# ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
# TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

# TOÁN CHO KHOA HỌC MÁY TÍNH
# LỚP: CS115.O12
# TÌM HIỂU MÔ HÌNH PERCEPTRON

**Họ và tên: Trần Đình Khánh Đăng**

**MSSV: 22520195**

# MỤC LỤC

# CHƯƠNG 1: GIỚI THIỆU BÀI TOÁN PERCEPTRON

## I. Perceptron là gì:

Perceptron là một thuật toán Classification cho trường hợp đơn giản nhất: chỉ có hai class (lớp) (*bài toán với chỉ hai class được gọi là binary classification*) và cũng chỉ hoạt động được trong một trường hợp rất cụ thể. Tuy nhiên, nó là nền tảng cho một mảng lớn quan trọng của Machine Learning là Neural Networks và sau này là Deep Learning.

## II. Các ứng dụng thực tiễn:

Vì Perceptron là một thuật toán phân loại mạnh mẽ nên có thể được dùng trong các ứng dụng thực tiễn như:

- Trong lĩnh vực tài chính: Phân loại các giao dịch tài chính thành giao dịch hợp pháp và giao dịch gian lận.
- Trong lĩnh vực y tế: Phân loại các bệnh nhân thành bệnh nhân khỏe mạnh và bệnh nhân mắc bệnh.
- Trong lĩnh vực bán lẻ: Đề xuất các sản phẩm dựa trên lịch sử mua của khách hàng.
- Trong lĩnh vực giao thông: Sử dụng để điều khiển giao thông thông minh.

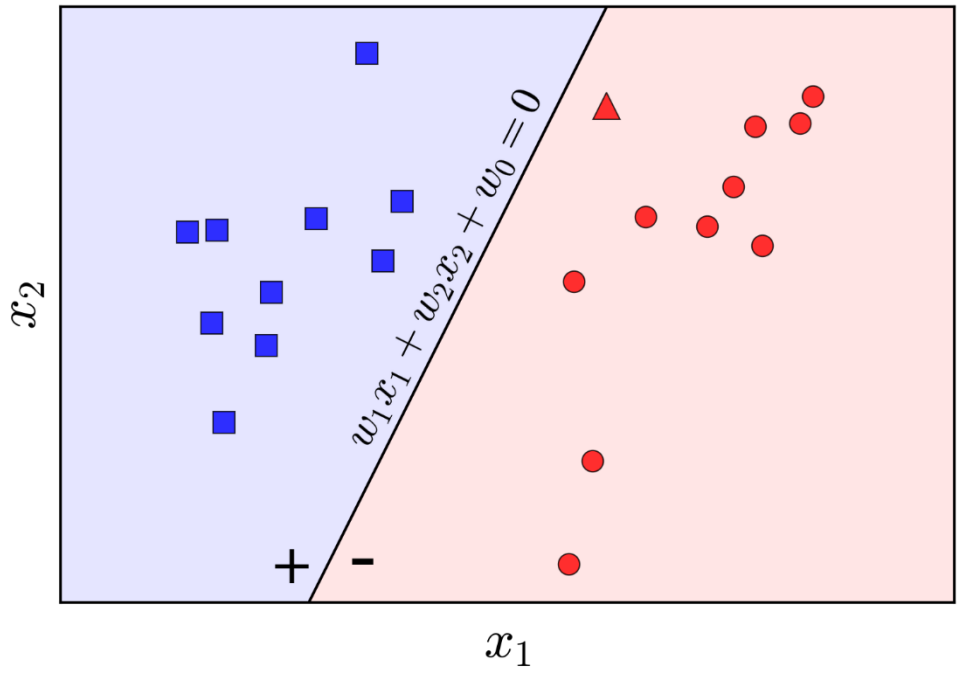
# CHƯƠNG 2: THUẬT TOÁN PERCEPTRON

Đối với thuật toán Perceptron, ý tưởng cơ bản là xuất phát từ một tập nghiệm dự đoán nào đó, qua mỗi vòng lặp, nghiệm sẽ được cập nhật tới vị trí tốt hơn. Việc cập nhật này dựa trên việc giảm giá trị của một hàm mất mát nào đó. Nhưng để có thể hiểu hơn về cách xây dựng hàm mất mát, ta cần xem xét qua các lưu ý sau:

