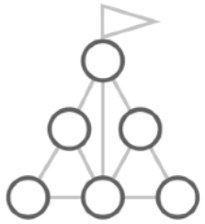


Deep Learning - Optimization and Backpropagation

Itthi Chatnuntawech

Optimizing Our Model

- To train our model f_W, we adjust its weights W in a way that decreases your loss function
- Particularly, we are minimizing our loss function with respect to W

prepared output/ labels/targets — prepared input

$$L(y, \hat{y}) = L(y, f_W(x))$$

predicted output — Weights

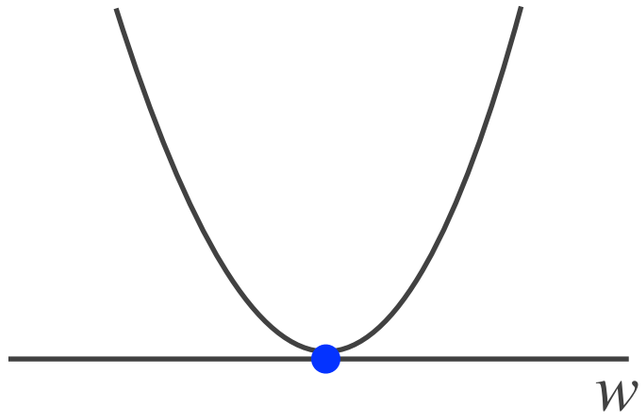
Calculus!

Compute the derivative of $L(y, f_W(x))$ with respect to W and set it to 0.

Example: Optimization

- Goal: Find w that minimizes the following loss function

$$L(w) = w^2$$



Method 1: Compute the derivative and set it to 0

Gradient $\dfrac{dL(w)}{dw} = \dfrac{dw^2}{dw} = 2w \quad = 0 \quad \Rightarrow \quad w = 0$

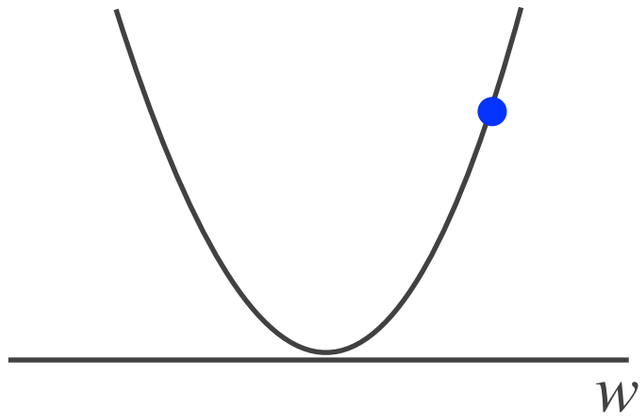
In our case, we have $L(y, f_W(x))$. Not simple to compute the gradient with respect to W.

Even when we have a way to compute the gradient and manage to set it to zero, we might not be able to get a closed-form solution for it.

Example: Optimization

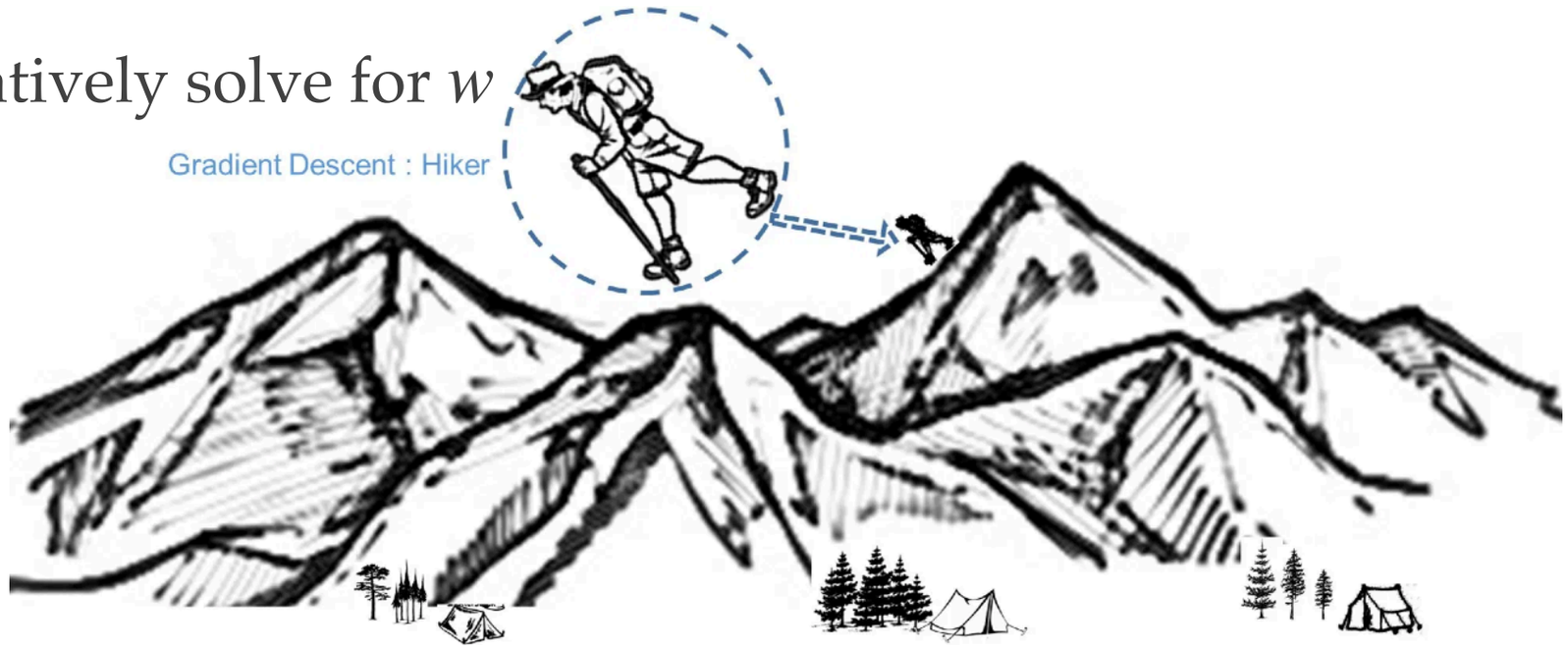
- Goal: Find w that minimizes the following loss function

$$L(w) = w^2$$



- Randomly start somewhere
- Gradually move to the minimum of the function

Method 2: Iteratively solve for w

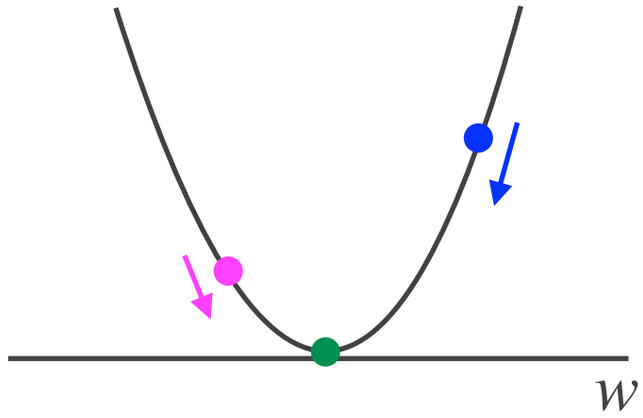


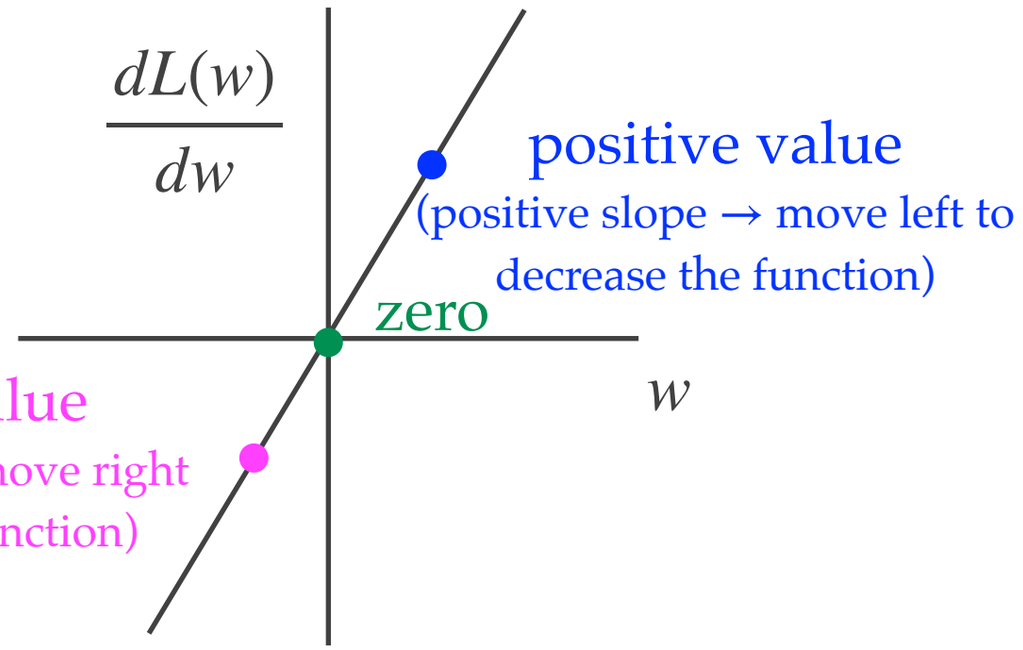
Gradient Descent for Logistic Regression Simplified – Step by Step Visual Guide

Example: Optimization

- Goal: Find w that minimizes the following loss function

$$L(w) = w^2$$





Method 2: Iteratively solve for w

1. Guess an initial solution w_0 and pick α

For $k = 0,1,2,...$

2. Compute $\frac{dL(w)}{dw}|_{w=w_{(k)}} = 2w|_{w=w_{(k)}} = 2w_{(k)}$

Iteration k

The negative of the gradient $-\frac{dL(w)}{dw}$ tells us the direction we should move to decrease the function

3. Compute a new solution $w_{(k+1)} := w_{(k)} - \alpha \frac{dL(w)}{dw}|_{w=w_{(k)}} = w_{(k)} - \alpha * 2w_{(k)}$

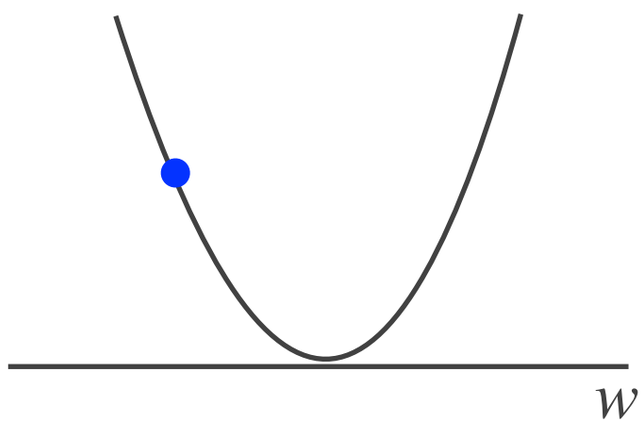
step size / learning rate α indicates how far we want to move

