Bài 6 - Lấy mẫu lại (Resampling)

Thống kê máy tính và ứng dụng (CLC)

Vũ Quốc Hoàng (vqhoang@fit.hcmus.edu.vn)
FIT - HCMUS

Ngày 12 tháng 4 năm 2022

Nội dung

Nội dung

Giới thiệu

Lấy mẫu lại (resampling) là các kĩ thuật mô phỏng mẫu từ chính **dữ liệu** (data) đã lấy. (Các phương pháp Monte Carlo mô phỏng mẫu từ các mô hình sinh dữ liệu.) Các kĩ thuật này thường cho kết quả kém chính xác hơn các phương pháp Monte Carlo nhưng đơn giản và có thể được dùng trong nhiều trường hợp hơn.

Các phương pháp lấy mẫu lại chính

- Permutation resampling,
- Bootstrapping,
- Cross-validation.

Nội dung

Permutation resampling

Lấy mẫu hoán vị (permutation resampling) là kĩ thuật **lấy mẫu không hoàn lại** (sampling without replacement) trên dữ liệu, tức là bố trí lại thứ tự hay chọn ra các hoán vị hay "xào" dữ liệu. Kĩ thuật này thường được dùng trong kiểm định thống kê, được gọi là **kiểm định hoán vị** (permutation test).

Ví dụ. Bảng dữ liệu sau ghi nhận chiều cao (theo mét) của 10 nữ và 10 nam thanh niên.

Nữ	1.66	1.58	1.58	1.56	1.63	1.51	1.67	1.57	1.61	1.59
Nam	1.77	1.66	1.66	1.64	1.73	1.57	1.78	1.65	1.71	1.68

Từ bảng dữ liệu ta thấy chiều cao trung bình của nữ, nam lần lượt là 1.60, 1.68 (mét). Hỏi: nữ có "thật sự" thấp hơn nam hay không?

Permutation resampling - Ví dụ (tt)

Dùng **kiểm định t** (t-test), với các giả thiết

1. 10 nữ và 10 nam được chọn ngẫu nhiên (độc lập),
2. chiều cao nữ và nam có phân phối chuẩn,
3. phương sai của chiều cao giống nhau trong cả 2 nhóm.

kiểm định "chiều cao trung bình của nữ nhỏ hơn chiều cao trung bình của nam" (kiểm định một phía với đối thuyết "less"), ta có **giá trị** p (p-value) khoảng 0.0012.
Như vậy: nữ "thật sự" thấp hơn nam.

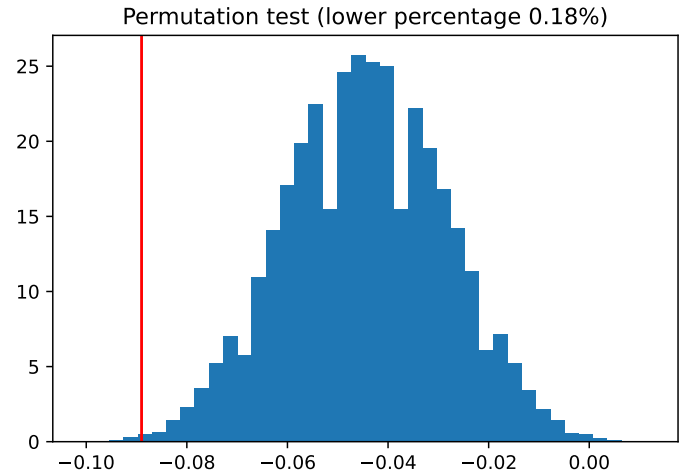
Tuy nhiên, các giả thiết trên có thể không đúng. Dùng kiểm định hoán vị với ý tưởng: nếu nữ không "thật sự" thấp hơn nam, tức là khác biệt giữa chiều cao trung bình của nữ với nam ($1.60 - 1.68 = -0.08$ mét) chỉ là "do ngẫu nhiên" thì khác biệt này "không hiếm thấy" khi ta bố trí lại ngẫu nhiên các chiều cao vào 2 nhóm.

