Multi-Layer Perceptrons and Back-Propagation

Xiaolong Wang

Last Class

- Linear classifiers: Logistic Regression
- Optimization: Gradient Descent
- Regularization: L2 Norm
- SoftMax

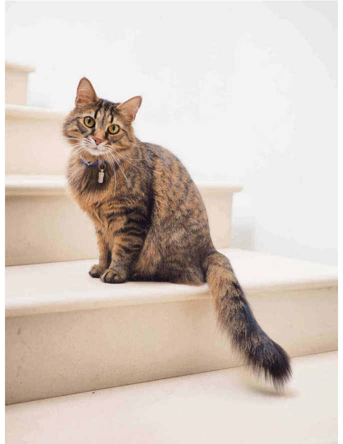
This Class

- Multi-layer Neural Networks

- Training Neural Networks with back-propagation

Multi-Layer Perceptrons

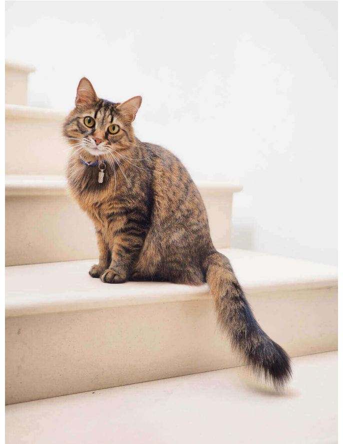
Linear Classifier using pixels as features



Scores for each class

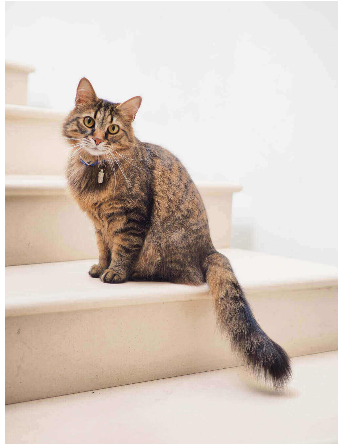
$$f(x) = Wx$$

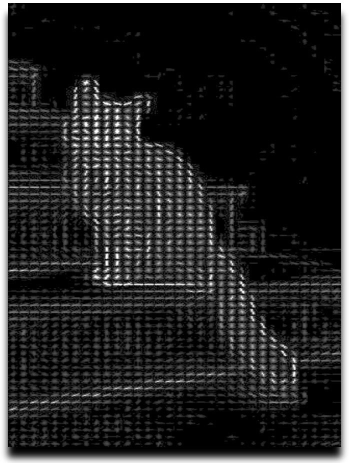
Extract feature representation



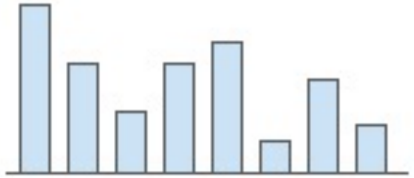
→ Feature representation → Scores for each class

Extract feature representation



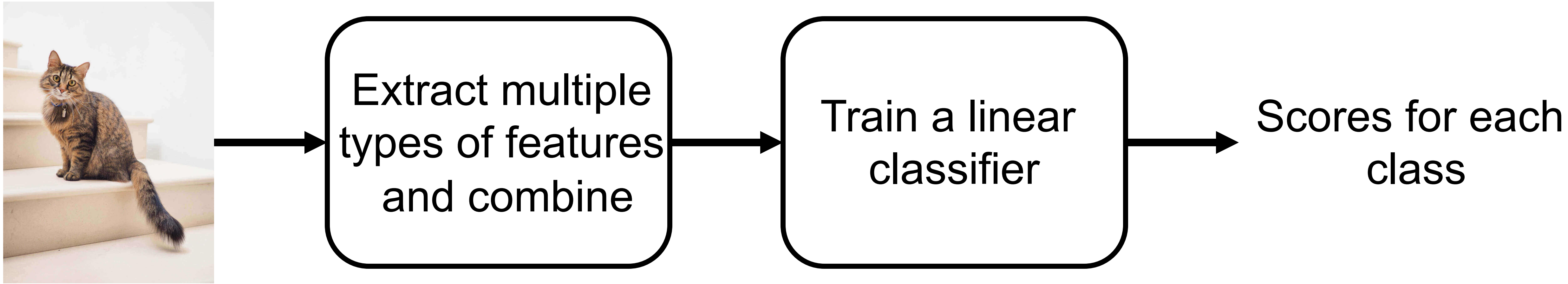


Scores for each class



Histogram of Gradients
(represent image in a much lower dimension)

Traditional Computer Vision Pipeline



Neural Networks

- Learn the features automatically instead of designing manually
- Learn the features and the classifier end-to-end together
- Using multiple layers

Multi-Layer Perceptrons

- Linear classifier: $f(x) = Wx$

- 2-Layer Neural Network: $f(x) = W_2 \, \mathrm{act}(W_1 x)$
  - 2 layers of weights W_1 and W_2
  - act is an activation function which leads to the nonlinearity

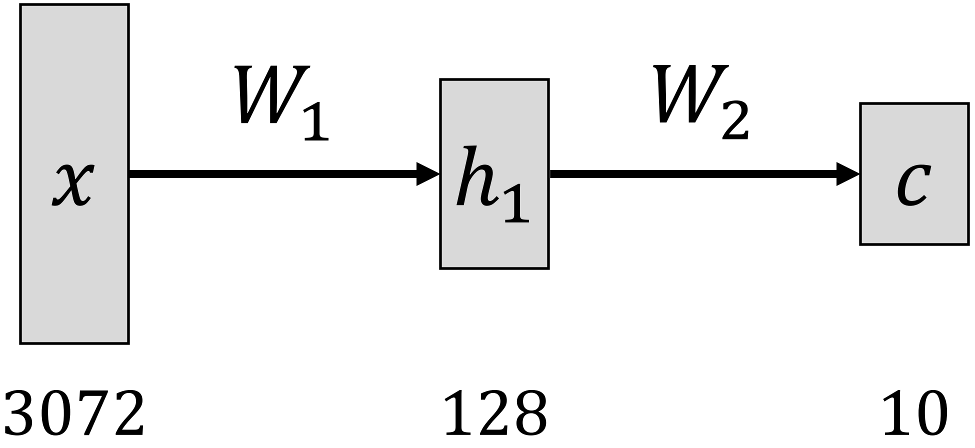
- $x \in \mathbb{R}^d, W_1 \in \mathbb{R}^{h_1 \times d}$, $W_2 \in \mathbb{R}^{c \times h_1}$
  - d is the dimension of input data, h_1 is the dimension of the hidden layer, c is the dimension of output class

Multi-Layer Perceptrons

- 2-Layer Neural Network: $f(x) = W_2 \, \mathrm{act}(W_1 x)$
- Why non-linearity between $W_1 \in \mathbb{R}^{h_1 \times d}$ and $W_2 \in \mathbb{R}^{c \times h_1}$?
  - Without activation function, we can have a simple weight $W = W_2 W_1$ instead of two sets of weights
- 3-Layer Neural Network: $f(x) = W_3 \, \mathrm{act}\big(W_2 \, \mathrm{act}(W_1 x)\big)$
  - $x \in \mathbb{R}^d, W_1 \in \mathbb{R}^{h_1 \times d}$, $W_2 \in \mathbb{R}^{h_2 \times h_1}$, $W_3 \in \mathbb{R}^{c \times h_2}$

Example: Training network for CIFAR-10

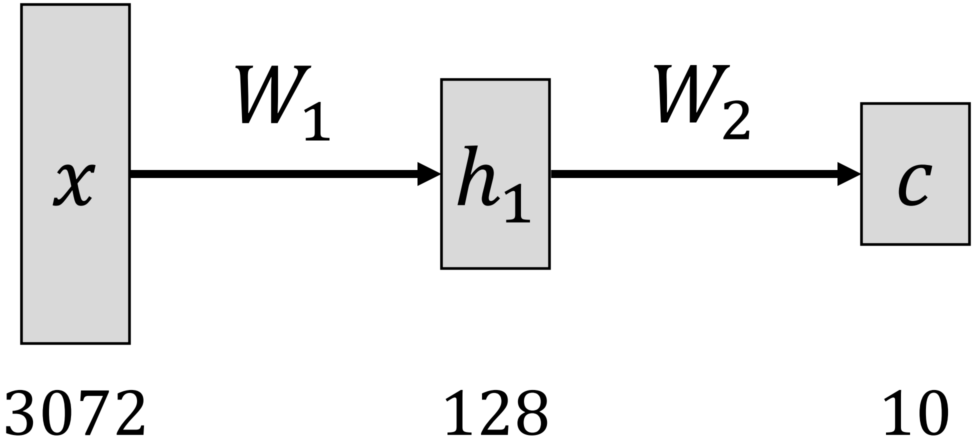
- 2-Layer Neural Network: $f(x) = W_2 \, \mathrm{act}(W_1 x)$



- $x \in \mathbb{R}^{3072}, W_1 \in \mathbb{R}^{128 \times 3072}, W_2 \in \mathbb{R}^{10 \times 128}$ $(32 \times 32 \times 3 = 3072)$

Example: Training network for CIFAR-10

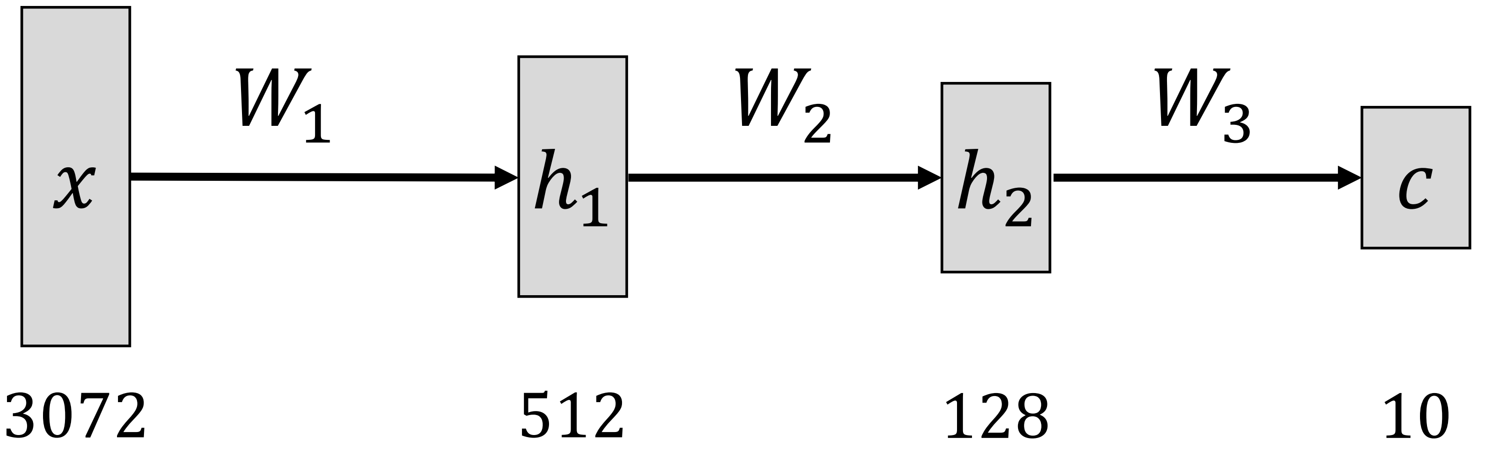
- 2-Layer Neural Network: $f(x) = W_2 \text{ act}(W_1 x)$



Learn 128 shared templates instead of 10 separate ones

Example: Training network for CIFAR-10

- 3-Layer Neural Network: $f(x) = W_3 \text{ act}\big(W_2 \text{ act}(W_1 x)\big)$

