Intro

- In this video we will talk about MLP implementation details

Dense layer as a matrix multiplication

- We can compute 2 neurons with linear activation, 3 inputs and no bias term as a matrix multiplication:

$$(x_1 \quad x_2 \quad x_3) \cdot \begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \\ w_{3,1} & w_{3,2} \end{bmatrix} = (z_1 \quad z_2)$$

which is equivalent to:

$$z_1 = x_1 w_{1,1} + x_2 w_{2,1} + x_3 w_{3,1}$$
$$z_2 = x_1 w_{1,2} + x_2 w_{2,2} + x_3 w_{3,2}$$

and is written as: $xW = z$

- Matrix multiplication can be done faster on both CPU (e.g. **BLAS**) and GPU (e.g. **cuBLAS**).

- Matrix multiplication with **numpy** is much faster than **Python** loops.

Backward pass for a dense layer

- Forward pass:

$$xW = z \qquad\qquad (x_1 \quad x_2 \quad x_3) \cdot \begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \\ w_{3,1} & w_{3,2} \end{bmatrix} = (z_1 \quad z_2)$$

- For backward pass we need $\frac{\partial L}{\partial W}$, where $L(z_1, z_2)$ is our scalar loss.

$$\frac{\partial L}{\partial W} = \begin{bmatrix} \frac{\partial L}{\partial w_{1,1}} & \frac{\partial L}{\partial w_{1,2}} \\ \frac{\partial L}{\partial w_{2,1}} & \frac{\partial L}{\partial w_{2,2}} \\ \frac{\partial L}{\partial w_{3,1}} & \frac{\partial L}{\partial w_{3,2}} \end{bmatrix}$$

which is convenient for SGD:

$$W_{new} = W - \gamma \frac{\partial L}{\partial W}$$

Backward pass for a dense layer

- Forward pass:

$$xW = z \qquad \begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} \cdot \begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \\ w_{3,1} & w_{3,2} \end{bmatrix} = \begin{pmatrix} z_1 & z_2 \end{pmatrix}$$

- For backward pass we need $\frac{\partial L}{\partial W}$, where $L(z_1, z_2)$ is our scalar loss.

$$\frac{\partial L}{\partial W} = \begin{bmatrix} \frac{\partial L}{\partial w_{1,1}} & \frac{\partial L}{\partial w_{1,2}} \\ \frac{\partial L}{\partial w_{2,1}} & \frac{\partial L}{\partial w_{2,2}} \\ \frac{\partial L}{\partial w_{3,1}} & \frac{\partial L}{\partial w_{3,2}} \end{bmatrix}$$

let's apply a chain rule

$$\frac{\partial L}{\partial w_{i,j}} = \sum_k \frac{\partial L}{\partial z_k} \frac{\partial z_k}{\partial w_{i,j}} = \frac{\partial L}{\partial z_j} \frac{\partial z_j}{\partial w_{i,j}} = \frac{\partial L}{\partial z_j} x_i$$

$$z_j = x_1 w_{1,j} + x_2 w_{2,j} + x_3 w_{3,j}$$

Backward pass for a dense layer

- Forward pass:

$$xW = z \qquad\qquad (x_1 \quad x_2 \quad x_3) \cdot \begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \\ w_{3,1} & w_{3,2} \end{bmatrix} = (z_1 \quad z_2)$$

- For backward pass we need $\frac{\partial L}{\partial W}$, where $L(z_1, z_2)$ is our scalar loss.

$$\frac{\partial L}{\partial W} = \begin{bmatrix} \frac{\partial L}{\partial w_{1,1}} & \frac{\partial L}{\partial w_{1,2}} \\ \frac{\partial L}{\partial w_{2,1}} & \frac{\partial L}{\partial w_{2,2}} \\ \frac{\partial L}{\partial w_{3,1}} & \frac{\partial L}{\partial w_{3,2}} \end{bmatrix}$$

$$\frac{\partial L}{\partial w_{i,j}} = \sum_k \frac{\partial L}{\partial z_k} \frac{\partial z_k}{\partial w_{i,j}} = \frac{\partial L}{\partial z_j} \frac{\partial z_j}{\partial w_{i,j}} = \frac{\partial L}{\partial z_j} x_i$$

$$\frac{\partial L}{\partial z} = \left(\frac{\partial L}{\partial z_1} \quad \frac{\partial L}{\partial z_2} \right) \qquad \text{gradient vector}$$

$$\frac{\partial L}{\partial W} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \cdot \left(\frac{\partial L}{\partial z_1} \quad \frac{\partial L}{\partial z_2} \right) = x^T \frac{\partial L}{\partial z}$$

Forward pass for mini-batches

- Usually we do forward pass in mini-batches.

