临床预测模型构建&机器学习(R语言进阶)

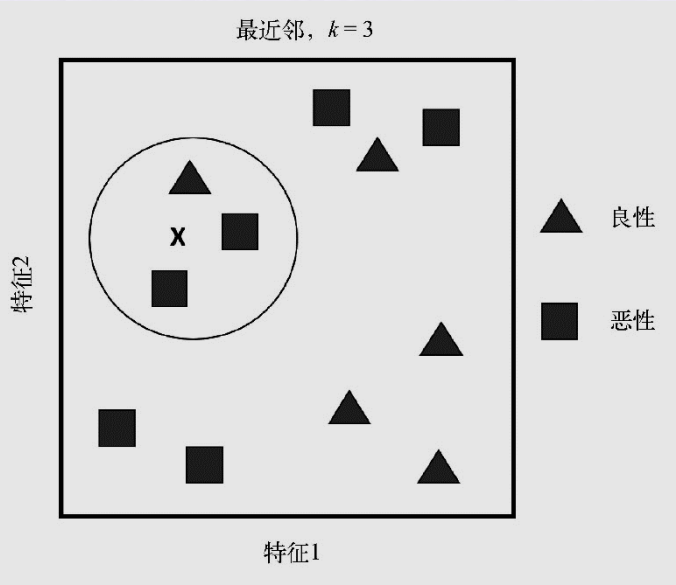
第13章 K最近邻法与支持向量机在医学中应用

周支瑞

CONTENT

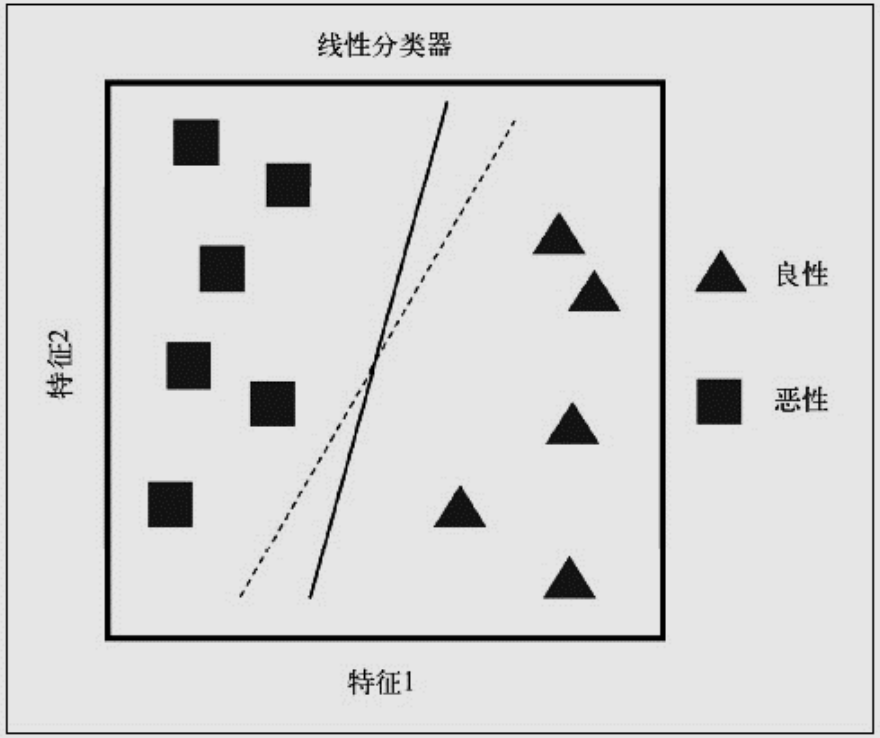
K最近邻法 原理

- KNN中没有参数，这种方法是所谓的“基于实例的学习”。简言之，保存被标记过的实例（输入和相应的输出标记），什么都不做直至一个新的输入模式请求一个输出值（Battiti和Brunato，2014，p.11）。这种方法通常称为懒惰学习，因为不产生具体的模型参数。用于训练的实例本身就是知识。要预测任何一个新实例（新数据点）时，需要对训练数据进行搜索，找到一个最类似于新实例的实例。KNN处理分类问题的方法是找最近的点（最近邻）来确定正确的分类。



