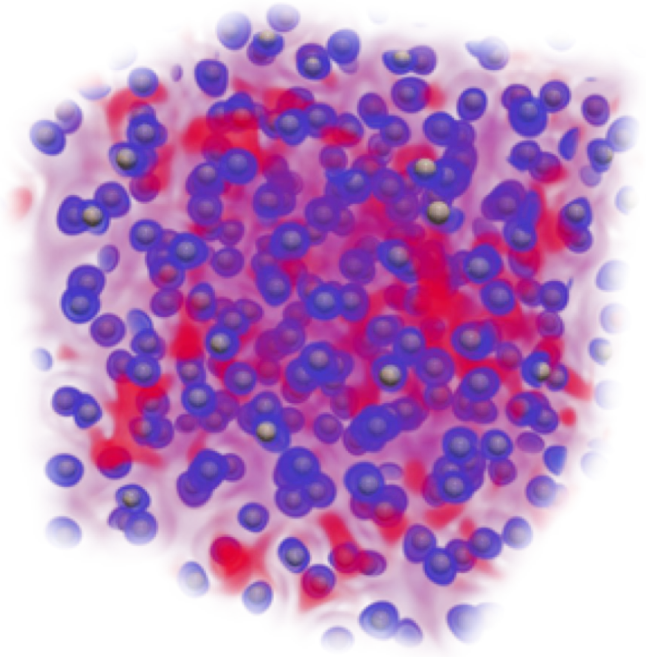


Atomistic Simulation Methods

Machine Learning Interatomic Potentials (1/2)

Helmholtz-Zentrum Dresden-Rossendorf · Center for Advanced Systems Understanding · Machine Learning for Materials Design · Attila Cangi · a.cangi@hzdr.de

INSTITUTE OF

PARTICIPATING INSTITUTIONS

FUNDED BY

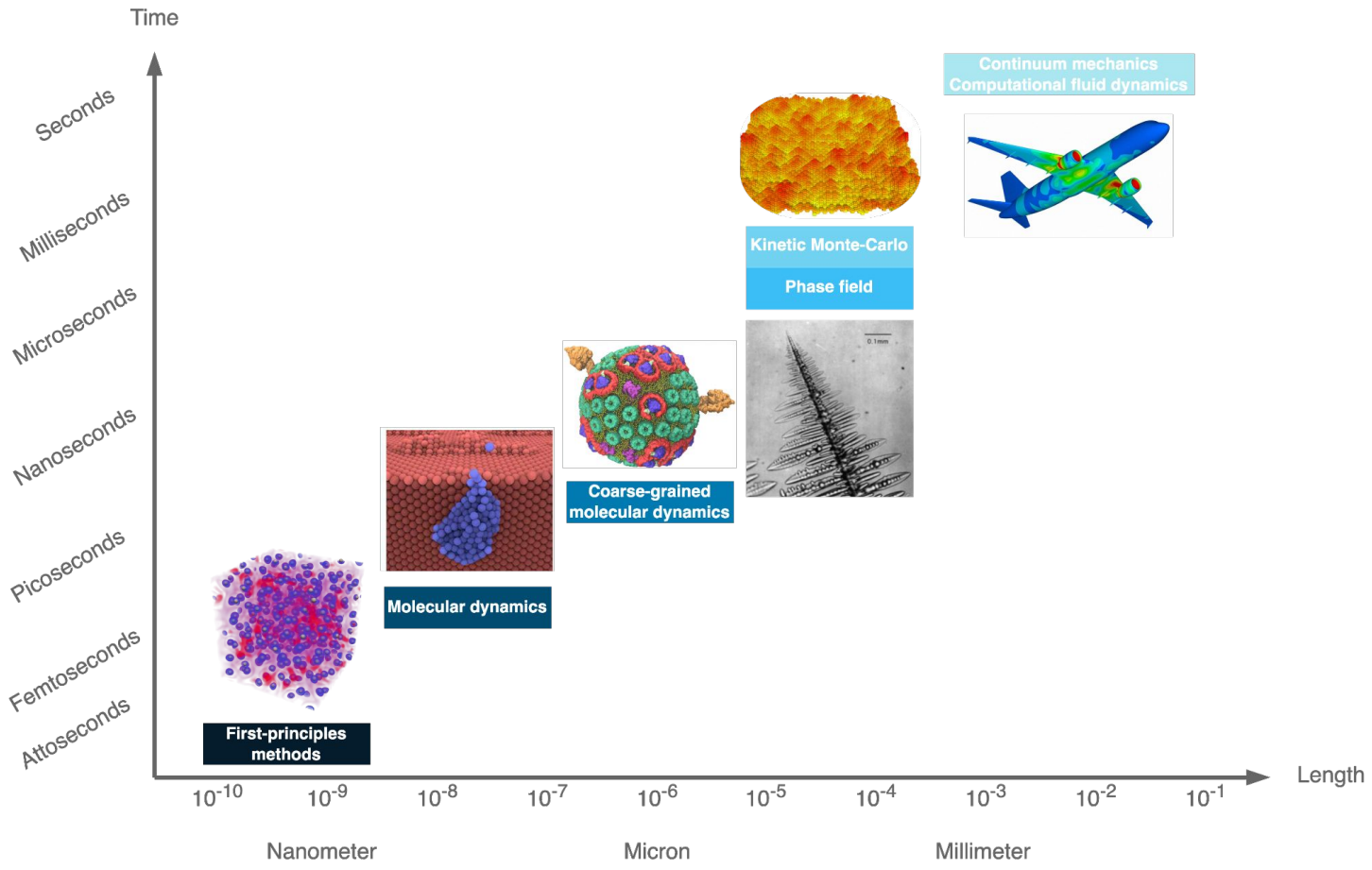
Diese Maßnahme wird mitfinanziert mit Steuermitteln auf Grundlage des vom Sächsischen Landtag beschlossenen Haushaltes.

Attila Cangi | a.cangi@hzdr.de

1. Introduction

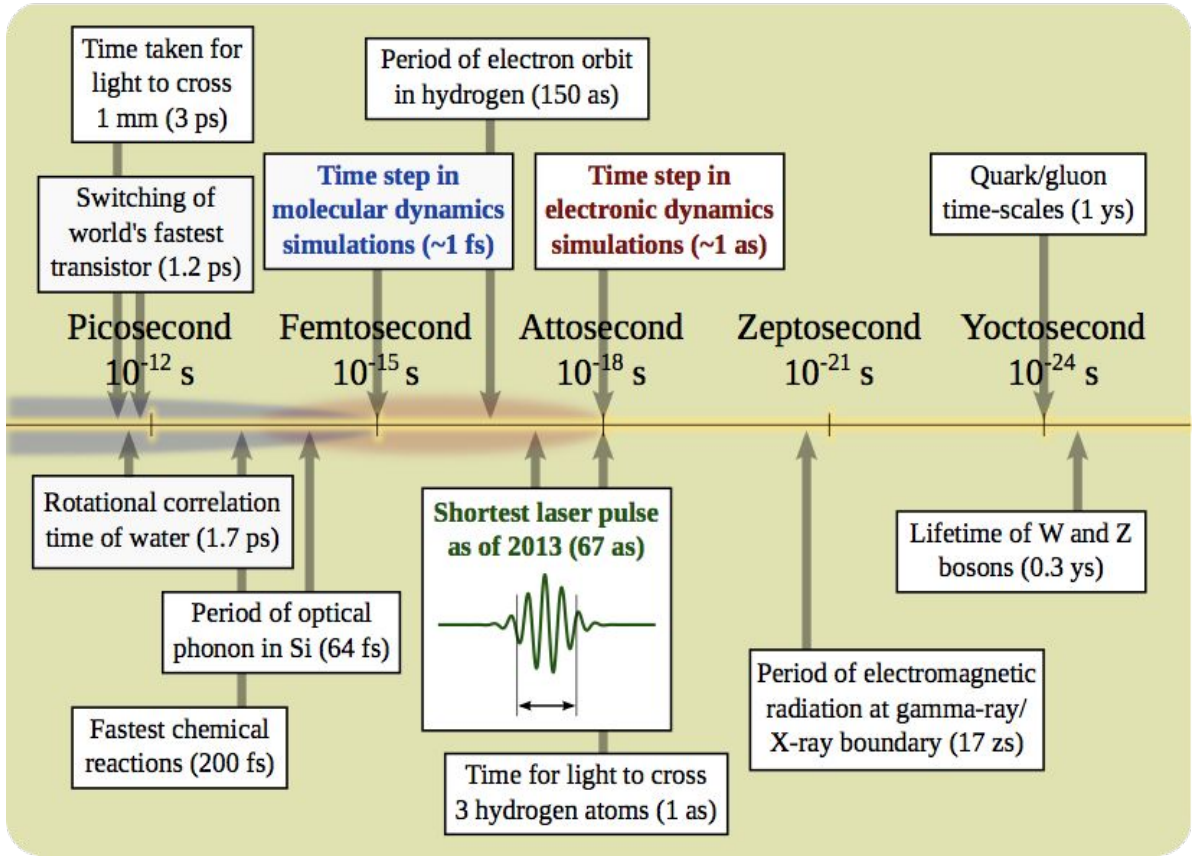
1. Introduction

Materials simulations require multiple length and time scales



1. Introduction

Time scales of first-principles and atomistic methods



Courtesy of Kay Dewhurst, Max Planck Institute of Microstructure Physics (2015).

- **Electronic dynamics**
  - Time-dependent DFT
  - Timesteps of attoseconds
- **First-principles and atomistic molecular dynamics**
  - DFT and classical molecular dynamics
  - Timesteps of femtoseconds

2. Born-Oppenheimer Potential Energy Surface

2.1 Molecular Hamiltonian

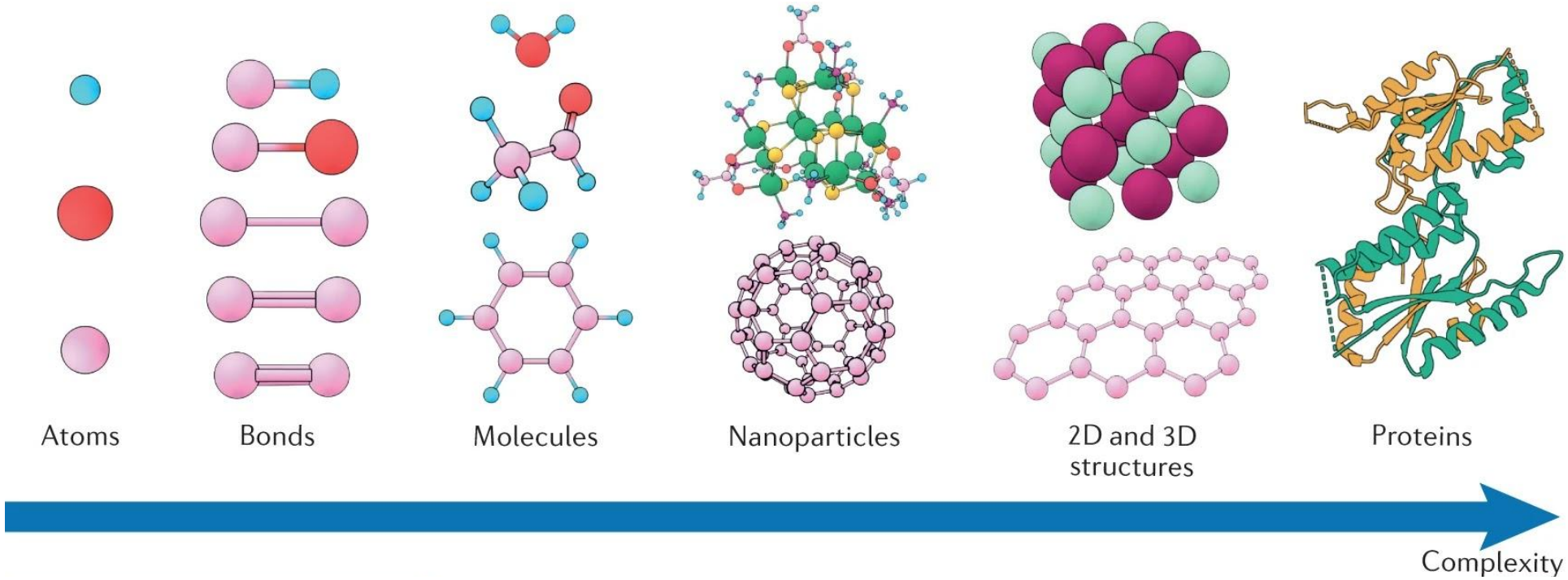
2.2 Schrödinger Equation of Coupled Electrons and Ions

2.3 Born-Oppenheimer Approximation

2.4 Born-Oppenheimer Potential Energy Surface

2.5 Classical Interatomic Potentials

2.1 Molecular Hamiltonian



N. Fedik et al., Nature Reviews Chemistry 6, 653 (2022).

2.1 Molecular Hamiltonian

- Consider a collection of
  - **electrons** $\underline{\mathbf{r}} = \{\mathbf{r}_1, \ldots, \mathbf{r}_{N_e}\}, \quad \mathbf{r}_j \in \mathbb{R}^3$
  - **and ions** $\underline{\mathbf{R}} = \mathbf{R}_1, \ldots, \mathbf{R}_{N_i}\}, \quad \mathbf{R}_\alpha \in \mathbb{R}^3, \quad M_\alpha(\text{mass}), \quad Z_\alpha(\text{charge})$
- Non-relativistic quantum mechanics.
- We work within atomic units $\hbar = m_e = e^2 = 1$
  - Energies are expressed in Hartree
  - Lengths are expressed in Bohr radii
- For the sake of brevity we do not take into account spins.

Attila Cangi | a.cangi@hzdr.de

Born-Oppenheimer approximation

Consider the molecular Hamiltonian of interacting electrons and ions:

$$H = T_e + V_{ee} + T_i + V_{ii} + V_{ei}$$

Consider the mass of an electron
$m_e \approx 9.11 \cdot 10^{-31}$ kg
versus the mass of a proton
$m_p \approx 1.67 \cdot 10^{-27}$ kg.
$\rightarrow \frac{m_p}{m_e} \approx 1836$.

That means we can consider the ions to be clamped.

So we can consider the so-called electronic Hamiltonian

$$H_e = T_e + V_{ee} + V_{ei} + V_{ii}$$

and the corresponding electronic Schrödinger equation:

$$H_e \phi_k(\underline{r};\underline{R}) = E_k \phi_k(\underline{r};\underline{R})$$

Because the ions are clamped, ϕ_k depends on the electronic coordinates and, in addition, only parametrically on the coordinates of the ions.

Now recall the Schrödinger equation of the full problem:

$$H(\underline{r},\underline{R})\,\psi(\underline{r},\underline{R}) = E\,\psi(\underline{r},\underline{R})$$

Under the assumption of clamped ions, we can make the following ansatz for the wave functions:

$$\psi(\underline{r},\underline{R}) = \sum_k \phi_k(\underline{r};\underline{R})\,\chi_k(\underline{R})$$

That means we separate the solution into a part that depends on the electrons and one part that depends on the ions. Furthermore, we are expanding in the basis of the electronic solution.

