An introduction to Deep Learning

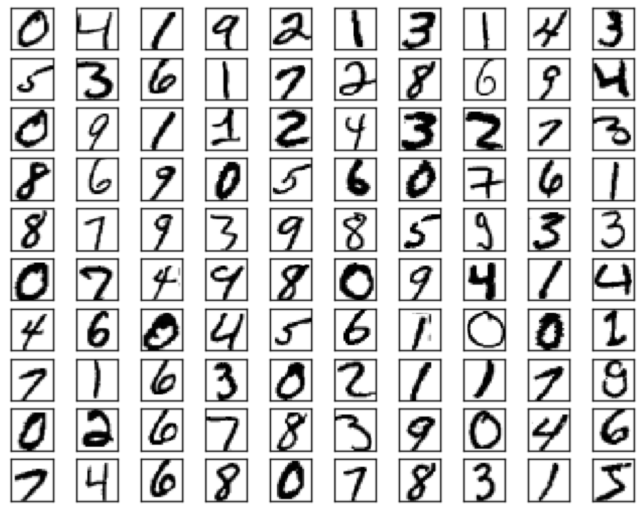
More at neuralnetworksanddeeplearning.com

Winter School on Quantitative Biology
Learning and Artificial Intelligence

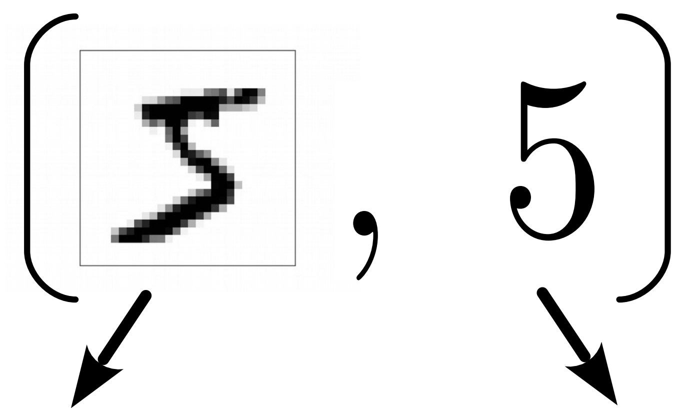
ICTP – November 2018

Let us play with digits...
... the MNIST database

Classification of handwritten digits



70’000 input/output pairs



How to use them?

Input x

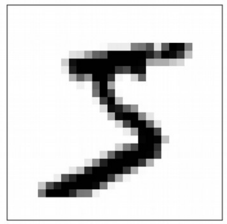
$28 \times 28, \ [0,\ 1]^{784}$

Correct output $y(x)$

$\{0, 1 \ldots 9\}$

Let us play with digits...

... the MNIST database

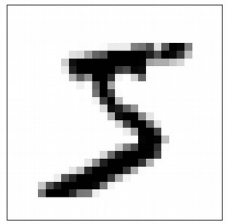
$\Big\{ \Big($  $, \; 5 \Big) \quad \ldots \quad \Big\}$

Input x

Correct output $y(x)$

Let us play with digits...

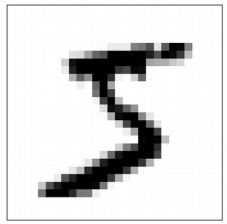
... the MNIST database

$\{ ($  $, \ 5) \quad \ldots \}$

Input x Correct output $y(x)$

$$f_{\bar{w},\bar{b}} : [0,1]^{784} \to \{0, 1 \ldots 9\}$$

Let us play with digits...
... the MNIST database

$\Big\{\Big($  $,\ 5\Big)\ \ldots\ \Big\}$

Input x Correct output $y(x)$

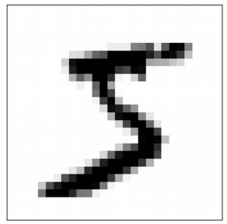
$$f_{\bar{w},\bar{b}}:\ [0,1]^{784} \to \{0,\ 1\ldots 9\}$$

A parametrized function...

How to **construct** it?

Let us play with digits...

... the MNIST database

$\Big\{ \Big($  $,\ 5 \Big) \quad \dots \Big\}$

Input x Correct output $y(x)$

$$f_{\bar{w},\bar{b}} : [0,1]^{784} \to \{0, 1 \dots 9\}$$

A parametrized function...

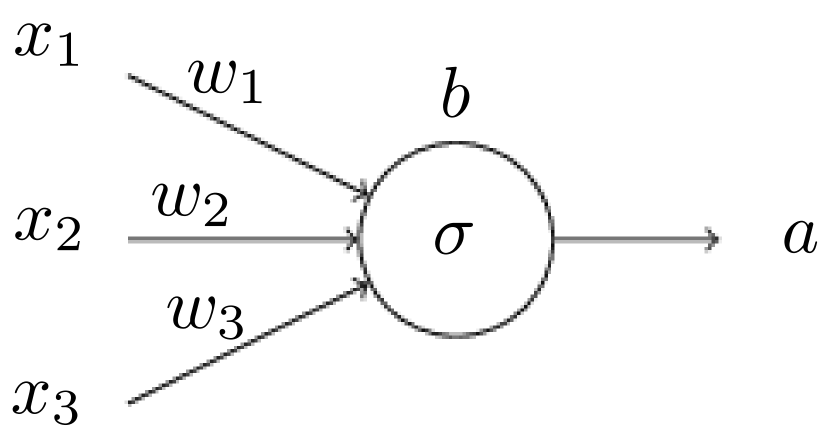
How to **construct** it?

Feed-Forward Neural Networks

Feed-Forward Neural Networks…

…the ingredients

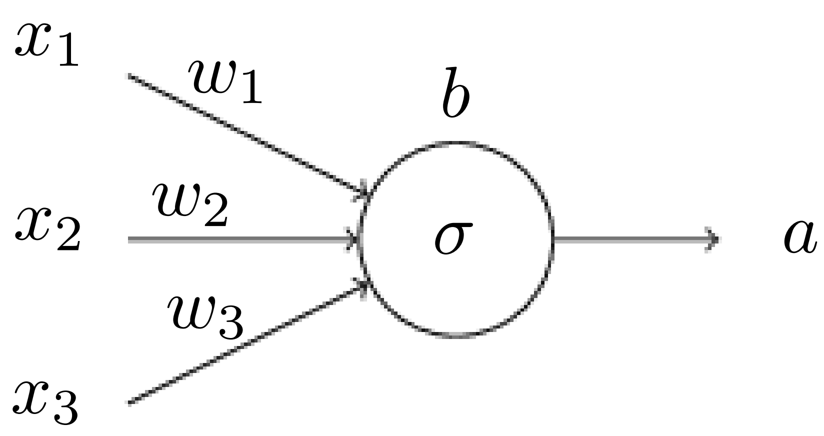
from the input
or
from other neurons



to output
or
to other neurons

Feed-Forward Neural Networks...
...the ingredients

from the input
or
from other neurons

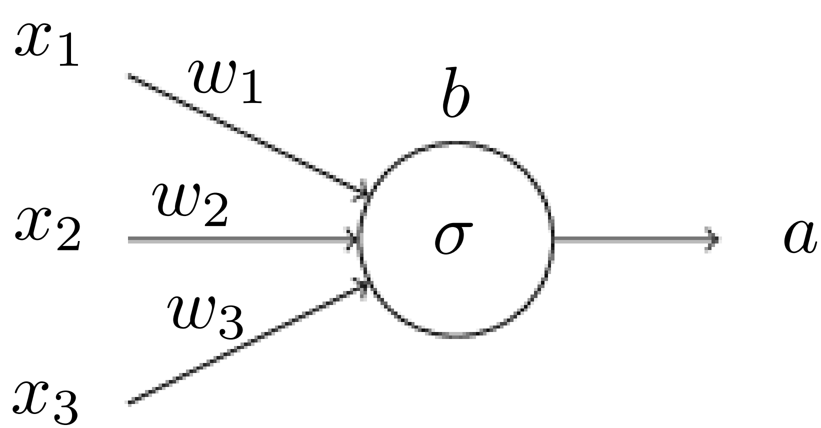


to output
or
to other neurons

$$\underbrace{\textbf{weights } w \qquad \textbf{bias } b \qquad \textbf{activity } a}$$

