Phương pháp số trong Khoa học dữ liệu

T.S. Nguyễn Thị Hoài Thương

Trường Đại học Khoa học tự nhiên TP.HCM
Khoa Toán Tin-học
Bộ môn Giải tích

ngththuong@hcmus.edu.vn

Ngày 17 tháng 7 năm 2022

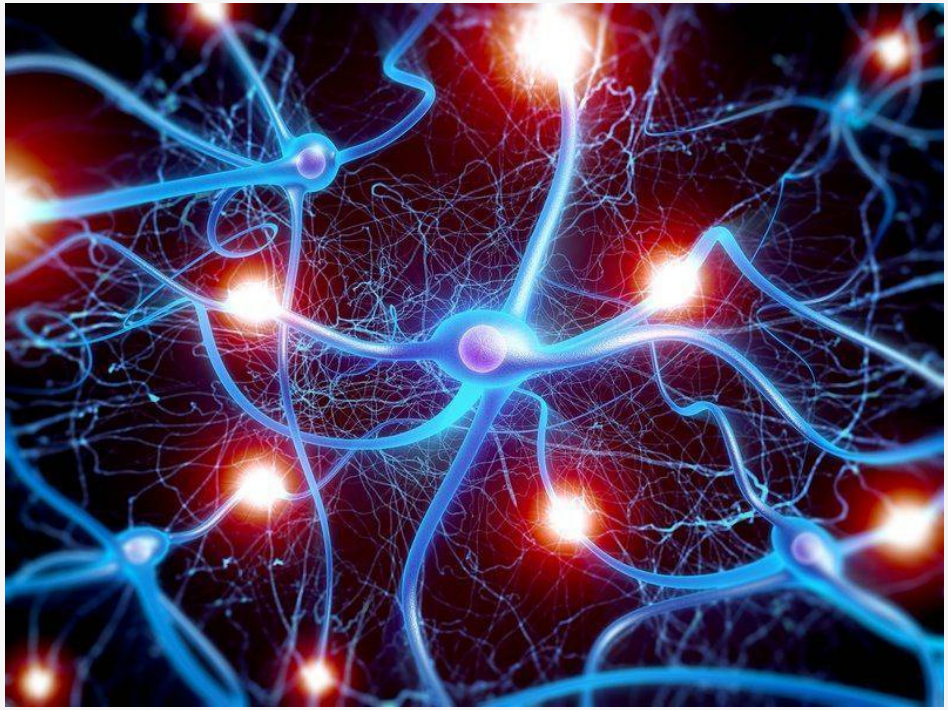
Mục lục

1. Mạng nơ-ron nhân tạo
   - Giới thiệu
   - Mô hình hóa nơ-ron sinh học
   - Cấu trúc của mạng nơ-ron
   - Quá trình lan truyền mạng (propagation) của mạng nơ-ron
   - Huấn luyện một mạng nơ-ron

Giới thiệu

- Mạng nơ-ron nhân tạo (Artificial Neural Network) gọi tắt là mạng nơ-ron (neural network):
  - Là mô hình xử lí thông tin được mô phỏng dựa trên cách thức xử lí thông tin của hệ thống thần kinh của sinh vật.
  - Nó được tạo ra từ một số lượng lớn các đơn vị xử lí (hay nơ-ron) kết nối với nhau thông qua các liên kết (gọi là trọng số liên kết) làm việc như một thể thống nhất để giải quyết một vấn đề cụ thể nào đó.
- Mạng nơ-ron giống như não con người, được học hỏi kinh nghiệm (thông qua huấn luyện), có khả năng lưu trữ những kinh nghiệm hiểu biết (tri thức) và sử dụng những tri thức đó trong việc dự đoán các dữ liệu chưa biết.
- Các ứng dụng của mạng nơ-ron được sử dụng trong rất nhiều lĩnh vực như điện, điện tử, kinh tế, quân sự,... để giải quyết các bài toán có độ phức tạp và đòi hỏi có độ chính xác cao như điều khiển tự động, khai phá dữ liệu, nhận dạng.

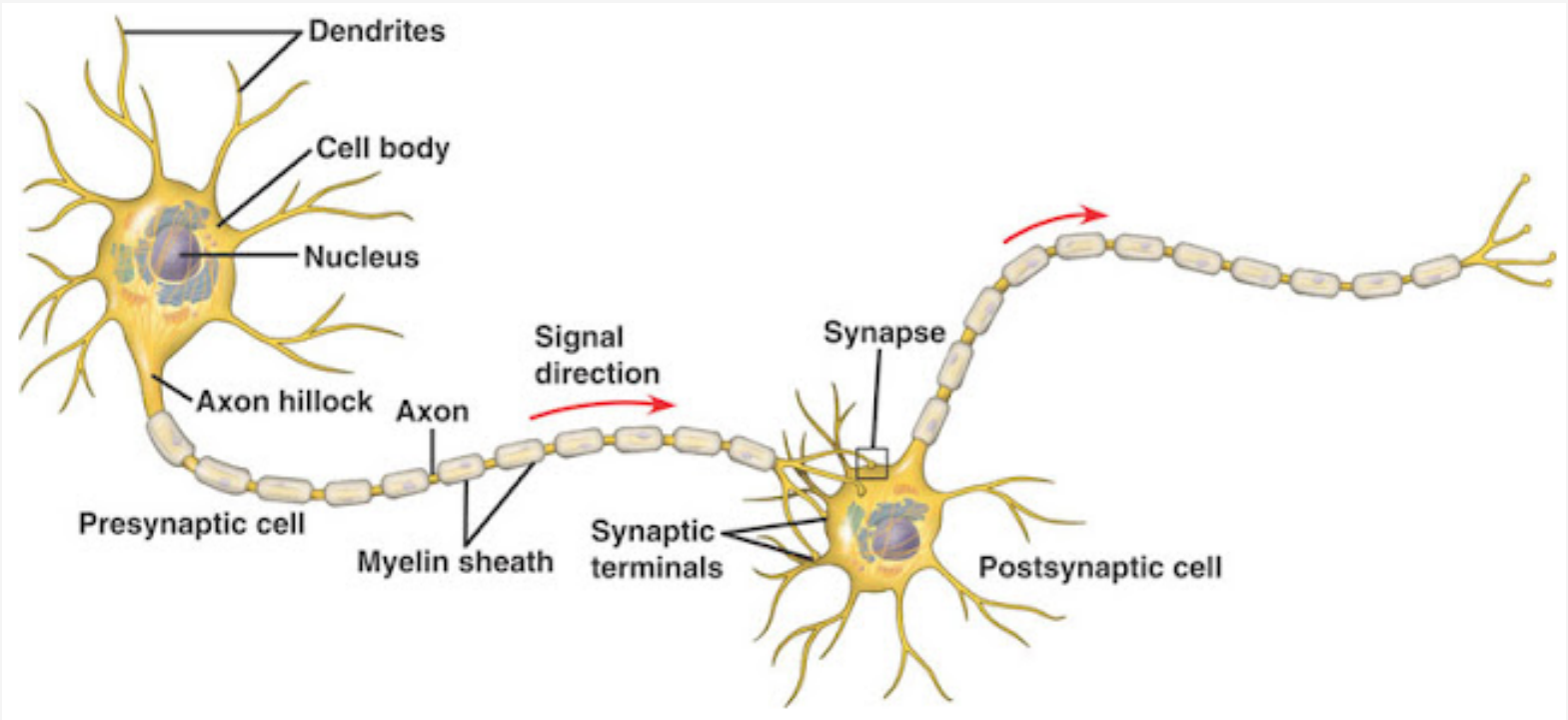
Mô hình nơ-ron sinh học



Hình 1: Mạng nơ-ron sinh học

- Nơ-ron là đơn vị cơ bản cấu tạo nên hệ thống thần kinh và là một phần quan trọng nhất của não.
- Theo các nhà sinh học, hệ thống thần kinh của con người khoảng 10 triệu nơ-ron và mỗi nơ-ron liên kết với 10000 nơ-ron khác.

Mô hình nơ-ron sinh học

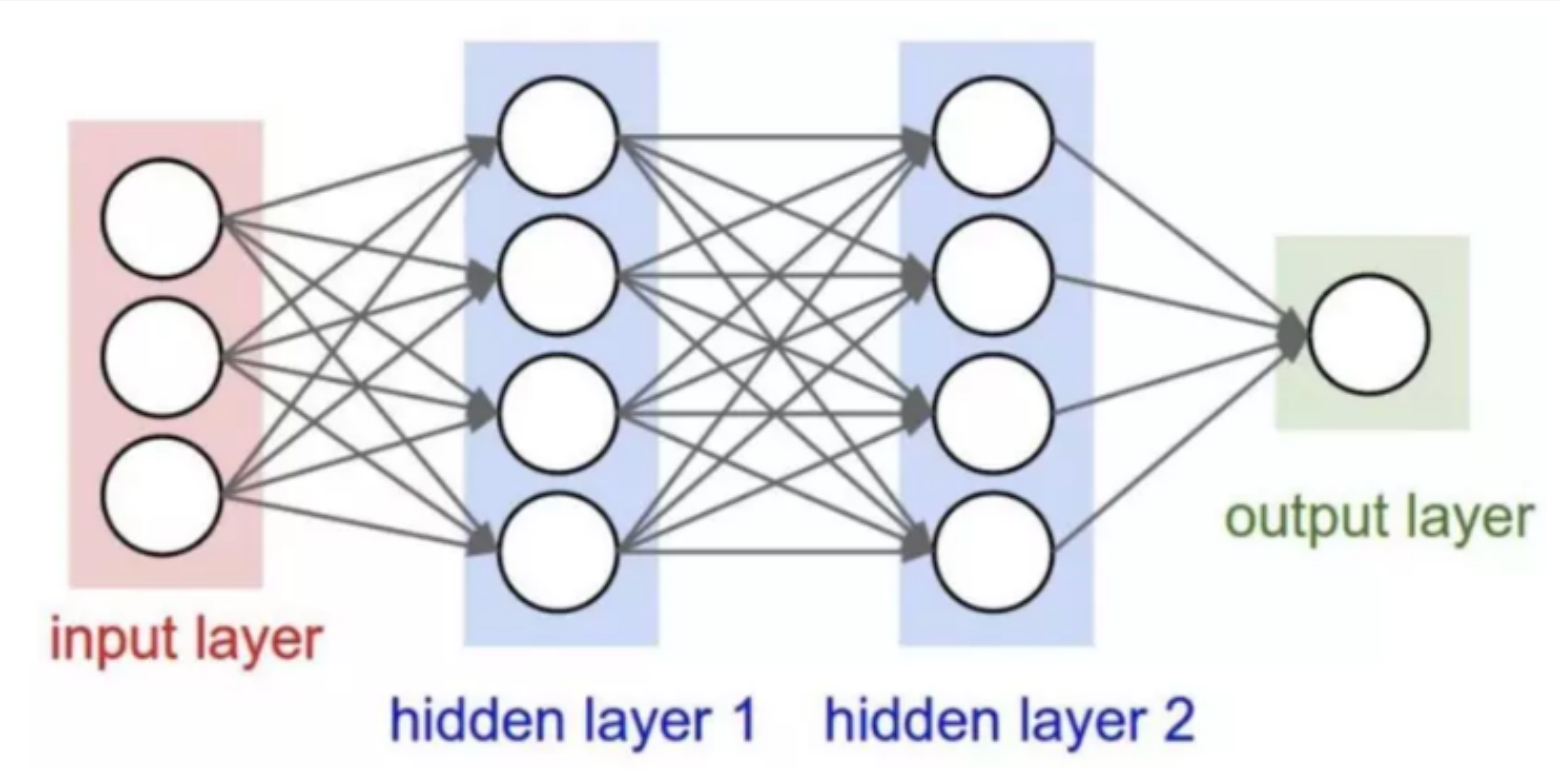


Hình 2: Cấu tạo của nơ-ron.

- Ở mỗi nơ-ron có phần thân (cell body) chứa nhân (nucleus), các tín hiệu đầu vào qua sợi nhánh (dendrites) và các tín hiệu đầu ra qua sợi trục (axon) kết nối với các nơ-ron khác thông qua các khớp nối thần kinh (synapse).
- Hiểu đơn giản, mỗi nơ-ron nhận dữ liệu đầu vào qua sợi nhánh và truyền dữ liệu đầu ra qua sợi trục để đến các sợi nhánh của các nơ-ron khác.

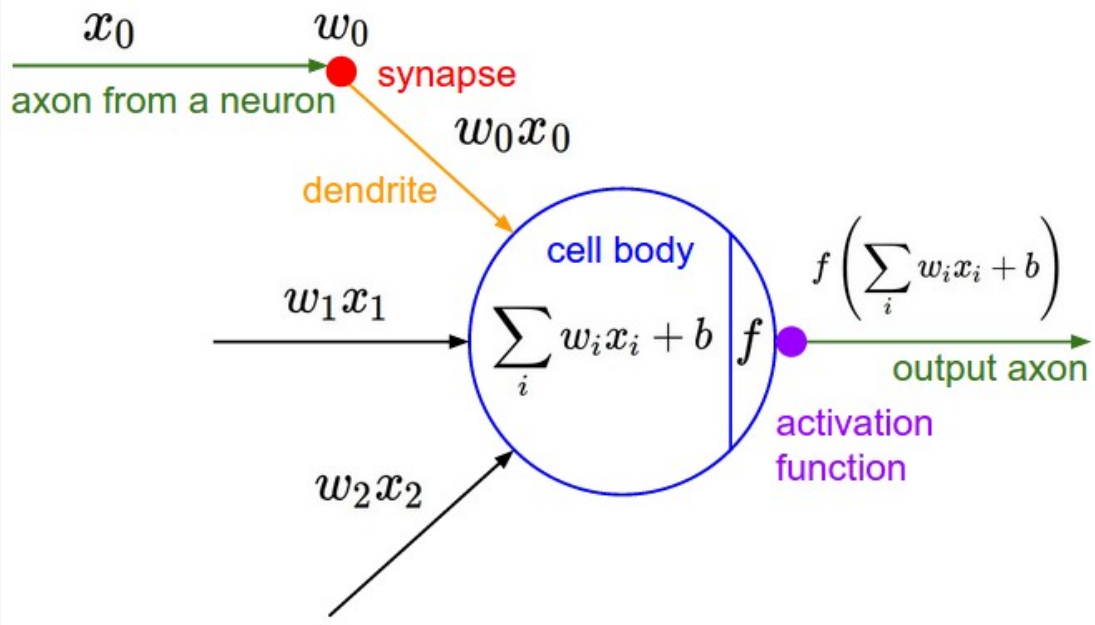
Mô hình mạng nơ-ron

- Mạng nơ-ron được xây dựng dựa trên mạng nơ-ron sinh học. Nó gồm các nơ-ron (nút) nối với nhau và xử lí thông tin bằng cách truyền theo các kết nối và tính giá trị tại các nút.



