Welcome to

BS6207

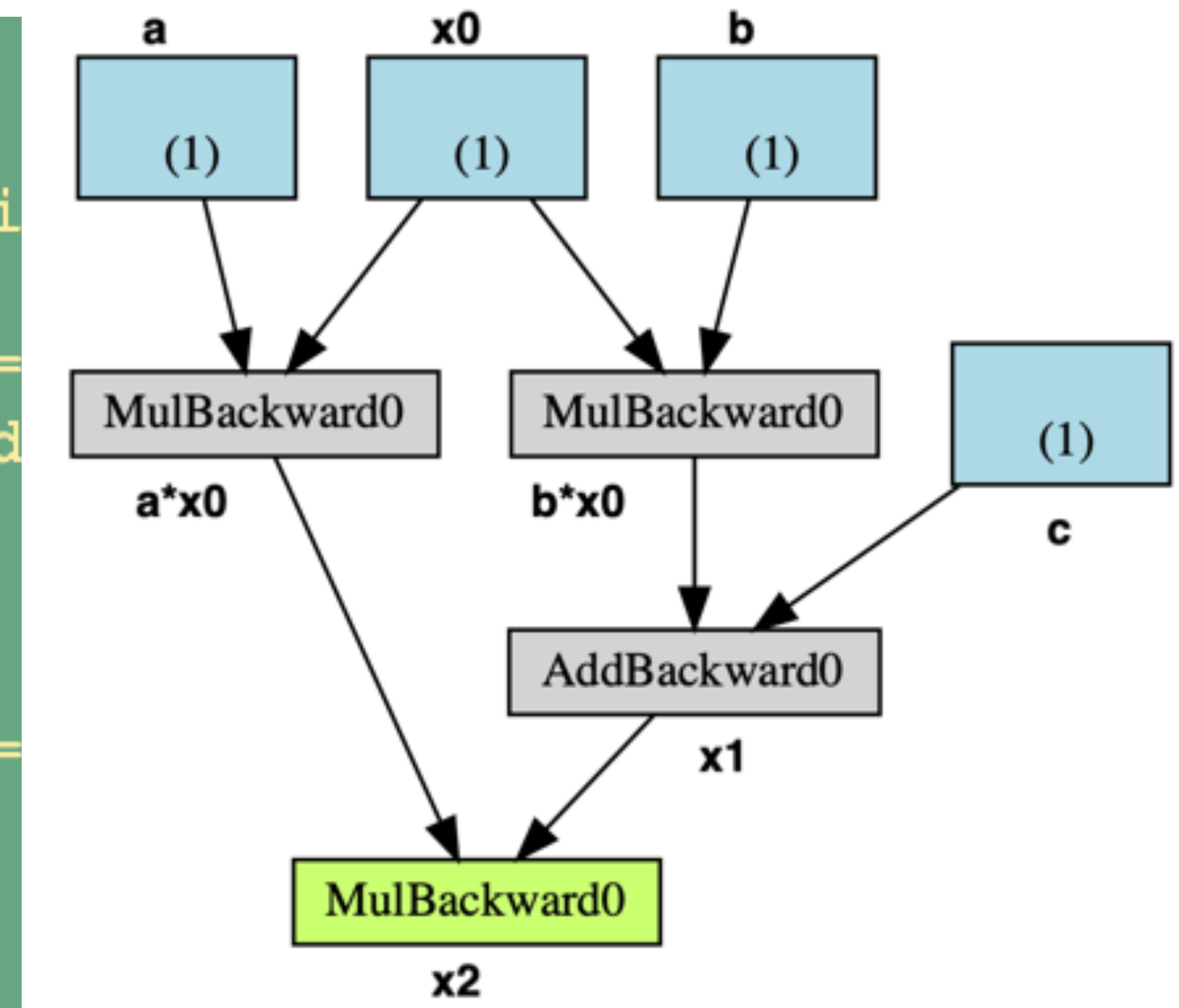
2022

Lee Hwee Kuan

Back propagation and gradient descend

```python
import torch
from torchviz import make_dot
from torch.autograd import Variable

# =======================================================
def print_compute_tree(name,node):
    dot = make_dot(node)
    #print(dot)
    dot.render(name)
# =======================================================

if __name__=='__main__':

    a  = Variable(torch.tensor([1.0]),requires_grad=True)
    b  = Variable(torch.tensor([1.0]),requires_grad=True)
    c  = Variable(torch.tensor([1.0]),requires_grad=True)
    x0 = Variable(torch.tensor([.5]),requires_grad=True)

    # updater step - first step
    x1 = b*x0 + c
    x2 = a*x0*x1 # one step

    print_compute_tree('tree_ex' ,x2)
```

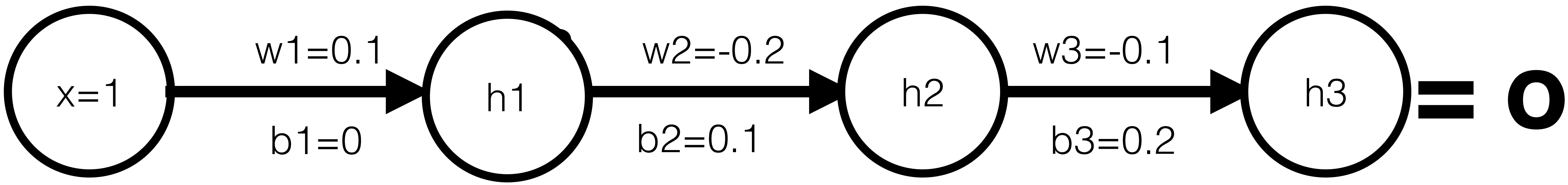
```
import torch
from torchviz import make_dot
from torch.autograd import Vari

# ==============================
def print_compute_tree(name,nod
    dot = make_dot(node)
    #print(dot)
    dot.render(name)
# ==============================

if __name__=='__main__':

    a  = Variable(torch.tensor([1.0]),requires_grad=True)
    b  = Variable(torch.tensor([1.0]),requires_grad=True)
    c  = Variable(torch.tensor([1.0]),requires_grad=True)
    x0 = Variable(torch.tensor([.5]),requires_grad=True)

    # updater step - first step
    x1 = b*x0 + c
    x2 = a*x0*x1 # one step

    print_compute_tree('tree_ex' ,x2)
```



Forward pass

$$z_1 = w_1 x + b_1 \qquad z_2 = w_2 h_1 + b_2 \qquad z_3 = w_3 h_2 + b_3$$

$$h_1 = \sigma(z_1) \qquad h_2 = \sigma(z_2) \qquad h_3 = \sigma(z_3)$$

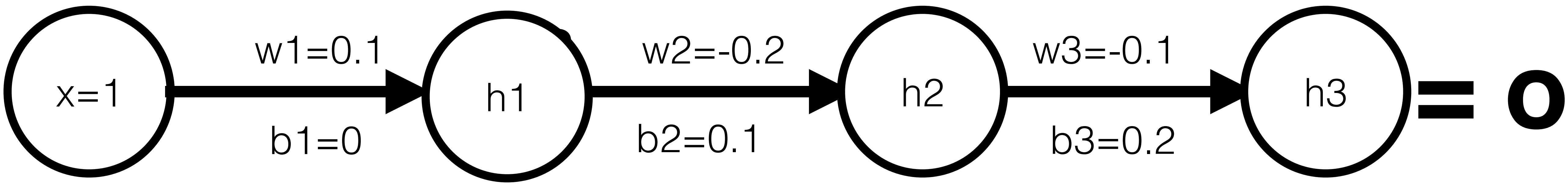


Compute h1,h2,h3 using Relu : please spend 5 minutes on this

Now we put in real numbers

$$z_1 = w_1 x + b_1 \qquad z_2 = w_2 h_1 + b_2 \qquad z_3 = w_3 h_2 + b_3$$

$$h_1 = \sigma(z_1) \qquad h_2 = \sigma(z_2) \qquad h_3 = \sigma(z_3)$$



z1 = 0.1*1+0 = 0.1
h1 = 0.1

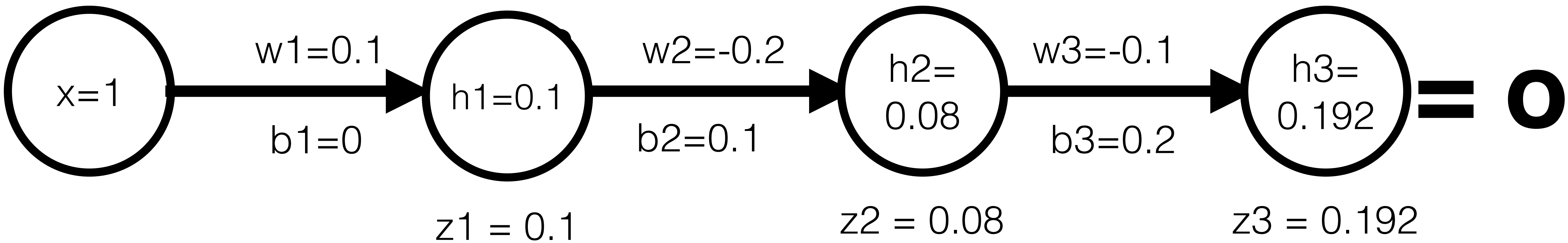
z2 = -0.2*0.1+0.1 = 0.08
h1 = 0.08

z3 = -0.1*0.08+0.2 = 0.192
h3 = 0.192

Backward pass, compute all the gradients

$$z_1 = w_1 x + b_1 \qquad z_2 = w_2 h_1 + b_2 \qquad z_3 = w_3 h_2 + b_3$$

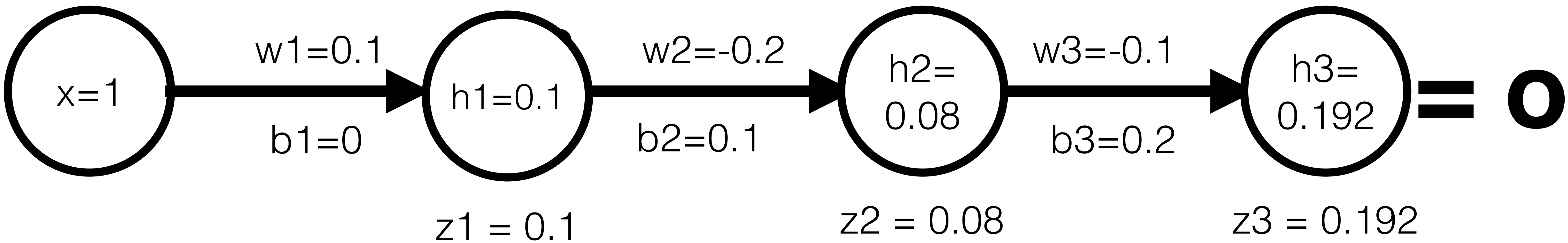
$$h_1 = \sigma(z_1) \qquad h_2 = \sigma(z_2) \qquad h_3 = \sigma(z_3)$$



Backward pass, compute all the gradients

$$z_1 = w_1 x + b_1 \qquad z_2 = w_2 h_1 + b_2 \qquad z_3 = w_3 h_2 + b_3$$

$$h_1 = \sigma(z_1) \qquad h_2 = \sigma(z_2) \qquad h_3 = \sigma(z_3)$$



$$\frac{\partial h_3}{\partial z_3} = 1$$

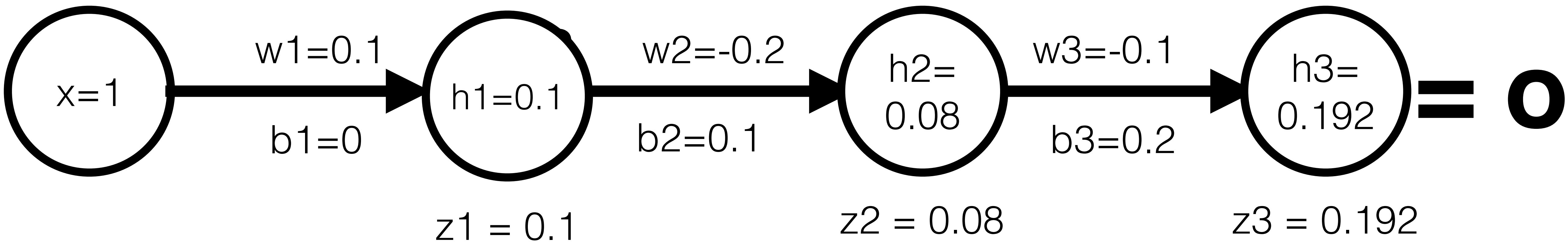
$$\frac{\partial h_3}{\partial z_2} = \frac{\partial h_3}{\partial z_3} \frac{\partial z_3}{\partial h_2} \frac{\partial h_2}{\partial z_2}$$

$$\frac{\partial h_3}{\partial z_1} = \frac{\partial h_3}{\partial z_3} \frac{\partial z_3}{\partial h_2} \frac{\partial h_2}{\partial z_2} \frac{\partial z_2}{\partial h_1} \frac{\partial h_1}{\partial z_1} = \frac{\partial h_3}{\partial z_2} \frac{\partial z_2}{\partial h_1} \frac{\partial h_1}{\partial z_1}$$

Backward pass, compute all the gradients

$$z_1 = w_1 x + b_1 \qquad z_2 = w_2 h_1 + b_2 \qquad z_3 = w_3 h_2 + b_3$$

$$h_1 = \sigma(z_1) \qquad h_2 = \sigma(z_2) \qquad h_3 = \sigma(z_3)$$



$$\frac{\partial h_3}{\partial z_3} = 1$$

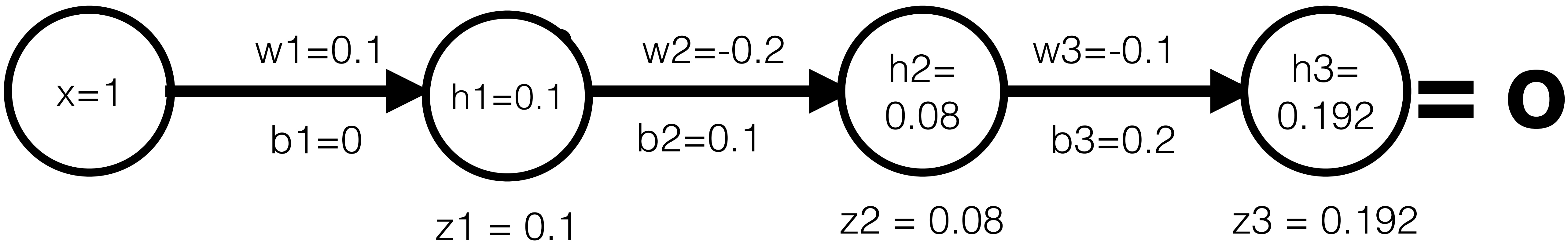
$$\frac{\partial h_3}{\partial z_2} = ? \qquad \frac{\partial h_3}{\partial z_2} = \frac{\partial h_3}{\partial z_3}\frac{\partial z_3}{\partial h_2}\frac{\partial h_2}{\partial z_2} = (1)(-0.1)(1) = -0.1$$

$$\frac{\partial h_3}{\partial z_1} = ? = \frac{\partial h_3}{\partial z_2}\frac{\partial z_2}{\partial h_1}\frac{\partial h_1}{\partial z_1} = (-0.1)(-0.2)(1) = 0.02$$

Backward pass, compute all the gradients

$$z_1 = w_1 x + b_1 \qquad z_2 = w_2 h_1 + b_2 \qquad z_3 = w_3 h_2 + b_3$$

$$h_1 = \sigma(z_1) \qquad h_2 = \sigma(z_2) \qquad h_3 = \sigma(z_3)$$



$$\frac{\partial h_3}{\partial w_3} = \frac{\partial h_3}{\partial z_3}\frac{\partial z_3}{\partial w_3} = (1)(0.08) = 0.08 \qquad \frac{\partial h_3}{\partial z_3} = 1$$

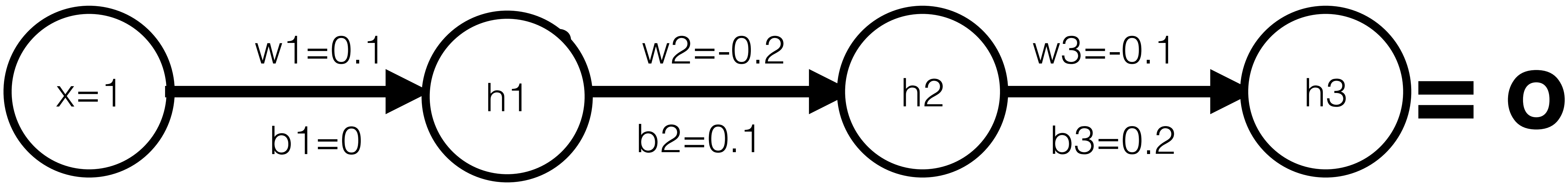
$$\frac{\partial h_3}{\partial w_2} = \frac{\partial h_3}{\partial z_2}\frac{\partial z_2}{\partial w_2} = (-0.1)(0.1) = -0.01 \qquad \frac{\partial h_3}{\partial z_2} = -0.1$$

$$\frac{\partial h_3}{\partial w_1} = \frac{\partial h_3}{\partial z_1}\frac{\partial z_1}{\partial w_1} = (0.02)(1) = 0.02 \qquad \frac{\partial h_3}{\partial z_1} = 0.02$$

Backward pass, compute all the gradients

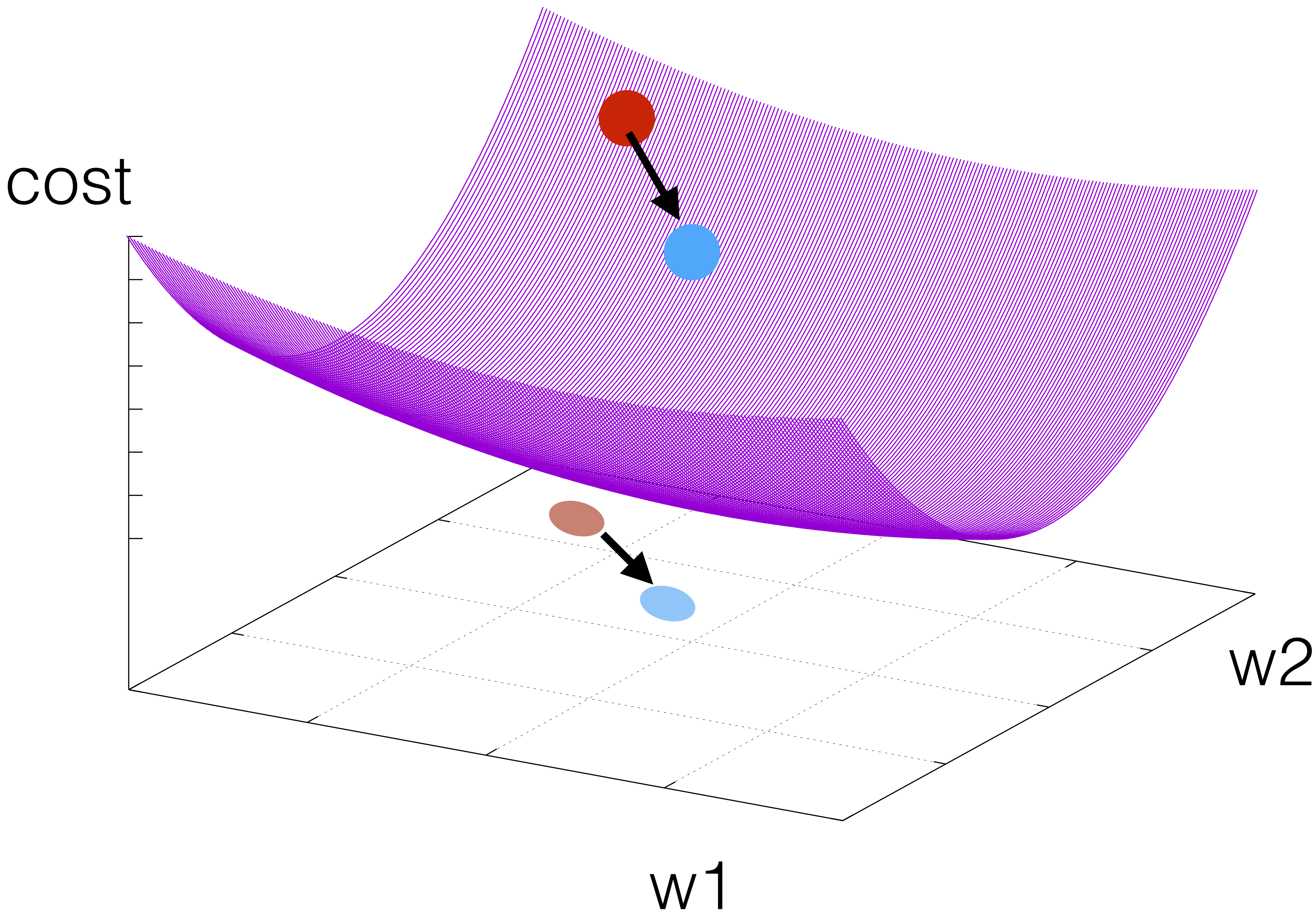
$$z_1 = w_1 x + b_1 \qquad z_2 = w_2 h_1 + b_2 \qquad z_3 = w_3 h_2 + b_3$$

$$h_1 = \sigma(z_1) \qquad h_2 = \sigma(z_2) \qquad h_3 = \sigma(z_3)$$



Please spend 2 minutes
to compute gradients for

$$\frac{\partial h_3}{\partial b_3}, \frac{\partial h_3}{\partial b_2}, \frac{\partial h_3}{\partial b_1},$$



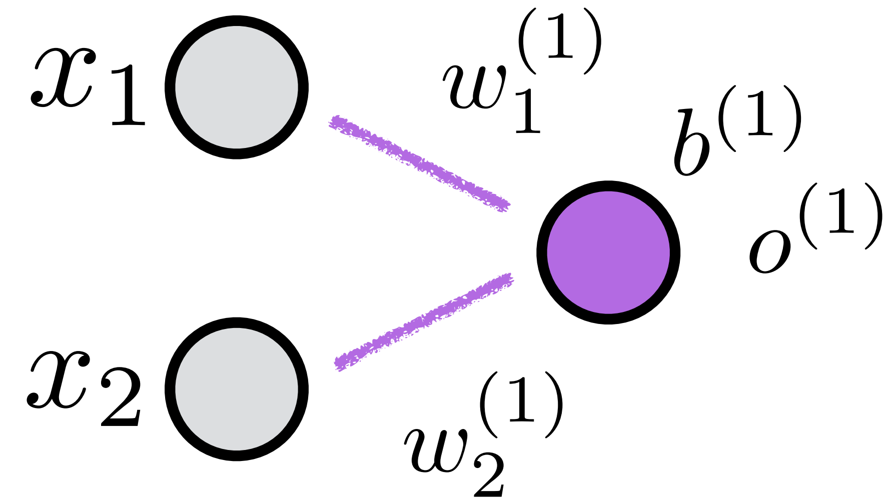
$$w_j(t+1) = w_j(t) - \eta \frac{\partial C}{\partial w_j}$$

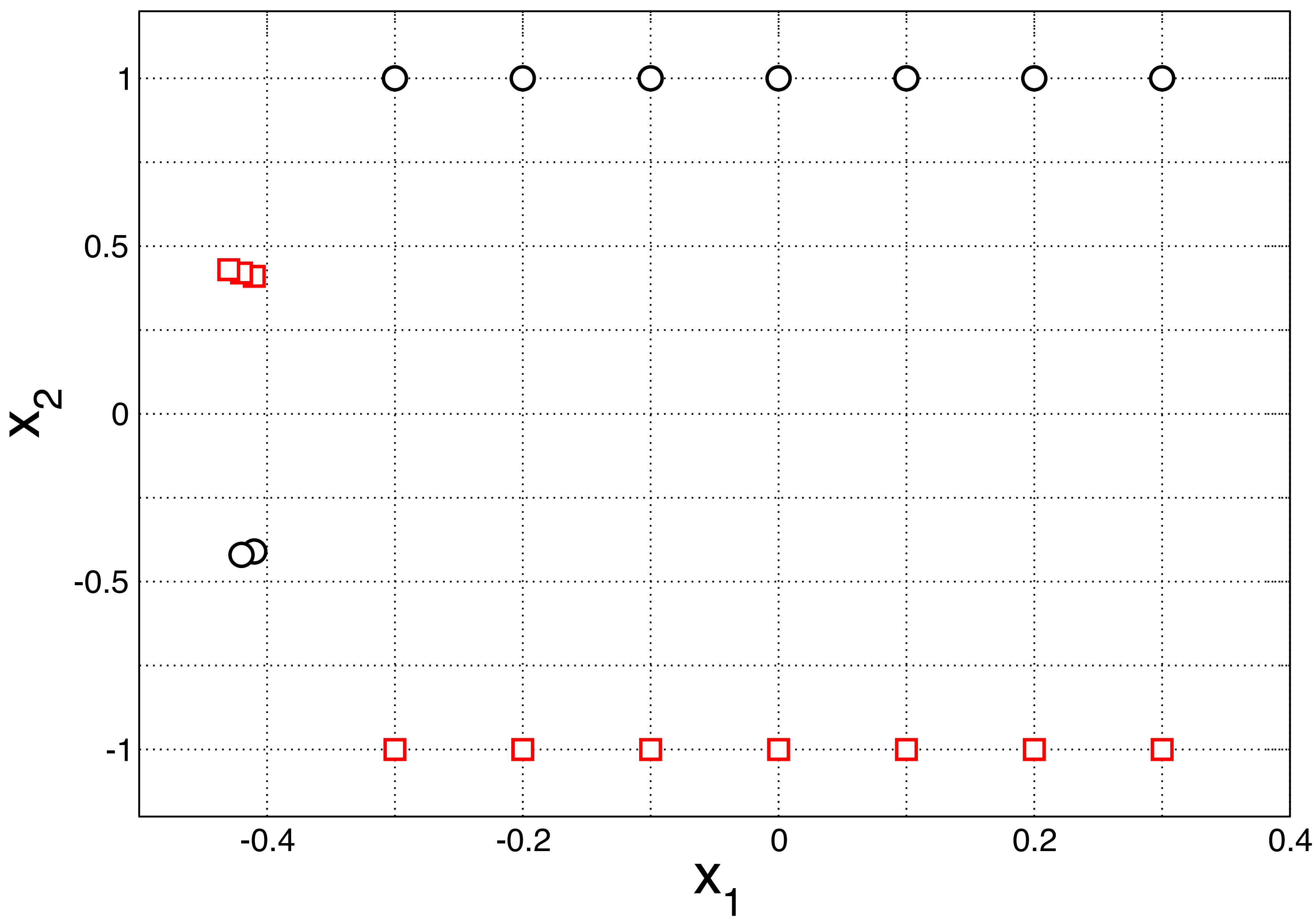
Local minimum problem

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

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

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

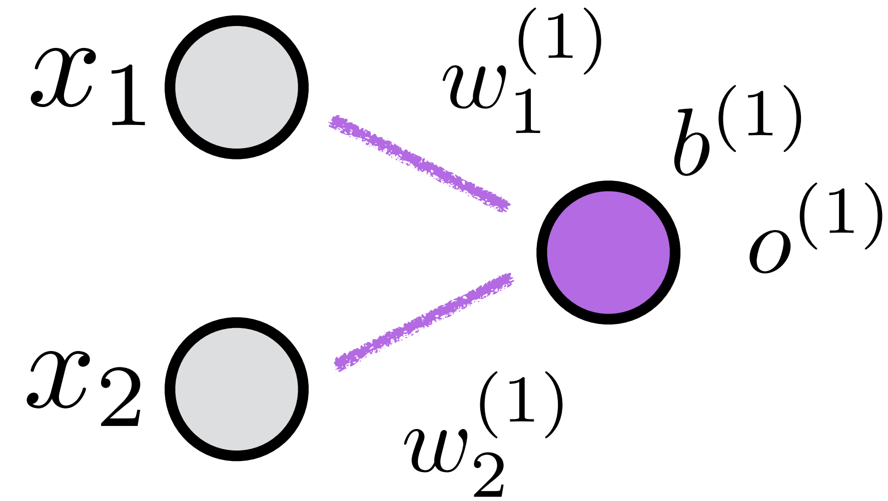


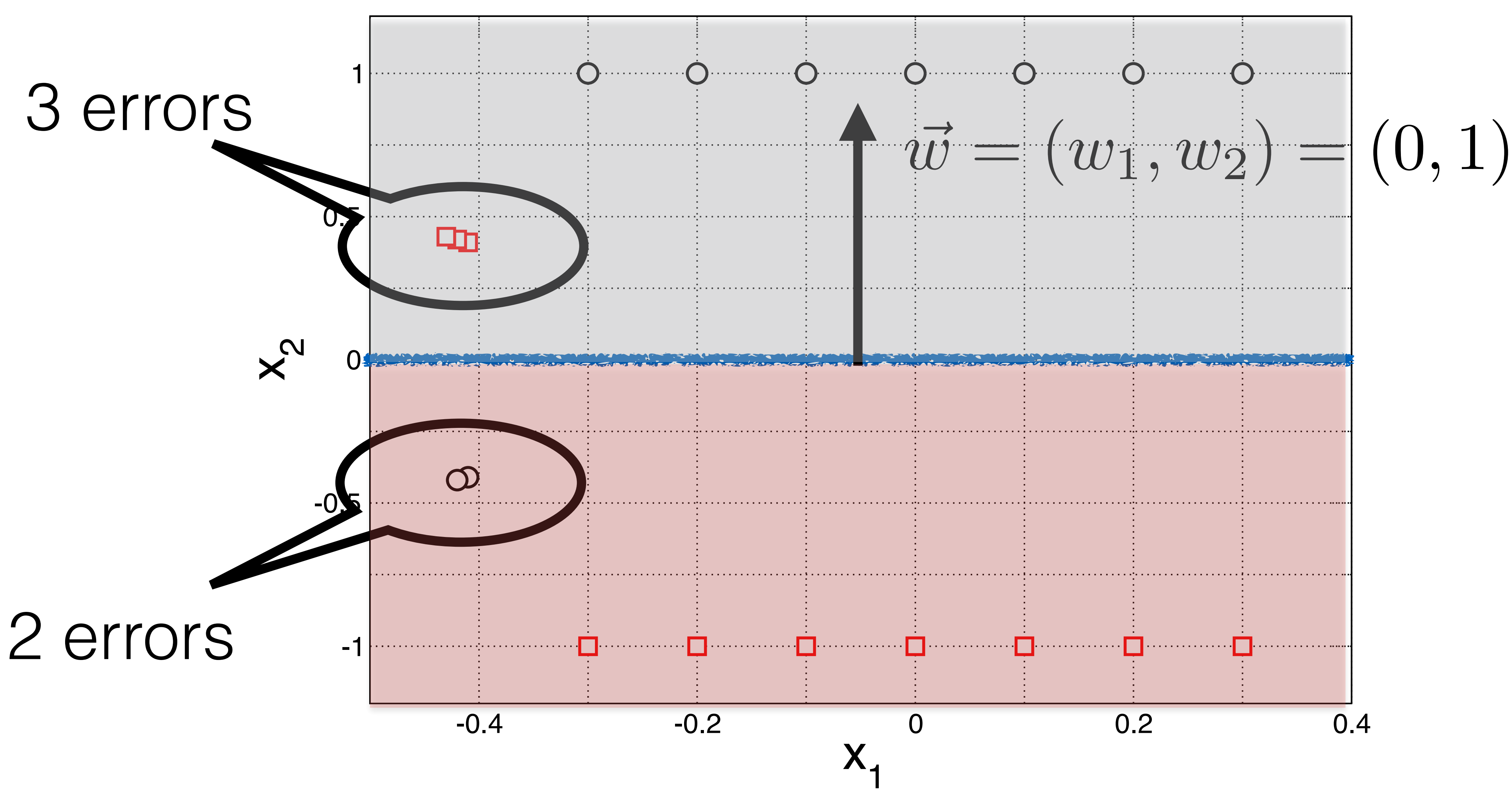


$$o_i = \sigma(w_1 x_1 + w_2 x_2) \text{ with } w_2 = 1$$

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

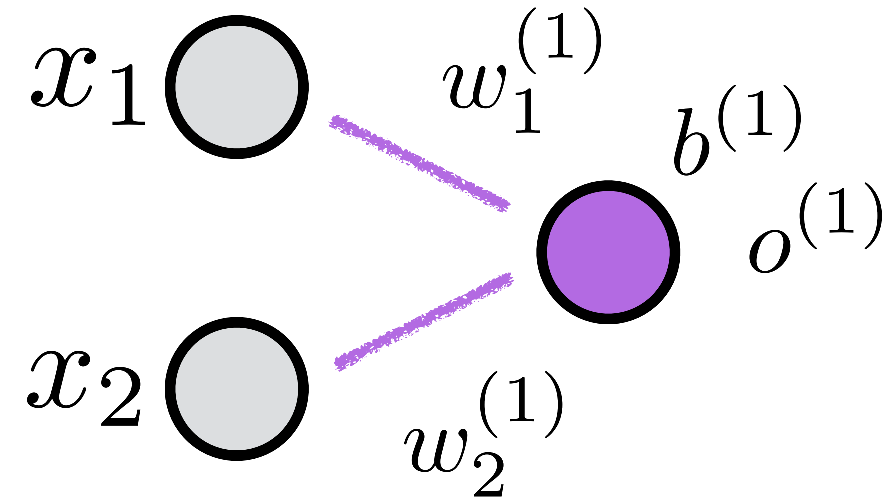
$$C(w_1) = \frac{1}{n}\sum_i (y_i - o_i)^2$$



