Week 5 Discussion 1A

Joe Lin and Krystof Latka
Learning Assistants

November 1, 2024

Midway Review

K-Nearest Neighbors

What is true about the K-Nearest Neighbor (KNN) classifier?

K-Nearest Neighbors

What is true about the K-Nearest Neighbor (KNN) classifier?

a. The L1 distance metric typically produces a smooth decision boundary, while the L2 distance metric is sharp and angled.
b. It needs to be trained for a long time, but testing is very fast.
c. We can tune for the most optimal value of k for a given dataset.
d. It is an unsupervised learning algorithm.

K-Nearest Neighbors

What is true about the K-Nearest Neighbor (KNN) classifier?

a. The L1 distance metric typically produces a smooth decision boundary, while the L2 distance metric is sharp and angled.
b. It needs to be trained for a long time, but testing is very fast.
c. We can tune for the most optimal value of k for a given dataset.
d. It is an unsupervised learning algorithm.

Regression

Which of the following statements is correct regarding the loss functions used in Linear, Logistic, and Softmax Regression tasks?

Regression

Which of the following statements is correct regarding the loss functions used in Linear, Logistic, and Softmax Regression tasks?

a. Linear Regression uses mean squared error (MSE), Logistic Regression uses cross entropy loss (CE), and Softmax Regression uses mean absolute error (MAE).
b. Linear Regression uses MSE, Logistic Regression uses hinge loss, and Softmax Regression uses CE.
c. Linear Regression uses MSE, Logistic Regression uses binary cross entropy loss (BCE), and Softmax Regression uses CE.
d. All three use CE.

Regression

Which of the following statements is correct regarding the loss functions used in Linear, Logistic, and Softmax Regression tasks?

a. Linear Regression uses mean squared error (MSE), Logistic Regression uses cross entropy loss (CE), and Softmax Regression uses mean absolute error (MAE).
b. Linear Regression uses MSE, Logistic Regression uses hinge loss, and Softmax Regression uses CE.
c. Linear Regression uses MSE, Logistic Regression uses binary cross entropy loss (BCE), and Softmax Regression uses CE.
d. All three use CE.

Gradient Descent

Problem: With naive **stochastic gradient descent**, optimization can get stuck at local minima.

Gradient Descent

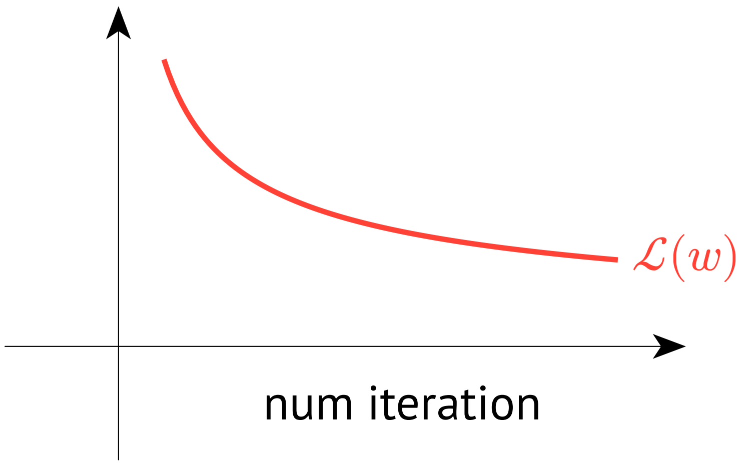
Problem: With naive **stochastic gradient descent**, optimization can get stuck at local minima.

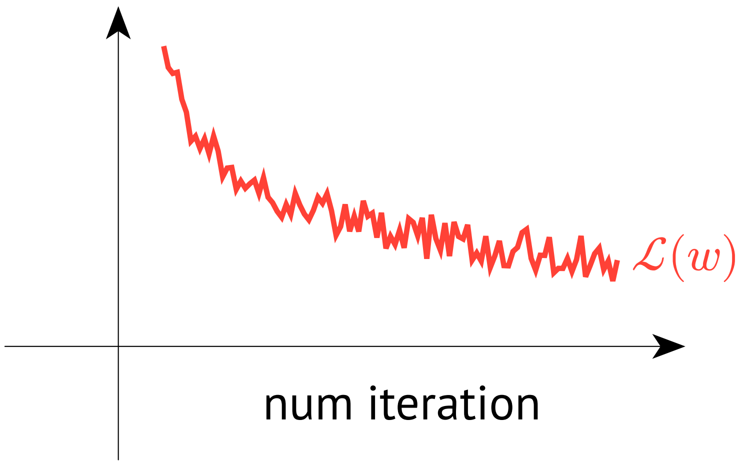
Solution: Use a running mean of gradients to build up **momentum** in a general direction.

$$v_{t+1} = \rho v_t + \nabla f(x_t)$$
$$x_{t+1} = x_t - \alpha v_{t+1}$$

Gradient Descent

Which of the following are true about the training loss curves below?





Gradient Descent

a. Graph A is **Mini-Batch Gradient Descent** – gradient estimates using batch and minibatch are equivalent.
b. Graph A is **Batch Gradient Descent** – accurate gradient estimates, resulting in a decrease in loss at every iteration.
c. Graph B is **Mini-Batch Gradient Descent** – noisy gradient estimates, resulting in oscillations in the loss trajectory.
d. Graph B is **Batch Gradient Descent** – considering more data points, so introduces more noise to gradient calculation.

Gradient Descent

a. Graph A is **Mini-Batch Gradient Descent** – gradient estimates using batch and minibatch are equivalent.
b. Graph A is **Batch Gradient Descent** – accurate gradient estimates, resulting in a decrease in loss at every iteration.
c. Graph B is **Mini-Batch Gradient Descent** – noisy gradient estimates, resulting in oscillations in the loss trajectory.
d. Graph B is **Batch Gradient Descent** – considering more data points, so introduces more noise to gradient calculation.

Regularization

Which of the following are valid techniques to perform regularization?

Regularization

Which of the following are valid techniques to perform regularization?

a. Data Augmentation
b. Norm Penalties
c. Model Ensembling
d. Dropout

Regularization

Which of the following are valid techniques to perform regularization?

a. Data Augmentation
b. Norm Penalties
c. Model Ensembling
d. Dropout

Hyperparameters

You are tasked with training a **Fully Connected Neural Network** with batch normalization and dropout. Which of the following are **hyperparameters**?

Hyperparameters

You are tasked with training a **Fully Connected Neural Network** with batch normalization and dropout. Which of the following are **hyperparameters**?

a. Loss Function $\mathcal{L}$
b. Dropout Probability p
c. Batch Normalization's γ and β
d. Batch Size B

Hyperparameters

You are tasked with training a **Fully Connected Neural Network** with batch normalization and dropout. Which of the following are **hyperparameters**?

a. Loss Function $\mathcal{L}$
b. Dropout Probability p
c. Batch Normalization's γ and β
d. Batch Size B

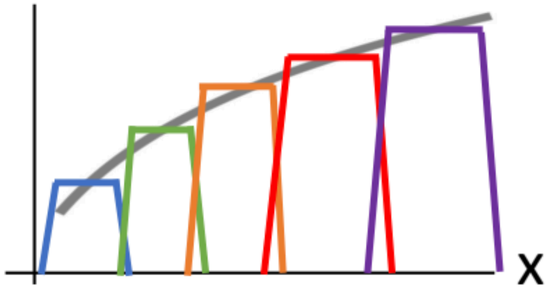
Universal Approximation Theorem

What does this claim?

Universal Approximation Theorem

What does this claim? With enough neurons, a neural network can approximate any function f.

However, this does not mean a neural network can efficiently learn such an approximation.



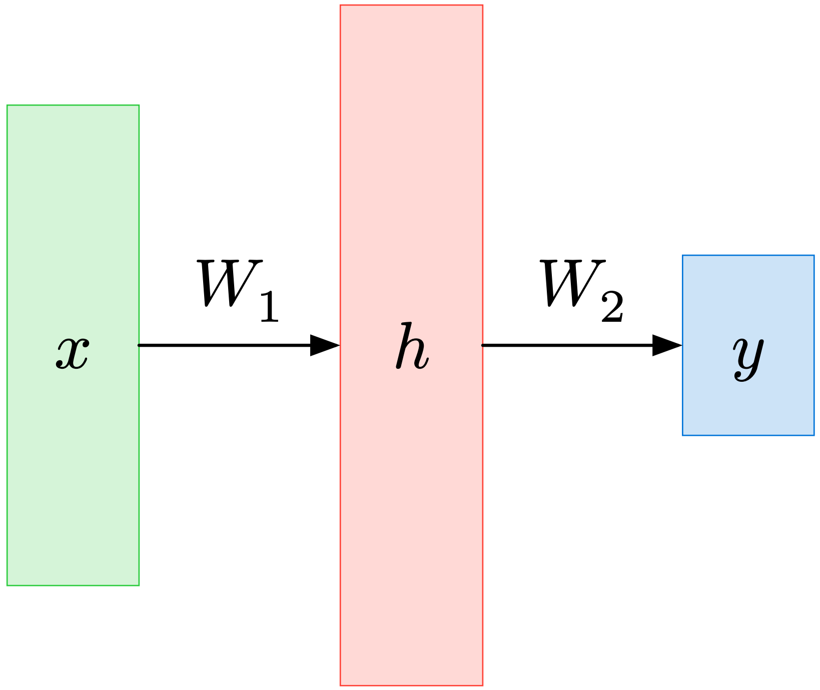
Approximate functions with bumps!

Neural Networks

A **Neural Network** is a computational model that makes decisions and predictions in a way inspired by biological systems.

Generally consists of:

- Input x
- Hidden Layer(s) h_i
- Output Layer y
- Activation Functions $\varphi(x)$



Neural Networks

Suppose we have a neural network $F : \mathbb{R}^{N\times(10\cdot 10)} \to \mathbb{R}^{N\times 1}$ that predicts a class based on a grayscale image.

```
F = nn.Sequential(
  nn.Linear(100, 10, bias=False),
  nn.ReLU(),
  nn.BatchNorm(10),
  nn.Linear(10, 1, bias=True)
)
```

How many **trainable parameters** does F have?

Neural Networks

Suppose we have a neural network $F : \mathbb{R}^{N\times(10\cdot 10)} \to \mathbb{R}^{N\times 1}$ that predicts a class based on a grayscale image.

```
F = nn.Sequential(
  nn.Linear(100, 10, bias=False),
  nn.ReLU(),
  nn.BatchNorm(10),
  nn.Linear(10, 1, bias=True)
)
```

How many **trainable parameters** does F have? $100 \cdot 10 + 2 \cdot 10 + 10 \cdot 1 + 1 = 1031$

Activation Functions

What are desirable properties of activation functions?

Activation Functions

What are desirable properties of activation functions? Non-linear, differentiable, supplies non-vanishing gradients, ...

Why do we need non-linearity anyways?

Activation Functions

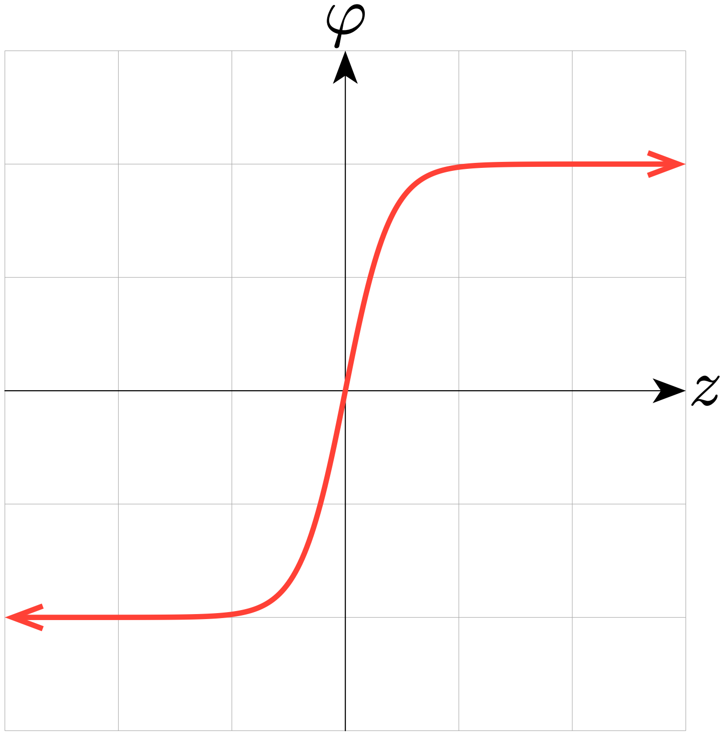
What are desirable properties of activation functions? Non-linear, differentiable, supplies non-vanishing gradients, ...

