



# Tài liệu tham khảo

---

- Vũ Hữu Tiệp, *Machine Learning cơ bản*, 2018. Link download <https://github.com/tiepvupsu/ebookMLCB>
- Blog: [https:// machinelearningcoban.com](https://machinelearningcoban.com)
- Facebook Page: [https:// www.facebook.com/machinelearningbasicvn/](https://www.facebook.com/machinelearningbasicvn/)
- Facebook Group: [https:// www.facebook.com/ groups/machinelearningcoban/](https://www.facebook.com/groups/machinelearningcoban/)
- Interactive Learning: <https://fundaml.com>
- Bài giảng Học máy của PGS.TS Nguyễn Hữu Quỳnh
- Bài giảng Học máy của PGS.TS Nguyễn Thanh Tùng
- Bài giảng Học máy của TS Nguyễn Thị Kim Ngân

# Perceptron Learning Algorithm

**TS. Nguyễn Thị Kim Ngân**

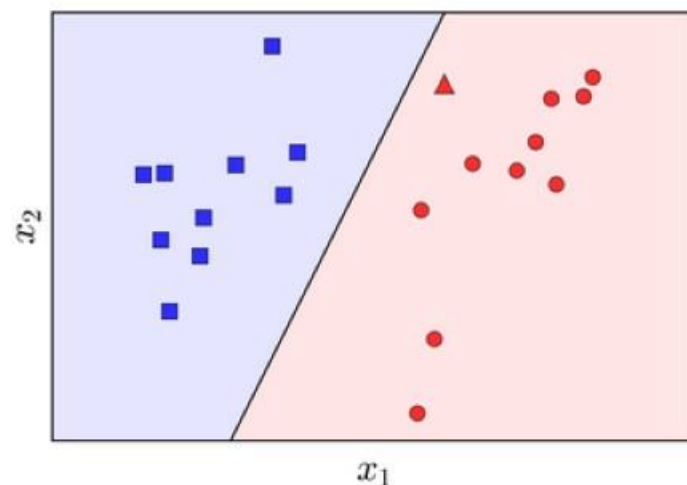
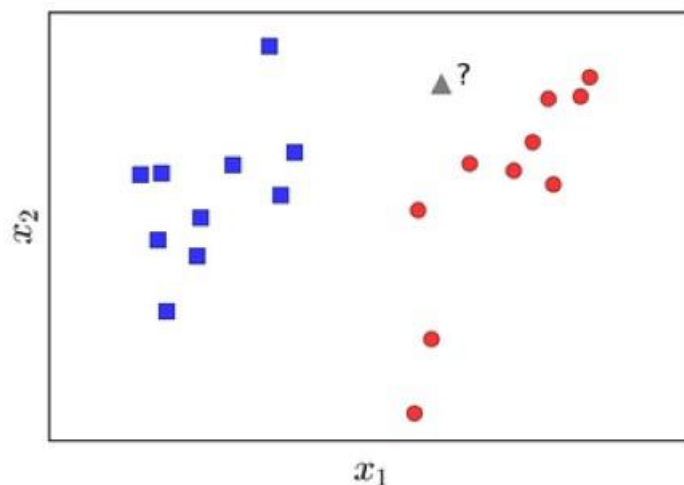
# Giới thiệu

Perceptron là một thuật toán Classification cho trường hợp đơn giản nhất:

chỉ có hai class (*binary classification*)

cũng chỉ hoạt động được trong một trường hợp rất cụ thể

Giả sử chúng ta có hai tập hợp dữ liệu đã được gán nhãn





# Bài toán Perceptron

---

*Cho hai class được gán nhãn, hãy tìm một đường phẳng sao cho:*

*toàn bộ các điểm thuộc class 1 nằm về 1 phía,*

*toàn bộ các điểm thuộc class 2 nằm về phía còn lại của đường phẳng đó.*

Nếu tồn tại một đường phẳng phân chia hai class thì ta gọi hai class đó là *linearly separable*.

Các thuật toán classification tạo ra các boundary là các đường phẳng được gọi chung là Linear Classifier.





# Thuật toán Perceptron (PLA)

---

ý tưởng cơ bản của PLA:

xuất phát từ một nghiệm dự đoán nào đó,  
qua mỗi vòng lặp, nghiệm sẽ được cập nhật tới một vị trí tốt hơn  
cập nhật dựa trên việc giảm giá trị của một hàm mất mát nào đó.

