Università degli
Studi di Pavia

Deep Learning

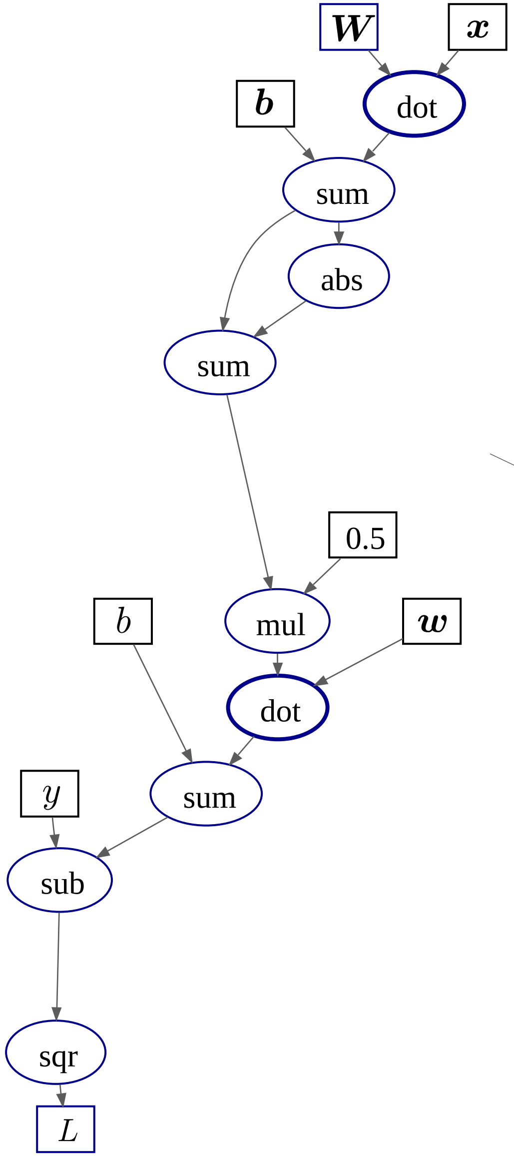
03- Flow Graphs & Automatic Differentiation

Marco Piastra

This presentation can be downloaded at:
http://vision.unipv.it/DL

Flow Graphs

An aside: Flow Graph

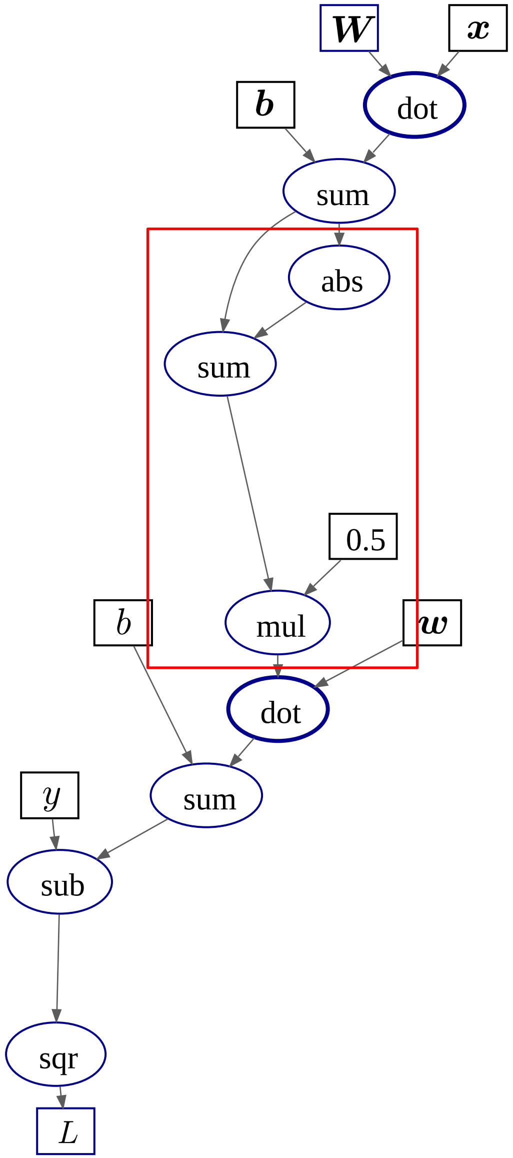


$$L(\tilde{y}, y) = (\boldsymbol{w} \cdot \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b - y)^2$$

Item-wise loss function, FF neural network with ReLU as non-linearity

The above expression translates into this flow graph

An aside: Flow Graph



$$L(\tilde{y}, y) = (\boldsymbol{w} \cdot \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b - y)^2$$

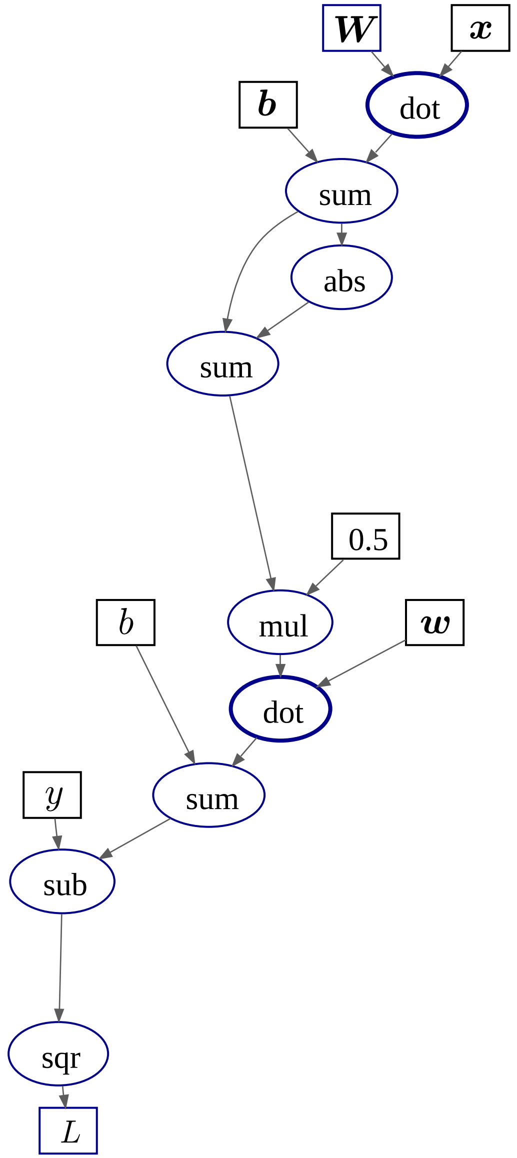
Item-wise loss function, FF neural network with ReLU as non-linearity

$$\mathrm{ReLU}(x) := \max(0, x)$$

$$\mathrm{ReLU}(x) = \frac{1}{2}(x + |x|)$$

(equivalent expression)

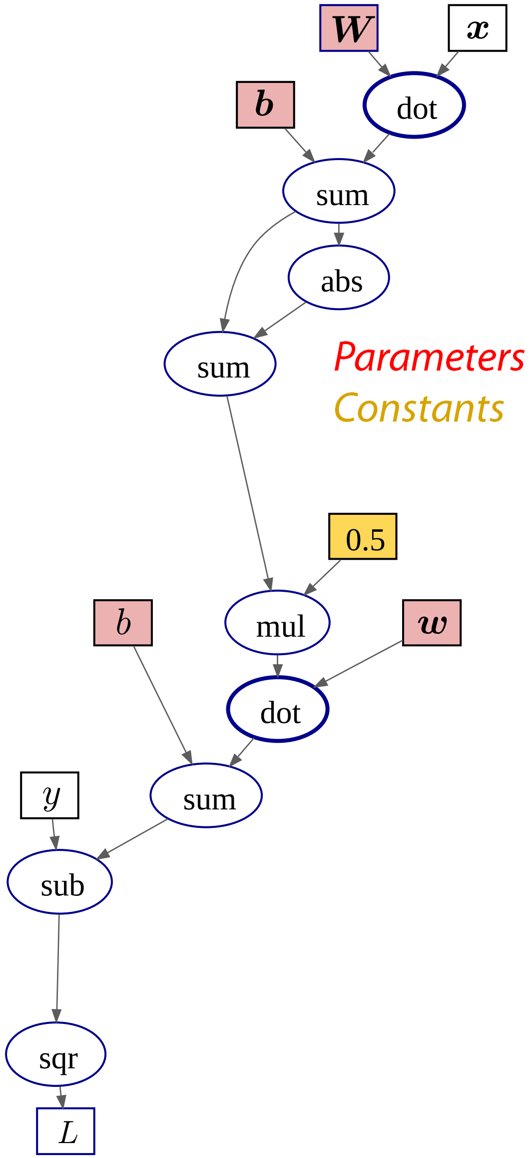
An aside: Flow Graph



$$L(\tilde{y}, y) = (\boldsymbol{w} \cdot \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b - y)^2$$

Item-wise loss function, FF neural network with ReLU as non-linearity

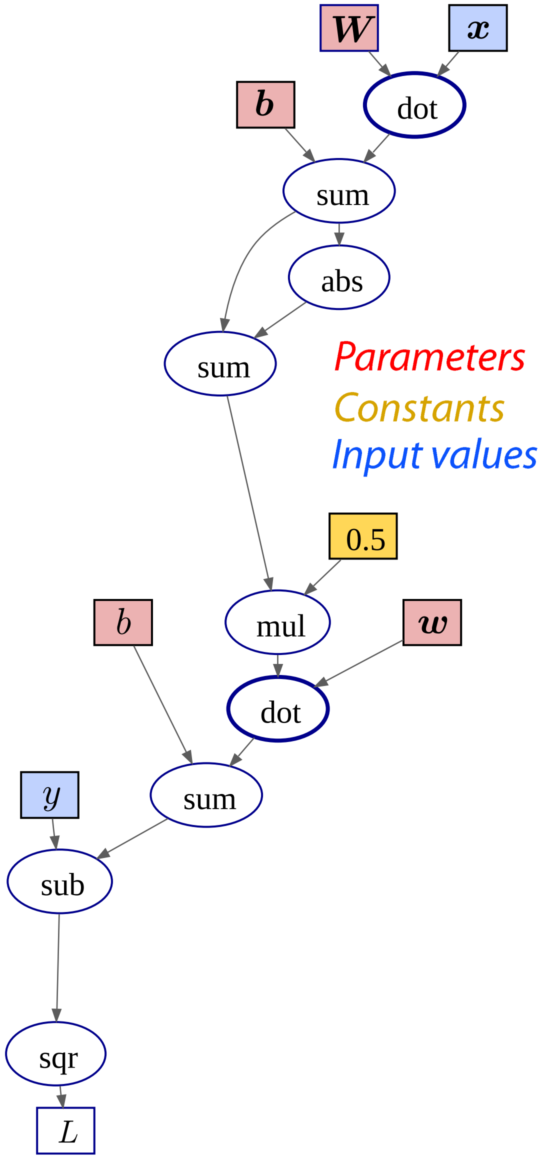
An aside: Flow Graph



$$L(\tilde{y}, y) = (\boldsymbol{w} \cdot \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b - y)^2$$

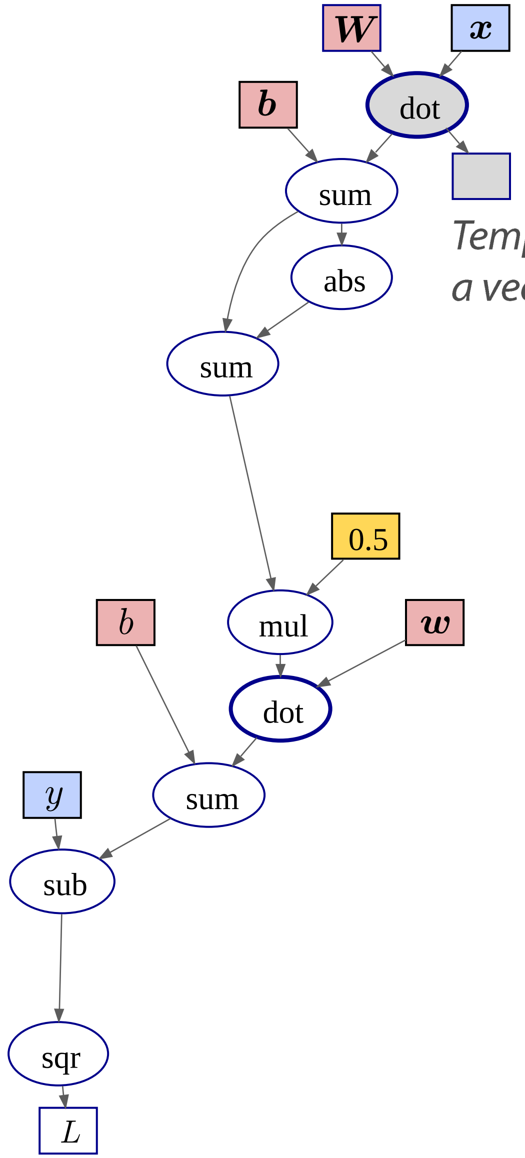
Item-wise loss function, FF neural network with ReLU as non-linearity

An aside: Flow Graph



$$L(\tilde{y}, y) = (\boldsymbol{w} \cdot \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b - y)^2$$

Item-wise loss function, FF neural network with ReLU as non-linearity



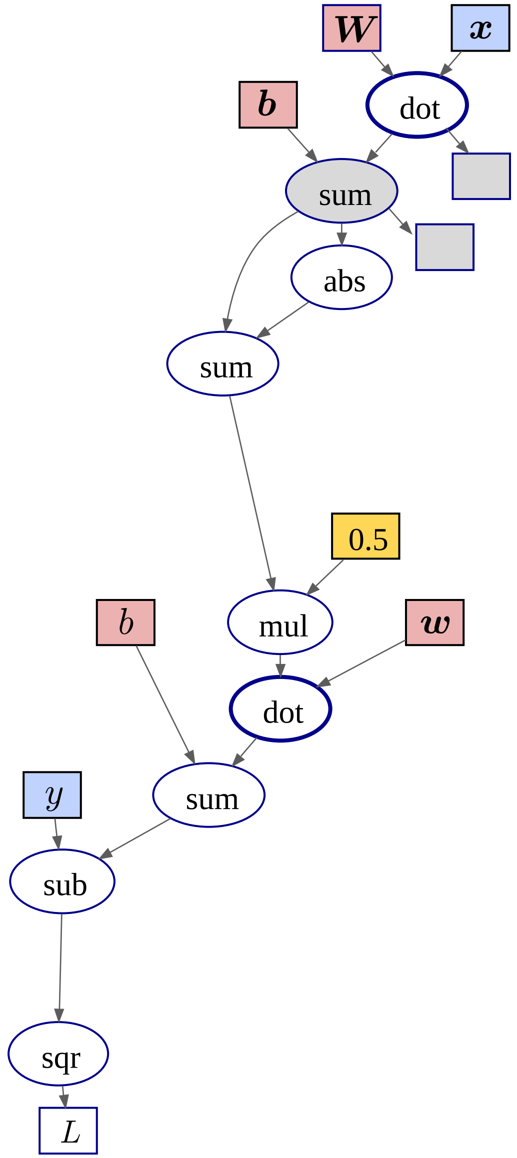
Temporary value: a vector

- ***Computing the Flow Graph***

$$L(\tilde{y}, y) = (\boldsymbol{w} \cdot \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b - y)^2$$