Why do we need non-linearity anyways? Suppose we had a linear activation φ. You can think of this as another matrix.

$$
\begin{aligned}
y &= W_2\varphi(W_1x + b_1) + b_2 \\
&= W_2\varphi W_1x + W_2\varphi b_1 + b_2 \\
&= (\varphi W_2W_1)x + (\varphi W_2b_1 + b_2)
\end{aligned}
$$

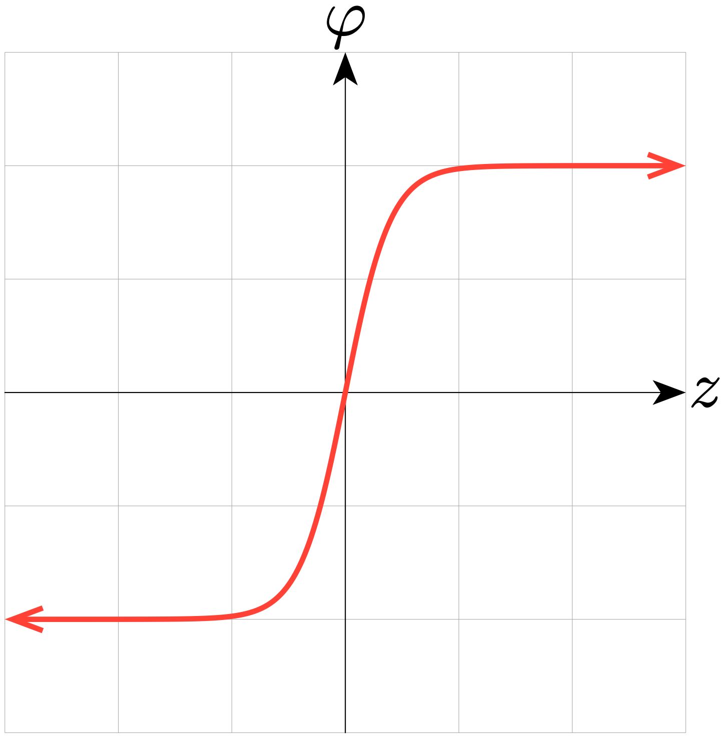
Same as $y = W_3x + b_3$, where $W_3 = \varphi W_2W_1$ and $b_3 = \varphi W_2b_1 + b_2$.

Activation Functions



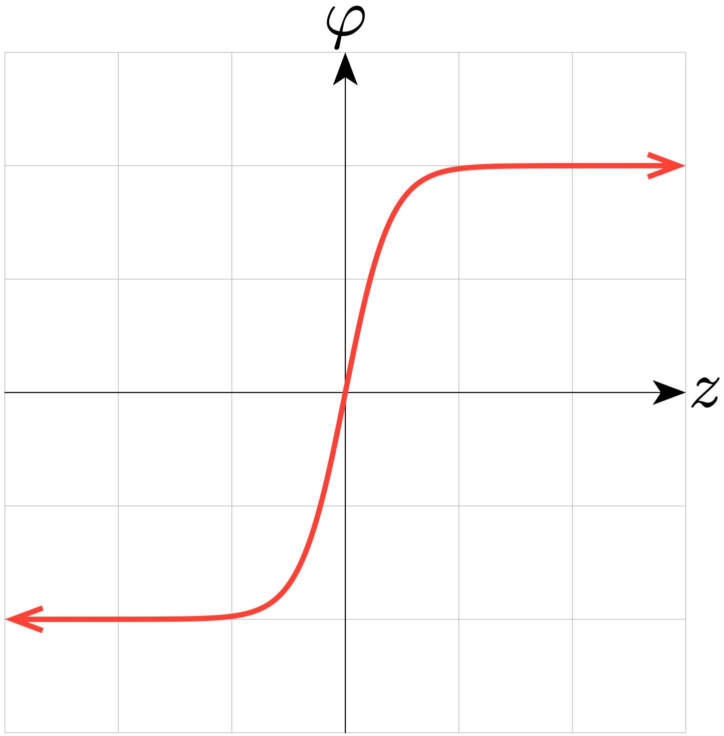
- Name:
- Formula:
- Properties:

Activation Functions



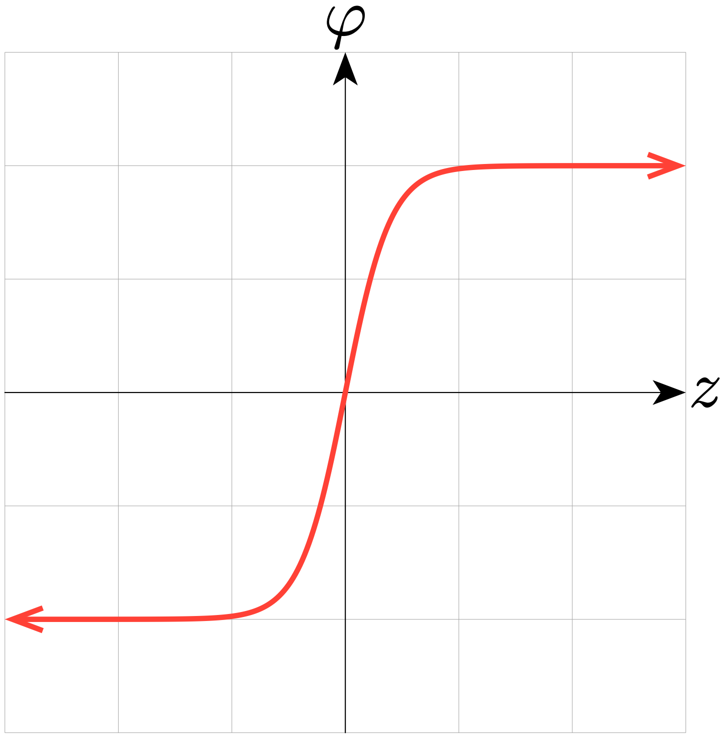
- Name: tanh
- Formula:
- Properties:

Activation Functions



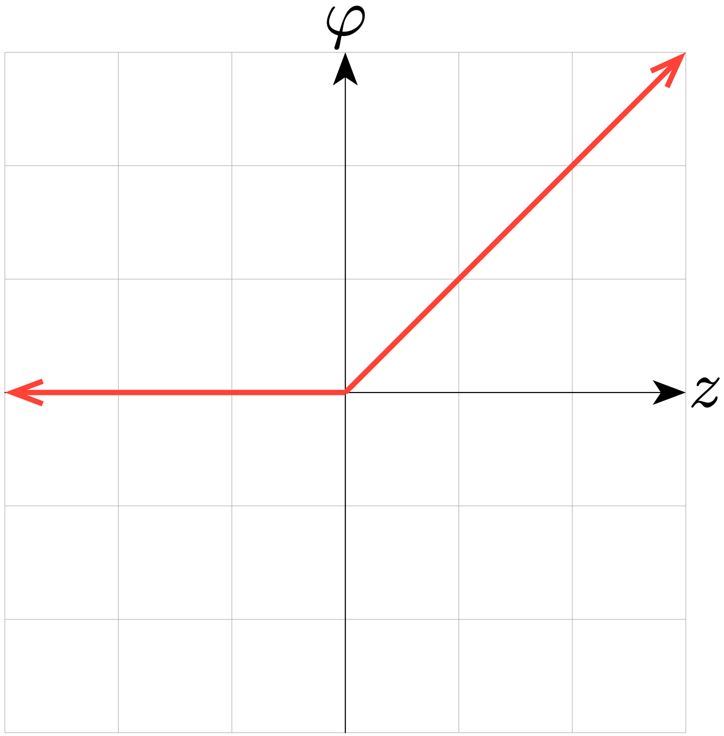
- Name: tanh
- Formula: $\varphi(z) = \frac{e^{2x}-1}{e^{2x}+1}$
- Properties:

Activation Functions



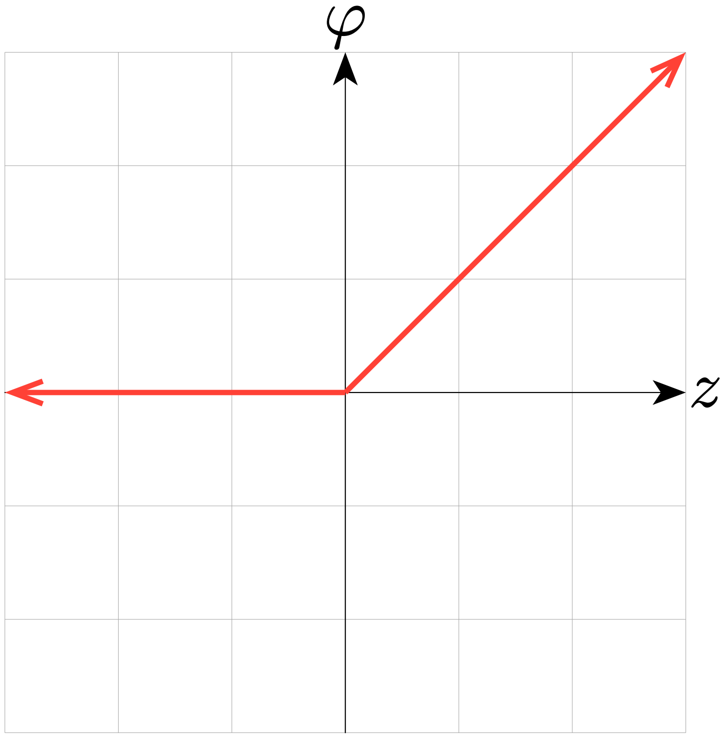
- Name: tanh
- Formula: $\varphi(z) = \frac{e^{2x}-1}{e^{2x}+1}$
- Properties: Non-linear, differentiable everywhere, $\nabla \approx 0$ when $z \to \pm\infty$

Activation Functions



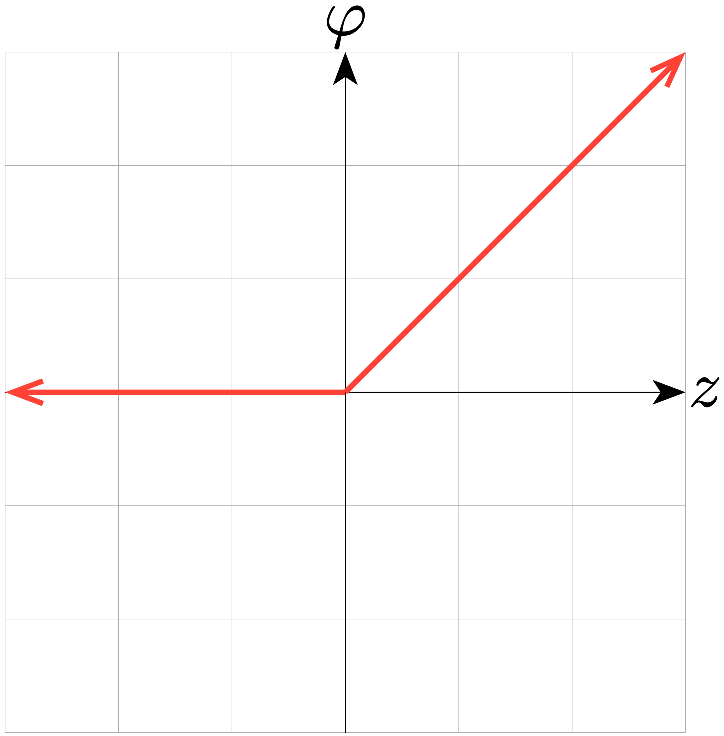
- Name:
- Formula:
- Properties:

Activation Functions



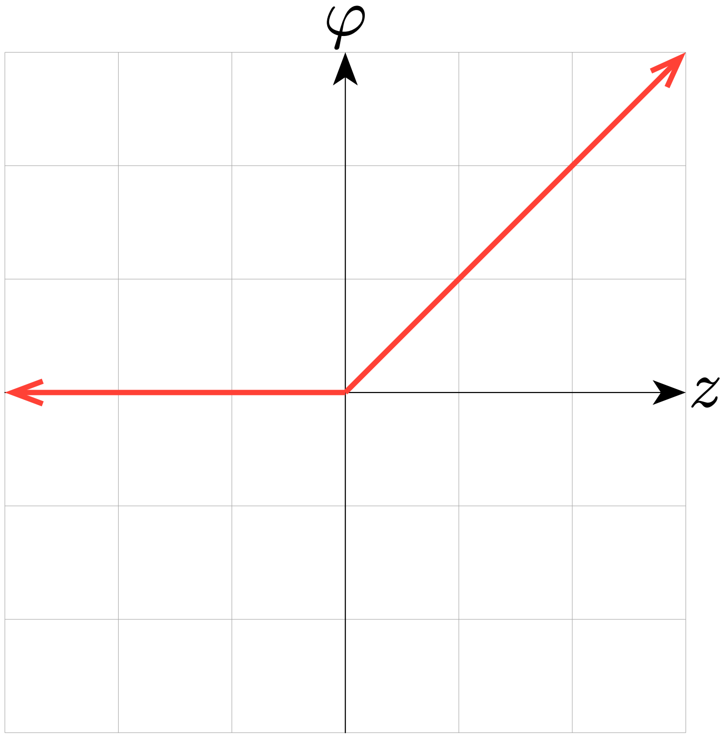
- Name: Rectified Linear Unit (ReLU)
- Formula:
- Properties:

Activation Functions



- Name: Rectified Linear Unit (ReLU)
- Formula: $\varphi(z) = \max(0, z)$
- Properties:

Activation Functions



- Name: Rectified Linear Unit (ReLU)
- Formula: $\varphi(z) = \max(0, z)$
- Properties: Non-linear, differentiable everywhere except $x = 0$, $\nabla > 0$ when $z > 0$

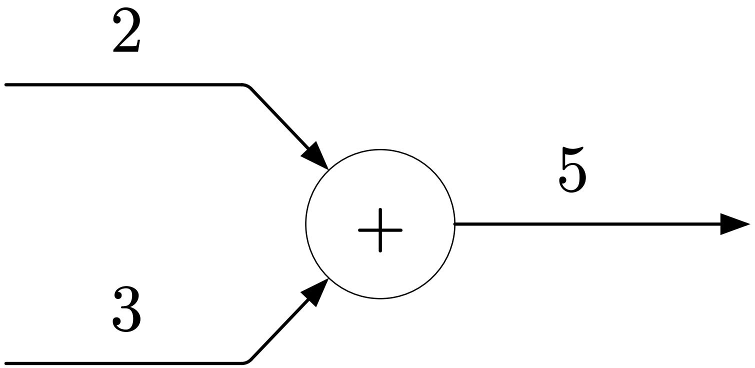
Backpropagation

Helpful gradient gates to remember!

Backprropagation

Helpful gradient gates to remember!

