Projet - Pénalisation et stabilité

Nadia Ouhssaine

12 mai 2019

Régression linéaire pénalisée

1 - Chargement des données

Le jeu de données choisi est issu du package `curatedOvarianData`. Il s'agit d'un jeu de données de 110 patientes atteintes d'un cancer de l'ovaire séreux au stade avancé et ayant subi une chirurgie primaire et une chimiothérapie à base de platine et taxane. Nous allons tenter de prédire la variable phénotypique "summarygrade". Le grade d'un cancer est défini par l'apparence des cellules cancéreuses comparé à celles des cellules normales. "summarygrade" prend la valeur "high" ou la valeur "low". Plus le grade est bas plus les cellules cancéreuses ont l'air de cellules normales. Et à l'inverse plus le grade est haut plus les cellules cancéreuses ont l'air anormal. La catégorie "low" concerne les cellules cancéreuses de bas grade (1 et 2) et la catégorie "high" concerne les cellules cancéreuses de haut grade (3). (source : https://bioconductor.org/packages/release/data/experiment/manuals/curatedOvarianData/man/curatedOvarianData.pdf http://www.cancer.ca/fr-ca/cancer-information/cancer-type/breast/grading/?region=on)

Chargement des package :

```
source("https://bioconductor.org/biocLite.R")
biocLite("curatedOvarianCancer")
```

```
## Warning: package 'curatedOvarianCancer' is not available (for R version
## 3.4.4)
```

```
biocLite("Biobase")
```

```
## Warning in install.packages(pkgs = doing, lib = lib, ...): installation of
## package 'Biobase' had non-zero exit status
```

```
library('Biobase')
```

```
library("curatedOvarianData")

data(package="curatedOvarianData")
```

Chargement du dataset :

```
data("GSE17260_eset")
```

Récupération des données d'intéret à l'aide de `exprs` et `pData` :

```
expressionData<-exprs(GSE17260_eset)
otherData<-pData(GSE17260_eset@phenoData)
```

Transformation en valeurs binaire (0,1) de la variables phénotypique :

```
Y<-(otherData$summarygrade=="high")*1
```

2 - Mise en place de la procédure de prédiction avec Lasso

Afin de pouvoir tester la performance des procédures qui seront mise en place, il faut préalablement séparer notre jeu de données en un jeu d'apprentissage (2/3 du jeu de données) et un jeu test (le tier restant).

```
ovarian<-data.frame(X=I(t(expressionData)), Y=Y)
```

```
train <- rbinom(length(ovarian$Y),1,2/3)
ovarian.train <- c()
ovarian.test <- c()
ovarian.train$X <- ovarian$X[train==1,]
ovarian.train$Y <- ovarian$Y[train==1]
ovarian.test$X <- ovarian$X[train==0,]
ovarian.test$Y <- ovarian$Y[train==0]
```

On souhaite mettre en place une procédure de régression logistique pénalisée de type Lasso. On rappelle que lors d'une telle régression l'estimation des paramétres β nécessite le paramétre λ qui est la pénalité Lasso. Celle-ci sert à obtenir des régressions parcimonieuse (ie obtenir beaucoup de coefficients nuls). Ainsi dans la régression Lasso plus λ est grand plus les solutions sont parcimonieuse.