Item-wise loss function, FF neural network with ReLU as non-linearity

An aside: Flow Graph



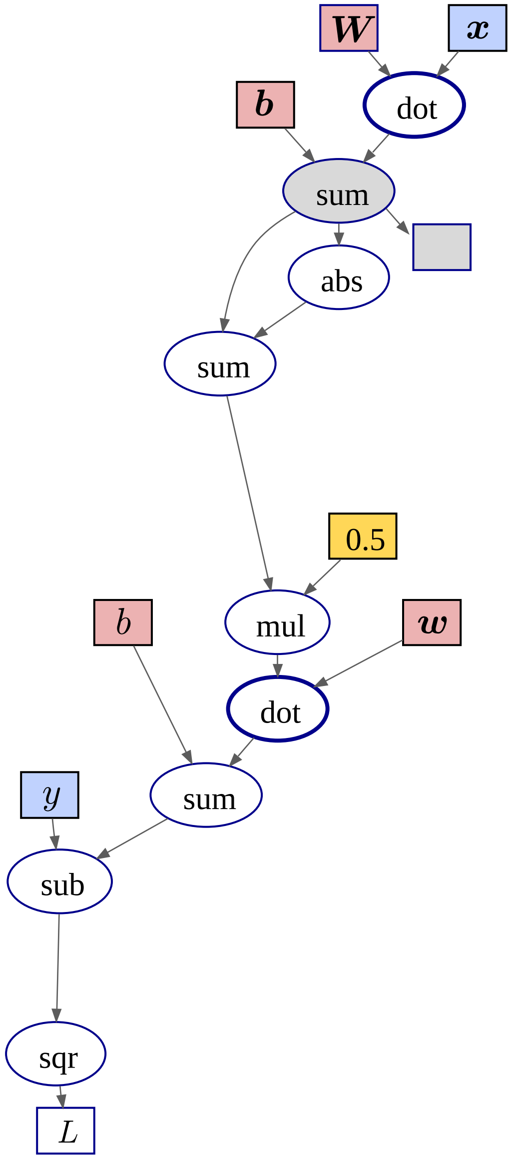
This is no longer necessary

- ***Computing the Flow Graph***

$$L(\tilde{y}, y) = (\boldsymbol{w} \cdot \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b - y)^2$$

Item-wise loss function, FF neural network with ReLU as non-linearity

An aside: Flow Graph

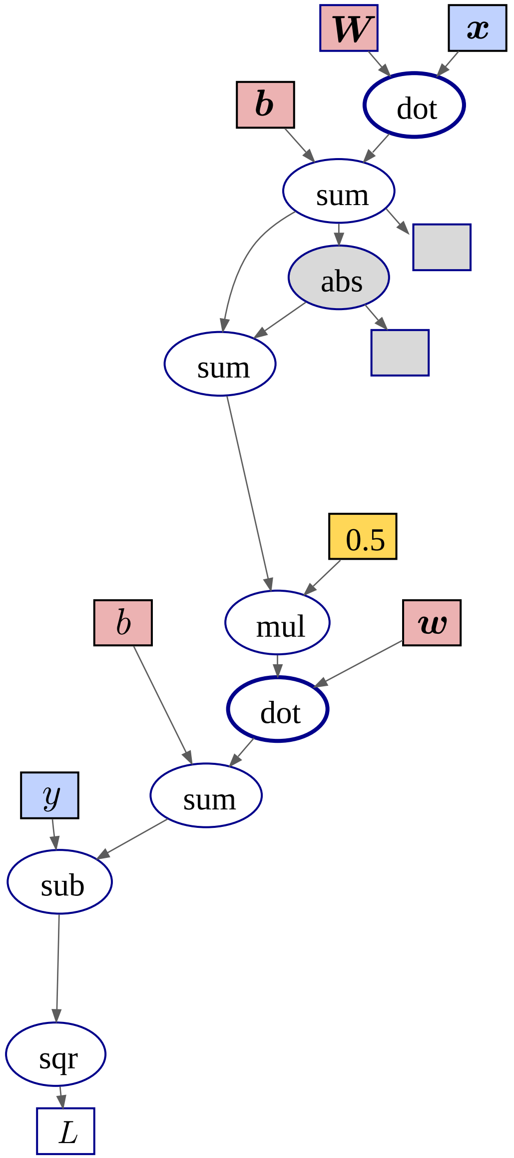


- ***Computing the Flow Graph***

$$L(\tilde{y}, y) = (\boldsymbol{w} \cdot \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b - y)^2$$

Item-wise loss function, FF neural network with ReLU as non-linearity

An aside: Flow Graph

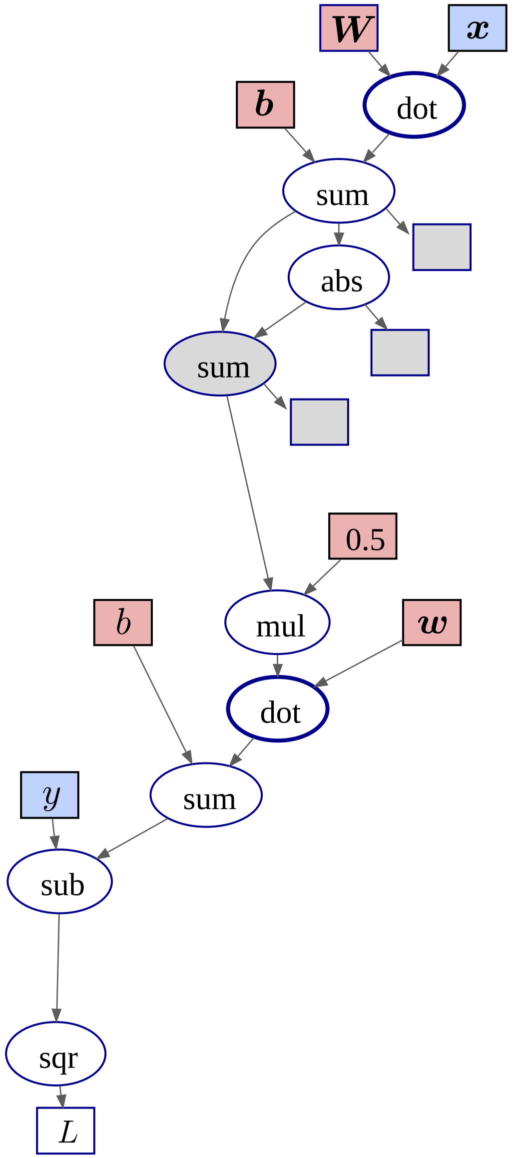


- ***Computing the Flow Graph***

$$L(\tilde{y}, y) = (\boldsymbol{w} \cdot \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b - y)^2$$

Item-wise loss function, FF neural network with ReLU as non-linearity

An aside: Flow Graph

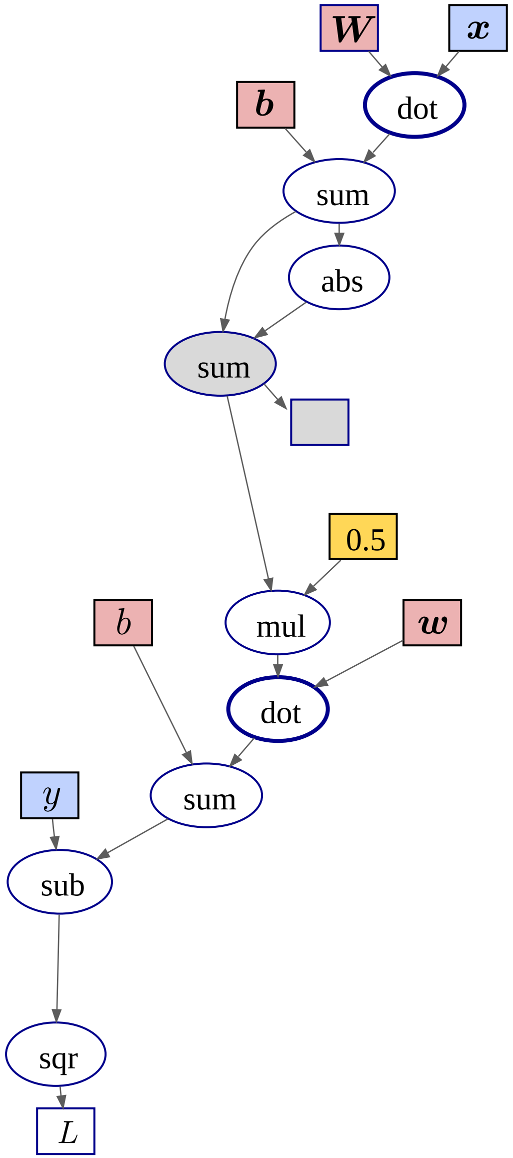


- ***Computing the Flow Graph***

$$L(\tilde{y}, y) = (\boldsymbol{w} \cdot \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b - y)^2$$

Item-wise loss function, FF neural network with ReLU as non-linearity

An aside: Flow Graph

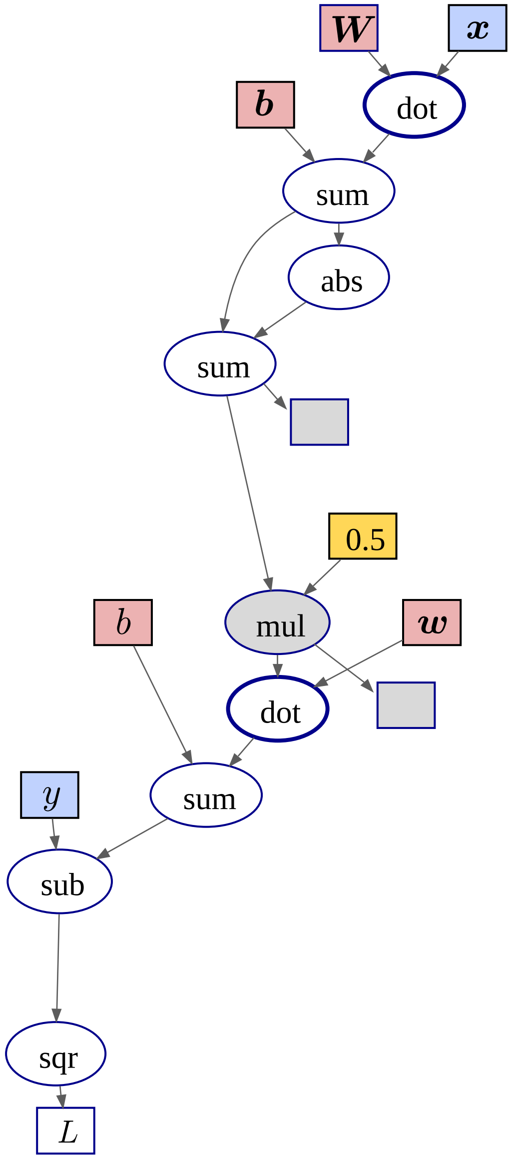


- **Computing the Flow Graph**

$$L(\tilde{y}, y) = (\boldsymbol{w} \cdot \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b - y)^2$$

Item-wise loss function, FF neural network with ReLU as non-linearity

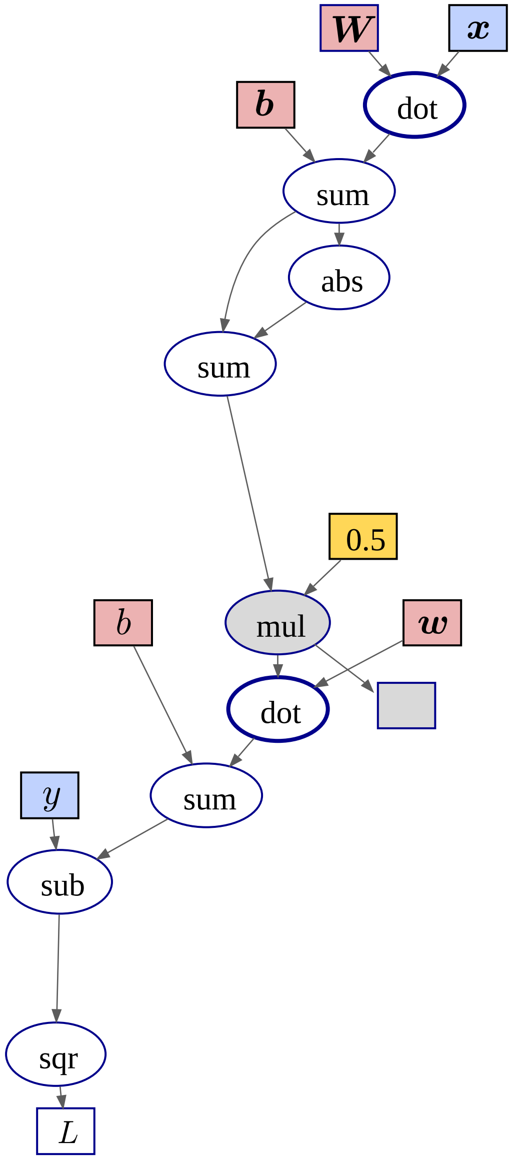
An aside: Flow Graph



- ***Computing the Flow Graph***

$$L(\tilde{y}, y) = (\boldsymbol{w} \cdot \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b - y)^2$$

Item-wise loss function, FF neural network with ReLU as non-linearity

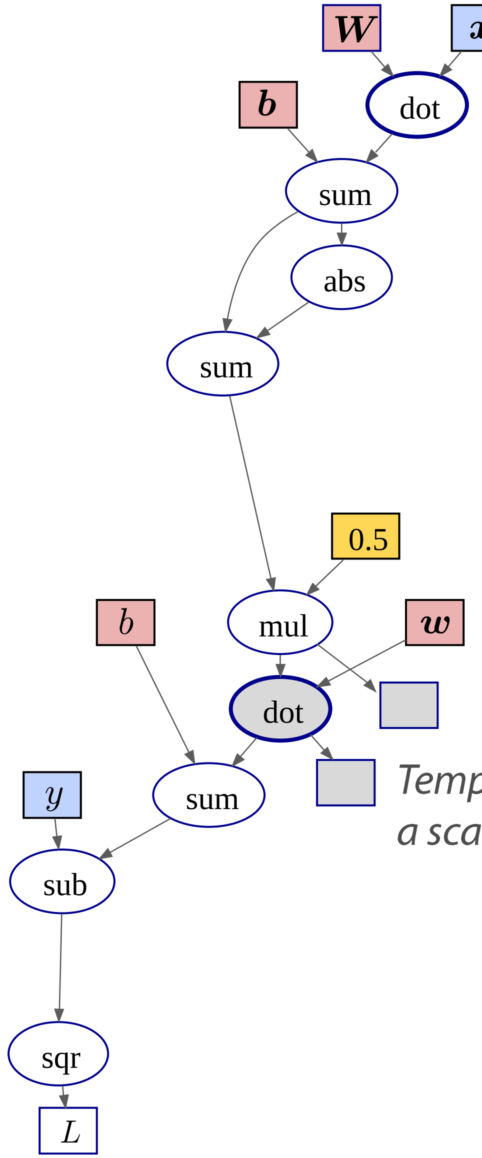


- **Computing the Flow Graph**

$$L(\tilde{y}, y) = (\boldsymbol{w} \cdot \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b - y)^2$$

Item-wise loss function, FF neural network with ReLU as non-linearity

An aside: Flow Graph

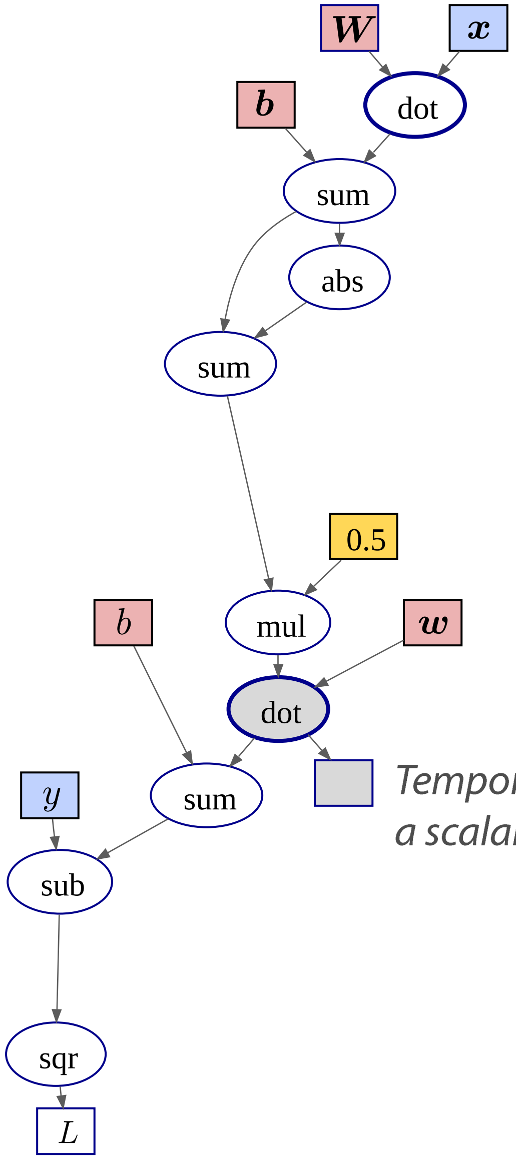


Temporary value: a scalar

- **Computing the Flow Graph**

$$L(\tilde{y}, y) = (\boldsymbol{w} \cdot \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b - y)^2$$