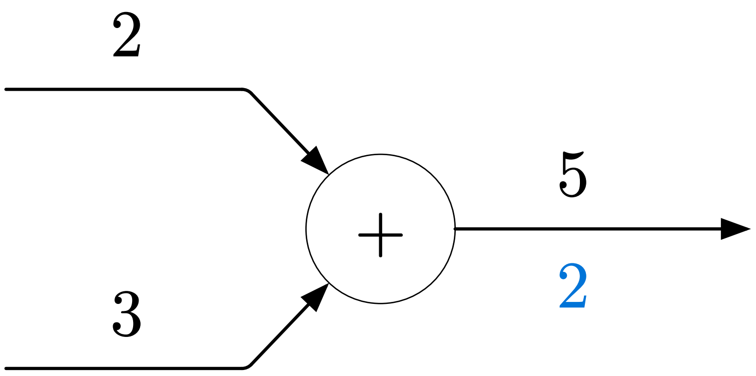
Add Gate



Backpropagation

Helpful gradient gates to remember!

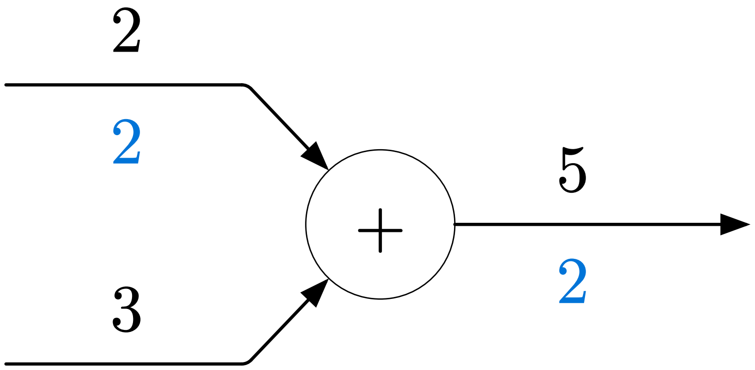
Add Gate



Backpropagation

Helpful gradient gates to remember!

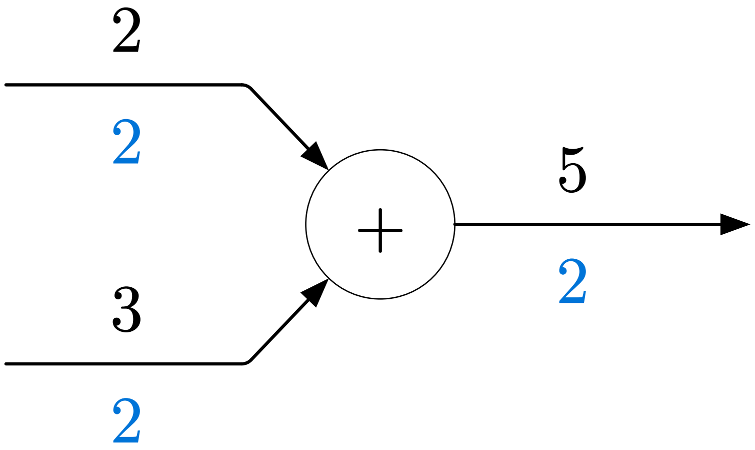
Add Gate



Backpropagation

Helpful gradient gates to remember!

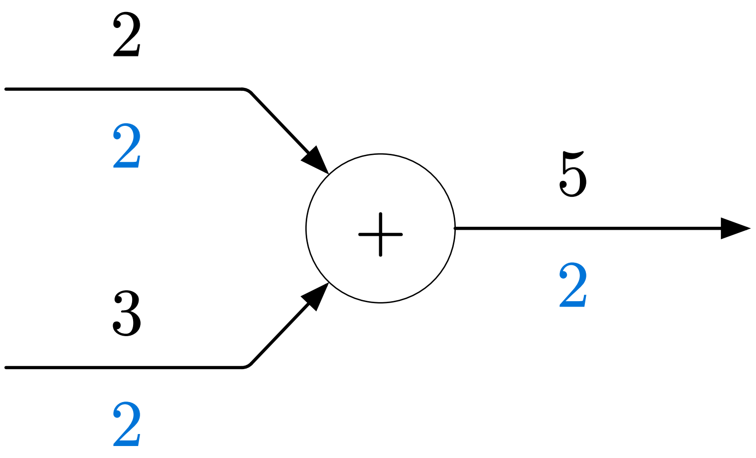
Add Gate



Backpropagation

Helpful gradient gates to remember!

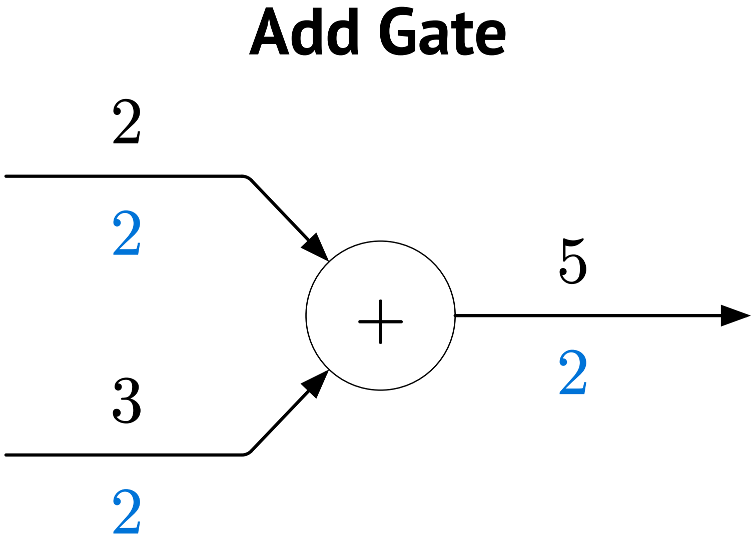
Add Gate



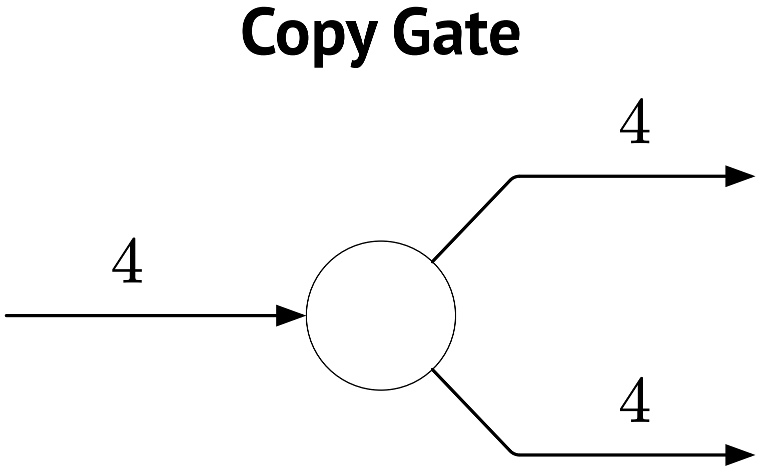
Gradient Distributor

Backpropagation

Helpful gradient gates to remember!



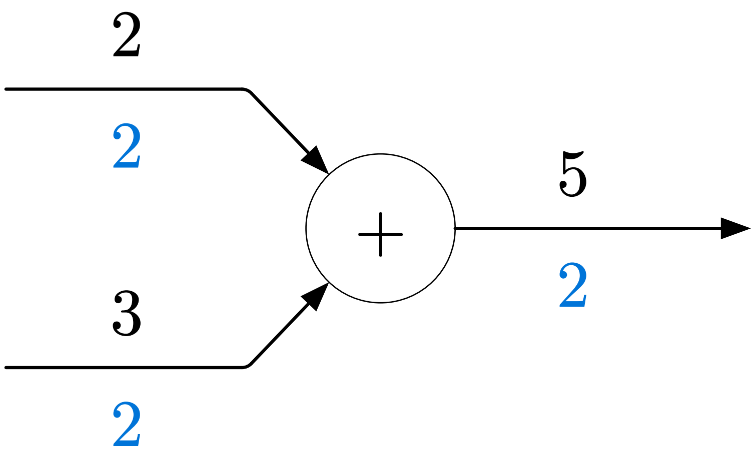
Gradient Distributor



Backpropagation

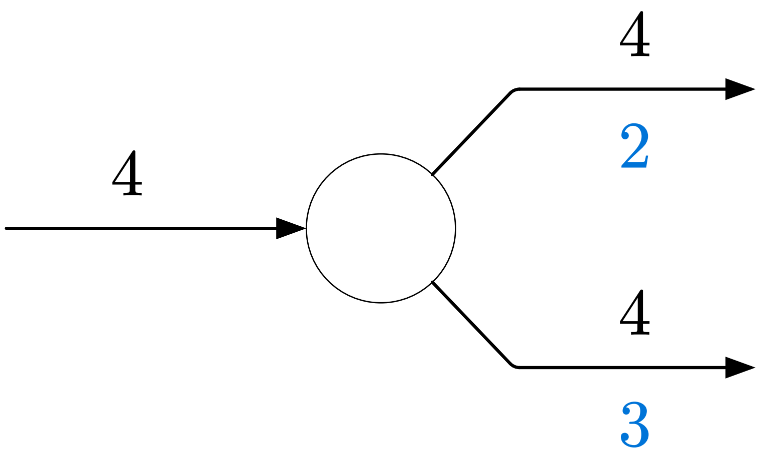
Helpful gradient gates to remember!

Add Gate



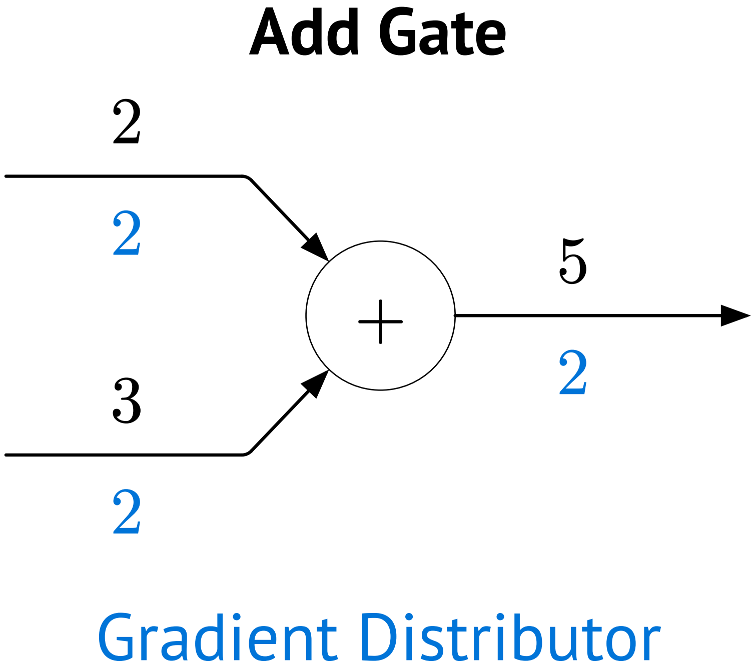
Gradient Distributor

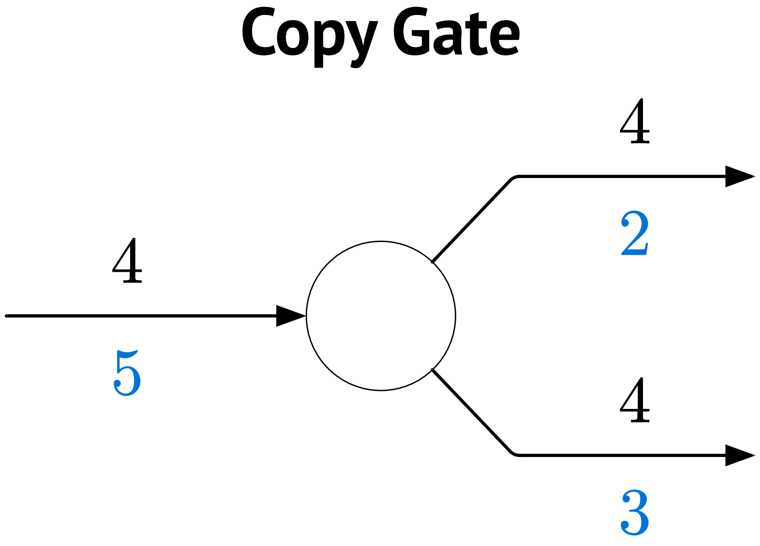
Copy Gate



Backpropagation

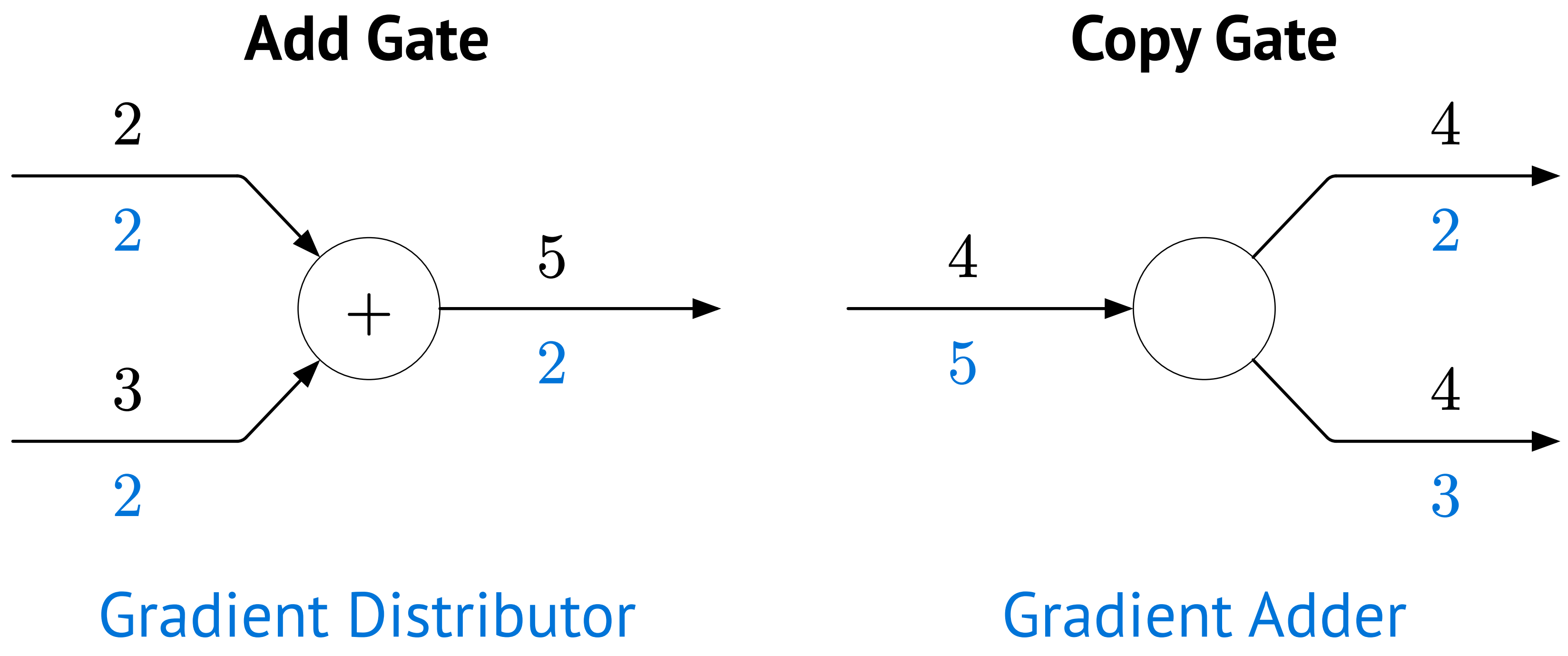
Helpful gradient gates to remember!





