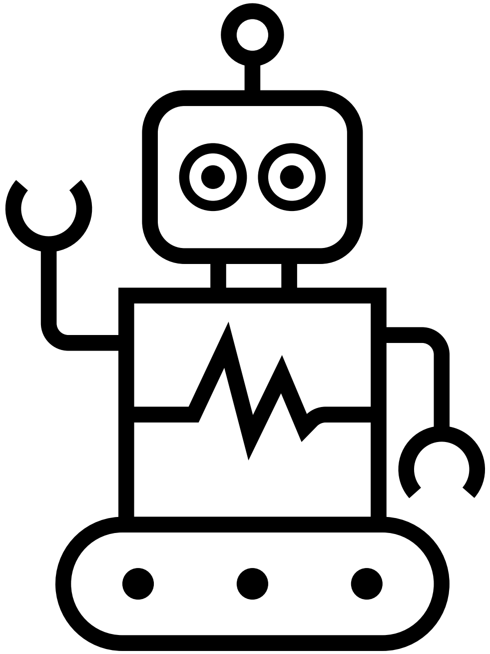
Machine Learning

…for physicists?



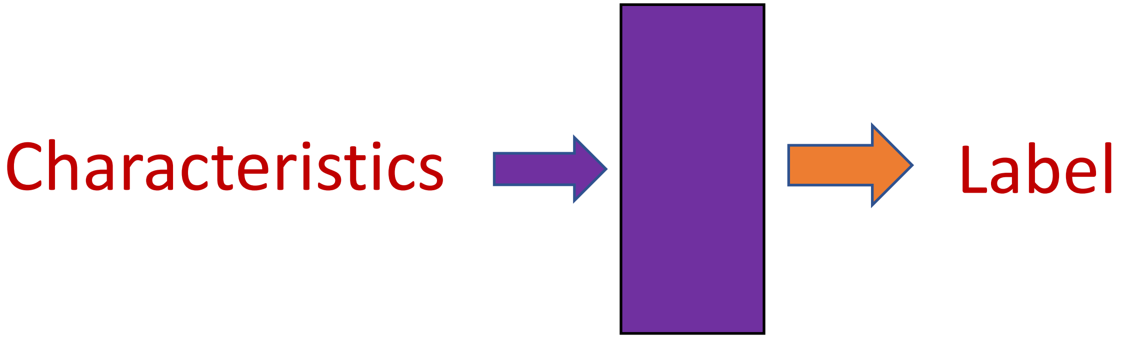
Teachers:

Menno Bokdam

Hans Kanger

Course code	**202100224**			
Short name	Machine Learning			
Contact person	**Bokdam, M.**			
EC amount	3 or 5 EC	Instructional language	☐NL	☒ EN

Machine Learning in its ‘essence’



$$y = F[A, B, C, D, E, \Phi, \Omega, \ldots]$$

Machine Learning

…for physicists?

As physicists we love building ‘toy’ models:

- Ising model
- Planar capacitor model
- Incompressible flow
- The ‘ideal’ gas
- Cow as a point particle
- … and so on…

This works well, but:

- Is often limited by our physical/chemical intuition
- It is not always easy to systematically improve the accuracy of the model
- Requires higher order theory

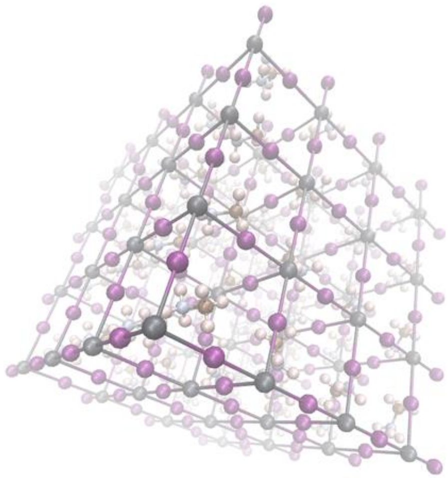
But has clear advantages as well:

- Physically intuitive model
- Often converges to the ‘exact’ solution in limiting situations
- Thereby solutions are bound and do not unexpectedly diverge.

Machine-Learning models can be complementary:

- A model can be constructed purely on (experimental) ‘data’
- Complexity of the model beyond ‘fitted’ functions