Feed-Forward Neural Networks...
...the ingredients

from the input
or
from other neurons



to output
or
to other neurons

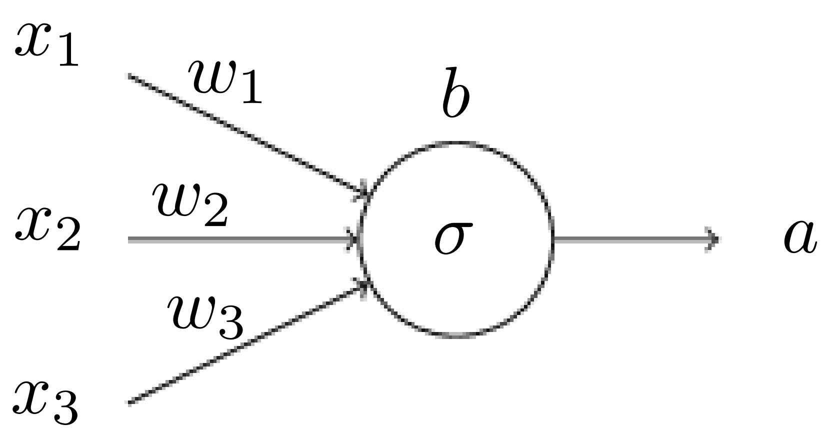
weights w **bias** b **activity** a

weighed input $z = b + \sum_i w_i \, x_i$

activation function σ, $\quad a = \sigma(z)$

Feed-Forward Neural Networks...
...the ingredients

from the input
or
from other neurons



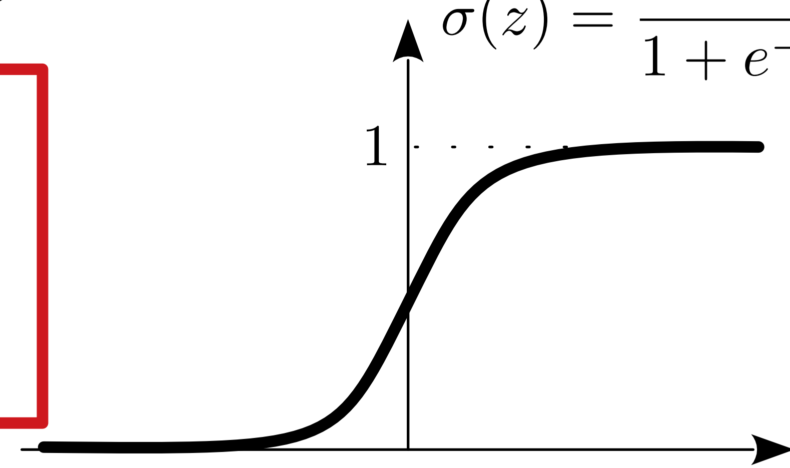
to output
or
to other neurons

weights w **bias** b **activity** a

weighed input $z = b + \sum_i w_i x_i$

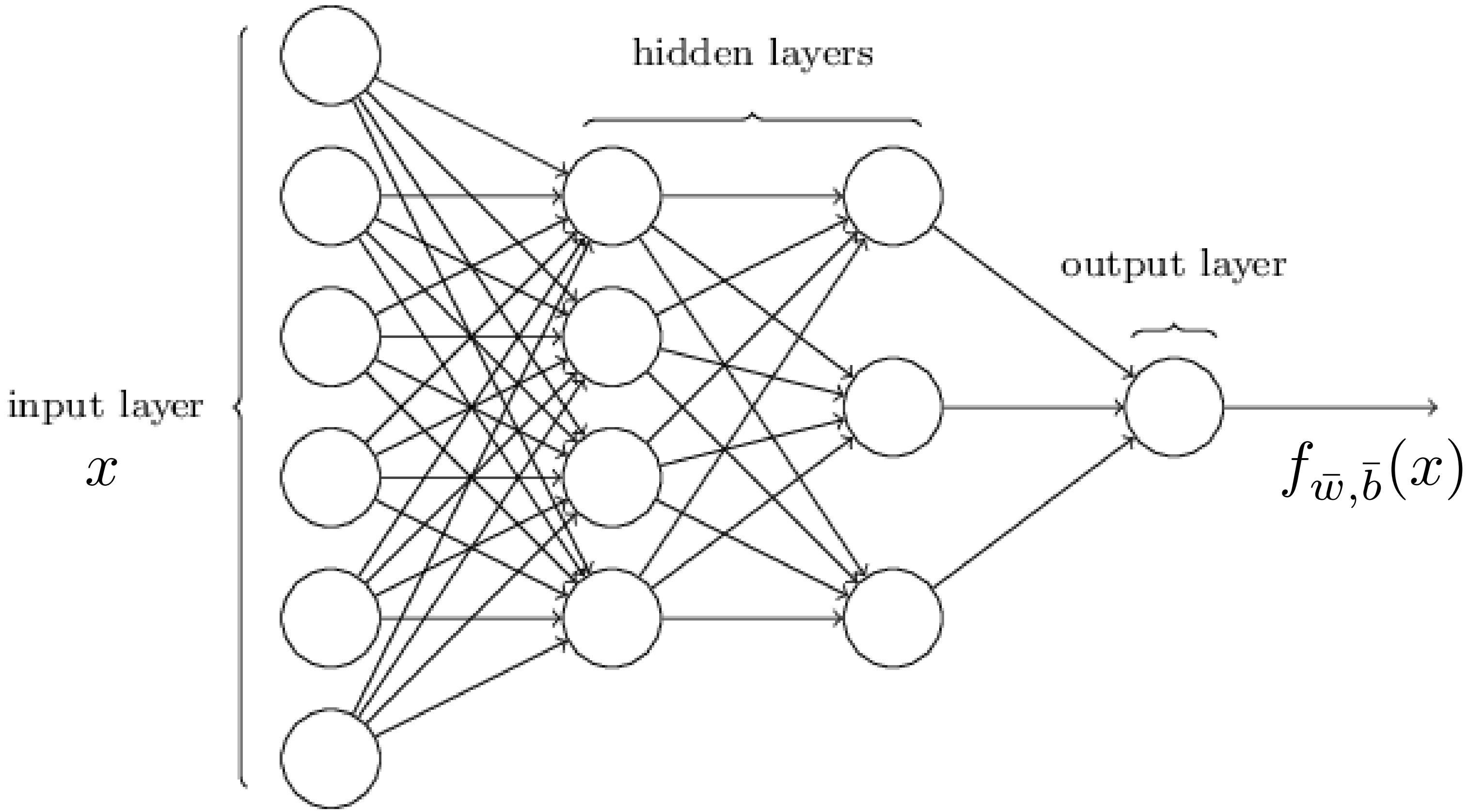
activation function σ, $a = \sigma(z)$

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

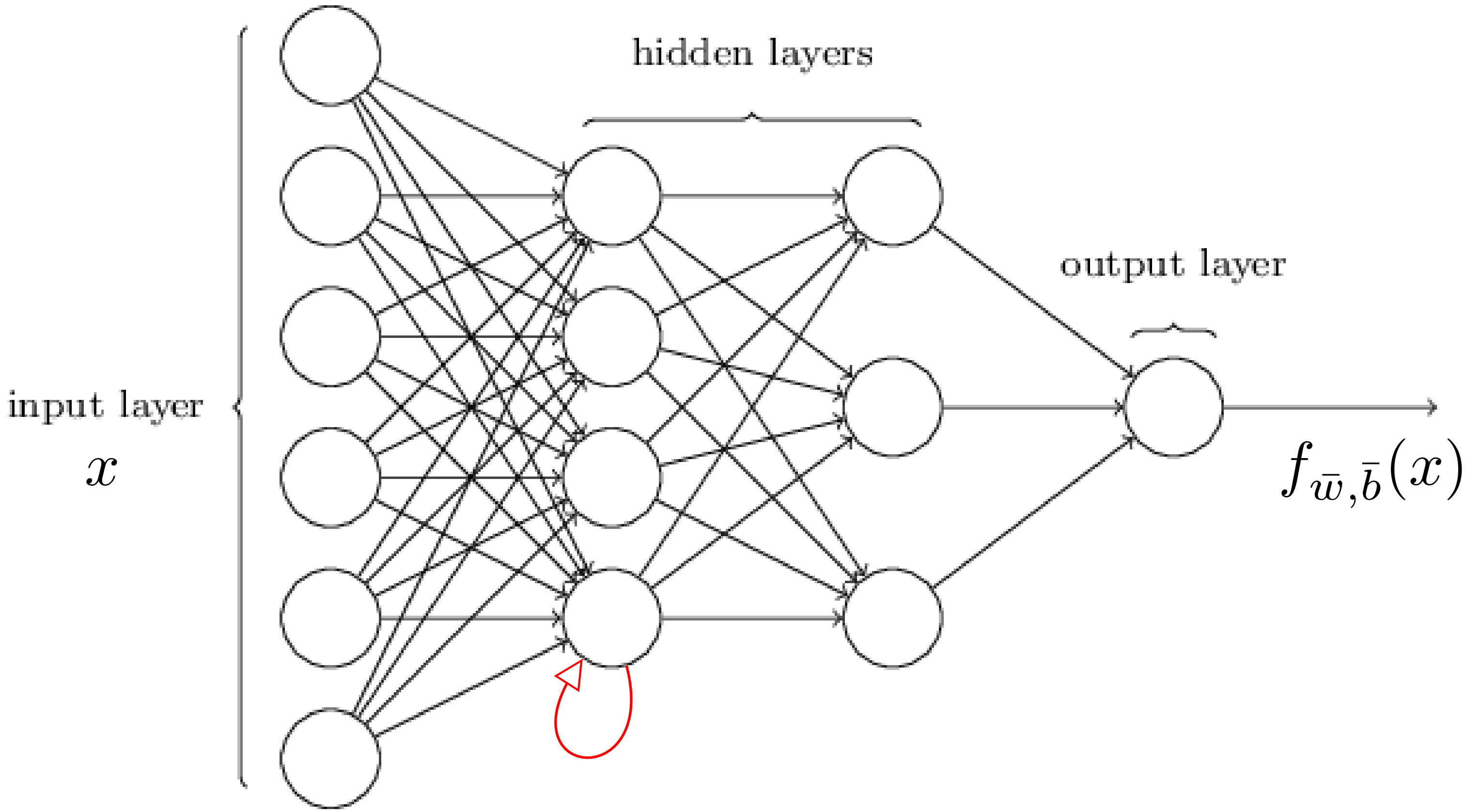


Feed-Forward Neural Networks...