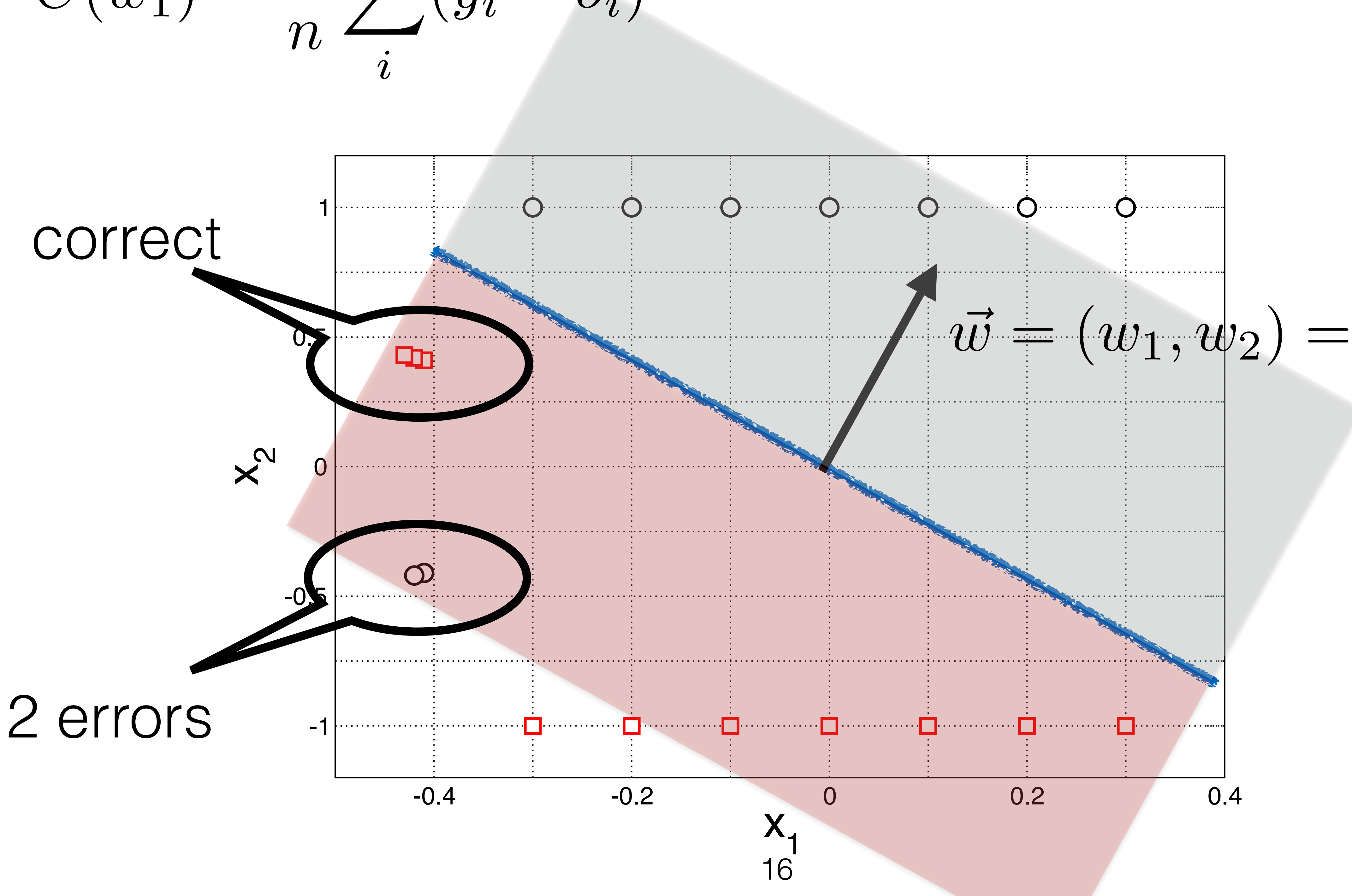


$$o_i = \sigma(w_1 x_1 + w_2 x_2) \text{ with } w_2 = 1$$

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

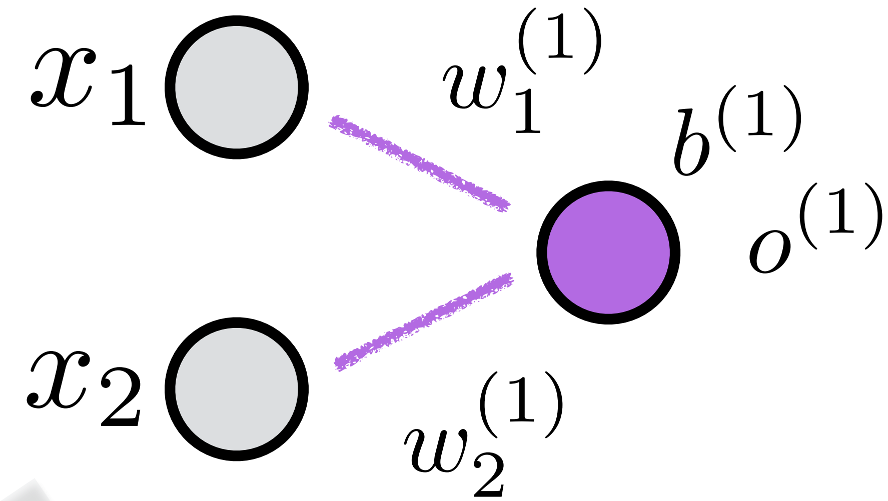


$$C(w_1) = \frac{1}{n} \sum_i (y_i - o_i)^2$$

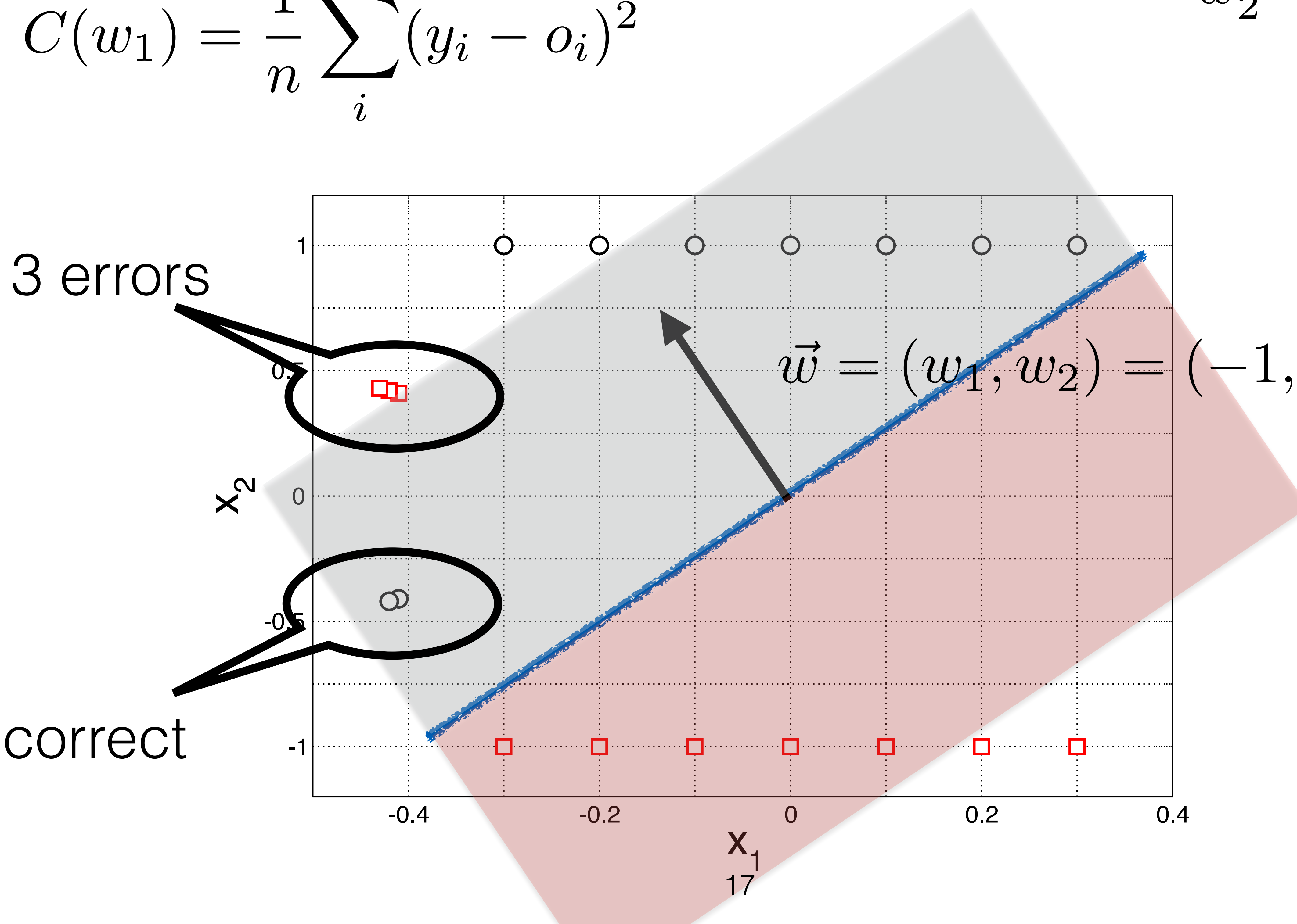


$$o_i = \sigma(w_1 x_1 + w_2 x_2) \text{ with } w_2 = 1$$

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



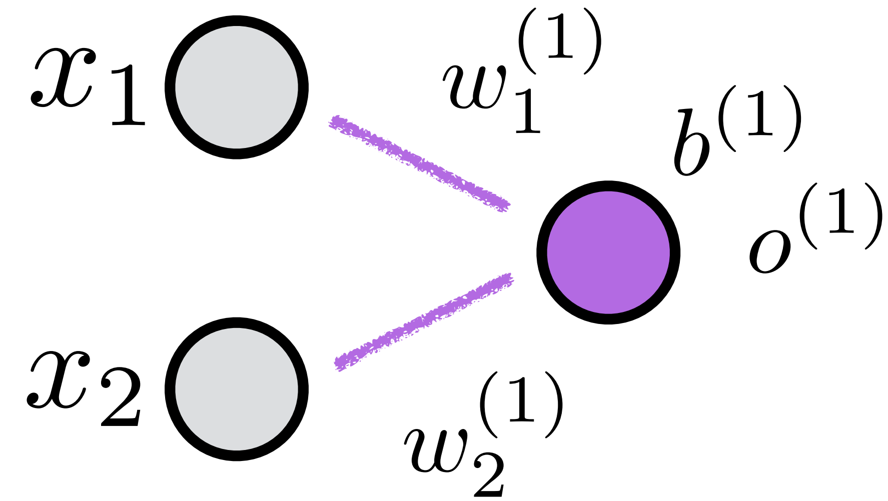
$$C(w_1) = \frac{1}{n} \sum_i (y_i - o_i)^2$$

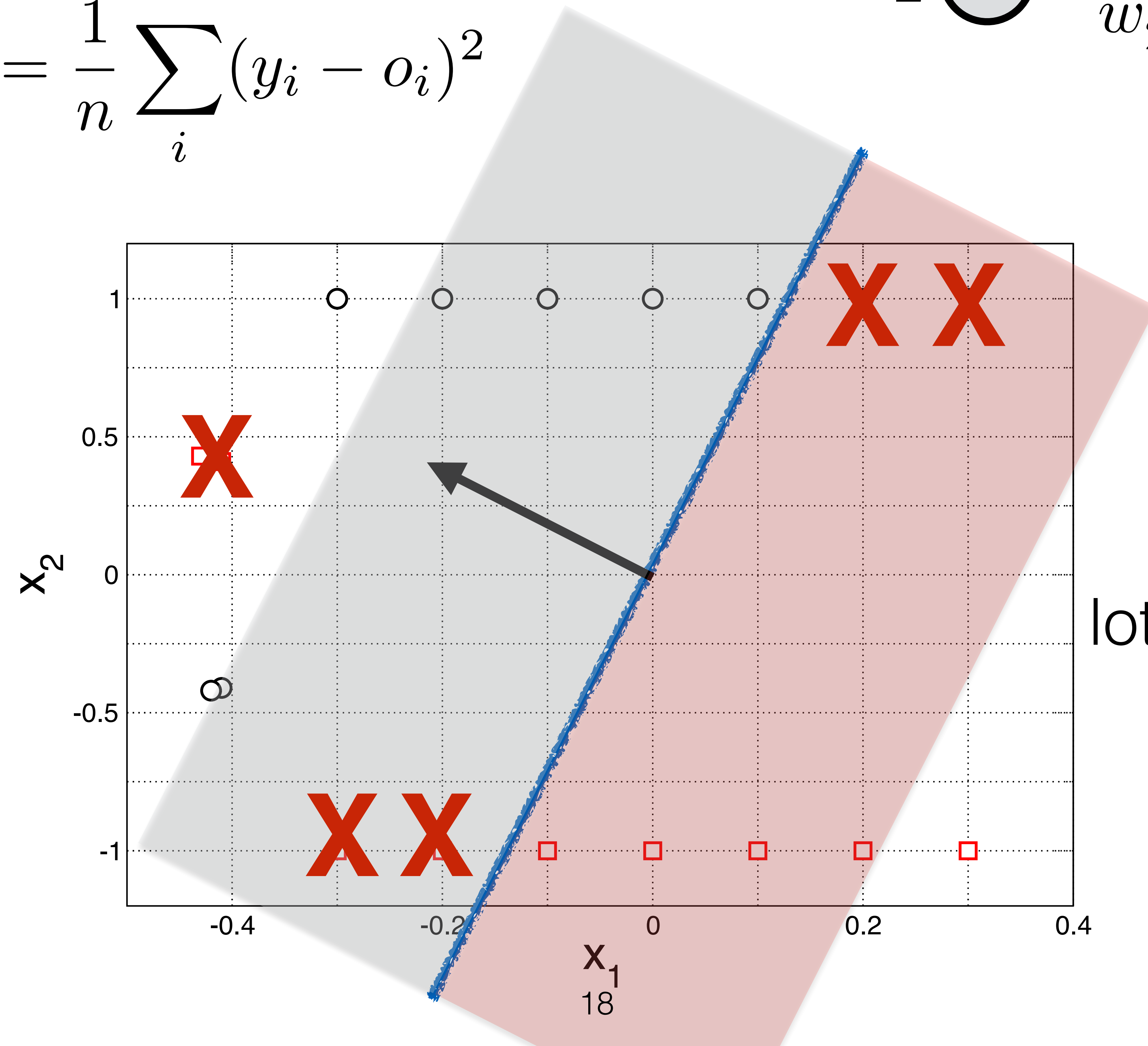


$$o_i = \sigma(w_1 x_1 + w_2 x_2) \text{ with } w_2 = 1$$

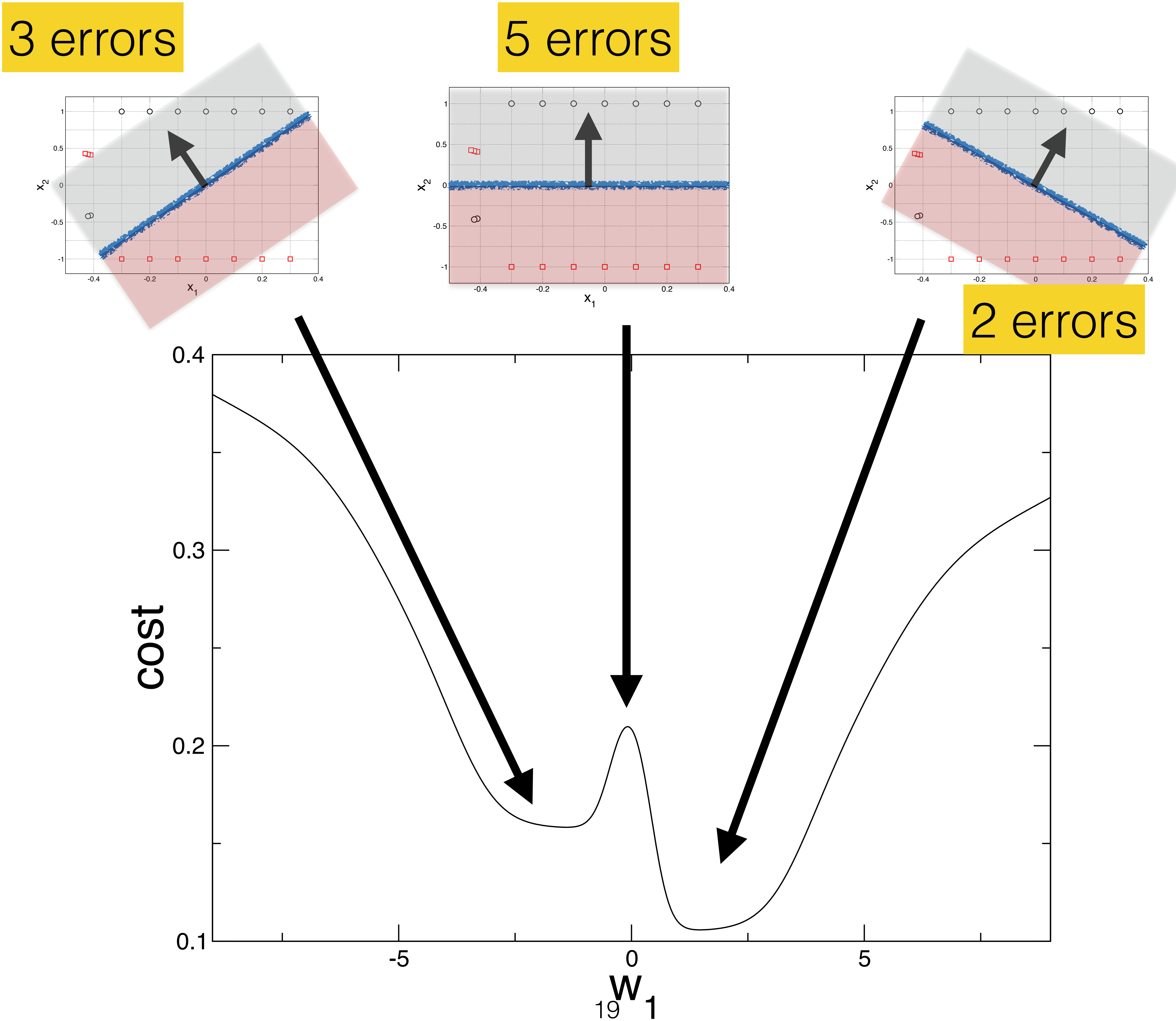
$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

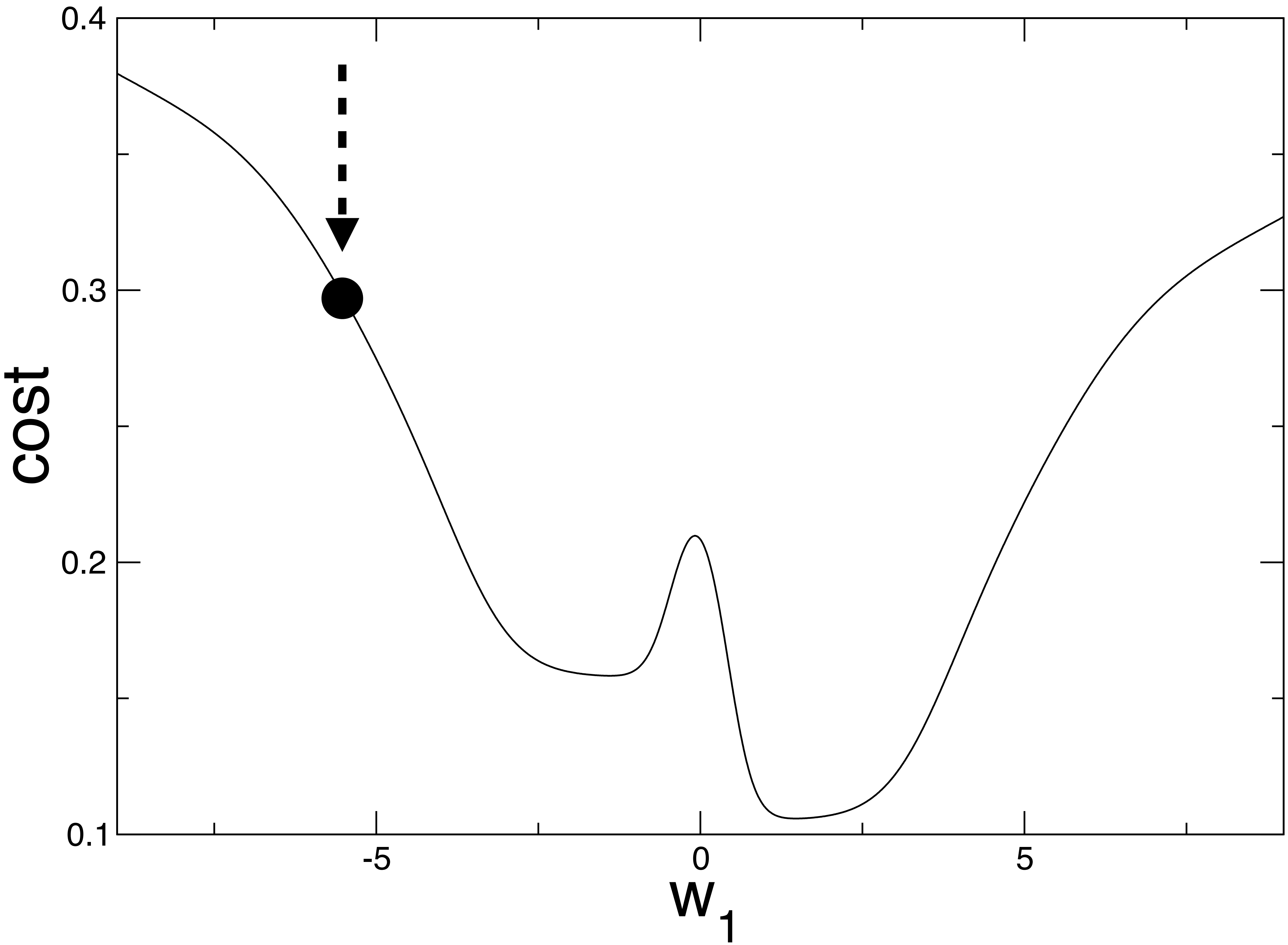
$$C(w_1) = \frac{1}{n} \sum_i (y_i - o_i)^2$$

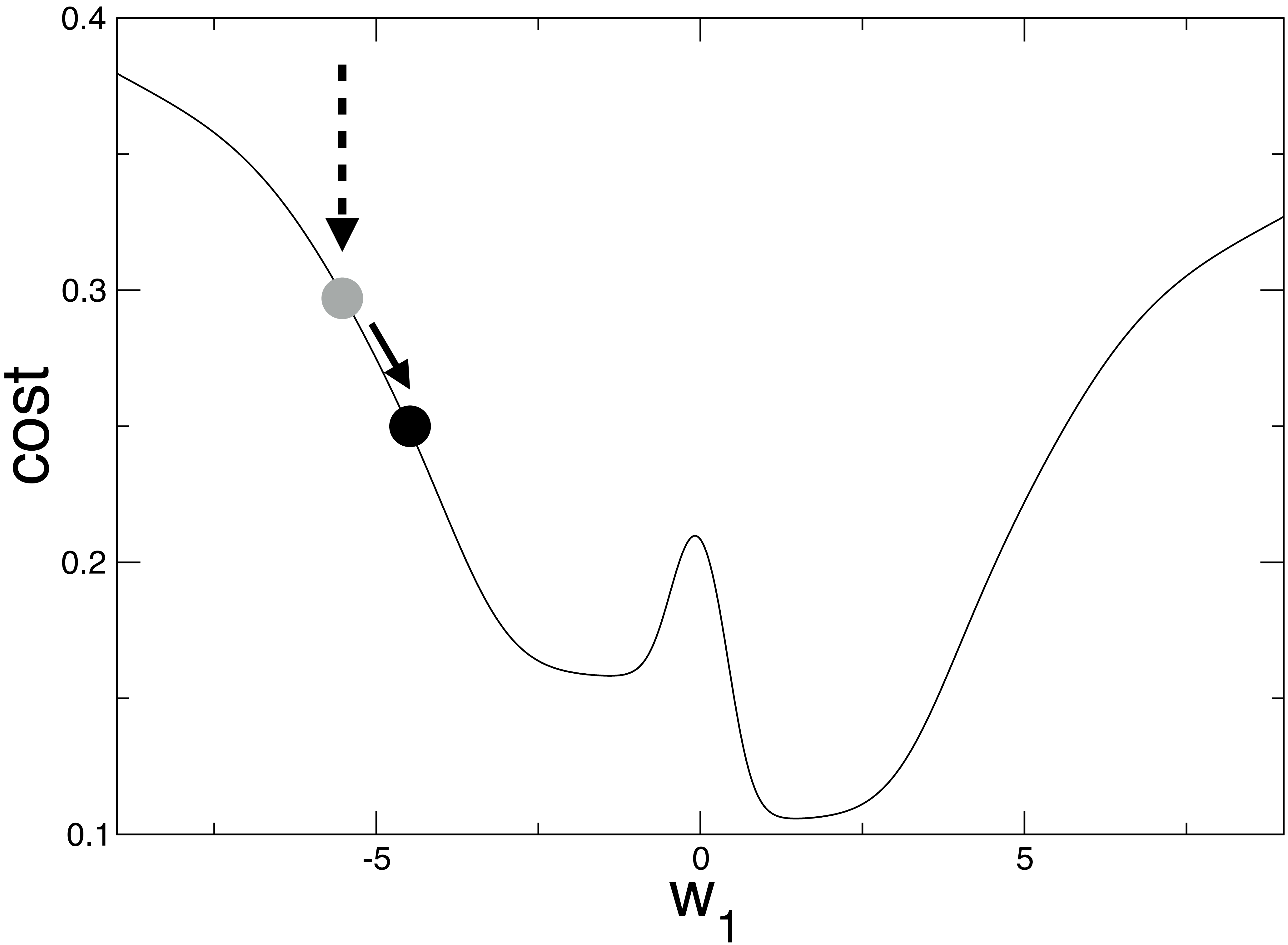




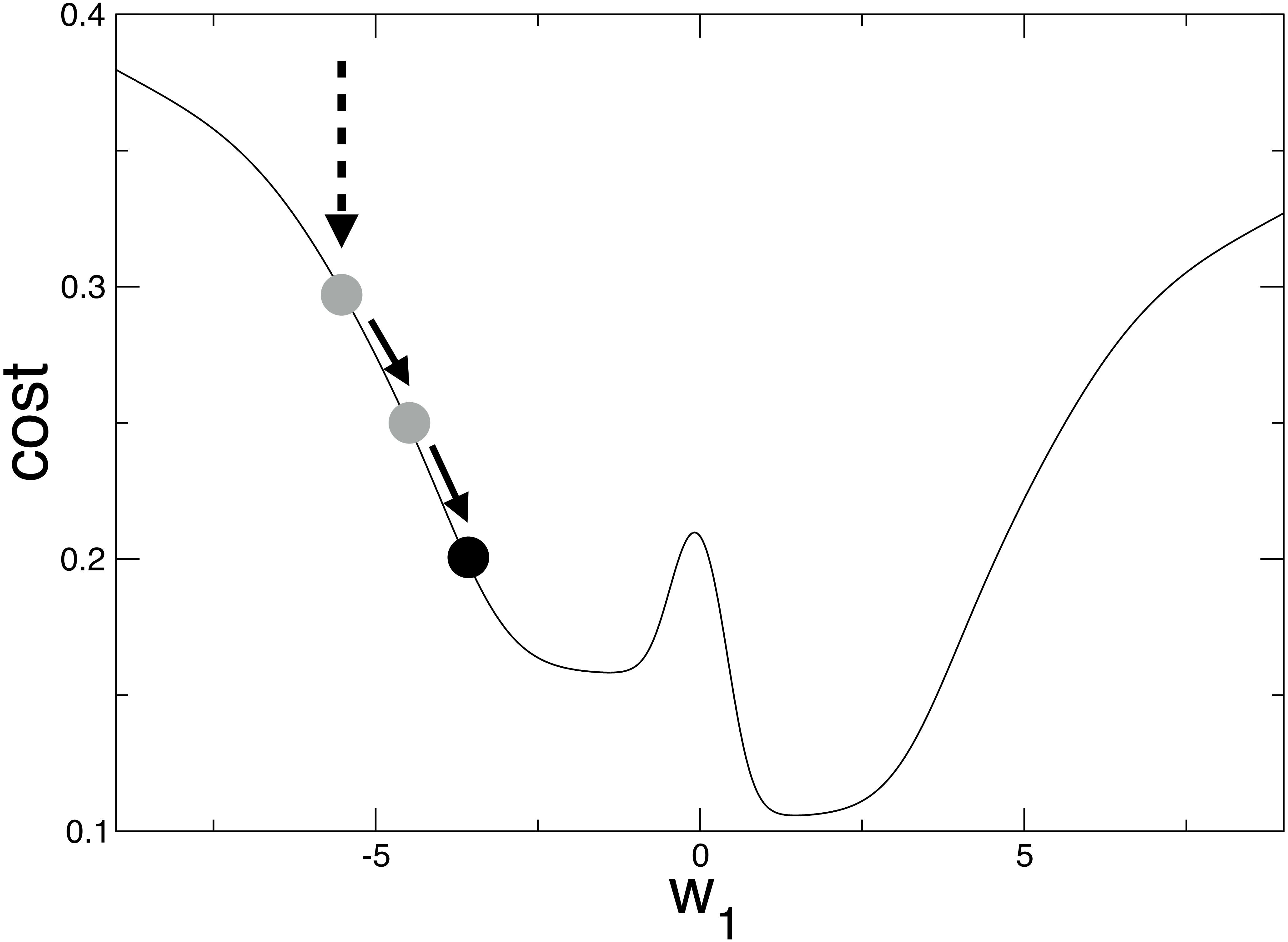
lots of errors!


