ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

SEMINAR BÀI GIẢNG MÔN HỌC

PHƯƠNG PHÁP TOÁN TRONG PHÂN TÍCH DỮ LIỆU THỊ GIÁC

MẠNG NEURAL HỌC SÂU

2020 – 2021

Thông tin nhóm:

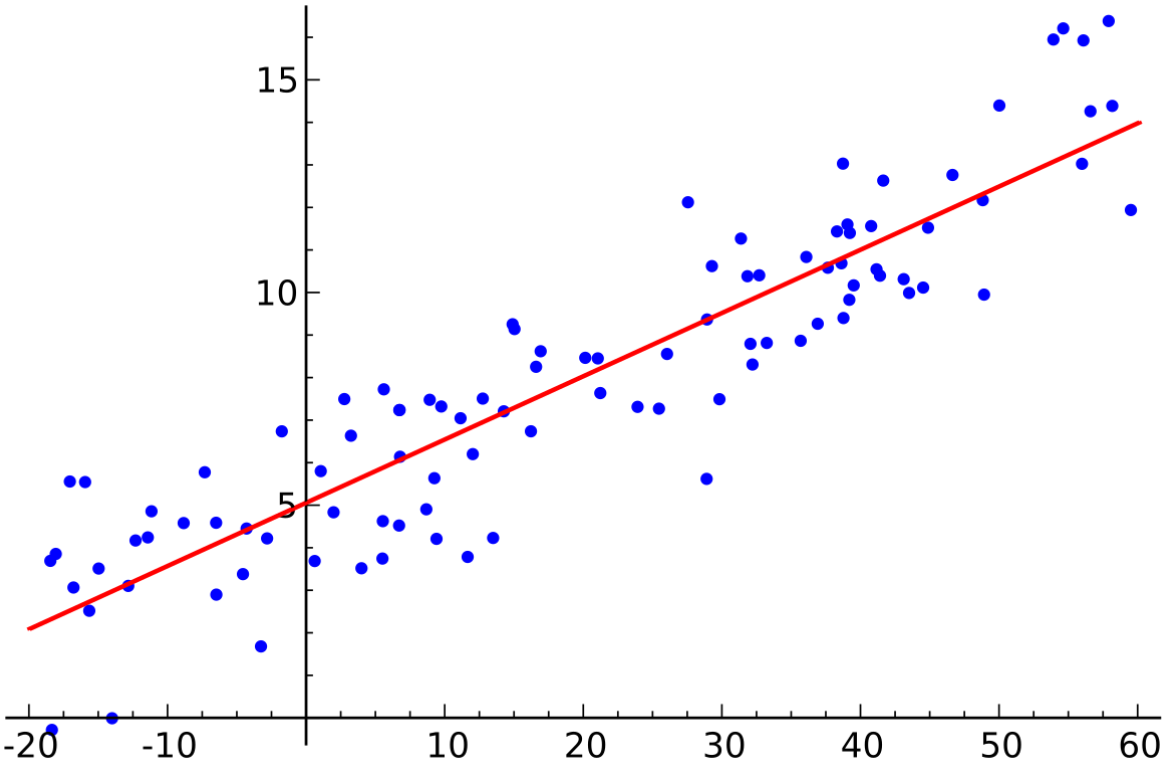
- Nguyễn Hoàng Thái Duy 1712019
- Bùi Văn Hợp 1712046
- Âu Dương Tấn Sang 1712145

TP Hồ Chí Minh, ngày 12 tháng 1 năm 2020

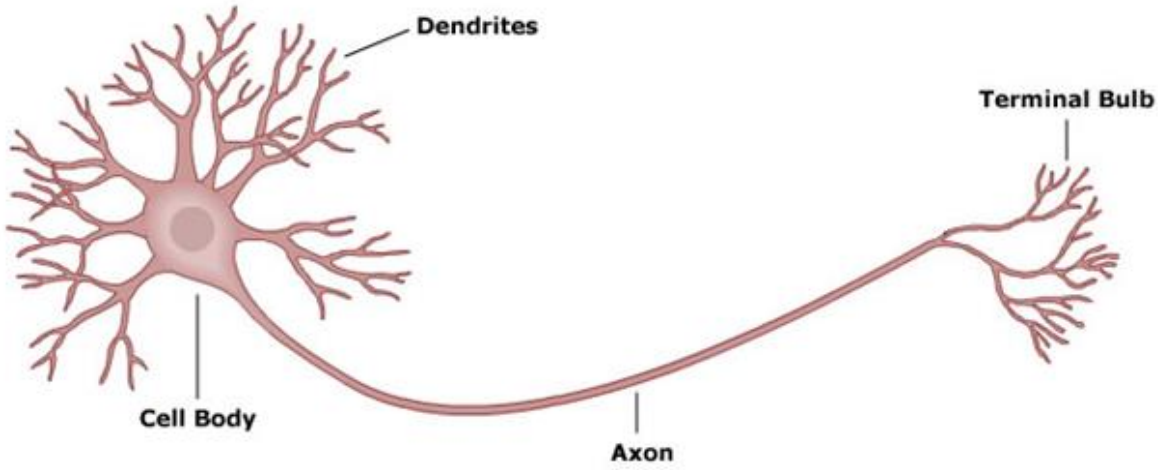
MẠNG NEURON HỌC SÂU

1. Single Layer Perceptron (SLP) :
    a. Nhắc lại về Linear Regression :

Hồi quy tuyến tính (Linear Regression) là một thuật toán hồi quy mà đầu ra là một hàm số tuyến tính, $y = f(x) = w^T x;\ với\ x \in R^{1 \times d}$;với d là số chiều đầu vào của vector x, và y là đầu ra của hàm tuyến tính, $y \in R$.