We now insert this into the Schrödinger equation

$$H\psi(\underline{r},\underline{R}) = E\,\psi(\underline{r},\underline{R})$$

$$H(\underline{r},\underline{R}) \sum_k \phi_k(\underline{r};\underline{R})\chi_k(\underline{R}) = E \sum_k \phi_k(\underline{r};\underline{R})\chi_k(\underline{R})$$

$$\left(T_n + H_e\right) \sum_k \phi_k(\underline{r};\underline{R})\chi_k(\underline{R}) = E \sum_k \phi_k(\underline{r};\underline{R})\chi_k(\underline{R})$$

Now multiply from the left with $\phi_{k'}(\underline{r};\underline{R})$ and integrate over $\underline{r}$.

$$\int d^3\underline{r} \sum_k \phi_{k'}(\underline{r};\underline{R})\left[T_n + H_e\right]\phi_k(\underline{r};\underline{R})\,\chi_k(\underline{R}) = E \int d^3\underline{r} \sum_k \phi_{k'}(\underline{r};\underline{R})\phi_k(\underline{r};\underline{R})\chi_k(\underline{R})$$

Some properties:

① Electronic wave functions are orthogonal
$\int d^3\underline{r}\, \phi_{k'}(\underline{r};\underline{R})\,\phi_k(\underline{r};\underline{R}) = \delta_{k'k}$

② Then:

$$\int d^3\underline{r} \sum_k \phi_{k'}(\underline{r};\underline{R})\, H_e\, \phi_k(\underline{r};\underline{R})$$

$$= \int d^3\underline{r} \sum_k E_k(\underline{R})\,\phi_{k'}(\underline{r};\underline{R})\phi_k(\underline{r};\underline{R})$$

$$= \sum_k E_k(\underline{R}) \int d^3\underline{r}\, \phi_{k'}(\underline{r};\underline{R})\phi_k(\underline{r};\underline{R})$$

$$= \sum_k E_k(\underline{R})\,\delta_{k'k} = E_{k'}(\underline{R})$$

Now insert this above:

$$\Rightarrow \int d^3\underline{r} \sum_k \chi_k(\underline{R})\,\phi_{k'}(\underline{r};\underline{R})\,T_i(\underline{R})\,\phi_k(\underline{r};\underline{R})$$

$$+ \int d^3\underline{r} \sum_k \phi_{k'}\,\phi_k\,T_i(\underline{R})\chi_k(\underline{R})$$

$$+ E_{k'}(\underline{R})\,\chi_{k'}(\underline{R})$$

$$= E \sum_k \delta_{k'k}\,\chi_k(\underline{R})$$

$$\Rightarrow \int d^3\underline{r} \sum_k \chi_k(\underline{R})\,\phi_{k'}\,T_i\,\phi_k$$

$$+ \sum_k \delta_{k'k}\,T_i(\underline{R})\chi_{k'}(\underline{R})$$

$$+ E_{k'}(\underline{R})\,\chi_{k'}(\underline{R})$$

$$= E\,\chi_{k'}(\underline{R})$$

$$\Rightarrow \int d^3\underline{r} \sum_k \chi_k(\underline{R})\,\phi_{k'}\,T_i\,\phi_k \quad (*)$$

$$+ \left[T_i(\underline{R}) + E_{k'}(\underline{R})\right]\chi_{k'}(\underline{R})$$

$$= E\,\chi_{k'}(\underline{R})$$

This term is called non-adiabatic coupling term.
When this term is neglected, we obtain the Born-Oppenheimer approximation:

① Equation for the ions:

$$\left[T_i(\underline{R}) + E_{k'}(\underline{R})\right]\chi_{k'}(\underline{R}) = E\,\chi_{k'}(\underline{R})$$

② Equation for the electrons:

$$\left[T_e + V_{ee} + V_{ei} + V_{ii}\right]\phi_k(\underline{r};\underline{R}) = E_k(\underline{R})\,\phi_k(\underline{r};\underline{R})$$

2.1 Molecular Hamiltonian

$$\hat{H} = \hat{T}^i(\underline{\boldsymbol{R}}) + \hat{T}^e(\underline{\boldsymbol{r}}) + \hat{V}^{ii}(\underline{\boldsymbol{R}}) + \hat{V}^{ei}(\underline{\boldsymbol{r}}, \underline{\boldsymbol{R}}) + \hat{V}^{ee}(\underline{\boldsymbol{r}})$$

Kinetic energy of ions

$$\hat{T}^i(\underline{\mathbf{R}}) = \sum_{\alpha}^{N_i} -\frac{\nabla_\alpha^2}{2M_\alpha}$$

Kinetic energy of electrons

$$\hat{T}^e(\underline{\mathbf{r}}) = \sum_{j}^{N_e} -\frac{\nabla_j^2}{2}$$

Interaction between ions

$$\hat{V}^{ii}(\underline{\mathbf{R}}) = \sum_{\alpha}^{N_i} \sum_{\beta \neq \alpha}^{N_i} \frac{Z_\alpha Z_\beta}{2|\mathbf{R}_\alpha - \mathbf{R}_\beta|}$$

Interaction between electrons

$$\hat{V}^{ee}(\underline{\mathbf{r}}) = \sum_{j}^{N_e} \sum_{k \neq j}^{N_e} \frac{1}{2|\mathbf{r}_j - \mathbf{r}_k|}$$

Interaction between electrons and ions

$$\hat{V}^{ei}(\underline{\mathbf{r}}; \underline{\mathbf{R}}) = -\sum_{j}^{N_e} \sum_{\alpha}^{N_i} \frac{Z_\alpha}{|\mathbf{r}_j - \mathbf{R}_\alpha|}$$

Attila Cangi | a.cangi@hzdr.de

2.2 Schrödinger Equation of Coupled Electrons and Ions

$$\hat{H} = \hat{T}^i(\underline{\boldsymbol{R}}) + \hat{T}^e(\underline{\boldsymbol{r}}) + \hat{V}^{ii}(\underline{\boldsymbol{R}}) + \hat{V}^{ei}(\underline{\boldsymbol{r}}, \underline{\boldsymbol{R}}) + \hat{V}^{ee}(\underline{\boldsymbol{r}})$$

$$\hat{H}\Psi(\underline{\mathbf{r}}, \underline{\mathbf{R}}) = E\,\Psi(\underline{\mathbf{r}}, \underline{\mathbf{R}})$$

- Solving the time-independent Schrödinger equation for the fully coupled problem is computationally extremely challenging.
- Reducing the complexity of the electron-ion Hamiltonian is possible by separating ionic and electronic degrees of freedom.

➜ **Born-Oppenheimer approximation**

2.3 Born-Oppenheimer Approximation

- Separate ionic and electronic degrees of freedom with a separation ansatz (remember three-dimensional particle in a box or harmonic oscillator problem)

$$\Psi(\underline{\mathbf{r}}, \underline{\mathbf{R}}) = \Phi(\underline{\mathbf{r}}; \underline{\mathbf{R}})\, \chi(\underline{\mathbf{R}})$$

- Obtain two Schrödinger equations. One for the electrons:

$$\left[\hat{T}^e(\underline{\mathbf{r}}) + \hat{V}^{ei}(\underline{\mathbf{r}}; \underline{\mathbf{R}}) + \hat{V}^{ei}(\underline{\mathbf{r}}; \underline{\mathbf{R}}) + \hat{V}^{ii}(\underline{\mathbf{R}})\right] \Phi(\underline{\mathbf{r}}; \underline{\mathbf{R}}) = E_n(\underline{\mathbf{R}}) \Phi(\underline{\mathbf{r}}; \underline{\mathbf{R}})$$

- and one for the ions:

$$\left[\hat{T}^i(\underline{\mathbf{R}}) + E_n(\underline{\mathbf{R}})\right] \chi(\underline{\mathbf{R}}) = E \chi(\underline{\mathbf{R}})$$

- Solving these coupled equations is feasible but still computationally demanding.

2.3 Born-Oppenheimer Approximation

- In a final simplification, we consider the ions as classical point-like particles.
- The dynamics of the ions follows Newtonian equations of motion:

$$M_\alpha \frac{\partial^2 \mathbf{R}_\alpha}{\partial t^2} = -\nabla_{\mathbf{R}_\alpha} E_n(\underline{\mathbf{R}}) = \boldsymbol{F}_\alpha$$

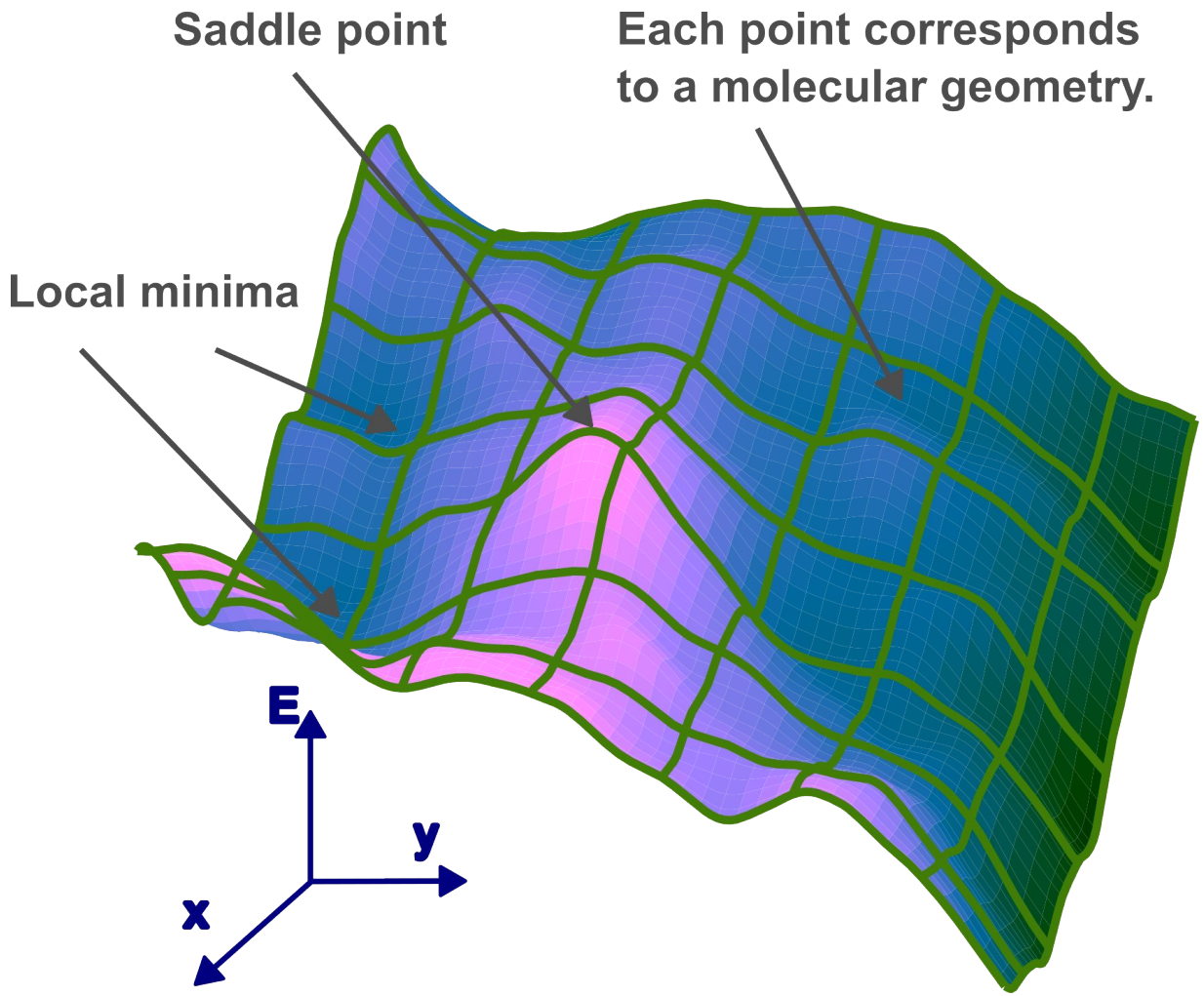
- The forces on the ions are obtained from the gradient of the eigenvalue from the electronic Schrödinger equation:

$$\left[\hat{T}^e(\underline{\mathbf{r}}) + \hat{V}^{ei}(\underline{\mathbf{r}};\underline{\mathbf{R}}) + \hat{V}^{ei}(\underline{\mathbf{r}};\underline{\mathbf{R}}) + \hat{V}^{ii}(\underline{\mathbf{R}})\right] \Phi(\underline{\mathbf{r}};\underline{\mathbf{R}}) = E_n(\underline{\mathbf{R}})\Phi(\underline{\mathbf{r}};\underline{\mathbf{R}})$$

- The eigenvalue of the electronic Schrödinger equation is a potential energy surface on which the ionic dynamics takes place.

- Remarks:
  - The separation of electronic and ionic degrees of freedom is feasible because the inverse dependence of the kinetic energy on the particle mass results in a far smaller kinetic energy for the ions than for the electrons.
  - This approximation is valid as long as the motion of the ions happens on a much larger time scale than the motion of the electrons.

2.4 Born-Oppenheimer Potential Energy Surface



2.5 Classical Interatomic Potentials

Overview of interatomic potentials (force fields)

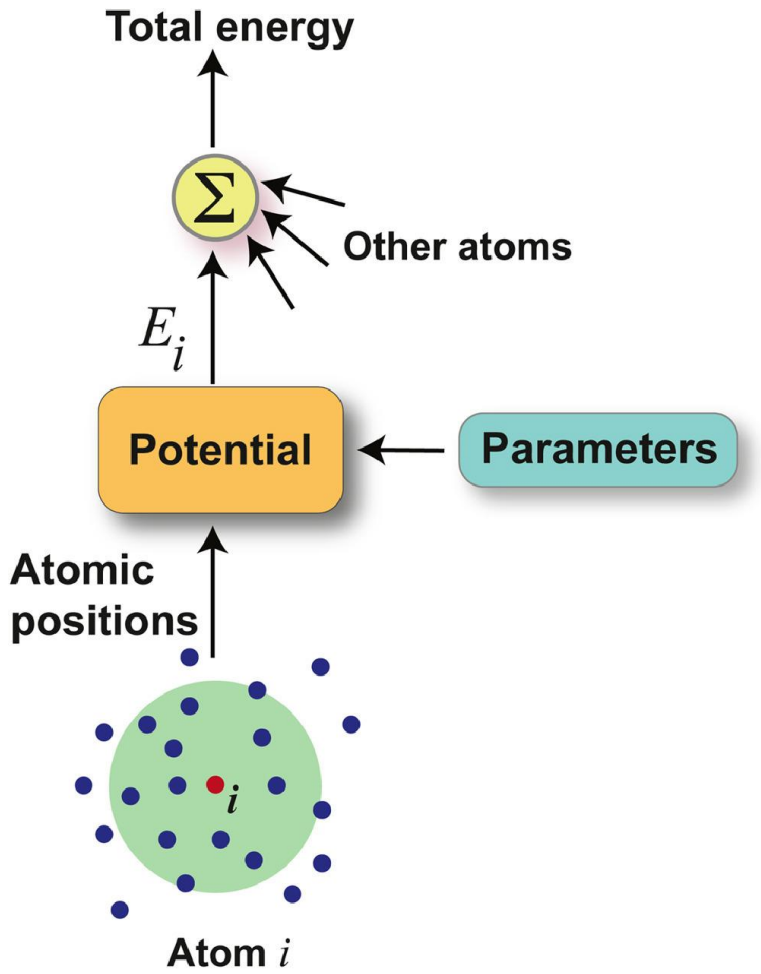
Pair potentials	Many-body potentials	Machine learning potentials
Hard-sphere potential	EAM potentials	More details later…
Buckingham potential	MEAM potentials	
Morse potential	Bond order potentials	
Lennard-Jones potential	Tersoff potentials	
CHARMM potentials	ReaxFF potentials	
...	…	

2.5 Classical Interatomic Potentials

Workflow of classical interatomic potentials

Fitting the PES with traditional interatomic potentials:

- Energy of an atom is computed using atomic coordinates within the cutoff sphere (green)
- With fixed values of the potential parameters
- The atomic energies of all atoms of the system are summed up to obtain the total energy.



