Neural networks for time series analysis

Oleksandr Honchar | University of Verona
March 2018

Day 2: data and simple neural nets

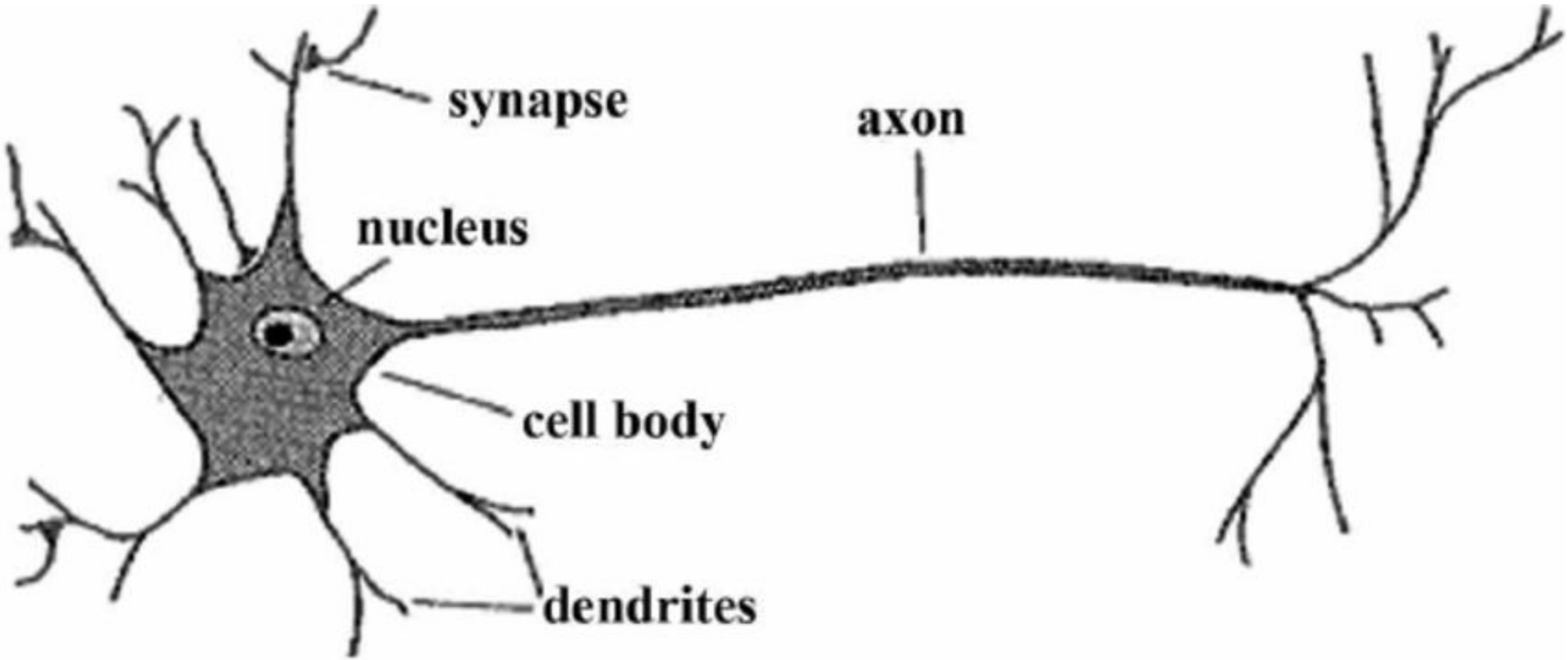
Introduction to machine learning and time series analysis

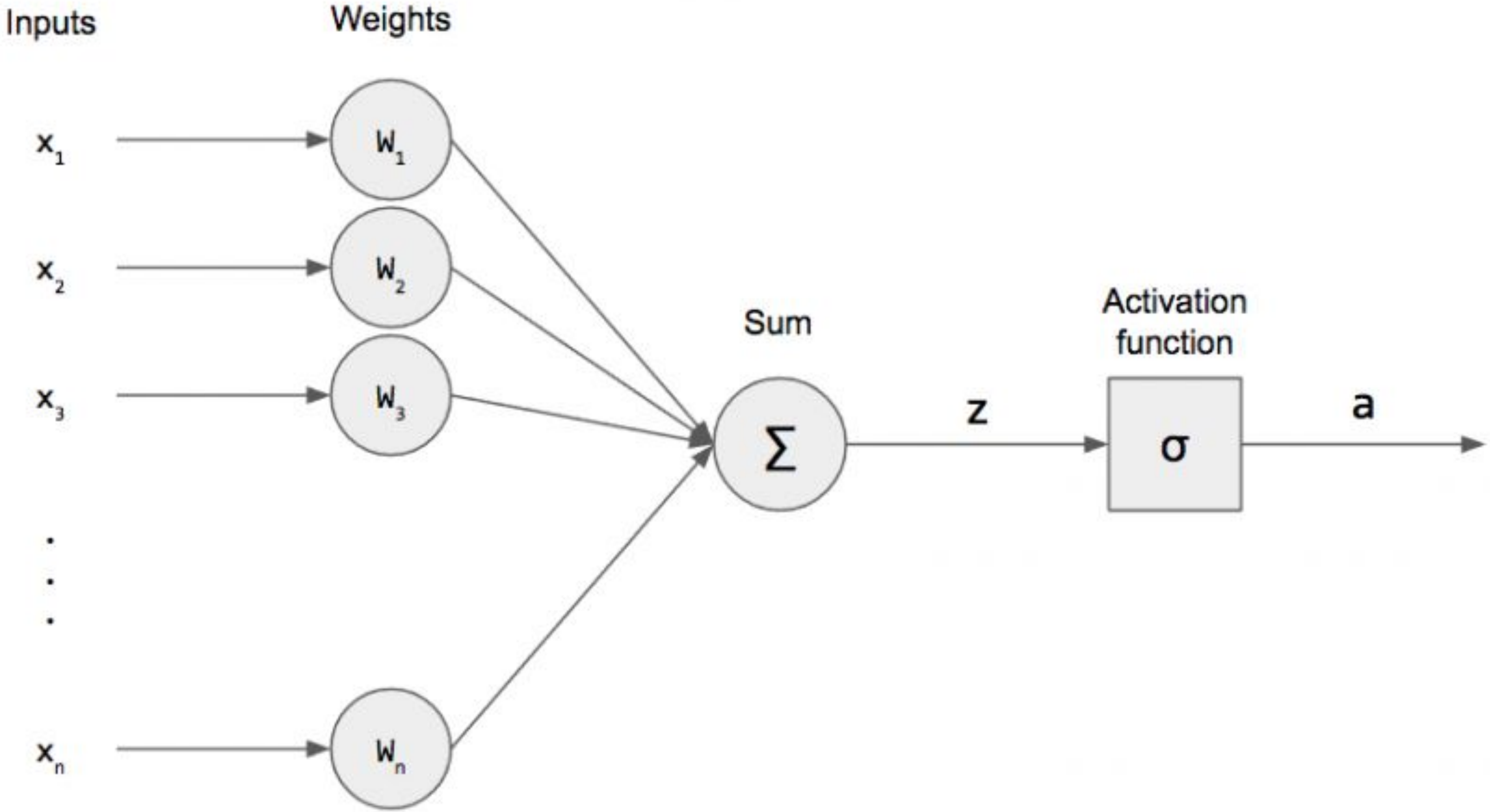
Data preparation and feedforward neural networks