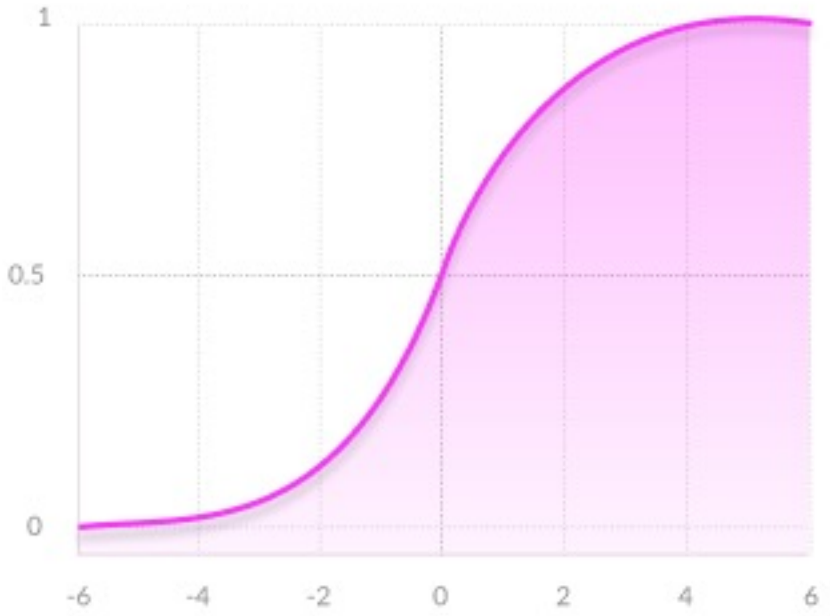


- $x \in \mathbb{R}^{3072}, W_1 \in \mathbb{R}^{512\times 3072}, W_2 \in \mathbb{R}^{128\times 512}, W_3 \in \mathbb{R}^{10\times 128}$

Activation Function

- Sigmoid function:

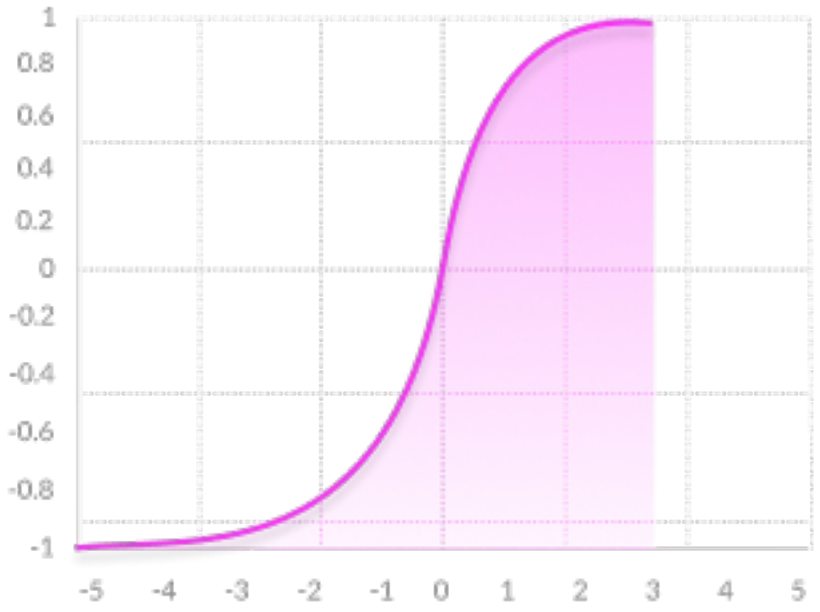
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



Activation Function

- tanh function:

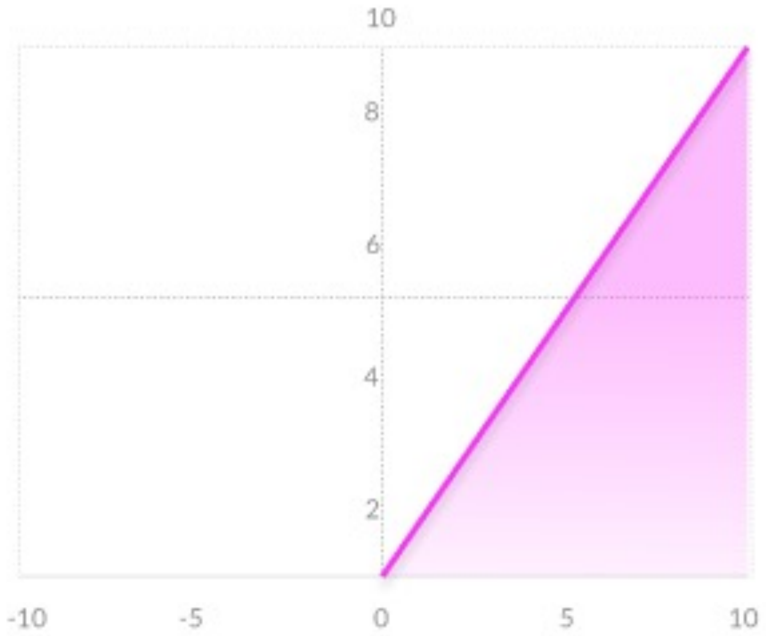
$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^{2x} - 1}{e^{2x} + 1}$$



Activation Function

- Most commonly used: ReLU function:

$$\max(0, x)$$



Activation Function

- Sigmoid function:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

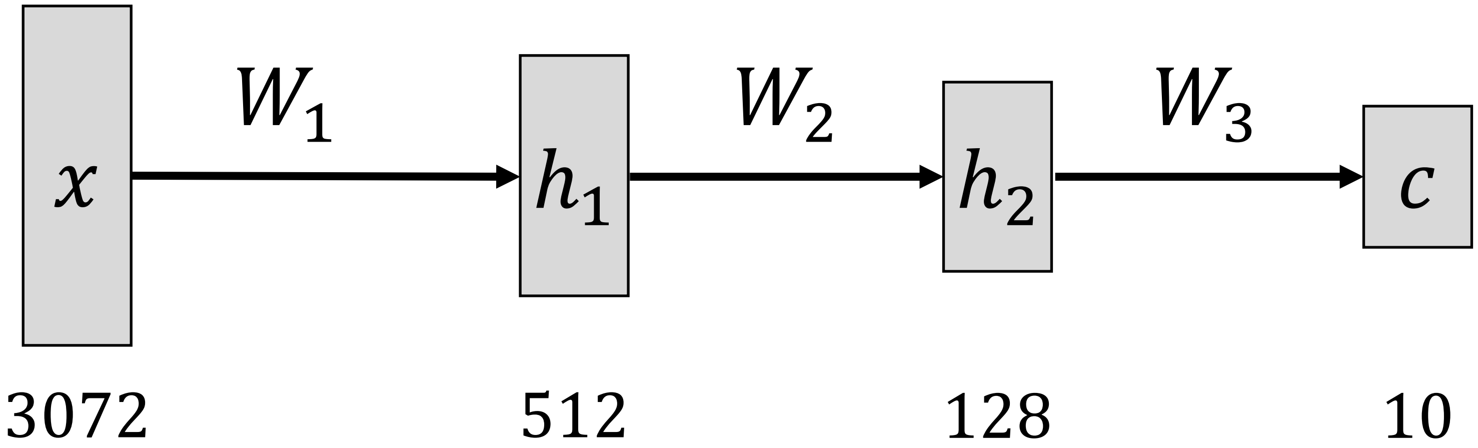
- tanh function:

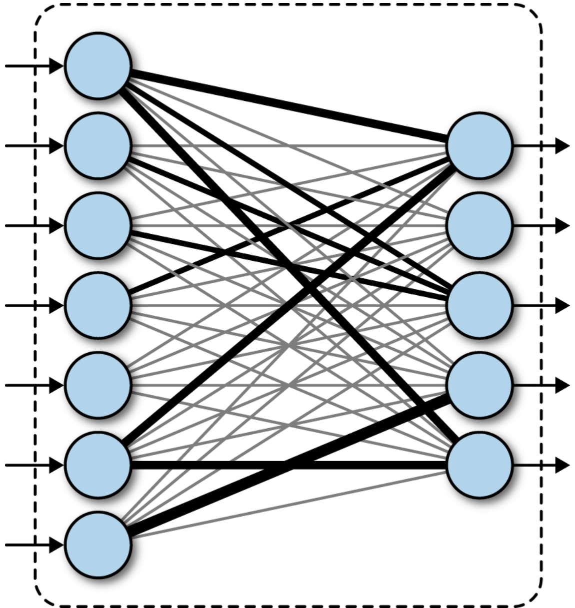
$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^{2x} - 1}{e^{2x} + 1}$$

- ReLU function:

$$\max(0, x)$$

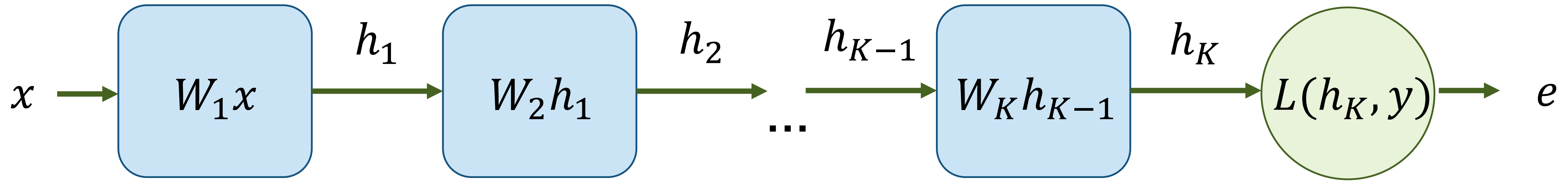
MLP = Fully Connected Network





Training MLP with Back-Propagation

The computation graph of MLP



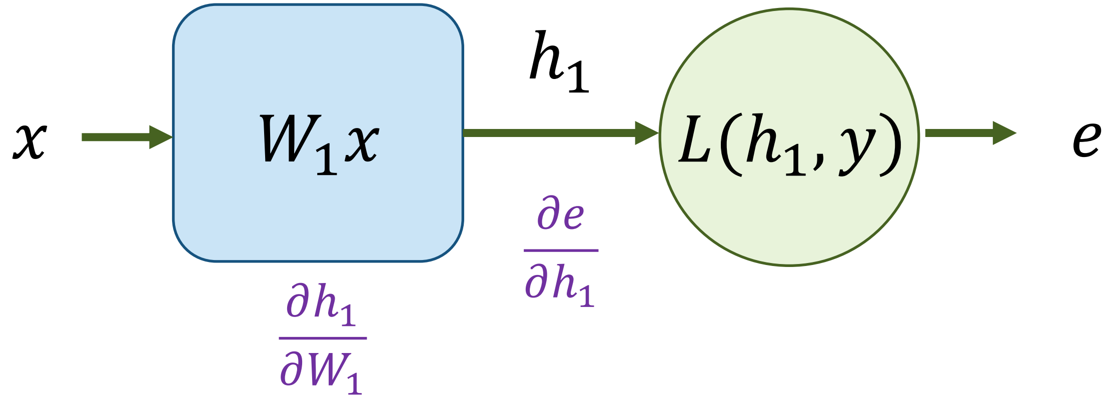
- Update the weights with SGD:

$$W_k \leftarrow W_k - \alpha \frac{\partial e}{\partial W_k}$$

- How to compute $\frac{\partial e}{\partial W_k}$ for each layer?