Y. Mishin, Acta Materialia 214, 116980 (2021).

2.5 Classical Interatomic Potentials

Pair potentials

Pair potentials parametrize the PES in terms of pair-wise (two-body) interactions:

$$E = V(r), \quad r = |\mathbf{r}_i - \mathbf{r}_j|$$

- 😊 Simple and analytical mathematical functions
- 😊 Efficient to evaluate
- 😊 Run much faster than quantum mechanical calculations (DFT)

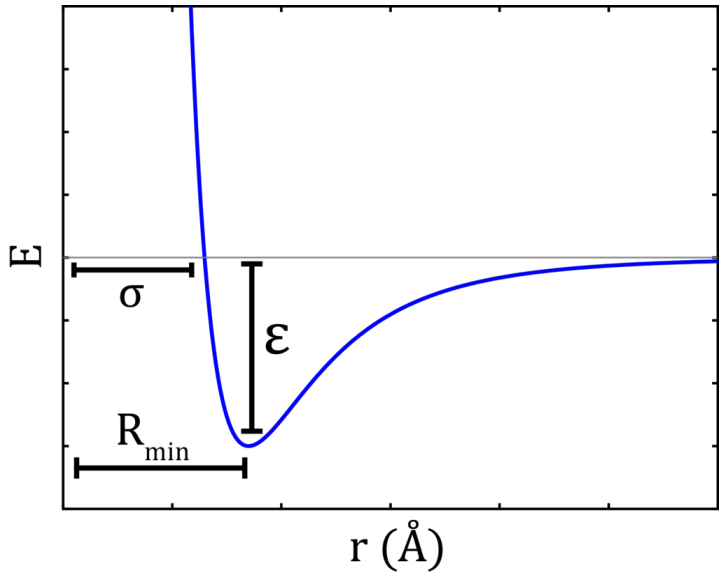
- 😕 How do you find an accurate functional form?
- 😕 Parameters need to be determined empirically
- 😕 Are not as transferable than more complex potentials

2.5 Classical Interatomic Potentials

Simple examples of pair potentials

Lennard-Jones potential

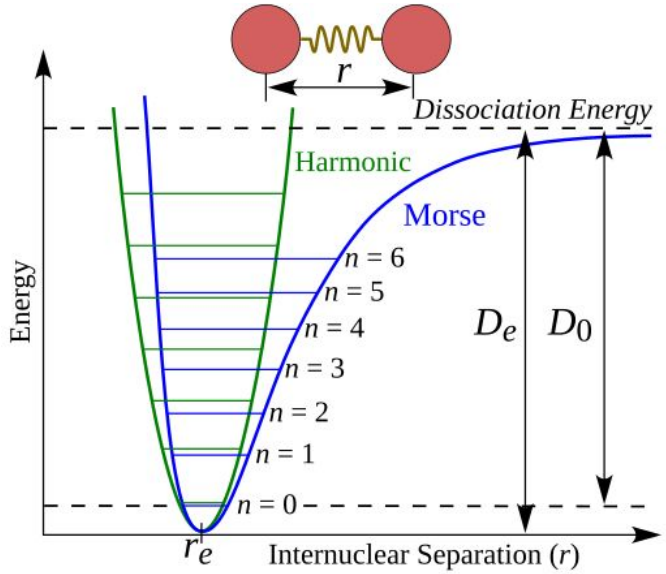
$$V(r) = 4\epsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right]$$



Christophe Rowley, Wikimedia Commons.

Morse potential

$$V(r) = D_e \left[1 - e^{-a(r-r_e)} \right]^2$$



Mark Somoza March, Wikimedia Commons.

2.5 Classical Interatomic Potentials

More complicated pair potentials

Classical interatomic potentials parametrize the PES in terms of structural, vibrational, and other energy-based models from first-principles calculations (DFT).

CHARMM22 potential:

$$
\begin{aligned}
V(r) = & \sum_{\substack{b \\ \text{"bonds"}}} k_b(b-b_0)^2 + \sum_{\substack{\theta \\ \text{"angles"}}} k_\theta(\theta-\theta_0)^2 + \sum_{\substack{\phi \\ \text{"dihedrals"}}} k_\phi[1+\cos(n\phi-\delta)] \\
& + \sum_{\substack{\omega \\ \text{"impropers"}}} k_\omega(\omega-\omega_0)^2 + \sum_{\substack{u \\ \text{"Urey-Bradley"}}} k_u(u-u_0)^2 \\
& + \sum_{\substack{ij \\ \text{"nonbonded"}}} \left(\epsilon_{ij} \left[\left(\frac{R_{ij}}{r_{ij}} \right)^{12} - 2 \left(\frac{R_{ij}}{r_{ij}} \right)^{6} \right] + \frac{q_i q_j}{\epsilon_r r_{ij}} \right)
\end{aligned}
$$

Difficulty of determining parameters:

"An automated least-squares procedure often leads to a combination of "unphysical" parameters that reproduce the input data. More meaningful parameter values, which have a wider range of applicability, were obtained manually with "reasonable" parameter ranges for the optimization in the iterative refinement procedure …"

A. D. MacKerell et al., J. Phys. Chem. B 102 (18), 3586 (1998).

3. Machine Learning Potentials

3.1 Introduction to Machine Learning

3.2 Neural Networks

3.3 Overview of Machine Learning Potentials

3.4 Data and Descriptors

3.5 Examples of Machine Learning Potentials

Attila Cangi | a.cangi@hzdr.de

3.1 Introduction to Machine Learning

What is machine learning?

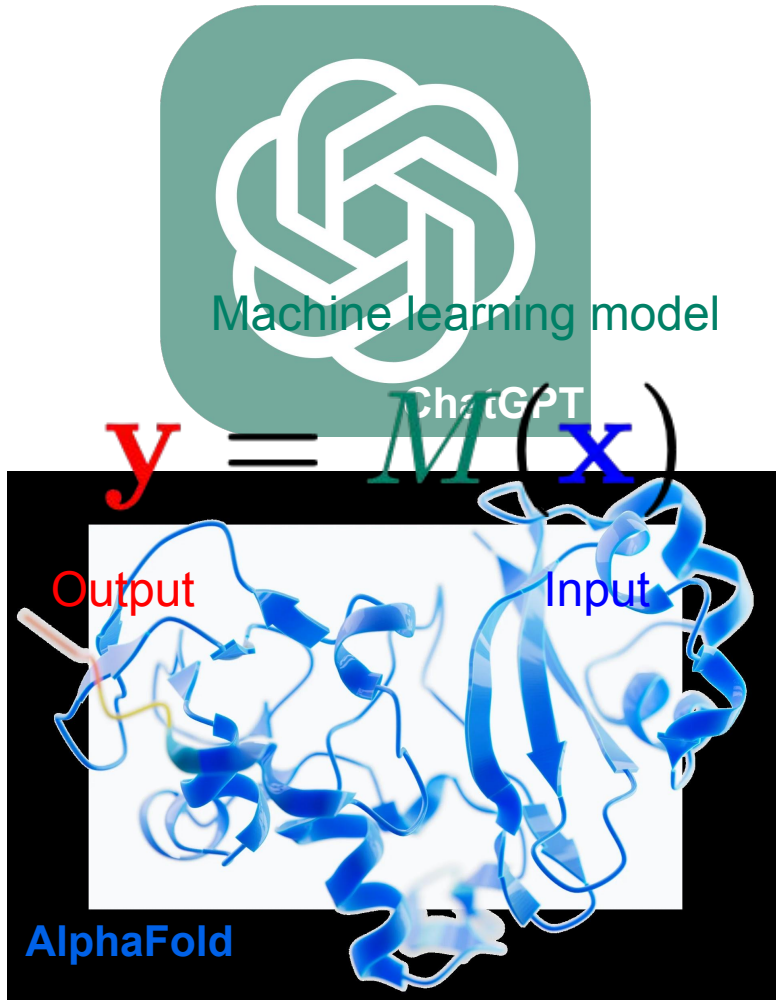
“The use and development of computer systems that are able to learn and adapt without following explicit instructions, by using algorithms and statistical models to analyse and draw inferences from patterns in data.”

