Разбор домашнего задания №3 Тренировки по ML

Young&&Yandex ШАД

girafe ai

Arkadii Lysiakov

Outline

01 NN pipeline

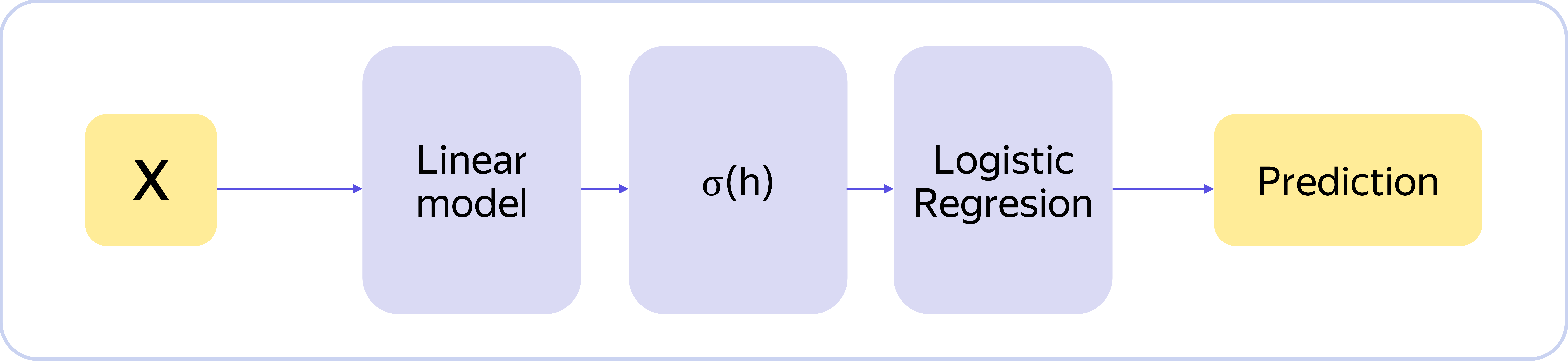
02 Activation functions

03 Backpropagation

04 Gradient optimization

05 Regularization (bonus part)

NN pipeline: example



E. g. two logistic regressions one after another

Actually, it’s a neural network

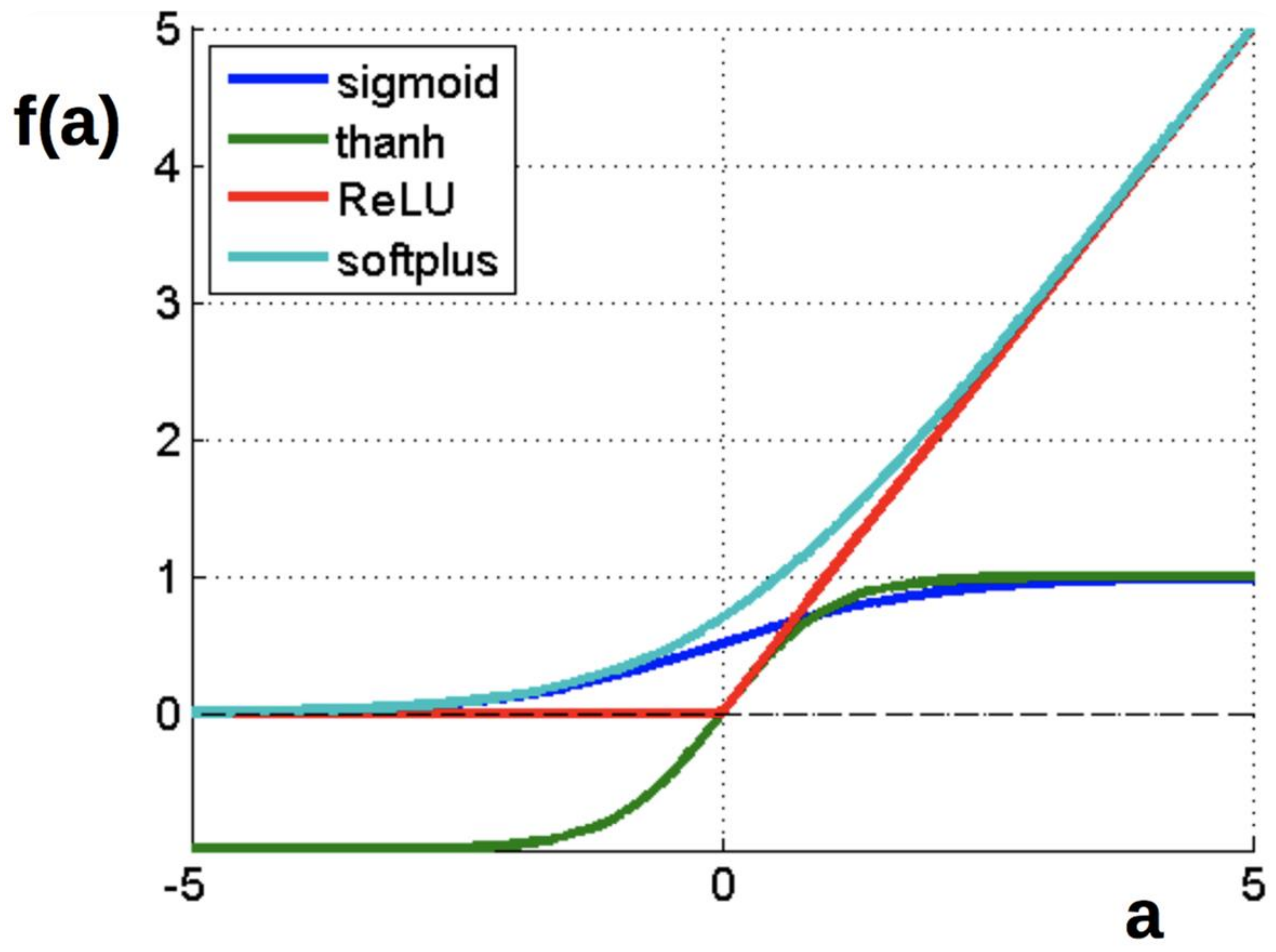
Activation functions: nonlinearities

