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

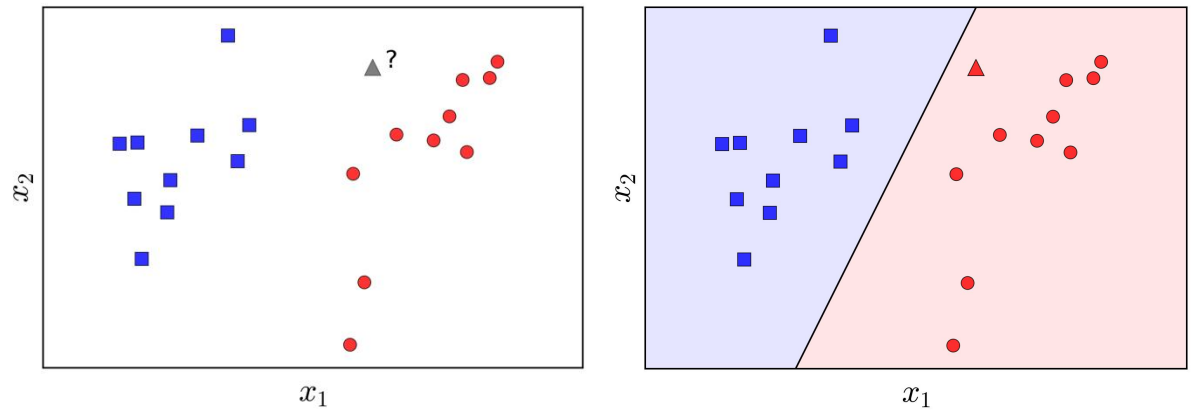
TOÁN CHO KHOA HỌC M ÁY TÍNH
LỚP: CS115.O12
GIỚI THIỆU MÔ HÌNH PERCEPTRON

Tên: Lê Minh Nhựt

MSSV: 222521060

I. Giới thiệu Bài toán Perceptron

Bài toán Perceptron với tên đầy đủ là Perceptron Learning Algorithm (PLA) là 1 bài toán phân loại nhị phân (Binary Classification), trong đó dữ liệu được chia thành 2 lớp. Mục tiêu của bài toán là xây dựng 1 thuật toán có thể phân loại chính xác các điểm dữ liệu mới dựa trên dữ liệu đã có. Tuy chỉ là 1 thuật toán Classification cho các trường hợp đơn giản nhất nhưng nó có vai trò là nền tảng quan trọng trong Neural Networks thuộc Machine Learning và sau này là Deep Learning.



Hình 1: Perceptron

Trong bài toán Perceptron, dữ liệu có thể được biểu diễn dưới dạng một tập hợp các điểm trên mặt phẳng. Mỗi điểm có 1 nhãn, chúng ta cần xác định cho điểm đó thuộc lớp nào. Trong hình Hình 1 bên trái, có 2 class gồm tập các điểm màu xanh và tập các điểm màu đỏ. Chúng ta sẽ xây dựng một 1 bộ phân lớp (classifier) để xác định màu nhãn của điểm dữ liệu.

Nói cách khác, để giải quyết bài toán ta cần đi tìm 1 biên giới (boundary) để phân cách giữa 2 lãnh thổ. Và đơn giản nhất chính là không gian 2 chiều khi boundary của nó là 1 đường thẳng. Nếu dữ liệu được phân tách tuyến tính, thuật toán Perceptron sẽ tìm ra đường thẳng có thể phân tách các điểm dữ liệu thành 2 lớp. Tuy nhiên, nếu dữ liệu không được phân tách tuyến tính, thuật toán có thể không tìm ra đường thẳng phân tách chính xác. Vì thế ta cần phải giả sử rằng có tồn tại đường thẳng có thể phân định lãnh thổ giữa

2 class (linear separable). Trong Hình 1 là ví dụ về việc có đường thẳng phân tách các điểm dữ liệu thành 2 phần lãnh thổ xanh và đỏ tương ứng.

Một số ví dụ về bài toán Perceptron:

- Phân loại email thành spam hoặc không phải spam.

- Phân loại hình ảnh người hoặc không phải người.

- Phân loại văn bản thành tích cực hoặc tiêu cực.

