uma cirurgia da coluna para corrigir cifose congênita. Veja Chambers e Hastie (1992). Os dados contém 81 observações, com as variáveis:

Y : cifose: uma variável qualitativa (atributo), com os valores *ausente* e *presente*, indicando se cifose estava ausente ou presente após a cirurgia;

X_1 : Age, em meses;

X_2 : Number, o número de vértebras envolvidas;

X_3 : Start, o número da primeira vértebra (a partir do topo) operada.

- O sumário do ajuste e a Figura 5 estão ilustradas a seguir.

Árvores–Exemplo 3

Classification tree:

```
tree(formula = Kyphosis ~ Age + Number + Start, data = kyphosis)
Number of terminal nodes: 10
Residual mean deviance: 0.5809 = 41.24 / 71
Misclassification error rate: 0.1235 = 10 / 81
```

- Vemos que a taxa do erro de classificação é 12,35% e o desvio (*deviance*) é definido por

$$-2 \sum_m \sum_{mk} K n_{mk} \log \hat{p}_{mk},$$

onde n_{mk} é o número de observações no m -ésimo nó terminal que pertence à k -ésima classe. Um desvio pequeno indica um bom ajuste aos dados de treinamento. O desvio médio da saída acima é dado pelo desvio dividido por $n - n_0$, n_0 sendo o número de nós terminais, no caso, 10.

- As regiões que determinam a figura são retângulos no espaço tridimensional, logo é complicado, ou não é possível, vê-las em um gráfico.

Árvores–Exemplo 3

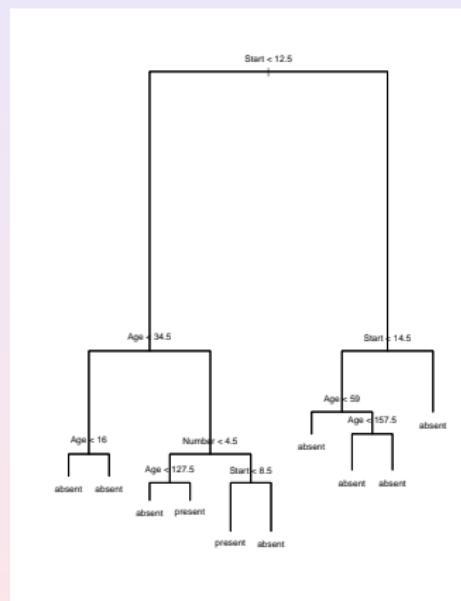


Figura 5: Árvore para o Exemplo 3, com 3 preditores.

Árvores—Exemplo 3

Para poder ver uma partição, vamos considerar somente dois preditores, Start e Age. A árvore correspondente está na Figura 6 e a partição na Figura 7.

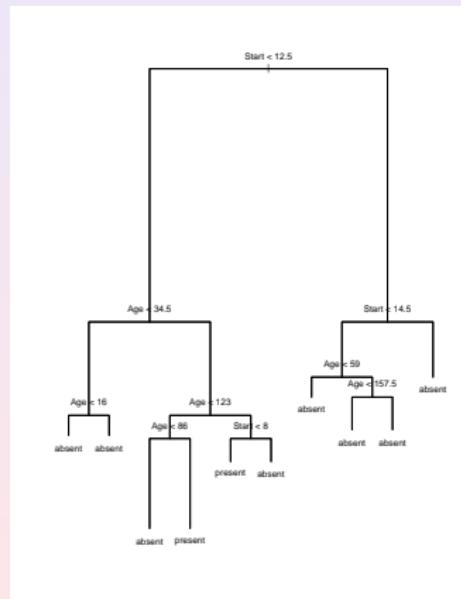


Figura 6: Árvore para o Exemplo 3, com 2 preditores, Start e Age.

Árvores–Exemplo 3

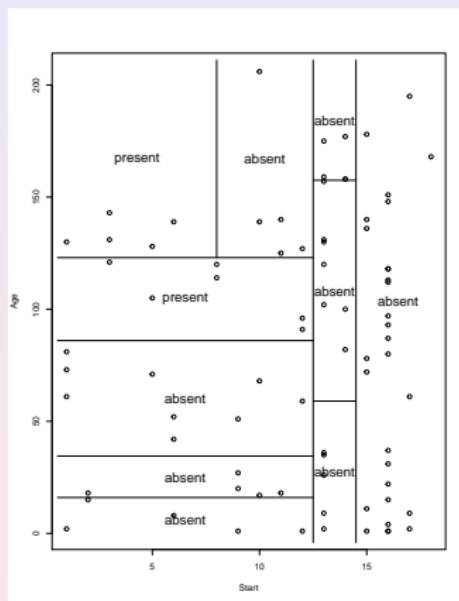


Figura 7: Regiões para o Exemplo 10.3, com 2 preditores.

Árvores–Exemplo 3

Com relação à Figura 26, algumas regiões que podem ser deduzidas são (sendo $\mathbf{X} = (X_1, X_2, X_3)'$):

$R_1 = \{\mathbf{X} : X_1 < 16, X_3 < 12, 5\}$, o valor previsto de Y é ausente;

$R_2 = \{\mathbf{X} : 16 < X_1 < 34, 5, X_3 < 12, 5\}$, o valor previsto de Y é ausente;

$R_3 = \{\mathbf{X} : 34, 5 < X_1 < 127, 5, X_2 < 4, 5, X_3 < 12, 5\}$, o valor previsto de Y é ausente etc.

Poda de uma árvore

- No Exemplo 2 (coronarias), vimos que a árvore deve ser podada, inspecionando o parâmetro de complexidade, CP, que indica valores entre 0,011 e 0,023, com nós entre 5 e 7.
- Uma regra empírica para se efetuar a poda da árvore consiste em escolher a **menor árvore** para a qual o valor de **xerror** é menor do que a soma do menor valor observado de **xerror** com seu erro padrão **xstd**. No exemplo em estudo, o menor valor de **xerror** é 0,87915 e o de seu erro padrão **xstd** é 0,043687 de maneira que a árvore a ser construída por meio de poda deverá ser a menor para a qual o valor de **xerror** seja menor que 0,922837 ($= 0,87915 + 0,043687$), ou seja a árvore com 4 subdivisões e 5 nós terminais.
- A poda juntamente com o gráfico da árvore podada e a tabela com os correspondentes valores preditos podem ser obtidos com os comandos a seguir:

Poda de uma árvore

```
lesaoobspoda <- prune(lesaoobs, cp = 0.015 , "CP", minsplit=20, xval=25)
rpart.plot(lesaoobspoda, clip.right.labs = TRUE, under = FALSE,
           extra = 101, type=4)
rpart.rules(lesaoobspoda, cover = TRUE)

L03                                              cover
0.33 when SEXO is 0 & IDADE1 < 63 & GLIC < 134    13%
0.35 when SEXO is 1 & IDADE1 < 41                      4%
0.59 when SEXO is 0 & IDADE1 >= 63 & GLIC < 134     10%
0.77 when SEXO is 1 & IDADE1 >= 41                     69%
0.85 when SEXO is 0                                     & GLIC >= 134    4%
```

Poda de uma árvore

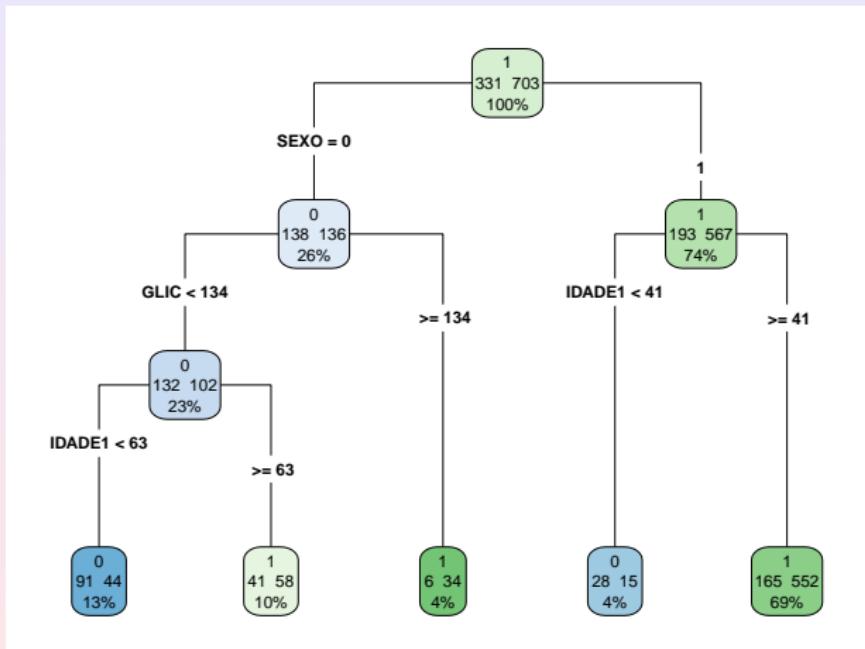


Figura 8: Árvore (podada) ajustada aos dados do Exemplo 2.

Poda de uma árvore

- O número indicado na parte superior de cada nó da Figura 8 indica a classe majoritária, na qual são classificadas os elementos do conjunto de treinamento; o valor à esquerda no centro do nó representa a frequência desses elementos pertencentes à classe $LO_3 = 0$ e o valor à direita corresponde à frequência daqueles pertencentes à classe $LO_3 = 1$. Na última linha aparece a porcentagem de elementos correspondentes a cada nó terminal.
- A tabela de classificação para o conjunto de dados original (treinamento + validação) obtida a partir da árvore podada é

0	1
0	119 212
1	59 644

Nessa tabela, as linhas indicam a classificação real e as colunas informam a classificação predita pela árvore podada. O erro de classificação, 26,2%, é ligeiramente maior que o erro obtido com a árvore original, bem mais complexa.

Bagging

- Usualmente, árvores de decisão produzem resultados com grande variância, ou seja, dependendo de como o conjunto de dados é subdividido em conjuntos de treinamento e de teste, as árvores produzidas podem ser diferentes. As técnicas que descreveremos a seguir têm a finalidade de reduzir essa variância.
- A técnica de **agregação bootstrap** (*bootstrap aggregating*) ou, simplesmente **bagging**, é um método para gerar múltiplas versões de um previsor (ou classificador) a partir de vários conjuntos de treinamento e, com base nessas versões, construir um previsor (ou classificador) agregado.
- O ideal seria ter vários conjuntos de treinamento, construir previsores e tomar uma média (no caso de regressão) dos previsores. Todavia, na prática, isso não acontece. Uma maneira de prosseguir, é usar réplicas bootstrap do conjunto de treinamento.

Bagging

- Suponha que tenhamos o conjunto de treinamento $(x_1, y_1), \dots, (x_n, y_n)$ e considere o caso de y quantitativa (regressão). Considere B réplicas bootstrap desse conjunto e para cada réplica b , obtemos o previsor de y , $\hat{f}^{*b}(x)$ e construímos o previsor

$$\hat{f}_{\text{bag}} = \frac{1}{B} \sum_{b=1}^B \hat{f}^{*b}(x). \quad (1)$$

- No caso de classificação, para uma observação teste, considere a classe prevista para cada uma das B árvores e, então, tome como previsor a classe de maior ocorrência comum entre os B previsores (voto majoritário). Especificamente,

$$\hat{c}_{\text{bag}}(x) = \operatorname{argmax}_k [\#\{(b | \hat{c}^b(x) = k)\}]$$

em que $\#\{A\}$ denota a cardinalidade do conjunto A .

- O número de réplicas bootstrap sugerido por Breiman (1996) é cerca de 25. Para detalhes sobre bootstrap veja Efron e Tibshirani (1993) e Notas de Capítulo.

Bagging—Exemplo 4

- **Exemplo 4.** A técnica bagging pode ser aplicada aos dados do Exemplo 2 por meio dos comandos

```
> set.seed(054)
>
> # train bagged model

> lesaoobsbag <- bagging(
+   formula = L03 ~ SEXO + IDADE1+ GLIC,
+   data = coronarias3,
+   nbagg = 200,
+   coob = TRUE,
+   control = rpart.control(minsplit = 20, cp = 0.015)
+ )
>
> lesaoobspred <- predict(lesaoobsbag, coronarias3)
> table(coronarias3\$L03, predict(lesaoobsbag, type="class"))
```

	0	1
0	117	214
1	78	625

- Variando a semente do processo aleatório, as taxas de erros de classificação giram em torno de 27% a 28%.

Bagging—Exemplo 4

Quatro das 200 árvores obtidas em cada réplica bootstrap podem ser obtidas por meio dos comandos a seguir e estão representadas na Figura 9.

```
as.data.frame(coronarias4)
clr12 = c("#8dd3c7", "#ffffb3", "#bebada", "#fb8072")
n = nrow(coronarias4)
par(mfrow=c(2,2))
sed=c(1,10,22,345)
for(i in 1:4){
  set.seed(sed[i])
  idx = sample(1:100, size=n, replace=TRUE)
  cart = rpart(L03 ~ DIAB + IDADE1 + SEXO, data=coronarias4[idx,], mod
  prp(cart, type=1, extra=1, box.col=clr12[i])
}
```

Bagging—Exemplo 4

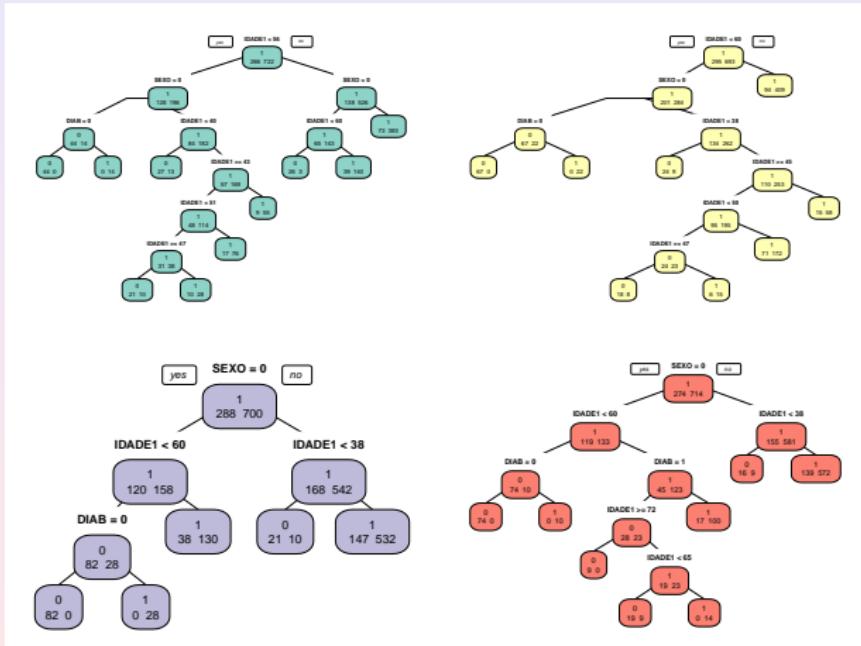


Figura 9: Exemplos de Árvores obtidas por bagging para os dados do Exemplo 2.

Boosting I

- ① O objetivo do procedimento **boosting** é reduzir o viés e a variância em modelos utilizados para aprendizado supervisionado.
- ② A ideia básica é considerar um conjunto de previsores (**classificadores fracos**) de modo a obter um **previsor (classificador) forte**.
- ③ Classificadores fracos têm taxas de erro de classificação altas. No caso binário, por exemplo, isso corresponde a uma taxa próxima de 0,50, que seria obtida com uma decisão baseada num lançamento de moeda. Um classificador forte, por outro lado, tem uma taxa de erro de classificação baixa.
- ④ Diferentemente da técnica bagging, em que B árvores são geradas independentemente por meio de bootstrap, com cada observação tendo a mesma probabilidade de ser selecionada em cada um dos conjuntos de treinamento, no procedimento boosting, as B árvores são geradas **sequencialmente** a partir de um único conjunto de treinamento, com probabilidades de seleção (pesos) diferentes atribuídos às observações.

Boosting II

- ⑤ Observações mal classificadas em uma árvore recebem pesos maiores para seleção na árvore subsequente (obtida do mesmo conjunto de treinamento), com a finalidade de dirigir a atenção aos casos em que a classificação é mais difícil.
- ⑥ Em ambos os casos, o classificador final é obtido por meio da aplicação dos B classificadores fracos gerados com as diferentes árvores, por meio do voto majoritário. Além dos pesos atribuídos às observações no processo de geração dos classificadores fracos, o procedimento boosting atribui pesos a cada um deles, em função das taxas de erros de classificação de cada um.
- ⑦ Essencialmente, o classificador forte pode ser expresso como

$$\hat{c}_{\text{boost}}(x) = \sum_{b=1}^B \hat{c}^b(x) w(b) \quad (2)$$

em que $w(b)$ é o peso atribuído ao classificador $\hat{c}^b(x)$.

Boosting III

- ⑧ Se por um lado, o procedimento bagging raramente reduz o viés quando comparado com aquele obtido com uma única árvore de decisão, por outro, ele tem a característica de evitar o sobreajuste. Essas características são invertidas com o procedimento boosting.
- ⑨ Existem vários algoritmos para a implementação de boosting. O mais usado é o algoritmo conhecido como **AdaBoost** (de *adaptive boosting*), desenvolvido por Freund e Schapire (1997). Dada a dificuldade do processo de otimização de (2), esse algoritmo considera um processo iterativo de otimização que produz bons resultados embora não sejam ótimos.
- ⑩ A ideia é adicionar o melhor classificador fraco numa determinada iteração ao classificador fraco obtido na iteração anterior, ou seja, considerar o processo

$$\hat{c}_{\text{boost}}^b(\mathbf{x}) = \hat{c}_{\text{boost}}^{b-1}(\mathbf{x}) + \hat{c}^b(\mathbf{x})w(b)$$

em que $\hat{c}_{\text{boost}}^b(\mathbf{x})$ é o classificador com o melhor incremento no desempenho relativamente ao classificador $\hat{c}_{\text{boost}}^{b-1}(\mathbf{x})$.

Boosting IV

- 11 Nesse contexto, a escolha de $[\hat{c}^b(\mathbf{x}), w(b)]$ deve satisfazer

$$[\hat{c}^b(\mathbf{x}), w(b)] = \operatorname{argmin}_{[c^b(\mathbf{x}), w(b)]} \{E[\hat{c}_{\text{boost}}^{b-1}(\mathbf{x}) + \hat{c}^b(\mathbf{x})w(b)]\}$$

em que $E[c^b]$ denota o erro de classificação do classificador c^b .

- 12 Consideremos, por exemplo, um problema de classificação binária baseado num conjunto de treinamento com N observações. No algoritmo **AdaBoost**, o classificador sempre parte de um único nó (conhecido como **stump**), em que cada observação tem peso $1/N$.
- 13 O ajuste por meio desse algoritmo é realizado por meio dos seguintes passos:
- 1) Ajuste o melhor classificador (fraco) com os pesos atuais e repita os passos seguintes para os $B - 1$ classificadores subsequentes.
 - 2) Calcule o valor do peso a ser atribuído ao classificador fraco corrente a partir de alguma métrica que indique quanto esse classificador contribui para o classificador forte corrente.
 - 3) Atualize o classificador forte adicionando o classificador fraco multiplicado pelo peso calculado no passo anterior.

Boosting V

- 4) Com esse classificador forte, calcule os pesos atribuídos às observações de forma a indicar quais devem ser o foco da próxima iteração (pesos atribuídos às observações mal classificadas devem ser maiores que pesos atribuídos a observações bem classificadas).
- 14) Os algoritmos para implementação de boosting têm 3 parâmetros:
 - (i) o número de árvores, B , que pode ser determinado por validação cruzada;
 - (ii) **parâmetro de encolhimento** (*shrinkage*), $\lambda > 0$, pequeno, da ordem de 0,01 ou 0,001, que controla a velocidade do aprendizado;
 - (iii) o número de divisões em cada árvore, d ; como vimos, o AdaBoost, usa $d = 1$ e, em geral, esse valor funciona bem.
- 15) O pacote **gbm** implementa o boosting e o pacote **adabag** implementa o algoritmo AdaBoost.

Boosting–Exemplo 5

Exemplo 5. Os dados do CD-esteira contêm informações de 16 variáveis, sendo 4 qualitativas (Etiologia, Sexo, Classe funcional NYHA e Classe funcional WEBER) além de 13 variáveis quantitativas [Idade, Altura, Peso, Superfície corporal, Índice de massa corpórea (IMC), Carga, Frequência cardíaca (FC), VO₂ RER, VO₂/FC, VE/VO₂,VE/VCO₂] em diversas fases (REP, LAN, PCR e PICO) de avaliação de um teste de esforço realizado em esteira ergométrica. A descrição completa das variáveis está disponível no arquivo dos dados. Neste exemplo vamos usar as variáveis Carga, Idade, IMC e Peso na fase LAN (limiar anaeróbico) como preditores e VO₂ (consumo de oxigênio, como resposta. Para efeito do exemplo, essas variáveis, com as respectivas unidades de medida, serão denotadas como:

Y : consumo de oxigênio (VO₂), em $mL/kg/min$;

X_1 : carga na esteira, em W (Watts);

X_2 : índice de massa corpórea (IMC), em kg/m^2 .

X_3 : Idade, em anos;

X_4 : Peso, em kg .

Boosting–Exemplo 5

Vamos usar o pacote gbm. Com o comando `summary()` obtemos a influência relativa dos preditores e a figure respectiva, Figura 10.

```
summary(boost.esteira)
var      rel.inf
CARGA   43.46858
IMC     24.22825
Idade   16.84917
Peso    15.45401
```

Vemos que os preditores CARGA e IMC são os mais importantes. Podemos produzir gráficos parciais de dependência para essas duas variáveis (Figura 11). Vemos que o consumo de oxigênio cresce com a CARGA e diminui com o IMC.

Boosting–Exemplo 5

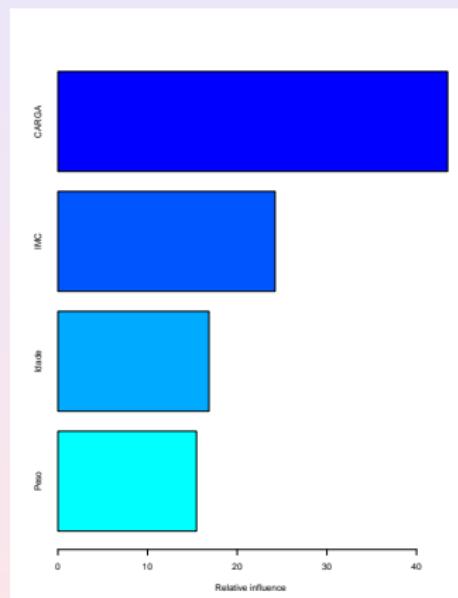


Figura 10: Influência elativa para os preditores do Exemplo 5.

Boosting–Exemplo 5

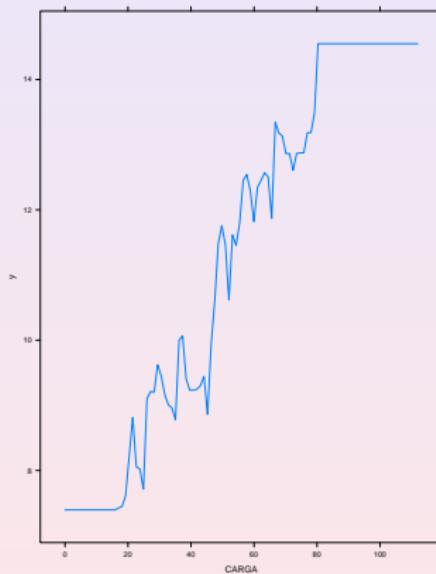


Figura 11: Consumo de oxigênio versus CARGA .

Boosting–Exemplo 5

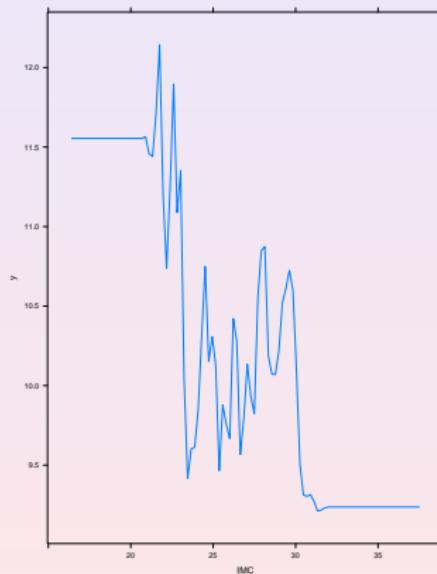


Figura 12: Consumo de oxigênio versus IMC .

Boosting–Exemplo 5

Agora usamos o modelo via *boosting* para prever VO2 para o conjunto teste.

```
yhat.boost=predict(boost.esteira,newdata=esteira2[-train,],n.trees=5000)
mean((yhat.boost-boston.test)^2)
[1] 2.063152e-05
```

Vemos que o EQM obtido via *boosting* é consideravelmente menor do que aquele obtido via *bagging*.

Florestas

- Bagging e florestas aleatórias têm o mesmo objetivo: diminuir a variância e o viés de procedimentos baseados em árvores de decisão.
- Enquanto os dois primeiros enfoques são baseados em um conjunto de B árvores utilizando o mesmo conjunto de p variáveis preditoras em cada um deles, o enfoque conhecido por **florestas aleatórias** utiliza diferentes conjuntos das variáveis preditoras na construção de cada árvore.
- Na construção de um novo nó, em vez de escolher a melhor variável dentre as p disponíveis no conjunto de treinamento, o algoritmo de florestas aleatórias seleciona a melhor delas dentre um conjunto de $m < P$ selecionadas ao acaso. Usualmente, escolhe-se $m \approx \sqrt{p}$.

Florestas

- Formalmente, para cada árvore j , um vetor aleatório θ_j é gerado, independentemente de vetores prévios $\theta_1, \dots, \theta_{j-1}$, mas com a mesma distribuição. A árvore usando o conjunto de treinamento e θ_j resulta num classificador $\hat{f}_j(\mathbf{x}, \theta_j)$. Cada árvore vota na classe mais popular para o vetor \mathbf{x} .
- A acurácia das árvores aleatórias é tão boa quanto a do *AdaBoost* e, às vezes, melhor. O resultado obtido por intermédio do algoritmo de árvores aleatórias em geral é mais robusto com relação a valores atípicos e ruído além de ser mais rápido do que bagging e boosting.
- O pacote **randomForest** do R implementa florestas. Nesse caso, o número de preditores tem que ser menor do que p .

Florestas–Exemplo 6

- Exemplo 6. Vamos usar o conjunto de dados `esteira2` e as variáveis CARGA e IMC para crescer uma floresta. Obtemos o sumário abaixo:

Call:

```
randomForest(formula = V02 ~ Idade + CARGA + IMC + Peso,  
             data = esteira2, mtry = 2, importance = TRUE,  
subset = train)
```

Type of random forest: regression

Number of trees: 500

No. of variables tried at each split: 2

Mean of squared residuals: 4.164505

Percentage Var explained: 52.23

- Obtendo-se previsões para o conjunto teste, temos o sumário abaixo:

```
yhat.rf=predict(rf.esteira,newdata=esteira2[-train,])  
mean((yhat.rf-esteira.test)^2)  
[1] 3.713565
```

- O EQM de previsão via floresta é menor do que aquele via *bagging*, mas maior do que o via *boosting*.

Florestas–Exemplo 6

Um sumário da importância de cada preditor é dado abaixo:

```
summary(boost.esteira)
var rel.inf
CARGA 43.46858
IMC    24.22825
Idade   16.84917
Peso    15.45401
```

e um gráfico está na Figura 13. Novamente, as variáveis CARGA e IMC são as mais importantes.

Concluindo, sobre os três algoritmos, *bagging*, floresta e *boosting*, o último é o que apresentou o maior poder preditivo.

Florestas–Exemplo 6

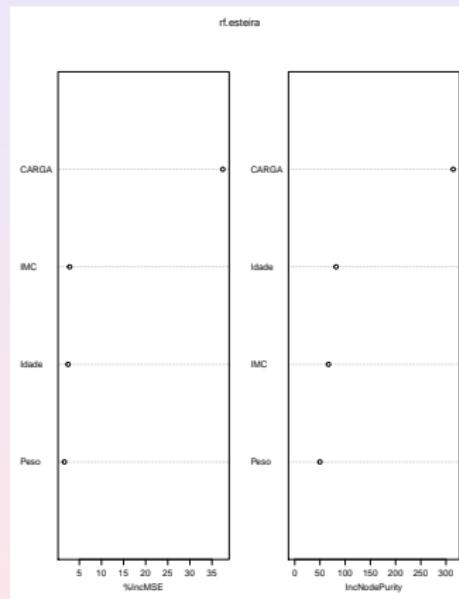


Figura 13: Importância relativa os preditores do Exemplo 6.

Árvores para regressão

- Consideremos uma situação com variáveis preditoras X_1, \dots, X_p e uma variável resposta quantitativa Y . Quando a relação entre variáveis resposta e preditoras for compatível com um modelo linear (no caso de uma relação polinomial, por exemplo), o uso de regressão linear é conveniente e obtemos modelos com interpretabilidade e poder preditivo satisfatórios.
- Quando essa condição não for satisfeita (no caso de modelos não lineares, por exemplo), o uso de árvores pode ser mais apropriado. Além disso, algumas das variáveis preditoras podem ser qualitativas e nesse caso não é necessário transformá-las em **variáveis fictícias** (*dummy variables*) como no caso de modelos de regressão usuais.
- A ideia subjacente é similar àquela empregada em modelos de classificação: subdividir o espaço gerado pelas variáveis explicativas em várias regiões e adotar como previsores, as respostas médias em cada região. As regiões são selecionadas de forma a produzir o menor **erro quadrático médio** ou o menor **coeficiente de determinação**.
- A construção de árvores de regressão pode ser concretizada por meio dos pacotes `tree` e `rpart` entre outros. Ilustraremos a construção de uma árvore para regressão por meio de um exemplo.

Árvores para regressão - Exemplo 7

- Os dados do arquivo **antracose** foram extraídos de um estudo cuja finalidade era avaliar o efeito da idade (**idade**), tempo vivendo em São Paulo (**tmunic**), horas diárias em trânsito (**htransp**), carga tabágica (**cargatabag**), classificação sócio-econômica (**ses**), densidade de tráfego na região onde o indivíduo morou (**densid**) e distância mínima entre a residência a vias com alta intensidade de tráfego (**distmin**) num índice de antracose (**antracose**) que é uma medida de fuligem (*black carbon*) depositada no pulmão.
- Como esse índice varia entre 0 e 1, consideramos

$$\logrc = \log[\text{índice de antracose}/(1 - \text{índice de antracose})]$$

como variável resposta.

- Inicialmente, construímos uma árvore de regressão para os dados por meio de validação cruzada. Os comandos do pacote **rpart** para gerar a árvore apresentada na Figura 14 são dados a seguir.

Árvores para regressão - Exemplo

```
pulmaotree2 <- rpart(formula = logrc ~ idade + tmunic + htransp +
    cargatabag + ses + densid + distmin, data=pulmao,
    method="anova", xval = 30, cp=0.010)
rpart.plot(pulmaotree2, clip.right.labs = TRUE, under = FALSE,
            extra = 101, type=4)
```

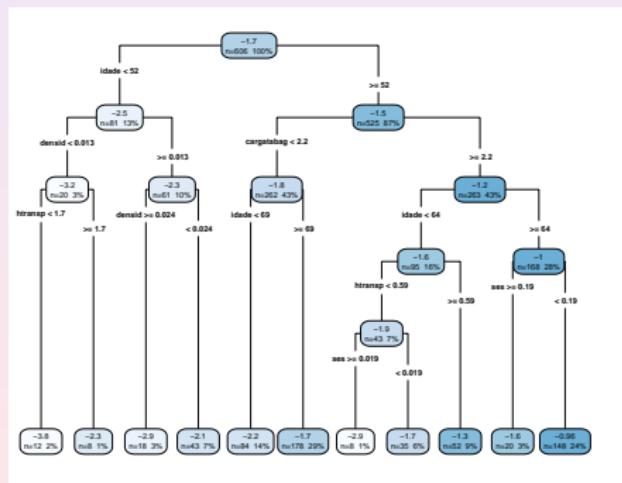


Figura 14: Árvore de decisão para os dados do Exemplo 7.

Árvores para regressão - Exemplo 7

- Os valores apresentados na parte superior de cada nó são as previsões associadas às regiões correspondentes. Na parte inferior encontram-se o número e a porcentagem de elementos incluídos em cada região.
- Para evitar um possível sobreajuste da árvore proposta, convém avaliar o efeito de seu tamanho segundo algum critério. No pacote **rpart** esse critério é baseado no parâmetro de complexidade (*CP*) que está relacionado com o número de nós terminais e na relação $1 - R^2$ em que R^2 tem a mesma interpretação do coeficiente de determinação utilizado em modelos lineares.
- Com essa finalidade, podemos utilizar o comando **rsq.rpart(pulmaotree2)**, que gera a tabela com os valores de *CP* e os gráficos apresentados na Figura 15.

Árvores para regressão - Exemplo 7

Variables actually used in tree construction:

```
[1] cargatabag densid      htransp      idade      ses
```

Root node error: 765.87/606 = 1.2638

n= 606

	CP	nsplit	rel error	xerror	xstd
1	0.087230	0	1.00000	1.00279	0.077419
2	0.062020	1	0.91277	0.96678	0.076624
3	0.025220	2	0.85075	0.91078	0.072406
4	0.024106	3	0.82553	0.91508	0.073489
5	0.015890	4	0.80142	0.85814	0.069326
6	0.014569	5	0.78553	0.87882	0.070312
7	0.011698	6	0.77097	0.89676	0.070309
8	0.011667	7	0.75927	0.90730	0.069025
9	0.011347	8	0.74760	0.91268	0.069066
10	0.010289	9	0.73625	0.91536	0.069189
11	0.010000	10	0.72596	0.91250	0.069202

Árvores para regressão - Exemplo 7

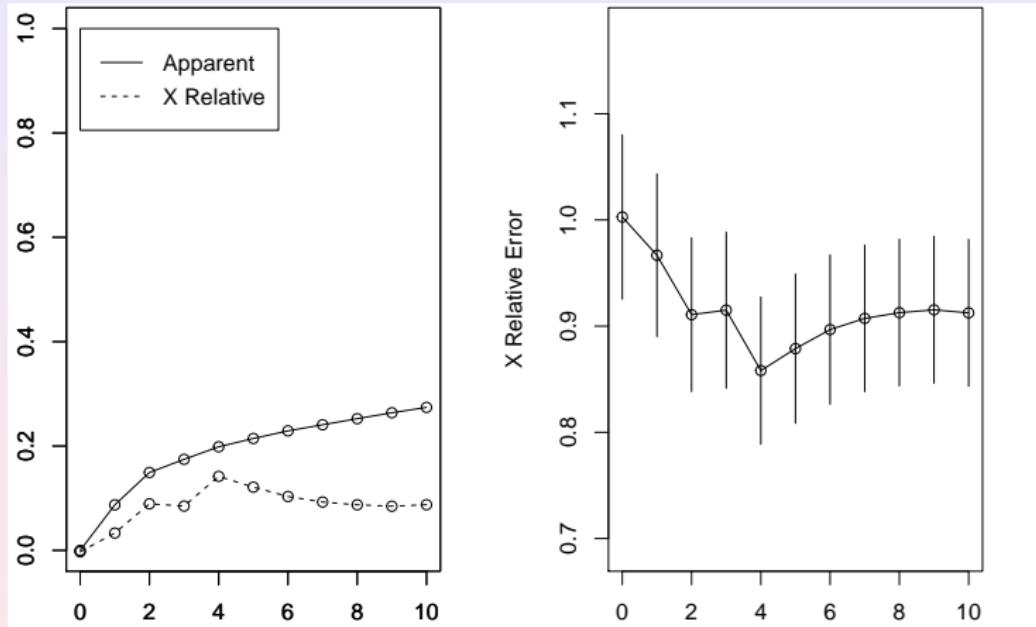


Figura 15: Efeito do número de divisões para a árvore ajustada aos dados do Exemplo 7.

Árvores para regressão - Exemplo 7

Utilizando o mesmo critério aplicado no caso de classificação, a sugestão é podar a árvore, fixando o número de subdivisões em 4, correspondendo a 5 nós terminais.

O gráfico da árvore podada (disposto na Figura 16), além da regras de partição do espaço das variáveis explicativas podem ser obtidos com os comandos

```
cpmin <- pulmaotree2$cptable[which.min(pulmaotree2$cptable[, "xerror"]),
                                "CP"]
pulmaotree2poda <- prune(pulmaotree2, cp = cpmin)
rpart.plot(pulmaotree2poda, clip.right.labs = TRUE, under = FALSE,
            extra = 101, type=4)
pulmaotree2poda$cptable
      CP nsplit rel error      xerror      xstd
1 0.08722980      0 1.0000000 1.0027880 0.07741860
2 0.06201953      1 0.9127702 0.9667786 0.07662398
3 0.02521977      2 0.8507507 0.9107847 0.07240566
4 0.02410635      3 0.8255309 0.9150809 0.07348898
5 0.01588985      4 0.8014246 0.8581417 0.06932582
> rpart.rules(pulmaotree2poda, cover = TRUE)
logrc                      cover
-2.5 when idade < 52          13%
-2.2 when idade is 52 to 69 & cargatabag < 2.2  14%
-1.7 when idade >= 69 & cargatabag < 2.2      29%
-1.6 when idade is 52 to 64 & cargatabag >= 2.2  16%
-1.0 when idade >= 64 & cargatabag >= 2.2      28%
```

Árvores para regressão - Exemplo 7

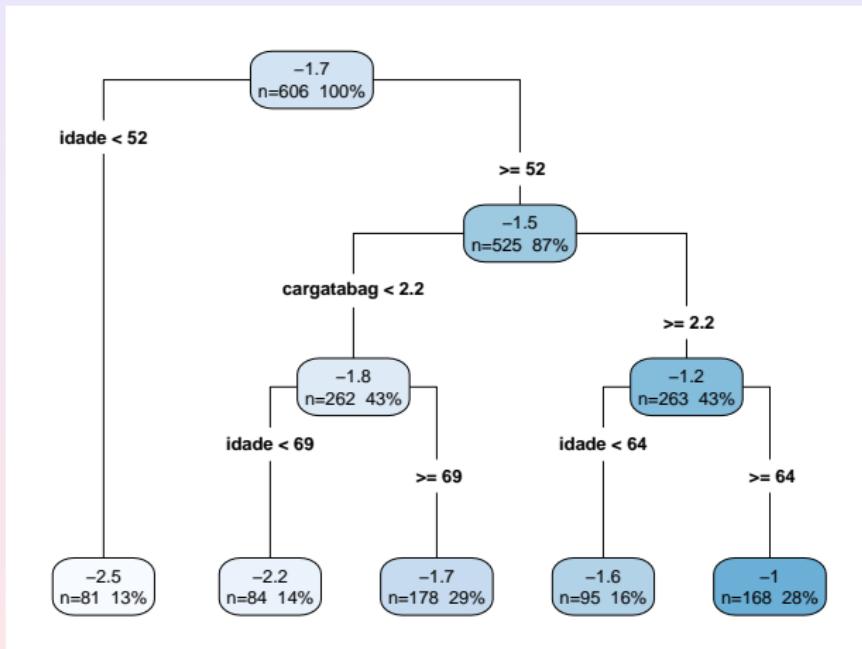


Figura 16: Árvore podada ajustada aos dados do Exemplo.

Árvores para regressão - Exemplo 7

Valores preditos para o conjunto de dados com o respectivo *RMSE* são obtidos por meio dos comandos

```
rmse = function(actual, predicted) {  
  sqrt(mean((actual - predicted)^2))  
}  
  
rmse(predpulmatree2poda, pulmao$logrc)  
[1] 1.006402
```

Convém notar que embora a utilização de árvores de decisão gere um modelo bem mais simples do que aquele obtido por meio de análise regressão, os objetivos são bem diferentes.

Nesse contexto, o modelo baseado em árvores deve ser utilizado apenas quando o objetivo é fazer previsões, pois pode deixar de incluir variáveis importantes para propósitos inferenciais. No Exemplo, uma das variáveis mais importantes para entender o processo de deposição de fuligem no pulmão é o numero de horas gastas em trânsito. Essa variável foi incluída significativamente no ajuste do modelo de regressão mas não o foi no modelo baseado em árvores.

Há um algoritmo **boosting** aplicável na construção de árvores para regressão. Veja Hastie et al. (2017, cap. 10) para detalhes.

Referências

- Breiman, L. (2001). Random forests. *Machine Learning*, **45**, 5–32.
- Chambers, J. M and Hastie, T. J. (1992:). *Statistical Models in S*. California: Wadsworth and Brooks/Cole.
- Hastie, T. J. and Tibshirani, R. J. (1990). *Generalized Additive Models*. London: Chapman and Hall.
- James, G., Witten, D., Hastie, T. and Tibshirani, R. (2017). *Introduction to Statistical Learning*. Springer.
- Morettin, P. A. e Singer, J. M. (2022). *Estatística e Ciência de Dados*. LTC, Rio de Janeiro.

