crossvalidation

yan

2016/11/12

crossvalidation

cross-validation can be used to test the predictive performance of a series of estimated models. In general, a model depend on two kinds of parameters, model parameters θ (such as coefficients of a regression model) and tunning parameters λ (such as indicating the complexity of a regression model). Cross-validation can be used to estimate these parameters simultaneously.

computation

1.get a series of values of tunning parameters $\Lambda = \{\lambda_1, \ldots, \lambda_m\}$

2.divid the obsevation data into k folds

3.treat the ith $(1 \leq i \leq k)$ fold data as the test data and others the traning data

4.using the traning data estimate θ_{ij} uder λ_j seperatly

5.for every λ_j, predictive error is $mspe_j = \frac{1}{k}\sum_{i=1}^{k} mspe_{ij}$

6.minimize $mspe_j$, for example, $mspe_p = \min\{mspe_1, \ldots, mspe_m\}$, then $\theta_{.p}, \lambda_p$ is the best estimation for the parameters.

R code

A big problem we must faced with is over-fitting when using spline to fit the model. crossvalidation is used to choose the best order of spline model as well as the coefficients odf the model.

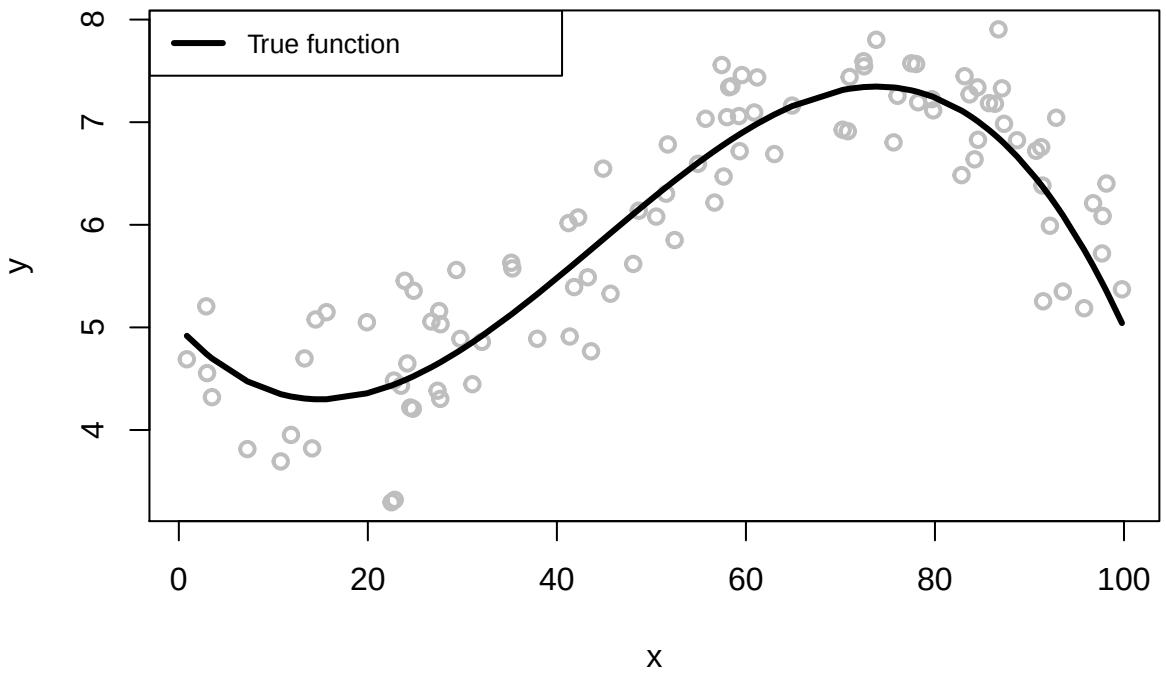
```
#rm(list=ls())
# Generate the training and test samples
seed <- 1809
set.seed(seed)
gen_data <- function(n, beta, sigma_eps) {
  eps <- rnorm(n, 0, sigma_eps)##observation error #generate data from norm distribution N(0,sigma_eps)
  x <- sort(runif(n, 0, 100))##generate data from uniform distribution U(0,100)
  X <- cbind(1, poly(x, degree = (length(beta) - 1), raw = TRUE))##Orthogonal Polynomials
  y <- as.numeric(X %*% beta + eps)

  return(data.frame(x = x, y = y))
}
# Fit the models
require(splines)
```

```
## Loading required package: splines
```

