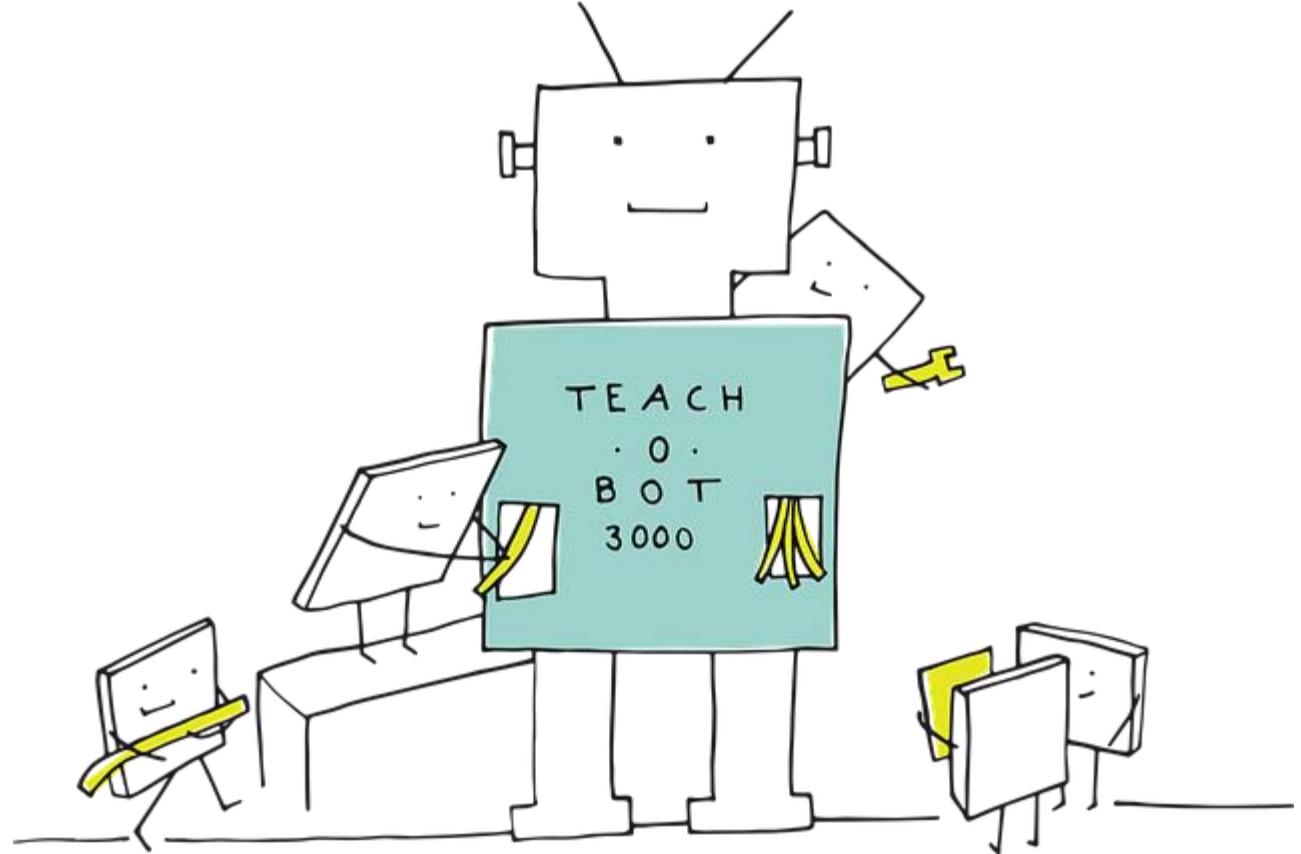




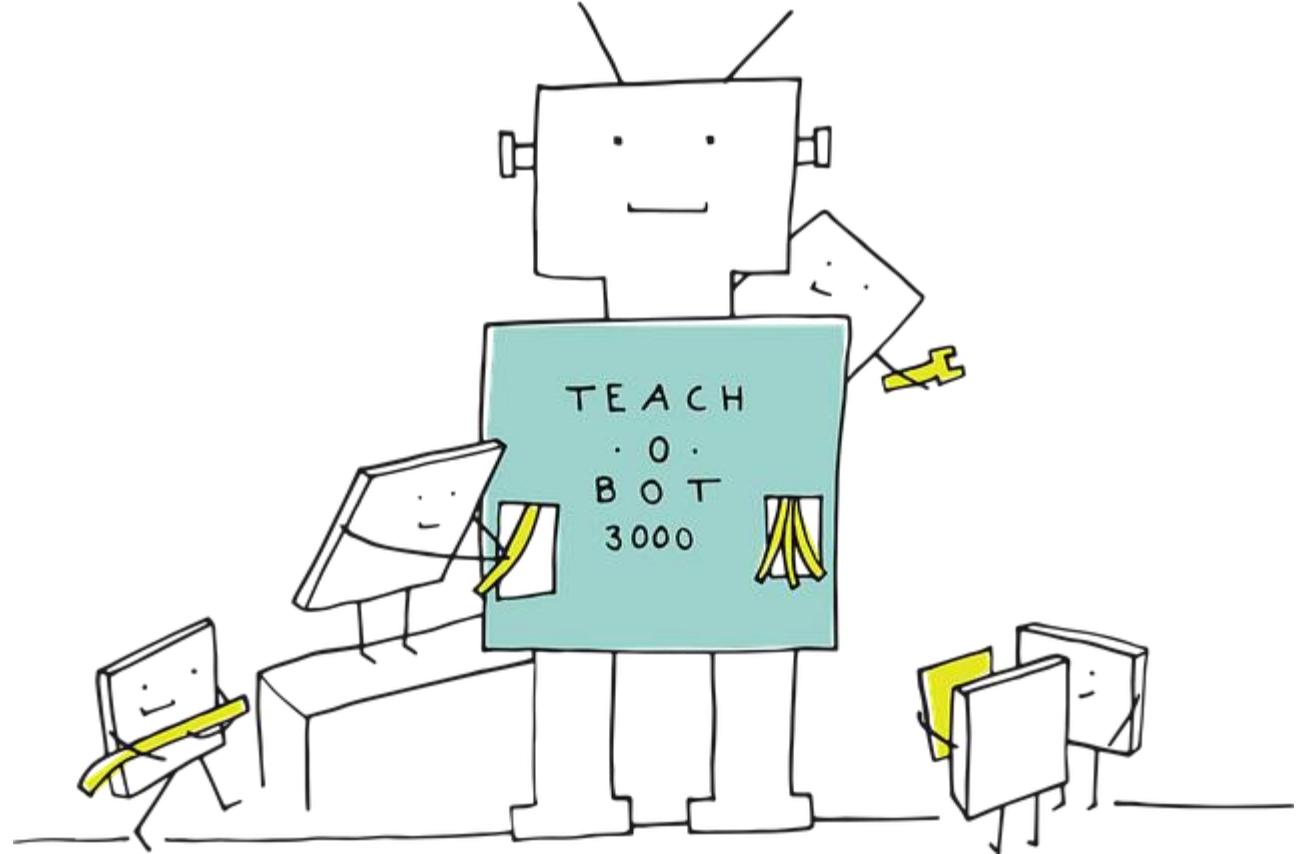
**SOICT**

**HANOI UNIVERSITY OF  
SCIENCE AND TECHNOLOGY**  
SCHOOL OF INFORMATION AND  
COMMUNICATION TECHNOLOGY



**BÁO CÁO TUẦN 1**  
**HỌC MÁY CƠ BẢN**

**NHÓM 8**



*Ngày 30 tháng 9 năm 2023*

# MỤC LỤC - TÓM TẮT NỘI DUNG

1. Làm quen với học máy
2. Học có giám sát: Các bài toán *Hồi quy* và *Phân loại*
3. Học không giám sát: Các bài toán *Phân cụm* và *Giảm chiều*
4. Cơ bản về Học sâu: Mạng thần kinh nhân tạo



**SOICT**

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY  
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

1. Làm quen với học máy
2. Học có giám sát: Các bài toán *Hồi quy* và *Phân loại*
3. Học không giám sát: Các bài toán *Phân cụm* và *Giảm chiều*
4. Cơ bản về Học sâu: Mạng thần kinh nhân tạo

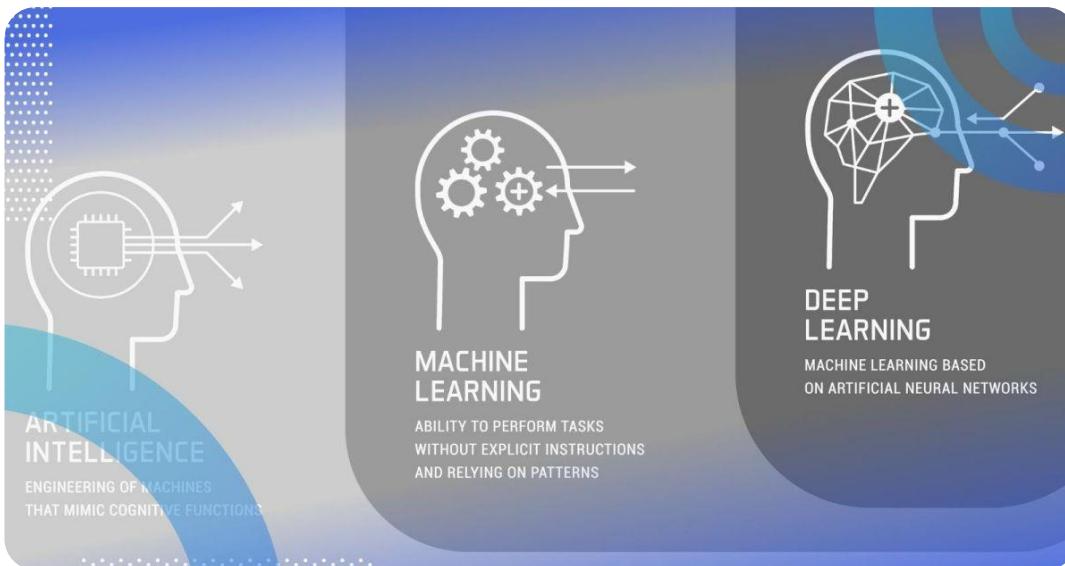


**SOICT**

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY  
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

# KHÁI NIỆM HỌC MÁY

Học máy là lĩnh vực con của *Khoa học dữ liệu* và *Trí tuệ nhân tạo*. Học sâu là một phần của Học máy.



## TRÍ TUỆ NHÂN TẠO

Máy móc mô phỏng hành vi thông minh của con người.

## HỌC MÁY

Máy móc học từ dữ liệu mà không cần lập trình cụ thể.

Sử dụng mạng thần kinh nhân tạo nhiều lớp để xử lý dữ liệu.

## HỌC SÂU

## THÔNG KÊ

## TỐI ƯU

# PHÂN LOẠI HỌC MÁY PHỔ BIẾN

## HỌC CÓ GIÁM SÁT

- Mô hình được huấn luyện trên tập dữ liệu đã được gắn nhãn.
- Mục tiêu là dự đoán nhãn đầu ra cho dữ liệu mới.

## HỌC KHÔNG GIÁM SÁT

- Mô hình được huấn luyện trên dữ liệu không có nhãn.
- Mục tiêu chủ yếu là tìm ra cấu trúc hoặc mô hình trong dữ liệu.

## HỌC TĂNG CƯỜNG

- Tác tử tương tác với môi trường để tối ưu hóa phần thưởng.
- Có phần thưởng (hoặc hình phạt) dựa trên hành động của tác tử.

## PHÂN LOẠI

Nhãn rời rạc và hữu hạn.

## HỘI QUY

Nhãn là số thực cụ thể.

## PHÂN CỤM

Tổ chức dữ liệu thành các nhóm (cụm) dựa trên sự giống nhau giữa chúng.