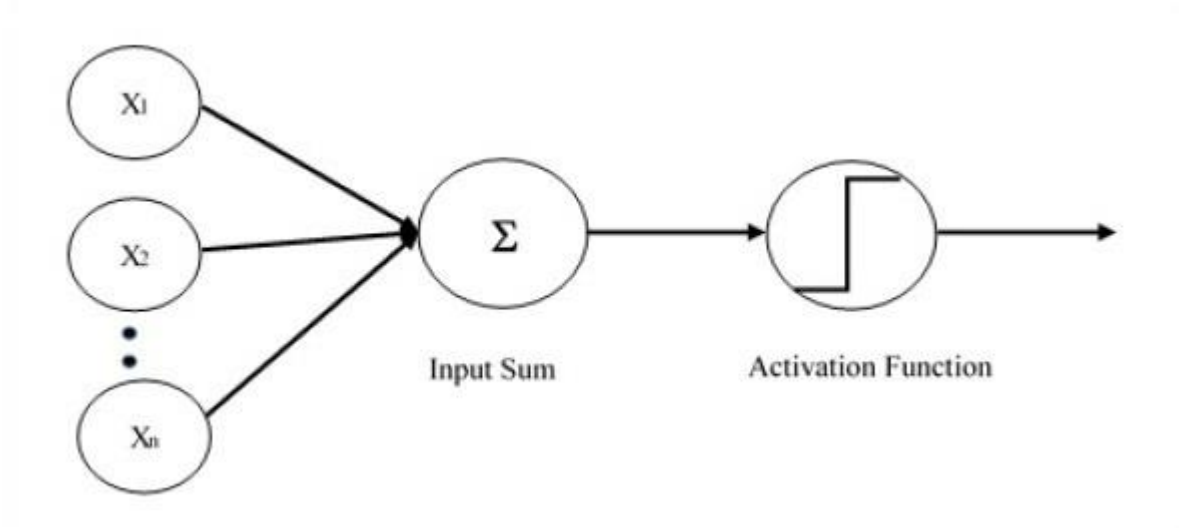
b. Ý tưởng về mạng thần kinh :



Người ta lấy ý tưởng từ mạng thần kinh của con người, nơi mà những Dendrite (các nhánh thần kinh) được tập hợp lại tại một Cell Body (nhân neuron) để truyền đi một tín hiệu thông tin.

Từ đây người ta áp dụng lí thuyết của Hồi quy tuyến tính vào một quá trình tính toán dữ liệu, và áp dụng một hàm kích hoạt (activation function), nhằm 2 mục đích sau:

- Đối với Hồi quy tuyến tính, kết quả là một số thực không bị giới hạn, chức năng của hàm kích hoạt sẽ chuẩn hoá lại giá trị đầu ra bằng một hàm số (có thể chuẩn hoá theo nhiều cách, đưa về dạng [-1, 1] hoặc [0, 1]).
- Giảm bớt chi phí tính toán về không gian lưu trữ, cũng như truyền tải dữ liệu, vì một kết quả đầu ra không bị giới hạn sẽ rất khó khăn trong việc lưu trữ, cũng như tính toán sau này.



Từ bài toán sơ khởi của Hồi quy tuyến tính và áp dụng một hàm kích hoạt (activation function), chúng ta có được 2 mô hình áp dụng:

- Mô hình Perceptron Learning Algorithm (PLA) là một thuật toán sử dụng hàm kích hoạt là hàm dấu (sign), $y = sign(f(x))$. Với hàm dấu, $sign(x) = 1 \, nếu \, x >= 0 \, và \, ngược \, lại \, sign(x) = -1 \, nếu \, x < 0$.
- Mô hình Logistic Regression sử dụng hàm kích hoạt là một hàm sigmoid (hay còn gọi là hàm logistics). Hàm sigmoid sẽ được trình bày kĩ hơn trong phần 3. Kết quả của Logistic Regression sẽ nằm trong khoảng [0, 1]. Điểm nổi bật ở mô hình này là nó còn mang ý nghĩa về mặt xác suất hơn so với PLA.

c. Tổng kết về Single Layer Perceptron (SLP) :

Chúng ta sẽ tổng kết về mô hình Single Layer Perceptron như sau:

Mô hình học:

Với một tập dữ liệu bao gồm N mẫu dữ liệu$(x^{(i)}, y^{(i)}), 1 <= i <= N$và một hàm kích hoạt $g(x)$.

