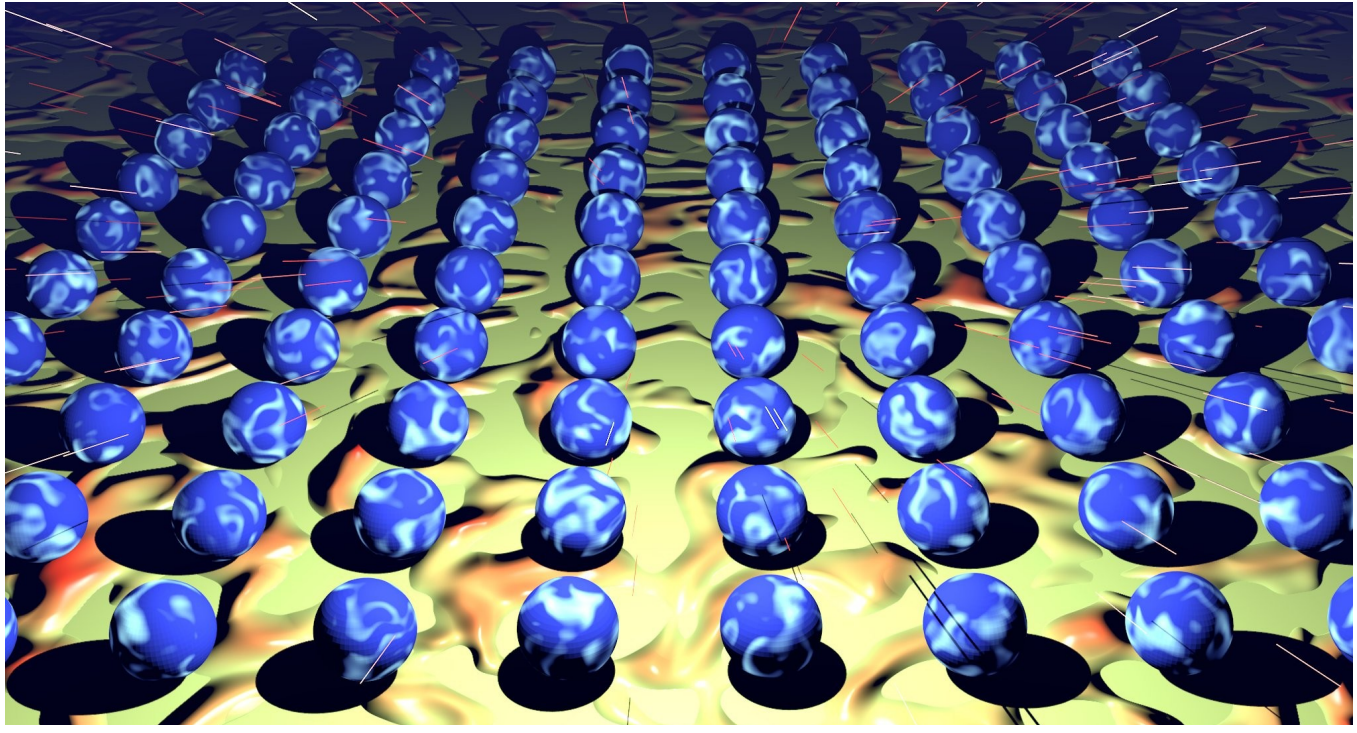
Classical machine learning for quantum materials



May 15th 2023

Today’s learning outcomes

- The relation between a many-body wavefunction and a machine learning problem
- Why some methods for solving wavefunctions can be used for machine learning
- Why methods from machine learning provide insights in phases of quantum materials

Outline

- Tensor-networks for quantum many-body wavefunctions
- Neural-networks for quantum many-body wavefunctions
  - Exercise #1 for this week
- Tensor-networks for supervised learning
- Neural-networks for density functional theory
- Using machine learning for quantum materials
  - Exercise #2 for this week

Schedule

- 3 min pair discussion
- 15 min lecture
- 3 min pair discussion
- 15 min lecture
- 3 min pair discussion
- 10 min break
- 20 min lecture

Parametrizing many-body wavefunctions with tensor networks

Previously, in session 8: The quantum Heisenberg dimer

What is the ground state of this quantum Hamiltonian?

$$\mathcal{H} = \vec{S}_0 \cdot \vec{S}_1$$

The ground state is unique, and does not break time-reversal

$$|GS\rangle = \frac{1}{\sqrt{2}}[|\uparrow\downarrow\rangle - |\downarrow\uparrow\rangle] \qquad \langle \vec{S}_i \rangle = 0$$

The state is maximally entangled

Can we have a macroscopic version of this ground state? $\langle \vec{S}_i \rangle = 0$

The quantum many-body problem

Let us go back to a simple many-body problem $\mathcal{H} = \sum_{ij} J_{ij} \vec{S}_i \cdot \vec{S}_j$

And let us imagine that we have L different sites on our system and S=1/2

For example, for L=2 sites the elements of the basis are

$|\uparrow\uparrow\rangle \quad |\uparrow\downarrow\rangle \quad |\downarrow\uparrow\rangle \quad |\downarrow\downarrow\rangle$

For L=3 sites the elements of the basis are

$|\uparrow\uparrow\uparrow\rangle \quad |\uparrow\uparrow\downarrow\rangle \quad |\uparrow\downarrow\uparrow\rangle \quad |\uparrow\downarrow\downarrow\rangle$

$|\downarrow\uparrow\uparrow\rangle \quad |\downarrow\uparrow\downarrow\rangle \quad |\downarrow\downarrow\uparrow\rangle \quad |\downarrow\downarrow\downarrow\rangle$