Item-wise loss function, FF neural network with ReLU as non-linearity

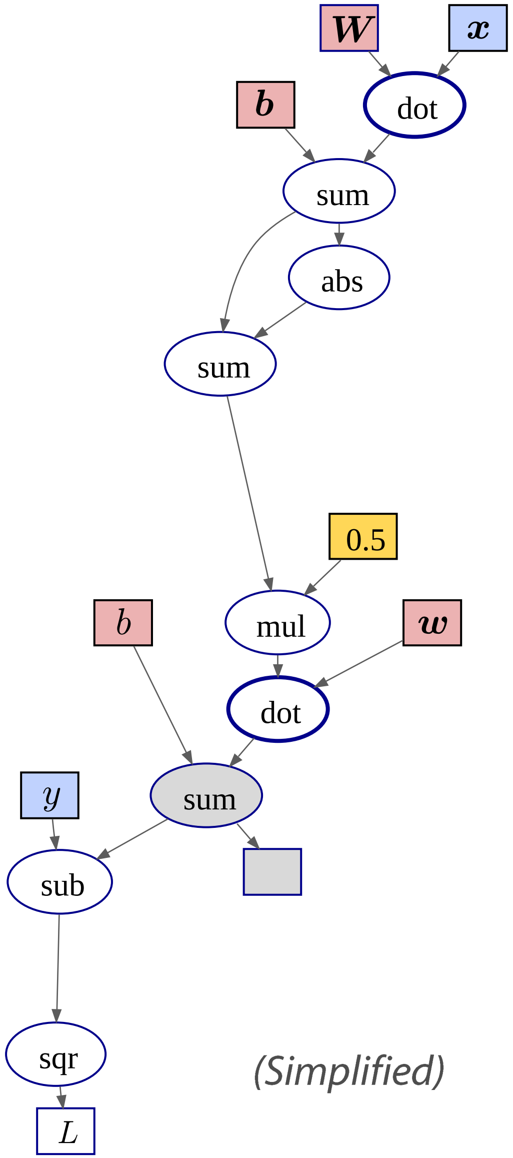


Temporary value: a scalar

- ***Computing the Flow Graph***

$$L(\tilde{y}, y) = (\boldsymbol{w} \cdot \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b - y)^2$$

Item-wise loss function, FF neural network with ReLU as non-linearity

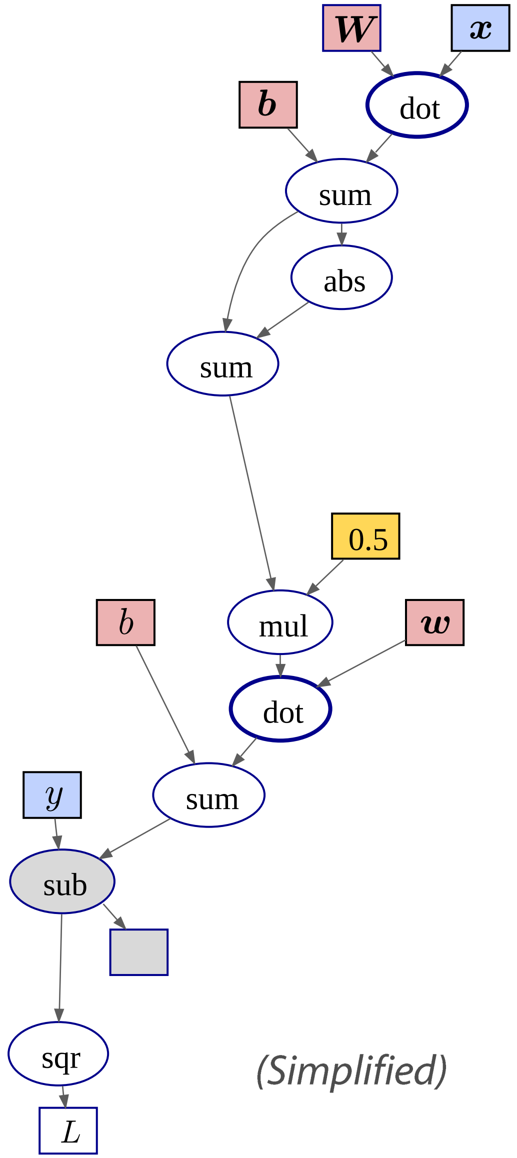


(Simplified)

- **Computing the Flow Graph**

$$L(\tilde{y}, y) = (\boldsymbol{w} \cdot \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b - y)^2$$

Item-wise loss function, FF neural network with ReLU as non-linearity



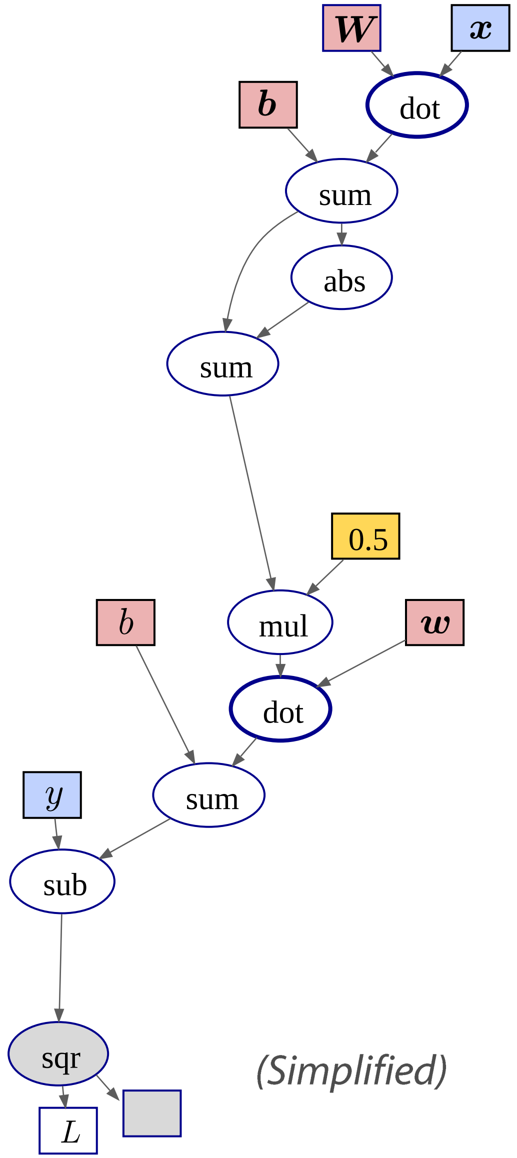
(Simplified)

- ***Computing the Flow Graph***

$$L(\tilde{y}, y) = (\boldsymbol{w} \cdot \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b - y)^2$$

Item-wise loss function, FF neural network with ReLU as non-linearity

An aside: Flow Graph



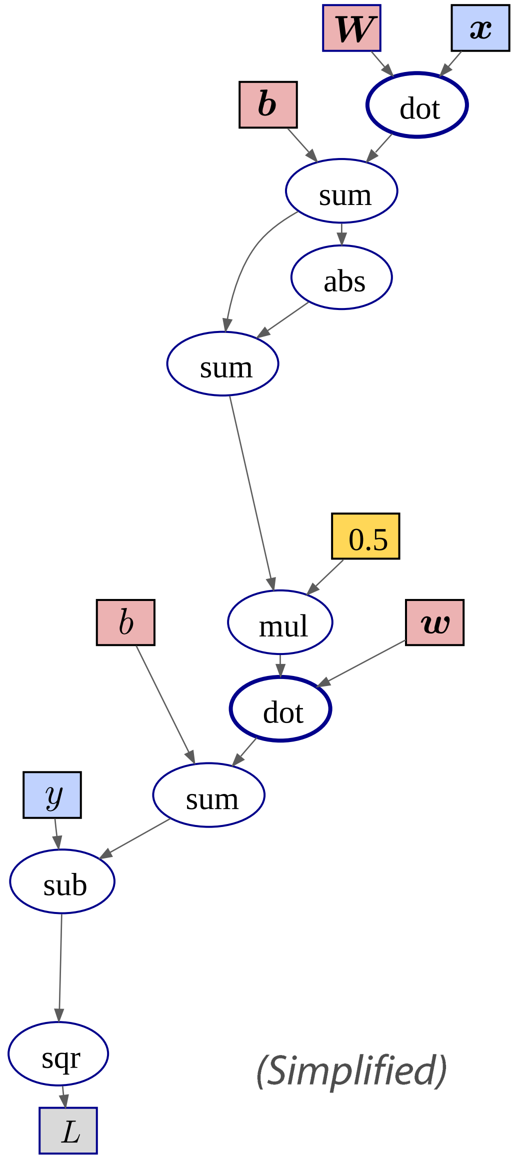
(Simplified)

- **Computing the Flow Graph**

$$L(\tilde{y}, y) = (\boldsymbol{w} \cdot \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b - y)^2$$

Item-wise loss function, FF neural network with ReLU as non-linearity

An aside: Flow Graph



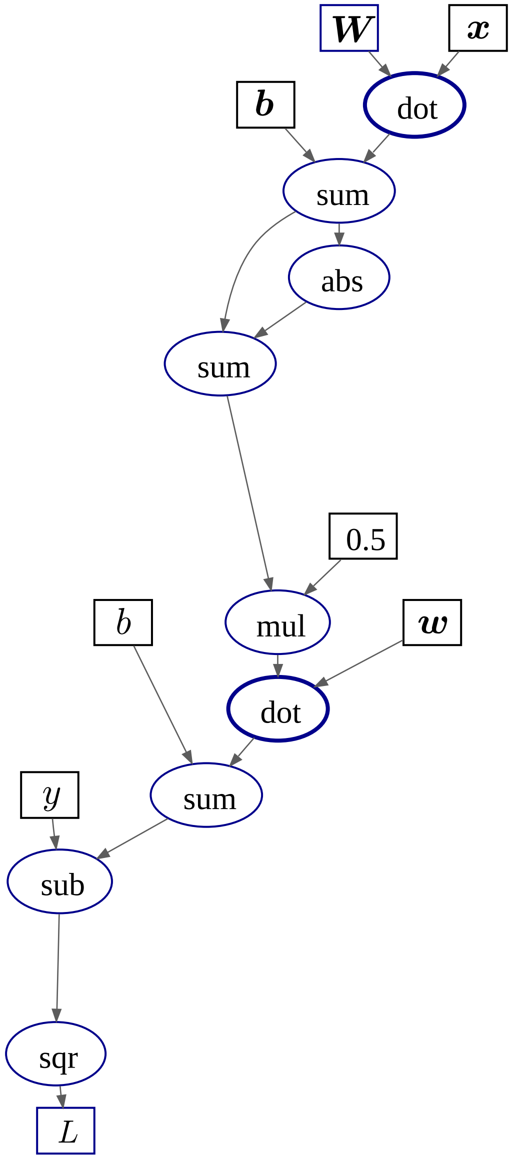
(Simplified)

- ***Computing the Flow Graph***

$$L(\tilde{y}, y) = (\boldsymbol{w} \cdot \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b - y)^2$$

Item-wise loss function, FF neural network with ReLU as non-linearity

Autodiff:
Automatic Differentiation of Flow Graphs

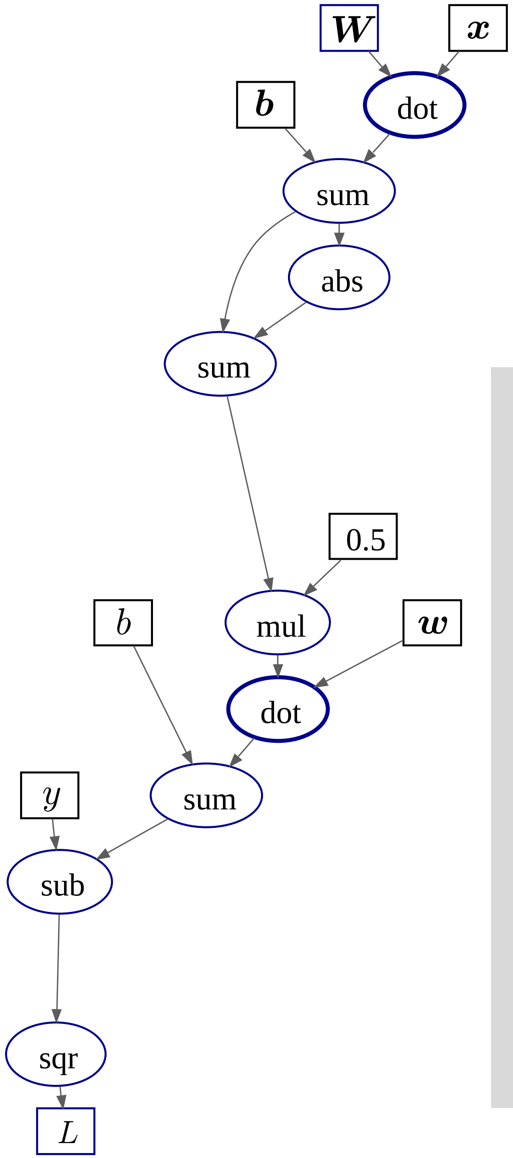


- **Computing one gradient of the flow graph**

$$\frac{\partial}{\partial \boldsymbol{W}}(\boldsymbol{w} \cdot \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b - y)^2$$

This is the gradient we want to compute
(remember this is just one of the four)

Computing Gradients



- ***Computing one gradient of the flow graph***

$$\frac{\partial}{\partial \boldsymbol{W}}(\boldsymbol{w} \cdot \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b - y)^2$$

This is the gradient we want to compute (remember this is just one of the four)

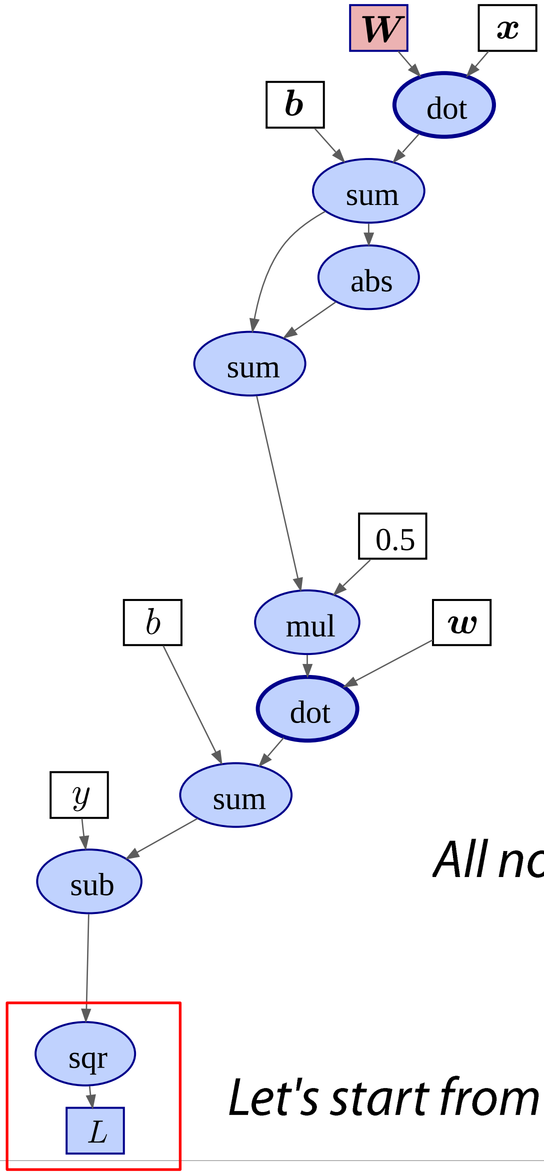
Chain rule for derivatives (*single argument*)

$$\frac{\partial}{\partial \boldsymbol{\vartheta}} f(g(\boldsymbol{\vartheta})) = \frac{\partial}{\partial g(\boldsymbol{\vartheta})} f(g(\boldsymbol{\vartheta})) \frac{\partial}{\partial \boldsymbol{\vartheta}} g(\boldsymbol{\vartheta})$$

