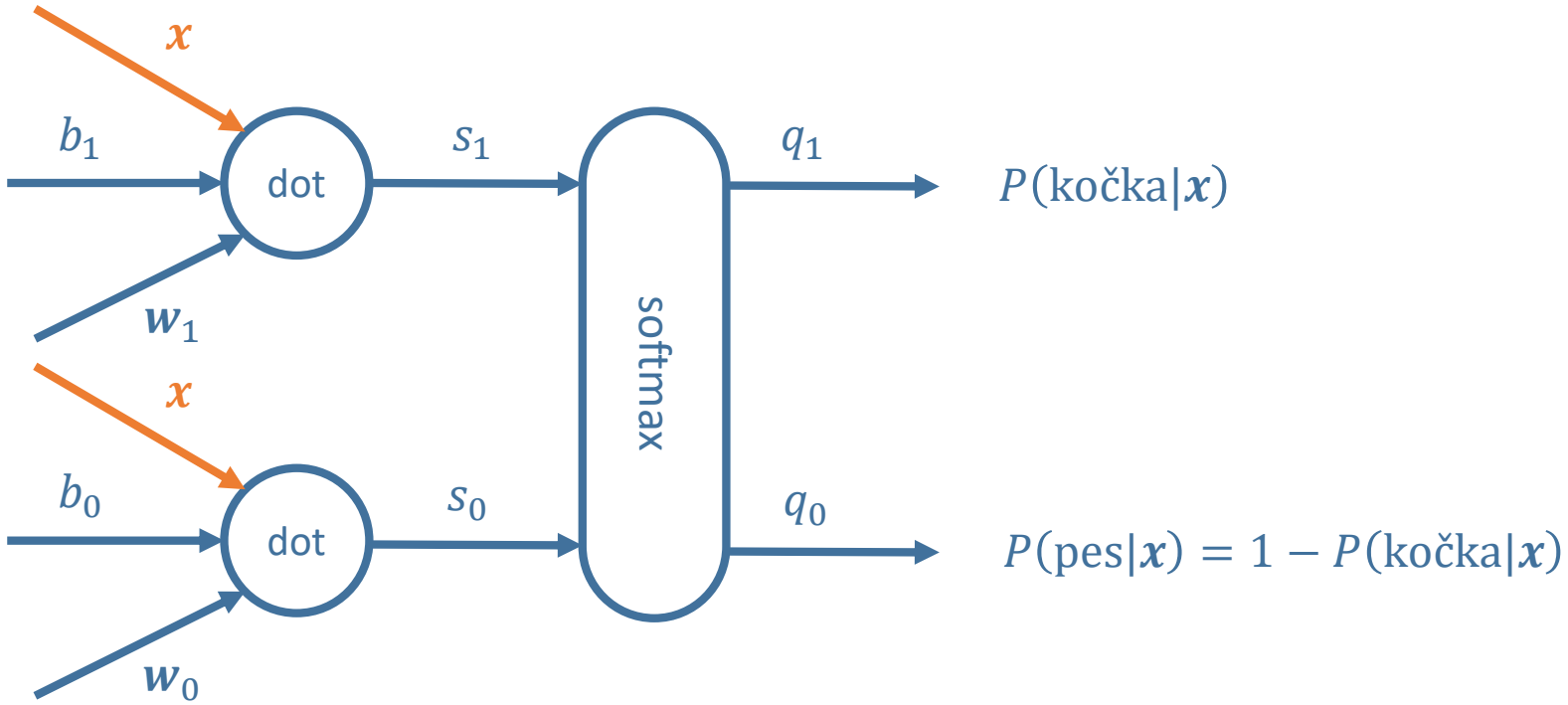
Aplikace neuronových sítí

Zpětná propagace

Softmax logistická regrese



$$P(\text{kočka}|\boldsymbol{x}) = \frac{e^{s_1}}{e^{s_0} + e^{s_1}}$$

$$P(\text{pes}|\boldsymbol{x}) = \frac{e^{s_0}}{e^{s_0} + e^{s_1}}$$

podmínka na součet pravděpodobností zachována:

$$P(\text{kočka}|\boldsymbol{x}) + P(\text{pes}|\boldsymbol{x}) = \frac{e^{s_1} + e^{s_0}}{e^{s_0} + e^{s_1}} = 1$$

Diferencovatelné výpočetní grafy

Řetízkové pravidlo (chain rule)

- Pro výpočet derivace složených funkcí formy $z = f(y) = f\big(g(x)\big)$ používáme

$$\frac{\mathrm{d}z}{\mathrm{d}x} = \frac{\mathrm{d}z}{\mathrm{d}y} \cdot \frac{\mathrm{d}y}{\mathrm{d}x}$$

- “derivace vnější krát derivace vnitřní”

Příklad na řetízkové pravidlo

$$z = \underbrace{(x_1 + ax_2)}_{g}^{2} \quad f$$

vnitřní funkce g:

$$y = x_1 + ax_2$$

$$\frac{\partial y}{\partial x_1} = 1 \qquad \frac{\partial y}{\partial x_2} = a$$

vnější funkce f:

$$z = y^2$$

$$\frac{\partial z}{\partial y} = 2y$$

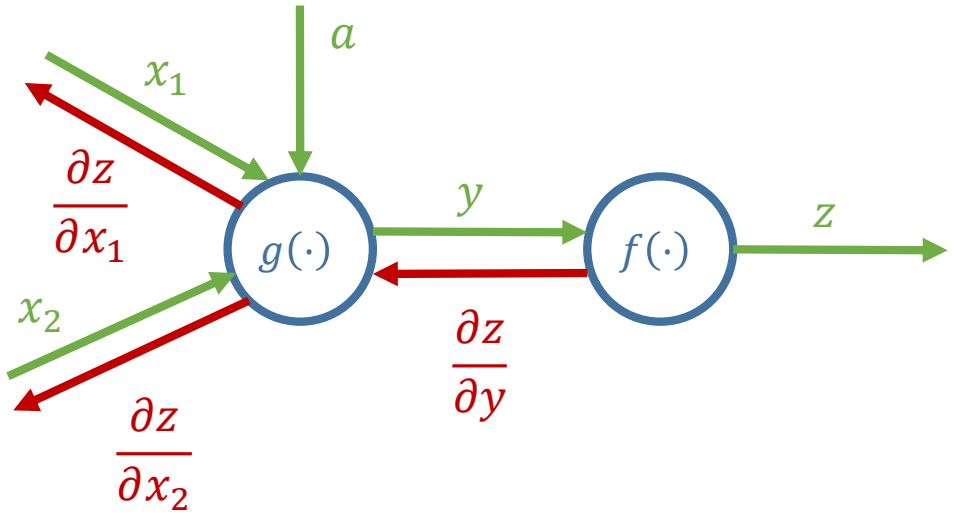
aplikace pravidla:

$$\frac{\partial z}{\partial x_1} = \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x_1} = 2y \cdot 1 = 2(x_1 + ax_2)$$

$$\frac{\partial z}{\partial x_2} = \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x_2} = 2y \cdot a = 2a(x_1 + ax_2)$$

Funkce $z = (x_1 + ax_2)^2$ jako graf

$$z = (x_1 + ax_2)^2$$

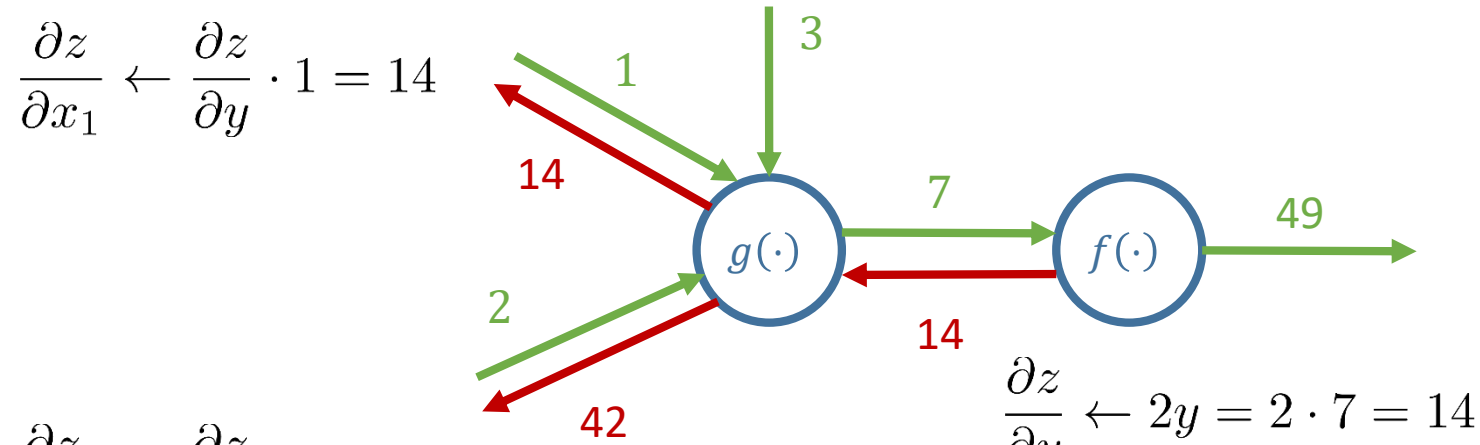


$$\frac{\partial z}{\partial x_1} = \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x_1} = 2y \cdot 1 = 2(x_1 + ax_2)$$

$$\frac{\partial z}{\partial x_2} = \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x_2} = 2y \cdot a = 2a(x_1 + ax_2)$$

Funkce $z = (x_1 + ax_2)^2$ jako graf

$$z = (x_1 + ax_2)^2$$



$$\frac{\partial z}{\partial x_1} \leftarrow \frac{\partial z}{\partial y} \cdot 1 = 14$$

$$\frac{\partial z}{\partial x_2} \leftarrow \frac{\partial z}{\partial y} \cdot a = 42$$

$$\frac{\partial z}{\partial y} \leftarrow 2y = 2 \cdot 7 = 14$$

Funkce $z = (x_1 + ax_2)^2$ jako graf podrobně

dopředný průchod

(1) $x_2' \leftarrow ax_2$

(2) $y \leftarrow x_1 + x_2'$

(3) $z \leftarrow y^2$

zpětný průchod

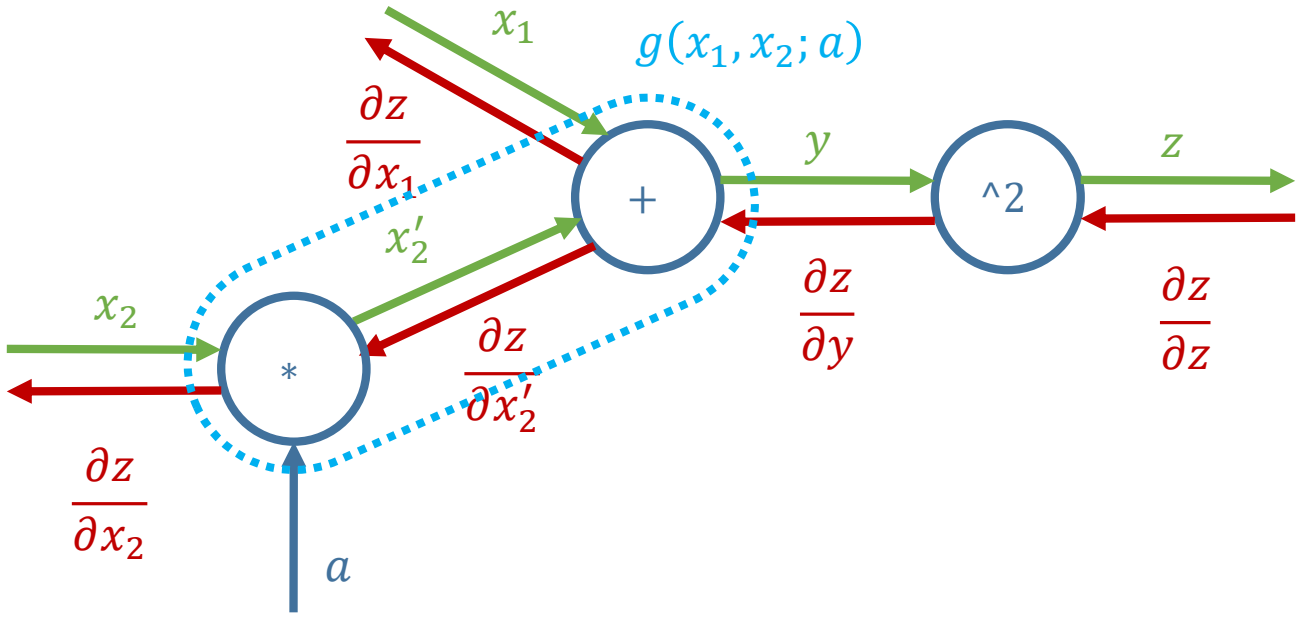
(0) $\frac{\partial z}{\partial z} = 1$

(1) $\frac{\partial z}{\partial y} \leftarrow \frac{\partial z}{\partial z} \cdot \frac{\partial z}{\partial y} = 2y$

(2) $\frac{\partial z}{\partial x_1} = \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x_1} = \frac{\partial z}{\partial y}$

$\frac{\partial z}{\partial x_2'} = \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x_2'} = \frac{\partial z}{\partial y}$

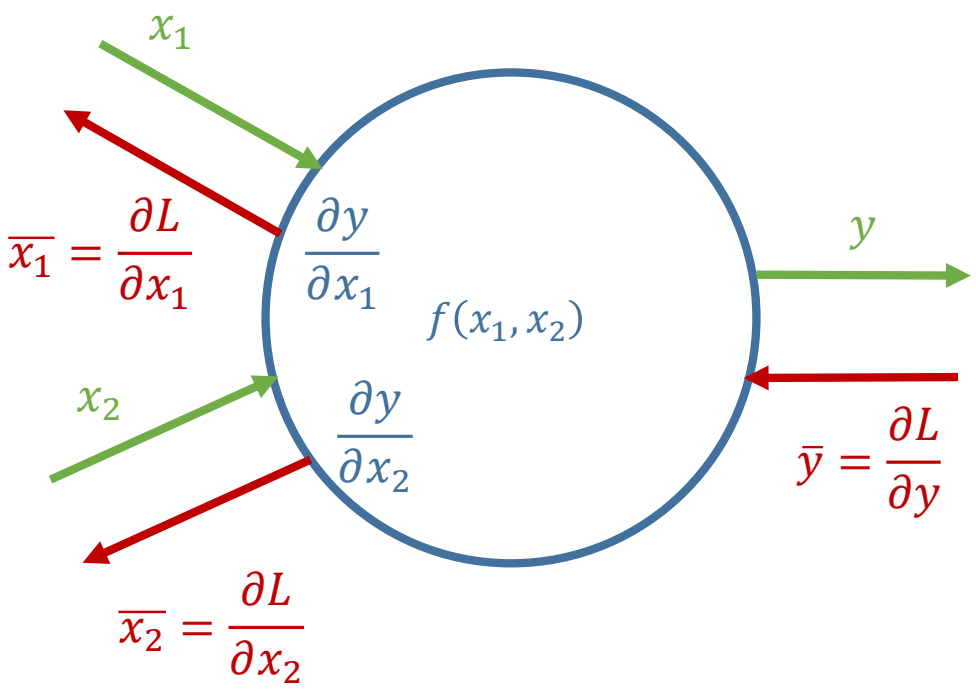
(3) $\frac{\partial z}{\partial x_2} = \frac{\partial z}{\partial x_2'} \cdot \frac{\partial x_2'}{\partial x_2} = \frac{\partial z}{\partial x_2'} \cdot a$



Uzel grafu a lokální gradient

- Na funkci lze nahlížet jako na orientovaný výpočetní graf
- Jednotlivé operace jsou uzly
- Hrany repezentují návaznosti vstupů a výstupů
- **Jakobián:**

$$J = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_D} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_C}{\partial x_1} & \cdots & \frac{\partial y_C}{\partial x_D} \end{bmatrix}$$



pruhová notace: https://people.maths.ox.ac.uk/gilesm/files/NA-08-01.pdf

Python kód funkce $z = (x_1 + ax_2)$ se zpětnou propagací

dopředný průchod

(1) $x_2' \leftarrow ax_2$

(2) $y \leftarrow x_1 + x_2'$

(3) $z \leftarrow y^2$

zpětný průchod

(0) $\frac{\partial z}{\partial z} = 1$

(1) $\frac{\partial z}{\partial y} \leftarrow \frac{\partial z}{\partial z} \cdot \frac{\partial z}{\partial y} = 2y$

(2) $\frac{\partial z}{\partial x_1} = \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x_1} = \frac{\partial z}{\partial y}$

$\frac{\partial z}{\partial x_2'} = \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x_2'} = \frac{\partial z}{\partial y}$

(3) $\frac{\partial z}{\partial x_2} = \frac{\partial z}{\partial x_2'} \cdot \frac{\partial x_2'}{\partial x_2} = \frac{\partial z}{\partial x_2'} \cdot a$

