## Practical Machine Learning

# week1 学习笔记

相关资源

* The elements of statistical learning
* <https://www.coursera.org/course/ml> 机器学习
* [List of machine learning resources on Quora](http://www.quora.com/Machine-Learning/What-are-some-good-resources-for-learning-about-machine-learning-Why)
* [List of machine learning resources from Science](http://www.sciencemag.org/site/feature/data/compsci/machine_learning.xhtml)
* [Advanced notes from MIT open courseware](http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-867-machine-learning-fall-2006/lecture-notes/)
* [Advanced notes from CMU](http://www.stat.cmu.edu/~cshalizi/350/)
* [Kaggle - machine learning competitions](http://www.kaggle.com/)

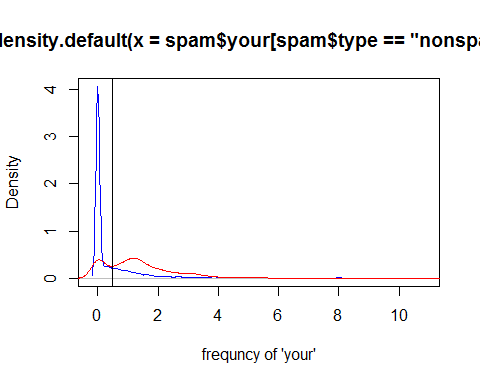
## 1. the components of a predictor

* question
* input data: 注意garbage in => garbage out,more data is better than better models
* features
* algorithm：准确，简单，解释性，快速，可大规模
* parameters
* evaluation e.g.垃圾邮件分类

library(kernlab)  
data(spam)  
head(spam)

比如查看'your'这个词在两类下的分布,并且假设选择0.5的分界线

plot(density(spam$your[spam$type=="nonspam"]),  
 col="blue",xlab="frequncy of 'your'")  
lines(density(spam$your[spam$type=="spam"]),  
 col="red")  
abline(v=0.5)



按照以上的一个简单的分类方法进行分类，效果如下

prediction=ifelse(spam$your>0.5,"spam","nonspam")  
table(spam$type,prediction)

## prediction  
## nonspam spam  
## nonspam 2112 676  
## spam 468 1345

## 2.一些概念吧

### sample error

* in sample error：training set
* out of sample error:test set泛化误差。这个才是真正有用的！

一般来说：in sample error< out of sample error, 会有过拟合的问题。

### study design

* 训练集+测试集+验证集(可无)
* 交叉验证CV

### types of errors

TP,FP,TN,FN这些概念在C6 统计推断那门课中有一个图的展示  
[<http://en.wikipedia.org/wiki/Sensitivity_and_specificity>](http://en.wikipedia.org/wiki/Sensitivity_and_specificity)

课件中的例子讲的挺清楚的。

一些evaluation标准：

* MSE,RMSE，对异常值敏感
* median absolute deviation：比较robust
* sensitivity
* specificity
* accuracy
* concordance：分类数据，比如可以用  
  <http://en.wikipedia.org/wiki/Cohen%27s_kappa>



### ROC曲线

[wiki解释](http://zh.wikipedia.org/wiki/ROC%E6%9B%B2%E7%BA%BF)

在二分类预测中结果是0,1.但是实际模型算出来的一般是一个连续的数，比如[0,1]上的一个数。  
每选择一个cut off point就会有不同的sensitivity和specificity。将这些点在图上表示出来就构成了ROC Curve

* x轴：FP=1-specificity
* y轴：sensitivity，TP

用来平衡sensitivity和specificity，希望两者都大

ROC曲线围成的面积记为AUC,一般来说，面积越大越好

### 交叉验证

* k重交叉验证
* leve one out

### the data set

predict x use data that are related to x.

