临床预测模型构建&机器学习(R语言进阶)

第14章 分类回归树在医学研究中应用

周支瑞

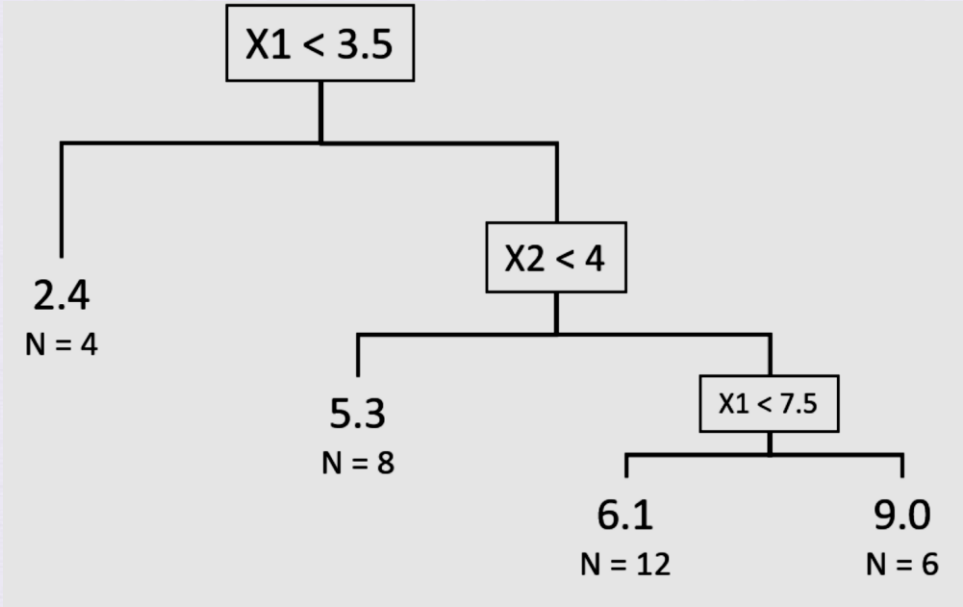
CONTENT

回归树 概念

- 树方法精髓就是划分特征，从第一次分裂开始就要考虑如何最大程度改善RSS，然后持续进行“树杈”分裂，直到树结束。后面的划分并不作用于全数据集，而仅作用于上次划分时落到这个分支之下的那部分数据。这个自顶向下的过程被称为“递归划分”。这个过程是贪婪的，贪婪的含义是指算法在每次分裂中都追求最大程度减少RSS，而不管以后的划分中表现如何。这样做可能会生成一个带有无效分支的树，尽管偏差很小，但是方差很大。为了避免这个问题，生成完整的树之后，你要对树进行剪枝，得到最优的解。
- 这种方法的优点是可以处理高度非线性关系，但它还存在一些潜在的问题：一个观测被赋予所属终端节点的平均值，这会损害整体预测效果（高偏差）。相反，如果你一直对数据进行划分，树的层次越来越深，这样可以达到低偏差的效果，但是高方差又成了问题。和其他方法一样，你也可以用交叉验证来选择合适的深度。

回归树 概念

- 假设数据集中有30个观测，响应变量的范围是1～10，有两个预测特征X1和X2，范围都是0～10。整个树有3个分支和4个终端节点。每次都按照基本的if-then语句，或者使用R函数ifelse()进行分裂。第一次分裂的条件是X1是否小于3.5，响应变量被分出4个观测，平均值为2.4，剩余26个观测。包含4个观测的左侧分支是一个终端节点，因为更深的分裂已经不能明显改善RSS了。树的这部分划分包含的4个观测的预测值就是平均值2.4。下一次分裂的条件是X2＜4，最后是X1＜7.5。



有3个分支和4个节点的回归树，节点下面标有均值和观测数量

分类树 概念

- 分类树与回归树的运行原理是一样的，区别在于决定分裂过程的不是RSS，而是误差率。这里的误差率不是你想的那样，简单地由误分类的观测数除以总观测数算出。实际上，进行树分裂时，误分类率本身可能会导致这样一种情况：你可以从下次分裂中获得一些有用信息，但误分类率却没有改善。
- 我们看一个例子。假设有一个节点N0，节点中有7个标号为N0的观测和3个标号为Yes的观测，我们就可以说误分类率为30%。记住这个数，然后通过另一种误差测量方式进行计算，这种方式称为基尼指数。单个节点的基尼指数计算公式如下：基尼指数 = 1 - (类别1的概率)^2 - (类别2的概率)^2
- 所以，对于N0，基尼指数为1 - (0.7)^2 - (0.3)^2，等于0.42，与之相对的误分类率为30%。