...the architecture

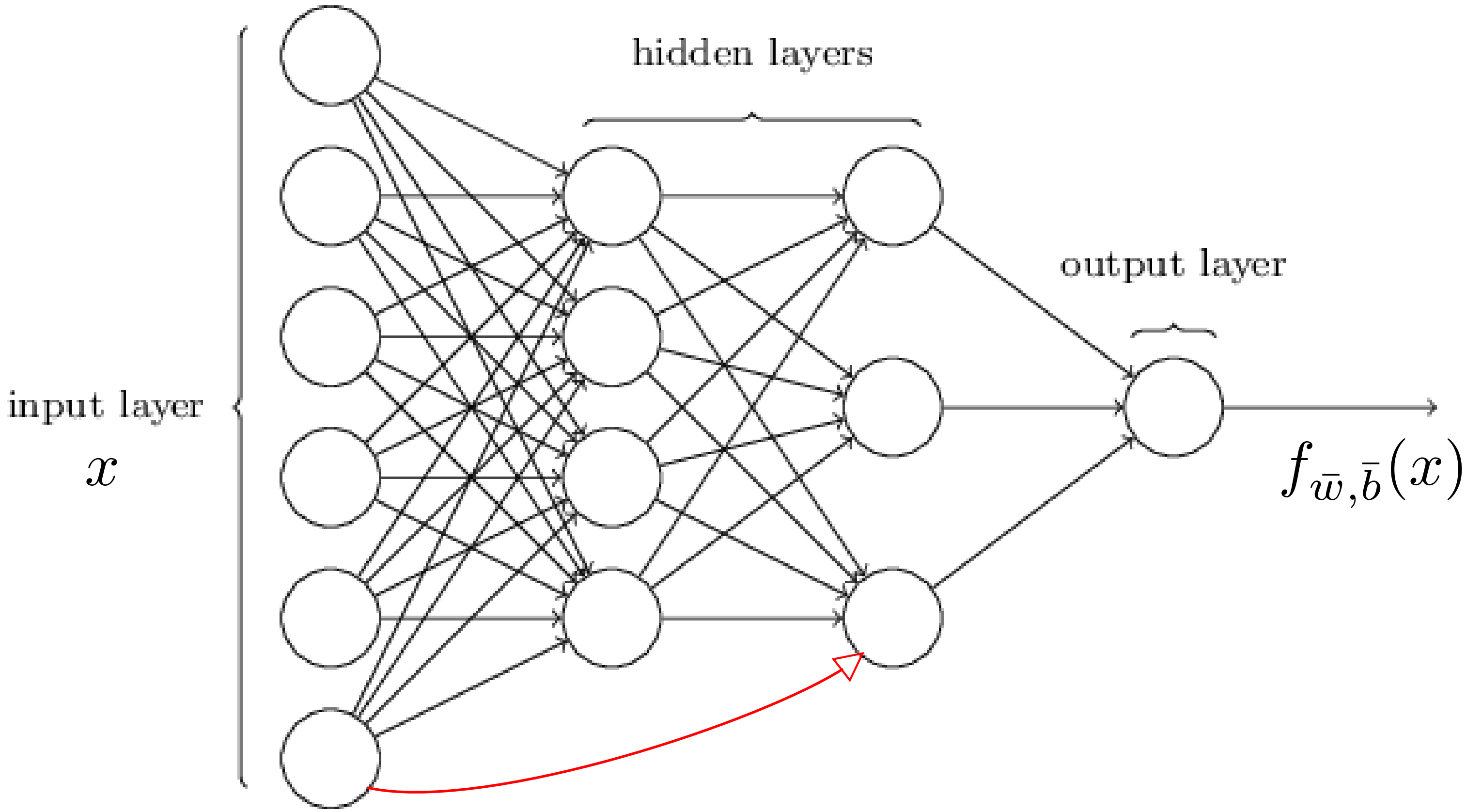


Feed-Forward Neural Networks…
…the architecture



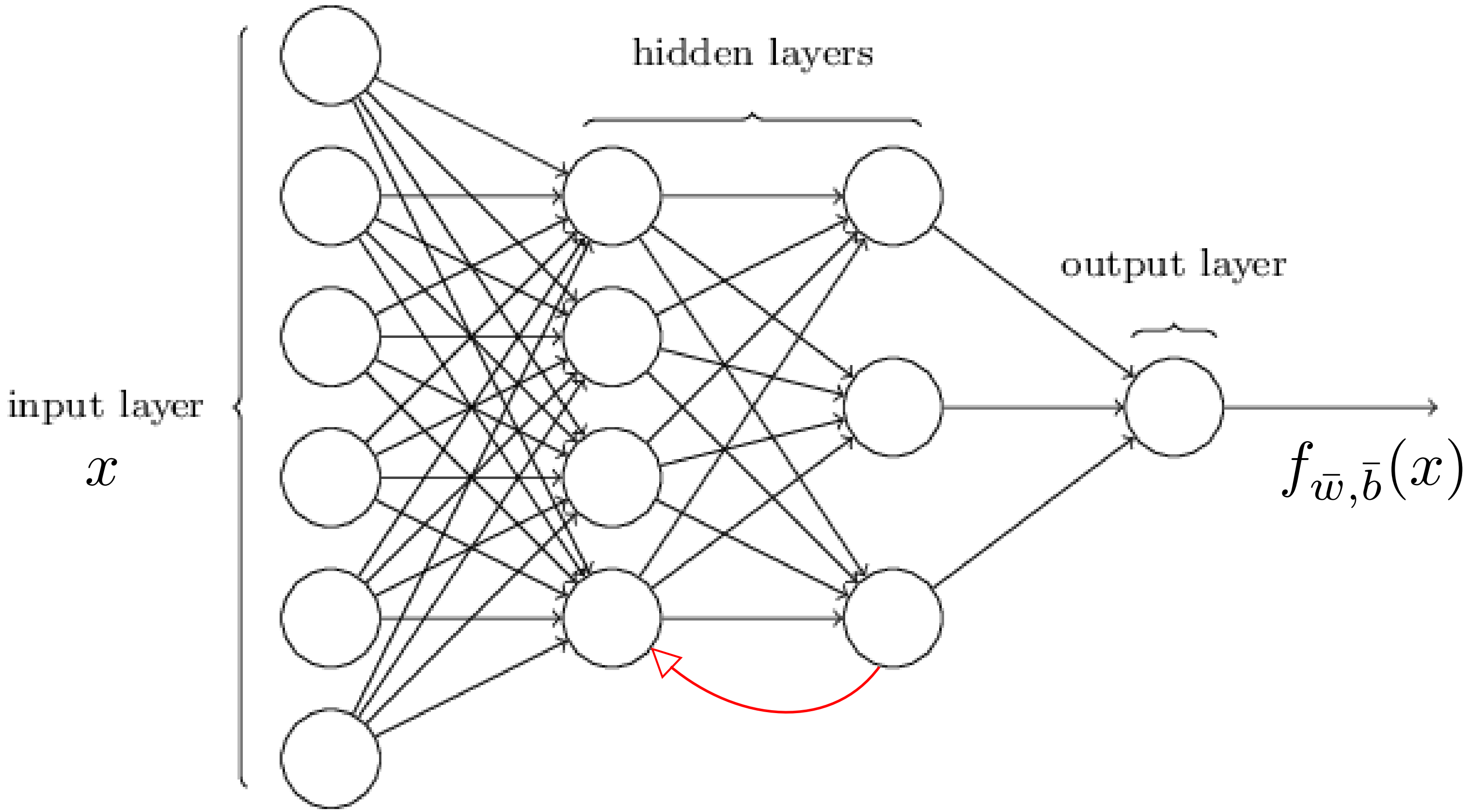