Kết quả dự đoán của SLP có dạng: $\hat{y} = g(f(x)) = g(w^T x); \; x \in \{x^{(i)}\}$

Với một hàm lỗi dựa vào bài toán chúng ta cần làm:

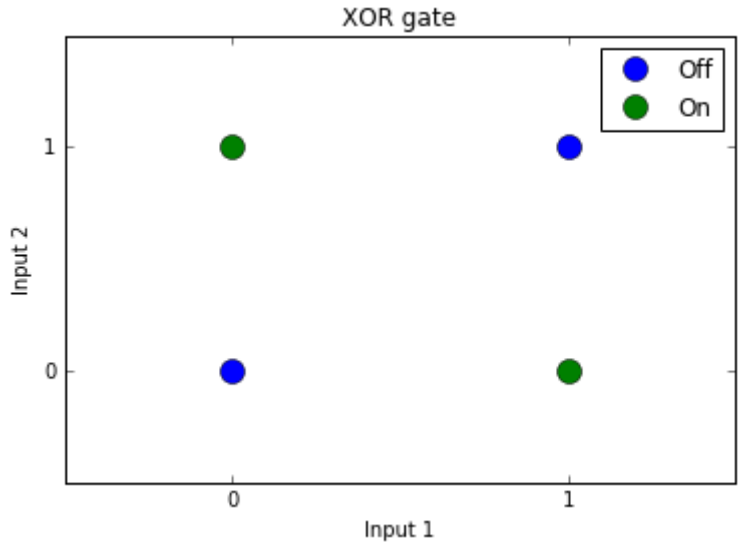
- Bài toán hồi quy, hàm lỗi thường được sử dụng là $\mathcal{L}(y, \hat{y}) = \frac{1}{2}(y - \hat{y})^2$.
- Bài toán phân lớp nhị phân thì hàm lỗi được sử dụng là hàm cross-entropy $\mathcal{L}(y, \hat{y}) = -[y \times log(\hat{y}) + (1 - y) \times log(1 - \hat{y})]$

2. Multiple Layer Perceptron (MLP) :

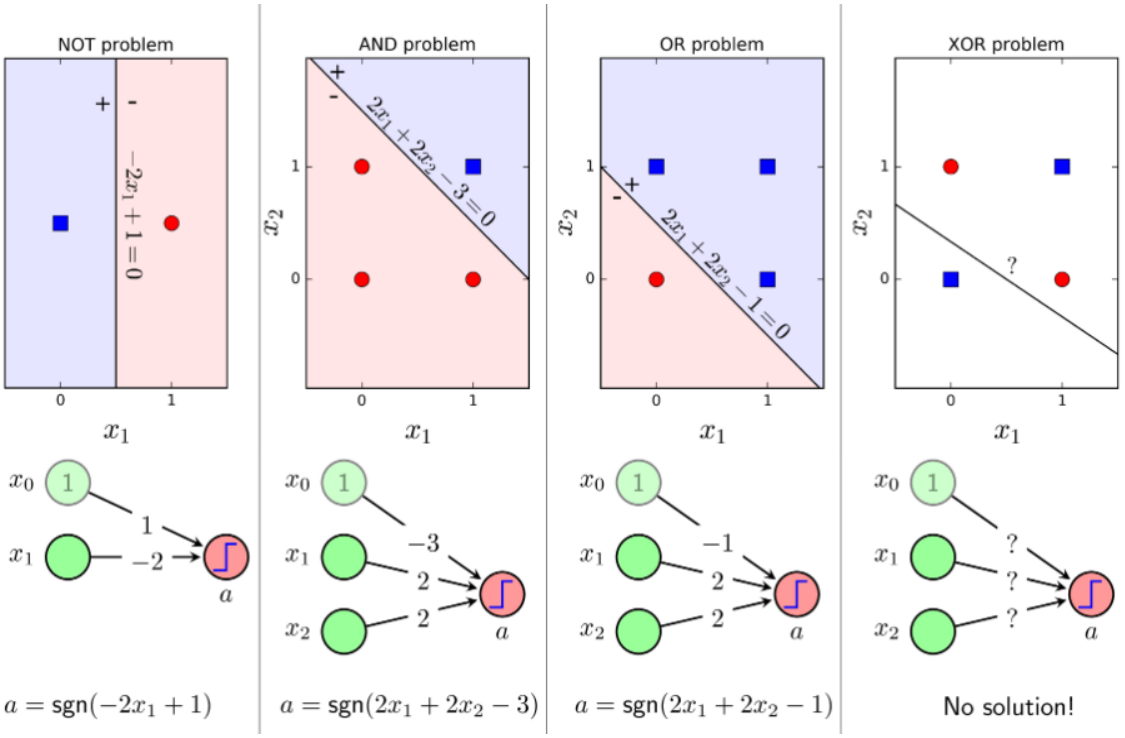
Từ bài toán SLP ở trên, chúng ta đặt ra một câu hỏi là liệu mô hình SLP có đủ khả năng học những dữ liệu phức tạp hay chưa ? Cùng nhau xem xét 1 ví dụ về khả năng học của mô hình SLP với môt dữ liệu từ hàm XOR ở bản như sau:

Data	AND	OR	XOR
(0, 0)	0	0	0
(0, 1)	0	1	1
(1, 0)	0	1	1
(1, 1)	1	1	0

Nói thêm về hàm XOR, đây là hàm chỉ nhận vào 1 cặp 2 giá trị, với mỗi giá trị trong thuộc tập hợp {0, 1}, nếu 2 giá trị giống nhau thì kết quả ra 0, khác nhau thì kết quả ra 1. Được biểu diễn trong miền không gian 2 chiều như sau:

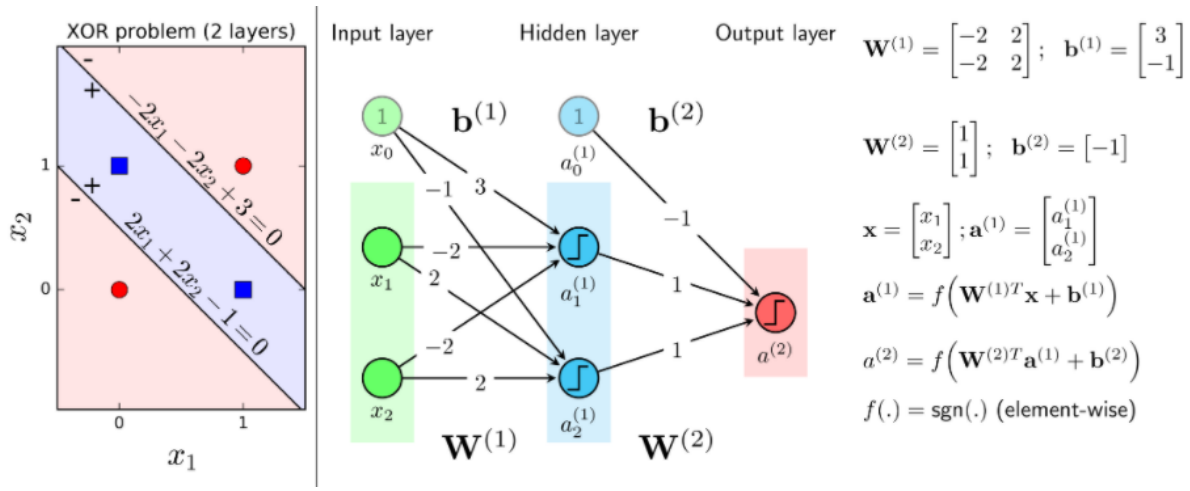


Ta giả sử với mô hình học SLP với hàm kích hoạt là hàm sign; tức là mô hình PLA ở trên. Vậy khả năng học của mô hình sẽ được thể hiện như sau:



- Ta thấy rằng PLA có thể học được những bộ dữ liệu khả tách tuyến tính (tức là chỉ sử dụng 1 đường thẳng để phân lớp dữ liệu), ví dụ như học hàm NOT, AND, OR như ở trên.
- Nhưng với hàm XOR đang xét thì không thể nào sử dụng 1 đường thẳng để phân tách 2 lớp của dữ liệu.

Từ đây ta thấy được điểm hạn chế của mô hình SLP. Vậy hướng giải quyết là sử dụng thêm 1 hoặc nhiều đường phân tách nữa để hỗ trợ cho việc phân lớp, và đó chính là lý do của mô hình Multiple Layer Perceptron ra đời.

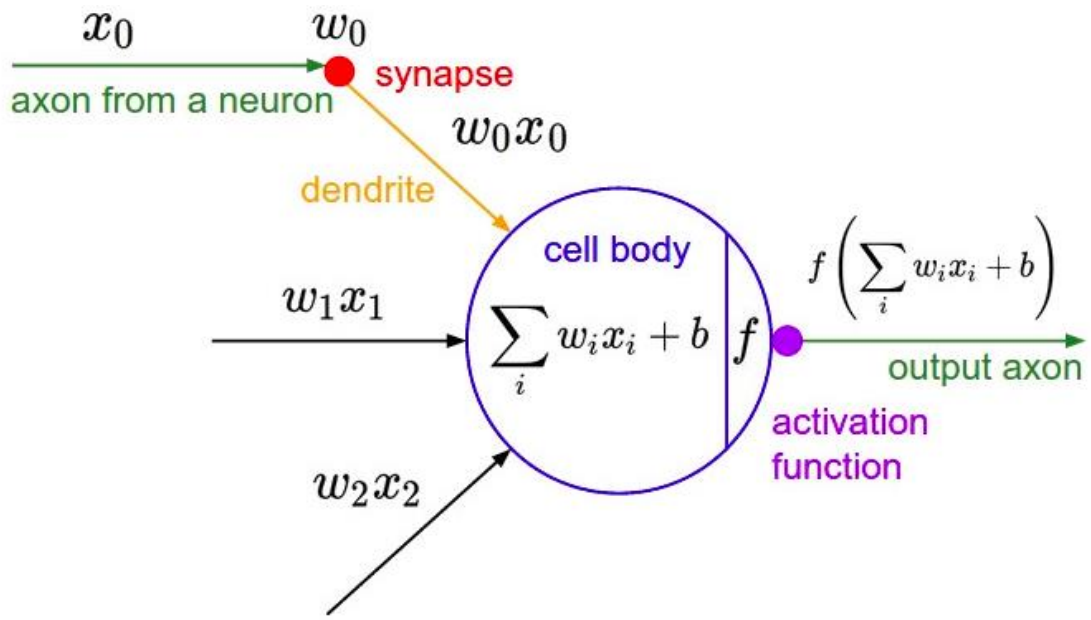


Mô hình MLP hiểu đơn giản là việc chồng nhiều lớp SLP liên tục lại với nhau trong đó: Lớp đầu tiên được gọi là lớp Input chính là vector đầu vào x, lớp cuối cùng chính là đầu ra Lớp Output chính là kết quả cuối cùng của mô hình; các lớp ở giữa được gọi là Lớp ẩn; kết quả của lớp trước chính là đầu vào của lớp sau và chúng đều được đưa qua một hàm kích hoạt.

