Homework 6

2019150432 임효진

December 9, 2020

1

(a)

```
library(ISLR)
library(tidyverse)
head(Default)
```

```
##   default student   balance    income
## 1      No      No  729.5265 44361.625
## 2      No     Yes  817.1804 12106.135
## 3      No      No 1073.5492 31767.139
## 4      No      No  529.2506 35704.494
## 5      No      No  785.6559 38463.496
## 6      No     Yes  919.5885  7491.559
```

```
glm_fit=glm(default~income+balance, data = Default,
            family = "binomial")
glm_fit
```

```
##
## Call:  glm(formula = default ~ income + balance, family = "binomial",
##     data = Default)
##
## Coefficients:
## (Intercept)       income      balance
##  -1.154e+01    2.081e-05    5.647e-03
##
## Degrees of Freedom: 9999 Total (i.e. Null);  9997 Residual
## Null Deviance:       2921
## Residual Deviance: 1579  AIC: 1585
```

(b)

```
library(caret)
test_index=createDataPartition(Default$default, times=1,
                               p=0.5, list = F)
```

```
train_set=Default %>% slice(-test_index)
val_set=Default %>% slice(test_index)
```

```
glm_fit=mutate(train_set,
               default=as.numeric(default=="Yes")) %>%
    glm(default~income+balance, data=., family = "binomial")

glm_fit
```

```
##
## Call:  glm(formula = default ~ income + balance, family = "binomial",
##     data = .)
##
## Coefficients:
## (Intercept)       income      balance
##  -1.135e+01    1.202e-05    5.684e-03
##
## Degrees of Freedom: 4998 Total (i.e. Null);  4996 Residual
## Null Deviance:       1457
## Residual Deviance: 778.7     AIC: 784.7
```

```
p_hat=predict(glm_fit, val_set, type="response")
y_hat=ifelse(p_hat>0.5, "Yes", "No") %>% factor()
```

```
mean(y_hat!=val_set$default)
```

```
## [1] 0.02819436
```

(c)

```
test_index=createDataPartition(Default$default, times=1,
                               p=0.5, list = F)
train_set=Default %>% slice(-test_index)
val_set=Default %>% slice(test_index)
glm_fit=mutate(train_set,
               default=as.numeric(default=="Yes")) %>%
    glm(default~income+balance, data=., family = "binomial")
p_hat=predict(glm_fit, val_set, type="response")
y_hat=ifelse(p_hat>0.5, "Yes", "No") %>% factor()
mean(y_hat!=val_set$default)
```

```
## [1] 0.02759448
```

```
test_index=createDataPartition(Default$default, times=1,
                               p=0.5, list = F)
train_set=Default %>% slice(-test_index)
val_set=Default %>% slice(test_index)
glm_fit=mutate(train_set,
               default=as.numeric(default=="Yes")) %>%
```

```
    glm(default~income+balance, data=., family = "binomial")
p_hat=predict(glm_fit, val_set, type="response")
y_hat=ifelse(p_hat>0.5, "Yes", "No") %>% factor()
mean(y_hat!=val_set$default)
```

```
## [1] 0.02539492
```

```
test_index=createDataPartition(Default$default, times=1,
                               p=0.5, list = F)
train_set=Default %>% slice(-test_index)
val_set=Default %>% slice(test_index)
glm_fit=mutate(train_set,
               default=as.numeric(default=="Yes")) %>%
    glm(default~income+balance, data=., family = "binomial")
p_hat=predict(glm_fit, val_set, type="response")
y_hat=ifelse(p_hat>0.5, "Yes", "No") %>% factor()
mean(y_hat!=val_set$default)
```

```
## [1] 0.02659468
```

(d)

```
test_index=createDataPartition(Default$default, times=1,
                               p=0.5, list = F)
train_set=Default %>% slice(-test_index)
val_set=Default %>% slice(test_index)
glm_fit=mutate(train_set,
               default=as.numeric(default=="Yes")) %>%
    glm(default~income+balance+student,
        data=., family = "binomial")
p_hat=predict(glm_fit, val_set, type="response")
y_hat=ifelse(p_hat>0.5, "Yes", "No") %>% factor()
mean(y_hat!=val_set$default)
```

```
## [1] 0.0269946
```

There doesn't seem to be a reduction of test error when the student variable was added.

2

(a)

```
set.seed(1)
glm_fit=glm(default~income+balance, data=Default,
            family="binomial")
summary(glm_fit)
```

```
##
```

```
## Call:
## glm(formula = default ~ income + balance, family = "binomial",
##     data = Default)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4725  -0.1444  -0.0574  -0.0211   3.7245
##
## 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 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1579.0  on 9997  degrees of freedom
## AIC: 1585
##
## Number of Fisher Scoring iterations: 8
```

(b)

```
boot.fn=function(data, index){
    glm_fit=glm(default~income+balance, data=data,
                family = "binomial", subset=index)
    return(coef(glm_fit))
}
```

(c)

```
library(boot)
boot(Default, boot.fn, 1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Default, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##          original        bias     std. error
```

```
## t1* -1.154047e+01 -3.945460e-02 4.344722e-01
## t2*  2.080898e-05  1.680317e-07 4.866284e-06
## t3*  5.647103e-03  1.855765e-05 2.298949e-04
```

(d)

The estimated standard error by using glm() and boot() are similar.

