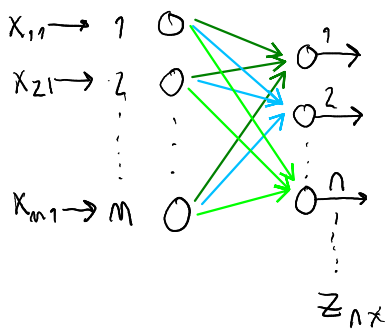


pesos * input + bias = $z_{n\times 1}$

$$W_{n\times m}\, x_{m\times 1} + b_{n\times 1} = z_{n\times 1} \qquad (3.1)$$

Usando un batch para entrenar con p ejemplos a la vez:

$$W_{n\times m}\, x_{m\times p} + [b_{n\times 1} \cdots b_{n\times 1}] = z_{n\times p}$$

Broadcast

$$W_{n\times m} I_{m\times p} + b_{n\times p} = z_{n\times p}$$

batch — p predicciones a la vez

Función de activación softmax a la salida: → probabilidad

$$\hat{y}_k = P(Y=k) = \frac{e^{z_{k1}}}{\sum_j e^{z_{j1}}} \qquad (4.1)$$

→ $k = 1, \ldots, n$

→ $j = 1, \ldots, n$

Usando batch: (4.2)

$$\hat{y}_{ki} = P(Y=k \mid X=x_i) = \frac{e^{z_{ki}}}{\sum_j e^{z_{ji}}}$$

→ $k = 1, \ldots, n$

→ $i = 1, \ldots, p$

→ $j = 1, \ldots, n$

$$\hat{y} = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

$$\sum_k \hat{y}_{ki} = \frac{\sum_k e^{z_{ki}}}{\sum_j e^{z_{ji}}} = 1 \checkmark$$

$\hat{y}_{ki}$ más alto = predicción del ejemplo i.

Loss Function para el ejemplo i = Cross entropy = Xentropy

$$L_i = -\sum_k y_{ki} \ln(\hat{y}_{ki}) \qquad (5.1)$$

y_{ki} valor correcto de la salida

y_{ki} son los elementos de un one-hot vector:

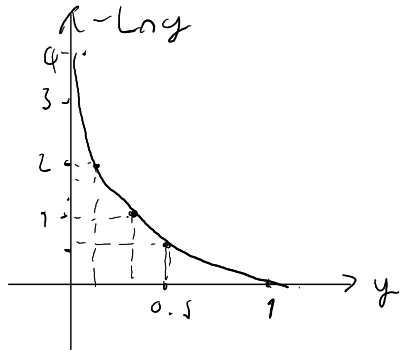
$$y_{ki} = 0, 0, \ldots, 0, 1, 0, \ldots, 0$$

k elemento → clase esperada (etiqueta)

$$y_{ki} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \leftarrow k \text{ elemento} \qquad \leftarrow n \text{ elemento}$$

$$\to L_i = -\ln(\hat{y}_{ki}) = -\ln\left(\frac{e^{z_{ki}}}{\sum_j e^{z_{ji}}}\right) \qquad (5.2)$$

z_{ki} → Da la salida esperada para el ejemplo i



Cost Function para los pesos, bias del batch:

$$J(W,b) = \frac{1}{p}\sum_i L_i \qquad (5.3)$$

$$J(W,b) = \frac{1}{p}\sum_i -\ln\left(\frac{e^{z_{ki}}}{\sum_j e^{z_{ji}}}\right)$$

→ $i = 1, \ldots, p$

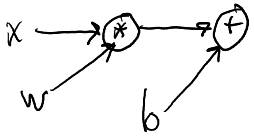
Para una sola neurona, una sola entrada:

$$wx + b = z \qquad (6.1)$$

$$\downarrow$$

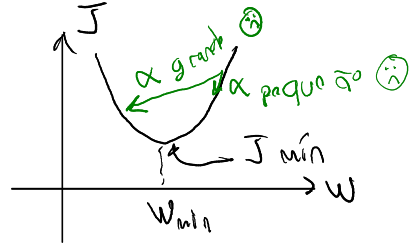
$J(w, b)$.

Gráfica computacional: Cada operación es un nodo



Gradient descent:

$$\rightarrow \frac{\partial J}{\partial w} = \text{Lím} \frac{J(w+h) - J(w)}{h} \qquad (6.2)$$



$$w = w - \alpha \frac{\partial J}{\partial w} \qquad (6.3)$$

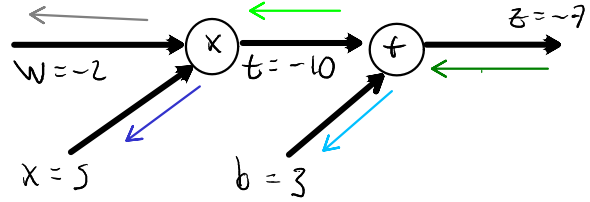
α: learning rate = step size

De igual manera para b:

$$b = b - \alpha \frac{\partial J}{\partial b} \qquad (6.4)$$

Backpropagation:

Gráfica computacional:



$$\rightarrow z = t + b$$
$$z = wx + b$$

Como es la última (y única) capa de la NN:

$$\frac{dz}{dz} = 1 \qquad (7.1)$$

$$\rightarrow \frac{dz}{db} = \frac{dz}{dz}\frac{dz}{db} = 1$$

$$\rightarrow \frac{dz}{dt} = \frac{dz}{dz}\frac{dz}{dt} = 1$$

$$\rightarrow \frac{dz}{dx} = \frac{dz}{dt}\frac{dt}{dx} = w = -2 \qquad (7.2)$$

$$\rightarrow \frac{dz}{dw} = \frac{dz}{dt}\frac{dt}{dw} = x = 5 \qquad (7.3)$$

Ejm: $h = 0.1$ ↑ cambio de w

$$\rightarrow w = w_0 + h = -2 + 0.1 = -1.9$$

$$\rightarrow t = wx_0 = (-1.9)(5) = -9.5$$

$$\rightarrow z = t + b_0 = (-9.5) + (3) = -6.5$$

$$\rightarrow z = z_0 + h\frac{dz}{dw}$$

$$z = -7 + (0.1)5$$

$$z = -6.5$$ ✓

Ejm: $h = 0.1$

↑ Cambio de x

$\rightarrow x = x_0 + h = 5 + 0.1 = 5.1$

$\rightarrow t = w_0 x = (-2)(5.1) = -10.2$

$\rightarrow z = t + b_0 = (-10.2) + (3) = -7.2$

$\rightarrow z = z_0 + h\frac{dz}{dx}$

$z = -7 + (0.1)(-2)$

$z = -7.2$ ✓

Ejm: $h = 0.1$

↑ Cambio de b:

$\rightarrow b = b_0 + h = 3 + (0.1) = 3.1$

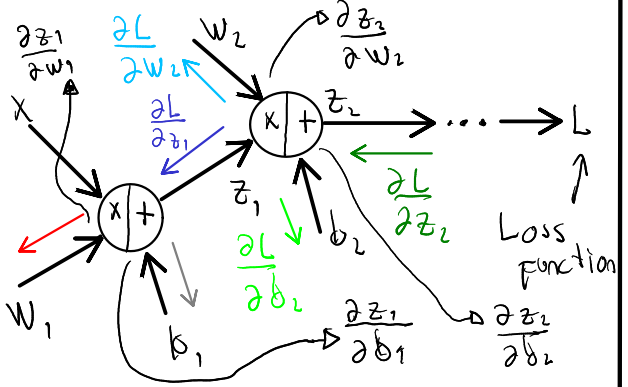
$\rightarrow z = t_0 + b = (-10) + (3.1) = -6.9$

$\rightarrow z = z_0 + h\frac{dz}{db}$

$z = -7 + (0.1)(1) = -6.9$ ✓

Para dos perceptrons:

Gráfica computacional



$\frac{\partial L}{\partial w_2} = ?$ $\frac{\partial L}{\partial b_2} = ?$ $\frac{\partial L}{\partial w_1} = ?$ $\frac{\partial L}{\partial b_1} = ?$

$\rightarrow \frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial z_2}\frac{\partial z_2}{\partial b_2}$ $\rightarrow b_2 = b_2 - \alpha\frac{\partial L}{\partial b_2}$

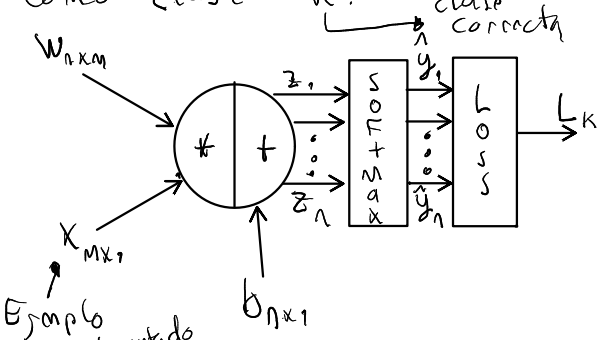
$\rightarrow \frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial z_2}\frac{\partial z_2}{\partial w_2}$ $\rightarrow w_2 = w_2 - \alpha\frac{\partial L}{\partial w_2}$

$\rightarrow \frac{\partial L}{\partial z_1} = \frac{\partial L}{\partial z_1}\frac{\partial z_1}{\partial z_2}$

$\rightarrow \frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial z_1}\frac{\partial z_1}{\partial b_1}$ $\rightarrow b_1 = b_1 - \alpha\frac{\partial L}{\partial b_1}$

$\rightarrow \frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial z_1}\frac{\partial z_1}{\partial w_1}$ $\rightarrow w_1 = w_1 - \alpha\frac{\partial L}{\partial w_1}$

Para un solo ejemplo etiquetado como clase K:



$W_{n\times m}\, x_{m\times 1} + b_{n\times 1} = z_{n\times 1}$

↑ scores

$\rightarrow L = -\sum_K y_K \ln \hat{y}_K$ ← (5.1)

$\uparrow (0\ 0\ \dots\ 1\ \dots\ 0)$

↑ K elemento

$L = -\ln \hat{y}_K = -\ln\left(\frac{e^{z_K}}{\sum_j e^{z_j}}\right)$ ← (5.2)

De la salida esperada para el ejemplo

$j = 1, \dots, n$

$L = -\ln\left(\frac{e^{z_K}}{\sum_j e^{z_j}}\right) = \ln \sum_j e^{z_j} - z_K$

$\rightarrow \frac{\partial L}{\partial z_i} = ?$

$\frac{\partial L}{\partial z_i} = \frac{\partial \ln(\sum_j e^{z_j})}{\partial z_i} - \frac{\partial z_K}{\partial z_i}$

$= \frac{1}{\sum_j e^{z_j}}\frac{\partial}{\partial z_i}\sum_j e^{z_j} - \delta_{iK}$

↑ Delta de Kronecker

$= \frac{e^{z_i}}{\sum_j e^{z_j}} - \delta_{iK}$

$\frac{\partial L}{\partial z_i} = \hat{y}_i - y_K$ (10.1)

↳ Predicción para la clase i

Para el score de la salida esperada (clase correcta):

$\frac{\partial L}{\partial z_K} = \hat{y}_K - y_K$ (10.2)

Para inicializar W y b en una NN con pocas capas se puede:

→ $W_{n\times m} = np.random.randn(n,m) * 0.01$ (11.1)

$n \times m$

→ $b_{n\times 1} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} = np.zeros((n,1))$ (11.2)

→ $z = W @ x + b$ (11.3)

$\frac{\partial L}{\partial z}$ en forma matricial: ← (10.2)

$$\frac{\partial L}{\partial z_{n\times 1}} = \hat{y}_{n\times 1} - \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix}_{n\times 1} \quad (11.4)$$

← K elemento ↑ etiqueta del ejemplo

one hot vector = $y_{n\times 1}$

→ $\frac{\partial L}{\partial W} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial W} = \frac{\partial L}{\partial z} X$

$$\frac{\partial L}{\partial W_{n\times m}} = \frac{\partial L}{\partial z_{n\times 1}} \cdot (X_{m\times 1})^T \quad (11.5)$$

$$\frac{\partial L}{\partial W_{n\times m}} = (\hat{y}_{n\times 1} - y_{n\times 1}) \cdot (X_{m\times 1})^T \quad (11.6)$$

$$= (y\_hat - y) @ X.T$$

→ $\frac{\partial L}{\partial b} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial b} = \frac{\partial L}{\partial z}(1)$

$$\frac{\partial L}{\partial b_{n\times 1}} = \frac{\partial L}{\partial z_{n\times 1}} \cdot (1)_{1\times 1} = \frac{\partial L}{\partial z_{n\times 1}}$$

$$= \hat{y}_{n\times 1} - y_{n\times 1} \quad (11.7)$$

Después se puede actualizar W y b: (6.3) ↓ (11.8)