## GIẢM CHIỀU

Biến đổi dữ liệu từ không gian nhiều chiều về không gian chiều thấp hơn, giữ lại thông tin quan trọng nhất.

## MODEL-BASED

Cố gắng xây dựng một mô hình của môi trường và sử dụng mô hình đó để đưa ra quyết định.

## MODEL-FREE

Học trực tiếp từ kinh nghiệm thông qua việc tương tác với môi trường



**SOICT**

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY  
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

# GRADIENT DESCENT

Ta sẽ thiết lập công thức để tìm đến tối ưu địa phương  $x^*$  của một hàm số  $f(x)$  khả vi liên tục bằng cách lặp lại các giá trị của  $x$ . Không mất tính tổng quát, ta có thể giả sử bài toán có mục đích cực tiểu.

Gọi  $x_t$  là kết quả sau đúng  $t$  vòng lặp.

## Nhận xét:

- Nếu  $f'(x_t) > 0$  thì  $x_t$  ở bên phải  $x^*$  và cần giảm, nếu  $f'(x_t) < 0$  thì  $x_t$  ở bên trái  $x^*$  và cần tăng. Tóm lại:

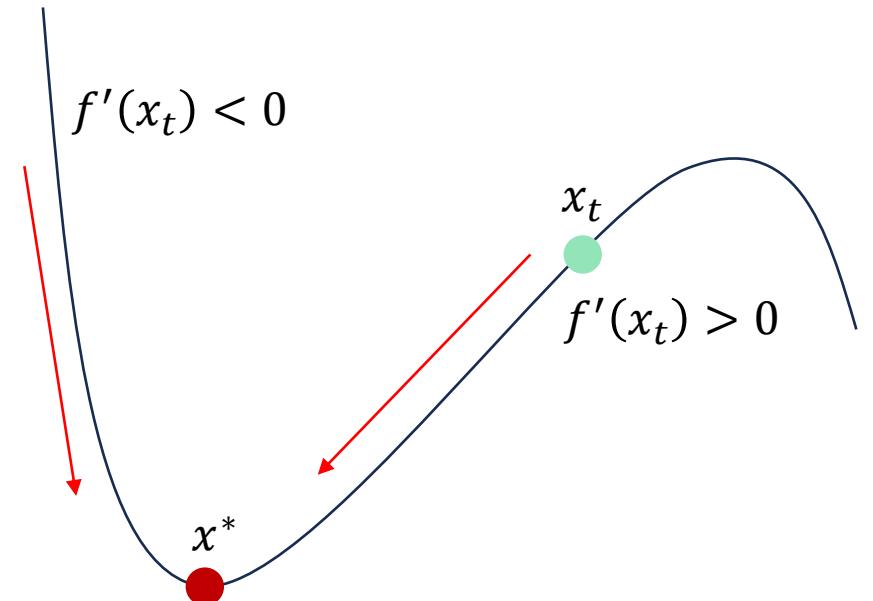
$$x_{t+1} = x_t + \Delta$$

Trong đó  $\Delta$  trái dấu với  $f'(x_t)$ .

- $f'(x^*) = 0$  và càng gần  $x^*$  thì đạo hàm càng gần 0. Do đó, ta chọn  $\Delta$  tỉ lệ thuận với  $-f'(x_t)$  nghĩa là:

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

Trong đó  $\eta$  là *tốc độ học*. Điều kiện dừng  $f'(x^*) \cong 0$ .



# GRADIENT DESCENT

Việc chọn  $\eta$  rất quan trọng.

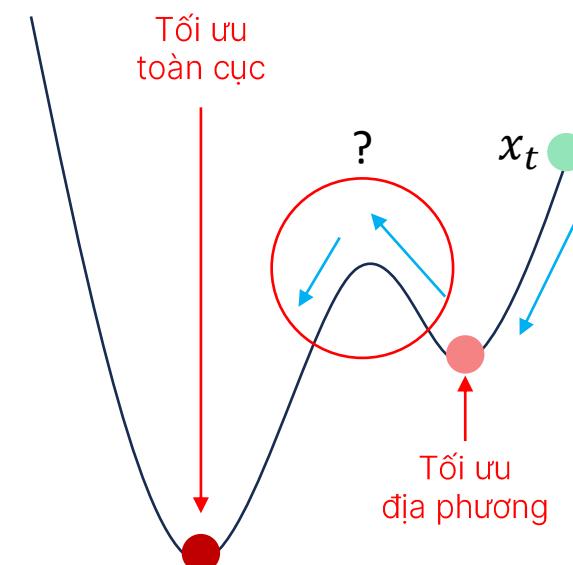
- Nếu  $\eta$  lớn thì thuật toán có thể nhanh đến gần tối ưu địa phương, nhưng lại quanh quẩn vùng này.
- Nếu  $\eta$  nhỏ, thuật toán mất nhiều vòng lặp để hội tụ, thậm chí không thể đến được đích.

Với hàm nhiều biến, trong đó  $x \in \mathbb{R}^n$  và cần tìm cực tiểu địa phương của  $f(x)$  thì ta cập nhật như sau:

$$x_{t+1} = x_t - \eta \frac{\partial f}{\partial x}(x_t)$$

Trong trường hợp muốn tìm tối ưu toàn cục thực sự, nếu áp dụng giải thuật này, sẽ có trường hợp rơi vào tối ưu địa phương thay vì tối ưu toàn cục.

Nếu có đủ “vận tốc”,  $x_t$  sẽ trượt khỏi tối ưu địa phương và tiếp tục rơi để đến với tối ưu toàn cục.



# GRADIENT DESCENT: CÁC BIẾN THỂ

## GRADIENT DESCENT VỚI MOMENTUM

Ta có thể coi vận tốc ban đầu  $v_0 = 0$  và vận tốc mới là tổng hợp của vận tốc cũ cùng độ dốc:

$$v_{t+1} = \gamma v_t + \eta f'(x_t)$$

Với  $0 < \gamma < 1$ . Vị trí mới được cập nhật thành:

$$x_{t+1} = x_t - v_t$$

## NESTEROV ACCELERATED GRADIENT

Để cải tiến tốc độ hội tụ của phương pháp trên, thay vì di chuyển theo gradient cũ, ta sẽ “nhìn trước” gradient, nghĩa là:

$$v_{t+1} = \gamma v_t + \eta f'(x_t - \gamma v_t)$$

Vị trí mới được cập nhật thành:

$$x_{t+1} = x_t - v_t$$

## STOCHASTIC GRADIENT DESCENT

Trong thuật toán này:

- Tại 1 thời điểm, ta chỉ tính đạo hàm của hàm mất mát dựa trên chỉ một điểm dữ liệu rồi cập nhật  $x$  dựa trên đạo hàm này.
- Việc này được thực hiện với từng điểm trên toàn bộ dữ liệu, sau đó lặp lại quá trình trên.

Việc cập nhật từng điểm làm giảm tốc độ lướt qua tất cả dữ liệu, nhưng ngược lại, thuật toán này cần ít hơn số lần lướt qua tất cả điểm dữ liệu để đạt được nghiệm tối ưu.

Công thức cập nhật:

$$x_{t+1} = x_t - \frac{\partial f}{\partial w}(w; x_i, y_i)$$



**SOICT**

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY  
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

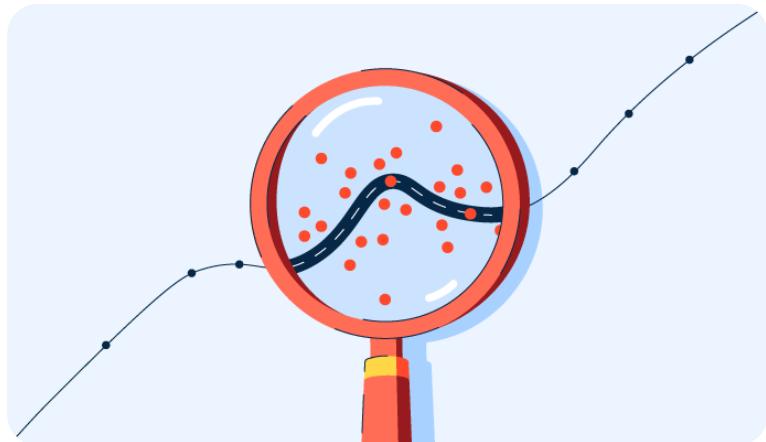
1. Làm quen với học máy
2. Học có giám sát: Các bài toán *Hồi quy* và *Phân loại*
3. Học không giám sát: Các bài toán *Phân cụm* và *Giảm chiều*
4. Cơ bản về Học sâu: Mạng thần kinh nhân tạo



**SOICT**

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY  
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

# CÁC BÀI TOÁN HỒI QUY



## PHÁT BIỂU BÀI TOÁN

Tập dữ liệu huấn luyện:

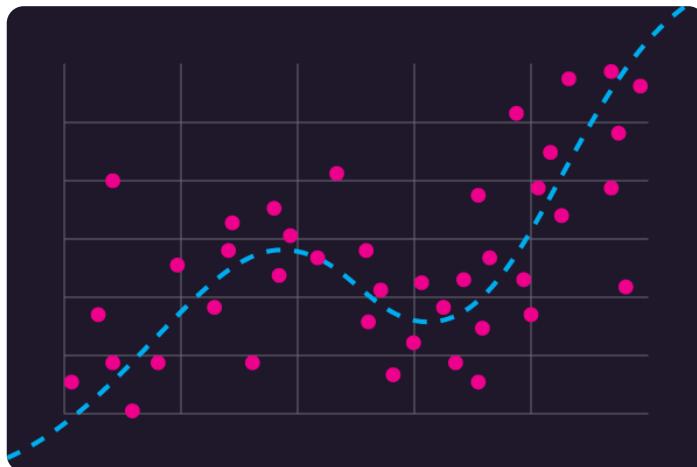
$$\{(x_i, y_i)\}_{i=1}^n \subseteq \mathbb{R}^p \times \mathbb{R}$$

Mục tiêu tổng quát:

$$y_i \cong f(x_i), \forall i = 1, 2, \dots, n$$

## ỨNG DỤNG

- Dựa vào quy luật phát hiện được để dự đoán.
- Phát hiện xu hướng, xác định quan hệ giữa các biến.



## MÔ HÌNH PHỔ BIÊN

Tùy vào dạng của hàm dự đoán mà chia thành một số phương pháp:

- Hồi quy tuyến tính
- Hồi quy đa thức
- Hồi quy logistic
- ....

## LƯU Ý

- Overfitting: Mô hình quá phức tạp so với dữ liệu, dẫn đến việc mô hình hoạt động tốt trên tập huấn luyện nhưng không tốt trên dữ liệu mới.
- Underfitting: Mô hình quá đơn giản và không nắm bắt được mối quan hệ trong dữ liệu.

# HỒI QUY TUYẾN TÍNH

Trong bài toán này, biến dự đoán  $y$  phụ thuộc tuyến tính vào  $n$  biến độc lập  $x_1, x_2, x_3, \dots, x_n$

Kí hiệu  $X = (1, x_1, x_2, \dots, x_n)^T$  để chỉ biến phản ánh

$X_i = (1, x_{1i}, x_{2i}, \dots, x_{ni})^T$  là biến phản ánh trong quan sát thứ  $i$

Tập dữ liệu huấn luyện gồm  $N$  bộ số  $\{(X_i, y_i)\}_{i=1}^N$  đã biết.

