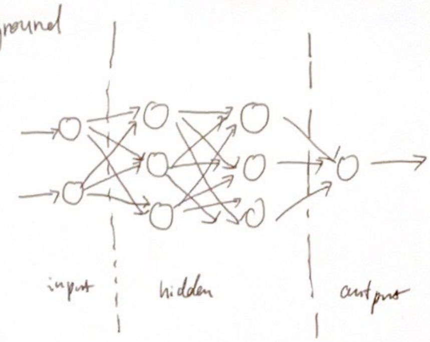
Background



$f_i()$: the i^{th} fully connected layer

o : output

x : input   ~~$x \in \mathbb{R}^{(B,I)}$~~

$o = f_3(f_2(f_1(x)))$

$O = a(W_1X_1 + W_2X_2 + \dots + W_nX_n + b)$

b: bias

a: activation function, a non linear function

W_i: the i^{th} weight of the linear regression function

$O^{(i)} = a(W^{(i)}x^{(i)} + b^{(i)})$

$$o = a^{(3)}(W^{(3)}a^{2}(W^{(2)}a^{(1)}(W^{(1)}x^{(1)} + b^{(1)}) + b^{(2)}) + b^{(3)})$$

Stochastic gradient descent (SGD)

L : loss function

z : the output of the linear regression

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial a} \cdot \frac{\partial a}{\partial z} \cdot \frac{\partial z}{\partial w}$$

or

$$\frac{\partial L}{\partial \theta} = \frac{\partial L}{\partial a} \cdot \frac{\partial a}{\partial z} \cdot \frac{\partial z}{\partial \theta}$$

to calculate the gradient of the weights

chain rule: $\frac{d}{dx} f(g(h(x)))$ ~~$f(g(x)) g'(h(x)) \cdot h'(x)$~~

$$f'(g(h(x))) \cdot g'(h(x)) \cdot h'(x)$$

to calculate the gradient of the activation function

additive rule of derivation

$$\frac{d}{dx}(f(x) + g(x) + h(x)) = f'(x) + g'(x) + h'(x)$$

$X \in \mathbb{R}^{(B,I)}$ $W \in \mathbb{R}^{(I,O)}$

output $\in \mathbb{R}^{(B,O)}$

linear class

__init__

forward

backward

update_parameters

$Y = xW + b$

update_parameters : • update the weights and biases of the network.

$$W_{ij} \leftarrow W_{ij} - \eta \frac{\partial L}{\partial W_{ij}}$$

$$\theta_{ij} \leftarrow \theta_{ij} - \eta \frac{\partial L}{\partial \theta_i}$$

* η : the learning rate

forward : • computes the regression agaist the inputs.

$$f = W_{1j}x_1 + W_{2j}x_2 + \dots + W_{nj}x_n + \theta_j$$

n : input size

W_{ij} : the weight for input entry i in regression unit j

θ_j : the bias term for regression unit j.

backford : given the gradient of the loss function with respect to the outputs

- computes the gradients of weights and biases.
- saves them as internal variable

$$\frac{\partial L}{\partial W_{ij}} = \frac{\partial L}{\partial z} \cdot \frac{\partial z}{\partial W_{ij}} \quad O_i \cdot X_j \quad \text{or} \quad \sum O_{ik} \cdot X_{jk}$$

$$\frac{\partial L}{\partial \theta_i} = \frac{\partial L}{\partial z} \cdot \frac{\partial z}{\partial \theta_i} = O_i \text{ , or } \sum O_{ik}$$

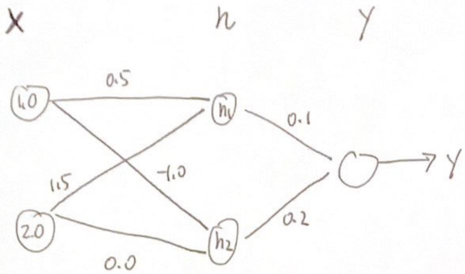
$$\frac{\partial L}{\partial X_j} = \frac{\partial L}{\partial z} \cdot \frac{\partial z}{\partial W_{ij}} = O_i \cdot W_j \quad \text{or} \quad \sum_k O_{ik} \cdot W_{jk}$$