Permutation resampling - Ví dụ (tt)

```
def rand_perm(x1, x2):
    n1, n2 = len(x1), len(x2)
    inds = np.arange(n1 + n2)
    np.random.shuffle(inds)
    x = np.hstack((x1, x2))
    return np.mean(x[inds[:n1]]) - np.mean(x[n1:])

N = 100000
samples = np.fromiter((rand_perm(female_height,
                male_height) for _ in range(N)), "float")
diff = np.mean(female_height) - np.mean(male_height)
np.mean(samples <= diff)
# 0.00176
```

Permutation resampling - Ví dụ (tt)



Nội dung

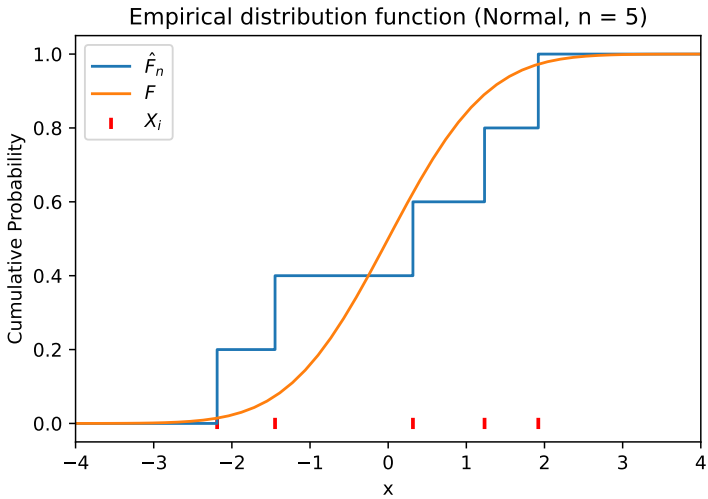
Hàm phân phối thực nghiệm

Cho mẫu ngẫu nhiên $D = \{X_i : i = 1, ..., n\}$ từ phân phối F, nghĩa là $X_1, ..., X_n$ độc lập và cùng phân phối với **hàm phân phối** (distribution function, cumulative distribution function, cdf) F. **Hàm phân phối thực nghiệm** (empirical distribution function) của D là hàm được cho bởi

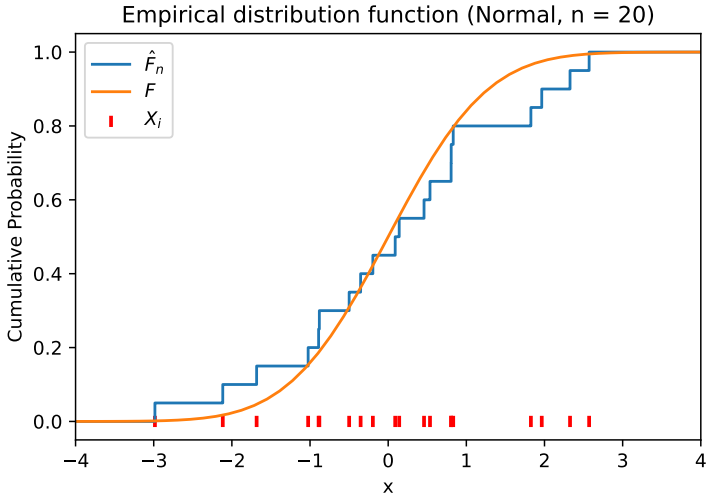
$$\hat{F}_D(x) = \hat{F}_n(x) = \hat{F}(x) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}_{(-\infty,x]}(X_i) = \frac{\text{số lượng } X_i \leq x}{n}, x \in \mathbb{R}.$$

- $\hat{F}_n$ là hàm phân phối của biến ngẫu nhiên rời rạc nhận các giá trị $X_1, ..., X_n$ với cùng xác suất $\frac{1}{n}$.
- "$\hat{F}_n \approx F$" khi n "đủ lớn".

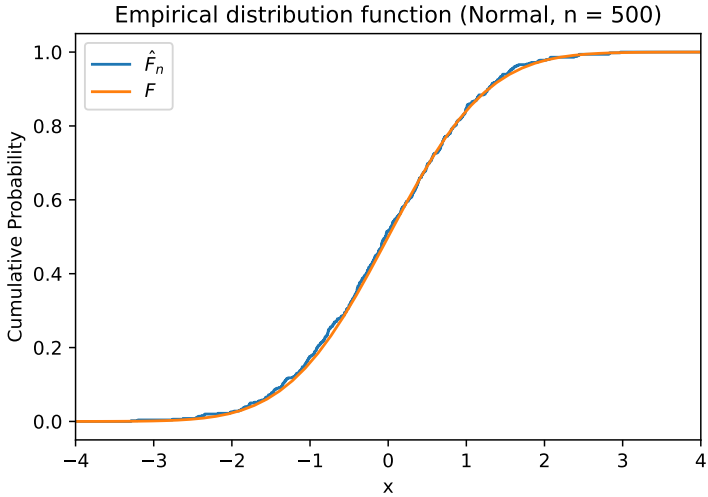
Hàm phân phối thực nghiệm (tt)