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

$$f(a) = \tanh(a)$$

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

$$f(a) = \log(1 + e^{a})$$

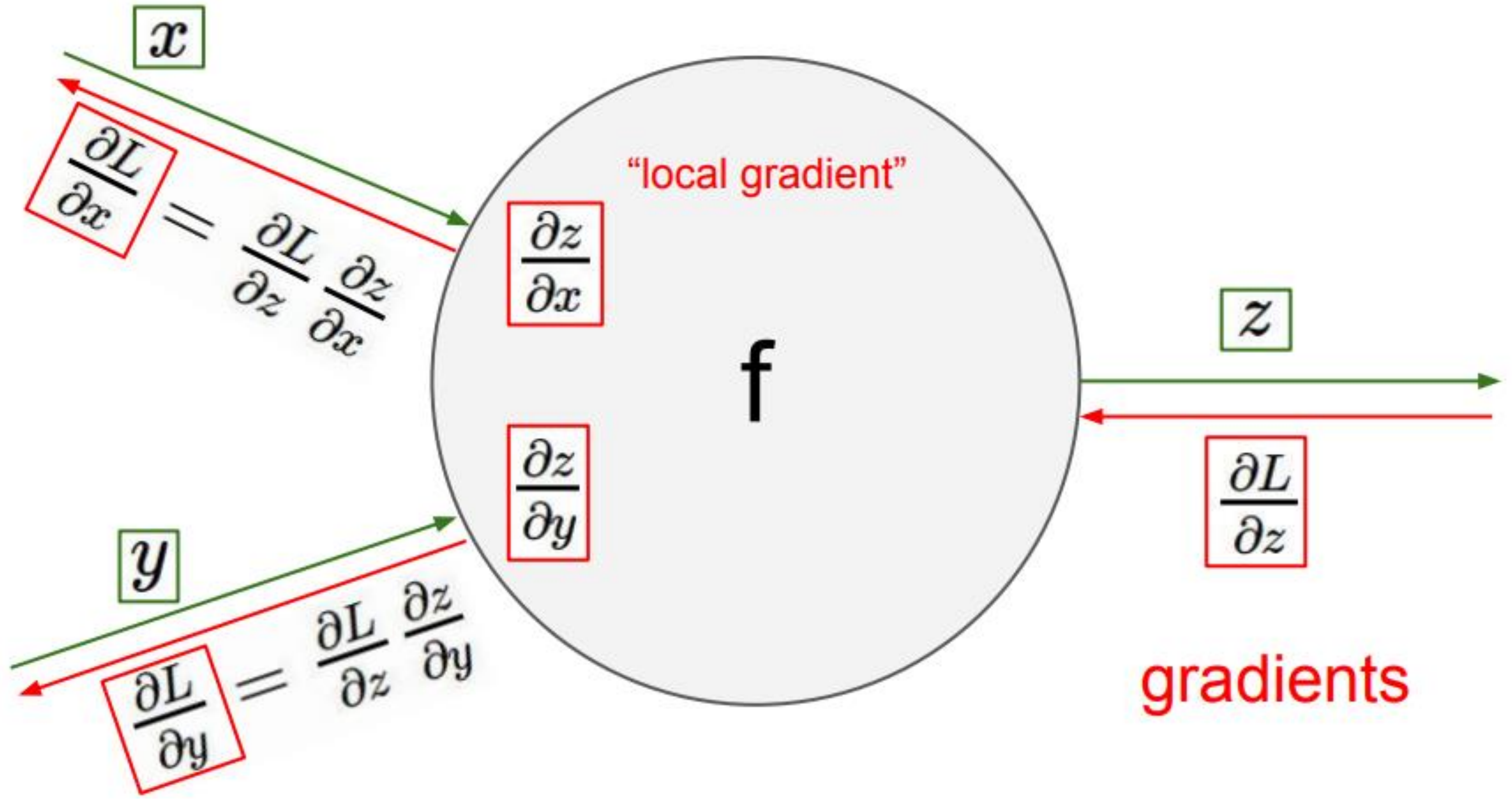


Backpropagation and chain rule

Chain rule is just simple math:

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$$

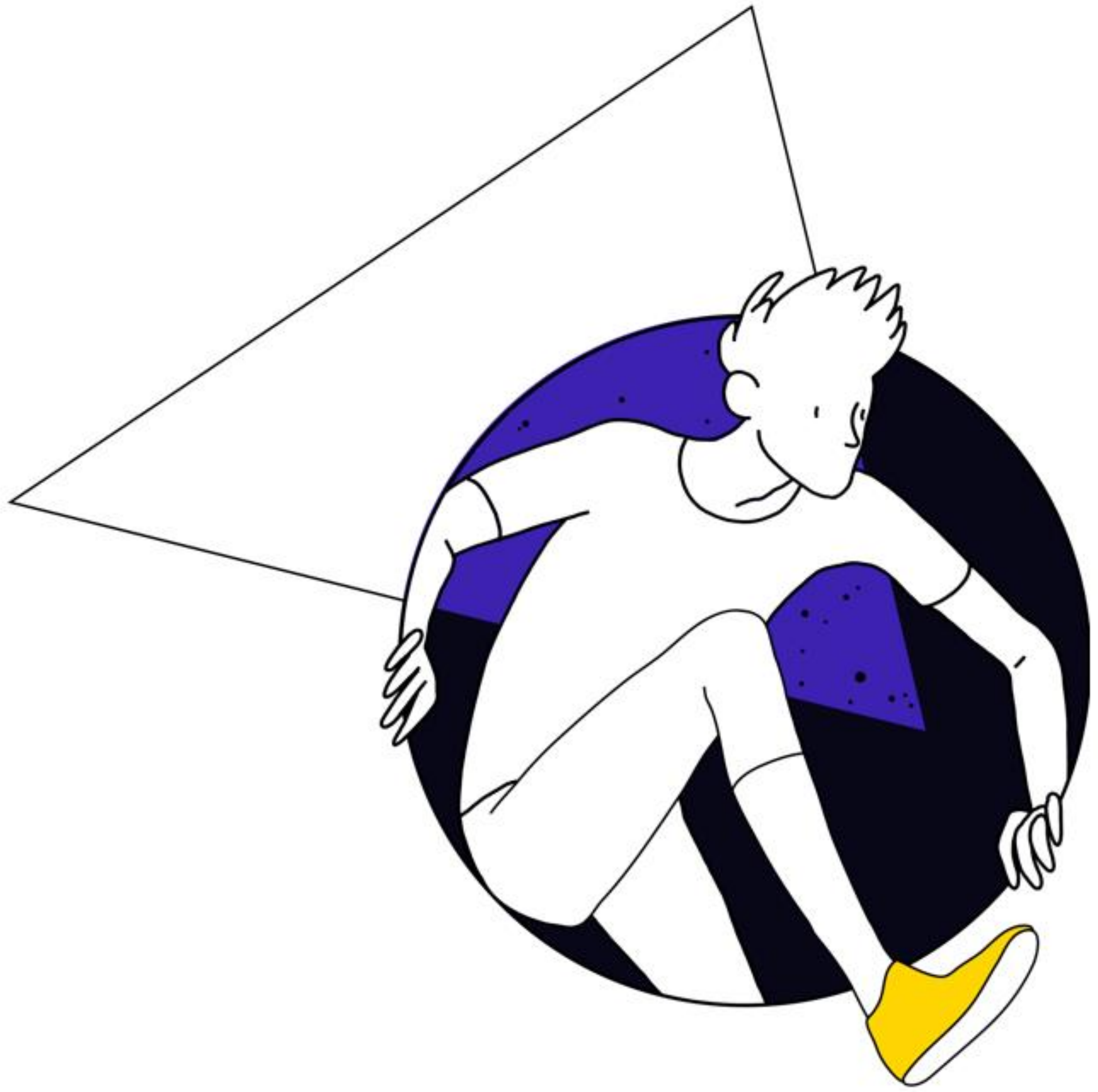
Backprop is just way to use it in NN training



