# 最简单的机器学习方法-K最近邻

KNN(K-NEAREST NEIGHBOR)

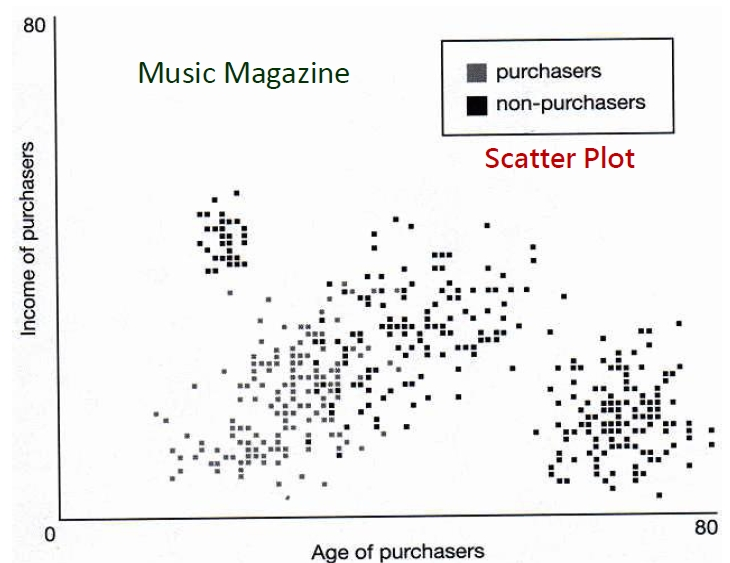
李御玺([leeys@mail.mcu.edu.tw](mailto:leeys@mail.mcu.edu.tw))

## K最近邻简介

在预测型数据挖掘中可以用K最近算法来进行分类和数字预测.

第一个例子是用来进行分类预测的例子, 预测某一个客户会不会购买某本音乐杂志. 要做K最近进行预测, 必须要有输入属性和目标属性. 这里的目标属性为客户的Age和Income, 将这两个属性输入之后, 一个点就代表一个客户. 浅灰色为购买杂志的用户, 黑色为未购买杂志的用户. 使用散点图把购买的情况显示出来. 从图中可以看到, 购买杂志的用户大多集中在左下区域.

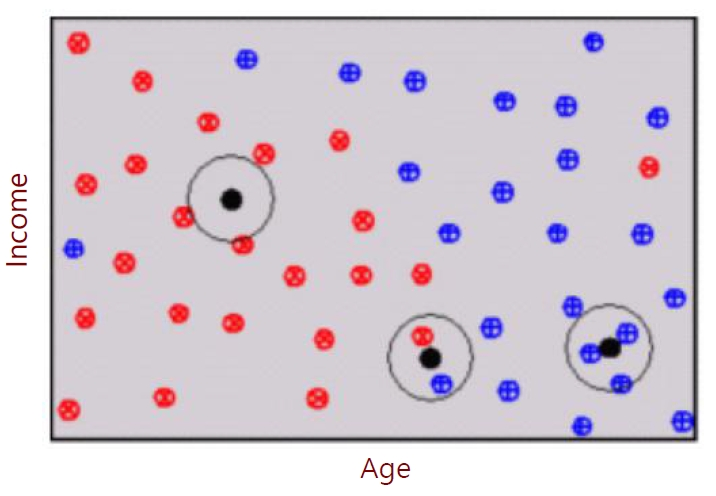
K最近方法的预测原理是, 找到与要预测的对象输入属性最接近的K个人, 用这K个人的行为来预测目标对象的行为.



对上面进行简化, 红色点代表购买杂志的客户, 蓝色点代表没有购买杂志的客户.

如果有一个新客户, 要预测他会不会购买杂志, 就计算他的代表点与已知购买及未购买的客户的距离, 找出与此客户距离最短的K个人, 即在输入属性上与这个要进行预测的新客户最相似的K个人. 例如K为3时, 找到的与预测客户距离最近的3个人如图所示. 最左边的客户, 由于与他距离最近的3个客户都购买了杂志, 所以预测这个新客户也会购买杂志, 并且购买的概率为3/3, 即100%. 同样, 与最右边的新客户距离最近的3个人都未购买杂志, 所以预测这个新客户也不会购买杂志, 且不会购买的概率为3/3, 也为100%, 所以他购买杂志的概率就为0. 中间的客户, 与它距离最近的3个客户有2个没有购买, 预测他不购买杂志的概率大, 即为2/3, 只有1/3的概率会购买杂志.

K最近邻的原理与聚类有些相似, 计算新客户与旧客户之间的距离, 使用距离最短的K个旧客户的行为来预测新客户的行为.



## K最近邻

• Customers of the same cluster will show the same behavior. 在同一个聚类中的所有客户会表示出相似的行为. 与某个客户最接近的K个人, 可以认为他们处于同一个聚类中. 如果与某客户最接近的K个人都会购买某杂志, 那么他也很可能会购买此杂志.

• Do as your neighbors do. 做邻居做的事情. 如果上面这两个假设不成立, K最近邻的预测结果就很不好. 如何能使上面两个假设成立呢, 选择的输入属性要与预测的事情有非常紧密的关联. 对上面的例子来说, 选择的输入属性要与是否会购买杂志之间要是非常相关的属性. 如果选择的输入属性与是否会购买杂志之间不相关, 那么在输入属性上接近的客户, 对目标客户的预测效果就不好, 就不会做邻居所做的事情. 所以要做邻居所做的事情的前提, 就是输入属性要选择的好.

