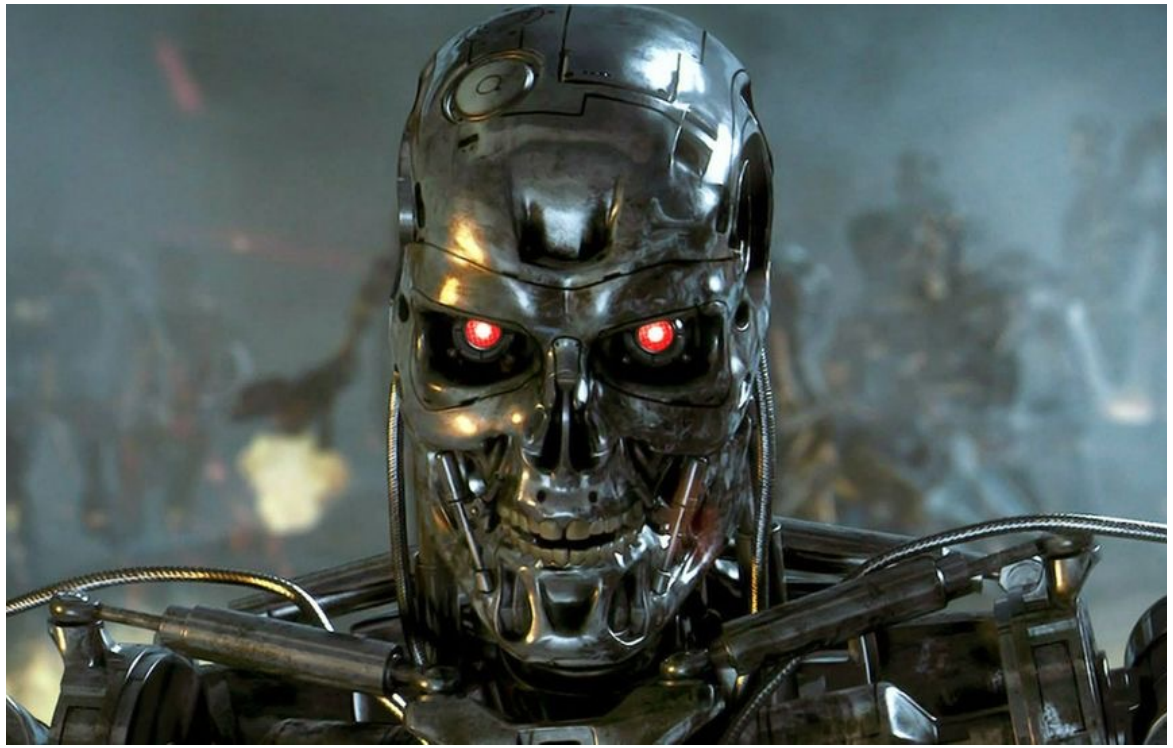
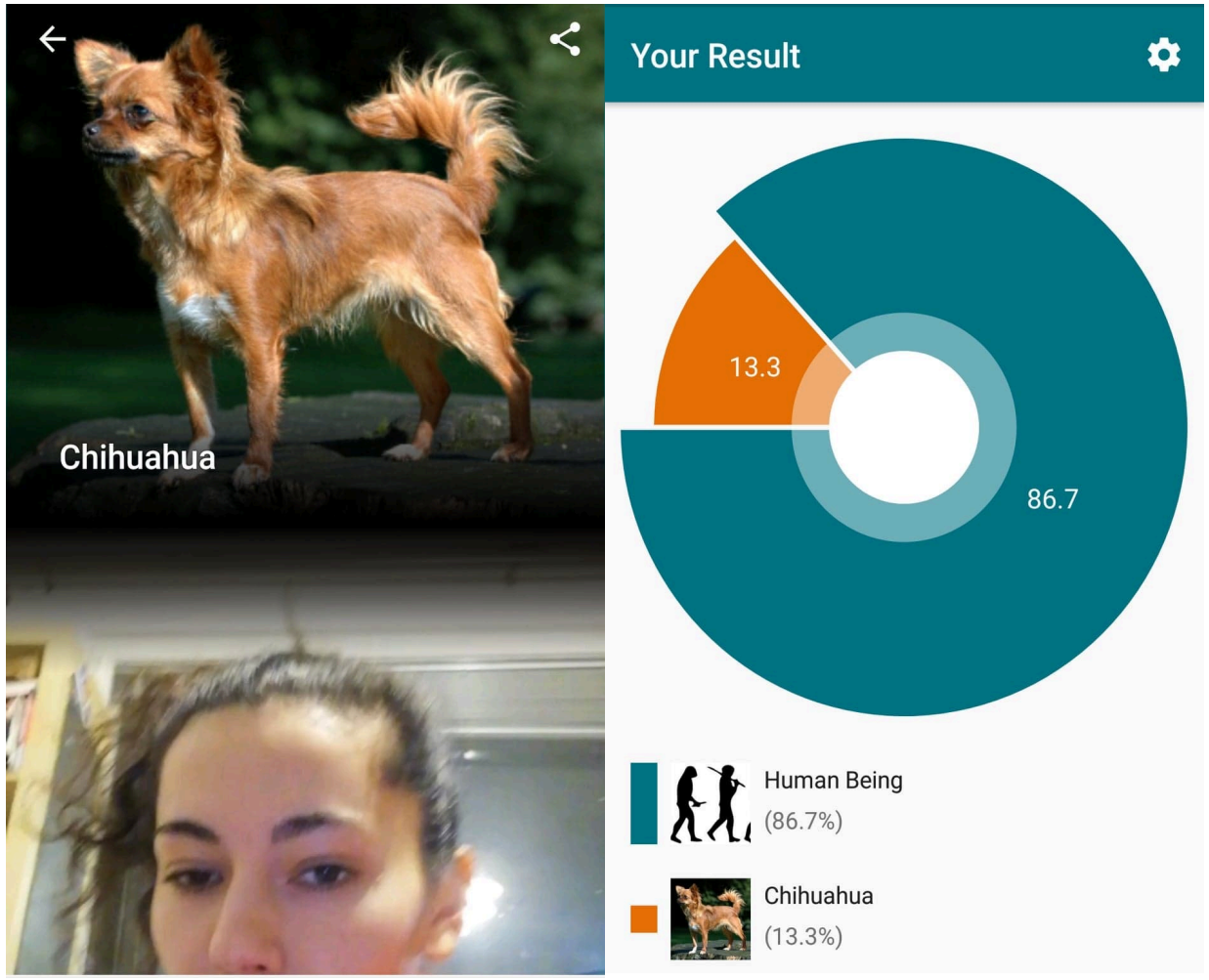
A Painter that Does Not Exist - Generative Image Production with Machines

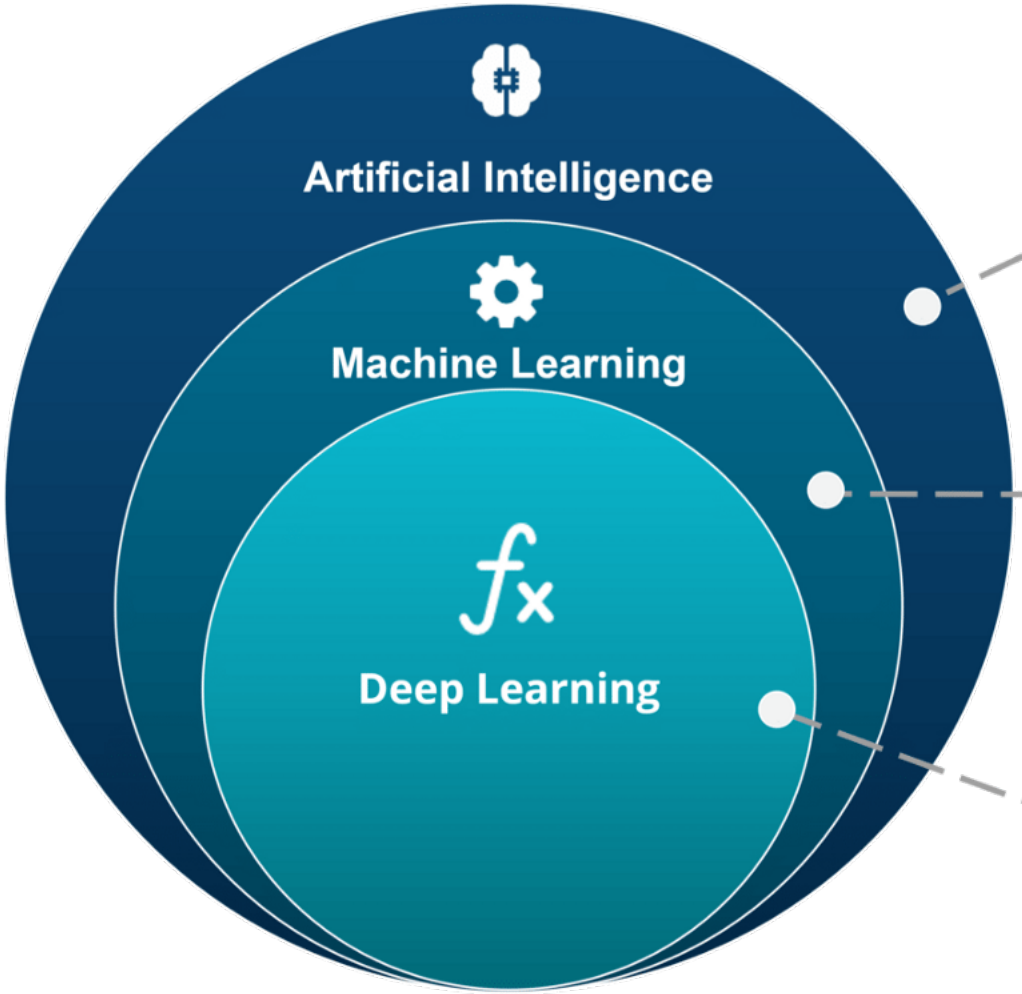
AI, machine learning, deep learning, …

Expectation



Reality





ARTIFICIAL INTELLIGENCE

A technique which enables machines to mimic human behaviour

MACHINE LEARNING

Subset of AI technique which use statistical methods to enable machines to improve with experience