Machine learning for atoms

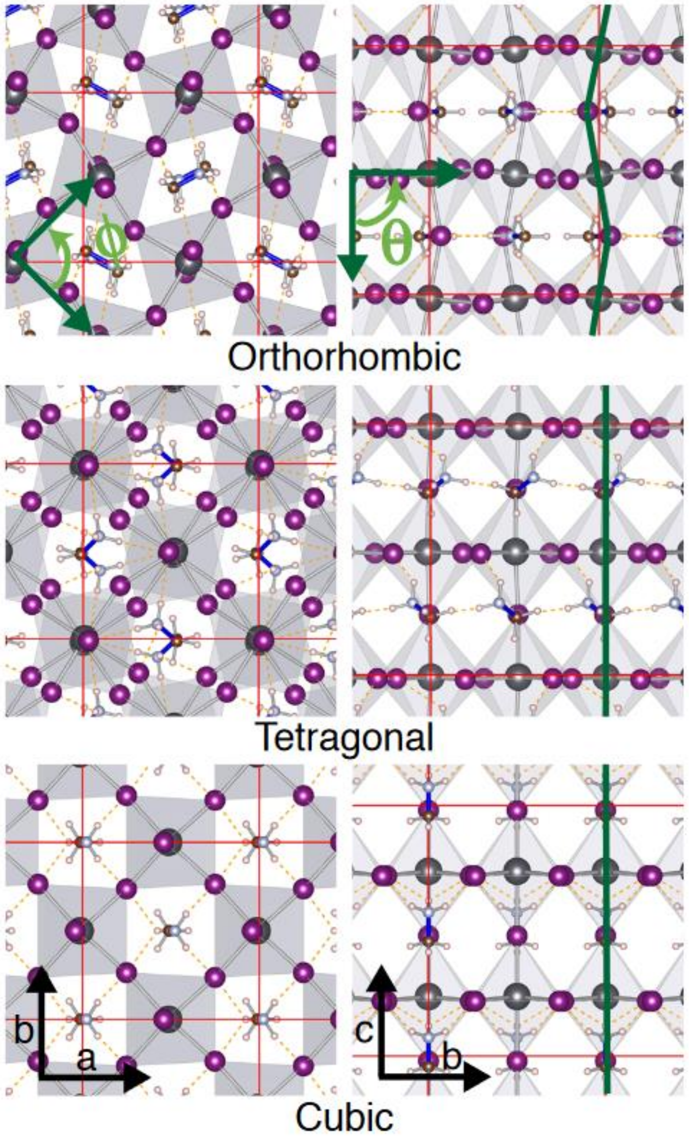


Energie

E_O

E_T

E_C



Characteristics

- $\mathbf{R}_i$: Coordinates
- S_i: Atomic spiecies

Labels

E: Potential energy (J)

$\mathbf{F}_i$: Force on atom i

UNIVERSITY OF TWENTE.

Machine learning for atoms

Start structure:
$\{\mathbf{R}_i$, Type i$\}$

Solve Schrödinger equation.

Not even the supercomputer in Amsterdam is fast enough!

$\{\mathbf{F}_i\}$

Propagate structure in time:
$\mathbf{v}_i(t)=v_i(t)+\mathbf{F}_i/m_i{*}dt$
$\mathbf{R}_i(t+1)=\mathbf{R}_i(t)+\mathbf{v}_i(t)\ {*}dt$

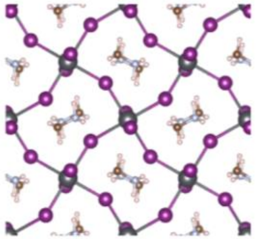
$\mathbf{R}_i(t+1)$

UNIVERSITY OF TWENTE.

Machine Learning

...for physicists?

Example: Machine-Learning Force Fields for Solid State Physics

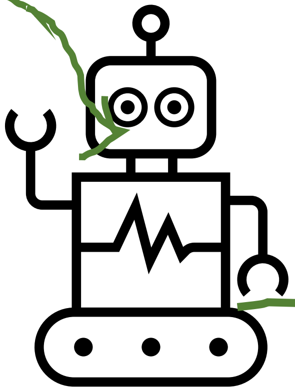


“Exact” theory, but computationally untractable

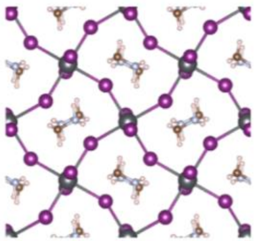
As usual in many-body electronic structure calculations, the nuclei of the treated molecules or clusters are seen as fixed (the Born–Oppenheimer approximation), generating a static external potential V, in which the electrons are moving. A stationary electronic state is then described by a wavefunction $\Psi(\mathbf{r}_1, \ldots, \mathbf{r}_N)$ satisfying the many-electron time-independent Schrödinger equation

$$\hat{H}\Psi = \left[\hat{T} + \hat{V} + \hat{U}\right]\Psi = \left[\sum_{i=1}^{N}\left(-\frac{\hbar^2}{2m_i}\nabla_i^2\right) + \sum_{i=1}^{N} V(\mathbf{r}_i) + \sum_{i<j}^{N} U\left(\mathbf{r}_i, \mathbf{r}_j\right)\right]\Psi = E\Psi,$$

Source: Wikipedia.org



“Mean-field” theory, computationally tractable, but limited in spatial and time dimensions



Here DFT provides an appealing alternative, being much more versatile, as it provides a way to systematically map the many-body problem, with $\hat{U}$, onto a single-body problem without $\hat{U}$. In DFT the key variable is the electron density $n(\mathbf{r})$, which for a normalized Ψ is given by