Feed-Forward Neural Networks...

...the architecture



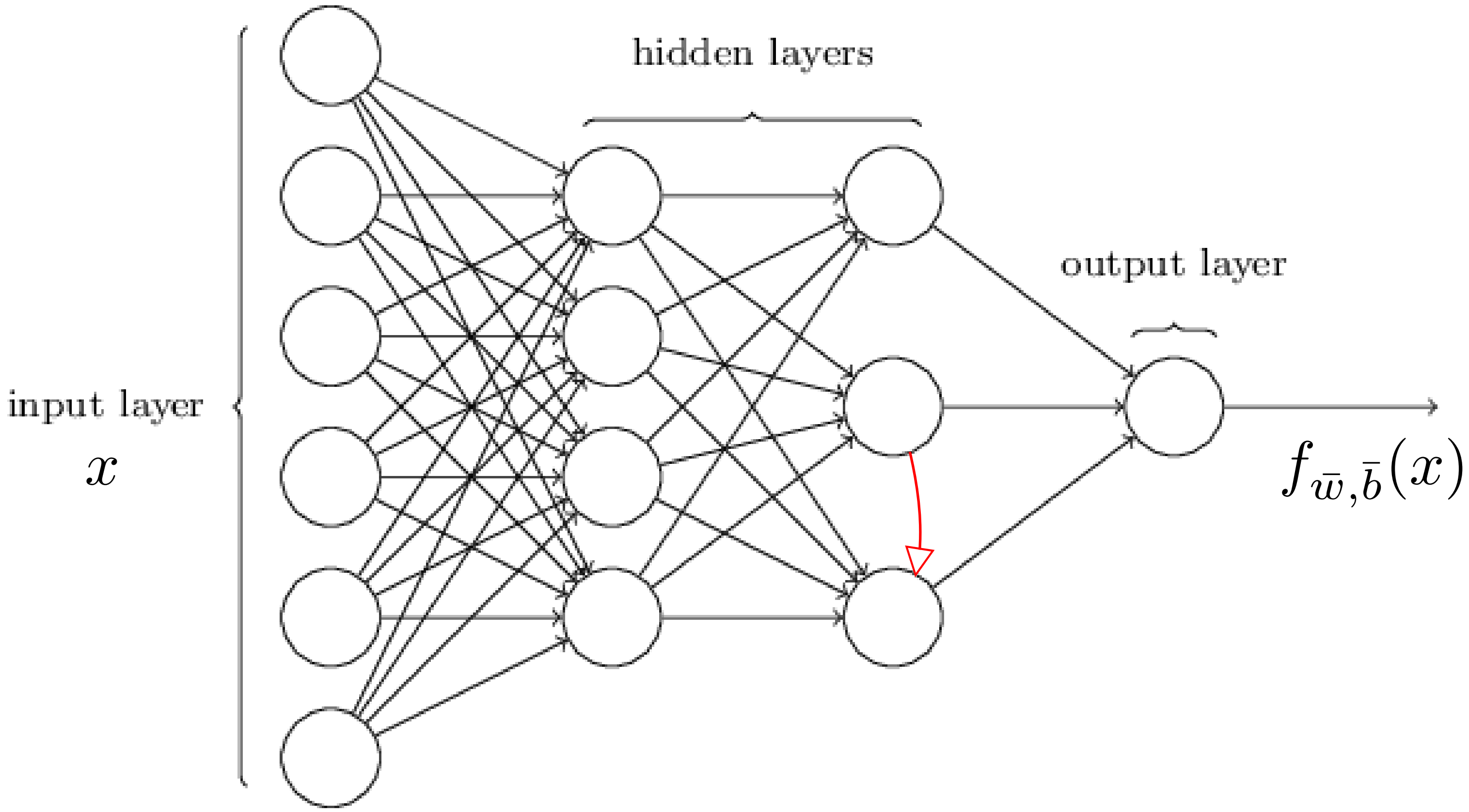
Feed-Forward Neural Networks...