G/S ma trận chứa các điểm dữ liệu:

$$X = [x_1, x_2, \dots, x_N] \in \mathbb{R}^{d \times N}$$

Mỗi cột  $x_i \in \mathbb{R}^{d \times 1}$  là một điểm dữ liệu trong không gian  $d$  chiều

G/S các nhãn của mỗi điểm dữ liệu được lưu trong một véc tơ hàng

$$y = [y_1, y_2, \dots, y_N] \in \mathbb{R}^{1 \times N}$$

Với  $y_i = 1$  nếu  $x_i$  thuộc class 1 và  $y_i = -1$  nếu  $x_i$  thuộc class 2



# Thuật toán Perceptron (PLA)

---

G/S, tại một thời điểm, ta tìm được đường boundary là đường phẳng có phương trình:

$$f_w(x) = w_1x_1 + w_2x_2 + \dots + w_dx_d + w_0 = 0$$

Hay

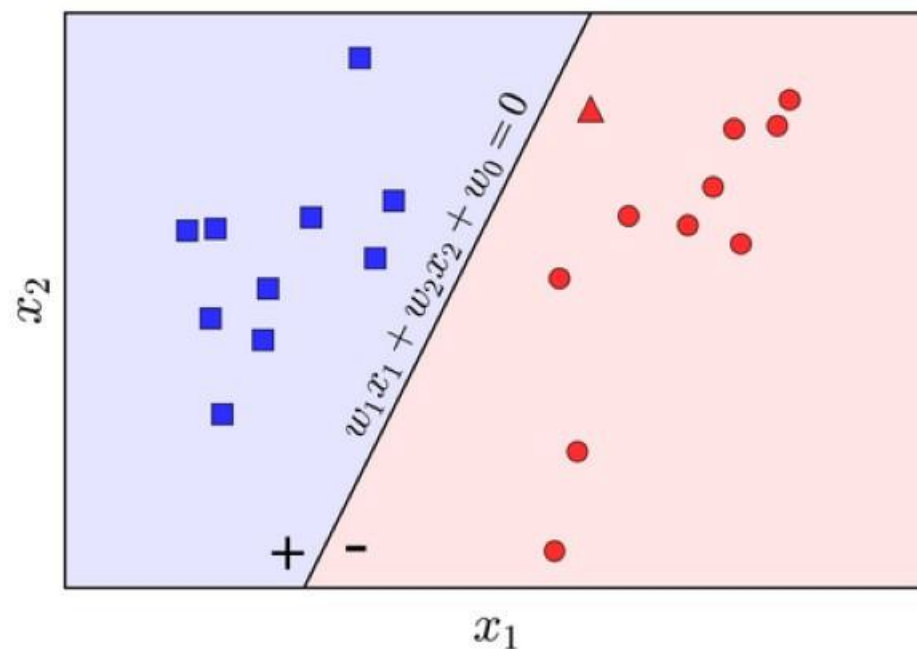
$$w^T x = 0$$

$$w = (w_0, w_1, w_2, \dots, w_d)$$

$$x = (1, x_1, x_2, \dots, x_d)$$

# Thuật toán Perceptron (PLA)

Để đơn giản, ta làm việc với trường hợp mỗi điểm dữ liệu chỉ có hai chiều ( $d=2$ )  
G/S đường thẳng  $w_1x_1 + w_2x_2 + w_0 = 0$  là nghiệm cần tìm như hình:





# Thuật toán Perceptron (PLA)

---

Như vậy, các điểm nằm về cùng một phía so với đường thẳng làm cho  $f_w(x)$  mang cùng dấu

Các dấu này tương ứng với nhãn  $y$  của mỗi lớp

Vậy, nếu tìm được  $w$  (nghiệm của bài toán Perceptron) và một điểm  $x$  chưa có nhãn, ta có thể xác định class của nó bởi:

$$label(x) = 1 \text{ nếu } w^T x \geq 0, \text{ ngược lại } -1$$

Hay:

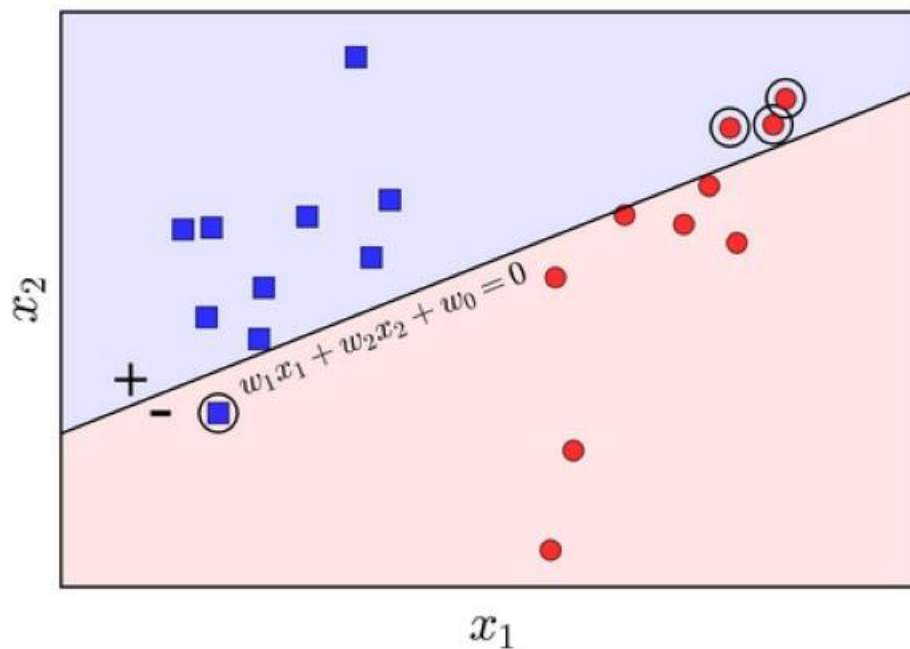
$$label(x) = \text{sgn}(w^T x)$$



# Xây dựng hàm mất mát

Tiếp theo, ta cần xây dựng hàm mất mát với tham số  $w$  bất kỳ.

Vẫn trong không gian hai chiều, giả sử đường thẳng  $w_1x_1 + w_2x_2 + w_0 = 0$  được cho như:





# Xây dựng hàm mất mát

---

Điều chúng ta mong muốn là không có điểm nào bị misclassified.

Hàm mất mát đơn giản nhất chúng ta nghĩ đến là hàm *đếm* số lượng các điểm bị misclassified và tìm cách tối thiểu hàm số này:

$$J_1(w) = \sum_{x_i \in M} (-y_i \operatorname{sgn}(w^T x_i))$$

M: số điểm phân lớp lỗi

Hạn chế quan trọng: hàm số này là rời rạc, không tính được đạo hàm theo  $w$  nên rất khó tối ưu.



# Xây dựng hàm mất mát

---

Xét hàm mất mát:

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

Khi một  $x_i$  (bị phân lớp sai) nằm càng xa boundary thì giá trị  $-y_i w^T x_i$  càng lớn

Giá trị nhỏ nhất của hàm mất mát này bằng 0 nếu không có điểm nào bị phân lớp sai

Hàm này trừng phạt nặng những điểm lấn sâu sang lãnh thổ của lớp khác



# Xây dựng hàm mất mát

---

Tại một thời điểm, nếu ta chỉ quan tâm đến điểm bị phân lớp sai, hàm  $J(w)$ :  
tính được đạo hàm

Ta có thể sử dụng giảm Gradient để tìm  $w$

Với một điểm  $x_i$  bị phân lớp sai, hàm mất mát trở thành:

$$J(w, x_i, y_i) = -y_i w^T x_i$$

Đạo hàm:

$$\nabla_w J(w, x_i, y_i) = -y_i x_i$$





# Xây dựng hàm mất mát

Quy tắc cập nhật:

$$w = w + \eta y_i x_i$$

$$w_{t+1} = w_t + \eta f'(w) = w_t + \eta y_i x_i$$

Ta có một quan sát nhỏ

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

Nếu  $y_i = 1$ :

vì  $x_i$  bị phân lớp sai nên  $w_t^T x_i < 0$

vì  $y_i = 1$  nên  $y_i \|x_i\|_2^2 = \|x_i\|_2^2 \geq 1$

do đó,  $w_{t+1}^T x_i > w_t^T x_i$

$w_{t+1}$  tiến về phía làm cho  $x_i$  được phân lớp đúng



# Xây dựng hàm mất mát

---

Đến đây, cảm nhận của chúng ta với thuật toán này là:

cứ chọn đường boundary đi.

Xét từng điểm một, nếu điểm đó bị misclassified thì tiến đường boundary về phía làm cho điểm đó được classified đúng.

Có thể thấy rằng, khi di chuyển đường boundary này:

các điểm trước đó được classified đúng có thể lại bị misclassified.

Mặc dù vậy, ta sẽ tìm được đường phẳng phân chia hai lớp, miễn là hai lớp đó là linearly separable.