Batch of 2: $$\begin{pmatrix} x_{1,1} & x_{1,2} & x_{1,3} \\ x_{2,1} & x_{2,2} & x_{2,3} \end{pmatrix} \cdot \begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \\ w_{3,1} & w_{3,2} \end{bmatrix} = \begin{pmatrix} z_{1,1} & z_{1,2} \\ z_{2,1} & z_{2,2} \end{pmatrix}$$

Matrix notation: $$XW = Z$$

1st neuron for 2nd sample: $$z_{\mathbf{2},1} = x_{\mathbf{2},1} w_{1,1} + x_{\mathbf{2},2} w_{2,1} + x_{\mathbf{2},3} w_{3,1}$$

Backward pass for mini-batches

- SGD with mini-batches takes a sum of losses for each sample.

Batch of 2: $$\begin{pmatrix} x_{1,1} & x_{1,2} & x_{1,3} \\ x_{2,1} & x_{2,2} & x_{2,3} \end{pmatrix} \cdot \begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \\ w_{3,1} & w_{3,2} \end{bmatrix} = \begin{pmatrix} z_{1,1} & z_{1,2} \\ z_{2,1} & z_{2,2} \end{pmatrix}$$

SGD step: $$\frac{\partial L_b}{\partial W} = \frac{\partial L\left(z_{\mathbf{1},1}, z_{\mathbf{1},2}\right)}{\partial W} + \frac{\partial L\left(z_{\mathbf{2},1}, z_{\mathbf{2},2}\right)}{\partial W}$$

For one sample: $$\frac{\partial L}{\partial w_{i,j}} = \frac{\partial L}{\partial z_j} \frac{\partial z_j}{\partial w_{i,j}} = \frac{\partial L}{\partial z_j} x_i$$ *we know this*

For 2 samples: $$\frac{\partial L_b}{\partial w_{i,j}} = \frac{\partial L}{\partial z_{\mathbf{1},j}} x_{\mathbf{1},i} + \frac{\partial L}{\partial z_{\mathbf{2},j}} x_{\mathbf{2},i}$$

Backward pass for mini-batches

- SGD with mini-batches takes a sum of losses for each sample.

Batch of 2: $\begin{pmatrix} x_{1,1} & x_{1,2} & x_{1,3} \\ x_{2,1} & x_{2,2} & x_{2,3} \end{pmatrix} \cdot \begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \\ w_{3,1} & w_{3,2} \end{bmatrix} = \begin{pmatrix} z_{1,1} & z_{1,2} \\ z_{2,1} & z_{2,2} \end{pmatrix}$

SGD step: $\frac{\partial L_b}{\partial W} = \frac{\partial L\left(z_{\mathbf{1},1}, z_{\mathbf{1},2}\right)}{\partial W} + \frac{\partial L\left(z_{\mathbf{2},1}, z_{\mathbf{2},2}\right)}{\partial W}$

For 2 samples: $\frac{\partial L_b}{\partial w_{i,j}} = \frac{\partial L}{\partial z_{1,j}} x_{1,i} + \frac{\partial L}{\partial z_{2,j}} x_{2,i}$

$\frac{\partial L_b}{\partial W} = X^T \frac{\partial L}{\partial Z}$ | $X^T = \begin{pmatrix} x_{1,1} & x_{2,1} \\ x_{1,2} & x_{2,2} \\ x_{1,3} & x_{2,3} \end{pmatrix}$ | $\frac{\partial L}{\partial Z} = \begin{pmatrix} \frac{\partial L}{\partial z_{1,1}} & \frac{\partial L}{\partial z_{1,2}} \\ \frac{\partial L}{\partial z_{2,1}} & \frac{\partial L}{\partial z_{2,2}} \end{pmatrix}$

Backward pass for mini-batches

- SGD with mini-batches takes a sum of losses for each sample.