3 errors
5 errors
2 errors
x1
x2
cost
0.4
0.3
0.2
0.1
-5
0
5
w1

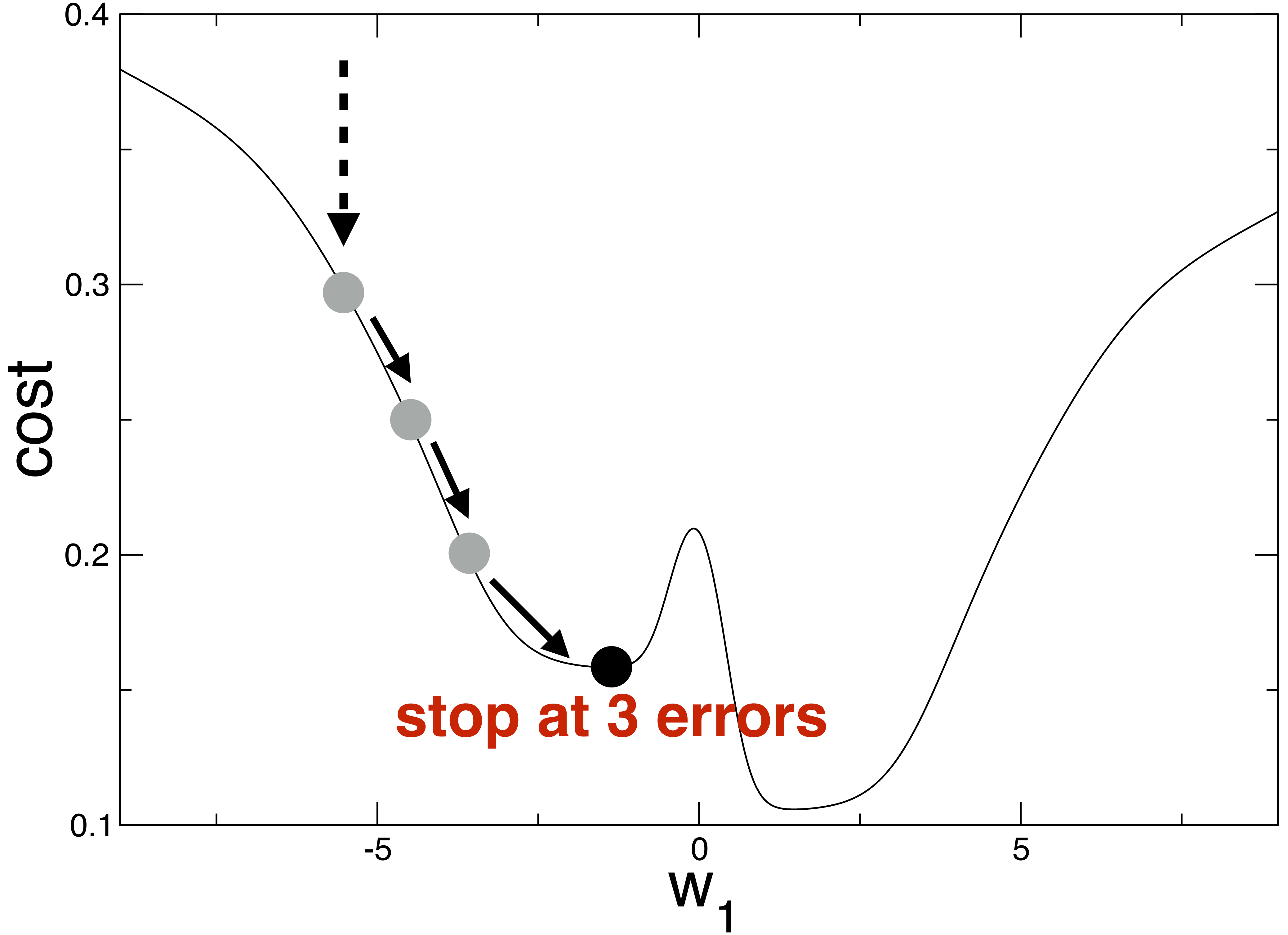

0.4
0.3
0.2
0.1
cost
-5
0
5
w_1



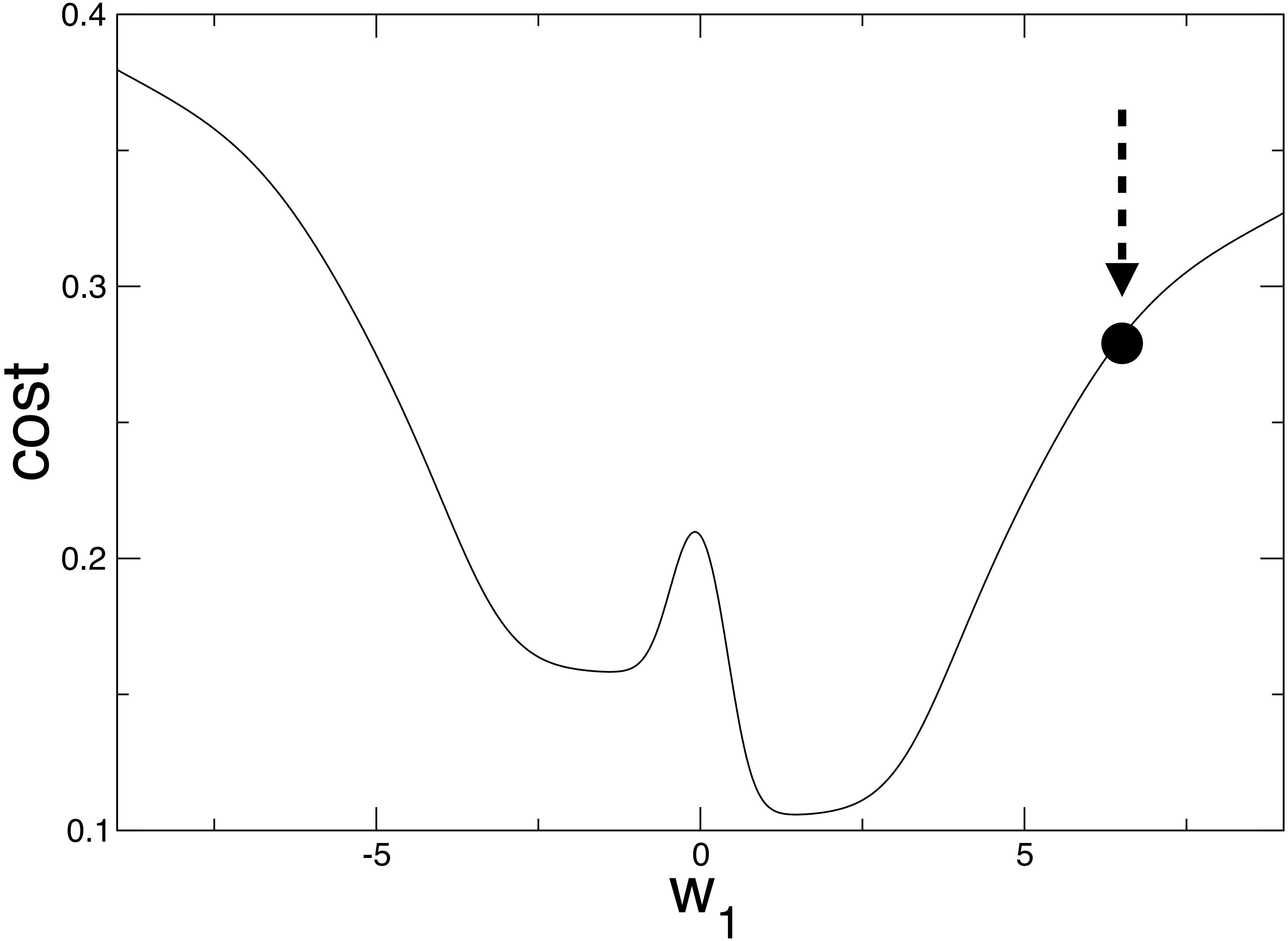
0.4
0.3
0.2
0.1
cost
-5
0
5
w_1

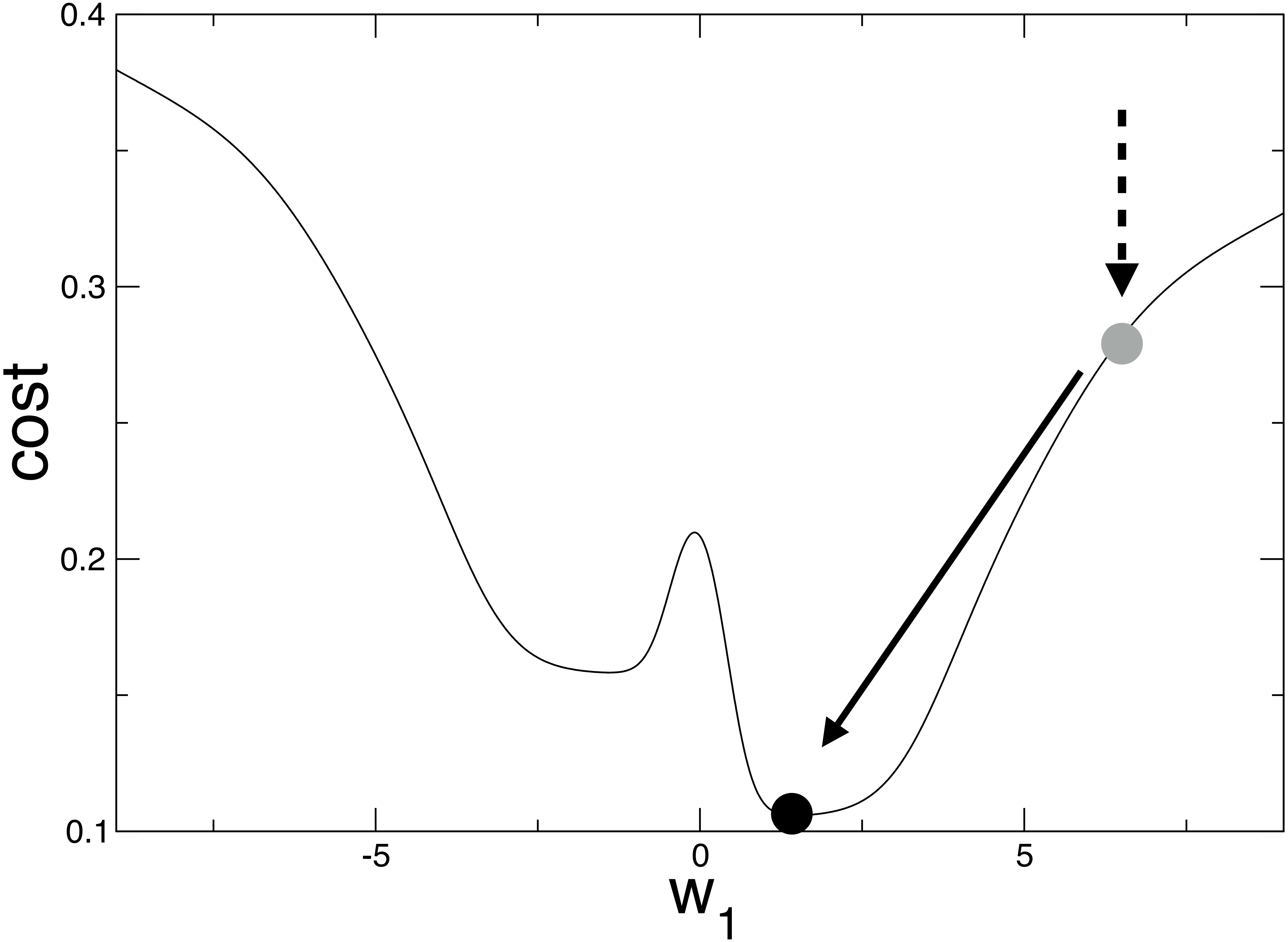


0.4
0.3
0.2
0.1
cost
-5
0
5
w1



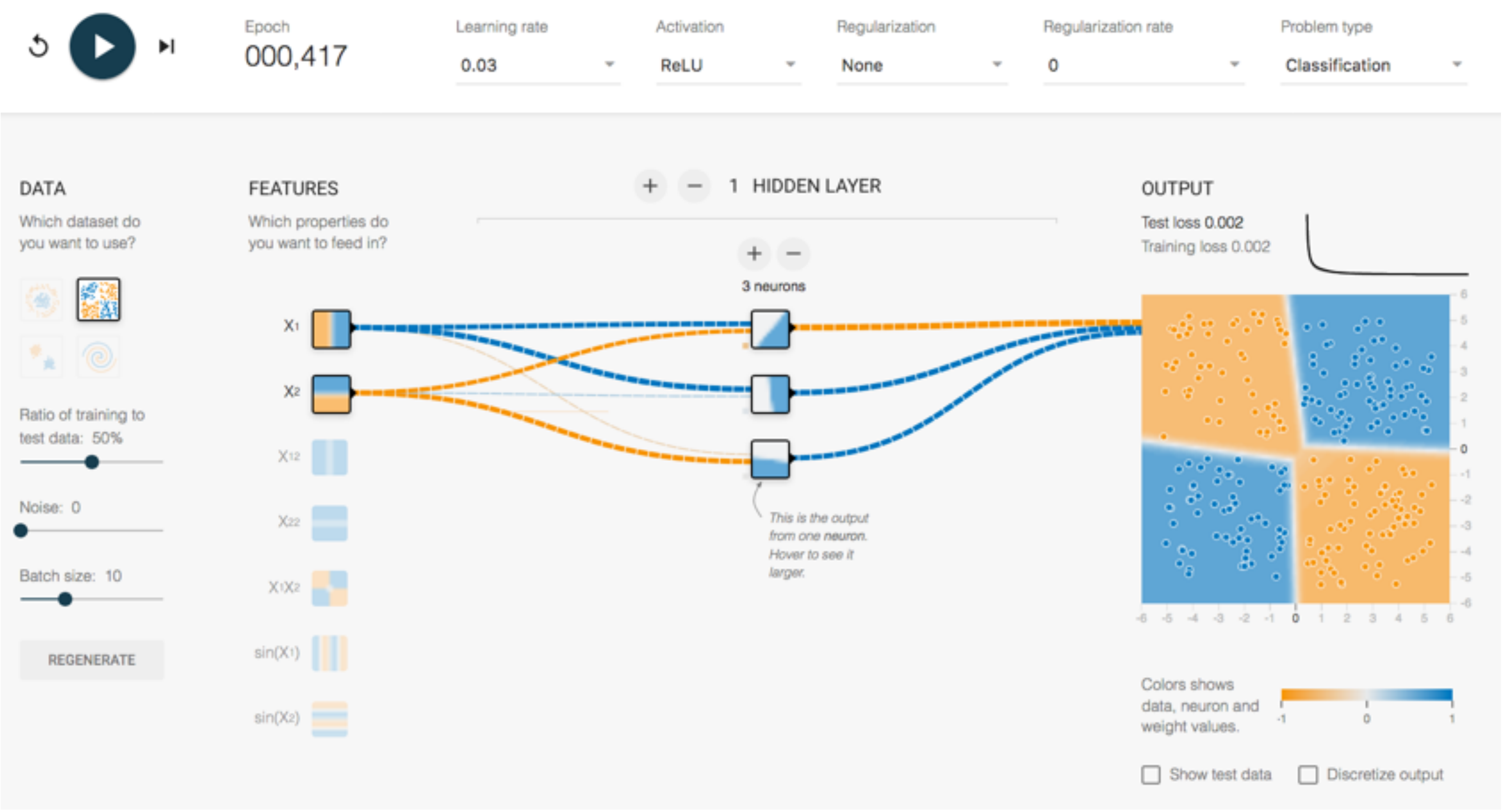
0.4
0.3
0.2
0.1
cost
-5
0
5
w_1
stop at 3 errors


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

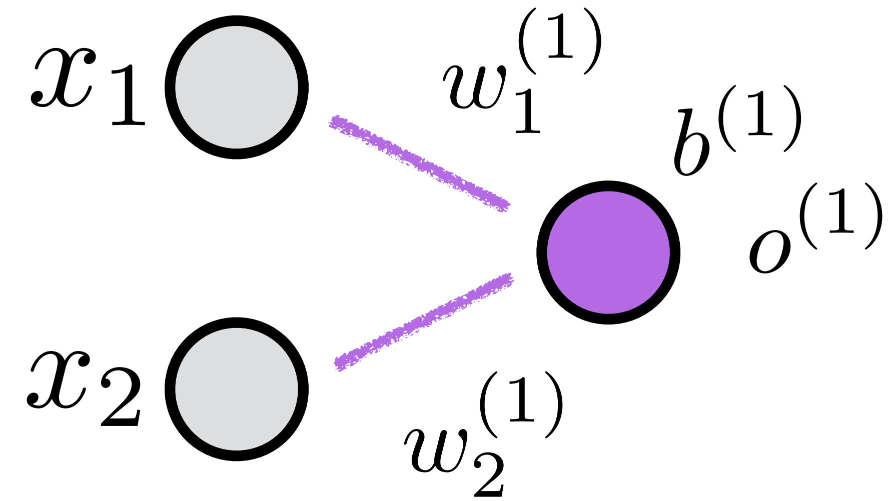
Strategies for overcoming local minimum problem

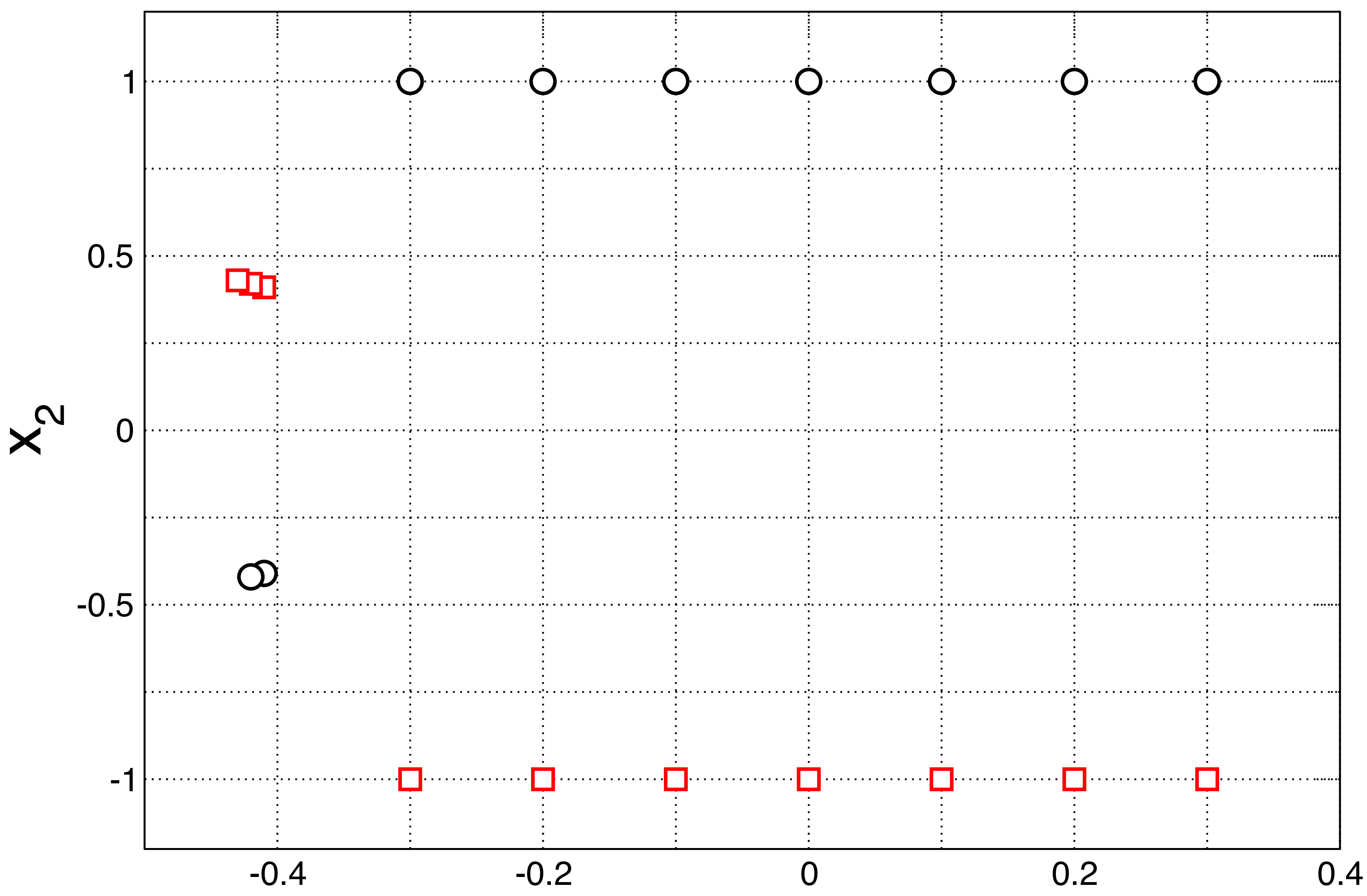
1. Stochastic gradient descend
2. Adam method, momentum

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

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

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



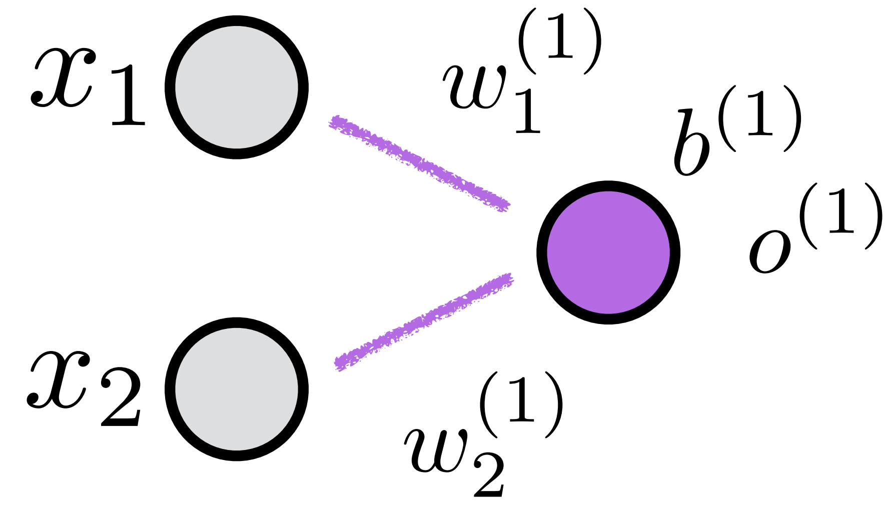


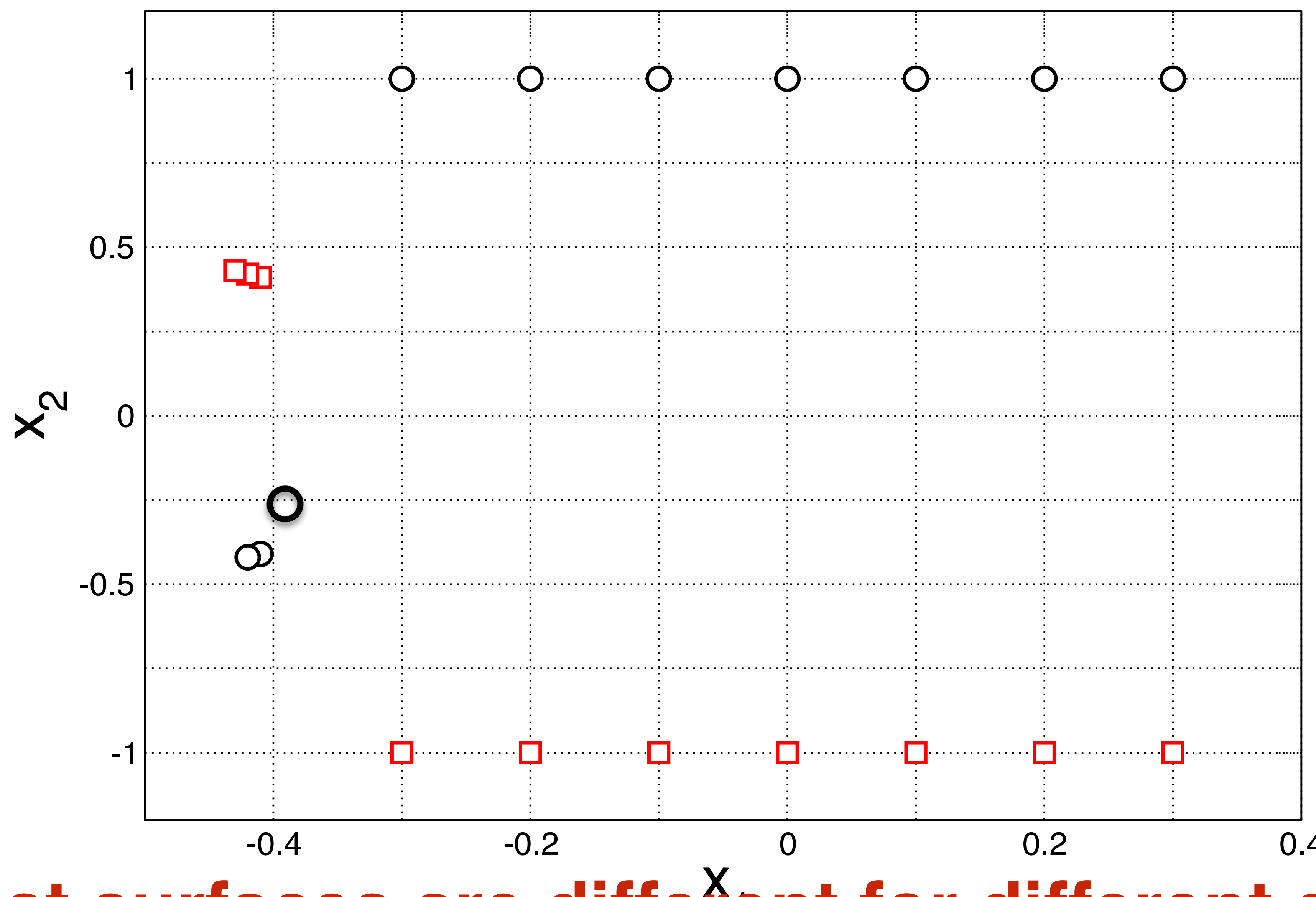
cost surfaces are different for different data sets

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

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

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



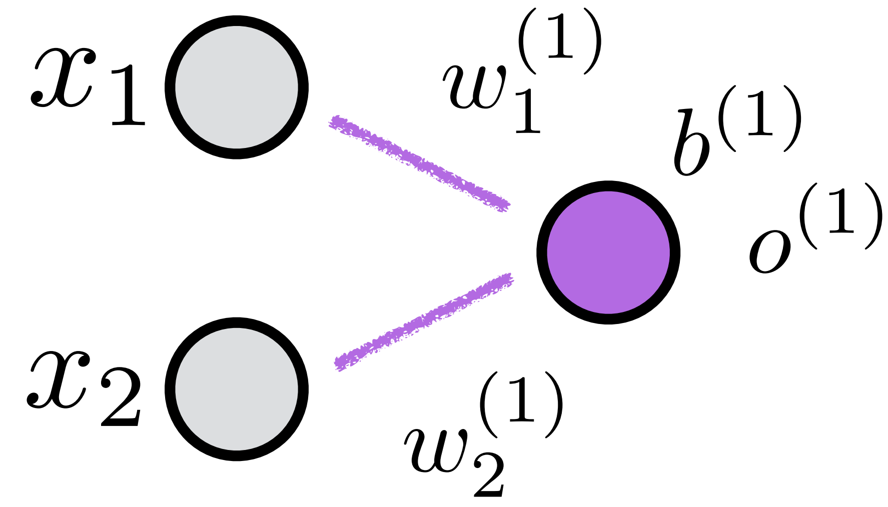


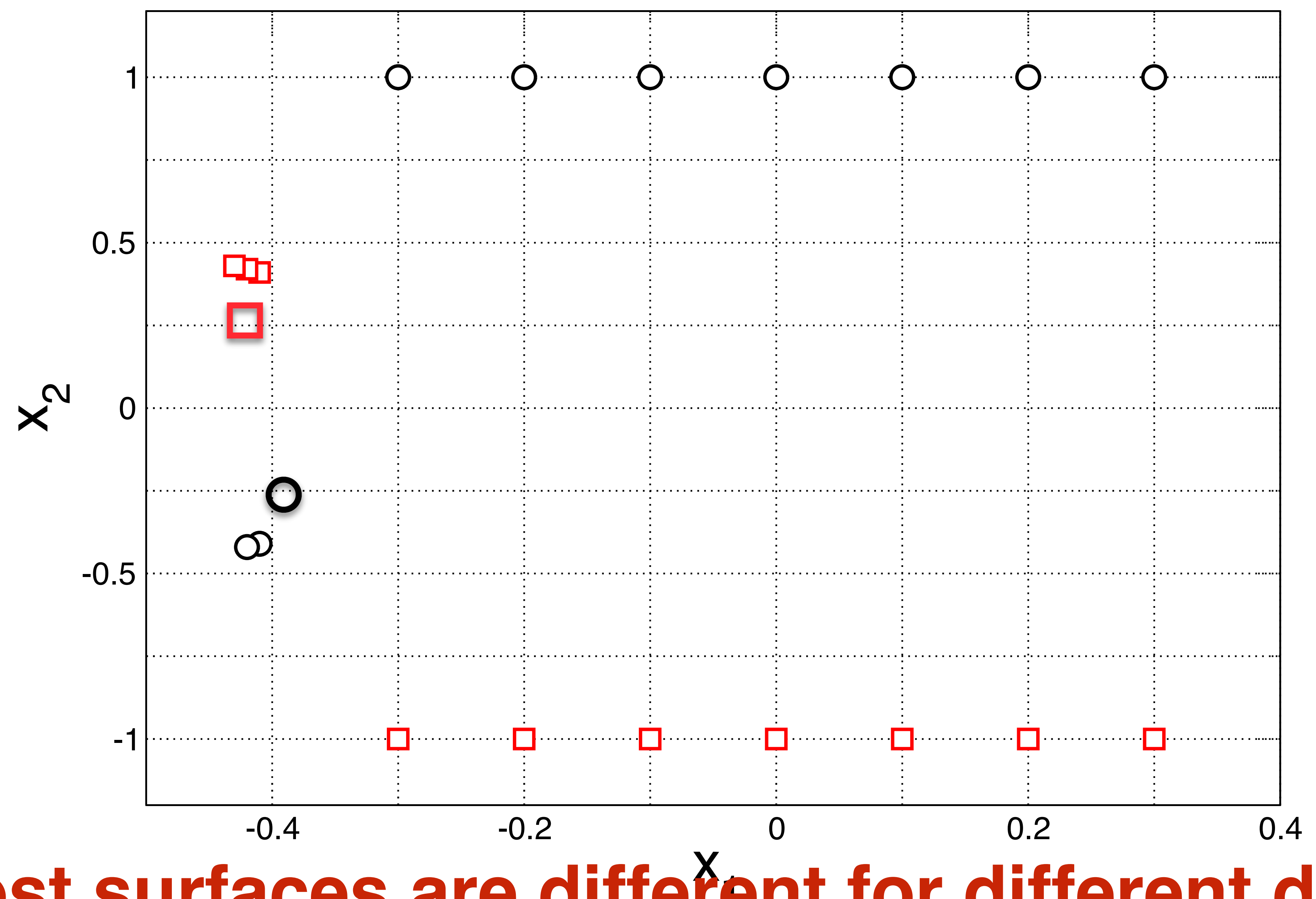
cost surfaces are different for different data sets

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

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

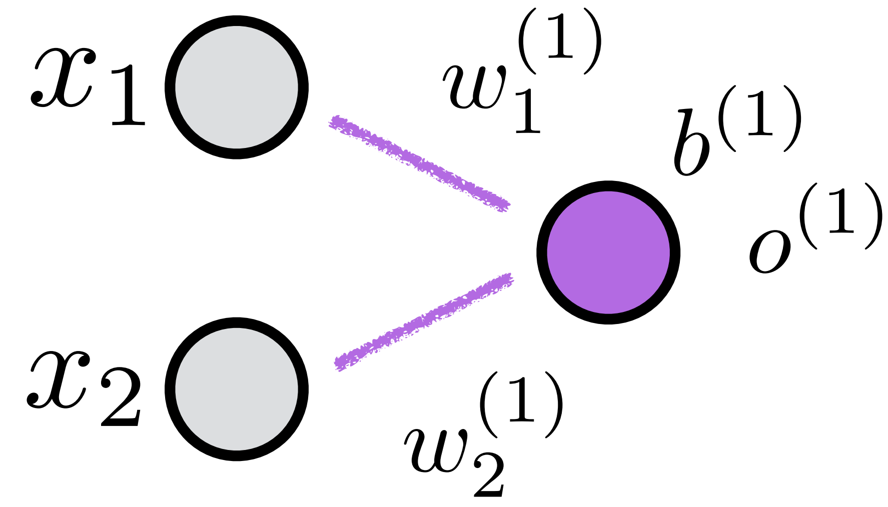
$$C(w_1) = \frac{1}{n} \sum_i (y_i - o_i)^2$$





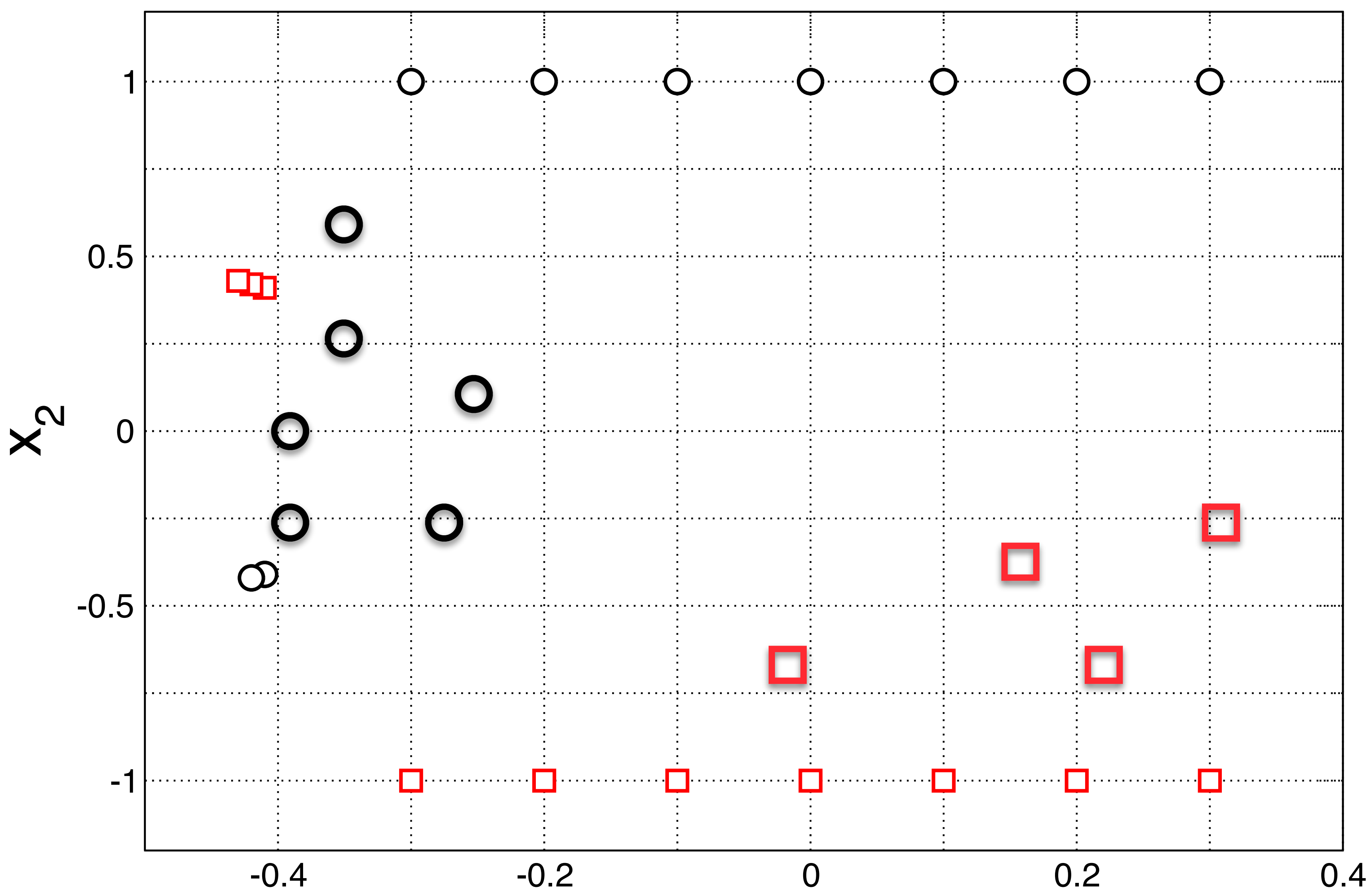
cost surfaces are different for different data sets

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



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

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

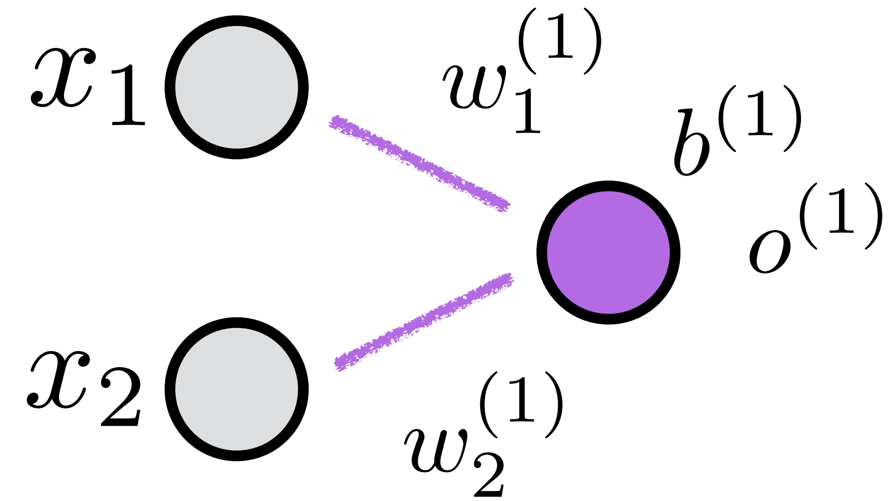


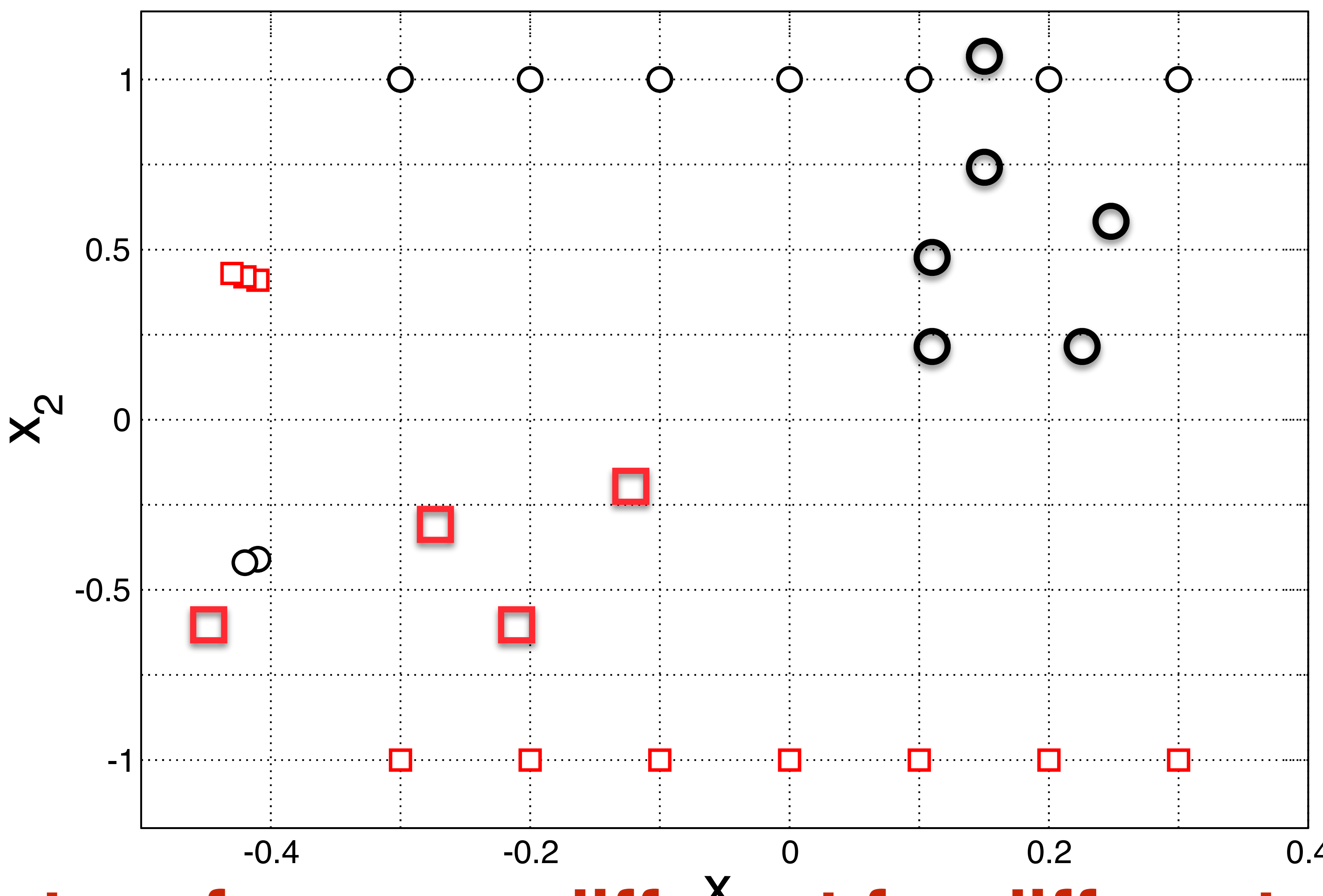
cost surfaces are different for different data sets

