P9120 HW2 answer

Guojing Wu / UNI: gw2383

10/14/2019

1. Consider a two-class logistic regression problem with $x \in \mathbb{R}$. Characterize the maximum-likelihood estimates of the slope and intercept parameter if the sample x_i for the two classes are separated by a point $x_0 \in \mathbb{R}$. Generalize this result to (a) $x \in \mathbb{R}^p$ (see Figure 4.16), and (b) more than two classes.

For logistic regression, the log-likelihood looks like: $l(\beta) = \sum_{i=1}^{N}[y_i(\beta_0 + \beta_1 x_i) - log(1 + exp(\beta_0 + \beta_1 x_i))]$. And since the data are separated by a point $x_0 \in \mathbb{R}$, we define:

$$y_i = \begin{cases} 0, & x_i \le x_0 \\ 1, & x_i > x_0 \end{cases}$$

Then we can rewrite the log-likelihood as:

$$\begin{aligned} l(\beta) = & \sum_{i=1, x_i \le x_0}^{N} -log(1 + exp(\beta_0 + \beta_1 x_i)) \\ & + \sum_{i=1, x_i > x_0}^{N} [\beta_0 + \beta_1 x_i - log(1 + exp(\beta_0 + \beta_1 x_i))] \\ = & \sum_{i=1, x_i \le x_0}^{N} -log(1 + exp(\beta_0 + \beta_1 x_0 + \beta_1 (x_i - x_0)) \\ & + \sum_{i=1, x_i > x_0}^{N} [\beta_0 + \beta_1 x_0 + \beta_1 (x_i - x_0) - log(1 + exp(\beta_0 + \beta_1 x_0 + \beta_1 (x_i - x_0)))] \\ = & \sum_{i=1, x_i \le x_0}^{N} -log(1 + exp(\beta_1 (x_i - x_0)) + \sum_{i=1, x_i > x_0}^{N} [\beta_1 (x_i - x_0) - log(1 + exp(\beta_1 (x_i - x_0)))] \end{aligned}$$

Above we let $\beta_0 + \beta_1 x_0 = 0$ to simplify the equation.

For the first part of the equation $\sum_{i=1, x_i \le x_0}^{N} -log(1 + exp(\beta_1 (x_i - x_0)) \le 0$, when $\beta_1 \uparrow$, $exp(\beta_1 (x_i - x_0)) \downarrow$, so $-log(1 + exp(\beta_1 (x_i - x_0)) \uparrow$, this part is monotone increasing along with β_1. For the second part, we let $f(x) = x - log(1 + e^x)$, $f'(x) = 1 - \frac{e^x}{1+e^x} = \frac{1}{1+e^x} > 0$, so the second part is also monotone increasing along with β_1. Hence, the solution for one dimension separable logistic regression can be characterzied by $\beta_1 \to +\infty$ and $\beta_0 = -\beta_1 x_0 \to -sign(x_0)\infty$.

(a)

When generalize this result to $x \in \mathbb{R}^p$, all data are separated by a hyperplane $H_{\omega,b}$, the solution for p dimension separable logistic regression can be characterzied by $||\omega||_2^2 \to +\infty$.

(b)

When generalize this result to more than two classes, e.g., K classes, we then have K-1 hyperplanes H_{ω_i,b_i} that can separate all the data, the solution for K classes separable logistic regression can be characterzied by $||\omega_i||_2^2 \to +\infty$, for $i \in \{1, 2, ..., K-1\}$

2. Show that the truncated power basis functions in (5.3) represent a basis for a cubic spline with the two knots as indicated.

Based on all $h_i(X)$, we write $f(x) = \sum_{m=1}^{6} \beta_m h_m(x)$. Then we need to show the continuity of $f(x)$, $f'(x)$ and $f''(x)$ at knots ξ_1 and ξ_2.