3

(1)

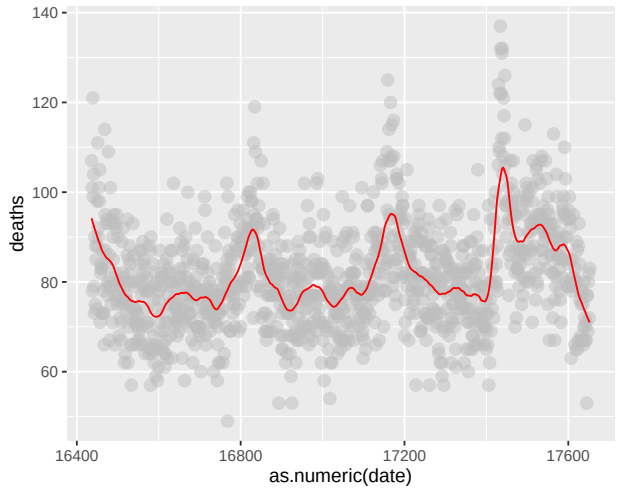
```
library(tidyverse)
library(purrr)
library(pdftools)
library(dslabs)
library(lubridate)

fn <- system.file("extdata", "RD-Mortality-Report_2015-18-180531.pdf",
                  package="dslabs")
dat <- map_df(str_split(pdf_text(fn), "\n"), function(s){
  s <- str_trim(s)
  header_index <- str_which(s, "2015")[1]
  tmp <- str_split(s[header_index], "\\s+", simplify = TRUE)
  month <- tmp[1]
  header <- tmp[-1]
  tail_index  <- str_which(s, "Total")
  n <- str_count(s, "\\d+")
  out <- c(1:header_index, which(n == 1),
           which(n >= 28), tail_index:length(s))
  s[-out] %>%  str_remove_all("[^\\d\\s]") %>% str_trim() %>%
    str_split_fixed("\\s+", n = 6) %>% .[,1:5] %>% as_tibble() %>%
    setNames(c("day", header)) %>%
    mutate(month = month, day = as.numeric(day)) %>%
    gather(year, deaths, -c(day, month)) %>%
    mutate(deaths = as.numeric(deaths))
}) %>%
  mutate(month = recode(month,
                        "JAN" = 1, "FEB" = 2, "MAR" = 3,
                        "APR" = 4, "MAY" = 5, "JUN" = 6,
                        "JUL" = 7, "AGO" = 8, "SEP" = 9,
                        "OCT" = 10, "NOV" = 11, "DEC" = 12)) %>%
  mutate(date = make_date(year, month, day)) %>%
  filter(date <= "2018-05-01")
any(is.na(dat))
```

```
## [1] TRUE
```

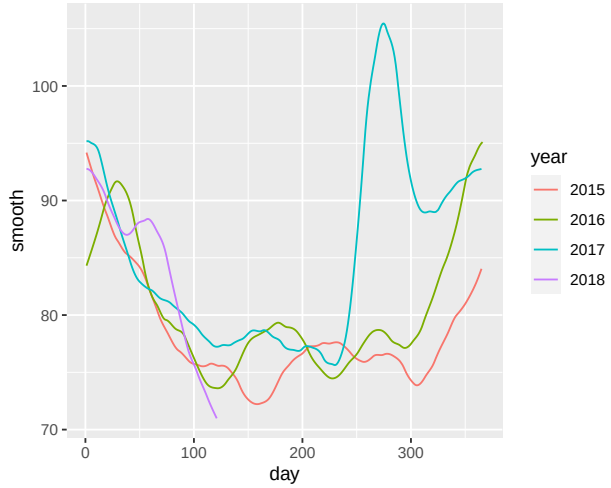
```r
dat=dat[!is.na(dat$deaths), ]
```

```r
total_days=diff(range(as.numeric(dat$date)))
span=62/total_days
fit=loess(deaths~as.numeric(dat$date), degree = 1, span = span,
          data=dat)
dat %>% mutate(smooth=fit$fitted) %>%
    ggplot(aes(as.numeric(date), deaths))+
    geom_point(size=3, alpha=.5, color="grey")+
    geom_line(aes(as.numeric(date), smooth), color="red")
```



(2)

```r
dat %>% mutate(smooth=fit$fitted, day=yday(date)) %>%
    ggplot(aes(day, smooth, col=year))+
    geom_line()
```



