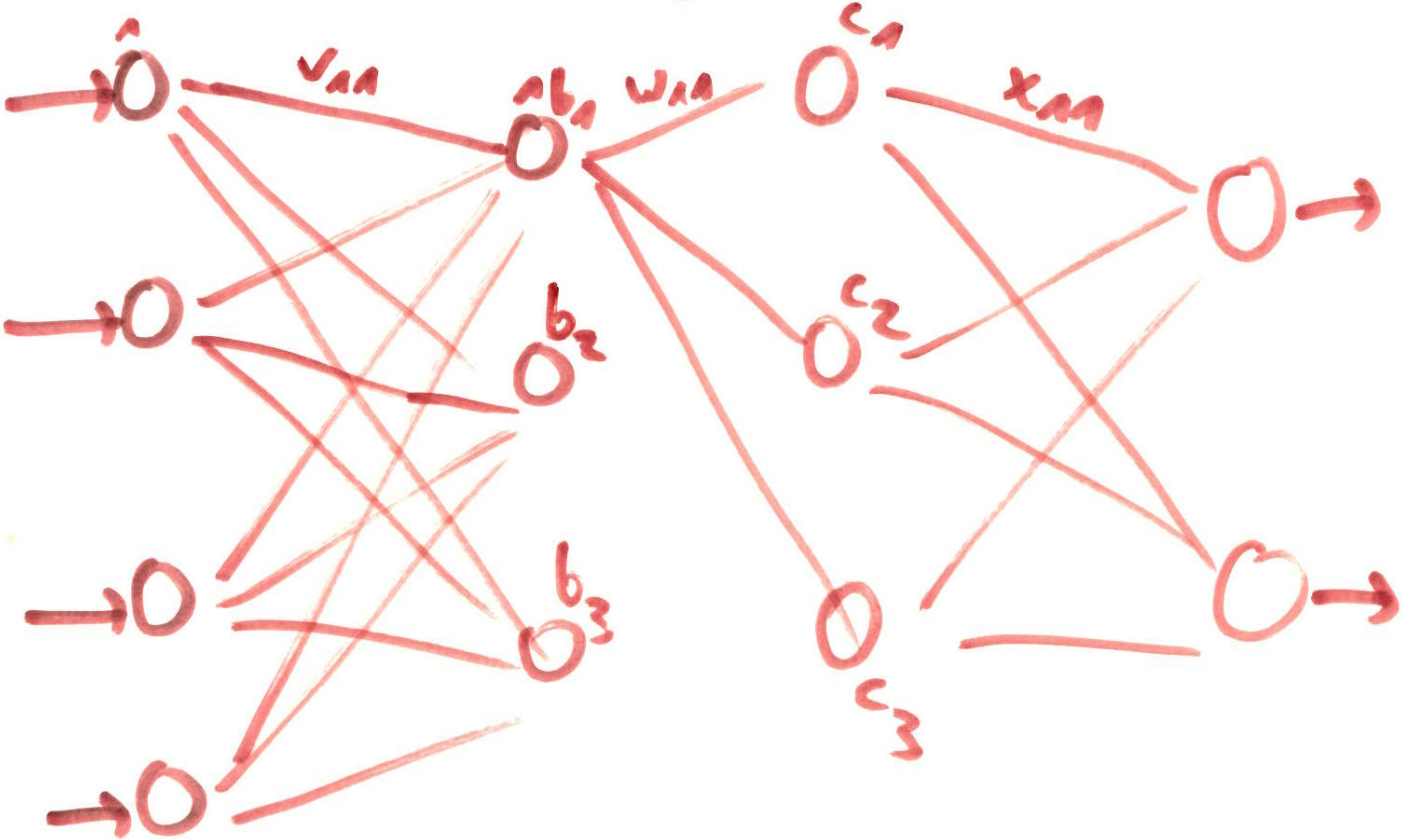
(1) Eingabeschicht verborg. Schichten Ausgabeschicht