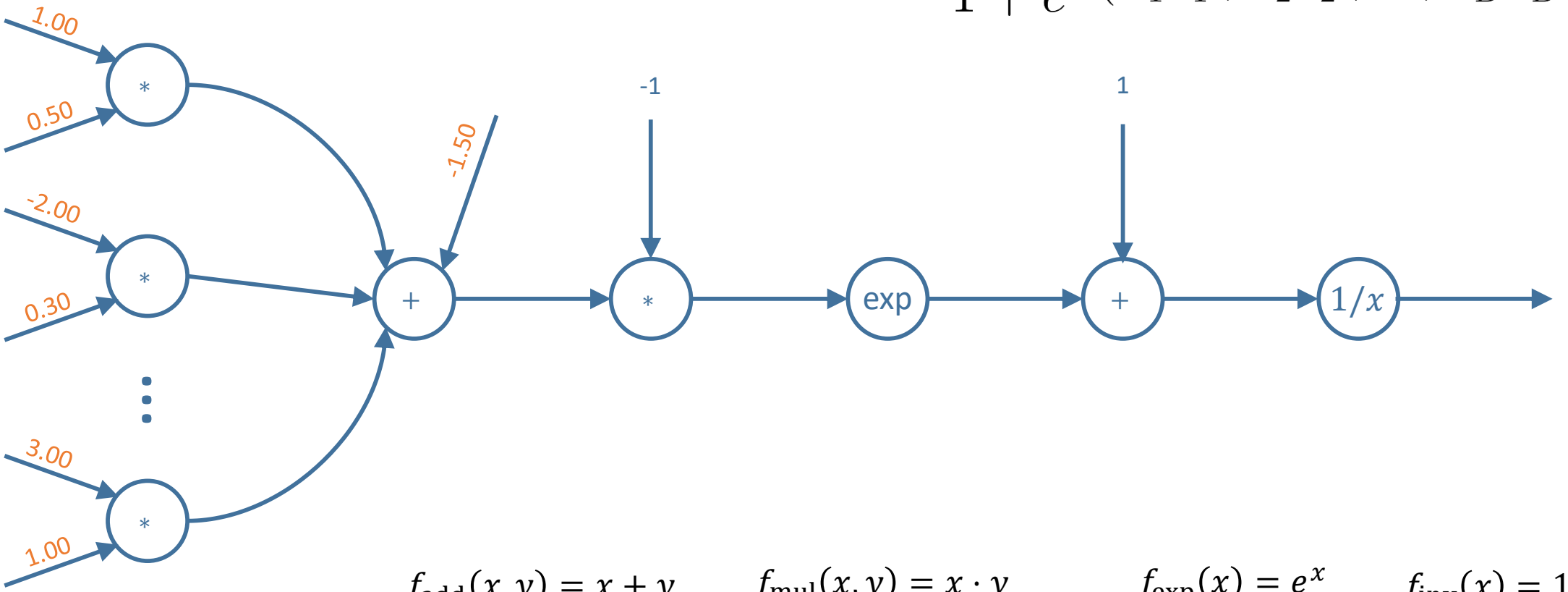
```python
def forward(x1, x2, a):
    # (1) forward
    x2_ = a * x2

    # (2) forward
    y = x1 + x2_

    # (3) forward
    z = y * y

    cache = x2, a, y
    return z, cache
```

```python
def backward(dz, cache):
    x2, a, y = cache

    # (3) backward
    dy = dz * 2 * y
    # (2) backward
    dx1 = dy
    dx2_ = dy
    # (1) backward
    dx2 = dx2_ * a
    da = dx2_ * x2
    return dx1, dx2, da
```

```
>>> x1, x2, a = 1, 2, 3
>>> z, cache = forward(x1, x2, a)
>>> dx1, dx2, da = backward(1.0, cache)
>>> dx1, dx2, da
(14.0, 42.0, 28.0)
```

Binární logistická regrese se sigmoidem jako graf

$$q = \frac{1}{1 + e^{-(w_1 x_1 + w_2 x_2 + \ldots + w_D x_D + b)}}$$



$f_{\text{add}}(x, y) = x + y$

$\frac{\partial f_{\text{add}}}{\partial x} = \frac{\partial f_{\text{add}}}{\partial y} = 1$

$f_{\text{mul}}(x, y) = x \cdot y$

$\frac{\partial f_{\text{mul}}}{\partial x} = y, \frac{\partial f_{\text{mul}}}{\partial y} = x$

$f_{\text{exp}}(x) = e^x$

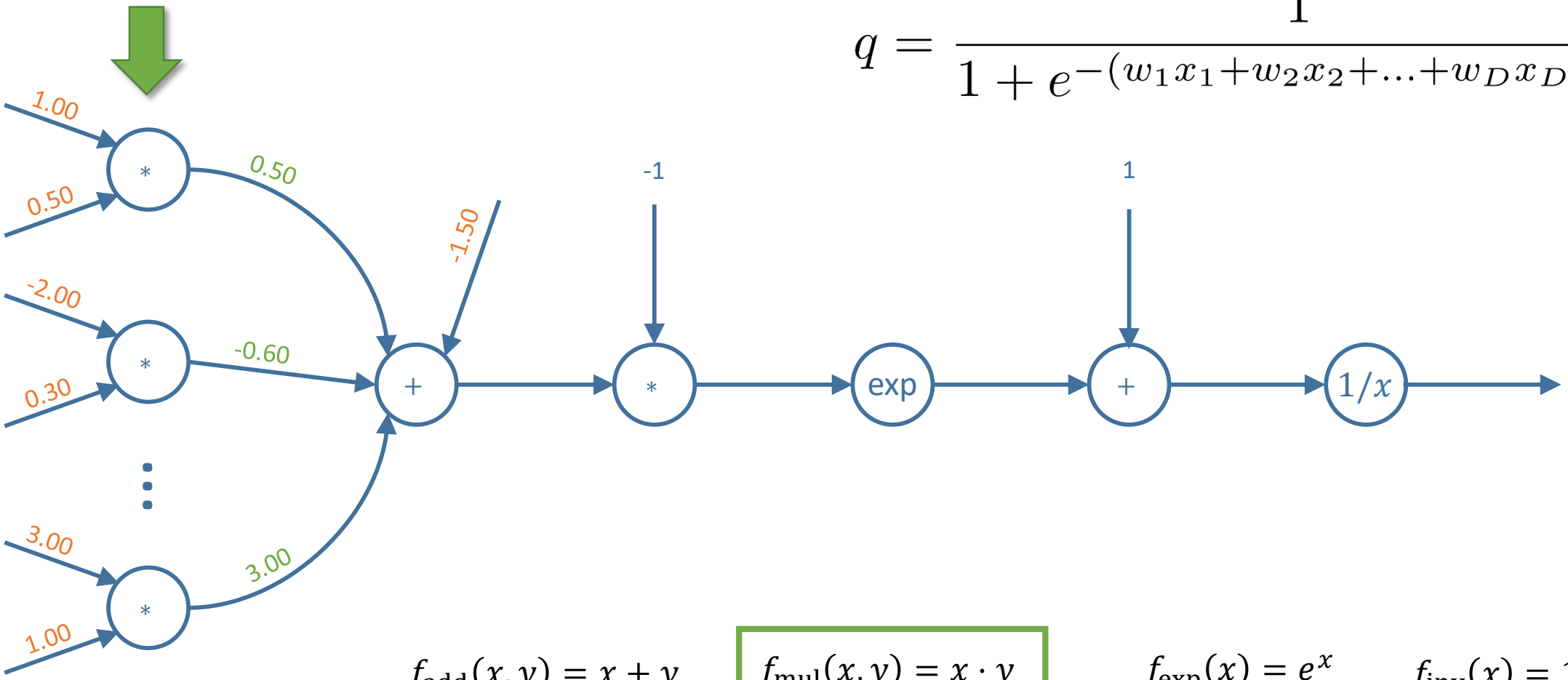
$\frac{\partial f_{\text{exp}}}{\partial x} = e^x$

$f_{\text{inv}}(x) = 1/x$

$\frac{\partial f_{\text{inv}}}{\partial x} = -1/x^2$

Binární logistická regrese se sigmoidem jako graf

$$q = \frac{1}{1 + e^{-(w_1 x_1 + w_2 x_2 + \ldots + w_D x_D + b)}}$$



$f_{\text{add}}(x, y) = x + y$

$f_{\text{mul}}(x, y) = x \cdot y$

$f_{\text{exp}}(x) = e^x$

$f_{\text{inv}}(x) = 1/x$

$\frac{\partial f_{\text{add}}}{\partial x} = \frac{\partial f_{\text{add}}}{\partial y} = 1$

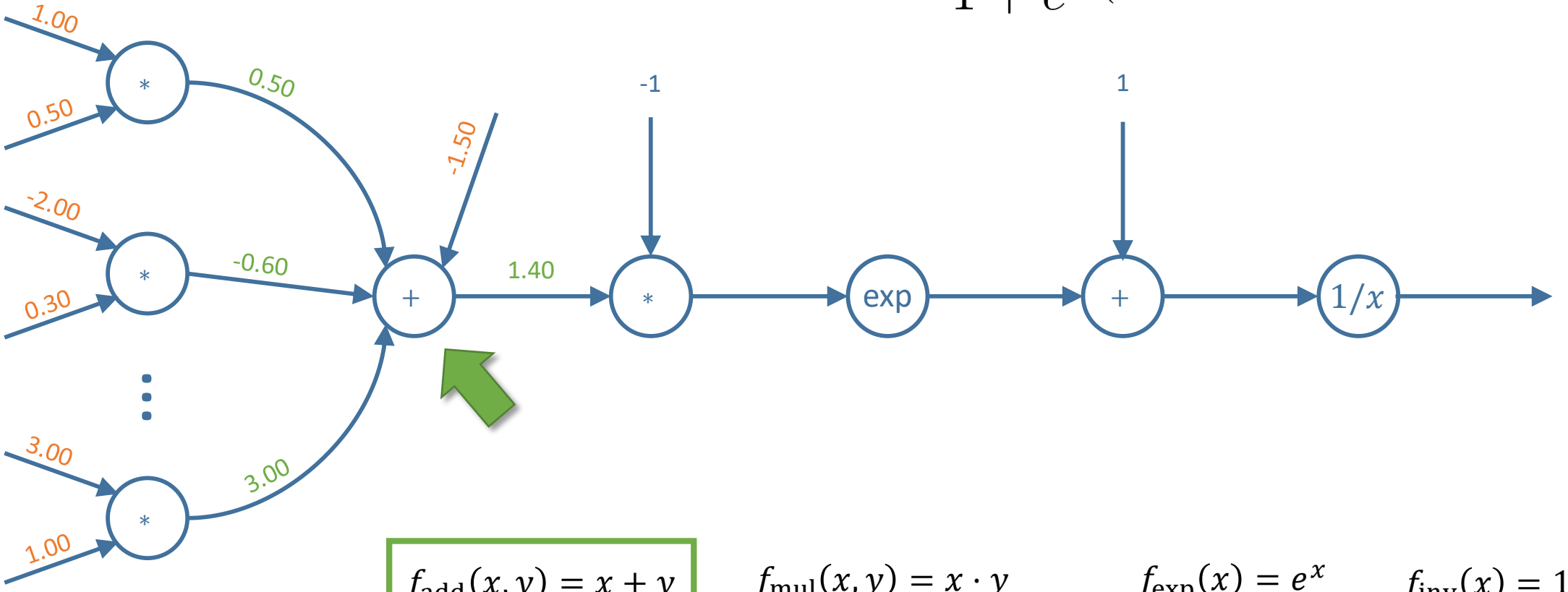
$\frac{\partial f_{\text{mul}}}{\partial x} = y, \frac{\partial f_{\text{mul}}}{\partial y} = x$

$\frac{\partial f_{\text{exp}}}{\partial x} = e^x$

$\frac{\partial f_{\text{inv}}}{\partial x} = -1/x^2$

Binární logistická regrese se sigmoidem jako graf

$$q = \frac{1}{1 + e^{-(w_1 x_1 + w_2 x_2 + \ldots + w_D x_D + b)}}$$

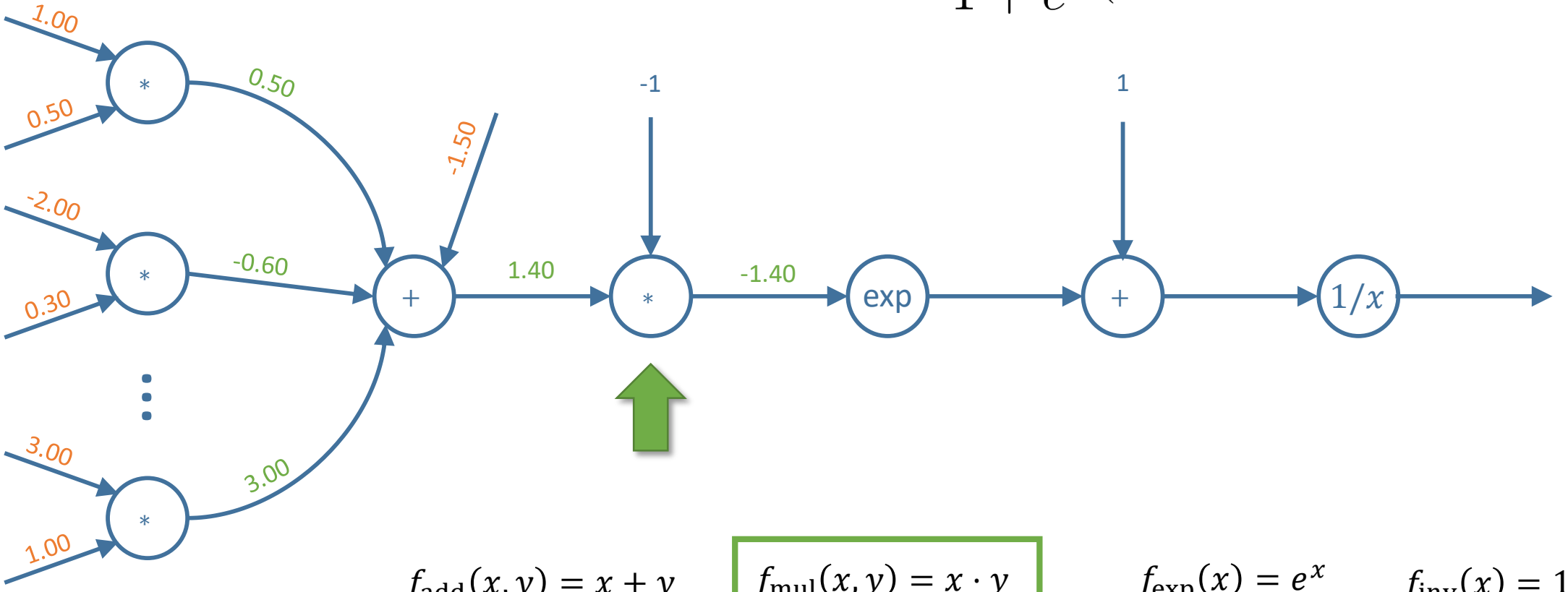


$$f_{\text{add}}(x, y) = x + y \qquad f_{\text{mul}}(x, y) = x \cdot y \qquad f_{\text{exp}}(x) = e^x \qquad f_{\text{inv}}(x) = 1/x$$

$$\frac{\partial f_{\text{add}}}{\partial x} = \frac{\partial f_{\text{add}}}{\partial y} = 1 \qquad \frac{\partial f_{\text{mul}}}{\partial x} = y, \frac{\partial f_{\text{mul}}}{\partial y} = x \qquad \frac{\partial f_{\text{exp}}}{\partial x} = e^x \qquad \frac{\partial f_{\text{inv}}}{\partial x} = -1/x^2$$

Binární logistická regrese se sigmoidem jako graf

$$q = \frac{1}{1 + e^{-(w_1 x_1 + w_2 x_2 + \ldots + w_D x_D + b)}}$$



$f_{\text{add}}(x, y) = x + y$

$f_{\text{mul}}(x, y) = x \cdot y$

$f_{\text{exp}}(x) = e^x$

$f_{\text{inv}}(x) = 1/x$

$\frac{\partial f_{\text{add}}}{\partial x} = \frac{\partial f_{\text{add}}}{\partial y} = 1$

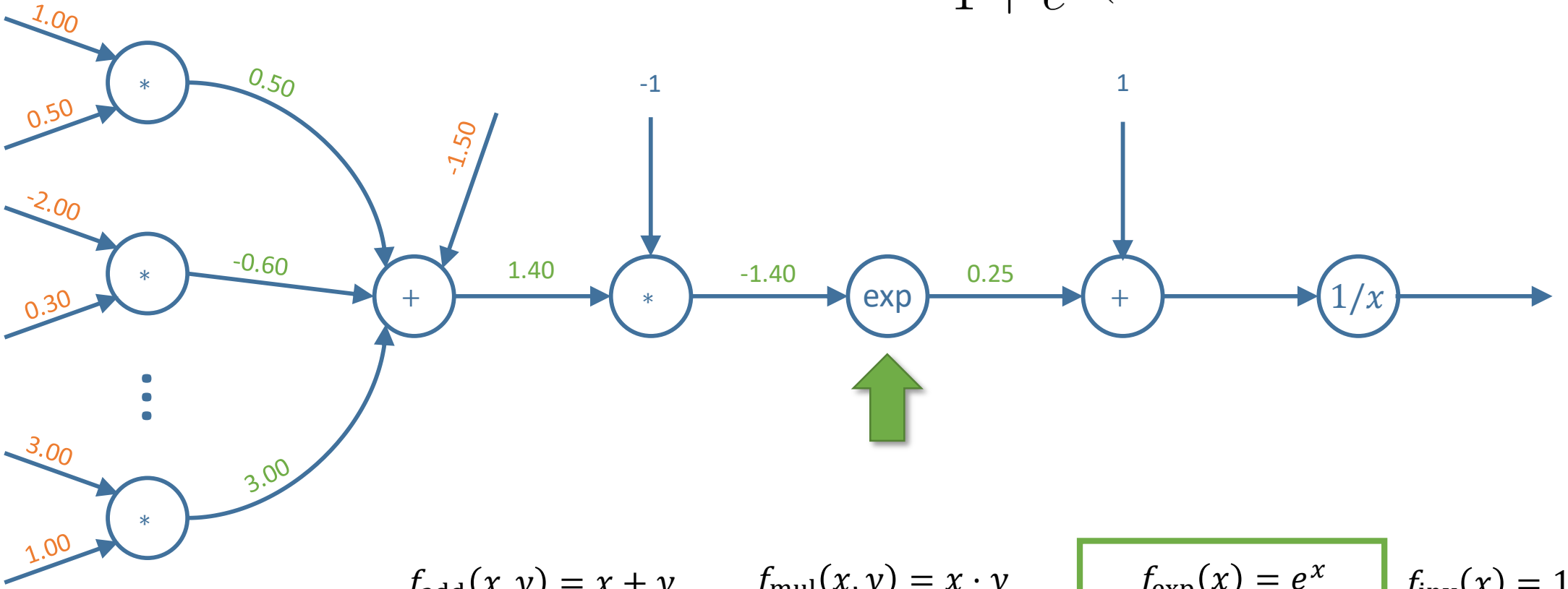
$\frac{\partial f_{\text{mul}}}{\partial x} = y, \frac{\partial f_{\text{mul}}}{\partial y} = x$

