Learning and predicting with statistical models

Kevin R. Coombes
May 6, 2019

Contents

1 Introduction 1

2 Simulated DataSet 1
   2.1 Training Data ................................................. 2
   2.2 Test Data .................................................. 5

3 Feature Selection 8

4 Fitting Models and Making Predictions 10
   4.1 K Nearest Neighbors ........................................ 10
   4.2 Classification and regression trees ....................... 10
   4.3 Linear/Logistic Regression ................................ 12
   4.4 Compound Covariate Prediction ............................ 13
   4.5 Support Vector Machines .................................. 14
   4.6 Neural Networks ............................................ 15
   4.7 Random Forests .............................................. 17

1 Introduction

We start, as usual, by loading the appropriate package:

> library(Modeler)

2 Simulated DataSet

In order to have something to test our models against, we simulate a dataset that has enough underlying structure to make it interesting. First, we set the random seed so that the results will be reproducible.

> set.seed(234843)

Next, we define the simulation parameters. We will simulate a dataset with nFeatures rows representing genes, only nSignif of which are significantly associated with the outcome of interest. We assume that both the training set and test set come from the same population, which is actually a mixture of two types, A and B, where the probability of belonging to type B is given by pB. The significant genes are assumed to be differentially expressed between the two types, with the difference in means following a normal distribution (\( \Delta \sim \text{Norm}(\delta, \sigma) \)).
> nFeatures <- 10000
> nSignif <- 100
> pB <- 0.4
> delta <- 1
> sigma <- 0.3
> nTrain <- 100
> nTest <- 100

For cleanup purposes, we specify the names of things we can safely remove later.

> paramlist <- c("nFeatures", "nSignif", "pB",
+     "delta", "sigma", "nTrain", "nTest")

In addition to simulating the class assignment (A or B), we will also simulate a continuous outcome that represents a probability of belonging to class B. The continuous outcome (Figure 1) will follow a beta distribution with parameters \(\alpha\) and \(\beta\).

> alpha <- 0.75
> beta <- 0.95
> round(100*pbeta(seq(0.1, 0.9, 0.1), alpha, beta), 1)

[1] 17.1 28.8 39.1 48.7 57.7 66.4 74.9 83.2 91.4

> xx <- seq(0, 1, length=300)
> yy <- dbeta(xx, alpha, beta)

Now we can actually start the simulation. For the differentially expressed genes, we make it equally likely that they are higher in A or higher in B.

> signed <- -1 + 2*rbinom(nSignif, 1, 0.5)

As noted above, the magnitude of the difference follows a normal distribution.

> offsets <- c(signed*rnorm(nSignif, delta, sigma), # can change in either direction
+     rep(0, nFeatures - nSignif)) # but most don't change at all

2.1 Training Data

To simulate the training dataset, we first simulate the continuous outcomes (interpreted as the probability of belonging to class B). These are transformed using a logit function so they lie on the entire real line.

> lp <- function(p) log(p/(1-p))
> ea <- function(a) 1/(1+exp(-a))
> pOut <- rbeta(nTrain, alpha, beta)
> trainOutcome <- lp(pOut)

The binary classes for the simulated samples are obtained by dichotomizing the probabilities.

> trainClass <- factor(c("cyan", "magenta")[1 + 1*(pOut > 0.5)])
> summary(trainClass)
Figure 1: Probability of belonging to class B is simulated from this distribution, Beta(0.75, 0.95).
> isB <- trainClass=="magenta"
> summary(isB)

          Mode FALSE TRUE
          logical    57    43

Now we put together the training dataset.

> testData <- matrix(rnorm(nFeatures*nTrain), ncol=nTrain) # pure noise
> testData[,isB] <- sweep(testData[,isB], 1, offsets, "+")
> testData <- t(scale(t(testData)))
> dimnames(testData) <- list(paste("gene", 1:nFeatures, sep='',
+                            paste("trainsamp", 1:nTrain, sep=''))
>
2.2 Test Data

We use the same procedure to simulate the test dataset, starting with continuous outcomes.

```r
> pOut <- rbeta(nTest, alpha, beta)
> testOutcome <- lp(pOut)
```

We convert the continuous outcomes to binary class assignments.
testClass <- factor(c("cyan", "magenta")[(1 + 1*(pOut > 0.5))])

summary(testClass)

cyan magenta
  57   43

isB <- testClass=="magenta"
summary(isB)

Mode FALSE TRUE
logical 57   43

And we then generate the simulated microarray data.

testData <- matrix(rnorm(nFeatures*nTest), ncol=nTest) # pure noise
testData[,isB] <- sweep(testData[,isB], 1, offsets, "+")
testData <- t(scale(t(testData)))
dimnames(testData) <- list(paste("gene", 1:nFeatures, sep=''),
                          +      paste("testsamp", 1:nTest, sep=''))
At this point, we can clean up the work space.

> rm(list=paramlist)
> rm(pOut, isB, signed, offsets)
> rm(xx, yy, alpha, beta)
> rm(paramlist)
3 Feature Selection

Here we implement a simple feature selection scheme. We first perform gene-by-gene t-tests on the training data to identify genes that are differentially expressed between the two classes.