DEEP LEARNING

Subset of ML which make the computation of multi-layer neural network feasible

House price prediction

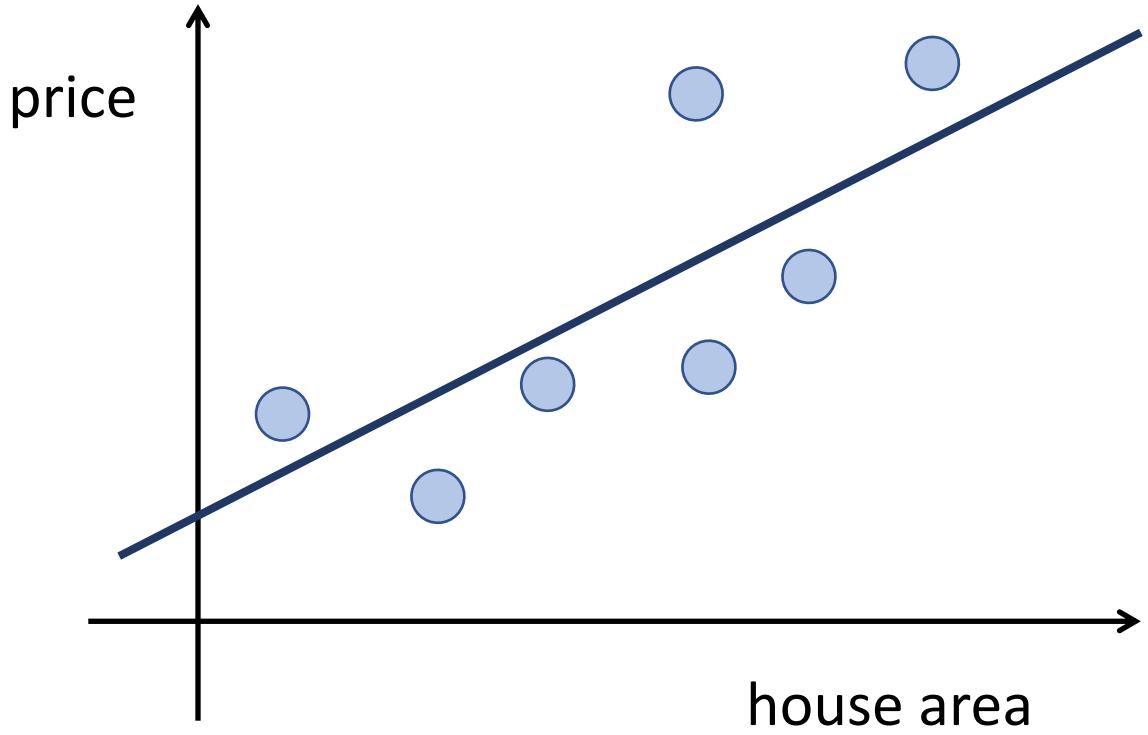
- house area,
- n.o. rooms/ bathrooms,
- neighborhood,
- building age,
- future rennovations,
- etc.

House price prediction

- house area $= x_1$
- n.o. rooms/ bathrooms $= x_2$
- neighborhood $= x_3$
- building age $= x_4$
- future rennovations $= x_5$

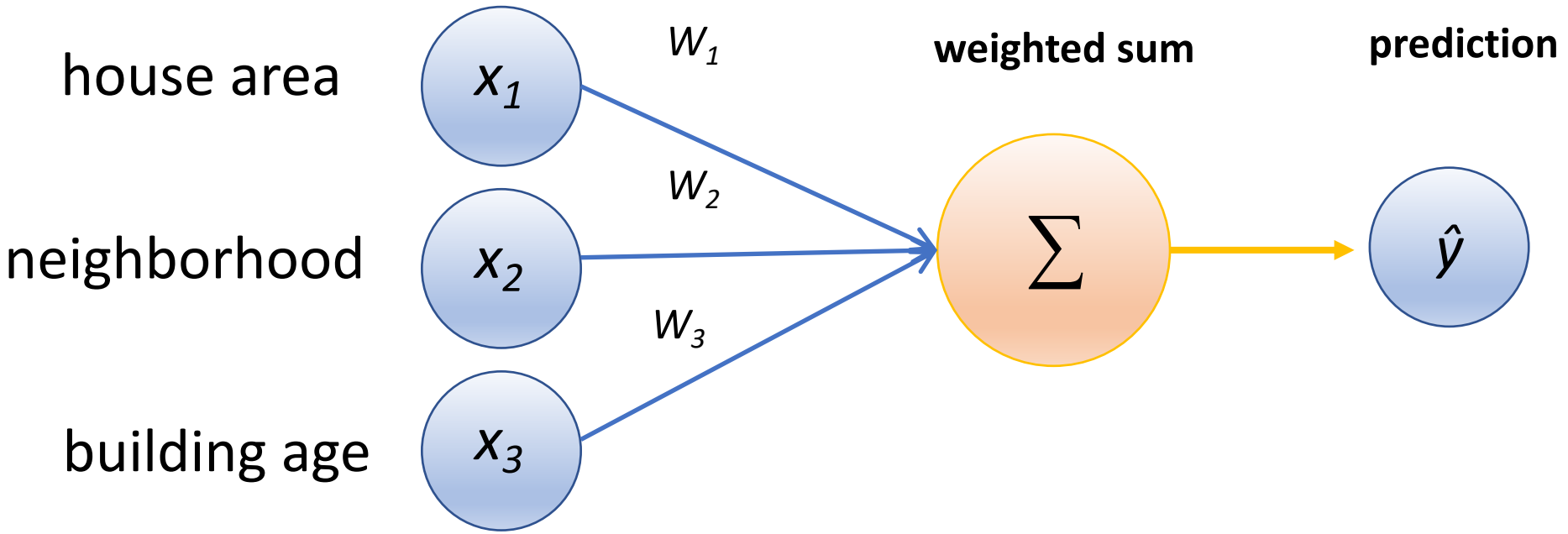
$$\longrightarrow \quad w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5 = \hat{y}$$