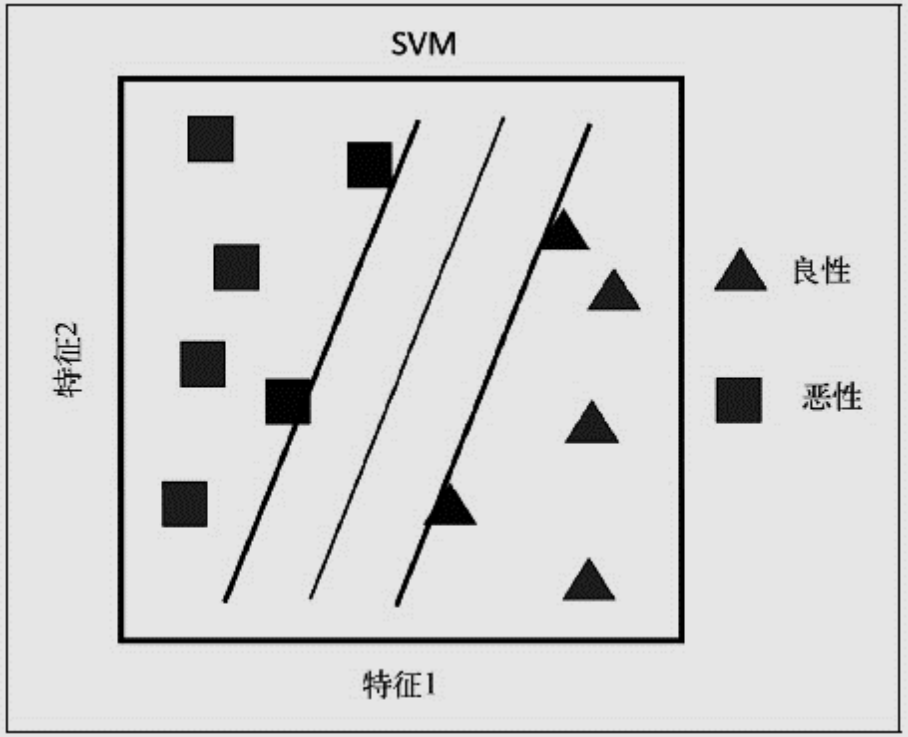
支持向量机原理

- SVM表现优异，被认为是最好的“开箱即用”的分类器之一（James，G.，2013）。为了切实理解这个主题，我们再看一个简单的可视化例子。右图中的分类任务是线性可分的，但虚线和实线只不过是无数线性可行解中的两个。如果问题的维度多于2，你就需要分割超平面。任意一个点落入线性分类器错误一侧的概率，对于虚线来说，这个概率要高于实线



支持向量机原理

- 任意一个点落入线性分类器错误一侧的概率，对于虚线来说，这个概率要高于实线，这说明实线具有更高的分类安全边际。正如Battiti和Brunato所说，SVM是具有最大可能边际的线性分类器，支持向量就是最接近安全边际两侧的那些向量。右图说明了这个思想。细实线就是最优线性分类器，它建立了上面提到的最大可能边际，提高了一个新观测落入分类器正确一侧的概率。两条粗黑线对应着安全边际，阴影数据点构成了支持向量。分类器之间的距离被称为边际。



SVM最优化问题及其约束条件

（1）找出使边际最大的权重值。
（2）满足约束条件：没有（或尽量少的）数据点位于边际之内。

CONTENT

典型案例分析

- 数据来自美国国家糖尿病消化病肾病研究所，这个数据集包括532个观测，8个输入特征以及1个二值结果变量（Yes/No）。这项研究中的患者来自美国亚利桑那州中南部，是皮玛族印第安人的后裔。选择皮玛印第安人进行这项研究是因为，半数成年皮玛印第安人患有糖尿病。而这些患有糖尿病的人中，有95%超重。研究仅限于成年女性，病情则按照世界卫生组织的标准进行诊断为2型糖尿病。这种糖尿病的患者胰腺功能并未完全丧失，还可以产生胰岛素，因此又称“非胰岛素依赖型”糖尿病。我们的任务是研究那些糖尿病患者，并对这个人群中可能导致糖尿病的风险因素进行预测。

