INF721

2024/2

UFV

Deep Learning

L6: Backpropagation

Logistics

Announcements

- PA1: Logistic Regression is due this monday (30/09)!
- We don't have class this monday. It's a holiday!

Last Lecture

- Non-linearly Separable Problems
- Multilayer Perceptron
  - Forward Pass
  - Hypothesis Space (Composite Functions)
- Categorical Cross-Entropy Loss

Lecture Outline

- Gradient Descent for Neural Networks
- Computational Graph
- Backpropagation
- Examples:
  - Logistic Regression
  - Multilayer Perceptron

Gradient Descent for Neural Networks

```python
def optimize(x, y, lr, n_iter):
  # Init weights with rand. vals. close to 0
  W_1, b_1, W_2, b_2 = init_weights_rand()

  for t in range(n_iter):
    # Predict x labels
    y_hat = forward(W_1, b_1, W_2, b_2)

    # Compute gradients
    dw_1 = ?, dw_2 = ?
    db_1 = ?, db_2 = ?

    # Update weights
    W_1 = W_1 - lr * dw_1
    b_1 = b_1 - lr * db_1
    ...

  return W_1, b_1, W_2, b_2
```

MLP (2 Layers)

$$\mathbf{z}^{[1]} = W^{[1]}\mathbf{x} + \mathbf{b}^{[1]}$$
$$\mathbf{a}^{[1]} = g^{[1]}(\mathbf{z}^{[1]})$$
$$z^{[2]} = W^{[2]}\mathbf{a}^{[1]} + b^{[2]}$$
$$\hat{y} = \sigma(z^{[2]})$$

BCE Loss Function (Binary Classification)

$$L(h) = -\frac{1}{n}\sum_{i=1}^{n}(y_i \; log \; \hat{y}_i + (1 - y_i) \; log \; (1 - \hat{y}_i))$$

Gradients

$$\frac{\partial L}{\partial W_1} = ? \qquad \frac{\partial L}{\partial W_2} = ?$$
$$\frac{\partial L}{\partial b_1} = ? \qquad \frac{\partial L}{\partial b_2} = ?$$

UFV

Computing the gradients of a Neural Network

Linear models are simple enough so we can compute gradients by hand:

- Linear Regression: $\frac{\partial L}{\partial w} = (\hat{y} - y)x$, $\frac{\partial L}{\partial b} = (\hat{y} - y)$
- Logistic Regression: $\frac{\partial L}{\partial w} = (\hat{y} - y)x$, $\frac{\partial L}{\partial b} = (\hat{y} - y)$

However, as the size of our models grows, this becames impractical:

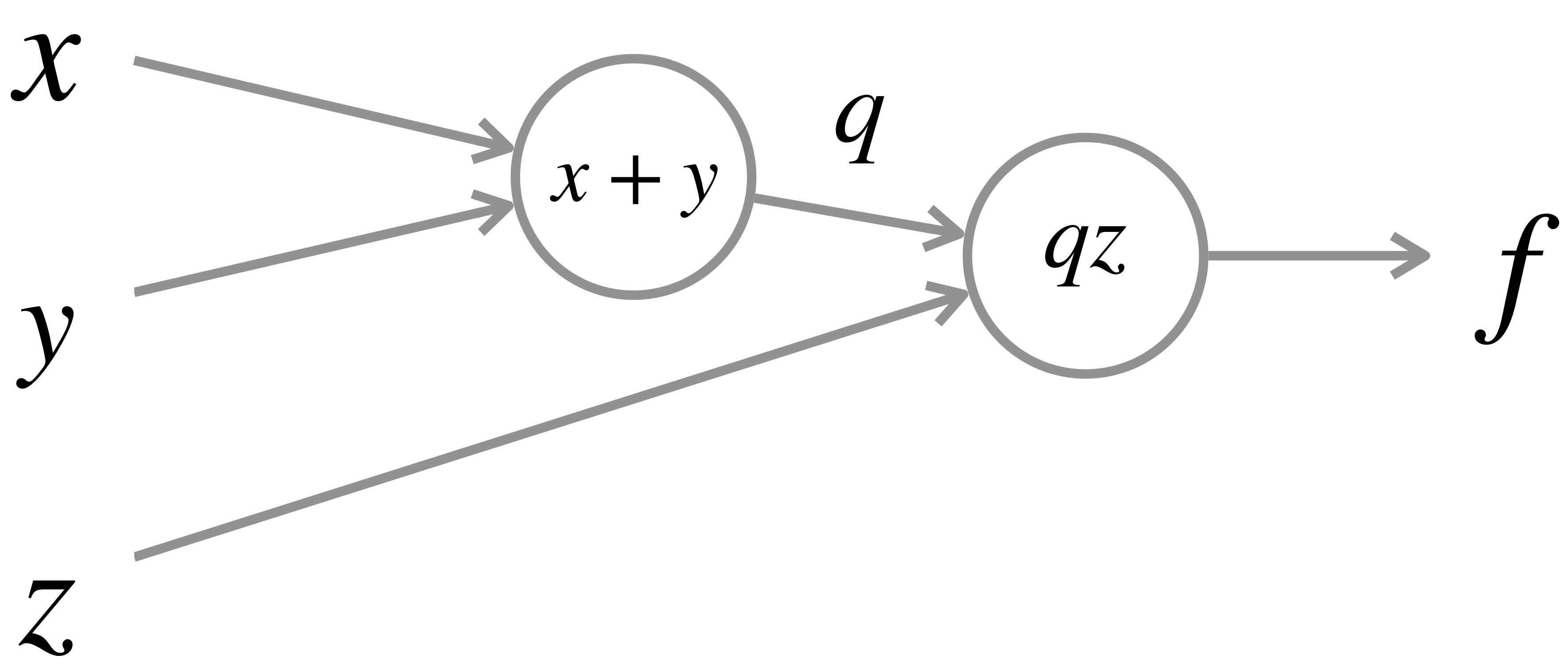
- It's easy to make mistakes
- It's not flexible – if we change the model or loss function, we have to recompute the gradients!
- **Solution: backpropagation!**