Pour choisir la valeur de λ optimal il faut réaliser une validation croisée sur une régression Lasso avec `cv.glmnet`. Cette fonction permet de tester (sous validation croisée) la régression pour plusieurs valeurs de λ. Ainsi le lambda ayant le taux d'erreurs le plus faible sera donc considéré comme notre λ optimal. On obtient notamment ce plus petit λ à l'aide de `cv.glmnet$lambda.min`. On effectue ensuite la régression Lasso sur la base de ce λ dont on a renvoyé la liste des coefficients non nuls.

```
library(glmnet)
```

```
procedureLasso<-function(xtrain,ytrain){
  lassocv<<-cv.glmnet(xtrain,ytrain,family="binomial",
                      type.measure="class",alpha=1)
  bestlasso<<-which(lassocv$lambda==lassocv$lambda.min) #indice de lambda min parmi les lambda
  lasso<<-glmnet(xtrain,ytrain,family="binomial",alpha=1)
  liste<<-names(lasso$beta[abs(lasso$beta[,bestlasso])>0,bestlasso])
  lasso<<-glmnet(xtrain,ytrain,family="binomial",
                 alpha=1,lambda=lassocv$lambda.min)
}
```

```
#Test
Lasso<-procedureLasso(ovarian.train$X,ovarian.train$Y)
print(liste)
```

```
##  [1] "ACAA1"             "B3GALT2"           "B3GAT1"
##  [4] "BARX2"             "BAZ2B"             "BPGM"
##  [7] "BTG2"              "C15orf41"          "CA5B"
## [10] "CALN1"             "CENPP"             "CHTOP"
## [13] "CRYBA1"            "DHFR"              "DOK4"
## [16] "FAM110C"           "FBXO32"            "HS6ST1P1///HS6ST1"
## [19] "HSCB"              "ICA1"              "KPNA5"
## [22] "LRRC31"            "LTBP4"             "MCM6"
## [25] "MEIS2"             "METTL14"           "MGMT"
## [28] "MTMR4"             "NKTR"              "OXSM"
## [31] "PLXNA2"            "PRKAG2"            "SFTPA1"
## [34] "SH3RF1"            "SPANXN3"           "TEX19"
## [37] "THSD7A"            "TMEM41B"           "TSLP"
## [40] "UBE2D3P1"          "YBX2"              "ZNF451"
```

3 - Mise en place de la procédure de prédiction avec Elastic-Net

On souhaite cette fois mettre en place une procédure de régression logistique pénalisée de type Elastic-Net. On rappelle que lors d'une telle régression l'estimation des paramétres β nécessite le paramétre λ qui est la prénalité Elastic-Net, et le paramétre α qui est compris entre 0 et 1 (0, 1 exclus).

Pour choisir les hyperparamétres optimaux de α et λ, nous allons réaliser conjointement des validations croisées en variant la valeur de α. On fera varier α à échelle de 0.1, et on réalise une validation croisée pour chacun des α. On choisira ensuite le alpha ayant le taux d'erreur de prédiction le plus faible. Pour choisir la valeur de λ optimal, on reprend la méthode précédement avec le minimum de λ dans `cv.glmnet`.

```
## Validation croisée pour le choix de alpha
for(i in 1:9){
  assign(paste("elnet", i, sep=""),cv.glmnet(ovarian.train$X,ovarian.train$Y,type.measure="class",alpha=
}
```

```
pred1 <- predict(elnet1,ovarian.test$X, s=elnet1$lambda.min,type="class")
pred2 <- predict(elnet2,ovarian.test$X, s=elnet2$lambda.min,type="class")
pred3 <- predict(elnet3,ovarian.test$X, s=elnet3$lambda.min,type="class")
pred4 <- predict(elnet4,ovarian.test$X ,s=elnet4$lambda.min,type="class")
pred5 <- predict(elnet5,ovarian.test$X, s=elnet5$lambda.min,type="class")
pred6 <- predict(elnet6,ovarian.test$X, s=elnet6$lambda.min,type="class")
pred7 <- predict(elnet7,ovarian.test$X, s=elnet7$lambda.min,type="class")
pred8 <- predict(elnet8,ovarian.test$X, s=elnet8$lambda.min,type="class")
pred9 <- predict(elnet9,ovarian.test$X, s=elnet9$lambda.min,type="class")

(alpha1<-mean(ovarian.test$Y != pred1))
```

```
## [1] 0.3428571
```

```
(alpha2<-mean(ovarian.test$Y != pred2))
```

```
## [1] 0.4
```

```
(alpha3<-mean(ovarian.test$Y != pred3))
```

```
## [1] 0.3428571
```

```
(alpha4<-mean(ovarian.test$Y != pred4))
```

```
## [1] 0.3714286
```

```
(alpha5<-mean(ovarian.test$Y != pred5))
```

```
## [1] 0.4857143
```

```
(alpha6<-mean(ovarian.test$Y != pred6))
```

```
## [1] 0.4857143
```

```
(alpha7<-mean(ovarian.test$Y != pred7))
```

```
## [1] 0.4285714
```

```
(alpha8<-mean(ovarian.test$Y != pred8))
```

```
## [1] 0.5142857
```

```
(alpha9<-mean(ovarian.test$Y != pred9))
```

```
## [1] 0.3428571
```

```
alphamin<-which.min(c(alpha1,alpha2,alpha3,alpha4,alpha5,alpha5,alpha6,alpha7,alpha8,alpha9))/10
```