MAE 5905: Introdução à Ciência de Dados

Pedro A. Morettin

Instituto de Matemática e Estatística
Universidade de São Paulo
pam@ime.usp.br
<http://www.ime.usp.br/~pam>

Aula 12

18 de maio de 2023

Sumário

- 1 Análise de Agrupamentos
- 2 Estratégias de agrupamento
- 3 Algoritmos hierárquicos
- 4 Algoritmos de partição: K-médias
- 5 AA-Tópicos Adicionais
- 6 Outras distâncias

Preliminares

- Na Análise de Agrupamentos (AA), o objetivo é agrupar “pontos” pertencentes a determinado espaço em “grupos” de acordo com alguma medida de distância, de modo que pontos num mesmo grupo tenham uma pequena distância entre eles.
- A AA é, às vezes, chamada de segmentação de dados. O espaço mencionado acima pode ser um espaço Euclidiano, eventualmente de dimensão grande, ou pode ser um espaço não Euclidiano, por exemplo quando queremos agrupar documentos segundo certos tópicos.
- O problema da AA insere-se naquilo que chamamos anteriormente de aprendizado não supervisionado, ou seja, temos apenas um conjunto de variáveis preditoras (inputs), não há uma variável resposta, e o objetivo é descrever associações e padrões entre essas variáveis.
- Para alcançar nosso objetivo, podemos usar várias técnicas de agrupamento, e nesta aula iremos discutir algumas delas.

Preliminares

A AA é usada em muitas áreas e aplicações, como:

- i) segmentação de imagens como em fMRI (imagens por ressonância magnética funcional), em que se pretende particionar a imagem em áreas de interesse. Veja, por exemplo, Sato et al. (2007);
- ii) bioinformática, por exemplo na análise de expressão de genes gerados de *microarrays* ou sequenciamento de DNA ou proteínas. Veja Hastie et al. (2009) ou Fujita et al. (2007), por exemplo.
- iii) reconhecimento de padrões, de objetos e caracteres, por exemplo identificação de textos escritos a mão em linguística;
- iv) redução (ou compressão) de grandes conjuntos de dados, a fim de escolher grupos de dados de interesse.

AA-Exemplo 1

Exemplo 1. Consideremos as medidas das variáveis altura (X_1 , em cm), peso (X_2 , em kg), idade (X_3 , em anos) e sexo (X_4 , M ou F) em 12 indivíduos dispostas na Tabela 1.

Tabela 1: Dados de 12 indivíduos

Ind.	Altura	Peso	Idade	Sexo	Alt.Padr.	Peso Padr.	Id. Padr.
A	180	75	30	M	0,53	0,72	0,38
B	170	70	28	F	-0,57	-0,13	-0,02
C	165	65	20	F	-1,12	-0,97	-1,61
D	175	72	25	M	-0,02	0,21	-0,61
E	190	78	28	M	1,63	1,23	-0,02
F	185	78	30	M	1,08	1,23	0,38
G	160	62	28	F	-1,67	-1,48	-0,02
H	170	65	19	F	-0,57	-0,97	-1,81
I	175	68	27	M	-0,02	-0,47	-0,22
J	180	78	35	M	0,53	1,23	1,38
K	185	74	35	M	1,08	0,55	1,38
L	167	64	32	F	-0,90	-1,14	0,78
μ	175,17	70,75	28,08	—	0	0	0
σ	9,11	5,91	5,02	—	1	1	1

AA-Exemplo 1

- Nessa tabela, a média de cada variável é indicada por μ e o desvio padrão por σ . As colunas 6, 7 e 8 trazem as variáveis padronizadas, ou seja,

$$Z_i = \frac{X_i - \mu_{X_i}}{\sigma_i}, \quad (1)$$

para $i=1,2,3$

- Para a variável X_4 (sexo), esta padronização, é claro, não faz sentido. A Figura 1 apresenta um gráfico de dispersão de X_3 versus X_2 .

AA-Exemplo 1

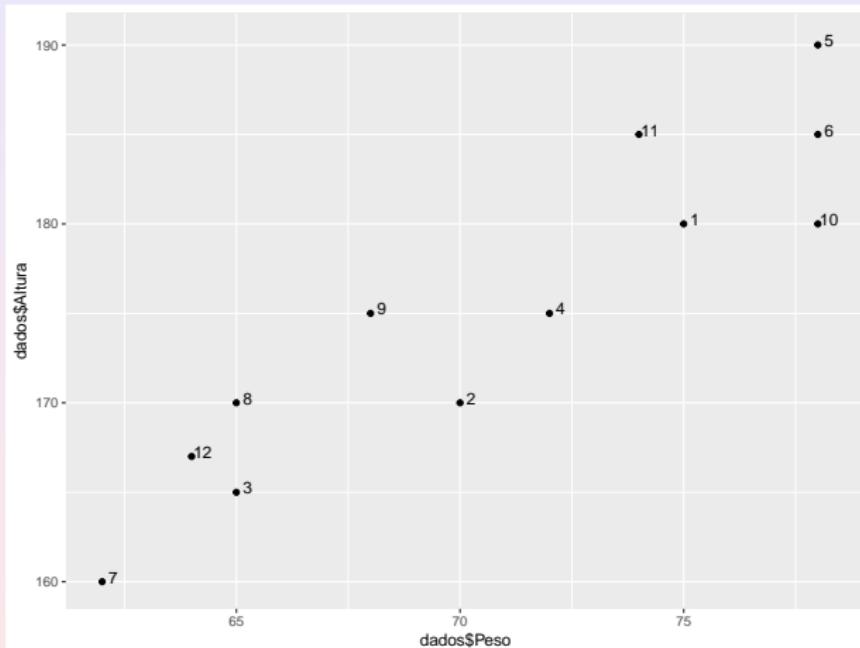


Figura 1: Gráfico de Altura *versus* Peso para os dados da Tabela 1, A=1, B=2 etc.

AA-Exemplo 1

- O objetivo é agrupar pontos que estejam próximos. Para isso, definamos a **distância Euclidiana** entre dois pontos $\mathbf{x} = (x_1, x_2)$ e $\mathbf{y} = (y_1, y_2)$ como

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}.$$

- Nesse caso, a dimensão do espaço é dois. No caso geral de um espaço Euclidiano p -dimensional a distância é definida por

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(x_1 - y_1)^2 + \dots + (x_p - y_p)^2}.$$

- Além da distância Euclidiana, podemos usar outras distâncias, como

$$d_1 = |x_1 - y_1| + \dots + |x_p - y_p| : \text{distância } L_1 \text{ ou Manhattan}, \quad (2)$$

$$d_2 = \max_{1 \leq i \leq p} |x_i - y_i| : \text{distância } L_\infty. \quad (3)$$

- Para espaços não Euclidianos, há outras definições de distância, como **Hamming**, **cosseno**, **Jaccard**, **edit** etc. (Ver Notas de Capítulo).

AA–Exemplo 1

- Podemos obter a **matriz de similaridade** dos dados, segundo dada distância.
- Consideremos n indivíduos, para os quais observamos p variáveis. A matriz de similaridade será uma matriz $n \times n$, $D = [d(i,j)]_{i,j=1}^n$, simétrica, com zeros na diagonal principal (Pode-se considerar uma matriz reduzida, de ordem $(n - 1) \times n$, para evitar os zeros).
- Na Tabela 2 temos essa matriz para os dados da Tabela 5, usando a distância Euclidiana. Algumas distâncias em ordem crescente são:

$$\begin{aligned}d(C, L) &= 2,24, \\d(A, J) &= 3,00, \\d(H, L) &= 3,16, \\d(D, I) &= 4,00, \\d(F, K) &= 4,00, \\d(B, H) &= 5,00, \\d(C, H) &= 5,00, \\d(E, F) &= 5,00, \\d(F, J) &= 5,00, \\d(A, K) &= 5,10.\end{aligned}$$

AA-Exemplo 1

Essas distâncias (indicadas em negrito) nos dão uma ideia de como agrupar pontos.

Tabela 2: Matriz de similaridade, distância Euclidiana

	A	B	C	D	E	F	G	H	I	J	K
B	11,18										
C	18,03	7,07									
D	5,83	5,38	12,21								
E	10,44	21,54	28,18	16,16							
F	5,83	17,00	23,85	11,66	5,00						
G	23,85	12,81	5,83	18,03	34,00	29,68					
H	14,14	5,00	5,00	8,60	23,85	19,85	10,44				
I	8,60	5,39	10,44	4,00	18,03	14,14	16,16	5,83			
J	3,00	12,81	19,85	7,81	10,00	5,00	25,61	16,40	11,18		
K	5,10	15,52	21,93	10,20	6,40	4,00	27,73	17,49	11,66	6,40	
L	17,03	6,71	2,24	11,31	26, 93	22,80	7,28	3,16	8,94	19,10	20,59

Tipos de algoritmos

Podemos dividir os algoritmos de agrupamento em três grupos (Hastie et al., 2009):

- a) **combinatórios**: trabalham diretamente com os dados, não havendo referência à sua distribuição;
- b) **baseados em modelos**: supõem que os dados sejam uma amostra aleatória simples de uma população, com uma densidade de probabilidade, que é uma mistura de densidades componentes, cada uma descrevendo um grupo;
- c) **bump hunters** : tratam de estimar, de modo não paramétrico, as modas da densidade. As observações mais próximas a cada moda definem os grupos.

Tipos de algoritmos

Por sua vez, os algoritmos combinatórios podem ser classificados em dois grupos:

- i) **algoritmos hierárquicos**: que ainda podem ser subdivididos em **aglomerativos** e **divisivos**. No primeiro caso, iniciamos o procedimento de modo que cada ponto forma um grupo e vamos combinando grupos com base em suas proximidades, usando alguma definição de proximidade (como uma distância). Paramos quando fixamos um número de grupos ou obtemos grupos que não são desejáveis, por alguma razão. No segundo caso, partimos de um único grupo e por divisões sucessivas obtemos 1,2 etc. grupos. Neste texto, usaremos o método aglomerativo.
- ii) **algoritmos de partição** (ou obtidos por associação de pontos): os grupos obtidos formam uma partição do conjunto total de pontos. Os pontos são considerados em alguma ordem e cada um deles é associado ao grupo no qual ele melhor se ajusta. O método chamado de **K-médias** pertence a esse grupo de algoritmos.

Tipos de algoritmos

- Se estivermos num espaço Euclíadiano, quando usamos alguma distância entre os pontos, grupos podem ser caracterizados pelo seu **centróide** (média das coordenadas dos pontos).
- Num espaço não Euclíadiano, não existe a noção de centróide, e deveremos usar alguma outra maneira de caracterizar os grupos.
- Se o conjunto de pontos for muito grande, há o que se chama de **maldição da dimensionalidade** (*curse of dimensionality*). Em grandes dimensões, **quase** todos os pares de pontos têm a mesma distância entre si e quaisquer dois vetores são **quase** sempre ortogonais.

AH–Exemplo 1

- Para ilustrar o método, vamos voltar ao Exemplo 1 e procuremos os pontos mais próximos, baseando-nos na Figura 1.
- Consideremos as variáveis X_1 (altura) e X_2 (peso) e o gráfico da Figura 1. Temos um espaço Euclidiano de dimensão 2 e a proximidade entre dois pontos medida pela distância Euclidiana (DE).
- Cada grupo será representado pelo seu centróide e a regra de agrupamento será, pois: calcular a distância Euclidiana entre os centróides de dois grupos quaisquer e escolher, para agrupar, os dois grupos com a menor DE.

AH-Exemplo 1

- i) Inicialmente, cada ponto é considerado um grupo, que coincide com seu centróide.
- ii) Consultando a Tabela 2, $C=(65, 165)$ e $L=(64; 167, 64)$ são os mais próximos com DE $d(C, L) = \sqrt{5,00} = 2,24$ e centróide $c(C, L) = (64, 5; 166)$. Esses dois pontos formam o primeiro agrupamento, $\mathcal{G}_1 = \{C, L\}$. A distância entre esses pontos, 2,24, será chamada **nível do agrupamento ou junção**.
- iii) A seguir, teríamos que recalcular as distâncias entre os pontos, considerando agora os grupos A, B, CL, D, E, F, G, H, I, J, K. A maioria das distâncias não muda, mudando somente as distâncias dos pontos ao centróide de C e L. Pela Tabela 2, a DE de $A = (75, 180)$ a $J = (78, 180)$ é 3, logo agrupamos A e J, formando o grupo $\mathcal{G}_2 = \{A, J\}$, com centróide $c(\mathcal{G}_2) = (76, 5, 180)$ e nível 3,00.
- iv) Agora, DE entre $D = (72, 175)$ e $I = (68, 175)$ é 4,00, obtendo-se o agrupamento, $\mathcal{G}_3 = \{D, I\}$, com centróide $c(\mathcal{G}_3) = (70, 175)$ e nível 4,00.

AH–Exemplo 1

- v) A seguir, os pontos mais próximos são $F = (78, 185)$ e $K = (74, 185)$, com DE igual a 4,00, obtendo-se $\mathcal{G}_4 = \{F, K\}$, centróide $c(\mathcal{G}_4) = (76, 185)$ e nível 4,00.
- vi) Consideramos o ponto $H = (65, 170)$, cuja DE ao centróide de C e L é $d(H, c(\mathcal{G}_1)) \approx 4,03$. Portanto, obtemos outro agrupamento, $\mathcal{G}_5 = \{C, H, L\}$, com centróide $c(\mathcal{G}_5) = (64, 7; 167, 3)$ e nível 4,03.
- vii) A seguir, agrupamos $B = (70, 170)$ com $\mathcal{G}_3 = \{D, I\}$, notando que a DE entre B e o centróide desse grupo é 5,00, passando a ser esse valor o nível da junção. O novo grupo é $\mathcal{G}_6 = \{B, D, I\}$, com centróide $c(\mathcal{G}_6) = (70; 173, 3)$.
- viii) Agrupamos, agora, os grupos $\mathcal{G}_2 = \{A, J\}$ e $\mathcal{G}_4 = \{F, K\}$, com DE entre seus centróides de 5,02, formando-se o grupo $\mathcal{G}_7 = \{A, F, J, K\}$, com centróide $c(\mathcal{G}_7) = (76, 25; 182, 5)$ e nível 5,02.

AH–Exemplo 1

- ix) Agregamos, a seguir, o ponto $G = (62, 160)$ ao grupo $\mathcal{G}_5 = \{C, H, L\}$, obtendo-se o novo grupo $\mathcal{G}_8 = \{C, G, H, L\}$, com centróide $c(\mathcal{G}_8) = (64; 165, 5)$ e nível 7,31.
- x) Finalmente, agrupamos o ponto $E = (78, 190)$ ao grupo $\mathcal{G}_7 = \{A, F, J, K\}$, obtendo-se o grupo $\mathcal{G}_9 = \{A, E, F, J, K\}$, com centróide $c(\mathcal{G}_9) = (76, 6; 184)$ e nível 7,50.

A Figura 2 mostra esses agrupamentos. Podemos prosseguir, agrupando-se dois desses grupos (aqueles que possuem a menor DE entre os respectivos centróides) e, finalmente, agrupar os dois grupos restantes.

AH-Exemplo 1

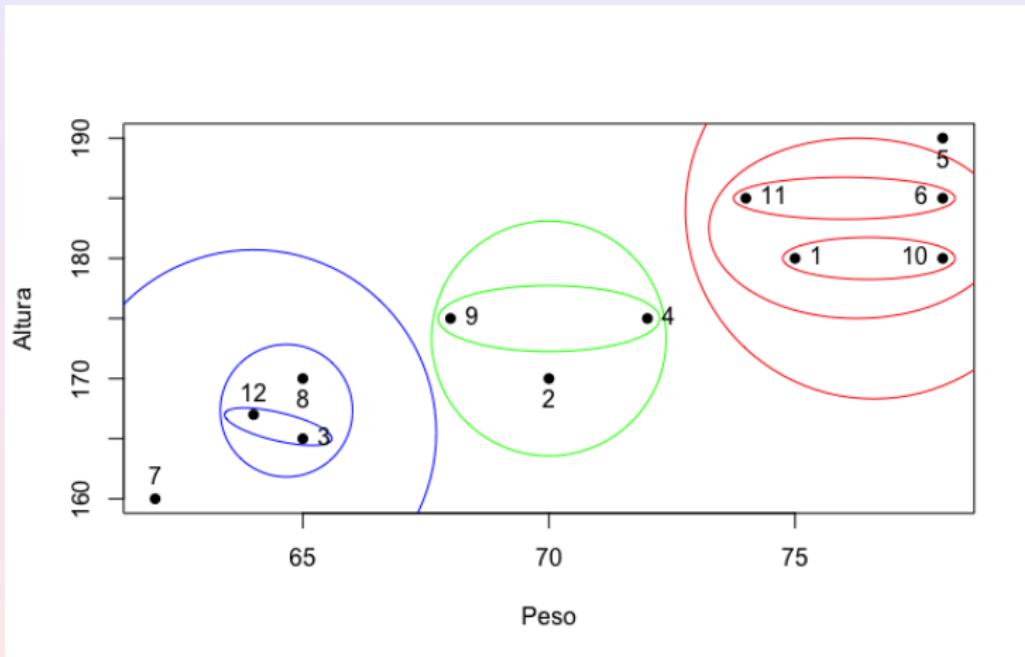


Figura 2: Agrupamentos obtidos para o Exemplo 1.

AH-Dendrograma

- Um gráfico que sumariza o procedimento ([dendrograma](#)) está na Figura 3. No eixo vertical da figura colocamos os níveis, no horizontal, os pontos de modo conveniente. Nessa figura, usamos a distância Euclidiana.
- Na Tabela 3 temos um resumo do método hierárquico, usando distância Euclidiana e centróides, para agrupar os pontos, para o Exemplo 1.

Tabela 3: Resumo do procedimento de agrupamento para o Exemplo 1

Passo	Agrupamento	Nível
1	C, L	2,24
2	A, J	3,00
3	D, I	4,00
4	F, K	4,00
5	H, CL	4,03
6	B, DI	5,00
7	AJ, KF	5,02
8	G, CHL	7,31
9	E, AFJK	7,50

AH-Dendrograma

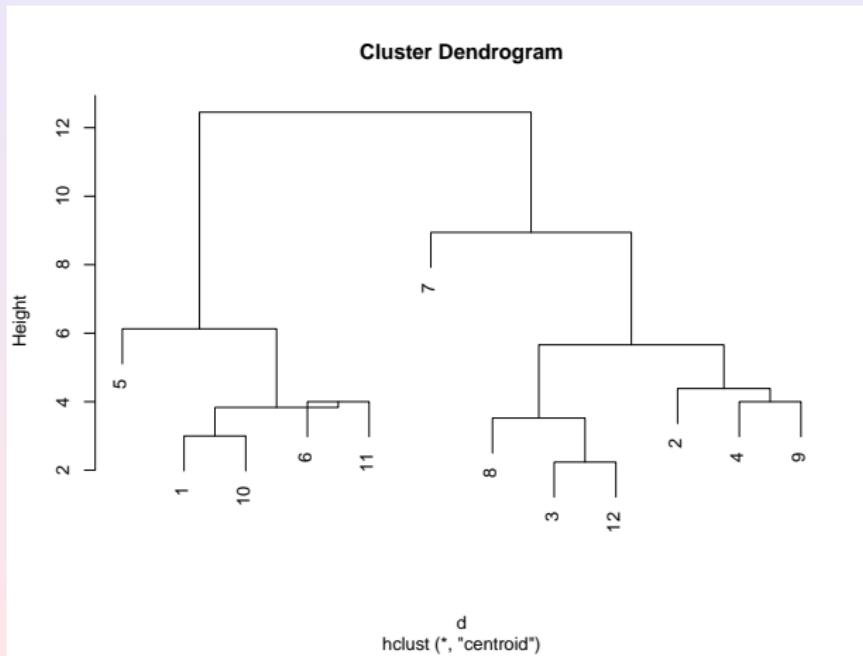


Figura 3: Dendrograma para o Exemplo 1.

AH–Interpretação

- Interpretando o resultado, com esses três grupos, vemos que o primeiro contém as pessoas menos pesadas e mais baixas (4 pessoas), depois aquele que contém pessoas com pesos e alturas intermediárias (3 pessoas) e, finalmente, o grupo que contém as pessoas mais pesadas e mais altas (5 pessoas).
- Se esse for objetivo, podemos parar aqui. Se o objetivo é obter dois grupos, um com pessoas mais baixas e menos pesadas e, outro, com pessoas mais altas e mais pesadas, continuamos a agrupar mais uma vez, obtendo os grupos $\mathcal{G}_{10} = \{B, C, D, G, H, I, L\}$ e \mathcal{G}_9 .
- Um dos objetivos da construção de grupos é **classificar** um novo indivíduo em algum dos grupos obtidos. O problema da classificação está intimamente ligado ao problema de AA, e já foi tratado em capítulos anteriores.
- Um pacote do repositório R que pode ser usado é o **cluster**. Após carregar o pacote em sua área por meio de `library(cluster)`, temos que informar a distância (`euclidian`, `maximum`, `manhattan` etc.) a usar e o método de agrupamento (`centroid`, `average`, `median` etc.). O pacote contém várias funções para mostrar em que grupo estão as unidades, obter o dendrograma, fornecer a ordem para fazer o dendrograma etc.

K-médias

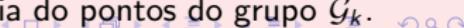
- O método de K-médias tem por objetivo partitionar os pontos em K grupos, de tal modo que a soma dos quadrados das distâncias dos pontos aos centros dos agrupamentos (**clusters**) seja minimizada. É um método baseado em centróides, como vimos anteriormente, e pertence à classe de algoritmos (ii) discutida antes, e requer que o espaço seja Euclidiano.
- Usualmente, o valor de K é conhecido e deve ser fornecido pelo usuário mas é possível obtê-lo por tentativa e erro.

O algoritmo mais comum é devido a Hartigan and Wong (1979) é usado como *default* em pacotes computacionais. Outros algoritmos são os de MacQueen (1967), Lloyd (1957) eForgy (1965).

- A função **kmeans** do pacote **cluster** pode ser utilizada.
- A ideia básica consiste em definir grupos em que a variação interna seja minimizada. Esta é, em geral, definida como a soma das DE ao quadrado entre pontos e o centróide correspondente:

$$W(\mathcal{G}_k) = \sum_{x_i \in \mathcal{G}_k} (x_i - \mu_k)^2, \quad (4)$$

em que x_i é um ponto no grupo \mathcal{G}_k e μ_k é a média do pontos do grupo \mathcal{G}_k .



K-médias

O algoritmo consiste nos seguintes passos:

- i) especifique K e selecione K pontos que pareçam estar em diferentes grupos;
- ii) considere esses pontos como os centróides iniciais desses grupos;
- iii) associe cada observação ao centróide mais próximo, baseado na distância Euclidiana entre essa e o centróide;
- iv) para cada um dos K grupos, recalcule o centróide após cada ponto ser incluído.
- v) iterativamente, minimize a soma total de quadrados dentro dos grupos, até que os centróides não mudem muito (o R usa 10 iterações como *default*). A soma total de quadrados dentro dos grupos é definida por

$$\sum_{j=1}^K W(\mathcal{G}_j) = \sum_{j=1}^K \sum_{x_i \in \mathcal{G}_j} (x_i - \mu_j)^2. \quad (5)$$

K-médias: Exemplo 2

Exemplo 2. Consideremos 100 simulações de duas variáveis com distribuição normal, uma com média 0 e desvio padrão 0,3 e outra, com média 1 e desvio padrão também 0,3. Na Figura 4 apresentamos os dois grupos com os respectivos centros, resultantes da aplicação do algoritmo **K-means** com $K = 2$.

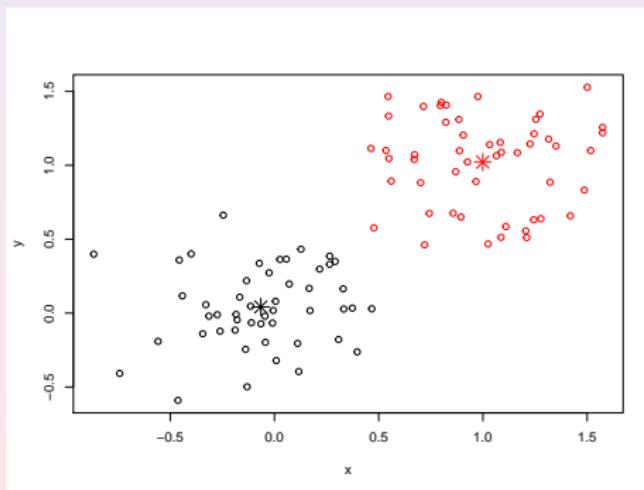


Figura 4: Uso do pacote kmeans para o exemplo simulado

K-médias: Exemplo 3

- **Exemplo 3.**: Consideremos os dados da Tabela 1 e as variáveis Peso e Altura. Usando o resultado do procedimento hierárquico, suponha que tenhamos $K = 3$ grupos.
- Usando a função `kmeans`, obtemos o gráfico da Figura 5. Nessa figura temos os três grupos (com tamanhos 3,4 e 5) em diferentes cores e os centros de cada grupo.
- Esses centróides são dados pelo programa, $(63, 67; 164)$, $(68, 75; 172, 5)$ e $(76, 60; 184, 0)$. As somas de quadrados dentro dos grupos são 30,67, 51,75 e 85,20, respectivamente. A soma de quadrados total entre grupos é 87,1.

K-médias: Exemplo 3

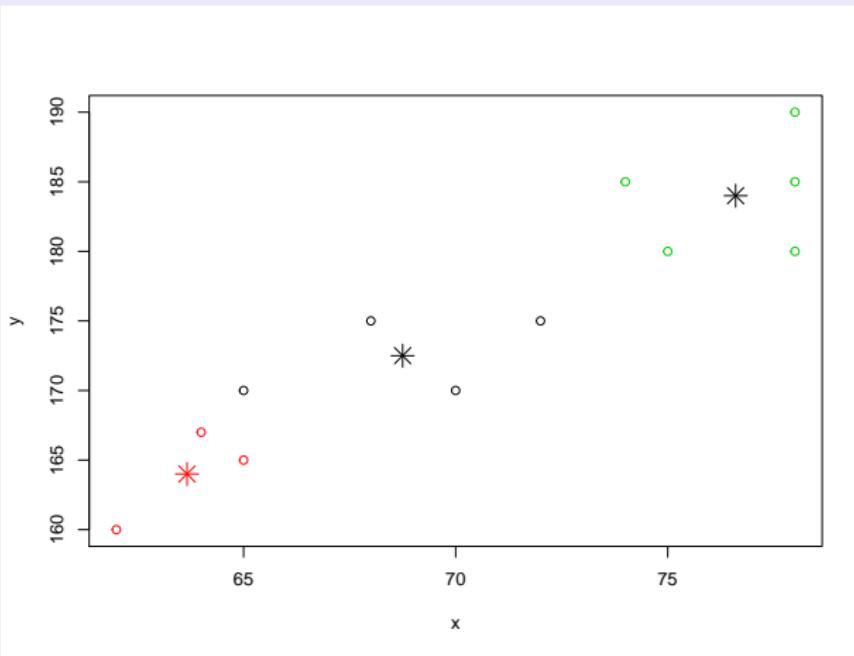


Figura 5: Uso do pacote `kmeans` para o Exemplo 3.

K-médias: Exemplo 4

- **Exemplo 4:** Vamos considerar os dados de um conjunto de 4.000 motoristas encarregados de fazer entregas de determinados produtos. Há várias variáveis envolvidas, mas iremos considerar somente duas, nomeadamente, X_1 : distância média percorrida por cada motorista (em milhas) e X_2 : porcentagem média do tempo em que o motorista esteve acima do limite de velocidade por mais de 5 milhas por hora. Há dados do setor urbano e rural.
- Os dados podem ser obtidos de
https://raw.githubusercontent.com/datasets/introduction-to-k-means-Clustering/master/Data/data_1024.csv.
- Na Figura 6 apresentamos um diagrama de dispersão dos dados, segundo essas duas variáveis, mostrando claramente dois grupos distintos: grupo 1, contendo os motoristas que fazem entregas no setor urbano e grupo 2, contendo motoristas que fazem entregas no setor rural.

K-médias: Exemplo 4

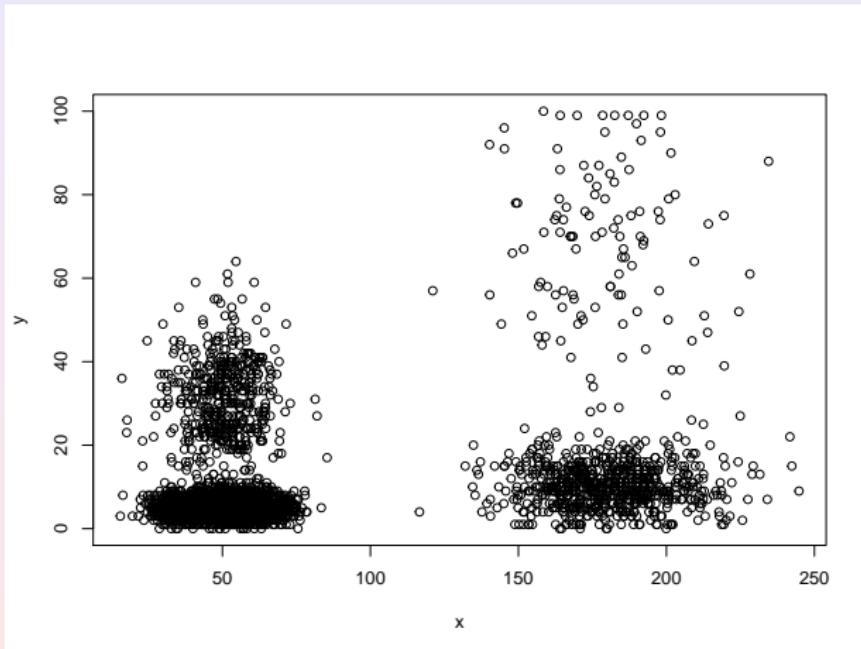


Figura 6: Gráfico de dispersão de X_1 versus X_2 para o Exemplo 4.

K-médias: Exemplo 4

Utilizando o algoritmo K-médias com $K = 2$, obtemos:

Grupo 1: Centróide = (50,05, 8,83)

Grupo 2: Centróide = (180,02, 18,29)

Na Figura 7 temos os dois grupos representados.

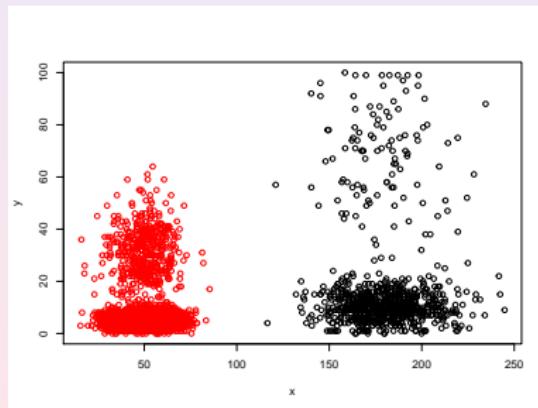


Figura 7: Grupos para o Exemplo 4 com $K=2$.

K-médias: Exemplo 4

Se tomarmos $K = 4$, obtemos o gráfico da Figura 8. Agora, os motoristas são separados por aqueles que seguem ou não a velocidade limite, além da divisão zona urbana/rural.

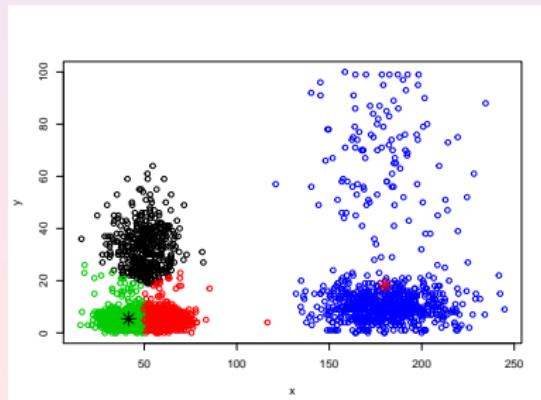
Os centróides são:

Grupo 1: (50,61, 33,06),

Grupo 2: (57,90, 5,28),

Grupo 3: (41,52, 5,40),

Grupo 4: (180,10, 18,31).



K-médias: Exemplo 5

- **Exemplo 5:** Vamos, agora, considerar dados do Uber, na cidade de Nova Iorque (NYC). Esses dados podem ser obtidos no site

[www.kaggle.com/fivethirtyeight/
uber-pickups-in-new-york-city/
data](http://www.kaggle.com/fivethirtyeight/uber-pickups-in-new-york-city/data)

e contém cerca de 4,5 milhões de corridas do Uber de abril a setembro de 2014, além de outros dados do Uber de 2015 e outras de companhias.

- NYC tem 5 distritos: Brooklyn, Queens, Manhattan, Bronx e Staten Island. O conjunto de dados que vamos usar, de 2014, tem informação detalhada sobre a localização do início da corrida com as seguintes colunas:
Date/Time: dia e hora do início da corrida;
Lat: a latitude da localidade;
Lon: a longitude da localidade;
Base: o código da base da companhia afiliada àquela corrida.
- Os nomes dos arquivos são da forma `uber-raw-data-month.csv`, em que **month** deve ser substituído por `apr14`, `aug14`, `jul14`, `jun14`, `may14`, `sept14`. Em nosso exemplo, vamos usar somente os dados de abril de 2014.

K-médias: Exemplo 5

- Para ler os dados usamos o comando:

```
> read.csv("https://raw.githubusercontent.com/fivethirtyeight/  
uber-tlc-foil-response/master/uber-trip-data/uber-raw-data-apr14.csv")
```

- Iremos usar o pacote **kmeans** do R e, dependendo do que se quer, outros pacotes, como **dplyr**, **VIM**, **lubridate** ou **ggmap**, poderão ser empregados.
- Com o comando **summary(apr14)**, obtemos:

```
summary(apr14)
      Date.Time           Lat             Lon
4/7/2014 20:21:00 :   97   Min.   :40.07   Min.   :-74.77
4/7/2014 20:22:00 :   87   1st Qu.:40.72   1st Qu.:-74.00
4/30/2014 17:45:00:   78   Median :40.74   Median :-73.98
4/30/2014 18:43:00:   70   Mean    :40.74   Mean   :-73.98
4/30/2014 19:00:00:   70   3rd Qu.:40.76   3rd Qu.:-73.97
4/30/2014 16:55:00:   67   Max.    :42.12   Max.   :-72.07
(Other)                  :564047
```

K-médias: Exemplo 5

Para ver os dados correspondentes (até dia), temos:

```
head(apr14, n=10)
  Date.Time      Lat      Lon     Base Year Month Day
1 2014-04-01 00:11:00 40.7690 -73.9549 B02512 2014     4    1
2 2014-04-01 00:17:00 40.7267 -74.0345 B02512 2014     4    1
3 2014-04-01 00:21:00 40.7316 -73.9873 B02512 2014     4    1
4 2014-04-01 00:28:00 40.7588 -73.9776 B02512 2014     4    1
5 2014-04-01 00:33:00 40.7594 -73.9722 B02512 2014     4    1
6 2014-04-01 00:33:00 40.7383 -74.0403 B02512 2014     4    1
7 2014-04-01 00:39:00 40.7223 -73.9887 B02512 2014     4    1
8 2014-04-01 00:45:00 40.7620 -73.9790 B02512 2014     4    1
9 2014-04-01 00:55:00 40.7524 -73.9960 B02512 2014     4    1
10 2014-04-01 01:01:00 40.7575 -73.9846 B02512 2014     4    1
```

K-médias: Exemplo 5

Finalmente, usando o `kmeans` e `ggmap` obtemos a Figura 34. Detalhes dos scripts necessários, estão na página do livro.

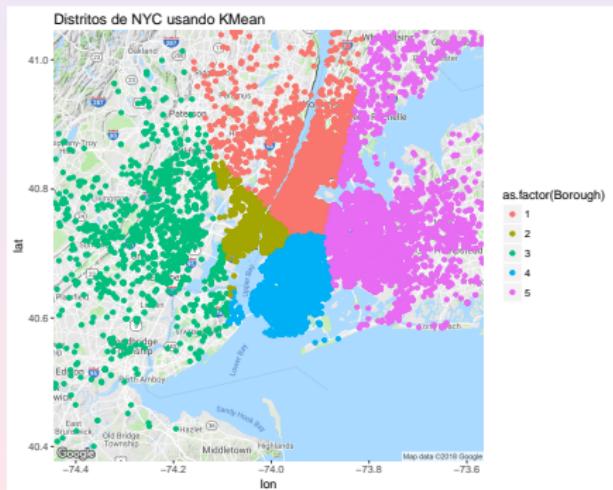


Figura 9: Grupos obtidos para o Exemplo 5.

Matriz Cofenética

- Esta matriz, **S**, digamos, contém as distâncias entre os objetos a partir do dendograma. Por exemplo, a distância entre os pontos C e L é dada pelo nível em que os dois foram agrupados, nesse caso, 2,24.
- A seguir, verificamos a proximidade entre essa matriz e a matriz de proximidade, **D**.
- Essa proximidade é dada pelo coeficiente de correlação entre os valores de **D** e de **S**, chamado de **coeficiente de correlação cofenético**.
- No Exemplo 12.1, esse valor é 0,80, e pode ser considerado um valor adequado.

AA–Outros algoritmos

Vimos exemplos com duas variáveis. Podemos ter mais variáveis, mas seu número deve ser menor que o número de indivíduos. Quando temos mais do que duas dimensões, alguns algoritmos foram propostos e são, basicamente, variantes de algoritmos hierárquicos e de K -means.

- Algoritmo BFR** [(Bradley, Fayyad and Reina, (1998)]. Esse algoritmo é uma variante do K -means, para o caso de um espaço Euclidiano de alta dimensão mas é baseado numa suposição muito forte: a forma dos agrupamentos segue uma distribuição normal, em cada dimensão, ao redor do respectivo centróide, e além disso as dimensões são independentes. Então, a forma dos grupos deve ser uma elipsóide, com os eixos paralelos aos eixos da dimensão, podendo eventualmente ser um círculo, mas não podem, por exemplo, terem os eixos do elipsóide oblíquos aos eixos da dimensão ou terem formas mais complicadas. Esse algoritmo não está contemplado no R.
- Algoritmo CURE** (de *Clustering Using Representatives*). Esse algoritmo usa procedimentos hierárquicos e os grupos podem ter quaisquer formas mas também não está disponível no R. No entanto, ele pode ser obtido no pacote **pyclustering**, que usa as linguagens Python e C++.

AA–Outros algoritmos

- c) **Density-based algorithms** (DBSCAN): dado um conjunto de pontos em algum espaço, esse algoritmo agrupa pontos que estão dispostos em uma vizinhança com maior densidade, colocando como *outliers* os pontos que estão em regiões de baixa densidade. Veja Ester et al. (1996).

AA-distância para strings

- Como salientamos anteriormente, para espaços não Euclidianos (ENE) há outras formas de distância.
- No caso de sequências (**strings**) $x = x_1x_2 \cdots x_n$ e $y = y_1y_2 \cdots y_n$, uma distância conveniente é a distância **edit**, que dá o número de inserções e exclusões de caracteres que converterão x em y .
- Por exemplo, considere $x = abcd$ e $y = acde$. A distância **edit** entre essas sequências é $d_e(x, y) = 2$, pois temos que excluir b e inserir e depois do d .
- Uma outra maneira de obter essa distância é considerar uma subsequência mais longa comum (SML) a x e y . No exemplo, é acd . Então, a distância **edit** é dada por

$$d_e(x, y) = \ell(x) + \ell(y) - 2\ell(\text{SML}),$$

em que ℓ indica o comprimento de cada sequência. No exemplo,
 $d_e(x, y) = 4 + 4 - 2 \times 3 = 2$.

AA-distância para strings

- Outra distância que pode ser usada é a **distância Hamming**, que dá o número de componentes em que dois vetores (de mesma dimensão) diferem. Por exemplo, se $x = 110010$ e $y = 100101$, então a distância Hamming entre eles é $d_H(x, y) = 4$.
- Um problema em ENE é representar um grupo, pois não podemos, por exemplo, calcular o centróide de dois pontos. No exemplo acima, como obter uma sequência entre x e y ? Usando a distância *edit* poder-se-ia selecionar algo parecido ao centróide (**o grupóide**), escolhendo-se a sequência que minimiza, por exemplo, a soma das distâncias dessa com as outras sequências do grupo previamente selecionado.

AA-algoritmos aglomerativos

Considere os algoritmos hierárquicos e dois grupos quaisquer, A e B , e a distância entre eles, $d(A, B)$. Como vimos, comumente usamos algoritmos aglomerativos que ainda podem ser divididos em (Hastie et al., 2009):

- a) algoritmos com ligação simples (**single linkage**), para os quais toma-se a distância mínima entre os pares, ou seja,

$$d_{SL}(A, B) = \min_{i \in A, j \in B} d(i, j). \quad (6)$$

Essa técnica é também conhecida como técnica do **vizinho mais próximo**;

- b) algoritmos com ligação completa (**complete linkage**), para os quais toma-se a máxima distância entre os pares:

$$d_{CL}(A, B) = \max_{i \in A, j \in B} d(i, j); \quad (7)$$

AA-algoritmos aglomerativos

- c) algoritmos com ligação média (**group average**), que toma a distância média entre os grupos:

$$d_{GA}(A, B) = \frac{1}{N_A N_B} \sum_{i \in A} \sum_{j \in B} d(i, j). \quad (8)$$

Aqui, N_A e N_B indicam os números de observações em cada grupo.

Ligações simples produzem grupos com diâmetros grandes e ligações completas produzem grupos com diâmetros pequenos; agrupamentos por médias representam um compromisso entre esses dois extremos.

