CHẨN ĐOÁN TÌNH TRẠNG KHỐI U CỦA BỆNH NHÂN MẮC BỆNH UNG THƯ VÚ

Breast Cancer classification

N.Tiến Đạt N.Thành Trung N.Thị Ánh

Báo cáo giữa kì

HUS
VNU UNIVERSITY OF SCIENCE

MATHEMATICS
INFORMATICS
MECHANICS

Nội dung

Giới thiệu bài toán

Ung thư vú là một trong những bệnh ung thư phổ biến và gây tử vong cao ở phụ nữ. Phát hiện sớm có thể nâng tỷ lệ sống sót lên trên 90%. Trong bối cảnh dữ liệu y tế ngày càng phong phú, học máy trở thành công cụ hiệu quả hỗ trợ chẩn đoán chính xác và kịp thời.
Đề tài sử dụng bộ dữ liệu *Breast Cancer Wisconsin (Diagnostic)* để phân loại khối u lành tính và ác tính, thông qua các bước tiền xử lý, giảm chiều (PCA, LDA) và xây dựng nhiều mô hình học máy. Việc lựa chọn đề tài không chỉ mang tính thực tiễn, học thuật mà còn thể hiện mong muốn đóng góp cho lĩnh vực công nghệ y tế.

Nội dung

Dữ liệu

Bộ dữ liệu *Breast Cancer Wisconsin (Diagnostic)* gồm 569 mẫu với 33 cột, dùng để phân loại khối u lành tính (Benign) và ác tính (Malignant) dựa trên các đặc trưng hình thái tế bào.

- **id**: mã định danh mẫu.
- **diagnosis**: nhãn phân loại, gồm `M` (Malignant) và `B` (Benign).
- **30 đặc trưng số thực**: mỗi đặc trưng được tính theo 3 nhóm: Mean (trung bình), SE (Standard Error), và Worst (giá trị lớn nhất). Ví dụ: `radius_mean`, `radius_se`, `radius_worst`.
- **Unnamed: 32**: cột rỗng, không mang thông tin và sẽ bị loại bỏ.

Xử lý dữ liệu

- **Tổng quan về DataFrame (df):**
  - Kiểm tra số hàng và cột: `df.shape`
  - Kiểm tra tên cột và kiểu dữ liệu: `df.dtypes`
  - Kiểm tra số lượng giá trị không thiếu: `df.count()`
  - Kiểm tra dung lượng bộ nhớ: `df.info()`
- **Kiểm tra giá trị thiếu:**
  - Số lượng giá trị thiếu (NaN): `df.isna().sum()`
- **Thống kê biến mục tiêu:**
  - Đếm số mẫu có nhãn: `df['diagnosis'].value_counts()`
- **Mã hóa nhãn bằng Label Encoding:**
  - Chuyển nhãn từ dạng chữ cái sang số: ”B”→ 0 (lành tính), ”M”→ 1 (ác tính)
- **Trực quan hóa dữ liệu bằng Seaborn:**
  - Biểu diễn số lượng mẫu mỗi loại: `sns.catplot(x='diagnosis', data=df, kind='count')`
- **Loại bỏ cột không cần thiết:**
  - Xóa các cột không cần thiết: `df.drop(['id', 'Unnamed: 32'], axis=1, inplace=True)`

Xử lý dữ liệu

- **Tách biến đầu vào và đầu ra:**
  - Tách X là dữ liệu đầu vào, y là biến mục tiêu: `X = df.drop(['diagnosis'], axis=1), y = df['diagnosis']`
- **Chia tập dữ liệu:**
  - Chia tập dữ liệu với 30% dữ liệu làm tập kiểm tra: `train_test_split(X, y, test_size=0.3, random_state=42, stratify=y)`
- **Chuẩn hóa dữ liệu:**
  - Chuẩn hóa dữ liệu: `StandardScaler` chuẩn hóa theo phân phối chuẩn với trung bình 0 và độ lệch chuẩn 1.
  - Công thức chuẩn hóa:

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

Trong đó:

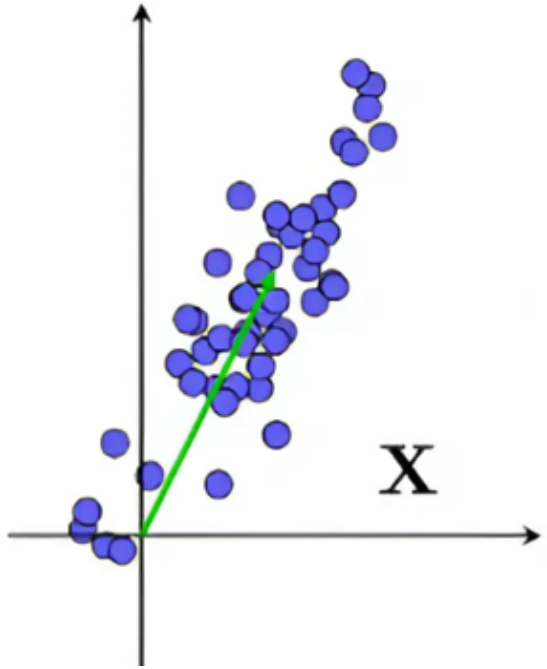
- x là giá trị dữ liệu ban đầu
- μ là trung bình đặc trưng
- σ là độ lệch chuẩn

Phân tích thành phần chính PCA

- **Ý tưởng:** Giảm số chiều dữ liệu từ D chiều về K chiều. Chỉ giữ lại K chiều quan trọng nhất và loại bỏ $D - K$ chiều ít ảnh hưởng nhất.
- **Mục tiêu** của phương pháp này là tối ưu hóa lượng thông tin quan trọng nhất, đồng thời giảm thiểu sự phức tạp của mô hình.
- Việc giữ lại các đặc trưng quan trọng giúp giảm chi phí tính toán mà vẫn duy trì hiệu suất mô hình cao.
- **Thách thức:** Làm thế nào để xác định được các đặc trưng quan trọng nhất trong một không gian dữ liệu lớn.