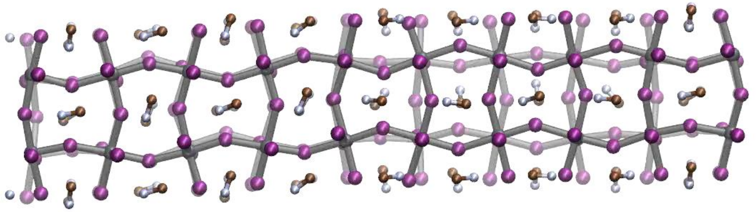
$$n(\mathbf{r}) = N \int \mathrm{d}^3\mathbf{r}_2 \cdots \int \mathrm{d}^3\mathbf{r}_N\, \Psi^*(\mathbf{r}, \mathbf{r}_2, \ldots, \mathbf{r}_N)\Psi(\mathbf{r}, \mathbf{r}_2, \ldots, \mathbf{r}_N).$$

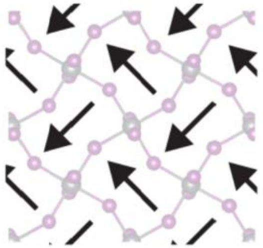
$$E[n] = T[n] + U[n] + \int V(\mathbf{r})n(\mathbf{r})\,\mathrm{d}^3\mathbf{r}$$

Source: Wikipedia.org

with respect to $n(\mathbf{r})$, assuming one has reliable expressions for $T[n]$ and $U[n]$. A successful minimization of the energy functional will yield the ground-state density n_0 and thus all other ground-state observables.

225 K



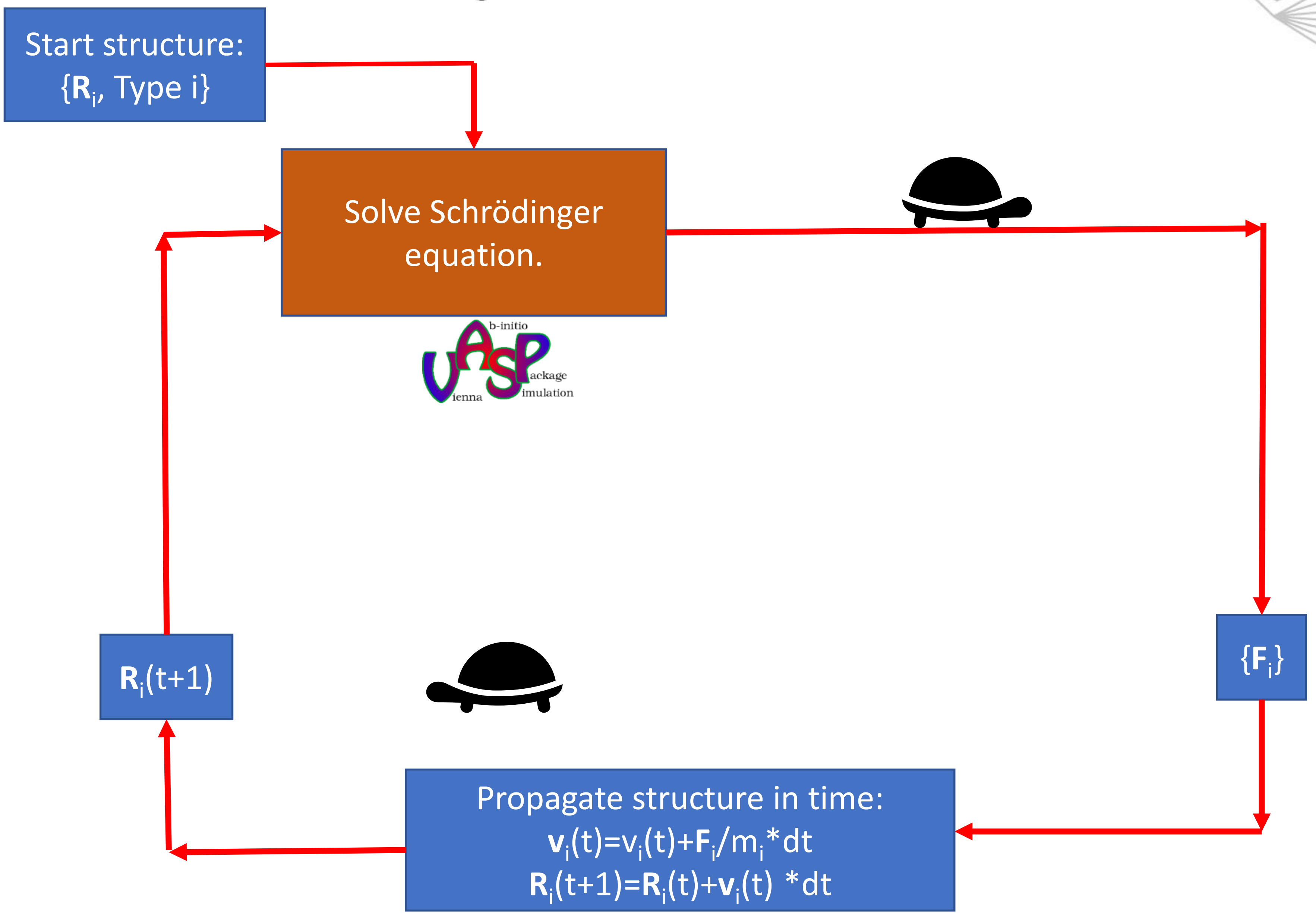


“Model” potential energy surface, quick and often “dirty”