案例数据概览

- 数据集包含了532位女性患者的信息，存储在两个数据框中。数据集变量如下：
  （1）npreg：怀孕次数。
  （2）glu：血糖浓度，由口服葡萄糖耐量测试给出。
  （3）bp：舒张压（单位为mm Hg）。
  （4）skin：三头肌皮褶厚度（单位为mm）。
  （5）bmi：身体质量指数。
  （6）ped：糖尿病家族影响因素。
  （7）age：年龄。
  （8）type：是否患有糖尿病（是/否）。
- 数据集包含在MASS这个R包中，一个数据框是Pima.tr，另一个数据框的是Pima.te。我们不将它们分别作为训练集和测试集，而是将其合在一起，然后建立自己的训练集和测试集，目的是学习如何使用R完成这样的任务。

加载必要程序包

- library(class)
- library(kknn)
- library(e1071)
- library(kernlab)
- library(caret)
- library(MASS)
- library(reshape2)
- library(ggplot2)
- library(pROC)

CONTENT

数据准备--合并数据框

- 现在加载数据集并检查其结构，确保结构相同。检查数据集结构之后，我们确信可以把两个数据框合成一个。这非常容易，使用rbind()函数即可，它的功能是绑定行并追加数据。如果你的每个数据框都有相同的观测并想追加特征，则应该使用cbind()函数按列绑定数据。给你的新数据框命名也非常简单，可以使用语法:newdata=rbind(data frame1, data fram2)。

代码如下：

- data(Pima.tr)
- str(Pima.tr)
- data(Pima.te)
- str(Pima.te)
- pima <- rbind(Pima.tr, Pima.te)
- str(pima)

数据准备--数据探索

- 通过箱线图进行探索性分析。为此，要使用结果变量“type”作为ID变量的值。和Logistic回归一样，melt()函数会融合数据并准备好用于生成箱线图的数据框。我们将新的数据框命名为pima.melt，标准化数据，绘制箱线图，并且划分训练集与验证集，R代码如下所示：

代码如下：

```
pima.melt <- melt(pima, id.var = "type")
ggplot(data = pima.melt, aes(x = type, y = value)) +
        geom_boxplot() + facet_wrap(~ variable, ncol = 2)
pima.scale <- data.frame(scale(pima[, -8]))
#scale.pima = as.data.frame(scale(pima[,1:7], byrow=FALSE)) #do not create own
function
str(pima.scale)
pima.scale$type <- pima$type
```

代码如下：

```
pima.scale.melt <- melt(pima.scale, id.var = "type")
ggplot(data=pima.scale.melt, aes(x = type, y = value)) +
        geom_boxplot() + facet_wrap(~ variable, ncol = 2)
cor(pima.scale[-8])
table(pima.scale$type)
set.seed(502)
ind <- sample(2, nrow(pima.scale), replace = TRUE, prob = c(0.7, 0.3))
train <- pima.scale[ind == 1, ]
test <- pima.scale[ind == 2, ]
str(train)
str(test)
```

CONTENT

KNN建模

- 使用KNN建模关键在于选择最合适的参数（k）。在确定k值方面，caret包又可以大显身手了。先建立一个供实验用的输入网格，k值从2到20，每次增加1。使用expand.grid()和seq()函数可以轻松实现。在caret包中，作用于KNN函数的参数非常简单直接，就是.k。
- 选择参数时，还是使用交叉验证。先建立一个名为control的对象，然后使用caret包中的trainControl()函数。
- 使用train()函数建立计算最优k值的对象，train()函数也在caret包中。别忘了在进行任何随机抽样之前，都要先设定随机数种子。使用train()函数建立对象时，需要指定模型公式、训练数据集名称和一个合适的方法。模型公式和以前一样－y ~ x，方法就是knn。这些参数设定之后，R代码就可以建立对象并计算最优k值了，代码如下所示：

代码如下：

```
grid1 <- expand.grid(.k = seq(2, 20, by = 1))
control = trainControl(method = "cv")
set.seed(123)
knn.train <- train(type ~ ., data = train,
                   method = "knn",
                   trControl = control,
                   tuneGrid = grid1)
knn.train
```

kappa值计算

- 一致性百分比是分类器的分类结果与实际分类相符合的程度（就是正确率），期望一致性百分比是分类器靠随机选择获得的与实际分类相符合的程度。Kappa统计量的值越大，分类器的分类效果越好，Kappa为1时达到一致性的最大值。下面将模型应用到测试数据集上，通过一个完整的例子说明如何计算正确率和Kappa。使用class包中的knn()函数实现。要使用这个函数，至少需要指定4个参数：训练数据、测试数据、训练集中的正确标记、k值。

