TOÁN CHO KHOA HỌC MÁY TÍNH

HỒI QUY SOFTMAX

TS. Lương Ngọc Hoàng

Nội dung

1. Giới thiệu hồi quy softmax
2. Hàm mất mát & Tính toán gradient
3. Cài đặt & Phân tích hồi quy softmax

HỒI QUY SOFTMAX

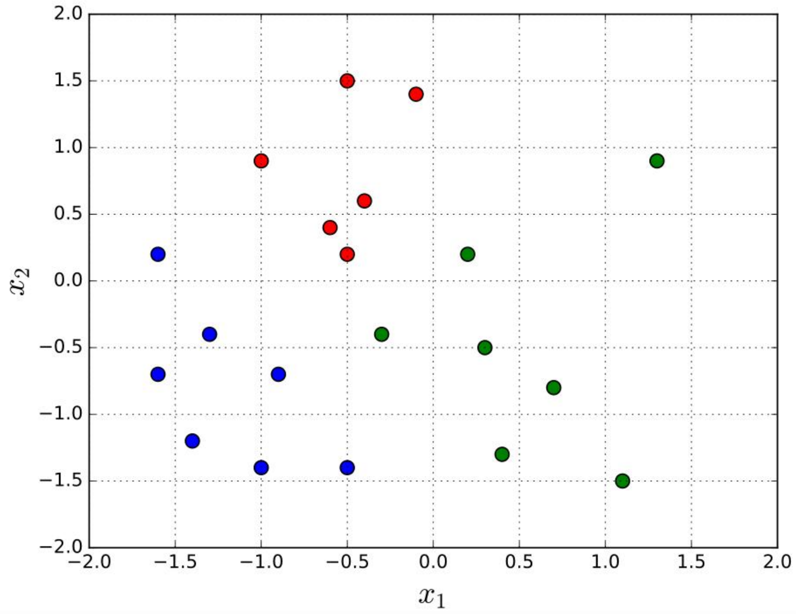
SOFTMAX REGRESSION

Bài toán phân lớp

Bài toán phân lớp (classification):

Với mỗi điểm dữ liệu đầu vào $\boldsymbol{x} = (x_1, x_2)$, ta muốn đầu ra của mô hình dự đoán nhãn (label) của $\boldsymbol{x}$ là nhãn 0 (lớp đỏ), nhãn 1 (lớp xanh lá), hay nhãn 2 (lớp xanh dương).

Ta muốn đầu ra của mô hình dự đoán xác suất điểm dữ liệu $\boldsymbol{x} = (x_1, x_2)$ thuộc về lớp đỏ (nhãn 0), lớp xanh lá (nhãn 1), và lớp xanh dương (nhãn 2) là bao nhiêu.

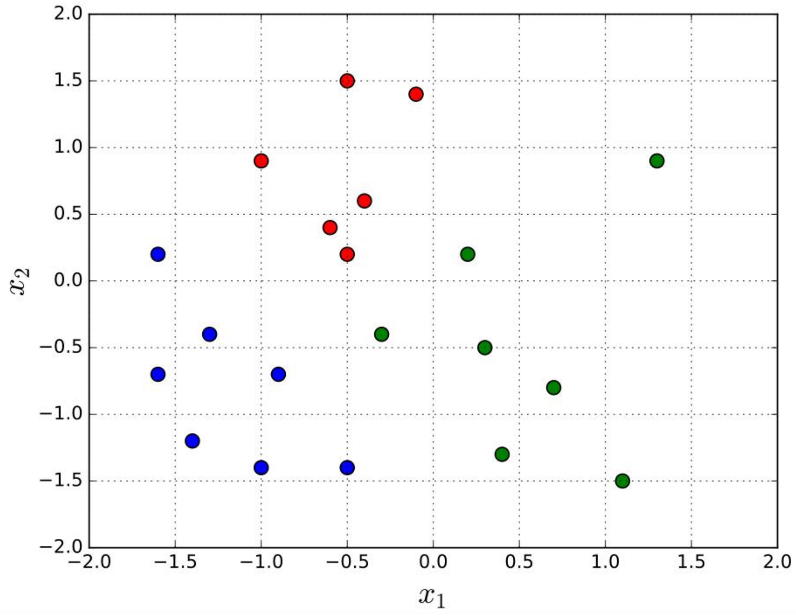


Có thể sử dụng Hồi quy logistic không?

- Đầu ra của mô hình hồi quy logistic là một giá trị xác suất $a \in (0,1)$ nên phù hợp với bài toán phân lớp nhị phân có hai nhãn $y \in \{0,1\}$ nhưng bài toán trên có 3 nhãn $y \in \{0,1,2\}$.

Bài toán phân lớp

Ta muốn đầu ra của mô hình dự đoán xác suất điểm dữ liệu $\boldsymbol{x} = (x_1, x_2)$ thuộc về lớp đỏ (nhãn 0), lớp xanh lá (nhãn 1), và lớp xanh dương (nhãn 2) là bao nhiêu.



Đầu ra của hồi quy logistic là $a \in (0,1)$ không phù hợp với trường hợp có nhiều hơn 2 nhãn, ví dụ $y \in \{0,1,2\}$.

Giải pháp:

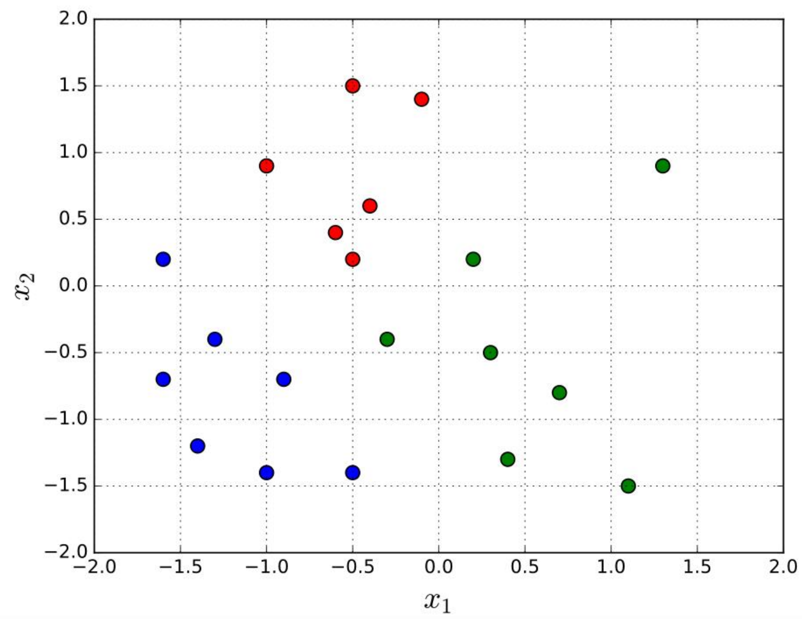
Biểu diễn nhãn mỗi điểm dữ liệu thành one-hot vector:

$$\text{Đỏ } 0 \sim \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \text{ Xanh lá } 1 \sim \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \text{ Xanh dương } 2 \sim \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Câu hỏi: Có thể sử dụng 3 mô hình hồi quy logistic độc lập, dự đoán xác suất mỗi lớp $a_1, a_2, a_3 \in (0,1)$?

Bài toán phân lớp

Ta muốn đầu ra của mô hình dự đoán xác suất điểm dữ liệu $\boldsymbol{x} = (x_1, x_2)$ thuộc về lớp đỏ (nhãn 0), lớp xanh lá (nhãn 1), và lớp xanh dương (nhãn 2) là bao nhiêu.



Sử dụng 3 mô hình hồi quy logistic độc lập. Mỗi mô hình hồi quy logistic có đầu ra là:

$$a_1 = \sigma(z_1) = \frac{1}{1+e^{-z_1}} \text{ với } z_1 = w_{1,1}x_1 + w_{1,2}x_2 + b_1$$

$$a_2 = \sigma(z_2) = \frac{1}{1+e^{-z_2}} \text{ với } z_2 = w_{2,1}x_1 + w_{2,2}x_2 + b_2$$

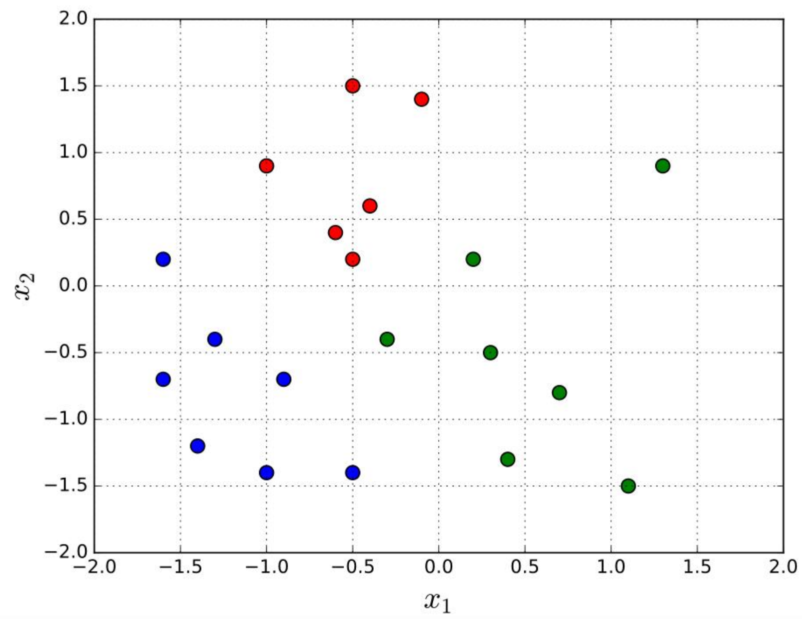
$$a_3 = \sigma(z_3) = \frac{1}{1+e^{-z_3}} \text{ với } z_3 = w_{3,1}x_1 + w_{3,2}x_2 + b_3$$

Vấn đề của phương pháp trên?

Mỗi mô hình hồi quy logistic dự đoán xác suất mỗi lớp $a_1, a_2, a_3 \in (0,1)$ một cách độc lập → Ta không thể ràng buộc $a_1 + a_2 + a_3 = 1$.

Bài toán phân lớp

Ta muốn đầu ra của mô hình dự đoán xác suất điểm dữ liệu $\boldsymbol{x} = (x_1, x_2)$ thuộc về lớp đỏ (nhãn 0), lớp xanh lá (nhãn 1), và lớp xanh dương (nhãn 2) là bao nhiêu.