Convolutional, recurrent neural networks and overfitting

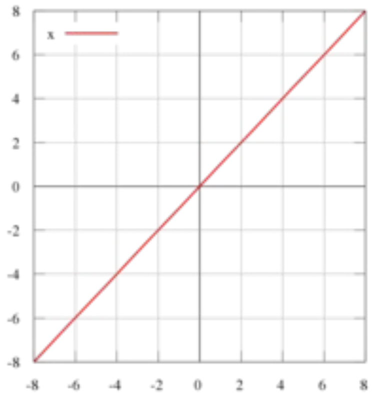
Building a trading strategy and further applications

neural networks

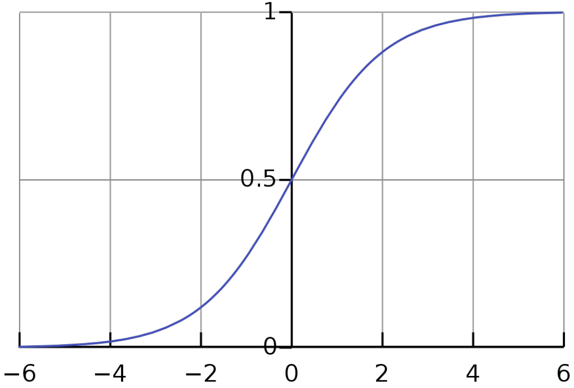

synapse
axon
nucleus
cell body
dendrites



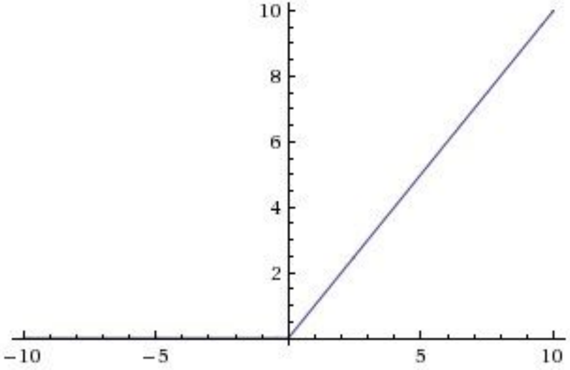
Inputs
Weights
x1
x2
x3
xn
W1
W2
W3
Wn
Sum
Σ
z
Activation
function
σ
a



