Regression and polynomial approximation to CEF

Foundations of Statistical Inference (PLSC 503)

Suppose X and Y have some complicated relationship:

$$Y = 5X + 2X^2 + 3X^3 + 4X^4 + 5X^5 + Z$$

Let $X \sim \mathrm{U}(-5, 5)$, $Z \sim \mathcal{N}(0, 1)$ and assume $X \perp\!\!\!\perp Z$

Question: What is $\mathbb{E}[Y|X = x]$?

$$\begin{aligned}
\mathbb{E}[Y|X = x] &= \mathbb{E}\left[5X + 2X^2 + 3X^3 + 4X^4 + 5X^5 + Z|X = x\right] \\
&= 5x + 2x^2 + 3x^3 + 4x^4 + 5x^5 + \mathbb{E}[Z|X = x] \\
&= 5x + 2x^2 + 3x^3 + 4x^4 + 5x^5 + \mathbb{E}[Z] \\
&= 5x + 2x^2 + 3x^3 + 4x^4 + 5x^5
\end{aligned}$$

Question: What is $\mathbb{E}[Y|X=2]$?

$$\begin{aligned}\mathbb{E}[Y|X=2] &= 5x + 2x^2 + 3x^3 + 4x^4 + 5x^5 \\ &= 10 + 8 + 24 + 64 + 160 = 266\end{aligned}$$

```r
# E[Y|X = 2]:
cef <- function(x){
  5*x + 2*x^2 + 3*x^3 + 4*x^4 + 5*x^5
}

(theta1 <- cef(x = 2))
```

```
## [1] 266
```

Question: What is $\mathbb{E}\left[\frac{\partial \mathbb{E}[Y|X=x]}{\partial x}\right]$?

$$\mathbb{E}\left[\frac{\partial \mathbb{E}[Y|X=x]}{\partial X}\right] = \mathbb{E}\left[5 + 4X + 9X^2 + 16X^3 + 25X^4\right]$$
$$= 5 + 4\mathbb{E}[X] + 9\mathbb{E}[X^2] + 16\mathbb{E}[X^3] + 25\mathbb{E}[X^4]$$

Fact: the n-th moment for $X \sim \mathrm{U}(a, b)$ is

$$\mathbb{E}[X^n] = \frac{1}{n+1}\sum_{k=0}^{n} a^k b^{n-k}$$

e.g. $\mathbb{E}[X^2] = \frac{1}{3}\left(a^2 + ab + b^2\right)$

We can also use the integrate() function in R:

```r
# 1st-4th moments for X ~ U(-5,5)
E_X <- integrate(function(x) x/10, -5, 5)$value
E_X2 <- integrate(function(x) x^2/10, -5, 5)$value
E_X3 <- integrate(function(x) x^3/10, -5, 5)$value
E_X4 <- integrate(function(x) x^4/10, -5, 5)$value

# Average Partial Derivative (APD):
(theta2 <- 5 + 4*E_X + 9*E_X2 + 16*E_X3 + 25*E_X4)
```

```
## [1] 3205
```

Let's write a function to generate IID draws from the joint distribution:

```r
# Function to generate IID draws from (X,Y)
gen_data <- function(n){
  x <- runif(n, min = -5, max = 5)
  z <- rnorm(n)
  y <- 5*x + 2*x^2 + 3*x^3 + 4*x^4 + 5*x^5 + z
  data.frame(x = x, y = y)
}

# Take a single draw
gen_data(n = 1)
```

```
##           x        y
## 1 0.7662568 9.874236
```

Let's generate a sample of $n = 10,000$, and then use the CEF to predict Y:

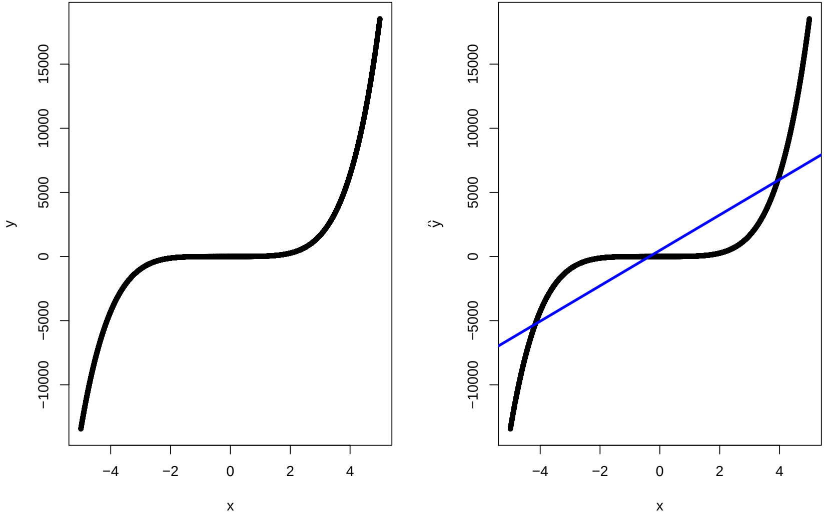
```
set.seed(503)

# Generate large sample
toy_df <- gen_data(n = 10000)

# Compute predicted values via the CEF
toy_df$y_pred <-
  with(toy_df, 5*x + 2*x^2 + 3*x^3 + 4*x^4 + 5*x^5)
```