source

Backpropagation: matrix form

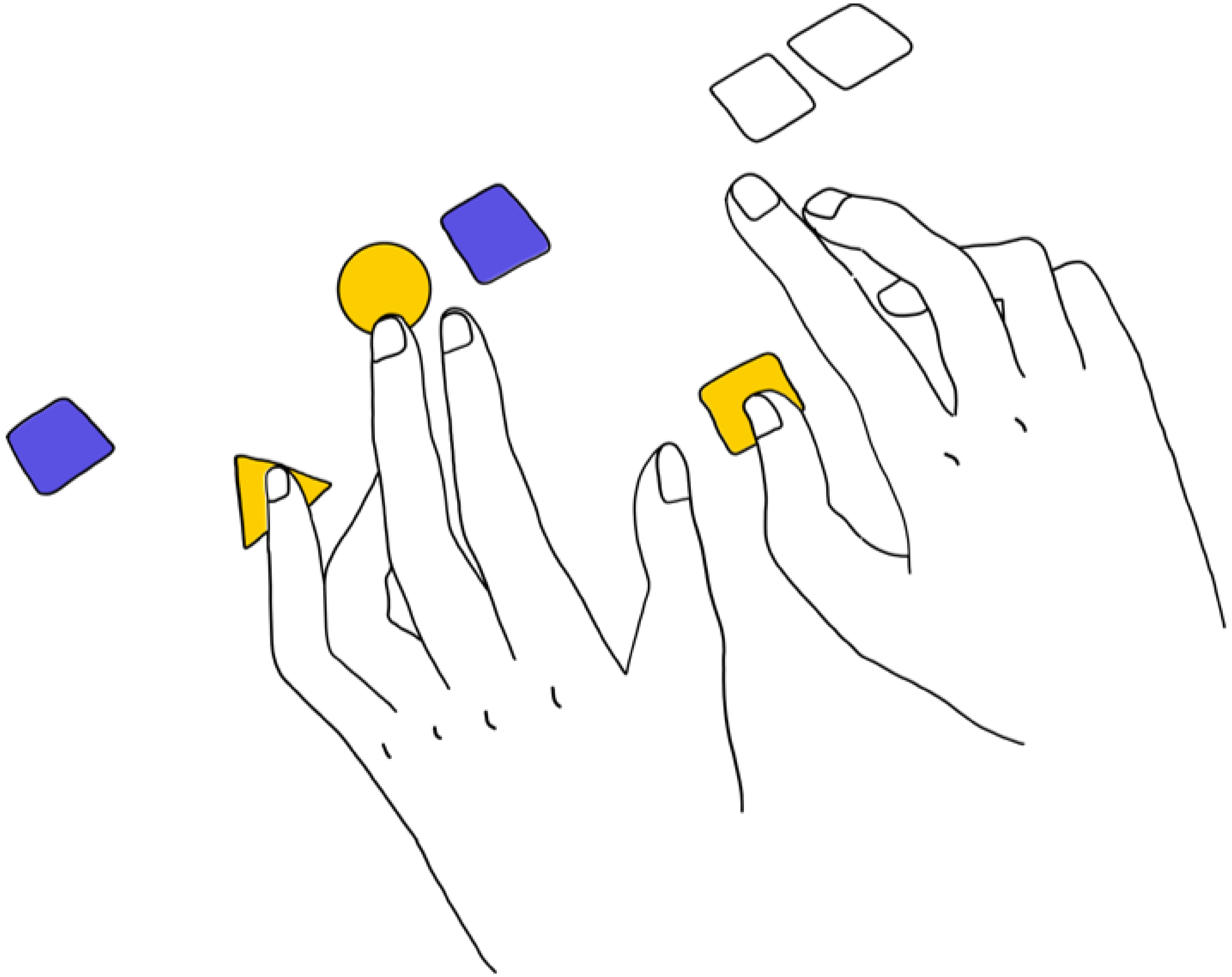
	scalar x	vector $\mathbf{x}$
scalar f	$\frac{\partial f}{\partial x}$	$\frac{\partial f}{\partial \mathbf{x}}$
vector $\mathbf{f}$	$\frac{\partial \mathbf{f}}{\partial x}$	$\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$



source

Backpropagation: matrix form

$$
\begin{aligned}
\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial}{\partial \mathbf{x}} f_1(\mathbf{x}) \\ \frac{\partial}{\partial \mathbf{x}} f_2(\mathbf{x}) \\ \ldots \\ \frac{\partial}{\partial \mathbf{x}} f_m(\mathbf{x}) \end{bmatrix} &= \begin{bmatrix} \frac{\partial}{\partial x_1} f_1(\mathbf{x}) & \frac{\partial}{\partial x_2} f_1(\mathbf{x}) & \ldots & \frac{\partial}{\partial x_n} f_1(\mathbf{x}) \\ \frac{\partial}{\partial x_1} f_2(\mathbf{x}) & \frac{\partial}{\partial x_2} f_2(\mathbf{x}) & \ldots & \frac{\partial}{\partial x_n} f_2(\mathbf{x}) \\ & \ldots & & \\ \frac{\partial}{\partial x_1} f_m(\mathbf{x}) & \frac{\partial}{\partial x_2} f_m(\mathbf{x}) & \ldots & \frac{\partial}{\partial x_n} f_m(\mathbf{x}) \end{bmatrix} \\
&= \begin{bmatrix} \frac{\partial}{\partial x_1} x_1 & \frac{\partial}{\partial x_2} x_1 & \ldots & \frac{\partial}{\partial x_n} x_1 \\ \frac{\partial}{\partial x_1} x_2 & \frac{\partial}{\partial x_2} x_2 & \ldots & \frac{\partial}{\partial x_n} x_2 \\ & \ldots & & \\ \frac{\partial}{\partial x_1} x_n & \frac{\partial}{\partial x_2} x_n & \ldots & \frac{\partial}{\partial x_n} x_n \end{bmatrix} \\
& \quad (\text{and since } \frac{\partial}{\partial x_j} x_i = 0 \text{ for } j \neq i) \\
&= \begin{bmatrix} \frac{\partial}{\partial x_1} x_1 & 0 & \ldots & 0 \\ 0 & \frac{\partial}{\partial x_2} x_2 & \ldots & 0 \\ & & \ddots & \\ 0 & 0 & \ldots & \frac{\partial}{\partial x_n} x_n \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 \\ & & \ddots & \\ 0 & 0 & \ldots & 1 \end{bmatrix} \\
&= I \quad (I \text{ is the identity matrix with ones down the diagonal})
\end{aligned}
$$