Referências

- Hastie, T., Tibshirani, R. and Friedman, J. (2017). *The Elements of Statistical Learning*, 2nd Edition, Springer.
- James, G., Witten, D., Hastie, T. and Tibshirani, R. (2017). *Introduction to Statistical Learning*. Springer.
- Morettin, P. A. e Singer, J. M. (2022). *Estatística e Ciência de Dados*. LTC: Rio de Janeiro.

MAE 5905: Introdução à Ciência de Dados

Pedro A. Morettin

Instituto de Matemática e Estatística
Universidade de São Paulo
pam@ime.usp.br
<http://www.ime.usp.br/~pam>

Aula 13

23 de maio de 2023

Sumário

1 Redução da dimensionalidade

2 Análise de componentes principais

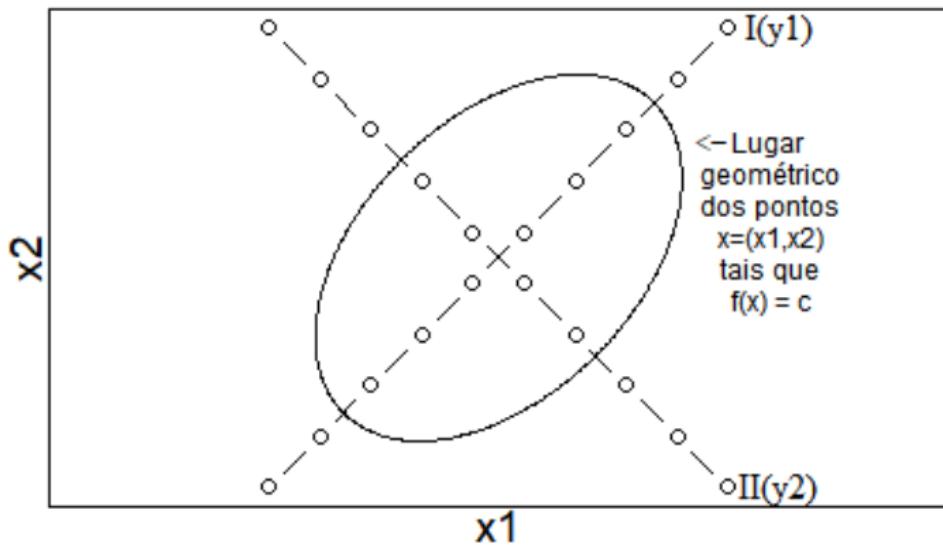
Preliminares

- As técnicas de Análise de Componentes Principais (ACP), Análise de Fatorial (AF) e Análise de Componentes Independentes (ACI) têm como objetivo reduzir a dimensionalidade de observações multivariadas com base em sua estrutura de dependência.
- A ideia que as permeia é a obtenção de poucos fatores, obtidos como funções das características observadas, que conservem, pelo menos aproximadamente, a estrutura de covariância das variáveis originais.
- Esses poucos fatores vão substituir as variáveis originais em análises subsequentes, servindo, por exemplo, como variáveis explicativas em modelos de regressão. Por esse motivo, a interpretação dessas novas variáveis é muito importante.
- Dessas técnicas, ACP e AF são bastante conhecidas há muito tempo. A ACI (ou ICA, em Inglês) é mais recente, da década de 1990–2000, e não muito contemplada em textos de Estatística.

ACP – introdução

- A técnica de Componentes Principais consiste numa transformação ortogonal dos eixos de coordenadas de um sistema multivariado.
- A orientação dos novos eixos é determinada por meio da partição sequencial da variância total das observações em porções cada vez menores de modo que, ao primeiro eixo transformado, corresponda o maior componente da partição da variância; ao segundo eixo transformado, a parcela seguinte e assim por diante.
- Se os primeiros eixos forem tais que uma grande parcela da variância seja explicada por eles, poderemos desprezar os demais e trabalhar apenas com os primeiros em análises subsequentes.
- Consideremos duas variáveis X_1 e X_2 com distribuição normal bivariada com média μ e matriz de covariâncias Σ .
- O gráfico correspondente aos pontos em que a função densidade de probabilidade é constante é uma elipse; um exemplo está apresentado na Figura 1. Admitimos dados com distribuição normal apenas para finalidade didática. Em geral, essa suposição não é necessária.

ACP – introdução



ACP – metodologia

- À medida em que a correlação entre X_1 e X_2 aumenta, o comprimento do eixo maior da elipse também aumenta e o do eixo menor diminui até que a elipse se degenera em um segmento de reta no caso limite em que as variáveis são perfeitamente correlacionadas, i.e., em que o correspondente coeficiente de correlação linear é igual a 1.
- Na Figura 1, o eixo I corresponde ao eixo maior e o eixo II, ao menor. O eixo I pode ser expresso por intermédio de uma combinação linear de X_1 e X_2 , ou seja

$$Y_1 = \beta_1 X_1 + \beta_2 X_2 \quad (1)$$

- No caso extremo, em que X_1 e X_2 são perfeitamente correlacionadas, toda a variabilidade pode ser explicada por meio de Y_1 .
- Quando a correlação entre X_1 e X_2 não é perfeita, Y_1 explica apenas uma parcela de sua variabilidade. A outra parcela é explicada por meio de um segundo eixo, a saber

$$Y_2 = \gamma_1 X_1 + \gamma_2 X_2. \quad (2)$$

ACP – metodologia

Na Figura 2 apresentamos um gráfico de dispersão correspondente a n observações (X_{1i}, X_{2i}) do par (X_1, X_2) .

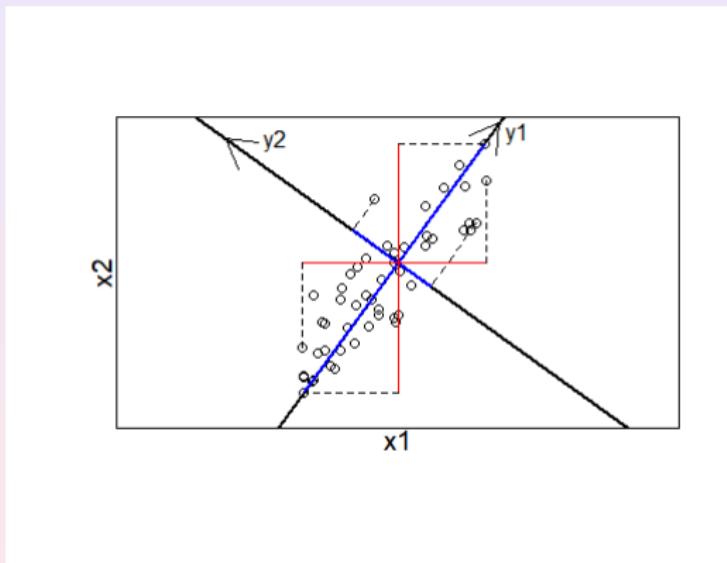


Figura: Gráfico de dispersão de n observações do par (X_1, X_2) .

ACP – metodologia

- A variabilidade no sistema de eixos correspondente a (X_1, X_2) pode ser expressa como

$$\sum_{i=1}^n (X_{1i} - \bar{X}_1)^2 + \sum_{i=1}^n (X_{2i} - \bar{X}_2)^2$$

em que \bar{X}_1 e \bar{X}_2 são, respectivamente, as médias dos n valores de X_{1i} e X_{2i} .

- No sistema de eixos correspondente a (Y_1, Y_2) , a variabilidade é expressa como

$$\sum_{i=1}^n (Y_{1i} - \bar{Y}_1)^2 + \sum_{i=1}^n (Y_{2i} - \bar{Y}_2)^2$$

em que \bar{Y}_1 e \bar{Y}_2 têm interpretações similares a \bar{X}_1 e \bar{X}_2 .

ACP – metodologia

- Se a correlação entre X_1 e X_2 for “grande”, é possível obter valores de β_1 , β_2 , γ_1 , γ_2 de tal forma que Y_1 e Y_2 em (1) e (2) sejam tais que

$$\sum_{i=1}^n (Y_{1i} - \bar{Y}_1)^2 \gg \sum_{i=1}^n (Y_{2i} - \bar{Y}_2)^2.$$

- Nesse caso, podemos utilizar apenas Y_1 como variável para explicar a variabilidade de X_1 e X_2 .
- Para descrever o processo de obtenção das componentes principais, consideremos o caso geral em que $\mathbf{x}_1, \dots, \mathbf{x}_n$ corresponde a uma amostra aleatória de uma variável $\mathbf{X} = (X_1, \dots, X_p)^\top$ com p componentes e para a qual o vetor de médias é μ e a matriz de covariâncias é Σ .

ACP – metodologia

- Sejam $\bar{\mathbf{x}} = n^{-1} \sum_{i=1}^n \mathbf{x}_i$ e $\mathbf{S} = (n-1)^{-1} \sum_{i=1}^n (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\top$, respectivamente, o vetor de médias amostrais e a matriz de covariâncias amostral.
- A técnica consiste em procurar sequencialmente p combinações lineares (denominadas **componentes principais**) de X_1, \dots, X_p tais que à primeira corresponda a maior parcela de sua variabilidade, à segunda, a segunda maior parcela e assim por diante e que, além disso, sejam não correlacionadas entre si.
- A primeira componente principal é a combinação linear

$$Y_1 = \boldsymbol{\beta}_1^\top \mathbf{X} = \beta_{11}X_1 + \dots + \beta_{1p}X_p$$

com $\boldsymbol{\beta}_1 = (\beta_{11}, \dots, \beta_{1p})^\top$, para a qual a variância $Var(Y_1) = \boldsymbol{\beta}_1^\top \boldsymbol{\Sigma} \boldsymbol{\beta}_1$ é máxima.

ACP – metodologia

- Uma estimativa da primeira componente principal calculada com base na amostra é a combinação linear $\widehat{Y}_1 = \widehat{\beta}_1^\top \mathbf{x}$ para a qual $\widehat{\text{Var}}(\widehat{Y}_1) = \widehat{\beta}_1^\top \mathbf{S} \widehat{\beta}_1$ é máxima.
- Este problema não tem solução sem uma restrição adicional, pois se tomarmos $\widehat{\beta}_1^* = c\widehat{\beta}_1$ com c denotando uma constante arbitrária, podemos tornar a variância $\widehat{\text{Var}}(\widehat{Y}_1) = \text{Var}(\widehat{\beta}_1^* \mathbf{x})$ arbitrariamente grande, tomando c arbitrariamente grande.
- A restrição adicional mais usada consiste em padronizar β_1 por meio de $\beta_1^\top \beta_1 = 1$.
- Consequentemente, o problema de determinação da primeira componente principal se resume a obter $\widehat{\beta}_1$ tal que

$$\text{Maximize } \beta_1^\top \Sigma \beta_1, \quad \text{sujeito a } \beta_1^\top \beta_1 = 1.$$

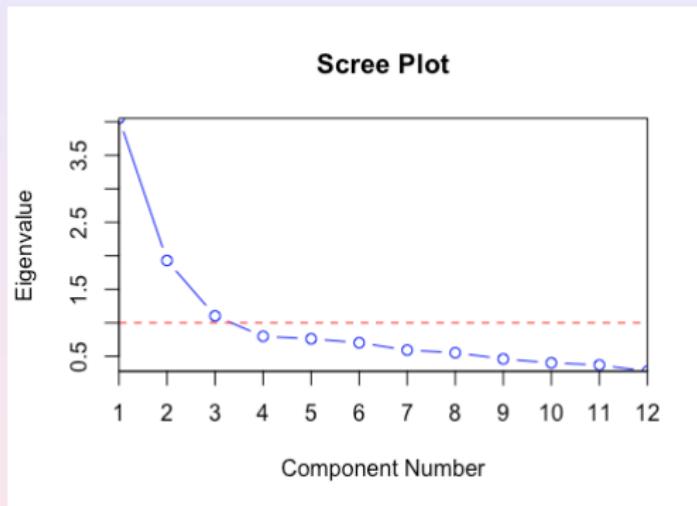
ACP – metodologia

- A solução desse problema pode ser encontrada por meio da aplicação de **multiplicadores de Lagrange**.
- Dada a primeira componente principal, obtém-se a segunda, $Y_2 = \beta_2^\top \mathbf{X}$, por meio da maximização de $\beta_2^\top \Sigma \beta_2$ sujeito a $\beta_2^\top \beta_2 = 1$ e $\beta_1^\top \beta_2 = 0$ (para garantir a ortogonalidade).
- Note que a ortogonalidade das componentes principais implica que a soma de suas variâncias seja igual à variância total do sistema de variáveis. Esse procedimento é repetido até a determinação da p -ésima componente principal.
- Os coeficientes das componentes principais estimadas são os autovetores $\hat{\beta}_1, \dots, \hat{\beta}_p$ da matriz \mathbf{S} e suas variâncias são os autovalores $\hat{\lambda}_1 \geq \hat{\lambda}_2 \geq \dots \geq \hat{\lambda}_p$ correspondentes.

ACP – metodologia

- Como a variância total do sistema é $\text{tr}(\mathbf{S}) = \sum_{i=1}^p \hat{\lambda}_i$, a contribuição da i -ésima componente principal é $\hat{\lambda}_i/\text{tr}(\mathbf{S})$. Lembrando que $\mathbf{S} = \sum_{i=1}^p \hat{\lambda}_i \hat{\beta}_i \hat{\beta}_i^\top$ podemos verificar quanto bem ela pode ser aproximada com um menor número de componentes principais. Detalhes podem ser obtidos na Nota de Capítulo 2.
- Na prática, a determinação do número de componentes principais a reter como novo conjunto de variáveis para futuras análises pode ser realizada por meio do **elbow plot**, também conhecido como **scree plot**, que consiste num gráfico cartesiano com os autovalores no eixo vertical e os índices correspondentes às suas magnitudes (em ordem decrescente) no eixo das abscissas.
- Um exemplo está apresentado na Figura 3

ACP – metodologia



ACP – metodologia

- A ideia é acrescentar componentes principais até que sua contribuição para a explicação da variância do sistema não apresente contribuições “relevantes”. Com base na Figura 3, apenas as duas primeiras componentes principais poderiam ser suficientes, pois a contribuição das seguintes é apenas marginal.
- Suponhamos que r componentes principais, Y_1, \dots, Y_r explicam uma parcela “substancial” da variabilidade do sistema multivariado. Então para a k -ésima unidade amostral, podemos substituir os valores das variáveis originais $\mathbf{x}_k = (x_{1k}, \dots, x_{pk})^\top$ pelos correspondentes **escores** associados às componentes principais, nomeadamente, $\hat{Y}_{1k}, \dots, \hat{Y}_{rk}$ com $\hat{Y}_{ik} = \hat{\beta}_i^\top \mathbf{x}_k$.
- Infelizmente, nem a matriz de covariâncias nem os correspondentes autovalores são invariantes relativamente a mudanças de escala. Em outras palavras, mudanças nas unidades de medida das características X_1, \dots, X_p podem acarretar mudanças na forma e na posição dos elipsóides correspondentes a pontos em que a função densidade é constante.

ACP – metodologia

- É difícil interpretar combinações lineares de características com unidades de medida diferentes e uma possível solução é padronizá-las por meio de transformações do tipo $Z_{ij} = (X_{ij} - \bar{X}_i)/S_i$ em que \bar{X}_i e S_i representam, respectivamente a média e o desvio padrão de X_{ij} antes da obtenção das componentes principais.
- A utilização da matriz de correlações **R** obtida por meio dessa transformação pode ser utilizada na determinação das componentes principais; no entanto, os resultados são, em geral, diferentes e não é possível passar de uma solução a outra por meio de uma mudança de escala dos coeficientes.
- Se as características de interesse foram medidas com as mesmas unidades, é preferível extrair as componentes principais utilizando a matriz de covariâncias amostrais **S**.

ACP – metodologia

- Lembrando que a i -ésima componente principal é $Y_i = \beta_{i1}X_1 + \dots + \beta_{ip}X_p$ do sistema multivariado, se as variáveis X_1, \dots, X_p tiverem variâncias similares ou forem variáveis padronizadas, os coeficientes β_{ij} indicam a importância e a direção da j -ésima variável relativamente à i -ésima componente principal.
- Nos casos em que as variâncias das variáveis originais são diferentes, convém avaliar sua importância relativa na definição das componentes principais por meio dos correspondentes coeficientes de correlação. O vetor de covariâncias entre as variáveis originais e a i -ésima componente principal é

$$\text{Cov}(\mathbf{I}_p \mathbf{X}, \boldsymbol{\beta}_i \mathbf{X}) = \mathbf{I}_p \boldsymbol{\Sigma} \boldsymbol{\beta}_i = \boldsymbol{\Sigma} \boldsymbol{\beta}_i = \lambda_i \boldsymbol{\beta}_i$$

pois $(\boldsymbol{\Sigma} - \lambda_i \mathbf{I}_p) \boldsymbol{\beta}_i = \mathbf{0}$ (ver Nota de Capítulo 1).

ACP – metodologia

- Uma estimativa desse vetor de covariâncias é $\widehat{\lambda}_i \widehat{\beta}_i$.
- Consequentemente, uma estimativa do coeficiente de correlação entre a j -ésima variável original e a i -ésima componente principal é

$$\widehat{\text{Corr}}(X_j, \beta_{ij}) = \frac{\widehat{\lambda}_i \widehat{\beta}_{ij}}{\sqrt{\widehat{\lambda}_i} s_j} = \frac{\sqrt{\widehat{\lambda}_i} \widehat{\beta}_{ij}}{s_j}$$

em que s_j é o desvio padrão amostral de X_j .

- As componentes principais podem ser encaradas como um conjunto de variáveis latentes (ou fatores) não correlacionados que servem para descrever o sistema multivariado original sem as dificuldades relacionadas com sua estrutura de correlação. Os coeficientes associados a cada componente principal servem para descrevê-las em termos das variáveis originais.
- A comparação entre unidades realizada por meio da componente principal Y_i é independente da comparação baseada na componente Y_j . No entanto, essa comparação pode ser ilusória se essas componentes principais não tiverem uma interpretação simples.

ACP – metodologia

- Como, em geral, isso não é a regra, costuma-se utilizar essa técnica como um passo intermediário para a obtenção de um dos possíveis conjuntos de combinações lineares ortogonais das variáveis originais passíveis de interpretação como variáveis latentes.
- Esses conjuntos estão relacionados entre si por meio de rotações rígidas (transformações ortogonais) e são equivalentes no sentido de aproximar as correlações entre as variáveis originais. Apesar de essas rotações rígidas implicarem uma perda da característica de ordenação das componentes principais em termos de porcentagem de explicação da variabilidade do sistema multivariado, muitas vezes produzem ganhos interpretativos.
- Há muitas opções computacionais para a análise de componentes principais no sistema R, dentre as quais destacamos (funções e pacotes entre parênteses) `prcomp` (stats), `princomp` (stats) e `pca` (FactoMineR). Como há vários métodos tanto para a extração quanto para a rotação das componentes principais, nem sempre os resultados coincidem. A análise deve ser realizada por tentativa e erro tendo como objetivo um sistema com interpretação adequada.

ACP – Exemplo 1

- **Exemplo 1.** Num estudo em que se pretendia avaliar o efeito de variáveis climáticas na ocorrência de suicídios por enforcamento na cidade de São Paulo foram observadas X_1 = temperatura máxima, X_2 = temperatura mínima, X_3 = temperatura média, X_4 = precipitação e X_5 = nebulosidade diárias para o período de 01/07/2006 e 31/07/2006. Os dados estão disponíveis em <http://www.ime.usp.br/~jmsinger/MorettinSinger/suicidios.xls> e detalhes em Zerbini et al. (2018)].
- Para reduzir o número de variáveis a serem utilizadas como variáveis explicativas numa regressão logística tento como variável resposta a ocorrência de suicídios por enforcamento nesse período consideramos uma análise de componentes principais.
- Os coeficientes das cinco componentes bem como as porcentagem da variância total do sistema explicada por cada uma delas (além da porcentagem acumulada correspondente) estão indicados na Tabela 1.

ACP – Exemplo 1

Tabela: Coeficientes das componentes principais e porcentagens da variância explicada: Exemplo 1

Variável	Componentes principais				
	CP1	CP2	CP3	CP4	CP5
Temperatura máxima	0,93	-0,25	0,05	0,26	0,06
Temperatura mínima	0,91	0,29	-0,18	-0,24	0,06
Temperatura média	0,99	-0,02	-0,03	0,00	-0,11
Precipitação	0,12	0,75	0,66	0,02	0,00
Nebulosidade	-0,12	0,85	-0,50	0,14	-0,01
% Variância	54	28	14	3	0
% Acumulada	54	82	97	100	100

ACP – Exemplo 1

- Uma análise do gráfico da escarpa sedimentar correspondente representado na Figura 4, conjuntamente com um exame da variância acumulada na Tabela 1 sugere que apenas duas componentes principais podem ser empregadas como resumo, retendo 82% da variância total.
- A primeira componente principal pode ser interpretada como **percepção térmica** e segunda, **percepção de acinzentamento**. Valores dessas novas variáveis para cada unidade amostral são calculadas como
 $CP_1 = 0,93 * tempmax + 0,91 * tempmin + 0,99 * tempmed$ e
 $CP_2 = 0,75 * precip + 0,85 * nebul$, com as variáveis originais devidamente padronizadas.
- O gráfico *biplot* disposto na Figura 5 é conveniente para representar a relação entre as duas componentes principais e as variáveis originais.

ACP – Exemplo 1

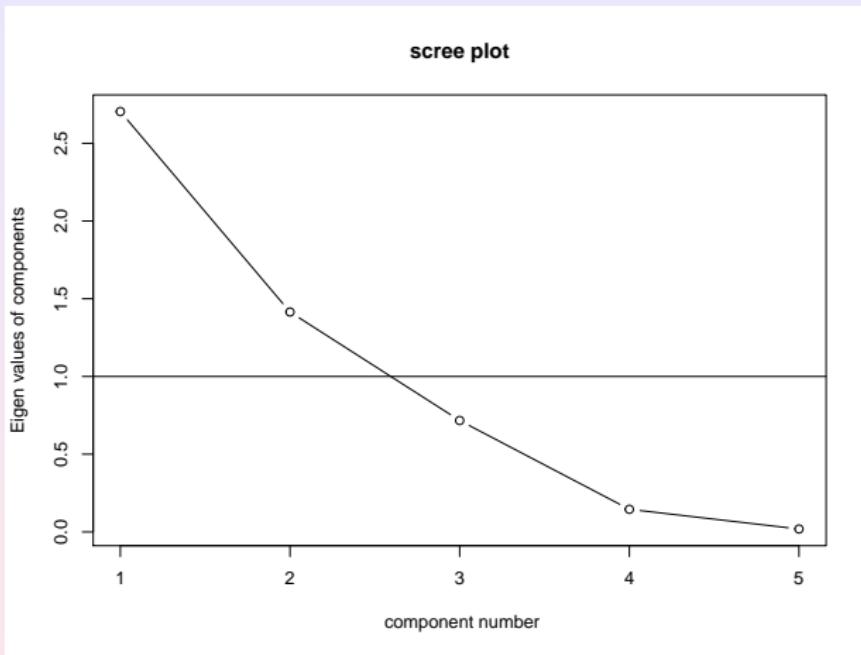


Figura: Gráfico da escarpa sedimentar (ou do cotovelo) para os dados do Exemplo 1.

ACP – Exemplo 1

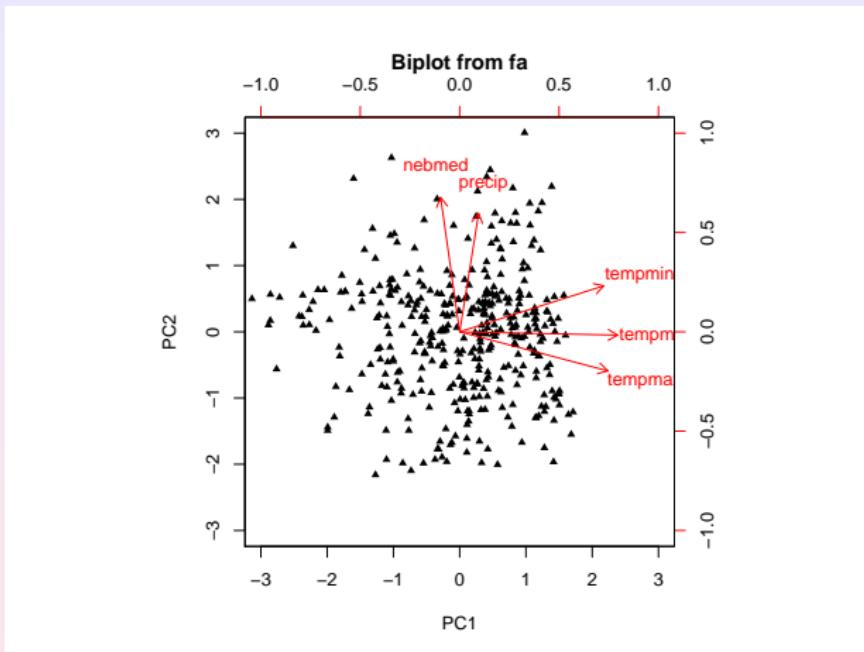


Figura: Gráfico *biplot* para os dados do Exemplo 1.

Referências

- Hastie, T., Tibshirani, R. and Friedman, J. (2017). *The Elements of Statistical Learning*, 2nd Edition, Springer.
- Härdle, W.K. and Simar, L. (2015). *Applied Multivariate Statistical Analysis*. Springer.
- James, G., Witten, D., Hastie, T. and Tibshirani, R. (2017). *Introduction to Statistical Learning*. Springer.
- Morettin, P. A. e Singer, J. M. (2022). *Estatística e Ciência de Dados*. LTC: Rio de Janeiro.

MAE 5905: Introdução à Ciência de Dados

Pedro A. Morettin

Instituto de Matemática e Estatística
Universidade de São Paulo
pam@ime.usp.br
<http://www.ime.usp.br/~pam>

Aula 14

22 de maio de 2023

Sumário

1 Análise Fatorial

Preliminares

- Primeiramente observemos que para explicar as relações entre p variáveis são necessárias p componentes principais; por esse motivo, o modelo adotado não é o ideal.
- O fato de as componentes principais não serem correlacionadas e ordenadas com variâncias decrescentes e o fato de que a técnica corresponde a uma fatoração da matriz de covariâncias da variáveis originais fazem com que a aproximação obtida quando consideramos apenas as primeiras componentes principais seja razoável.
- No entanto, essa técnica pode introduzir um erro sistemático na reprodução das correlações originais, pois podem existir uma ou mais dessas variáveis que sejam muito correlacionadas com as componentes principais desprezadas do que com aquelas retidas na análise.
- Outra observação importante, é que essa técnica utiliza toda a informação sobre cada uma das variáveis originais, embora seja razoável imaginar que uma parcela de sua variabilidade seja específica, nada tendo a ver com as demais variáveis do conjunto sob investigação.
- Além disso, pode-se suspeitar que os “verdadeiros fatores” responsáveis pela geração das observações tenham todos a mesma importância ou que sejam correlacionados entre si.

Preliminares

- Alguns desses problemas podem ser solucionados por meio da técnica de **Análise Fatorial**. A ideia que a fundamenta está baseada na partição da variância de cada variável do sistema multivariado numa **variância comum** e numa **variância específica**. Além disso, supõe-se que as correlações entre as p variáveis são geradas por um número $m < p$ de **variáveis latentes** (ou **fatores**).
- A **vantagem** dessa técnica relativamente àquela de componentes principais está na habilidade de reprodução da estrutura de correlações originais por meio de um pequeno número de fatores sem os erros sistemáticos que podem ocorrer quando simplesmente desprezamos algumas componentes principais.
- As **desvantagens** da Análise Fatorial estão na maior dificuldade de cálculo dos escores fatoriais e na existência de múltiplas soluções. Na realidade a estrutura de correlações das variáveis originais pode ser igualmente reproduzida por qualquer outro conjunto de variáveis latentes de mesma dimensão. A não ser que se imponham restrições adicionais, infinitas soluções equivalentes sempre existirão.

AF – metodologia

- Consideremos um vetor $\mathbf{X} = (X_1, \dots, X_p)^\top$ com média $\boldsymbol{\mu} = (\mu_1, \dots, \mu_p)^\top$ e matriz de covariâncias $\boldsymbol{\Sigma}$ com elementos σ_{ij} , $i, j = 1, \dots, p$.
- O modelo utilizado para Análise Fatorial de dados provenientes da observação das p variáveis agrupadas X_1, \dots, X_p é

$$X_i - \mu_i = \lambda_{i1}F_1 + \dots + \lambda_{im}F_m + e_i = \sum_{j=1}^m \lambda_{ij}F_j + e_i, \quad i = 1, \dots, p \quad (1)$$

em que F_j é o j -ésimo **fator comum** a todas as variáveis, λ_{ij} é o parâmetro (chamado de **carga factorial**) que indica a importância desse fator na composição da i -ésima variável e e_j é um **fator específico** para essa variável.

- Os coeficientes dos fatores, λ_{ij} , identificam o peso relativo de cada variável no componente. Quanto maior for o valor absoluto do coeficiente, mais importante é a variável correspondente ao estimar o fator. Os **escores fatoriais** são os valores estimados dos fatores.

AF – metodologia

- Como dito acima, as cargas fatoriais indicam o quanto um fator explica uma variável e variam de -1 a +1.
- Cargas próximas de -1 ou +1 indicam que o fator explica fortemente a variável. Cargas próximas de zero indicam que o fator tem pouca influência sobre a variável.
- Cargas fatoriais são difíceis de interpretar quando não houver rotação. Esta simplifica a estrutura das cargas e torna os fatores mais claramente distinguíveis e fáceis de interpretar.
- Há diversos métodos de rotação (ver abaixo) e devemos escolher aquele que proporciona melhor interpretação.

AF – metodologia

- Em notação matricial, o modelo pode ser escrito como

$$\mathbf{X} - \boldsymbol{\mu} = \Lambda \mathbf{f} + \mathbf{e} \quad (2)$$

em que Λ é a matriz com dimensão $p \times m$ de cargas fatoriais, $\mathbf{f} = (F_1, \dots, F_m)^\top$ é o vetor cujos elementos são os fatores comuns e $\mathbf{e} = (e_1, \dots, e_p)$ é um vetor cujos elementos são os fatores específicos.

- Adicionalmente, supomos que $E(\mathbf{f}) = \mathbf{0}$, $Cov(\mathbf{f}) = \mathbf{I}_m$, $E(\mathbf{e}) = \mathbf{0}$ e $Cov(\mathbf{e}) = \psi = \text{diag}(\psi_1, \dots, \psi_p)$ e que $Cov(\mathbf{f}, \mathbf{e}) = \mathbf{0}$. Os elementos de ψ são as **variâncias específicas**.

AF – metodologia

- Para avaliar a relação entre a estrutura de covariâncias de \mathbf{X} e os fatores, observemos que

$$\begin{aligned}
 \text{Cov}(X_i, X_k) &= \text{Cov}\left(\sum_{j=1}^m \lambda_{ij} F_j, \sum_{\ell=1}^m \lambda_{k\ell} F_\ell\right) \\
 &= \sum_{j=1}^m \sum_{\ell=1}^m \lambda_{ij} \lambda_{k\ell} E(F_j F_\ell) + E(e_i e_j) \quad (3) \\
 &= \sum_{j=1}^m \sum_{\ell=1}^m \lambda_{ij} \lambda_{k\ell} + E(e_i e_j)
 \end{aligned}$$

- Consequentemente, $\text{Cov}(X_i, X_k) = \sigma_{ik} = \sum_{j=1}^m \lambda_{ij} \lambda_{kj}$ se $i \neq k$ e $\text{Cov}(X_i, X_i) = \sigma_{ii} = \sum_{j=1}^m \lambda_{ij}^2$. O termo $\sum_{j=1}^m \lambda_{ij}^2$ é conhecido por **comunalidade** da i -ésima variável.
- Em notação matricial, podemos escrever

$$\boldsymbol{\Sigma} = \boldsymbol{\Lambda} \boldsymbol{\Lambda}^\top + \boldsymbol{\psi}$$

e o objetivo é estimar os elementos de $\boldsymbol{\Lambda}$ e $\boldsymbol{\psi}$.

AF – metodologia

- A comunidade de cada variável é a proporção da variabilidade explicada pelos fatores.
- Quanto mais próxima de 1, melhor é a explicação da variável pelo fator. Decidimos acrescentar um fator se ele contribuir significativamente ao ajuste de algumas variáveis.
- A variância de um fator fornece a variabilidade nos dados explicada pelo fator. Se usarmos CP para extrair fatores, e não usarmos rotação, a variância de cada fator é igual ao seu autovalor.
- Rotação muda a distribuição da proporção da variabilidade explicada por cada fator. Mas a variação total explicada por todos os fatores se mantém.
- Quanto maior a variância de um fator, mais ele explica a variabilidade nos dados. A porcentagem da variância explicada por cada fator varia de zero a 1.

AF – metodologia

- Em Análise de Componentes Principais, consideramos o modelo linear $\mathbf{Y} = \mathbf{BX}$ em que \mathbf{Y} é o vetor cujos elementos são as componentes principais e $\mathbf{B} = (\beta_1^T, \dots, \beta_p^T)^T$ é a matriz cuja i -ésima linha contém os coeficientes da i -ésima componente principal.
- A matriz de covariâncias de \mathbf{X} é fatorada como $\Sigma = \Lambda \Lambda^T$. Em Análise Fatorial, consideramos o modelo linear $\mathbf{X} = \Lambda \mathbf{f} + \mathbf{e}$ e a matriz de covariâncias de \mathbf{X} é fatorada como $\Sigma = \Lambda \Lambda^T + \psi$.
- Uma diferença entre os dois enfoques é que enquanto a fatoração de Σ é única em Análise de Componentes Principais, ela não o é em Análise Fatorial, pois se \mathbf{T} for uma matriz ortogonal (i.e., $\mathbf{T}\mathbf{T}^T = \mathbf{I}_m$), obteremos

$$\Sigma = \Lambda \Lambda^T + \psi = \Lambda \mathbf{T} \mathbf{T}^T \Lambda^T + \psi = \Lambda \mathbf{T} (\Lambda \mathbf{T})^T + \psi$$

e embora as cargas fatoriais $\Lambda \mathbf{T}$ sejam diferentes das cargas fatoriais Λ , a habilidade de reproduzir a matriz de covariâncias Σ não se altera. Escolhendo matrizes ortogonais diferentes, podemos determinar cargas fatoriais diferentes. A escolha de uma transformação conveniente será discutida posteriormente.

AF – metodologia

Uma análise factorial consiste dos seguintes passos:

- a) Estimação dos parâmetros do modelo (λ_{ij} e ψ_i) a partir de um conjunto de observações das variáveis X_1, \dots, X_p .
- b) Interpretação dos fatores determinados a partir das cargas fatoriais obtidas em a). Com esse objetivo considera-se a **rotação** dos fatores por meio de transformações ortogonais.
- c) Estimação dos valores dos fatores comuns, chamados **escores fatoriais** para cada unidade amostral a partir dos valores das cargas fatoriais e das variáveis observadas.

AF – metodologia

- Existem duas classes de métodos para estimação dos parâmetros do modelo fatorial. Na primeira classe consideramos o **método de máxima verossimilhança** e na segunda, métodos heurísticos como o **método do fator principal** ou o **método do centroide**.
- Para o método de máxima verossimilhança, supomos adicionalmente que as variáveis X_1, \dots, X_p seguem uma distribuição normal (multivariada) e que o número de fatores m é conhecido. Os estimadores são obtidos por meio da solução do sistema de equações (ver Nota de Capítulo 3)

$$\begin{aligned} \mathbf{S}\psi^{-1}\Lambda &= \Lambda(\mathbf{I}_m + \Lambda^\top\psi^{-1}\Lambda) \\ \text{diag}(\mathbf{S}) &= \text{diag}(\Lambda\Lambda^\top + \psi) \end{aligned} \tag{4}$$

que deve ser resolvido por meio de métodos iterativos. Detalhes podem ser encontrados em Morrison (1972).

- Uma das vantagens desse método é que as mudanças de escala das variáveis originais alteram os estimadores apenas por uma mudança de escala.

AF – metodologia

- Se uma das variáveis X_1, \dots, X_p for multiplicada por uma constante, os estimadores das cargas fatoriais correspondentes ficam multiplicados pela mesma constante e o estimador da variância específica associada fica multiplicado pelo quadrado da constante. Dessa forma, podemos fazer os cálculos com as variáveis padronizadas (substituindo a matriz de covariâncias amostral \mathbf{S} pela correspondente matriz de correlações amostrais \mathbf{R} e posteriormente escrever os resultados em termos das unidades de medida originais).
- O método do fator principal está intimamente relacionado com a técnica utilizada na análise de componentes principais. Segundo esse método, os fatores são escolhidos obedecendo à ordem decrescente de sua contribuição à communalidade total do sistema multivariado.
- Nesse contexto, o processo tem início com a determinação de um fator F_1 cuja contribuição à communalidade total é a maior possível; em seguida, um segundo fator não correlacionado com F_1 e tal que maximize a communalidade residual é obtido. O processo continua até que a communalidade total tenha sido exaurida.

AF – metodologia

Na prática, as communalidades e as variâncias específicas devem ser estimadas com base nos dados amostrais. Embora existam vários métodos idealizados para essa finalidade, nenhum se mostra superior aos demais. Dentre os estimadores mais comuns para a communalidade de uma variável X_i , destacamos:

- i) o quadrado do **coeficiente de correlação múltipla** entre a variável X_i e as demais;
- ii) o maior valor absoluto dos elementos de i -ésima linha da matriz de correlações amostrais;
- iii) estimadores obtidos de análises preliminares por meio de processos iterativos.

AF – metodologia

Outro problema prático é a determinação do número de fatores a incluir na análise. Os critérios mais utilizados para esse fim são:

- i) determinação do número de fatores por meio de algum conhecimento *a priori* sobre a estrutura dos dados;
- ii) número de componentes principais correspondentes a autovalores da matriz **R** maiores que 1;
- iii) explicação de certa proporção (escolhida arbitrariamente) da communalidade ou da variância total.

AF – metodologia

Um algoritmo comumente utilizado para a obtenção das cargas fatoriais e das variâncias específicas é

- i) Obter as p componentes principais com base na matriz de correlações amostrais \mathbf{R} .
- ii) Escolher m fatores segundo um dos critérios mencionados.
- iii) Substituir os elementos da diagonal principal de \mathbf{R} por estimadores das communalidades correspondentes por meio de um dos métodos descritos acima, obtendo a chamada **matriz de correlações reduzida**, \mathbf{R}^* .
- iv) Extrair m fatores da matriz \mathbf{R}^* , obtendo novos estimadores das communalidades que vão substituir aqueles obtidos anteriormente na diagonal principal.
- v) Repetir o processo dos itens ii) - iv) até que a diferença entre dois conjuntos sucessivos de estimadores das communalidades seja desprezável.

AF – metodologia

- O método do centroide foi desenvolvido por Thurstone (1947) para simplificar os cálculos mas não é muito utilizado em virtude das recentes facilidades computacionais; os resultados obtidos por intermédio desse método não diferem muito daqueles obtidos pelo método do fator principal.
- Como a interpretação dos fatores numa análise factorial é uma característica importante em aplicações práticas, pode-se utilizar a técnica de **rotação dos fatores** para obter resultados mais palatáveis.
- Consideremos um exemplo em que cinco variáveis A, B, C, D e E são representadas num espaço factorial bidimensional conforme a representação da Figura 1.

AF – metodologia

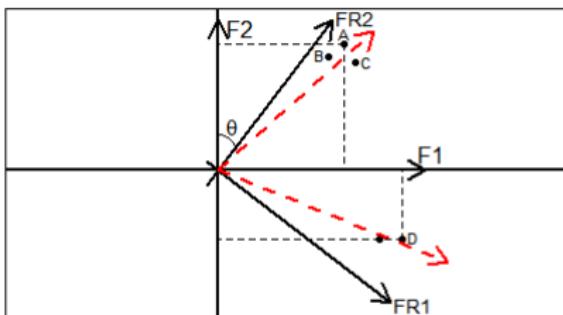


Figura: Representação de cinco variáveis num espaço vetorial bidimensional.

AF – metodologia

Como ilustrado na Tabela 1, as cargas fatoriais relativas ao fator F_1 são altas e positivas para todas as variáveis. Por outro lado, apenas as variáveis A, B e C têm cargas positivas no fator F_2 ; as cargas das variáveis D e E são negativas nesse fator.

Tabela: Cargas fatoriais para as variáveis A, B, C, D e E

Variável	Fatores iniciais		Fatores rotacionados	
	F_1	F_2	FR_1	FR_2
A	0,75	0,63	0,14	0,95
B	0,69	0,57	0,14	0,90
C	0,80	0,49	0,18	0,92
D	0,85	-0,42	0,94	0,09
E	0,76	-0,42	0,92	0,07

AF – metodologia

- Dois aglomerados de variáveis podem ser identificados na Figura 1: um formado pelas variáveis A, B e C e o outro pelas variáveis D e E .
- Apesar disso, esses aglomerados não são evidentes nas cargas fatoriais da Tabela 1. Uma rotação dos fatores (com os eixos rotulados FR_1 e FR_2) como aquela indicada na figura juntamente com as novas cargas fatoriais apresentadas na Tabela 1 ressaltam a separação entre os dois conjuntos de variáveis.
- Na solução inicial, cada variável é explicada por dois fatores enquanto que na solução obtida com a rotação dos fatores, apenas um deles é suficiente para explicar a correspondente estrutura de covariância.
- Em princípio, também podemos considerar rotações oblíquas, que são bem mais flexíveis, pois os fatores não precisam ser necessariamente ortogonais. Essa característica pode até ser considerada mais realista, pois a ortogonalidade não é determinante da relação entre os fatores. Os eixos realçados em vermelho na Figura 1 correspondem a uma dessas rotações oblíquas.