The quantum many-body problem

Let us go back to a simple many-body problem

$$\mathcal{H} = \sum_{ij} J_{ij} \vec{S}_i \cdot \vec{S}_j$$

And let us imagine that we have L different sites on our system and S=1/2

For L=4 sites, the elements of the basis are

$$|\uparrow\uparrow\uparrow\uparrow\rangle \quad |\uparrow\uparrow\uparrow\downarrow\rangle \quad |\uparrow\uparrow\downarrow\uparrow\rangle \quad |\uparrow\uparrow\downarrow\downarrow\rangle$$
$$|\uparrow\downarrow\uparrow\uparrow\rangle \quad |\uparrow\downarrow\uparrow\downarrow\rangle \quad |\uparrow\downarrow\downarrow\uparrow\rangle \quad |\uparrow\downarrow\downarrow\downarrow\rangle$$
$$|\downarrow\uparrow\uparrow\uparrow\rangle \quad |\downarrow\uparrow\uparrow\downarrow\rangle \quad |\downarrow\uparrow\downarrow\uparrow\rangle \quad |\downarrow\uparrow\downarrow\downarrow\rangle$$
$$|\downarrow\downarrow\uparrow\uparrow\rangle \quad |\downarrow\downarrow\uparrow\downarrow\rangle \quad |\downarrow\downarrow\downarrow\uparrow\rangle \quad |\downarrow\downarrow\downarrow\downarrow\rangle$$

The quantum many-body problem

Let us go back to a simple many-body problem $\mathcal{H} = \sum_{ij} J_{ij} \vec{S}_i \cdot \vec{S}_j$

And let us imagine that we have L different sites on our system and S=1/2

For a system of size L, what is the dimension of the Hilbert space?

Option #1

$2L$

Option #2

L^2

Option #3

2^L

The quantum many-body problem

Let us go back to a simple many-body problem $\mathcal{H} = \sum_{ij} J_{ij} \vec{S}_i \cdot \vec{S}_j$

A typical wavefunction is written as

$$|\Psi\rangle = \sum c_{s_1, s_2, \ldots, s_L} |s_1, s_2, \ldots s_L\rangle$$

We need to determine in total 2^L coefficients

Is there an efficient way of storing so many coefficients?

Previously, in session 9: The matrix-product state ansatz

For this wavefunction $|\Psi\rangle = \sum c_{s_1,s_2,\ldots,s_L} |s_1, s_2, \ldots s_L\rangle$

Let us imagine to propose a parametrization in this form

$$c_{s_1,s_2,\ldots,s_L} = M_1^{s_1} M_2^{s_2} \ldots M_L^{s_3}$$

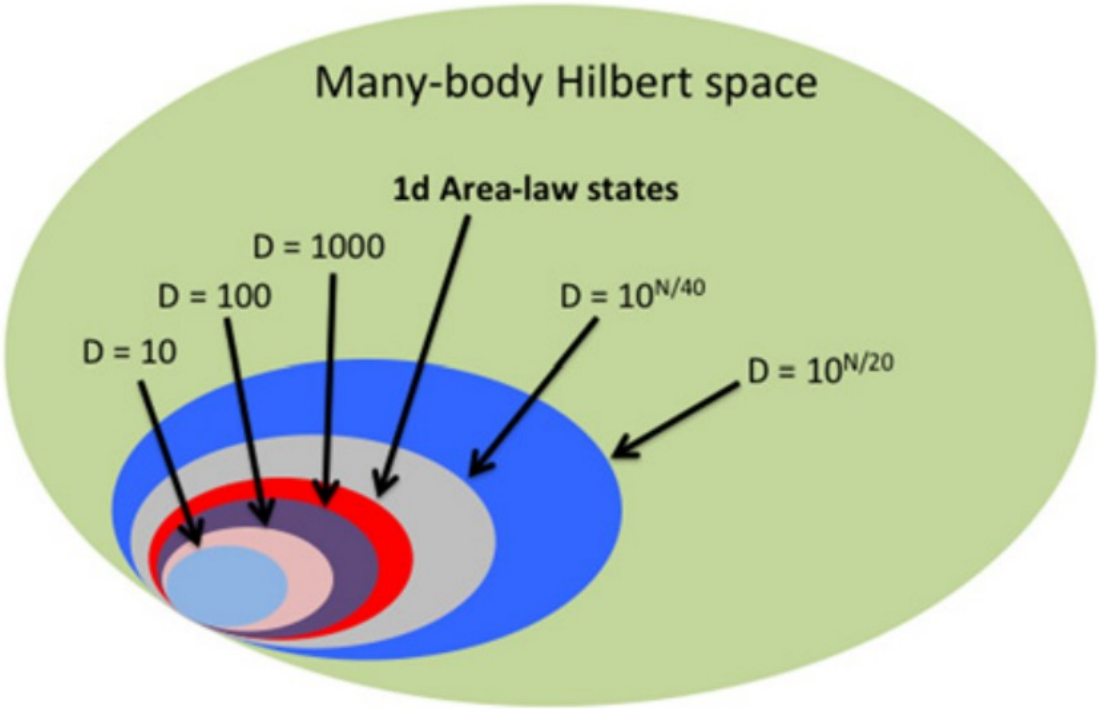
dimension 2^L

dimension $\sim LD^2$

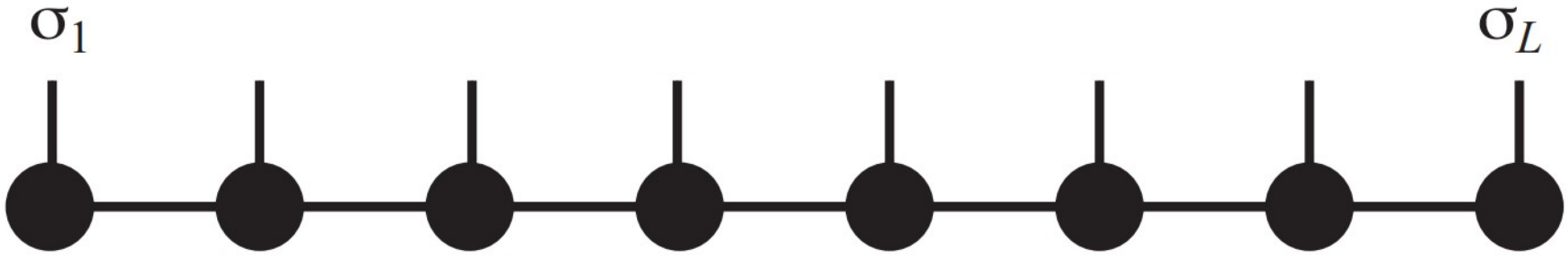
(D dimension of the matrix)

