Lab06: Classification (Logistic Regression, LDA, QDA, KNN)

Author: davide.eynard@gmail.com **Notebook:** Didattica
Created: 15 Dec 2014 00:00:57
Updated: 15 Dec 2014 14:40:21

---

0) Generate some data to play with

```
set.seed(20)
# per class observations for training and test sets
trainObs = 50
testObs = 100
trainSize = 2*trainObs
testSize = 2*testObs

# Both training and test are normal distributions, with the same variance, and
slightly overlapped.
# Play with mean and variance and see what happens to classification results
when sets overlap more/less
train.x = rbind(cbind(rnorm(trainObs,5,.8), rnorm(trainObs,5,.8)),
                cbind(rnorm(trainObs,6,.8), rnorm(trainObs,6,.8)))
test.x = rbind(cbind(rnorm(testObs,5,.8), rnorm(testObs,5,.8)),
                cbind(rnorm(testObs,6,.8), rnorm(testObs,6,.8)))

# look at training and test sets
plot(train.x)
points(train.x[(trainObs+1):trainSize,], col='red')
plot(test.x)
points(test.x[(testObs+1):testSize,], col='red')

# build ground truth classes
train.y = rep(0,trainSize)
train.y[(trainObs+1):trainSize] = 1
test.y = rep(0,testSize)
test.y[(testObs+1):testSize] = 1

# build data frames (to be used by the fit and pred functions)
train.df = data.frame(train.x)
test.df = data.frame(test.x)
```

1) Do a recap on Logistic Regression

```
logr.fit = glm(train.y ~ X1 + X2, data = train.df, family = binomial)
```

```
summary(logr.fit)

logr.probs = predict(logr.fit, test.df, type="response")

logr.pred = rep(0,testSize)
logr.pred[logr.probs>.5] = 1

table(logr.pred,test.y)
mean(logr.pred==test.y)
```

2) Comment on results table

		Predicted class		
		− or Null	+ or Non-null	Total
True class	− or Null	True Neg. (TN)	False Pos. (FP)	N
	+ or Non-null	False Neg. (FN)	True Pos. (TP)	P
	Total	N^*	P^*	

TABLE 4.6. *Possible results when applying a classifier or diagnostic test to a population.*

Name	Definition	Synonyms
False Pos. rate	FP/N	Type I error, 1−Specificity
True Pos. rate	TP/P	1−Type II error, power, sensitivity, recall
Pos. Pred. value	TP/P^*	Precision, 1−false discovery proportion
Neg. Pred. value	TN/N^*	

TABLE 4.7. *Important measures for classification and diagnostic testing, derived from quantities in Table 4.6.*

3) Extensions to LDA

```
library(MASS)
?lda
?qda

lda.fit = lda(train.y ~ X1 + X2, data = train.df)
lda.fit
lda.pred = predict(lda.fit, test.df)
lda.class = lda.pred$class
table(lda.class,test.y)
mean(lda.class == test.y)
```

4) Extension to QDA

```
qda.fit = qda(train.y ~  X1 + X2, data = train.df)
qda.fit
```

```
qda.pred = predict(qda.fit, test.df)
qda.class = qda.pred$class
table(qda.class,test.y)
mean(qda.class == test.y)
```

For a geometric interpretation of the covariance matrix, check:
http://www.visiondummy.com/2014/04/geometric-interpretation-covariance-matrix/

5) Extension to KNN

```
library(class)
knn.pred = knn(train.df, test.df, train.y, k = 1)
table(knn.pred,test.y)
mean(knn.pred==test.y)
```