On crée finalement la fonction de procédure de la régression Elastic-Net avec la valeur optimal alphamin :

```
procedureElnet<-function(xtrain,ytrain){
  elnetcv<<-cv.glmnet(xtrain,ytrain,family="binomial",
                      type.measure="class",alpha=alphamin)
  bestelnet<<-which(elnetcv$lambda==elnetcv$lambda.min) #indice de lambda min parmi les lambda
  elastic<<-glmnet(xtrain,ytrain,family="binomial",alpha=alphamin)
  liste<<-names(elastic$beta[abs(elastic$beta[,bestelnet])>0,bestelnet])
  elastic<<-glmnet(xtrain,ytrain,family="binomial",
                alpha=alphamin,lambda=elnetcv$lambda.min)
}
```

```
#Test
Elasticnet<-procedureElnet(ovarian.train$X,ovarian.train$Y)
print(liste)
```

```
##   [1] "ACAA1"
##   [2] "ADAM11"
##   [3] "ADAM20"
##   [4] "ADAM20P1"
##   [5] "AGR2"
##   [6] "AGR3"
##   [7] "AIP"
##   [8] "AIPL1"
##   [9] "AKAP10"
##  [10] "AKAP8L"
##  [11] "ALG3"
##  [12] "ALPK1"
##  [13] "ANO8"
##  [14] "AP3M2"
##  [15] "APBA2"
##  [16] "APCDD1"
##  [17] "ARFIP2"
##  [18] "ARL15"
##  [19] "ARL4P"
##  [20] "ARSG"
##  [21] "ARSJ"
##  [22] "ATF2"
##  [23] "ATP6V1C1"
##  [24] "ATRX"
##  [25] "B3GALT2"
##  [26] "B3GAT1"
##  [27] "B4GALNT3"
##  [28] "BARX2"
##  [29] "BAZ2B"
##  [30] "BCAS1"
##  [31] "BICD2"
##  [32] "BIVM"
##  [33] "BPGM"
##  [34] "BPIFB3"
##  [35] "BPIFB4"
##  [36] "BRIP1"
##  [37] "BTBD7"
```

```
##  [38] "BTG2"
##  [39] "BTG4"
##  [40] "C11orf95"
##  [41] "C12orf61"
##  [42] "C15orf41"
##  [43] "C1GALT1"
##  [44] "C4orf27"
##  [45] "C9orf3"
##  [46] "CA5B"
##  [47] "CALN1"
##  [48] "CAMP"
##  [49] "CARTPT"
##  [50] "CCDC40"
##  [51] "CDC6"
##  [52] "CDKN1B"
##  [53] "CDR2L"
##  [54] "CEBPB"
##  [55] "CENPCP1"
##  [56] "CENPP"
##  [57] "CEP55"
##  [58] "CERCAM"
##  [59] "CHDH"
##  [60] "CHIAP2"
##  [61] "CHTOP"
##  [62] "CLDND2"
##  [63] "CLSPN"
##  [64] "CLSTN2"
##  [65] "COG2"
##  [66] "CRYBA1"
##  [67] "CTSW"
##  [68] "DBIL5P2"
##  [69] "DCAF7"
##  [70] "DCK"
##  [71] "DDX43"
##  [72] "DECR1"
##  [73] "DHFR"
##  [74] "DHFRP1///DHFR"
##  [75] "DHRS1"
##  [76] "DNAJC21"
##  [77] "DOK4"
##  [78] "DOPEY1"
##  [79] "DPYSL3"
##  [80] "DTL"
##  [81] "DUSP1"
##  [82] "DUTP1"
##  [83] "DYNLT1"
##  [84] "EEF1A1P25"
##  [85] "EEF1B2P5"
##  [86] "ELOVL3"
##  [87] "ESRP2"
##  [88] "FAM105A"
##  [89] "FAM110C"
##  [90] "FAM135A"
##  [91] "FAM160A2"
```