Computational Graph

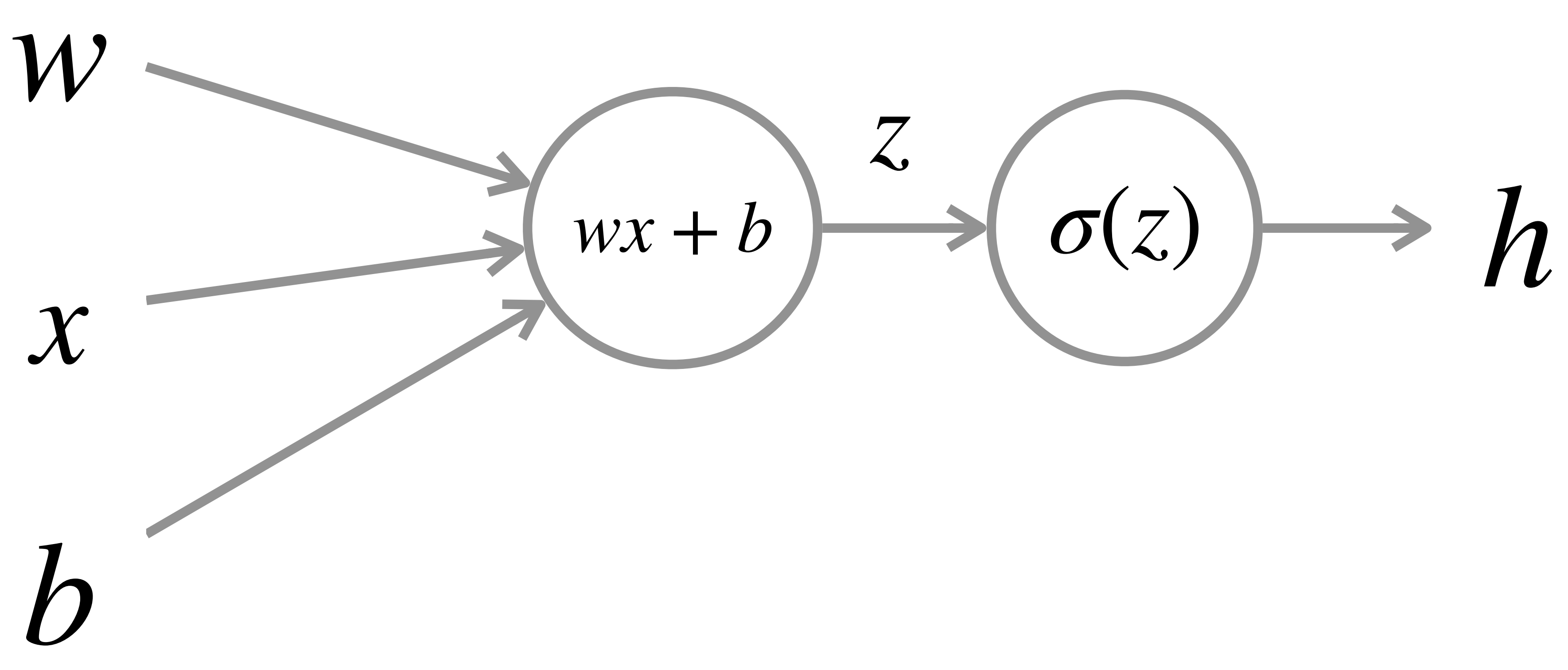
A **computional graph** is a directed graph that represents mathematical operations:

- A node is a function of its inputs
- An edge represents a function argument

$$f(x, y, z) = (x + y)z$$



$$h(w, x, b) = \sigma(wx + b)$$



UFV

Backpropagation

Chain rule

$$\frac{\mathrm{d}}{\mathrm{d}x} f(g(x)) = \frac{\mathrm{d}f}{\mathrm{d}g} \cdot \frac{\mathrm{d}g}{\mathrm{d}x}$$