Mục tiêu là xác định vector hệ số  $w = (w_0, w_1, w_2, \dots, w_n)^T$  để dự đoán gần đúng:

$$y \cong f(X) = X^T w = w_0 + w_1 x_1 + w_2 x_2 + \dots + w_n x_n$$

Sai số bình phương của dự đoán so với lần quan sát thứ  $i$  là:

$$e_i^2 = (y_i - f(X_i))^2 = (y_i - X_i^T w)^2$$

Trung bình sai số trong  $N$  quan sát là:

$$\frac{1}{N} \sum_{i=1}^N e_i^2 = \frac{1}{N} \sum_{i=1}^N (y_i - X_i^T w)^2$$



**SOICT**

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY  
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

# BÌNH PHƯƠNG TỐI THIỂU

Vì N cố định và để lấy đạo hàm đơn giản, xét hàm mục tiêu:

$$\mathcal{L}(w) = \frac{1}{2} \sum_{i=1}^N (y_i - X_i^T w)^2 = \frac{1}{2} \|y - Xw\|_2^2$$

Với  $y = (y_1, y_2, \dots, y_N)^T$  và  $X = (X_1, X_2, \dots, X_N)$  là các ma trận dữ liệu huấn luyện.

Tóm lại, cần giải bài toán tối ưu không ràng buộc:

$$\min_{w \in \mathbb{R}^{n+1}} \mathcal{L}(w) = \min_{w \in \mathbb{R}^{n+1}} \left\{ \frac{1}{2} \|y - Xw\|_2^2 \right\}$$

Hàm mục tiêu khả vi, lấy đạo hàm theo vector  $w$  có:

$$\frac{\partial \mathcal{L}}{\partial w}(w) = X^T(Xw - y); \frac{\partial \mathcal{L}}{\partial w}(w) = 0 \Leftrightarrow X^T Xw = X^T y$$

Nếu ma trận  $X^T X$  khả nghịch thì bài toán có nghiệm cực tiểu toàn cục duy nhất  $w^* = (X^T X)^{-1} X^T y$

Ngược lại, thì bài toán vô nghiệm, phương pháp này không hiệu quả.



**SOICT**

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY  
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

# HỒI QUY RIDGE

Câu hỏi đặt ra:

- Có hay không quan hệ đồng tuyến giữa các biến độc lập  $x_1, x_2, x_3, \dots, x_n$ ?
- Có hay không giải pháp khác cho bài toán?
- Khi số biến tăng lên, độ lớn của hệ số hồi quy có thể điều chỉnh không?

Ta vẫn sử dụng hàm mục tiêu  $\mathcal{L}(w) = \frac{1}{N} \|y - \mathcal{X}w\|_2^2$  nhưng khác với bình phương tối thiểu, hồi quy Ridge tương đương bài toán tối ưu có ràng buộc:

$$\min_{\|w\|_2^2 \leq t} \mathcal{L}(w) = \min_{\|w\|_2^2 \leq t} \frac{1}{N} \|y - \mathcal{X}w\|_2^2$$

Theo phương pháp nhân tử Larange, đưa về bài toán tối ưu với hệ số điều chỉnh  $\lambda > 0$  như sau:

$$\min_{w \in \mathbb{R}^{n+1}} \mathcal{L}^*(w) = \min_{w \in \mathbb{R}^{n+1}} \left\{ \frac{1}{N} \|y - \mathcal{X}w\|_2^2 + \lambda \|w\|_2^2 \right\}$$



**SOICT**

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY  
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

# HỒI QUY RIDGE

Hàm mục tiêu khả vi, lấy đạo hàm theo vector  $w$  có:

$$\frac{\partial \mathcal{L}}{\partial w}(w) = \frac{2}{N} \mathcal{X}^T (\mathcal{X}w - y) + 2\lambda w = \frac{2}{N} [(\mathcal{X}^T \mathcal{X} + N\lambda I_{n+1})w - \mathcal{X}^T y]$$

Nghiệm cực tiểu toàn cục  $w^* = (\mathcal{X}^T \mathcal{X} + N\lambda I_{n+1})^{-1} \mathcal{X}^T y$

Mátrận  $\mathcal{X}^T \mathcal{X} + N\lambda I_{n+1}$  luôn khả nghịch với  $\lambda > 0$  nên bài toán hồi quy Ridge luôn có nghiệm.

Nhận xét:

- Khi  $\lambda \rightarrow 0$ , kết quả thu được gần giống phương pháp bình phương tối thiểu.
- Hồi quy Ridge giúp giảm độ lớn các hệ số do có ràng buộc tối ưu.
- Hồi quy Ridge cho phép điều chỉnh hệ số  $\lambda$ , cung cấp một dải rộng hơn các giải pháp.



**SOICT**

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY  
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

# HỒI QUY LASSO

Trong mô hình dự đoán:

$$y \cong f(X) = X^T w = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$

- Liệu có ít nhất một trong các biến  $x_1, x_2, x_3, \dots, x_n$  hữu ích trong việc giải thích hoặc dự báo biến  $y$ ?
- Tất cả các biến  $x_1, x_2, x_3, \dots, x_n$  hay chỉ một tập con các biến là cần thiết trong mô hình?

Hồi quy Lasso là bài toán tối ưu có ràng buộc:

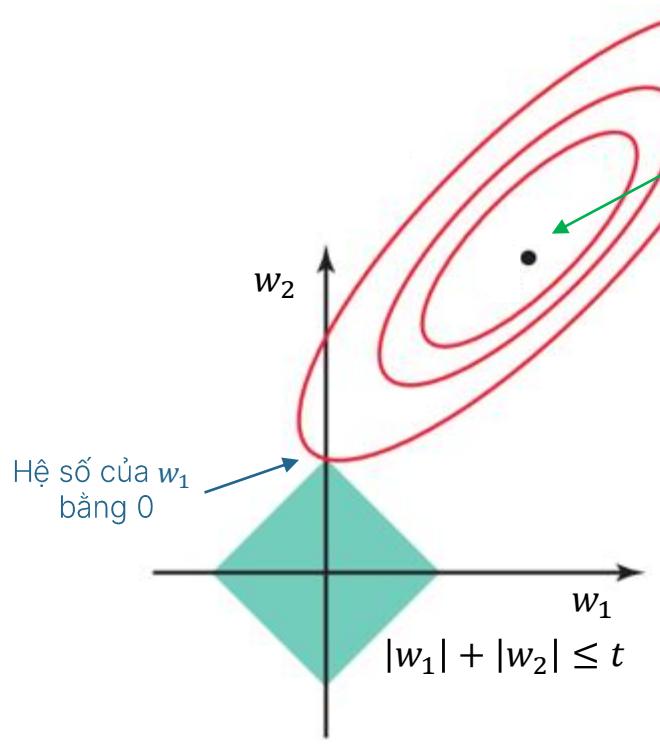
$$\min_{\|w\|_1 \leq t} \mathcal{L}(w) = \min_{\|w\|_1 \leq t} \frac{1}{N} \|y - \mathcal{X}w\|_2^2$$

Tương đương bài toán tối ưu với hệ số điều chỉnh  $\lambda > 0$  là:

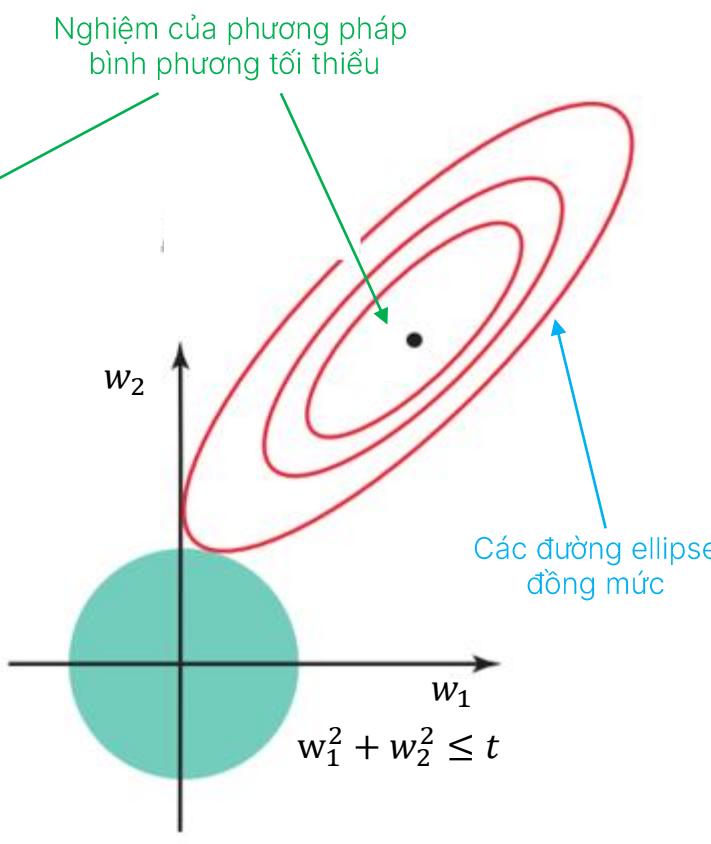
$$\min_{w \in \mathbb{R}^{n+1}} \mathcal{L}^*(w) = \min_{w \in \mathbb{R}^{n+1}} \left\{ \frac{1}{N} \|y - \mathcal{X}w\|_2^2 + \lambda \|w\|_1 \right\}$$

Đây là hàm lồi (không chặt), không khả vi tại một số điểm. Nghiệm cực tiểu địa phương của bài toán cũng là nghiệm cực tiểu toàn cục.

# HỒI QUY LASSO



Lasso Regression



Ridge Regression

- Với những  $t$  đủ nhỏ, một số hệ số  $w_i \cong 0$  do bị ràng buộc bởi chuẩn  $\|\cdot\|_1$ , do đó ta có thể lược bỏ biến phụ thuộc  $x_i$  khỏi mô hình.
- Bằng việc điều chỉnh tham số  $\lambda$  có thể thu được nhiều giải pháp cho mô hình.

# HỒI QUY LOGISTIC

Mô hình logistic regression dự đoán xác suất xảy ra một sự kiện là tổ hợp tuyến tính của một hay nhiều biến độc lập. Kí hiệu:

- $X = (X_1, X_2, \dots, X_n)$  để chỉ biến phản ánh.
- $X_i = (x_{1i}, x_{2i}, \dots, x_{ni}) \in \mathbb{R}^n$  là biến phản ánh trong quan sát thứ  $i$ .
- $y_i \in \{0,1\}$  là đầu ra trong quan sát thứ  $i$ , tương ứng với 2 nhãn.

Tập dữ liệu huấn luyện gồm  $N$  bộ số  $\{(X_i, y_i)\}_{i=1}^N$  đã biết.

Mục tiêu là xác định vector hệ số  $w = (w_1, w_2, \dots, w_n)$  để dự đoán gần đúng:

$$y \cong f(X) = \theta(w^T X)$$

Trong đó  $\theta$  là logistic function, thường sử dụng nhất là hàm sigmoid:

$$\theta(s) = \frac{1}{1 + e^{-s}} = \sigma(s).$$

Hàm số này có các đặc tính sau:

- Là hàm số nhận giá trị trong  $(0, 1)$ .
- Nếu coi điểm có tung độ  $0.5$  làm điểm phân chia thì các điểm càng xa điểm này về bên trái có giá trị càng gần  $0$ , càng xa điểm này về bên phải có giá trị càng gần  $1$ .
- Là hàm số trơn, có lợi trong việc tối ưu.