Chain rule for derivatives (*multiple arguments*)

$$\frac{\partial}{\partial \boldsymbol{\vartheta}} f(g(\boldsymbol{\vartheta}), h(\boldsymbol{\vartheta})) = \frac{\partial}{\partial g(\boldsymbol{\vartheta})} f(g(\boldsymbol{\vartheta}), h(\boldsymbol{\vartheta})) \frac{\partial}{\partial \boldsymbol{\vartheta}} g(\boldsymbol{\vartheta}) \; + \; \frac{\partial}{\partial h(\boldsymbol{\vartheta})} f(g(\boldsymbol{\vartheta}), h(\boldsymbol{\vartheta})) \frac{\partial}{\partial \boldsymbol{\vartheta}} h(\boldsymbol{\vartheta})$$

Computing Gradients

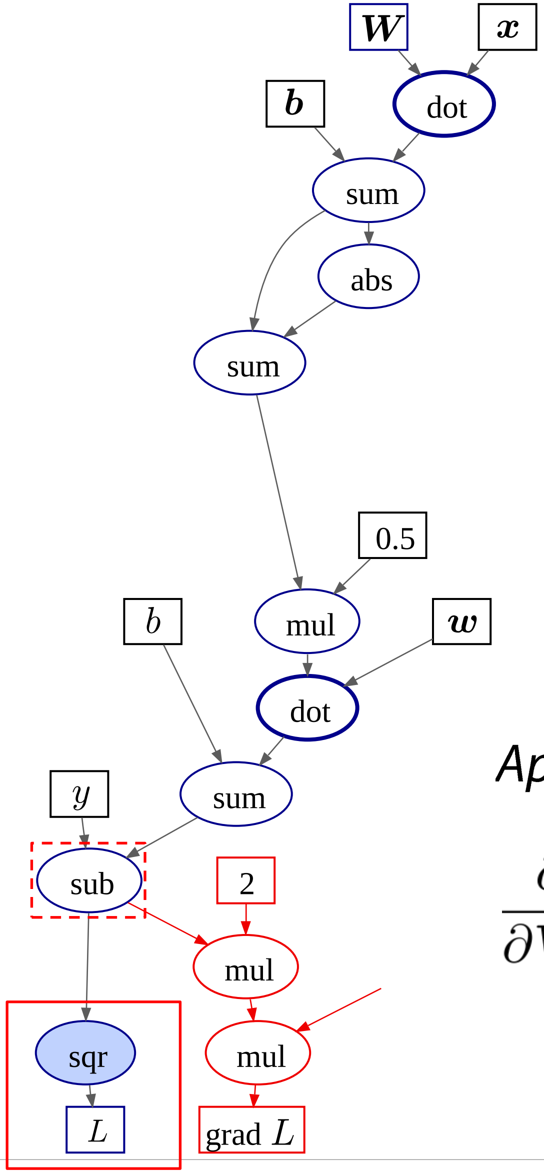


$$\frac{\partial}{\partial \boldsymbol{W}}(\boldsymbol{w} \cdot \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b - y)^2$$

All nodes depending on $\boldsymbol{W}$ are marked in blue

*Let's start from here (i.e. **backpropagation**, a.k.a. **reverse accumulation**)*

Computing Gradients



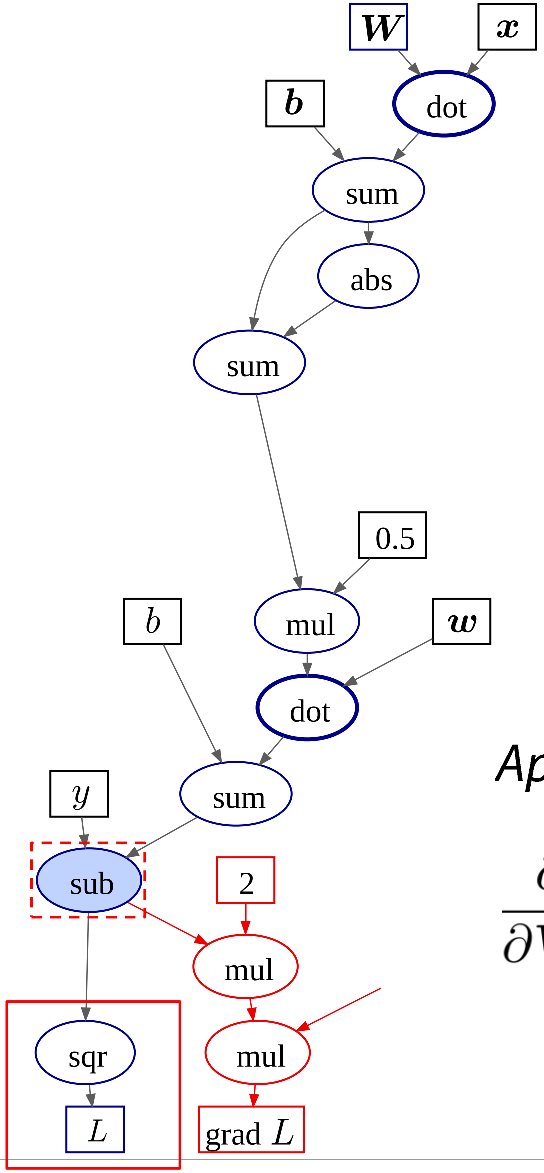
$$\frac{\partial}{\partial \boldsymbol{W}}(\boldsymbol{w} \cdot \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b - y)^2$$

Apply the chain rule to the sqr *node*

$$\frac{\partial}{\partial \boldsymbol{W}} f(\boldsymbol{W})^2 = \frac{\partial}{\partial f(\boldsymbol{W})} f(\boldsymbol{W})^2 \frac{\partial}{\partial \boldsymbol{W}} f(\boldsymbol{W})$$

$$= 2 \cdot f(\boldsymbol{W}) \cdot \frac{\partial}{\partial \boldsymbol{W}} f(\boldsymbol{W})$$

Computing Gradients



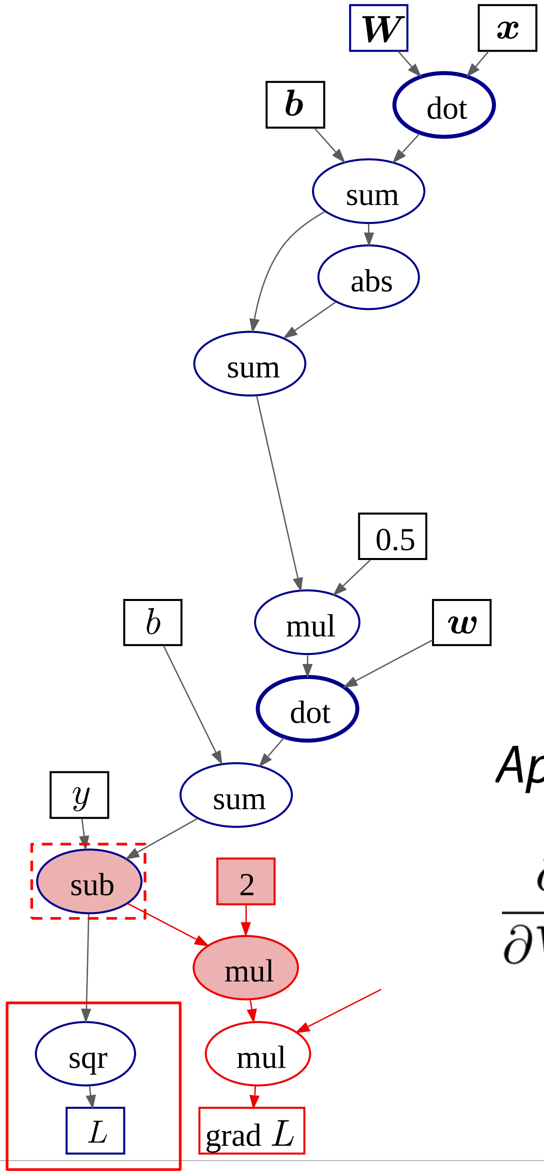
$$\frac{\partial}{\partial \boldsymbol{W}}(\boldsymbol{w} \cdot \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b - y)^2$$

Apply the chain rule to the sqr *node*

$$\frac{\partial}{\partial \boldsymbol{W}} f(\boldsymbol{W})^2 = \frac{\partial}{\partial f(\boldsymbol{W})} f(\boldsymbol{W})^2 \frac{\partial}{\partial \boldsymbol{W}} f(\boldsymbol{W})$$

$$= 2 \cdot f(\boldsymbol{W}) \cdot \frac{\partial}{\partial \boldsymbol{W}} f(\boldsymbol{W})$$

Computing Gradients



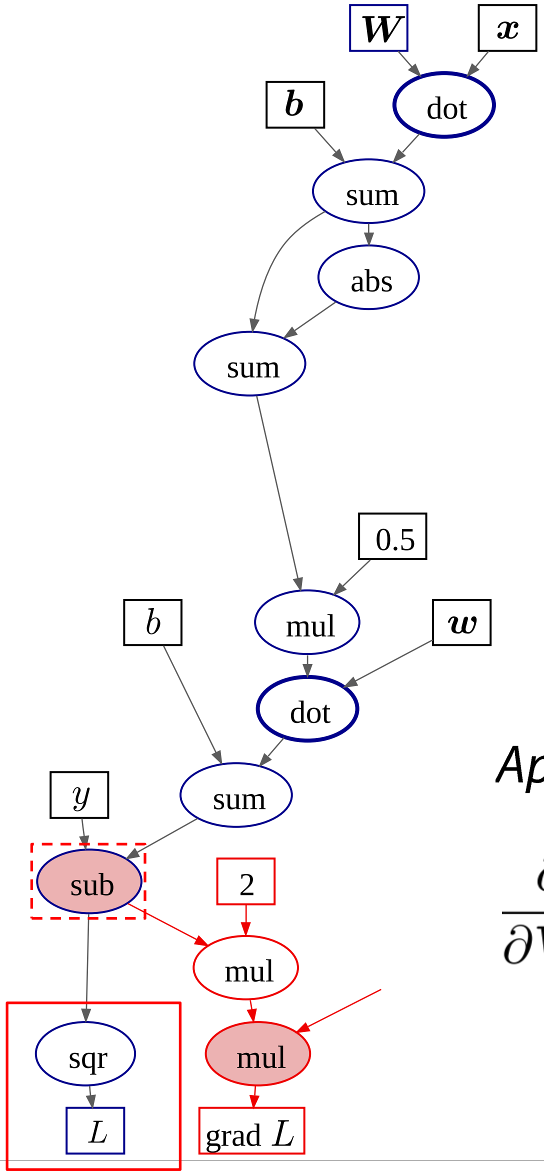
$$\frac{\partial}{\partial \boldsymbol{W}}(\boldsymbol{w} \cdot \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b - y)^2$$

Apply the chain rule to the sqr *node*

$$\frac{\partial}{\partial \boldsymbol{W}} f(\boldsymbol{W})^2 = \frac{\partial}{\partial f(\boldsymbol{W})} f(\boldsymbol{W})^2 \frac{\partial}{\partial \boldsymbol{W}} f(\boldsymbol{W})$$

$$= 2 \cdot f(\boldsymbol{W}) \cdot \frac{\partial}{\partial \boldsymbol{W}} f(\boldsymbol{W})$$

Computing Gradients



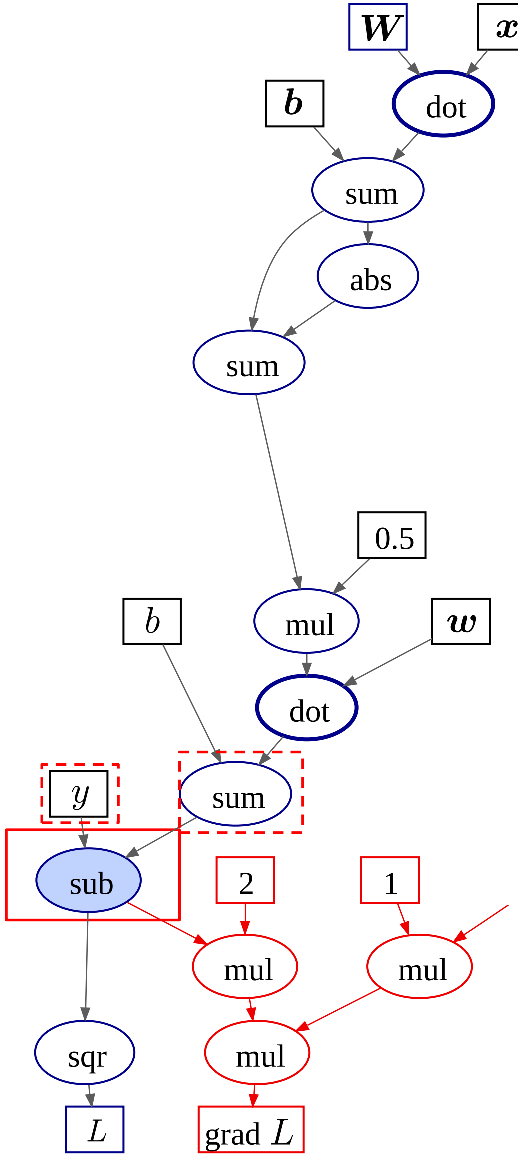
$$\frac{\partial}{\partial \boldsymbol{W}}(\boldsymbol{w} \cdot \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b - y)^2$$

Apply the chain rule to the sqr *node*

$$\frac{\partial}{\partial \boldsymbol{W}} f(\boldsymbol{W})^2 = \frac{\partial}{\partial f(\boldsymbol{W})} f(\boldsymbol{W})^2 \frac{\partial}{\partial \boldsymbol{W}} f(\boldsymbol{W})$$

$$= 2 \cdot f(\boldsymbol{W}) \cdot \frac{\partial}{\partial \boldsymbol{W}} f(\boldsymbol{W})$$