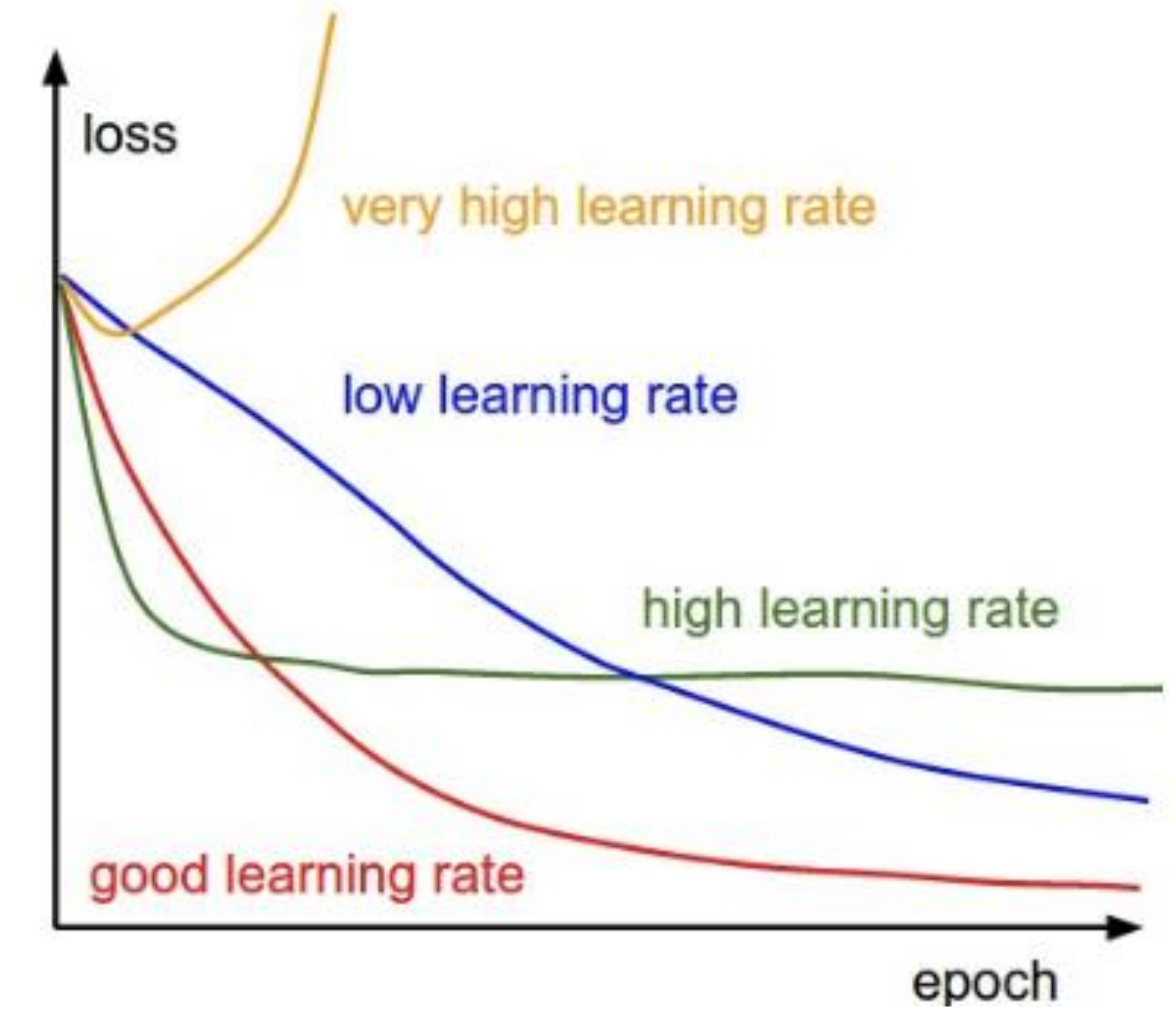


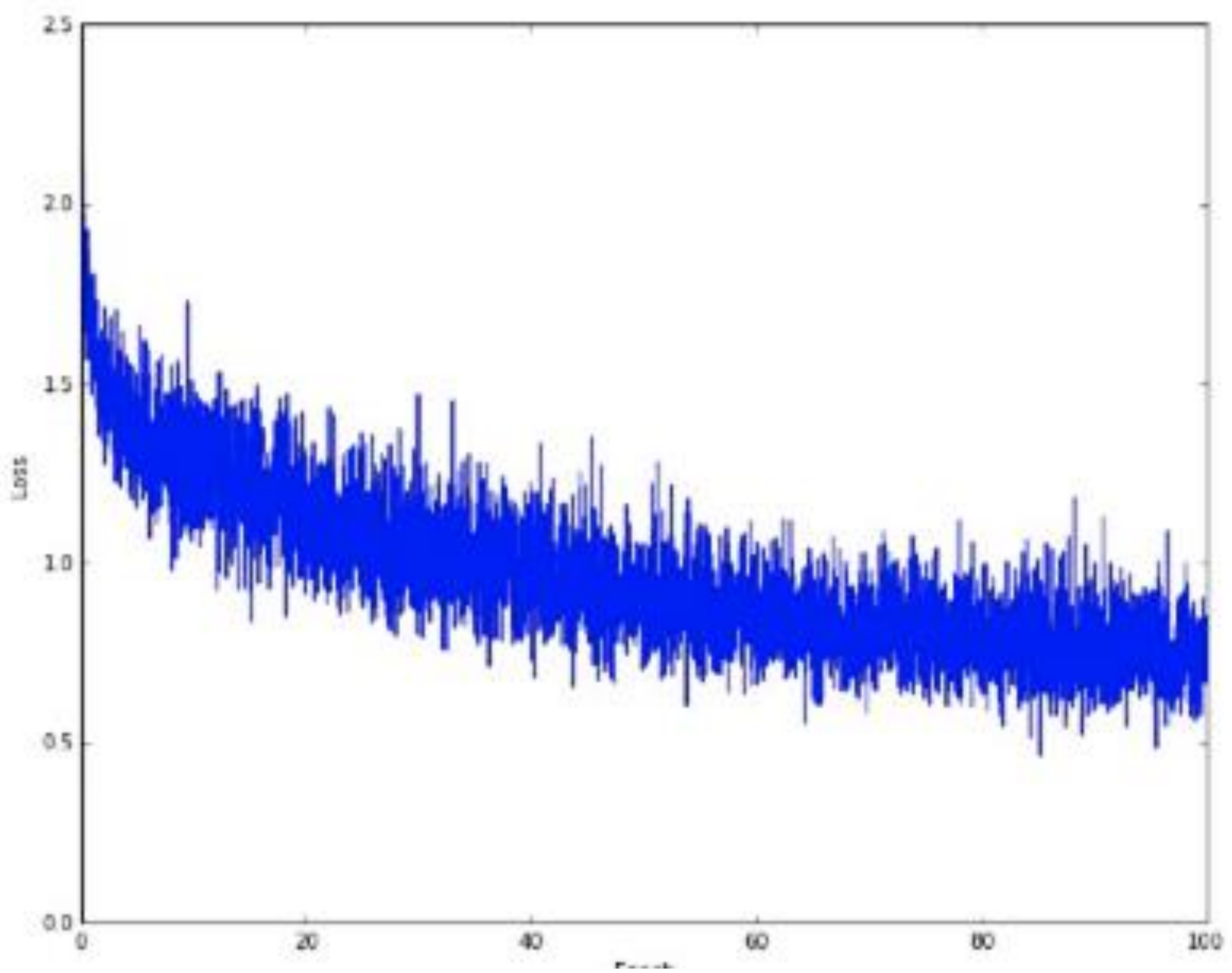
source

Gradient optimization

Stochastic gradient descent is used to optimize NN parameters

$$x_{t+1} = x_t - \text{learning rate} \cdot dx$$





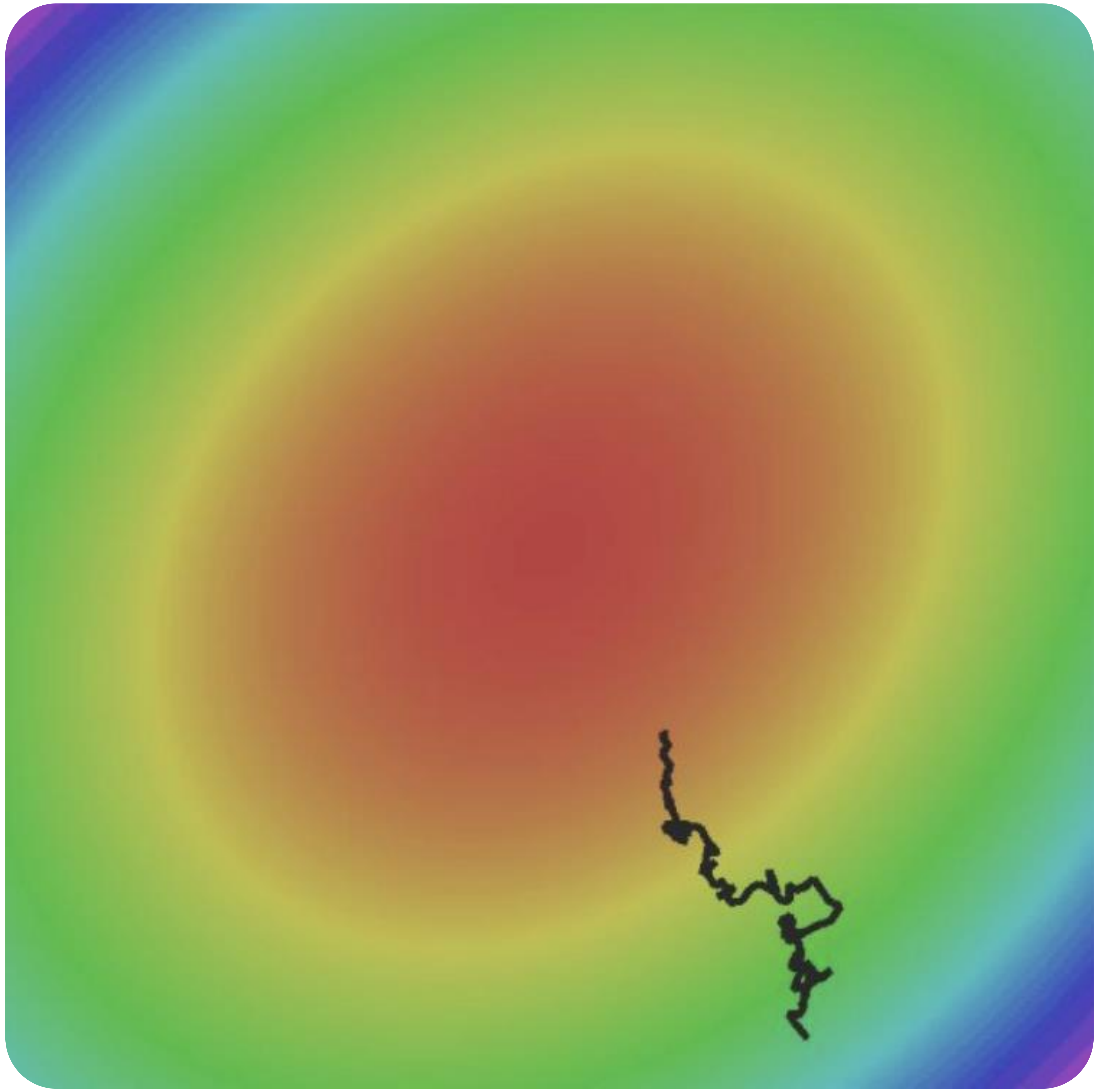
source

Optimization: SGD

$$L(W) = \frac{1}{N}\sum_{i=1}^{N} L_i(x_i, y_i, W)$$

$$\nabla_W L(W) = \frac{1}{N}\sum_{i=1}^{N} \nabla_W L_i(x_i, y_i, W)$$

Averaging over minibatches =>noisy gradient



source

Second idea: different dimensions are different

01 Adagrad:
SGD with cache

$$\text{cache}_{t+1} = \text{cache}_t + (\nabla f(x_t))^2$$

$$x_{t+1} = x_t - \alpha \frac{\nabla f(x_t)}{\text{cache}_{t+1} + \varepsilon}$$