Tại một thời điểm, giả sử ta tìm được boundary là đường phẳng có phương trình:

$$\boldsymbol{f_w(x)} = \sum_{i=0}^{d} \boldsymbol{x_i}\omega_i = \omega_1\boldsymbol{x_1} + \cdots + \omega_d\boldsymbol{x_d} + \omega_0 = \mathbf{w}^T\bar{\boldsymbol{x}} = 0$$

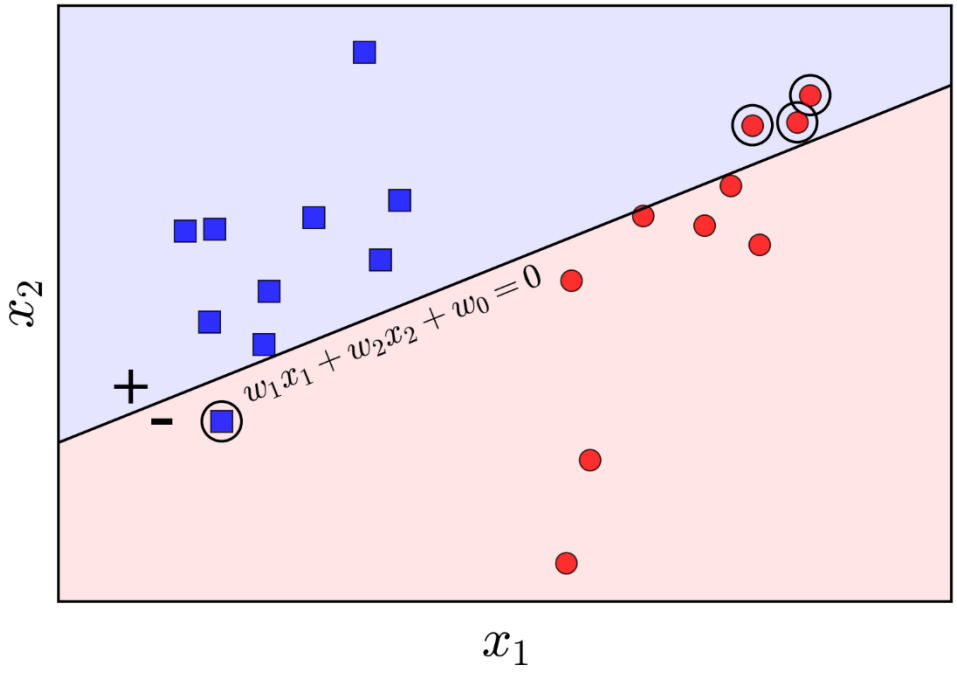
Để cho đơn giản, ta hãy cùng làm việc với trường hợp mỗi điểm dữ liệu có số chiều $d = 2$, giả sử đường thẳng $\omega_1\boldsymbol{x}_1 + \omega_2\boldsymbol{x}_2 + \omega_0 = 0$ chính là nghiệm cần tìm như hình 2 dưới đây:




*Hình 2: Phương trình đường thẳng boundary.*

# I. Xây dựng hàm mất mát:

Tiếp theo, chúng ta cần xây dựng hàm mất mát với tham số w bất kỳ. Vẫn trong không gian hai chiều, giả sử đường thẳng $\omega_1 \boldsymbol{x}_1 + \omega_2 \boldsymbol{x}_2 + \omega_0 = 0$ được cho như Hình 3 dưới đây:




*Hình 3: Đường thẳng bất kỳ và các điểm bị misclassified được khoanh tròn.*

Trong trường hợp này, các điểm được khoanh tròn là các điểm bị misclassified (phân lớp lỗi). 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 misclassied và tìm cách tối thiểu hàm số này:

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

trong đó $\boldsymbol{\mathcal{M}}$ là tập hợp các điểm bị misclassifed (*tập hợp này thay đổi theo* w). Với mỗi điểm $\boldsymbol{x_i} \in \boldsymbol{\mathcal{M}}$, vì điểm này bị misclassified nên $y_i$ và sgn($\mathbf{w}^T \boldsymbol{x}$) khác nhau, và vì thế $-y_i$sgn($\mathbf{w}^T \boldsymbol{x_i}$) = 1. Vậy $\boldsymbol{J}_1(\mathbf{w})$ chính là hàm *đếm* số lượng các

điểm bị misclassified. Khi hàm số này đạt giá trị nhỏ nhất bằng 0 thì ta không còn điểm nào bị misclassified.