* i : index of the unit

j : index of the weight

k : index of the complete input

An Easy Example about how linear class works, especially the forward function



$X = [1.0 \ , \ 2.0]$

$W = \begin{bmatrix} 0.5 \ , & -1.0 \\ 1.5 \ , & 0.0 \end{bmatrix}$

$b = \cancel{\begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix}} \quad [0.1 \quad 0.2]$

$Y = X @ W + b$

$Y = [1.0 \quad 2.0] \cdot \begin{bmatrix} 0.5 & -1.0 \\ 1.5 & 0.0 \end{bmatrix} + \cancel{\begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix}} \ [0.1 \ \ 0.2] = [3.6 \quad -0.8]$

$$\left.\begin{array}{l} X \in \mathbb{R}^{1\times 2} \\ W \in \mathbb{R}^{2\times 2} \\ b \in \mathbb{R}^{2\times 1} \\ Y \in \mathbb{R}^{1\times 2} \end{array}\right\} \xrightarrow{\text{Generalize It}} \begin{array}{l} X \in \mathbb{R}^{1\times n} \\ W \in \mathbb{R}^{n\times 2} \\ b \in \mathbb{R}^{n\times 1} \\ Y \in \mathbb{R}^{1\times 2} \end{array}$$

backwards

in forward propogation

$z = x \cdot W + b$

The Gradient ~~of~~ weights

$$\frac{\partial L}{\partial W} = \left(\frac{\partial z}{\partial W}\right)^T \cdot \frac{\partial L}{\partial z}$$

$$z = XW \Rightarrow \frac{\partial z}{\partial W} = X$$

$$\Rightarrow \frac{\partial L}{\partial W} = X^T \cdot \frac{\partial L}{\partial z}$$

e.g $X = [1.0 \quad 2.0]$

$$X^T = \begin{bmatrix} 1.0 \\ 2.0 \end{bmatrix}$$

grad_output = $[1.0 \quad 2.0]$

grad_W = X^T @ grad_output

$$= \begin{bmatrix} 1.0 \\ 2.0 \end{bmatrix} \times \begin{bmatrix} 1.0 & 2.0 \end{bmatrix} = \begin{bmatrix} 1.0 & 2.0 \\ 2.0 & 4.0 \end{bmatrix}$$

L : loss function

W : weights Matrix

X : current input, can be batch

$\frac{\partial L}{\partial z}$: the gradient from last layer, here called as grad_output

X^T : the transpose of X

The gradient of bias

$$\text{grad biases} = \frac{\partial L}{\partial \theta_i} = \sum_k \cancel{d_{ik}}\, O_{ik}$$

O_{ik} : grad_output[k,i]

in human words: the gradient of the loss with respect to the output at dimension i for the k-th input sample.

$\frac{\partial L}{\partial \theta_i}$: The gradient of each bias θ_i is the sum of the loss gradients at i-th output dimension across all training samples.

e.g. grad_output = np.array([
[0.1 -0.2]
[0.3 -0.1]
[0.2 0.0]
])

np.sum(grad_output, axis=0, keepdims=True)

= [0.1+0.3+0.2, -0.2+(-0.1)+0.0]] = [[0.6, 0.3]]

aus aspect of matrix:

$$\text{grad\_output}: \begin{bmatrix} 0.1 & -0.2 \\ 0.3 & -0.1 \\ 0.2 & 0.0 \end{bmatrix} \rightarrow \text{sum up column wise}$$

⑩ ~~Backword~~ Back propagation

The Gradient of output.

① because: $z = xW + b$

$\Rightarrow \quad \frac{\partial L}{\partial x} = \frac{\partial L}{\partial z} \cdot W^T$

$\frac{\partial L}{\partial z}$ means: How the loss L change if the output z changes?

it's the gradient, which flows into this layer from output.

it represents how the loss changs with the output of this layer, z

Therefore, $\frac{\partial L}{\partial z}$ ist gradient_output

in Python: grad_input = grad_output @ weight.T

an concret example:

$X = [1 \quad 2]$

$W = \begin{bmatrix} 0.5 & -1 \\ 1.5 & 0 \end{bmatrix}$

$b = [0.1 \quad 0.2]$

$z = X @ W + b = [3.6 \quad -0.8]$

grad_output = dL/dz = [1 2]

grad_input = grad_output @ W.T = [-1.5 1.5]