Mô hình mạng nơ-ron

- Một nơ-ron là 1 đơn vị xử lí thông tin và là thành phần cơ bản của một mạng nơ-ron. Cấu trúc của một nơ-ron được mô tả như hình sau:



Hình 3: Mô hình hóa một nơ-ron.

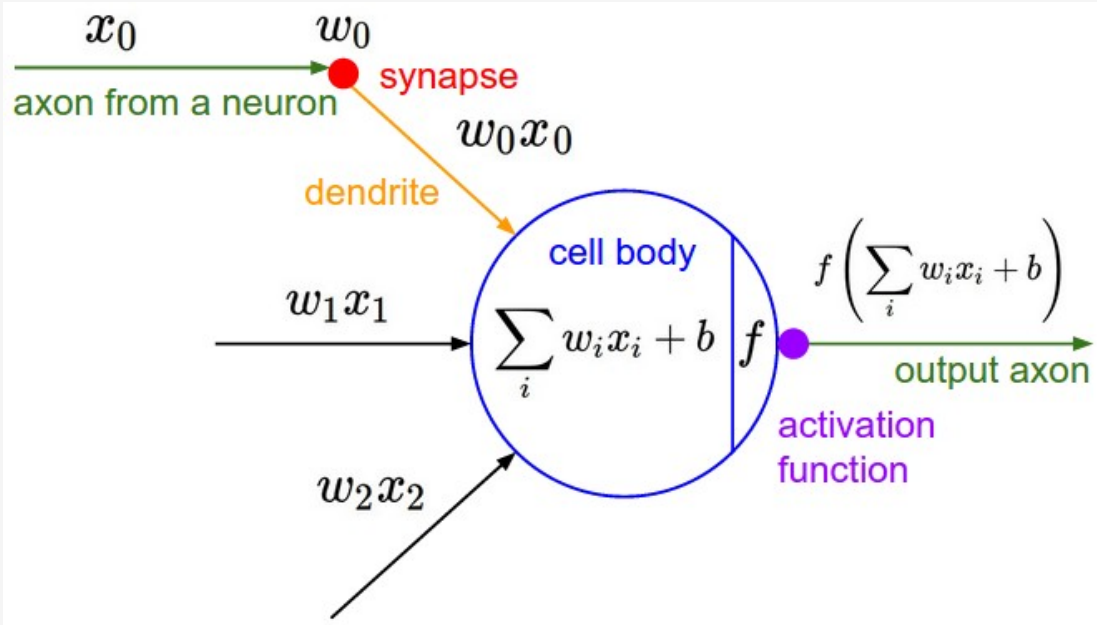
  - Một nơ-ron sẽ được một hoặc nhiều đầu vào x_i. Các đầu vào đóng vai trò như các sợi nhánh thần kinh (dendrites) nhận tính hiệu từ các nơ-ron khác.
  - Các giá trị đầu vào x_i được điều phối tầm ảnh hưởng bởi các trọng số tương ứng w_i của nó và được gọi là trọng số liên kết.

Trọng số liên kết

- Đây là thành phần rất quan trọng của một mạng nơ-ron. Nó thể hiện mức độ quan trọng (độ mạnh) của dữ liệu đầu vào đối với quá trình xử lí thông tin (quá trình chuyển đổi dữ liệu từ nút này sang nút khác).
- Thông thường, các trọng số này được khởi tạo một cách ngẫu nhiên ở thời điểm khởi tạo mạng nơ-ron và được cập nhật liên tục trong quá trình huấn luyện (training).

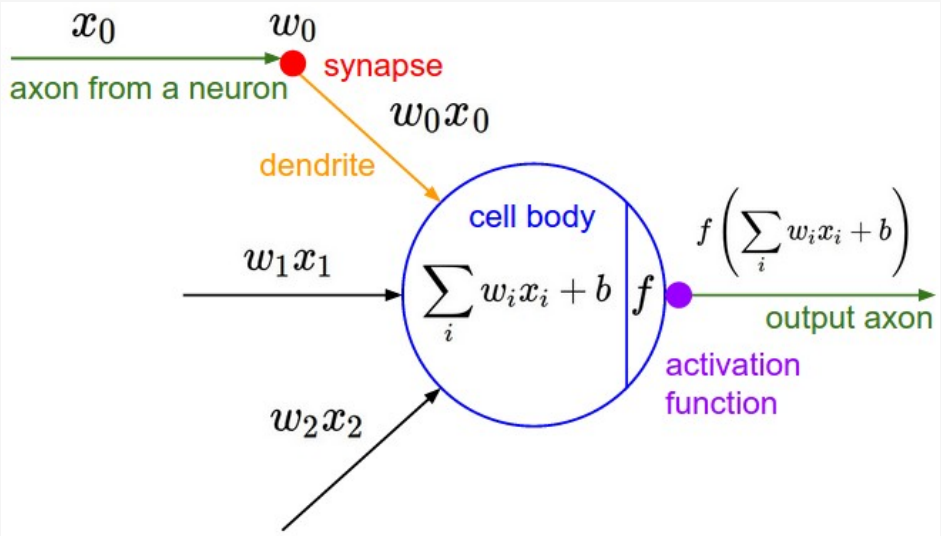
Trọng số liên kết

- Tại mỗi nút, chúng ta sẽ biến đổi những dữ liệu đầu vào này bằng cách tính tổng các giá trị đầu vào với trọng số liên kết tương ứng trên các đầu vào.



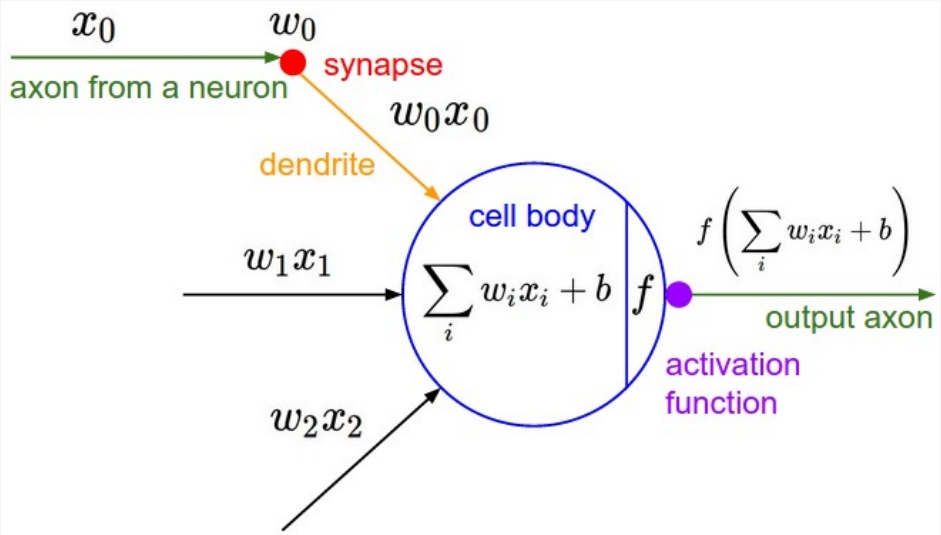
  - Nếu w_i và x_i là các giá trị dương thì giá trị đầu vào x_i góp phần làm cho giá trị đầu ra tăng. Trong trường hợp này, các khớp nối được gọi là **các khớp kích thích** (excitatory synapse).
  - Trong trường hợp giá trị của w_i là số âm thì các trọng số w_i làm cho đảo giá trị nhận được từ x_i. Các khớp nối có giá trị trọng số âm được gọi là **khớp nối hạn chế** (Inhibitory synapse).

Giá trị độ lệch (bias)



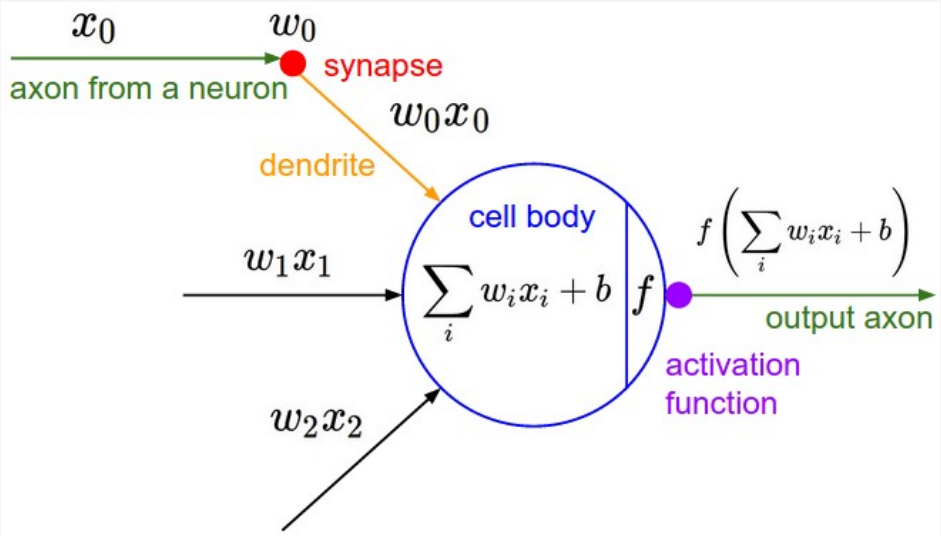
- Ngoài các trọng số, một thành phần khác được áp dụng cho thông tin đầu vào, được gọi là giá trị đầu vào (bias).
- Nó được thêm vào kết quả của phép nhân trọng số với giá trị đầu vào.
- Bởi vì kết quả của phép nhân luôn là một hàm tuyến tính đi qua gốc tọa độ nên việc sử dụng bias giúp mạng nơ-ron có thể dịch chuyển hàm tuyến tính này một cách linh hoạt hơn để mô hình khớp với dữ liệu được huấn luyện.
- Bias cũng là một tham số được học (learning) trong quá trình huấn luyện mạng nơ-ron.

Hàm kích hoạt (Activation function)



- Là một thành phần rất quan trọng trong mô hình mạng nơ-ron. Nó quyết định khi nào một nơ-ron được kích hoạt hay không được kích hoạt. Liệu thông tin mà nơ-ron nhận được có liên quan đến thông tin được đưa ra hay nên bỏ qua dựa trên kết quả tổng hợp các tín hiệu đầu vào mà nó nhận được từ các nơ-ron trước đó.

Hàm kích hoạt (Activation function)



- Nếu không có hàm kích hoạt thì trọng số liên kết và bias chỉ đơn giản như 1 hàm biến đổi tuyến tính. Giải 1 hàm tuyến tính sẽ đơn giản hưn nhiều nhưng sẽ khó có thể mô hình hóa à giải được những vấn đề phức tạp.
- Hàm kích hoạt hỗ trợ quá trình lan truyền ngược (back-propagation) với việc cung cấp các lỗi để cập nhật lại các giá trị trọng số liên kết và bias. Việc này giúp mô hình có khả năng tự hoàn thiện và đạt được kết quả như ta mong muốn.

Hàm kích hoạt (Activation function)

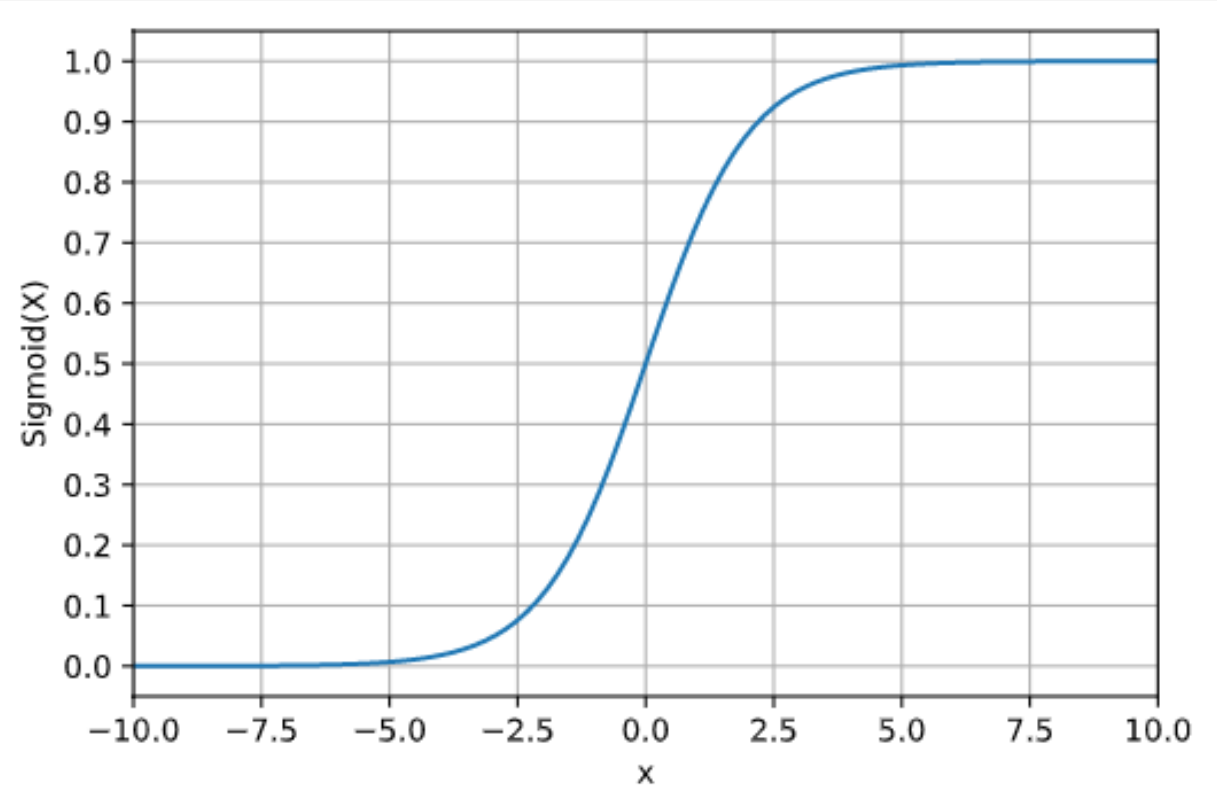
Tại sao hàm kích hoạt luôn là các hàm phi tuyến (non-linear) mà không phải là hàm tuyến tính?

- Lí do xuất phát chính từ cấu trúc của mạng nơ-ron.
- Giả sử chúng ta đều sử dụng các hàm tuyến tính để làm hàm kích hoạt thì không khác gì việc chúng ta dùng thêm một tầng ẩn nữa vì các phép biến đổi cũng chỉ đơn thuần là nhân thêm với một trọng số liên kết w nào đó.
- Với các phép biến đổi đơn giản như vậy thì mô hình mạng nơ-ron sẽ không có khả năng học được những mối quan hệ giữa các dữ liệu, cũng như không có khả năng giải quyết được những bài toán phức tạp như xử lý ảnh hay xử lí ngôn ngữ tự nhiên,...

