Module 5 - Week 1

A Bridge to Linear Regression

TimeSeries Team

Ngày 18 tháng 10 năm 2025

I. Simple Linear Regression

1.1 QUIZ

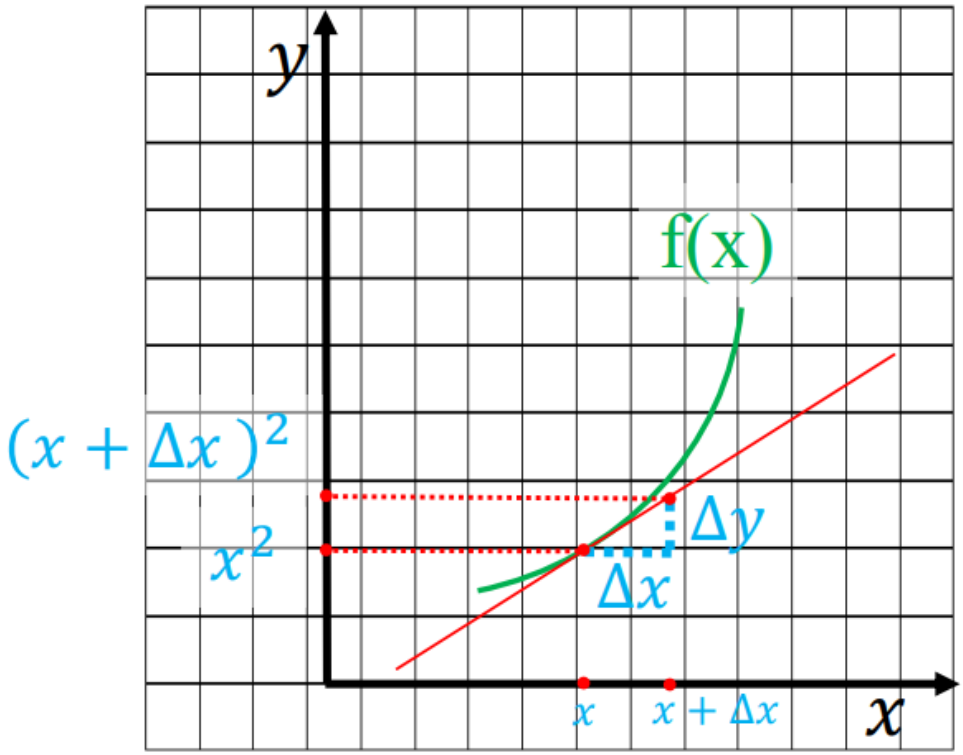
1. Khi thay đổi hệ số w, đường thẳng sẽ thay đổi như thế nào?
   (a) Đường thẳng dịch chuyển song song lên trên hoặc xuống dưới.
   (b) Đường thẳng dịch chuyển song song theo trục hoành.
   (c) **Độ nghiêng của đường thẳng thay đổi.**
   (d) Đường thẳng không thay đổi.
2. Khi thay đổi hệ số b, điều gì xảy ra với đường thẳng?
   (a) Độ nghiêng của đường thẳng thay đổi.
   (b) **Đường thẳng dịch chuyển song song.**
   (c) Đường thẳng trở nên thẳng đứng.
   (d) Đường thẳng đi qua gốc tọa độ.

Hệ số w là **độ dốc** của đường thẳng $y = wx + b$. Khi w thay đổi, **độ nghiêng (góc xoay)** của đường thẳng thay đổi, làm cho đường thẳng xoay quanh trục tung. Hệ số b là **hệ số chặn (intercept)**, xác định vị trí đường thẳng cắt trục tung. Khi b thay đổi, đường thẳng **dịch chuyển song song lên hoặc xuống** mà không thay đổi độ nghiêng.

Trong bài toán *Linear Regression*, mục tiêu là **điều chỉnh** các hệ số w và b sao cho đường thẳng $y = wx + b$ **nằm gần các điểm dữ liệu nhất** (nghĩa là tổng sai số bình phương giữa dự đoán và giá trị thực là nhỏ nhất).

1.2 Đạo hàm là gì?

Ý nghĩa hình học của đạo hàm Giả sử ta vẽ đường **tiếp tuyến** với đồ thị hàm số $y = f(x)$ tại điểm $(x_0, f(x_0))$. Tại điểm đó, ta có thể dựng một **tam giác vuông nhỏ** bởi:



- Cạnh kề nằm trên trục hoành, độ dài là Δx.
- Cạnh đối nằm song song với trục tung, độ dài là $\Delta y = f(x_0 + \Delta x) - f(x_0)$.
- Góc α là góc giữa tiếp tuyến và trục hoành.

Theo định nghĩa lượng giác:

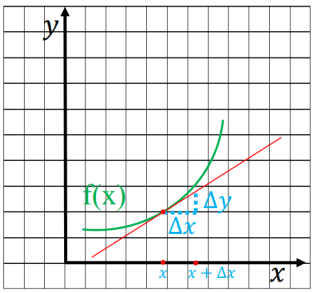
$$\tan(\alpha) = \frac{\text{cạnh đối}}{\text{cạnh kề}} = \frac{\Delta y}{\Delta x}$$

Khi $\Delta x \to 0$, đường secant trở thành **đường tiếp tuyến**, và khi đó:

$$\tan(\alpha) = \lim_{\Delta x \to 0} \frac{\Delta y}{\Delta x} = \frac{dy}{dx} = f'(x_0)$$

Đạo hàm $f'(x_0)$ chính là hệ số góc (độ dốc) của tiếp tuyến tại điểm đó.

❖ **Đạo hàm cho hàm liên tục**



$\frac{d}{dx}f(x), \frac{dy}{dx}, y', f'(x)$

Đạo hàm = $\frac{\text{Thay đổi theo } y}{\text{Thay đổi theo } x} = \frac{\Delta y}{\Delta x}$

$$\frac{d}{dx}f(x) = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

Δx cần tiến về 0 để đường tiếp tuyến tiến về hàm f(x) trong vùng lân cận tại x

Vì sao khi $\Delta x \to 0$ thì đường secant trở thành tiếp tuyến?

Giả sử ta có hai điểm trên đồ thị hàm $y = f(x)$:

$$A(x, f(x)) \quad \text{và} \quad B(x + \Delta x, f(x + \Delta x))$$

Đường thẳng đi qua A và B được gọi là **đường secant**, có hệ số góc:

$$m_{\text{secant}} = \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

Khi ta **giảm dần** Δx, điểm B tiến gần đến A. Khi đó:

- Hai điểm A và B gần như trùng nhau.
- Đường secant bắt đầu **xoay quanh điểm** A.
- Khi $\Delta x \to 0$, đường secant **tiệm cận** tới một vị trí cố định — chính là **đường tiếp tuyến** của đồ thị tại A.

Toán học mô tả quá trình này bằng giới hạn:

$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

Kết luận: Đạo hàm $f'(x)$ biểu diễn **độ dốc tức thời** (tốc độ biến thiên tại một điểm), chính là **hệ số góc của đường tiếp tuyến** tại điểm $(x, f(x))$.

Lưu ý: Phân biệt phương trình đạo hàm và phương trình tiếp tuyến

- **Phương trình đạo hàm** cho ta *công thức tổng quát biểu diễn độ dốc (slope) của hàm tại mọi điểm.*

  $$f'(x) = \text{slope của tiếp tuyến tại } (x, f(x))$$

  Ví dụ: nếu $f(x) = x^2$, thì $f'(x) = 2x$. Điều này có nghĩa là tại mỗi giá trị x, độ dốc của tiếp tuyến thay đổi theo $2x$.

- **Phương trình tiếp tuyến** lại là *phương trình của một đường thẳng cụ thể*, được xác định tại một điểm duy nhất $(x_0, f(x_0))$, có độ dốc $f'(x_0)$:

  $$y - f(x_0) = f'(x_0)(x - x_0)$$

  Đây là phương trình tuyến tính mô tả chính đường tiếp tuyến tại điểm x_0 trên đồ thị.

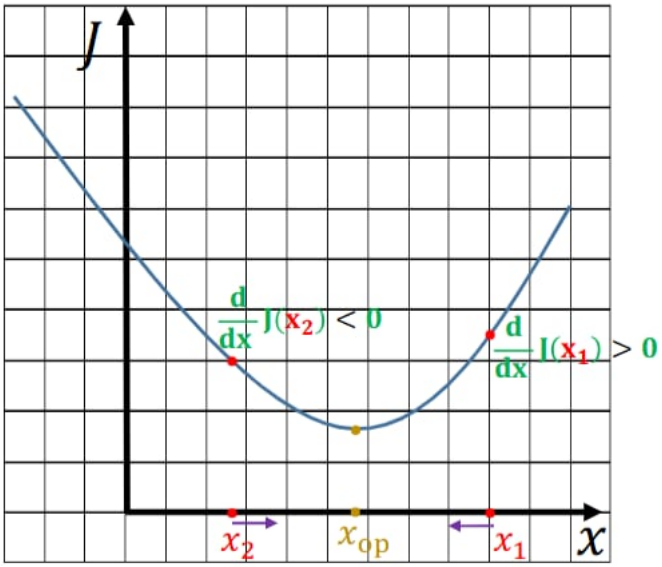
- Nói cách khác:

  $$f'(x) \quad \to \text{công thức cho "độ dốc tại mọi điểm"}$$
  $$y - f(x_0) = f'(x_0)(x - x_0) \quad \to \text{đường thẳng cụ thể tại } x_0$$

- **Tóm lại:** Đạo hàm $f'(x)$ là *hàm độ dốc*, còn phương trình tiếp tuyến là *ứng dụng của đạo hàm tại một điểm cụ thể.*

1.3 Hàm bậc hai và gradient descent

Xét một điểm x_1, ta đặt câu hỏi: *Làm thế nào để tìm được một điểm x_2 sao cho $f(x_2) < f(x_1)$, tức là tiến gần hơn tới giá trị tối ưu (cực tiểu)?*



Quan sát đồ thị của hàm parabol, ta nhận thấy:

- Khi x_1 nằm bên trái điểm cực tiểu ($x_1 < 0$), đạo hàm $f'(x_1)$ mang giá trị **âm**. Điều này có nghĩa là **hàm đang giảm theo hướng tăng của** x. Để tiến về cực tiểu, ta cần **tăng** x, tức là di chuyển *ngược chiều với dấu của đạo hàm.*
- Ngược lại, khi x_1 nằm bên phải điểm cực tiểu ($x_1 > 0$), đạo hàm $f'(x_1)$ **dương**. Khi đó hàm đang tăng theo hướng tăng của x, nên để tiến về cực tiểu, ta phải **giảm** x, một lần nữa là *đi ngược chiều đạo hàm.*

Từ hai quan sát trên, ta có thể hiểu một cách trực giác rằng cần *đi ngược chiều đạo hàm* để tiến về cực tiểu. Tuy nhiên, để chứng minh chặt chẽ hơn, ta xét dưới góc nhìn **đạo hàm theo hướng** (directional derivative).

(1) Đạo hàm theo hướng

Gọi $u \in \mathbb{R}^d$ là **vector đơn vị** với $\|u\| = 1$. Giả sử ta đang ở điểm x_t và muốn di chuyển một bước rất nhỏ Δx theo hướng của vector đơn vị u, ta *phân tách* độ lớn và hướng bằng

$$\Delta x \;=\; \varepsilon\, u.$$

với ϵ là độ dịch chuyển rất nhỏ. Khi đó, nếu f khả vi tại x_t, khai triển Taylor bậc nhất cho ta

$$f(x_t + \varepsilon u) = f(x_t) + \varepsilon\, \nabla f(x_t)^\top u \;+\; o(\varepsilon) \quad (\varepsilon \to 0).$$

Đạo hàm theo hướng u tại x_t được **định nghĩa** bởi giới hạn

$$D_u f(x_t) := \lim_{\varepsilon \to 0^+} \frac{f(x_t + \varepsilon u) - f(x_t)}{\varepsilon} = \nabla f(x_t)^\top u.$$

Nói cách khác, $\nabla f(x_t)^\top u$ *đo tốc độ biến thiên cục bộ* của f khi ta đi từ x_t theo hướng u.

