PD Dr. Jutta Rogal, Prof. Dr. Cecilia Clementi

Freie Universität Berlin

Machine Learning for Molecular Physics

WiSe 2021/22

L02 - General Introduction to Machine Learning

Image/Slide Credits: Miles Stoudenmire, David Wolfe Corne, Geoffrey A. Hinton, Frank Noé, Cecilia Clementi

Poll: Machine Learning

Freie Universität Berlin

1. Have you heard about ‘supervised learning’?
   - Yes
   - No

2. Have you heard about ‘unsupervised learning’?
   - Yes
   - No

3. Have you programmed any type of ML algorithm?
   - Yes
   - No

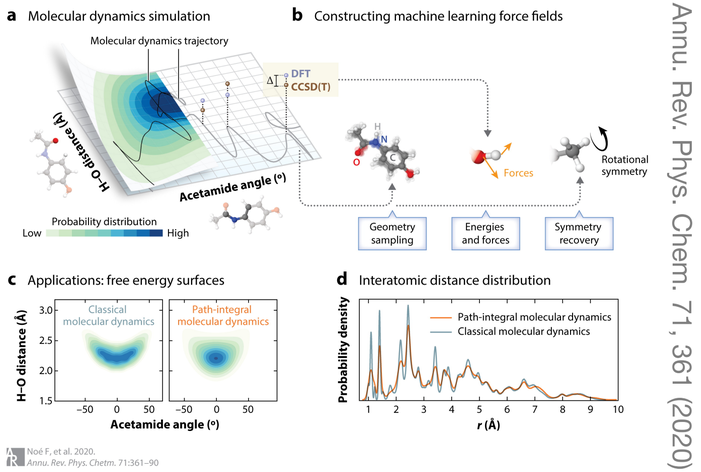
Machine learning - Industry, science, society

language processing

Image Credit: Getty

self-driving cars

medicine



Annu. Rev. Phys. Chem. 71, 361 (2020)

materials science/physics/chemistry

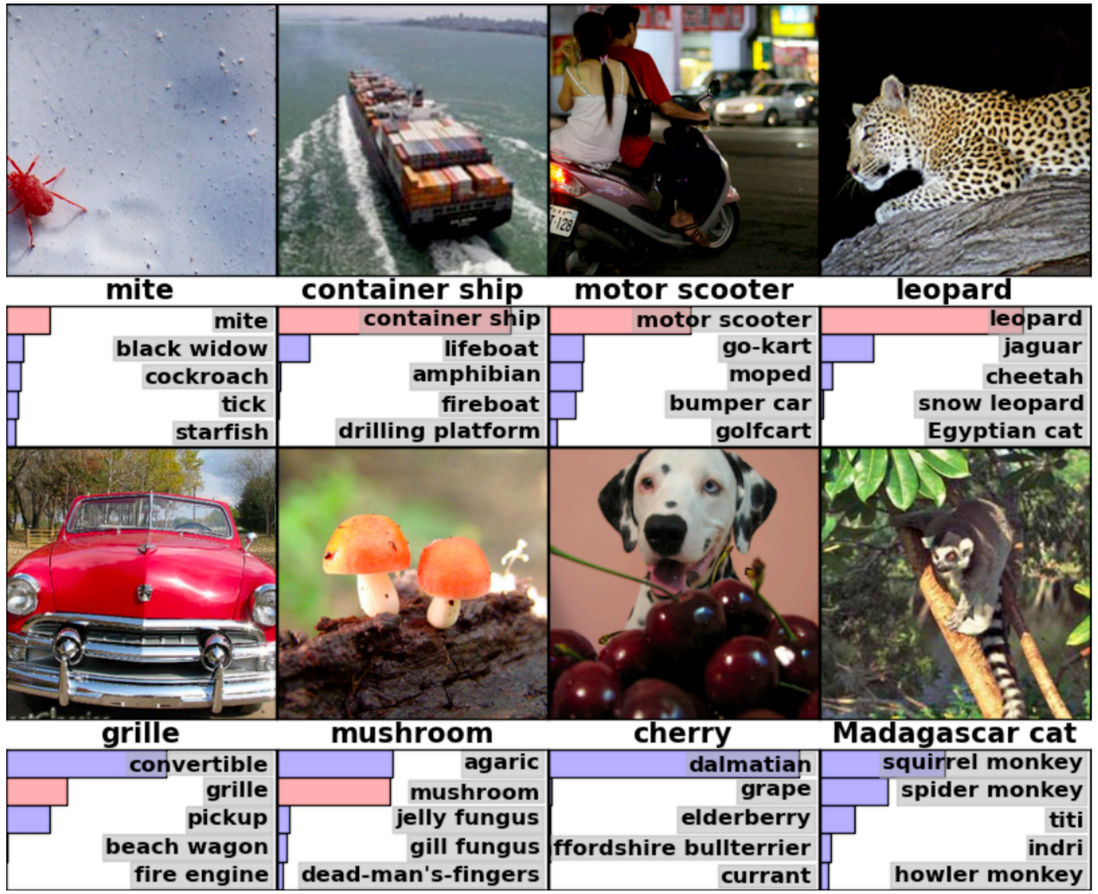
Image recognition

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca



2012 ILSVRC: paper with > 88000 citations

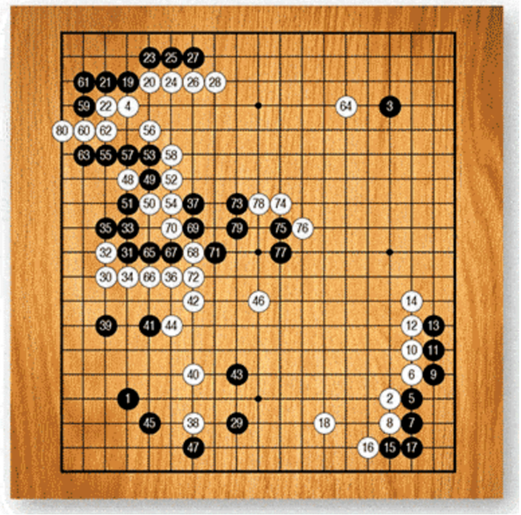
ImageNet:

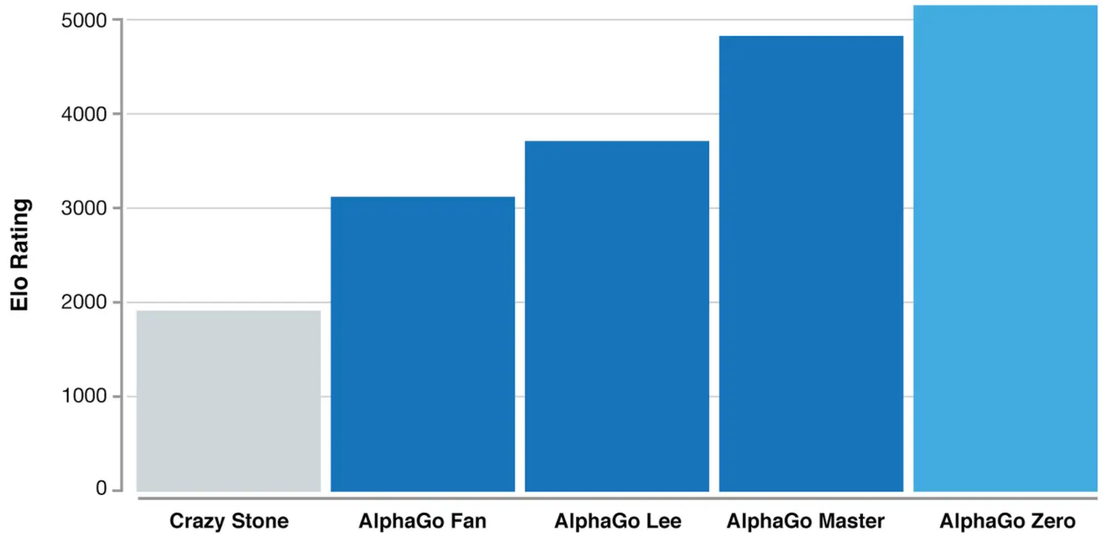
- 1.2 million training images (150k test)
- 15 % neural net error
- 26 % second best

