Medical Image Processing for Interventional Applications

Programming Exercise: Deep Learning

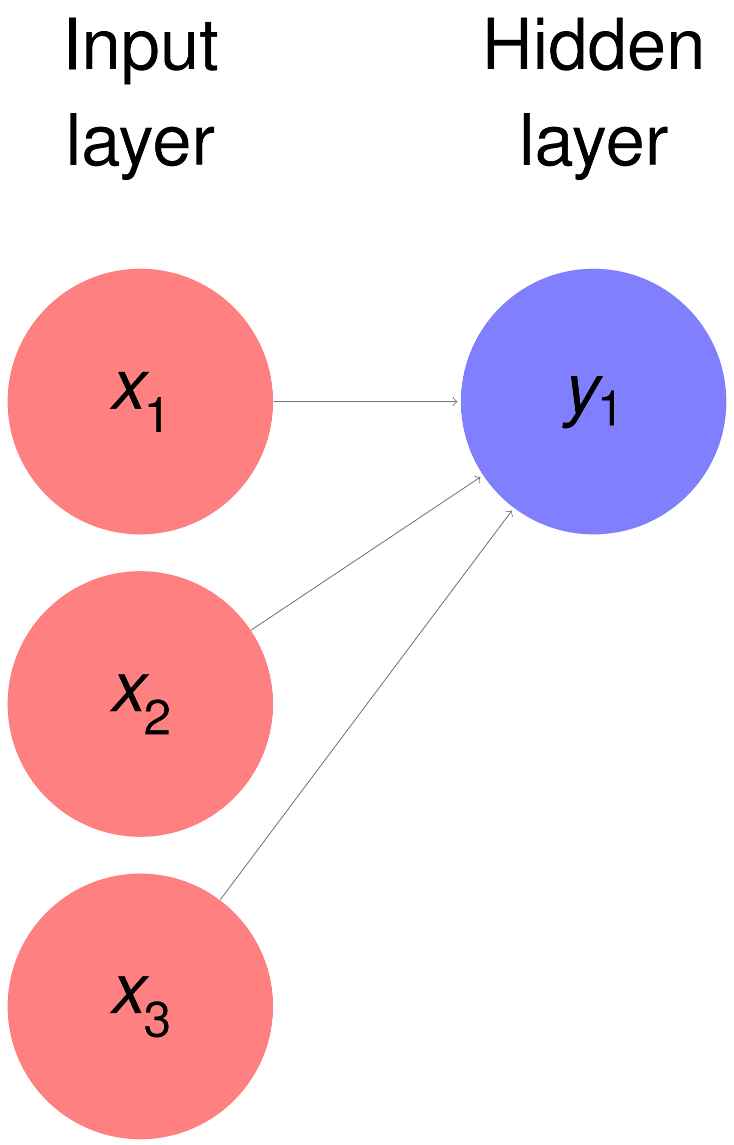
Online Course – Exercise P4

Siming Bayer, Daniel Stromer, Tobias Würfl, Frank Schebesch, Andreas Maier
Pattern Recognition Lab (CS 5)