...the architecture

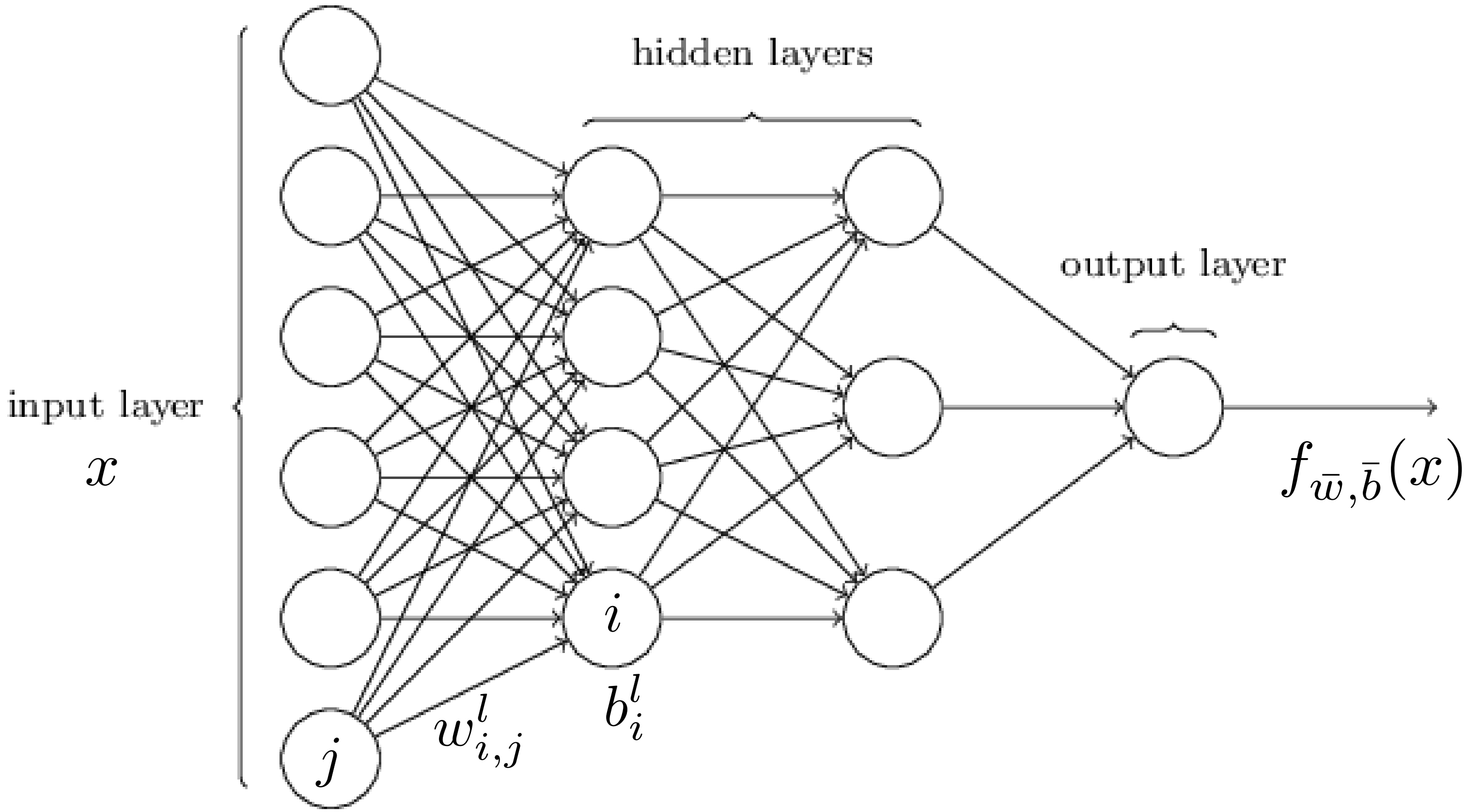


Feed-Forward Neural Networks…

…the architecture



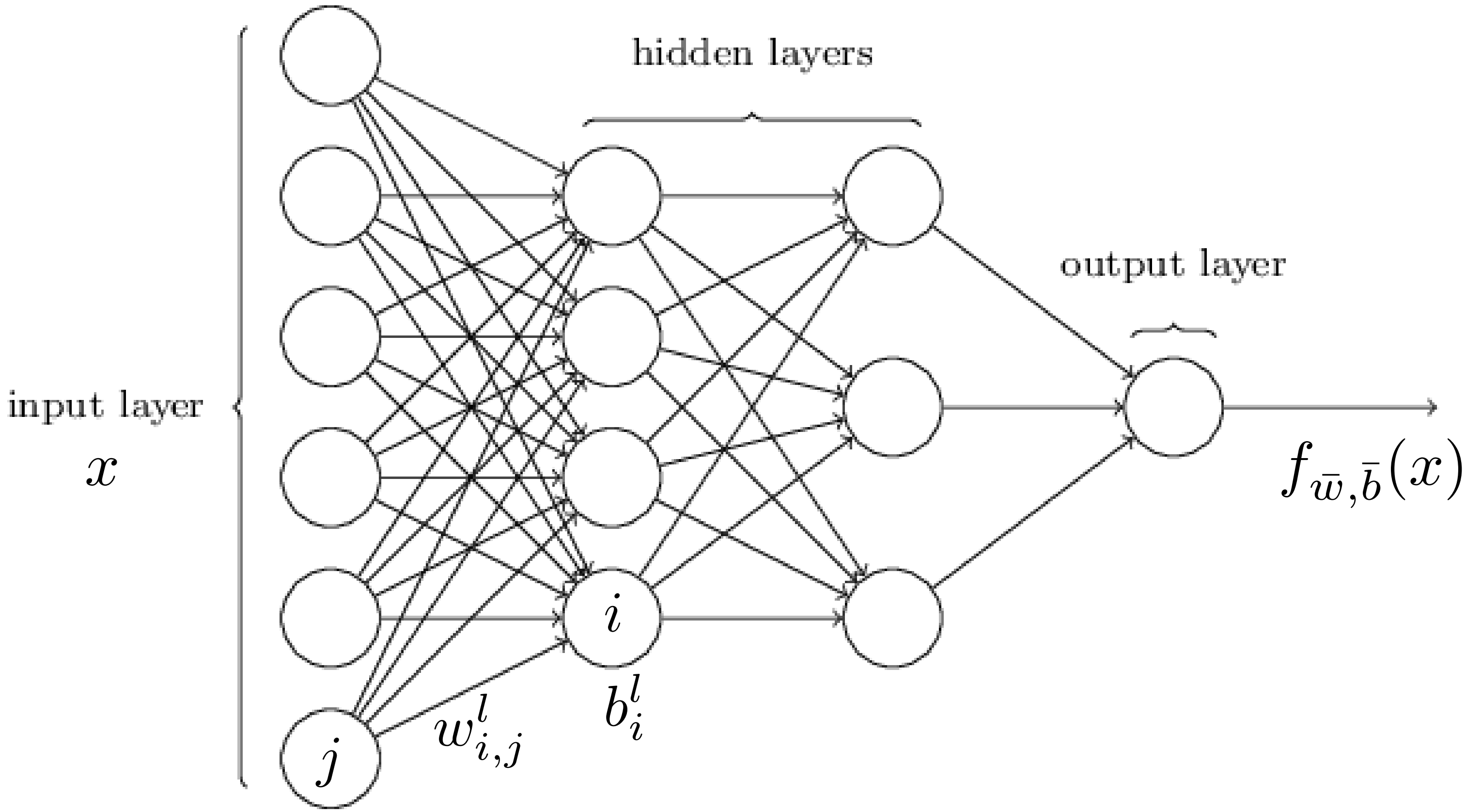
Feed-Forward Neural Networks...
...the architecture