Các bước thực hiện PCA

Bước 1: Tính vecto kỳ vọng trên toàn bộ dữ liệu



Hình: Bước 1

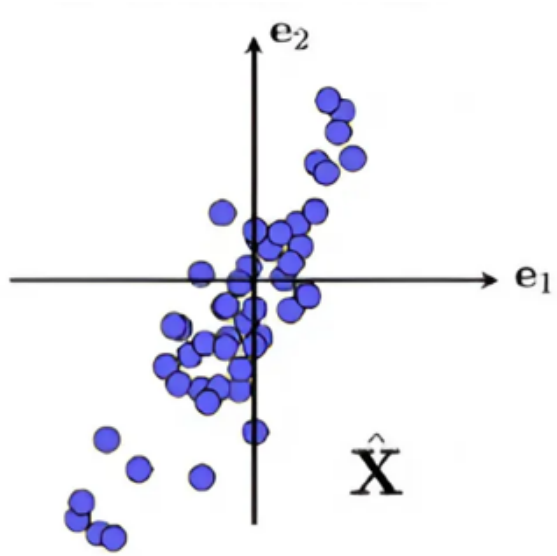
Mục tiêu: giúp loại bỏ phần nhiễu không cần thiết trong quá trình phân tích.

Các bước thực hiện PCA

Bước 2: Tính dữ liệu chuẩn hóa $\hat{\mathbf{x}}_n$

Chuẩn hóa dữ liệu giúp đưa tất cả các điểm dữ liệu về trung tâm (mean = 0), bằng cách trừ đi vector trung bình:

$$\hat{\mathbf{x}}_n = \mathbf{x}_n - \bar{\mathbf{x}}; \quad n = 1, 2, \ldots, N.$$



Việc này giúp loại bỏ sự lệch tâm của dữ liệu.

Các bước thực hiện PCA

Bước 3: Tính ma trận hiệp phương sai

- Ma trận hiệp phương sai $\mathbf{S}$ được tính từ dữ liệu chuẩn hóa $\hat{\mathbf{X}}$:

$$\mathbf{S} = \frac{1}{N}\hat{\mathbf{X}}\hat{\mathbf{X}}^T$$

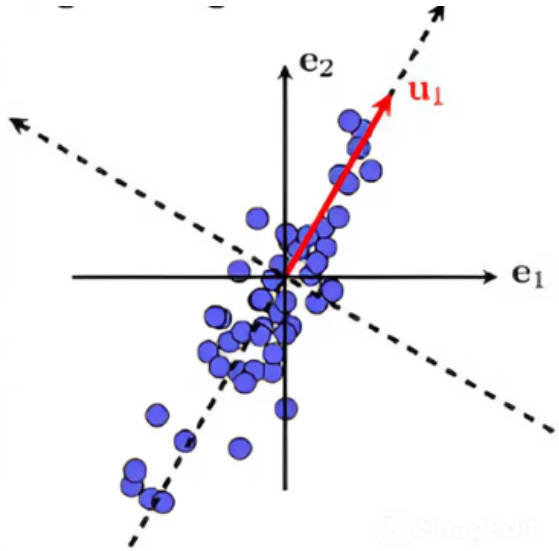
- Trong đó:
  - $\hat{\mathbf{X}}$ là ma trận dữ liệu đã được chuẩn hóa.
  - $\mathbf{S}$ là ma trận hiệp phương sai, cơ sở để tìm các thành phần chính trong PCA thông qua phân tích giá trị riêng.
- Các phần tử trong ma trận hiệp phương sai biểu thị sự tương quan cộng tính giữa các cặp dữ liệu.

Bước 4: Tính các trị riêng λ_i và vector riêng $\mathbf{u}_i$ của ma trận hiệp phương sai.
Các giá trị riêng λ_i do lượng mức độ biến thiên của dữ liệu đọc theo vectơ riêng tương ứng, và các vectơ riêng là các hướng mới mà đọc theo đó, dữ liệu có sự biến thiên lớn nhất.

Các bước thực hiện PCA

Bước 5: Chọn các thành phần chính

Lựa chọn k vectơ riêng đầu tiên tương ứng với k trị riêng lớn nhất giúp giữ lại phần lớn thông tin quan trọng của dữ liệu.



Các vectơ riêng này tạo thành cơ sở của không gian mới, nơi dữ liệu sẽ được chiếu xuống.

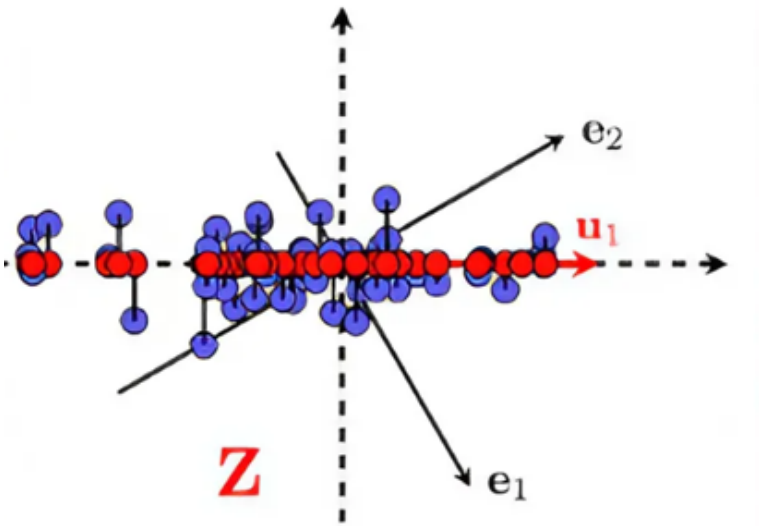
Các cột của $\{\mathbf{U}_j\}_{j=1}^{k}$ là hệ cơ sở trực chuẩn.