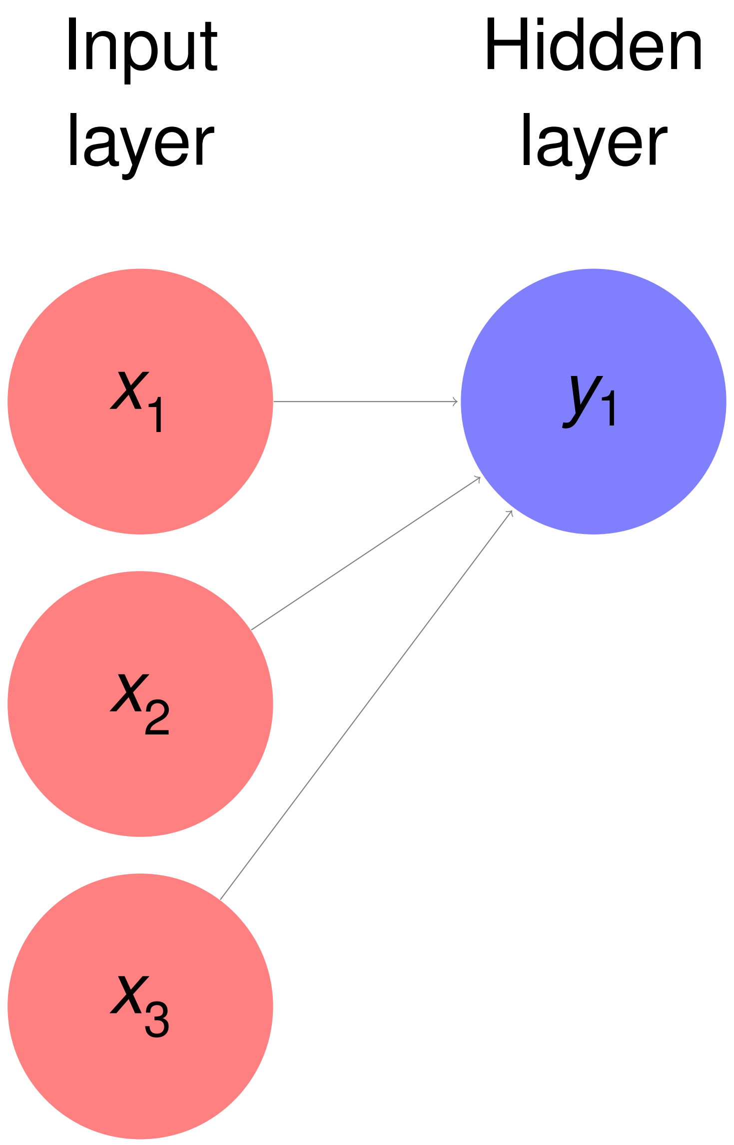
virtuelle hochschule bayern

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

Fully Connected Layer (Forward)



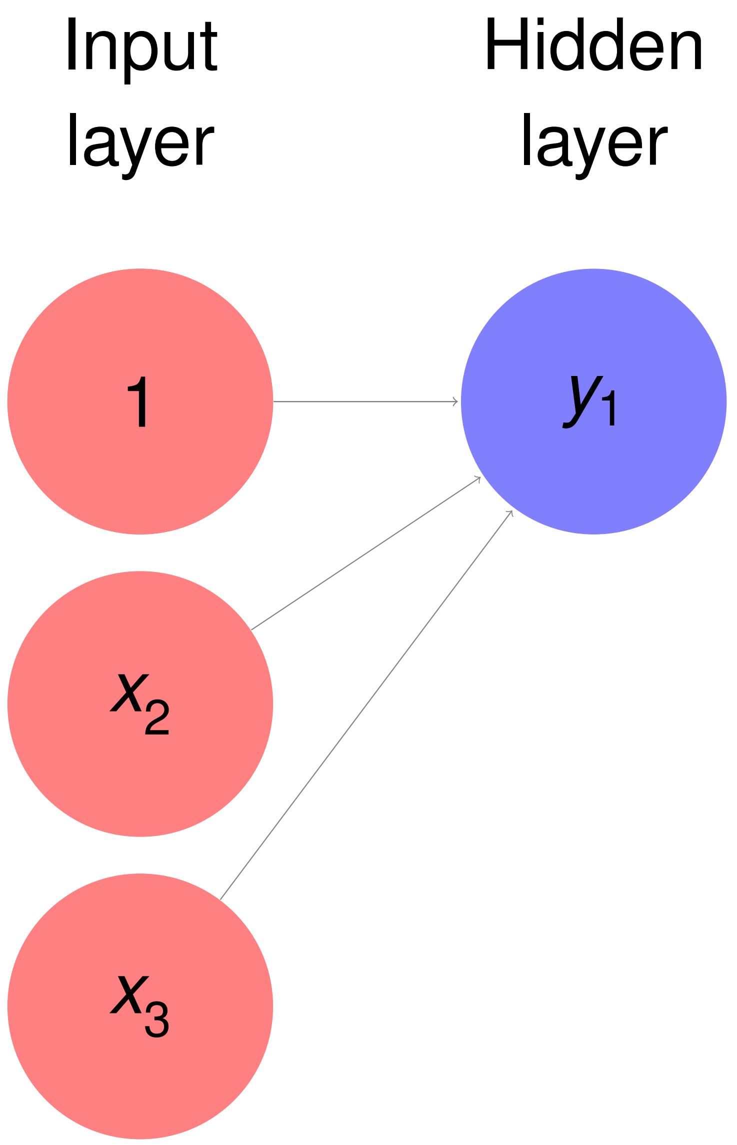
Fully Connected Layer (Forward)



