INSTITUTO FEDERAL
Santa Catarina
Câmpus São José

Aplicação de Técnicas de Aprendizagem de Máquina utilizando R

Prof. Mário de Noronha Neto

INSTITUTO FEDERAL
Santa Catarina
Câmpus
São José

O material utilizado neste curso foi elaborado pelos professores Mario de Noronha Neto (IFSC) e Richard Demos Souza (UFSC)

Classificação utilizando o *Nearest Neighbors* (k-NN)

O classificador *nearest neighbor* é definido por sua característica de classificar exemplos não rotulados, atribuindo-lhes a classe de exemplos rotulados semelhantes. Utilizações frequentes:

Em geral, são adequados para tarefas de classificação em que as relações entre as características e as classes são numerosas, complicadas, ou extremamente difícil de entender, mas os itens das classes semelhantes tendem a ser bastante homogênos. Outra maneira de dizer isso seria dizer que, se um conceito é difícil de definir, mas você consegue identicar quando o vê, o k-NN pode ser apropriado. Por outro lado, se os dados são ruidosos e, portanto, não existe distinção clara entre os grupos, está técnica pode ter dificuldades para identificar os limites das classes. Alguns exemplos de utilização:

- Aplicações de visão computacional
- Identificação de padrões em dados genéticos, detectando proteínas específicas ou doenças

Fonte: Brett Lantz – Machine Learning with R – Second Edition

INSTITUTO FEDERAL
Santa Catarina
Câmpus São José

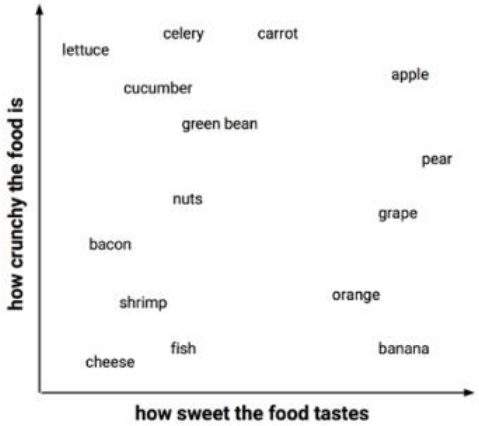
Pontos fortes e fracos do k-NN

Strengths	Weaknesses
• Simple and effective • Makes no assumptions about the underlying data distribution • Fast training phase	• Does not produce a model, limiting the ability to understand how the features are related to the class • Requires selection of an appropriate *k* • Slow classification phase • Nominal features and missing data require additional processing

Fonte: Brett Lantz – Machine Learning with R – Second Edition

Funcionamento do k-NN

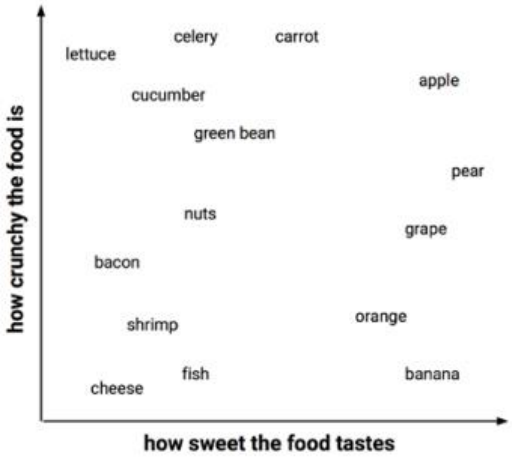
Ingredient	Sweetness	Crunchiness	Food type
apple	10	9	fruit
bacon	1	4	protein
banana	10	1	fruit
carrot	7	10	vegetable
celery	3	10	vegetable
cheese	1	1	protein