**SOICT**

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY  
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

# HỒI QUY LOGISTIC: HÀM MẤT MÁT

Ta có thể giả sử xác suất để điểm dữ liệu  $X_i$  rơi vào class 1 là  $f(w^T X_i)$ , rơi vào class 0 là  $1 - f(w^T X_i)$ :

$$\begin{cases} P(y_i = 1 | X_i, w) = f(w^T X_i) \\ P(y_i = 0 | X_i, w) = 1 - f(w^T X_i) \end{cases} \Rightarrow P(y_i | X_i, w) = z_i^{y_i} (1 - z_i)^{1-y_i}$$

Trong đó  $z_i = f(w^T X_i)$ . Để mô hình gần với dữ liệu nhất ta cần xác suất trên đạt giá trị lớn nhất. Xét toàn bộ tệp dữ liệu huấn luyện. Giả sử các điểm dữ liệu sinh ra độc lập với nhau thì:

$$P(y | X, w) = \prod_{i=1}^N P(y_i | X_i, w) = \prod_{i=1}^N z_i^{y_i} (1 - z_i)^{1-y_i}$$

Lấy logarit Neper để biến tích thành tổng, lấy ngược dấu để được một hàm mà ta coi là hàm mất mát:

$$\mathcal{J}(w) = -\ln P(y | X, w) = -\sum_{i=1}^N (y_i \ln z_i + (1 - y_i) \ln(1 - z_i))$$

Để  $P(y | X, w)$  lớn nhất ta cần  $\mathcal{J}(w)$  nhỏ nhất, tức là giải bài toán tối ưu:

$$w^* = \operatorname{argmin} \mathcal{J}(w)$$



**SOICT**

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY  
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

# HỒI QUY LOGISTIC

Tối ưu hàm mất mát: Ta sử dụng thuật toán Stochastic Gradient Descent (SGD). Hàm mất mát với chỉ một điểm dữ liệu  $(X_i, y_i)$  là:

$$\mathcal{J}(w, X_i, y_i) = -(y_i \ln z_i + (1 - y_i) \ln(1 - z_i))$$

với đạo hàm riêng:

$$\frac{\partial \mathcal{J}(w, X_i, y_i)}{\partial w} = -\left(\frac{y_i}{z_i} - \frac{1 - y_i}{1 - z_i}\right) \frac{\partial z_i}{\partial w} = \frac{z_i - y_i}{z_i(1 - z_i)} \frac{\partial z_i}{\partial w} \quad (1)$$

Ta có  $z_i = f(w^T X_i)$

Nếu chọn  $f$  là hàm số *sigmoid* và đặt  $s = w^T X_i$  thì ta dễ kiểm tra  $\frac{\partial z_i}{\partial s} = z_i(1 - z_i)$ , tức là:

$$\frac{\partial z_i}{\partial w} = \frac{\partial z_i}{\partial s} \frac{\partial s}{\partial w} = z_i(1 - z_i) X_i \quad (2)$$

Thay (2) vào (1) ta được:

$$\frac{\partial \mathcal{J}(w, X_i, y_i)}{\partial w} = (z_i - y_i) X_i$$

Công thức cập nhật (theo thuật toán SGD) cho logistic regression là

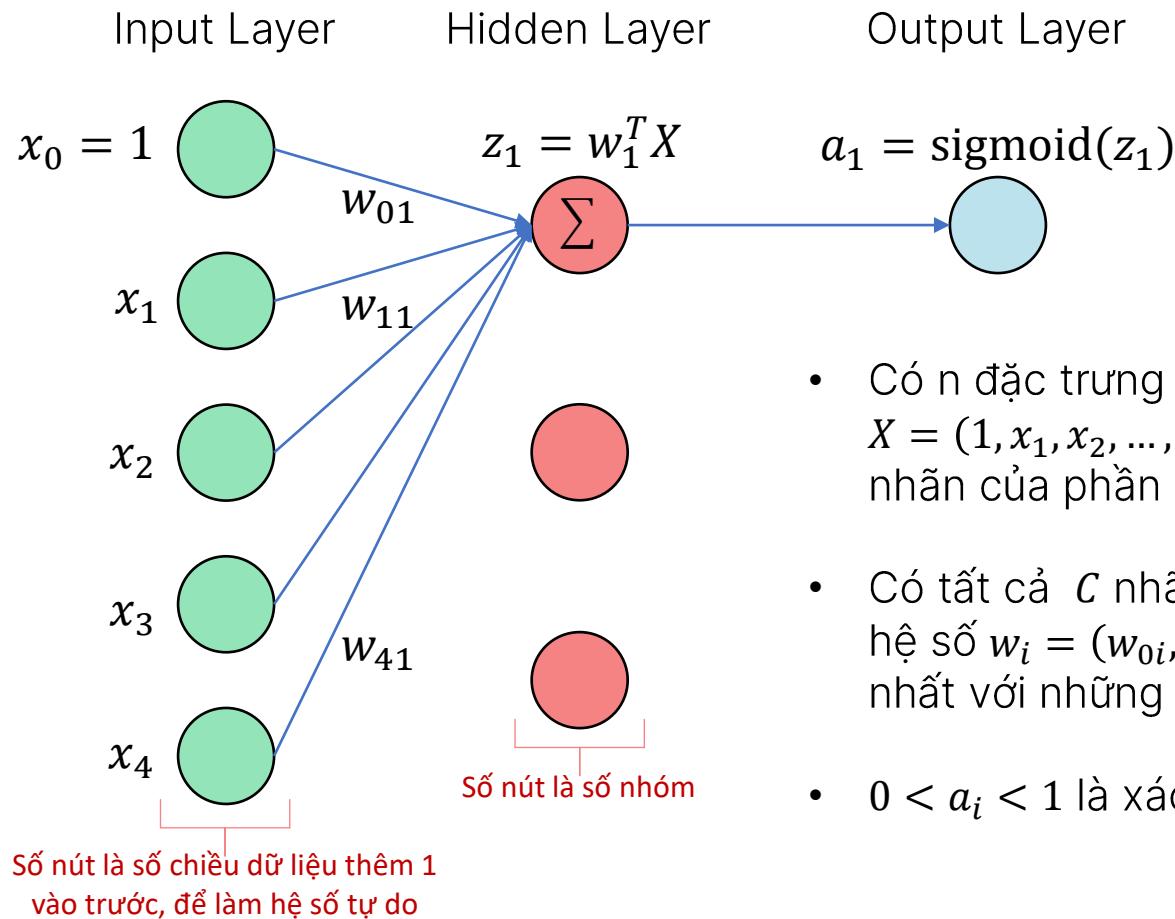
$$w = w + \eta(y_i - z_i) X_i$$



**SOICT**

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY  
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

# MÔ HÌNH HỒI QUY LOGISTIC



- Có  $n$  đặc trưng  $x_1, x_2, \dots, x_n$  và ta bổ sung 1 vào trước để có ma trận  $X = (1, x_1, x_2, \dots, x_n)^T$ . Dữ liệu huấn luyện là  $\{(X_i, y_i)\}_{i=1}^N$  trong đó  $y_i$  là nhãn của phần tử  $X_i$ .
- Có tất cả  $C$  nhãn hay là  $y_i \in \{1, 2, \dots, C\}$ . Với mỗi nhãn  $i$ , ta tìm vector hệ số  $w_i = (w_{0i}, w_{1i}, \dots, w_{ni})^T \in \mathbb{R}^{n+1}$  sao cho  $a_i = \text{sigmoid}(w_i^T X)$  gần 1 nhất với những điểm  $X$  mang nhãn  $i$ .
- $0 < a_i < 1$  là xác suất dữ liệu  $X_i$  rơi vào nhãn  $i$ .

# HỒI QUY SOFTMAX

Trong phần trước, ta đã thiết kế mạng nơ-ron đơn giản cho bài toán phân loại dữ liệu theo  $C$  nhãn bằng hồi quy logistic. Tuy nhiên, hạn chế của hồi quy logistic là nó không đảm bảo tính đúng đắn thống kê.

Với hồi quy softmax, ta đảm bảo tính chất sau xảy ra:

$$\sum_{i=1}^C a_i = 1$$

Ngoài ra, ta có thể cho rằng nếu  $z_i = w_i^T X$  càng lớn thì xác suất  $X$  rơi vào nhãn  $i$  càng cao, và đây là đại lượng không xác định âm, dương.

Vậy, ta chọn:

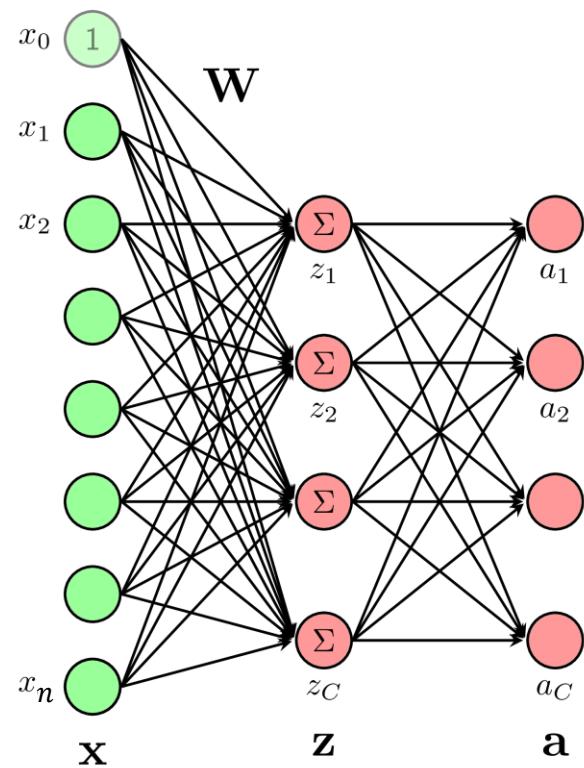
$$a_i = \frac{e^{z_i}}{\sum_{j=1}^C e^{z_j}}$$

Hàm số này thỏa mãn: phụ thuộc vào tất cả  $z_i$ , thuộc  $(0,1)$ , có tổng bằng 1 và đảm bảo thứ tự các  $z_i$ .

# MÔ HÌNH HỒI QUY SOFTMAX

Mục tiêu là tìm ma trận trọng số  $W = (w_1, w_2, \dots, w_C) \in \mathbb{R}^{(n+1) \times C}$  nhằm tính toán:

$$P(y_k = i|x_k; W) = a_i$$



$$x_i \xrightarrow{w_{ij}} \Sigma \rightarrow z_j$$

$w_{0j}$ : biases, don't forget!

$n$ : data dimension

$C$ : number of classes

$$\mathbf{x} \in \mathbb{R}^{n+1}$$

$$\mathbf{W} \in \mathbb{R}^{(n+1) \times C}$$

$$z_i = \mathbf{w}_i^T \mathbf{x}$$

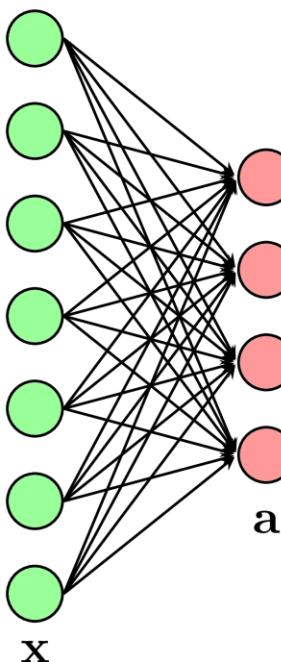
$$\mathbf{z} = \mathbf{W}^T \mathbf{x} \in \mathbb{R}^C$$

$$\mathbf{a} = \text{softmax}(\mathbf{z}) \in \mathbb{R}^C$$

$$a_i > 0, \quad \sum_{i=1}^C a_i = 1$$

$$\mathbf{z} = \text{softmax}(\mathbf{W}^T \mathbf{x})$$

short form



Ta có thể biểu diễn mô hình hồi quy này dưới dạng mạng nơ-ron.

- Khác với hồi quy logistic, hồi quy softmax có sử dụng thêm mối quan hệ giữa các điểm dữ liệu.
- Hồi quy softmax cho phép phân loại nhiều hơn 2 nhãn.