Hàm phân phối thực nghiệm (tt)



Hàm phân phối thực nghiệm (tt)



Bootstrapping

Để lấy mẫu cỡ m từ hàm phân phối thực nghiệm $\hat{F}_D$ của $D = \{X_1, ..., X_n\}$, ta có thể làm như sau

1. Sinh $I_1, ..., I_m \overset{\text{iid}}{\sim} \mathcal{U}\{1, ..., n\}$,
2. Trả về mẫu $\{X_{I_1}, ..., X_{I_m}\}$ (mẫu này còn được gọi là **mẫu boostrap** cỡ m từ D).

Đây chính là **lấy mẫu có hoàn lại** (sampling with replacement) từ D.

Giả sử ta có mẫu ngẫu nhiên $D = \{X_1, ..., X_n\}$ và dùng một **thống kê** (statistic) $\hat{\theta} = \hat{\theta}(D)$ trên mẫu để ước lượng một **tham số** (parameter) $\theta = \theta(F)$ trên hàm phân phối F.

- Nếu có thể sinh mẫu từ F thì dùng các phương pháp Monte Carlo (đã học), ta sinh $D_1, ..., D_N$ và tính $\hat{\theta}(D_1), ..., \hat{\theta}(D_N)$. Từ đó ta có thể đánh giá được ước lượng $\hat{\theta}$.
- Nếu ta chỉ có một mẫu dữ liệu D thì dùng phương pháp lấy mẫu từ $\hat{F}_D$, ta sinh các mẫu bootstrap cỡ n là $D_1^*, ..., D_N^*$ và tính $\hat{\theta}(D_1^*), ..., \hat{\theta}(D_N^*)$. Với "$\hat{F}_n \approx F$", ta vẫn có thể đánh giá được $\hat{\theta}$.

Bootstrapping - Ví dụ 1

Giả sử ta muốn khảo sát chiều cao trung bình của một tổng thể là tất cả cư dân của một thành phố. Từ tổng thể, ta thu thập được một mẫu dữ liệu ngẫu nhiên gồm chiều cao của $n = 20$ người như bảng dưới.

1.56 1.47 1.59 1.65 1.62 1.78 1.69 1.49 1.92 1.55
1.65 1.52 1.65 1.60 1.71 1.48 1.69 1.65 1.59 1.74

Gọi μ là chiều cao trung bình của cư dân thành phố, $\bar{x}$ là chiều cao trung bình của $n = 20$ người trong mẫu dữ liệu

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i.$$

Ta không biết tham số μ là bao nhiêu, nhưng một cách "hợp lý", ta có thể dùng thống kê $\bar{x}$ để ước lượng μ. Tức là, ta đoán rằng

$$\mu \approx \bar{x} = 1.63.$$

Bootstrapping - Ví dụ 1 (tt)

Tuy nhiên, ta không "tin cậy" lắm vào ước lượng này vì tùy theo mẫu dữ liệu thu thập mà ta có kết quả ước lượng khác nhau. Thay vì đưa ra một giá trị ước lượng, ta đưa ra một khoảng ước lượng cho μ, trong đó có tính đến sự biến động của $\bar{x}$ theo mẫu dữ liệu cỡ $n = 20$ có thể thu thập từ tổng thể.

Theo định lý giới hạn trung tâm, khi cỡ mẫu n đủ lớn, ta có phân phối của $\bar{x}$ "xấp xỉ chuẩn", cụ thể là

$$\bar{x} \sim \mathcal{N}\left(\mu, \sigma^2/n\right)$$

với σ^2 là phương sai của chiều cao của một người trong tổng thể. Từ đó, ta có khoảng ước lượng $1 - \alpha$ cho μ là

$$\bar{x} \pm z_{1-\alpha/2} se$$

với $z_{1-\alpha/2}$ là phân vị mức $1 - \alpha/2$ của phân phối chuẩn tắc $Z \sim \mathcal{N}(0, 1)$ còn se là sai số chuẩn của $\bar{x}$, tức là

$$se = \sqrt{Var(\bar{x})} = \frac{\sigma}{\sqrt{n}}$$

Bootstrapping - Ví dụ 1 (tt)