Lưu ý

Nếu $\nabla f(x_t) = 0$ thì $D_u f(x_t) = 0$ với mọi hướng u; đây là điều kiện dừng bậc nhất (có thể là cực tiểu, cực đại hoặc yên ngựa).

(2) Hướng giảm nhanh nhất của hàm

Ta muốn tìm hướng đơn vị u làm f giảm nhanh nhất tại x_t, tức là nghiệm của bài toán

$$\min_{\|u\|=1} \nabla f(x_t)^\top u.$$

Theo bất đẳng thức Cauchy–Schwarz,

$$\nabla f(x_t)^\top u \;\geq\; -\|\nabla f(x_t)\|\,\|u\| = -\|\nabla f(x_t)\|.$$

Dấu bằng xảy ra khi và chỉ khi

$$u \;=\; -\frac{\nabla f(x_t)}{\|\nabla f(x_t)\|}.$$

Vì vậy, **hướng giảm nhanh nhất** tại x_t chính là *hướng ngược chiều gradient*:

$$u^* \;=\; -\frac{\nabla f(x_t)}{\|\nabla f(x_t)\|}.$$

(3) Quy luật cập nhật Gradient Descent

Thay vì bước vô cùng nhỏ, ta dùng bước hữu hạn với hệ số học $\eta > 0$:

$$x_{t+1} \;=\; x_t \;-\; \eta\,\nabla f(x_t).$$

- $\nabla f(x_t)$ chỉ *hướng tăng nhanh nhất* của f; do đó $-\nabla f(x_t)$ là hướng giảm nhanh nhất.
- η điều chỉnh độ dài bước: lớn quá có thể "vọt" khỏi vùng tốt, nhỏ quá thì hội tụ chậm.

Vì sao bước trên làm f giảm? (đảm bảo giảm với η nhỏ)

Nếu f **khả vi liên tục** tại x_t, thì từ khai triển Taylor:

$$f(x_t - \eta\nabla f(x_t)) = f(x_t) \;-\; \eta\,\|\nabla f(x_t)\|^2 \;+\; o(\eta).$$

Suy ra tồn tại $\eta_0 > 0$ sao cho với mọi $0 < \eta < \eta_0$,

$$f(x_{t+1}) < f(x_t) \quad \text{khi } \nabla f(x_t) \neq 0.$$

Mạnh hơn, nếu f có **gradient L-Lipschitz** (tức là f L-*smooth*), thì bất đẳng thức trơn cho ta

$$f(x_t - \eta\nabla f(x_t)) \;\leq\; f(x_t) - \left(\eta - \tfrac{L}{2}\eta^2\right)\|\nabla f(x_t)\|^2.$$

Do đó chọn $0 < \eta < \frac{2}{L}$ sẽ đảm bảo vế phải nhỏ hơn $f(x_t)$ (trừ khi $\nabla f(x_t) = 0$).

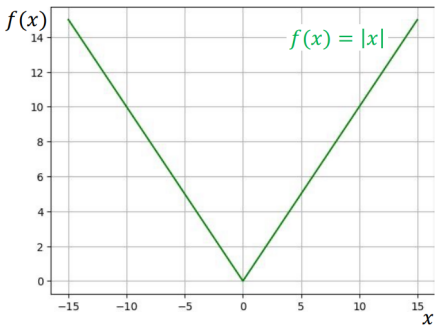
Điều kiện để Gradient Descent có nghĩa

- $f(x)$ phải **liên tục khả vi** (continuously differentiable) trong vùng lân cận điểm xét.
- Gradient $\nabla f(x)$ phải tồn tại để chỉ hướng dốc cục bộ.
- Khi đó, phép cập nhật
$$x_{t+1} = x_t - \eta \nabla f(x_t)$$
đảm bảo rằng với η đủ nhỏ, $f(x_{t+1}) < f(x_t)$.

Tóm lại: Gradient chỉ hướng tăng nhanh nhất của hàm, nên để giảm hàm, ta phải **đi ngược chiều gradient**.

1.4 Case study: Mean Absolute Error

❖ **Case study: Mean Absolute Error**



$$f'(a) = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}$$

$$f'(x) = \frac{x}{|x|} \quad \text{for } x \neq 0$$

Cho hàm $f(x) = |x|$. Tại $x > 0$, ta có:

$$\lim_{x \to 0^+} f'(x) = 1.$$

Tại $x < 0$, ta có:

$$\lim_{x \to 0^-} f'(x) = -1.$$

Như vậy, vì hai giới hạn một bên không bằng nhau, nên hàm **không khả vi tại** $x = 0$. Về mặt lý thuyết, ta không thể áp dụng **Gradient Descent** tại điểm này vì đạo hàm không tồn tại, mặc dù $x = 0$ chính là vị trí tối ưu. Trong thực tế, ta xử lý bằng cách **cài đặt subgradient** hoặc dùng **hàm làm trơn** (như Huber) để có thể áp dụng tối ưu hóa cho các hàm như *MAE* và *MSE*.

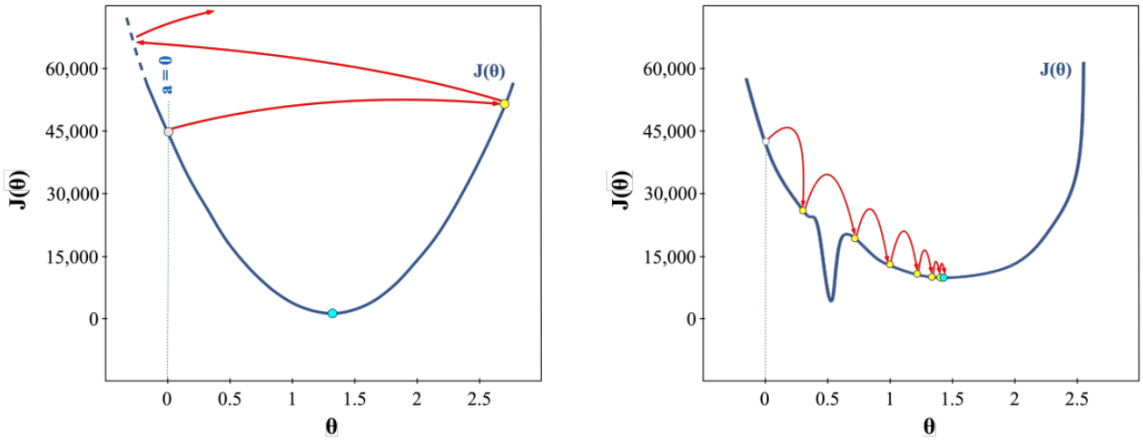
1.5 Learning rate

Learning rate (tốc độ học) là hệ số η điều chỉnh độ lớn của mỗi bước cập nhật trong thuật toán Gradient Descent. Nếu η **quá lớn**, điểm tham số θ có thể **nhảy vượt qua cực tiểu** (hình trái), khiến

quá trình huấn luyện dao động mạnh hoặc phân kỳ. Ngược lại, nếu η **quá nhỏ**, việc cập nhật diễn ra chậm, mất nhiều thời gian để hội tụ.

Vì vậy, ta cần chọn **learning rate phù hợp** để mô hình hội tụ ổn định về điểm cực tiểu toàn cục hoặc cực tiểu cục bộ (hình phải). Khi quan sát biểu đồ *loss* trong quá trình huấn luyện:

- Nếu đường loss giảm chậm và đi ngang → learning rate đang quá nhỏ.
- Nếu loss dao động hoặc tăng dần → mô hình có thể đang **phân kỳ**, cần **giảm learning rate**.



Hình 1: **Ảnh hưởng của tốc độ học (Learning Rate).** Hình trái: learning rate quá lớn khiến mô hình nhảy qua cực tiểu và không hội tụ. Hình phải: với hàm *không lồi (non-convex)*, learning rate lớn khiến mô hình bỏ qua điểm tối ưu toàn cục (global optimum) và chỉ hội tụ đến điểm tối ưu cục bộ (local optimum).

1.6 Tối ưu hóa hàm bình phương

Cho hàm bình phương $f(x) = x^2$, ta thực hiện các bước lặp như sau:

- **Khởi tạo** giá trị ban đầu x_0.
- **Tính đạo hàm** tại điểm hiện tại x_t: $f'(x_t) = 2x_t$.
- **Cập nhật** x bằng cách **di chuyển ngược hướng với đạo hàm**:

$$x_{t+1} = x_t - \eta f'(x_t)$$

  trong đó η là **tốc độ học (learning rate)**.

Quá trình được lặp lại cho đến khi x_t hội tụ về điểm cực tiểu $x^* = 0$, tức là vị trí mà $f'(x^*) = 0$.

1.7 Tối ưu hóa hàm 2D

Xét hàm hai biến:

$$f(x, y) = x^2 + y^2$$

Ta vẫn áp dụng phương pháp Gradient Descent tương tự như với hàm một biến. Ý tưởng là dùng đạo hàm riêng phần cho từng biến — tức là khi tính đạo hàm theo x, ta xem y là hằng số, và ngược lại.

$$\frac{\partial f}{\partial x} = 2x, \qquad \frac{\partial f}{\partial y} = 2y$$

Tại mỗi bước lặp, ta cập nhật từng biến theo công thức:

$$\begin{cases} x_{t+1} = x_t - \eta \frac{\partial f}{\partial x}(x_t, y_t) \\ y_{t+1} = y_t - \eta \frac{\partial f}{\partial y}(x_t, y_t) \end{cases}$$

Trong đó η là **learning rate** (tốc độ học). Phương pháp này còn được gọi là **tối ưu hóa tách biến (coordinate-wise optimization)** hoặc hiểu theo cách trực quan là "*chia để trị*": ta lần lượt tối ưu từng biến, trong khi giữ các biến còn lại cố định.

1.8 Hàm hợp và chain rule

Xét hai hàm số:

$$f : x \mapsto f(x), \quad g : f \mapsto g(f)$$

Khi đó, **hàm hợp** $h(x)$ được xác định bởi:

$$h(x) = g(f(x))$$

Mục tiêu là tìm đạo hàm của $h(x)$ theo x. Theo **quy tắc dây chuyền (Chain Rule)**:

$$\frac{d}{dx} g(f(x)) = \frac{dg}{df} \cdot \frac{df}{dx}$$

Dựa vào chain rule:

- Biến x đi qua hàm f để tạo ra giá trị trung gian $f(x)$.
- Sau đó, $f(x)$ được đưa vào hàm g để sinh ra $g(f(x))$.
- Khi x thay đổi, cả f và g đều thay đổi — do đó ta nhân hai tốc độ biến thiên lại:

$$\text{tốc độ thay đổi của } g \text{ theo } x = \left(\frac{dg}{df}\right)\left(\frac{df}{dx}\right)$$

Ví dụ minh họa:

$$f(x) = 2x - 1, \quad g(f) = f^2$$

Khi đó:

$$\frac{d}{dx} g(f(x)) = \frac{dg}{df} \cdot \frac{df}{dx} = (2f) \cdot (2) = 4(2x - 1)$$

1.9 Giả định của mô hình hồi quy tuyến tính

Mô hình hồi quy tuyến tính là một trong những công cụ thống kê phổ biến và mạnh mẽ nhất, nhưng để các kết quả ước lượng đáng tin cậy, mô hình cần thoả mãn một số **giả định cơ bản**. Nếu các giả định này bị vi phạm, các hệ số hồi quy, kiểm định và dự đoán có thể trở nên *sai lệch hoặc không còn ý nghĩa thống kê.*

1.9.1 Giả định 1: Tuyến tính (Linearity)

Ý nghĩa: Mối quan hệ giữa các biến độc lập X_i và biến phụ thuộc Y được giả định là tuyến tính, được mô tả bởi phương trình:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \varepsilon$$

Trong đó:

- β_i: hệ số hồi quy, biểu thị mức thay đổi trung bình của Y khi X_i tăng một đơn vị (giữ các biến khác không đổi).
- ε: sai số ngẫu nhiên.