Các bước thực hiện PCA

Bước 6: Chiếu xuống không gian mới

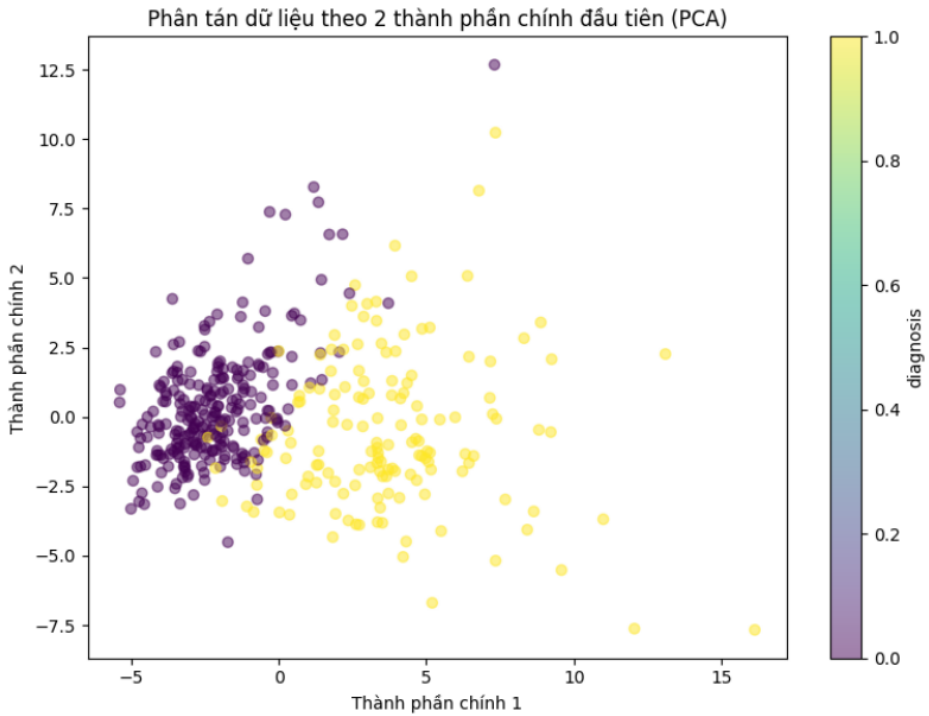
Tọa độ của các điểm dữ liệu trong không gian mới được tính bằng cách chiếu dữ liệu lên các trục chính:

$$\mathbf{Z} = \mathbf{U}_k^T \hat{\mathbf{X}}.$$



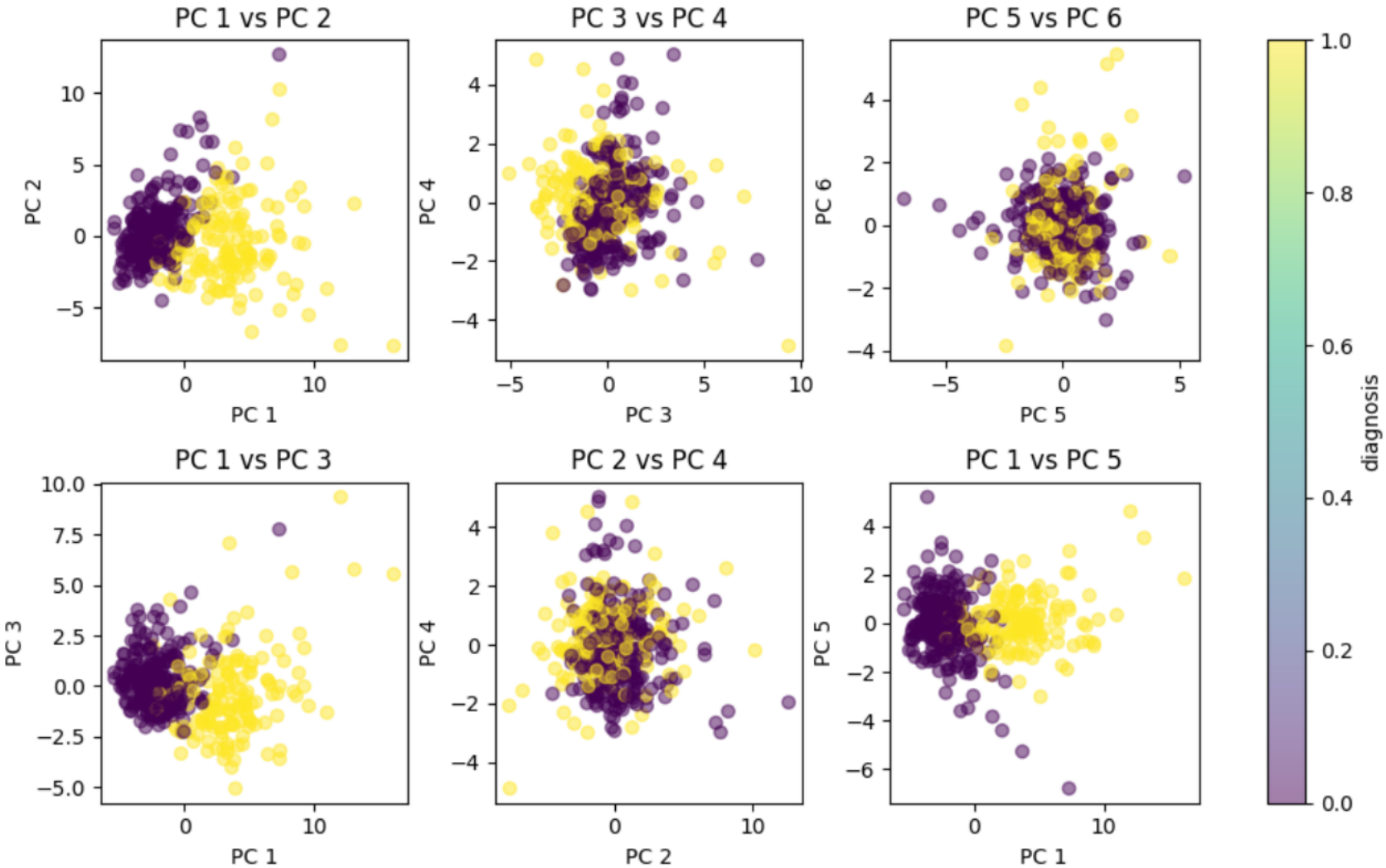
Dữ liệu đã được chuyển về không gian mới, giúp giảm chiều và tối ưu hóa quá trình tính toán.

Kết quả thu được



Ảnh 1：Phân tán dữ liệu theo 2 thành phần chính đầu tiên (PCA)

Kết quả thu được



Ảnh 1 : Biểu đồ phân tán cho 6 thành phần chính đầu tiên

Phân tích phân biệt tuyến tính LDA

Phân tích phân biệt tuyến tính:

- LDA là phương pháp giảm chiều dữ liệu, tối đa hóa khả năng phân biệt giữa các nhóm dữ liệu đã gán nhãn.
- Khác với PCA, LDA sử dụng thông tin nhãn lớp để xác định hướng chiếu tối ưu.
- Mục tiêu: tìm trục chiếu sao cho dữ liệu có sự phân tách rõ ràng giữa các lớp.
- LDA không chỉ giảm chiều mà còn nâng cao khả năng phân loại.