Một số hàm kích hoạt phổ biến

Hàm sigmoid hay còn gọi là hàm logistic. Đây là một hàm được dùng phổ biến nhất trong các mạng nơ-ron nhiều lớp. Hàm được định nghĩa như sau:

$$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}.$$



Hình 4: Đồ thị hàm Sigmoid.

Một số hàm kích hoạt phổ biến

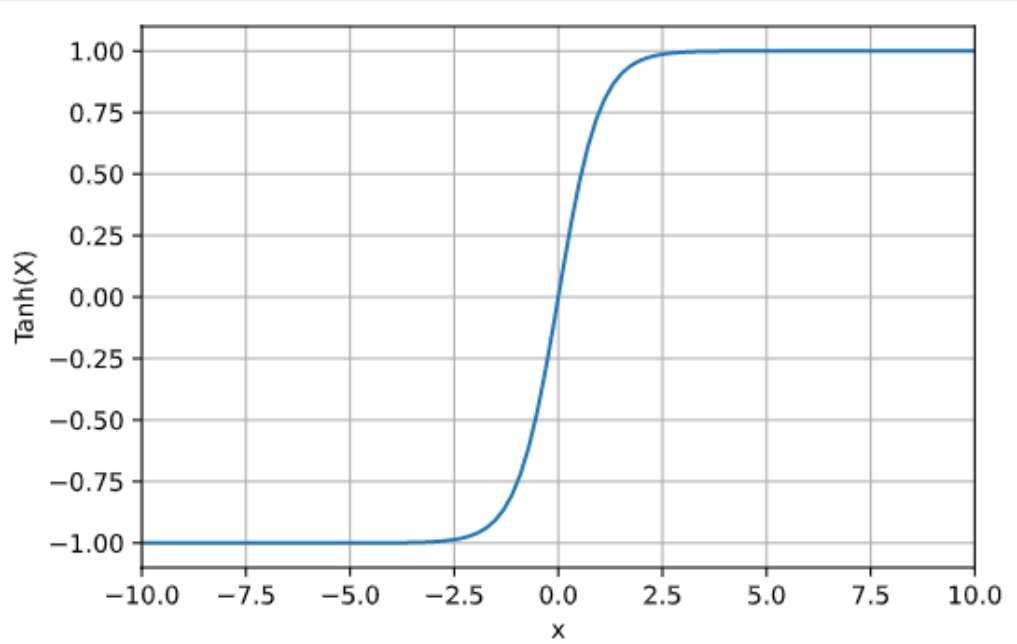
Từ đồ thị trên ta thấy rằng:

- Hàm sigmoid nhận đầu vào là một số thực và chuyển thành một giá trị trong khoảng $(0, 1)$. Nó phù hợp với những bài toán phân loại nhị phân (những bài toán chỉ có giá trị đầu ra là $\{0, 1\}$).
- Nếu đầu vào là số thực âm rất nhỏ thì đầu ra tiệm cận với 0. Ngược lại, nếu đầu vào là một số dương rất lớn thì sẽ cho đầu ra là một số tiệm cận với 1.
- Trong quá khứ, hàm sigmoid hay được dùng vì có đạo hàm rất đẹp. Tuy nhiên, hiện nay hàm sigmoid rất ít được dùng vì khi đầu vào có giá trị tuyệt đối lớn (rất âm hoặc rất dương), gradient của hàm số này sẽ rất gần 0. Điều này đồng nghĩa với việc các trọng số liên kết sẽ gần như không được cập nhật.

Một số hàm kích hoạt phổ biến

Hàm tanh (hyperbolic tangent function) là hàm phi tuyến, tương tự hàm sigmoid. Tuy nhiên, hàm tanh nhận đầu vào là một số thực và chuyển thành một giá trị trong khoảng $(-1, 1)$. Hàm tanh được xác định bởi phương trình như sau

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}.$$



Hình 5: Đồ thị hàm tanh.

Một số hàm kích hoạt phổ biến

- Hàm tanh cũng có thể làm cho gradient bị triệt tiêu tương tự như hàm sigmoid. Quá trình triệt tiêu gradient là quá trình gradient giảm dần khi lan truyền ngược (backward propagion).
- Tuy nhiên, vì có quá nhiều lớp được thiết kế nên khi lan truyền đến các lớp ở xa, gradient trở nên nhỏ và có thể bằng 0, dẫn đến không có giá trị cập nhật cho các trọng số trong lớp đó.
- Hàm tanh còn có thể được biểu diễn bằng hàm sigmoid như sau:

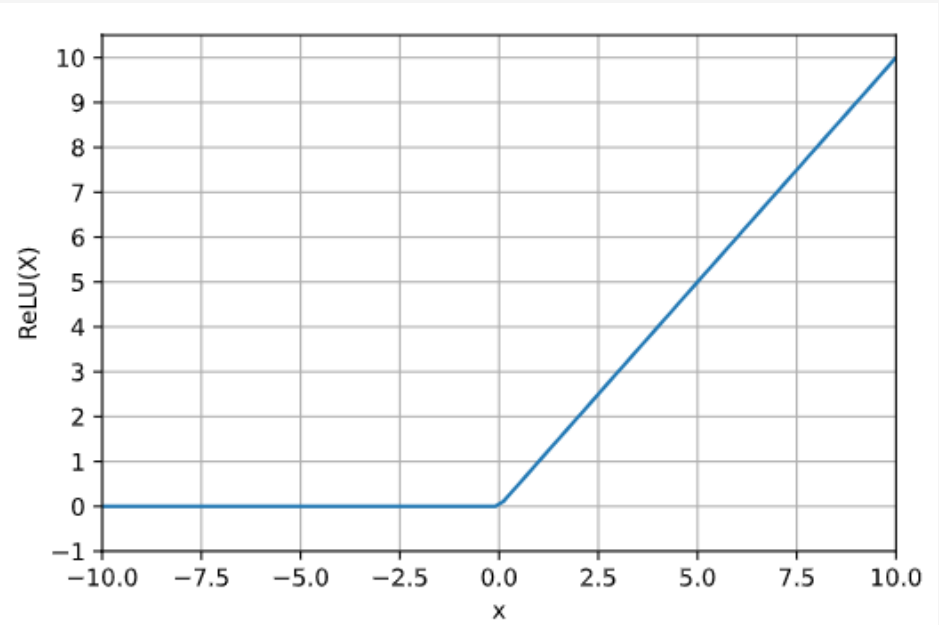
$$\tanh(x) = 2\sigma(2x) - 1.$$

Một số hàm kích hoạt phổ biến

Hàm đơn vị tuyến tính được chỉnh lưu ReLU (Rectified linear unit)

- Đây là hàm đang được sử dụng khá nhiều trong những năm gần đây khi huấn luyện mạng nơ-ron.
- Hàm ReLU đơn giản là lọc các giá trị nhỏ hơn 0 và được thể hiện bởi công thức

$$f(x) = \max(0, x).$$



Hình 6: Đồ thị hàm ReLU.

- Hàm ReLU chỉ có thể được sử dụng trong những lớp ẩn của mạng nơ-ron.

Một số hàm kích hoạt phổ biến

Một số ưu điểm khá vượt trội của *ReLU* so với *sigmoid* và *tanh*:

- Tốc độ hội tụ nhanh hơn hẳn. ReLU có tốc độ hội tụ nhanh gấp 6 lần Tanh. Điều này có thể do ReLU không bị bão hòa (khắc phục được tình trạng triệt tiêu gradient) ở 2 đầu như hàm sigmoid và tanh.
- Tính toán nhanh hơn. Tanh và sigmoid sử dụng hàm *exp* và công thức phức tạp hơn ReLU rất nhiều. Do vậy, sẽ tốn nhiều chi phí hơn để tính toán.

Một số hàm kích hoạt phổ biến

Nhược điểm:

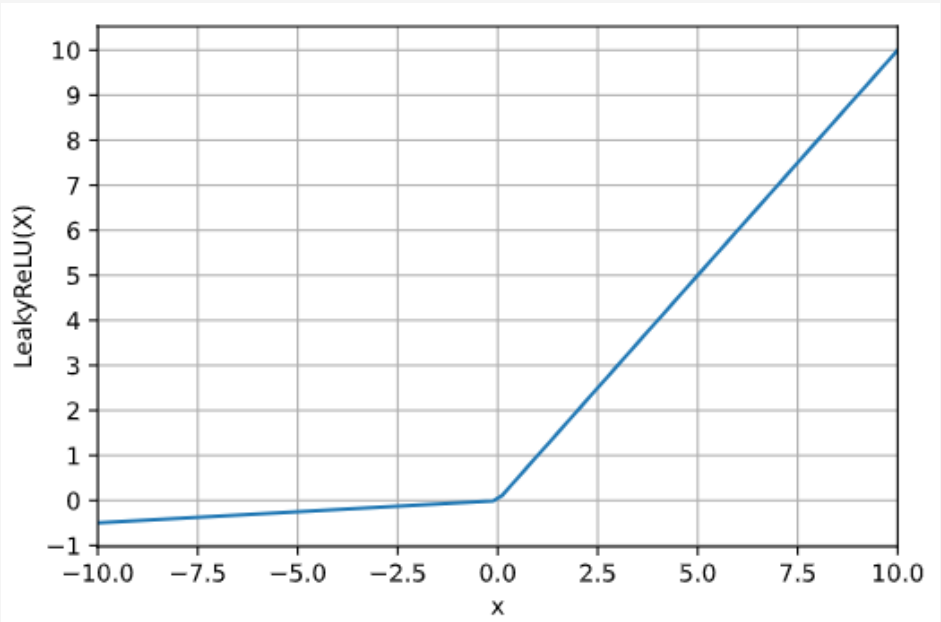
- Với các nút có giá trị nhỏ hơn 0, qua hàm kích hoạt ReLU sẽ thành 0, hiện tượng này được gọi là "Dying ReLU". Cụ thể là trọng số liên kết của nơ-ron được cập nhật và tổng của trọng số liên kết với các giá trị đầu vào của nơ-ron nhỏ hơn 0 dẫn đến đầu ra bằng 0 và gradient bằng 0.
- Nếu các nút bị chuyển thành 0 thì sẽ không có ý nghĩa với bước tuyến tính hóa hàm kích hoạt ở lớp tiếp theo và các hệ số tương ứng từ nút ấy cũng không được cập nhật với thuật toán gradient descent $\Rightarrow$ Reaky LU ra đời.
- Khi learning rate lớn, các trọng số liên kết có thể thay đổi theo cách làm cho tất cả nơ-ron dừng cập nhật.

Một số hàm kích hoạt phổ biến

Leaky ReLU là một cố gắng trong việc loại bỏ "Dying ReLU" và được định nghĩa như sau

$$f(x) = \begin{cases} x, & x \geq 0 \\ \alpha x, & x < 0 \end{cases}$$

với α là hằng số nhỏ.



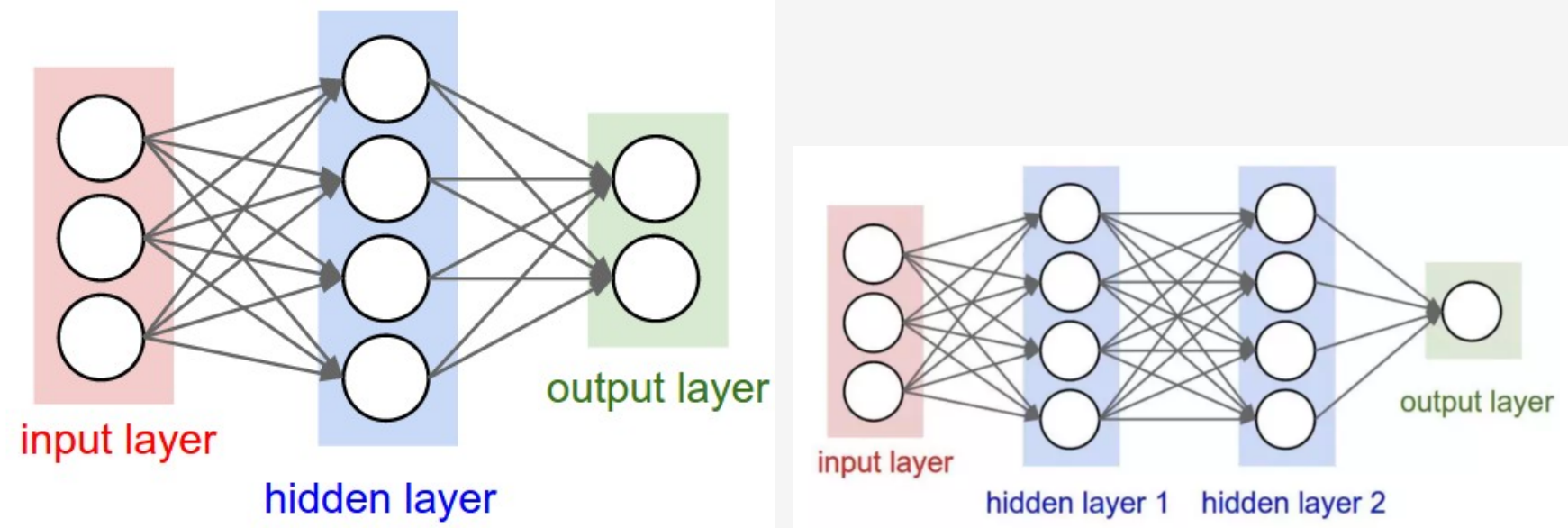
Hình 7: Đồ thị hàm Leaky ReLU.

Một số hàm kích hoạt phổ biến

- Thay vì trả về giá trị 0 với các đầu vào nhỏ hơn 0 thì Leaky ReLU tạo ra một đường xiên có độ dốc nhỏ.
- Khi $x > 0$ thì đạo hàm của Leaky ReLU theo x bằng 1. Khi $x < 0$ thì đạo hàm của Leaky ReLU theo x bằng α. Vì vậy, hàm kích hoạt nàu phù hợp với những bài toán cần đạo hàm cố định, không phụ thuộc vào giá trị đầu vào x. Do đó, tốc độ học được đảm bảo dù giá trị đầu vào x có lớn đến bao nhiêu. Còn hàm sigmoid và tanh phù hợp cho những bài toán có tốc độ học giảm đi khi giá trị tuyệt đối của giá trị đầu vào x lớn.

Các tầng/lớp của mạng nơ-ron

Mạng nơ-ron được cấu tọa từ nhiều tầng/lớp (layer) lại với nhau. Trong mỗi lớp lại bao gồm nhiều nơ-ron (nút)



Hình 8: Mạng nơ-ron 1 lớp ẩn (bên trái) và mạng nơ-ron nhiều lớp ẩn (bên phải).