$\frac{\partial f_{\text{exp}}}{\partial x} = e^x$

$\frac{\partial f_{\text{inv}}}{\partial x} = -1/x^2$

Binární logistická regrese se sigmoidem jako graf

$$q = \frac{1}{1 + e^{-(w_1 x_1 + w_2 x_2 + \ldots + w_D x_D + b)}}$$

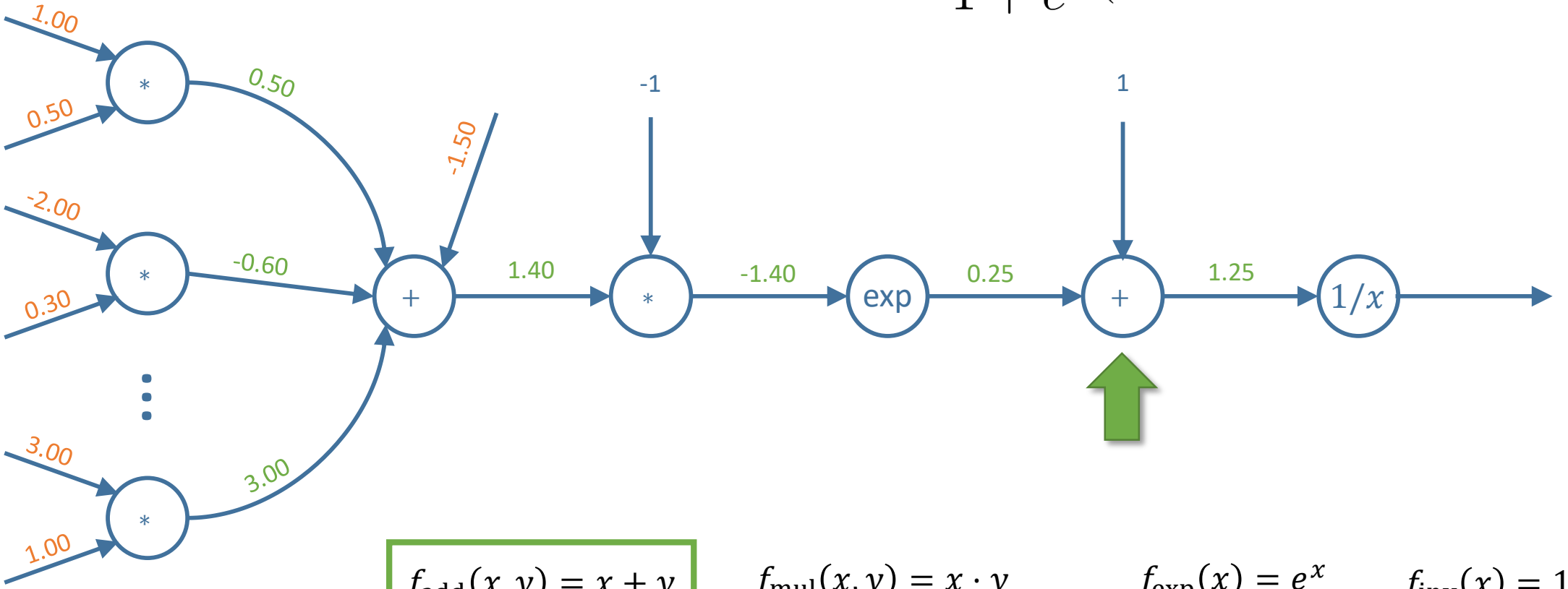


$f_{\text{add}}(x, y) = x + y$ $\quad$ $f_{\text{mul}}(x, y) = x \cdot y$ $\quad$ $f_{\text{exp}}(x) = e^x$ $\quad$ $f_{\text{inv}}(x) = 1/x$

$\frac{\partial f_{\text{add}}}{\partial x} = \frac{\partial f_{\text{add}}}{\partial y} = 1$ $\quad$ $\frac{\partial f_{\text{mul}}}{\partial x} = y, \frac{\partial f_{\text{mul}}}{\partial y} = x$ $\quad$ $\frac{\partial f_{\text{exp}}}{\partial x} = e^x$ $\quad$ $\frac{\partial f_{\text{inv}}}{\partial x} = -1/x^2$

Binární logistická regrese se sigmoidem jako graf

$$q = \frac{1}{1 + e^{-(w_1 x_1 + w_2 x_2 + \ldots + w_D x_D + b)}}$$



$$f_{\text{add}}(x, y) = x + y$$

$$f_{\text{mul}}(x, y) = x \cdot y$$

$$f_{\exp}(x) = e^x$$

$$f_{\text{inv}}(x) = 1/x$$

$$\frac{\partial f_{\text{add}}}{\partial x} = \frac{\partial f_{\text{add}}}{\partial y} = 1$$

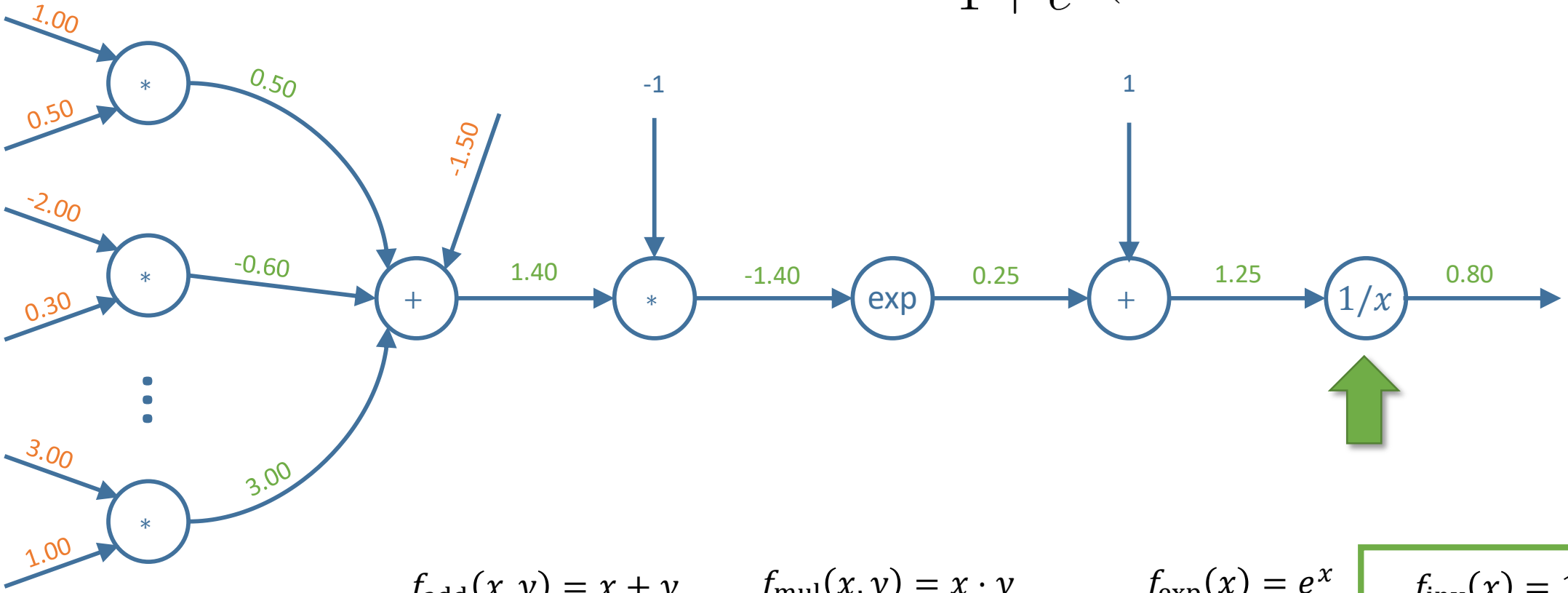
$$\frac{\partial f_{\text{mul}}}{\partial x} = y, \frac{\partial f_{\text{mul}}}{\partial y} = x$$

$$\frac{\partial f_{\exp}}{\partial x} = e^x$$

$$\frac{\partial f_{\text{inv}}}{\partial x} = -1/x^2$$

Binární logistická regrese se sigmoidem jako graf

$$q = \frac{1}{1 + e^{-(w_1 x_1 + w_2 x_2 + \ldots + w_D x_D + b)}}$$



$f_{\text{add}}(x, y) = x + y$

$f_{\text{mul}}(x, y) = x \cdot y$

$f_{\exp}(x) = e^x$

$f_{\text{inv}}(x) = 1/x$

$\frac{\partial f_{\text{add}}}{\partial x} = \frac{\partial f_{\text{add}}}{\partial y} = 1$

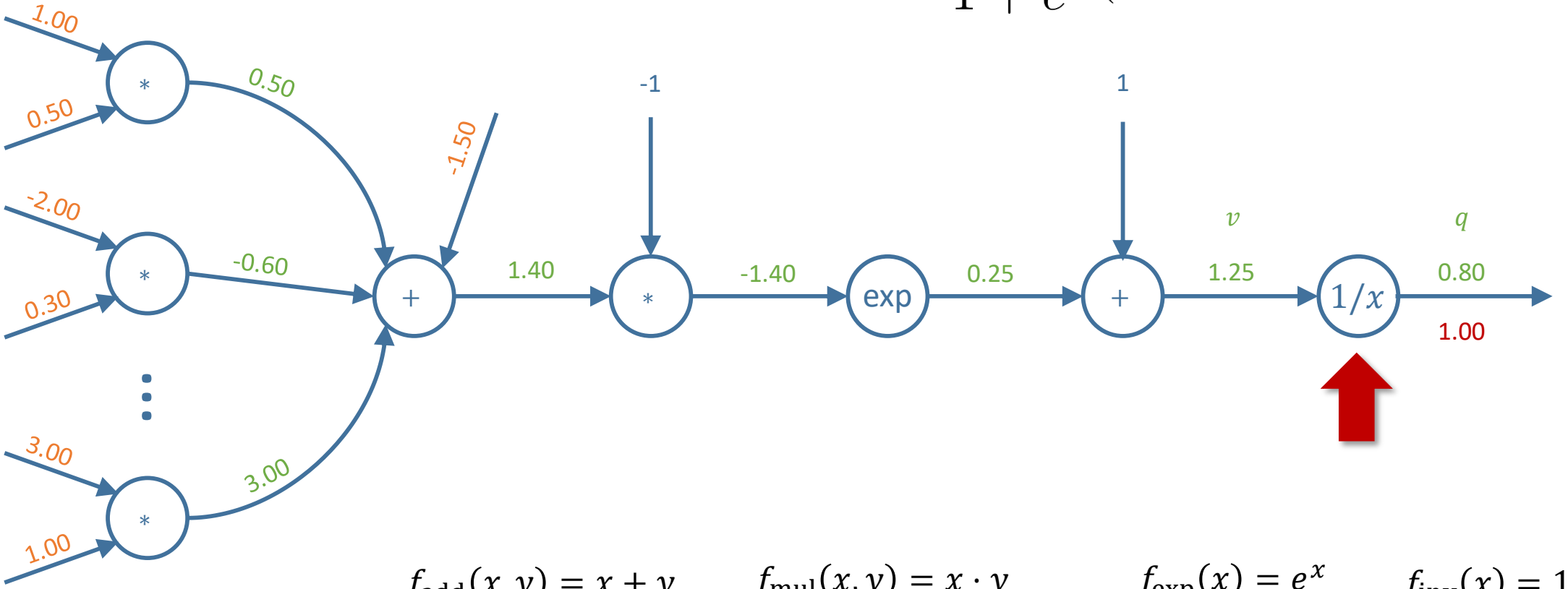
$\frac{\partial f_{\text{mul}}}{\partial x} = y, \frac{\partial f_{\text{mul}}}{\partial y} = x$

$\frac{\partial f_{\exp}}{\partial x} = e^x$

$\frac{\partial f_{\text{inv}}}{\partial x} = -1/x^2$

Binární logistická regrese se sigmoidem jako graf

$$q = \frac{1}{1 + e^{-(w_1 x_1 + w_2 x_2 + \ldots + w_D x_D + b)}}$$

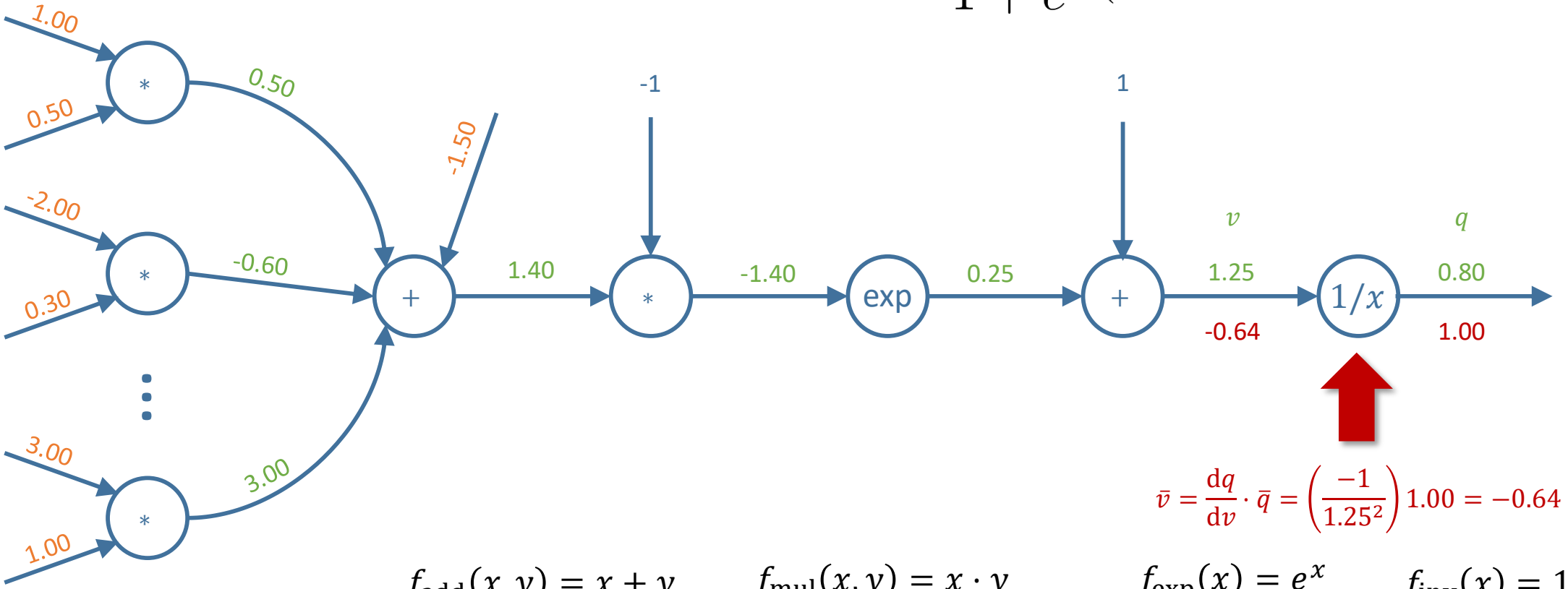


$$f_{\text{add}}(x, y) = x + y \qquad f_{\text{mul}}(x, y) = x \cdot y \qquad f_{\text{exp}}(x) = e^x \qquad f_{\text{inv}}(x) = 1/x$$

$$\frac{\partial f_{\text{add}}}{\partial x} = \frac{\partial f_{\text{add}}}{\partial y} = 1 \qquad \frac{\partial f_{\text{mul}}}{\partial x} = y, \frac{\partial f_{\text{mul}}}{\partial y} = x \qquad \frac{\partial f_{\text{exp}}}{\partial x} = e^x \qquad \frac{\partial f_{\text{inv}}}{\partial x} = -1/x^2$$