Backpropagation

Helpful gradient gates to remember!



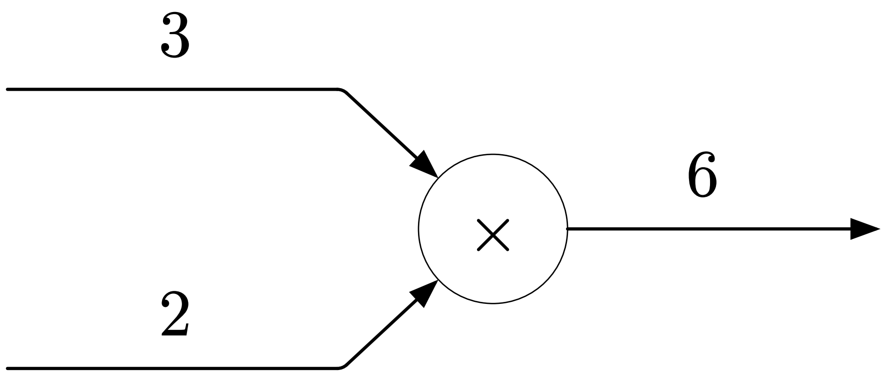
Backpropagation

Helpful gradient gates to remember!

Backpropagation

Helpful gradient gates to remember!

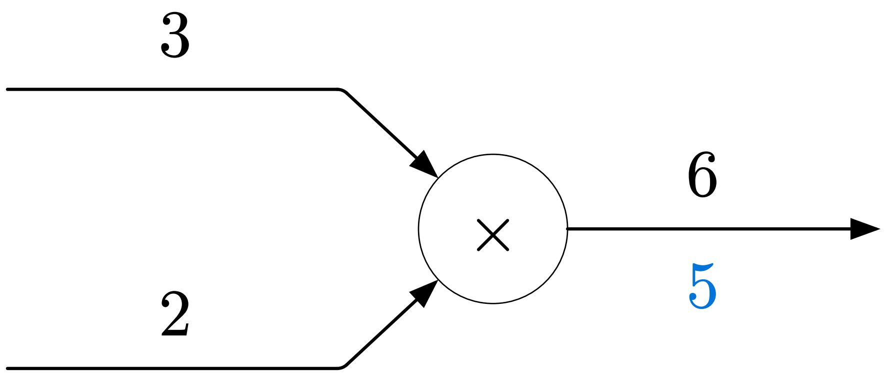
Multiply Gate



Backpropagation

Helpful gradient gates to remember!

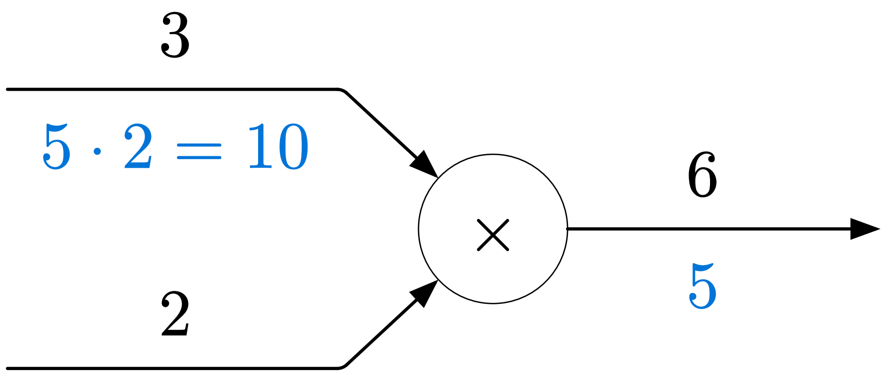
Multiply Gate



Backpropagation

Helpful gradient gates to remember!

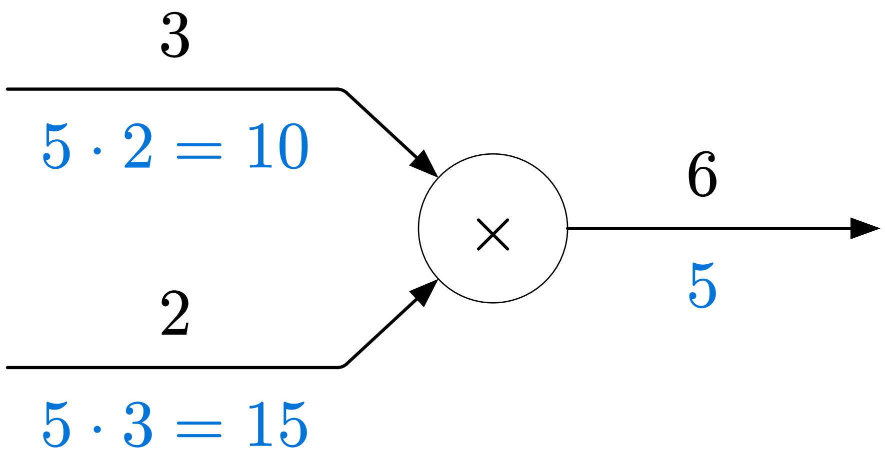
Multiply Gate



Backpropagation

Helpful gradient gates to remember!

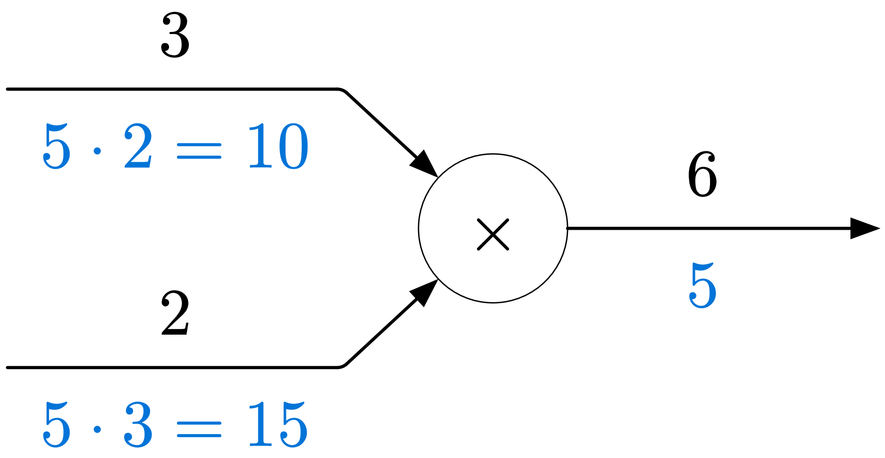
Multiply Gate



Backpropagation

Helpful gradient gates to remember!

Multiply Gate

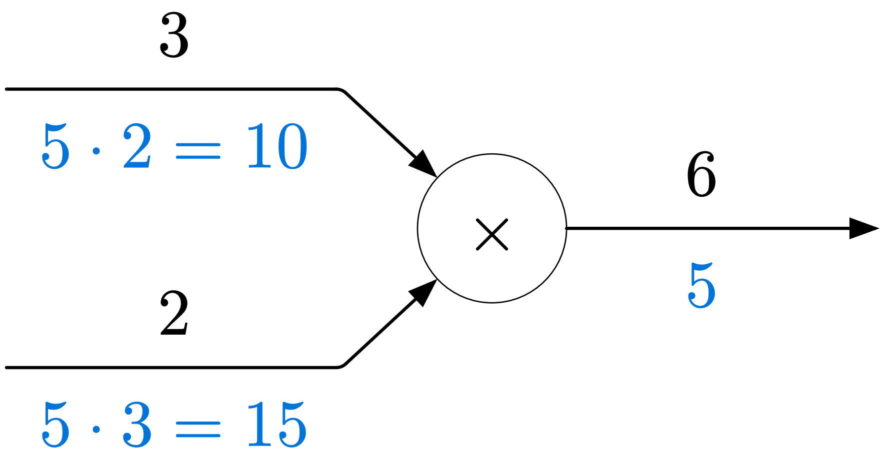


Swap Multiplier

Backpropagation

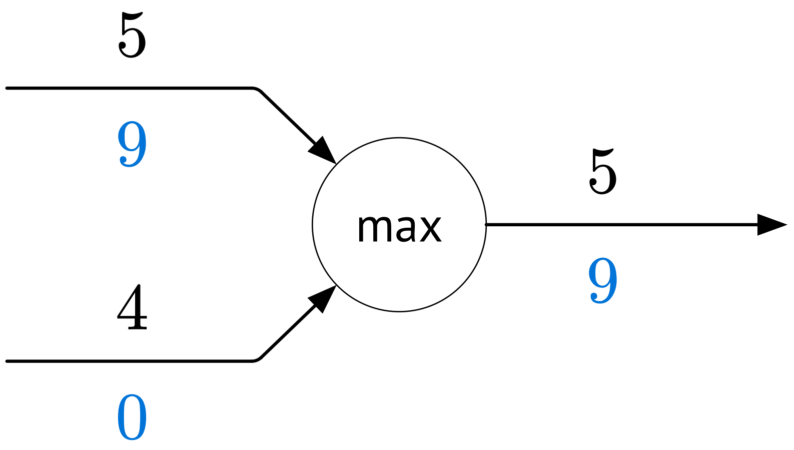
Helpful gradient gates to remember!

Multiply Gate



Swap Multiplier

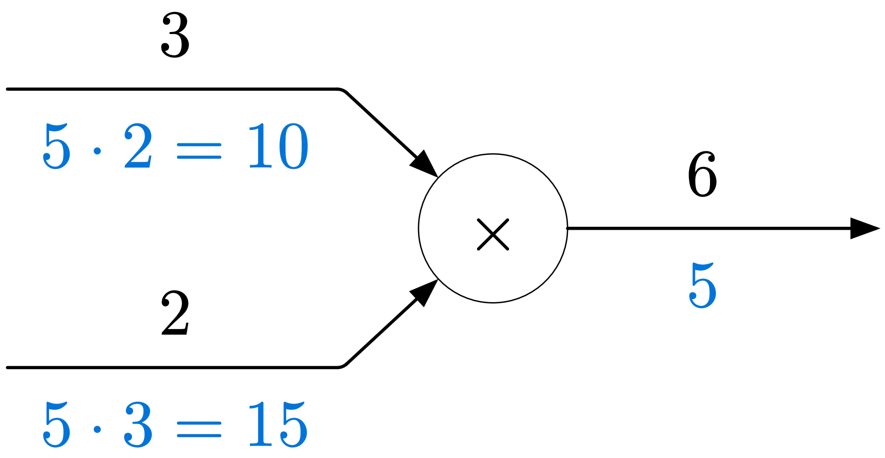
Max Gate



Backpropagation

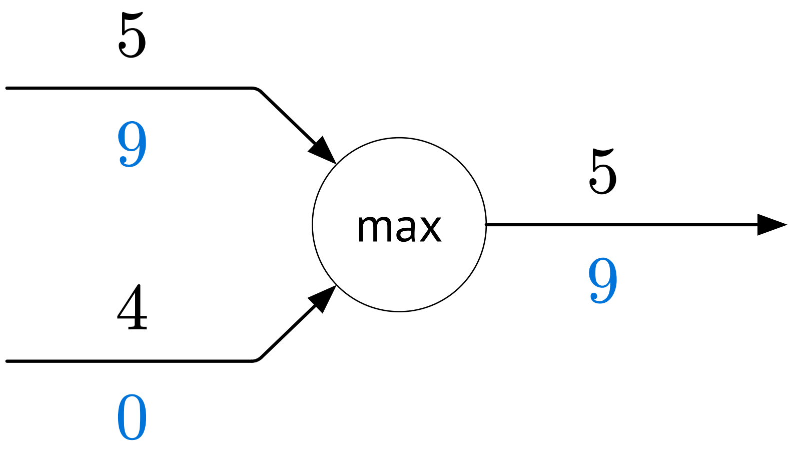
Helpful gradient gates to remember!