Success at tasks previously thought impossible

Freie Universität Berlin

AlphaGo





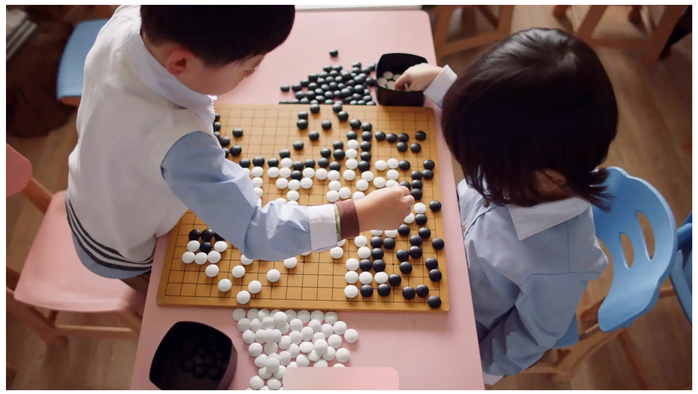


Image Credit: deepmind.com

Success at tasks previously thought impossible

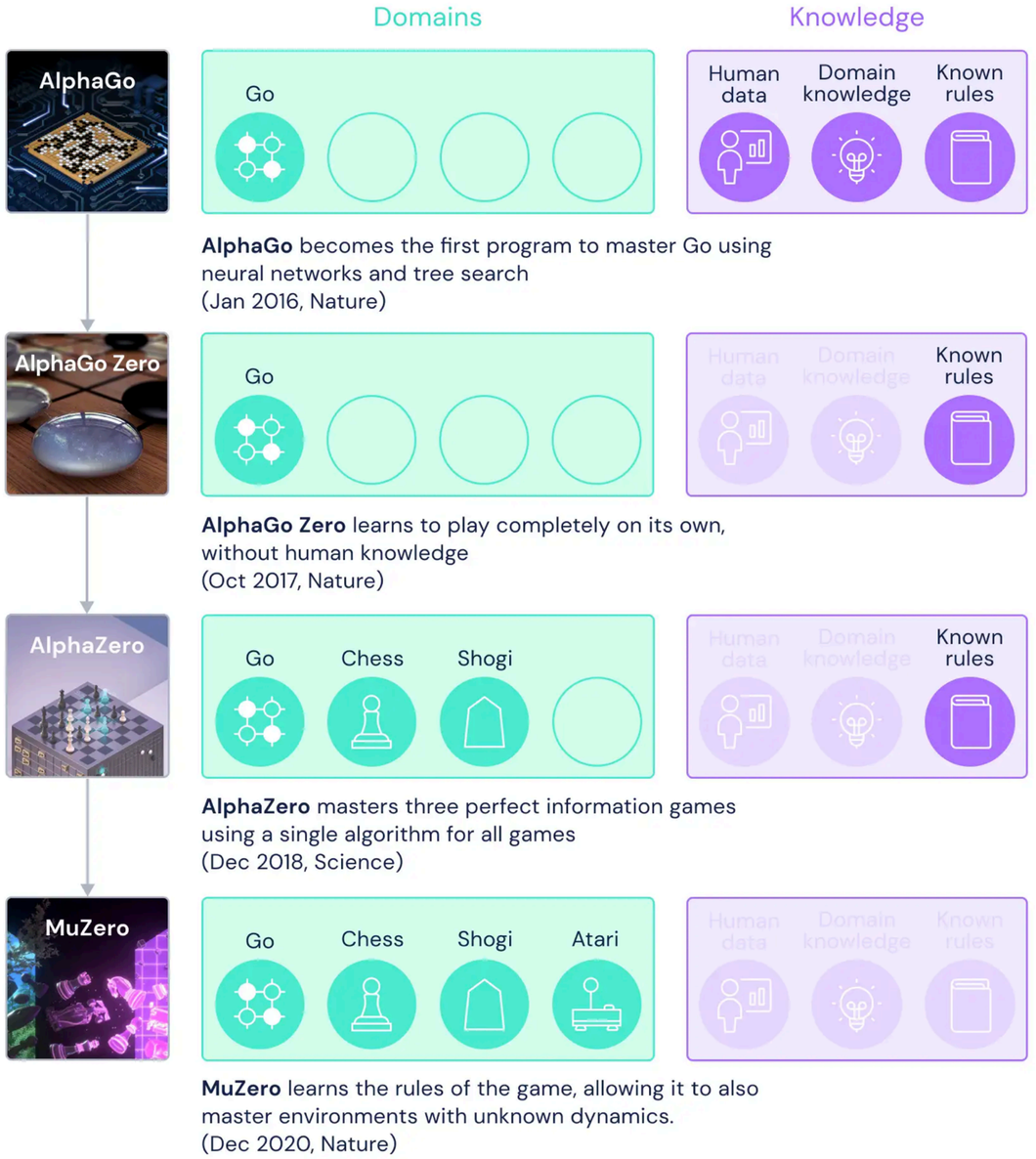


Image Credit: deepmind.com

Success at tasks previously thought impossible

Freie Universität Berlin

nature

Explore content | About the journal | Publish with us

nature > articles > article

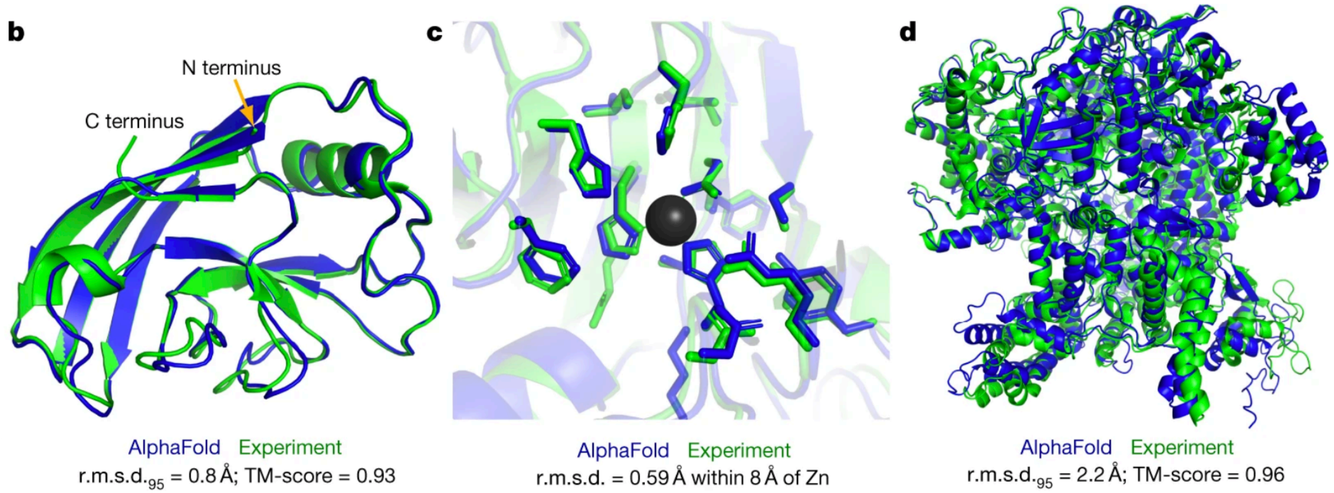
Article | Open Access | Published: 15 July 2021

Highly accurate protein structure prediction with AlphaFold

John Jumper, Richard Evans, [...] Demis Hassabis

Nature **596**, 583–589 (2021) | Cite this article

402k Accesses | **2805** Altmetric | Metrics



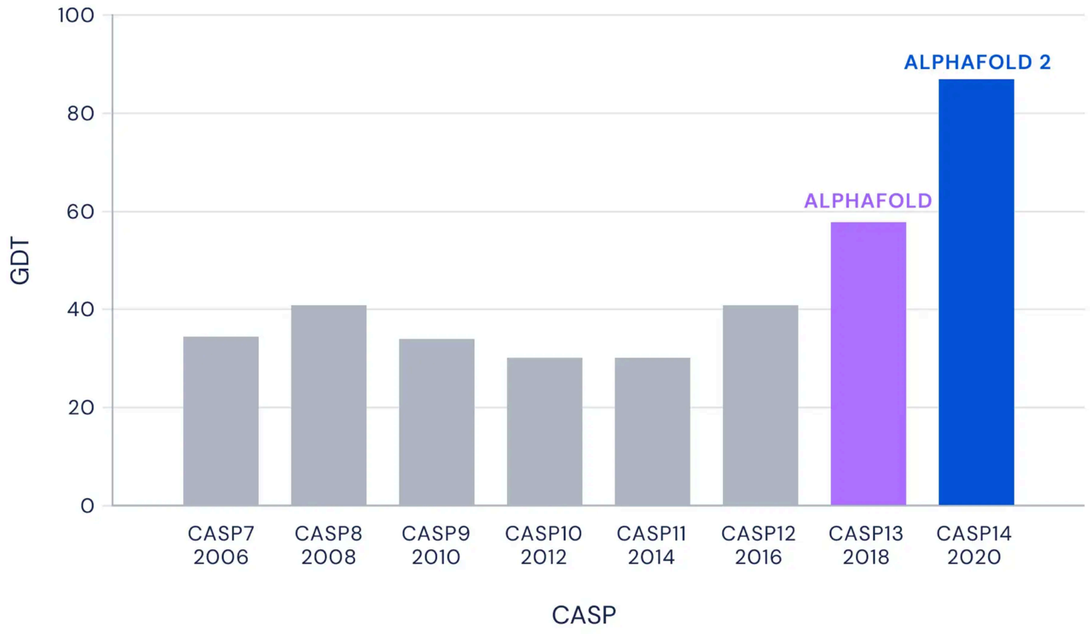


Image Credit: deepmind.com

What is machine learning?

Freie Universität Berlin

- data driven problem solving
- any system that, given more data, performs increasingly better at some task
- framework/philosophy, not single method

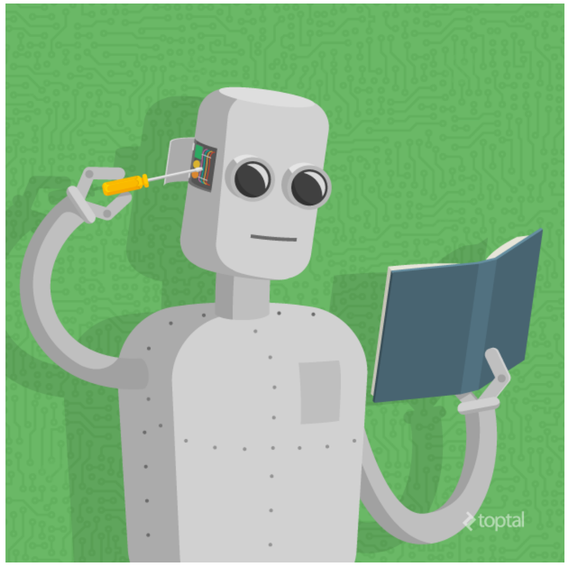
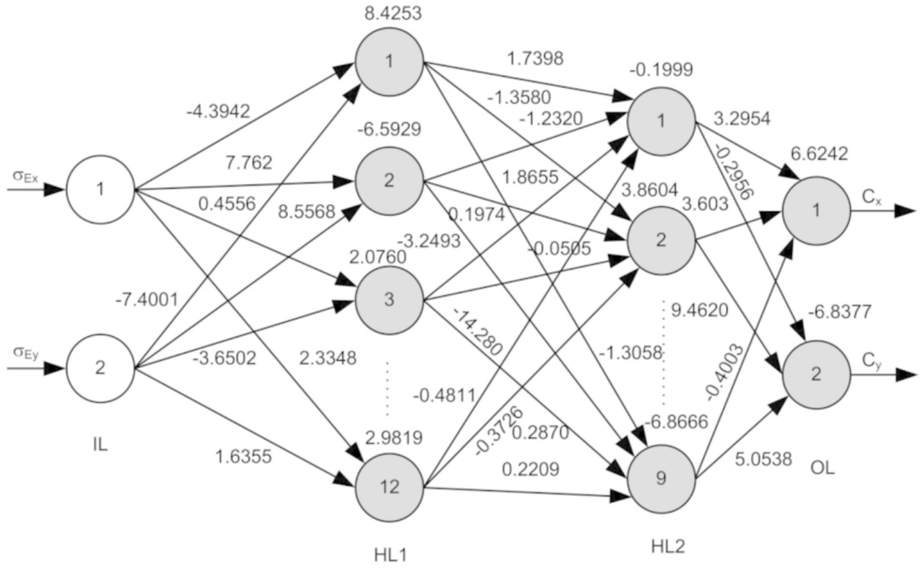


Image Credit: Toptal

What is machine learning?