Ví dụ: Giả sử ta dự đoán mức đường huyết (mg/dL) dựa trên *tuổi* và *BMI*:

$$\text{BloodGlucose} = 50 + 0.9 \times \text{Age} + 2.1 \times \text{BMI} + \varepsilon$$

Điều này ngụ ý rằng mỗi khi BMI tăng 1 đơn vị, mức đường huyết trung bình tăng khoảng 2.1 mg/dL, nếu các yếu tố khác giữ nguyên.

Kiểm tra: Quan sát biểu đồ phần dư (residual plot) hoặc scatter plot giữa phần dư và giá trị dự đoán $\hat{Y}$. Nếu xuất hiện dạng cong, giả định tuyến tính có thể bị vi phạm. Có thể khắc phục bằng cách thêm các biến bậc hai hoặc sử dụng mô hình hồi quy phi tuyến.

1.9.2 Giả định 2: Không đa cộng tuyến (No Multicollinearity)

Ý nghĩa: Các biến độc lập không được có tương quan mạnh với nhau. Nếu có, mô hình sẽ khó xác định ảnh hưởng riêng của từng biến.

Ví dụ:

$$\text{Weight} = \beta_0 + \beta_1 \text{Height} + \beta_2 \text{ArmLength} + \varepsilon$$

Nếu chiều cao và chiều dài cánh tay có hệ số tương quan $r = 0.95$, ta gặp hiện tượng đa cộng tuyến.

Phát hiện: Sử dụng *Variance Inflation Factor (VIF)*:

$$\text{VIF}_i = \frac{1}{1 - R_i^2}$$

Nếu VIF > 10, cho thấy đa cộng tuyến mạnh. Cách xử lý: loại bỏ một biến, hoặc gộp hai biến tương quan cao thành một chỉ số tổng hợp.

1.9.3 Giả định 3: Phần dư có phân phối chuẩn (Normality of Errors)

Ý nghĩa: Phần dư $\varepsilon_i = y_i - \hat{y}_i$ được giả định tuân theo phân phối chuẩn:

$$\varepsilon_i \sim N(0, \sigma^2)$$

Ví dụ: Trong mô hình dự đoán huyết áp dựa trên tuổi và cân nặng, các sai số nên phân bố đối xứng quanh 0.

Kiểm tra:

- Biểu đồ Q–Q plot (Quantile–Quantile).
- Kiểm định Shapiro–Wilk hoặc Kolmogorov–Smirnov.

Khắc phục: Biến đổi biến phụ thuộc, chẳng hạn:

$$Y' = \log(Y) \quad \text{hoặc} \quad Y' = \sqrt{Y}$$

1.9.4 Giả định 4: Phương sai không đổi (Homoscedasticity)

Ý nghĩa: Phương sai của sai số là hằng số cho mọi giá trị của X:

$$Var(\varepsilon_i) = \sigma^2$$

Nếu phương sai thay đổi theo X, hiện tượng này được gọi là *heteroscedasticity*.

Ví dụ: Trong mô hình dự đoán thu nhập theo số năm kinh nghiệm, phần dư ở nhóm thu nhập cao thường phân tán hơn → phương sai thay đổi.

Kiểm tra:

- Biểu đồ phần dư so với giá trị dự đoán $\hat{y}$.
- Kiểm định Breusch–Pagan hoặc White test.

Khắc phục:

- Sử dụng *Weighted Least Squares (WLS)*.
- Biến đổi logarit biến phụ thuộc.

1.9.5 Giả định 5: Không có tự tương quan (No Autocorrelation)

Ý nghĩa: Sai số của các quan sát phải độc lập với nhau. Nếu phần dư ở thời điểm t có xu hướng tương tự với $t + 1$, mô hình vi phạm giả định này.

Ví dụ: Trong chuỗi thời gian, doanh số tháng này thường bị ảnh hưởng bởi doanh số tháng trước → có tự tương quan.

Kiểm tra: *Durbin–Watson Test:*

$$DW = \frac{\sum_{t=2}^{n}(e_t - e_{t-1})^2}{\sum_{t=1}^{n} e_t^2}$$

Giá trị DW gần 2 là tốt (dao động trong khoảng 0–4).

Khắc phục:

- Thêm biến trễ (*lag variables*) như Y_{t-1} hoặc X_{t-1}.
- Sử dụng mô hình hồi quy chuỗi thời gian như ARIMA hoặc GLS.

0.1 Tổng kết

Nếu bất kỳ giả định nào bị vi phạm, mô hình hồi quy tuyến tính có thể không còn phù hợp, và cần xem xét các phương pháp thay thế như hồi quy phi tuyến, hồi quy robust hoặc biến đổi dữ liệu phù hợp hơn.

Bảng 1: Tóm tắt các giả định của mô hình hồi quy tuyến tính

Giả định	**Ý nghĩa**	**Cách kiểm tra**	**Khắc phục**
Tuyến tính	Quan hệ giữa X và Y là tuyến tính	Scatter plot, Residual plot	Thêm biến bậc hai, log-transform
Không đa cộng tuyến	Các biến độc lập không tương quan mạnh	VIF	Loại biến, gộp biến, PCA
Phân phối chuẩn phần dư	Sai số có phân phối chuẩn, trung bình 0	Q–Q plot, Shapiro–Wilk	Log-transform, robust model
Phương sai không đổi	Phần dư có phương sai ổn định	Residual vs fitted plot	Weighted least squares
Không tự tương quan	Sai số độc lập theo thời gian	Durbin–Watson test	Thêm biến trễ, mô hình ARIMA

II. Problem Solving

2.1 Solution cho vấn đề 1 – Hàm chỉ có một biến a

Xét hàm một biến:

$$f(a) = ax$$

Như đã thảo luận ở phần trên, ta chỉ cần thực hiện ba bước:

1. Khởi tạo giá trị ban đầu của a.
2. Tính đạo hàm theo a: $\frac{df}{da} = x$.
3. Cập nhật giá trị a theo quy tắc:

$$a_{t+1} = a_t - \eta \frac{df}{da}$$

2.2 Solution cho vấn đề 2 – Hàm có thêm biến b

Xét hàm có biến b:

$$f(b) = x + b$$

Ta cũng thực hiện tương tự:

1. Khởi tạo b.
2. Tính đạo hàm theo b: $\frac{df}{db} = 1$.
3. Cập nhật b theo hướng ngược với đạo hàm:

$$b_{t+1} = b_t - \eta \frac{df}{db}$$

2.3 Solution cho vấn đề 3 – Hàm có cả hai biến a và b

Xét hàm hai biến:

$$f(a, b) = ax + b$$

Ta khởi tạo đồng thời hai tham số a và b, sau đó tính đạo hàm riêng phần cho từng biến:

$$\frac{\partial f}{\partial a} = x, \qquad \frac{\partial f}{\partial b} = 1$$

và cập nhật từng biến độc lập:

$$\begin{cases} a_{t+1} = a_t - \eta \frac{\partial f}{\partial a} \\ b_{t+1} = b_t - \eta \frac{\partial f}{\partial b} \end{cases}$$

Như vậy, ta đã mở rộng từ bài toán 1 biến sang nhiều biến bằng cách đạo hàm và cập nhật từng tham số riêng biệt.

III: Towards Linear Regression

3.1. Bài toán hai samples

Xét hàm tuyến tính:

$$f(a, b) = ax + b$$

với hai cặp dữ liệu mẫu:

$$(x_1, y_1) = (-2, -1), \qquad (x_2, y_2) = (5, 13)$$

Mục tiêu là tìm giá trị a, b sao cho hàm $f(a, b)$ **tiệm cận tốt nhất** với các giá trị thực y_i.
Quy trình tối ưu:

1. **Khởi tạo** giá trị ban đầu cho a, b (ví dụ $a = 0$, $b = 0$).
2. Với mỗi sample (x_i, y_i):
   - Tính giá trị dự đoán: $f_i = ax_i + b$
   - Tính sai số: $g(f_i) = (f_i - y_i)^2$
3. **Đạo hàm riêng phần:**
$$\frac{\partial g}{\partial a} = 2(f_i - y_i)x_i, \qquad \frac{\partial g}{\partial b} = 2(f_i - y_i)$$
4. **Cập nhật tham số:**
$$\begin{cases} a_{t+1} = a_t - \eta \sum_i \frac{\partial g}{\partial a} \\ b_{t+1} = b_t - \eta \sum_i \frac{\partial g}{\partial b} \end{cases}$$

   với $\eta = 0.1$ là **learning rate**.

3.2. Các cách thức tối ưu hóa dựa vào Gradient

Trong thực tế, khi có nhiều dữ liệu (x_i, y_i), ta có thể tối ưu theo hai hướng khác nhau:
(1) Cập nhật từng mẫu (Sample by Sample) hoặc
(2) Cập nhật sau khi tính trung bình Gradient trên nhiều mẫu.
Approach 1 – Tối ưu theo từng sample (Stochastic Gradient Descent)

- Mỗi lần, ta chọn một cặp dữ liệu (x_i, y_i).
- Tính giá trị dự đoán $f_i = ax_i + b$ và sai số $g_i = (f_i - y_i)^2$.
- Đạo hàm riêng phần:

$$\frac{dg_i}{da} = \frac{dg_i}{df_i} \cdot \frac{df_i}{da}, \qquad \frac{dg_i}{db} = \frac{dg_i}{df_i} \cdot \frac{df_i}{db}.$$

- Cập nhật tham số a, b sau mỗi mẫu:

$$a_{t+1} = a_t - \eta \frac{dg_i}{da}, \qquad b_{t+1} = b_t - \eta \frac{dg_i}{db}.$$

Đây là phương pháp **Stochastic Gradient Descent (SGD)**, giúp mô hình cập nhật nhanh và có thể vượt qua local minima nhờ tính "ngẫu nhiên" của từng mẫu.
Approach 2 – Tối ưu trung bình nhiều sample (Batch / Mini-batch Gradient Descent)

- Lấy một nhóm (hoặc toàn bộ) N samples để tính trung bình Gradient:

$$\frac{1}{N}\sum_i \frac{dg_i}{da} = \frac{1}{N}\sum_i \left(\frac{dg_i}{df_i} \cdot \frac{df_i}{da}\right), \qquad \frac{1}{N}\sum_i \frac{dg_i}{db} = \frac{1}{N}\sum_i \left(\frac{dg_i}{df_i} \cdot \frac{df_i}{db}\right).$$

- Sau đó cập nhật a, b cùng lúc bằng trung bình gradient:

$$a_{t+1} = a_t - \eta \frac{1}{N}\sum_i \frac{dg_i}{da}, \qquad b_{t+1} = b_t - \eta \frac{1}{N}\sum_i \frac{dg_i}{db}.$$

Phương pháp này được gọi là **Batch Gradient Descent** (nếu dùng toàn bộ dữ liệu) hoặc **Mini-batch Gradient Descent** (nếu chỉ dùng một nhóm nhỏ các mẫu tại mỗi bước).
So sánh hai phương pháp:

Đặc điểm	**Approach 1 (SGD)**	**Approach 2 (Batch / Mini-batch)**
Cập nhật	Sau mỗi sample	Sau khi trung bình nhiều sample
Tốc độ cập nhật	Nhanh hơn, nhiễu hơn	Ổn định hơn, chậm hơn
Độ hội tụ	Dao động quanh cực tiểu	Mượt, ổn định
Tính ngẫu nhiên	Cao	Thấp

Thảo luận mở rộng: Vấn đề Normalization
Khi ta thay đổi thang đo của đầu vào x, ví dụ như:

$$x' = 100x$$

thì hàm tuyến tính:

$$f(a, b) = ax + b$$

sẽ trở thành:

$$f'(a,b) = a(100x) + b = (100a)x + b$$

Điều này có nghĩa là **giá trị đạo hàm riêng phần theo** a cũng thay đổi 100 lần:

$$\frac{\partial f'}{\partial a} = 100x$$

$\Rightarrow$ gradient trở nên lớn hơn gấp nhiều lần, khiến quá trình cập nhật bước nhảy (update step)

$$a_{t+1} = a_t - \eta \frac{\partial f}{\partial a}$$

trở nên quá mạnh hoặc thậm chí gây **phân kỳ** (divergence).
Kết luận:

- Việc **thay đổi độ lớn của input** x **ảnh hưởng trực tiếp đến gradient**, làm thay đổi tốc độ cập nhật tham số.
- Để đảm bảo các biến có cùng "đơn vị thay đổi", ta cần **chuẩn hóa (Normalization)** hoặc **chuẩn hóa z-score (Standardization)**:

$$x_{\text{norm}} = \frac{x - \mu}{\sigma}$$

- Khi dữ liệu được chuẩn hóa, gradient sẽ ổn định hơn, giúp mô hình hội tụ nhanh và ổn định.

