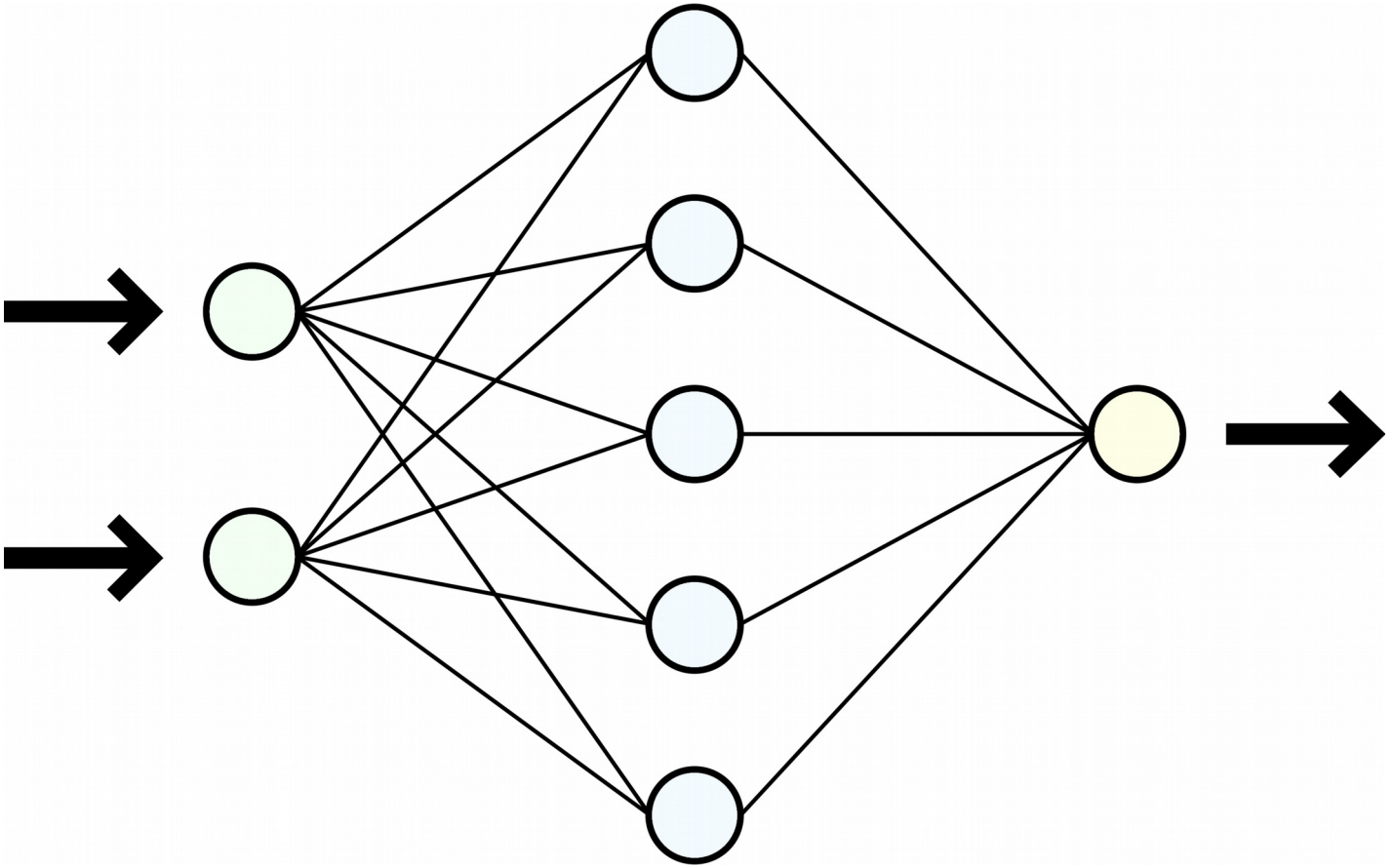
Advanced data analysis

Neural networks and deep learning

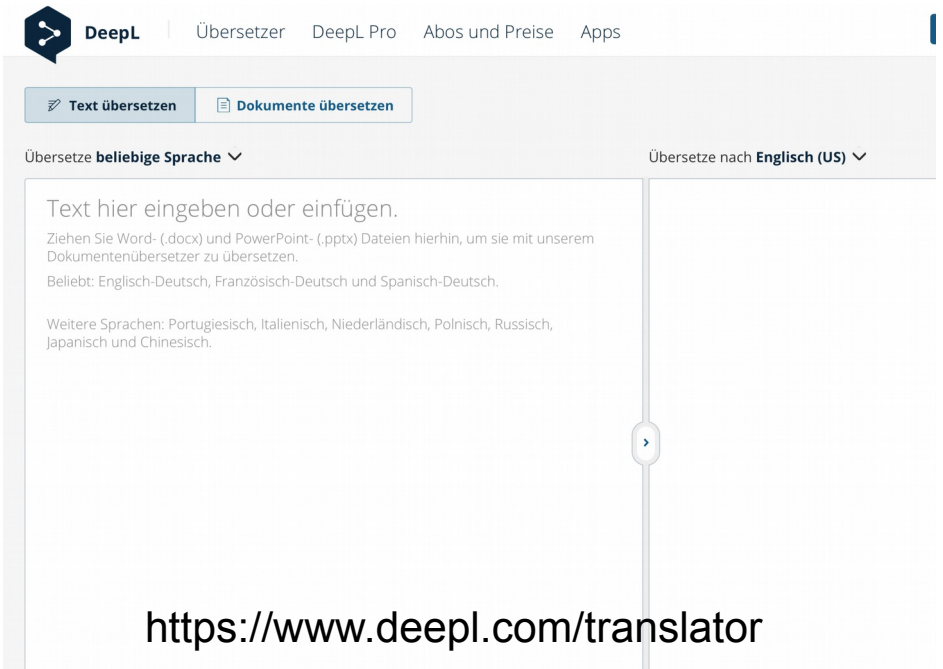


Stefan Kunz & Ralf B. Schäfer

University of Koblenz-Landau 2021/22

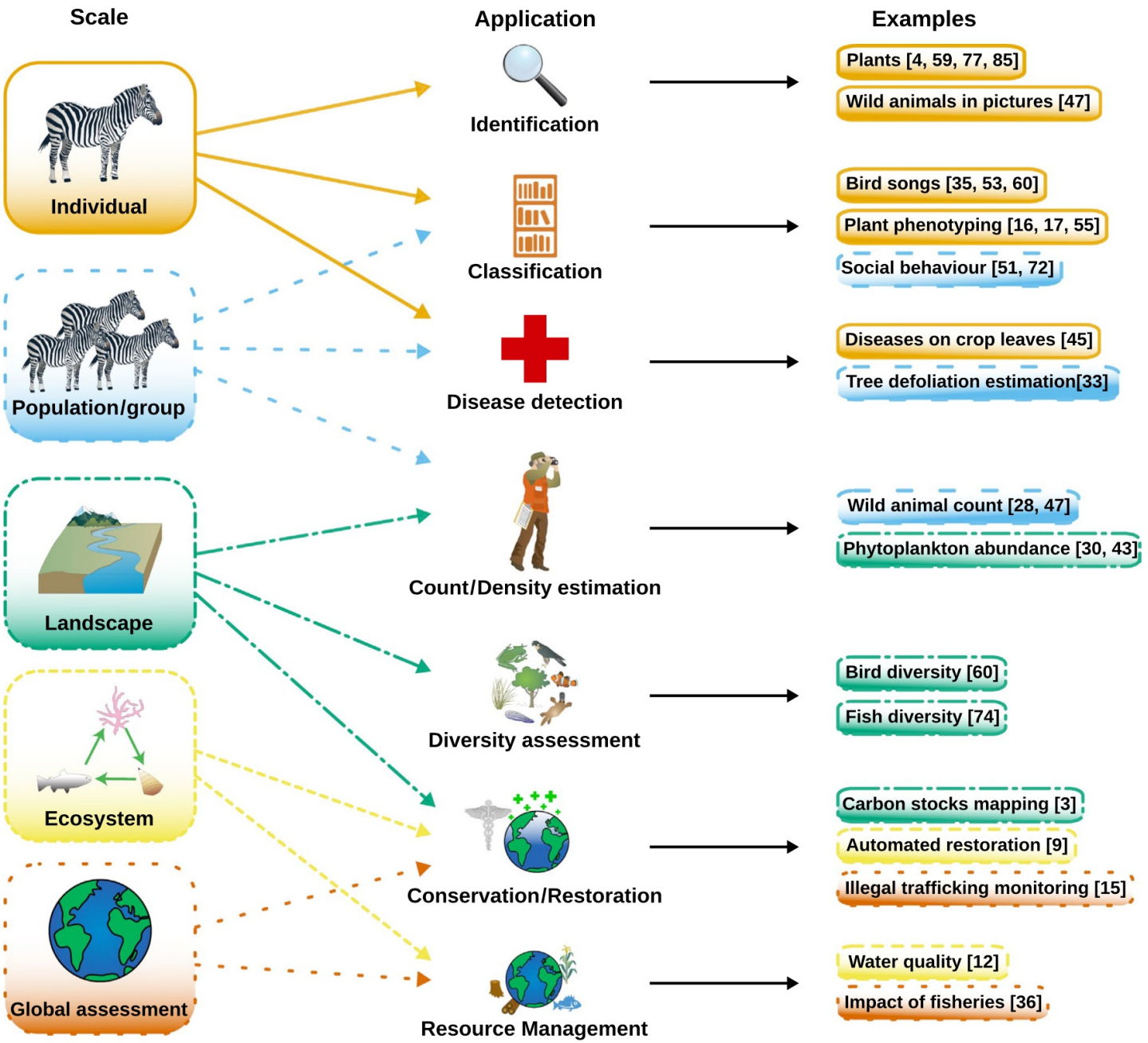
Deep Learning: Popular applications

www.tesla.com



https://www.deepl.com/translator

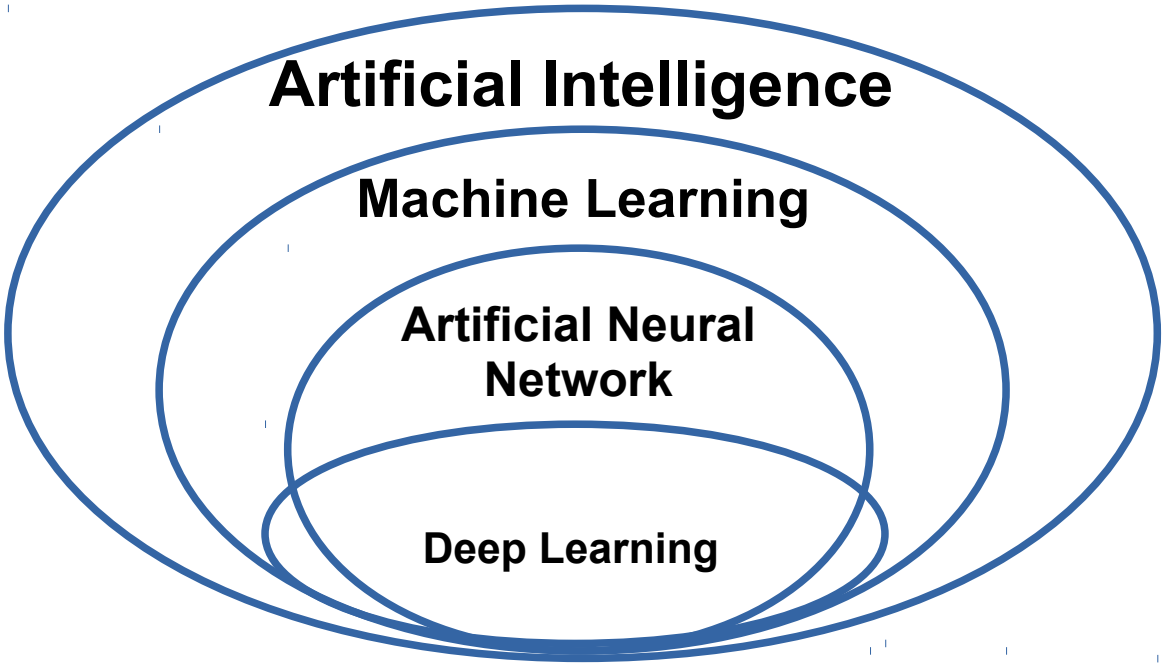
Deep Learning: Environmental applications



Christin et al. 2019 *Methods in Ecology and Evolution* 10: 1632-1644

Terminology

- Deep Learning (DL) often implemented as part of Artificial Neural Networks (ANN), which are part of Machine Learning (ML), which is part of Artificial Intelligence (AI)
- Definition AI: „the effort to automate intellectual tasks normally performed by humans“ (Chollet & Allaire 2018: 3)



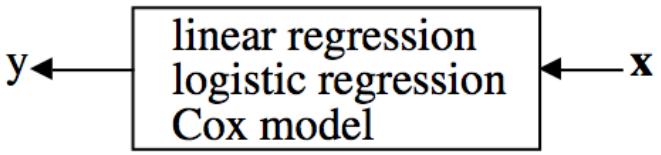
Recap: Two cultures

Data modelling culture (**Classical statistics**)

Common data model

$$\text{response variables} = f(\text{predictor variables, random noise, parameters})$$

Estimate parameters from data



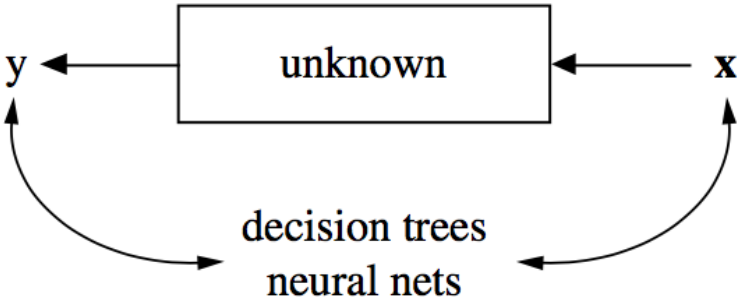
Model validation: Check model assumptions

Methods appropriate for all research goals, but partly less powerful

Example: Linear model, GLM

Algorithmic modelling culture (**Machine Learning**)

Find algorithm that operates on x to predict y



Model validation: Predictive accuracy

Methods primarily appropriate for prediction and exploration, and partly estimation

Examples: CART, Random Forest, Boosted Regression Trees

Breiman 2001 *Statistical Science* 16: 199

Recap: Two cultures

Data modelling culture (**Classical statistics**)

Algorithmic modelling culture (**Machine Learning**)

	Traditional regressions methods	Pure prediction algorithms
1.	Surface plus noise models (continuous, smooth)	Direct prediction (possibly discrete, jagged)
2.	Scientific truth (long-term)	Empirical prediction accuracy (possibly short-term)
3.	Parametric modeling (causality)	Nonparametric (black box)
4.	Parsimonious modeling (researchers choose covariates)	Anti-parsimony (algorithm chooses predictors)
5.	x $p \times n$: with $p \ll n$ (homogeneous data)	$p \gg n$, both possibly enormous (mixed data)
6.	Theory of optimal inference (mle, Neyman–Pearson)	Training/test paradigm (Common Task Framework)

Efron 2020 *Journal of the American Statistical Association* 115: 636-655

ANNs

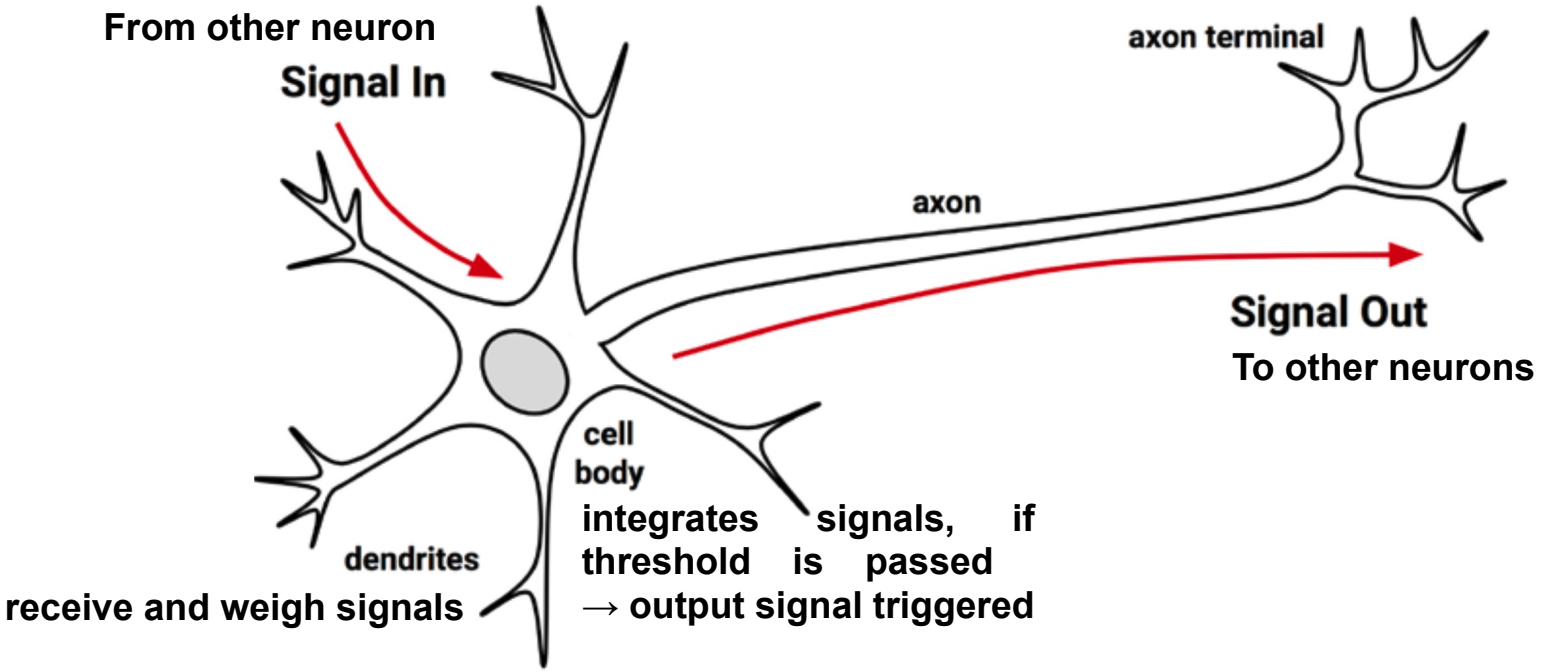
"*There has been a great deal of hype surrounding neural networks, making them seem magical and mysterious. As we make clear in this section, they are just nonlinear statistical models.*" (Hastie et al. 2017: 392)