Một điểm quan trọng, hàm số này là rời rạc, không tính được đạo hàm theo **w** nên rất khó tối ưu. Vậy nên chúng ta một hàm mất mát khác để việc tối ưu khả thi hơn.

Xét hàm mất mát sau:

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

Hàm $\boldsymbol{J}()$ khác một chút với hàm $\boldsymbol{J}_1()$ ở việc bỏ đi hàm sgn. Nhận xét rằng khi một điểm misclassified $\boldsymbol{x_i}$ nằm càng xa boundary thì giá trị $-y_i \mathbf{w}^T \boldsymbol{x_i}$ sẽ càng lớn, nghĩa là sự sai lệch càng lớn. Giá trị nhỏ nhất của hàm mất mát này cũng bằng 0 nếu không có điểm nào bị misclassifed. Hàm mất mát này cũng được cho là tốt hơn hàm $\boldsymbol{J}_1()$ vì nó *trừng phạt* rất nặng những điểm *lấn sâu sang lãnh thổ của class kia*. Trong khi đó, $\boldsymbol{J}_1()$ *trừng phạt* các điểm misclassified như nhau (đều = 1), bất kể chúng xa hay gần với đường biên giới.

Với *một* điểm dữ liệu xi bị misclassified, hàm mất mát trở thành:

$$\boldsymbol{J}(\boldsymbol{w}; \boldsymbol{x_i}; y_i) = -y_i \boldsymbol{w}^T \boldsymbol{x_i}$$

Đạo hàm tương ứng:

$$\nabla_{\boldsymbol{W}} \boldsymbol{J}(\boldsymbol{w}; \boldsymbol{x_i}; y_i) = -y_i \boldsymbol{x_i}$$

Vậy qui tắc cập nhật:

$$\mathbf{w} = \mathbf{w} + \eta y_i \boldsymbol{x_i}$$

với $\eta$ là learning rate được chọn bằng 1. Ta có một quy tắc cập nhật rất gọn là: $\mathbf{w}_{t+1} = \mathbf{w}_t + y_i \boldsymbol{x_i}$. Nói cách khác, với mỗi điểm $\boldsymbol{x_i}$ bị misclassifed, ta chỉ cần nhân điểm đó với nhãn $y_i$ của nó, lấy kết quả cộng vào **w** ta sẽ được **w** mới.

Ta có một quan sát nhỏ ở đây:

$$\begin{aligned} \mathbf{w}_{t+1}^T \boldsymbol{x_i} &= (\mathbf{w}_t + y_i \boldsymbol{x_i})^T \boldsymbol{x_i} \\ &= \mathbf{w}_t^T \boldsymbol{x_i} + y_i \left|\left| \boldsymbol{x_i} \right|\right|_2^2 \end{aligned}$$

Nếu $y_i = 1$, vì $\boldsymbol{x_i}$ bị misclassifed nên $\mathbf{w}_t^T \boldsymbol{x_i} < 0$. Cũng vì $y_i = 1$ nên $y_i \left|\left|\boldsymbol{x_i}\right|\right|_2^2 = \left|\left|\boldsymbol{x_i}\right|\right|_2^2 \geq 1$ (chú ý $\boldsymbol{x_0} = 1$), nghĩa là $\mathbf{w}_{t+1}^T \boldsymbol{x_i} > \mathbf{w}_t^T \boldsymbol{x_i}$. Lý giải bằng lời, $\mathbf{w}_{t+1}$ tiến về phía làm cho $\boldsymbol{x_i}$ được phân lớp đúng. Điều tương tự xảy ra nếu $y_i = -1$.