# week2 学习笔记

## caret package

统一的框架，资源:  
1. caret包自身说明文档  
2. <http://www.edii.uclm.es/~useR-2013/Tutorials/kuhn/user_caret_2up.pdf>

## data slicing

比如createDataPartition，createFolds

library(caret)  
library(kernlab)  
data(spam)  
#将数据按照比例进行训练集和测试集分割  
intrain<-createDataPartition(y=spam$type,p=0.75,list=F)  
training<-spam[intrain,]  
testing<-spam[-intrain,]  
dim(training)

#k-fold  
folds<-createFolds(y=spam$type,k=10,list=T,returnTrain=TRUE)  
str(folds)

folds<-createResample(y=spam$type,times=10,list=T)  
sapply(folds,length)

## 3.training options

仍然以spam为例，假设训练集是75%

library(caret)  
library(kernlab)  
data(spam)  
set.seed(2)  
intrain<-createDataPartition(y=spam$type,p=0.75,list=F)  
training<-spam[intrain,]  
testing<-spam[-intrain,]  
#进行模型训练  
modelfit<-train(type~.,data=training,method="glm")

## 4.plotting predictors

以ISLR包中的wage数据为例

# 导入数据  
library(ISLR)  
library(ggplot2)  
library(caret)

## Loading required package: lattice

data(Wage)  
str(Wage)

分配训练集和测试集

intrain<-createDataPartition(y=Wage$wage,p=0.7,list=F)  
training<-Wage[intrain,]  
testing<-Wage[-intrain,]  
dim(training)

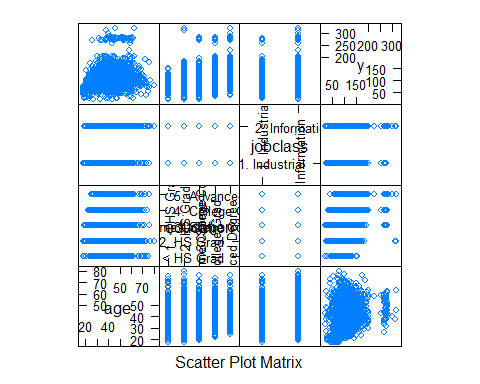
## [1] 2102 12

dim(testing)

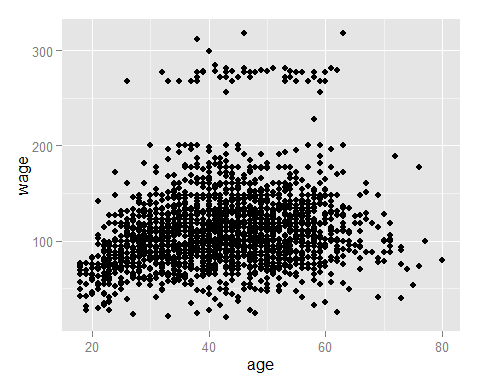
## [1] 898 12

绘图，caret包中有个featurePlot函数

featurePlot(x=training[,c("age","education","jobclass")],  
 y=training$wage,  
 plot="pairs")



qplot(age,wage,data=training)



## 5.preprocessing

preProcess(x, method=)

比如数据标准化  
preProcess(x, method=c("center", "scale"))

也可以直接在train函数中进行数据处理  
train(type~.,data=training,preProcess=c("center","scale"),method="glm")

box-cox变换  
preobj<-preProcess(x, method="BoxCox")  
predict(preobj,x)

缺失值插补  
method="knnImpute","medianImpute"

## 6.解释变量X

dummyVars() 创建哑变量  
nearZeroVar() 去掉只有很小变异的变量

library(splines)  
  
bsbasis<-bs(training$age,df=3) #创建3次多项式  
lm<-lm(wage~bsbasis,data=training)  
plot(training$age,training$wage)  
points(training$age,predict(lm,newdata=training),  
 col="red",pch=19)

## 7.PCA

prcomp()

label=c(1,0,1,0,1)  
x=1:5  
y=rnorm(5)  
#注意color其实是从1开始的，所以直接下面的方式是没有颜色区分的  
plot(x,y,col=label)  
#可以采用的方式是  
plot(x,y,col=label+1)  
plot(x,y,col=as.factor(label))

