Test (easy)

Alessandro De Bettin

20 marzo 2016

Errors and Cross-Validation

First off, I need to write a function that computes the desired test error. It is the following. Since the data I'll consider have only one type of censoring, I decided to use as an input to the function an object of class "Surv" ("survival" package); this is a quick and easy way to handle right censoring. The function outputs the sum of all errors.

```
surv.loss <- function(observed,predicted){ #observed is an object of type Surv
  loss <- rep(0,length(predicted))
  loss[observed[,2] == 0] <- as.numeric(predicted[observed[,2] == 0] <     observed[observed[,2] == 0,1]
  loss[observed[,2] == 1] <- as.numeric((predicted[observed[,2] == 1] < observed[observed[,2] == 1,1]/2
                                          (predicted[observed[,2] == 1] > observed[observed[,2] == 1,1]
  sum(loss)
}
```

```
surv.loss2 <- function(observed,predicted){ #observed is an object of type Surv, predicted is a matrix
  if(is.vector(predicted)) surv.loss(observed,predicted)
  else
  {
    loss <- matrix(0,nrow=nrow(predicted),ncol=ncol(predicted))
    loss[observed[,2] == 0,] <- as.numeric(predicted[observed[,2] == 0,] < observed[observed[,2] == 0,1
    loss[observed[,2] == 1,] <- as.numeric((predicted[observed[,2] == 1,] < observed[observed[,2] == 1,
                                           (predicted[observed[,2] == 1,] > observed[observed[,2] == 
    loss
  }
}
```

"surv.loss2" is the same but can handle matrices of predictions (every column is the prediction of a different model).

I need to write some functions to implement a cross-validation procedure. The first one has the task of splitting the indexes of the observations into "nfold" parts, outputting a list of "nfold" elements each of which is a vector of indexes.

```
cv.index <- function(data,nfold){
  n <- nrow(data)
  split(sample(1:n), rep(1:nfold, length = n))
}
```

The next function computes the cross-validated error for models fitted with the "survreg" function of the "survival" package. The inputs are: the distribution of the response, the data-frame of the explanatory variables, the response variable (of class "Surv") and the number of folds used for the cross-validation. The output is the sum of the errors the procedure makes in each of the "nfold" steps of the cross-validation.

```
surv.cv <- function(dist,X,y,nfold){
  index <- cv.index(cbind(X,y),nfold)
```

```r
  err <- rep(0,nfold)
  for(i in 1:nfold){
    mod <- survreg(y[-index[[i]]]~.,data=X[-index[[i]],],dist=dist)
    err[i] <- surv.loss(y[index[[i]]],predict(mod,newdata=X[index[[i]],]))
  }
  sum(err)
}
```

Then, I need a function for estimating the cross-validated error using the adaptive elastic-net AFT model of the "AdapEnetClass" package. The package has a built-in cross-validation function with mean of squared error loss. The function is the following.

```r
cv.AWEnet
```

```
## function (X, Y, delta, weight, lambda2, maxit, K = 10, fraction = seq(from = 0,
##     to = 1, length = 100), plot.it = F, se = TRUE, AEnet = T,
##     all.folds = NULL)
## {
##     bds <- sort(lambda2)
##     cv.Enet <- function(X, Y, delta, weight, lambda2, maxit,
##         K, fraction, AEnet) {
##         if (is.null(all.folds))
##             all.folds <- cv.folds(length(Y), K)
##         residmat <- matrix(0, length(fraction), K)
##         for (i in seq(K)) {
##             omit <- all.folds[[i]]
##             if (AEnet)
##                 fit <- AEnet.aft(X, Y, delta, weight, lambda2,
##                   maxit)
##             else fit <- WEnet.aft(X, Y, delta, weight, lambda2,
##                 maxit)
##             fit <- predict(fit, X[omit, , drop = FALSE], mode = "fraction",
##                 s = fraction)$fit
##             if (length(omit) == 1)
##                 fit <- matrix(fit, nrow = 1)
##             residmat[, i] <- apply((Y[omit] - fit)^2, 2, mean)
##         }
##         cv <- apply(residmat, 1, mean)
##         cv.error <- sqrt(apply(residmat, 1, var)/K)
##         object <- list(index = fraction, cv = cv, cv.error = cv.error,
##             all.folds = all.folds, mode = "fraction")
##         if (plot.it)
##             plotCVLars(object, se = se)
##         invisible(object)
##     }
##     index <- NULL
##     for (lambda in bds) {
##         if (AEnet) {
##             cvEnet <- cv.Enet(X, Y, delta, weight, lambda, maxit,
##                 K, fraction, AEnet = T)
##             s <- cvEnet$index[which.min(cvEnet$cv)]
##             cv.mse <- which.min(cvEnet$cv)
##             cv.error <- which.min(cvEnet$cv.error)
```

```
##              index <- rbind(index, c(lambda, s, cv.mse, cv.error))
##          }
##          else {
##              cvEnet <- cv.Enet(X, Y, delta, weight, lambda, maxit,
##                  K, fraction, AEnet = F)
##              gama <- cvEnet$index[which.min(cvEnet$cv)]
##              s <- gama * sqrt(1 + lambda)
##              cv.mse <- which.min(cvEnet$cv)
##              cv.error <- which.min(cvEnet$cv.error)
##              index <- rbind(index, c(lambda, s, cv.mse, cv.error))
##          }
##      }
##      list(index = index)
## }
## <environment: namespace:AdapEnetClass>
```