one **weight** per arrow
one **bias** per neuron

$w^l_{i,j}$ b^l_i

Feed-Forward Neural Networks…
…the architecture

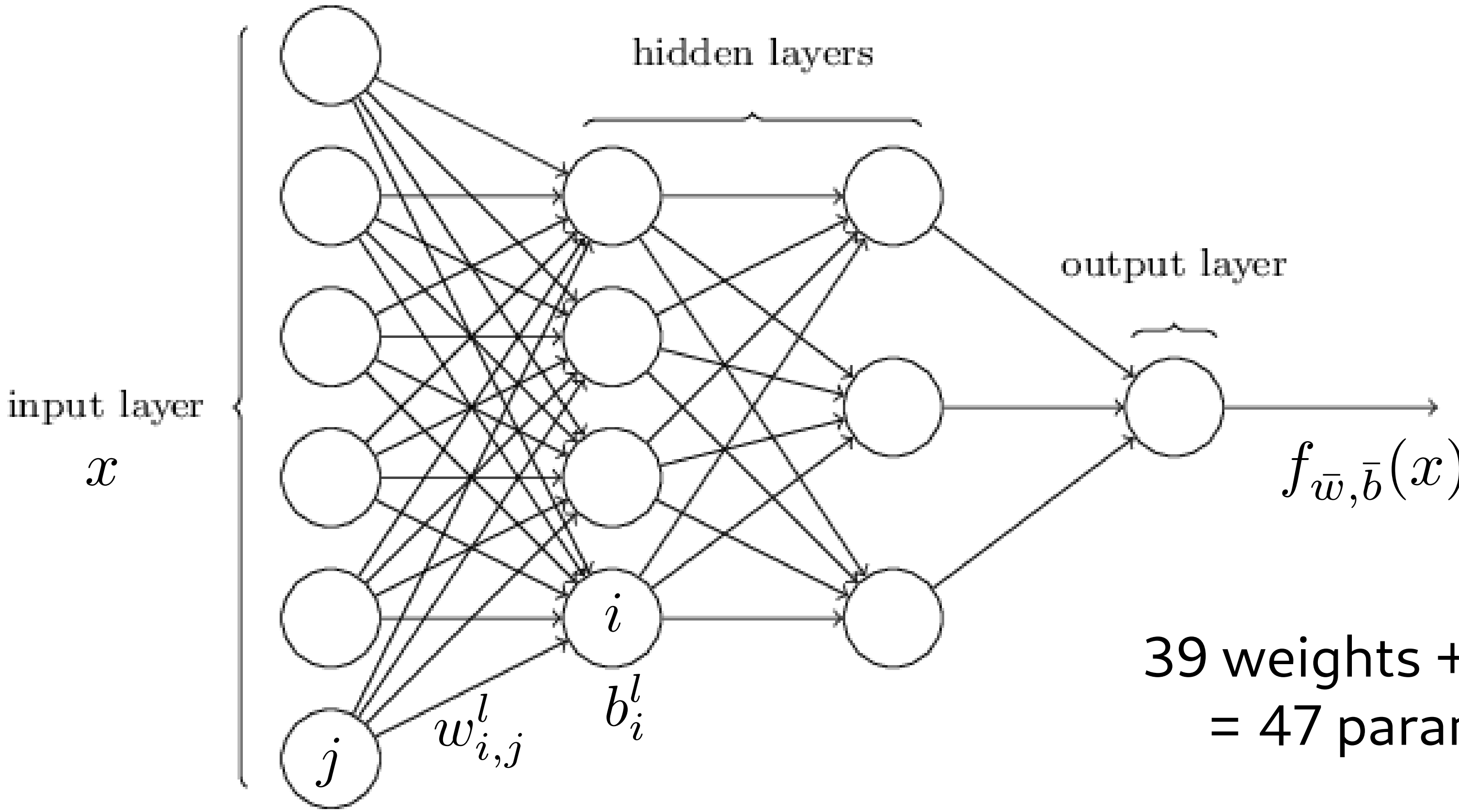


one **weight** per arrow
one **bias** per neuron

$$w^l_{i,j} \qquad b^l_i$$

$$\begin{aligned} a^l_i &= \sigma\left(z^l_i\right) \\ &= \sigma\left(b^l_i + \sum_j w^l_{i,j} a^{l-1}_j\right) \end{aligned}$$

Feed-Forward Neural Networks…
…the architecture



39 weights + 8 biases
= 47 parameters

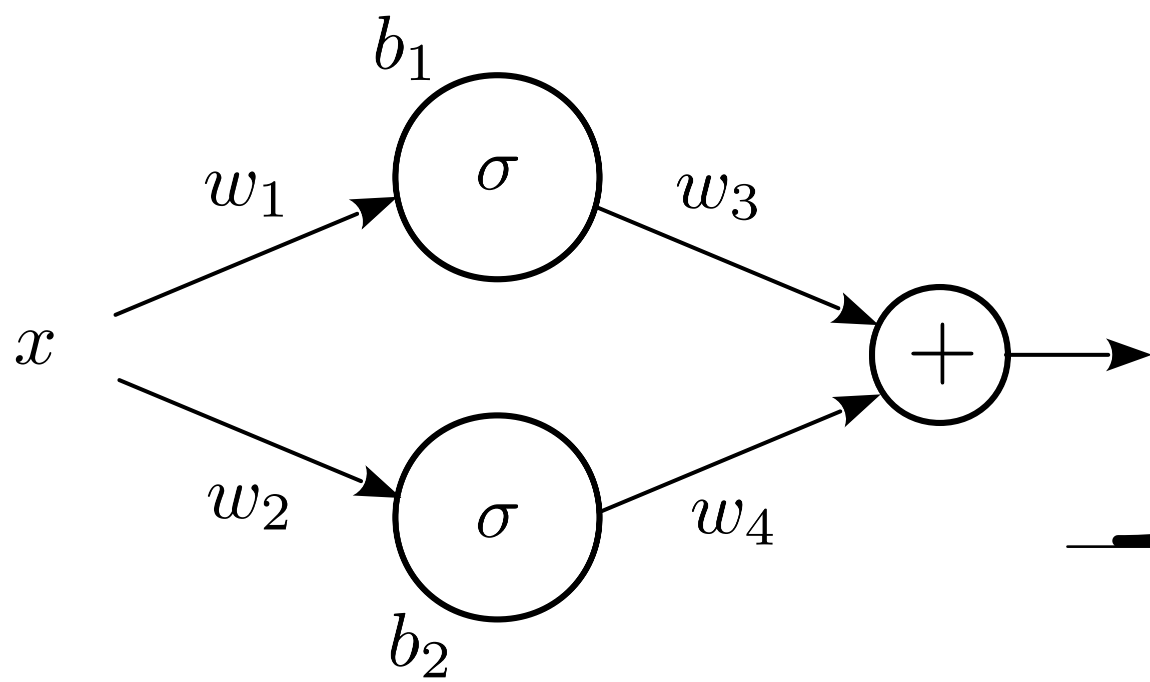
one **weight** per arrow
one **bias** per neuron