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

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

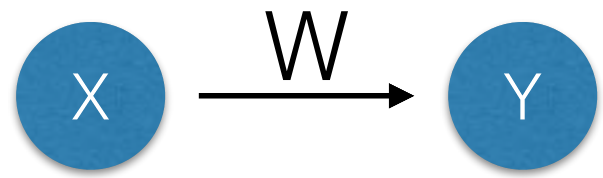
$$C(w_1) = \frac{1}{n} \sum_i (y_i - o_i)^2$$





cost surfaces are different for different data sets

x	y
0.97	2.0
0.016	0.025
0.87	1.4
0.70	1.5
0.11	0.19
0.023	0.048
0.65	1.4
0.27	0.55
0.21	0.40
0.087	0.19



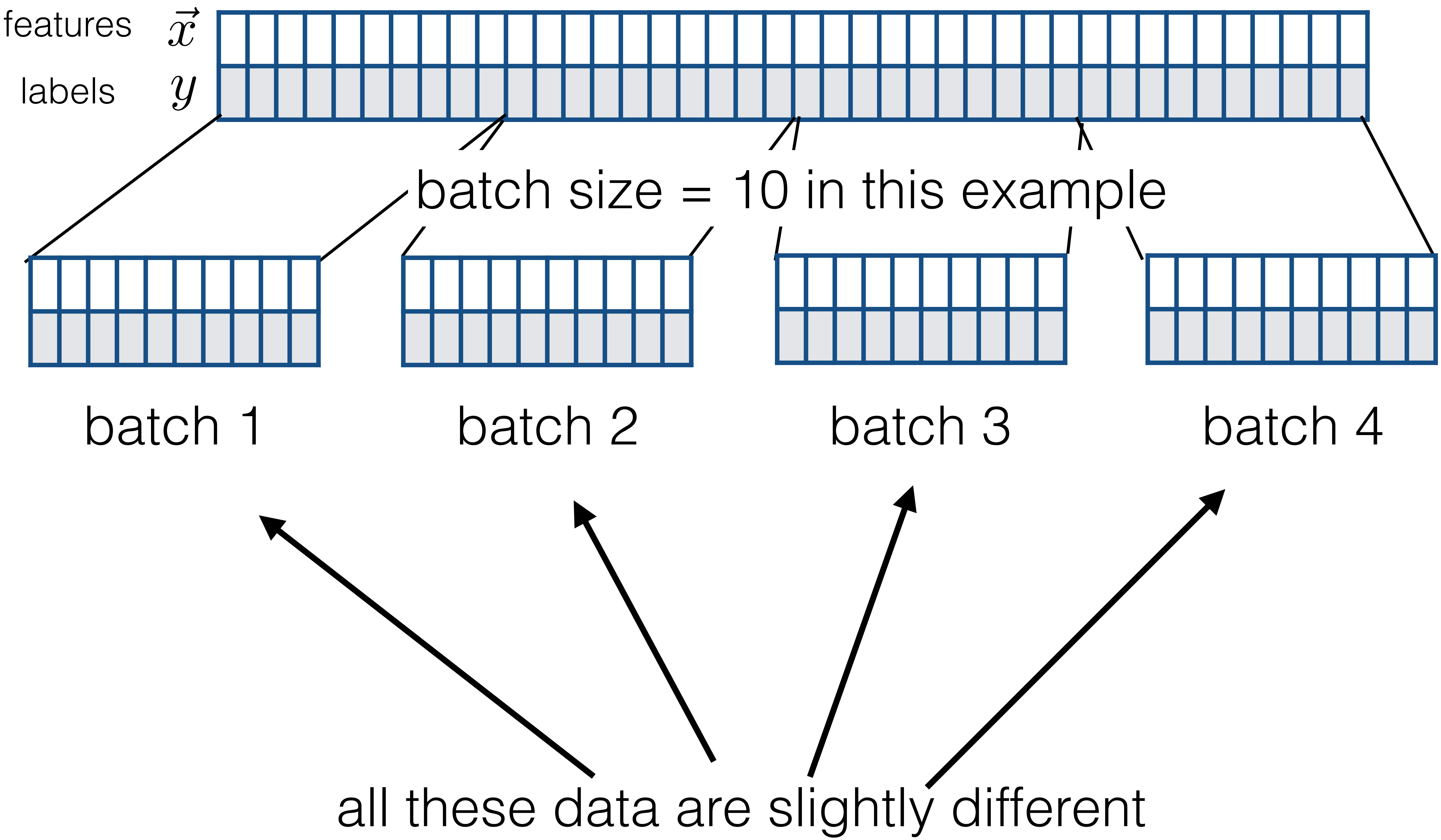
Use the loss function

$L(w|x,y) = \text{sum } (y_i - wx_i)^2$

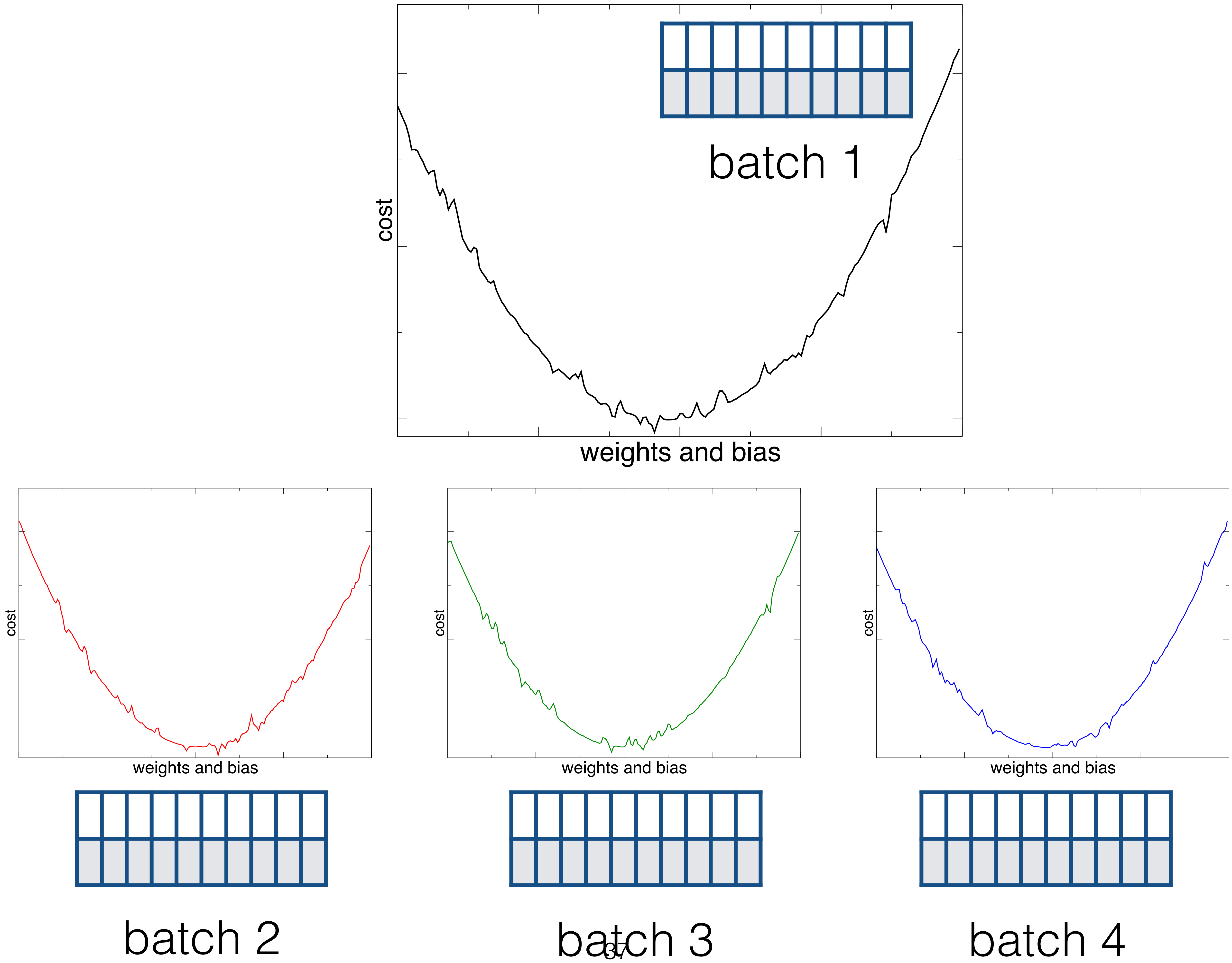
- Randomly choose 3 data points {x,y}
- Plot $L(w|x,y)$ versus w
- Repeat the above several times

Overlay your plots

Minibatch gradient descend

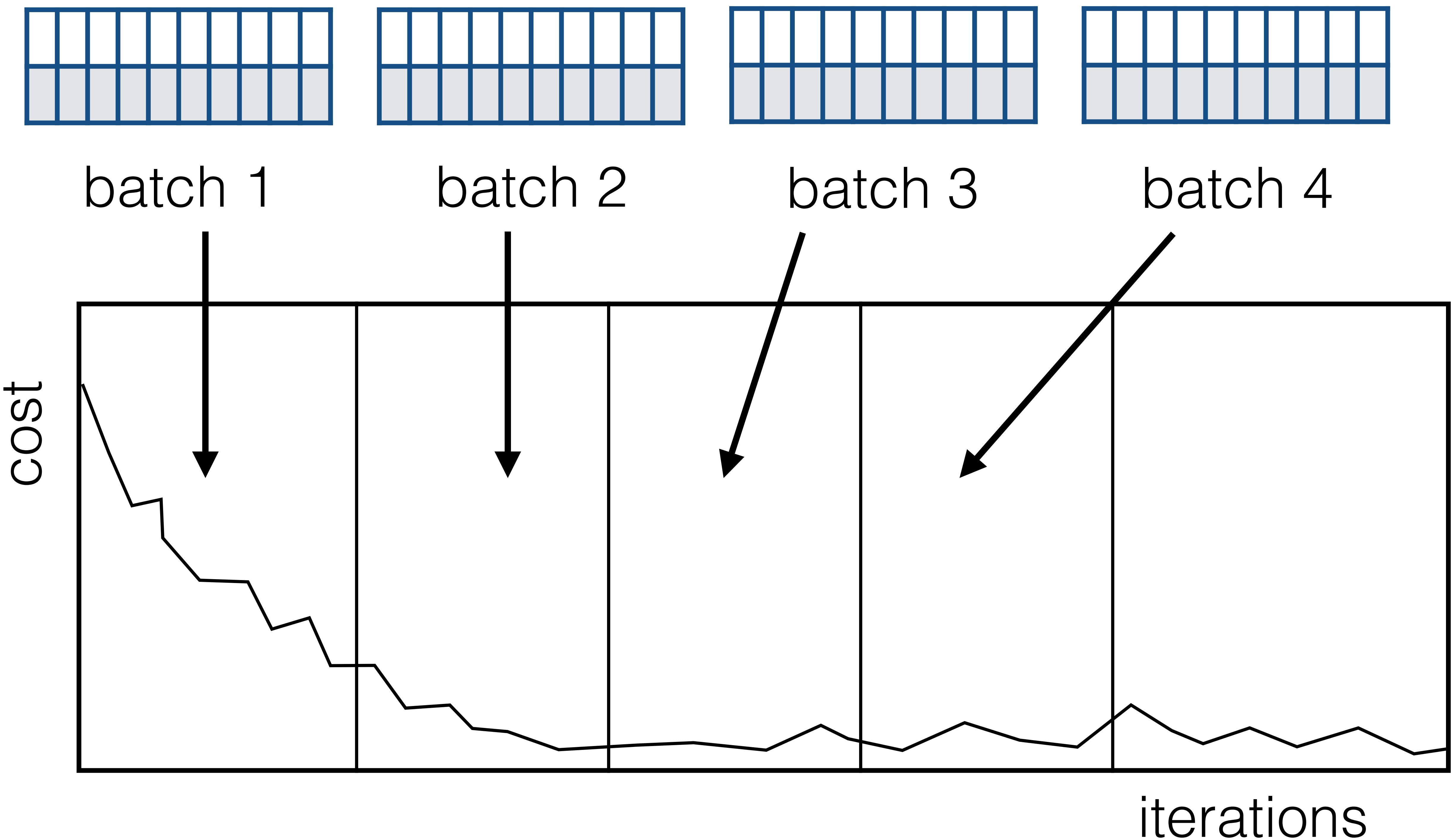


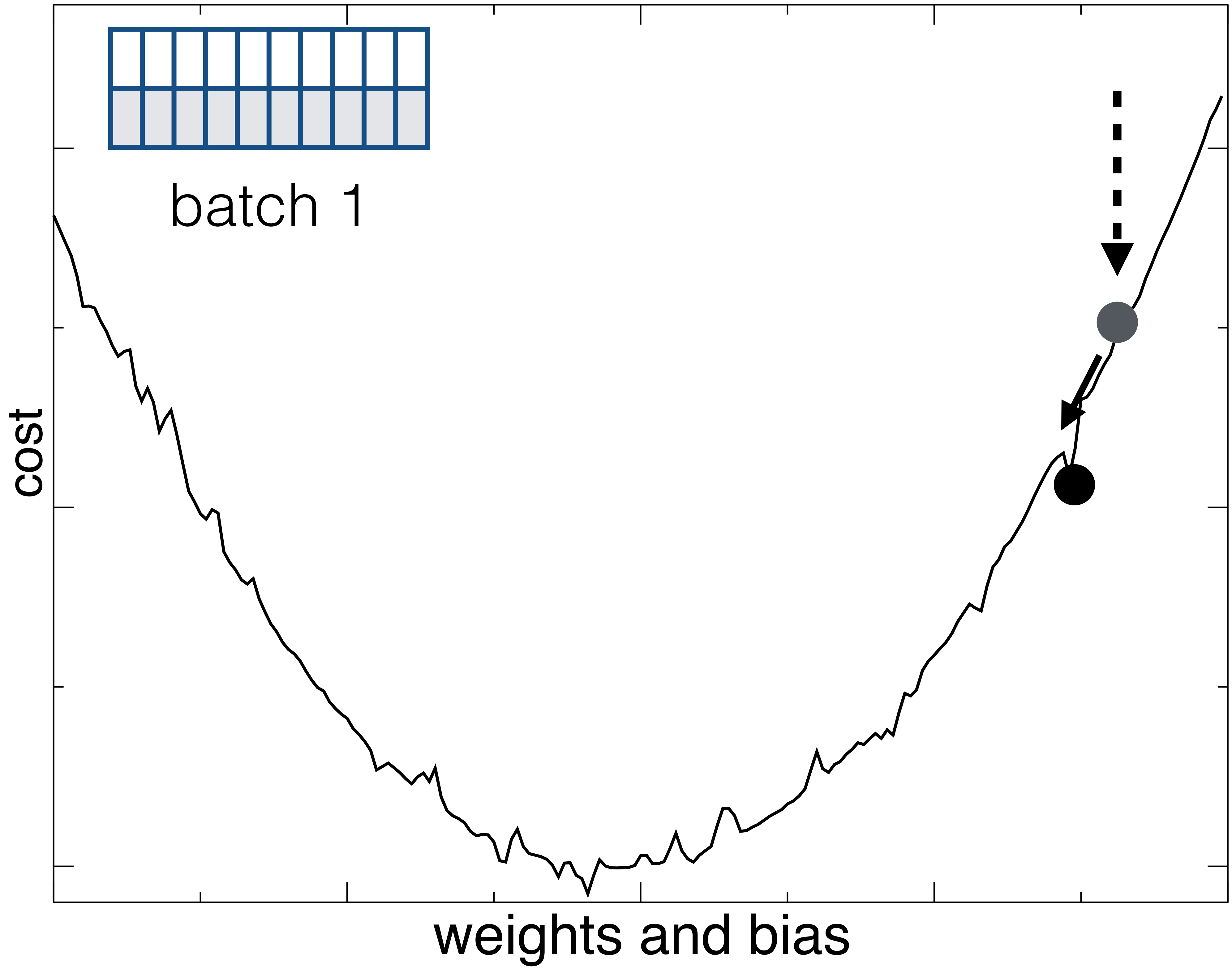
cost surfaces are different for different data sets

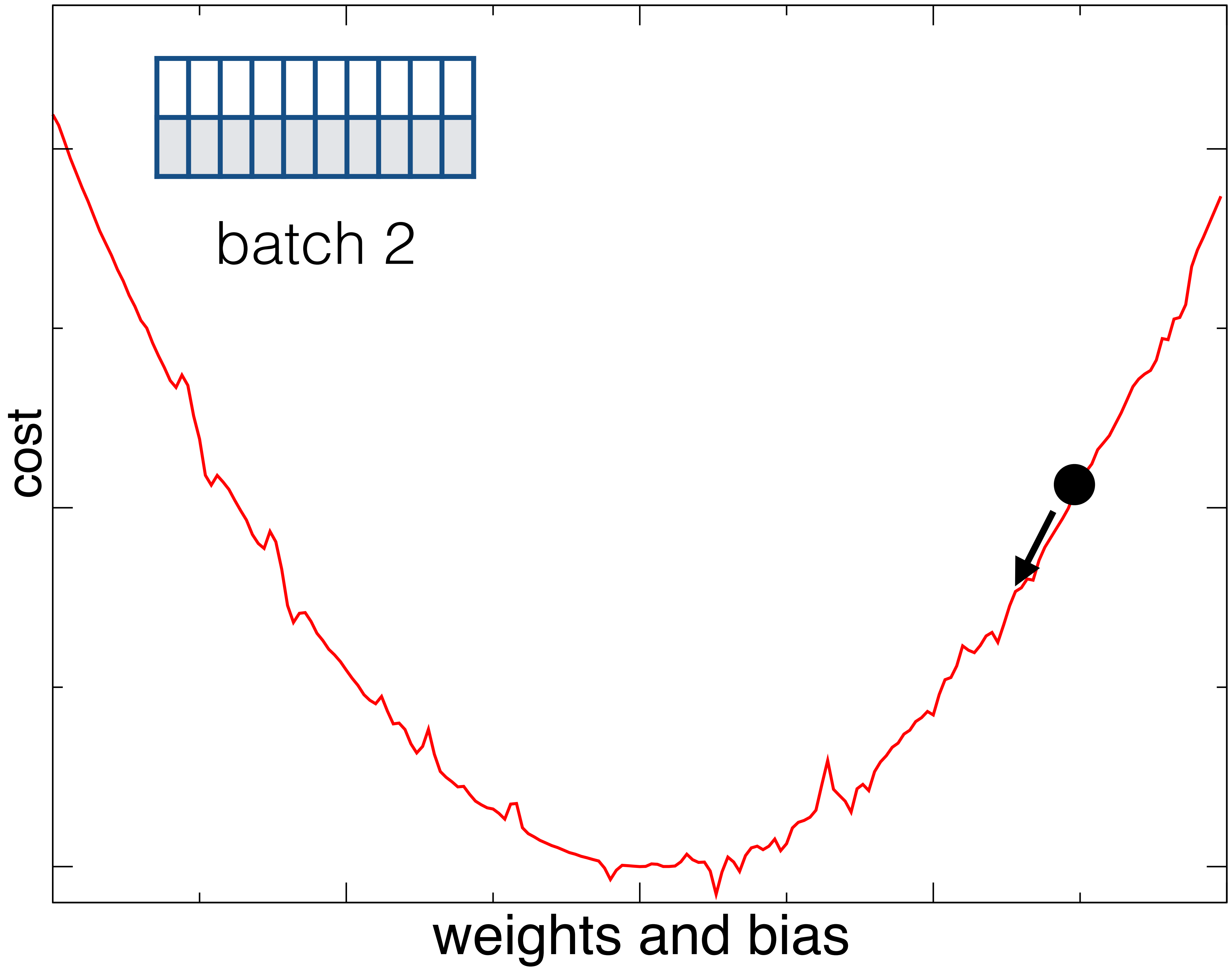


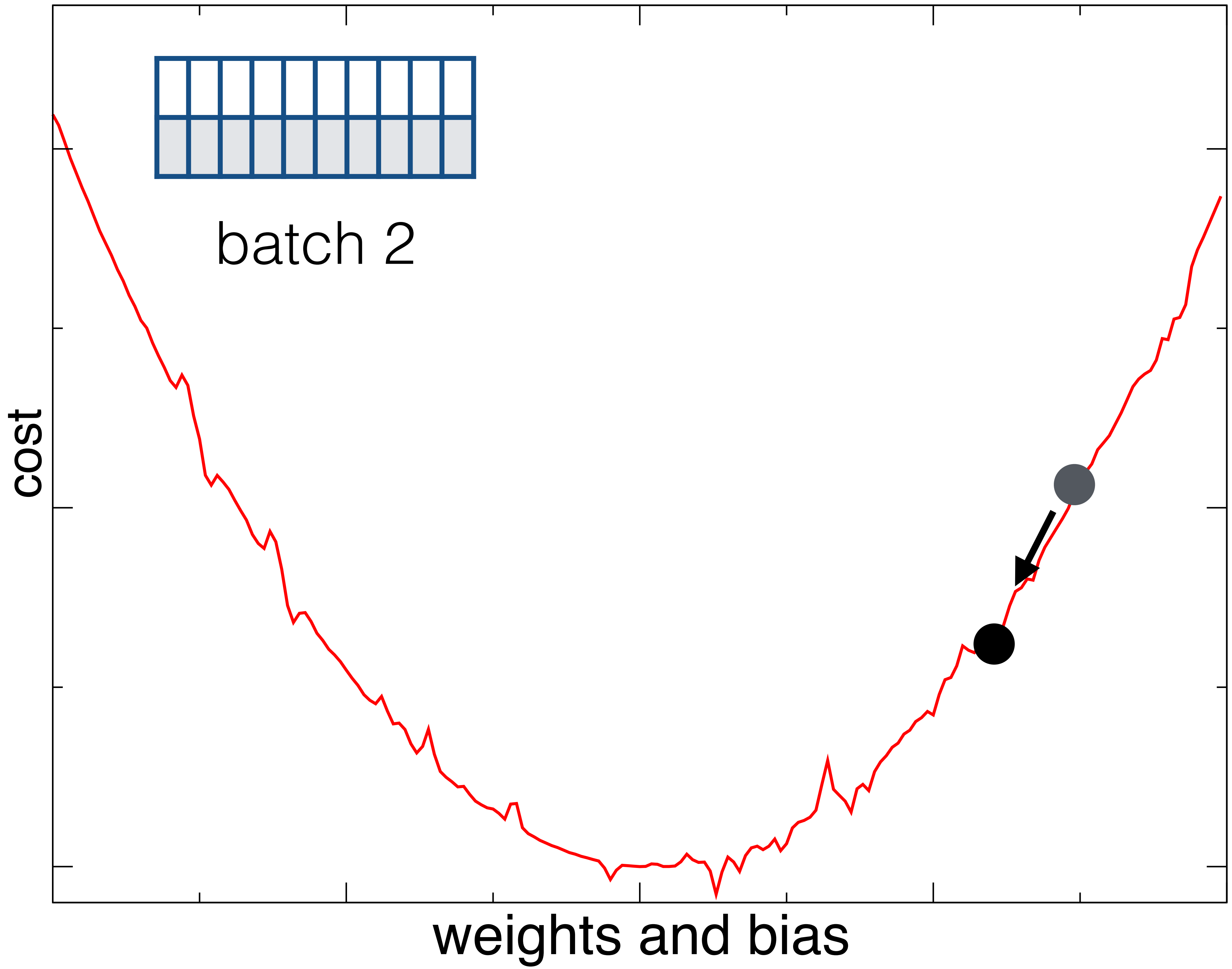
batch 1
cost
weights and bias
cost
weights and bias
batch 2
cost
weights and bias
batch 3
cost
weights and bias
batch 4