→ $W_{n\times m} = W_{n\times m} - \alpha \frac{\partial L}{\partial W_{n\times m}}$

↳ Bjm: 0.01

→ $b_{n\times 1} = b_{n\times 1} - \alpha \frac{\partial L}{\partial b_{n\times 1}}$ (11.9) ↑ (6.4)

For a batch with r examples:

$W_{n\times m} X_{m\times r} + b_{n\times r} = z_{n\times r}$

Loss functions of r examples:

$$L_{1\times r} = Lr = -\ln\left[\frac{(e^{z_{ki}})_{1\times r}}{\left(\sum_n^n e^{z_{ji}}\right)_{1\times r}}\right]$$

De la salida esperada para el ejemplo i

Using Jacobian matriz:

$$\frac{\partial L_r}{\partial z_n} = \begin{pmatrix} \frac{\partial L_1}{\partial z_1^3} & \cdots & \frac{\partial L_1}{\partial z_n^3} \\ \vdots & \ddots & \vdots \\ \frac{\partial L_r}{\partial z_1^3} & \cdots & \frac{\partial L_r}{\partial z_n^3} \end{pmatrix} = \frac{\partial L}{\partial z_{r\times n}}$$

For row i (example i) of r examples:

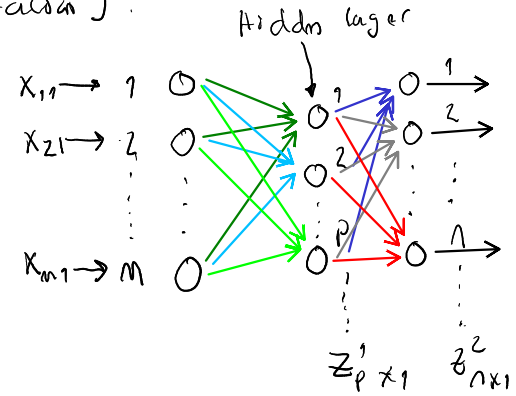
$$\frac{\partial L}{\partial z_{1\times n}} = \hat{y}_{1\times n} - y_{1\times n}$$

→ $\frac{\partial L}{\partial z_{r\times n}} = \hat{y}_{r\times n} - y_{r\times n}$

↓ one hot vectors:

$\begin{pmatrix} 0 & 1 & \cdots & 0 \\ 1 & 0 & \cdots & 0 \\ & & \vdots & \\ 0 & 0 & \cdots & 1 \end{pmatrix}$

Deep learning con dos capas y solo funciones lineales de activación (sin funciones de activación):



$$\rightarrow z^1_{p\times 1} = W^1_{p\times m} x_{m\times 1} + b^1_{p\times 1}$$

$$\rightarrow z^2_{n\times 1} = W^2_{n\times p} z_{p\times 1} + b^2_{n\times 1}$$

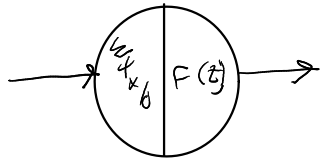
$$= W^2_{n\times p}\left(W^1_{p\times m} x_{m\times 1} + b^1_{p\times 1}\right) + b^2_{n\times 1}$$

$$= \left(W^2 W^1\right)_{n\times m} x_{m\times 1} + \left(W^2 b^1\right)_{n\times 1} + b^2_{n\times 1}$$

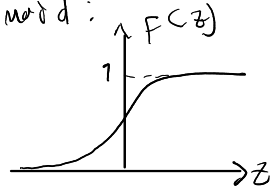
$$= \left(W^2 W^1\right)_{n\times m} x_{m\times 1} + \left(W^2 b^1 + b^2\right)_{n\times 1} \quad (12.1)$$

Entonces las hidden layer no actúan, dando igual usar la sola capa de salida con n perceptrones

Funciones de activación:

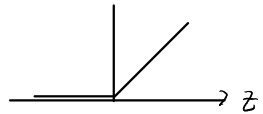


Sigmoid:



$$\sigma(z) = \frac{1}{1+e^{-z}} \quad (12.2)$$

ReLU: Rectified Linear Unit:



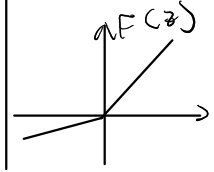
$z < 0 \rightarrow f(z) = 0$

$z \geq 0 \rightarrow f(z) = z$

$\max(0, z)$ (12.3)

máximo entre 0 y z

Leaky ReLU: (sim)

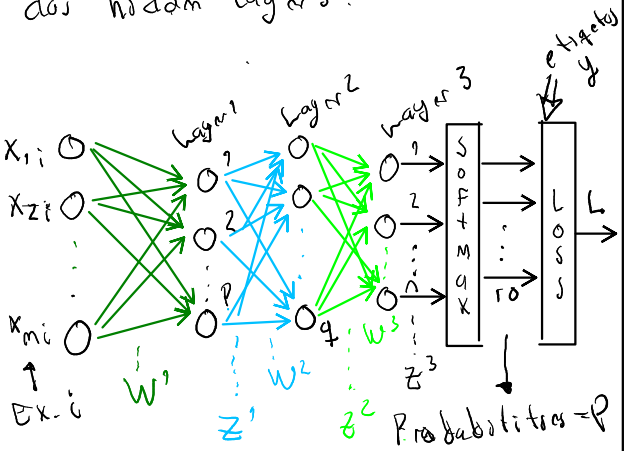


$z < 0 \rightarrow f(z) = -0.01\, z$

$z \geq 0 \rightarrow f(z) = z$

$\max(0.01\, z, z)$ (12.4)

Usando Funciones de activación con dos hidden layers:



R: Ejemplos totales

Usando un batch de r ejemplos:

$\frac{R}{r}$ = pasadas para entrenar con los R elementos

→ Forwardpass, Backpropagation

→ $X_{n\times r}$ ← Pixels, ↳ Batch

→ $z^1_{p\times r}$ → $a^1_{p\times r}$

→ $z^2_{q\times r}$ → $a^2_{q\times r}$

→ $z^3_{n\times r}$

→ $P_{n\times r}$

→ $L_{1\times r}$ → cost ↳ escalar.