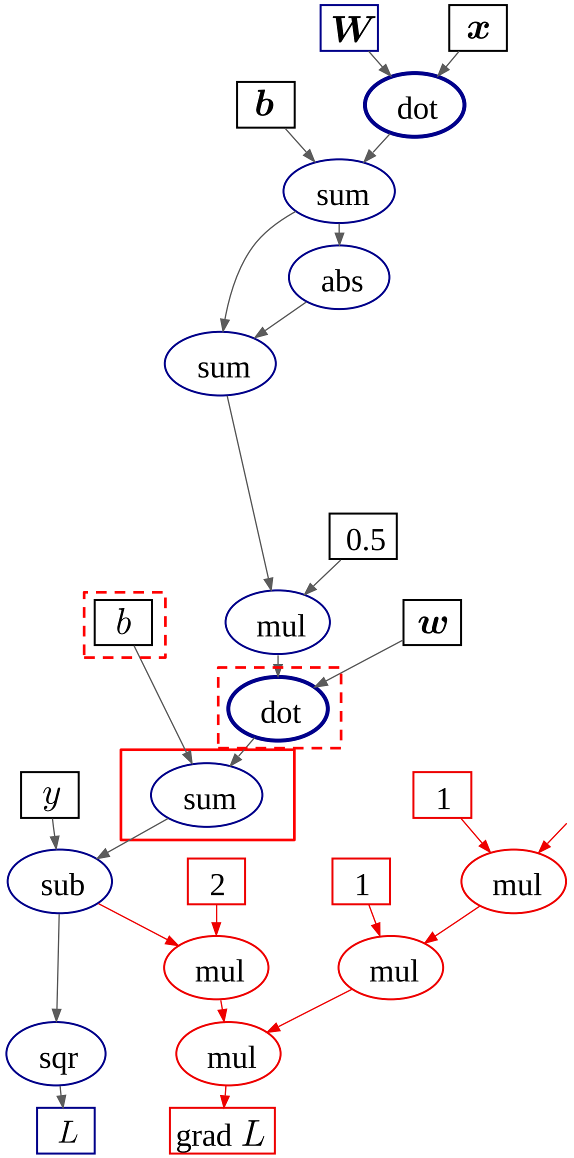
Computing Gradients



$$\frac{\partial}{\partial \boldsymbol{W}}(\boldsymbol{w} \cdot \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b - y)^2$$

$$\frac{\partial}{\partial \boldsymbol{W}}(f(\boldsymbol{W}) - y) = \frac{\partial}{\partial f(\boldsymbol{W})}(f(\boldsymbol{W}) - y)\frac{\partial}{\partial \boldsymbol{W}} f(\boldsymbol{W})$$
$$= 1 \cdot \frac{\partial}{\partial \boldsymbol{W}} f(\boldsymbol{W})$$

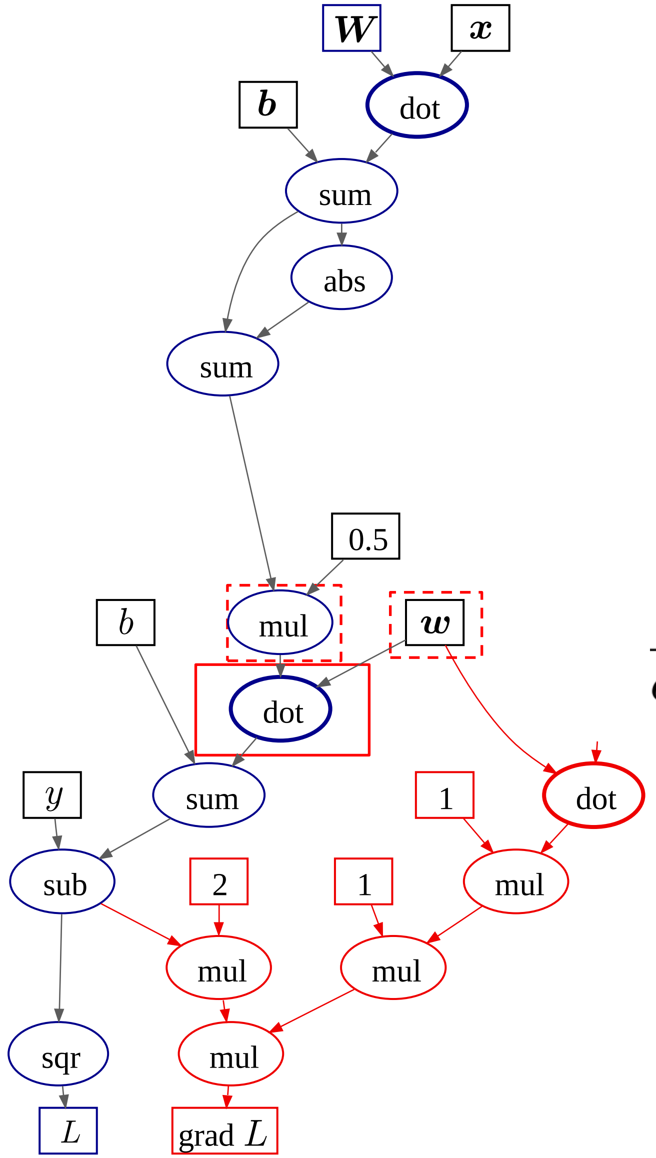
Computing Gradients



$$\frac{\partial}{\partial \boldsymbol{W}}(\boldsymbol{w} \cdot \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b - y)^2$$

$$\frac{\partial}{\partial \boldsymbol{W}}(f(\boldsymbol{W}) + b) = \frac{\partial}{\partial f(\boldsymbol{W})}(f(\boldsymbol{W}) + b)\frac{\partial}{\partial \boldsymbol{W}} f(\boldsymbol{W})$$
$$= 1 \cdot \frac{\partial}{\partial \boldsymbol{W}} f(\boldsymbol{W})$$

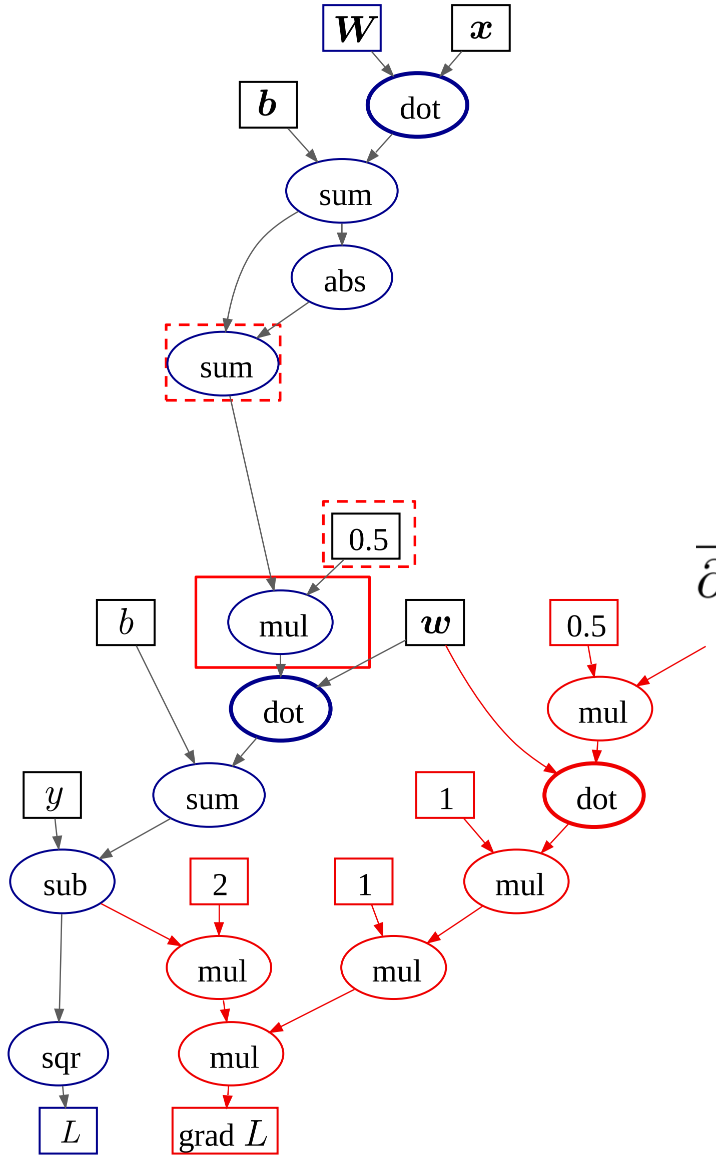
Computing Gradients



$$\frac{\partial}{\partial \boldsymbol{W}}(\boldsymbol{w} \cdot \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b - y)^2$$

$$\frac{\partial}{\partial \boldsymbol{W}}(\boldsymbol{w} \cdot f(\boldsymbol{W})) = \frac{\partial}{\partial f(\boldsymbol{W})}(\boldsymbol{w} \cdot f(\boldsymbol{W}))\frac{\partial}{\partial \boldsymbol{W}} f(\boldsymbol{W})$$
$$= \boldsymbol{w} \cdot \frac{\partial}{\partial \boldsymbol{W}} f(\boldsymbol{W})$$

Computing Gradients

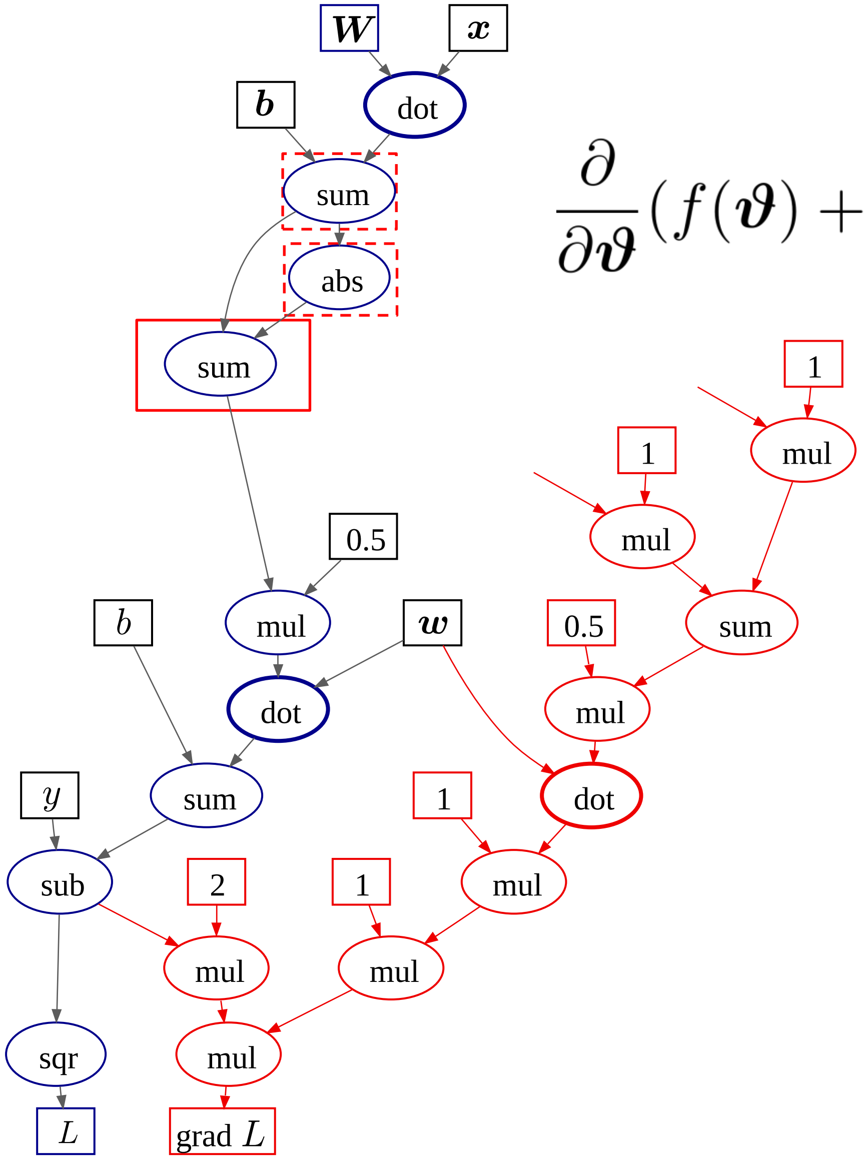


$$\frac{\partial}{\partial \boldsymbol{W}}(\boldsymbol{w} \cdot \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b - y)^2$$

$$\frac{\partial}{\partial \boldsymbol{W}}(0.5 \cdot f(\boldsymbol{W})) = \frac{\partial}{\partial f(\boldsymbol{W})}(0.5 \cdot f(\boldsymbol{W}))\frac{\partial}{\partial \boldsymbol{W}}f(\boldsymbol{W})$$

$$= 0.5 \cdot \frac{\partial}{\partial \boldsymbol{W}}f(\boldsymbol{W})$$

Computing Gradients

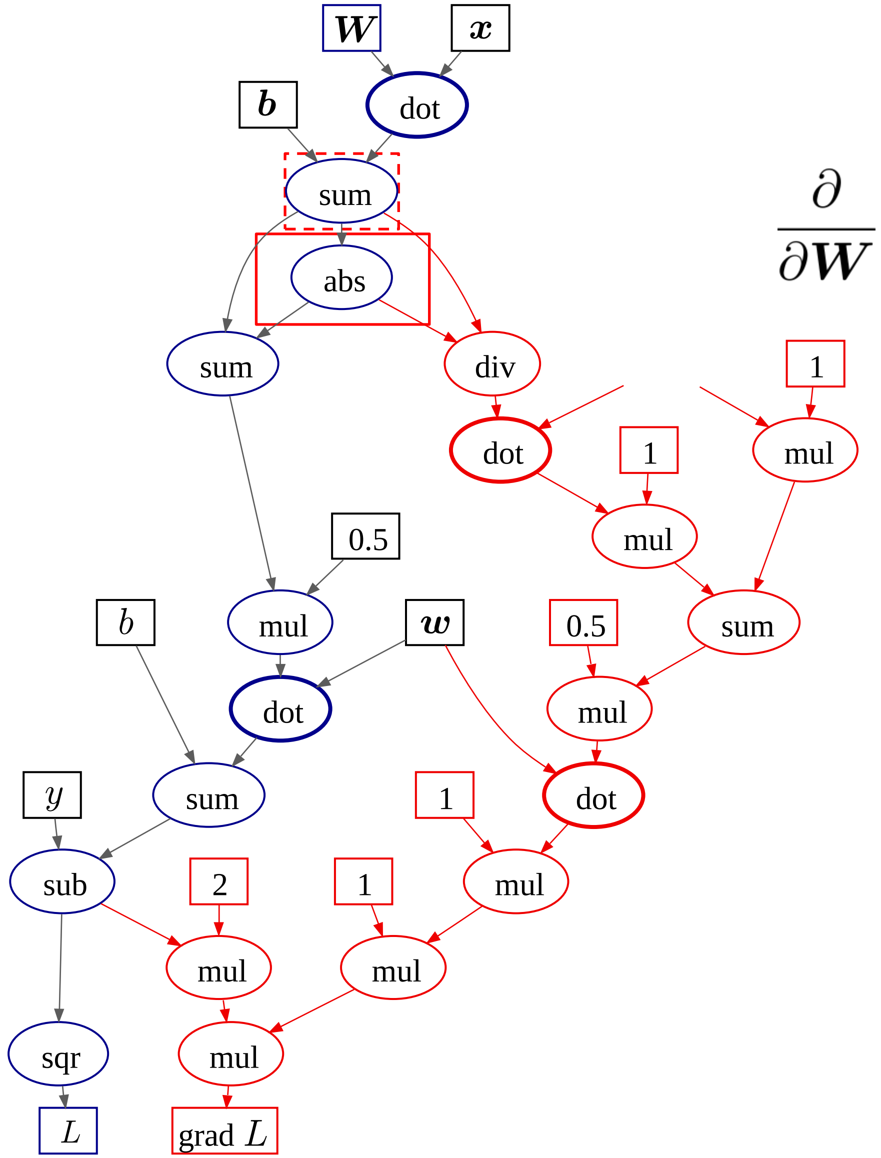


$$\frac{\partial}{\partial \boldsymbol{\vartheta}}(f(\boldsymbol{\vartheta}) + g(\boldsymbol{\vartheta})) = \frac{\partial}{\partial f(\boldsymbol{\vartheta})}(f(\boldsymbol{\vartheta}) + g(\boldsymbol{\vartheta}))\frac{\partial}{\partial \boldsymbol{\vartheta}} f(\boldsymbol{\vartheta})$$

$$+ \frac{\partial}{\partial g(\boldsymbol{\vartheta})}(f(\boldsymbol{\vartheta}) + g(\boldsymbol{\vartheta}))\frac{\partial}{\partial \boldsymbol{\vartheta}} g(\boldsymbol{\vartheta})$$

$$= 1 \cdot \frac{\partial}{\partial \boldsymbol{\vartheta}} f(\boldsymbol{\vartheta}) + 1 \cdot \frac{\partial}{\partial \boldsymbol{\vartheta}} g(\boldsymbol{\vartheta})$$