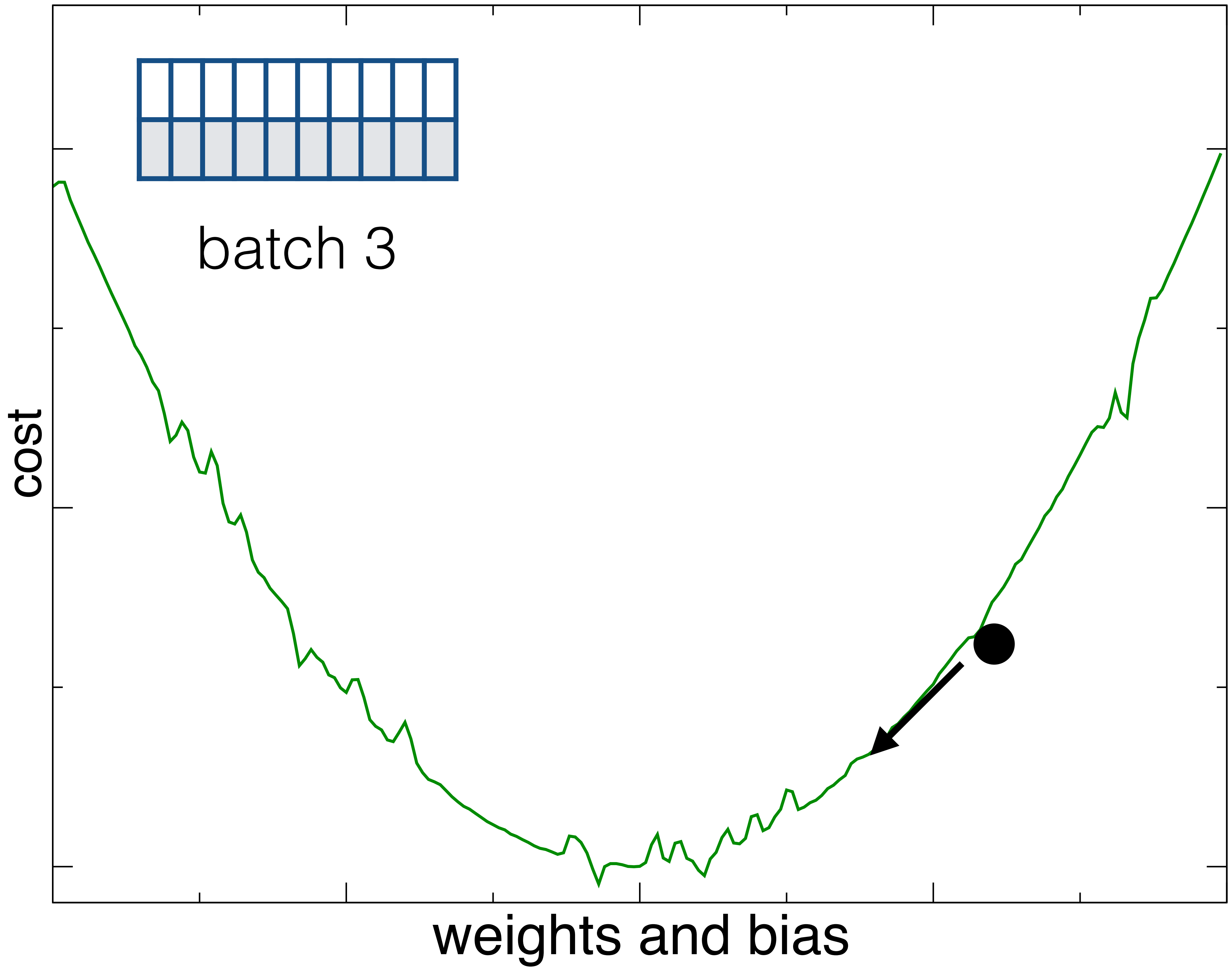
Always remember to shuffle the data

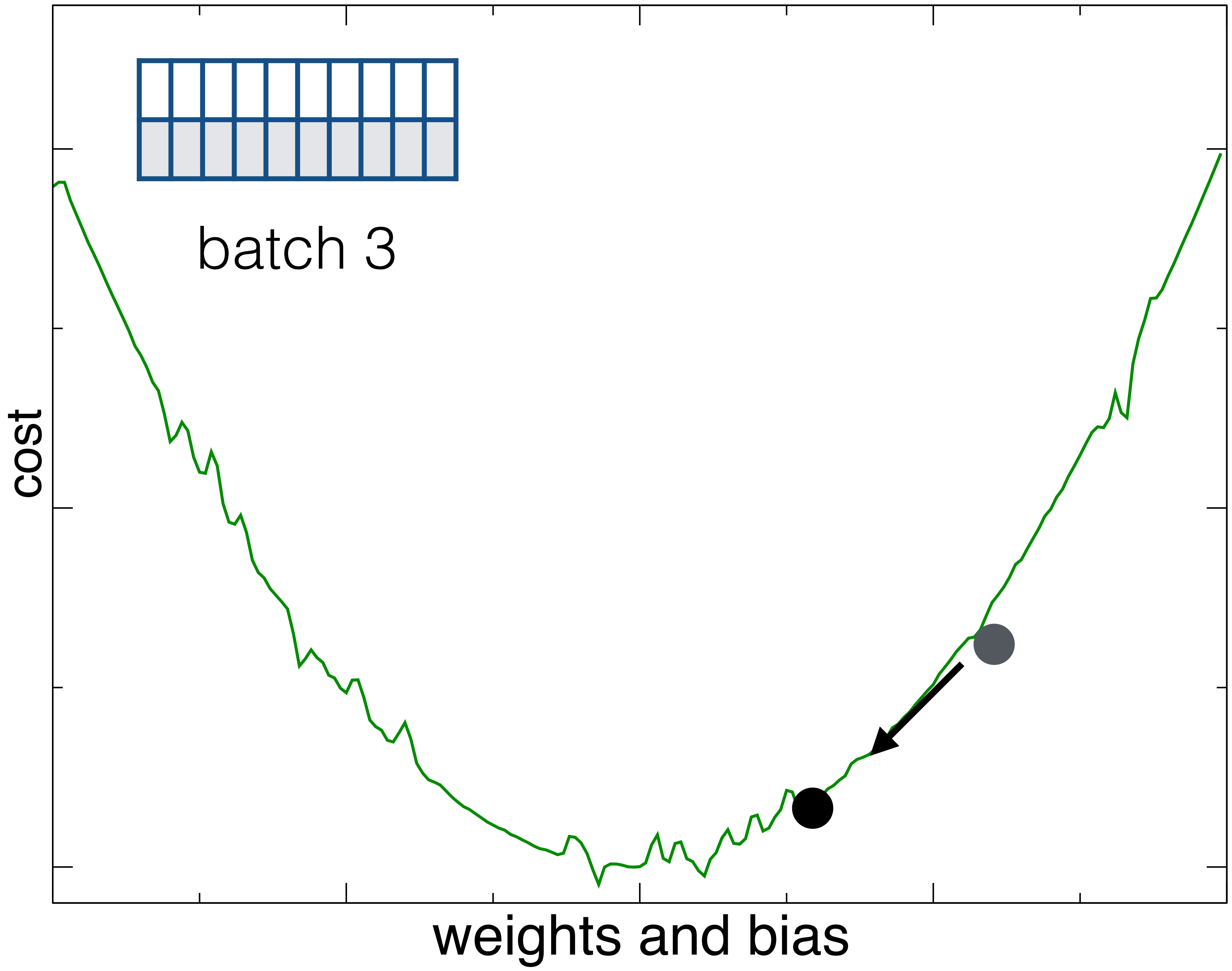


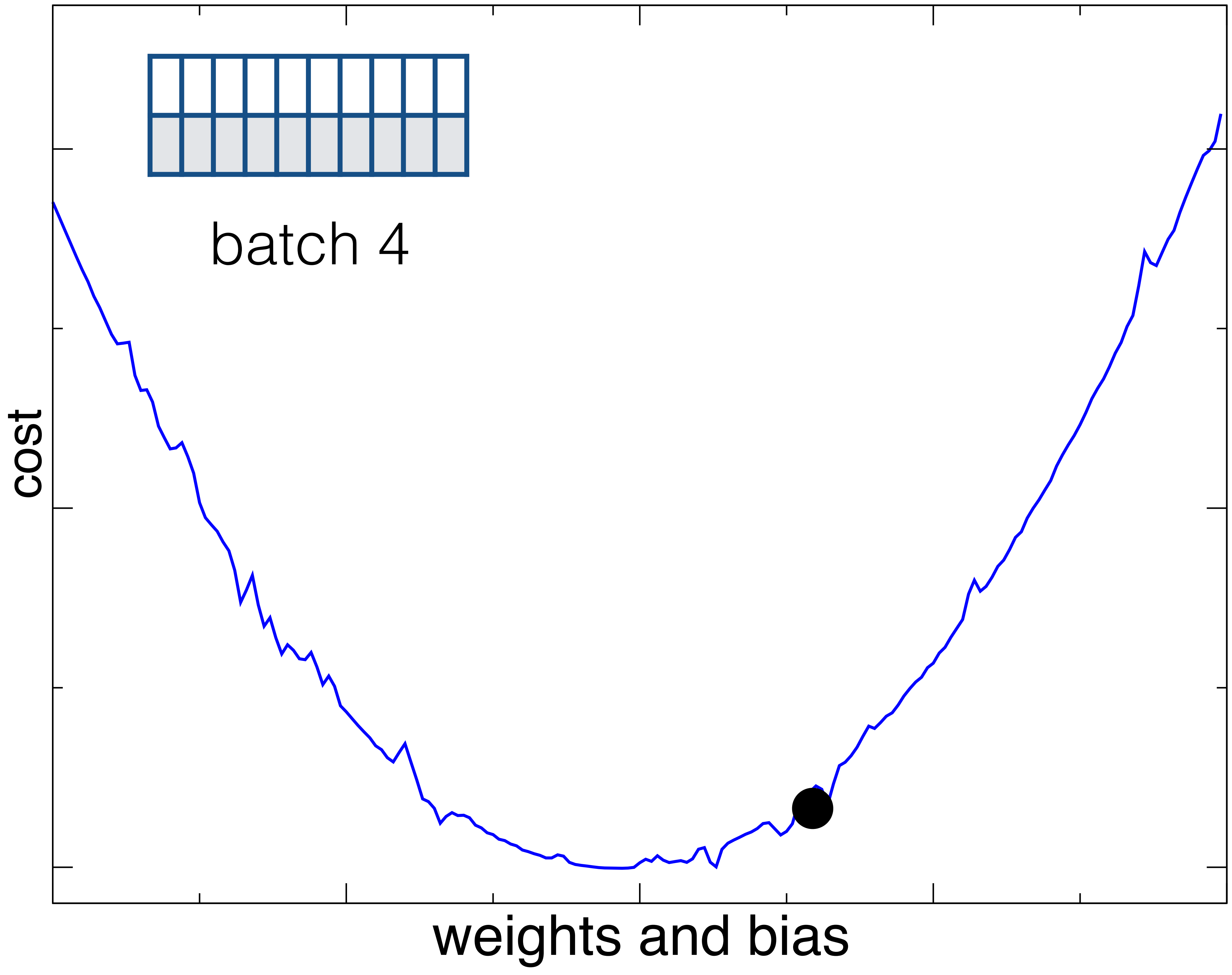

batch 1
cost
weights and bias


batch 2
cost
weights and bias

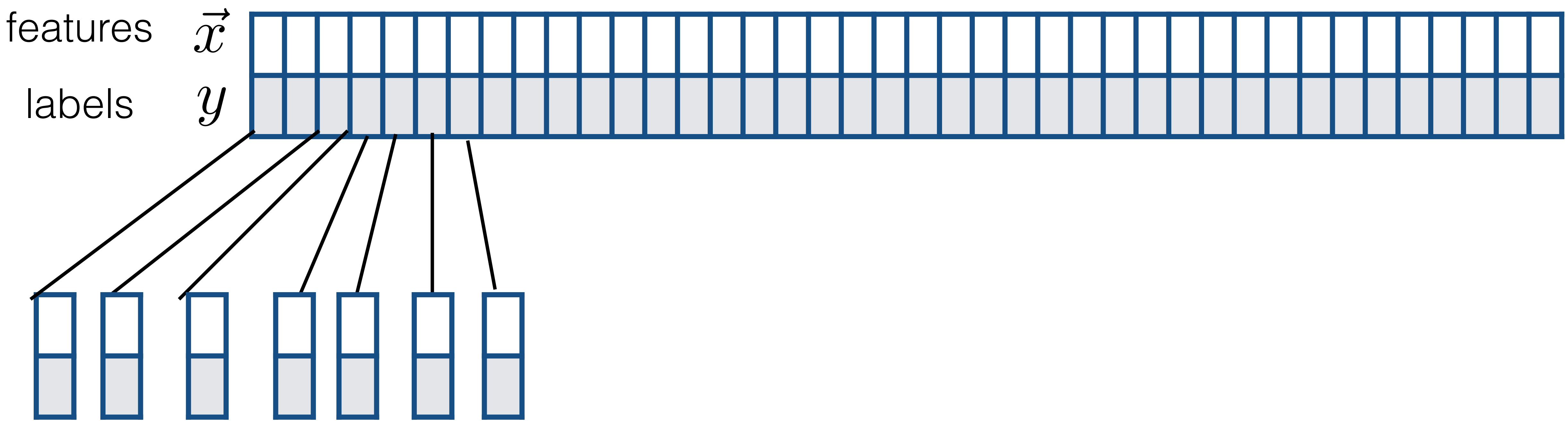

batch 2
cost
weights and bias


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

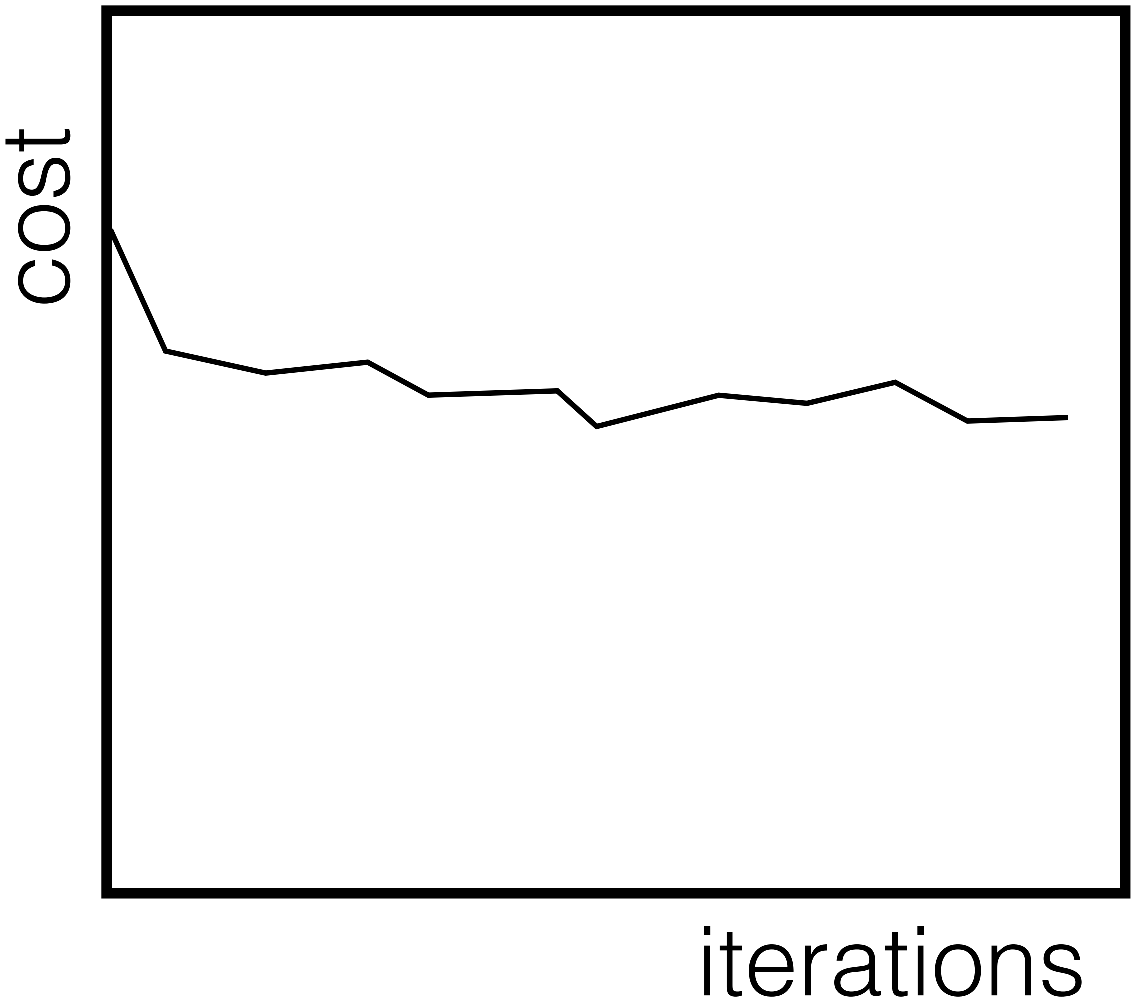
Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ 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)

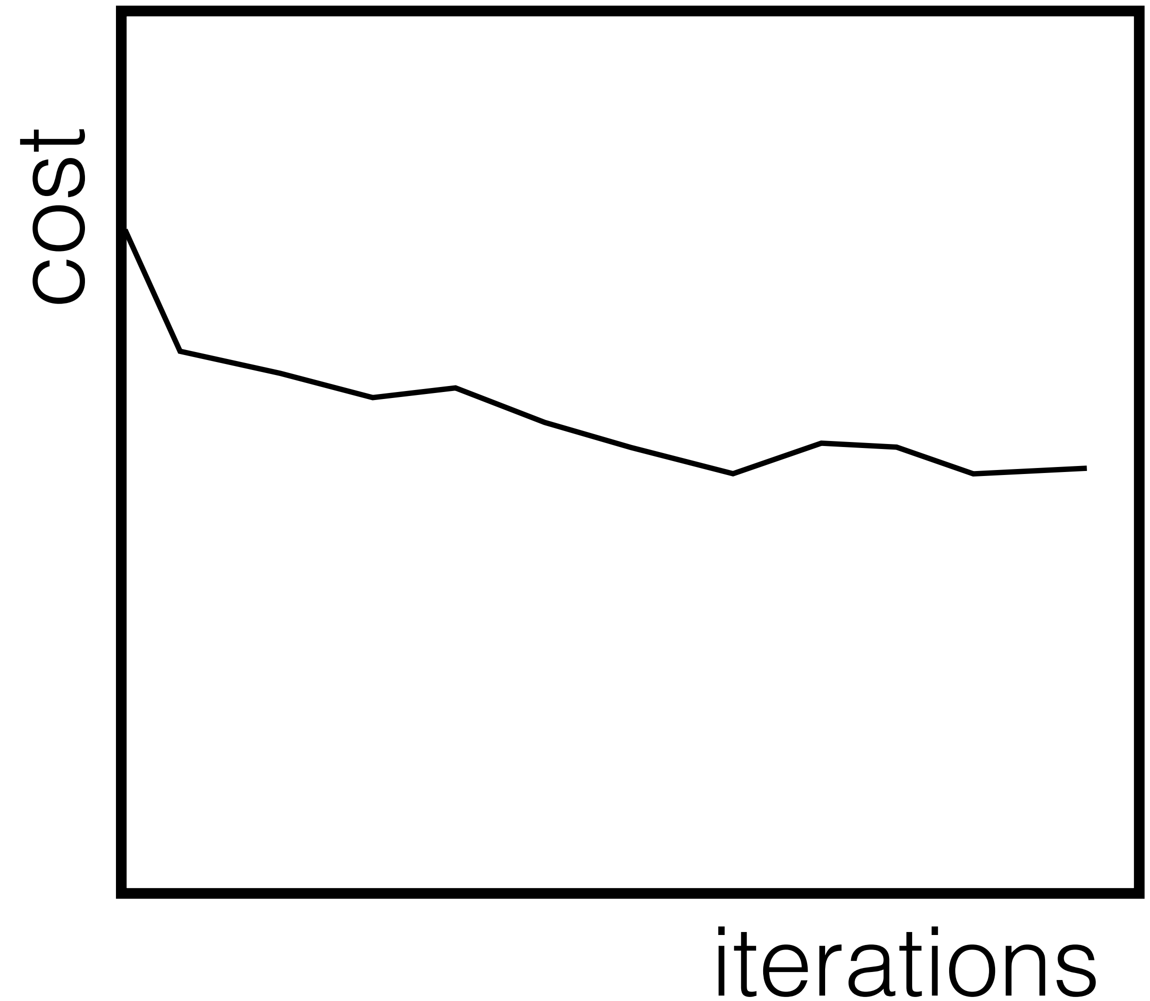
average the gradient direction over the past

move in the direction with "constant" step size

Signs of trouble
always look at cost versus iterations plots



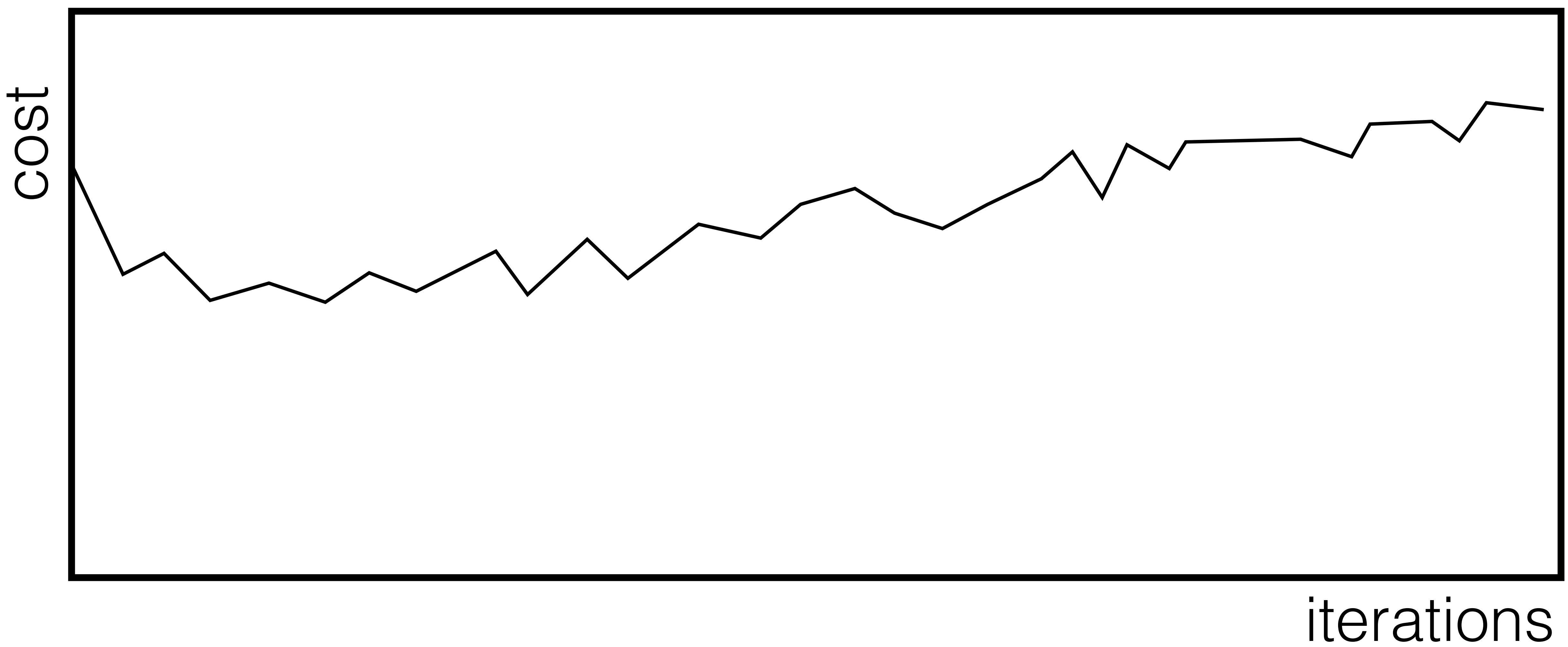
Cost not decreasing
looks like a local minimum



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

Signs of trouble

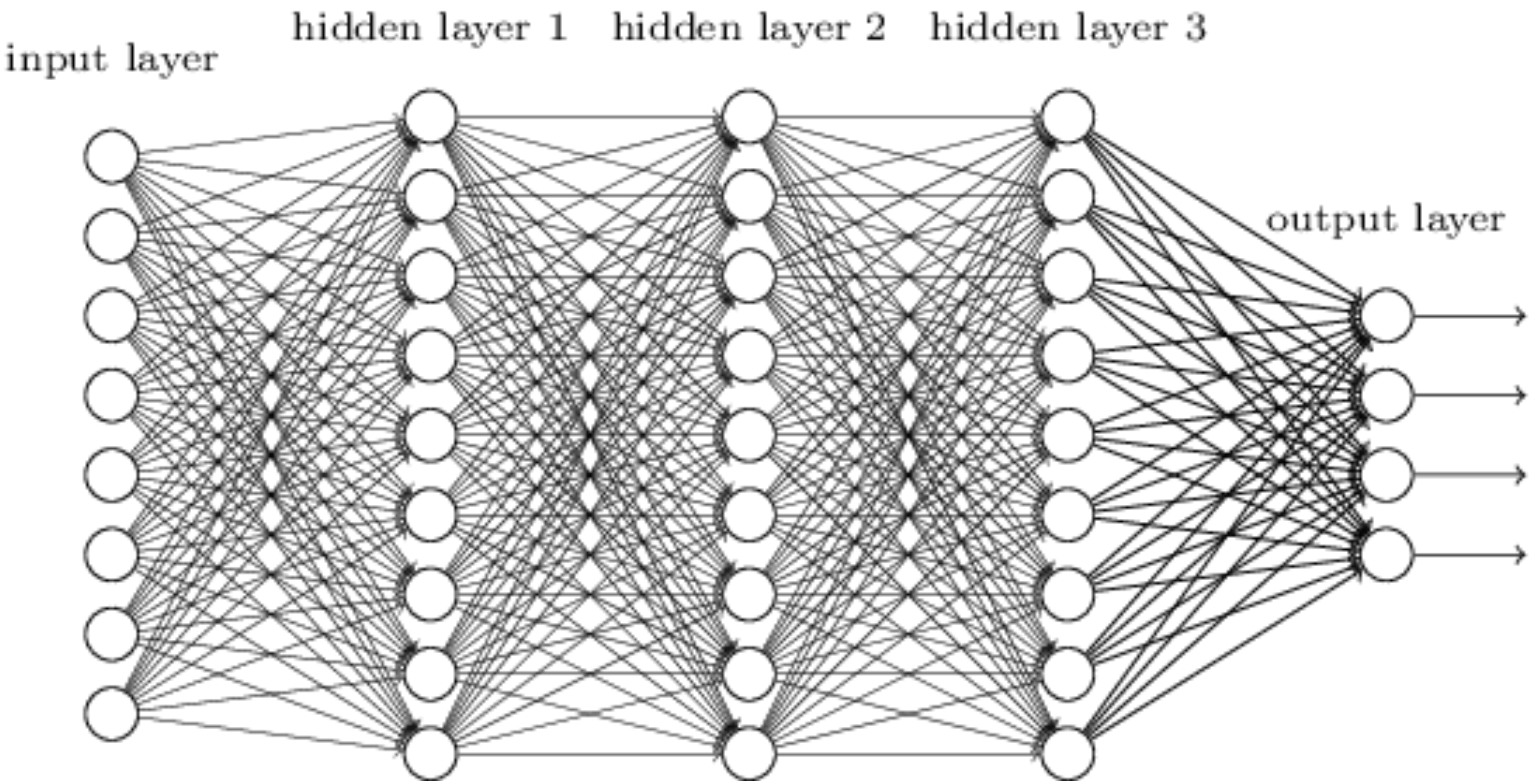
always look at cost versus iterations plots



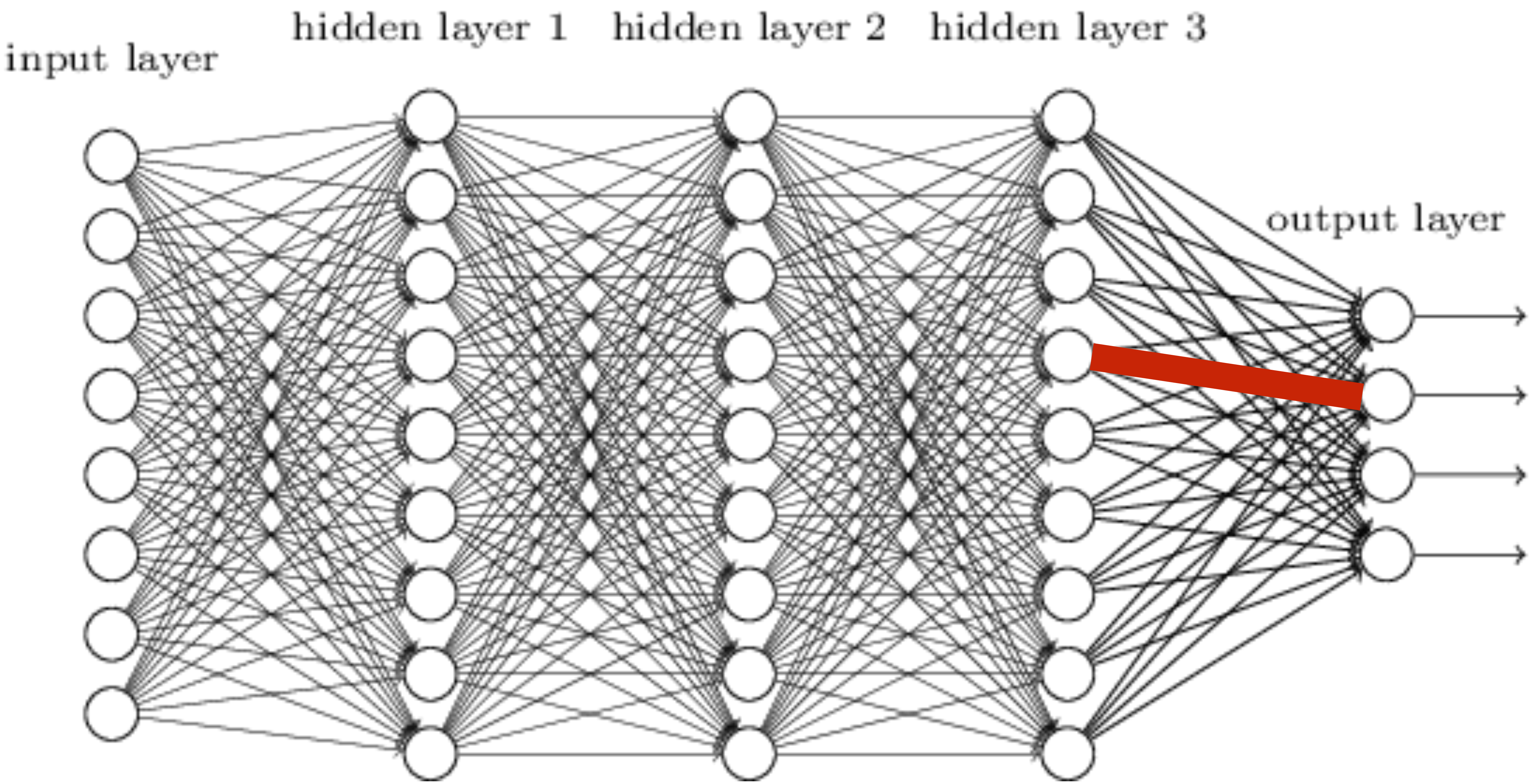
Cost actually increasing

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

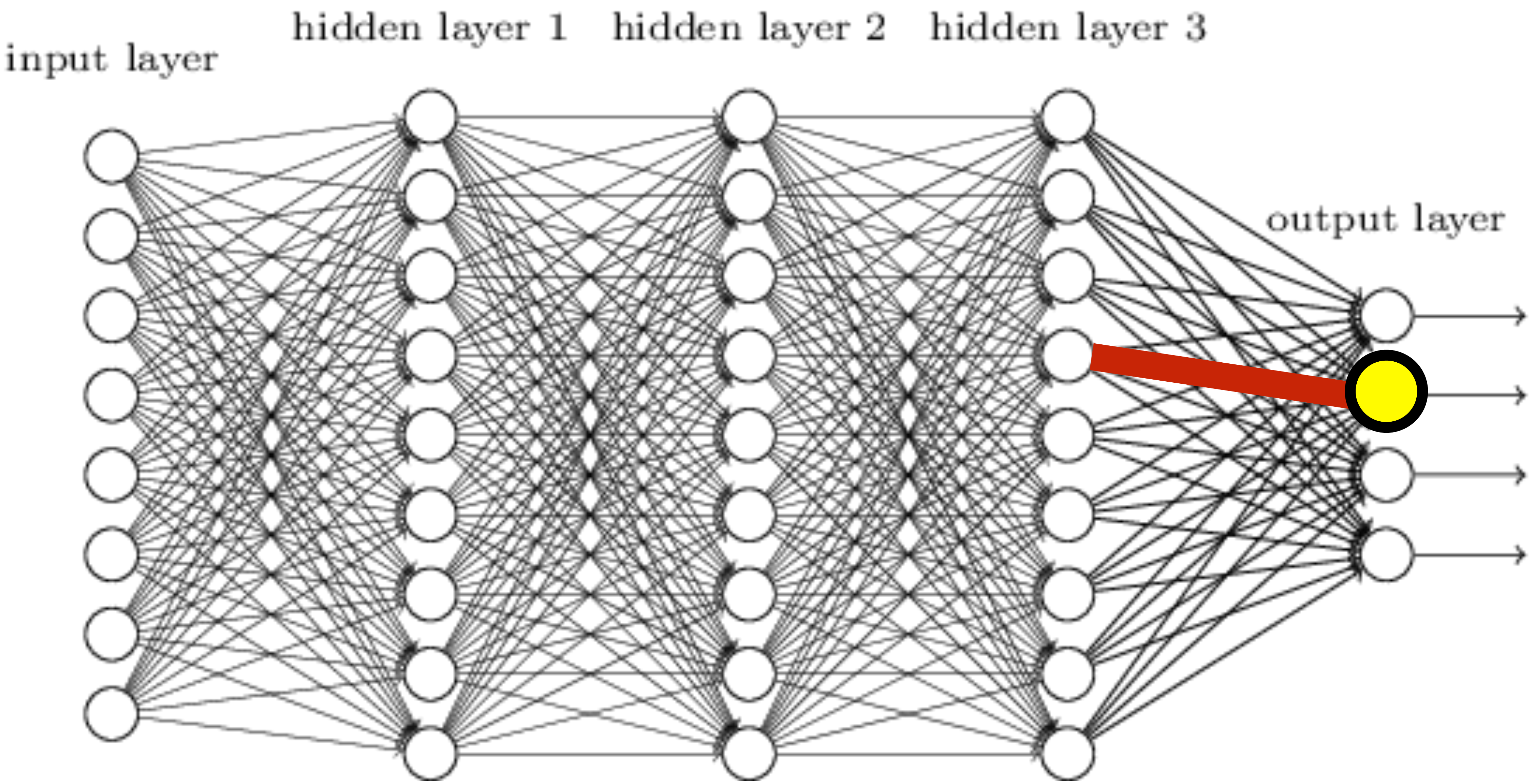
Vanishing gradient problem



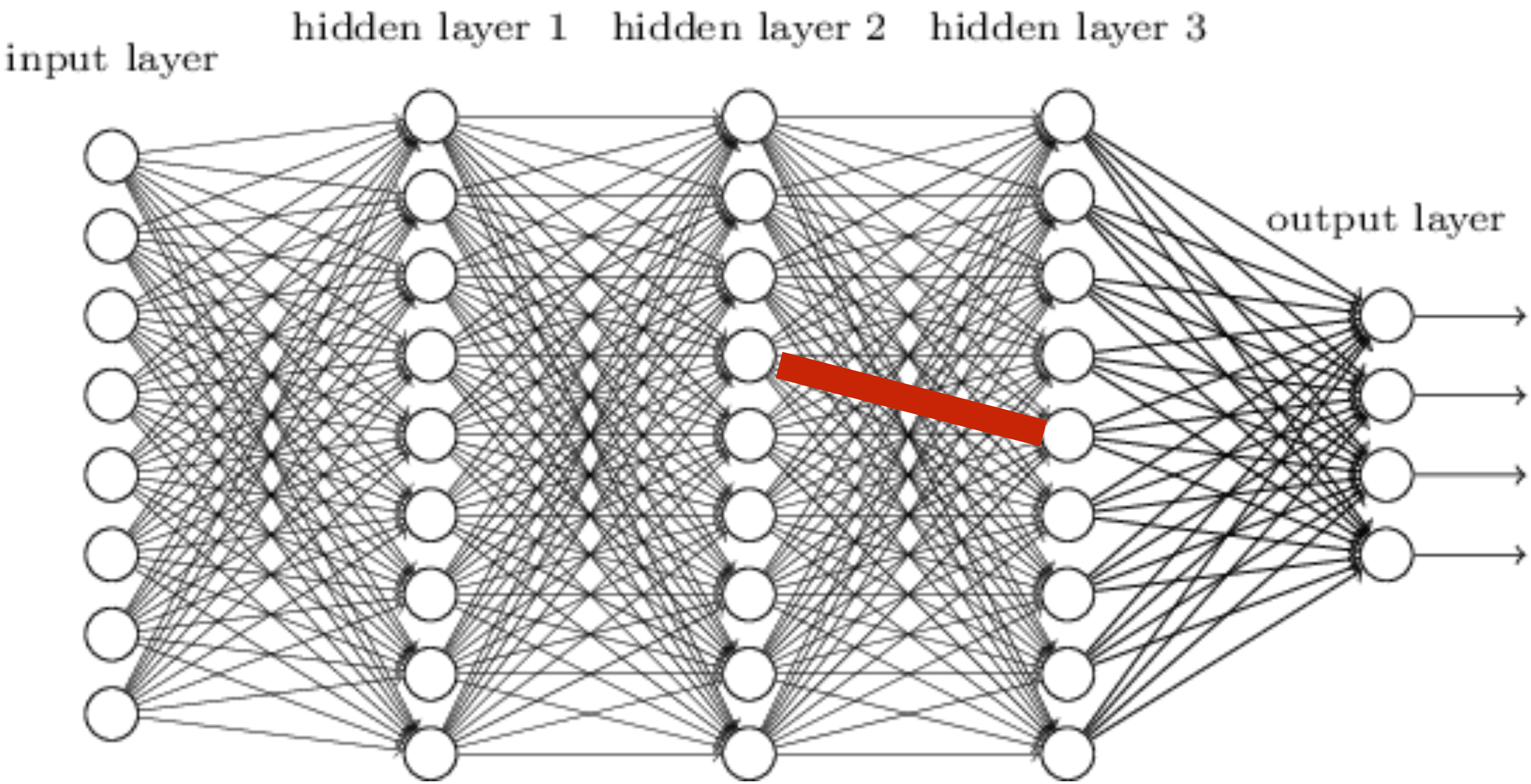
https://stats.stackexchange.com/questions/234891/difference-between-convolution-neural-network-and-deep-learning



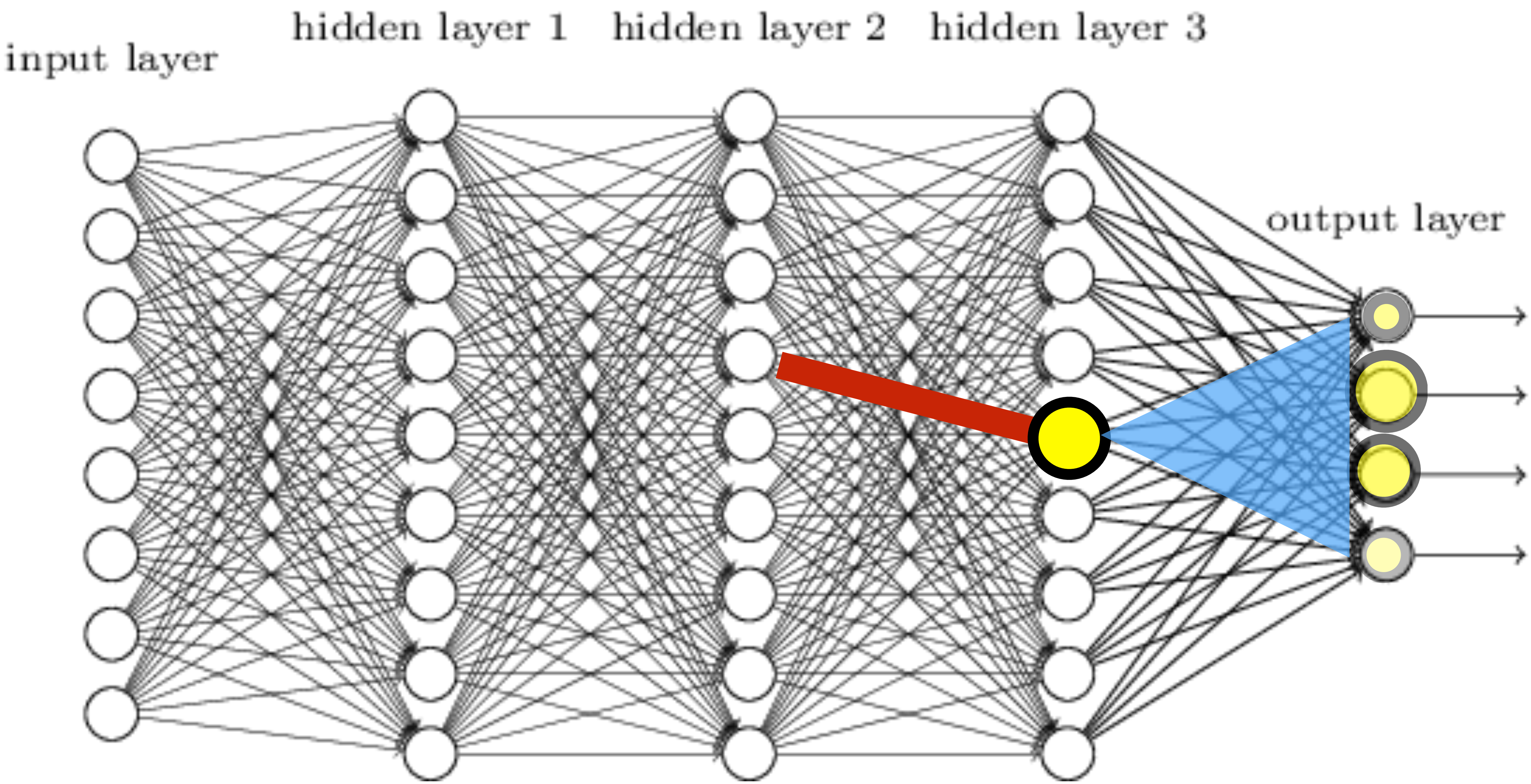
https://stats.stackexchange.com/questions/234891/difference-between-convolution-neural-network-and-deep-learning



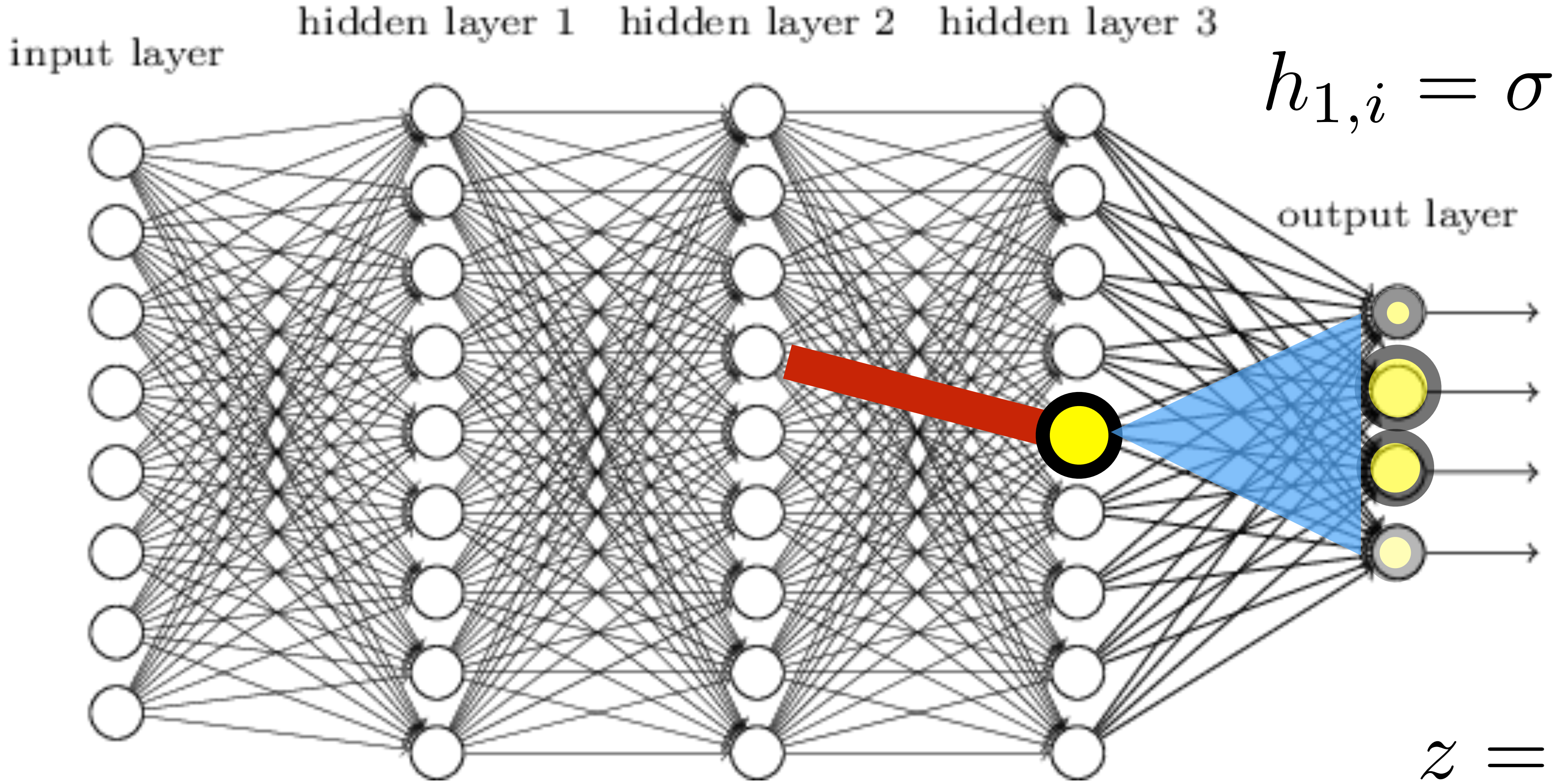
https://stats.stackexchange.com/questions/234891/difference-between-convolution-neural-network-and-deep-learning