AF – metodologia

O objetivo de qualquer rotação é obter fatores interpretáveis e com a estrutura mais simples possível. Nesse sentido, Thurstone (1947) sugere condições para se obter uma estrutura mais simples, nomeadamente:

- i) Cada linha da matriz de cargas fatoriais Λ deve conter pelo menos um valor nulo.
- ii) Cada coluna da matriz de cargas fatorais deveria ter pelo menos tantos valores nulos quantas forem as colunas.
- iii) Para cada par de colunas deve haver algumas variáveis com cargas fatoriais pequenas numa delas e altas na outra.
- iv) Para cada par de colunas uma grande porcentagem das variáveis deve ter cargas fatoriais não nulas em ambas.
- v) Para cada par de colunas deve haver somente um pequeno número de variáveis com cargas fatoriais altas em ambas.

AF – metodologia

Como consequência dessas sugestões,

- i) Muitas variáveis (representadas como vetores no espaço dos fatores) devem ficar próximas dos eixos.
- ii) Muitas variáveis devem ficar próximas da origem quando o número de fatores for grande.
- iii) Somente um pequeno número de variáveis ficam longe dos eixos.

AF – metodologia

A principal crítica às sugestões de Thurstone é que na prática poucas são as situações que admitem uma simplificação tão grande. O que se procura fazer é simplificar as linhas e colunas da matriz de cargas fatoriais e os métodos mais comumente empregados com essa finalidade são:

- **Método Varimax** em que se procura simplificar a complexidade fatorial, tentando-se obter fatores com poucos valores grandes e muitos valores nulos ou pequenos na respectiva coluna da matriz de cargas fatoriais. Após uma rotação Varimax, cada variável original tende a estar associada com poucos (preferencialmente, um) fatores e cada fator tende a se associar com poucas variáveis. Esse é o método mais utilizado na prática.
- **Método Quartimax** em que se procura maximizar o número de fatores necessários para explicar cada variável. Em geral, esse método produz um fator associado em que muitas variáveis têm cargas altas ou médias, o que nem sempre é conveniente para a interpretação.
- **Método Equimax**, uma mistura dos métodos Varimax e Quartimax.
- **Método Promax**, utilizado para rotações oblíquas.

AF – estimação

- Um dos objetivos tanto da Análise de Componentes Principais quanto da Análise Fatorial, é substituir as p variáveis originais X_1, \dots, X_p por um número menor, digamos, m em análises subsequentes.
- No caso de componentes principais, podem-se utilizar as estimativas $\hat{Y}_{ik} = \hat{\beta}_i \mathbf{x}_k$, $i = 1, \dots, m$ para substituir os valores \mathbf{x}_k observados para a k -ésima unidade amostral.
- Esse processo é mais complicado quando lidamos com a obtenção dos valores dos fatores F_1, \dots, F_m (denominados **escores fatoriais**) em Análise Fatorial, que não podem ser estimados no sentido estatístico usual, pois os fatores não são observáveis.
- Com esse objetivo, o **método de Bartlett** (1937) consiste em considerar (2) como um modelo de regressão heterocedátilo em que se supõe que as matrizes de cargas fatoriais, Λ e de variâncias específicas ψ , são conhecidas e se considera o termo e como um vetor de erros.

AF – estimação

Minimizando

$$Q(\mathbf{f}) = \mathbf{e}^\top \boldsymbol{\psi}^{-1} \mathbf{e} = (\mathbf{x} - \boldsymbol{\mu} - \mathbf{\Lambda f})^\top \boldsymbol{\psi}^{-1} (\mathbf{x} - \boldsymbol{\mu} - \mathbf{\Lambda f})$$

obtemos

$$\hat{\mathbf{f}} = [\mathbf{\Lambda}^\top \boldsymbol{\psi}^{-1} \mathbf{\Lambda}]^{-1} \mathbf{\Lambda}^\top \boldsymbol{\psi}^{-1} (\mathbf{x} - \boldsymbol{\mu})$$

e substituindo $\mathbf{\Lambda}$, $\boldsymbol{\psi}$ e $\boldsymbol{\mu}$, respectivamente, por estimativas $\hat{\mathbf{\Lambda}}$, $\hat{\boldsymbol{\psi}}$ e $\bar{\mathbf{x}}$, podemos construir os escores fatoriais para a k -ésima unidade amostral como

$$\hat{\mathbf{f}}_k = [\hat{\mathbf{\Lambda}}^\top \hat{\boldsymbol{\psi}}^{-1} \hat{\mathbf{\Lambda}}]^{-1} \hat{\mathbf{\Lambda}}^\top \hat{\boldsymbol{\psi}}^{-1} (\mathbf{x}_k - \bar{\mathbf{x}}).$$

AF – estimação

- Alternativamente, no **método de regressão**, supõe-se que os fatores comuns, \mathbf{f} e específicos \mathbf{e} são independentes e têm distribuições normais multivariadas com dimensões m e p , respectivamente, de forma que o par $(\mathbf{X} - \mu, \mathbf{f})$ também tem uma distribuição normal multivariada de dimensão $p + m$ com matriz de covariâncias

$$\begin{bmatrix} \Lambda \Lambda^\top + \psi & \Lambda \\ \Lambda^\top & I_m \end{bmatrix}.$$

- Utilizando propriedades da distribuição normal multivariada, segue que a distribuição condicional de \mathbf{f} dado $\mathbf{X} - \mu$ também é normal multivariada com vetor de médias

$$E(\mathbf{f}|\mathbf{X} - \mu) = \Lambda^\top [\Lambda \Lambda^\top + \psi]^{-1} (\mathbf{X} - \mu)$$

e matriz de covariâncias

$$Cov(\mathbf{f}|\mathbf{X} - \mu) = I_m - \Lambda^\top [\Lambda \Lambda^\top + \psi]^{-1} \Lambda.$$

AF – Número de fatores

- O termo $\hat{\Lambda}^T [\hat{\Lambda}\hat{\Lambda}^T + \psi]^{-1}$ corresponde aos coeficientes de uma regressão multivariada tendo os fatores como variáveis respostas e $\mathbf{X} - \mu$ como variáveis explicativas. Utilizando as estimativas $\hat{\Lambda}$, $\hat{\psi}$, podemos calcular os escores fatoriais para a k -ésima unidade amostral (com valores das variáveis originais \mathbf{x}_k) por meio de

$$\hat{\mathbf{f}}_k = \hat{\Lambda}^T [\hat{\Lambda}\hat{\Lambda}^T + \hat{\psi}]^{-1} (\mathbf{x}_k - \bar{\mathbf{x}}).$$

Pacotes do R que podem ser utilizados para a AF são o [psych](#), o [GPArotation](#) e o [robustfa](#).

- Para determinar o número de fatores a considerar, podemos usar vários critérios:
 - [1] **Teste Scree**: devido a Cattel (1966), consiste do gráfico dos autovalores contra o número de fatores (ou CPs, de uma ACP). Procura-se o ponto ([elbow](#)) onde a inclinação muda drasticamente.
 - [2] **Regra de Kaiser–Guttman**: devida a Guttman(1954) e Kaiser (1960), é similar ao teste scree, mas consideram-se componentes ou fatores com autovalores maiores do que 1. Esse é o gráfico que temos usado.
 - [3] **Análise paralela**: devida a Horn (1965), propõe reter somente autovalores que sejam superiores ou iguais à média dos autovalores obtidos de k matrizes de correlações calculadas com n observações aleatórias.



AF – Número de fatores

A estratégia da Análise Paralela é:

- (a) Gere n v.a.s de acordo com uma $N(0, 1)$ independentemente para p variáveis;
- (b) calcule a matriz de correlações de Pearson;
- (c) calcule os autovalores dessa matriz;
- (d) repita passos (a)-(c) k vezes;
- (e) calcule uma medida de localização, ML, para os k vetores de autovalores: média, mediana, p -quantil etc;
- (f) Substitua o valor 1 da regra de Guttman–Kaiser, ou seja, conte o número de autovalores maiores que ML.

Há, também, soluções não gráficas a esses testes.

- Ótima coordenada:

$$n_{oc} = \#\{(\lambda_i \geq 1) \text{ e } (\lambda_i \geq \lambda \text{ previsto pelo teste scree K-G})\}$$

ou

$$n_{oc} = \#\{(\lambda_i \geq ML \text{ e } (\lambda_i \geq \lambda \text{ previsto pelo teste scree da AP})\}$$

- **Fator de aceleração:** coloca ênfase na coordenada onde a inclinação da curva muda abruptamente

AF – Exemplo 1

- **Exemplo 1.** Num estudo planejado para avaliar o nível de poluição atmosférica por meio de medidas de elementos químicos depositados em cascas de árvores, obtiveram-se observações da concentração de Al, Ba, Cu, Fe, Zn, P, Cl, Sr e Ca entre outros elementos em 193 unidades da espécie *Tipuana tipu* na cidade de São Paulo.
- Esses dados constituem um subconjunto daqueles disponíveis em <http://www.ime.usp.br/~jmsinger/MorettinSinger/arvores.xls> O objetivo aqui é obter um conjunto de fatores que permitam identificar características comuns a essas variáveis. Os resultados provenientes de uma análise de componentes principais estão dispostos na Tabela 2.

AF – Exemplo 1

Tabela: Coeficientes de componentes principais (CP) para os dados do Exemplo 1

	CP1	CP2	CP3	CP4	CP5	CP6	CP7	CP8	CP9
Al	0.90	-0.11	0.09	0.21	-0.16	0.19	-0.08	-0.17	0.17
Ba	0.88	-0.16	0.09	0.10	-0.10	0.27	0.09	0.31	-0.01
Cu	0.82	0.18	-0.05	-0.23	0.31	-0.18	-0.31	0.08	0.01
Fe	0.95	-0.10	0.07	0.10	-0.03	0.10	0.00	-0.18	-0.19
Zn	0.83	0.16	-0.13	-0.22	0.29	-0.21	0.31	-0.05	0.05
P	0.25	0.69	-0.25	0.53	-0.20	-0.28	0.00	0.05	-0.01
Cl	0.17	0.53	0.60	-0.42	-0.39	-0.07	0.01	0.00	0.00
Mg	-0.24	0.22	0.78	0.35	0.40	0.05	0.02	0.00	0.00
Ca	-0.20	0.77	-0.33	-0.12	0.15	0.47	0.00	-0.04	0.00
% Var	0.45	0.17	0.13	0.08	0.07	0.06	0.02	0.02	0.01
% Acum	0.45	0.61	0.75	0.83	0.89	0.95	0.97	0.99	1.00

AF – Exemplo 1

Uma análise da porcentagem da variância explicada pelas componentes principais juntamente com um exame do gráfico da escarpa sedimentar (**scree plot**) correspondente, apresentado na Figura 2 sugere que três fatores, que explicam 75% da variância total do sistema de variáveis originais poderiam contemplar uma representação adequada.

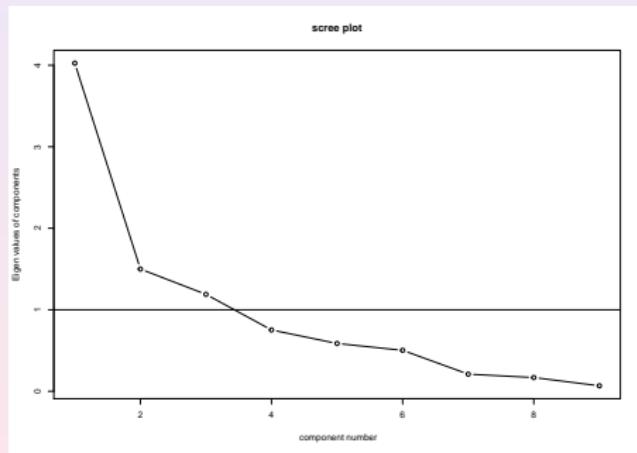


Figura: Gráfico da escarpa sedimentar para os dados do Exemplo 1.

AF – Exemplo 1

As cargas fatoriais correspondentes três fatores rotacionados obliquamente juntamente com as communalidades e especificidades correspondentes estão dispostos na Tabela 3.

Tabela: Cargas fatoriais, communalidades e especificidades correspondentes a uma análise factorial para os dados do Exemplo 1

	Fator 1	Fator 2	Fator 3	Comunalidade	Especificidade
Al	0.91	-0.11	0.03	0.82	0.18
Ba	0.86	-0.13	0.00	0.75	0.25
Cu	0.75	0.27	-0.04	0.66	0.34
Fe	0.97	-0.08	0.01	0.95	0.05
Zn	0.74	0.27	-0.13	0.68	0.32
P	0.20	0.47	0.05	0.25	0.75
Cl	0.22	0.27	0.39	0.22	0.78
Mg	-0.04	0.02	0.73	0.54	0.46
Ca	-0.21	0.67	0.01	0.49	0.51

AF – Exemplo 1

Os gráficos das Figuras 3 e 4 também podem ser utilizados para identificar os fatores.

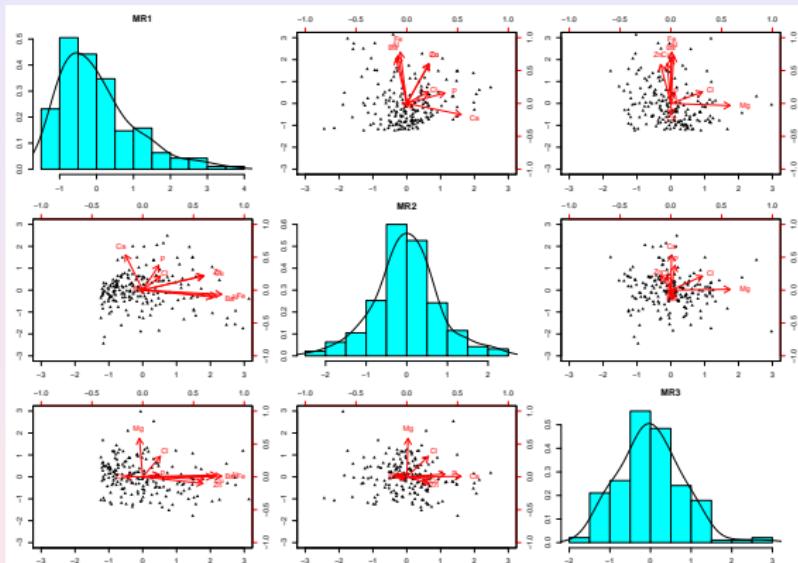


Figura: Gráfico *biplot* correspondente aos três fatores considerados para os dados do Exemplo 1.

AF – Exemplo 1

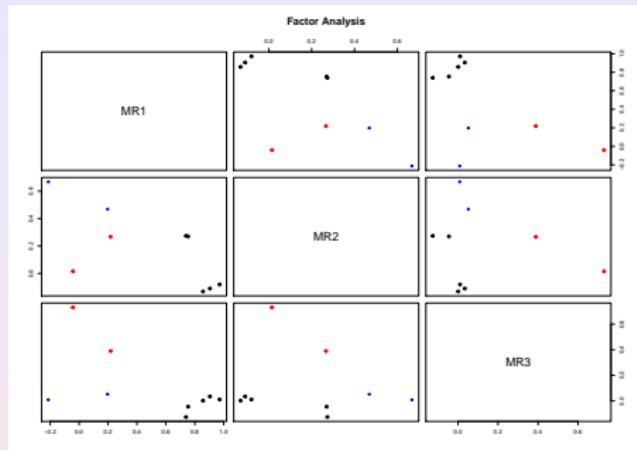


Figura: Gráfico cartesiano correspondente aos três fatores considerados para os dados do Exemplo 1.

O Fator 1 tem cargas fatoriais concentradas nos elementos Al, Ba, Cu, Fe e Zn, que estão associados à poluição de origem veicular gerada por desgaste de freios e ressuspensão de poeira; o Fator 2 está tem cargas fatoriais concentradas em P e Ca, características da saúde arbórea e o Fator 3 tem as cargas fatoriais concentradas em Cl e Mg.

AF – Exemplo 2

- **Exemplo 2.** Vamos retomar o Exemplo 1 da Aula 13, em que se pretendia avaliar os efeitos de variáveis climáticas na ocorrência de suicídios por enforcamento na cidade de São Paulo.
- Nesse exemplo obtivemos duas CPs que explicavam 82% da variância total.
- Vamos usar a função `factanal()` do pacote `stats`. Essa função usa MV numa matriz de covariâncias ou numa matriz de dados.
- Obtemos os resultados abaixo:

Call:

```
factanal(x = clim1, factors = 2, rotation = "varimax")
```

Uniquenesses:

	tempmax	tempmin	tempmed	precip	nebmed
	0.061	0.020	0.005	0.890	0.482

Loadings:

	Factor1	Factor2
tempmax	0.932	-0.266
tempmin	0.883	0.447
tempmed	0.997	
precip		0.323
nebmed	-0.142	0.706

	Factor1	Factor2
SS loadings	2.668	0.874
Proportion Var	0.534	0.175
Cumulative Var	0.534	0.708

Test of the hypothesis that 2 factors are sufficient.
The chi square statistic is 11.03 on 1 degree of freedom.
The p-value is 0.000895

AF – Exemplo 2

- Na saída, encontramos o termo **uniqueness**, ou **ruído**, corresponde à proporção da variabilidade que **não pode ser explicada** pela combinação linear dos fatores(variância específica). Valor alto para uma variável indica que o fator não contribui bem para a sua variância. É a diagonal da matriz ψ . Nesse caso, os fatores não contribuem para a variância de Precipitação e Nebulosidade.

- A matriz λ das cargas fatoriais é dada a seguir:

```
> load<-fafit\$loadings[,1:2]
      Factor1    Factor2
tempmax  0.93205114 -0.26553163
tempmin  0.88305960  0.44690513
tempmed  0.99713621  0.02715943
precip   0.07422757  0.32313465
nebmed   -0.14166420  0.70579832
```

- A matriz ψ é dada por

```
> Psi <-diag(fafit\$uniquenesses)
```

	[,1]	[,2]	[,3]	[,4]	[,5]
[1,]	0.0607741	0.0000000	0.000	0.0000	0.0000000
[2,]	0.0000000	0.02048162	0.000	0.0000	0.0000000
[3,]	0.0000000	0.0000000	0.005	0.0000	0.0000000
[4,]	0.0000000	0.0000000	0.000	0.8901	0.0000000
[5,]	0.0000000	0.0000000	0.000	0.0000	0.4817635

- A comunalidade é obtida tomando os quadrados das cargas:

```
> apply(fafit\$loadings^2,1,sum)
tempmax tempmin tempmed precip nebmed
```

AF – Exemplo 2

- Não foi possível considerar 3 fatores, pois o programa não aceita valor maior do que 2 para 5 variáveis.
- A proporção da variância explicada pelos fatores é 70,8%, menor do que as CPs.
- A estimativa da matriz Σ é dada por

	tempmax	tempmin	tempmed	precip
tempmax	1.00000048	0.7043893	0.92217026	-0.01661858
tempmin	0.70438926	1.0000001	0.89266839	0.20995790
tempmed	0.92217026	0.8926684	1.00001826	0.08279115
precip	-0.01661858	0.2099579	0.08279115	1.00002576
nebmed	-0.31945006	0.1903270	-0.12208942	0.21755251
	nebmed			
tempmax	-0.3194501			
tempmin	0.1903270			
tempmed	-0.1220894			
precip	0.2175525			
nebmed	0.9999835			

AF – Exemplo 2

- A matriz residual estimada é

	tempmax	tempmin	tempmed	precip	nebmed
tempmax	0.000000	-0.000390	0.000104	-0.015332	0.009215
tempmin	-0.000390	0.000000	0.000035	-0.008720	0.000870
tempmed	0.000104	0.000035	-0.000018	0.003171	-0.000927
precip	-0.015332	-0.008720	0.003171	-0.000026	0.072404
nebmed	0.009215	0.000870	-0.000927	0.072404	0.000017

e vemos que os valores são próximos de zero, indicando que o modelo fatorial está adequado.

- Para determinar o número de fatores podemos obter os autovalores da matriz de correlações estimada:

```
> ev<-eigen(cor(clima1))  
eigen() decomposition  
values
```

```
[1] 2.70399548 1.41461150 0.71677349 0.14576231 0.01885722
```

mostrando que tomamos 2 fatores, correspondentes aos valores próprios maiores que 1.

AF – Exemplo 2

A seguir temos alguns gráficos mencionados anteriormente sobre a determinação do número de fatores.

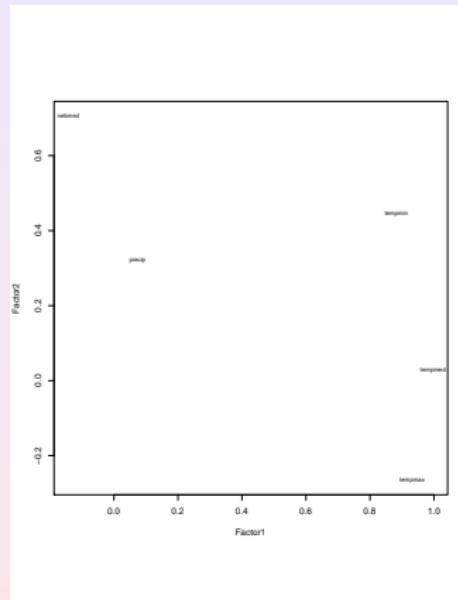


Figura: Gráfico do fator 1 vs fator 2 para o Exemplo2

AF – Exemplo 2

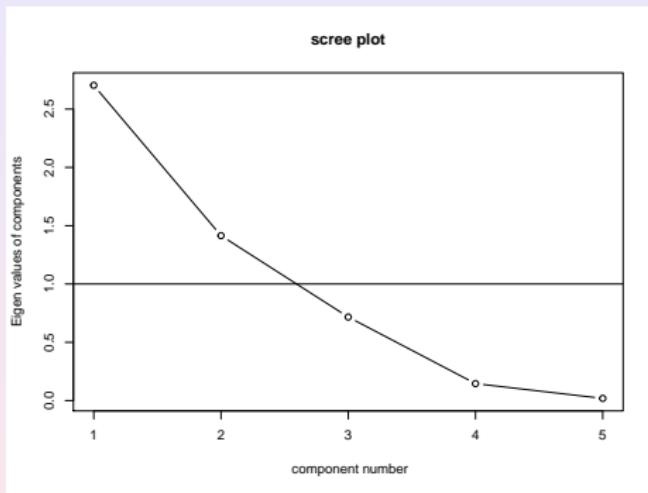


Figura: Scree plot para Exemplo 2.

AF – Exemplo 2

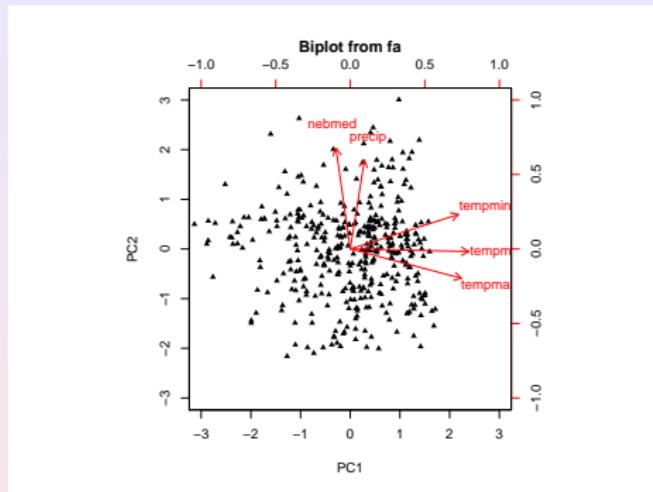


Figura: Biplot para Exemplo 2.

AF – Exemplo 2

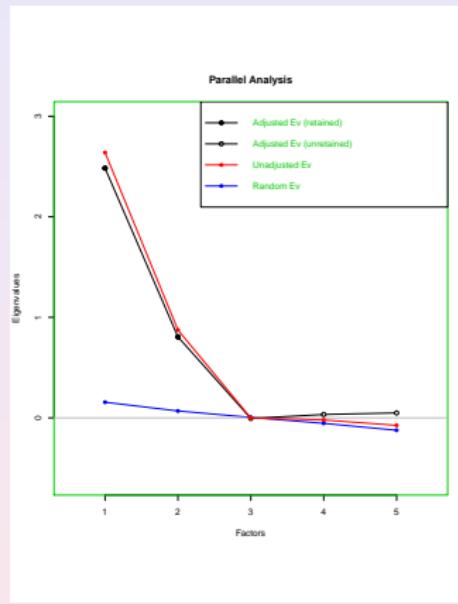


Figura: Análise Paralela para Exemplo 2.

AF – Exemplo 2

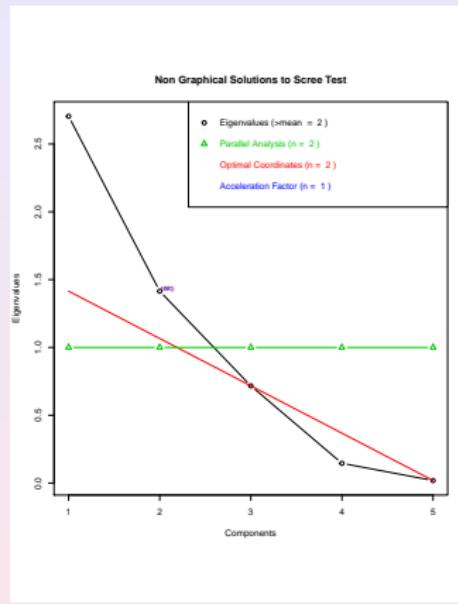


Figura: Soluções não gráficas para Exemplo 2.

Referências

- Bartlett, M.S. (1937). The statistical conception of mental factors. *British Journal of Psychology*, **28**, 97-104.
- Hastie, T., Tibshirani, R. and Friedman, J. (2017). *The Elements of Statistical Learning*, 2nd Edition, Springer.
- Härdle, W.K. and Simar, L. (2015). *Applied Multivariate Statistical Analysis*. Springer.
- James, G., Witten, D., Hastie, T. and Tibshirani, R. (2017). *Introduction to Statistical Learning*. Springer.
- Morettin, P. A. e Singer, J. M. (2022). *Estatística e Ciência de Dados*. LTC: Rio de Janeiro.
- Morrison, D.F. (1976). *Multivariate Statistical Methods*, 2nd Ed. New York: McGraw-Hill.
- Thurstone, L.L. (1947). *Multiple Factor Analysis: A development and expansion of vectors of the mind..* Chicago: University of Chicago Press.

MAE 5905: Introdução à Ciência de Dados

Pedro A. Morettin

Instituto de Matemática e Estatística
Universidade de São Paulo
pam@ime.usp.br
<http://www.ime.usp.br/~pam>

Aula 15

7 de junho de 2023

Sumário

1 Análise de Componentes Independentes

2 Decomposição de matrizes

Preliminares

- Vimos, no estudo de componentes principais, que essas são não correlacionadas, ou ortogonais. No caso da análise fatorial, há a suposição adicional de normalidade. Na análise de componentes independentes (ICA, em Inglês), a ideia é obter componentes independentes e não gaussianas.
- A ICA é relativamente recente, sendo introduzida na década de 1980 no contexto de redes neurais e encontra aplicações em processamento de sinais biomédicos, separação de sinais de áudio, séries temporais financeiras etc.
- As referências principais nessa área são Hyvärinen e Oja (1997), Hyvärinen (1999), Hyvärinen et al. (1999), Stone (2004) e Gegembauer (2010).
- Vamos supor que os dados consistem de um número p de variáveis, $\mathbf{X} = (X_1, \dots, X_p)^\top$, e consideramos a transformação

$$Y_i = \sum_{j=1}^p w_{ij} X_j(t), \text{ para } i = 1, \dots, q, \quad (1)$$

onde $q < p$ e os w_{ij} são alguns coeficientes que definem a representação.

- Reunindo os coeficientes w_{ij} em uma matrix \mathbf{W} , de ordem $q \times p$, e os Y_j num vetor \mathbf{Y} , de ordem $q \times 1$, obtemos

$$\mathbf{Y} = \mathbf{WX}$$

Separação cega de fontes

- A diferença com CP e AF é que escolhemos \mathbf{W} de modo que as variáveis Y_i sejam independentes e não gaussianas. ICA é um caso particular da **Separação Cega de Fontes** (*blind source separation*).
- Considere a situação onde existem pessoas conversando na mesma sala, assim emitindo sinais de fala; ou telefones celulares emitindo suas ondas de rádio. Suponha que haja vários sensores ou receptores que estão em diferentes posições, sendo assim cada mistura das fontes originais é gravada com pesos um pouco diferentes.
- Com o objetivo de simplificar a exposição, digamos que existem três fontes que dão origem aos sinais, e também três sinais observados, denotados por $X_1(t)$, $X_2(t)$ e $X_3(t)$, os quais são as amplitudes dos sinais gravados no tempo t , e por $S_1(t)$, $S_2(t)$ e $S_3(t)$ os sinais originais. Os $X_i(t)$ são uma combinação linear dos $S_i(t)$ com coeficientes constantes a_{ij} , os quais dão as misturas dos pesos e que dependem da distância entre as fontes e os sensores e se supõe que sejam desconhecidos:

$$X_1(t) = a_{11}S_1(t) + a_{12}S_2(t) + a_{13}S_3(t), \quad (3)$$

$$X_2(t) = a_{21}S_1(t) + a_{22}S_2(t) + a_{23}S_3(t),$$

$$X_3(t) = a_{31}S_1(t) + a_{32}S_2(t) + a_{33}S_3(t).$$

Separação cega de fontes

- Em geral, não conhecemos os valores a_{ij} e a fonte do sinal $S_i(t)$ também é desconhecida. O que gostaríamos de fazer é encontrar os sinais originais a partir das misturas $X_i(t)$. Isso se chama **Separação Cega de Fontes** (*Blind Source Separation*).
- O termo **cega** significa que temos pouca ou nenhuma informação a respeito das fontes.
- Supomos que os coeficientes de mistura a_{ij} sejam diferentes o suficiente, para tornar a matriz que formam invertível. Então, existe uma matriz **W** com coeficiente w_{ij} , tal que podemos escrever os S_i como:

$$S_i(t) = w_{i1}X_1(t) + w_{i2}X_2(t) + w_{i3}X_3(t), \quad i = 1, 2, 3, \quad (4)$$

- Tal matriz **W** pode ser encontrada como a inversa da matriz que consiste dos coeficientes de mistura a_{ij} em (3) se conhecermos os coeficientes a_{ij} .

Separação cega de fontes

- A questão agora é: como podemos estimar os coeficientes w_{ij} em (4) ? Observamos os sinais X_1 , X_2 e X_3 e queremos achar uma matriz \mathbf{W} de modo que a representação seja dada pelos sinais da fonte original S_1 , S_2 e S_3 .
- Uma solução simples ao problema pode ser encontrada, considerando independência estatística dos sinais. De fato, se os sinais são não gaussianos, isto é suficiente para determinar os coeficientes w_{ij} dado que os sinais

$$Y_i(t) = w_{i1}X_1(t) + w_{i2}X_2(t) + w_{i3}X_3(t), \quad i = 1, 2, 3, \quad (5)$$

sejam estatisticamente independentes. Se os sinais Y_i , são independentes, então eles são iguais aos sinais originais S_i , $i = 1, 2, 3$ (a menos da multiplicação por um escalar). Usando apenas esta informação de independência estatística, podemos de fato estimar os coeficientes da matriz \mathbf{W} .

ICA – metodologia

- O que vimos acima nos leva a seguinte definição da ICA. Dado um vetor de variáveis aleatórias $\mathbf{X} = (X_1, X_2, \dots, X_p)^\top$, supõe-se que estas sejam geradas como uma mistura linear de componentes independentes $\mathbf{S} = (S_1, S_2, \dots, S_p)^\top$:

$$\begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_p \end{pmatrix} = \mathbf{A} \begin{pmatrix} S_1 \\ S_2 \\ \vdots \\ S_p \end{pmatrix}, \quad (6)$$

ou

$$\mathbf{X} = \mathbf{AS}, \quad (7)$$

onde $\mathbf{A} = [a_{ij}]$ é uma matriz desconhecida (chamada *mixing*).

- A análise de componentes independentes consiste agora em estimarmos tanto a matriz \mathbf{A} como os S_i , onde apenas observamos os X_i . Observe que supomos que o número de componentes independentes S_i é igual ao número de variáveis observadas; isso é uma simplificação que não é completamente necessária.

ICA – metodologia

- Pode ser demonstrado que este problema é bem definido, isto é, o modelo em (6) pode ser estimado se e somente se os componentes S_i são não gaussianos.
- Essa é uma necessidade fundamental que também explica a principal diferença entre ICA e análise fatorial, na qual a normalidade dos dados é levada em conta.
- De fato, ICA pode ser considerada como uma análise fatorial não gaussiana, visto que nesta também modelamos os dados como uma mistura linear de alguns fatores latentes.
- Além da estimação, tem-se que encontrar um algoritmo para executar os cálculos necessários. Como o princípio de estimação usa funções não quadráticas, os cálculos necessários, usualmente, não podem ser expressos usando álgebra linear simples, podendo ser extremamente complexos. Algoritmos numéricos são assim uma parte integral dos métodos de estimação ICA.

ICA – metodologia

- Os métodos numéricos são tipicamente baseados na otimização de algumas funções objetivas. O método básico de otimização é o método do gradiente. De interesse particular tem-se o algoritmo de ponto fixo chamado **FastICA**, que tem sido usado para explorar uma particular estrutura dos problemas ICA. Por exemplo, podemos usar esses métodos para encontrar o máximo de não normalidade como medido pelo valor absoluto do excesso de curtose.
- Estimando-se a matriz \mathbf{A} , supondo-se que exista sua inversa $\mathbf{A}^{-1} = \mathbf{W}$, obtemos

$$\mathbf{S} = \mathbf{W}\mathbf{X}. \quad (8)$$

- Há duas ambiguidades no procedimento da ICA. A primeira é que não podemos determinar as variâncias das S_i , pois se as multiplicarmos por uma constante C , basta dividir as colunas correspondentes de \mathbf{A} por C . Dessa maneira, o que fazemos é centrar as componentes, isto é, $E(S_i) = 0$, para todo i , e fixar as magnitudes das componentes independentes (c.i.'s), de modo que $E(S_i^2) = 1$, para todo i .

ICA – metodologia

- A segunda ambiguidade é que, contrariamente ao que ocorre com a ACP, não podemos determinar a ordem das c.i., pois como \mathbf{S} e \mathbf{A} são desconhecidas, podemos mudar livremente a ordem dos termos em (7).
- Por que variáveis gaussianas não são permitidas? Para responder, considere o caso particular (3), suponha que \mathbf{A} seja **ortogonal** e que as $S_i(t)$ sejam gaussianas. Então, as variáveis $X_i(t)$ são gaussianas (combinações lineares de v.a.s independentes e gaussianas), não correlacionadas (por que?) e de variância unitária. Sua densidade conjunta é dada por

$$f(x_1, x_2, x_3) = \frac{1}{(2\pi)^{3/2}} e^{-(x_1^2 + x_2^2 + x_3^2)/2},$$

que é simétrica, logo não contém nenhuma informação sobre as direções das colunas de \mathbf{A} . Logo, \mathbf{A} não pode ser estimada, ou ainda, \mathbf{A} não é identificada para componentes gaussianas independentes.

ICA – metodologia

- Seja $y = \mathbf{v}' \mathbf{X} = \sum_{i=1}^p v_i X_i$, onde \mathbf{v} é um vetor $p \times 1$. Se \mathbf{v} fosse uma das linhas de $\mathbf{A}^{-1} = \mathbf{W}$, essa combinação linear seria uma das c.i.'s S_i . O problema reduz-se, então, a determinar um vetor \mathbf{v} nessas condições.
- Pode-se provar que maximizando-se a não gaussianidade de $\mathbf{v}' \mathbf{X}$ nos dá uma das componentes de \mathbf{S} . Para tanto, teremos que obter $2p$ máximos locais, 2 para cada c.i. (correspondendo a S_i e $-S_i$). Como as c.i.'s são não correlacionadas, podemos restringir a busca no espaço que fornece estimativas não correlacionadas com as anteriores e isso corresponde a **branquear (whitening)** as observações.
- Como obter componentes independentes? Podemos escolher duas formas como proxy de independência , que por sua vez determinam a forma do algoritmo ICA a usar:
 1. Minimização da informação mútua (MIM);
 2. Maximização da não gaussianidade (MNG).
- A família de algoritmos que usa MIM utiliza medidas como a **Divergência de Kullback-Leibler** e **Máxima Entropia**. A família que aborda a não gaussianidade, usa **curtose** e **negentropia**.

ICA – Pré-processamento

- Como passos de pré-processamento, os algoritmos usam centragem (subtrair a média para obter um sinal de média zero), branqueamento e redução da dimensionalidade, usualmente via ACP e decomposição em valores singulares. O branqueamento assegura que todas as dimensões são tratadas igualmente antes que o algoritmo seja aplicado.
- Se a $\text{Cov}(X, Y) = E(XY) - E(X)E(Y) = 0$ dizemos que X e Y são não correlacionadas. Se X e Y são independentes, então são não correlacionadas. Porém, não correlação não implica em independência, ou seja covariância zero não implica necessariamente em independência, a não ser que (X, Y) tenham distribuição gaussiana bivariada.
- Uma propriedade um pouco mais forte que não correlação é **brancura** (*whiteness*). Branqueamento de um vetor de média zero, \mathbf{y} , significa que seus componentes são não correlacionados e suas variâncias são unitárias. Em outras palavras, a matriz de covariância (assim como a matriz de correlação) de \mathbf{y} é igual a matriz identidade:

$$E[\mathbf{y}\mathbf{y}^\top] = \mathbf{I}. \quad (9)$$

ICA – Pré-processamento

- Consequentemente, branqueamento significa que transformamos linearmente o vetor de dados observados \mathbf{x} através de uma multiplicação linear com alguma matriz \mathbf{V}

$$\mathbf{z} = \mathbf{V}\mathbf{x}, \quad (10)$$

obtendo então um novo vetor \mathbf{z} que é **branco**. O branqueamento é sempre possível. Um método bastante popular é usando a **decomposição em valores singulares** (*singular value decomposition-SVD*) da matriz de covariâncias da variável centrada, digamos $\tilde{\mathbf{X}}$:

$$E[\tilde{\mathbf{X}}\tilde{\mathbf{X}}^\top] = \mathbf{E}\mathbf{D}\mathbf{E}^\top, \quad (11)$$

onde \mathbf{E} é a matriz ortogonal de autovetores de $E[\tilde{\mathbf{X}}\tilde{\mathbf{X}}^\top]$ e \mathbf{D} é a matriz diagonal de seus autovalores, $\mathbf{D} = \text{diag}(d_1, \dots, d_p)$.

- Branqueamento pode agora ser feito pela matriz de branqueamento

$$\mathbf{V} = \mathbf{E}\mathbf{D}^{-1/2}\mathbf{E}^\top. \quad (12)$$

ICA – Pré-processamento

- O branqueamento transforma \mathbf{A} em $\tilde{\mathbf{A}}$, tal que

$$\tilde{\mathbf{X}} = \mathbf{ED}^{-1/2} \mathbf{E}^\top \mathbf{A} \mathbf{S} = \tilde{\mathbf{A}} \mathbf{S}.$$

- O procedimento torna $\tilde{\mathbf{A}}$ ortogonal, pois

$$E[\tilde{\mathbf{X}} \tilde{\mathbf{X}}^\top] = \tilde{\mathbf{A}} E(\mathbf{S} \mathbf{S}^\top) \tilde{\mathbf{A}}^\top = \tilde{\mathbf{A}} \tilde{\mathbf{A}}^\top = \mathbf{I}.$$

- Ainda, o processo de branqueamento reduz o número de parâmetros a estimar, de p^2 para $p(p - 1)/2$, devido à ortogonalidade da matriz $\tilde{\mathbf{A}}$.

ICA – Algoritmos

- Há vários algoritmos para a análise de componentes independentes de um conjunto de dados: Algoritmo do Gradiente usando a Curtose, Algoritmo de Ponto Fixo usando Curtose, Algoritmo do Gradiente usando Negentropia, Algoritmo de Ponto Fixo usando Negentropia, Algoritmos para Estimação de Máxima Verossimilhança. Veja Gegembauer (2010) para detalhes.
- Algoritmos tradicionalmente utilizados para a ICA incluem [infomax](#), [FastICA](#) e [JADE](#).
- O pacote [ica](#), no R, contempla esses algoritmos.
- Outro pacote: [ProDenICA](#), este desenvolvido por T. Hastie e R. Tibshirani, que apresenta a particularidade de estimar a densidade de cada componente.

ICA – Algoritmos

- Um número ξ diz-se um **ponto fixo** de uma função φ se $\varphi(\xi) = \xi$.
- Exemplo: $\varphi(x) = x^2 - 2$ tem dois pontos fixos, -1 e 2
- Encontrar pontos fixos e raízes de polinômios algébricos são problemas equivalentes.
- Se existir um ponto fixo de uma função $\varphi(x)$ num intervalo $[a, b]$, ele pode ser encontrado pelo **algoritmo do ponto fixo**:

$$x^{(k+1)} = \varphi(x^{(k)}), \quad k \geq 0. \quad (13)$$

O algoritmo pode não convergir.