$$\begin{pmatrix} w_1 \\ \vdots \\ w_n \end{pmatrix}^T \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} + w_{n+1} = y$$

$$\mathbf{w}^T \mathbf{x} = y$$

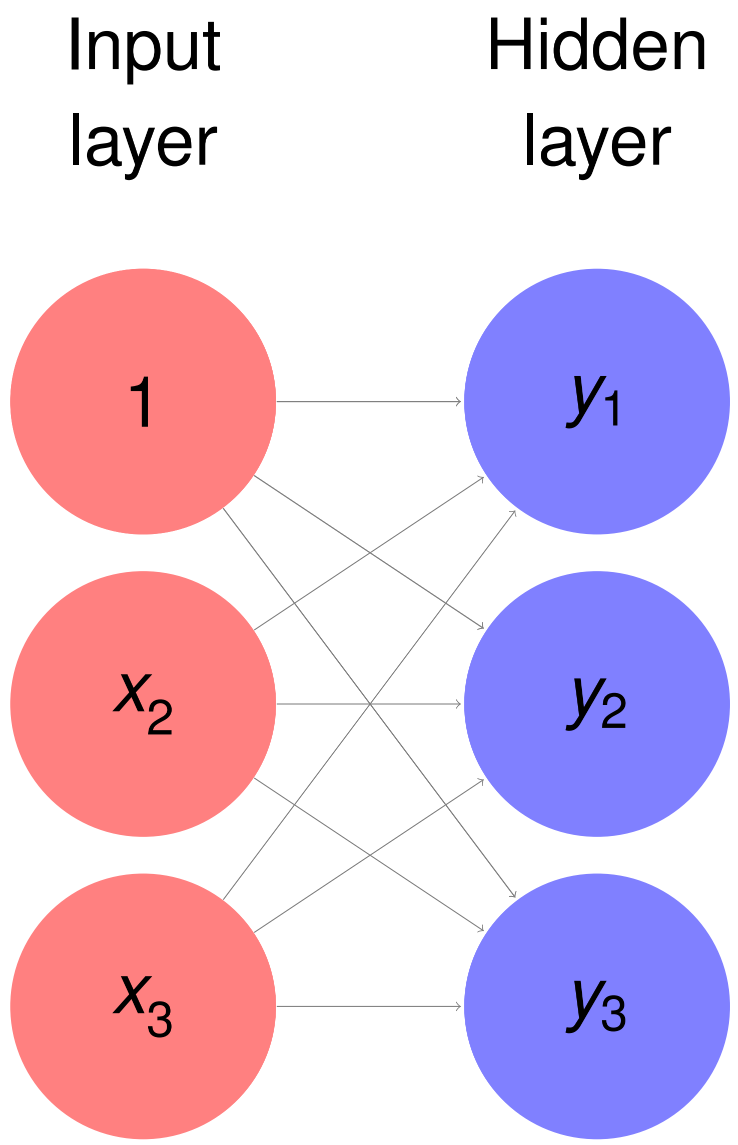
Fully Connected Layer (Forward)