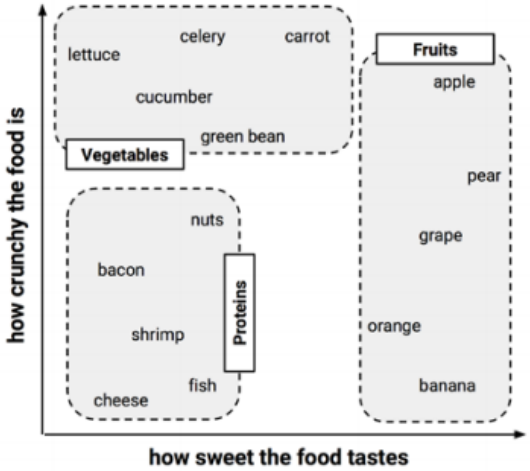


Fonte: Brett Lantz – Machine Learning with R – Second Edition

INSTITUTO FEDERAL
Santa Catarina
Câmpus
São José

Funcionamento do k-NN

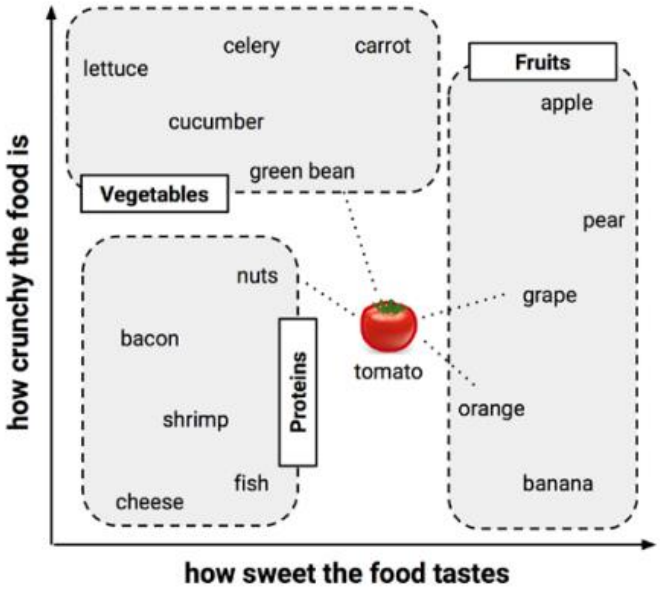




Fonte: Brett Lantz – Machine Learning with R – Second Edition

Funcionamento do k-NN

O tomate é uma fruta ou um vegetal?



Fonte: Brett Lantz – Machine Learning with R – Second Edition

Medidas de similaridade com distância

Distância Euclideana:

$$\text{dist}(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + ... + (p_n - q_n)^2}$$

Distância entre o tomate e o feijão verde:

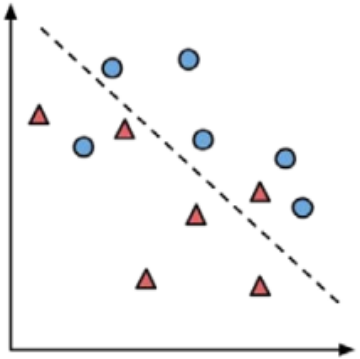
$$\text{dist}(\text{tomato, green bean}) = \sqrt{(6-3)^2 + (4-7)^2} = 4.2$$

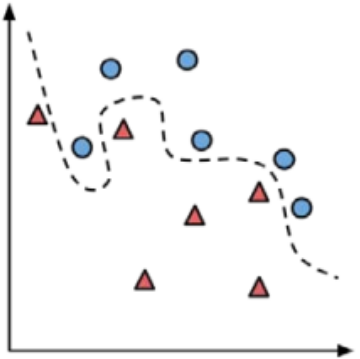
Ingredient	Sweetness	Crunchiness	Food type	Distance to the tomato
grape	8	5	fruit	*sqrt((6 - 8)^2 + (4 - 5)^2) = 2.2*
green bean	3	7	vegetable	*sqrt((6 - 3)^2 + (4 - 7)^2) = 4.2*
nuts	3	6	protein	*sqrt((6 - 3)^2 + (4 - 6)^2) = 3.6*
orange	7	3	fruit	*sqrt((6 - 7)^2 + (4 - 3)^2) = 1.4*

Para k = 1, a laranja é o vizinho mais próximo do tomate. Neste caso, como a laranja é uma fruta, o tomate será classificado como uma fruta. Para k = 3, os três vizinhos mais próximos são a laranja, a uva as nozes. Neste caso, por dois votos a um, o tomate novamente será classificado como uma fruta

Fonte: Brett Lantz – Machine Learning with R – Second Edition

Escolhendo valores adequados para ‘k’





Valores elevados de ‘k’ reduz o impacto causado por ruído nos dados, mas pode deixar o aprendizado tendencioso de forma a ignorar padrões menores, mas importantes para o processo de classificação

Valores muito baixos de ‘k podem permitir que um ruído ou um *outlier* influenciem na classificação

Uma regra frequentemente utilizada para a escolha inicial de k é o inteiro mais próximo da raiz quadrada do número de dados para o treinamento.