Multiple Layer Perceptron hoạt động giống như việc ánh xạ dữ liệu input sang một không gian khác. Trước mỗi layer có một ma trận trọng số, ma trận này đại diện cho một ánh xạ tuyến tính từ đầu ra của layer trước sang layer hiện tại..

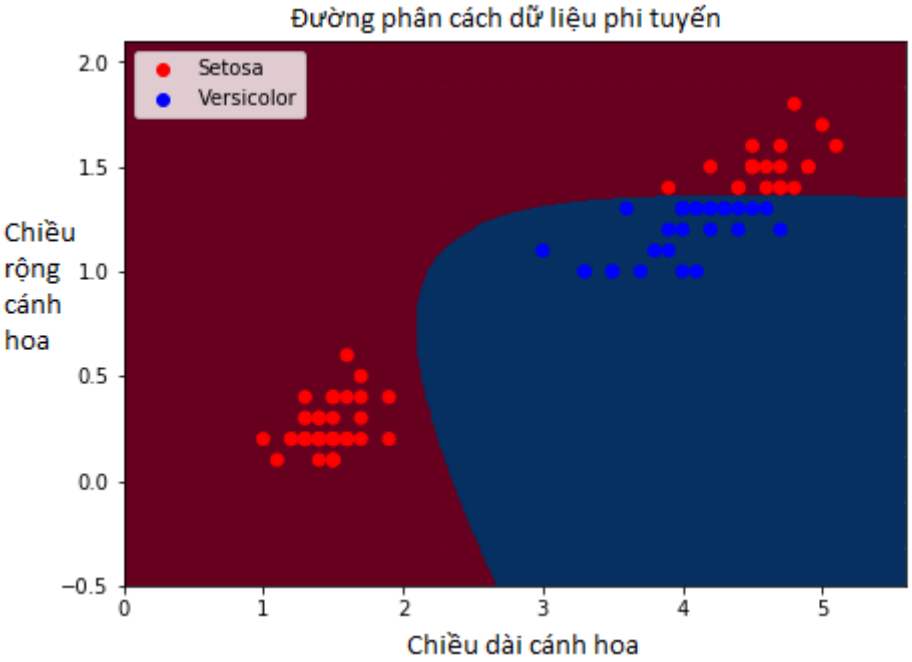
3. Activation Function :
    a. Ý nghĩa của một hàm kích hoạt phi tuyến :

    Hàm kích hoạt (activation function) mô phỏng tỷ lệ truyền xung qua axon của một neuron thần kinh



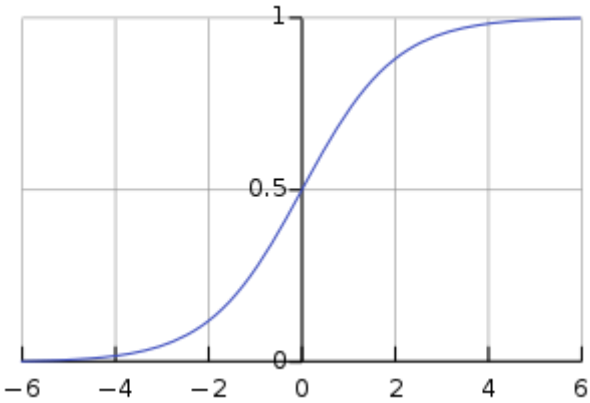
Về bản chất, mạng neuron là tổ hợp các phép tuyến tính. Bản chất của công thức trên là một tổ hợp tuyến tính giữa các giá trị input x và bộ trọng số w, do đó, khi áp dụng chúng với các dữ liệu mà có dạng tuyến tính, tức là dữ liệu mà ta có thể kẻ một đường thẳng để phân cách giữa chúng thì công thức tổ hợp trên đã đủ để giúp cho mô hình máy học có thể hoạt động tốt, chúng ta không cần tới hàm kích hoạt (activation function)

Trong thực tế, với nhiều bộ dữ liệu với phân bố dữ liệu phức tạp hơn, ta cần phải sử dụng **một hàm kích hoạt phi tuyến**, khi mà **một hàm kích hoạt tuyến** tính **không thể tìm ra một cách phân lớp đủ tốt**. Lúc này, hàm kích hoạt phi tuyến đóng vai trò như một người trung gian có nhiệm vụ chuyển đổi, nén hoặc chế biến output z từ tuyến tính trở thành phi tuyến tính, chúng ta thêm yếu tố "phi tuyến" này để làm cho mô hình học có khả năng "học" được những cấu trúc phức tạp hơn của dữ liệu.