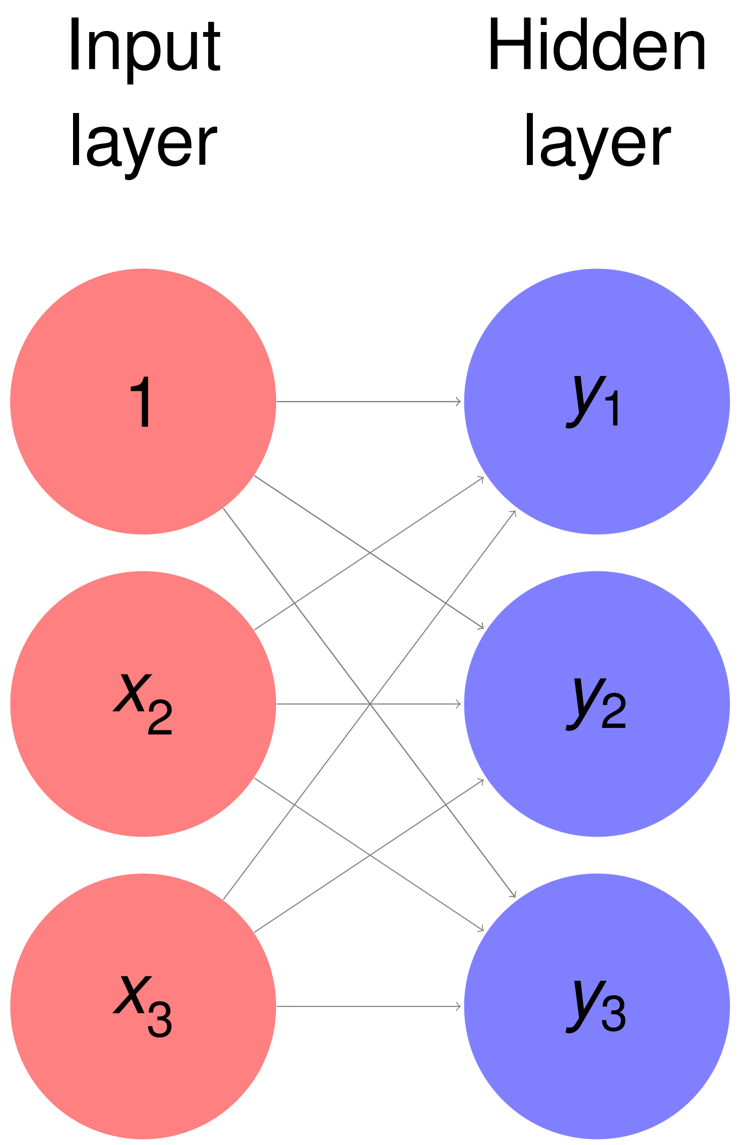
$$\begin{pmatrix} w_1 \\ \vdots \\ w_n \\ w_{n+1} \end{pmatrix}^T \begin{pmatrix} x_1 \\ \vdots \\ x_n \\ 1 \end{pmatrix} = y$$

$$\mathbf{w}^T \mathbf{x} = y$$

Fully Connected Layer (Forward)



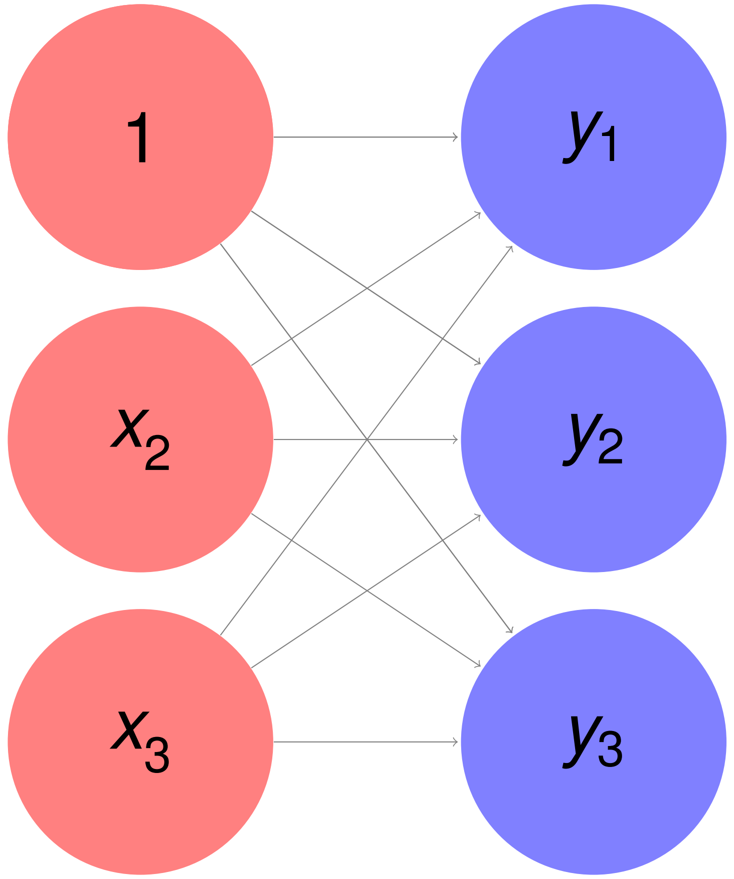
Fully Connected Layer (Forward)