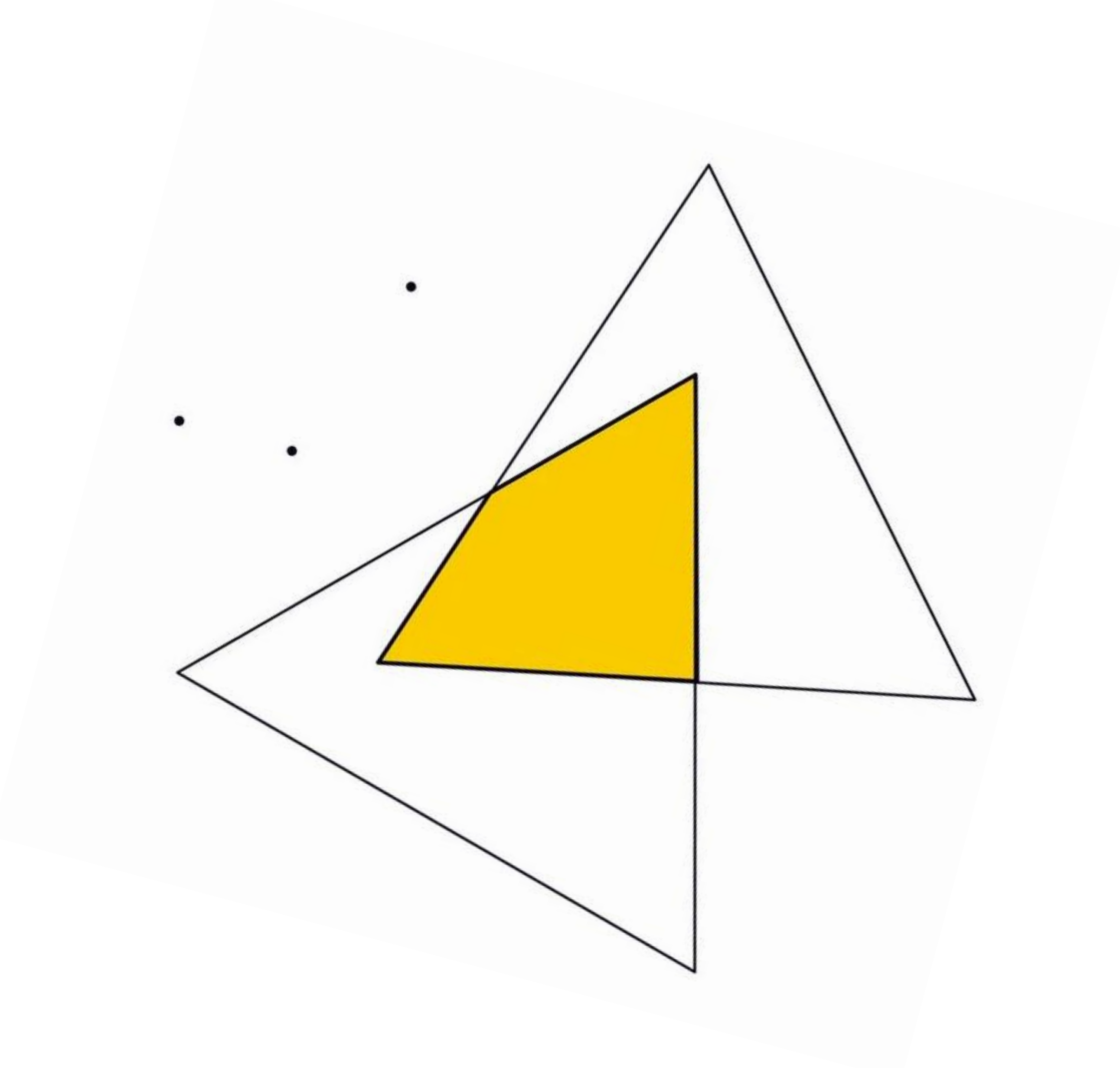
Second idea: different dimensions are different

01 Adagrad:
SGD with cache

$$\text{cache}_{t+1} = \text{cache}_t + (\nabla f(x_t))^2$$

$$x_{t+1} = x_t - \alpha \frac{\nabla f(x_t)}{\text{cache}_{t+1} + \varepsilon}$$

Problem: gradient fades with time



Second idea: different dimensions are different

01 Adagrad:
SGD with cache

$$\text{cache}_{t+1} = \text{cache}_t + (\nabla f(x_t))^2$$

$$x_{t+1} = x_t - \alpha \frac{\nabla f(x_t)}{\text{cache}_{t+1} + \varepsilon}$$

02 RMSProp: SGD with cache
with exp. Smoothing

$$\text{cache}_{t+1} = \beta\text{cache}_t + (1-\beta)(\nabla f(x_t))^2$$

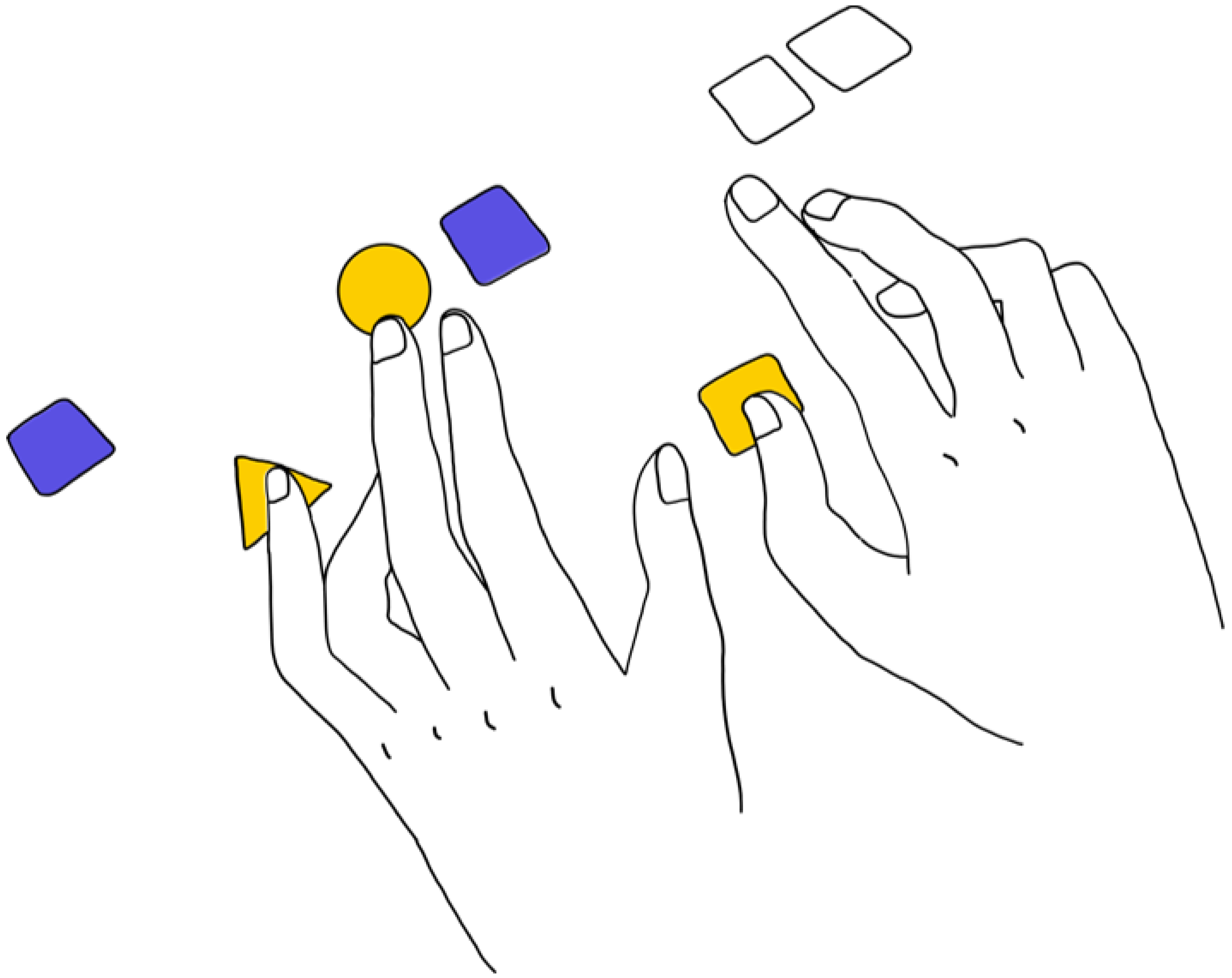
$$x_{t+1} = x_t - \alpha \frac{\nabla f(x_t)}{\text{cache}_{t+1} + \varepsilon}$$