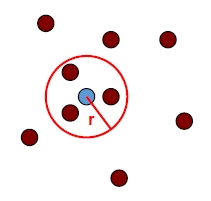
• Not really a learning technique. 最近邻方法其实并不是一个学习的算法. 因为所有学习的算法都是给定一个训练的数据, 使用学习的算法从这个训练数据中截取其中隐含的信息, 在学习完成后, 就可以离开这个训练数据, 对其它数据进行预测. 但K近邻就不符合这个标准. 如上面的例子, 训练数据就是已知购买未购买的客户的Age-Income属性特征. 使用K近邻法其实并没有从训练数据上学习到任何的知识, 只是在训练数据中进行搜寻, 搜寻到与要预测的对象最接近的K个对象. 无法脱离训练数据对新的对象进行预测. 虽然如此, K近邻方法往往能够得到比较好的结果

• Disadvantage:

• Inefficiency. 在对一个新的对象进行预测时, 要计算与原数据集中所有的对象的距离, 计算完成后还要进行排序, 然后找出距离最短的K个人, 再使用这K个人的行为来预测新的对象的行为. 当数据集非常大时, 计算的时间就非常长.

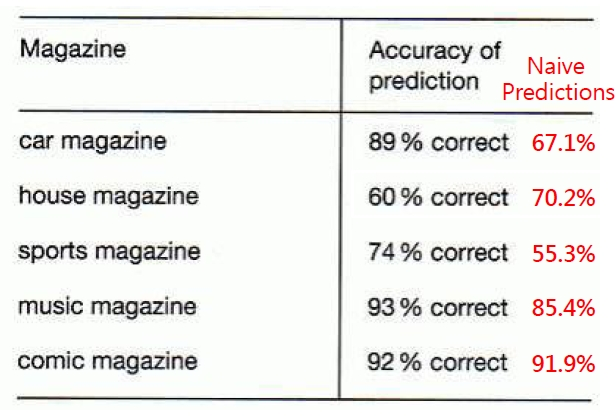
• It is difficult to understand that the performance of k-nearest neighbor is better

than naïve prediction. 二是很难理解为什么K近邻的预测效果会优于简单的预测. 简单/全猜预测就是如果一个群体中购买的人较多, 就预测其中的所有人都购买. 即一个群体中大多数人是怎样, 就猜测所有人都这样. 如果K近邻的输入变量选择的好, 结果往往会优于简单预测, 如果输入变量选的不好, 结果可比简单的预测还要差.



KNN对分类型数据的预测, 采用投票的结果进行预测, KNN也可以对数字进行预测. 找出来K个最近邻的点, 然后计算平均值, 用最近邻的K个人的平均值来预测新客户的值.

## K最近邻范例



对house magazine的预测, 为了提高它的预测精度, 可能要选择更好的输入变量, 或者使用其它的预测模型.

## K最近邻模型建置

• 实作KNN需考虑以下3件事

• 数据的前处理(Data Preprocessing)

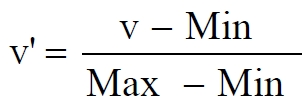
• 属性有Scale的问题(Age vs. Income)

• 距离的计算(Distance Calculation)

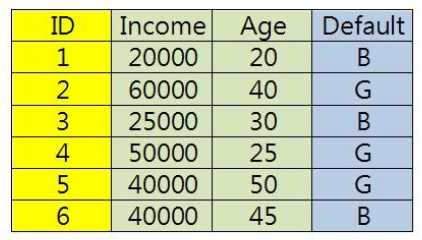
• 预测概率的推估(Predicted Probability)

### 数据前处理

• 极值正规化(Min-Max Normalization) [0, 1]



在聚类时每个输入字段都必须是数值型, KNN方法的输入字段也必须是数值型. 并且通常会把数值转换成[0,1]之间的数值.