Không thể ràng buộc $a_1 + a_2 + a_3 = 1$ với:

$a_1 = \sigma(z_1) = \frac{1}{1+e^{-z_1}}$ với $z_1 = w_{1,1}x_1 + w_{1,2}x_2 + b_1$

$a_2 = \sigma(z_2) = \frac{1}{1+e^{-z_2}}$ với $z_2 = w_{2,1}x_1 + w_{2,2}x_2 + b_2$

$a_3 = \sigma(z_3) = \frac{1}{1+e^{-z_3}}$ với $z_3 = w_{3,1}x_1 + w_{3,2}x_2 + b_3$

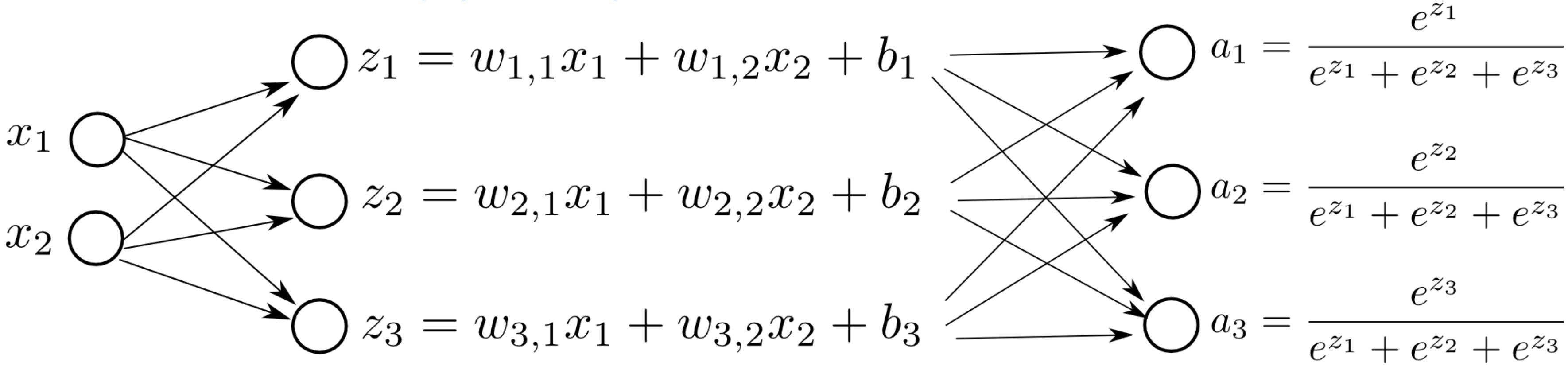
Giải pháp: Thay vì sử dụng hàm logistic sigmoid trên từng z_i, ta sử dụng hàm softmax trên đồng thời tất cả các giá trị z_i với $(a_1, a_2, a_3) = \mathrm{softmax}(z_1, z_2, z_3)$.

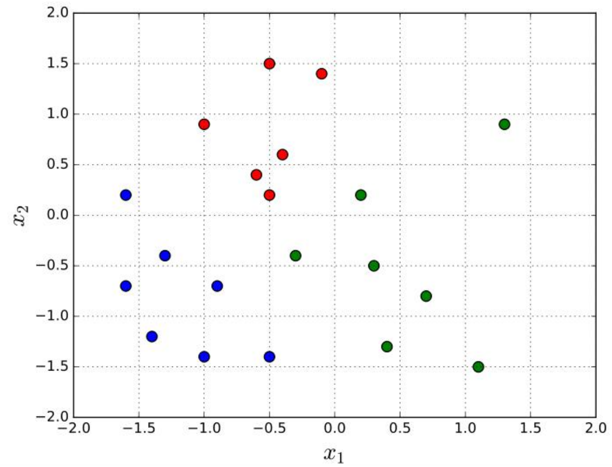
$$a_1 = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3}}, \quad a_2 = \frac{e^{z_2}}{e^{z_1} + e^{z_2} + e^{z_3}}, \quad a_3 = \frac{e^{z_3}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

Ta có thể ràng buộc $a_1 + a_2 + a_3 = 1$.

Hồi quy softmax (softmax regression)

Ta muốn đầu ra của mô hình dự đoán xác suất điểm dữ liệu $\boldsymbol{x} = (x_1, x_2)$ thuộc về lớp đỏ (nhãn 0), lớp xanh lá (nhãn 1), và lớp xanh dương (nhãn 2) là bao nhiêu.





Mô hình hồi quy softmax có các tham số là:

$$W = \begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,1} \\ w_{3,1} & w_{3,2} \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

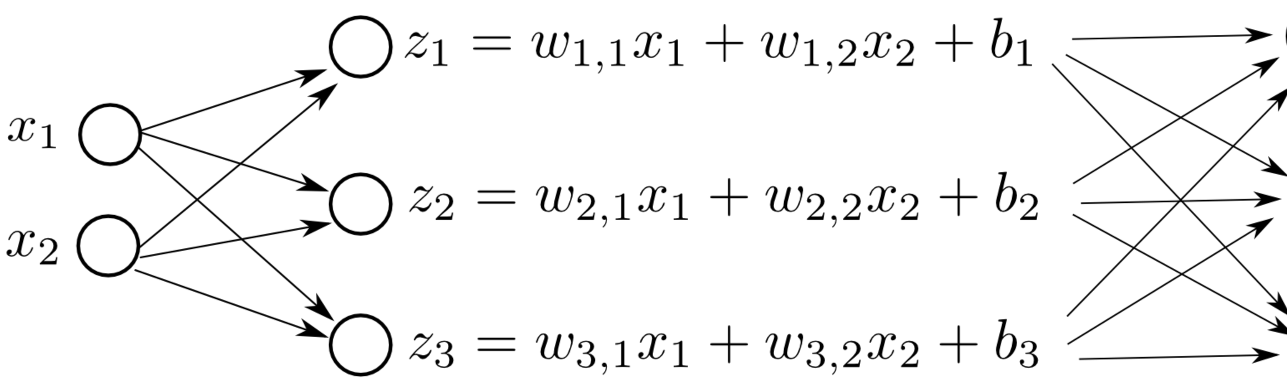
Với mỗi điểm dữ liệu $\boldsymbol{x} = (x_1, x_2)$, ta thực hiện dự đoán:

$$\boldsymbol{z} = W\boldsymbol{x} + \boldsymbol{b}$$

$$\boldsymbol{a} = \text{softmax}(\boldsymbol{z})$$

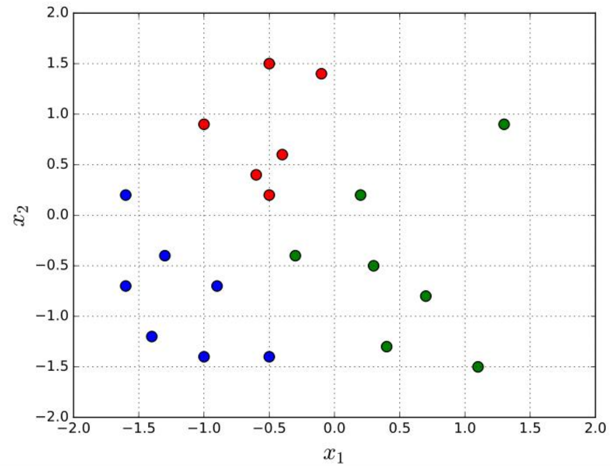
Hồi quy softmax (softmax regression)

Ta muốn đầu ra của mô hình dự đoán xác suất điểm dữ liệu $\boldsymbol{x} = (x_1, x_2)$ thuộc về lớp đỏ (nhãn 0), lớp xanh lá (nhãn 1), và lớp xanh dương (nhãn 2) là bao nhiêu.



$$z_1 = w_{1,1}x_1 + w_{1,2}x_2 + b_1$$
$$z_2 = w_{2,1}x_1 + w_{2,2}x_2 + b_2$$
$$z_3 = w_{3,1}x_1 + w_{3,2}x_2 + b_3$$

$$a_1 = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3}}$$
$$a_2 = \frac{e^{z_2}}{e^{z_1} + e^{z_2} + e^{z_3}}$$
$$a_3 = \frac{e^{z_3}}{e^{z_1} + e^{z_2} + e^{z_3}}$$




```
W = np.array([[ 0.31, 3.95],
              [ 7.07, -0.23],
              [-6.27, -2.35]])

b = np.array([[ 1.2  ],
              [ 2.93 ],
              [-4.14 ]])
```

```
def forward(W, b, x):
    z = np.matmul(W, x) + b
    a = softmax(z)

    return z, a
```

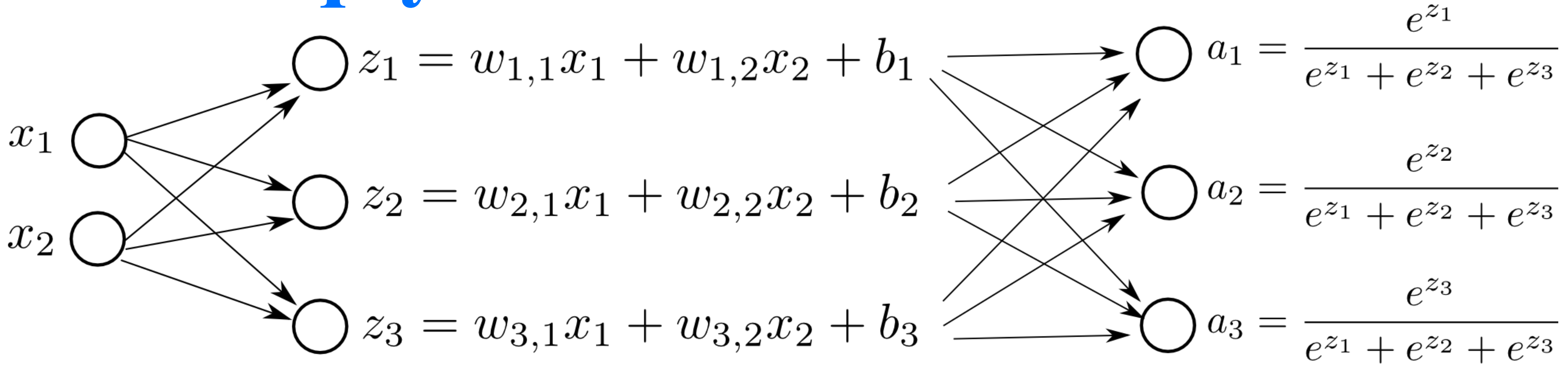
```
# X[0,:] has the shape (1,2)
# reshape from (1,2) --> (2,1)
x = X[0,:].reshape(2,1)
z, a = forward(W, b, x)

print(x.squeeze())
print(z.squeeze())
print(a.squeeze())
print(y[0])
```

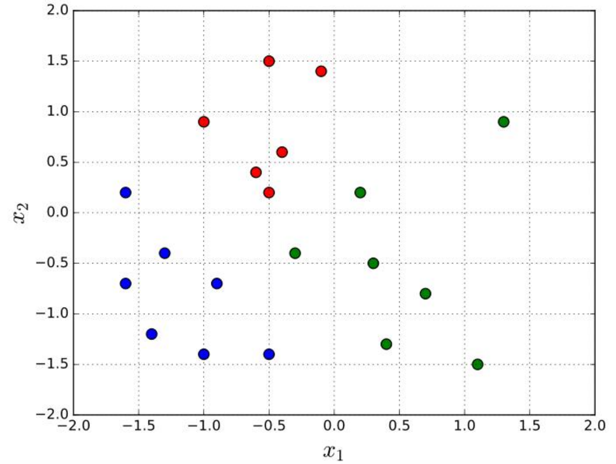
```
[-0.1 1.4]
[ 6.699 1.901 -6.803]
[0.992 0.008 0. ]
0
```

HÀM MẤT MÁT & TÍNH TOÁN GRADIENT

LOSS FUNCTION & GRADIENT COMPUTATION

Hồi quy softmax – Hàm mất mát





Ta cần học giá trị ma trận W và vector $\boldsymbol{b}$ để phân biệt tốt các điểm dữ liệu đỏ , xanh lá, xanh dương trong tập dữ liệu huấn luyện.