- Around since 1940ies, three waves (with different names)
- Third wave: Deep Learning since 2006 owed to explosion of image data and computational power

ANNs

"*There has been a great deal of hype surrounding neural networks, making them seem magical and mysterious. As we make clear in this section, they are just nonlinear statistical models.*" (Hastie et al. 2017: 392)

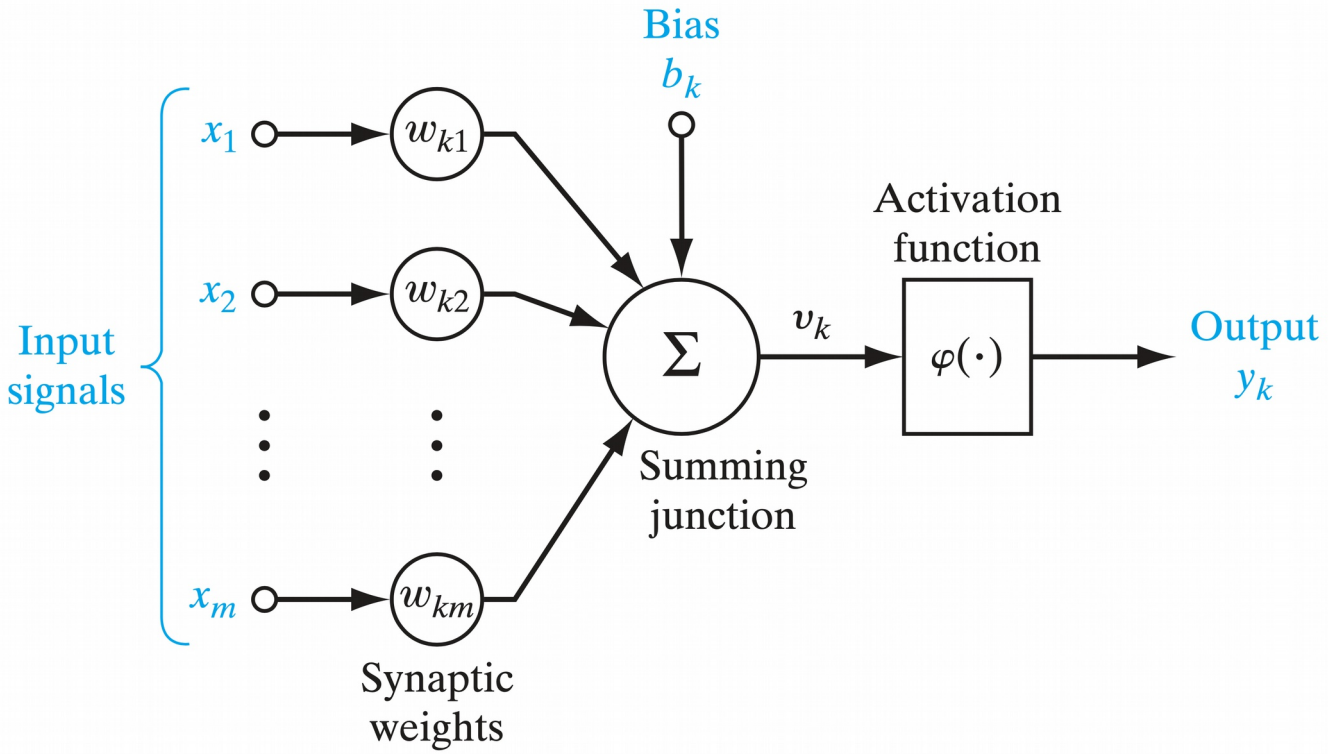
- Around since 1940ies, three waves (with different names)
- Third wave: Deep Learning since 2006 owed to explosion of image data and computational power
- Designed as conceptual model of human brain



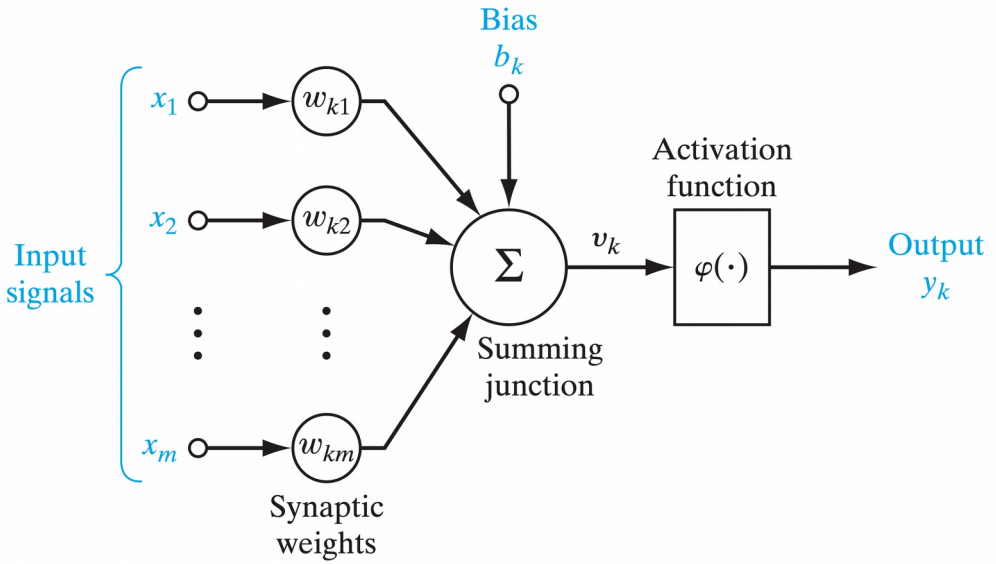
Lantz 2019: 219

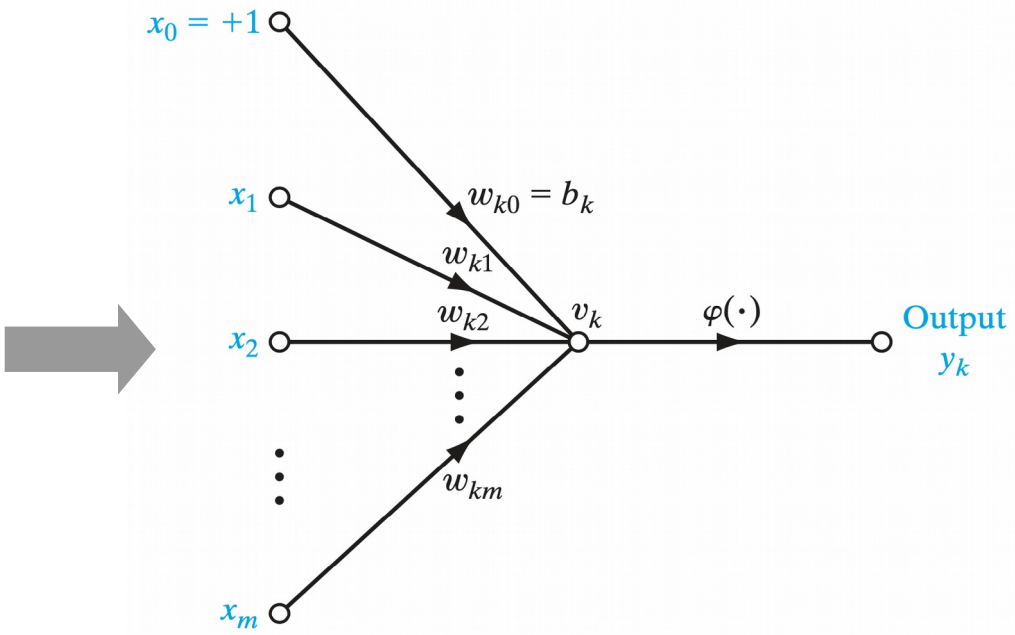
Model of Neuron k

- Three core elements of neuron:
  - **Synapse** (connecting link) responding to input x with weight w
  - **Adder** summing up (summing junction)
  - **Activation function** (normalizes output to a permissible range)



Model of Neuron k





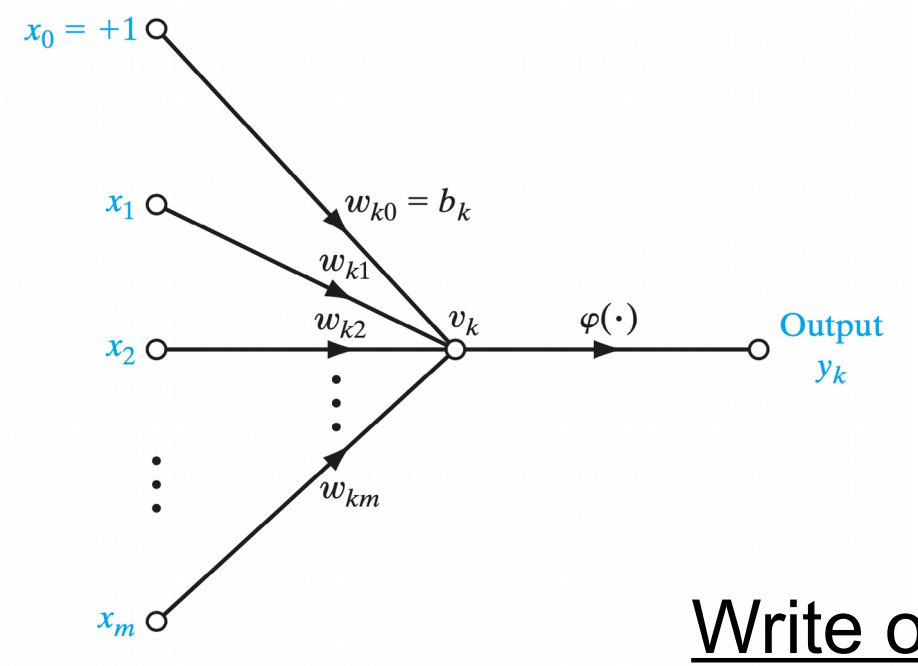
Model formulation:

$$y_k = \varphi(v_k)$$

$$v_k = \sum_{j=0}^{m} w_{k,j} x_j$$

Haykin 2009: 11, 17

Model of Neuron k



Model formulation:

$$y_k = \varphi(v_k)$$

$$v_k = \sum_{j=0}^{m} w_{k,j} x_j$$

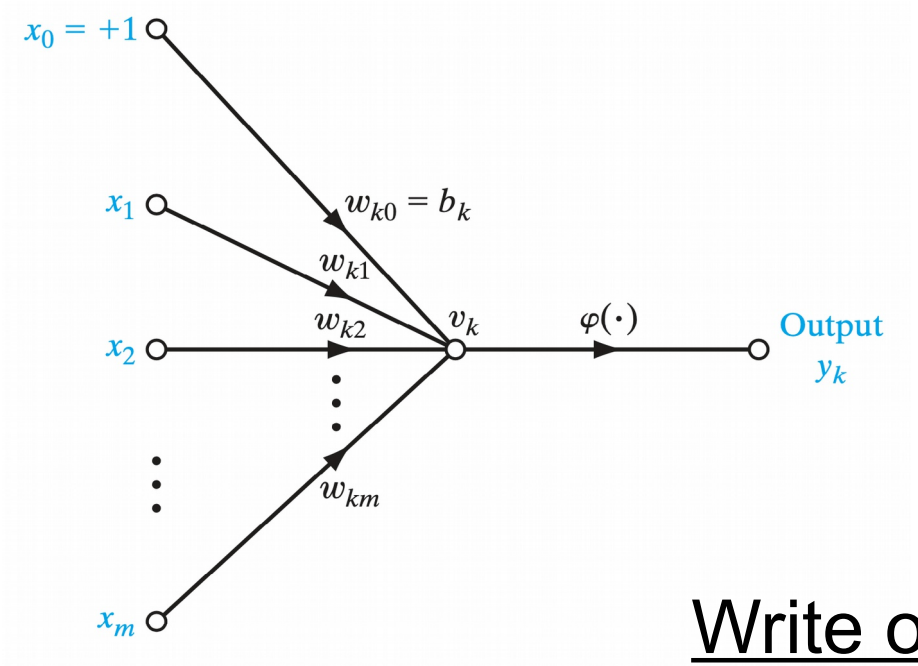
Write out sum sign:

$$v_k = w_{k0} x_0 + w_{k1} x_1 + w_{k2} x_2 + \ldots + w_{km} x_m \Leftrightarrow$$

$$v_k = w_{k0} + w_{k1} x_1 + w_{k2} x_2 + \ldots + w_{km} x_m$$

Haykin 2009: 11, 17

Model of Neuron k



Model formulation:

$$y_k = \varphi(v_k)$$

$$v_k = \sum_{j=0}^{m} w_{k,j} x_j$$

Write out sum sign:

$$v_k = w_{k0}x_0 + w_{k1}x_1 + w_{k2}x_2 + \ldots + w_{km}x_m \Leftrightarrow$$

$$v_k = w_{k0} + w_{k1}x_1 + w_{k2}x_2 + \ldots + w_{km}x_m$$

Note similarity to the linear model for the identity function:

$$\varphi(v_k) = v_k$$

Linear model formulation:

$$y_i = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \ldots + \beta_p x_{i,p} + \varepsilon_i$$

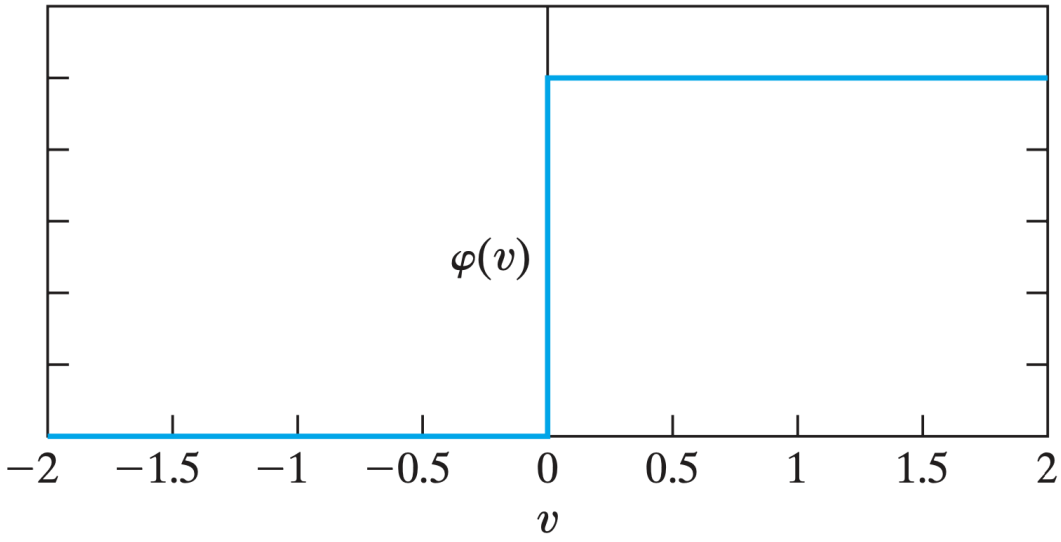
Haykin 2009: 11, 17

Activation functions

Step function

- Parallels biology, neuron fires or not (Classical McCulloch–Pitts model)
- Not widely used

$$\varphi(v_k) = \begin{cases} 1 & \text{if } v \geq 0 \\ 0 & \text{if } v < 0 \end{cases}$$

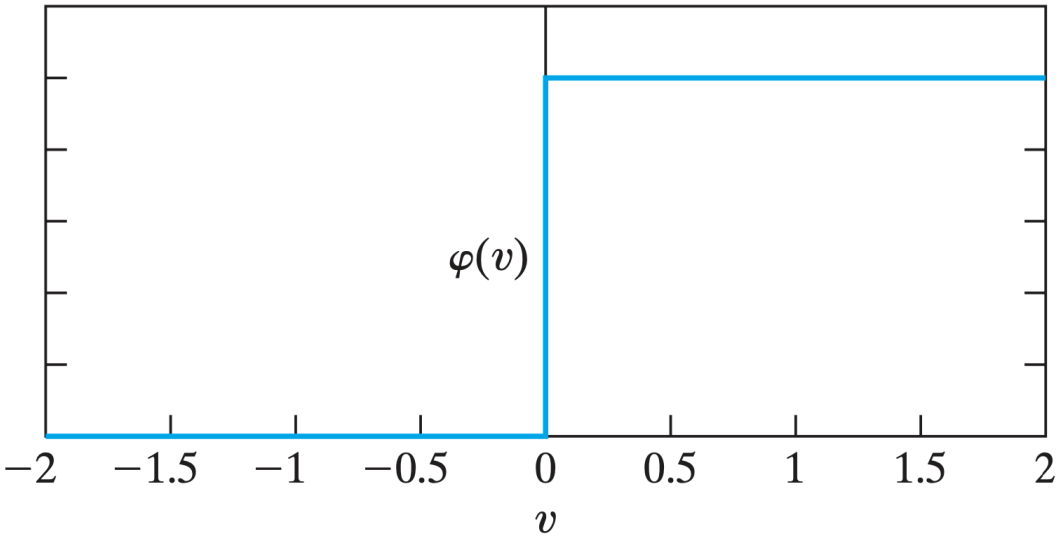


Activation functions

Step function

- Parallels biology, neuron fires or not (Classical McCulloch–Pitts model)
- Not widely used