Binární logistická regrese se sigmoidem jako graf

$$q = \frac{1}{1 + e^{-(w_1 x_1 + w_2 x_2 + \ldots + w_D x_D + b)}}$$



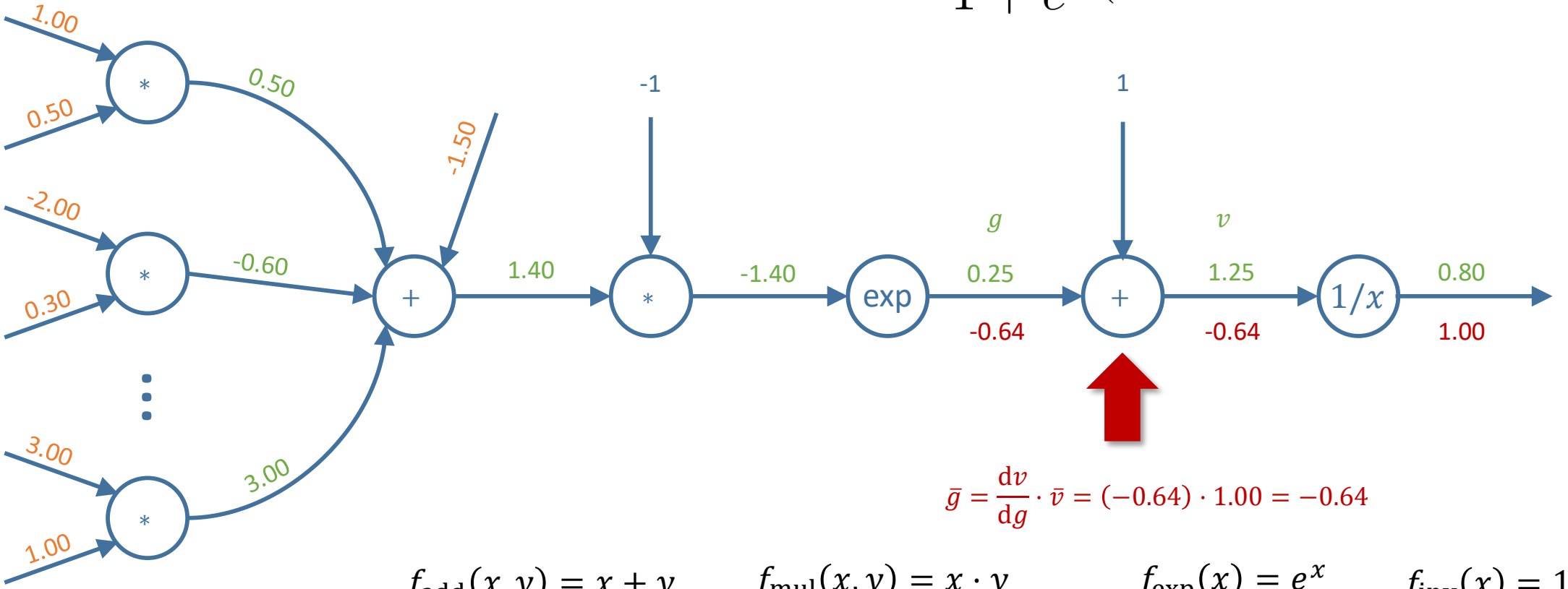
$$\bar{v} = \frac{\mathrm{d}q}{\mathrm{d}v} \cdot \bar{q} = \left(\frac{-1}{1.25^2}\right) 1.00 = -0.64$$

$$f_{\text{add}}(x, y) = x + y \qquad f_{\text{mul}}(x, y) = x \cdot y \qquad f_{\text{exp}}(x) = e^x \qquad f_{\text{inv}}(x) = 1/x$$

$$\frac{\partial f_{\text{add}}}{\partial x} = \frac{\partial f_{\text{add}}}{\partial y} = 1 \qquad \frac{\partial f_{\text{mul}}}{\partial x} = y, \frac{\partial f_{\text{mul}}}{\partial y} = x \qquad \frac{\partial f_{\text{exp}}}{\partial x} = e^x \qquad \boxed{\frac{\partial f_{\text{inv}}}{\partial x} = -1/x^2}$$

Binární logistická regrese se sigmoidem jako graf

$$q = \frac{1}{1 + e^{-(w_1 x_1 + w_2 x_2 + \ldots + w_D x_D + b)}}$$



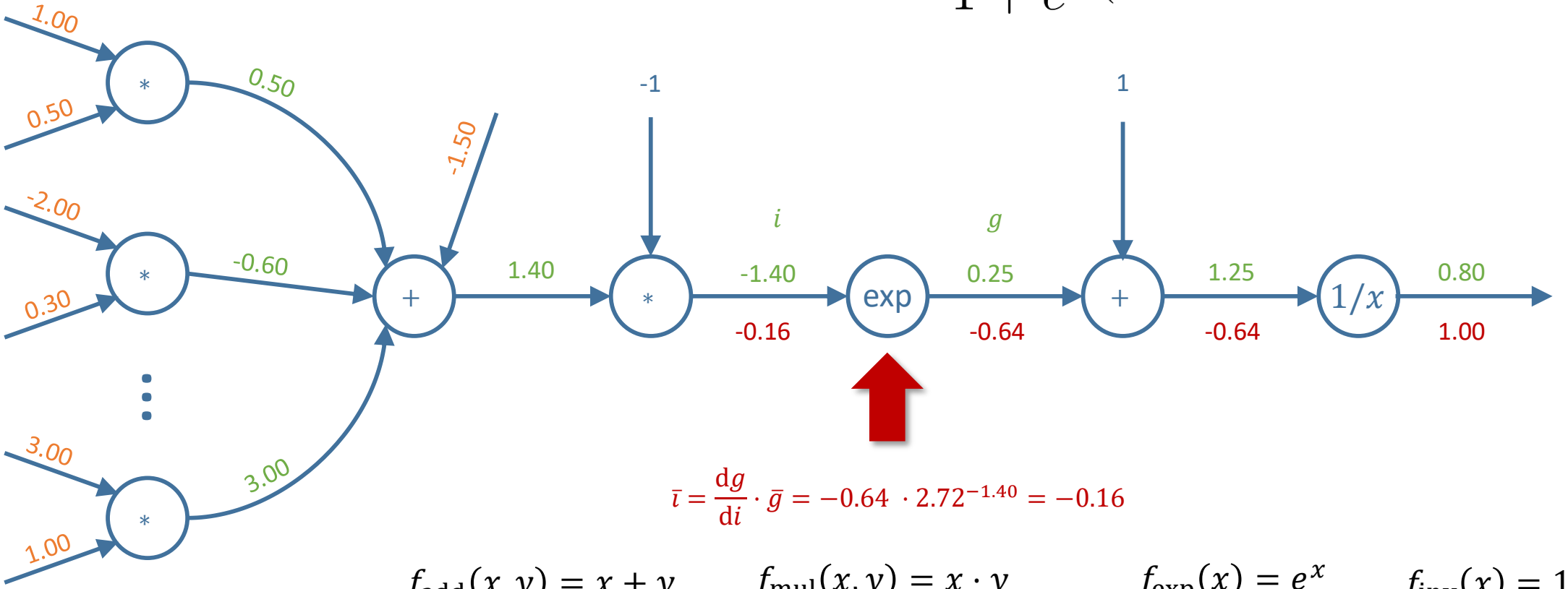
$$\bar{g} = \frac{\mathrm{d}v}{\mathrm{d}g} \cdot \bar{v} = (-0.64) \cdot 1.00 = -0.64$$

$f_{\text{add}}(x, y) = x + y$ $\quad f_{\text{mul}}(x, y) = x \cdot y$ $\quad f_{\exp}(x) = e^x$ $\quad f_{\text{inv}}(x) = 1/x$

$$\frac{\partial f_{\text{add}}}{\partial x} = \frac{\partial f_{\text{add}}}{\partial y} = 1 \qquad \frac{\partial f_{\text{mul}}}{\partial x} = y, \frac{\partial f_{\text{mul}}}{\partial y} = x \qquad \frac{\partial f_{\exp}}{\partial x} = e^x \qquad \frac{\partial f_{\text{inv}}}{\partial x} = -1/x^2$$

Binární logistická regrese se sigmoidem jako graf

$$q = \frac{1}{1 + e^{-(w_1 x_1 + w_2 x_2 + \ldots + w_D x_D + b)}}$$



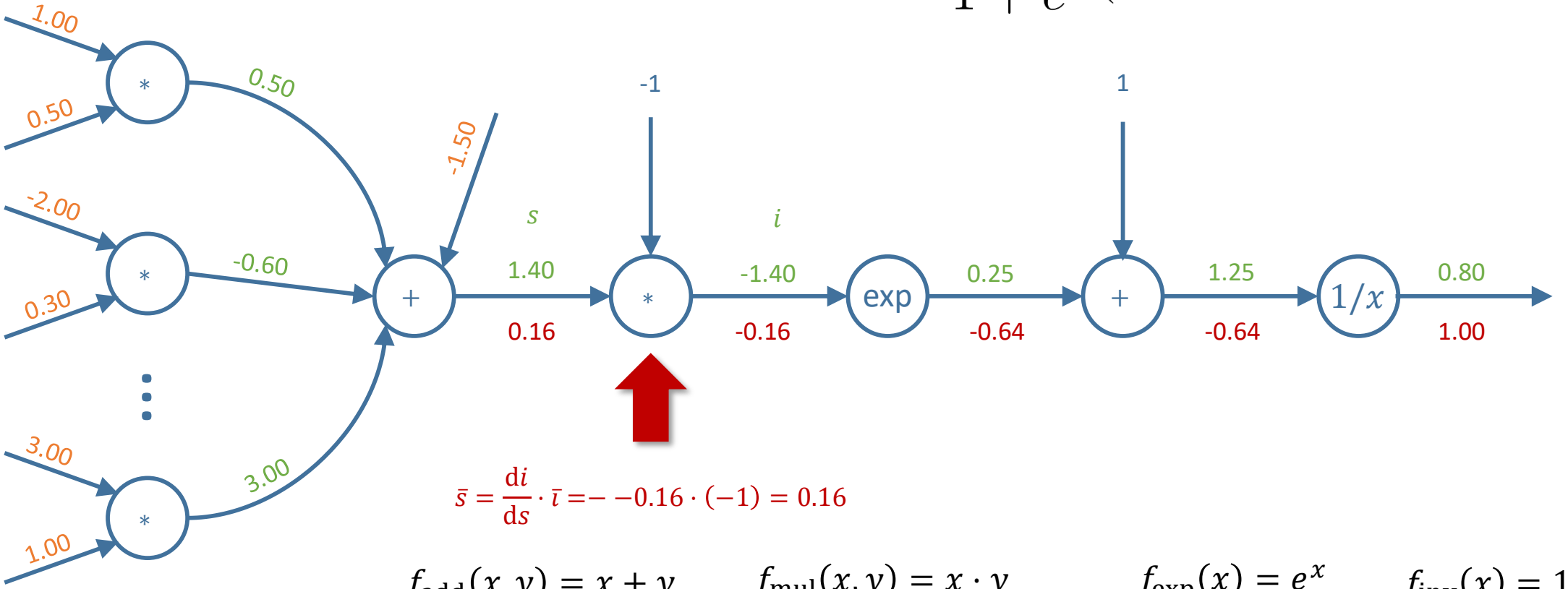
$$\bar{\imath} = \frac{\mathrm{d}g}{\mathrm{d}i} \cdot \bar{g} = -0.64 \cdot 2.72^{-1.40} = -0.16$$

$$f_{\text{add}}(x, y) = x + y \qquad f_{\text{mul}}(x, y) = x \cdot y \qquad f_{\exp}(x) = e^x \qquad f_{\text{inv}}(x) = 1/x$$

$$\frac{\partial f_{\text{add}}}{\partial x} = \frac{\partial f_{\text{add}}}{\partial y} = 1 \qquad \frac{\partial f_{\text{mul}}}{\partial x} = y, \frac{\partial f_{\text{mul}}}{\partial y} = x \qquad \boxed{\frac{\partial f_{\exp}}{\partial x} = e^x} \qquad \frac{\partial f_{\text{inv}}}{\partial x} = -1/x^2$$

Binární logistická regrese se sigmoidem jako graf

$$q = \frac{1}{1 + e^{-(w_1 x_1 + w_2 x_2 + \ldots + w_D x_D + b)}}$$



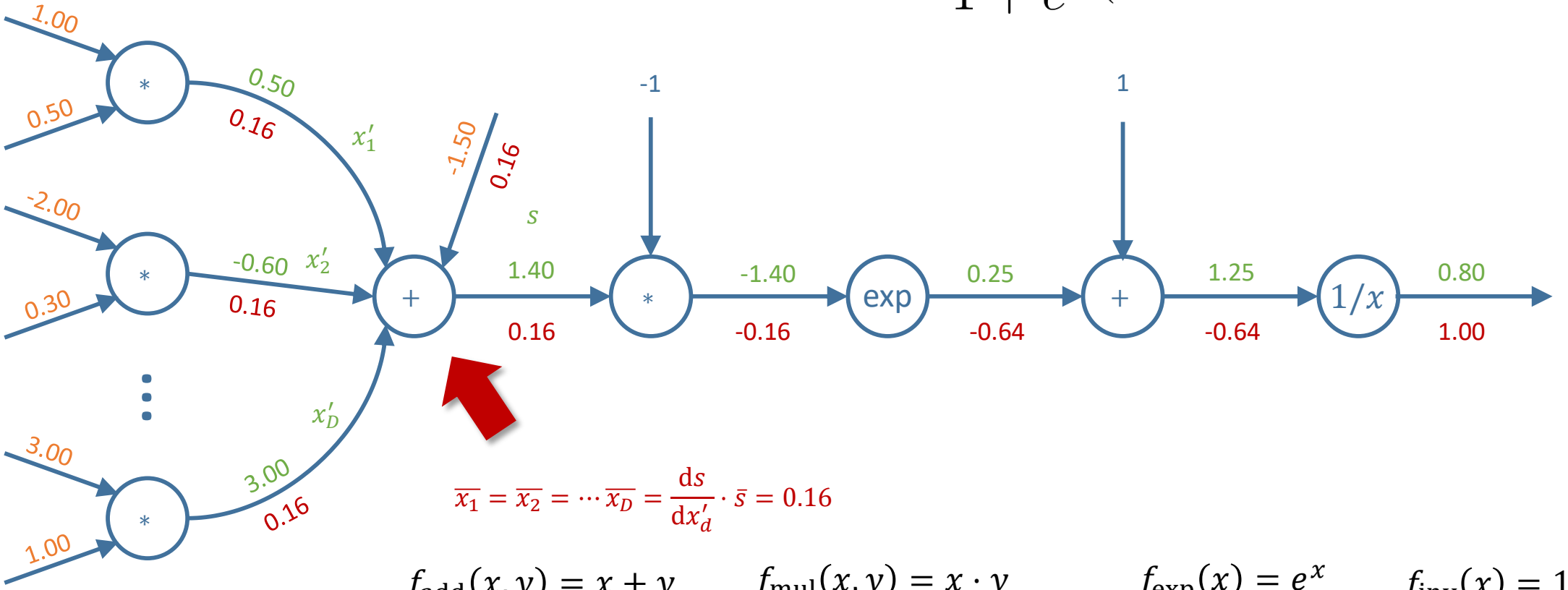
$$\bar{s} = \frac{\mathrm{d}i}{\mathrm{d}s} \cdot \bar{\imath} = - -0.16 \cdot (-1) = 0.16$$

$f_{\text{add}}(x, y) = x + y$ $\quad f_{\text{mul}}(x, y) = x \cdot y$ $\quad f_{\exp}(x) = e^x$ $\quad f_{\text{inv}}(x) = 1/x$

$\frac{\partial f_{\text{add}}}{\partial x} = \frac{\partial f_{\text{add}}}{\partial y} = 1$ $\quad \boxed{\frac{\partial f_{\text{mul}}}{\partial x} = y, \frac{\partial f_{\text{mul}}}{\partial y} = x}$ $\quad \frac{\partial f_{\exp}}{\partial x} = e^x$ $\quad \frac{\partial f_{\text{inv}}}{\partial x} = -1/x^2$

Binární logistická regrese se sigmoidem jako graf

$$q = \frac{1}{1 + e^{-(w_1 x_1 + w_2 x_2 + \ldots + w_D x_D + b)}}$$



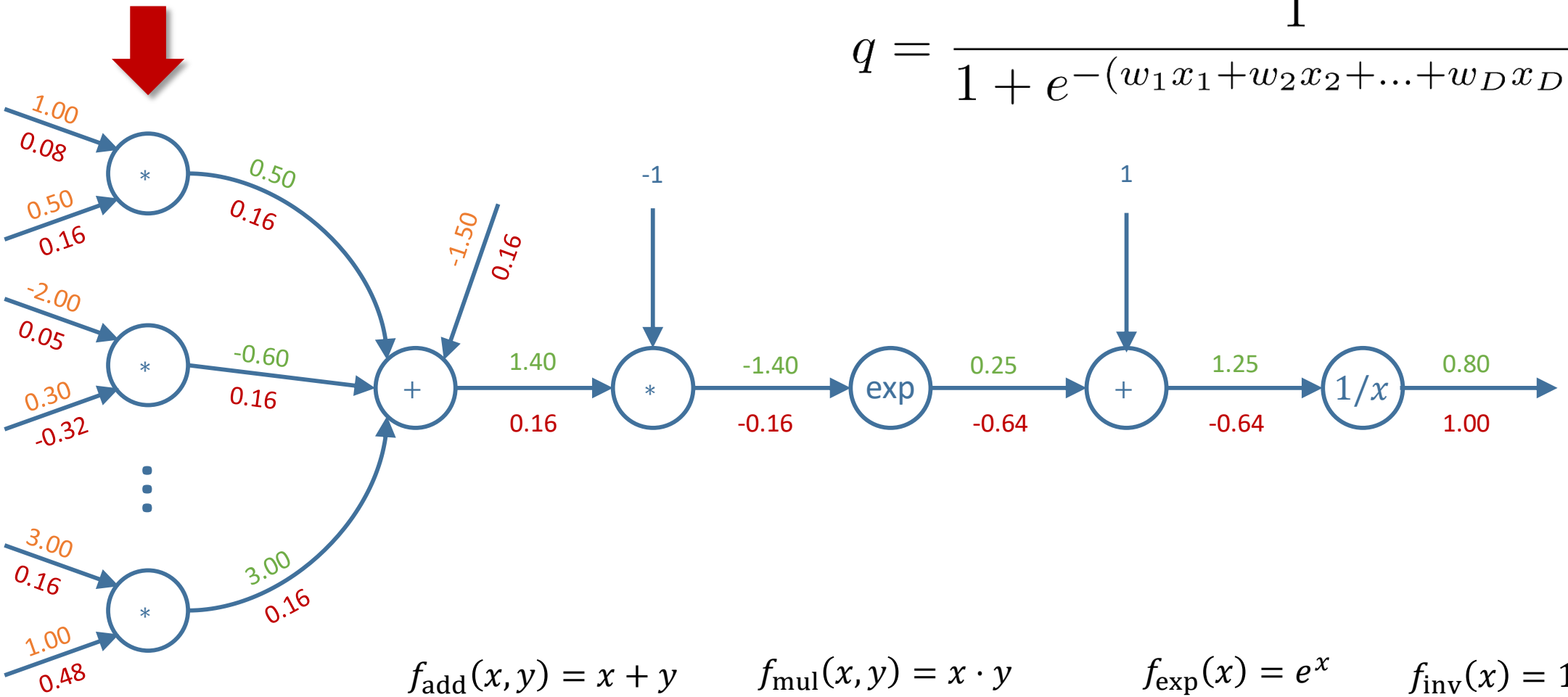
$$\overline{x_1} = \overline{x_2} = \cdots \overline{x_D} = \frac{\mathrm{d}s}{\mathrm{d}x'_d} \cdot \bar{s} = 0.16$$

$$f_{\text{add}}(x, y) = x + y \qquad f_{\text{mul}}(x, y) = x \cdot y \qquad f_{\exp}(x) = e^x \qquad f_{\text{inv}}(x) = 1/x$$

$$\frac{\partial f_{\text{add}}}{\partial x} = \frac{\partial f_{\text{add}}}{\partial y} = 1 \qquad \frac{\partial f_{\text{mul}}}{\partial x} = y, \frac{\partial f_{\text{mul}}}{\partial y} = x \qquad \frac{\partial f_{\exp}}{\partial x} = e^x \qquad \frac{\partial f_{\text{inv}}}{\partial x} = -1/x^2$$