代码如下：

```
knn.test <- knn(train[, -8], test[, -8], train[, 8], k = 17)
table(knn.test, test$type)
(77+28)/147
#calculate Kappa
prob.agree <- (77+28)/147
prob.chance <- ((77+26)*(77+16)+(16+28)*(26+28))/(147*147)
prob.chance
kappa <- (prob.agree - prob.chance) / (1 - prob.chance)
kappa
library(fmsb)
mytable<-table(knn.test, test$type)
Kappa.test(mytable, conf.level=0.95)
```

加权最近邻法

- Kappa只是“Fair agreement”，在测试集上的正确率仅比70%高一点，所以应该看看是否可以使用加权最近邻法得到更好的结果。加权最近邻法提高了离观测更近的邻居的影响力，降低了远离观测的邻居的影响力。观测离空间点越远，对它的影响力的惩罚就越大。要使用加权最近邻法，需要kknn包中的train.kknn()函数来选择最优的加权方式。
- train.kknn()函数使用我们前面介绍过的LOOCV选择最优参数，比如最优的K最近邻数量、二选一的距离测量方式，以及核函数。

加权最近邻法

- 我们集中讨论两种加权方法：triangular和epanechnikov。赋予权重之前，算法对所有距离进行标准化处理，使它们的值都在0和1之间。triangular加权方法先算出1减去距离的差，再用差作为权重去乘这个距离。epanechnikov加权方法是用3/4乘以(1 - 距离的平方)。为了方便比较，我们把这两种加权方法和标准的不加权方法放在一起。
- 先指定随机数种子，然后使用kknn()函数建立训练集对象，这个函数要求指定的参数有：k值的最大值kmax、距离distance（1表示绝对值距离，2表示欧氏距离）、核函数。在我们的模型中，kmax设定为25，distance为2，代码如下：

代码如下：

```
set.seed(123)
kknn.train <- train.kknn(type ~ ., data = train,
                         kmax = 25, distance = 2,
                         kernel = c("rectangular", "triangular", "epanechnikov"))
plot(kknn.train)
kknn.train
kknn.pred <- predict(kknn.train, newdata = test)
table(kknn.pred, test$type)
```

CONTENT

SVM建模

- 使用e1071包构建SVM模型，先从线性支持向量分类器开始，然后转入非线性模型。e1071包中有一个非常好的用于SVM的函数--tune.svm()，它可以帮助我们选择调优参数及核函数。tune.svm()使用交叉验证使调优参数达到最优。我们先建立一个名为linear.tune的对象，然后使用summary()函数看看其中的内容。代码如下所示：

代码如下：

```
#linear tune
set.seed(123)
linear.tune <- tune.svm(type ~ ., data = train,
                        kernel = "linear",
                        cost = c(0.001, 0.01, 0.1, 1, 5, 10))
summary(linear.tune)
best.linear <- linear.tune$best.model
tune.test <- predict(best.linear, newdata = test)
table(tune.test, test$type)
(82+30)/147
```

参数调优

- 线性支持向量分类器在训练集和测试集上表现得都比KNN稍好。e1071包中有一个用于SVM的非常好的函数 -- tune.svm()，可以帮助我们选择调优参数或核函数。现在看看非线性模型能否表现更好，依然使用交叉验证选择调优参数。我们试验的第一个核函数是多项式核函数，需要调整优化两个参数：多项式的阶（degree）与核系数（coef0）。设定多项式的阶是3、4和5，核系数从0.1逐渐增加到4，代码如下所示：

代码如下：

```
#SVM with e1071; tune the poly only
set.seed(123)
poly.tune <- tune.svm(type ~ ., data = train,
                      kernel = "polynomial",
                      degree = c(3, 4, 5),
                      coef0 = c(0.1, 0.5, 1, 2, 3, 4))
summary(poly.tune)
best.poly <- poly.tune$best.model
poly.test <- predict(best.poly, newdata = test)
table(poly.test, test$type)
(81 + 26) / 147
```

