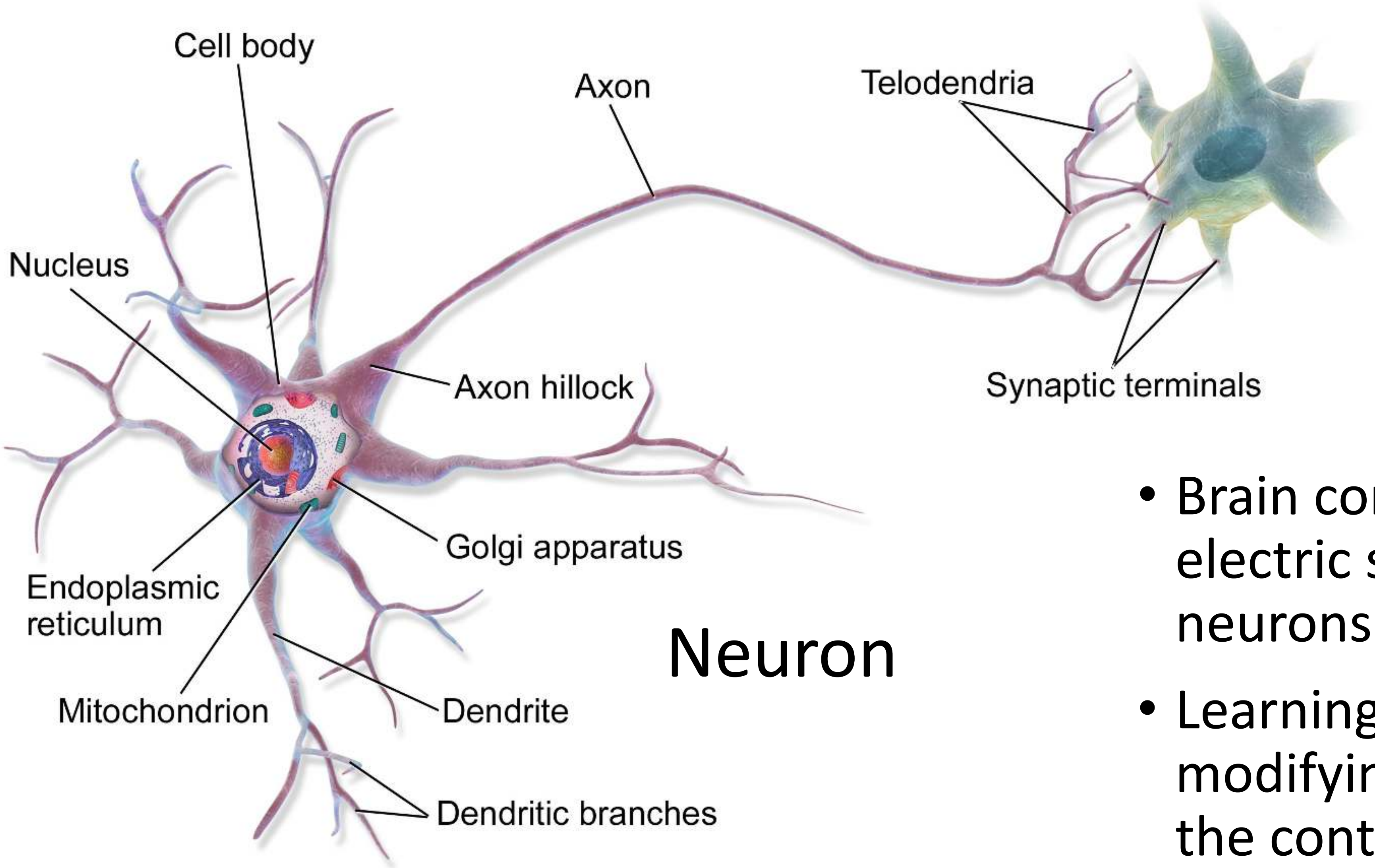
Neural Networks 1

Anders Krogh
Center for Health Data Science
and Department of Computer Science
University of Copenhagen