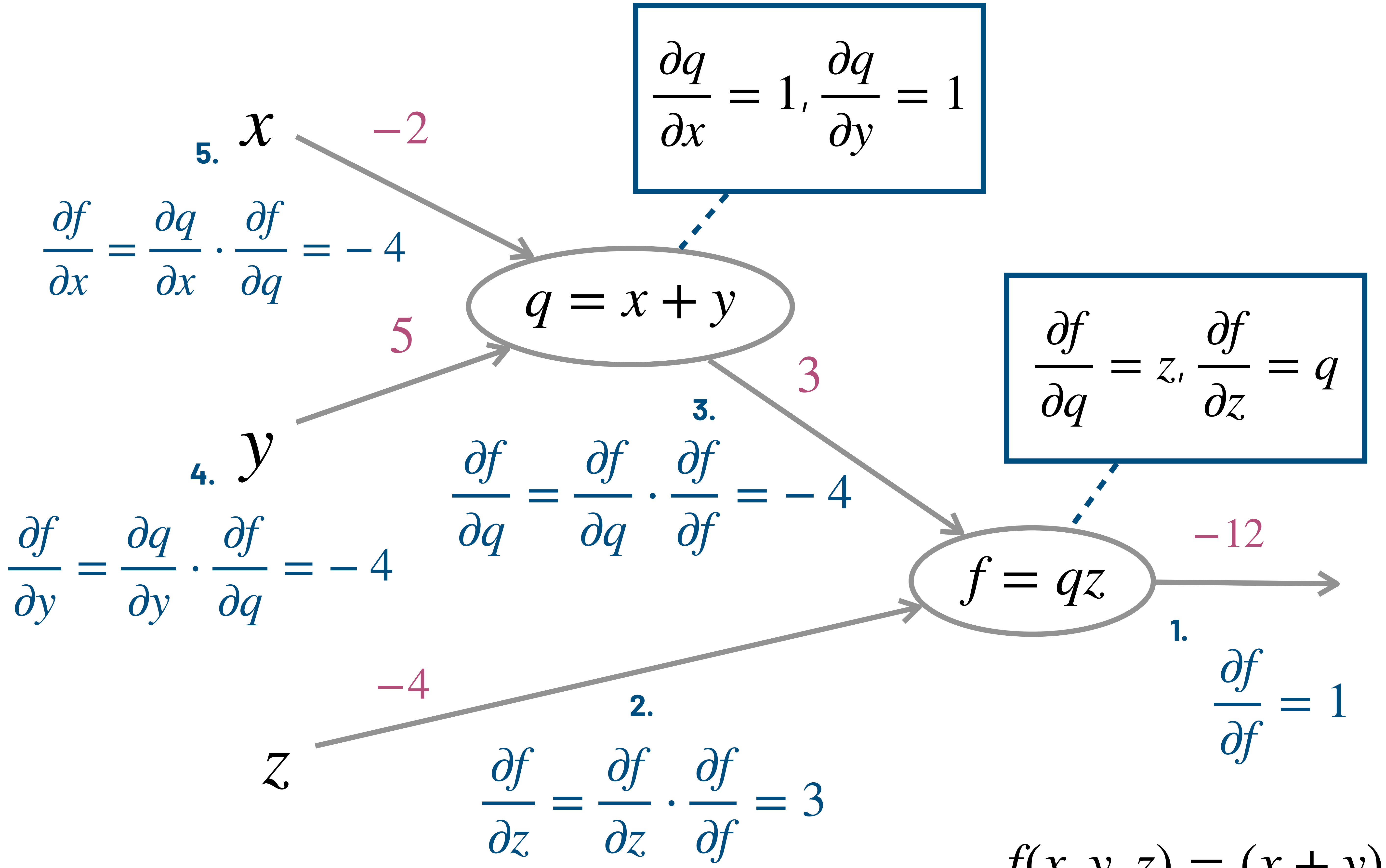
Backpropagation is an algorithm that uses a computational graph with the chain rule of calculus to compute the gradients of a given function f.

1. **Forward pass**
   Compute the outputs of f
   - Nodes store partial results

2. **Backward pass**
   Compute the derivativative of the output with respect to each input