Xét về tính chất của mỗi tầng, chúng ta có thể chia làm 3 tầng chính:

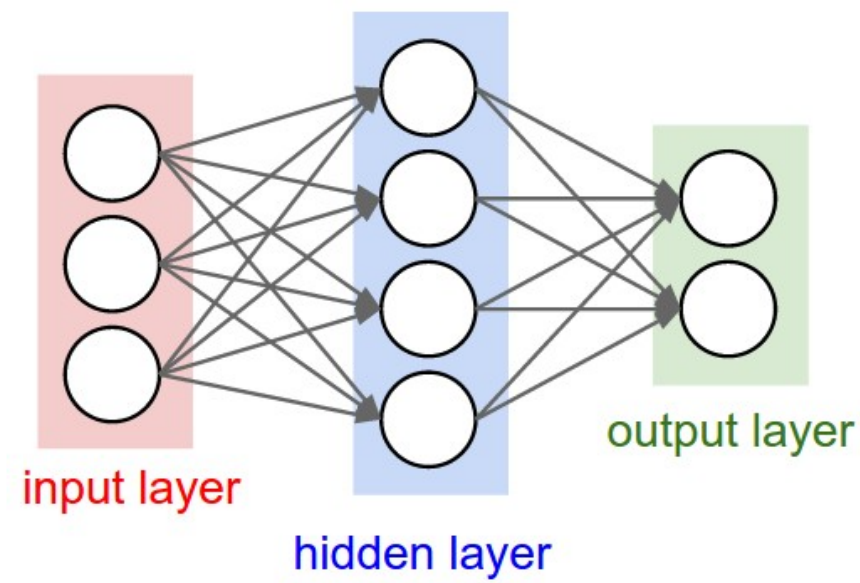
- **Tầng đầu vào** (input layer) là tầng chứa các thông tin đầu vào mà ta muốn đưa vào mạng lưới của ta.

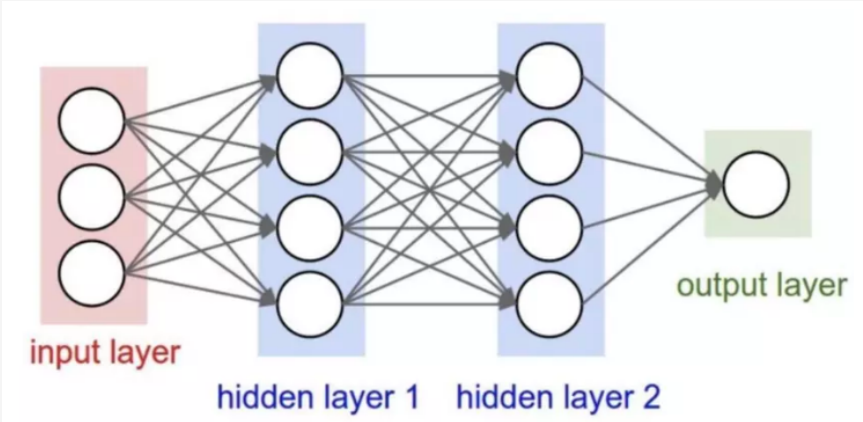
Các tầng/lớp của mạng nơ-ron

- **Tầng đầu ra** (output layer) là tầng chứa các thông tin đầu ra. Kết quả của mạng lưới sẽ được chứa ở đây. Tùy vào từng trường hợp cụ thể mà tầng đầu ra có thể có một hoặc nhiều đơn vị cấu tạo (nơ-ron). Tuy nhiên, trong một mạng nơ-ron sẽ chỉ có 1 tầng đầu vào và một tầng đầu ra.
- **Tầng ẩn** (hidden layer)
  - Là tầng nằm giữa tầng đầu vào và tầng đầu ra.
  - Nó thể hiện cho quá trình suy luận logic của mạng.
  - Nó có nhiệm vụ là nhận dữ liệu đầu vào từ các nơ-ron ở tầng trước đó và thông qua một hàm kích hoạt để chuyển đổi các giá trị đầu vào này cho các tầng xử lí tiếp theo.
  - Theo thứ tự từ trái sang phải, các tầng ẩn được đánh số thứ tự để dễ phân biệt. Bắt đầu từ tầng gần tầng đầu nhất là 1,2,3,... Tuy nhiên, càng nhiều tầng ẩn đồng nghĩa với mạng lưới của chúng ta càng phức tạp và khó đoán. Đây là một điều "khá phiền" để sửa nếu những dự đoán của nó không như ta mong muốn.

Các tầng/lớp của mạng nơ-ron

Giống như sợi trục và khớp nối thần kinh để nói các nơ-ron trong tế bào thần kinh của chúng ta. Những đơn vị cấu tạo nên mạng nơ-ron cũng cần những liên kết giữa chúng, là các mũi tên được vẽ ở hình sau





Theo nguyên tắc, ta sẽ bắt đầu nối từ tầng đầu vào, qua các tầng ẩn 1,2,3,... và cuối cùng là tầng đầu ra.

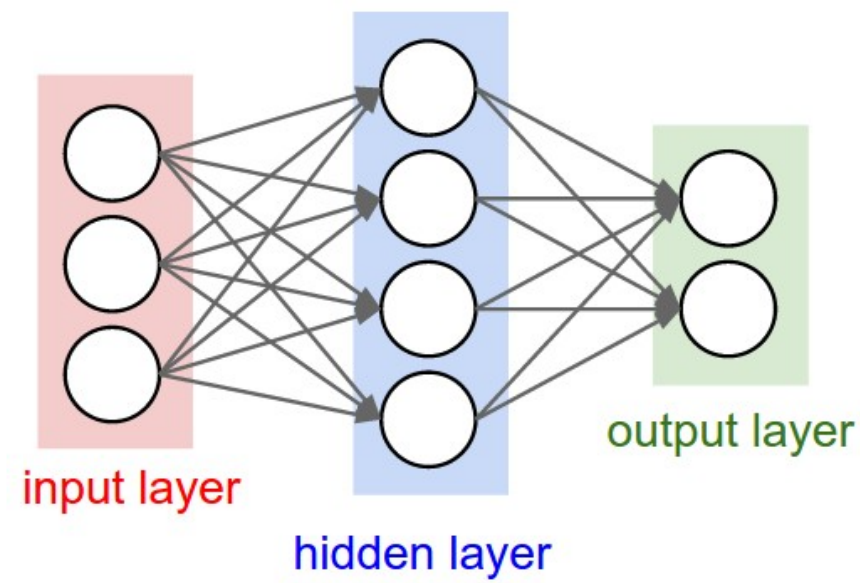
Phân loại mạng nơ-ron

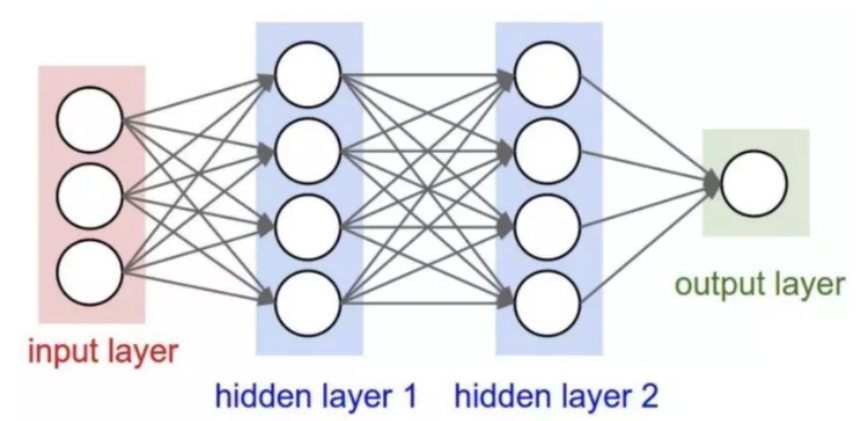
Dựa trên cách thức liên kết giữa các nơ-ron, mạng nơ ron được chia làm hai loại:

- Mạng truyền thẳng (Feed-forward neural network)
- Mạng hồi quy (Recurrent neural network)

Phân loại mạng nơ-ron

Mạng truyền thẳng:

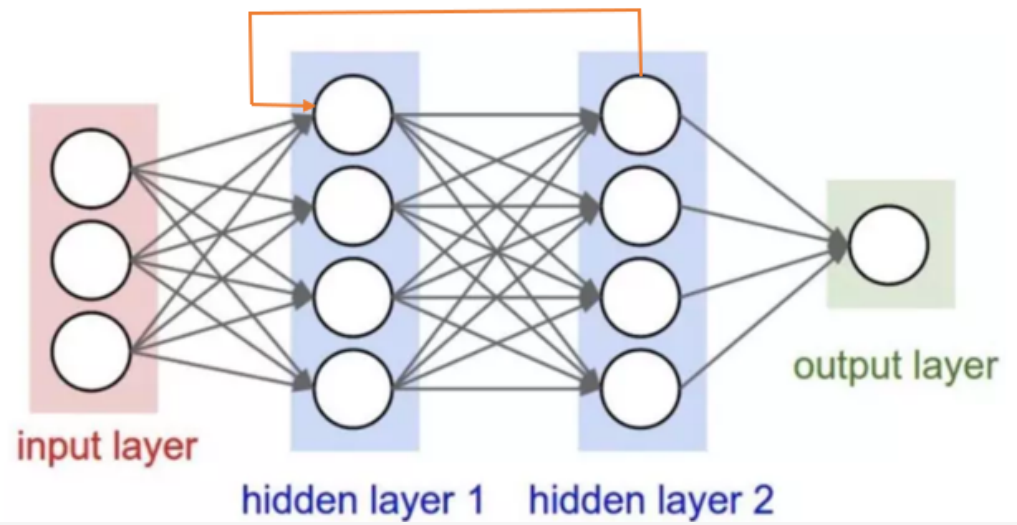




- Bao gồm 1 tầng đầu vào, 1 tầng đầu ra và 1 hoặc nhiều tầng ẩn.
- Dòng dữ liệu từ tầng đầu vào đến tầng đầu ra chỉ được truyền thẳng, không có các liên kết phản hồi (gửi ngược thông tin về lại).
- Mạng nơ-ron truyền thẳng được dùng trong nhiều ứng dụng như phân lớp, nhận dạng, dự đoán,...

Phân loại mạng nơ-ron

Mạng hồi quy:



- Có chứa liên kết ngược. Một nơ-ron trong mạng nơ-ron phản hồi có thể truyền dữ liệu đầu ra của nó cho các nơ-ron khác trong cùng hoặc các tầng trên.
- Các thuật toán sẽ được điều chỉnh đồng thời bởi các tín hiệu từ các nơ-ron khác dựa trên tập dữ liệu tiền tri thức.
- Việc hiệu chuẩn lặp đi lặp lại góp phần vào độ mạnh và độ chính xác của mạng nơ-ron.
- Mạng nơ-ron phản hồi thường được sử dụng để phân tích hình ảnh, chuẩn đoán và dự đoán kết quả.

Quá trình lan truyền mạng (propagation) của mạng nơ-ron

Ví dụ: Tạo một mạng nơ-ron mô phỏng phép toán XOR (hay cổng logic XOR)

- **Đầu vào:** 0 và 1.
- **Đầu ra:** phép toán XOR tuân theo quy luật sau:
  - Nếu 2 giá trị đầu vào giống nhau thì giá trị đầu ra là 0.
  - Nếu 2 giá trị đầu vào khác nhau thì giá trị đầu ra bằng 1.

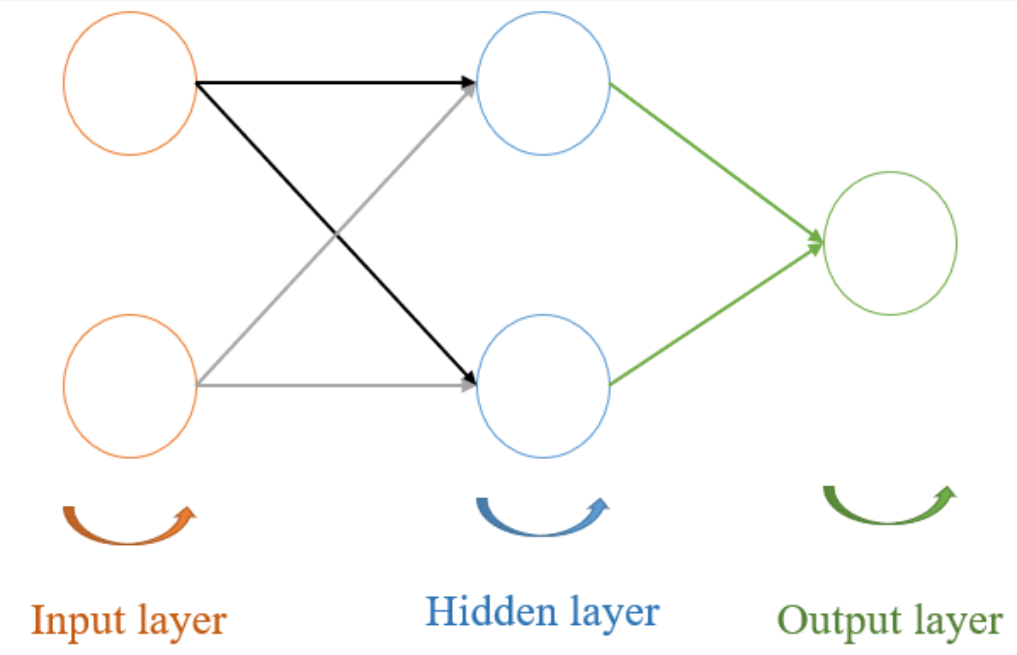
Bảng chân lí:

Đầu vào 1	Đầu vào 2	Đầu ra
0	0	0
1	1	0
1	0	1
0	1	1

Quá trình lan truyền mạng (propagation) của mạng nơ-ron

Ta thấy rằng mạng nơ-ron của chúng ta thiết kế gồm có 3 tầng:

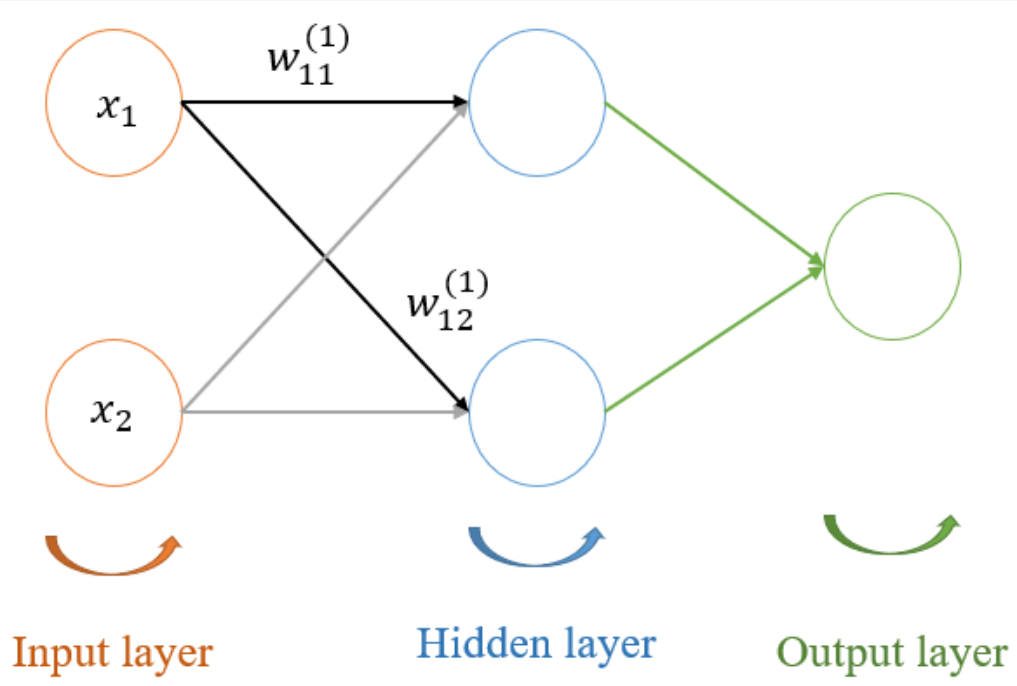
- **Tầng đầu vào:** có 2 nơ-ron (nút) tương ứng với 2 giá trị đầu vào.
- **Tầng đầu ra:** có 1 nơ-ron (nút) để xuất ra kết quả của phép toán XOR.
- **Tầng ẩn:** chúng ta giả sử rằng nó có 2 nơ-ron



Quá trình lan truyền mạng (propagation) của mạng nơ-ron

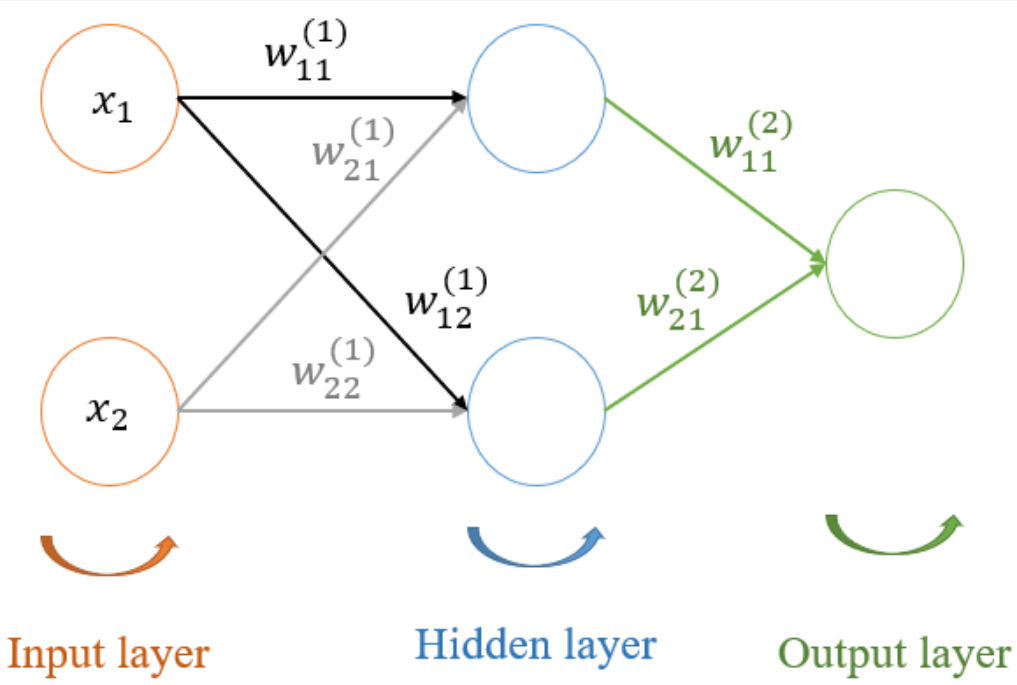
Đặt

- Đặt x_1 và x_2 lần lượt là 2 giá trị đầu vào.
- $w_{11}^{(1)}$ và $w_{12}^{(1)}$ lần lượt là các trọng số liên kết giữa giá trị đầu vào x_1 với nơ-ron thứ nhất và thứ hai của tầng ẩn.



Quá trình lan truyền mạng (propagation) của mạng nơ-ron

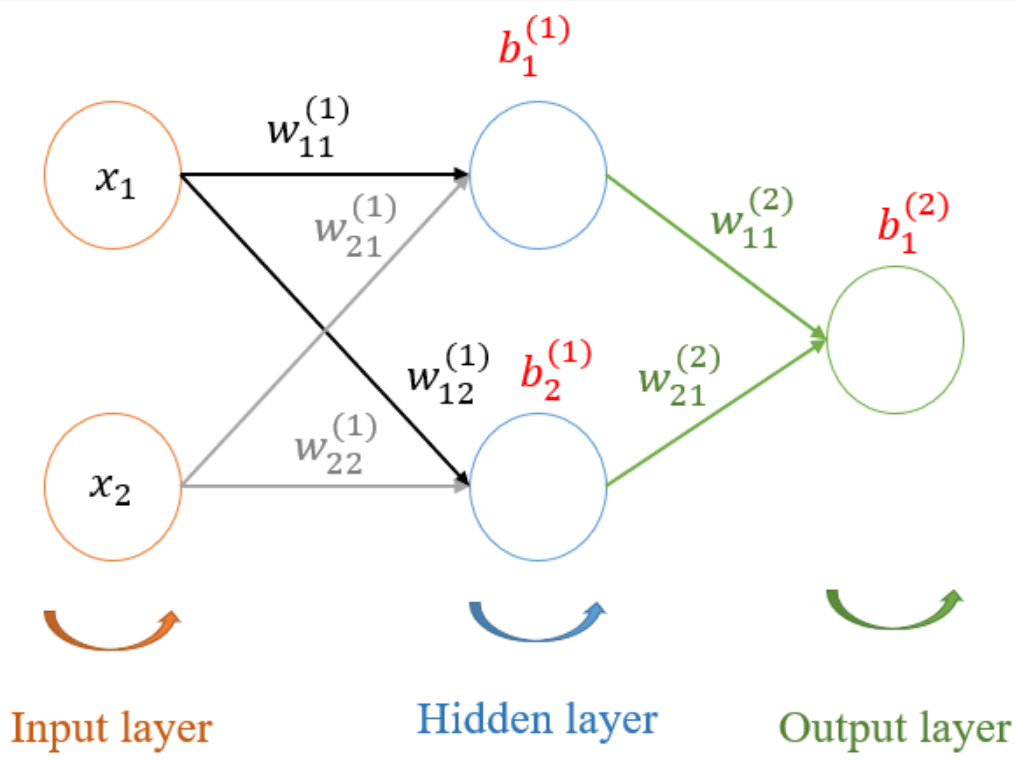
- $w_{21}^{(1)}$ và $w_{22}^{(1)}$ lần lượt là các trọng số liên kết giữa giá trị đầu vào x_2 với nơ-ron thứ nhất và thứ hai của tầng ẩn.
- $w_{11}^{(2)}$ và $w_{21}^{(2)}$ lần lượt là trọng số liên kết giữa nơ-ron thứ nhất và thứ hai của tầng ẩn với nơ-ron của tầng đầu ra.



Quá trình lan truyền mạng (propagation) của mạng nơ-ron

Do trong quá trình xử lí của mỗi nơ-ron ở tầng ẩn và tầng đầu ra đều cần có các giá trị độ lệch (bias) nên ta kí hiệu

- $b_1^{(1)}$ và $b_2^{(1)}$ là các giá trị độ lệch tương ứng với nơ-ron thứ nhất và nơ-ron thứ hai của tầng ẩn.
- $b_1^{(2)}$ là giá trị độ lệch tương ứng với nơ-ron ở tầng đầu ra.



Quá trình lan truyền mạng (propagation) của mạng nơ-ron

Tại mỗi nơ-ron trong tầng ẩn và tầng đầu ra:

- Liên kết với tất cả các nơ-ron ở tầng trước đó với các trọng số liên kết w_{ij}^{ℓ}.
- Mỗi nơ-ron đều có một giá trị độ lệch b_i^{ℓ} riêng.
- Diễn ra hai bước chính:
  - Tính tổng trọng số liên kết với các giá trị đầu ra của tầng trước đó.
  - Áp dụng hàm kích hoạt f để đưa ra giá trị đầu ra của nơ-ron đó.

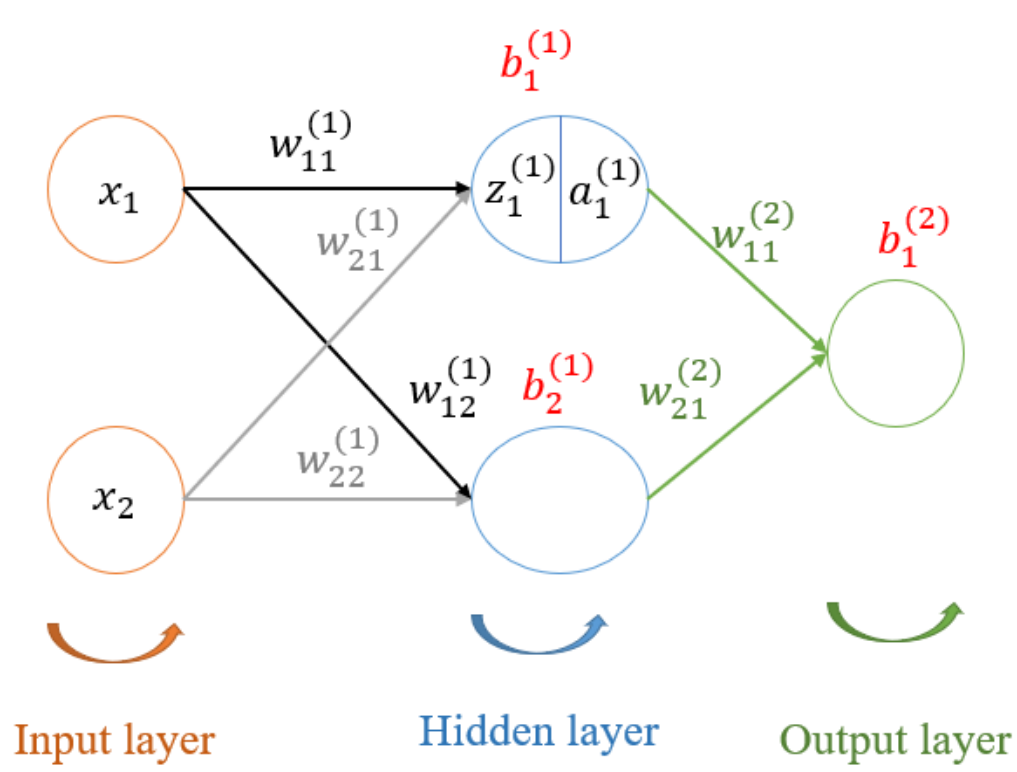
$\Rightarrow$ Quá trình này được gọi là quá trình lan truyền (propagation).

Quá trình lan truyền mạng (propagation) của mạng nơ-ron

Ví dụ:

- Tại nơ-ron thứ nhất của tầng ẩn, ta tiến hành các tính toán sau:

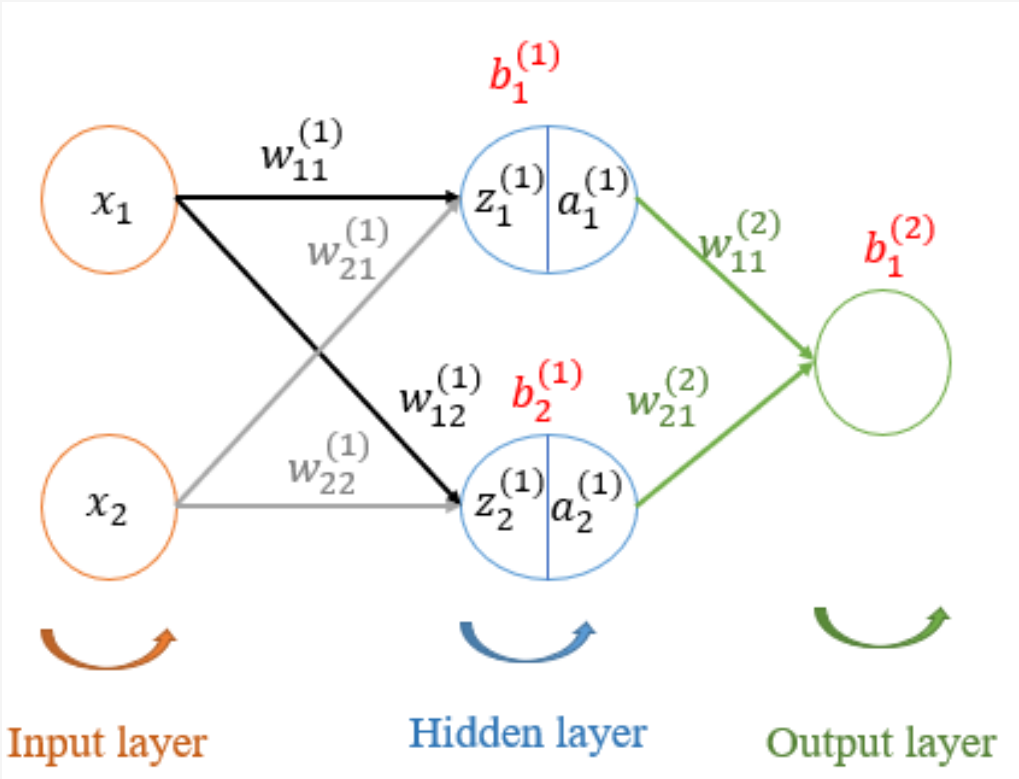
$$\begin{aligned} z_1^{(1)} &:= x_1 w_{11}^{(1)} + x_2 w_{21}^{(1)} + b_1^{(1)} \\ a_1^{(1)} &:= f(z_1^{(1)}). \end{aligned} \tag{1.1}$$



Quá trình lan truyền mạng (propagation) của mạng nơ-ron

- Tại nơ-ron thứ hai của tầng ẩn, ta tiến hành các tính toán sau:

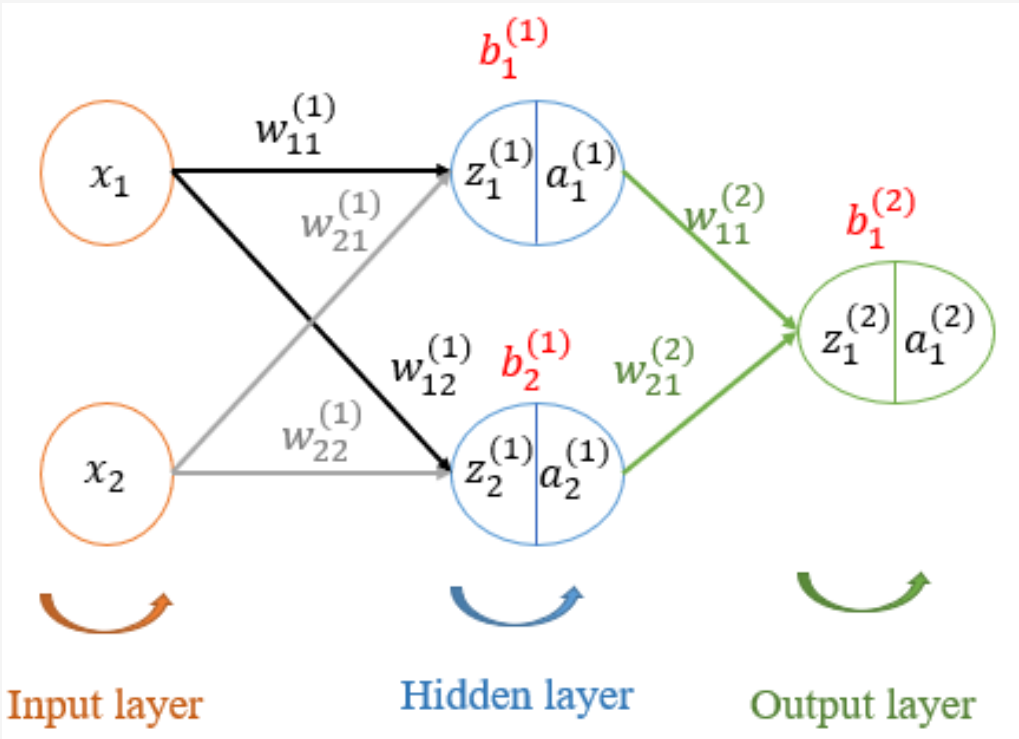
$$
\begin{aligned}
z_2^{(1)} &:= x_1 w_{12}^{(1)} + x_2 w_{22}^{(1)} + b_2^{(1)} \\
a_2^{(1)} &:= f(z_2^{(1)}).
\end{aligned}
\tag{1.2}
$$



Quá trình lan truyền mạng (propagation) của mạng nơ-ron

- Tại nơ-ron của tầng ẩn, ta tiến hành các tính toán sau:

$$
\begin{aligned}
z_1^{(2)} &:= a_1^{(1)} w_{11}^{(2)} + a_2^{(1)} w_{21}^{(2)} + b_1^{(2)} \\
a_1^{(2)} &:= f(z_1^{(2)}).
\end{aligned}
\tag{1.3}
$$



Quá trình lan truyền mạng (propagation) của mạng nơ-ron

Lưu ý:

- $\widehat{y_i} = a_1^{(2)}$ chính là giá trị mà mạng nơ-ron này dự đoán.
- Giả sử rằng chúng ta chọn hàm kích hoạt phổ biến nhất là hàm *sigmoid*.

Huấn luyện một mạng nơ-ron

Huấn luyện (traing) một mạng nơ-ron là quá trình cập nhật các trọng số liên kết $w_{ij}^{(\ell)}$ và các giá trị độ lệch (bias) $b_i^{(\ell)}$ sao cho với một tập các giá trị đầu vào $\mathbf{x}_i$, mạng nơ-ron có khả năng tạo ra giá trị đầu ra gần với giá trị mà chúng ta mong muốn.

Cost function

Cost function: Sau khi tính được giá trị $\widehat{y_i}$ thì chúng ta sẽ so sánh giá trị $\widehat{y_i}$ với giá trị thực tế y_i xem nó khác nhau như thế nào.
Ví dụ: Giả sử với $x_1 = 1, x_2 = 0$, giá trị đầu ra (giá trị dự đoán) $\widehat{y_i}$ của mạng nơ-ron là 0.45. Tuy nhiên giá trị thực tế $y_i = 1$. Cost function sẽ chỉ ra sự khác biệt (sai số) giữa $\widehat{y_i}$ và y_i như thế nào.

- Đơn giản nhất chúng ta chọn cost function có dạng

$$J(\widehat{y_i}, y_i) = |\widehat{y_i} - y_i| = 0.55.$$

Mặc dù hàm tính toán đơn giản nhưng nó không hiệu quả cho các tính toán về sau. Do đó, chúng ta không dùng hàm này.

Cost function

- Ở đây, chúng ta sẽ định nghĩa một cost function khác

$$J(\widehat{y}_i, y_i) = -\left(y_i \ln(\widehat{y}_i) + (1 - y_i) \ln(1 - \widehat{y}_i)\right).$$

$\rightsquigarrow$ Hàm này có vẻ rất phức tạp nhưng thực tế thì nó rất có ích cho các tính toán sau này.
$\rightsquigarrow$ Trong trường hợp $y_i = 1$ thì $J(\widehat{y}_i, 1) = -\ln(\widehat{y}_i)$. Nếu chúng ta muốn giá trị J nhỏ thì chúng ta chỉ cần tăng giá trị $\widehat{y}_i$ lên.
$\rightsquigarrow$ Trong trường hợp $y_i = 0$ thì $J(\widehat{y}_i, 1) = -\ln(1 - \widehat{y}_i)$. Nếu chúng ta muốn giá trị J nhỏ thì chúng ta chỉ cần giảm giá trị $\widehat{y}_i$ xuống.
Do đó với cost function như thế này thì nó sẽ giúp chúng ta định hướng được việc chúng ta sẽ phải thay đổi trọng số liên kết và giá trị độ lệch như thế nào để $\widehat{y}_i$ tăng lên hay $\widehat{y}_i$ giảm xuống.

Loss function

Loss function là giá trị trung bình cộng của cost function trên toàn bộ dữ liệu được cho dưới dạng sau

$$L(\mathbf{w}, \mathbf{b}) = \frac{1}{m} \sum_{i=1}^{m} J(\widehat{y}_i, y_i)$$

trong đó

- m là số lượng dữ liệu trong tập dữ liệu.
- **w** và **b** là tập hợp các trọng số liên kết và giá trị độ lệch trong mạng nơ-ron.

$\rightsquigarrow$ Loss function chỉ phụ thuộc vào các trọng số liên kết và giá trị độ lệch của mạng nơ-ron. Khi giá trị của các trọng số liên kết và giá trị độ lệch thay đổi thì loss function sẽ thay đổi.

$\rightsquigarrow$ Mục tiêu của quá trình huấn luyện là giảm giá trị của loss function bằng cách vứi mỗi bước lặp thì chúng ta sẽ hiệu chỉnh giá trị của các trọng số liên kết và giá trị độ lệch sao cho giá trị của loss function sẽ giảm xuống tới mức tối ưu nhất có thể.

Quá trình huấn luyện của mạng nơ-ron

- **Bước 1:** Để bắt đầu quá trình huấn luyện mang nơ-ron, chúng ta cần khởi tạo giá trị ngẫu nhiên cho các trọng số liên kết và giá trị độ lệch.
- **Bước 2:**Thực hiện quá trình lan truyền trong mạng nơ-ron để tìm ra giá trị dự đoán $\widehat{y_i}$ (**quá trình lan truyền tiến** (forward propagation)). Với các dữ liệu ở tầng đầu vào, ta tiến hành:
  - Tính toán các giá trị z_i^ℓ ở mỗi nơ-ron
  - Áp dụng hàm kích hoạt để tìm giá trị a_i^ℓ.
  - Tính toán giá trị loss function tương ứng.

Quá trình huấn luyện của mạng nơ-ron

Ví dụ: Quá trình huấn luyện ở lần đầu tiên của mạng nơ-ron được xây dựng ở ví dụ trên

x_1	x_2	$w_{11}^{(1)}$	$w_{21}^{(1)}$	$b_1^{(1)}$	$w_{12}^{(1)}$	$w_{22}^{(1)}$	$b_2^{(1)}$	$z_1^{(1)}$	$a_1^{(1)}$
0	0	0.16	-0.07	0.37	-0.66	0.63	-0.98	0.37	0.59
0	1	0.16	-0.07	0.37	-0.66	0.63	-0.98	0.30	0.57
1	0	0.16	-0.07	0.37	-0.66	0.63	-0.98	0.53	0.63
1	1	0.16	-0.07	0.37	-0.66	0.63	-0.98	0.46	0.61

$z_2^{(1)}$	$a_2^{(1)}$	$w_{11}^{(2)}$	$w_{21}^{(2)}$	$b_1^{(2)}$	$z_1^{(2)}$	$\widehat{y}_i = a_1^{(2)}$	y_i	$J(\widehat{y}_i, y_i)$
-0.98	0.27	0.41	-0.90	0.35	0.35	0.59	0	0.8816
-0.35	0.41	0.41	-0.90	0.35	0.21	0.55	1	0.5921
-1.64	0.16	0.41	-0.90	0.35	0.46	0.61	1	0.4886
-1.01	0.27	0.41	-0.90	0.35	0.36	0.59	0	0.8899

$$\Rightarrow L(\mathbf{w}, \mathbf{b}) = \frac{1}{4} \sum_{i=1}^{4} J(\widehat{y}_i, y_i) = 0.713.$$

Quá trình huấn luyện của mạng nơ-ron

- **Bước 3:** Nếu loss function có giá trị quá lớn thì chúng ta tiến hành hiệu chỉnh các trọng số liên kết và giá trị độ lệch $\Rightarrow$ **quá trình lan truyền ngược** (backward propagation).
  $\Rightarrow$ Dùng thuật toán Gradient descent để hiệu chỉnh các trọng số liên kết và giá trị độ lệch sao cho loss function có giá trị nhỏ nhất.

Thuật toán Gradient descent

Ta có

$$L(\mathbf{w}, \mathbf{b}) = \frac{1}{m} \sum_{i=1}^{m} J(\widehat{y}_i, y_i).$$

Để áp dụng thuật toán Gradient descent cho việc hiệu chỉnh các trọng số liên kết và giá trị độ lệch, ta cần tính đạo hàm của L theo $\mathbf{w}$ và theo $\mathbf{b}$. Để từ đó, ta thực hiện các bước lặp sau

$$\begin{cases} \mathbf{w}_{t+1} = \mathbf{w}_t - \eta \dfrac{\partial L}{\partial \mathbf{w}}(\mathbf{w}_t, b_t), \quad t > 0 \\ \\ b_{t+1} = b_t - \eta \dfrac{\partial L}{\partial b}(\mathbf{w}_t, b_t) \end{cases}$$

trong đó η là learning rate.

Tính đạo hàm L theo **w** và theo **b**

Để thuận tiện cho các bước tính toán tiếp theo, chúng ta lưu ý các tính toán sau:

- Với $J(a,b) = -\left(b\ln(a) + (1-b)\ln(1-a)\right)$, ta có

$$\frac{\partial J}{\partial a} = -\frac{b}{a} + \frac{1-b}{1-a}. \tag{1.4}$$

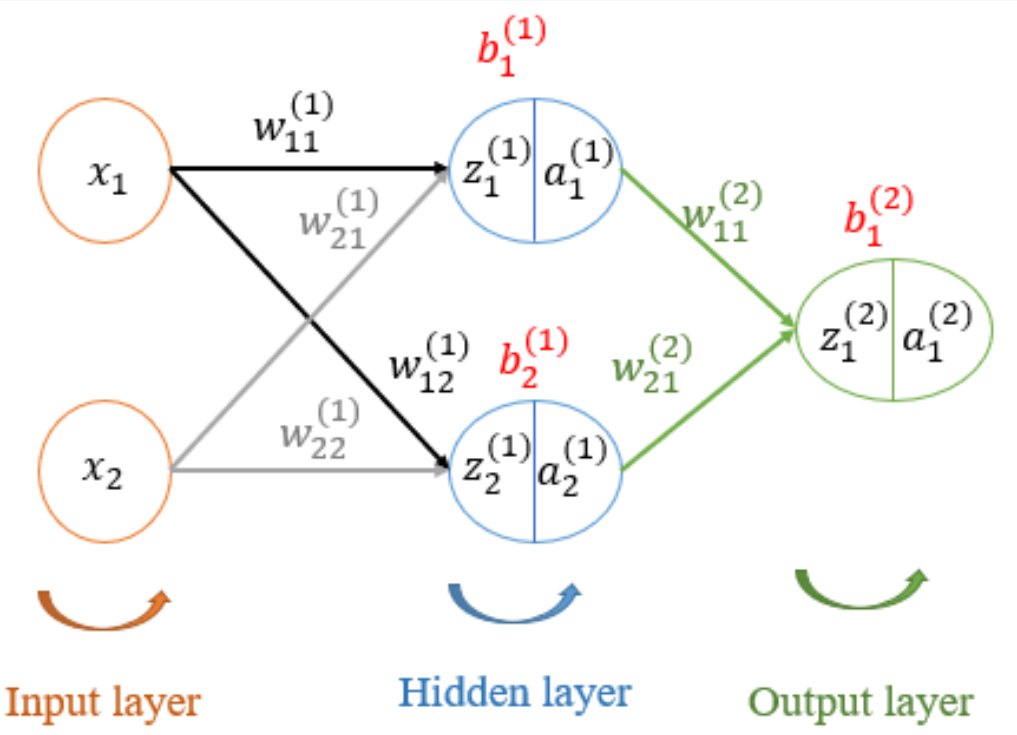
- Với $f(x) = \dfrac{1}{1+e^{-x}}$, ta có

$$\frac{\partial f}{\partial x} = f(x)(1-f(x)). \tag{1.5}$$

Tính đạo hàm L theo **w** và theo **b**

Ở ví dụ trước, ta thấy rằng mạng nơ-ron bao gồm các trọng số liên kết và giá trị độ lệch sau:

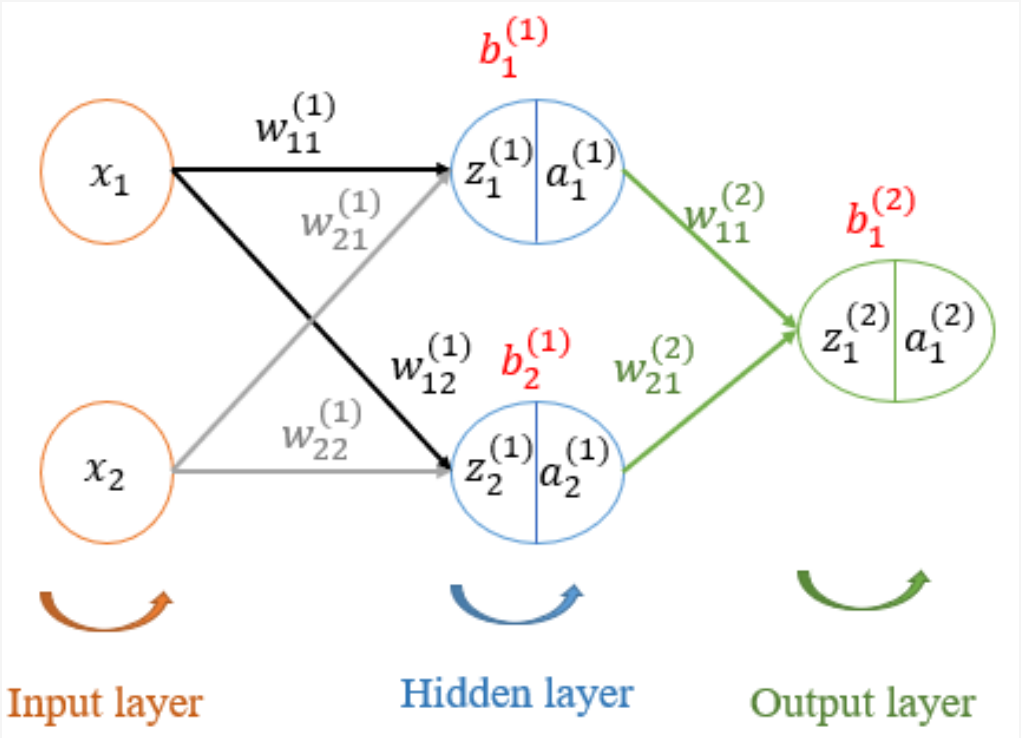
- $w_{11}^{(1)}, w_{12}^{(1)}, w_{21}^{(1)}, w_{22}^{(1)}, w_{11}^{(2)}, w_{21}^{(2)}$
- $b_1^{(1)}, b_2^{(1)}, b_1^{(2)}$