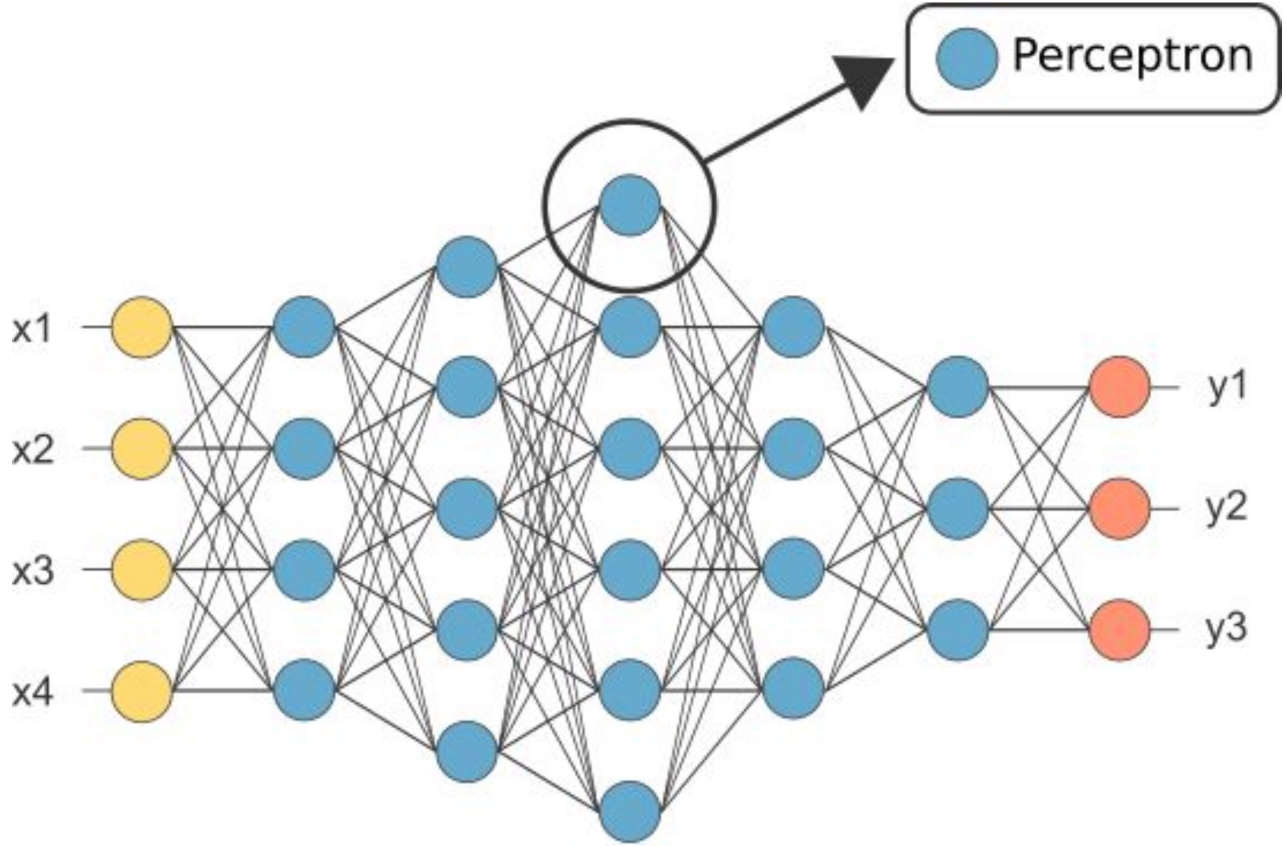
$$f(x) = x$$



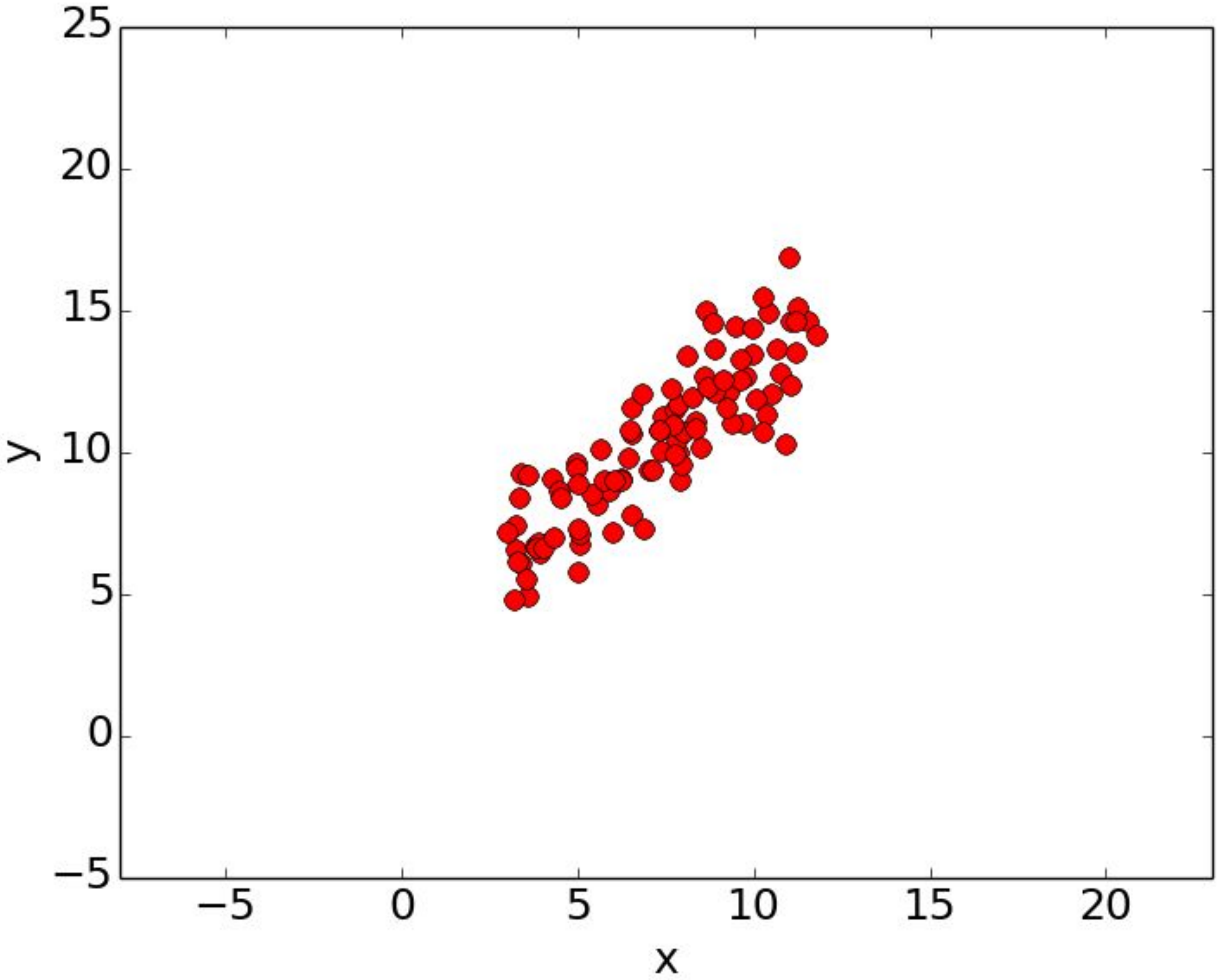
$$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$$



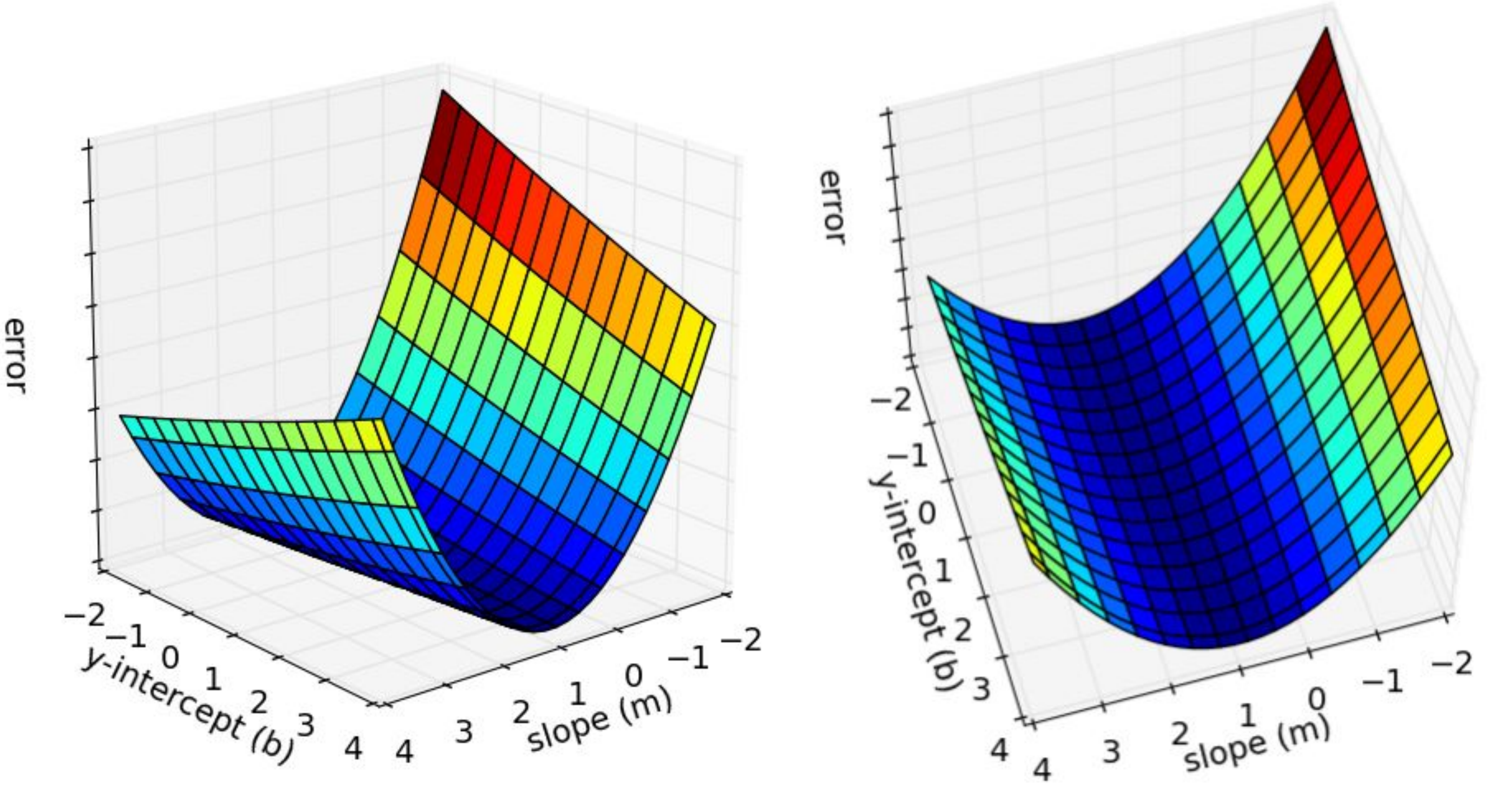
$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$