(a) continuity of $f(x)$

left limit of $f(x)$ at ξ_1

$$\begin{aligned} &\forall h > 0 \\ f(\xi_1 - h) &= \beta_1 + \beta_2(\xi_1 - h) + \beta_3(\xi_1 - h)^2 + \beta_4(\xi_1 - h)^3 \\ &\quad + \beta_5(\xi_1 - h - \xi_1)_+^3 + \beta_6(\xi_1 - h - \xi_2)_+^3 \\ &= \beta_1 + \beta_2(\xi_1 - h) + \beta_3(\xi_1 - h)^2 + \beta_4(\xi_1 - h)^3 \\ lim_{h\to 0} f(\xi_1 - h) &= \beta_1 + \beta_2\xi_1 + \beta_3\xi_1^2 + \beta_4\xi_1^3 \end{aligned}$$

right limit of $f(x)$ at ξ_1

$$\begin{aligned} &\forall h > 0 \\ f(\xi_1 + h) &= \beta_1 + \beta_2(\xi_1 + h) + \beta_3(\xi_1 + h)^2 + \beta_4(\xi_1 + h)^3 \\ &\quad + \beta_5(\xi_1 + h - \xi_1)_+^3 + \beta_6(\xi_1 + h - \xi_2)_+^3 \\ &= \beta_1 + \beta_2(\xi_1 + h) + \beta_3(\xi_1 + h)^2 + \beta_4(\xi_1 + h)^3 + \beta_5 h^3 + \beta_6(\xi_1 + h - \xi_2)_+^3 \\ lim_{h\to 0} f(\xi_1 + h) &= \beta_1 + \beta_2\xi_1 + \beta_3\xi_1^2 + \beta_4\xi_1^3 \end{aligned}$$

Left limit = right limit, so $f(x)$ is continuous at ξ_1. In a similar way we can prove that $f(x)$ is also continuous at ξ_2

(b) continuity of $f'(x)$

left limit of $f'(x)$ at ξ_1

$$\begin{aligned} f'_-(\xi_1) &= lim_{h\to 0} \frac{f(\xi_1) - f(\xi_1 - h)}{h} \\ &= lim_{h\to 0} \frac{\beta_1 + \beta_2\xi_1 + \beta_3\xi_1^2 + \beta_4\xi_1^3 - \beta_1 - \beta_2(\xi_1 - h) - \beta_3(\xi_1 - h)^2 - \beta_4(\xi_1 - h)^3}{h} \\ &= lim_{h\to 0} \frac{\beta_2 h + 2\beta_3\xi_1 h + 3\beta_4\xi_1^2 h + O(h^2) + O(h^3)}{h} \\ &= \beta_2 + 2\beta_3\xi_1 + 3\beta_4\xi_1^2 \end{aligned}$$

right limit of $f'(x)$ at ξ_1

$$
\begin{aligned}
f'_+(\xi_1) &= lim_{h\to 0}\frac{f(\xi_1+h)-f(\xi_1)}{h} \\
&= lim_{h\to 0}\frac{\beta_1+\beta_2(\xi_1+h)+\beta_3(\xi_1+h)^2+\beta_4(\xi_1+h)^3+\beta_5 h^3+\beta_6(\xi_1+h-\xi_2)^3_+-\beta_1-\beta_2\xi_1-\beta_3\xi_1^2-\beta_4\xi_1^3}{h} \\
&= lim_{h\to 0}\frac{\beta_2 h+2\beta_3\xi_1 h+3\beta_4\xi_1^2 h+\beta_6(\xi_1+h-\xi_2)^3_+ + O(h^2)+O(h^3)}{h} \\
&= \beta_2+2\beta_3\xi_1+3\beta_4\xi_1^2
\end{aligned}
$$

Left limit = right limit, so $f'(x)$ is continuous at ξ_1. In a similar way we can prove that $f'(x)$ is also continuous at ξ_2

(c) continuity of $f''(x)$

In a similar way we can prove that $f''(x)$ is continuous at ξ_1 and ξ_2, $f''_-(\xi_1) = f''_+(\xi_1) = 6\beta_4\xi_1^2$.

Hence we proved that power basis functions in (5.3) represent a basis for a cubic spline with the two knots.

3. A simulation study

(a) Generate a vector x consisting of 50 points drawn at random from Uniform[0,1]