随机森林 概念

- 随机森林技术在模型构建过程中使用两种方法以提高模型预测能力。
- **第一个方法称为：自助聚集法或称装袋法。**在装袋法中，使用数据集的一次随机抽样建立一个独立树，抽样的数量大概为全部观测的2/3（请记住，剩下的1/3被称为袋外数据，out-of-bag）。这个过程重复几十次或上百次，最后取平均结果。其中每个树都任其生长，不进行任何基于误差测量的剪枝，这意味着每个独立树的方差都很大。但是，通过对结果的平均化处理可以降低方差，同时又不增加偏差。
- **第二个方法如下：对数据进行随机抽样（装袋）的同时，独立树每次分裂时对输入特征也进行随机抽样。**在randomForest包中使用随机抽样数的默认值来对预测特征进行抽样。对于分类问题，默认值为所有预测特征数量的平方根；对于回归问题，默认值为所有预测特征数量除以3。在模型调优过程中，每次树分裂时，算法随机选择的预测特征数量是可变的。通过每次分裂时对特征的随机抽样，可以减轻高度相关的预测特征的影响，这种预测特征在由装袋法生成的独立树中往往起主要作用。独立树彼此之间的相关性减少后，对结果的平均化可以使泛化效果更好，对于异常值影响也更加不敏感，比仅进行装袋的效果要好。

梯度提升 概念

- 梯度提升法的主要思想是，先建立一个某种形式的初始模型（线性、样条、树或其他），称为基学习器；然后检查残差，在残差的基础上围绕损失函数拟合模型。损失函数测量模型和现实之间的差别，例如，在回归问题中可以用误差的平方。一直继续这个过程，直到满足某个特定的结束条件。这与下面的情形有点相似：一个学生进行模拟考试，100道题中错了30道，然后只研究那30道错题；在下次模考中，30道题中又错了10道，然后只研究那10道题，以此类推。如果你想更加深入地研究背后的理论，可以参考Frontiers in Neurorobotics, Gradient boosting machines,a tutorial, Natekin A., Knoll A. (2013)

CONTENT

案例分析

- 在前几章中，我们建立了一些模型，本章的总体目标是看看我们能否提高这些模型的预测能力。对于回归问题，重新回到第5章中的前列腺癌数据集，以0.444的均方误差作为基准，看看能否改善。对于分类问题，我们使用第3章中的乳腺癌数据集和第13章中的皮玛印第安人糖尿病数据集。在乳腺癌数据集中，我们取得了97.6%的预测正确率。在糖尿病数据集中，正确率是79.6%，我们希望再提高一些。随机森林和提升方法都会用于这3个数据集，简单树模型只用于乳腺癌数据集和前列腺癌数据集。

CONTENT

回归树

- 我们直接跳到前列腺癌数据集，但首先要加载所需的R包。先用prostate数据集做回归。使用ifelse()函数将gleason评分编码为指标变量，划分训练数据集和测试数据集，训练数据集为pros.train，测试数据集为pros.test；
- 要在训练数据集上建立回归树，需要使用party包中的rpart()函数，语法与我们在其他建模技术中使用过的非常类似；
- 完成上面的代码后，可以用统计图比较完整树和剪枝树。由partykit包生成的树图明显优于party包生成的，在plot()函数中，你可以简单地使用as.party()函数作为包装器函数。代码如下：

代码如下：

```
####加载程序包
library(rpart) #classification and regression trees
library(partykit) #treeplots
library(MASS) #breast and pima indian data
library(ElemStatLearn) #prostate data
library(randomForest) #random forests
library(xgboost) #gradient boosting
library(caret) #tune hyper-parameters
```

代码如下：

```
data(prostate)
prostate$gleason <- ifelse(prostate$gleason == 6, 0, 1)
pros.train <- subset(prostate, train == TRUE)[, 1:9]
pros.test = subset(prostate, train == FALSE)[, 1:9]

set.seed(123)
tree.pros <- rpart(lpsa ~ ., data = pros.train)
tree.pros$cptable
plotcp(tree.pros)
cp <- min(tree.pros$cptable[5, ])
prune.tree.pros <- prune(tree.pros, cp = cp)
plot(as.party(tree.pros))
plot(as.party(prune.tree.pros))
party.pros.test <- predict(prune.tree.pros,
                           newdata = pros.test)
rpart.resid <- party.pros.test - pros.test$lpsa #calculate residual
mean(rpart.resid^2)
```