Historical Data

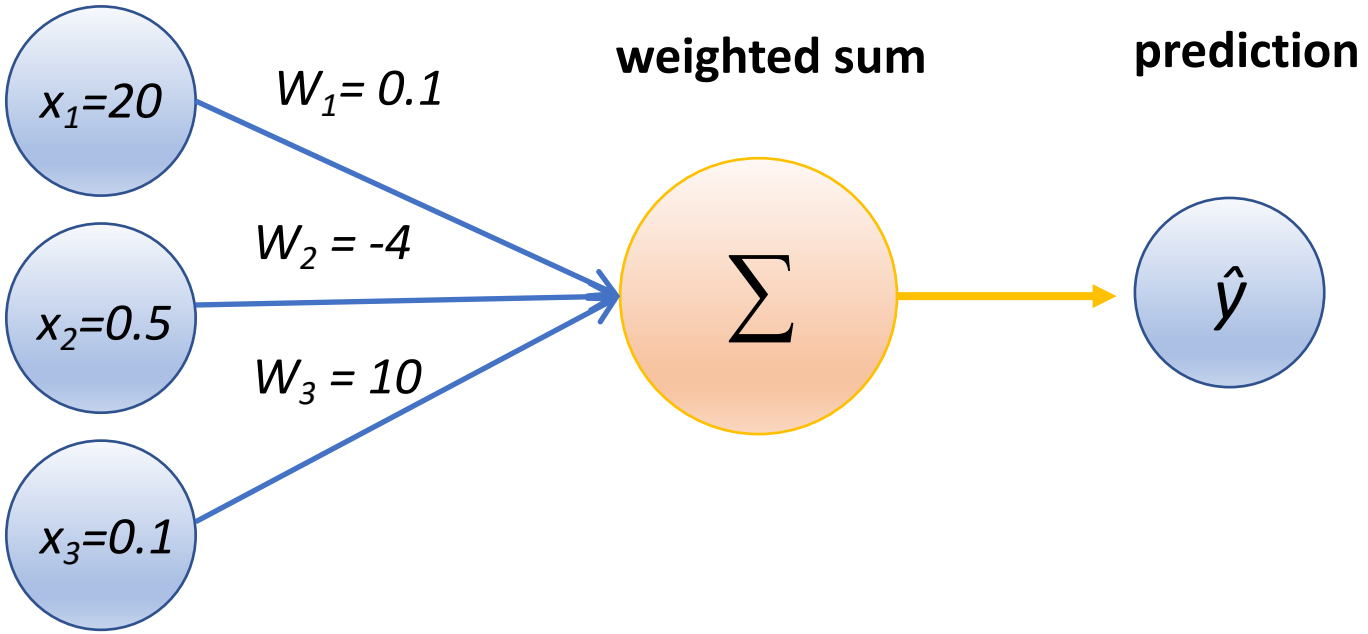


$$\hat{y} = wx + intercept$$

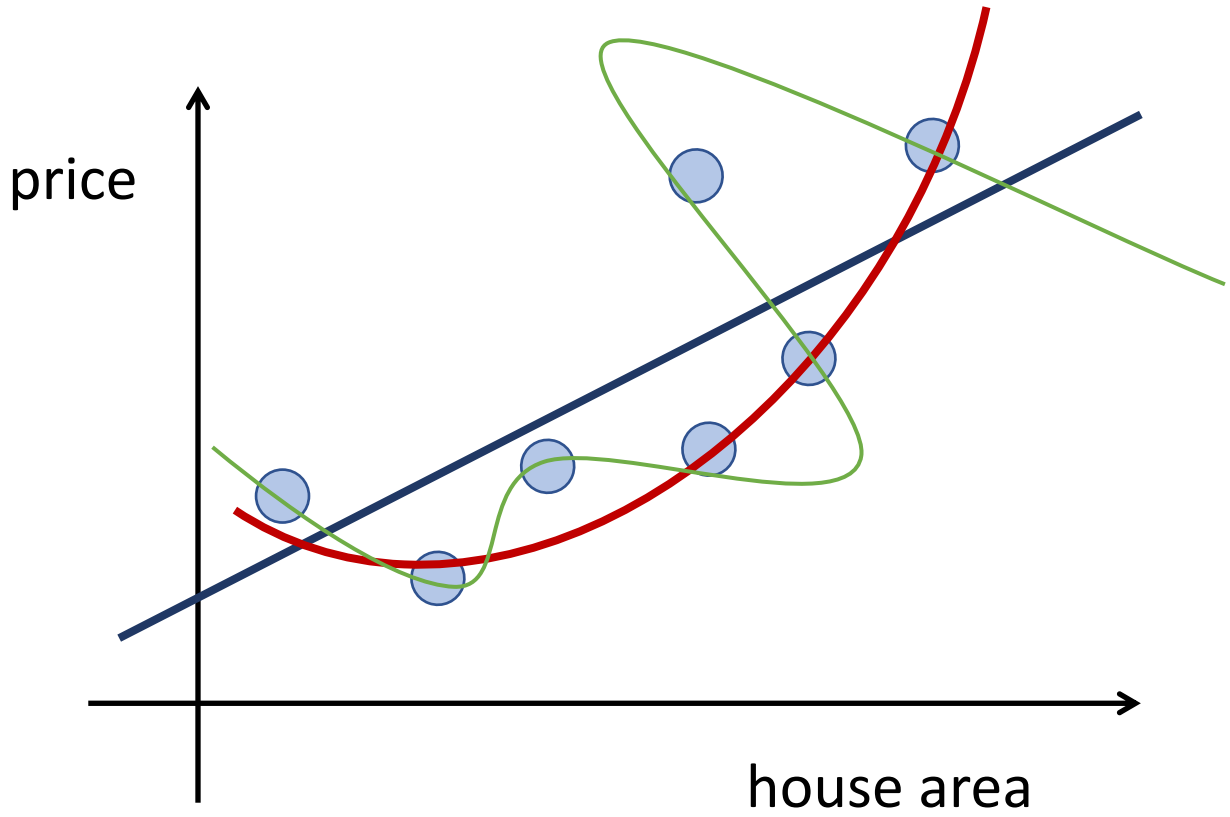
$$\hat{y} = w_1 x_1 + \cdots + w_n x_n$$



$$\hat{y} = 20 * 0.1 + 0.5 * (-4) + 0.1 * 10 = 1$$



Historical Data

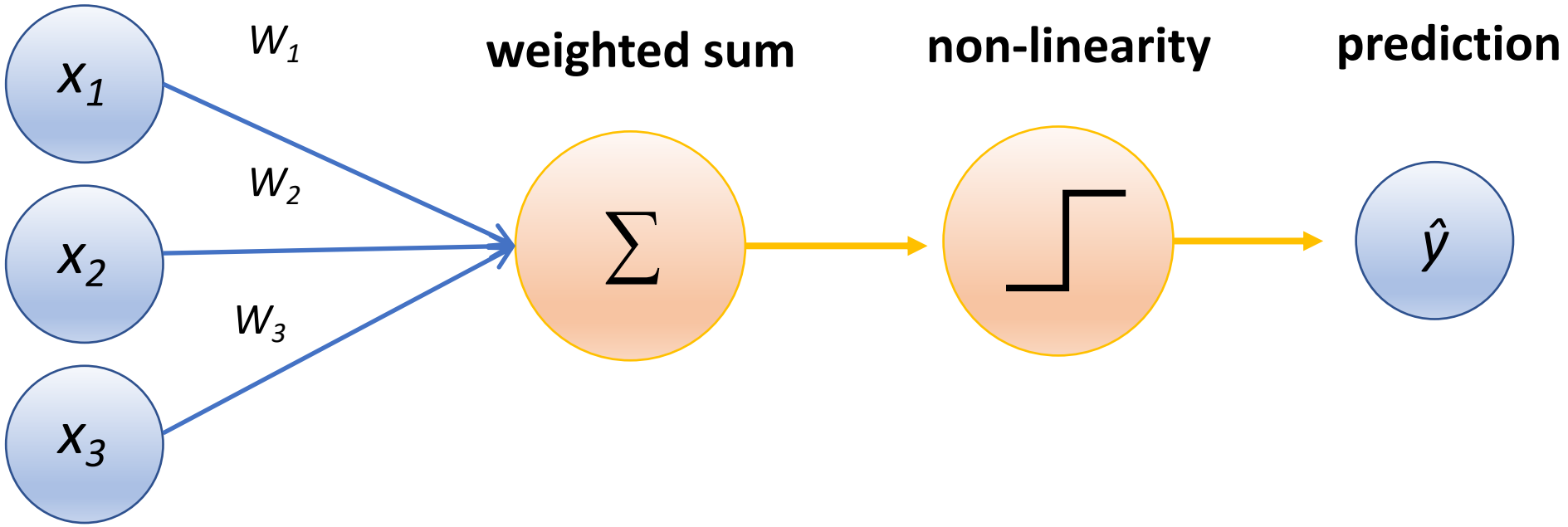


$$\hat{y} = wx$$

$$\hat{y} = w_1 x^2 + w_2 x + w_3$$

non-linear transformation

$$\hat{y} = \sigma(w_1 x_1 + \cdots + w_n x_n)$$



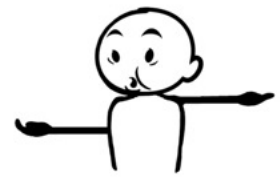
Sigmoid

$$y = \frac{1}{1+e^{-x}}$$

Tanh

$$y = \tanh(x)$$

Step Function



$$y = \begin{cases} 0, & x<n \\ 1, & x \geq n \end{cases}$$

Softplus

$$y = \ln(1+e^{x})$$

ReLU

$$y = \begin{cases} 0, & x<0 \\ x, & x \geq 0 \end{cases}$$

Softsign

$$y = \frac{x}{(1+|x|)}$$

ELU

$$y = \begin{cases} \alpha(e^{x}-1), & x<0 \\ x, & x \geq 0 \end{cases}$$

Log of Sigmoid

$$y = \ln\left(\frac{1}{1+e^{-x}}\right)$$

Swish

$$y = \frac{x}{1+e^{-x}}$$

Sinc

$$y = \frac{\sin(x)}{x}$$

Leaky ReLU

$$y = \max(0.1x, x)$$

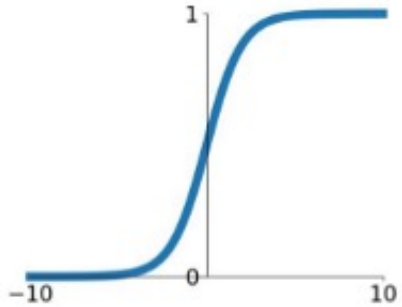
Mish

$$y = x(\tanh(\text{softplus}(x)))$$

Activation Functions

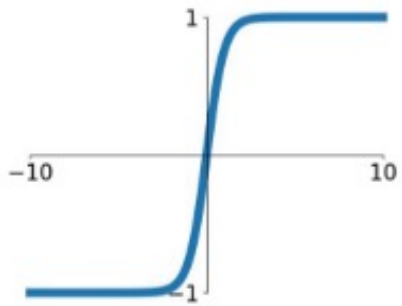
Sigmoid

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



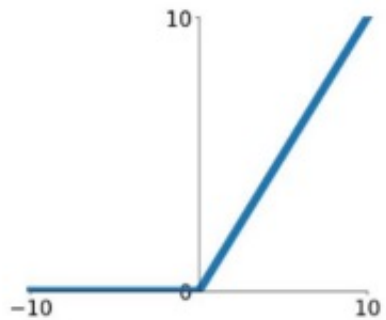
tanh

$$\tanh(x)$$



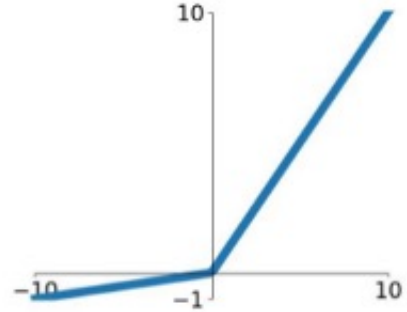
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

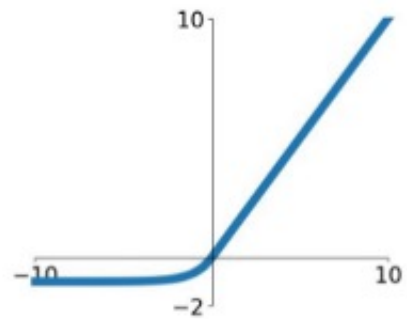


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

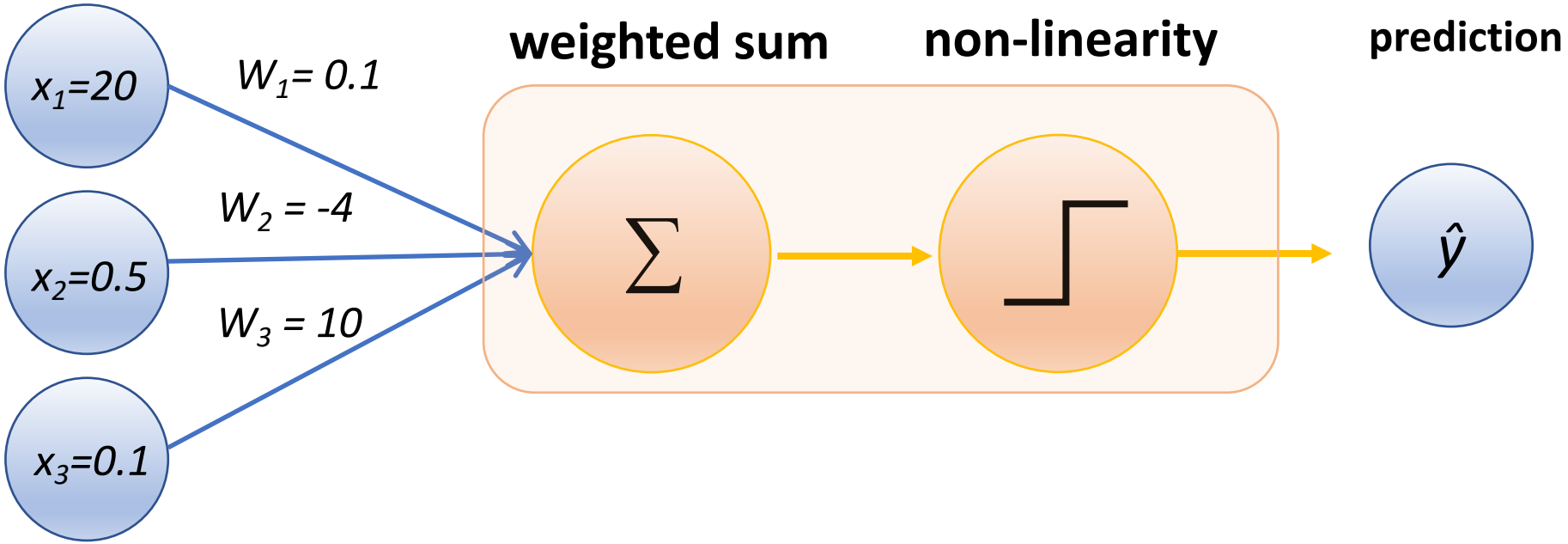
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