Software 1.0

Software 2.0



Software 2.0

Andrej Karpathy Nov 11, 2017 · 9 min read

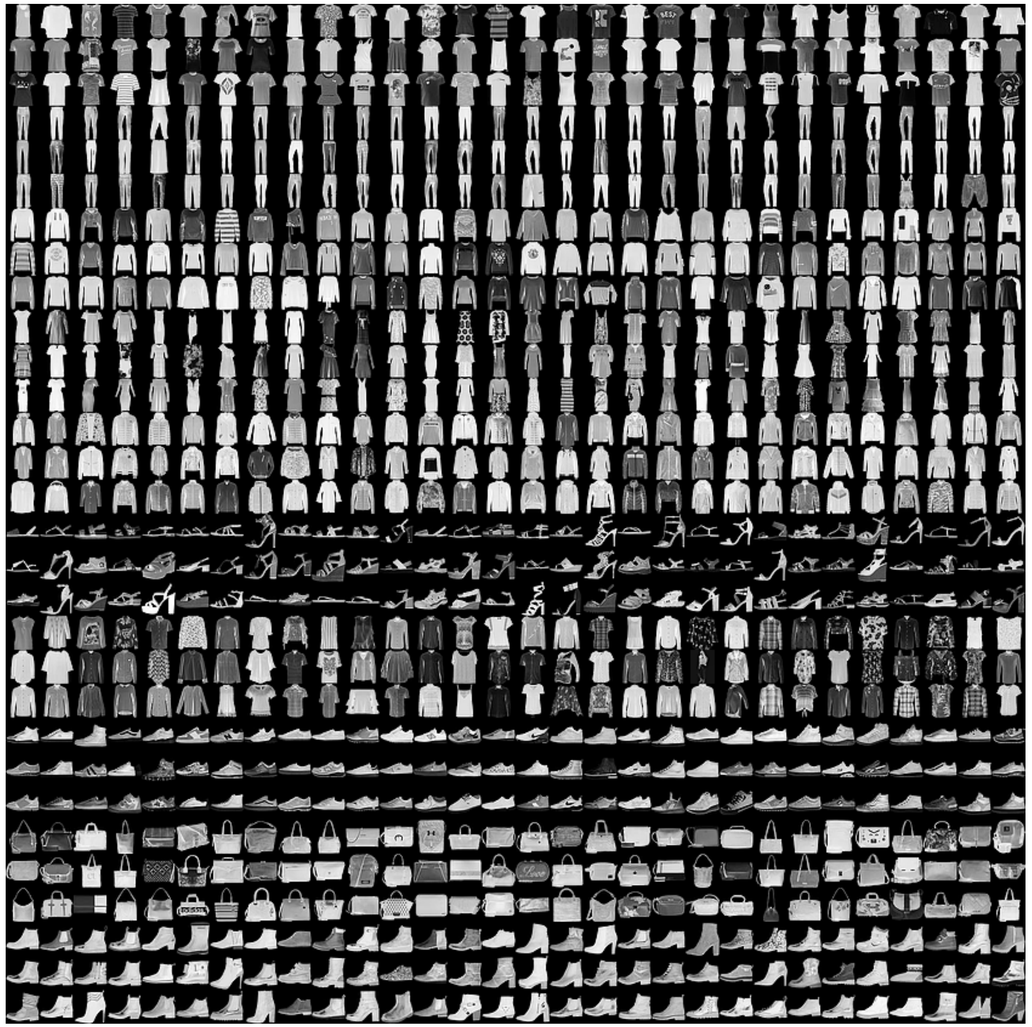
https://karpathy.medium.com/software-2-0-a64152b37c35

Basic concepts

Dataset: example - fashion MNIST

Freie Universität Berlin

- 70 000 labeled images
- 10 classes
- 28x28 grayscale images

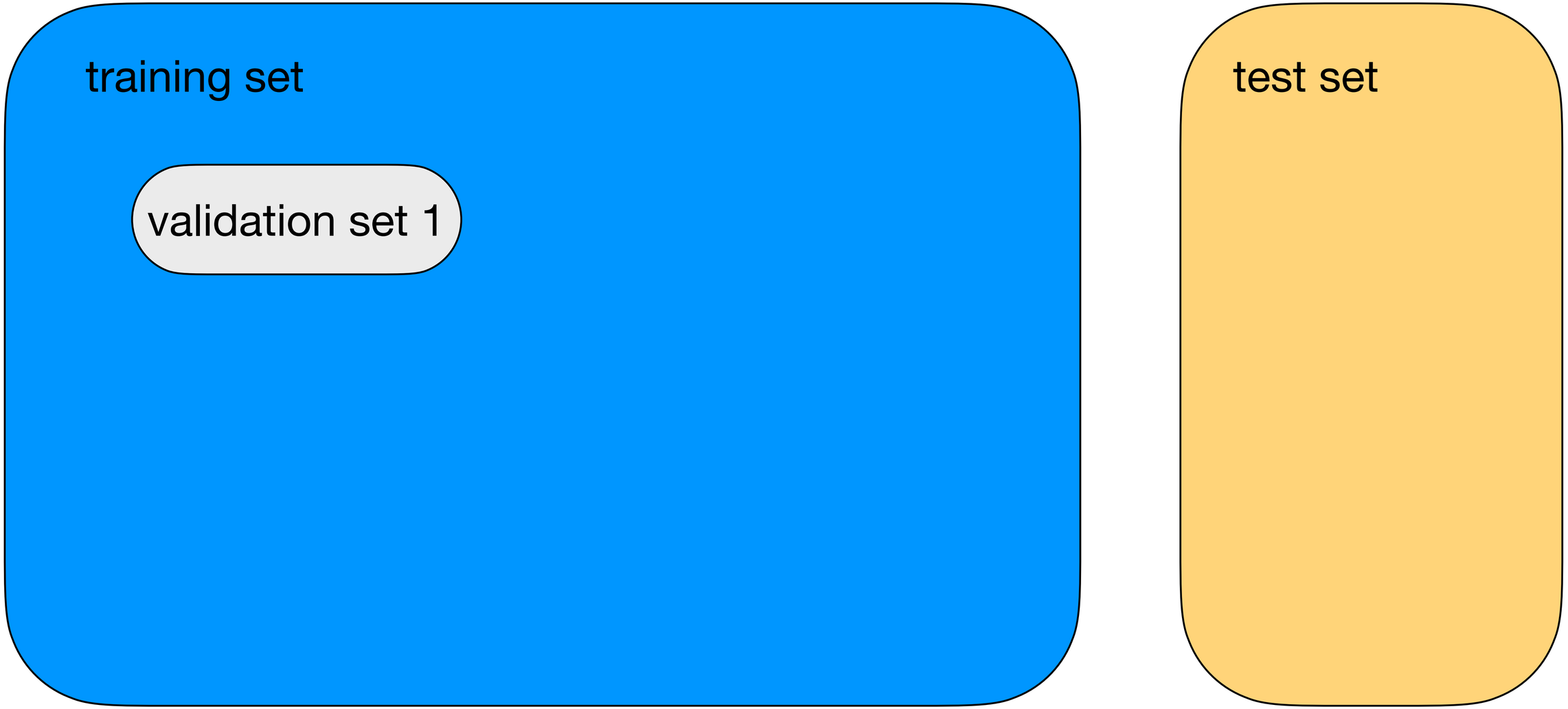


https://github.com/zalandoresearch/fashion-mnist

Dataset partioning

Dataset partioning

Dataset partioning



Dataset partioning

Dataset partioning

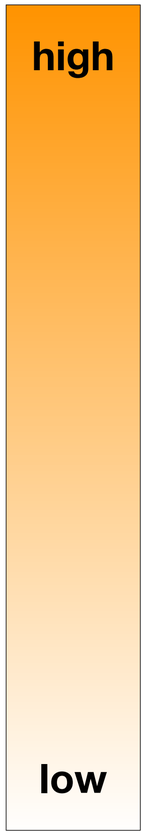
Dataset partioning

Types of learning tasks

Freie Universität Berlin

- supervised learning (labeled data)
- unsupervised learning (unlabeled data)
- reinforcement learning ('reward' data)

