Statistical methods for bioinformatics
Resampling methods

Cedric Lood

April 12, 2016

1 Conceptual exercises

1.1 Question 1

Here is a small reminder of the rules that can be used when dealing with variance and covariance of random variables that are not independent (which is the case of 2 financial instruments):

- $Var(X+Y) = Var(X) + Var(Y) + 2Cov(X,Y)$
- $Var(aX) = a^2 Var(X)$
- $Cov(aX, bY) = abCov(X,Y)$

With these, we can expand the original formulation:

- $Var(\alpha X + (1-\alpha)Y)$
- $Var(\alpha X) + Var((1-\alpha)Y) + 2Cov(\alpha X, (1-\alpha)Y)$
- $\alpha^2 Var(X) + (1-\alpha)^2 Var(Y) + 2\alpha(1-\alpha)Cov(X,Y)$
- $\alpha^2\sigma_X^2 + (1-\alpha)^2\sigma_Y^2 + 2(-\alpha^2+\alpha)\sigma_{XY}$

We will now work to minimize this last expression. For this we need to find the derivative of the expression :

- $f(\alpha) = \alpha^2\sigma_X^2 + (1-\alpha)^2\sigma_Y^2 + 2(-\alpha^2+\alpha)\sigma_{XY}$
- $\frac{df}{d\alpha} = 2\alpha\sigma_X^2 - 2(1-\alpha)\sigma_Y^2 - 2(2\alpha-1)\sigma_{XY}$

Finally, let's solve for the derivative equal to 0 to find the minimum (it is a minimum because the function is quadratic with a leading term (x^2) positive):

- $2\alpha\sigma_X^2 - 2(1-\alpha)\sigma_Y^2 - 2(2\alpha-1)\sigma_{XY} = 0$
- $\alpha\sigma_X^2 - (1-\alpha)\sigma_Y^2 - (2\alpha-1)\sigma_{XY} = 0$ (division by 2)
- $\alpha(\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}) - \sigma_Y^2\sigma_{XY} = 0$ (grouping by α)
- $\alpha = \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}}$
- QED

1.2 Question 4

Bootstrap would be appropriate to estimate the standard error on the prediction. One way to start would be to define a function that receives a dataset, builds a model and returns the prediction.
The function would in a second step be repeatedly called from a bootstrap procedure, with datasets generated using random sampling with replacement. Combining the predictions returned, one can then estimate the standard error.

2 Applied exercises

2.1 Question 5

```
## a
set.seed(1)
attach(Default)
glm.fit.default <- glm(default~income+balance, data=Default, family = binomial)
glm.probs <- predict(glm.fit.default, Default, type="response")
contrasts(default)
glm.pred <- rep("No", 10000)
glm.pred[glm.probs>.5]="Yes"

table(glm.pred, default)

## b
train <- sample(10000, 5000)
glm.fit.vs <- glm(default~income+balance, data=Default, subset=train, family=binomial)
glm.probs.vs <- predict(glm.fit.vs, Default, type="response")[-train]
contrasts(default)
glm.pred.vs <- rep("No", 10000)
glm.pred.vs[glm.probs.vs>.5]="Yes"

table(glm.pred.vs, default)
mean(glm.pred != Default[-train, ]$default)

## c
for (i in 1:3) {
    set.seed(i)
    train <- sample(10000, 5000)
    glm.fit.vs <- glm(default~income+balance, data=Default, subset=train, family=binomial)
    glm.probs.vs <- predict(glm.fit.vs, Default, type="response")[-train]
    glm.pred.vs <- rep("No", 10000)
    glm.pred.vs[glm.probs.vs>.5]="Yes"
    print(mean(glm.pred != Default[-train, ]$default))
}
[1] 0.0472
[1] 0.0474
[1] 0.0452
```

```
## d
for (i in 1:3) {
    set.seed(i)
    train <- sample(10000, 5000)
    glm.fit.vs <- glm(default~income+balance+student, data=Default, subset=train, family=binom
    glm.probs.vs <- predict(glm.fit.vs, Default, type="response")[-train]
    glm.pred.vs <- rep("No", 10000)
    glm.pred.vs[glm.probs.vs>.5]="Yes"
    print(mean(glm.pred != Default[-train, ]$default))
}
[1] 0.0472
[1] 0.0474
[1] 0.0452
```

For c) and d) as can be seen from the output of the scripts (included above), there does not seem to be a reduction of the test error rate when adding the dummy *student* variable.