This process often involves three main steps

- Input data
- Training
- Prediction

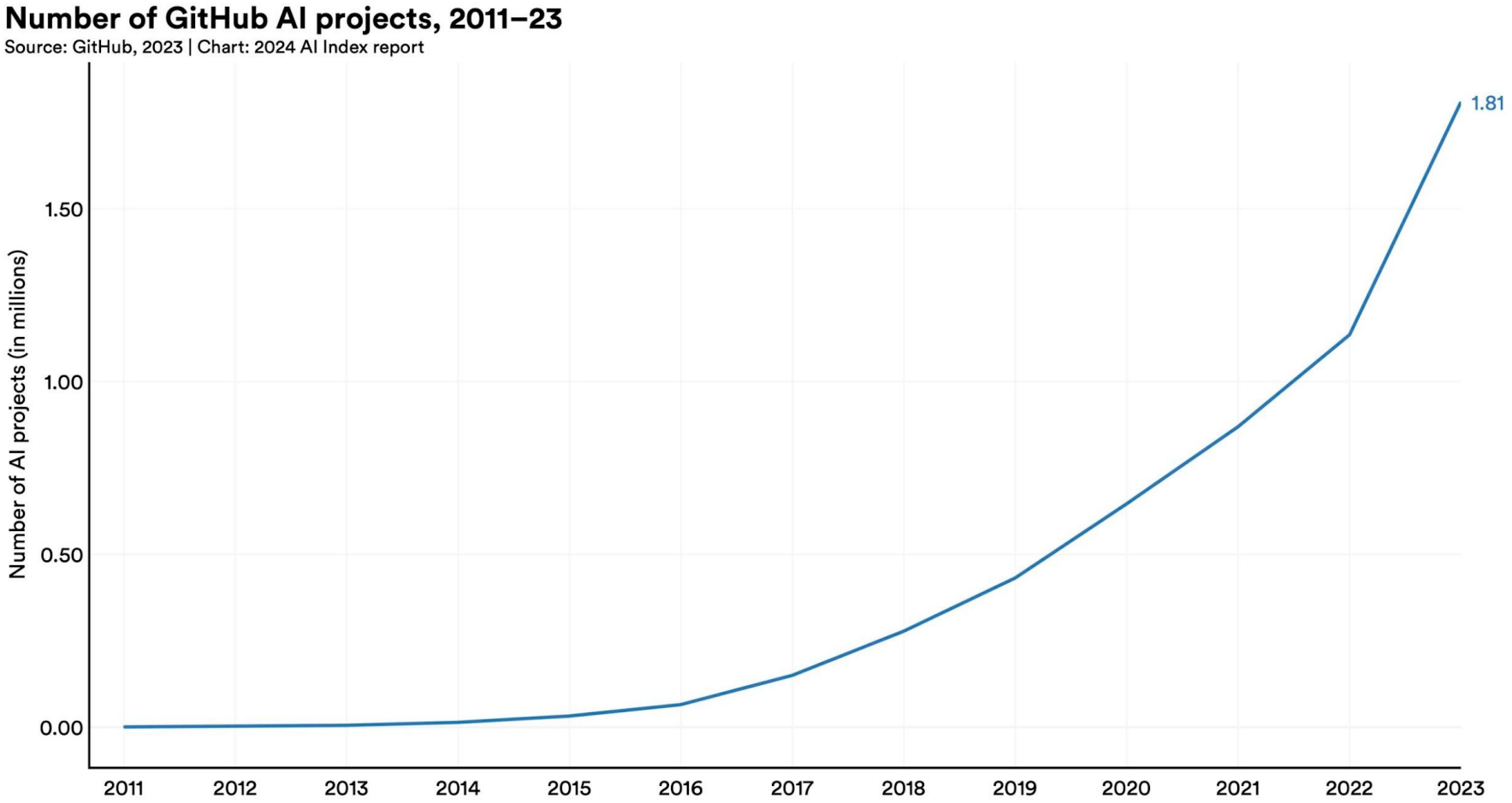
Key categories of machine learning

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning



3.1 Introduction to Machine Learning

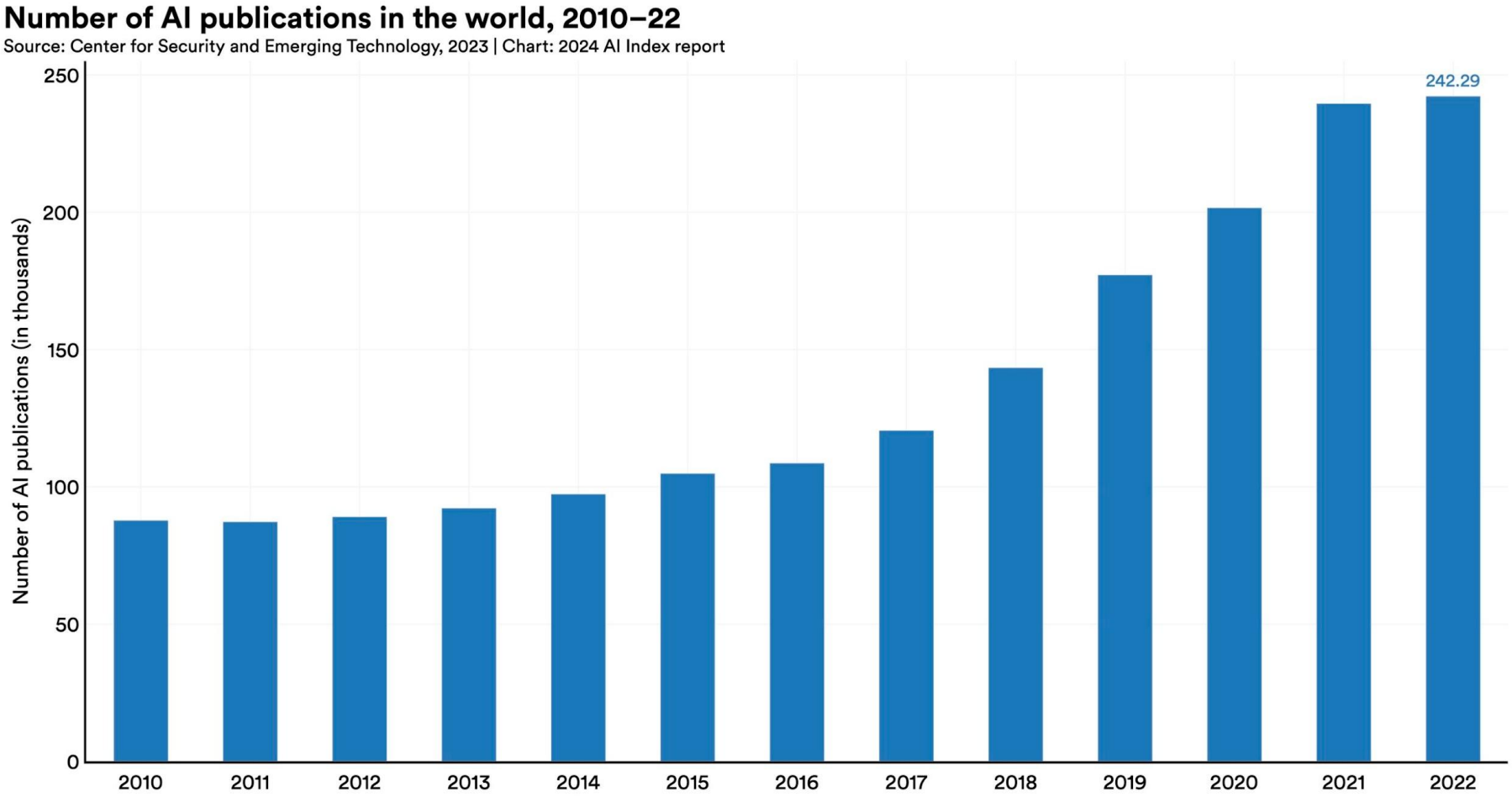
Machine learning trends



AI Index Report 2023, https://aiindex.stanford.edu/report/.

3.1 Introduction to Machine Learning

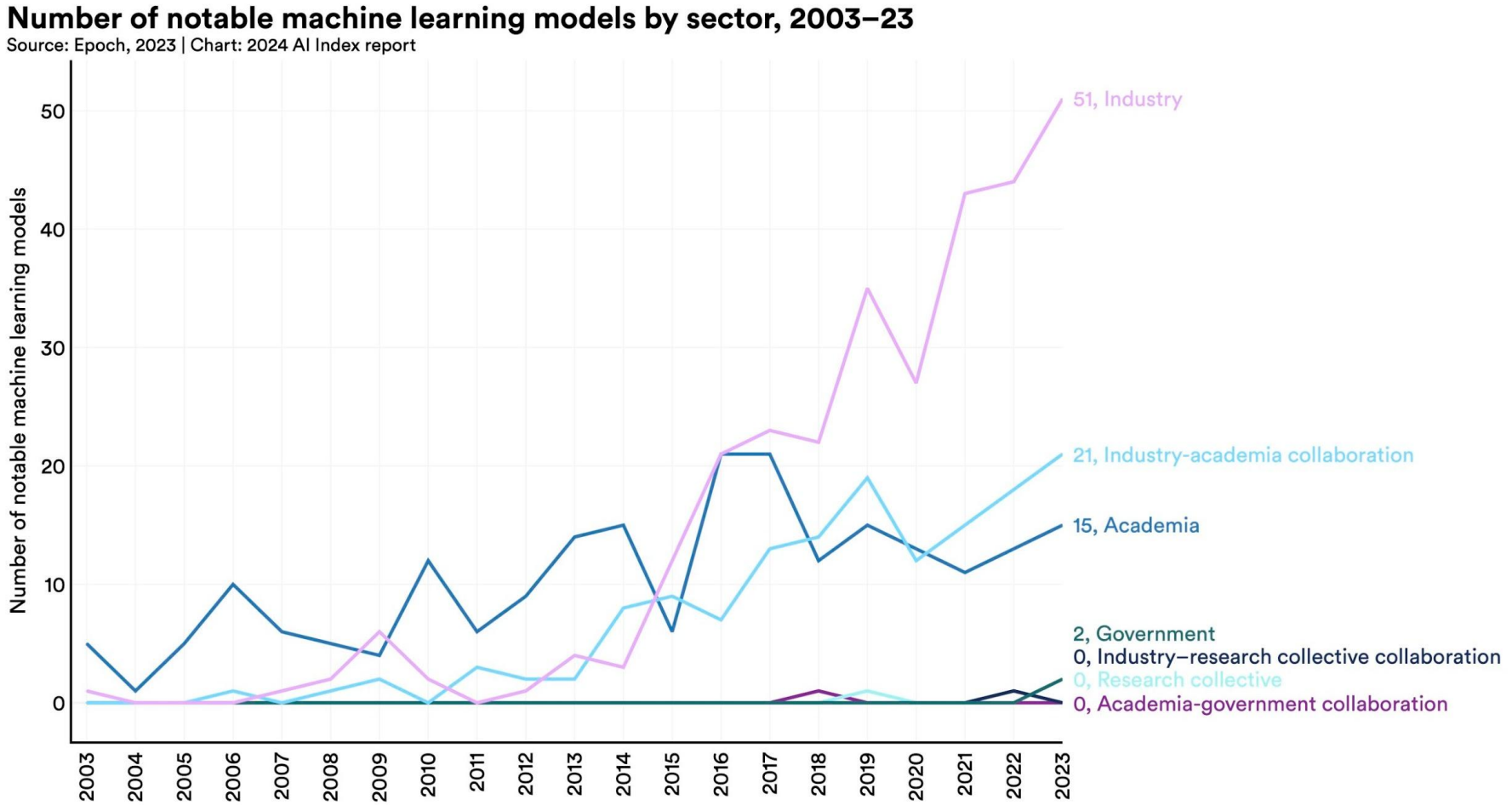
Machine learning trends



AI Index Report 2023, https://aiindex.stanford.edu/report/.

3.1 Introduction to Machine Learning

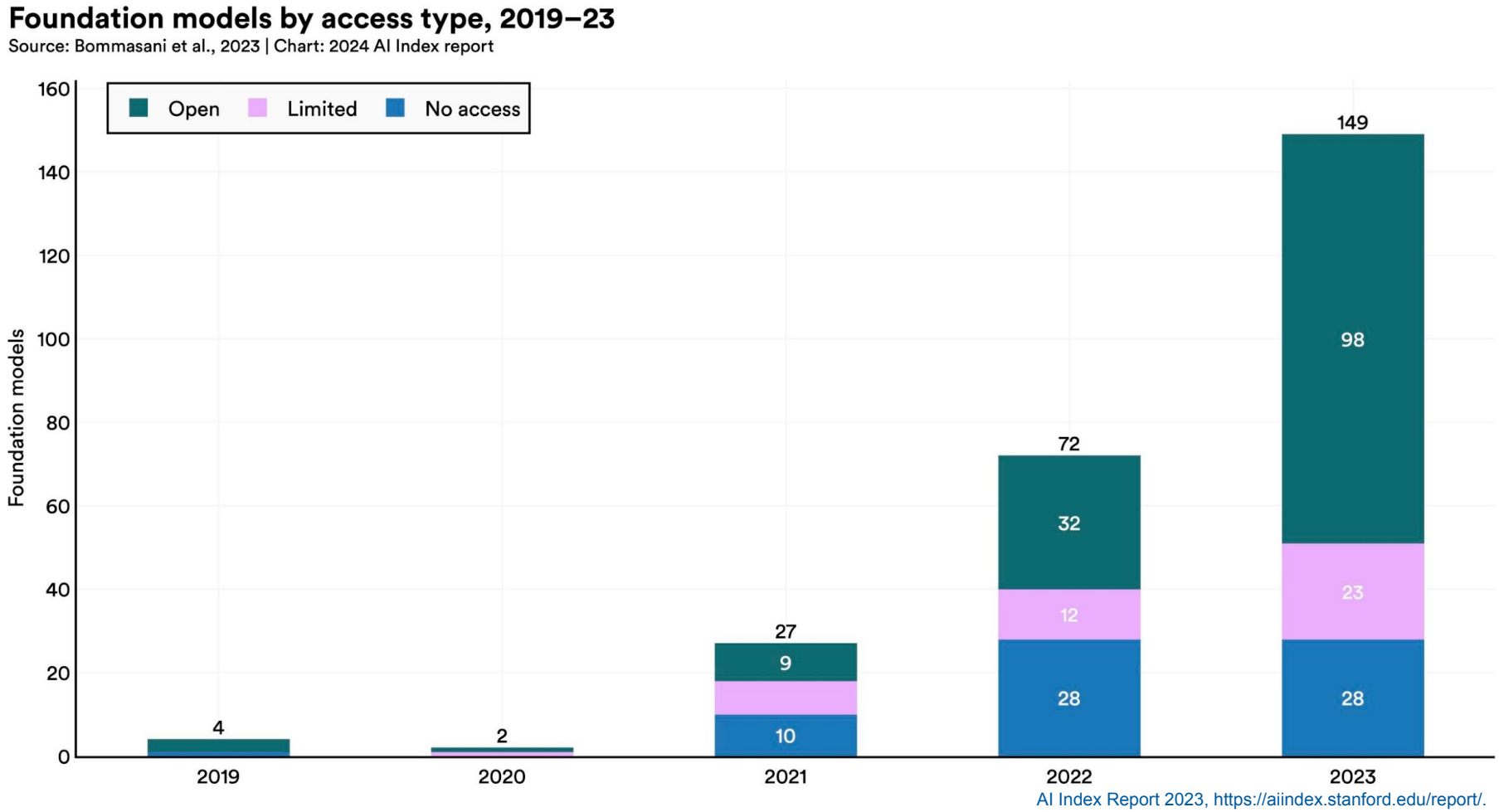
Machine learning trends



AI Index Report 2023, https://aiindex.stanford.edu/report/.

3.1 Introduction to Machine Learning

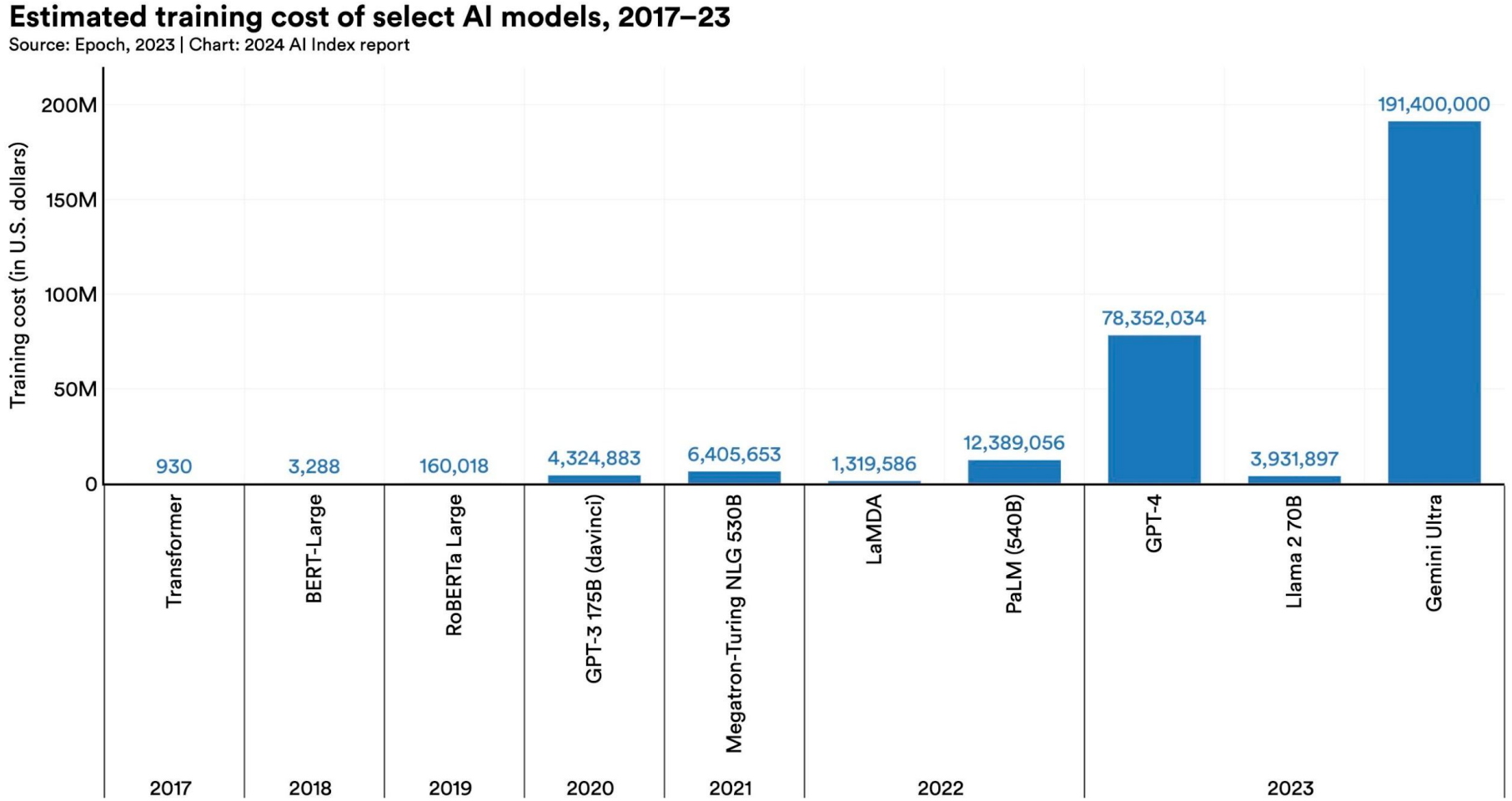
Machine learning trends



AI Index Report 2023, https://aiindex.stanford.edu/report/.