HeaDS.ku.dk

Inspiration from the brain



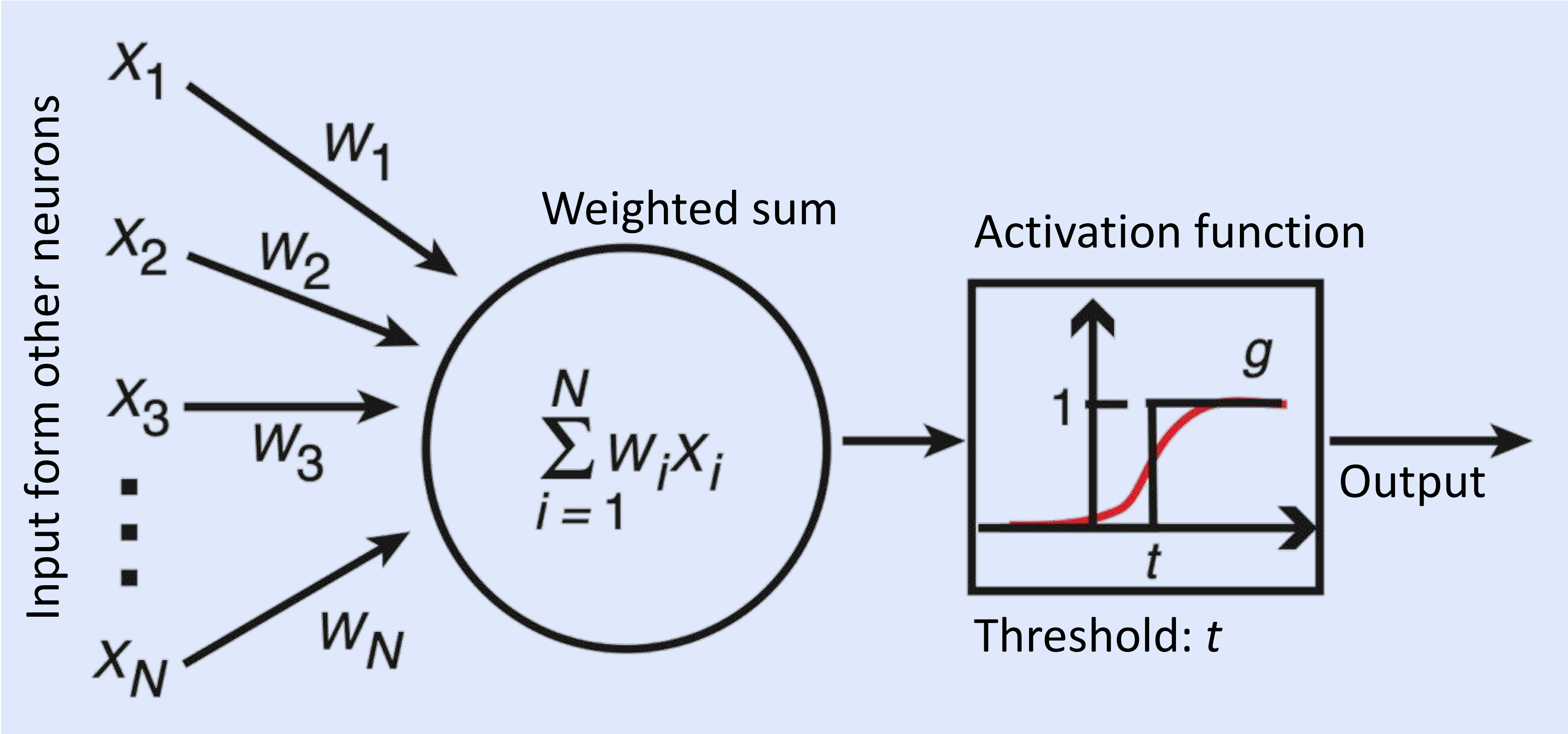
Neuron

- Brain computes by sending electric signals between neurons
- Learning happens by modifying the strengths of the contacts – the synapses