# CROSS ENTROPY

Với  $p = \{p_i\}_{i=1}^C$  và  $q = \{q_i\}_{i=1}^C$  là hai phân phối xác suất rời rạc. Ta định nghĩa *cross entropy* cho bởi:

$$H(p, q) = - \sum_{i=1}^C p_i \ln(q_i)$$

## Ứng dụng trong học máy:

- $p$  thường là phân phối xác suất thực sự và  $q$  là phân phối mà mô hình dự đoán.
- Cross entropy thường được sử dụng như một hàm mất mát (loss function) trong các mô hình phân loại. Mục tiêu là giảm thiểu giá trị cross entropy, làm cho phân phối dự đoán  $q$  gần với phân phối thực sự  $p$  nhất có thể.
- Khi một sự kiện thực sự xảy ra (với xác suất thực sự gần 1) nhưng mô hình dự đoán xác suất của nó gần với 0, logarit của một số rất nhỏ sẽ cho một giá trị âm lớn. Điều này có nghĩa là mô hình sẽ bị trừng phạt rất mạnh mẽ.



**SOICT**

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY  
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

# THIẾT LẬP HÀM MẤT MÁT

Trong hồi quy logistic, giá trị thực sự là phân phối  $y = \{y_i\}_{i=1}^N$  và giá trị dự đoán là  $a = \{a_i\}_{i=1}^N$  ứng với việc sử dụng vector trọng số  $w$  và  $a = \text{sigmoid}(w^T X)$ .

Khi đó, cross entropy lấy làm hàm mất mát cho bởi:

$$\mathcal{J}(w) = - \sum_{i=1}^N y_i \ln(a_i) + (1 - y_i) \ln(1 - a_i)$$

Trong hồi quy softmax, cross entropy của một điểm  $X_i$  được tính bằng:

$$\mathcal{J}(W; X_i, y_i) = - \sum_{j=1}^C y_{ji} \ln(a_{ji})$$

Trong đó  $(y_{ji}, a_{ji})$  là cặp giá trị thực sự và dự đoán của  $X_i$  đối với nhãn  $j$ .

Nên cross entropy lấy trên toàn bộ tập huấn luyện  $\mathcal{X} = (X_1, X_2, \dots, X_N)$  và  $\mathcal{Y} = (y_1, y_2, \dots, y_N)^T$  là:

$$\mathcal{J}(W; \mathcal{X}, \mathcal{Y}) = - \sum_{i=1}^N \sum_{j=1}^C y_{ji} \ln(a_{ji})$$



**SOICT**

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY  
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

# TỐI ƯU HÀM MẤT MÁT

Với chỉ một cặp  $(X_i, y_i)$  ta có:

$$\mathcal{J}_i(W) = \mathcal{J}(W; X_i, y_i) = -\sum_{j=1}^c y_{ji} \ln(a_{ji}) = -\sum_{j=1}^c y_{ji} \ln\left(\frac{e^{w_j^T X_i}}{\sum_{k=1}^c e^{w_k^T X_i}}\right) = -\sum_{j=1}^c \left[ y_{ji} w_j^T X_i - y_{ji} \ln\left(\sum_{k=1}^c e^{w_k^T X_i}\right) \right]$$

Theo tính chất xác suất, ta có  $\sum_{j=1}^c y_{ji} = 1$  nên rút gọn thành:

$$\mathcal{J}_i(W) = -\left(\sum_{j=1}^c y_{ji} w_j^T X_i\right) + \ln\left(\sum_{k=1}^c e^{w_k^T X_i}\right)$$

Đạo hàm riêng:

$$\frac{\partial \mathcal{J}_i(W)}{\partial w_j} = -y_{ji} X_i + \frac{e^{w_j^T X_i}}{\sum_{k=1}^c e^{w_k^T X_i}} X_i = (a_{ji} - y_{ji}) X_i = e_{ji} X_i$$

Với  $e_{ji} = a_{ji} - y_{ji}$  là sai số. Và do đó:

$$\frac{\partial \mathcal{J}_i(W)}{\partial W} = X_i \times (e_{1i}, e_{2i}, \dots, e_{ci}) = X_i e_i^T \Rightarrow \frac{\partial \mathcal{J}(W)}{\partial W} = X E^T = X(A - Y)^T$$



**SOICT**

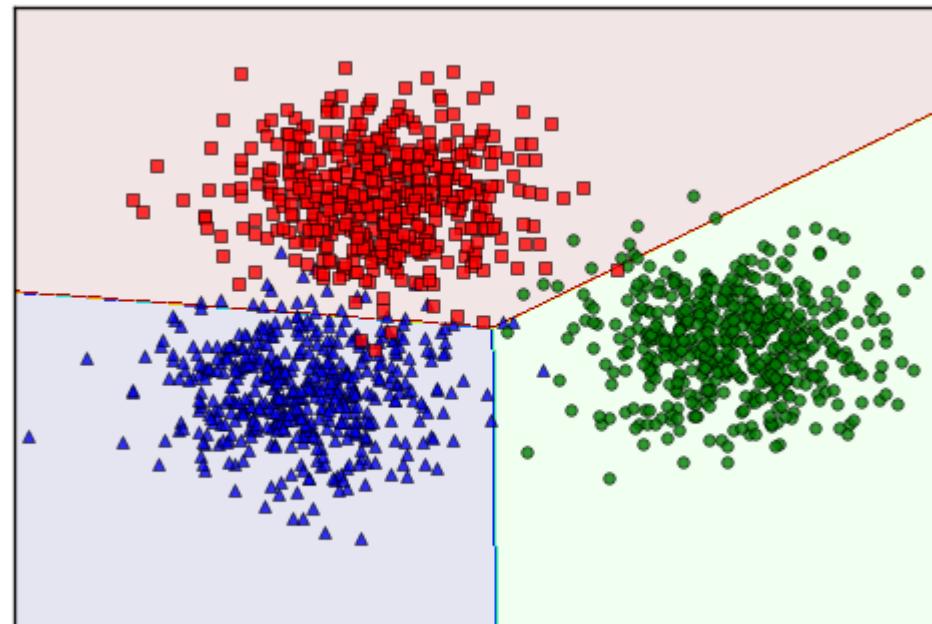
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY  
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

# NHẬN XÉT

Do đó, theo stochastic gradient descent, công thức cập nhật  $W$  là:

$$W = W + \eta X_i(y_i - a_i)^T$$

Công thức này tương tự như đã đề cập trong hồi quy logistic.



- Sau cùng, hồi quy softmax vạch ra ranh giới giữa các nhãn, và ranh giới này là siêu phẳng.
- Trong trường hợp  $e^{z_i}$  quá lớn gây khó khăn tính toán, ta có thể thay các  $z_i$  bởi  $z_i - t$  và:

$$\frac{e^{z_i-t}}{\sum_{j=1}^C e^{z_j-t}} = \frac{e^{-t} e^{z_i}}{e^{-t} \sum_{j=1}^C e^{z_j}} = a_i$$

Nghĩa là không ảnh hưởng đến dự đoán và kết quả thu được. Thường chọn  $t = \min_i z_i$ .

# K LÁNG GIỀNG GẦN NHẤT

Thuật toán KNN áp dụng trong cả bài toán Phân loại và Hồi quy.

- Trong Phân loại, nhãn (nhóm) của điểm dữ liệu mới được suy ra trực tiếp từ nhãn của một số điểm dữ liệu gần nó nhất.
- Trong Hồi quy, giá trị của dự đoán cho điểm dữ liệu mới được lấy theo trung bình của một số điểm gần nó nhất.

Giả sử tập huấn luyện  $\{(x_i, y_i)\}_{i=1}^N$  trong đó  $x_i \in \mathbb{R}^n$  là các điểm trên không gian Euclid n chiều và  $y_i \in \mathbb{R}$  là nhãn của điểm  $x_i$ .

Để xác định nhãn của điểm dữ liệu mới z ta thực hiện các bước sau:

- Bước 1: Với mỗi phần tử  $x_i$ , ta tính “khoảng cách” giữa z và  $x_i$ .
- Bước 2: Chọn ra  $k$  phần tử trong tập huấn luyện có “khoảng cách” gần nhất đối với z.
- Bước 3: Trong  $k$  phần tử đó, nhãn nào xuất hiện nhiều nhất thì gán nhãn đó cho z.



**SOICT**

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY  
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

# K LÁNG GIỀNG GẦN NHẤT

Vậy “*khoảng cách*” giữa phần tử  $z$  tới mỗi phần tử  $x$  trong bộ dữ liệu sẽ được tính như thế nào?

Đối với mỗi bài toán, chúng ta sẽ có những cách tính “khoảng cách” khác nhau dựa trên đặc trưng của các bài toán ấy. Với  $a = (a_1, a_2, \dots, a_n) \in \mathbb{R}^n$  và  $b = (b_1, b_2, \dots, b_n) \in \mathbb{R}^n$ , ta định nghĩa:

- Khoảng cách Euclid bậc 2:  $d(a, b) = \sqrt{\sum_{i=1}^n (a_i - b_i)^2}$
- Khoảng cách Euclid bậc  $p > 0$ :  $d(a, b) = \sqrt[p]{\sum_{i=1}^n (a_i - b_i)^p}$
- Khoảng cách Manhattan:  $d(a, b) = \sum_{i=1}^n |a_i - b_i|$
- Khoảng cách Hamming dùng cho bài toán nhị phân  $a, b \in \{0,1\}^n$

$$d(a, b) = \sum_{i=1}^n \text{dif}(a_i, b_i)$$

Trong đó,  $\text{dif}(a_i, b_i) = 1$  nếu  $a_i \neq b_i$  và  $\text{dif}(a_i, b_i) = 0$  trong trường hợp ngược lại.



**SOICT**

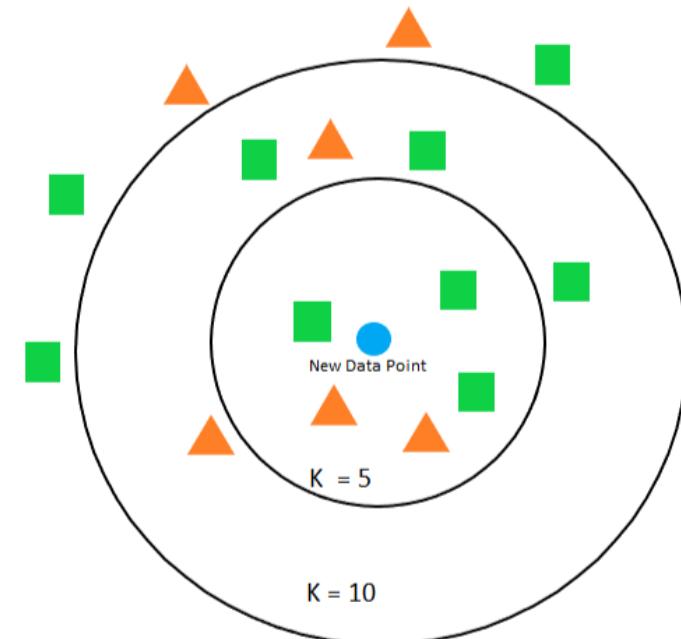
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY  
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

# K LÁNG GIỀNG GẦN NHẤT

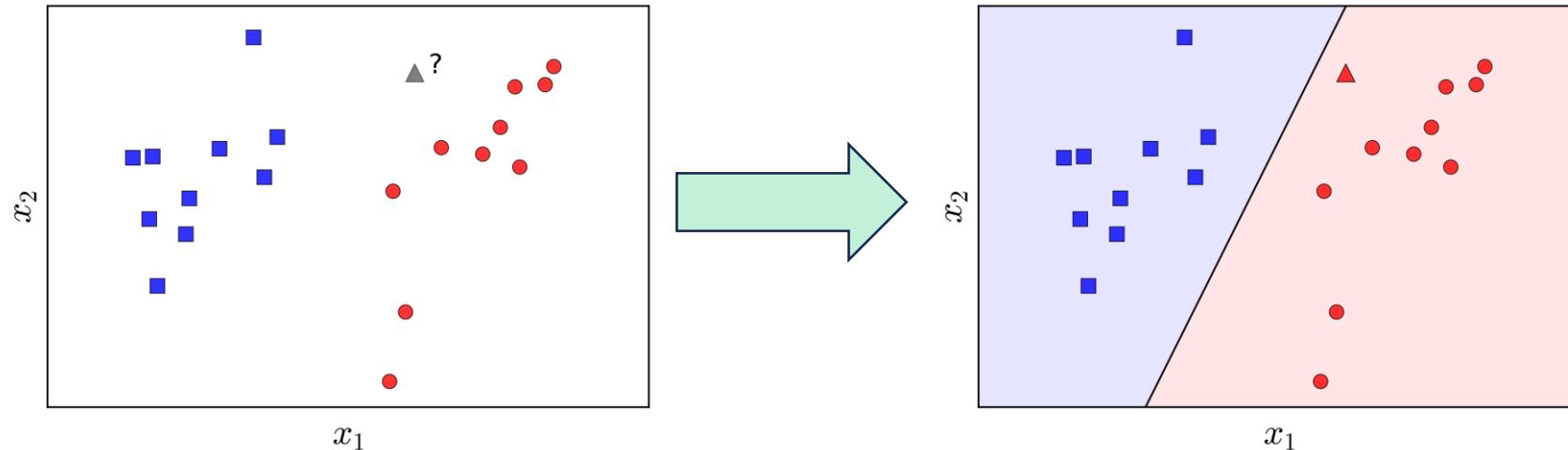
Có rất nhiều cách để ta có thể tính khoảng cách giữa phần tử  $z$  tới mỗi phần tử  $x_i$  trong bộ dữ liệu. Nhưng, ta nên chọn bao nhiêu *k láng giềng* của  $z$  để thuật toán có thể hoạt động một cách tốt nhất?