Giả thiết

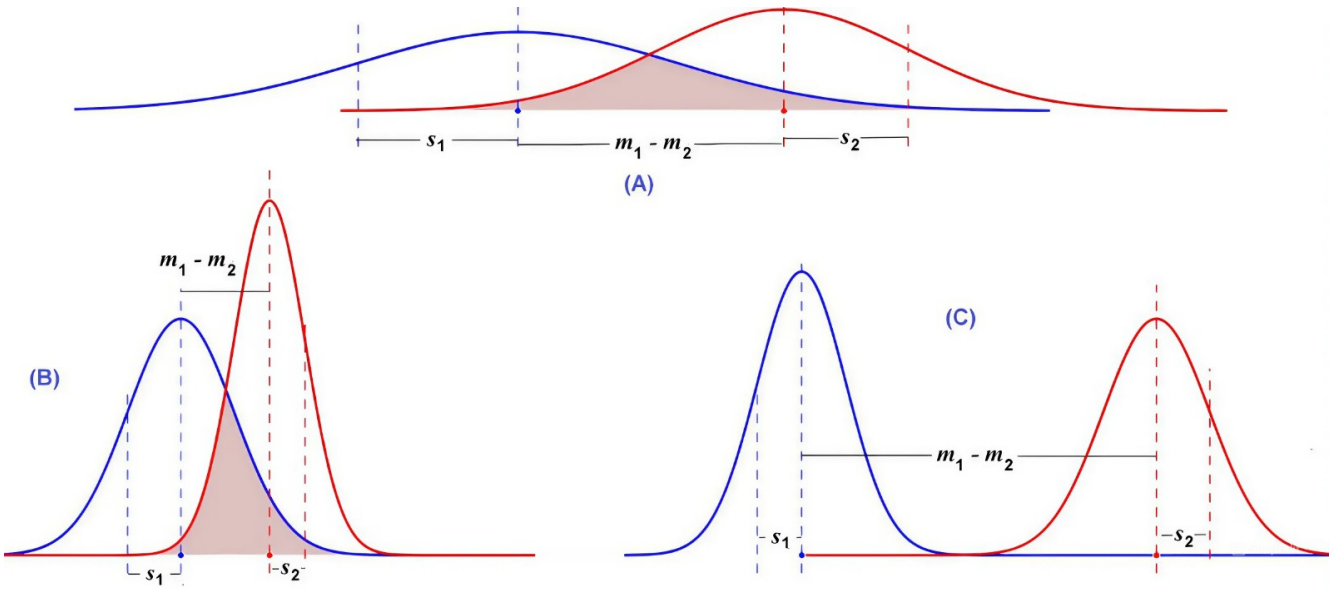
Ta xét bài toán phân loại nhị phân (binary classification) với hai nhãn: $\{01, 02\}$. Đặt:

- X_1: Tập dữ liệu có nhãn 01
- X_2: Tập dữ liệu có nhãn 02

Giả sử khi chiều dữ liệu xuống một chiều bất kỳ, ta có:

- Kỳ vọng (mean): m_1, m_2
- Phương sai (variance): s_1^2, s_2^2

Ví dụ



- (A): $|m_1 - m_2|$ lớn, s_1, s_2 cũng lớn $\rightarrow$ dữ liệu phân tán mạnh $\rightarrow$ vùng chồng lấn lớn $\rightarrow$ **độ phân biệt thấp**.
- (B): s_1, s_2 nhỏ nhưng $|m_1 - m_2|$ cũng nhỏ $\rightarrow$ dữ liệu gần nhau $\rightarrow$ vùng chồng lấn lớn $\rightarrow$ **độ phân biệt thấp**.
- (C): s_1, s_2 nhỏ và $|m_1 - m_2|$ lớn $\rightarrow$ vùng chồng lấn ít $\rightarrow$ **độ phân biệt cao**.

Bước 1: Tính S_W và S_B

- **Phương sai trong lớp (within-class variance)**:

$$S_W = \sum_{n \in C_1} (x_n - m_1)(x_n - m_1)^T + \sum_{n \in C_2} (x_n - m_2)(x_n - m_2)^T$$

S_W thể hiện sự phân tán trong mỗi lớp tức là sự biến động của các điểm trong lớp so với kỳ vọng của lớp đó

- **Phương sai giữa các lớp (between-class variance)**:

$$S_B = (m_1 - m_2)(m_1 - m_2)^T$$

S_B thể hiện sự phân tán giữa các lớp, tức là khoảng cách giữa các kỳ vọng m_1 và m_2 của hai lớp.

Bước 2: Tính $A = S_W^{-1} S_B$

Sau khi tính toán được S_W và S_B, ta tính ma trận A như sau:

$$A = S_W^{-1} S_B$$

Ma trận A giúp xác định các ”hướng phân biệt”dữ liệu, nhằm tối ưu hóa việc phân chia các lớp trong không gian dữ liệu..

Bước 3: Tính $L = \max\{\lambda_i\}$

Tính giá trị riêng của ma trận A và chọn giá trị riêng lớn nhất:

$$L = \max\{\lambda_1, \lambda_2, \ldots, \lambda_d\}$$

Trong đó λ_i là các giá trị riêng của ma trận A.

Chú ý: Với mỗi giá trị riêng λ, sẽ có vô số vector riêng tương ứng.

Mục tiêu của bước này: Tìm ra các hướng có giá trị riêng lớn nhất, nhằm giảm chiều dữ liệu và tăng cường khả năng phân loại.

Bước 4: Chọn w sao cho

Sau khi có giá trị riêng lớn nhất, ta chọn vector phân biệt w sao cho:

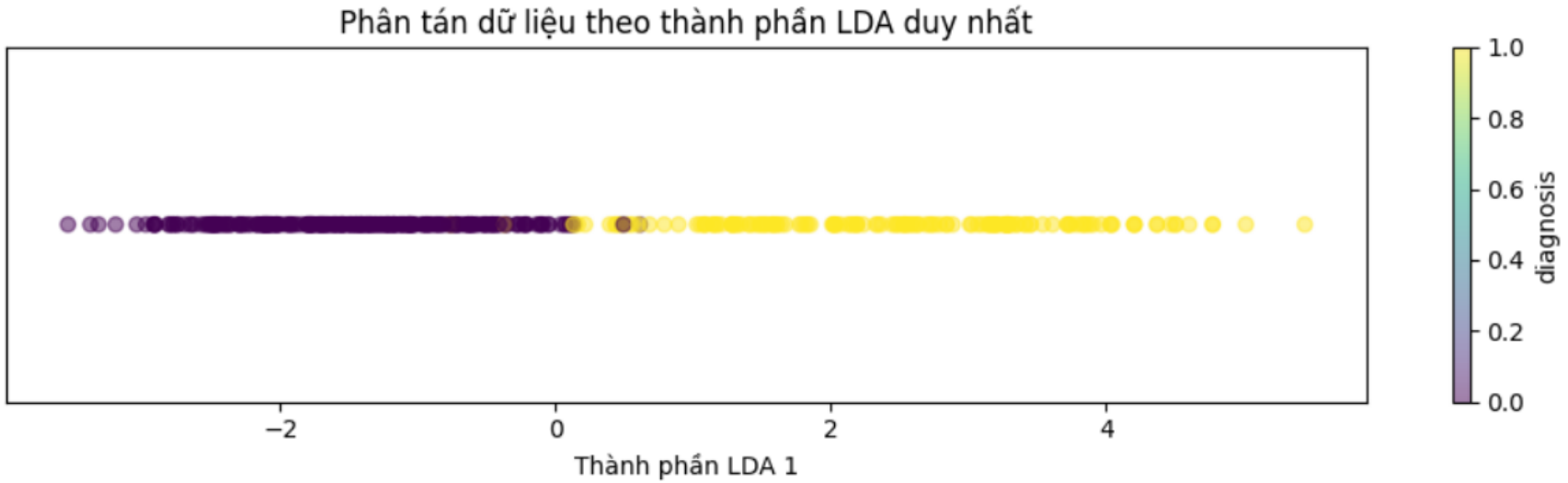
$$(m_1 - m_2)^T w = L$$

Vector w được tính như sau:

$$w = S_W^{-1}(m_1 - m_2) \cdot \beta \quad (\text{với } \beta > 0)$$

Trong đó β là hệ số điều chỉnh, giúp tối ưu hóa sự phân biệt giữa các lớp.

Trực quan hóa LDA



Dữ liệu được phân tán dọc theo trục thành phần LDA 1, với một số chồng lấn nhẹ giữa các nhóm.
Điều này có thể chỉ ra rằng LDA đã thực hiện phân tách tốt, nhưng không hoàn toàn rõ ràng giữa hai nhóm.

Nội dung

Mô hình KNN

Nếu bạn biết 5 người bạn thân của ai đó, liệu bạn có thể đoán được người đó là người như thế nào?

Trong học máy, có một thuật toán tư duy tương tự – đó chính là **KNN**.
K-Nearest Neighbors (KNN) là một thuật toán đơn giản nhưng hiệu quả. Nó phân loại một điểm dữ liệu mới dựa trên *"K người hàng xóm gần nhất"* trong không gian đặc trưng.

Hình: Minh họa mô hình KNN

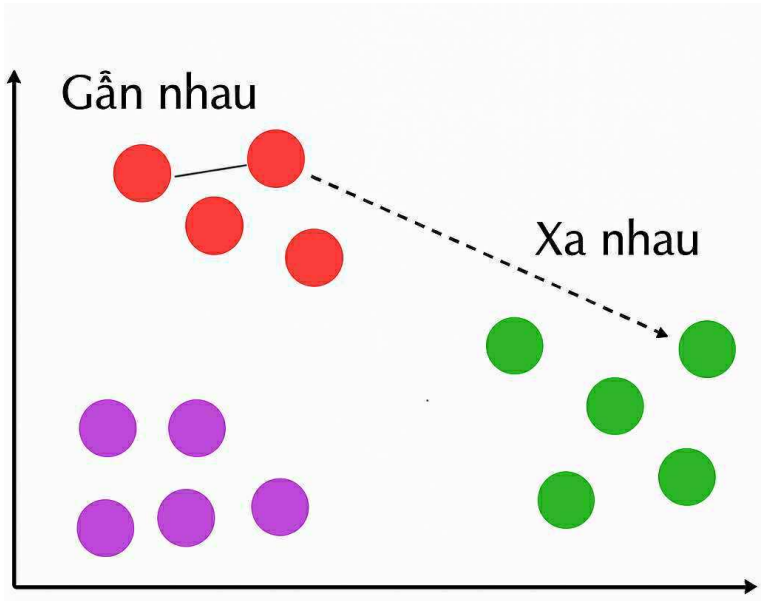
Các bước thực hiện KNN

1. Với mỗi giá trị x ở trong tập dự đoán ,ta tính khoảng cách của x đến tất cả các điểm trong tập huấn luyện.
2. Tìm k phần tử trong tập huấn luyện gần x nhất
3. Dự đoán y_pred theo $\frac{\sum y_i}{K}$ với:

$$y_{\text{pred}} = \begin{cases} 1, & \text{nếu } \frac{\sum y_i}{K} \geq 0.5 \\ 0, & \text{nếu } \frac{\sum y_i}{K} < 0.5 \end{cases}$$

KNN và ví dụ

Giả định của KNN: Những điểm dự liệu giống nhau sẽ có khoảng cách gần nhau hơn và những điểm dữ liệu khác nhau sẽ có khoảng cách xa nhau hơn.