Back-Propagation

- 1-layer case



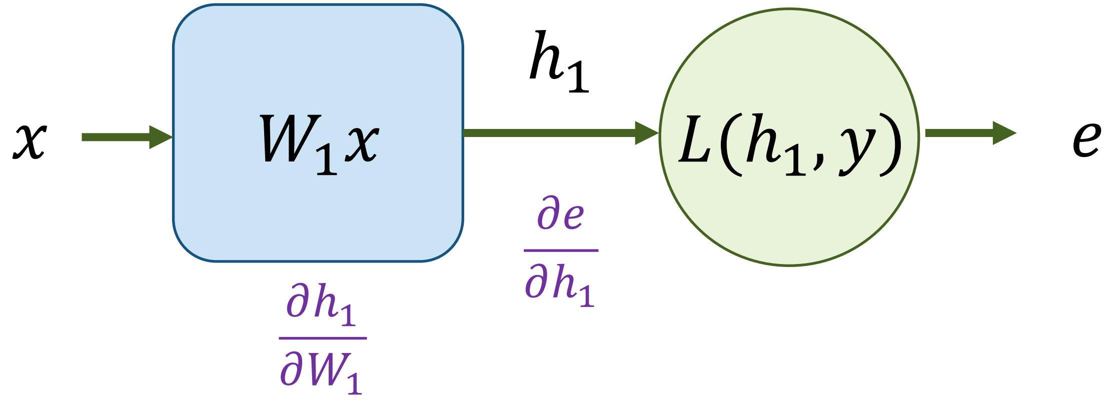
$$e = L(h_1, y) = L(W_1 x, y)$$

The chain rule:

$$\frac{\partial e}{\partial W_1} = \frac{\partial e}{\partial h_1} \frac{\partial h_1}{\partial W_1}$$

Back-Propagation

- 1-layer case



L2 loss example: $e = (y - h_1)^2$:

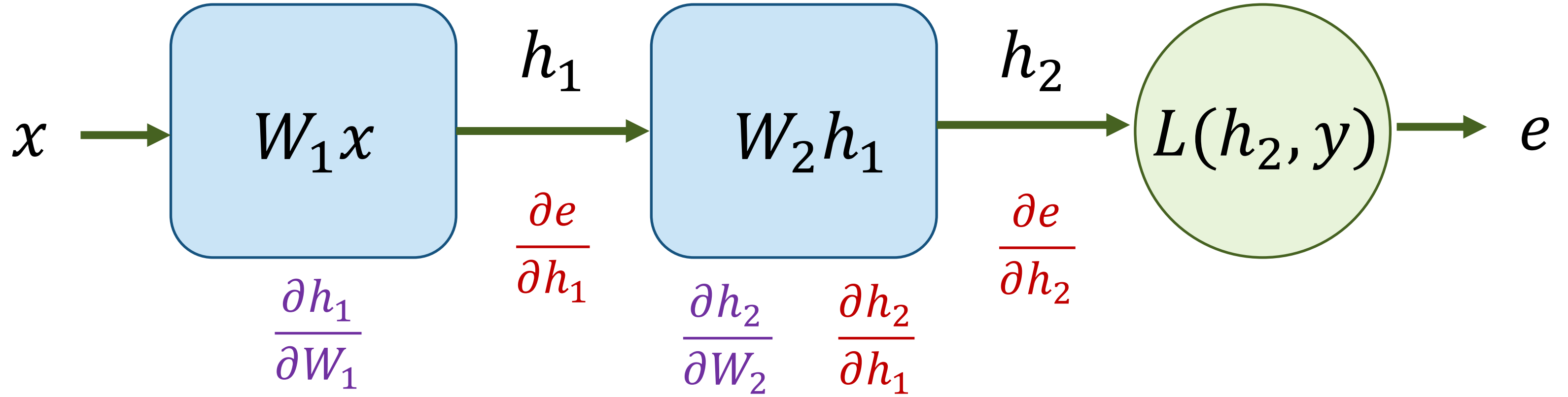
$$\frac{\partial e}{\partial h_1} = -2(y - h_1)\,, \qquad \frac{\partial h_1}{\partial W_1} = x$$

Using the chain rule:

$$\frac{\partial e}{\partial W_1} = \frac{\partial e}{\partial h_1}\frac{\partial h_1}{\partial W_1} = -2(y - h_1)x$$

Back-Propagation

- 2-layer case



- Easy one:

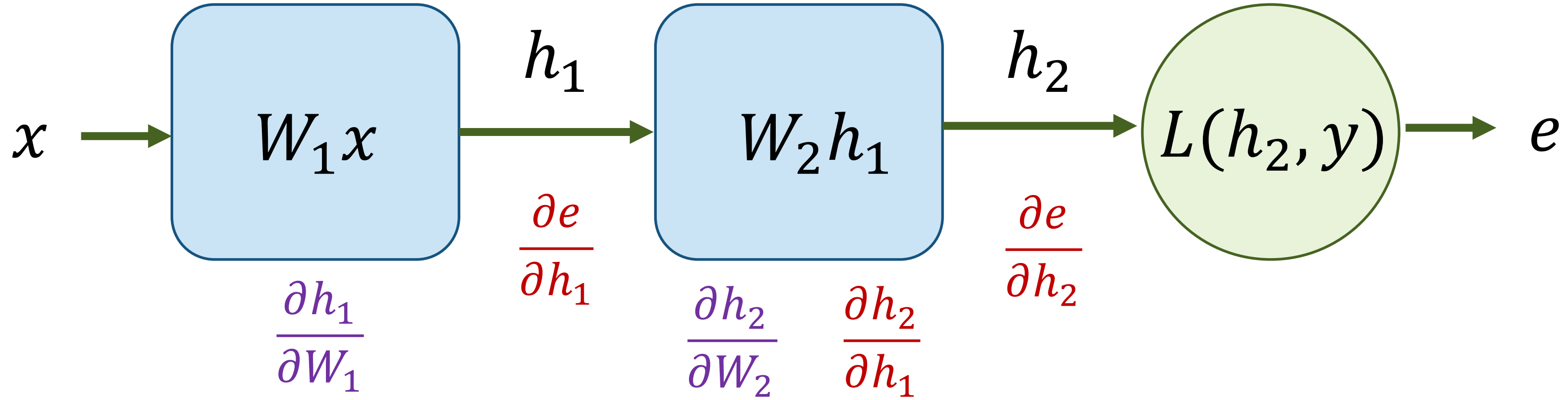
$$\frac{\partial e}{\partial W_2} = \frac{\partial e}{\partial h_2} \frac{\partial h_2}{\partial W_2}$$

- How to compute $\frac{\partial e}{\partial W_1}$?

$$\frac{\partial e}{\partial W_1} = \frac{\partial e}{\partial h_1} \frac{\partial h_1}{\partial W_1} = \frac{\partial e}{\partial h_2} \frac{\partial h_2}{\partial h_1} \frac{\partial h_1}{\partial W_1}$$

Back-Propagation

- 2-layer case



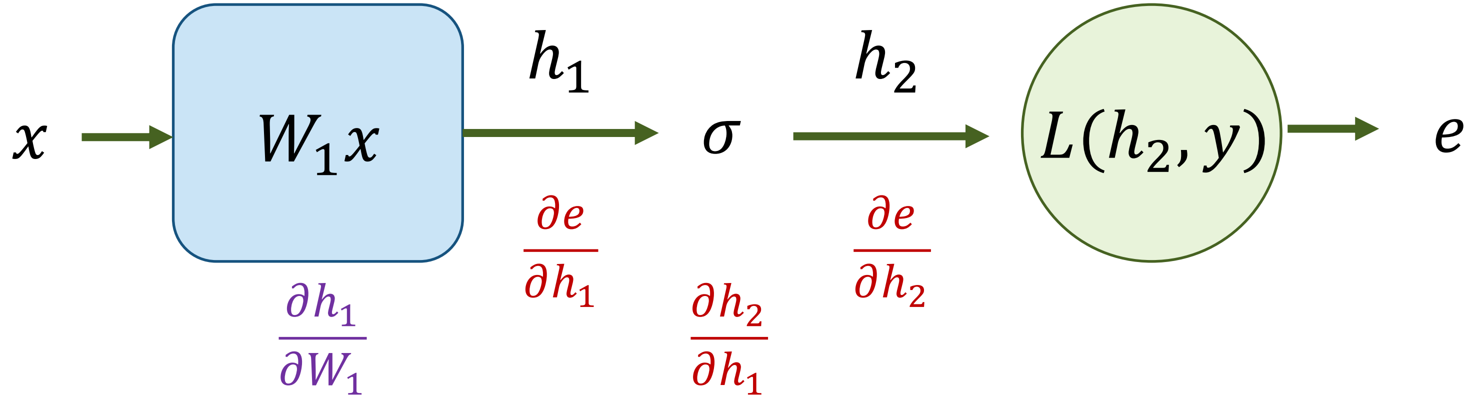
L2 loss example: $e = (y - h_2)^2$:

$$\frac{\partial e}{\partial h_2} = -2(y - h_2)\,, \qquad \frac{\partial h_2}{\partial h_1} = W_2, \qquad \frac{\partial h_1}{\partial W_1} = x$$

$$\frac{\partial e}{\partial W_1} = \frac{\partial e}{\partial h_2}\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial W_1} = -2(y - h_2)W_2 x$$

Back-Propagation

- 1-layer case

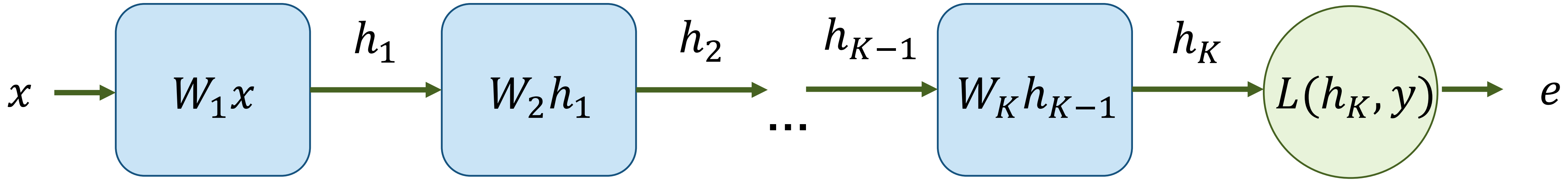


$$\frac{\partial e}{\partial W_1} = \frac{\partial e}{\partial h_1}\frac{\partial h_1}{\partial W_1} = \frac{\partial e}{\partial h_2}\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial W_1}$$

L2 loss example: $e = (y - h_2)^2$:

$$\frac{\partial e}{\partial h_2} = -2(y - h_2)\,, \qquad \frac{\partial h_2}{\partial h_1} = \sigma'(h_1) = \sigma(h_1)(1 - \sigma(h_1)), \qquad \frac{\partial h_1}{\partial W_1} = x$$