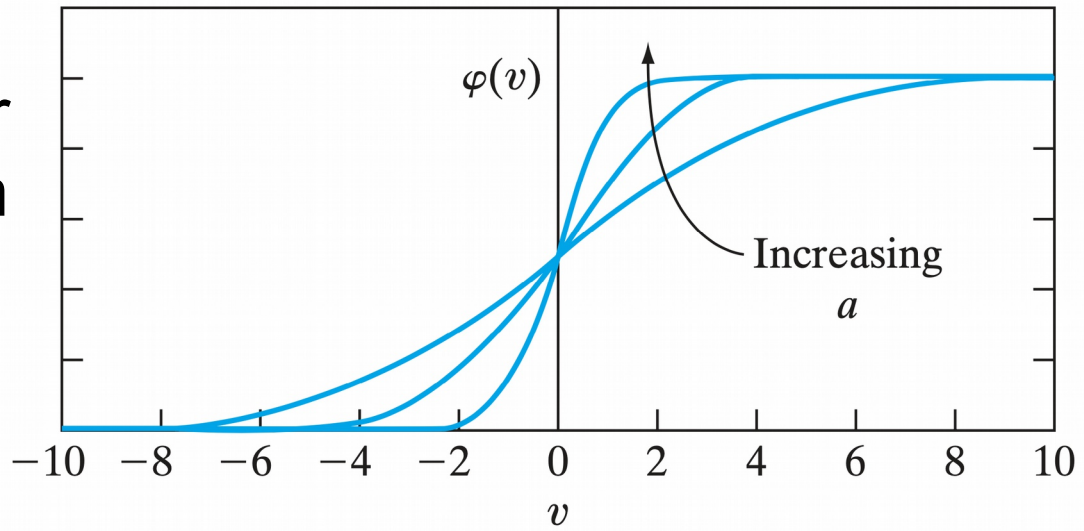
$$\varphi(v_k) = \begin{cases} 1 & \text{if } v \geq 0 \\ 0 & \text{if } v < 0 \end{cases}$$



Sigmoid function

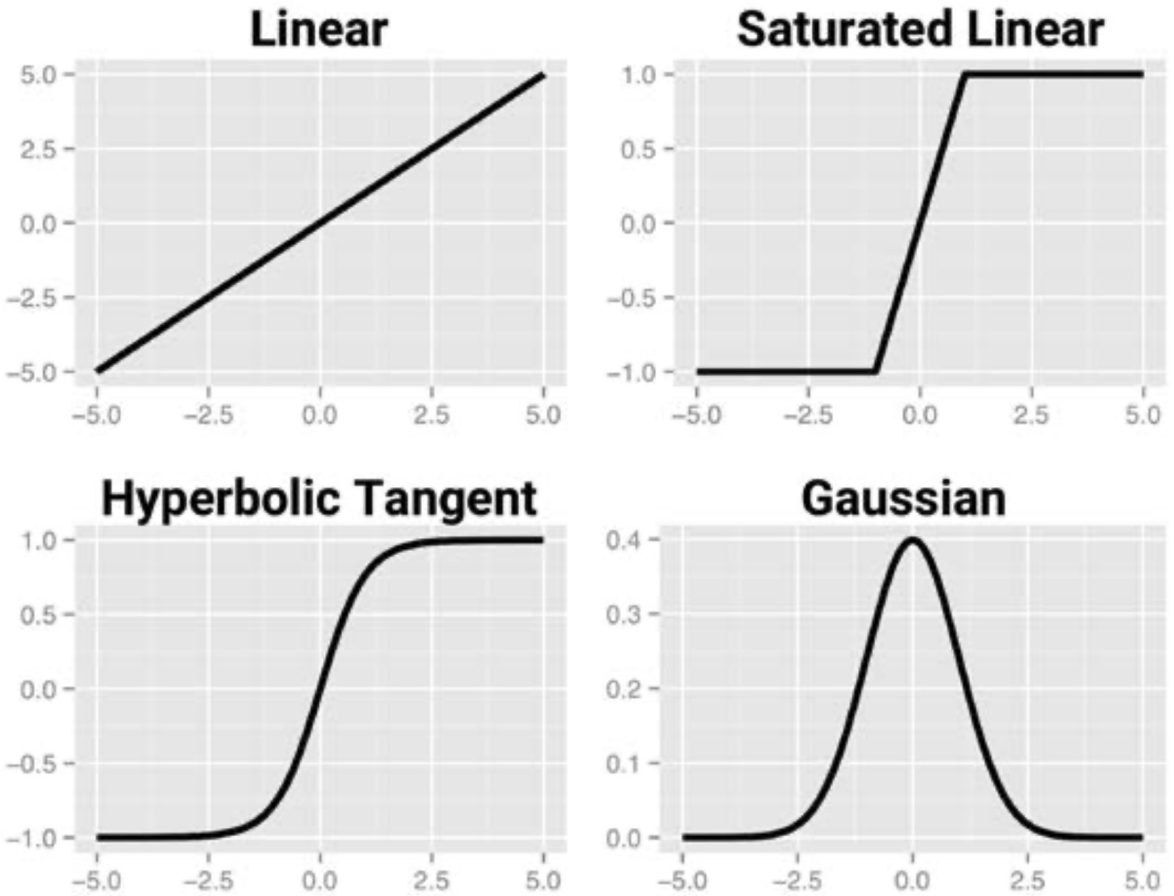
- Differentiable → Crucial for efficient optimization algorithm
- Widely used

$$\varphi(v_k) = \frac{1}{1+e^{-av}}$$



Activation functions

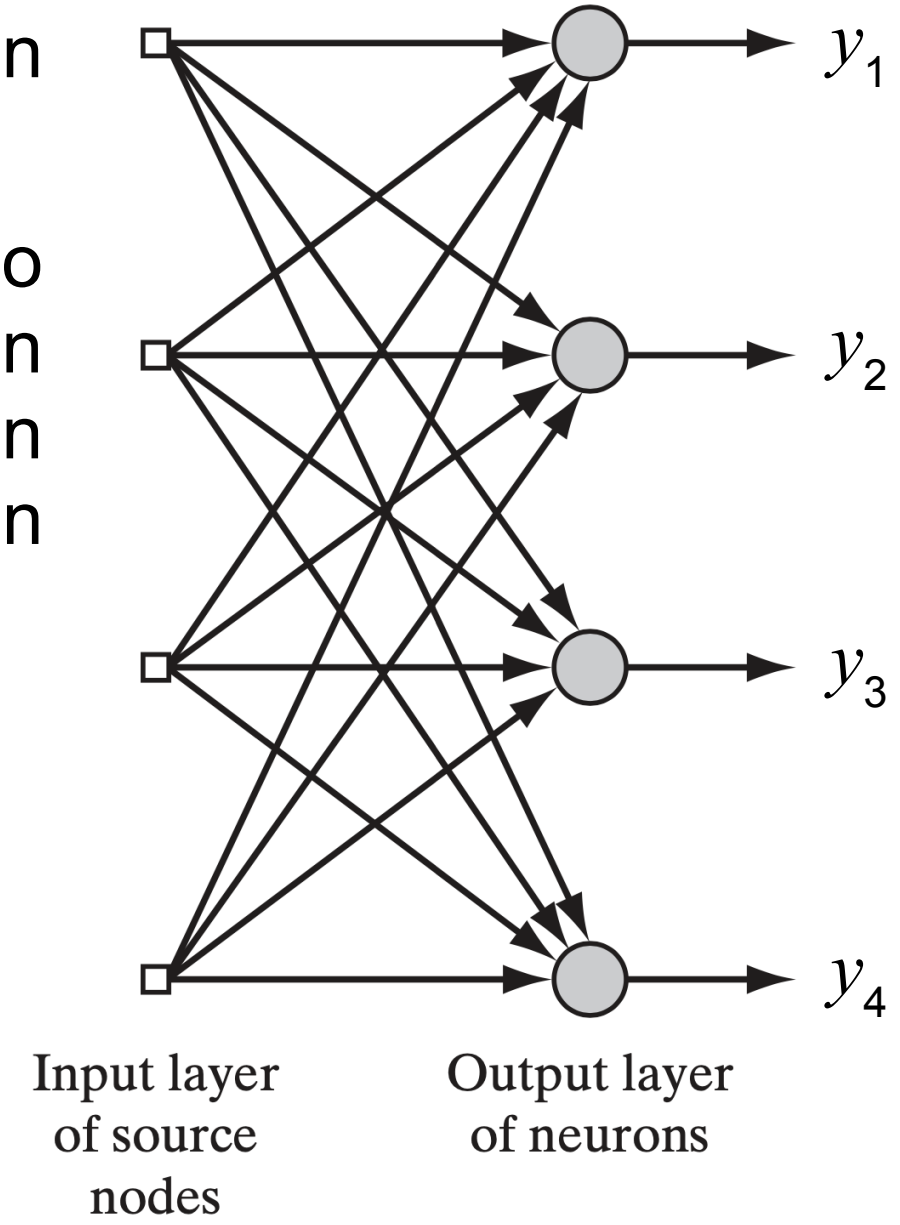
- Main difference is output signal range → Select final activation function based on response (e.g. probabilities)



Lantz 2019: 244

Networks of neurons

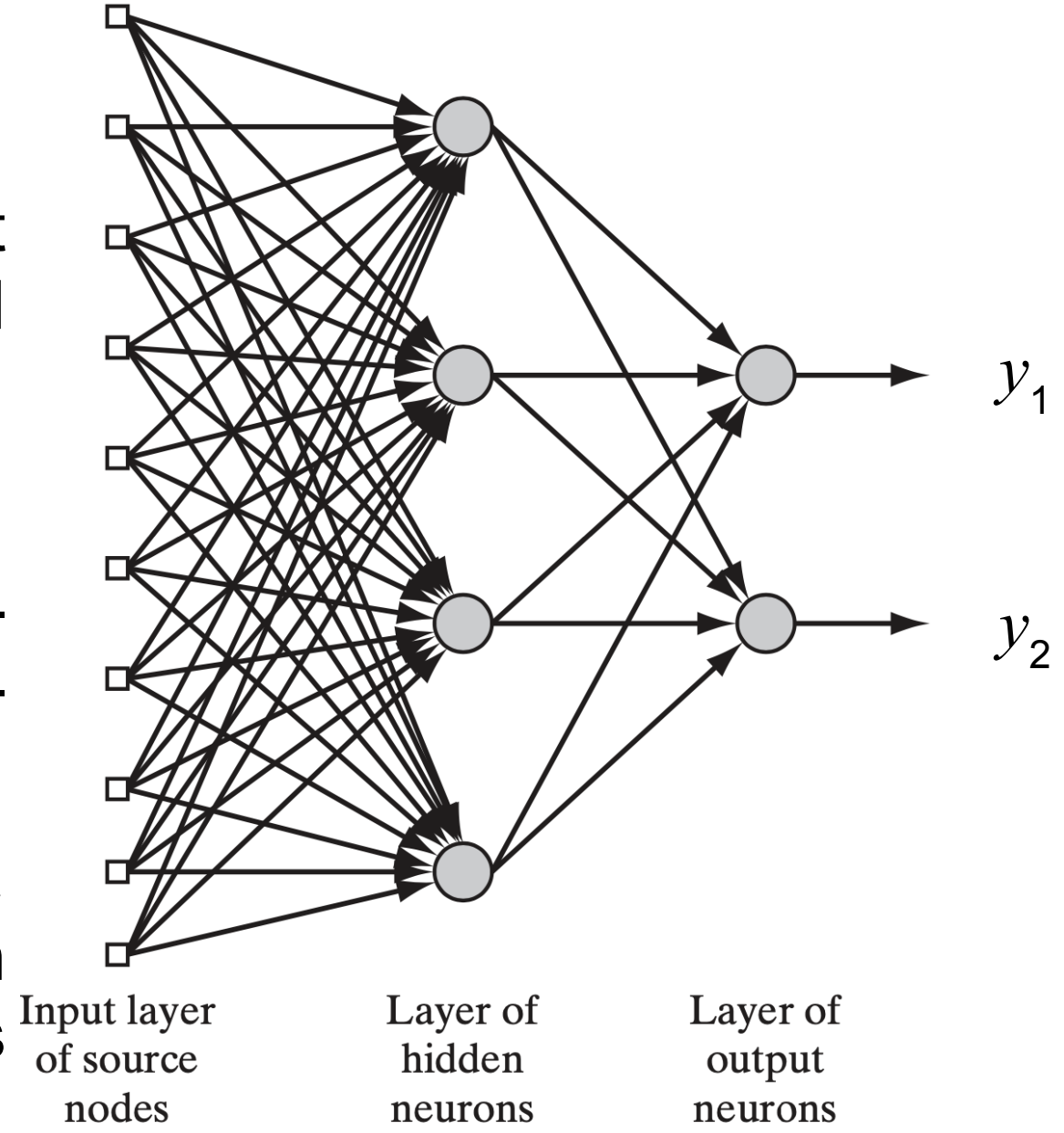
- Can be used for classification and regression
- Number of output neurons (also called nodes) depends on response (i.e. one neuron in regression, multiple in classification)
- Single-layer feedforward network



Haykin 2009: 21

Networks of neurons

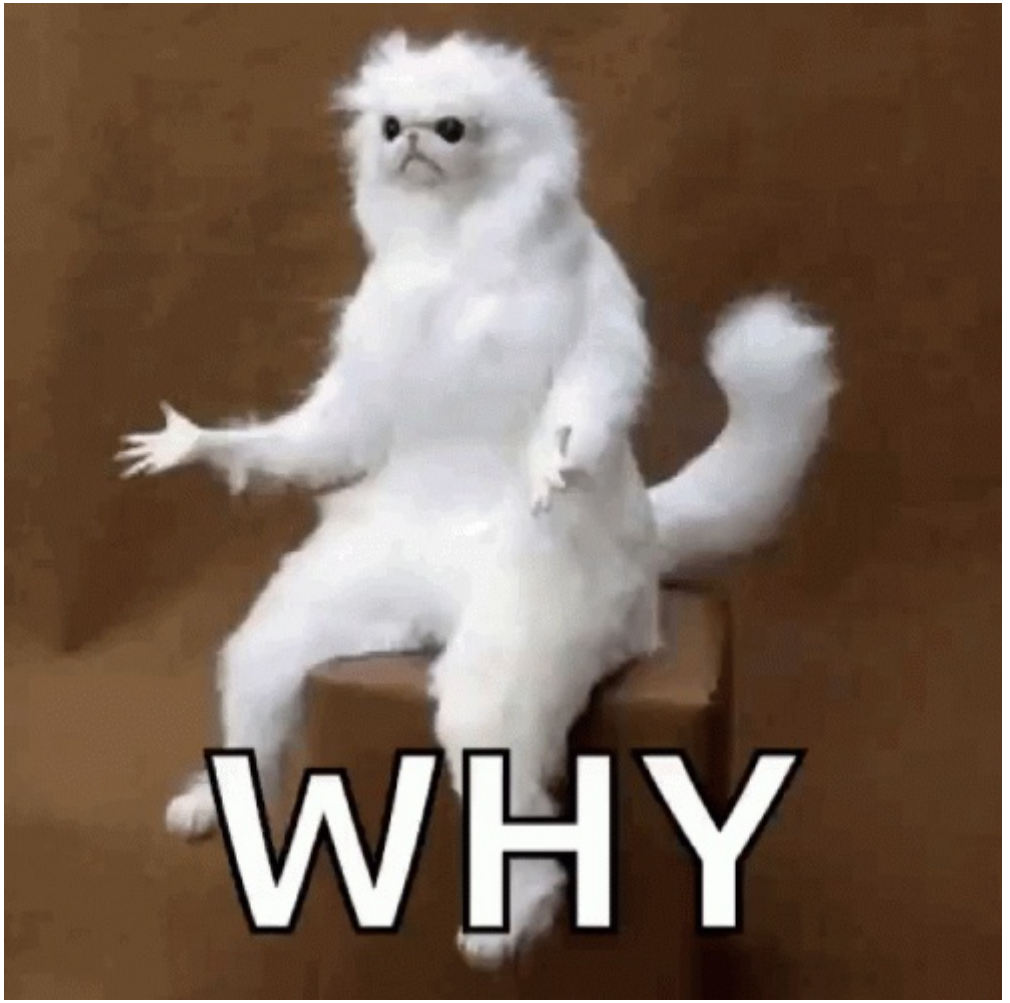
- Multilayer feedforward network
- Layers between input and output layer called „hidden“
- Fully connected
- 10-4-2 network: either 1-hidden-layer NN or 2-layer NN
- Shallow network *vs.* deep network with multiple hidden layers → Deep Learning



Haykin 2009: 22

Characterisation of networks

- Activation functions: can differ between layers
- Network topology:
  - Number of layers
  - Number of neurons per layer
  - Feeding direction
  - Connection type
- Training algorithm: how weights are learned

