Chương 9. Neural Network

TRÍ TUỆ NHÂN TẠO

Artificial Intelligence

Đoàn Vũ Thịnh
Khoa Công nghệ Thông tin
Đại học Nha Trang
Email: thinhdv@ntu.edu.vn

Nha Trang, 06-2023

Chương 9. Neural Network

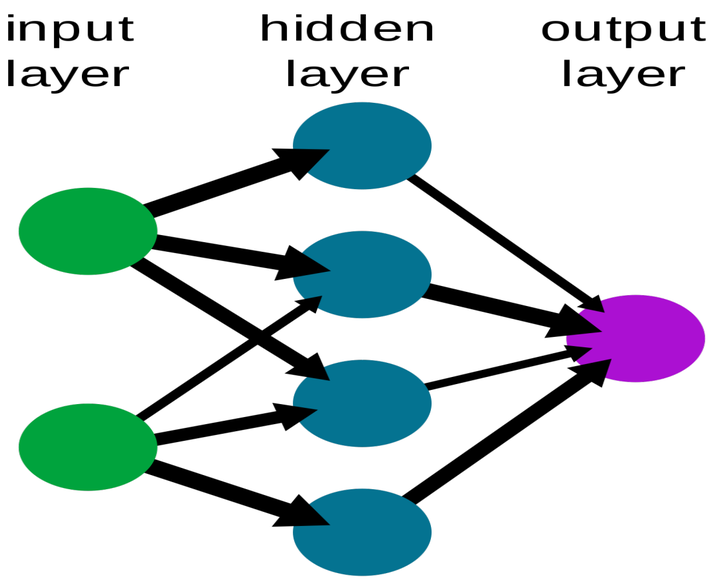
Khái niệm

- Mạng nơ-ron là một mô hình máy học sử dụng một mạng lưới các chức năng để hiểu và dịch dữ liệu đầu vào ở dạng này thành dạng khác ở đầu mong muốn.
- Mạng nơ-ron nhân tạo lấy cảm hứng từ cách hoạt động của hệ thần kinh sinh học trong não người.
- Mục tiêu của mạng nơ-ron là mô phỏng khả năng học và tìm hiểu thông qua việc tự điều chỉnh dự đoán dựa trên dữ liệu đầu vào.

Chương 9. Neural Network

Mô hình mạng nơ-ron

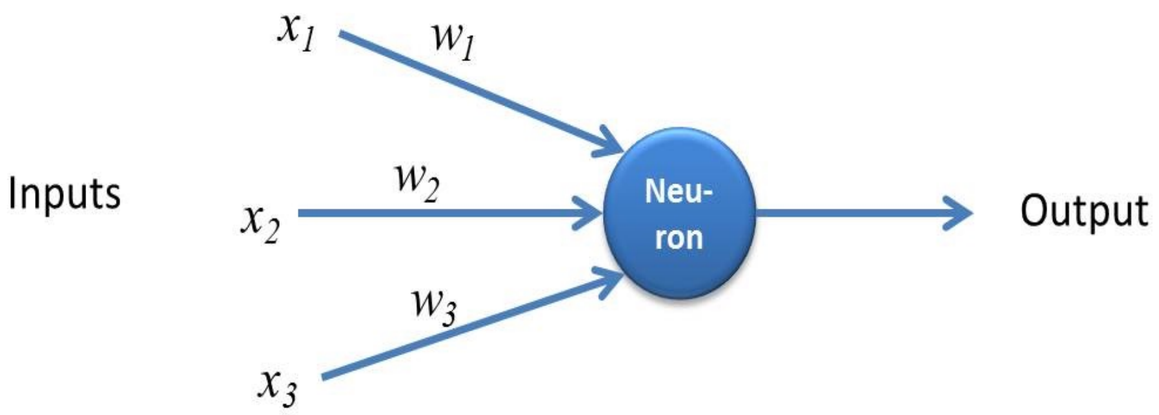
- Cấu trúc mạng nơ-ron có 3 lớp như sau:
- Lớp đầu vào: Thông tin bên ngoài được đưa vào mô hình để học và rút ra kết luận. Các nút đầu vào chuyển thông tin sang lớp lớp ẩn.
- Lớp ẩn: Nơi tất cả các tính toán được thực hiện trên dữ liệu đầu vào. Số lượng lớp ẩn không cố định cho các bài toán khác nhau. Mạng đơn giản nhất bao gồm một lớp ẩn duy nhất.



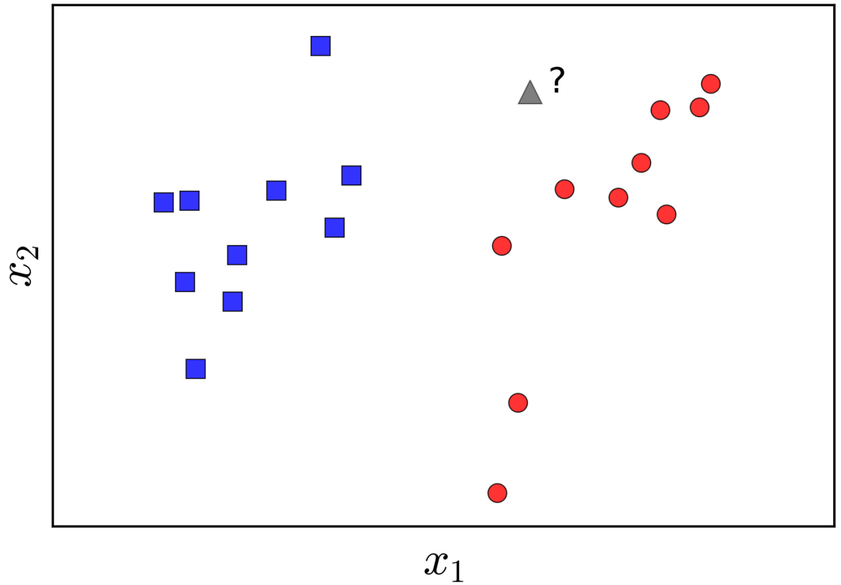
- Lớp đầu ra: Kết luận của mô hình, có thể có một hoặc nhiều nút ở đầu ra. Khi bài toán thuộc lớp phân loại nhị phân, đầu ra có 1 node và trong trường hợp phân loại nhiều lớp, các node đầu ra có thể nhiều hơn 1.

Giải thuật PLA (Perceptron Learning Algorithm)

- Perceptron (Frank Rosenblatt, 1950) là một mô hình neural network đơn giản, được sử dụng để thực hiện một tác vụ nhị phân (bài toán với chỉ hai class được gọi là binary classification).



- Perceptron Learning Algorithm được sử dụng để huấn luyện mô hình phân loại nhị phân, tức đi tìm một siêu phẳng (hyperplane) phân cách giữa hai lớp dữ liệu, tức là phân tách chúng thành hai phần.

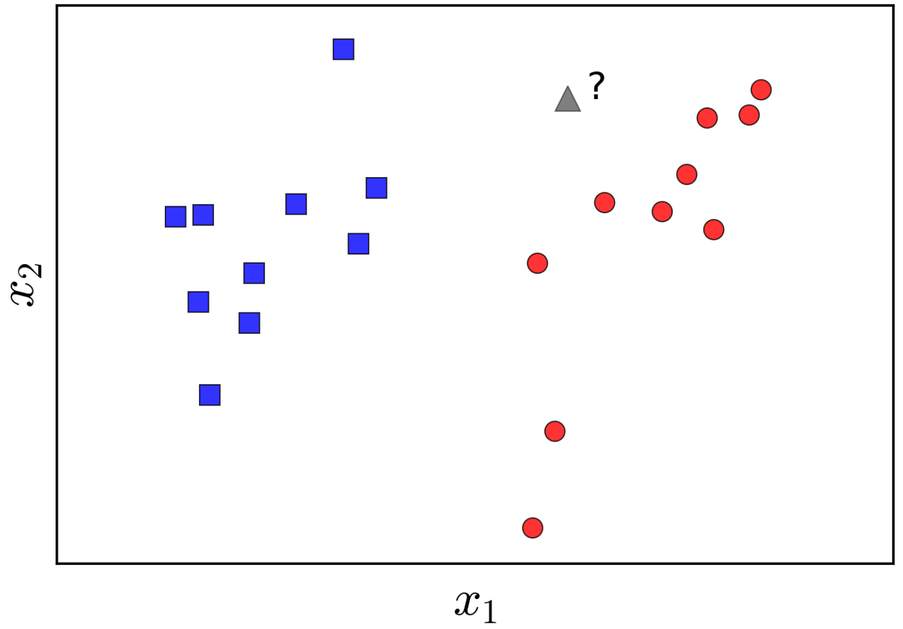


Giải thuật PLA (Perceptron Learning Algorithm)

- Hiểu theo một cách khác, chúng ta cần tìm lãnh thổ của mỗi class sao cho, với mỗi một điểm mới, ta chỉ cần xác định xem nó nằm vào lãnh thổ của class nào rồi quyết định nó thuộc class đó.
- Để tìm lãnh thổ của mỗi class, chúng ta cần đi tìm biên giới (boundary) giữa hai lãnh thổ này. Và boundary đơn giản nhất trong không gian 2 chiều là một đường thẳng, trong không gian ba chiều là một mặt phẳng, trong không gian nhiều chiều là một siêu mặt phẳng (hyperplane) (gọi chung là đường phẳng).
- Những boundary phẳng này khá đơn giản vì nó có thể biểu diễn dưới dạng toán học bằng một hàm số đơn giản có dạng tuyến tính (linear).

Giải thuật PLA (Perceptron Learning Algorithm)

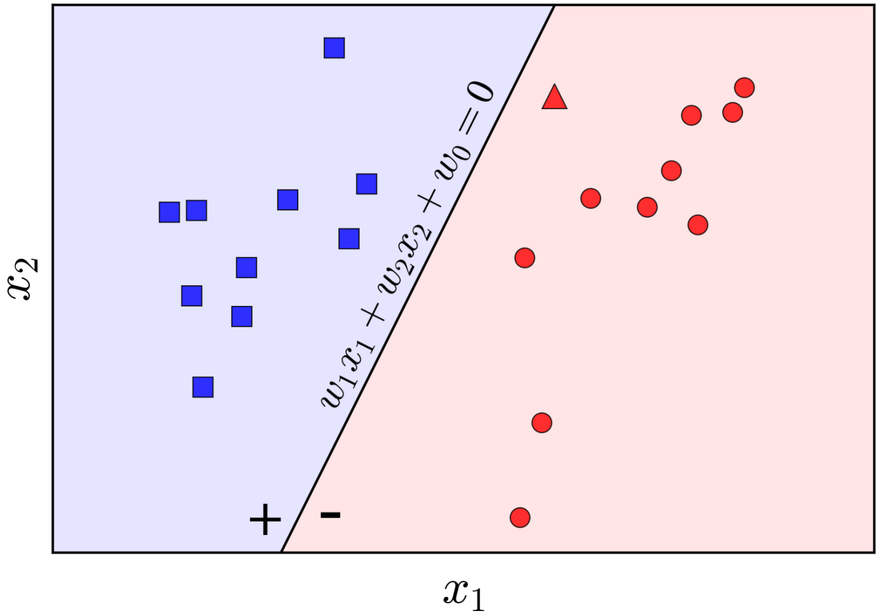
- Giả sử chúng ta có hai lớp dữ liệu, lớp 0 và lớp 1, và mỗi điểm dữ liệu được biểu diễn bằng hai đặc trưng (features) x1 và x2. Mục tiêu của chúng ta là tìm một đường thẳng sao cho nó có thể phân tách hai lớp dữ liệu.
- Đường thẳng này có dạng: $x_2 w_2 + x_1 w_1 + b = 0$



- Với w_1, w_2 là các trọng số đặc trưng, còn được b gọi là ngưỡng.
- Nếu một điểm dữ liệu (x_1, x_2) nằm phía trên đường thẳng $(x_1 w_1 + x_2 w_2 + b > 0)$ – nó thuộc về lớp 1; nếu nó nằm bên dưới đường thẳng $(x_1 w_1 + x_2 w_2 + b < 0)$ – nó thuộc về lớp 2.