Back-Propagation with MLP



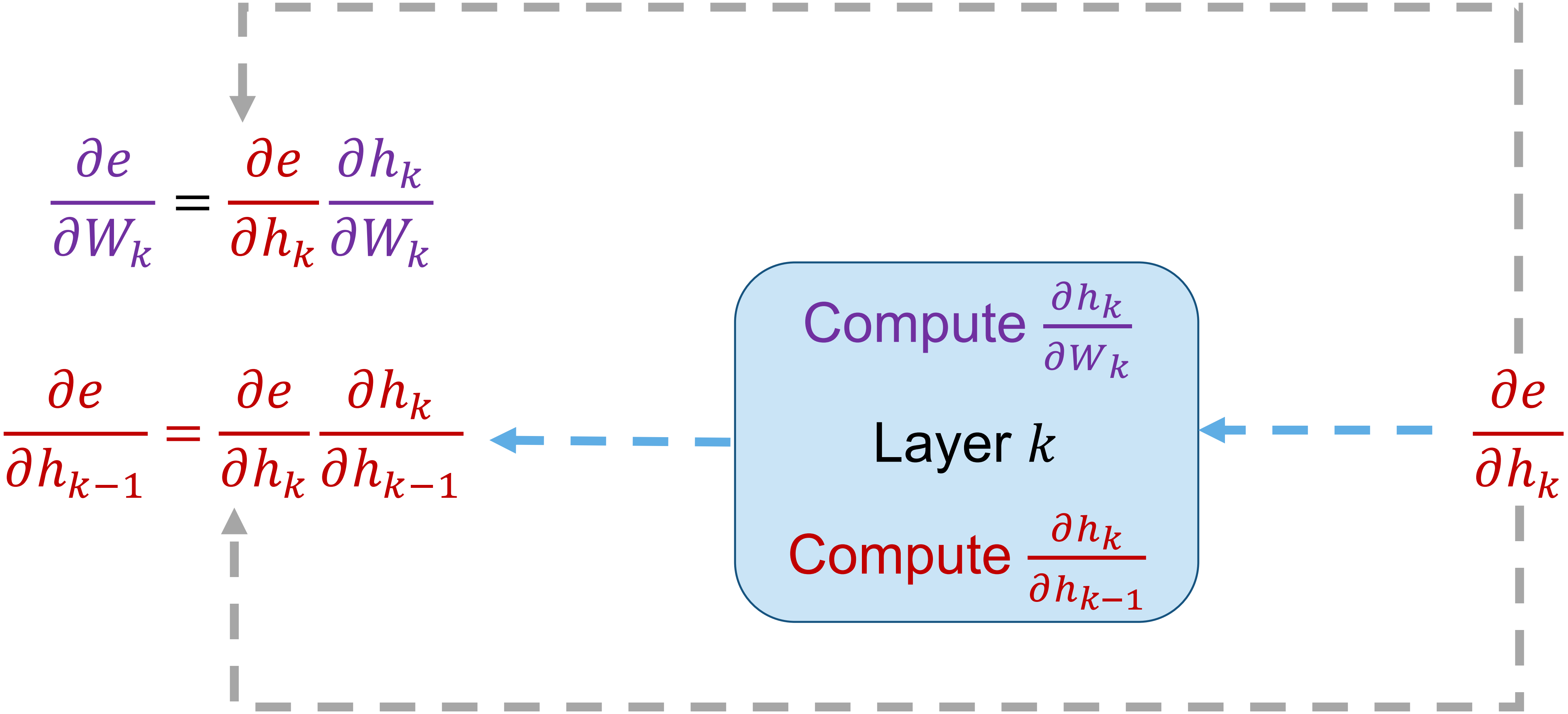
For any layer:

$$\frac{\partial e}{\partial W_k} = \frac{\partial e}{\partial h_k} \frac{\partial h_k}{\partial W_k} = \frac{\partial e}{\partial h_K} \cdots \frac{\partial h_{k+2}}{\partial h_{k+1}} \frac{\partial h_{k+1}}{\partial h_k} \frac{\partial h_k}{\partial W_k}$$

Gradient from the higher layer

Gradient from the current layer

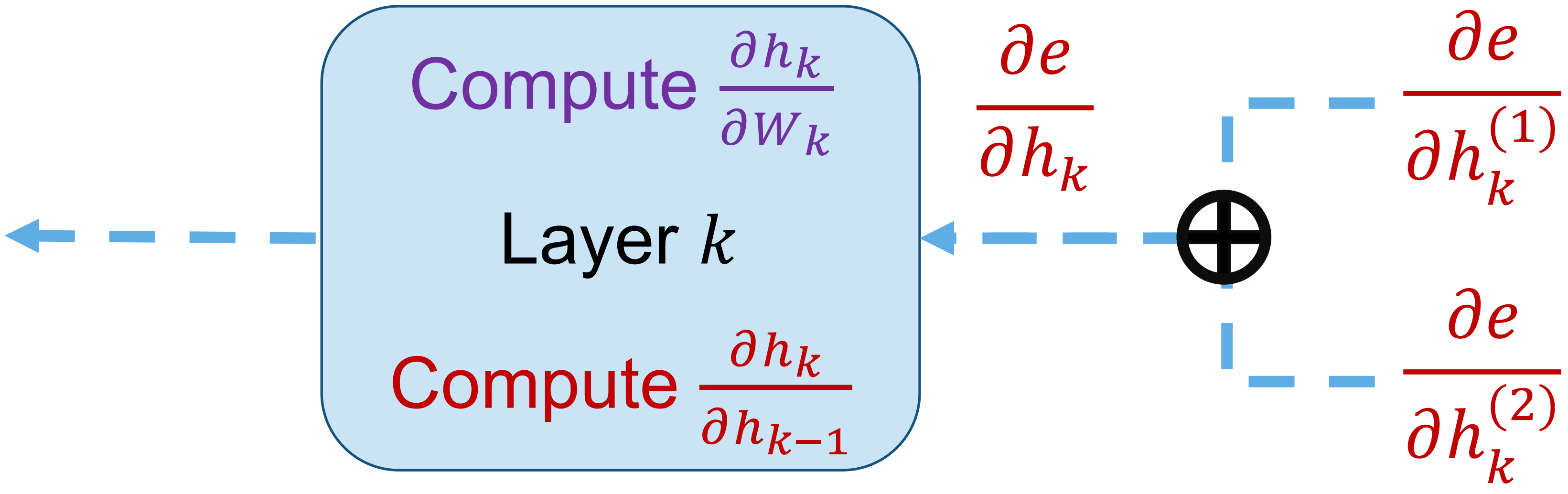
Back-Propagation with 1-layer



Back-Propagation with 1-layer

$$\frac{\partial e}{\partial W_k} = \frac{\partial e}{\partial h_k} \frac{\partial h_k}{\partial W_k}$$

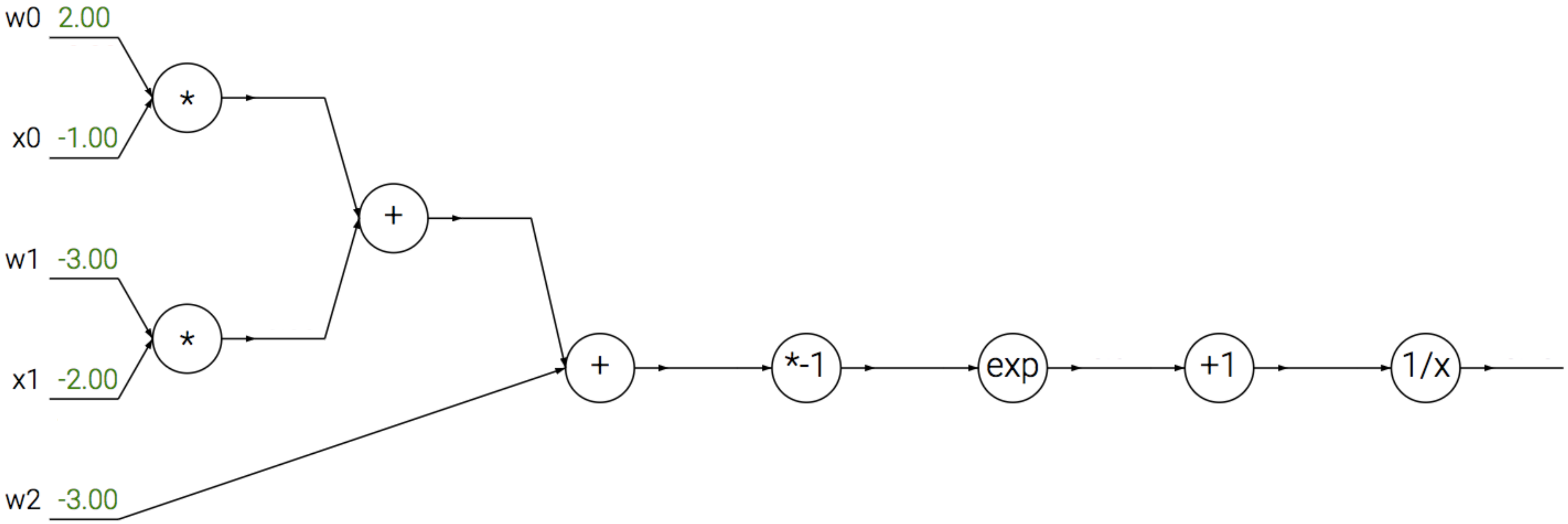
$$\frac{\partial e}{\partial h_{k-1}} = \frac{\partial e}{\partial h_k} \frac{\partial h_k}{\partial h_{k-1}}$$



An example for Back-Propagation

http://cs231n.stanford.edu/, https://slazebni.cs.illinois.edu/fall20/

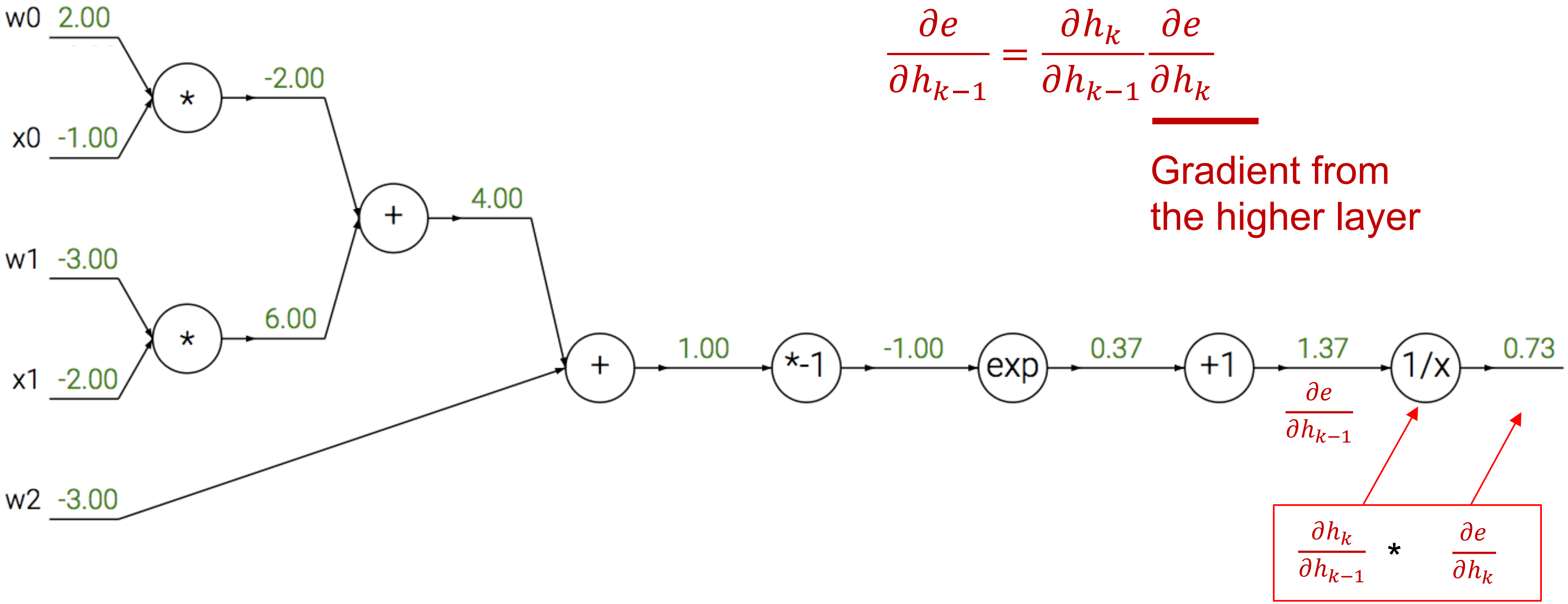
$$f(x, w) = \frac{1}{1 + \exp[-(w_0 x_0 + w_1 x_1 + w_2)]}$$



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

$$\frac{\partial e}{\partial h_{k-1}} = \frac{\partial h_k}{\partial h_{k-1}} \frac{\partial e}{\partial h_k}$$