- O método de Newton-Raphson é um algoritmo de ponto fixo.
- **Teorema de convergência do ponto fixo:** Se φ for contínua e $x^{(k)}$ gerada por (13) e se existir ξ tal que $\lim_{k \rightarrow \infty} x^{(k)} = \xi$, então ξ é um ponto fixo de φ .

ICA – Algoritmos

- O pacote **FastICA**, desenvolvido por Hyvärinen and Oja (1997), usa uma aproximação da negrentropia e o algoritmo é baseado em iterações de ponto fixo. É mais robusto que métodos baseados na curtose.
- **JADE**: Joint approximate diagonalization of eigenmatrices. Usa métodos matriciais.
- **InfoMax**: pacote que minimiza a **informação mútua**:

$$I(X, Y) = H(X) - H(X|Y), \quad (14)$$

onde $H(X|Y)$ é a entropia condicional ($H(X, Y) - H(Y)$) e $H(X)$ é a entropia de X .

- No caso contínuo,

$$I(X, Y) = \int f(x, y) \log \frac{f(x, y)}{f_X(x)f_Y(y)} dx dy. \quad (15)$$

- $I(X, Y) = 0$ se, e somente se, X e Y independentes.
- Amari et al. (1996): algoritmo InfoMax.

ICA – Algoritmos

- Uma maneira de extrair as componentes é forçá-las a serem as mais não normais possíveis.
- Negentropia pode ser usada para estimar não-normalidade; é uma medida de distância da normalidade:

$$N(X) = H(X_{\text{normal}}) - H(X). \quad (16)$$

- Para uma dada matriz de covariância, a distribuição que tem a maior entropia é a gaussiana. Assim, a negentropia é uma medida estritamente positiva da não-gaussianidade.
- Como é difícil calcular a negentropia pela definição (16), usa-se a aproximação de Hyvärinen and Oja.

ICA – Exemplo 1

- **Exemplo 1.** Vamos retomar o exemplo visto na seção de Análise Fatorial, sobre poluição, com 9 variáveis.
- Usaremos o pacote `ica` do R, que tem notação diferente para as diferentes matrizes. Em cada caso faremos a correspondência entre as notações.
- A matriz **M** (*mixing*) estimada, de ordem 9×9 , correspondente à matriz **A** do texto, é

```
[,1]          [,2]          [,3]          [,4]          [,5]          [,6]          [,7]          [,8]
[1,] -139.3790190 -149.7021007 -325.3324853 1077.925146 155.3171887 68.243758 111.7946050 -274.782659
[2,] -0.4866963  -0.2949352 -0.5650103  1.190811  0.3524588 0.3555581 0.4229097 -0.879921
[3,] -12.4332654  2.4281842 -25.8117720  42.551213 13.2787874 13.166554 16.9792887 -77.566097
[4,] -42.4416295 -15.1698381 -86.3634728 405.249890 356.4777419 14.649072 -24.5665422 -99.761388
[5,] -264.2307634 -730.1007230 224.9821643 -230.382230 38.0537401 123.889785 9.4797710 87.720764
[6,] -109.9753576 -35.9892578 -184.6838704 661.491067 263.0009705 33.477923 308.4517253 -40.708621
[7,] -121.4505476  21.0283925 -136.7716779 -19.803719 5.3052447 19.311117 15.3286940 2.762610
[8,] -17.9760476 -19.9296505  2.3521578 10.763812 14.0445497 232.900590 -0.7320548 4.559599
[9,] -7362.9405123 3856.0539441 3007.1595557 -1949.521311 -1347.9118353 1263.932257 -977.7231474 -677.758695
[,9]
[1,] 15.571397
[2,] -1.274516
[3,] -8.172252
[4,] -1.558269
[5,] 227.872428
[6,] 57.382165
[7,] 14.380424
[8,] 6.372196
[9,] -609.643712
```

ICA – Exemplo 1

- Temos, por exemplo,

$$X_1 = -139,379S_1 - 149,702S_2 + \dots + 15,571S_9.$$

- O programa também fornece as matrizes :

\mathbf{S} , de ordem 193×9 , matriz estimada das componentes independentes, obtida por

$$\mathbf{S} = \mathbf{W}\mathbf{X} = \mathbf{Y}\mathbf{R};$$

\mathbf{W} , de ordem 9×9 , matriz *un-mixing*, tal que $\mathbf{X}\mathbf{Y}\mathbf{W} = \mathbf{S}$. \mathbf{W} é escolhida de modo a maximizar a negentropia, com a restrição de que \mathbf{W} seja ortogonal;

\mathbf{Y} , de ordem 193×9 , matriz pré-branqueada;

\mathbf{Q} , de ordem 9×15 , matriz de pré-branqueamento, tal que $\mathbf{Y} = \mathbf{Q}\mathbf{X}$;

\mathbf{R} , de ordem 9×9 ; matriz de rotação ortogonal, tal que $\mathbf{S} = \mathbf{Y}\mathbf{R}$.

ICA – Exemplo 1

- Fornece, também:

vafs : variância explicada por cada c.i.

alg : algoritmo usado (parallel ou deflation)

fun : função contraste (para aproximar a negentropia)

alpha : tuning parameter

iter : número de iterações do algoritmo

- As variâncias explicadas pelas c.i's são dadas por

0.600985581 0.170645639 0.102434106 0.062162898 0.022555357

0.018513146 0.011778296 0.006196794 0.004728182

- Vemos que as três primeiras componentes explicam cerca de 88% da variância dos dados. Na Figura 1 temos os gráficos das primeiras duas componentes independentes.

ICA – Exemplo 1

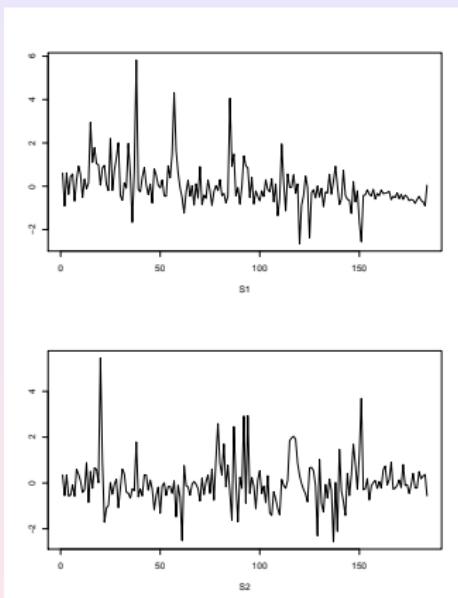


Figura: Gráficos das duas primeiras c.i.'s para o Exemplo 13.3.

ICA – Exemplo 1

A matriz **W** estimada é dada por

```
[,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,] -1.750287e-04 -0.074821293 0.0018890899 -1.219544e-04 -4.394832e-04 -9.771709e-05 -1.951470e-03 9.258294e-04
[2,] -3.688197e-04 -0.193146765 0.0014816133 1.790061e-04 -1.018530e-03 5.109103e-04 -8.073011e-05 5.379454e-04
[3,] -2.673837e-04 -0.024630937 0.0003486251 1.736989e-05 2.822484e-04 6.088192e-04 -5.353500e-03 -1.075091e-04
[4,] 1.176365e-03 -0.020637504 -0.0041025331 -1.933696e-04 -1.798993e-04 -3.679419e-05 -1.609738e-03 5.375905e-05
[5,] -1.511512e-03 0.035410750 0.0012706791 2.872724e-03 1.468919e-04 5.799778e-04 5.234842e-04 1.298423e-05
[6,] -2.234726e-05 -0.004478221 0.0004852519 -1.619936e-04 -1.281089e-04 -1.639626e-05 -1.038226e-04 4.386960e-03
[7,] -1.548268e-03 0.105156093 0.0053742429 -2.024956e-03 9.977071e-05 3.208043e-03 -9.018445e-04 -2.629357e-04
[8,] 1.119943e-04 0.110425897 -0.0148241124 -5.879721e-05 -1.557664e-04 5.604000e-04 1.003537e-03 5.855453e-04
[9,] 3.545273e-04 -0.752508234 0.0074526851 -6.531333e-05 2.558401e-04 4.482189e-04 7.568601e-04 3.558750e-04
[,9]
[1,] -8.288335e-05
[2,] 4.515382e-05
[3,] 7.534598e-05
[4,] 2.056474e-05
[5,] -1.503345e-05
[6,] -3.322196e-06
[7,] -1.102442e-05
[8,] -4.811093e-06
[9,] 1.592838e-06
```

Assim, por exemplo,

$$S_1 = -0,000175X_1 - 0,0748X_2 + \dots + 0,0000829X_9.$$

Vemos que somente as variáveis $X_3=Zn$, e $X_4=Ba$ são as mais importantes para explicar as componentes independentes S_i , $i = 1, \dots, 9$.

ICA – Exemplo 1

O pacote ProDenICA do R, fornece as densidades estimadas das c.i.'s, mostradas na Figura 2.

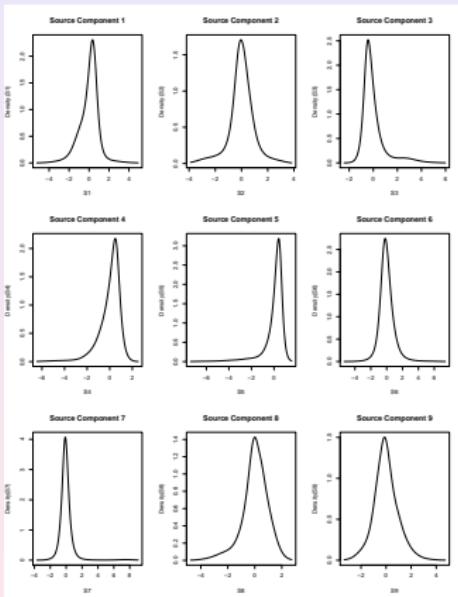


Figura: Gráficos das densidades das c.i.'s para o Exemplo 1.

ICA – Exemplo 1

Vamos considerar, agora, três componentes independentes. A matriz **A** agora fica

	[,1]	[,2]	[,3]
[1,]	130.0588599	-1172.278514	-50.1191462
[2,]	0.4506871	-1.538268	-0.2837152
[3,]	17.7456231	-64.094123	-17.9385466
[4,]	46.0218183	-489.157909	-30.7345628
[5,]	215.8328265	141.834601	846.2085520
[6,]	85.0325691	-771.625038	-24.1875542
[7,]	76.2477919	-26.575128	-18.3258463
[8,]	61.9279843	-24.349130	49.9983656
[9,]	8353.6861764	3325.603526	-2506.6567435

ICA – Exemplo 1

- A matriz **W** fica

```
[,1]          [,2]          [,3]          [,4]          [,5]          [,6]          [,7]          [,8]
[1,] 0.0002104413 2.494246e-07 7.925442e-06 8.257729e-05 3.307134e-04 1.416700e-04 1.255409e-05 3.473013e-05
[2,] -0.0005069541 -7.063526e-07 -2.903874e-05 -2.097610e-04 -8.165741e-06 -3.338968e-04 -2.364408e-05 -2.260694e-05
[3,] 0.0000287935 -1.201862e-07 -1.273793e-05 -2.839277e-06 1.091458e-03 2.931156e-05 7.179379e-06 8.385744e-05
[,9]
[1,] 1.056008e-04
[2,] 1.310325e-05
[3,] -2.959094e-05
```

- A variância explicada por cada c.i. é mostrada abaixo:

```
[1] 0.77283297 0.14705258 0.07750351
```

- Vemos que as duas primeiras c.i.'s explicam 92% da variância dos dados. Usando-se a função **fastICA** do pacote **fastICA** do R, podemos fazer alguns gráficos.
- A Figura 3 mostra os dados pré-processados, as componentes principais estimadas e as componentes independentes estimadas. Na Figura 4 temos os gráficos das três componentes.

ICA – Exemplo 1

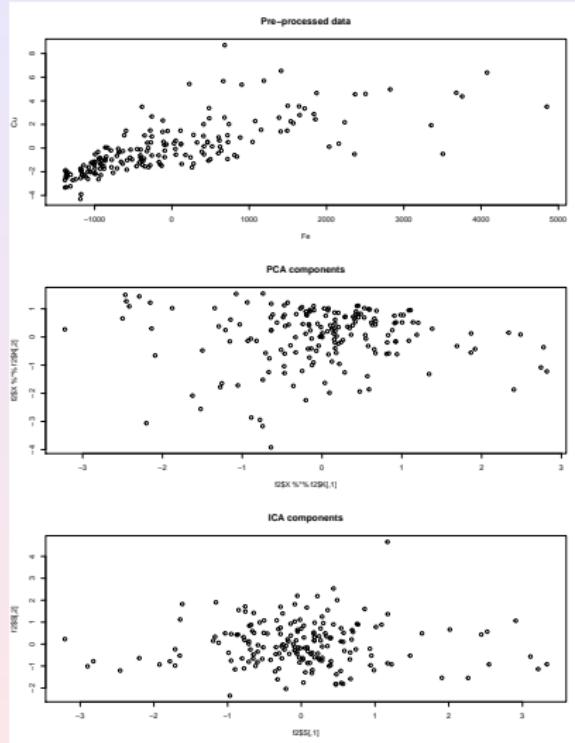


Figura: Gráficos dos dados, CP's e CI's para o Exemplo 1.

ICA – Exemplo 1

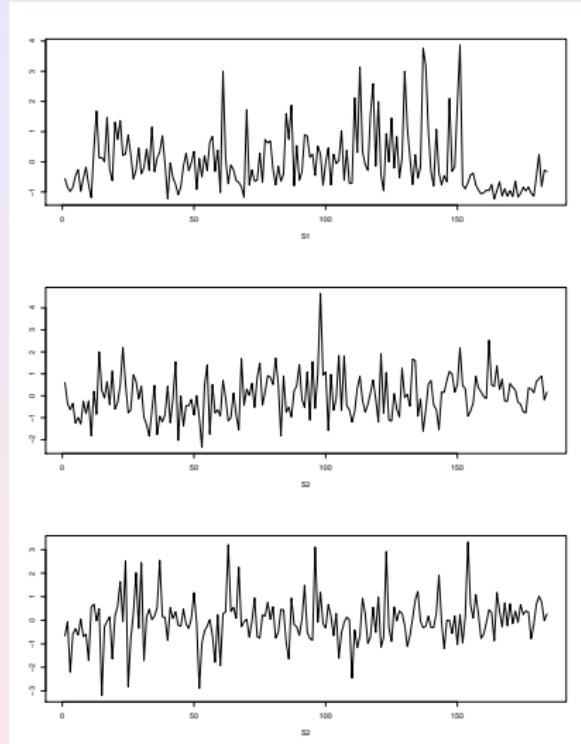


Figura: Gráficos das três componentes independentes.

Decomposição espectral

- Seja \mathbf{A} uma matriz quadrada de ordem m . Segue-se que $|\mathbf{A} - \lambda\mathbf{I}|$ é um polinômio de ordem m em λ e terá m raízes complexas $\lambda_1, \dots, \lambda_m$. Essas raízes são chamadas *raízes características* ou *autovalores* de \mathbf{A} .
- Como $\mathbf{A} - \lambda_j\mathbf{I}$ é singular, $j = 1, \dots, m$, existe um vetor \mathbf{a}_j , cujas coordenadas não são todas nulas, tal que $(\mathbf{A} - \lambda_j\mathbf{I})\mathbf{a}_j = \mathbf{0}$, ou seja, $\mathbf{A}\mathbf{a}_j = \lambda_j\mathbf{a}_j$, $j = 1, \dots, m$. Os vetores $\mathbf{a}_1, \dots, \mathbf{a}_m$ são chamados *vetores característicos* ou *autovetores* de \mathbf{A} .
- Os seguintes resultados são válidos.
 - (1) O posto de \mathbf{A} , $\rho(\mathbf{A})$, dá o número de autovalores de \mathbf{A} não nulos.
 - (2) $\text{tr}(\mathbf{A}) = \sum_{j=1}^m \lambda_j$.
 - (3) $|\mathbf{A}| = \prod_{j=1}^m \lambda_j$.
 - (4) Se \mathbf{A} é uma matriz simétrica, real, todos os seus autovalores são reais, e para cada autovalor real existe um autovetor real.
 - (5) Se \mathbf{A} é simétrica, real, os autovetores correspondentes a autovalores distintos são ortogonais.
 - (6) Se \mathbf{A} é não negativa definida, então $\lambda_j \geq 0$, $j = 1, \dots, m$.
 - (7) Se \mathbf{A} é simétrica, de ordem $m \times m$, existe uma matriz ortogonal \mathbf{X} , tal que

$$\mathbf{X}' \mathbf{A} \mathbf{X} = \Lambda = \text{diag}\{\lambda_1, \dots, \lambda_m\},$$

ou $\mathbf{A} = \mathbf{X} \Lambda \mathbf{X}'$, onde os λ_j são os autovalores de \mathbf{A} e as colunas de \mathbf{X} são os correspondentes autovetores.

Decomposição espectral

- O resultado (7) é chamado o *teorema espectral* para matrizes simétricas. Segue-se que a *decomposição espectral* de \mathbf{A} é dada por

$$\mathbf{A} = \sum_{j=1}^m \lambda_j \mathbf{x}_j \mathbf{x}_j'$$

onde \mathbf{x}_j é o autovetor correspondente a λ_j .

- Se \mathbf{A} é uma matriz quadrada de ordem m , positiva definida, existe uma matriz triangular inferior \mathbf{T} , com elementos da diagonal principal positivos, tal que

$$\mathbf{T}^{-1} \mathbf{A} (\mathbf{T}')^{-1} = \mathbf{I}_m, \quad \text{ou} \quad \mathbf{A} = \mathbf{T} \mathbf{T}'.$$

- Esta decomposição de \mathbf{A} é chamada *decomposição de Cholesky*.

Decomposição em valores singulares - DVS

- DVS é uma fatoração de uma matriz qualquer em outras três matrizes com características importantes.
- Dada uma matriz \mathbf{A} , de ordem $m \times n$, $m \geq n$, não necessariamente de posto máximo, uma DVS de \mathbf{A} é uma fatoração da forma $A = \mathbf{U}\Sigma\mathbf{V}^\top$, onde:
 - \mathbf{U} , de ordem $m \times n$, é ortogonal;
 - \mathbf{V} , de ordem $m \times n$, é ortogonal;
 - Σ é diagonal se $m = n$, caso contrário adicionam-se $m - n$ linhas de zeros em Σ .
 - Os valores $\sigma_1, \dots, \sigma_n$ da diagonal principal de Σ são chamados **valores singulares** de Σ e $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n > 0$.
 - As colunas de \mathbf{U} , u_1, \dots, u_m são chamados **vetores singulares à esquerda** de \mathbf{A} , enquanto as colunas de \mathbf{V} , v_1, \dots, v_n são chamados **vetores singulares à direita** de \mathbf{A} , de tal forma que $\mathbf{A}v_j = \sigma_j u_j$.

Decomposição em valores singulares - DVS

- Decomposição por Autovalores $\mathbf{A} = \mathbf{X}\Lambda\mathbf{X}^{-1}$;
 - usa a mesma base \mathbf{X} para os espaços linha e coluna;
 - geralmente não usa base ortonormal;
 - é definida somente para matrizes quadradas.
- DVS
 - usa bases diferentes \mathbf{V} , \mathbf{U} para os espaços linha e coluna;
 - usa bases ortonormais;
 - existe para todas as matrizes.
- Entretanto, para matrizes simétricas positivas definidas, a decomposição por autovalores e a DVS são idênticas.

Aplicações da DVS

- Cálculo de propriedades de matrizes;
- Aproximação de \mathbf{A} por matrizes de posto baixo;
- Processamento de Sinais e Imagens
 - Compressão de Imagens;
 - Eliminação de ruídos (ruídos normalmente possuem σ_j pequenos);
 - Recuperação de Informações.

Curtose

- 1) Seja X uma variável aleatória qualquer, com média μ e variância σ^2 . Então, a assimetria de X é definida por

$$A(X) = E \left(\frac{(X - \mu)^3}{\sigma^3} \right), \quad (17)$$

enquanto a curtose de X é definida por

$$K(X) = E \left(\frac{(X - \mu)^4}{\sigma^4} \right). \quad (18)$$

- 2) Para uma distribuição normal, $A = 0$ e $K = 3$, donde a quantidade $e(X) = K(X) - 3$ ser chamada excesso de curtose. Distribuições com caudas pesadas têm curtose maior do que 3, e esta pode mesmo ser infinita.

Curtose

- 3) Com uma amostra X_1, \dots, X_T de X , considere o r -ésimo momento amostral

$$m_r = \frac{1}{T} \sum_{t=1}^T (X_t - \bar{X})^r,$$

onde $\hat{\mu} = \bar{X}$. Substituindo os momentos verdadeiros de X pelos respectivos momentos amostrais, obtemos os estimadores

$$\hat{A}(X) = \frac{m_3}{m_2^{3/2}} = \frac{1}{T} \sum_{t=1}^T \left(\frac{X_t - \bar{X}}{\hat{\sigma}} \right)^3, \quad (19)$$

$$\hat{K}(X) = \frac{m_4}{m_2^2} = \frac{1}{T} \sum_{t=1}^T \left(\frac{X_t - \bar{X}}{\hat{\sigma}} \right)^4, \quad (20)$$

respectivamente, onde $\hat{\sigma}^2 = \sum_{t=1}^T (X_t - \bar{X})^2 / T$. Segue-se que $\hat{e}(X) = \hat{K}(X) - 3$.

- 4) Pode-se provar que, se tivermos uma amostra de uma distribuição normal, e T for grande, então

$$\hat{A} \sim \mathcal{N}(0, 6/T), \quad \hat{K} \sim \mathcal{N}(3, 24/T). \quad (21)$$

Esses fatos podem ser utilizados para testar a normalidade de uma série (Jarque-Bera).

Referências

- Amari, S., Cichocki, A. and Yang, H. H. (1996). A new learning algorithm for blind signal separation. *Advances in Neural Information Processing Systems*, **8**, 757–763.
- Gegembauer, H. V. (2010). *Análise de Componentes Independentes com Aplicações em Séries Temporais Financeiras*. Dissertação de Mestrado, IME-USP.
- Hyvärinen, A. and Oja, E. (1997). A fast fixed point algorithm for independent component analysis. *Neural Computation*, **9**, 1483–1492.
- Hyvärinen, A. (1999). Fast and robust fixed-point algorithm for independent component analysis. *IEEE Transactions on Neural Network*, **10**, 626–634.
- Hyvärinen, A. and Oja, E. (2000). Independent component analysis: Algorithms and applications. *Neural Networks*, **13**, 411–430.
- Hyvärinen, A., Karhunen, J. and Oja, E. (2001). *Independent Component Analysis*. New York: Wiley.

MAE 5905: Introdução à Ciência de Dados

Pedro A. Morettin

Instituto de Matemática e Estatística
Universidade de São Paulo
pam@ime.usp.br
<http://www.ime.usp.br/~pam>

Aula 16 - Parte 1

13 de junho de 2023

Sumário

1 Redes Neurais

2 Perceptron

3 RN com uma camada

Preliminares

- Uma **rede neural** (RN) é um conjunto de algoritmos construídos para identificar relações entre as variáveis de um conjunto de dados por intermédio de um processo que tenta imitar a maneira com que neurônios interagem no cérebro.
- Cada neurônio numa rede neural é uma função à qual dados são alimentados e transformados numa resposta como em modelos de regressão. Essa resposta é repassada para um ou mais neurônios da rede que ao final do processo produz um resultado ou recomendação.
- Quando falamos em arquitetura de uma RN, estamos nos referindo à maneira como os neurônios estão organizados em camadas. Há três classes diferentes de arquiteturas :
 1. **RN com uma única camada**. Nessa classe, há uma camada de entrada e uma de saída. Essa RN é do tipo **feedforward**. Essa rede também é chamada **perceptron**.
 2. **RN multicamadas**, também do tipo **feedforward**. São redes com uma ou mais camadas escondidas. Cada neurônio de uma camada tem como entradas os neurônios da camada precedente.
 3. **RN recorrentes**. Nesse caso, pelo menos um neurônio conecta-se com um neurônio da camada precedente, criando um **ciclo** (**feedback**).

História das RN

- As contribuições pioneiras para a área de Redes Neurais (também denominadas redes neurais, termo derivado de neurônio) foram as de McCulloch e Pitts (1943), que introduziram a unidade lógica com limiar (*thresholded logic unit*) e de Hebb (1949), que postulou a primeira regra para aprendizado organizado.
- Rosenblatt (1957) introduziu o **perceptron** e Widrow e Hoff (1960) o **adaline** (*adaptive linear neuron*).
- Depois, seguiu-se o que foi chamado de **primeiro inverno neural**. Minsky e Papert (1969) escreveram um livro, chamado *Perceptron*, em que apresentam o problema **XOR** (*exclusive OR*). O problema consistia em usar redes neurais (o perceptron) para prever saídas da lógica XOR, dadas duas entradas binárias, como na Tabela 1. Uma função XOR deve retornar um valor verdadeiro (1, ponto verde na figura) se as duas entradas não são iguais e um valor falso (0, ponto vermelho na Figura 1) se elas são iguais.

História das RN

Tabela 1: Problema XOR

Entrada 1	Entrada 2	Saída
0	0	0
0	1	1
1	1	0
1	0	1

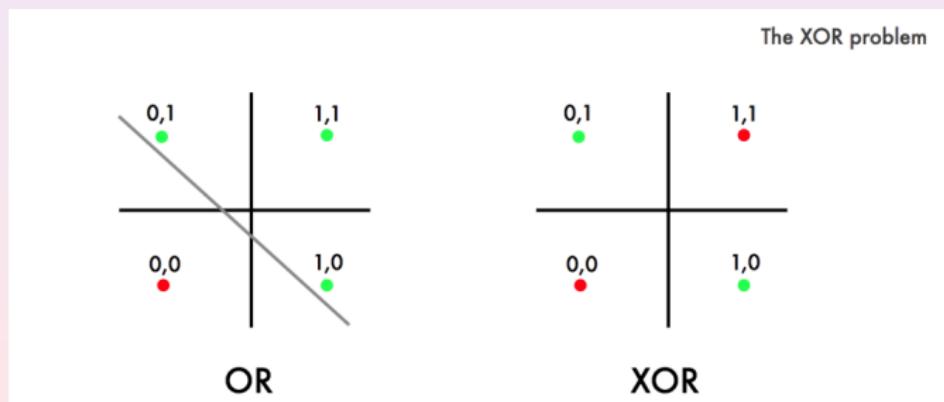


Figura 1: Os problemas lógicos (a) OR e (b) XOR.

História das RN

- Esse é um problema de classificação usando o perceptron, mas este supõe que as duas classes sejam linearmente separáveis, o que elas não são, no caso XOR, conforme a Figura 1. O problema OR (inclusive) é separável por uma reta.
- Este problema seria solucionado com a introdução, nas décadas de 1980 e 1990, das redes com várias camadas ocultas, o algoritmo de retroalimentação (*backpropagation*) e as redes neurais convolucionais. Veja LeCun (2015) para detalhes.
- Um **segundo inverno neural** ocorre na década de 1990, no qual a contribuição mais significativa, fora do contexto de redes neurais, foi a de Cortes e Vapnik (1995), que introduziram as máquinas de suporte vetorial (*support vector machines*).

História das RN

- A partir de 2006 foram desenvolvidas as **redes neurais profundas** (*deep neural networks*, DNN), com menção especial às **redes neurais recorrentes** (RNN) e as **redes neurais convolucionais** (CNN). É a chamada **era das GPU** (Graphic Processing Units).
- Em especial, destacamos uma competição promovida pela ImageNet, que mantém uma base de dados com milhões de imagens para fins de treinamento e classificação, usando técnicas de ciências de dados. Krizhevsky et al. (2012) venceram essa competição em 2012, usando uma CNN com sete camadas ocultas, denominada **Alex net** (primeiro nome do primeiro autor) e que levou seis dias para ser treinada. A proporção de erros dessa rede foi de 15,3%. Em 2015, o vencedor usou uma CNN com 150 camadas ocultas e a proporção de erros foi de 3,5%.
- A seguir faremos uma revisão dessas redes neurais.

Perceptron

- Rosenblatt (1958) introduziu o algoritmo *perceptron* como o primeiro modelo de aprendizado supervisionado. A ideia do algoritmo é atribuir pesos w_i aos dados de entrada \mathbf{x} , iterativamente, até que o processo tenha uma precisão pré-especificada para a tomada de decisão. No caso de classificação binária, o objetivo é classificar elementos do conjunto de dados com valor das variáveis preditoras \mathbf{x} (dados de entrada) em uma de duas classes, rotuladas aqui por +1 e -1.
- O algoritmo *perceptron* (programado para um computador IBM 704) foi implementado em uma máquina chamada Mark I, planejada para reconhecimento de imagens. O modelo subjacente consiste de uma combinação linear das entradas, \mathbf{x} , com a incorporação de um viés externo, cujo resultado é comparado com um limiar, definido por meio de uma **função de ativação**. As funções de ativação usualmente empregadas são as funções **degrau** ou **sigmoide**. Outras funções de ativação serão vistas a seguir.
- Se $\mathbf{x} = (1, x_1, x_2, \dots, x_p)^\top$ contém as entradas, $\mathbf{w} = (-b, w_1, w_2, \dots, w_p)^\top$ são os pesos, a saída é dada por

$$v = \sum_{i=0}^p w_i x_i = \mathbf{w}^\top \mathbf{x}.$$

Perceptron

- Definamos uma **função de ativação** $f(v)$ tal que se $f(v) \geq b$, a amostra é classificada na classe $+1$ e se $f(v) < b$, é classificada na classe -1 . Ou seja,

$$f(v) = \begin{cases} 1, & \text{se } v \geq b \\ -1, & \text{se } v < b. \end{cases}$$

- O parâmetro b é chamado de **viés**. A ideia de Rosenblatt era obter um algoritmo (regra de aprendizado), segundo a qual os pesos são atualizados a fim de se obter uma fronteira de decisão **linear**, que permite discriminar entre as duas classes linearmente separáveis. Veja a Figura 2.

Perceptron

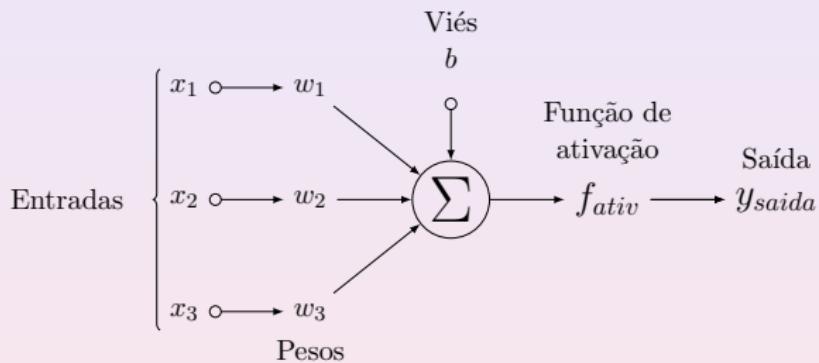


Figura 2: Diagrama de um perceptron.

Algoritmo do perceptron

- [1] Inicialize todos os pesos como sendo zero ou valores aleatórios pequenos;
- [2] Para cada amostra de treinamento $\mathbf{x}^{(i)}$:
 - (a) calcule os valores da saída;
 - (b) atualize os pesos.

A atualização dos pesos é feita segundo a regra de aprendizagem

$$\Delta w_j = \eta(\text{alvo}^{(i)} - \text{saída}^{(i)})x_j^{(i)},$$

onde η é a taxa de aprendizagem (um valor entre 0 e 1), **alvo** é o rótulo verdadeiro da classe e **saída** é o rótulo da classe prevista. Todos os pesos são atualizados simultaneamente. Por exemplo, no caso de duas variáveis, x_1 e x_2 , teremos que atualizar w_0 , w_1 e w_2 .

Algoritmo do perceptron

- É fácil ver que nos dois casos para os quais o perceptron prevê o rótulo da classe verdadeira, $\Delta w_j = 0$, para todo j . No caso de previsão errônea, $\Delta w_j = 2\eta x_j^{(i)}$ ou $\Delta w_j = -2\eta x_j^{(i)}$.
- Uma observação importante é que a convergência do perceptron somente é garantida se as duas classes são linearmente separáveis. Se não forem, podemos fixar um número máximo de passadas sobre os dados de treinamento (**épocas**) ou um limiar para o número máximo de classificações erradas tolerável.

Perceptron – Exemplo 1

Exemplo 1. Consideremos um exemplo simulado. Simulamos 50 valores de X_1 e X_2 , geradas segundo uma normal padrão e Y será igual a +1 se $X_2 > 0,5X_1 + 0,2$ e igual a -1, caso contrário. Veja a Figura 3.

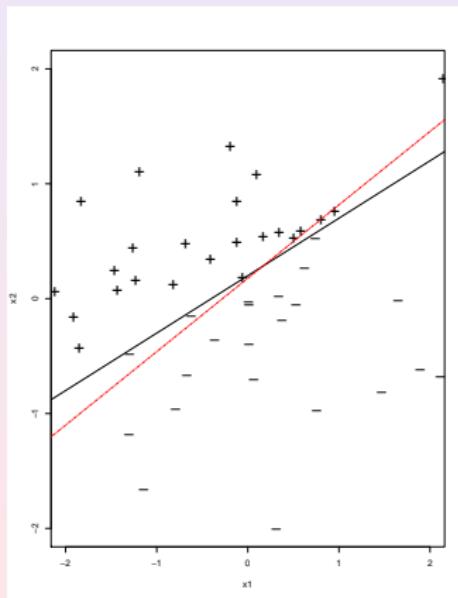


Figura 3. Exemplo 1: reta simulada (preto); perceptron (vermelho).

Perceptron – Exemplo 1

Usando o *script* para o perceptron (veja a página do livro), obtemos as estimativas dos coeficientes

\$w

x1	x2
-0.5384671	0.8426465

\$b

[1] -0.1495257

e, portanto a reta $x_2 = 0,17745 + 0,639x_1$, que está apresentada também na Figura 2. Ambas as retas estão próximas.

Perceptron – Exemplo 2

Exemplo 2. Vamos considerar o conjunto de dados [Iris](#) e duas variáveis,

x_1 : comprimento de sépala

x_2 : comprimento de pétala

e duas espécies, Iris versicolor (+1) e Iris setosa (-1). Veja a Figura 4.

Usando, novamente, o *script* para o perceptron, obtemos

\$w

```
sepal      petal
1 0.1419446 0.9898746
```

\$b

```
[1] -2.855304
```

de modo que obtemos a reta $x_2 = 2,8844 - 0,14335x_1$, representada na Figura 4.

Perceptron – Exemplo 2

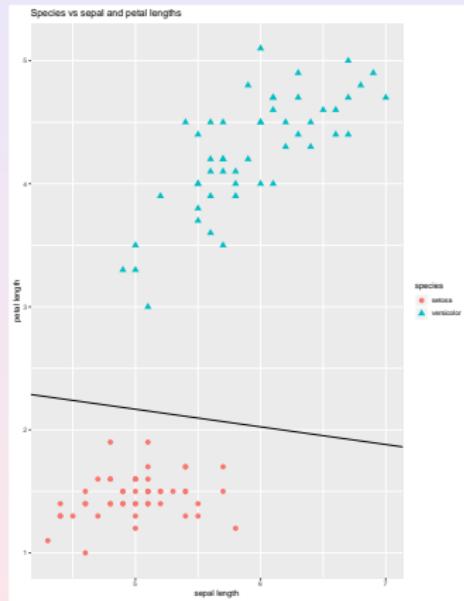


Figura 4. Iris data com duas variáveis e reta perceptron.

RN com uma camada

Atualmente, a RN mais simples consiste de entradas, de uma camada intermediária escondida e das saídas. Na Figura 5, se $K = 1$ temos o caso de regressão e, no caso de classificação, teremos K classes, sendo que $Z_i = -1$ ou $Z_i = +1$.

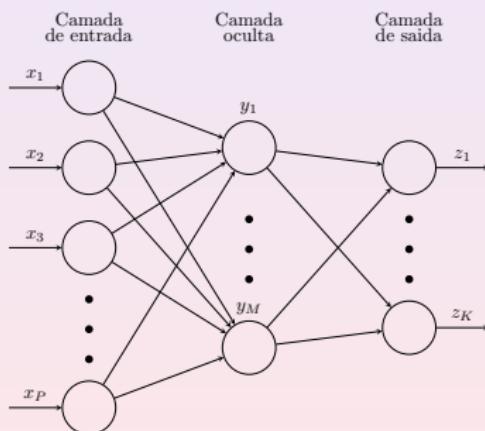


Figura 5. Rede neural com uma camada escondida.

RN com uma camada

- Consideremos os seguintes vetores de variáveis,
 $\mathbf{X} = (X_1, \dots, X_p)^\top$, $\mathbf{Y} = (Y_1, \dots, Y_M)^\top$ e $\mathbf{Z} = (Z_1, \dots, Z_K)^\top$ e sejam, os vetores de pesos $\boldsymbol{\alpha}_j$, $j = 1, \dots, M$, $\boldsymbol{\beta}_k$, $k = 1, \dots, K$, de ordens $p \times 1$ e $M \times 1$, respectivamente.
- A RN da Figura 5 pode ser representada pelas equações:

$$Y_j = h(\alpha_{0j} + \boldsymbol{\alpha}_j^\top \mathbf{X}), \quad j = 1, \dots, M, \quad (1)$$

$$Z_k = g(\beta_{0k} + \boldsymbol{\beta}_k^\top \mathbf{Y}), \quad k = 1, \dots, K. \quad (2)$$

- As funções h e g são chamadas **funções de ativação** e geralmente são usadas as seguintes funções:

RN com uma camada

a) logística (ou sigmoide),

$$f(x) = \frac{1}{1 + e^{-x}},$$

b) tangente hiperbólica,

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}},$$

c) ReLU (rectified linear unit),

$$f(x) = \max(0, x).$$

d) Uma modificação da ReLU é *leaky ReLU*, dada por

$$f(x) = \begin{cases} x, & \text{se } x > 0, \\ 0,01x, & \text{se } x < 0. \end{cases}$$

Funções de ativação

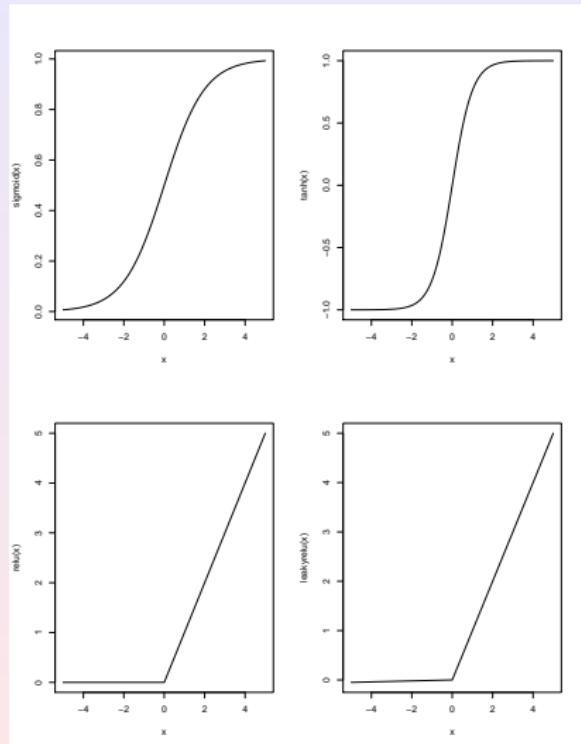


Figura 6. Algumas funções de ativação.

RN com uma camada

- A função (d) é bastante utilizada, pois é fácil de otimizar, o gradiente pode ser facilmente calculado e converge mais rápido do que sigmoides. Todavia, ela não é derivável na origem, e no algoritmo **backpropagation**, por exemplo, necessitamos de derivabilidade.
- Pode-se usar a mesma função de ativação em (1) e (2). Também é comum que haja uma saída única, Z , na Figura 5.
- Os pesos α_{0j} e β_{0k} têm o mesmo papel de b no perceptron e representam vieses. Os Y_j constituem a camada escondida e não são observáveis.
- No lugar de (1)–(2), podemos considerar a saída da RN expressa na forma

$$f(\mathbf{x}, \mathbf{w}) = \varphi \left(\sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) \right), \quad (3)$$

onde $\varphi(\cdot)$ é a identidade, no caso de regressão e uma função não linear, no caso de classificação; w_j são pesos a serem determinados.

RN com uma camada

Com essa formulação, os seguintes passos são usualmente utilizados na análise de redes neurais (RN) (Bishop, 2006):

- (i) Considere as ativações

$$a_j = \sum_{i=0}^p w_{ji}^{(1)} x_i, \quad j = 1, \dots, M, \quad (4)$$

onde incluímos os vieses $w_{j0}^{(1)}$ nos pesos $\mathbf{w}_j^{(1)} = (w_{j0}, w_{j1}, \dots, w_{jp})^\top$, $j = 1, \dots, M$, fazendo $x_0 = 1$. O índice (1) indica a primeira camada da RN.

- (ii) Cada ativação a_j é transformada usando uma função de ativação $h(\cdot)$, resultando

$$y_j = h(a_j). \quad (5)$$

Aqui, podemos usar uma das funções acima. Dizemos que os y_j são as unidades escondidas.

RN com uma camada

(iii) Considere as ativações de saída

$$a_k = \sum_{j=0}^M w_{kj}^{(2)} z_j, \quad k = 1, 2, \dots, K, \quad (6)$$

onde, novamente incluimos os vieses $w_{k0}^{(2)}$ no vetor \mathbf{W} .