Now let's make a plot, comparing the X with Y, $\mathbb{E}[Y|X = x]$, and a simple linear regression

```r
par(mfrow = c(1, 2))
# Plot Y ~ X,
plot(y ~ x, toy_df, pch = 20)
# Plot E[Y|X=x] ~ X,
plot(y_pred ~ x, toy_df, pch = 20,
     ylab = expression(hat(y)))
# Overlay simple regression line
abline(lm(y ~ x, toy_df), lwd = 3, col= "blue")
```



y
−10000
−5000
0
5000
10000
15000
−4
−2
0
2
4
x
ŷ
−10000
−5000
0
5000
10000
15000
−4
−2
0
2
4
x

Suppose $n = 50$ and we use a linear approximation. Then

$$\hat{\theta}_1 := \hat{f}(x) \approx \hat{\beta}_0 + \hat{\beta}_1 x$$

and

$$\hat{\theta}_2 := E[\hat{f}'(X)] \approx \hat{\beta}_1$$

```
linear_approx <- function(n = 50, X = 2){
  samp_df <- gen_data(n)
  lm_fit <- lm(y ~ x, samp_df)
  est_theta1 <- sum(c(1,X)*lm_fit$coefficients)
  est_theta2 <- sum(c(0,1)*lm_fit$coefficients)
  c(est_theta1, est_theta2)
}
```

Let's also try a more flexible approach, e.g. a 3rd order polynomial

$$\hat{\theta}_1 := \hat{f}(x) \approx \hat{\beta}_0 + \hat{\beta}_1 x + \hat{\beta}_2 x^2 + \hat{\beta}_3 x^3$$

and

$$\hat{\theta}_2 := E[\hat{f}'(X)] \approx \hat{\beta}_1 + 2\hat{\beta}_2 \mathbb{E}[X] + 3\hat{\beta}_3 \mathbb{E}[X]^2$$

Where $\mathbb{E}[X] = 0$ and $\mathbb{E}[X^2] \approx 8.3$ since $X \sim \mathrm{U}(-5, 5)$

```r
poly_approx3 <- function(n = 50, X = 2){
  samp_df <- gen_data(n)
  lm_fit <- lm(y ~ x + I(x^2)+ I(x^3), samp_df)
  coef_vec <- lm_fit$coefficients
  est_theta1 <- sum(X^(0:3)*coef_vec)
  est_theta2 <- sum(0:3*c(0, 1, 0, 8.3)*coef_vec)
  c(est_theta1, est_theta2)
}
```

Now let's compare bias, variance, and MSE for each

```r
est_linear <- replicate(10000, linear_approx())
est_poly3 <- replicate(10000, poly_approx3())

# Bias:
(bias_linear <-
    apply(est_linear, 1, mean) - c(theta1, theta2))
```

```
## [1]  2988.029 -1830.145
```

```r
(bias_poly3 <-
    apply(est_poly3, 1, mean) - c(theta1, theta2))
```

```
## [1] -441.0592 -443.4778
```

```r
# Variance:
(var_linear <- apply(est_linear, 1, var))
```

```
## [1] 432078.51  37674.69
```

```r
(var_poly3 <- apply(est_poly3, 1, var))
```

```
## [1] 29791.42 12346.19
```

```r
# MSE:
(mse_linear <- bias_linear^2 + var_linear)
```

```
## [1] 9360394 3387107
```

```r
(mse_poly3 <- bias_poly3^2 + var_poly3)
```

```
## [1] 224324.6 209018.8
```

Polynomial approximation was better across all metrics here. What if we tried a 6th order polynomial?

```
poly_approx6 <- function(n = 50, X = 2){
  samp_df <- gen_data(n)
  lm_fit <- lm(y ~ poly(x, 6, raw = TRUE), samp_df)
  coef_vec <- lm_fit$coefficients
  est_theta1 <- sum(X^(0:6)*coef_vec)
  est_theta2 <- sum(0:6*c(0, 1, 0, 8.3, 0, 125, 0)*coef_vec
  c(est_theta1, est_theta2)
}
```

```r
est_poly6 <- replicate(10000, poly_approx6())

# Bias:
bias_poly3
```

```
## [1] -441.0592 -443.4778
```

```r
(bias_poly6 <-
    apply(est_poly6, 1, mean) - c(theta1, theta2))
```

```
## [1] -0.003800301 -0.300183345
```

```r
# Variance:
var_poly3
```

```
## [1] 29791.42 12346.19
```

```r
(var_poly6 <- apply(est_poly6, 1, var))
```

```
## [1] 0.11602159 0.06303925
```

```r
# MSE:
mse_poly3
```

```
## [1] 224324.6 209018.8
```

```r
(mse_poly6 <- bias_poly6^2 + var_poly6)
```

```
## [1] 0.1160360 0.1531493
```

Even better!! What if we tried a 12th order polynomial?

We can keep going, approximating the CEF to arbitrary precision

At some point, however, the approximation might get *too flexible*

Eventually, we will be **overfitting**

Read: Weierstrass Approximation Theorem and Chapter 4 of A&M