$$\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$$

   - Node knows it's own derivatives
   - Apply the chain rule



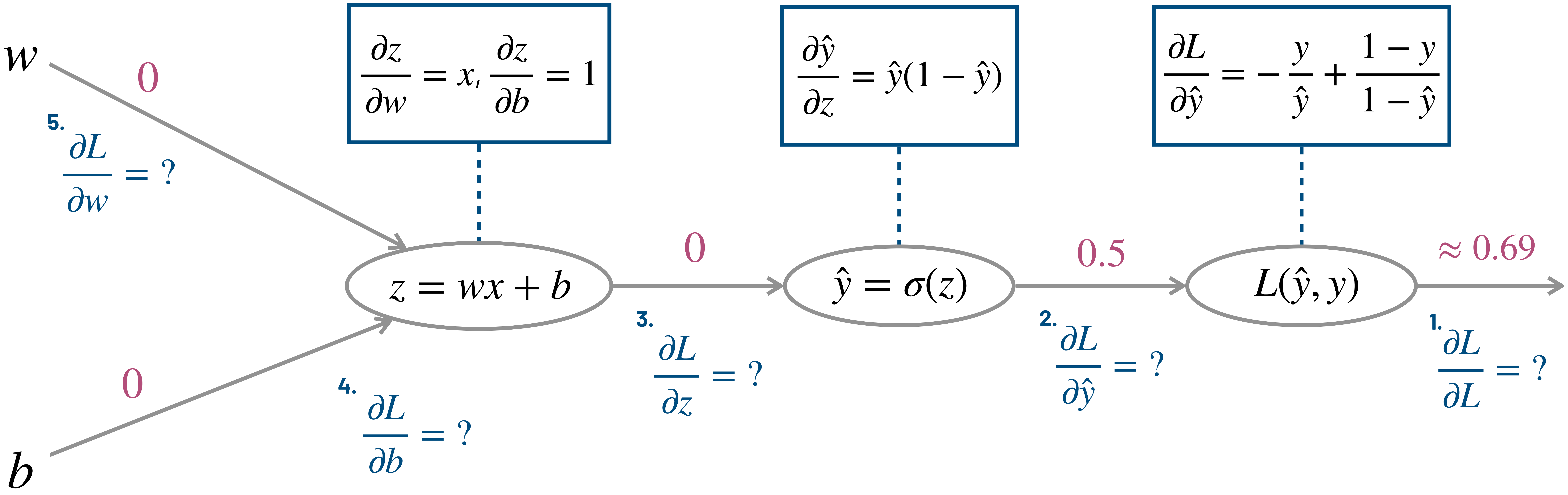
$$f(x, y, z) = (x + y)z$$

Backpropagation for Logistic Regression

We typicall use backpropagation to compute the gradients of a loss function with respect to weights of a neural network

Logistic Regression: $\hat{y} = h(x) = \dfrac{1}{1 + e^{-(wx+b)}}$

BCE Loss: $L(\hat{y}, y) = -\, y \; log \; \hat{y} + (1 - y) \; log \; (1 - \hat{y}))$



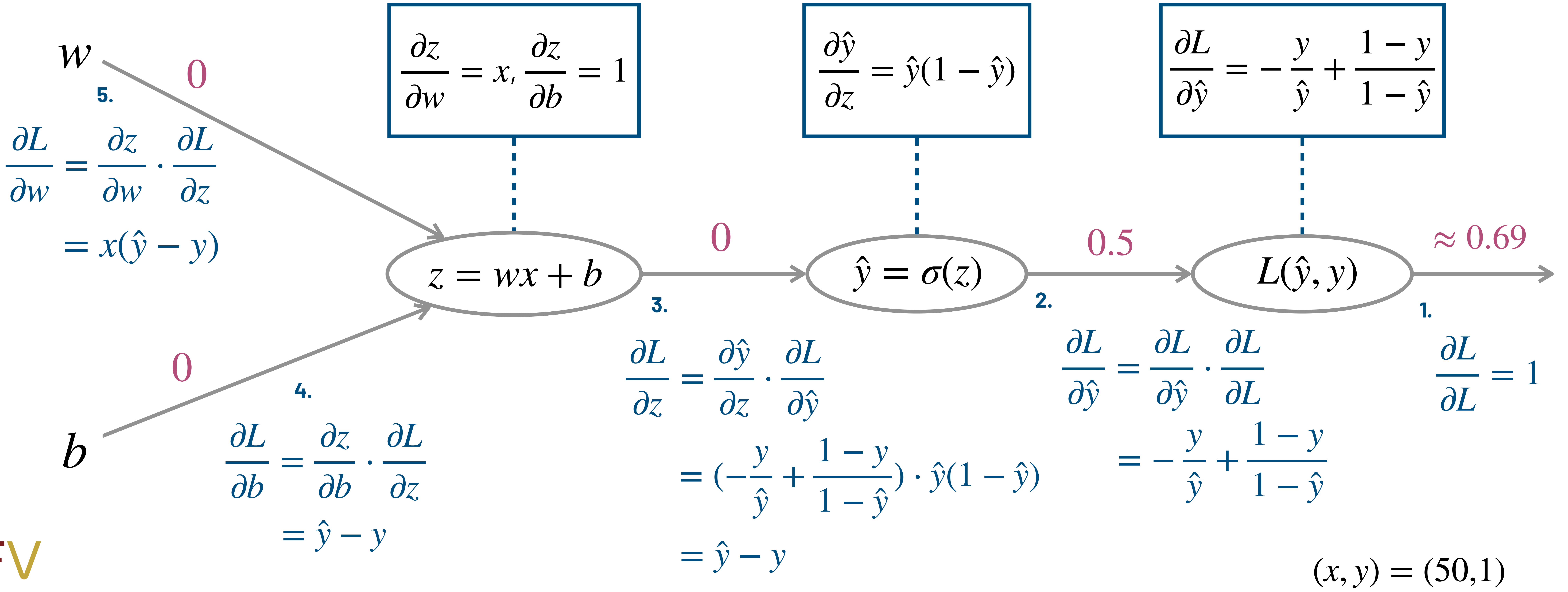
$(x, y) = (50{,}1)$

Backpropagation for Logistic Regression

We typicall use backpropagation to compute the gradients of a loss function with respect to weights of a neural network