Gradient from the higher layer

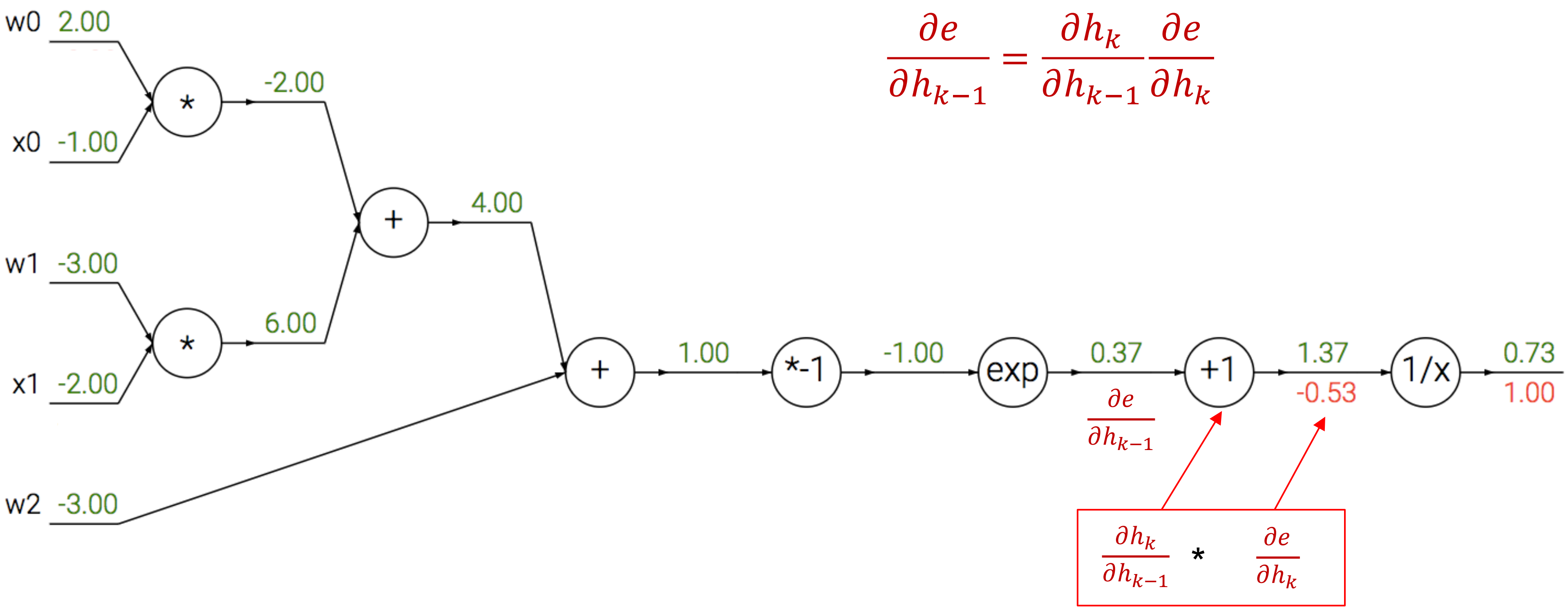


$$\frac{\partial h_k}{\partial h_{k-1}} = (1/x)' = -1/x^2$$

$$\frac{\partial e}{\partial h_{k-1}} = -\frac{1}{1.37^2} * 1 = -0.53$$

$$f(x,w) = \frac{1}{1+\exp[-(w_0x_0+w_1x_1+w_2)]}$$

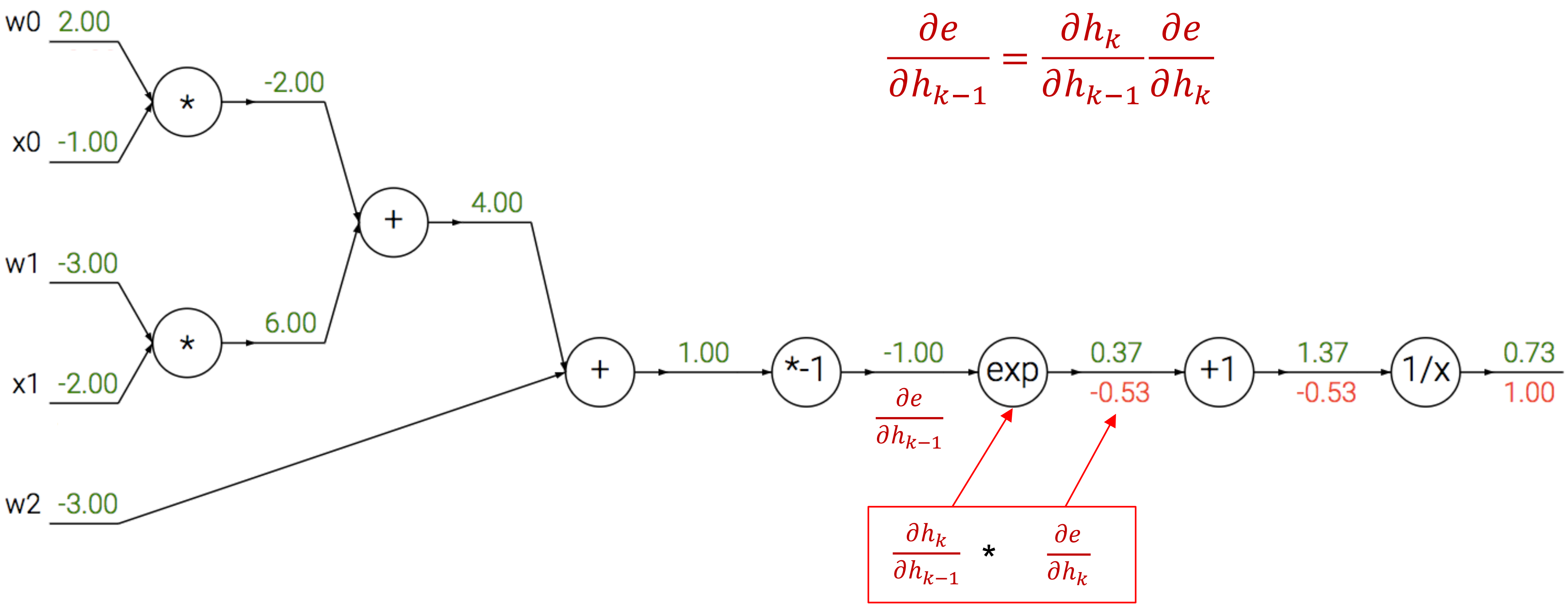
$$\frac{\partial e}{\partial h_{k-1}} = \frac{\partial h_k}{\partial h_{k-1}} \frac{\partial e}{\partial h_k}$$



$$\frac{\partial e}{\partial h_{k-1}} = 1 * \frac{\partial e}{\partial h_k}$$

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

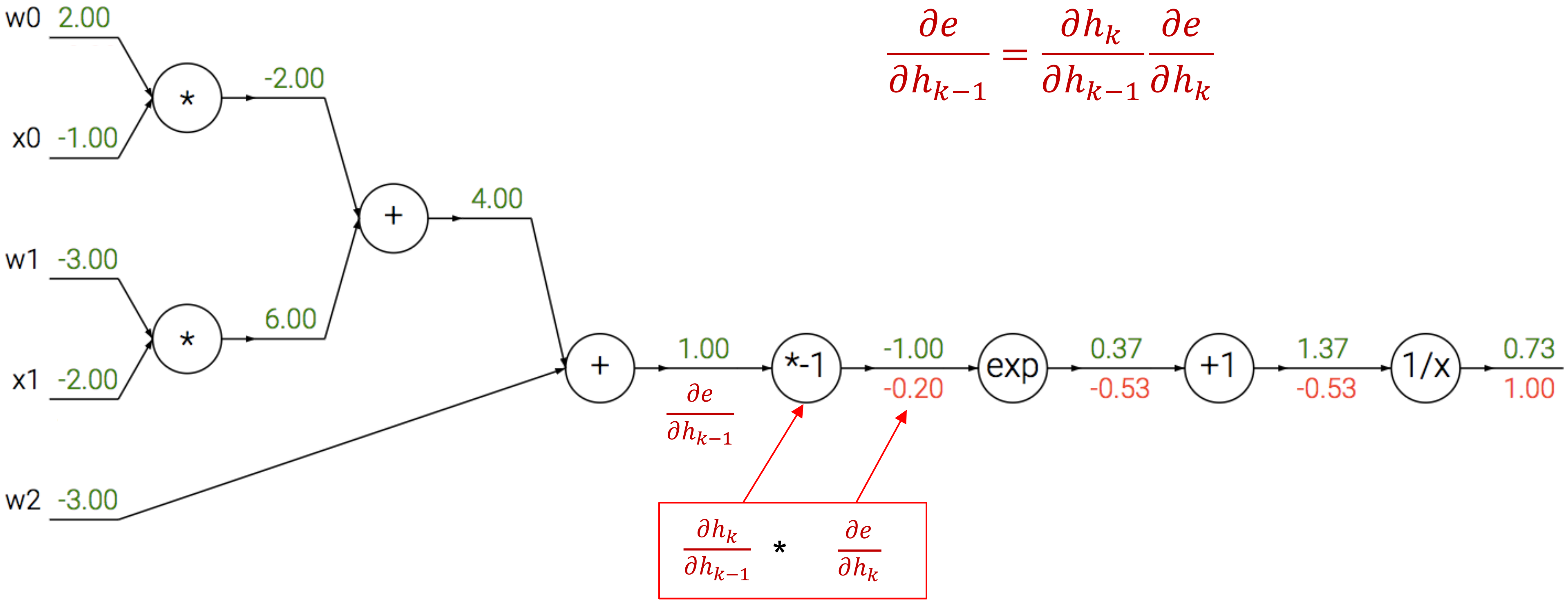
$$\frac{\partial e}{\partial h_{k-1}} = \frac{\partial h_k}{\partial h_{k-1}} \frac{\partial e}{\partial h_k}$$



$$\exp(-1) * (-0.53) = -0.20$$

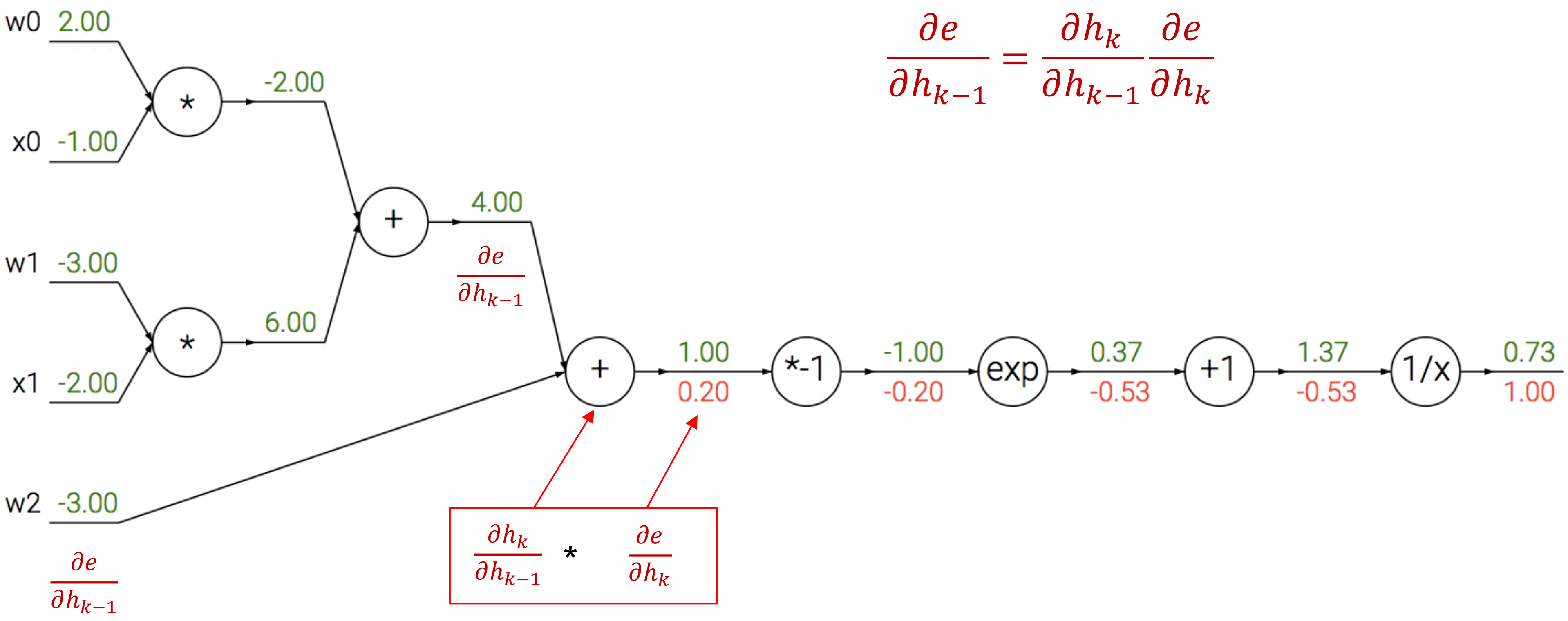
$$f(x,w) = \frac{1}{1 + \exp[-(w_0 x_0 + w_1 x_1 + w_2)]}$$

$$\frac{\partial e}{\partial h_{k-1}} = \frac{\partial h_k}{\partial h_{k-1}} \frac{\partial e}{\partial h_k}$$