Giả sử ta có 3 điểm dữ liệu như sau:

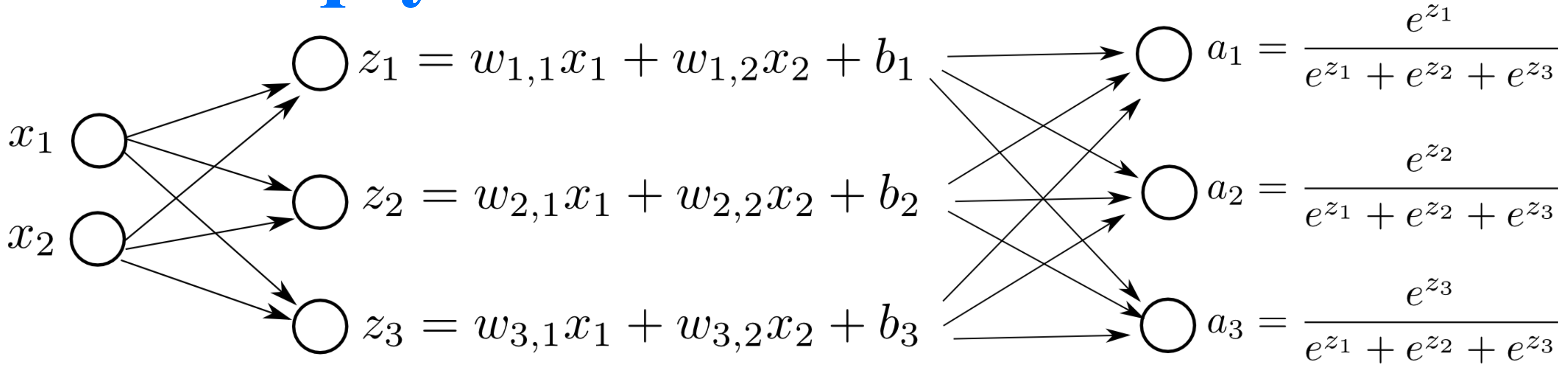
$$\boldsymbol{x}^{(1)} = \begin{bmatrix} -0.1 \\ 1.4 \end{bmatrix},\ y^{(1)} = 0,\ \boldsymbol{x}^{(2)} = \begin{bmatrix} 1.3 \\ 0.9 \end{bmatrix},\ y^{(2)} = 1,\ \boldsymbol{x}^{(3)} = \begin{bmatrix} -1.4 \\ -1.1 \end{bmatrix},\ y^{(3)} = 2$$

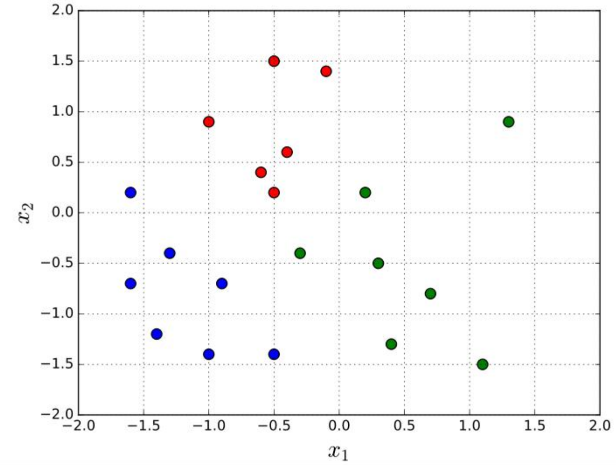
Giả sử ta có một mô hình (một bộ phân lớp – classifier) cho ra dự đoán:

$$\boldsymbol{y}^{(1)} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix},\ \boldsymbol{a}^{(1)} = \begin{bmatrix} 0.9 \\ 0.1 \\ 0 \end{bmatrix},\ \boldsymbol{y}^{(2)} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix},\ \boldsymbol{a}^{(2)} = \begin{bmatrix} 0.1 \\ 0.8 \\ 0.1 \end{bmatrix},\ \boldsymbol{y}^{(3)} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix},\ \boldsymbol{a}^{(3)} = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.7 \end{bmatrix}$$

Đây có phải là một mô hình tốt? Khả năng mô hình này dự đoán đúng 3 điểm dữ liệu trên có cao?

Hồi quy softmax – Hàm mất mát





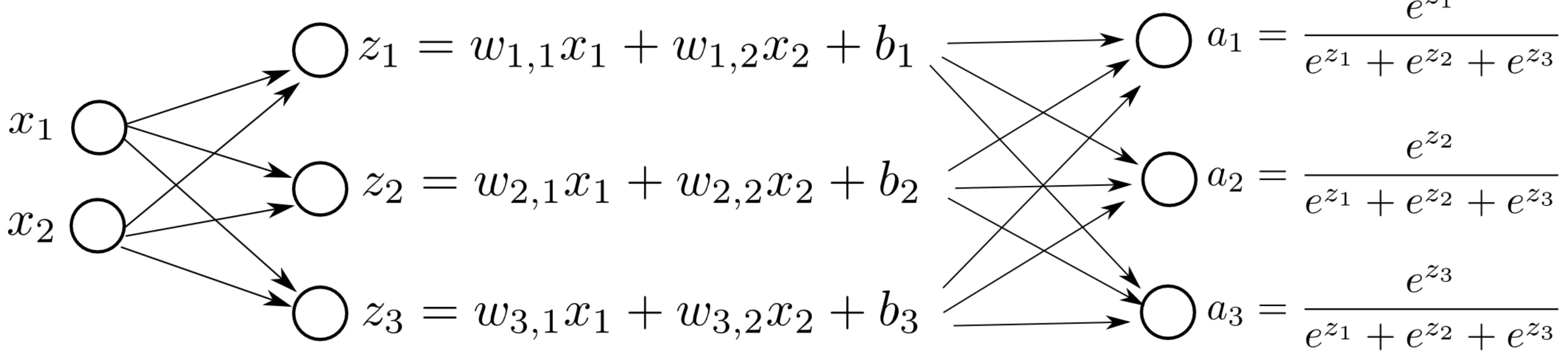
Ta cần học giá trị ma trận W và vector $\boldsymbol{b}$ để phân biệt tốt các điểm dữ liệu đỏ , xanh lá, xanh dương trong tập dữ liệu huấn luyện.

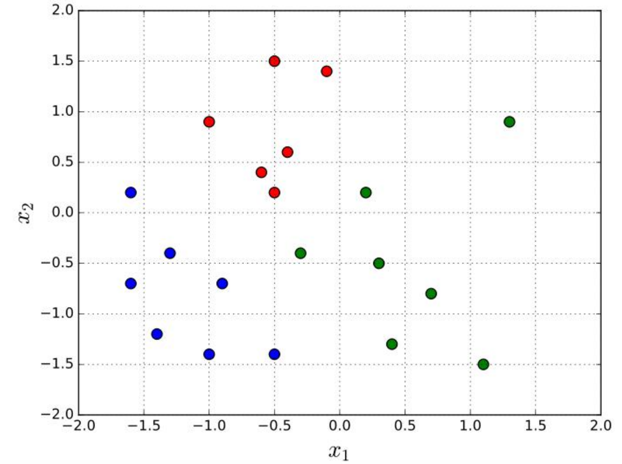
Ví dụ: Nếu một điểm dữ liệu có nhãn 1

$\boldsymbol{y}^{(i)} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$ thì nếu mô hình dự đoán $\boldsymbol{a}^{(i)} = \begin{bmatrix} 0.1 \\ 0.8 \\ 0.1 \end{bmatrix}$ sẽ tốt hơn một mô hình khác dự đoán $\boldsymbol{a}^{(i)} = \begin{bmatrix} 0.2 \\ 0.6 \\ 0.2 \end{bmatrix}$

Ta muốn mô hình (bộ phân lớp – classifier) trả về dự đoán $\boldsymbol{a}$ giống với biểu diễn one-hot vector của nhãn $\boldsymbol{y}$ của mỗi điểm dữ liệu.

Hồi quy softmax – Hàm mất mát





Khả năng của mô hình với bộ giá trị tham số W, b dự đoán $\boldsymbol{a}^{(i)}$ khớp với nhãn one-hot vector $\boldsymbol{y}^{(i)}$ của điểm dữ liệu thứ i trong tập dữ liệu là:

$$\prod_{j=1}^{3} \left(a_j^{(i)}\right)^{y_j^{(i)}}$$

Ví dụ: với điểm dữ liệu có nhãn $\boldsymbol{y}^{(i)} = (1,0,0)$ và mô hình dự đoán $\boldsymbol{a}^{(i)} = (0.9,0.1,0.0)$, ta có:

$$\prod_{j=1}^{3} \left(a_j^{(i)}\right)^{y_j^{(i)}} = 0.9^1 \times 0.1^0 \times 0.0^0 = 0.9$$

Ước lượng hợp lý cực đại – Cross Entropy Loss

- Ta cần học giá trị ma trận W và vector $\boldsymbol{b}$ để phân biệt tốt các điểm dữ liệu đỏ, xanh lá, xanh dương trong tập dữ liệu huấn luyện.
- Ta cần tìm giá trị ma trận W và vector $\boldsymbol{b}$ để cực đại hóa khả năng mô hình thực hiện dự đoán $\boldsymbol{a}^{(i)}$ khớp với nhãn $\boldsymbol{y}^{(i)}$ của tất cả N điểm dữ liệu trong tập huấn luyện:

$$\widehat{W}, \widehat{\boldsymbol{b}} = \underset{W,\boldsymbol{b}}{\arg\max} \prod_{i=1}^{N} \prod_{j=1}^{3} \left(a_j^{(i)}\right)^{y_j^{(i)}} \quad \text{với } \boldsymbol{a}^{(i)} = \text{softmax}\left(W\boldsymbol{x}^{(i)} + \boldsymbol{b}\right)$$

$$= \underset{W,\boldsymbol{b}}{\arg\max} \log\left(\prod_{i=1}^{N} \prod_{j=1}^{3} \left(a_j^{(i)}\right)^{y_j^{(i)}}\right) = \underset{W,\boldsymbol{b}}{\arg\max} \sum_{i=1}^{N} \sum_{j=1}^{3} \log\left(a_j^{(i)}\right)^{y_j^{(i)}} = \underset{W,\boldsymbol{b}}{\arg\max} \sum_{i=1}^{N} \sum_{j=1}^{3} y_j^{(i)} \log\left(a_j^{(i)}\right)$$

$$= \underset{W,\boldsymbol{b}}{\arg\min} - \sum_{i=1}^{N} \sum_{j=1}^{3} y_j^{(i)} \log\left(a_j^{(i)}\right) = \underset{W,\boldsymbol{b}}{\arg\min} \sum_{i=1}^{N} L\left(W, \boldsymbol{b}, \boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}\right) = \underset{W,\boldsymbol{b}}{\arg\min}\, J(W, \boldsymbol{b})$$

