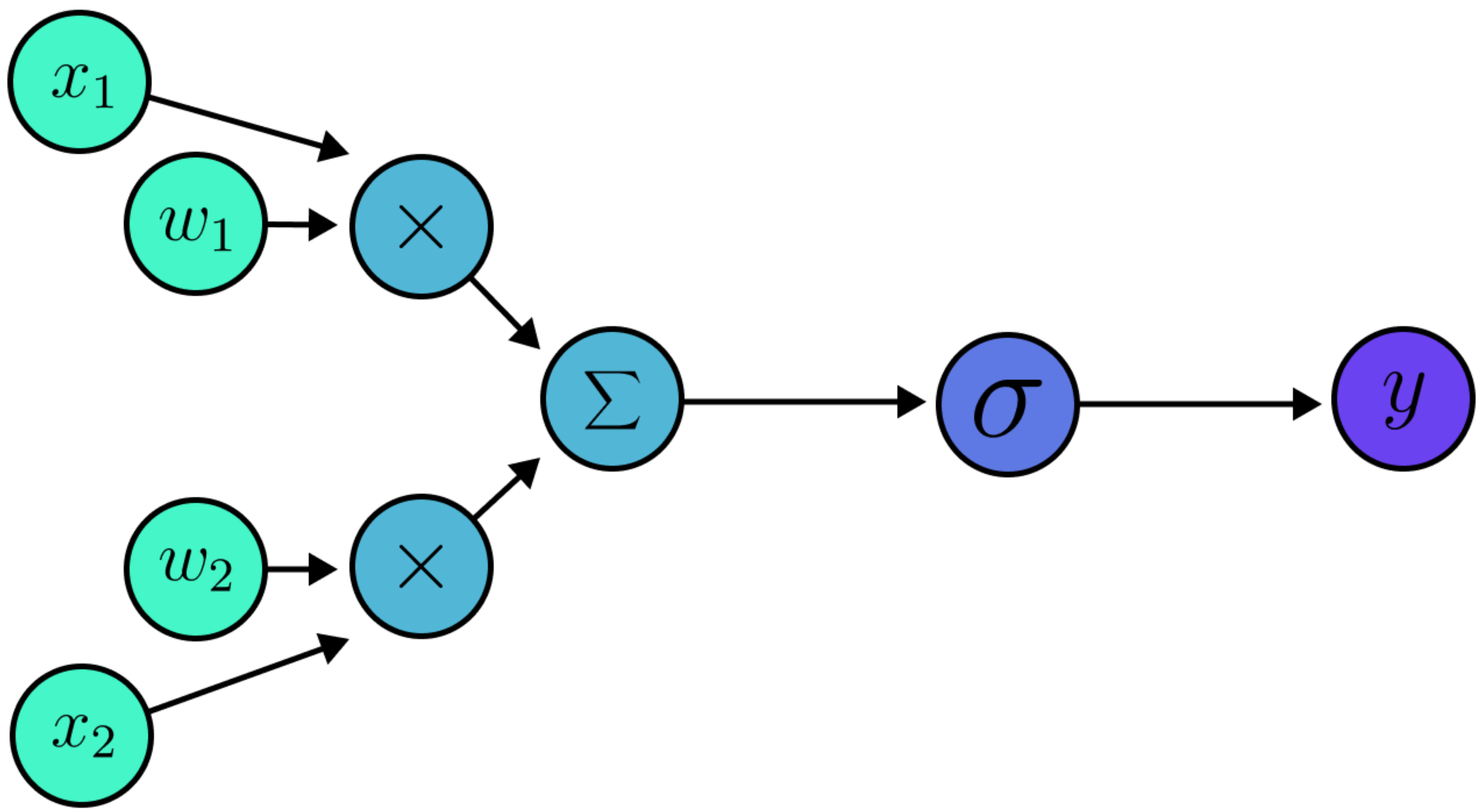
Autograd

Computational graphs, backpropagation and the automatic gradient computation

Jan Vlk
Czech Technical University in Prague
Faculty of Electrical Engineering, Department of cybernetics