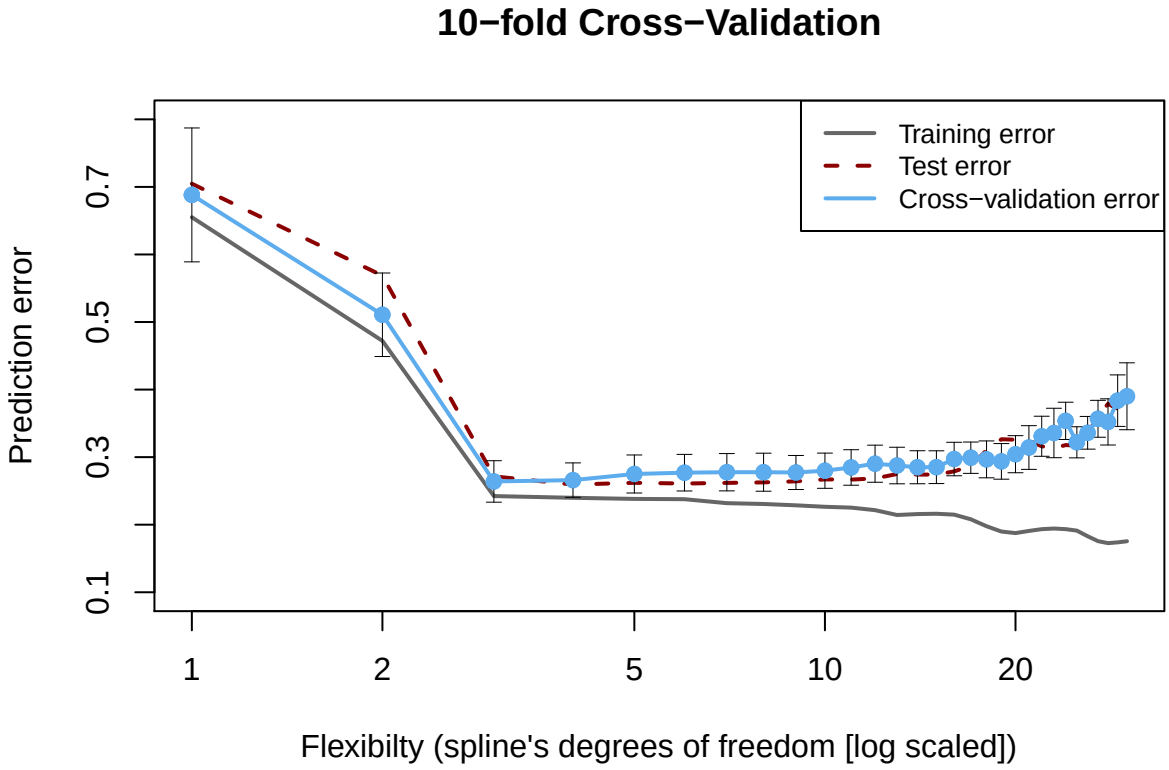
```r
n_rep <- 100
n_df <- 30
df <- 1:n_df
beta <- c(5, -0.1, 0.004, -3e-05)
sigma_eps <- 0.5

##R code-crossvalidation
set.seed(seed)
n_train <- 100
xy <- gen_data(n_train, beta, sigma_eps)
x <- xy$x
y <- xy$y
X <- cbind(1, poly(x, degree = (length(beta) - 1), raw = TRUE))
plot(y ~ x, col = "gray", lwd = 2)
lines(x, X %*% beta, lwd = 3, col = "black")
legend(x = "topleft", legend = c("True function"), lwd = 3, col = "black", text.width = 32, cex = 0.85)
fitted_models <- apply(t(df), 2, function(degf) lm(y ~ ns(x, df = degf)))##fit a natural spline
mse <- sapply(fitted_models, function(obj) deviance(obj)/nobs(obj))

n_test <- 10000
xy_test <- gen_data(n_test, beta, sigma_eps)
pred <- mapply(function(obj, degf) predict(obj, data.frame(x = xy_test$x)), fitted_models, df)
te <- sapply(as.list(data.frame(pred)), function(y_hat) mean((xy_test$y - y_hat)^2))

n_folds <- 10
folds_i <- sample(rep(1:n_folds, length.out = n_train))# devide tranining data into k folds
cv_tmp <- matrix(NA, nrow = n_folds, ncol = length(df))
for (k in 1:n_folds) {
  test_i <- which(folds_i == k)
  train_xy <- xy[-test_i, ]#traning data except row test_i
  test_xy <- xy[test_i, ]##test model using row test_i
  x <- train_xy$x
  y <- train_xy$y
  #for each order(df), estimate the coefficients of the model
  fitted_models <- apply(t(df), 2, function(degf) lm(y ~ ns(x, df = degf)))
  x <- test_xy$x
  y <- test_xy$y
  pred <- mapply(function(obj, degf) predict(obj, data.frame(ns(x, df = degf))),
                 fitted_models, df)
  cv_tmp[k, ] <- sapply(as.list(data.frame(pred)), function(y_hat) mean((y -
                                                                         y_hat)^2))
}
cv <- colMeans(cv_tmp)

library(Hmisc)
```

```
## Loading required package: lattice

## Loading required package: survival

## Loading required package: Formula

## Loading required package: ggplot2
```

```
##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:base':
##
##     format.pval, round.POSIXt, trunc.POSIXt, units
```




```
plot(df, mse, type = "l", lwd = 2, col = gray(0.4), ylab = "Prediction error",
     xlab = "Flexibilty (spline's degrees of freedom [log scaled])", main = paste0(n_folds,
                                                                                   "-fold Cross-Validati
lines(df, te, lwd = 2, col = "darkred", lty = 2)
cv_sd <- apply(cv_tmp, 2, sd)/sqrt(n_folds)
errbar(df, cv, cv + cv_sd, cv - cv_sd, add = TRUE, col = "steelblue2", pch = 19,
       lwd = 0.5)
lines(df, cv, lwd = 2, col = "steelblue2")
points(df, cv, col = "steelblue2", pch = 19)
legend(x = "topright", legend = c("Training error", "Test error", "Cross-validation error"),
       lty = c(1, 2, 1), lwd = rep(2, 3), col = c(gray(0.4), "darkred", "steelblue2"),
       text.width = 0.4, cex = 0.85)
```



References

1. http://www.milanor.net/blog/cross-validation-for-predictive-analytics-using-r/
2. https://en.wikipedia.org/wiki/Cross-validation_(statistics)