Logistic Regression: $\hat{y} = h(x) = \dfrac{1}{1 + e^{-(wx+b)}}$

BCE Loss: $L(\hat{y}, y) = -\, y \; log \; \hat{y} + (1 - y) \; log \; (1 - \hat{y}))$

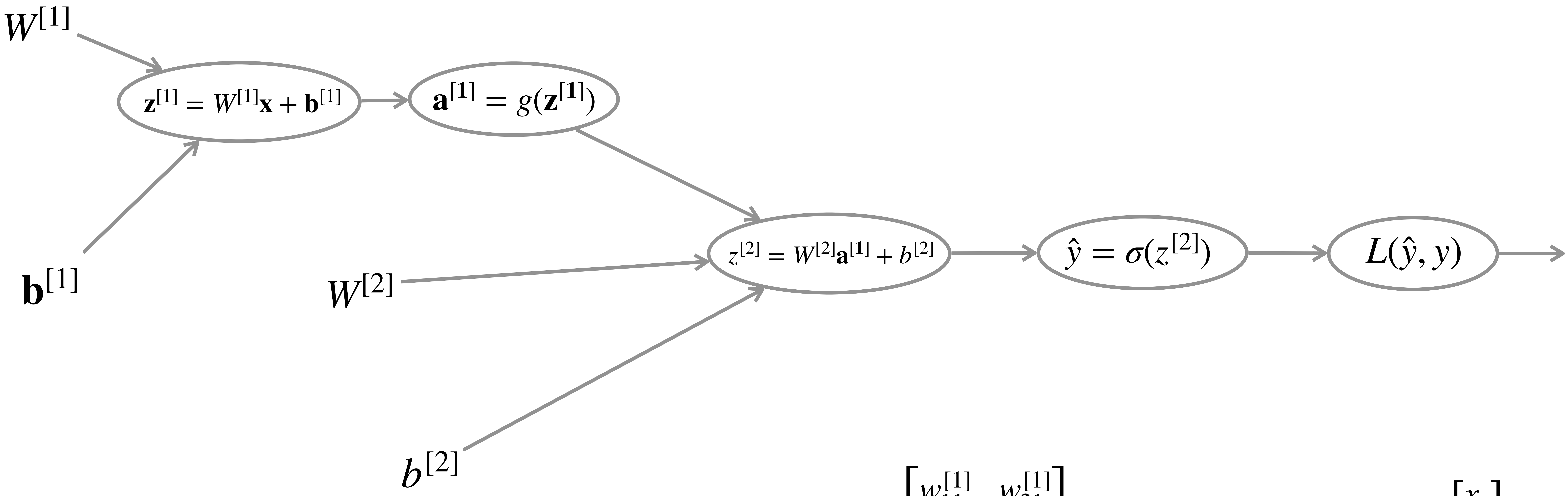


Backpropagation for 2-Layer MLP

MLP: $\mathbf{z}^{[1]} = W^{[1]}\mathbf{x} + \mathbf{b}^{[1]} \quad z^{[2]} = W^{[2]}\mathbf{a}^{[1]} + b^{[2]}$