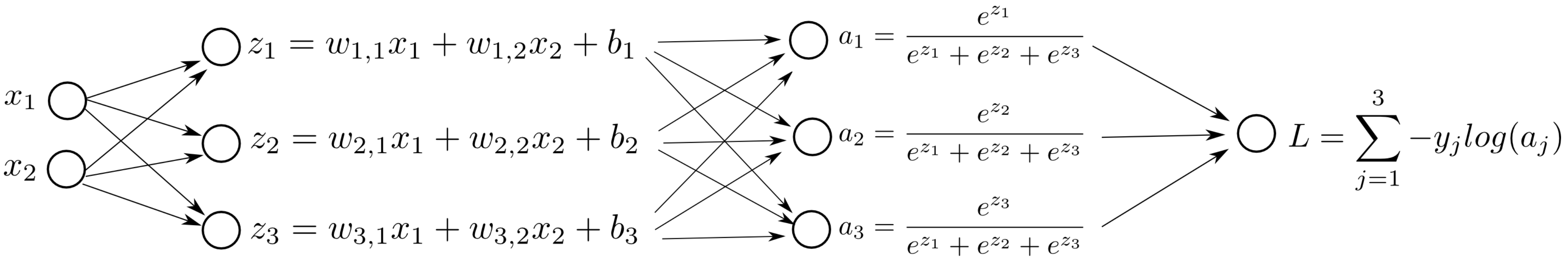
- Đây là hàm mất mát Cross Entropy (CE) Loss - trường hợp tổng quát của hàm Binary Cross Entropy (BCE) Loss trong hồi quy logistic.

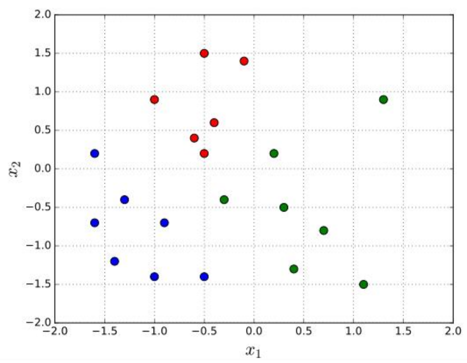
Hồi quy softmax – Cross Entropy Loss

$$\boldsymbol{z} = W\boldsymbol{x} + \boldsymbol{b}$$

$$\boldsymbol{a} = \text{softmax}(\boldsymbol{z})$$

Hàm mất mát Cross Entropy (CE) so sánh xác suất dự đoán của mô hình $\boldsymbol{a}$ với nhãn của một điểm dữ liệu $\boldsymbol{y}$:

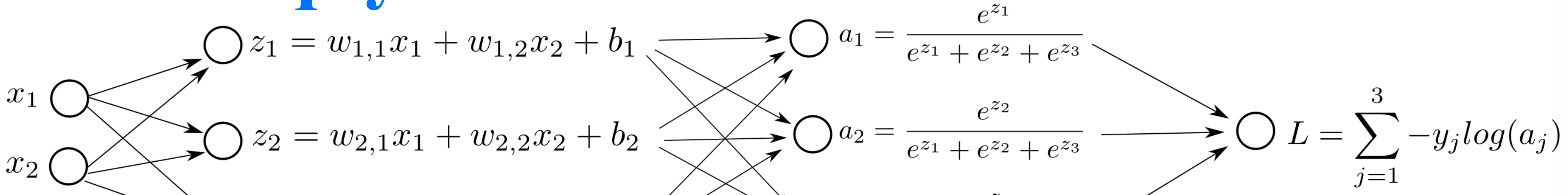


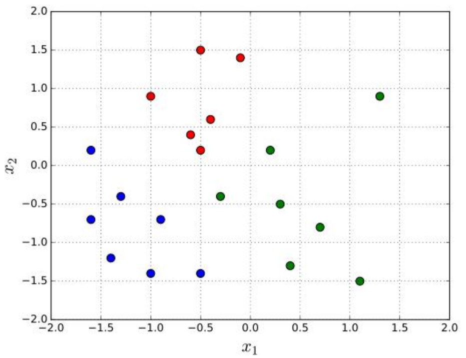


Trong trường hợp tổng quát với C lớp và N điểm dữ liệu, hàm Cross Entropy (CE) Loss là:

$$J = -\sum_{i=1}^{N}\sum_{j=1}^{C} y_j^{(i)} \log\left(a_j^{(i)}\right)$$

Hồi quy softmax – Gradient Descent





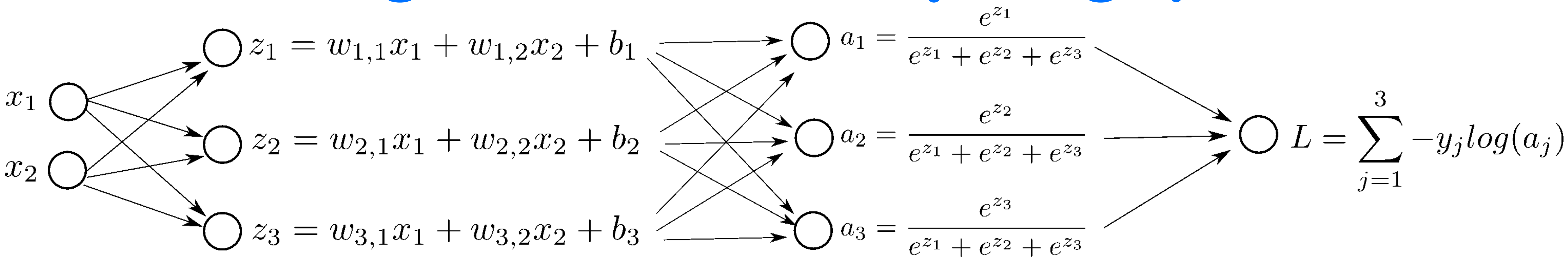
Ta cần học giá trị ma trận W và vector $\boldsymbol{b}$ để phân biệt tốt các điểm dữ liệu đỏ , xanh lá, xanh dương trong tập dữ liệu huấn luyện.

- Khởi tạo với bộ tham số ngẫu nhiên W và $\boldsymbol{b}$.
- Cải thiện chất lượng mô hình bằng cách cập nhật W và $\boldsymbol{b}$ qua từng vòng lặp:

$$W = W - \alpha \frac{\partial L}{\partial W}$$

$$\boldsymbol{b} = \boldsymbol{b} - \alpha \frac{\partial L}{\partial \boldsymbol{b}}$$

Tính toán gradient – Lan truyền ngược



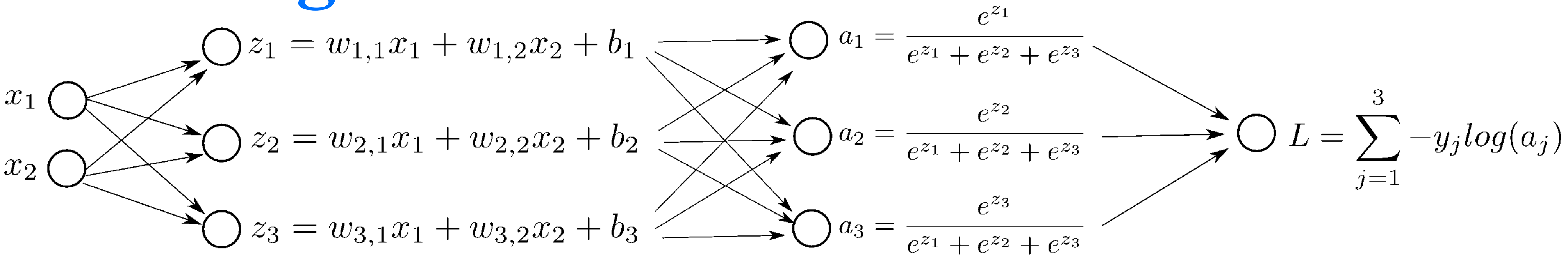
Áp dụng quy tắc dây chuyền (chain rule), ta thực hiện lan truyền ngược để tính đạo hàm của hàm mất mát theo các tham số W và $\boldsymbol{b}$ của mô hình. Ví dụ:

$$\frac{\partial L}{\partial w_{2,1}} = \sum_{i=1}^{3} \frac{\partial L}{\partial a_i} \frac{\partial a_i}{\partial z_2} \frac{\partial z_2}{\partial w_{2,1}}$$

Vậy ta có:

$$\frac{\partial L}{\partial w_{m,n}} = \sum_{i=1}^{3} \frac{\partial L}{\partial a_i} \frac{\partial a_i}{\partial z_m} \frac{\partial z_m}{\partial w_{m,n}}, \quad \frac{\partial L}{\partial b_m} = \sum_{i=1}^{3} \frac{\partial L}{\partial a_i} \frac{\partial a_i}{\partial z_m} \frac{\partial z_m}{\partial b_m}$$

Tính toán gradient



$$\frac{\partial L}{\partial w_{m,n}} = \sum_{i=1}^{3} \frac{\partial L}{\partial a_i}\frac{\partial a_i}{\partial z_m}\frac{\partial z_m}{\partial w_{m,n}}, \quad \frac{\partial L}{\partial b_m} = \sum_{i=1}^{3} \frac{\partial L}{\partial a_i}\frac{\partial a_i}{\partial z_m}\frac{\partial z_m}{\partial b_m}$$

$$\frac{\partial z_m}{\partial w_{m,n}} = \frac{\partial}{\partial w_{m,n}}\left(w_{m,1}x_1 + w_{m,2}x_2 + \ldots + w_{m,n}x_n + \ldots + w_{m,d}x_d + b_m\right) = x_n$$