(iv) Finalmente, essas ativações são transformadas por meio de uma nova função de ativação, resultando nas saídas z_k da RN, sendo que no caso de regressão, $z_k = a_k$ e no caso de classificação,

$$Z_k = \sigma(a_k), \quad (7)$$

sendo que $\sigma(a)$ em geral é a logística dada em (a) acima.

RN com uma camada

- Combinando, obtemos

$$f_k(\mathbf{x}, \mathbf{W}) = \sigma \left(\sum_{j=0}^M w_{kj}^{(2)} h \left(\sum_{i=0}^p w_{ji}^{(1)} x_i \right) \right). \quad (8)$$

- O procedimento para obter (8) é chamado **forward propagation** da informação.
- Podemos generalizar considerando camadas adicionais.
- A nomenclatura de tal rede difere segundo autores e pacotes computacionais. Pode ser chamada de RN com 3 camadas ou uma RN com uma camada escondida. Bishop (2006) sugere chamá-la de uma RN com duas camadas, referindo-se aos pesos $w_{ji}^{(1)}$ e $w_{kj}^{(2)}$. Ainda, o pacote **neuralnet** estipula o número de neurônios, M , na camada escondida.

RN com uma camada–Exemplo 3

- **Exemplo 3.** Vamos considerar os dados do Exemplo 1 e vamos criar uma rede neuronal com uma camada escondida com 3 neurônios e função de ativação tangente hiperbólica. Para tanto, usamos o pacote **neuralnet** do R.
- ```
nn=neuralnet(y~x1+x2,data=df, hidden=3, act.fct="tanh",
+linear.output=FALSE)# hidden=3: single layer with 3 neurons
plot(nn)
```
- Obtemos a Figura 7. Nesse exemplo,  $p = 2$ ,  $M = 3$  e  $K = 1$ . As equações (3)–(4) ficam:

$$\begin{aligned}Y_1 &= -4,179 - 2,761X_1 - 19,010X_2 \\Y_2 &= 2,650 - 2,109X_1 - 0,978X_2 \\Y_3 &= 0,582 + 1,472X_1 - 1,232X_2.\end{aligned}$$

- Também, teremos três vetores  $\alpha_1, \alpha_2$  e  $\alpha_3$ , de ordens  $2 \times 1$ ,  $\mathbf{X} = (X_1, X_2)$ ,  $\mathbf{Y} = (Y_1, Y_2, Y_3)$  e  $Z = g_1(W) = W = \beta_{01} + \beta_1^\top \mathbf{Y}$ , com  $\beta_1 = (\beta_1, \beta_2, \beta_3)$ ,  $\beta_{01} = -0,540$ . Ou seja,

$$Z = -0,540 - 3,019Y_1 - 3,538Y_2 - 21,641Y_3.$$

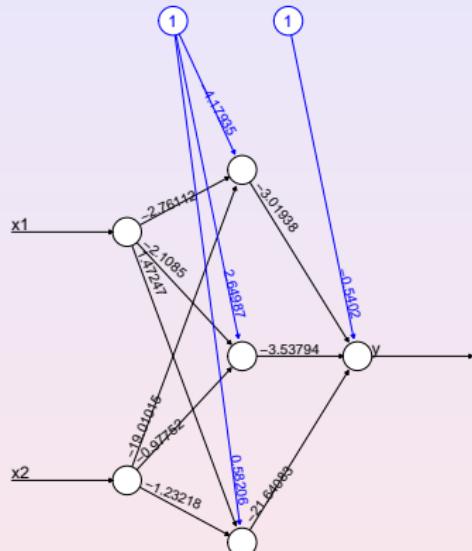
## RN com uma camada—Exemplo 3

Os pesos finais são dados por:

```
result.matrix
```

|                       | [ ,1 ]        |
|-----------------------|---------------|
| error                 | 0.003675542   |
| reached.threshold     | 0.007275895   |
| steps                 | 217.000000000 |
| Intercept.to.1layhid1 | -4.179347633  |
| x1.to.1layhid1        | -2.761123538  |
| x2.to.1layhid1        | -19.010153927 |
| Intercept.to.1layhid2 | 2.649870079   |
| x1.to.1layhid2        | -2.108496830  |
| x2.to.1layhid2        | -0.977518529  |
| Intercept.to.1layhid3 | 0.582063062   |
| x1.to.1layhid3        | 1.472465125   |
| x2.to.1layhid3        | -1.232181252  |
| Intercept.to.y        | -0.540201763  |
| 1layhid1.to.y         | -3.019378642  |
| 1layhid2.to.y         | -3.537936157  |
| 1layhid3.to.y         | -21.640828433 |

## RN com uma camada—Exemplo 3



Error: 0.003676 Steps: 217

Figura 7. Rede neural com uma camada escondida para o Exemplo 3.

## RN com uma camada—Exemplo 3

Vamos ver como fazer previsões para um conjunto de teste. Suponha que tenhamos 5 observações de  $x$ , a saber

| $x_1$      | $x_2$       |
|------------|-------------|
| 0.6180602  | 0.14261227  |
| 0.5053168  | 1.48586481  |
| 0.3615404  | -0.04880704 |
| -0.1707795 | -1.09666571 |
| -0.6284641 | -0.54107065 |

que chamaremos de *test*. Usamos os comandos

```
test=data.frame(x1,x2)
```

```
Predict=compute(nn,test)
```

```
Predict$net.result
```

```
[,1]
[1,] -1.0000000
[2,] 1.0000000
[3,] 1.0000000
[4,] -0.9993473
[5,] -0.9997057
```

## RN com uma camada–Exemplo 3

Convert probabilities into binary classes setting threshold level =0.5

```
prob<-Predict\$/net.result
pred<-ifelse(prob>0.5,1,0)
pred
```

```
[,1]
[1,] 0
[2,] 1
[3,] 1
[4,] 0
[5,] 0
```

onde agora 0 corresponde a -1 e 1 a +1. Ou seja, classificamos a segunda e terceira observações testes na classe +1 e a primeira, quarta e quinta na classe -1.

## RN com uma camada–Exemplo 4

- **Exemplo 14.4.** Vamos considerar novamente o conjunto Iris, mas somente a espécie Setosa e dois preditores, Petal.Length e Petal.Width. Para uma rede neuronal com três neurônios na camada escondida, obtemos a Figura 8.
- Os pesos obtidos estão indicados abaixo, e vemos que foram necessárias 41 iterações, o limiar foi 0,00994 e o erro de classificação 0,0125.

## RN com uma camada—Exemplo 4

|                                  |              |
|----------------------------------|--------------|
| error                            | 0.012533834  |
| reached.threshold                | 0.009935566  |
| steps                            | 41.000000000 |
| Intercept.to.1layhid1            | -5.266797930 |
| Petal.Length.to.1layhid1         | 1.029448892  |
| Petal.Width.to.1layhid1          | 4.035261475  |
| Intercept.to.1layhid2            | 4.783018629  |
| Petal.Length.to.1layhid2         | -1.313269314 |
| Petal.Width.to.1layhid2          | -2.377663815 |
| Intercept.to.1layhid3            | 3.230647942  |
| Petal.Length.to.1layhid3         | -0.630478727 |
| Petal.Width.to.1layhid3          | -2.790017234 |
| Intercept.to.Species == "setosa" | -1.171205492 |
| 1layhid1.to.Species == "setosa"  | -3.662576010 |
| 1layhid2.to.Species == "setosa"  | 3.664317725  |
| 1layhid3.to.Species == "setosa"  | 3.064883525  |

## RN com uma camada—Exemplo 4

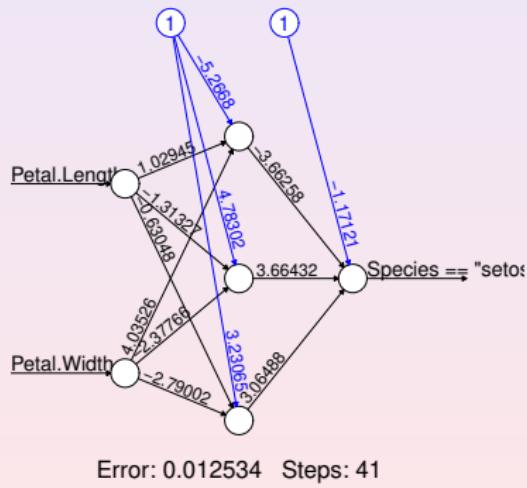


Figura 8. Rede neural com uma camada escondida para o Exemplo 4.

## O algoritmo backpropagation

- O ajuste de modelos de RN é feito minimizando uma função perda, usualmente a soma dos quadrados dos resíduos, no caso de regressão, onde a minimização é feita sobre os pesos. No caso de classificação, usamos a entropia. Nos dois casos é usado um algoritmo chamado de **backpropagation** (BP).
- O algoritmo BP calcula o gradiente da função perda com respeito aos pesos da rede, atualizando para trás uma camada de cada vez, a fim de minimizar a perda. Comumente usa-se **gradient descent** ou variações, como **stochastic gradient descent**.
- É necessário escolher valores iniciais e regularização (usando uma função penalizadora), porque o algoritmo de otimização é não convexo e instável.

# O algoritmo backpropagation

De modo geral, o problema de otimização da RN pode ser posto na forma:

$$\hat{\mathbf{w}} = \arg \min_{\mathbf{w}} \tilde{Q}_n(\mathbf{w}) = \arg \min_{\mathbf{w}} [\lambda_1 Q_n(\mathbf{w}) + \lambda_2 Q^*(\mathbf{w})], \quad (9)$$

com  $\lambda_1, \lambda_2 > 0$ , sendo

$$Q_n(\mathbf{w}) = \sum_{i=1}^n [y_i - f(\mathbf{x}_i, \mathbf{w})]^2, \quad (10)$$

e  $Q^*(\mathbf{w})$  um termo de regularização, que pode ser escolhido entre aqueles estudados no Capítulo 6 (lasso, ridge ou elastic net).

## O algoritmo backpropagation

- Podemos pensar uma rede neural como em (3), ou seja, uma função não linear paramétrica (determinística) de uma entrada  $\mathbf{x}$ , tendo  $\mathbf{z}$  como saída.
- Suponha que tenhamos os vetores de treinamento  $\mathbf{x}_i$ , e os vetores alvos (saídas)  $\mathbf{z}_i$ ,  $i = 1, \dots, n$  e considere a soma dos quadrados dos erros (10), ligeiramente modificada

$$Q_n(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^n \|z_i - f(\mathbf{x}_i, \mathbf{w})\|^2, \quad (11)$$

que queremos minimizar.

# O algoritmo backpropagation

[1] Vamos tratar primeiramente o problema de **regressão** e considerar a rede agora com um erro aleatório  $\varepsilon_i$  acrescentado antes da saída, de modo que agora temos um modelo probabilístico. Por simplicidade, consideremos a saída  $\mathbf{z} = (z_1, \dots, z_n)^\top$  e os erros com distribuição normal, com média zero e variância  $\sigma^2$ , de modo que

$$\mathbf{z} \sim N_n(f(\mathbf{x}_i, \mathbf{w}), \sigma^2 \mathbf{I}).$$

No caso em questão, a função de ativação de saída é a identidade. Chamando  $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n]$ , a verossimilhança pode ser escrita como

$$L(\mathbf{z}|\mathbf{X}, \mathbf{w}, \sigma^2) = \prod_{i=1}^n p(z_i|\mathbf{x}_i, \mathbf{w}, \sigma^2).$$

Maximizar a log-verossimilhança é equivalente a minimizar (10), de modo que obtemos  $\hat{\mathbf{w}}_{MV}$ . Encontrado  $\hat{\mathbf{w}}_{MV}$ , podemos obter o estimador de MV de  $\sigma^2$ . Como temos uma função não linear,  $Q_n(\mathbf{w})$  é não convexa, portanto na prática podemos obter máximos não locais da verossimilhança.

# O algoritmo backpropagation

(2) No caso de **classificação binária**, para a qual, por exemplo,  $z = +1$  indica a classe  $C_1$  e  $z = 0$  indica a classe  $C_2$ , considere a RN com saída única  $z$  com função de ativação logística,

$$z = \sigma(a) = \frac{1}{1 + e^{-a}},$$

de modo que  $0 \leq f(\mathbf{x}, \mathbf{w}) \leq 1$ . Podemos interpretar  $f(\mathbf{x}, \mathbf{w}) = P(C_1|\mathbf{x})$  e  $P(C_2|\mathbf{x}) = 1 - f(\mathbf{x}, \mathbf{w})$ . Segue que a distribuição de  $z$ , dado  $\mathbf{x}$ , é dada por

$$f(z|\mathbf{x}, \mathbf{w}) = f(\mathbf{x}, \mathbf{w})^z [1 - f(\mathbf{x}, \mathbf{w})]^{1-z}. \quad (12)$$

## O algoritmo backpropagation

- Considerando as observações de treinamento i.i.d., a função erro será dada pela entropia cruzada

$$Q_n(\mathbf{w}) = - \sum_{i=1}^n [z_i \log z_i - (1 - z_i) \log(1 - z_i)], \quad (13)$$

onde  $z_i = f(\mathbf{x}_i, \mathbf{w})$ .

- Temos que otimizar os pesos  $\mathbf{w}$ , ou seja, encontrar o valor que minimiza  $Q_n(\mathbf{w})$ . Usualmente, temos que obter o gradiente de  $Q_n$ , que vamos indicar por  $\nabla Q_n$ , que aponta para a maior taxa de crescimento de  $Q_n$ . Supondo-se que  $Q_n$  seja uma função contínua suave de  $\mathbf{w}$ , o valor mínimo ocorre no ponto onde o gradiente anula-se.

## O algoritmo backpropagation

- Procedimentos numéricos são usados e há uma literatura extensa sobre o assunto. As técnicas que usam o gradiente, começam fixando-se um valor inicial  $\mathbf{w}^{(0)}$  e os pesos são iterados na forma

$$\mathbf{w}^{(r+1)} = \mathbf{w}^{(r)} - \lambda \nabla Q_n(\mathbf{w}^{(r)}),$$

na qual  $\lambda$  é chamada a **taxa de aprendizado**.

- Esse método é chamado de **gradiente descendente** e usa todo o conjunto de treinamento. É um algoritmo não muito eficiente e, na prática, usa-se o algoritmo **backpropagation** (BP) para calcular o gradiente da SQE em uma RN.

# O algoritmo backpropagation

- Aqui vamos nos basear em (1)–(2) e em Hastie et al. (2017). Podemos usar também (4)–(8), veja Bishop (2006). Vamos chamar de  $\mathbf{w} = (\alpha_0, \dots, \alpha_M, \beta_0, \dots, \beta_K)^\top$  e considerar  $Q_n(\mathbf{w})$  escrita de modo geral como

$$Q_n(\mathbf{w}) = \sum_{k=1}^K \sum_{i=1}^n (z_{ik} - f(\mathbf{x}_i, \mathbf{w}))^2, \quad (14)$$

e seja

$$y_{mi} = h(\boldsymbol{\alpha}_m^\top \mathbf{x}_i), \quad \mathbf{y}_i = (y_{1i}, \dots, y_{Mi})^\top. \quad (15)$$

- Considerando os  $\mathbf{x}_i$  i.i.d, como em (14), escrevemos  $Q_n(\mathbf{w}) = \sum_{i=1}^n Q_i(\mathbf{w})$  e as derivadas

$$\frac{\partial Q_i}{\partial \beta_{km}} = -2(z_{ik} - f(\mathbf{x}_i, \mathbf{w}))g'(\boldsymbol{\beta}_k^\top \mathbf{y}_i)y_{mi}, \quad (16)$$

$$\frac{\partial Q_i}{\partial \alpha_{m\ell}} = -\sum_{k=1}^K 2(z_{ik} - f(\mathbf{x}_i, \mathbf{w}))g'(\boldsymbol{\beta}_k^\top \mathbf{y}_i)\beta_{km}h'(\boldsymbol{\alpha}_m^\top \mathbf{x}_i)x_{i\ell}. \quad (17)$$

## O algoritmo backpropagation

- O gradiente na  $(r + 1)$ -ésima iteração é dado por

$$\beta_{km}^{(r+1)} = \beta_{km}^{(r)} - \lambda_r \sum_{i=1}^n \frac{\partial Q_i}{\partial \beta_{km}^{(r)}}, \quad (18)$$

$$\alpha_{m\ell}^{(r+1)} = \alpha_{m\ell}^{(r)} - \lambda_r \sum_{i=1}^n \frac{\partial Q_i}{\partial \alpha_{m\ell}^{(r)}}, \quad (19)$$

sendo  $\lambda_r$  a taxa de aprendizagem.

- Escrevamos as derivadas (16) e (17) como

$$\frac{\partial Q_i}{\partial \beta_{km}} = \delta_{ki} y_{mi}, \quad (20)$$

$$\frac{\partial Q_i}{\partial \alpha_{m\ell}} = s_{mi} x_{i\ell}, \quad (21)$$

onde  $\delta_{ki}$  e  $s_{mi}$  são os erros do modelo atual na saída e camadas escondidas, respectivamente.

# O algoritmo backpropagation

- De suas definições, esses erros satisfazem

$$s_{mi} = h(\boldsymbol{\alpha}_m^\top \mathbf{x}_i) \sum_{k=1}^K \beta_{km} \delta_{ki}, \quad (22)$$

conhecidas como **equações de backpropagation**, que podem ser usadas para implementar (18) e (19) em duas passadas:

- (a) **uma para a frente**, onde os pesos atuais são fixos e os valores previstos de  $f(\mathbf{x}, \mathbf{w})$  são calculados a partir de (22);
- (b) **uma para trás**, os erros  $\delta_{ki}$  são calculados e propagados para trás via (20) e (21) para obter os erros  $s_{mi}$ .
- Os dois conjuntos de erros são então usados para calcular os gradientes para atualizar (18) e (19) via (20) e (21). Uma **época de treinamento** refere-se a uma passada por todo o conjunto de treinamento.

## Referências

- Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. New York: Springer.
- Hastie, T., Tibshirani, R. and Friedman, J. (2017). *The Elements of Statistical Learning*, 2nd Edition, Springer.
- Hebb, D. O. (1949). *The organization of behavior*. New York: Wiley.
- McCulloch, W. S. and Pitts, W. A. (1943). Logical calculus of the ideas immanent in nervous activity. *Butt. Math. Biophysics*, S, 115–133.
- Pereira, B.B., Rao, C.R. and Oliveira, F. B. (2020). *Statistical Learning Using Neural Networks: A Guide for Statisticians and Data Scientists with Python*. Chapman and Hall.

## Referências

- Rosenblatt, F. (1958). The perceptron: A theory of statistical separability in cognitive systems. Buffalo: Cornell Aeronautical Laboratory, Inc. Report No. VG-1196-G-1.
- Rumelhart, D. E., Hinton, G. E. and Williams, R. J. (1986a). Learning representations by back-propagating errors. *Nature*, **323**, 533–536.
- Rumelhart, D. E., Hinton, G. E. and Williams, R. J. (1986b). Learning internal representations by error propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Vol. 1: Foundations (eds Rumelhart, D. E. and McClelland, J. L.) 318–362 (MIT, Cambridge, 1986).

# MAE 5905: Introdução à Ciência de Dados

Pedro A. Morettin

Instituto de Matemática e Estatística  
Universidade de São Paulo  
pam@ime.usp.br  
<http://www.ime.usp.br/~pam>

## Aula 16 - Parte 2

15 de junho de 2023

# Sumário

1 Deep Learning

2 Redes Neurais Recorrentes

3 Redes Neurais Long-Short-Term Memory - LSTM

# Deep learning

- Como vimos anteriormente, ML (aprendizado DE máquina) envolve procedimentos segundo os quais um sistema computacional adquire a habilidade de aprender extraindo padrões de dados. Ou seja, tanto na programação clássica, como em ML, nada é conseguido sem dados, que podem tomar a forma de números, vetores, curvas, imagens, palavras etc.
- Vimos que o desempenho de um algoritmo de ML depende da representação desses dados, por meio de funções pré-determinadas, como a logística, tangente hiperbólica etc. A escolha dessa representação é crucial.
- No caso de termos várias camadas intermediárias obtém-se o que é chamado aprendizado profundo (**deep learning** - DL). Nesse caso, o sistema computacional (SC) aprende a partir dos dados (e exemplos) em termos de uma hierarquia de conceitos, cada um definido por meio de sua relação com conceitos mais simples, ou ainda, o SC aprende conceitos complicados a partir de conceitos simples.

# Deep learning

- Um exemplo de modelo DL é o multilayer perceptron, ou seja, uma aplicação que leva os dados de entrada a valores de saída, por meio de composição de funções mais simples.
- Segundo Goodfellow et al. (2016), modelos DL remontam à década de 1940, com os nomes de Cibernética (1940-1960), coneccionismo acoplado a redes neurais (1980-1990) e ressurge como DL em 2006.
- Por sua vez, os tamanhos de conjuntos de dados expandiram-se de small data, da ordem de centenas ou milhares na década de 1900–1980, a dezenas ou centenas de milhares de dados após 1990 (big data). Um exemplo bastante estudado na área é o MNIST (Modified National Institute of Standards and Technology), que consiste de fotos de dígitos escritos a mão.

# Deep learning

- A partir de 2000, conjuntos de dados ainda maiores surgiram, contendo dezenas de milhões de dados, notadamente dados obtidos na Internet. Para analisar esses megadados foi necessário o desenvolvimento de CPU's mais rápidas e as chamadas GPU's (**Graphics Processing Units**). Essas são usadas em celulares, computadores pessoais, estações de trabalho e consoles de jogos e são muito eficientes em computação gráfica e processamento de imagens. Sua estrutura altamente paralela as torna eficientes para algoritmos que processam grandes blocos de dados em paralelo.
- A complexidade de um algoritmo é proporcional ao número de observações, número de preditores, número de camadas e número de épocas de treinamento. Para detalhes sobre esses tópicos, veja Hastie et al. (2017) e Chollet (2018).

# Deep learning

Como dissemos acima, DL tornou-se proeminente nas décadas de 2006–2010. Conforme Chollet (2018), as seguintes aplicações tornaram-se possíveis:

- classificação de imagens ao nível quase humano (NQH);
- reconhecimento de falas ao NQH;
- transcrições de escritas a mão ao NQH;
- aperfeiçoamento de técnicas de ML;
- aperfeiçoamento de conversões textos-para-falas;
- carros autônomos ao NQH;
- aperfeiçoamento de buscas na Internet;
- assistentes digitais, como o Google Now e Amazon Alexa;
- habilidade para responder questões sobre linguagens naturais.

# Deep learning

- Todavia, muitas aplicações levarão muito tempo para serem factíveis e questões relativas ao desenvolvimento, por SC, da inteligência ao nível humano, não devem ser seriamente consideradas no presente estágio de desenvolvimento. Em particular, muitas afirmações feitas nas décadas de 1960–1970 e 1980–1985 sobre **expert systems** e AI em geral, que não se concretizaram, levaram a um decréscimo de investimento em pesquisa na área.
- Em termos de software, há diversas bibliotecas para ML e DL, como **Theano**, **Torch**, **MXNet** e **TensorFlow**.

# Deep learning

- Para analisar redes neurais com várias camadas é possível usar o pacote **keras** do R, que por sua vez usa o pacote **Tensorflow** com capacidades CPU e GPU. Modelos *deep learning* que podem ser analisados incluem **redes neurais recorrentes** (*recurrent neural networks* - RNN), **redes Long Short-Term Memory** (LSTM), **Convolutional Neural Network** (CNN)etc.
- As redes LSTM são apropriadas para captar dependências de longo prazo e implementar previsão de séries temporais.
- As redes CNN representam o estado da arte atualmente, usadas para classificação de imagens, linguagens naturais e outras aplicações.
- Para uma excelente revisão sobre aprendizado profundo veja o artigo de LeCun et al. (2015) e as referências nele contidas, sobre os principais desenvolvimentos na área até 2015. A seguir daremos breves introduções sobre RNN e CNN, com alguns exemplos.

## Redes neurais recorrentes - RNN

- O termo RNN é usado indiscriminadamente para se referir a classes amplas de redes com uma estrutura geral similar. Essas redes exibem um comportamento temporal dinâmico.
- RNN foram baseadas nos trabalhos de David Rumelhart em 1986 e John Hopfield em 1982. LSTM foram inventadas por Hochreiter e Schmidhuber em 1997 e estabeleceram recordes de acurácia em diversas aplicações.
- RNN tem sido um foco importante de pesquisa desde a década de 1990. Elas são planejadas para aprender padrões sequenciais ou variando no tempo.
- RNN têm sido aplicada a uma variedade grande de problemas. No final da década de 1980 foram introduzidas RNN parciais por vários pesquisadores, como Rumelhart, Hinton e Williams (veja Rumelhart, 1986) para aprender sequências de caracteres. Outras aplicações incluem problemas em sistemas dinâmicos com sequências temporais de eventos, estimação da potência de turbinas, previsões financeiras, sínteses musicais, previsões de cargas elétricas, previsão de vazões de rios etc.

## Redes neurais recorrentes - RNN

- A arquitetura de RNN pode ser:
  - (a) **completamente interconectada**: todos os estados são alcançados por qualquer estado, inclusive cada estado de si próprio;
  - (b) **parcialmente conectada**: nem todos os estados são alcançados da partir de um estado qualquer. Essas redes são usadas para aprender sequência de caracteres.
- Pesquisadores desenvolveram uma variedade de esquemas pelos quais métodos gradiente, e em especial, aprendizado por backpropagation, podem ser estendidos a RNN. Werbos introduziu um procedimento que aproxima a evolução de uma RNN como uma sequência de redes estáticas usando métodos gradiente (*backpropagation through time*). Outras abordagens podem ser encontradas em Lapedes e Farber(1986), Pineda (1987), Almeida (1987), Sato (1990). As várias tentativas para estender backpropagation para RNN estão sumarizadas em Pearlmutter (1995).

## Redes neurais recorrentes - RNN

- Do ponto de vista de séries temporais, RNN podem ser vistas como modelos não lineares de espaços de estados. Veja Morettin e Toloi (2020) para detalhes sobre esses modelos.
- Uma RNN processa um elemento por vez de uma sequência de entrada  $X_t$ , mantendo sua unidades ocultas num **vetor de estados**  $S_t$ , que contém informação sobre a história passada da sequência (LeCun et al. 2015). Denotando por  $Y_t$  a sequência processada no estado  $t$ , uma rede neural recorrente processa sequências de entradas iterativamente,

$$Y_t = f(Y_{t-1}, X_t),$$

e as previsões são feitas segundo

$$\hat{Y}_{t+h|t} = g(h_t),$$

com  $f$  e  $g$  denotando funções a serem definidas.

- Todas as redes neurais recorrentes têm a forma de uma rede neural que se repete como na Figura 1: A, B e W são parâmetros (matrizes) invariantes no tempo. O algoritmo BP pode ser aplicado à rede desdobrada.

## Redes neurais recorrentes - RNN

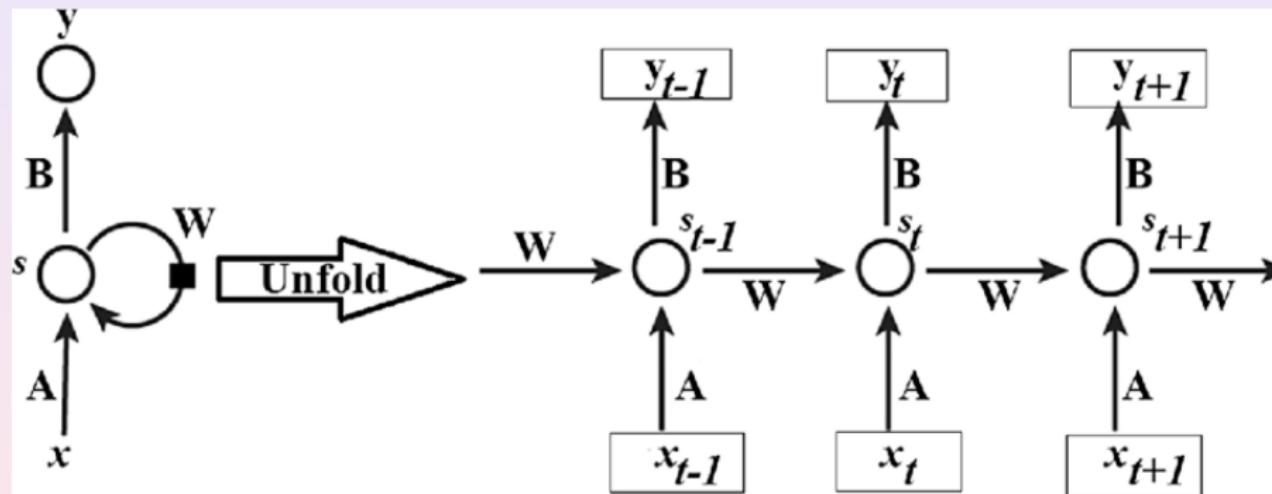


Figura 1: RNN e seu desdobramento no tempo.

## Redes neurais recorrentes - RNN

- Redes neurais recorrentes são difíceis de implementar pois sofrem do chamado **problema do gradiente evanescente** (*vanishing gradient problem*).
- Uma solução foi proposta por Hochreiter e Schmidhuber (1997) por meio de uma variante das redes neurais recorrentes, chamada **rede com memórias de curto e longo prazos** (*Long-Short-Term Memory - LSTM*), capazes de “aprender” dependências de longo prazo.
- No caso de séries temporais, há modelos para descrever processos com memória curta (*short memory*), como os modelos *ARIMA* (autorregressivos, integrados e de média móveis), e modelos para descrever memória longa (*long memory*), como os modelos *ARFIMA* (autorregressivos, integrados fracionários e de médias móveis). Veja, por exemplo, Morettin e Toloi (2018).
- Uma rede *LSTM* modela simultaneamente as memórias de curto e longo prazos. Discutimos brevemente essas redes na seção seguinte.

# LSTM

- Cerca de 2007, LSTMs revolucionaram o reconhecimento de fala, superando modelos tradicionais em aplicações nessa área. Em 2009, uma rede **Connectionist Temporal Classification (CTC)-trained LSTM** foi a primeira RNN a vencer competições em reconhecimento de padrões.
- LSTMs também melhoraram o reconhecimento de grandes vocabulários de falas e sínteses de textos para falas e foi usada no Google Android. Em 2015, o reconhecimento de falas do Google obteve um salto de desempenho da ordem de 49% por meio de uma rede **CTC-trained LSTM**.
- Redes LSTM quebraram recordes para melhorar traduções de textos, modelagem de linguagens e processamento de múltiplas línguas. LSTM combinada com **convolutional neural networks (CNNs)** melhoraram a captação automática de imagens.

# LSTM

- Uma rede LSTM consiste de blocos de memória, chamadas **células** (*cells*), conectadas por meio de camadas (*layers*).
- A informação nas células está contida no estado  $C_t$  e no estado escondido  $h_t$  e é regulada por mecanismos chamados **portas** (*gates*), por meio de funções de ativação *sigmoid* e *tanh*.
- A função sigmoide tem como saídas números entre 0 e 1, com 0 indicando *nada passa* e 1 indicando *tudo passa*. A rede LSTM tem, portanto, a habilidade de adicionar ou desprezar informação do estado da célula (condicionalmente).
- Em geral, as portas têm como entradas os estados escondidos do instante anterior,  $h_{t-1}$  e da entrada atual  $x_t$  e as multiplica por pesos matriciais,  $\mathbf{W}$ , e um viés (*bias*)  $b$  é adicionado ao produto.

# LSTM

Há três portas principais:

- (i) **Forget gate**: esta porta determina qual informação será desprezada do estado da célula. A saída será 0, significando **apagar** e 1, implicando **lembrar todos**; formalmente,

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f).$$

- (ii) **Input gate**: nesse passo, a função de ativação *tanh* criará um vetor de candidatos potenciais como segue:

$$\hat{C}_t = \tanh(W_c[h_{t-1}, x_t] + b_c).$$

A camada sigmoide cria um filtro atual como segue:

$$U_t = \sigma(W_u[h_{t-1}, x_t] + b_u).$$

A seguir o estado anterior  $C_{t-1}$  é atualizado como

$$C_t = f_t C_{t-1} + U_t \hat{C}_t.$$

# LSTM

- (iii) **Output gate:** nesse passo, a camada sigmoide filtra o estado que vai para a saída:

$$O_t = \sigma(W_0[h_{t-1}, x_t] + b_0).$$

- (iv) O estado  $C_t$  é então passado por meio da função  $\tanh$  para escalar os valores para o intervalo  $[-1, 1]$ . Finalmente, o estado escalonado é multiplicado pela saída filtrada para se obter o estado escondido  $h_t$  a ser passado para a próxima célula:  $h_t = O_t \times \tanh(C_t)$ .

A arquitetura de uma rede LSTM está mostrada na Figura 2.

# LSTM

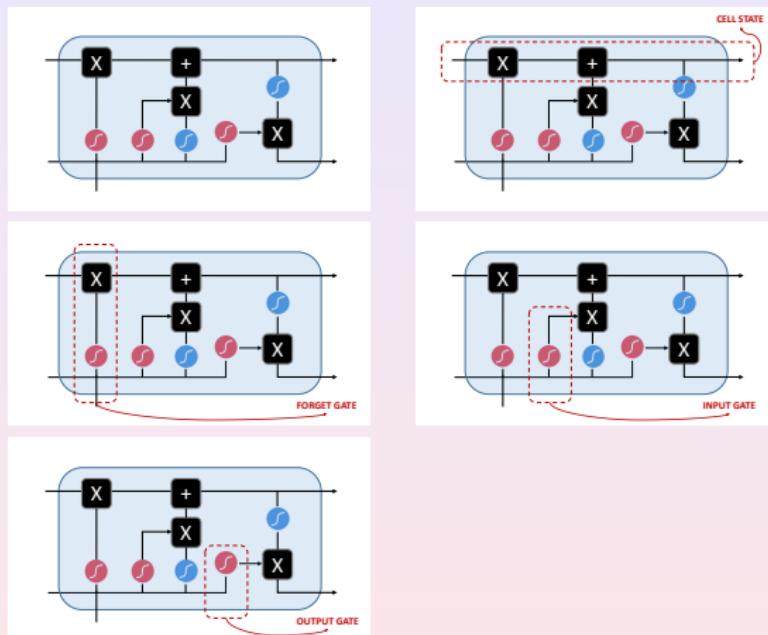


Figura 2: Arquitetura da célula de uma rede LSTM.

## LSTM - Exemplo 1

- Consideremos os preços diárias das ações da Petrobrás entre 31/08/1998 e 29/09/2010, totalizando  $n = 2990$  observações, constantes do arquivo acoes.
- O gráfico da série está indicado na Figura 3, mostrando o seu caráter não estacionário.
- Como a rede LSTM funciona melhor para séries estacionárias ou mais próximas possíveis da estacionariedade, tomemos a série de primeiras diferenças, definida como  $Y_t = X_t - X_{t-1}$ , com  $X_t$  denotando a série original. A série de primeiras diferenças também está apresentada na Figura 3.
- Depois de obtidas as previsões deve-se fazer a transformação inversa para obter as previsões com a série original.
- Nosso intuito é fazer previsões para os dados de um conjunto de validação, com 897 observações, ajustando uma rede LSTM aos dados de um conjunto de treinamento, consistindo de 2093 observações.

## LSTM - Exemplo 1

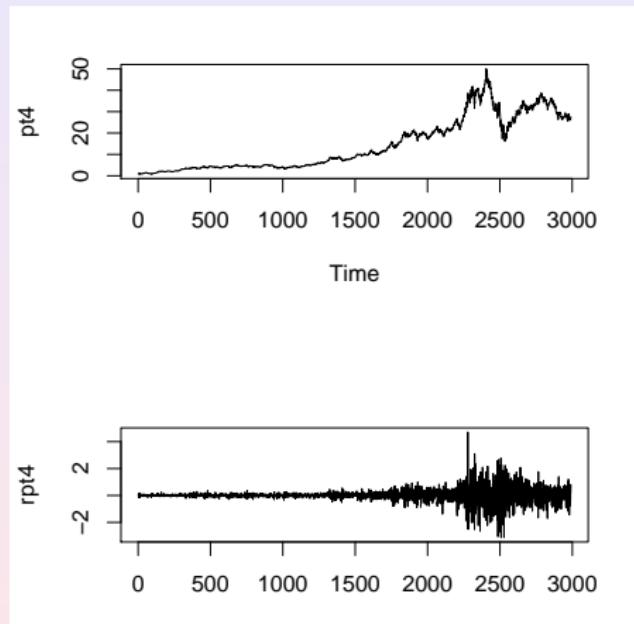


Figura 3: Série de preços das ações da Petrobrás e respectivas diferenças.

## LSTM - Exemplo 1

- Como as redes LSTM supõem dados na forma de aprendizado supervisionado, ou seja, com uma variável resposta  $Y$  e uma variável preditora  $X$ , tomamos valores defasados da série de forma que os valores obtidos até o instante  $t - k$  serão considerados como variáveis preditoras e o valor no instante  $t$  será a variável resposta. Neste exemplo, tomamos  $k=1$ .
- Para esse efeito, consideramos os comandos

```
lag_transform <- function(x, k=1){
 lagged = c(rep(NA,k),x[1:(length(x)-k)])
 DF = as.data.frame(cbind(lagged,x))
 colnames(DF) <- c(paste0('x-',k),'x')
 DF[is.na(DF)] <- 0
 return(DF)
}

supervised = lag_transform(Petrobras4, 1)
head(supervised)
```

## LSTM - Exemplo 1

- obtendo, para as 6 primeiras observações, a variável preditora na coluna  $x-1$  e a variável resposta, na coluna  $x$ .

|   | $x-1$ | $x$   |
|---|-------|-------|
| 1 | 0.00  | 0.08  |
| 2 | 0.08  | 0.07  |
| 3 | 0.07  | -0.06 |
| 4 | -0.06 | -0.04 |
| 5 | -0.04 | -0.04 |
| 6 | -0.04 | -0.03 |

- Para a divisão entre dados treinamento (70%) e de validação (30%), podemos utilizar os comandos

```
N <- nrow(supervised)
n <- round(N*0.7,digits=0)
train <- supervised[1:n,]
test <- supervised[(n+1):N,]
```

## LSTM - Exemplo 1

- Em seguida, normalizamos os dados de entrada para que pertençam ao intervalo de variação da função de ativação, que é a sigmoide, com variação em [-1,1].
- Os valores mínimo e máximo do conjunto de treinamento são usados para normalizar os conjuntos de treinamento e de validação além dos valores preditos, por meio dos comandos

```
scale_data <- function(train, test, feature_range=c(0,1)){
 x<-train
 fr_min<-feature_range[1]
 fr_max<-feature_range[2]
 std_train<-((x-min(x))/(max(x)-min(x)))
 std_test<-(test-min(x))/(max(x)-min(x))
 scaled_train<-std_train*(fr_max-fr_min)+fr_min
 scaled_test<-std_test*(fr_max-fr_min)+fr_min
 return(list(scaled_train=as.vector(scaled_train),
 scaled_test=as.vector(scaled_test),
 scaler=c(min=min(x),max=max(x))))
}
```

## LSTM - Exemplo 1

- continuando:

```
Scaled <- scale_data(train,test, c(-1,1))
y_train <- Scaled$scaled_train[,2]
x_train <- Scaled$scaled_train[,1]
y_test <- Scaled$scaled_test[,2]
x_test <- Scaled$scaled_test[,1]
```

- Para reverter os valores previstos à escala original, consideramos

```
invert_scaling <- function(scaled, scaler, feature_range=c(0,1)){
 min=scaler[1]
 max=scaler[2]
 t=length(scaled)
 mins=feature_range[1]
 maxs=feature_range[2]
 inverted_dfs=numeric(t)
 for(i in 1:t){
 X=(scaled[i]-mins)/(maxs-mins)
 rawValues=X*(max-min)+min
 inverted_dfs[i]<-rawValues
 }
 return(inverted_dfs)
```

## LSTM - Exemplo 1

- A partir deste ponto iniciamos a modelagem. Com essa finalidade, precisamos fornecer o lote de entrada na forma de um vetor tridimensional, [samples, timesteps, features] a partir do estado atual [samples, features], em que samples é o número de observações em cada lote (tamanho do lote), timesteps é o número de passos para uma dada observação (para este exemplo, timesteps=1) e features=1, para o caso univariado como no exemplo.
- O tamanho do lote deve ser função dos tamanhos das amostras de treinamento e de validação. Usualmente esse valor é 1. Também devemos especificar stateful = TRUE de modo que após processar um lote de amostras os estados internos sejam reutilizados para as amostras do lote seguinte. Os comandos correspondentes são

```
dim(x_train) <- c(length(x_train), 1, 1)
X_shape2 <- dim(x_train)[2]
X_shape3 <- dim(x_train)[3]
batch_size=1
units=1
model <- keras_model_sequential()
model %>% layer_lstm(units,
 batch_input_shape = c(batch_size, X_shape2, X_shape3),
 stateful=TRUE) %>% layer_dense(units=1)
```

## LSTM - Exemplo 1

- O modelo pode, então ser compilado, com a especificação do erro quadrático médio como função perda.
- O algoritmo de otimização é o *Adaptive Monument Estimation (ADAM)*. Usamos a acurácia como métrica para avaliar o desempenho do modelo.

```
model %>% compile(
 loss='mean_squared_error', optimizer=optimizer_adam(lr=0.02, decay=1e-6),
 metrics=c('accuracy'))
```

Por meio do comando `summary(model)` obtemos:

Model: "sequential"

| Layer (type)            | Output Shape | Param # |
|-------------------------|--------------|---------|
| lstm (LSTM)             | (1, 1)       | 12      |
| dense (Dense)           | (1, 1)       | 2       |
| Total params: 14        |              |         |
| Trainable params: 14    |              |         |
| Non-trainable params: 0 |              |         |

## LSTM - Exemplo 1

- Para ajustar o modelo, a rede LSTM exige o comando `shuffle=FALSE` cuja finalidade é evitar o embaralhamento do conjunto de treinamento e manter a dependência entre  $x_i$  e  $x_{i+t}$ .
- Além disso o algoritmo requer a redefinição do estado da rede após cada iteração (época).
- Os comandos para o ajuste do modelo são

```
Epochs=50
for(i in 1:Epochs){
 model %>% fit(x_train, y_train, epochs=1, batch_size=batch_size,
 verbose=1,shuffle=FALSE)
 model %>% reset_states()}
```

são geradas 50 épocas (iterações) da rede neural e em cada época é possível visualizar o valor da função perda e a acurácia.

