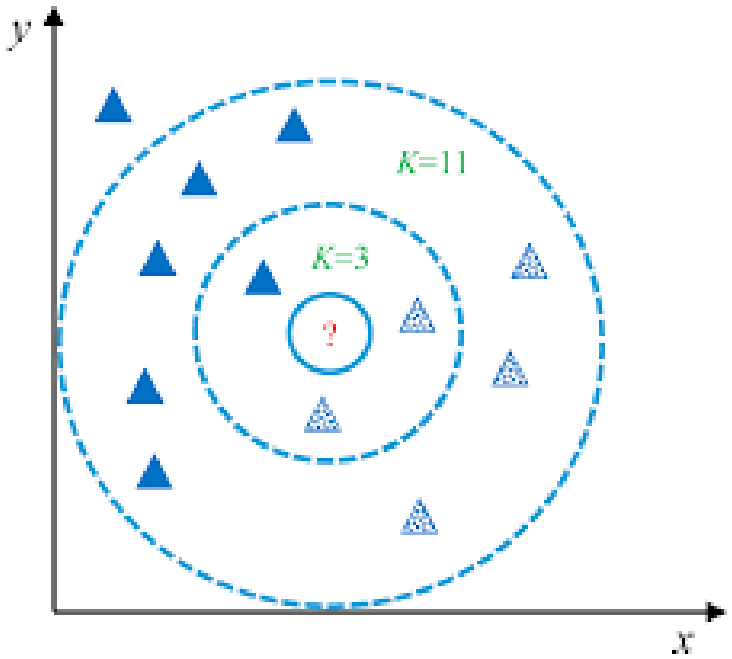
Curso de Estatística Multivariada
Método kNN de Classificação



Brasília - DF
Dezembro de 2018

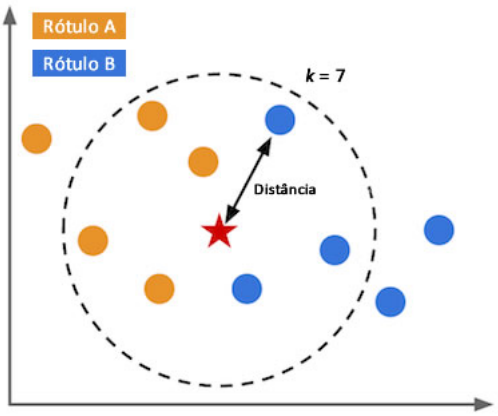
Sumário

Lista I - kNN

Algoritmo kNN

O classificador k vizinhos mais próximos (do ingês: k nearest neighboors kNN) foi proposto por Fukunaga e Narendra em 1975. É um dos classificadores mais simples de ser implementado, de fácil compreensão e ainda hoje pode obter bons resultados dependendo de sua aplicação.

A ideia principal do kNN é determinar o rótulo de classificação de uma amostra baseado nas amostras vizinhas advindas de um conjunto de treinamento. Nada melhor do que um exemplo para explicar o fucionamento do algoritmo como o da figura abaixo, na qual temos um problema de classificação com dois rótulos de classe e com k = 7. No exemplo, são aferidas as distâncias de uma nova amostra, representada por uma estrela, às demais amostras de treinamento, representadas pelas bolinhas azuis e amarelas. A variável k representa a quantidade de vizinhos mais próximos que serão utilizados para averiguar de qual classe a nova amostra pertence. Com isso, das sete amostras de treinamento mais próximas da nova amostra, 4 são do rótulo A e 3 do rótulo B. Portanto, como existem mais vizinhos do rótulo A, a nova amostra receberá o mesmo rótulo deles, ou seja, A. Dois pontos chaves que devem ser determinados para



aplicação do kNN são: a métrica de distância e o valor de k. Para métrica de distância a mais utilizada é a distância Euclidiana, descrita por:

$$D = \sqrt{(p_1 - q_1)^2 + \cdots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

onde $P = (p_1, \cdots, p_n)$ e $Q = (q_1, \cdots, q_n)$ são dois pontos n -dimensionais. No exemplo acima, essa distância seria calculada entre as bolinhas (azuis e laranjas) e a estrela (a nova entrada). Como o exemplo é 2D, cada uma cada ponto teria seu valor em x e em y. Para problemas com dimensões maiores a abordagem é a exatamente a mesma.

Em relação ao valor k, não existe um valor único para a constante, a mesma varia de acordo com a base de dados. É recomendável sempre utilizar valores ímpares/primos, mas o valor ótimo varia de base para base. Todavia, você pode deixar o desempenho geral do modelo bem lento na etapa de seleção de k. Uma sugestão é utilizar $k = \sqrt{n}$, com n igual ao número de elementos da base de treinamento. Outra maneira é simplesmente testar um conjunto de valores e encontrar k empiricamente.