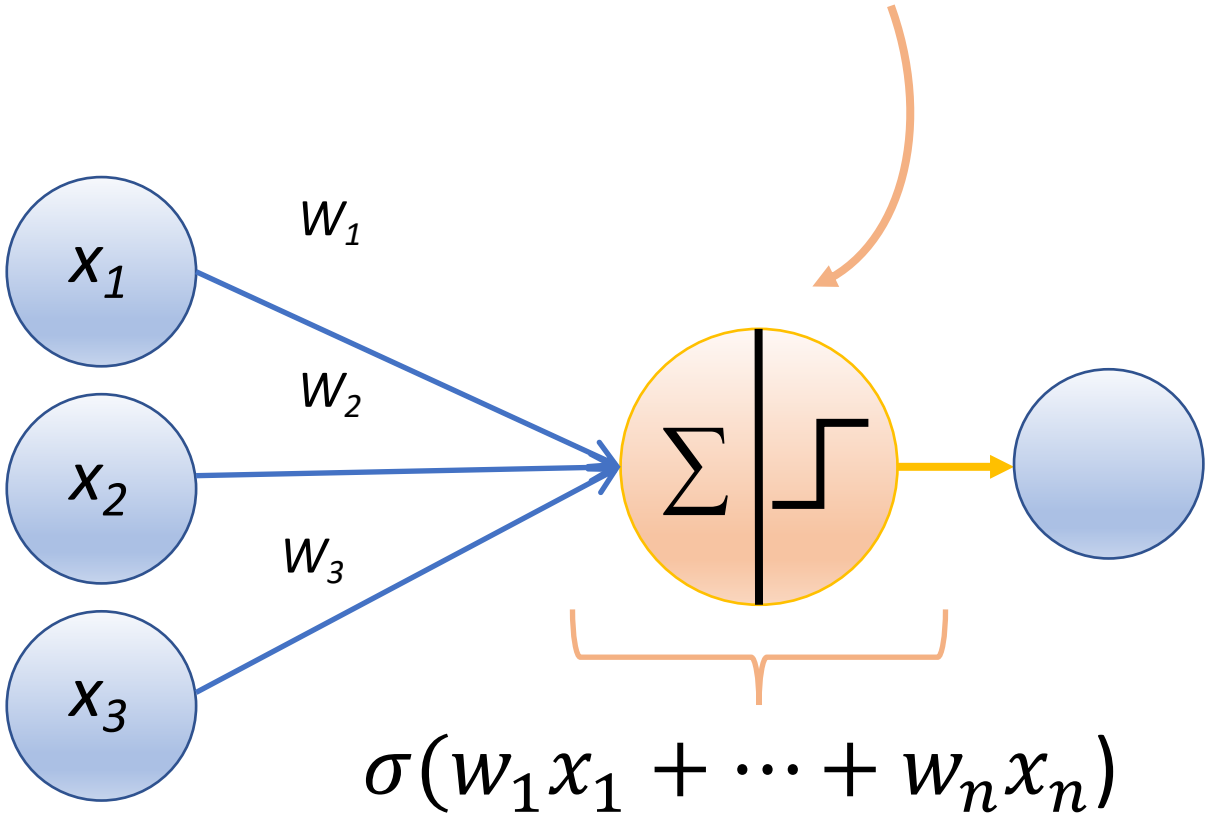


$$\hat{y} = \sigma(20 * 0.1 + 0.5 * (-4) + 0.1 * 10) \approx 0.73$$

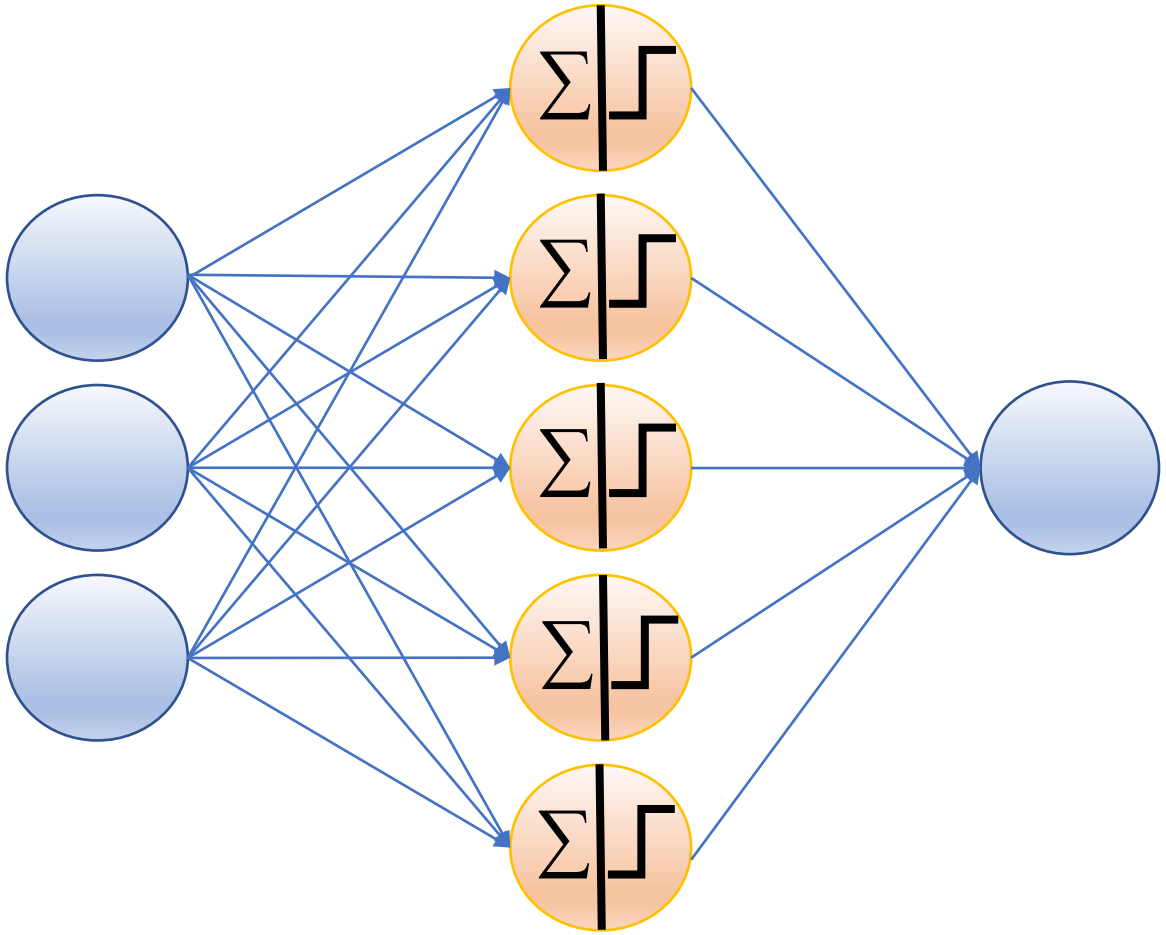
$$\sigma = \frac{1}{1 + e^{-x}}$$



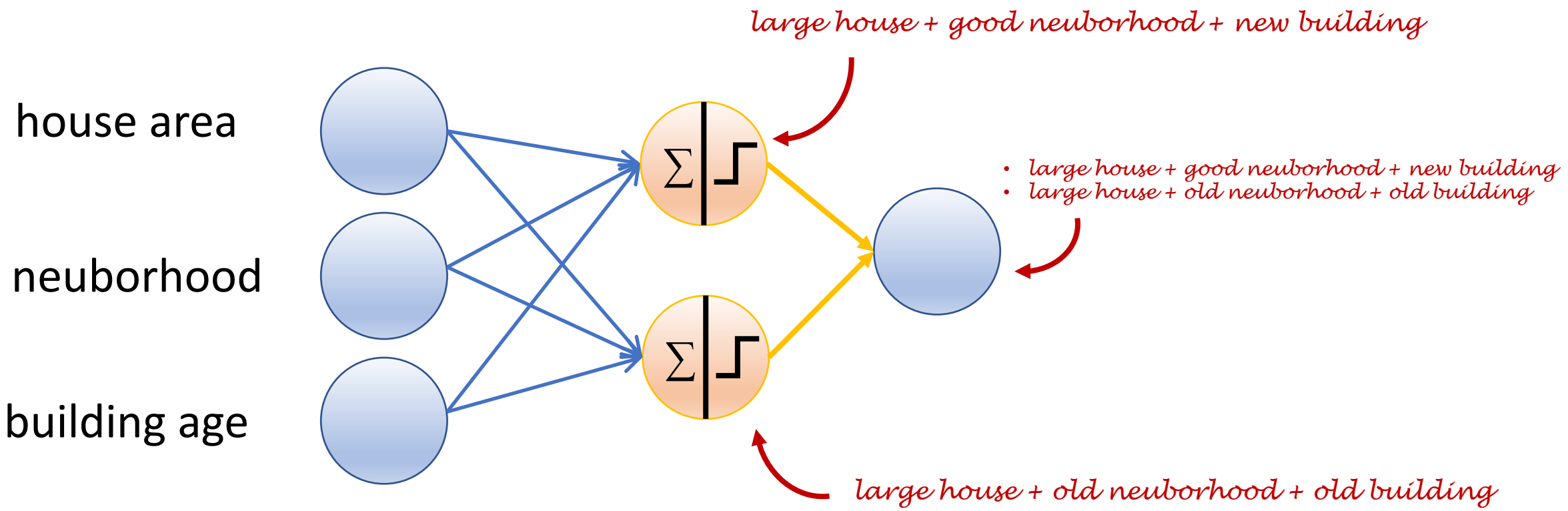
“Atom” of the ANN – artificial neuron or unit of ANN