Tính đạo hàm L theo **w** và theo **b**

- Tại nơ-ron ở tầng đầu ra, ta sẽ tính

$$\frac{\partial L}{\partial w_{11}^{(2)}}, \frac{\partial L}{\partial w_{21}^{(2)}}, \frac{\partial L}{\partial b_1^{(2)}}.$$

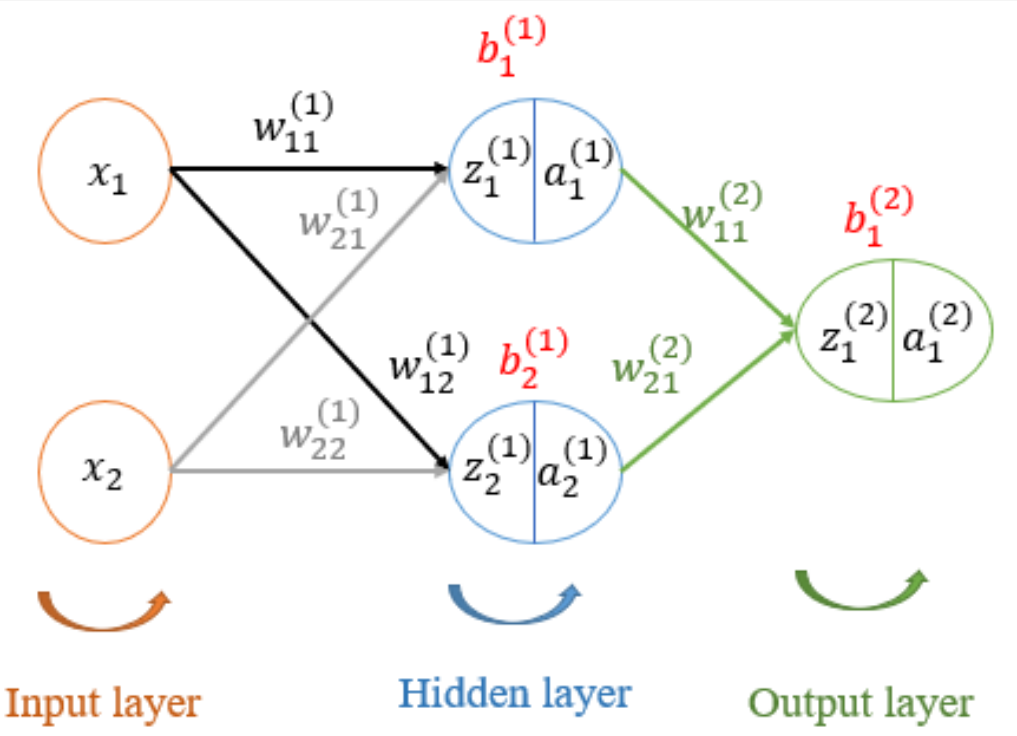


Tính đạo hàm L theo $\mathbf{w}$ và theo $\mathbf{b}$

Ta có:

$$\begin{aligned}
\frac{\partial L}{\partial w_{11}^{(2)}} &= \frac{1}{m}\sum_{i=1}^{m}\frac{\partial J}{\partial w_{11}^{(2)}}(\widehat{y}_i, y_i) \\
\frac{\partial L}{\partial w_{21}^{(2)}} &= \frac{1}{m}\sum_{i=1}^{m}\frac{\partial J}{\partial w_{21}^{(2)}}(\widehat{y}_i, y_i) \\
\frac{\partial L}{\partial b_{1}^{(2)}} &= \frac{1}{m}\sum_{i=1}^{m}\frac{\partial J}{\partial b_{1}^{(2)}}(\widehat{y}_i, y_i)
\end{aligned} \tag{1.6}$$

Tính đạo hàm L theo **w** và theo **b**



Bởi vì

$$J(\widehat{y}_i, y_i) = -\left(y_i \ln(\widehat{y}_i) + (1 - y_i) \ln(1 - \widehat{y}_i)\right)$$

$$\widehat{y}_i = a_1^{(2)} = f(z_1^{(2)}) = \frac{1}{1 + e^{-z_1^{(2)}}}$$

$$z_1^{(2)} = a_1^{(1)} w_{11}^{(2)} + a_2^{(1)} w_{21}^{(2)} + b_1^{(2)}$$

Tính đạo hàm L theo $\mathbf{w}$ và theo $\mathbf{b}$

nên áp dụng quy tắc dây chuyền kết hợp với (1.4) và (1.5), ta có

$$\begin{aligned}\frac{\partial J}{\partial w_{11}^{(2)}} &= \frac{\partial J}{\partial \widehat{y}_i} \times \frac{\partial \widehat{y}_i}{\partial z_1^{(2)}} \times \frac{\partial z_1^{(2)}}{\partial w_{11}^{(2)}} \\ &= \left(-\frac{y_i}{\widehat{y}_i} + \frac{1-y_i}{1-\widehat{y}_i}\right) \times \widehat{y}_i(1-\widehat{y}_i) \times a_1^{(1)} \\ &= (\widehat{y}_i - y_i)a_1^{(1)}\end{aligned} \tag{1.7}$$

Tương tự, ta có:

$$\begin{aligned}\frac{\partial J}{\partial w_{21}^{(2)}} &= \frac{\partial J}{\partial \widehat{y}_i} \times \frac{\partial \widehat{y}_i}{\partial z_1^{(2)}} \times \frac{\partial z_1^{(2)}}{\partial w_{21}^{(2)}} \\ &= (\widehat{y}_i - y_i)a_2^{(1)} \\ \frac{\partial J}{\partial b_1^{(2)}} &= \frac{\partial J}{\partial \widehat{y}_i} \times \frac{\partial \widehat{y}_i}{\partial z_1^{(2)}} \times \frac{\partial z_1^{(2)}}{\partial b_1^{(2)}} \\ &= \widehat{y}_i - y_i.\end{aligned} \tag{1.8}$$

Tính đạo hàm L theo $\mathbf{w}$ và theo $\mathbf{b}$

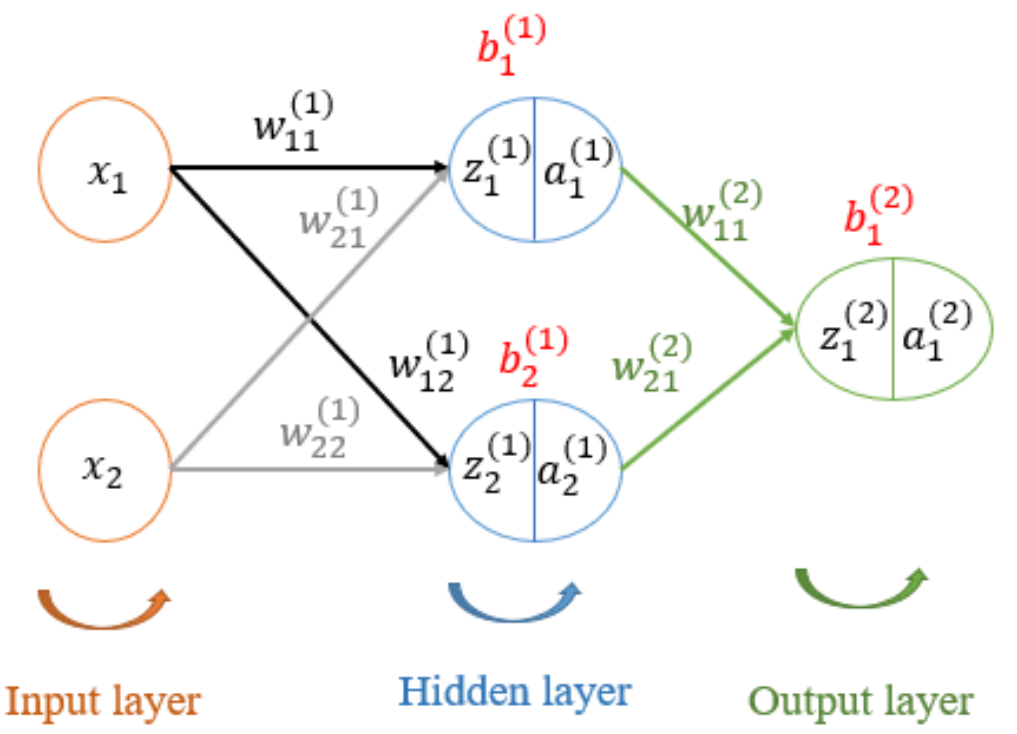
Thay (1.7) và (1.8) vào (1.6), ta được

$$\begin{aligned}
\frac{\partial L}{\partial w_{11}^{(2)}} &= \frac{a_1^{(1)}}{m} \sum_{i=1}^{m} \widehat{y}_i - y_i \\
\frac{\partial L}{\partial w_{21}^{(2)}} &= \frac{a_2^{(1)}}{m} \sum_{i=1}^{m} \widehat{y}_i - y_i \\
\frac{\partial L}{\partial b_1^{(2)}} &= \frac{1}{m} \sum_{i=1}^{m} \widehat{y}_i - y_i
\end{aligned} \tag{1.9}$$

Tính đạo hàm L theo **w** và theo **b**

- Tại nơ-ron thứ nhất của tầng ẩn, ta sẽ tính

$$\frac{\partial L}{\partial w_{11}^{(1)}}, \frac{\partial L}{\partial w_{21}^{(1)}}, \frac{\partial L}{\partial b_1^{(1)}}$$

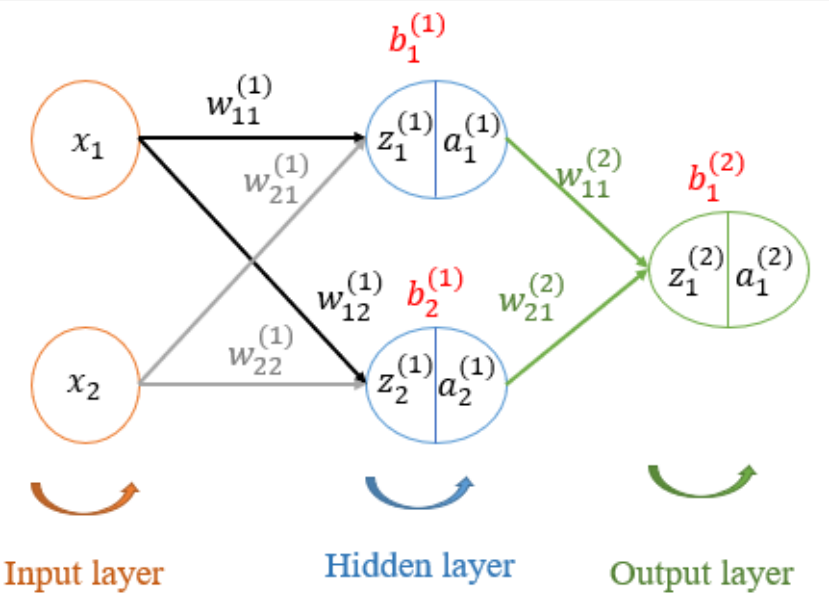


Tính đạo hàm L theo $\mathbf{w}$ và theo $\mathbf{b}$

Ta có:

$$\begin{aligned}
\frac{\partial L}{\partial w_{11}^{(1)}} &= \frac{1}{m}\sum_{i=1}^{m}\frac{\partial J}{\partial w_{11}^{(1)}}(\widehat{y}_i, y_i) \\
\frac{\partial L}{\partial w_{21}^{(1)}} &= \frac{1}{m}\sum_{i=1}^{m}\frac{\partial J}{\partial w_{21}^{(1)}}(\widehat{y}_i, y_i) \\
\frac{\partial L}{\partial b_{1}^{(1)}} &= \frac{1}{m}\sum_{i=1}^{m}\frac{\partial J}{\partial b_{1}^{(1)}}(\widehat{y}_i, y_i)
\end{aligned} \tag{1.10}$$

Tính đạo hàm L theo **w** và theo **b**



Bởi vì

$$J(\widehat{y}_i, y_i) = -\left(y_i \ln(\widehat{y}_i) + (1 - y_i) \ln(1 - \widehat{y}_i)\right)$$

$$\widehat{y}_i = a_1^{(2)} = f(z_1^{(2)}) = \frac{1}{1 + e^{-z_1^{(2)}}}$$

$$z_1^{(2)} = a_1^{(1)} w_{11}^{(2)} + a_2^{(1)} w_{21}^{(2)} + b_1^{(2)}$$

$$a_1^{(1)} = f(z_1^{(1)}) = \frac{1}{1 + e^{-z_1^{(1)}}}$$

$$z_1^{(1)} = x_1 w_{11}^{(1)} + x_2 w_{21}^{(1)} + b_1^{(1)}$$

Tính đạo hàm L theo **w** và theo **b**

nên áp dụng quy tắc dây chuyền kết hợp với (1.4) và (1.5), ta có

$$\begin{aligned}\frac{\partial J}{\partial w_{11}^{(1)}} &= \frac{\partial J}{\partial \widehat{y}_i} \times \frac{\partial \widehat{y}_i}{\partial z_1^{(2)}} \times \frac{\partial z_1^{(2)}}{\partial a_1^{(1)}} \times \frac{\partial a_1^{(1)}}{\partial z_1^{(1)}} \times \frac{\partial z_1^{(1)}}{\partial w_{11}^{(1)}} \\ &= \left(-\frac{y_i}{\widehat{y}_i} + \frac{1-y_i}{1-\widehat{y}_i}\right) \times (\widehat{y}_i(1-\widehat{y}_i)) \times w_{11}^{(2)} \times a_1^{(1)}(1-a_1^{(1)}) \times x_1 \\ &= (\widehat{y}_i - y_i) w_{11}^{(2)} a_1^{(1)}(1-a_1^{(1)}) x_1\end{aligned} \tag{1.11}$$