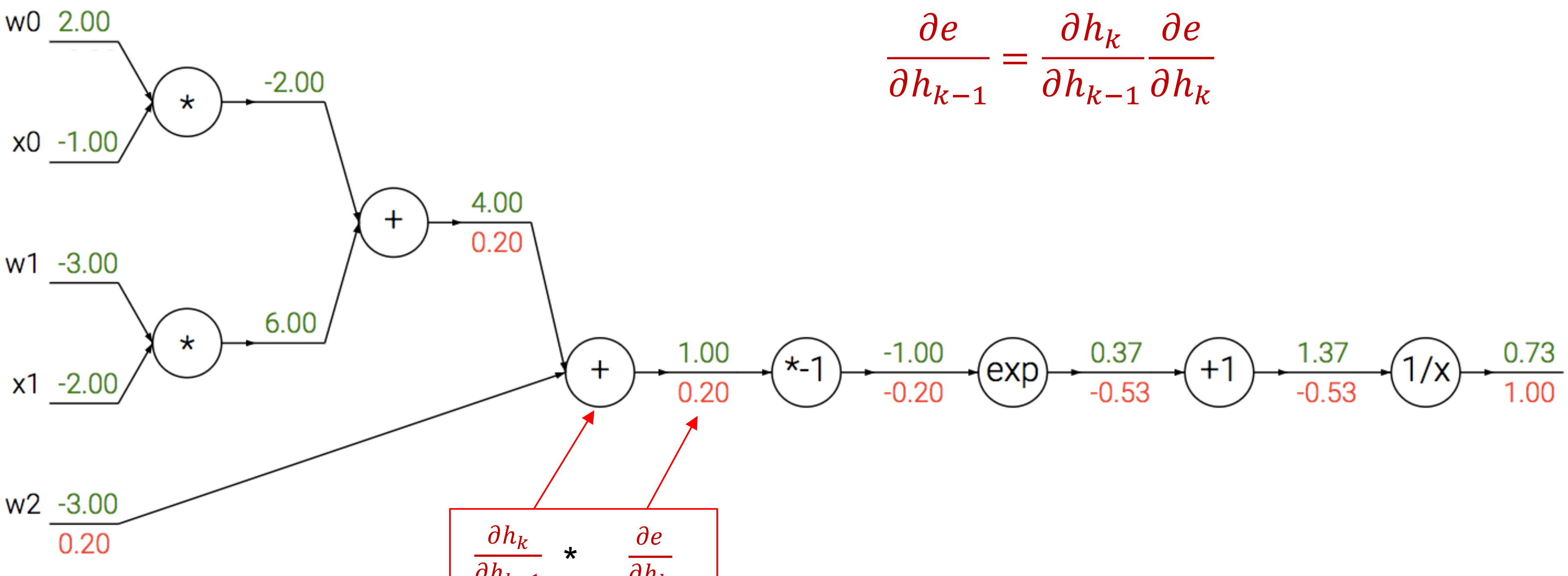
$$f(x,w) = \frac{1}{1 + \exp[-(w_0 x_0 + w_1 x_1 + w_2)]}$$

$$\frac{\partial e}{\partial h_{k-1}} = \frac{\partial h_k}{\partial h_{k-1}} \frac{\partial e}{\partial h_k}$$



$$f(x,w) = \frac{1}{1+\exp[-(w_0x_0 + w_1x_1 + w_2)]}$$

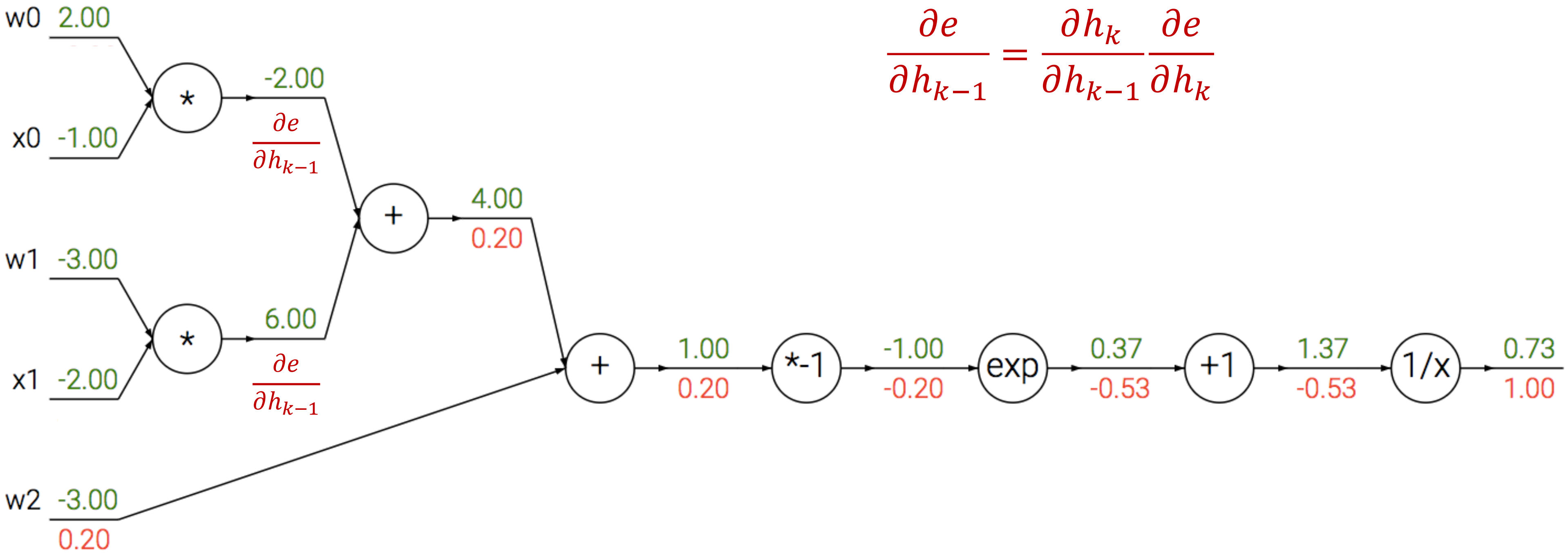
$$\frac{\partial e}{\partial h_{k-1}} = \frac{\partial h_k}{\partial h_{k-1}} \frac{\partial e}{\partial h_k}$$



$\frac{\partial h_k}{\partial h_{k-1}} * \frac{\partial e}{\partial h_k}$

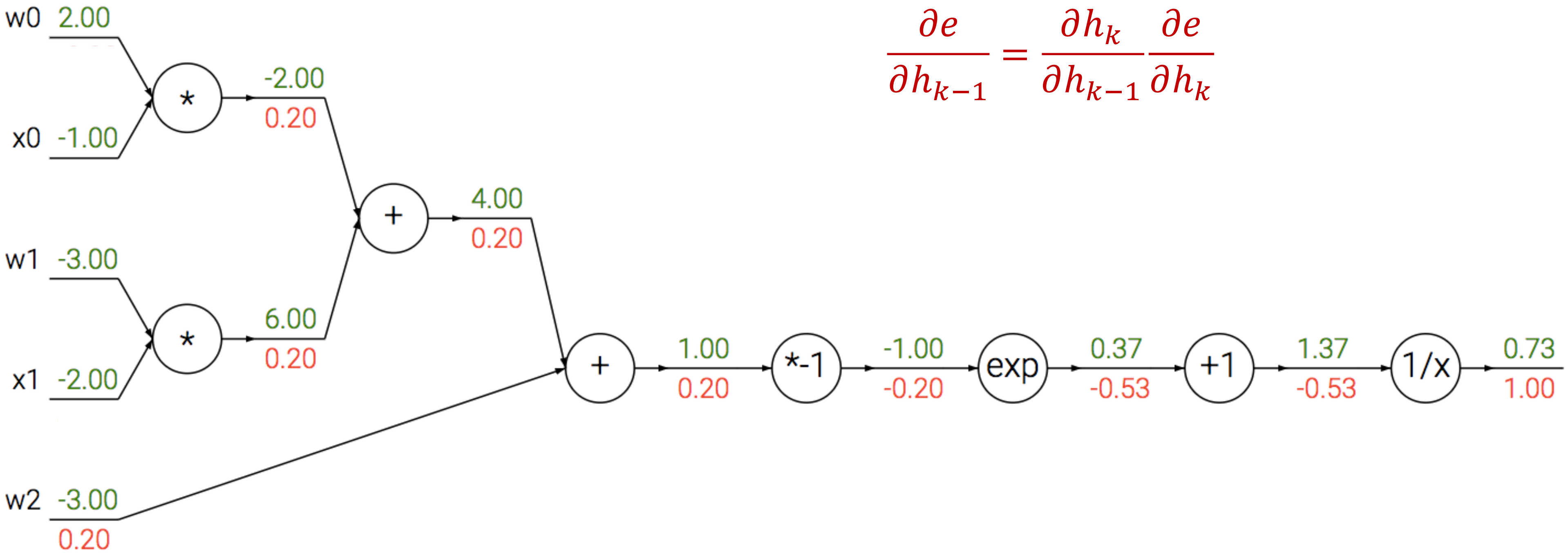
$$f(x,w) = \frac{1}{1 + \exp[-(w_0 x_0 + w_1 x_1 + w_2)]}$$

$$\frac{\partial e}{\partial h_{k-1}} = \frac{\partial h_k}{\partial h_{k-1}} \frac{\partial e}{\partial h_k}$$