$\mathbf{a}^{[1]} = g^{[1]}(\mathbf{z}^{[1]}) \quad \hat{y} = \sigma(z^{[2]})$

BCE Loss: $L(\hat{y}, y) = -y \; log \; \hat{y} + (1-y) \; log \; (1-\hat{y}))$



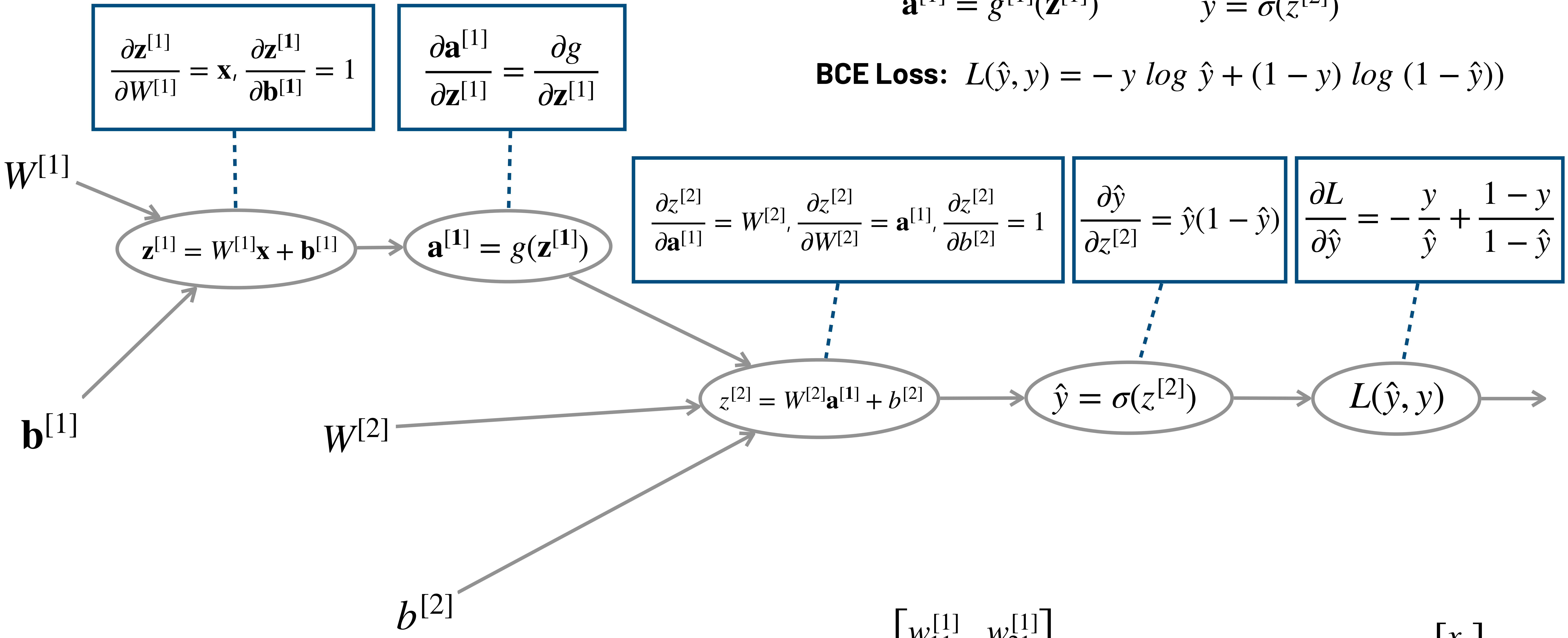
$$W^{[1]} = \begin{bmatrix} w_{11}^{[1]} & w_{21}^{[1]} \\ w_{12}^{[1]} & w_{22}^{[1]} \end{bmatrix} \quad W^{[2]} = \begin{bmatrix} w_{11}^{[2]} & w_{21}^{[2]} \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Backpropagation for 2-Layer MLP

MLP: $\mathbf{z}^{[1]} = W^{[1]}\mathbf{x} + \mathbf{b}^{[1]}$ $\quad z^{[2]} = W^{[2]}\mathbf{a}^{[1]} + b^{[2]}$

$\mathbf{a}^{[1]} = g^{[1]}(\mathbf{z}^{[1]})$ $\quad \hat{y} = \sigma(z^{[2]})$

BCE Loss: $L(\hat{y}, y) = -y \log \hat{y} + (1-y) \log (1-\hat{y}))$



$$W^{[1]} = \begin{bmatrix} w_{11}^{[1]} & w_{21}^{[1]} \\ w_{12}^{[1]} & w_{22}^{[1]} \end{bmatrix} \quad W^{[2]} = \begin{bmatrix} w_{11}^{[2]} & w_{21}^{[2]} \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

UFV

Backpropagation for 2-Layer MLP

MLP: $\mathbf{z}^{[1]} = W^{[1]}\mathbf{x} + \mathbf{b}^{[1]}$ $\quad z^{[2]} = W^{[2]}\mathbf{a}^{[1]} + b^{[2]}$

$\mathbf{a}^{[1]} = g^{[1]}(\mathbf{z}^{[1]})$ $\quad \hat{y} = \sigma(z^{[2]})$

BCE Loss: $L(\hat{y}, y) = -y \log \hat{y} + (1-y) \log (1-\hat{y}))$

$$W^{[1]} = \begin{bmatrix} w_{11}^{[1]} & w_{21}^{[1]} \\ w_{12}^{[1]} & w_{22}^{[1]} \end{bmatrix} \quad W^{[2]} = \begin{bmatrix} w_{11}^{[2]} & w_{21}^{[2]} \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

UFV

Backpropagation for 2-Layer MLP

MLP: $\mathbf{z}^{[1]} = W^{[1]}\mathbf{x} + \mathbf{b}^{[1]}$ $\quad z^{[2]} = W^{[2]}\mathbf{a}^{[1]} + b^{[2]}$