Perceptron
x1
x2
x3
x4
y1
y2
y3

learning


25
20
15
10
5
0
−5
y
−5
0
5
10
15
20
x

$$\text{Error}_{(\text{m,b})} = \frac{1}{N}\sum_{i=1}^{N}(y_i - (mx_i + b))^2$$



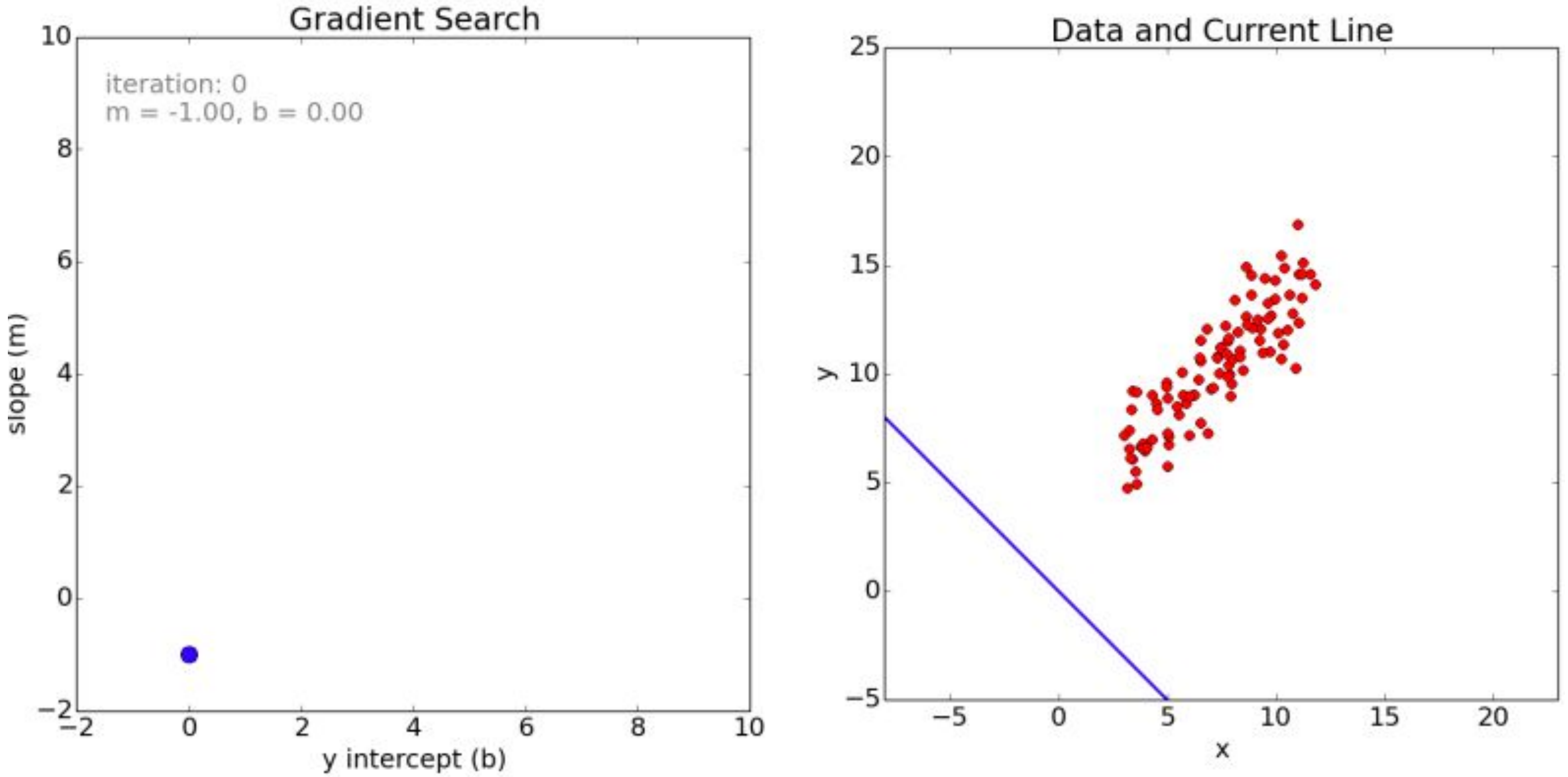
error
−2 −1 0 1 2 3 4
y-intercept (b)
4 3 2 1 0 −1 −2
slope (m)
error
−2 −1 0 1 2 3 4
y-intercept (b)
4 3 2 1 0 −1 −2
slope (m)

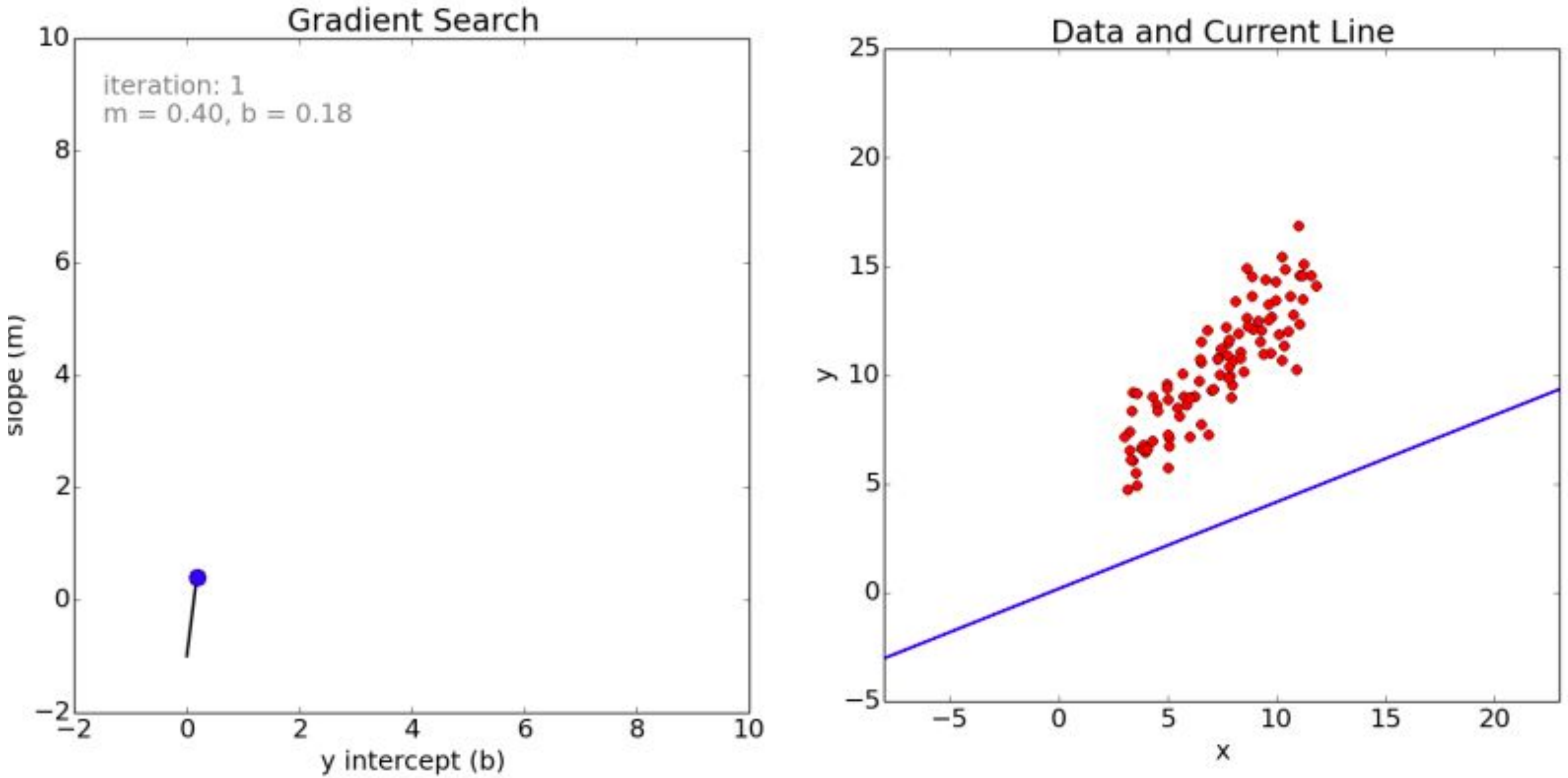
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta).$$