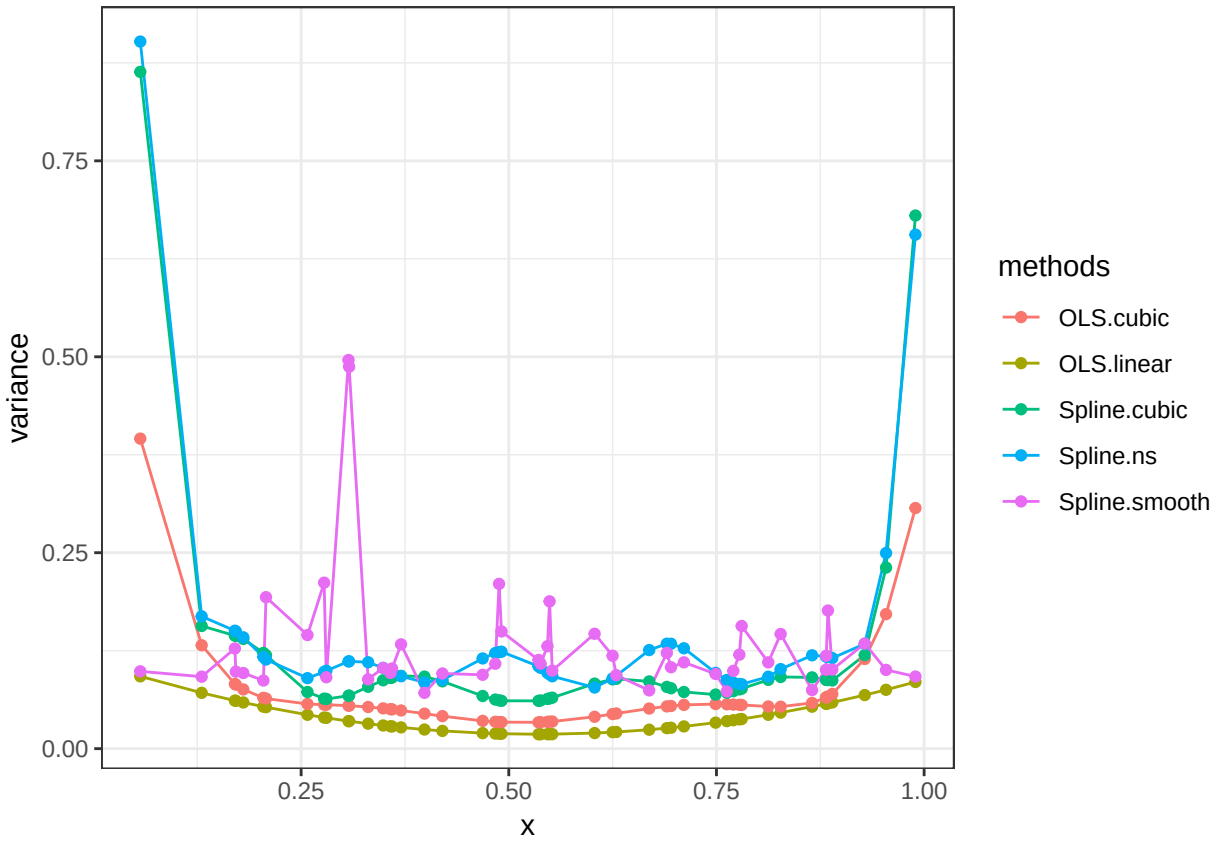
(b) Generate 100 training sets. Each training set consists of 50 pairs of (X, Y), with $(X_1, ..., X_{50}) = \mathbf{x}$ and $Y_i = sin^3(2\pi X_i^3) + \epsilon_i$ for $i = 1, ..., 50$, where ϵ_i is drawn from the standard normal distribution. For each training set, do following:

- Fit the data with methods/models listed below
  - OLS with linear model: $\beta_0 + \beta_1 X$
  - OLS with cubic polynomial model: $\beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3$
  - Cubic spline (or B-spline) with 2 knots at 0.33 and 0.66
  - Natural cubic spline with 5 knots at 0.1, 0.3, 0.5, 0.7, 0.9
  - Smoothing spline with tuning parameter chosen by GCV
- Compute the vector of fitted value $\hat{y}$ obtained from each method/model

(c) Now for each method/model, you obtain a matrix of fitted values, with the i-th row and j-th column value $\hat{y}_{ij}$ representing the fitted value at $X = x_i$ from the j-th training set.

(d) For each method/model, compute the pointwise variance of fitted values across the 100 training sets. This gives you a vector of pointwise variance.

Plot the pointwise variance curves (against x) for each method/-model. (Note: Your plot would be similar to Figure 5.3 of [ESL].)



4. The South African heart disease data is described on page 122 of the textbook. This data set can be found on the text book website. Divide the dataset into a training set consisting of the first 300 observations, and a test set consisting of the remaining observations. Apply logistic regression, LDA and QDA on the training set. For each method, report the test error and its standard error over the test set. Briefly discuss your results.

logistic	LDA	QDA
0.2530864	0.2530864	0.2592593

From the above, we can tell that the overall error of our LDA and logistic regression models are the same, QDA is slightly higher. Generally speaking, LDA assumes that the observations are drawn from a Gaussian distribution with a common covariance matrix across each class, and so

- LDA can provide some improvements over logistic regression when this assumption holds.
- Logistic regression can outperform LDA if these Gaussian assumptions are not met

Both LDA and logistic regression produce linear decision boundaries. QDA, on the other-hand, provides a non-linear quadratic decision boundary, so:

- when the true decision boundaries are linear, then the LDA and logistic regression approaches will tend to perform well
- when the decision boundary is moderately non-linear, QDA may give better results.