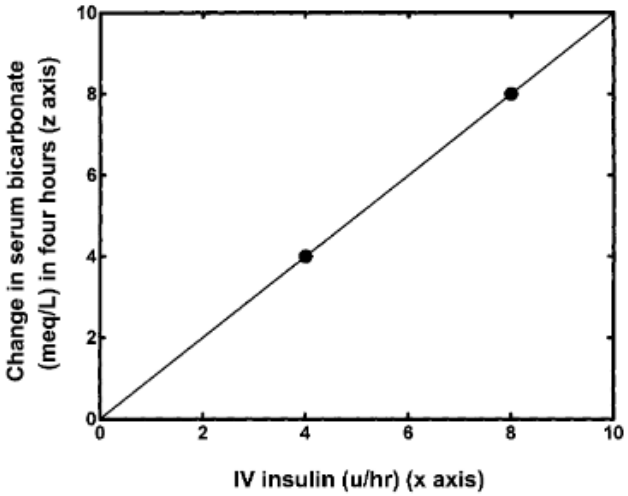
IV. Multiple Linear Regression in Medical [?]

4.1 Giới thiệu

Hồi quy tuyến tính bội là tổng quát hoá của hồi quy tuyến tính đơn khi có nhiều hơn một biến dự báo. Nếu nhà nghiên cứu nghi ngờ rằng kết cục quan tâm có thể liên quan hoặc phụ thuộc vào hơn một biến dự báo, cách tiếp cận hồi quy đơn có thể không phù hợp. Khi đó, một mô hình hồi quy bội cho phép xem xét đồng thời nhiều biến dự báo có thể được sử dụng.

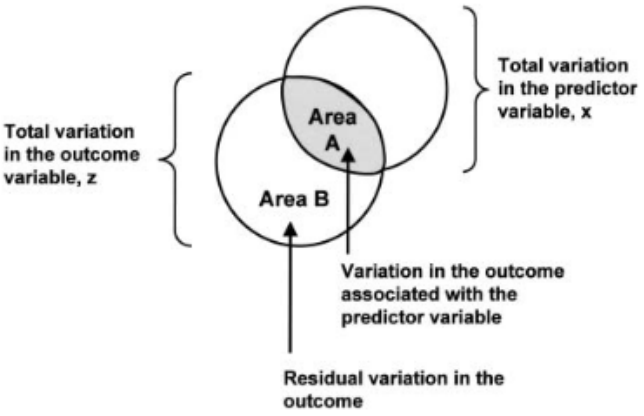
Các ứng dụng của hồi quy tuyến tính đơn trong nghiên cứu y học còn hạn chế, vì trong hầu hết tình huống tồn tại nhiều biến dự báo liên quan. Các kỹ thuật thống kê đơn biến như hồi quy tuyến tính đơn chỉ sử dụng một biến dự báo; chúng có thể đúng về mặt toán học nhưng lại gây hiểu lầm về mặt lâm sàng. Hồi quy tuyến tính bội là một kỹ thuật toán học dùng để mô hình hoá mối liên hệ giữa nhiều biến độc lập (dự báo) và một biến phụ thuộc (kết cục). Kỹ thuật này được dùng trong nghiên cứu y học để mô hình hoá dữ liệu quan sát, cũng như trong các nghiên cứu chẩn đoán và điều trị nơi mà kết cục phụ thuộc vào nhiều yếu tố. Mặc dù nhìn chung bị giới hạn ở dữ liệu có thể biểu diễn bằng hàm tuyến tính, hồi quy bội hưởng lợi từ khung toán học phát triển tốt, cung cấp nghiệm duy nhất và khoảng tin cậy chính xác cho các hệ số hồi quy.

Ví dụ: trong Hồi quy tuyến tính đơn 2, nhà nghiên cứu khảo sát mối liên hệ giữa cường độ điều trị insulin và mức độ cải thiện toan hoá máu (acidosis) ở bệnh nhân nhiễm toan ceton đái tháo đường (DKA). Việc cải thiện toan hoá dường như phụ thuộc vào cường độ insulin, nhưng có thể còn nhiều yếu tố quan trọng khác: mức độ nặng ban đầu của đợt DKA, mức độ nặng bệnh nền, truyền dịch tĩnh mạch (IV) v.v. Hồi quy tuyến tính bội cho phép đưa tất cả các yếu tố tiềm tàng này vào *một* mô hình, từ đó ước lượng chính xác hơn đóng góp của từng yếu tố với kết cục, lẫn mối liên hệ tổng thể giữa chúng với kết cục, và cả quan hệ giữa các biến dự báo với nhau.

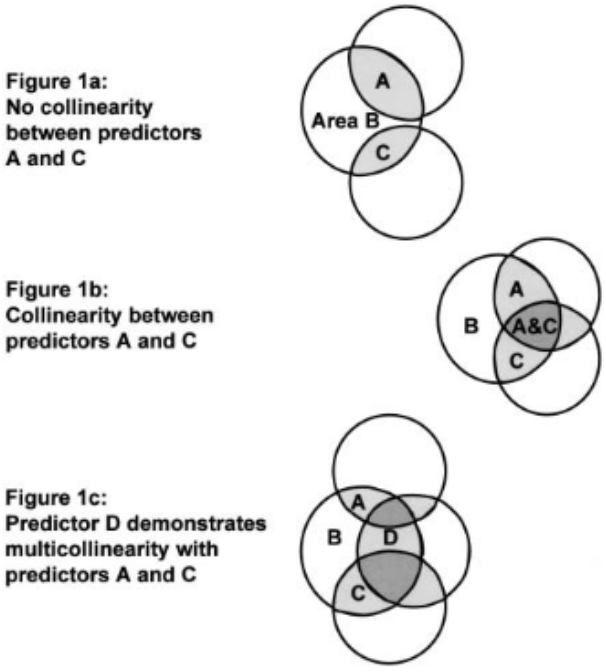


Hình 2: $z = 1x$, $R^2 = 1.0$. IV = intravenous.

Về mặt sơ đồ 3 và 0.1, việc thêm một biến dự báo tương ứng với thêm một "hình tròn" chồng lấn lên hình tròn của biến kết cục; vùng chồng lấn mới ký hiệu là C (Hình 1A, 1B). Nhìn chung, thêm biến dự báo sẽ tăng phần kết cục được giải thích (A + C) và giảm phần sai số/dư (B). Tuỳ quan hệ giữa hai biến dự báo, hai hình tròn dự báo có thể chồng lên nhau (A và C chồng lấn), phản ánh mức độ dư thừa và cộng tuyến trong mô hình.



Hình 3: Simple linear regression schematic



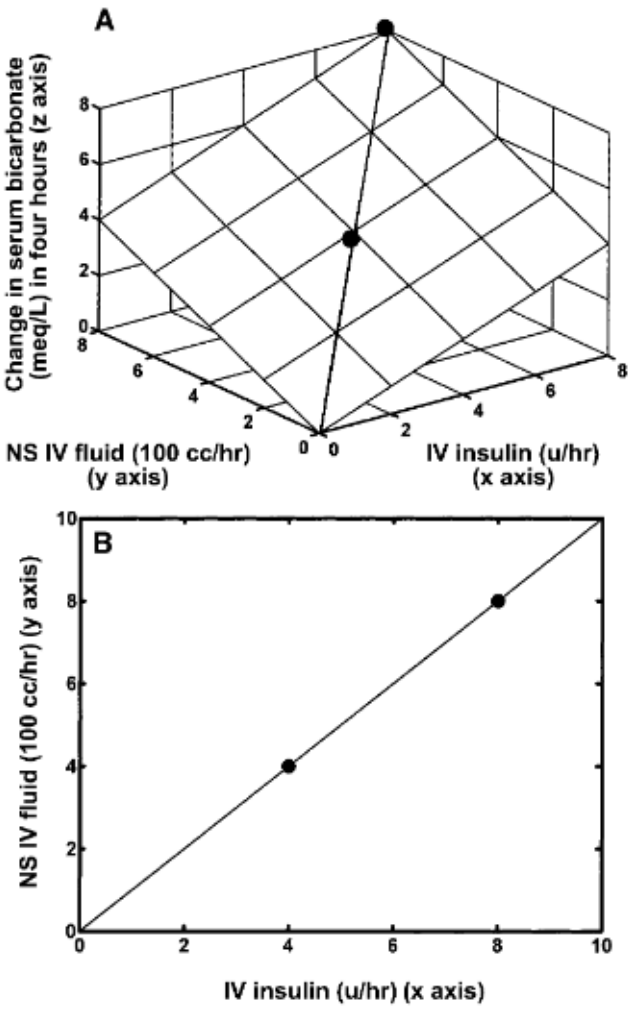
Hình 4: Multivariate schematics

4.2 Mô hình hồi quy tuyến tính bội

Mô hình bội xây dựng trên nền tảng của hồi quy đơn; bốn giả định cơ bản của hồi quy tuyến tính đơn cũng cần đúng đối với hồi quy bội. Bên cạnh các khái niệm của hồi quy đơn (vẫn phù hợp), ta cần bổ sung một số khái niệm mới. Để trực quan, xét trường hợp có *hai* biến dự báo và *một* biến kết cục (tổng ba biến), khi đó có thể hình dung trong không gian 3D. Với nhiều hơn hai biến dự báo, nguyên lý tương tự nhưng khó hình dung hơn.

Phương trình mô hình bội (mặt phẳng hồi quy) với biến kết cục z và hai biến dự báo x, y:

$$z = k_1 x + k_2 y + c \tag{1}$$



Hình 5: (A) $z = 0.5x + 0.5y$, $R^2 = 1.0$. (B) $y = 1x$, $R^2_{\text{pred}} = 1.0$. NS = normal saline; IV = intravenous.

trong đó k_1, k_2 là các hệ số (độ nghiêng theo trục x và y), còn c là tung độ gốc của z khi $x = y = 0$. Vì z là hàm *tuyến tính* theo từng biến dự báo nên bề mặt là *mặt phẳng* (không cong).

Tương tự hồi quy đơn, mặt phẳng tối ưu được ước lượng theo nguyên lý *bình phương tối thiểu*, bằng cách tối thiểu hoá tổng bình phương phần dư SS_{res}. Khi đó, vẫn có:

$$SS_{\text{tot}} = SS_{\text{reg}} + SS_{\text{res}}, \qquad R^2 = \frac{SS_{\text{reg}}}{SS_{\text{tot}}}.$$

Tuy nhiên, SS_{reg} ở đây là đóng góp *tổng hợp* của *cả hai* biến dự báo, và R được gọi là *hệ số tương quan bội*. Hệ số xác định R^2 cho biết tỷ lệ biến thiên của z được giải thích bởi *toàn bộ* các biến dự báo trong mô hình tuyến tính. Trên sơ đồ (0.1), $R^2 = \dfrac{A + C}{A + B + C}$ (phần chồng lấn chỉ được tính một lần).