A controlled way of parametrizing the Hilbert space

Sketch of the space parametrized with bond dimension D



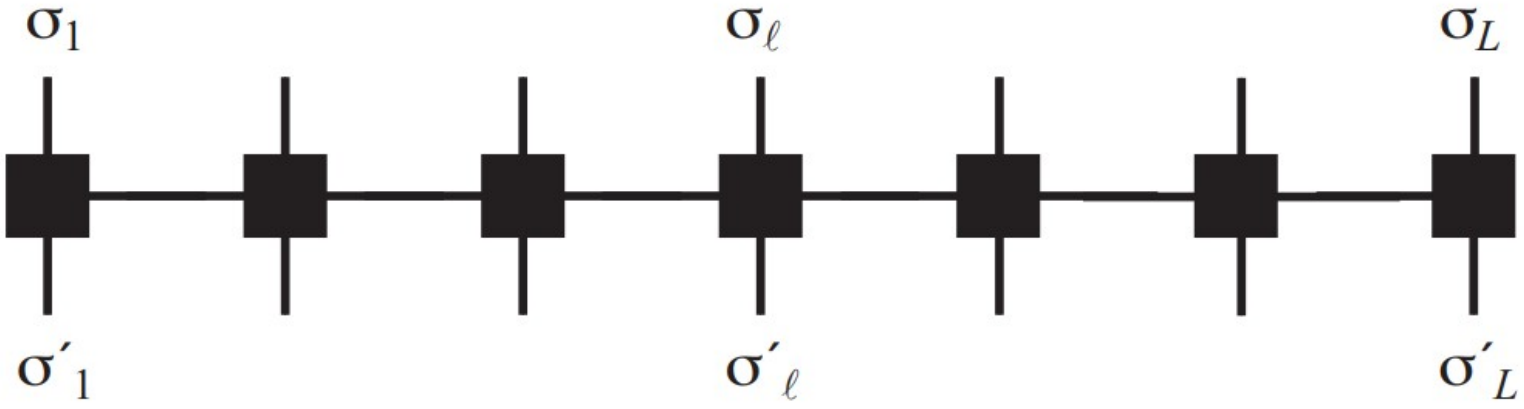
The matrix product state representation



$$|\psi\rangle = \sum_{\boldsymbol{\sigma}} M^{\sigma_1} \dots M^{\sigma_L} |\boldsymbol{\sigma}\rangle$$

Matrix product operators

Operators (like the Hamiltonian) can be represented in an analogous form



This is the Hamiltonian H

Ground state calculations

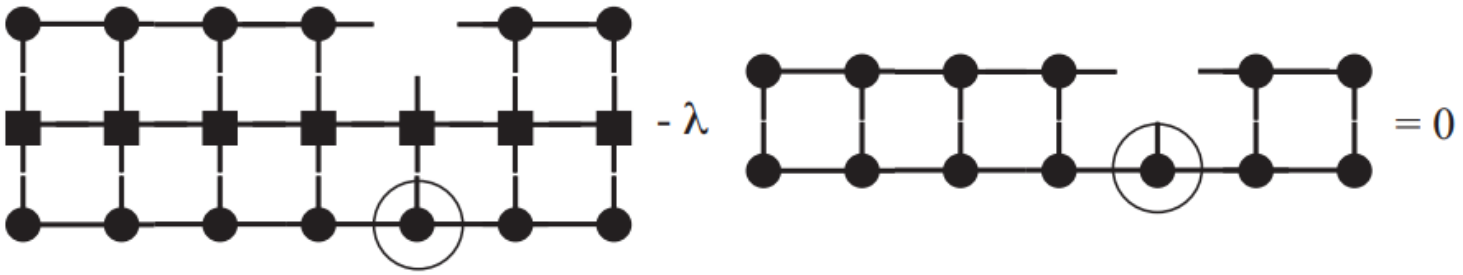
To compute a ground state, we just have to minimize

$$E = \frac{\langle \Psi | H | \Psi \rangle}{\langle \Psi | \Psi \rangle}$$

$$E = \langle \Psi | H | \Psi \rangle - \lambda \langle \Psi | \Psi \rangle$$

This can be done by minimizing the energy with respect to each matrix

$$\frac{\delta E}{\delta M} = 0$$

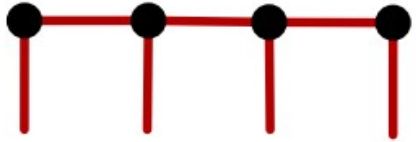


This algorithm is known as the density-matrix renormalization group

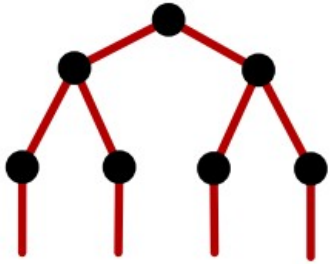
Generic tensor networks states

Tensor networks can be extended to deal with higher dimensional/critical systems

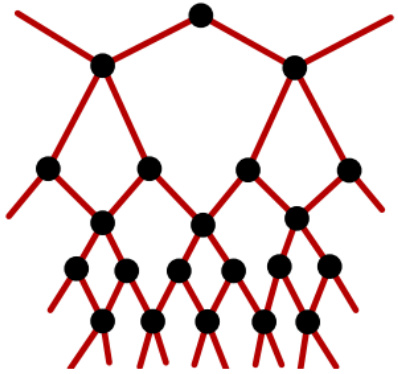
Matrix product state



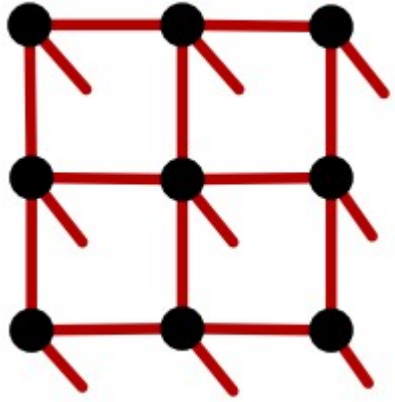
Tree tensor network



Multiscale renormalization ansatz



Projected-entangled pair states



(this will be relevant later)

For discussion (3 min)

Given a matrix product state for a very specific wavefunction

$$|\psi\rangle = \sum_{\boldsymbol{\sigma}} M^{\sigma_1} \dots M^{\sigma_L} |\boldsymbol{\sigma}\rangle$$

Are the matrices M^{s_i} unique, or are there multiple matrix product states consistent with the same wavefunction?