→ $W^1_{p\times n}$ → $b^1_{p\times 1}$

→ $W^2_{q\times p}$ → $b^2_{q\times 1}$

→ $W^3_{n\times q}$ → $b^3_{n\times 1}$

↳ Training

Forwardpass:

Broadcast

$$z^1_{p\times r} = W^1_{p\times n} X_{n\times r} + (b^1_{p\times 1} \cdots b^1_{p\times 1})_{p\times r}$$

Después de aplicar la función de activación de la capa 1:

$$a^1_{p\times r} = F^1(z^1_{p\times r})$$

$$z^2_{q\times r} = W^2_{q\times p} a^1_{p\times r} + \underbrace{(b^2_{q\times 1} \cdots b^2_{q\times 1})_{q\times r}}_{b^2_{q\times r}}$$

$$a^2_{q\times r} = F^2(z^2_{q\times r})$$

$$z^3_{n\times r} = W^3_{n\times q} a^2_{q\times r} + b^3_{n\times r}$$

(11.4) ↑ Broadcast

Softmax:

$$P_{n\times r} = \frac{(e^{z^3})_{n\times r}}{\left(\sum_i e^{z^3_{ji}}\right)_{q\times r}} = \hat{y}_{n\times r}$$

(4.7)

suma cada columna (eje n)

Cross entropy:

$$L_{1\times r} = -\ln\left(P_{a_1}, P_{b_2} \cdots P_{z_r}\right)_{1\times r}$$

etiqueta ej. 1, etiqueta ej. 2

$$\text{cost} = \frac{\sum_i^r L_{1i}}{r}$$

(5.3)

Backpropagation:

$$\frac{\partial L}{\partial z^3_{n\times r}} = P_{n\times r} - y_{n\times r}$$

$y_{n\times r}$: one hot vector para cada ejemplo:

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 1 \end{pmatrix} \begin{matrix} \leftarrow 1 \\ \leftarrow 2 \\ \vdots \\ \leftarrow n \end{matrix}$$

1 2 ⋯ r

etiquetas

$$\frac{\partial L}{\partial a^2} = \frac{\partial L}{\partial z^3}\frac{\partial z^3}{\partial a^2}$$

$$\frac{\partial L}{\partial a^2_{q\times r}} = \left(W^3_{n\times q}\right)^T \frac{\partial L}{\partial z^3_{n\times r}}$$

Must be normalized

$$\frac{\partial L}{\partial W^3} = \frac{\partial L}{\partial z^3}\frac{\partial z^3}{\partial W^3}$$

$$\frac{\partial L}{\partial W^3_{n\times q}} = \frac{\partial L}{\partial z^3_{n\times r}} \frac{\left(a^2_{q\times r}\right)^T}{r}$$

normalized

$$\rightarrow \frac{\partial L}{\partial b^3} = \frac{\partial L}{\partial z^3} \cancel{\frac{\partial z^3}{\partial b^3}}^{I}$$

$$\frac{\partial L}{\partial b^3_{n\times 1}} = \begin{pmatrix} \sum_j^r \frac{\partial L}{\partial z^3_{1j}} \\ \vdots \\ \sum_j^r \frac{\partial L}{\partial z^3_{nj}} \end{pmatrix}_{n\times 1} \quad \text{sum elements of each row}$$

$/r$ normalized

W^3 normalized

$$\rightarrow \frac{\partial L}{\partial z^2} = \frac{\partial L}{\partial a^2} \frac{\partial a^2}{\partial z^2}$$

$$\frac{\partial L}{\partial z^2_{q\times r}} = \left(\frac{\partial L}{\partial a^2_{ij}} \cdot \frac{\partial F^2(z^2_{ij})}{\partial z^2} \right)_{q\times r}$$

$$= \frac{\partial F^2(z^2)}{\partial z^2} = g'(z^2)$$

$$g'(z^2_{ij})$$

$$\rightarrow \frac{\partial L}{\partial a^1} = \frac{\partial L}{\partial z^2} \frac{\partial z^2}{\partial a^1}$$

$$\frac{\partial L}{\partial a^1_{p\times r}} = (W^2_{q\times p})^T \frac{\partial L}{\partial z^2_{q\times r}}$$

must be normalized

$$\rightarrow \frac{\partial L}{\partial W^2} = \frac{\partial L}{\partial z^2} \frac{\partial z^2}{\partial W^2}$$

$$\frac{\partial L}{\partial W^2_{q\times p}} = \frac{\partial L}{\partial z^2_{q\times r}} (a^1_{p\times r})^T$$

$$\frac{\partial L}{\partial z^2} \leftarrow \frac{\partial L}{\partial a^2}$$

W^3 must be normalized

$$\rightarrow \frac{\partial L}{\partial b^2} = \frac{\partial L}{\partial z^2} \cancel{\frac{\partial z^2}{\partial b^2}}^{I}$$

$$\frac{\partial L}{\partial b^2_{q\times 1}} = \begin{pmatrix} \sum_j^r \frac{\partial L}{\partial z^2_{1j}} \\ \vdots \\ \sum_j^r \frac{\partial L}{\partial z^2_{qj}} \end{pmatrix}_{q\times 1} \quad /\text{dit}$$

$$\rightarrow \frac{\partial L}{\partial z^1} = \frac{\partial L}{\partial a^1} \frac{\partial a^1}{\partial z^1}$$

W^2 normalized

$$\frac{\partial L}{\partial z^1_{p\times r}} = \left(\frac{\partial L}{\partial a^1_{ij}} \frac{\partial F^1(z^1_{ij})}{\partial z^1} \right)_{p\times r}$$

W^2 normalized

$$\rightarrow \frac{\partial L}{\partial W^1} = \frac{\partial L}{\partial z^1} \frac{\partial z^1}{\partial W^1}$$

$$\frac{\partial L}{\partial W^1_{p\times m}} = \frac{\partial L}{\partial z^1_{p\times r}} (x_{m\times r})^T$$

$$\rightarrow \frac{\partial L}{\partial b^1} = \frac{\partial L}{\partial z^1} \cancel{\frac{\partial z^1}{\partial b^1}}^{I}$$

$$\frac{\partial L}{\partial b^1_{p\times 1}} = \begin{pmatrix} \sum_j^r \frac{\partial L}{\partial z^1_{1j}} \\ \vdots \\ \sum_j^r \frac{\partial L}{\partial z^1_{pj}} \end{pmatrix}_{p\times 1}$$