Trong hồi quy đơn, kiểm định ý nghĩa của quan hệ hồi quy tương đương kiểm định t với tỷ số giữa hệ số dốc và sai số chuẩn. Trong hồi quy bội, vì có nhiều hệ số dốc, kiểm định ý nghĩa tổng thể dùng *ANOVA* để so sánh đóng góp mô hình với phần dư, điều chỉnh theo số biến dự báo và số quan sát:

$$F = \frac{MS_{\text{reg}}}{MS_{\text{res}}} = \frac{SS_{\text{reg}}/k_{\text{tot}}}{SS_{\text{res}}/(n - k_{\text{tot}} - 1)}.$$

Nếu F đủ lớn (theo bậc tự do phù hợp), bác bỏ giả thuyết không: mô hình tuyến tính bội giải thích được một phần đáng kể biến thiên của kết cục.

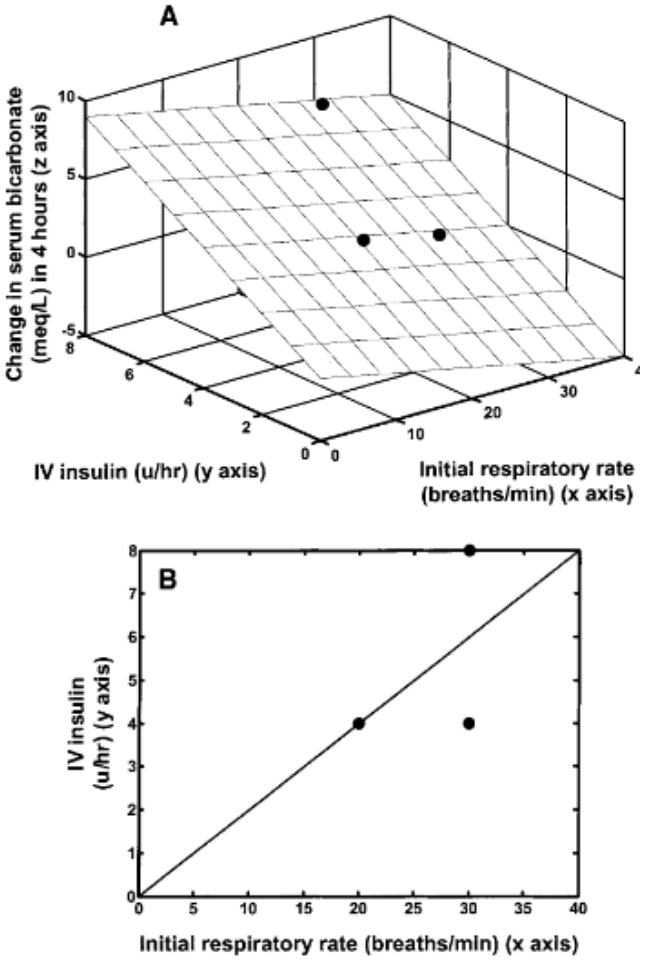
4.3 Quan hệ giữa các biến dự báo

Khác biệt quan trọng giữa hồi quy đơn và bội là: mô hình bội *bao gồm cả quan hệ tuyến tính giữa các biến dự báo với nhau*. Những quan hệ này — gọi là **đa cộng tuyến** (multicollinearity) — có thể ảnh hưởng mạnh đến hệ số mô hình và độ chính xác (SE) của chúng.

4.3.1 Cộng tuyến

Trong hình 2, nhà nghiên cứu đã khảo sát mối liên hệ giữa **cường độ điều trị insulin tĩnh mạch** (IV) và **tốc độ cải thiện tình trạng nhiễm toan ceton đái tháo đường (DKA)** ở hai bệnh nhân tiểu đường. Kết quả cho thấy biến dự báo và biến kết cục có mối quan hệ tỷ lệ thuận: cứ mỗi *một đơn vị insulin mỗi giờ* được truyền, nồng độ bicarbonate huyết thanh tăng thêm *1 mEq/L* sau bốn giờ điều trị.

Nhà nghiên cứu sau đó quay lại bộ dữ liệu và xem xét thêm liệu **cường độ truyền dung dịch muối sinh lý (normal saline – NS)** qua đường tĩnh mạch có liên quan đến tốc độ hồi phục DKA hay không.



Hình 6: (A) $z = -0.1x + 1.25y - 1$, $R^2 = 1.0$. (B) $y = 0.2x$, $R^2_{\text{pred}} = 0.25$. IV = intravenous.

Biến kết cục (mức tăng nồng độ bicarbonate huyết thanh sau 4 giờ) được biểu diễn dưới dạng hàm của hai biến dự báo: *cường độ truyền insulin* và *cường độ truyền dịch NS*, như thể hiện trong Hình 2A (6). Có thể thấy rằng, tốc độ cải thiện DKA dường như liên quan đến cả hai yếu tố này.

Trong mô hình hồi quy đơn ban đầu, mối quan hệ giữa insulin và cải thiện bicarbonate được biểu diễn là:

$$z = x.$$

Liệu mô hình có hai biến dự báo đúng đắn có thể là:

$$z = x + y \ ?$$

Câu trả lời là **không**. Nếu cường độ insulin là 4 đơn vị/giờ và tốc độ truyền dịch NS là 4 (tính theo 100 mL/giờ), thì giá trị dự đoán của bicarbonate sẽ là $z = 4 + 4 = 8$ mEq/L sau bốn giờ điều trị — gấp đôi so với thực tế quan sát (4 mEq/L). Như vậy, mô hình này đã *phóng đại* hiệu ứng điều trị.

Một mô hình chính xác hơn sẽ là:

$$z = \tfrac{1}{2}x + \tfrac{1}{2}y.$$

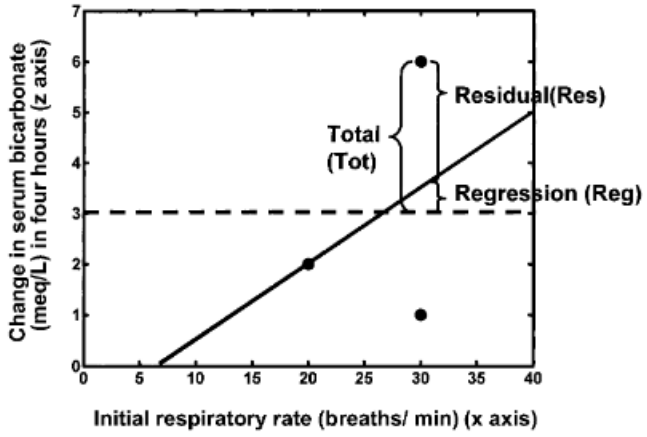
Khi thêm biến y vào mô hình, hệ số của biến x (insulin) giảm 50%, từ 1 còn 0.5. Lý do là dữ liệu cho thấy có hiện tượng **cộng tuyến** giữa x và y trong mặt phẳng (x, y).

Các Hình 2B đến 5B là các đồ thị hai chiều của cùng dữ liệu như trong Hình 2A–5A, nhưng chỉ thể hiện trục x và y. Cách biểu diễn này giúp quan sát rõ hơn mối quan hệ giữa hai biến dự báo. Trong Hình 2B, có thể thấy giá trị y tăng cùng với giá trị x: bệnh nhân được truyền nhiều insulin hơn cũng đồng thời được truyền nhiều dịch NS hơn. Hai biến này cùng tăng tuyến tính — một ví dụ của **cộng tuyến dương**.

Trong mô hình hồi quy đơn (2), toàn bộ sự cải thiện bicarbonate được quy cho insulin. Nhưng trong mô hình hồi quy bội (Hình A 5), mức cải thiện này được **phân chia đều** cho hai yếu tố: insulin và NS. Khi hiện tượng cộng tuyến xuất hiện, độ lớn của các hệ số hồi quy sẽ thay đổi tuỳ vào biến nào được đưa vào mô hình. Khi hai biến cùng tăng theo hướng dương, hệ số của mỗi biến có xu hướng giảm — bởi vì ảnh hưởng tới kết cục phải được "chia sẻ" giữa các biến dự báo.

4.3.2 Thí dụ mở rộng về thay đổi dấu hệ số

Với mô hình tuyến tính đơn, nhà nghiên cứu giả thuyết rằng **nhịp thở ban đầu cao hơn** có thể liên quan đến mức cải thiện DKA lớn hơn. Trong Hình 7, dữ liệu cho thấy bệnh nhân có nhịp thở cao hơn thực sự cải thiện nhiều hơn.



Hình 7: $z = 0.15x - 1$, $R^2 = 0.11$.

Tuy nhiên, ở mô hình tuyến tính bội, nhà nghiên cứu nhận ra rằng cần đưa thêm **cường độ insulin** vào mô hình. Hình 6 minh hoạ mối quan hệ giữa **mức tăng bicarbonate huyết thanh** và hai biến dự báo: **nhịp thở ban đầu** (x) và **cường độ insulin** (y).

Kết quả cho thấy mức tăng bicarbonate vẫn cao hơn ở bệnh nhân được truyền nhiều insulin, nhưng thú vị là, mức cải thiện lại *giảm* ở bệnh nhân có nhịp thở ban đầu cao. Nghĩa là, dấu của hệ số x (liên quan nhịp thở ban đầu) đã **chuyển từ dương sang âm**.

Một lần nữa, điều này xảy ra vì có hiện tượng **cộng tuyến dương** giữa hai biến: nhịp thở ban đầu cao hơn đồng nghĩa với bệnh nặng hơn, và do đó bệnh nhân được điều trị insulin tích cực hơn. Mối liên hệ này được thể hiện trong đồ thị giữa hai biến x và y (Hình 6B). Như vậy, sự cải thiện bicarbonate từng được quy cho "thở nhanh hơn" trong phân tích đơn biến thực ra đến từ việc điều trị insulin mạnh hơn. Sau khi phân tích cùng bộ dữ liệu nhưng với hai biến dự báo thay vì một, nhà nghiên cứu kết luận rằng không còn bằng chứng cho thấy bệnh nhân có thể *tự cải thiện DKA bằng cách tăng nhịp thở*.

4.3.3 Kết luận về hiện tượng cộng tuyến

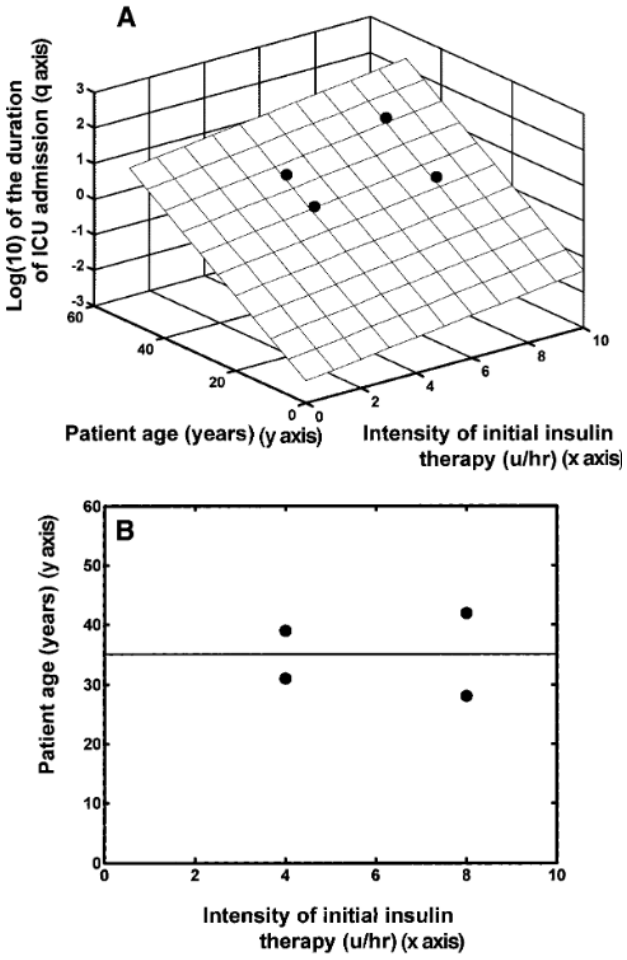
Các ví dụ trên minh hoạ **tác động của cộng tuyến và nhiễu đồng biến (confounding)** giữa hai biến dự báo, được biểu diễn trong sơ đồ Hình 5B, nơi hai vùng dự báo A và C chồng lấn.