3.1 Introduction to Machine Learning

Machine learning trends



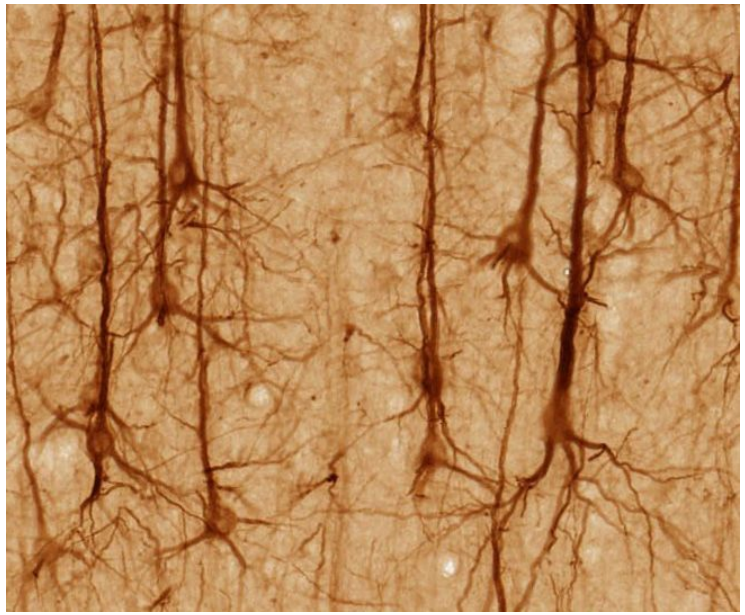
AI Index Report 2023, https://aiindex.stanford.edu/report/.

3.2 Neural Networks

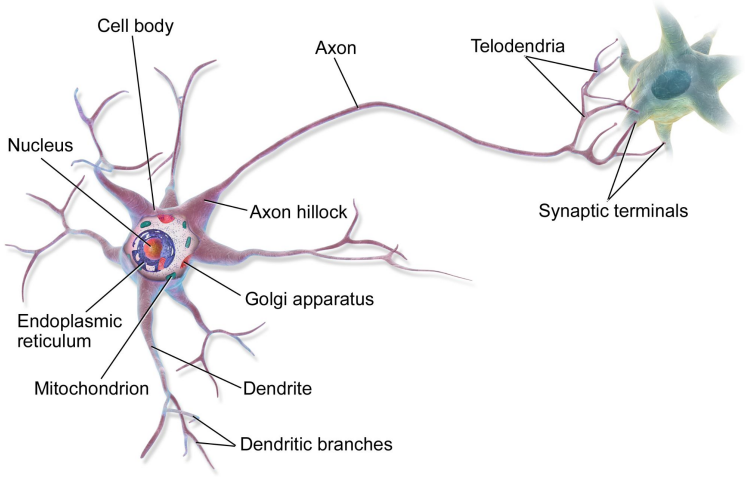
Biological neurons

A neuron, also known as a nerve cell, is the basic functional unit of the nervous system. Neurons are specialized cells that receive, process and transmit information through electrical and chemical signals.
A neuron is made up of three main parts:

- **Cell body (soma)**
  Contains the nucleus and other organelles necessary for basic cell functions.
- **Dendrites**
  Tree-like projections that receive signals from other nerve cells and transmit them to the cell body.
- **Axon**
  A long, thin extension that conducts electrical signals away from the cell body to other nerve cells, muscles or glands. The end of the axon branches into axon terminals that release neurotransmitters to transmit signals.



Neurons in the cerebral cortex.



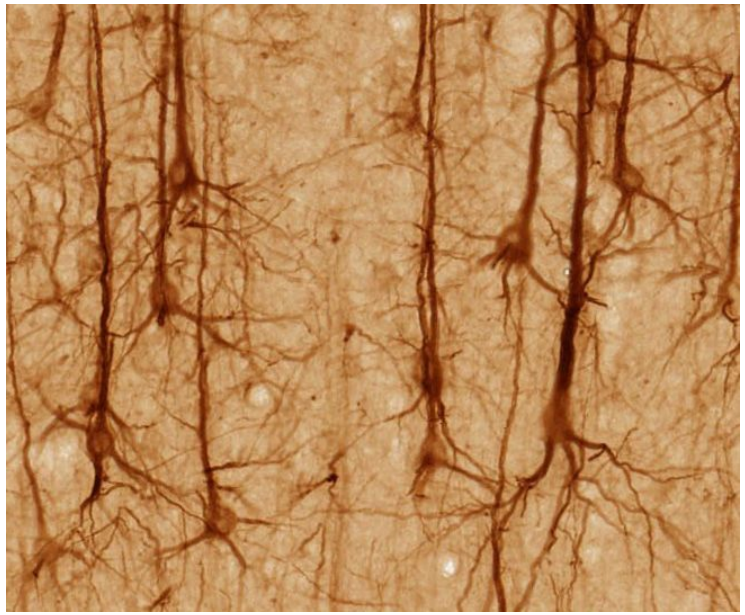
3.2 Neural Networks

Biological neurons

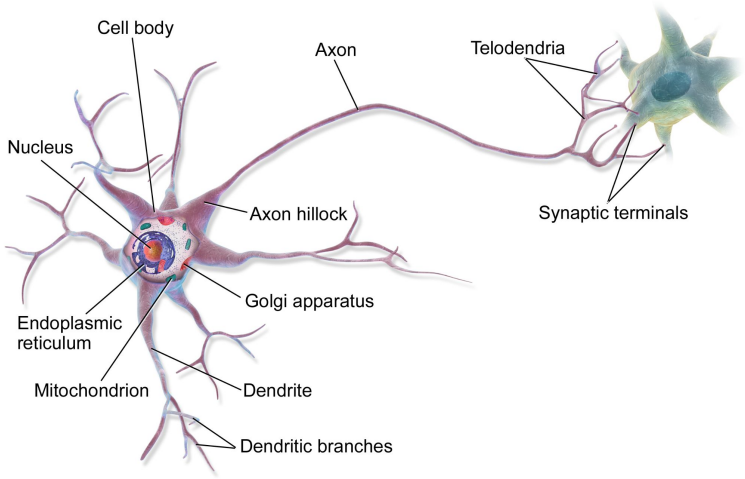
The function of a neuron can be divided into several steps:

- **Reception of signals**
  Dendrites receive chemical signals from neighboring neurons. These signals lead to changes in the membrane potential of the neuron.
- **Generation of an action potential**
  When the membrane potential reaches a certain threshold, an action potential is triggered. This is a rapid change in the membrane potential that spreads along the axon.
- **Transmission of the action potential**
  The action potential travels along the axon to the axon terminals. This occurs through a series of depolarizations and repolarizations of the cell membrane.

This variability of information transmission is reflected in the weights of artificial neural networks.

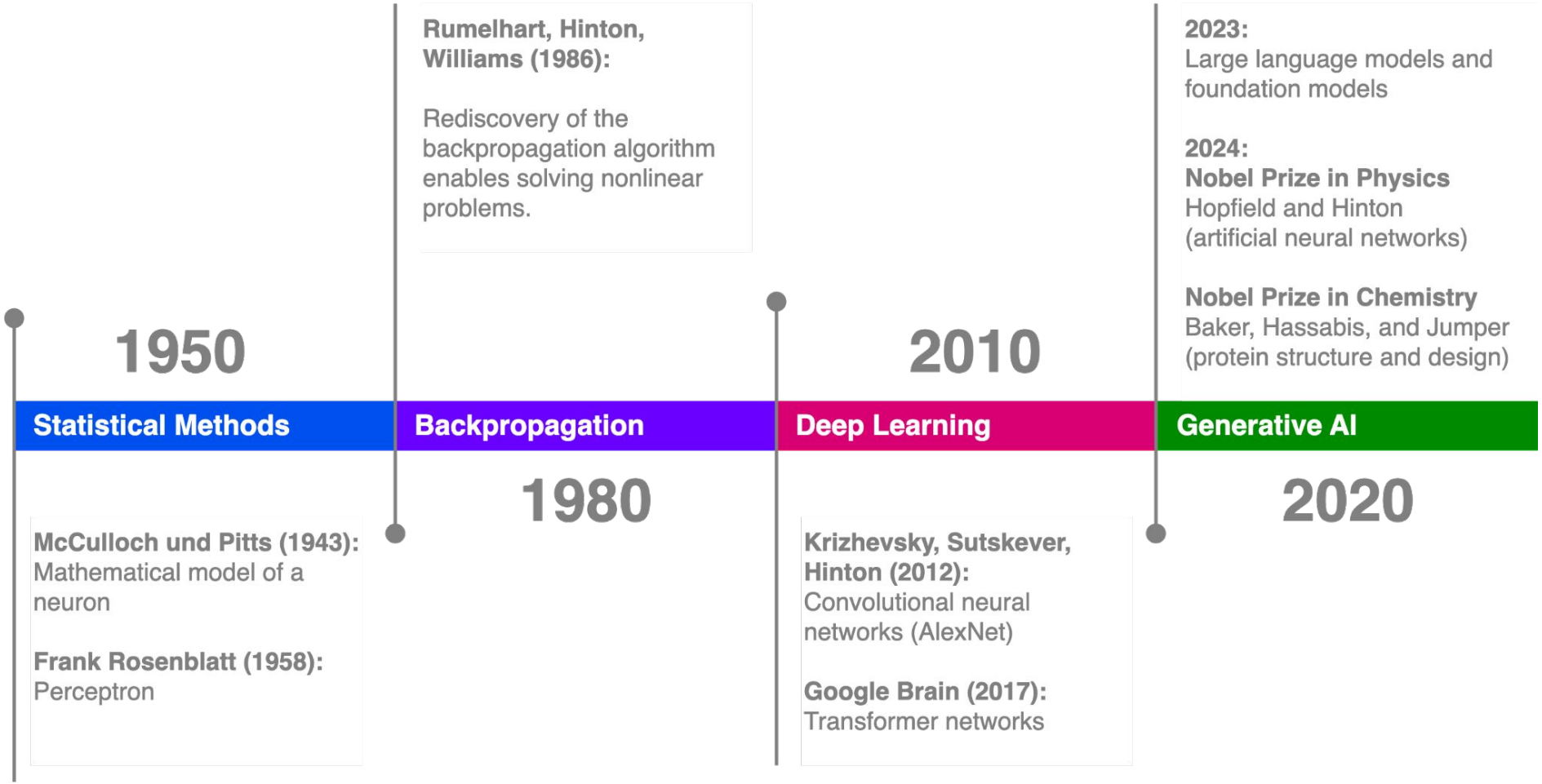


Neurons in the cerebral cortex.



3.2 Neural Networks

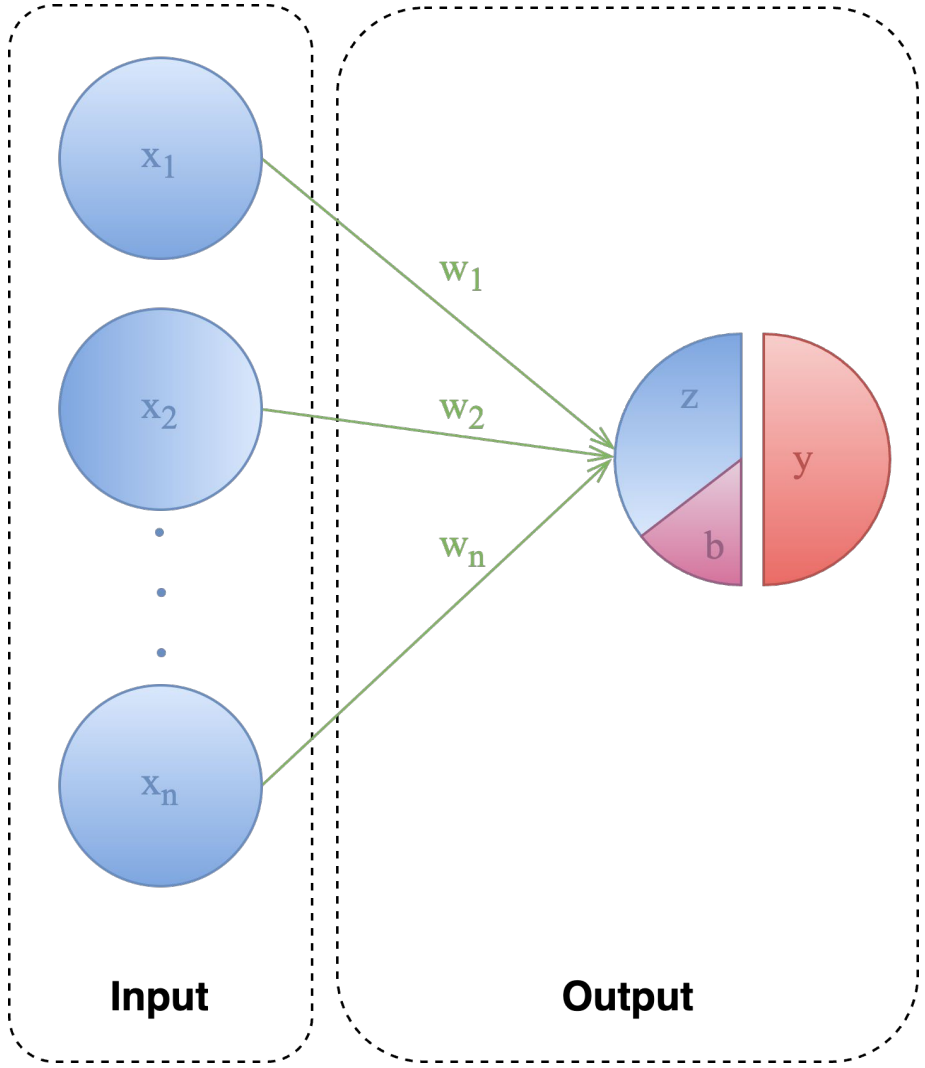
Short history of artificial neural networks



3.2 Neural Networks

Perceptron: Structure

- The simple perceptron model consists of an input layer and an output layer.
- In the input layer, the input is $(x_1, x_2, ..., x_n)$.
- The output layer consists of a single neuron. It contains the network input (z) and the output value (y).
- This network can be used for binary classification, i.e. the network can decide for an input whether it belongs to a certain category.
- The classification is expressed by the output value (y).