Slide 29 Lecture 6 of Geoff Hinton's Coursera class

Adam

Let's combine the momentum idea and RMSProp normalization:

$$v_{t+1} = \gamma v_t + (1-\gamma)\nabla f(x_t)$$
$$\text{cache}_{t+1} = \beta \text{cache}_t + (1-\beta)(\nabla f(x_t))^2$$
$$x_{t+1} = x_t - \alpha \frac{v_{t+1}}{\text{cache}_{t+1} + \varepsilon}$$



Adam full form involves bias correction term. See link for more info.

Regularization

$$L = \frac{1}{N}\sum_{i=1}^{N}\sum\nolimits_{i \neq y_i} \max(0, f\,(x_i; W)_j \; - f\,(x_i; W)_{y_i} + 1\,) + \lambda R(W)$$

Adding some extra term to the loss function

Common cases:

01 L2 regularization:

$R(W) = \|W\|_2^2$

02 L1 regularization:

$R(W) = \|W\|_1$

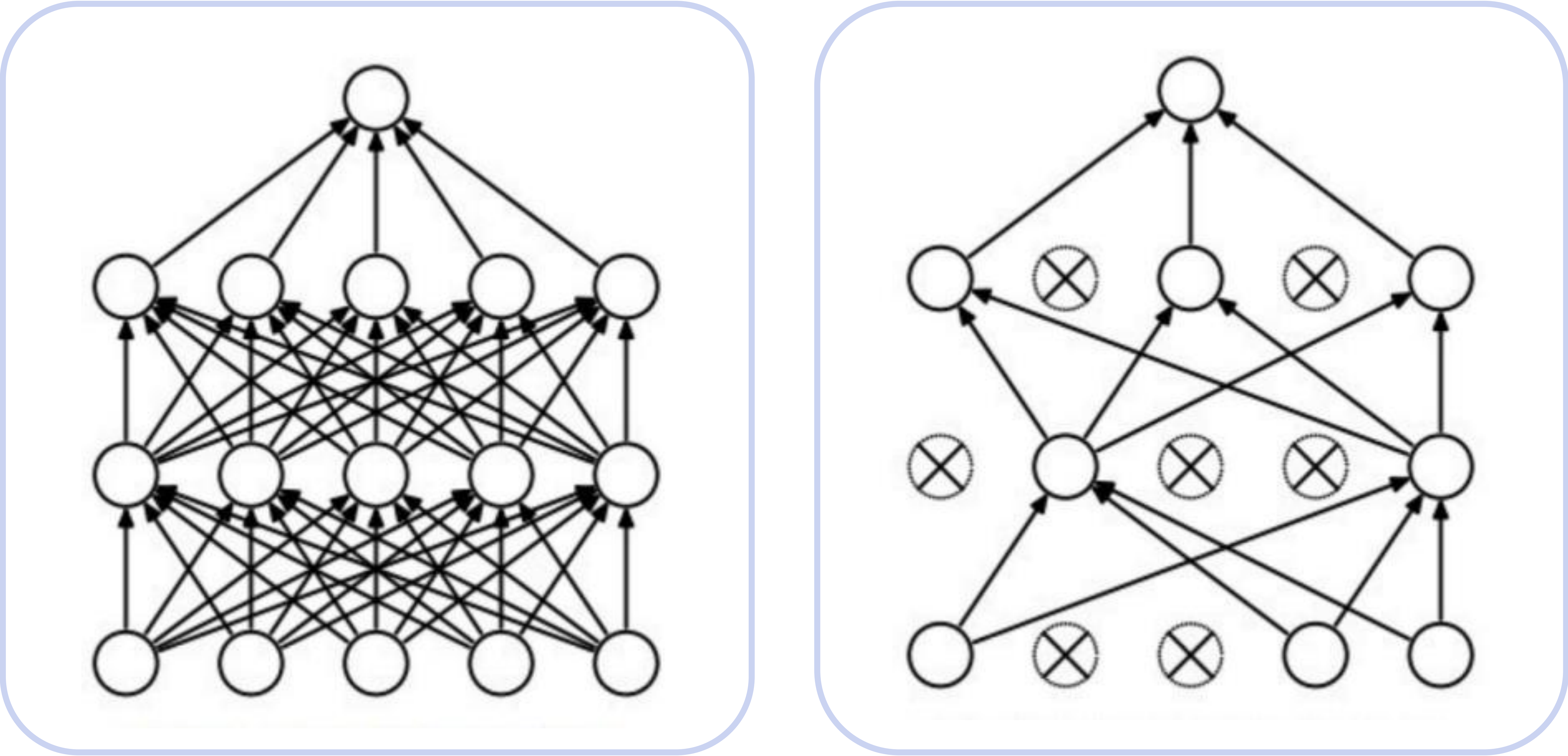
03 Elastic Net (L1 + L2):