CTU
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Computational graph



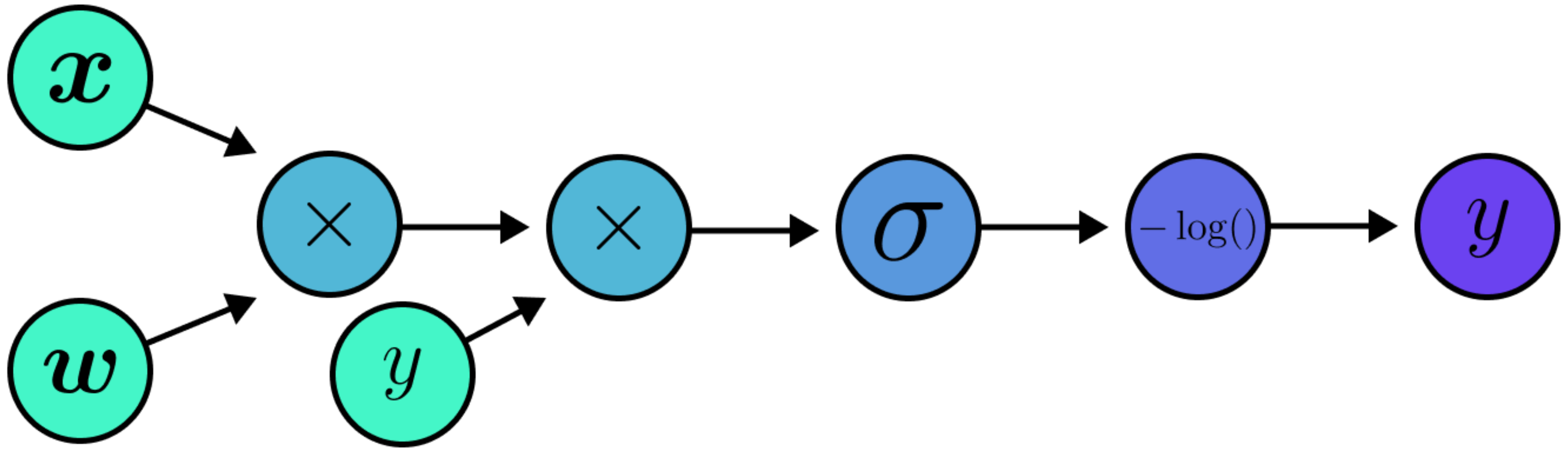
a) $\boldsymbol{x} = \begin{bmatrix} 1 \\ 3 \end{bmatrix}, \boldsymbol{w} = \begin{bmatrix} 0.5 \\ 1.3 \end{bmatrix}$

b) $\boldsymbol{x} = \begin{bmatrix} 1 \\ 3 \end{bmatrix}, \boldsymbol{w} = \begin{bmatrix} 0.5 \\ -1.3 \end{bmatrix}$

c) $\boldsymbol{x} = \begin{bmatrix} 0.4 \\ 0.9 \end{bmatrix}, \boldsymbol{w} = \begin{bmatrix} -1.5 \\ 0.8 \end{bmatrix}$

CTU
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Training weights



$$\boldsymbol{x} = \begin{bmatrix} 0.7 \\ 0.4 \\ 1.2 \end{bmatrix}, \boldsymbol{w} = \begin{bmatrix} 0.3 \\ 0.3 \\ 0.3 \end{bmatrix}, y = 1$$

Individual task

You are given the following equation

$$y = \sin\left(\boldsymbol{w}^\top \boldsymbol{x}\right) - b$$

with

$$\boldsymbol{x} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \boldsymbol{w} = \begin{bmatrix} \frac{\pi}{2} \\ \pi \end{bmatrix}, b = 0, \hat{y} = 2$$