```
##  [92] "FAM225B"
##  [93] "FAM46D"
##  [94] "FAM92B"
##  [95] "FBXO32"
##  [96] "FDPS"
##  [97] "FDPSP4///UBR1"
##  [98] "FHL3"
##  [99] "FMN2"
## [100] "FOS"
## [101] "FOXD4///FOXD4L1///FOXD4L6///FOXD4L3"
## [102] "GATAD2B"
## [103] "GJB7"
## [104] "GNE"
## [105] "GOLGB1"
## [106] "GPR55"
## [107] "GRSF1"
## [108] "HECW1"
## [109] "HEPACAM"
## [110] "HIGD1AP1"
## [111] "HIST1H3G"
## [112] "HPDL"
## [113] "HRSP12"
## [114] "HS6ST1P1///HS6ST1"
## [115] "HSCB"
## [116] "HSPA14"
## [117] "HSPB1P2"
## [118] "HSPD1P8"
## [119] "ICA1"
## [120] "IER2"
## [121] "IFNW1"
## [122] "IL1RAPL1"
## [123] "IL2"
## [124] "IL20"
## [125] "IQCG"
## [126] "IRF5"
## [127] "JOSD2"
## [128] "KCNH2"
## [129] "KCTD11"
## [130] "KCTD12"
## [131] "KIAA0232"
## [132] "KIAA0922"
## [133] "KPNA5"
## [134] "LARP7"
## [135] "LATS1"
## [136] "LGSN"
## [137] "LHX5"
## [138] "LIMS1"
## [139] "LINC00309"
## [140] "LINC00690"
## [141] "LINC00846"
## [142] "LINC01116"
## [143] "LINGO4///RORC"
## [144] "LRAT"
## [145] "LRP5L"
```

```
## [146] "LRRC31"
## [147] "LRRCC1"
## [148] "LTBP4"
## [149] "MAB21L1"
## [150] "MACROD2-IT1"
## [151] "MAGEA1"
## [152] "MAGEA9B///MAGEA9"
## [153] "MAGEB18"
## [154] "MALAT1"
## [155] "MARCH3"
## [156] "MBTD1"
## [157] "MCM4"
## [158] "MCM6"
## [159] "MED22"
## [160] "MEIS2"
## [161] "METTL10"
## [162] "METTL14"
## [163] "MGMT"
## [164] "MIR497HG"
## [165] "MMP26"
## [166] "MND1"
## [167] "MTERFD1"
## [168] "MTMR3"
## [169] "MTMR4"
## [170] "MTNR1B"
## [171] "MTPN"
## [172] "MYL12B"
## [173] "MYO3B"
## [174] "MYSM1"
## [175] "NAAA"
## [176] "NACAP1"
## [177] "NDUFAF4P3"
## [178] "NEK6"
## [179] "NEUROG3"
## [180] "NKIRAS1"
## [181] "NKTR"
## [182] "NKX3-1"
## [183] "NLGN1"
## [184] "NLGN4Y"
## [185] "NMUR2"
## [186] "NOL6"
## [187] "NOP16"
## [188] "NOVA1"
## [189] "NOX5"
## [190] "NPM3"
## [191] "NPY1R"
## [192] "NRIP1"
## [193] "NUP205"
## [194] "NUP62"
## [195] "NUP93"
## [196] "OR3A1"
## [197] "OR7E18P"
## [198] "OXSM"
## [199] "P4HA2"
```

```