Batch of 2: $\begin{pmatrix} x_{1,1} & x_{1,2} & x_{1,3} \\ x_{2,1} & x_{2,2} & x_{2,3} \end{pmatrix} \cdot \begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \\ w_{3,1} & w_{3,2} \end{bmatrix} = \begin{pmatrix} z_{1,1} & z_{1,2} \\ z_{2,1} & z_{2,2} \end{pmatrix}$

SGD step: $\frac{\partial L_b}{\partial W} = \frac{\partial L(z_{\mathbf{1},1}, z_{\mathbf{1},2})}{\partial W} + \frac{\partial L(z_{\mathbf{2},1}, z_{\mathbf{2},2})}{\partial W}$

For 2 samples: $\frac{\partial L_b}{\partial \boldsymbol{w}_{\mathbf{3},\mathbf{2}}} = \frac{\partial L}{\partial z_{1,2}} x_{1,3} + \frac{\partial L}{\partial z_{2,2}} x_{2,3}$ *let's check*

$\frac{\partial L_b}{\partial W} = X^T \frac{\partial L}{\partial Z}$ | $X^T = \begin{pmatrix} x_{1,1} & x_{2,1} \\ x_{1,2} & x_{2,2} \\ \boldsymbol{x}_{\mathbf{1},\mathbf{3}} & \boldsymbol{x}_{\mathbf{2},\mathbf{3}} \end{pmatrix}$ | $\frac{\partial L}{\partial Z} = \begin{pmatrix} \frac{\partial L}{\partial z_{1,1}} & \boldsymbol{\frac{\partial L}{\partial z_{1,2}}} \\ \frac{\partial L}{\partial z_{2,1}} & \boldsymbol{\frac{\partial L}{\partial z_{2,2}}} \end{pmatrix}$

Backward pass for X (used in MLP)

- $\boldsymbol{X}$ could contain outputs of a previous hidden layer:

Batch of 2: $$\begin{pmatrix} x_{1,1} & x_{1,2} & x_{1,3} \\ x_{2,1} & x_{2,2} & x_{2,3} \end{pmatrix} \cdot \begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \\ w_{3,1} & w_{3,2} \end{bmatrix} = \begin{pmatrix} z_{1,1} & z_{1,2} \\ z_{2,1} & z_{2,2} \end{pmatrix}$$

SGD step: $$\frac{\partial L_b}{\partial X} = \frac{\partial L\left(z_{\mathbf{1},1}, z_{\mathbf{1},2}\right)}{\partial X} + \frac{\partial L\left(z_{\mathbf{2},1}, z_{\mathbf{2},2}\right)}{\partial X}$$

For 1 sample: *let's apply a chain rule*

$$\frac{\partial L\left(z_{\boldsymbol{i},1}, z_{\boldsymbol{i},2}\right)}{\partial x_{\boldsymbol{i},j}} = \sum_k \frac{\partial L}{\partial z_{i,k}} \frac{\partial z_{i,k}}{\partial x_{i,j}} = \sum_k \frac{\partial L}{\partial z_{i,k}} w_{j,k} = \sum_k \frac{\partial L}{\partial z_{i,k}} w_{k,j}^T$$

For 2 samples: $$\frac{\partial L_b}{\partial X} = \frac{\partial L}{\partial Z} W^T$$

contributes 1 non-zero row

Fast Python implementation

- Forward pass for a **dense layer** with **numpy**:

```python
def forward_pass(X, W):
    return X.dot(W)
```

$$XW = Z$$

- Backward pass with **numpy**:

```python
def backward_pass(X, W, dZ):
    dX = dZ.dot(W.T)
    dW = X.T.dot(dZ)
    return dX, dW
```

$$\frac{\partial L_b}{\partial X} = \frac{\partial L}{\partial Z} W^T$$

$$\frac{\partial L_b}{\partial W} = X^T \frac{\partial L}{\partial Z}$$

- One more reason to have $\frac{\partial L}{\partial Z}$ in backward pass **interface**:
  - Otherwise, we would need $\frac{\partial Z}{\partial X}$ and $\frac{\partial Z}{\partial W}$, which is scary!

Summary

- Forward pass for a **dense layer** is a matrix multiplication
- Backward pass is a matrix multiplication as well
- Efficient for mini-batches on both CPU and GPU
- Easy to code it with **numpy**
- In the next video we will take a quick look at other matrix derivatives