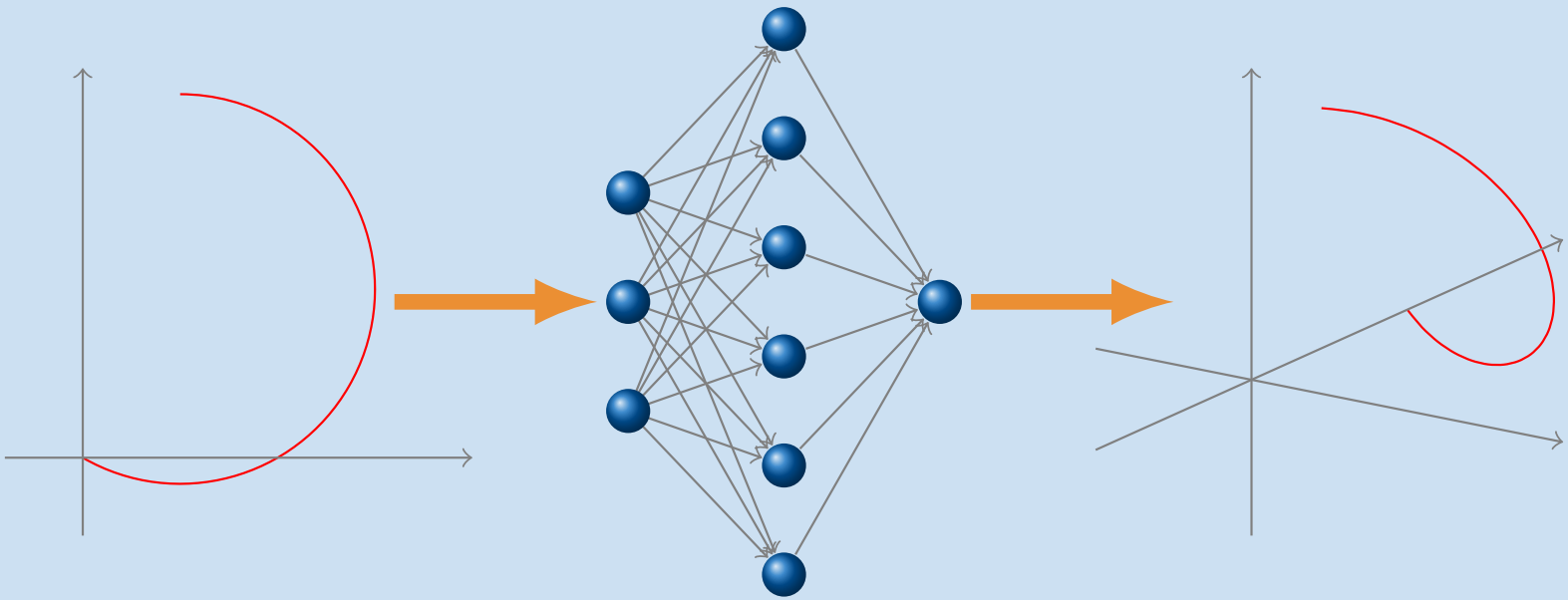
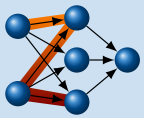
Neural Networks in the Belle II Track Trigger

Ringberg Young Scientist Workshop 2016

June 8, 2016

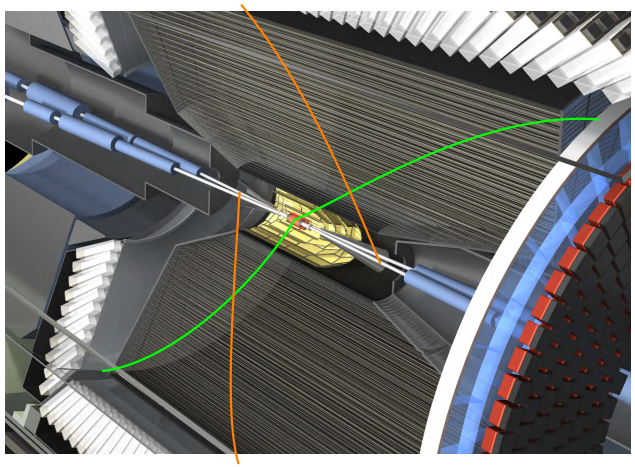




Neuro team
S. Bähr, C. Kiesling, **S. Neuhaus**, S. Skambraks

The Neurotrigger

estimate z-vertex of single tracks to reject machine background in Belle II

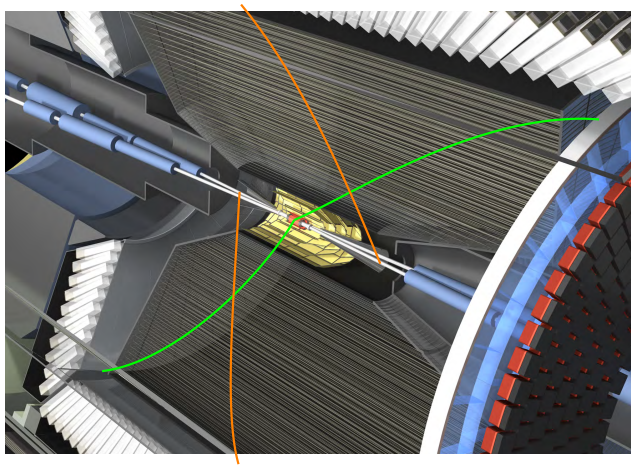


Why neural networks?

- fast ($< 1\,\mu s$), parallel, deterministic runtime
- nonlinear, noise robust
- ...

The Neurotrigger

estimate z-vertex of single tracks to reject machine background in Belle II



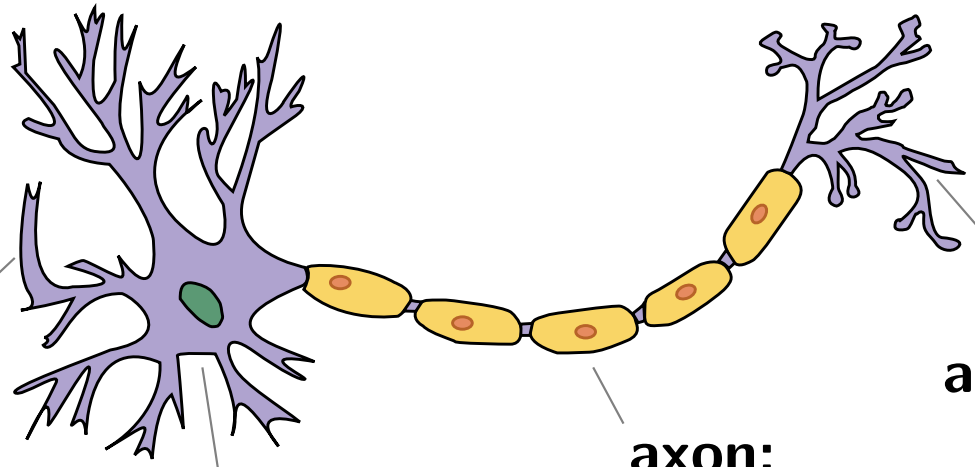
Why neural networks?