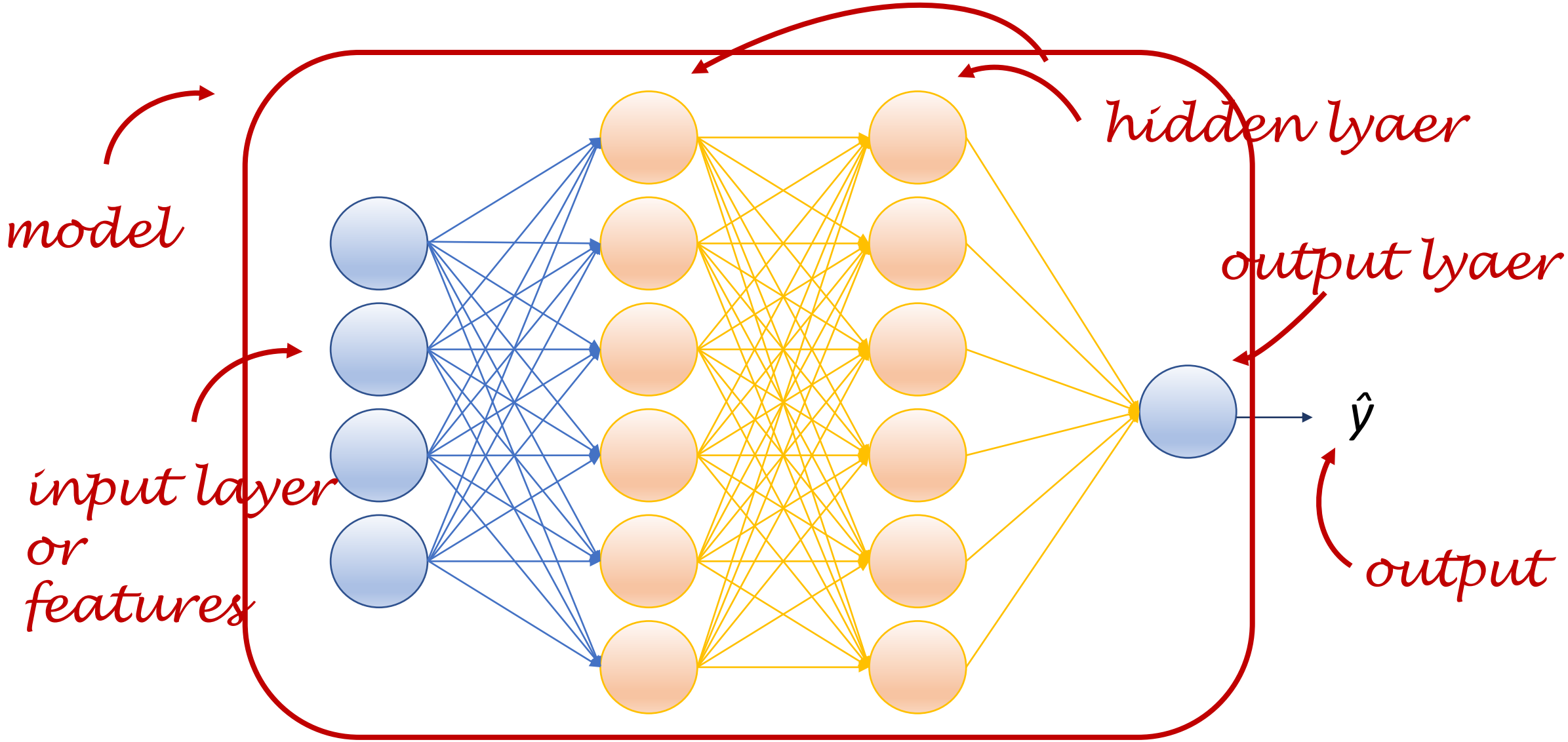
ANN – stacked elementary units



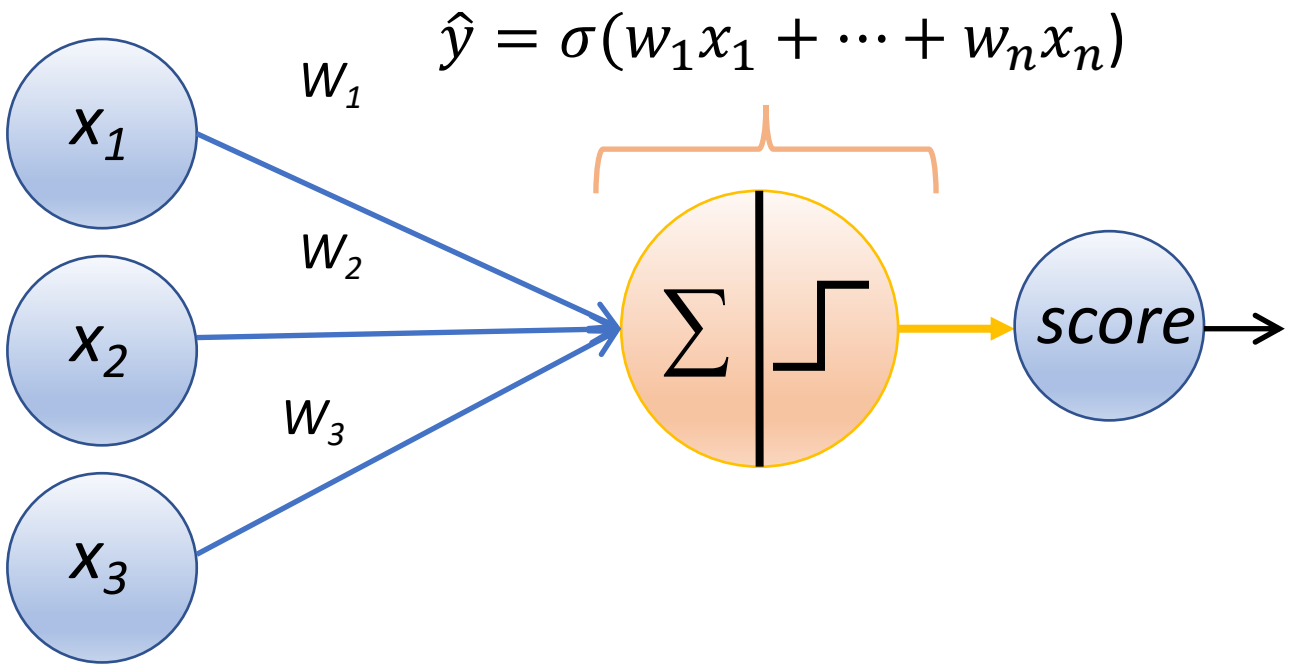
Why stacking neurons?



ANN – stacked elementary units

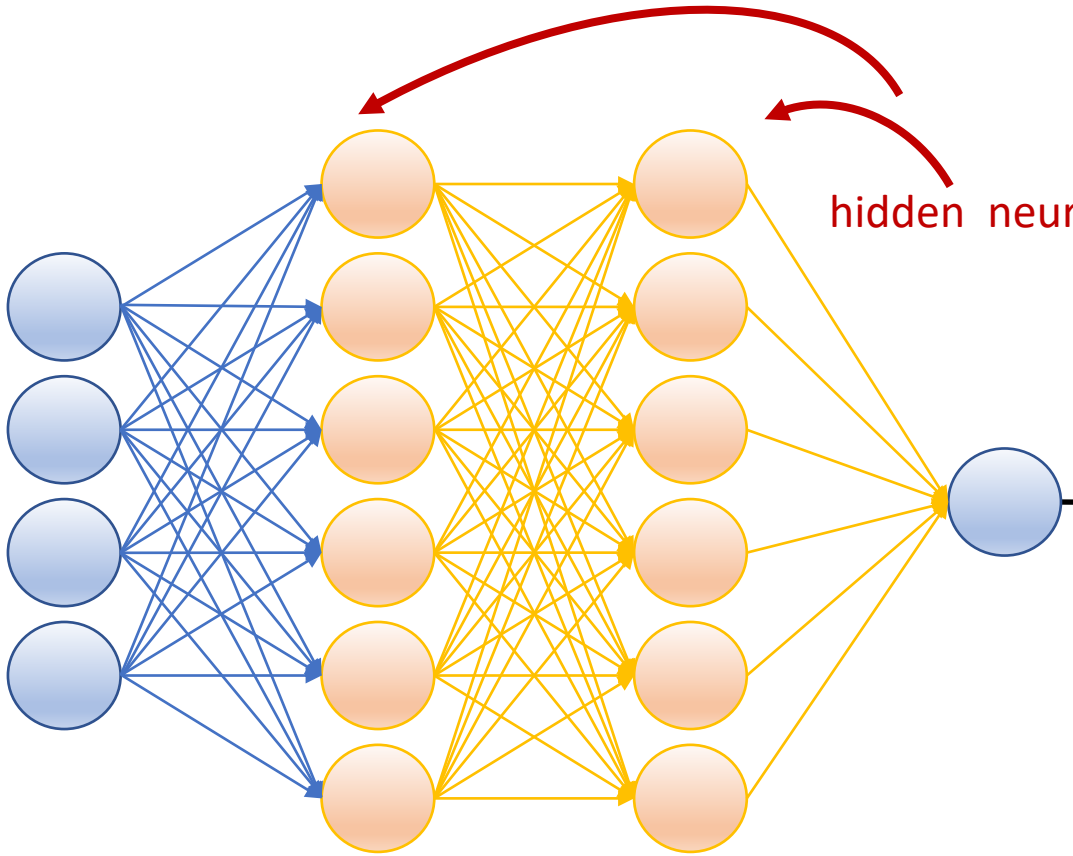


Activation functions



$$\begin{cases} \hat{y} = score, & if\ regression \\ \hat{y} = \sigma(score), & if\ classification \\ \sigma = \dfrac{1}{1 + e^{-x}} \end{cases}$$

Activation functions



hidden neurons with non-linear function (usually ReLU)