分类树

- 对于分类问题，我们先像第3章一样准备好乳腺癌数据。加载数据之后，你要删除患者ID，对特征进行重新命名，删除一些缺失值，然后建立训练数据集和测试数据集。
- 先生成树，然后检查输出中的表格，找到最优分裂次数：
- 交叉验证误差仅在两次分裂后就达到了最小值（见第3行）。现在可以对树进行剪枝，再在图中绘制剪枝树，看看它在测试集上的表现：
- 从对树图的检查中可以发现，细胞大小均匀度是第一个分裂点，第二个是nuclei。完整树还有一个分支是细胞浓度。使用predict()函数并指定type=“class”，在测试集上进行预测。

代码如下：

```
########CART breast cancer
data(biopsy)
biopsy <- biopsy[, -1]
names(biopsy) <- c("thick", "u.size", "u.shape", "adhsn", "s.size", "nucl", "chrom", "n.nuc", "mit", "class")
biopsy.v2 <- na.omit(biopsy)
set.seed(123) #random number generator
ind <- sample(2, nrow(biopsy.v2), replace = TRUE, prob = c(0.7, 0.3))
biop.train <- biopsy.v2[ind == 1, ] #the training data set
biop.test <- biopsy.v2[ind == 2, ] #the test data set
str(biop.test)
set.seed(123)
tree.biop <- rpart(class ~ ., data = biop.train)
tree.biop$cptable
cp <- min(tree.biop$cptable[3, ])
prune.tree.biop = prune(tree.biop, cp <- cp)
# plot(as.party(tree.biop))
plot(as.party(prune.tree.biop))
rparty.test <- predict(prune.tree.biop, newdata = biop.test,
                       type = "class")
table(rparty.test, biop.test$class)
(136+64)/209
```

随机森林回归

- 我们还是首先处理前列腺癌数据集，然后处理乳腺癌数据集和皮玛印第安人糖尿病数据集，使用的程序包是randomForest。建立一个随机森林对象的通用语法是使用randomForest()函数，指定模型公式和数据集这两个基本参数。回想一下每次树迭代默认的变量抽样数，对于回归问题，是p/3；对于分类问题，是p的平方根，p为数据集中预测变量的个数。对于大规模数据集，就p而言，你可以调整mtry参数，它可以确定每次迭代的变量抽样数值。如果p小于10，可以省略上面的调整过程。想在多特征数据集中优化mtry参数时，可以使用caret包，或使用randomForest包中的tuneRF()函数。了解这些情况之后，即可建立随机森林模型并检查模型结果。代码如下所示：

代码如下：

```
#################RF
set.seed(123)
rf.pros <- randomForest(lpsa ~ ., data = pros.train)
rf.pros
plot(rf.pros)
which.min(rf.pros$mse)
set.seed(123)
rf.pros.2 <- randomForest(lpsa ~ ., data = pros.train, ntree = 75)
rf.pros.2
varImpPlot(rf.pros.2, scale = TRUE,
          main = "Variable Importance Plot - PSA Score")
importance(rf.pros.2)
rf.pros.test <- predict(rf.pros.2, newdata = pros.test)
#plot(rf.pros.test, pros.test$lpsa)
rf.resid <- rf.pros.test - pros.test$lpsa #calculate residual
mean(rf.resid^2)
```

随机森林分类

- 你可能会对随机森林回归模型的表现感到失望，但这种技术的真正威力在于解决分类问题。我们从乳腺癌诊断数据开始，这个过程和在回归问题中的做法几乎一样：

代码如下：

```
set.seed(123)
rf.biop <- randomForest(class ~ ., data = biop.train)
rf.biop
plot(rf.biop)
which.min(rf.biop$err.rate[, 1])
set.seed(123)
rf.biop.2 <- randomForest(class ~ ., data = biop.train, ntree = 19)
#getTree(rf.biop,1)
rf.biop.2
rf.biop.test <- predict(rf.biop.2,
                        newdata = biop.test,
                        type = "response")
table(rf.biop.test, biop.test$class)
(139 + 67) / 209
varImpPlot(rf.biop.2)
```

随机森林分类

- 现在，开始应对皮玛印第安人糖尿病模型的艰巨挑战。我们先做好数据准备，代码如下所示：

代码如下：

```
data(Pima.tr)
data(Pima.te)
pima <- rbind(Pima.tr, Pima.te)
set.seed(502)
ind <- sample(2, nrow(pima), replace = TRUE, prob = c(0.7, 0.3))
pima.train <- pima[ind == 1, ]
pima.test <- pima[ind == 2, ]
set.seed(321)
rf.pima <- randomForest(type ~ ., data = pima.train)
rf.pima
# plot(rf.pima)
which.min(rf.pima$err.rate[,1])
set.seed(321)
rf.pima.2 <- randomForest(type ~ ., data = pima.train, ntree = 80)
rf.pima.2
rf.pima.test <- predict(rf.pima.2,
                        newdata = pima.test,
                        type = "response")
table(rf.pima.test, pima.test$type)
(75+33)/147
#varImpPlot(rf.pima.2)
```