- fast ($< 1\,\mu s$), parallel, deterministic runtime
- nonlinear, noise robust
- ...

How does it work?

- multi layer perceptron
- function approximation
- classification
- training methods
- application in the neurotrigger

Biological neuron vs. perceptron

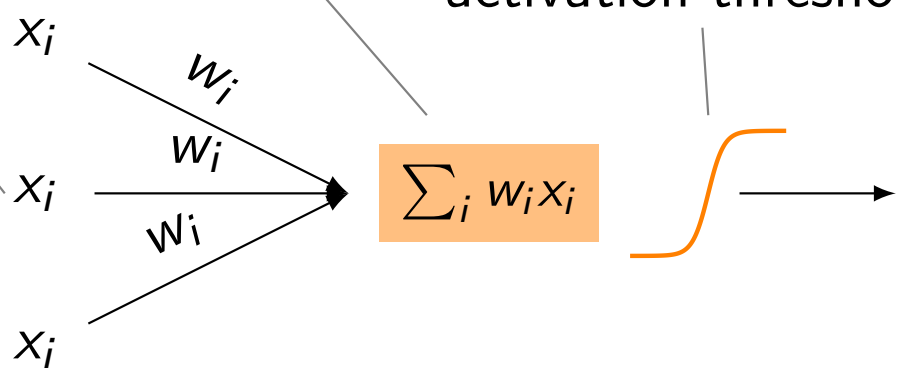


dendrites: weighted input signals

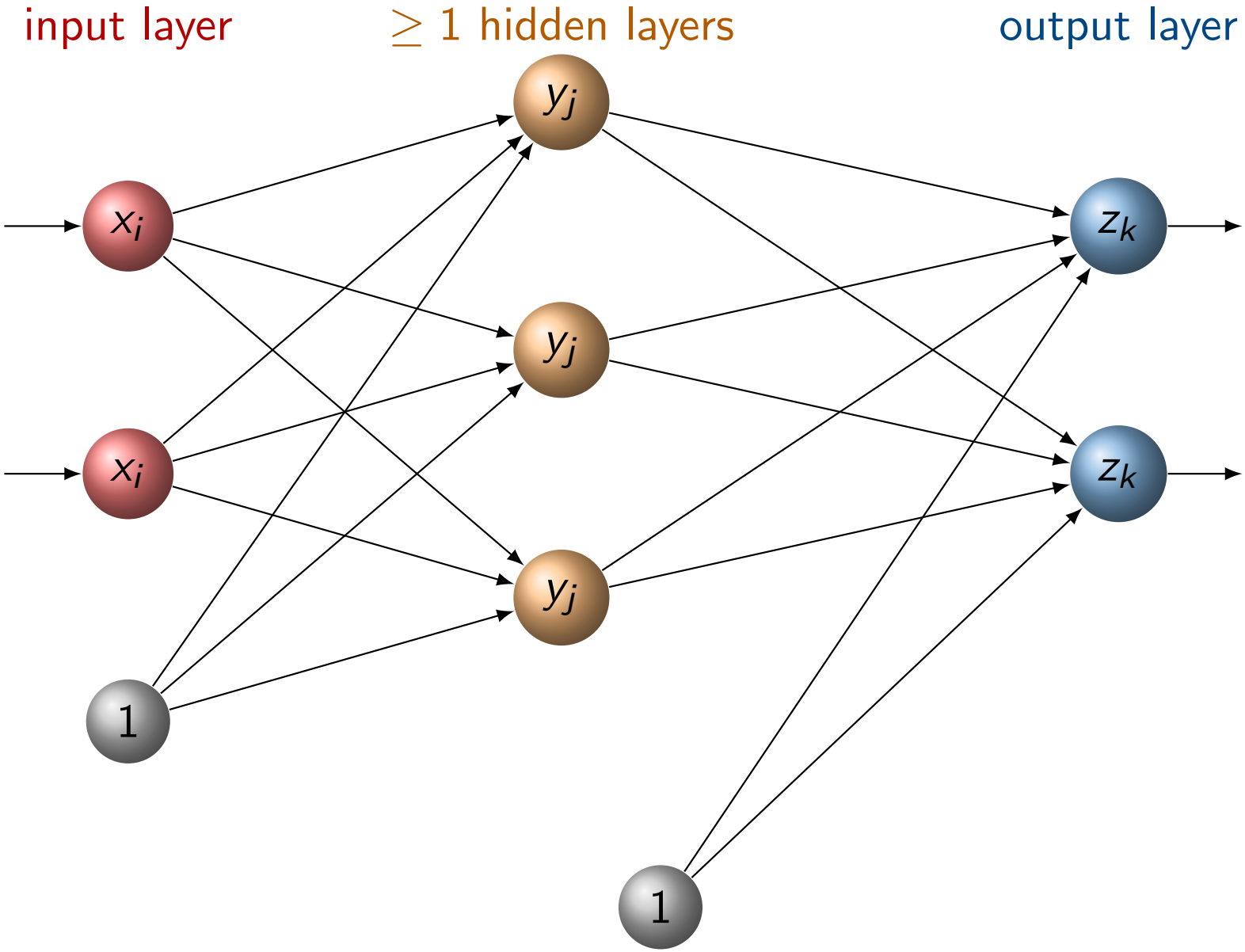
cell body: sum inputs

axon: pulse when signal sum reaches activation threshold