Binární logistická regrese se sigmoidem jako graf

$$q = \frac{1}{1 + e^{-(w_1 x_1 + w_2 x_2 + \ldots + w_D x_D + b)}}$$



$f_{\text{add}}(x, y) = x + y$

$f_{\text{mul}}(x, y) = x \cdot y$

$f_{\text{exp}}(x) = e^x$

$f_{\text{inv}}(x) = 1/x$

$\frac{\partial f_{\text{add}}}{\partial x} = \frac{\partial f_{\text{add}}}{\partial y} = 1$

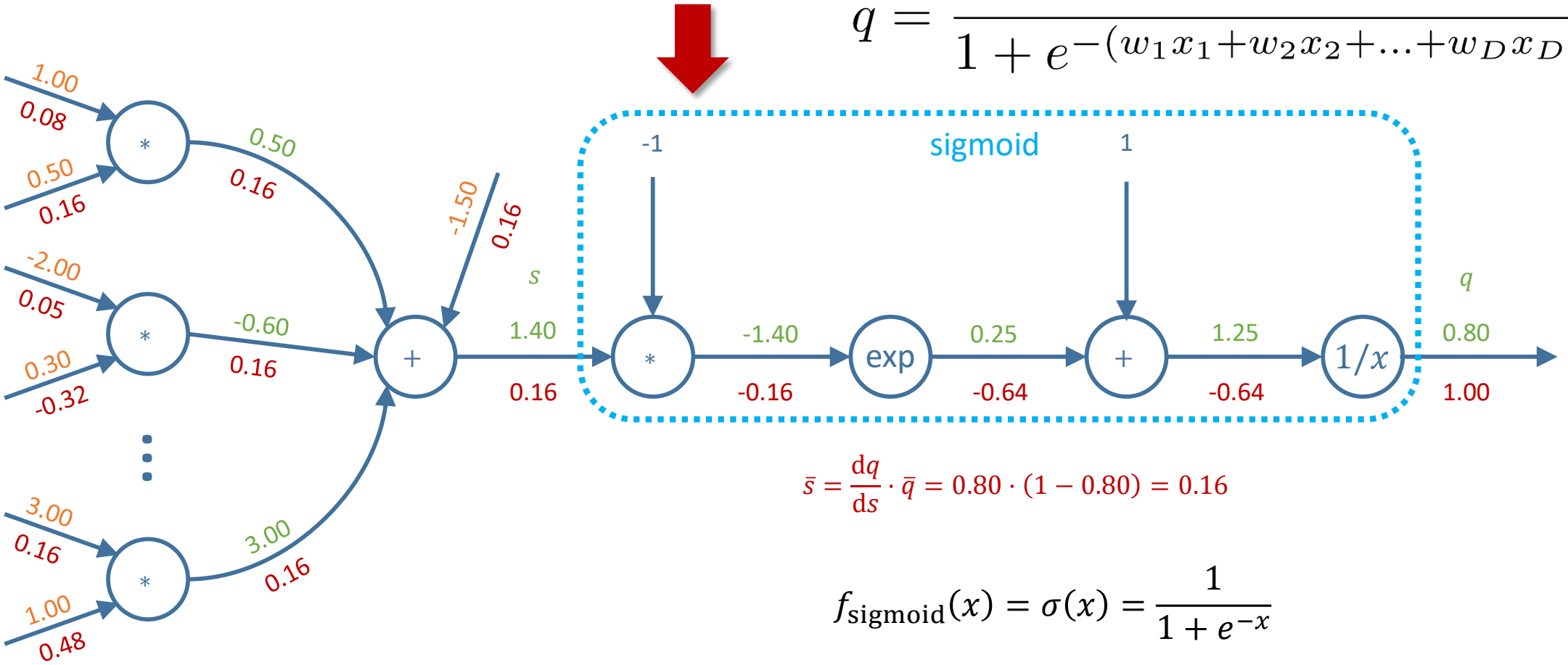
$\frac{\partial f_{\text{mul}}}{\partial x} = y, \frac{\partial f_{\text{mul}}}{\partial y} = x$

$\frac{\partial f_{\text{exp}}}{\partial x} = e^x$

$\frac{\partial f_{\text{inv}}}{\partial x} = -1/x^2$

Binární logistická regrese se sigmoidem jako graf

$$q = \frac{1}{1 + e^{-(w_1 x_1 + w_2 x_2 + \ldots + w_D x_D + b)}}$$



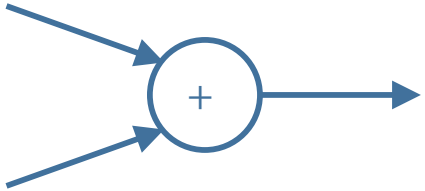
$$\bar{s} = \frac{\mathrm{d}q}{\mathrm{d}s} \cdot \bar{q} = 0.80 \cdot (1 - 0.80) = 0.16$$

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

$$\frac{\partial f_{\text{sigmoid}}}{\partial x} = \sigma(x)\big(1 - \sigma(x)\big)$$

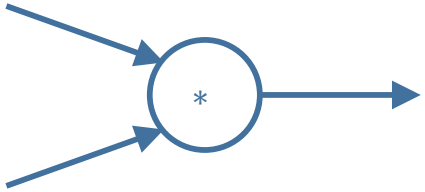
Druhy operací

- Operace v grafu se opakují → platí I pro výpočet gradientu
- V grafu se opakují pouze 4 typy **vrstev**:
  1. součet
  2. násobení
  3. exponenciální funkce
  4. inverze



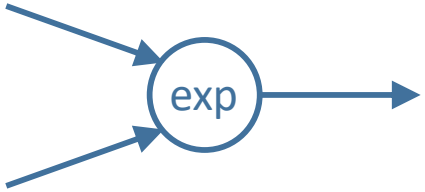
$$f_{\text{add}}(x, y) = x + y$$

$$\frac{\partial f_{\text{add}}}{\partial x} = \frac{\partial f_{\text{add}}}{\partial y} = 1$$



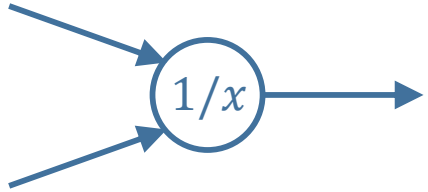
$$f_{\text{mul}}(x, y) = x \cdot y$$

$$\frac{\partial f_{\text{mul}}}{\partial x} = y, \frac{\partial f_{\text{mul}}}{\partial y} = x$$



$$f_{\text{exp}}(x) = e^x$$

$$\frac{\partial f_{\text{exp}}}{\partial x} = e^x$$

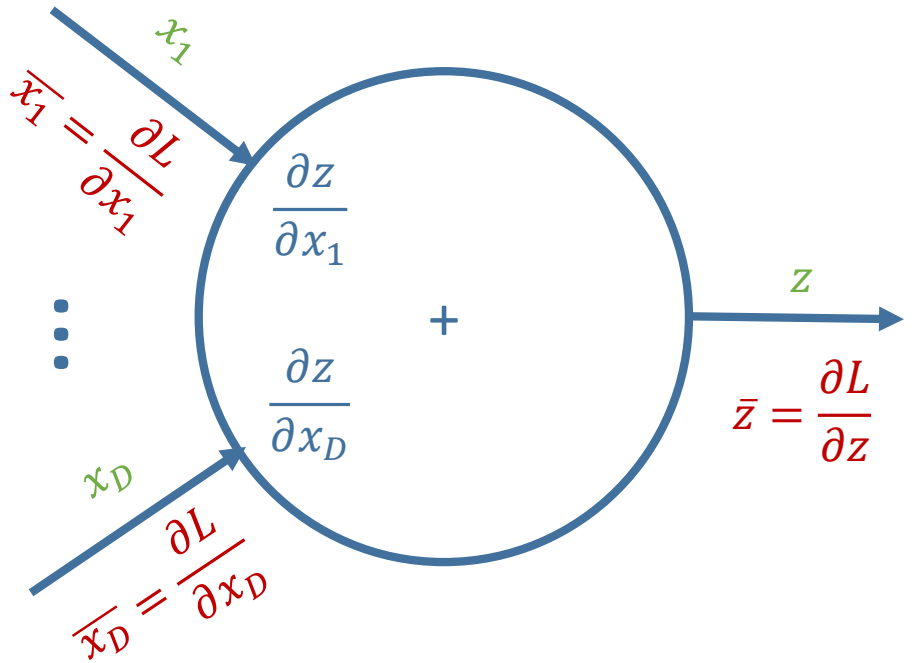


$$f_{\text{inv}}(x) = 1/x$$

$$\frac{\partial f_{\text{inv}}}{\partial x} = -1/x^2$$

Sčítání

```python
class AddNode(object):

    def forward(x_vec):
        # cache
        self.dim = x_vec.shape[0]
        z = np.sum(x_vec)
        return z

    def backward(dz):
        dx_vec = dz * np.ones(self.dim)
        return dx_vec
```



ve zpětném režimu “rozdistribuje” příchozí gradient do všech vstupů:

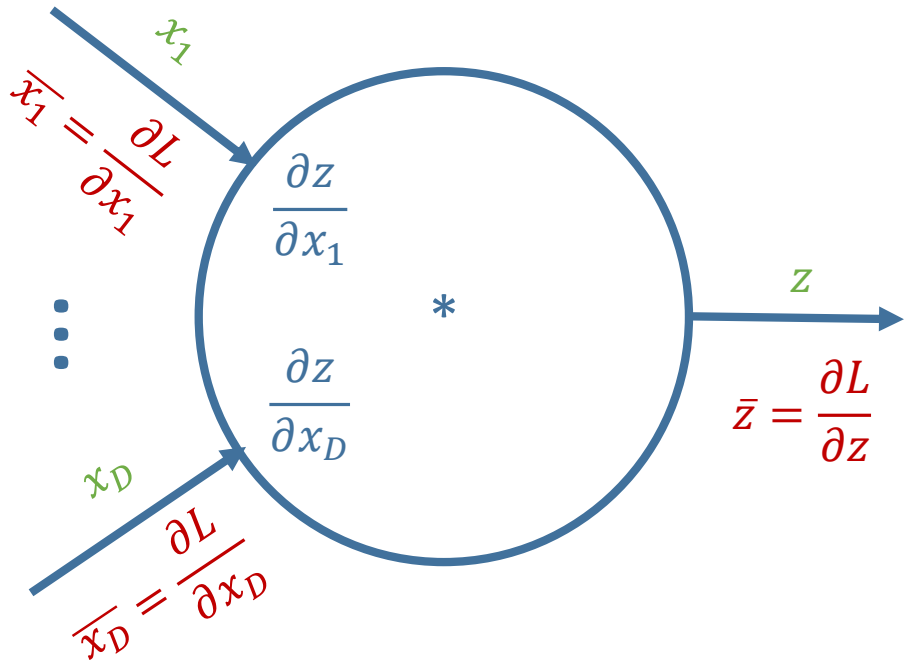
$$\overline{x_d} = \bar{z}, \qquad d = 1, \dots, D$$

forward: $$z = \sum_{d=1}^{D} x_d$$

backward: $$\frac{\partial z}{\partial x_d} = 1$$

Násobení

```python
class MultiplyNode(object):

    def forward(x_vec):
        # cache
        self.x = x_vec
        self.z = np.prod(self.x)
        return self.z

    def backward(dz):
        # dx1 = (x1 * x2 * ... xD) / x1
        dx_vec = dz * self.z / self.x
        return dx_vec
```



forward: $$z = \prod_{d=1}^{D} x_d$$

backward: $$\frac{\partial z}{\partial x_d} = \prod_{i \neq d} x_i$$

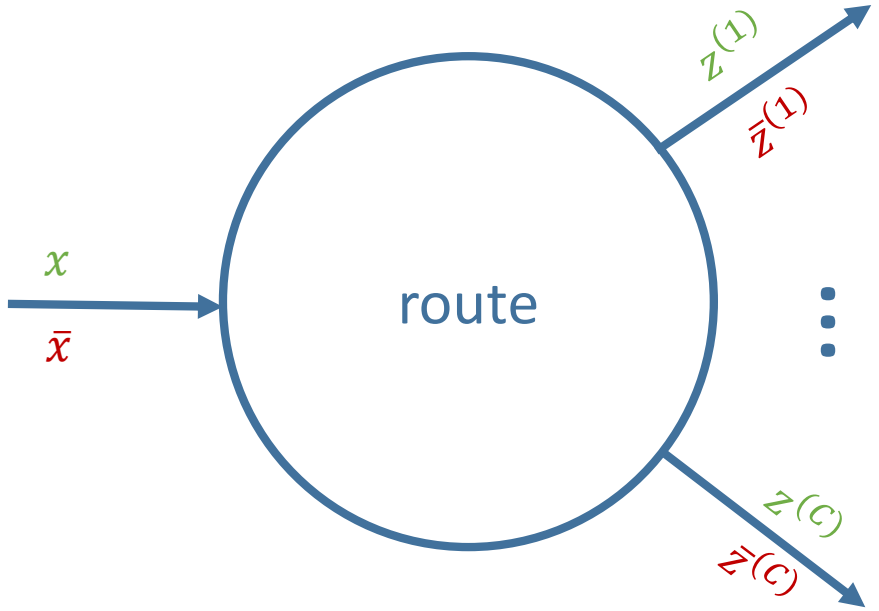
Výpočetní graf pro zabalení funkcí do jednoho bloku

```
class ComputationalGraph(object):
    def add_node(node, *parents):
        self.nodes.append(node)
        self.parents[node] = parents

    def forward(*inputs):
        for node in topologically_sorted(self.nodes):
            inputs = [self.outputs[p] for p in self.parents[node]]
            self.outputs[node] = node.forward(*inputs)
        return top_nodes_outputs  # napr. loss (treba cross entropy)

    def backward(dout):
        for node in reversed(topologically_sorted(self.nodes)):
            node_grads = node.backward(self.grads[node])  # retizkove pravidlo
            for p, g in zip(self.parents[node], node_grads):
                self.grads[p] = g
        return bottom_nodes_grads
```

graf nesmí obsahovat cykly!

Opakované použití jednoho výstupu

- Výstup vstupuje do více než jednoho bloku v další vrstvě
- Konečný výsledek (např. loss) je závislý na obou mezivýsledcích
- Všimněme si duality vůči bloku sčítání



forward: $z^{(c)} = x$

backward: $\bar{x} = \sum_{c=1}^{C} \bar{z}^{(c)}$

Příklad: lineární vrstva, gradient na vstup

- Lineární vrstva

$$\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$$

- Rozvinutý zápis

$$\begin{bmatrix} z_1 \\ \vdots \\ z_C \end{bmatrix} = \begin{bmatrix} w_{11} & \dots & w_{1D} \\ \vdots & \ddots & \vdots \\ w_{C1} & \dots & w_{CD} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_D \end{bmatrix} + \begin{bmatrix} b_1 \\ \vdots \\ b_C \end{bmatrix}$$

- c-tý výstup

$$z_c = \sum_{i=1}^{D} w_{ci} x_i$$

- Gradient pro jeden vstup a výstup

$$\frac{\partial z_c}{\partial x_d} = w_{cd}$$

- Každé x_d je ale použito C-krát: vyskytuje se ve výpočtech všech $z_1, \dots, z_C$, a proto

$$\overline{x_d} = \sum_{c=1}^{C} w_{cd} \overline{z_c} = \boldsymbol{w_d}\overline{\boldsymbol{z}}$$

kde $\boldsymbol{w}_d$ je d-tý sloupec $\boldsymbol{W}$

- Celý gradient vektorově:

$$\overline{\boldsymbol{x}} = \boldsymbol{W}^\top \overline{\boldsymbol{z}}$$

Příklad: lineární vrstva, gradient na váhy

- Pro c-tý výstup pouze c-tý řádek – např. z_1 nezávisí na $w_{21}, \ldots, w_{2D}$

$$z_c = \sum_{i=1}^{D} w_{ci} x_i \quad \Rightarrow \quad \frac{\partial z_c}{\partial w_{ij}} = \begin{cases} x_j & \text{pokud} \quad i = c \\ 0 & \text{pokud} \quad \text{jinak} \end{cases}$$

- Celkový gradient tedy je

$$\overline{w_{ij}} = \sum_{c=1}^{C} \overline{z_c} \cdot \frac{\partial z_c}{\partial w_{ij}} = \overline{z_i} \cdot x_j$$

x co bylo na vstupu při dopředném průchodu → **cache!**

- Vektorově

$$\overline{\boldsymbol{W}} = \frac{\partial L}{\partial \boldsymbol{W}} = \begin{bmatrix} \overline{z_1} \\ \vdots \\ \overline{z_C} \end{bmatrix} \begin{bmatrix} x_1 & \ldots & x_D \end{bmatrix} = \begin{bmatrix} \overline{z_1} \cdot x_1 & \ldots & \overline{z_1} \cdot x_d \\ \vdots & \ddots & \vdots \\ \overline{z_C} \cdot x_1 & \ldots & \overline{z_C} \cdot x_d \end{bmatrix} = \overline{z} \cdot \boldsymbol{x}^\top$$