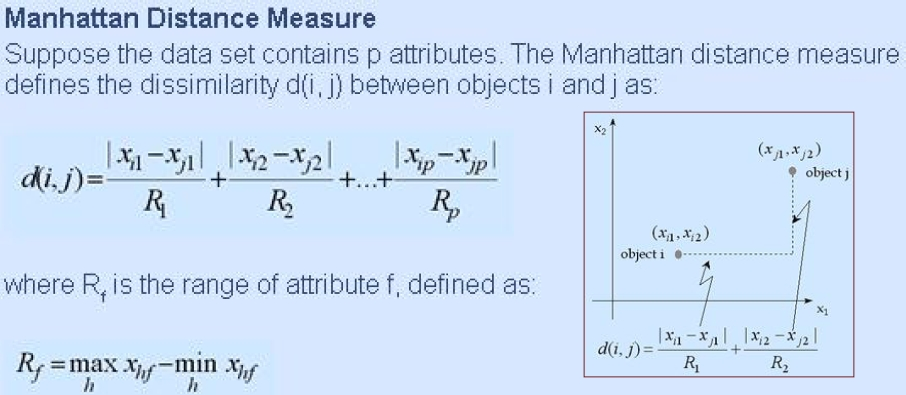


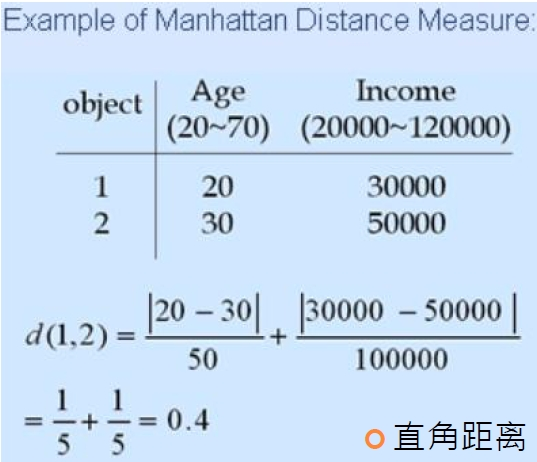
### 距离的计算

#### 曼哈顿距离/街区距离/直角距离

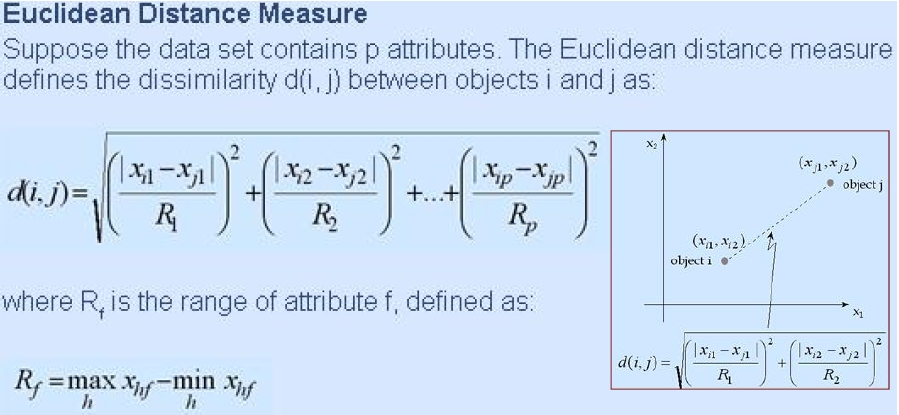
• Manhattan Distance / City Block Distance

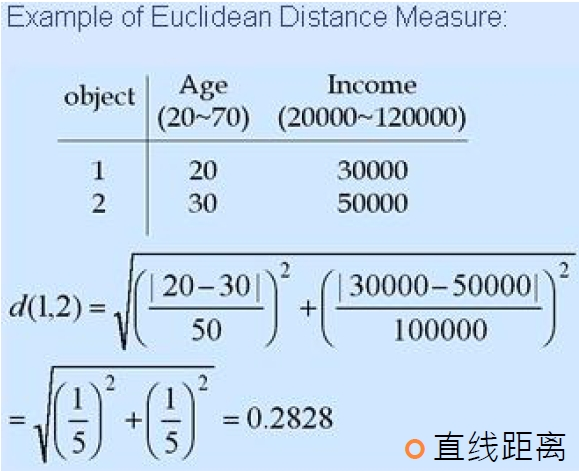
求曼哈顿距离的过程中就隐含着对数据进行标准化的过程, 也可以先对数据进行极值标准化, 然后把不同维度的数据直接相加, 就得到了曼哈顿距离.





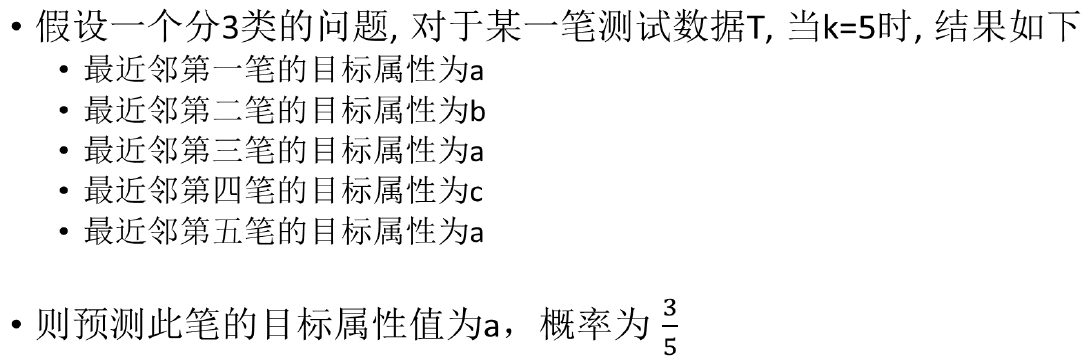
#### Euclidean Distance (欧几里得距离)





欧氏距离比街区距离用的更多, 很多包中都提供欧氏距离的计算, 有的不开根号直接使用平方和作为欧氏距离.

### 预测概率的推估



如果是分类型预测, 投票决定, 如果是数值型预测, 求平均值.

## K最近邻的应用：电影推荐



假设: 选择10项兴趣中同样兴趣的人也喜欢同样的电影.



简化为7个兴趣. 构建出不同的兴趣组合, 并使用数字来表示. 如果有一个新人注册网站了, 选择了0, 3, 6的兴趣, 要根据他所选择的兴趣给向他推荐他可能会喜欢的电影.

