Extensiones del modelo lineal: regresión local

Contents

1 Algoritmo

- Para $x_i,\ i = 1, \ldots, n$:
  - Se eligen un total de $k = s * n$ puntos alrededor de x_i.
  - Se ajusta un modelo de regresión lineal en x_i utilizando los k puntos.
  - Los modelos más utilizados son la recta y el polinomio de segundo orden.
  - El valor predicho en cada punto es $x_i \Rightarrow \hat{f}(x_i) = X\hat{\beta}$.
- El parámetro s controla la suavidad de la curva ($s \in [0, 1]$).
- Se pueden estimar otras funciones polinómicas distintas a la recta.

2 Estimacion del modelo

Vamos a utilizar el paquete *loess*. En este paquete: - el parámetro s se llama *span*. - el grado del polinomio local se indica con *degree*.

```
d = read.csv("datos/Wage.csv")
d = d[d$wage<250,]
d = d[d$wage<250,]
```

```
m1 = loess(wage ~ age, data = d, span = 0.2, degree = 2)
m2 = loess(wage ~ age, data = d, span = 0.5, degree = 2)
summary(m1)
```

```
## Call:
## loess(formula = wage ~ age, data = d, span = 0.2, degree = 2)
##
## Number of Observations: 2921
## Equivalent Number of Parameters: 16.27
## Residual Standard Error: 30.18
## Trace of smoother matrix: 17.99  (exact)
##
## Control settings:
##   span     :  0.2
##   degree   :  2
##   family   :  gaussian
```

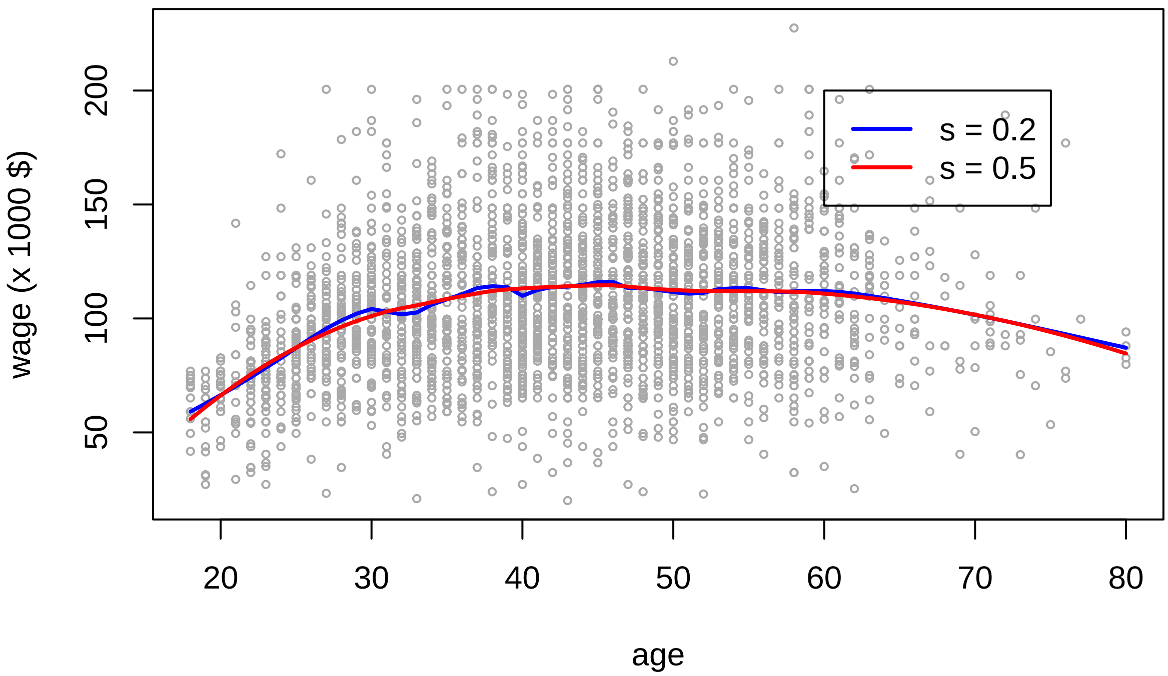
```
##    surface   :  interpolate      cell = 0.2
##    normalize:  TRUE
##   parametric:  FALSE
## drop.square:  FALSE
```

3 Prediccion

3.1 Predicción puntual

```
age_grid = seq(from = min(d$age), to = max(d$age), by = 1)
# con loess hay que utizar se = T, ya que interval = "" no funciona
yp1 = predict(m1, newdata = data.frame(age = age_grid), se = T)
yp2 = predict(m2, newdata = data.frame(age = age_grid), se = T)
```

```
plot(d$age,d$wage, cex = 0.5, col = "darkgrey", ylab = "wage (x 1000 $)", xlab = "age")
#
lines(age_grid, yp1$fit, col = "blue", lwd = 2)
lines(age_grid, yp2$fit, col = "red", lwd = 2)
#
legend(60,200, legend = c("s = 0.2", "s = 0.5"), col = c("blue","red"), lty = 1, lwd = 2)
```



3.2 Intervalo de confianza

Recordemos que:

$$\hat{y}_p - t_{\alpha/2} se(\hat{y}_p) \leq E[y_p] \leq \hat{y}_p + t_{\alpha/2} se(\hat{y}_p)$$