4. Repeat step 2 and 3 until converge

Example: Optimization

- Goal: Find w that minimizes the following loss function

$$L(w) = w^2$$



$$w_{(0)} = -2, \alpha = 0.5$$

$$w_{(1)} = w_{(0)} - \alpha * 2w_{(0)} = -2 - 0.5 * 2 * -2 = 0$$

$$w_{(2)} = w_{(1)} - \alpha * 2w_{(1)} = 0$$

$$w_{(3)} = w_{(2)} - \alpha * 2w_{(2)} = 0$$

Method 2: Iteratively solve for w

1. Guess an initial solution w_0 and pick α

For $k = 0,1,2,...$

2. Compute $\frac{dL(w)}{dw}|_{w=w_{(k)}} = 2w|_{w=w_{(k)}} = 2w_{(k)}$

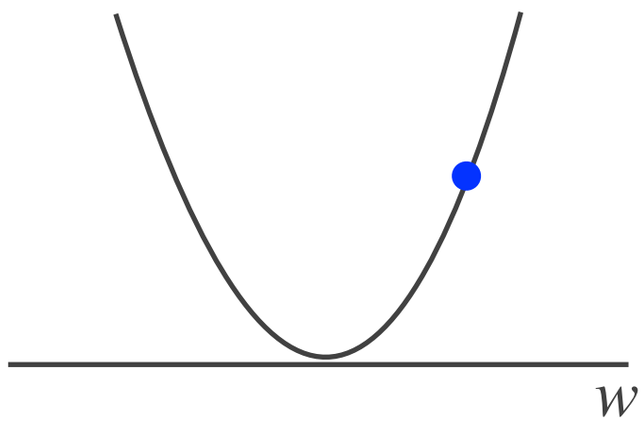
3. Compute a new solution $w_{(k+1)} := w_{(k)} - \alpha \frac{dL(w)}{dw}|_{w=w_{(k)}} = w_{(k)} - \alpha * 2w_{(k)}$

4. Repeat step 2 and 3 until converge

Example: Optimization

- Goal: Find w that minimizes the following loss function

$$L(w) = w^2$$



$$w_{(0)} = 2, \alpha = 0.5$$

$$w_{(1)} = w_{(0)} - \alpha * 2w_{(0)} = 2 - 0.5 * 2 * 2 = 0$$

$$w_{(2)} = w_{(1)} - \alpha * 2w_{(1)} = 0$$

$$w_{(3)} = w_{(2)} - \alpha * 2w_{(2)} = 0$$

Method 2: Iteratively solve for w

1. Guess an initial solution w_0 and pick α

For $k = 0,1,2,...$

2. Compute $\frac{dL(w)}{dw}|_{w=w_{(k)}} = 2w|_{w=w_{(k)}} = 2w_{(k)}$

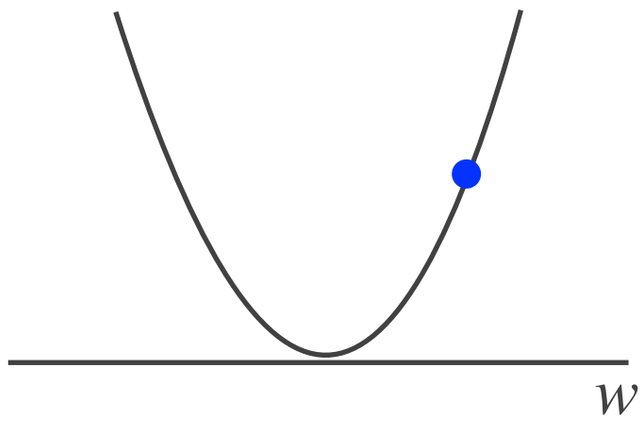
3. Compute a new solution $w_{(k+1)} := w_{(k)} - \alpha \frac{dL(w)}{dw}|_{w=w_{(k)}} = w_{(k)} - \alpha * 2w_{(k)}$

4. Repeat step 2 and 3 until converge

Example: Optimization

- Goal: Find w that minimizes the following loss function

$$L(w) = w^2$$



$$w_{(0)} = 2, \alpha = 0.25$$

$$w_{(1)} = w_{(0)} - \alpha * 2w_{(0)} = 2 - 0.25 * 2 * 2 = 1$$

$$w_{(2)} = w_{(1)} - \alpha * 2w_{(1)} = 1 - 0.25 * 2 * 1 = 0.5$$

$$w_{(3)} = w_{(2)} - \alpha * 2w_{(2)} = 0.5 - 0.25 * 2 * 0.5 = 0.25$$

$$w_{(4)} = w_{(3)} - \alpha * 2w_{(3)} = 0.125$$

$$w_{(5)} = w_{(4)} - \alpha * 2w_{(4)} = 0.0625$$

Method 2: Iteratively solve for w

1. Guess an initial solution w_0 and pick α

For $k = 0,1,2,...$

2. Compute $\frac{dL(w)}{dw}|_{w=w_{(k)}} = 2w|_{w=w_{(k)}} = 2w_{(k)}$

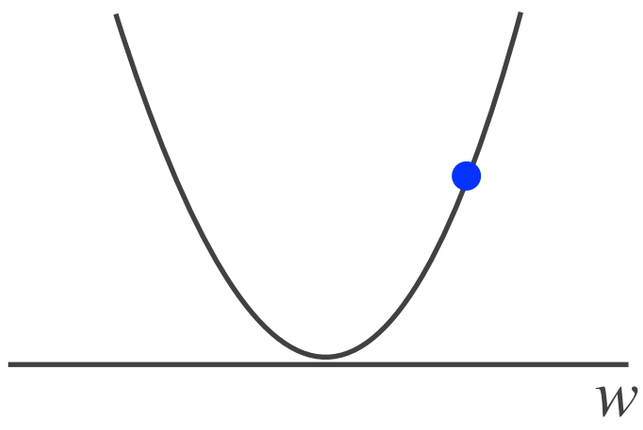
3. Compute a new solution $w_{(k+1)} := w_{(k)} - \alpha \frac{dL(w)}{dw}|_{w=w_{(k)}} = w_{(k)} - \alpha * 2w_{(k)}$

4. Repeat step 2 and 3 until converge

Example: Optimization

- Goal: Find w that minimizes the following loss function

$$L(w) = w^2$$



$$w_{(0)} = 2, \alpha = 1$$

$$w_{(1)} = w_{(0)} - \alpha * 2w_{(0)} = 2 - 1 * 2 * 2 = -2$$

$$w_{(2)} = w_{(1)} - \alpha * 2w_{(1)} = -2 - 1 * 2 * -2 = 2$$

$$w_{(3)} = w_{(2)} - \alpha * 2w_{(2)} = -2$$

$$w_{(4)} = w_{(3)} - \alpha * 2w_{(3)} = 2$$

$$w_{(5)} = w_{(4)} - \alpha * 2w_{(4)} = -2$$

Method 2: Iteratively solve for w

1. Guess an initial solution w_0 and pick α

For $k = 0,1,2,...$

2. Compute $\frac{dL(w)}{dw}|_{w=w_{(k)}} = 2w|_{w=w_{(k)}} = 2w_{(k)}$

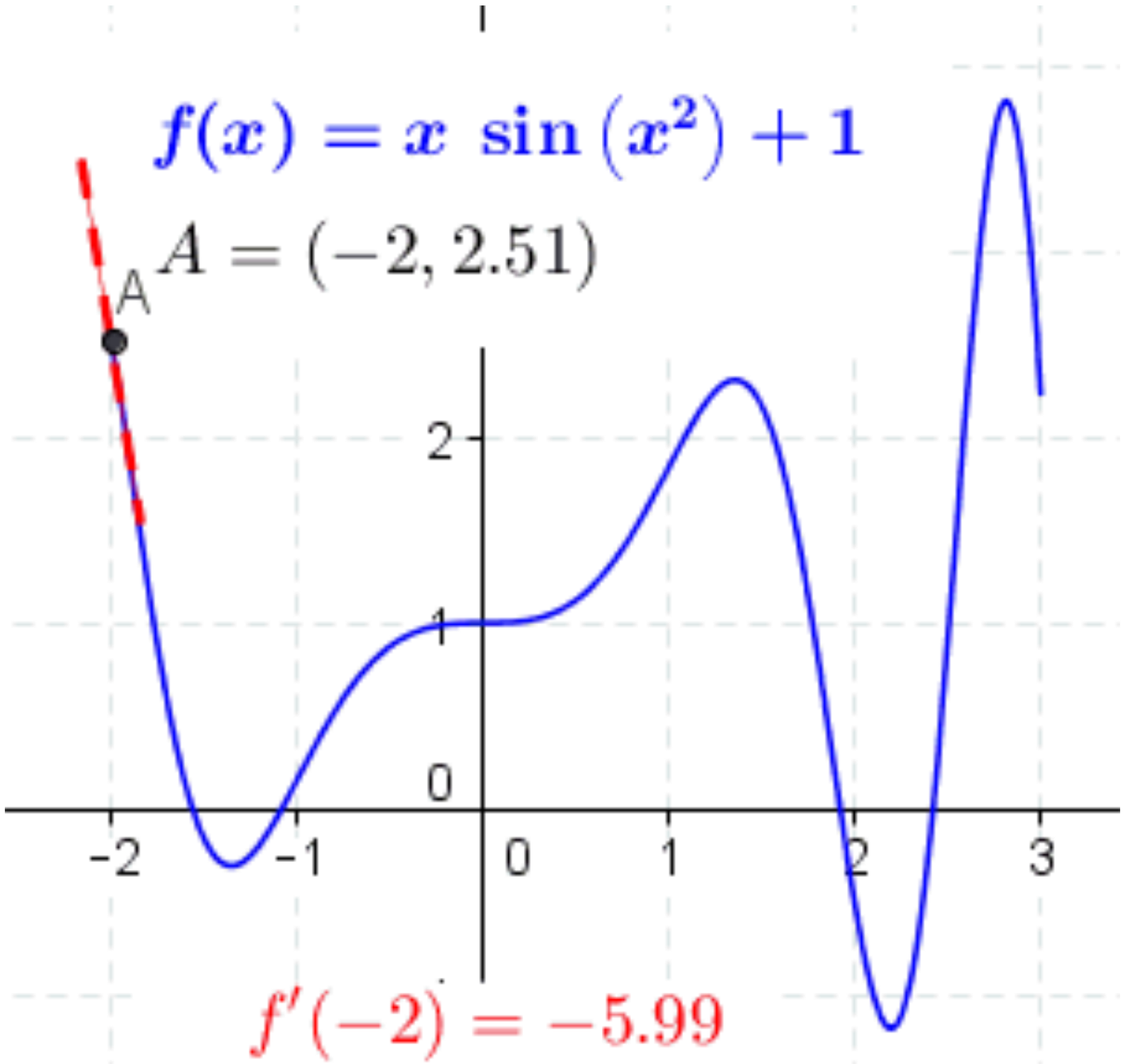
3. Compute a new solution $w_{(k+1)} := w_{(k)} - \alpha \frac{dL(w)}{dw}|_{w=w_{(k)}} = w_{(k)} - \alpha * 2w_{(k)}$