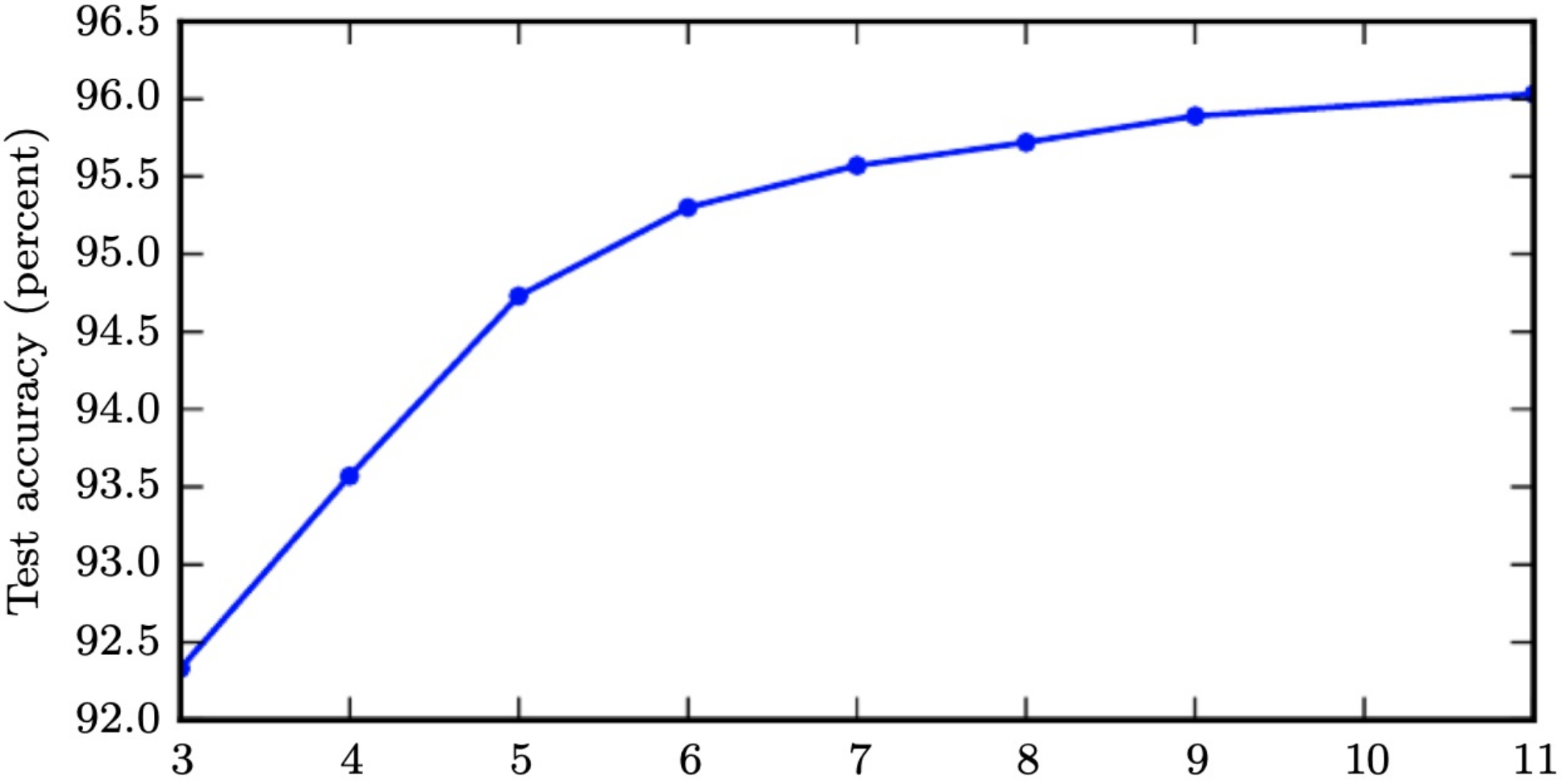


ALL THIS LAYER STUFF?

https://tenor.com/view/confused-white-persian-guardian-why-gif-11908780

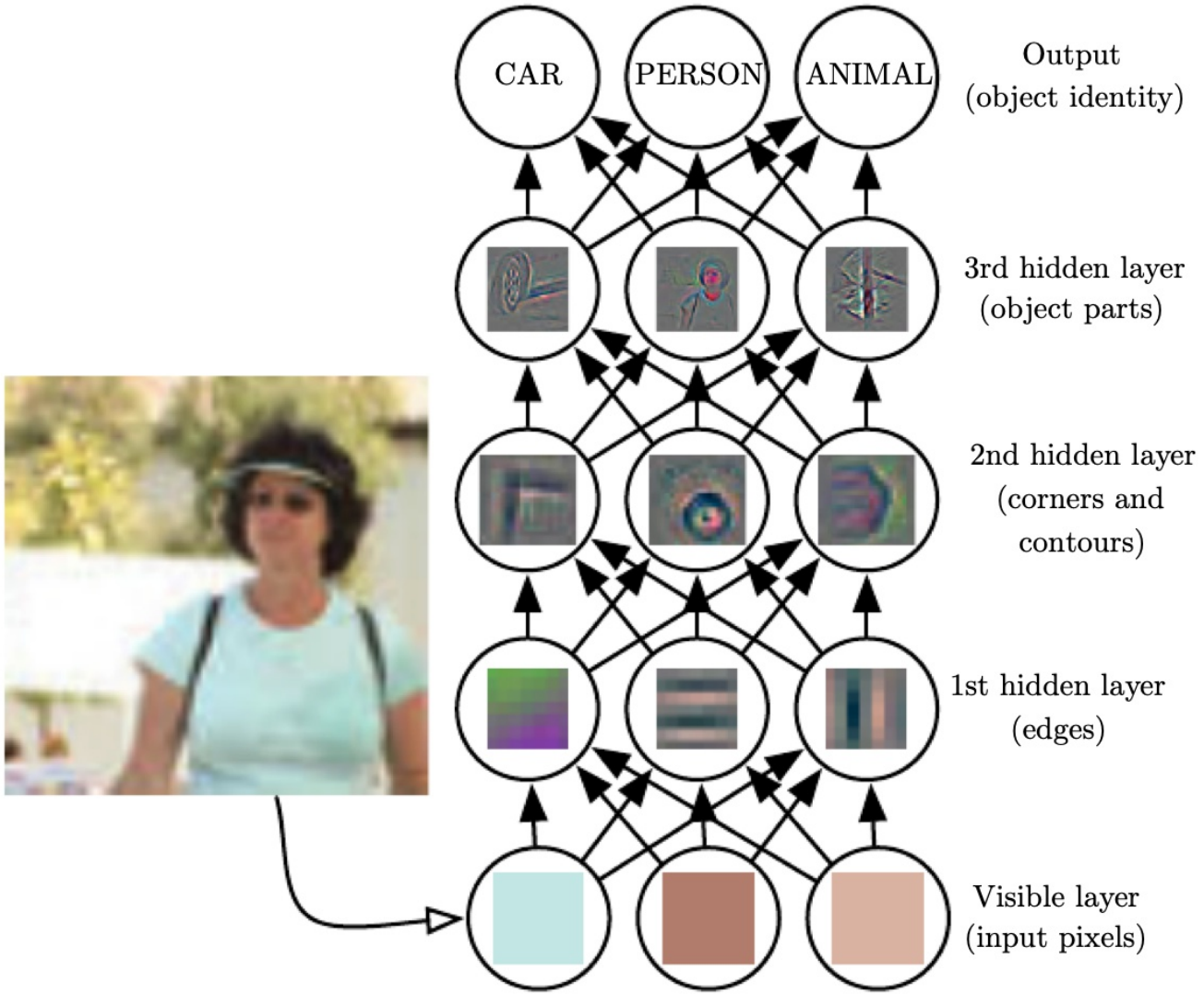
Why multiple layers?

Empirically, more layers typically result in better accuracy on the test set



Why multiple layers?

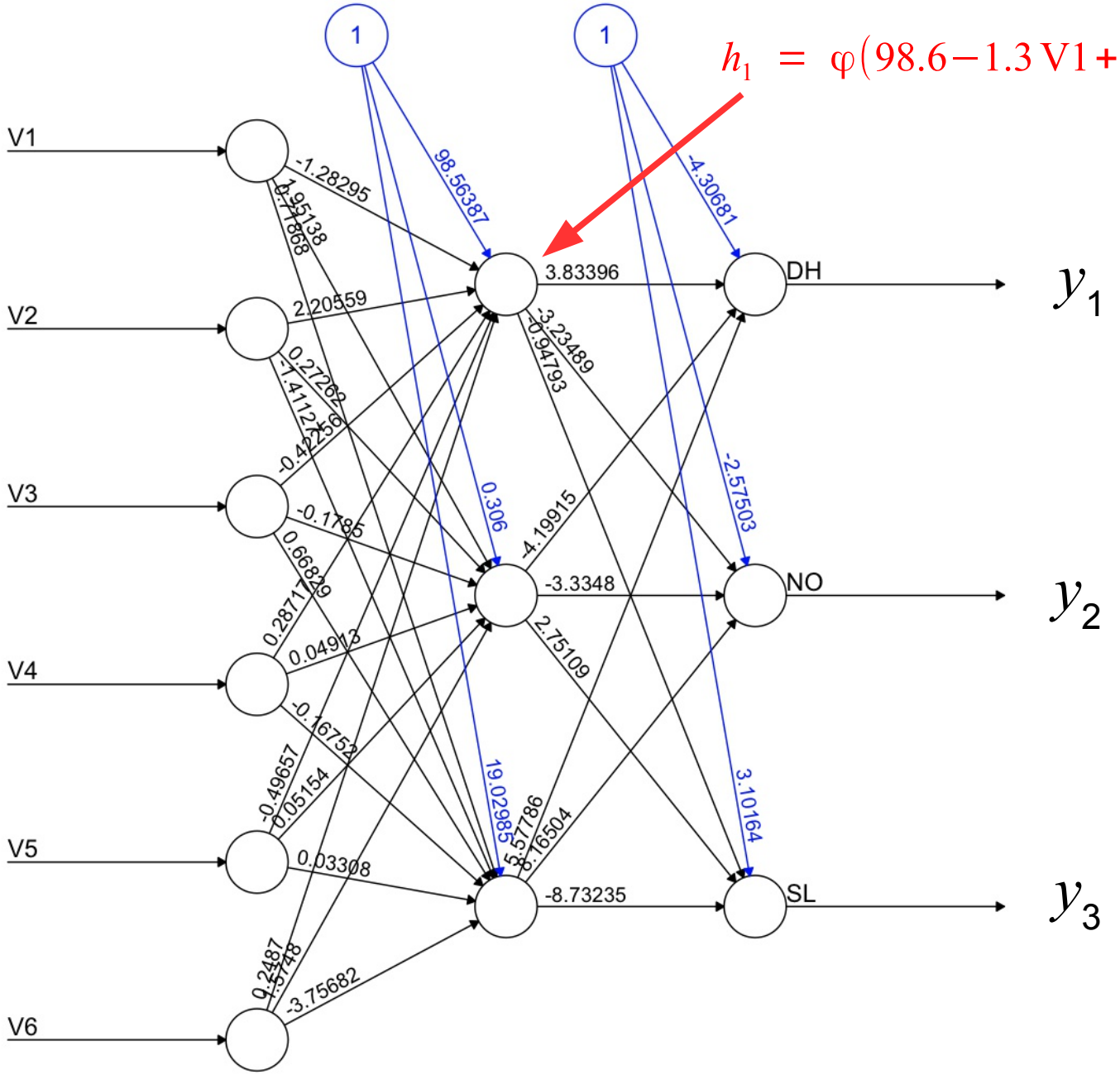
Different characteristics of picture captured by each layer



Building an own ANN exercise with Stefan Kunz

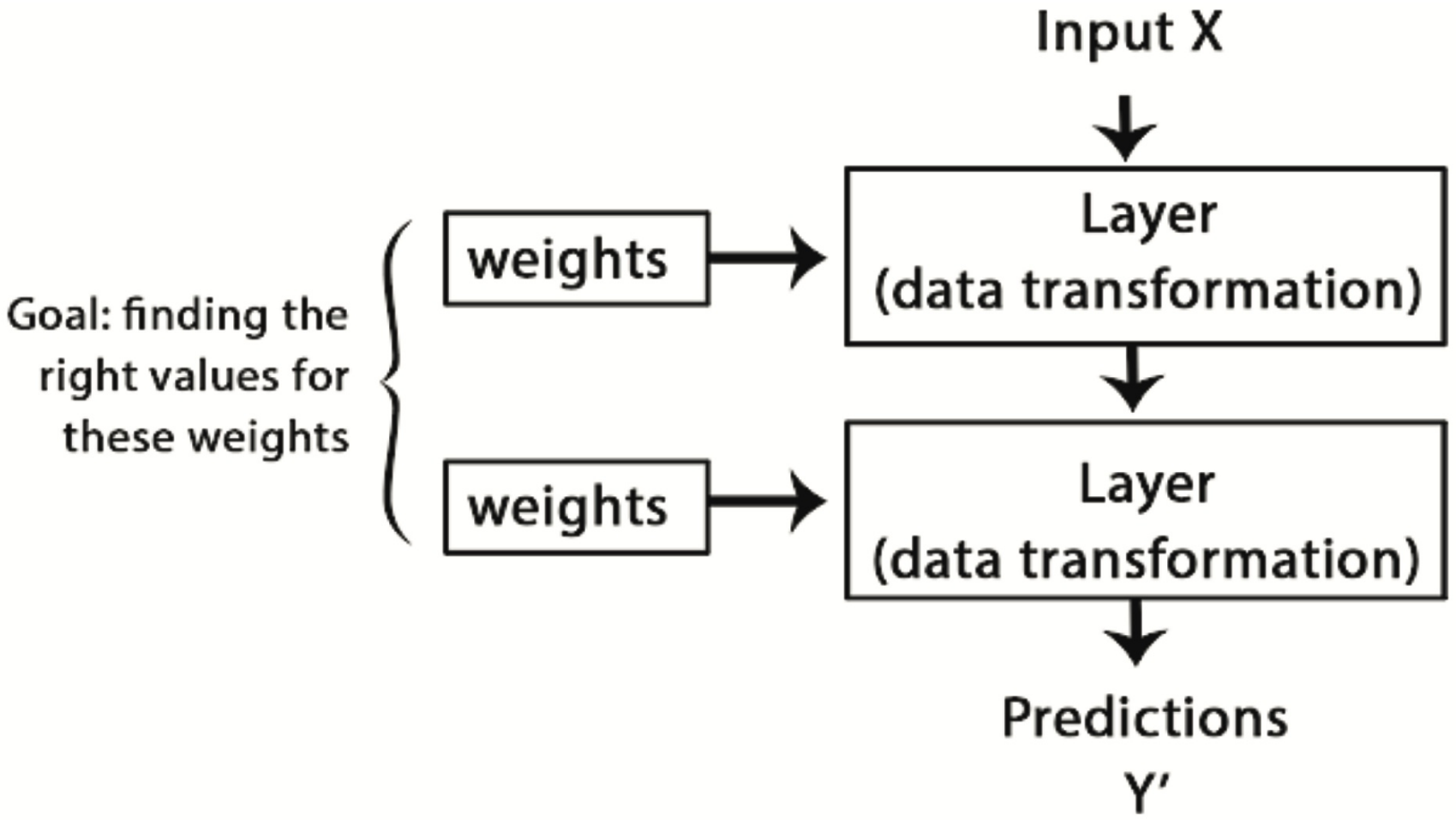
Interpretation

$$h_1 = \varphi(98.6 - 1.3\,V1 + 2.2\,V2 - 0.42\,V3 + \ldots + 0.25\,V6)$$



Error: 28.0394 Steps: 9894

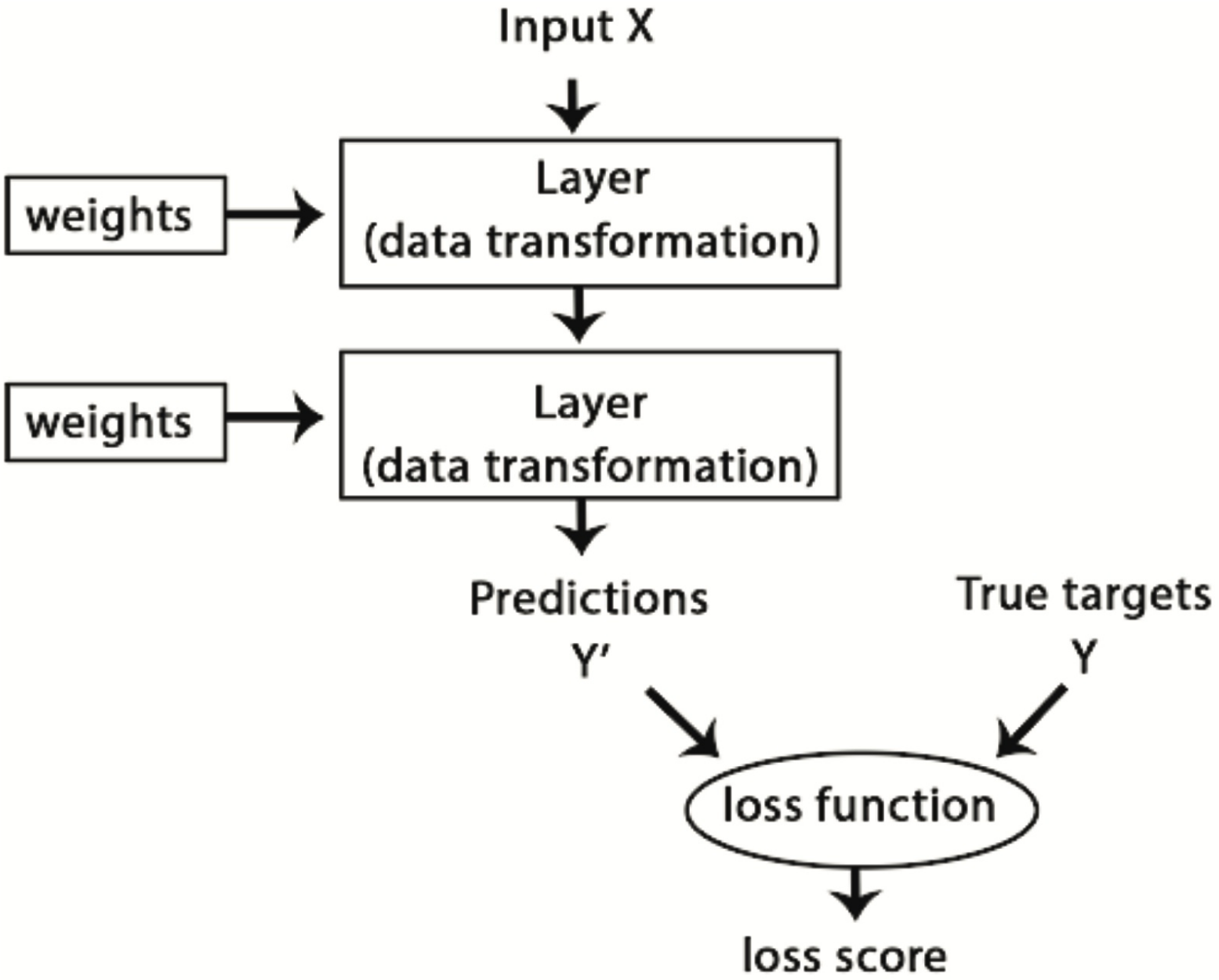
How are the weights determined?