Sigmoid jako vrstva

- Dopředný průchod

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

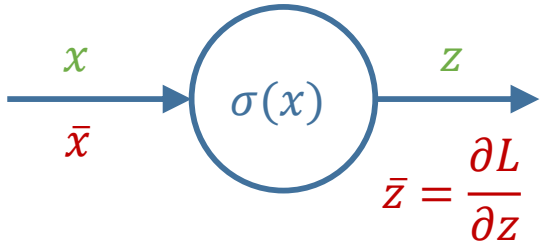
- Derivace funkce sigmoid

$$\frac{\mathrm{d}\sigma(x)}{\mathrm{d}x} = \sigma(x)\big(1 - \sigma(x)\big)$$

- Zpětný průchod

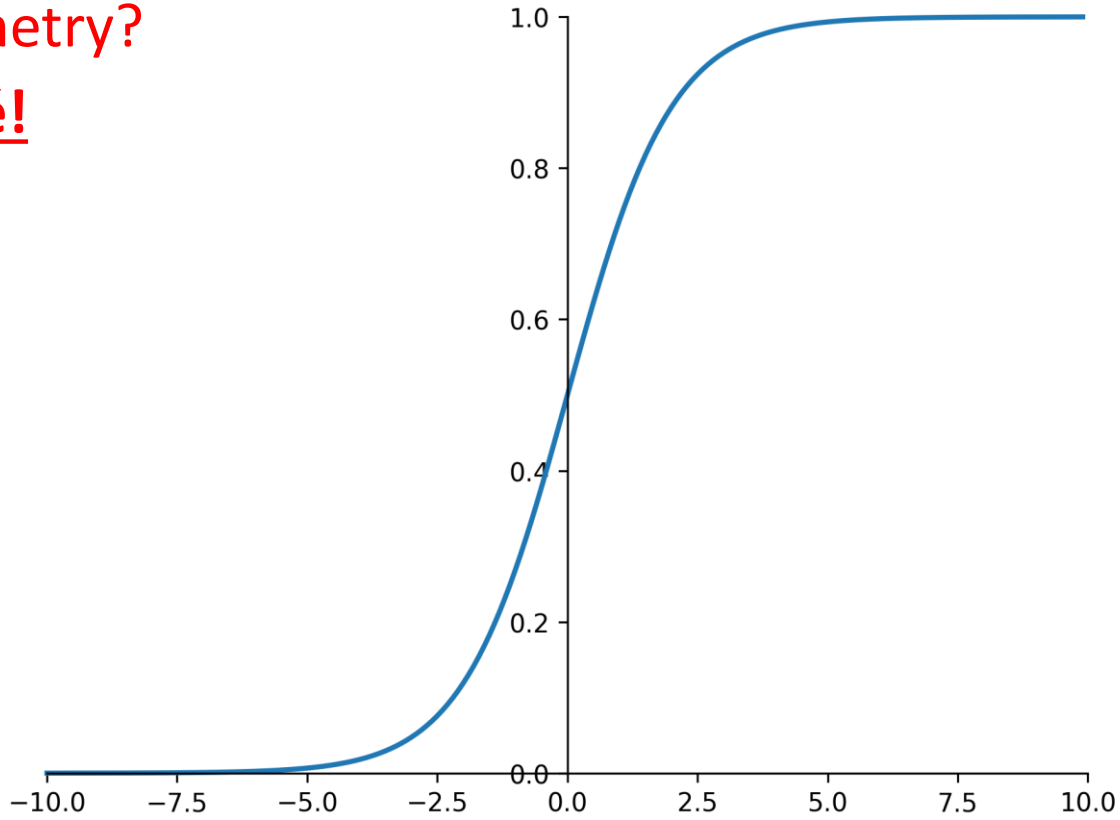
$$\bar{x} = \bar{z} \cdot \frac{\mathrm{d}z}{\mathrm{d}x} = \bar{z} \cdot z \cdot (1 - z)$$

z … **cache!**



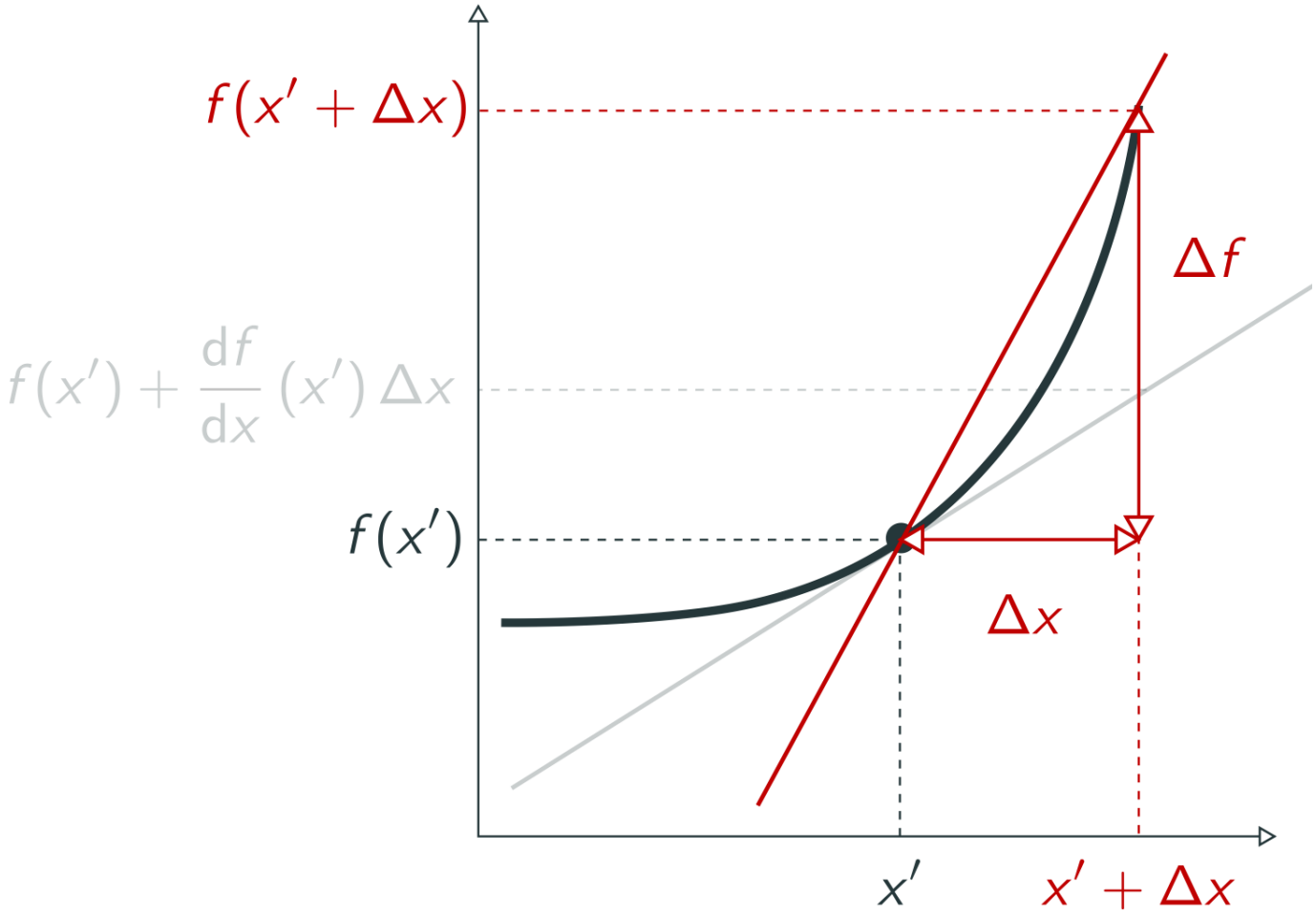
Parametry?

Žádné!



Numerický gradient

Aproximace derivace dopřednou diferencí



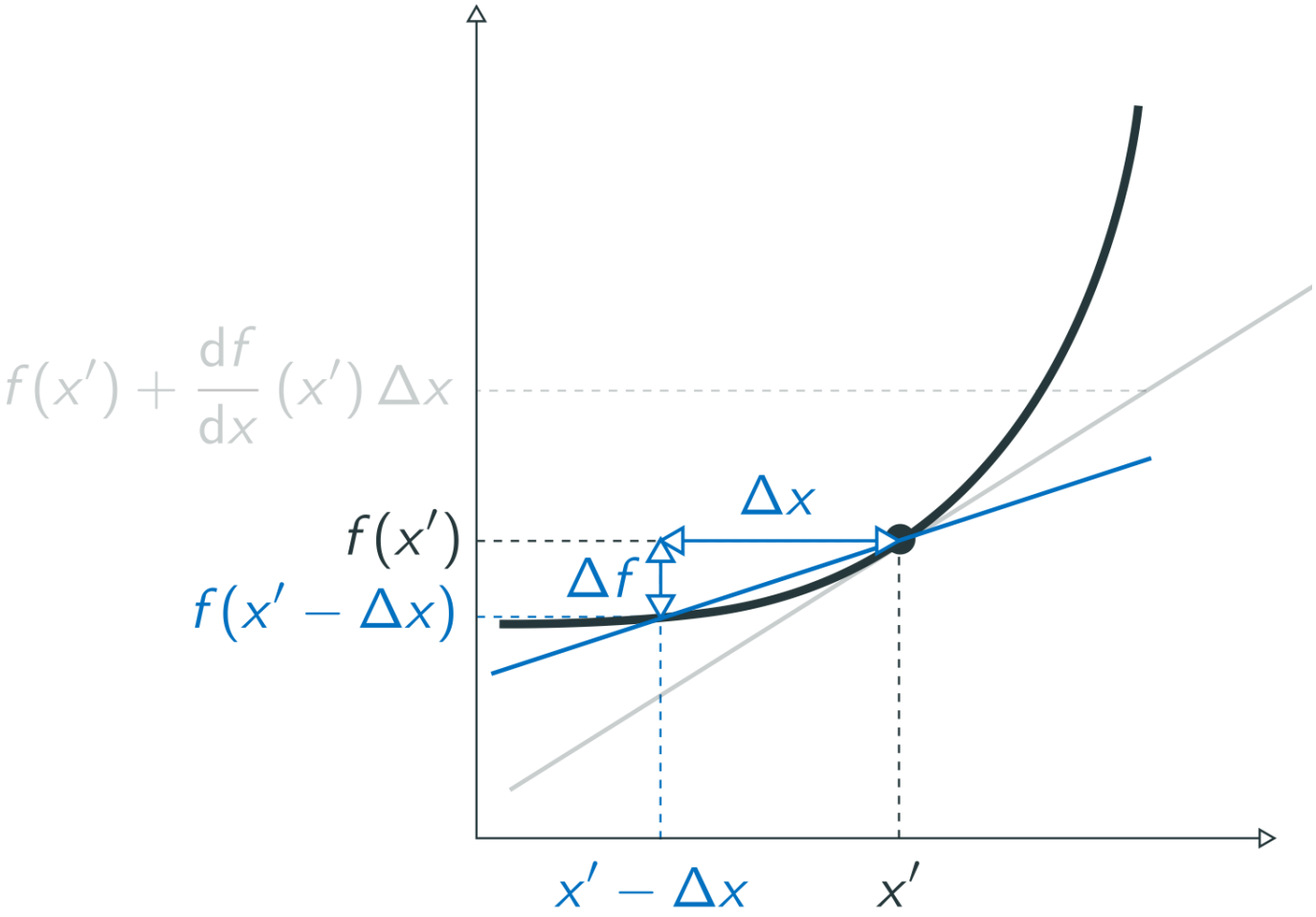
Derivace:

$$\frac{\mathrm{d}f}{\mathrm{d}x} = \lim_{\Delta x \to 0} \frac{\Delta f}{\Delta x}$$

Aproximace dopředně:

$$\frac{\mathrm{d}f}{\mathrm{d}x} \approx \frac{f(x' + \Delta x) - f(x')}{\Delta x}$$

Aproximace derivace zpětnou diferencí



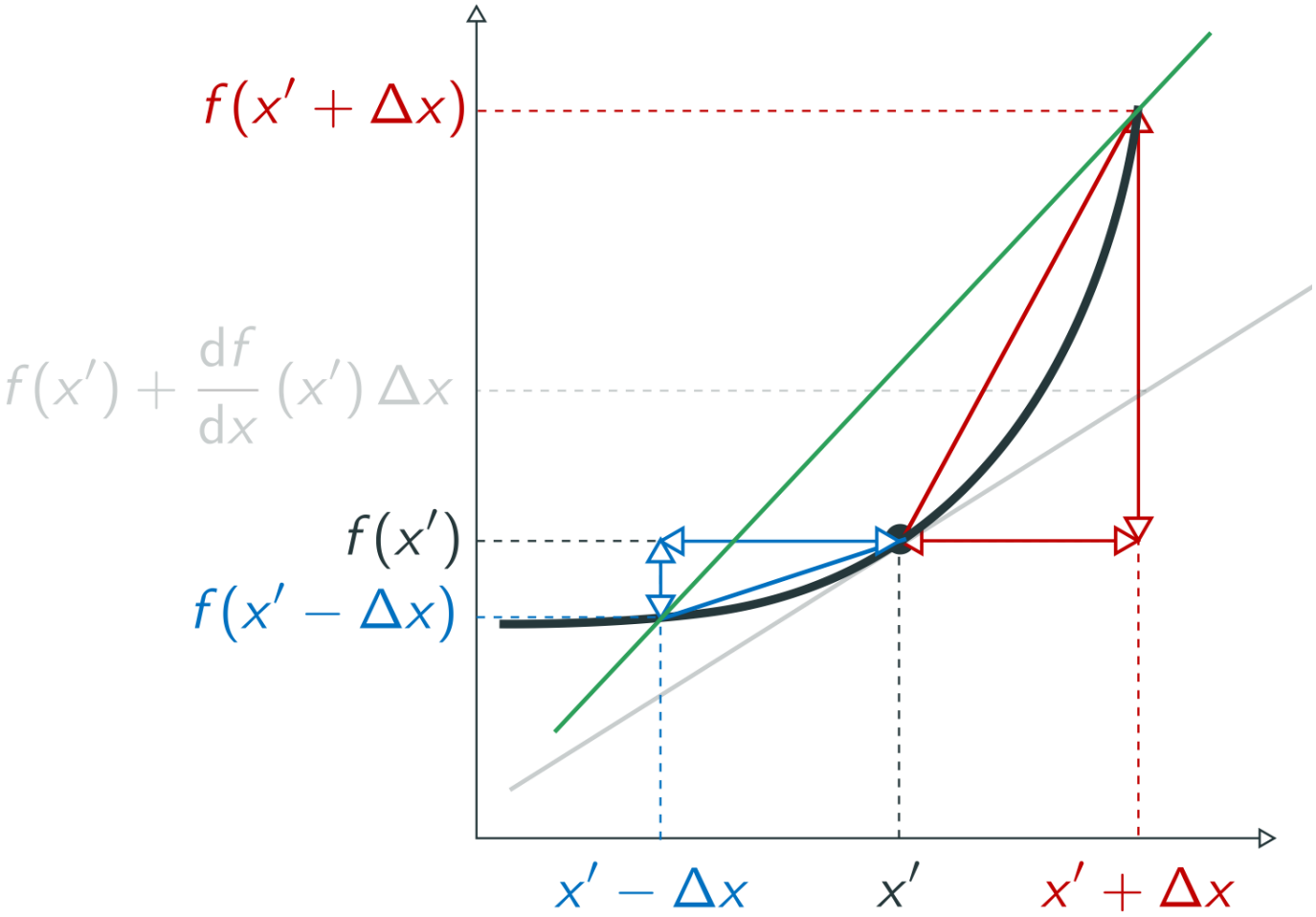
Derivace:

$$\frac{\mathrm{d}f}{\mathrm{d}x} = \lim_{\Delta x \to 0} \frac{\Delta f}{\Delta x}$$

Aproximace zpětně:

$$\frac{\mathrm{d}f}{\mathrm{d}x} \approx \frac{f(x') - f(x' - \Delta x)}{\Delta x}$$

Aproximace derivace centrální diferencí



Derivace:

$$\frac{\mathrm{d}f}{\mathrm{d}x} = \lim_{\Delta x \to 0} \frac{\Delta f}{\Delta x}$$

Aproximace centrálně:

$$\frac{\mathrm{d}f}{\mathrm{d}x} \approx \frac{f(x' + \Delta x) - f(x' - \Delta x)}{2\Delta x}$$