$$\frac{\partial}{\partial \mathrm{m}} = \frac{2}{N}\sum_{i=1}^{N} -x_i(y_i - (mx_i + b))$$

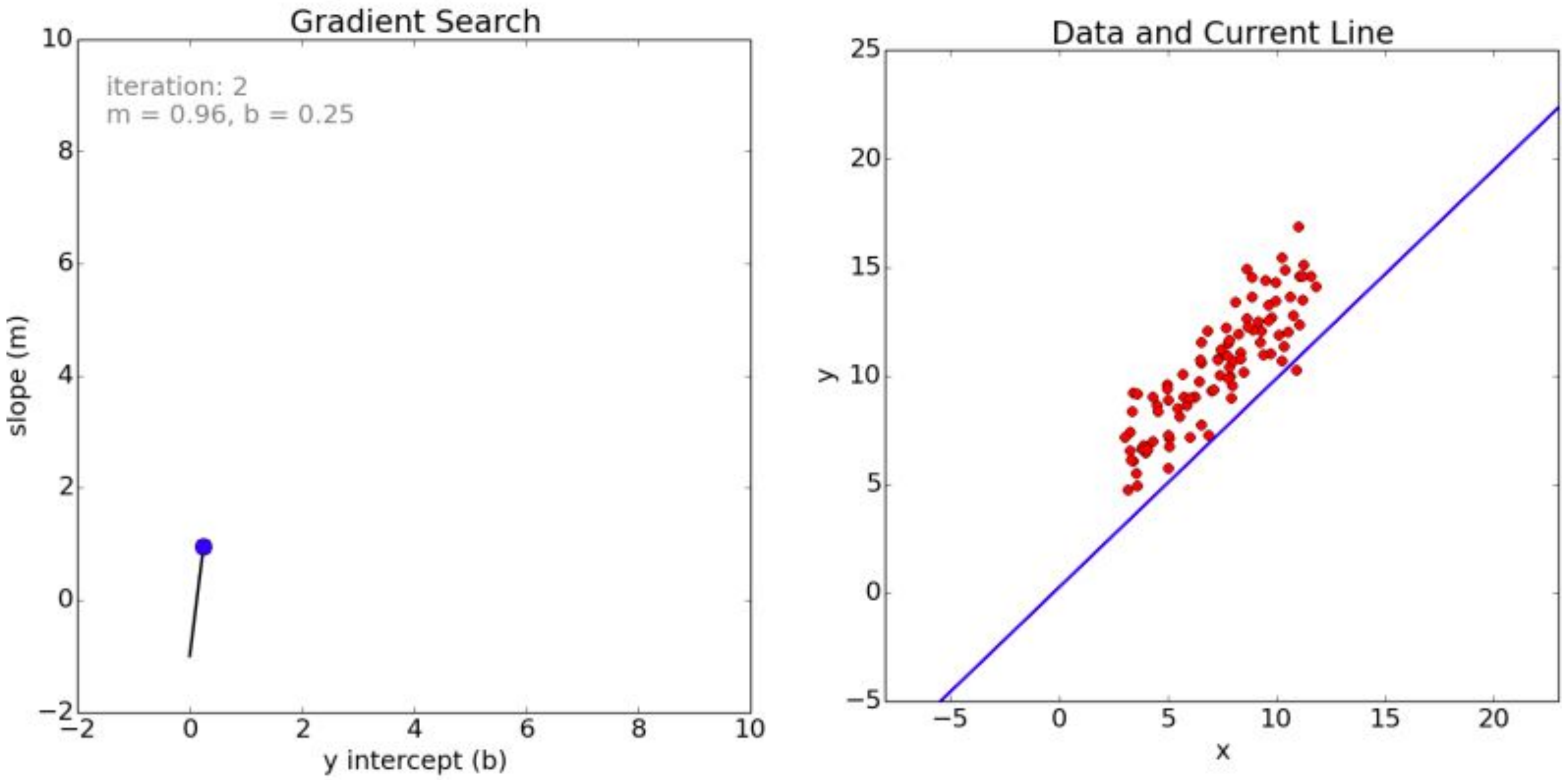
$$\frac{\partial}{\partial \mathrm{b}} = \frac{2}{N}\sum_{i=1}^{N} -(y_i - (mx_i + b))$$

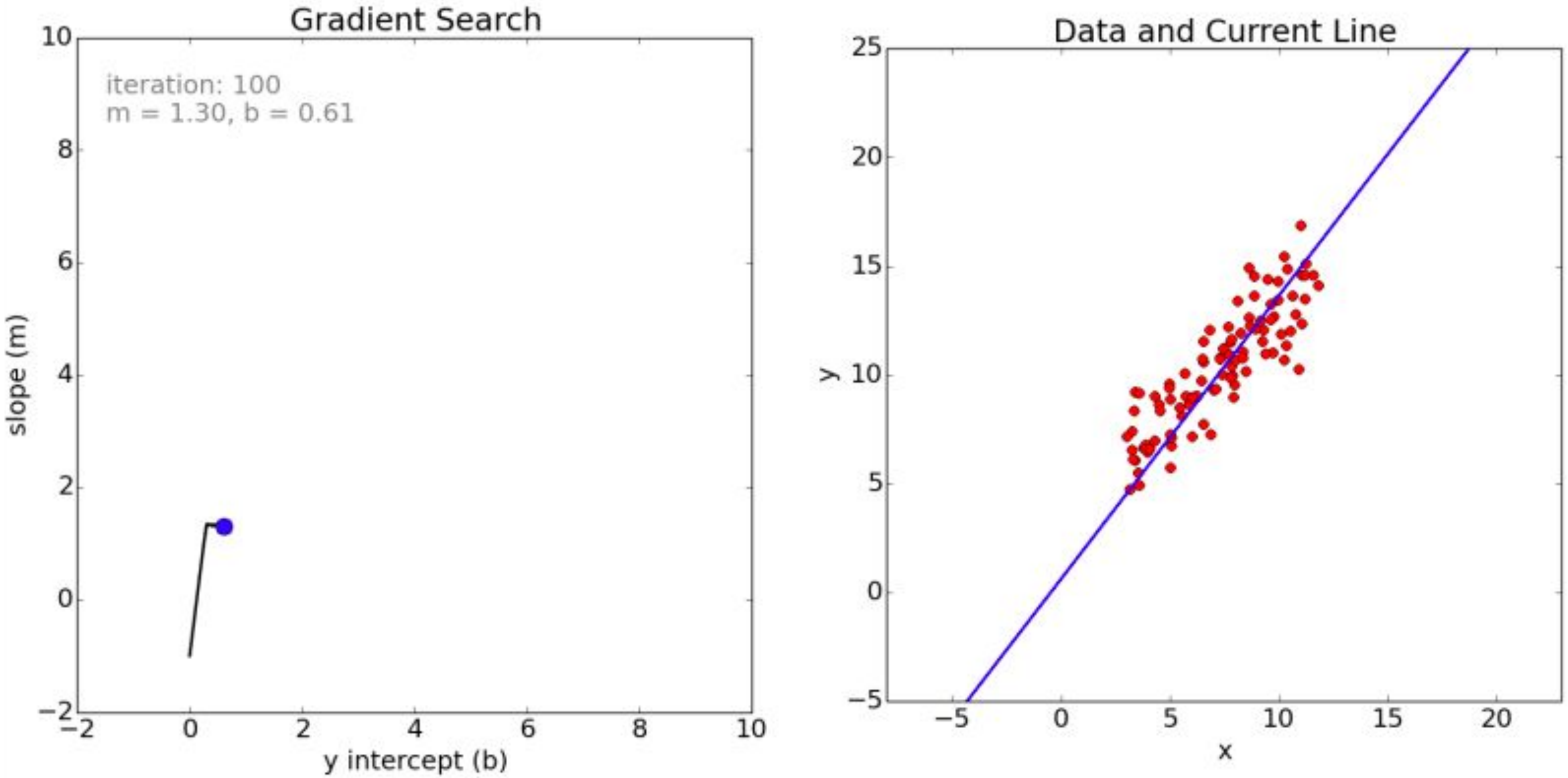
https://www.youtube.com/watch?v=kJgx2RcJKZY