b. Các hàm kích hoạt thông dụng :

- Hàm sigmoid có công thức là $\sigma(x) = \frac{1}{1+exp(-x)}$với $\sigma(x) = [0,1]$



Hàm sigmoid là một hàm kích hoạt mang ý nghĩa về mặt xác suất vì kết quả đầu ra sẽ nằm trong khoảng [0, 1]. Hàm này được sử dụng ở những mô hình máy học cũ, nhưng vì những bất lợi của nó như chi phí tính toán lớn, dễ khiến mạng học sâu rơi vào tình trạng vanishing gradient (tiêu biến gradient), nên hàm này chỉ còn được sử dụng ở lớp cuối cùng để đầu ra kết quả.

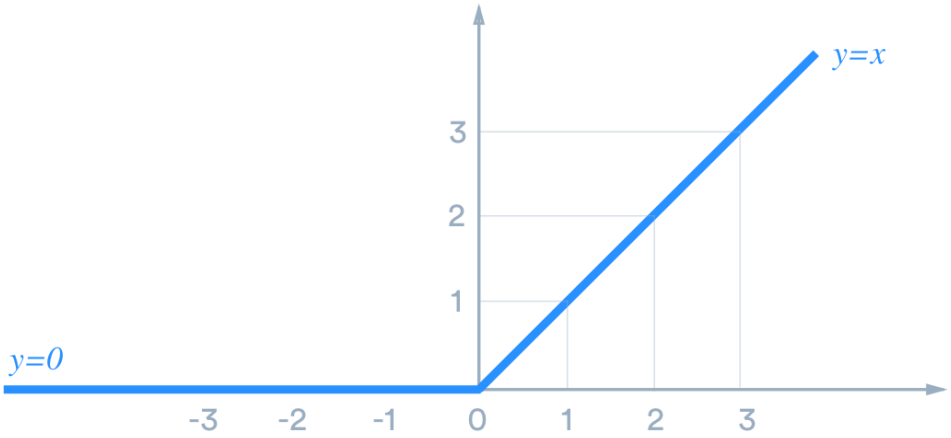
- Hàm Softmax là một hàm kích hoạt tổng quát hơn cho sigmoid, được sử dụng cho việc phân nhiều lớp.

Công thức : $softmax(x_i) = \frac{exp(x_i)}{\sum_{j=1}^{C} exp(x_j)}; \forall i = 1, 2, \dots C$

- Hàm tanh là một hàm kích hoạt khắc phục những nhược điểm của hàm sigmoid, đó là tạo ra kết quả trung tâm có thể bằng 0 so với hàm sigmoid là 0.5. Tuy nhiên nhược điểm như dễ gây vanishing gradient hay tốc độ tính toán lớn vẫn còn.

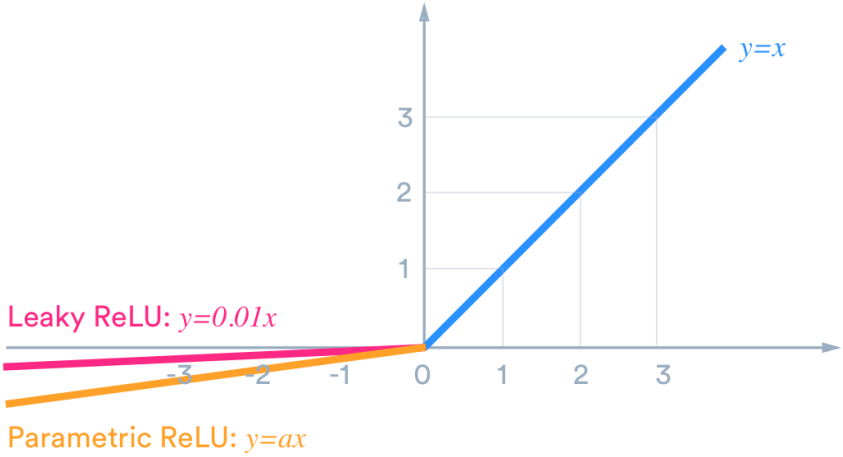
  Công thức : $tanh(x) = \frac{exp(x)}{1 + exp(x)}$
- Hàm ReLU. Đây là hàm kích hoạt được sử dụng rộng rãi nhất trong các mạng học sâu với tính đơn giản, không gây ra những hiệu ứng như các hàm sigmoid hay tanh. Tuy nhiên có 1 nhược điểm là hàm ReLU sẽ ngay lập tức triệt tiêu những tín hiệu nào nhỏ hơn 0, nó sẽ được khắc phục bởi hàm leaky ReLU ở sau.

  Công thức: $ReLU(x) = max(x, 0)$



- Hàm leaky ReLU là một hàm kích hoạt khắc phục được nhược điểm triệt tiêu hoàn toàn những tín hiệu âm của ReLU, cũng được sử dụng trong nhiều trường hợp.