Giải thuật PLA (Perceptron Learning Algorithm)

- Thuật toán PLA hoạt động như sau:
- 1. Khởi tạo ngẫu nhiên các trọng số: w_1, w_2, b hoặc đặt chúng bằng 0.
- 2. Duyệt qua tất cả các điểm dữ liệu huấn luyện.
- 3. Đối với mỗi điểm dữ liệu, nếu điểm được phân lớp sai (nằm sai bên kia của đường thẳng), cập nhật trọng số và bias theo cách sao cho đường thẳng phân tách đúng điểm đó.
- 4. Lặp lại bước 2 và 3 cho đến khi không còn điểm dữ liệu nào được phân lớp sai hoặc sau một số lần lặp tối đa.



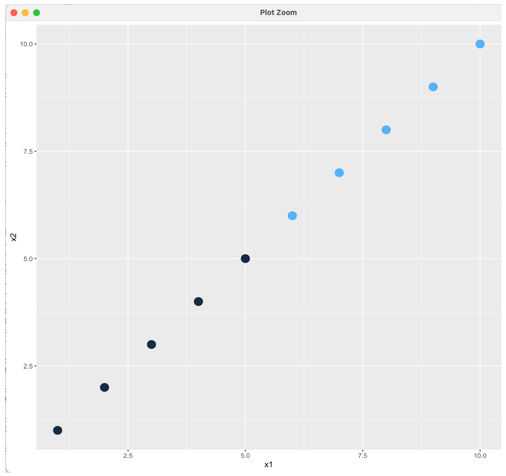
$$w_1 \leftarrow w_1 + \propto . x_1$$
$$w_2 \leftarrow w_2 + \propto . x_2$$
$$b \leftarrow b + \propto$$

Ở đây, α là tốc độ học (learning rate) là một tham số được đặt trước.

Chương 9. Neural Network

- Giải thuật PLA (Perceptron Learning Algorithm)

x1	x2	Label
1	1	0
2	2	0
3	3	0
4	4	0
5	5	0
6	6	1
7	7	1
8	8	1
9	9	1
10	10	1




```
// Ham tinh du doan dau ra cua du lieu
int predict(double w1, double w2, double b, DataPoint data)
{
  double result = w1 * data.x1 + w2 * data.x2 + b;
  return result > 0 ? 1 : 0;
}

void trainPLA(double *w1, double *w2, double *b, DataPoint *data, int numData, double learningRate)
{
  int converged = 0;
  int count = 0;
  while (converged==0) {//chua hoi tu
    converged = 1;
    for (int i = 0; i < numData; i++) {
      int predicted = predict(*w1, *w2, *b, data[i]);
      if (predicted != data[i].label) {
        *w1 += learningRate * (data[i].label - predicted) * data[i].x1;
        *w2 += learningRate * (data[i].label - predicted) * data[i].x2;
        *b += learningRate * (data[i].label - predicted);
        converged = 0;
      }
    }
    count++;
  }
  printf("Giai thuat PLA ket thuc sau %d buoc\n", count);
}
```

Chương 9. Neural Network

- Giải thuật PLA (Perceptron Learning Algorithm)

x1	x2	Label	Pred	w1	w2	b
1	1	0	0	0	0	0
2	2	0	0	0	0	0
3	3	0	0	0	0	0
4	4	0	0	0	0	0
5	5	0	0	0	0	0
6	6	1	0	0.6	0.6	0.1
7	7	1	1(8.5)	0.6	0.6	0.1
8	8	1	1	0.6	0.6	0.1
9	9	1	1	0.6	0.6	0.1
10	10	1	1	0.6	0.6	0.1
count=1						

$w1 = 0,\ w2 = 0,\ b = 0,\ \alpha = 0.1$

```
if (predicted != data[i].label) {
 *w1 += learningRate * (data[i].label - predicted) * data[i].x1;
 *w2 += learningRate * (data[i].label - predicted) * data[i].x2;
 *b += learningRate * (data[i].label - predicted);
 converged = 0;
}

int predict(double w1, double w2, double b, DataPoint data)
{
  double result = w1 * data.x1 + w2 * data.x2 + b;
  return result > 0 ? 1 : 0;
}
```

Chương 9. Neural Network

▪ Giải thuật PLA (Perceptron Learning Algorithm)

Dựa vào kết quả học ở trên ta có các hệ số của phương trình đường thẳng:

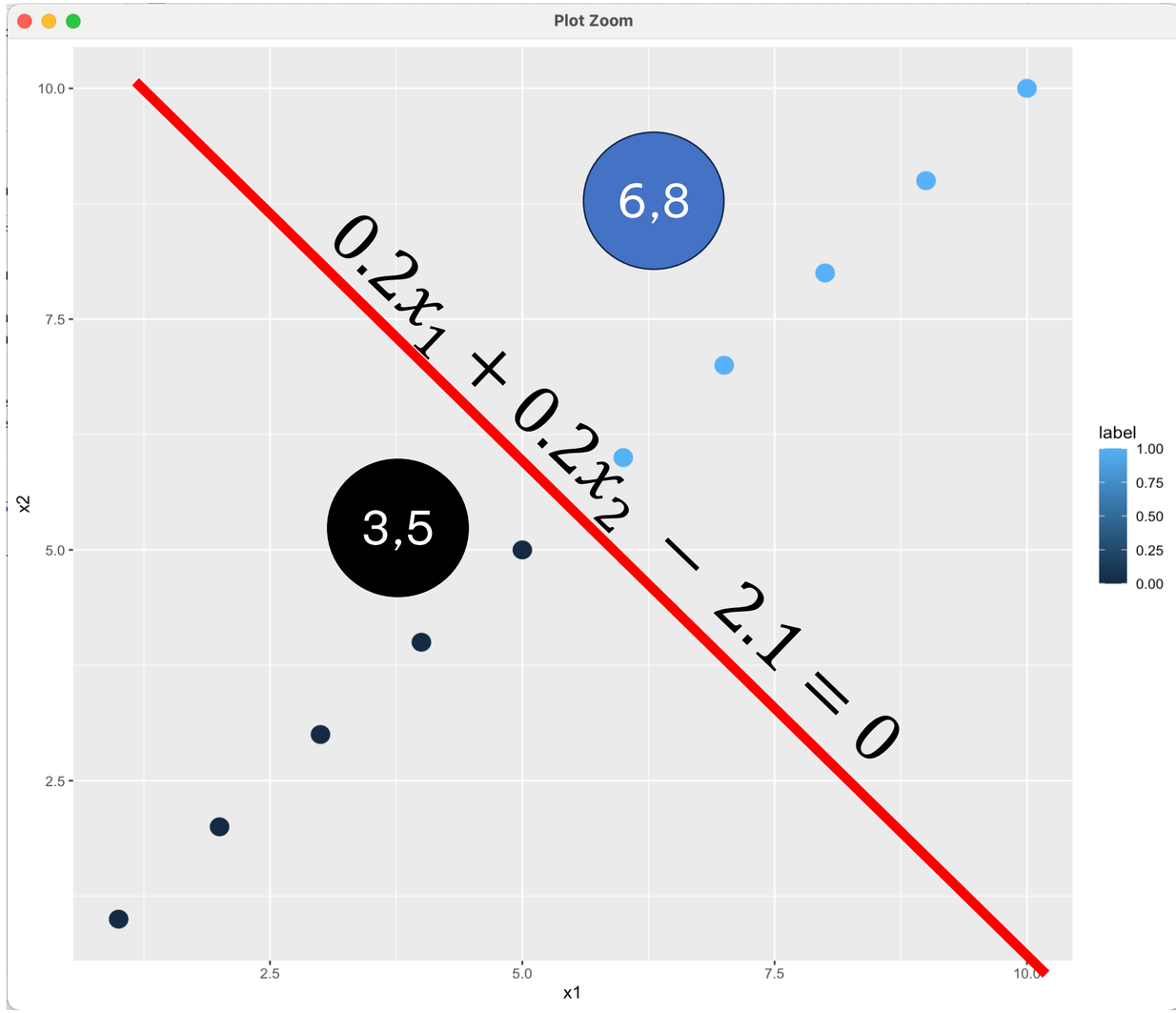
$$x_2 w_2 + x_1 w_1 + b = 0$$

Với: w1 = 0.2, w2 = 0.2, b = -2.1

Xét 1 điểm mới (x1,x2) = (6,8)
Thế vào PT trên ta có:
0.2*6 + 0.2*8 – 2.1 = 0.7 >0

```
int predict(double w1, double w2, double b, DataPoint data)
{
double result = w1 * data.x1 + w2 * data.x2 + b;
return result > 0 ? 1 : 0;
}
```

