you may register yourself
in IVLE
here, before lecture starts

Please do final project registration!!

Welcome to

CS5242

Neural Networks and Deep Learning

Lee Hwee Kuan & Wang Wei

Teaching assistant

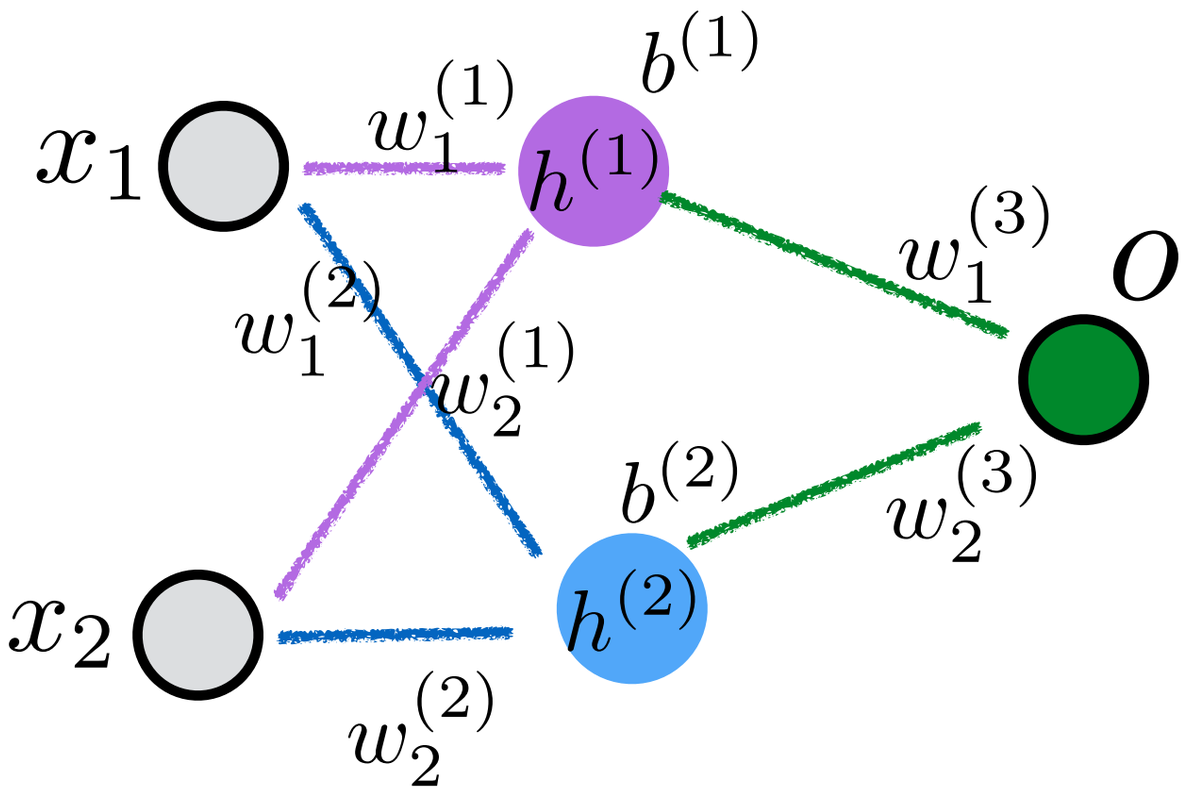
Connie Kou Khor Li, Ji Xin, Ouyang Kun

CS5242 mirror site:

https://web.bii.a-star.edu.sg/~leehk/cs5424.html

Short review

Basic construction of neural networks

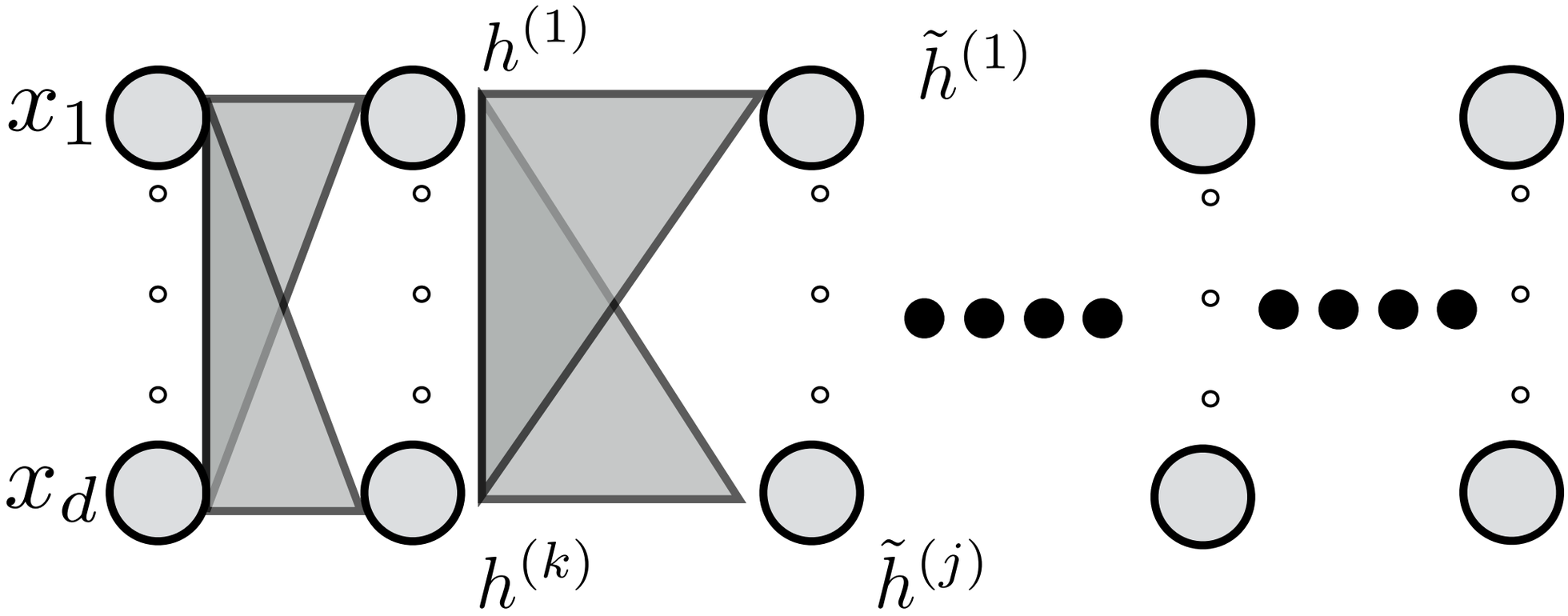


$$h^{(1)} = \sigma(w_1^{(1)} x_1 + w_2^{(1)} x_2 + b^{(1)})$$

$$h^{(2)} = \sigma(w_1^{(2)} x_1 + w_2^{(2)} x_2 + b^{(2)})$$

$$o = \sigma(w_1^{(3)} h^{(1)} + w_2^{(3)} h^{(2)} + b^{(3)})$$

Basic construction of neural networks

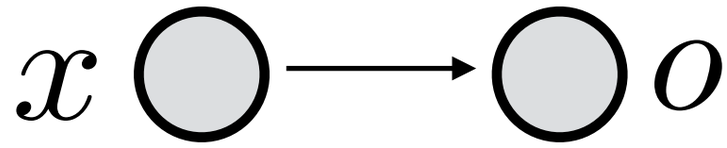


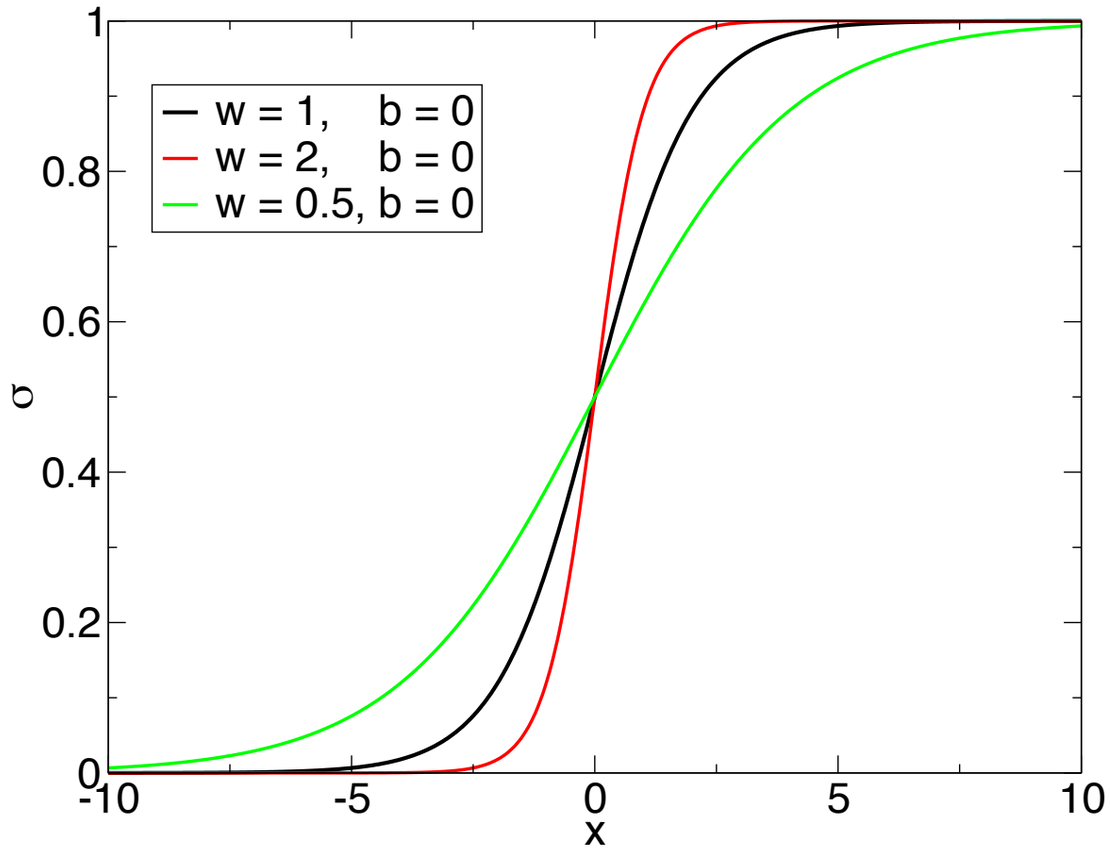
$$h^{(1)} = \sigma(\vec{w}_1 \cdot \vec{x} + b_1)$$
$$h^{(k)} = \sigma(\vec{w}_k \cdot \vec{x} + b_k)$$
$$\tilde{h}^{(j)} = \sigma(\vec{w}_j \cdot \vec{h} + b_j)$$