Computing Gradients



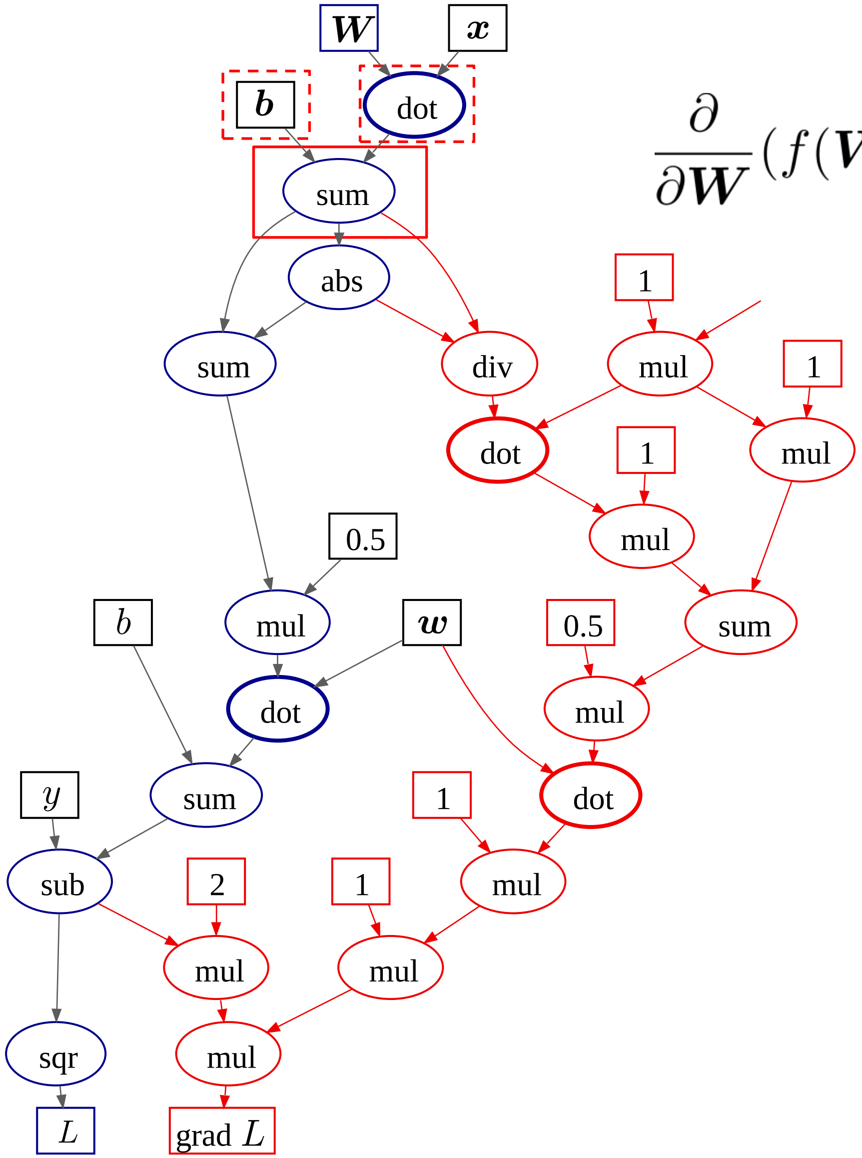
$$\frac{\partial}{\partial \boldsymbol{W}}|f(\boldsymbol{W})| = \frac{\partial}{\partial f(\boldsymbol{W})}|f(\boldsymbol{W})|\frac{\partial}{\partial \boldsymbol{W}}f(\boldsymbol{W})$$

$$= \frac{f(\boldsymbol{W})}{|f(\boldsymbol{W})|} \cdot \frac{\partial}{\partial \boldsymbol{W}}f(\boldsymbol{W})$$

Clearly, this term is not defined for any $W_{ij} = 0$

(Typically, this is a <u>*protected*</u> *division* $\frac{x}{0} := 1$ *)*

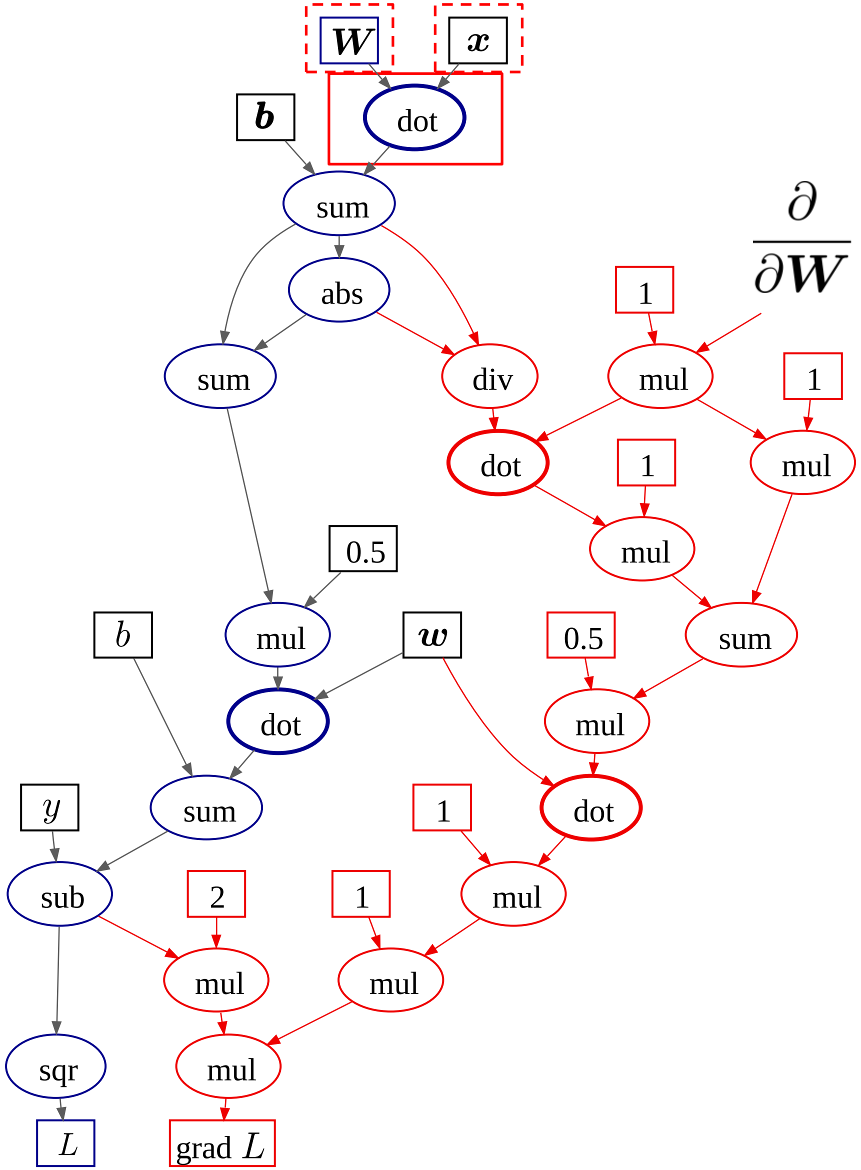
Computing Gradients



$$\frac{\partial}{\partial \boldsymbol{W}}(f(\boldsymbol{W}) + \boldsymbol{b}) = \frac{\partial}{\partial f(\boldsymbol{W})}(f(\boldsymbol{W}) + \boldsymbol{b})\frac{\partial}{\partial \boldsymbol{W}} f(\boldsymbol{W})$$

$$= 1 \cdot \frac{\partial}{\partial \boldsymbol{W}} f(\boldsymbol{W})$$

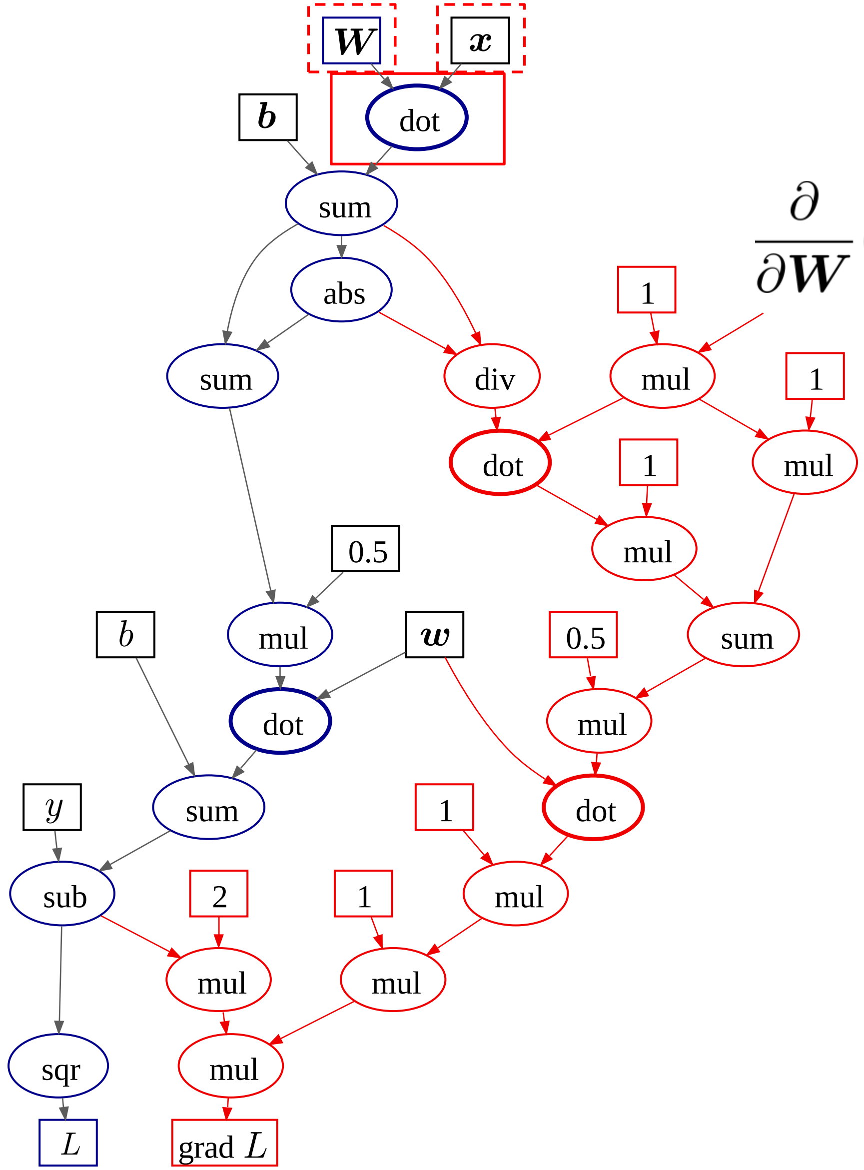
Computing Gradients



$$\frac{\partial}{\partial \boldsymbol{W}}(\boldsymbol{W} \cdot \boldsymbol{x})$$

Well, this is tricky….

Computing Gradients



$$\frac{\partial}{\partial \boldsymbol{W}}(\boldsymbol{W} \cdot \boldsymbol{x})$$

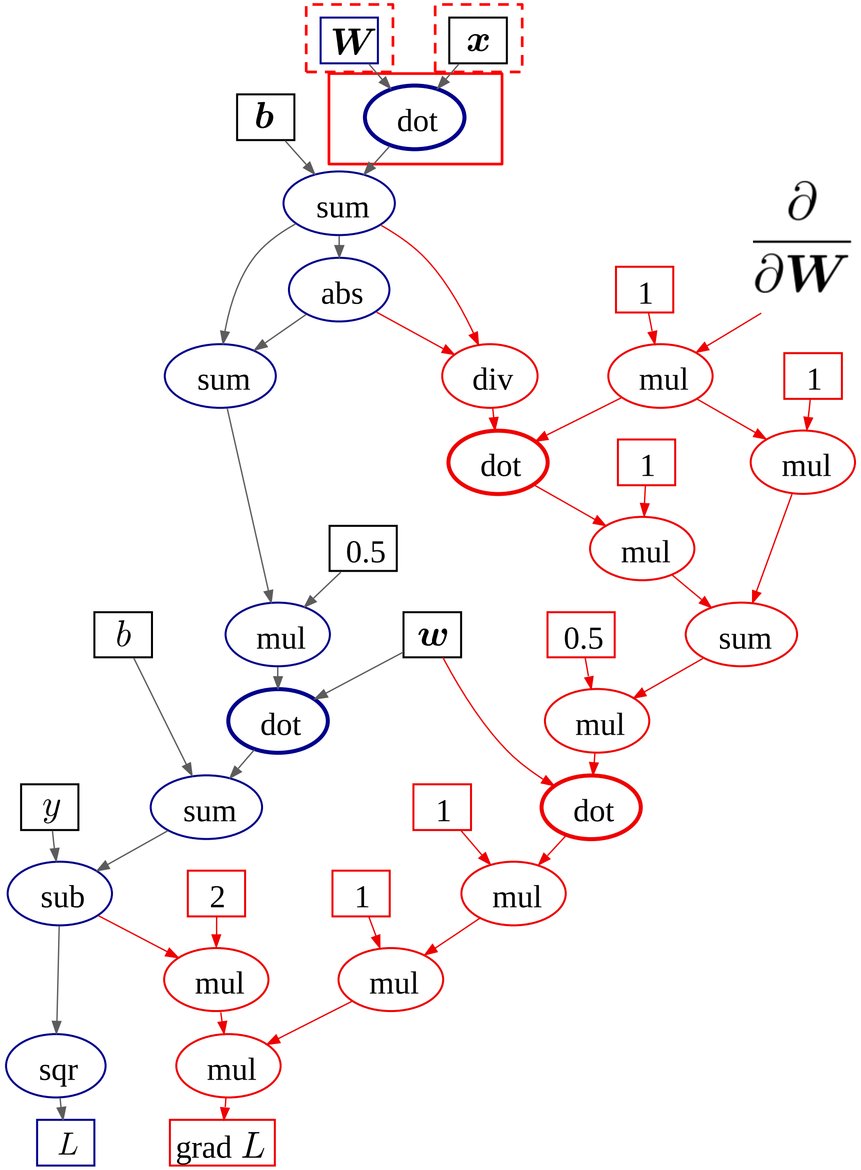
This is a third-order tensor

$$\left(\frac{\partial}{\partial \boldsymbol{W}}(\boldsymbol{W} \cdot \boldsymbol{x})\right)_{ijk} = \frac{\partial}{\partial W_{kj}}(\boldsymbol{W} \cdot \boldsymbol{x})_i$$

Its ijk-th component

Note the inversion of indices

Computing Gradients



$$\frac{\partial}{\partial \boldsymbol{W}}(\boldsymbol{W} \cdot \boldsymbol{x})$$

This is a third-order tensor

$$\left(\frac{\partial}{\partial \boldsymbol{W}}(\boldsymbol{W} \cdot \boldsymbol{x})\right)_{ijk} = \frac{\partial}{\partial W_{kj}}(\boldsymbol{W} \cdot \boldsymbol{x})_i$$