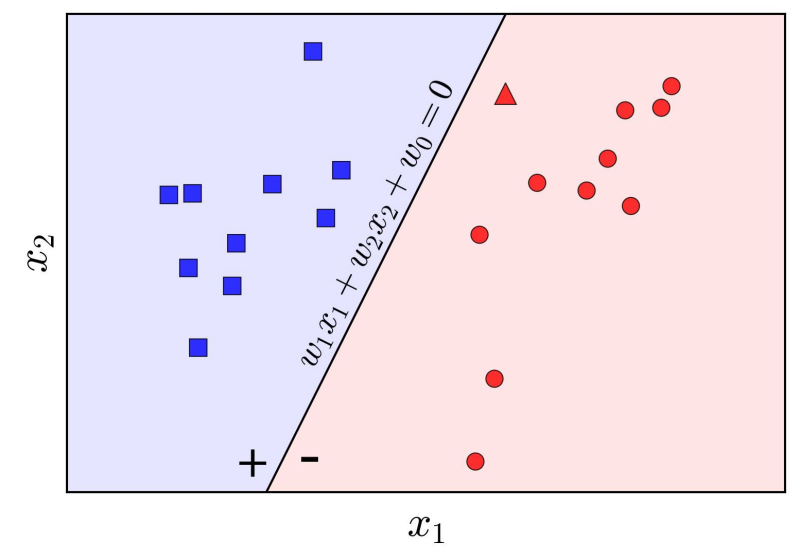
II. Thuật toán Perceptron (PLA).

Giả sử với tập ta các các điểm dữ liệu $\boldsymbol{x_i}$ tương với với các nhãn $\boldsymbol{y_i} \in \{-1, 1\}$.

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

$$\boldsymbol{f_w(x)} = \sum_{i=0}^{d} \boldsymbol{x_i}\omega_i = \omega_1\boldsymbol{x_1} + ... + \omega_d\boldsymbol{x_d} + \omega_0$$

Giả sử đường thẳng $\omega_1 x_1 + \omega_2 x_2 + \omega_0 = 0$ là nghiệm cần tìm như Hình 2 bên dưới:



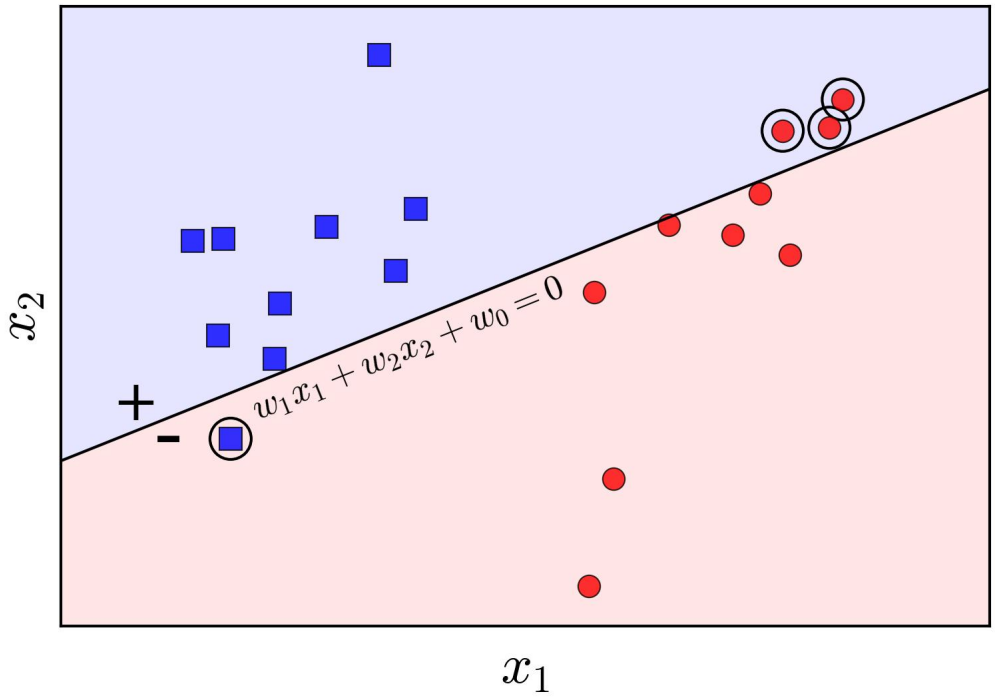
Hình 2: Phương trình đường thẳng boundary.