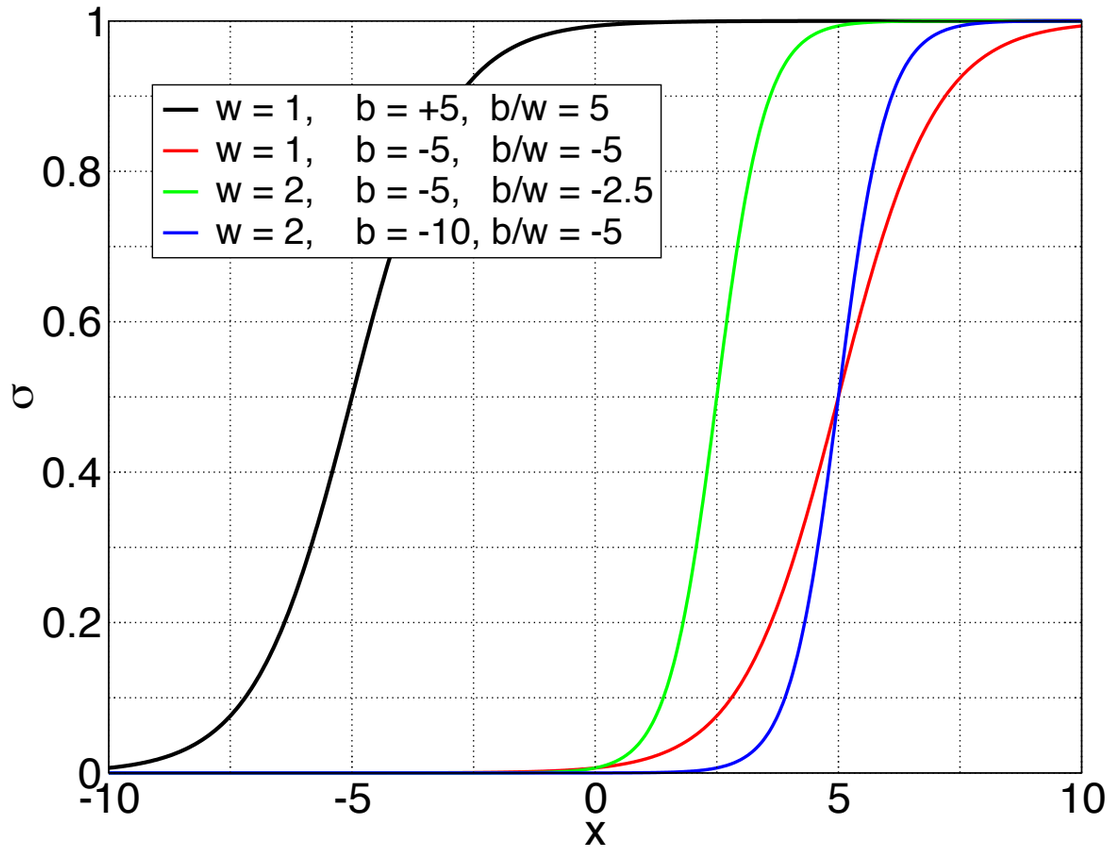
Role of parameters w and b

$$x \in \mathbb{R}$$

$$o = \frac{1}{1 + \exp(-wx - b)}$$



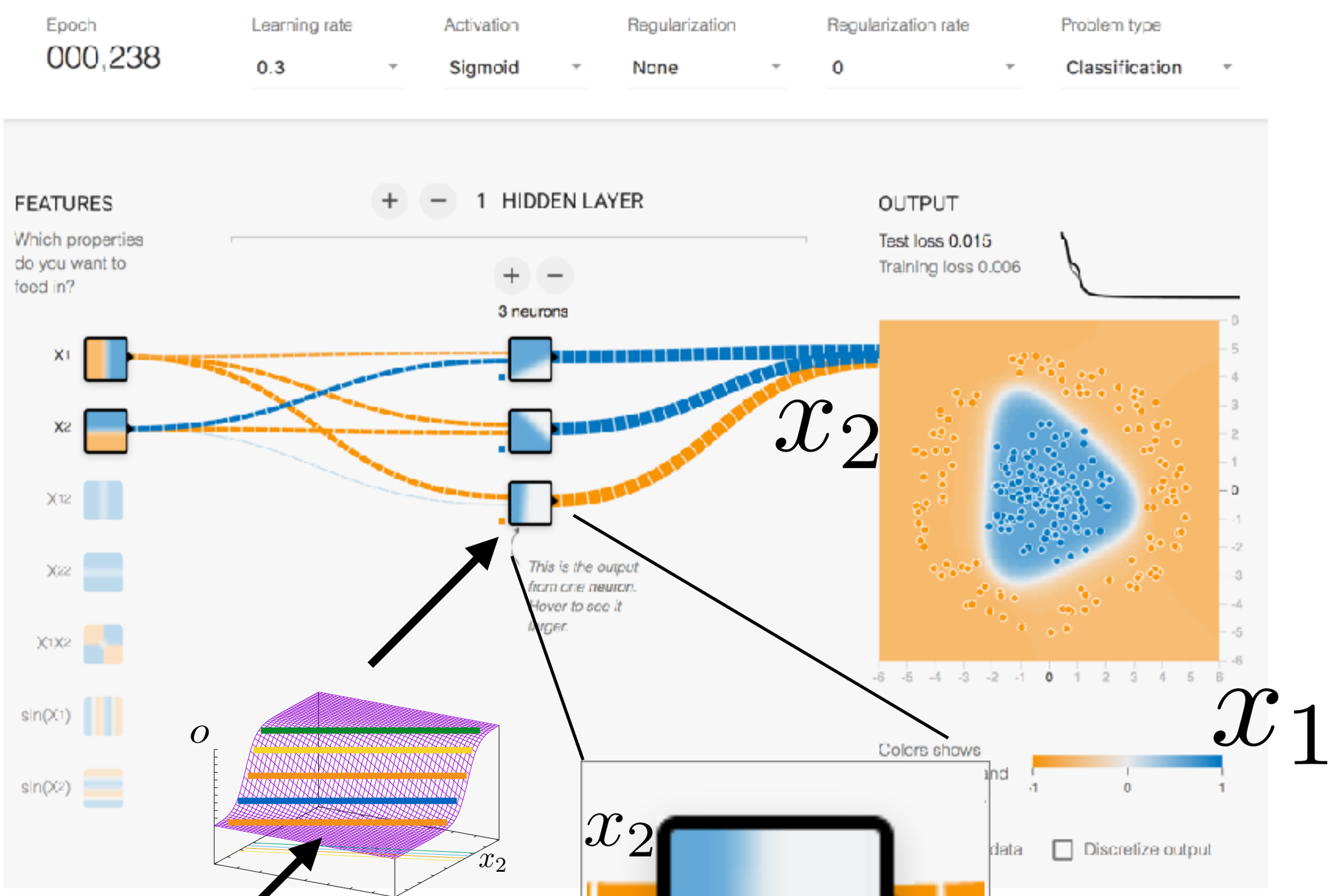




Role of parameters w and b

Do it yourself : plot ReLU graph for various w b

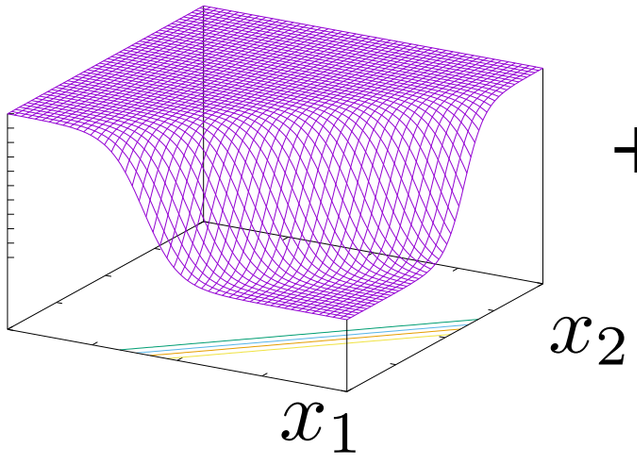
Function picture of network



level sets form straight lines

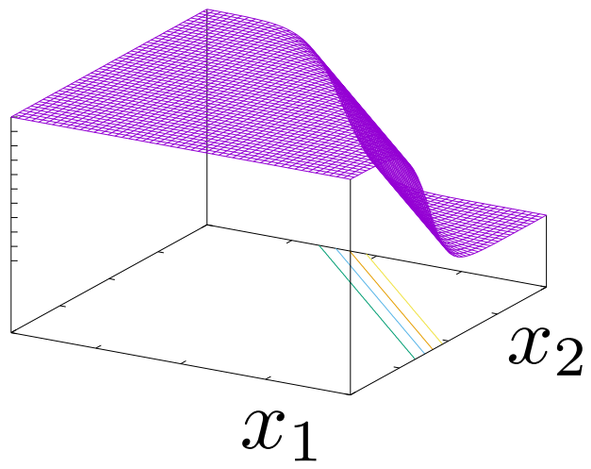
Function view of neural network

$h_1(x_1, x_2)$



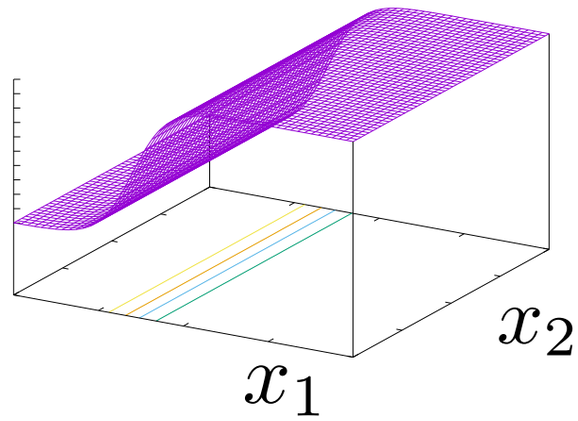
+

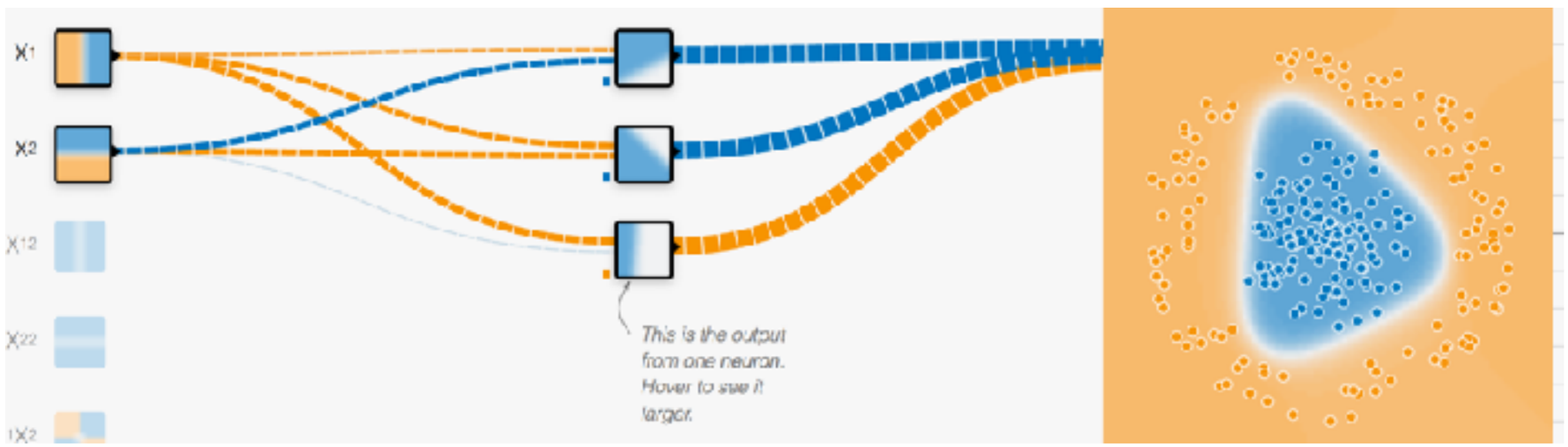
$h_2(x_1, x_2)$

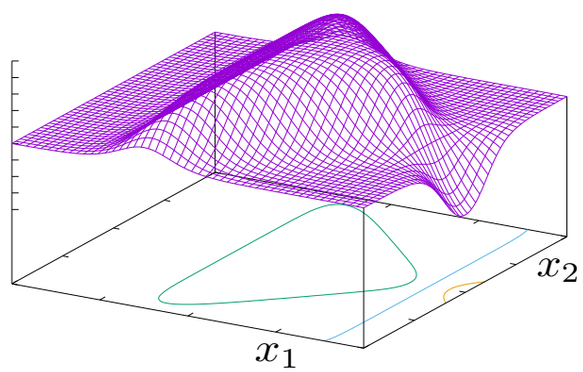


+

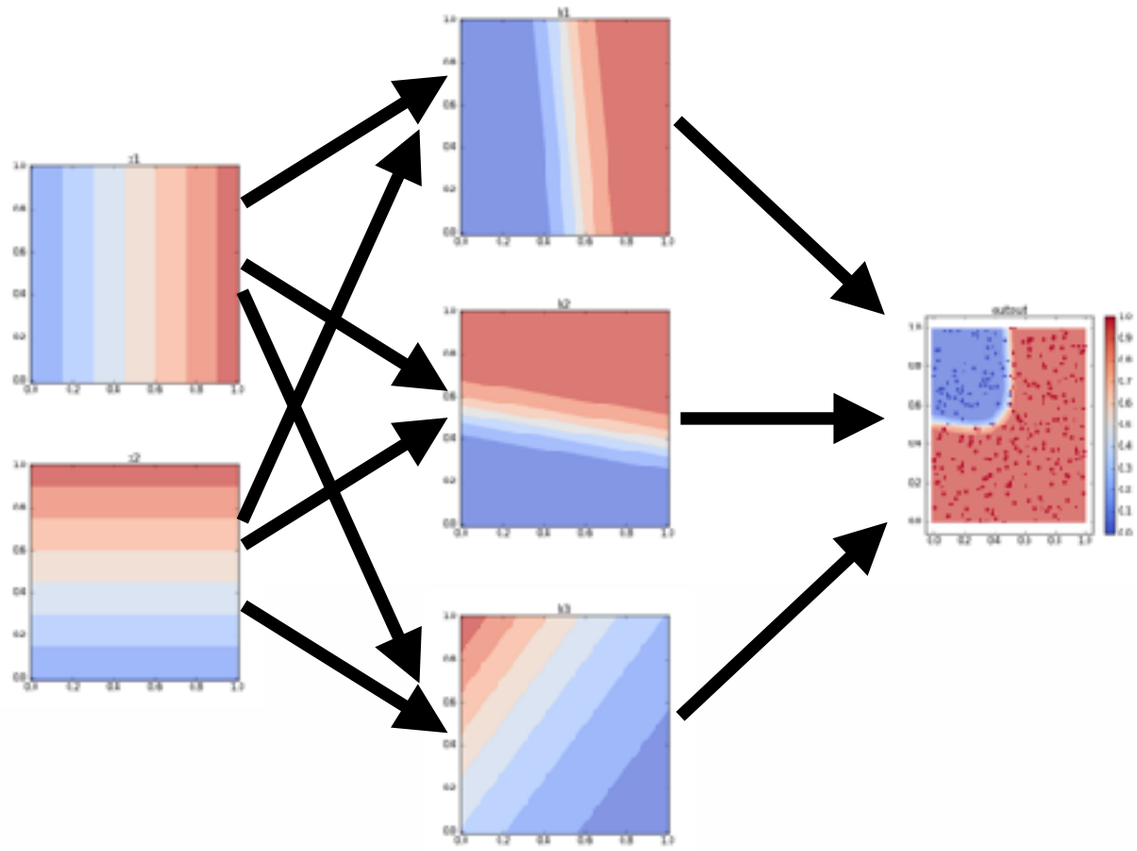
$-h_3(x_1, x_2)$

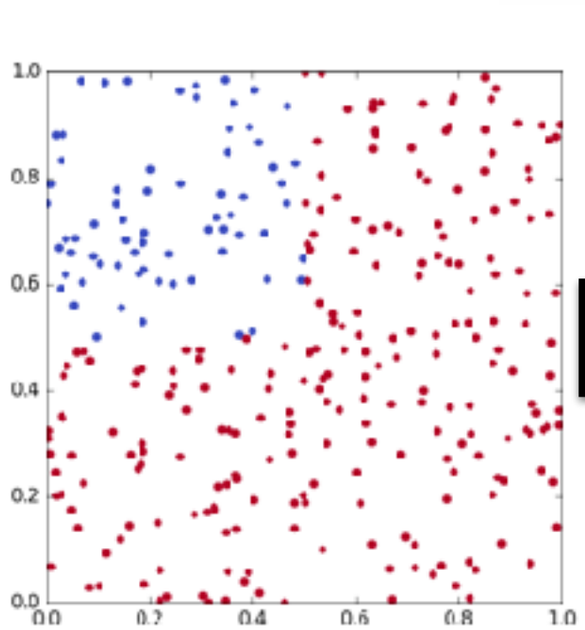


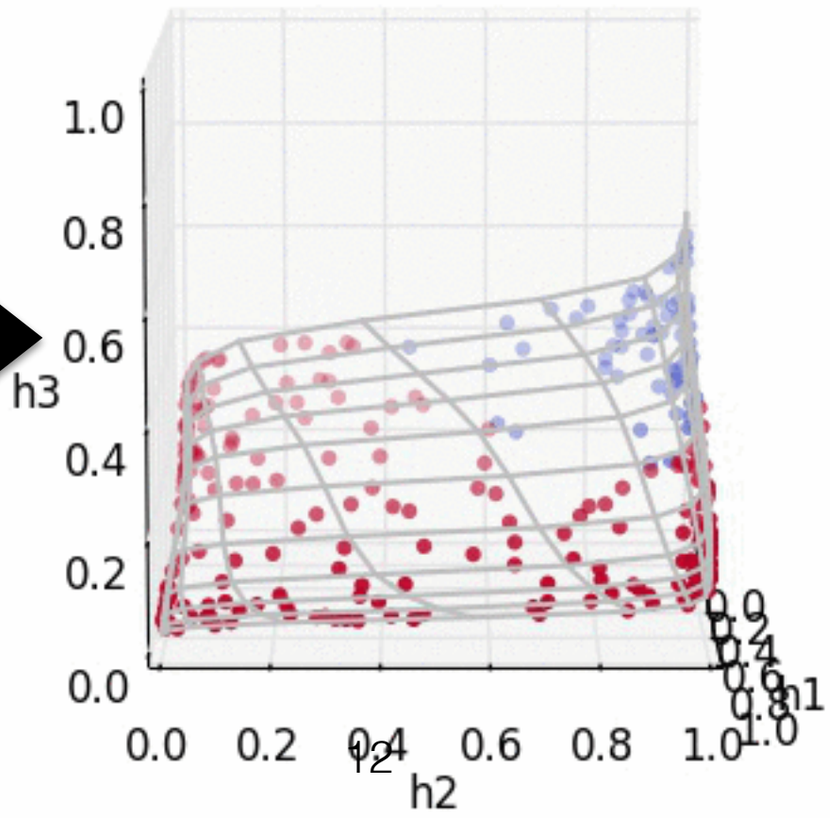




Manifold view of neural network





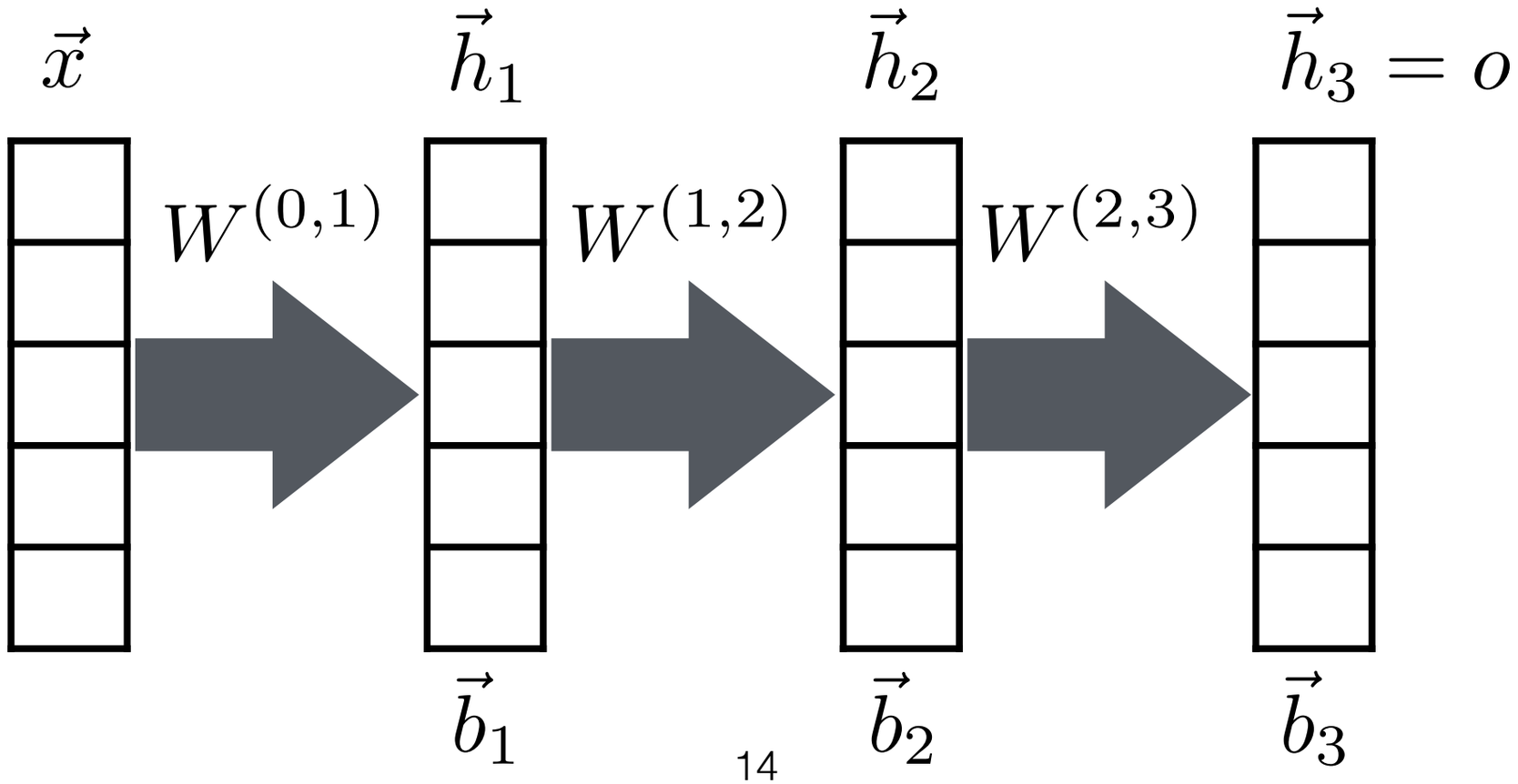


Matrix notation

$$h_{1,i} = \sigma\left(\sum_j W^{(0,1)}_{j,i} x_j + b_i\right) \qquad i = 1, \cdots$$

$$\vec{h}_1 = \sigma[W^{(0,1)} \cdot \vec{x} + \vec{b}_1]$$ element wise operation

$$\sigma[(z_1, z_2, z_3, \cdots)] = (\sigma(z_1), \sigma(z_2), \sigma(z_3))$$

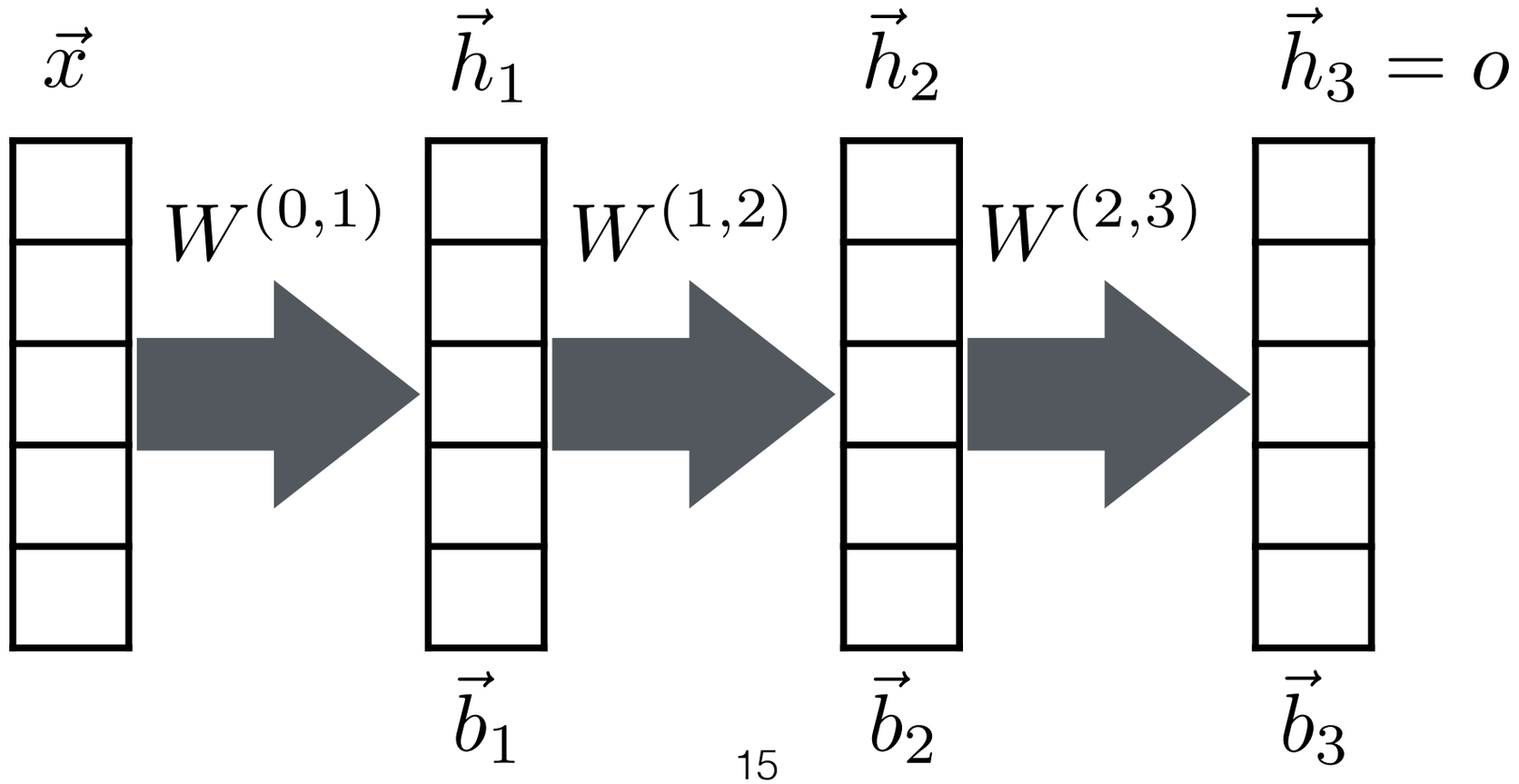


$$\vec{h}_1 = \sigma[W^{(0,1)} \cdot \vec{x} + \vec{b}_1]$$

$$\vec{h}_2 = \sigma[W^{(1,2)} \cdot \vec{h}_1 + \vec{b}_2]$$

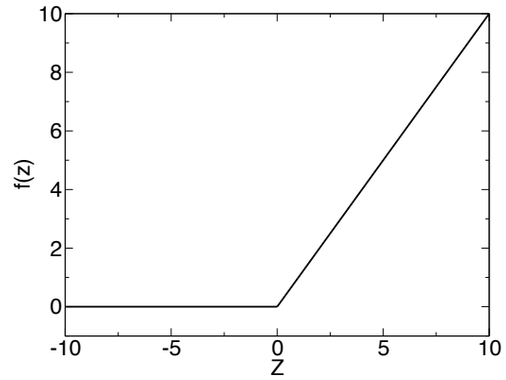
$$\vec{h}_3 = \sigma[W^{(2,3)} \cdot \vec{h}_2 + \vec{b}_3]$$

$$\sigma[(z_1, z_2, z_3, \cdots)] = (\sigma(z_1), \sigma(z_2), \sigma(z_3))$$



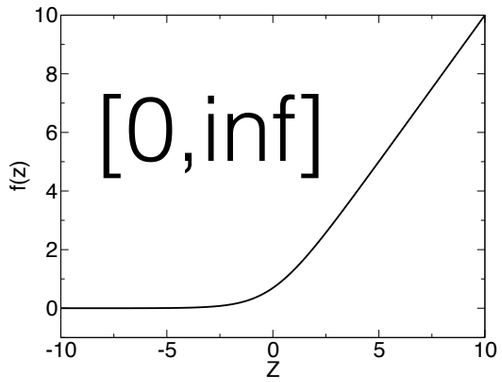
Activation functions

Rectified linear $f(z) = \max(0, z)$



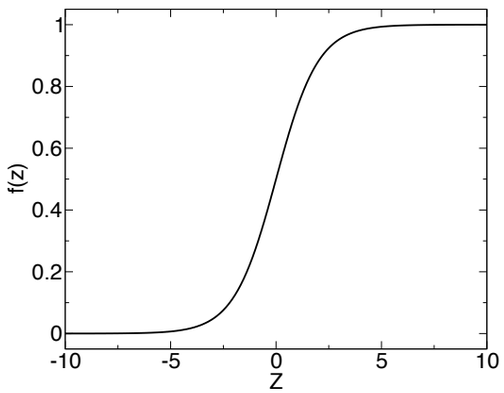
[0,inf]

Soft plus $f(z) = \ln(1 + \exp(z))$



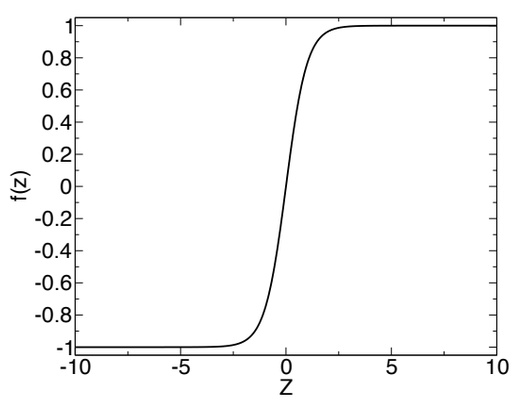
[0,inf]

Sigmoid $f(z) = \dfrac{1}{1 + \exp(-z)}$



[0,1]

Hyperbolic tangent $f(z) = \tanh(z)$



[-1,1]

so many examples in https://en.wikipedia.org/wiki/Activation_function

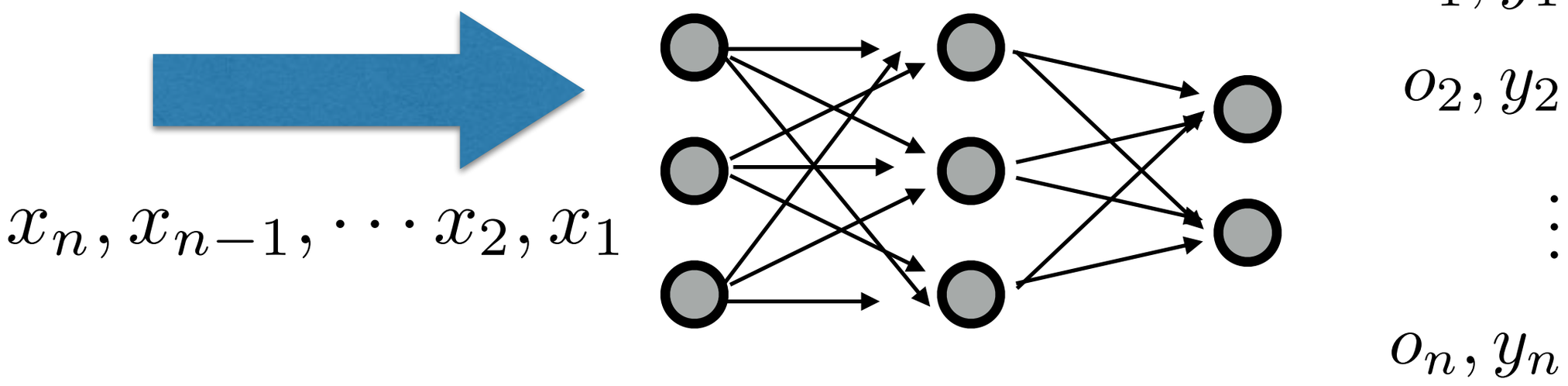
Cost function

Square error

Given data points $(x_i, y_i), i = 1, \cdots n$

$$l(y_i, o_i) = \|y_i - o_i\|^2$$

$$C = \frac{1}{n} \sum_i l(y_i, o_i)$$



Adjust the weights until C = 0 (or close to zero)

Cross entropy cost function

Two class example $\quad y_i \in \{0, 1\}$

$$C = -\frac{1}{n} \sum_i y_i \log[o(x_i)] + (1 - y_i) \log[1 - o(x_i)]$$

Three class example

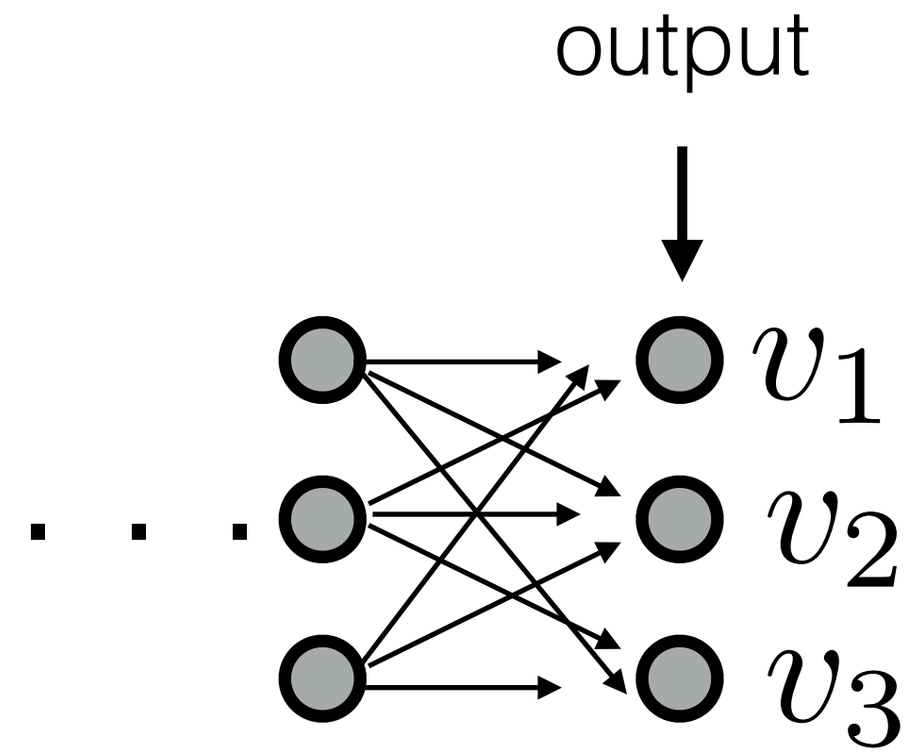
$$y_i \in \{(1,0,0), (0,1,0), (0,0,1)\}$$

$$m = \exp(v_1) + \exp(v_2) + \exp(v_3)$$

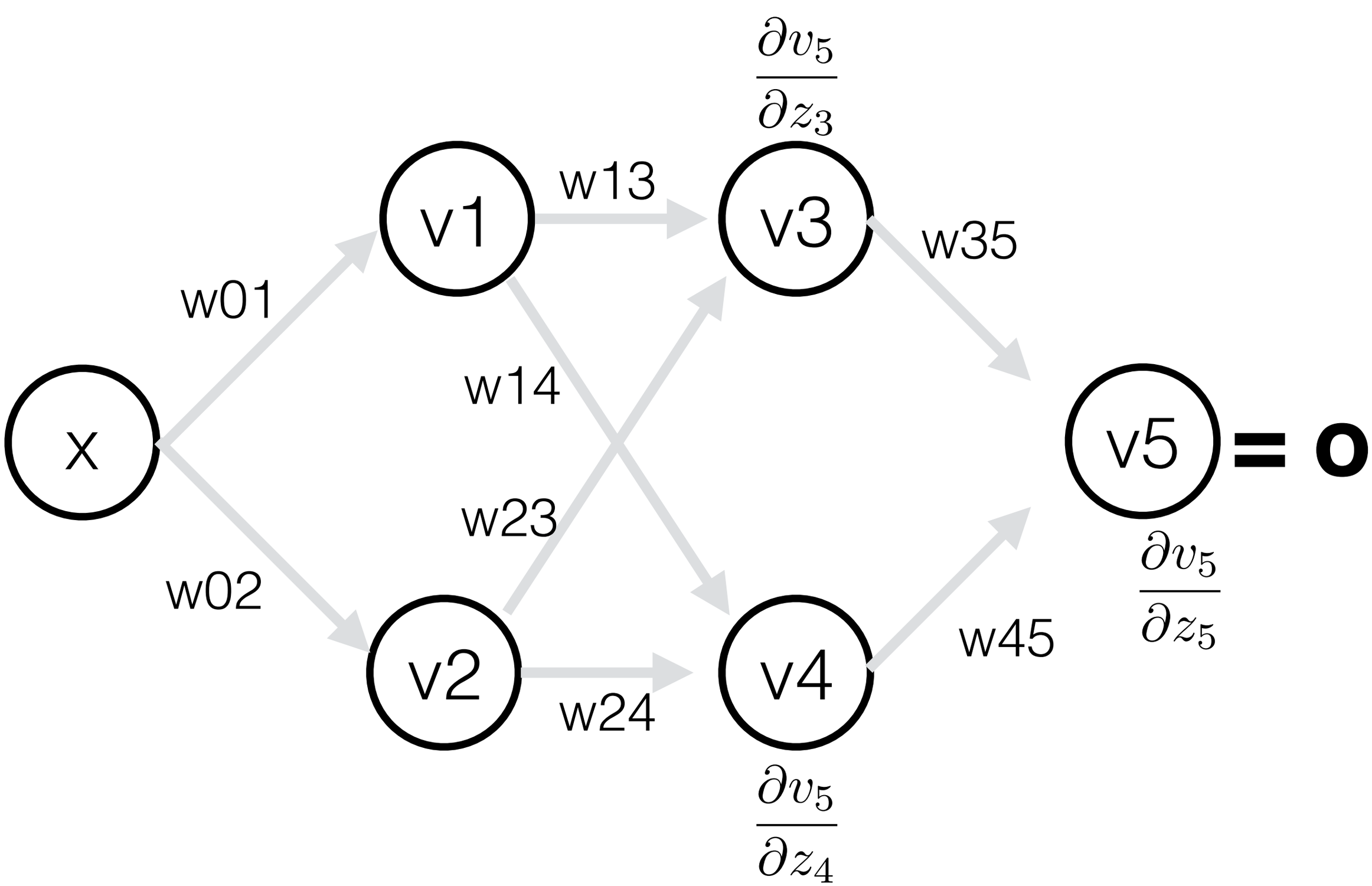
$$o_i = (\frac{\exp(v_1)}{m}, \frac{\exp(v_2)}{m}, \frac{\exp(v_3)}{m})$$