Training for each epoch, for each batch:

$$\rightarrow W^1_{p\times m} = W^1_{p\times m} - \alpha \frac{dL}{dW^1_{p\times m}}$$

$$\rightarrow b^1_{p\times 1} = b^1_{p\times 1} - \alpha \frac{dL}{db^1_{p\times 1}}$$

$$\rightarrow W^2_{q\times p} = W^2_{q\times p} - \alpha \frac{dL}{dW^2_{q\times p}}$$

$$\rightarrow b^2_{q\times 1} = b^2_{q\times 1} - \alpha \frac{dL}{db^2_{q\times 1}}$$

$\rightarrow W^3_{n\times q} = W^3_{n\times q} - \alpha \frac{dL}{dW^3_{n\times q}}$

$\rightarrow b^3_{n\times 1} = b^3_{n\times 1} - \alpha \frac{dL}{db^3_{n\times 1}}$

Accuracy using validation data: after each epoch

Forward (Validation Data)

$\rightarrow$ Softmax ($Z^3_{n\times r}$)

$\rightarrow P_{n\times r}$:

Prediction = $Pe = (P_{max,1} \cdots P_{max,r})_{1\times r}$

max row of col 1 ; max row col r

Accuracy = $\frac{\sum_{i=1}^{\#batches} \sum_{j=1}^{r} \delta_{Pe_j = Label_j}}{\#\text{Validation data}}$ (Validation data)

$\rightarrow$ %

$\rightarrow = \#batches \times r$

Prediction of a image after

epochs: $\rightarrow$ Optimal

$\rightarrow W^1_{p\times m} \rightarrow b^1_{p\times 1}$

$\rightarrow W^2_{q\times p} \rightarrow b^2_{q\times 1}$

$\rightarrow W^3_{n\times q} \rightarrow b^3_{n\times 1}$

$\rightarrow image_{1\times m} = X_{1\times m}$ $\rightarrow$ Total pixels

$\downarrow$

Forward ($X_{1\times m}$)

$\downarrow$

$Z^3_{n\times 1}$

$\downarrow$ softmax

$\hat{y}_{n\times 1} = P_{n\times 1}$

$\downarrow$

Prediction = Pe = P_{max} of $P_{n\times 1}$

Usando Matriz Jacobiana:

$\frac{\partial L_r}{\partial z^3_n}$ → Loss function de r ejemplos

$$\rightarrow \frac{\partial L_r}{\partial z^3_n} = \begin{pmatrix} \frac{\partial L_1}{\partial z^3_1} & \cdots & \frac{\partial L_1}{\partial z^3_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial L_r}{\partial z^3_1} & \cdots & \frac{\partial L_r}{\partial z^3_n} \end{pmatrix} = \frac{\partial L}{\partial z_{r\times n}}$$

$$\frac{\partial L}{\partial z^3_{r\times n}} = \hat{y}_{r\times n} - y_{r\times n}$$

↓ one hot vectors:

$$\begin{pmatrix} 0 & 1 & \cdots & 0 \\ 1 & 0 & \cdots & 0 \\ \vdots & & & \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

$$\rightarrow \frac{\partial L_r}{\partial a^2_q} = \frac{\partial L_r}{\partial z^3_n} \frac{\partial z^3_n}{\partial a^2_q}$$

$$\frac{\partial L}{\partial a^2_{r\times q}} = \frac{\partial L}{\partial z^3_{r\times n}} \frac{\partial z^3}{\partial a^2_{n\times q}}$$

$$= (\hat{y}_{r\times n} - y_{r\times n})\, W^3_{n\times q}$$

$$\rightarrow \frac{\partial L_r}{\partial W^3_{n\times q}} = \frac{\partial L_r}{\partial z^3_n} \frac{\partial z^3_n}{\partial W^3_{n\times q}}$$

$$\frac{\partial L}{\partial W^3_{r\times n\times q}} = (\hat{y}_{r\times n} - y_{r\times n})\, a^2_q$$

↳ Producto tensorial?

$$\rightarrow \frac{\partial L_r}{\partial b^3_n} = \frac{\partial L_r}{\partial z^3_n} \frac{\partial z^3_n}{\partial b^3_n}$$

$$\frac{\partial L}{\partial b^3_{r\times n}} = \frac{\partial L}{\partial z^3_{r\times n}} \quad I_{n\times n} = \frac{\partial L}{\partial z^3_{r\times n}}$$

$$\rightarrow \frac{\partial L_r}{\partial W^2_{q\times p}} = \frac{\partial L_r}{\partial z^3_n} \frac{\partial z^3_n}{\partial a^2_q} \frac{\partial a^2_q}{\partial z^2_q} \frac{\partial z^2_q}{\partial W^2_{q\times p}}$$

$$= (\hat{y}_{r\times n} - y_{r\times n})\, W^3_{n\times q} \left(\frac{\partial F^2(z^2)}{\partial z^2}\right)_{q\times q} a^2_p$$

$r\times q$

$$= \frac{\partial L}{\partial W^2_{r\times q\times p}} \rightarrow \frac{\partial L_r}{\partial z^2_q}$$

$$\rightarrow \frac{\partial L_r}{\partial b^2_q} = \frac{\partial L_r}{\partial z^3_n} \frac{\partial z^3_n}{\partial a^2_q} \frac{\partial a^2_q}{\partial z^2_q} \frac{\partial z^2_q}{\partial b^2_q}$$

↳ $I_{q\times q}$

$$= \frac{\partial L}{\partial b^2_{r\times q}}$$

$$\rightarrow \frac{\partial L_r}{\partial W^1_{p\times m}} = \frac{\partial L_r}{\partial z^3_n} \frac{\partial z^3_n}{\partial a^2_q} \frac{\partial a^2_q}{\partial z^2_q} \frac{\partial z^2_q}{\partial a^1_p} \frac{\partial a^1_p}{\partial z^1_p} \frac{\partial z^1_p}{\partial W^1_{p\times m}}$$

$$\frac{\partial L}{\partial W^1_{r\times p\times m}} = \frac{\partial L_r}{\partial z^2_q} W^2_{q\times p} \left(\frac{\partial F^1(z^1)}{\partial z^1}\right)_{p\times p} X_{m\times p}$$