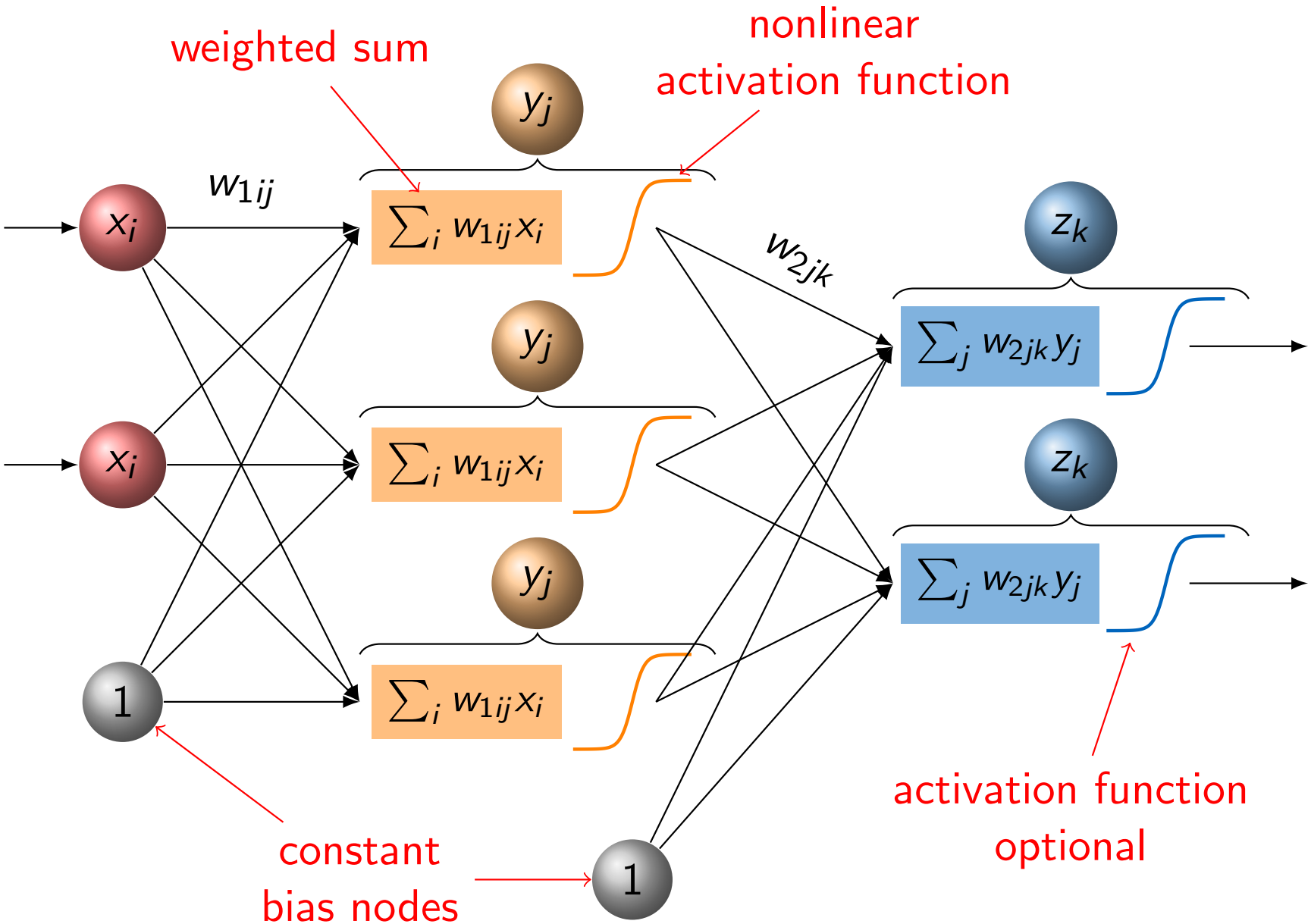
axon terminal: output signal



Multi Layer Perceptron (MLP)



Multi Layer Perceptron (MLP)

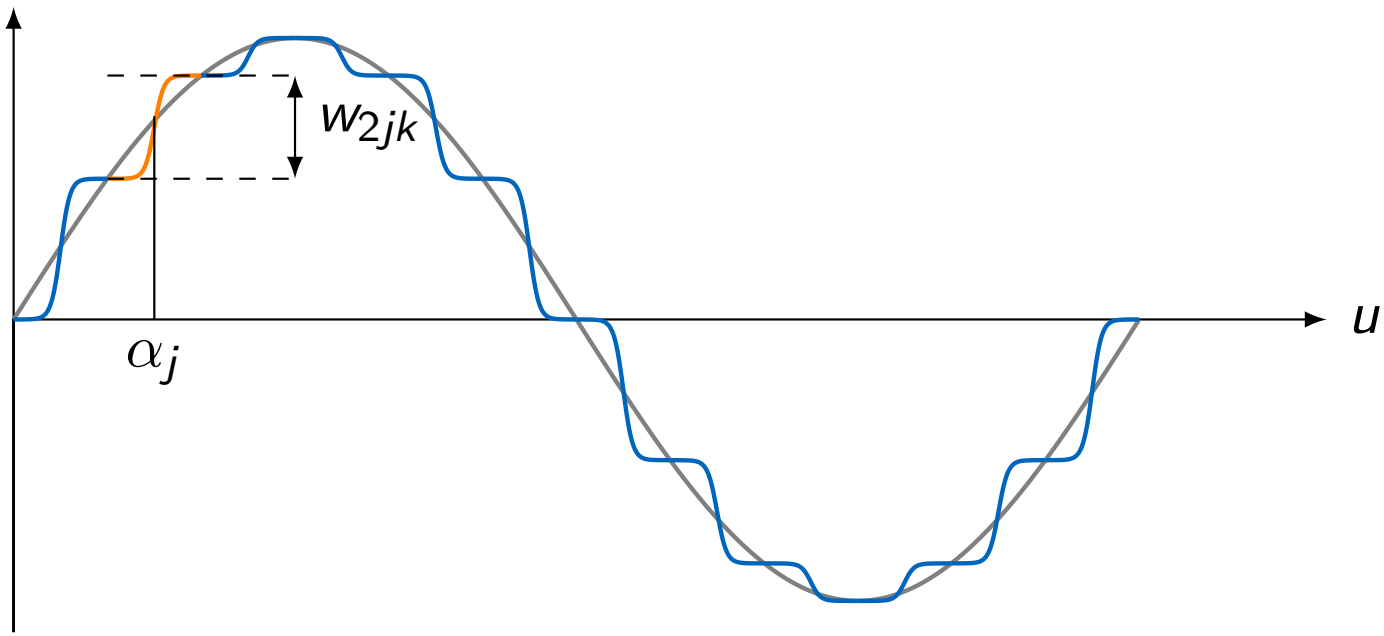


General function approximation

Fourier series: $f(\vec{x}) = \sum_{\vec{k}} c_{\vec{k}} \cdot \exp(\underbrace{2\pi i \vec{k} \cdot \vec{x}}_{u(\vec{x})})$

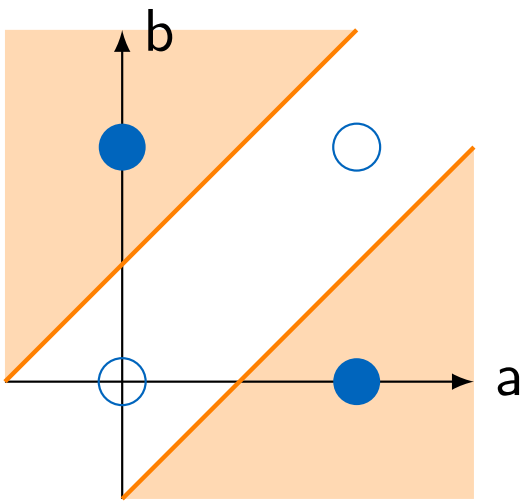
Hidden node: $\tanh\left(\sum_i w_{1ij} x_i\right) = \tanh\left(\beta_j(u(\vec{x}) - \alpha_j)\right)$