A mathematical model of the neuron

McCulloch and Pitts
proposed this model in 1943

This is the basis for most artificial neural network models

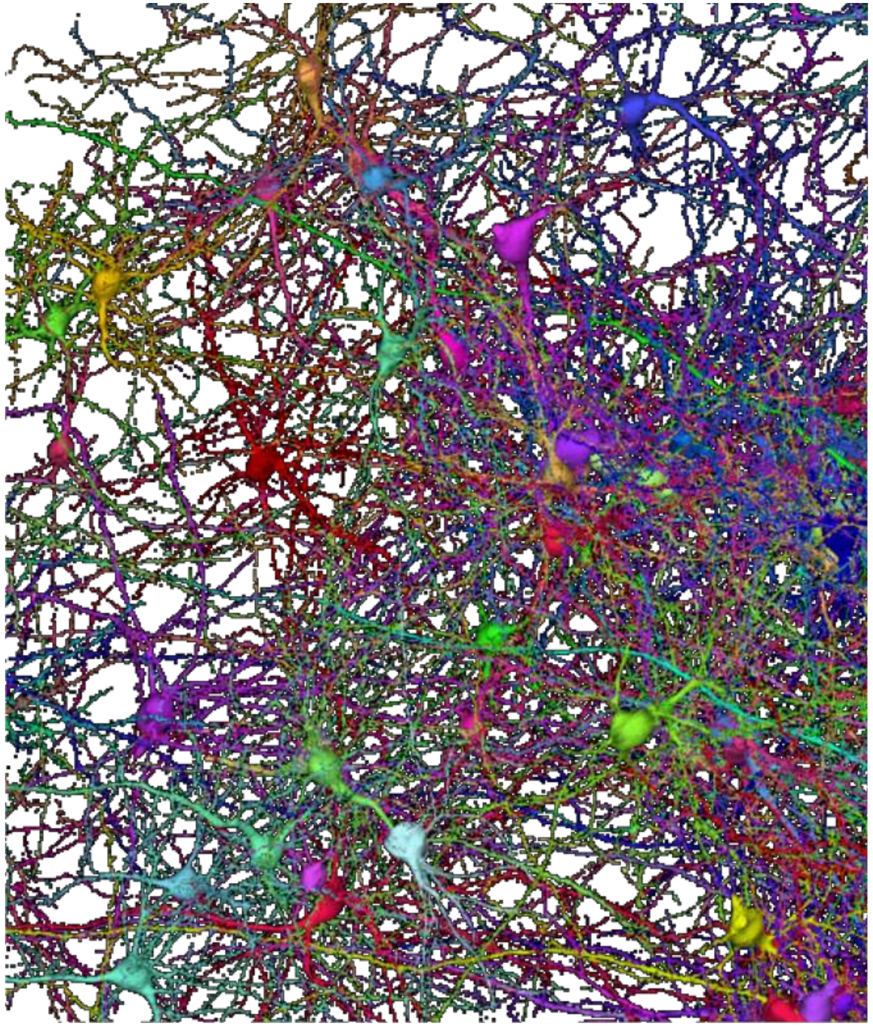


Sigmoid activation function (red curve)

$$g(h) = \frac{e^{h-t}}{1 + e^{h-t}}$$

Figure from A Krogh: What are artificial neural networks? Nat. Biotech. 26, p. 195, 2008