径向基核函数模型

- 这个模型的表现还不如线性模型。下面测试径向基核函数，此处只需找出一个参数gamma，在0.1~4中依次检验。如果gamma过小，模型就不能解释决策边界的复杂性；如果gamma过大，模型就会严重过拟合。我们还是要在测试集上进行验证，和前面一样。

代码如下：

```
#tune the rbf
set.seed(123)
rbf.tune <- tune.svm(type ~ ., data = train,
                     kernel = "radial",
                     gamma = c(0.1, 0.5, 1, 2, 3, 4))
summary(rbf.tune)
best.rbf <- rbf.tune$best.model
rbf.test <- predict(best.rbf, newdata = test)
table(rbf.test, test$type)
(73+21)/147
```

sigmoid核函数模型

- 提高性能的最后一招只能是kernel=“sigmoid”了。需要找出两个参数 -- gamma和核系数（coef0）

代码如下：

```
#tune the sigmoid
set.seed(123)
sigmoid.tune <- tune.svm(type ~ ., data = train,
                         kernel = "sigmoid",
                         gamma = c(0.1, 0.5, 1, 2, 3, 4),
                         coef0 = c(0.1, 0.5, 1, 2, 3, 4))
summary(sigmoid.tune)
best.sigmoid <- sigmoid.tune$best.model
sigmoid.test <- predict(best.sigmoid, newdata = test)
table(sigmoid.test, test$type)
(82+35)/147
```

CONTENT

混淆矩阵

- KNN在测试集上最好的正确率只有71%左右，相反，通过SVM可以获得接近80%的正确率。在简单选择具有最优正确率的模型（本案例中是使用sigmoid核函数的SVM模型）之前，先看看如何通过对混淆矩阵的深入研究来比较各种模型。
- 我们继续使用caret包的confusionMatrix()函数。请注意，在前面的内容中，我们使用过InformationValue包中的同名函数。但caret包中的这个函数会生成我们评价和选择最优模型所需的所有统计量。先从建立的最后一个模型开始，使用的语法和基础的table()函数一样，不同之处是要指定positive类，代码如下所示：

代码如下：

- confusionMatrix(sigmoid.test, test$type, positive = "Yes")
- confusionMatrix(tune.test, test$type, positive = "Yes")

CONTENT

SVM中的特征选择

- 我们要再次用到caret包，因为它可以基于kernlab包在线性SVM中进行交叉验证。要完成上面的任务，需要设定随机种子数，在caret包中的rfeControl()函数中指定交叉验证方法，使用rfe()函数执行一个递归的特征选择过程，最后检验模型在测试集上的运行情况。在rfeControl()中，你需要根据使用的模型指定functions参数。可以使用几种不同的functions参数，此处使用lrFuncs。如果想看可用的functions参数列表，最好使用?rfeControl和?caretFuncs命令查看相关文档。本例代码如下：

代码如下：

```
set.seed(123)
rfeCNTL <- rfeControl(functions = lrFuncs, method = "cv", number = 10)
svm.features <- rfe(train[, 1:7], train[, 8],
                    sizes = c(7, 6, 5, 4),
                    rfeControl = rfeCNTL,
                    method = "svmLinear")
svm.features
svm.5 <- svm(type ~ glu + ped + npreg + bmi + age,
             data = train,
             kernel = "linear")
svm.5.predict = predict(svm.5, newdata=test[c(1,2,5,6,7)])
table(svm.5.predict, test$type)
```

小结

- 本章讨论了两种新的分类技术：KNN和SVM，并在一个小规模数据集上分别使用这两种技术建立了模型，以预测某个人是否患有糖尿病，还对模型进行了比较。在KNN部分，我们介绍了不加权和加权的最近邻算法。在预测某个人是否患有糖尿病方面，这些方法的表现都不如SVM。我们研究了如何使用e1701包建立线性的和非线性的支持向量机，并对其进行调优。使用功能极其强大而又全面的caret包比较线性和非线性支持向量机的预测能力，然后发现，使用sigmoid核函数的非线性支持向量机具有最好的性能。最后，我们简单介绍了如何使用caret包进行粗略的特征选择。因为对于那些像SVM一样使用黑盒技术的方法来说，特征选择确实是个艰巨的挑战。

请在此处输入小标题

感谢观看

THANKS

丁香园特邀讲师 周支瑞