4. Repeat step 2 and 3 until converge

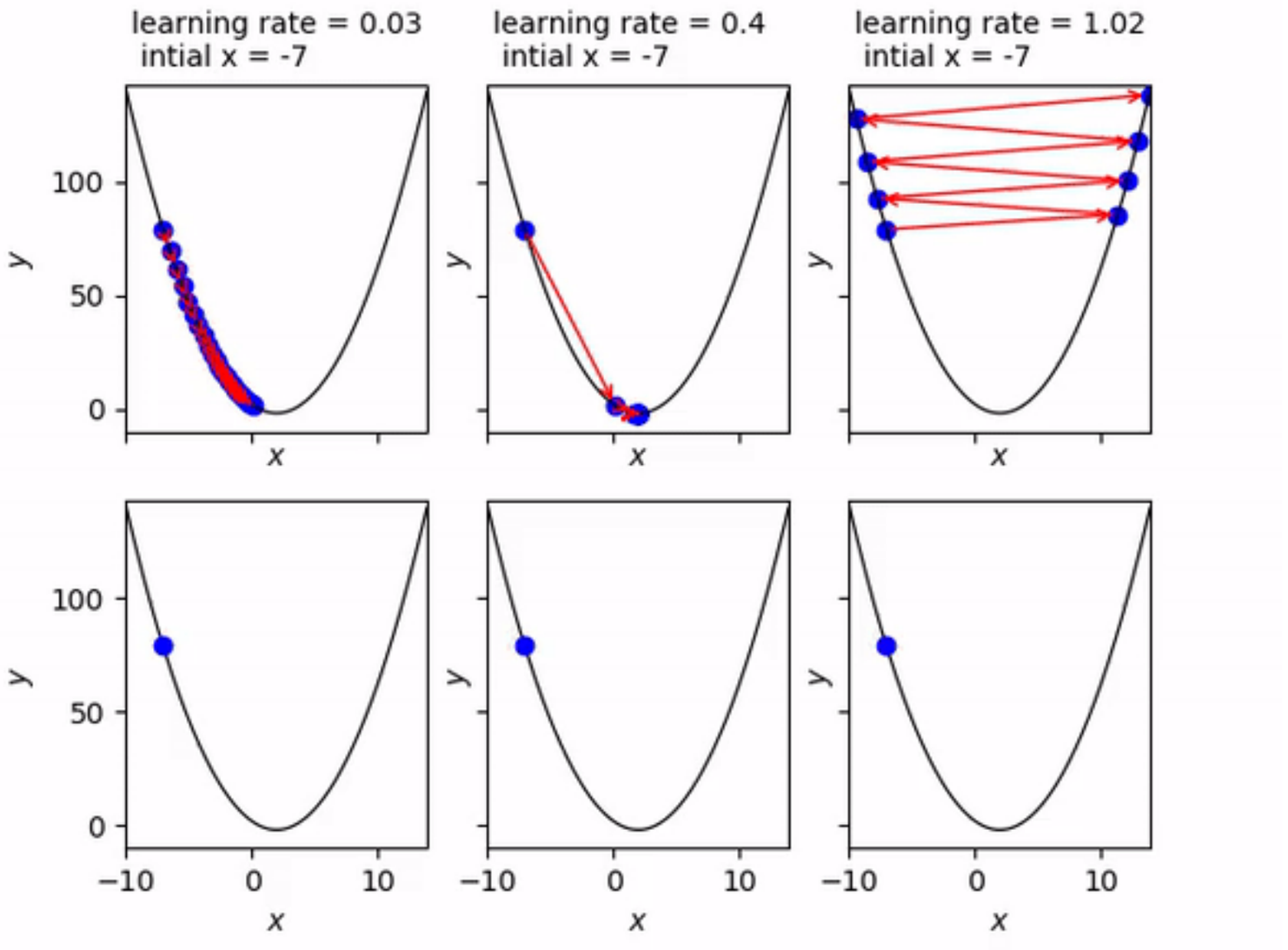
α: learning rate

keywords: local minimum/maximum, global minimum/maximum

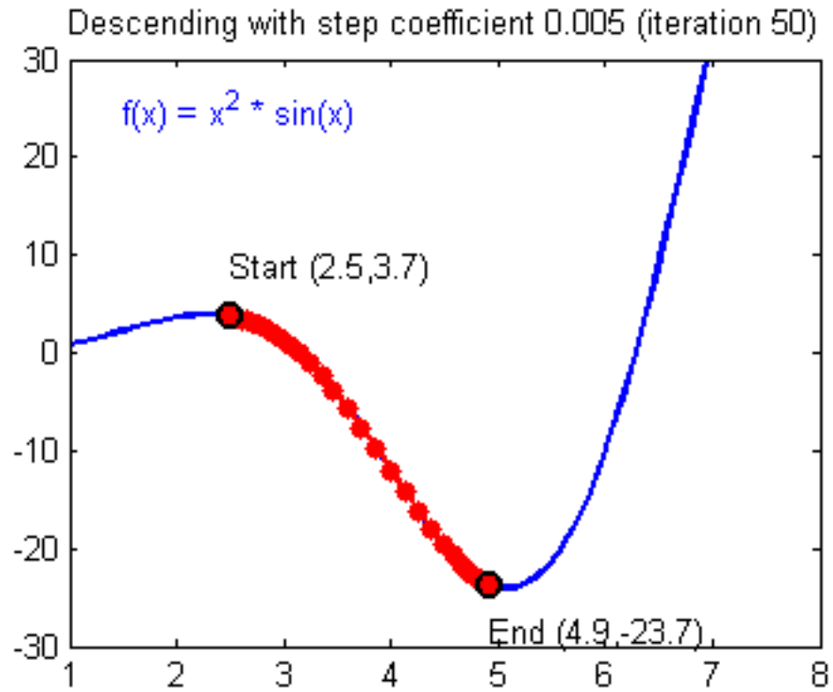
Example: Gradient



Optimization: Learning Rate



Fixed learning rate

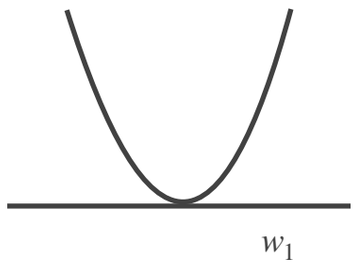


Adaptive learning rate

Gradient

1-dimensional case

$$L(w_1) = w_1^2$$



Derivative of $L(w_1)$

$$\frac{dL(w_1)}{dw_1} = \frac{dw_1^2}{dw_1} = 2w_1$$

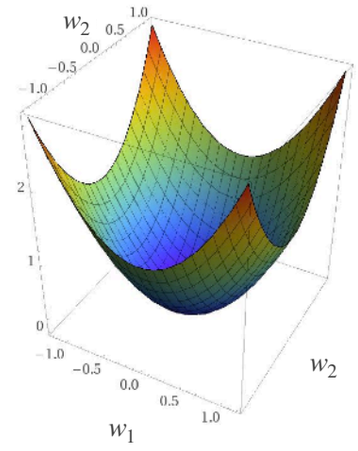
Weight update

$$w_{1,(k+1)} := w_{1,(k)} - \alpha \frac{dL(w_1)}{dw_1}\Big|_{w_1=w_{1,(k)}}$$

$$= w_{1,(k)} - 2w_{1,(k)}$$

2-dimensional case

$$L(w_1, w_2) = w_1^2 + w_2^2$$



Partial derivatives of $L(w_1, w_2)$

$$\frac{\partial L(w_1, w_2)}{\partial w_1} = \frac{\partial}{\partial w_1}(w_1^2 + w_2^2) = 2w_1$$

$$\frac{\partial L(w_1, w_2)}{\partial w_2} = \frac{\partial}{\partial w_2}(w_1^2 + w_2^2) = 2w_2$$

Gradient $\nabla_w L(w_1, w_2)$

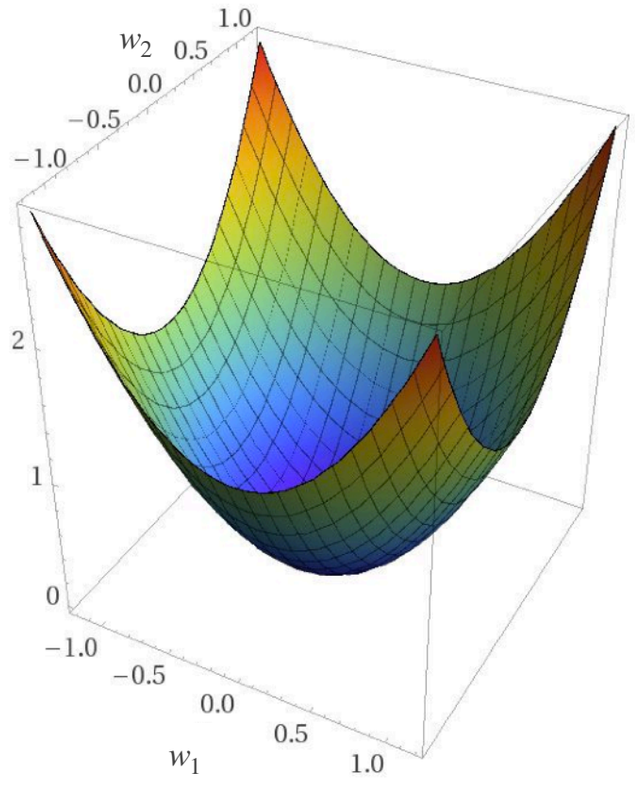
Weight update

$$\begin{bmatrix} w_{1,(k+1)} \\ w_{2,(k+1)} \end{bmatrix} = \begin{bmatrix} w_{1,(k)} \\ w_{2,(k)} \end{bmatrix} - \alpha \begin{bmatrix} \frac{\partial L(w_1, w_2)}{\partial w_1}\Big|_{w_1=w_{1,(k)}} \\ \frac{\partial L(w_1, w_2)}{\partial w_2}\Big|_{w_2=w_{2,(k)}} \end{bmatrix}$$

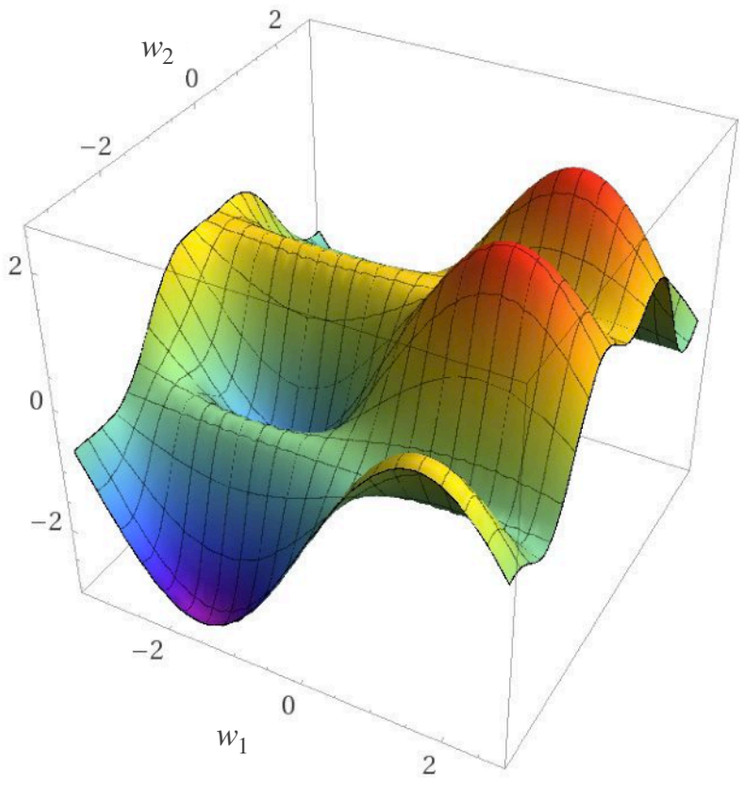
$$= \begin{bmatrix} w_{1,(k)} \\ w_{2,(k)} \end{bmatrix} - \alpha \begin{bmatrix} 2w_{1,(k)} \\ 2w_{2,(k)} \end{bmatrix}$$