Thuật toán sẽ hoạt động hiệu quả khi ta chọn số *láng giềng* lớn hơn 1 nhưng không quá nhiều, bởi vì:

- Khi ta chọn số láng giềng của  $z$  lớn hơn 1, ta có thể giảm thiểu được việc gặp phải những điểm dữ liệu nhiễu/lỗi hơn.
- Khi ta chọn số láng giềng của  $z$  quá nhiều, ta sẽ tăng tỉ lệ gặp phải những điểm dữ liệu không liên quan hoặc lỗi làm ảnh hưởng tới chất lượng chọn nhãn cho  $z$ , tăng khả năng Overfitting.



# PERCEPTRON LEARNING ALGORITHM



**Bài toán:** Dữ liệu được gán nhãn gồm 2 nhóm, tìm siêu phẳng phân chia 2 nhóm với giả sử tồn tại siêu phẳng như thế.

Tập dữ liệu huấn luyện gồm  $N$  bộ số  $\{(x_i, y_i)\}_{i=1}^N$  đã biết, trong đó  $x_i \in \mathbb{R}^n$  và  $y_i \in \{-1, 1\}$  để chỉ nhóm của điểm  $x_i$ .

Mở rộng  $x_i$  thành vector  $n + 1$  chiều, bằng cách thêm  $x_0 = 1$  vào trước. Ta cần tìm vector hệ số  $w = (w_0, w_1, \dots, w_n) \in \mathbb{R}^{n+1}$  để siêu phẳng ( $H$ ):  $w^T x = 0$  là ranh giới phân chia 2 nhóm.

# PERCEPTRON LEARNING ALGORITHM

Nhãn (nhóm) của  $x$  xác định bởi hàm dấu  $\text{sgn}(w^T x)$ .

Sử dụng hàm mất mát sau đây:

$$\mathcal{L}(w) = \sum_{x_i \in \mathcal{M}} (-y_i w^T x_i)$$

Trong đó  $\mathcal{M}$  là tập những điểm bị phân loại lỗi. Nếu  $x_i \in \mathcal{M}$  thì  $y_i$  và  $w^T x_i$  trái dấu nên  $\mathcal{L}(w) \geq 0$ .

Với một điểm dữ liệu phân loại lỗi,  $\mathcal{L}_i(y_i, w, x_i) = -y_i w^T x_i$  có đạo hàm riêng:

$$\frac{\partial \mathcal{L}_i(y_i, w, x_i)}{\partial w} = -y_i x_i$$

Vậy quy tắc cập nhật là:

$$w \rightarrow w + \eta y_i x_i$$

Trong đó  $\eta$  là đặc trưng học được chọn bằng 1. Quy tắc có thể viết lại thành:

$$w_{t+1} = w_t + y_i x_i$$

# PERCEPTRON LEARNING ALGORITHM

Khi đó:

$$w_{t+1}^T x_i = (w_t + y_i x_i)^T x_i = w_t^T x_i + y_i \|x_i\|_2^2$$

Nếu  $y_i = 1$  và  $x_i$  bị phân loại lỗi thì  $w_t^T x_i < 0$  và  $y_i \|x_i\|_2^2 = \|x_i\|_2^2 > 0$  nên  $w_{t+1}^T x_i > w_t^T x_i$ .  
Tương tự nếu  $y_i = -1$ . Tóm lại,  $w$  luôn tiến về phía làm cho  $x_i$  phân loại đúng.

## Tóm tắt PLA:

- Bước 1: Chọn ngẫu nhiên một vector hệ số  $w$  với các phần tử gần bằng 0.
- Bước 2: Duyệt ngẫu nhiên qua từng điểm dữ liệu.

Nếu  $x_i$  được phân lớp đúng thì ta không cần làm gì.

Nếu  $x_i$  bị phân lớp sai, cập nhật  $w$  theo công thức:

$$w = w + y_i x_i$$

- Bước 3: Kiểm tra xem còn bao nhiêu điểm bị phân lớp lỗi. Nếu không còn thì dừng thuật toán, nếu còn thì quay lại bước 2. Thuật toán sẽ hội tụ sau hữu hạn bước.

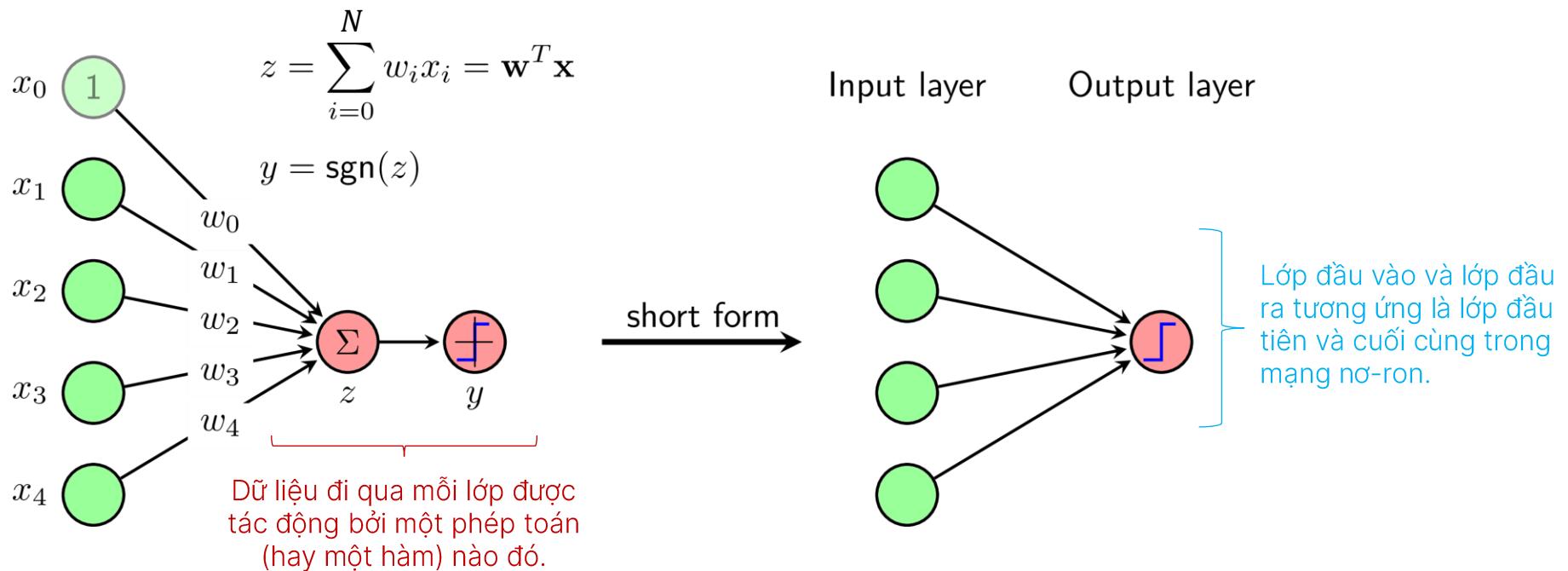


**SOICT**

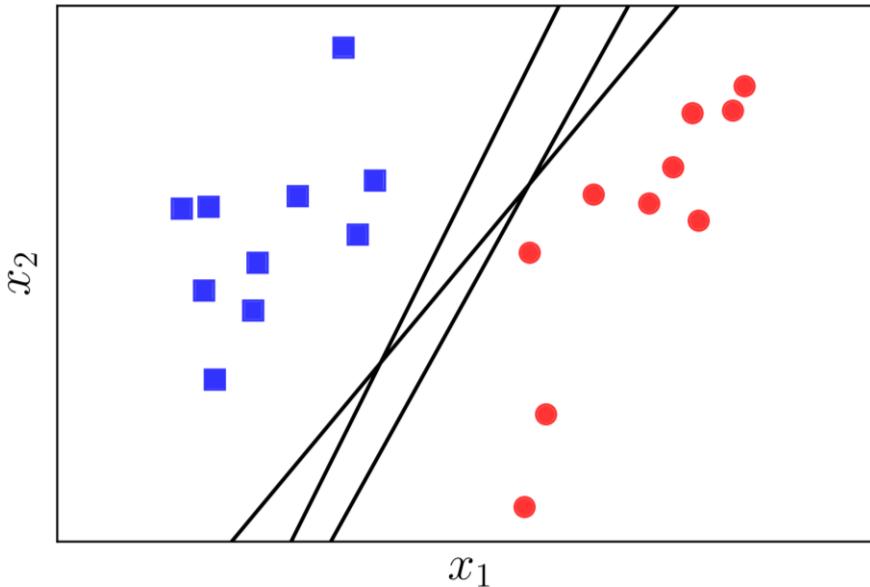
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY  
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

# PERCEPTRON LEARNING ALGORITHM

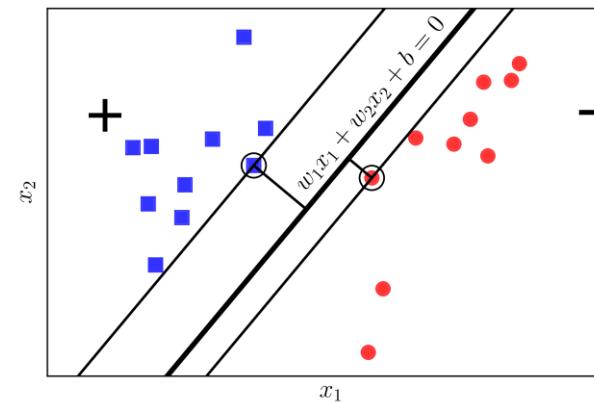
- Bài toán PLA có vô số nghiệm và *không biết* nghiệm nào tốt nhất.
- Bài toán PLA chỉ áp dụng khi dữ liệu là phân biệt tuyến tính thực sự.
- Giải thuật PLA có thể mô phỏng dưới dạng mạng nơ-ron đơn giản:



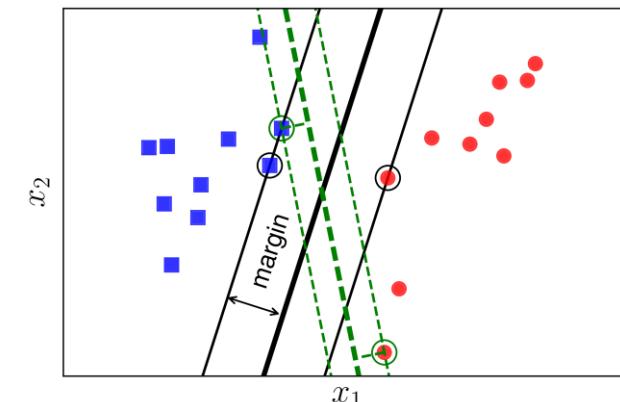
# SUPPORT VECTOR MACHINE



Giả sử 2 nhóm dữ liệu và tồn tại cách phân chia hai nhóm dữ liệu này bởi một siêu phẳng. Sẽ có vô số cách như vậy. Trong chúng, siêu phẳng nào *tốt nhất*?