Many connected neurons ➜ neural network



Screenshot from https://h01-release.storage.googleapis.com/gallery.html

Artificial neural network

Feed-forward NN

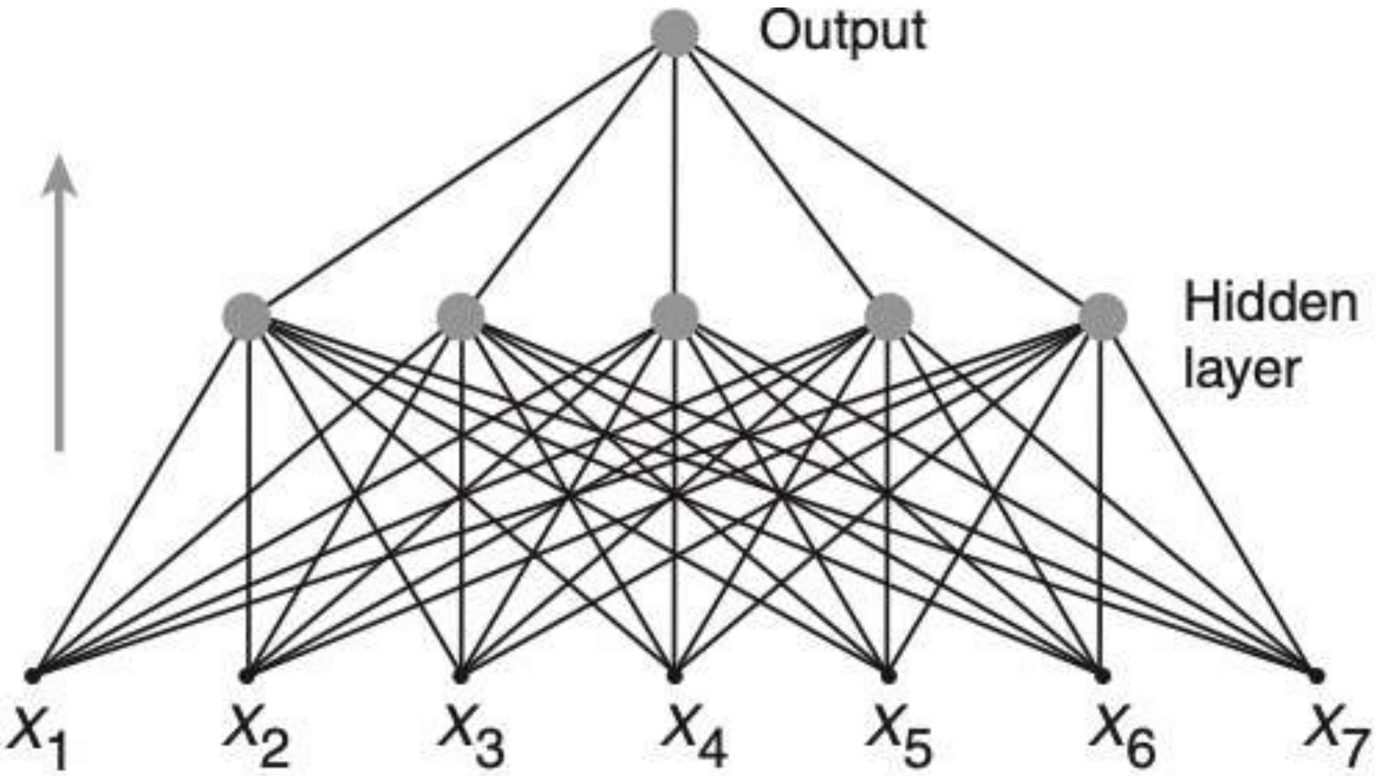
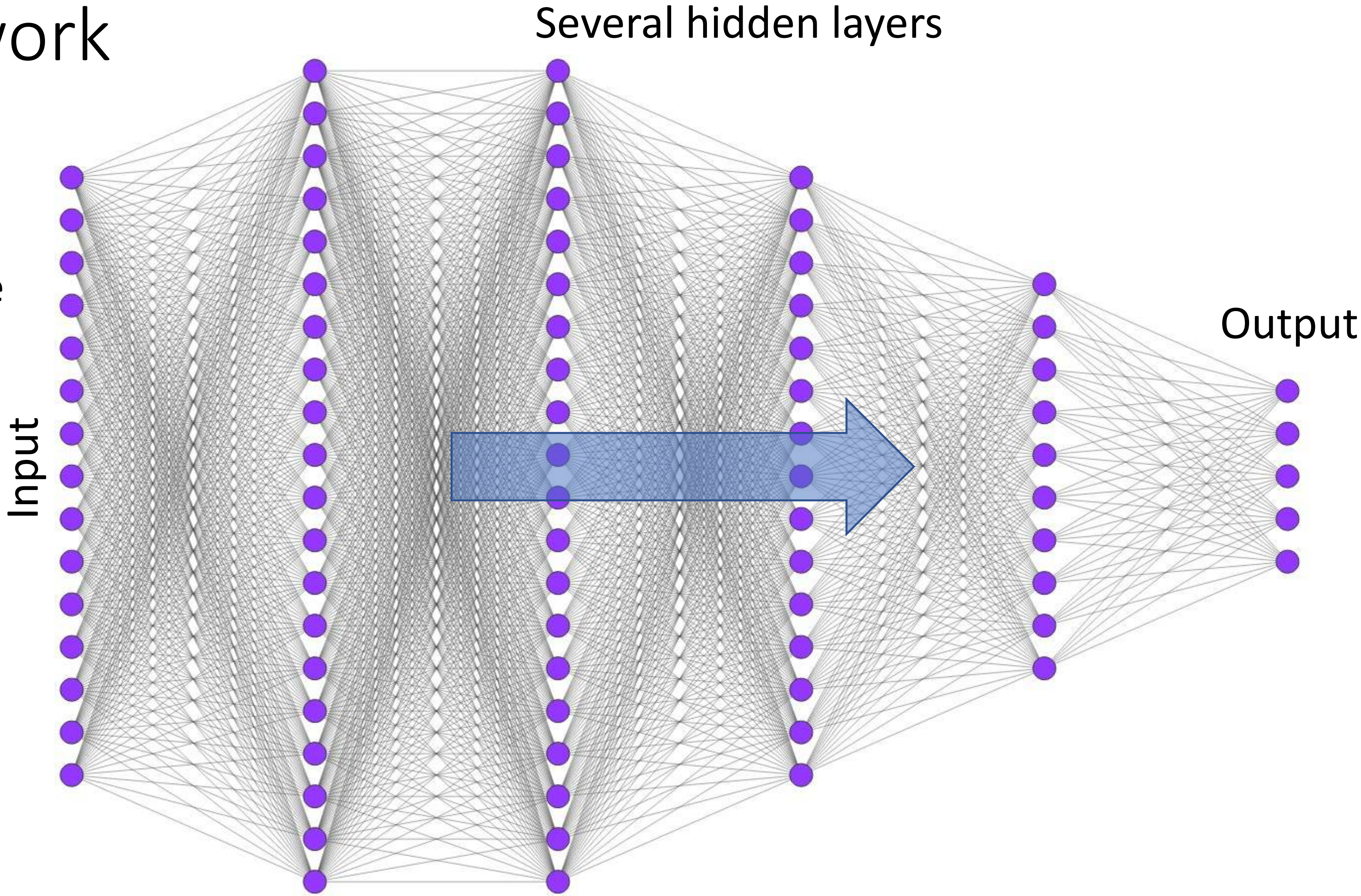