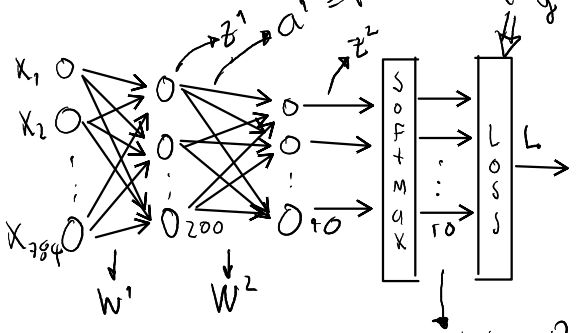
$$\frac{\partial L_r}{\partial z^1_p}$$

$$\rightarrow \frac{\partial L_r}{\partial b^1_p} = \frac{\partial L_r}{\partial z^1_p} \frac{\partial z^1_p}{\partial b^1_p}$$

↳ $I_{p\times p}$

$$= \frac{\partial L}{\partial b^1_{r\times p}}$$

Ex: MNIST:



→ $X_{64\times784}$ ← Pixels
↑ Batch

→ $z^1_{200\times64}$

→ $a^1_{200\times64}$

→ $z^2_{10\times64}$

→ $P_{10\times64}$

→ $L_{1\times64}$ → cost ← escalar.

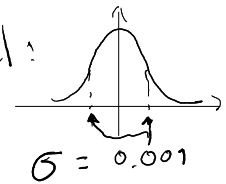
Pixels
→ $W^1_{200\times784}$
→ $b^1_{200\times1}$
→ $W^2_{10\times200}$
→ $b^2_{10\times1}$
← Training

Inicialmente:

$W^1_{200\times784} =$

np.random.randn(200,784) · 0.001

Distribuição normal:



→ $b^1_{200\times1} =$ np.zeros((200,1))

→ $W^2_{10\times200} =$

np.random.randn(10,200) · 0.001

→ $b^2_{10\times1} =$ np.zeros((10,1))

Forward:

$$z^1_{200\times64} = W^1_{200\times784}\,(X_{64\times784})^T + (b^1\ b^1\ \dots\ b^1)_{200\times64}$$

(1 … 64)

→ $a^1_{200\times64} = ReLU(z^1_{200\times64})$

→ $z^2_{10\times64} = W^2_{10\times200}\,a^1_{200\times64} + (b^2\ b^2\ \dots\ b^2)_{10\times64}$

Softmax:

$$P_{10\times64} = \frac{(e^{z^2})_{10\times64}}{\left(\sum_{ij}^{10} e^{z^2_{ji}}\right)_{1\times64}} = \hat{y}_{10\times64} \quad (C4.1)$$

soma cada coluna de $(e^{z^2})_{10\times64}$

Cross entropy:

(C5.2)

$$(e^L)_{1\times64} = (P_{a_1}\ P_{b_2}\ \dots\ P_{z64})$$

etiqueta $e_{j=1}$ … etiqueta $e_{j=2}$

→ $L_{1\times64} = -Ln(e^L)_{1\times64}$

→ $cost = \frac{\sum_{i}^{64} L_{1i}}{64}$ (C5.3)

Backward:

$$\frac{\partial L}{\partial z^2}_{10\times64} = \hat{y}_{10\times64} - y_{10\times64}$$

etiquetas ← one hot vector por cada ejemplo:

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & 1 \end{pmatrix} \begin{matrix} \leftarrow 1 \\ \leftarrow 2 \\ \vdots \\ \leftarrow 10 \end{matrix}$$

1 2 … 64

→ $\frac{\partial L}{\partial W^2} = \frac{\partial L}{\partial z^2}\frac{\partial z^2}{\partial W^2}$

$$\frac{\partial L}{\partial W^2_{10\times200}} = \frac{\frac{\partial L}{\partial z^2_{10\times64}}}{64}\,(a^1_{200\times64})^T$$

→ $\frac{\partial L}{\partial b^2} = \frac{\partial L}{\partial z^2}\frac{\partial z^2}{\partial b^2}$ (∂z²/∂b² = I)

$$\frac{\partial L}{\partial b^2_{10\times1}} = \begin{pmatrix} \sum_{j}^{64} \frac{\partial L}{\partial z^2_{1j}} \\ \vdots \\ \sum_{j}^{64} \frac{\partial L}{\partial z^2_{10j}} \end{pmatrix}_{10\times1} / 64$$

→ soma elementos de cada row

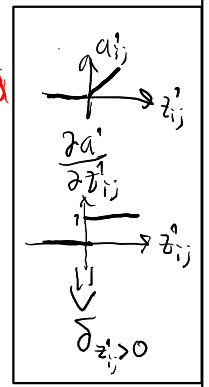
$$\frac{\partial L}{\partial a^1} = \frac{\partial L}{\partial z^2}\frac{\partial z^2}{\partial a^1}$$

most to be normalized

$$\frac{\partial L}{\partial a^1}_{200\times 64} = (W^2_{10\times 200})^T \frac{\partial L}{\partial z^2}_{10\times 64}$$

$$\frac{\partial L}{\partial z^1} = \frac{\partial L}{\partial a^1}\frac{\partial a^1}{\partial z^1}$$

normalized



$$\frac{\partial L}{\partial z^1}_{200\times 64} = \left(\frac{\partial L}{\partial a^1_{ij}}\,\delta_{z^1_{ij}>0}\right)_{200\times 64}$$

element ij of $\left(\frac{\partial L}{\partial a^1}\right)_{200\times 64}$

When z^1_{ij} element > 0: $\delta_{z^1_{ij}>0} = 1$ (of $z^1_{200\times 64}$)