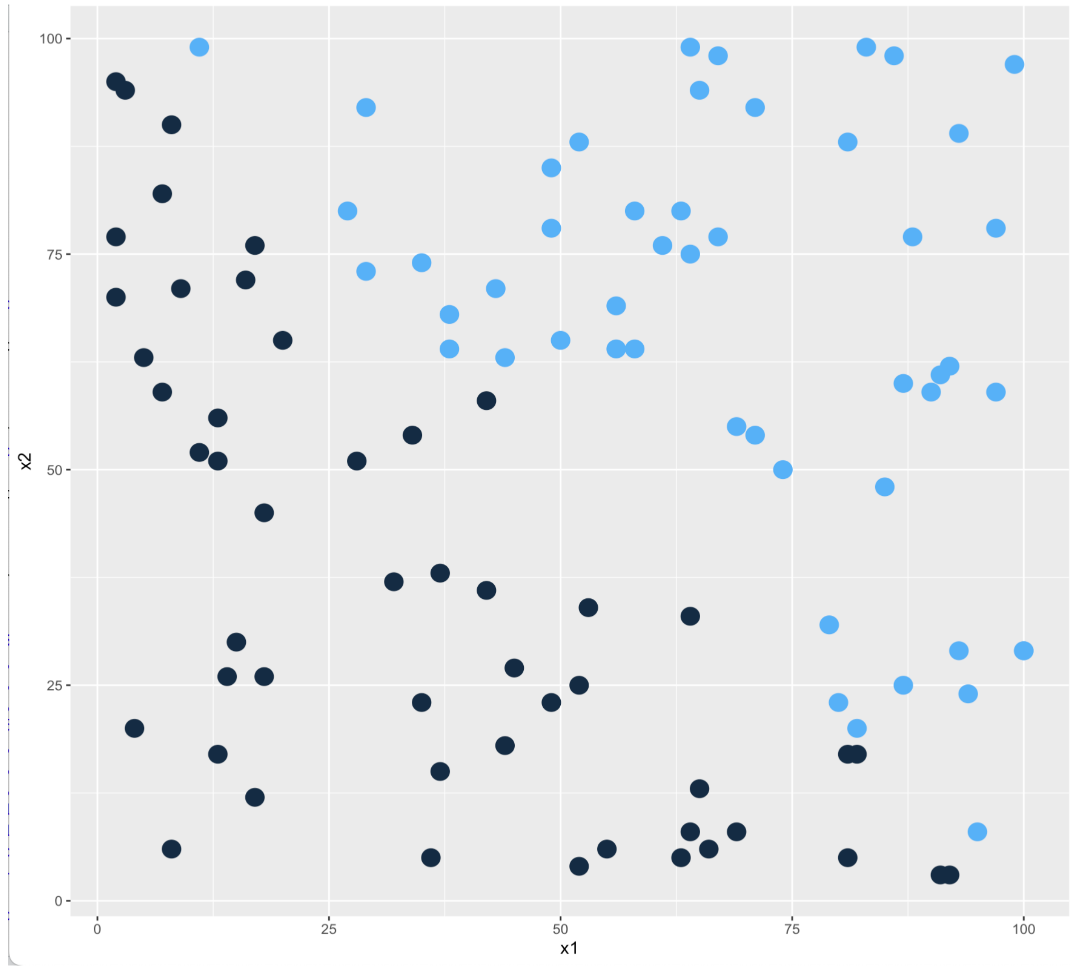
0.2*3 + 0.2*5 – 2.1 = -0.5 <0 ~ 0



Chương 9. Neural Network

- Giải thuật PLA (Perceptron Learning Algorithm)

X1=
2,2,2,3,4,5,7,7,8,8,9,11,13,13,13,14,15,16,17,17,18,18,20,28,32,34,35,36,37,37,42,42,44,4
5,49,52,52,53,55,63,64,64,65,66,69,81,81,82,91,92,11,27,29,29,35,38,38,43,44,49,49
,50,52,56,56,58,58,61,63,64,64,65,67,67,69,71,71,74,79,80,81,82,83,85,86,87,87,88,9
0,91,92,93,93,94,95,97,97,99,100,100
X2=
70,95,77,94,20,63,82,59,6,90,71,52,17,51,56,26,30,72,12,76,26,45,65,51,37,54,23,5,15
,38,58,36,18,27,23,25,4,34,6,5,33,8,13,6,8,5,17,17,3,3,99,80,73,92,74,68,64,71,63,85,7
8,65,88,64,69,64,80,76,80,75,99,94,77,98,55,54,92,50,32,23,88,20,99,48,98,60,25,
77,59,61,62,89,29,24,8,78,59,97,29,29
Label =
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1



Icon for R

Icon for RStudio

```
library(ggplot2)
df <- data.frame(x1=c(...), x2=c(...), label=c(...))
ggplot(df, aes(x1, x2)) + geom_point(aes(color=label))
```

Giai thuat PLA ket thuc sau 92 buoc
w1 = 1.300000, w2 = 1.400000, b = -132.800000

Chương 9. Neural Network

Giải thuật Multi-Layer Perceptron (MLP)

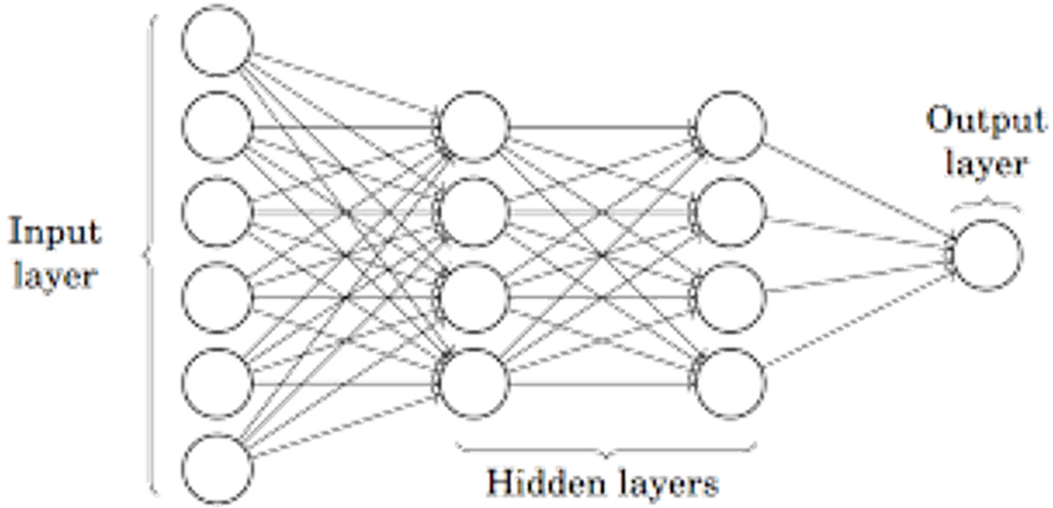
- MLP là một mô hình nơ-ron nhân tạo phức tạp hơn so với Perceptron.
- Nó bao gồm ít nhất hai lớp nơ-ron: lớp đầu vào, lớp đầu ra và một hoặc nhiều lớp ẩn giữa chúng.
- Các nơ-ron trong MLP được kết nối qua các trọng số, và mô hình này sử dụng các hàm kích hoạt phi tuyến tính (ví dụ: hàm sigmoid, hàm tanh, hàm ReLU) để biểu diễn các phức tạp hơn.
- MLP là một loại mạng nơ-ron feedforward, có khả năng học và biểu diễn các hàm phức tạp, bao gồm cả việc phân loại đa lớp và học giám sát.

Mạng nơ-ron feedforward (FFNN) là một loại mạng nơ-ron nhân tạo trong đó thông tin di chuyển từ lớp đầu vào đến lớp đầu ra mà không có chu trình hoặc kết nối phản hồi. Cụ thể, trong mạng nơ-ron feedforward, dữ liệu chỉ di chuyển theo một hướng, từ trước ra sau, đi qua các lớp nơ-ron trung gian (hidden layers) trước khi đến lớp đầu ra.

Chương 9. Neural Network

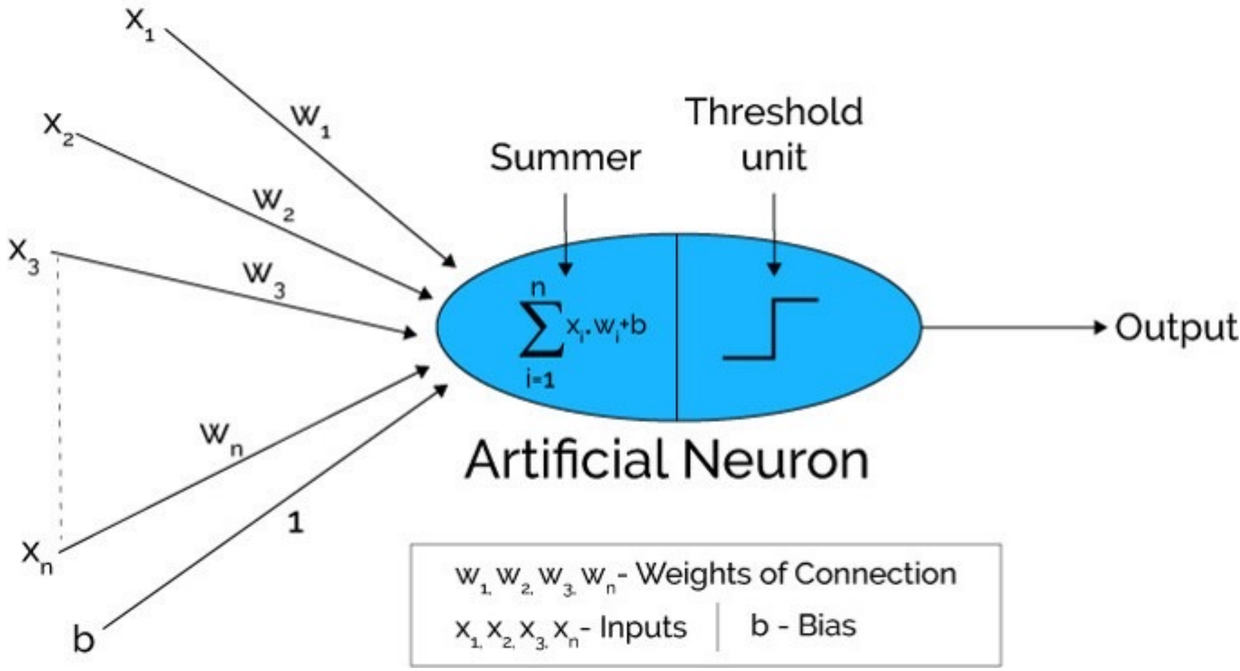
Giải thuật Multi-Layer Perceptron (MLP)

- MLP còn được gọi là Artificial Neural Networks (ANN)
- Trong hình trên, mạng ANN gồm có 4 lớp được kết nối với nhau:
  - 1 đầu vào với 6 node
  - Lớp ẩn 1 với 4 node ẩn /4 perceptron
  - Lớp ẩn 2 với 4 node ẩn
  - 1 node ở đầu ra