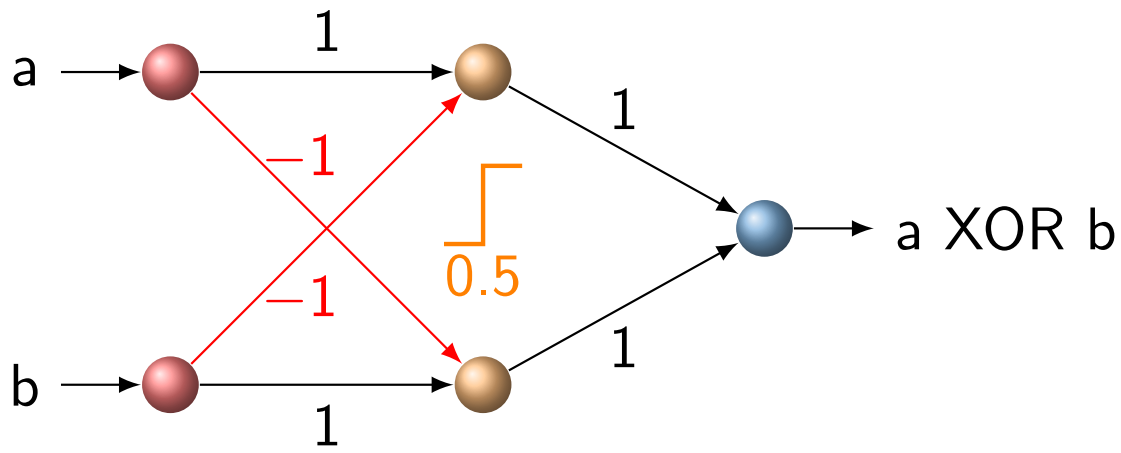
Output node: $\sum_j w_{2jk} \cdot \tanh(\beta_j(u(\vec{x}) - \alpha_j)) \approx \sin(u(\vec{x}))$

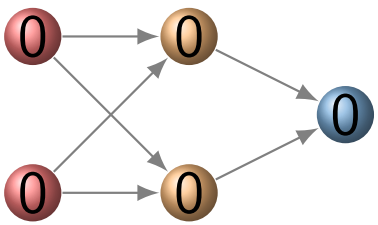


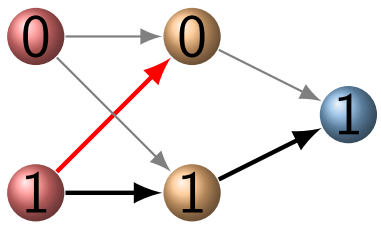
Classification: XOR

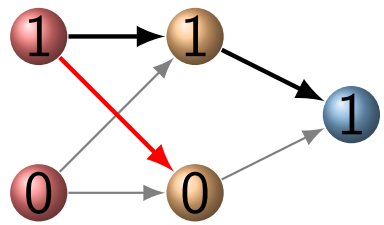
need at least 2 separating lines = hidden nodes

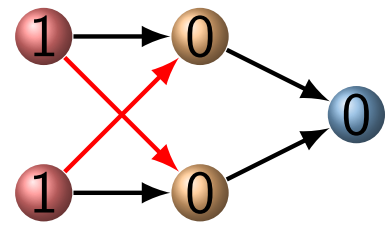










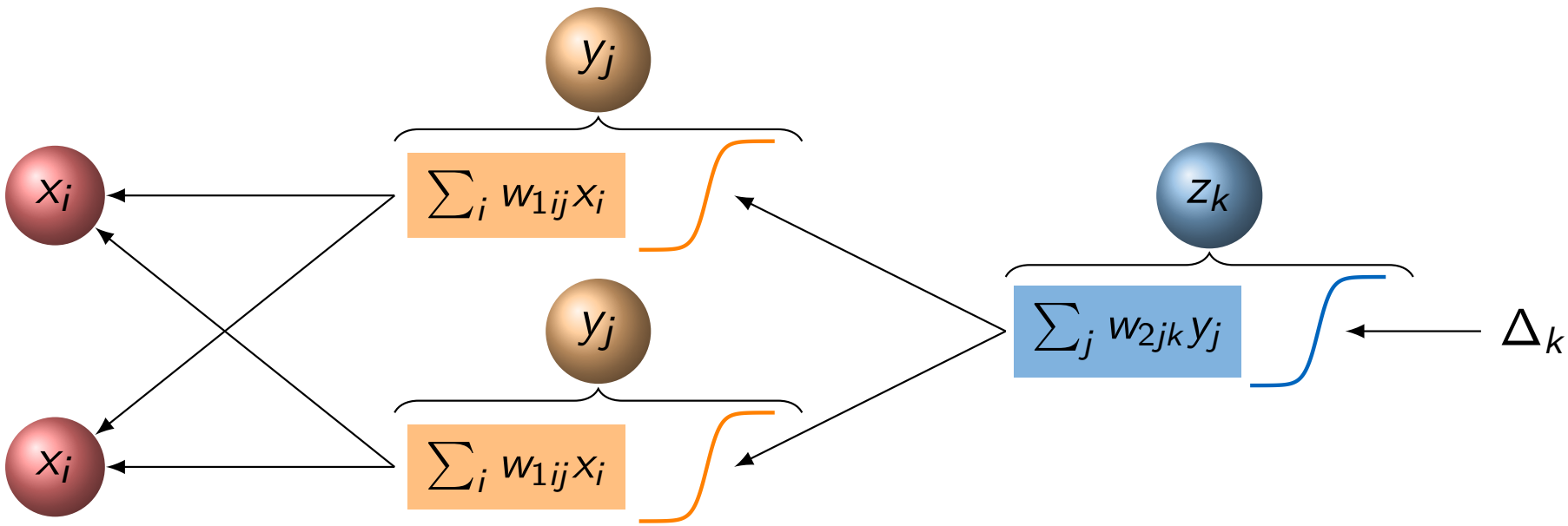


Training with backpropagation

training samples $(\vec{x}^s, \vec{t}^s)$

$\rightarrow$ minimize **cost function** $E = \sum_{k,s}(z_k(\vec{x}^s) - t_k^s)^2 = \sum_{k,s}(\Delta_k^s)^2$

weight updates: $\Delta w = -\frac{\partial E}{\partial w} \cdot \delta$, ($\delta$: learning rate)