也可以再preProcess(method="pca")中处理 或者

modelfit<-train(training$type~.,method="glm",  
 preProcess="pca",data=training)  
confusionMatrix(testing$type,predict(modelfit,testing))

## 8.regression

# week3 学习笔记

## 1.决策树

除了之前的rpart,party等专门的包之外，caret中也有统一模式的：

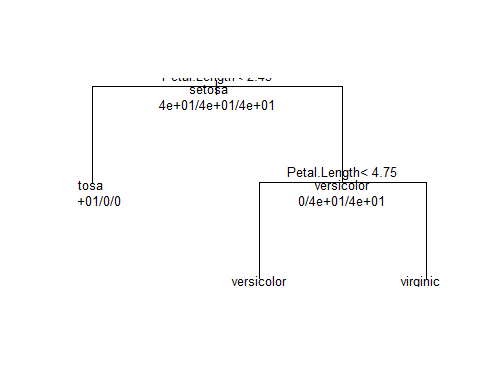
library(caret)  
data(iris)  
#训练集和测试集  
intrain<-createDataPartition(y=iris$Species,p=0.7,list=FALSE)  
training<-iris[intrain,]  
testing<-iris[-intrain,]  
modelfit<-train(Species~.,method="rpart",data=training)

## Loading required package: rpart

#展示  
print(modelfit$finalModel)

## n= 105   
##   
## node), split, n, loss, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 105 70 setosa (0.33333 0.33333 0.33333)   
## 2) Petal.Length< 2.45 35 0 setosa (1.00000 0.00000 0.00000) \*  
## 3) Petal.Length>=2.45 70 35 versicolor (0.00000 0.50000 0.50000)   
## 6) Petal.Length< 4.75 31 1 versicolor (0.00000 0.96774 0.03226) \*  
## 7) Petal.Length>=4.75 39 5 virginica (0.00000 0.12821 0.87179) \*

plot(modelfit$finalModel,uniform=T)  
text(modelfit$finalModel,use.n=T,all=T,cex=0.8)



#预测  
pre\_out<-predict(modelfit,newdata=testing)  
table(pre\_out,testing$Species) # confusionMatrix

##   
## pre\_out setosa versicolor virginica  
## setosa 15 0 0  
## versicolor 0 14 0  
## virginica 0 1 15

train函数主要是用来做分类和回归的，主要的参数有

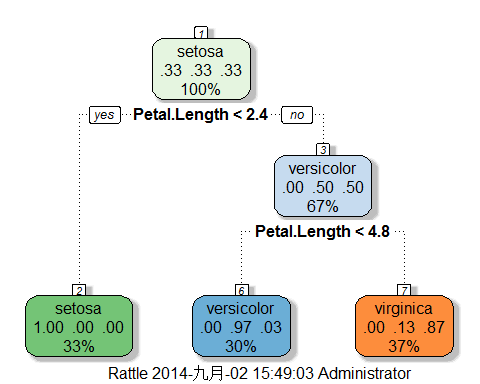
train(x,y,method=,preProcess,...)  
  
- 其中method方法有很多，可以通过`names(getModelInfo())`函数进行显示，并且可以使用自定义函数  
- 最后得到的object所含有的属性主要有：  
finalModel

通过上面的plot画出来的决策树很难看，有个美化版的

library(rattle)

## Rattle: A free graphical interface for data mining with R.  
## XXXX 3.1.0 Copyright (c) 2006-2014 Togaware Pty Ltd.  
## 键入'rattle()'去轻摇、晃动、翻滚你的数据。

fancyRpartPlot(modelfit$finalModel)