Ngắn gọn ta có:

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

1. Xây dựng hàm mất mát (loss function)

Trong không gian 2 chiều, giả sử đường thẳng $\omega_1 x_1 + \omega_2 x_2 + \omega_0 = 0$ với tham số **w** bất kì được cho như Hình 3 dưới đây:



Hình 3: Đường thẳng bất kì và các điểm mất mát

Để không có điểm nào bị misclassified (các điểm được khoanh tròn trên hình 3), chúng ta cần xây dựng hàm mát đơn giản nhất là hàm đếm số lượng các điểm bị misclassied và tìm cách tối thiểu hàm số này:

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

trong đó, mỗi điểm $\boldsymbol{x_i} \in \boldsymbol{\mathcal{M}}$ (là tập hợp các điểm bị misclassifed), vì điểm này bị misclassified nên y_i và sgn($\mathbf{w}^T\mathbf{x}$) khác nhau, và vì thế $-y_i$sgn($\mathbf{w}^T \boldsymbol{x_i}$) = 1. Vậy $\boldsymbol{J_1(w)}$ chính là hàm đếm số lượng các điểm bị misclassified. Và ta không còn điểm nào bị misclassified khi hàm đạt giá trị nhỏ nhất là 0. Nhưng do đây là hàm rời rạc, không tính được đạo hàm, nên cần tìm 1 hàm mất mát khác để thuận lợi cho việc tối ưu hơn. Thử xét hàm mất mát sau đây:

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

Hàm $\boldsymbol{J}()$ đã được bỏ đi hàm sgn() so với hàm $\boldsymbol{J_1}$. Giá trị $-y_i \mathbf{w}^T \boldsymbol{x}_i$ sẽ càng lớn khi 1 điểm misclassified nằm càng xa boundary (sai lệch sẽ càng lớn). Min của hàm này cũng bằng 0 nếu không có điểm nào bị misclassifed. Hàm $\boldsymbol{J}()$ được xem là tốt hơn vì chúng thể hiện rất rõ những điểm lấn sâu sang lãnh thổ class kia. Trong khi, các điểm bị hàm $\boldsymbol{J_1}()$ trừng phạt như nhau bất kể xa hay gần biên.

Tại một thời điểm, nếu chúng ta chỉ quan tâm tới các điểm bị misclassified thì hàm số $\boldsymbol{J}(\boldsymbol{w})$ khả vi (tính được đạo hàm), vậy chúng ta có thể sử dụng Gradient Descent hoặc Stochastic Gradient Descent (SGD) để tối ưu hàm mất mát này. Với ưu điểm của SGD cho các bài toán large-scale, chúng ta sẽ làm theo thuật toán này.

Với một điểm dữ liệu $\boldsymbol{x}_i$ bị misclassified, hàm mất mát trở thành:

$$\boldsymbol{J}(\boldsymbol{w}; \boldsymbol{x}_i; \boldsymbol{y}_i) = -y_i \boldsymbol{w}^T \boldsymbol{x}_i$$

Đạo hàm tương ứng:

$$\nabla_{\boldsymbol{W}} \boldsymbol{J}(\boldsymbol{w}; \boldsymbol{x}_i; \boldsymbol{y}_i) = -\mathrm{y_i} \boldsymbol{x}_i$$

Vậy quy tắc cập nhật là:

$$\mathbf{w} = \mathbf{w} + \eta \mathrm{y}_i \boldsymbol{x}_i$$

Với $\eta = 1$ (learning rate). Ta được:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \mathrm{y}_i \boldsymbol{x}_i.$$

Ta quan sát được rằng:

$$\mathbf{w}_{t+1}^T \boldsymbol{x}_i = (\boldsymbol{w}_t + \mathrm{y}_i \boldsymbol{x}_i)^T \boldsymbol{x}_i$$

$$= \mathbf{w}_{t+1}^T \boldsymbol{x}_i + \mathrm{y}_i ||\boldsymbol{x}_i||_2^2$$