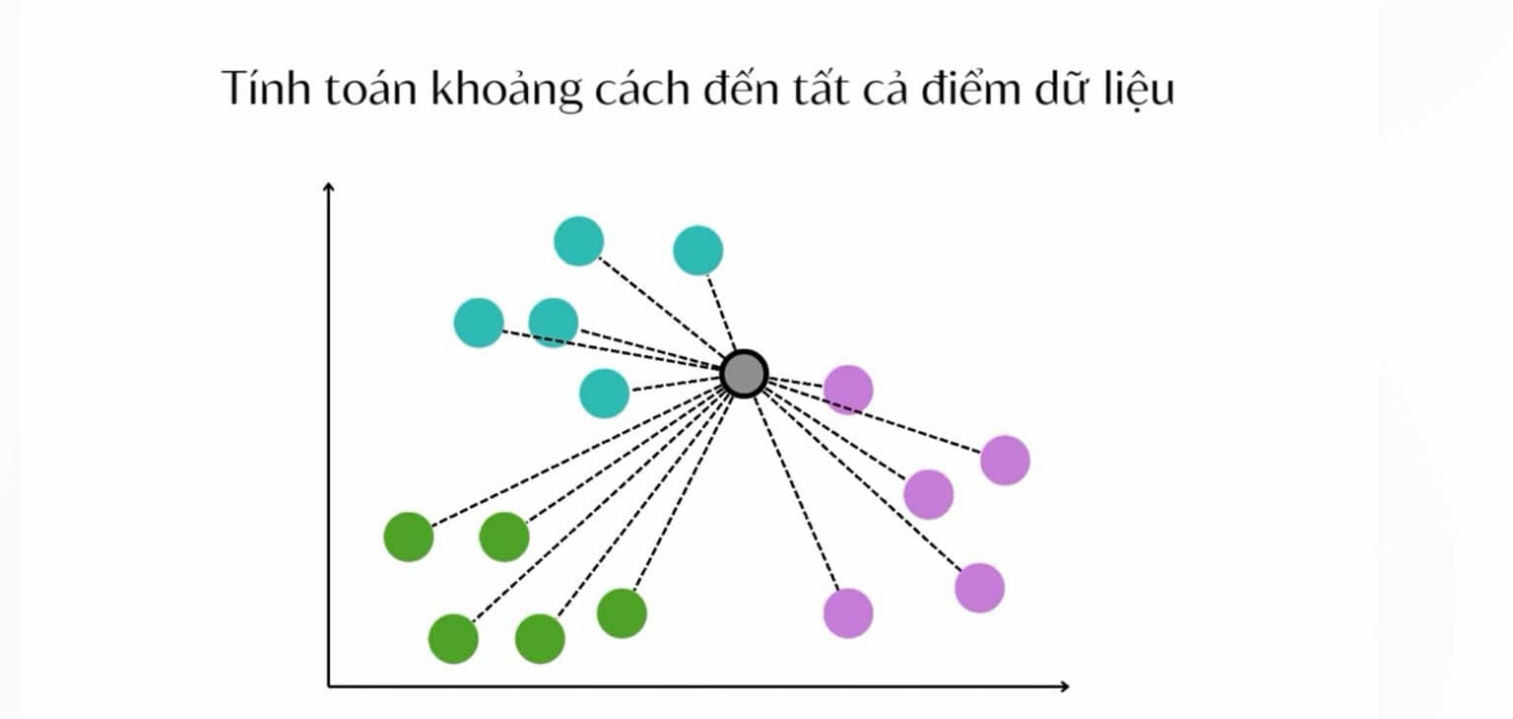


Lấy 1 ví dụ,chúng ta có 1 tập dữ liệu gồm 3 loại quả tương ứng: màu xanh lá đại diện cho quả táo , màu xanh dương đại diện quả cam và màu tím đại diện quả thanh long.

Vậy KNN sẽ hoạt động ra sao ?

KNN và ví dụ

Đầu tiên chúng ta phải tính toán khoảng cách từ điểm dữ liệu mới này đến tất cả các điểm dữ liệu mà chúng ta đã lắm trong tay.



KNN và ví dụ

Khoảng cách này được tính xác định như sau:

- **Trường hợp $d = 1$ (dữ liệu một chiều)**:
  Khoảng cách giữa điểm cần dự đoán x và từng điểm huấn luyện x_i được tính theo công thức:
  $$d_i = |x - x_i|$$
- **Trường hợp $d > 1$ (dữ liệu nhiều chiều)**:
  Khi dữ liệu là vector nhiều chiều, khoảng cách giữa x và x_i được tính bằng công thức chuẩn Euclid (L2 norm):
  $$d_i = \|x - x_i\|_2 = \sqrt{\sum_{j=1}^{d} (x_j - x_{i,j})^2}$$

  Trong đó:
  $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ là vector đặc trưng của điểm cần phân loại ;
  $\mathbf{x}_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,d})$ là vector đặc trưng của điểm huấn luyện thứ i;
  x_j: đặc trưng thứ j của $\mathbf{x}$;
  $x_{i,j}$: đặc trưng thứ j của điểm huấn luyện thứ i.

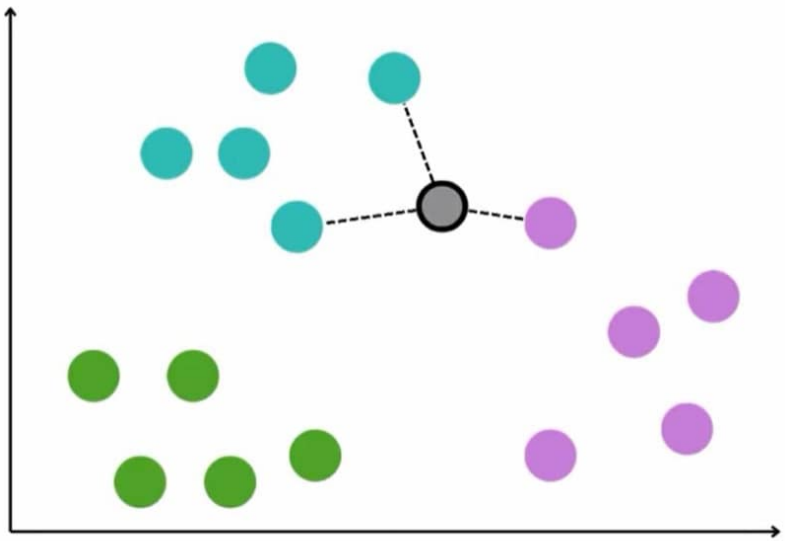
KNN và ví dụ

Với công thức khoảng cách trong tay thì chúng ta sẽ có thể xác định được K điểm dữ liệu gần với điểm dữ liệu này nhất.

K có thể bao gồm nhiều giá trị khác nhau.Tuy nhiên cách được sử dụng nhiều nhất là

$$K = \sqrt{n}$$

Trong trường hợp ví dụ này , lấy K = 3 (Lưu ý K là số lẻ để tránh trường hợp xảy ra tỷ lệ hòa (trung bình bằng 0.5))



Áp dụng KNN vào bài toán

Từ những phân tích trên, ta sẽ áp dụng mô hình KNN để phân loại u lành tính (Benign - 0) hay ác tính(Malignant - 1) trong tập dữ liệu.