Optimization

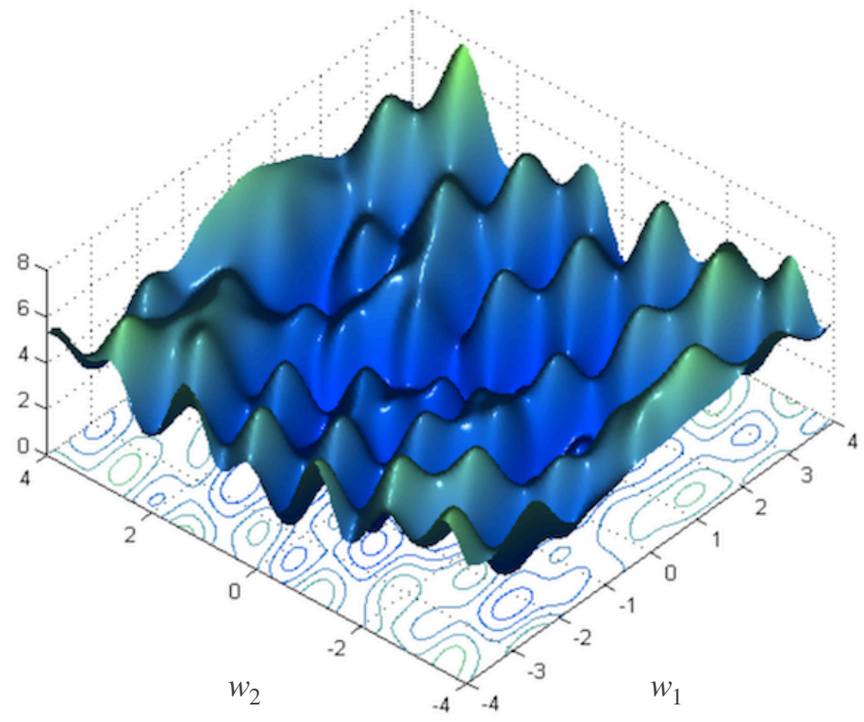
Real Life



No local minima problem (global minimum = local minimum)



Could get stuck in local minima



@#Q#@%!!!!!!

Furthermore, how do we compute a complicated loss function such as the one in our example $L(y, f_W(x))$?

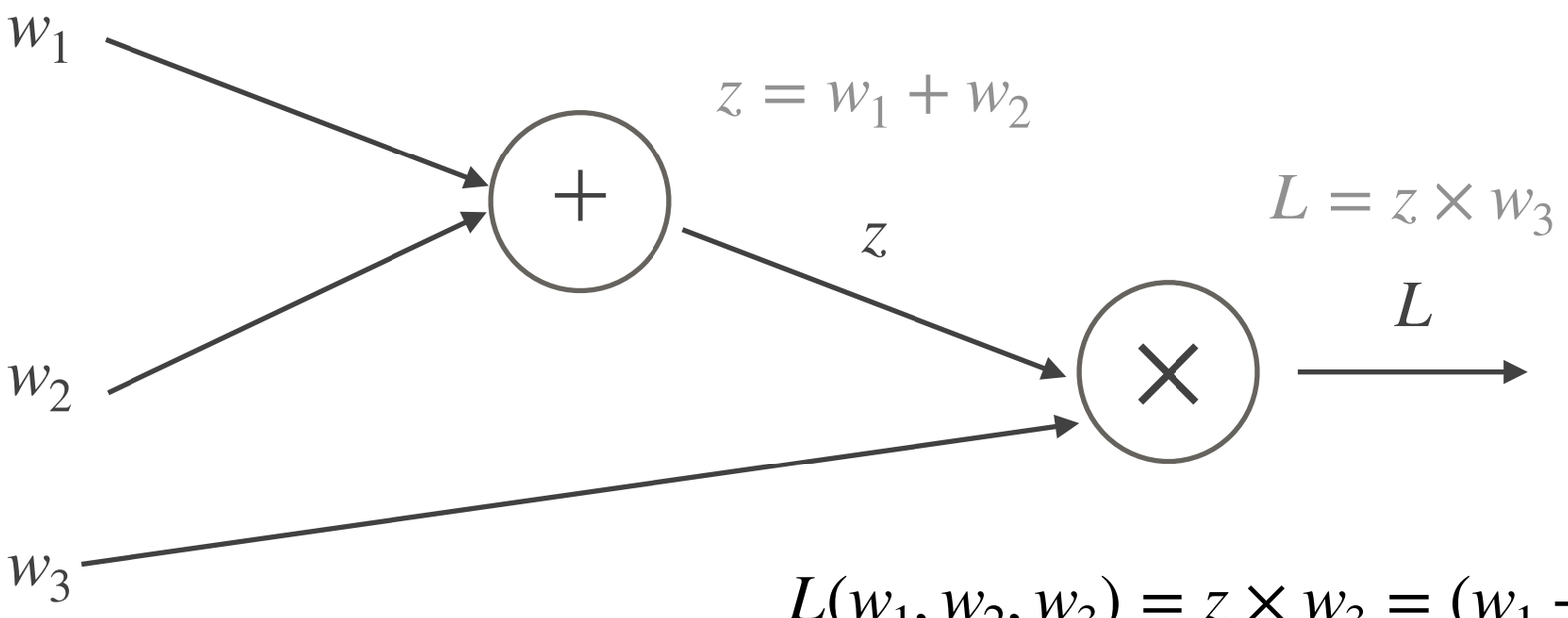
Backpropagation!

O'Reilly Media

Yoshua Bengio

Backpropagation

Computation graph



$$L(w_1, w_2, w_3) = z \times w_3 = (w_1 + w_2) \times w_3$$

Exercise: Compute the gradient vector

Use chain rule

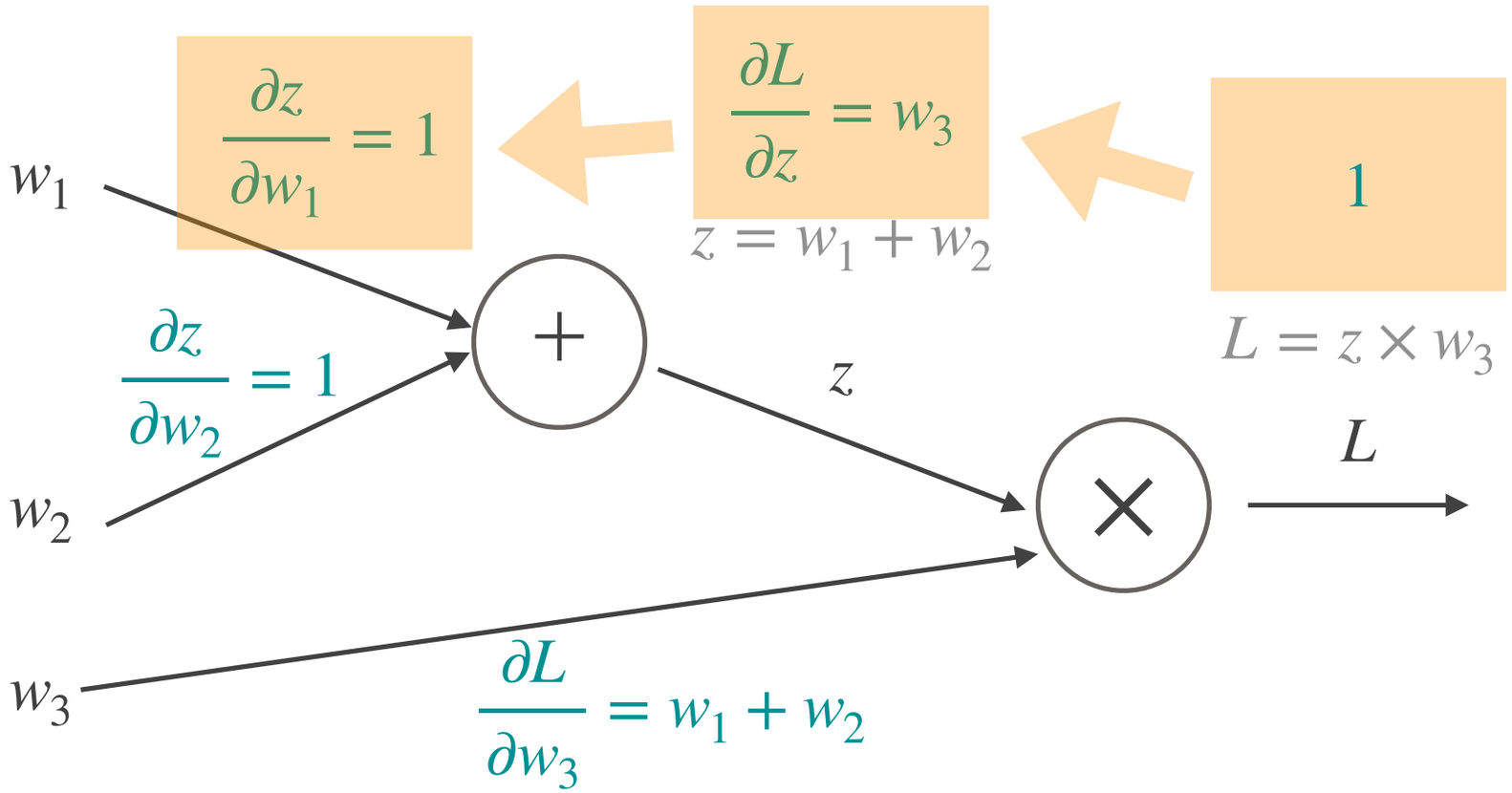
$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial w_1} = \frac{\partial w_3 z}{\partial z}\frac{\partial (w_1 + w_2)}{\partial w_1} = w_3(1 + 0) = w_3$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial w_2} = \frac{\partial w_3 z}{\partial z}\frac{\partial (w_1 + w_2)}{\partial w_2} = w_3(0 + 1) = w_3$$

$$\frac{\partial L}{\partial w_3} = \frac{\partial z w_3}{\partial w_3} = z = w_1 + w_2$$

$$\nabla_w L(w_1, w_2, w_3) = \begin{bmatrix} w_3 \\ w_3 \\ w_1 + w_2 \end{bmatrix}$$