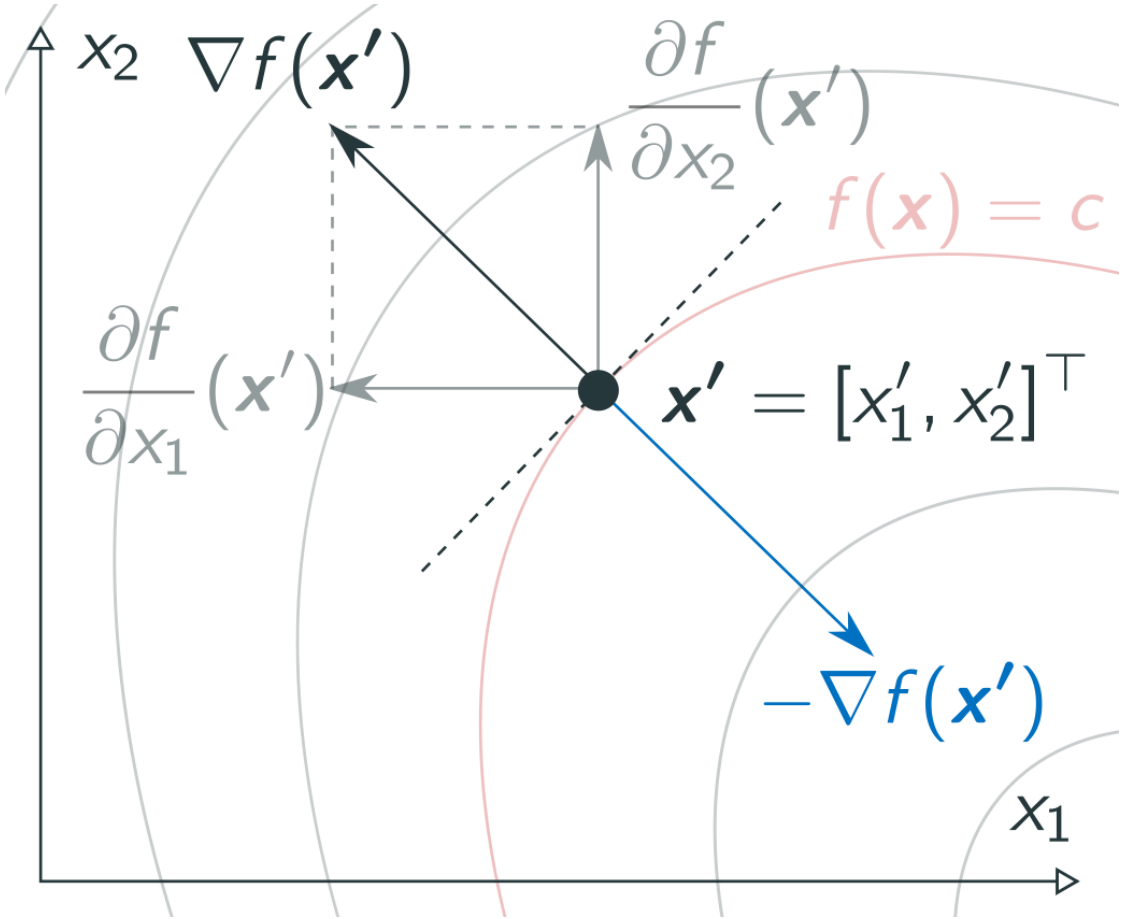
výhodou menší chyba aproximace $O(\Delta x^2)$ → častější použití než jednostranné

Aproximace ve více rozměrech

- Gradient je vektor parciálních derivací

$$\nabla f(\boldsymbol{x}') = \left[\frac{\partial f}{\partial x_1}(x'_1), \ldots, \frac{\partial f}{\partial x_D}(x'_D)\right]^\top$$

- Aproximace tedy nutné provést **pro každou** proměnnou funkce f



Příklad aproximace derivace numerickou diferencí

$$f(\boldsymbol{x}) = \sum x_d^2 \qquad \boldsymbol{x} = [1.0, 0.8, 1.3]^T$$

numerický odhad:

$$\widehat{\nabla f(x)} = \begin{bmatrix} \frac{f(x_1 + \Delta x, x_2, x_3) - f(x_1, x_2, x_3)}{\Delta x} \\ \frac{f(x_1, x_2 + \Delta x, x_3) - f(x_1, x_2, x_3)}{\Delta x} \\ \frac{f(x_1, x_2, x_3 + \Delta x) - f(x_1, x_2, x_3)}{\Delta x} \end{bmatrix} = \begin{bmatrix} \frac{f(1.0 + 0.001, 0.8, 1.3) - f(1.0, 0.8, 1.3)}{0.001} \\ \frac{f(1.0, 0.8 + 0.001, 1.3) - f(1.0, 0.8, 1.3)}{0.001} \\ \frac{f(1.0, 0.8, 1.3 + 0.001) - f(1.0, 0.8, 1.3)}{0.001} \end{bmatrix} = \begin{bmatrix} 2.001 \\ 1.601 \\ 2.601 \end{bmatrix}$$

skutečný gradient:

$$\nabla f(\boldsymbol{x}) = [2x_1, 2x_2, 2x_3]^\top = [2.0, 1.6, 2.6]^\top$$

hluboké neurosítě bežně miliony parametrů → pro vyčíslení gradientu nutné vyhodnotit jejich výstup min. jednou pro každý parametr a to v každém kroku gradient descentu!

Centrální diference v numpy

```
def eval_numerical_gradient_array(f, x, df, h=1e-5):
    grad = np.zeros_like(x)
    it = np.nditer(x, flags=['multi_index'], op_flags=['readwrite'])
    while not it.finished:
        ix = it.multi_index

        oldval = x[ix]
        x[ix] = oldval + h
        pos = f(x).copy()
        x[ix] = oldval - h
        neg = f(x).copy()
        x[ix] = oldval

        grad[ix] = np.sum((pos - neg) * df) / (2 * h)
        it.iternext()
        return grad
```

http://cs231n.github.io/

Kontrola gradientu

- Numerická aproximace je sice **pomalá**, ale vždy až na toleranci **správná** → vhodná **pouze pro kontrolu** implementace backward metody
- V příkladu vyšlo

$$\widehat{\nabla f} = [2.001, 1.601, 2.601]^T$$
$$\nabla f = [2.0, 1.6, 2.6]^T$$

- Relativní chyba

$$\delta = \max \frac{\left|\widehat{\nabla f} - \nabla f\right|}{\left|\widehat{\nabla f}\right| + |\nabla f|} = \max\left(\frac{0.001}{4.001}, \frac{0.001}{3.201}, \frac{0.001}{5.201}\right) = 3.124 \cdot 10^{-4}$$

- Kód v numpy z předmetu cs231n Stanfordovy univerzity:

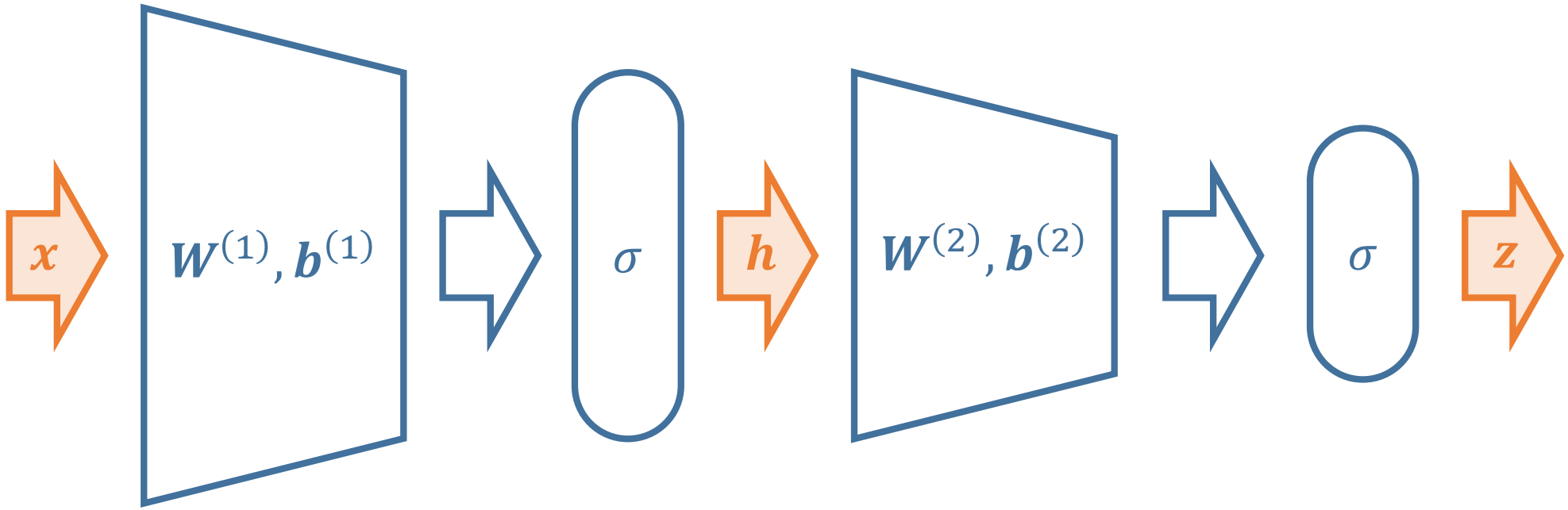
```
>>> def rel_error(x, y):
>>>     return np.max(np.abs(x - y) / (np.maximum(1e-8, np.abs(x) + np.abs(y))))
>>> rel_error(np.array([2.001, 1.601, 2.601]), np.array([2.0, 1.6, 2.6]))
0.00031240237425800993
```

Vícevrstvé sítě

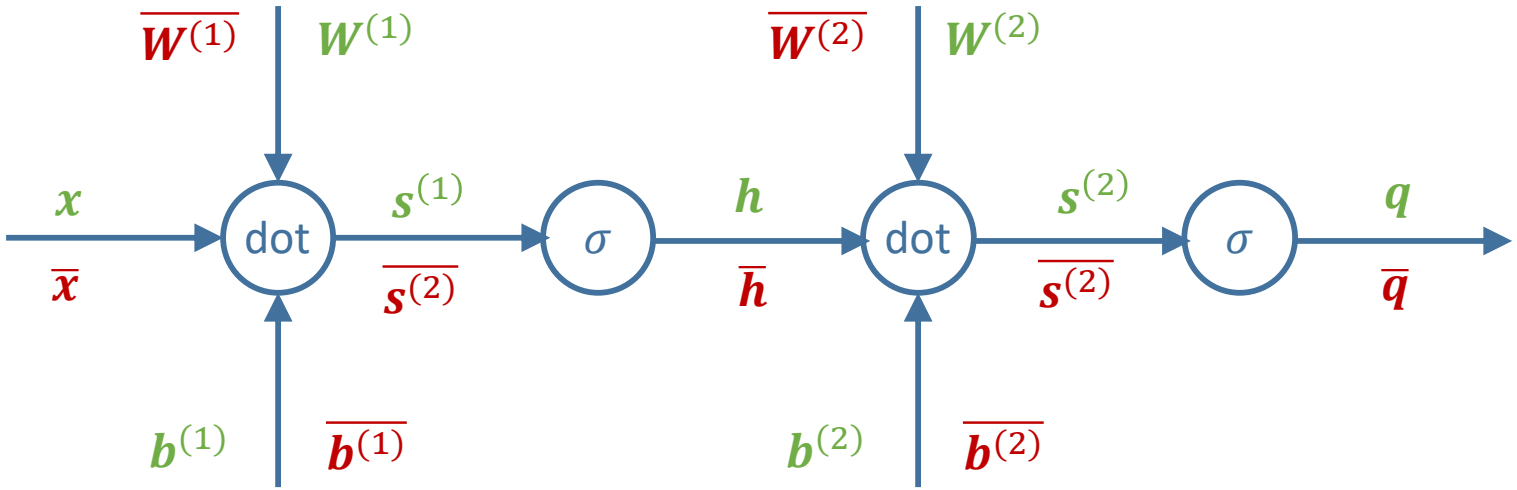
Vícevrstvý perceptron

- Bloky s definovaným forward a backward chováním lze libovolně skládat za sebe
- Např. dvouvrstvý perceptron

$$z = \sigma\left\{\boldsymbol{W}^{(2)}\sigma\left(\boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)}\right) + \boldsymbol{b}^{(2)}\right\}$$



Vícevrstvý perceptron



forward:

x $\Rightarrow$ $s^{(1)} \leftarrow W^{(1)}x + b^{(1)}$ $\Rightarrow$ $h \leftarrow \sigma(s^{(1)})$ $\Rightarrow$ $s^{(2)} \leftarrow W^{(2)}h + b^{(2)}$ $\Rightarrow$ $q \leftarrow \sigma(s^{(2)})$

backward:

$\overline{x} \leftarrow {W^{(1)}}^T \overline{s^{(1)}}$

$\overline{W^{(1)}} \leftarrow \overline{s^{(1)}} h^T$

$\overline{b^{(1)}} \leftarrow \overline{s^{(1)}}$

$\Leftarrow$ $\overline{s^{(1)}} \leftarrow h(1-h)\overline{h}$ $\Leftarrow$ $\overline{h} \leftarrow {W^{(2)}}^T \overline{s^{(2)}}$

$\overline{W^{(2)}} \leftarrow \overline{s^{(2)}} h^T$

$\overline{b^{(2)}} \leftarrow \overline{s^{(2)}}$

$\Leftarrow$ $\overline{s^{(2)}} \leftarrow q(1-q)\overline{q}$ $\Leftarrow$ $\overline{q}$

Dvouvrstvý perceptron v numpy na 11 řádků

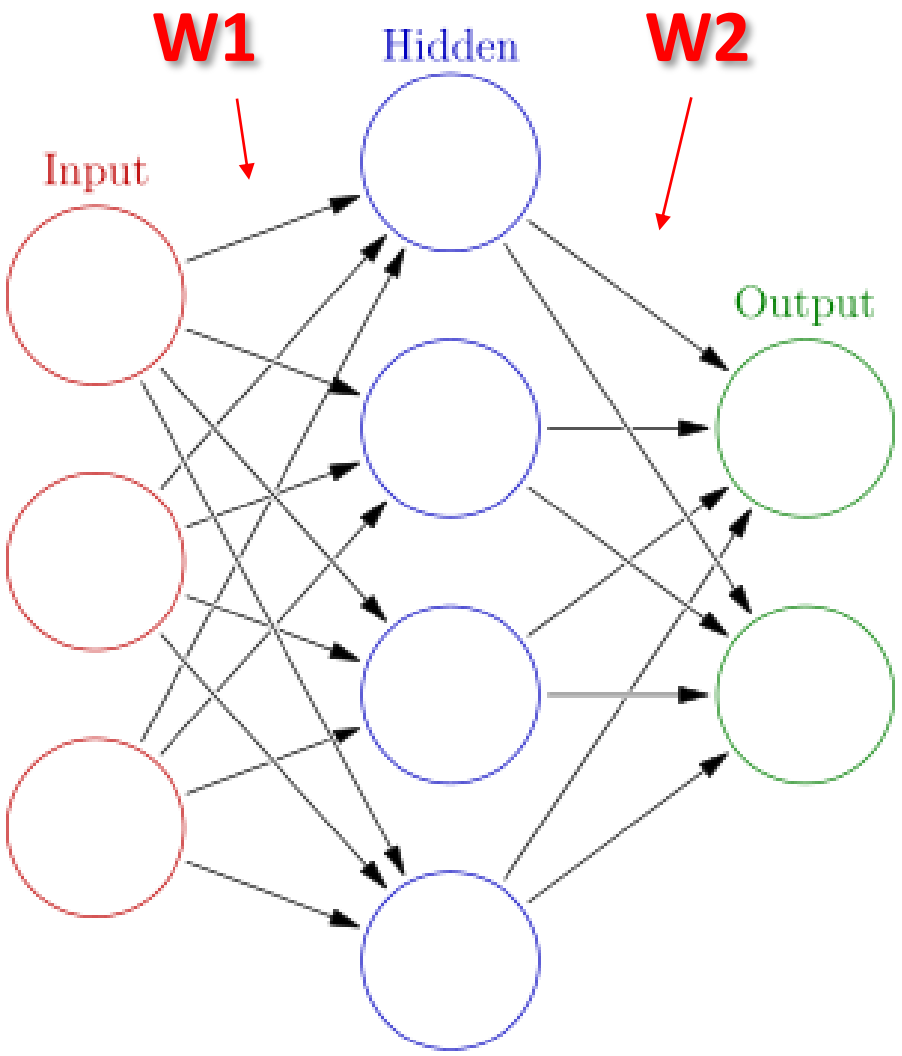
```
X = np.array([[0, 0, 1], [0, 1, 1], [1, 0, 1], [1, 1, 1]])
y = np.array([[0, 1, 1, 0]]).T
syn0 = 2*np.random.random((3, 4)) - 1
syn1 = 2*np.random.random((4, 1)) - 1
for j in xrange(60000):
    l1 = 1 / (1 + np.exp(-(np.dot(X, syn0))))
    l2 = 1 / (1 + np.exp(-(np.dot(l1, syn1))))
    l2_delta = (y - l2) * (l2 * (1 - l2))
    l1_delta = l2_delta.dot(syn1.T) * (l1 * (1 - l1))
    syn1 += l1.T.dot(l2_delta)
    syn0 += X.T.dot(l1_delta)
```

http://iamtrask.github.io/2015/07/12/basic-python-network/

Inicializace

Náhodná inicializace

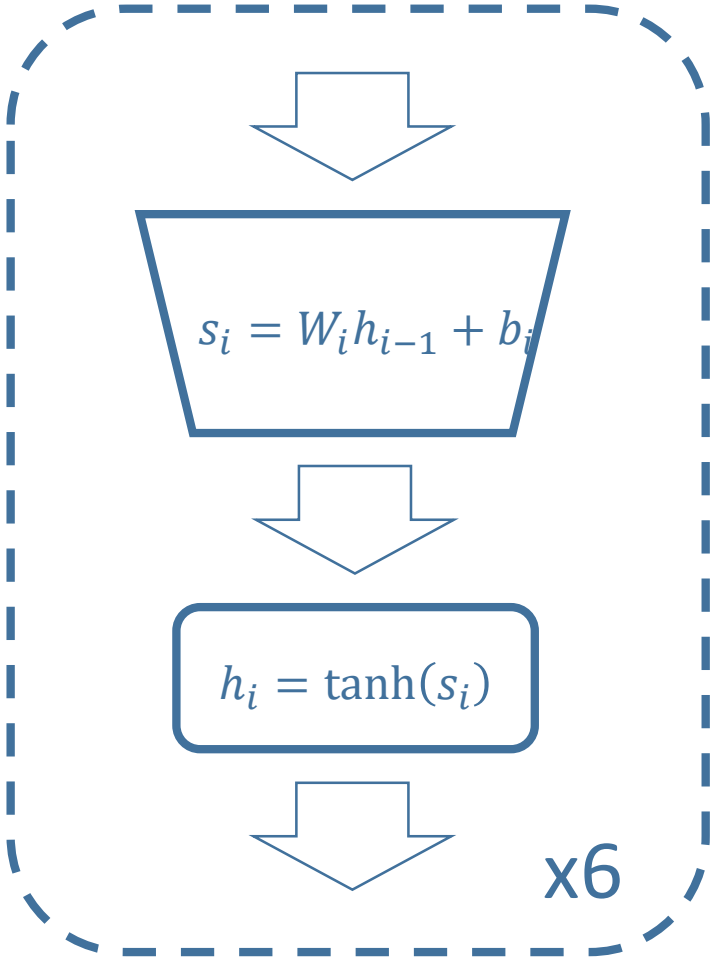
```
W1 = 0.01 * np.random.randn(3, 4)
W2 = 0.01 * np.random.randn(4, 2)
```