3.2 Neural Networks

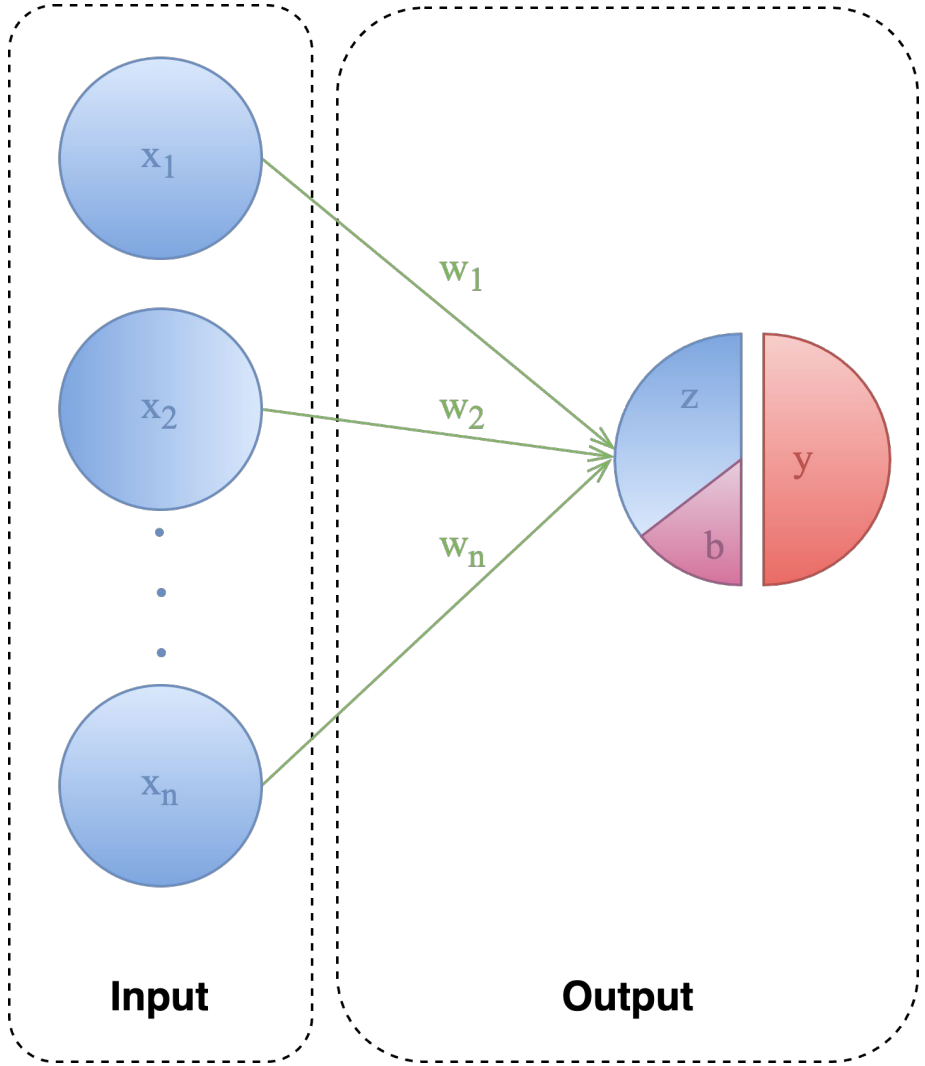
Perceptron: Forward propagation

Input

$$\vec{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}, \quad n \in \mathbb{R}, \quad x_i \in \mathbb{R}$$

Weights

$$\vec{w} = \begin{pmatrix} w_1 \\ \vdots \\ w_n \end{pmatrix}, \quad w_i \in$$

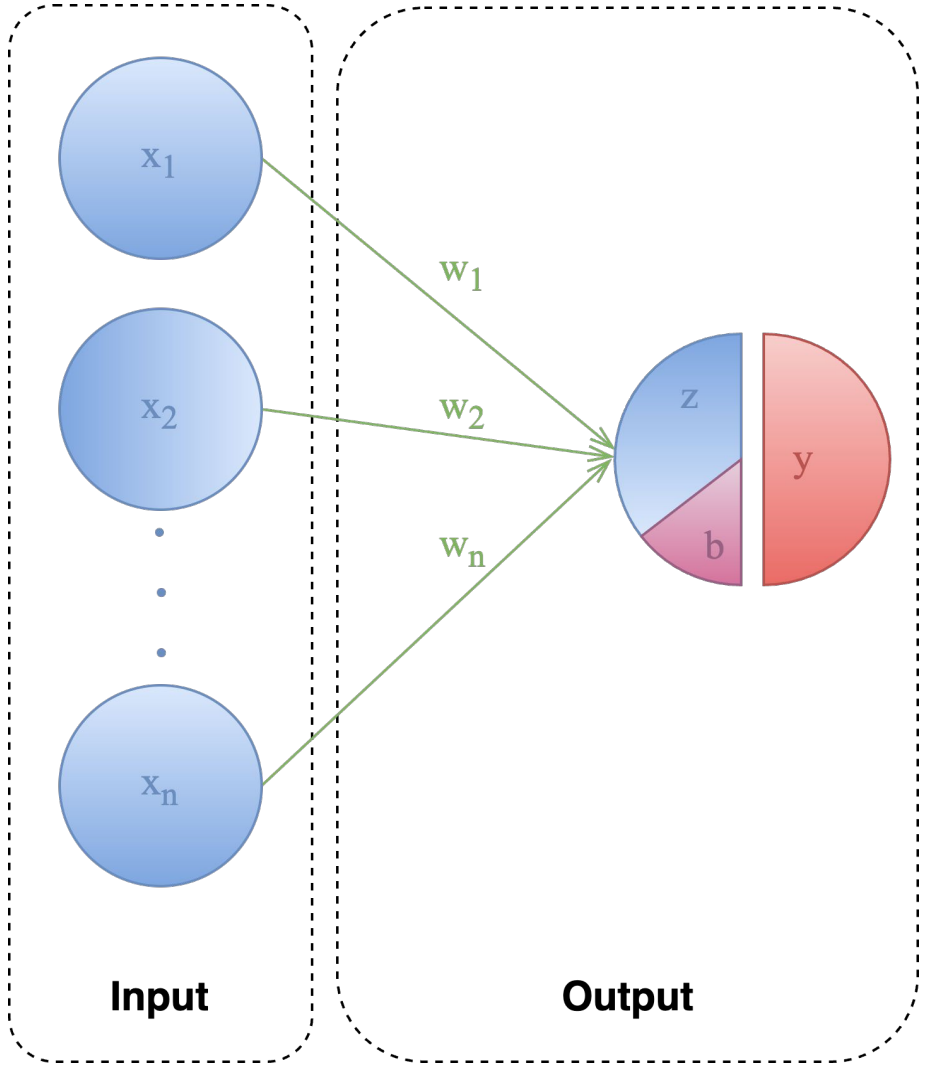


3.2 Neural Networks

Perceptron: Forward propagation

Net input

$$
\begin{aligned}
z &= w_1 x_1 + \cdots + w_n x_n \\
&= \vec{w}^T \cdot \vec{x} \\
&= (w_1, \ldots, w_n) \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}
\end{aligned}
$$



3.2 Neural Networks

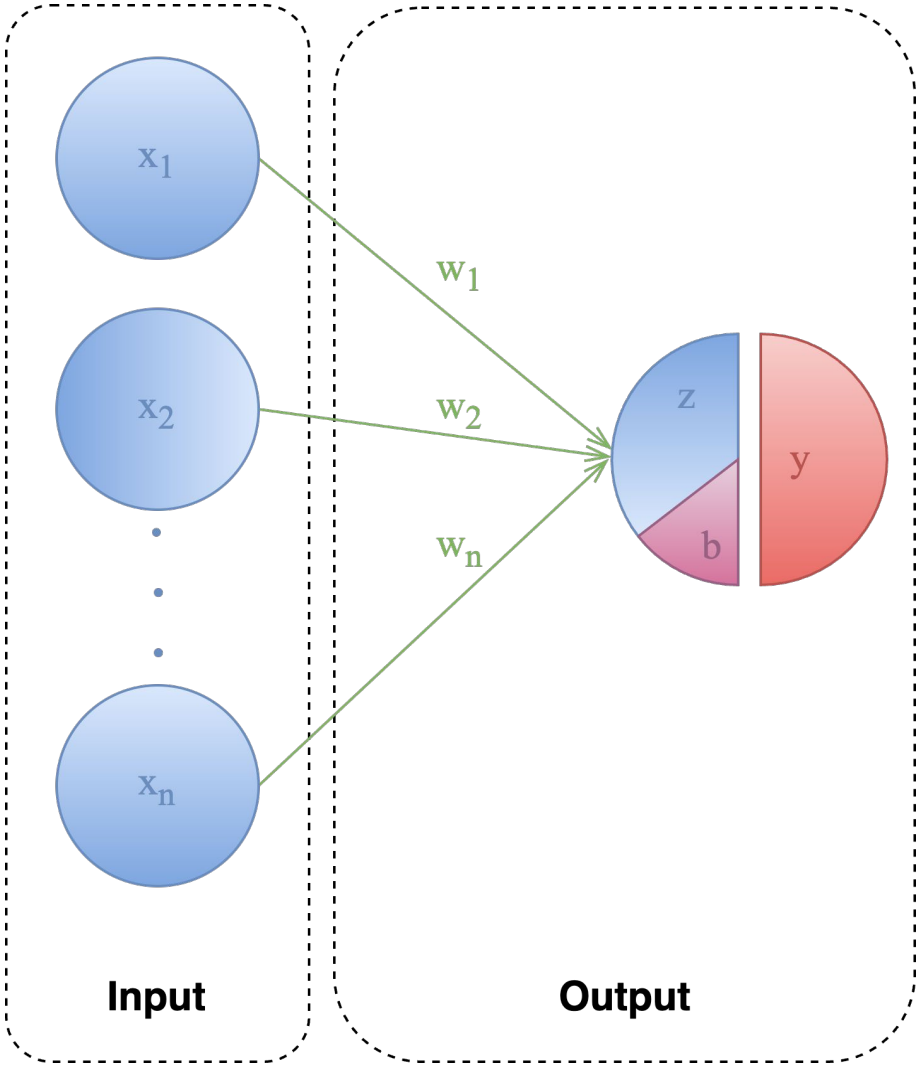
Perceptron: Forward propagation

Activation and output

In the second and final step, we calculate the activation of the output neuron, which also corresponds to the output of the perceptron model.

An activation function is applied to the network input:

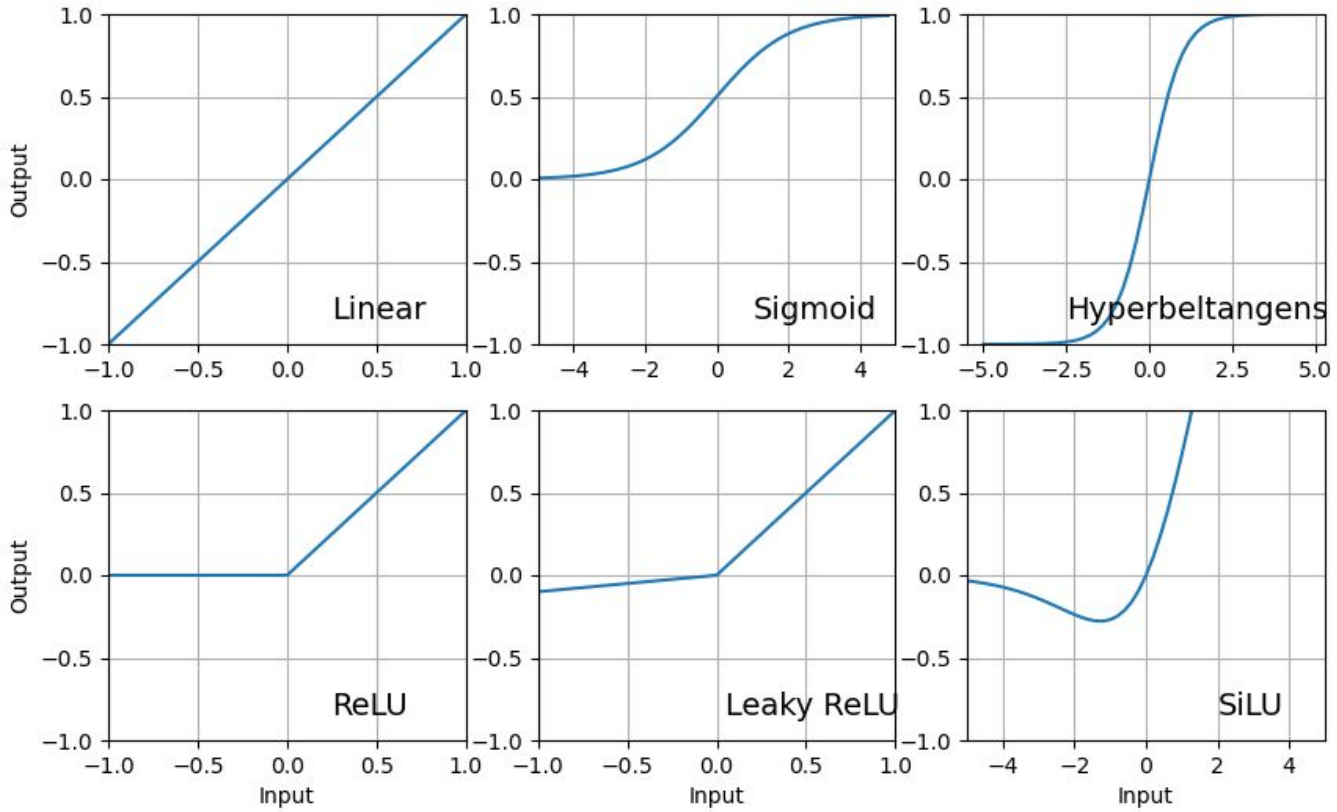
$$y = g(z + b)$$



3.2 Neural Networks

Perceptron: Forward propagation

Examples of Activation Functions



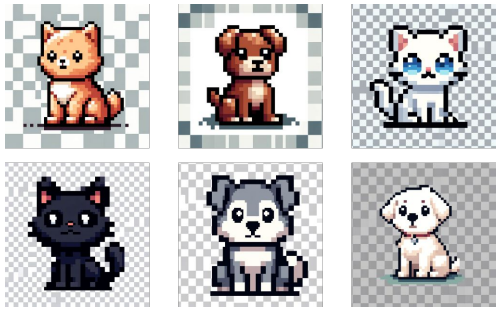
3.2 Neural Networks

Perceptron: Learning process

In the context of the perceptron model, we understand learning as the **gradual adaptation of the weights to the desired target function** with the help of training data.

Labeled Data

Training data is, for example, a series of data with a label.



Data labeled as cat and non-cat.

The training data can therefore be written as pairs of feature vectors and labels:

$$(\vec{x}^k, y^k) \quad k \in \{1, \ldots, N\}$$

With a neural network, we must distinguish between the calculated output of the current network and the correct output of a training example.

3.2 Neural Networks

Perceptron: Learning process

Learning Step

Learning means that we calculate the output for each training example and then adjust the weights. This is called a learning step.
We can therefore adjust the weight vector for the pairs of feature vectors and labels in each learning step by adding a change of all weights to the current value:

$$w_i := w_i + \Delta w_i$$

We will define a concrete rule for adjusting the weights in a moment. First let's look at the properties of the weight update:

- If the calculated output is greater than the reference value, the weight update should be negative, i.e. the weight of this neuron should be weakened.
- If the calculated output is smaller than the reference value, the weight update should be positive, i.e. the weight is increased.
- If the calculated output and the reference value are the same, no weighting update should take place.