a) Draw a computational graph

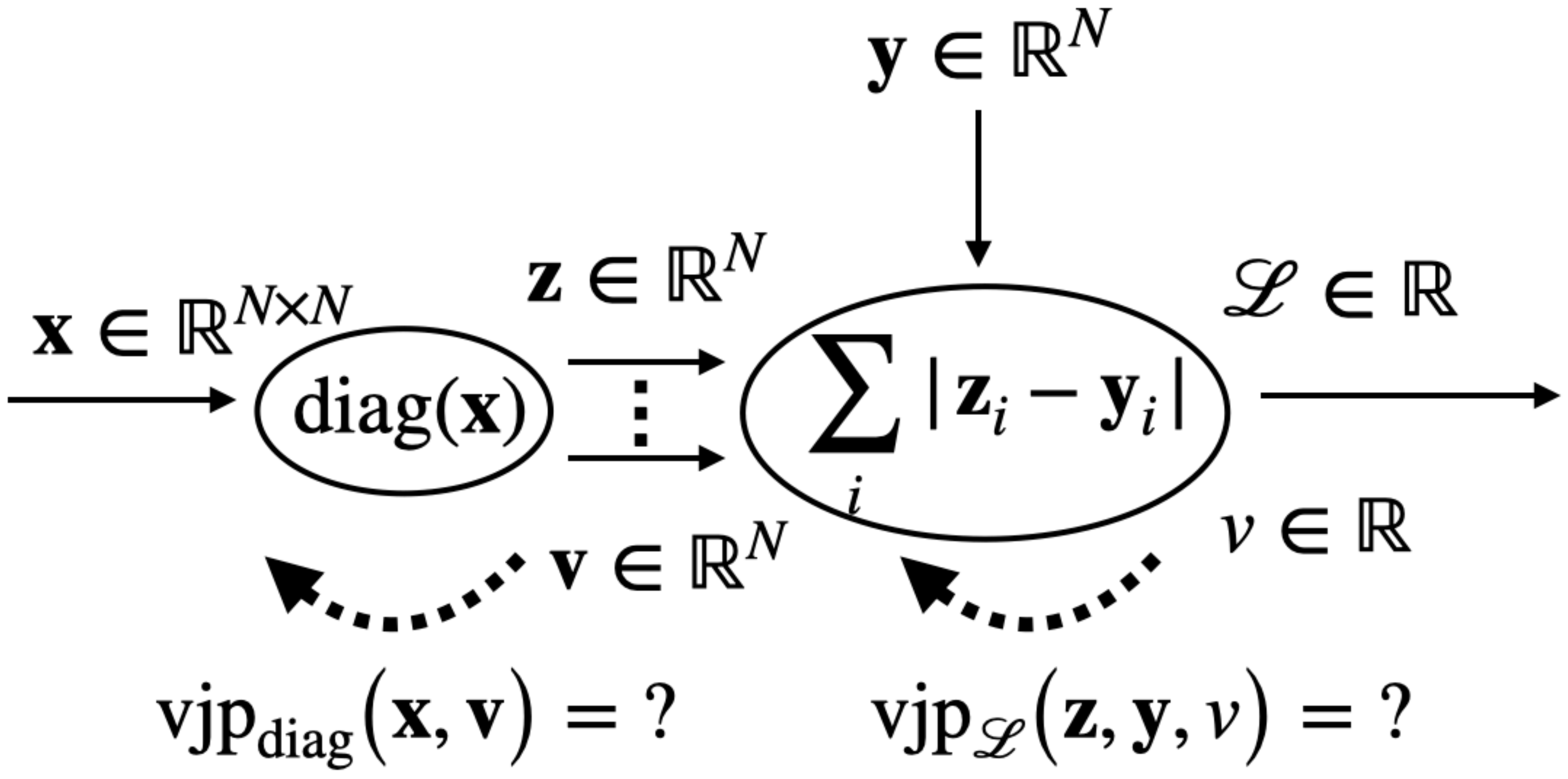
b) Compute feedforward pass

c) Calculate gradient $\frac{\partial y}{\partial \boldsymbol{w}}$

d) Calculate MSE loss, add to graph

e) Update weights with $\alpha = 0.5$

Vector-Jacobian product



a) Implementation of $\mathrm{vjp}_{\mathrm{diag}}(\boldsymbol{v}, \boldsymbol{x})$, where $\boldsymbol{v}$ is the upstream gradient

b) Derive the $\dfrac{\partial \mathscr{L}}{\partial \boldsymbol{x}}$

Backpropagation in code

Under the hood of autograd library

CTU
CZECH TECHNICAL UNIVERSITY IN PRAGUE

CTU
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Activation functions

Sigmoid

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

$$\frac{\partial f(x)}{\partial x} = \frac{e^{-x}}{(e^{-x} + 1)^2}$$

ReLU

$$f(x) = \max(0,\ x)$$

$$\frac{\partial f(x)}{\partial x} = 1,\ x \geq 0 \\ = 0,\ x < 0$$

Tanh

$$f(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

$$\frac{\partial f(x)}{\partial x} = \frac{4e^{2x}}{\left(e^{2x} + 1\right)^2}$$

CTU
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Logistic loss

$$f(x, y) = \log\left(1 + e^{-xy}\right)$$

$$\frac{\partial f(x, y)}{\partial x} = -\frac{y}{1 + e^{xy}}$$

HW2 Autograd

Creating your own library

For next week

Matrix multiplication

Mathematical derivatives

$$\begin{bmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & & \vdots \\ x_{m1} & \cdots & x_{mn} \end{bmatrix} \times \begin{bmatrix} y_{11} & \cdots & y_{1m} \\ \vdots & & \vdots \\ y_{n1} & \cdots & y_{nm} \end{bmatrix} = \begin{bmatrix} x_{11} \cdot y_{11} + \cdots + x_{1n} \cdot y_n 1 & \cdots & x_{11} \cdot y_{1m} + \cdots + x_{1n} \cdot y_{nm} \\ \vdots & & \vdots \\ x_{m1} \cdot y_{11} + \cdots + x_{mn} \cdot y_{n1} & \cdots & x_{m1} \cdot y_{1m} + \cdots + x_{mn} \cdot y_{nm} \end{bmatrix}$$

$$\frac{\partial f}{\partial \boldsymbol{Y}} = \boldsymbol{X}^\top \quad \frac{\partial f}{\partial \boldsymbol{X}} = \boldsymbol{Y}^\top$$