Appendix

```
knitr::opts_chunk$set(echo = FALSE, message = FALSE, warning = FALSE, comment = "")
library(splines)
library(MASS)  # for LDA and QDA
library(caret)
library(tidyverse)
options(knitr.table.format = "latex")
theme_set(theme_bw())
# Generate vector x
set.seed(100)
x = runif(50, min = 0, max = 1)
# generate Y matrix 100x50
Y = matrix(nrow = 100, ncol = 50)
generate <- function(y) return(sin(2 * pi * x ^ 3) ^ 3 + rnorm(50))
Y <- t(apply(Y, 1, generate))

# fit OLS with linear model
fit.OLS.linear <- function(y) {
  tmp <- lm(y ~ x)
  return(tmp$fitted.values)
}
predict.OLS.linear <- t(apply(Y, 1, fit.OLS.linear))

# fit OLS with cubic
fit.OLS.cubic <- function(y) {
  tmp1 <- data.frame(x, x^2, x^3)
  tmp2 <- lm(y ~ tmp1$x + tmp1$x.2 + tmp1$x.3)
  return(tmp2$fitted.values)
}
predict.OLS.cubic <- t(apply(Y, 1, fit.OLS.cubic))

# fit cubic spline with 2 knots
fit.spline.cubic <- function(y) {
  tmp <- lm(y ~ bs(x, knots = c(0.33, 0.66)))
  return(tmp$fitted.values)
}
predict.spline.cubic <- t(apply(Y, 1, fit.spline.cubic))
```

```r
# fit natural cubic spline with 5 knots
fit.ns.cubic <- function(y) {
  tmp <- lm(y ~ ns(x, knots = c(0.1, 0.3, 0.5, 0.7, 0.9)))
  return(tmp$fitted.values)
}
predict.ns.cubic <- t(apply(Y, 1, fit.ns.cubic))

# fit natural cubic spline with 5 knots
fit.ss.GCV <- function(y) {
  tmp <- smooth.spline(x = x, y = y, cv = TRUE)
  return(predict(tmp)$y)
}
predict.ss.GCV <- t(apply(Y, 1, fit.ss.GCV))

# calculate pointwise variance
pwVar <- tibble(x = x,
                OLS.linear = apply(predict.OLS.linear, 2, var),
                OLS.cubic = apply(predict.OLS.cubic, 2, var),
                Spline.cubic = apply(predict.spline.cubic, 2, var),
                Spline.ns = apply(predict.ns.cubic, 2, var),
                Spline.smooth = apply(predict.ss.GCV, 2, var))

# plot
pwVar %>% gather(OLS.linear:Spline.smooth, key = "methods", value = "variance") %>%
  ggplot(aes(x = x, y = variance, group = methods, col = methods)) +
  geom_line() +
  geom_point()

# load data
sadh.dat <- read.table("SAHD.txt", sep = ',', header = T) %>%
  as.tibble() %>%
  select(-row.names) %>%
  mutate(chd = as.factor(chd))

train.dat = sadh.dat[c(1:300),]
test.dat = sadh.dat[c(301:nrow(sadh.dat)),] %>% select(-chd)
test.true = sadh.dat[c(301:nrow(sadh.dat)),] %>% select(chd) %>% unlist()

# fit logistic regression
fit.logis <- glm(chd ~ ., data = train.dat, family = "binomial")
predict.logis <- predict(fit.logis, test.dat, type = "response")
predict.logis <- ifelse(predict.logis > 0.5, 1, 0)

# fit LDA
fit.lda <- lda(chd ~ ., data = train.dat)
predict.lda <- predict(fit.lda, test.dat)

# fit QDA
fit.qda <- qda(chd ~ ., data = train.dat)
predict.qda <- predict(fit.qda, test.dat)

# test error
test.error <- tibble(logistic = mean(test.true != predict.logis),
```

```
                      LDA = mean(test.true != predict.lda$class),
                      QDA = mean(test.true != predict.qda$class))
test.error %>% knitr::kable()
```