# Thuật toán Perceptron

---

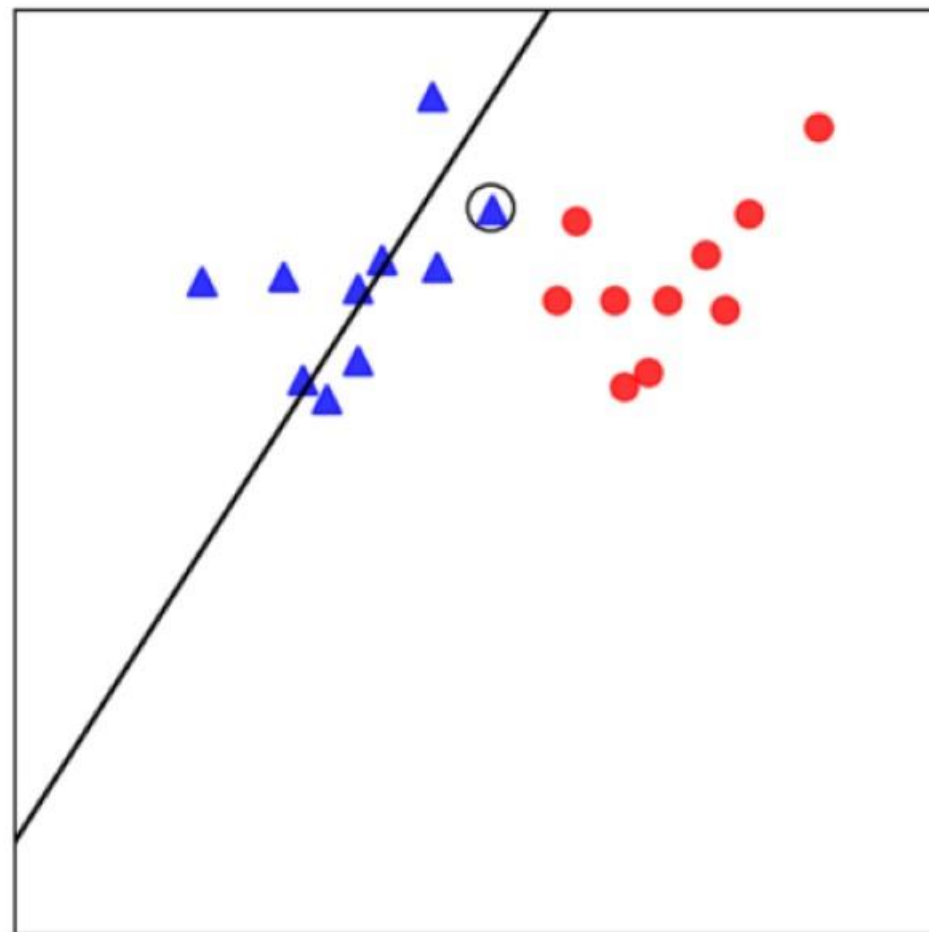
1. Tại thời điểm  $t = 0$ , chọn ngẫu nhiên một vector hệ số  $w_0$
2. Tại thời điểm  $t$ , nếu không có điểm dữ liệu nào bị phân lớp lỗi, dừng thuật toán.
3. Giả sử  $x_i$  là một điểm bị phân lớp lỗi. Cập nhật

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

4. Thay đổi  $t = t + 1$  rồi quay lại Bước 2

# Ví dụ trên Python

Ta sẽ tạo hai nhóm dữ liệu  
mỗi nhóm có 10 điểm,  
mỗi điểm dữ liệu có hai chiều