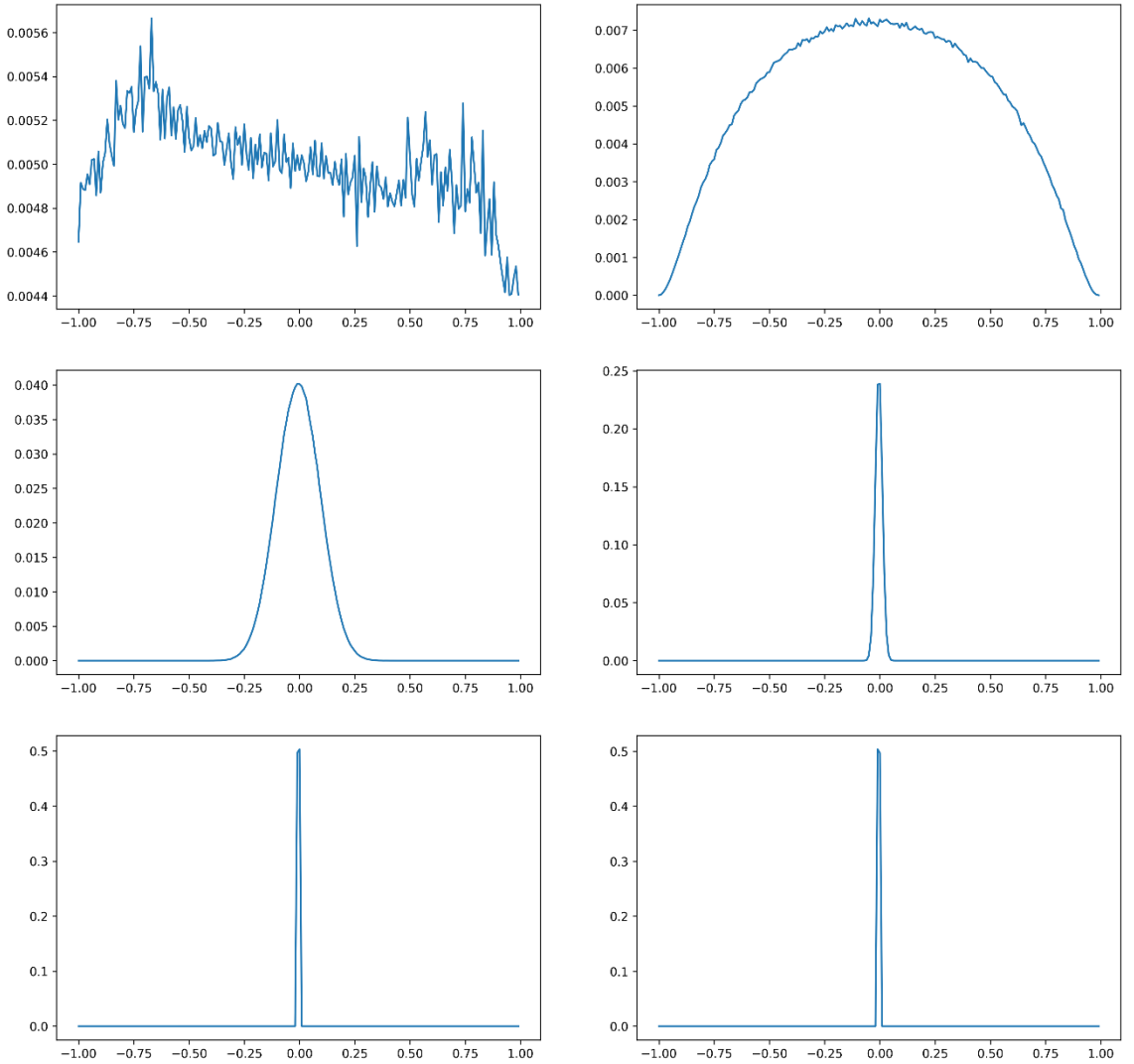
- Proč náhodně? Proč ne všechno na nuly?
- Potřebujeme narušit symetrii (break the symmetry)
  - Řekněme, že váhy jsou nuly a hidden vrstva používá sigmoid
  - do output pak jdou samé 0.5
  - tzn., že i gradient bude všude téměř stejný
  - **chceme opak:** aby každý neuron dělal něco jiného

Příliš malé váhy

- 6 vrstev feed forward síť dle obr.
- histogram hodnot skrytých vrstev

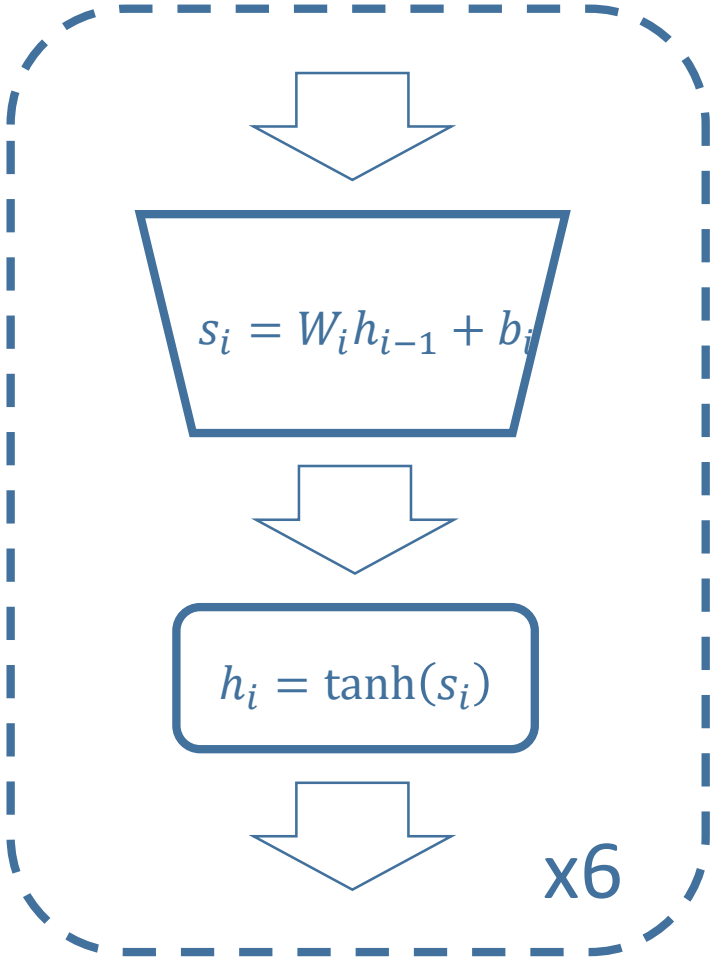



```
Wi = 0.01 * np.random.randn(fan_in, fan_out)
```

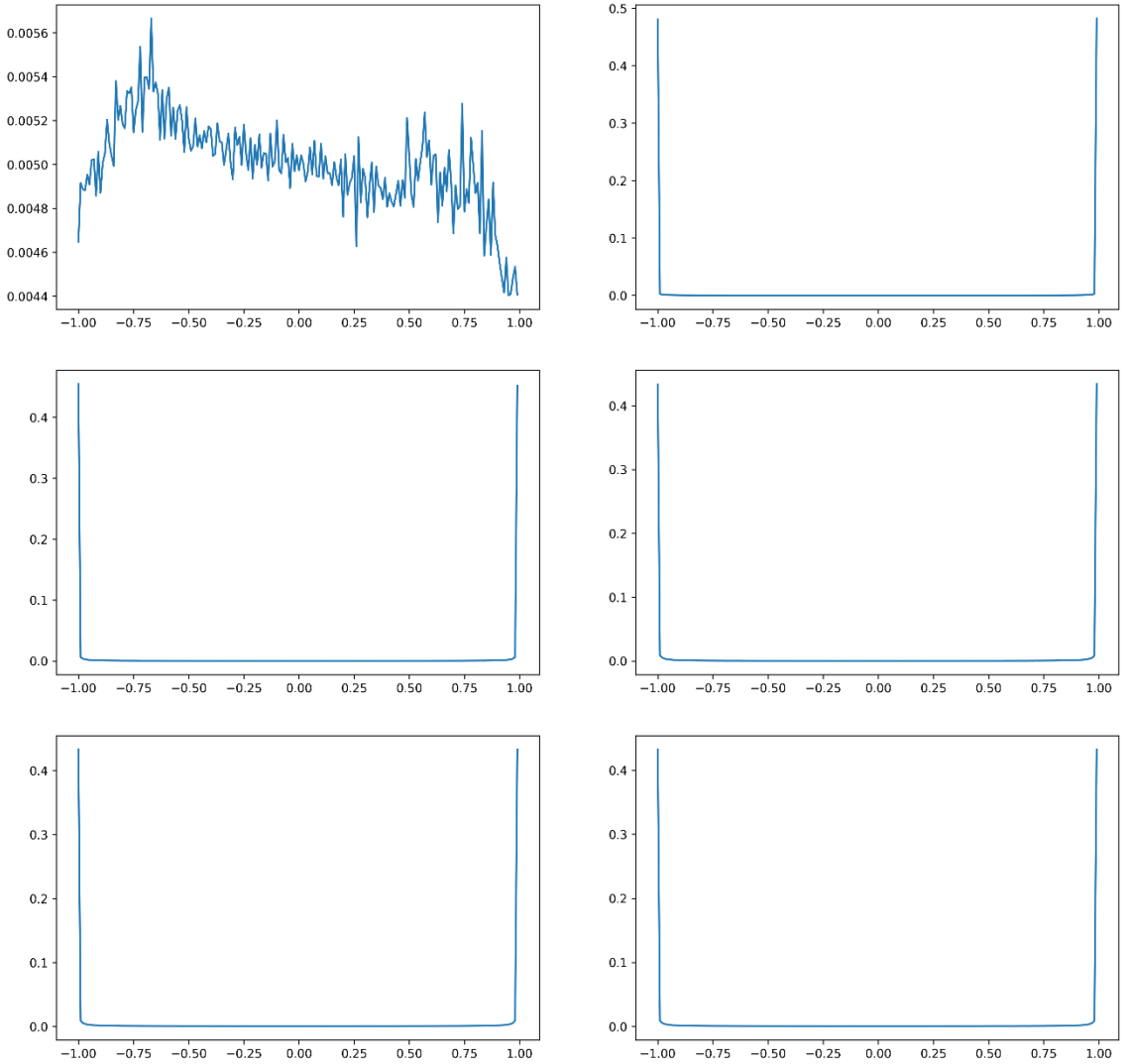


Příliš velké váhy

- 6 vrstev feed forward síť dle obr.
- histogram hodnot skrytých vrstev



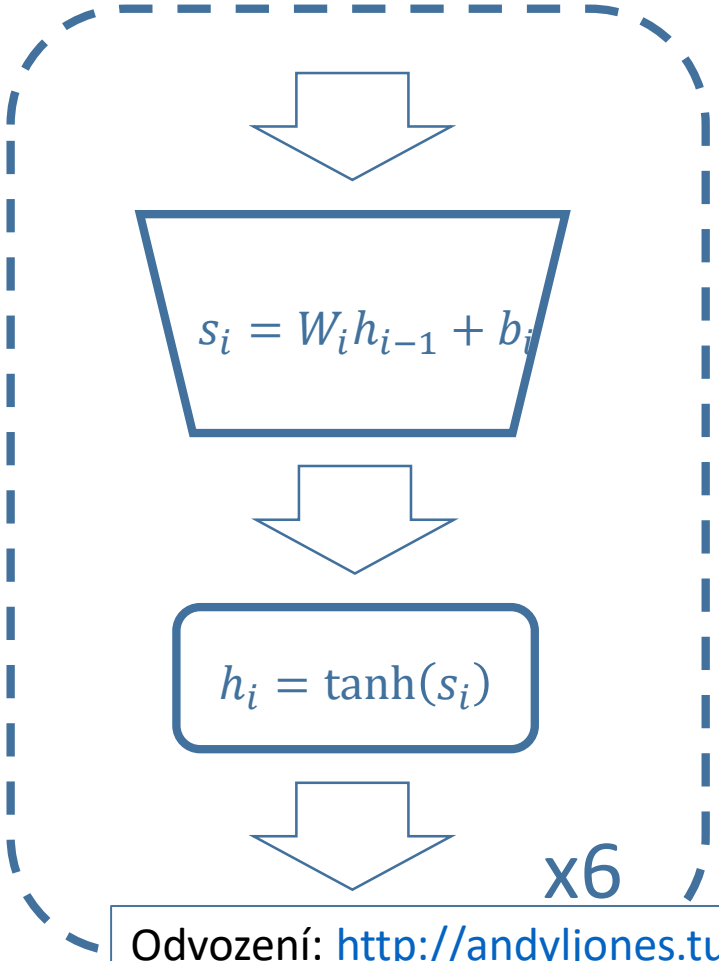

```
Wi = 1.00 * np.random.randn(fan_in, fan_out)
```

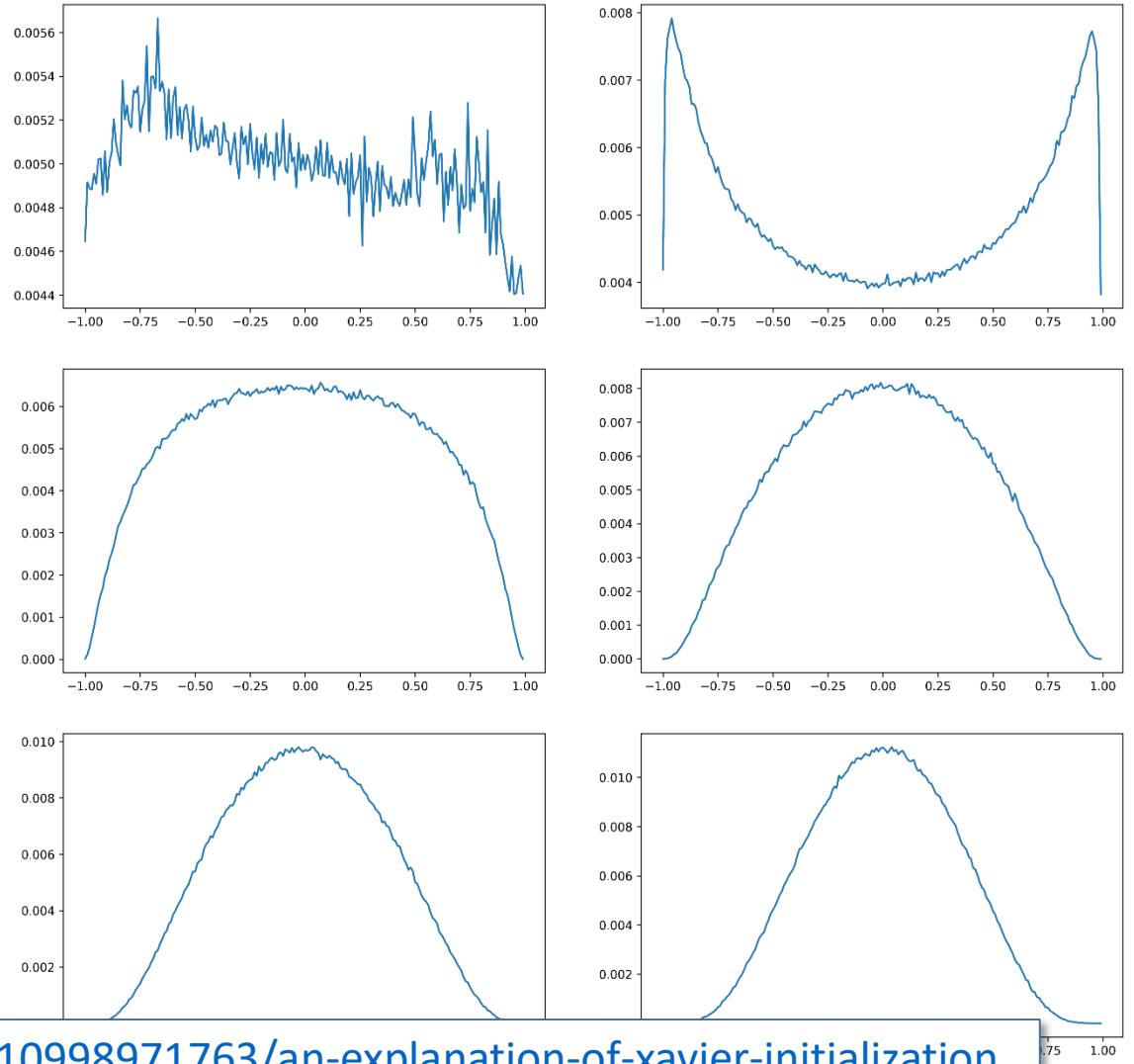


Xavier / Glorot inicializace

- 6 vrstev feed forward síť dle obr.
- histogram hodnot skrytých vrstev

```
Wi = np.random.randn(fan_in, fan_out) / np.sqrt(fan_in)
```





Odvození: http://andyljones.tumblr.com/post/110998971763/an-explanation-of-xavier-initialization

Špatná inicializace