Multiply Gate



Swap Multiplier

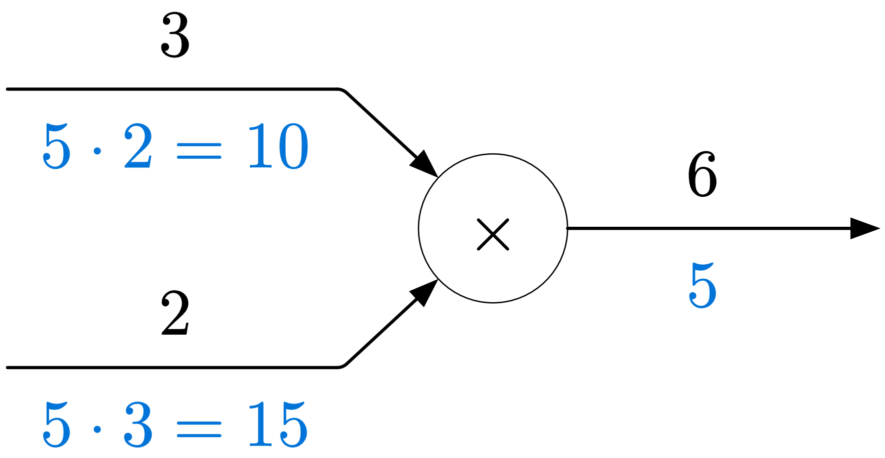
Max Gate



Backpropagation

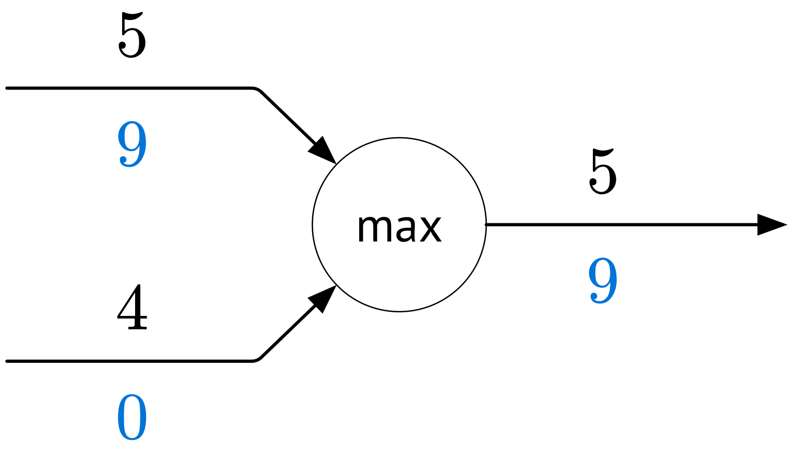
Helpful gradient gates to remember!

Multiply Gate



Swap Multiplier

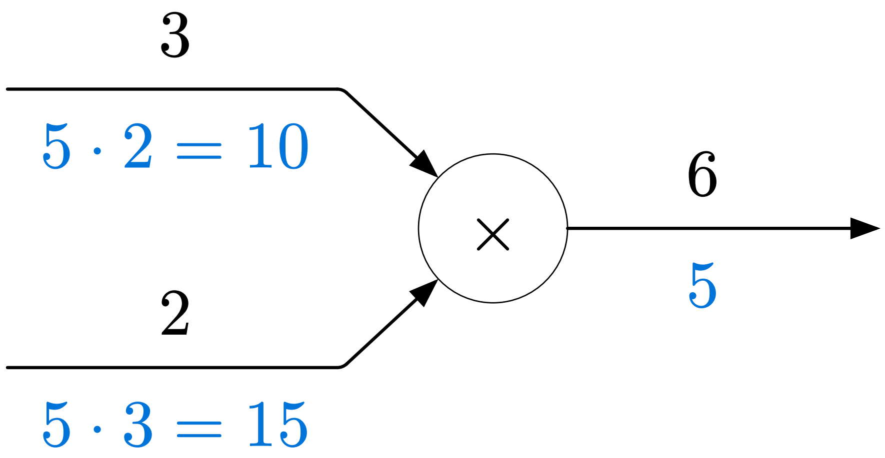
Max Gate



Backpropagation

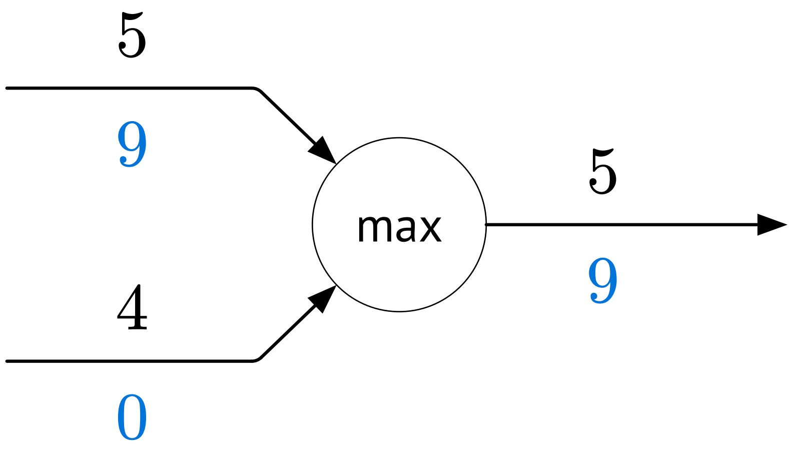
Helpful gradient gates to remember!

Multiply Gate



Swap Multiplier

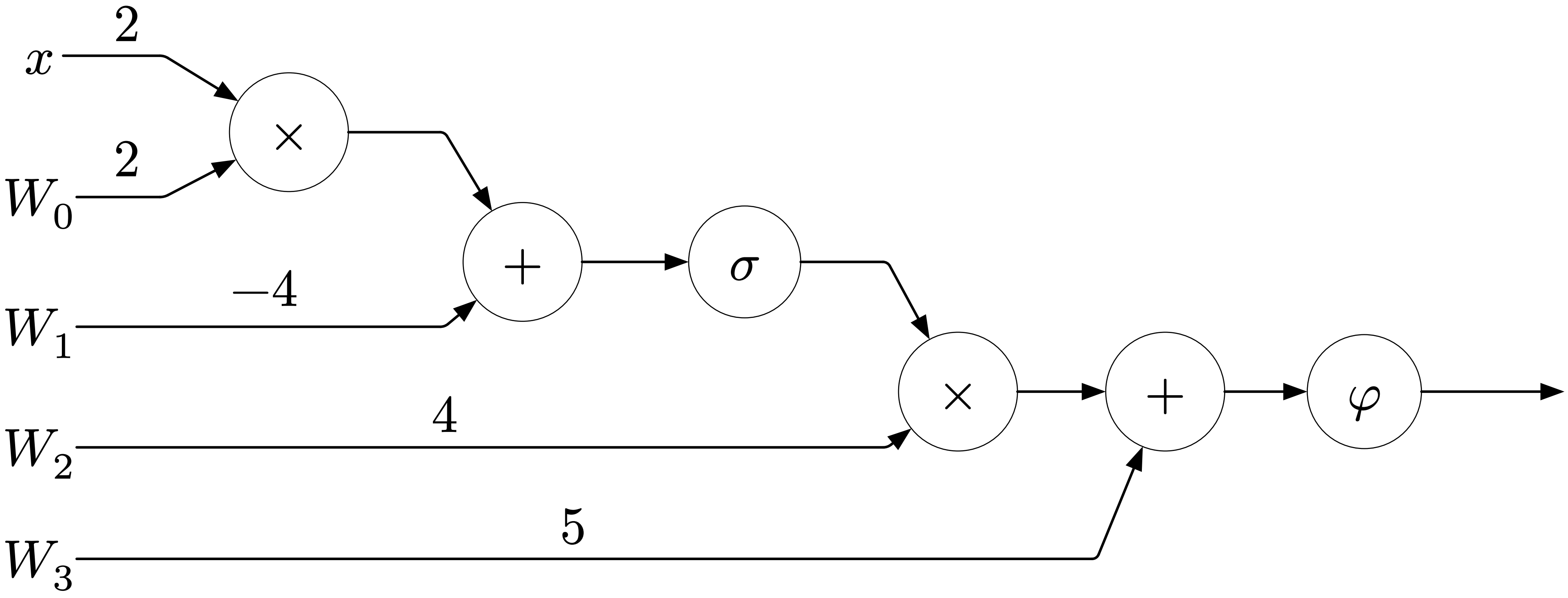
Max Gate



Gradient Router

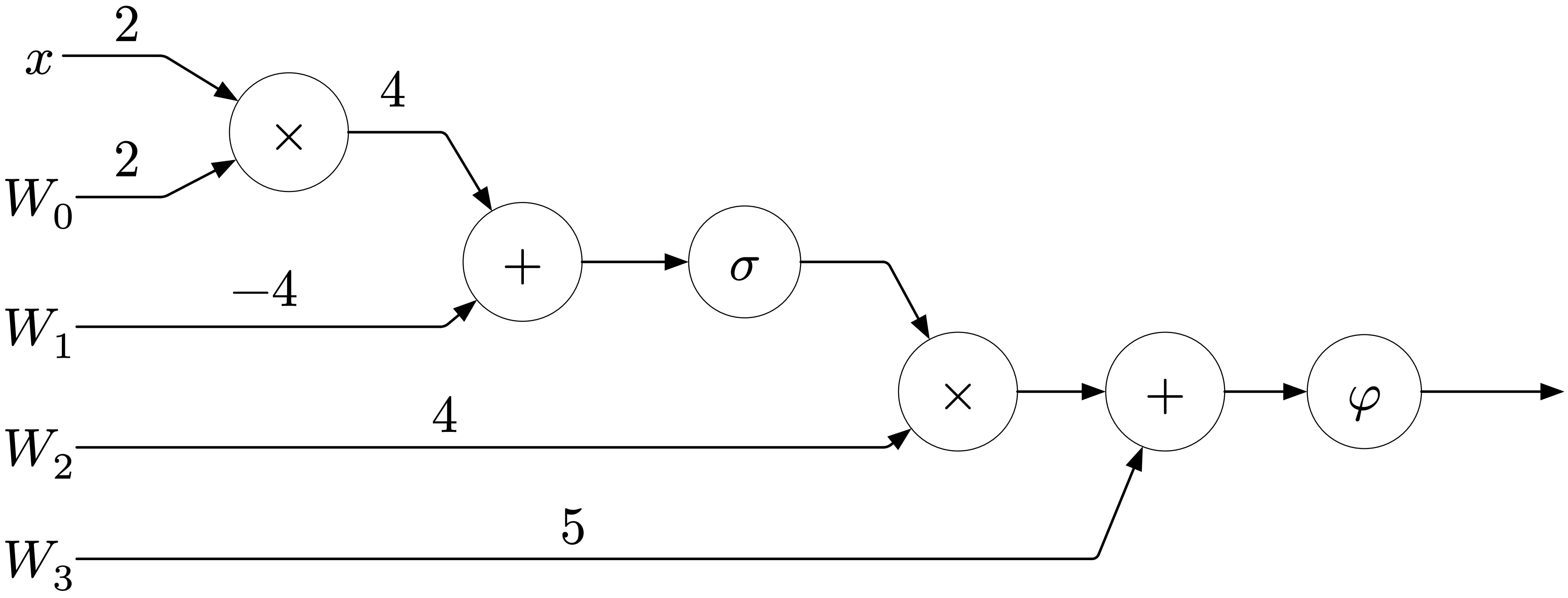
Backpropagation

Compute the forward and backward pass. φ is a ReLU activation.



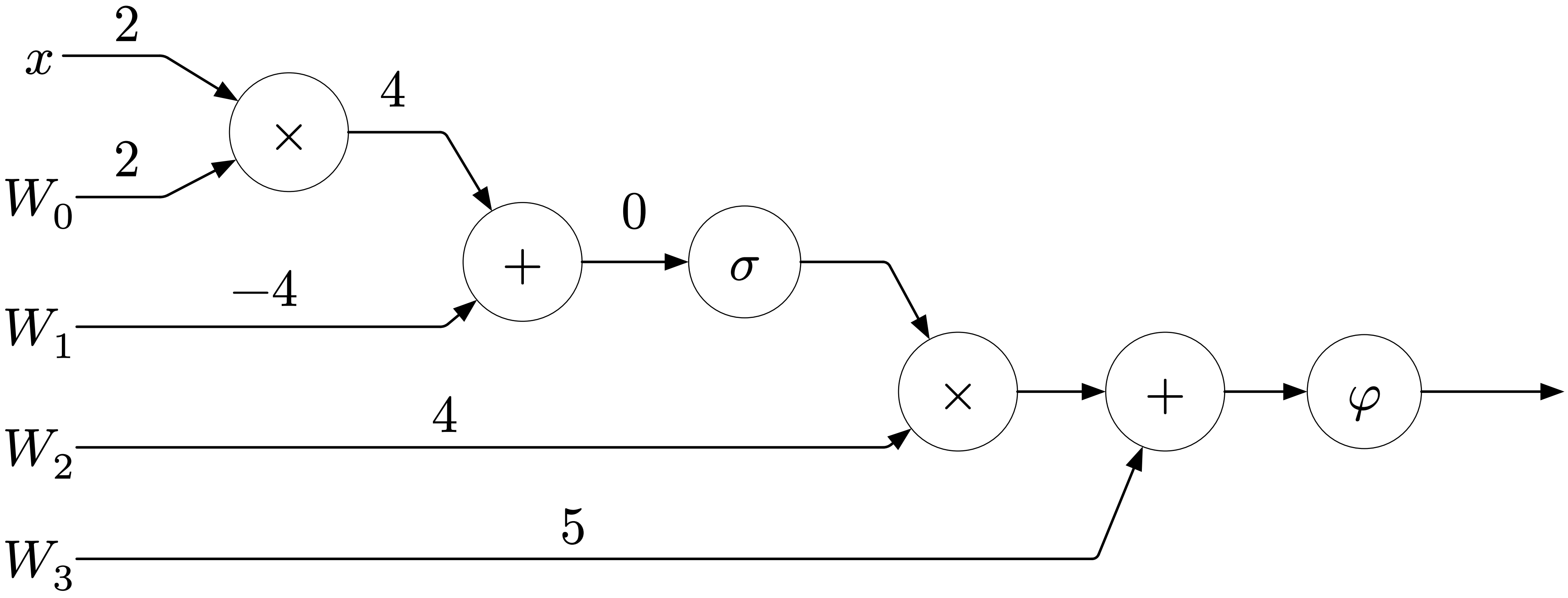
Backpropagation

Compute the forward and backward pass. φ is a ReLU activation.