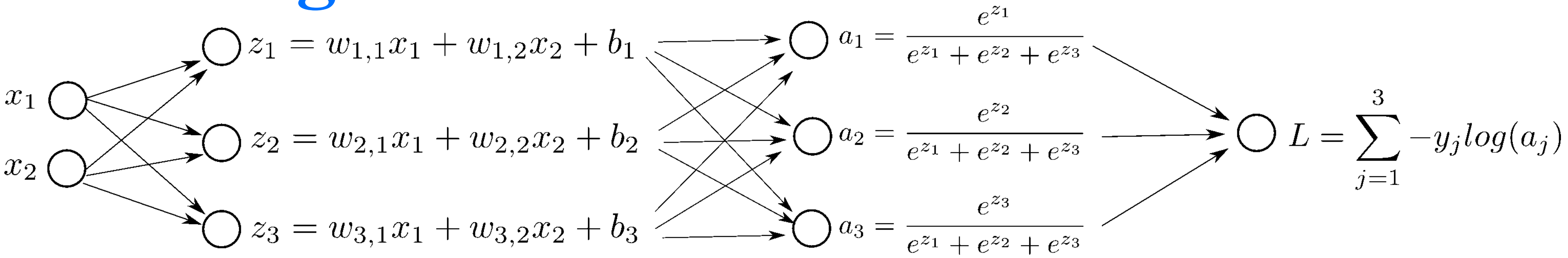
$$\frac{\partial z_m}{\partial b_m} = \frac{\partial}{\partial b_m}\left(w_{m,1}x_1 + w_{m,2}x_2 + \ldots + w_{m,n}x_n + \ldots + w_{m,d}x_d + b_m\right) = 1$$

$$\frac{\partial z_1}{\partial w_{1,1}} = x_1, \quad \frac{\partial z_1}{\partial w_{1,2}} = x_2, \quad \frac{\partial z_1}{\partial b_1} = 1$$

$$\frac{\partial z_2}{\partial w_{2,1}} = x_1, \quad \frac{\partial z_2}{\partial w_{2,2}} = x_2, \quad \frac{\partial z_2}{\partial b_2} = 1$$

$$\frac{\partial z_3}{\partial w_{3,1}} = x_1, \quad \frac{\partial z_3}{\partial w_{3,2}} = x_2, \quad \frac{\partial z_3}{\partial b_3} = 1$$

Tính toán gradient

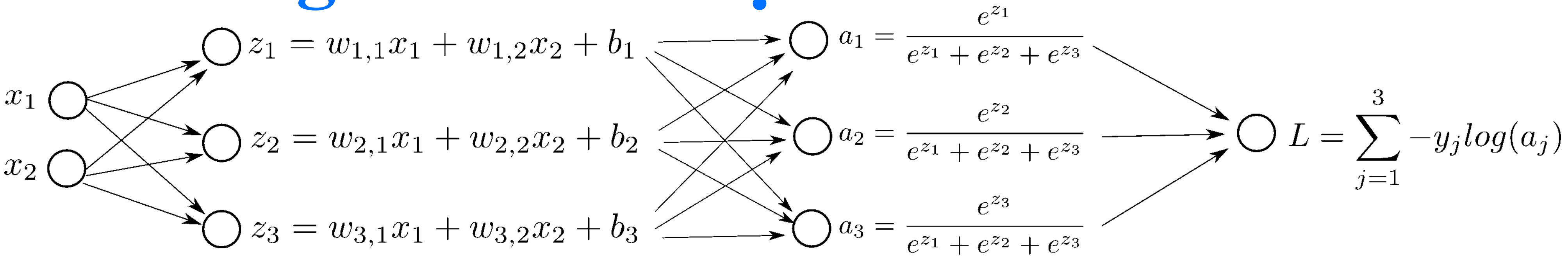


$$\frac{\partial L}{\partial w_{m,n}} = \sum_{i=1}^{3} \frac{\partial L}{\partial a_i} \frac{\partial a_i}{\partial z_m} \frac{\partial z_m}{\partial w_{m,n}}, \quad \frac{\partial L}{\partial b_m} = \sum_{i=1}^{3} \frac{\partial L}{\partial a_i} \frac{\partial a_i}{\partial z_m} \frac{\partial z_m}{\partial b_m}$$

$$\frac{\partial L}{\partial a_i} = \frac{\partial}{\partial a_i}\left(-\sum_{j=1}^{C} y_j \log\left(a_j\right)\right) = \frac{\partial}{\partial a_i}(-\,y_i \log\left(a_i\right)) = \frac{-\,y_i}{a_i}$$

$$\frac{\partial L}{\partial \boldsymbol{a}} = \begin{bmatrix} \frac{\partial L}{\partial a_1} \\ \frac{\partial L}{\partial a_2} \\ \frac{\partial L}{\partial a_3} \end{bmatrix} = \begin{bmatrix} \frac{-\,y_1}{a_1} \\ \frac{-\,y_2}{a_2} \\ \frac{-\,y_3}{a_3} \end{bmatrix}$$

Tính toán gradient – Đạo hàm softmax



$$\frac{\partial L}{\partial w_{m,n}} = \sum_{i=1}^{3} \frac{\partial L}{\partial a_i} \frac{\partial a_i}{\partial z_m} \frac{\partial z_m}{\partial w_{m,n}}, \quad \frac{\partial L}{\partial b_m} = \sum_{i=1}^{3} \frac{\partial L}{\partial a_i} \frac{\partial a_i}{\partial z_m} \frac{\partial z_m}{\partial b_m}$$

$$\frac{\partial a_i}{\partial z_m} = \frac{\partial}{\partial z_m}\left(\frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}\right)$$

Ta có:

$$\frac{d}{dx}\frac{f(x)}{g(x)} = \frac{f'(x)g(x) - g'(x)f(x)}{(g(x))^2}$$

Với e^{z_i} đóng vai trò $f(x)$ và $\sum_{j=1}^{C} e^{z_j}$ đóng vai trò $g(x)$.

Ta có 2 trường hợp $i = m$ và $i \neq m$.

Tính toán gradient – Đạo hàm softmax

- Trường hợp $i = m$

$$\frac{\partial a_i}{\partial z_m} = \frac{\partial}{\partial z_m}\left(\frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}\right) = \frac{(e^{z_i})'\left(\sum_{j=1}^{C} e^{z_j}\right) - \left(\sum_{j=1}^{C} e^{z_j}\right)' e^{z_i}}{\left(\sum_{j=1}^{C} e^{z_j}\right)^2}$$

$$= \frac{e^{z_i}\left(\sum_{j=1}^{C} e^{z_j}\right) - (e^{z_m})e^{z_i}}{\left(\sum_{j=1}^{C} e^{z_j}\right)^2}$$

$$= \frac{e^{z_i}\left(\sum_{j=1}^{C} e^{z_j} - e^{z_m}\right)}{\left(\sum_{j=1}^{C} e^{z_j}\right)^2} = \frac{e^{z_m}\left(\sum_{j=1}^{C} e^{z_j} - e^{z_m}\right)}{\left(\sum_{j=1}^{C} e^{z_j}\right)^2}$$

$$= \frac{e^{z_m}}{\sum_{j=1}^{C} e^{z_j}} \frac{\left(\sum_{j=1}^{C} e^{z_j} - e^{z_m}\right)}{\sum_{j=1}^{C} e^{z_j}} = \frac{e^{z_m}}{\sum_{j=1}^{C} e^{z_j}}\left(\frac{\sum_{j=1}^{C} e^{z_j}}{\sum_{j=1}^{C} e^{z_j}} - \frac{e^{z_m}}{\sum_{j=1}^{C} e^{z_j}}\right)$$

$$= a_m(1 - a_m)$$

Tính toán gradient – Đạo hàm softmax

➢ Trường hợp $i \neq m$

$$\frac{\partial a_i}{\partial z_m} = \frac{\partial}{\partial z_m}\left(\frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}\right) = \frac{(e^{z_i})'\left(\sum_{j=1}^{C} e^{z_j}\right) - \left(\sum_{j=1}^{C} e^{z_j}\right)' e^{z_i}}{\left(\sum_{j=1}^{C} e^{z_j}\right)^2}$$

$$= \frac{0 - (e^{z_m})e^{z_i}}{\left(\sum_{j=1}^{C} e^{z_j}\right)^2}$$

$$= \frac{-(e^{z_m})e^{z_i}}{\left(\sum_{j=1}^{C} e^{z_j}\right)^2} =$$

$$= -\frac{e^{z_m}}{\sum_{j=1}^{C} e^{z_j}} \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}$$

$$= -a_m a_i$$

Tính toán gradient – Đạo hàm softmax

$$\frac{\partial a_i}{\partial z_m} = \begin{cases} a_m(1 - a_m), & \text{nếu } i = m \\ -\,a_m a_i, & \text{nếu } i \neq m \end{cases}$$

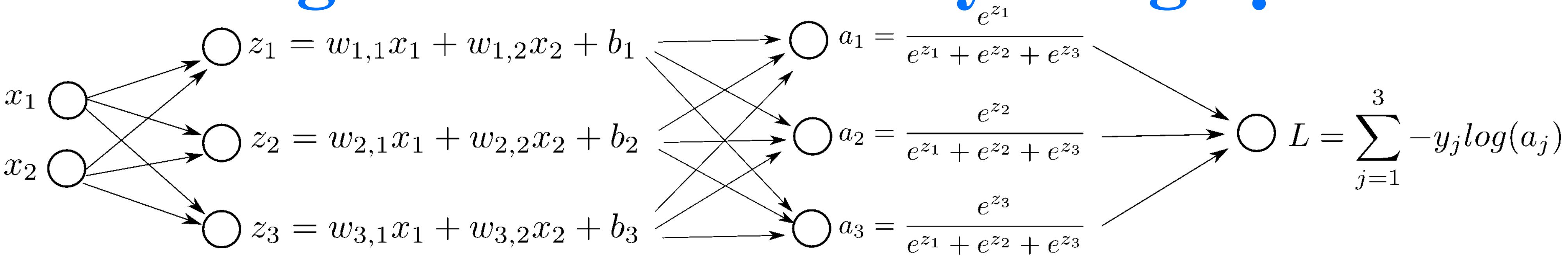
$$\frac{\partial \boldsymbol{a}}{\partial \boldsymbol{z}} = \begin{bmatrix} \frac{\partial a_1}{\partial z_1} & \frac{\partial a_2}{\partial z_1} & \frac{\partial a_3}{\partial z_1} \\ \frac{\partial a_1}{\partial z_2} & \frac{\partial a_2}{\partial z_2} & \frac{\partial a_3}{\partial z_2} \\ \frac{\partial a_1}{\partial z_3} & \frac{\partial a_2}{\partial z_3} & \frac{\partial a_3}{\partial z_3} \end{bmatrix} = \begin{bmatrix} a_1(1 - a_1) & a_1(-\,a_2) & a_1(-\,a_3) \\ a_2(-\,a_1) & a_2(1 - a_2) & a_2(-\,a_3) \\ a_3(-\,a_1) & a_3(-\,a_2) & a_3(1 - a_3) \end{bmatrix}$$

$$= \begin{bmatrix} a_1 & a_1 & a_1 \\ a_2 & a_2 & a_2 \\ a_3 & a_3 & a_3 \end{bmatrix} \circ \left(\begin{bmatrix} (1 - a_1) & (-\,a_2) & (-\,a_3) \\ (-\,a_1) & (1 - a_2) & 0 \\ (-\,a_1) & (-\,a_2) & (1 - a_3) \end{bmatrix} \right)$$

$$= \begin{bmatrix} a_1 & a_1 & a_1 \\ a_2 & a_2 & a_2 \\ a_3 & a_3 & a_3 \end{bmatrix} \circ \left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} a_1 & a_2 & a_3 \\ a_1 & a_2 & a_3 \\ a_1 & a_2 & a_3 \end{bmatrix} \right)$$

$$= \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \circ \left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix} \right) = \boldsymbol{a}\boldsymbol{1}^T \circ (\mathbf{I} - \boldsymbol{1}\boldsymbol{a}^T)$$