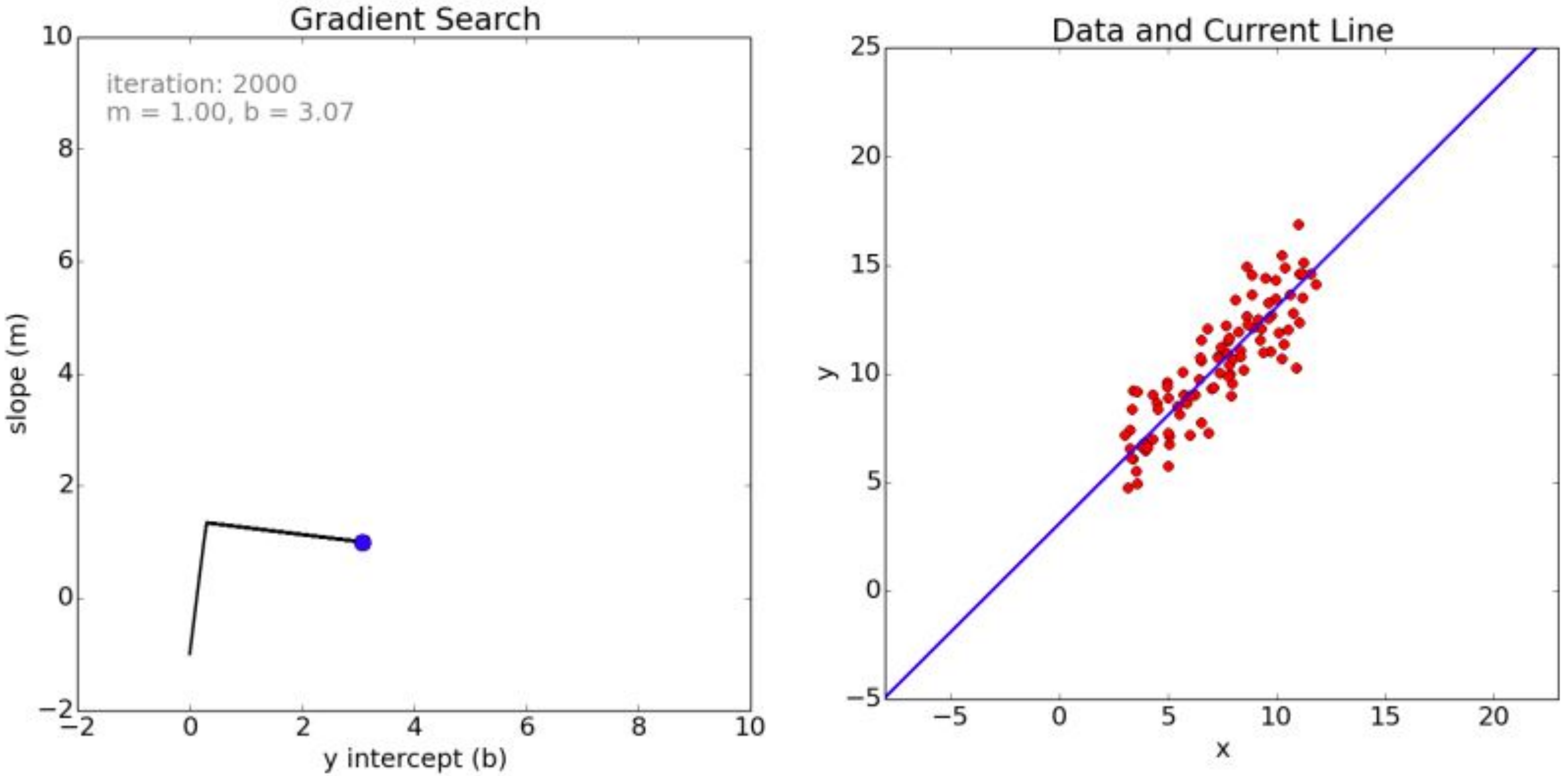

Gradient Search
iteration: 0
m = -1.00, b = 0.00
slope (m)
y intercept (b)
Data and Current Line
y
x