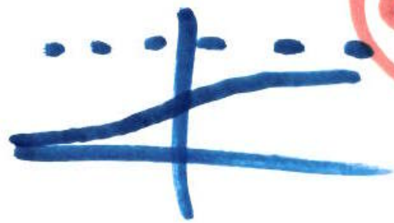
Eingabe $x = \begin{pmatrix} x_1 \\ \vdots \\ x_? \end{pmatrix}$

$\rightsquigarrow$ Aktivierung der ersten Schicht:

$$y = \sigma(\hat{y})$$

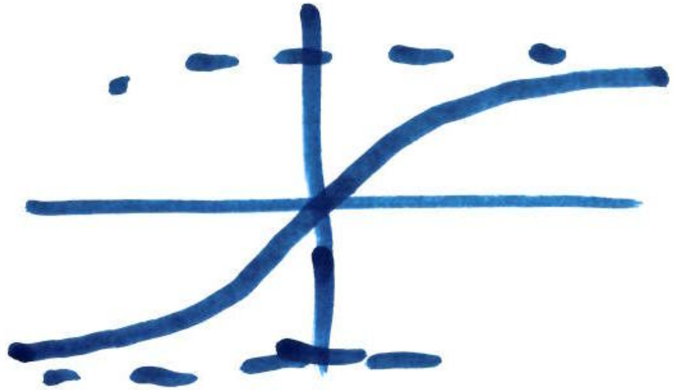
$$\hat{y} = \underbrace{V}_{= \begin{pmatrix} \cdot & \cdots & \cdot \\ \vdots & & \vdots \\ \cdot & \cdots & \cdot \end{pmatrix}} x + \underbrace{b}_{= \begin{pmatrix} b_1 \\ \vdots \\ b_? \end{pmatrix}}$$

① $\sigma(x) = \frac{e^x}{1+e^x}$?



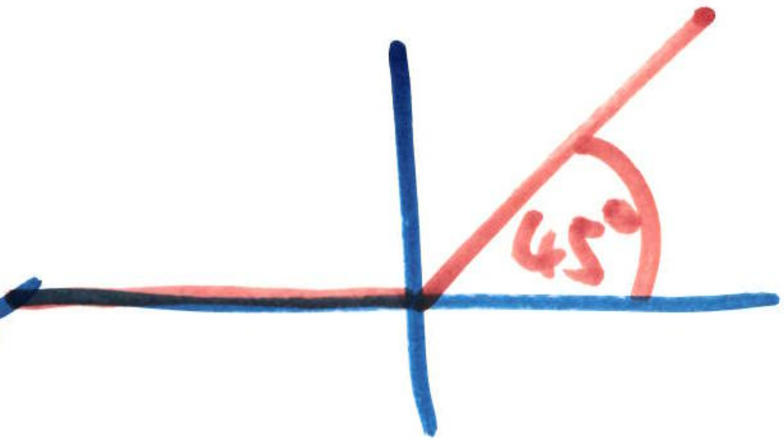
2a

② tanh(x)

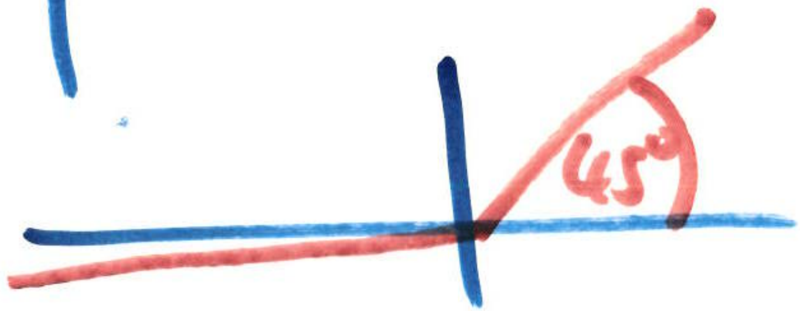


③ RELU

Rectified Linear



④



$$\hat{y}_1 = V_{11}x_1 + V_{12}x_2 + \dots + V_{1?}x_? + b_1$$

$$y_1 = \sigma(\hat{y}_1)$$

Das WUNDER des Lernens

Re
3

Ziel: Finde Gewichte und Biases, s.d. die Abbildung

$$x \longmapsto z \leftarrow \text{Ausgabevektor}$$

möglichst stark einer idealen Abb.

$$x \longmapsto \text{das richtige } z$$

ähnelt.