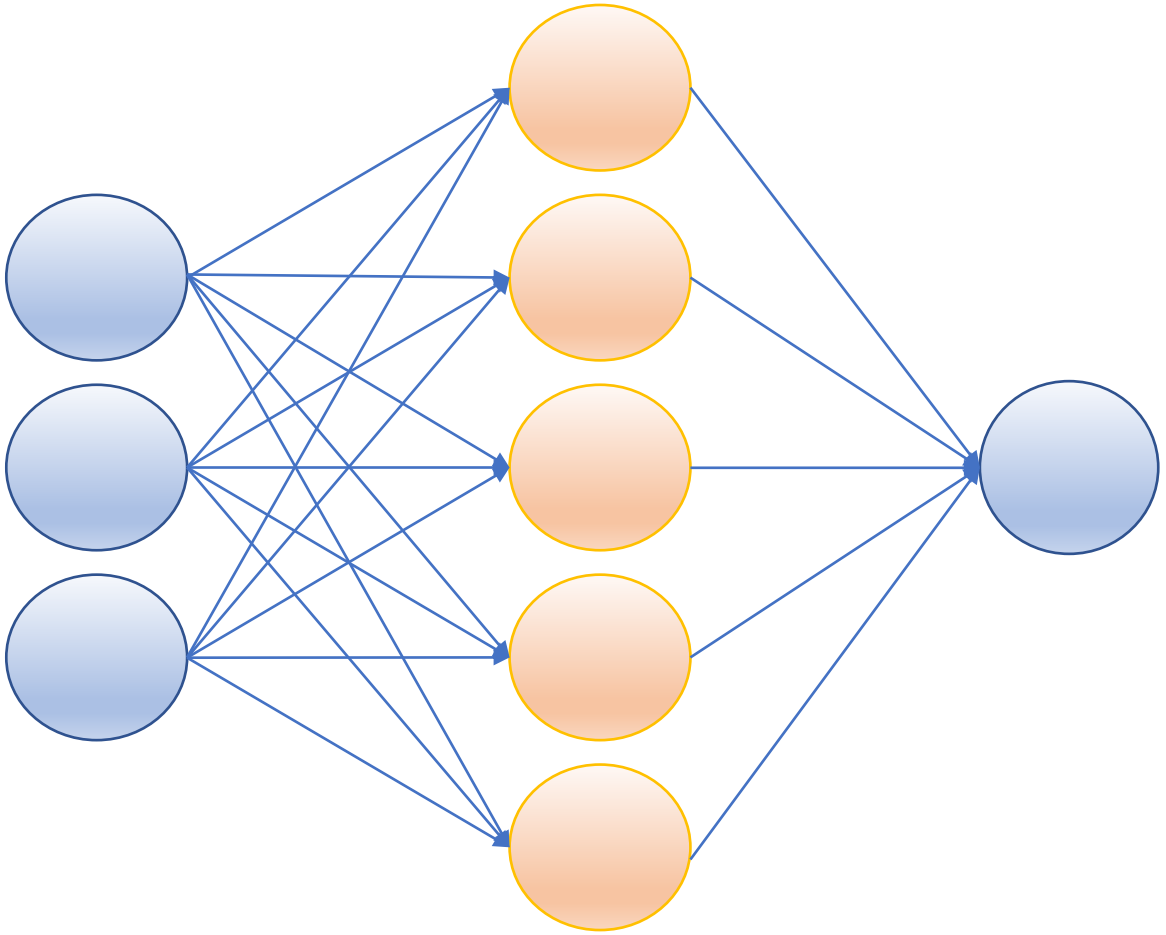
score

$$\hat{y} = score, \quad \text{if regression}$$

$$\hat{y} = \sigma(score), \text{ if classification}$$

ANN PLAYGROUND

How to find good model parameters?

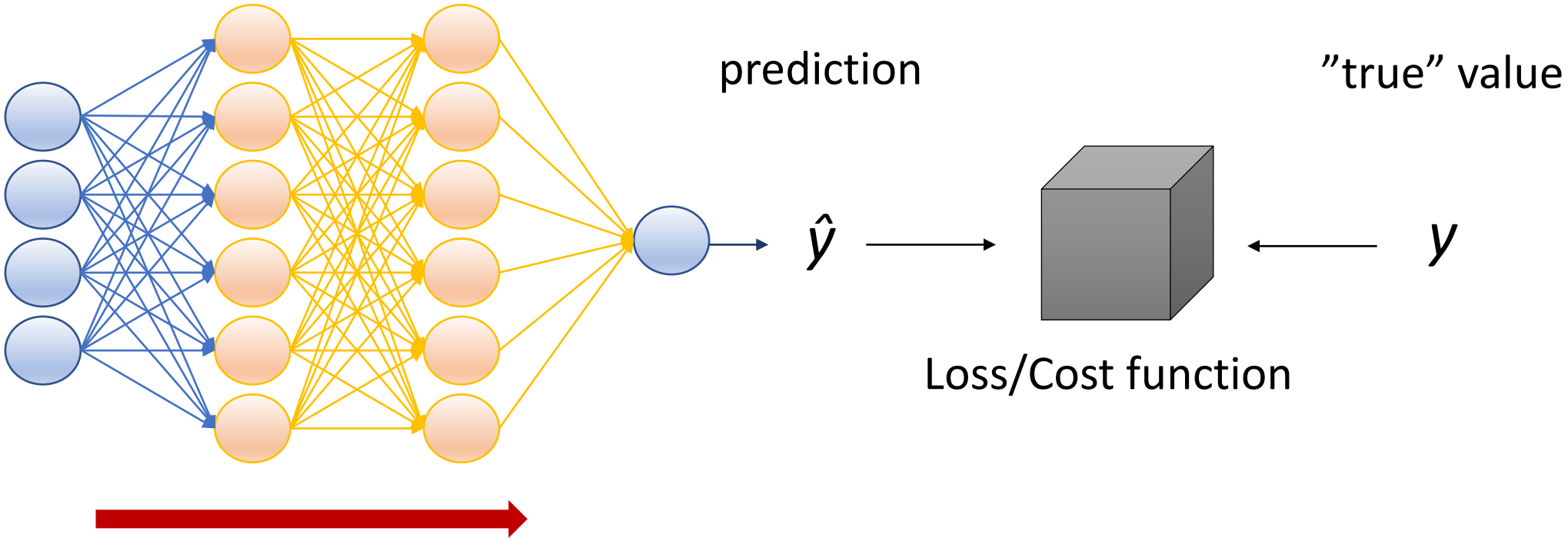


n.o. parameters = 3*5 + 5*1

$w_1, w_2, w_3, w_4, w_5, \ldots$???