a priori knowledge



Supervised learning

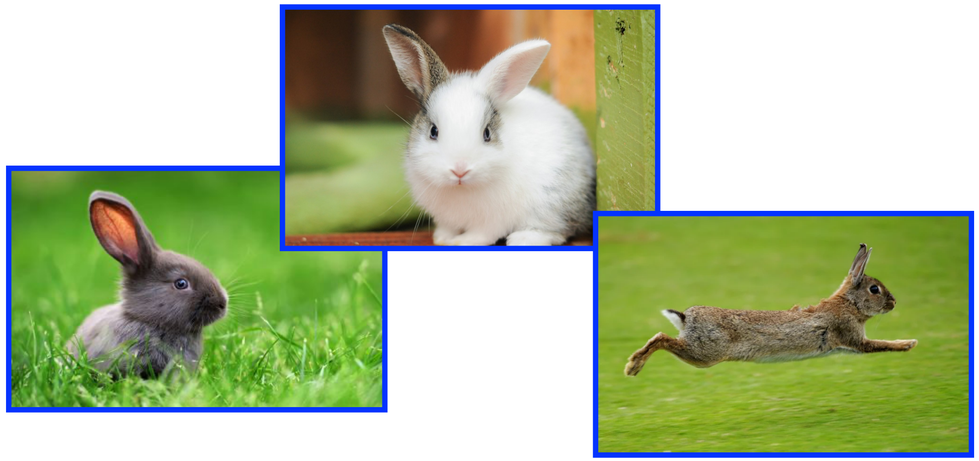
Freie Universität Berlin

- given labeled training data (labels A and B)
- find decision function $f(\mathbf{x})$

$$f(\mathbf{x}) > 0 \qquad \mathbf{x} \in A$$

$$f(\mathbf{x}) < 0 \qquad \mathbf{x} \in B$$

- example: identify photos of alligators and rabbits



Supervised learning

Freie Universität Berlin

- given training set $\{\mathbf{x}_j, y_j\}$, minimise cost function

$$C = \frac{1}{N_T} \sum_j (f(\mathbf{x}_j) - y_j)^2 \qquad y_j = \begin{cases} +1 & \mathbf{x}_j \in A \\ -1 & \mathbf{x}_j \in B \end{cases}$$

  by varying adjustable parameters of f

- cost function measures distance of trial function $f(\mathbf{x})$ from idealised ‘indicator’ function y_j

Unsupervised learning

Freie Universität Berlin

- given unlabeled training data $\{\mathbf{x}_j\}$
  - find function $f(\mathbf{x})$ such that $f(\mathbf{x}_j) \simeq p(\mathbf{x}_j)$
  - find function $f(\mathbf{x})$ such that $|f(\mathbf{x}_j)|^2 \simeq p(\mathbf{x}_j)$
  - find data clusters and which data belongs to each cluster
  - discover reduced representation of data for analysis or other learning tasks

Unsupervised learning

Freie Universität Berlin

- typical approach for inferring $p(\mathbf{x})$
  - given data $\{\mathbf{x}_j\}$, maximise log likelihood

$$\mathcal{L} = \sum_j \log p(\mathbf{x}_j)$$

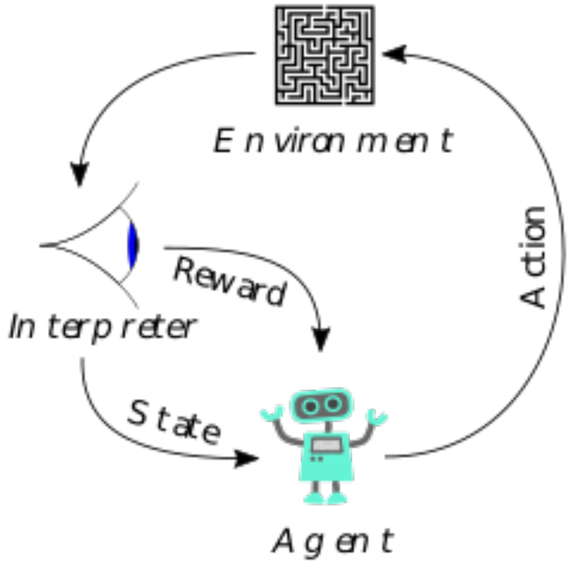
    by varying p

- can view log likelihood as distance measure between $p(\mathbf{x})$ and $p_{\mathrm{data}}(\mathbf{x}) = \sum_j \delta(\mathbf{x} - \mathbf{x}_j)$

  ("Kullback-Leibler divergence")

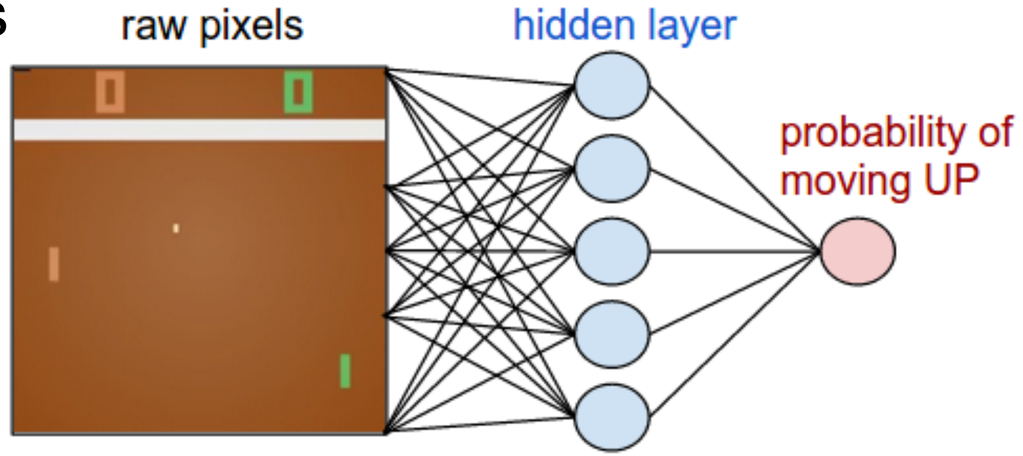
Reinforcement learning

- many flavours, but with common features
  - environment and agent with states s_n
  - agent actions a_n
  - reward $R(s_n)$ for being in state s_n



- goal: determine policy $P(s_n) \rightarrow a_n$,
  best actions to maximise reward in fewest steps

- example: learning “Pong” by observing
  screen state



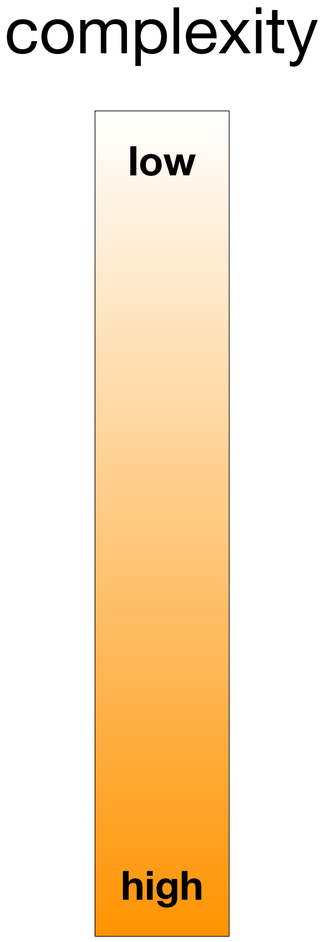
http://karpathy.github.io/2016/05/31/rl/

Model architectures

Types of models

Three most used models:

- linear model
- kernel learning / support vector machines
- neural networks



Linear supervised learning

$$f(\mathbf{x}) = \sum_{n=1}^{m} w_n x_n + w_0 = \mathbf{w} \cdot \mathbf{x} + w_0$$

- where $\mathbf{w} = (w_1, w_2, \ldots, w_m)$ and w_0 are the weights to be learned
- given training set $\{\mathbf{x}^{(j)}, y^{(j)}\}$, minimise cost function

$$C = \frac{1}{N_T} \sum_j (f(\mathbf{x}^{(j)}) - y^{(j)})^2$$

by varying adjustable parameters (weights) of f

Linear supervised learning

- cost function for linear model

$$C = \frac{1}{2N_T} \sum_j (\mathbf{w} \cdot \mathbf{x}^{(j)} - y^{(j)})^2$$

- gradient with respect to n^{th} component

$$\frac{\partial C}{\partial w_n} = \frac{1}{N_T} \sum_j (\mathbf{w} \cdot \mathbf{x}^{(j)} - y^{(j)})\, x_n^{(j)}$$

- update w_n with negative gradient times small step α

$$w_n \leftarrow w_n - \alpha \frac{\partial C}{\partial w_n}$$

Regularisation

- find the ‘simplest’ model (keep weights small)
- ridge regression (L2 / Tikhonov regularisation)

$$C = \frac{1}{2N_T} \sum_j (\mathbf{w} \cdot \mathbf{x}^{(j)} - y^{(j)})^2 + \lambda ||\mathbf{w}||_2^2$$

Sparsity-inducing regularisation

- L0, penalise number of parameters

$$C = \frac{1}{2N_T}\sum_j (\mathbf{w}\cdot\mathbf{x}^{(j)} - y^{(j)})^2 + \lambda||\mathbf{w}||_0$$

- L1 / LASSO

$$C = \frac{1}{2N_T}\sum_j (\mathbf{w}\cdot\mathbf{x}^{(j)} - y^{(j)})^2 + \lambda||\mathbf{w}||_1$$

- elastic net, sparsity and smoothness

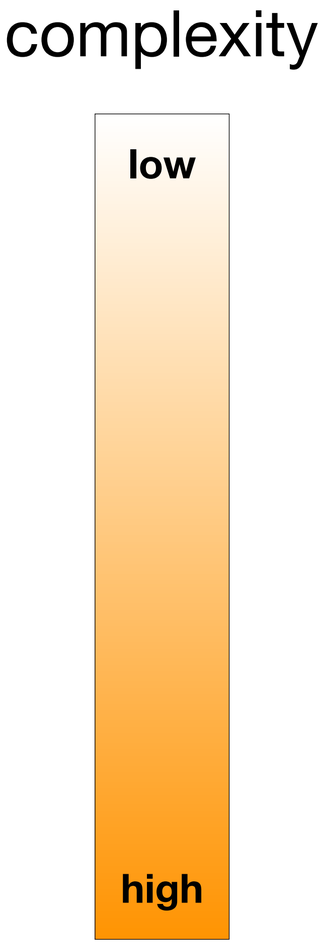
$$C = \frac{1}{2N_T}\sum_j (\mathbf{w}\cdot\mathbf{x}^{(j)} - y^{(j)})^2 + \lambda(\alpha||\mathbf{w}||_1 + (1-\alpha)||\mathbf{w}||_2^2)$$