Resumidamente, a grande vantagem do kNN é sua abordagem simples de ser compreendida e implementada. Todavia, calcular distância é tarefa custosa e caso o problema possua grande número de amostras o algoritmo pode consumir muito tempo computacional. Além disso, o método é sensível à escolha do k. Na sequência é apresentado um pseudocódigo do algoritmo:

1. **inicialização**
   (a) Preparar o conjunto de dados de treinamento e de teste.
   (b) informar o valor de k.
2. **para cada nova amostra faça**
   (a) Calcular distância para todas as amostras
   (b) Determinar o conjunto das k's distâncias mais próximas
   (c) O rótulo com mais representantes no conjunto dos k's vizinhos será escolhido.
3. **fim**
4. **retornar**: conjunto de rótulos de classificação.

Exercícios

Questão 1 - Calcule a distância ente os pontos $(0, 0, 1)$ e $(1, 1, 0)$.

Questão 2 - Escreva o código em R que calcule a distância ente os pontos $(0.70, 0.40)$ e $(0.40, 0.50)$, que é 0.32.

Questão 3 - A distância e o rotulo de observações de uma base de treinamento em relação a um exemplar A são representadas pelos pares ordenados: $(0.32, @)$, $(0.09, \#)$, $(0.36.\#)$, $(0.14, A)$, $(0.54, \#)$, $(0.61, @)$ e $(0.22, @)$. Qual o rótulo deve ser atribuído a A para k=3?

Implementando o kNN no R

O pacote `class` disponibiliza a função `knn()` que aplica o algoritmo kNN em um conjuto de dados. Um resumo dos parâmetros que essa função recebe são apresentados no quadro abaixo.

`y_estimado <- knn(treino,teste,rotulo,k)`

i) `treino` é um data frame que contém os dados numéricos com os atributos descritivos usados como conjuto de treinamento.

ii) `teste` é um data frame que contém dados numéricos com atributos descritivos para a realização do teste.

iii) `rotulo` é um vetor com as classes para cada exemplar do `treino`.

iv) `k` é uma variável de tipo inteiro que indica o número de vizinhos mais próximos a serem consultados.

v) `y_estimado` é uma variável que recebe a saída da função - a classe atribuída as observações em `teste`.

Observe que a funçãoo `knn()` exige que o conjunto de atributos descritivos esteja armazenado em uma estrutura de dados distinta daquela na qual está armazenado o conjunto de rótulos correspondente a cada cada observação do conjunto de dados do treinamento. Assim, para utilizar a função `knn()` no data frame iris é necessário informar a função que os os atributos descritivos estão nas 4 primeiras colunas e os rótulos estão na 5 coluna.

```
## Um exemplo de knn com o Iris data set data(iris)
## dividindo a base em treino e test
idxs <- sample(1:nrow(iris),as.integer(0.7*nrow(iris)))
treino <- iris[idxs,]
teste <- iris[-idxs,]
## Analizando s três vizinhos mais proximos
nn3 <- knn(treino[,-5],teste[,-5],treino[,5],k=3)
## Matri de Confusão
table(teste[,'Species'],nn3)
```

Exercícios

Questão 1 - Explique a linha de comando:

`idxs <-sample(1:nrow(iris),as.integer(0.7*nrow(iris))).`

Questão 2 - Qual a ação desempenhada pelos comandos: `treino[,5]`, `iris[-idxs,]`?

Estudo de caso: Deteção de câncer de próstata

O aprendizado automático da máquina encontra uso extensivo na indústria farmacêutica, por exemplo, a detecção de crescimento oncogênico (células cancerosas). O programa R pode ser utilizado na aprendizagem de máquinas para construir modelos para prever o crescimento anormal de células, ajudando assim na detecção de câncer e beneficiando o sistema de saúde.

O processo de construção deste modelo usando o algoritmo kNN por meio do programa R é apresentado a seguir.

Etapa 1- Coleta de dados

A base de dados analisada é formada por informaçoes de 100 pacientes obtidas em um Digital Rectal Exame - DRE. São observadas 10 variáveis, das quais 8 são variáveis numéricas, uma é categórica e uma varável identificadora. Essas variáveis são:

1. id (identificação do paciente)
2. diagnosis_result (resultado do diagnóstico)
3. Radius (raio)
4. Texture (textura)
5. Perimeter (perimetro)
6. Area (área)
7. Smoothness (suavidade)
8. Compactness (compacidade)
9. Symmetry (simetria)
10. Fractal dimension (dimensão do fractal)

São necessários dezenas de parâmetros importantes para medir a probabilidade de crescimento canceroso, mas, para fins didáticos, serão utilizados 8 deles.