$$l(y_i, o_i) = y_i \cdot \log(o_i)$$

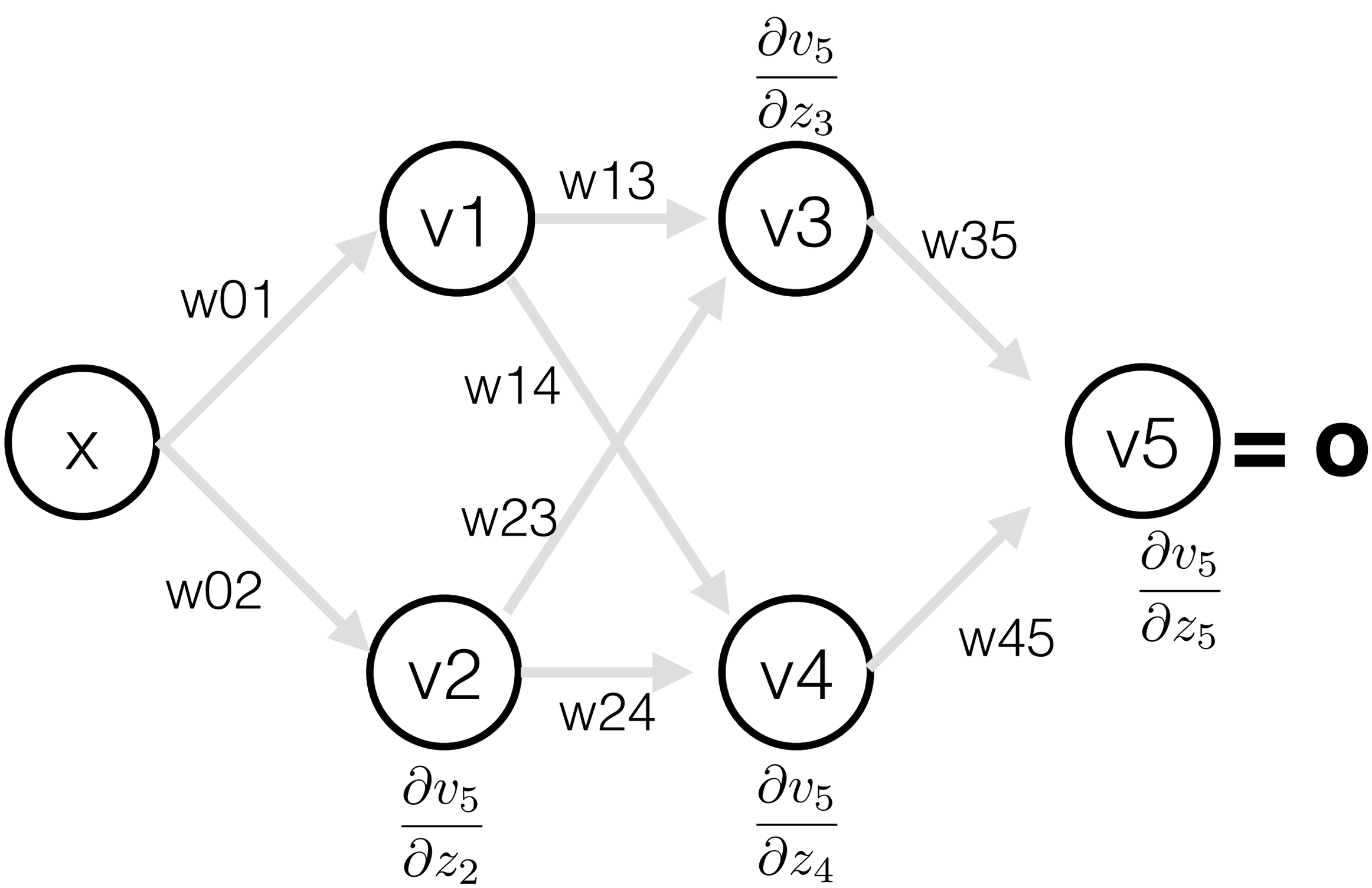
$$C = \frac{1}{n} \sum_i l(y_i, o_i)$$



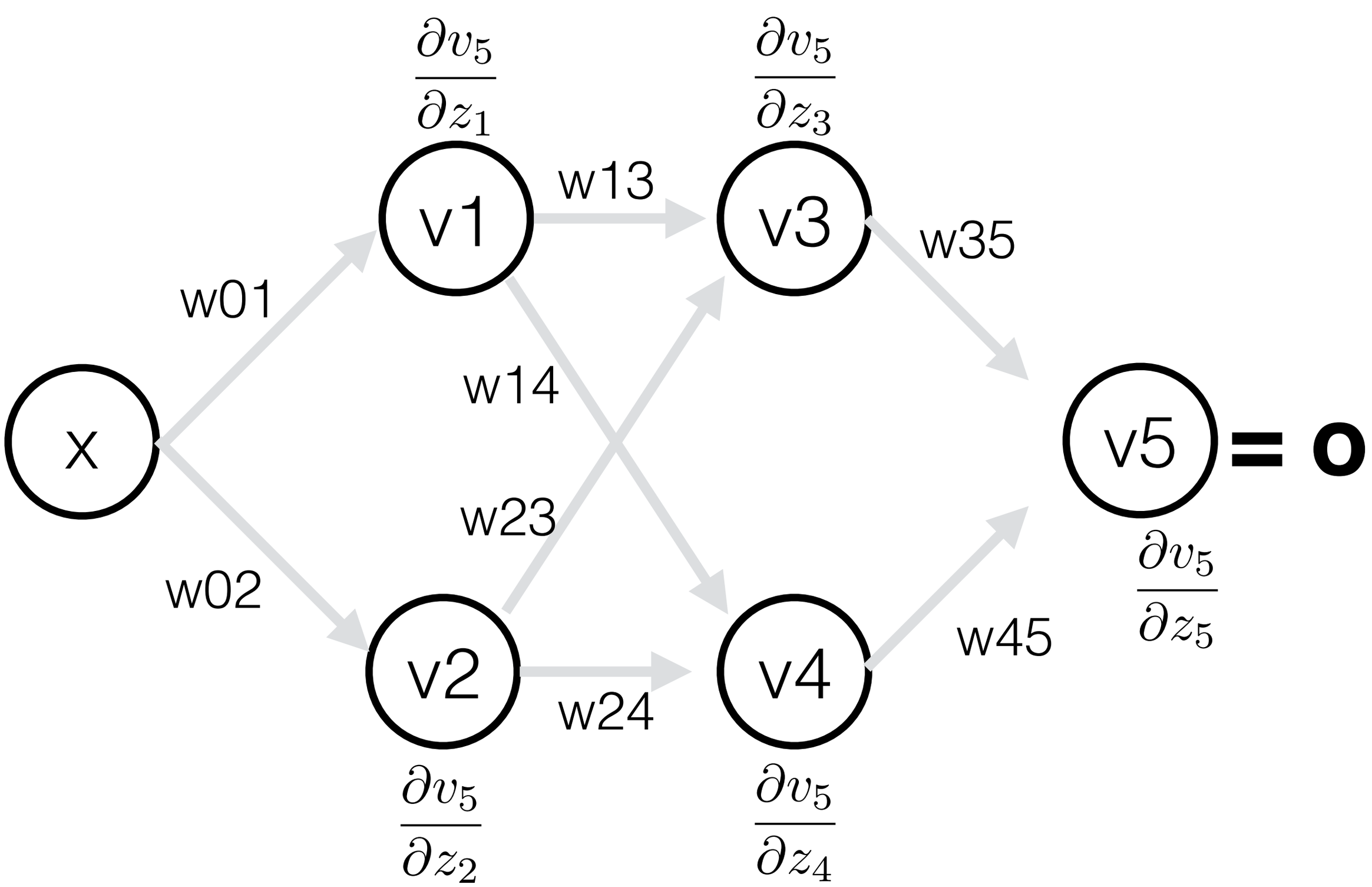
Back propagation and gradient descend



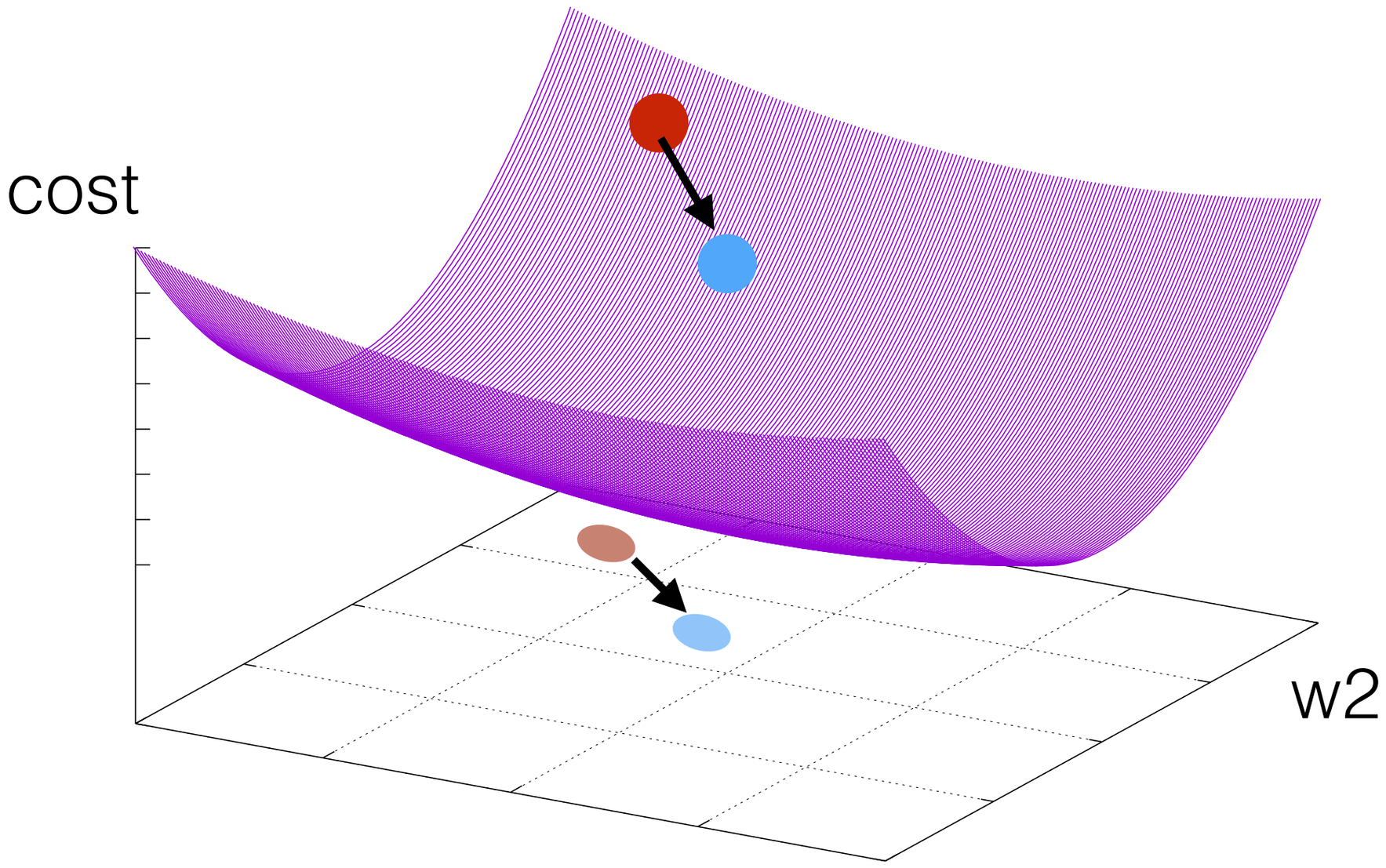
$$\frac{\partial v_5}{\partial z_4} = \frac{\partial v_5}{\partial z_5}\sigma'(z_4)w_{45}$$



$$\frac{\partial v_5}{\partial z_2} = \frac{\partial v_5}{\partial z_4}\sigma'(z_2)w_{24} + \frac{\partial v_5}{\partial z_3}\sigma'(z_2)w_{23}$$



This is call the back propagation algorithm



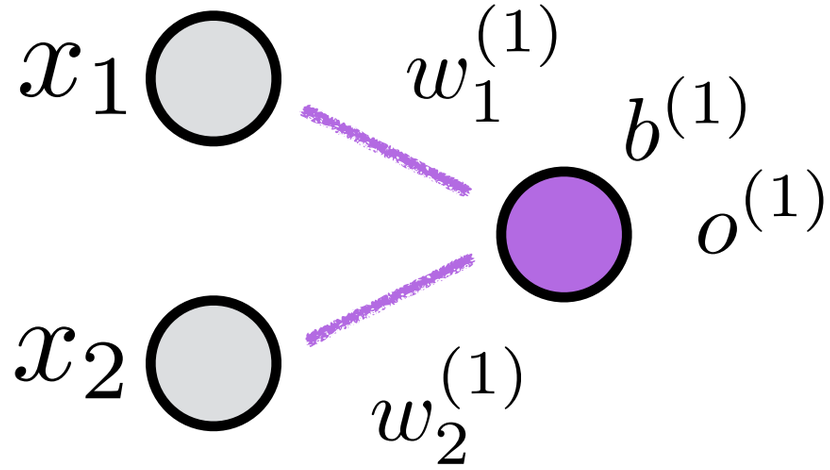
$$w_j(t+1) = w_j(t) - \eta \frac{\partial C}{\partial w_j}$$

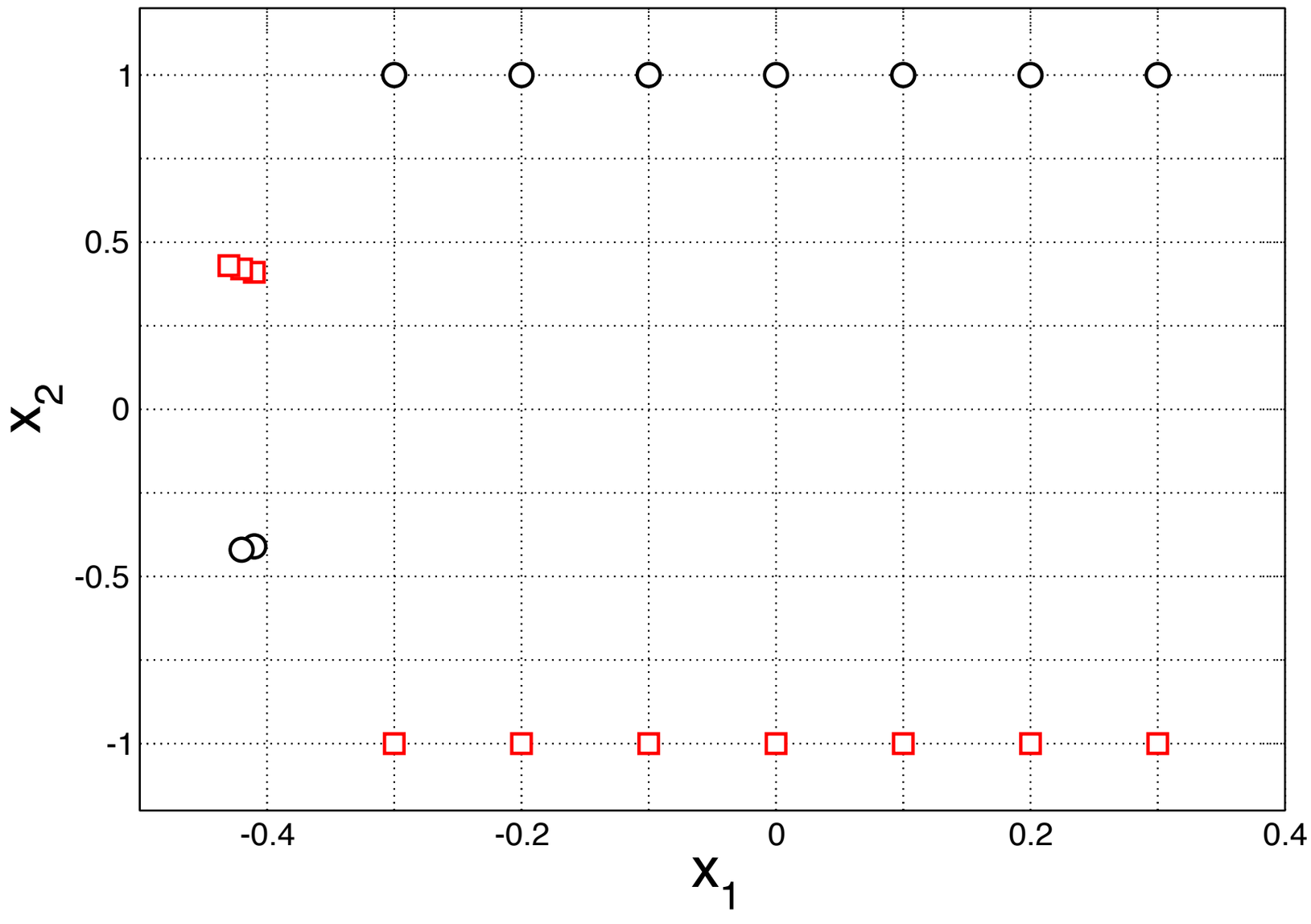
Local minimum problem

$$o_i = \sigma(w_1 x_1 + w_2 x_2) \text{ with } w_2 = 1$$

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

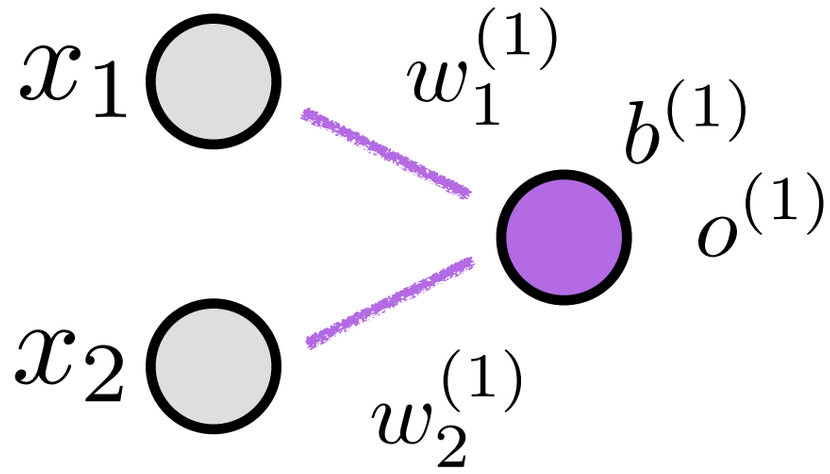
$$C(w_1) = \frac{1}{n} \sum_i (y_i - o_i)^2$$





$$o_i = \sigma(w_1 x_1 + w_2 x_2) \text{ with } w_2 = 1$$

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$



$$C(w_1) = \frac{1}{n} \sum_i (y_i - o_i)^2$$

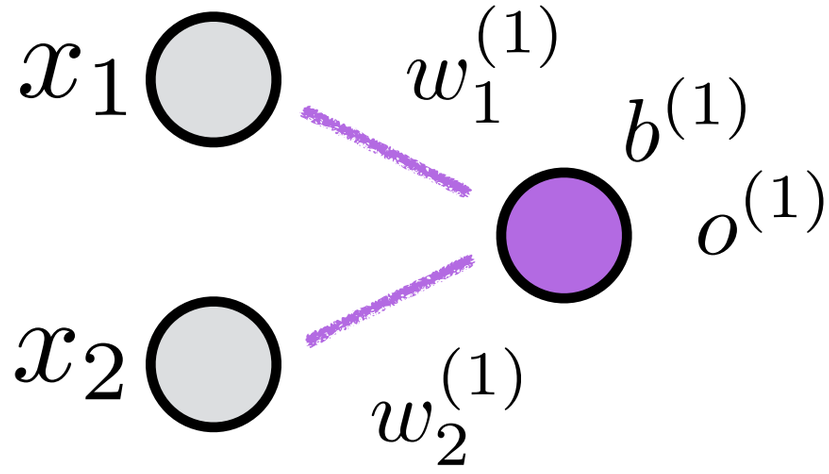
3 errors

2 errors

$\vec{w} = (w_1, w_2) = (0, 1)$

$$o_i = \sigma(w_1 x_1 + w_2 x_2) \text{ with } w_2 = 1$$

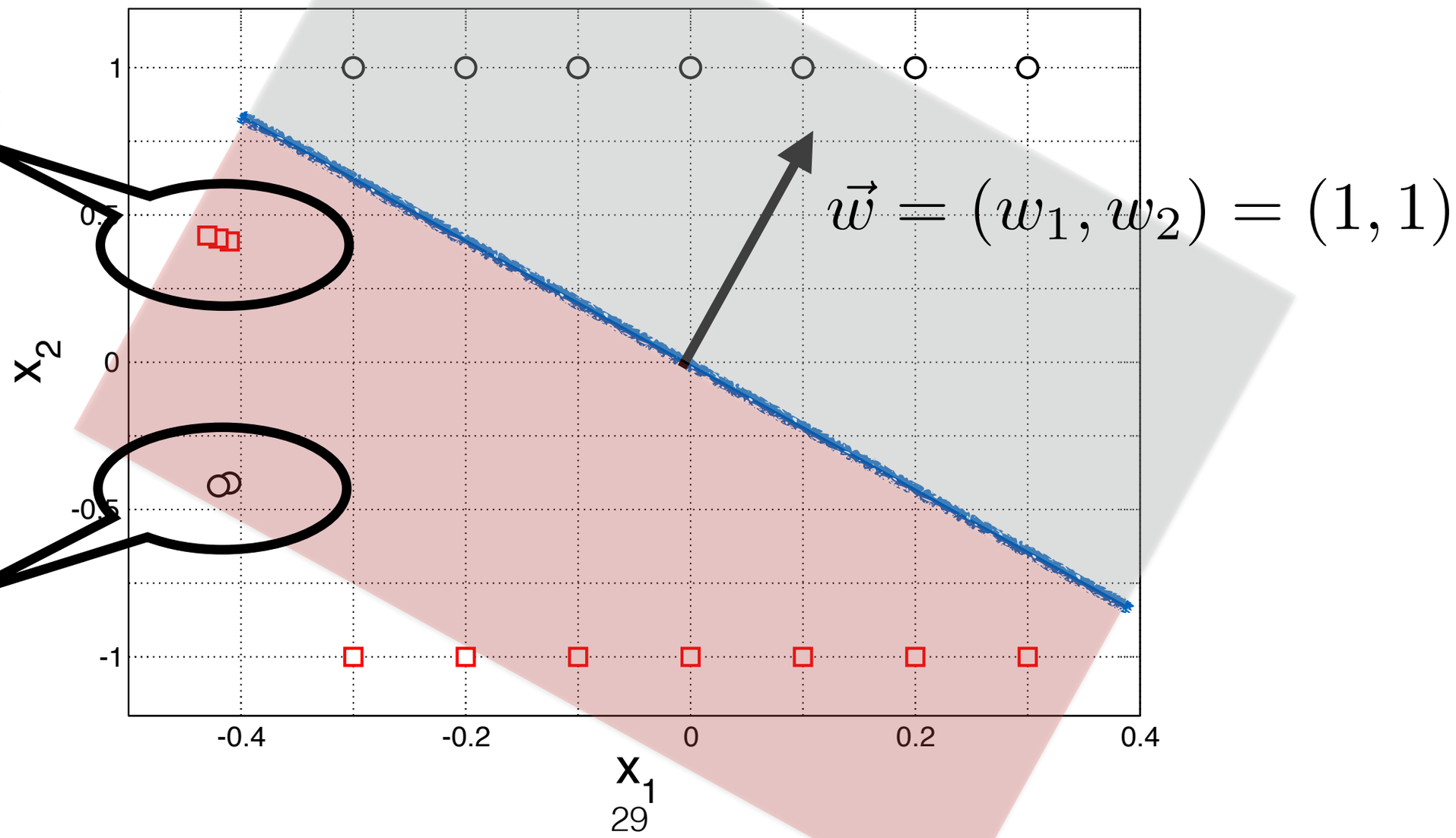
$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$



$$C(w_1) = \frac{1}{n} \sum_i (y_i - o_i)^2$$

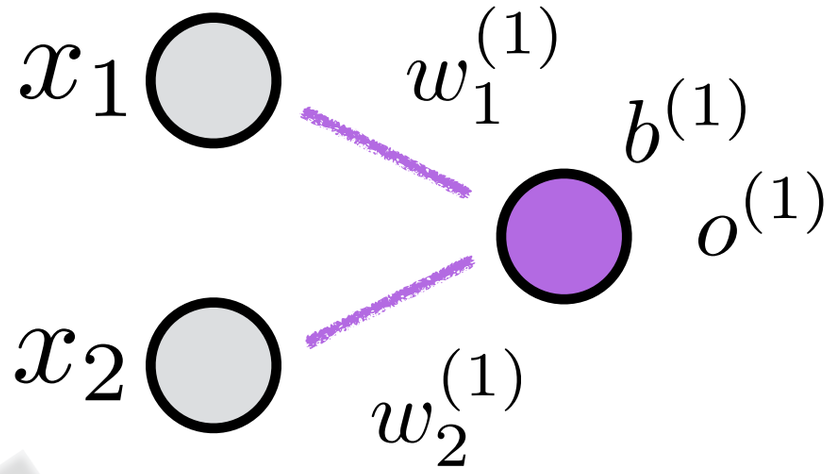
correct

2 errors



$$o_i = \sigma(w_1 x_1 + w_2 x_2) \text{ with } w_2 = 1$$

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$



$$C(w_1) = \frac{1}{n} \sum_i (y_i - o_i)^2$$

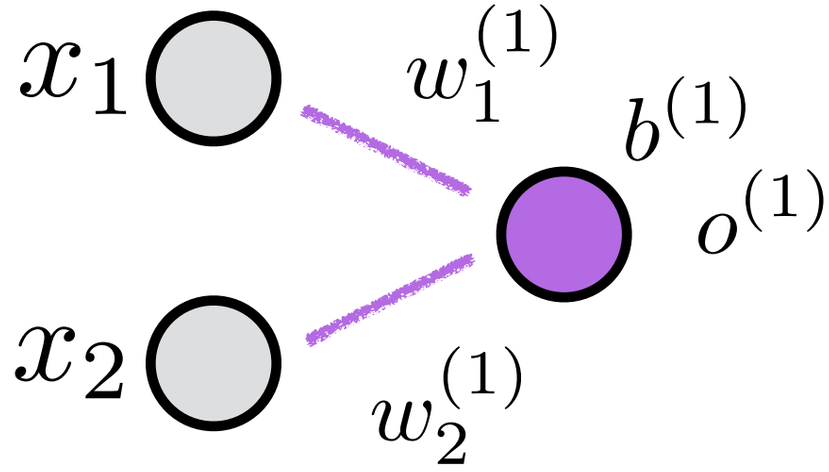
3 errors

correct

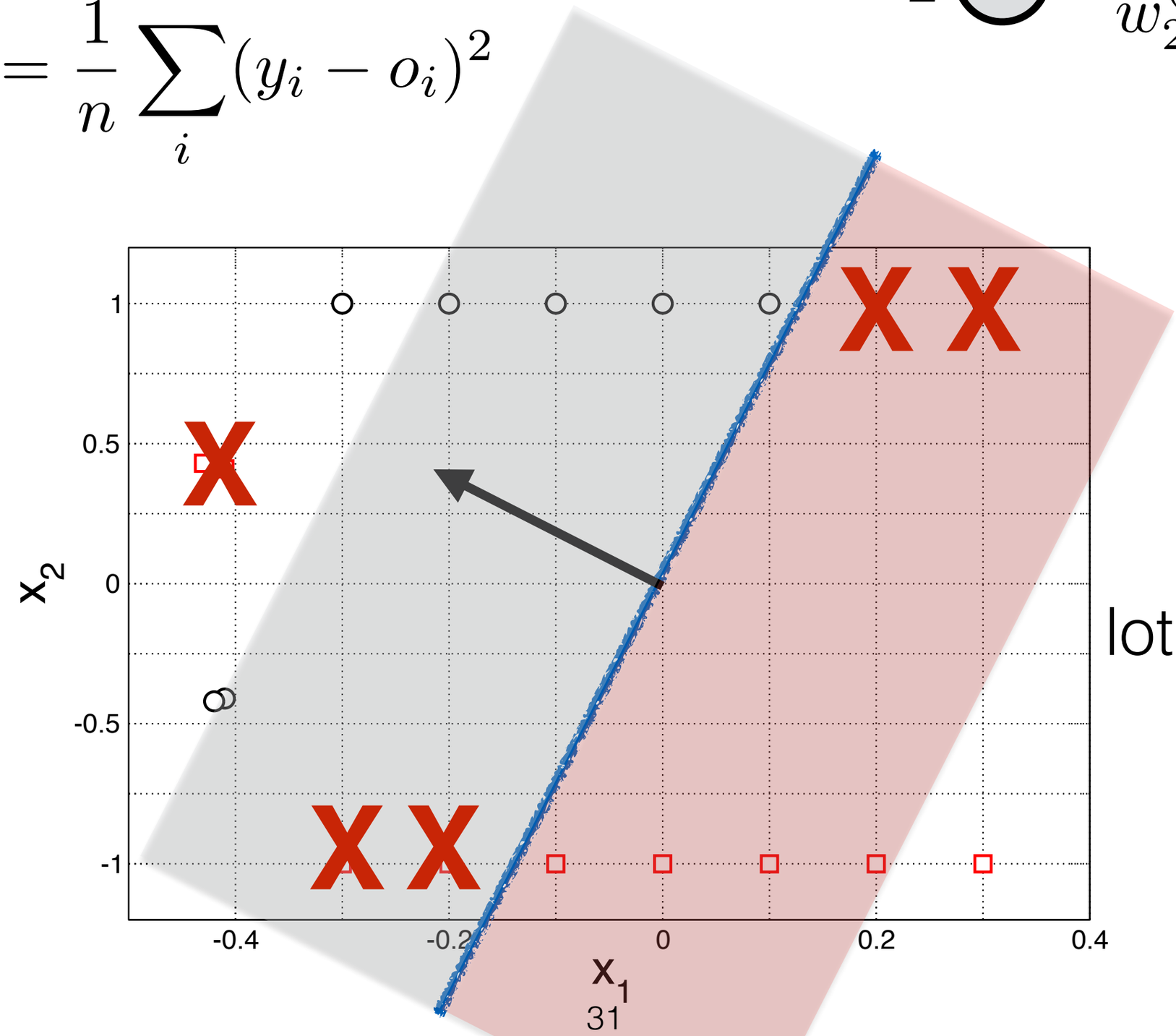
$$\vec{w} = (w_1, w_2) = (-1, 1)$$

$$o_i = \sigma(w_1 x_1 + w_2 x_2) \text{ with } w_2 = 1$$

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

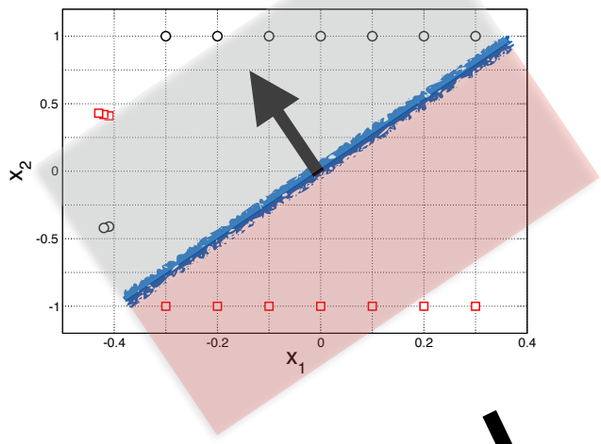


$$C(w_1) = \frac{1}{n} \sum_i (y_i - o_i)^2$$

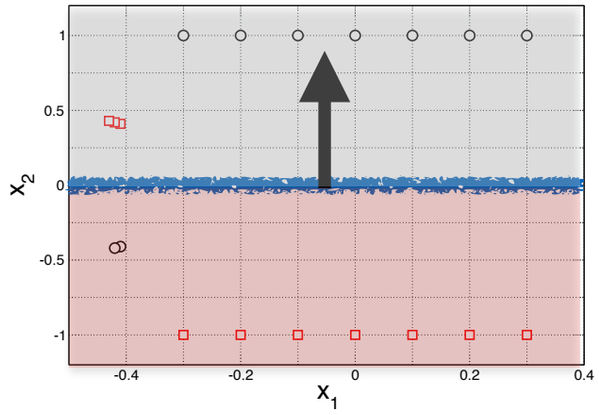


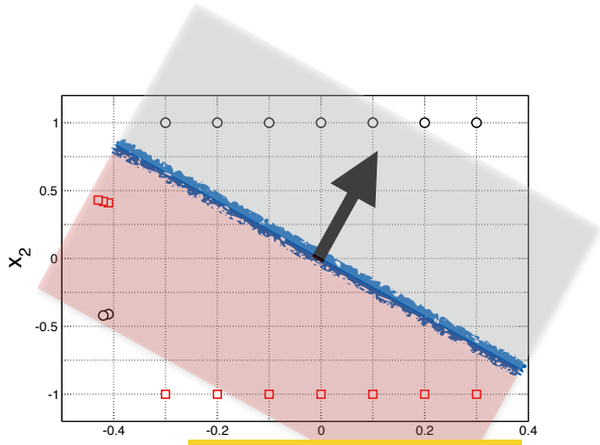
lots of errors!