Number of weights for l layers and d neurons/nodes is ld^2

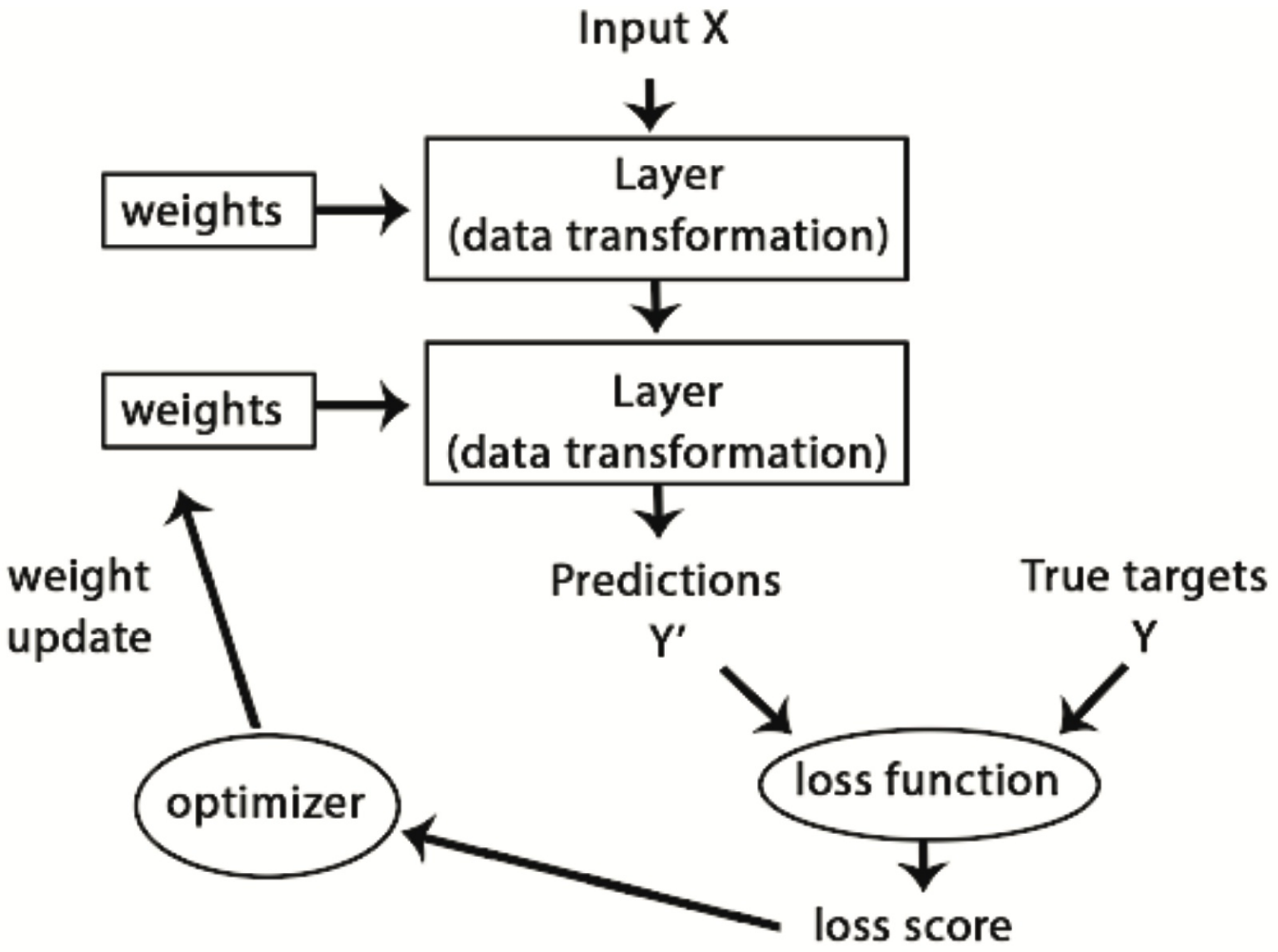
Chollet & Allaire 2017: 8

How are the weights determined?



Chollet & Allaire 2017: 9

How are the weights determined?



Determining weights that minimize the loss function is called learning (deep if multiple layers)

Chollet & Allaire 2017: 10

Loss functions

Aim: Determine parameters θ that reduce loss function $R(\theta)$

Loss function for the regression case (RSS):

$$R(\theta) = R(w) = \sum_{i=1}^{n}(y_i - f(x_i))^2 = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Loss function for the classification case (Cross-Entropy):

$$R(\theta) = R(w) = -\sum_{i=1}^{n}\sum_{j=1}^{q} y_{iq}\log(\hat{\pi}_{iq})$$

Loss functions

Aim: Determine parameters θ that reduce loss function $R(\theta)$

Loss function for the regression case (RSS):

$$R(\theta) = R(w) = \sum_{i=1}^{n}(y_i - f(x_i))^2 = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Loss function for the classification case (Cross-Entropy):

$$R(\theta) = R(w) = -\sum_{i=1}^{n}\sum_{j=1}^{q} y_{iq}\log(\hat{\pi}_{iq})$$

Calculate Cross-Entropy for single case of binary classification

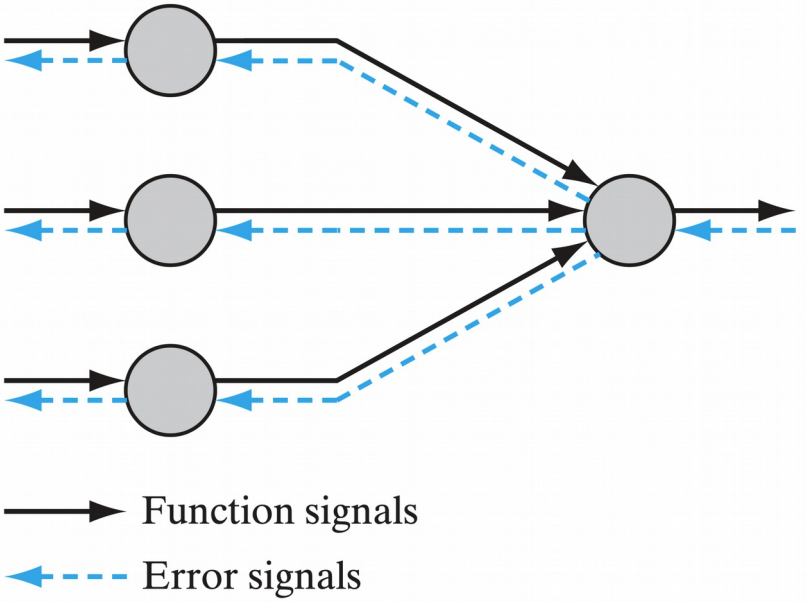
(q = 2, $y_{1,1}$ = 1, $y_{1,2}$ = 0)

$$R(\theta \mid y_{1,1} = 1, \hat{\pi}_{1,1} = 1) = -1\log(1) = 0$$
$$R(\theta \mid y_{1,1} = 1, \hat{\pi}_{1,1} = 0.5) = -1\log(0.5) = 0.7$$
$$R(\theta \mid y_{1,1} = 1, \hat{\pi}_{1,1} = 0.1) = -1\log(0.1) = 2.3$$

Loss function increases with difference between y and $\hat{\pi}$

Learning algorithm: Backpropagation

- Learning: Combination of Backpropagation and Stochastic gradient descent (SGD)
- Forward propagation: Compute all neurons and predicted y in network
- Backpropagation: Compute gradient of change in network based on loss function
- SGD: Update weights based on opposite direction of gradient.



Haykin 2009: 125

Backpropagation

Remember: $y_k = \varphi(v_k)$ with $v_k = \sum_{j=0}^{m} w_{k,j} x_j$

$$R(\theta) = R(w) = \sum_{i=1}^{n} (y_i - f(x_i))^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

The change in the loss function for small changes in weights is: $\frac{\partial R}{\partial w_k} \Delta w_k$

Aim: Minimize R

Backpropagation

The change in the loss function for small changes in weights is: $\frac{\partial R}{\partial w_k}\Delta w_k$

Aim: Minimize R

If $\frac{\partial R}{\partial w_k}$ is large in absolute terms, then choose Δw_k with opposite sign to decrease R. If $\frac{\partial R}{\partial w_k}$ is small, changing Δw_k has minor effect. $\frac{\partial R}{\partial w_k}$ provides an error measure for the neuron k. Therefore, we define δ_k^L as error for k in layer L:

$$\delta_k^L = \frac{\partial R}{\partial w_k^L}\varphi'\left(v_k^L\right).$$

Backpropagation

$$\delta^L = \frac{\partial R}{\partial w_k^L} \varphi'\left(v_k^L\right)$$

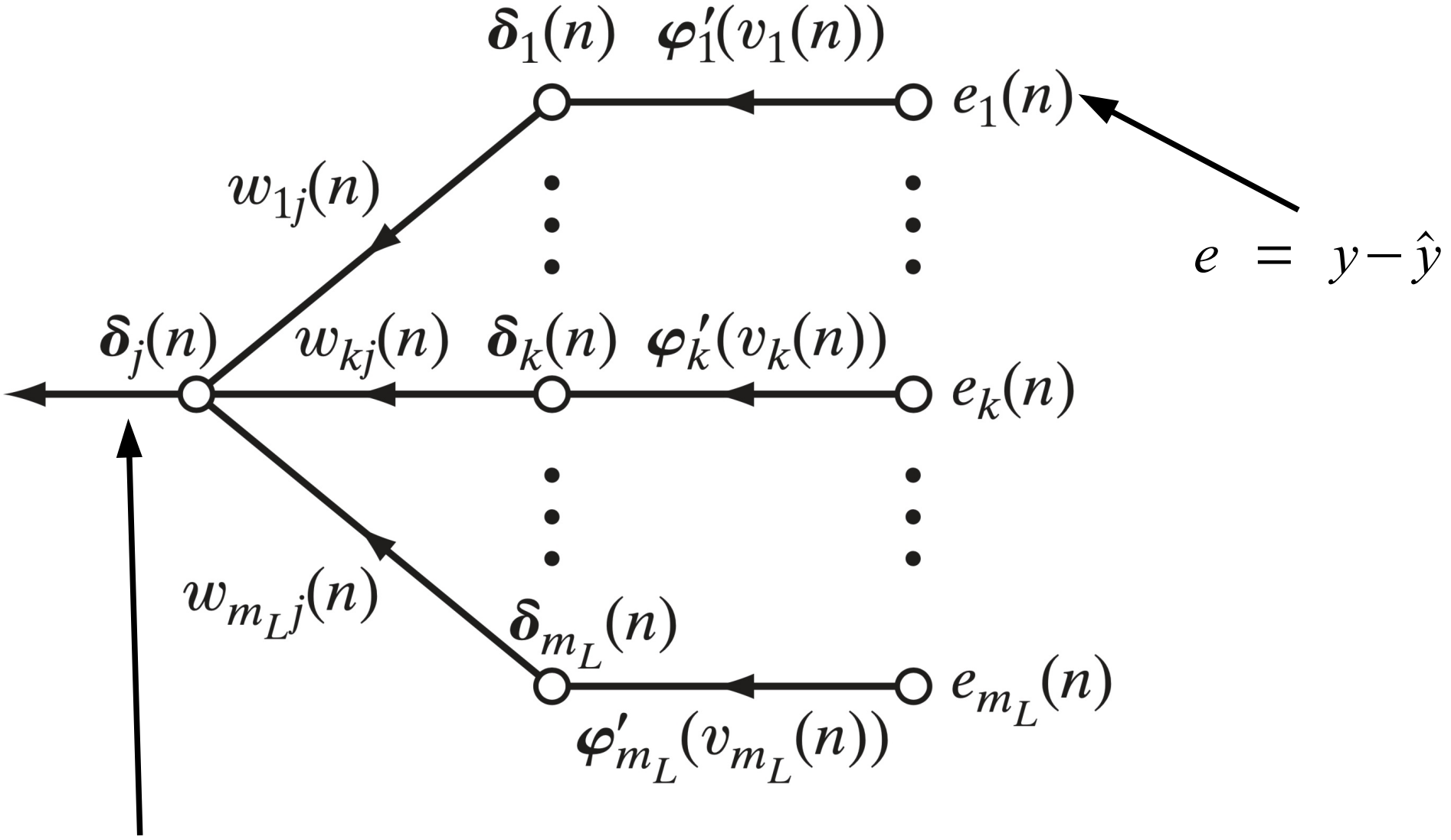
In matrix-based form this is:

$\delta^L = \nabla_w R \circ \varphi'\left(v^L\right)$ and called local gradient.

Calculate based on next layer for specific layer l:

$$\delta^l = \left(\left(w^{l+1}\right)^T \delta^{l+1}\right) \circ \varphi'\left(v^L\right)$$

Backpropagation



Express in terms of previous layers

Haykin 2009: 134

Stochastic gradient descent (SGD)

Remember: $\frac{\partial R}{\partial w_k} \Delta w_k$

- We know how we can calculate all error measures in the network ($\rightarrow$ Backpropagation) via partial derivatives
- But how do we set Δw ?

Stochastic gradient descent (SGD)

Remember: $\frac{\partial R}{\partial w_k} \Delta w_k$

- We know how we can calculate all error measures in the network (→ Backpropagation) via partial derivatives
- But how do we set Δw ?

$$\Delta w_k = -\eta \frac{\partial R}{\partial w_k} \Leftrightarrow w_k^{\text{new}} - w_k^{\text{old}} = -\eta \frac{\partial R}{\partial w_k} \Leftrightarrow w_k^{\text{new}} = w_k^{\text{old}} - \eta \frac{\partial R}{\partial w_k}$$

Learning rate

General: $\theta^{\text{new}} = \theta^{\text{old}} - \eta \nabla_\theta R$

Stochastic gradient descent (SGD)

Simple linear regression

$$R(\beta) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \qquad \hat{y}_i = \beta_0 + \beta_1 x_i$$

Quadratic loss function

Stochastic gradient descent (SGD)

Simple linear regression

$$R(\beta) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \qquad \hat{y}_i = \beta_0 + \beta_1 x_i$$