Running the code on the data used in the package example I noticed that the cross-validation would always suggest the same choice for the penalty tuning parameter; that didn't make sense, so I read all the function throughout, and I noticed that at each cross-validation step the model would be estimated on all of the data. Doing so, the function does not compute an estimate of the cross-validated error, but of the training error. So, I modified the function in order to make a proper cross-validation, using the error loss function that I had already written. I marked with a "#" the modified lines.

```
cv.AWEnet2 <- function (X, Y, delta, weight, lambda2, maxit, K = 10, fraction = seq(from = 0,
                                                                                   to = 1, length = 100
                        all.folds = NULL)
{
  require(survival) #
  bds <- sort(lambda2)
  cv.Enet <- function(X, Y, delta, weight, lambda2, maxit,
                      K, fraction, AEnet) {
    if (is.null(all.folds))
      all.folds <- cv.folds(length(Y), K)
    residmat <- matrix(0, length(fraction), K)
    for (i in seq(K)) {
      omit <- all.folds[[i]]
      if (AEnet)
        fit <- AEnet.aft(X[-omit,], Y[-omit], delta[-omit], weight, lambda2, #
                         maxit)
      else fit <- WEnet.aft(X[-omit,], Y[-omit], delta[-omit], weight, lambda2, #
                            maxit)
      fit <- predict(fit, X[omit, , drop = FALSE], mode = "fraction",
                     s = fraction)$fit
      if (length(omit) == 1)
        fit <- matrix(fit, nrow = 1)
      residmat[, i] <- apply(surv.loss2(Surv(Y[omit],delta[omit]),fit),2,sum) #
    }
    cv <- apply(residmat, 1, sum) #
    cv.error <- sqrt(K*apply(residmat, 1, var)) #
    object <- list(index = fraction, cv = cv, cv.error = cv.error,
                   all.folds = all.folds, mode = "fraction")
    if (plot.it)
      plotCVLars(object, se = se)
    invisible(object)
```

```
  }
  index <- NULL
  for (lambda in bds) {
    if (AEnet) {
      cvEnet <- cv.Enet(X, Y, delta, weight, lambda, maxit,
                        K, fraction, AEnet = T)
      s <- cvEnet$index[which.min(cvEnet$cv)]
      cv.mse <- which.min(cvEnet$cv)
      cv.error <- which.min(cvEnet$cv.error)
      index <- rbind(index, c(lambda, s, cv.mse, cv.error))
    }
    else {
      cvEnet <- cv.Enet(X, Y, delta, weight, lambda, maxit,
                        K, fraction, AEnet = F)
      gama <- cvEnet$index[which.min(cvEnet$cv)]
      s <- gama * sqrt(1 + lambda)
      cv.mse <- which.min(cvEnet$cv)
      cv.error <- which.min(cvEnet$cv.error)
      index <- rbind(index, c(lambda, s, cv.mse, cv.error))
    }
  }
  list(index = index,cv=cvEnet$cv[index[3]]) # #cv in order to compare with other models
}
```

My version of the cross-validation function outputs also an estimate of the cross-validation error relative to the model with the optimal tuning parameter, in order to make model comparison easy. As for the cross-validation function for "survreg" models, the error is calculated summing the "nfold" errors.