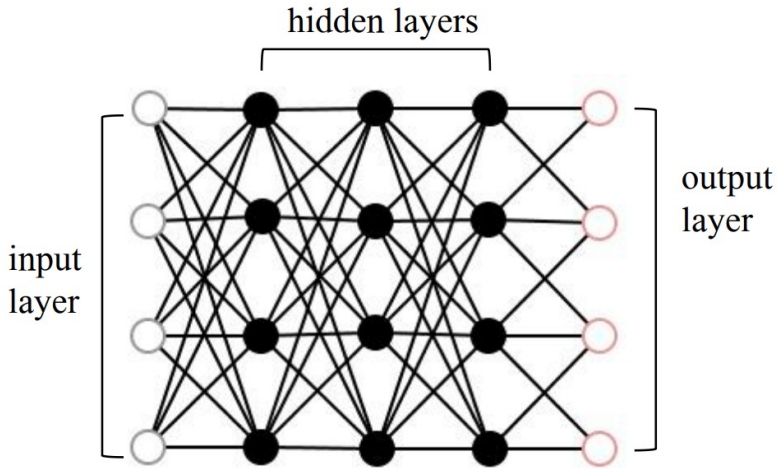
Can you give a one line demonstration of it?

Neural network quantum states

The basics of deep neural networks

Deep neural networks are “general function approximators”

$\vec{x}$



$\vec{y}$

A deep neural network parametrices a function of the form $f(\lambda, \vec{x}) = \vec{y}$

λ Parameters of the function

$\vec{x}$ Input of the function

A hierarchy of variational methods

Mean-field $J_{ij}\vec{S}_i \cdot \vec{S}_j \approx J_{ij}\langle\vec{S}_i\rangle \cdot \vec{S}_j + \ldots$

Minimizing over product states (no entanglement) $|\uparrow\downarrow\uparrow\uparrow\downarrow\downarrow\uparrow\uparrow\downarrow\uparrow\rangle$

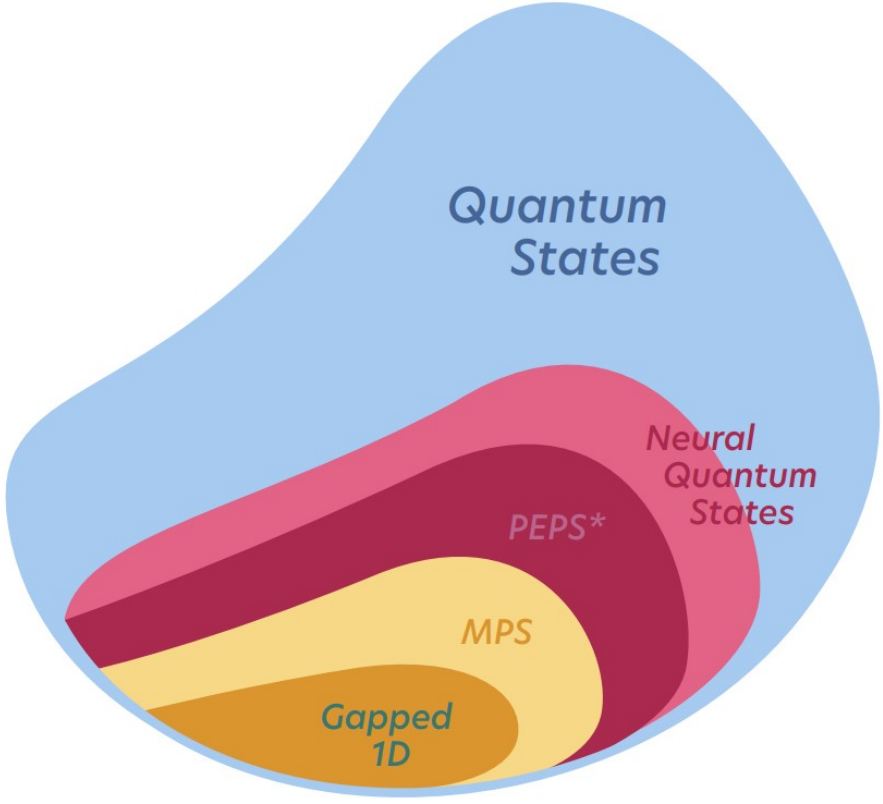
Matrix product states $|\psi\rangle = \sum_{\boldsymbol{\sigma}} M^{\sigma_1} \ldots M^{\sigma_L} |\boldsymbol{\sigma}\rangle$

Finite entanglement bounded by the bond dimension

Can we have more generic efficient variational wavefunctions for systems intractable with the ones above?