### 最近邻for 0\*3\*6

对0, 3, 6这样的兴趣组合, 看它与资料库中哪个用户的兴趣比较接近. 对应的兴趣相同, similarity+1, 对应的兴趣不同, Distance+1

C1: 1 0 0 1 0 0 1

M1: 0 1 1 1 0 0 1

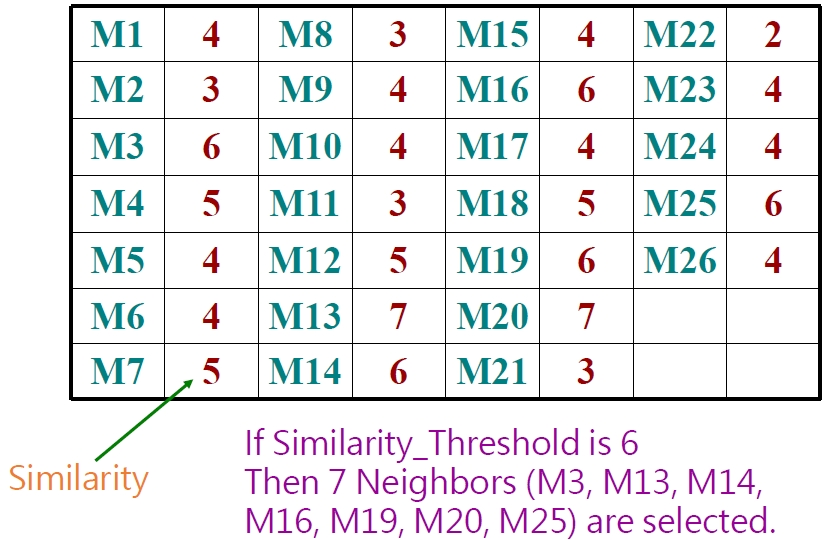
Distance = 3 or Similarity = 4

C1: 1 0 0 1 0 0 1

M2: 0 1 1 1 0 1 1

Distance = 4 or Similarity = 3

计算与所有原数据中用户的相似度, 只有相似度在6以上的才算是最近邻的用户.



### 找到7个邻居

列出来所有邻居喜欢的电影, Like Movies中为喜欢的电影编号.

• Neighbor 1:

• Like Movies 111 134 388 262 261 266 268 012 260 184 238 091 104 142 038

• Neighbor 2:

• 240 256 290 441 442 442 510 518 518 520 522 001 005 016 184

• Neighbor 3:

• none

• Neighbor 4:

• 402 193 228 179 227 111 204 364

• Neighbor 5:

• 280

• Neighbor 6:

• 193

• Neighbor 7:

• 186 189 193 214 239 179 227 263 240

### 推荐的电影列表 for 0\*3\*6

从7个邻居喜欢的电影列表中找出来共同喜欢的电影, 按照电影喜欢的人数进行排序, 把这个列表从上到下推荐给新客户.

Count = 03 Movie = 卧虎藏龙(193)

Count = 02 Movie = 尖峰时刻(184)

Count = 02 Movie = 蛇眼(240)

Count = 02 Movie = 美丽人生(442)

Count = 02 Movie = 厄夜丛林(518)

Count = 02 Movie = 楚门的世界(111)

Count = 02 Movie = 全民公敌(179)

Count = 02 Movie = 神鬼传奇(227)

## R语言实作

• 以R语言进行K最近邻分类模型建置及算法参数调整

• 以R语言进行K最近邻数字预测模型建置及算法参数调整

## K最近邻-分类范例1-疾病预测

• 给定以下的疾病诊断数据集

• 这个数据集包含1个病人代号的字段、5个条件字段(喉咙痛、发烧、淋巴腺踵胀、充血、头痛)以及一个目标字段(诊断结果)

条件字段即输入字段. 构建一个分类的模型, 以病人的5个条件字段为输入字段, 根据已有的数据, 预测他的诊断结果是什么. 使用KNN找到输入字段即病症上接近的人, 看这些邻居得到哪些疾病, 最后根据投票决定他得到哪一种疾病.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 病人代号 | 喉咙痛 | 发烧 | 淋巴腺肿胀 | 充血 | 头痛 | 诊断结果 |
| 1 | Yes | Yes | Yes | Yes | Yes | 链球菌喉炎 |
| 2 | No | No | No | Yes | Yes | 过敏 |
| 3 | Yes | Yes | No | Yes | No | 感冒 |
| 4 | Yes | No | Yes | No | No | 链球菌喉炎 |
| 5 | No | Yes | No | Yes | No | 感冒 |
| 6 | No | No | No | Yes | No | 过敏 |
| 7 | No | No | Yes | No | No | 链球菌喉炎 |
| 8 | Yes | No | No | Yes | Yes | 过敏 |
| 9 | No | Yes | No | Yes | Yes | 感冒 |
| 10 | Yes | Yes | No | Yes | Yes | 感冒 |

• 利用KNN预测下列病人的诊断结果 (K=3)

• Distance 的计算方式

• Distance (Yes, No) = 1, 病症不一致, 距离为1

• Distance (Yes, Yes) = 0, 病症相同, 距离为0

• Distance (No, No) = 0

• 利用欧几里得距离计算两个客户的Distance

### 软件实现 class:knn

# 关于病症诊断的案例

# Read Diseases Dataset

data <- read.csv("D:/R\_edu/data/diseases.csv")

# 把目标字段和输入字段分开, 并且对输入字段进行处理, 转变为数值型变量

data.y <- data[,c("Diagnosis")]

# 输入字段去除掉第1列和最后1列

data.x <- data[,-c(1,7)]

# 把分类型数据转化为数值型数据. 把Yes转变为1, No转变为0. 并把x都转变为数值型的.

# Data Preprocessing

replacev <- function(x) {

x = sub("Yes","1",x)

x = sub("No","0",x)

x = as.integer(x)

return (x)

}

# 通过laaply把data.x中每一列的数据都转变为数值型的.

data.x <- as.data.frame(lapply(data.x, replacev))

# 通过cbind把x和y拼接到一起, 目标字段是分类型变量, 不需要进行转换, 所以还使用原来的数据. 得到的就是已经处理好过的数据.

data <- cbind(data.x, Diagnosis=data.y)

# 产生训练数据集和测试数据集. 80%做为训练数据集, 20%作为测试数据集.

# Generate Training & Test Datasets

set.seed(102)

# 从1到10中随机选择8个数据

select <- sample(1:nrow(data),nrow(data)\*0.8)