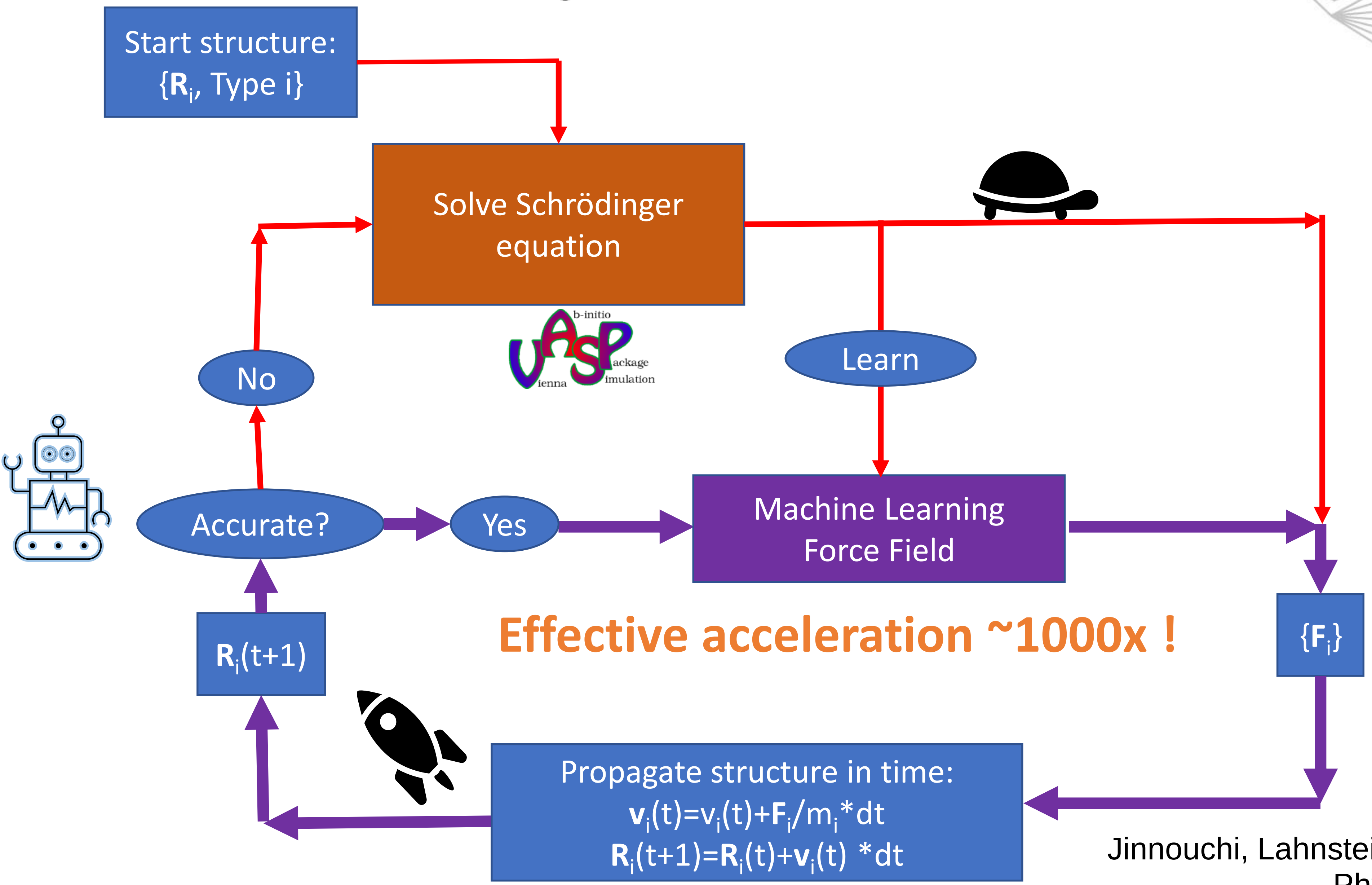
$$H_{\mathrm{lr}} = \frac{1}{2}\sum_{i=1}^{N}\sum_{j\in r_c} U(\mathbf{p}_i, \mathbf{p}_j, \mathbf{n}_{ij}). \qquad U(\mathbf{p}_i, \mathbf{p}_j, \mathbf{n}_{ij}) = \frac{|\mathbf{p}|^2}{4\pi\varepsilon_0\varepsilon_r}\frac{1}{r_{ij}^3}[\hat{\mathbf{p}}_i\hat{\mathbf{p}}_j - 3(\hat{\mathbf{p}}_i\cdot\hat{\mathbf{n}}_{ij})(\hat{\mathbf{p}}_j\cdot\hat{\mathbf{n}}_{ij})],$$