Backpropagate



Exercise: Compute the gradient vector

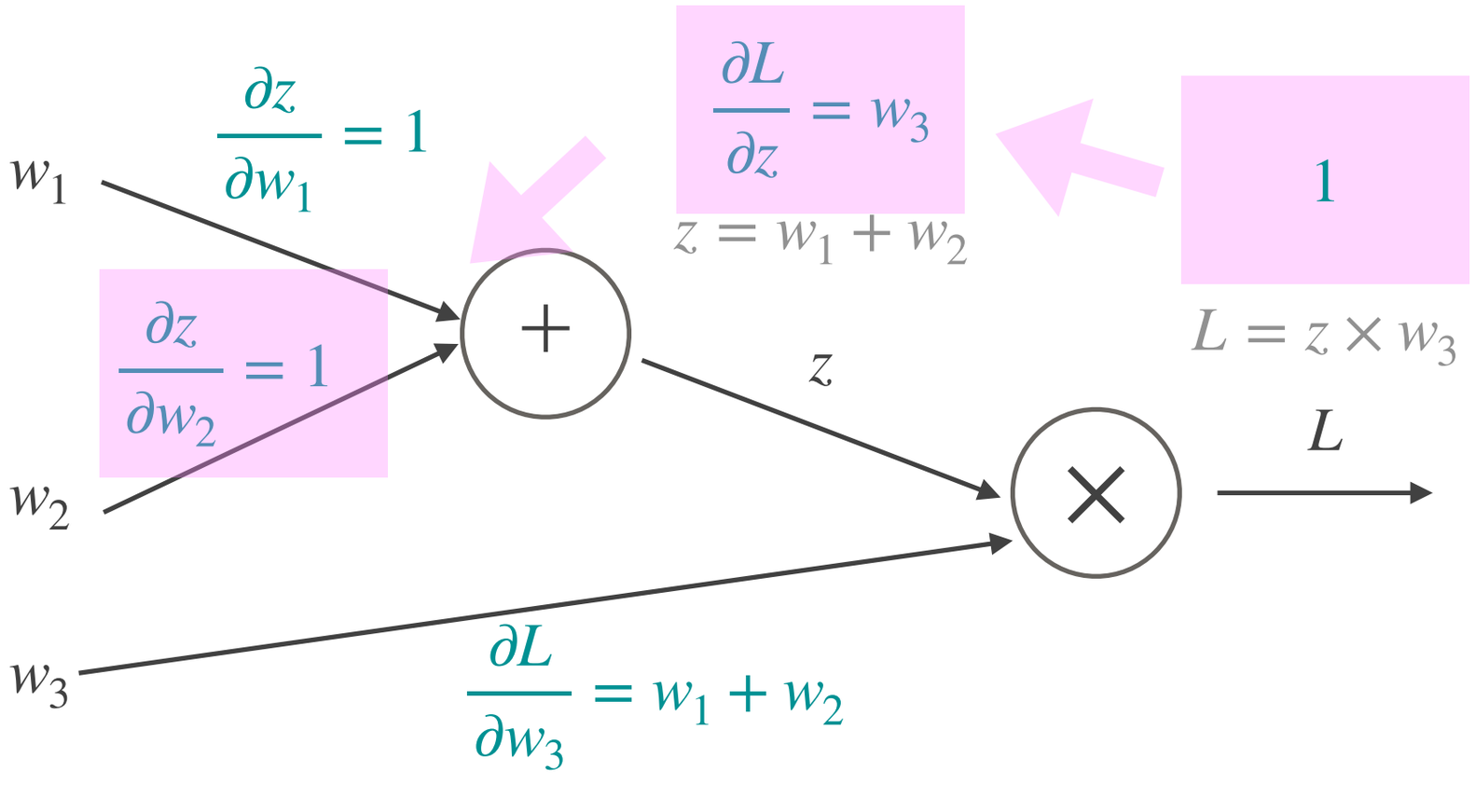
$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial w_1} = \frac{\partial w_3 z}{\partial z}\frac{\partial (w_1 + w_2)}{\partial w_1} = w_3(1+0) = w_3$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial w_2} = \frac{\partial w_3 z}{\partial z}\frac{\partial (w_1 + w_2)}{\partial w_2} = w_3(0+1) = w_3$$

$$\frac{\partial L}{\partial w_3} = \frac{\partial z w_3}{\partial w_3} = z = w_1 + w_2$$

$$\nabla_w L(w_1, w_2, w_3) = \begin{bmatrix} w_3 \\ w_3 \\ w_1 + w_2 \end{bmatrix}$$

Backpropagate



Exercise: Compute the gradient vector

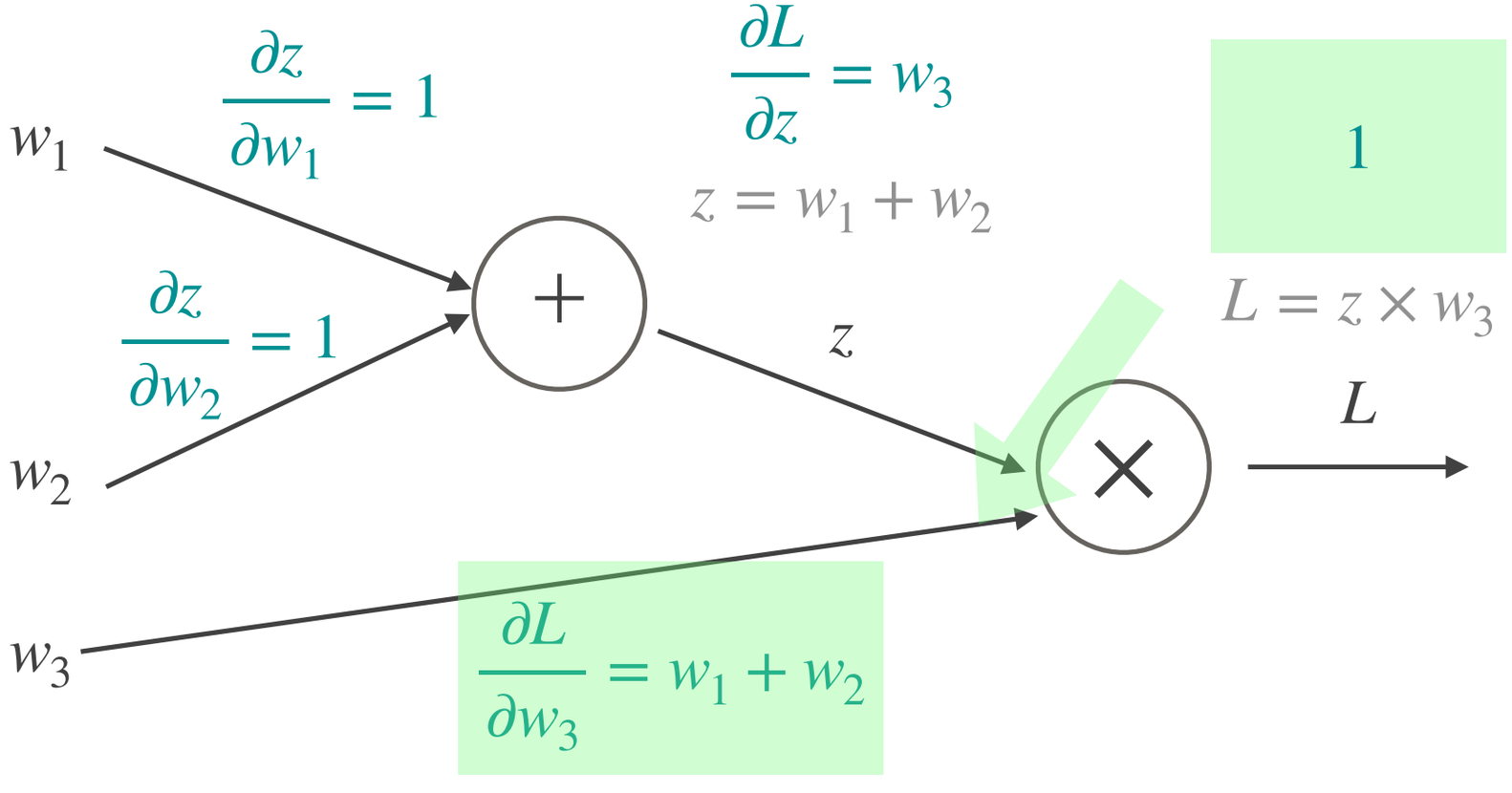
$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial w_1} = \frac{\partial w_3 z}{\partial z}\frac{\partial (w_1 + w_2)}{\partial w_1} = w_3(1+0) = w_3$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial w_2} = \frac{\partial w_3 z}{\partial z}\frac{\partial (w_1 + w_2)}{\partial w_2} = w_3(0+1) = w_3$$

$$\frac{\partial L}{\partial w_3} = \frac{\partial z w_3}{\partial w_3} = z = w_1 + w_2$$

$$\nabla_w L(w_1, w_2, w_3) = \begin{bmatrix} w_3 \\ w_3 \\ w_1 + w_2 \end{bmatrix}$$

Backpropagate



Exercise: Compute the gradient vector

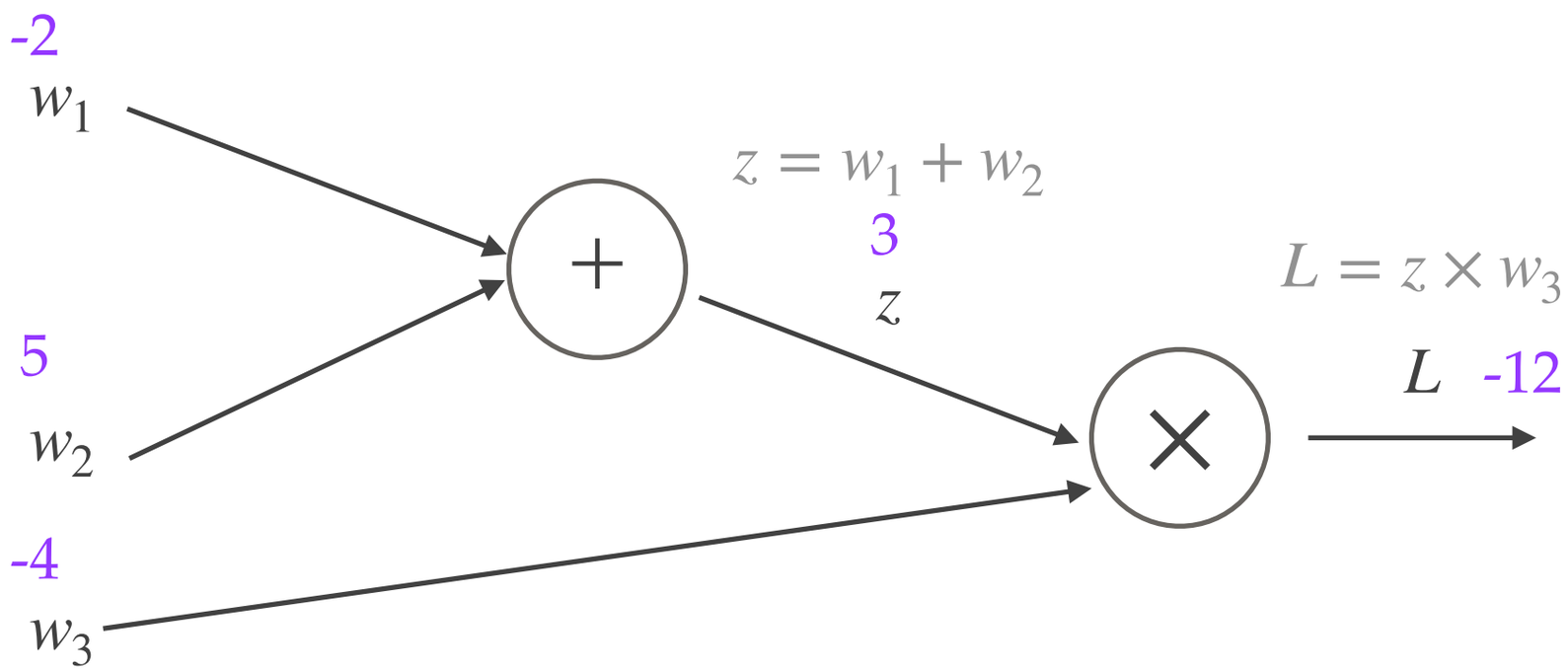
$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial w_1} = \frac{\partial w_3 z}{\partial z}\frac{\partial (w_1 + w_2)}{\partial w_1} = w_3(1+0) = w_3$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial w_2} = \frac{\partial w_3 z}{\partial z}\frac{\partial (w_1 + w_2)}{\partial w_2} = w_3(0+1) = w_3$$