Fonte: Brett Lantz – Machine Learning with R – Second Edition

Preparando os dados para uso no k-NN

Normalização

$$X_{new} = \frac{X - \min(X)}{\max(X) - \min(X)}$$

dummy coding para variáveis nominais

$$\text{male} = \begin{cases} 1 & \text{if x = male} \\ 0 & \text{otherwise} \end{cases}$$

$$\text{hot} = \begin{cases} 1 & \text{if x = hot} \\ 0 & \text{otherwise} \end{cases}$$

$$\text{medium} = \begin{cases} 1 & \text{if x = medium} \\ 0 & \text{otherwise} \end{cases}$$

Fonte: Brett Lantz – Machine Learning with R – Second Edition

Considerações sobre o k-NN

- No processo de aprendizagem o algoritmo não aprende, apenas armazena os dados de forma estruturada. Isto permite um treinamento rápido, porém o processo de predições tente a ser mais lento em comparação ao treinamento.

- A forma de aprendizado do k-NN também é conhecida como aprendizagem baseado em instâncias ou aprendizagem por repetição

- Métodos baseados em instâncias não constroem um modelo. Desta forma, nenhum parâmetro é aprendido sobre os dados. Por outro lado, este tipo de método permite encontrar padrões naturais nos dados

Fonte: Brett Lantz – Machine Learning with R – Second Edition

Exemplo: Diagnóstico de Câncer de Mama

Passo 1: Coleta de dados

Dataset utilizado: http://archive.ics.uci.edu/ml

Características:

- 569 exemplos de biópsias, cada uma com 32 características
- As características são o número de identificação, o diagnóstico de câncer e outros 30 valores numéricos de referência contendo a média, erro padrão e o valor do pior caso para as características do núcleo da célula apresentadas no quadro ao lado.
- O diagnóstico é codificado como ‘M’ para maligno e ‘B’ para benigno

- Radius
- Texture
- Perimeter
- Area
- Smoothness
- Compactness
- Concavity
- Concave points
- Symmetry
- Fractal dimension

Fonte: Brett Lantz – Machine Learning with R – Second Edition

INSTITUTO FEDERAL
Santa Catarina
Câmpus
São José

Passo 2: Explorando e preparando os dados

```
# import the CSV file
wbcd <- read.csv("wisc_bc_data.csv", stringsAsFactors = FALSE)

# examine the structure of the wbcd data frame
str(wbcd)

'data.frame':  569 obs. of  32 variables:
 $ id                : int  87139402 8910251 905520 ...
 $ diagnosis         : chr  "B" "B" "B" "B" ...
 $ radius_mean       : num  12.3 10.6 11 11.3 15.2 ...
 $ texture_mean      : num  12.4 18.9 16.8 13.4 13.2 ...
 $ perimeter_mean    : num  78.8 69.3 70.9 73 97.7 ...
 $ area_mean         : num  464 346 373 385 712 ...

# drop the id feature
wbcd <- wbcd[-1]
```

Fonte: Brett Lantz – Machine Learning with R – Second Edition

Passo 2: Explorando e preparando os dados

```
# table of diagnosis
table(wbcd$diagnosis)
```

```
  B   M
357 212
```

```
# recode diagnosis as a factor
wbcd$diagnosis <- factor(wbcd$diagnosis, levels = c("B", "M"),
                         labels = c("Benign", "Malignant"))
```

```
# table or proportions with more informative labels
round(prop.table(table(wbcd$diagnosis)) * 100, digits = 1)
```

```
   Benign Malignant
     62.7      37.3
```

Fonte: Brett Lantz – Machine Learning with R – Second Edition

INSTITUTO FEDERAL
Santa Catarina
Câmpus São José

Passo 2: Explorando e preparando os dados

```
# summarize three numeric features
summary(wbcd[c("radius_mean", "area_mean", "smoothness_mean")])
```

```
  radius_mean      area_mean      smoothness_mean
 Min.   : 6.981   Min.   : 143.5   Min.   :0.05263
 1st Qu.:11.700   1st Qu.: 420.3   1st Qu.:0.08637
 Median :13.370   Median : 551.1   Median :0.09587
 Mean   :14.127   Mean   : 654.9   Mean   :0.09636
 3rd Qu.:15.780   3rd Qu.: 782.7   3rd Qu.:0.10530
 Max.   :28.110   Max.   :2501.0   Max.   :0.16340
```

Observe que as escalas das variáveis *area_mean* ([143.5 2501.0]) e *smoothness_mean* ([0.05 0.16]) são bem distantes. Como o cálculo da distância no k-NN depende da escala das características de entrada, isto pode causar problemas no classificador