Nhóm đỏ gần với siêu phẳng hơn nhóm xanh. Đây là các phân chia chưa hợp lý.



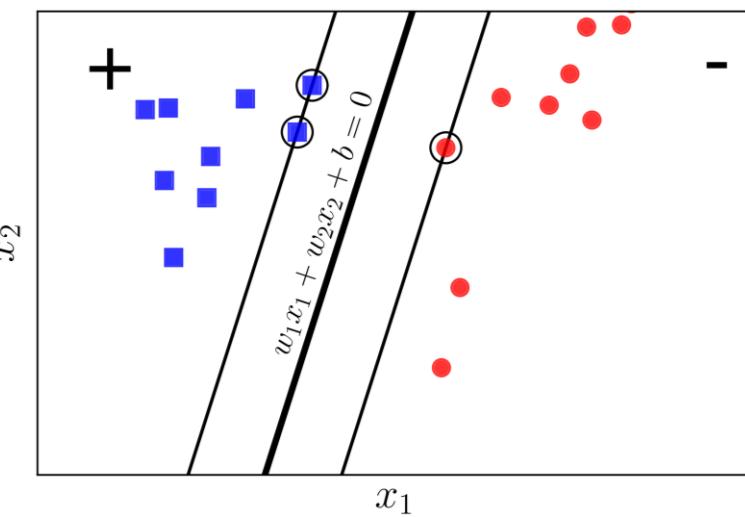
Đường màu xanh quá gần với nhóm xanh và nhóm đỏ mặc dù cách đều điểm gần nhất trong hai nhóm.

→ “Lề” của mỗi nhóm là khoảng cách từ siêu phẳng phân chia đến điểm gần nhất.  
Bài toán SVM cực đại “lề” của 2 nhóm.

# SUPPORT VECTOR MACHINE

Khoảng cách từ điểm  $x_0 \in \mathbb{R}^n$  đến siêu phẳng ( $H$ ):  $w^T x + c = 0$  cho bởi:

$$d(x_0, H) = \frac{|w^T x_0 + c|}{\|w\|_2}$$



Tập dữ liệu huấn luyện gồm  $N$  bộ số  $\{(x_i, y_i)\}_{i=1}^N$  đã biết, trong đó  $x_i \in \mathbb{R}^n$  và  $y_i \in \{-1, 1\}$  để chỉ nhóm của điểm  $x_i$ . Khi đó:

$$d(x_i, H) = \frac{|w^T x_i + c|}{\|w\|_2} = \frac{y_i(w^T x_i + c)}{\|w\|_2}$$

Lề của phép phân chia là:

$$\text{margin} = \min_{i=1,N} \frac{y_i(w^T x_i + c)}{\|w\|_2}$$

Bài toán SVM tương đương bài toán tối ưu không ràng buộc:

$$\max_{w \in \mathbb{R}^n, c \in \mathbb{R}} \left\{ \min_{i=1,N} \frac{y_i(w^T x_i + c)}{\|w\|_2} \right\}$$

# SUPPORT VECTOR MACHINE

Thay  $w$  bởi  $kw$  và  $c$  bởi  $kc$  thì siêu phẳng ( $H$ ) không đổi. Khoảng cách từ các điểm đến siêu phẳng cũng không đổi. Do đó, ta giả sử  $y_i(w^T x_i + c) = 1$  với những điểm gần siêu phẳng nhất.

Ta đưa về bài toán tối ưu có ràng buộc:

$$\max_{w \in \mathbb{R}^n, c \in \mathbb{R}} \frac{1}{\|w\|_2}$$

với điều kiện:  $y_i(w^T x_i + c) \geq 1, \forall i = \overline{1, N}$

Điều chỉnh hàm mục tiêu, ta đưa về bài toán tương đương sau:

$$\min_{w \in \mathbb{R}^n, c \in \mathbb{R}} \frac{1}{2} \|w\|_2^2$$

với điều kiện:  $1 - y_i(w^T x_i + c) \leq 0, \forall i = \overline{1, N}$

Hàm mục tiêu lồi chặt, tập ràng buộc cũng là tập lồi. Bài toán SVM có nghiệm tối ưu toàn cục duy nhất. Theo phương pháp toán tử Larange, với  $\lambda \geq 0$ , bài toán sẽ tối ưu hàm số:

$$\mathcal{L}(w, c, \lambda) = \frac{1}{2} \|w\|_2^2 + \sum_{i=1}^N \lambda_i (1 - y_i(w^T x_i + c))$$



**SOICT**

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY  
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

# SUPPORT VECTOR MACHINE

Với  $\lambda \geq 0$ , ta đặt ra:

$$g(\lambda) = \min_{w,c} \mathcal{L}(w, c, \lambda)$$

Cực tiểu  $\mathcal{L}(w, c, \lambda)$  ta xét hệ phương trình đạo hàm riêng:

$$\begin{cases} \frac{\partial \mathcal{L}(w, c, \lambda)}{\partial w} = w - \sum_{i=1}^N \lambda_i y_i x_i = 0 \\ \frac{\partial \mathcal{L}(w, c, \lambda)}{\partial c} = -\sum_{i=1}^N \lambda_i y_i = 0 \end{cases} \Leftrightarrow \begin{cases} w = \sum_{i=1}^N \lambda_i y_i x_i \\ \sum_{i=1}^N \lambda_i y_i = 0 \end{cases}$$

Với ma trận  $\mathcal{M} = (y_1 x_1, y_2 x_2, \dots, y_N x_N)$  và  $1 = (1, 1, \dots, 1)$  ta có thể viết lại:

$$g(\lambda) = -\frac{1}{2} \lambda^T \mathcal{M}^T \mathcal{M} \lambda + 1^T \lambda = -\frac{1}{2} \|\mathcal{M} \lambda\|_2^2 + 1^T \lambda$$

là hàm lõm, trên tập ràng buộc  $\lambda \geq 0$  và  $\sum_{i=1}^N \lambda_i y_i = 0$  cũng lồi (đa giác lồi) do đó có nghiệm cực đại toàn cục duy nhất.

Thuật toán	Ưu điểm	Nhược điểm	Mục tiêu
Logistic Regression	<ul style="list-style-type: none"><li>Đơn giản và dễ hiểu.</li><li>Tốt cho dữ liệu tuyến tính.</li><li>Dễ diễn giải mô hình.</li></ul>	<ul style="list-style-type: none"><li>Không phù hợp với dữ liệu phi tuyến.</li><li>Có thể bị ảnh hưởng bởi các biến không cần thiết.</li></ul>	<ul style="list-style-type: none"><li>Phân loại nhị phân</li></ul>
Decision Tree	<ul style="list-style-type: none"><li>Dễ diễn giải.</li><li>Không yêu cầu chuẩn hóa dữ liệu.</li><li>Tự động lựa chọn tính năng quan trọng.</li></ul>	<ul style="list-style-type: none"><li>Dễ bị quá khớp (overfitting).</li><li>Có thể không ổn định (các thay đổi nhỏ trong dữ liệu có thể dẫn đến cây khác nhau).</li></ul>	<ul style="list-style-type: none"><li>Phân loại và hồi quy</li></ul>
Random Forest	<ul style="list-style-type: none"><li>Giảm overfitting so với Decision Tree.</li><li>Xử lý dữ liệu lớn.</li><li>Tự động lựa chọn tính năng quan trọng</li></ul>	<ul style="list-style-type: none"><li>Khó diễn giải hơn Decision Tree.</li><li>Cần nhiều tài nguyên tính toán hơn.</li></ul>	<ul style="list-style-type: none"><li>Phân loại và hồi quy</li></ul>
Support Vector Machine (SVM)	<ul style="list-style-type: none"><li>Tốt cho dữ liệu chiều cao.</li><li>Phân cách lớp dữ liệu tốt với một ranh giới rõ ràng.</li></ul>	<ul style="list-style-type: none"><li>Không hiệu quả với dữ liệu lớn.</li><li>Khó diễn giải.</li><li>Cần chọn hạt nhân (kernel) phù hợp.</li></ul>	<ul style="list-style-type: none"><li>Phân loại nhị phân và đa lớp</li></ul>
K-Nearest Neighbors (KNN)	<ul style="list-style-type: none"><li>Đơn giản và dễ hiểu.</li><li>Không cần giả sử về phân phối dữ liệu.</li></ul>	<ul style="list-style-type: none"><li>Tốn kém về mặt tính toán với dữ liệu lớn.</li><li>Cần lựa chọn K phù hợp.</li></ul>	<ul style="list-style-type: none"><li>Phân loại và hồi quy, dữ liệu có cấu trúc</li></ul>
Naive Bayes	<ul style="list-style-type: none"><li>Đơn giản và nhanh chóng.</li><li>Tốt cho dữ liệu chiều cao và dữ liệu văn bản.</li></ul>	<ul style="list-style-type: none"><li>Giả sử các tính năng độc lập (điều này có thể không chính xác).</li></ul>	<ul style="list-style-type: none"><li>Phân loại, đặc biệt là phân loại văn bản</li></ul>
Gradient Boosting Machines (GBM)	<ul style="list-style-type: none"><li>Thường có hiệu suất cao.</li><li>Xử lý dữ liệu phi tuyến.</li><li>Tự động lựa chọn tính năng quan trọng.</li></ul>	<ul style="list-style-type: none"><li>Cần nhiều thời gian để huấn luyện.</li><li>Cần điều chỉnh nhiều tham số.</li></ul>	<ul style="list-style-type: none"><li>Phân loại và hồi quy</li></ul>
XGBoost	<ul style="list-style-type: none"><li>Phiên bản tối ưu của GBM.</li><li>Nhanh và hiệu quả.</li><li>Được thiết kế để tối ưu hóa tính toán.</li></ul>	<ul style="list-style-type: none"><li>Cần điều chỉnh nhiều tham số.</li></ul>	<ul style="list-style-type: none"><li>Phân loại và hồi quy</li></ul>
LightGBM	<ul style="list-style-type: none"><li>Tối ưu cho dữ liệu lớn.</li><li>Nhanh hơn nhiều so với GBM và XGBoost.</li></ul>	<ul style="list-style-type: none"><li>Cần điều chỉnh nhiều tham số.</li></ul>	<ul style="list-style-type: none"><li>Phân loại và hồi quy, dữ liệu lớn</li></ul>
AdaBoost	<ul style="list-style-type: none"><li>Đơn giản để cài đặt.</li><li>Kết hợp nhiều mô hình yếu thành một mô hình mạnh.</li></ul>	<ul style="list-style-type: none"><li>Dễ bị ảnh hưởng bởi nhiễu trong dữ liệu.</li><li>Có thể overfit nếu dữ liệu không cân bằng.</li></ul>	<ul style="list-style-type: none"><li>Phân loại</li></ul>
CatBoost	<ul style="list-style-type: none"><li>Tối ưu cho dữ liệu danh mục (categorical data).</li><li>Tự động điều chỉnh tham số</li></ul>	<ul style="list-style-type: none"><li>Tương đối mới so với các thuật toán khác.</li></ul>	<ul style="list-style-type: none"><li>Phân loại và hồi quy, dữ liệu có nhiều thuộc tính danh mục</li></ul>