Examples

Lung dataset

```
library(survival)

dati <- lung

dati$inst <- NULL #not worth it

dati$status <- as.factor(dati$status) #these are factors
dati$sex <- as.factor(dati$sex)
```

The "inst" variable is a factor with many levels, using it as a predictor would not be worth it (the data-set does not contain enough observations) therefore I eliminate it. "status" represents the censoring; even though it is not necessary I prefer to transform it to a factor. "sex" is definitely not a numerical variable. The data-set contains some NAs. Since NAs are not too many, I decide to substitute them with the mean of the variable they are relative to. This should minimize the information loss.

```
dati$ph.ecog[is.na(dati$ph.ecog)] <- mean(dati$ph.ecog, na.rm = TRUE)
dati$ph.karno[is.na(dati$ph.karno)] <- mean(dati$ph.karno, na.rm = TRUE)
dati$pat.karno[is.na(dati$pat.karno)] <- mean(dati$pat.karno, na.rm = TRUE)
dati$meal.cal[is.na(dati$meal.cal)] <- mean(dati$meal.cal, na.rm = TRUE)
dati$wt.loss[is.na(dati$wt.loss)] <- mean(dati$wt.loss, na.rm = TRUE)
```

It is now time to fit some models. First, let's create a "Surv" object containing the observed values.

```
y <- Surv(dati$time,event = dati$status==2)
```

The first model I estimate is a Weibull one.

```
sf1 <- survreg(y~.,data=dati[,-c(1,2)],dist="weibull") #weibull model
summary(sf1)
```

```
##
## Call:
## survreg(formula = y ~ ., data = dati[, -c(1, 2)], dist = "weibull")
##                 Value Std. Error       z        p
## (Intercept)  6.92e+00   0.880741  7.8528 4.07e-15
## age         -8.36e-03   0.006750 -1.2388 2.15e-01
## sex2         4.23e-01   0.122968  3.4406 5.80e-04
## ph.ecog     -4.31e-01   0.128756 -3.3449 8.23e-04
## ph.karno    -1.10e-02   0.006579 -1.6736 9.42e-02
## pat.karno    9.02e-03   0.004937  1.8270 6.77e-02
## meal.cal    -1.14e-05   0.000171 -0.0669 9.47e-01
## wt.loss      7.48e-03   0.004683  1.5977 1.10e-01
## Log(scale)  -3.33e-01   0.061617 -5.4053 6.47e-08
##
## Scale= 0.717
##
## Weibull distribution
## Loglik(model)= -1135.3   Loglik(intercept only)= -1153.9
##  Chisq= 37.15 on 7 degrees of freedom, p= 4.4e-06
## Number of Newton-Raphson Iterations: 5
## n= 228
```

"meal.cal" and "wt.loss"" are non significant, since they bring similar information, I might remove one of them.

```
anova(sf1,survreg(y~.,data=dati[,-c(1,2,8)],dist="weibull")) #I remove meal.cal, the one with more NAs
```

```
##                                                               Terms
## 1 age + sex + ph.ecog + ph.karno + pat.karno + meal.cal + wt.loss
## 2            age + sex + ph.ecog + ph.karno + pat.karno + wt.loss
##   Resid. Df    -2*LL Test Df     Deviance  Pr(>Chi)
## 1       219 2270.552      NA           NA        NA
## 2       220 2270.557    = -1 -0.004460703 0.9467501
```

```
dati$meal.cal <- NULL
```

The anova test suggests that the model fitted without "meal.cal" is statistically equivalent to the former. For the moment I eliminate just one variable: "meal.cal"; I choose this one because it is the variable with the biggest number of NAs.

Now we have a set of explanatory variables, so let's fit various models using the "survreg" function.