Types of models

Freie Universität Berlin

Three most used models:

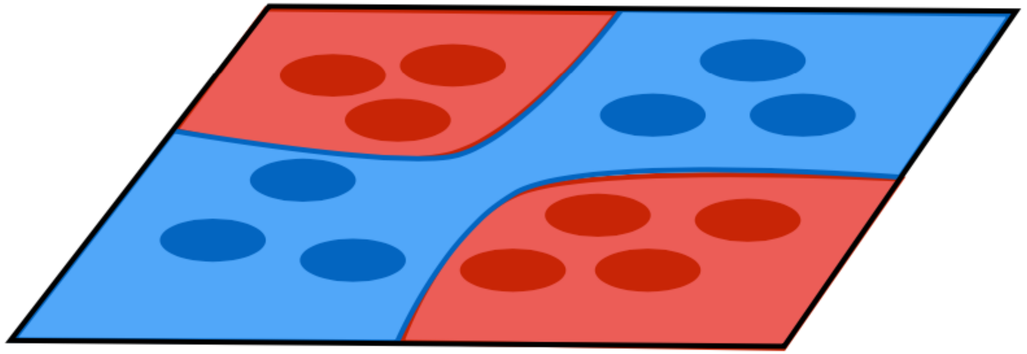
- linear model
- kernel learning / support vector machines
- neural networks



Kernel methods

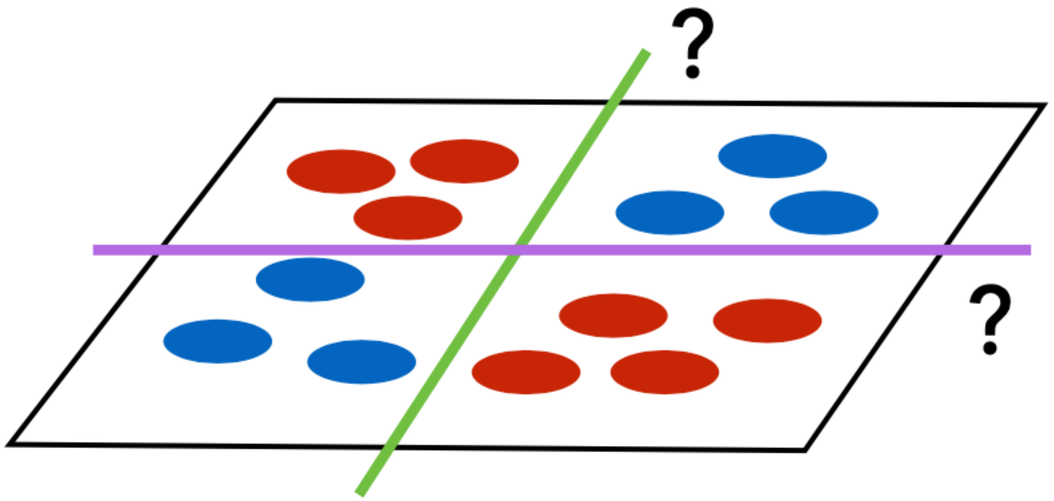
Freie Universität Berlin

- want $f(\mathbf{x})$ to separate classes, e.g.



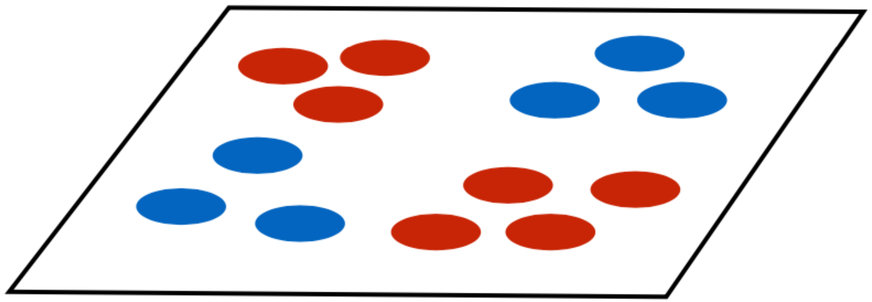
- linear classifier may be insufficient

$$f(\mathbf{x}) = \sum_{n=1}^{m} w_n x_n + w_0 = \mathbf{w} \cdot \mathbf{x} + w_0$$



Kernel methods

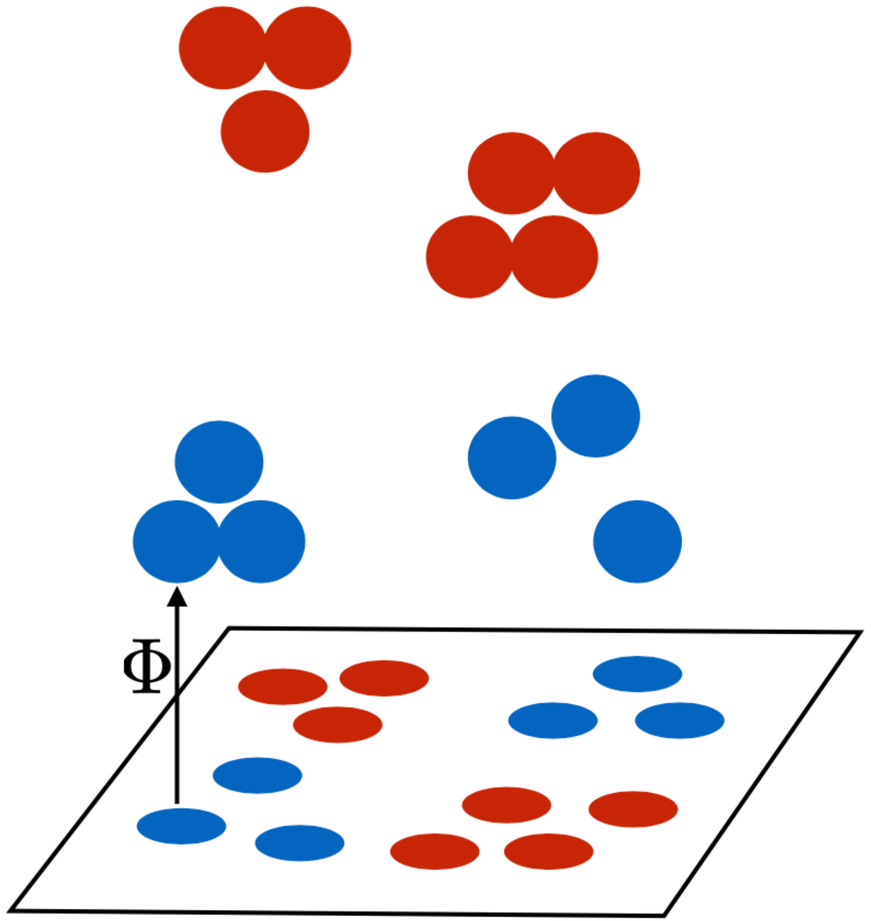
- apply non-linear *feature map* $\mathbf{x} \rightarrow \mathbf{\Phi}(\mathbf{x})$



Kernel methods

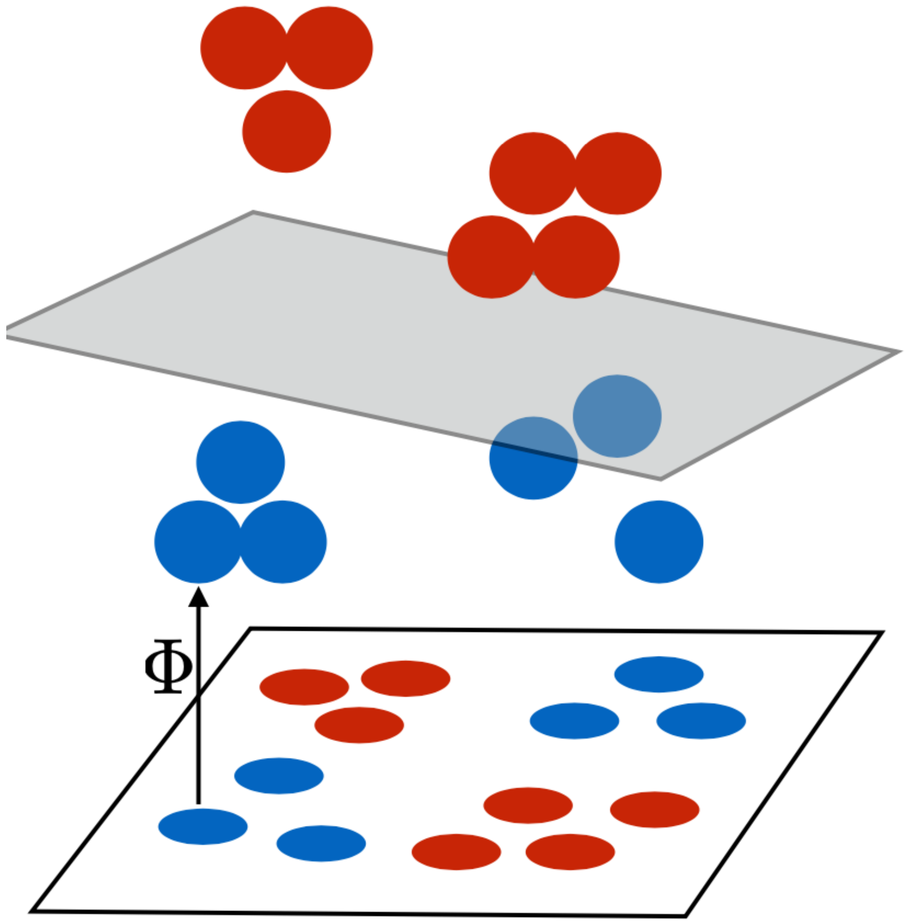
Freie Universität Berlin

- apply non-linear *feature map* $\mathbf{x} \rightarrow \mathbf{\Phi}(\mathbf{x})$