When $z^1_{ij} \leq 0$: $\delta_{z^1_{ij}>0} = 0$

$$\frac{\partial L}{\partial W^1} = \frac{\partial L}{\partial z^1}\frac{\partial z^1}{\partial W^1}$$

$$\frac{\partial L}{\partial W^1}_{200\times 784} = \frac{\partial L}{\partial z^1}_{200\times 64} X_{64\times 784}$$

most to be normalized

$W^2 = \frac{\partial z^2}{\partial a^1}$

$\frac{\partial L}{\partial z^1}$

$\frac{\partial L}{\partial a^1}$

$$\frac{\partial L}{\partial b^1} = \frac{\partial L}{\partial z^1}\frac{\partial z^1}{\partial b^1}$$

$$\frac{\partial L}{\partial b^1}_{200\times 1} = \left(\sum_j^{64} \frac{\partial L}{\partial z^1_{1j}}, \ldots, \sum_j^{64} \frac{\partial L}{\partial z^1_{200j}}\right)_{200\times 1} / 64$$

Training for each epoch, for each batch:

$\alpha = 0.001$

$$W^1_{200\times 784} = W^1_{200\times 784} - \alpha \frac{dL}{dW^1_{200\times 784}}$$

$$b^1_{200\times 1} = b^1_{200\times 1} - \alpha \frac{dL}{db^1_{200\times 1}}$$

$$W^2_{10\times 200} = W^2_{10\times 200} - \alpha \frac{dL}{dW^2_{10\times 200}}$$

$$b^2_{10\times 1} = b^2_{10\times 1} - \alpha \frac{dL}{db^2_{10\times 1}}$$

Accuracy using validation data: after each epoch

Forward (Validation Data)

$\rightarrow$ Softmax $(Z^2_{10\times 64})$

$\rightarrow P_{10\times 64}$:

Predictions $= Pe = (P_{max,1}, \ldots P_{max,64})_{1\times 64}$

max row of col 1, ... max row of col. 64

$$\text{Accuracy} = \frac{\sum_{i=1}^{\#\text{batches}} \sum_{j=1}^{64} \delta_{Pe_j = \text{Label}_j}}{\#\text{Validation data}}$$

Validation data

Prediction of a image after # epochs: Optimal

- $W^1_{200\times 784}$
- $b^1_{200\times 1}$
- $W^2_{10\times 200}$
- $b^2_{10\times 1}$

$\rightarrow$ image$_{1\times 784} = X_{1\times 784}$

$$\rightarrow Z^1_{200\times 1} = W^1_{200\times 784}(X_{1\times 784})^T + b^1_{200\times 1}$$

$$\rightarrow a^1_{200\times 1} = ReLU(Z^1_{200\times 1})$$

$$\rightarrow Z^2_{10\times 1} = W^2_{10\times 200} a^1_{200\times 1} + b^2_{10\times 1}$$

$$\rightarrow P_{10\times 1} = \frac{(e^{Z^2})_{10\times 1}}{\sum_{j_1}^{10} e^{Z^2_{j_1}}} = \hat{y}_{10\times 1}$$

Prediction $= Pe = P_{max}$ of $P_{10\times 1}$

Min. value of pixels in MNIST:

min = 0 ← Black

Max value: (17.2)

max = 255 ← White

Normalize pixels to mean=0, std=1:

of every pixels / of every samples } mean, std (p → numbers, s)

of each pixel } data

$$\rightarrow Norm_{sxp} = \frac{data_{sxp} - mean}{std} \quad (17.3)$$

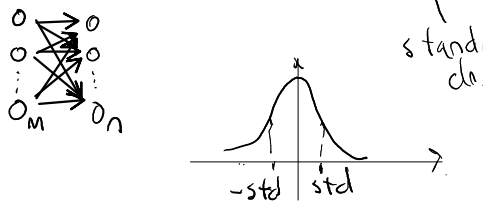
→ Norm.mean() = 0

→ Norm.std() = 1 (17.4)

Matrix nxm w rnd #s by normal distribution, mean 0: (17.1)

$$W_{nxm} = randn(n, m) * std$$

(std ← standar desviation)



Xavier He init: (17.5)

$$W_{nxm} = randn(n, m)\left(\frac{1}{\sqrt{m}}\right)$$

Kaiming init: (17.6)

$$W_{nxm} = randn(n, m)\left(\frac{1}{\sqrt{m/2}}\right)$$

Vanishing gradient problem:

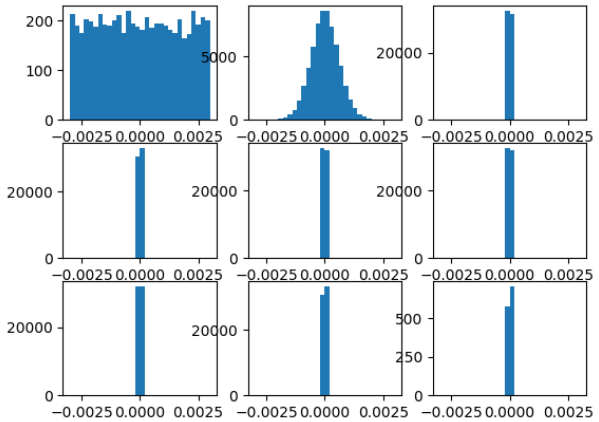
Forward:

$$z^1 = W^1 x + b^1$$
$$z^2 = W^2 f(z^1) + b^2$$
$$\vdots$$
$$z^i = W^i f(z^{i-1}) + b^i$$

For normal init: (17.7)

$$z^2 < z^1$$
$$\vdots$$
$$z^i < z^{i-1}$$

Histograms to $z^1, \dots, z^i$:



So, Backward:

$$\rightarrow \frac{\partial L}{\partial W^i} = \frac{\partial L}{\partial z^i}\frac{\partial z^i}{\partial W^i}$$

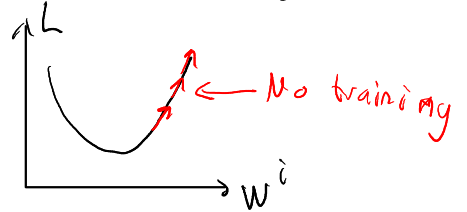
(fix; $f(z^{i-1}) \approx 0$) (17.8)

$$\rightarrow \frac{\partial L}{\partial W^{i-1}} = \frac{\partial L}{\partial z^{i-1}}\frac{\partial z^{i-1}}{\partial W^{i-1}}$$

(fix; $f(z^{i-2}) \approx 0$)