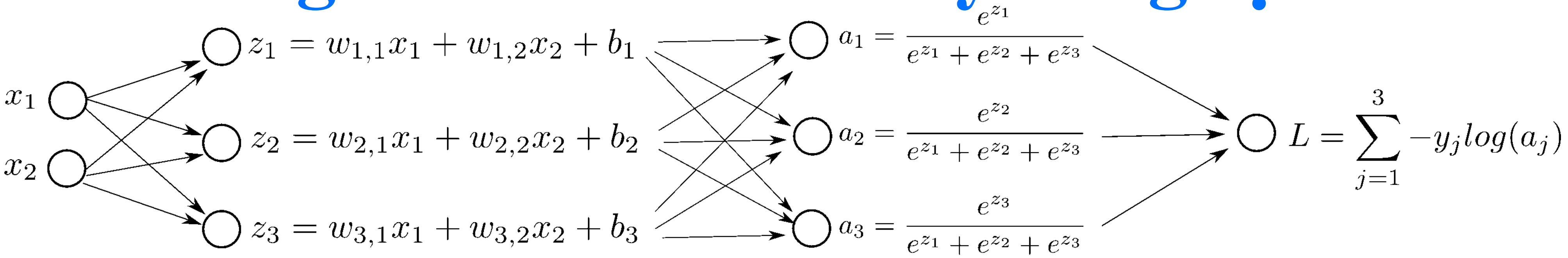
Tính toán gradient – Lan truyền ngược



$$\frac{\partial L}{\partial W} = \begin{bmatrix} \frac{\partial L}{\partial w_{1,1}} & \frac{\partial L}{\partial w_{1,2}} \\ \frac{\partial L}{\partial w_{2,1}} & \frac{\partial L}{\partial w_{2,2}} \\ \frac{\partial L}{\partial w_{3,1}} & \frac{\partial L}{\partial w_{3,2}} \end{bmatrix} = \begin{bmatrix} \frac{\partial L}{\partial z_1}\frac{\partial z_1}{\partial w_{1,1}} & \frac{\partial L}{\partial z_1}\frac{\partial z_1}{\partial w_{1,2}} \\ \frac{\partial L}{\partial z_2}\frac{\partial z_2}{\partial w_{2,1}} & \frac{\partial L}{\partial z_2}\frac{\partial z_2}{\partial w_{2,2}} \\ \frac{\partial L}{\partial z_3}\frac{\partial z_3}{\partial w_{3,1}} & \frac{\partial L}{\partial z_3}\frac{\partial z_3}{\partial w_{3,2}} \end{bmatrix} = \begin{bmatrix} \frac{\partial L}{\partial z_1}x_1 & \frac{\partial L}{\partial z_1}x_2 \\ \frac{\partial L}{\partial z_2}x_1 & \frac{\partial L}{\partial z_2}x_2 \\ \frac{\partial L}{\partial z_3}x_1 & \frac{\partial L}{\partial z_3}x_2 \end{bmatrix} = \frac{\partial L}{\partial \mathbf{z}}\begin{bmatrix} x_1 & x_2 \end{bmatrix} = \frac{\partial L}{\partial \mathbf{z}}\boldsymbol{x}^T$$

$$\frac{\partial L}{\partial \boldsymbol{b}} = \begin{bmatrix} \frac{\partial L}{\partial b_1} \\ \frac{\partial L}{\partial b_2} \\ \frac{\partial L}{\partial b_3} \end{bmatrix} = \begin{bmatrix} \frac{\partial L}{\partial z_1}\frac{\partial z_1}{\partial b_1} \\ \frac{\partial L}{\partial z_2}\frac{\partial z_3}{\partial b_2} \\ \frac{\partial L}{\partial z_3}\frac{\partial z_3}{\partial b_3} \end{bmatrix} = \begin{bmatrix} \frac{\partial L}{\partial z_1}1 \\ \frac{\partial L}{\partial z_2}1 \\ \frac{\partial L}{\partial z_3}1 \end{bmatrix} = \begin{bmatrix} \frac{\partial L}{\partial z_1} \\ \frac{\partial L}{\partial z_2} \\ \frac{\partial L}{\partial z_3} \end{bmatrix} = \frac{\partial L}{\partial \mathbf{z}}$$

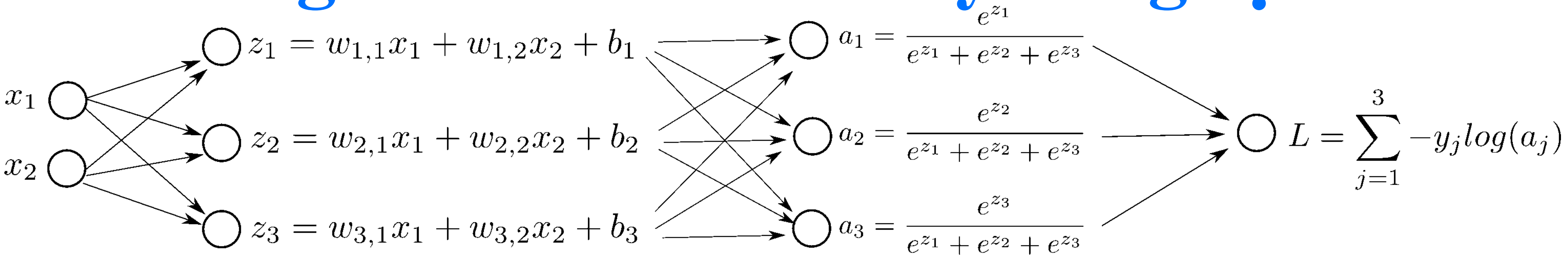
Tính toán gradient – Lan truyền ngược



$$\frac{\partial L}{\partial \mathbf{z}} = \begin{bmatrix} \frac{\partial L}{\partial z_1} \\ \frac{\partial L}{\partial z_2} \\ \frac{\partial L}{\partial z_3} \end{bmatrix} = \begin{bmatrix} \frac{\partial L}{\partial a_1}\frac{\partial a_1}{\partial z_1} + \frac{\partial L}{\partial a_2}\frac{\partial a_2}{\partial z_1} + \frac{\partial L}{\partial a_3}\frac{\partial a_3}{\partial z_1} \\ \frac{\partial L}{\partial a_1}\frac{\partial a_1}{\partial z_2} + \frac{\partial L}{\partial a_2}\frac{\partial a_2}{\partial z_2} + \frac{\partial L}{\partial a_3}\frac{\partial a_3}{\partial z_2} \\ \frac{\partial L}{\partial a_1}\frac{\partial a_1}{\partial z_3} + \frac{\partial L}{\partial a_2}\frac{\partial a_2}{\partial z_3} + \frac{\partial L}{\partial a_3}\frac{\partial a_3}{\partial z_3} \end{bmatrix} = \begin{bmatrix} \frac{\partial a_1}{\partial z_1} & \frac{\partial a_2}{\partial z_1} & \frac{\partial a_3}{\partial z_1} \\ \frac{\partial a_1}{\partial z_2} & \frac{\partial a_2}{\partial z_2} & \frac{\partial a_3}{\partial z_2} \\ \frac{\partial a_1}{\partial z_3} & \frac{\partial a_2}{\partial z_3} & \frac{\partial a_3}{\partial z_3} \end{bmatrix} \begin{bmatrix} \frac{\partial L}{\partial a_1} \\ \frac{\partial L}{\partial a_2} \\ \frac{\partial L}{\partial a_3} \end{bmatrix} = \frac{\partial \mathbf{a}}{\partial \mathbf{z}} \frac{\partial L}{\partial \mathbf{a}}$$

$$\frac{\partial \mathbf{a}}{\partial \mathbf{z}} = \begin{bmatrix} a_1(1 - a_1) & a_1(-a_2) & a_1(-a_3) \\ a_2(-a_1) & a_2(1 - a_2) & a_2(-a_3) \\ a_3(-a_1) & a_3(-a_2) & a_3(1 - a_3) \end{bmatrix}, \quad \frac{\partial L}{\partial \mathbf{a}} = \begin{bmatrix} \frac{-y_1}{a_1} \\ \frac{-y_2}{a_2} \\ \frac{-y_3}{a_3} \end{bmatrix}$$

Tính toán gradient – Lan truyền ngược

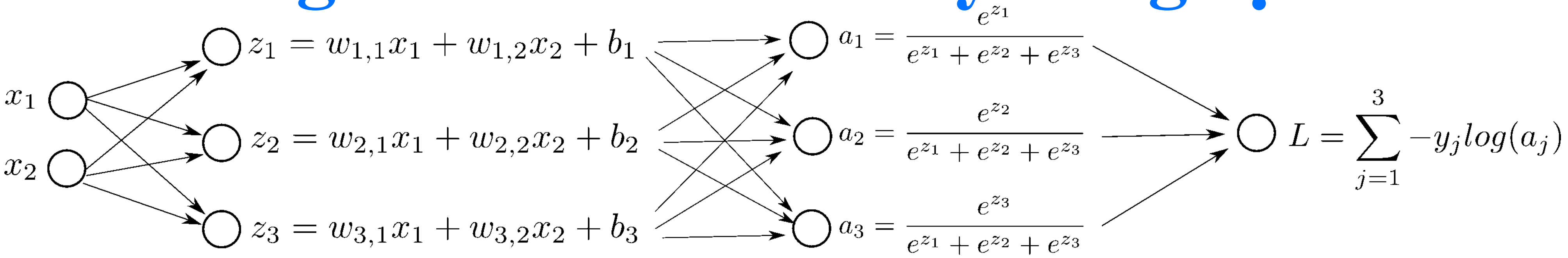


$$\frac{\partial L}{\partial \boldsymbol{z}} = \frac{\partial \boldsymbol{a}}{\partial \boldsymbol{z}}\frac{\partial L}{\partial \boldsymbol{a}} = \begin{bmatrix} a_1(1-a_1) & a_1(-a_2) & a_1(-a_3) \\ a_2(-a_1) & a_2(1-a_2) & a_2(-a_3) \\ a_3(-a_1) & a_3(-a_2) & a_3(1-a_3) \end{bmatrix} \begin{bmatrix} \frac{-y_1}{a_1} \\ \frac{-y_2}{a_2} \\ \frac{-y_3}{a_3} \end{bmatrix}$$

$$= \begin{bmatrix} -y_1(1-a_1) + y_2a_1 + y_3a_1 \\ y_1a_2 - y_2(1-a_2) + y_3a_2 \\ y_1a_3 + y_2a_3 - y_3(1-a_3) \end{bmatrix} = \begin{bmatrix} (a_1-1) & a_1 & a_1 \\ a_2 & (a_2-1) & a_2 \\ a_3 & a_3 & (a_3-1) \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

$$= \left(\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = (\boldsymbol{a}\mathbf{1}^T - \mathbf{I})\boldsymbol{y} = \boldsymbol{a}\mathbf{1}^T\boldsymbol{y} - \mathbf{I}\boldsymbol{y} = \boldsymbol{a} - \boldsymbol{y}$$