3 errors

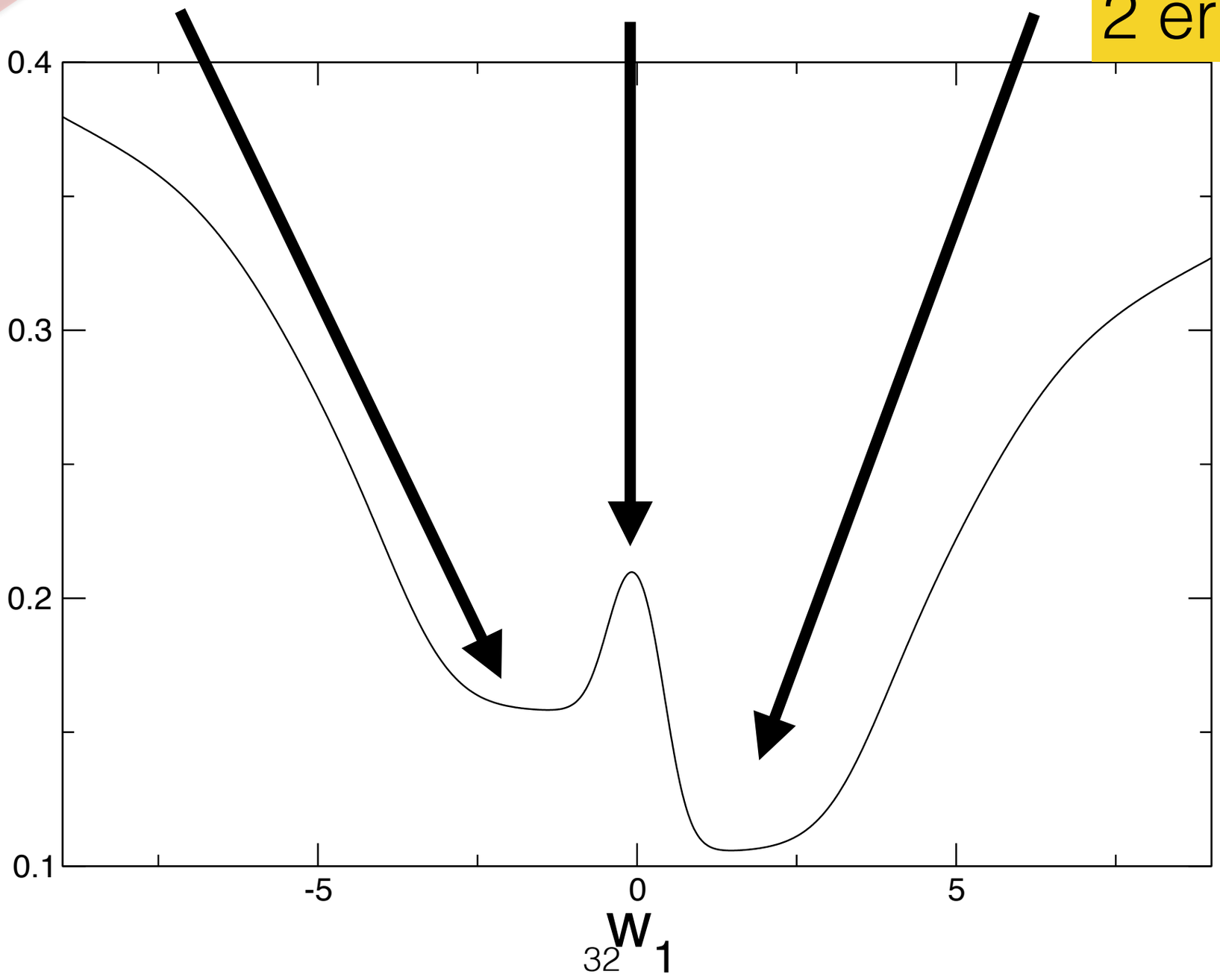


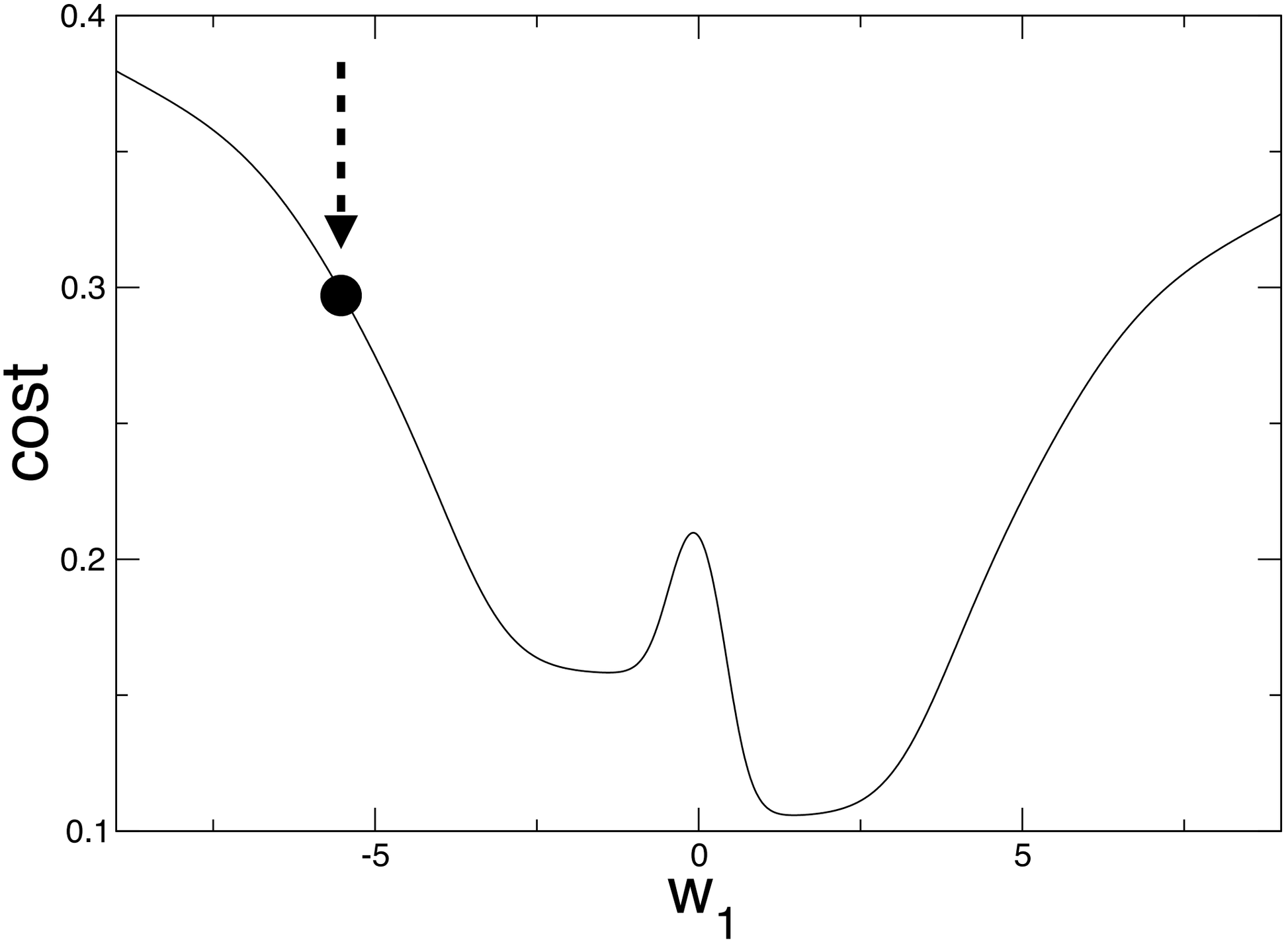
5 errors

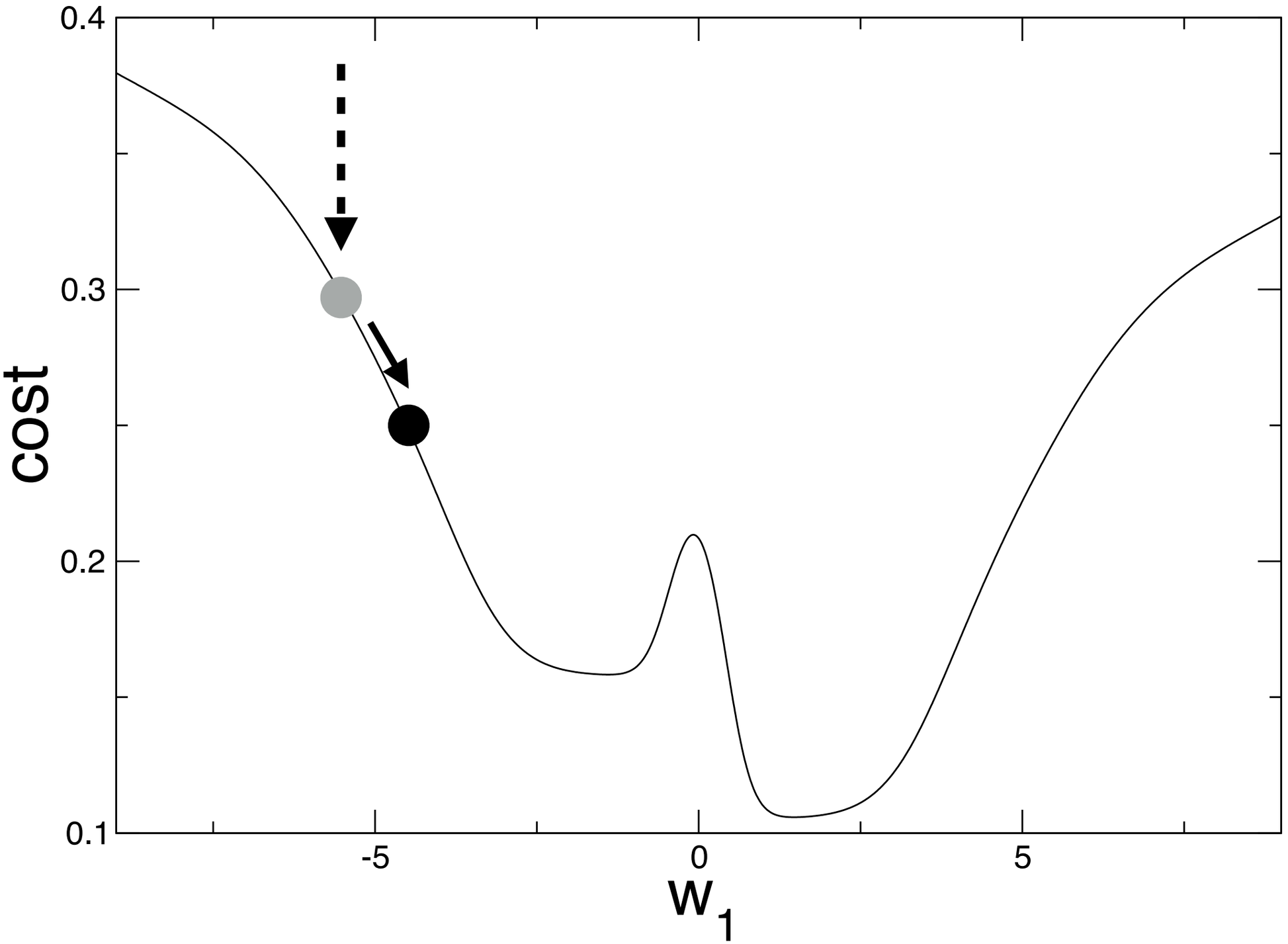


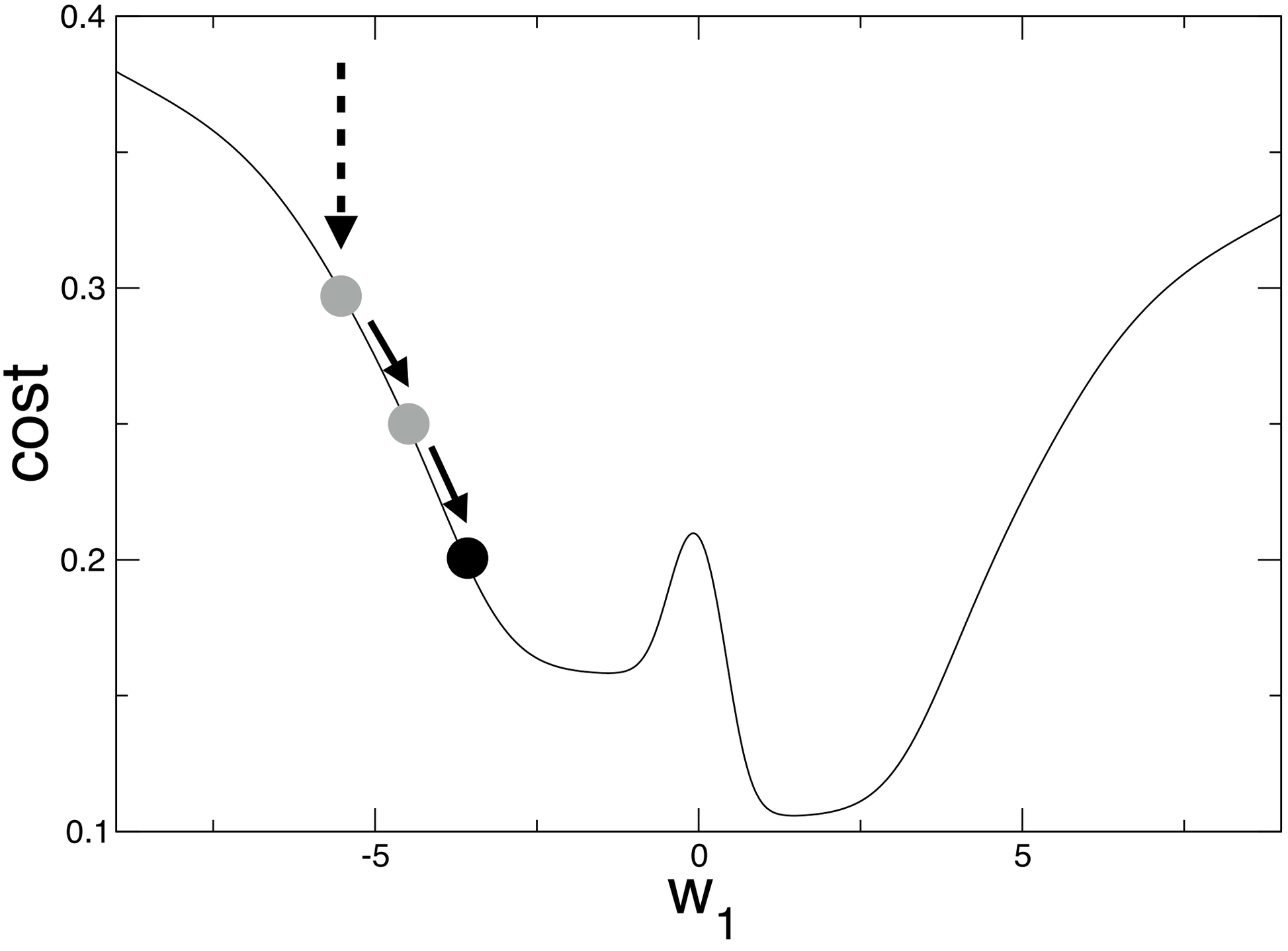


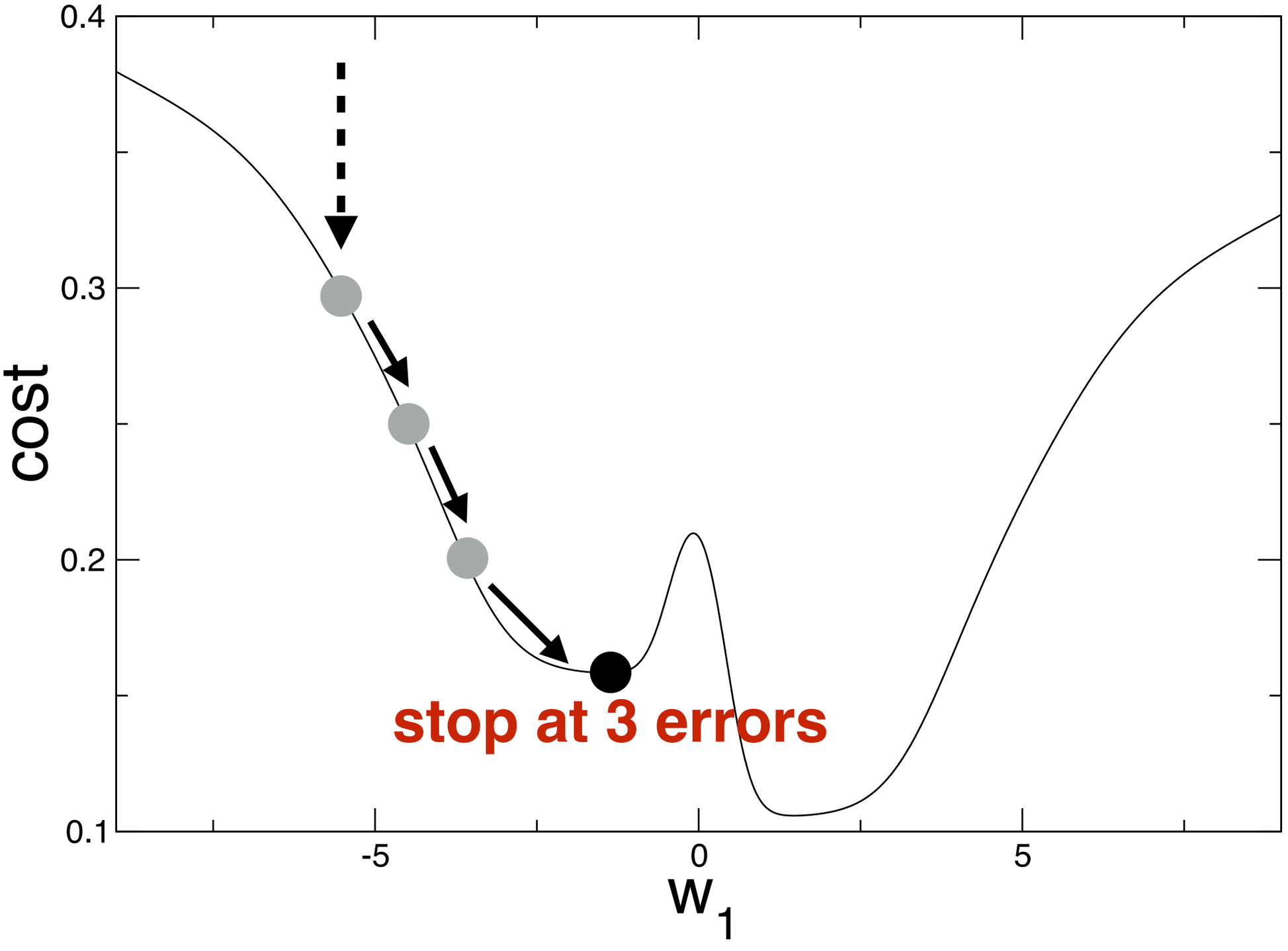
2 errors

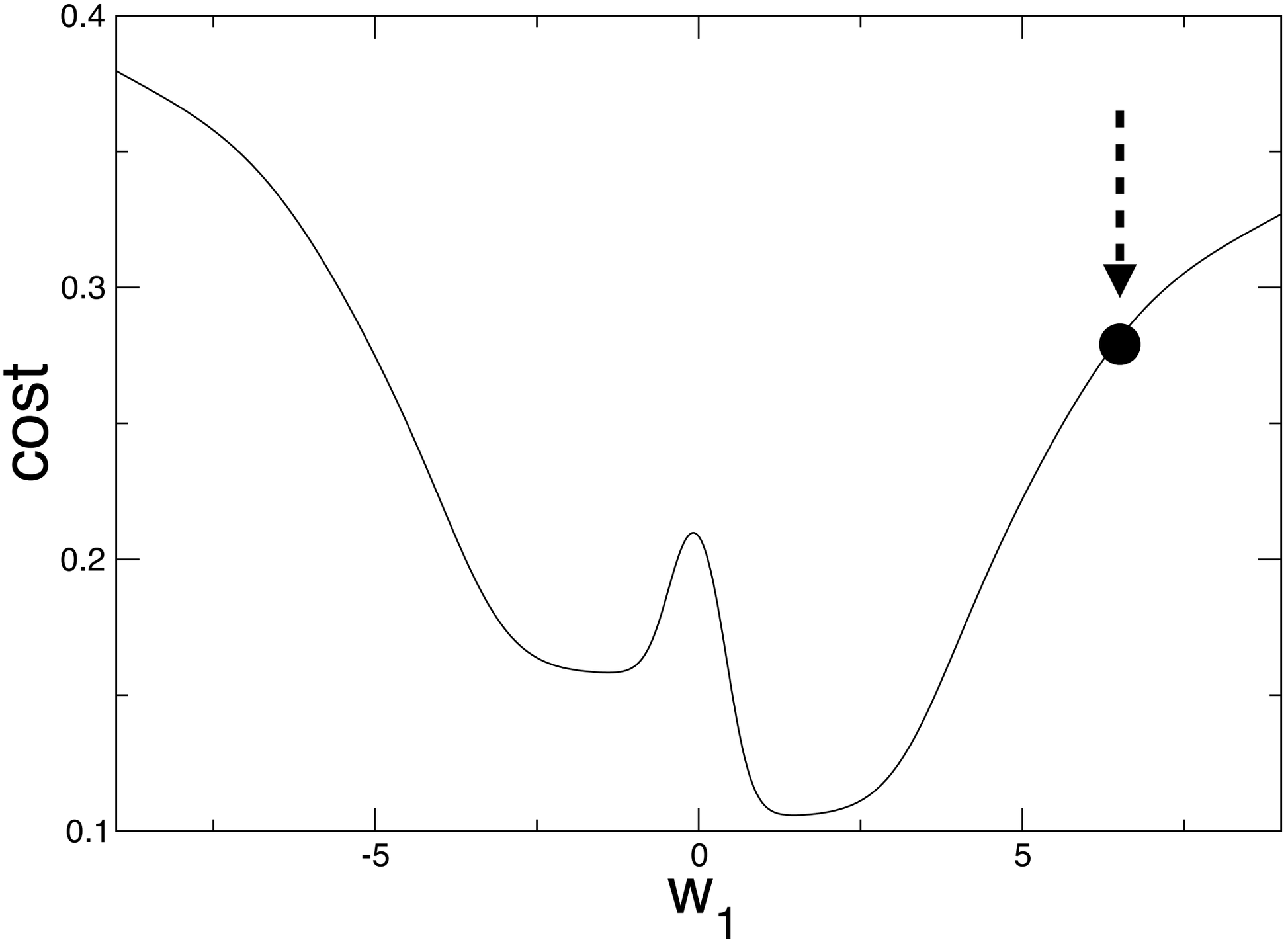


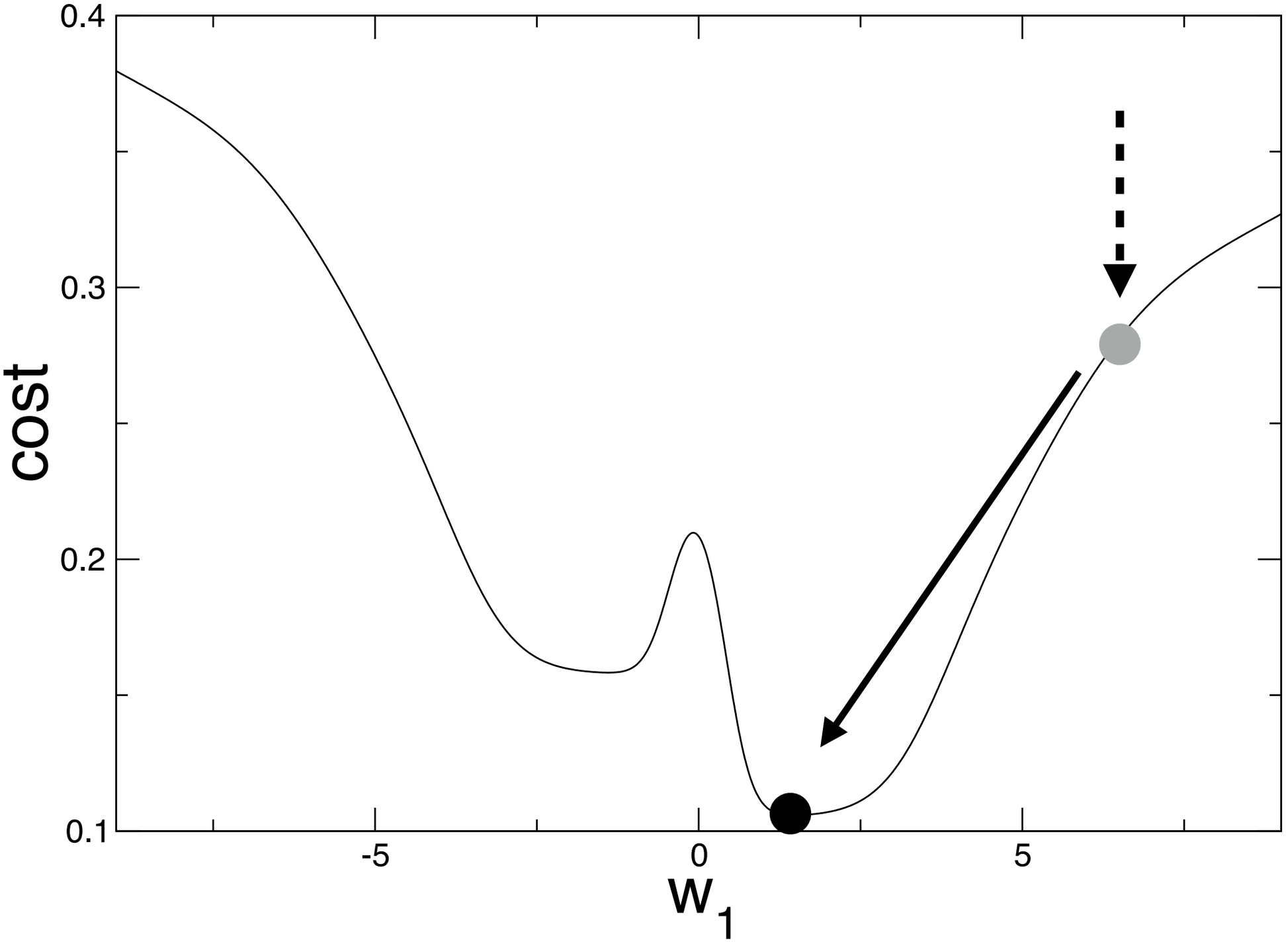

0.4
0.3
0.2
0.1
cost
-5
0
5
W1


0.4
0.3
0.2
0.1
cost
-5
0
5
W1


0.4
0.3
0.2
0.1
cost
-5
0
5
w1

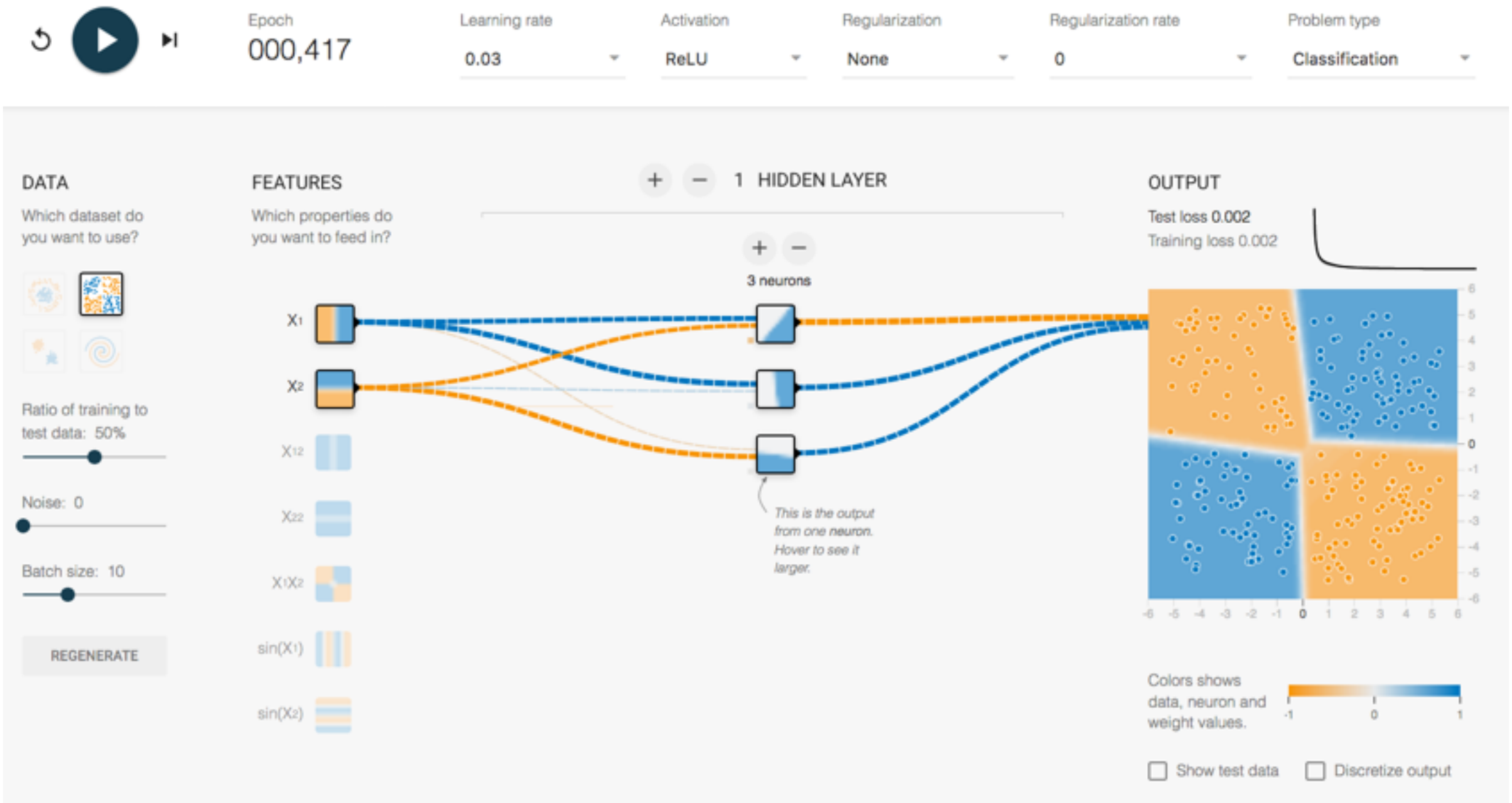

0.4
0.3
0.2
0.1
cost
-5
0
5
w1
stop at 3 errors


0.4
0.3
0.2
0.1
cost
-5
0
5
W1


0.4
0.3
0.2
0.1
cost
-5
0
5
W1

show playground XOR example

Good solution example

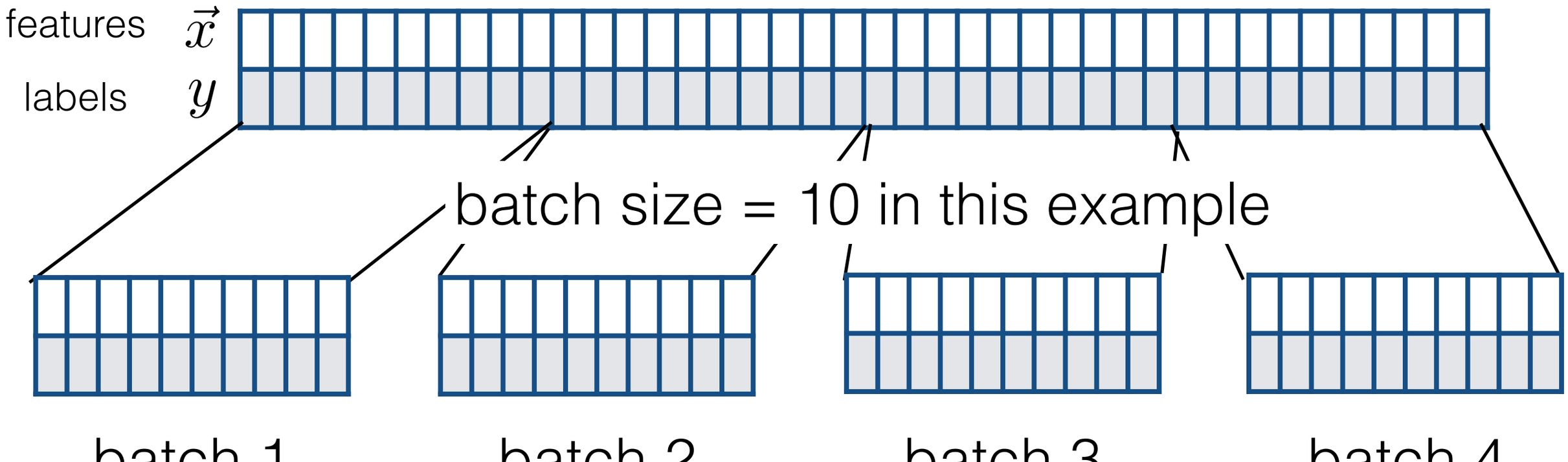


Local minimum examples

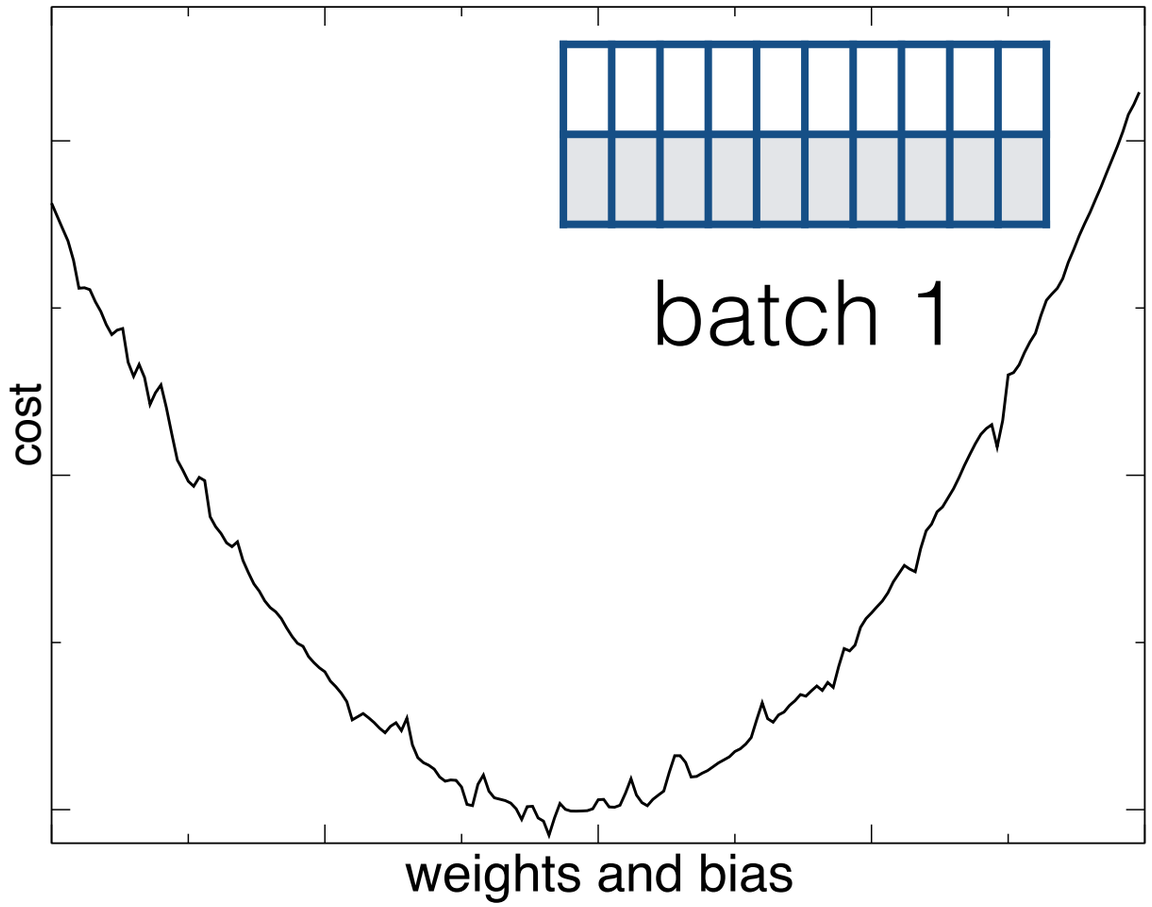
Strategies for overcoming local minimum problem

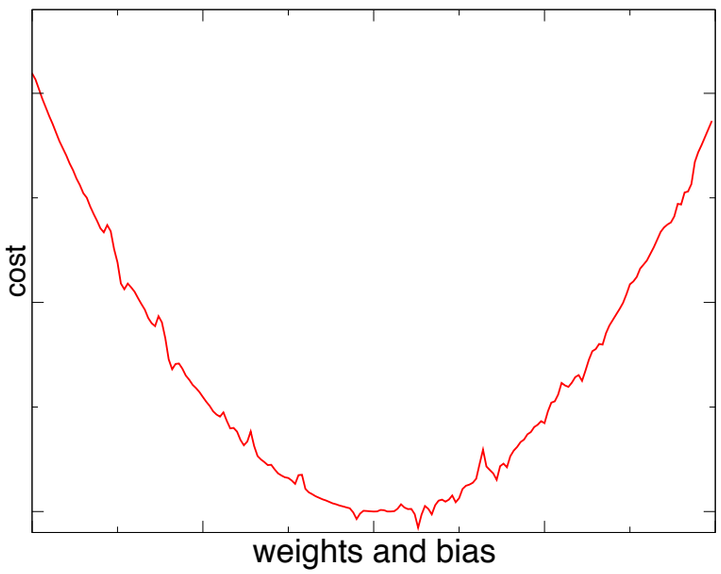
1. Stochastic gradient descend
2. Adam method, momentum

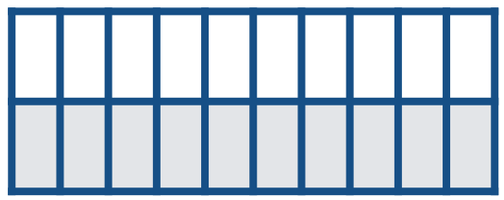
Minibatch gradient descend



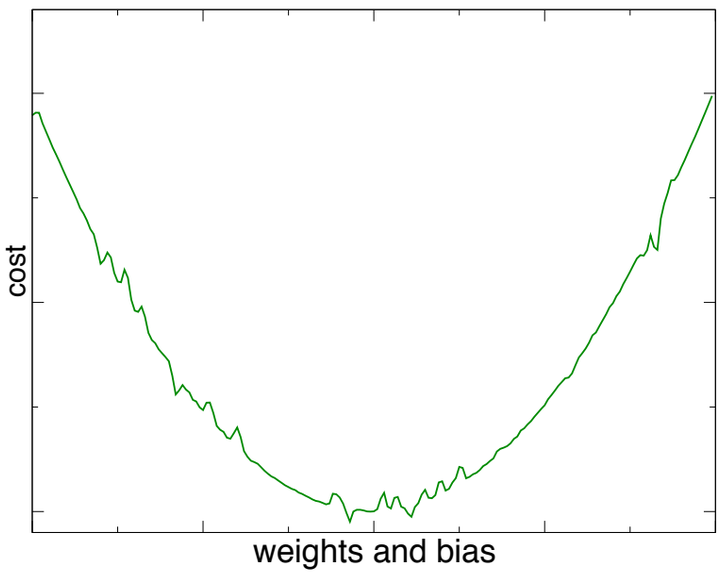
cost surfaces are different for different data sets