Thuật ngữ **đa cộng tuyến** (*multicollinearity*) được dùng để mô tả các hiệu ứng cộng tuyến tương tự nhưng xảy ra giữa ba hoặc nhiều biến dự báo. Cả hai ví dụ đều thể hiện **cộng tuyến dương**. Trong một số trường hợp khác, giá trị của một biến dự báo có thể *giảm* khi biến kia *tăng* — khi đó gọi là **cộng tuyến âm**.

Trong các tập dữ liệu có nhiều biến dự báo, thường tồn tại các dạng *đa cộng tuyến phức tạp*, bao gồm cả các mối quan hệ cộng tuyến dương và âm đan xen lẫn nhau.

4.3.4 Không cộng tuyến

Như hình 8, nếu thêm biến tuổi (y) vào mô hình dự báo thời gian nằm ICU (log-transform), và tuổi *không* liên quan tuyến tính với cường độ insulin (x), thì hệ số theo x (insulin) giữ nguyên khi thêm y. Không có cộng tuyến $\Rightarrow$ mỗi biến đóng góp độc lập vào mô hình; thêm biến phù hợp thường làm giảm SS_{res} và *không* làm thay đổi hệ số đã có.



Hình 8: (A) $q = 0.097x + 0.074y - 2.37$, $R^2 = 1.0$. (B) $y = 0x + 35$, $R^2_{\text{pred}} = 0$. ICU = intensive care unit.

4.3.5 Đo lường mức độ cộng tuyến

Mức độ cộng tuyến giữa hai biến dự báo được định lượng bằng *hệ số xác định giữa các biến dự báo* R^2_{pred} (để phân biệt với R^2 giữa dự báo và kết cục). Với nhiều biến, R^2_{pred} của một biến là phần R^2 khi hồi quy biến đó theo *tất cả* các biến còn lại (tức mức chồng lấn của "hình tròn" biến đó với các hình còn lại trong hình 5).

4.4 Định lượng bất định của hệ số (SE) và VIF

Trong hồi quy đơn, sai số chuẩn (SE) của hệ số có dạng:

$$SE(\text{coef})_{\text{đơn}} = \left[\frac{SS_{\text{res}}}{n-2} \cdot \frac{1}{\sum(X-\bar{X})^2}\right]^{1/2}.$$

Trong hồi quy bội, công thức tương tự nhưng nhân thêm *căn bậc hai của hệ số phóng đại phương sai* (VIF) để phản ánh bất định tăng do cộng tuyến:

$$\text{VIF} = \frac{1}{1-R^2_{\text{pred}}}, \qquad SE(\text{coef})_{\text{bội}} = \left[\frac{SS_{\text{res}}}{n-k_{\text{tot}}-1} \cdot \frac{1}{\sum(X-\bar{X})^2} \cdot \frac{1}{1-R^2_{\text{pred}}}\right]^{1/2}.$$

Nếu thêm một biến *không cộng tuyến*, thường SS_{res} giảm (mô hình tốt hơn) $\Rightarrow$ SE các hệ số khác *giảm*. Ngược lại, thêm biến *cộng tuyến* làm VIF $> 1 \Rightarrow$ SE tăng. Tổng hiệu ứng là cân bằng giữa (i) thông tin giải thích mới giúp giảm SS_{res} và (ii) dư thừa/cộng tuyến làm tăng VIF.

4.5 Tăng lực (power)

Lực kiểm định phụ thuộc độ lớn hiệu ứng và bất định. Ngoài cách tăng n, một cách khác để *giảm* bất định là thêm các biến dự báo *không cộng tuyến* (tận dụng thêm thông tin trên mỗi quan sát), từ đó tăng power cho suy luận về các biến quan tâm (ví dụ chuyển từ kiểm định t đơn biến sang ANOVA hai yếu tố trong ví dụ truyền kali và bicarbonate).

4.6 Leverage trong mô hình đa biến

Đánh giá *leverage* trong đa biến tương tự đơn biến. *Cook's distance* phản ánh thay đổi *đồng thời* của tất cả các hệ số và tung độ gốc khi loại một điểm dữ liệu; giá trị bất thường gợi ý điểm có ảnh hưởng quá mức.

4.7 Hiệu ứng tương tác

Nếu tác động kết hợp của hai liệu pháp (ví dụ kali và bicarbonate) lớn hơn tổng riêng lẻ, ta đưa *tương tác* xy vào mô hình:

$$z = k_1 x + k_2 y + k_3\, xy + c. \tag{2}$$

Thuật ngữ tương tác $k_3\, xy$ làm bề mặt "cong" (không còn là mặt phẳng thuần tuý), phản ánh hiệu ứng hiệp đồng. Thêm tương tác vừa có thể *giảm* SS_{res} (thêm thông tin thực) vừa có thể *tăng* VIF (cộng tuyến với x hoặc y), do đó SE của các hệ số có thể tăng hoặc giảm tuỳ tương quan giữa hai hiệu ứng này.

4.8 Lựa chọn biến phù hợp (xây dựng mô hình)

Mục tiêu: đưa vào các biến dự báo cung cấp *thông tin độc lập đáng kể*, tránh dư thừa cộng tuyến quá mức. Cách tiếp cận dựa thuần tuý vào thống kê đơn biến (screening) hoặc chọn biến tự động (stepwise) dễ gây thiên lệch và sai lầm khoa học; không khuyến khích. Thực hành tốt: thiết kế thu thập dữ liệu cho *những biến nền tảng quan trọng* theo tri thức hiện tại; sau đó kiểm tra giả định mô hình tuyến tính, đóng góp độc lập của từng biến, và mức đa cộng tuyến. Cuối cùng, *xác thực* mô hình trên tập dữ liệu khác hoặc bằng *cross–validation*.

4.9 Xử lý đa cộng tuyến

Khi cộng tuyến là không tránh khỏi:

- Thu thập thêm dữ liệu (nếu khả thi).
- Kết hợp biến thành *chỉ số tổng hợp* (ví dụ thang điểm nguy cơ).
- Thay thế bằng các biến *căn nguyên hơn*, ít cộng tuyến hơn (tương tự ý tưởng phân tích thành phần chính — PCA).
- Dùng biến thể bình phương tối thiểu có *quy chuẩn hoá* như **ridge regression** (chấp nhận một ít chệch để giảm phương sai/SE khi cộng tuyến cao).

4.10 Kết luận

Đa số bài toán lâm sàng có bản chất đa biến; do đó tiếp cận đơn biến thường sai lệch, có thể dẫn đến hệ số dự báo không đúng (về lượng hoặc về dấu) và kết luận suy luận thiếu chính xác. Hồi quy tuyến tính bội là công cụ hữu ích để mô hình hoá nhiều hiện tượng trong nghiên cứu y học. Khi các giả định được thoả mãn, mô hình cung cấp ước lượng chính xác cho hệ số và sai số chuẩn của chúng, giúp hiểu rõ tầm quan trọng tương đối của các biến dự báo, và hỗ trợ tiên lượng cho dữ liệu tương lai. Ứng dụng bao gồm mô hình hoá chẩn đoán, tiên lượng và đáp ứng điều trị.

V. Vectorization

5.1 Khái niệm cơ bản

Trong huấn luyện mô hình hồi quy tuyến tính, mỗi mẫu huấn luyện có thể được biểu diễn dưới dạng:

$$\hat{y}^{(i)} = w_1 x_1^{(i)} + w_2 x_2^{(i)} + \cdots + w_n x_n^{(i)} + b$$

Nếu tính toán lần lượt từng mẫu (`for-loop`), thời gian thực thi sẽ tỷ lệ thuận với số mẫu N. Khi dữ liệu lớn, phương pháp này trở nên chậm và tốn tài nguyên. Giải pháp là **vectorization**, giúp gộp toàn bộ dữ liệu huấn luyện vào ma trận để tính toán song song bằng phép nhân ma trận thay vì vòng lặp.

5.2 Biểu diễn ma trận cho toàn bộ dữ liệu

Ta viết lại toàn bộ dữ liệu đầu vào X dưới dạng ma trận kích thước $N \times d$, trong đó:

$$X = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \dots & x_d^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \dots & x_d^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{(N)} & x_2^{(N)} & \dots & x_d^{(N)} \end{bmatrix}$$

Thêm cột bias ($x_0 = 1$) vào đầu để tính gộp hệ số chặn b, khi đó:

$$X_b = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \dots & x_d^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \dots & x_d^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(N)} & x_2^{(N)} & \dots & x_d^{(N)} \end{bmatrix}$$

Hệ số mô hình được viết gọn:

$$\theta = \begin{bmatrix} b \\ w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix}$$

Dự đoán cho toàn bộ dữ liệu được tính bằng:

$$\hat{\mathbf{y}} = X_b \theta$$

Phép nhân ma trận này thay thế toàn bộ vòng lặp N lần tính toán riêng lẻ, giúp tận dụng tối đa khả năng tính toán song song của CPU/GPU.

5.3 Hàm mất mát và đạo hàm dưới dạng vector

Hàm mất mát trung bình bình phương sai số (MSE):

$$L = \frac{1}{N}(\hat{\mathbf{y}} - \mathbf{y})^\top(\hat{\mathbf{y}} - \mathbf{y})$$

Đạo hàm vector hóa theo θ:

$$\nabla_\theta L = \frac{2}{N} X_b^\top (X_b\theta - \mathbf{y})$$

Cập nhật tham số theo Gradient Descent:

$$\theta \leftarrow \theta - \eta \nabla_\theta L$$

Vectorization giúp giảm đáng kể thời gian tính toán vì các phép nhân và cộng ma trận được tối ưu hóa ở mức phần cứng.

5.4 Ví dụ minh họa với NumPy

```
import numpy as np

# Generate a simple dataset
X = np.array([[1, 2],
              [2, 3],
              [3, 4],
              [4, 5]])  # 4 samples, 2 features
y = np.array([[5], [7], [9], [11]])

# Add bias term (x0 = 1)
X_b = np.c_[np.ones((X.shape[0], 1)), X]  # shape = (4, 3)

# Initialize parameters randomly
theta = np.random.randn(3, 1)
eta = 0.01       # learning rate
n_iter = 1000    # number of iterations

# Vectorized Gradient Descent
for epoch in range(n_iter):
    y_pred = X_b @ theta             # vectorized prediction
    error = y_pred - y               # compute residuals
    gradients = (2 / X_b.shape[0]) * X_b.T @ error
    theta -= eta * gradients         # parameter update

print("Learned parameters:\n", theta)
```

Listing 1: Vectorized Gradient Descent for Linear Regression using NumPy

5.5 Hiệu năng và lợi ích của vectorization

- **Tốc độ:** Phép nhân ma trận được tối ưu hóa trong các thư viện như BLAS/LAPACK.
- **Độ chính xác:** Giảm sai số làm tròn so với lặp từng phần tử.
- **Tính gọn:** Code ngắn gọn, dễ hiểu, tránh lỗi lập chỉ mục.
- **Khả năng mở rộng:** Tương thích với GPU và framework như TensorFlow, PyTorch.

