L2Workshop1

Ye

6/28/2017

Linear regression with number of independent predictors from 2 to 500.

$Y_{i,j} = \beta_0 + \beta_1 X_{i,1} + ... + \beta_j X_{i,j} + \epsilon_i; i = 1, ..., 500; j = 2, ..., 500.$

```
set.seed(8394756)
Epsilon = rnorm(500, 0, 1)
X = rnorm(500*500, 0, 2)
dim(X) = c(500, 500)
colnames(X) = paste0("X", 1:500)
slopesSet = runif(500, 1, 3)
Y = sapply(2:500, function(z) 1 + X[, 1:z] %*% slopesSet[1:z] + Epsilon)
```

Relative importance measures

```
m10<-lm(Y~.,data=data.frame(Y=Y[,9],X[,1:10]))
suppressMessages(library(relaimpo))
```

```
## Warning: package 'survey' was built under R version 3.2.5
```

```
(metrics10<-calc.relimp(m10, type = c("lmg", "first", "last","betasq", "pratt")))
```

```
## Response variable: Y
## Total response variance: 158.83
## Analysis based on 500 observations
##
## 10 Regressors:
## X1 X2 X3 X4 X5 X6 X7 X8 X9 X10
## Proportion of variance explained by model: 99.39%
## Metrics are not normalized (rela=FALSE).
##
## Relative importance metrics:
##
##            lmg       last       first     betasq       pratt
## X1  0.19560555 0.19646714 0.194007078 0.20037565 0.197165650
## X2  0.11249716 0.12026966 0.106423431 0.12184251 0.113872288
## X3  0.04166570 0.03134834 0.050541833 0.03192668 0.040170052
## X4  0.08812983 0.08820169 0.089158286 0.08906743 0.089112846
## X5  0.10333322 0.11166911 0.093944252 0.11447436 0.103702498
## X6  0.03482742 0.02530716 0.043800097 0.02557253 0.033467588
## X7  0.06596825 0.07302726 0.059769112 0.07479363 0.066860668
## X8  0.14578162 0.13378483 0.155419968 0.13655158 0.145680617
## X9  0.01225964 0.02395703 0.003158503 0.02446557 0.008790597
## X10 0.19380573 0.19416396 0.193739431 0.19637208 0.195051313
##
## Average coefficients for different model sizes:
##
```

```
##            1X       2Xs       3Xs       4Xs       5Xs       6Xs       7Xs
## X1  2.6422656 2.6469399 2.6516363 2.6563495 2.6610782 2.6658261 2.6706019
## X2  2.1457397 2.1564036 2.1684913 2.1820346 2.1970674 2.2136255 2.2317473
## X3  1.3707301 1.3473586 1.3221964 1.2951695 1.2662020 1.2352158 1.2021308
## X4  1.8596519 1.8550694 1.8515382 1.8490861 1.8477414 1.8475333 1.8484925
## X5  1.8791240 1.9014318 1.9234188 1.9451469 1.9666840 1.9881041 2.0094873
## X6  1.3491917 1.3194464 1.2884986 1.2562690 1.2226737 1.1876238 1.1510252
## X7  1.4799992 1.4949009 1.5108951 1.5280047 1.5462541 1.5656694 1.5862791
## X8  2.5486469 2.5358348 2.5218355 2.5066341 2.4902129 2.4725506 2.4536228
## X9  0.3608987 0.4310316 0.5015341 0.5723897 0.6435849 0.7151097 0.7869572
## X10 2.8116264 2.8127583 2.8139800 2.8153518 2.8169382 2.8188083 2.8210355
##           8Xs       9Xs     10Xs
## X1  2.6754209 2.6803049 2.685284
## X2  2.2514744 2.2728511 2.295926
## X3  1.1668640 1.1293299 1.089440
## X4  1.8506509 1.8540427 1.858704
## X5  2.0309202 2.0524958 2.074314
## X6  1.1127785 1.0727787 1.030915
## X7  1.6081142 1.6312087 1.655600
## X8  2.4334007 2.4118513 2.388937
## X9  0.8591253 0.9316159 1.004436
## X10 2.8236980 2.8268787 2.830665
```

```
slotNames(metrics10)
```

```
##  [1] "var.y"       "R2"          "R2.decomp"   "lmg"         "pmvd"
##  [6] "first"       "last"        "betasq"      "pratt"       "genizi"
## [11] "car"         "lmg.rank"    "pmvd.rank"   "first.rank"  "last.rank"
## [16] "betasq.rank" "pratt.rank"  "genizi.rank" "car.rank"    "lmg.diff"
## [21] "pmvd.diff"   "first.diff"  "last.diff"   "betasq.diff" "pratt.diff"
## [26] "genizi.diff" "car.diff"    "namen"       "nobs"        "ave.coeffs"
## [31] "type"        "rela"        "always"      "alwaysnam"   "groupdocu"
## [36] "call"
```

```
c(sum10.lmg=sum(metrics10@lmg),
  sum10.first=sum(metrics10@first),
  sum10.last=sum(metrics10@last),
  m10.R2=summary(m10)$r.squared)
```

```
##   sum10.lmg sum10.first  sum10.last      m10.R2
##   0.9938741   0.9899620   0.9981962   0.9938741
```