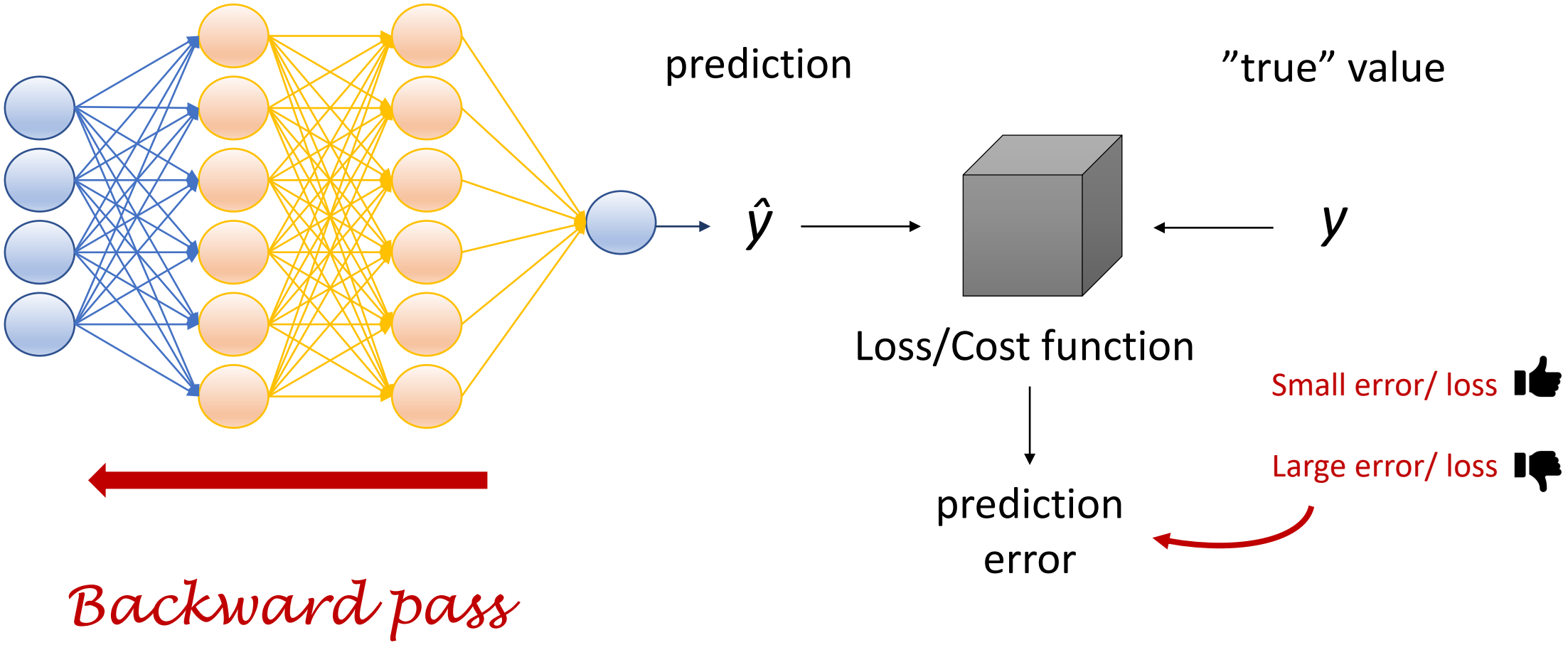
$\sigma(w_1 x_1 + \cdots + w_n x_n)$

Gradient Descent Algorithm

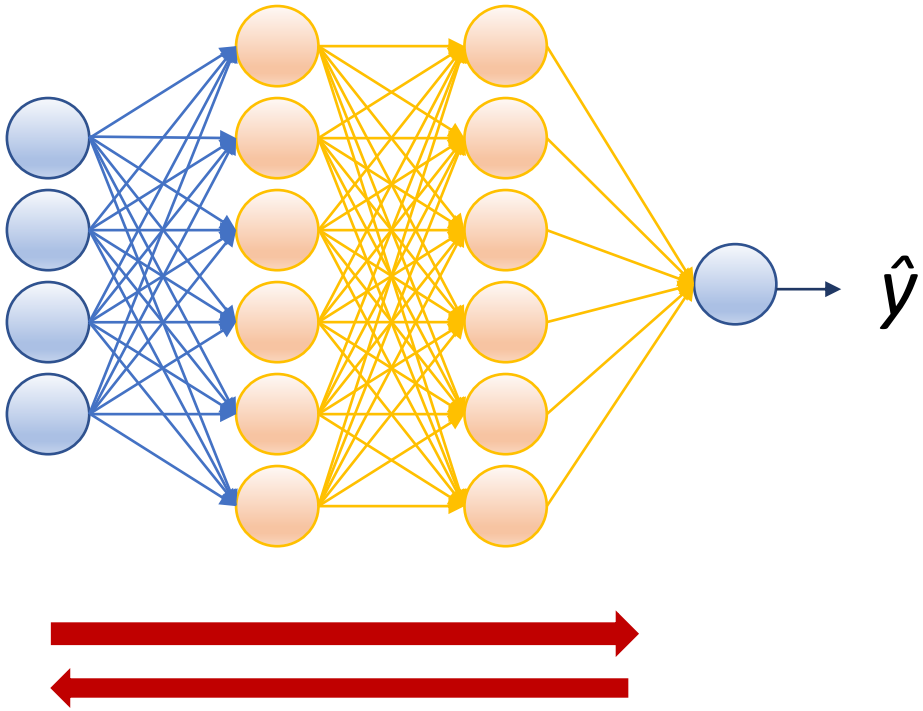


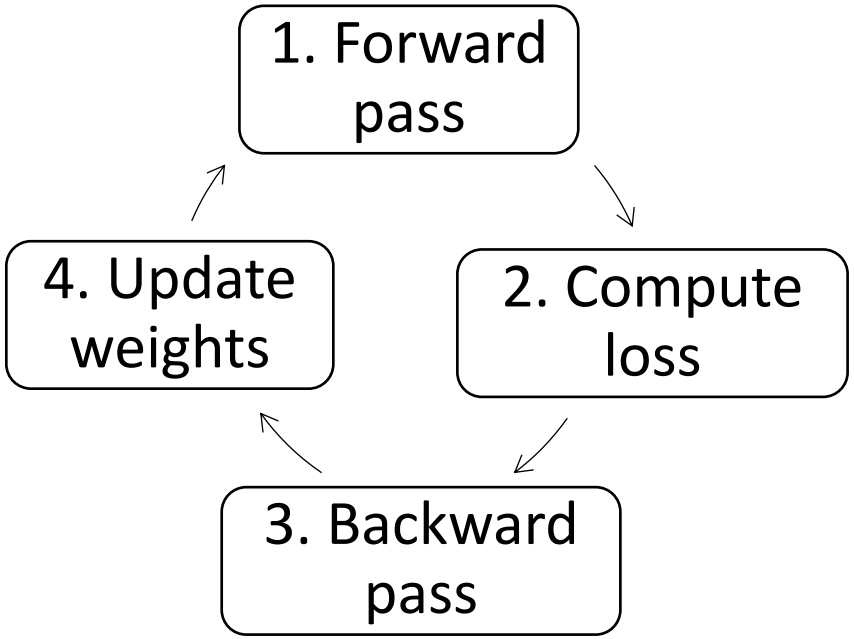
Forward pass

Gradient Descent Algorithm

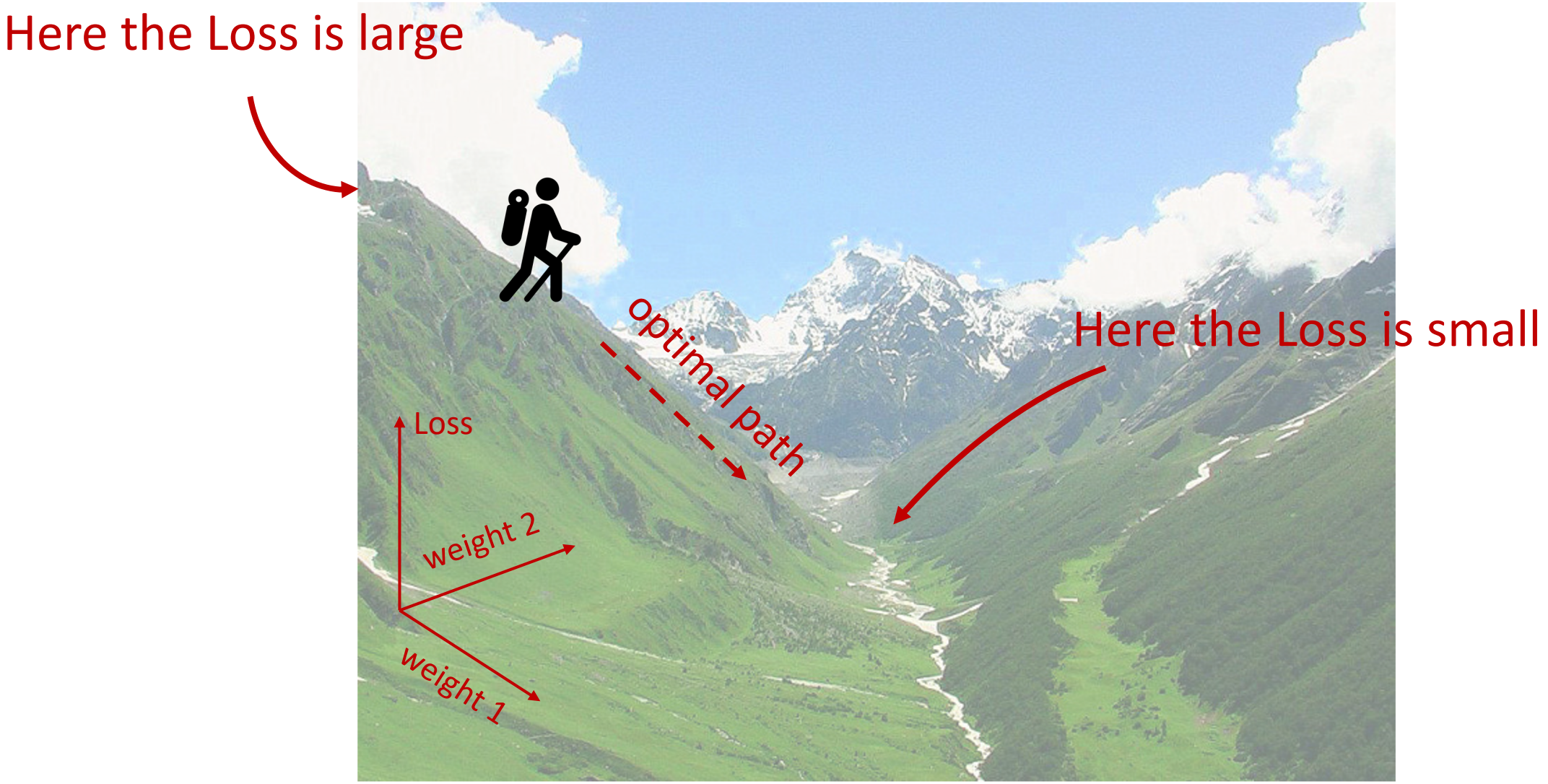


Gradient Descent Algorithm





Loss landscape hiker



Gradient Descent Algorithm

Most important hyper-parameters to set:

- Learning rate (step size) – by what amount we adjust/ tune/ update model's parameters (weights)

Learning rate

Large step size

Small step size

Gradient Descent Algorithm

Most important hyper-parameters to set:

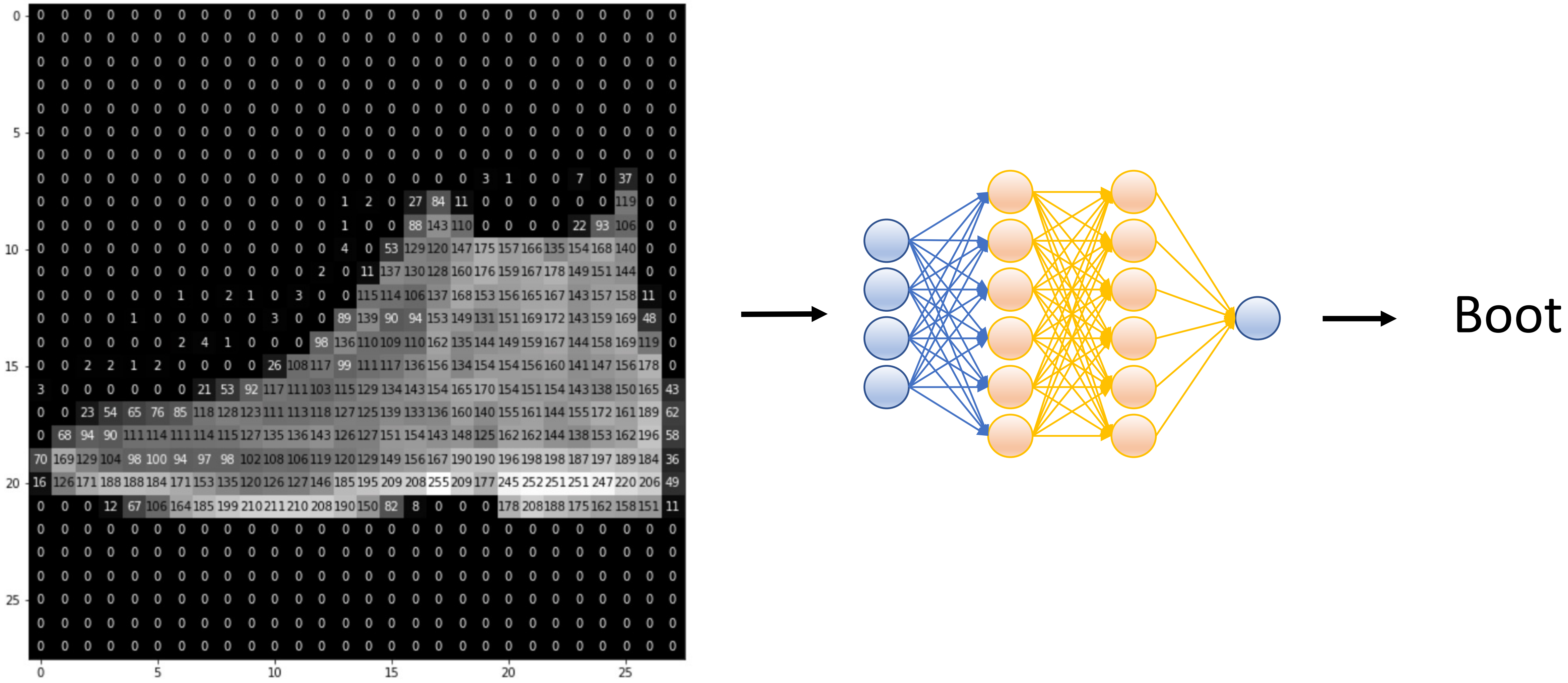
- Learning rate (step size) – by what amount we adjust/ tune/ update model's parameters (weights)
- Number of itearations (epochs) - how many times we update model's weight