Nếu không biết giá trị của σ, ta có thể xấp xỉ sai số chuẩn bằng $se \approx \hat{se} = \frac{s}{\sqrt{n}}$ với s là độ lệch chuẩn của mẫu dữ liệu

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}.$$

Thật ra công thức ước lượng trên chưa tốt lắm, khi dùng s thay cho σ, ta có thể dùng phân phối Student để xấp xỉ tốt hơn phân phối chuẩn, cụ thể ta có

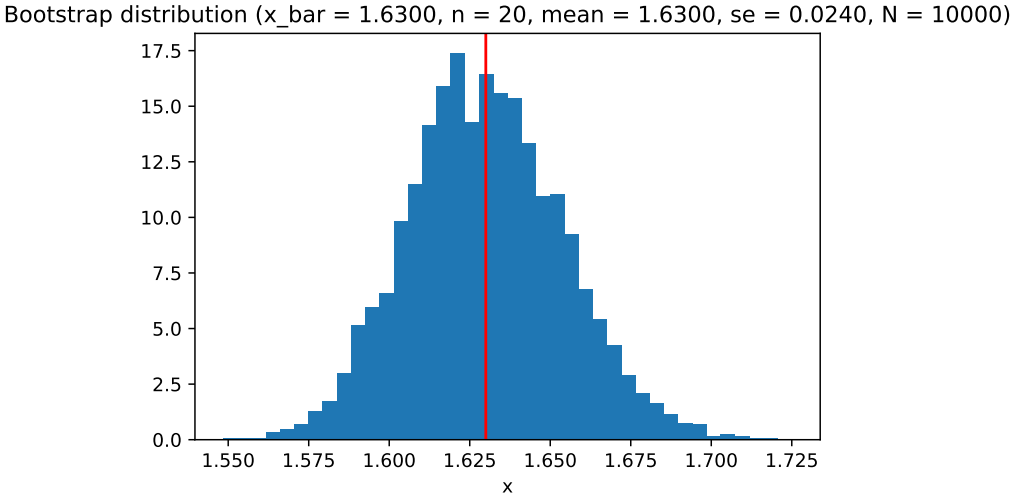
$$\frac{\bar{x} - \mu}{s/\sqrt{n}} \sim t^{n-1}$$

Từ đó, ta có khoảng ước lượng $1 - \alpha$ cho μ là $\bar{x} \pm t^{n-1}_{1-\alpha/2}\frac{s}{\sqrt{n}}$ với $t^{n-1}_{1-\alpha/2}$ là phân vị mức $1 - \alpha/2$ của phân phối Student với bậc tự do là $n - 1$.

Bootstrapping - Ví dụ 1 (tt)

```
x = np.array([...])
n = len(x)
x_bar = np.mean(x)
N = 10000
boot_dist = [np.mean(np.random.choice(x, size=n, replace=
    True)) for _ in range(N)]
plt.hist(boot_dist, bins=40, density=True)
anpha = 1 - 95/100
print(np.quantile(boot_dist, [anpha/2, 1 - anpha/2]))
# [1.585 1.679]
se = np.std(x)/(n**0.5)
t = stats.t.ppf(1 - anpha/2, df=n-1)
print([x_bar - t*se, x_bar + t*se])
# [1.5798984074733546, 1.6801015925266456]
```

Bootstrapping - Ví dụ 1 (tt)



Bootstrapping - Ví dụ 2

Hồi qui tuyến tính (linear regression) y theo x với dữ liệu

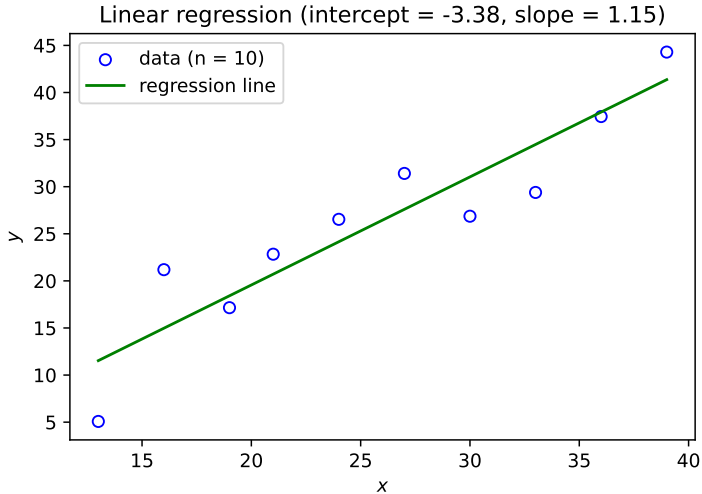
x	y	x	y
13	5.0768	27	31.4085
16	21.1897	30	26.8648
19	17.1548	33	29.3894
21	22.8325	36	37.4476
24	26.5348	39	44.2920