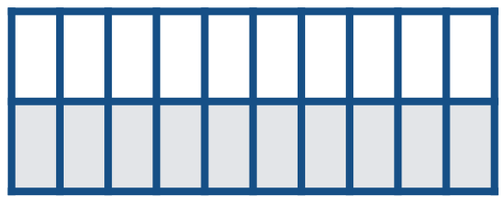




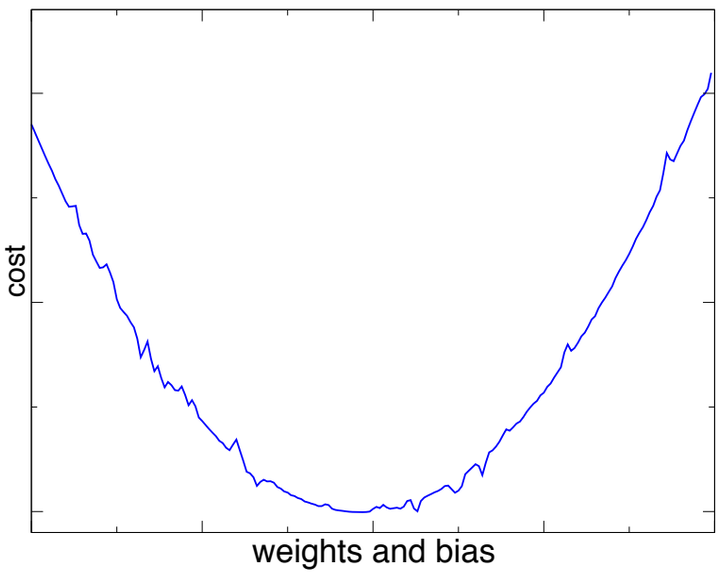


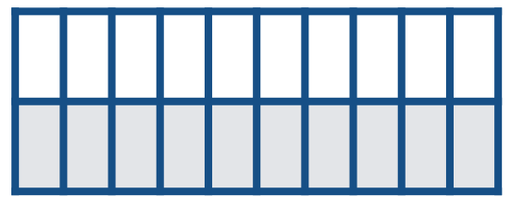
batch 2





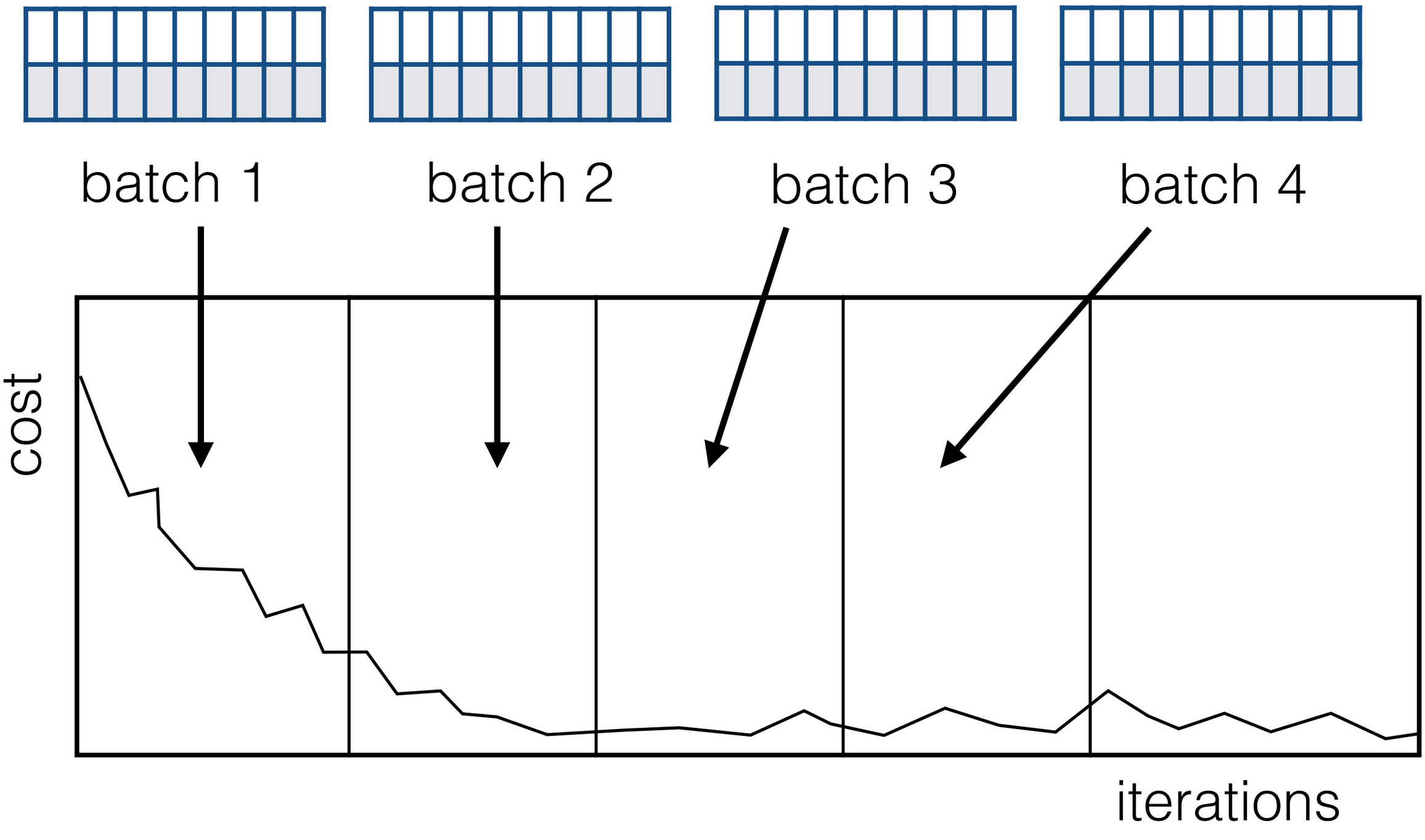
batch 3

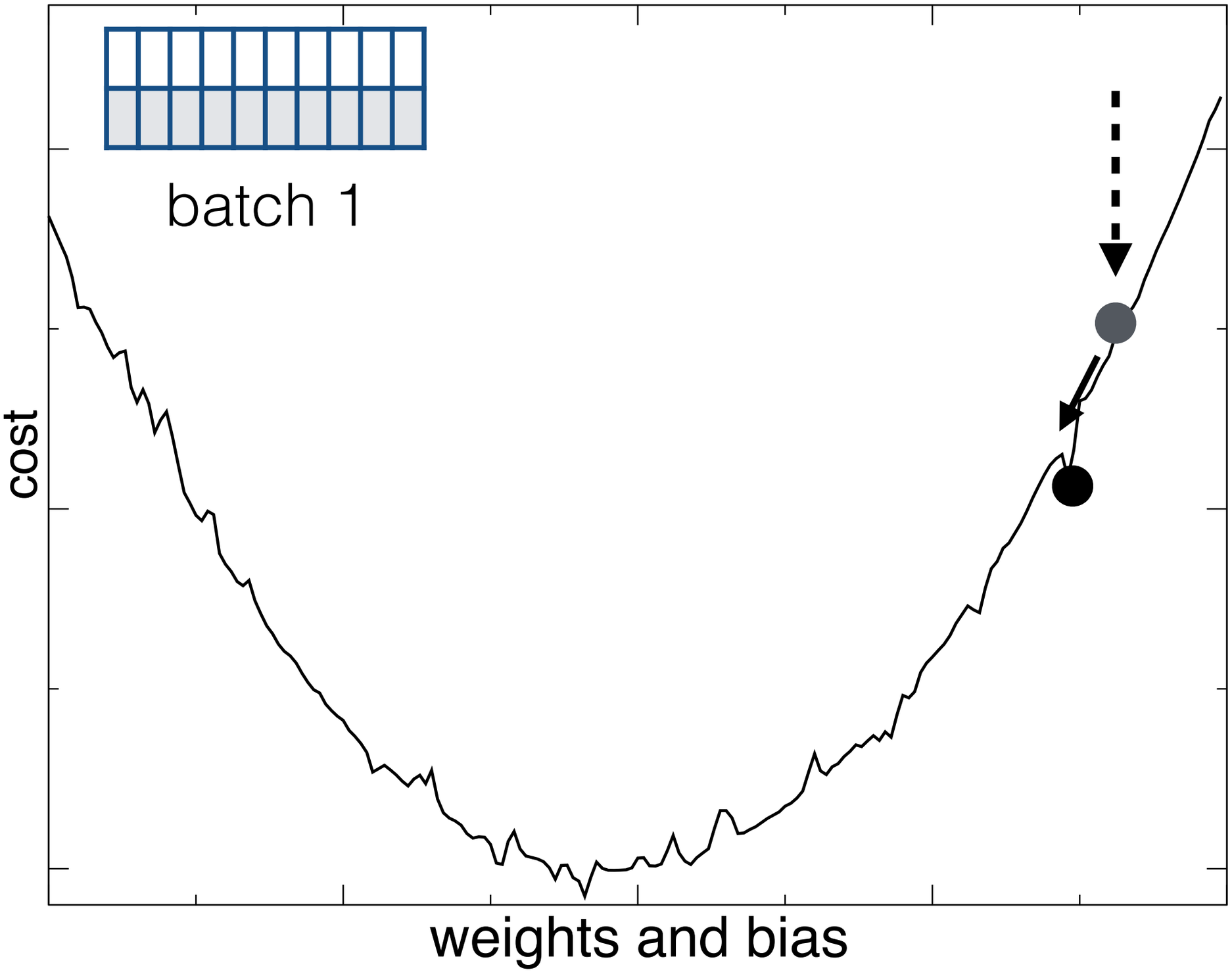


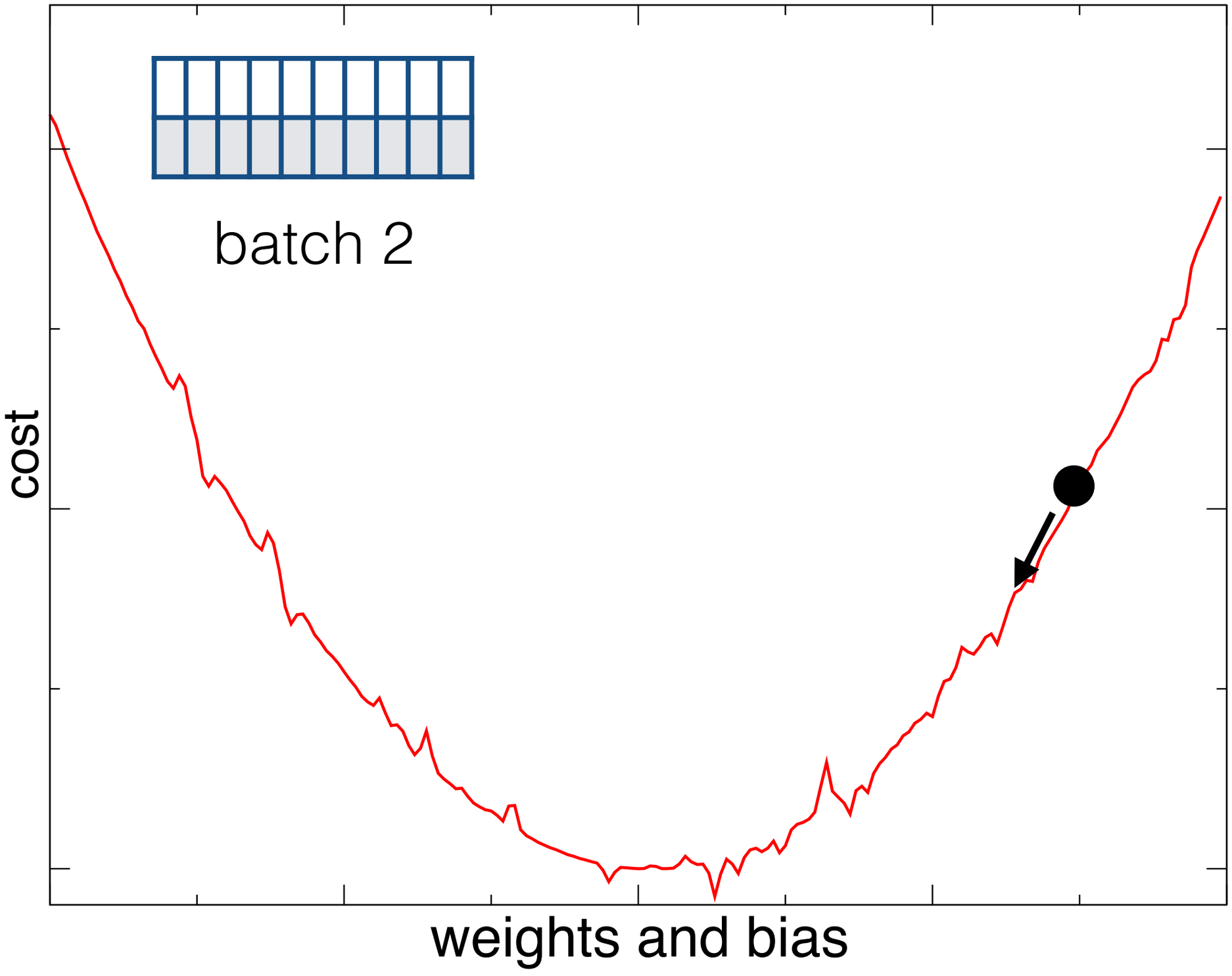


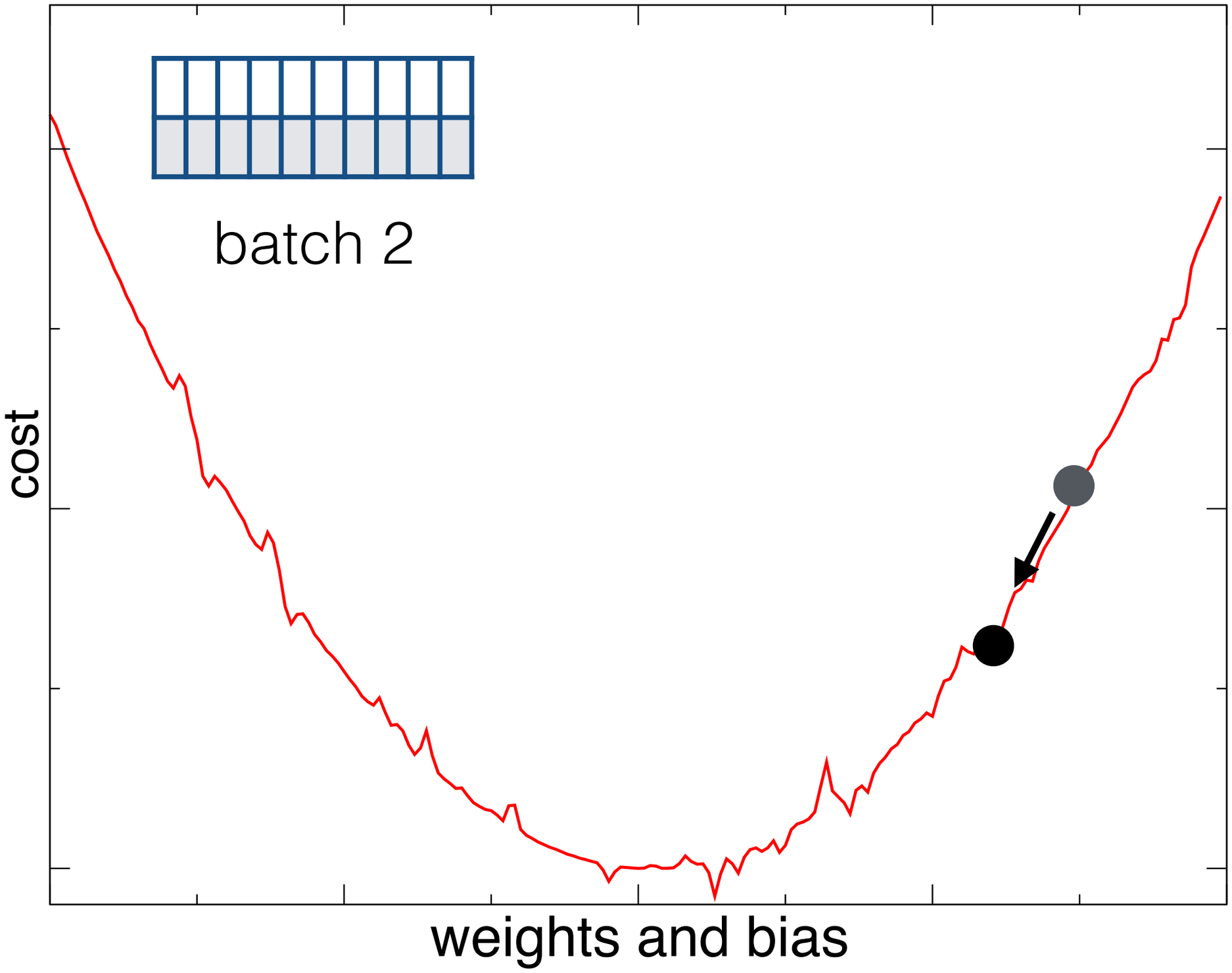
batch 4

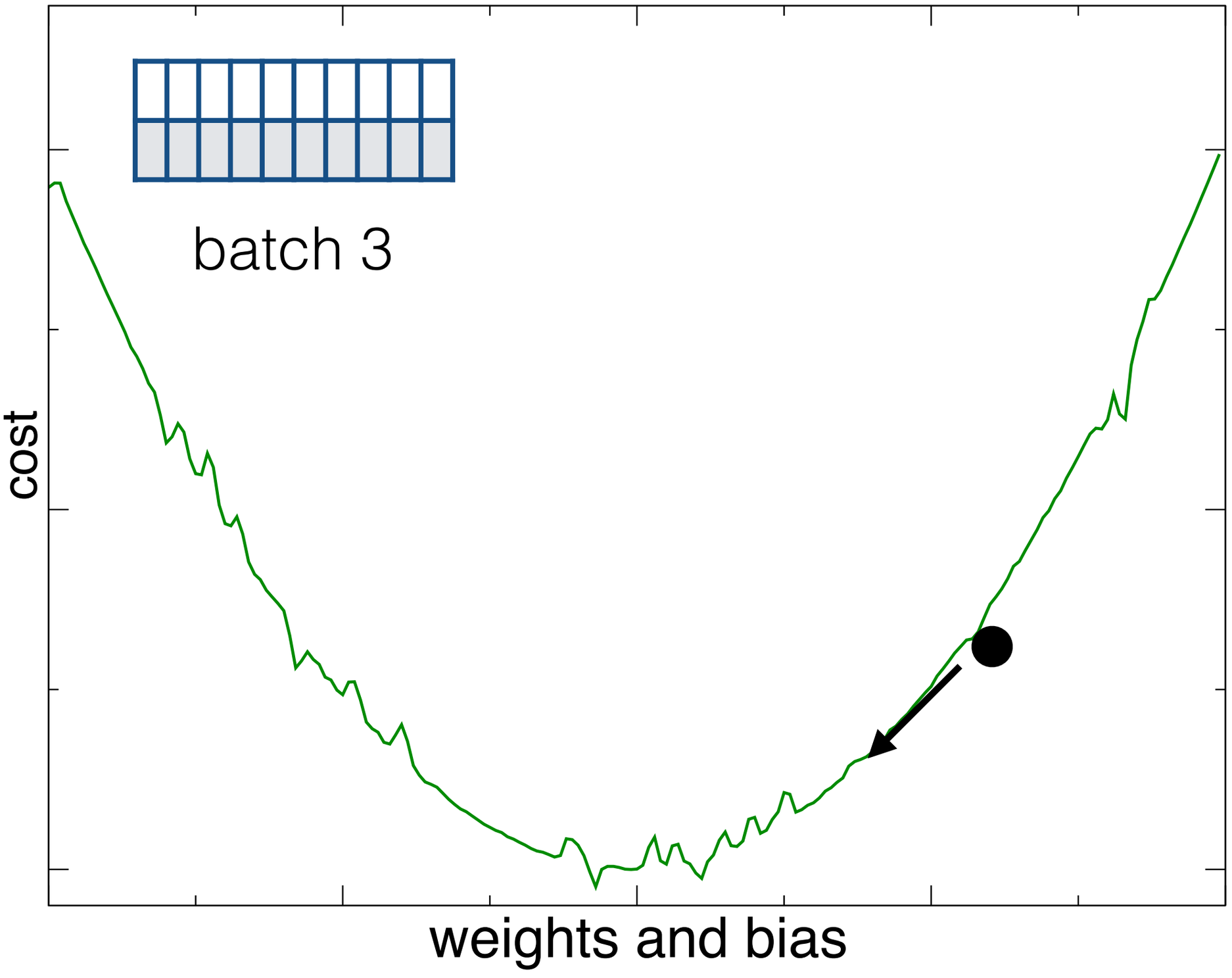
Always remember to shuffle the data

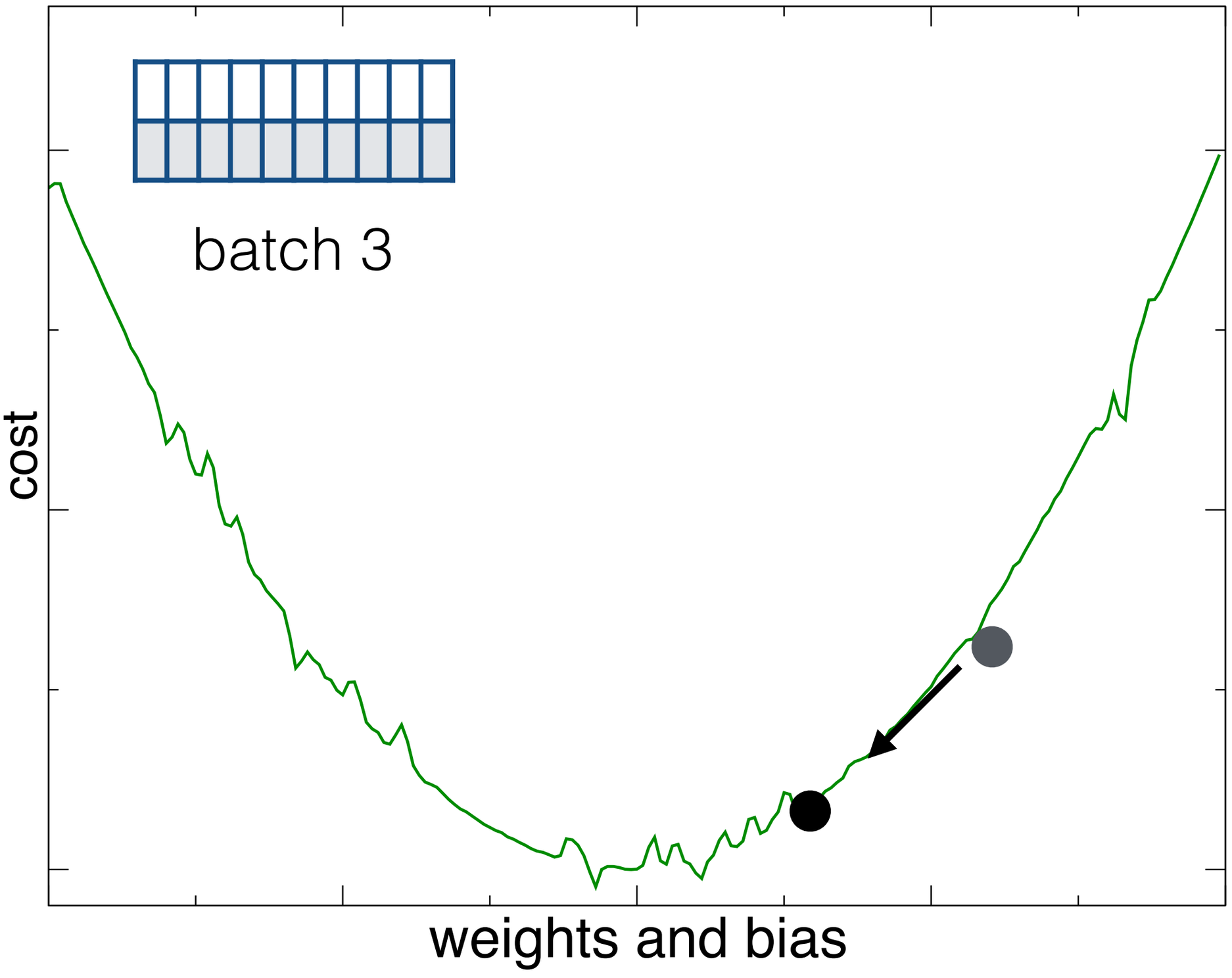


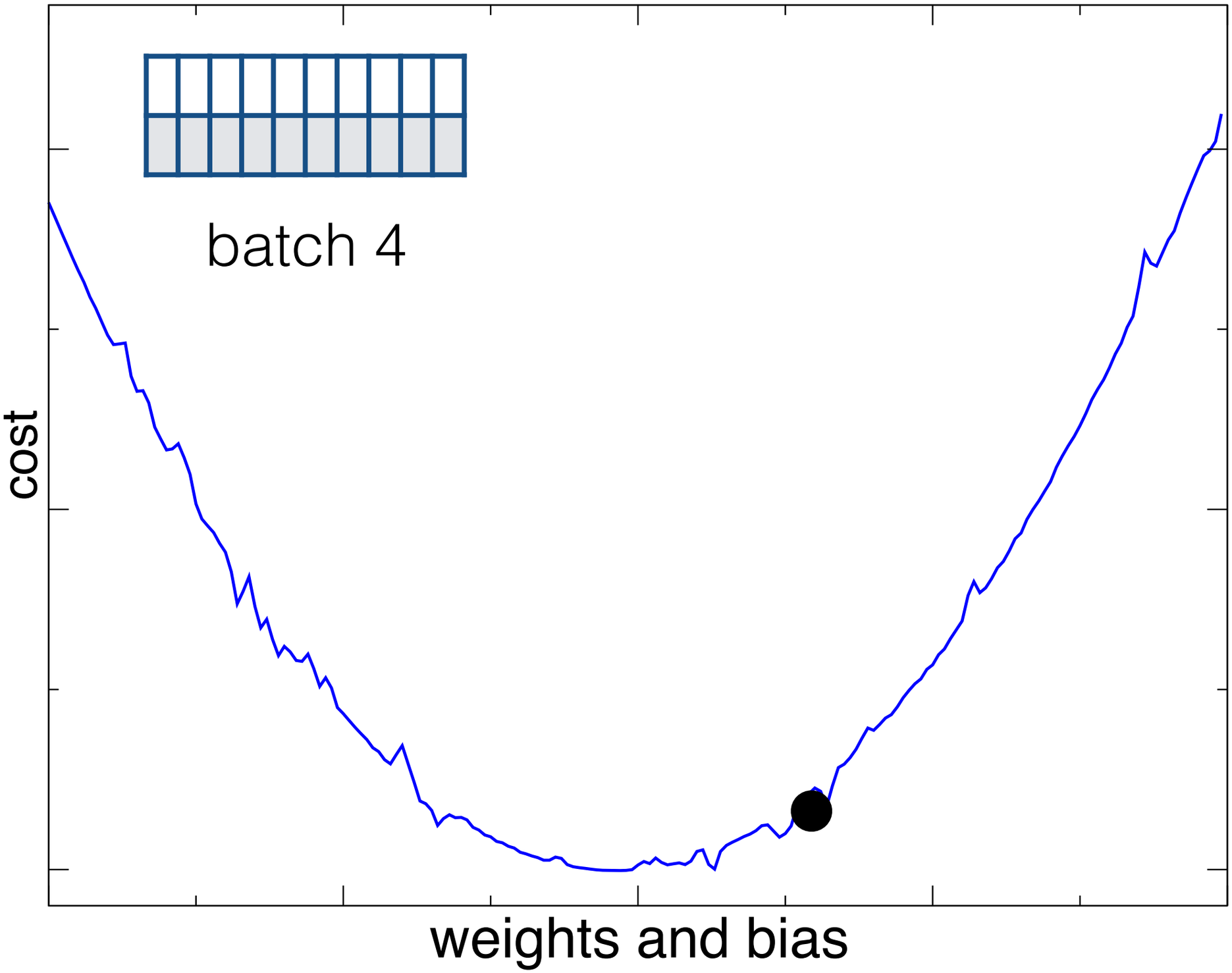

batch 1
cost
weights and bias


batch 2
cost
weights and bias

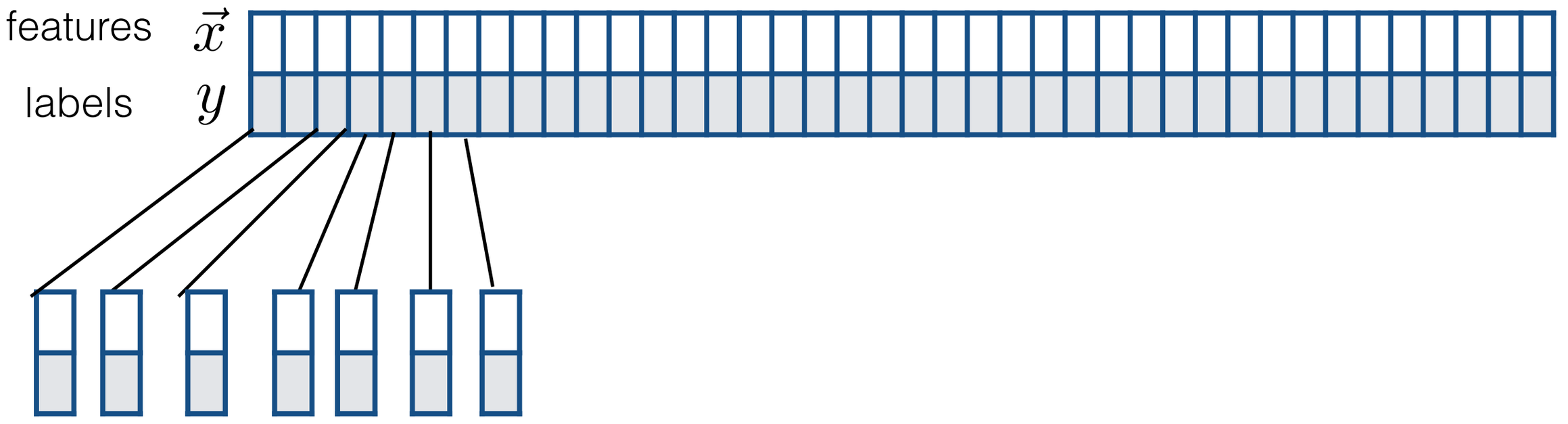

batch 2
cost
weights and bias


batch 3
cost
weights and bias


batch 3
cost
weights and bias


batch 4
cost
weights and bias

Stochastic gradient descend



use batch size = 1 for stochastic gradient descend

Adam optimisation

Published as a conference paper at ICLR 2015

ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION

Diederik P. Kingma*
University of Amsterdam
dpkingma@uva.nl

Jimmy Lei Ba*
University of Toronto
jimmy@psi.utoronto.ca