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

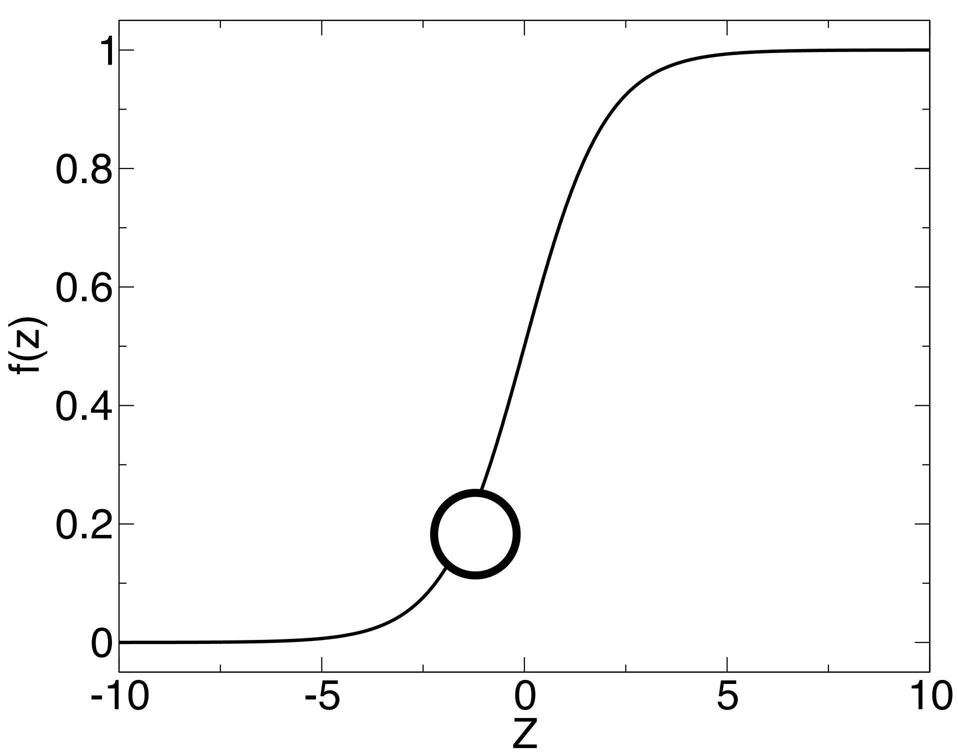


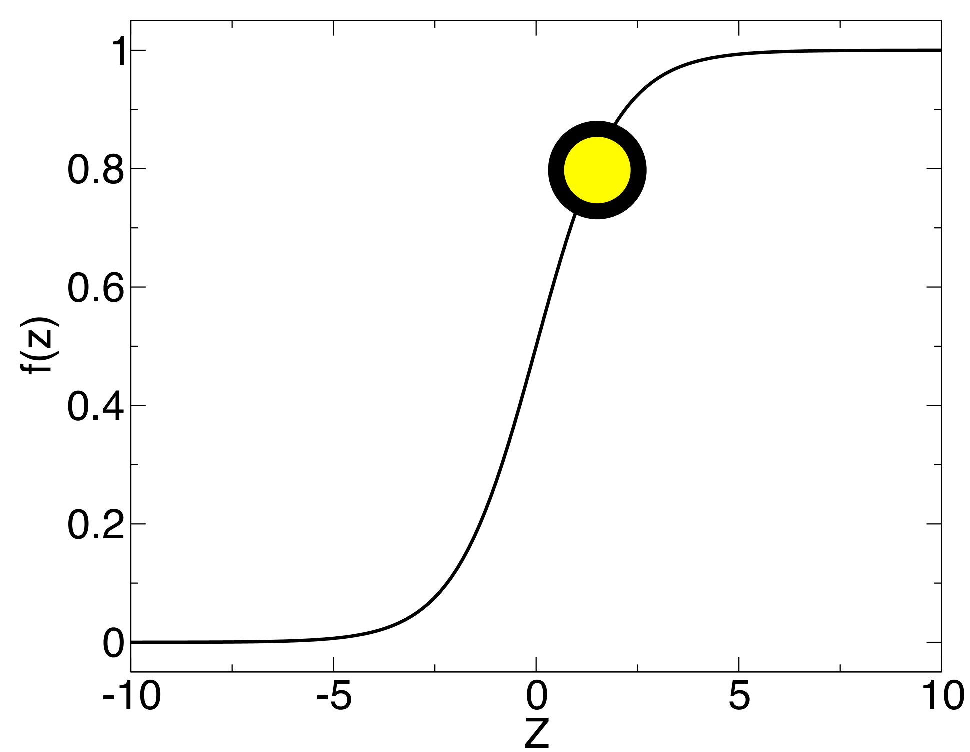
https://stats.stackexchange.com/questions/234891/difference-between-convolution-neural-network-and-deep-learning



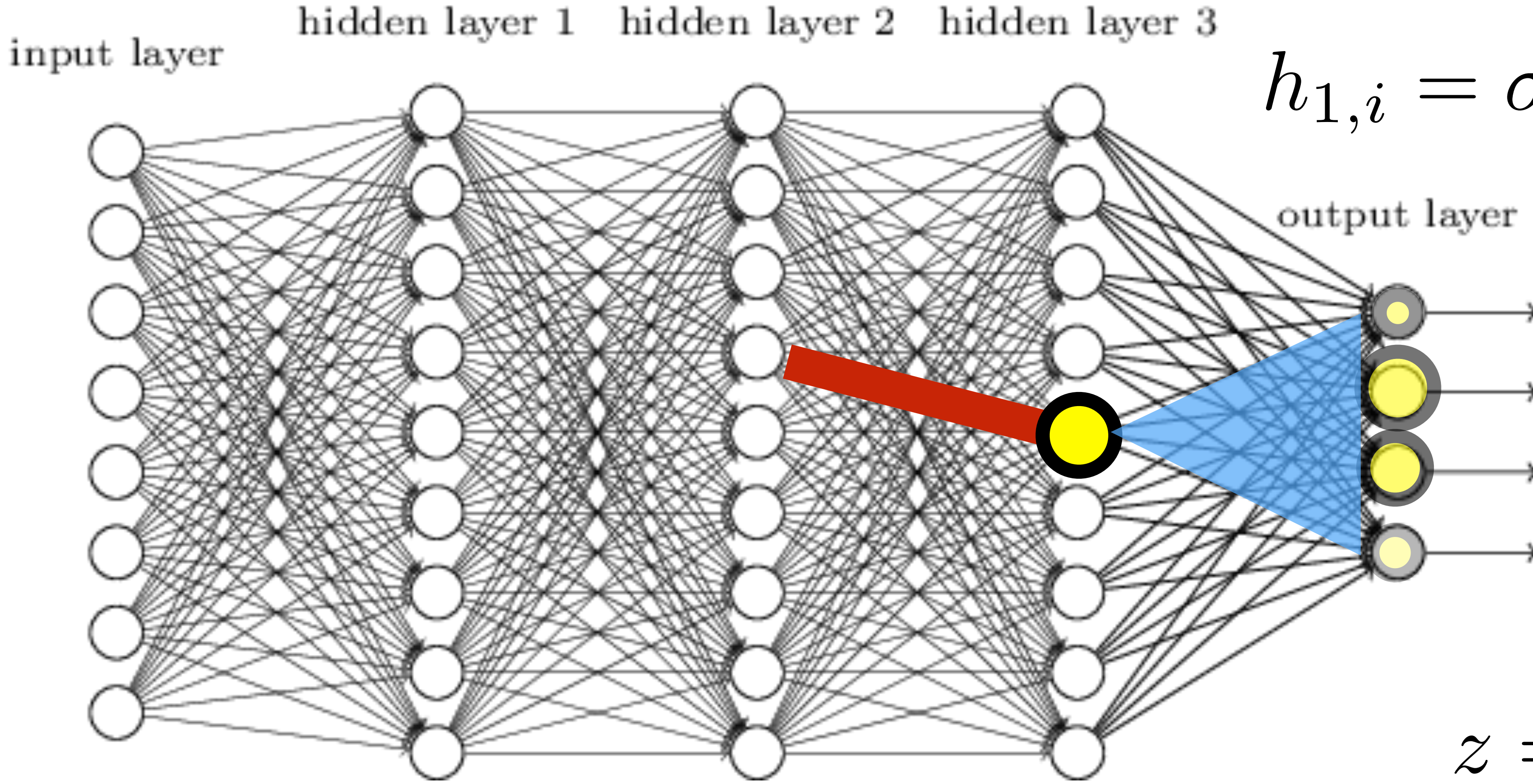
$$h_{1,i} = \sigma\left(\sum_j W_{j,i}^{(0,1)} x_j + b_i\right)$$

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



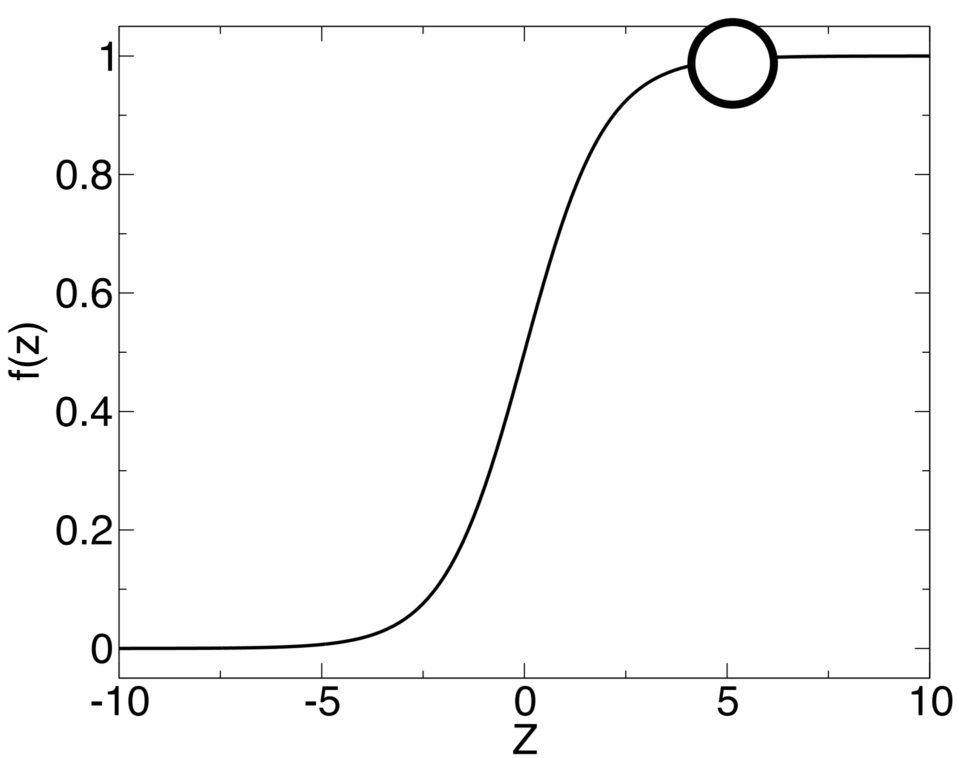


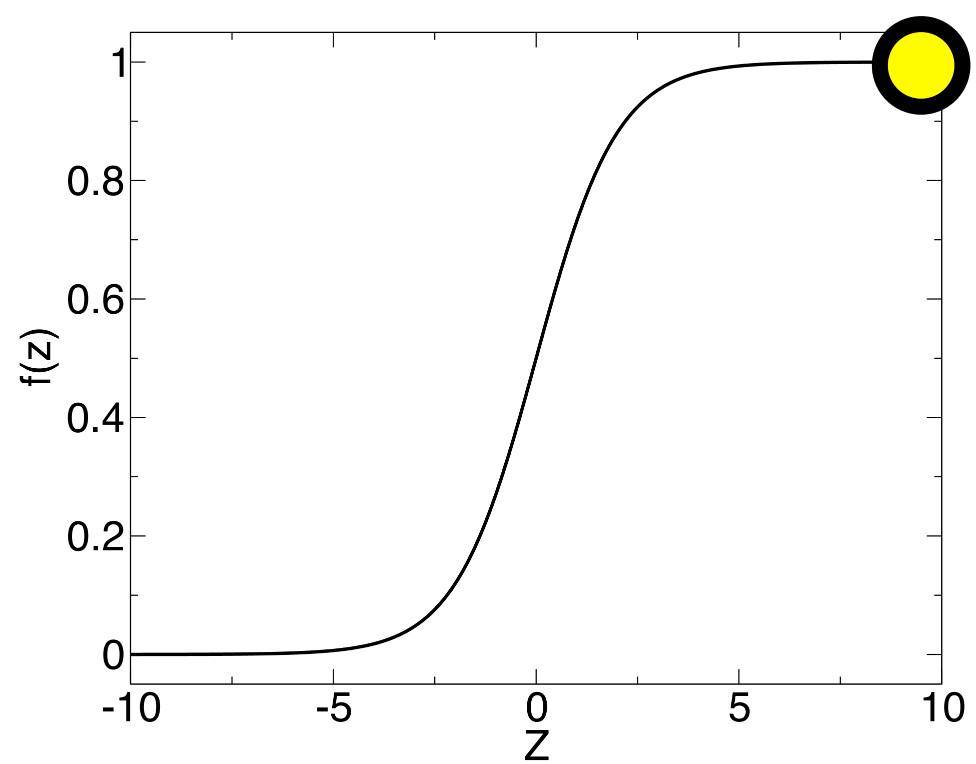
https://stats.stackexchange.com/questions/234891/difference-between-convolution-neural-network-and-deep-learning



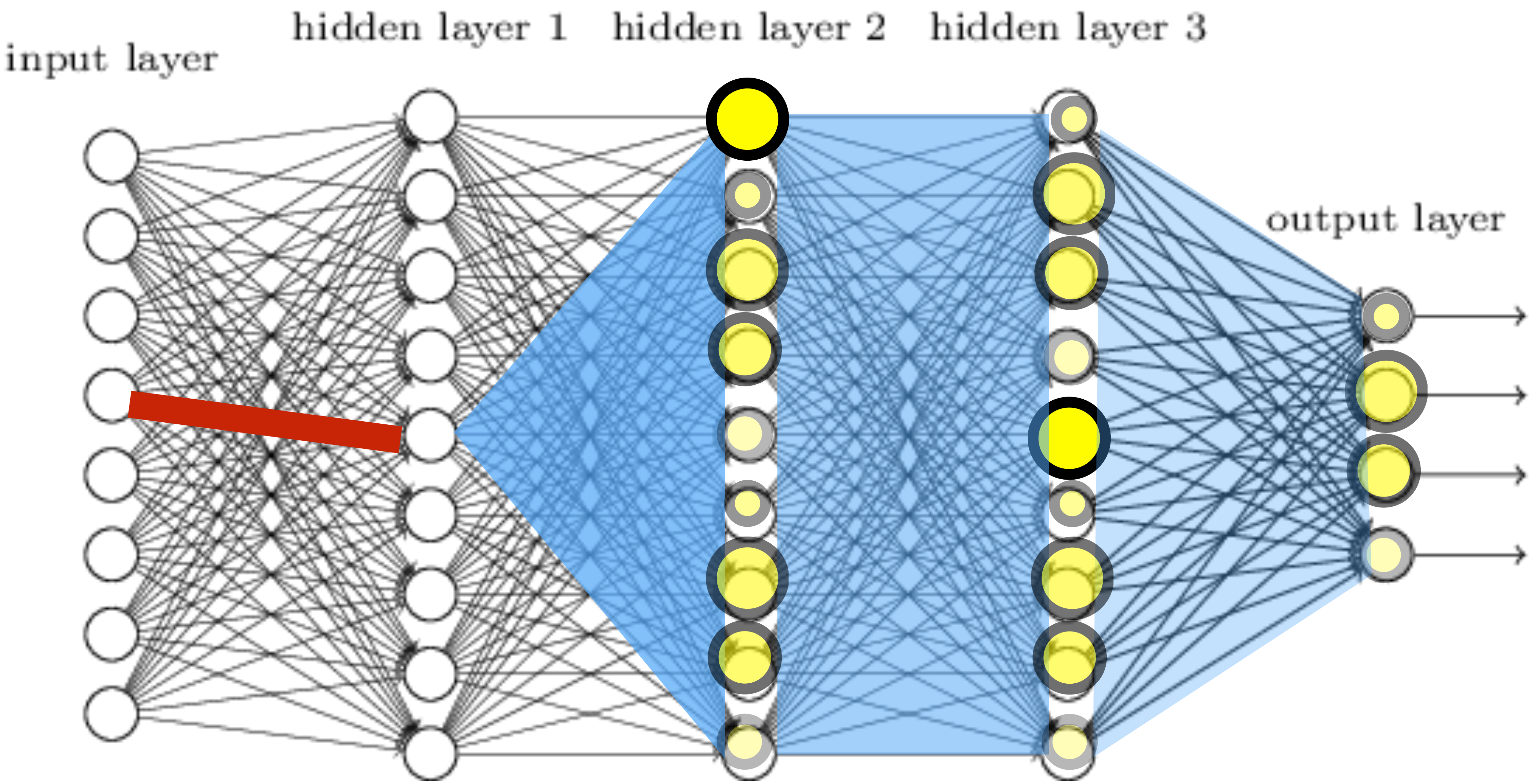
$$h_{1,i} = \sigma\left(\sum_j W_{j,i}^{(0,1)} x_j + b_i\right)$$

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

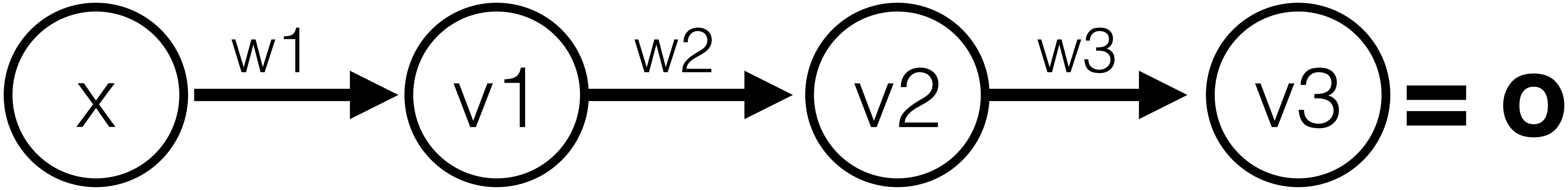




https://stats.stackexchange.com/questions/234891/difference-between-convolution-neural-network-and-deep-learning



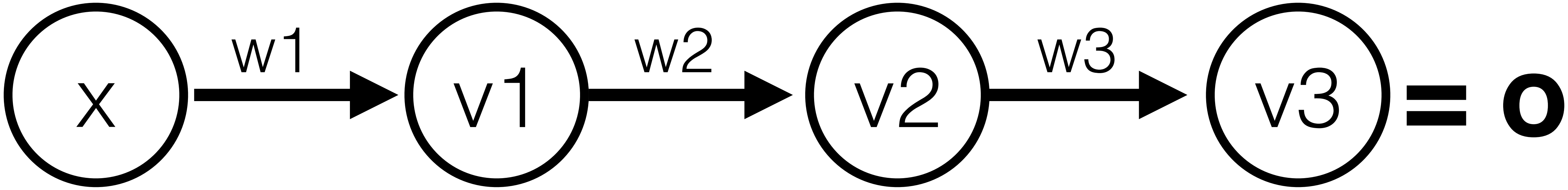
https://stats.stackexchange.com/questions/234891/difference-between-convolution-neural-network-and-deep-learning



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

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

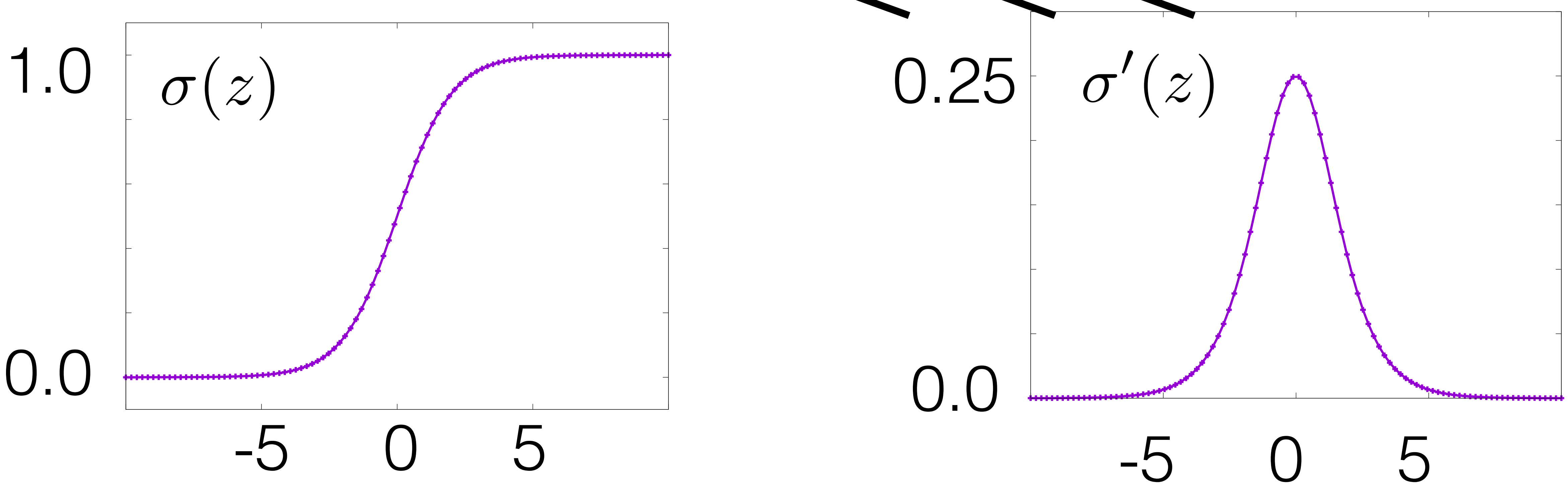
$$\begin{aligned}\frac{\partial v_3}{\partial w_1} &= \frac{\partial v_3}{\partial z_3}\frac{\partial z_3}{\partial w_1} = \sigma'(z_3)w_3\frac{\partial v_2}{\partial w_2} = \sigma'(z_3)w_3\sigma'(z_2)w_2\frac{\partial v_1}{\partial w_1}\\ &= \sigma'(z_3)w_3\sigma'(z_2)w_2\sigma'(z_1)x\\ &= \sigma'(z_3)\sigma'(z_2)\sigma'(z_1)w_3w_2x\end{aligned}$$



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

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

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



Strategies to overcoming vanishing gradient problem

short-cuts (residual net)

these will be covered later in the course

Lets play a game

You guess a number, if it is a ‘good’ number, I pay you $1, else you pay me $1.

I have a hidden rule to define what is a good number. . .

Of course I am not telling you my rule

What you can know if you keep buying until you find out the rule

We can assume that my rule does not change

9931
8937
You lose

1728
5952
You win

Guess what is my rule write on the board

9931
8937
You lose

1728
5952
You win

Guess what is my rule

- .odd/even
- .prime ***
- .>6000
- .div3
- .last 2 digit even
- .div12
- .include 3 -> bad
- .sum digit <=21
- .have 2 as a digit
- .first 2 digit is prime

9931 8937 6222 0328 1002	You lose	1728 5952 0064	You win

9931
8937
6222
0328
1002

You lose

1728
5952
0064

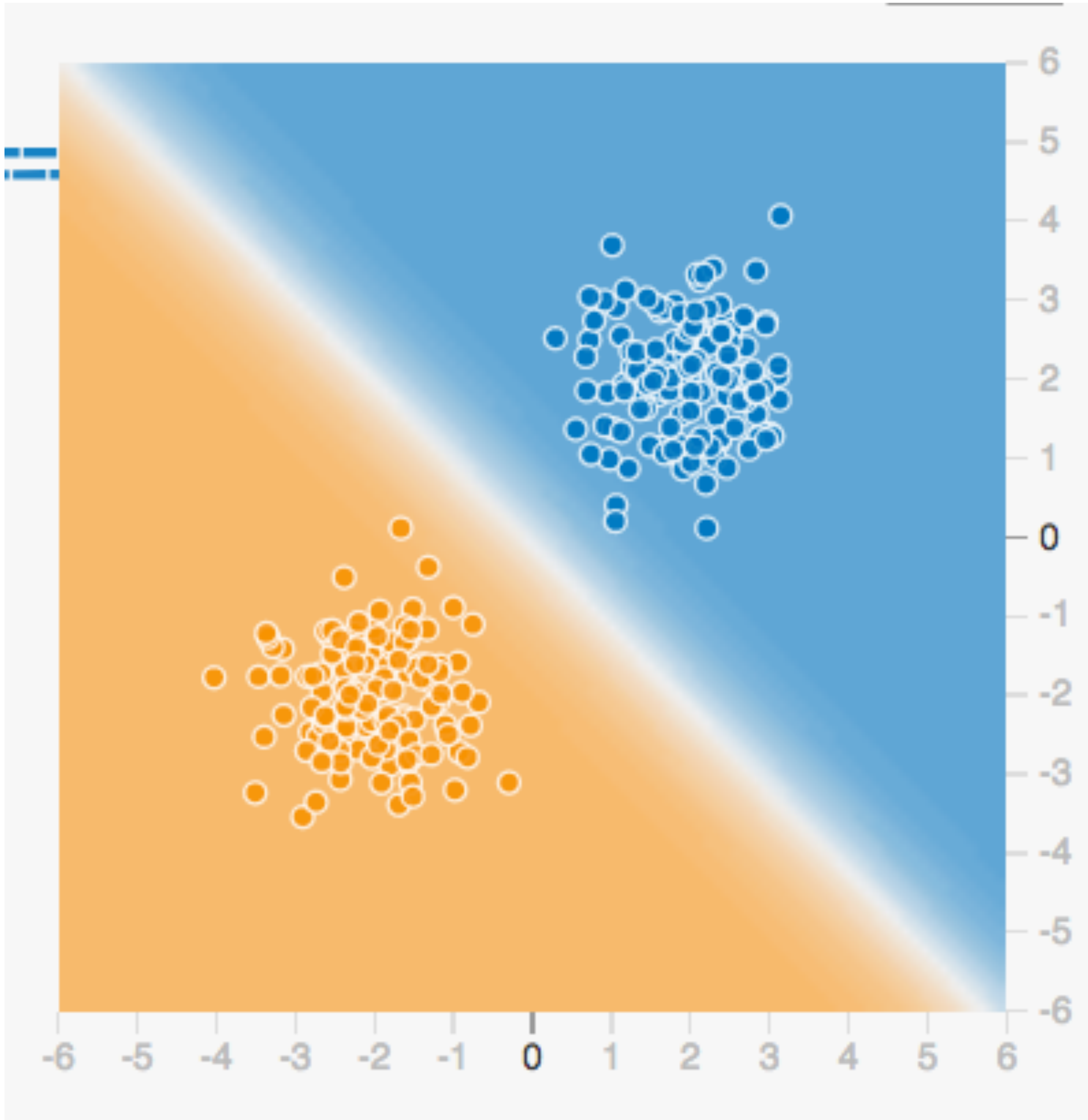
You win

Guess what is my rule, type in the chat please

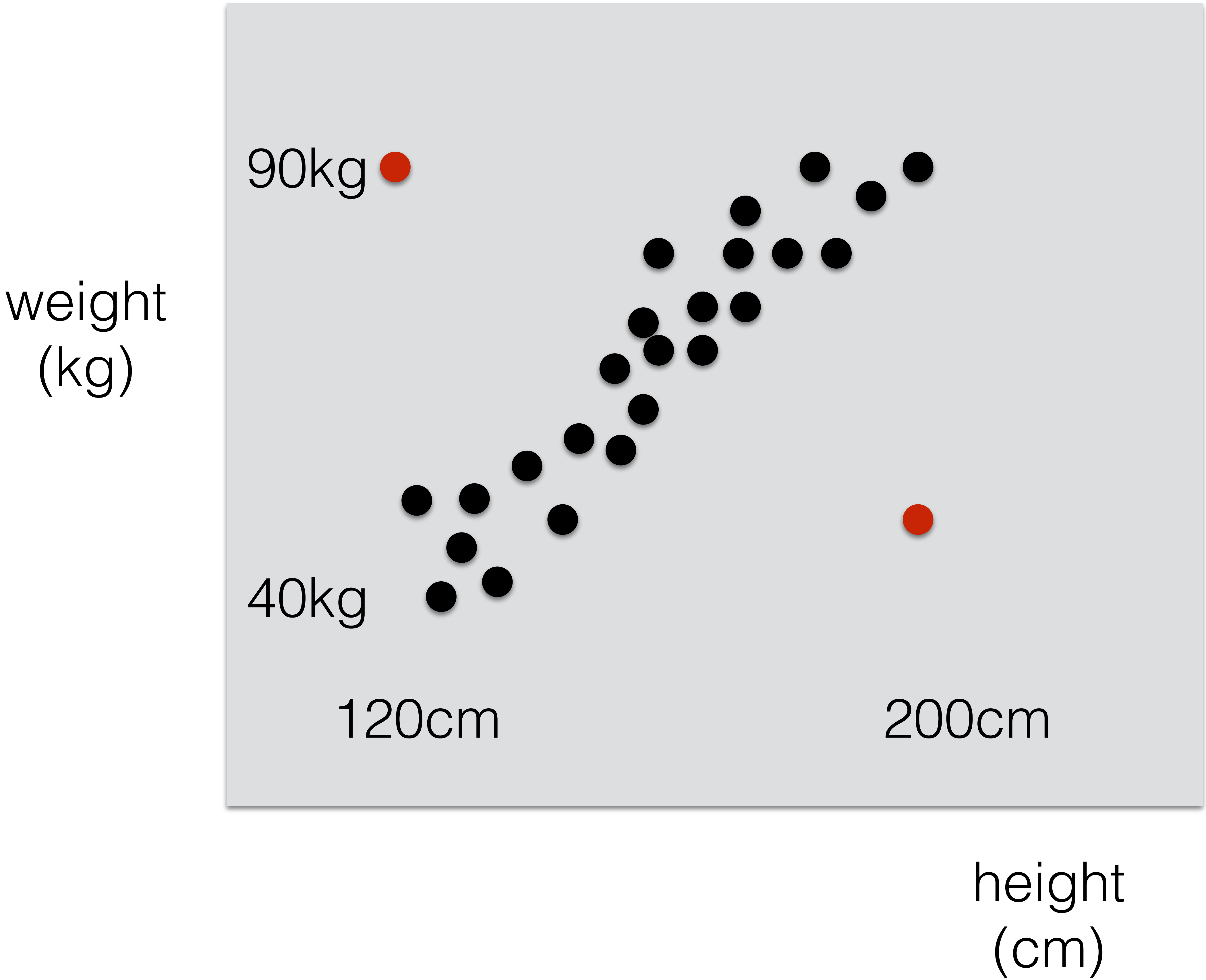
- .odd/even - eliminate
- .prime - eliminated
- .>6000 - eliminated
- .div3
- .last 2 digit even - eliminated
- .div12
- .include 3 -> bad - eliminated
- .sum digit <=21 - eliminated
- .have 2 as a digit - eliminated
- .first 2 digit is prime
- Contains 2^6

Fundamental problem of Neural Networks
Data space and data manifold

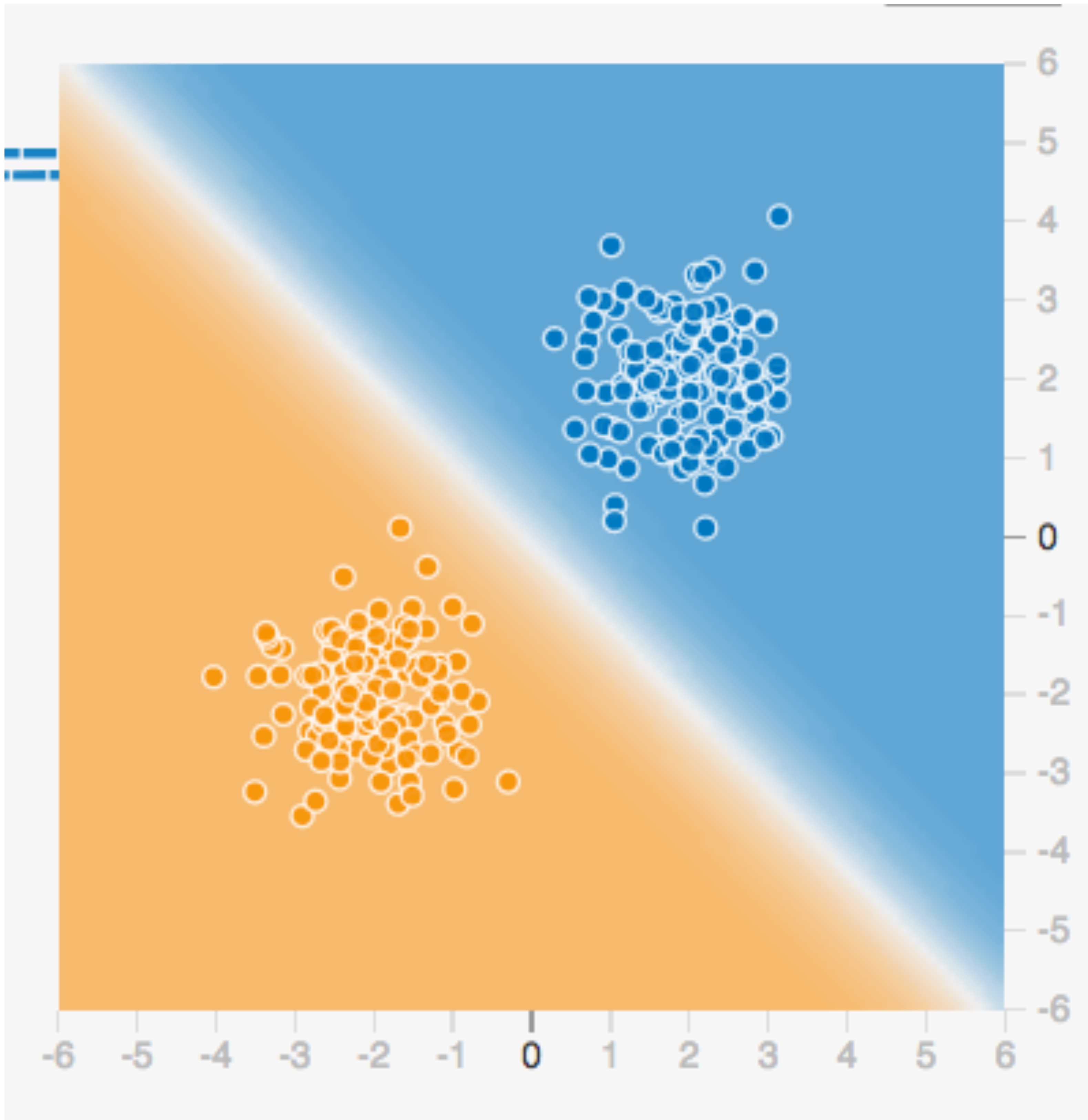
What is ‘wrong’ with this data set?