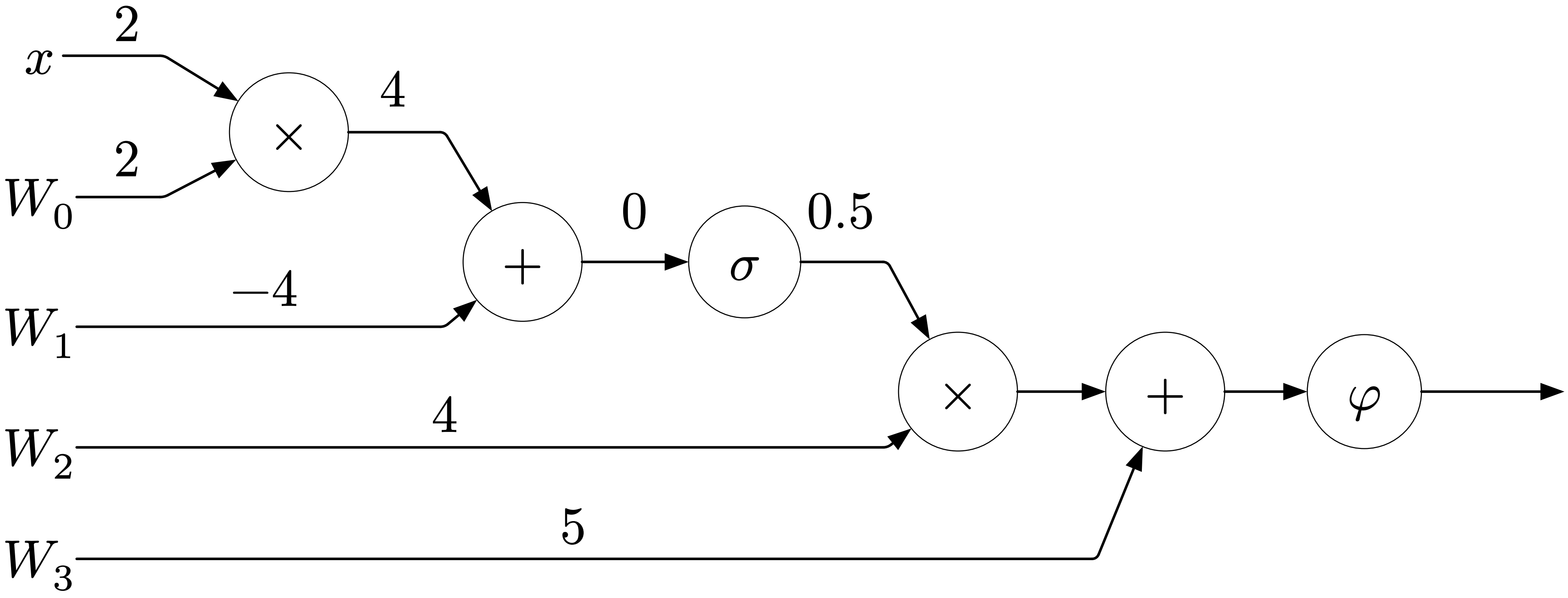
Backpropagation

Compute the forward and backward pass. φ is a ReLU activation.



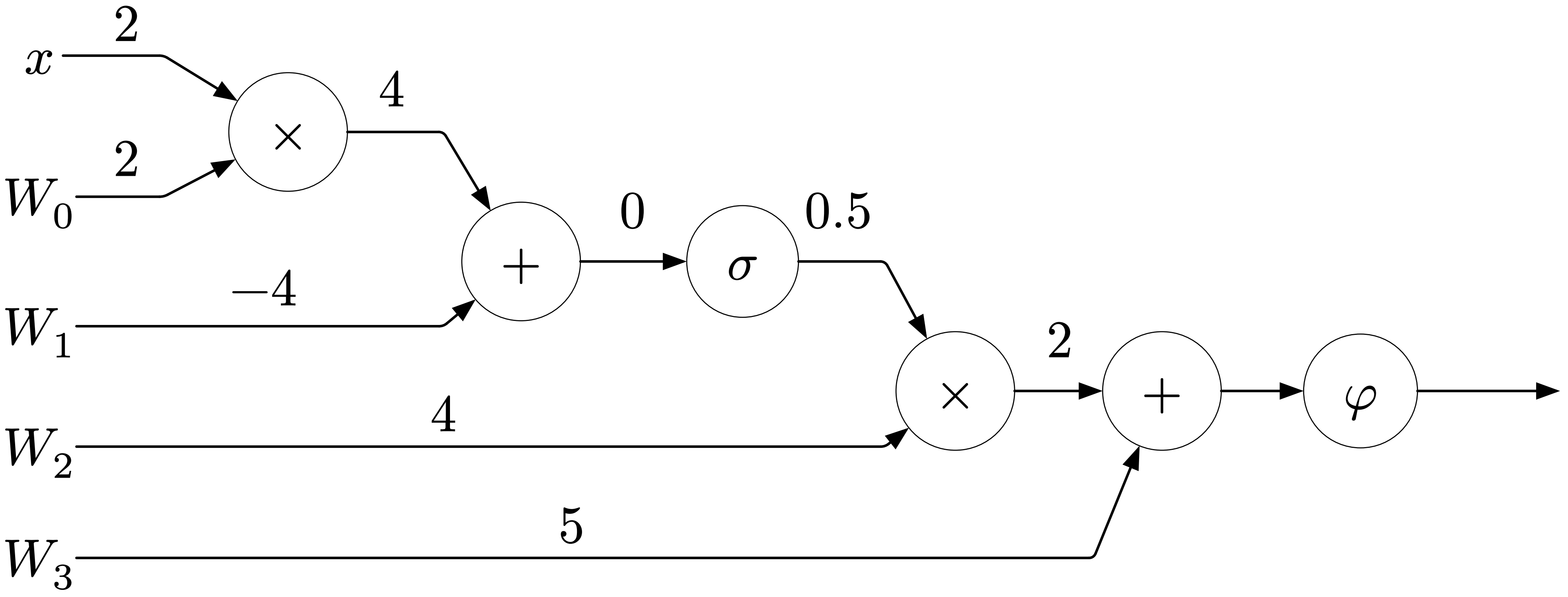
Backpropagation

Compute the forward and backward pass. φ is a ReLU activation.



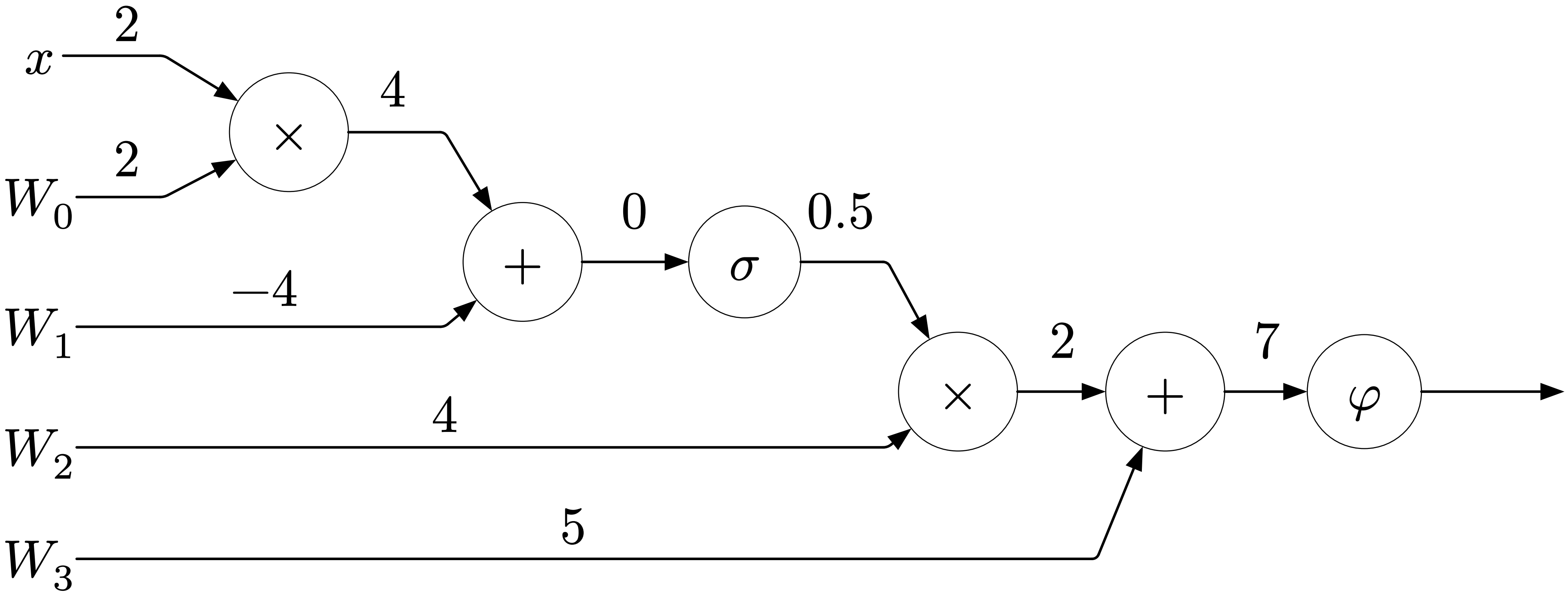
Backpropagation

Compute the forward and backward pass. φ is a ReLU activation.



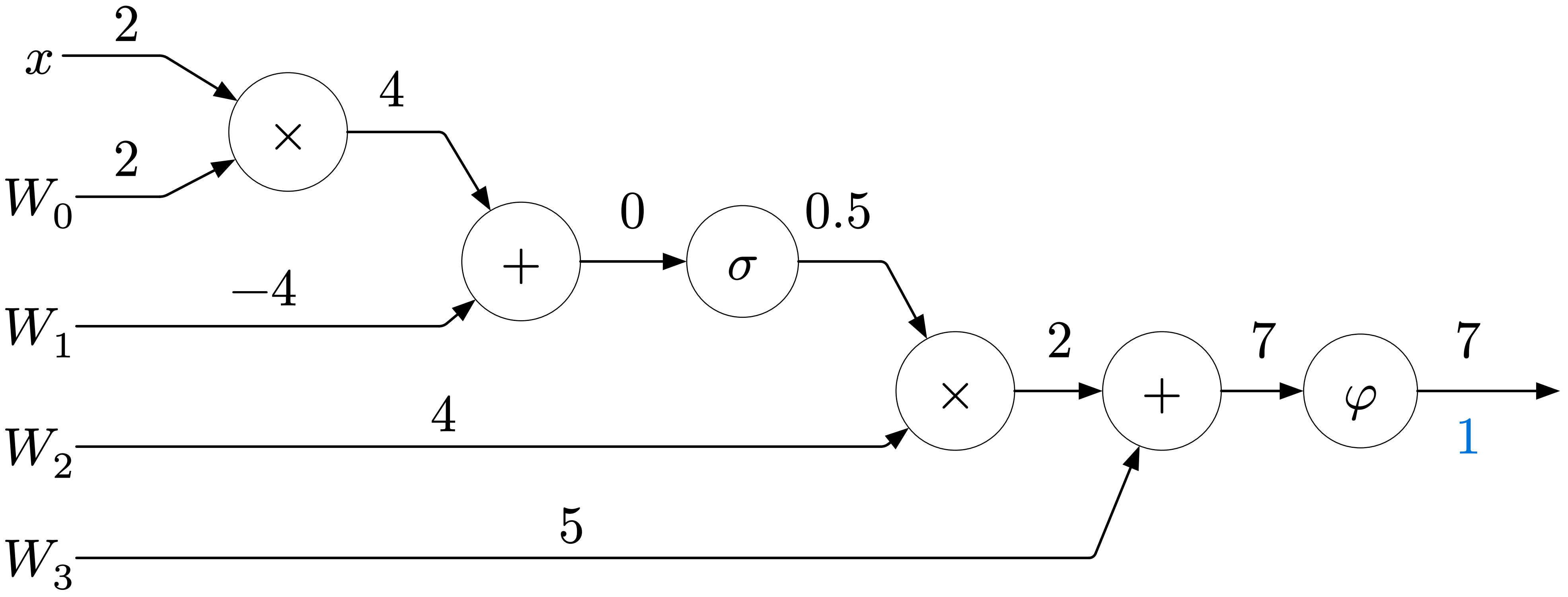
Backpropagation

Compute the forward and backward pass. φ is a ReLU activation.