Mô hình

$$y = \beta_0 + \beta_1 x + \epsilon,$$

β_0: intercept, β_1: slope, ϵ: biến ngẫu nhiên mô tả "nhiễu" (noise).

Bootstrapping - Ví dụ 2 (tt)



Bootstrapping - Ví dụ 2 (tt)

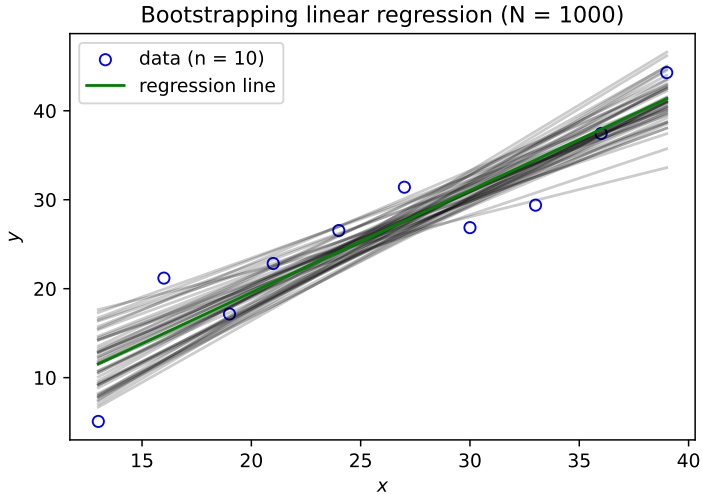
Để phân tích "sâu hơn" mô hình (chẳng hạn đánh giá mức độ chắc chắn), ta có thể dùng kĩ thuật bootstrapping. Cụ thể:

1. Từ dữ liệu $D = \{(x_i, y_i) : i = 1, ..., n\}$, lấy mẫu bootstrap cùng cỡ với dữ liệu $D^* = \{(x_i^*, y_i^*) : i = 1, ..., n\}$. (Lưu ý, x_i, y_i "đi chung" với nhau.)
2. Từ mẫu bootstrap D^* tìm β_0^*, β_1^* (bằng kĩ thuật đã học).
3. Lặp lại bước 1-2 nhiều lần (N lần) để được phân phối bootstrap cho (β_0, β_1).
4. Từ phân phối bootstrap đưa ra các phân tích đánh giá (chẳng hạn, khoảng tin cậy cho β_1 hay khoảng tin cậy của y theo x được cho).

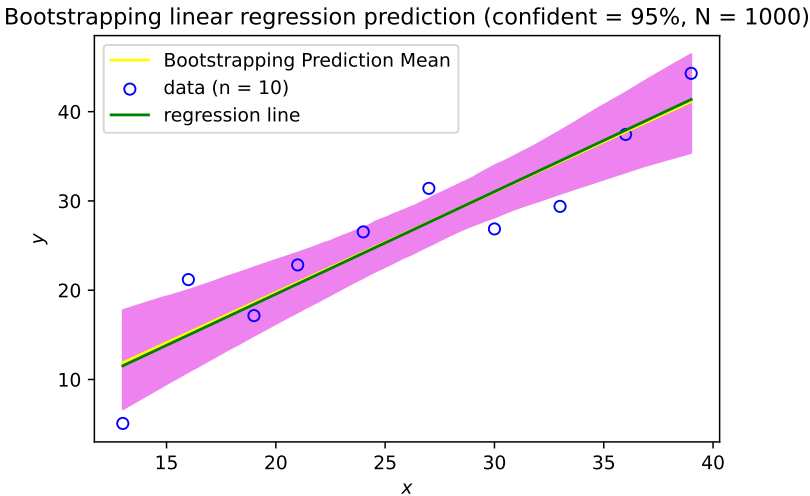
Bootstrapping - Ví dụ 2 (tt)

```
def linregress_bootstrap(x, y, N):
    intercept, slope  = [], []
    for _ in range(N):
        index = np.random.choice(np.arange(len(x)), size=
           len(x), replace=True)
        x_boots, y_boots = x[index], y[index]
        lin_model = stats.linregress(x_boots, y_boots)
        intercept.append(lin_model.intercept)
        slope.append(lin_model.slope)
    return {"intercept": np.array(intercept),
            "slope": np.array(slope)}
boots_dist = linregress_bootstrap(x, y, 1000)
np.quantile(boots_dist["slope"], [0.025, 0.975])
# [0.64 1.43]
```

