E-Net Comparison

S. Eanes

2/21/2020

Setup data

```r
coral <- read.csv("~/School/Fossil Coral/data/coral_3weighted.csv")

#omit data with no response
coral <- coral[!is.na(coral$U238),]

#select three largest species
genus_trim <- c("Acropora", "Porites")
#remove coral with age > 10
coral <- coral[which(coral$Genus %in% genus_trim),]  %>% dplyr::filter(Age < 10)
coral <- coral[(coral$Calcite <= 1 | is.na(coral$Calcite)),]

#clean up a nice dataframe
coral.df <- coral %>% mutate(Temperature = Temp) %>%
  select(U238,pH,TAlk,Salinity,Temperature,OmegaA,TCO2) %>%
  scale() %>%
  data.frame()
nrow(coral.df)
```

```
## [1] 700
```

Compute Lasso RMSE

```r
##LASSO
lasso <- function() {
  lambda.grid <- exp(seq(-7.5,0,length=100))
  cv.lasso <- cv.glmnet(U238 ~ ., data=coral.df,alpha=1,lambda=lambda.grid,intercept=F)
  #plot(cv.lasso)

  mod.lasso1.grid <- glmnet(U238 ~ ., data=coral.df, alpha=1,lambda=lambda.grid,intercept=F)
  #uncomment and run alone for plots
  #plot_glmnet(mod.lasso1.grid,"lambda",label=8)

  mod.lasso.min <- glmnet(U238 ~ ., data=coral.df,alpha=1,lambda=cv.lasso$lambda.min,intercept=F)
  mod.lasso.1se <- glmnet(U238 ~ ., data=coral.df,alpha=1,lambda=cv.lasso$lambda.1se,intercept=F)
  cv.lasso$lambda.1se; cv.lasso$lambda.min
  p1 <- predict(mod.lasso.min,coral.test)
  p2 <- predict(mod.lasso.1se,coral.test)

  lmse.min <- RMSE(p1,coral.test$U238)
  lmse.1se <- RMSE(p2,coral.test$U238)
  c(lmse.min,lmse.1se)
}
```

Compute Ridge RMSE

```r
#RIDGE
ridge <- function() {
  lambda.grid <- exp(seq(-8,5,length=100))
  cv.ridge <- cv.glmnet(U238 ~ ., data=coral.df,alpha=0,lambda=lambda.grid,intercept=F)
  #plot(cv.ridge)

  mod.ridge1.grid <- glmnet(U238 ~ ., data=coral.df, alpha=0,lambda=lambda.grid,intercept=F)
  #plot_glmnet(mod.ridge1.grid,"lambda",label=8)

  mod.ridge.min <- glmnet(U238 ~ ., data=coral.df,alpha=0,lambda=cv.ridge$lambda.min,intercept=F)
  mod.ridge.1se <- glmnet(U238 ~ ., data=coral.df,alpha=0,lambda=cv.ridge$lambda.1se,intercept=F)
  cv.ridge$lambda.min; cv.ridge$lambda.1se

  p1 <- predict(mod.ridge.min,coral.test)
  p2 <- predict(mod.ridge.1se,coral.test)

  rmse.min <- RMSE(p1,coral.test$U238)
  rmse.1se <- RMSE(p2,coral.test$U238)
  c(rmse.min,rmse.1se)
}
```

Compute Elastic Net ($\alpha = .5$) RMSE

```r
#ELASTIC-NET
enet <- function(a1=.5){
  lambda.grid <- exp(seq(-10,2,length=100))
  cv.net <- cv.glmnet(U238 ~ ., data=coral.df, alpha=a1,lambda=lambda.grid,intercept=F)
  #plot(cv.net)

  mod.enet.grid <- glmnet(U238 ~ ., data=coral.df, alpha=a1,lambda=lambda.grid,intercept=F)
  #plot_glmnet(mod.enet.grid,"lambda",label=8)

  mod.enet.min <- glmnet(U238 ~ ., data=coral.df,alpha=a1,lambda=cv.net$lambda.min,intercept=F)
  mod.enet.1se <- glmnet(U238 ~ ., data=coral.df,alpha=a1,lambda=cv.net$lambda.1se,intercept=F)

  p1 <- predict(mod.enet.min,coral.test)
  p2 <- predict(mod.enet.1se,coral.test)

  emse.min <- RMSE(p1,coral.test$U238)
  emse.1se <- RMSE(p2,coral.test$U238)
  c(emse.min,emse.1se)
}
```