Estimates of the cross-validated error are the following.

```r
set.seed(313)
surv.cv("weibull",dati[,-c(1,2)],y,5)
```

```
## [1] 90
```

```r
surv.cv("exponential",dati[,-c(1,2)],y,5)
```

```
## [1] 90
```

```r
surv.cv("loglogistic",dati[,-c(1,2)],y,5) #best model
```

```
## [1] 81
```

```r
surv.cv("lognormal",dati[,-c(1,2)],y,5)
```

```
## [1] 91
```

The log-logistic model seems to be the winner. Let's try to improve the model by removing non significant variables.

```r
sf3 <- survreg(y~.,data=dati[,-c(1,2)],dist="loglogistic")
summary(sf3)
```

```
##
## Call:
## survreg(formula = y ~ ., data = dati[, -c(1, 2)], dist = "loglogistic")
##                 Value Std. Error      z        p
## (Intercept)  5.22185    1.05399  4.954 7.26e-07
## age         -0.00711    0.00743 -0.957 3.39e-01
## sex2         0.50117    0.13459  3.724 1.96e-04
## ph.ecog     -0.28370    0.15659 -1.812 7.00e-02
## ph.karno     0.00196    0.00930  0.211 8.33e-01
## pat.karno    0.01002    0.00519  1.929 5.37e-02
## wt.loss      0.00506    0.00507  0.999 3.18e-01
## Log(scale)  -0.63650    0.06578 -9.676 3.82e-22
##
## Scale= 0.529
##
## Log logistic distribution
## Loglik(model)= -1141.2   Loglik(intercept only)= -1160.9
##  Chisq= 39.38 on 6 degrees of freedom, p= 6e-07
## Number of Newton-Raphson Iterations: 4
## n= 228
```

```r
surv.cv("loglogistic",dati[,-c(1,2,3,6,8)],y,5)
```

```
## [1] 78
```

Now, let's make a confrontation with the adaptive elastic-net model. The initial weights are estimated using the Buckley-James method.

```r
X <- model.matrix(y~.,data=dati[,-c(1,2)])[,-1]

l<-mrbj(cbind(y[,1], y[,2]) ~ X, mcsize=100, trace=FALSE, gehanonly=FALSE)

cv.AWEnet2(X,y[,1],y[,2],l$enet,3,10,5)
```

```
## $index
##      [,1]       [,2] [,3] [,4]
## [1,]    3 0.03030303    4    1
##
## $cv
## [1] 98
```

The model is not better than the log-logistic one.

Ovarian data

```r
dati <- ovarian
```

The description of the data suggests to transform some variables into factors.

```r
dati$fustat <- factor(dati$fustat)
dati$resid.ds <- factor(dati$resid.ds)
dati$rx <- factor(dati$rx)
dati$ecog.ps <- factor(dati$ecog.ps)
```

These are the cross-validated errors for the "survfit" models.

```r
y <- Surv(dati$futime,event = dati$fustat == 1)

surv.cv("weibull",dati[,-c(1,2)],y,10)
```

```
## [1] 11
```

```r
surv.cv("exponential",dati[,-c(1,2)],y,10)
```

```
## [1] 13
```

```r
surv.cv("loglogistic",dati[,-c(1,2)],y,10)
```

```
## [1] 11
```

```r
surv.cv("lognormal",dati[,-c(1,2)],y,10) #winner
```

```
## [1] 9
```

I choose a 10-fold cross-validation. In this case the best model appears to be the log-normal one.

MCLcleaned data

Since the number of predictors is bigger than the number of observations, I can only fit the elastic-net AFT model. The "ID" is not useful in this situation, therefore I remove it.

```
utils::data(MCLcleaned, package="AdapEnetClass")

dati <- MCLcleaned

dati$ID <- NULL

y <- Surv(dati$time,event = dati$cens)

X <- model.matrix(y~.,data=dati[,-c(1,2)])[,-1]
```

The initial weigths are calculated with the "Enet.wls" function. I now estimate the error using a few values for the ridge penalization.

```
wt <- Enet.wls(X,y[,1],y[,2])$beta
lambda2 <- c(15,20,25)
sapply(lambda2, function(x) cv.AWEnet2(X,y[,1],y[,2],wt,x,10)$cv)
```

```
## [1] 80 82 82
```