Number of itearations (epochs)

N.o. epochs is large

N.o. epochs is smal

Hands on: shallow ANN for classification



Tools

Here we keep our code

Python libraries:

operations with arrays/ matrices

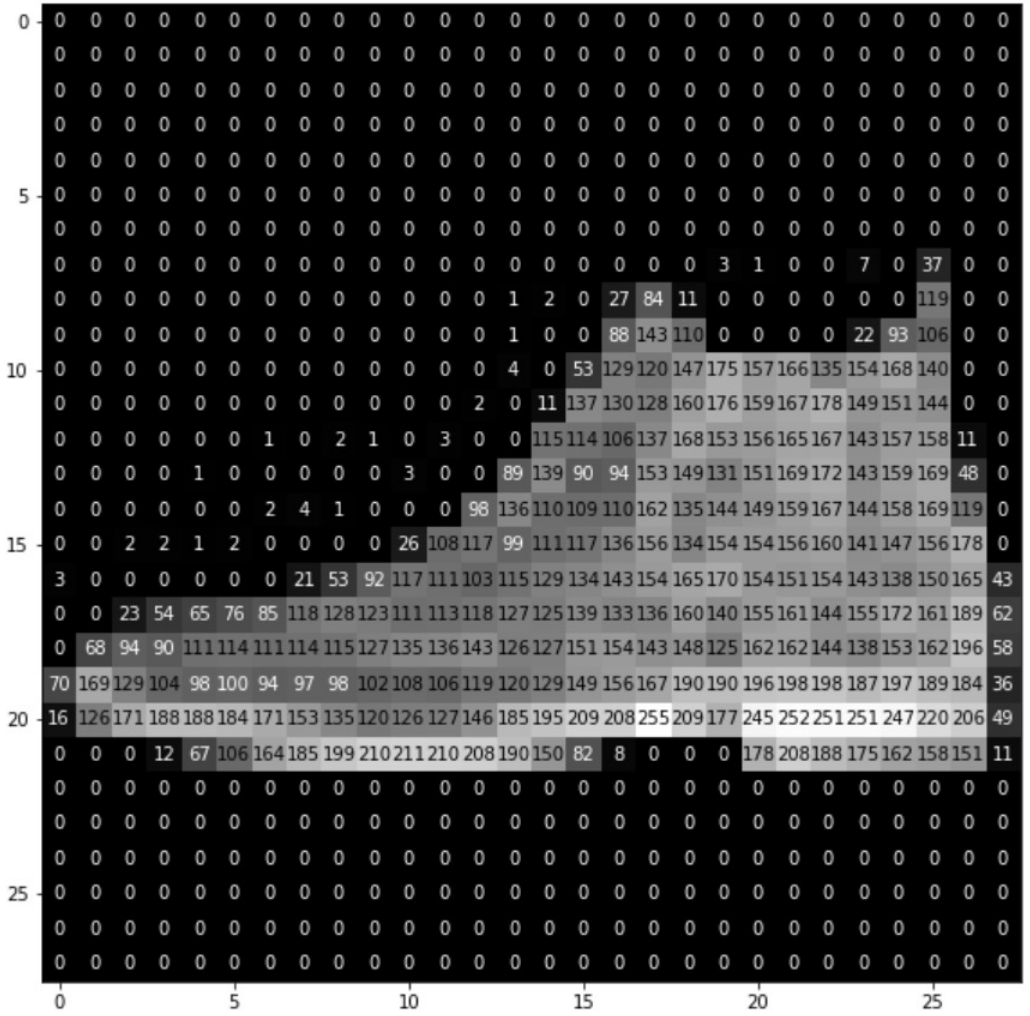
graphs

Here we run a code

deep learning

Data

- Data point: 28x28 grayscale image of shopping item
- Features: 28x28=784 pixel values (range 0-255)
- Labels: Product category (boot, shirt, hat, …). 10 classes in total.



Typical Workflow

1. Data preprossing (scale 0-1), splitting (training/validation/test).
2. Define ANN model
3. Choose Loss function, Optimizer
4. Train ANN
5. Evaluate on unseen (test) dataset

Data preprossing

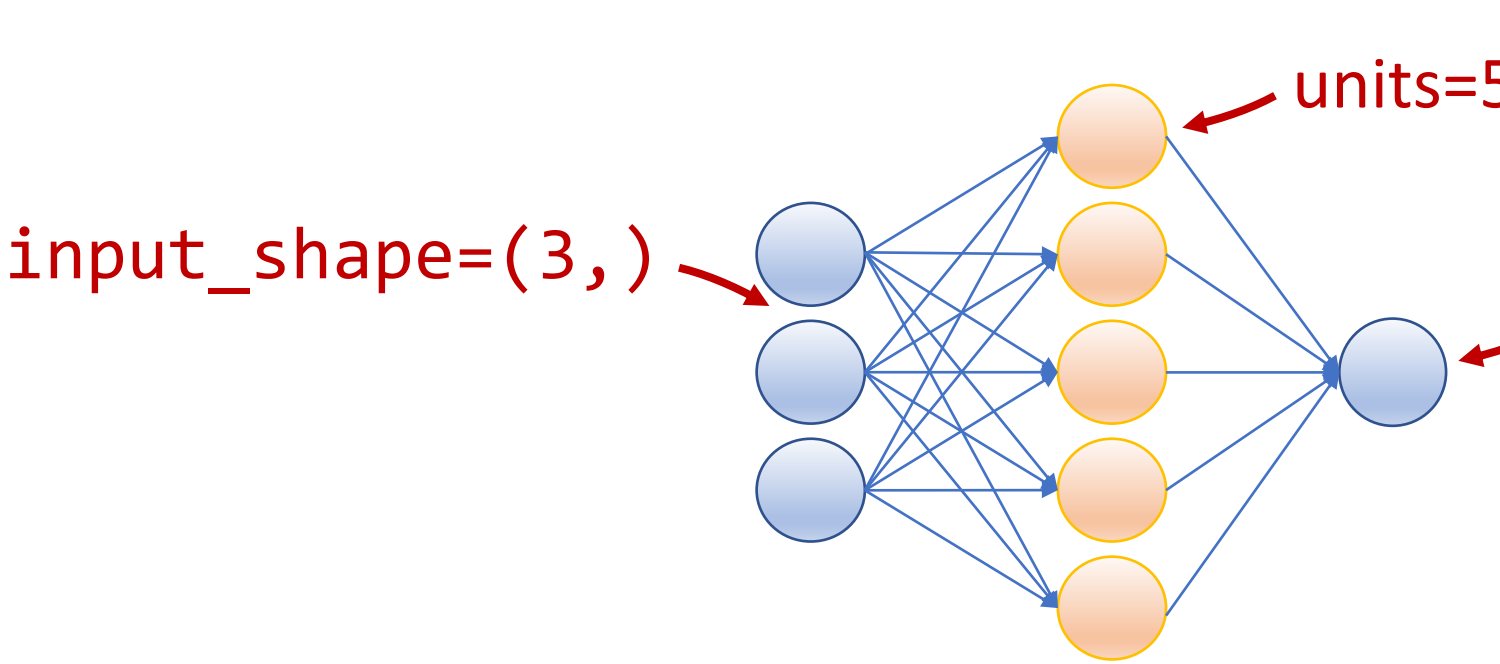
1. Scaling features to 0-1: makes it easier for algoritm to find “good” model parameters

2. Splitting (training/validation/test datasets):

    - Training – to find “good” model parameters
    - Validation – to choose model hyper-parameters (learning rate, n.o. epochs)
    - Test – final performance evaluation

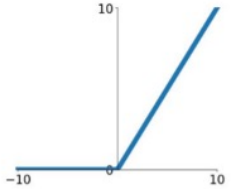
Define ANN model

```
model = keras.Sequential([
        layers.Dense(units=5, activation='relu',input_shape=(3,)),
        layers.Dense(units= 1, activation='sigmoid')
        ])
```



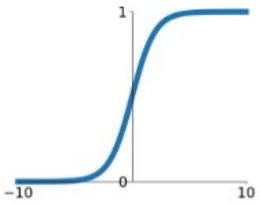
ReLU

$\max(0, x)$



Sigmoid

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



Choose Loss function, Optimizer

Variation of basic Gradient Descent Algorithm

```
model.compile(optimizer='RMSprop',
              loss='sparse_categorical_crossentropy',
              metrics=['sparse_categorical_accuracy'])
```

Compare predictions and labels

Easy to understand performance measure

Train ANN

Features and labels

```
history = model.fit(X_trainval,
                    y_trainval,
                    validation_split=0.2,
                    batch_size=32,
                    epochs=20,
                    verbose=1)
```

Training data = 80%
Validation = 20%

Feed data by batches of 32

Printing info during training

Evaluate on test dataset

```
test_loss, test_accuracy = model.evaluate(X_test, y_test, verbose=0)
print('Accuracy on test dataset:', test_accuracy)
```