Tính toán gradient – Lan truyền ngược



Gradient của Cross Entropy (CE) Loss của hồi quy softmax (xét trên 1 điểm dữ liệu):

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial \boldsymbol{z}}[x_1 \quad x_2] = \frac{\partial L}{\partial \boldsymbol{z}}\boldsymbol{x}^T = (\boldsymbol{a} - \boldsymbol{y})\boldsymbol{x}^T$$

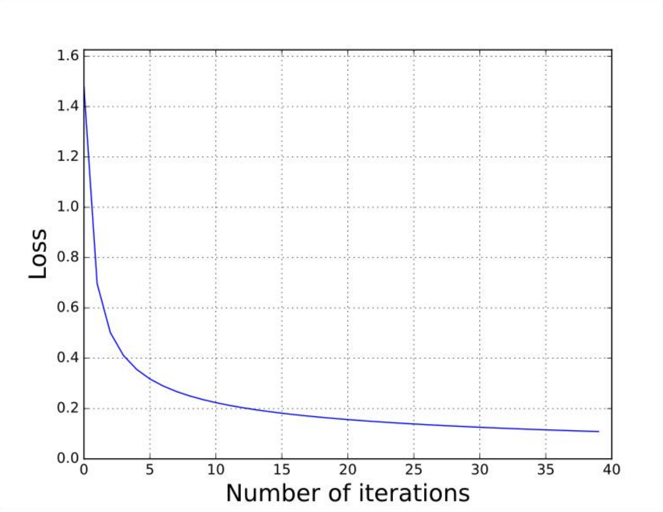
$$\frac{\partial L}{\partial \boldsymbol{b}} = \frac{\partial L}{\partial \boldsymbol{z}} = (\boldsymbol{a} - \boldsymbol{y})$$

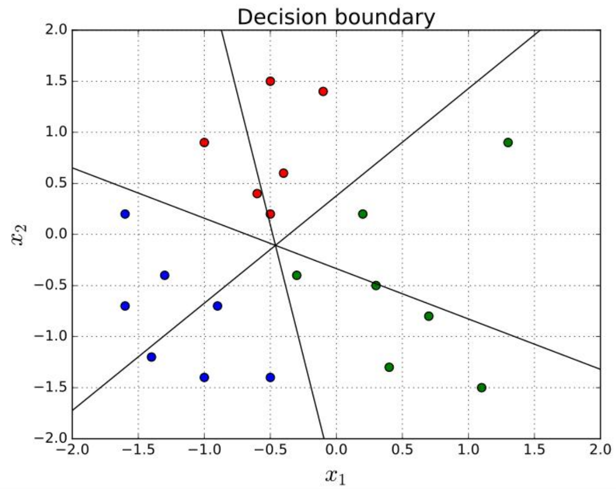
Gradient (xét trên cả tập dữ liệu):

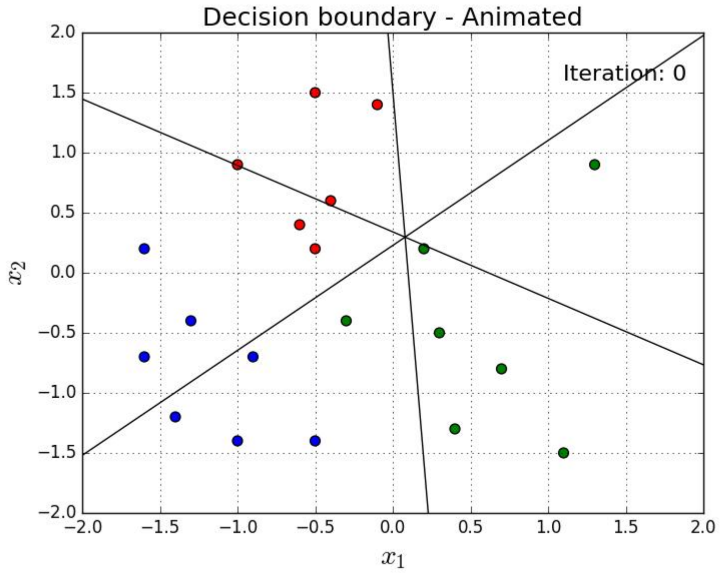
$$\frac{\partial J}{\partial W} = \sum_{i=1}^{N} \left(\boldsymbol{a}^{(i)} - \boldsymbol{y}^{(i)}\right) \boldsymbol{x}^{(i)^T}, \quad \frac{\partial J}{\partial \boldsymbol{b}} = \sum_{i=1}^{N} \left(\boldsymbol{a}^{(i)} - \boldsymbol{y}^{(i)}\right)$$

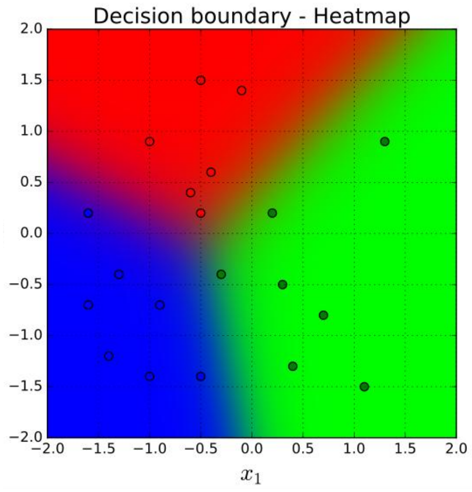
Gradient Descent

```
learning_rate = 2.0
num_epochs = 40
def compute_gradient(x, y, z, a):
    dz = a - y
    dW = np.matmul(dz, x.T)
    db = dz.copy()
    return dW, db

def gradient_descent(W, b, dW, db, learning_rate):
    W = W - learning_rate * dW
    b = b - learning_rate * db
    return W, b

W = np.random.rand(3,2)
b = np.zeros((3,1))
for i in range(num_epochs):
    dW = np.zeros(W.shape)
    db = np.zeros(b.shape)
    L = 0
    for j in range(X.shape[0]):
        x_j = X[j,:].reshape(2,1)
        y_j = Y[j,:].reshape(3,1)
        z_j, a_j = forward(W, b, x_j)
        loss_j = compute_loss(y_j, z_j)
        dW_j, db_j = compute_gradient(x_j, y_j, z_j, a_j)

        dW += dW_j
        db += db_j
        L += loss_j
    dW = (1.0/X.shape[0]) * dW
    db = (1.0/X.shape[0]) * db
    L = (1.0/X.shape[0]) * L
    W, b = gradient_descent(W, b, dW, db, learning_rate)
```





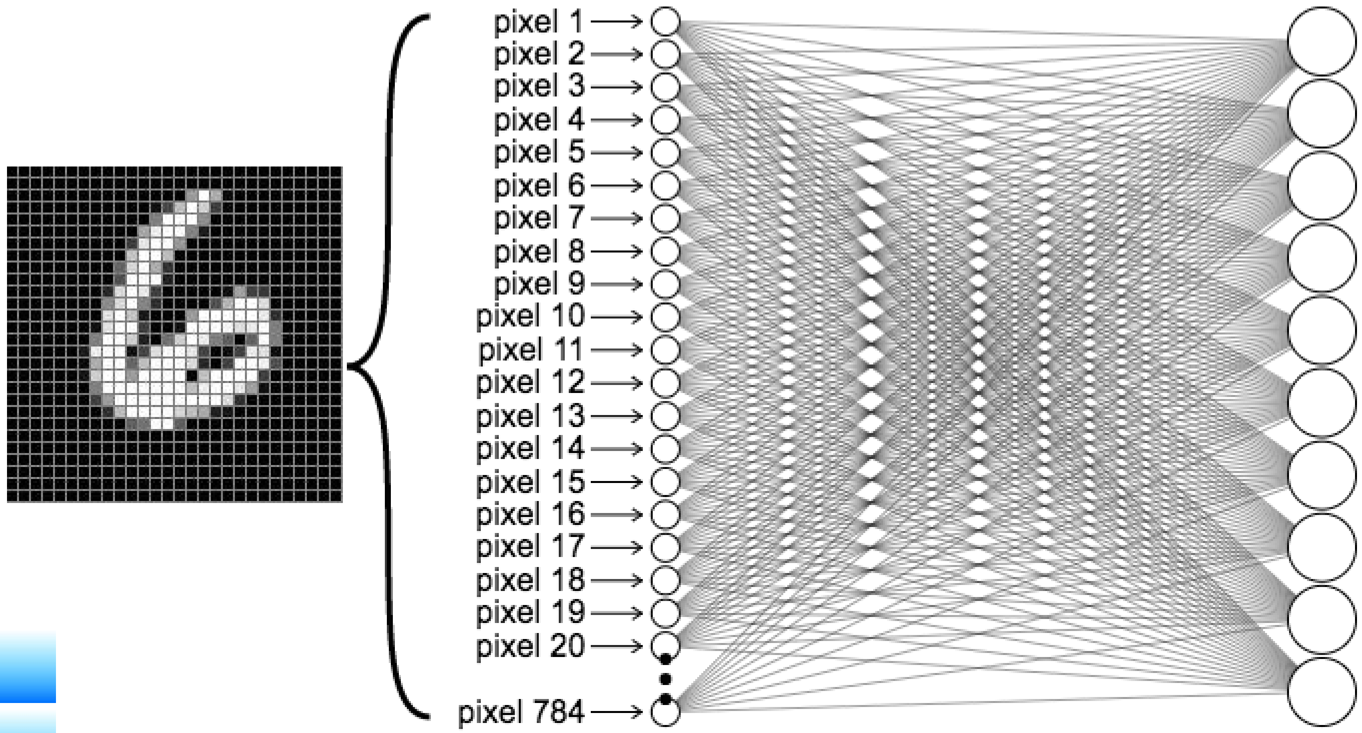




CÀI ĐẶT & PHÂN TÍCH

IMPLEMENTATION & ANALYSIS

MNIST – Phân loại chữ số viết tay




```python
import torch
import torch.nn as nn
import torch.optim as optim
import torch.nn.functional as F
from torch.utils.data import DataLoader
import torchvision.datasets as datasets
import torchvision.transforms as transforms
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

# Create a Softmax Regression Model
class SoftmaxRegression(nn.Module):
    def __init__(self, input_size, num_classes):  # digits 0, 1, 2, ..., 9
        super(SoftmaxRegression, self).__init__()
        self.layer = nn.Linear(in_features=input_size, out_features=num_classes)

    def forward(self, x):
        out = self.layer(x)
        # out = torch.softmax(out)
        return out

# Hyperparameters
input_size = 784 # 28x28 image
num_classes = 10 # 10 digits 0, 1, 2, ...., 9
learning_rate = 0.001
batch_size = 64
num_epochs = 5

model = SoftmaxRegression(input_size=input_size,
                          num_classes=num_classes).to(device)
criterion = nn.CrossEntropyLoss()  # In PyTorch, Softmax + NLL
optimizer = optim.SGD(model.parameters(), lr=learning_rate)
```