Compute many times

Cross-validation can produce slightly different results from run to run, and the random split of test-validation can also produce differing results. To adjust for this, we do the test-train split 10 times, and compute the above regularized regression rmse 100 times for each test-train split, averaging the results.

```
wrapper_func <- function(){
  lvals <- lasso()
  rvals <- ridge()
  evals <- enet()
  c(lvals,rvals,evals)
}
```

```
n <- 100
resamp <- 10
averages <- data.frame(matrix(rep(0,resamp*6),ncol=6))
colnames(averages) <- c("LMSE.min","LMSE.1se","RMSE.min","RMSE.1se","EMSE.min","EMSE.1se")

for(i in 1:resamp){
  #print(i)
  MSEs <- data.frame(matrix(rep(0,n*6),ncol=6))

  samp <- sample(1:10,nrow(coral.df),replace=T)==i
  coral.test <- coral.df[samp,]
  coral.df <- coral.df[!samp,]

  for(j in 1:n){
    MSEs[j,] <- wrapper_func()
  }
  averages[i,] <- colMeans(MSEs)
}
```

Comparison RMSE

```
sort(colMeans(averages))
```

```
##  RMSE.min  EMSE.min  LMSE.min  EMSE.1se  LMSE.1se  RMSE.1se
## 0.7700257 0.7701087 0.7702796 0.8128418 0.8130233 0.8134544
```

Find an optimal alpha

```
#CV alpha param

lambda.grid <- exp(seq(-10,2,length=100))
```

```
mod.net <- cva.glmnet(U238 ~ ., data=coral.df,lambda=lambda.grid)
ALPHA <- mod.net$alpha
mod.net$modlist[[1]]$cvm
```

```
##   [1] 0.7516836 0.7433004 0.7346112 0.7256574 0.7164840 0.7071381 0.6976664
##   [8] 0.6881146 0.6785247 0.6689354 0.6593801 0.6498883 0.6404854 0.6311951
##  [15] 0.6220383 0.6130364 0.6042122 0.5955912 0.5872036 0.5790793 0.5712514
##  [22] 0.5637539 0.5566213 0.5498860 0.5435754 0.5377086 0.5322983 0.5273538
##  [29] 0.5228677 0.5188318 0.5152302 0.5120319 0.5092163 0.5067444 0.5045834
##  [36] 0.5026988 0.5010558 0.4996211 0.4983623 0.4972513 0.4962587 0.4953623
##  [43] 0.4945423 0.4937793 0.4930591 0.4923709 0.4917005 0.4910466 0.4904027
##  [50] 0.4897676 0.4891363 0.4885185 0.4879095 0.4873197 0.4867530 0.4862149
##  [57] 0.4857111 0.4852506 0.4848353 0.4844725 0.4841651 0.4839133 0.4837190
##  [64] 0.4835836 0.4834997 0.4834681 0.4834799 0.4835341 0.4836251 0.4837431
##  [71] 0.4838878 0.4840459 0.4842174 0.4843995 0.4845824 0.4847679 0.4849480
##  [78] 0.4851241 0.4852928 0.4854523 0.4856067 0.4857472 0.4858799 0.4860032
##  [85] 0.4861155 0.4862162 0.4863135 0.4863971 0.4864746 0.4865462 0.4866093
##  [92] 0.4866687 0.4867166 0.4867636 0.4868047 0.4868407 0.4868750 0.4869024
##  [99] 0.4869307 0.4869531
```

```
a1 <- ALPHA[which.min(sapply(mod.net$modlist, function(mod) {print(cbind(min(mod$cvm),lambda.grid[which
```

```
##           [,1]      [,2]
## [1,] 0.4834681 0.1198862
##           [,1]      [,2]
## [1,] 0.4834679 0.1198862
##           [,1]      [,2]
## [1,] 0.4834658 0.1198862
##           [,1]      [,2]
## [1,] 0.4833944 0.1198862
##           [,1]      [,2]
## [1,] 0.4833922 0.1198862
##           [,1]      [,2]
## [1,] 0.4834076 0.1198862
##           [,1]      [,2]
## [1,] 0.4833796 0.1198862
##          [,1]      [,2]
## [1,] 0.483413 0.1198862
##           [,1]      [,2]
## [1,] 0.4830747 0.1353353
##           [,1]      [,2]
## [1,] 0.4829945 0.1527752
##           [,1]      [,2]
## [1,] 0.4826917 0.1527752
```

```
a1
```

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