Figure from A Krogh: What are artificial neural networks? Nat. Biotech. 26, p. 195, 2008

Deep network

By adding layers of units, the capabilities of the network is increased

Input and output dimensions are usually defined by the problem

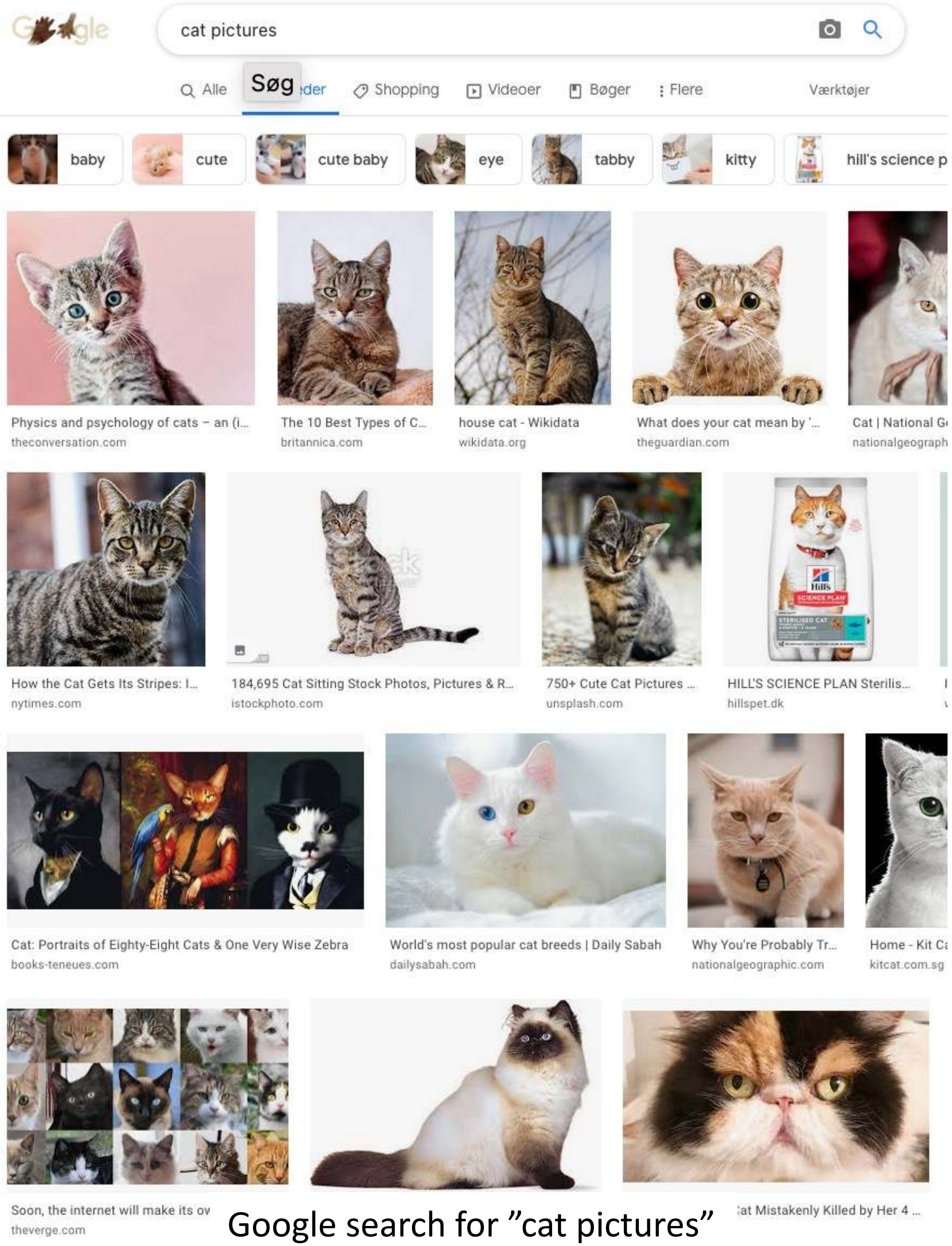


Neural net drawing made at http://alexlenail.me/NN-SVG/index.html

Learning from examples

- Humans learn from examples
- By seeing enough pictures of cats, a child can learn to recognize a cat

- Artificial neural networks also learn from examples



Google search for "cat pictures"