3.2 Neural Networks

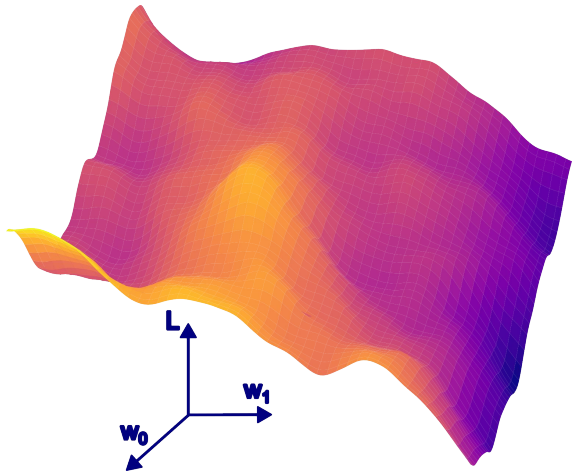
Perceptron: Gradient descent

Loss Function

In order to derive a suitable learning rule, we must first define the term loss function:

$$L(w) = \frac{1}{2N} \sum_{k=1}^{N} \left[y^k - \bar{y}^k \right]^2$$

It is a multidimensional function of the weights or the weight vector

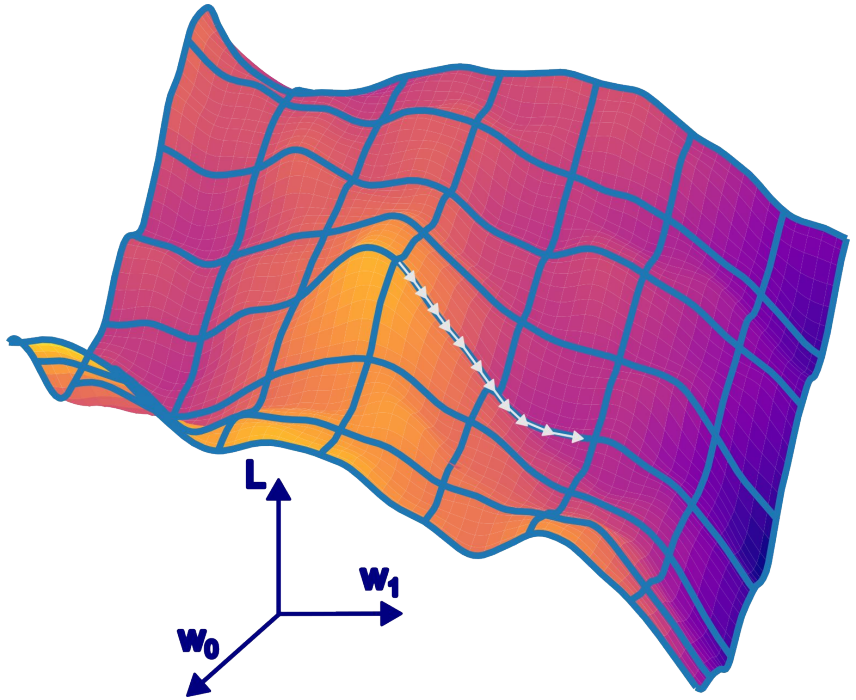


3.2 Neural Networks

Perceptron: Gradient descent

Learning Rule: Minimum of the Loss Landscape

We can now use the loss function to derive a learning rule by setting ourselves the goal of minimizing the value of the loss function. This means that we look for valleys in the parameter landscape.



3.2 Neural Networks

Perceptron: Weight update

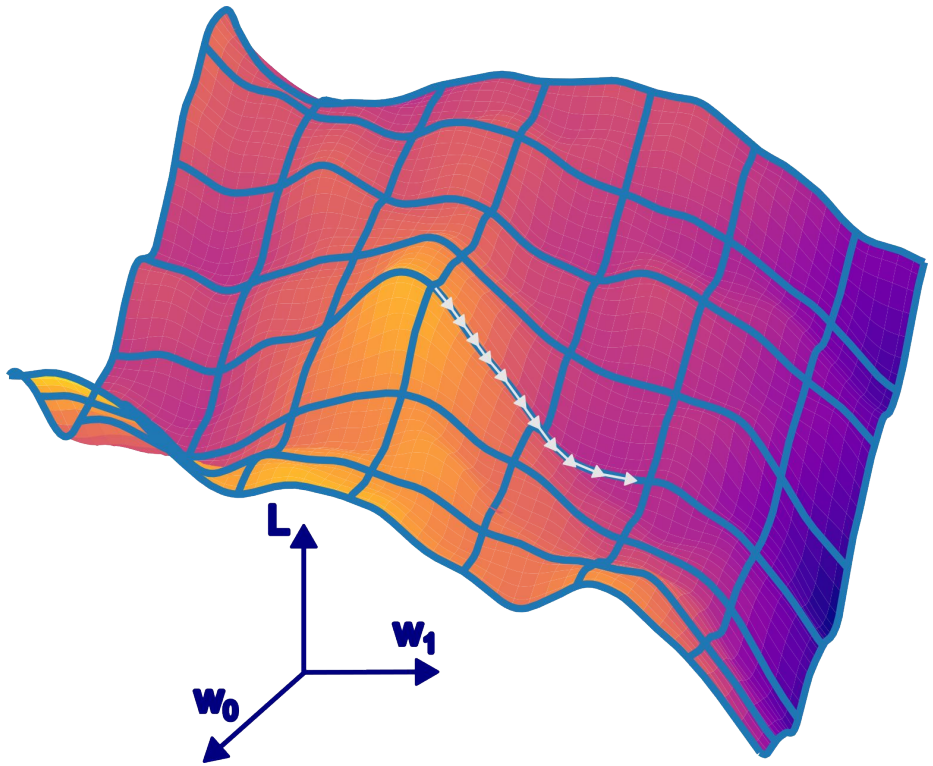
Use the gradient of the loss landscape, which informs us about the largest increase in the error landscape. We therefore take the negative gradient and thus obtain our learning rule:

$$\Delta\vec{w} = -\alpha \nabla L(w)$$

$$= -\alpha \begin{pmatrix} \frac{\partial L(w)}{\partial w_1} \\ \vdots \\ \frac{\partial L(w)}{\partial w_n} \end{pmatrix}$$

where we have also introduced the learning rate.

As can be seen in the figure, updating the weights in the direction of the negative gradient causes us to run in the direction of the minimum of the loss landscape.

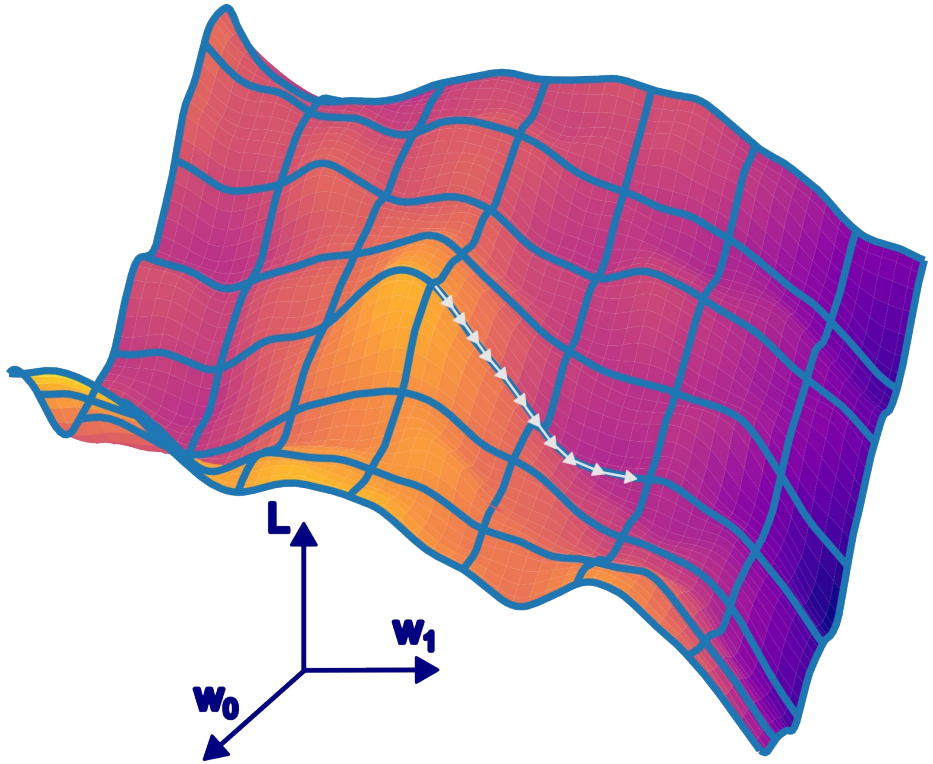


3.2 Neural Networks

Perceptron: Weight update

Rewrite the loss terms, so we can take a derivative with respect to the weights:

$$
\begin{aligned}
L(w) &= \frac{1}{2N}\sum_{k=1}^{N}\left[y^k - \bar{y}^k\right]^2 \\
&= \frac{1}{2N}\sum_{k=1}^{N}\left[y^k - g(z^k)\right]^2 \\
&= \frac{1}{2N}\sum_{k=1}^{N}\left(y^k - z^k\right)^2 \\
&= \frac{1}{2N}\sum_{k=1}^{N}\left(y^k - \vec{w}^T\vec{x}^k\right)^2 \quad .
\end{aligned}
$$

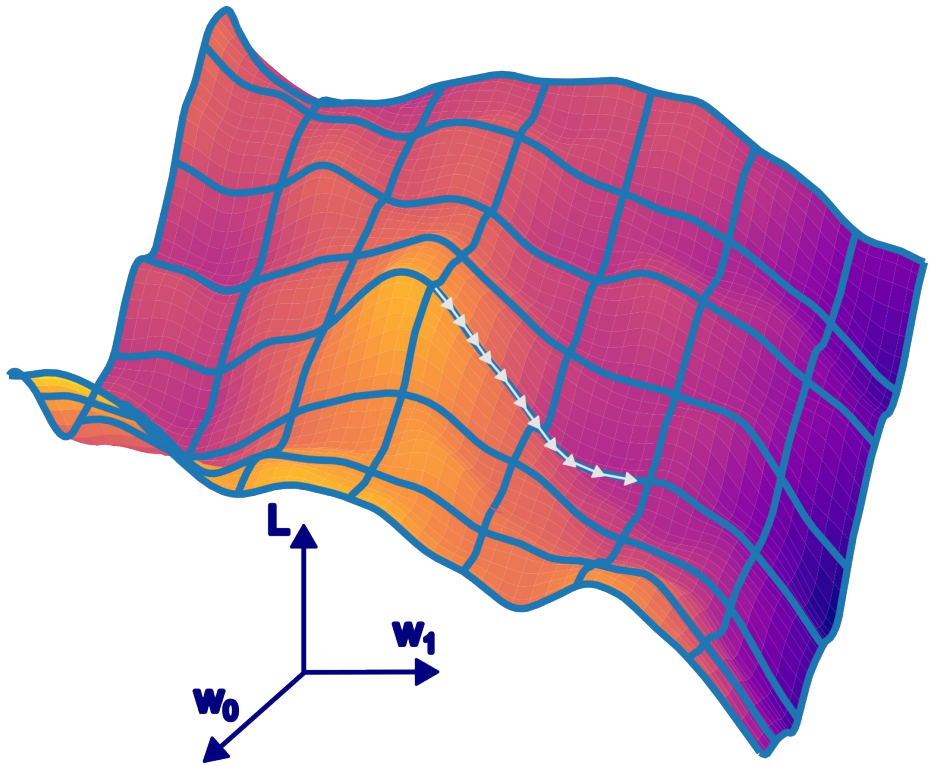


3.2 Neural Networks

Perceptron: Weight update

Now we can evaluate the derivative:

$$
\begin{aligned}
\frac{\partial L}{\partial w_i} &= \frac{\partial}{\partial w_i}\frac{1}{2N}\sum_{k=1}^{N}\left(y^k - \vec{w}^T\vec{x}^k\right)^2 \\
&= \frac{1}{2N}\sum_{k=1}^{N}\frac{\partial}{\partial w_i}\left(y^k - \vec{w}^T\vec{x}^k\right)^2 \\
&= \frac{1}{2N}\sum_{k=1}^{N}2\left(y^k - \vec{w}^T\vec{x}^k\right)\frac{\partial}{\partial w_i}\left(y^k - \vec{w}^T\vec{x}^k\right) \\
&= \frac{1}{N}\sum_{k=1}^{N}\left(y^k - \vec{w}^T\vec{x}^k\right)\frac{\partial}{\partial w_i}\left[-(w_1,\ldots,w_i,\ldots,w_n)\begin{pmatrix}x_1^k\\ \vdots\\ x_i^k\\ \vdots\\ x_n^k\end{pmatrix}\right] \\
&= -\frac{1}{N}\sum_{k=1}^{N}\left(y^k - \vec{w}^T\vec{x}^k\right)\frac{\partial}{\partial w_i}\left[(w_1x_1^k + \cdots + w_ix_i^k + \cdots + w_nx_n^k)\right] \\
&= -\frac{1}{N}\sum_{k=1}^{N}\left(y^k - \vec{w}^T\vec{x}^k\right)x_i^k \ .
\end{aligned}
$$



3.2 Neural Networks

Perceptron: Learning Algorithm

With the help of the derived learning rule, we can now formulate a learning algorithm.

1. initialize all weights $\vec{w} = (w_0, \ldots, w_n)$.
2. for each epoch:
   - Set $\Delta w_i = 0$
   - For each set of training data $(x^k, y^k), k = 1, \ldots, N$:
     - Calculate output y^k.
     - Calculate weight update: $\Delta w_i^k = \Delta w_i^k + (y^k - \bar{y}^k) x_i^k$.
   - Calculate the mean of all weight updates over the training data: $\Delta w_i = \frac{\alpha}{N} \sum_{k=1}^{N} \Delta w_i^k$.
   - Update all weights $w_i = w_i + \Delta w_i$

3.2 Neural Networks

Perceptron: Batch processing of training data

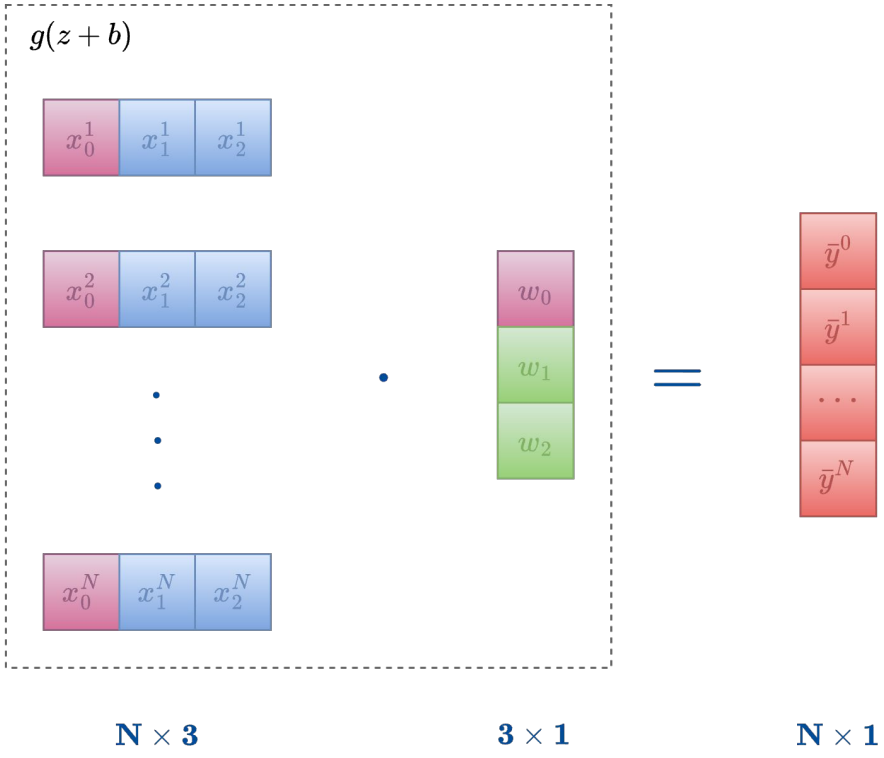
We have implemented the perceptron in such a way that the matrix x contains all feature vectors.