The hierarchy of quantum states

Different variational states allow accessing different parts of the Hilbert space



MPS: matrix product states

PEPS: projected entanlged pair states

The quantum many-body problem

Let us go back to a simple many-body problem $\mathcal{H} = \sum_{ij} J_{ij} \vec{S}_i \cdot \vec{S}_j$

A typical wavefunction is written as

$$|\Psi\rangle = \sum c_{s_1, s_2, \ldots, s_L} |s_1, s_2, \ldots s_L\rangle$$

We need to determine in total 2^L coefficients

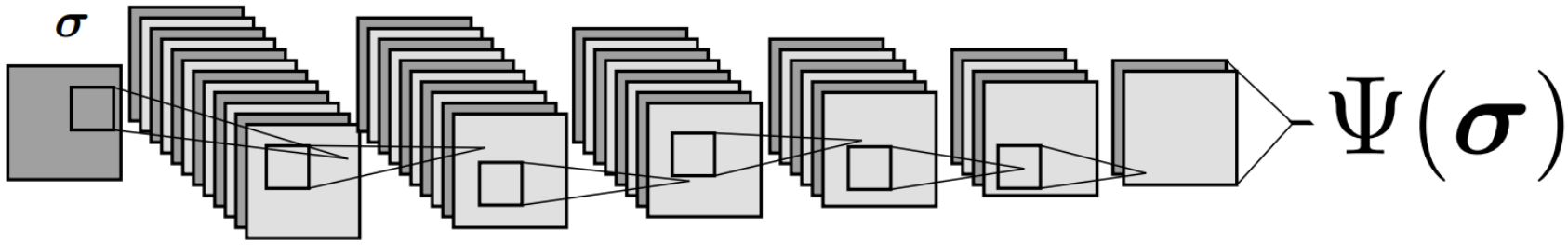
Before we stored them in a tensor-network, but could we do in a neural-network?

Neural-network quantum states

Do not store the coefficient, but find the right function that generates them

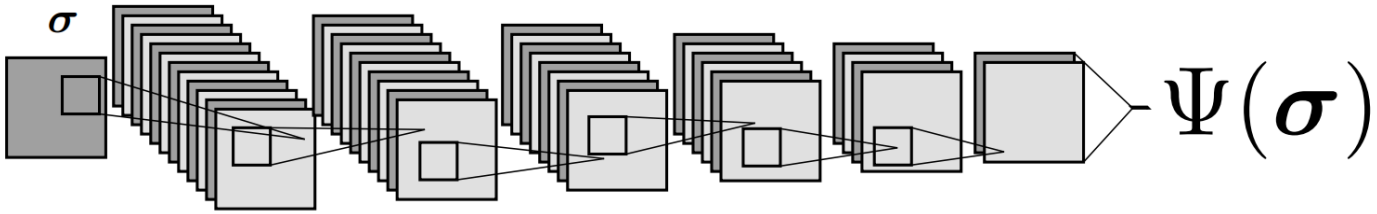
$$c_{s_1,s_2,\ldots,s_L} = f(s_1, s_2, \ldots, s_L)$$

Deep neural network



The idea is similar as tensor networks, but exploiting a machine-learning architecture

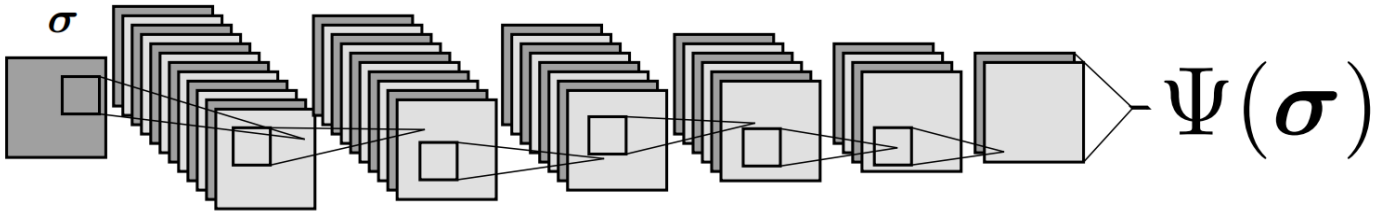
Neural-network quantum states



How do we find the right neural network for the ground state?

$$|\Psi\rangle = \sum c_{s_1,s_2,..,s_L} |s_1, s_2, ...s_L\rangle \qquad c_{s_1,s_2,....,s_L} = f(s_1, s_2, ..., s_L)$$

Neural-network quantum states



How do we find the right neural network for the ground state?

$$|\Psi\rangle = \sum c_{s_1,s_2,..,s_L} |s_1, s_2, ...s_L\rangle \qquad c_{s_1,s_2,....,s_L} = f(s_1, s_2, ..., s_L)$$

Optimize the parameters of the network to minimize $E = \dfrac{\langle\Psi|H|\Psi\rangle}{\langle\Psi|\Psi\rangle}$

Approximate expectation values

In general, we cannot exactly compute expectation values,
and a Monte Carlo method is required

Decompose this

$$\langle \hat{O} \rangle = \frac{\left\langle \Psi_{\boldsymbol{\theta}} | \hat{O} | \Psi_{\boldsymbol{\theta}} \right\rangle}{\langle \Psi_{\boldsymbol{\theta}} \mid \Psi_{\boldsymbol{\theta}} \rangle}$$

Into two terms

$$P(s) = \frac{|\langle \Psi_\theta \mid s \rangle|^2}{\sum_s |\langle \Psi_\theta \mid s \rangle|^2}$$

$$O_{\mathrm{loc}}(s) = \sum_{s'} \left\langle s | \hat{O} | s' \right\rangle \frac{\langle s' \mid \Psi_\theta \rangle}{\langle s \mid \Psi_\theta \rangle}$$

So that it can be expressed with a probability distribution

$$\langle \hat{O} \rangle = \sum_s P(s) O_{\mathrm{loc}}(s) = \langle O_{\mathrm{loc}}(s) \rangle_P$$

$$\langle \hat{O} \rangle \approx \frac{1}{M} \sum_{i=1}^{M} O_{\mathrm{loc}} \left(s^{(i)} \right)$$

Monte Carlo sampling

Finding the ground state with neural network quantum states

The Hamiltonian is expressed in local terms

$$E(\boldsymbol{\theta}) \approx \frac{1}{M} \sum_{i}^{M} E_{\mathrm{loc}}\left(s^{(i)}\right) \qquad E_{\mathrm{loc}}(s) = \sum_{s'} \langle s|\hat{H}|s\rangle \frac{\langle s' \mid \Psi\rangle}{\langle s \mid \Psi\rangle}$$