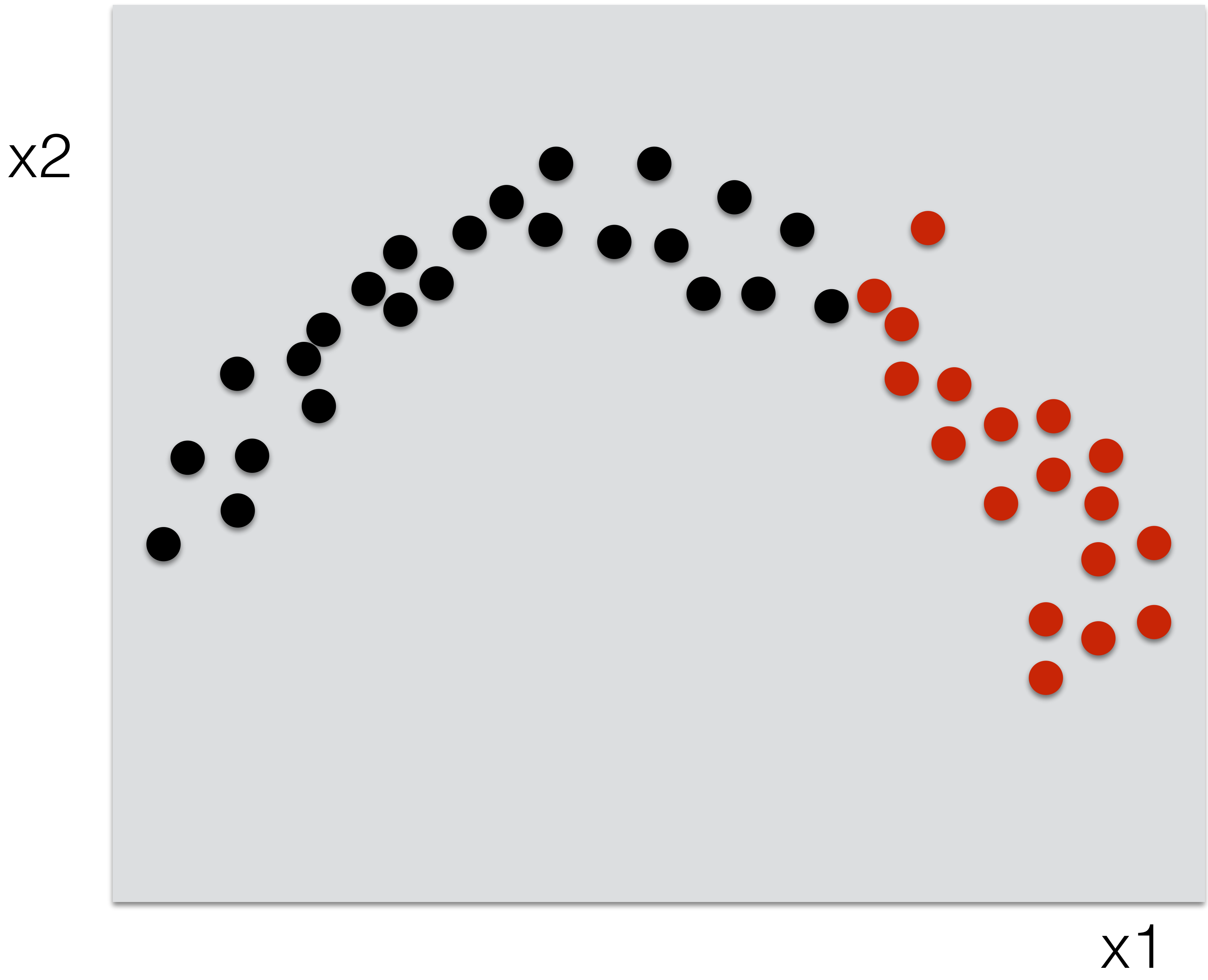
height and weight example



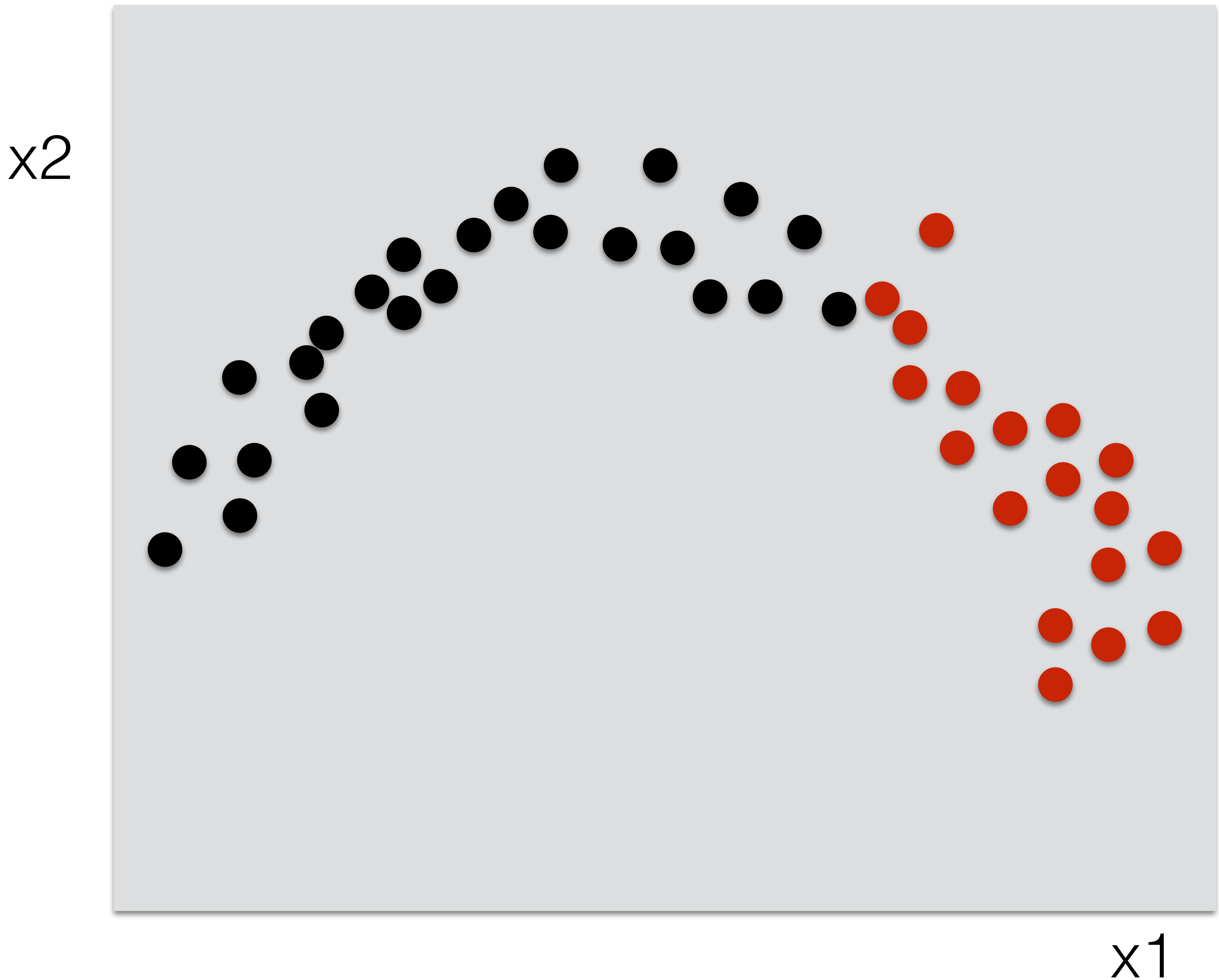
What is ‘wrong’ with this data set?



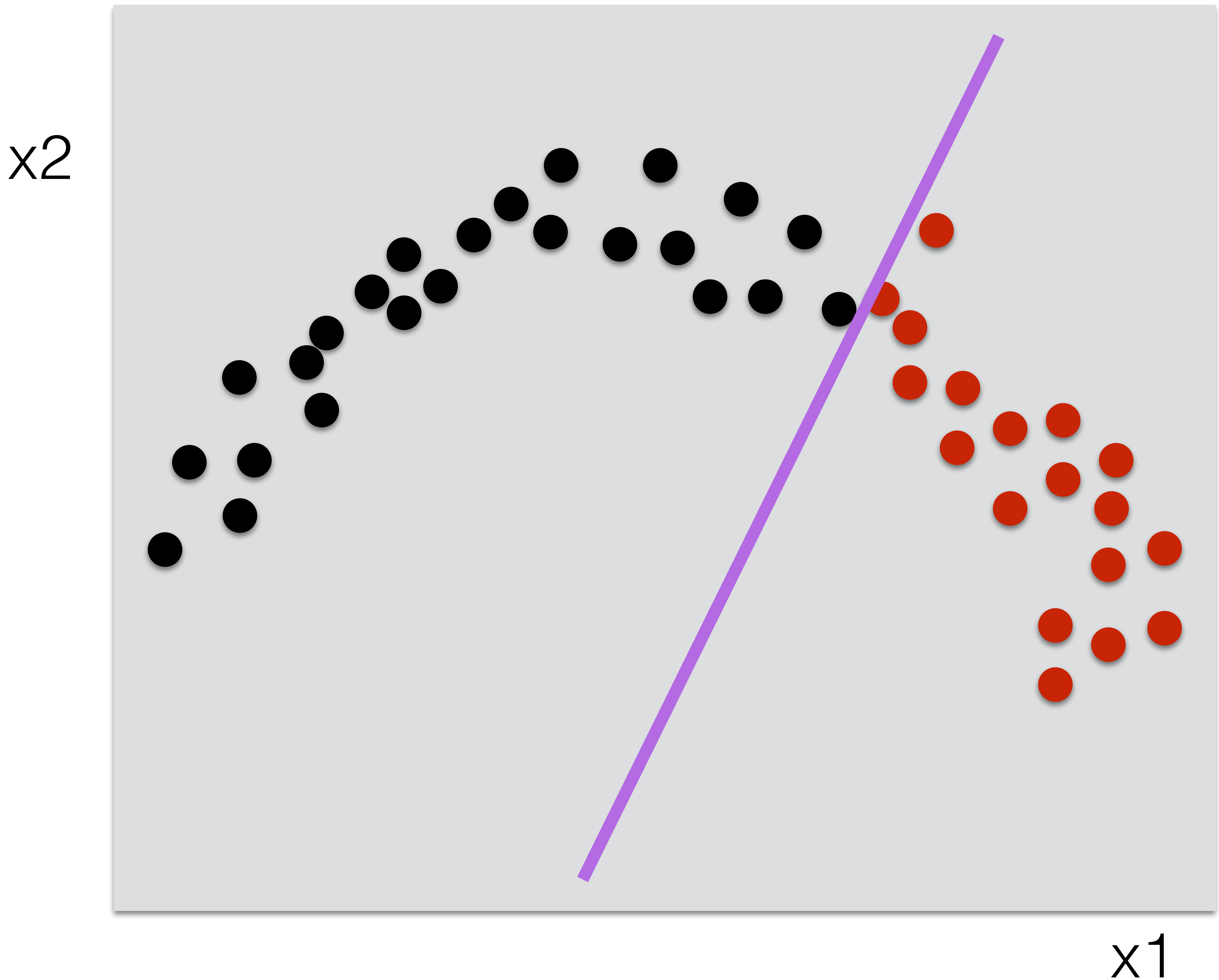
hypothetical data



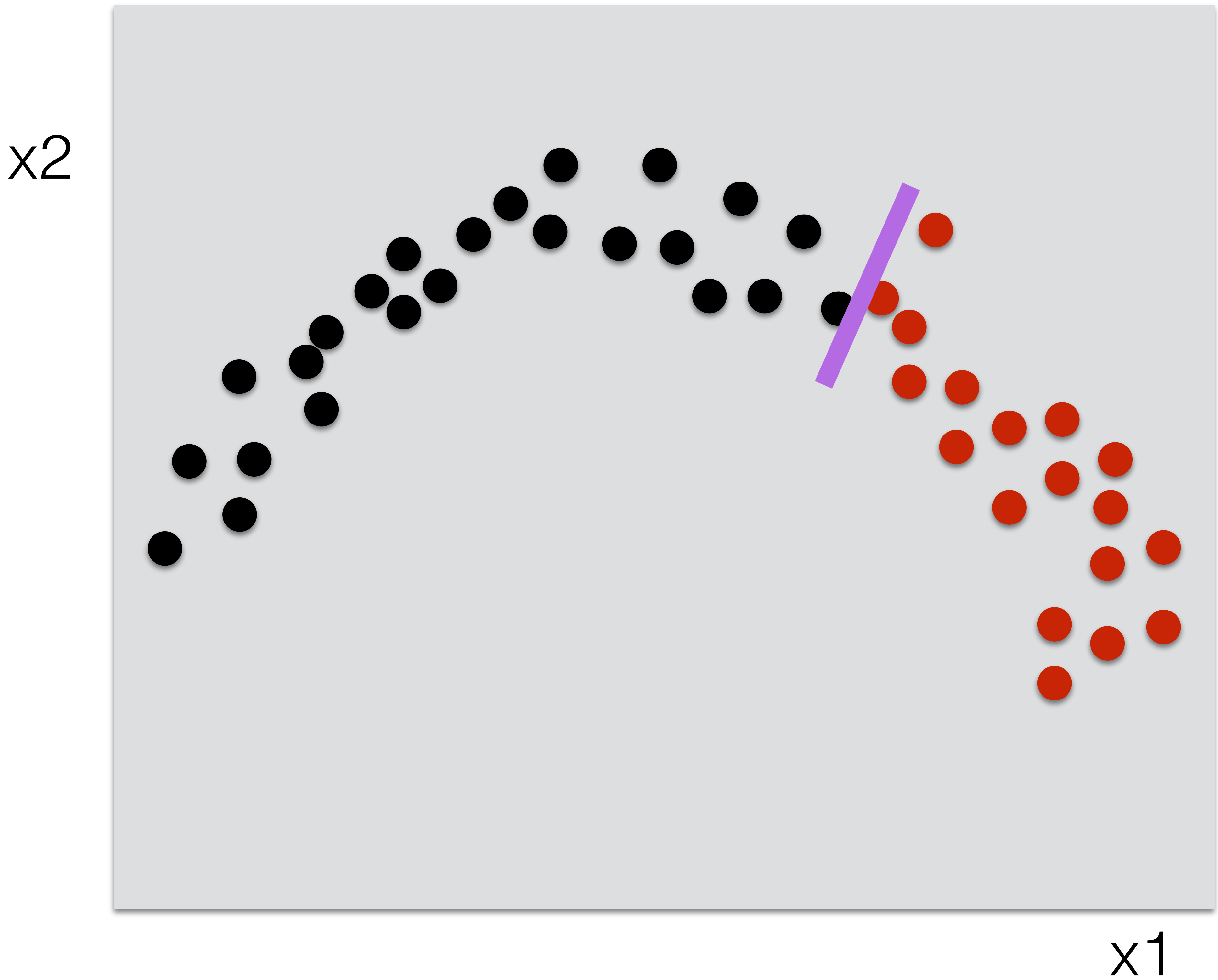
how to draw the decision boundaries?



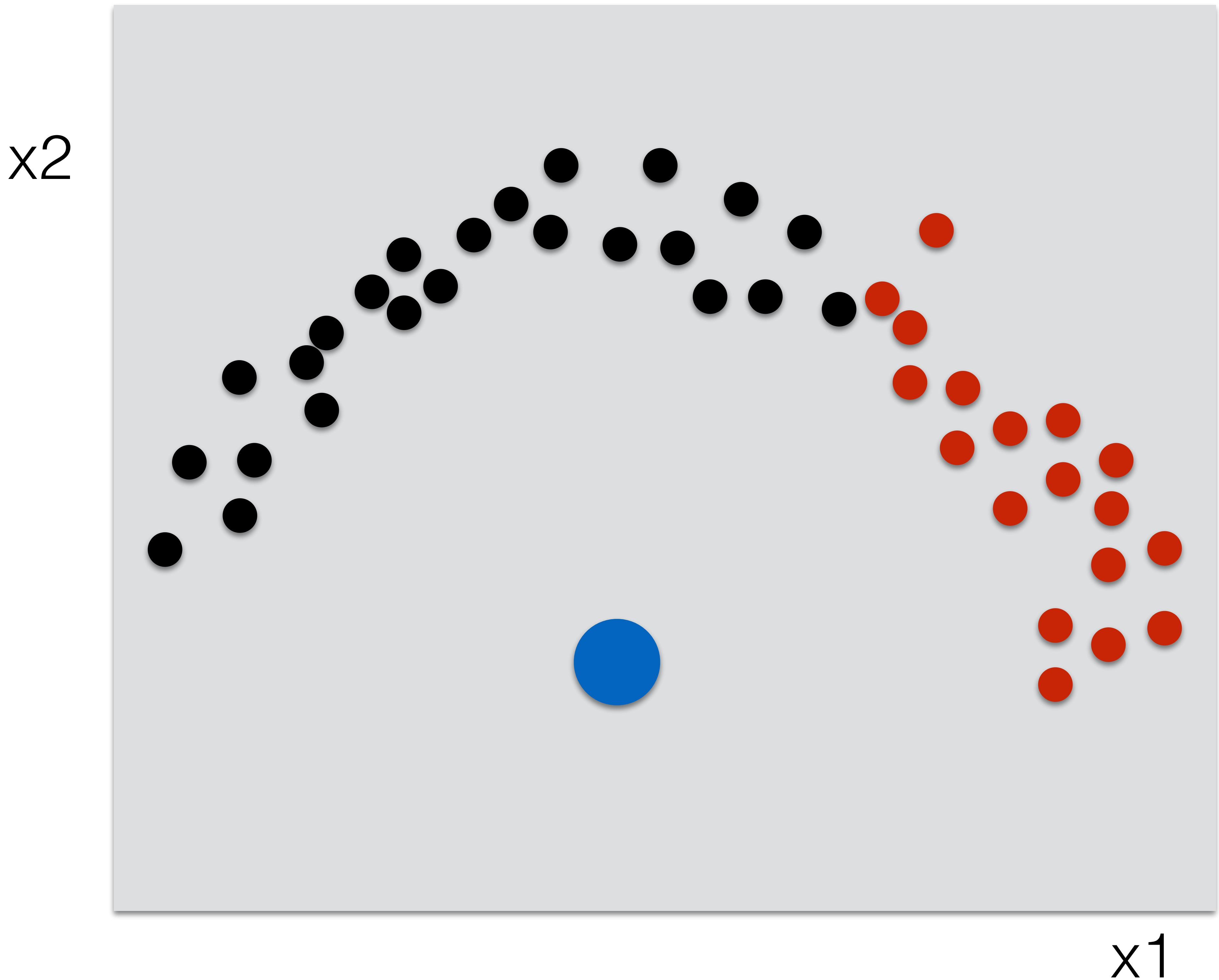
how to draw the decision boundaries?



is this a better boundary?

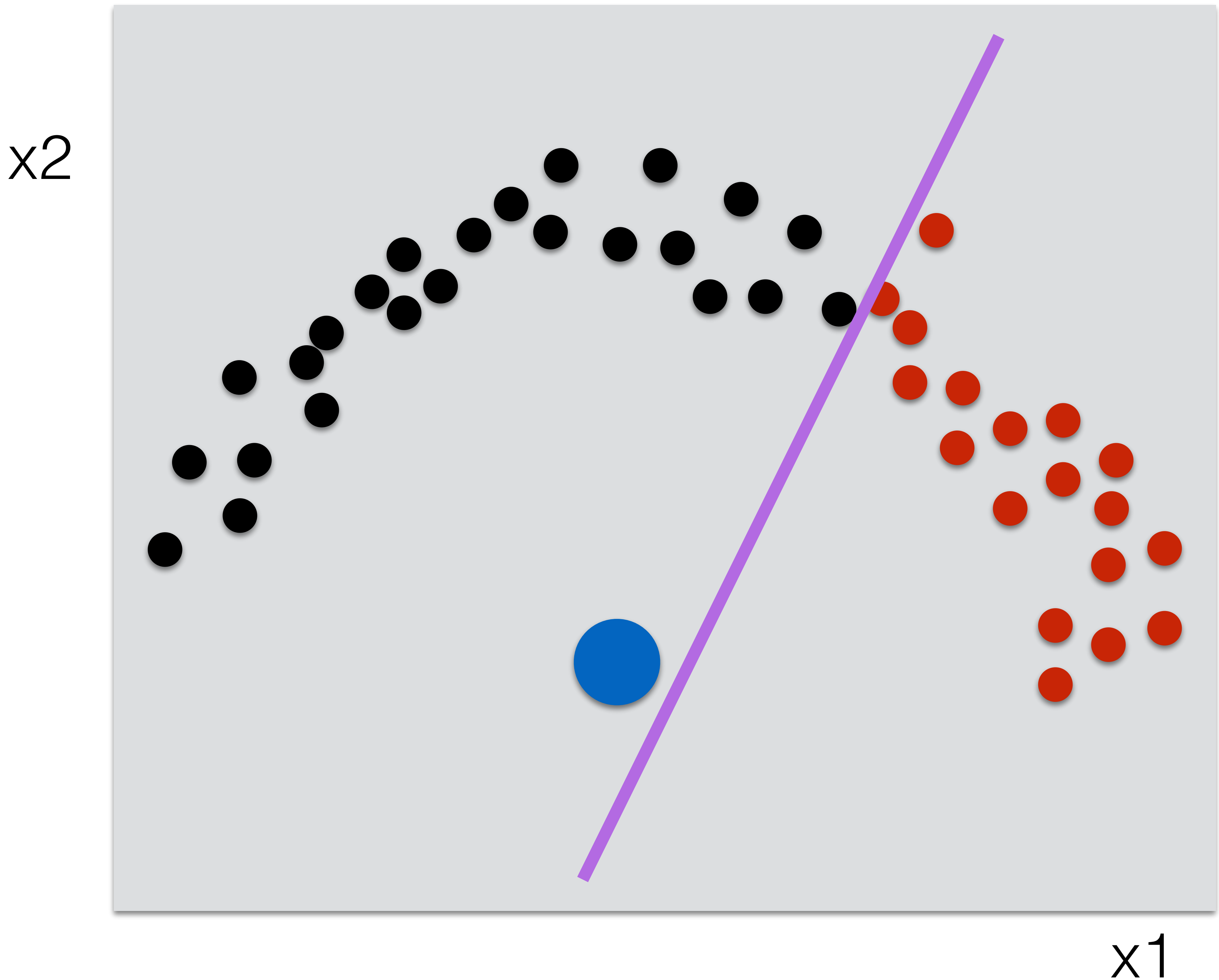


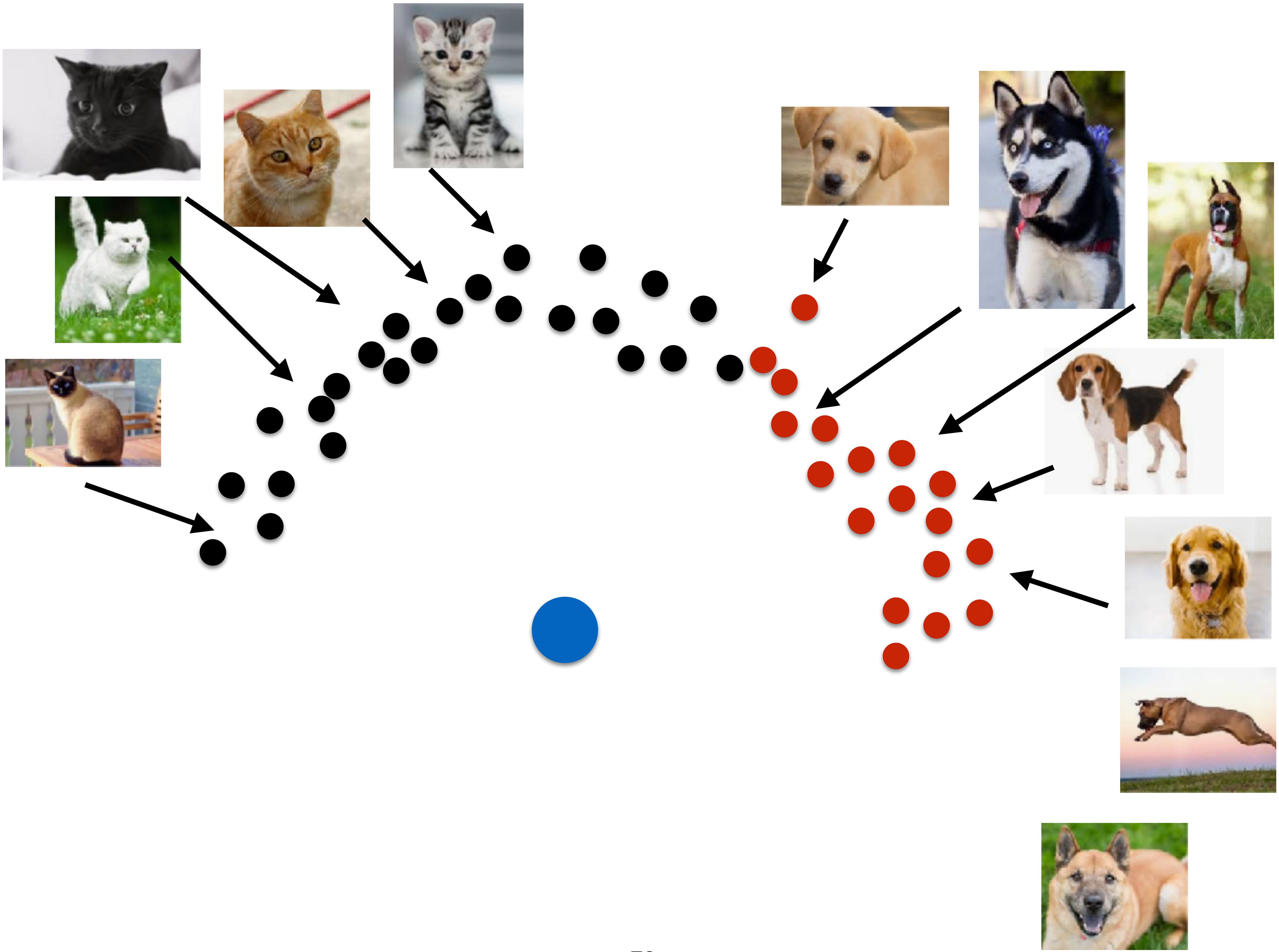
how do you classify the blue point?

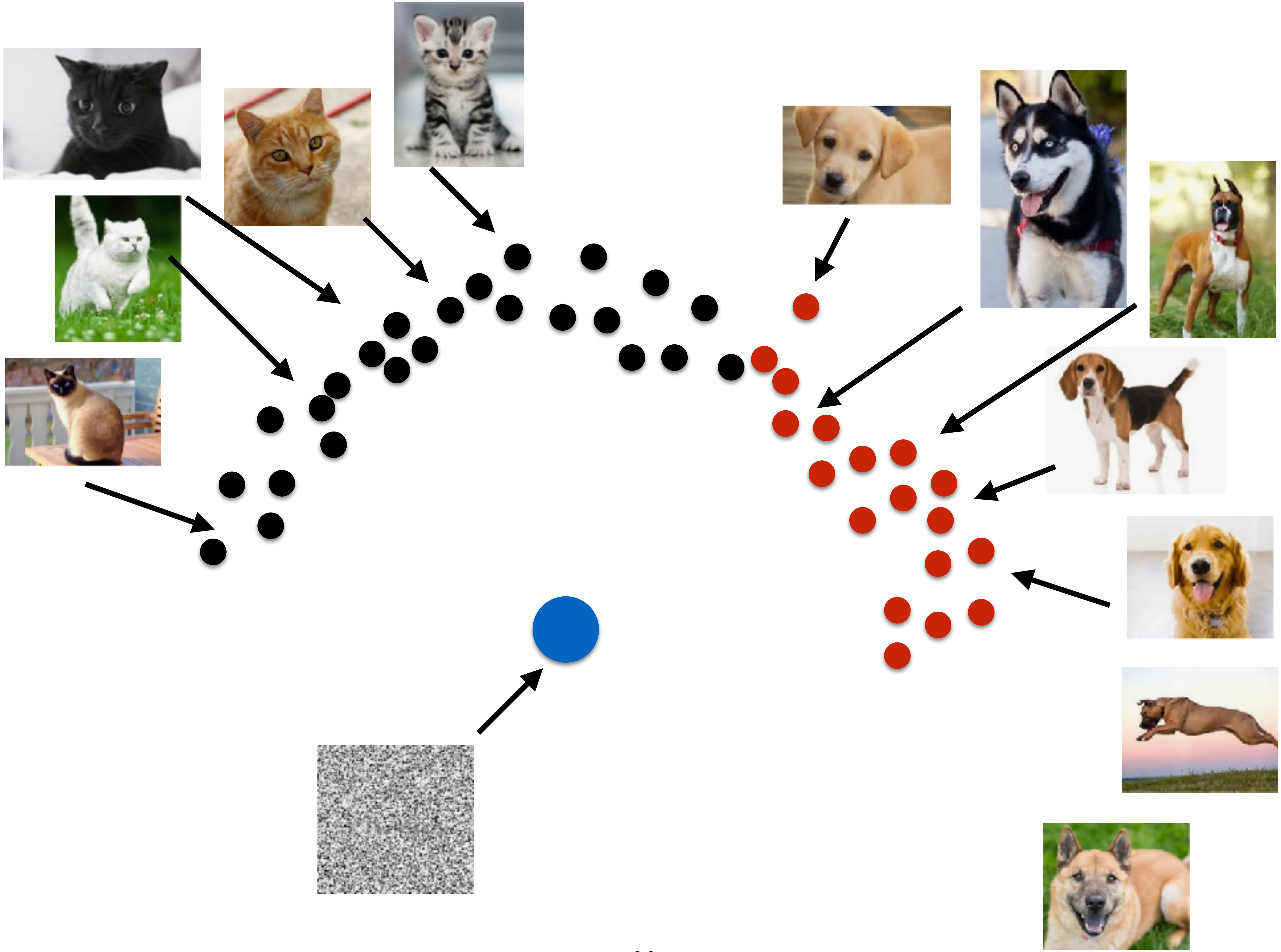


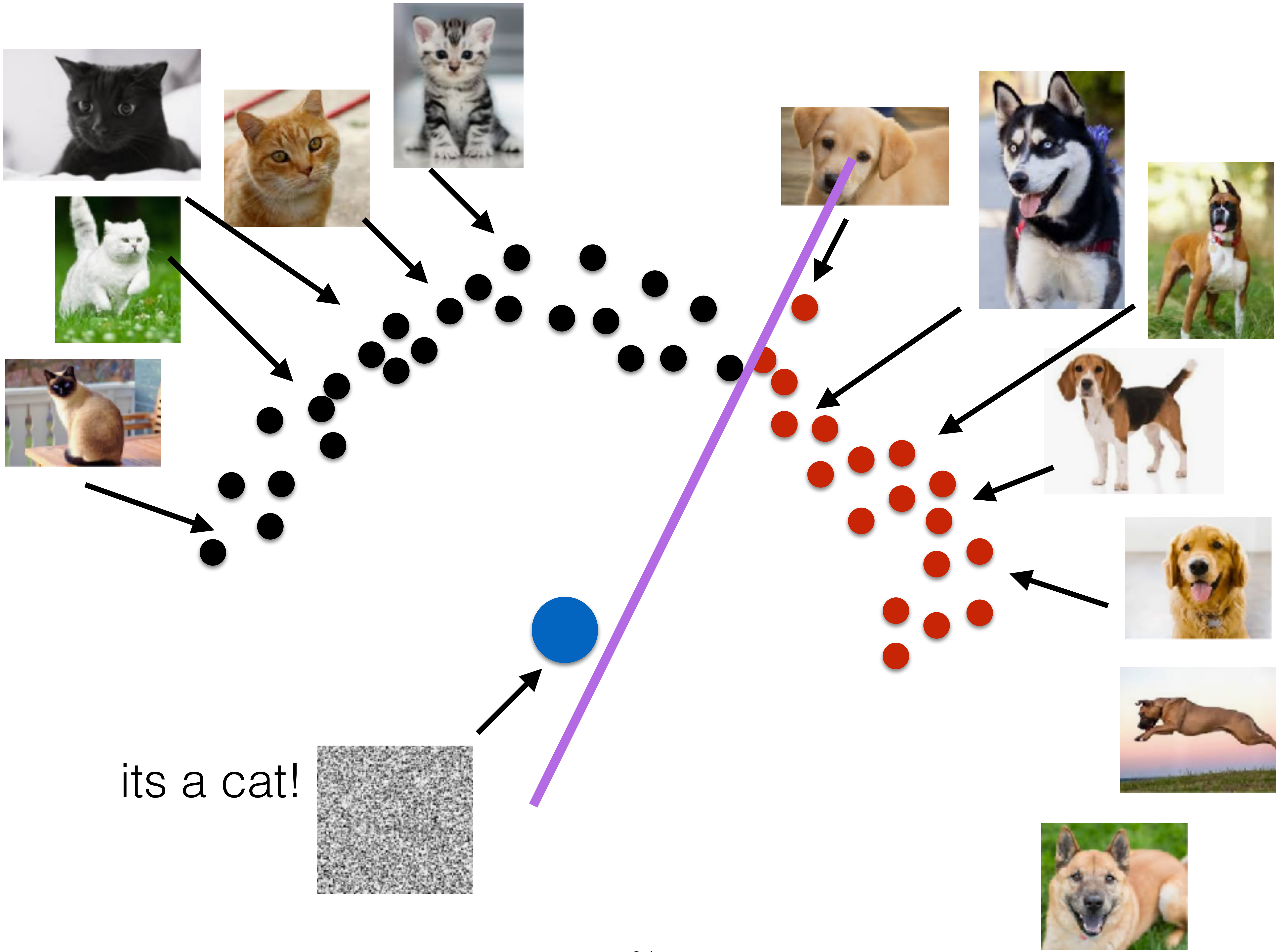
(a) Black
(b) Red
(c) unknown

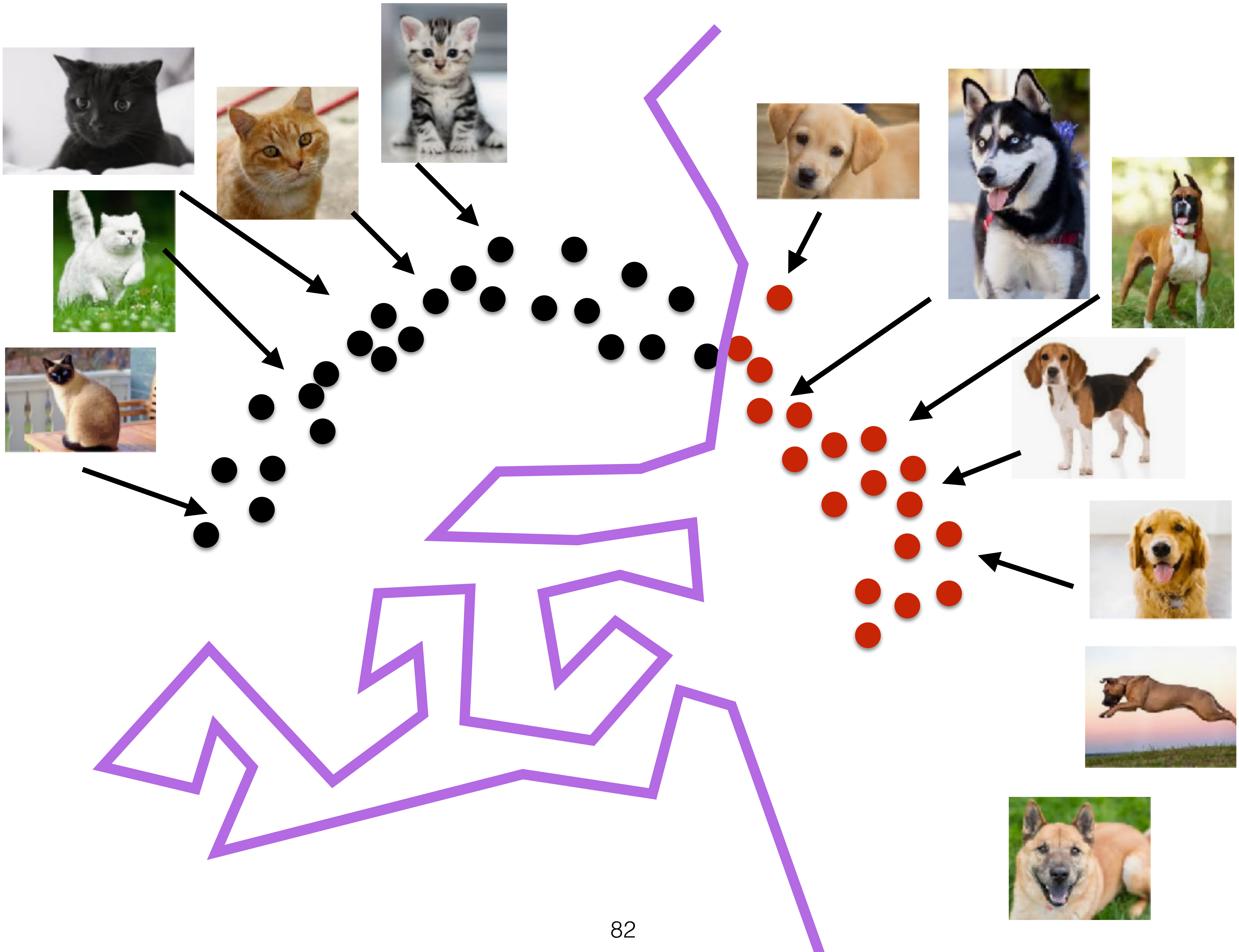
how to draw the decision boundaries?