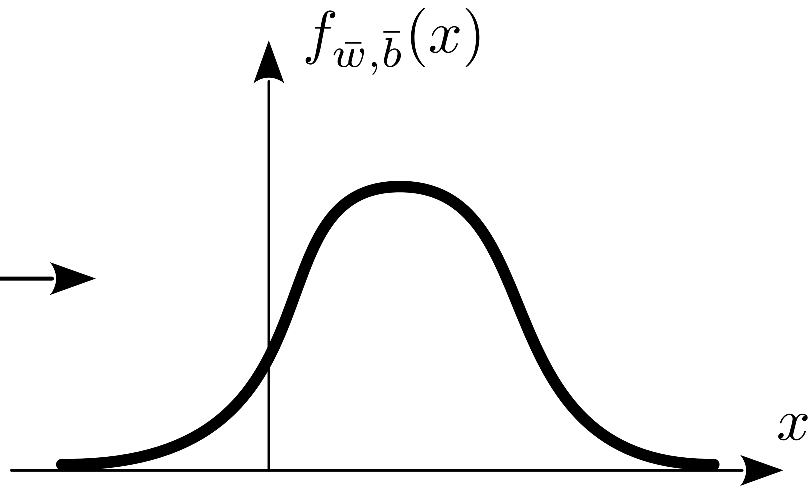
$w_{i,j}^l \qquad b_i^l$

$$a_i^l = \sigma\left(z_i^l\right) = \sigma\left(b_i^l + \sum_j w_{i,j}^l a_j^{l-1}\right)$$

Feed-Forward Neural Networks…

…universal function approximators





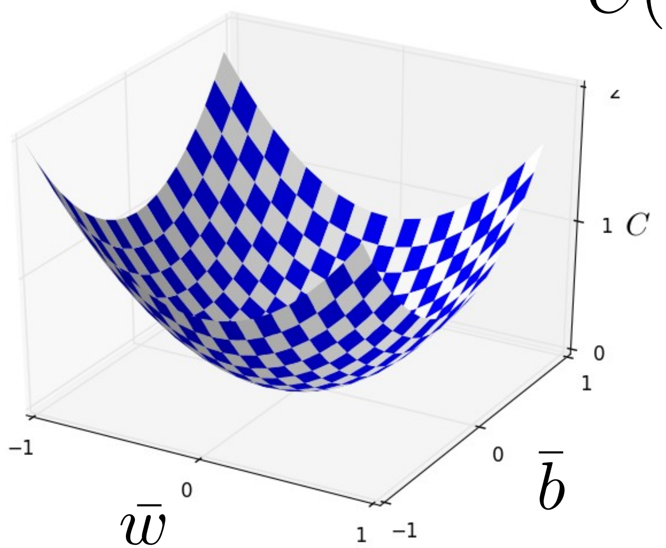
$$f_{\bar{w},\bar{b}}(x) = w_3\,\sigma\big(b_1 + w_1 x\big) + w_4\,\sigma\big(b_2 + w_2 x\big)$$

[Cybenko, G. (1989) *Approximations by superpositions of sigmoidal functions*]

Learning through gradient descent

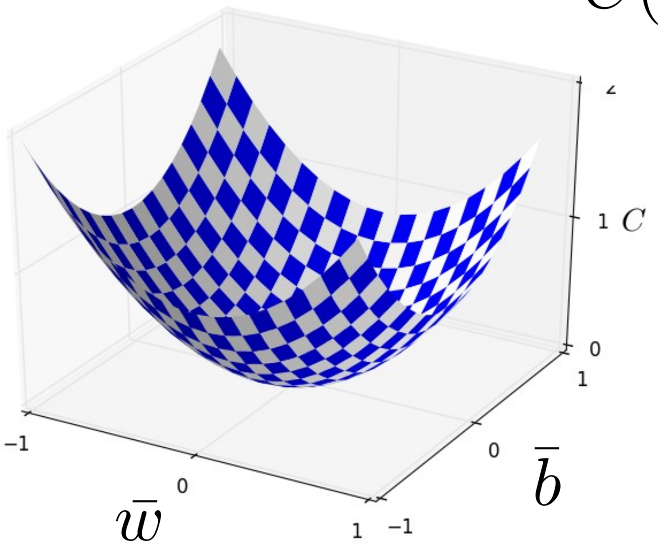
Minimize a **cost function**:

$$C(\bar{w}, \bar{b}) = \left\langle \frac{1}{2} \left\| \text{output} - \text{desired one} \right\|^2 \right\rangle_{\text{training data}}$$



Learning through gradient descent

Minimize a **cost function**:



$$C(\bar{w},\bar{b}) = \left\langle \frac{1}{2} \left\| \text{output} - \text{desired one} \right\|^2 \right\rangle_{\text{training data}}$$

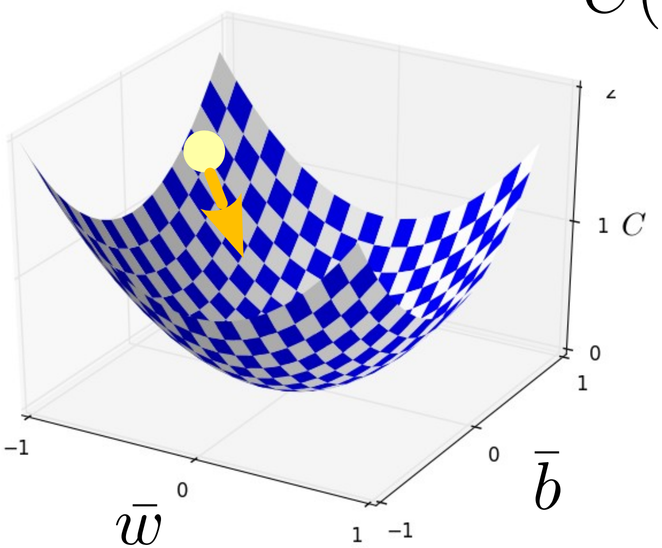
$$= \frac{1}{2N} \sum_x \sum_i \left(a_i^L(x) - y_i(x) \right)^2$$

training data

neurons in layer L (output)