Source: PHYSICAL REVIEW LETTERS 122, 225701 (2019)
PHYSICAL REVIEW B 100, 094106 (2019)

Machine learning for atoms

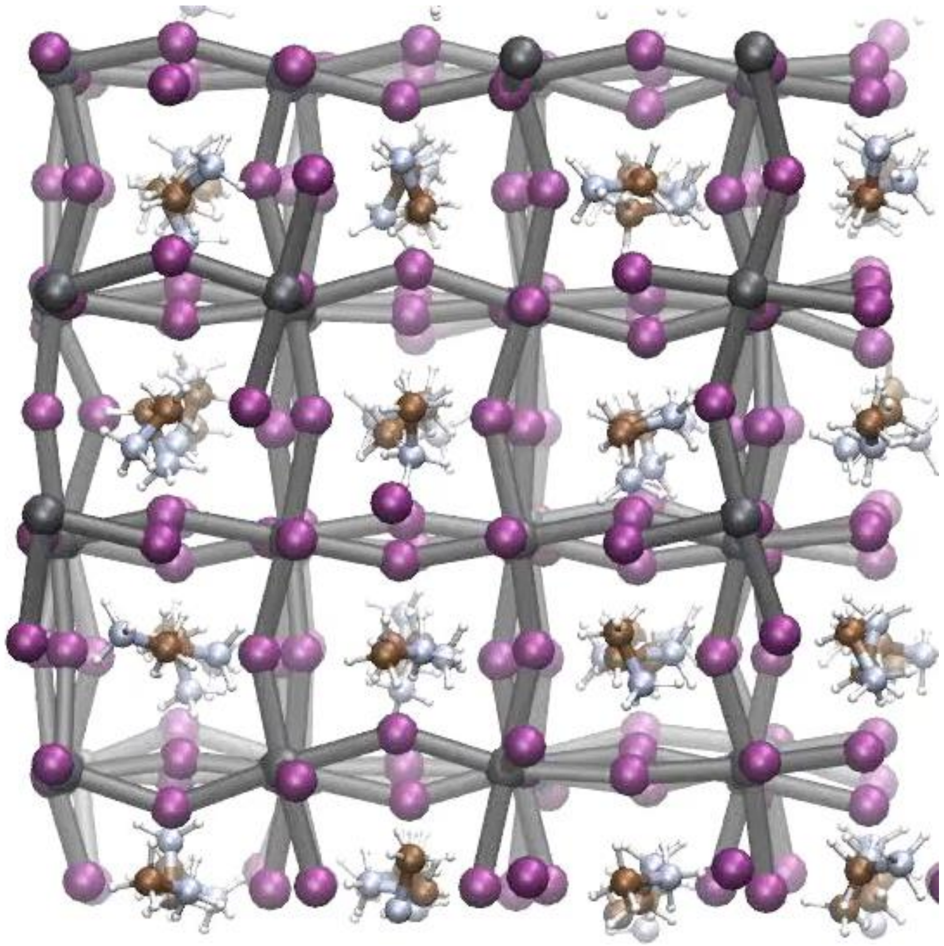


Machine learning for atoms

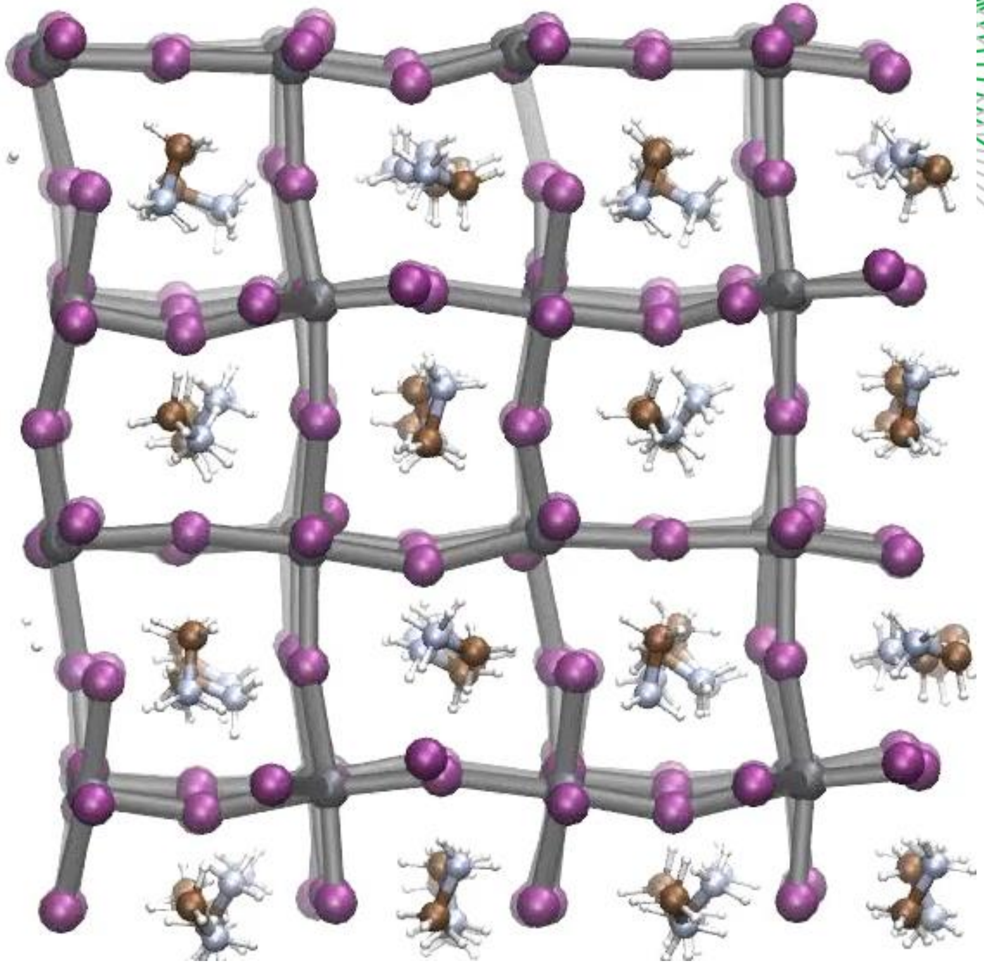


Jinnouchi, Lahnsteiner, Karsai, Kresse and **Bokdam**
Phys. Rev. Lett. **122**, 225701 (2019)

Result a ‘movie’ on the atomic scale



27^o C



127^o C

Machine Learning

the course

’Learning’ can happen in (at least) three ways:

(1) Supervised learning;
(2) Unsupervised learning;
(3) Reinforcement learning.

For 3EC: You will get acquainted with the first two; supervised learning will be dominant.

For 5EC: You will get acquainted with all three; supervised learning and reinforcement learning will be dominant.

We will treat unsupervised learning but do not discuss it here.