Quadratic loss function

Apply SGD

$$\theta^{\text{new}} = \theta^{\text{old}} - \eta \nabla_\theta R$$

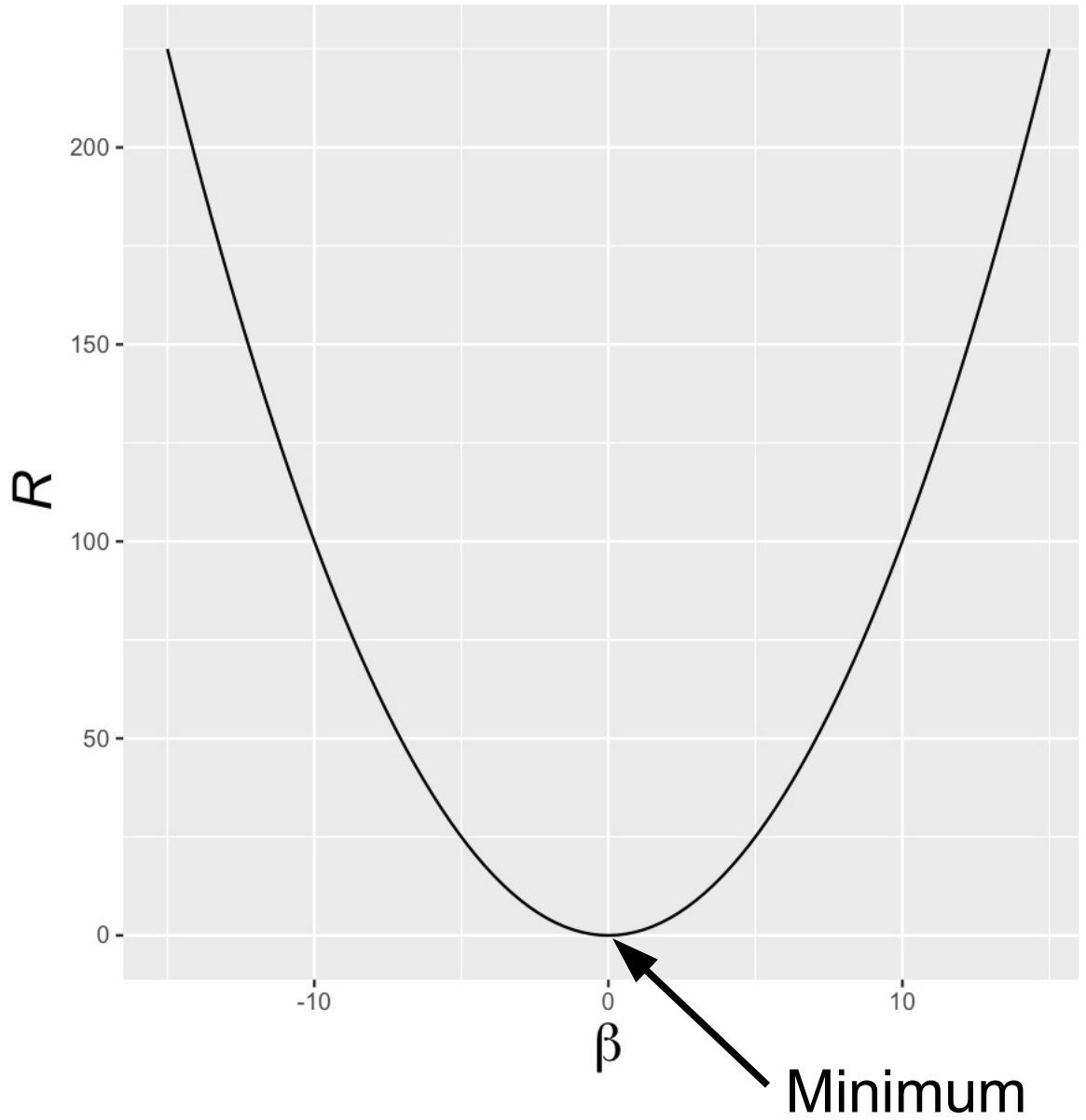
$$\begin{bmatrix} \beta_0^{\text{new}} \\ \beta_1^{\text{new}} \end{bmatrix} = \begin{bmatrix} \beta_0^{\text{old}} \\ \beta_1^{\text{old}} \end{bmatrix} - \eta \begin{bmatrix} \frac{\partial}{\partial \beta_0} (y_i - \beta_0 + \beta_1 x_i)^2 \\ \frac{\partial}{\partial \beta_1} (y_i - \beta_0 + \beta_1 x_i)^2 \end{bmatrix}$$

$$= \begin{bmatrix} \beta_0^{\text{old}} \\ \beta_1^{\text{old}} \end{bmatrix} - \eta \begin{bmatrix} 2(y_i - \beta_0 + \beta_1 x_i) \\ 2x_i(y_i - \beta_0 + \beta_1 x_i) \end{bmatrix}$$

Straight lines

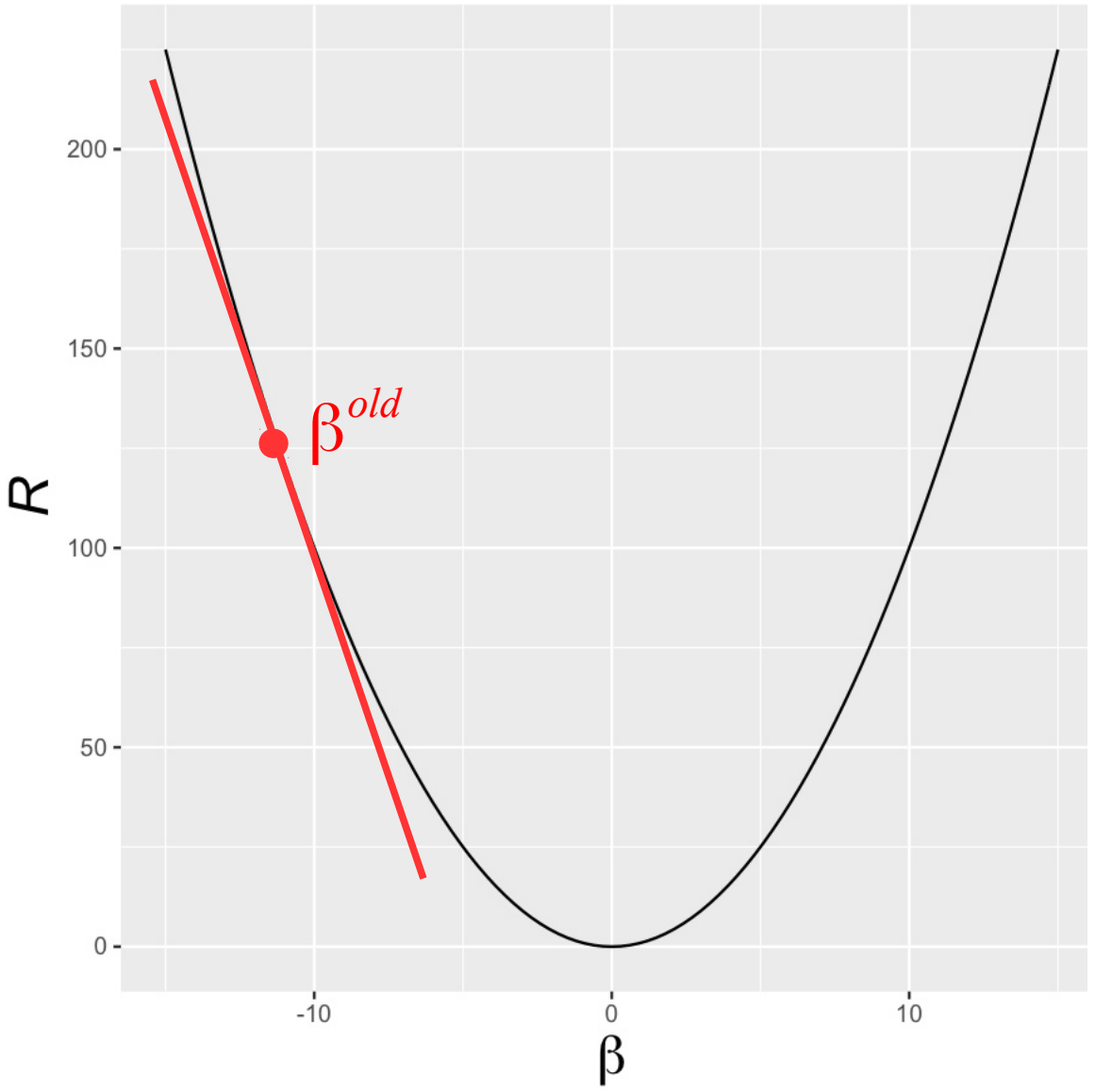
Stochastic gradient descent (SGD)

$$\theta^{\text{new}} = \theta^{\text{old}} - \eta \nabla_{\theta} R$$



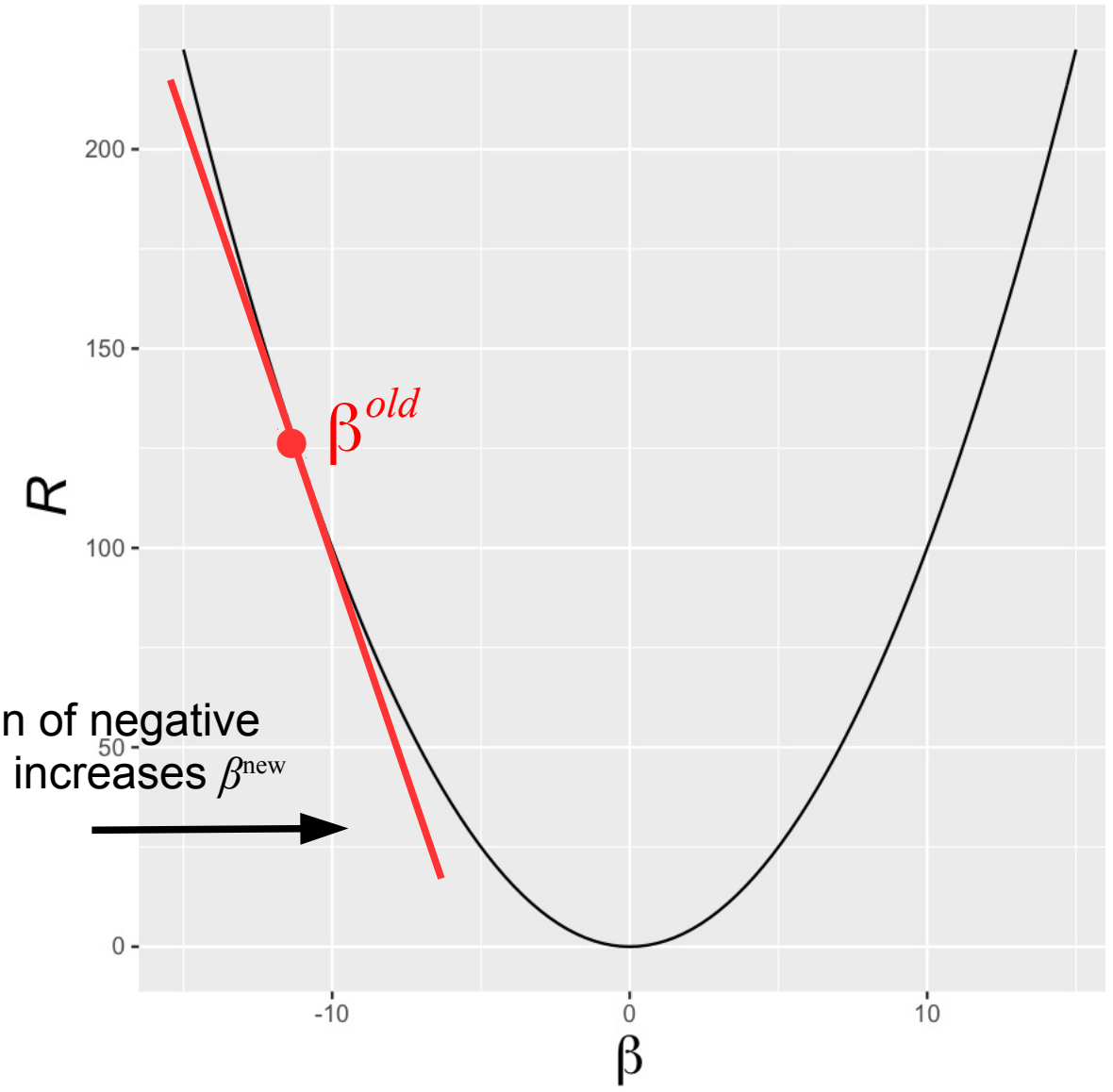
Stochastic gradient descent (SGD)

$$\theta^{\text{new}} = \theta^{\text{old}} - \eta \nabla_\theta R$$



Stochastic gradient descent (SGD)

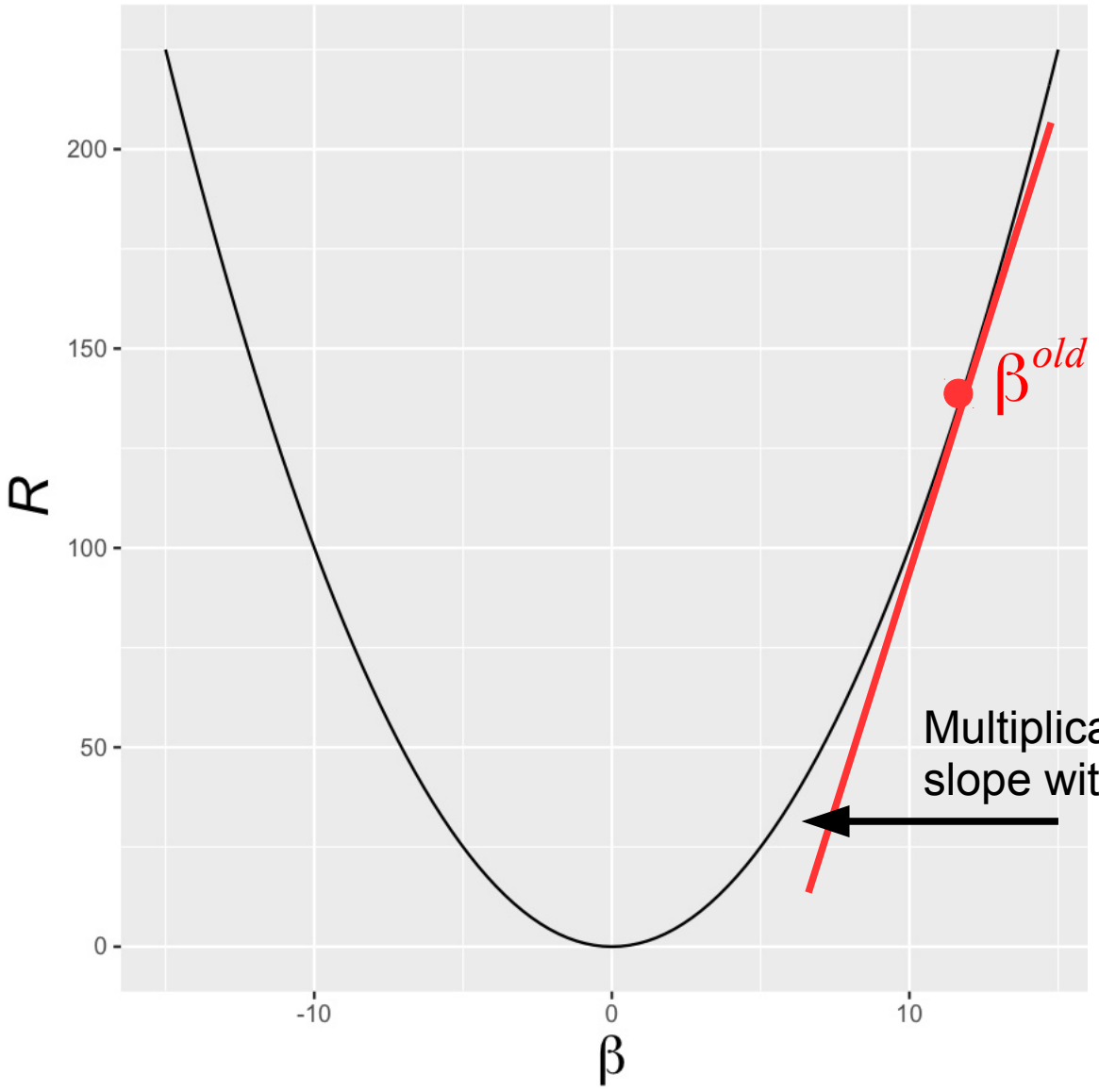
$$\theta^{\text{new}} = \theta^{\text{old}} - \eta \nabla_\theta R$$



Multiplication of negative slope with η increases β^{new}

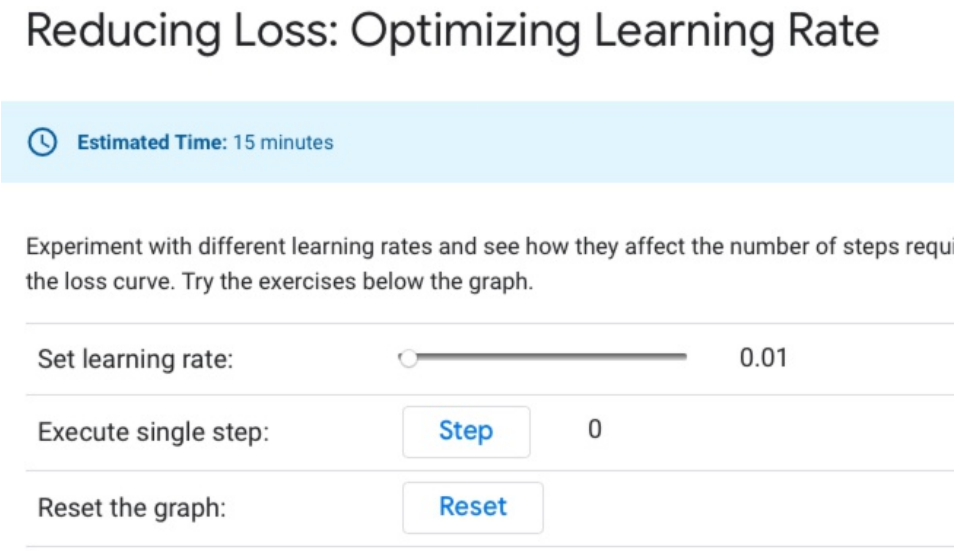
Stochastic gradient descent (SGD)

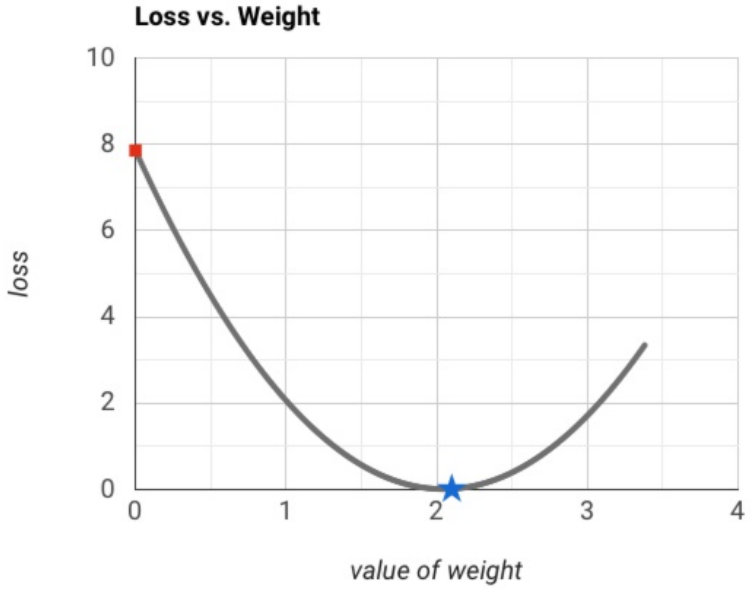
$$\theta^{new} = \theta^{old} - \eta \nabla_\theta R$$