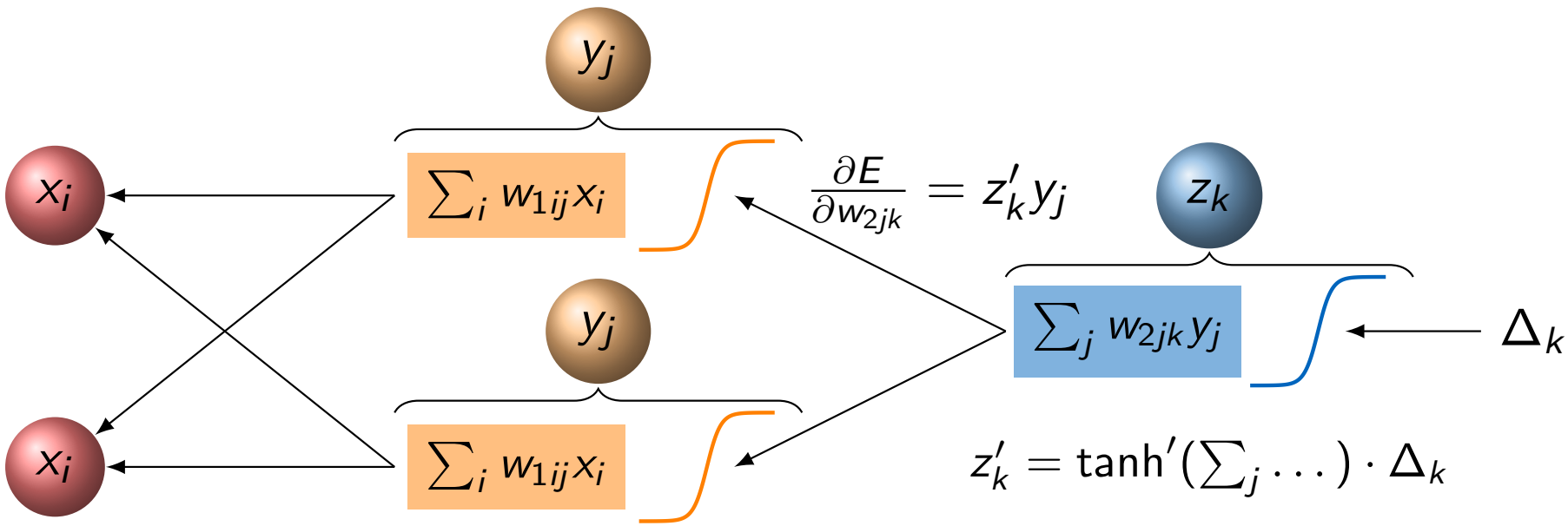


Training with backpropagation

training samples $(\vec{x}^s, \vec{t}^s)$

$\rightarrow$ minimize **cost function** $E = \sum_{k,s}(z_k(\vec{x}^s) - t_k^s)^2 = \sum_{k,s}(\Delta_k^s)^2$

weight updates: $\Delta w = -\frac{\partial E}{\partial w} \cdot \delta$, ($\delta$: learning rate)

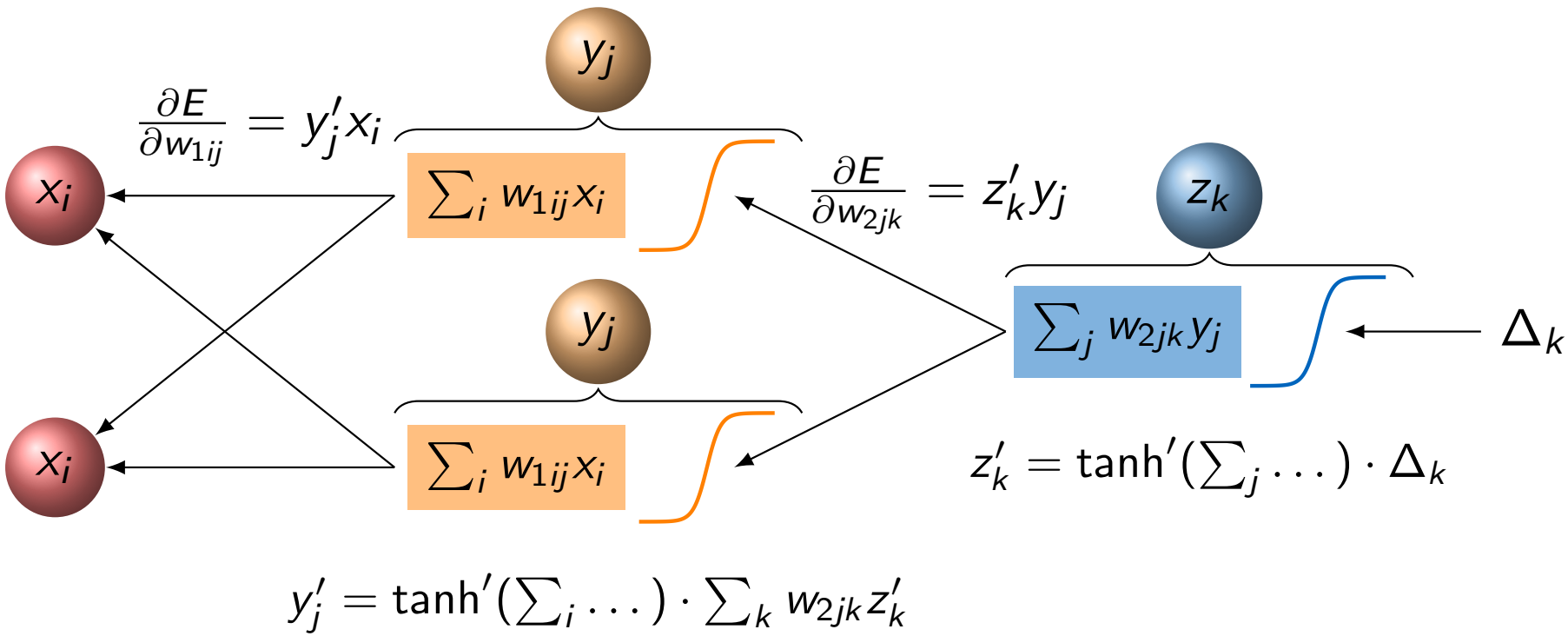


Training with backpropagation

training samples $(\vec{x}^s, \vec{t}^s)$

$\rightarrow$ minimize **cost function** $E = \sum_{k,s}(z_k(\vec{x}^s) - t_k^s)^2 = \sum_{k,s}(\Delta_k^s)^2$

weight updates: $\Delta w = -\frac{\partial E}{\partial w} \cdot \delta$, ($\delta$: learning rate)

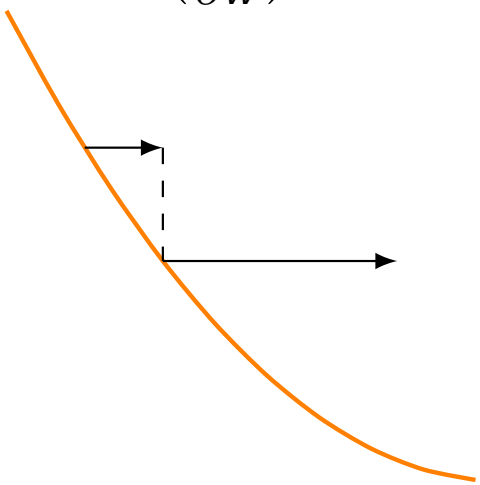


RPROP: flexible learning rate

weight updates: $\Delta w = -\text{sign}\left(\frac{\partial E}{\partial w}\right) \cdot \delta_{\text{epoch}}(w)$