Etapa 2 - Preparando e explorando os dados

Os comandos abaixos fazem a leitura dos dados, apresentam as variáveis e as primeiras 5 observações.

```
# O arq. Prostate_Cancer.csv deve estar
# na sua area de trabalho setwd()
prc <- read.csv("Prostate_Cancer.csv", stringsAsFactors = F)
str(prc)
head(prc)
```

Se observarmos o conjunto de dados, a primeira variável `id` é de natureza única e pode ser removida porque não fornece informações úteis.

```
prc <- prc [-1]
```

O conjunto de dados contém pacientes com diagnóstico de câncer maligno (M) ou benigno (B), nas seguintes quantidades

```
tabela (prc$diagnosis_result)
```

A variável `diagnosis_result` é a nossa variável alvo, ou seja, esta variável determinará os resultados do diagnóstico com base nas 8 variáveis numéricas. Caso desejemos renomear B como "Benigno"e M como "Maligno"e ver os resultados na forma de porcentagem, podemos escrever como:

```
prc$diagnostic <- factor (prc$diagnosisresult,
                  levels = c ("B", "M"),
                  labels = c ("Benigno", "Maligno"))
round (prop.table (table (prc$diagnosis)) * 100, digits = 1)
```

Normalizando os dados numéricos

Nesta analise a normalização dos dados numéricos é aconselhada, uma vez que a escala utilizada para os valores para cada variável pode ser diferente. Assim,todos os dados são tranformados para uma escala comum.

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

Uma vez criada a funçao que normaliza os dados, o código seguinte muda a escala dos 8 atributos descritivos:

```
prc_n <- as.data.frame (lapply (prc [2: 9], normalize))
```

Para verificar a ação da normalização sobre os valores da variável `radius` execute o comando

```
summary(prc_n$radius)
```

Criando conjuntos de treinamento e teste de dados

O algoritmo kNN é aplicado ao conjunto de dados de treinamento e os resultados são verificados no conjunto de dados do teste.

Para isso, dividimos o conjunto de dados em 2 porções na proporção de 65: 35 (assumido) para o conjunto de treinamento e dados de teste, respectivamente. Você pode usar uma proporção diferente.

O data frame `prc_n` foi divido em `prc_train` e `prc_test`

```
prc_train <- prc_n [1:65,]
prc_test <- prc_n [66:100,]
```

Um valor em branco em cada uma das instruções acima indica que todas as linhas e colunas devem ser incluídas. A variável alvo é `diagnosis_result` que não é incluída nos conjuntos de treinamento e de teste.

```
prc_train_labels <- prc [1:65, 1]
prc_test_labels <- prc [66:100, 1]
```

Etapa 3 - Treinar um modelo de dados

A função knn () precisa ser usada para treinar um modelo para o qual precisamos instalar o pacote `class`. A função knn () identifica os vizinhos k mais próximos usando a distância euclidiana, onde k é um número especificado pelo usuário. Você precisa digitar os seguintes comandos para usar knn ()

```
install.packages ("class")
library (classe)
```

O código abaixo usa a função knn () para classificar os dados do teste

```
prc_test_pred <- knn (train = prc_train, test = prc_test,
                      cl = prc_train_labels, k = 10)
```

O valor para k é geralmente escolhido como a raiz quadrada do número de observações. `knn ()` retorna um `fator` com os rótulos previstos para cada observação da base validação que são atribuídas a `prc_test_pred`

Etapa 4 - Avalie o desempenho do modelo

Com objetivo de verificar a precisão dos valores previstos em `prc_test_pred`,isto é, o número de coincidências com os valores conhecidos em (prc_test_labels). Isso pode ser realizado por meio da função `CrossTable()` disponível no pacote `gmodels`, como a seguir.

```
install.packages ("gmodels")
CrossTable(x=prc_test_labels, y=prc_test_pred, prop.chisq=F)
```

```
   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|

Total Observations in Table:  35

                | prc_test_pred
prc_test_labels |         B |         M | Row Total |
----------------|-----------|-----------|-----------|
              B |         7 |        12 |        19 |
                |     0.368 |     0.632 |     0.543 |
                |     0.875 |     0.444 |           |
                |     0.200 |     0.343 |           |
----------------|-----------|-----------|-----------|
              M |         1 |        15 |        16 |
                |     0.062 |     0.938 |     0.457 |
                |     0.125 |     0.556 |           |
                |     0.029 |     0.429 |           |
----------------|-----------|-----------|-----------|
   Column Total |         8 |        27 |        35 |
                |     0.229 |     0.771 |           |
----------------|-----------|-----------|-----------|
```

Os dados de teste consiste de 35 observações. Dos quais 7 casos foram precisamente previstos (TN-> Verdadeiros Negativos) como benignos (B), que constituem 20,0%. Além disso, 15 de 35 observações foram preditas com precisão (TP-> Verdadeiros Positivos) como malignas (M), o que representa 42,9%. Assim, um total de 15 de 35 previsões onde TP, isto é, Verdadeiro Positivo na natureza.