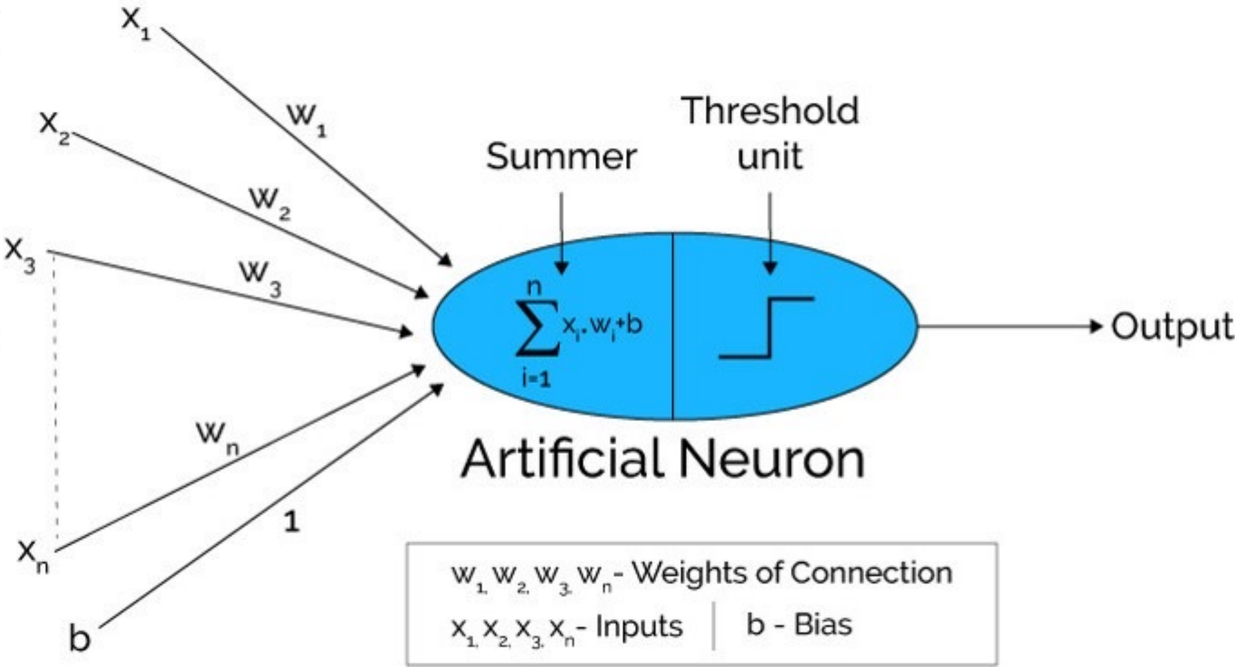
Nguyên lý hoạt động của ANN

- B1. Các đầu vào với các trọng số w_i được chuyển đến lớp ẩn
- B2. Mỗi lớp ẩn chứa số lượng neural bất kỳ. Mỗi nơ ron được kết nối với tất cả đầu vào tương ứng.
- B3. Tất cả các tính toán được thực hiện trong lớp ẩn.
- Việc tính toán trong lớp ẩn được thực hiện thông qua 2 bước sau:



Nguyên lý hoạt động của ANN

- B3.1. Nhân tất cả đầu vào với trọng số (w_i) – trọng số biểu diễn độ dốc (gradient) hoặc hệ số (coefficient) của từng biến. Giá trị ngưỡng (bias) được thêm vào biểu thức trên – ngưỡng là một hằng số giúp cho mô hình đạt được kết quả tốt nhất có thể.



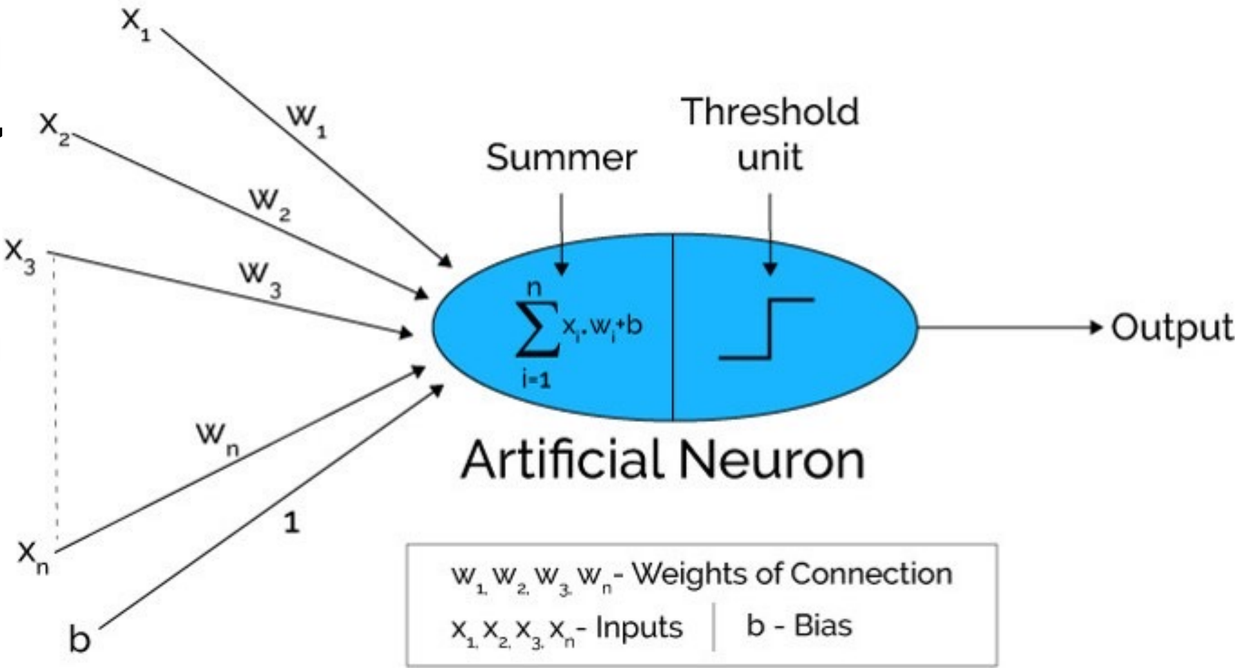
- $z = w_1 * x_1 + w_2 * x_2 + w_3 * x_3 + \cdots + b$
- Trong đó: w_i – trọng số của biến đầu vào

  x_i – biến đầu vào

  b – hệ số (giá trị ngưỡng)

Nguyên lý hoạt động của ANN

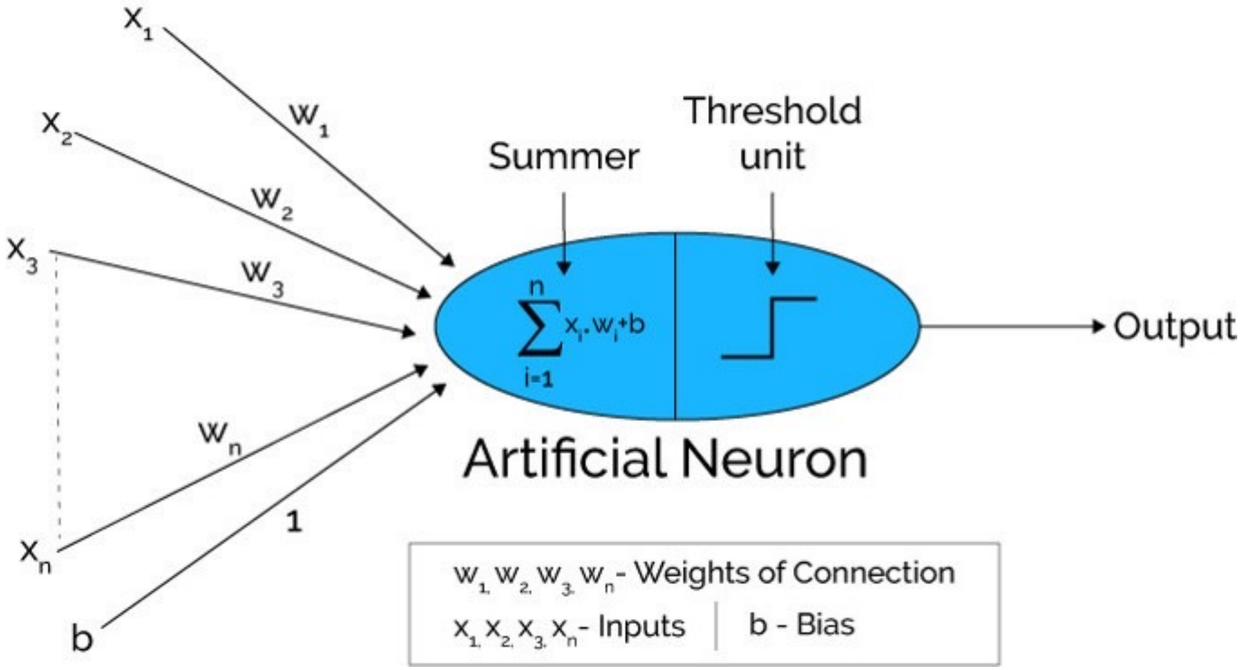
- B3.2. Một hàm kích hoạt được sử dụng cho phương trình tuyến tính z như sigmoid(x) hoặc tanh(x) tuỳ thuộc vào dữ liệu đầu vào, trước khi chuyển sang lớp tiếp theo.



Chương 9. Neural Network

Nguyên lý hoạt động của ANN

- B4. Sau khi tính toán hàm z trong từng lớp ẩn, chúng ta chuyển sang lớp cuối cùng (lớp đầu ra).
- B5. Sau khi nhận được dự đoán từ lớp đầu ra, giá trị sai số (sự khác biệt giữa đầu ra thực tế và đầu ra dự đoán) sẽ được tính toán để đánh giá hiệu quả của mô hình.

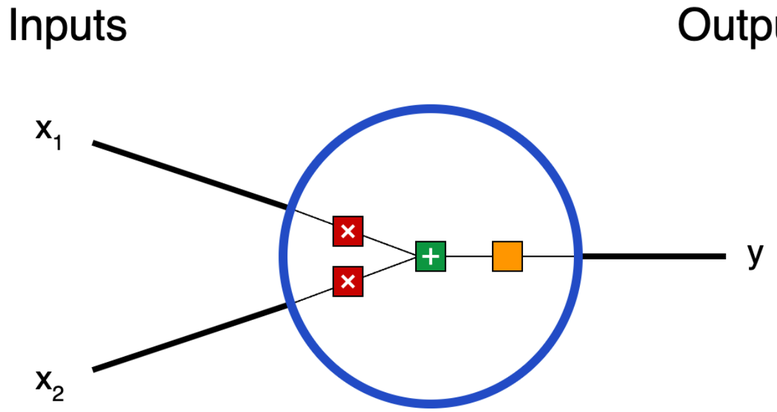


- Nếu giá trị sai số lớn, trọng số của kết nối giữa các nơ-ron được điều chỉnh để giảm sai số giữa đầu ra dự đoán và đầu ra thực tế thông qua thuật toán lan truyền ngược (backpropagation).