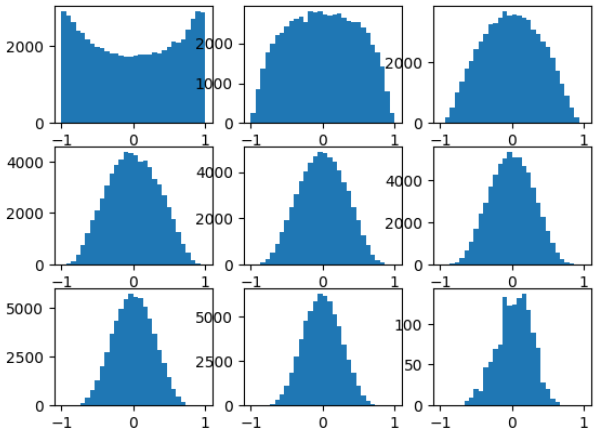
Training:

$$\rightarrow W^i = W^i - \alpha \frac{\partial L}{\partial W^i} \quad (17.9)$$



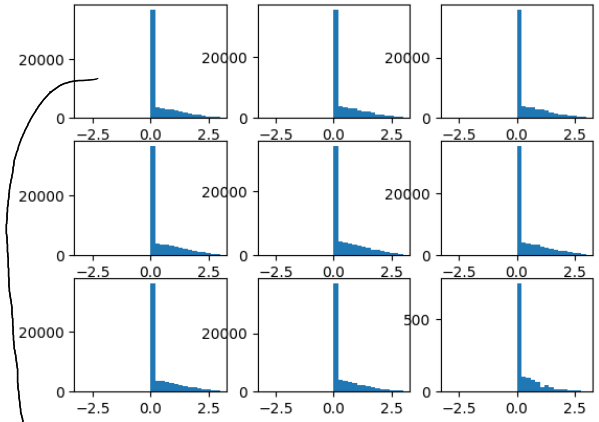
Ex. of No Vanishing problem using Xavier, f = tanh and MNIST ↓ ✗.

Histogram of $z^1, \ldots, z^9$:



Ex. of No Vanishing problem using Kaiming, f = ReLU, MNIST:

Histogram of $z^1, \ldots, z^9$:



std = 0.8127

std = 0.594

No Vanishing

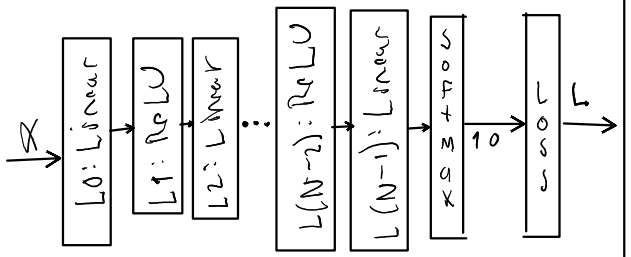
Tips to avoid Vanishing:

Normal → None $f(z)$

Xavier → $\tanh(z)$ (17.10)

Kaiming He → ReLU(z)
↳ $\tanh(z)$

Ex. MNIST using POO and sequential layer model:
N layers



Linear: Identity activation function

$\rightarrow X_{b \times 784}$ ← Pixels
↑ Batch

$\rightarrow z^0_{o_0 \times b}$ ← Linear
$\rightarrow a^1_{o_1 \times b}$ ← ReLU
$\rightarrow z^2_{o_2 \times b}$ ← Linear
$\rightarrow a^3_{o_3 \times b}$ ← ReLU
⋮
$\rightarrow z^{N-1}_{10 \times b}$ ← Linear
$\rightarrow P_{10 \times b}$
$\rightarrow L_{1 \times b}$ → cost
escalar

$\rightarrow W^0_{o_0 \times 784}$
$\rightarrow b^0_{o_0 \times 1}$
$\rightarrow W^2_{o_2 \times i_2}$
$\rightarrow b^2_{o_2 \times 1}$
⋮
$\rightarrow W^{N-1}_{10 \times i_{N-1}}$
$\rightarrow b^{N-1}_{10 \times 1}$

↳ Training of W & b of Linear Layers

Full connected:

$\rightarrow i_2 = o_0$
↑ inputs of L2 ↖ outputs of L0

ReLU layers:

$\rightarrow i_1 = o_0$ $\quad \rightarrow i_1 = o_1$
$\rightarrow i_2 = o_2$ $\quad \rightarrow i_3 = o_3$
⋮ ⋮
$\rightarrow i_{N-2} = o_{N-3}$ $\quad \rightarrow i_{N-1} = o_{N-1}$

Normalization of training, validation and testing images:

→ Mean of pixels of training images = mean
→ Standart desviation of training images = std

$\rightarrow X_{784} = \frac{X_{784} - \text{Mean}}{\text{std}}$ ← (17.3)

Initialization of W linear layers:

Kaiming He: ← (17.6)

$W^L_{o_L \times i_L} = \text{randn}(o_L, i_L) \left(\frac{1}{\sqrt{i_L / 2}}\right)$

Initialization of b of linear layers:

$b^L_{o_L \times 1} = \text{zeros}((o_L, 1))$

Forward of linear layers:

$\rightarrow z^0_{o_0 \times b} = W^0_{o_0 \times 784} (X_{b \times 784})^T + (b^0 b^0 \cdots)_{o_0 \times b}$

⋮

$\rightarrow z^L_{o_L \times b} = W^L_{o_L \times i_L} \; a^{L-1}_{i_L \times b} + (b^L b^L \cdots)_{o_L \times b}$

$i_{L-1} = o_{L-1} = i_L$

L is even

Forward of ReLU layer:

$a^L_{o_L \times b} = \text{ReLU}(z^{L-1}_{o_{L-1} \times b})$

$o_{L-1} = o_L = i_L$

L is odd

Softmax and cross entropy use the last layer L^{N-1}:

$\rightarrow z^{N-1}_{10 \times b}$

For backward:

$\rightarrow z^{N-1}.\text{grad} = \frac{\partial L}{\partial z^{N-1}}$

$\rightarrow W^{N-1}.\text{grad} = \frac{\partial L}{\partial W^{N-1}}$

$= \frac{\partial L}{\partial z^{N-1}} \frac{\partial z^{N-1}}{\partial W^{N-1}}$

$= \frac{\partial L}{\partial z^{N-1}} (a^{N-2})^T$

$\rightarrow b^{N-1}.\text{grad} = \frac{\partial L}{\partial b^{N-1}}$

⋮

α must be small during learning ← (6.3)