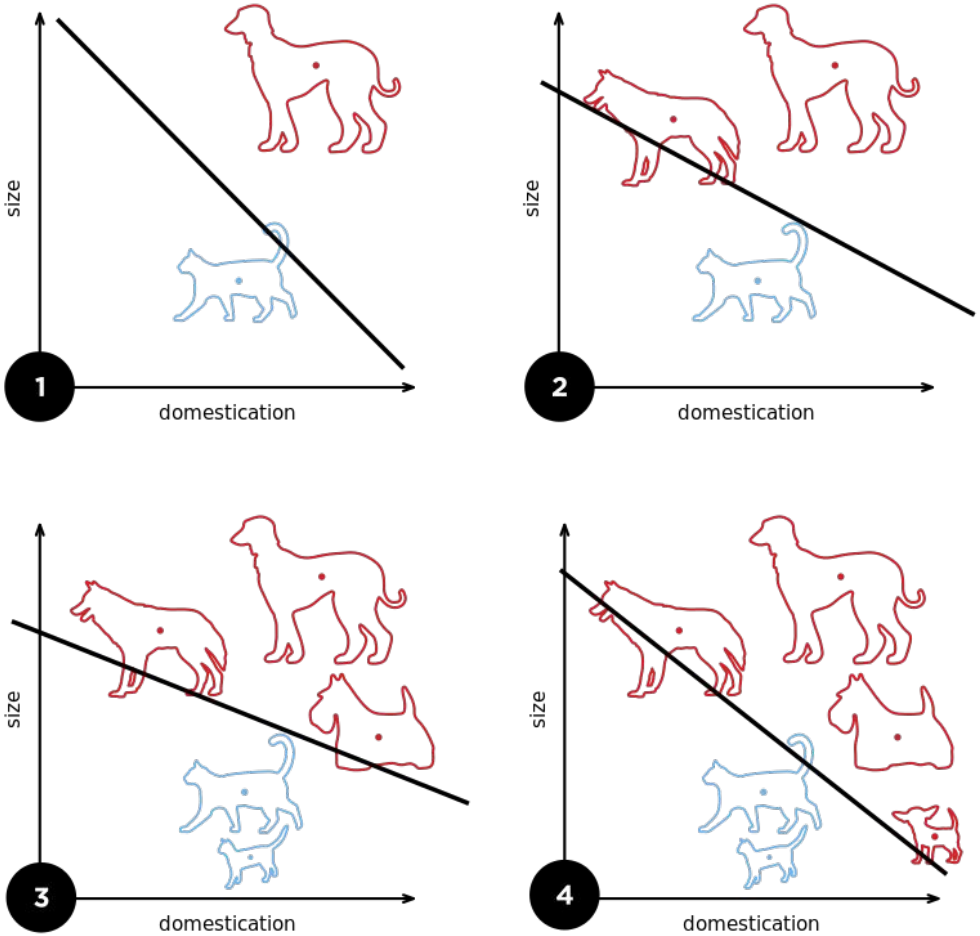


Gradient Search
iteration: 1
m = 0.40, b = 0.18
slope (m)
y intercept (b)
10
8
6
4
2
0
−2
−2
0
2
4
6
8
10
Data and Current Line
y
x
25
20
15
10
5
0
−5
−5
0
5
10
15
20

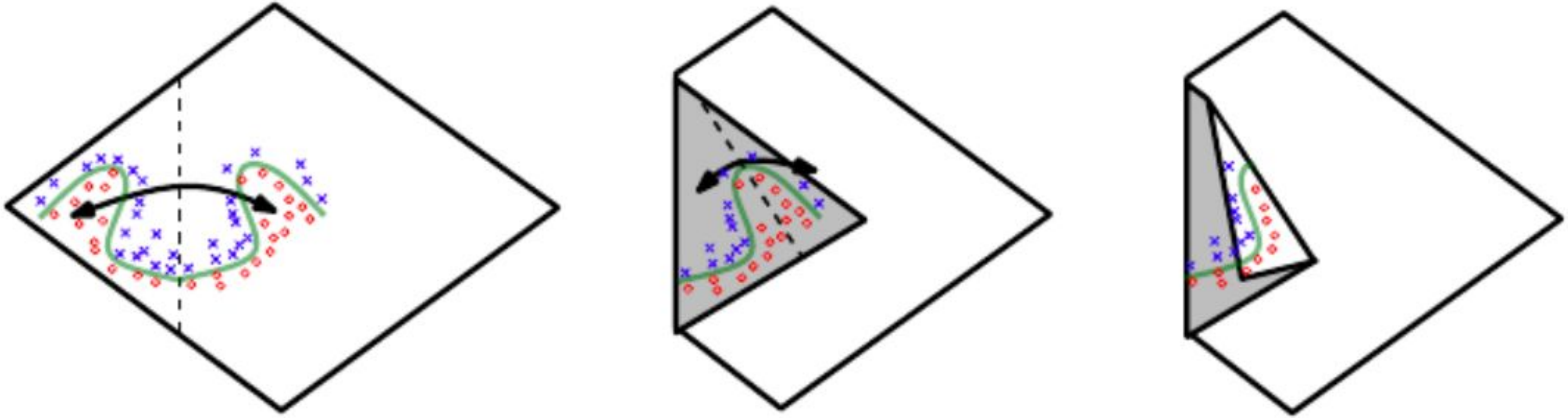

Gradient Search
iteration: 2
m = 0.96, b = 0.25
slope (m)
y intercept (b)
10
8
6
4
2
0
−2
−2
0
2
4
6
8
10
Data and Current Line
y
x
25
20
15
10
5
0
−5
−5
0
5
10
15
20


Gradient Search
iteration: 100
m = 1.30, b = 0.61
slope (m)
y intercept (b)
Data and Current Line
y
x


Gradient Search
iteration: 2000
m = 1.00, b = 3.07
slope (m)
y intercept (b)
Data and Current Line
y
x

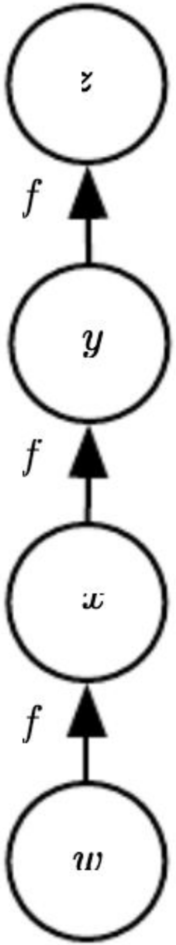

1
size
domestication
2
size
domestication
3
size
domestication
4
size
domestication

http://srome.github.io/Visualizing-the-Learning-of-a-Neural-Network-Geometrically/



https://playground.tensorflow.org

backpropagation



$$
\begin{aligned}
&\frac{\partial z}{\partial w} \\
=&\frac{\partial z}{\partial y}\frac{\partial y}{\partial x}\frac{\partial x}{\partial w} \\
=&f'(y)f'(x)f'(w) \\
=&f'(f(f(w)))f'(f(w))f'(w).
\end{aligned}
$$

- Labeled (*supervised*) training set: $T = \{(\mathbf{x}_k, \mathbf{y}_k) \mid k = 1, ..., N\}$
- Online **criterion function**: $C = \frac{1}{2}\sum_{n=1}^{d_o}(y_n - \hat{y}_n)^2$ where $\hat{y}_n$ is n-th MLP output
- Weight-update rule: $\Delta w_{ij} = -\eta \frac{\partial C}{\partial w_{ij}}$ (*Note*: w_{ij} is the connection weight between j-th unit in a given layer and i-th unit in the following layer)
- Activation function for i-th unit: $f_i(a_i)$, where:
  - $f_i : \mathcal{R} \rightarrow \mathcal{R}$
  - $a_i = \sum_j w_{ij} f_j(a_j)$ is the input to i-th unit (*Note*: the sum is extended to all the units in the previous layer)

https://www.uow.edu.au/~markus/teaching/CSCI323/Lecture_MLP.pdf

BP Case 1: i is in the output layer

$$\begin{aligned}
\frac{\partial C}{\partial w_{ij}} &= \frac{1}{2}\sum_{n=1}^{d_o} \frac{\partial (y_n - \hat{y}_n)^2}{\partial w_{ij}} && (1)\\
&= \frac{1}{2}\frac{\partial (y_i - \hat{y}_i)^2}{\partial w_{ij}}\\
&= -(y_i - \hat{y}_i)\frac{\partial \hat{y}_i}{\partial w_{ij}}\\
\frac{\partial \hat{y}_i}{\partial w_{ij}} &= \frac{\partial f_i(a_i)}{\partial w_{ij}} && (2)\\
&= \frac{\partial f_i(a_i)}{\partial a_i}\frac{\partial a_i}{\partial w_{ij}}\\
&= f_i'(a_i)\frac{\partial \sum_l w_{il}\hat{y}_l}{\partial w_{ij}}\\
&= f_i'(a_i)\hat{y}_j
\end{aligned}$$

https://www.uow.edu.au/~markus/teaching/CSCI323/Lecture_MLP.pdf

where the sum over l is extended to all the units in the (first) hidden layer.