# 除去最后一列作为训练集的x

train.x <- data[select,-c(6)]

# 最后一列作为训练集的y

train.y <- data[select,c("Diagnosis")]

# 取反后除去最后一列作为测试数据集的x

test.x <- data[-select,-c(6)]

# 取反后的最后一列作为训练数据集标准的y

test.y <- data[-select,c("Diagnosis")]

# 构建KNN模型

# Build KNN Model

library(class)

# 设定训练数据和测试数据, 对测试数据中的每一个数据, 都从训练数据中去找邻居. 对训练数据的y进行投票, k=3, 找到3个最近邻. prob=T除了投票数据之外, 还要计算出来概率. use.all=F, 有时候找到第3个最近邻时发现后面的数据如k=4,k=5等也是测试集中数据的最近邻, 此时就只取到第3个数据, 忽略其后的最近邻? 还是从3,4,5,6等距离相等的中随机挑选一个?

# class:knn这个函数只能进行分类型字段的预测, 即目标字段一定要是分类型的, 不能对数值型的字段进行预测. 并且这个函数只能

test.y\_hat <- class::knn(train = train.x, test = test.x, cl=train.y, k=3, prob = T, use.all=F)

# 查看测试数据的结果, 显示所有测试数据的诊断结果和对应的概率

test.y\_hat

[1] Allergy Cold

attr(,"prob")

[1] 0.6666667 0.6666667

Levels: Allergy Cold Strep\_throat

# 查看测试数据集的标准结果, 与KNN方法得出来的数据一致

test.y

[1] Allergy Cold

Levels: Allergy Cold Strep\_throat

# 单独查看测试数据结果的对应概率

attr(test.y\_hat,"prob")

[1] 0.6666667 0.6666667

# 模型评估, 看KNN的结果是否与标准的结果相同, 得到与标准结果相同的比率. 如果两笔数据都相同, sum的结果为2, 除以length(test.y), 得到1, 即是100%相同

# Model Evaluation

accuracy.knn <- sum(test.y\_hat==test.y) / length(test.y)

accuracy.knn

[1] 1

# 把KNN的结果和标准结果对比的逻辑值保存下来

agreement\_KNN <- test.y\_hat == test.y

agreement\_KNN

[1] TRUE TRUE

# 列联表分析. 交叉表. 有了交叉表, 就可以在此基础上进行其它分析. 测试数据有3个, 标准结果也有3个, 如果对角线上的数据为1, 表示测试数据和标准结果相同, 如果非对象线上的值为1, 表示测试结果与标准结果不同.

table(test.y\_hat, test.y)

test.y

test.y\_hat Allergy Cold Strep\_throat

Allergy 1 0 0

Cold 0 1 0

Strep\_throat 0 0 0

## K最近邻-分类范例2-保险购买预测

• 给定以下保险数据集

• 这个数据集包含1个客户编号的字段、4个条件字段(收入、是否有

信用卡保险、性别、年龄)以及一个目标字段(是否接受寿险推销)

对于各个输入变量, 一般认为收入越高, 接受保险的可能性越大. 如果过去接受过保险, 通常接受寿险的可能性也会越大. 女性比男性接受的比例也更大. 年龄越大接受保险的可能性也越大, 但年龄也不能太大.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 客户编号 | 收入 | 是否有信用卡保险 | 性别 | 年龄 | 是否接受寿险推销 |
| 1 | 40-50K | No | Male | 40-49 | No |
| 2 | 30-40K | No | Female | 40-49 | Yes |
| 3 | 40-50K | No | Male | 40-49 | No |
| 4 | 30-40K | Yes | Male | 40-49 | Yes |
| 5 | 50-60K | No | Female | 30-39 | Yes |
| 6 | 20-30K | Yes | Female | 50-59 | No |
| 7 | 30-40K | Yes | Male | 30-39 | Yes |
| 8 | 20-30K | No | Male | 20-29 | No |
| 9 | 30-40K | No | Female | 40-49 | Yes |
| 10 | 40-50K | No | Female | 40-49 | Yes |

• 利用KNN预测下列客户是否接受寿险推销 (K = 3)

• Distance 的计算方式

• 收入: 20-30K (0.33) 30-40K (0.33) 40-50K (0.33) 50-60K. 把收入转变为数值型变量. 如果有3个分隔间隔, 每个间隔都设置为0.33. 这样, 20-30K就可以转变为0, 30-40K转变为0.33, 40-50K转变为0.66, 50-60K转变为1. 如果收入为数值型字段, 本来就是要把它标准化为0-1之间的数值.

• 是否有信用卡保险: Yes (1) No

• 性别: Male (1) Female

• 年龄: 20-29 (0.33) 30-39 (0.33) 40-49 (0.33) 50-59

• 利用欧几里得距离计算两个客户的Distance, 找到K个距离最短的邻居, 以邻居的行为预测新客户的行为.