$R(W) = \beta\|W\|_2^2 + \|W\|_1$

Regularization: Dropout

Some neurons are “dropped” during training

Prevents overfitting



(a) Standard Neural Net

(b) After applying dropout

Regularization: Batch normalization

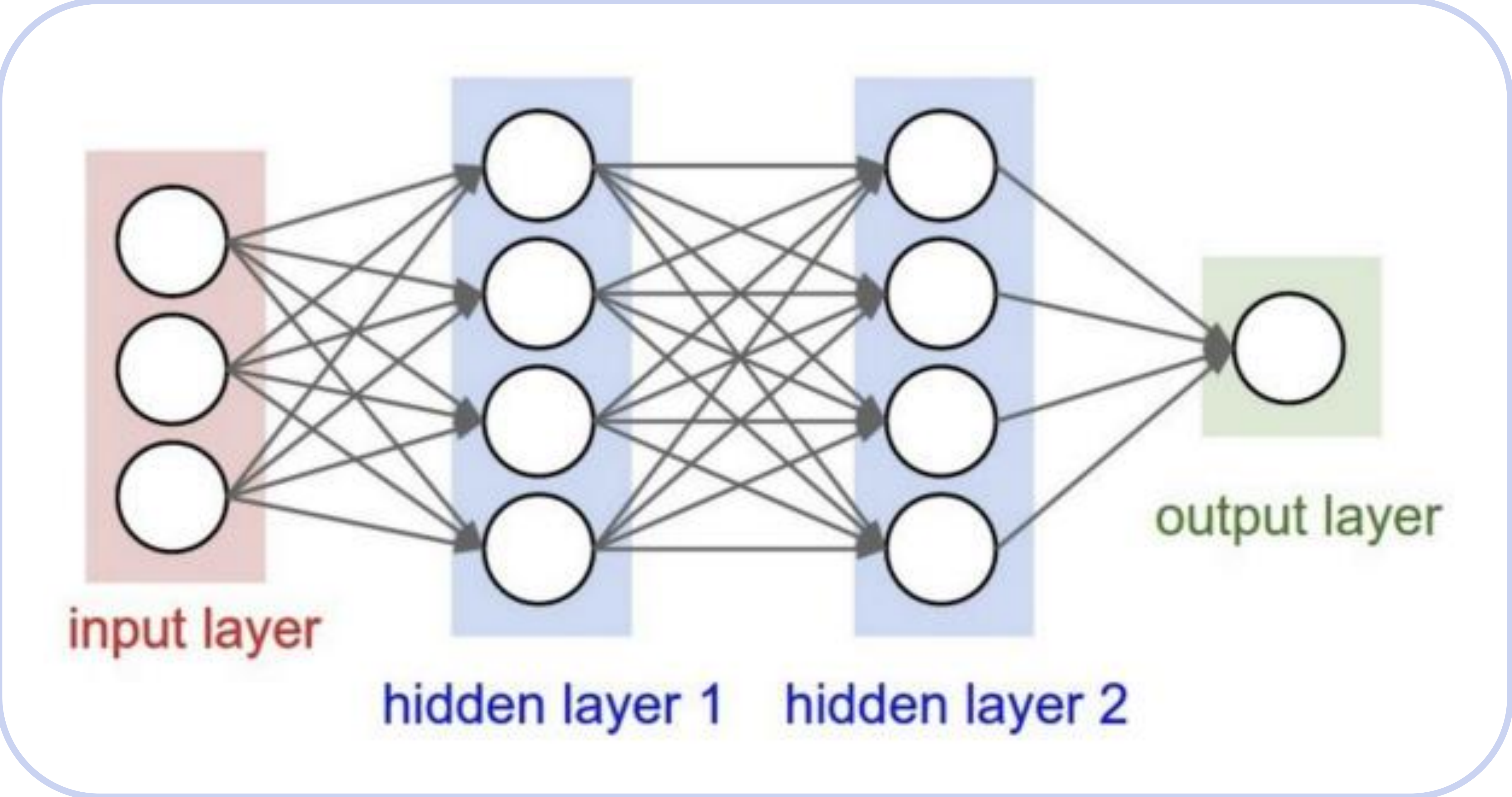
Problem:

01 Consider a neuron in any layer beyond first

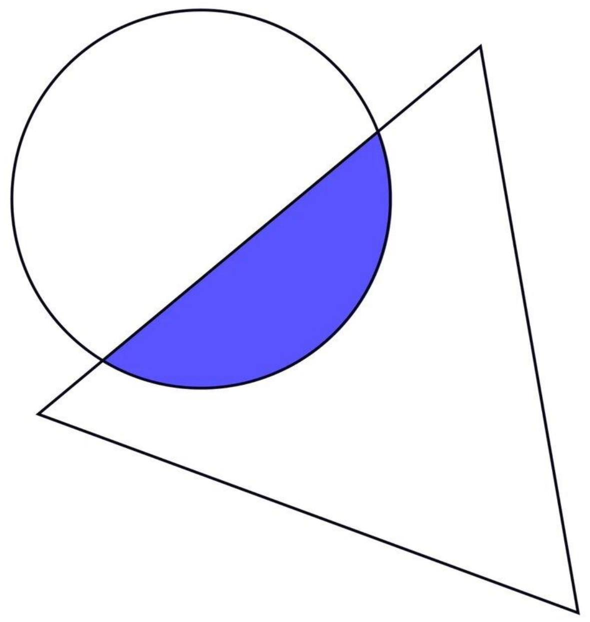
02 At each iteration we tune it's weights towards better loss function

03 Now the neuron needs to be re-tuned for it's new inputs

04 But we also tune it's inputs. Some of them become larger, some — smaller

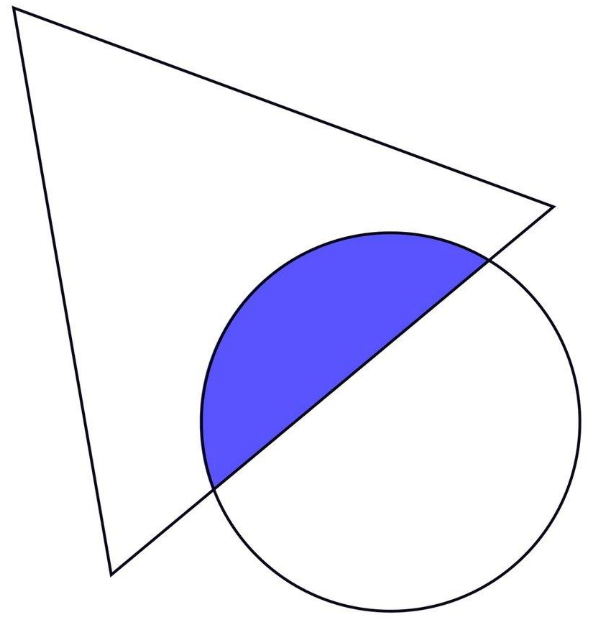


Regularization: Batch normalization



Normalize activation of a hidden layer (zero mean unit variance):

$$h_i = \frac{h_i - \mu_i}{\sqrt{\sigma_i^2}}$$



Update μ_i, σ_i^2 with moving average while training:

$$\mu_{i+1} = \alpha\, mean_{batch} + (1 - \alpha)\, \mu_i$$

$$\sigma_{i+1}^2 = \alpha\, variance_{batch} + (1 - \alpha)\sigma_i^2$$

Thanks for attention!

Questions?

girafe ai