2.2 Question 6

```
## a
set.seed(1)
glm.fit <- glm(default~income+balance, data=Default, family=binomial)
summary(glm.fit)

## b
boot.fn <- function(data, index){
    model <- glm(default~income+balance, data=data, family=binomial, subset=index)
    return(coef(model))
}

## c
boot(Default, boot.fn, 50)
```

This is the output of the call to the summary function (a) and the boot function (c):

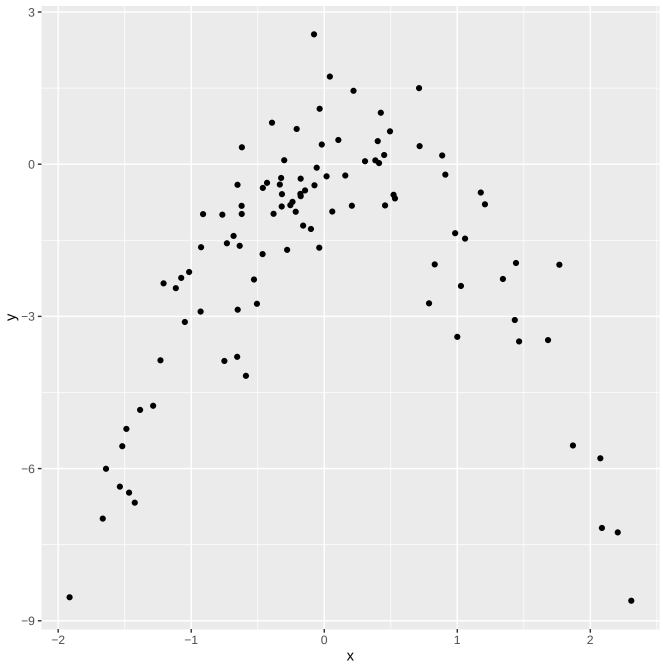
```
#### SUMMARY FUNCTION
Call:
glm(formula = default ~ income + balance, family = binomial,
    data = Default)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.154e+01  4.348e-01 -26.545  < 2e-16 ***
income       2.081e-05  4.985e-06   4.174 2.99e-05 ***
balance      5.647e-03  2.274e-04  24.836  < 2e-16 ***

#### BOOTSTRAP FUNCTION

ORDINARY NONPARAMETRIC BOOTSTRAP
```

```
Call:
boot(data = Default, statistic = boot.fn, R = 50)


Bootstrap Statistics :
         original        bias     std. error
t1* -1.154047e+01  1.181200e-01 4.202402e-01
t2*  2.080898e-05 -5.466926e-08 4.542214e-06
t3*  5.647103e-03 -6.974834e-05 2.282819e-04
```

The estimated standard errors are in both cases similar.

2.3 Question 8

```
## a
set.seed(1)
y <- x - 2 * x^2 + rnorm(100)

## b
qplot(x, y)
ggsave("qplot.pdf")
```



```
## c
quadratic.dataset <- data.frame(x, y)
for (i in 1:4) {
    glm.fit <- glm(y~poly(x,i))
    print(cv.glm(quadratic.dataset, glm.fit)$delta)
}
```

```
[1] 5.890979 5.888812
[1] 1.086596 1.086326
[1] 1.102585 1.102227
[1] 1.114772 1.114334

## d
set.seed(2)
for (i in 1:4) {
    glm.fit <- glm(y~poly(x,i))
    print(cv.glm(quadratic.dataset, glm.fit)$delta)
}
[1] 5.890979 5.888812
[1] 1.086596 1.086326
[1] 1.102585 1.102227
[1] 1.114772 1.114334
```

The results for c) and d) are exactly the same because the LOOCV proceeds by evaluating systematically all the folds by leaving out only one observation at each step.
For e) you can see that the LOOCV test error is the lowest when using the quadratic model. This follows logically from the true form of the (artificial) hypothesis that we started with. Note that in general, you don't have access to the true function.