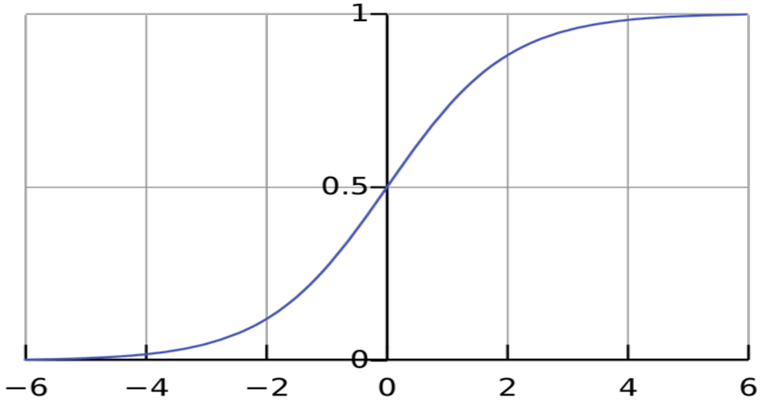
Chương 9. Neural Network

Ví dụ mạng nơ-ron không có lớp ẩn

- Một nơ ron có 2 đầu vào x_1, x_2 và một đầu ra y. Mỗi đầu vào có trọng số w hàm đầu ra f được xác định như sau:
  - $x_1 \rightarrow x_1 * w_1, x_2 \rightarrow x_2 * w_2$
  - $(x_1 * w_1) + (x_2 * w_2) + b$
  - $y = f(x_1 * w_1 + x_2 * w_2 + b)$

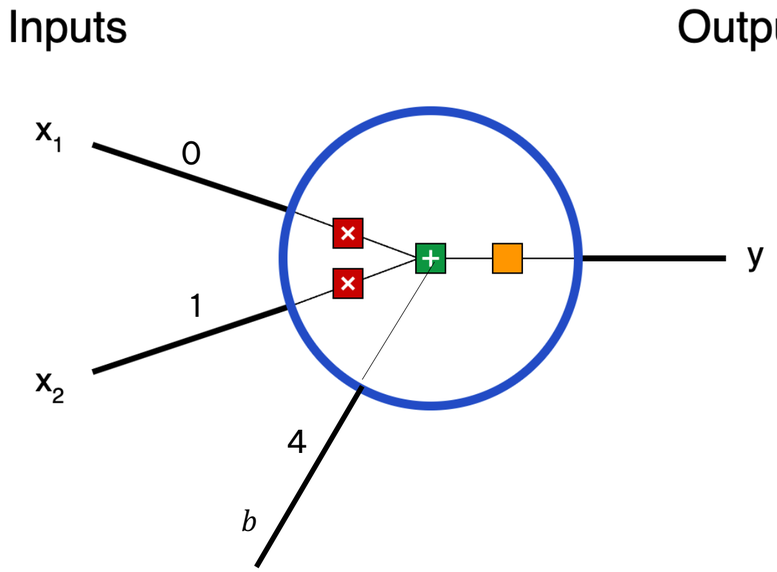


- Hàm kích hoạt được sử dụng để biến đầu vào không giới hạn thành đầu ra có dạng đẹp, có thể dự đoán được, thường được sử dụng là hàm sigmoid.
- *Hàm sigmoid cho ra các số trong phạm vi (0,1). Chúng ta có thể coi nó như là việc nén các số trong phạm vi* $-\infty$ *đến* $+\infty$ *thành 0 đến 1*



Mô hình mạng nơ-ron

- Xem xét ví dụ sau:
- Giả sử chúng ta có một nơron 2 đầu vào sử dụng hàm kích hoạt sigmoid và có các tham số sau:



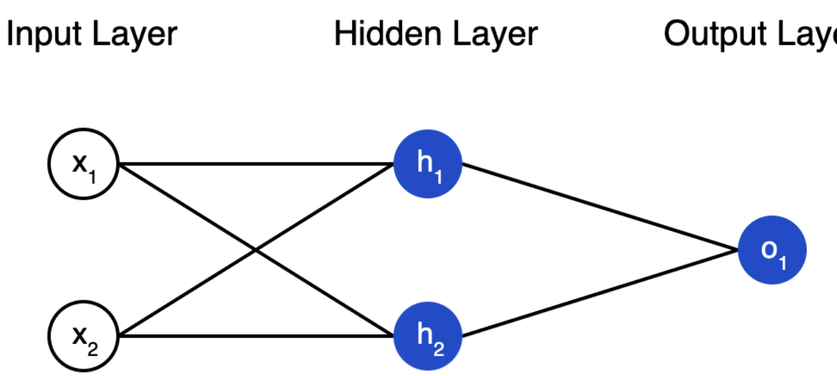
- $\boldsymbol{w} = [\mathbf{0}, \mathbf{1}], b = 4, x = [2,3]$
- $z = (x.w)$

  $= (x_1 * w_1 + x_2 * w_2 + b) = (2 * 0 + 3 * 1 + 4) = 7$

- hàm sigmoid: $f(x) = \frac{1}{1+e^{-x}}$
- $y = f(z) = \frac{1}{1+e^{-7}} \sim 0.99909$, sai số là 0.0091
- Đầu ra của Nơ-ron = 0.999 với giá trị đầu vào[2,3], quá trình này được gọi là quá trình chuyển tiếp (feedforward)

Ví dụ mạng nơ-ron có 1 lớp ẩn

- Mạng nơ ron như hình có 2 đầu vào x_1, x_2, 1 lớp ẩn với 2 nơ ron h_1, h_2, 1 đầu ra o_1, được kết nối với 2 đầu vào h_1, h_2.
- Trọng số của tất cả đầu vào $w = [0,1]$, độ lệch $b = 0$ và có cùng hàm kích hoạt sigmoid.
- Giả sử ta có đầu vào $x = [2,3]$



- $h_1 = h_2 = f(x_1 * w_1 + x_2 * w_2 + b)$

$$= f(2 * 0 + 3 * 1 + 0) = f(3)$$

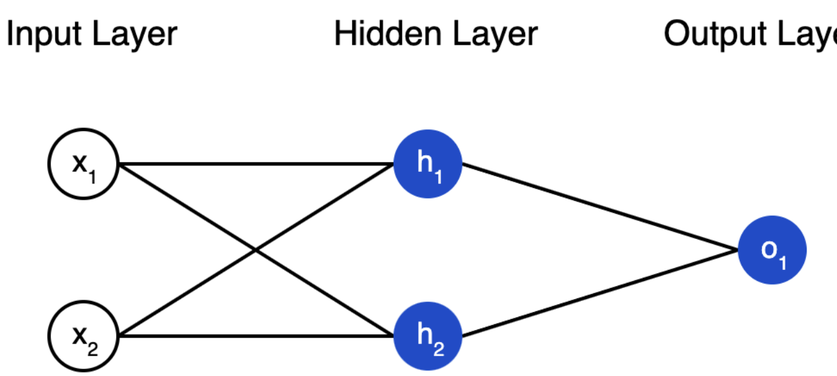
$$= \frac{1}{1+e^{-3}} = 0.9526$$

- $o = f(h_1 * w_1 + h_2 * w_2 + b)$

$$= f(0.925 * (0 + 1) + 0) = f(0.925) = \frac{1}{1+e^{-0.925}} = 0.7216$$

Ví dụ mạng nơ-ron có 1 lớp ẩn

- Mạng nơ ron như hình có 2 đầu vào x_1, x_2, 1 lớp ẩn với 2 nơ ron h_1, h_2, 1 đầu ra o_1, được kết nối với 2 đầu vào h_1, h_2.
- Trọng số của tất cả đầu vào $w = [0,1]$, độ lệch $b = 0$ và có cùng hàm kích hoạt sigmoid.
- Giả sử ta có đầu vào $x = [2,3]$



- $h_1 = h_2 = f(x_1 * w_1 + x_2 * w_2 + b)$

$$= f(2 * 0 + 3 * 1 + 0) = f(3)$$

$$= \frac{1}{1+e^{-3}} = 0.9526$$

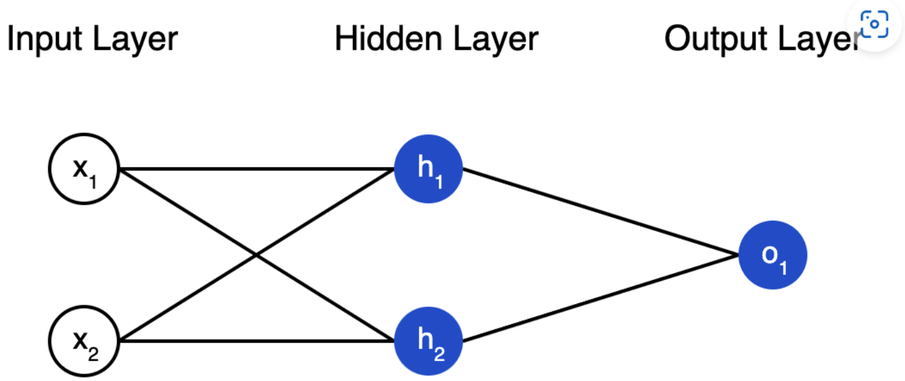
- $o = f(h_1 * w_1 + h_2 * w_2 + b)$

$$= f(0.925 * (0 + 1) + 0) = f(0.925) = \frac{1}{1+e^{-0.925}} = 0.7216$$

Chương 9. Neural Network

Huấn luyện mô hình

Name	Weight (lb)	Height (in)	Gender
Alice	133	65	F
Bob	160	72	M
Charlie	152	70	M
Diana	120	60	F



- Hãy huấn luyện mô hình NN như hình để có thể kết luận về Giới tính (Gender) của ai đó khi biết chiều cao và cân nặng của họ.
- Chúng ta quy định: Nam = 1 và Nữ =0 và hiệu chỉnh lại bảng số liệu như bên:

Name	W=W- 135	H=H - 66	Gender
Alice	-2	-1	1
Bob	25	6	0
Charlie	17	4	0
Diana	-15	-6	1

Thông thường ta biến đổi các giá trị theo giá trị trung bình

Huấn luyện mô hình

- Trước khi huấn luyện mô hình, chúng ta cần xác định tính hiệu quả của mô hình. Quá trình này gọi là Xây dựng hàm mất mát (Loss).
- Hàm mất mát phải phù hợp với bài toán cụ thể mà bạn đang giải quyết. Ví dụ, trong bài toán phân loại nhị phân (binary classification), bạn có thể sử dụng hàm mất mát như Cross-Entropy Loss. Trong bài toán hồi quy (regression), Mean Squared Error (MSE) thường được sử dụng.
- Dựa vào tính chất của dữ liệu và mục tiêu của bài toán, bạn có thể quyết định liệu dữ liệu của bạn thuộc về loại phân loại nhị phân hay hồi quy. Dưới đây là một số điểm quan trọng để xác định loại bài toán:

▪ Huấn luyện mô hình

Phân loại Nhị phân (Binary Classification):

- ❖ Khi mục tiêu của bài toán là phân loại một đối tượng hoặc sự kiện thành 2 lớp (nhóm) khác nhau
- ❖ Kết quả mong muốn là một giá trị rời rạc hoặc nhãn thuộc một trong 2 lớp.

Hồi Quy (Regression):

- ❖ Khi mục tiêu của bài toán là dự đoán hoặc ước tính một giá trị số thực dựa trên dữ liệu đầu vào.
- ❖ Kết quả mong muốn là một giá trị liên tục hoặc số thực, không phải là 1 nhãn.

Chương 9. Neural Network

▪ Huấn luyện mô hình

- ✓ Dự đoán xem một email là "spam" hoặc "không spam" dựa trên nội dung và đặc điểm của email.
- ✓ Dự đoán xem một học sinh có đậu kỳ thi (lớp 1) hay không đậu (lớp 0).

- ○ Dự đoán nhiệt độ dự kiến trong ngày tiếp theo dựa trên dữ liệu thời tiết hiện tại.
- ○ Dự đoán doanh số bán hàng dự kiến của một sản phẩm trong tương lai dựa trên các yếu tố tiếp thị.