$$\frac{\partial L}{\partial w_3} = \frac{\partial z w_3}{\partial w_3} = z = w_1 + w_2$$

$$\nabla_w L(w_1, w_2, w_3) = \begin{bmatrix} w_3 \\ w_3 \\ w_1 + w_2 \end{bmatrix}$$

With actual numbers

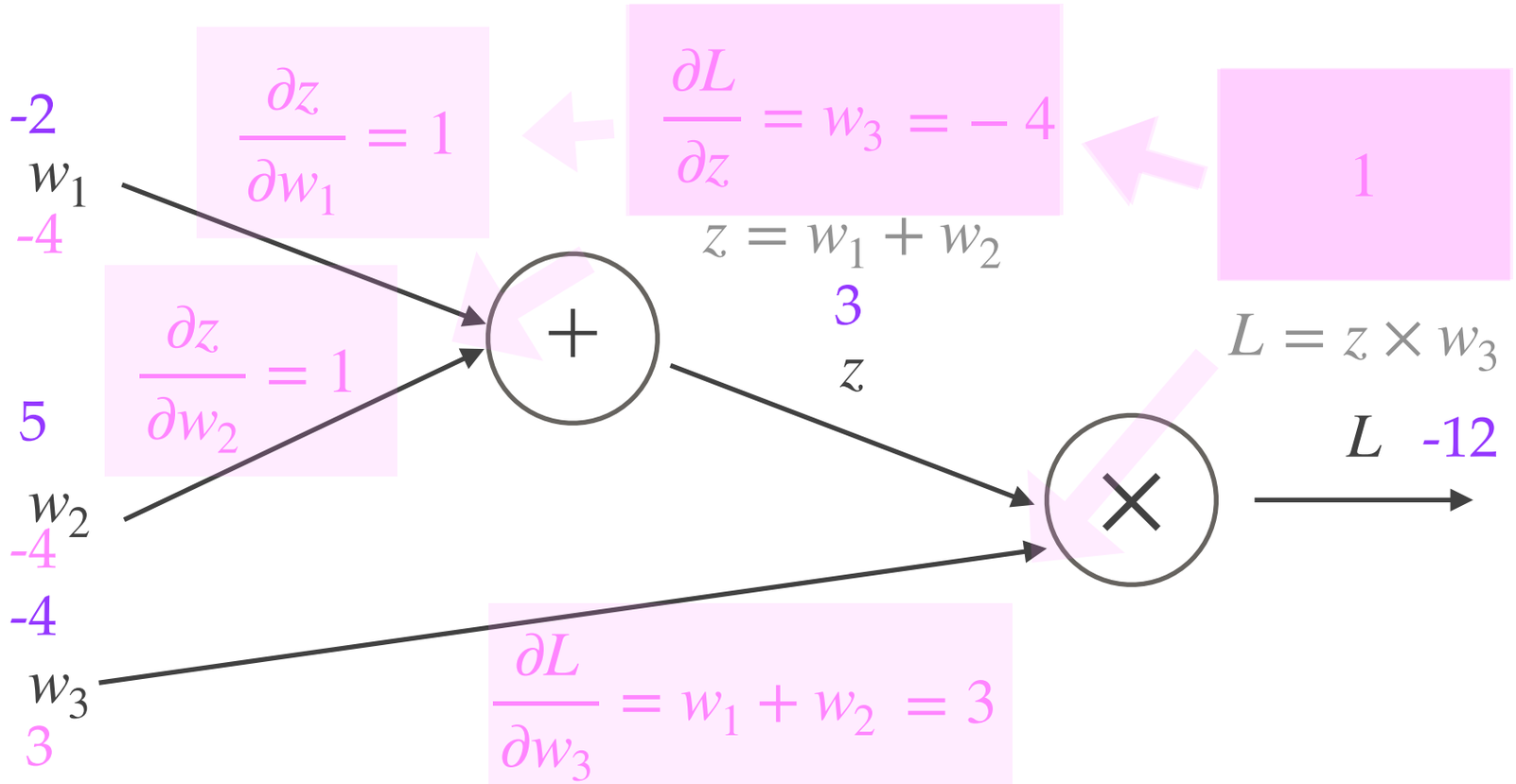


Forward pass

$z = -2 + 5 = 3$ $L = -4 \times 3 = -12$

We calculate the values of the variables in the computation graph

With actual numbers



Forward pass

$z = -2 + 5 = 3$ $L = -4 + 3 = -12$

We calculate the values of the variables in the computation graph

Backward pass

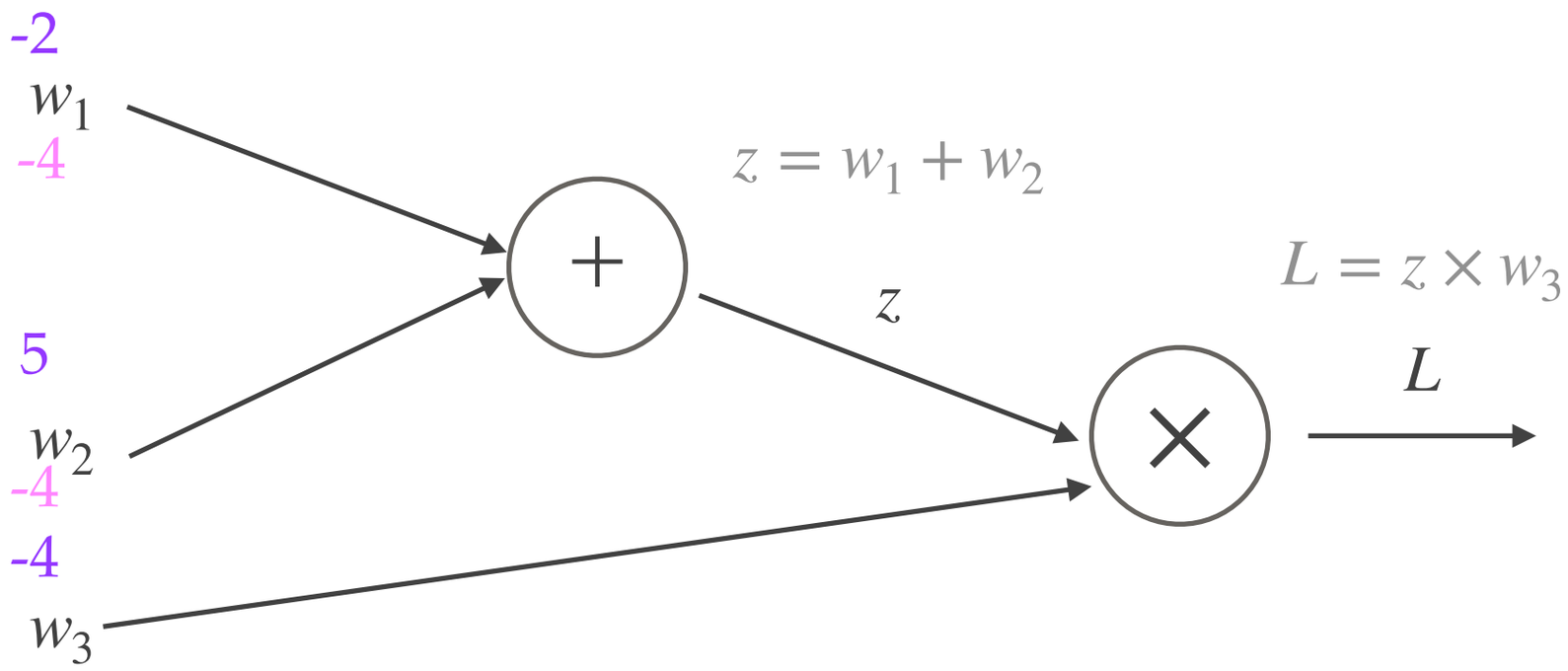
We calculate the gradients and evaluate them using the values from the forward pass

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial w_1} = -4 \times 1 = -4 \qquad \frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial w_2} = -4 \times 1 = -4 \qquad \frac{\partial L}{\partial w_3} = z = 3$$

$$\nabla_w L = \begin{bmatrix} -4 \\ -4 \\ 3 \end{bmatrix}$$

Modified from CS231n Convolutional Neural Networks for Visual Recognition

Update the weights



Given

$$w_{(0)} = \begin{bmatrix} -2 \\ 5 \\ -4 \end{bmatrix}, \quad \nabla_w L = \begin{bmatrix} -4 \\ -4 \\ 3 \end{bmatrix} \quad \text{at } w_{(0)} \text{ and } \alpha = 1 \quad \Rightarrow \quad L(w_{(0)}) = -12$$