Gradient for minimization with back propagation

$$\frac{\partial E(\boldsymbol{\theta})}{\partial \theta_p} = 2\,\mathrm{Re}\left[\left\langle \left(E_{\mathrm{loc}}(s) - \langle E_{\mathrm{loc}}(s)\rangle\right) O_p^*(s)\right\rangle\right] \quad O_p(s) = \frac{\partial}{\partial \theta_p} \log \langle s \mid \Psi_{\boldsymbol{\theta}}\rangle = \left\langle s \left| \hat{O}_p \right| s \right\rangle$$

Finding the ground state with neural network quantum states

The variational principle implies that we need to minimize the energy to find the ground state

$$E(\boldsymbol{\theta}) = \frac{\left\langle \Psi_{\boldsymbol{\theta}} | \hat{H} | \Psi_{\boldsymbol{\theta}} \right\rangle}{\langle \Psi_{\boldsymbol{\theta}} \mid \Psi_{\boldsymbol{\theta}} \rangle} \geq E_0$$

Iteratively minimize the energy until convergence is reached

Compute the energy and the gradient

$$E(\boldsymbol{\theta}) \qquad \frac{\partial E(\boldsymbol{\theta})}{\partial \theta_p}$$

Redefine the variational parameters for the next iteration

$$\theta_p^{(s+1)} = \theta_p^{(s)} - \eta \frac{\partial E(\boldsymbol{\theta})}{\partial \theta_p}$$

Learning rate

Advantages of neural-network quantum-states

- Not bounded by the area-law (suitable for 2D)
- Can directly use architectures and resources developed for machine learning
- Certain architectures are equivalent to tensor-networks (Boltzmann machines)

For discussion (3 min)

We have rewritten expectation value of an operator as

$$\langle \hat{O} \rangle = \sum_s P(s) O_{\mathrm{loc}}(s) = \langle O_{\mathrm{loc}}(s) \rangle_P$$

with

$$P(s) = \frac{|\langle \Psi_\theta \mid s \rangle|^2}{\sum_s |\langle \Psi_\theta \mid s \rangle|^2}$$

$$O_{\mathrm{loc}}(s) = \sum_{s'} \left\langle s|\hat{O}|s' \right\rangle \frac{\langle s' \mid \Psi_\theta \rangle}{\langle s \mid \Psi_\theta \rangle}$$

Why is it convenient to rewrite the expectation values in this form to perform Monte Carlo?

What would happen if $P(s)$ **has an oscillating sign in** $\langle \hat{O} \rangle = \sum_s P(s) O_{\mathrm{loc}}(s)$ **?**

Machine learning with neural networks

Some examples of machine learning

Supervised learning

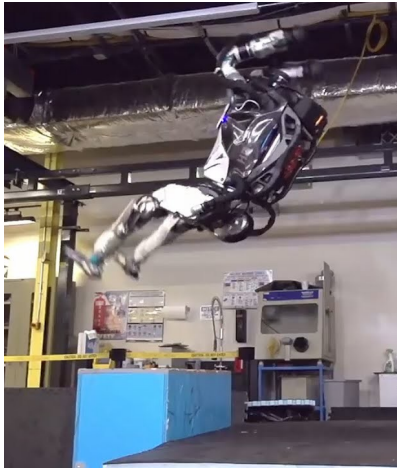
“Dog”

Labeled prediction

Generative learning

Probability Modeling

Reinforcement learning

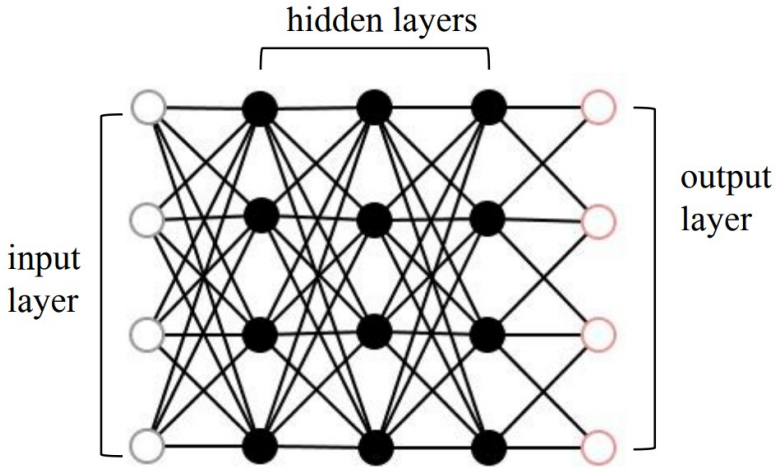


Reward-based decision

(and others)

The basics of deep neural networks

Neural networks are a
family of parametric functions



The parameters are optimized to minimize a certain functional

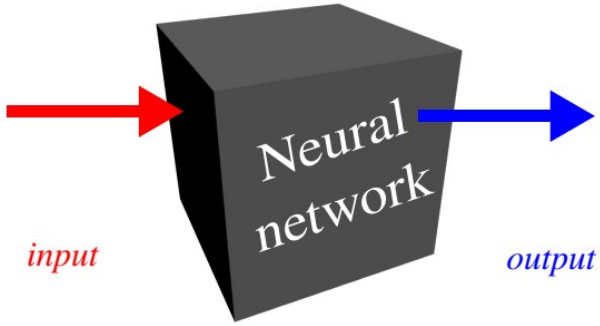
$$\chi = \mathrm{LOSS}[\vec{y}_{\mathrm{real}} - \vec{y}_{\mathrm{predicted}}] = \mathrm{LOSS}[\vec{y}_{\mathrm{real}} - f(\vec{x}_{\mathrm{real}})]$$

For example $\chi \sim |\vec{y}_{\mathrm{real}} - f(\vec{x}_{\mathrm{real}})|^2$

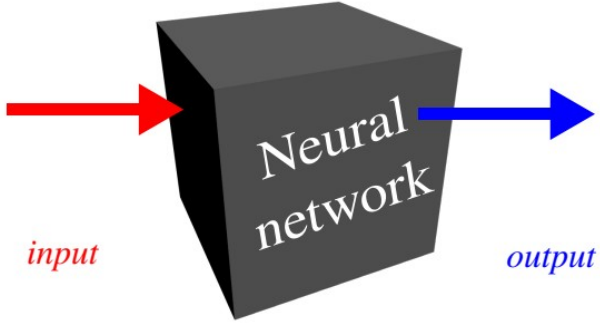
A simple classification problem

How to distinguish between two different animals with an algorithm?