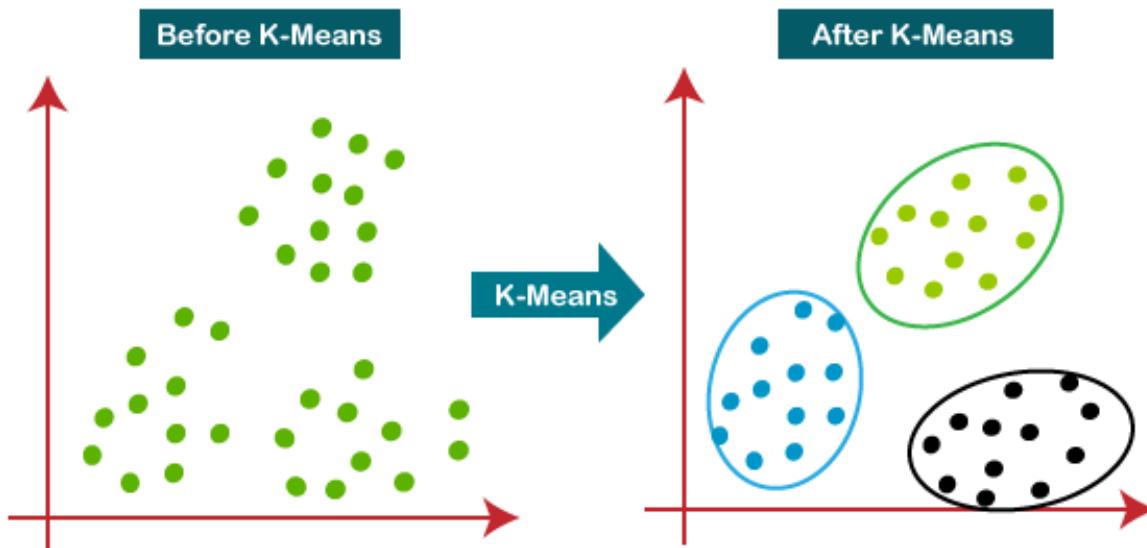
1. Làm quen với học máy
2. Học có giám sát: Các bài toán *Hồi quy* và *Phân loại*
3. Học không giám sát: Các bài toán *Phân cụm* và *Giảm chiều*
4. Cơ bản về Học sâu: Mạng thần kinh nhân tạo



**SOICT**

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY  
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

# PHÂN CỤM K-MEANS



**Bài toán:** Chúng ta không biết nhãn của từng điểm dữ liệu. Làm sao để phân dữ liệu vào các cụm sao cho dữ liệu trong mỗi cụm (cluster) có tính chất giống nhau?

- Có  $N$  điểm dữ liệu  $X = (x_1, x_2, \dots, x_N) \in \mathbb{R}^{n \times N}$  và  $C < N$  là số cụm mà ta muốn phân chia.
- Mục tiêu là xác định nhãn  $y_i$  của điểm  $x_i$  và tất cả  $C$  trọng tâm  $m_i$  của  $C$  cụm.
- Trong đó  $m_i \in \mathbb{R}^{n \times 1}$  và  $y_i = (y_{i1}, y_{i2}, \dots, y_{iC})$  chỉ cụm theo quy tắc: Nếu  $x_i$  thuộc cụm thứ  $k$  thì  $y_{ik} = 1$  và  $y_{ij} = 0, \forall j \neq k$ .
- Nghĩa là, với mỗi  $i$  ta có ràng buộc:

$$y_{ij} \in \{0,1\}, \forall j = 1, C \wedge \sum_{j=1}^C y_{ij} = 1$$

# THIẾT LẬP HÀM MẤT MÁT

Với trọng tâm  $m_k$  của cụm  $k$  và điểm  $x_i$  thuộc cụm này sẽ có sai số  $\|x_i - m_k\|_2^2$ , và ta mong muốn sai số này nhỏ nhất, nghĩa là càng gần trọng tâm thì điểm càng có khả năng thuộc cụm.

Vì  $x_i$  thuộc cụm  $k$  nên  $y_{ik} = 1$  và  $y_{ij} = 0, \forall j \neq k$ . Do đó:

$$\|x_i - m_k\|_2^2 = y_{ik}\|x_i - m_k\|_2^2 = \sum_{j=1}^C y_{ij}\|x_i - m_j\|_2^2$$

Sai số cho toàn bộ dữ liệu là:

$$\mathcal{L}(y, \mathcal{M}) = \sum_{i=1}^N \sum_{j=1}^C y_{ij}\|x_i - m_j\|_2^2$$

Trong đó  $y = (y_1, y_2, \dots, y_N)$  và  $\mathcal{M} = (m_1, m_2, \dots, m_C)$ .



**SOICT**

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY  
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

# TỐI ƯU HÀM MẤT MÁT

Vậy, ta phải giải bài toán tối ưu có ràng buộc:

$$(\mathcal{Y}^*, \mathcal{M}^*) = \arg \min_{\mathcal{Y}, \mathcal{M}} \sum_{i=1}^N \sum_{j=1}^C y_{ij} \|x_i - m_j\|_2^2$$

với điều kiện  $y_{ij} \in \{0,1\}, \forall i, j \wedge \sum_{j=1}^C y_{ij} = 1, \forall i$

*Giả sử tìm được các tâm*, bài toán tìm cụm  $\mathcal{Y}$  cho tất cả điểm có thể chia thành bài toán tìm vector cụm  $y_i$  cho một điểm xác định  $x_i$ , nghĩa là bài toán tối ưu:

$$y_i = \arg \min_{y_i} \sum_{j=1}^C y_{ij} \|x_i - m_j\|_2^2$$

với điều kiện  $y_{ij} \in \{0,1\}, \forall j = \overline{1, C} \wedge \sum_{j=1}^C y_{ij} = 1$

Nhưng  $x_i$  thuộc cụm  $k$  nên chỉ có  $y_{ik} = 1$ . Do đó, thực chất bài toán trên là:

$$k = \arg \min_k \|x_i - m_k\|_2^2$$

Như vậy,  $x_i$  thuộc cụm có trọng tâm gần nó nhất.



**SOICT**

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY  
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

# TỐI ƯU HÀM MẤT MÁT

Giả sử tìm được cụm cho từng điểm, ta sẽ tìm trọng tâm từng cụm, tức là giải bài toán sau:

$$m_j = \arg \min_{m_j} \sum_{j=1}^C y_{ij} \|x_i - m_j\|_2^2$$

Hàm  $l(m_j) = \sum_{j=1}^C y_{ij} \|x_i - m_j\|_2^2$  lồi, khả vi. Nghiệm cực tiểu địa phương cũng là cực tiểu toàn cục.  
Lấy đạo hàm ta có:

$$\frac{\partial l(m_j)}{\partial m_j} = 2 \sum_{j=1}^C y_{ij} (m_j - x_i); \frac{\partial l(m_j)}{\partial m_j} = 0 \Leftrightarrow m_j = \frac{\sum_{j=1}^C y_{ij} x_i}{\sum_{j=1}^C y_{ij}}$$

Thực tế  $\sum_{j=1}^C y_{ij} x_i$  là tổng các điểm dữ liệu thuộc cụm  $j$  còn  $\sum_{j=1}^C y_{ij}$  là số điểm thuộc cụm  $j$ .  
Vậy,  $m_j$  là trung bình cộng các điểm thuộc cụm  $j$ .



**SOICT**

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY  
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

# THUẬT TOÁN K-MEANS

**Đầu vào:** Dữ liệu  $X$  và số lượng cụm cần tìm  $C$ .

**Đầu ra:** Các center  $M$  và vector cụm cho từng điểm dữ liệu  $Y$ .

Các bước của thuật toán:

1. Chọn  $C$  điểm bất kỳ làm các trọng tâm ban đầu.
2. Phân mỗi điểm dữ liệu vào cụm có trọng tâm gần nó nhất.
3. Nếu việc gán dữ liệu vào từng cụm ở bước 2 không thay đổi so với vòng lặp trước nó thì ta dừng thuật toán.
4. Cập nhật trọng tâm cho từng cụm bằng cách lấy trung bình cộng của tất cả các điểm dữ liệu đã được gán vào cụm đó sau bước 2.
5. Quay lại bước 2.



**SOICT**

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY  
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

# NHẬN XÉT: HẠN CHẾ

Là một giải thuật đơn giản trong lớp bài toán Phân cụm, K-means clustering có nhiều hạn chế:

- Phải biết trước số cụm cần phân chia.
- Phụ thuộc vào các trọng tâm khởi tạo ban đầu.
- Các cụm cần có hình dạng lồi, rất khó để phân cụm có hình dạng đặc biệt và phức tạp.
- Các cụm cần có số lượng điểm tương đương nhau.
- Giải thuật có thể bị mắc kẹt ở tối ưu địa phương.
- Khi dữ liệu có số chiều lớn, khoảng cách giữa các điểm trở nên ít phân biệt, khiến việc phân cụm trở nên khó khăn.



**SOICT**

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY  
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Giải thuật	Ưu điểm	Hạn chế
K-means	<ul style="list-style-type: none"> <li>Đơn giản, dễ hiểu.</li> </ul>	<ul style="list-style-type: none"> <li>Phụ thuộc vào khởi tạo.</li> <li>Giả định về hình dạng và kích thước cụm.</li> <li>Yêu cầu xác định số lượng cụm trước.</li> </ul>
DBSCAN	<ul style="list-style-type: none"> <li>Không cần xác định số lượng cụm trước.</li> <li>Có khả năng phát hiện cụm có hình dạng bất kỳ.</li> <li>Xử lý nhiễu tốt.</li> </ul>	<ul style="list-style-type: none"> <li>Khó xử lý dữ liệu có mật độ biến đổi.</li> </ul>
Agglomerative Hierarchical Clustering	<ul style="list-style-type: none"> <li>Cung cấp cấu trúc phân cấp.</li> <li>Không cần xác định số lượng cụm trước.</li> </ul>	<ul style="list-style-type: none"> <li>Tốn nhiều tài nguyên với dữ liệu lớn.</li> </ul>
Gaussian Mixture Model (GMM)	<ul style="list-style-type: none"> <li>Mô hình hỗn hợp Gaussian cho phép biểu diễn cụm dạng ellipsoid.</li> <li>Cung cấp độ đo mềm cho việc thuộc về cụm.</li> </ul>	<ul style="list-style-type: none"> <li>Yêu cầu giả định về hình dạng cụm.</li> <li>Tính toán phức tạp.</li> </ul>
Spectral Clustering	<ul style="list-style-type: none"> <li>Phù hợp với dữ liệu có cấu trúc đồ thị.</li> <li>Có thể phát hiện cụm không lồi.</li> </ul>	<ul style="list-style-type: none"> <li>Tính toán phức tạp với dữ liệu lớn.</li> </ul>
Affinity Propagation	<ul style="list-style-type: none"> <li>Tự động xác định số lượng cụm.</li> <li>Không cần khởi tạo trung tâm cụm.</li> </ul>	<ul style="list-style-type: none"> <li>Tính toán phức tạp.</li> <li>Không hiệu quả với dữ liệu lớn.</li> </ul>
BIRCH	<ul style="list-style-type: none"> <li>Thiết kế cho dữ liệu lớn.</li> <li>Tạo ra một cây phân cấp nhỏ gọn.</li> </ul>	<ul style="list-style-type: none"> <li>Không hiệu quả với dữ liệu không gian cao.</li> </ul>
Mean Shift	<ul style="list-style-type: none"> <li>Không giả định về số lượng hoặc hình dạng cụm.</li> <li>Cố gắng tìm "đỉnh" của hàm mật độ.</li> </ul>	<ul style="list-style-type: none"> <li>Tính toán phức tạp.</li> <li>Lựa chọn bằng thông hợp lý là quan trọng.</li> </ul>