MNIST – Phân loại chữ số viết tay

```
# Load the dataset to our Google Drive
train_dataset = datasets.MNIST(root='/content/drive/MyDrive/datasets/mnist',
                    train=True, transform=transforms.ToTensor(), download=True)
test_dataset = datasets.MNIST(root='/content/drive/MyDrive/datasets/mnist',
                    train=False, transform=transforms.ToTensor(), download=True)
train_loader = DataLoader(dataset=train_dataset, batch_size=batch_size, shuffle=True)
test_loader = DataLoader(dataset=test_dataset, batch_size=batch_size, shuffle=False)

# Train our Softmax Regression model
for epoch in range(num_epochs):
  for batch_idx, (data, targets) in enumerate(train_loader):
    data = data.to(device) # Put images to the GPU if GPU is available
    targets = targets.to(device) # Put labels to the GPU if available
    # (64, 1, 28, 28) ----> (64, 784)
    data = data.reshape(data.shape[0], -1)

    # forward pass
    scores = model(data)  # Pass one batch of training examples.
    loss = criterion(scores, targets) # Compute the loss function value J

    # backward pass
    optimizer.zero_grad()  # Empty the memory
    loss.backward()  # Compute the gradient dJ/dw
    optimizer.step()  # One step of gradient descent

    if (batch_idx+1) % 100 == 0:
      print(f'Epoch {epoch+1}/{num_epochs}, Batch {batch_idx+1}, Loss {loss.item():.2f}')
```

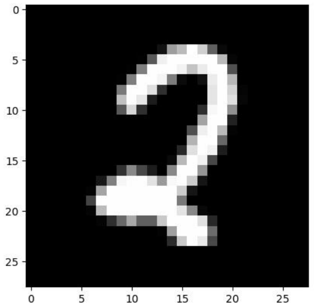
```
Epoch 1/5, Batch 100, Loss 2.25
Epoch 1/5, Batch 200, Loss 2.10
Epoch 1/5, Batch 300, Loss 2.00
Epoch 1/5, Batch 400, Loss 1.93
Epoch 1/5, Batch 500, Loss 1.85
Epoch 1/5, Batch 600, Loss 1.87
Epoch 1/5, Batch 700, Loss 1.72
Epoch 1/5, Batch 800, Loss 1.64
Epoch 1/5, Batch 900, Loss 1.61
Epoch 2/5, Batch 100, Loss 1.53
.......................................................
Epoch 5/5, Batch 100, Loss 0.97
Epoch 5/5, Batch 200, Loss 0.89
Epoch 5/5, Batch 300, Loss 0.96
Epoch 5/5, Batch 400, Loss 0.90
Epoch 5/5, Batch 500, Loss 0.86
Epoch 5/5, Batch 600, Loss 0.70
Epoch 5/5, Batch 700, Loss 0.89
Epoch 5/5, Batch 800, Loss 0.80
Epoch 5/5, Batch 900, Loss 0.73
```

MNIST – Phân loại chữ số viết tay

```
# Evaluate the accuracy of the trained model
def check_accuracy(loader, model):
    num_corrects = 0
    num_samples = 0

    model.eval() # Put model into evaluation mode
    with torch.no_grad():
        for data, targets in loader:
            data = data.to(device)
            targets = targets.to(device)
            data = data.reshape(data.shape[0], -1)

            # forward
            scores = model(data)  # Scores of a batch ---> in the form of (64, 10)
            #image 1: [s0, s1, s2, ...., s9]  ---> s2, 2
            #image 2: [s0, s1, s2, ...., s9]  ---> s10, 10
            #.....
            #image 64:[s0, s1, s2, ...., s9] ---> s7, 7
            _, preds = scores.max(1)

            num_corrects += (preds == targets).sum()
            num_samples += preds.size(0)

        print(f'We get {num_corrects}/{num_samples} correct. Accuracy =
                {float(num_corrects)/float(num_samples)*100.0:.2f}')

check_accuracy(test_loader, model)
```

```
We get 8477/10000 correct. Accuracy = 84.77
```

```
import matplotlib.pyplot as plt
image, label = test_dataset[1802]
plt.imshow(image.squeeze(), cmap='gray')
```




```
image = image.to(device) # Put the image to the GPU if available
image = image.reshape(image.shape[0], -1) # Reshape the tensor

score = model(image)
print(score)
_, preds = score.max(1)
print(f'Our prediction is {preds.item()}')
```

```
tensor([[-0.6124, -1.3164, 2.3769, -1.1956, -0.4008, -1.1456,
0.4172, 0.2381, 1.4189, -0.6130]], device='cuda:0',
grad_fn=<AddmmBackward0>)
Our prediction is 2
```

```
torch.nn.functional.softmax(score, dim=1)
```

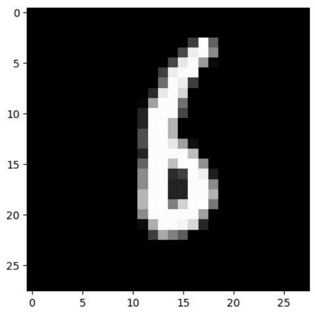
```
tensor([[0.0267, 0.0132, 0.5298, 0.0149, 0.0329, 0.0156,
0.0746, 0.0624, 0.2032, 0.0266]], device='cuda:0',
grad_fn=<SoftmaxBackward0>)
```

MNIST – Phân loại chữ số viết tay

```
# Evaluate the accuracy of the trained model
def check_accuracy(loader, model):
    num_corrects = 0
    num_samples = 0

    model.eval() # Put model into evaluation mode
    with torch.no_grad():
        for data, targets in loader:
            data = data.to(device)
            targets = targets.to(device)
            data = data.reshape(data.shape[0], -1)

            # forward
            scores = model(data)  # Scores of a batch ---> (64, 10)
            #image 1: [s0, s1, s2, ...., s9]  ---> s2, 2
            #image 2: [s0, s1, s2, ...., s9]  ---> s10, 10
            #.....
            #image 64:[s0, s1, s2, ...., s9] ---> s7, 7
            _, preds = scores.max(1)

            num_corrects += (preds == targets).sum()
            num_samples += preds.size(0)

        print(f'We get {num_corrects}/{num_samples} correct. Accuracy =
              {float(num_corrects)/float(num_samples)*100.0:.2f}')


check_accuracy(test_loader, model)
```

```
We get 8477/10000 correct. Accuracy = 84.77
```

```
import matplotlib.pyplot as plt
image, label = test_dataset[1100]
plt.imshow(image.squeeze(), cmap='gray’)
```




```
image = image.to(device) # Put the image to the GPU if available
image = image.reshape(image.shape[0], -1) # Reshape the tensor

score = model(image)
print(score)
_, preds = score.max(1)
print(f'Our prediction is {preds.item()}’)
```

```
tensor([[-1.9898, 1.4574, 0.1530, -0.2754, -0.8373, 0.2092,
1.7860, -0.9584, 0.6460, -0.1126]], device='cuda:0',
grad_fn=<AddmmBackward0>)
Our prediction is 6
```

```
torch.nn.functional.softmax(score, dim=1)
```

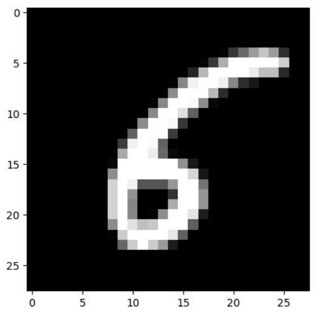
```
tensor([[0.0080, 0.2501, 0.0679, 0.0442, 0.0252, 0.0718,
0.3474, 0.0223, 0.1111, 0.0520]], device='cuda:0',
grad_fn=<SoftmaxBackward0>)
```

MNIST – Phân loại chữ số viết tay

```python
# Evaluate the accuracy of the trained model
def check_accuracy(loader, model):
    num_corrects = 0
    num_samples = 0

    model.eval() # Put model into evaluation mode
    with torch.no_grad():
      for data, targets in loader:
        data = data.to(device)
        targets = targets.to(device)
        data = data.reshape(data.shape[0], -1)

        # forward
        scores = model(data)  # Scores of a batch ---> (64, 10)
        #image 1: [s0, s1, s2, ...., s9]  ---> s2, 2
        #image 2: [s0, s1, s2, ...., s9]  ---> s10, 10
        #.....
        #image 64:[s0, s1, s2, ...., s9] ---> s7, 7
        _, preds = scores.max(1)

        num_corrects += (preds == targets).sum()
        num_samples += preds.size(0)

        print(f'We get {num_corrects}/{num_samples} correct. Accuracy =
              {float(num_corrects)/float(num_samples)*100.0:.2f}')


check_accuracy(test_loader, model)
```

```
We get 8477/10000 correct. Accuracy = 84.77
```

```python
import matplotlib.pyplot as plt
image, label = test_dataset[217]
plt.imshow(image.squeeze(), cmap='gray’)
```




```python
image = image.to(device) # Put the image to the GPU if available
image = image.reshape(image.shape[0], -1) # Reshape the tensor

score = model(image)
print(score)
_, preds = score.max(1)
print(f'Our prediction is {preds.item()}’)
```

```
tensor([[ 0.0747, 0.5930, -0.2984, -0.2330, -0.0943, 1.5958,
0.3566, -1.4400, 0.8870, -0.9277]], device='cuda:0',
grad_fn=<AddmmBackward0>)
Our prediction is 5
```

```python
torch.nn.functional.softmax(score, dim=1)
```

```
tensor([[0.0730, 0.1227, 0.0503, 0.0537, 0.0617, 0.3343,
0.0968, 0.0161, 0.1646, 0.0268]], device='cuda:0',
grad_fn=<SoftmaxBackward0>)
```

Phân tích tham số softmax regression

```
model.state_dict()
OrderedDict([
            ('layer.weight',
              tensor([[ 0.0349, -0.0092,  0.0314,  ...,  0.0069,  0.0223, -0.0041],
                  [ 0.0033,  0.0243, -0.0173,  ..., -0.0064, -0.0092,  0.0318],
                  [ 0.0033, -0.0335,  0.0165,  ...,  0.0218, -0.0354,  0.0228],
                  ...,
                  [ 0.0270, -0.0133,  0.0106,  ...,  0.0226, -0.0202, -0.0300],
                  [-0.0277,  0.0045,  0.0304,  ...,  0.0312, -0.0137, -0.0322],
                  [-0.0027,  0.0342,  0.0262,  ..., -0.0110,  0.0317,  0.0154]],
                 device='cuda:0')),
           ('layer.bias',
             tensor([-0.0714,  0.0465, -0.0165,  0.0134, -0.0157,  0.0391, -
0.0322,  0.0303, -0.0566, -0.0136], device='cuda:0'))])

weight_digit_0 = model.state_dict()['layer.weight'][0]
weight_matrix = weight_digit_0.reshape(28, 28).cpu().numpy()
plt.imshow(weight_matrix, cmap='gray')
```