A simple classification problem



“Cat”



“Dog”

A simple classification problem

If we represent the image as a matrix, we just need to find the right function

$\vec{x}_1$

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} & a_{04} & a_{05} & a_{06} & a_{07} \\ a_{10} & a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} & a_{17} \\ a_{20} & a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} & a_{27} \\ a_{30} & a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} & a_{37} \\ a_{40} & a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} & a_{47} \\ a_{50} & a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} & a_{57} \\ a_{60} & a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66} & a_{67} \\ a_{70} & a_{71} & a_{72} & a_{73} & a_{74} & a_{75} & a_{76} & a_{77} \\ a_{80} & a_{81} & a_{82} & a_{83} & a_{84} & a_{85} & a_{86} & a_{87} \\ a_{90} & a_{91} & a_{92} & a_{93} & a_{94} & a_{95} & a_{96} & a_{97} \end{bmatrix}$$

$$f(\vec{x}_1) = \vec{y}_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

“Cat”

$\vec{x}_2$

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} & a_{04} & a_{05} & a_{06} & a_{07} \\ a_{10} & a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} & a_{17} \\ a_{20} & a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} & a_{27} \\ a_{30} & a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} & a_{37} \\ a_{40} & a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} & a_{47} \\ a_{50} & a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} & a_{57} \\ a_{60} & a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66} & a_{67} \\ a_{70} & a_{71} & a_{72} & a_{73} & a_{74} & a_{75} & a_{76} & a_{77} \\ a_{80} & a_{81} & a_{82} & a_{83} & a_{84} & a_{85} & a_{86} & a_{87} \\ a_{90} & a_{91} & a_{92} & a_{93} & a_{94} & a_{95} & a_{96} & a_{97} \end{bmatrix}$$

$$f(\vec{x}_2) = \vec{y}_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

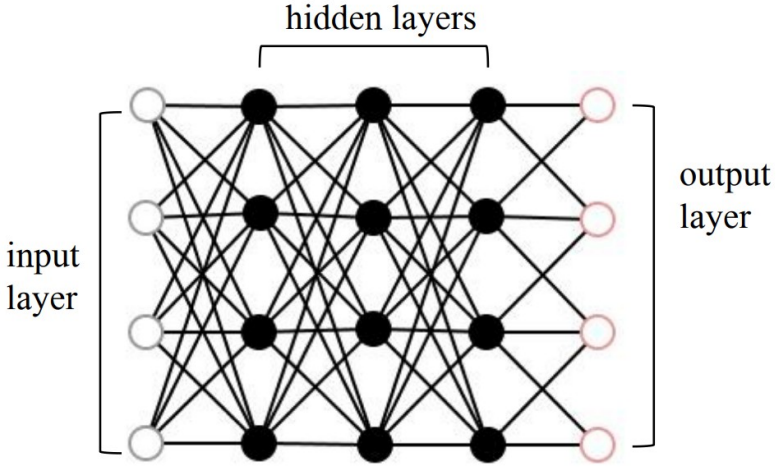
“Dog”

How do we find the function implementing this operation?

Supervised learning in a nutshell

Take a neural-network

Input the image (NxN matrix)
Output a 2D vector

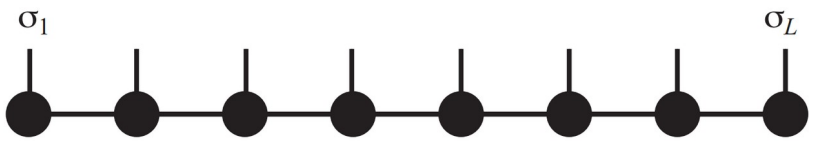


Take a few examples and minimize $\chi = \mathrm{LOSS}[\vec{y}_{\mathrm{real}} - f(\vec{x}_{\mathrm{real}})]$

After the minimization (training), the neural-network will be able to classify new examples