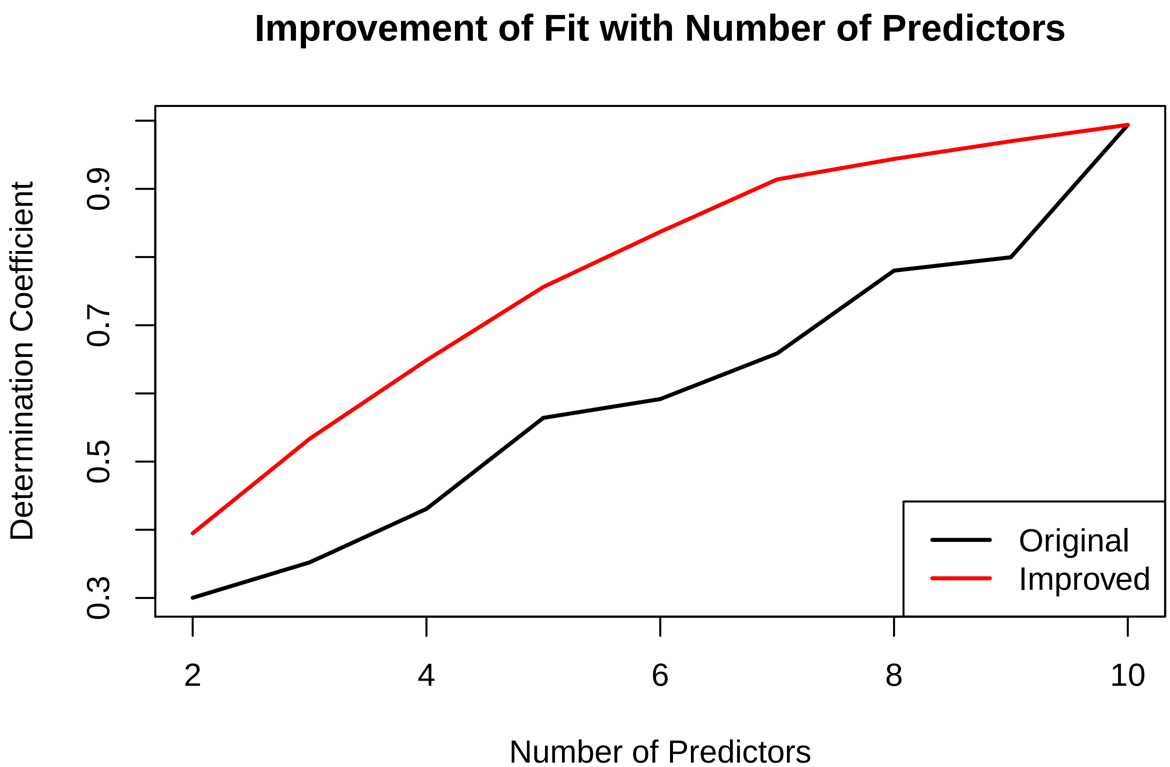
The goal of each measure is to decompose the total R2R2 into contributions by different predictors. The measure lmg is the closest to that target. The measure first typically underestimates R^2. The measure last typically overestimates it.

```
(metrics10.lmg.rank<-metrics10@lmg.rank)
```

```
##  X1  X2  X3  X4  X5  X6  X7  X8  X9 X10
##   1   4   8   6   5   9   7   3  10   2
```

```
orderedPedictors<-X[,1:10][,order(metrics10.lmg.rank)]
originalR2.10<-sapply(2:10,function(z) summary(lm(Y~.,data=data.frame(Y=Y[,9],X[,1:z])))$r.squared)
improvedR2.10<-sapply(2:10,function(z) summary(lm(Y~.,data=data.frame(Y=Y[,9],orderedPedictors[,1:z])))$
matplot(2:10,cbind(originalR2.10,improvedR2.10),type="l",lty=1,lwd=2,col=c("black","red"),
        main="Improvement of Fit with Number of Predictors",
        xlab="Number of Predictors",ylab="Determination Coefficient")
```

```r
legend("bottomright",legend=c("Original","Improved"),lty=1,lwd=2,col=c("black","red"))
```



PCA

We simulate X randomly, why it is possible to run PCA?

Why all the same for lmg, last and first? Because of orthogonality.

Why such significant improvement? Created meta-features (orthogonal predictor).

Can we use more columns than rows for PCA? Why PCA still works? We only need some pairwise correlation coefficients to decompose covariance matrix.

```r
xPCA = prcomp(X[,1:10], center = TRUE, scale. = TRUE)
```