5.6 Kiểm chứng kết quả với nghiệm đóng

Để kiểm chứng, ta so sánh với nghiệm đóng (Normal Equation):

$$\theta^* = (X_b^\top X_b)^{-1} X_b^\top \mathbf{y}$$

```
theta_best = np.linalg.inv(X_b.T @ X_b) @ X_b.T @ y
print("Closed-form theta:\n", theta_best)
```

Listing 2: Closed-form solution comparison

Nếu kết quả của θ từ Gradient Descent hội tụ đúng, hai nghiệm sẽ gần như trùng nhau.

5.7. Minh họa hình học (Geometric Interpretation of Linear Regression and Vectorization)

Linear Regression không chỉ là một bài toán tối ưu thuần túy mà còn có ý nghĩa hình học sâu sắc trong không gian Euclidean. Mỗi thành phần trong công thức vector hóa đều có thể được biểu diễn dưới dạng hình học, giúp ta hiểu rõ hơn cơ chế hội tụ, vai trò của vector trọng số, và cách mô hình tìm đường "phẳng tối ưu" giữa các điểm dữ liệu.

5.7.1. Không gian dự đoán và vector biểu diễn

Xét tập dữ liệu $X_b \in \mathbb{R}^{N\times(d+1)}$, trong đó mỗi hàng là một mẫu dữ liệu có thêm phần tử bias. Giả sử mỗi cột của X_b tương ứng với một đặc trưng (feature), khi đó không gian của tất cả các kết hợp tuyến tính của các cột này được gọi là **cột không gian** (*column space*) của X_b, ký hiệu là:

$$\mathcal{C}(X_b) = \{X_b\theta \mid \theta \in \mathbb{R}^{d+1}\}$$

Mỗi vector $\hat{\mathbf{y}} = X_b\theta$ là một điểm nằm trong không gian này, tức là tổ hợp tuyến tính của các cột trong X_b.

5.7.2. Vector dự đoán, sai số và gradient

Khi mô hình dự đoán $\hat{\mathbf{y}}$, ta có thể mô tả ba vector chính trong quá trình học:

$$\textbf{(1) Prediction Vector: } \hat{\mathbf{y}} = X_b\theta$$

$$\textbf{(2) Error Vector: } \mathbf{e} = \hat{\mathbf{y}} - \mathbf{y}$$

$$\textbf{(3) Gradient Vector: } \nabla_\theta = \frac{2}{N} X_b^\top \mathbf{e}$$

- $\hat{\mathbf{y}}$ là **vector dự đoán** nằm trong không gian cột của X_b.
- $\mathbf{e}$ là **vector sai số**, chỉ phương vuông góc với không gian cột khi mô hình đạt tối ưu.
- ∇_θ là **vector gradient** trong không gian tham số ($d+1$ chiều), chỉ hướng cần di chuyển để giảm loss.

Cả ba vector này liên hệ chặt chẽ với nhau thông qua phép nhân ma trận và chiếu hình học (projection).

5.7.3. Diễn giải hình học của nghiệm tối ưu

Tại điểm tối ưu θ^*, mô hình thỏa mãn điều kiện đạo hàm bằng 0:

$$X_b^\top (X_b\theta^* - \mathbf{y}) = 0$$

Điều này có nghĩa là $\mathbf{y} - X_b\theta^*$ (vector sai số) trực giao với toàn bộ không gian cột của X_b. Nói cách khác, $\hat{\mathbf{y}}$ là **hình chiếu trực giao** (*orthogonal projection*) của $\mathbf{y}$ lên không gian $\mathcal{C}(X_b)$.

Khi đó:

$$\hat{\mathbf{y}} = P_X\mathbf{y}, \quad \text{với } P_X = X_b(X_b^\top X_b)^{-1}X_b^\top$$

trong đó P_X là ma trận chiếu (projection matrix). Tính chất quan trọng:

$$P_X^\top = P_X, \quad P_X^2 = P_X$$

Tức là phép chiếu này không làm thay đổi các vector đã nằm trong không gian $\mathcal{C}(X_b)$.

5.7.4. Minh họa bằng hình học 3D (trường hợp 2 đặc trưng)

Để hình dung, xét bài toán có 2 đặc trưng x_1, x_2. Không gian dữ liệu lúc này là 3 chiều: (x_1, x_2, y). Mục tiêu của Linear Regression là tìm một **mặt phẳng** có phương trình:

$$\hat{y} = w_1x_1 + w_2x_2 + b$$

Các điểm dữ liệu thật (x_1, x_2, y) nằm rải rác trong không gian. Mô hình tìm mặt phẳng sao cho **tổng bình phương khoảng cách theo phương thẳng đứng (error vector)** giữa các điểm thật và mặt phẳng là nhỏ nhất.

Ở trạng thái hội tụ: - Các vector sai số $\mathbf{e}_i$ đều vuông góc với mặt phẳng dự đoán. - Tổng tất cả các vector sai số có hướng ngược với gradient trung bình.

5.7.5. Vai trò hình học của Vectorization

Vectorization giúp ta chuyển toàn bộ quá trình tính toán này vào không gian đại số tuyến tính:

$$\hat{\mathbf{y}} = X_b\theta, \quad \mathbf{e} = \hat{\mathbf{y}} - \mathbf{y}, \quad \nabla_\theta = \frac{2}{N}X_b^\top \mathbf{e}$$

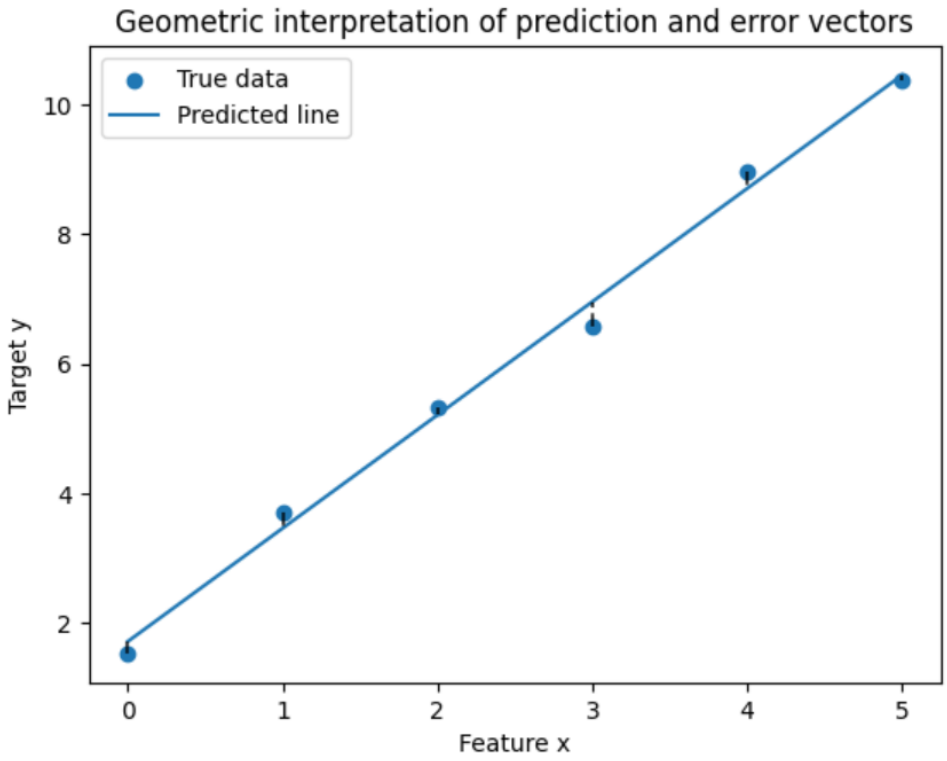
Thay vì tính toán từng điểm, ta đang thực hiện phép chiếu đồng thời của toàn bộ tập dữ liệu lên không gian cột của X_b. Quá trình Gradient Descent thực chất là một chuỗi phép chiếu dần dần, tiến gần đến hình chiếu trực giao chính xác (projection của $\mathbf{y}$).

5.7.6. Minh họa trực quan (Python Example)

```python
import numpy as np
import matplotlib.pyplot as plt

# Simple 2D example: y = 2x + 1
X = np.linspace(0, 5, 6).reshape(-1, 1)
y = 2 * X + 1 + np.random.randn(6, 1) * 0.5  # noisy data
X_b = np.c_[np.ones((X.shape[0], 1)), X]

# Closed-form solution
theta_best = np.linalg.inv(X_b.T @ X_b) @ X_b.T @ y
y_pred = X_b @ theta_best

# Plot data and regression line
plt.scatter(X, y, color="blue", label="True data")
```

```
plt.plot(X, y_pred, color="red", label="Predicted line")

# Draw error vectors
for i in range(len(X)):
    plt.plot([X[i], X[i]], [y[i], y_pred[i]], "k--", alpha=0.7)

plt.xlabel("Feature x")
plt.ylabel("Target y")
plt.legend()
plt.title("Geometric interpretation of prediction and error vectors")
plt.show()
```

Listing 3: Geometric Visualization of Prediction and Error Vectors



Hình 9: Minh họa bằng Python

5.7.7. Kết nối giữa đại số và hình học

Từ góc nhìn đại số tuyến tính:

$$\min_{\theta} \|y - X_b\theta\|^2 \quad \Rightarrow \quad \text{tìm } \hat{\mathbf{y}} \in \mathcal{C}(X_b) \text{ sao cho } \|y - \hat{\mathbf{y}}\| \text{ nhỏ nhất.}$$

Từ góc nhìn hình học:

- Mỗi bước cập nhật gradient là một bước di chuyển vuông góc trong không gian tham số θ.
- Mỗi lần cập nhật $\hat{\mathbf{y}} = X_b\theta$ là một phép chiếu gần hơn đến mặt phẳng chứa các vector trong $\mathcal{C}(X_b)$.
- Khi gradient bằng 0, ta đạt điểm mà vector sai số vuông góc hoàn toàn với không gian cột — tương đương với nghiệm tối ưu.

VI. Regulization

6.1. Tại sao cần Regularization?

Trong thực tế, không phải lúc nào mô hình hồi quy tuyến tính (Linear Regression) cũng hoạt động tốt. Nếu bạn có quá nhiều đặc trưng (d) trong khi số lượng mẫu huấn luyện (N) lại nhỏ, hoặc dữ liệu chứa nhiều nhiễu (noise), mô hình có thể bị **overfit** — tức là khớp hoàn hảo với dữ liệu huấn luyện nhưng lại dự đoán kém trên dữ liệu mới.

- Khi $d > N$: ma trận $X_b^\top X_b$ trở nên gần suy biến (singular), khiến nghiệm $\theta = (X_b^\top X_b)^{-1} X_b^\top y$ không ổn định hoặc thậm chí không tồn tại.
- Khi dữ liệu có nhiễu hoặc biến đầu vào tương quan mạnh (multicollinearity), mô hình dễ bị dao động mạnh khi có thay đổi nhỏ trong dữ liệu.

Regularization xuất hiện như một "cơ chế bảo vệ", giúp làm mềm mô hình bằng cách **thêm một thành phần phạt (penalty term)** vào hàm mất mát. Điều này buộc mô hình ưu tiên các nghiệm có trọng số nhỏ hơn — tức là các đường (hoặc mặt phẳng) hồi quy đơn giản hơn, mượt hơn và tổng quát tốt hơn.

Một cách hình tượng: > Nếu Linear Regression là một người học ghi nhớ tất cả dữ liệu huấn luyện, thì Regularization là người thầy giúp họ học "cái bản chất" thay vì học vẹt.

6.2 Ridge Regression (L2 Regularization)

Ridge Regression là phương pháp regularization cổ điển và dễ hiểu nhất. Ý tưởng rất đơn giản: nếu mô hình đang cố gắng phóng đại trọng số để giảm sai số, hãy thêm một "chi phí" cho việc có trọng số lớn.