1. Chia dữ liệu thành tập huấn luyện (train) và kiểm tra (test) theo tỉ lệ 7:3.
2. Tính khoảng cách giữa điểm cần dự đoán và các điểm trong tập huấn luyện. Với dữ liệu gốc hoặc dữ liệu sau khi giảm chiều bằng PCA (trường hợp nhiều chiều, $d > 1$), ta sử dụng công thức khoảng cách Euclid:

$$d_i = \|x - x_i\|_2 = \sqrt{\sum_{j=1}^{d} (x_j - x_{i,j})^2}$$

Trong code, `KNeighborsClassifier` mặc định sử dụng khoảng cách Euclid. Tuy nhiên, ta cũng có thể tự định nghĩa hàm tính khoảng cách như sau:

```
def distance(array, value):
    array = np.array(array)
    return np.linalg.norm(array - value, ord = 2, axis=<dim_axis>)
```

Với dữ liệu LDA số chiều sau khi giảm còn một chiều nên ta có thể áp dụng hàm tính sau:

```
def distance(array, value):
    array = np.array(array)
    return np.absolute(array - value)
```

Áp dụng KNN vào bài toán

3. Chọn K điểm dữ liệu gần nhất
   Trong bài toán này, do dữ liệu có 569 mẫu nên ta chọn $K = 23$ điểm dữ liệu gần nhất đến các mẫu dữ liệu trong tập huấn luyện.

   Bài toán sử dụng phương pháp `weights='uniform'` — mỗi hàng xóm trong số K hàng xóm gần nhất có ảnh hưởng bằng nhau đến quyết định nhãn.

   Công thức dự đoán:

$$y_{\text{pred}} = \begin{cases} 1, & \text{nếu } \dfrac{\sum y_i}{K} \geq 0.5 \\ 0, & \text{nếu } \dfrac{\sum y_i}{K} < 0.5 \end{cases}$$

Áp dụng KNN vào bài toán

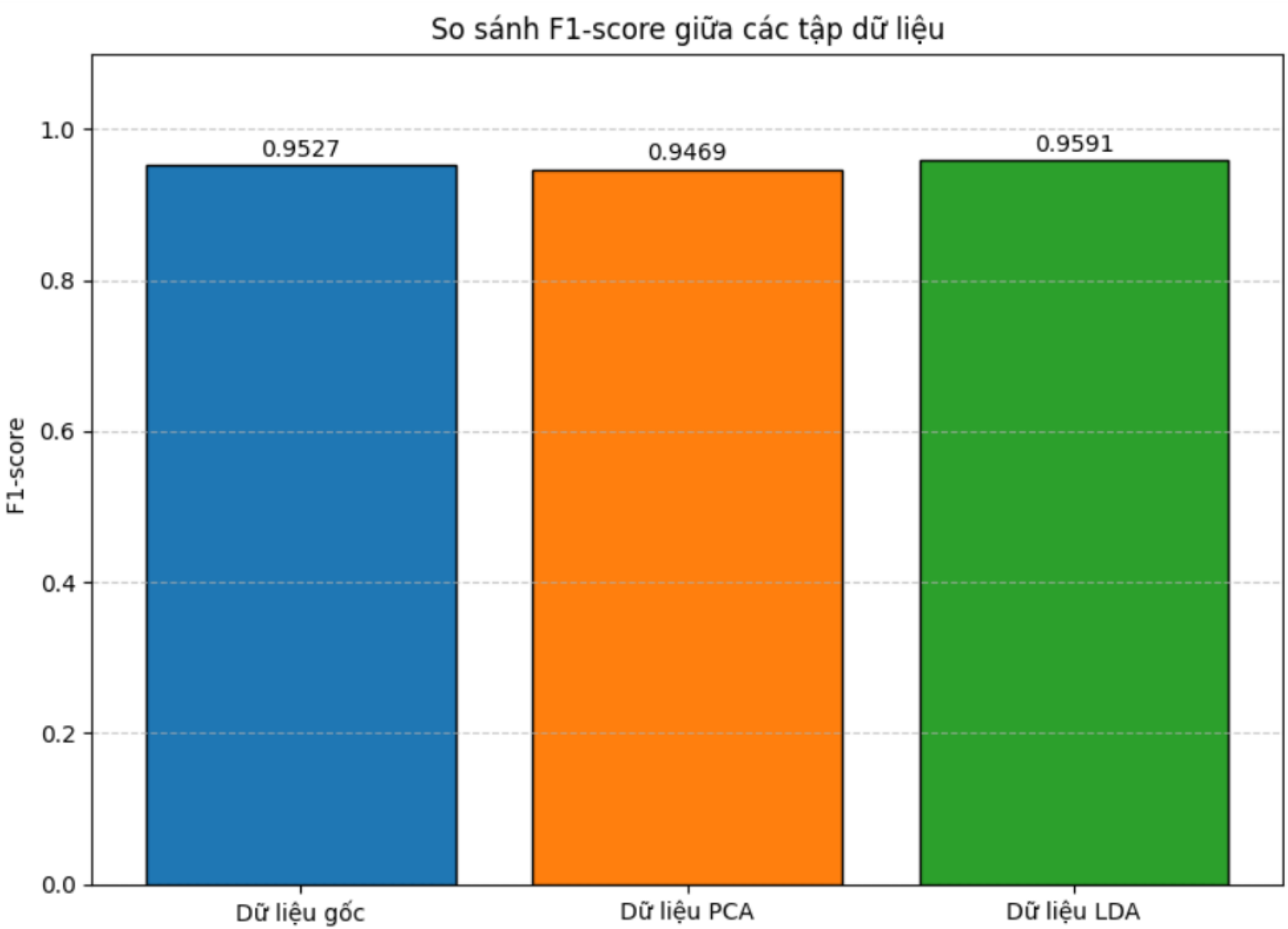
3. **Ví dụ minh họa:**
   Giả sử $K = 3$, một mẫu kiểm tra có 3 hàng xóm gần nhất:
   - Hàng xóm 1: Nhãn $y_1 = 0$ (Benign), khoảng cách $d_1 = 1$.
   - Hàng xóm 2: Nhãn $y_2 = 1$ (Malignant), khoảng cách $d_2 = 2$.
   - Hàng xóm 3: Nhãn $y_3 = 1$ (Malignant), khoảng cách $d_3 = 4$.

   Tính toán:

$$\sum y_i = y_1 + y_2 + y_3 = 0 + 1 + 1 = 2$$

$$\frac{\sum y_i}{K} = \frac{2}{3} \approx 0.667 \geq 0.5 \Rightarrow y_{\text{pred}} = 1 \quad (\text{Malignant})$$

Kết quả


So sánh F1-score giữa các tập dữ liệu
F1-score
1.0
0.8
0.6
0.4
0.2
0.0
0.9527
0.9469
0.9591
Dữ liệu gốc
Dữ liệu PCA
Dữ liệu LDA

Logistic Regression

Hồi quy logistic mô hình hóa **xác suất có điều kiện** (conditional probability) của một điểm dữ liệu thuộc về lớp dương (thường là lớp 1). Cụ thể, mô hình ước lượng xác suất $P(y = 1|\mathbf{x}; \boldsymbol{\theta})$, trong đó:

- $\mathbf{x}$ là vector các đặc trưng đầu vào ($\mathbf{x} \in \mathbb{R}^{n+1}$, với $x_0 = 1$ là hệ số chặn).
- $\boldsymbol{\theta}$ là vector các tham số (trọng số) của mô hình ($\boldsymbol{\theta} \in \mathbb{R}^{n+1}$).