Tương tự, ta có

$$\begin{aligned}\frac{\partial J}{\partial w_{21}^{(1)}} &= \frac{\partial J}{\partial \widehat{y}_i} \times \frac{\partial \widehat{y}_i}{\partial z_1^{(2)}} \times \frac{\partial z_1^{(2)}}{\partial a_1^{(1)}} \times \frac{\partial a_1^{(1)}}{\partial z_1^{(1)}} \times \frac{\partial z_1^{(1)}}{\partial w_{21}^{(1)}} \\ &= (\widehat{y}_i - y_i) w_{11}^{(2)} a_1^{(1)}(1-a_1^{(1)}) x_2 \\ \frac{\partial J}{\partial b_1^{(1)}} &= \frac{\partial J}{\partial \widehat{y}_i} \times \frac{\partial \widehat{y}_i}{\partial z_1^{(2)}} \times \frac{\partial z_1^{(2)}}{\partial a_1^{(1)}} \times \frac{\partial a_1^{(1)}}{\partial z_1^{(1)}} \times \frac{\partial z_1^{(1)}}{\partial b_1^{(1)}} \\ &= (\widehat{y}_i - y_i) w_{11}^{(2)} a_1^{(1)}(1-a_1^{(1)}).\end{aligned} \tag{1.12}$$

Tính đạo hàm L theo **w** và theo **b**

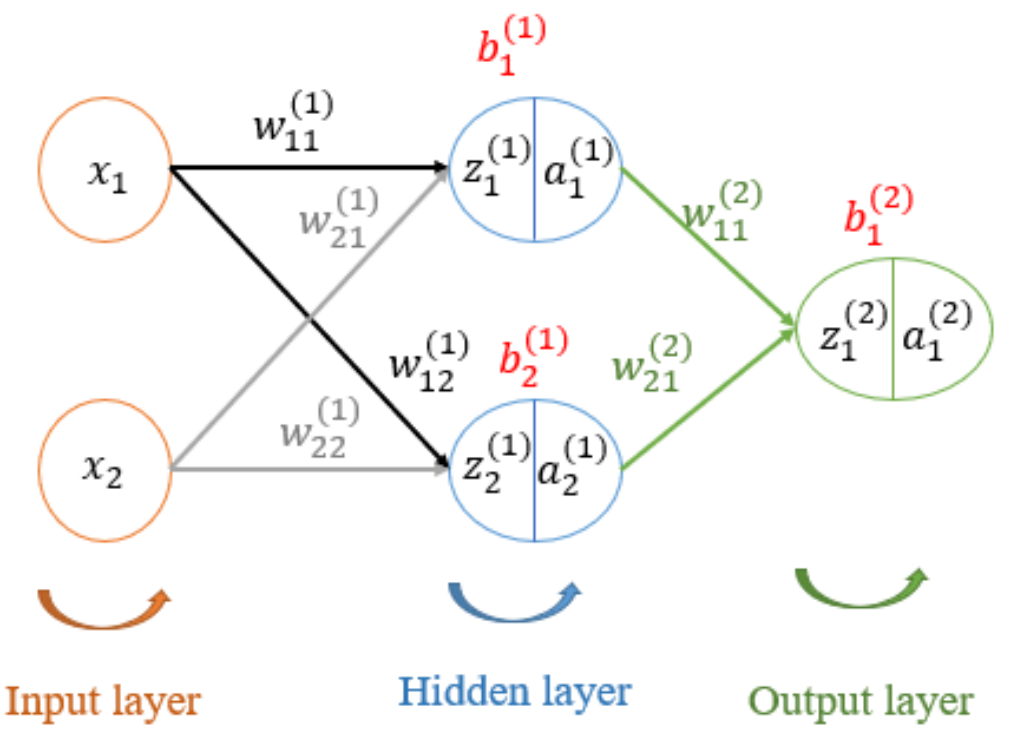
Thay (1.11) và (1.12) vào (1.10), ta được

$$\begin{aligned}
\frac{\partial L}{\partial w_{11}^{(1)}} &= w_{11}^{(2)} a_1^{(1)}(1-a_1^{(1)})x_1 \times \frac{1}{m}\sum_{i=1}^{m}\widehat{y}_i - y_i \\
\frac{\partial L}{\partial w_{21}^{(1)}} &= w_{11}^{(2)} a_1^{(1)}(1-a_1^{(1)})x_2 \times \frac{1}{m}\sum_{i=1}^{m}\widehat{y}_i - y_i \qquad (1.13) \\
\frac{\partial L}{\partial b_1^{(1)}} &= w_{11}^{(2)} a_1^{(1)}(1-a_1^{(1)}) \times \frac{1}{m}\sum_{i=1}^{m}\widehat{y}_i - y_i
\end{aligned}$$

Tính đạo hàm L theo **w** và theo **b**

- Tại nơ-ron thứ hai của tầng ẩn, ta sẽ tính

$$\frac{\partial L}{\partial w_{12}^{(1)}}, \frac{\partial L}{\partial w_{22}^{(1)}}, \frac{\partial L}{\partial b_2^{(1)}}$$

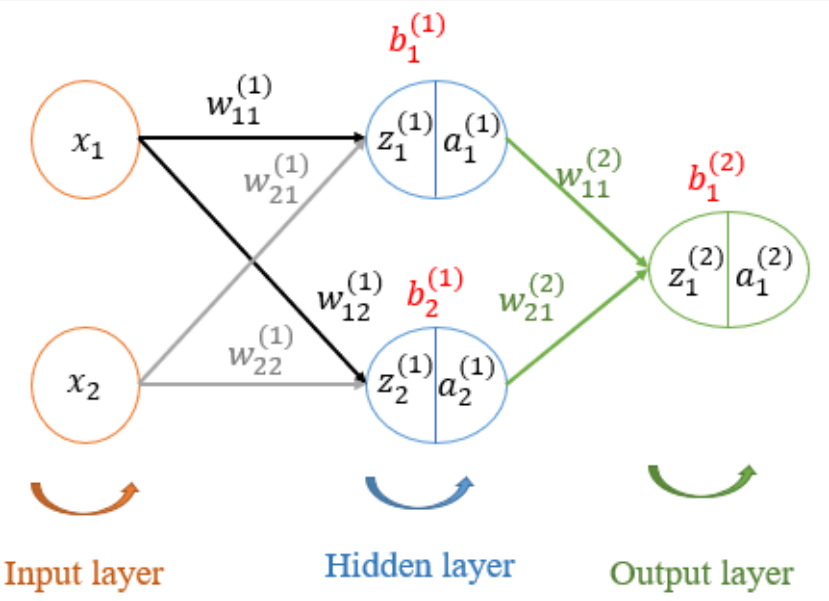


Tính đạo hàm L theo $\mathbf{w}$ và theo $\mathbf{b}$

Ta có:

$$\begin{aligned}
\frac{\partial L}{\partial w_{12}^{(1)}} &= \frac{1}{m}\sum_{i=1}^{m}\frac{\partial J}{\partial w_{12}^{(1)}}(\widehat{y}_i, y_i) \\
\frac{\partial L}{\partial w_{22}^{(1)}} &= \frac{1}{m}\sum_{i=1}^{m}\frac{\partial J}{\partial w_{22}^{(1)}}(\widehat{y}_i, y_i) \\
\frac{\partial L}{\partial b_{2}^{(1)}} &= \frac{1}{m}\sum_{i=1}^{m}\frac{\partial J}{\partial b_{2}^{(1)}}(\widehat{y}_i, y_i)
\end{aligned} \tag{1.14}$$

Tính đạo hàm L theo **w** và theo **b**



Bởi vì

$$J(\widehat{y}_i, y_i) = -\left(y_i \ln(\widehat{y}_i) + (1 - y_i) \ln(1 - \widehat{y}_i)\right)$$

$$\widehat{y}_i = a_1^{(2)} = f(z_1^{(2)}) = \frac{1}{1 + e^{-z_1^{(2)}}}$$

$$z_1^{(2)} = a_1^{(1)} w_{11}^{(2)} + a_2^{(1)} w_{21}^{(2)} + b_1^{(2)}$$

$$a_2^{(1)} = f(z_2^{(1)}) = \frac{1}{1 + e^{-z_2^{(1)}}}$$

$$z_2^{(1)} = x_1 w_{12}^{(1)} + x_2 w_{22}^{(1)} + b_2^{(1)}$$

Tính đạo hàm L theo $\mathbf{w}$ và theo $\mathbf{b}$

nên áp dụng quy tắc dây chuyền kết hợp với (1.4) và (1.5), ta có

$$\begin{aligned}\frac{\partial J}{\partial w_{12}^{(1)}} &= \frac{\partial J}{\partial \widehat{y}_i} \times \frac{\partial \widehat{y}_i}{\partial z_1^{(2)}} \times \frac{\partial z_1^{(2)}}{\partial a_2^{(1)}} \times \frac{\partial a_2^{(1)}}{\partial z_2^{(1)}} \times \frac{\partial z_2^{(1)}}{\partial w_{12}^{(1)}} \\ &= \left(-\frac{y_i}{\widehat{y}_i} + \frac{1-y_i}{1-\widehat{y}_i}\right) \times (\widehat{y}_i(1-\widehat{y}_i)) \times w_{21}^{(2)} \times a_2^{(1)}(1-a_2^{(1)}) \times x_1 \\ &= (\widehat{y}_i - y_i) w_{21}^{(2)} a_2^{(1)}(1-a_2^{(1)}) x_1\end{aligned} \tag{1.15}$$

Tương tự, ta có

$$\begin{aligned}\frac{\partial J}{\partial w_{22}^{(1)}} &= \frac{\partial J}{\partial \widehat{y}_i} \times \frac{\partial \widehat{y}_i}{\partial z_1^{(2)}} \times \frac{\partial z_1^{(2)}}{\partial a_2^{(1)}} \times \frac{\partial a_2^{(1)}}{\partial z_2^{(1)}} \times \frac{\partial z_2^{(1)}}{\partial w_{22}^{(1)}} \\ &= (\widehat{y}_i - y_i) w_{21}^{(2)} a_2^{(1)}(1-a_2^{(1)}) x_2 \\ \frac{\partial J}{\partial b_2^{(1)}} &= \frac{\partial J}{\partial \widehat{y}_i} \times \frac{\partial \widehat{y}_i}{\partial z_1^{(2)}} \times \frac{\partial z_1^{(2)}}{\partial a_2^{(1)}} \times \frac{\partial a_2^{(1)}}{\partial z_2^{(1)}} \times \frac{\partial z_2^{(1)}}{\partial b_2^{(1)}} \\ &= (\widehat{y}_i - y_i) w_{21}^{(2)} a_2^{(1)}(1-a_2^{(1)}).\end{aligned} \tag{1.16}$$

Tính đạo hàm L theo $\mathbf{w}$ và theo $\mathbf{b}$

Thay (1.15) và (1.16) vào (1.14), ta được

$$\begin{aligned}
\frac{\partial L}{\partial w_{12}^{(1)}} &= w_{21}^{(2)} a_2^{(1)}(1-a_2^{(1)})x_1 \times \frac{1}{m}\sum_{i=1}^{m}(\widehat{y}_i - y_i) \\
\frac{\partial L}{\partial w_{22}^{(1)}} &= w_{21}^{(2)} a_2^{(1)}(1-a_2^{(1)})x_2 \times \frac{1}{m}\sum_{i=1}^{m}(\widehat{y}_i - y_i) \qquad (1.17) \\
\frac{\partial L}{\partial b_2^{(1)}} &= w_{21}^{(2)} a_2^{(1)}(1-a_2^{(1)}) \times \frac{1}{m}\sum_{i=1}^{m}(\widehat{y}_i - y_i).
\end{aligned}$$

Thuật toán Gradient descent

Sau khi tính đạo hàm L theo **w** và **b**, ta áp dụng thuật toán Gradient descent cho ví dụ trên với learning rate $\eta = 0.1$ và được kết quả sau

Step	$w_{11}^{(1)}$	$w_{21}^{(1)}$	$b_1^{(1)}$	$w_{12}^{(1)}$	$w_{22}^{(1)}$	$b_2^{(1)}$	$w_{11}^{(2)}$	$w_{21}^{(2)}$	$b_1^{(2)}$	L
0	0.16	-0.07	0.37	-0.66	0.63	-0.98	0.41	-0.9	0.35	0.713
1	0.16	-0.07	0.37	-0.66	0.63	-0.98	0.4	-0.9	0.34	0.712
2	0.16	-0.07	0.37	-0.66	0.63	-0.98	0.4	-0.9	0.33	0.711
3	0.16	-0.07	0.37	-0.66	0.63	-0.98	0.39	-0.9	0.33	0.710
4	0.16	-0.07	0.37	-0.65	0.63	-0.97	0.39	-0.91	0.32	0.709
5	0.16	-0.07	0.37	-0.65	0.63	-0.97	0.39	-0.91	0.31	0.708
...	...	...	...	...	...	...	...	...	...	...
100000	7.69	7.69	-3.55	6.34	6.34	-9.68	14.74	-15.39	-7.02	0.001

Thuật toán Gradient descent

Bây giờ , ta sẽ dùng mạng nơ-ron sau khi đã huấn luyện xong cho từng giá trị đầu vào của ví dụ trước để kiểm tra xem giá trị đầu ra của nó có gần với giá trị mình mong muốn hay không?

x_1	x_2	$w_{11}^{(1)}$	$w_{21}^{(1)}$	$b_1^{(1)}$	$w_{12}^{(1)}$	$w_{22}^{(1)}$	$b_2^{(1)}$	$z_1^{(1)}$	$a_1^{(1)}$
0	0	7.69	7.69	-3.55	6.34	6.34	-9.68	-3.55	0.03
0	1	7.69	7.69	-3.55	6.34	6.34	-9.68	4.14	0.98
1	0	7.69	7.69	-3.55	6.34	6.34	-9.68	4.14	0.98
1	1	7.69	7.69	-3.55	6.34	6.34	-9.68	11.83	1.00

$a_2^{(1)}$	$w_{11}^{(2)}$	$w_{21}^{(2)}$	$b_1^{(2)}$	$z_1^{(2)}$	$\widehat{y}_i = a_1^{(2)}$	y_i	J
0.00	14.74	-15.39	-7.02	-6.61	0.001346	0	0.006
0.03	14.74	-15.39	-7.02	6.96	0.999054	1	0.0004
0.03	14.74	-15.39	-7.02	6.96	0.999054	1	0.0004
0.95	14.74	-15.39	-7.02	-6.94	0.000967	0	0.0004

Vì vậy, ta được $L = 0.0005$.

Bài tập

Bài tập: Dùng Python để mô phỏng lại quá trình huấn luyện mạng nơ-ron trên.