Houve 1 caso de falsos negativos (FN), o que significa 1 paciente doente não foi identificado. O FN representa uma ameaça potencial por isso um dos objetivos para aumentar a precisão do modelo é reduzir

os FN's.

Há 12 casos de Positivos Falsos (FP), significando que 12 casos eram de natureza benigna, mas foram preditos como malignos.

A precisão total do modelo (**acurácia**) é de 60% ((TN + TP) / 35), o que mostra que pode haver chances de melhorar o desempenho do modelo

Melhorar o desempenho do modelo

Isso pode ser levado em consideração repetindo as etapas 3 e 4 e alterando o valor `k`. Geralmente, é a raiz quadrada das observações e, neste caso, tomamos `k = 10`, que é uma raiz quadrada perfeita de 100. O valor k pode flutuar em torno do valor de 10 para obter maior precisão do modelo .

Exercícios

Questão 1 - Na área de saúde o objetivo é buscar o valor de FN (falso negativo) o mais baixo possível. Por que?

Questão 2 - Qual o efeito da função abaixo sobre um vetor de dados x?

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

Questão 3 - Qual crítica pode ser feita a forma de selecionar a base de teste e treinamento abaixo?

```
prc_train <- prc_n [1:65,]
prc_test <- prc_n [66:100,]
```

Questão 4 - Defina o termo matriz de confusão?

Questão 5 - Qual a acurácia observada quando se aplica o método `Knn` a base de dados `iris`, dado que a base treinamento e a de validação é o próprio data frame `iris` e o número de vizinhos a considerá é `k=10`?

Atividade I

Questão 1 - O arquivo **Feijao.RData** contém o data frame `feijao` que tém a variável peso apenas. Os valores armazenados são as massas, em gramas, de 140 grãos de certo tipo de feijão.

Execute as seguintes ações:

i- Faça a pasta que contém o arquivo **Feijão.RData** ser a área de trabalho.(`setwd()`)

ii- Use a função `load()` para carregar o arquivo **Feijao.RData**.

iii- Faça uma análise descritiva da variável `peso`.

iv- Crie a variável `classe` no data frame `feijao`, que armazene o valor **A** para os grãos 25% mais pesados e **B** para os demais. (`quantile()` e `ifelse()`)

v- Aplique o método kNN para classificar um grão de em função do seu peso em uma das categorias **A** ou **B**. A base de treino deve ter 40% das observações.(`sample()` e `knn()`)

Questão 2 - O arquivo **nodulos.csv** traz a idade, quantidade de nódulos e se tem câncer ou não (1 - não , 2 - sim) de 306 mulheres, as rescpetivas variáveis são: `idade`, `nodulo` e `status`.

Execute as seguintes ações:

i- Faça a pasta que contém o arquivo **nodulos.csv** ser a área de trabalho.(`setwd()`)

ii- Use a função `read.csv2()` para criar o data frame `dados` a partir do arquivo **nodulo.csv**.

iii- Faça uma análise descritiva de todas as variáveis. (`summary()`, `table()` e `boxplot()`).

iv- Aplique o método kNN para classificar uma paciente nas categorias **1** ou **2**. A base de trinamento deve ter 80% das observações.(`sample()` e `knn()`)

Questão 3 - O arquivo **gravidez.csv** traz a duracao da gestação de 1000 mulheres, em dias.

Execute as seguintes ações:

i- Faça a pasta que contém o arquivo **gravidez.csv** ser a área de trabalho.(`setwd()`)

ii- Use a função `read.csv2()` para criar o data frame `dados` a partir do arquivo **gravidez.csv**.

iii- Calcule a média e a variância da variável `duracao`.

iv- Crie as seguintes variáveis no data frame `dados`, onde x é a variável `duracao`.

a) $X1 = \dfrac{x - min(x)}{max(x) - min(x)}$

b) $X2 = \dfrac{x - mean(x)}{max(x) - min(x)}$

c) $X3 = \dfrac{x - mean(x)}{sd(x)}$

v- Calcule a média e a variância das variáveis X1, X2 e X3.