From Eqs. (1) and (2) we have:

$$\frac{\partial C}{\partial w_{ij}} = -(y_i - \hat{y}_i) f_i'(a_i) \hat{y}_j \qquad (3)$$

We define:

$$\delta_i = (y_i - \hat{y}_i) f_i'(a_i) \qquad (4)$$

We substitute it into Eq. (3), and we can (finally) write:

$$\Delta w_{ij} = \eta \delta_i \hat{y}_j \qquad (5)$$

https://www.uow.edu.au/~markus/teaching/CSCI323/Lecture_MLP.pdf

3P Case 2: unit j in the (topmost) hidden layer

Let w_{jk} be the weight between k-th unit in the previous layer (either hidden, or input layer) and j-th unit in the topmost hidden layer:

$$\Delta w_{jk} = -\eta \frac{\partial C}{\partial w_{jk}} \tag{6}$$

Again:

$$\begin{aligned} \frac{\partial C}{\partial w_{jk}} &= \frac{1}{2} \sum_{n=1}^{d_o} \frac{\partial (y_n - \hat{y}_n)^2}{\partial w_{jk}} \\ &= -\sum_{n=1}^{d_o} (y_n - \hat{y}_n) \frac{\partial \hat{y}_n}{\partial w_{jk}} \end{aligned} \tag{7}$$

where:

https://www.uow.edu.au/~markus/teaching/CSCI323/Lecture_MLP.pdf

$$\begin{aligned}
\frac{\partial \hat{y}_n}{\partial w_{jk}} &= \frac{\partial f_n(a_n)}{\partial w_{jk}} \\
&= \frac{\partial f_n(a_n)}{\partial a_n} \frac{\partial a_n}{\partial w_{jk}} \\
&= f_n'(a_n) \frac{\partial a_n}{\partial w_{jk}}
\end{aligned} \tag{8}$$

and

$$\begin{aligned}
\frac{\partial a_n}{\partial w_{jk}} &= \frac{\partial \sum_l w_{nl} \hat{y}_l}{\partial w_{jk}} \\
&= \sum_l w_{nl} \frac{\partial \hat{y}_l}{\partial w_{jk}} \\
&= w_{nj} \frac{\partial \hat{y}_j}{\partial w_{jk}}
\end{aligned} \tag{9}$$

https://www.uow.edu.au/~markus/teaching/CSCI323/Lecture_MLP.pdf

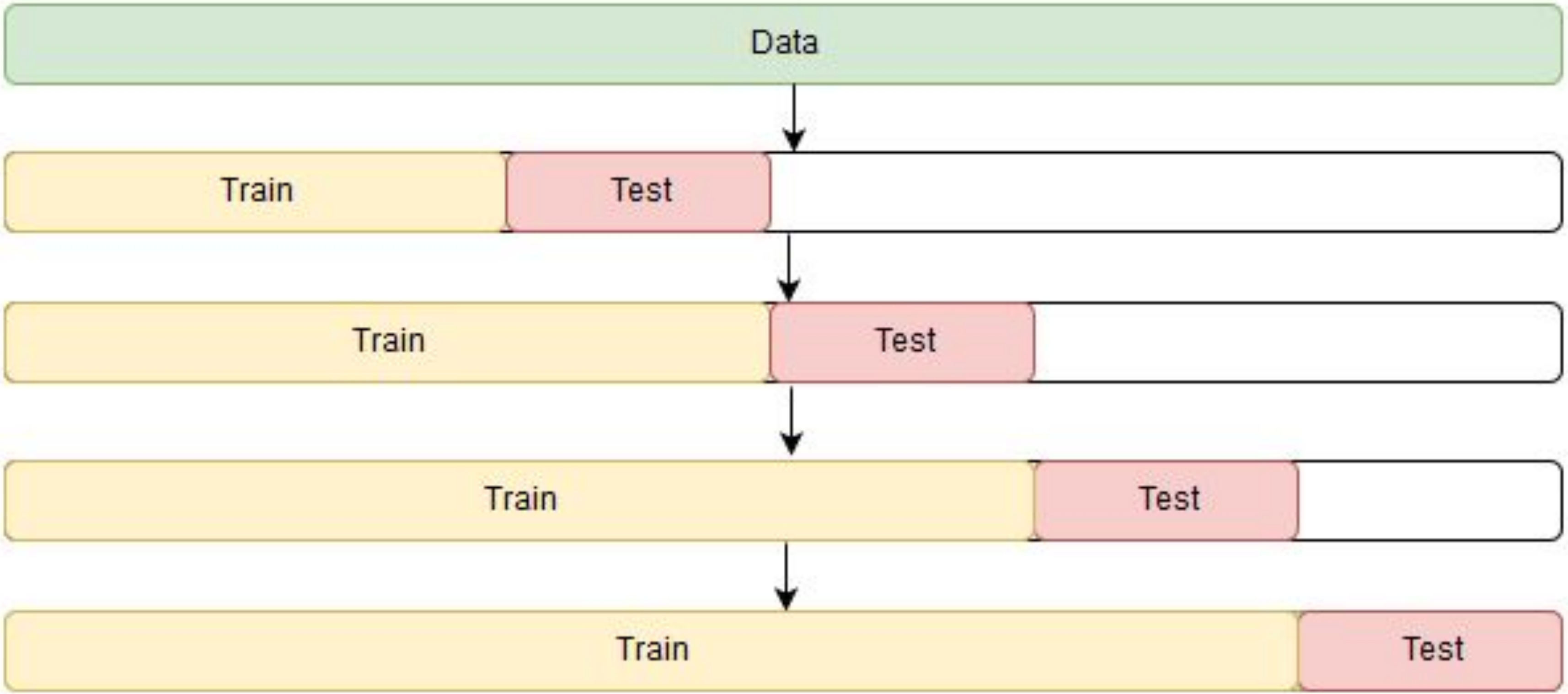
data preparation

- trend and seasonality removal
- first order difference
- clipping / outlier removal

- standardization $$\hat{a} = low + \frac{(high - low) * (a - \min A)}{\max A - \min A}$$

- normalization $$\hat{a} = \frac{a - \mu(a)}{\partial(a)}$$

evaluation


Data
Train
Test
Train
Test
Train
Test
Train
Test