Learning rate η

- If too high: May jump over minima
- If too low: May get trapped in local minima and takes long
- Algorithms available (e.g. adaptive learning rate)

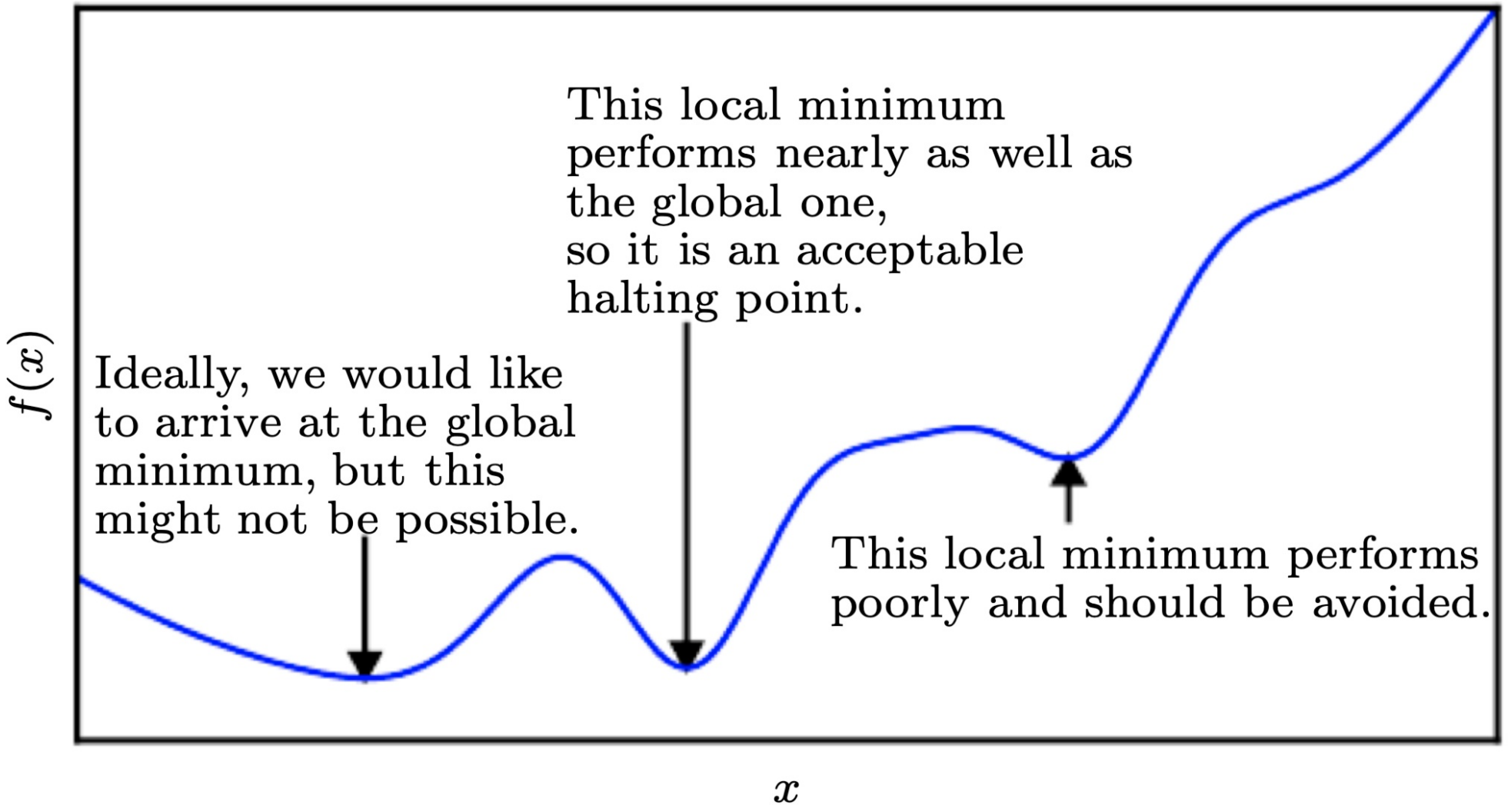




https://developers.google.com/machine-learning/crash-course/fitter/graph

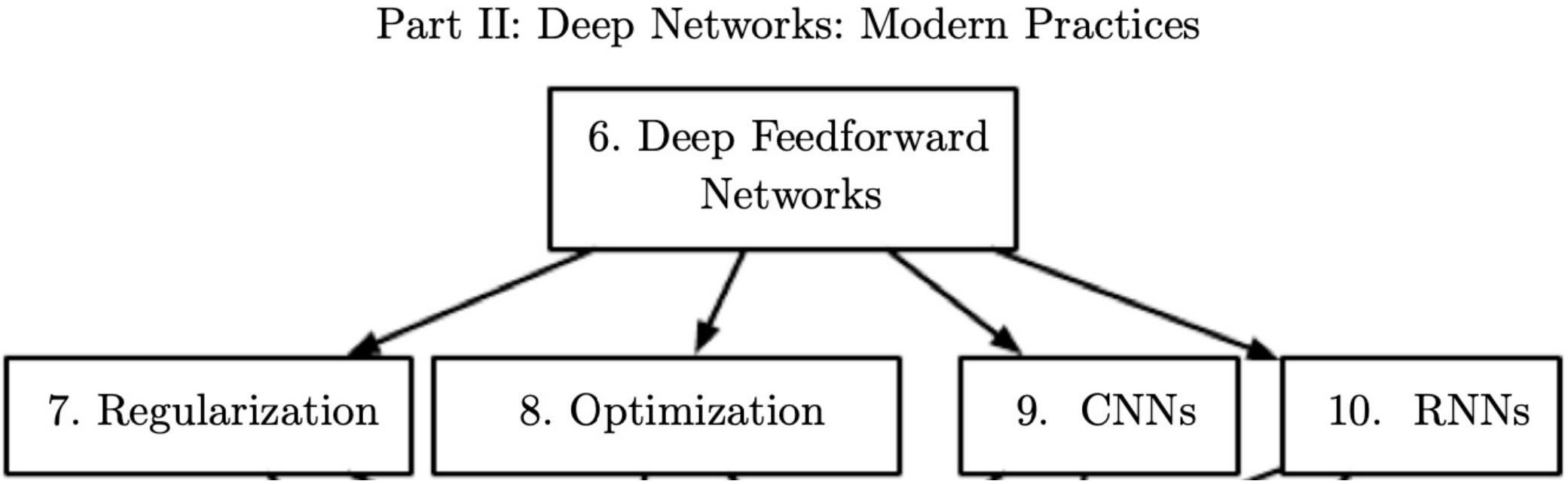
Finding optima

Global optimum may result in overfitting (→ consider regularization)



Goodfellow et al. 2016: 85

Modern practice in Deep Learning

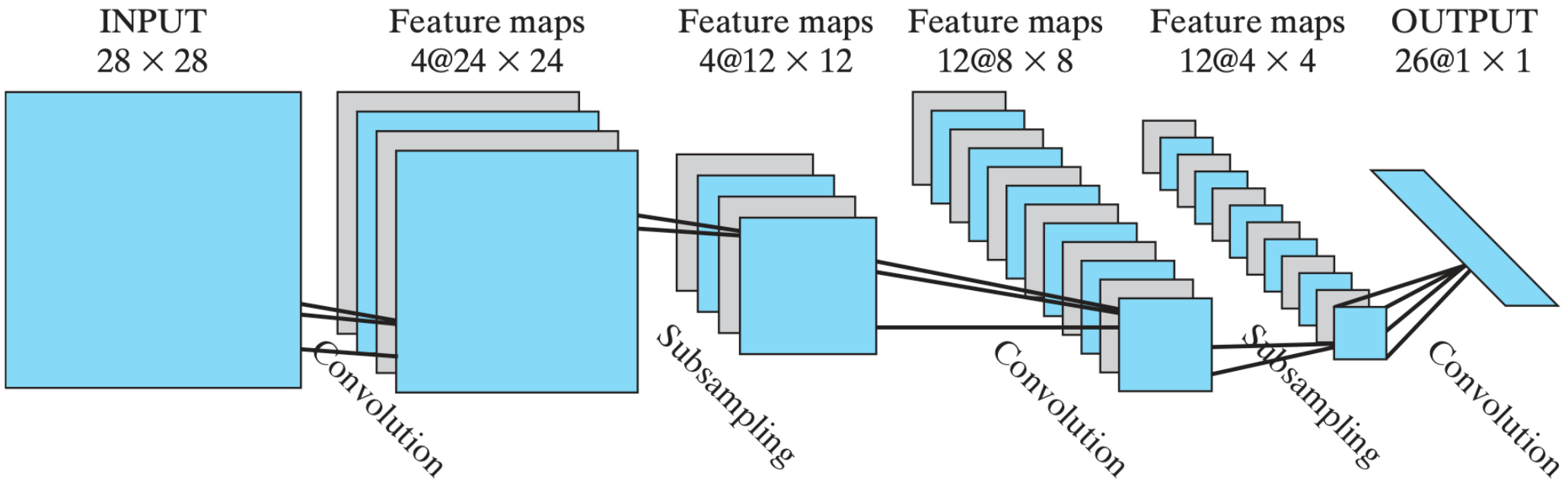


- Regularization → LASSO
- Optimization: Different algorithms

Convolutional Neural Networks (CNNs)

- Tailored towards data with spatial relationships such as images (but also audio data)
- Higher performance than other methods

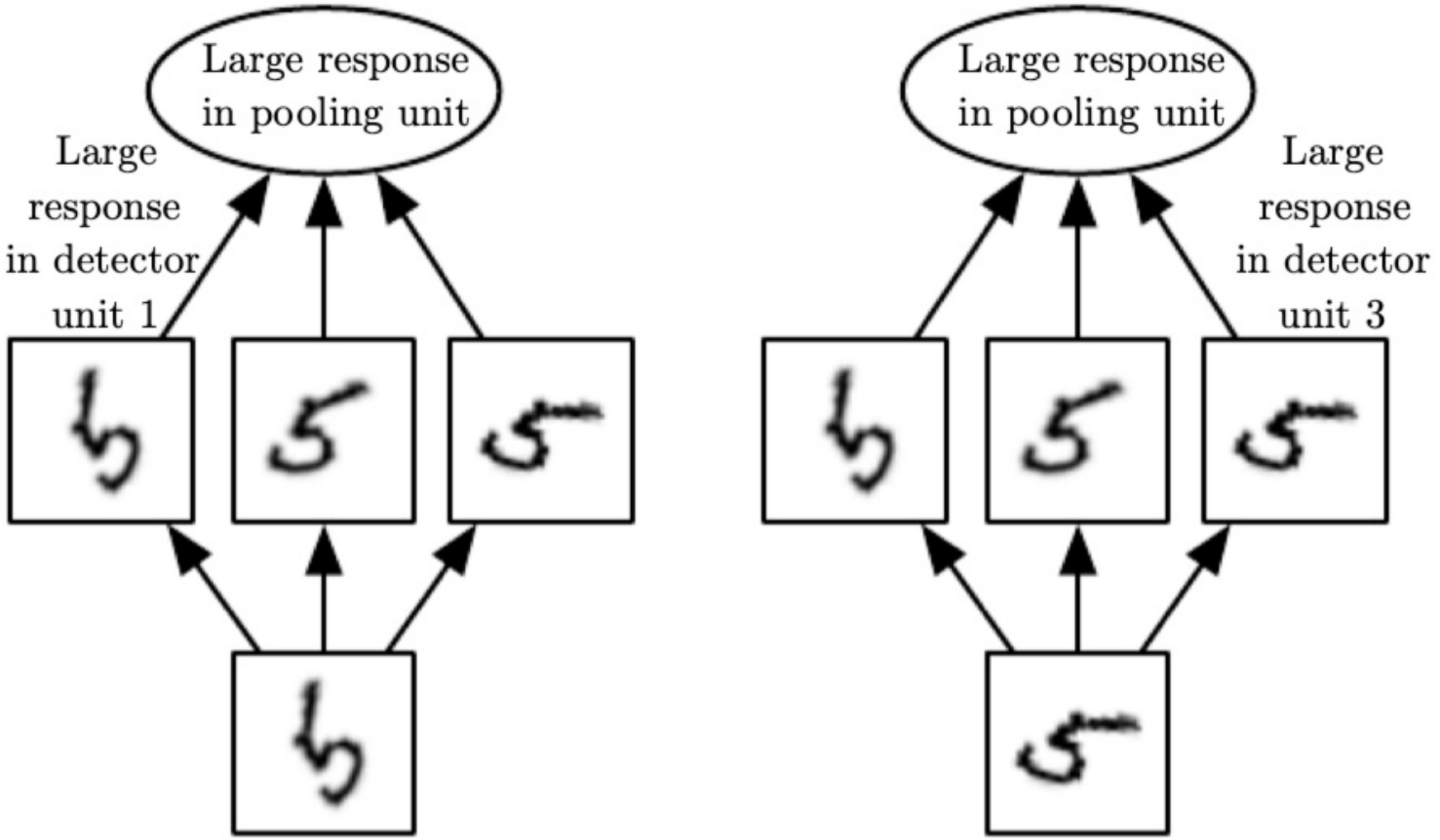
Example of 2D Image processing (Handwriting recognition)



Haykin 2009: 202

Convolutional Neural Networks (CNNs)

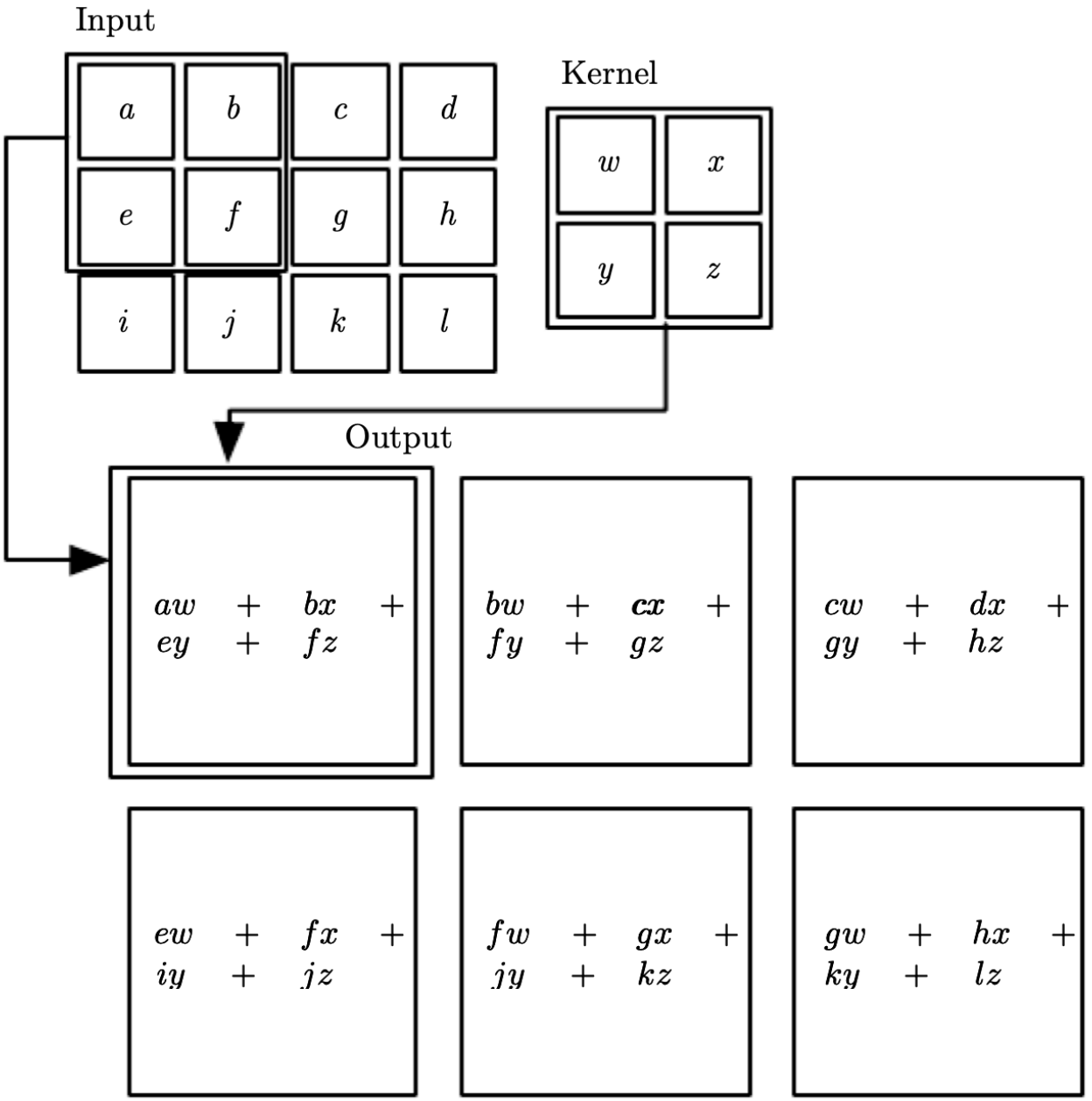
Multiple feature maps learned with separate parameters can lead to invariance to transformations of the input



Goodfellow et al. 2016: 344

Convolutional Neural Networks (CNNs)