Bootstrapping - Ví dụ 2 (tt)



Bootstrapping - Ví dụ 2 (tt)



Nội dung

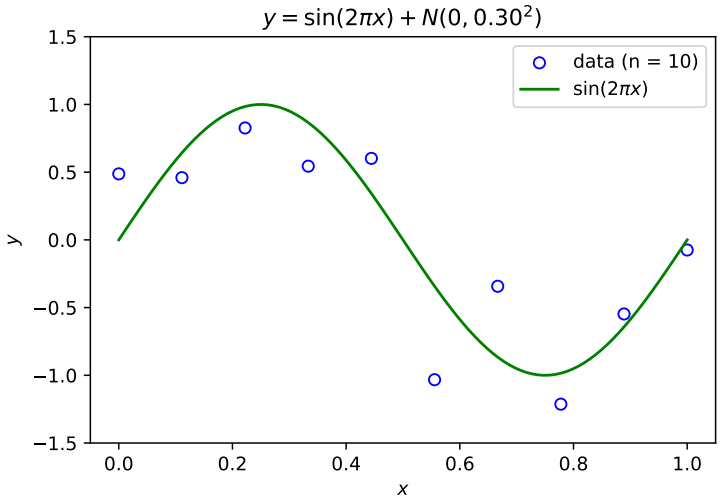
Ví dụ - Bình phương tối tiểu

Bài toán. Giả sử 2 đại lượng x, y có quan hệ nào đó, chẳng hạn

$$y = f(x) + \epsilon \text{ với } f(x) = \sin(2\pi x) \text{ và } \epsilon \sim \mathcal{N}(0, \sigma^2).$$

Cho tập dữ liệu $D = \{\mathbf{x}, \mathbf{y}\}$ với $\mathbf{x} = (x_1, x_2, ..., x_n)$ và $\mathbf{y} = (y_1, y_2, ..., y_n)$ tương ứng, xác định quan hệ giữa x, y; chẳng hạn tìm f.

Ví dụ - Bình phương tối tiểu (tt)



Ví dụ - Bình phương tối tiểu (tt)

Giải. "Để đơn giản" ta giả sử f có thể "được mô hình" bằng một đa thức bậc d có dạng

$$y = \sum_{j=0}^{d} w_j x^j = w_0 + w_1 x + w_2 x^2 + ... + w_d x^d, d \in \mathbb{N}, w_j \in \mathbb{R}.$$

Ta cần tìm d và $\mathbf{w} = (w_0, w_1, ..., w_d)$ "tốt nhất". Cụ thể, ta tìm d và $\mathbf{w}$ sao cho với x "nói chung", ta có $\hat{y} = \sum_{j=0}^{d} w_j x$ "gần" $y = f(x)$ nhất có thể.
Tuy nhiên, ta chỉ có các quan sát của y tương ứng với x trong tập dữ liệu D nên ta "tạm thời" tìm d và $\mathbf{w}$ làm cực tiểu **trung bình bình phương sai số** (mean squared error - MSE) trên dữ liệu

$$\mathcal{L}_D(d, \mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \sum_{j=0}^{d} w_j x_i^j \right)^2.$$

Ví dụ - Bình phương tối tiểu (tt)