## II. Tóm tắt giải thuật:

1) Chọn ngẫu nhiên một vector hệ số **w** với các phần tử gần 0.
2) Duyệt ngẫu nhiên qua từng điểm dữ liệu $\boldsymbol{x_i}$:
   - Nếu $\boldsymbol{x_i}$ được phân lớp đúng, tức $\text{sng}(\mathbf{w}^T \boldsymbol{x_i}) = y_i$, chúng ta không cần làm gì.
   - Nếu $\boldsymbol{x_i}$ bị misclassified, cập nhật **w** theo công thức:

$$\mathbf{w} = \mathbf{w} + y_i \boldsymbol{x_i}$$

3) Kiểm tra xem có bao nhiêu điểm bị misclassifed. Nếu không còn điểm nào, dừng thuật toán. Nếu còn, quay lại bước 2.

## III. Chứng minh sự hội tụ:

Giả sử rằng $\mathbf{w}^*$ là một nghiệm của bài toán (ta có thể giả sử việc này được vì chúng ta đã có giả thiết hai class là linearly separable - tức tồn tại nghiệm). Có thể thấy rằng, với mọi $\alpha > 0$, nếu $\mathbf{w}^*$ là nghiệm, $\alpha \mathbf{w}^*$ cũng là nghiệm của bài toán. Xét dãy số không âm $u_\alpha(t) = \left|\left|\mathbf{w_t} - \alpha \mathbf{w}^*\right|\right|_2^2$. Với $\boldsymbol{x_i}$ là một điểm bị misclassified nếu dùng nghiệm $\mathbf{w_t}$ ta có:

$$\begin{aligned}
u_\alpha(t+1) &= \left|\left|\mathbf{w_{t+1}} - \boldsymbol{\alpha} \mathbf{w}'\right|\right|_2^2 \\
&= \left|\left|\mathbf{w_t} + \boldsymbol{x}_i y_i - \boldsymbol{\alpha} \mathbf{w}'\right|\right|_2^2 \\
&= \left|\left|\mathbf{w_t} - \boldsymbol{\alpha} \mathbf{w}'\right|\right|^2 + y_i^2 \left|\left|\boldsymbol{x}_i\right|\right|_2^2 + 2 y_i \boldsymbol{x}_i^T (\mathbf{w_t} - \boldsymbol{\alpha} \mathbf{w}') \\
&< u_\alpha(t) + \left|\left|\boldsymbol{x_i}\right|\right|_2^2 + 2 \boldsymbol{\alpha} \mathbf{w}' y_i \boldsymbol{x}_i^T
\end{aligned}$$

Dấu nhỏ hơn ở dòng cuối là vì $y_i^2 = 1$ và $2 y_i \boldsymbol{x}_i^T \mathbf{w_t} < 0$. Nếu ta đặt:

$$\beta^2 = \max_{i = 1,2,\dots,N} \left|\left|\boldsymbol{x}_i\right|\right|_2^2$$
$$\gamma = \min_{i = 1,2,\dots,N} 2y_i \boldsymbol{x}_i^{\boldsymbol{T}} \mathbf{w}_{\boldsymbol{t}}$$

và chọn $\alpha = \frac{\beta^2}{\gamma}$, ta có:

$$0 \leq u_{\boldsymbol{\alpha}}(t+1) < u_{\boldsymbol{\alpha}}(t) + \beta^2 - 2\boldsymbol{\alpha}\gamma = u_{\boldsymbol{\alpha}}(t) - \beta^2$$

Điều này nghĩa là: nếu luôn luôn có các điểm bị misclassified thì dãy $u_{\boldsymbol{\alpha}}(t)$ là dãy giảm, bị chặn dưới bởi 0, và phần tử sau kém phần tử trước ít nhất một lượng là $\beta^2 > 0$. Điều vô lý này chứng tỏ đến một lúc nào đó sẽ không còn điểm nào bị misclassified. Nói cách khác, thuật toán PLA hội tụ sau một số hữu hạn bước.