▪ Dựa vào tính chất của dữ liệu đầu vào, ta sử dụng sai số là hàm sai số bình phương trung bình (mean squared error -MSE) cho hàm mất mát.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_{true} - y_{pred})^2$$

- Giá trị MSE càng nhỏ càng tốt, vì nó đo lường mức độ sai lệch giữa dự đoán và giá trị thực tế. Mô hình tối ưu là mô hình có MSE thấp nhất.
- MSE tính toán sai số bằng cách bình phương sai số, điều này có thể tạo ra giá trị lớn khi có sai số lớn. Điều này có thể làm cho mô hình nhạy cảm với các các giá trị ngoại lai (outliers).
- MSE không cân nhắc thứ tự sai số, nghĩa là nó không quan tâm đến xem sai

Chương 9. Neural Network

▪ Huấn luyện mô hình

- Dựa vào tính chất của dữ liệu đầu vào, ta sử dụng sai số là hàm sai số bình phương trung bình (mean squared error -MSE) cho hàm mất mát.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_{true} - y_{pred})^2$$

- Giá trị MSE càng nhỏ càng tốt, vì nó đo lường mức độ sai lệch giữa dự đoán và giá trị thực tế. Mô hình tối ưu là mô hình có MSE thấp nhất.
- MSE tính toán sai số bằng cách bình phương sai số, điều này có thể tạo ra giá trị lớn khi có sai số lớn. Điều này có thể làm cho mô hình nhạy cảm với các các giá trị ngoại lai (outliers).
- MSE không cân nhắc thứ tự sai số, nghĩa là nó không quan tâm đến xem sai số là dương hay âm.

Chương 9. Neural Network

▪ Huấn luyện mô hình

▪ Tránh nhầm lẫn 2 khái niệm sai số bình phương (MSE) với Phương sai (Variance)

✓Sai số bình phương trung bình, là một hàm mất mát thường được sử dụng trong các bài toán hồi quy để đo lường mức độ sai lệch giữa các giá trị dự đoán của mô hình và giá trị thực tế của dữ liệu huấn luyện.

✓MSE đo lường sự chênh lệch bình phương trung bình giữa giá trị dự đoán và giá trị thực tế và được tính bằng cách lấy trung bình của bình phương của các sai số.

✓ Phương sai là một khái niệm thống kê đo lường sự biến đổi của dữ liệu. Nó cho biết độ lệch của dữ liệu so với giá trị trung bình. Nếu phương sai lớn, dữ liệu có xu hướng phân tán rộng hơn xung quanh giá trị trung bình.

✓ Công thức: $\delta^2 = \sum(x_i - \mu)^2$

✓ xi là giá trị của dữ liệu thứ i

✓ μ là giá trị trung bình của tập dữ liệu: $\Sigma x_i/N$

✓ N là số lượng giá trị trong tập dữ liệu

Huấn luyện mô hình

- Từ biểu thức: $MSE = \frac{1}{n}\sum_{i=1}^{n}(y_{true} - y_{pred})^2$
- n=4 (Alice, Bob, Charlie, Diana)
- y: đại diện cho biến dự đoán
- y_{true}: giá trị thực của biến gender
  (Alice =1, Bob=0)
- y_{pred}: giá trị dự đoán của biến
- $(y_{true} - y_{pred})^2$: sai số bình phương.
- Hàm này tính giá trị trung bình bình phương của các sai số. Sai số càng nhỏ chứng tỏ mô hình càng hiệu quả.

Name	W=W- 135	H=H - 66	Gender
Alice	-2	-1	1
Bob	25	6	0
Charlie	17	4	0
Diana	-15	-6	1

Training a network = trying to minimize its loss.

Huấn luyện mô hình

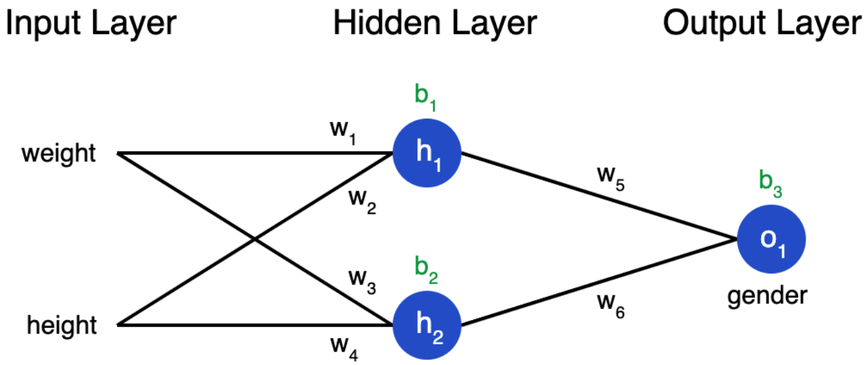
- Hãy tính độ mất mát nếu kết quả dự đoán cho bởi bảng sau:
- $MSE = \frac{1}{4}(1 + 0 + 0 + 1) = 0.5$
- Để giảm thiểu giá trị của hàm mất mát, chúng ta có thể thay đổi giá trị của trọng số (w) và hệ số (b)

Name	y_{true}	y_{pred}	$(y_{true} - y_{pred})^2$
Alice	1	0	1
Bob	0	0	0
Charlie	0	0	0
Diana	1	0	1

Huấn luyện mô hình

- Xem xét sai số bình phương trung bình của Alice:
- $MSE = \frac{1}{1}\sum_{i=1}^{1}(y_{true} - y_{pred})^2 = (1 - y_{pred})^2$
- Hàm mất mát cũng có thể được tính toán thông qua: $\mathcal{L} = (w_1, w_2, w_3, w_4, w_5, w_6, b_1, b_2, b_3)$
- **Giả sử ta cần thay đổi** giá trị w_1, hàm mất mát $\mathcal{L}$ sẽ thay đổi theo công thức đạo hàm riêng $\frac{\partial \mathcal{L}}{\partial w_1}$
- $\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial \mathcal{L}}{\partial y_{pred}} \cdot \frac{\partial y_{pred}}{\partial w_1}$ (công thức đạo hàm hàm hợp)

Name	W - 135	H - 66	Gender
Alice	-2	-1	1



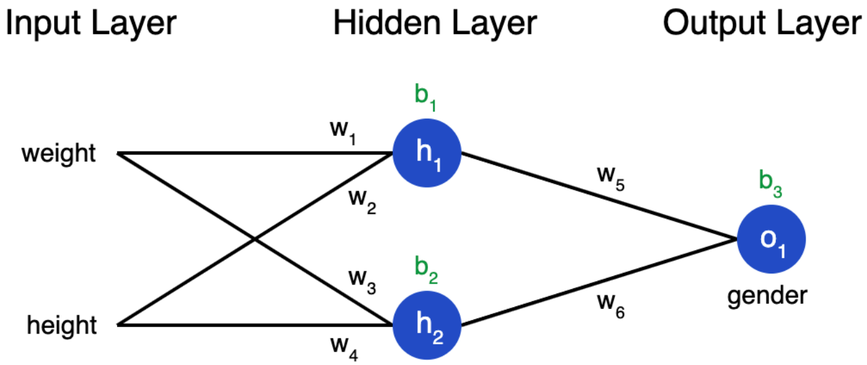
Chương 9. Neural Network

▪ Huấn luyện mô hình

- $Ta\ có\ L = (1 - y_{pred})^2$
- $\frac{\partial \mathcal{L}}{\partial y_{pred}} = \frac{\partial (1-y_{pred})^2}{\partial y_{pred}}$
- $\frac{\partial \mathcal{L}}{\partial y_{pred}} = \frac{2(1-y_{pred})(1-y_{pred})'}{\partial y_{pred}} = -2(1 - y_{pred})$
- $y_{pred} = o_1 = f(w_5 h_1 + w_6 h_2 + b_3)$: hàm kích hoạt
- Mà h_1 phụ thuộc w_1 còn h_2 thì không
- $\frac{\partial y_{pred}}{\partial w_1} = \frac{\partial y_{pred}}{\partial h_1} * \frac{\partial h_1}{\partial w_1}$
- $\frac{\partial y_{pred}}{\partial h_1} = w_5 * f'(w_5 h_1 + w_6 h_2 + b_3)$

(công thức đạo hàm hàm hợp)

Name	W - 135	H - 66	Gender
Alice	-2	-1	1



$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial \mathcal{L}}{\partial y_{pred}} * \frac{\partial y_{pred}}{\partial w_1}$$

Chương 9. Neural Network

Huấn luyện mô hình

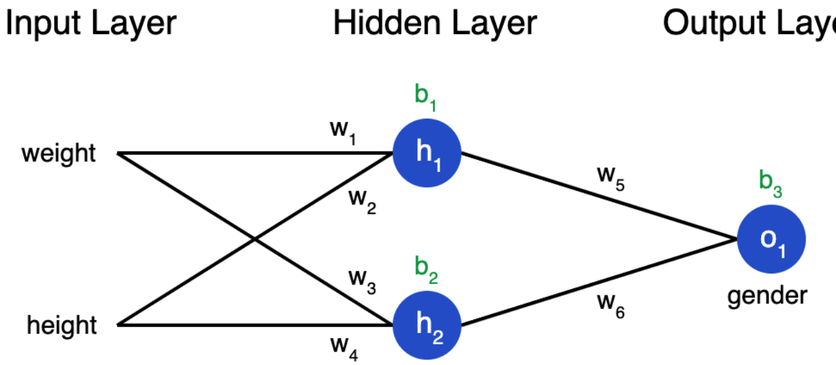
- $h_1 = f(w_1 x_1 + w_2 h_2 + b)$
- $\frac{\partial h_1}{\partial w_1} = x_1 * f'(w_1 x_1 + w_2 h_2 + b)$

(công thức đạo hàm hàm hợp)

Trong đó:

- x_1: $weight$
- x_2: $height$
- Công thức hàm sigmoid: $f(x) = \frac{1}{1+e^{-x}}$
- $f'(x)$: hàm kích hoạt của sigmoid
- $(\frac{f(x)}{g(x)})' = \frac{f'(x)g(x) - f(x)g'(x)}{g(x)^2}$

Name	W - 135	H - 66	Gender
Alice	-2	-1	1



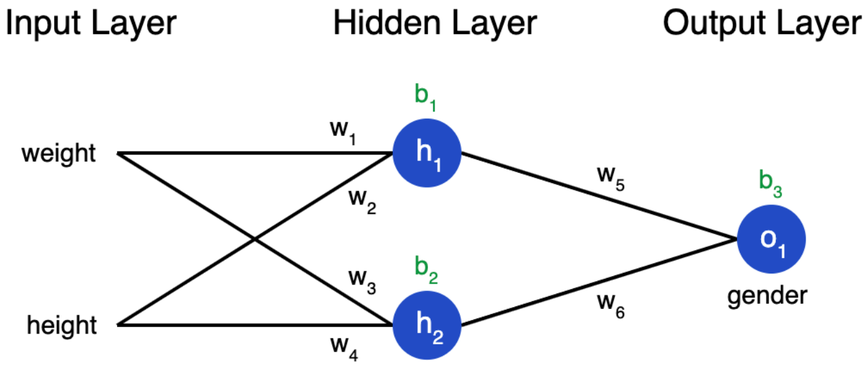
$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial \mathcal{L}}{\partial y_{pred}} * \frac{\partial y_{pred}}{\partial w_1}$$