Adam optimisation

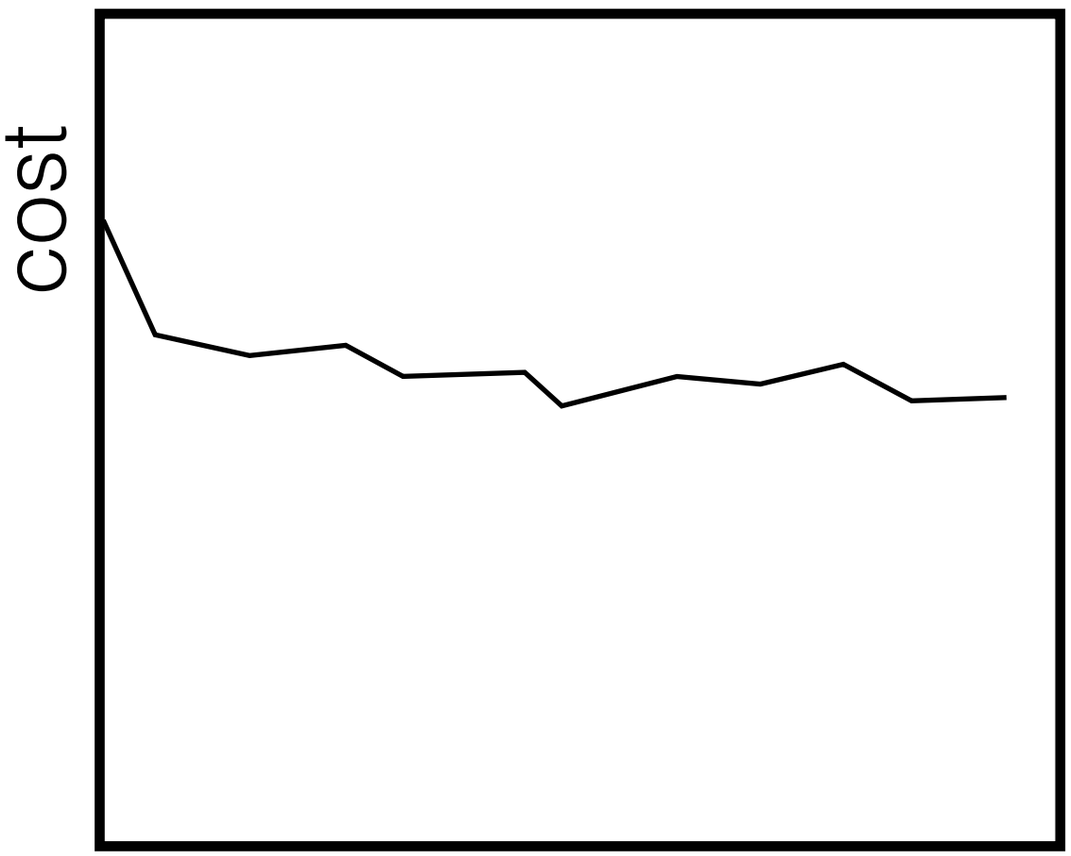
Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1st moment vector)
$v_0 \leftarrow 0$ (Initialize 2nd moment vector)
$t \leftarrow 0$ (Initialize timestep)
while θ_t not converged **do**
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
$\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
$\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
end while
return θ_t (Resulting parameters)

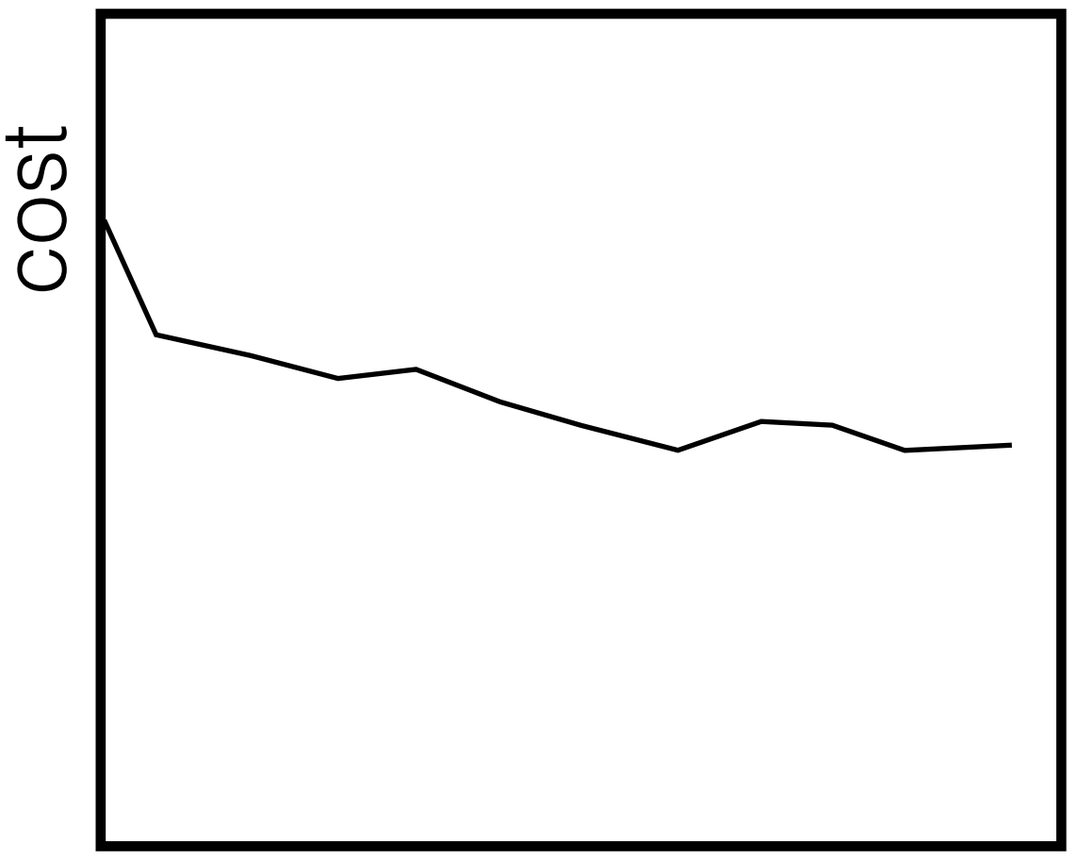
average the gradient direction over the past

move in the direction with "constant" step size

Signs of trouble
always look at cost versus iterations plots

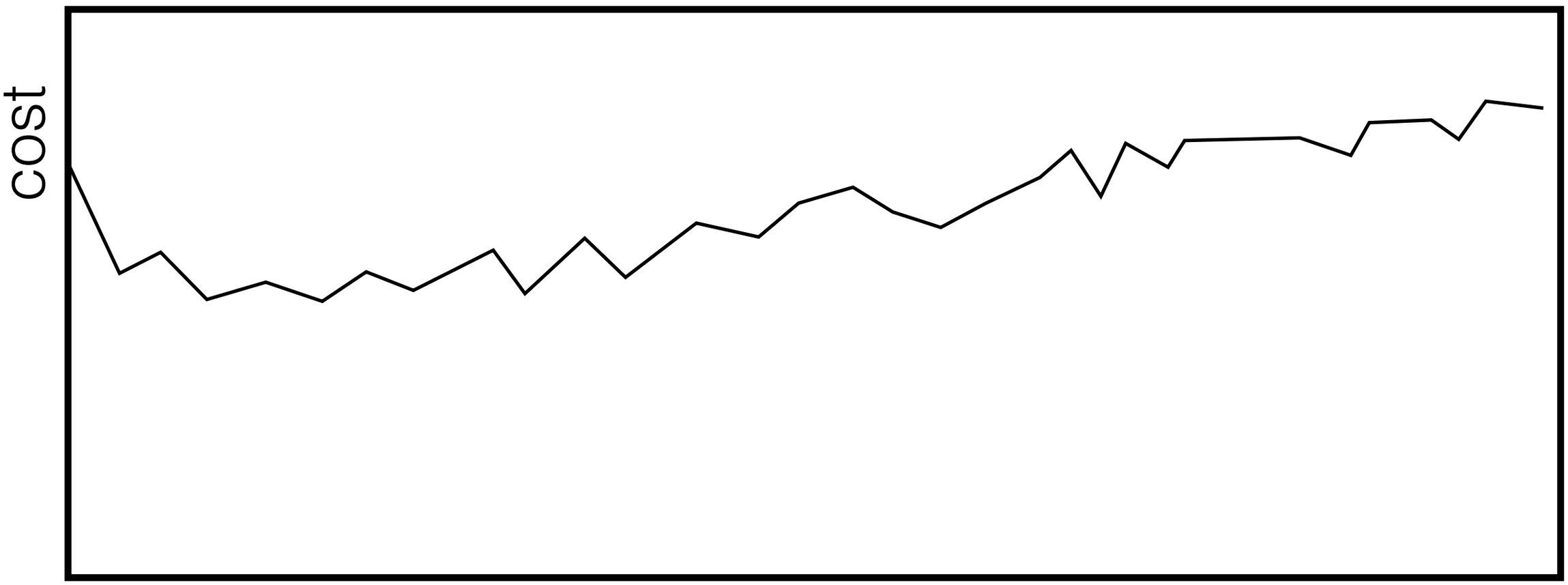


Cost not decreasing
looks like a local minimum



Cost decrease over slowly
looks like at very flat region
of cost surface

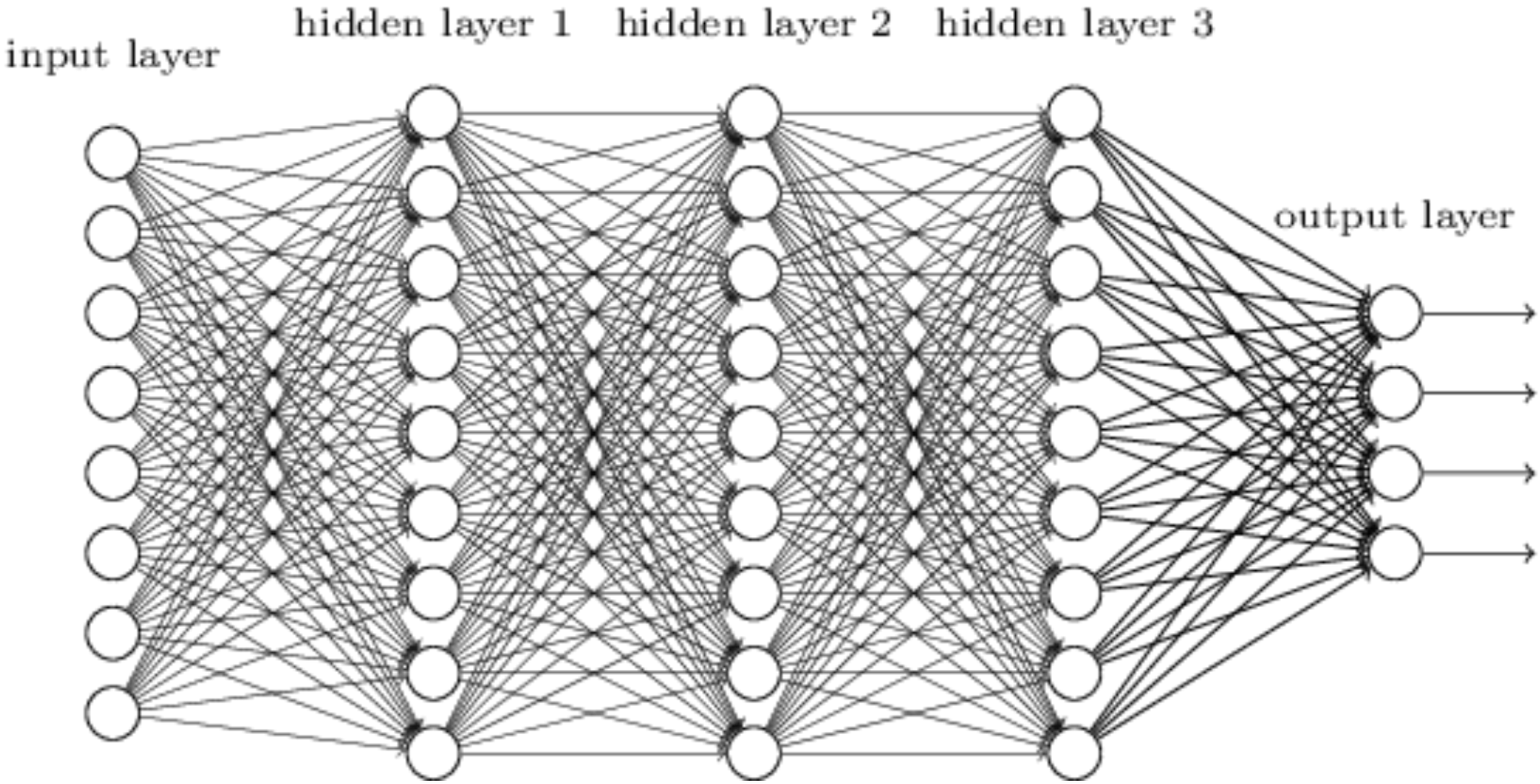
Signs of trouble
always look at cost versus iterations plots



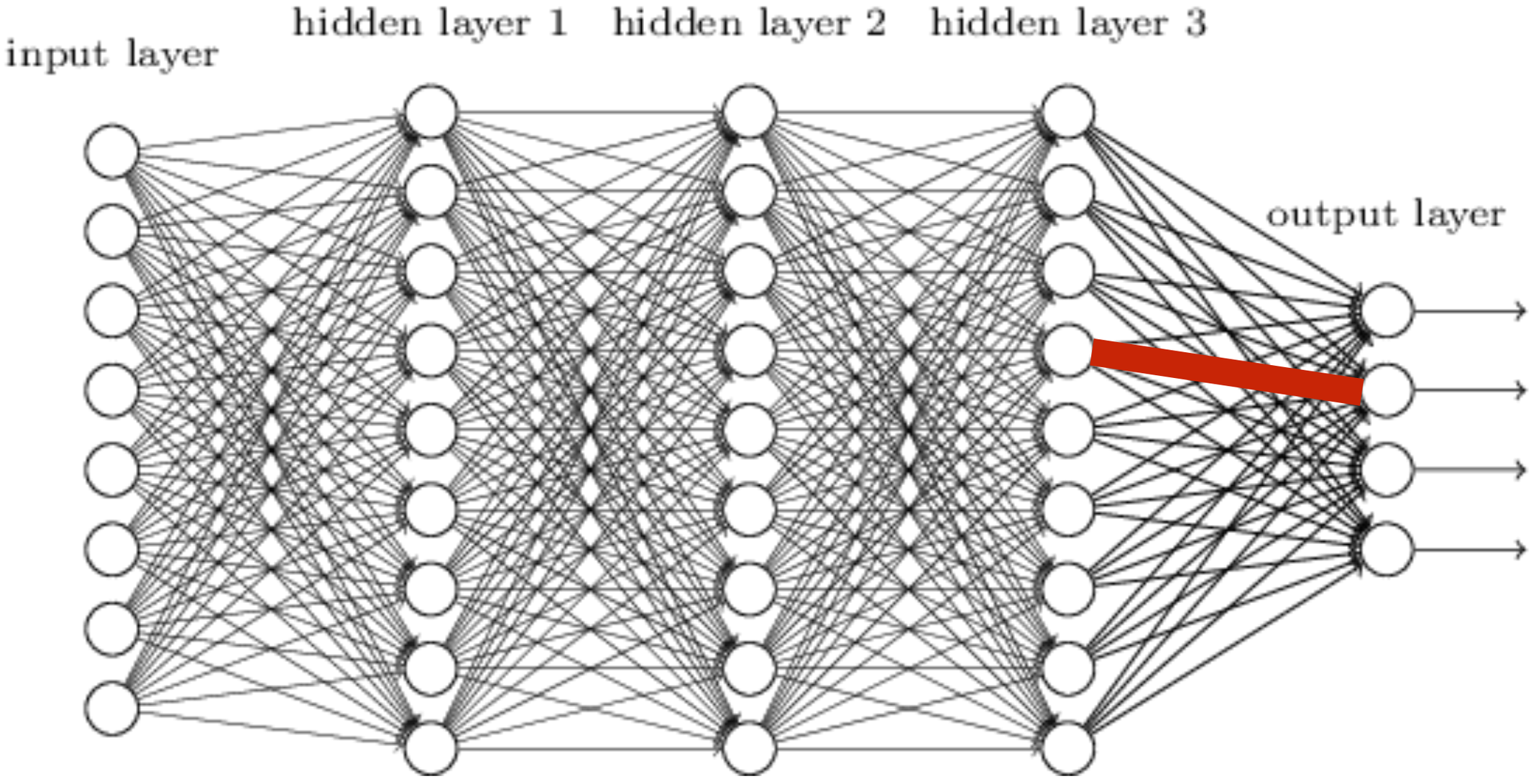
Cost actually increasing

Please check for a **bug** in your code!!