Supervised learning

examples

You are given data $\boldsymbol{x}_i \in \mathbb{R}^M, \;\; i = 1, \ldots, N$ with a label (binary: $0, 1$, multiclass: $0, 1, 2, \ldots$)

$M = 28^2, N = 100$ of 60000, multiclass $0, \ldots, 9$:

Supervised learning

examples

You are given data $\boldsymbol{x}_i \in \mathbb{R}^M, \;\; i = 1, \ldots, N$ with a label (binary: $0, 1$, multiclass: $0, 1, 2, \ldots$)

$M =$ Large, $N = 8$ of 25000, binary:

Supervised learning

examples

You are given data $\boldsymbol{x}_i, \;\; i = 1, \ldots, N$ with a label (binary: $0, 1$, multiclass: $0, 1, 2, \ldots$)

(1) You choose a method.

(2) Training dataset: The sample of data used to fit the model/using the method.

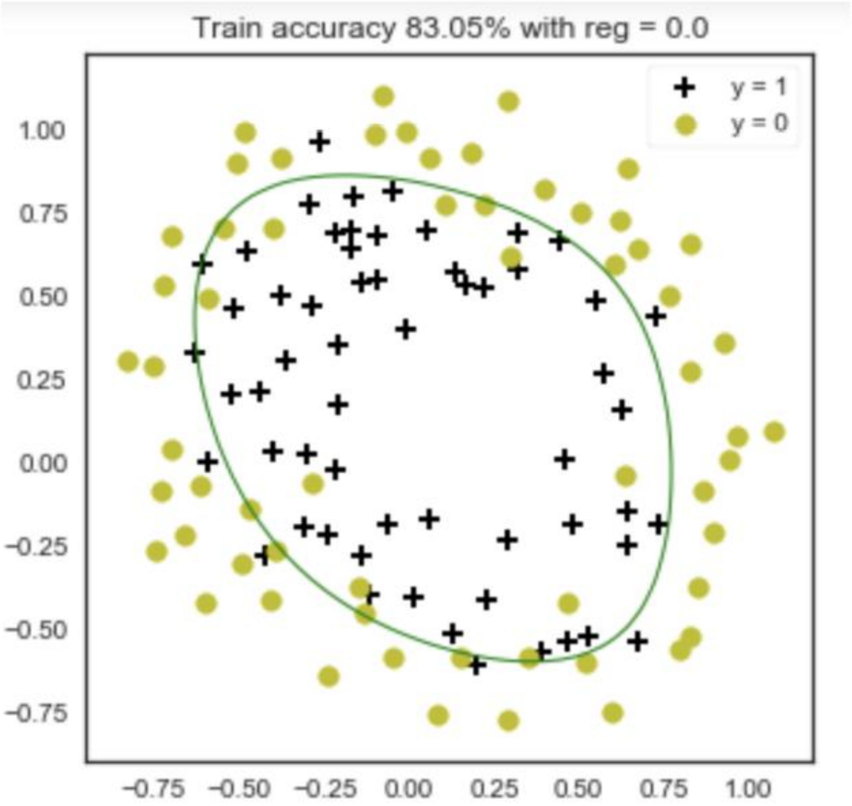
(3) Validation dataset: Is your chosen model/method 'correct'?