$$\begin{pmatrix} w_{1,1} & \dots & w_{1,m} \\ \vdots & \ddots & \vdots \\ w_{n,1} & \dots & w_{n,m} \\ w_{n+1,1} & \dots & w_{n+1,m} \end{pmatrix}^T \begin{pmatrix} x_1 \\ \vdots \\ x_n \\ 1 \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix}$$

$$\mathbf{Wx} = \mathbf{y}$$

Fully Connected Layer (Forward)



$$\begin{pmatrix} w_{1,1} & \dots & w_{1,m} \\ \vdots & \ddots & \vdots \\ w_{n,1} & \dots & w_{n,m} \\ w_{n+1,1} & \dots & w_{n+1,m} \end{pmatrix}^T \begin{pmatrix} x_{1,1} & \dots & x_{1,b} \\ \vdots & \ddots & \vdots \\ x_{n,1} & \dots & x_{n,b} \\ 1 & \dots & 1 \end{pmatrix} = \dots \tag{1}$$

$$\mathbf{W}\mathbf{X} = \mathbf{Y}$$

Fully Connected Layer (Backward)

- Return gradient with respect to $\mathbf{X}$:

Fully Connected Layer (Backward)

- Return gradient with respect to $\mathbf{X}$:

$$\mathbf{E}_{n-1} = \mathbf{W}^{\mathsf{T}}\mathbf{E}_{n} \tag{2}$$

- $\mathbf{E_n}$: **error_tensor** passed downward

Fully Connected Layer (Backward)

- Return gradient with respect to **X**:

$$\mathbf{E}_{n-1} = \mathbf{W}^{\mathsf{T}}\mathbf{E}_n \tag{2}$$

- Update **W** using gradient with respect to **W**:

- $\mathbf{E_n}$: **error_tensor** passed downward

Fully Connected Layer (Backward)

- Return gradient with respect to $\mathbf{X}$:

$$\mathbf{E}_{n-1} = \mathbf{W}^{\mathbf{T}} \mathbf{E}_n \tag{2}$$

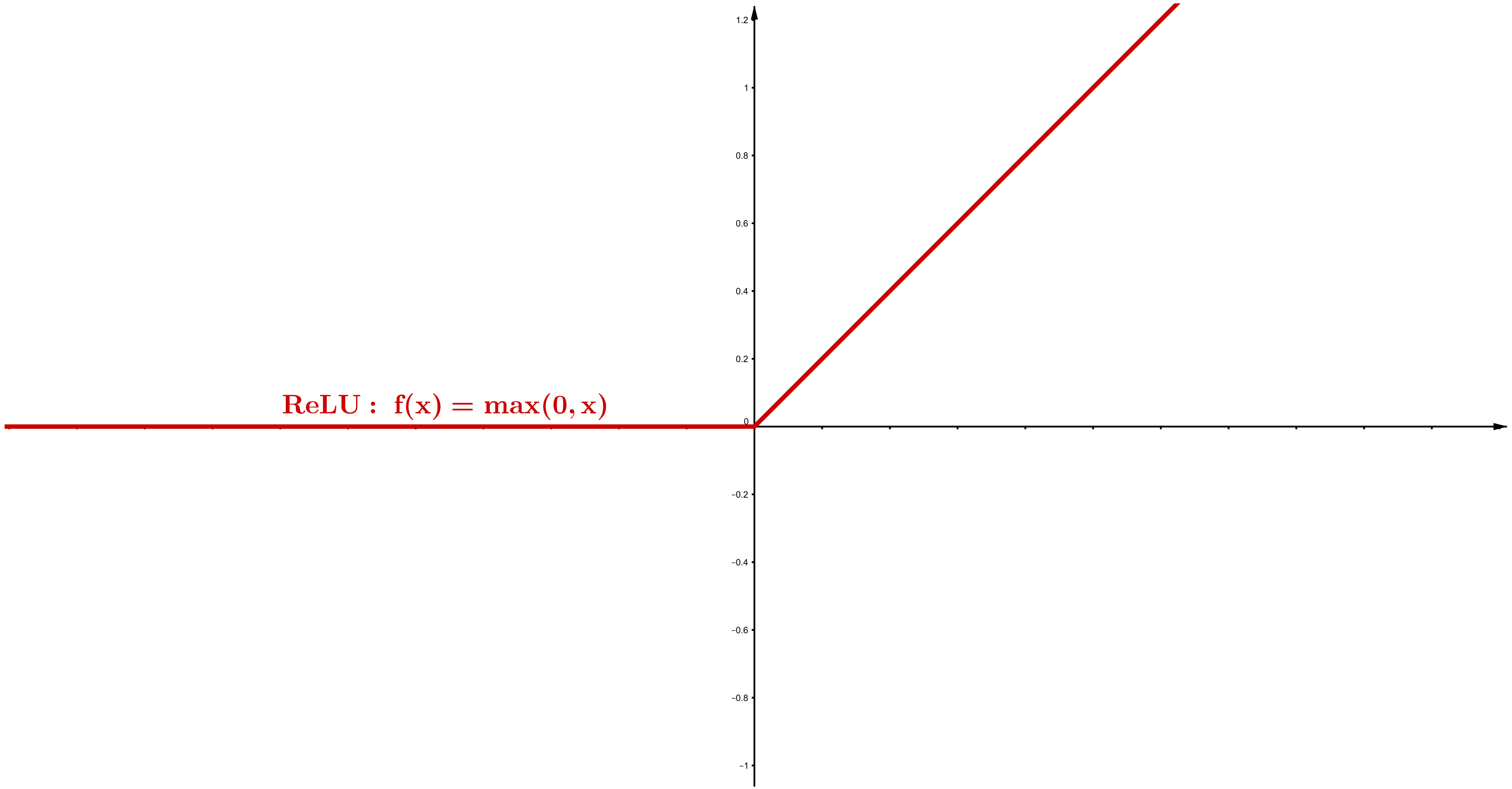
- Update $\mathbf{W}$ using gradient with respect to $\mathbf{W}$:

$$\mathbf{W}^{t+1} = \mathbf{W}^t - \delta \cdot \mathbf{E_n} \mathbf{X}^T \tag{3}$$

Note: Dynamic programming part of Backpropagation

- $\mathbf{E_n}$: **error_tensor** passed downward
- δ : learning rate **delta** individual to this layer

ReLU Activation Function (Forward)



ReLU Activation Function (Backward)

ReLU is not continuously differentiable!

ReLU Activation Function (Backward)

ReLU is not continuously differentiable!

$$e_{n-1} = \begin{cases} 0 & \text{if } x \leq 0 \\ e_n & \text{else} \end{cases} \tag{4}$$

Note: DP part of Backpropagation yet again

SoftMax Loss Function (Forward)

Labels as N-dimensional **one hot** vector $\mathbf{l}$: $\begin{pmatrix} \vdots \\ 1 \\ \vdots \end{pmatrix}$

SoftMax Loss Function (Forward)

Labels as N-dimensional **one hot** vector $\mathbf{l}$: $\begin{pmatrix} \vdots \\ 1 \\ \vdots \end{pmatrix}$

- Activation(Prediction) $\mathbf{y}$ for every element of the batch of size B:

$$y_i = \frac{\exp(x_i)}{\sum_{j=1}^{N} \exp(x_j)} \tag{5}$$

SoftMax Loss Function (Forward)

Labels as N-dimensional **one hot** vector $\mathbf{l}$: $\begin{pmatrix} \vdots \\ 1 \\ \vdots \end{pmatrix}$

- Activation(Prediction) $\mathbf{y}$ for every element of the batch of size B:

$$y_i = \frac{\exp(x_i)}{\sum_{j=1}^{N} \exp(x_j)} \tag{5}$$

- Loss:

$$loss = \sum_{b=1}^{B} -\log y_i \textbf{ where } l_i = 1 \tag{6}$$

SoftMax Loss Function (Backward)

For every element of the batch:

$$e_i = \begin{cases} y_i - 1 & \text{where } l_i = 1 \\ y_i & \text{else} \end{cases} \tag{7}$$