$\mathbf{a}^{[1]} = g^{[1]}(\mathbf{z}^{[1]})$ $\quad \hat{y} = \sigma(z^{[2]})$

BCE Loss: $L(\hat{y}, y) = -\, y \; log \; \hat{y} + (1 - y) \; log \; (1 - \hat{y}))$

$$\frac{\partial \mathbf{z}^{[1]}}{\partial W^{[1]}} = \mathbf{x}, \frac{\partial \mathbf{z}^{[\mathbf{1}]}}{\partial \mathbf{b}^{[1]}} = 1$$

$$\frac{\partial \mathbf{a}^{[1]}}{\partial \mathbf{z}^{[1]}} = \frac{\partial g}{\partial \mathbf{z}^{[1]}}$$

$$\frac{\partial z^{[2]}}{\partial \mathbf{a}^{[1]}} = W^{[2]}, \frac{\partial z^{[2]}}{\partial W^{[2]}} = \mathbf{a}^{[1]}, \frac{\partial z^{[2]}}{\partial b^{[2]}} = 1$$

$$\frac{\partial \hat{y}}{\partial z^{[2]}} = \hat{y}(1 - \hat{y})$$

$$\frac{\partial L}{\partial \hat{y}} = -\frac{y}{\hat{y}} + \frac{1 - y}{1 - \hat{y}}$$

$W^{[1]}$, $\mathbf{b}^{[1]}$ → $\mathbf{z}^{[1]} = W^{[1]}\mathbf{x} + \mathbf{b}^{[1]}$ → $\mathbf{a}^{[\mathbf{1}]} = g(\mathbf{z}^{[\mathbf{1}]})$ → $z^{[2]} = W^{[2]}\mathbf{a}^{[\mathbf{1}]} + b^{[2]}$ ← $W^{[2]}$, $b^{[2]}$

$z^{[2]} = W^{[2]}\mathbf{a}^{[\mathbf{1}]} + b^{[2]}$ → $\hat{y} = \sigma(z^{[2]})$ → $L(\hat{y}, y)$ →

1. $$\frac{\partial L}{\partial L} = 1$$

2. $$\frac{\partial L}{\partial \hat{y}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial L}{\partial L}$$

3. $$\frac{\partial L}{\partial z^{[2]}} = \frac{\partial \hat{y}}{\partial z^{[2]}} \cdot \frac{\partial L}{\partial \hat{y}}$$

4. $$\frac{\partial L}{\partial b^{[2]}} = \frac{\partial z^{[2]}}{\partial b^{[2]}} \cdot \frac{\partial L}{\partial z^{[2]}}$$

5. $$\frac{\partial L}{\partial W^{[2]}} = \frac{\partial z^{[2]}}{\partial W^{[2]}} \cdot \frac{\partial L}{\partial z^{[2]}}$$

6. $$\frac{\partial L}{\partial \mathbf{a}^{[1]}} = \frac{\partial z^{[2]}}{\partial \mathbf{a}^{[1]}} \cdot \frac{\partial L}{\partial z^{[2]}}$$

7. $$\frac{\partial L}{\partial Z^{[1]}} = \frac{\partial \mathbf{a}^{[1]}}{\partial \mathbf{z}^{[1]}} \cdot \frac{\partial L}{\partial a^{[1]}}$$

8. $$\frac{\partial L}{\partial b^{[1]}} = \frac{\partial \mathbf{z}^{[\mathbf{1}]}}{\partial \mathbf{b}^{[\mathbf{1}]}} \cdot \frac{\partial L}{\partial Z^{[1]}}$$

9. $$\frac{\partial L}{\partial W^{[1]}} = \frac{\partial \mathbf{z}^{[\mathbf{1}]}}{\partial W^{[1]}} \cdot \frac{\partial L}{\partial Z^{[1]}}$$

$$W^{[1]} = \begin{bmatrix} w_{11}^{[1]} & w_{21}^{[1]} \\ w_{12}^{[1]} & w_{22}^{[1]} \end{bmatrix} \quad W^{[2]} = \begin{bmatrix} w_{11}^{[2]} & w_{21}^{[2]} \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Backpropagation for 2-Layer MLP

MLP: $\mathbf{z}^{[1]} = W^{[1]}\mathbf{x} + \mathbf{b}^{[1]}$ $\quad z^{[2]} = W^{[2]}\mathbf{a}^{[1]} + b^{[2]}$

$\mathbf{a}^{[1]} = g^{[1]}(\mathbf{z}^{[1]})$ $\quad \hat{y} = \sigma(z^{[2]})$

BCE Loss: $L(\hat{y}, y) = -\,y\ log\ \hat{y} + (1-y)\ log\ (1-\hat{y}))$

$$\frac{\partial \mathbf{z}^{[1]}}{\partial W^{[1]}} = \mathbf{x}, \frac{\partial \mathbf{z}^{[1]}}{\partial \mathbf{b}^{[1]}} = 1$$

$$\frac{\partial \mathbf{a}^{[1]}}{\partial \mathbf{z}^{[1]}} = \frac{\partial g}{\partial \mathbf{z}^{[1]}}$$

$$\frac{\partial z^{[2]}}{\partial \mathbf{a}^{[1]}} = W^{[2]}, \frac{\partial z^{[2]}}{\partial W^{[2]}} = \mathbf{a}^{[1]}, \frac{\partial z^{[2]}}{\partial b^{[2]}} = 1$$

$$\frac{\partial \hat{y}}{\partial z^{[2]}} = \hat{y}(1-\hat{y})$$

$$\frac{\partial L}{\partial \hat{y}} = -\frac{y}{\hat{y}} + \frac{1-y}{1-\hat{y}}$$