# LSTM - Exemplo 1

- Para fazer previsões usamos a função predict() e em seguida invertemos a escala e as diferenças para retornar à série original.

```
L=length(x_test)
scaler=Scaled$scaler
predictions=numeric(L)
for(i in 1:L){
 X=x_test[i]
 dim(X)=c(1,1,1)
 yhat=model %>% predict(X, batch_size=batch_size)
 yhat=invert_scaling(yhat, scaler, c(-1,1))
 yhat
 yhat=yhat+Petrobras4[(n+i)]
 predictions[i] <- yhat}
```

- As previsões para as 897 observações do conjunto teste podem ser obtidas por meio da função prediction(). As 6 primeiras são apresentadas abaixo:

```
[1] -0.28833461 0.35166539 -0.16833461 -0.26833461 0.01166539
-0.03833461
```

## LSTM - Exemplo 1

- As observações do conjunto de validação e as correspondentes previsões estão representadas na Figura 4.
- Na Figura 5 colocamos a série para o conjunto de treinamento (em azul), a série no conjunto de validação (em vermelho) e a série de previsões (em verde).
- A raiz quadrada do erro quadrático médio, calculado da maneira habitual, é  $RMSE = 0,02835$ .

## LSTM - Exemplo 1

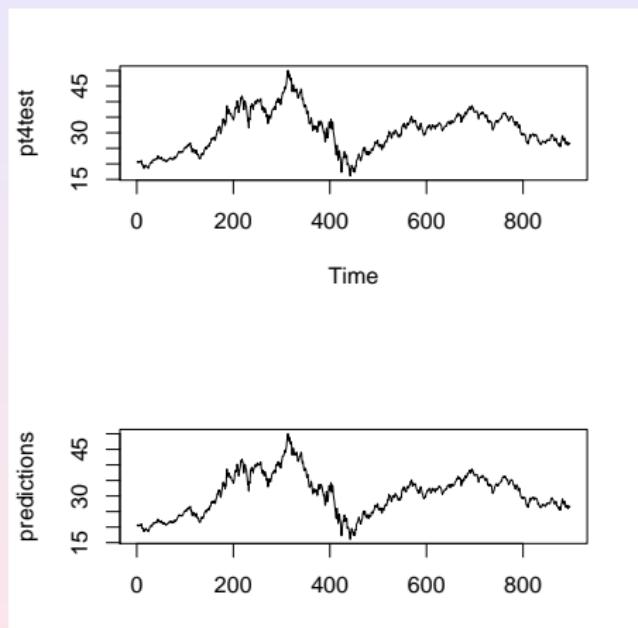


Figura 4: Série original e série de previsões de preços das ações da Petrobrás, no conjunto de validação.

## LSTM - Exemplo 1

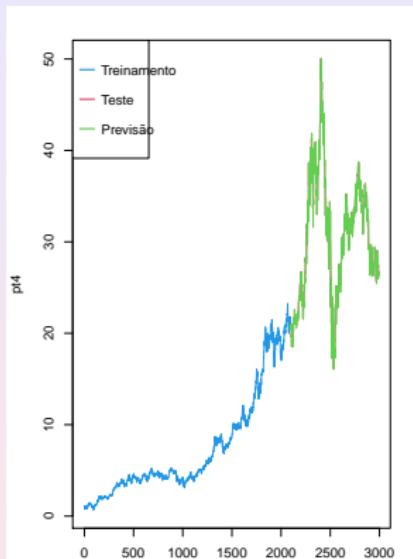


Figura 5: Série original de preços das ações da Petrobrás para o conjunto de treinamento (azul), para o conjunto de validação (vermelho) e série de previsões (verde).

## LSTM - Exemplo 2

- Consideraremos os dados do arquivo covid2 com 526 observações de casos e óbitos por COVID-19 no Brasil entre 19/03/2020 e 25/08/2021.
- O gráfico da série está na Figura 6.
- Separemos  $n = 369$  observações para o conjunto de treinamento e  $m = 157$  observações para o conjunto de validação. O objetivo é prever o número de casos no conjunto de validação.
- Após uma análise por meio de uma rede LSTM similar àquela do Exemplo 1, o comando `summary()` produz o seguinte resultado:

## LSTM - Exemplo 2

```
summary(model)
Model: "sequential"

Layer (type) Output Shape Param #
=====
lstm (LSTM) (1, 1) 12

dense (Dense) (1, 1) 2

Total params: 14
Trainable params: 14
Non-trainable params: 0

```

## LSTM - Exemplo 2

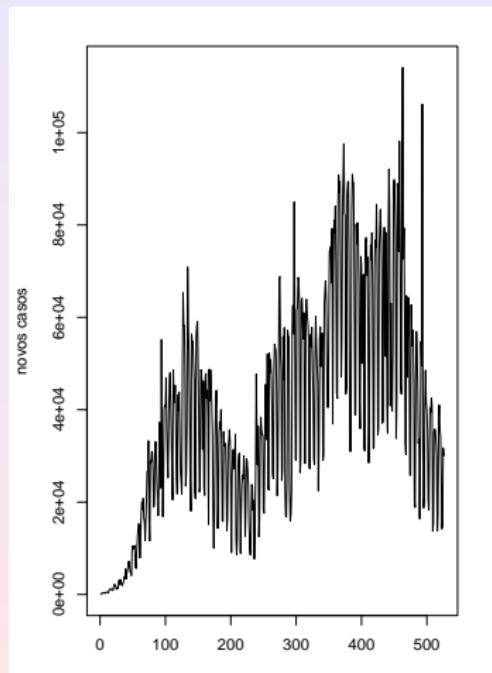


Figura 6: Série diária de novos infectados pelo vírus Covid-19 no Brasil.

## LSTM - Exemplo 2

A raiz quadrada do erro quadrático médio é  $RMSE = 20793,87$  e as previsões podem ser vistas na Figura 7.

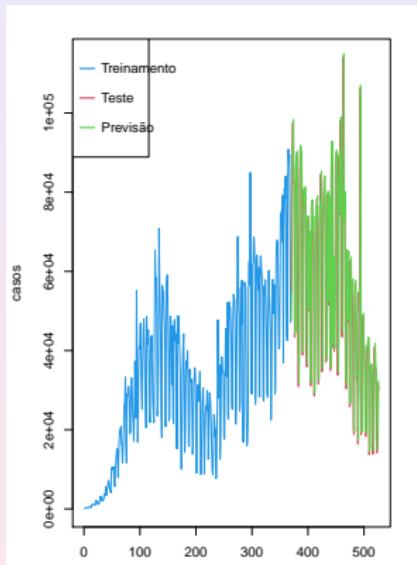


Figura 7: Série original de novos infectados por Covid-19 no Brasil para o conjunto de treinamento (azul), para o conjunto de validação (vermelho) e série de previsões (verde).

## Referências

- Chollet, F. (2018). *Deep Learning with R*. Manning.
- Goodfellow, I., Bengio, Y. and Courville, A. (2016). *Deep Learning*. The MIT Press.
- Hastie, T., Tibshirani, R. and Friedman, J. (2017). *The Elements of Statistical Learning*, 2nd ed. New York: Springer.
- LeCun, Y., Bengio, Y. and Hinton, G. (2015). Deep learning. *Nature*, **521**, 436–444.
- Morettin, P. A. e Singer, J. M. (2023). *Estatística e Ciência de Dados*. Segunda edição. LTC: Rio de Janeiro.

# MAE 5905: Introdução à Ciência de Dados

Pedro A. Morettin

Instituto de Matemática e Estatística  
Universidade de São Paulo  
pam@ime.usp.br  
<http://www.ime.usp.br/~pam>

## Aula 16 - Parte 3

20 de junho de 2023

# Sumário

1 Redes neurais convolucionais - CNN

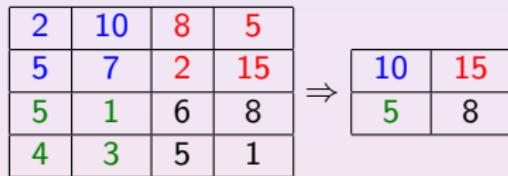
2 Redes generativas adversárias - GAN

# CNN

- As redes neurais convolucionais (*convolutional neural networks*, CNN) foram introduzidas por LeCun et al. (1998) e são muito eficientes para resolver problemas de classificação de imagens.
- De modo geral, as CNN são desenhadas para processar dados que surgem na forma de matrizes (*arrays*), como sequências (séries temporais e linguagem) unidimensionais, sinais de audio bidimensionais e audio e imagens tridimensionais. Como exemplo, podemos ter uma imagem colorida composta de três *arrays* bidimensionais, contendo intensidades de pixel nos três canais de cores (RGB) (LeCun et al., 2015).
- Nas CNN, para extrair padrões (*features*) dos dados, há quatro ideias básicas:
  - uso de muitas camadas;
  - camadas de convolução (*convolution layers*, CL);
  - camadas de agrupamento (*pooling layers*, PL);
  - camadas totalmente conectadas.

# CNN

- A CL é responsável por extrair os padrões locais (*feature maps*) da camada precedente. Esta é feita por meio de pesos (*filter banks*) de tamanhos reduzidos. Diferentes bancos de filtros são usados para os diversos padrões da imagem, como arestas, arranjos de arestas, partes de objetos familiares, cores etc. O resultado dessa soma ponderada local é passada por meio de uma não linearidade (ReLU).
- A PL, usada após uma camada convolucional, destina-se a reduzir a dimensão dos dados de entrada e juntar características locais numa só. Pode-se usar médias ou escolher o maior valor encontrado em subregiões. Este segundo procedimento é o mais utilizado e chamado *maxpooling*. Na Figura 1 temos um exemplo com uma imagem  $4 \times 4$  e um maxpooling com filtro  $2 \times 2$ . Essa técnica reduz a quantidade de dados para a camada seguinte, reduzindo também o custo de processamento e memória.



**Figura 1:** Exemplo de *maxpooling* no caso de uma imagem  $4 \times 4$ .

# CNN

- As camadas totalmente conectadas situam-se no final da rede e os padrões extraídos nas camadas de convolução anteriores são utilizadas para a classificação final da rede neural.
- As razões para essa arquitetura são (LeCun et al, 2015):
  - (i) em *arrays*, como imagens, grupos locais de valores são correlacionados, formando padrões (*motifs*) facilmente detectados;
  - (ii) as estatísticas locais dessas *arrays* são invariantes em relação à localização, donde a ideia de que os mesmos pesos são compartilhados por unidades em diferentes localizações.
- Os cálculos com CNN envolvem contrações em escalas múltiplas, linearização de simetrias hierárquicas e separação esparsa. Em muitas aplicações o número de amostras cresce linearmente com o número de dimensões.
- Como todo algoritmo de aprendizagem, uma CNN é baseada em alguma suposição de suavidade (regularidade) do classificador, digamos,  $f(\mathbf{x})$ , sendo  $\mathbf{x}$  o vetor de dados, e a natureza dessa regularidade é o problema matemático mais importante.
- A ideia é reduzir a dimensão de  $\mathbf{x}$  e isso pode ser feito definindo-se uma nova variável  $\phi(\mathbf{x})$ , em que  $\phi$  é um operador contração, que reduz a variabilidade de  $\mathbf{x}$ , aliada à separação de valores distintos de  $f(\mathbf{x})$ . Os aspectos matemáticos de uma CNN estão descritas em Mallat (2026) e Kohler et al. (2022).

# CNN - Exemplo 1

- Vejamos um exemplo de convolução com uma série temporal fictícia com  $n = 10$  observações como entrada:

|     |     |      |     |     |      |      |     |     |     |
|-----|-----|------|-----|-----|------|------|-----|-----|-----|
| 1,2 | 0,9 | -0,8 | 0,7 | 1,5 | -1,3 | -1,0 | 0,7 | 1,3 | 1,4 |
|-----|-----|------|-----|-----|------|------|-----|-----|-----|

Consideremos um filtro com coeficientes:

|   |   |   |
|---|---|---|
| 1 | 2 | 1 |
|---|---|---|

- A convolução dos três primeiros valores da série com os pesos do filtro resulta  $(1,2) \times 1 + (0,9) \times 2 + (-0,8) \times 1 = 2,2$ . Deslocando-se uma unidade de tempo e efetuando o produto dos valores seguintes pelos coeficientes do filtro obtemos o valor 0. Continuando, obtemos a série de saída

|     |   |     |     |      |      |     |     |
|-----|---|-----|-----|------|------|-----|-----|
| 2,2 | 0 | 2,1 | 2,4 | -2,1 | -2,6 | 1,7 | 4,7 |
|-----|---|-----|-----|------|------|-----|-----|

# CNN - Exemplo 1

- Para que tenhamos convolução e *maxpooling*, temos que adicionar dois zeros no começo e final da série (*padding*):

|   |   |     |     |      |     |     |      |      |     |     |     |   |   |
|---|---|-----|-----|------|-----|-----|------|------|-----|-----|-----|---|---|
| 0 | 0 | 1,2 | 0,9 | -0,8 | 0,7 | 1,5 | -1,3 | -1,0 | 0,7 | 1,3 | 1,4 | 0 | 0 |
|---|---|-----|-----|------|-----|-----|------|------|-----|-----|-----|---|---|

- A série convolvida e a saída após tomar o máximo de cada três observações estão mostradas a seguir:

|     |     |     |   |     |     |      |      |     |     |     |     |
|-----|-----|-----|---|-----|-----|------|------|-----|-----|-----|-----|
| 1,2 | 3,3 | 2,2 | 0 | 2,1 | 2,4 | -2,1 | -2,6 | 1,7 | 4,7 | 4,1 | 1,4 |
|-----|-----|-----|---|-----|-----|------|------|-----|-----|-----|-----|

|     |     |     |     |
|-----|-----|-----|-----|
| 3,3 | 2,4 | 1,7 | 4,7 |
|-----|-----|-----|-----|

- A Figura 2 ilustra uma CNN com série temporal como entrada. Se a entrada for outra *array*, como uma imagem, o esquema é o mesmo, obtendo-se não mais sequências unidimensionais, mas matrizes, como na Figura 1.

## CNN - Exemplo 1

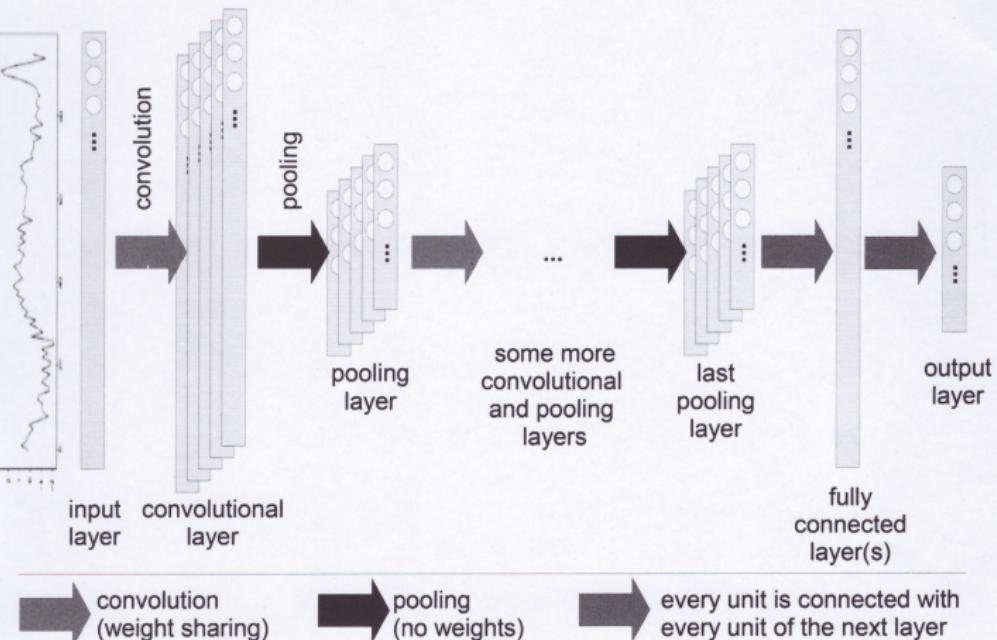


Figure 2: Rede neural convolucional.

## CNN - Exemplo 2

- Vamos usar os dados de fotos de dígitos escritos a mão, **Mnist** (Modified NIST(National Institute of Standards and Technology)).
- Esses dados contém 60.000 imagens de treinamento e 10.000 imagens de teste, dos quais a metade de cada conjunto foi retirada conjunto de treinamento do NIST, as outras duas metades foram retiradas do conjunto teste do NIST.
- Esses dados foram usados com diversos tipos de classificadores, dentre os quais classificadores lineares, KNN, SVM, florestas aleatórias e diversos tipos de redes neurais, incluindo as CNN. Usaremos um código constante do site

[https://rpubs.com/juanhklopper/example\\_of\\_a\\_CNN](https://rpubs.com/juanhklopper/example_of_a_CNN)

# CNN - Exemplo 2

Comentaremos alguns comandos:

- A primeira imagem é um 5:

```
> y_train[1,] # a primeira imagem \'e um 5
```

```
[1] 0 0 0 0 0 1 0 0 0 0
```

- Criando o modelo:

```
model <- keras_model_sequential() %>%
 layer_conv_2d(filters = 16,
 kernel_size = c(3,3),
 activation = 'relu',
 input_shape = input_shape) %>%
 layer_max_pooling_2d(pool_size = c(2, 2)) %>%
 layer_dropout(rate = 0.25) %>%
 layer_flatten() %>%
 layer_dense(units = 10,
 activation = 'relu') %>%
 layer_dropout(rate = 0.5) %>%
 layer_dense(units = num_classes,
 activation = 'softmax')
```

## CNN - Exemplo 2

- O comando `summary()` mostra que foram aprendidos 27.320 parâmetros.

```
> model %>% summary(),
```

| Layer (type)                 | Output Shape       | Param # |
|------------------------------|--------------------|---------|
| conv2d (Conv2D)              | (None, 26, 26, 16) | 160     |
| max_pooling2d (MaxPooling2D) | (None, 13, 13, 16) | 0       |
| dropout_1 (Dropout)          | (None, 13, 13, 16) | 0       |
| flatten (Flatten)            | (None, 2704)       | 0       |
| dense_1 (Dense)              | (None, 10)         | 27050   |
| dropout (Dropout)            | (None, 10)         | 0       |
| dense (Dense)                | (None, 10)         | 110     |

Total params: 27,320

Trainable params: 27,320

Non-trainable params: 0

## CNN - Exemplo 2

- O modelo é compilado com a função perda entropia cruzada e métrica acurácia:

```
model %>% compile(
 loss = loss_categorical_crossentropy,
 optimizer = optimizer_adadelta(),
 metrics = c('accuracy')
)
```

- Número de mini-batches e número de épocas:

```
batch_size <- 128
epochs <- 50
```

- Treinando o modelo com 50 iterações (épocas):

```
model %>% fit(
 x_train, y_train,
 batch_size = batch_size,
 epochs = epochs,
 validation_split = 0.2
)
```

## CNN - Exemplo 2

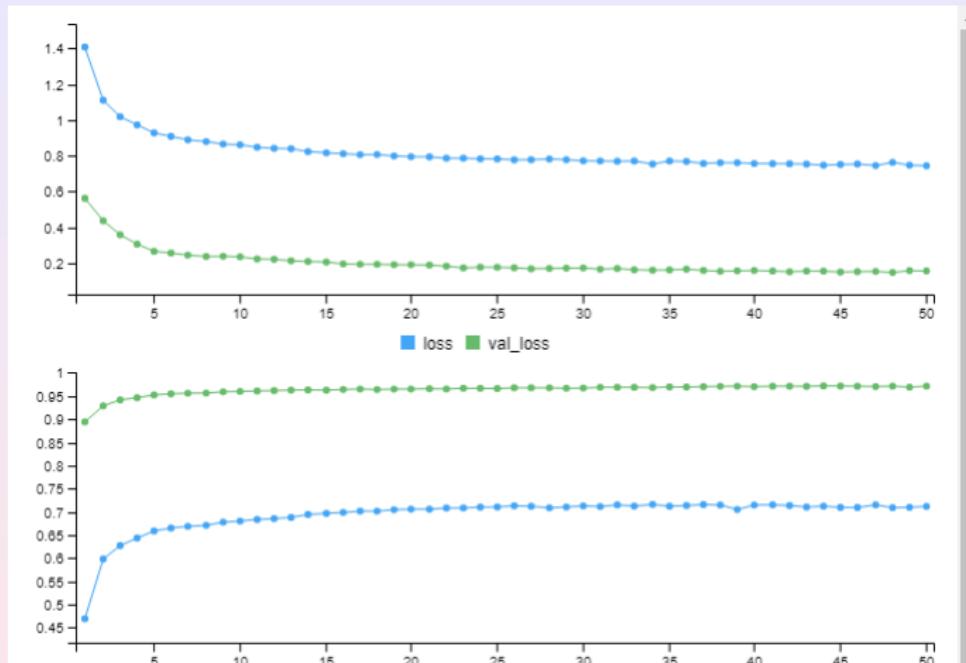
- Avaliando a perda e acurácia no conjunto teste:

```
> score <- model \%>\% evaluate(x_test,
+ y_test)
313/313 - 1s 2ms/step - loss: 0.1569 - accuracy: 0.9723
```

- Ou seja, a perda (dada pela entropia cruzada) no conjunto teste foi 0,1569 e a acurácia no conjunto teste 0,9723.
- O gráfico da Figura 3 mostra a evolução das perdas (medidas pela entropia cruzada) e da acurácia do procedimento.
- Um código em Python para o mesmo conjunto de dados pode ser encontrado em

[https://keras.io/examples/vision/mnist\\_convnet/](https://keras.io/examples/vision/mnist_convnet/)

## CNN - Exemplo 2



**Figura 3:** Perdas (entropia, painel superior) e acurácia(painel inferior) para o Exemplo 2.

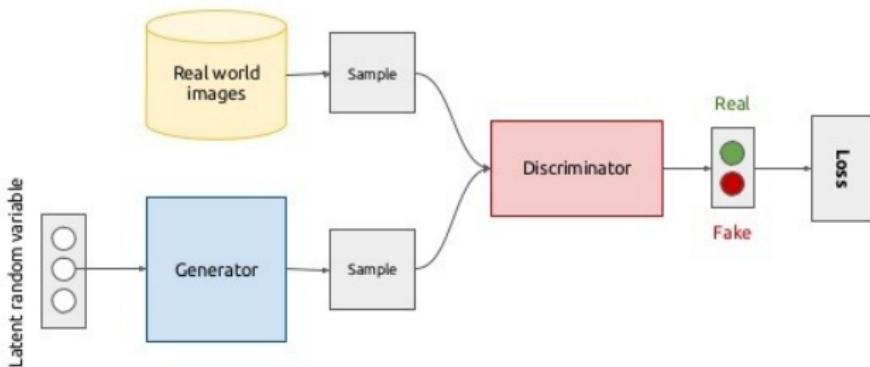
# Redes generativas adversárias

- As redes generativas adversárias (*generative adversarial networks*, GAN) foram introduzidas por Goodfellow et al. (2014). Elas consistem de duas redes neurais treinadas em oposição uma à outra: o **gerador** (generator) G e o **discriminador** (discriminator) D.
- Considere uma matriz de dados  $\mathbf{X}$ , e suponha que temos exemplos, considerados amostras i.i.d. de  $\mathbf{X}$ , com densidade de probabilidade  $p(\mathbf{x})$ , desconhecida.
- O gerador G tem como entrada um vetor de ruídos aleatórios  $\mathbf{z}$  e saída uma imagem  $X_{\text{falso}} = G(\mathbf{z})$ , com a mesma distribuição  $p(\mathbf{x})$  dos dados reais.
- Por sua vez, o discriminador D depara-se com um exemplo real,  $\mathbf{X} \sim p(\mathbf{x})$  ou um exemplo falso,  $X_{\text{falso}}$  gerado por G.
- Quem é apresentado é decidido jogando-se uma moeda honesta (a probabilidade de ser falso é  $1/2$ ). A moeda é lançada independentemente em cada rodada do jogo.
- O discriminador usa uma função  $D(\mathbf{x}) = P(\mathbf{x} \text{ real})$  (o classificador), tal que se  $D(\mathbf{x}) > 0,5$  então  $\mathbf{x}$  é real; se  $D(\mathbf{x}) < 0,5$ , então  $\mathbf{x}$  é falso.

# Redes gerativas adversárias

- Essa é a **regra de classificação**. D gostaria que  $D(x) \approx 1$  quando  $x$  for real e  $D(x) \approx 0$  quando falso.
- Na Figura 4 temos o esquema de uma rede neural GAN.

## Generative adversarial networks (conceptual)



**Figura 4:** Rede neural geradora adversária.

# Redes generativas adversárias

- Para gerar  $\mathbf{X}$  a partir de  $\mathbf{Z}$  usamos o Método Monte Carlo. De modo geral, a função de distribuição acumulada (f.d.a.) de um vetor  $\mathbf{X} = (X_1, \dots, X_d)$  pode ser escrita como

$$F_{X_1, \dots, X_d}(x_1, \dots, x_d) = F_{X_1}(x_1)F_{X_2|X_1}(x_2|x_1) \cdots F_{X_d|X_1, \dots, X_{d-1}}(x_d|x_1, \dots, x_{d-1}).$$

- A seguir, geramos  $d$  variáveis aleatórias i.i.d.  $\mathcal{U}(0, 1)$ , digamos  $U_1, \dots, U_d$  e geramos o vetor  $\mathbf{X}$  por meio de:

$$\begin{aligned} x_1 &= F_{X_1}^{-1}(u_1), \\ x_2 &= F_{X_2|X_1}^{-1}(u_2) \\ &\vdots \\ x_d &= F_{X_d|X_1, \dots, X_{d-1}}^{-1}(u_d). \end{aligned}$$

- Em nosso caso, sabemos que existe uma função  $G$  que transforma ruídos  $\mathbf{Z}$  num vetor  $\mathbf{X}$  com a densidade  $f(\mathbf{x})$ . Mas nesse caso, não conhecemos a distribuição de  $\mathbf{X}$  e será difícil obter a f.d.a.

# Redes generativas adversárias

- Como mencionamos acima, a solução é imaginar que exista essa  $G$  na forma de uma rede neural e o gerador  $G(\mathbf{z})$  terá parâmetros  $\theta^{(G)}$  (os pesos da rede neural). Teremos que aprender os parâmetros dessa rede de forma aproximada.
- A função perda do discriminador é dada por

$$J^{(D)}(\theta) = -\frac{1}{2} E_{\mathbf{p} \text{ real}}[\log(D(\mathbf{X}))] - \frac{1}{2} E_{\mathbf{z}}[\log((1 - D(G(\mathbf{z})))]. \quad (1)$$

- O tipo de jogo a usar é o de soma zero: a perda de um jogador é o ganho do outro e a soma das perdas dos dois jogadores é zero. Neste caso, a função perda do gerador é dada por

$$J^{(G)}(\theta^{(G)}, \theta^{(D)}) = -J^{(D)}(\theta^{(G)}, \theta^{(D)}). \quad (2)$$

- A solução de um jogo de soma zero é chamada **solução minimax**. O gerador  $G$  quer determinar seus pesos  $\theta^{(G)}$  que minimizem sua perda, dada por

$$J^{(G)}(\theta^{(G)}, \theta^{(D)}) = \frac{1}{2} E_{\mathbf{p} \text{ dados}}[\log(D(\mathbf{X}))] + \frac{1}{2} E_{\mathbf{z}}[\log((1 - D(G(\mathbf{z})))]. \quad (3)$$

# Redes generativas adversárias

- Note que  $\theta^{(G)}$  somente aparece na segunda parcela de (3). Portanto, podemos ignorar a primeira parcela ao otimizar essa expressão e usando apenas os dados  $\mathbf{Z}$ . A solução que minimiza as duas perdas individualmente é a **solução minimax**; para o jogador G será aquela que minimiza a perda  $J^{(G)}$  sobre os parâmetros  $\theta^{(G)}$  enquanto maximiza esta perda sobre  $\theta^{(D)}$  e procedimento similar para  $J^{(D)}$ .
- Um método iterativo usa otimização via gradiente descendente estocástico e este pode ou não convergir. Estudos mostram que GANs podem produzir bons resultados em dados com baixas variabilidade e resolução.
- Várias variantes do GAN surgiram para melhorá-lo:
  - DCGAN: usa redes convolucionais profundas;
  - cGAN, ACGAN: conditional GAN
  - WGAN, WGAN-GP: diferentes funções perda.
- Para detalhes veja Assunção (2022).

## Redes generativas adversárias - Exemplo

- Neste exemplo, voltamos a usar os dados Mnist e uma versão do GAN, a ACGAN. Um código em R pode ser encontrado em

[https://github.com/rstudio/keras/blob/main/vignettes/examples/mnist\\_acgan.R](https://github.com/rstudio/keras/blob/main/vignettes/examples/mnist_acgan.R)

- Vemos, abaixo, a última iteração do algoritmo, indicando as perdas do discriminador, 1,3604, e a do gerador, 0,9898.

Epoch 20/20

235/235 [=====] - 203s 863ms/step - d\_loss: 1.3604 - g\_loss: 0.9898

- Na Figura 5 notamos que, enquanto a função perda do discriminador diminui ao longo das épocas, o mesmo não acontece com a perda do gerador. Na Figura 6 temos o dígito 6 previsto corretamente pelo modelo.

## Redes generativas adversárias - Exemplo

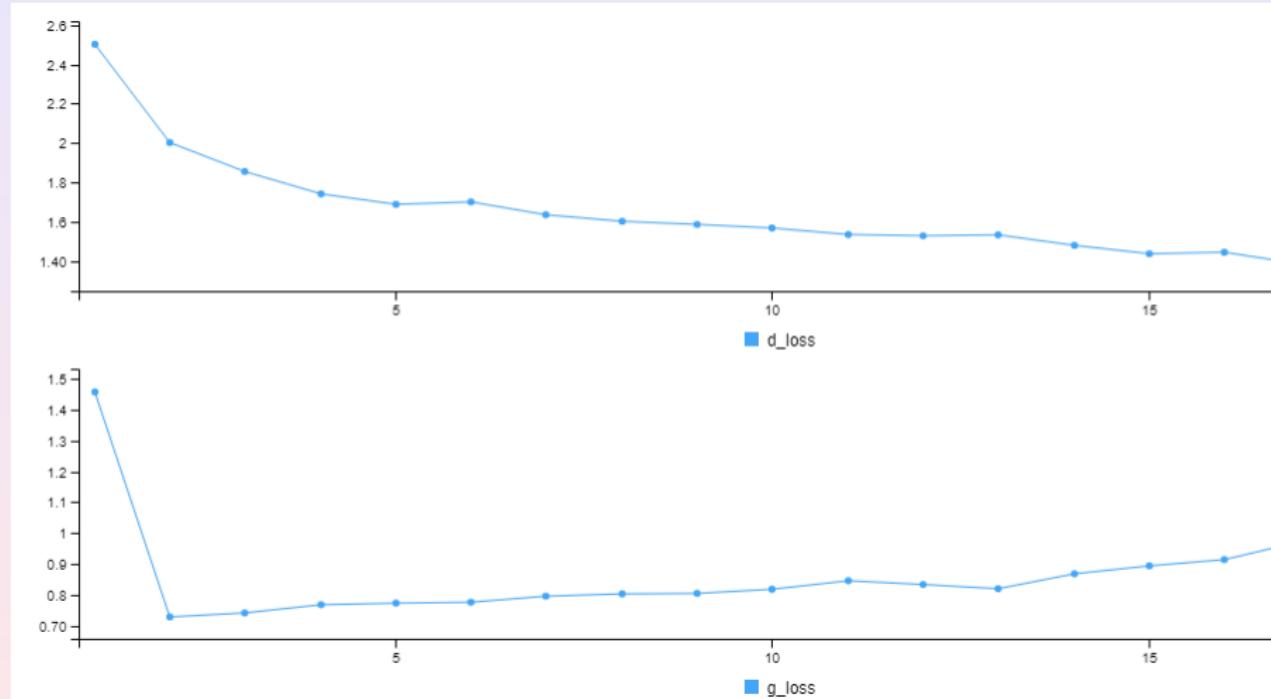


Figura 5: Comportamento das perdas do gerador e discriminador para o Exemplo.

## Redes generativas adversárias - Exemplo

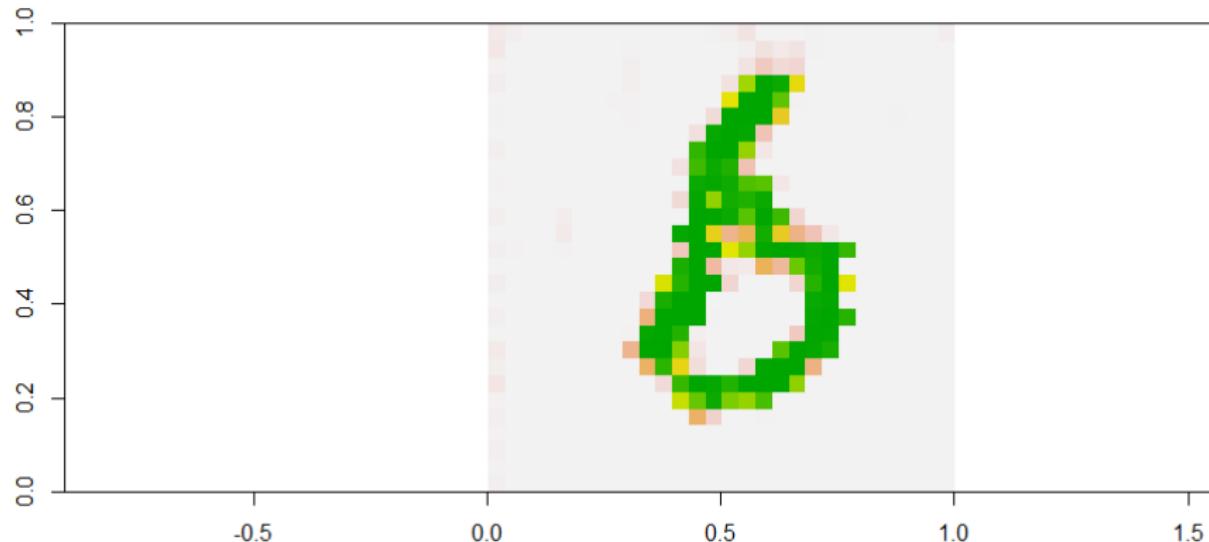


Figura 6: Dígito 6 previsto pelo modelo.

## Referências

- Assunção, R. (2022). Deep Learning. Short Course, SINAPE 2022, Gramado, RS.
- Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y. (2014). Generative adversarial networks, arXiv:1406.2661.
- Morettin, P. A. and Singer, J. M. (2023). *Estatística e Ciência de Dados*. 2a. edição. LTC.

# MAE 5905: Introdução à Ciência de Dados

Pedro A. Morettin

Instituto de Matemática e Estatística  
Universidade de São Paulo  
pam@ime.usp.br  
<http://www.ime.usp.br/~pam>

## Aula 17

26 de junho de 2023

# Sumário

## 1 Simulação estática

## 2 Métodos de reamostragem

## Preliminares

- Sabemos como construir alguns modelos probabilísticos para representar uma situação real, ou então para descrever um experimento aleatório. Notamos, também, que determinados um espaço amostral e probabilidades associadas aos pontos desse espaço, o modelo probabilístico ficará completamente determinado e poderemos, então, calcular a probabilidade de qualquer evento aleatório.
- Muitas vezes, mesmo construindo um modelo probabilístico, certas questões não podem ser resolvidas analiticamente e teremos que recorrer a **estudos de simulação** para obter aproximações de quantidades de interesse.
- Estudos de simulação tentam reproduzir num ambiente controlado o que se passa com um problema real. Para nossos propósitos, a solução de um problema real consistirá na simulação de **variáveis aleatórias** (**simulação estática**) ou de **processos estocásticos** (**simulação dinâmica**).
- A simulação de variáveis aleatórias deu origem aos chamados **métodos Monte Carlo** (MMC), que por sua vez supõem que o pesquisador disponha de um **gerador de números aleatórios**.

## Preliminares

- Um **número aleatório** (NA) representa o valor de uma variável aleatória uniformemente distribuída no intervalo  $(0, 1)$ . Originalmente, esses NA eram gerados manualmente ou mecanicamente, usando dados, roletas etc. Modernamente, usamos computadores para gerar NA.
- Os MMC apareceram durante a segunda guerra mundial , em pesquisas relacionadas à difusão aleatória de neutrons num material radioativo. Os trabalhos pioneiros devem-se a Metropolis e Ulam (1949), Metropolis et al. (1953) e von Neumann (1951). Veja Sóbol (1976), Hammersley e Handscomb (1964) e Ross (1997).
- Para ilustrar, suponha que se queira calcular a área da figura F contida no quadrado Q de lado unitário (Figura 1). Suponha que sejamos capazes de gerar pontos aleatórios em Q, de modo homogêneo, isto é, de modo a cobrir toda a área do quadrado, ou ainda, que estes pontos sejam *uniformemente distribuídos sobre Q*. Se gerarmos  $N$  pontos, suponha que  $N'$  desses caiam em F. Então, poderemos aproximar a área de F por  $N'/N$ . Quanto mais pontos gerarmos, melhor será a aproximação.
- Note que o problema em si não tem nenhuma componente aleatória: queremos calcular a área de uma figura plana. Mas, para resolver o problema, uma possível maneira foi considerar um mecanismo aleatório. Veremos que esse procedimento pode ser utilizado em muitas situações.

# Preliminares

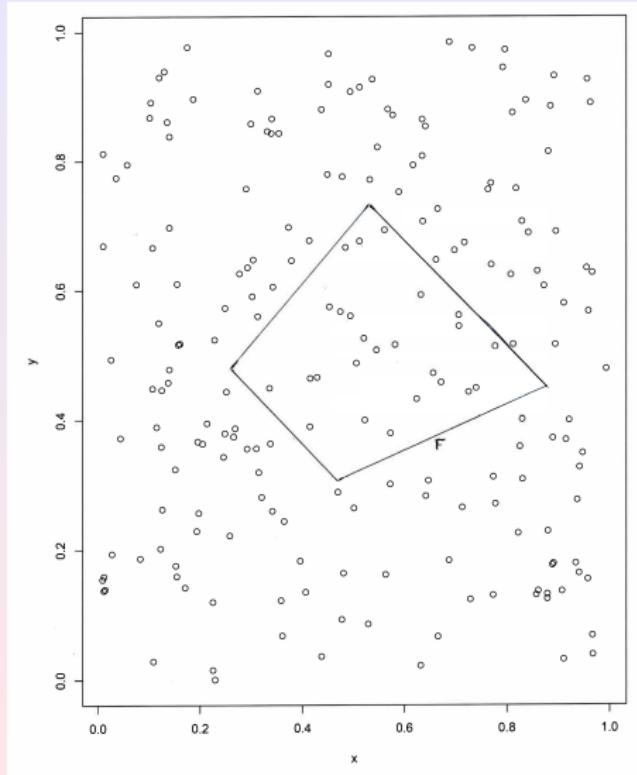


Figura: Área de uma figura por simulação.

## Preliminares

- Dissemos acima que um NA é um valor de uma variável aleatória uniformemente distribuída no intervalo  $(0, 1)$ . Vejamos algumas maneiras de obter um NA.
- **Exemplo 1.** Lance uma moeda 3 vezes e atribua o valor 1 se ocorrer cara e o valor 0 se ocorrer coroa. Os resultados possíveis são as *sequências* ou **números binários** abaixo:

000, 001, 010, 011, 100, 101, 110, 111.

- Cada um desses números binários corresponde a um número decimal. Por exemplo,  $(111)_2 = (7)_{10}$ , pois  $(111)_2 = 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$ . Considere a representação decimal de cada sequência acima e divida o resultado por  $2^3 - 1 = 7$ . Obteremos os números aleatórios  $0, 1/7, 2/7, \dots, 1$ . Qualquer uma das 8 sequências acima tem a mesma probabilidade, a saber,  $1/2^3 = 1/8$ .
- De modo geral, lançando-se a moeda  $n$  vezes, teremos  $2^n$  possibilidades e cada uma terá probabilidade  $1/2^n$  e os NA finais são obtidos por meio de divisão por  $2^n - 1$ .