### 软件实现 class::knn

# 关于保险推销的数据

# Read InsuPromotion Dataset

data <- read.csv("D:/R\_edu/data/InsuPromotion.csv")

# 把输入字段和目标字段分离开

data.y <- data[,c(5)]

data.y

data.x <- data[,-c(5)]

data.x

Income CreditCardInsu Sex Age

1 40-50K No Male 40-49

2 30-40K No Female 40-49

3 40-50K No Male 40-49

4 30-40K Yes Male 40-49

5 50-60K No Female 30-39

6 20-30K Yes Female 50-59

7 30-40K Yes Male 30-39

8 20-30K No Male 20-29

9 30-40K No Female 40-49

10 40-50K No Female 40-49

# 定义一个函数, 进行把数据集中的分类型变量转换为数值型变量. 因为有小数点, 最后再进一步转换为浮点double类型的.

# Data Preprocessing

replacev <- function(x) {

x = sub("20-30K","0",x)

x = sub("30-40K","0.33",x)

x = sub("40-50K","0.66",x)

x = sub("50-60K","1.0",x)

x = sub("20-29","0",x)

x = sub("30-39","0.33",x)

x = sub("40-49","0.66",x)

x = sub("50-59","1.0",x)

x = sub("Yes","1",x)

x = sub("No","0",x)

x = sub("Male","1",x)

x = sub("Female","0",x)

x = as.double(x)

return (x)

}

# 把x中所有的分类型变量转换为数值型变量

data.x <- as.data.frame(lapply(data.x, replacev))

data.x

Income CreditCardInsu Sex Age

1 0.66 0 1 0.66

2 0.33 0 0 0.66

3 0.66 0 1 0.66

4 0.33 1 1 0.66

5 1.00 0 0 0.33

6 0.00 1 0 1.00

7 0.33 1 1 0.33

8 0.00 0 1 0.00

9 0.33 0 0 0.66

10 0.66 0 0 0.66

# 把转换后的数据拼接到一起

data <- cbind(data.x, InsuPromotion=data.y)

data

# 产生训练数据集和测试数据集

# Generate Training & Test Datasets

set.seed(104)

select <- sample(1:nrow(data),nrow(data)\*0.8)

train.x <- data[select,-c(5)]

train.y <- data[select,c("InsuPromotion")]

test.x <- data[-select,-c(5)]

test.y <- data[-select,c("InsuPromotion")]

# 建立KNN模型进行预测

# Build KNN Model

library(class)

test.y\_hat <- class::knn(train = train.x, test = test.x, cl=train.y, k=3, prob = T, use.all=F)

test.y\_hat

[1] No Yes

attr(,"prob")

[1] 0.6666667 1.0000000

Levels: No Yes

# 查看结果的概率

attr(test.y\_hat,"prob")

[1] 0.6666667 1.0000000

# 查看标准答案, 二者一致

test.y

[1] No Yes

Levels: No Yes

# 对KNN模型的结果进行评估

# Model Evaluation

accuracy.knn <- sum(test.y\_hat==test.y) / length(test.y)

# 预测的准确率为100%

accuracy.knn

agreement\_KNN <- test.y\_hat == test.y

agreement\_KNN

[1] TRUE TRUE

table(test.y\_hat, test.y)

## K最近邻预测范例-宽频网络的数据

# Process Norminal Attribute

data = read.csv("D:/R\_edu/data/broadband.csv")

View(data)

nrow(data)

[1] 1114

names(data)

[1] "CUST\_ID" "GENDER" "AGE" "TENURE"

[5] "CHANNEL" "AUTOPAY" "ARPB\_3M" "CALL\_PARTY\_CNT"

[9] "DAY\_MOU" "AFTERNOON\_MOU" "NIGHT\_MOU" "AVG\_CALL\_LENGTH"

[13] "BROADBAND"

1为客户ID, 2为已转换为数值型变量的性别. 3为年龄, 4TENURE为跟顾客的往来期间. channel为已经转换为数值型的分类变量, 因为channel的取值为1,2,3,4, 需要转换为0,1的逻辑型分类字段, 把channel分拆为4个字段, 每个字段对应channel取值的1,2,3,4是否为真, channel为1时, 对应的4个字段值为1,0,0,0, channel为2时, 对应的4个字段为0,1,0,0, 如此类推.

# 把channel字段单独取出来进行处理.

data.c = data[,c("CHANNEL")]

# data.c默认是数值型字段, 需要转换为因子型字段

data.c = as.factor(data.c)

library("nnet")

# 神经网络也不接受类别型的字段,对于类别型字段也要进行类似的处理. 把数据从1列拆分为多列. 即摊平.

# 使用nnet包中的class.ind来对数据进行摊平

class.ind(data.c)

# 把数据转换为数据框, 默认的名称为1,2,3,4

dummy.c = as.data.frame(class.ind(data.c))

head(dummy.c)

# 重命名dummy.c的字段名称

names(dummy.c) <- c("Channel-1", "Channel-2", "Channel-3", "Channel-4")

dummy.c

head(dummy.c)

Channel-1 Channel-2 Channel-3 Channel-4

1 0 1 0 0

2 1 0 0 0

3 0 0 1 0

4 0 0 1 0

5 0 0 1 0

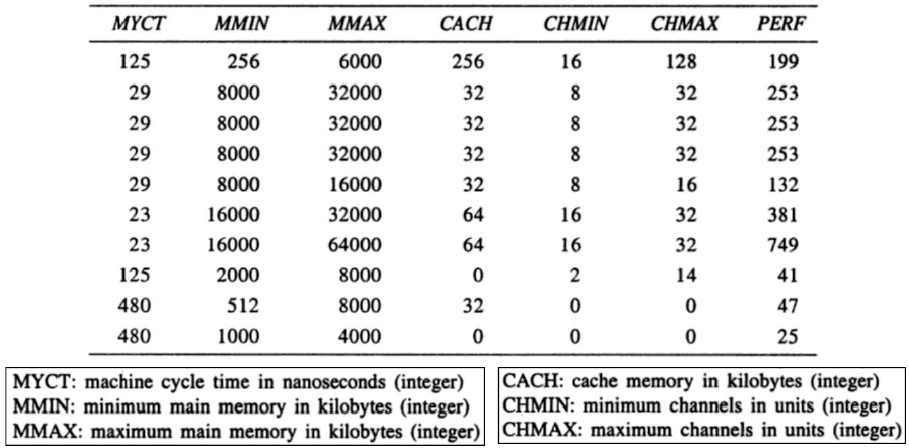
6 0 0 0 1

# 把摊平得到的新字段与channel列拼接到一起.

cbind(CHANNEL=data.c,dummy.c)