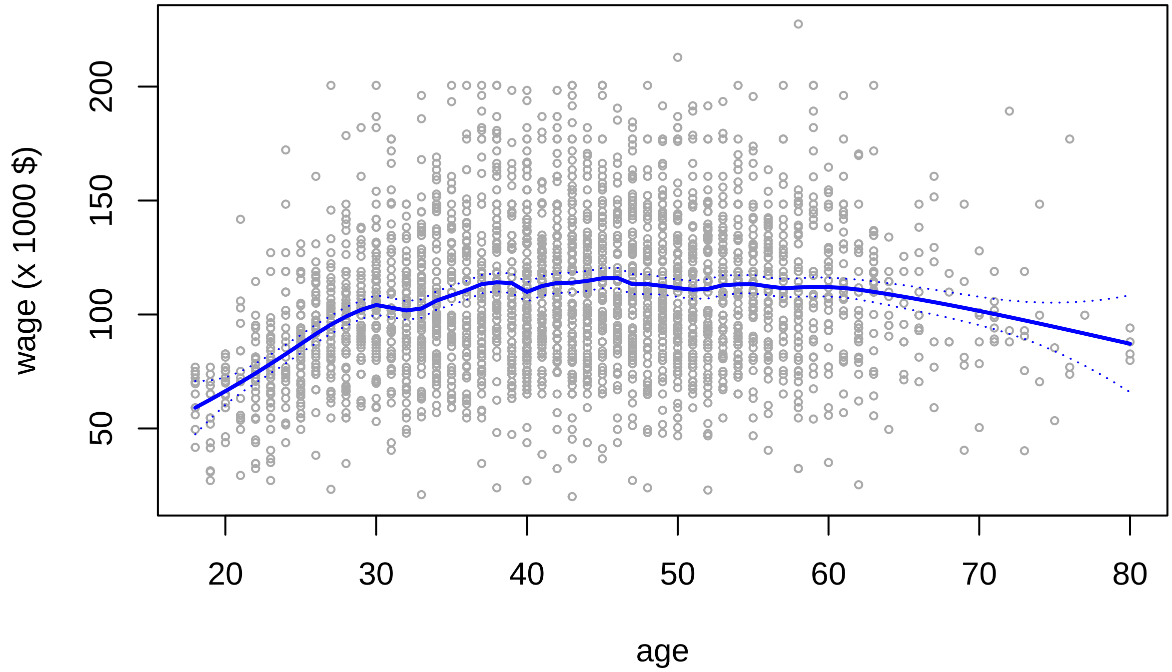
Como la t-student converge a la N(0,1) cuando n es grande (para n>30 la aproximación es aceptable), podemos utilizar:

$$\hat{y}_p - z_{\alpha/2} se(\hat{y}_p) \leq E[y_p] \leq \hat{y}_p + z_{\alpha/2} se(\hat{y}_p)$$

donde $z_{\alpha/2}$ es el valor de la N(0,1) que cumple que $P(z \leq z_{\alpha/2}|z \sim N(0,1)) = 0.1)$. Por tanto:

```
alfa = 0.05
yp11 = yp1$fit + qnorm(alfa/2)*yp1$se.fit # utilizamos la normal en lugar de la t-student
yp12 = yp1$fit + qnorm(1-alfa/2)*yp1$se.fit
```

```
plot(d$age,d$wage, cex = 0.5, col = "darkgrey", ylab = "wage (x 1000 $)", xlab = "age")
#
lines(age_grid, yp1$fit, col = "blue", lwd = 2)
lines(age_grid, yp11, col = "blue", lty = 3)
lines(age_grid, yp12, col = "blue", lty = 3)
```



3.3 Intervalo de predicción

En regresión lineal se tenía que:

$$\hat{y}_p - t_{\alpha/2}\hat{s}_R\sqrt{1+v_p} \leq y_p \leq \hat{y}_p + t_{\alpha/2}\hat{s}_R\sqrt{1+v_p}$$

y además $se(\hat{y}_p) = \hat{s}_R\sqrt{v_p}$. Por tanto, el intervalo de predicción se puede calcular usando:

$$\hat{y}_p - t_{\alpha/2}\sqrt{\hat{s}_R^2 + se(\hat{y}_p)^2} \leq y_p \leq \hat{y}_p + t_{\alpha/2}\sqrt{\hat{s}_R^2 + se(\hat{y}_p)^2}$$

```
sR = m1$s
alfa = 0.05
yp13 = yp1$fit + qnorm(alfa/2)*sqrt(sR^2 + yp1$se.fit^2)
yp14 = yp1$fit + qnorm(1-alfa/2)*sqrt(sR^2 + yp1$se.fit^2)
```

```
plot(d$age,d$wage, cex = 0.5, col = "darkgrey", ylab = "wage (x 1000 $)", xlab = "age")
#
lines(age_grid, yp1$fit, col = "blue", lwd = 2)
lines(age_grid, yp11, col = "blue", lty = 3)
lines(age_grid, yp12, col = "blue", lty = 3)
```

```
#
lines(age_grid, yp13, col = "red", lty = 3)
lines(age_grid, yp14, col = "red", lty = 3)
```