Chương 9. Neural Network

Huấn luyện mô hình

- Công thức hàm sigmoid: $f(x) = \frac{1}{1+e^{-x}}$
- $f'(x) = (\frac{1}{1+e^{-x}})' = \frac{0*(1+e^{-x})-(1+e^{-x})'*1}{(1+e^{-x})^2}$
- $f'(x) = \frac{o-(-e^{-x}))}{(1+e^{-x})^2} = \frac{e^{-x}}{(1+e^{-x})^2}$
- $f'(x) = \frac{1}{1+e^{-x}} * \frac{e^{-x}}{1+e^{-x}} = \frac{1}{1+e^{-x}} * (1 - \frac{1}{1+e^{-x}})$
- $f'(x) = f(x) * (1 - f(x))$

Name	W - 135	H - 66	Gender
Alice	-2	-1	1



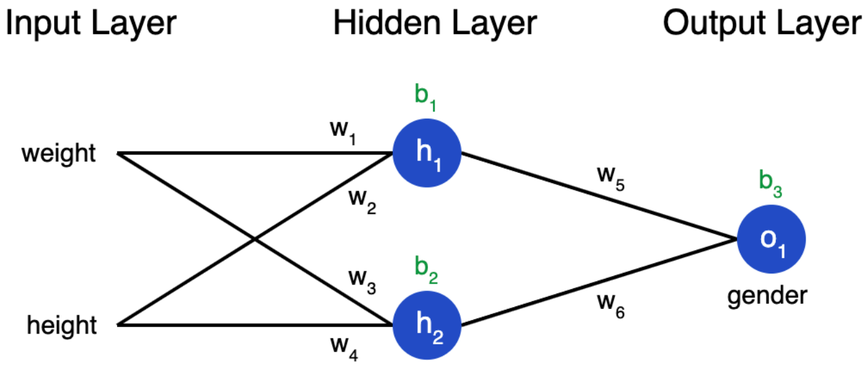
$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial \mathcal{L}}{\partial y_{pred}} * \frac{\partial y_{pred}}{\partial w_1}$$

$$(e^u)' = u' * e^u$$

Huấn luyện mô hình

- Sau khi phân rã $\frac{\partial \mathcal{L}}{\partial w_1}$ thành các thành phần riêng lẻ, chúng ta có thể tính toán
- $\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial \mathcal{L}}{\partial y_{pred}} * \frac{\partial y_{pred}}{\partial h_1} * \frac{\partial h_1}{\partial w_1}$
- Hệ thống tính đạo hàm riêng bằng cách tính ngược này được gọi là lan truyền ngược hoặc "backprop".

Name	W - 135	H - 66	Gender
Alice	-2	-1	1

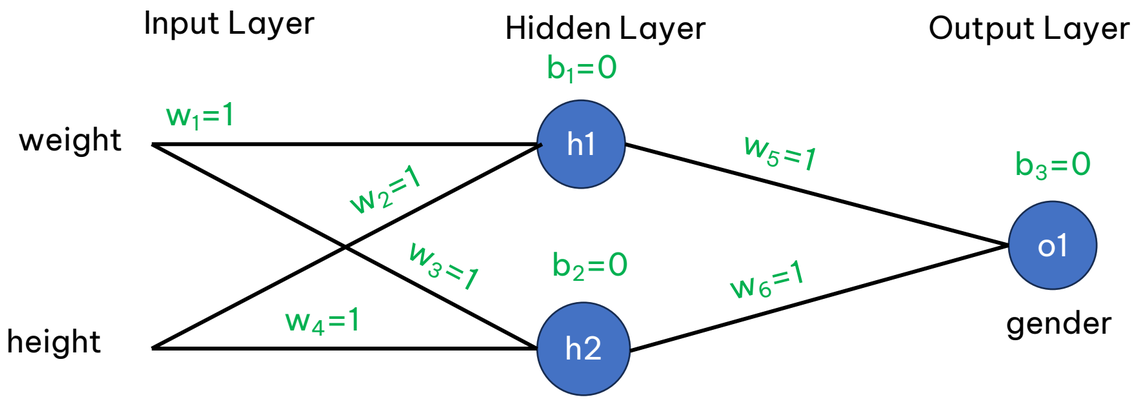


$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial \mathcal{L}}{\partial y_{pred}} * \frac{\partial y_{pred}}{\partial w_1}$$

Ví dụ về tính đạo hàm riêng cho mô hình

- Xem xét dữ liệu về Alice như sau:

Name	W - 135	H - 66	Gender
Alice	-2	-1	1



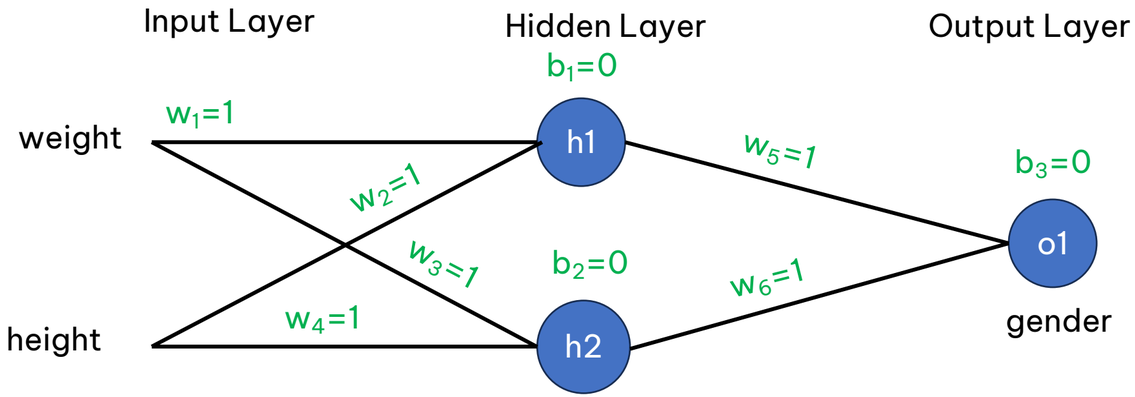
- $h_1 = (x_1 w_1 + x_2 w_2 + b_1) = f(-2 * 1 - 1 * 1 + 0)$
- $h_1 = f(-3) = \frac{1}{1+e^{-(-3)}} = 0.0474$
- $h_2 = h_1 = 0.0474$
- $o_1 = (h_1 w_5 + h_2 w_6 + b_3)$
- $o_1 = f(0.0474 * 1 + 0.0474 * 1 + 0)$
- $o_1 = f(0{,}0948) = \frac{1}{1+e^{(-0.0948)}} = 0.5238$
- Kết quả dự đoán của mô hình $y_{pred} = 0.524$ chưa đủ chắc chắn để có thể kết luận giới tính của ai đó nếu cho trước chiều cao và cân nặng.

$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial \mathcal{L}}{\partial y_{pred}} * \frac{\partial y_{pred}}{\partial h_1} * \frac{\partial h_1}{\partial w_1}$$

Ví dụ về tính đạo hàm riêng cho mô hình

- Tính toán giá trị cho hàm mất mát:

Name	W - 135	H - 66	Gender
Alice	-2	-1	1



$$\frac{\partial \mathcal{L}}{\partial y_{pred}} = -2(1 - y_{pred}) = -2(1 - 0.524) = -0.952$$

$$\frac{\partial y_{pred}}{\partial h_1} = w_5 * f'(w_5 h_1 + w_6 h_2 + b_3)$$

$$\frac{\partial y_{pred}}{\partial h_1} = 1 * f'(1 * 0.0474 + 1 * 0.0474 + 0)$$

$$\frac{\partial y_{pred}}{\partial h_1} = f'(0.0948)$$

- **Ta có:** $\boldsymbol{f'(x) = \frac{e^{-x}}{(1+e^{-x})^2} = f(x) * (1 - f(x))}$

$$\frac{\partial y_{pred}}{\partial h_1} = f(0.0948) * (1 - f(0.0948))$$

$$\frac{\partial y_{pred}}{\partial h_1} = 0.524 * (1 - 0.524) = 0.249$$

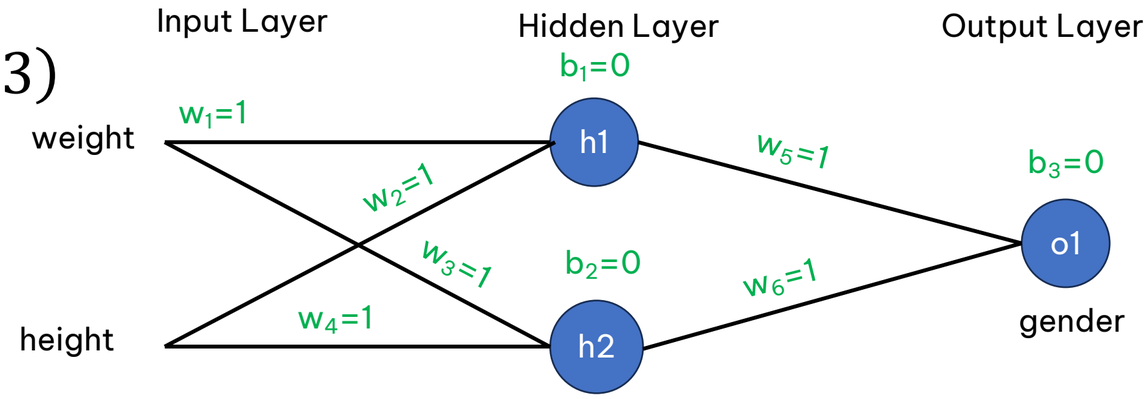
$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial \mathcal{L}}{\partial y_{pred}} * \frac{\partial y_{pred}}{\partial h_1} * \frac{\partial h_1}{\partial w_1}$$

$$f(0.0948) = \frac{1}{1 + e^{-(0.0948)}}$$

Ví dụ về tính đạo hàm riêng cho mô hình

- Tính toán giá trị cho hàm mất mát:

Name	W - 135	H - 66	Gender
Alice	-2	-1	1



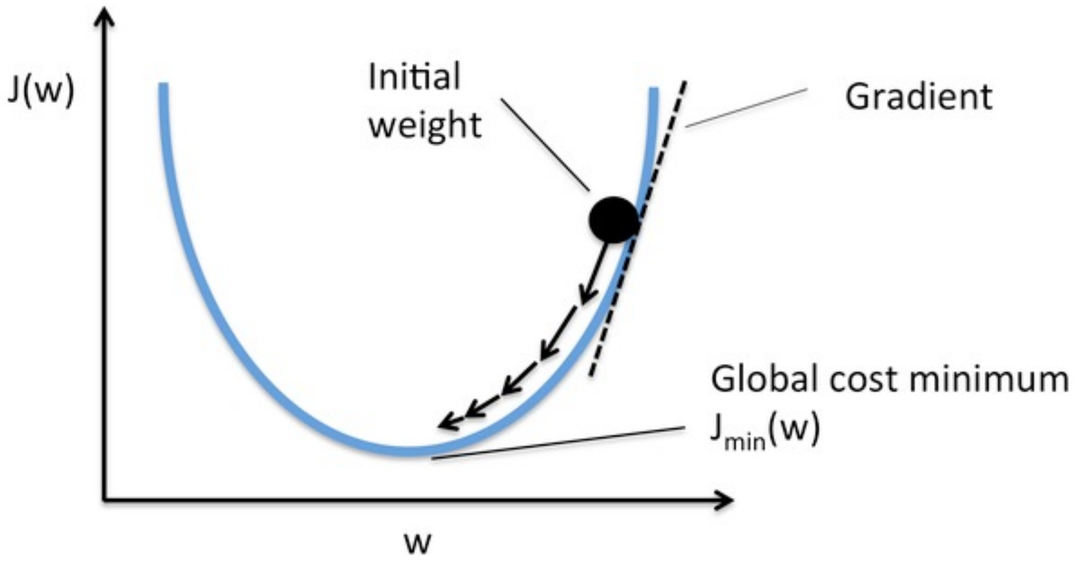
- $\frac{\partial h_1}{\partial w_1} = x_1 * f'(w_1 x_1 + w_2 x_2 + b_1)$
- $\frac{\partial h_1}{\partial w_1} = -2 * f'(1 * -2 + 1 * -1 + 0) = -2 * f'(3)$
- **Ta có:** $\boldsymbol{f'(x) = \frac{e^{-x}}{(1+e^{-x})^2} = f(x) * (1 - f(x))}$
- $\frac{\partial h_1}{\partial w_1} = -2 * f(3) * \big(1 - f(3)\big)$
- $\frac{\partial h_1}{\partial w_1} = -2 * 0.9526 * 0.0474 = -0.0904$
- $\frac{\partial \mathcal{L}}{\partial w_1} = -0.952 * 0.249 * -0.0904 = 0.0214$
- Khi ta tăng giá trị w_1, giá trị $\mathcal{L}$ cũng tăng nhẹ

$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial \mathcal{L}}{\partial y_{pred}} * \frac{\partial y_{pred}}{\partial h_1} * \frac{\partial h_1}{\partial w_1}$$

$$f(3) = \frac{1}{1+e^{-(3)}} = 0.9526$$

Giảm dần độ dốc ngẫu nhiên (Stochastic Gradient Descent - SGD)

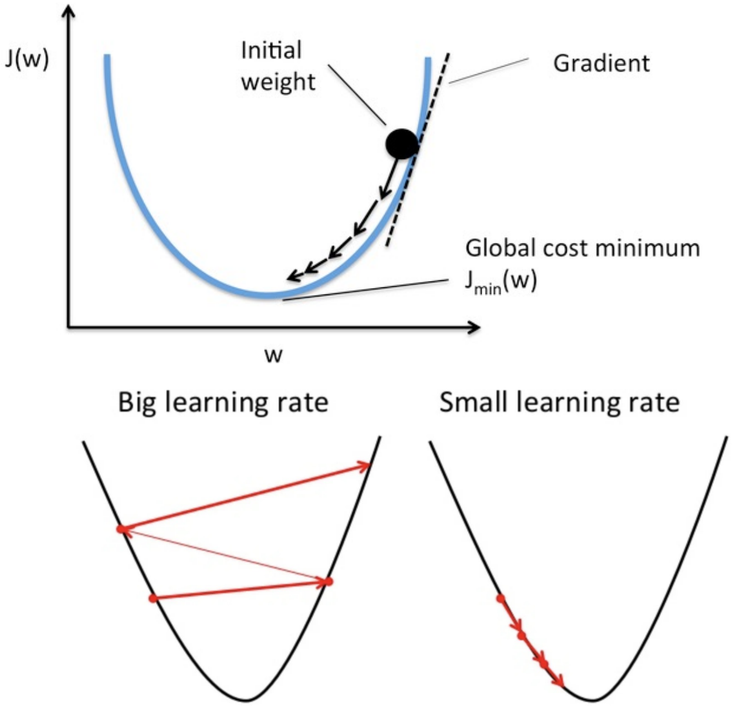
- SGD là một thuật toán tối ưu hóa được sử dụng rộng rãi trong học máy và học sâu để tối ưu hóa các mô hình máy học, đặc biệt là trong việc huấn luyện các mạng nơ-ron nhân tạo (neural networks). SGD là một biến thể của thuật toán Gradient Descent (GD).
- Trong hình, đường cong là hàm chi phí và mục đích là đạt đến giá trị sai số tối thiểu J_{min}



Thuật toán Gradient Descent

Giảm dần độ dốc ngẫu nhiên Stochastic Gradient Descent

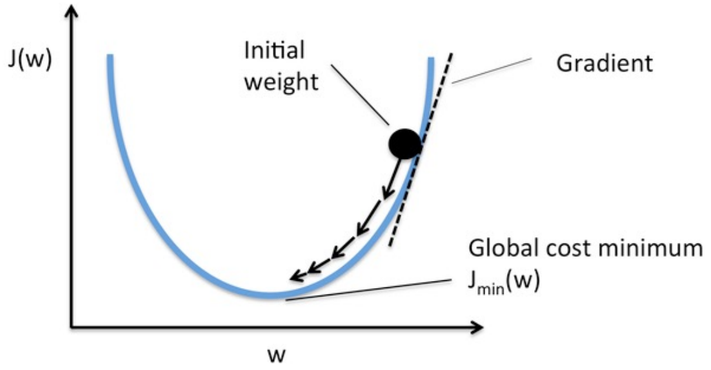
- Tốc độ học (a) trong phương trình cập nhật trọng số không nên quá nhỏ hoặc quá lớn đến mức không đạt được giá trị cực tiểu.

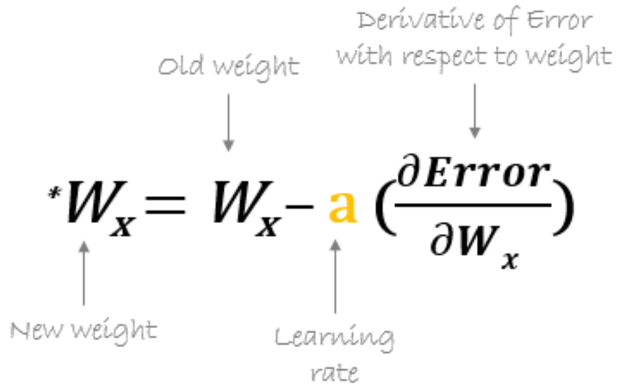


Chương 9. Neural Network

Giảm dần độc dốc ngẫu nhiên Stochastic Gradient Descent

- Các bước xác định cực tiểu hoá
- Bước 1, khởi tạo giá trị cho trọng số w và hệ số b
- Bước 2, tính toán độ dốc dựa vào đạo hàm riêng của hàm lỗi (error)
- Bước 3, sau đó, các trọng số mới (w_{new}) được tính bằng công thức bên dưới, trong đó *a* là tốc độ học
- Bước 4, quá trình tính toán được lặp lại từ các trọng số mới này cho đến khi chúng ta đạt đến mức tối thiểu toàn cục và giá trị error được giảm thiểu.





Giảm dần độc dốc ngẫu nhiên Stochastic Gradient Descent

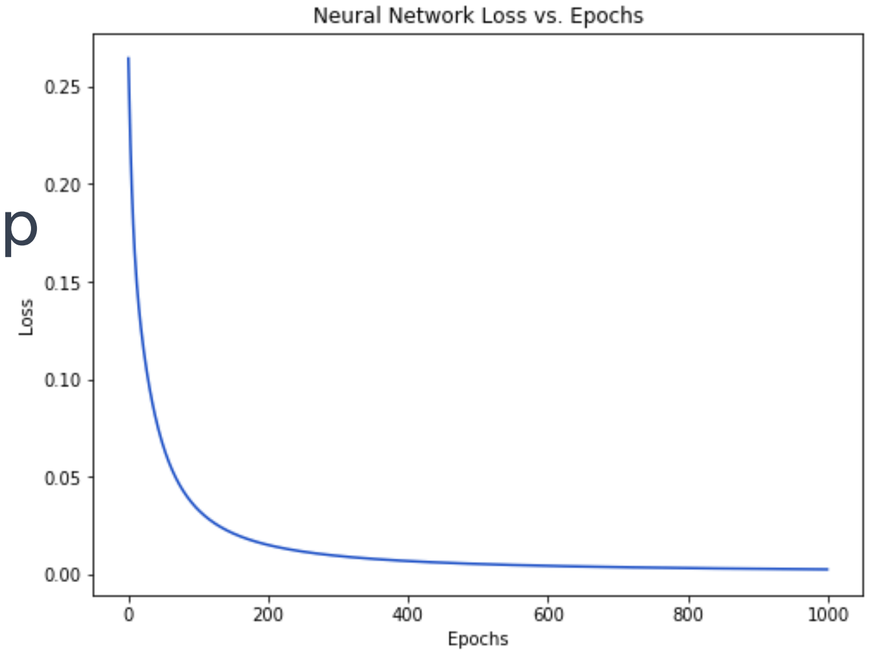
- Thuật toán giảm độ dốc ngẫu nhiên (SGD) giups chúng ta biết cách làm thế nào để thay đổi trọng số (w) và độ lệch (b) để giảm thiểu tối đa giá trị tổn thất (mất mát) thông qua phương trình sau:
- $w_1 \leftarrow w_1 - \eta \frac{\partial \mathcal{L}}{\partial w_1}$
- η: hằng số (learning rate) kiểm soát tốc độ huấn luyện dữ liệu.
- Nếu $\frac{\partial \mathcal{L}}{\partial w_1}$ >0, w_1 sẽ tăng, $\mathcal{L}$ sẽ giảm
- Nếu $\frac{\partial \mathcal{L}}{\partial w_1}$ <0, w_1 sẽ giảm, $\mathcal{L}$ sẽ giảm
- Nếu chúng ta thực hiện với mọi w_i và b_i, giá trị hàm mất mát sẽ giảm và mô hình được cải thiện.

Chương 9. Neural Network

Tóm lược quy trình huấn luyện dữ liệu

1. Chọn một mẫu ngẫu nhiên từ tập dữ liệu
2. Tính tất cả đạo hàm riêng của hàm mất mát theo trọng số (w) và hệ số (b)
3. Sử dung phương trình cập nhật để cập nhật lại giá trị cho w và b
4. Lặp lại bước 1

- Giá trị của hàm mất mát giảm dần theo số lần lặp



HẾT CHƯƠNG 9