#这个界面还不错

### rattle

Rattle是一个用于数据挖掘的R的图形交互界面（GUI），可用于快捷的处理常见的数据挖掘问题.安装很容易:

install.packages("rattle")  
library(rattle)  
#会出现提示  
ratle()  
#接着安装其他所需的东西

## 2.bagging

bootstrap aggregating  
- resample cases and recaculate predictions

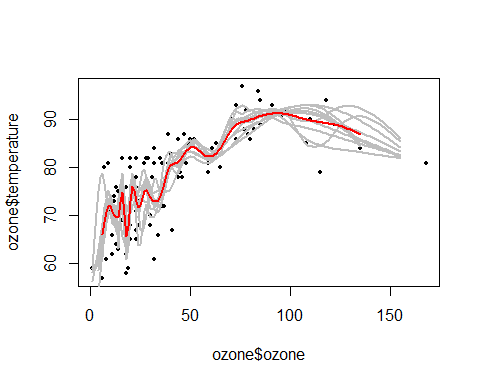
- average or majority vote

### 基本想法说明

#1)示例数据  
library(ElemStatLearn)  
data(ozone)  
ozone=ozone[order(ozone$ozone),]  
head(ozone)

## ozone radiation temperature wind  
## 17 1 8 59 9.7  
## 19 4 25 61 9.7  
## 14 6 78 57 18.4  
## 45 7 48 80 14.3  
## 106 7 49 69 10.3  
## 7 8 19 61 20.1

#2）bagged loess  
ll=matrix(NA,nrow=10,ncol=155)  
for(i in 1:10){  
 ss<-sample(1:dim(ozone)[1],replace=T)  
 ozone0=ozone[ss,];ozone0=ozone0[order(ozone0$ozone),]  
 loess0<-loess(temperature~ozone,data=ozone0,span=0.2)  
 ll[i,]<-predict(loess0,newdata=data.frame(ozone=1:155))  
 }  
#3)绘图展示  
plot(ozone$ozone,ozone$temperature,pch=19,cex=0.5)  
for(i in 1:10){  
 lines(1:155,ll[i,],col="grey",lwd=2)  
}  
lines(1:155,apply(ll,2,mean),col="red",lwd=2)

 caret包中的train函数中的method也有一些是使用bagging方法的，比如：bagEarth，treebag,bagFDA. 或者可以直接使用caret中的bag函数

## 3.random forest

* 1.Bootstrap 抽样
* 2.At each split, bootstrap variables
* 3.Grow multiple trees and vote

#示例  
library(caret)  
data(iris)  
intrain<-createDataPartition(y=iris$Species,p=0.7,list=FALSE)  
training<-iris[intrain,]  
testing<-iris[-intrain,]  
  
#method=rf  
#prox=TRUE是为了??  
modelfit<-train(Species~.,method="rf",data=training,prox=T)

## Loading required package: randomForest  
## randomForest 4.6-10  
## Type rfNews() to see new features/changes/bug fixes.

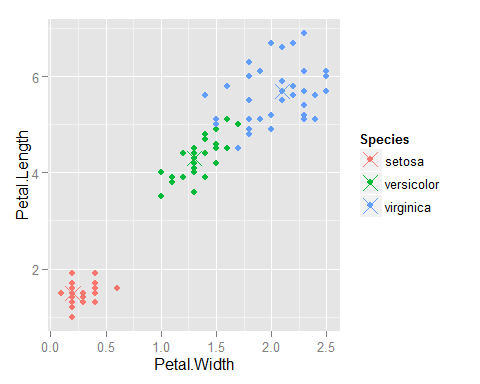
modelfit

## Random Forest   
##   
## 105 samples  
## 4 predictors  
## 3 classes: 'setosa', 'versicolor', 'virginica'   
##   
## No pre-processing  
## Resampling: Bootstrapped (25 reps)   
##   
## Summary of sample sizes: 105, 105, 105, 105, 105, 105, ...   
##   
## Resampling results across tuning parameters:  
##   
## mtry Accuracy Kappa Accuracy SD Kappa SD  
## 2 0.9 0.9 0.03 0.05   
## 3 0.9 0.9 0.03 0.05   
## 4 0.9 0.9 0.03 0.04   
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was mtry = 4.