This ensures that all training data is run through before the weights are updated. The feature vectors are stacked on top of each other to form an N x 3 input matrix.

It should be noted that the first column of the matrix represents the bias neuron, i.e. contains only ones.

By matrix multiplying the N x 3 input matrix with the 3 x 1 weight vector, we obtain an N x 1 output vector that contains all outputs for all training examples.

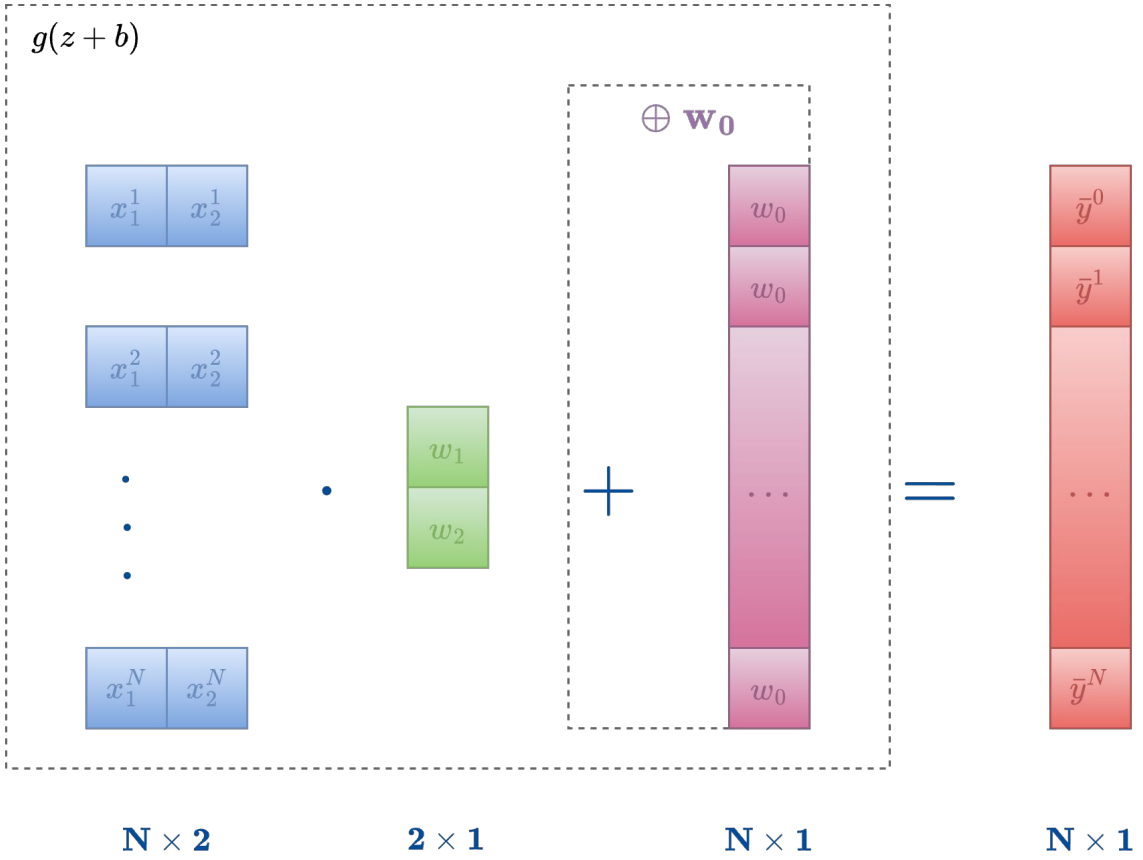
The input vectors are actually row vectors and stacked on top of each other.



3.2 Neural Networks

Perceptron: Batch processing of training data

However, we can make the implementation more efficient by “broadcasting” the weight of the bias neuron.



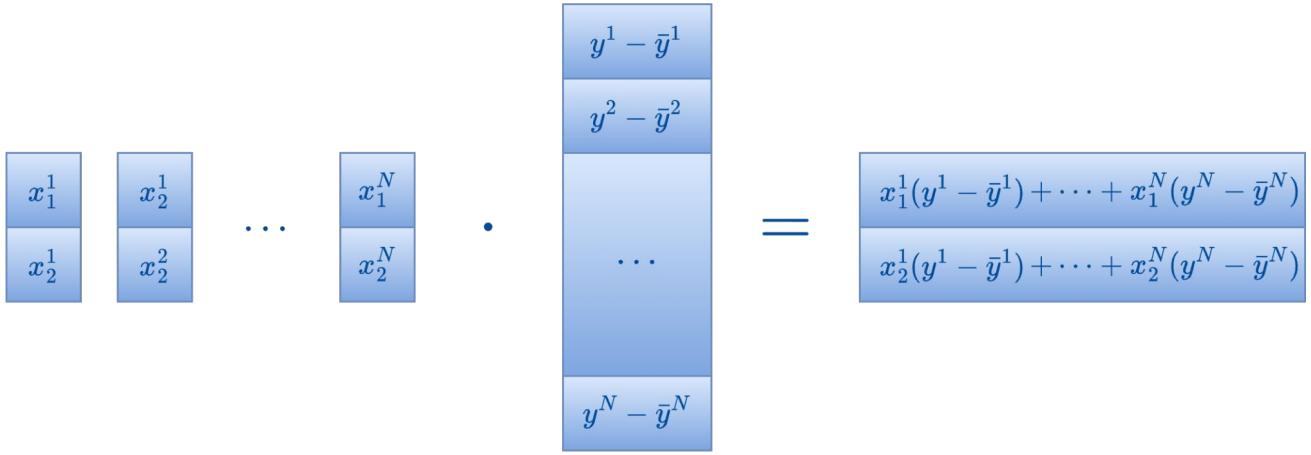
3.2 Neural Networks

Perceptron: Implementation of the weight update

Recall the definition of the weight update:

$$\vec{w} := \vec{w} + \Delta\vec{w}$$

$$\Delta w_i = -\alpha \frac{\partial L(w)}{\partial w_i} = \frac{\alpha}{N} \sum_{k=1}^{N} \left(y^k - \bar{y}^k\right) x_i^k$$

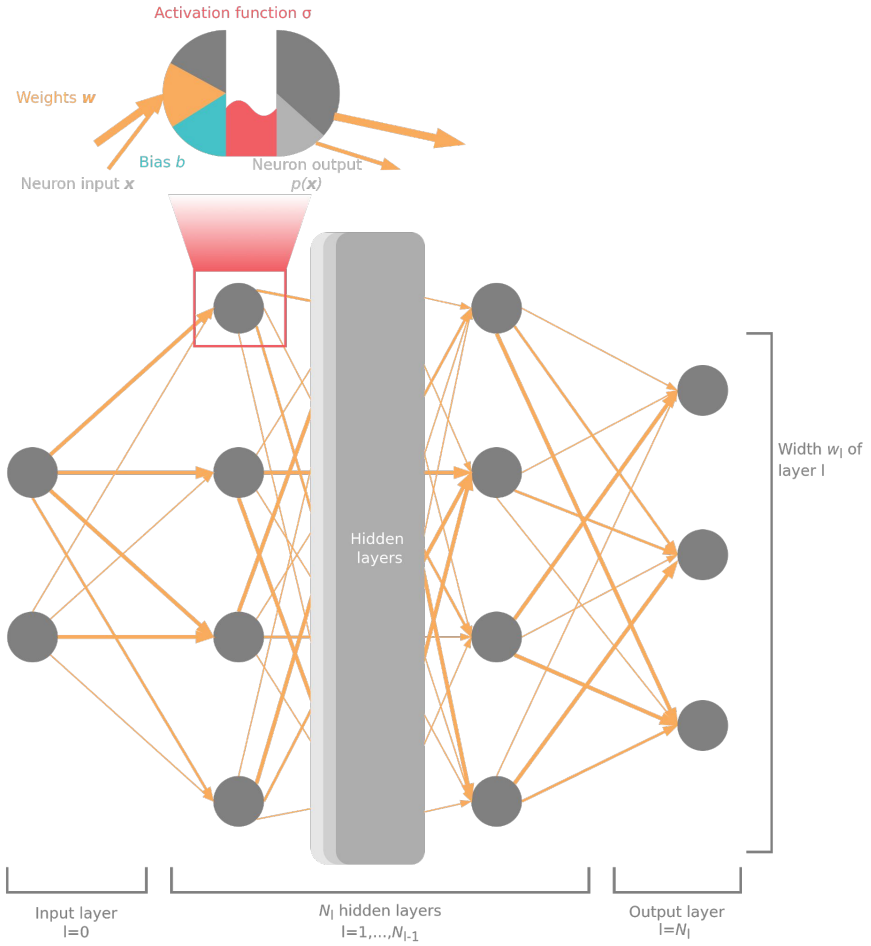


3.2 Neural Networks

Feedforward neural networks

- Composed of simple base units (neurons, perceptrons).
- The base units perform mathematical operations.
- Data flows in one direction
  - input layer — hidden layers — output layer
- Neurons apply activation functions to introduce non-linearity, enabling the network to model complex patterns.

$$p(\boldsymbol{x}) = \sigma(\boldsymbol{w} \cdot \boldsymbol{x} + b)$$



3.2 Neural Networks

Feedforward neural networks

- Hidden layer in a feedforward neural network

$$\boldsymbol{x}_{l+1} = \sigma(\mathbf{W}_l \boldsymbol{x}_l + \boldsymbol{b}_l)$$

- Weights and biases are adjusted during training using by minimizing a loss function.

$$\min_{\underline{\mathbf{W}},\underline{\boldsymbol{b}}} L(\underline{\boldsymbol{x}}, \underline{\boldsymbol{y}}, \underline{\mathbf{W}}, \underline{\boldsymbol{b}})$$

$$L(\underline{\boldsymbol{x}}, \underline{\boldsymbol{y}}, \underline{\mathbf{W}}, \underline{\boldsymbol{b}}) = \frac{1}{N_j} \sum_{j=1}^{N_j} (\tilde{\boldsymbol{y}}_j - \boldsymbol{y}_j)^2$$

$$= \frac{1}{N_j} \sum_{j=1}^{N_j} [M(\boldsymbol{x}_j, \underline{\mathbf{W}}, \underline{\boldsymbol{b}}) - \boldsymbol{y}_j]^2$$

- In each iteration of training the weights and biases of the network are determined by the backpropagation algorithm.

- Weights and biases are updated using stochastic gradient descent.

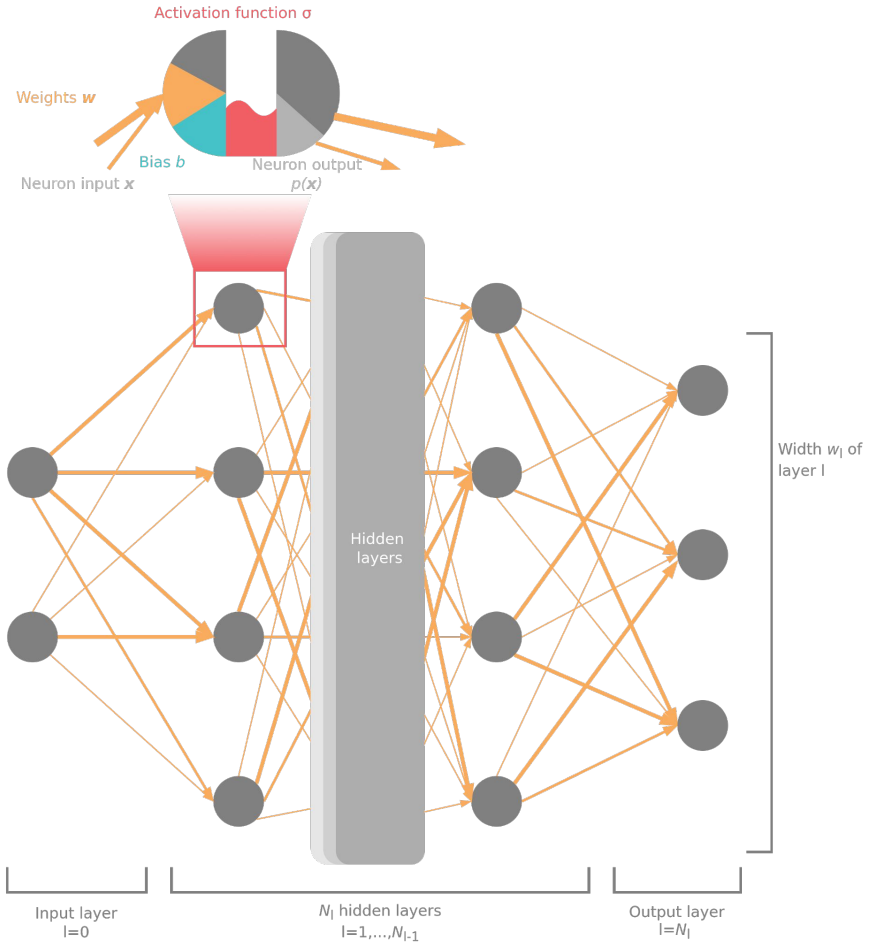
$$\boldsymbol{W}_{\iota+1,\mathrm{l},\chi} = \boldsymbol{W}_{\iota,\mathrm{l},\chi} - \frac{\gamma}{N_m} \sum_{\zeta=1}^{N_m} \nabla_{\boldsymbol{W}_{N_\mathrm{l}},\chi} L\left(\boldsymbol{x}_\zeta, \boldsymbol{y}_\zeta\right)$$

Attila Cangi | a.cangi@hzdr.de

3.2 Neural Networks

Feedforward neural networks

- Model parameters such as weights and biases are updated during training using stochastic gradient descent.
- What about other parameters, such as layer width, the number of layers, activation function, and learning rate?
- These are hyper-parameters and must adjusted during training separately.
- Hyperparameter optimization is the process of identifying the ideal model architecture and training procedures.



3.2 Neural Networks

Neural network tutorial

Tutorial on the basics of neural networks

- https://github.com/GDS-Education-Community-of-Practice/DSECOP

Data Science Education Community of Practice (DSECOP)

https://dsecop.org/