4

(1)

```
set.seed(1993)
data("tissue_gene_expression")
tissues <- c("cerebellum", "hippocampus")
ind <- which(tissue_gene_expression$y %in% tissues)
y <- droplevels(tissue_gene_expression$y[ind])
x <- tissue_gene_expression$x[ind, ]
x <- x[, sample(ncol(x), 10)]
```

```
train_lda=train(x, y, method="lda", data=x)
train_lda
```

```
## Linear Discriminant Analysis
##
## 69 samples
## 10 predictors
##  2 classes: 'cerebellum', 'hippocampus'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 69, 69, 69, 69, 69, 69, ...
## Resampling results:
##
##   Accuracy  Kappa
##   0.974834  0.9483923
```

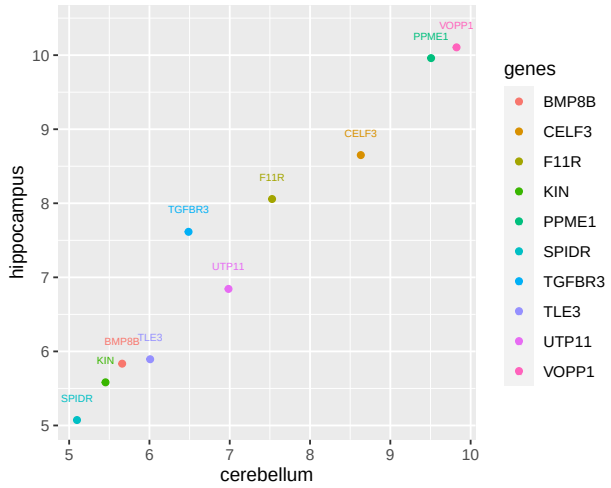
(2)

```
train_lda$finalModel
```

```
## Call:
## lda(x, grouping = y, data = ..1)
##
## Prior probabilities of groups:
##  cerebellum hippocampus
##   0.5507246   0.4492754
##
## Group means:
##                BMP8B    VOPP1     KIN   TGFBR3    PPME1     TLE3    SPIDR
## cerebellum  5.660265  9.82454 5.45312 6.486941 9.507488 6.008919 5.097781
## hippocampus 5.834400 10.10523 5.58260 7.614880 9.960071 5.894071 5.073355
##                 F11R    CELF3    UTP11
## cerebellum  7.527174 8.632449 6.984611
## hippocampus 8.057552 8.650689 6.843593
##
```

```
## Coefficients of linear discriminants:
##                LD1
## BMP8B  -1.6806192
## VOPP1  -0.1903705
## KIN     2.4938547
## TGFBR3  1.9078804
## PPME1   4.0014554
## TLE3   -2.0382325
## SPIDR  -0.8284707
## F11R    2.4599963
## CELF3  -1.0681965
## UTP11  -0.6983432
```

```
means=data.frame(t(train_lda$finalModel$means))
means %>% mutate(genes=row.names(means)) %>%
    ggplot(aes(cerebellum, hippocampus, col=genes))+
    geom_point()+
    geom_text(aes(label=genes), size=2, nudge_y = .3)
```



Genes PPME1 and VOPP1 seem to drive the algorithm.

(3)

```
set.seed(1993)
data("tissue_gene_expression")
ind <- which(tissue_gene_expression$y %in% c("cerebellum", "hippocampus"))
y <- droplevels(tissue_gene_expression$y[ind])
x <- tissue_gene_expression$x[ind, ]
x <- x[, sample(ncol(x), 10)]

train_qda=train(x,y,method="qda", data = x)
train_qda
```

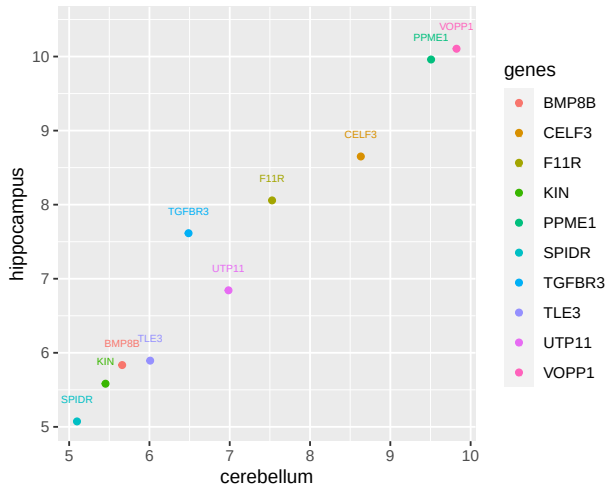
```
## Quadratic Discriminant Analysis
```

```
##
## 69 samples
## 10 predictors
##  2 classes: 'cerebellum', 'hippocampus'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 69, 69, 69, 69, 69, 69, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9256166  0.8544378
```

The accuracy is lower than LDA.

##(4)

```
means=data.frame(t(train_qda$finalModel$means))
means %>% mutate(genes=row.names(means)) %>%
    ggplot(aes(cerebellum, hippocampus, col=genes))+
    geom_point()+
    geom_text(aes(label=genes), size=2, nudge_y = .3)
```



The same genes in LDA are driving the algorithm in QDA too.