Công thức hàm mất mát của Ridge:

$$L_{ridge}(\theta) = \frac{1}{N}\|\mathbf{y} - X_b\theta\|^2 + \lambda\|\theta\|^2$$

Trong đó:

- $\|\mathbf{y} - X_b\theta\|^2$ là phần lỗi dự đoán (loss gốc).
- $\lambda\|\theta\|^2$ là phần phạt (penalty term), trong đó λ quyết định mức độ "ép nhỏ" trọng số.
- $\lambda > 0$ càng lớn thì mô hình càng bị "kiềm chế" — trọng số càng nhỏ, mô hình càng trơn tru.

Phần đạo hàm vector hóa (gradient) được viết như sau:

$$\nabla_\theta L_{ridge} = \frac{2}{N} X_b^\top (X_b\theta - \mathbf{y}) + 2\lambda\theta$$

và quy tắc cập nhật tham số trong Gradient Descent:

$$\theta \leftarrow \theta - \eta \nabla_\theta L_{ridge}$$

Ý tưởng này có thể coi như việc "làm mờ" bề mặt hàm mất mát, giúp tránh hiện tượng dao động mạnh và cải thiện khả năng hội tụ.

6.3 Lasso Regression (L1 Regularization)

Nếu Ridge phạt bằng bình phương trọng số, thì Lasso lại chọn cách phạt "mạnh tay" hơn — bằng trị tuyệt đối:

$$L_{lasso}(\theta) = \frac{1}{N}\|\mathbf{y} - X_b\theta\|^2 + \lambda\|\theta\|_1$$

$$\|\theta\|_1 = \sum_{j=1}^{d} |\theta_j|$$

Tác động của Lasso là **đẩy một số trọng số về đúng 0**. Điều này có nghĩa là Lasso không chỉ giảm độ lớn của trọng số, mà còn thực hiện *feature selection* tự động — loại bỏ các đặc trưng ít quan trọng ra khỏi mô hình.

Tuy nhiên, vì hàm trị tuyệt đối không khả vi tại 0, gradient của Lasso không thể tính trực tiếp. Thay vào đó, người ta dùng các thuật toán đặc biệt như **Coordinate Descent** hoặc **Subgradient Method** để tìm nghiệm gần đúng.

Một cách dễ hiểu: > Ridge "làm nhỏ" mọi trọng số, còn Lasso "loại bỏ" những trọng số yếu — giống như chọn lọc tự nhiên trong học máy.

6.4 So sánh Ridge và Lasso

Để dễ hình dung, bảng sau tóm tắt sự khác biệt chính giữa hai phương pháp:

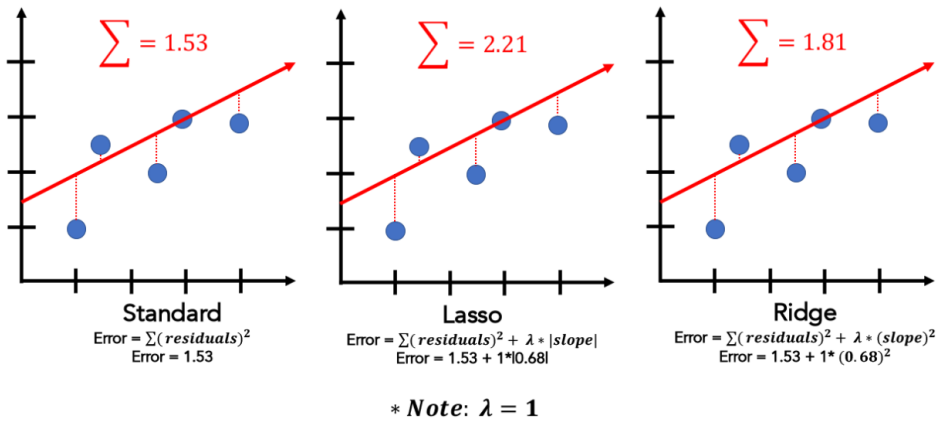
Đặc điểm	**Ridge (L2)**	**Lasso (L1)**
Công thức phạt	$\lambda \sum w_j^2$	$\lambda \sum \|w_j\|$
Hiệu ứng	Giảm độ lớn của trọng số	Ép một số trọng số về 0
Tác dụng chính	Giảm phương sai, ổn định mô hình	Lựa chọn đặc trưng tự động
Trường hợp lý tưởng	Dữ liệu có nhiều biến tương quan	Dữ liệu thưa hoặc có nhiều đặc trưng
Đạo hàm	Trơn, khả vi	Không trơn tại 0

Trong thực tế, Ridge thường được dùng khi mọi đặc trưng đều quan trọng ở một mức độ nào đó, còn Lasso phù hợp khi ta tin rằng chỉ một vài biến thật sự ảnh hưởng mạnh đến đầu ra.

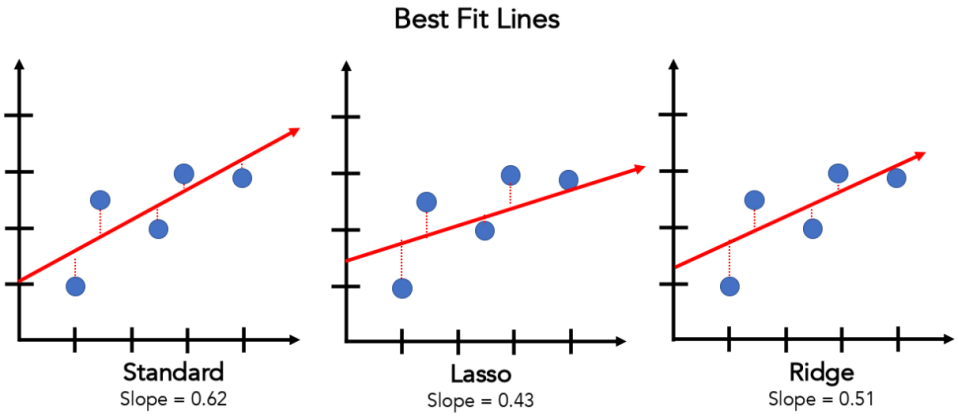
6.5 Minh họa ý tưởng bằng hình học

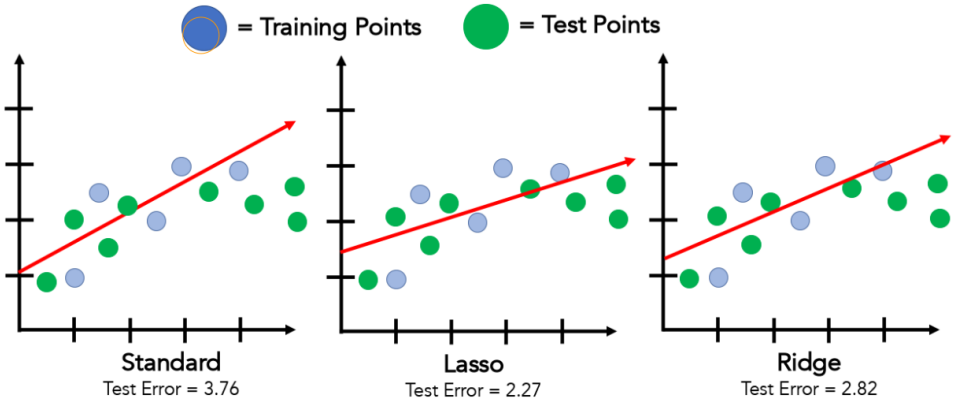
Hãy tưởng tượng hàm mất mát MSE có dạng như một **ellipsoid** trong không gian tham số θ. Khi thêm ràng buộc Ridge ($\|\theta\|_2^2 \leq c$), ta đang bao quanh vùng nghiệm bằng một **hình cầu**. Điểm tối ưu là nơi mà ellipsoid của MSE chạm vào hình cầu này.

Ngược lại, với Lasso ($\|\theta\|_1 \leq c$), vùng ràng buộc có hình thoi. Vì các cạnh của hình thoi có góc nhọn, nên điểm tiếp xúc thường nằm ngay trên trục tọa độ — tức là một vài trọng số bị "đè" về 0. Đây chính là lý do hình học khiến Lasso có khả năng loại bỏ đặc trưng.



$* \boldsymbol{Note}$: $\boldsymbol{\lambda = 1}$





6.6 Triển khai thực hành với NumPy

Ví dụ dưới đây minh họa cách triển khai Ridge Regression theo hướng vectorized — tận dụng tối đa khả năng xử lý ma trận của NumPy:

```
import numpy as np

# Generate training data
```

```
X = np.array([[1, 2], [2, 3], [3, 4], [4, 5]])
y = np.array([[5], [7], [9], [11]])

# Add bias term
X_b = np.c_[np.ones((X.shape[0], 1)), X]

# Initialize parameters
theta = np.random.randn(X_b.shape[1], 1)
eta = 0.01
n_iter = 1000
lam = 0.1   # regularization strength

# Vectorized Gradient Descent with Ridge Regularization
for epoch in range(n_iter):
    y_pred = X_b @ theta
    error = y_pred - y
    gradients = (2 / X_b.shape[0]) * (X_b.T @ error) + 2 * lam * theta
    gradients[0] -= 2 * lam * theta[0]  # exclude bias term from regularization
    theta -= eta * gradients

print("Ridge-regularized parameters:\n", theta)
```

Listing 4: Vectorized Ridge Regression with Gradient Descent (NumPy Implementation)

6.7 Analytical (Closed-form) Ridge Solution

Không chỉ có cách tính qua Gradient Descent, Ridge còn có nghiệm đóng — còn gọi là **Regularized Normal Equation**:

$$\theta^*_{ridge} = (X_b^\top X_b + \lambda I)^{-1} X_b^\top \mathbf{y}$$

Trong đó I là ma trận đơn vị kích thước $(d+1) \times (d+1)$, và phần tử đầu tiên (bias) không được regularize.

```
I = np.eye(X_b.shape[1])
I[0, 0] = 0  # do not regularize bias
theta_ridge_closed = np.linalg.inv(X_b.T @ X_b + lam * I) @ X_b.T @ y
print("Closed-form Ridge theta:\n", theta_ridge_closed)
```

Listing 5: Closed-form Ridge Regression Solution

6.8 Biểu diễn tổng quát và mở rộng sang Elastic Net

Công thức tổng quát của regularization là:

$$\min_\theta \frac{1}{N}\|y - X_b\theta\|^2 + \lambda R(\theta),$$

với:

$$R(\theta) = \begin{cases} \|\theta\|_2^2 & \text{(Ridge)} \\ \|\theta\|_1 & \text{(Lasso)} \end{cases}$$

Một biến thể phổ biến hơn là **Elastic Net**, kết hợp cả hai hình thức phạt:

$$L_{elastic} = \frac{1}{N}\|y - X_b\theta\|^2 + \lambda_1\|\theta\|_1 + \lambda_2\|\theta\|_2^2$$

Elastic Net giữ lại khả năng chọn lọc đặc trưng của Lasso, đồng thời duy trì độ ổn định của Ridge — một lựa chọn cân bằng tuyệt vời cho dữ liệu phức tạp.

6.9 Tổng kết Regularization

- Regularization giúp kiểm soát độ phức tạp của mô hình, ngăn overfitting và cải thiện khả năng tổng quát.
- Ridge làm mượt mô hình bằng cách giới hạn độ lớn của trọng số.
- Lasso đơn giản hóa mô hình bằng cách loại bỏ những trọng số không cần thiết.
- Elastic Net kết hợp sức mạnh của cả hai phương pháp.

Hiểu và áp dụng Regularization là một bước quan trọng để xây dựng mô hình học máy vững vàng, đặc biệt khi bạn bắt đầu xử lý các tập dữ liệu thực tế — nơi nhiễu, tương quan và thiếu dữ liệu là chuyện thường ngày.