Realist. Ziel:

Nimm Trainingsdaten

$$x^{(1)}, \dots, x^{(N)} \qquad N = 80000$$

mit bekannten gewünschten Ausgabeaktivierungen

$$\bar{z}^{(1)}, \dots, \bar{z}^{(N)}.$$

Definiere Kostenfunktion:

$$K(\ldots) := \sum_{i=1}^{N} \left\| \underset{\sim}{z}^{(i)} - \underset{\sim}{\bar{z}}^{(i)} \right\|^2$$

K hängt ab von allen Gewichten und Biases

was das NN als Ausgabe produziert

Sollergebnis für Trainingsdaten i

Länge des Vektors $\sqrt{(\,)^2+\ldots+(\,)^2}$

Lernen durch Gradientenabstieg:

1. Beginne mit willkürlichen Gewichten & Biases.

$$A = \begin{pmatrix} \vdots \\ \vdots \\ \vdots \end{pmatrix}$$

2. Berechne ∇K.

3. Neues $A :=$ altes $A - \eta \nabla K$

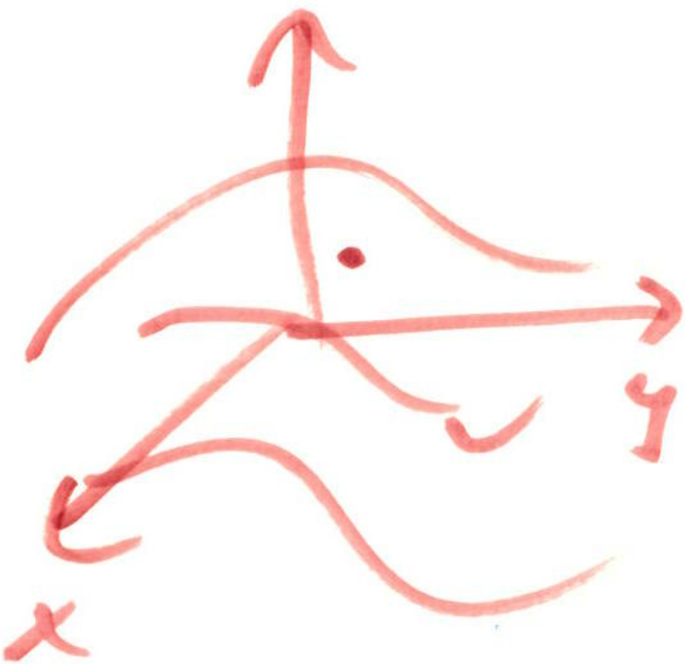
   Lernrate / Schrittweite, z.B. 0.01

4. Mache weiter bis 2.

$$f: \mathbb{R}^n \longrightarrow \mathbb{R}$$

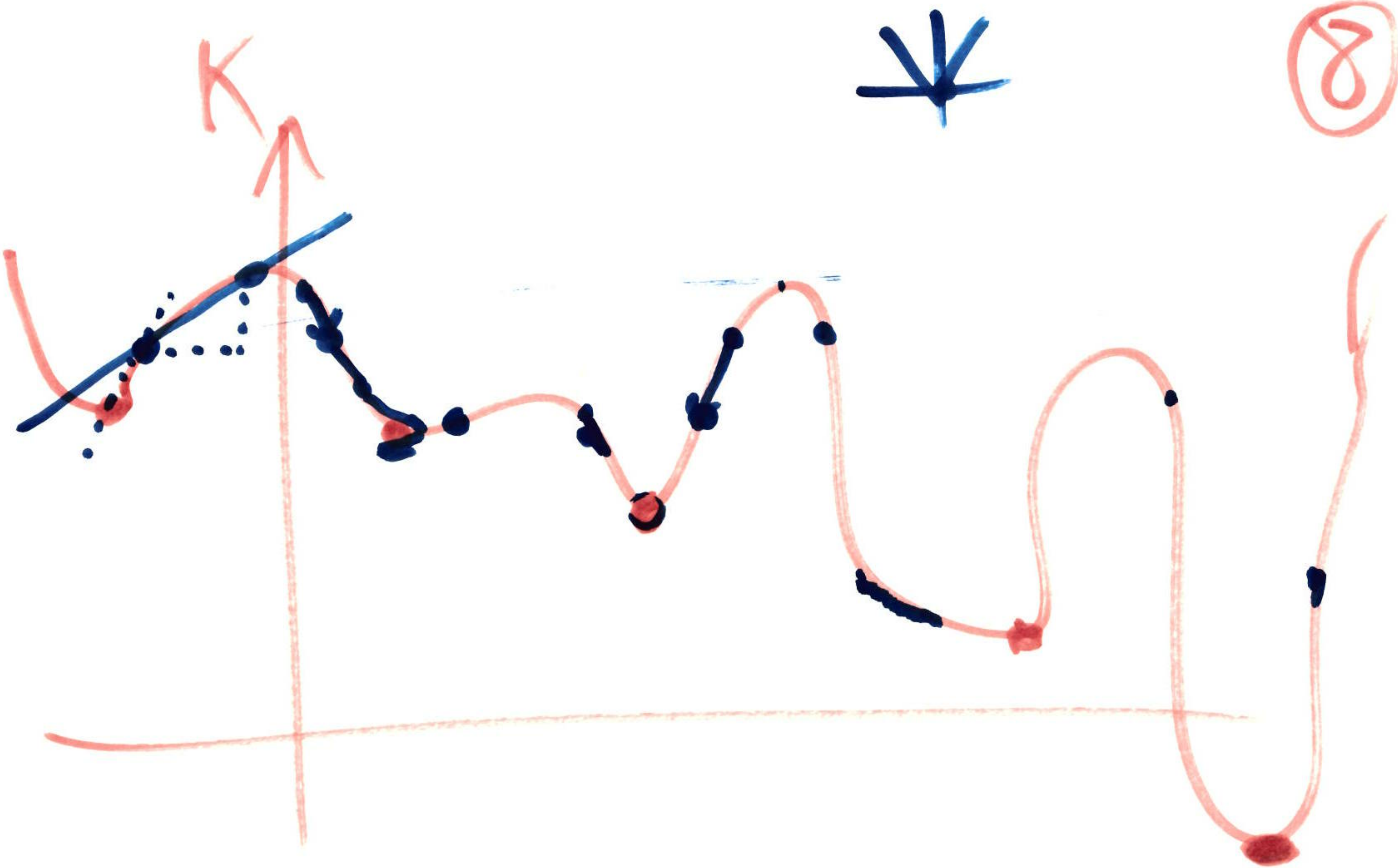
$$(\nabla f)(p) = \begin{pmatrix} \frac{\partial f}{\partial x_1}(p) \\ \vdots \\ \frac{\partial f}{\partial x_n}(p) \end{pmatrix}$$

$\nabla f(p)$ zeigt in Richtung des steilsten Anstiegs f



$$f\begin{pmatrix}x\\y\end{pmatrix} = x^2 - \sin(y)\cdot x$$

$$(\nabla f)\begin{pmatrix}x\\y\end{pmatrix} = \begin{pmatrix} 2x - \sin(y) \\ -x\cos(y) \end{pmatrix}$$