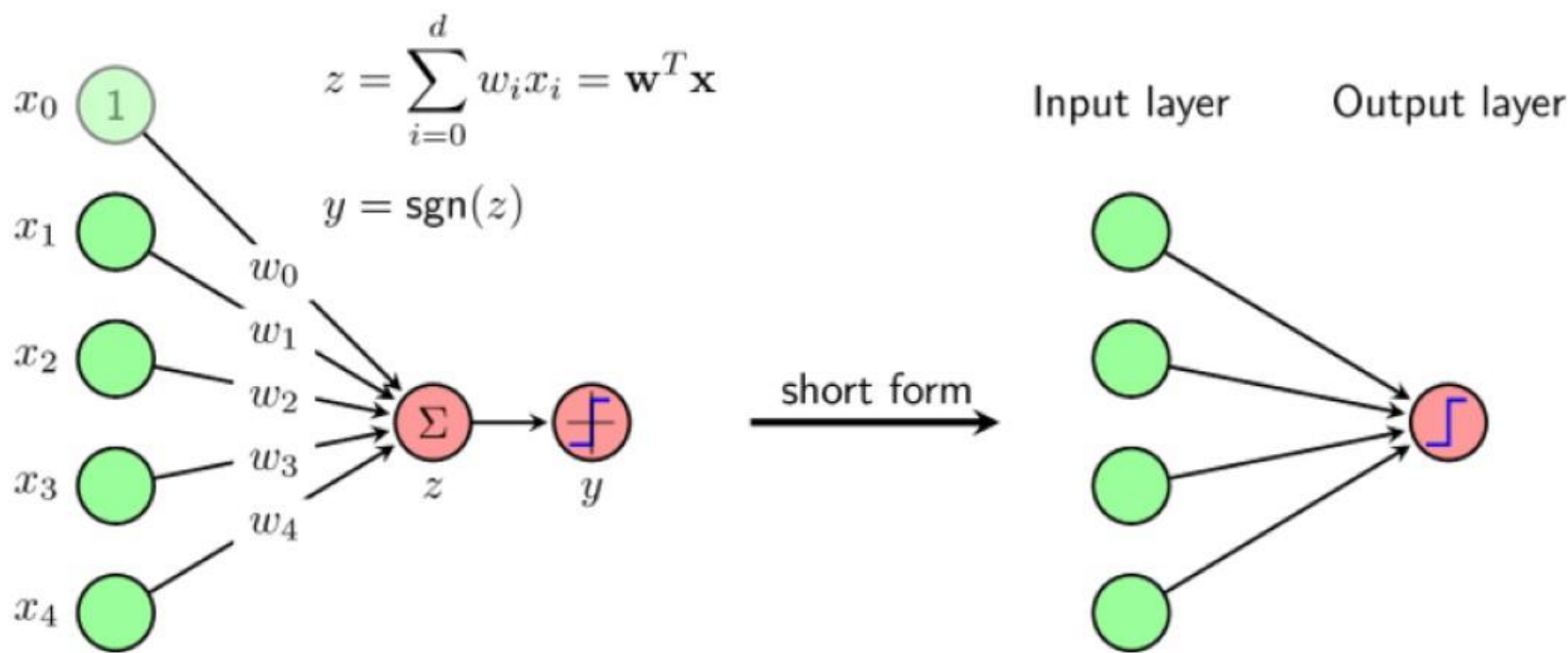


PLA: iter 13/18



# Mô hình Neural Network đầu tiên

Hàm số xác định class của Perceptron  $\text{label}(\mathbf{x}) = \text{sgn}(\mathbf{w}^T \mathbf{x})$  có thể được mô tả như hình vẽ (được gọi là network) dưới đây:





# Mô hình Neural Network đầu tiên

---

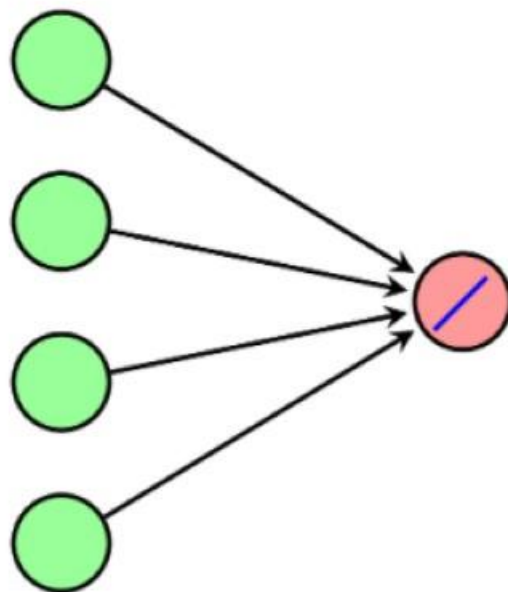
Trong thuật toán PLA: ta phải tìm các weights trên các mũi tên sao cho với mỗi  $x_i$  ở tập các điểm dữ liệu đã biết được đặt ở Input layer, output của network này trùng với nhãn  $y_i$  tương ứng

Hàm số  $y = \text{sgn}(z)$  còn được gọi là *activation function*. Đây chính là dạng đơn giản nhất của Neural Network.

# Mô hình Neural Network đầu tiên

Nếu ta thay *activation function* bởi  $y=z$ , ta sẽ có Neural Network mô tả thuật toán Linear Regression

Input layer      Output layer



# Mô hình Neural Network đầu tiên

Mô hình perceptron ở trên khá giống với một node nhỏ của dây thần kinh sinh học