Kernel methods

- apply non-linear *feature map* $\mathbf{x} \rightarrow \mathbf{\Phi}(\mathbf{x})$



decision function:

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x})$$

Kernel methods

Freie Universität Berlin

- key ideas:
  - never perform feature transformation $\mathbf{x} \rightarrow \mathbf{\Phi}(\mathbf{x})$ explicitly
  - instead define kernel function $K_{ij} = k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \mathbf{\Phi}(\mathbf{x}^{(i)}) \cdot \mathbf{\Phi}(\mathbf{x}^{(j)})$
  - kernel function implicitly corresponds to a large/infinite feature space
  - predicted values: $f(\mathbf{x}') = \sum_{n=1}^{N_T} a_n k(\mathbf{x}^{(n)}, \mathbf{x}')$ with $\mathbf{a} = (\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{f}(\mathbf{x})$

Kernel functions

- generalised polynomial kernel

$$k(\mathbf{x}, \mathbf{x}') = (\mathbf{x} \cdot \mathbf{x}' + c)^d$$

- Gaussian kernel / squared exponential kernel (infinite dimensional space)

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2\sigma^2}||\mathbf{x} - \mathbf{x}'||^2\right)$$

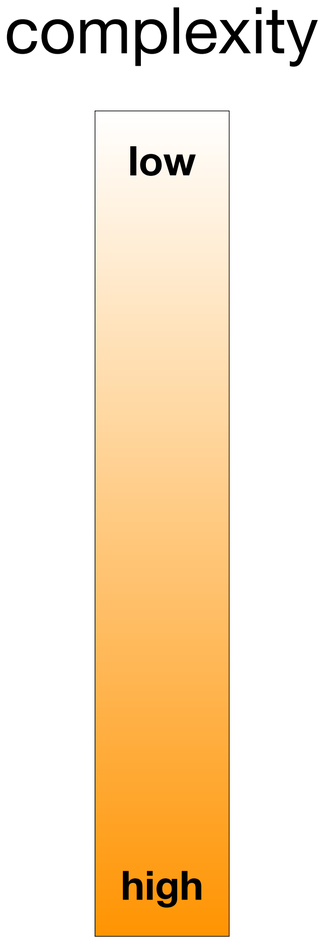
- radial basis function

$$k(\mathbf{x}, \mathbf{x}') = k\left(||\mathbf{x} - \mathbf{x}'||^2\right)$$

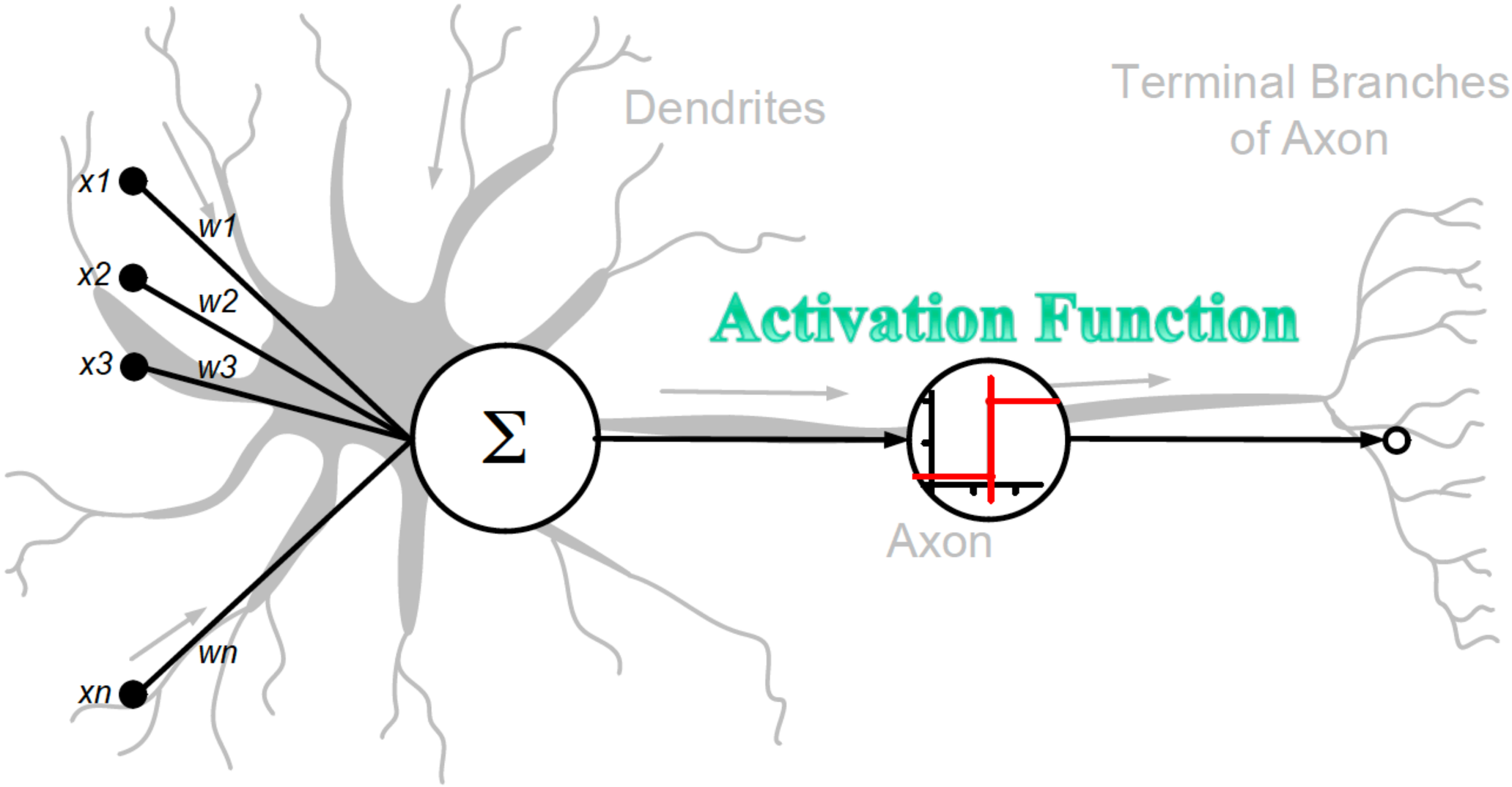
Types of models

Three most used models:

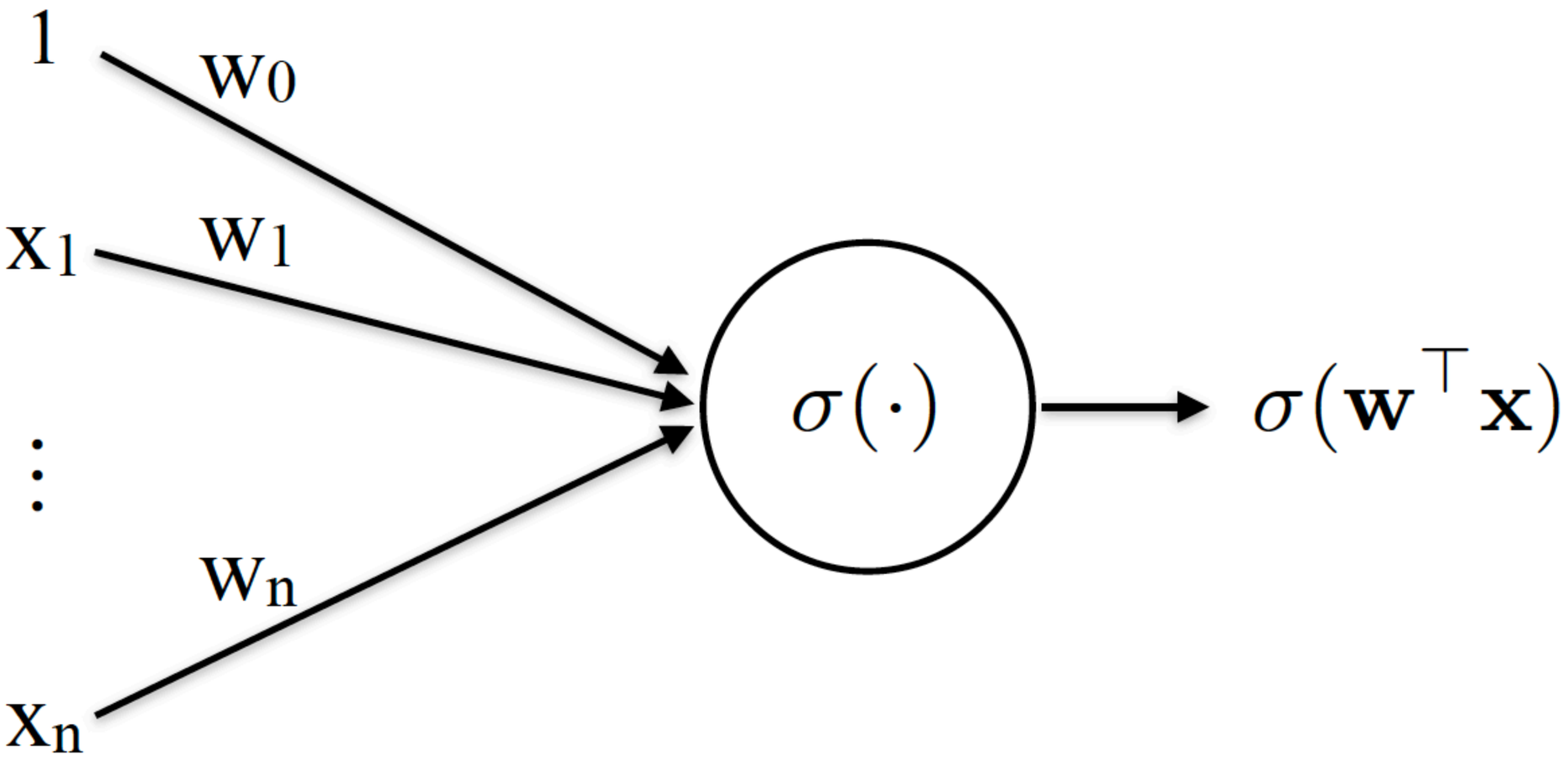
- linear model
- kernel learning / support vector machines
- neural networks