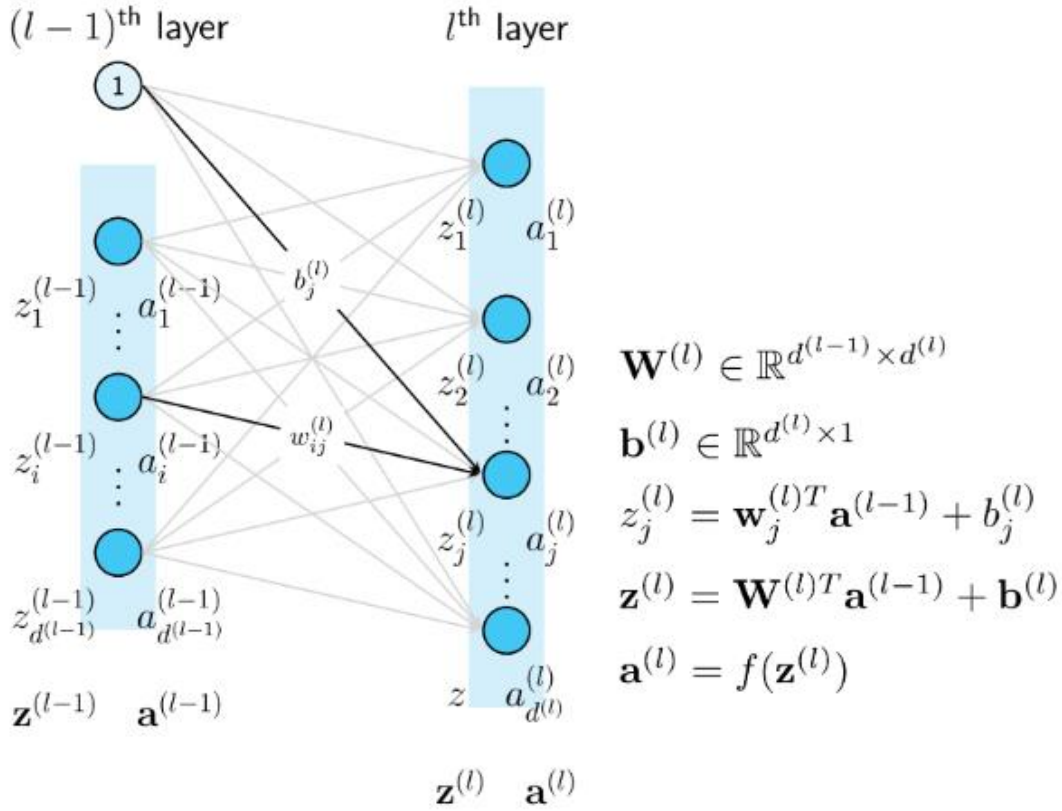
  Công thức: $leakyReLU(x) = max(x, 0.01 \times x)$hoặc tổng quát hơn là $leakyReLU(x, a) = max(x, a \times x);\ với\ a\ là\ hằng\ số\ rất\ nhỏ$



4. ForwardPropagation - BackPropagation:

   Kí hiệu:
   - Vector trước kích hoạt ở layer thứ l là $z^{(l)}$
   - Vector sau kích hoạt ở layer thứ $l$là $a^{(l)}$
   - Số chiều của $a^{(l)}$ và $z^{(l)}$là $d^{(l)}$

• Ma trận ánh xạ từ lớp $l-1$ sang lớp l là $W^{(l)}$
• Vector bias ngay sau ma trận ánh xạ là $b^{(l)}$
• Hàm kích hoạt là f(x) = [f(x0), f(x1), f(x2),...] $f(x) = [f(x_0), f(x_1), f(x_2), \dots]$



Với giai đoạn Forward Propagation hay là "Lan truyền xuôi", công thức tính toán như hình bên trên, hay nói bằng lời:

• Tín hiệu trước kích hoạt, $z^{(l)}$, được tính: $z_i^{(l)} = \Sigma(W_{i,j}^{(l)} a_j^{(l-1)}) + b_i^{(l)}$
• Tín hiệu sau kích hoạt, $a^{(l)}$, được tính: $a_i^{(l)} = f(z_i^{(l)})$

Để tìm được các W và b, phương pháp phổ biến nhất chính là **tối ưu bằng gradient descent.**

Để cập nhật $W_{i,j}^{(l)}$ ở layer thứ l bất kì, ta tính đạo hàm J theo $W_{i,j}^{(l)}$:

$$\frac{\partial J}{\partial W_{i,j}^{(l)}} = \frac{\partial z_i^{(l)}}{\partial W_{i,j}^{(l)}} \times \frac{\partial a_i^{(l)}}{\partial z_i^{(l)}} \times \frac{\partial J}{\partial a_i^{(l)}}$$

$$= a_j^{(l-1)} \times f'(z_i^{(l)}) \times \frac{\partial J}{\partial a_i^{(l)}} \quad (1)$$

(quy tắc chuỗi)

Nhân tử cuối cùng trong công thức trên được tính bằng quy tắc chuỗi, sử dụng đạo hàm của layer phía sau layer hiện tại:

$$\frac{\partial J}{\partial a_i^{(l)}} = \Sigma\left(\frac{\partial z_k^{(l+1)}}{\partial a_i^{(l)}} \times \frac{\partial J}{\partial z_k^{(l+1)}}\right)$$

(do $a_i^{(l)}$ dùng để tính tất cả $z_k^{(l+1)}$ ở layer phía sau phải lấy đạo hàm tất cả rồi cộng lại)

Thay vào (1):

$$\frac{\partial J}{\partial W_{i,j}^{(l)}} = a_j^{(l-1)} \times f'(z_i^{(l)}) \times \Sigma\left(\frac{\partial z_k^{(l+1)}}{\partial a_i^{(l)}} \times \frac{\partial J}{\partial z_k^{(l+1)}}\right) \quad (2)$$

Kiểm tra lại số chiều, hoàn toàn phù hợp (tất cả đều là số vô hướng).