(4) Test set: How well does the model predict the class of that data?

Supervised learning

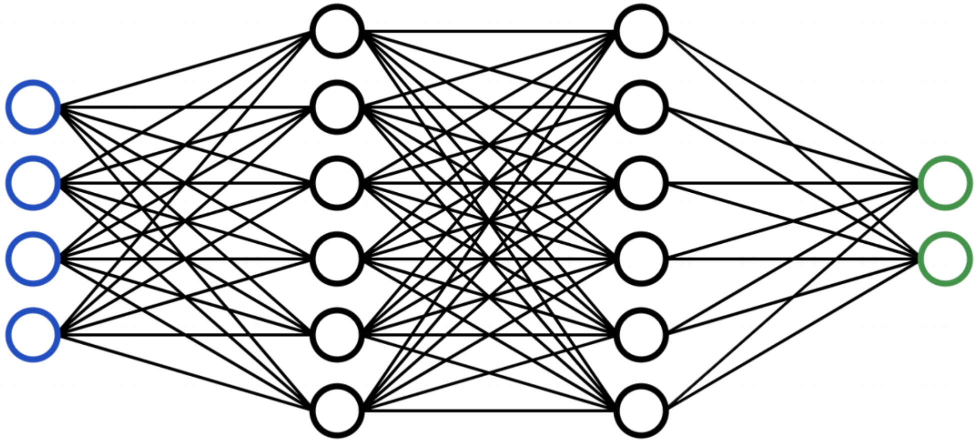
methods

- You start simple, low dimensional data, basic methods: Regression, Support Vector Machines, ...
- You will use Python notebooks. You can easily learn to program in Python as in the beginning the notebooks are pre-programmed.
- All ML methods use optimization; we use interactive methods for that as we will need that in case of neural networks.
- You will also use ML-packages to compare your results (or: you need to be able to work with packages as well).
- You will find that your codes are not fast: but you can use parts of your codes to ...



Build a neural net

- We supply a framework with functions that you will have to complete
- For testing, plotting and playing around the object oriented language Python is convenient
- They can run on your computer, but for large problems we work on Google's colab (GPUs);



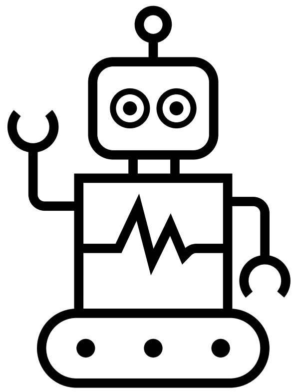
Machine Learning (3EC)

- You hand in the homework/codes previously discussed (work in pairs allowed)

- For the remaining 1 EC you get a data set (from Kaggle) that you have to analyse and write a report about.

  The work can be done individually or in the same pair as the homework

  The methods that you can use are the ones from this course, but you can try to find methods that are better suited.



Machine Learning (5EC)

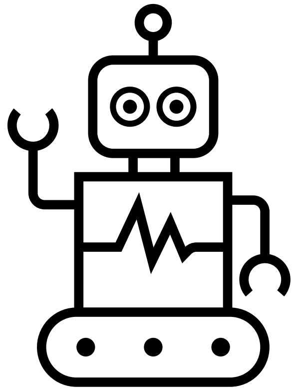
- You hand in the homework/codes previously discussed (work in pairs allowed)

- For the remaining 3 EC course you are going to work with reinforcement learning; basically you let a computer learn a game such that it beats you. You do this by letting him/her/it play the game many, many times.