Learning by minimizing the error

- Neural network is a function $f_w(\boldsymbol{x})$
- $\boldsymbol{x}$ is an input vector (e.g. pixel values in cat picture)
- Output: values between zero (no cat) and one (cat in picture)
- Parametrized by the weights *w* (*w* is symbolizing all the weights)

Train on a set of **labeled examples** (training set)

Input x:

Labels: t=1 (cat) t=0 (not cat)

Learning: Find the weights that give the desired output as close as possible

Minimize the error:

$$E(w) = \sum_i (f_w(\boldsymbol{x}_i) - t_i)^2$$

Sum is over training examples

Learning by minimizing the error

- Error is often called **loss** or **cost**
- Labels are also called **targets**
- An input vector is also called a **feature vector**

Train on a set of **labeled examples** (training set)

Input x:

Labels: t=1 (cat) t=0 (not cat)

Learning: Find the weights that give the desired output as close as possible

Minimize the error:

$$E(w) = \sum_i (f_w(\boldsymbol{x}_i) - t_i)^2$$

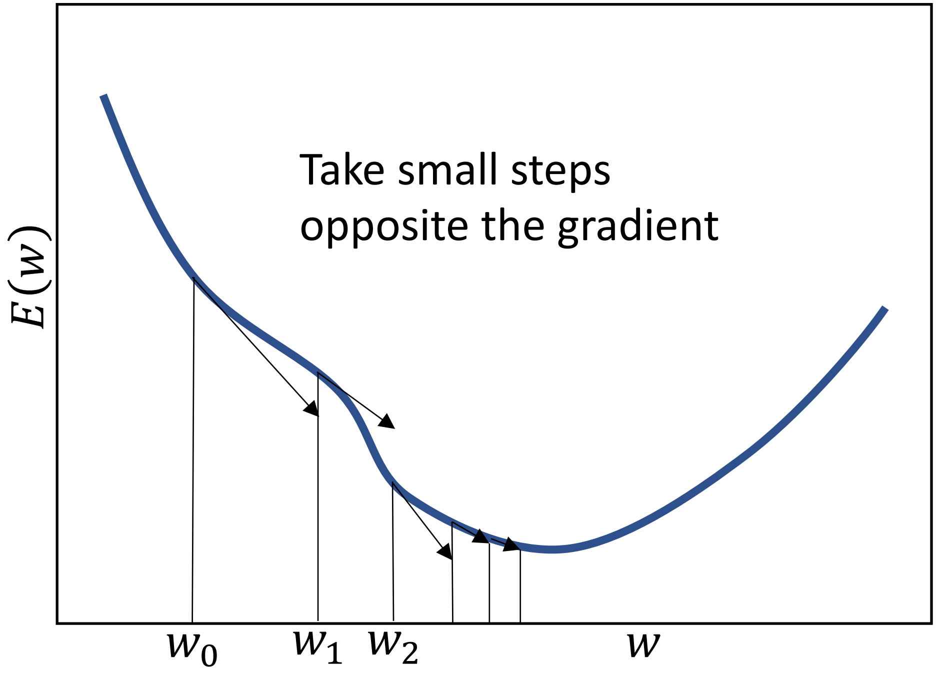
Sum is over training examples

Minimize the error by gradient descent

- A general method for function minimization
- Iterative procedure: take a small step in the direction opposite to the gradient in each iteration
- The gradient:
  - The partial derivative for each weight in the network
  - A vector that points in the direction of fastest growth of the function
- Uses a step size or learning rate ε

$$w_{i+1} = w_i - \varepsilon \frac{\partial E(w)}{\partial w}$$

One-dimensional example



Gradient descent leads to the famous Back-propagation algorithm for neural networks