6) Perform an extensive test to compare the different approaches on different datasets

```
logr.errs = NULL
lda.errs = NULL
qda.errs = NULL
knn1.errs = NULL
knn3.errs = NULL

set.seed(20)

for (i in 1:10 ) {
# per class observations for training and test sets
trainObs = 50
testObs = 1000
trainSize = 2*trainObs
testSize = 2*testObs

train.x = rbind(cbind(rnorm(trainObs,5,.8), rnorm(trainObs,5,.8)),
                cbind(rnorm(trainObs,6,.8), rnorm(trainObs,6,.8)))
test.x = rbind(cbind(rnorm(testObs,5,.8), rnorm(testObs,5,.8)),
                cbind(rnorm(testObs,6,.8), rnorm(testObs,6,.8)))

plot(test.x)
points(test.x[(testObs+1):testSize,], col='red')

train.y = rep(0,trainSize)
```

```
train.y[(trainObs+1):trainSize] = 1
test.y = rep(0,testSize)
test.y[(testObs+1):testSize] = 1

train.df = data.frame(train.x)
test.df = data.frame(test.x)

logr.fit = glm(train.y ~ X1 + X2, data = train.df, family = binomial)
lda.fit = lda(train.y ~ X1 + X2, data = train.df)
qda.fit = qda(train.y ~  X1 + X2, data = train.df)

logr.probs = predict(logr.fit, test.df, type="response")
logr.pred = rep(0,testSize)
logr.pred[logr.probs>.5] = 1
logr.errs[i] = mean(logr.pred!=test.y)

lda.pred = predict(lda.fit, test.df)
lda.class = lda.pred$class
lda.errs[i] = mean(lda.class != test.y)

qda.pred = predict(qda.fit, test.df)
qda.class = qda.pred$class
qda.errs[i] = mean(qda.class != test.y)

knn1.pred = knn(train.df, test.df, train.y, k = 1)
knn1.errs[i] = mean(knn1.pred!=test.y)

knn3.pred = knn(train.df, test.df, train.y, k = 3)
knn3.errs[i] = mean(knn3.pred!=test.y)

}


errs = data.frame(cbind(logr.errs, lda.errs, qda.errs, knn1.errs, knn3.errs))
library(psych)
error.bars(errs)
```

7) Extend to a non-linear case

```
logr.errs = NULL
lda.errs = NULL
qda.errs = NULL
knn1.errs = NULL
knn3.errs = NULL
```

```
set.seed(20)

for (i in 1:10 ) {

trainObs = 100
testObs = 100
trainSize = 2*trainObs
testSize = 2*testObs

X1 = rnorm(trainObs,mean=10,sd=.9)
X2 = X1 - 20 + rnorm(trainObs, mean=8, sd=.9)
Y1 = rnorm(trainObs,mean=10,sd=.9)
Y2 = - 2*Y1 + 10 + rnorm(trainObs, mean=11, sd=1.1)

train.x = rbind(cbind(X1,X2), cbind(Y1,Y2))
plot(train.x)
points(train.x[(trainObs+1):trainSize,], col='red')

X1 = rnorm(testObs,mean=10,sd=.9)
X2 = X1 - 20 + rnorm(testObs, mean=8, sd=.9)
Y1 = rnorm(testObs,mean=10,sd=.9)
Y2 = - 2*Y1 + 10 + rnorm(testObs, mean=11, sd=1.1)

test.x = rbind(cbind(X1,X2), cbind(Y1,Y2))
plot(test.x)
points(test.x[(testObs+1):testSize,], col='red')

# build ground truth classes
train.y = rep(0,trainSize)
train.y[(trainObs+1):trainSize] = 1
test.y = rep(0,testSize)
test.y[(testObs+1):testSize] = 1

# build data frames (to be used by the fit and pred functions)
train.df = data.frame(train.x)
test.df = data.frame(test.x)

logr.fit = glm(train.y ~ X1 + X2, data = train.df, family = binomial)
lda.fit = lda(train.y ~ X1 + X2, data = train.df)
qda.fit = qda(train.y ~  X1 + X2, data = train.df)

logr.probs = predict(logr.fit, test.df, type="response")
logr.pred = rep(0,testSize)
logr.pred[logr.probs>.5] = 1
```

```
logr.errs[i] = mean(logr.pred!=test.y)

lda.pred = predict(lda.fit, test.df)
lda.class = lda.pred$class
lda.errs[i] = mean(lda.class != test.y)

qda.pred = predict(qda.fit, test.df)
qda.class = qda.pred$class
qda.errs[i] = mean(qda.class != test.y)

knn1.pred = knn(train.df, test.df, train.y, k = 1)
knn1.errs[i] = mean(knn1.pred!=test.y)

knn3.pred = knn(train.df, test.df, train.y, k = 3)
knn3.errs[i] = mean(knn3.pred!=test.y)

}

errs = data.frame(cbind(logr.errs, lda.errs, qda.errs, knn1.errs, knn3.errs))
library(psych)
error.bars(errs)
```

8) Check out the curse of dimensionality exercise (4.4) and look for the online tutorial for a better explanation:
http://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/