## [200] "PAQR3"
## [201] "PAX8-AS1"
## [202] "PC"
## [203] "PCK1"
## [204] "PCNX"
## [205] "PHACTR3"
## [206] "PHF10"
## [207] "PHLDB3"
## [208] "PLEKHG1"
## [209] "PLGLB1"
## [210] "PLIN4"
## [211] "PLXNA2"
## [212] "PNLIPRP2"
## [213] "POLD1"
## [214] "PPFIBP2"
## [215] "PPP1R9A"
## [216] "PRDM10"
## [217] "PRKAG2"
## [218] "PSMG1///BRWD1"
## [219] "PTDSS1"
## [220] "PTDSS2"
## [221] "PXT1"
## [222] "RAB4B-EGLN2///EGLN2"
## [223] "RABEPK"
## [224] "RAC1P7"
## [225] "RAD51"
## [226] "RAD54B"
## [227] "RBM25"
## [228] "RCOR2"
## [229] "RDM1"
## [230] "REV3L"
## [231] "REXO4"
## [232] "RHD"
## [233] "RICTOR"
## [234] "RIN2"
## [235] "RMI2"
## [236] "RNF103"
## [237] "RNF165"
## [238] "RPL18AP8"
## [239] "RPL23AP82///FAM41C"
## [240] "RPL27AP5"
## [241] "RPL34P6///RPL34P34///RPL34"
## [242] "RPS15AP25"
## [243] "RPS4Y1"
## [244] "RPS6KA6"
## [245] "RRP8"
## [246] "RSPH10B///RSPH10B2"
## [247] "RSPO2"
## [248] "SACM1L"
## [249] "SCYL3"
## [250] "SDAD1"
## [251] "SDCCAG8"
## [252] "SEPT3"
## [253] "SESN1"
```

```
## [254] "SFTPA1"
## [255] "SH3RF1"
## [256] "SHKBP1"
## [257] "SLC22A3"
## [258] "SLC25A31"
## [259] "SLC30A4"
## [260] "SLC39A14"
## [261] "SLC9C1"
## [262] "SMAD1"
## [263] "SMG1P1///SMG1///NPIPB3///NPIPB4///NPIPB5"
## [264] "SMG7"
## [265] "SMPD4"
## [266] "SNHG8"
## [267] "SNRPA"
## [268] "SPAG16"
## [269] "SPANXN3"
## [270] "SRR"
## [271] "SSU72P1"
## [272] "STARD6"
## [273] "SYNE4"
## [274] "SYT3"
## [275] "SYTL3"
## [276] "TAS2R42"
## [277] "TBL1XR1"
## [278] "TBP"
## [279] "TBXA2R"
## [280] "TCEB3B"
## [281] "TCF15"
## [282] "TCF19"
## [283] "TEX19"
## [284] "TFF2"
## [285] "TFF3"
## [286] "THOC1"
## [287] "THSD7A"
## [288] "TIPIN"
## [289] "TMED9"
## [290] "TMEM254"
## [291] "TMEM260"
## [292] "TMEM41B"
## [293] "TMEM53"
## [294] "TMEM55A"
## [295] "TOB1"
## [296] "TOP2B"
## [297] "TOP3A"
## [298] "TPM2"
## [299] "TRAK1"
## [300] "TRAV34"
## [301] "TRPV5"
## [302] "TSLP"
## [303] "TSPAN1"
## [304] "TTC17"
## [305] "TTL"
## [306] "TULP4"
## [307] "UBE2D3P1"
```

```
## [308] "UBQLN1"
## [309] "ULK4"
## [310] "UNC5CL"
## [311] "UNC5D"
## [312] "USP47"
## [313] "VASH1"
## [314] "VDAC1"
## [315] "VDAC3"
## [316] "VIL1"
## [317] "VILL"
## [318] "YBX2"
## [319] "YEATS2"
## [320] "YES1"
## [321] "ZBED5"
## [322] "ZNF124"
## [323] "ZNF165"
## [324] "ZNF235"
## [325] "ZNF266"
## [326] "ZNF285"
## [327] "ZNF451"
## [328] "ZNF507"
## [329] "ZNF511"
## [330] "ZNRF3"
## [331] "ZP2"
## [332] "ZWILCH"
```