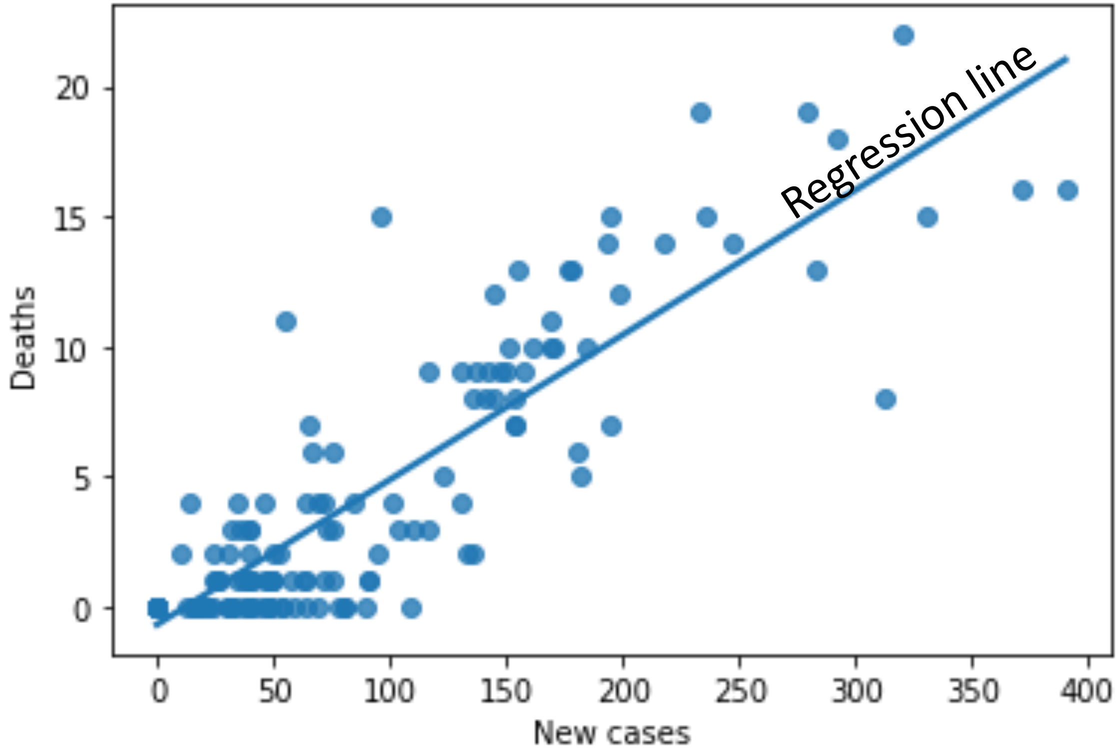
Fortunately we do not have to derive the math – modern neural network programs automatically calculate the gradients

Analogy to linear regression

Linear regression is also done by minimizing the squared error

- The error is the squared difference between the observed y for a given x and the "prediction" $y = ax + b$
- The a and b is found by minimizing the sum of the squared errors
- This can be done analytically leading to the formulas for linear regression
- It can also be done by gradient descent

Points show the number of deaths vs new Corona cases in Denmark per day from March to July 2020



There are many examples/animations of this online, e.g.:
https://towardsdatascience.com/gradient-descent-animation-1-simple-linear-regression-e49315b24672

Python example: Linear regression by gradient descent

- Open notebook in Github
- In this example, the gradient is calculated manually
- We do not have to do that again – pytorch will take care of it

More realistic networks

- The regression example has two “weights” *a* and *b*
- Normally the neural networks have thousands of weights and thresholds (some even millions)
- In the cat example there was a single output unit, but often we have many output units
- We often discriminate between networks for
  - Classification with binary targets (like cat/no cat)
  - Regression with continuous target values (like linear regression)
- Learning follows the same principles, but the error (loss) function may change

PyTorch

pytorch.org

- Pytorch is a Python package for neural networks
- It makes it easy to design and train neural networks, because of
  - Automated differentiation to calculate gradients
  - Efficient use of hardware (including GPUs)
- It uses tensors, which are multidimensional numerical arrays with many convenient mathematical operations (as in linear algebra)
- To start with pytorch, you need not worry about tensors

LET US TRY!