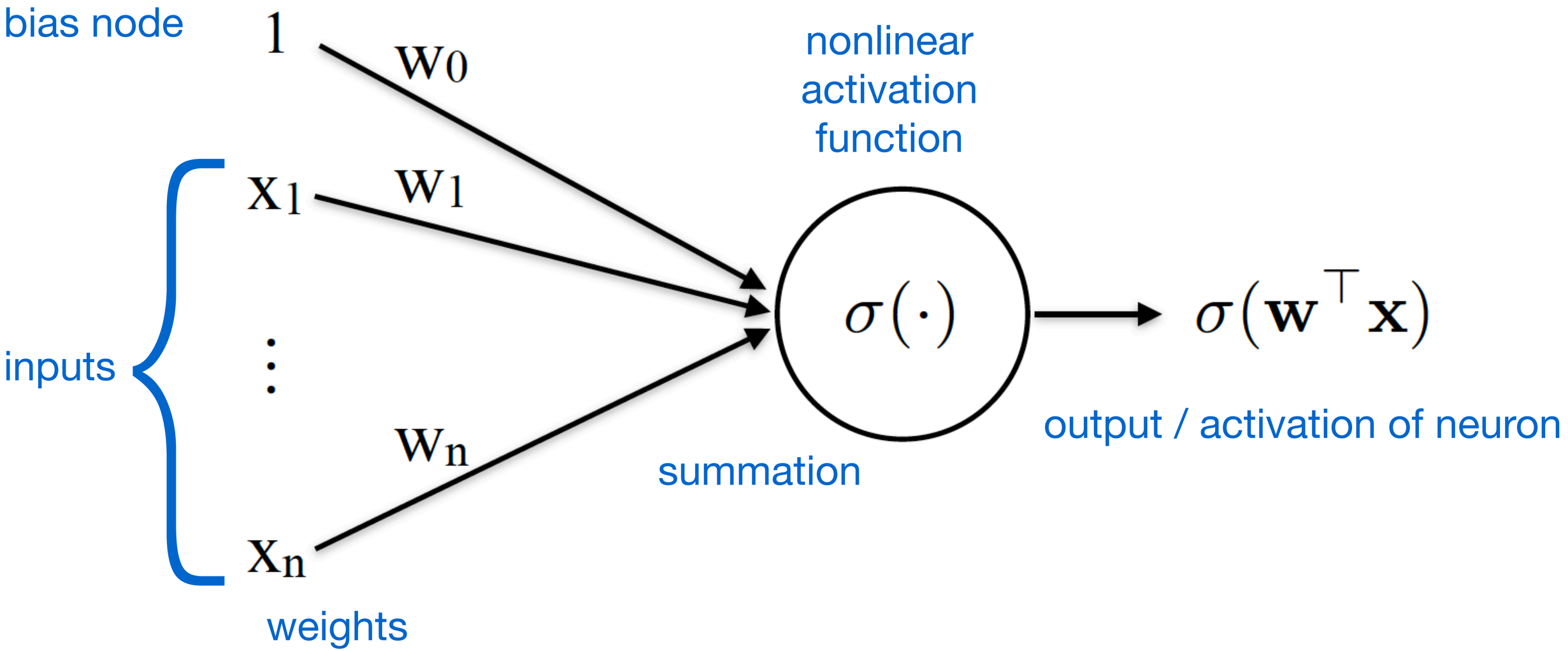
Neuron


Dendrites
Terminal Branches
of Axon
x1
w1
x2
w2
x3
w3
Σ
Activation Function
Axon
wn
xn

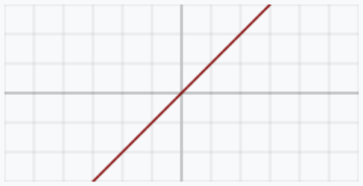
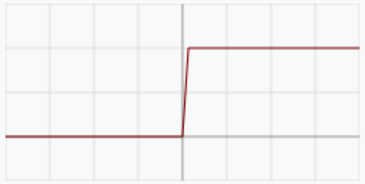
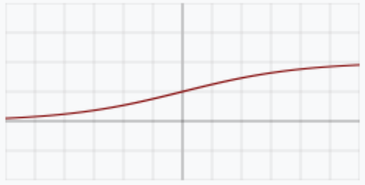
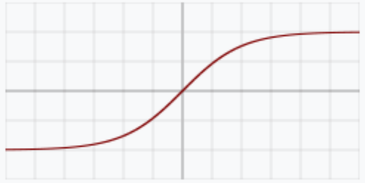
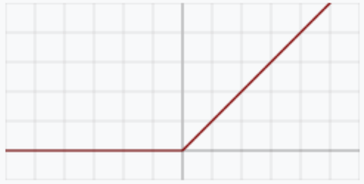
Artificial neuron



Artificial neuron



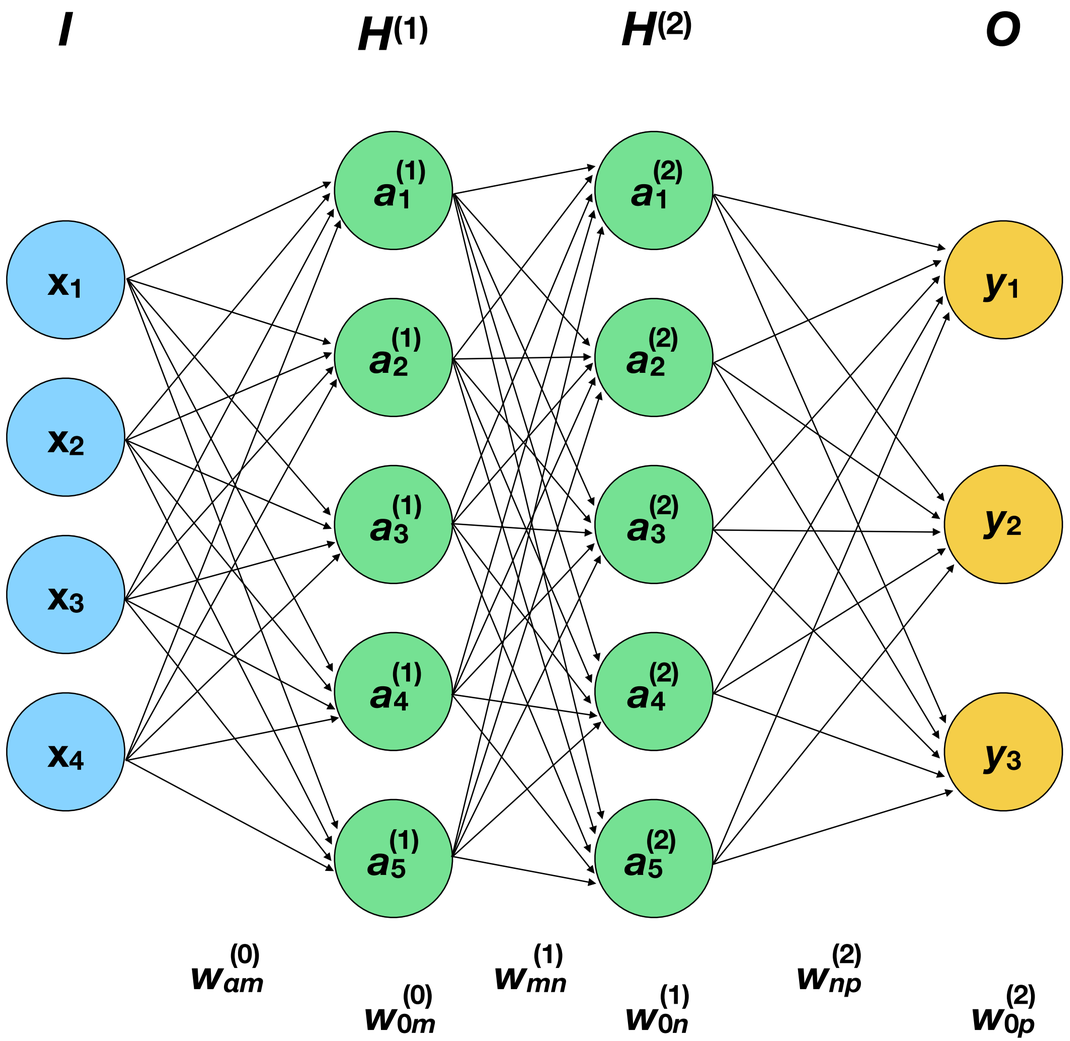
Activation functions

Identity		x
Binary step		$\begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$
Logistic, sigmoid, or soft step		$\sigma(x) = \frac{1}{1 + e^{-x}}$
Hyperbolic tangent (tanh)		$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$
Rectified linear unit (ReLU)[10]		$\begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$ $= \max\{0, x\} = x\mathbf{1}_{x>0}$

https://en.wikipedia.org/wiki/Activation_function

Feedforward neural network

- single-layer perceptron (Rosenblatt 1957)
- multi-layer perceptron (~ 1985)
- ‘deep’ NN: more than 1 hidden layer
- backpropagation: derivative of loss/cost function wrt. weights through network to update weights