4 - Comparaison des méthodes avec le jeu test

À l'aide de la fonction `predict` nous allons prédire si les individus du jeu de test ont un grade pour le cancer de 0 (low) ou de 1 (high). Il s'agira ensuite de comparer avec les vrais grade afin d'en estimer l'erreur de prédiction. On affichera également le tableau croisé répertoriant la répartition des individus entre 0 et 1.

```
predlasso <- predict(lassocv,ovarian.test$X,s=c(lassocv$lambda.min),type="class")
print(table(predlasso,ovarian.test$Y))
```

```
##
## predlasso  0  1
##         0 12  8
##         1 11  4
```

```
print(mean(ovarian.test$Y != predlasso))
```

```
## [1] 0.5428571
```

```
predelnet <- predict(elnetcv,ovarian.test$X,s=c(elnetcv$lambda.min),type="class")
print(table(predelnet,ovarian.test$Y))
```

```
##
## predelnet  0  1
##         0 16  6
##         1  7  6
```

```
print(mean(ovarian.test$Y != predelnet))
```

```
## [1] 0.3714286
```

On obtient des taux d'erreurs assez équivalent, aux alentours de 45-50%.

Etude de stabilité

5 - Faire une étude de stabilité

Pour effectuer une étude de stabilité, on génère un certain nombre d'échantillons bootstrap auxquels on applique les procédures de régression linéaire pénalisé. On commencera donc par implémenter une fonction bootstrap qui effectuera un tirage avec remise dans l'échantillon inital. On veillera à ce que la variable phénotypique soit également tiré lorsque l'individu correspondant est tiré.

```
fonction_bootstrap<- function(a,n){
  b<-data.frame(matrix(NA, nrow=110, ncol=2*n))
  for(i in seq(1,2*n,by=2)){
    indice<-sample(1:110,size=110,replace=T)
    b[i]<-a$X[indice,]
    b[i+1]<-a$Y[indice]
  }
return(b)
}
```

```
lassoBoot<-fonction_bootstrap(ovarian,50)
```

```
listeLasso<-c()
for(i in seq(1,50*2,2)){
Lasso<-procedureLasso(lassoBoot[,i],lassoBoot[,i+1])
listeLasso<-c(listeLasso,liste)
}
```

```
tablelasso<-data.frame(sort(table(listeLasso),decreasing = TRUE))
```

On pourrait proposer la signature suivante :

```
signatureLasso<-tablelasso[1:20,1]
signatureLasso
```

```
##  [1] ZNRF3
##  [2] MMP26
##  [3] NOVA1
##  [4] FOXD4///FOXD4L1///FOXD4L6///FOXD4L3
##  [5] MAB21L1
##  [6] RDM1
##  [7] ZNF451
##  [8] APCDD1
##  [9] BAZ2B
## [10] MCM4
## [11] NACAP1
## [12] PNRC1
## [13] AGR3
## [14] DDX43
## [15] ELOVL3
## [16] UBE2D3P1
## [17] ANO8
## [18] EEF1A1P25
## [19] SEPT3
## [20] YBX2
## 897 Levels: ZNRF3 MMP26 NOVA1 ... ZNF876P
```

```
listeElnet<-c()
for(i in seq(1,50*2,2)){
Elasticnet<-procedureElnet(lassoBoot[,i],lassoBoot[,i+1])
listeElnet<-c(listeElnet,liste)
}
```

```
tablelnet<-data.frame(sort(table(listeElnet),decreasing = TRUE))
```

On pourrait proposer la signature suivante :

```
signatureElnet<-tablelnet[1:20,1]
signatureElnet
```

```
##  [1] NOVA1
##  [2] ZNRF3
##  [3] MMP26
##  [4] BAZ2B
##  [5] PNRC1
##  [6] EEF1A1P25
##  [7] APCDD1
##  [8] BTG2
##  [9] ELOVL3
## [10] MCM4
## [11] NACAP1
## [12] RDM1
## [13] ZNF451
## [14] MAB21L1
## [15] FOXD4///FOXD4L1///FOXD4L6///FOXD4L3
## [16] FABP5P2///FABP5P7///FABP5
## [17] FABP5P3///FABP5
## [18] FABP5P7///FABP5
## [19] GINM1
## [20] NKTR
## 5272 Levels: NOVA1 ZNRF3 MMP26 BAZ2B PNRC1 EEF1A1P25 APCDD1 BTG2 ... ZYX
```

6 - Capacité de prédiction des signatures

On constate que plusieurs génes se retrouvent parmi les deux signatures, la signature Elastic Net semble toutefois plus pertinente car les fréquences des génes sélectionnés sont plus importantes. De même parmi les génes sélectionné (et ayant un coefficient non nul) dans la procédure de Lasso et Elastic-Net. On peut donc penser qu'il est possible d'obtenir une signature stable pour le cancer ovarien.