极限梯度提升分类

在这一节要使用已经加载的xgboost包。因为这种方法的效果如雷贯耳，所以我们直接用它解决最有挑战性的皮玛印第安人糖尿病问题。正如在提升算法简介中所说，我们要调整一些参数。

（1）nrounds：最大迭代次数（最终模型中树的数量）。
（2）colsample_bytree：建立树时随机抽取的特征数量，用一个比率表示，默认值为1（使用100%的特征）。
（3）min_child_weight：对树进行提升时使用的最小权重，默认为1。
（4）eta：学习率，每棵树在最终解中的贡献，默认为0.3。
（5）gamma：在树中新增一个叶子分区时所需的最小减损。
（6）subsample：子样本数据占整个观测的比例，默认值为1（100%）。
（7）max_depth：单个树的最大深度。

极限梯度提升分类

- 使用expand.grid()函数可以建立实验网格，以运行caret包的训练过程。对于前面列出的参数，如果没有设定具体值，那么即使有默认值，运行函数时也会收到出错信息。下面的参数取值是基于我以前的一些训练迭代而设定的。建议各位亲自实验参数调整过程。使用以下命令建立网格：
- 使用car包的train()函数之前，我要创建一个名为cntrl的对象，来设定trainControl的参数。这个对象会保存我们要使用的方法，以训练调优参数。我们使用5折交叉验证，代码如下所示：

代码如下：

```
########xgboost
#PIMA
grid = expand.grid(
  nrounds = c(75, 100),
  colsample_bytree = 1,
  min_child_weight = 1,
  eta = c(0.01, 0.1, 0.3), #0.3 is default,
  gamma = c(0.5, 0.25),
  subsample = 0.5,
  max_depth = c(2, 3)
)
```

代码如下：

```
grid
cntrl = trainControl(
  method = "cv",
  number = 5,
  verboseIter = TRUE,
  returnData = FALSE,
  returnResamp = "final"
)
set.seed(1)
train.xgb = train(
  x = pima.train[, 1:7],
  y = ,pima.train[, 8],
  trControl = cntrl,
  tuneGrid = grid,
  method = "xgbTree"
)
train.xgb
```

特征选择

- 由此可以得到最优的参数组合来建立模型。模型在训练数据上的正确率是81%，Kappa值是0.55。下面要做的事情有点复杂，但我认为这才是最佳实践。首先创建一个参数列表，供xgboost包的训练函数xgb.train()使用。然后将数据框转换为一个输入特征矩阵，以及一个带标号的数值型结果列表（其中的值是0和1）。接着，将特征矩阵和标号列表组合成符合要求的输入，即一个xgb.Dmatrix对象。代码如下：

代码如下：

```
param <- list(  objective        = "binary:logistic",
                booster          = "gbtree",
                eval_metric      = "error",
                eta              = 0.1,
                max_depth        = 2,
                subsample        = 0.5,
                colsample_bytree   = 1,
                gamma            = 0.5
)

x <- as.matrix(pima.train[, 1:7])
y <- ifelse(pima.train$type == "Yes", 1, 0)
train.mat <- xgb.DMatrix(data = x,
                         label = y)
```

代码如下：

```
set.seed(1)
xgb.fit <- xgb.train(params = param, data = train.mat, nrounds = 75)
xgb.fit
pred <- predict(xgb.fit, x)
# summary(pred)
# head(pred)
# head(y)
impMatrix <- xgb.importance(feature_names = dimnames(x)[[2]], model = xgb.fit)
impMatrix
xgb.plot.importance(impMatrix, main = "Gain by Feature")
```

代码如下：

```
library(InformationValue)
pred <- predict(xgb.fit, x)
optimalCutoff(y, pred)
pima.testMat <- as.matrix(pima.test[, 1:7])
xgb.pima.test <- predict(xgb.fit, pima.testMat)
y.test <- ifelse(pima.test$type == "Yes", 1, 0)
optimalCutoff(y.test, xgb.pima.test)
confusionMatrix(y.test, xgb.pima.test, threshold = 0.39)
1 - misClassError(y.test, xgb.pima.test, threshold = 0.39)
plotROC(y.test, xgb.pima.test)
```