> library(ClassComparison)
> mtt <- MultiTtest(trainData, trainClass)

We then use a beta-uniform-mixture (BUM) model to estimate the false discover rate (FDR).
> bum <- Bum(mtt@p.values)
> countSignificant(bum, alpha=0.01, by="FDR")

[1] 78

> countSignificant(bum, alpha=0.05, by="FDR")

[1] 91

>
geneset <- rownames(trainData)[selectSignificant(bum, alpha=0.05, by="FDR")]
length(geneset)
[1] 91

trainSubset <- trainData[geneset,]
testSubset <- testData[geneset,]

4 Fitting Models and Making Predictions

4.1 K Nearest Neighbors

Note that the KNN method works for binary class prediction, but does not work for regression.

knnFitted <- learn(modeler3NN, trainSubset, trainClass)
knnPredictions <- predict(knnFitted, testSubset)
table(knnPredictions, testClass)

knnPredictions cyan magenta
cyan 57 0
magenta 0 43

knnFitted <- learn(modeler5NN, trainSubset, trainClass)
knnPredictions <- predict(knnFitted, testSubset)
table(knnPredictions, testClass)

knnPredictions cyan magenta
cyan 57 0
magenta 0 43

4.2 Classification and regression trees

Classification

rpartFitted <- learn(modelerRPART, trainSubset, trainClass)
rpartPredictions <- predict(rpartFitted, testSubset, type='class')
table(rpartPredictions, testClass)

rpartPredictions cyan magenta
cyan 49 4
magenta 8 39

Regression

rpartFitted <- learn(modelerRPART, trainSubset, trainOutcome)
rpartPredictions <- predict(rpartFitted, testSubset)
table(rpartPredictions > 0, testClass)
testClass
cyan magenta
FALSE 54 10
TRUE 3 33
> cor(rpartPredictions, testOutcome)
[1] 0.6316475
> temp <- lm(testOutcome ~ rpartPredictions)
4.3 Linear/Logistic Regression

Classification

> # takes too long for the vignette, because of the "step"
> # across glm fits.
> lrFitted <- learn(modelerLR, trainSubset, trainClass)
> lrPredictions <- predict(lrFitted, testSubset)
> table(lrPredictions, testClass)

Regression

> lrFitted <- learn(modelerLR, trainSubset, trainOutcome)
> lrPredictions <- predict(lrFitted, testSubset)
> table(lrPredictions > 0, testClass)

<table>
<thead>
<tr>
<th></th>
<th>cyan</th>
<th>magenta</th>
</tr>
</thead>
<tbody>
<tr>
<td>FALSE</td>
<td>34</td>
<td>19</td>
</tr>
<tr>
<td>TRUE</td>
<td>23</td>
<td>24</td>
</tr>
</tbody>
</table>

> cor(lrPredictions, testOutcome)

[1] 0.12484

> temp <- lm(testOutcome ~ lrPredictions)
4.4 Compound Covariate Prediction

Classification only

> ccpFitted <- learn(modelerCCP, trainSubset, trainClass)
> ccpPredictions <- predict(ccpFitted, testSubset)
> table(ccpPredictions, testClass)
4.5 Support Vector Machines

Classification

> # takes too long for the vignette, because of the "step"
> # across glm fits.
> svmFitted <- learn(modelerSVM, trainSubset, trainClass)
> svmPredictions <- predict(svmFitted, testSubset)
> table(svmPredictions, testClass)

    testClass
  svmPredictions cyan magenta
    cyan   57    0
    magenta  0   43

Regression

> svmFitted <- learn(modelerSVM, trainSubset, trainOutcome)
> svmPredictions <- predict(svmFitted, testSubset)
> table(svmPredictions > 0, testClass)

    testClass
  svmPredictions cyan magenta
    FALSE   57    0
     TRUE    0   43

> cor(svmPredictions, testOutcome)

[1] 0.7775552

> temp <- lm(testOutcome ~ svmPredictions)
4.6 Neural Networks

Classification

> nnetFitted <- learn(modelerNNET, trainSubset, trainClass)

# weights: 466
initial value 77.599783
iter 10 value 0.002918
final value 0.000080
converged

> nnetPredictions <- predict(nnetFitted, testSubset)
> table(nnetPredictions, testClass)

<table>
<thead>
<tr>
<th>nnetPredictions</th>
<th>cyan</th>
<th>magenta</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.40115426339924e-06</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1.40129403562468e-06</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1.4017263397493e-06</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1.40181638348734e-06</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1.4018815155702e-06</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1.40188357795567e-06</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1.40188639659736e-06</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1.40188891908656e-06</td>
<td>50</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>43</td>
</tr>
</tbody>
</table>

Regression

> nnetFitted <- learn(modelerNNET, trainSubset, trainOutcome)

# weights: 466
initial value 528.161386
final value 484.616805
converged

> nnetPredictions <- predict(nnetFitted, testSubset)
> table(nnetPredictions > 0, testClass)

<table>
<thead>
<tr>
<th>testClass</th>
<th>cyan</th>
<th>magenta</th>
</tr>
</thead>
<tbody>
<tr>
<td>FALSE</td>
<td>57</td>
<td>43</td>
</tr>
</tbody>
</table>

> cor(nnetPredictions, testOutcome)

[,1]
[1,] NA

> temp <- lm(testOutcome ~ nnetPredictions)
4.7 Random Forests

Classification

> rfFitted <- learn(modelerRF, trainSubset, trainClass)
> rfPredictions <- predict(rfFitted, testSubset)
> table(rfPredictions, testClass)
testClass
rfPredictions cyan magenta
cyan 57 0
magenta 0 43

Regression

> rfFitted <- learn(modelerRF, trainSubset, trainOutcome)
> rfPredictions <- predict(rfFitted, testSubset)
> table(rfPredictions > 0, testClass)

<table>
<thead>
<tr>
<th></th>
<th>cyan</th>
<th>magenta</th>
</tr>
</thead>
<tbody>
<tr>
<td>FALSE</td>
<td>57</td>
<td>3</td>
</tr>
<tr>
<td>TRUE</td>
<td>0</td>
<td>40</td>
</tr>
</tbody>
</table>

> cor(rfPredictions, testOutcome)

[1] 0.73546

> temp <- lm(testOutcome ~ rfPredictions)