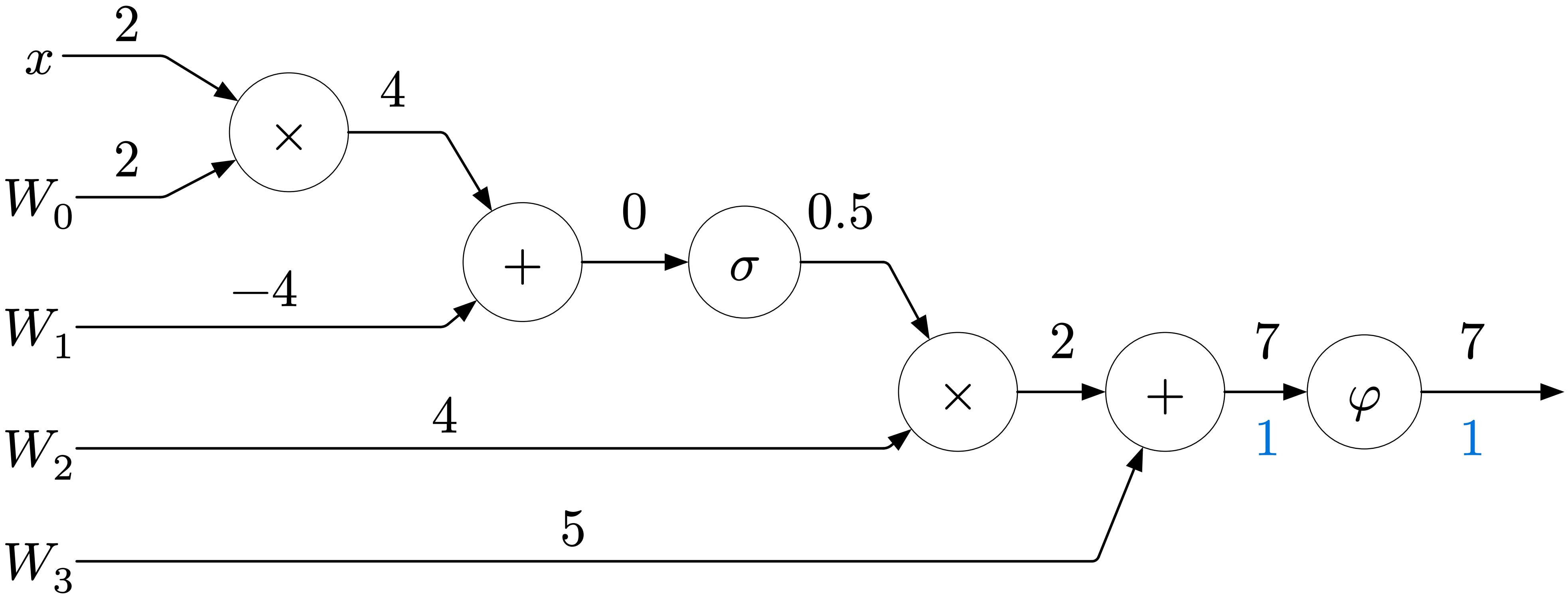
Backpropagation

Compute the forward and backward pass. φ is a ReLU activation.



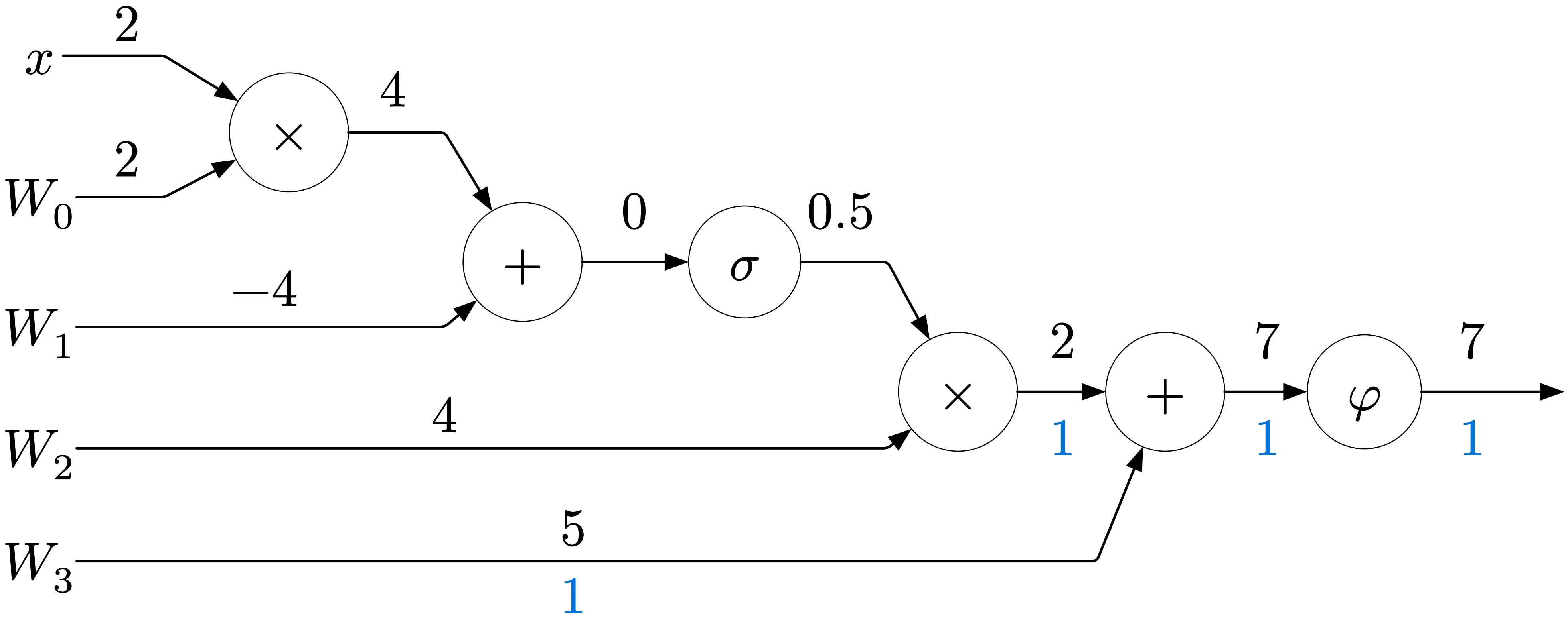
Backpropagation

Compute the forward and backward pass. φ is a ReLU activation.



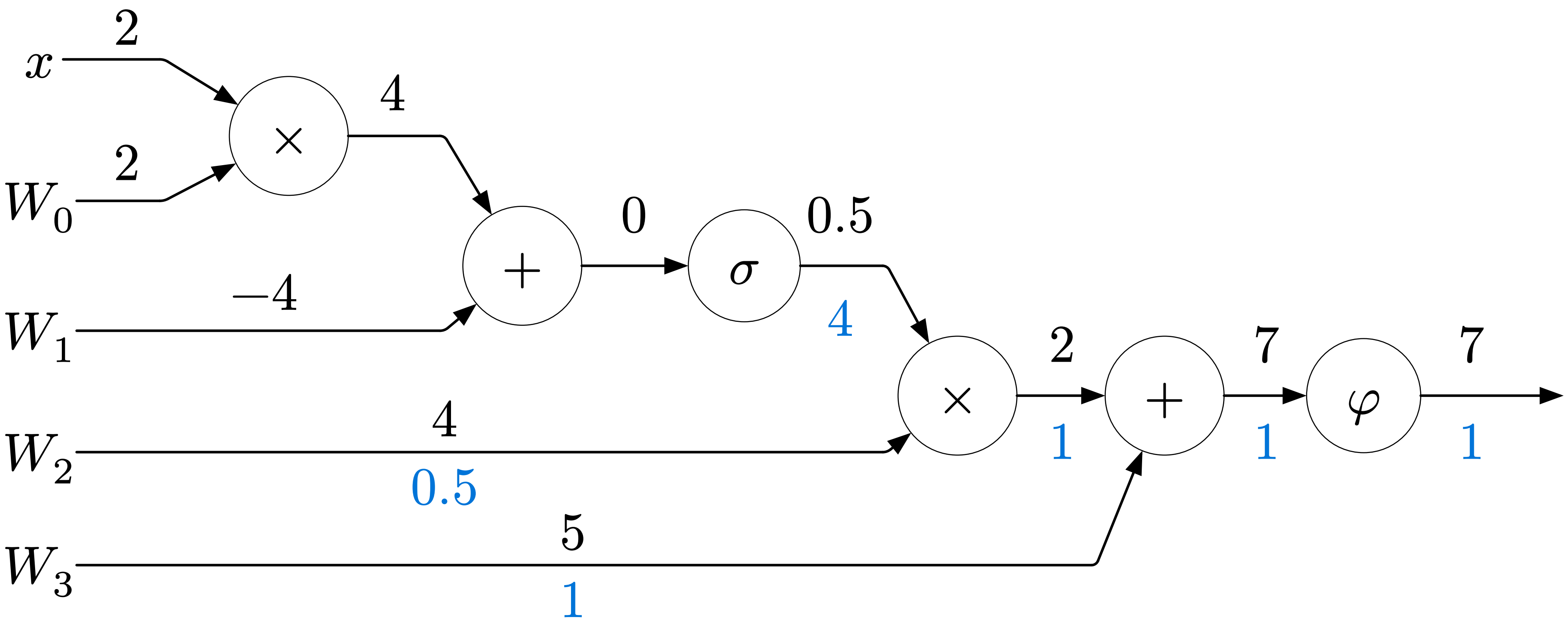
Backpropagation

Compute the forward and backward pass. φ is a ReLU activation.



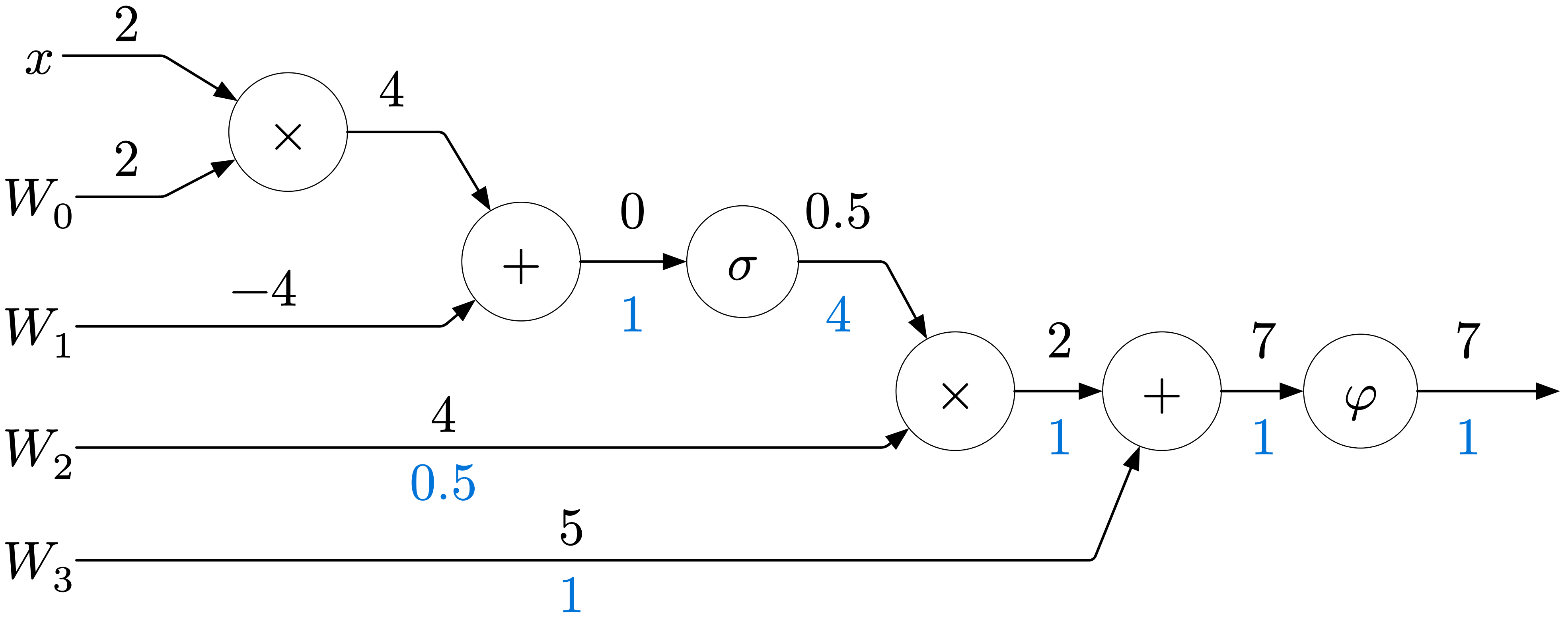
Backpropagation

Compute the forward and backward pass. φ is a ReLU activation.