Its ijk-th component

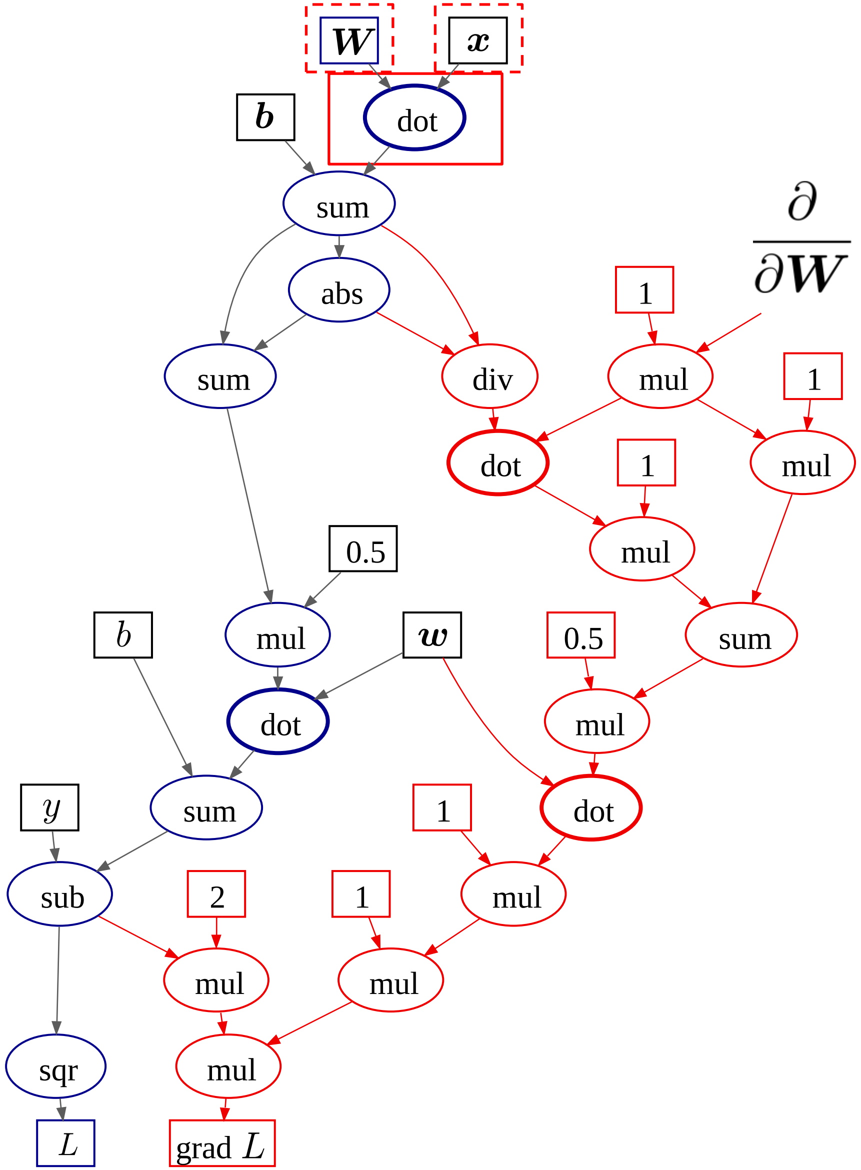
Note the inversion of indices

$$= \frac{\partial}{\partial W_{kj}}(\boldsymbol{W}_{i,:} \cdot \boldsymbol{x})$$

The i-th line in the matrix

$$= \begin{cases} 0 & k \neq i \\ x_j & k = i \end{cases}$$

Computing Gradients

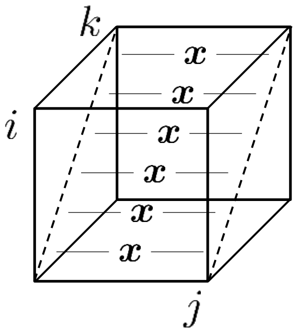


$$\frac{\partial}{\partial \boldsymbol{W}}(\boldsymbol{W} \cdot \boldsymbol{x})$$

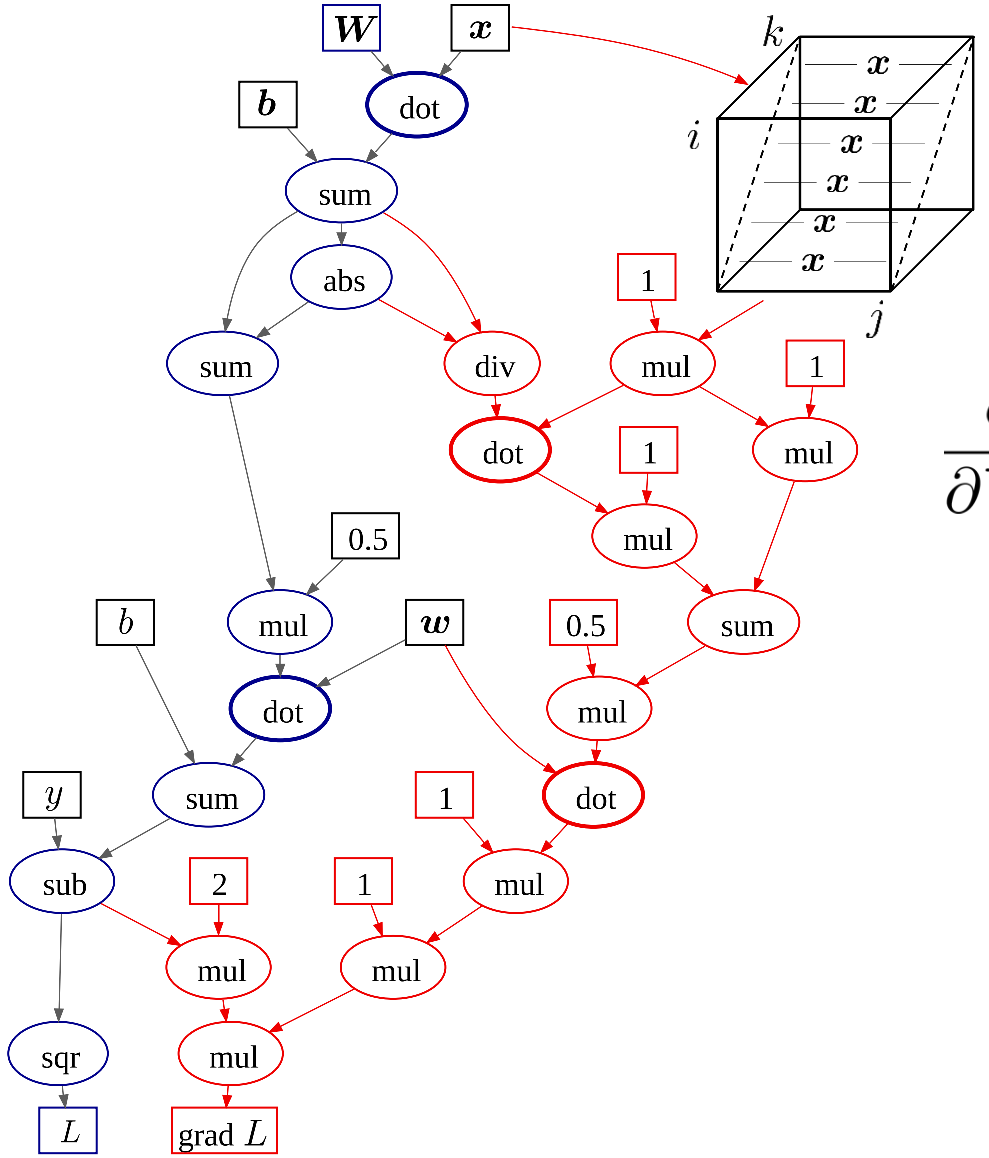
Putting it all together…

$$\left(\frac{\partial}{\partial \boldsymbol{W}}(\boldsymbol{W} \cdot \boldsymbol{x})\right)_{ijk} = \begin{cases} 0 & k \neq i \\ x_j & k = i \end{cases}$$

This 'thing' is a cube having copies of $\boldsymbol{x}$ *on one diagonal 'plane' and zeros elsewhere*

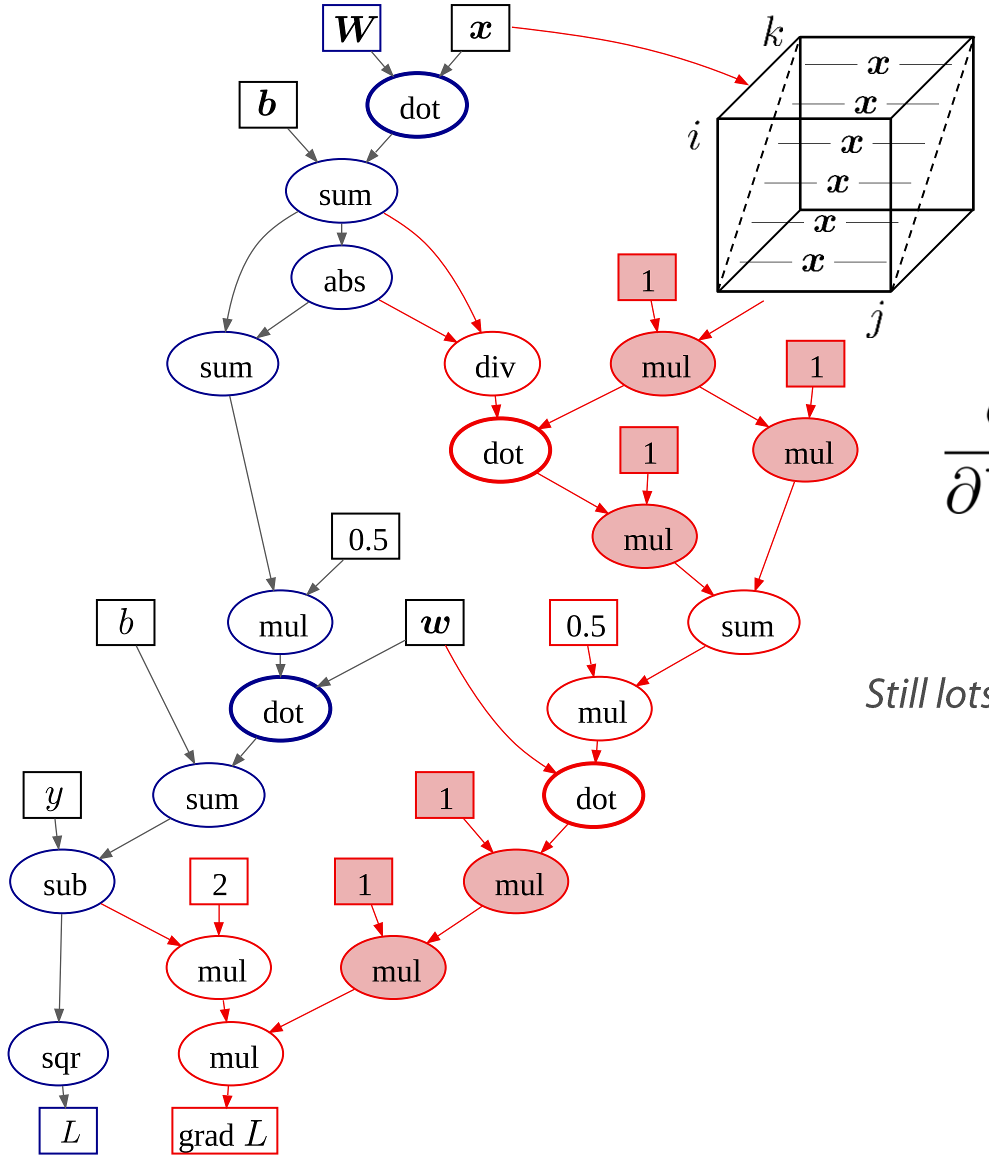


Computing Gradients



$$\frac{\partial}{\partial \boldsymbol{W}} (\boldsymbol{w} \cdot \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b - y)^2$$

Computing Gradients

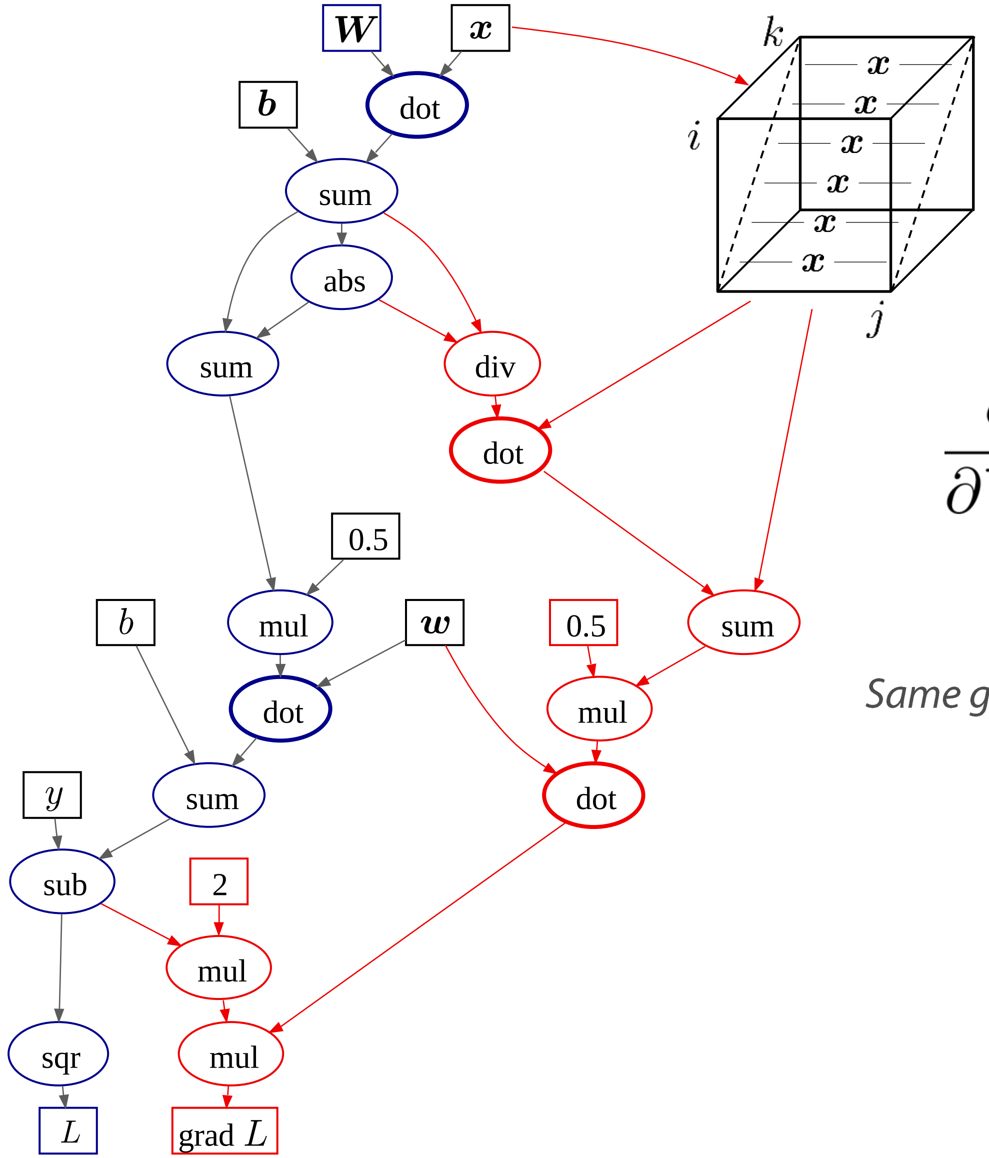


The representation of this can be optimized too

$$\frac{\partial}{\partial \boldsymbol{W}}(\boldsymbol{w} \cdot \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b - y)^2$$