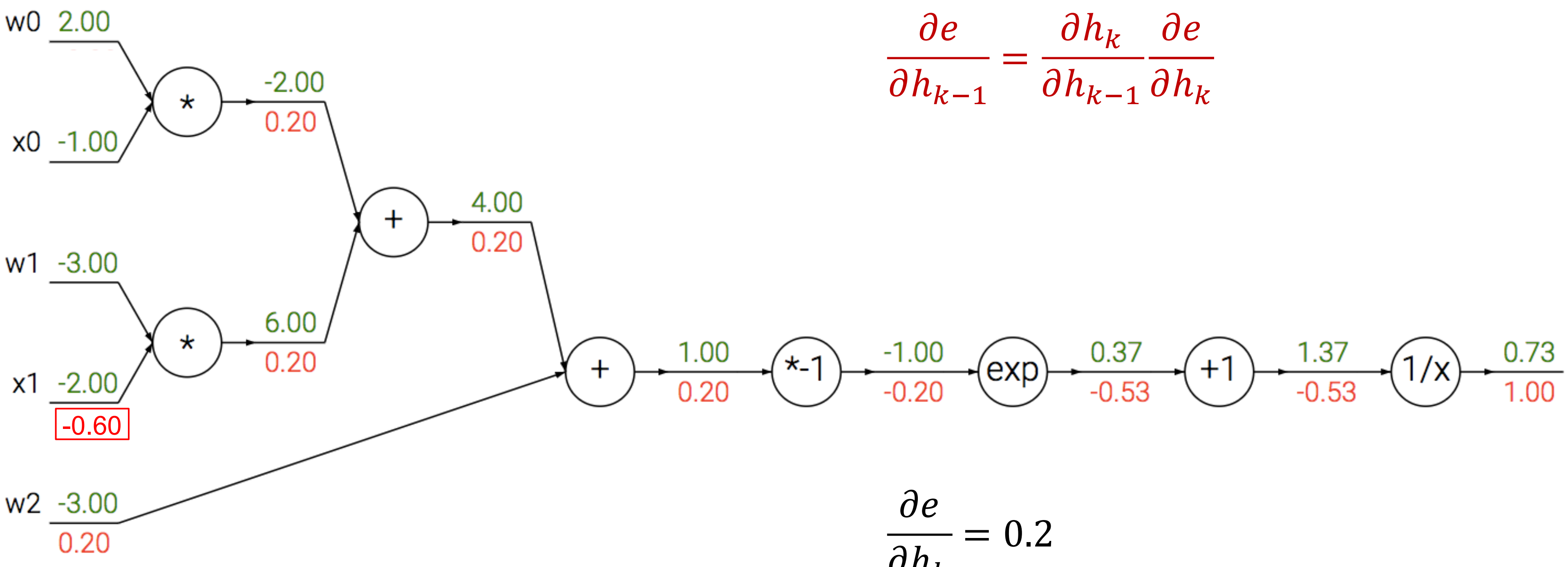
$$f(x,w) = \frac{1}{1+\exp[-(w_0x_0+w_1x_1+w_2)]}$$

$$\frac{\partial e}{\partial h_{k-1}} = \frac{\partial h_k}{\partial h_{k-1}}\frac{\partial e}{\partial h_k}$$



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

$$\frac{\partial e}{\partial h_{k-1}} = \frac{\partial h_k}{\partial h_{k-1}} \frac{\partial e}{\partial h_k}$$



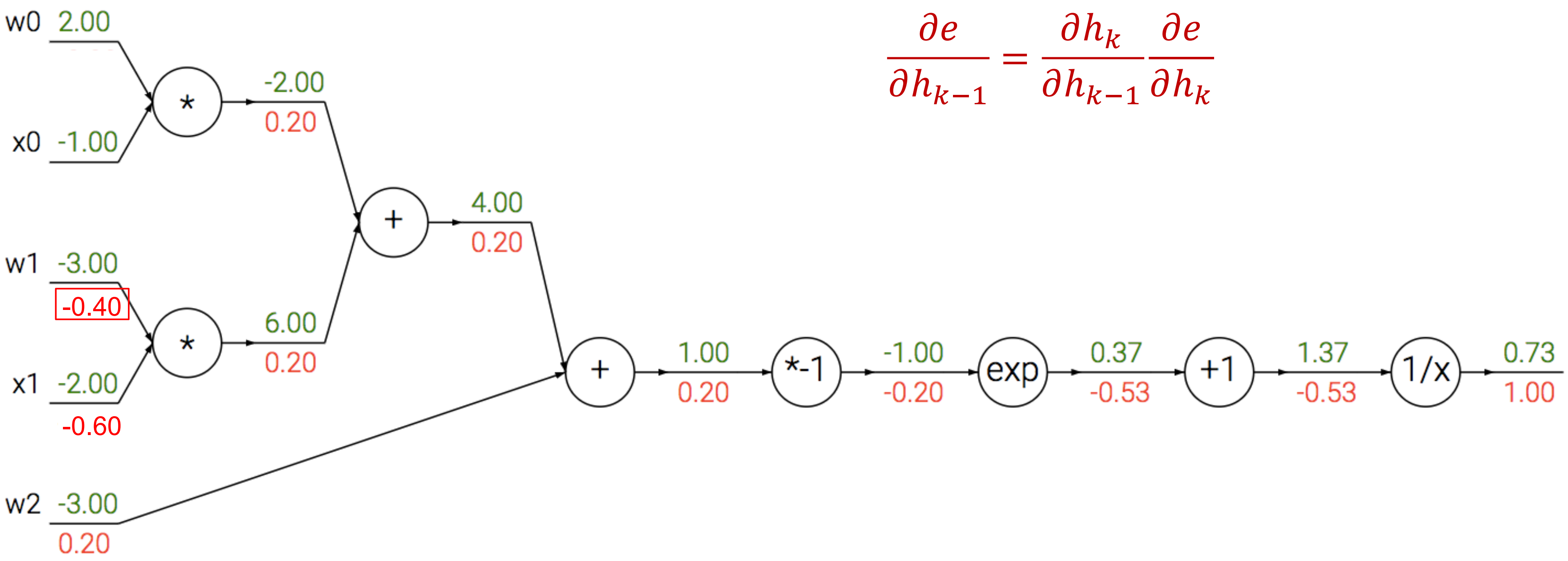
$$\frac{\partial e}{\partial h_k} = 0.2$$

$$\frac{\partial h_k}{\partial h_{k-1}} = w1 = -3.0$$

$$\frac{\partial e}{\partial h_{k-1}} = \frac{\partial h_k}{\partial h_{k-1}} \frac{\partial e}{\partial h_k} = -3.0 \ * 0.2 = -0.60$$

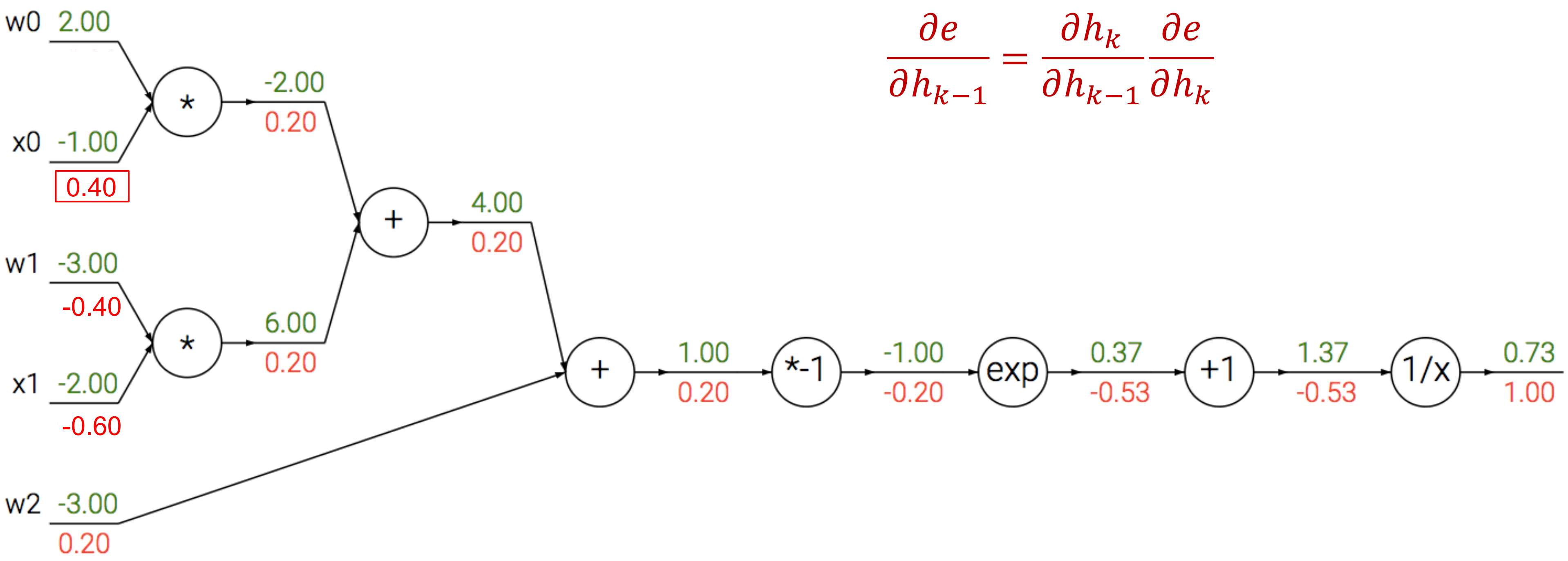
$$f(x,w) = \frac{1}{1 + \exp[-(w_0 x_0 + w_1 x_1 + w_2)]}$$

$$\frac{\partial e}{\partial h_{k-1}} = \frac{\partial h_k}{\partial h_{k-1}} \frac{\partial e}{\partial h_k}$$



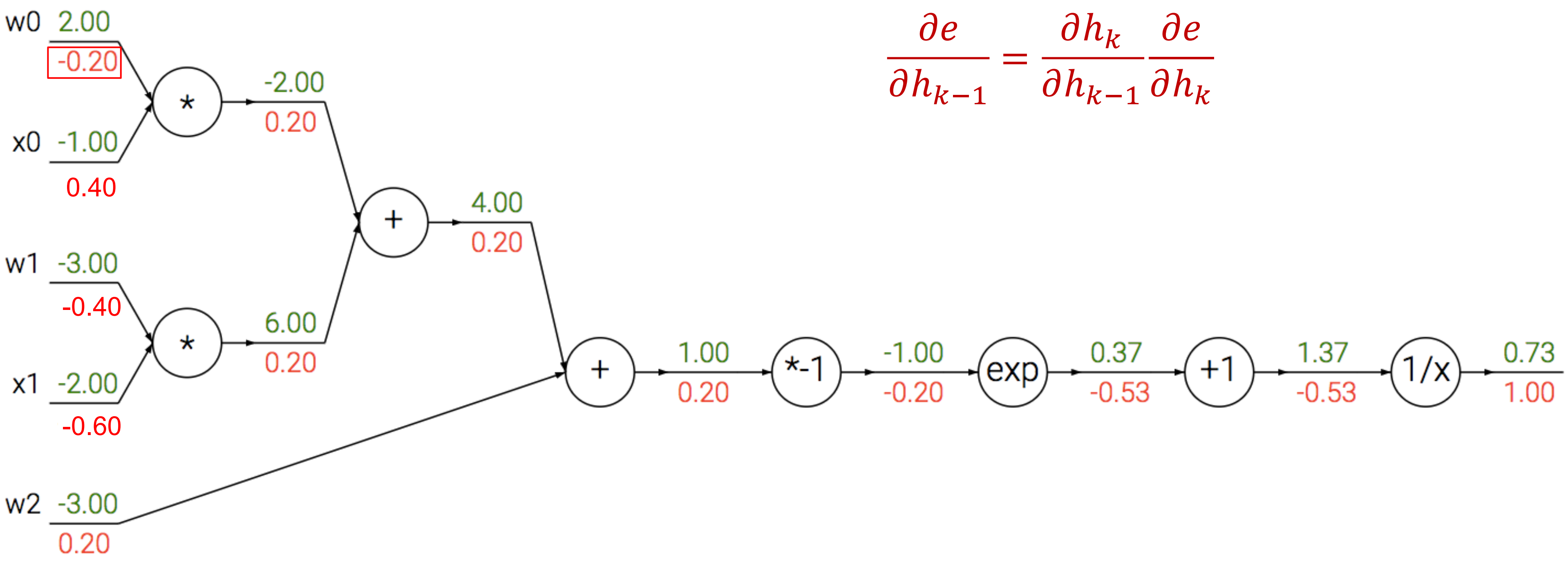
$$f(x,w) = \frac{1}{1+\exp[-(w_0x_0 + w_1x_1 + w_2)]}$$

$$\frac{\partial e}{\partial h_{k-1}} = \frac{\partial h_k}{\partial h_{k-1}} \frac{\partial e}{\partial h_k}$$