Backward pass in code (using vjp)

$$\mathrm{vjp}_f(\boldsymbol{v}, \boldsymbol{X}) = \boldsymbol{v} \times \frac{\partial f}{\partial \boldsymbol{X}} = \boldsymbol{v} \times \boldsymbol{Y}^\top$$

$$\mathrm{vjp}_f(\boldsymbol{v}, \boldsymbol{Y}) = \frac{\partial f}{\partial \boldsymbol{Y}} \times \boldsymbol{v} = \boldsymbol{X}^\top \times \boldsymbol{v}$$

https://math.stackexchange.com/questions/1846339/why-does-the-gradient-of-matrix-product-ab-w-r-t-a-equal-bt

CTU
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Regularization loss

$$f(\boldsymbol{x}, y) = y \cdot \sum_{i=1}^{n} x_i^2$$

$$\boldsymbol{x} \in \mathbb{R}^n,\ y \in \mathbb{R}$$

$$\frac{\partial f(\boldsymbol{x}, y)}{\partial x_i} = 2 \cdot y \cdot x_i$$

CTU CZECH TECHNICAL UNIVERSITY IN PRAGUE

Cross-entropy loss

$$f(\boldsymbol{x}, \boldsymbol{y}) = -\frac{1}{n}\sum_{i=1}^{n} [\![y_i = c_i]\!] \log(x_i)$$

$$\frac{\partial f(\boldsymbol{x}, \boldsymbol{y})}{\partial \boldsymbol{x}} = -\frac{1}{n}\sum_{i=1}^{n} [\![y_i = c_i]\!] \frac{1}{x_i}$$

$\boldsymbol{x} \in \mathbb{R}^n$ is a vector with class **probabilities** for input

$\boldsymbol{y} \in \mathbb{R}^n$ is a vector of correct class predictions for input

$\boldsymbol{c} \in \mathbb{R}^n$ is a vector of all possible classes

$[\![\cdot]\!]$ is an Iverson Bracket

CTU
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Cross-entropy loss with softmax

With proof, do not learn the proof just the result is enough

$$f(\boldsymbol{x},\boldsymbol{y}) = -\sum_i y_i \log\left(a_i\right), \quad a_i = h\left(x_i\right) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

$$\frac{\partial f}{\partial a_i} = \frac{\partial\left(-\sum_j y_j \cdot \log\left(a_j\right)\right)}{\partial a_i} = \frac{\partial\left(-y_i \cdot \log\left(a_i\right)\right)}{\partial a_i} = -\frac{y_i}{a_i}$$

$$\frac{\partial f}{\partial x_i} = \sum_j \frac{\partial f}{\partial a_j} \cdot \frac{\partial a_j}{\partial x_i}$$

$$\frac{\partial a_i}{\partial x_i} = \frac{\partial\left(\frac{e^{x_i}}{\sum_j e^{x_j}}\right)}{\partial x_i} = \frac{e^{x_i}}{\sum_j e^{x_j}}\left(\frac{e^{x_j}}{e^{x_j}} - \frac{e^{x_i}}{\sum_j e^{x_j}}\right) = a_i\left(1 - a_i\right)$$

$$\frac{\partial f}{\partial x_i} = \sum_{j \neq i} \frac{\partial f}{\partial a_j} \cdot \frac{\partial a_j}{\partial x_i} + \frac{\partial f}{\partial a_i} \cdot \frac{\partial a_i}{\partial x_i}$$

$$\frac{\partial f}{\partial a_j} = \frac{\partial\left(-\sum_k y_k \cdot \log(a_k)\right)}{\partial a_j} = -\frac{y_j}{a_j}$$

$$\frac{\partial f}{\partial x_i} = \sum_{j \neq i} y_j \cdot a_i - y_i\left(1 - a_i\right) = \sum_{j \neq i} y_j \cdot a_i + y_i \cdot a_i - y_i$$

$$\frac{\partial a_j}{\partial x_i} = \frac{\partial\left(\frac{e^{x_j}}{\sum_k e^{x_k}}\right)}{\partial x_i} = \frac{e^{x_j}}{\sum_k e^{x_k}}\left(\frac{\frac{\partial e^{x_j}}{\partial x_i}}{e^{x_j}} - \frac{\frac{\partial \sum_k e^{x_k}}{\partial x_i}}{\sum_k e^{x_k}}\right) = \frac{e^{x_j}}{\sum_k e^{x_k}}\left(0 - \frac{e^{x_i}}{\sum_k e^{x_k}}\right) = -a_j \cdot a_i$$

$$\frac{\partial f}{\partial x_i} = \sum_j y_j \cdot a_i - y_i = a_i \sum_j y_j - y_i$$

$$\boxed{\frac{\partial f}{\partial x_i} = a_i - y_i}$$

https://towardsdatascience.com/deriving-backpropagation-with-cross-entropy-loss-d24811edeaf9