Still lots of useless operations

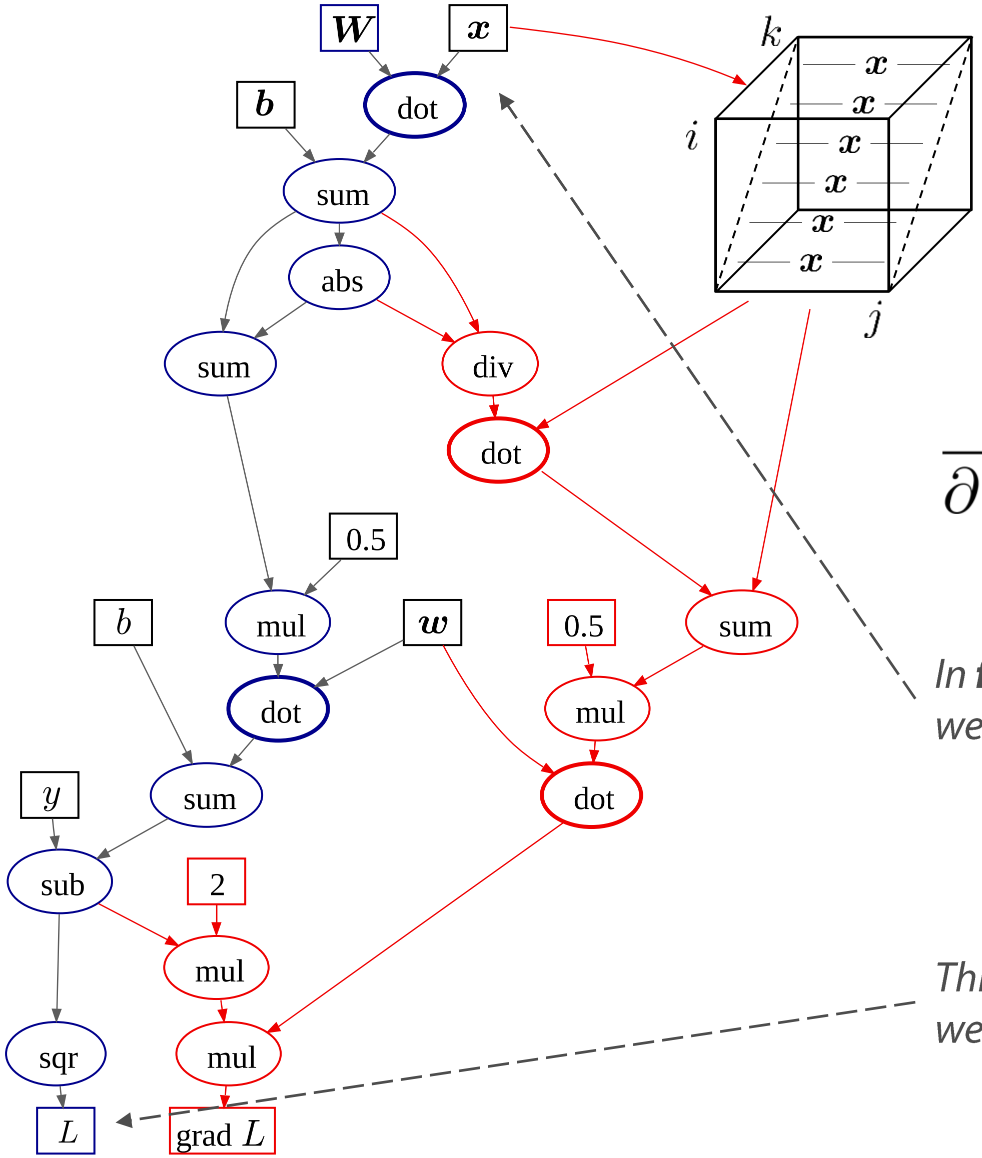
Computing Gradients



$$\frac{\partial}{\partial \boldsymbol{W}}(\boldsymbol{w} \cdot \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b - y)^2$$

Same graph, after some pruning

Computing Gradients



$$\frac{\partial}{\partial \boldsymbol{W}}(\boldsymbol{w} \cdot \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b - y)^2$$

*In **forward accumulation** mode*
we would have started from here

*This is autodiff with **reverse accumulation:***
we started from here and we proceeded in reverse

(Mini) Batches in Matrix Form

More on Matrix Forms

Say it with matrices…

We may want to get rid of the summation when computing the loss function

$$L(D) = \frac{1}{N} \sum_{D} ((\boldsymbol{w} \cdot g(\boldsymbol{W}\boldsymbol{x}^{(i)} + \boldsymbol{b}) + b) - y^{(i)})^2$$

Let's focus first on $\boldsymbol{W}\boldsymbol{x}$

by defining

$$\boldsymbol{X} := \begin{bmatrix} x_1^{(1)} & \dots & x_d^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(N)} & \dots & x_d^{(N)} \end{bmatrix}$$

*input data in matrix form (**item index first**)*

Then we can write

$$\boldsymbol{W}\boldsymbol{X}^T = \begin{bmatrix} | & & | \\ \boldsymbol{W}\boldsymbol{x}^{(1)} & \dots & \boldsymbol{W}\boldsymbol{x}^{(N)} \\ | & & | \end{bmatrix}$$

More on Matrix Forms

Say it with matrices…

We may want to get rid of the summation when computing the loss function

$$L(D) = \frac{1}{N} \sum_D ((\boldsymbol{w} \cdot g(\boldsymbol{W}\boldsymbol{x}^{(i)} + \boldsymbol{b}) + b) - y^{(i)})^2$$

Consider then $(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$

by defining

$$\hat{\boldsymbol{X}} := \begin{bmatrix} x_1^{(1)} & \dots & x_d^{(1)} & 1 \\ \vdots & \ddots & \vdots & \vdots \\ x_1^{(N)} & \dots & x_d^{(N)} & 1 \end{bmatrix} \qquad \hat{\boldsymbol{W}} := \begin{bmatrix} \boldsymbol{W} & \Big| \, \boldsymbol{b} \, \Big| \end{bmatrix}$$

Then we could write

$$\hat{\boldsymbol{W}}\hat{\boldsymbol{X}}^T = \begin{bmatrix} | & & | \\ \boldsymbol{W}\boldsymbol{x}^{(1)} + \boldsymbol{b} & \dots & \boldsymbol{W}\boldsymbol{x}^{(N)} + \boldsymbol{b} \\ | & & | \end{bmatrix}$$

Matrix $\hat{\boldsymbol{X}}$ includes two parameters: $\boldsymbol{W}$ *and* $\boldsymbol{b}$

this may be inconvenient for autodiff...

More on Matrix Forms

Say it with matrices…

We may want to get rid of the summation when computing the loss function

$$L(D) = \frac{1}{N} \sum_{D} ((\boldsymbol{w} \cdot g(\boldsymbol{W}\boldsymbol{x}^{(i)} + \boldsymbol{b}) + b) - y^{(i)})^2$$

Consider then $(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$

and let's keep the definition

$$\boldsymbol{X} := \begin{bmatrix} x_1^{(1)} & \dots & x_d^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(N)} & \dots & x_d^{(N)} \end{bmatrix}$$

It could be convenient to redefine the operator $+$ such that is interpreted as

$$\boldsymbol{W}\boldsymbol{X}^T + \boldsymbol{b} := \begin{bmatrix} | & & | \\ \boldsymbol{W}\boldsymbol{x}^{(1)} & \dots & \boldsymbol{W}\boldsymbol{x}^{(N)} \\ | & & | \end{bmatrix} + \underbrace{\begin{bmatrix} | & & | \\ \boldsymbol{b} & \dots & \boldsymbol{b} \\ | & & | \end{bmatrix}}_{N \text{ times}}$$

More on Matrix Forms

Say it with matrices…

We may want to get rid of the summation when computing the loss function

$$L(D) = \frac{1}{N}\sum_{D}((\boldsymbol{w}\cdot g(\boldsymbol{W}\boldsymbol{x}^{(i)} + \boldsymbol{b}) + b) - y^{(i)})^2$$

Consider then $(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$

and let's keep the definition

$$\boldsymbol{X} := \begin{bmatrix} x_1^{(1)} & \dots & x_d^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(N)} & \dots & x_d^{(N)} \end{bmatrix}$$

It could be convenient to redefine the operator $+$ such that is interpreted as

$$\boldsymbol{W}\boldsymbol{X}^T + \boldsymbol{b} := \begin{bmatrix} | & & | \\ \boldsymbol{W}\boldsymbol{x}^{(1)} & \dots & \boldsymbol{W}\boldsymbol{x}^{(N)} \\ | & & | \end{bmatrix} + \underbrace{\begin{bmatrix} | & & | \\ \boldsymbol{b} & \dots & \boldsymbol{b} \\ | & & | \end{bmatrix}}_{N \text{ times}}$$

*This is called **broadcasting***

More on Matrix Forms

Say it with matrices…

We may want to get rid of the summation when computing the loss function

$$L(D) = \frac{1}{N}\sum_{D}((\boldsymbol{w}\cdot g(\boldsymbol{W}\boldsymbol{x}^{(i)} + \boldsymbol{b}) + b) - y^{(i)})^2$$

Using broadcasting operators, we can express the above as

$$L(D) = \frac{1}{N}((\boldsymbol{w}\cdot g(\boldsymbol{W}\boldsymbol{X}^{T} + \boldsymbol{b}) + b) - \boldsymbol{y})^2$$

where

$$\boldsymbol{X} := \begin{bmatrix} x_1^{(1)} & \dots & x_d^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(N)} & \dots & x_d^{(N)} \end{bmatrix} \qquad \boldsymbol{y} := \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(N)} \end{bmatrix}$$

More on Matrix Forms

Say it with matrices…

We may want to get rid of the summation when computing the loss function

$$L(D) = \frac{1}{N} \sum_{D} ((\boldsymbol{w} \cdot g(\boldsymbol{W}\boldsymbol{x}^{(i)} + \boldsymbol{b}) + b) - y^{(i)})^2$$

Using broadcasting operators, we can express the above as

$$L(D) = \frac{1}{N}((\boldsymbol{w} \cdot g(\boldsymbol{W}\boldsymbol{X}^T + \boldsymbol{b}) + b) - \boldsymbol{y})^2$$

This is a matrix $g(\boldsymbol{W}\boldsymbol{X}^T + \boldsymbol{b}) \in \mathbb{R}^{h \times N}$
(Note the ***broadcast*** *with* $+$ *)*

More on Matrix Forms

Say it with matrices…

We may want to get rid of the summation when computing the loss function

$$L(D) = \frac{1}{N}\sum_{D}((\boldsymbol{w}\cdot g(\boldsymbol{W}\boldsymbol{x}^{(i)}+\boldsymbol{b})+b)-y^{(i)})^2$$

Using broadcasting operators, we can express the above as

$$L(D) = \frac{1}{N}((\boldsymbol{w}\cdot g(\boldsymbol{W}\boldsymbol{X}^T+\boldsymbol{b})+b)-\boldsymbol{y})^2$$

This is a ***row*** *vector*

$$\boldsymbol{w}\cdot g(\boldsymbol{W}\boldsymbol{X}^T+\boldsymbol{b}) = \boldsymbol{w}^T g(\boldsymbol{W}\boldsymbol{X}^T+\boldsymbol{b}) \in \mathbb{R}^N$$

(The 'dot' operator ***transposes <u>vectors</u>*** *automatically, as required)*

NOTE: automatic transposition applies ***to vectors only****!*
For any tensor beyond dimension 1, you need to do that on your own

More on Matrix Forms

Say it with matrices…

We may want to get rid of the summation when computing the loss function

$$L(D) = \frac{1}{N} \sum_{D} ((\boldsymbol{w} \cdot g(\boldsymbol{W}\boldsymbol{x}^{(i)} + \boldsymbol{b}) + b) - y^{(i)})^2$$

Using broadcasting operators, we can express the above as

$$L(D) = \frac{1}{N} ((\boldsymbol{w} \cdot g(\boldsymbol{W}\boldsymbol{X}^T + \boldsymbol{b}) + b) - \boldsymbol{y})^2$$

*This is also a **row** vector $\in \mathbb{R}^N$, after a **broadcast** on b*

More on Matrix Forms

Say it with matrices…

We may want to get rid of the summation when computing the loss function

$$L(D) = \frac{1}{N} \sum_{D} ((\boldsymbol{w} \cdot g(\boldsymbol{W}\boldsymbol{x}^{(i)} + \boldsymbol{b}) + b) - y^{(i)})^2$$

Using broadcasting operators, we can express the above as

$$L(D) = \frac{1}{N}((\boldsymbol{w} \cdot g(\boldsymbol{W}\boldsymbol{X}^T + \boldsymbol{b}) + b) - \boldsymbol{y})^2$$

*… whereas this is a **column** vector* $\in \mathbb{R}^N$

*This is also a **row** vector* $\in \mathbb{R}^N$*, after a **broadcast** on* b

More on Matrix Forms

Say it with matrices…

We may want to get rid of the summation when computing the loss function

$$L(D) = \frac{1}{N} \sum_{D} ((\boldsymbol{w} \cdot g(\boldsymbol{W}\boldsymbol{x}^{(i)} + \boldsymbol{b}) + b) - y^{(i)})^2$$

Using broadcasting operators, we can express the above as

$$L(D) = \frac{1}{N}((\boldsymbol{w} \cdot g(\boldsymbol{W}\boldsymbol{X}^T + \boldsymbol{b}) + b) - \boldsymbol{y})^2$$

*… whereas this is a **column** vector $\in \mathbb{R}^N$*

*This is also a **row** vector $\in \mathbb{R}^N$, after a **broadcast** on b*

*(Also, the $-$ operator **transposes** vectors automatically, as required)*

More on Matrix Forms

Say it with matrices…

We may want to get rid of the summation when computing the loss function

$$L(D) = \frac{1}{N} \sum_{D} ((\boldsymbol{w} \cdot g(\boldsymbol{W}\boldsymbol{x}^{(i)} + \boldsymbol{b}) + b) - y^{(i)})^2$$

Using broadcasting operators, we can express the above as

$$L(D) = \frac{1}{N}((\boldsymbol{w} \cdot g(\boldsymbol{W}\boldsymbol{X}^T + \boldsymbol{b}) + b) - \boldsymbol{y})^2$$

*… whereas this is a **column** vector* $\in \mathbb{R}^N$

*This is also a **row** vector* $\in \mathbb{R}^N$*, after a **broadcast** on* b

(Also, the $-$ *operator **transposes** vectors automatically, as required)*

A similar behavior of operators is standard in

NumPy TensorFlow PyTorch

More on Matrix Forms

Say it with matrices…

We may want to get rid of the summation when computing the loss function

$$L(D) = \frac{1}{N} \sum_{D} ((\boldsymbol{w} \cdot g(\boldsymbol{W}\boldsymbol{x}^{(i)} + \boldsymbol{b}) + b) - y^{(i)})^2$$

Using broadcasting operators, we can express the above as

$$L(D) = \frac{1}{N} ((\boldsymbol{w} \cdot g(\boldsymbol{W}\boldsymbol{X}^T + \boldsymbol{b}) + b) - \boldsymbol{y})^2$$

This is a matrix $\boldsymbol{W}\boldsymbol{X}^T \in \mathbb{R}^{h \times N}$

Ouch! No ***item index first …***

More on Matrix Forms

Say it with matrices…

We may want to get rid of the summation when computing the loss function

$$L(D) = \frac{1}{N} \sum_{D} ((\boldsymbol{w} \cdot g(\boldsymbol{W}\boldsymbol{x}^{(i)} + \boldsymbol{b}) + b) - y^{(i)})^2$$

Using broadcasting operators, we can express the above as

$$L(D) = \frac{1}{N} ((g(\boldsymbol{X}\boldsymbol{W}^T + \boldsymbol{b}) \cdot \boldsymbol{w} + b) - \boldsymbol{y})^2$$

This is a matrix $\boldsymbol{X}\boldsymbol{W}^T \in \mathbb{R}^{N \times h}$

Item index first!

More on Matrix Forms

Say it with matrices…

We may want to get rid of the summation when computing the loss function

$$L(D) = \frac{1}{N} \sum_{D} ((\boldsymbol{w} \cdot g(\boldsymbol{W}\boldsymbol{x}^{(i)} + \boldsymbol{b}) + b) - y^{(i)})^2$$

Using broadcasting operators, we can express the above as

$$L(D) = \frac{1}{N}((g(\boldsymbol{X}\boldsymbol{W}^T + \boldsymbol{b}) \cdot \boldsymbol{w} + b) - \boldsymbol{y})^2$$

This is a matrix $\boldsymbol{X}\boldsymbol{W}^T \in \mathbb{R}^{N \times h}$

This is a ***column*** *vector* $\in \mathbb{R}^h$
(it will be transposed automatically)

Item index first!