Vậy ta chỉ cần cập nhật:

$$W_{i,j}^{(l)} \leftarrow W_{i,j}^{(l)} - \eta . \frac{\partial J}{\partial W_{i,j}^{(l)}}$$

(η là hệ số học)

Từ (1) và (2) có thể thấy, $\frac{\partial J}{\partial z_i^{(l)}}$ được tính dựa vào các $\frac{\partial J}{\partial z_k^{(l+1)}}$. Do đó, việc tính đạo hàm và cập nhật được thực hiện từ layer cuối cùng đến layer đầu tiên. Với layer cuối cùng, ta dễ dàng tính được $\frac{\partial J}{\partial z_k^{(L-1)}}$ nhờ vào Ground Truth của dữ liệu (viết hàm lỗi $\hat{y} - y$, rồi đạo hàm chuỗi). Sau khi tính toán xong, ta lưu giá trị này lại để tận dụng vào việc đạo hàm ở layer phía trước: $\frac{\partial J}{\partial z_i^{(l)}} = f'(z_i^{(l)}) \times \Sigma\left(\frac{\partial z_k^{(l+1)}}{\partial a_i^{(l)}} \times \frac{\partial J}{\partial z_k^{(l+1)}}\right)$, tiết kiệm được rất nhiều tài nguyên tính toán.

Phương pháp này được gọi là BackPropagation (tạm dịch: lan truyền ngược). Tham số b cũng có thể được thực hiện tương tự (thay vì đạo hàm theo $W_{i,j}^{(l)}$ thì đạo hàm theo $b_i^{(l)}$).

5. Gradient Descent :

   Xét một hàm số $f: R^d \to R$với tập xác định D

   - Điểm $x^* \in D$được gọi là *cực tiểu toàn cục (*tương ứng *với cực đại toàn cục)* nếu $f(x) \geq f(x^*)$(tương ứng với cực đại toàn cục là $f(x) \leq f(x^*)$.
   - Điểm $x^* \in D$được gọi là *cực tiểu địa phương* (tương ứng *cực đại địa phương*) nếu tồn tại $\varepsilon > 0$sao cho $f(x) \geq f(x^*)$(tương ứng với cực đại địa phương là $f(x) \leq f(x^*)$) với mọi x nằm trong lân cận $\mathcal{V}(x, x^*) = \{x: x \in D, d(x, x^*) \leq \varepsilon\}$

   Trong các bài toán học sâu nói riêng và các bài toán tối ưu nói chung, chúng ta thường xuyên phải tìm các cực tiểu toàn cục của một hàm số. Nếu chỉ xét reieng cách àm khả vi, việc phải phương trình đạo hàm bằng 0 có thể phức tạp hoặc có vô số nghiệm.

   Với phương pháp nghiệm số, chúng ta có thể áp dụng một phương pháp là Phương pháp Newton-Raphson.

   Với phương pháp tối ưu xấp xỉ, chúng ta có thể sử dụng phương pháp Gradient Descent được dùng để tìm các cực tiểu địa phương.

   a. Phương pháp tìm nghiệm số Newton-Raphson :

      Gọi các nghiệm x của phương trình $f(x) = 0$, ta cần tìm những nghiệm xấp xỉ với một nghiệm $x_0$ban đầu. Chúng ta sẽ liên tục cập nhật nghiệm x_{n+1} dựa vào nghiệm trước đó x_n.

$$x_{n+1} = x_n - \frac{f(x_n)}{f\prime(x_n)}$$

      Về mặt hình học, $(x_{n+1}, 0)$ là điểm giao giữa trục x và tiếp tuyến của đồ thị của f tại $(x_n, f(x_n))$.

   b. Phương pháp Gradient Descent :

      So với phương pháp Newton-Raphson ở trên, phương pháp Gradient Descent sẽ tìm những cực tiểu địa phương với công thức đơn giản hơn:

$$x_{n+1} = x_n - \eta f'(x_n)$$

      Trong đó ηlà một số dương biểu thị cho tốc độ học (learning rate). Dấu - biểu hiện cho việc đi ngược lại đạo hàm (gradient) của biến. Tên gọi gradient descent xuất phát từ ý nghĩa này. Trong một số trường hợp, phương pháp Gradient Descent có thể không chính xác, nhưng một số biến thể của Gradient Descent giúp ta cải thiện việc tìm các cực tiểu địa phương tốt hơn và trong thực tế đây là phương pháp tìm kiếm nghiệm cho bài toán tối ưu tốt nhất.