Fonte: Brett Lantz – Machine Learning with R – Second Edition

INSTITUTO FEDERAL
Santa Catarina
Câmpus São José

Passo 2: Explorando e preparando os dados

Normalização numérica: Para normalizar estas características, criaremos uma função *normalize()* no R

```
# create normalization function
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}
```

```
# test normalization function - result should be identical
normalize(c(1, 2, 3, 4, 5))
normalize(c(10, 20, 30, 40, 50))
```

```
> normalize(c(1, 2, 3, 4, 5))
[1] 0.00 0.25 0.50 0.75 1.00
> normalize(c(10, 20, 30, 40, 50))
[1] 0.00 0.25 0.50 0.75 1.00
```

Fonte: Brett Lantz – Machine Learning with R – Second Edition

Passo 2: Explorando e preparando os dados

A função *lapply( )* aplica uma determinada função para cada elemento de uma lista. Com um *data frame* é uma lista de vetores do mesmo comprimento, utilizaremos a função *lapply ( )* para a normalização das 30 características numéricas do *data set*. O passo final é converter a lista retornada pela função *lapply ( )* para um *data frame* utilizando a função *as.data.frame ( ).*

```
# normalize the wbcd data
wbcd_n <- as.data.frame(lapply(wbcd[2:31], normalize))

# confirm that normalization worked
summary(wbcd_n$area_mean)

Min. 1st Qu.Median    Mean 3rd Qu.    Max.
0.0000  0.1174  0.1729  0.2169  0.2711  1.0000
```

Fonte: Brett Lantz – Machine Learning with R – Second Edition

INSTITUTO FEDERAL
Santa Catarina
Câmpus São José

Passo 2: Explorando e preparando os dados

Criando conjuntos de dados para treinamento e teste: Uma regra frequentemente utilizada é separar em torno de 80% do conjunto de dados para realização do treinamento e em torno de 20% para realização de testes para medir o desempenho do modelo gerado. Neste caso, utilizaremos os 469 primeiros para treinamento e os 100 restantes para testes. Caso os dados tenham sido coletados com algum tipo de ordenação, é importante coletar os dados de treinamento e teste de forma aleatória. É importante que cada conjunto de dados seja representativo.

```
# create training and test data
wbcd_train <- wbcd_n[1:469, ]
wbcd_test <- wbcd_n[470:569, ]
```

```
# create labels for training and test data
wbcd_train_labels <- wbcd[1:469, 1]
wbcd_test_labels <- wbcd[470:569, 1]
```

Fonte: Brett Lantz – Machine Learning with R – Second Edition

Passo 3: “Treinando o modelo”

O processo de classificação utilizando o k-NN não envolve a construção de um modelo. O processo simplesmente armazena os dados de entrada em um formato estruturado. Para implementar o k-NN será utilizado o pacote *class*

kNN classification syntax

using the `knn()` function in the `class` package

Building the classifier and making predictions:

```
p <- knn(train, test, class, k)
```

- `train` is a data frame containing numeric training data
- `test` is a data frame containing numeric test data
- `class` is a factor vector with the class for each row in the training data
- `k` is an integer indicating the number of nearest neighbors

The function returns a factor vector of predicted classes for each row in the test data frame.

Example:

```
wbcd_pred <- knn(train = wbcd_train, test = wbcd_test,
                 cl = wbcd_train_labels, k = 3)
```

k = 21. Tentativa para ‘sqrt (469)’. Número ímpar evita a possibilidade de ter um voto minerva em caso de empate

```
# load the "class" library
library(class)

wbcd_test_pred <- knn(train = wbcd_train, test = wbcd_test,
                      cl = wbcd_train_labels, k = 21)
```

Fonte: Brett Lantz – Machine Learning with R – Second Edition

Passo 4: Avaliando o desempenho do modelo

Utilizar a função *CrossTable( )* do pacote *gmodels* para avaliar o modelo

```
# load the "gmodels" library
library(gmodels)

# Create the cross tabulation of predicted vs. actual
CrossTable(x = wbcd_test_labels, y = wbcd_test_pred,
           prop.chisq = FALSE)
```

wbcd_test_labels	wbcd_test_pred: Benign	Malignant	Row Total
Benign	61 1.000 0.968 0.610	0 0.000 0.000 0.000	61 0.610
Malignant	2 0.051 0.032 0.020	37 0.949 1.000 0.370	39 0.390
Column Total	63 0.630	37 0.370	100

Verdadeiro negativo

Falso positivo

Falso negativo

Verdadeiro positivo

Taxa de sucesso = 98 %

Fonte: Brett Lantz – Machine Learning with R – Second Edition