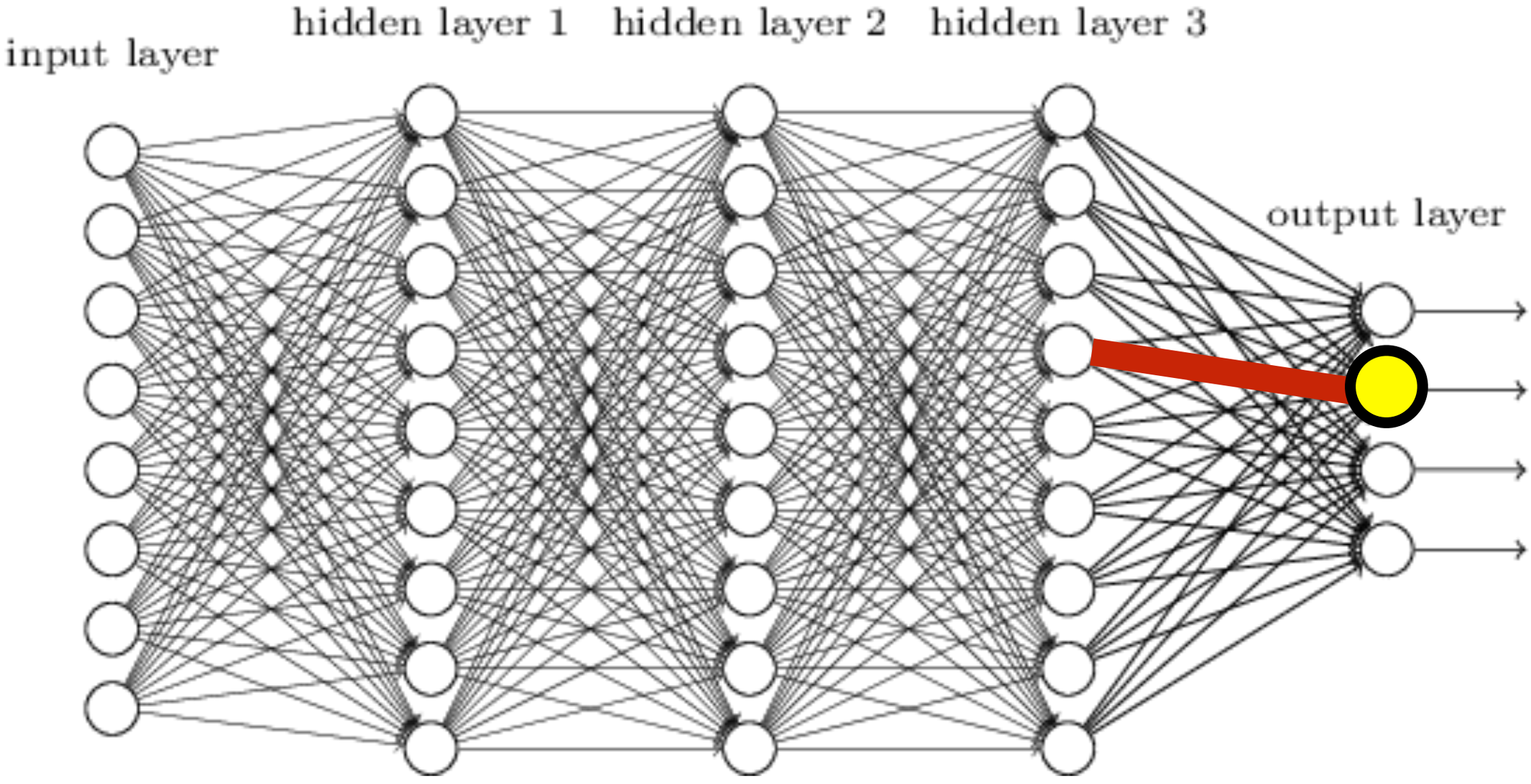
Vanishing gradient problem



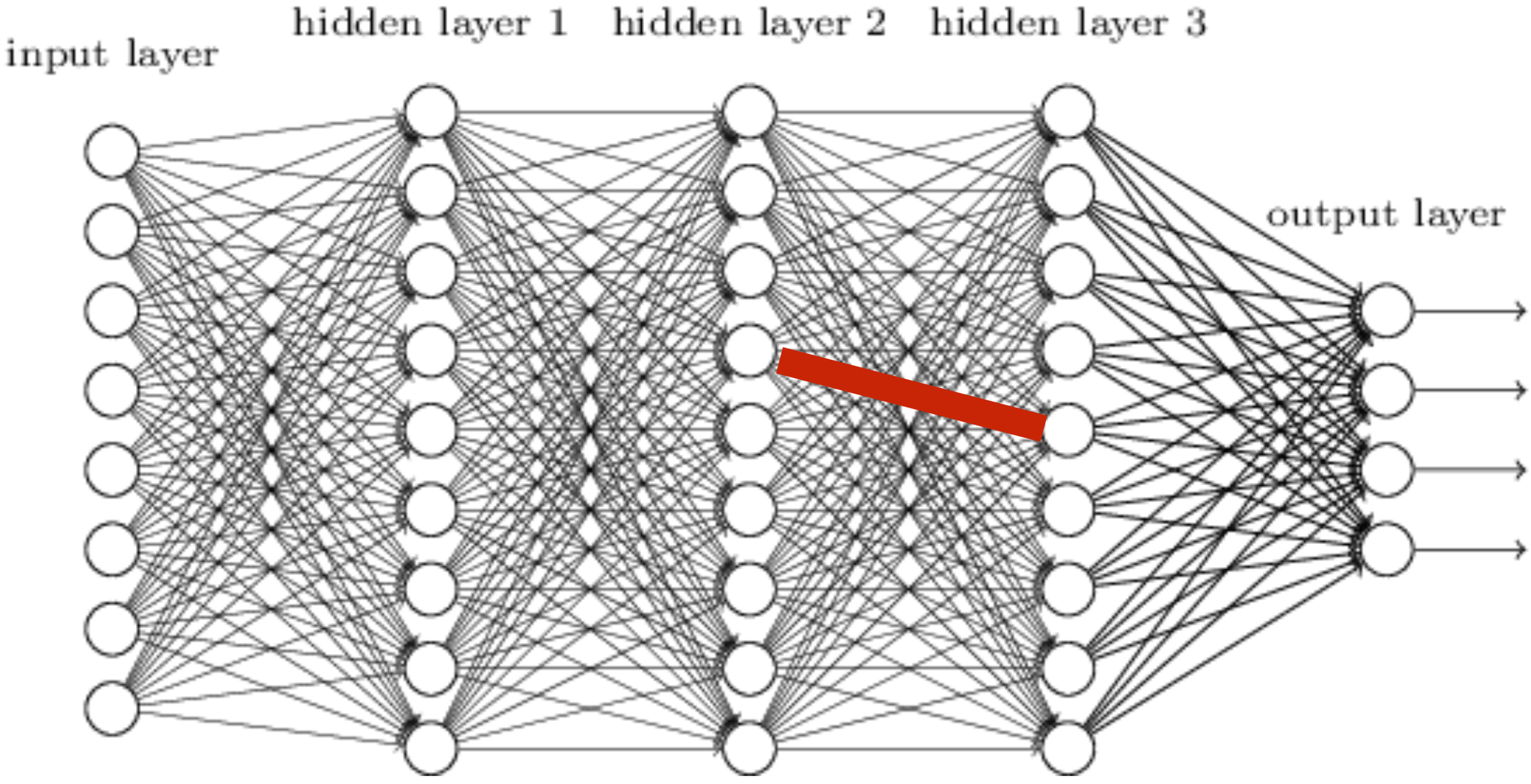
https://stats.stackexchange.com/questions/234891/difference-between-convolution-neural-network-and-deep-learning



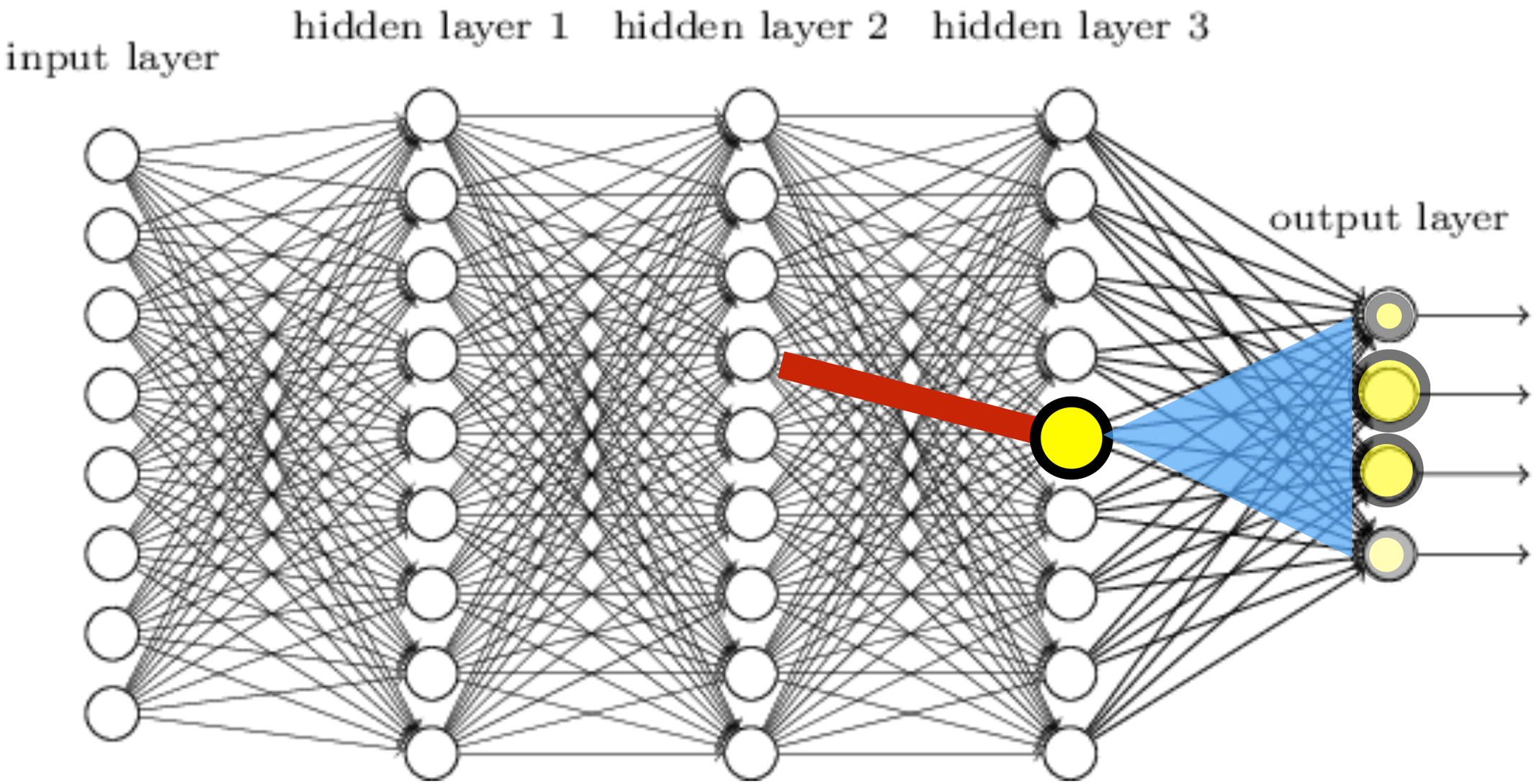
https://stats.stackexchange.com/questions/234891/difference-between-convolution-neural-network-and-deep-learning



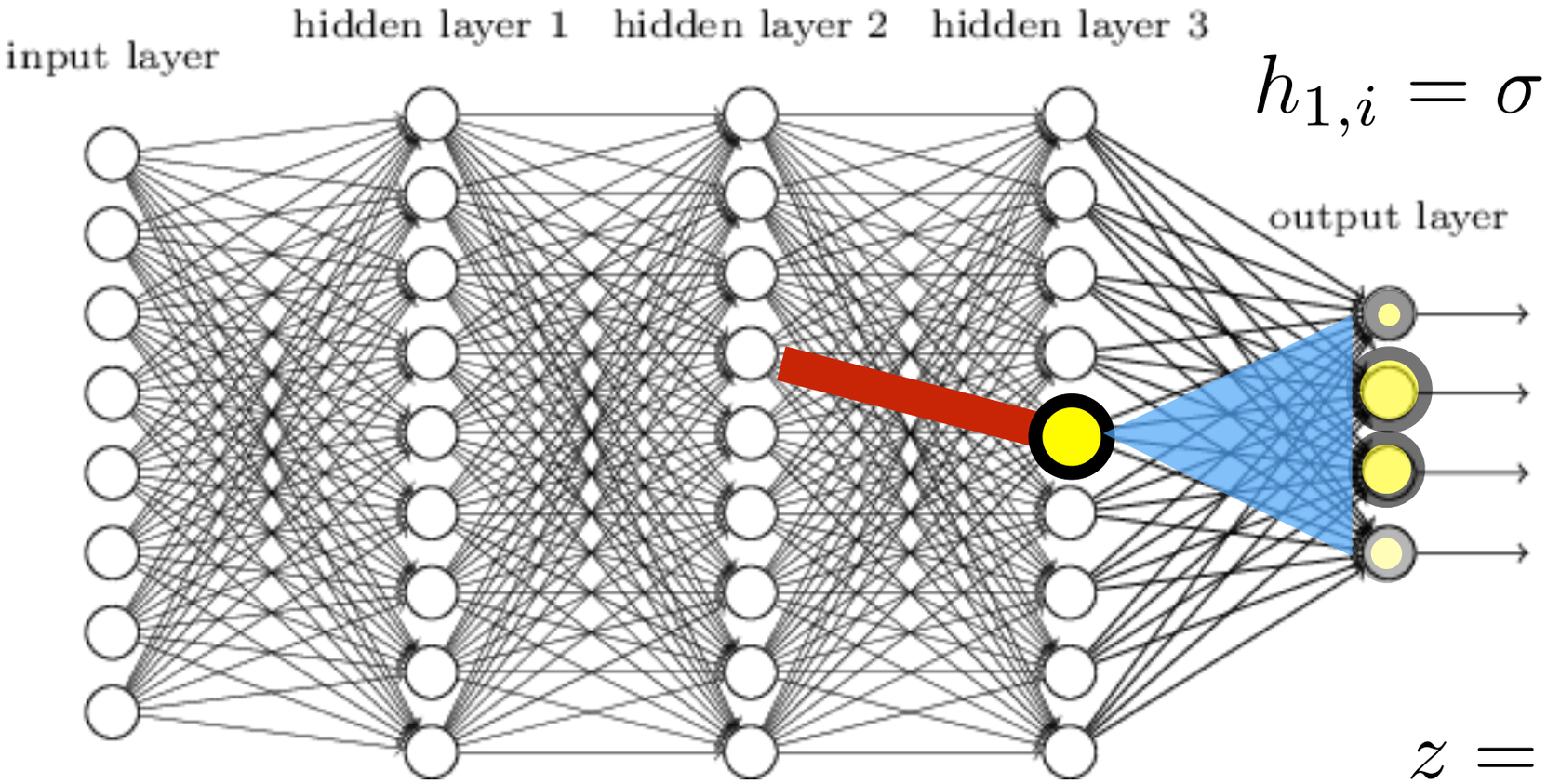
https://stats.stackexchange.com/questions/234891/difference-between-convolution-neural-network-and-deep-learning



https://stats.stackexchange.com/questions/234891/difference-between-convolution-neural-network-and-deep-learning

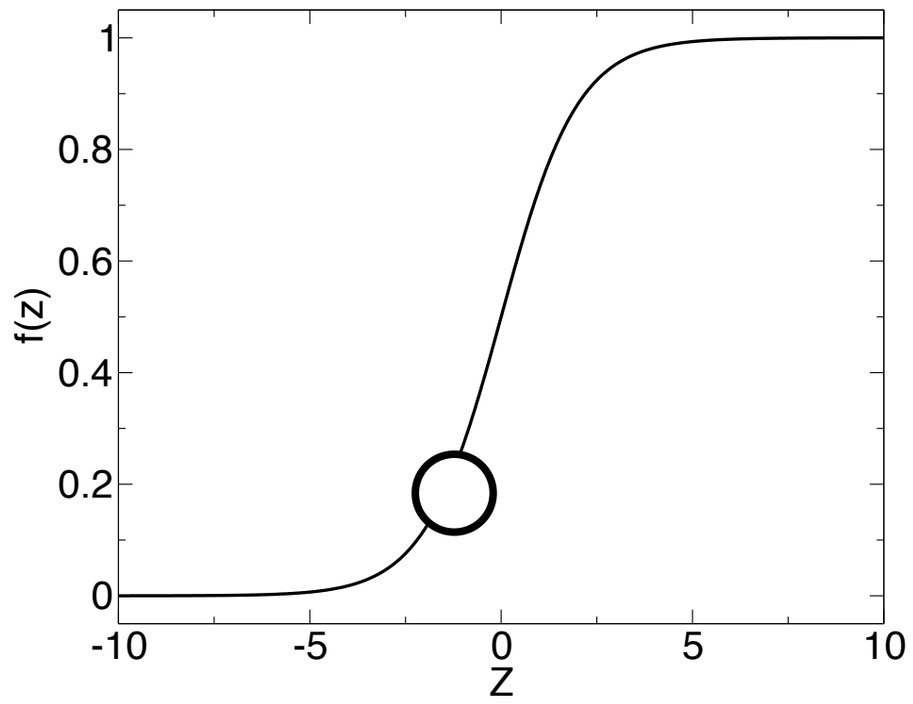


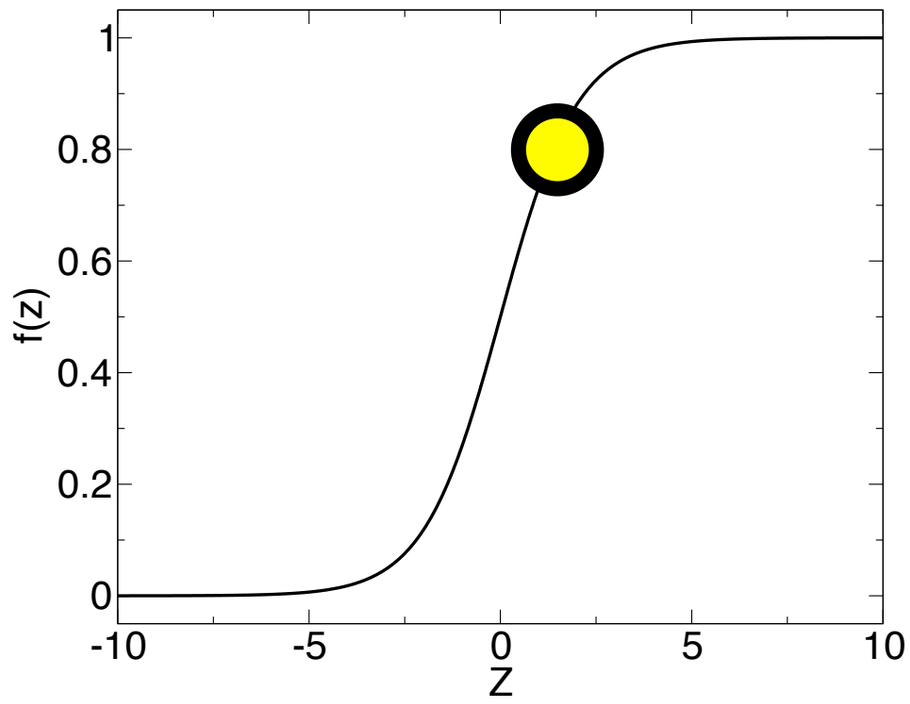
https://stats.stackexchange.com/questions/234891/difference-between-convolution-neural-network-and-deep-learning



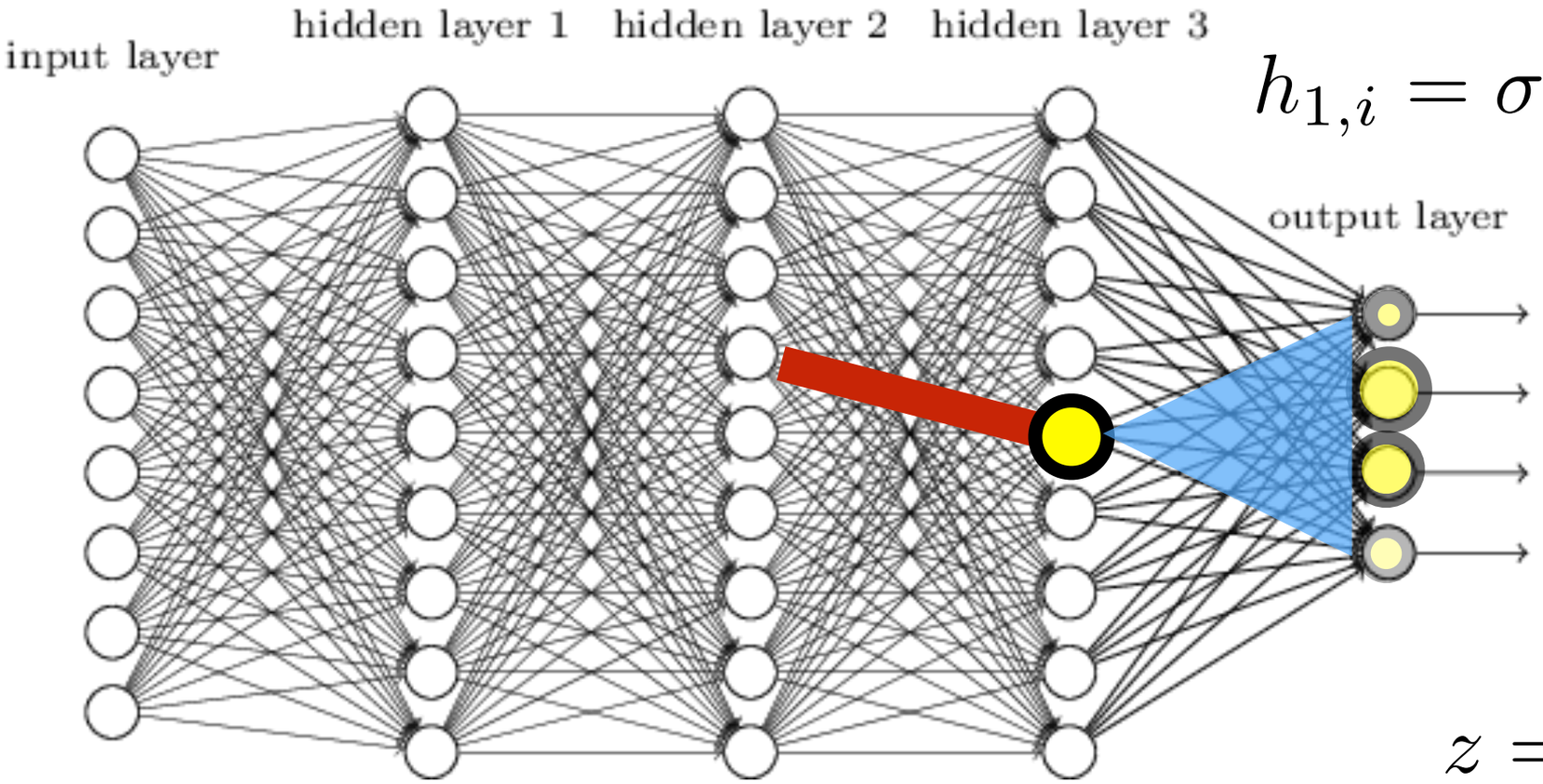
$$h_{1,i} = \sigma\left(\sum_j W_{j,i}^{(0,1)} x_j + b_i\right)$$

$$z = -2 \rightarrow z = 2$$



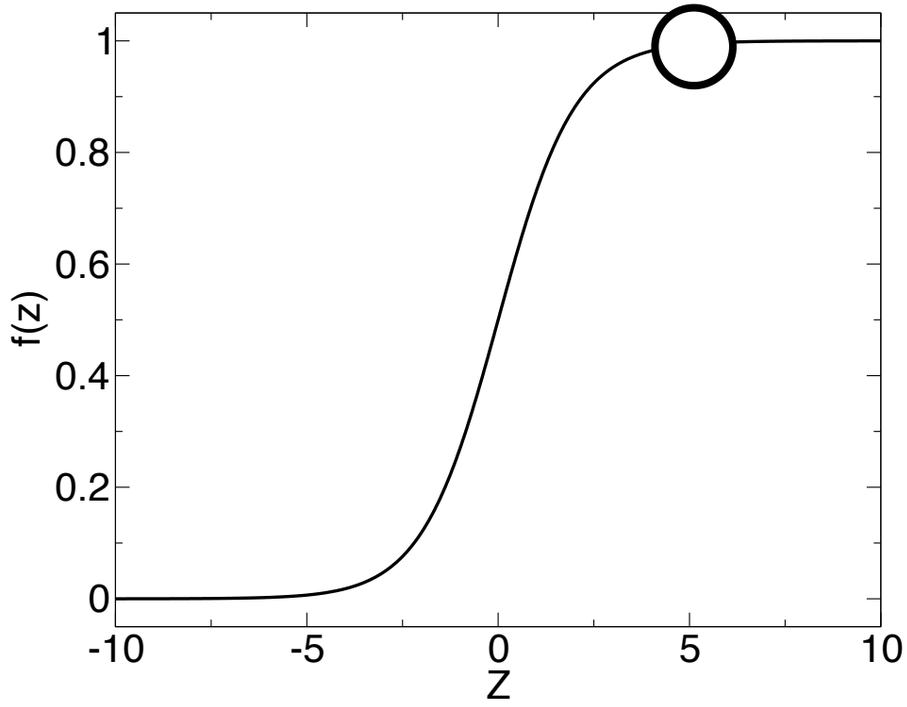


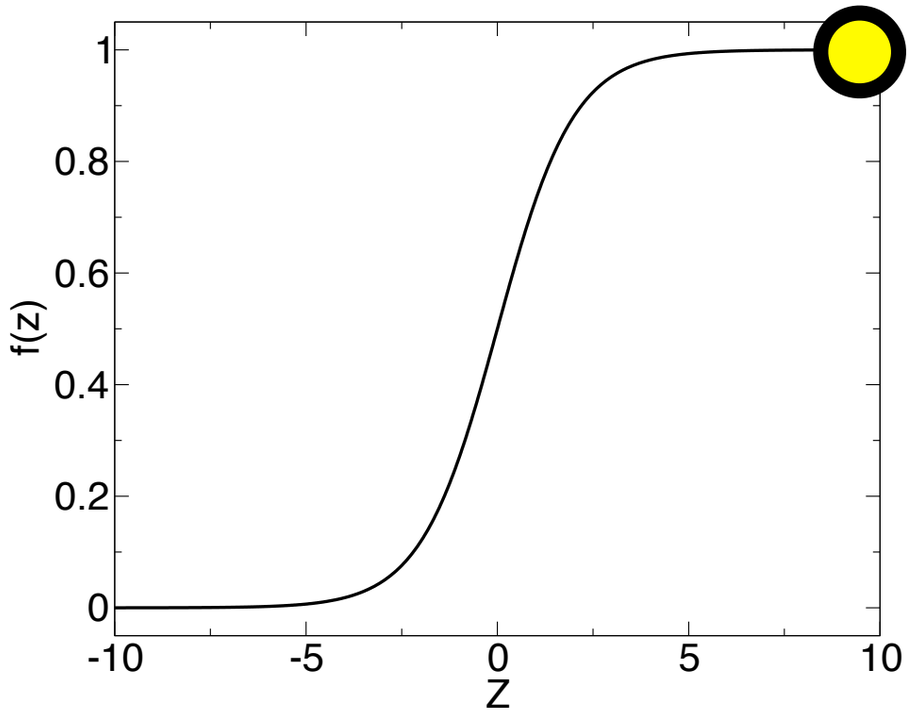
https://stats.stackexchange.com/questions/234891/difference-between-convolution-neural-network-and-deep-learning