## Preliminares

- O que se faz atualmente é fazer **simulação por meio de computadores**, que utiliza **números pseudo-aleatórios**, no lugar de NA.
- Os números pseudo-aleatórios (NPA) são obtidos por meio de técnicas que usam relações matemáticas recursivas **determinísticas**. Logo, um NPA gerado numa iteração dependerá do número gerado na iteração anterior, e portanto não será realmente aleatório, donde o nome pseudo-aleatório.
- Um método bastante utilizado em pacotes computacionais é o método congruencial
- O R usa o comando **runif(n,min,max)**, onde *n* é o número de valores a gerar e (*min*, *max*) é o intervalo no qual se quer gerar os NPA. No nosso caso, *min*=0 e *max*=1.
- Outros pacotes têm seus próprios geradores de NPA, como o MatLab.
- **Exemplo 2.** O comando  $u \leftarrow \text{runif}(10,0,1)$  pede para gerar 10 NA e guardá-los no vetor *u*.

```
> u<-runif(10,0,1)
```

```
> u
```

```
[1] 0.80850094 0.56611676 0.75882010 0.89910843 0.48447125
[6] 0.02119849 0.06239355 0.30022882 0.12722598 0.49714446
```

# Preliminares

Existem três grandes grupos de métodos de simulação:

- 1) **Métodos de Simulação Estática.** Aqui, os procedimentos têm por objetivo gerar amostras independentes. Citamos os métodos Monte Carlo, aceitação/rejeição e reamostragem ponderada.
- 2) **Métodos de Simulação por Imputação.** A ideia desses métodos é aumentar os dados, introduzindo **dados latentes**, com o intuito de facilitar a simulação. Dentre esses métodos citamos o algoritmo EM ( de *expectation-maximization*) e o algoritmo de dados aumentados.
- 3) **Métodos de Simulação Dinâmica.** Esses métodos são denominados atualmente por MCMC (**Markov Chain Monte Carlo**) e têm por objetivo construir uma cadeia de Markov, cuja distribuição de equilíbrio seja a distribuição da qual queremos amostrar. Os métodos mais importantes aqui são o amostrador de Gibbs e os algoritmos de Metropolis e Metropolis-Hastings.

## Métodos Monte Carlo

- Suponha uma v.a. com distribuição  $F$  e desejamos calcular a média de uma função qualquer  $h(X)$ . Suponha, ainda, que exista um método para simular uma amostra  $X_1, \dots, X_n$  de  $F$ . Nas seções seguintes veremos alguns desses métodos. Então, o método Monte Carlo (MMC) consiste em aproximar  $\mu_F = E_F[h(X)]$  por

$$\hat{\mu}_F = \hat{E}_F[h(X)] = \frac{1}{n} \sum_{i=1}^n h(X_i). \quad (1)$$

- Observe que (14) aproxima a integral  $\int h(x)dF(x)$  ou  $\int h(x)f(x)dx$ , se existir a densidade de  $X$ . A lei (forte) dos grandes números garante que, quando  $n \rightarrow \infty$ ,  $\hat{\mu}_F$  converge para  $\mu_F$  com probabilidade um.
- O erro padrão da estimativa (14) é dado pela raiz quadrada da variância de  $\hat{\mu}_F$ , denotada por  $\text{Var}(\hat{\mu}_F)$ . Esta, por sua vez, pode ser estimada por  $\widehat{\text{Var}}(\hat{\mu}_F)$  e, portanto, uma estimativa do erro padrão de  $\hat{\mu}_F$  será

$$\widehat{\text{EP}}(\hat{\mu}_F) = \frac{1}{\sqrt{n}} \left[ \sum_{i=1}^n (h(X_i) - \hat{\mu}_F)^2 \right]^{1/2} = O(n^{-1/2}). \quad (2)$$

## MMC – Exemplo 3

- **Exemplo 3.** Suponha que se queira calcular o valor esperado de  $h(X)$ , onde  $h(x) = \sqrt{1 - x^2}$  e  $F \sim \mathcal{U}(0, 1)$ . Então, se  $X_1, \dots, X_n$  for uma amostra da distribuição uniforme padrão,

$$\hat{E}_F[h(X)] = \frac{1}{n} \sum_{i=1}^n \sqrt{1 - X_i^2}.$$

- Por exemplo, gerando-se 1.000 valores de uma  $\mathcal{U}(0, 1)$ , obtivemos o valor 0,7880834. Observe que essa é também, uma estimativa de um quarto da área de um círculo unitário, ou seja,  $\pi/4 = 0,7853982$ . O erro padrão calculado por (2) é 0,0069437.
- Uma outra aplicação do MMC é na obtenção de amostras de distribuições marginais. Suponha, por exemplo, que as v.a.  $X$  e  $Y$  tenham densidade conjunta  $f(x, y)$  e marginais  $f_X(x)$  e  $f_Y(y)$ , respectivamente. Então,

$$f_Y(y) = \int_{-\infty}^{\infty} f(x, y) dx = \int_{-\infty}^{\infty} f_X(x) f_{Y|X}(y|x) dx, \quad (3)$$

onde  $f_{Y|X}(y|x)$  é a densidade condicional de  $Y$  dado que  $X = x$ .

## MMC – Exemplo 3

Para obter uma amostra de  $f_Y(y)$  procedemos como segue (**método da composição ou mistura**):

- obtemos um elemento  $x^*$  de  $f_X(x)$ ;
- fixado  $x^*$ , obtemos um elemento  $y^*$  de  $f_{Y|X}(y|x^*)$ .

Repetimos os passos (a) e (b)  $n$  vezes, obtendo-se  $(x_1, y_1), \dots, (x_n, y_n)$  como uma amostra de  $f(x, y)$ , enquanto que  $y_1, \dots, y_n$  representa uma amostra de  $f_Y(y)$ .

É óbvio que devemos saber como amostrar das densidades  $f_X(x)$  e  $f_{Y|X}(y|x)$ ;  $x^*$  é chamado o **elemento misturador**.

## Simulação de variáveis discretas

- Vimos que a geração de NAs corresponde a gerar valores de uma distribuição uniforme no intervalo  $(0, 1)$
- Se  $U \sim \mathcal{U}(0, 1)$  e se  $0 < a < b < 1$ , então

$$P(a < U < b) = b - a. \quad (4)$$

- Considere, agora, uma v.a. qualquer  $X$ , com a distribuição de probabilidades dada abaixo:

$$\begin{aligned} X &: & x_1, & x_2, & \dots, & x_n \\ p_j &: & p_1, & p_2, & \dots, & p_n \end{aligned}$$

## Simulação de variáveis discretas

- Geramos, agora, um NA  $u$ ; Coloque:

$$X = \begin{cases} x_1, & \text{se } u < p_1, \\ x_2, & \text{se } p_1 \leq u < p_1 + p_2, \\ \dots \\ x_j, & \text{se } p_1 + \dots + p_{j-1} \leq u < p_1 + \dots + p_j. \end{cases} \quad (5)$$

- Como

$$P(X = x_j) = P(p_1 + \dots + p_{j-1} \leq U < p_1 + \dots + p_j) = p_j,$$

usando (4), vemos que  $X$  tem a distribuição que queremos.

- **Exemplo 4. Simulação de Uma Distribuição de Bernoulli.**
- Suponha que  $X$  tenha uma distribuição de Bernoulli, com  $P(X = 0) = 1 - p = 0,48$  e  $P(X = 1) = p = 0,52$ . Para gerar valores de tal distribuição basta gerar NA  $u$  e concluir:
  - Se  $u < 0,48$ , coloque  $X = 0$ ;

Se  $u \geq 0,48$ , coloque  $X = 1$ .

## Simulação de variáveis discretas

- Por exemplo, suponha que geramos dez NA : 0, 11; 0, 82; 0, 00; 0, 43; 0, 56; 0, 60; 0, 72; 0, 42; 0, 08; 0, 53. Então, os dez valores gerados da distribuição em questão são 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, respectivamente.
- Exemplo 5. Simulação de Uma Distribuição Binomial.**
- Sabemos que se  $Y \sim b(n, p)$ , então  $Y$  é o número de sucessos num experimento de Bernoulli, com  $n$  repetições, e probabilidade de sucesso  $p$ .
- No Exemplo 4, obtivemos 5 sucessos, logo  $Y = 5$ . Portanto, se  $Y \sim b(10; 0,52)$ , e quisermos, digamos, gerar 20 valores dessa distribuição, basta considerar 20 experimentos de Bernoulli, sendo que em cada um deles repetimos o experimento  $n = 10$  vezes, com probabilidade de sucesso  $p = 0,52$ .
- Para cada experimento  $j$  consideramos o número de sucessos (número de 1),  $y_j$ ,  $j = 1, 2, \dots, 20$ . Obteremos, então, os 20 valores simulados  $y_1, \dots, y_{20}$  da v.a.  $Y$ . Observe que esses valores serão inteiros entre 0 e 20, inclusive esses dois valores.

# Simulação de variáveis discretas

- **Exemplo 6. Simulação de Uma Distribuição de Poisson.**

Se  $N \sim P(\lambda)$ , então  $P(N = j) = p_j$  é dada por

$$P(N = j) = \frac{e^{-\lambda} \lambda^j}{j!}, \quad j = 0, 1, \dots \quad (6)$$

- A geração de valores de uma distribuição de Poisson parte da seguinte relação recursiva, que pode ser facilmente verificada:

$$p_{j+1} = \frac{\lambda}{j+1} p_j, \quad j \geq 0. \quad (7)$$

- Seja  $F(j) = P(N \leq j)$  a função de distribuição acumulada (f.d.a.) de  $N$ .
- Considere  $j$  o valor atual gerado e queremos gerar o valor seguinte. Chamemos simplesmente  $p = p_j$  e  $F = F(j)$ . Então o algoritmo para se gerar os sucessivos valores é o seguinte:

## Simulação de variáveis discretas

Passo 1: Gere o NA  $u$ ;

Passo 2: Faça  $j = 0$ ,  $p = e^{-\lambda}$  e  $F = p$ ;

Passo 3: Se  $u < F$ , coloque  $N = j$ ;

Passo 4: Faça  $p = \frac{\lambda}{j+1}p$ ,  $F = F + p$  e  $j = j + 1$ ;

Passo 5: Volte ao Passo 3.

Note que, no Passo 2, se  $j = 0$ ,  $P(N = 0) = p_0 = e^{-\lambda}$  e  $F(0) = P(N \leq 0) = p_0$ .

## Simulação de variáveis discretas

Suponha, por exemplo, que queiramos simular valores de uma distribuição de Poisson com parâmetro  $\lambda = 2$ . Então  $e^{-2} = 0,136$ . Obtemos:

Passo 1: Geramos  $u = 0,35$ ;

Passo 2:  $j = 0$ ,  $p = 0,136$ ,  $F = 0,136$ ;

Passo 3:  $u > F$ ;

Passo 4:  $p = 2(0,136) = 0,272$ ,  $F = 0,136 + 0,272 = 0,408$ ,  $j = 1$ ;

Passo 5: voltemos ao Passo 3; com  $u < F$ , colocamos  $N = 1$ . Temos, portanto o primeiro valor gerado da distribuição. Prosseguimos com o algoritmo para gerar outros valores.

# Simulação de variáveis discretas

O R e a planilha Excel possuem subrotinas próprias para simular valores de uma dada distribuição de probabilidades. A Tabela 1 traz as distribuições discretas contempladas por cada um e os comandos apropriados.

Tabela 1- Opções de Distribuições Discretas

| Distribuição    | Excel(Par.)                | R(Par.)                   |
|-----------------|----------------------------|---------------------------|
| Bernoulli       | <code>Bernoulli(p)</code>  | –                         |
| Binomial        | <code>Binomial(n,p)</code> | <code>binom(n,p)</code>   |
| Geométrica      | –                          | <code>geom(p)</code>      |
| Hipergeométrica | –                          | <code>hyper(N,r,k)</code> |
| Poisson         | <code>Poisson(λ)</code>    | <code>pois(λ)</code>      |

## Simulação de variáveis contínuas

- Considere uma v.a.  $X$ , com função de distribuição acumulada  $F$ , representada na Figura 2. Usando-se um gerador de NA, obtemos um valor  $u$ . Marca-se esse valor no eixo das ordenadas e por meio da função inversa de  $F$  obtém-se o valor  $x$  da v.a.  $X$  no eixo das abscissas. Ou seja, estamos resolvendo a equação

$$F(x) = u, \quad (8)$$

ou  $x = F^{-1}(u)$ . Formalmente, estamos usando o **método da transformação integral**, consubstanciada no seguinte resultado. Suponha  $F$  estritamente crescente.

- **Proposição.** Se  $X$  for uma v.a. com f.d.a  $F$ , então a v.a.  $U = F(X)$  tem distribuição uniforme no intervalo  $[0, 1]$ .
- O resultado pode ser estendido para o caso de  $F$  ser não decrescente, usando-se uma definição mais geral de inversa.

## Simulação de variáveis contínuas

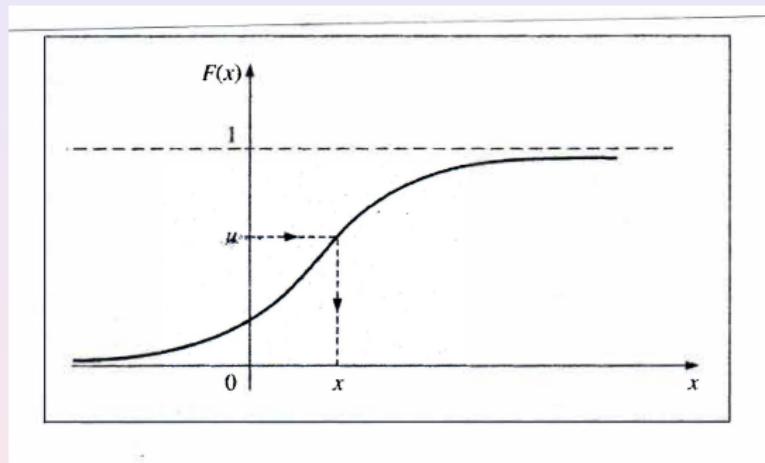


Figura: Função de distribuição acumulada de uma v.a.  $X$ .

## Simulação de variáveis contínuas

## • Exemplo 7. Simulação de uma Distribuição Exponencial.

Se a v.a.  $T$  tiver densidade dada por

$$f(t) = \frac{1}{\beta} e^{-t/\beta}, \quad t > 0, \tag{9}$$

a sua f.d.a. é dada por

$$F(t) = 1 - e^{-t/\beta}, \tag{10}$$

logo temos que resolver esta equação para gerar  $t$ .

• Tomando logaritmo na base  $e$ , temos

$$1 - u = e^{-t/\beta} \Leftrightarrow \log(1 - u) = -\frac{t}{\beta} \Leftrightarrow t = -\beta \log(1 - u).$$

Logo, gerado um NA, um valor da distribuição  $\text{Exp}(\beta)$  é dado por  $-\beta \log(1 - u)$ .

## Simulação de variáveis contínuas

- Por exemplo, suponha  $\beta = 2$  e queremos gerar 5 valores de  $T \sim \text{Exp}(2)$ . Gerados os valores

$u_1 = 0,57, u_2 = 0,19, u_3 = 0,38, u_4 = 0,33, u_5 = 0,31$  de uma distribuição uniforme em  $(0, 1)$  (os números aleatórios), obteremos

$$t_1 = (-2)(\log(0,43)) = 1,68, \quad t_2 = (-2)(\log(81)) = 0,42,$$

$$t_3 = (-2)(\log(0,62)) = 0,96, \quad t_4 = (-2)(\log(0,67)) = 0,80,$$

$$t_5 = (-2)(\log(0,69)) = 0,74.$$

- Podemos reduzir um pouco os cálculos se usarmos o seguinte fato: se  $U \sim \mathcal{U}(0, 1)$ , então  $1 - U \sim \mathcal{U}(0, 1)$ . Resulta que poderemos gerar os valores de uma exponencial por meio de

$$t = -\beta \log(u).$$

- Usando esta fórmula para os valores de  $U$  acima, obteremos os seguintes valores de  $T$  : 1,12; 3,32; 1,93; 0,96; 2,34.

## Simulação de variáveis contínuas

## • Exemplo 8. Simulação de uma Distribuição Normal.

Há vários métodos para gerar v.a. normais, mas uma observação importante é que basta gerar uma v.a. normal padrão, pois qualquer outra pode ser obtida desta. De fato, gerado um valor  $z_1$  da v.a.  $Z \sim N(0, 1)$ , para gerar um valor de uma v.a.  $X \sim N(\mu, \sigma^2)$  basta usar a transformação  $z = (x - \mu)/\sigma$  para obter

$$x_1 = \mu + \sigma z_1. \quad (11)$$

- Um método usa a transformação integral e uma tabela de probabilidades para a normal padrão.
- Vejamos um exemplo. Suponha que  $X \sim N(10; 0, 16)$ , ou seja,  $\mu = 10$  e  $\sigma = 0, 4$ . Temos que resolver a equação (7), ou seja,

$$\Phi(z) = u,$$

onde estamos usando a notação  $\Phi(z)$  para a f.d.a. da  $N(0, 1)$ .

- Por exemplo, gerado um NA  $u = 0, 230$ , temos que resolver

$$\Phi(z) = 0, 230,$$

ou seja, temos que encontrar o valor  $z$  tal que a área à sua esquerda, sob a curva normal padrão, seja 0, 230. Veja a Figura 3.

# Simulação de variáveis contínuas

Consultando uma tabela para a normal, encontramos que  $z = -0,74$ . Logo o valor gerado da normal em questão satisfaz

$$\frac{x - 10}{0,4} = -0,74,$$

ou seja,  $x = 10 + (0,4)(-0,74) = 9,704$ . Qualquer outro valor pode ser gerado da mesma forma.

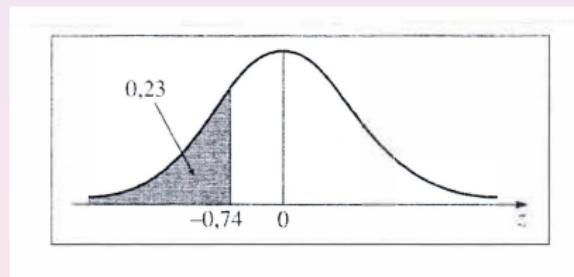


Figura: Geração de um valor  $z \sim N(0, 1)$ .

# Simulação de variáveis contínuas

- Há outros métodos mais eficientes. Alguns são variantes do método de Box - Müller (1958).
- Nesse método, são geradas duas v.a.  $Z_1$  e  $Z_2$ , independentes e  $N(0, 1)$ , por meio das transformações

$$\begin{aligned} Z_1 &= \sqrt{-2 \log U_1} \cos(2\pi U_2), \\ Z_2 &= \sqrt{-2 \log U_1} \sin(2\pi U_2), \end{aligned} \tag{12}$$

em que  $U_1$  e  $U_2$  são v.a. com distribuição uniforme em  $[0, 1]$ .

- Portanto, basta gerar dois NAs  $u_1$  e  $u_2$  e depois gerar  $z_1$  e  $z_2$  usando (12).
- O método de Box-Müller pode ser computacionalmente ineficiente, pois necessita calcular senos e cossenos. Uma variante, chamado de método polar, evita esses cálculos.

## Simulação de variáveis contínuas

Com o R podemos usar o comando `qnorm`, para obter um quantil de uma distribuição normal, a partir de sua f.d.a. Por exemplo, para gerar 1.000 valores de uma distribuição normal padrão, usamos:

```
u <- runif(1000,0,1) # gera 1000 observações de uma uniforme[0,1]
x <- qnorm(u,mean=0, sd = 1) # Calcula os quantis para o vetor
 u simulado da uniforme
```

```
par(mfrow=c(1,2))
hist(u, freq=FALSE, main="Histograma da amostra da distribuição
 Uniforme simulada")
hist(x, freq=FALSE, main="Histograma da variável X simulada
 a partir do resultado do Teorema 15.1")
```

Os histogramas, da uniforme e da normal, estão na Figura 4.

# Simulação de variáveis contínuas

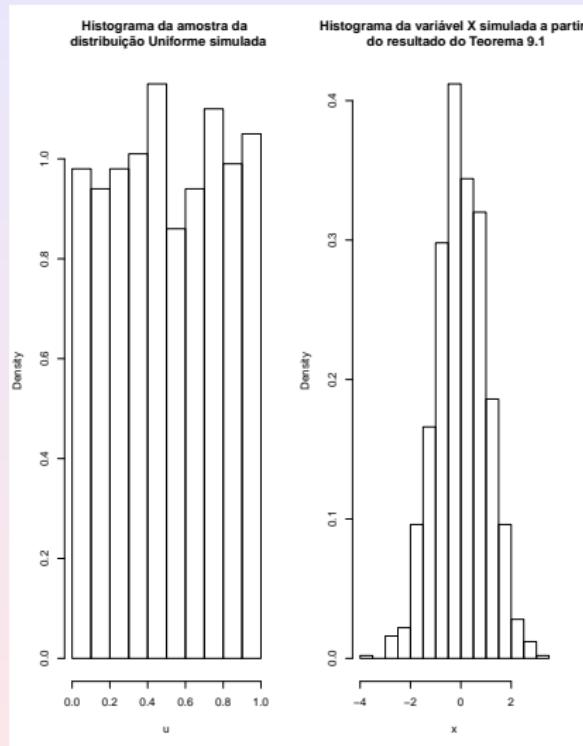


Figura: Simulação de uma normal padrão via R.

## Simulação de variáveis contínuas

O R e a planilha Excel têm subrotinas próprias para gerar muitas das distribuições estudadas nesta seção. A Tabela 2 mostra as opções disponíveis e os comandos apropriados. Além da  $N(0,1)$  o Excel usa a função INV para gerar algumas outras distribuições contínuas.

Tabela 2: Opções de Distribuições Contínuas

| Distribuição | Excel(Par.) | R(Par.)                       |
|--------------|-------------|-------------------------------|
| Normal       | Normal(0,1) | $\text{norm}(\mu, \sigma)$    |
| Exponencial  | –           | $\text{exp}(\beta)$           |
| t(Student)   | –           | $t(\nu)$                      |
| F(Snedecor)  | –           | $f(\nu_1, \nu_2)$             |
| Gama         | –           | $\text{gamma}(\alpha, \beta)$ |
| Qui-Quadrado | –           | $\text{chisq}(\nu)$           |
| beta         | –           | $\text{beta}(\alpha, \beta)$  |

# Simulação de vetores aleatórios

- É mais complicado simular distribuições bidimensionais. No caso de  $X$  e  $Y$  serem **independentes**, então

$$f(x, y) = f_X(x)f_Y(y), \quad \forall x, y,$$

se elas forem contínuas, por exemplo. Logo, para gerar um valor  $(x, y)$  da densidade conjunta  $f(x, y)$ , basta gerar a componente  $x$  da distribuição marginal de  $X$  e a componente  $y$  da distribuição marginal de  $Y$ , **independentemente**.

- No caso de v.a. **dependentes**, temos que vale a relação:

$$f(x, y) = f_X(x)f_{Y|X}(y|x).$$

- Logo, por essa relação, primeiramente geramos um valor  $x$  da distribuição marginal de  $X$  e fixado esse valor,  $x_0$ , digamos, geramos um valor da distribuição condicional de  $X$ , dado que  $X = x_0$ . Isso implica que devemos saber como gerar valores das distribuições  $f_X(x)$  e  $f_{Y|X}(y|x)$ .

## Simulação de vetores aleatórios

**Exemplo 8. Distribuição Normal Bidimensional.**

O método de Box-Müller gera valores de duas normais padrões independentes,  $Z_1$  e  $Z_2$ . Logo, se quisermos gerar valores da distribuição conjunta de  $X$  e  $Y$ , **independentes e normais**, com  $X \sim N(\mu_x, \sigma_x^2)$  e  $Y \sim N(\mu_y, \sigma_y^2)$ , basta considerar

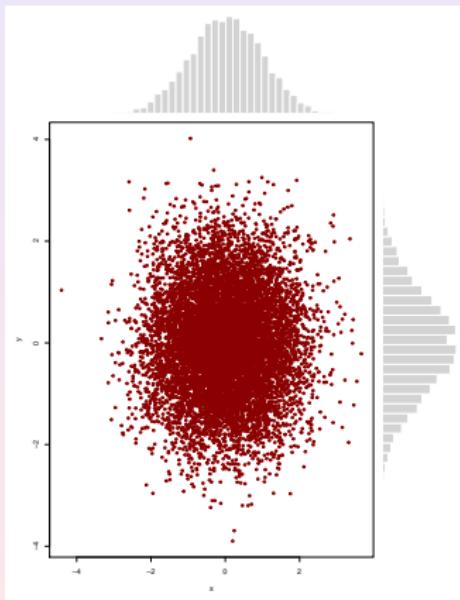
$$X = \mu_x + \sigma_x Z_1, \quad Y = \mu_y + \sigma_y Z_2.$$

O código em R, para o caso de duas normais padões, seria:

```
u1<-runif(10000,0,1)
u2<-runif(10000,0,1)
P<-data.frame(u1,u2)
x<-qnorm(u1)
y<-qnorm(u2)
```

## Simulação de vetores aleatórios

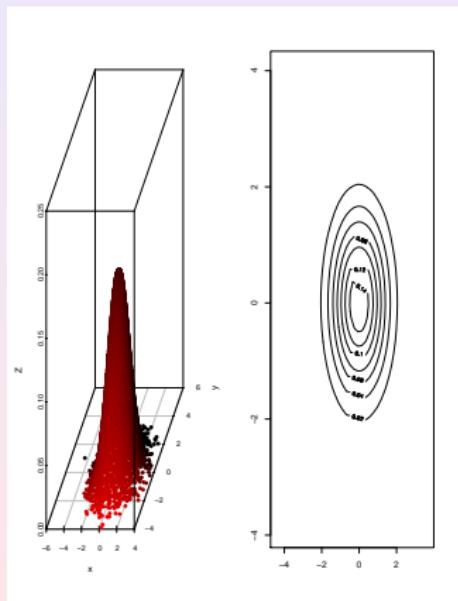
Na Figura 5 temos os histogramas das duas curvas juntamente com o diagrama de dispersão bidimensional obtidos gerando-se 10.000 valores cada uma dessas normais padrões independentes.



**Figura:** Simulação de duas normais independentes (nas margens) e gráfico de dispersão.

## Simulação de vetores aleatórios

Na Figura 6 temos a densidade bidimensional normal padrão e as respectivas curvas de nível.



**Figura:** Distribuição normal padrão bidimensional gerada e curvas de nível.

## Métodos de reamostragem

- Suponha que se queira simular uma amostra da densidade  $\pi$ , mas que isso não seja fácil. O que se pode fazer é proceder em duas etapas.
- Suponha que haja uma densidade  $g$ , da qual seja fácil gerar valores e que esteja próxima de  $\pi$ . Então:
  - (a) na primeira etapa simulamos uma amostra de  $g$ ;
  - (b) na segunda etapa, exercemos um mecanismo de correção, de modo que a amostra de  $g$  seja “direcionada” para tornar-se uma amostra de  $\pi$ .
- Em geral o que se faz é, ao simular um valor de  $g$ , este é aceito com certa probabilidade  $p$  e escolhendo-se  $p$  adequadamente podemos assegurar que o valor aceito seja um valor de  $\pi$ .

## Aceitação – rejeição

- Nesta situação, supomos que existe uma constante finita conhecida  $A$ , tal que  $\pi(x) \leq Ag(x)$ , para todo  $x$ . Ou seja,  $Ag$  serve como um envelope para  $\pi$  (Figura 7).
- Algoritmo:
  - [1]] Simule  $x^*$  de  $g(x)$ ;
  - [2]] simule  $u$  de uma distribuição  $\mathcal{U}(0, 1)$ , independentemente de  $x^*$ ;
  - [3]] se  $u \leq \pi(x^*)/Ag(x^*)$ , então aceite  $x^*$  como gerada de  $\pi(x)$ ; caso contrário, volte ao item 1.

## Aceitação – rejeição

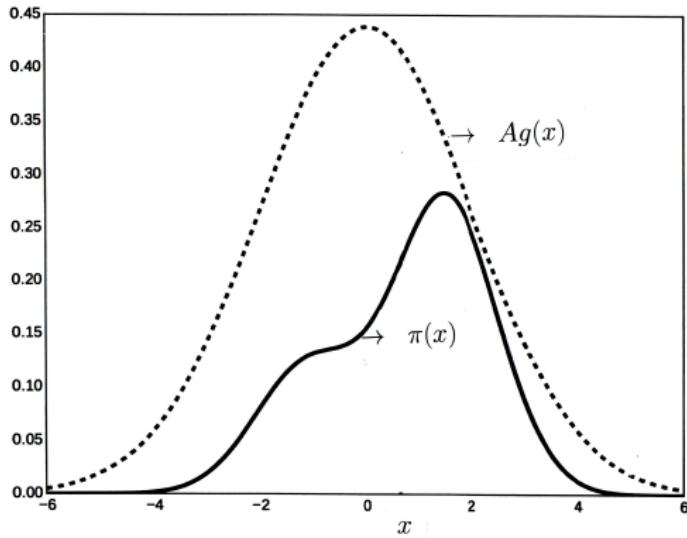


Figura: Densidades  $\pi$  (linha cheia) e  $Ag$  (linha pontilhada).

## Aceitação – rejeição

- Suponha que  $X \sim g(x)$ . Seja  $p(x)$  a probabilidade que  $x$  seja aceito; então,  $p(x) = \pi(x)/Ag(x)$ . Logo,

$$P(X \leq x \text{ e } X \text{ aceito}) = \int_{-\infty}^x p(y)g(y)dy$$

e

$$P(X \text{ aceito}) = \int_{-\infty}^{\infty} p(y)g(y)dy.$$

- Segue que

$$P(X \leq x \mid \text{aceito}) = \frac{\int_{-\infty}^x p(y)g(y)dy}{\int_{-\infty}^{\infty} p(y)g(y)dy} = \int_{\infty}^x \pi(y)dy,$$

logo os valores aceitos têm realmente distribuição  $\pi(x)$ .

- Também,

$$P(\text{aceitação}) = \int_{-\infty}^{\infty} p(y)g(y)dy = \frac{1}{A} \int_{-\infty}^{\infty} \pi(y)dy = \frac{1}{A}. \quad (13)$$

## Aceitação – rejeição

- Observe que  $\pi$  deve ser conhecida a menos de uma constante de proporcionalidade, ou seja, basta conhecer o que se chama o **núcleo** de  $\pi(x)$ .
- Devemos escolher  $g(x)$  de modo que ela seja facilmente simulável e de sorte que  $\pi(x) \approx Ag(x)$ , pois a chance de rejeição será menor. Também de (13), devemos ter  $A \approx 1$ .
- **Observações:** (a)  $0 < \pi(x^*)/Ag(x^*) \leq 1$ ;  
(b) O número de iterações,  $N$ , necessárias para gerar  $\pi$  é uma v.a. com distribuição geométrica, com probabilidade de sucesso

$$p = P(U \leq \pi(x^*)/Ag(x^*))P(N = n) = (1 - p)^{n-1}p, \quad n \geq 1.$$

Potanto, em média, o número de iterações necessárias é  $E(N) = 1/p$ .  
Como vimos acima,  $p = 1/A$ , logo  $E(N) = A$ . Logo, é desejável escolher  $g$  de modo que  $A = \sup_x \{\pi(x)/g(x)\}$ .

- (c) De (b), podemos dizer que o número esperado de iterações do algoritmo necessárias até que um valor de  $\pi$  seja gerado com sucesso é exatamente o valor da constante  $A$ .
- O método pode ser usado também para o caso de v.a.'s discretas.

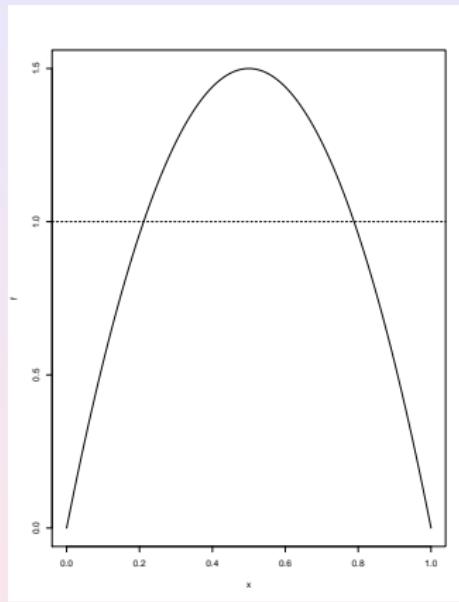
## Aceitação – rejeição

- **Exemplo 1.** Considere a densidade de uma Beta(2,2), ou seja,

$$\pi(x) = 6x(1-x), \quad 0 < x < 1.$$

- Suponha  $g(x) = 1$ ,  $0 < x < 1$ . O máximo de  $\pi(x)$  é 1,5, para  $x = 0,5$ .
- Logo, podemos tomar  $A = 1,5$  (o valor máximo de  $\pi(x)$ , para  $x = 0,5$ ), e teremos  $p = P(\text{aceitação}) = 1/A = 0,667$ .
- Portanto, para obter, por exemplo, uma amostra de tamanho 1.000 de  $\pi(x)$  deveremos simular em torno de 1.500 valores de uma uniforme padrão. Veja a Figura 8.

## Aceitação – rejeição



**Figura:** Densidades  $\pi$  (linha cheia) e  $g$  (linha tracejada) para o Exemplo 1.

## Aceitação – rejeição

Um algoritmo equivalente é o seguinte:

- 1) Simule  $x^*$  de  $g(x)$ ;
- 2) Simule  $y^*$  de  $\mathcal{U}(0, Ag(x^*))$ ;
- 3) Aceite  $x^*$  se  $y^* \leq \pi(x^*)$ ; caso contrário, volte a 1.

Usando o código R do capítulo, podemos obter as Figuras 9 e 10; a primeira mostra o histograma dos valores gerados (com a verdadeira curva adicionada) e a segunda mostra as regiões de aceitação e rejeição.

## Aceitação – rejeição

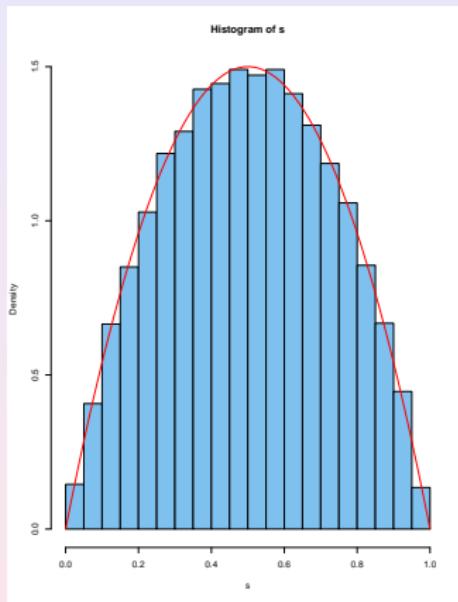
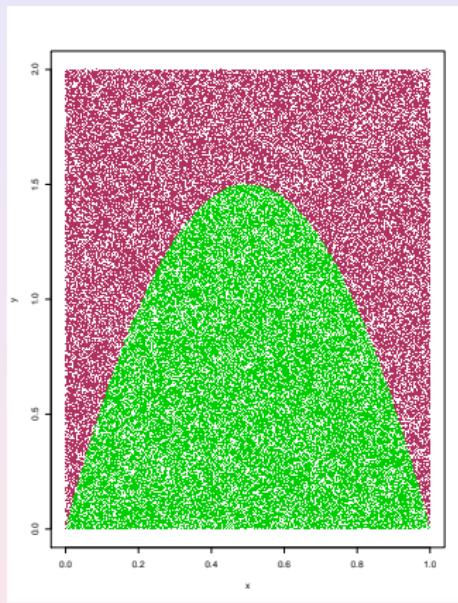


Figura: Histograma dos valores gerados e densidade (vermelho).

## Aceitação – rejeição



**Figura:** Região de aceitação (verde) e de rejeição (marrom).

# Reamostragem ponderada

Neste tipo de simulação temos essencialmente as duas etapas anteriores, mas  $Ag$  não precisa ser um envelope para  $\pi$ .

## Algoritmo:

- 1) Retire uma amostra  $x_1, \dots, x_n$  de  $g(x)$ ;
- 2) construa os pesos  $w_i$  dados por

$$w_i = \frac{\pi(x_i)/g(x_i)}{\sum_{j=1}^n \pi(x_j)/g(x_j)}, \quad i = 1, 2, \dots, n;$$

- 3) reamostre da distribuição de probabilidades discreta  $(x_i, w_i)$ ,  $i = 1, \dots, n$ .

## Reamostragem ponderada

- Então, a amostra resultante tem distribuição  $\pi$ . De fato, se  $x$  for um valor simulado pelo método,

$$F_x(a) = P(X \leq a) = \sum_{i:x_i \leq a} w_i = \frac{\sum_{i=1}^n (\pi(x_i)/g(x_i)) I_{\{x_i \leq a\}}}{\sum_{j=1}^n (\pi(x_j)/g(x_j))},$$

e o último termo converge, quando  $n \rightarrow \infty$  (lei forte dos grandes números), para

$$\frac{\int (\pi(x)/g(x)) I_{\{x \leq a\}} g(x) dx}{\int (\pi(x)/g(x)) dx} = \frac{\int \pi(x) I_{\{x \leq a\}} dx}{\int \pi(x) dx} = F_\pi(x).$$

- O método de reamostragem ponderada (**importance sampling**) é também usado para reduzir a variância da estimativa MC. Lembremos que o MMC consiste em aproximar  $E_F[h(X)]$  por

$$\hat{E}_F[h(X)] = \frac{1}{n} \sum_{i=1}^n h(X_i). \tag{14}$$

Suponha que em (14)  $F$  tenha densidade  $\pi$ . Então

$$\theta_\pi = E_\pi[h(X)] = \int h(x)\pi(x)dx = \int h(x)[\frac{\pi(x)}{g(x)}]g(x)dx. \tag{15}$$

## Reamostragem ponderada

- Se chamarmos  $\varphi(x) = h(x)\pi(x)/g(x)$ , teremos

$$\theta_\pi = \int \varphi(x)g(x)dx.$$

- Segue-se que, obtendo-se uma amostra  $x_1, \dots, x_n$  de  $g(x)$ , poderemos estimar (15) por

$$\hat{\theta}_\pi = \frac{1}{n} \sum_{i=1}^n \varphi(x_i) = \frac{1}{n} \sum_{i=1}^n w_i h(x_i), \quad (16)$$

onde  $w_i = \pi(x_i)/g(x_i)$ . Compare com o estimador MC de (14), que é não viesado, ao passo que (16) é viesado.

- Se quisermos um estimador não viesado, basta considerar

$$\theta_\pi^* = \frac{\sum_{i=1}^n w_i h(x_i)}{\sum_{i=1}^n w_i}. \quad (17)$$

- Note que o estimador dá mais peso a regiões onde  $g(x) < \pi(x)$ . Geweke (1989) provou que  $\theta_\pi^* \rightarrow \theta$ , com probabilidade um, se o suporte de  $g(x)$  inclui o suporte de  $\pi(x)$ ,  $X_i \sim \text{iid } g(x)$  e  $E[h(X)] < \infty$ . Mostrou, também, que o erro padrão da estimativa (17) é dado por

$$\frac{[\sum_{i=1}^n [h(x_i) - \theta_\pi^*]^2 w_i^2]^{1/2}}{\sum_{i=1}^n w_i}.$$

## Reamostragem ponderada

- **Exemplo 2.** Num modelo genético, 197 animais distribuem-se em quatro classes  $\mathbf{X} = (x_1, x_2, x_3, x_4)' = (125, 18, 20, 34)'$ , segundo as probabilidades  $(\theta + 2)/4, (1 - \theta)/4, (1 - \theta)/4, \theta/4$ , e o objetivo é estimar  $\theta$ .
- A verossimilhança é

$$L(\theta|\mathbf{X}) \propto (2 + \theta)^{125} (1 - \theta)^{38} \theta^{34}.$$

- Supondo uma priori constante , a densidade a posteriori é

$$p(\theta|\mathbf{X}) \propto (2 + \theta)^{125} (1 - \theta)^{38} \theta^{34}.$$

- Um estimador de  $\theta$  é a média dessa densidade a posteriori, que é estimada por (17). Aqui, suponha que  $g(\theta) \propto \theta^{34}(1 - \theta)^{38}$ ,  $0 < \theta < 1$ , ou seja, temos uma Beta (35,39).

## Reamostragem ponderada

- Portanto,

$$\hat{\theta} = \frac{\sum_{i=1}^n w_i \theta_i}{\sum_{i=1}^n w_i},$$

onde  $\theta_1, \dots, \theta_n$  é uma amostra de  $g(x)$  e os pesos  $w_i$  são dados por

$$w_i = \frac{(2 + \theta_i)^{125}}{\sum_{j=1}^n (2 + \theta_j)^{125}}, \quad i = 1, \dots, n.$$

Gerando-se 10.000 valores de uma Beta(35,39) obtivemos  $\hat{\theta} = 0,6180$ .

## Referências

- Box, G.E.P. and Müller, M.E.(1958). A note on the generation of random normal deviates. *The Annals of Statistics*, **29**, 610–611.
- Hammersley, J.M. and Handscomb, D.C. (1964). *Monte Carlo Methods*. New York: Wiley.
- Kleijnen, J. and Groenendall, W. (1994). *Simulation: A Statistical Perspective*. Chichester: Wiley.
- Ross, S.(1997). *Simulation*, 2nd Ed., New York: Academic Press.
- Sobol, I.M.(1976). *Método de Monte Carlo*. Moscow: Editorial MIR.
- von Neumann, J.(1951). Various techniques used in connection with random digits, Monte Carlo Method. *U.S. National Bureau of Standards Applied Mathematica Series*, **12**, 36–38.