Learning through gradient descent

Minimize a **cost function**:



$$C(\bar{w},\bar{b}) = \left\langle \frac{1}{2} \left\| \text{output} - \text{desired one} \right\|^2 \right\rangle_{\text{training data}}$$

$$= \frac{1}{2N} \sum_x \sum_i \left(a_i^L(x) - y_i(x) \right)^2$$

training data

neurons in layer L (output)

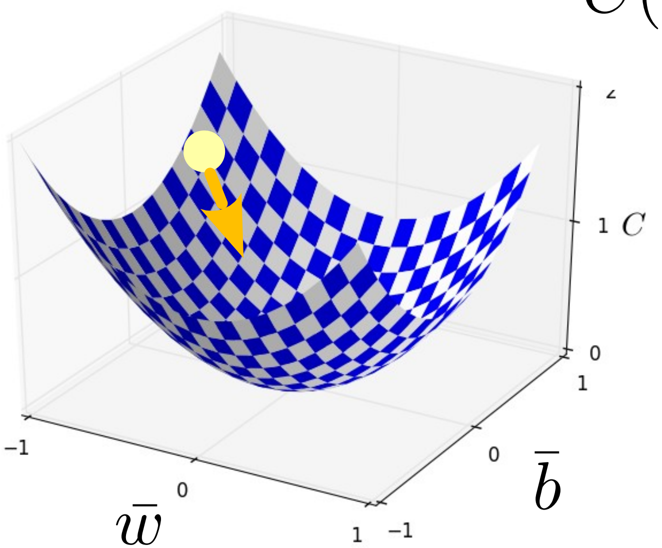
Gradient descent update rules:

$$\Delta w_{i,j}^l \propto -\frac{\partial C}{\partial w_{i,j}^l}$$

$$\Delta b_i^l \propto -\frac{\partial C}{\partial b_i^l}$$

Learning through gradient descent

Minimize a **cost function**:



$$C(\bar{w},\bar{b}) = \left\langle \frac{1}{2}\left\|\text{output} - \text{desired one}\right\|^2 \right\rangle_{\text{training data}}$$

$$= \frac{1}{2N}\sum_x \sum_i \left(a_i^L(x) - y_i(x)\right)^2$$

training data

neurons in layer L (output)

Gradient descent update rules:

$$\Delta w_{i,j}^l \propto -\frac{\partial C}{\partial w_{i,j}^l}$$

$$\Delta b_i^l \propto -\frac{\partial C}{\partial b_i^l}$$

$$\frac{\partial C}{\partial w_{i,j}^l},\ \frac{\partial C}{\partial b_i^l} = \ ?$$

The backpropagation algorithm

The **error**:

How much does the cost change as an effect of a change in the input of a given neuron in a given layer?

The backpropagation algorithm

The **error:**

How much does the cost change as an effect of a change in the input of a given neuron in a given layer?

$$\delta_i^l \equiv \frac{\Delta C}{\Delta z_i^l} \to \frac{\partial C}{\partial z_i^l}$$

The backpropagation algorithm

The **error**:

How much does the cost change as an effect of a change in the input of a given neuron in a given layer?

$$\delta_i^l \equiv \frac{\Delta C}{\Delta z_i^l} \to \frac{\partial C}{\partial z_i^l}$$

Small if the parameters are close to **optimal**.

The backpropagation algorithm

The **error**:

How much does the cost change as an effect of a change in the input of a given neuron in a given layer?

$$\delta_i^l \equiv \frac{\Delta C}{\Delta z_i^l} \to \frac{\partial C}{\partial z_i^l}$$

Small if the parameters are close to **optimal**.

$$\frac{\partial C}{\partial w_{ij}^l}$$

The backpropagation algorithm

The **error**:

How much does the cost change as an effect of a change in the input of a given neuron in a given layer?

$$\delta_i^l \equiv \frac{\Delta C}{\Delta z_i^l} \to \frac{\partial C}{\partial z_i^l}$$

Small if the parameters are close to **optimal**.

$$\frac{\partial C}{\partial w_{ij}^l} = \frac{\partial C}{\partial z_i^l} \frac{\partial z_i^l}{\partial w_{ij}^l}$$

chain rule

The backpropagation algorithm

The **error**:

How much does the cost change as an effect of a change in the input of a given neuron in a given layer?

$$\delta_i^l \equiv \frac{\Delta C}{\Delta z_i^l} \to \frac{\partial C}{\partial z_i^l}$$

Small if the parameters are close to **optimal**.

$$\frac{\partial C}{\partial w_{ij}^l} = \frac{\partial C}{\partial z_i^l} \frac{\partial z_i^l}{\partial w_{ij}^l} = \delta_i^l \, a_j^{l-1}$$

chain rule

The backpropagation algorithm

The **error**:

How much does the cost change as an effect of a change in the input of a given neuron in a given layer?

$$\delta_i^l \equiv \frac{\Delta C}{\Delta z_i^l} \to \frac{\partial C}{\partial z_i^l}$$

Small if the parameters are close to **optimal**.

$$\frac{\partial C}{\partial w_{ij}^l} = \frac{\partial C}{\partial z_i^l} \frac{\partial z_i^l}{\partial w_{ij}^l} = \delta_i^l \, a_j^{l-1}$$

chain rule

$$\frac{\partial C}{\partial b_i^l}$$

The backpropagation algorithm

The **error**:

How much does the cost change as an effect of a change in the input of a given neuron in a given layer?

$$\delta_i^l \equiv \frac{\Delta C}{\Delta z_i^l} \rightarrow \frac{\partial C}{\partial z_i^l}$$

Small if the parameters are close to **optimal**.

$$\frac{\partial C}{\partial w_{ij}^l} = \frac{\partial C}{\partial z_i^l}\,\frac{\partial z_i^l}{\partial w_{ij}^l} = \delta_i^l\, a_j^{l-1}$$

chain rule

$$\frac{\partial C}{\partial b_i^l} = \frac{\partial C}{\partial z_i^l}\,\frac{\partial z_i^l}{\partial b_i^l}$$

The backpropagation algorithm

The **error**:

How much does the cost change as an effect of a change in the input of a given neuron in a given layer?

$$\delta_i^l \equiv \frac{\Delta C}{\Delta z_i^l} \to \frac{\partial C}{\partial z_i^l}$$

Small if the parameters are close to **optimal**.

$$\frac{\partial C}{\partial w_{ij}^l} = \frac{\partial C}{\partial z_i^l} \frac{\partial z_i^l}{\partial w_{ij}^l} = \delta_i^l \, a_j^{l-1}$$

chain rule

$$\frac{\partial C}{\partial b_i^l} = \frac{\partial C}{\partial z_i^l} \frac{\partial z_i^l}{\partial b_i^l} = \delta_i^l$$

The backpropagation algorithm

Start from the **output** layer...

$$\delta_i^L = \frac{\partial C}{\partial z_i^L} = \frac{1}{N}\sum_x \Big(a_i^L(x) - y_i(x)\Big)\,\sigma'\big(z_i^L\big)$$

$$C = \frac{1}{N}\sum_x \sum_i \frac{1}{2}\big(a_i^L(x) - y_i(x)\big)^2$$

... and **back-propagate** to the previous ones

$$\delta_j^{l-1} = \frac{\partial C}{\partial z_j^{l-1}} = \sum_i \frac{\partial C}{\partial z_i^l}\,\frac{\partial z_i^l}{\partial z_j^{l-1}} = \sum_i \delta_i^l\, w_{ij}^l\,\sigma'\big(z_j^{l-1}\big)$$

The mini-batch update