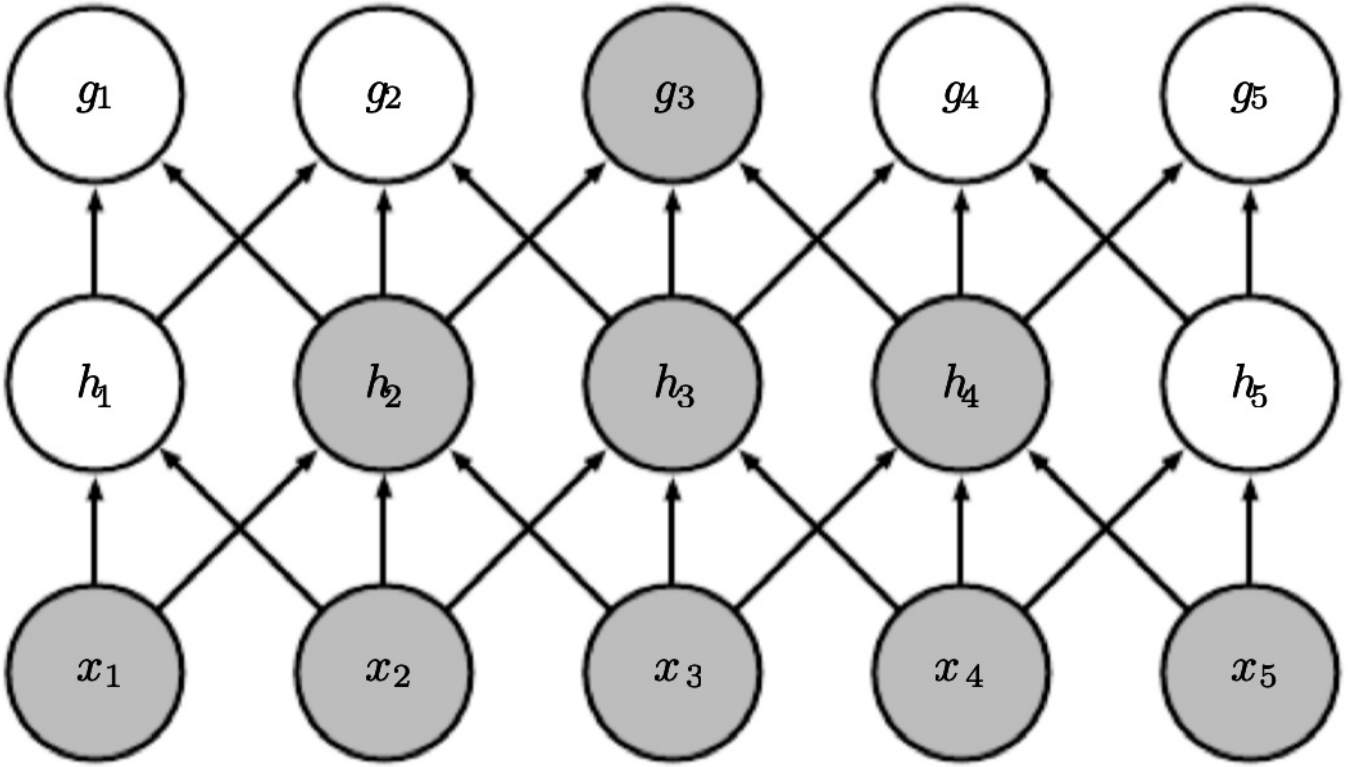
Example for Convolution



Goodfellow et al. 2016: 334

Convolutional Neural Networks (CNNs)

Kernel translates to sparsity in network (not fully connected)

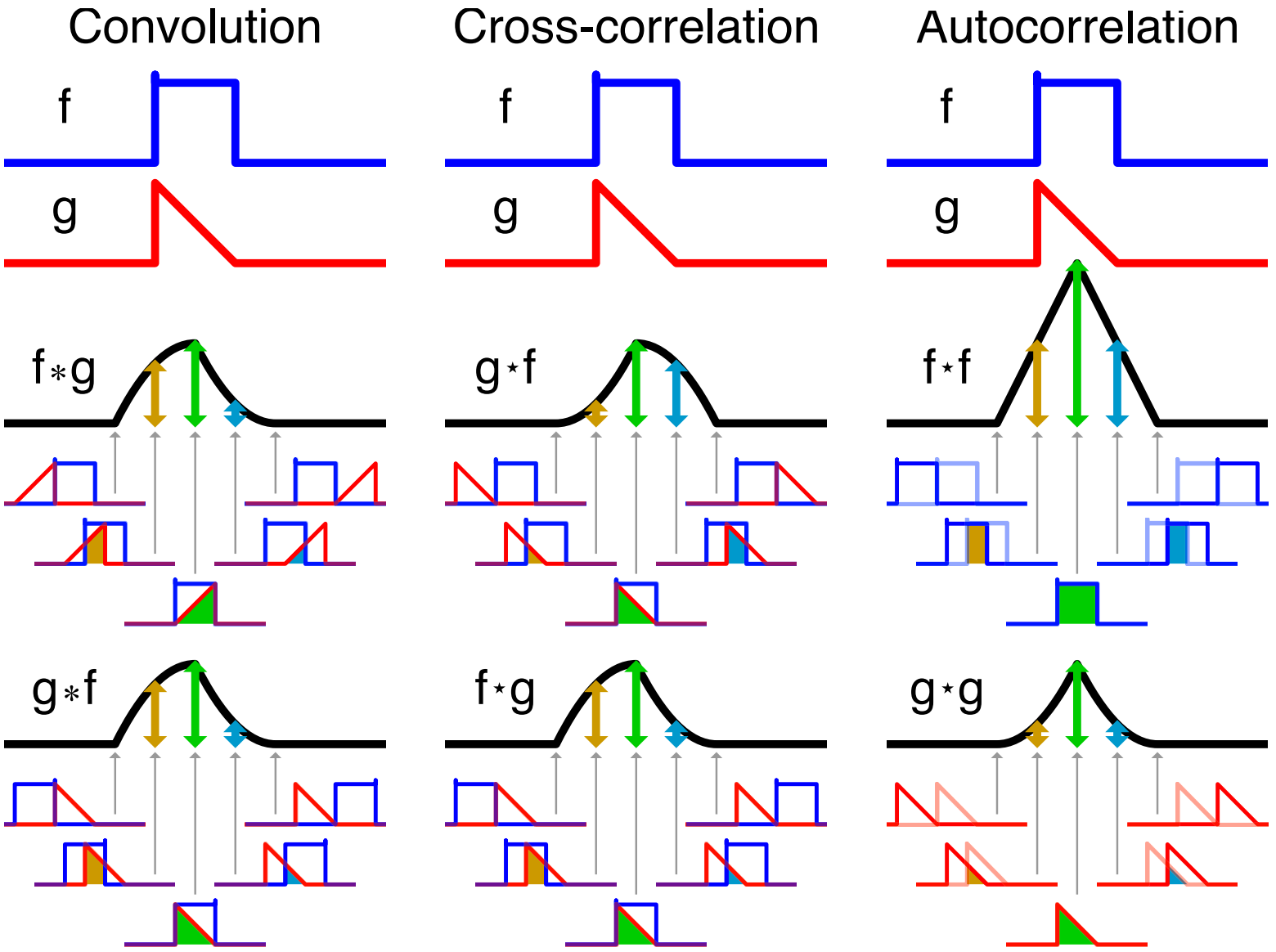


Goodfellow et al. 2016: 337

CNN exercise with Stefan Kunz

Mathematics: Convolution

$$(x * w)(t) = \int_0^t x(\tau) w(t-\tau)\, d\tau$$

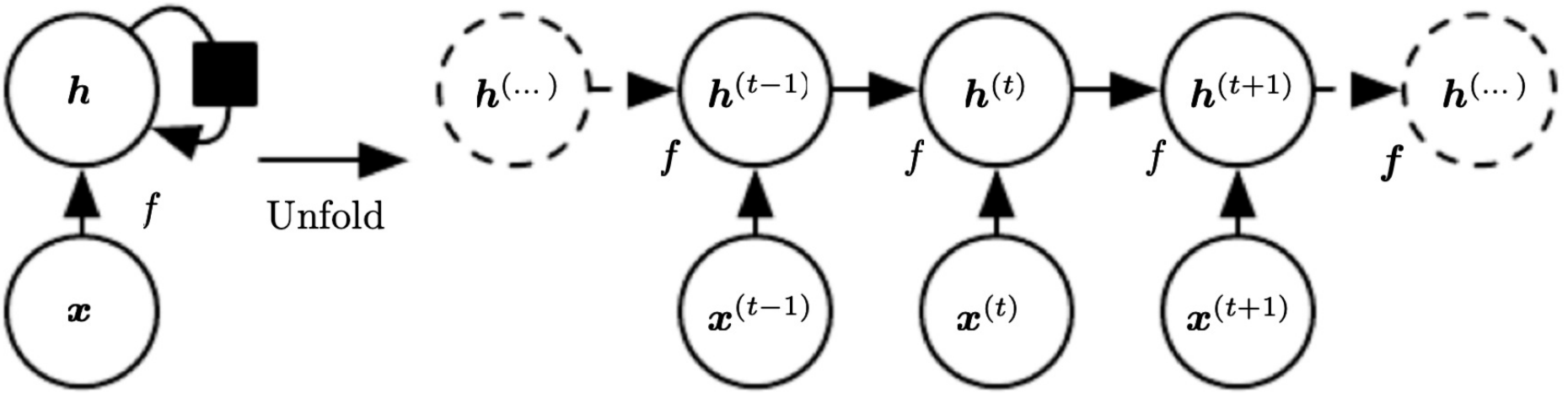


https://commons.wikimedia.org/w/index.php?curid=20206883

Recurrent neural networks

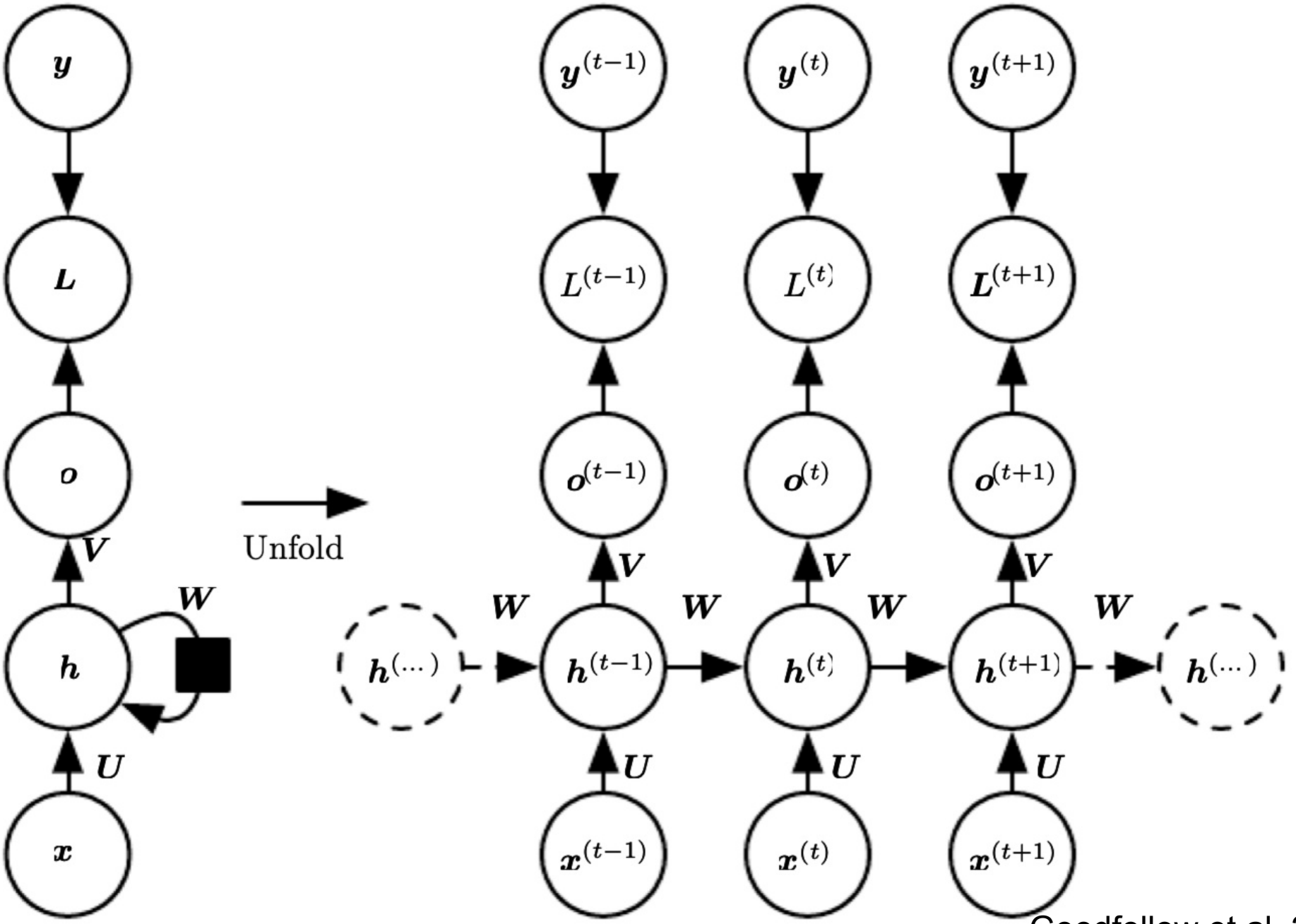
- Tailored towards sequential data (i.e. written text, speech, connected handwriting)
- Relies on parameter sharing like CNNs, not via convolution kernel but via connected functions

$$h^{(t)} = f(h^{(t-1)}, x^{(t)}, \theta)$$



Goodfellow et al. 2016: 376

Recurrent neural networks



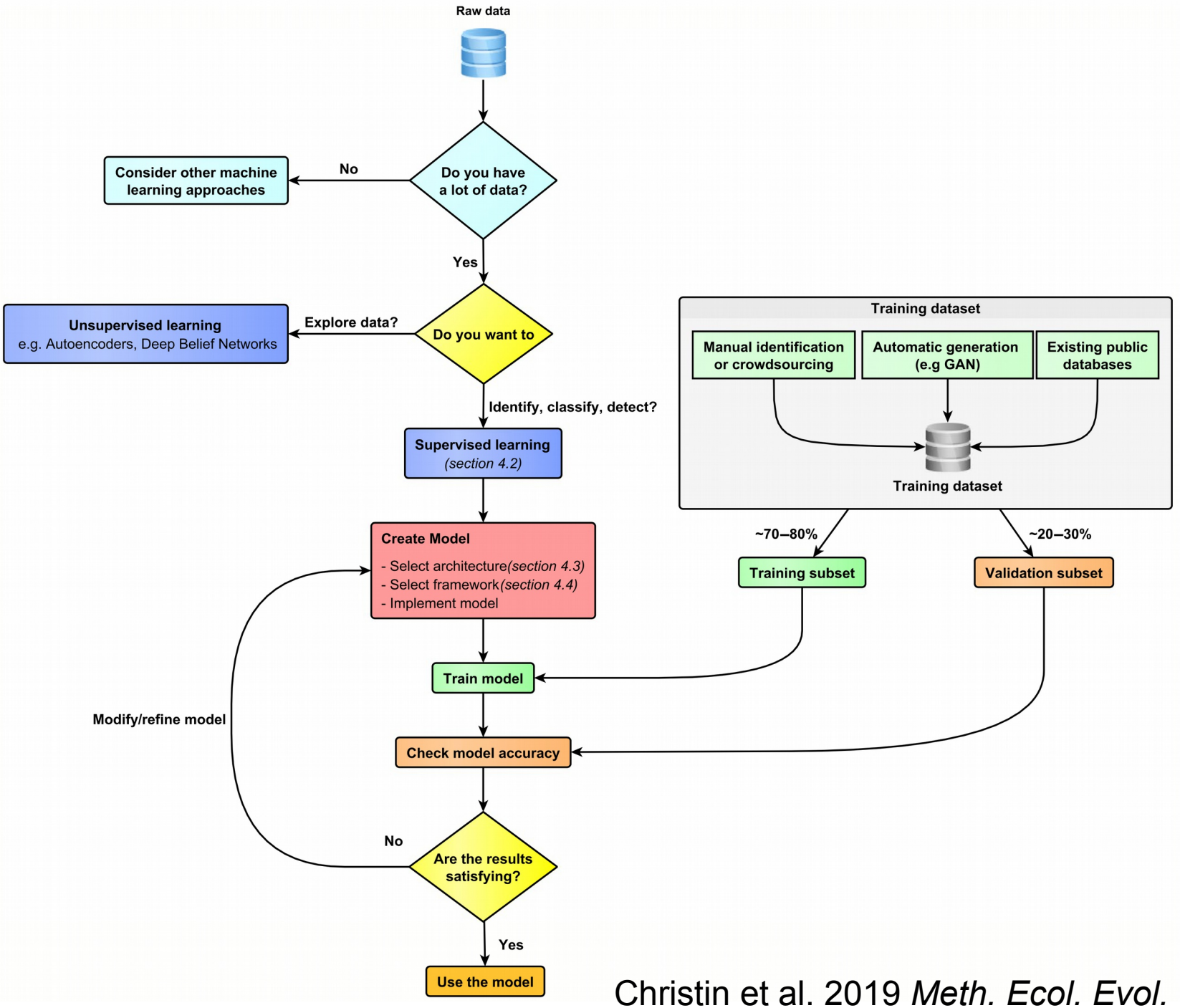
Goodfellow et al. 2016: 378

Critical appraisal

- High performance for prediction
- No strong assumptions
- Ideal for large and complex data sets

- Not useful for (mechanistic) understanding
- Learning limited to extracting information from huge amount of training examples (no abstract learning via definition)
- Risk of overfitting
- Lack of uncertainty estimates and statistical model → no theory for inference, diagnostics or model selection

Flowchart for using DL



Christin et al. 2019 *Meth. Ecol. Evol.* 10: 1632-1644