#get a single tree  
getTree(modelfit$finalModel,k=2)

## left daughter right daughter split var split point status prediction  
## 1 2 3 4 0.80 1 0  
## 2 0 0 0 0.00 -1 1  
## 3 4 5 4 1.75 1 0  
## 4 6 7 3 5.35 1 0  
## 5 8 9 3 4.85 1 0  
## 6 10 11 2 2.25 1 0  
## 7 0 0 0 0.00 -1 3  
## 8 12 13 2 3.00 1 0  
## 9 0 0 0 0.00 -1 3  
## 10 14 15 3 4.50 1 0  
## 11 0 0 0 0.00 -1 2  
## 12 0 0 0 0.00 -1 3  
## 13 0 0 0 0.00 -1 2  
## 14 0 0 0 0.00 -1 2  
## 15 0 0 0 0.00 -1 3

# class centers  
irisP <- classCenter(training[,c(3,4)], training$Species,   
 modelfit$finalModel$prox)  
irisP <- as.data.frame(irisP)  
irisP$Species <- rownames(irisP)  
p <- qplot(Petal.Width, Petal.Length, col=Species,data=training)  
p + geom\_point(aes(x=Petal.Width,y=Petal.Length,col=Species),size=5,shape=4,data=irisP)



也可以直接使用randomForest包中的randomForest函数

## 4.boosting

基本想法：

1. Take lots of (possibly) weak predictors
2. Weight them and add them up
3. Get a stronger predictor 逐步更新分类器

library(ISLR)  
data(Wage)  
library(caret)  
Wage <- subset(Wage,select=-c(logwage))  
inTrain <- createDataPartition(y=Wage$wage,  
 p=0.7, list=FALSE)  
training <- Wage[inTrain,]; testing <- Wage[-inTrain,]  
#model  
  
modFit <- train(wage ~ ., method="gbm",data=training,verbose=FALSE)

print(modFit)

## Stochastic Gradient Boosting   
##   
## 2102 samples  
## 10 predictors  
##   
## No pre-processing  
## Resampling: Bootstrapped (25 reps)   
##   
## Summary of sample sizes: 2102, 2102, 2102, 2102, 2102, 2102, ...   
##   
## Resampling results across tuning parameters:  
##   
## interaction.depth n.trees RMSE Rsquared RMSE SD Rsquared SD  
## 1 50 30 0.3 2 0.03   
## 1 100 30 0.3 2 0.03   
## 1 200 30 0.3 2 0.03   
## 2 50 30 0.3 2 0.03   
## 2 100 30 0.3 2 0.03   
## 2 200 30 0.3 2 0.03   
## 3 50 30 0.3 2 0.03   
## 3 100 30 0.3 2 0.03   
## 3 200 30 0.3 2 0.03   
##   
## Tuning parameter 'shrinkage' was held constant at a value of 0.1  
## RMSE was used to select the optimal model using the smallest value.  
## The final values used for the model were n.trees = 100,  
## interaction.depth = 2 and shrinkage = 0.1.

## 5.model based prediction

假设数据服从某种概率分布，使用贝叶斯定理去识别最优分类器

例如：判别分析 naive bayes：假设独立

示例

train(Species~.,data=training,method="lda") #线性判别分析  
method="nb" #naive bayes

# week4 学习笔记

## 1.regularized regression

岭回归、lasso caret包中method=

## 2.combining predictors

combine predictors的方法：

* 1.bagging,boosting,rf
* 2.combine different classifiers
  + model stacking
  + model ensembling

## 3.forecasting

时间序列 quantmod install.packages("quantmod")

## 4.无监督的预测

先聚类 得到类别label

clue包中的cl\_predict函数