CONTENT

模型选择

- 首先，在具有定量响应变量的前列腺癌数据集上，第5章已经得到了一个线性模型，但我们的努力没有改善这个模型。其次，随机森林在威斯康星乳腺癌数据集上的表现超过了第3章中的逻辑斯蒂回归模型。最后，我必须失望地宣布，在皮玛印第安人糖尿病数据集上，使用提升树不能得到比SVM模型更好的结果。
- 综上，对于前列腺癌数据集和乳腺癌数据集，我们得到了比较好的模型。第15章将介绍神经网络和深度学习的概念，我们会再次尝试改善糖尿病模型。

CONTENT

随机森林法特征选择

- 到现在为止，我们已经介绍了几种特征选择技术，比如正则化、最优子集以及递归特征消除。下面要介绍一种对于分类问题非常有效的特征选择方法，这种方法是通过随机森林实现的，需要使用Boruta包。有一篇论文详细介绍了这种方法如何选择所有相关特征： Kursa M., Rudnicki W. (2010), Feature Selection with the Boruta Package, Journal of Statistical Software, 36(11), 1 - 13
- 在这一节，我会简单介绍这种算法，然后将其应用于一个特征非常多的数据集。这种算法确实可以有效消除不重要的特征，使我们可以集中精力构建一个更简洁、更有效，也更有意义的模型，还是值得花时间学习的。

随机森林法特征选择

在更高的层次上，算法会复制所有输入特征，并对特征中的观测顺序进行重新组合，以去除相关性，从而创建影子特征。然后使用所有输入特征建立一个随机森林模型，并计算每个特征（包括影子特征）的正确率损失均值的Z分数。如果某个特征的Z分数显著高于影子特征的Z分数，那么这个特征就被认为是重要的；反之，这个特征就被认为是不重要的。然后，去除掉影子特征和那些已经确认了重要性的特征，重复上面的过程，直到所有特征都被赋予一个表示重要性的值。你可以设定随机森林迭代的最大次数。算法结束之后，每个初始特征都会被标记为确认、待定或拒绝。对于待定的特征，你必须自己确定是否要包括在下一次建模中。根据具体情况，你可以有以下几种选择：

（1）改变随机数种子，重复运行算法多次（k次），然后只选择那些在k次运行中都标记为“确认”的属性；

（2）将你的训练数据分为k折，在每折数据上分别进行算法迭代，然后选择那些在所有k折数据上都标记为“确认”的属性

随机森林法特征选择

- 请注意，所有这些工作都可以用几行代码完成。下面查看代码，并将其应用于Sonar数据集，这个数据集来自mlbench包。数据集中有208个观测，60个输入特征，以及1个用于分类的标号向量。标号是个因子，如果sonar对象是岩石，标号就是R；如果sonar对象是矿藏，标号则是M。首先加载数据，并快速进行数据探索：

代码如下

```
data(Sonar, package="mlbench")
dim(Sonar)
table(Sonar$Class)

library(Boruta)
set.seed(1)
feature.selection <- Boruta(Class ~ ., data = Sonar, doTrace = 1)
feature.selection$timeTaken
table(feature.selection$finalDecision)

fNames <- getSelectedAttributes(feature.selection) #withTentative = TRUE
fNames
Sonar.features <- Sonar[, fNames]
dim(Sonar.features)
```

小结

- 单个树模型虽易于构建和解释，但对于我们试图解决的多数问题不具备必需的预测能力。要想提高模型的预测能力，建议使用随机森林和梯度提升树。通过随机森林方法可以建立几十棵甚至几百棵树，这些树的结果会聚集成一个综合的预测。随机森林中的每棵树都是使用数据抽样建立的，抽样的方法称为自助法，预测变量也同样进行抽样。对于梯度提升方法，先创建一棵初始的、相对小规模的树，之后，基于残差或误分类会生成一系列树。这种技术的预期结果是建立一系列树，后面的树可以对前面的树的缺点加以改善，最终降低偏差和方差。我们还知道，在R中，可以使用随机森林作为特征选择的方法。
- 尽管这些方法确实非常强大，但在机器学习世界中它们不是万能的。不同的数据集要求分析者根据实际情况判断应该使用什么分析技术。对于分析者来说，技术的选择和调优参数的选择同等重要，有些预测模型是好的，但有些预测模型是伟大的，这种不断调整和细化的过程决定了二者之间的所有区别。

在此输入标题

感谢观看

THANKS

丁香园特邀讲师 周支瑞