## K最近邻-预测范例

• CPU效能预测



## neighbr::knn包实现

### 分类型预测: 疾病预测

# 使用neighbr::knn进行KNN预测. 不仅能做分类型字段的预测, 还可以做数值型字段的预测, 不仅能进行预测, 还可以给出最近邻的匹配结果

# neighbr::knn方法的缺点是在对分类型变量进行预测的时候, 只能给出来预测的结果, 不能给出来预测结果对应的几率. 如果想要得到几率值, 就要手动从最近邻的数据中计算得出几率.

# Read Diseases Dataset

data <- read.csv("D:/R\_edu/data/diseases.csv")

# 提取输入字段和目标字段

data.x = data[,-ncol(data)]

data.y = data[,ncol(data)]

# 输入字段只能接受数值型的, 要把分类型的输入字段转换为数值型的.

# Data Preprocessing

replace = function(x){

x=sub("Yes","1",x)

x=sub("No","0",x)

x=as.integer(x)

return(x)

}

# 输入字段转换为数值型字段

data.x <-as.data.frame(lapply(data.x,replace))

# 产生训练和测试数据集

# Generate Training & Test Datasets

set.seed(102)

select <- sample(1:nrow(data),nrow(data)\*0.8)

# 把目标字段也加入到训练数据集中

train\_set = cbind(data.x[select,],Diagnosis=data.y[select])

# 与class:knn的用法不太相同的是, 在neighbr::knn这个方法中, 训练数据需要有ID字段, 输入字段和目标字段

train\_set

Patient\_ID Sore\_Throat Fever Swollen\_Glands Congestion Headache Diagnosis

6 6 0 0 0 1 0 Allergy

5 5 0 1 0 1 0 Cold

7 7 0 0 1 0 0 Strep\_throat

4 4 1 0 1 0 0 Strep\_throat

1 1 1 1 1 1 1 Strep\_throat

3 3 1 1 0 1 0 Cold

8 8 1 0 0 1 1 Allergy

10 10 1 1 0 1 1 Cold

# 测试集与class:knn中的方法相同, 并不需要目标字段, 只需要有输入字段就可以了. 并且测试集不能有ID字段. 即测试集中只需要有输入字段, 不能有ID和目标字段

test\_set.x = data.x[-select,-1]

test\_set.x

Sore\_Throat Fever Swollen\_Glands Congestion Headache

2 0 0 0 1 1

9 0 1 0 1 1

test\_set.y = data.y[-select]

test\_set.y

[1] Allergy Cold

Levels: Allergy Cold Strep\_throat

# 构建KNN模型, train\_set包含ID,输入字段和目标字段. test\_set只有输入字段, 没有ID和目标字段. categorical\_target="Diagnosis", 设定分类型预测(训练数据集中)的目标字段, 根据此字段进行分类型预测. continuous\_target= NULL, 表示数据型预测的目标字段. comparison\_measure="euclidean", 表示计算距离时使用的是欧几里得距离. return\_ranked\_neighbors=3, 表示预测的结果中也返回3个最近的邻居. id="Patient\_ID"表示返回的邻居的ID就是Patient\_ID. 这就是为什么在训练 数据集中一定要有ID字段的原因.

# Build KNN Model

library(neighbr)

fit <- neighbr::knn(train\_set=train\_set,

test\_set=test\_set.x,

k=3,

categorical\_target="Diagnosis",

continuous\_target= NULL,

comparison\_measure="euclidean",

return\_ranked\_neighbors=3,

id="Patient\_ID")

mode(data)

class(data)

summary(fit)

Length Class Mode

call 9 -none- call

k 1 -none- numeric

categorical\_target 1 -none- character

continuous\_target 0 -none- NULL

comparison\_measure 1 -none- character

categorical\_scoring\_method 1 -none- character

continuous\_scoring\_method 1 -none- character

return\_ranked\_neighbors 1 -none- numeric

id 1 -none- character

features 5 -none- character

function\_name 1 -none- character

categorical\_levels 3 -none- character

num\_train\_rows 1 -none- numeric

num\_test\_rows 1 -none- numeric

train\_set 7 data.frame list

test\_set\_scores 4 data.frame list

# 查看测试数据的结果, 包含预测的结果和与它最近邻的3个邻居. 测试数据集2的最近邻为6,8,5, 在data中查看, 它们的结果分别是Allergy,Allergy,Cold, 所以预测的结果为Allergy. 测试集中9号的最近邻为5,10,6, 对应的标准结果为cold, cold, Allergy, 所以预测结果为Allergy

results <- fit$test\_set\_scores

results

categorical\_target neighbor1 neighbor2 neighbor3

2 Allergy 6 8 5

9 Cold 5 10 6

# 对模型的结果进行评估.

# Model Evaluation

# 取出来test\_set\_scores中的预测结果

test\_set.y\_hat <- fit$test\_set\_scores$categorical\_target

test\_set.y\_hat

[1] "Allergy" "Cold"