  You will get material to study;

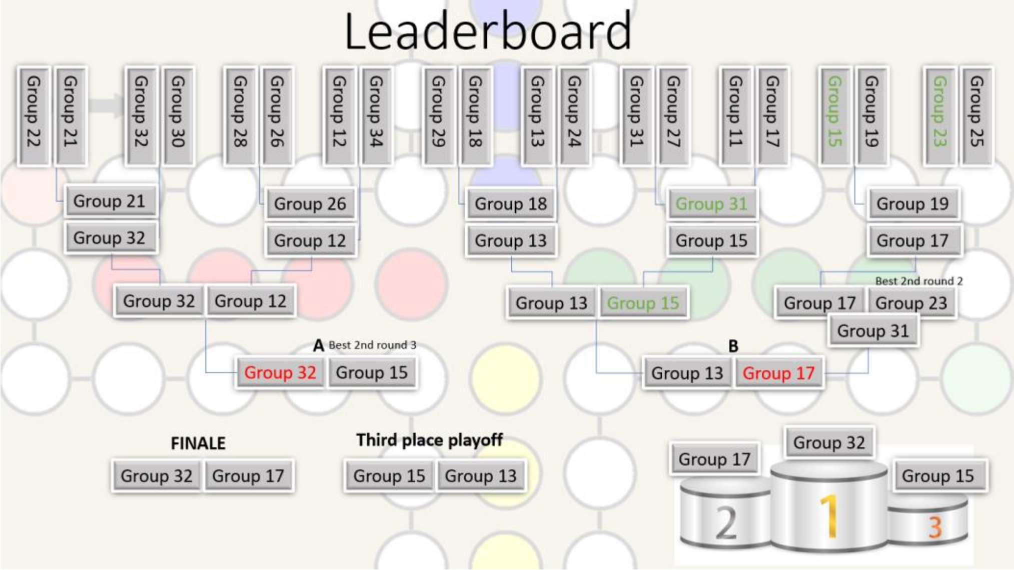
  You write a report and supply us with a working code, also on our computers;

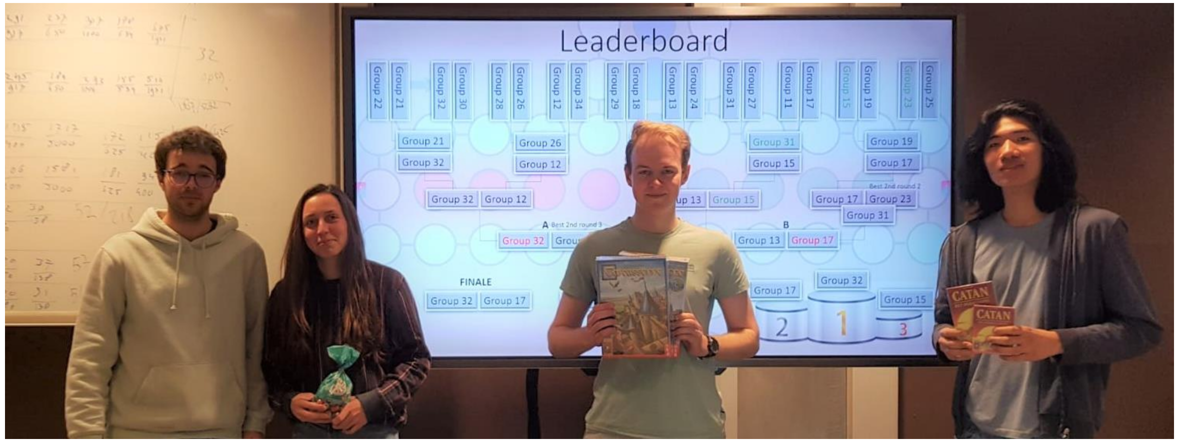
  You work in little groups (2/3 or 4?, will depend on total number of students);



Project Machine Learning (5EC)

- Final assignment of 2022: Mens erger je niet





Assignment: Build a self-learning model of the player which is statistically significant better then a first order strategy and compete in the class competition.

UNIVERSITY OF TWENTE.

Project Machine Learning (5EC)

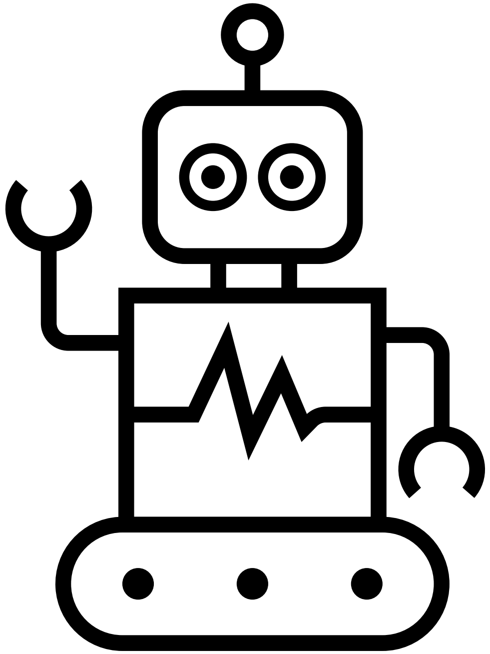
- Final assignment of 2021: Tiny Toons

Assignment: Build a self-learning model of the player which maximizes the total score.

UNIVERSITY OF TWENTE.

Machine Learning

...for you!



Teachers:

Menno Bokdam

Hans Kanger

course philosophy:

Can Do

Hands-on

Have fun!