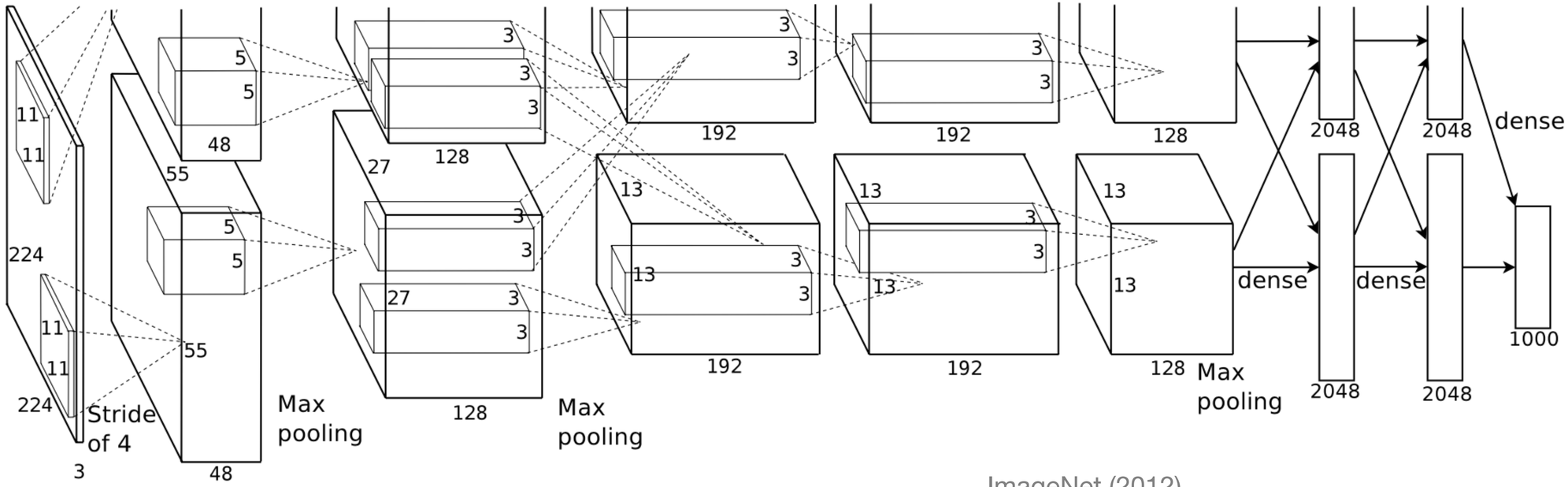
Universal approximation theorem

Freie Universität Berlin

- Hornik *et al.* (1989), Cybenko (1989)
- a fully connected NN with a *single* hidden layer can approximate any continuous univariate function with arbitrary accuracy in the limit of large hidden layer size
- however:
  - optimal solution might be difficult to find
  - might not be efficient to learn
  - deep NN even with narrow layers more efficient for many datasets

Convolutional neural networks

- ‘convolutional layers’
- sparser weight layers with few parameters



ImageNet (2012)

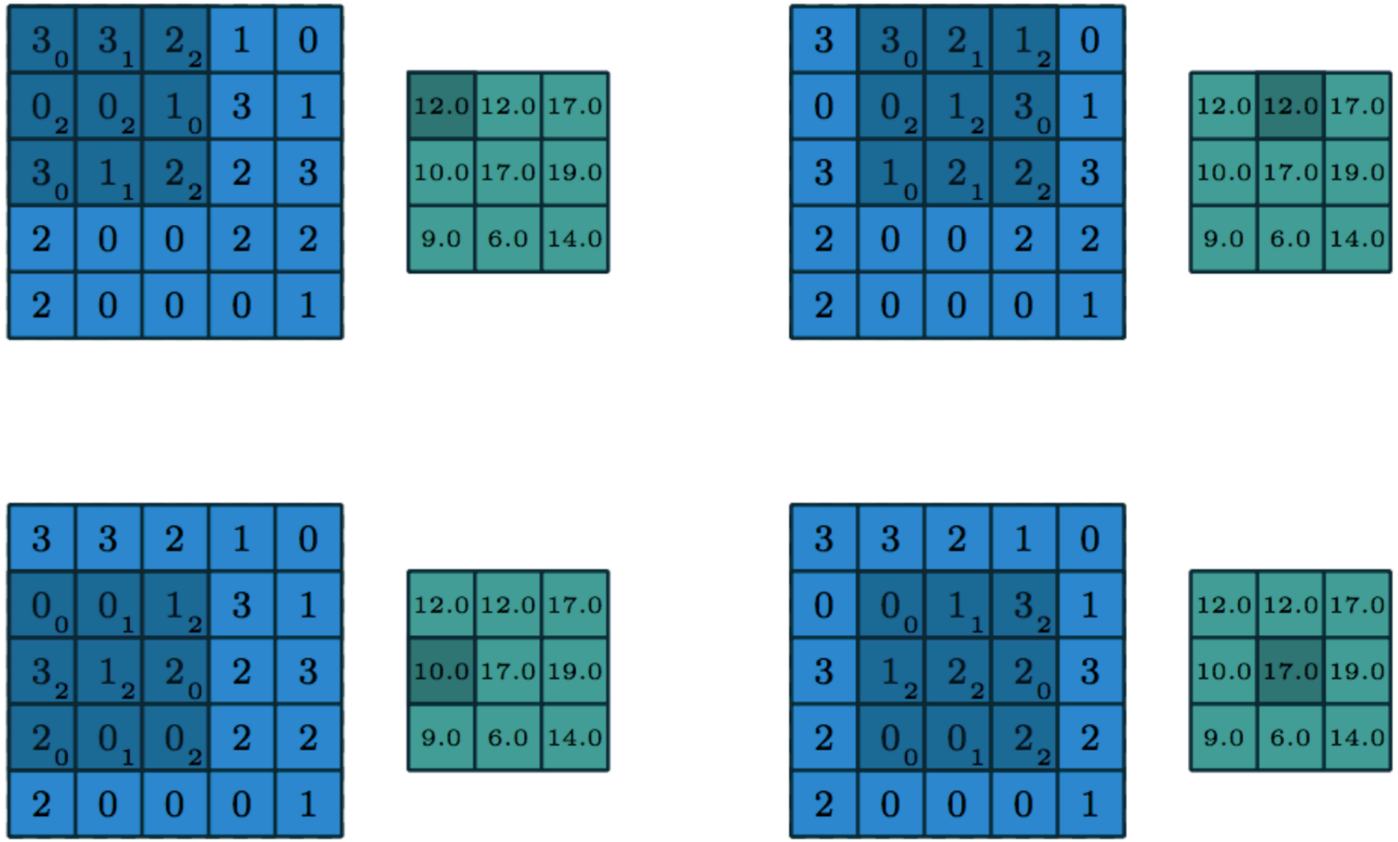
Convolutional filters

Freie Universität Berlin

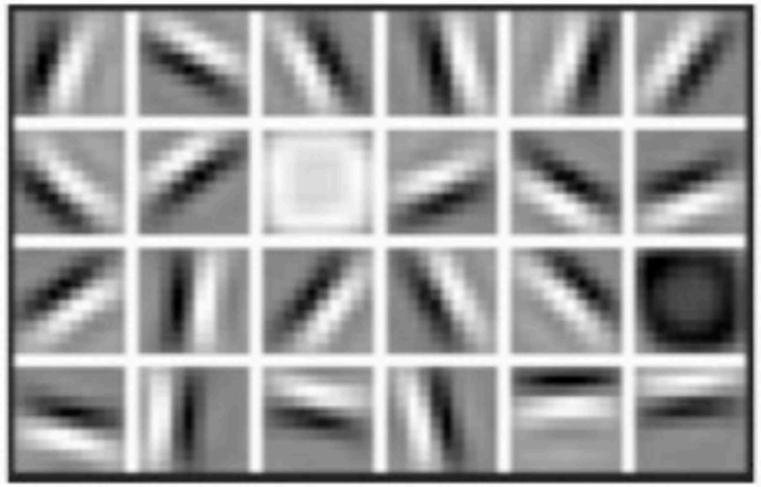
Convolutional Kernel / Filter

0	1	2
2	2	0
0	1	2

Apply convolutions



Example: face recognition



first layer representation

second layer representation

third layer representation

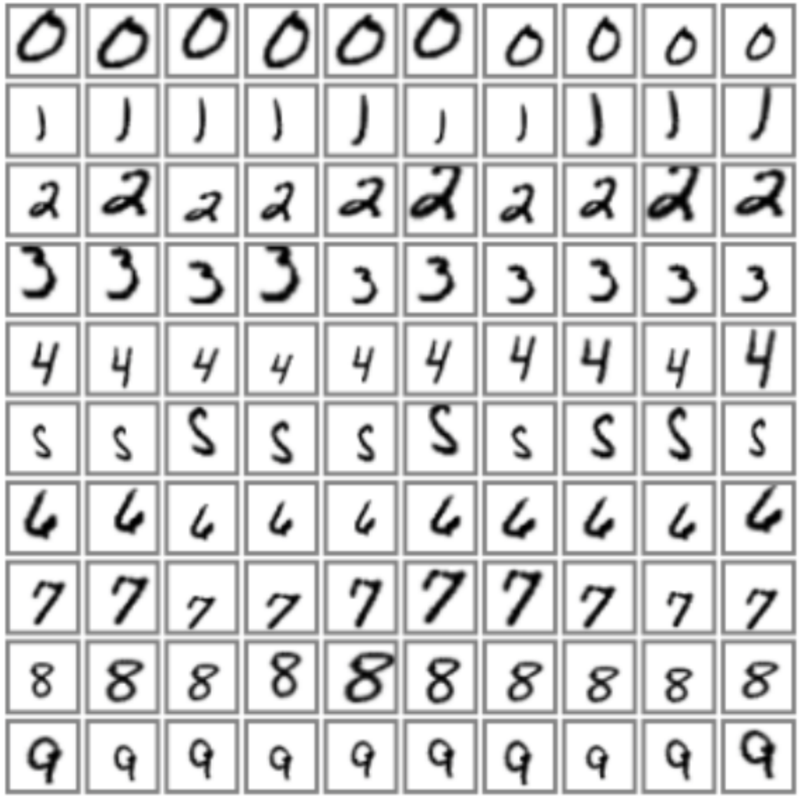
MNIST database

60 000 training images

10 000 test images, test error: 0.95 %

60 000 original + 540 000 distortion

test error: 0.8 %



Misclassified examples