Weight update

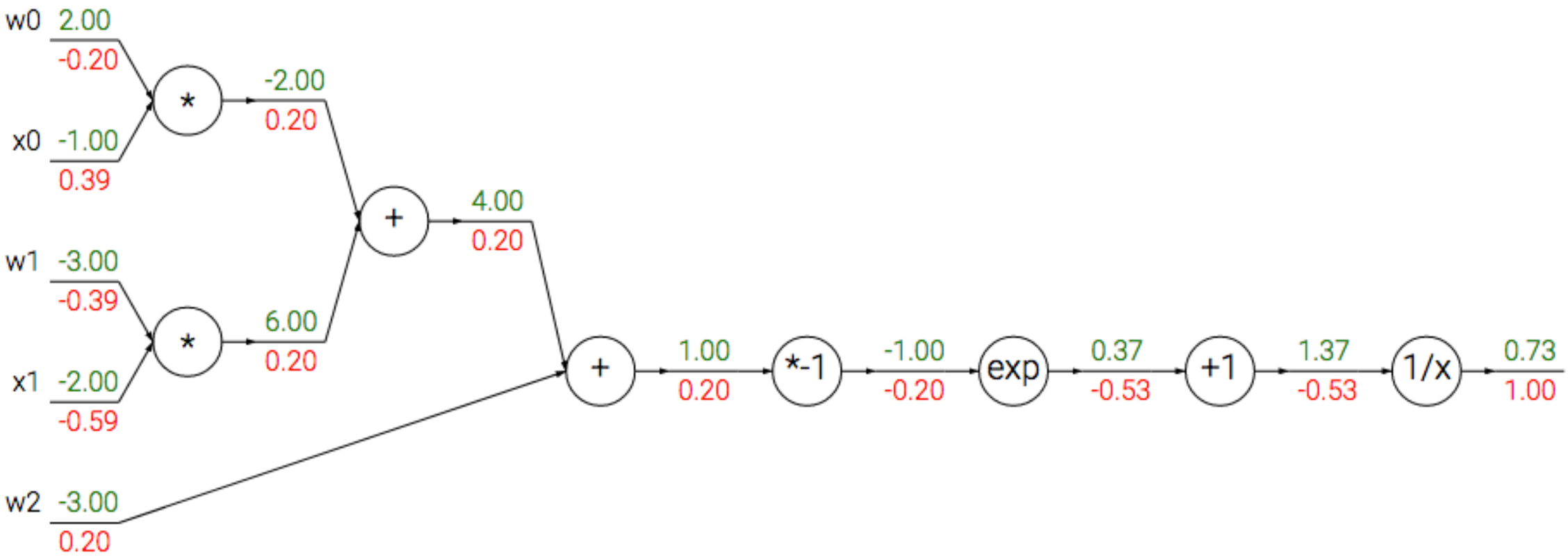
$$w_{(1)} = w_{(0)} - \alpha \nabla_w L = \begin{bmatrix} -2 \\ 5 \\ -4 \end{bmatrix} - 1 \begin{bmatrix} -4 \\ -4 \\ 3 \end{bmatrix} = \begin{bmatrix} 2 \\ 9 \\ -7 \end{bmatrix} \quad \Rightarrow \quad L(w_{(1)}) = -77$$

$$w_{(2)} = w_{(1)} - \alpha \nabla_w L = ? \quad \Rightarrow \quad L(w_{(2)}) = ?$$

Modified from CS231n Convolutional Neural Networks for Visual Recognition

Backpropagation

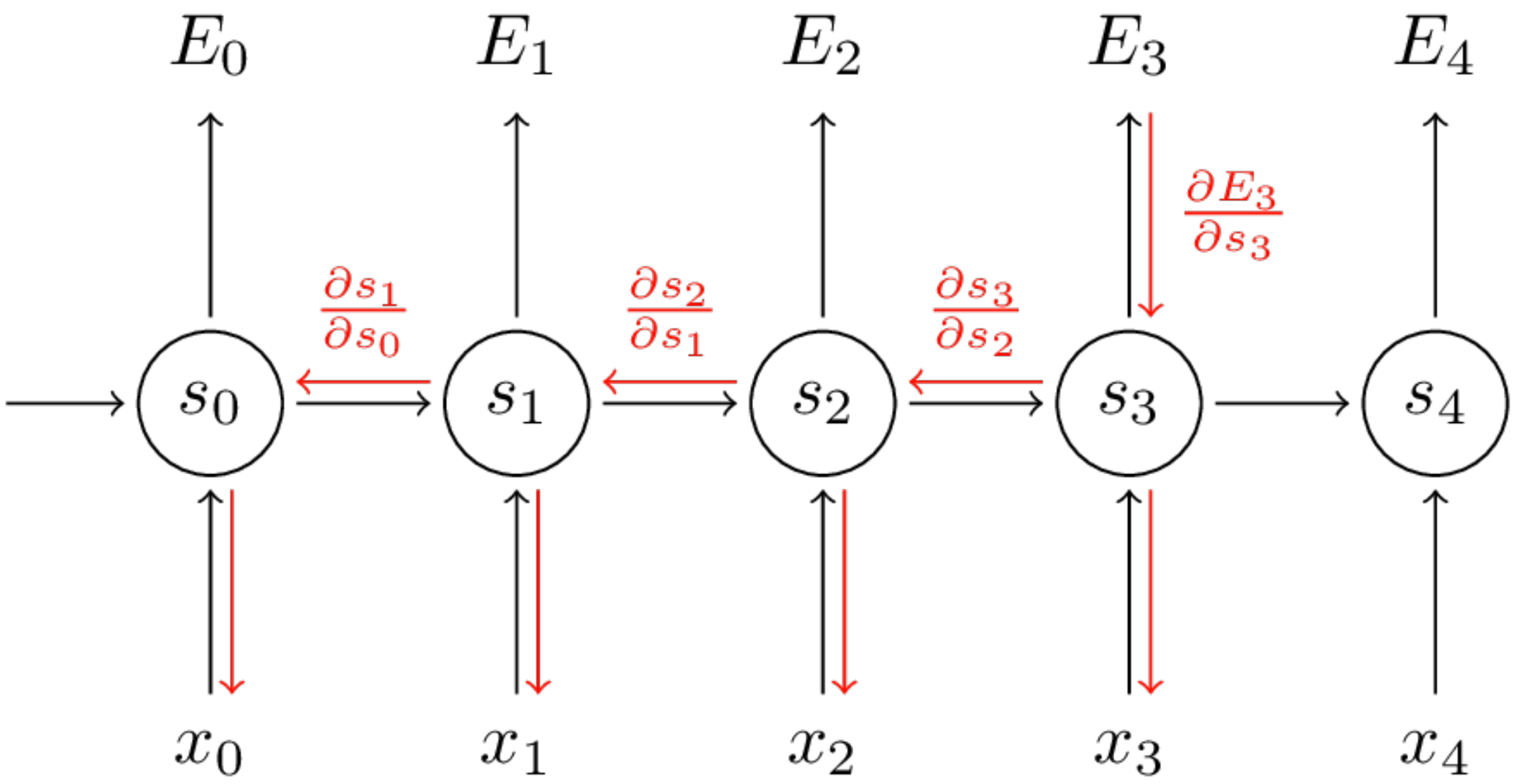
$$f(w,x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$



Example circuit for a 2D neuron with a sigmoid activation function. The inputs are [x0,x1] and the (learnable) weights of the neuron are [w0,w1,w2]. As we will see later, the neuron computes a dot product with the input and then its activation is softly squashed by the sigmoid function to be in range from 0 to 1.

Taken from CS231n Convolutional Neural Networks for Visual Recognition

Exploding/Vanishing Gradients



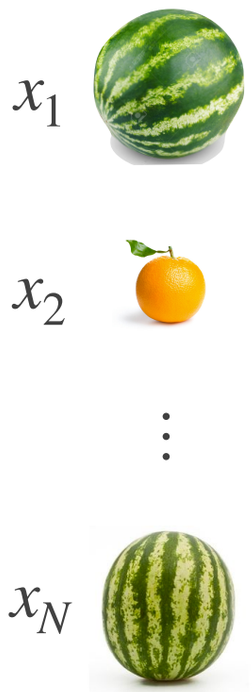
$0.05^5 = 3.125 \times 10^{-7}$ Very small

$15^5 = 759375$ Very large

Taken from Recurrent Neural Networks Tutorial, Part 3 – Backpropagation Through Time and Vanishing Gradients

Training Our Model

prepared inputs



f_W

deep learning model

predicted outputs	prepared outputs	loss values
$\hat{y}_1$	y_1	$L(y_1, \hat{y}_1)$
$\hat{y}_2$	y_2	$L(y_2, \hat{y}_2)$
$\vdots$	$\vdots$	$\vdots$
$\hat{y}_N$	y_N	$L(y_N, \hat{y}_N)$

Total loss $L(y, \hat{y}) = \sum_{i=1}^{N} L(y_i, \hat{y}_i) = \sum_{i=1}^{N} L(y_i, f_W(x_i))$

Update the weights

Compute the gradient of the total loss w.r.t. to W using backpropagation

$$\nabla_W L = \sum_{i=1}^{N} \nabla_W L(y_i, f_W(x_i))$$

Assign the new weights $W \leftarrow W - \alpha \nabla_W L$

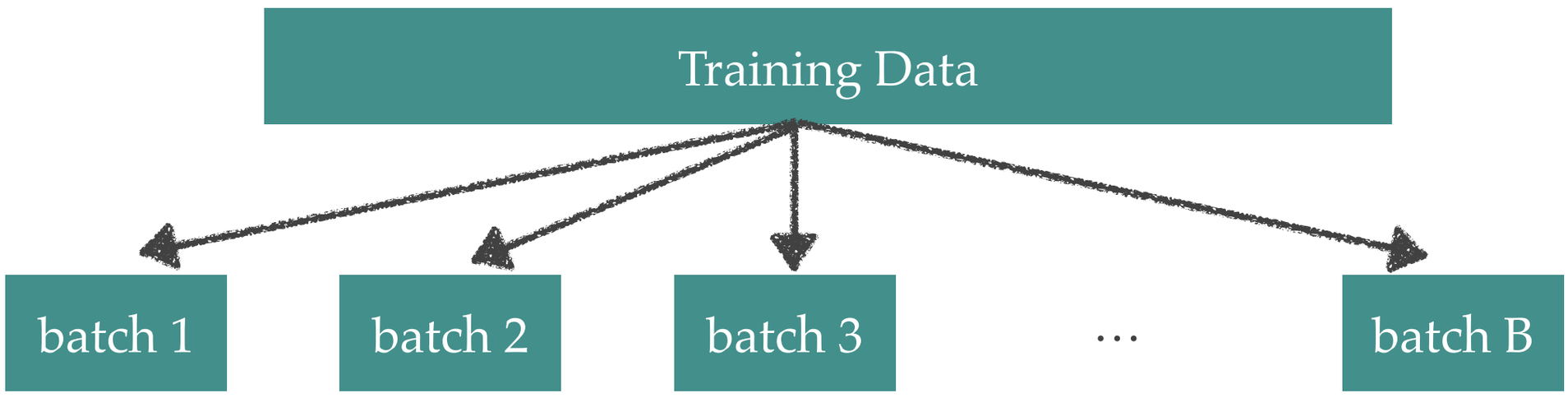
Stochastic Gradient Descent

- Every time we want to update W, we need to compute the loss functions and their gradients of all samples

$$\nabla_W L = \sum_{i=1}^{N} \nabla_W L(y_i, f_W(x_i))$$

  - For very large N, we might not be able to do it due to both the time and memory constraints 14,000,000 images?

