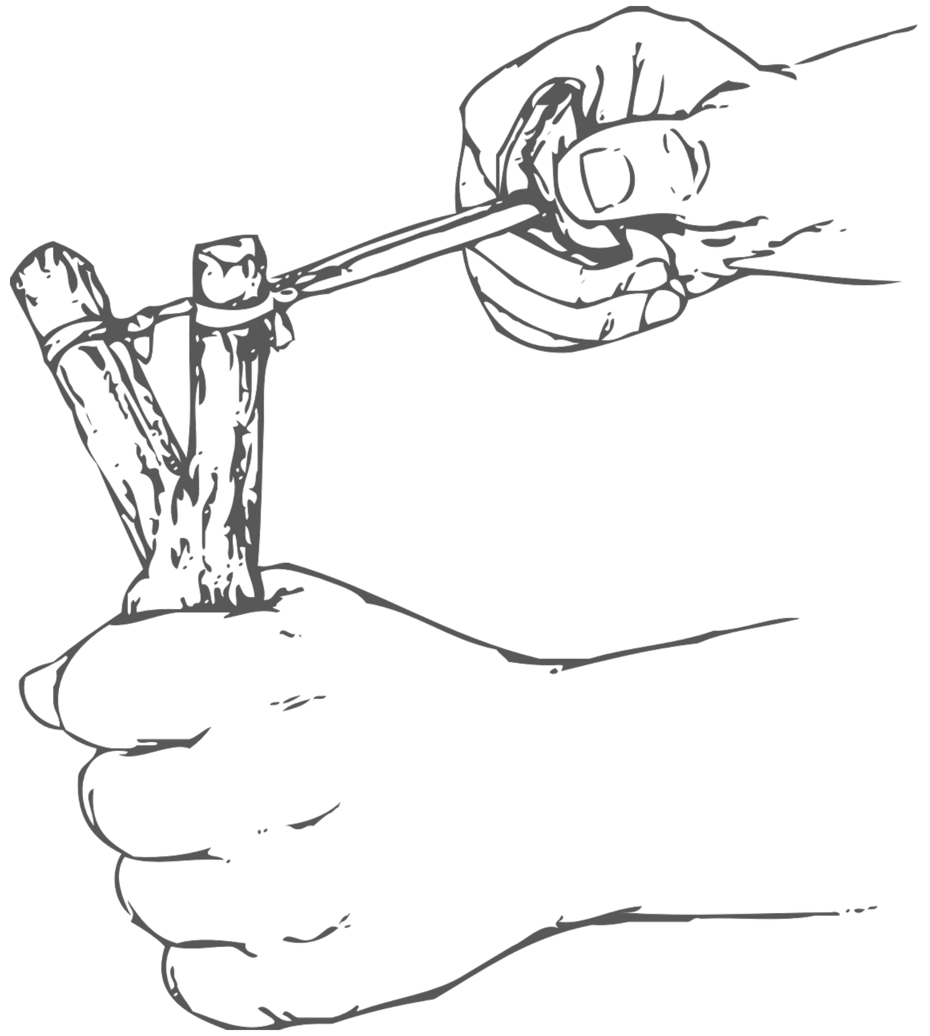
Neural·Pragmatic Natural Language Generation

N·P NLG

Learning goals



1. Become familiar with ANNs:
   a. mathematical notation in matrix-vector form
   b. weights & biases (slopes & intercepts), score, activation function, hidden layers, prediction
2. Be able to use PyTorch to implement a feed-forward ANN:
   a. building the model by hand
   b. using built-in helper functions (nn.Module, DataLoader …)

Units (neurons)

- input vector:

  $$\mathbf{x} = [x_1, \ldots, x_n]^T$$

- weight vector:

  $$\mathbf{w} = [w_1, \ldots, w_n]^T$$

- bias:

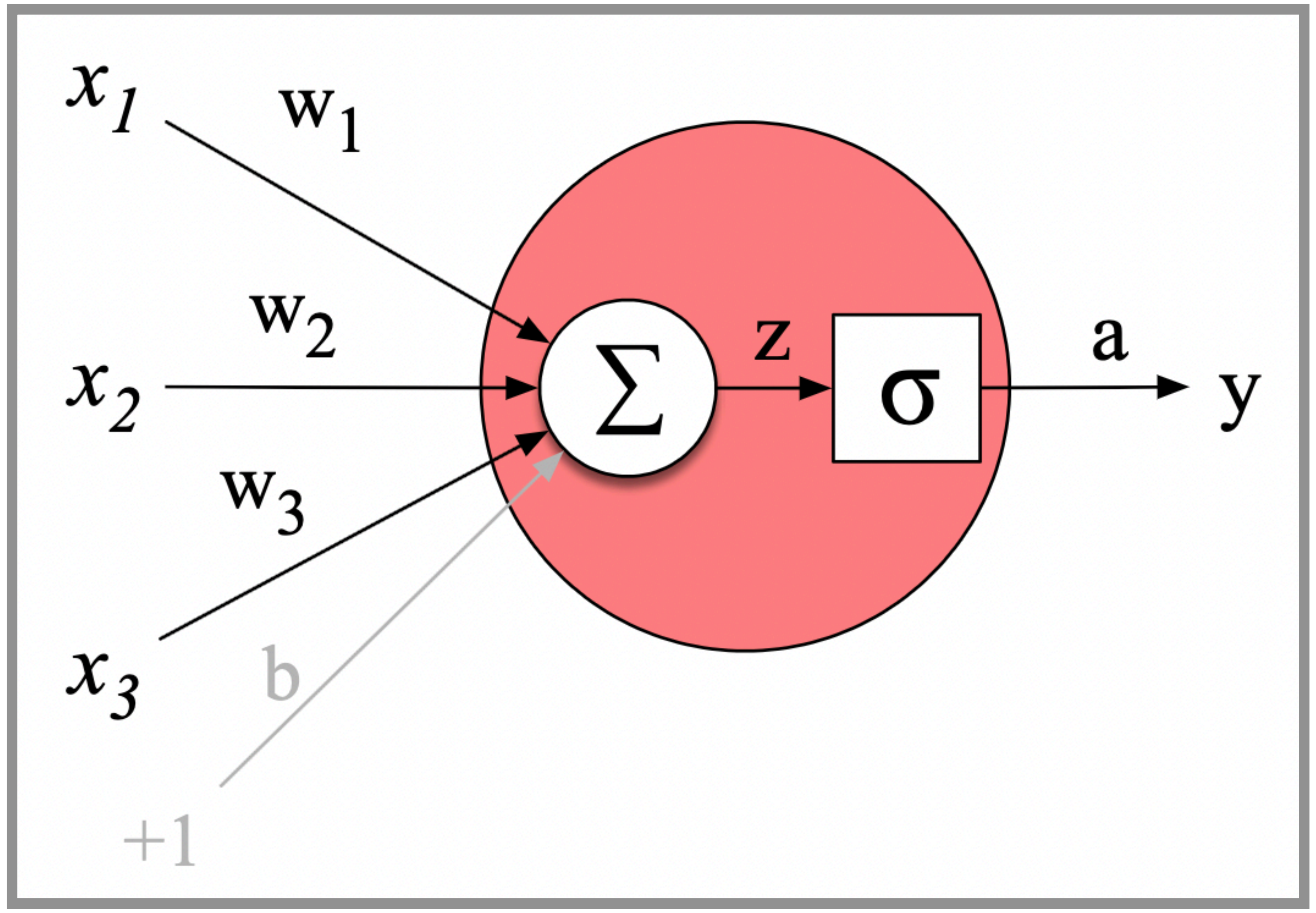
  $$b$$

- score:

  $$z = b + \sum_{j=1}^{n} w_j x_j = b + \mathbf{w} \cdot \mathbf{x}$$

- activation level:

  $a = f(z)$, where f is the **activation function**



Common activation functions

- perceptron:

$$f(z) = \delta_{z>0}$$

- sigmoid:

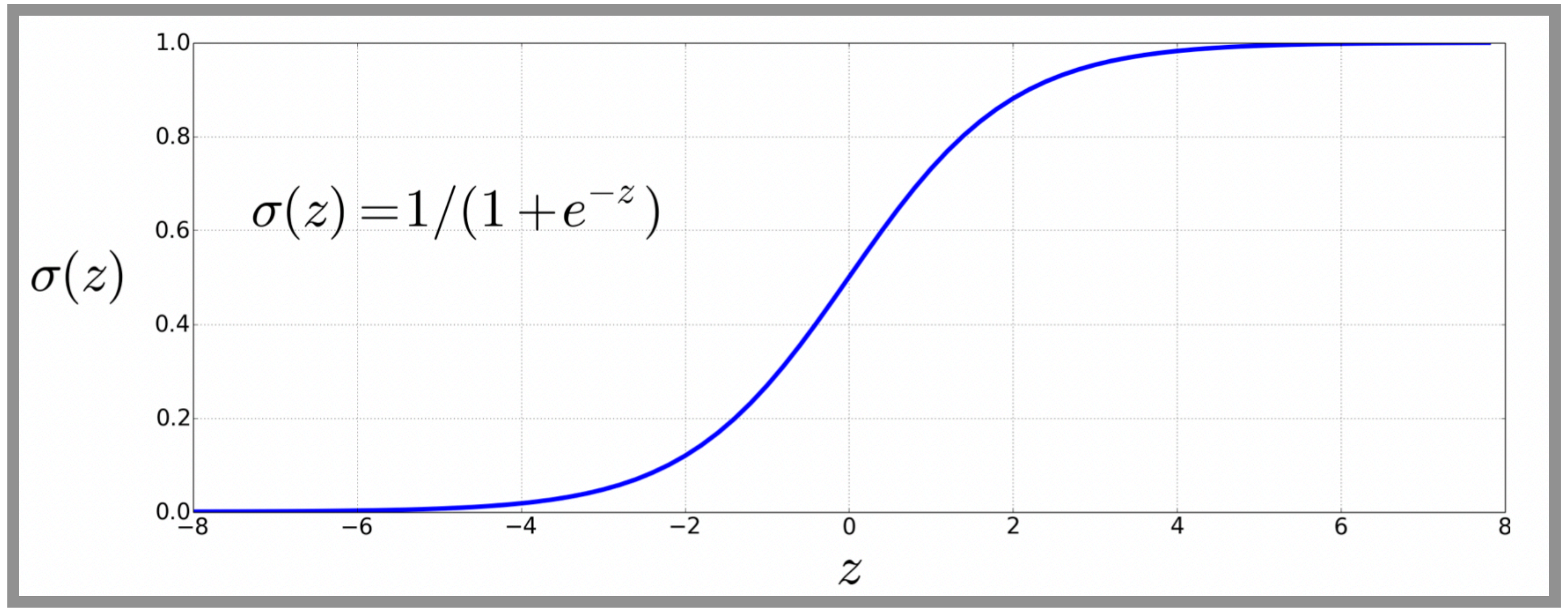
$$f(z) = \sigma(z) = \frac{1}{1 + \exp(-z)}$$