Cố định d và đặt ma trận

$$\mathbf{X} = \begin{pmatrix} 1 & x_1 & x_1^2 & \dots & x_1^d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^d \end{pmatrix},$$

ta có thể viết lại $\mathcal{L}_D(d, \mathbf{w})$ theo dạng ma trận - vector là

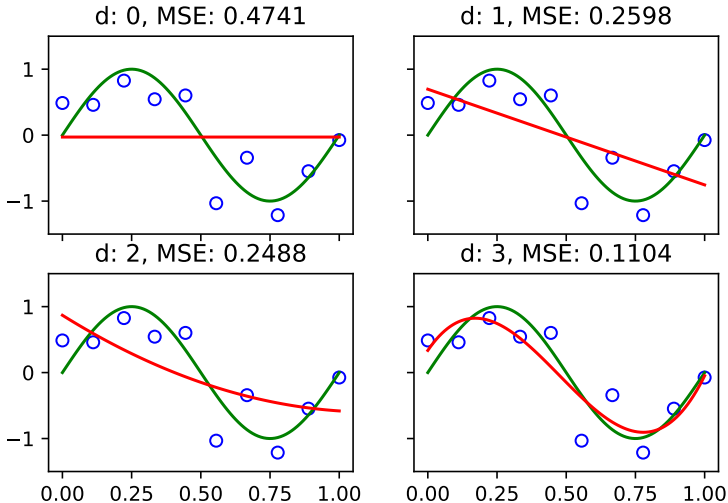
$$\mathcal{L}_D(d, \mathbf{w}) = \frac{1}{n} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2.$$

Phương pháp **bình phương tối tiểu** (least squares) cho $\mathbf{w}$ tốt nhất là

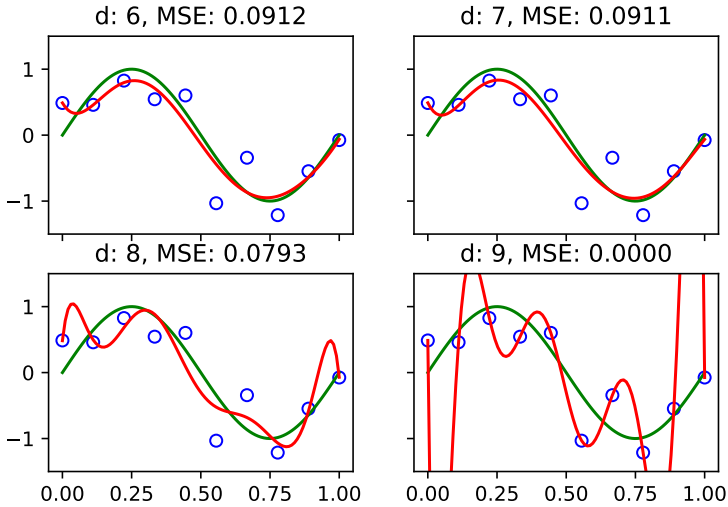
$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}}\, \mathcal{L}_D(d, \mathbf{w}) = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} = \mathbf{X}^\dagger\mathbf{y},$$

với $\mathbf{X}^\dagger$ là ma trận **giả nghịch đảo** (pseudo-inverse) của $\mathbf{X}$.

Ví dụ - Bình phương tối tiểu (tt)



Ví dụ - Bình phương tối tiểu (tt)



Ví dụ - Bình phương tối tiểu (tt)

Nhận xét

- Khi d nhỏ ($d = 0, 1, 2$), đa thức tìm được **"chưa khớp"** (underfitting) dữ liệu. Chưa-khớp không tốt vì mô hình còn quá đơn giản nên chưa giải thích được dữ liệu.
- Khi d lớn ($d = 8, 9$), đa thức tìm được **"quá khớp"** (overfitting) dữ liệu. Quá-khớp không tốt vì mô hình quá rắc rối nên giải thích luôn cả **"nhiễu"** (noise).
- Khi d càng lớn thì MSE trên dữ liệu càng nhỏ, mô hình càng ít lỗi trên dữ liệu. Tuy nhiên, ta cần mô hình ít lỗi trên x, y nói chung, đặc biệt là x, y không có trong dữ liệu. Tức là, mô hình cần có khả năng **tổng quát hóa** (generalization).

Hỏi: làm sao chọn được d tốt nhất?
Giải pháp: kiểm tra chéo.
Nguyên lý: cần đánh giá mô hình trên dữ liệu khác với dữ liệu dùng để **"huấn luyện"** (training) mô hình.