  - For stochastic gradient descent (SGD), we update W based on subsets of samples called a mini-batch



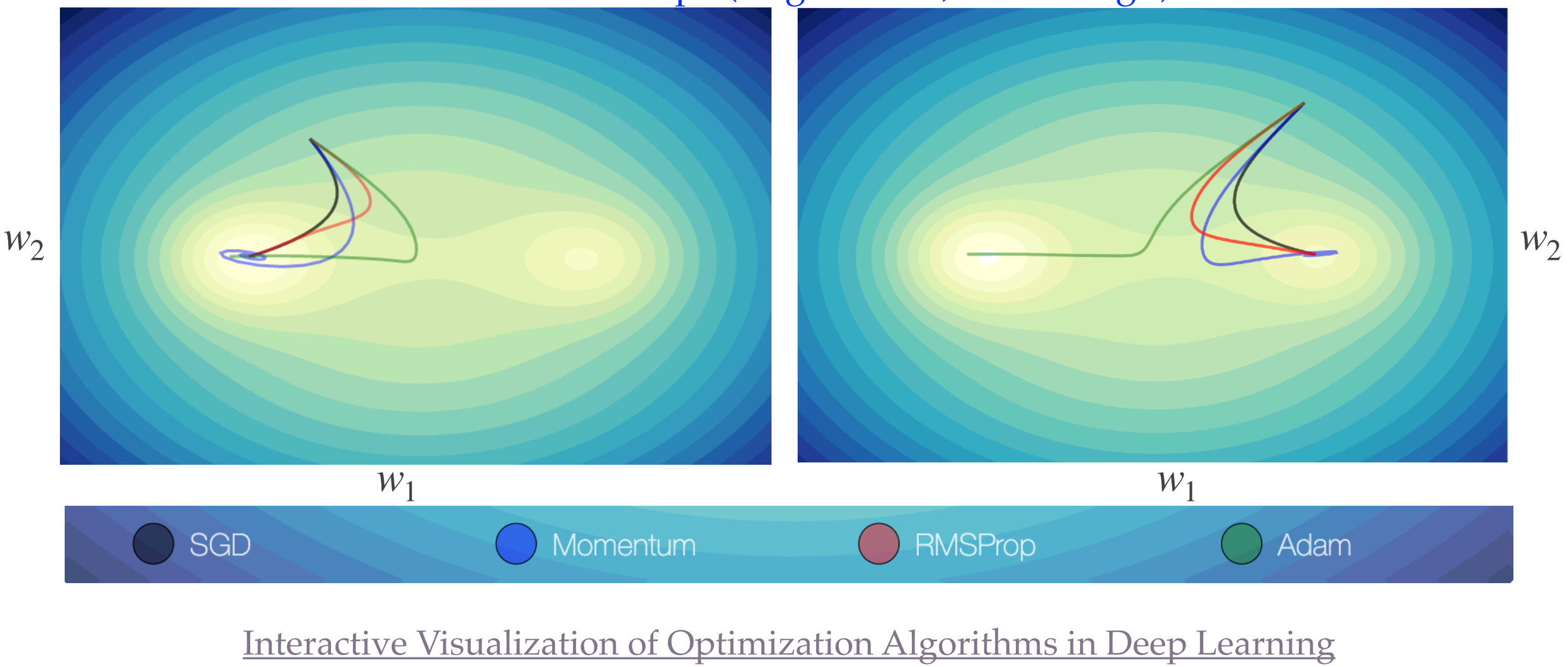
1 epoch = the model has seen the whole training data once (i.e., B batches in this case)

In this example, the weight matrix W is updated B times per epoch.

Optimizers

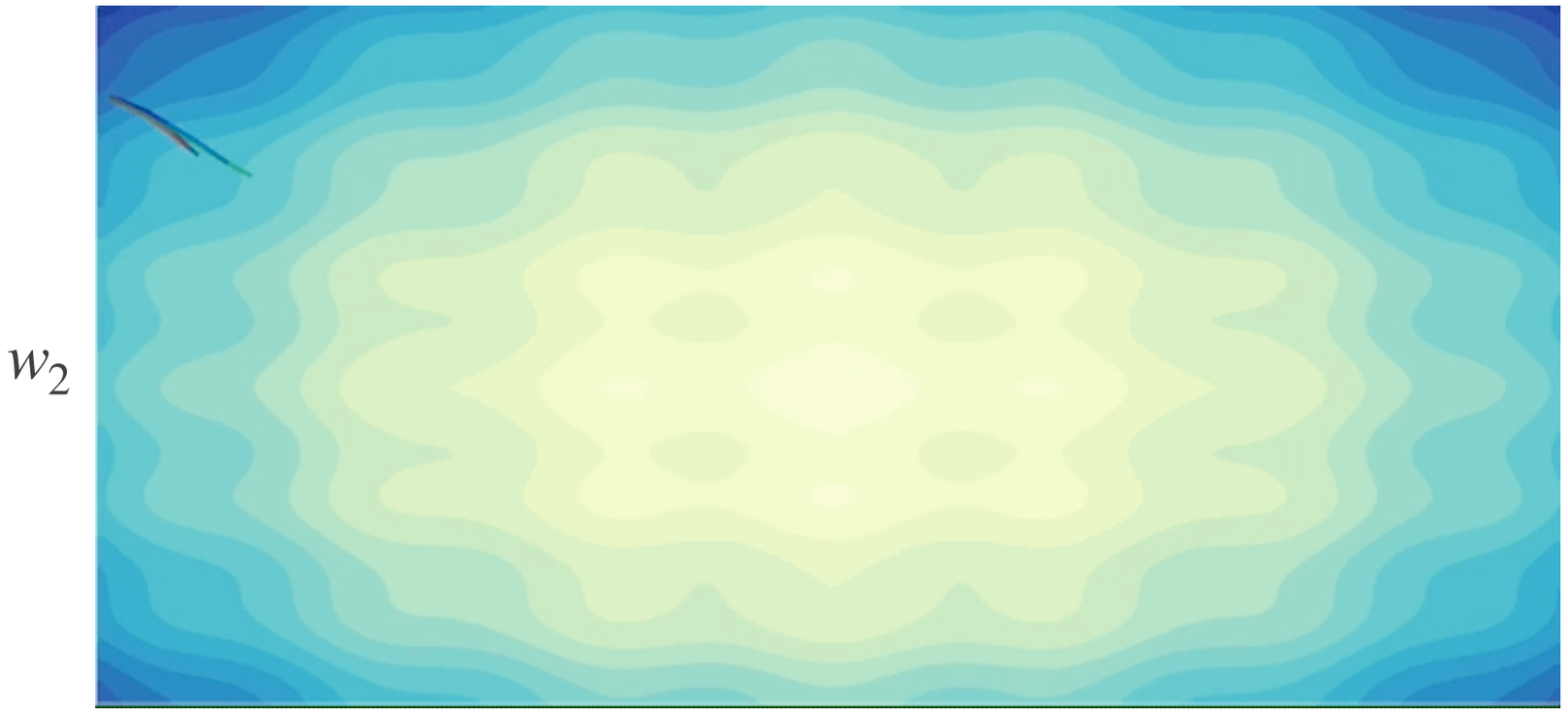
- Popular choices are
  - Stochastic gradient descent (SGD)
  - SGD with momentum
  - RMSprop
  - Adam
  - ADADELTA
  - AdaGrad

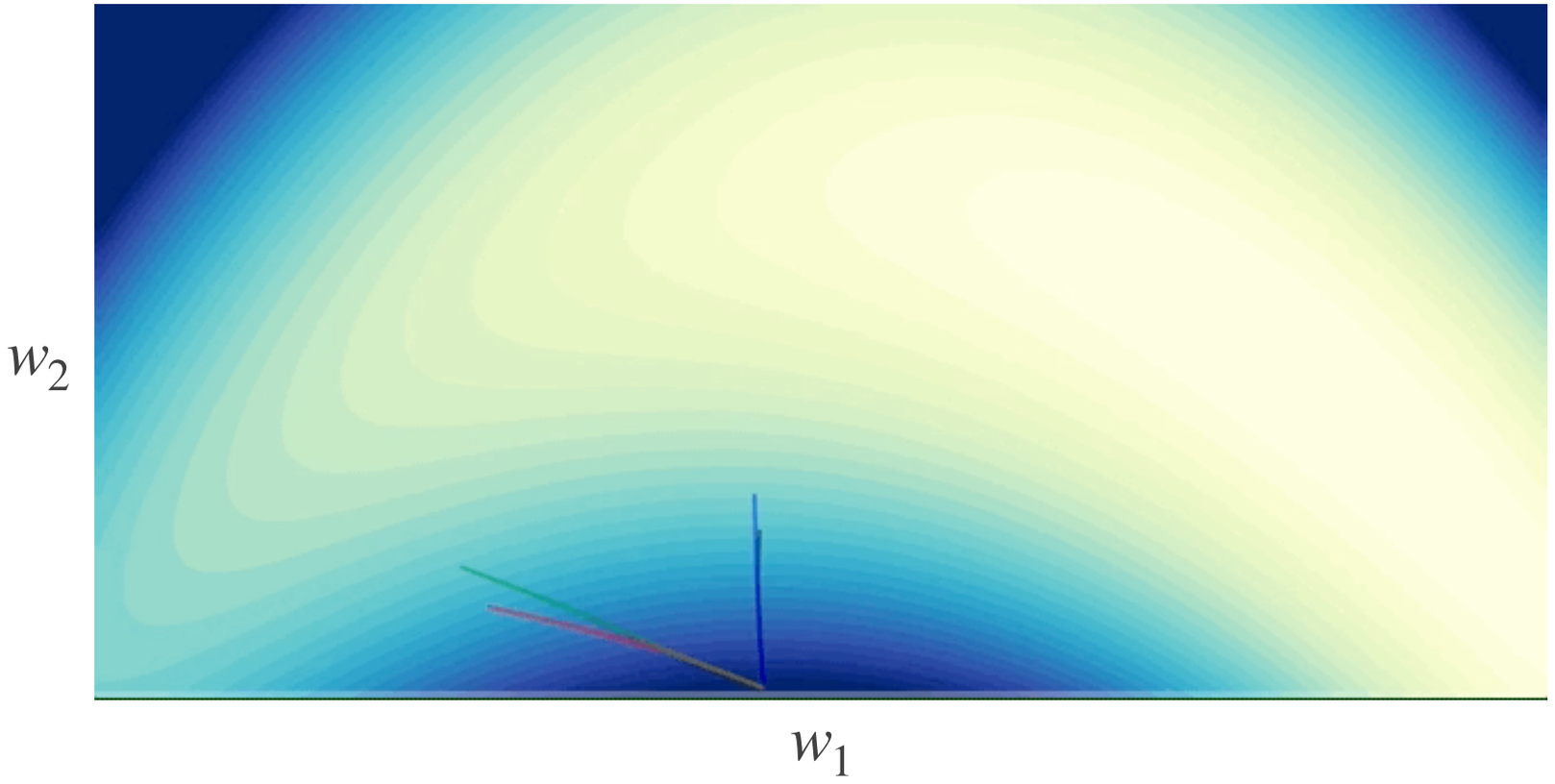
Loss landscape (bright = low, dark = high)



Interactive Visualization of Optimization Algorithms in Deep Learning

Loss landscape (bright = low, dark = high)





Interactive Visualization of Optimization Algorithms in Deep Learning