Giá trị đầu ra của mô hình, ký hiệu là $h_{\boldsymbol{\theta}}(\mathbf{x})$, biểu diễn xác suất này:

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = P(y = 1|\mathbf{x}; \boldsymbol{\theta})$$

Hàm Sigmoid (Logistic)

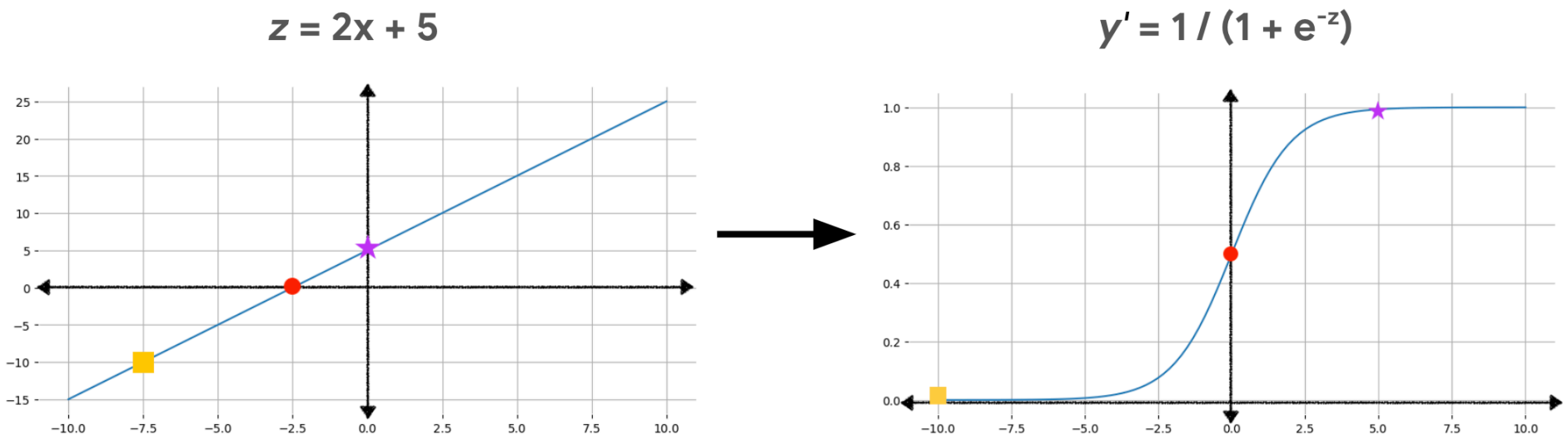
Để đảm bảo rằng giá trị đầu ra $h_{\boldsymbol{\theta}}(\mathbf{x})$ nằm trong khoảng $(0, 1)$, hồi quy logistic sử dụng **hàm sigmoid** (còn gọi là hàm logistic) để biến đổi tổ hợp tuyến tính của các đặc trưng và tham số ($\boldsymbol{\theta}^T\mathbf{x}$). Hàm sigmoid được định nghĩa là:

$$g(z) = \frac{1}{1 + e^{-z}}$$

Hàm này nhận đầu vào là một số thực bất kỳ ($z \in \mathbb{R}$) và trả về một giá trị trong khoảng $(0, 1)$. Do đó, giả thuyết (hypothesis) của mô hình hồi quy logistic được viết là:

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = g(\boldsymbol{\theta}^T\mathbf{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^T\mathbf{x}}}$$

Hàm Sigmoid (Logistic)



Hình: Trái: đồ thị của hàm tuyến tính z = 2x + 5, với ba điểm được làm nổi bật. Phải: Đường cong Sigmoid với cùng ba điểm được làm nổi bật sau khi được biến đổi bằng hàm sigmoid.

Quyết định lớp dựa trên xác suất

Sau khi tính được xác suất $h_{\boldsymbol{\theta}}(\mathbf{x})$, ta chuyển đổi nó thành dự đoán lớp cụ thể bằng quy tắc ngưỡng (thường là 0.5):

$$\hat{y} = \begin{cases} 1 & \text{nếu } h_{\boldsymbol{\theta}}(\mathbf{x}) \geq 0.5 \\ 0 & \text{nếu } h_{\boldsymbol{\theta}}(\mathbf{x}) < 0.5 \end{cases}$$

Điều này tương đương với việc kiểm tra $\boldsymbol{\theta}^T \mathbf{x} \geq 0$, vì hàm sigmoid $g(z) \geq 0.5$ khi và chỉ khi $z \geq 0$.

Hàm mất mát (Loss Function)

Để huấn luyện mô hình, ta cần một hàm mất mát đánh giá sai lệch giữa dự đoán và giá trị thực. Hồi quy logistic sử dụng **Log Loss** (hay **Binary Cross-Entropy**):

$$J(\boldsymbol{\theta}) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})) + (1-y^{(i)})\log(1-h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}))\right]$$

trong đó:

- m: số lượng mẫu huấn luyện.
- $y^{(i)}$: nhãn thực tế của mẫu thứ i.
- $h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})$: xác suất dự đoán cho mẫu thứ i.

Hàm mất mát này khuyến khích mô hình dự đoán xác suất gần 1 cho các mẫu có $y = 1$ và gần 0 cho các mẫu có $y = 0$.

Tối ưu hàm mất mát

Mục tiêu là tìm $\boldsymbol{\theta}$ sao cho $J(\boldsymbol{\theta})$ nhỏ nhất. Phương pháp phổ biến là **Gradient Descent**:

$$\boldsymbol{\theta} := \boldsymbol{\theta} - \alpha \nabla J(\boldsymbol{\theta})$$

với α là tốc độ học (learning rate). Gradient của hàm mất mát được tính như sau:

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^{m} \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right) x_j^{(i)}$$

Nội dung

Tài liệu tham khảo

1. TS. Cao Văn Chung, *Giáo trình môn học Học máy*. Trường đại học Khoa học tự nhiên. Đại học Quốc gia Hà Nội.