Nếu $\mathrm{y}_i = 1$, vì $\boldsymbol{x}_i$ bị misclassifed nên $\boldsymbol{w}^T \boldsymbol{x}_i < 0$. Do $\mathrm{y}_i = 1$ nên $\mathrm{y}_i ||\boldsymbol{x}_i||_2^2 = ||\boldsymbol{x}_i||_2^2 \geq 1$ (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, $\boldsymbol{w}_{t+1}$ tiến về phía làm cho $\boldsymbol{x}_i$ được phân lớp đúng. Điều tương tự xảy ra nếu $\mathrm{y}_i = -1$.

Tóm lại, sau 1 số bước hữu hạn, thuật toán sẽ hội tụ, miễn là 2 lớp đó là linearly separable.

2. Tóm tắt giải thuật PLA

B1: Chọn ngẫu nhiên 1 vector hệ số **w** với các phần tử gần 0 (để dễ dàng quan sát).

B2: Duyệt ngẫu nhiên qua từng điểm dữ liệu $\boldsymbol{x}_i$:

- Nếu được phân lớp đúng, tức $sgn(\mathbf{w}^T\boldsymbol{x}_i) = y_i$, chúng ta không cần làm gì.
- Nếu bị misclassifed, cập nhật w theo công thức:

$$\mathbf{w} = \mathbf{w} + y_i\boldsymbol{x}_i$$

B3: Kiểm tra xem có bao nhiêu điểm bị misclassifed. Nếu không còn điểm nào, dừng thuật toán. Nếu còn, quay lại B2.

3. Chứng minh sự hội tụ.

Do ban đầu ta có giả thuyết 2 class tồn tại nghiệm (linear separable), nên chúng ta có thể giả định rằng $\boldsymbol{w}''$ là 1 nghiệm của bài toán. Ta thấy, $\forall\alpha > 0$, nếu $\boldsymbol{w}''$ là nghiệm thì $\alpha\boldsymbol{w}''$ cũng là nghiệm của bài toán. Ta tiếp tục xét dãy số không âm $u_\alpha(t) = ||\boldsymbol{w}_t - \alpha\boldsymbol{w}''||_2^2$. Trong đó, $\boldsymbol{x}_i$ là 1 điểm misclassified nếu nghiệm được dùng là $\boldsymbol{w}_t$, ta được biểu thức :

$$\begin{aligned}
&u_\alpha(t+1) = ||\boldsymbol{w}_{t+1} - \alpha\boldsymbol{w}''||_2^2 \\
&= ||\boldsymbol{w}_t + \boldsymbol{x}_i y_i - \alpha\boldsymbol{w}''||_2^2 \\
&= ||\boldsymbol{w}_t - \alpha\boldsymbol{w}''||^2 + y_i^2||\boldsymbol{x}_i||_2^2 + 2y_i\boldsymbol{x}_i^T(\boldsymbol{w}_t - \alpha\boldsymbol{w}'') \\
&< u_\alpha(t) + ||\boldsymbol{x}_i||_2^2 + 2\alpha\boldsymbol{w}''y_i\boldsymbol{x}_i^T
\end{aligned}$$

Ta được dấu "<" là do $y_i^2 = 1$ và $2y_i\boldsymbol{x}_i^T\boldsymbol{w}_t < 0$. Khi đặt:

$$\beta^2 = \max_{i=1,2,\dots,N} ||\boldsymbol{x}_i||_2^2$$

$$\gamma = \min_{i=1,2,\dots,N} y_i\boldsymbol{x}_i^T\boldsymbol{w}''$$

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

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

Nghĩa là: nếu luôn luôn tồn tại các điểm bị misclassified thì dãy $u_\alpha(t)$ sẽ trở thành dãy giảm, bị chặn dưới bởi 0, và phần tử sau kém hơn phần tử trước ít nhất một lượng có giá trị là $\beta^2 > 0$. Sự bất hợp lí này chứng tỏ đến một lúc nào đó sẽ không còn điểm nào bị misclassified. Mặc khác có thể nói rằng, thuật toán PLA sẽ hội tụ sau một số hữu hạn bước.