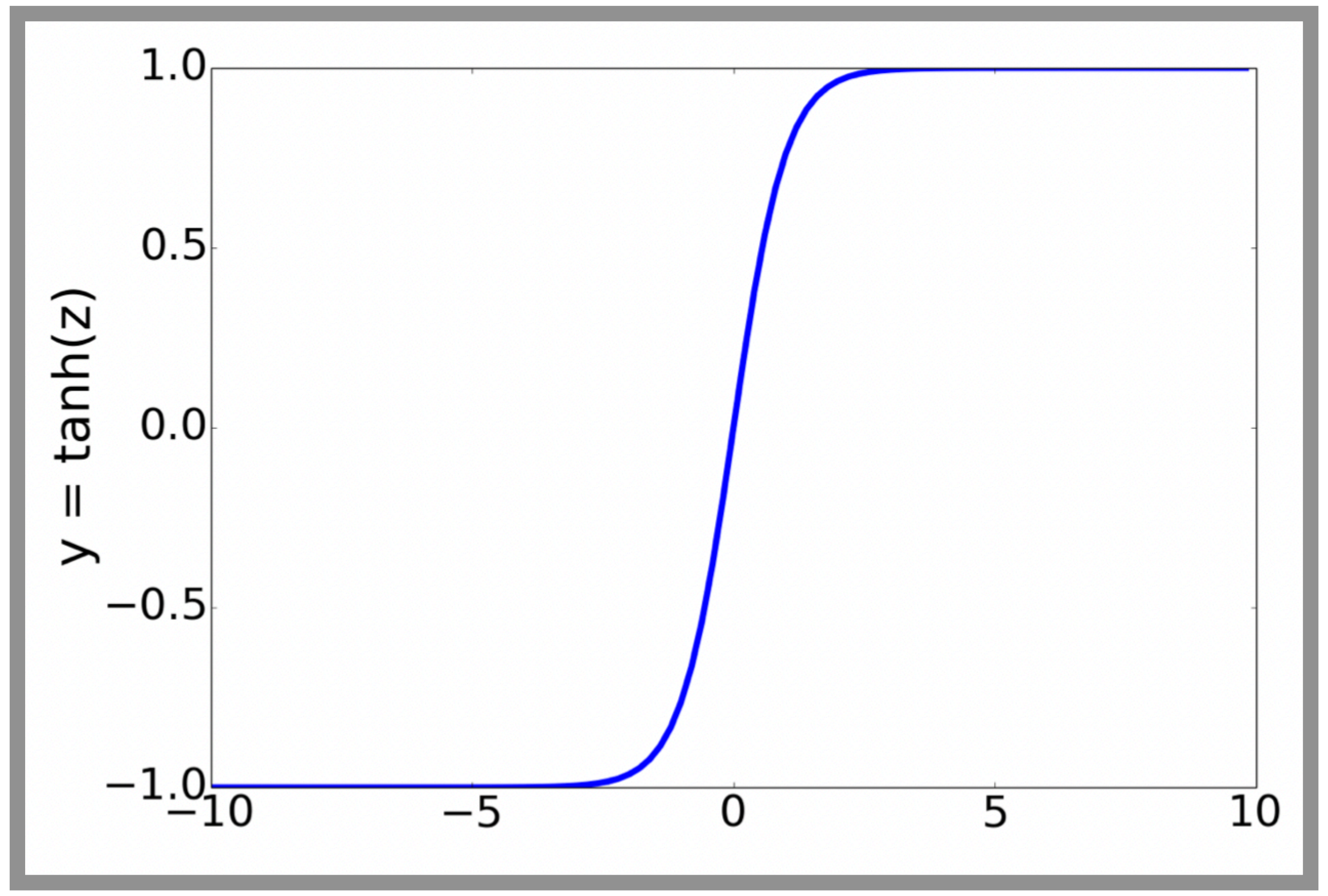
- hyperbolic tangent:

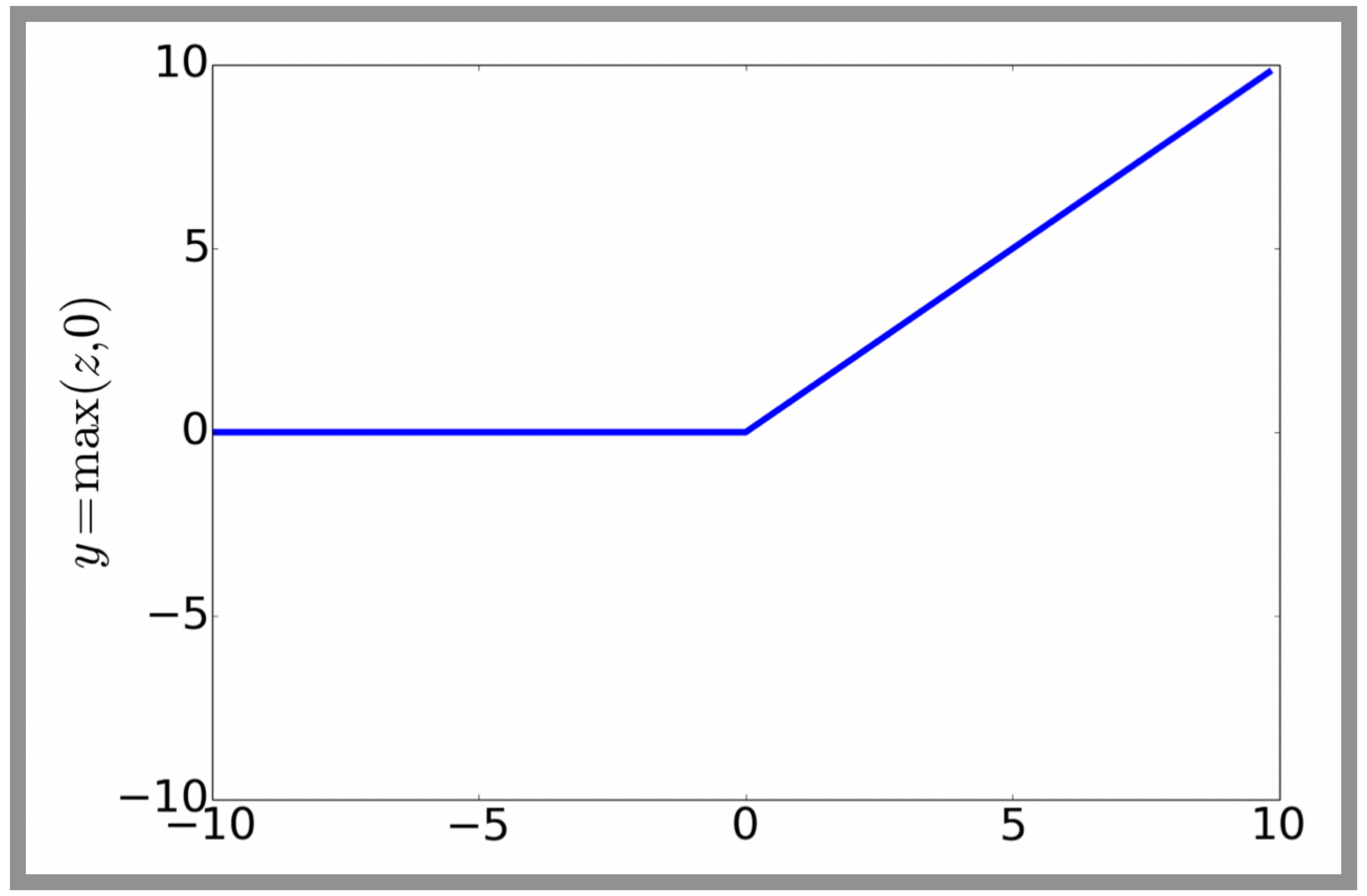
$$f(z) = \tanh(z) = \frac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)}$$

- rectified linear unit:

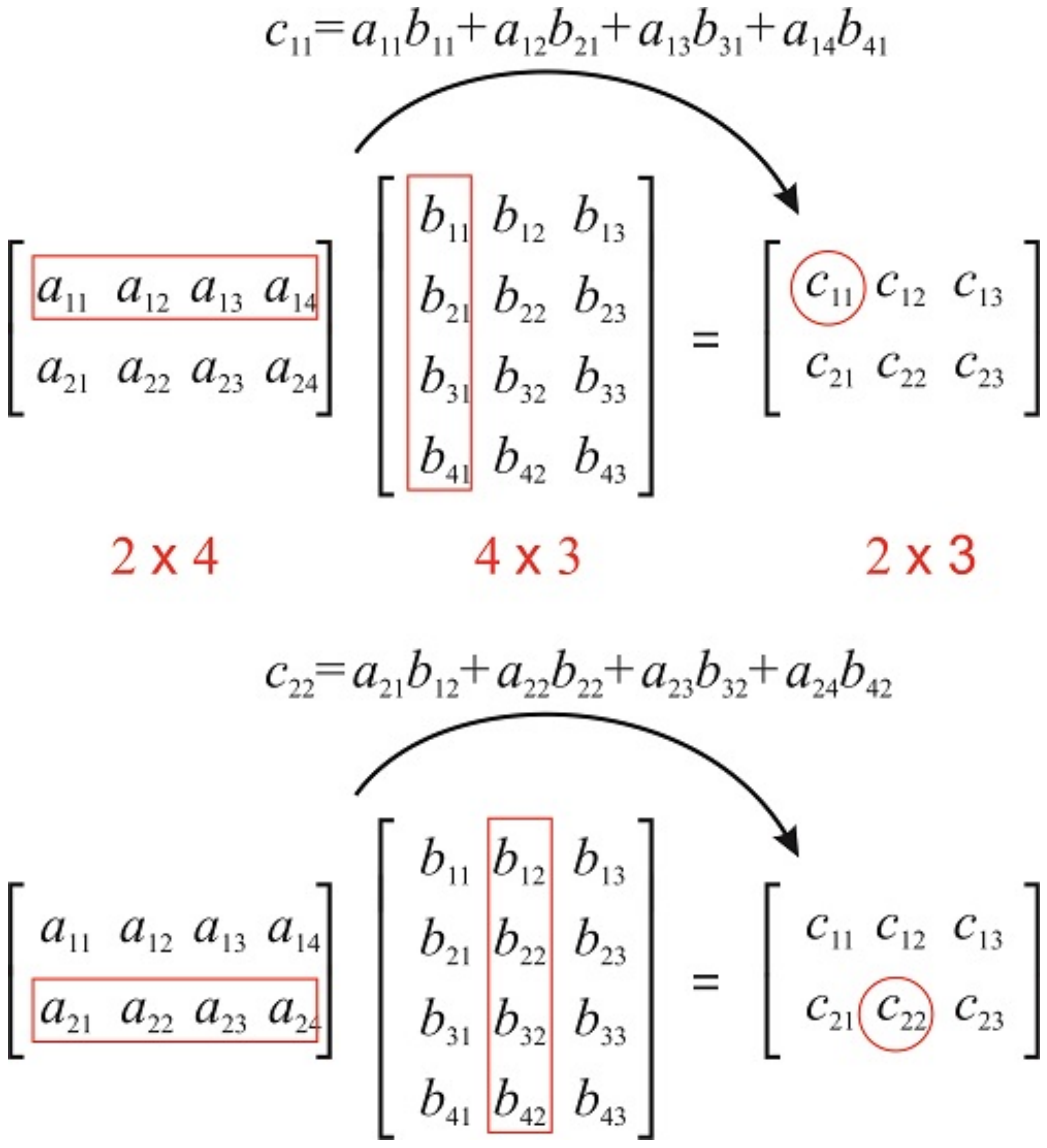
$$f(z) = \mathrm{ReLU}(z) = \max(z, 0)$$







Recap: Matrix Multiplication



Feedforward neural network (one hidden layer)

- input:

  $\mathbf{x} = [x_1, \ldots, x_{n_x}]^T$

- weight matrix:

  $\mathbf{W} \in \mathbb{R}^{n_k \times n_x}$

- bias vector:

  $\mathbf{b} = [b_1, \ldots, b_{n_k}]^T$

- activation vector hidden layer:

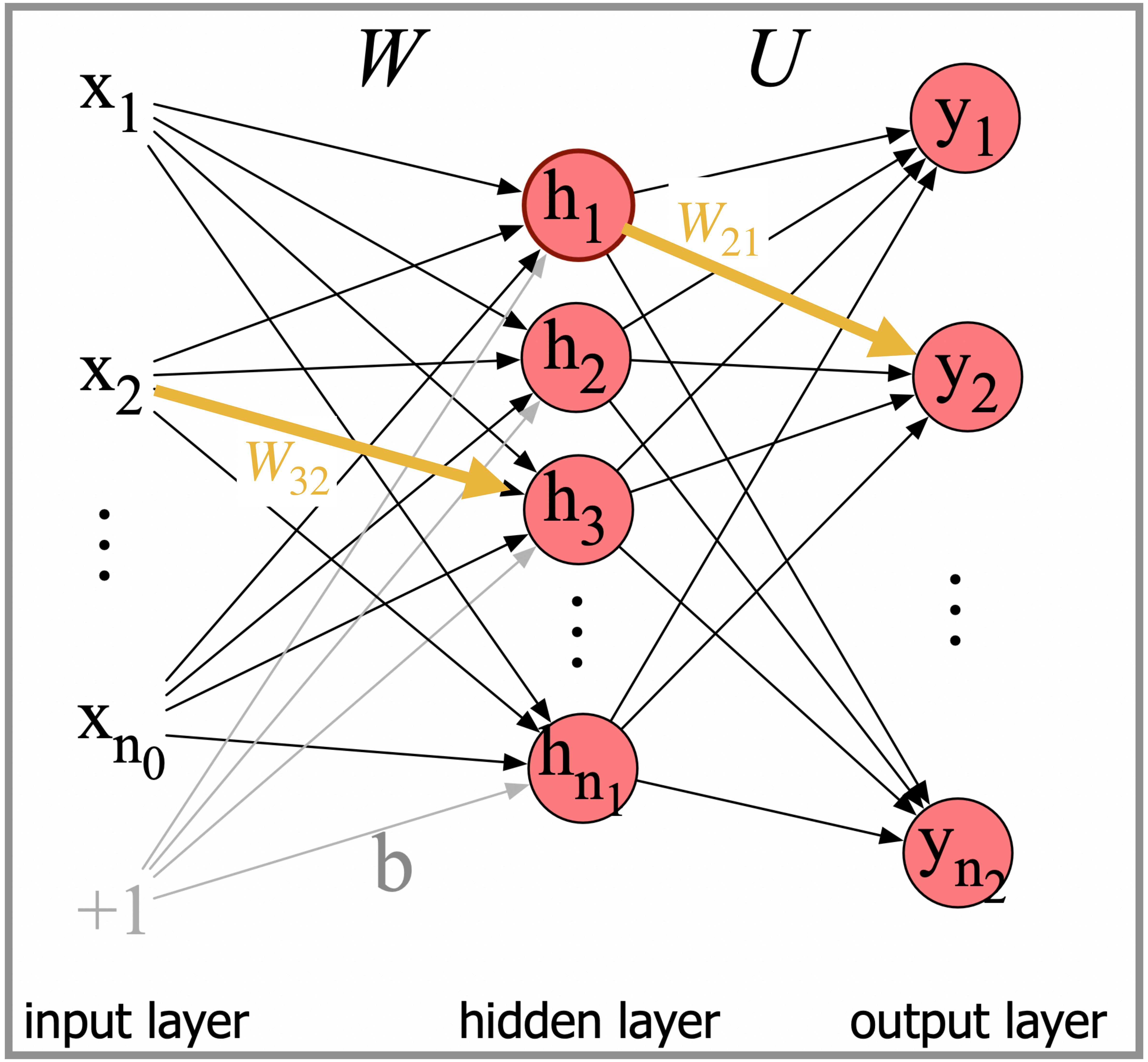
  $\mathbf{h} = f(\mathbf{Wx} + \mathbf{b})$, with $f \in \{\sigma, \tanh, \ldots\}$

- weight matrix:

  $\mathbf{U} \in \mathbb{R}^{n_y \times n_k}$

- prediction vector:

  $\mathbf{y} = g(\mathbf{Uh})$, with $g \in \{\sigma, \text{soft-max}, \ldots\}$



Feedforward neural network (n hidden layer)

- anchoring in input:

  $$\mathbf{a}^{[0]} = \mathbf{x} = [x_1, \ldots, x_{n_x}]^T$$

- activation at layer n:

  $$\mathbf{a}^{[n]} = f^{[n]}(\mathbf{W}^{[n]}\mathbf{a}^{[n-1]} + \mathbf{b}^{[n]})$$

  with $f^{[n]} \in \{\sigma, \text{tanh}, \ldots\}$ if n is a hidden layer, or

  with $f^{[n]} \in \{\sigma, \text{soft-max}, \ldots\}$ if n is the output layer