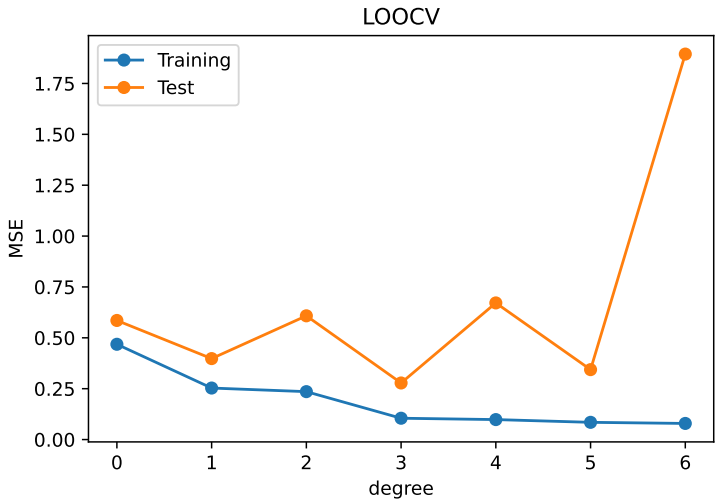
Cross-validation

- Kĩ thuật **kiểm tra chéo k-lần** (k-fold cross-valiation) chia dữ liệu thành k phần rời nhau, luân phiên dùng $k-1$ phần để huấn luyện mô hình và đánh giá trên 1 phần còn lại, lấy trung bình điểm đánh giá trong k lần.
- Kĩ thuật **kiểm tra chéo chừa-1** (leave-one-out cross-validation, LOOCV) là một trường hợp đặc biệt của k-fold CV với $k = n$.
- Có thể kiểm tra chéo "kĩ hơn" bằng kĩ thuật **kiểm tra chéo chừa-p** (leave-p-out cross-validation, LpO CV)
  - Xét tất cả các cách chia dữ liệu thành 2 phần,
  - một phần có cỡ $n-p$ dùng để huấn luyện,
  - và một phần có cỡ p dùng để đánh giá
  - sau cùng, lấy trung bình các đánh giá trên tất cả các cách chia.
- LOOCV cũng có thể được xem là LpO CV với $p = 1$.

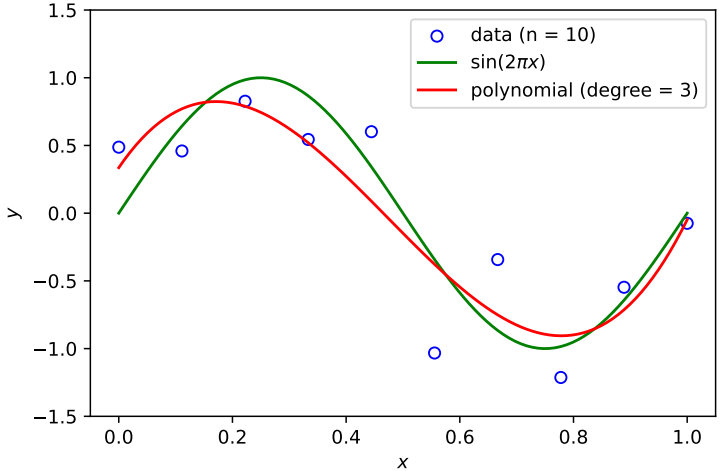
Cross-validation (tt)

- Vì trong LpO CV, tất cả các tập con của mẫu đều được dùng để đánh giá (và để huấn luyện) nên kĩ thuật này được gọi là **kiểm tra chéo hoàn chỉnh** (exhaustive cross-validation).
- Ngược lại, k-fold CV được gọi là **kiểm tra chéo không hoàn chỉnh** (non-exhaustive cross-validation).
- Kiểm tra chéo hoàn chỉnh thường cho đánh giá tốt hơn nhưng tốn nhiều thời gian tính toán hơn. Chẳng hạn, với LpO CV, ta cần huấn luyện và đánh giá trong $\binom{n}{p}$ lần so với k lần của k-fold CV.
- Xem thêm các kĩ thuật kiểm tra chéo khác tại `https://en.wikipedia.org/wiki/Cross-validation_(statistics)`.

Cross-validation - Ví dụ (tt)



Cross-validation - Ví dụ (tt)



Tài liệu tham khảo

Chapter 2, 6-9, 15, 21. Thomas M. Carsey, Jeffrey J. Harden. *Monte Carlo Simulation and Resampling Methods for Social Science*. SAGE Publications, 2014.

Chapter 7. Dirk P. Kroese, Joshua C.C. Chan. *Statistical Modeling and Computation*. Springer, 2014.

Chapter 8, 9. Jochen Voss. *An Introduction to Statistical Computing - A Simulation-based Approach*. John Wiley & Sons, 2014.