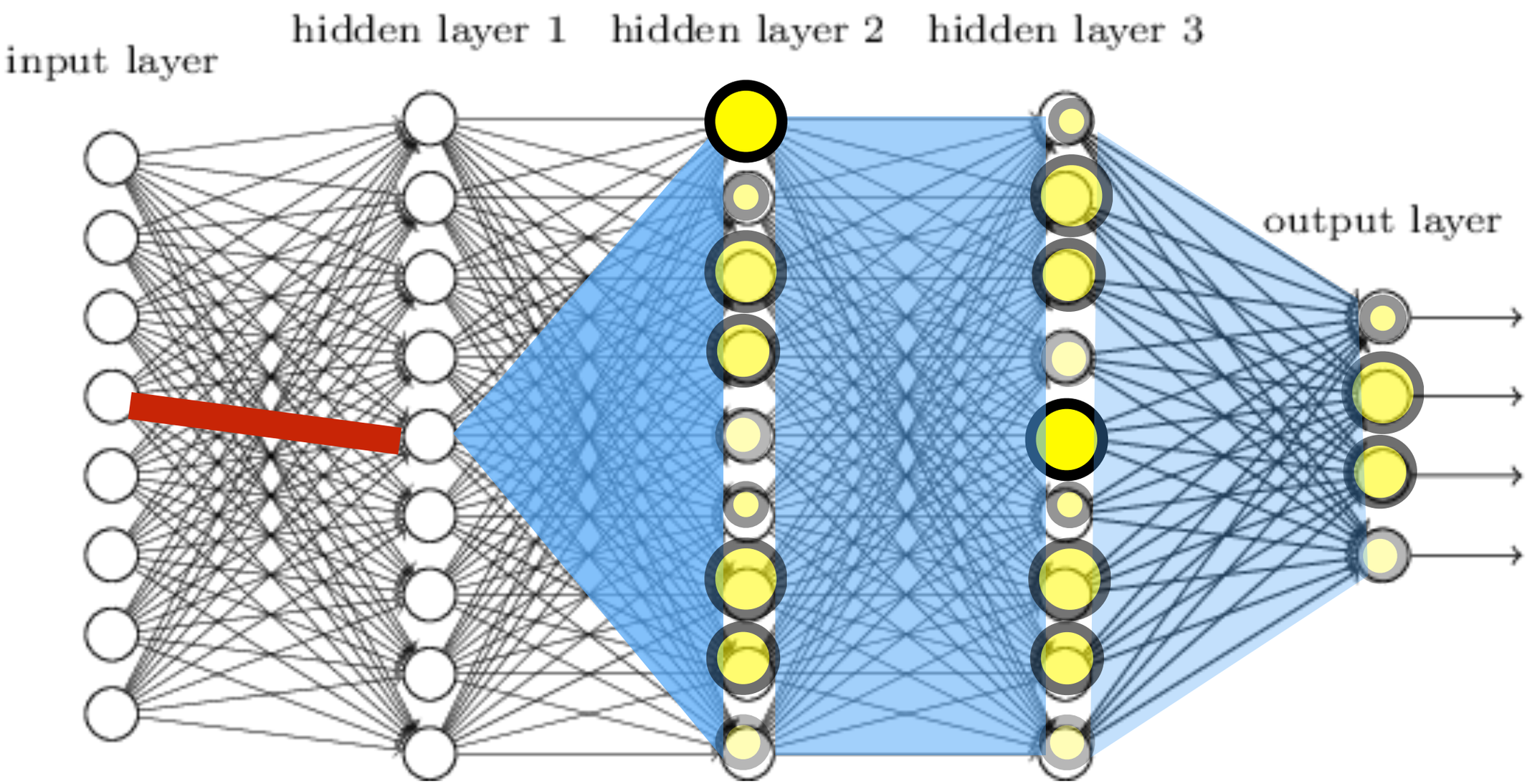
$$h_{1,i} = \sigma\left(\sum_j W_{j,i}^{(0,1)} x_j + b_i\right)$$

$$z = 5 \rightarrow z = 10$$

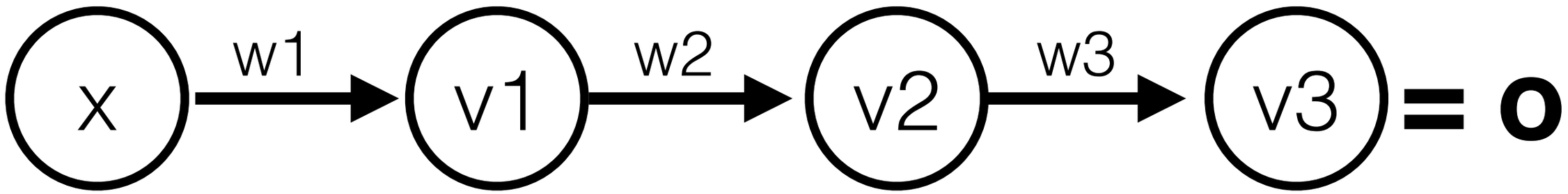




https://stats.stackexchange.com/questions/234891/difference-between-convolution-neural-network-and-deep-learning



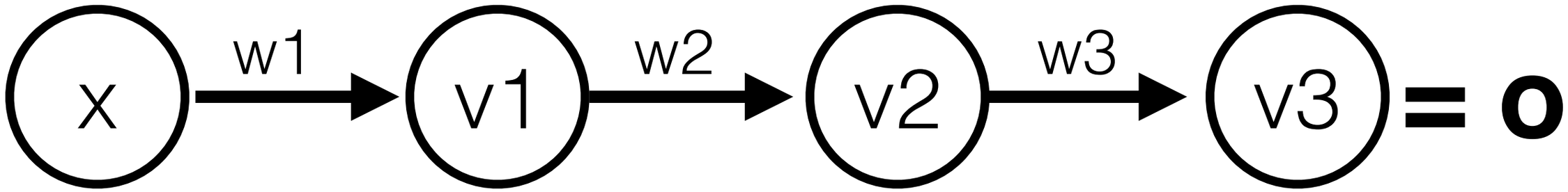
https://stats.stackexchange.com/questions/234891/difference-between-convolution-neural-network-and-deep-learning



$$\frac{\partial v_3}{\partial w_3} = \frac{\partial v_3}{\partial z_3}\frac{\partial z_3}{\partial w_3} = \sigma'(z_3)v_2$$

$$\frac{\partial v_3}{\partial w_2} = \frac{\partial v_3}{\partial z_3}\frac{\partial z_3}{\partial w_2} = \sigma'(z_3)w_3\frac{\partial v_2}{\partial w_2} = \sigma'(z_3)w_3\sigma'(z_2)v_1$$

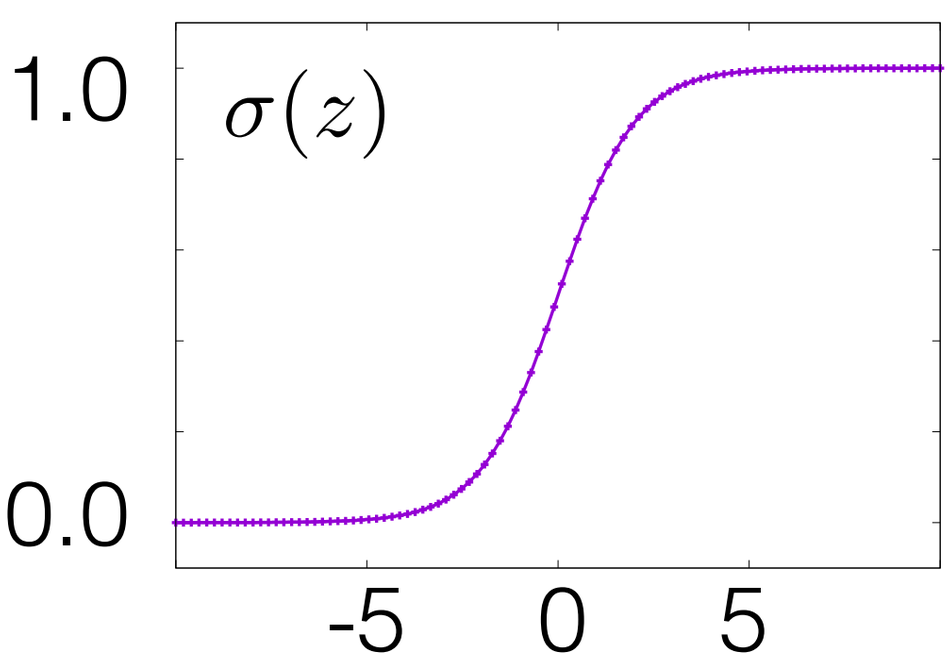
$$\begin{aligned}\frac{\partial v_3}{\partial w_1} &= \frac{\partial v_3}{\partial z_3}\frac{\partial z_3}{\partial w_1} = \sigma'(z_3)w_3\frac{\partial v_2}{\partial w_2} = \sigma'(z_3)w_3\sigma'(z_2)w_2\frac{\partial v_1}{\partial w_1} \\ &= \sigma'(z_3)w_3\sigma'(z_2)w_2\sigma'(z_1)x \\ &= \sigma'(z_3)\sigma'(z_2)\sigma'(z_1)w_3w_2x\end{aligned}$$

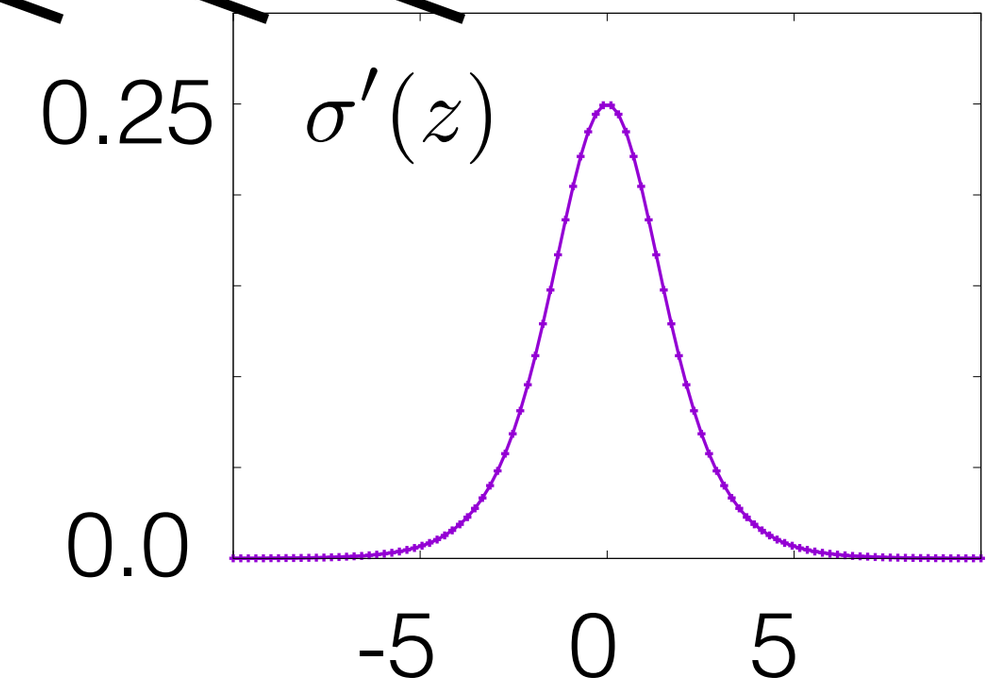


$$\frac{\partial v_3}{\partial w_1} = \frac{\partial v_3}{\partial z_3}\frac{\partial z_3}{\partial w_1} = \sigma'(z_3)w_3\frac{\partial v_2}{\partial w_2} = \sigma'(z_3)w_3\sigma'(z_2)w_2\frac{\partial v_1}{\partial w_1}$$

$$= \sigma'(z_3)w_3\sigma'(z_2)w_2\sigma'(z_1)x$$

$$= \sigma'(z_3)\sigma'(z_2)\sigma'(z_1)w_3w_2x$$





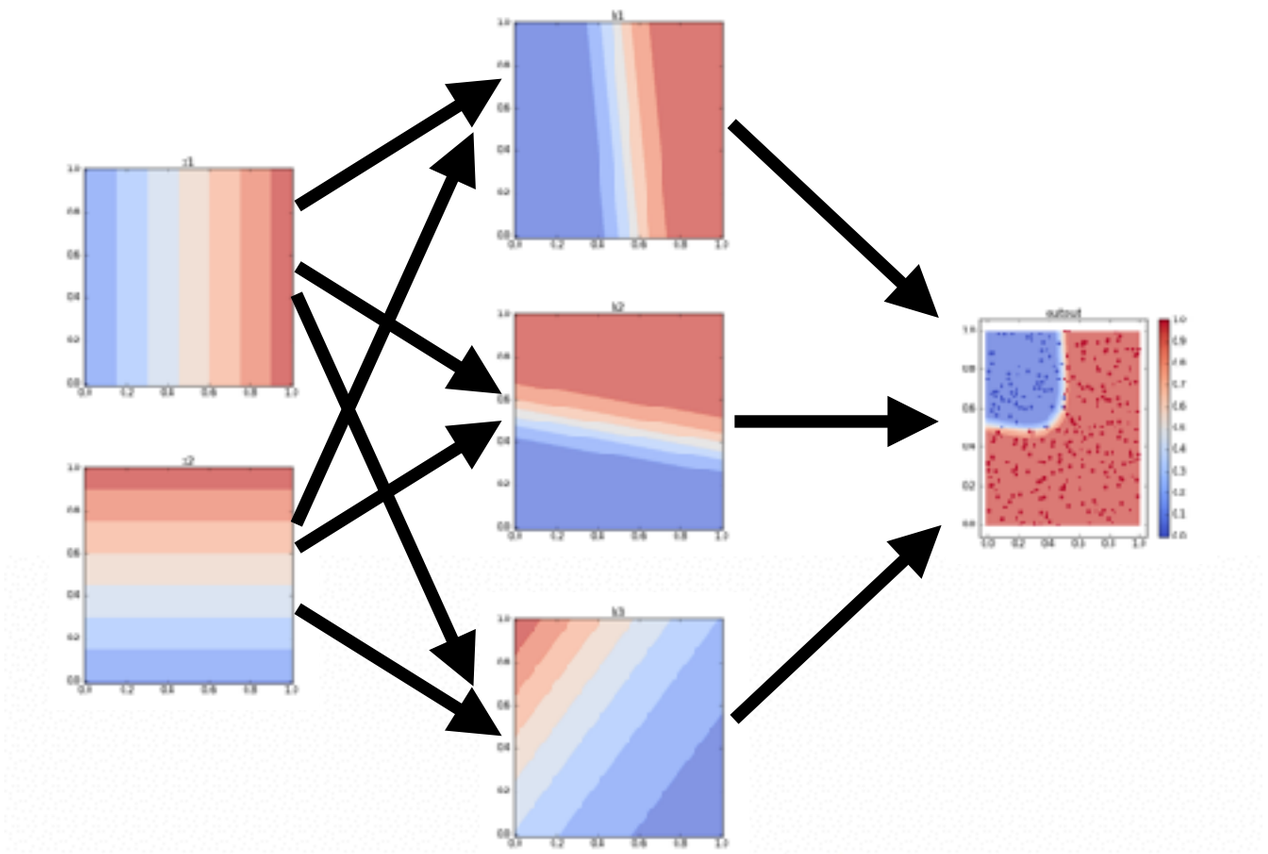
Strategies to overcoming vanishing gradient problem

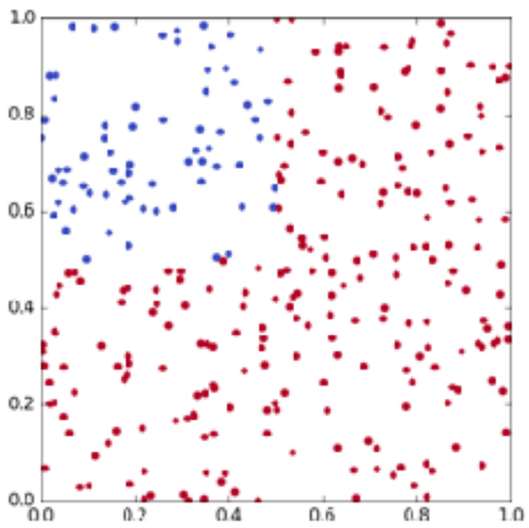
short-cuts (residual net)

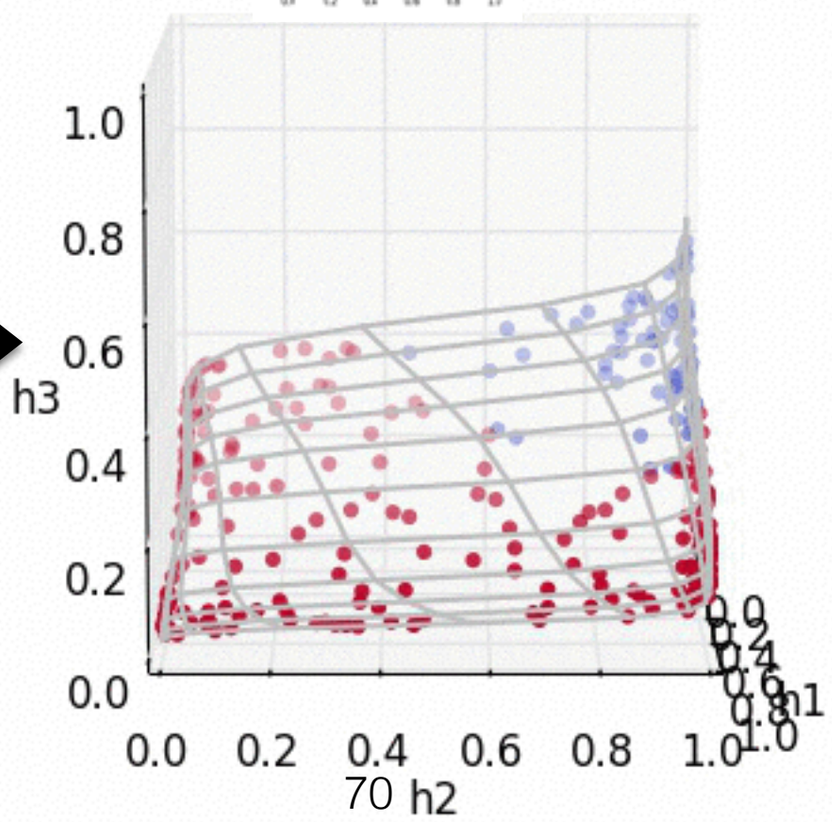
these will be covered later in the course

Question about role of bias in class last week

Question about role of bias in class last week

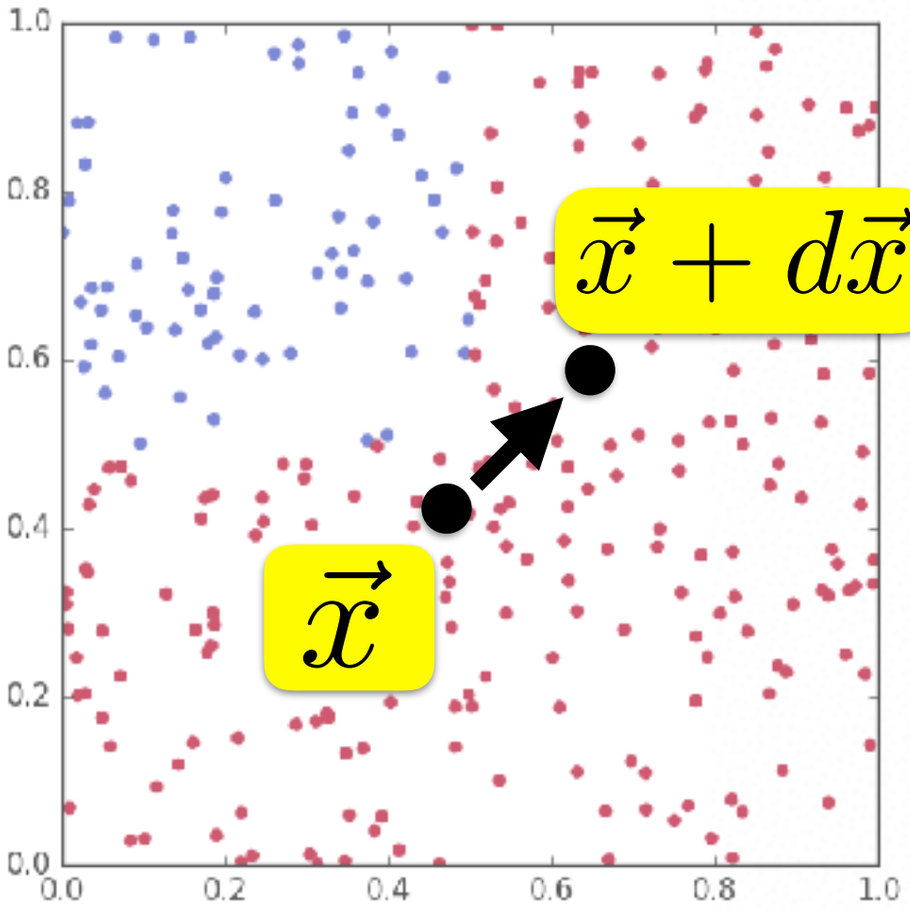


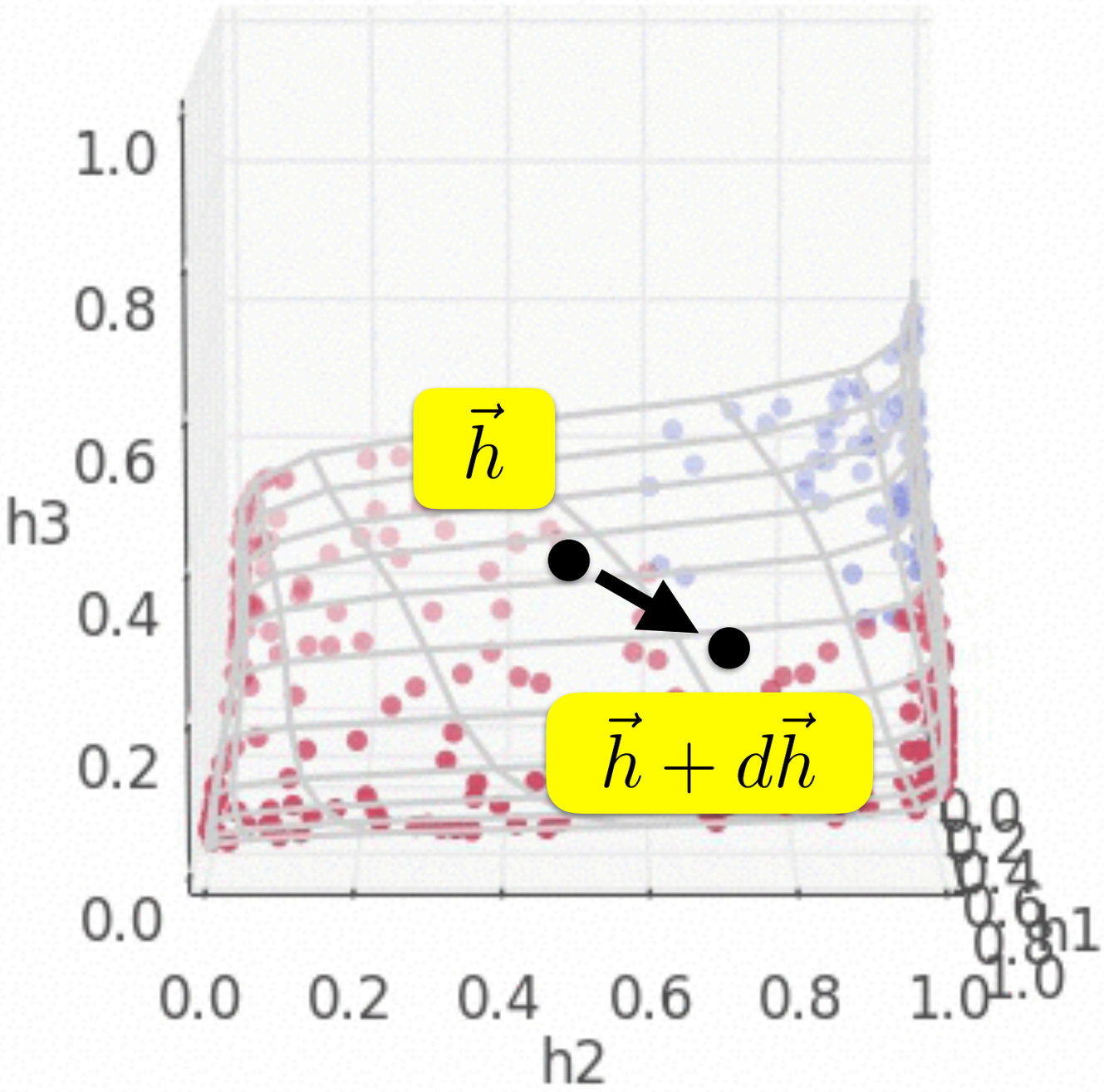




$$h_i = \sigma(\vec{w}_i \cdot \vec{x} + b_i)$$

$$i = 1, 2, 3 \qquad \vec{h} = (h_1, h_2, h_3)$$





Question:

Does change in bias changes the length of vector $d\vec{h}$?

$$h_i = \sigma(z_i) = \sigma(\vec{w}_i \cdot \vec{x} + b_i) \qquad \vec{h} = (h_1, h_2, h_3)$$
$$i = 1, 2, 3$$

$$dh_i = \sigma'(z_i)\,[\vec{w}_i \cdot d\vec{x}] \qquad \|d\vec{h}\|^2 = \sum_i \sigma'(z_i)^2\,[\vec{w}_i \cdot d\vec{x}]^2$$

Now change bias to $\tilde{b}_i = b_i + \Delta b_i$ to get $\tilde{z}_i = z_i + \Delta b_i$

Now new vector

$$d\tilde{h}_i = \sigma'(\tilde{z}_i)\,[\vec{w}_i \cdot d\vec{x}] = \sigma'(z_i + \Delta b_i)\,[\vec{w}_i \cdot d\vec{x}]$$

In general $\sigma'(z_i + \Delta b_i) \neq \sigma'(z_i)$

Therefore the length of vector $d\vec{h}$ changes when bias changes