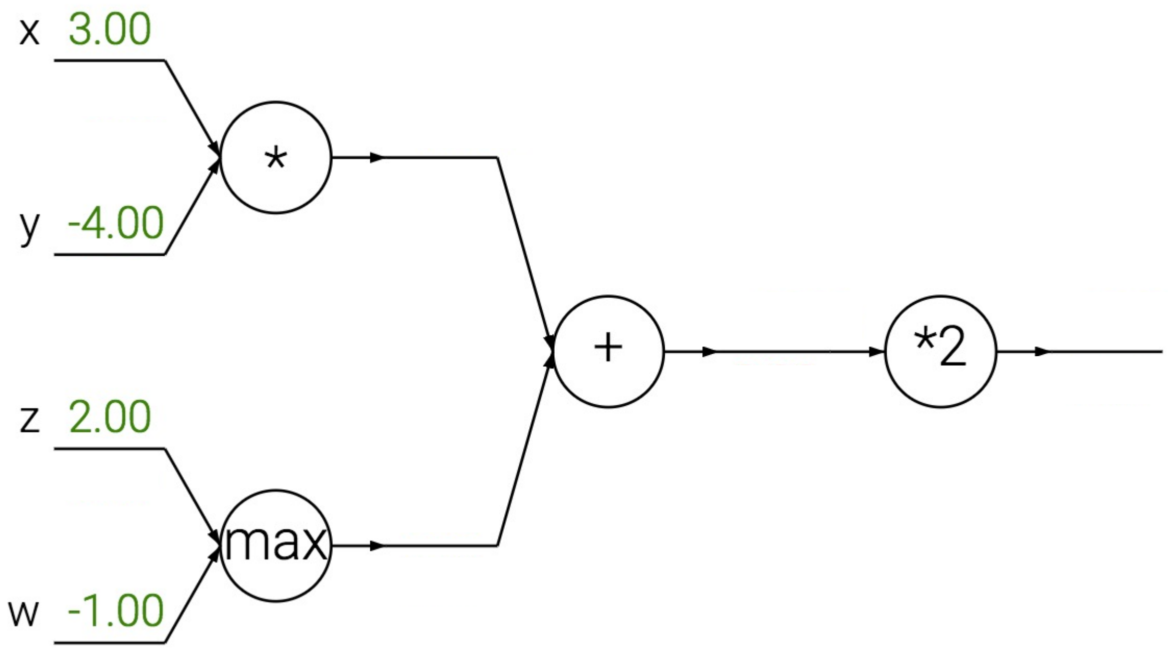


$$f(x,w) = \frac{1}{1+\exp[-(w_0x_0 + w_1x_1 + w_2)]}$$

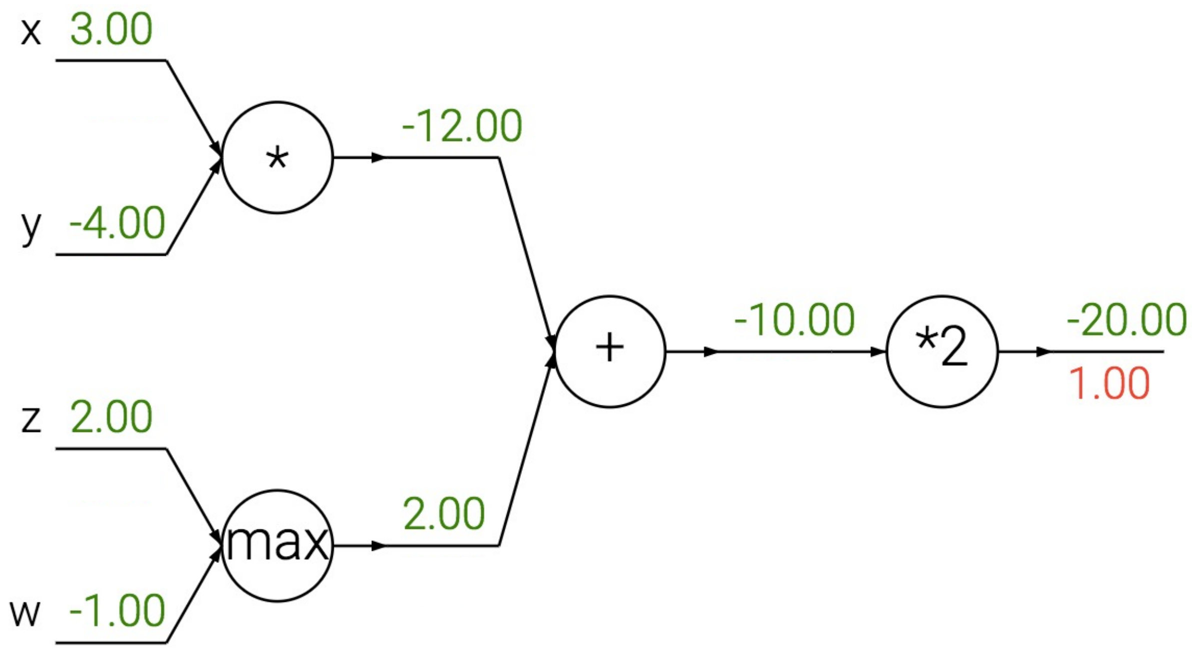
$$\frac{\partial e}{\partial h_{k-1}} = \frac{\partial h_k}{\partial h_{k-1}} \frac{\partial e}{\partial h_k}$$



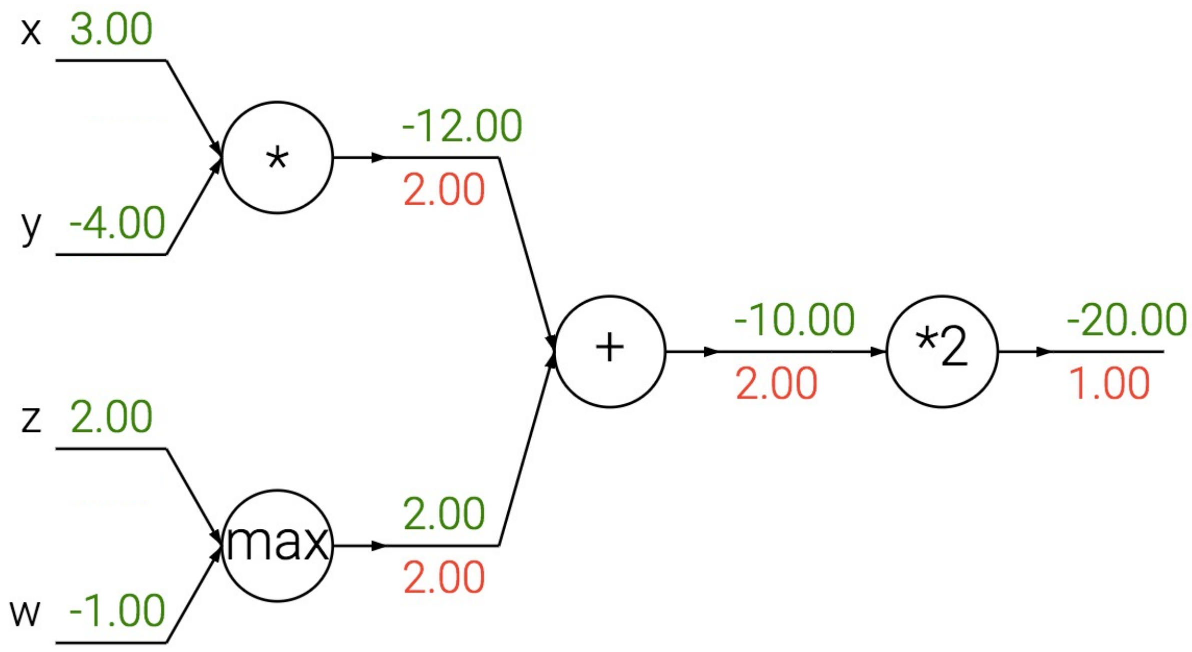
Another example



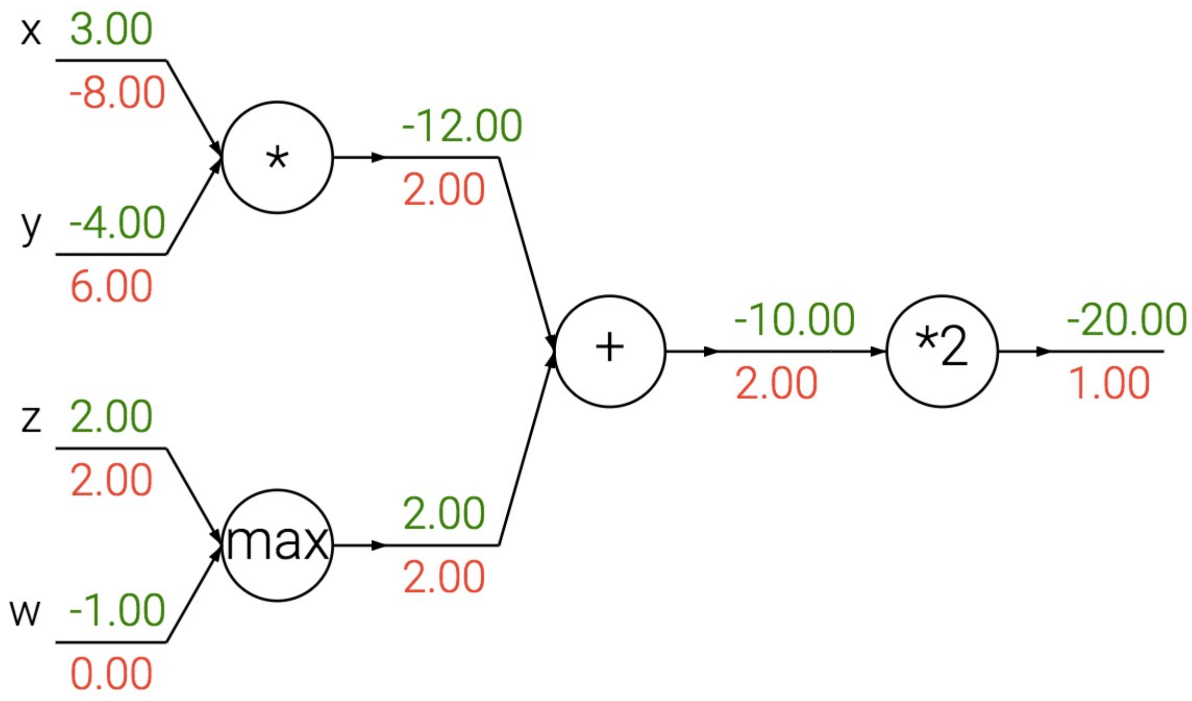
Another example



Another example



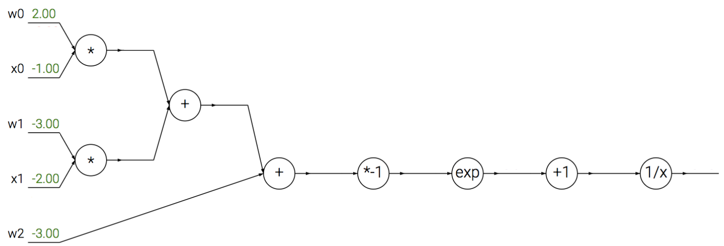
Another example



Good practice

- Derive the 2-layer network case yourself

- Good through the example and compute the gradients yourself



- Homework

Next Class

- Convolutional Neural Networks

- Training Convolutional Neural Networks with back-propagation