# 与标准答案进行对比, 计算预测的准确度

accuracy.knn <- sum(test\_set.y\_hat==test\_set.y)/length(test\_set.y)

accuracy.knn

[1] 1

### 连续型预测: 房价预测

# Read Home Prediction Dataset, 对房价进行预估的数据集

# 对目标字段为数值型的进行预测.

data = read.csv("D:/R\_edu/data/home\_prediction.csv")

# 取出来id字段, 输入字段和目标字段

# 第1列为id字段

data.id = data[,1]

# 除去第1列和最后一列为输入字段. 输入字段中不能有id和目标字段

data.x = data[,-c(1,ncol(data))]

data.x

Space Offices Entrances Age

1 2310 2 2.0 20

2 2333 2 2.0 12

3 2356 3 1.5 33

4 2379 3 2.0 43

5 2402 2 3.0 53

6 2425 4 2.0 23

7 2448 2 1.5 99

8 2471 2 2.0 34

9 2494 3 3.0 23

10 2517 4 4.0 55

11 2540 2 3.0 22

# 最后一列为目标字段, 即真实的房价数据

data.y = data[,ncol(data)]

# 数据预处理, 所有数值型的数据都要标准化为0-1之间

# Data Preprocessing, 定义一个极值标准化函数

normalize <- function(x) {

return((x - min(x)) / (max(x) - min(x)))

}

# 对所有输入变量进行极值标准化

data.x <- as.data.frame(lapply(data.x, normalize))

data.x

Space Offices Entrances Age

1 0.0 0.0 0.2 0.09195402

2 0.1 0.0 0.2 0.00000000

3 0.2 0.5 0.0 0.24137931

4 0.3 0.5 0.2 0.35632184

5 0.4 0.0 0.6 0.47126437

6 0.5 1.0 0.2 0.12643678

7 0.6 0.0 0.0 1.00000000

8 0.7 0.0 0.2 0.25287356

9 0.8 0.5 0.6 0.12643678

10 0.9 1.0 1.0 0.49425287

11 1.0 0.0 0.6 0.11494253

# 产生训练和测试数据集, 训练数据有8笔, 测试数据有3笔. 对行的选择和对列的选择. 抽样是对符合条件的行的选择,

# Generate Training & Test Datasets

set.seed(107)

select <- sample(1:nrow(data),nrow(data)\*0.8)

select

[1] 6 1 5 10 2 4 7 3

# 从原数据集中提取出用于训练的数据, 组合成ID+输入字段+目标字段的格式

train\_set = cbind(ID=data.id[select],data.x[select,],Value=data.y[select])

train\_set

ID Space Offices Entrances Age Value

6 6 0.5 1.0 0.2 0.12643678 169000

1 1 0.0 0.0 0.2 0.09195402 142000

5 5 0.4 0.0 0.6 0.47126437 139000

10 10 0.9 1.0 1.0 0.49425287 169000

2 2 0.1 0.0 0.2 0.00000000 144000

4 4 0.3 0.5 0.2 0.35632184 150000

7 7 0.6 0.0 0.0 1.00000000 126000

3 3 0.2 0.5 0.0 0.24137931 151000

# 提取出测试数据的输入字段和目标字段

test\_set.x = data.x[-select,]

test\_set.x

Space Offices Entrances Age

8 0.7 0.0 0.2 0.2528736

9 0.8 0.5 0.6 0.1264368

11 1.0 0.0 0.6 0.1149425

test\_set.y = data.y[-select]

test\_set.y

[1] 142900 163000 149000

# 构建KNN模型. 由于此时是对数值型变量进行预测, 所以categorical\_target=NULL, continuous\_target= "Value", 并使用ID字段找出最接近的5个近邻值

# Build KNN Model

library(neighbr)

fit <- neighbr::knn(train\_set=train\_set,

test\_set=test\_set.x,

k=5,

categorical\_target=NULL,

continuous\_target= "Value",

comparison\_measure="euclidean",

return\_ranked\_neighbors=5,

id="ID")

summary(fit)

Length Class Mode

call 9 -none- call

k 1 -none- numeric

categorical\_target 0 -none- NULL

continuous\_target 1 -none- character

comparison\_measure 1 -none- character

categorical\_scoring\_method 1 -none- character

continuous\_scoring\_method 1 -none- character

return\_ranked\_neighbors 1 -none- numeric

id 1 -none- character

features 4 -none- character

function\_name 1 -none- character

categorical\_levels 0 -none- NULL

num\_train\_rows 1 -none- numeric

num\_test\_rows 1 -none- numeric

train\_set 6 data.frame list

test\_set\_scores 6 data.frame list

results <- fit$test\_set\_scores

results

continuous\_target neighbor1 neighbor2 neighbor3 neighbor4 neighbor5

8 145200 5 4 2 1 3

9 155600 4 6 5 10 3

11 145200 5 4 2 1 3

test\_set.y\_hat <- results$continuous\_target

test\_set.y\_hat

[1] 145200 155600 145200

# 对模型结果进行评估

# Model Evaluation, 计算平均绝对误差

mae = sum(abs(test\_set.y\_hat-test\_set.y)) / length(test\_set.y)

# 对于15W的值, 误差为4500, 大概1/30, 所以误差还是比较小的

mae

[1] 4500

# 估计预测的精度. 大概为97%

(mean(test\_set.y)-mae)/mean(test\_set.y)

[1] 0.9703231

# 宽频网络的精准销售案例

# Process Norminal Attribute

data = read.csv("D:/R\_edu/data/broadband.csv")

View(data)

data.c = data[,c("CHANNEL")]

data.c = as.factor(data.c)

head(data.c,12)

[1] 2 1 3 3 3 4 4 1 1 3 3 3

Levels: 1 2 3 4