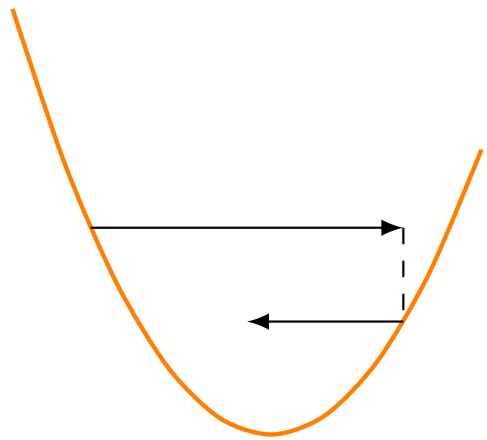
- depends only on sign of cost function derivative
- individual learning rate for each weight
- learning rate adjusted during training

$\left(\frac{\partial E}{\partial w}\right)^{epoch} \cdot \left(\frac{\partial E}{\partial w}\right)^{epoch-1} > 0$:



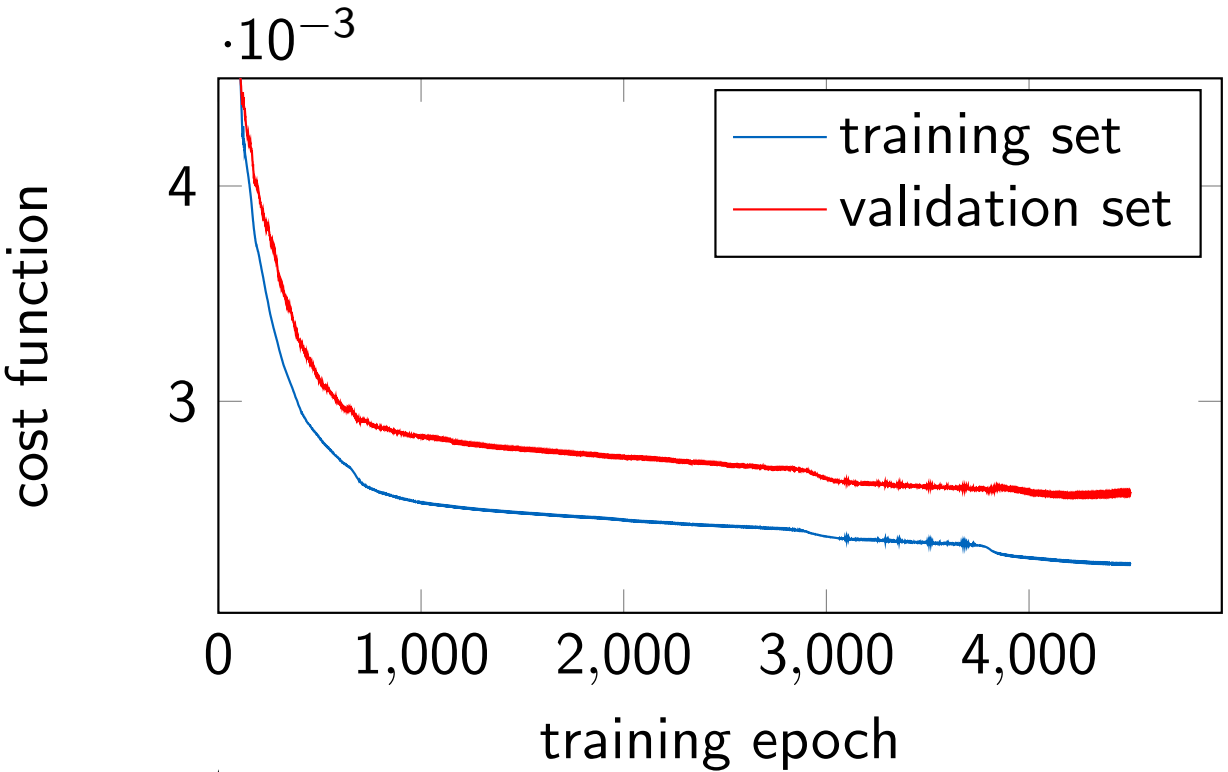
$\rightarrow$ increase $\delta(w)$

$\left(\frac{\partial E}{\partial w}\right)^{epoch} \cdot \left(\frac{\partial E}{\partial w}\right)^{epoch-1} < 0$:

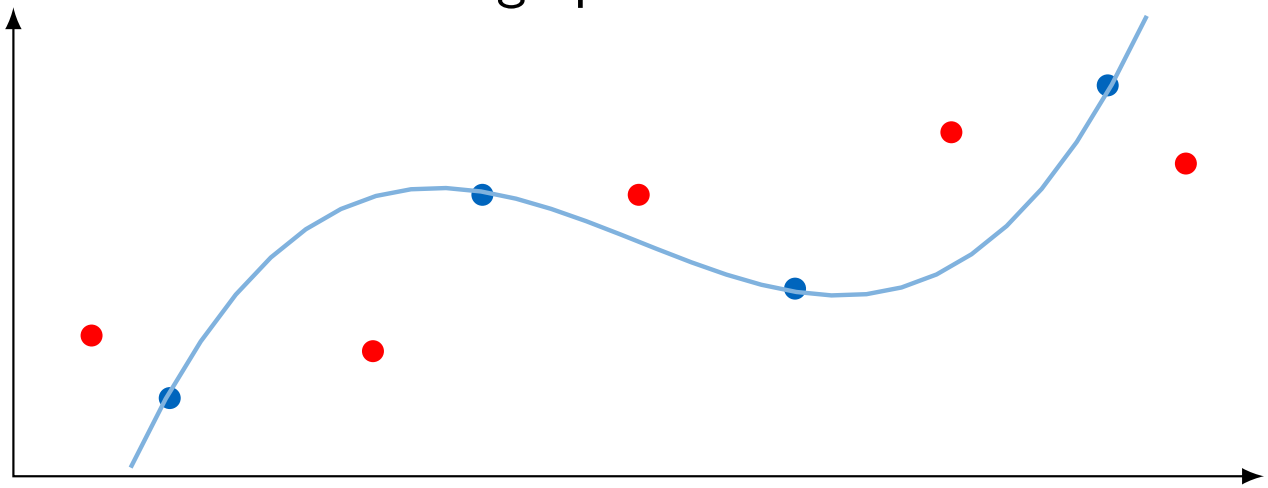


$\rightarrow$ decrease $\delta(w)$

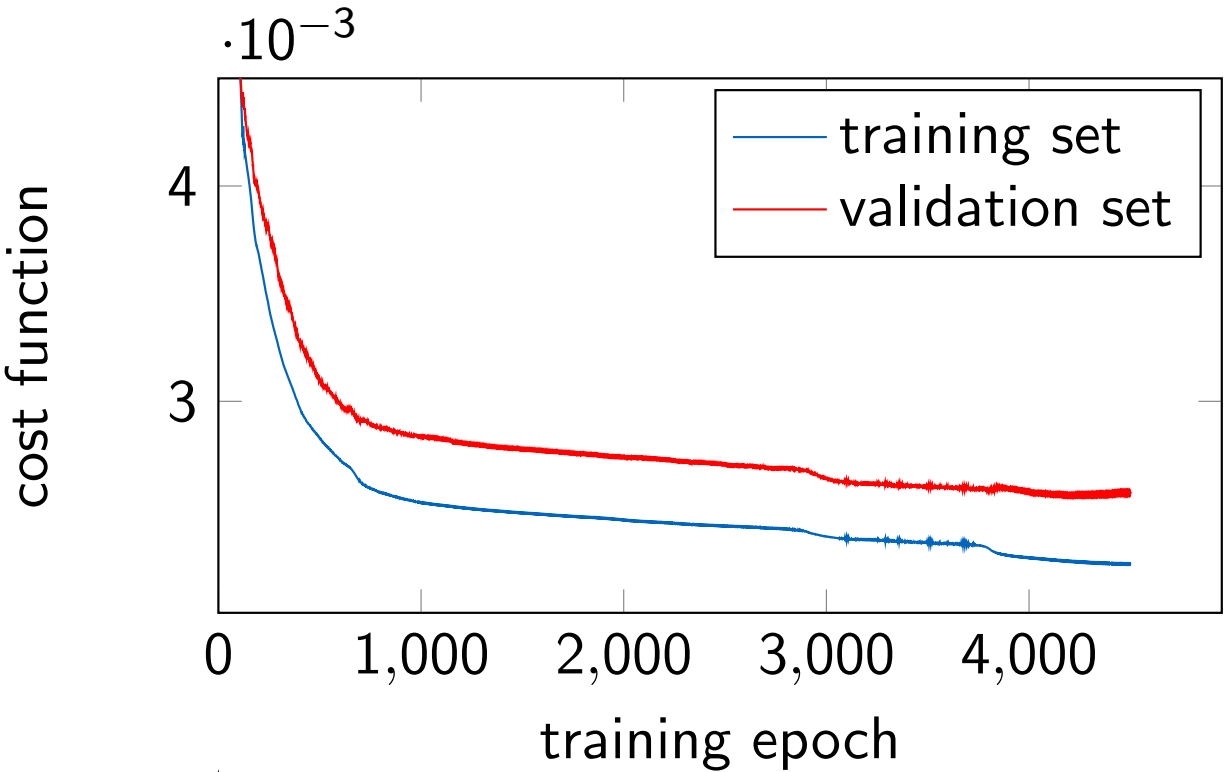
Overtraining



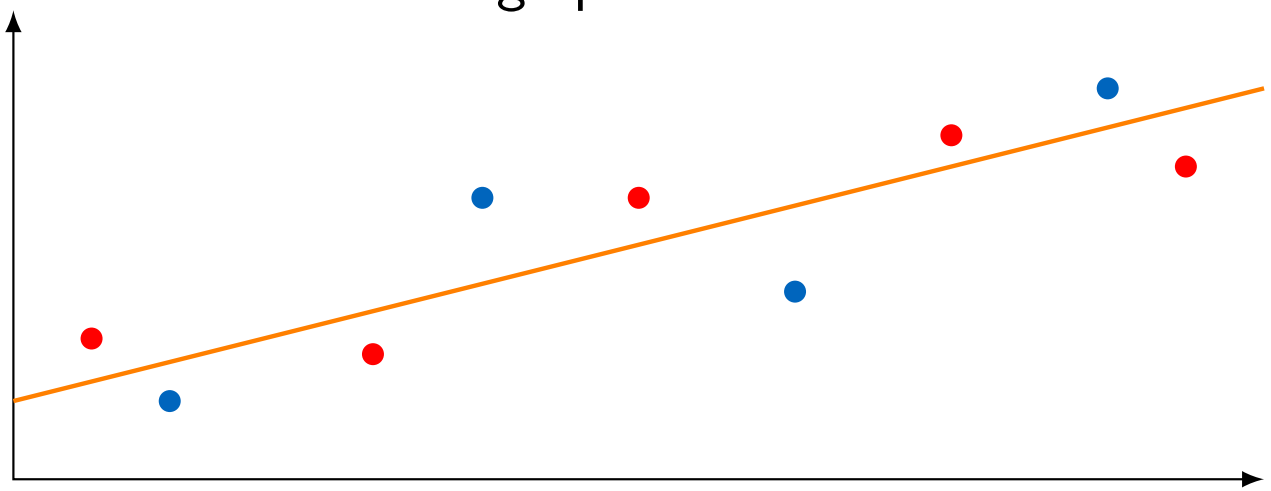
independent
validation set
to avoid
"learning by heart"



Overtraining

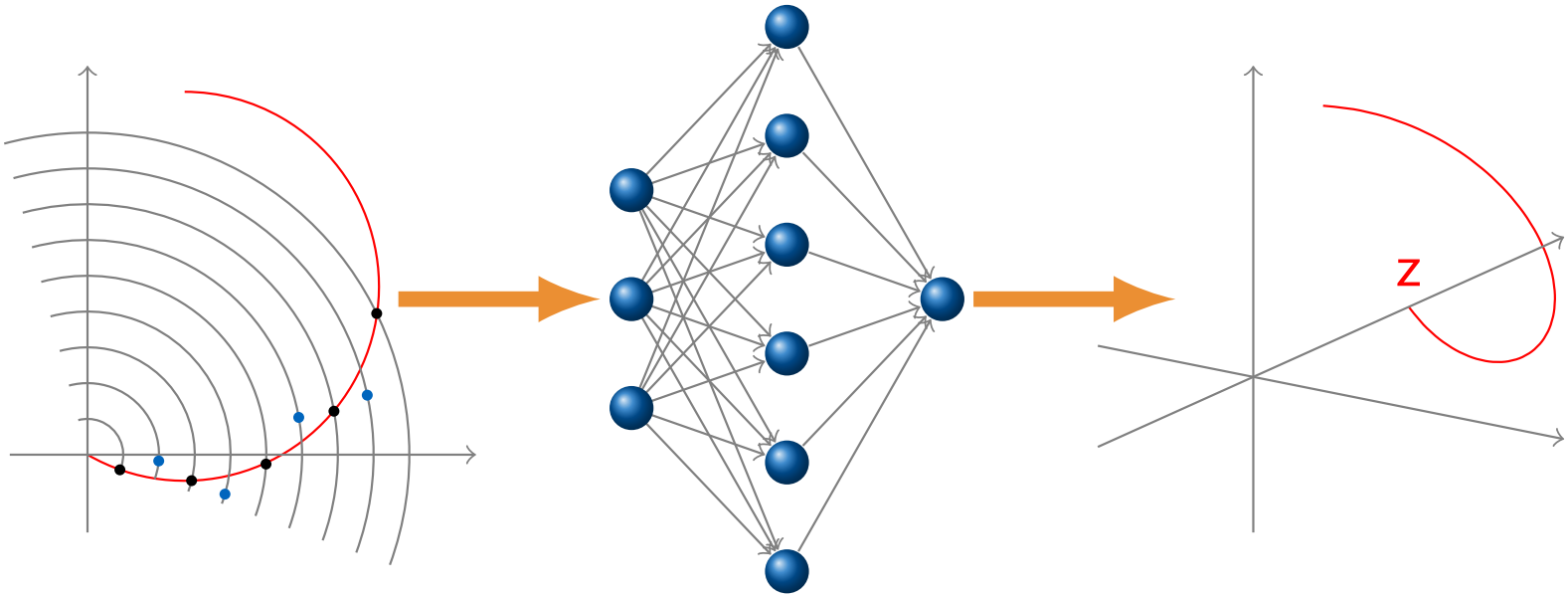


independent
validation set
to avoid
"learning by heart"