K

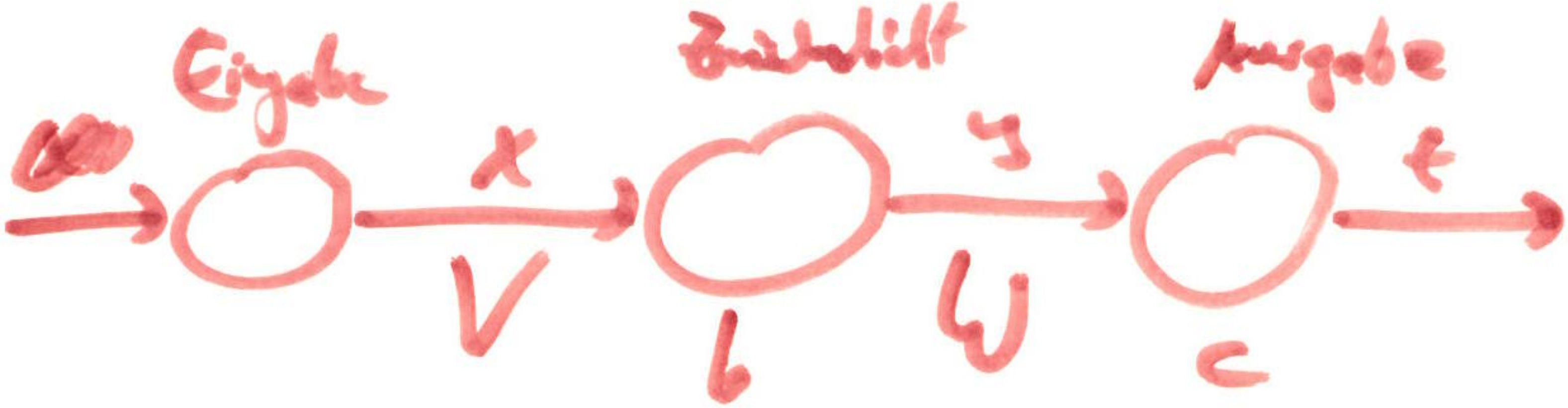
Wie berechnet man ∇K? „nabla K" ⑨

~~1~~ $\nabla K \approx \dfrac{K(A + \text{ein bisschen}) - K(A)}{\text{„ein bisschen"}}$ ← produziert in der Praxis Zufallszahlen

2 ∇K per Hand berechnen

~~3~~ ∇K durch „symb. Diff." bestimmen

4 ∇K durch „automat. Diff." bestimmen



$$\hat{y} = Vx + b \qquad \hat{z} = Wy + c$$

$$y = \sigma(\hat{y}) \qquad z = \sigma(\hat{z})$$

$$K = \sum \dots$$

ges: ∇K

genauer: $\nabla_{v_{1,1}} K, \dots, \nabla_{b_1} K,$

$\dots$

Def: $\delta_k := \frac{\partial K}{\partial \hat{z}_k}$, $\gamma_j := \frac{\partial K}{\partial \hat{y}_j}$

$\boxed{1}$ $\delta_k = \frac{\partial K}{\partial z_k} \sigma'(\hat{z}_k)$

$\boxed{2}$ $\gamma_j = (W^T \delta)_j \cdot \sigma'(\hat{y}_j)$

$\boxed{3}$ $\frac{\partial K}{\partial c_k} = \delta_k$, $\frac{\partial K}{\partial b_j} = \gamma_j$

$\boxed{4}$ $\frac{\partial K}{\partial w_{kj}} = \gamma_j \delta_k$, $\frac{\partial K}{\partial v_{ji}} = x_i \gamma_j$

Zu $\boxed{1}$:

(11a)

$$\delta_k = \frac{\partial K}{\partial \hat{z}_k} = \frac{\partial K}{\partial z_k} \cdot \frac{\partial z_k}{\partial \hat{z}_k}$$

Kettenregel

$$= \frac{\partial K}{\partial z_k} \cdot \sigma'(\hat{z}_k)$$

$$z_k = \sigma(\hat{z}_k)$$

$$y(x) \rightsquigarrow y'(x) = \frac{\partial y}{\partial x}$$

116

Convolutional NN