$W^{[1]}$ $\xrightarrow{9.}$ $\mathbf{z}^{[1]} = W^{[1]}\mathbf{x} + \mathbf{b}^{[1]}$ $\xrightarrow{7.}$ $\mathbf{a}^{[1]} = g(\mathbf{z}^{[1]})$ $\xrightarrow{6.}$ $z^{[2]} = W^{[2]}\mathbf{a}^{[1]} + b^{[2]}$ $\xrightarrow{3.}$ $\hat{y} = \sigma(z^{[2]})$ $\xrightarrow{2.}$ $L(\hat{y}, y)$ $\xrightarrow{1.}$

$\mathbf{b}^{[1]}$, $W^{[2]}$, $b^{[2]}$

1. $$\frac{\partial L}{\partial L} = 1$$

2. $$\frac{\partial L}{\partial \hat{y}} = -\frac{y}{\hat{y}} + \frac{1-y}{1-\hat{y}}$$

3. $$\frac{\partial L}{\partial z^{[2]}} = \hat{y} - y$$

4. $$\frac{\partial L}{\partial b^{[2]}} = \hat{y} - y$$

5. $$\frac{\partial L}{\partial W^{[2]}} = \mathbf{a}^{[1]}(\hat{y} - y)$$

6. $$\frac{\partial L}{\partial \mathbf{a}^{[1]}} = W^{[2]}(\hat{y} - y)$$

7. $$\frac{\partial L}{\partial Z^{[1]}} = \frac{\partial \mathbf{g}}{\partial \mathbf{z}^{[1]}} \times W^{[2]}(\hat{y} - y)$$

8. $$\frac{\partial L}{\partial b^{[1]}} = \frac{\partial \mathbf{g}}{\partial \mathbf{z}^{[1]}} \times W^{[2]}(\hat{y} - y)$$

9. $$\frac{\partial L}{\partial W^{[1]}} = \mathbf{x}\frac{\partial \mathbf{g}}{\partial \mathbf{z}^{[1]}} \times W^{[2]}(\hat{y} - y)$$

$$W^{[1]} = \begin{bmatrix} w_{11}^{[1]} & w_{21}^{[1]} \\ w_{12}^{[1]} & w_{22}^{[1]} \end{bmatrix} \quad W^{[2]} = \begin{bmatrix} w_{11}^{[2]} & w_{21}^{[2]} \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

UFV

Gradient Descent for Neural Networks

```
def optimize(x, y, lr, n_iter):
  # Init weights with rand. vals. close to 0
  W_1, b_1, W_2, b_2 = init_weights_rand()

  for t in range(n_iter):
    # Predict x labels
    y_hat = forward(W_1, b_1, W_2, b_2)

    # Compute gradients
    dw_1, db_1 , dw_2, db_2 = backward()

    # Update weights
    W_1 = W_1 - lr * dw_1
    b_1 = b_1 - lr * db_1
    ...

  return W_1, b_1, W_2, b_2
```

MLP (2 Layers)

$$\mathbf{z}^{[1]} = W^{[1]}\mathbf{x} + \mathbf{b}^{[1]}$$
$$\mathbf{a}^{[1]} = g^{[1]}(\mathbf{z}^{[1]})$$
$$z^{[2]} = W^{[2]}\mathbf{a}^{[1]} + b^{[2]}$$
$$\hat{y} = \sigma(z^{[2]})$$

BCE Loss Function (Binary Classification)

$$L(h) = -\frac{1}{n}\sum_{i=1}^{n}(y_i \; log \; \hat{y}_i + (1 - y_i) \; log \; (1 - \hat{y}_i))$$

Gradients

$$\frac{\partial L}{\partial W_1} = \mathbf{x}\frac{\partial \mathbf{g}}{\partial \mathbf{z}^{[1]}} \times W^{[2]}(\hat{y} - y) \qquad \frac{\partial L}{\partial W_2} = \mathbf{a}^{[1]}(\hat{y} - y)$$

$$\frac{\partial L}{\partial b_1} = \frac{\partial \mathbf{g}}{\partial \mathbf{z}^{[1]}} \times W^{[2]}(\hat{y} - y) \qquad \frac{\partial L}{\partial b^{[2]}} = \hat{y} - y$$

UFV

Next Lecture

L7: Evaluating Deep Learning Models

Metrics for evaluating the generalization deep learning models

- Acuracy/Error
- Learning Curve
- Cross-validation
- Confusion Matrix