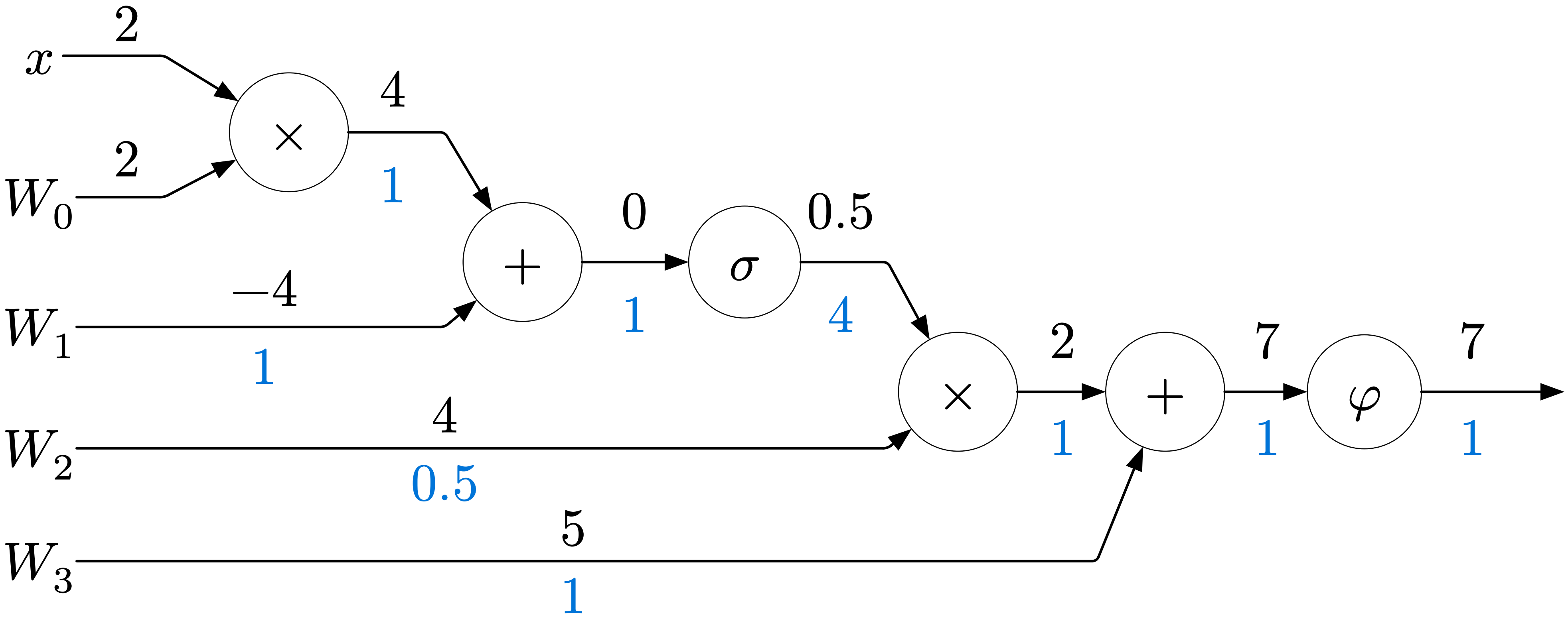
Backpropagation

Compute the forward and backward pass. φ is a ReLU activation.



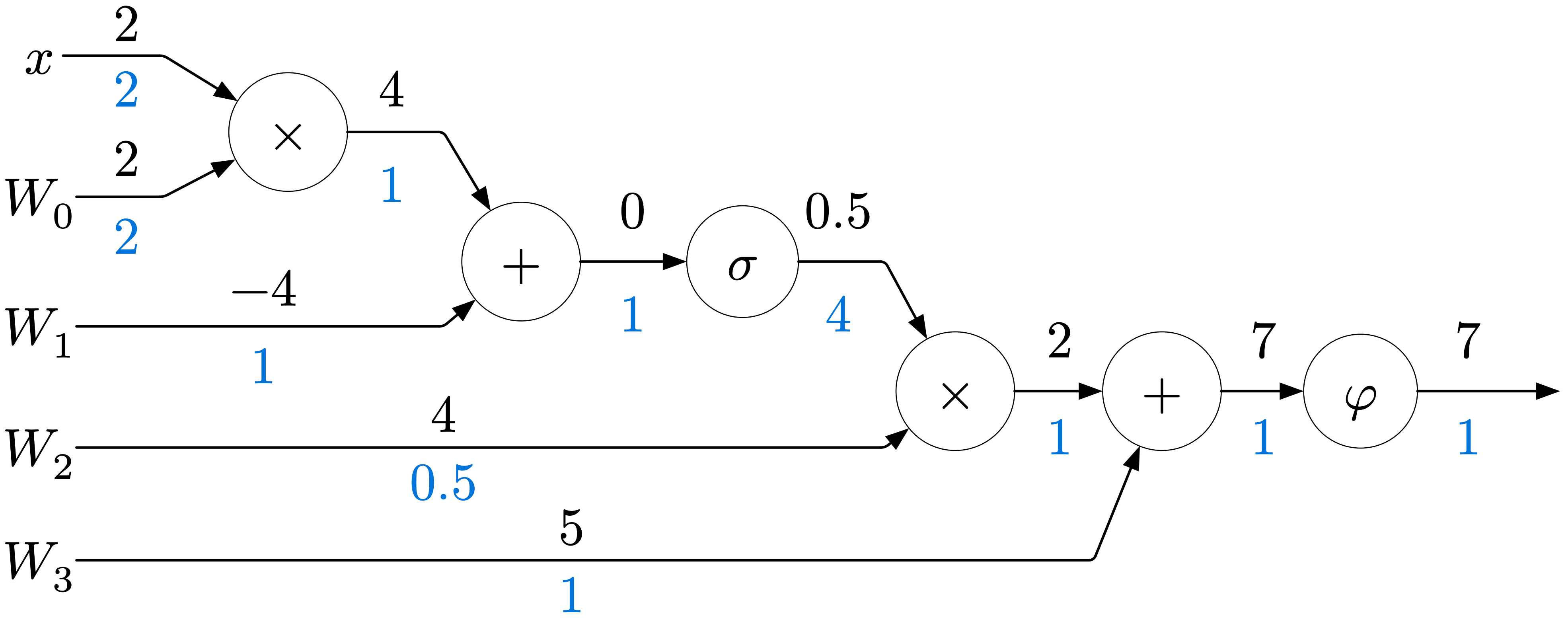
Backpropagation

Compute the forward and backward pass. φ is a ReLU activation.



Backpropagation

Compute the forward and backward pass. φ is a ReLU activation.



Normalization

Problem: Input features can have different scales, impacting the efficiency and performance of training.

Normalization

Problem: Input features can have different scales, impacting the efficiency and performance of training.

Solution: **Input Normalization** can be applied on the network inputs using dataset statistics.

Normalization

Problem: Input features can have different scales, impacting the efficiency and performance of training.

Solution: **Input Normalization** can be applied on the network inputs using dataset statistics.

$$\hat{\mu}_j = \frac{1}{N}\sum_{i=1}^{N} x_{i,j}$$

$$\sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N} \left(x_{i,j} - \hat{\mu}_j\right)^2$$

Normalization

Problem: Inputs to hidden layers are not normalized and we don't have prior feature statistics.

Normalization

Problem: Inputs to hidden layers are not normalized and we don't have prior feature statistics.

Solution: Aggregate statistics to normalize and learn scale + shift parameters with **Batch Normalization**.

Normalization

Problem: Inputs to hidden layers are not normalized and we don't have prior feature statistics.

Solution: Aggregate statistics to normalize and learn scale + shift parameters with **Batch Normalization**.

μ_j, σ_j^2 are running avg

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

$$y_{i,j} = \gamma_j \hat{x}_{i,j} + \beta_j$$

At test time, use aggregated statistics from training.

Convolutions

Which of the following is true about the **receptive field** of a convolutional neural network as more layers are added?

Convolutions

Which of the following is true about the **receptive field** of a convolutional neural network as more layers are added?

a. It increases and depends on the kernel size and stride of each layer.
b. It remains the same regardless of the depth of the network.
c. It only depends on the kernel size of the first layer.
d. It decreases with each additional convolutional layer.

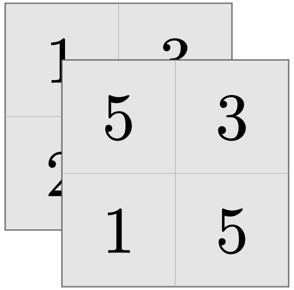
Convolutions

Which of the following is true about the **receptive field** of a convolutional neural network as more layers are added?

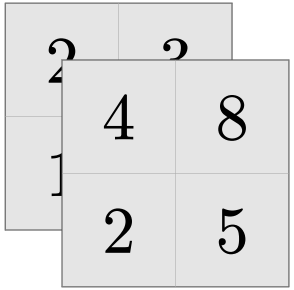
a. It increases and depends on the kernel size and stride of each layer.
b. It remains the same regardless of the depth of the network.
c. It only depends on the kernel size of the first layer.
d. It decreases with each additional convolutional layer.

Convolutions

Given an input $x \in \mathbb{R}^{H \times W \times C}$, where $H = W = C = 2$, compute the output of a convolution with a **kernel** $K \in \mathbb{R}^{2 \times 2 \times 2}$, **padding** of 1, and **stride** of 2.



Input



Kernel

Convolutions

	Input		Kernel	
Channel 1	1	3	2	3
	2	4	1	6
Channel 2	5	3	4	8
	1	5	2	5

Convolutions

$$\begin{bmatrix}0&0&0&0\\0&1&3&0\\0&2&4&0\\0&0&0&0\end{bmatrix} \quad \begin{bmatrix}2&3\\1&6\end{bmatrix} \quad \rightarrow \quad \begin{bmatrix}6&\ \\ \ &\ \end{bmatrix}$$

Convolutions

$$\begin{bmatrix}0&0&0&0\\0&1&3&0\\0&2&4&0\\0&0&0&0\end{bmatrix} \quad \begin{bmatrix}2&3\\1&6\end{bmatrix} \rightarrow \begin{bmatrix}6&3\\ & \end{bmatrix}$$

Convolutions

$$\begin{array}{|c|c|c|c|}\hline 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 3 & 0 \\ \hline 0 & 2 & 4 & 0 \\ \hline 0 & 0 & 0 & 0 \\ \hline \end{array} \quad \begin{array}{|c|c|}\hline 2 & 3 \\ \hline 1 & 6 \\ \hline \end{array} \quad \rightarrow \quad \begin{array}{|c|c|}\hline 6 & 3 \\ \hline 6 & \\ \hline \end{array}$$

Convolutions

$$\begin{array}{|c|c|c|c|}\hline 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 3 & 0 \\ \hline 0 & 2 & 4 & 0 \\ \hline 0 & 0 & 0 & 0 \\ \hline \end{array} \quad \begin{array}{|c|c|}\hline 2 & 3 \\ \hline 1 & 6 \\ \hline \end{array} \quad \rightarrow \quad \begin{array}{|c|c|}\hline 6 & 3 \\ \hline 6 & \boxed{8} \\ \hline \end{array}$$

Convolutions

$$\begin{array}{|c|c|c|c|}\hline 0 & 0 & 0 & 0 \\ \hline 0 & 5 & 3 & 0 \\ \hline 0 & 1 & 5 & 0 \\ \hline 0 & 0 & 0 & 0 \\ \hline \end{array} \quad \begin{array}{|c|c|}\hline 4 & 8 \\ \hline 2 & 5 \\ \hline \end{array} \quad \rightarrow \quad \begin{array}{|c|c|}\hline 25 & \\ \hline & \\ \hline \end{array}$$

Convolutions

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 5 & 3 & 0 \\ 0 & 1 & 5 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 4 & 8 \\ 2 & 5 \end{bmatrix} \quad \rightarrow \quad \begin{bmatrix} 25 & 6 \\ & \end{bmatrix}$$

Convolutions

$$\begin{bmatrix}0&0&0&0\\0&5&3&0\\0&1&5&0\\0&0&0&0\end{bmatrix} \qquad \begin{bmatrix}4&8\\2&5\end{bmatrix} \qquad \rightarrow \qquad \begin{bmatrix}25&6\\8&\end{bmatrix}$$

Convolutions

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 5 & 3 & 0 \\ 0 & 1 & 5 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 4 & 8 \\ 2 & 5 \end{bmatrix} \quad \rightarrow \quad \begin{bmatrix} 25 & 6 \\ 8 & 20 \end{bmatrix}$$

Convolutions

Which operation reduces feature map size the most?

Convolutions

Which operation reduces feature map size the most?

a. Global Average Pooling
b. 2×2 Max Pooling with stride of 3
c. Softmax activation
d. 5×5 Convolution

Convolutions

Which operation reduces feature map size the most?

a. Global Average Pooling
b. 2×2 Max Pooling with stride of 3
c. Softmax activation
d. 5×5 Convolution

Convolutions

A convolution layer has 16 **filters** $K \in \mathbb{R}^{7\times 7\times 8}$ (note: $\mathbb{R}^{H\times W\times C}$), padding of 2, and stride of 3. Compute the number of parameters, including bias.

Convolutions

A convolution layer has 16 **filters** $K \in \mathbb{R}^{7\times 7\times 8}$ (note: $\mathbb{R}^{H\times W\times C}$), padding of 2, and stride of 3. Compute the number of parameters, including bias.

$$16(7\cdot 7\cdot 8+1) = 16\cdot 7\cdot 7\cdot 8+16 = 6288$$

Convolutions

A convolution layer has 16 **filters** $K \in \mathbb{R}^{7\times 7\times 8}$ (note: $\mathbb{R}^{H\times W\times C}$), padding of 2, and stride of 3. Compute the number of parameters, including bias.

$$16(7\cdot 7\cdot 8+1) = 16\cdot 7\cdot 7\cdot 8 + 16 = 6288$$

State the dimension of output y for an input $x \in \mathbb{R}^{32\times 32\times 8}$?

Convolutions

A convolution layer has 16 **filters** $K \in \mathbb{R}^{7\times7\times8}$ (note: $\mathbb{R}^{H\times W\times C}$), padding of 2, and stride of 3. Compute the number of parameters, including bias.

$$16(7\cdot7\cdot8+1) = 16\cdot7\cdot7\cdot8+16 = 6288$$

State the dimension of output y for an input $x \in \mathbb{R}^{32\times32\times8}$?

$$H_{\text{out}} = \left\lfloor \frac{H_{\text{in}} - k + 2p}{s} \right\rfloor + 1 = 10, W_{\text{out}} = \left\lfloor \frac{W_{\text{in}} - k + 2p}{s} \right\rfloor + 1 = 10$$

$$C_{\text{out}} = 16$$

Convolutions

Which of the following is true about convolutions?

Convolutions

Which of the following is true about convolutions?

a. The number of biases is equal to the number of filters.
b. Training a convolutional neural network (CNN) involves no **inductive bias**.
c. The number of output channels is equal to the number of filters.
d. **Dilated convolutions** increase the receptive field.

Convolutions

Which of the following is true about convolutions?

a. The number of biases is equal to the number of filters.
b. Training a convolutional neural network (CNN) involves no **inductive bias**.
c. The number of output channels is equal to the number of filters.
d. **Dilated convolutions** increase the receptive field.