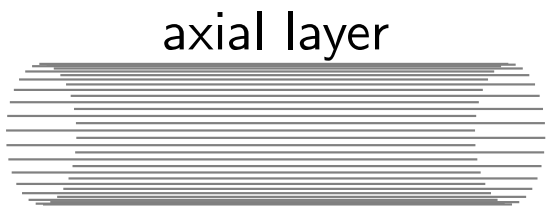


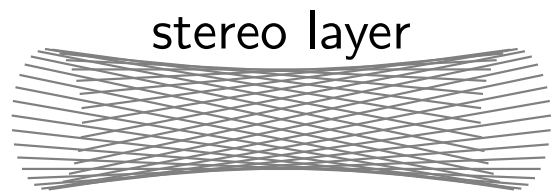
MLPs in the neurotrigger

Task: 2D track (circle) → 3D track (helix)

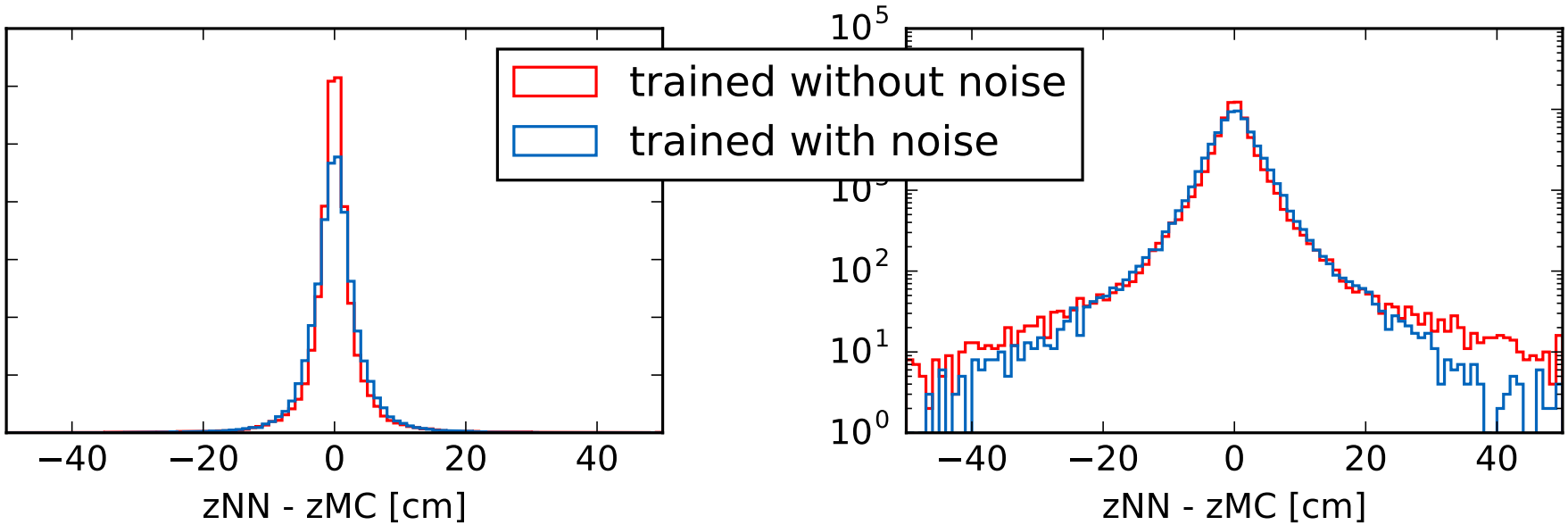


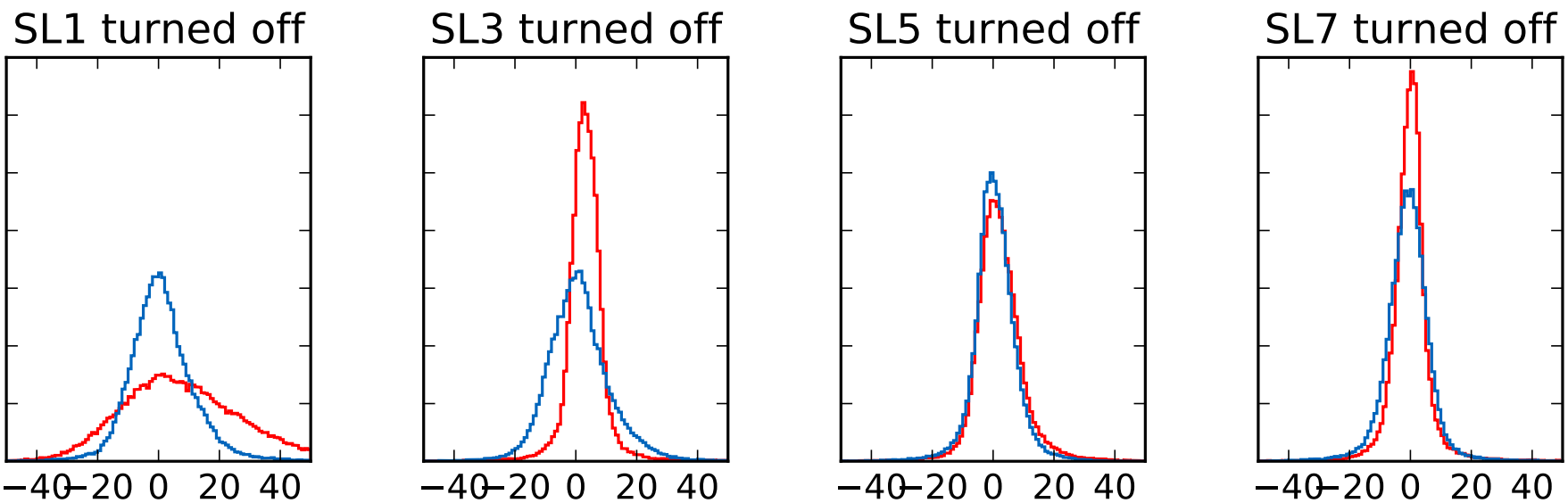
- input: wire hits from the central drift chamber
  - 9 super layers, alternating axial and stereo
  - 1 hit per super layer, scaled position and drift time
- output: continuous scaled z-vertex

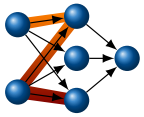




What does the MLP learn?







Summary

Multi Layer Perceptron

- ≥ 3 layers of perceptrons
- connection weights, nonlinear activation
- function approximation / classification

Training: backpropagation

- input/output samples
- minimize cost function

Belle II neurotrigger

- fast 3D track reconstruction