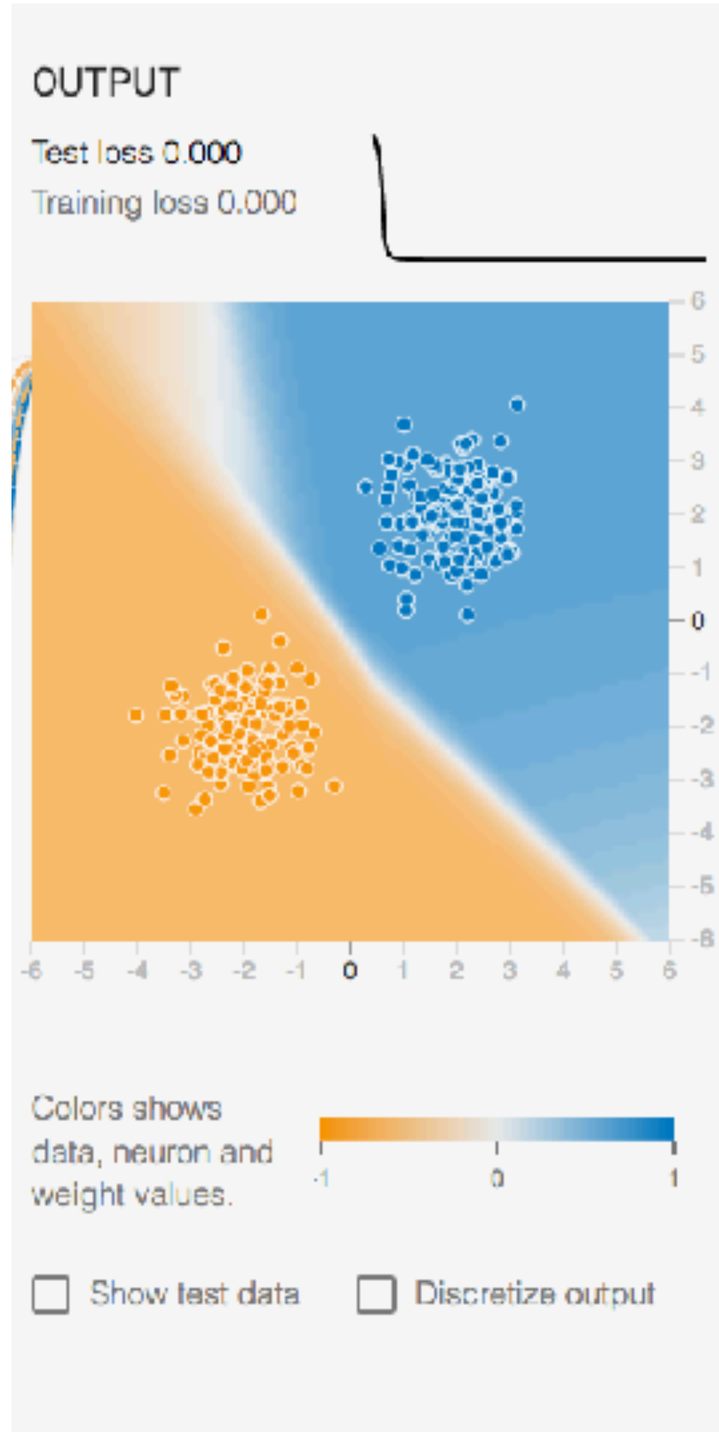


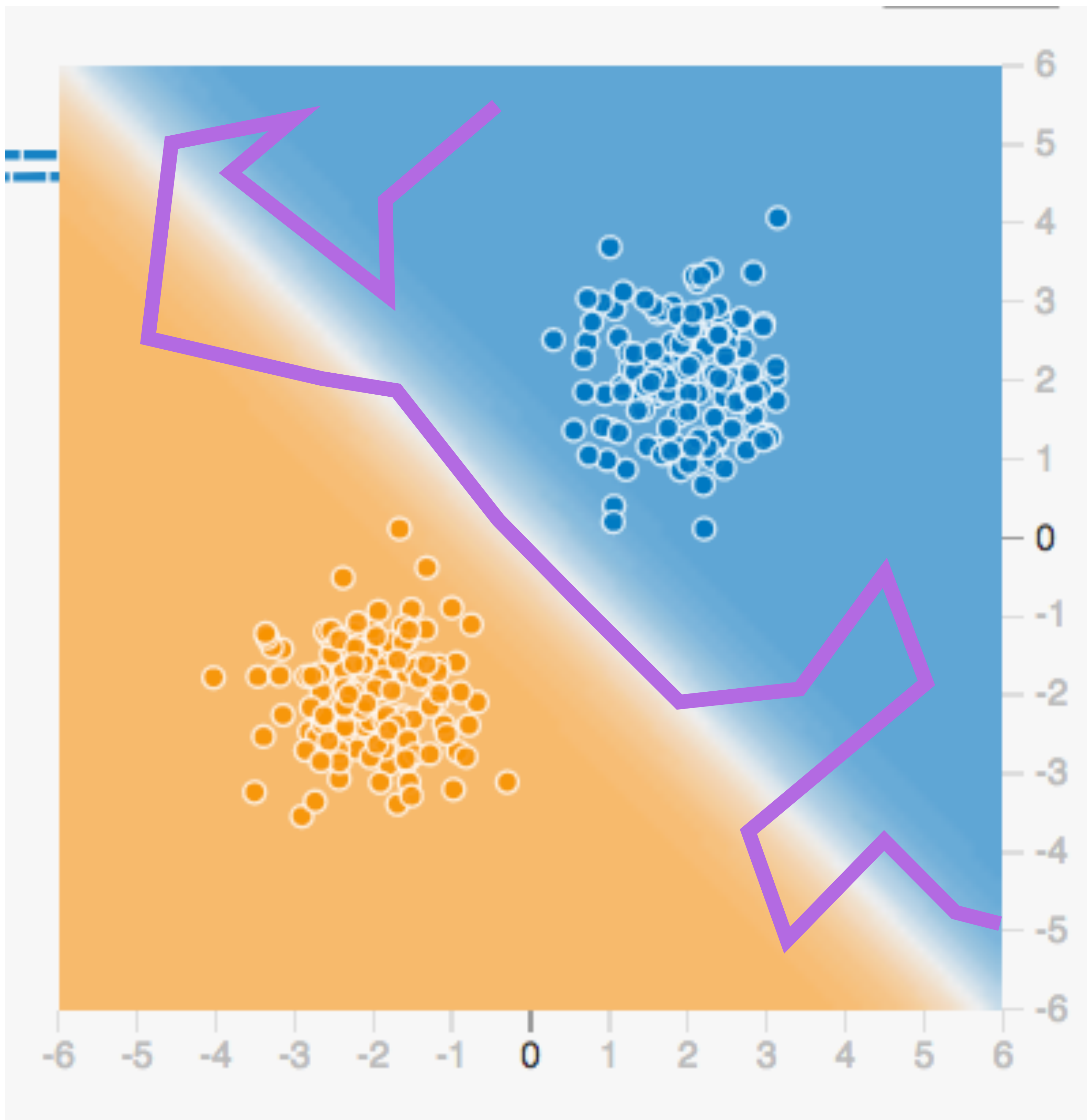





its a cat!




OUTPUT
Test loss 0.000
Training loss 0.000
6 5 4 3 2 1 0 -1 -2 -3 -4 -5 -6
-6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6
Colors shows data, neuron and weight values.
-1 0 1
Show test data
Discretize output

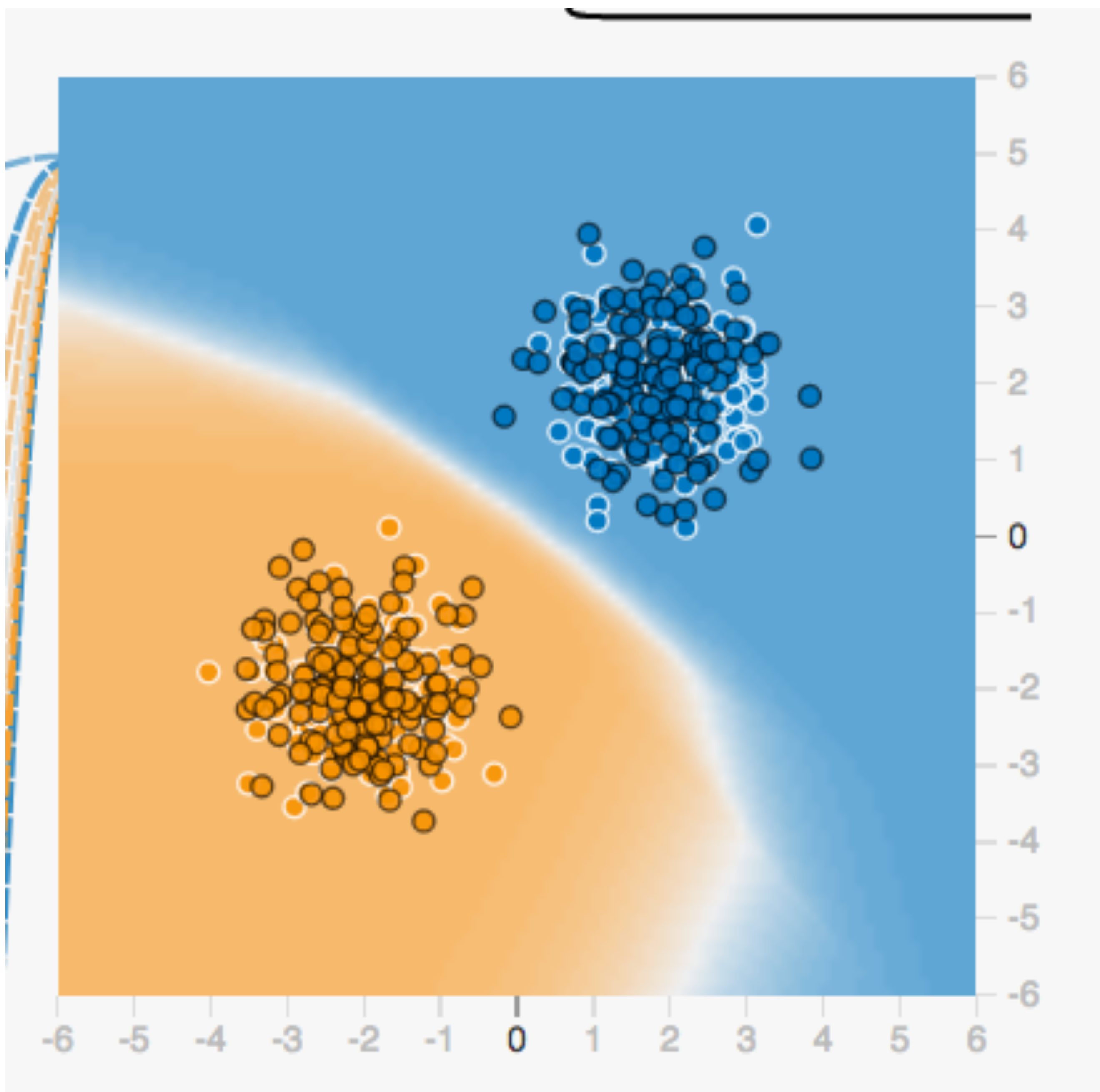


6
5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6

Closely related problem

how to design the architecture?

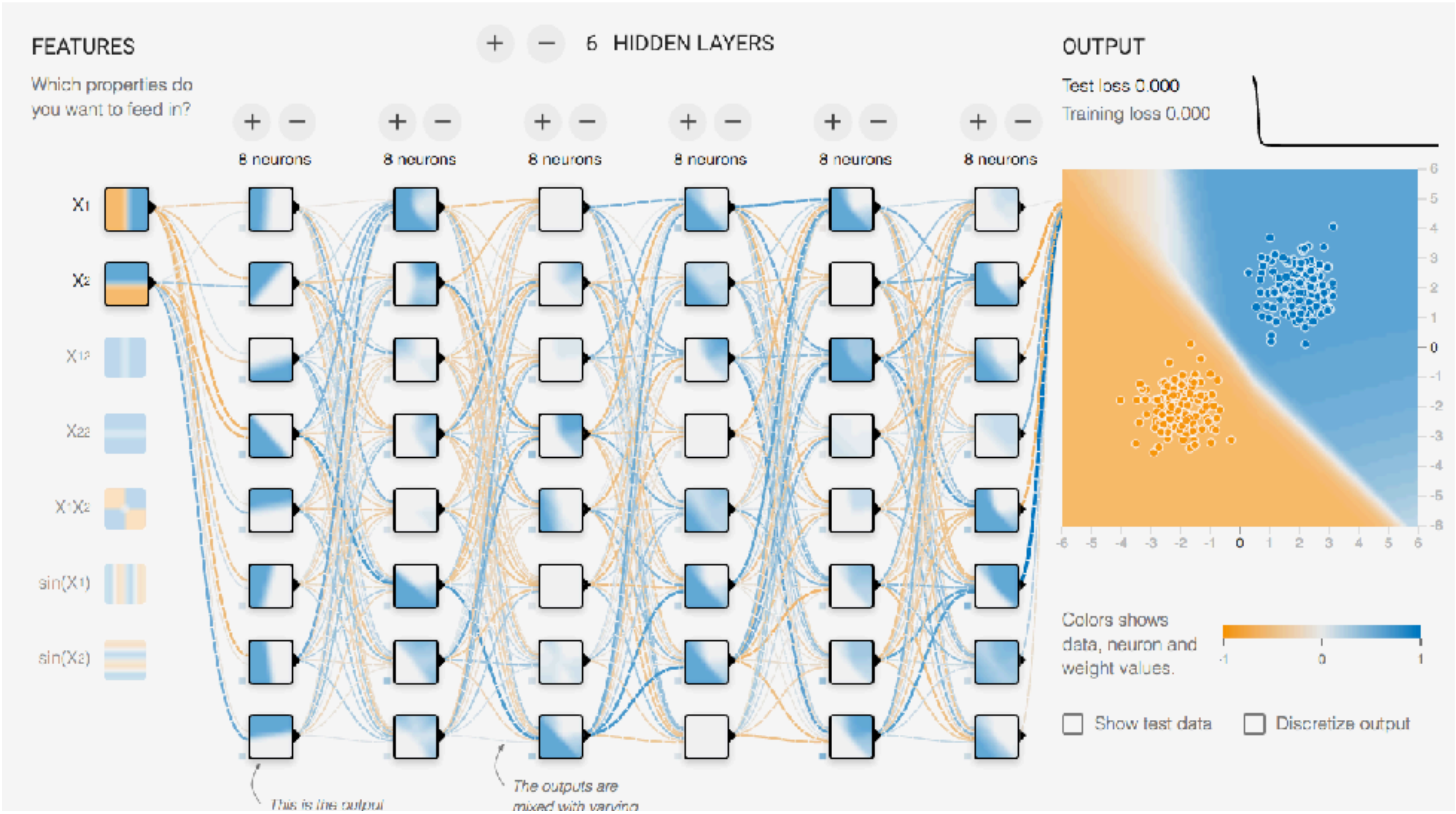
more layers always better?



6
5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6

try this network, run with

(1) relu
(2) tanh
(3) sigmoid
(4) linear

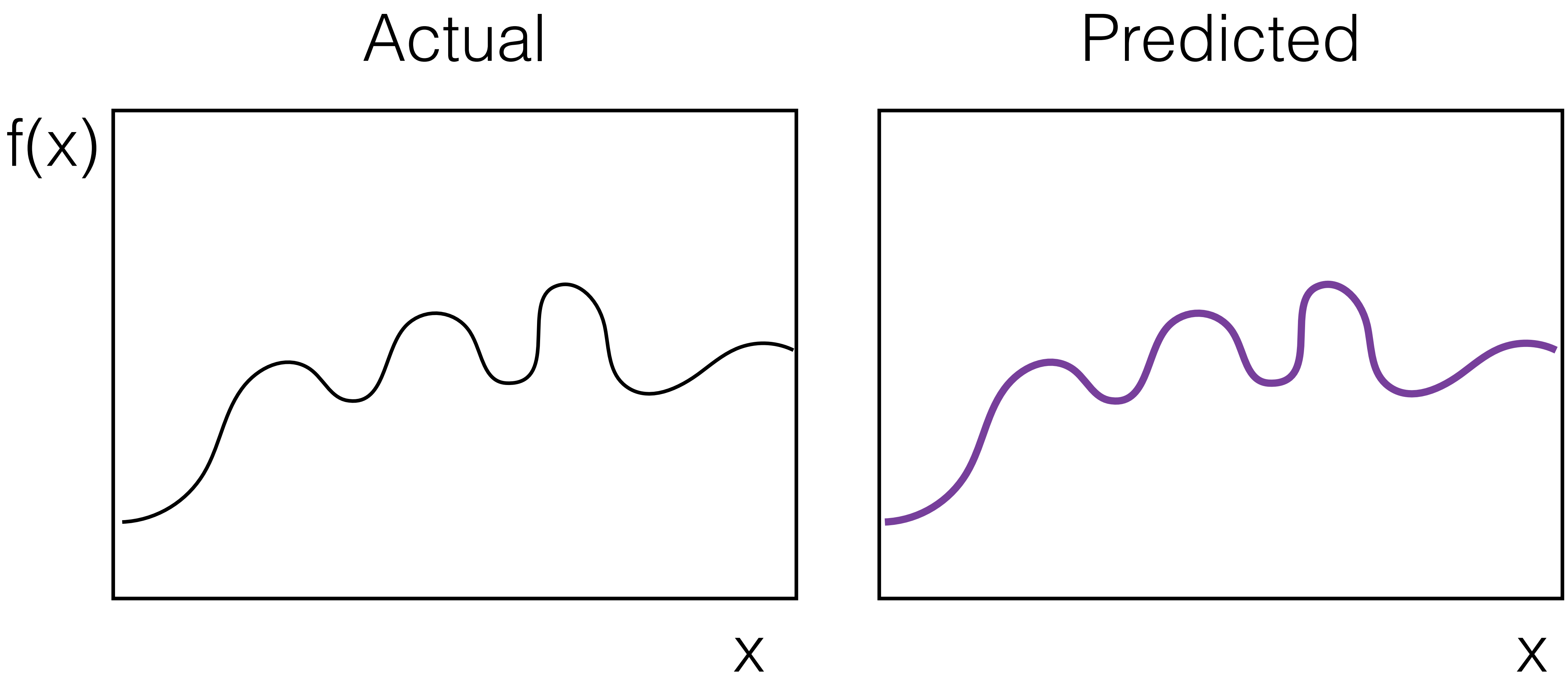


Neural network are universal approximations (under some conditions)

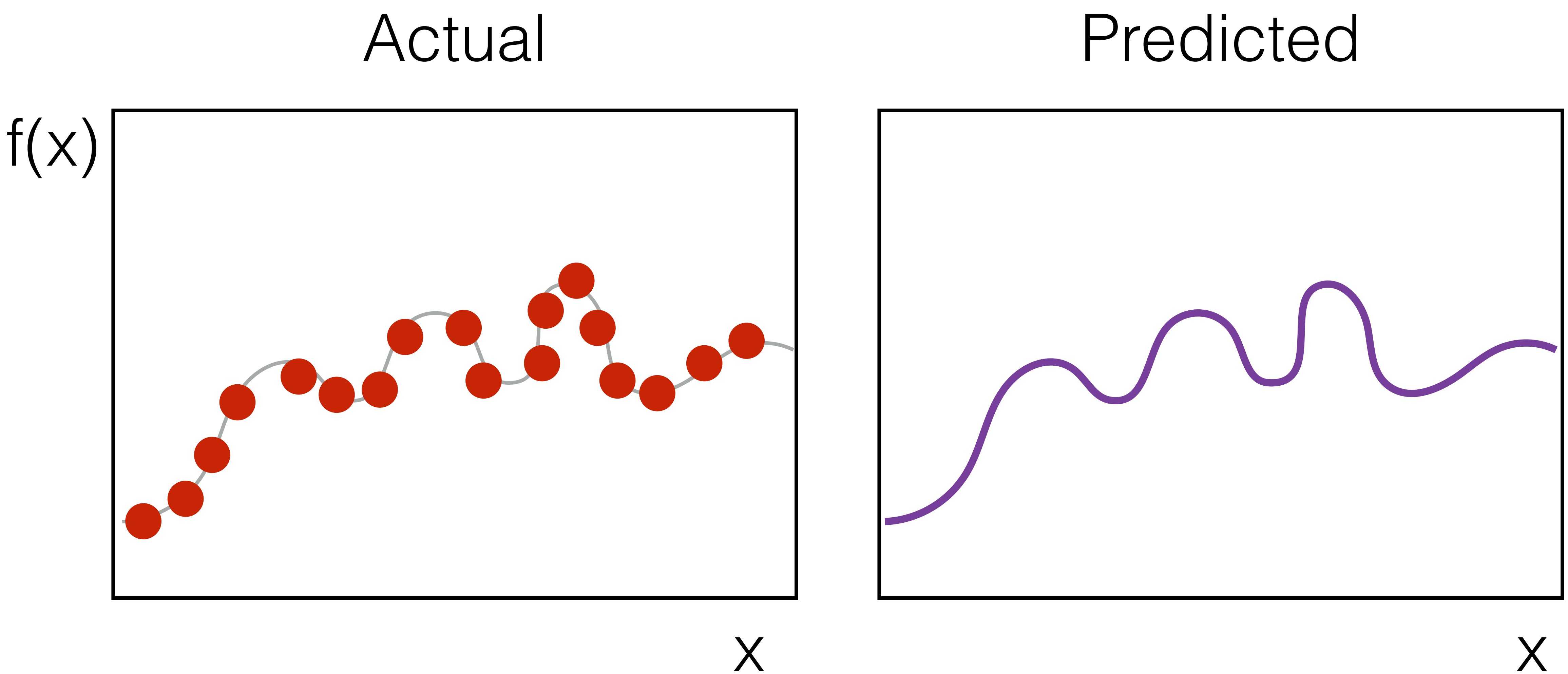
A good reference, read and ask
if you don't understand anything

http://neuralnetworksanddeeplearning.com/chap4.html

Is there a contradiction between
universal approximation theorem
and
problems with neural network?

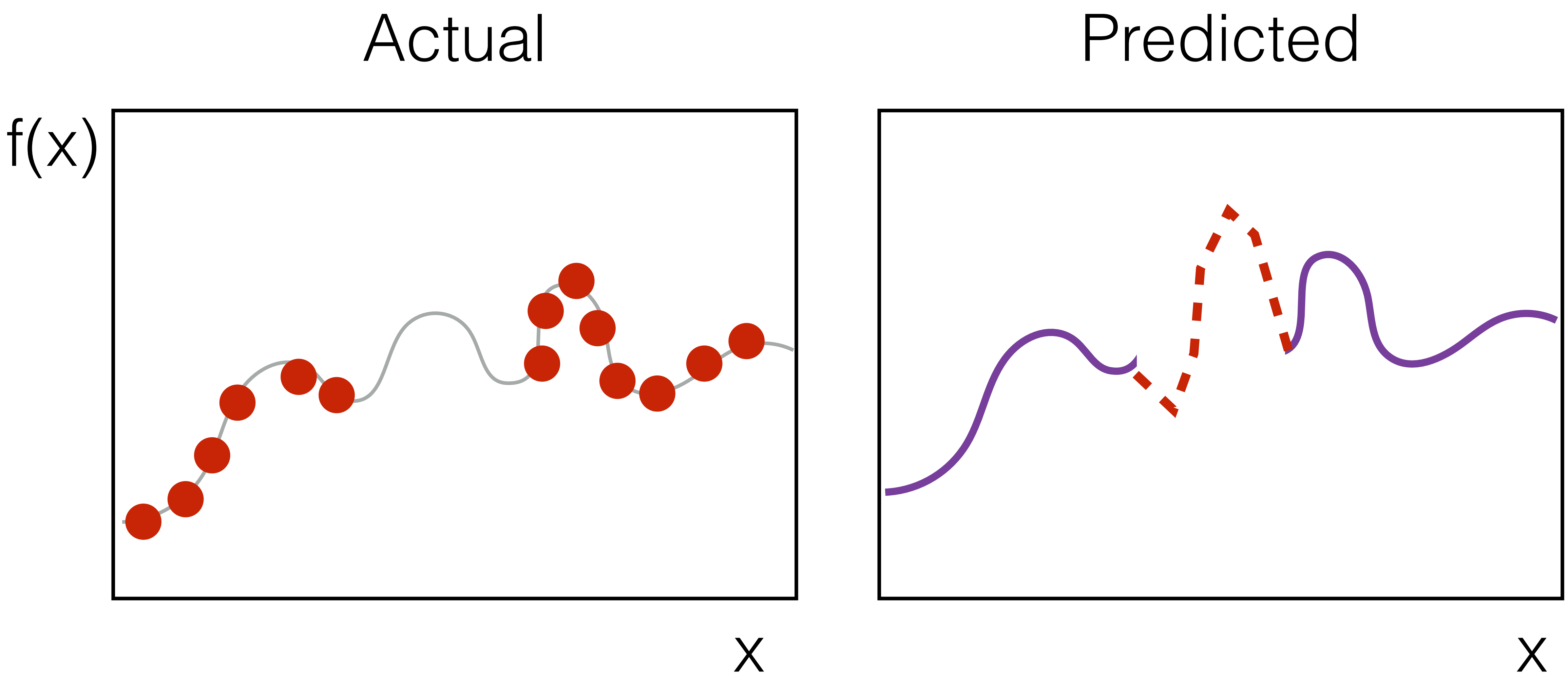


This is ok, the problem is the data

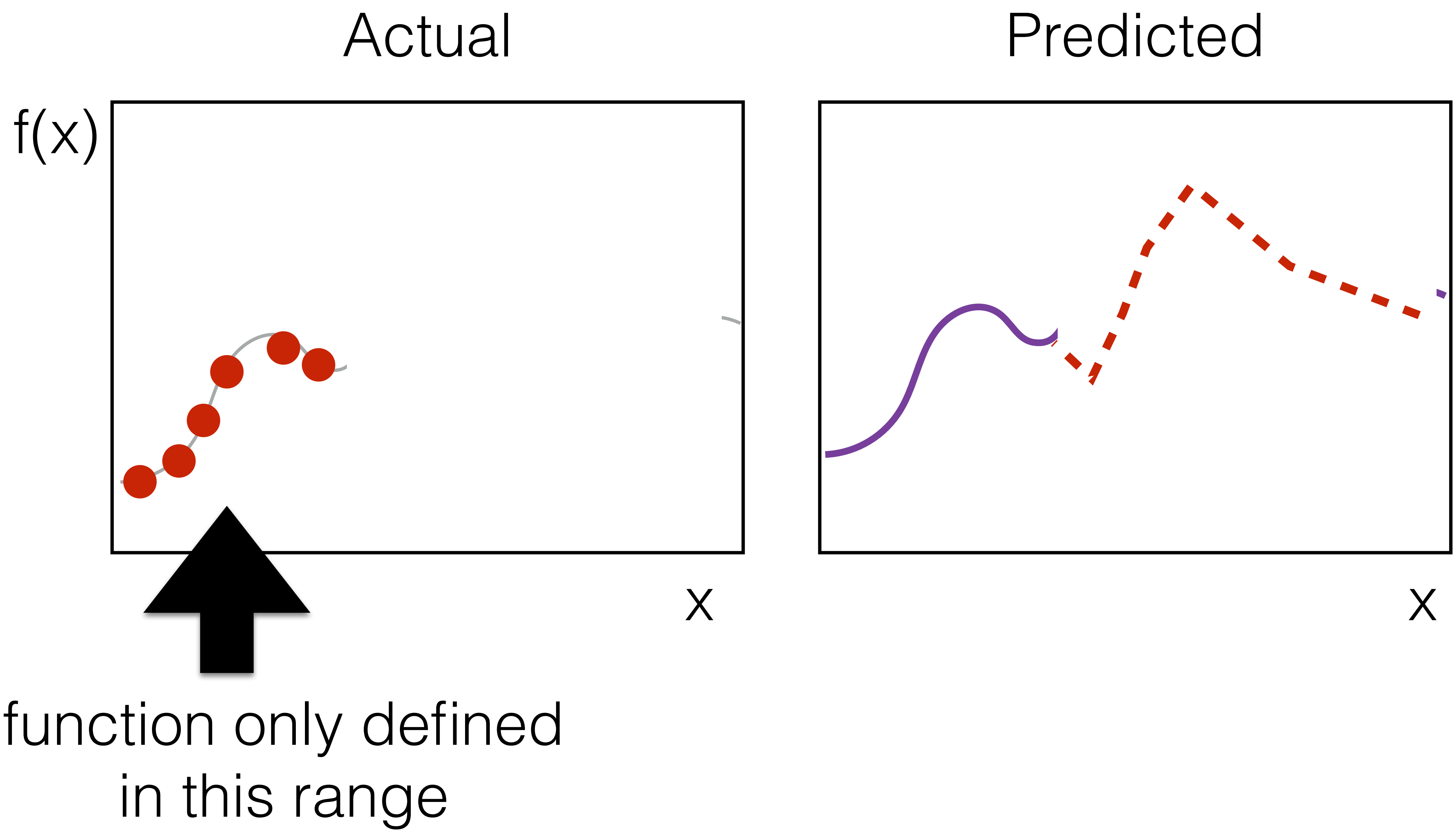


This is ok, the problem is the data

Problem #1: insufficient data



Problem #2: you define the data space wrongly



Auto-encoders

Concept of latent spaces,

data representation spaces,

data manifolds

What is the geometric interpretation of vector dot products?

Draw on the board, these vector dot products

x1	x2	y1	y2	x.y
0.01	1.51	0.11	0.99	
1.83	1.41	0.96	0.28	
0.70	0.93	0.94	-0.34	
0.81	1.17	0.0	1.00	
1.12	0.04	1.00	1.7	
1.71	1.41	0.95	0.30	
0.62	1.29	-0.5	0.86	
0.56	1.60	0.80	0.60	
0.26	0.86	-1	0	
1.94	1.94	0.5	1	

What is the geometric interpretation of matrix vector products?

0.3	-1.2
2.6	0.7
-0.5	-0.2

0.6
-1.2

0.7	0.4
-0.6	1.3
1.5	-1.4

0.6
-1.2

-0.2	2.1
-1.1	0.1
0.3	0.5

0.6
-1.2

What is the geometric interpretation of matrix vector products?

$$\mathrm{ReLU}\left(\begin{bmatrix}0.3 & -1.2\\ 2.6 & 0.7\\ -0.5 & -0.2\end{bmatrix}\begin{bmatrix}0.6\\ -1.2\end{bmatrix}\right)$$

$$\mathrm{ReLU}\left(\begin{bmatrix}-0.2 & 2.1\\ -1.1 & 0.1\\ 0.3 & 0.5\end{bmatrix}\begin{bmatrix}0.6\\ -1.2\end{bmatrix}\right)$$

$$\mathrm{ReLU}\left(\begin{bmatrix}0.7 & 0.4\\ -0.6 & 1.3\\ 1.5 & -1.4\end{bmatrix}\begin{bmatrix}0.6\\ -1.2\end{bmatrix}\right)$$