Using a tensor-network to classify images

Tensor-networks allow to parametrice high-dimensional functions

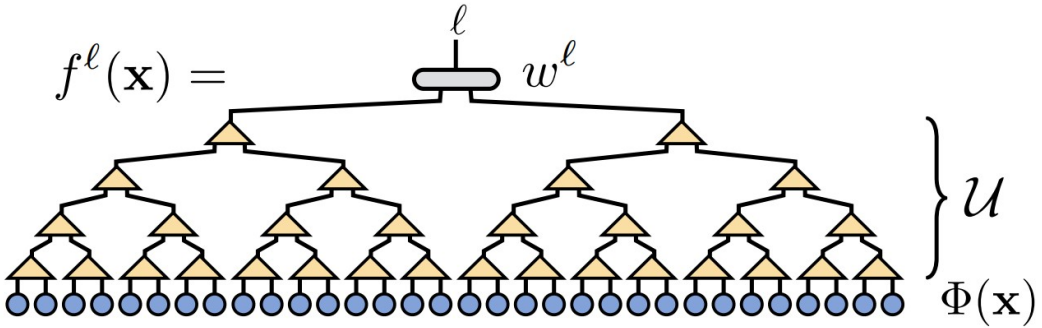


$$c_{s_1,s_2,....,s_L} = M_1^{s_1} M_2^{s_2} ....M_L^{s_3}$$

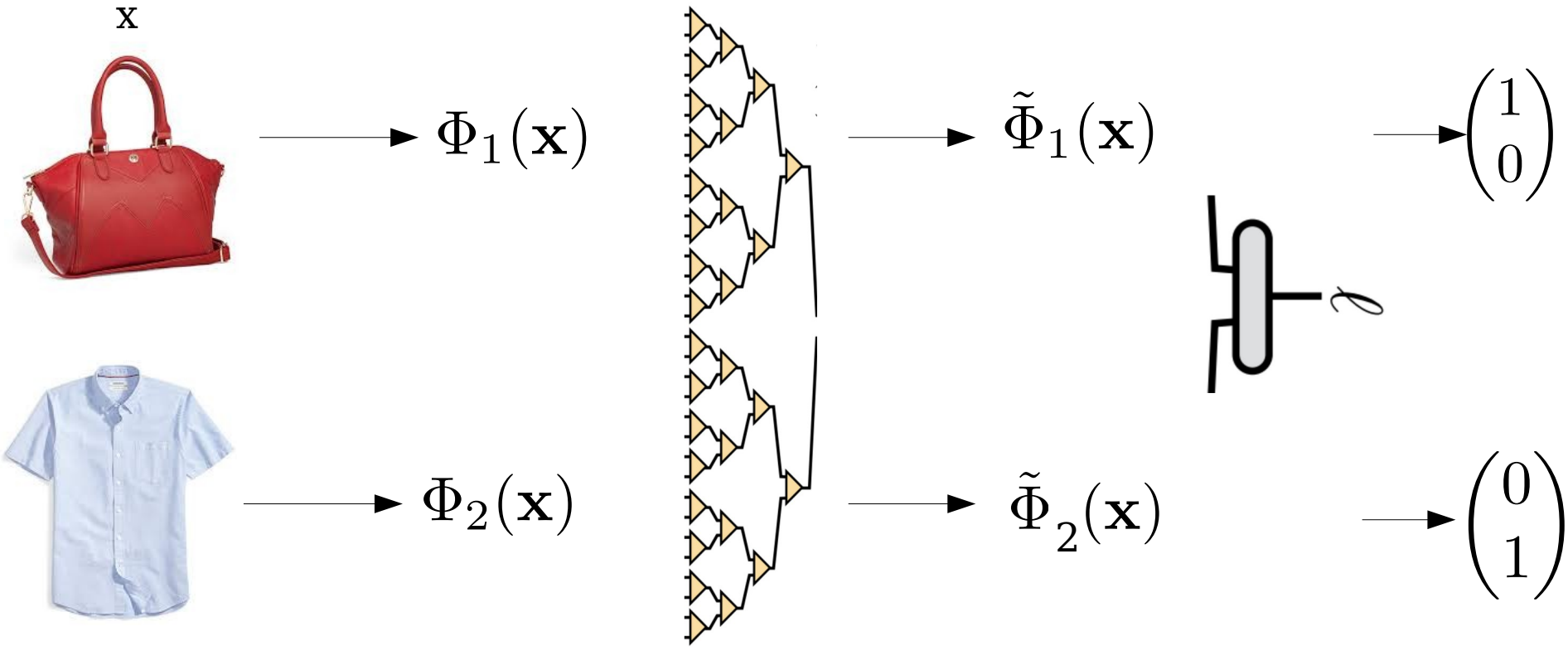
$$|\Psi\rangle = \sum c_{s_1,s_2,...,s_L} |s_1, s_2, ...s_L\rangle$$

Can we use tensor-network architectures “as if” they were neural networks?

Quantum many-body inspired machine learning



Using a tensor-network to classify images



$$\tilde{\Phi}^{t_1 t_2}(\mathbf{x}_j) = \sum_{s_1, s_2, \ldots, s_N} \mathcal{U}^{t_1 t_2}_{s_1 s_2 \cdots s_N} \Phi^{s_1 s_2 \cdots s_N}(\mathbf{x}_j) \ .$$

The fashion MNIST dataset classified with tensor-networks

Results

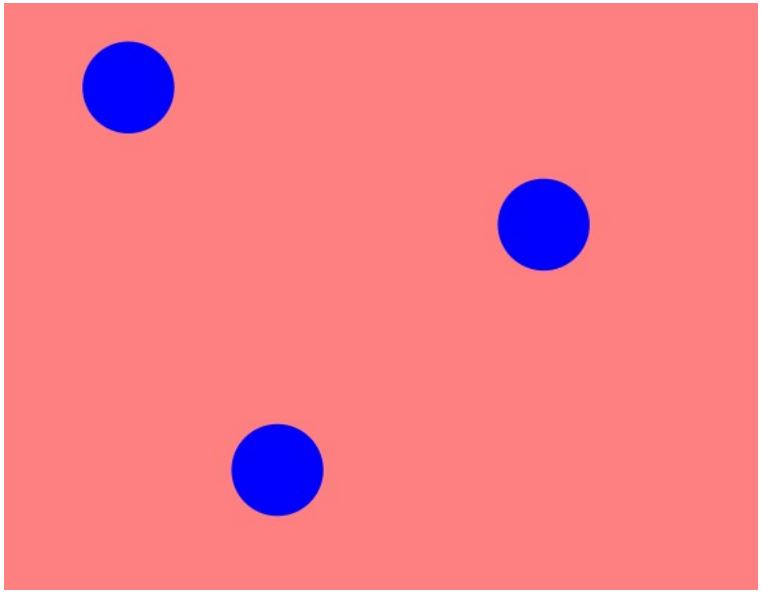
95.38% accuracy in training

88.97% accuracy in testing

Machine learning in density functional theory

Previously, in session 9: Density functional theory

Take the positions of all the atoms



And find an “accurate” single-particle effective Hamiltonian

Two major challenges

Problem #1: There is an unknown part of the formalism (exchange correlation)

Problem #2: Solving the Kohn-Sham equations becomes too demanding for large systems

The many-electron problem

The Hamiltonian for electrons in a solid

$$H_{\mathrm{el}} = -\frac{1}{2}\sum_{j=1}^{N}\nabla_j^2 - \sum_{j=1}^{N}\sum_{l=1}^{M}\frac{Z_l}{\tilde{r}_{jl}} + \sum_{j=1}^{N}\sum_{k>j}^{N}\frac{1}{r_{jk}}$$

Has an associated electronic density

$$\rho(\boldsymbol{r}) = N\int \mathrm{d}^3\boldsymbol{r}_2 \cdots \int \mathrm{d}^3\boldsymbol{r}_N \left|\Psi\left(\boldsymbol{r}, \boldsymbol{r}_2, \cdots, \boldsymbol{r}_N\right)\right|$$

The Hohenberg-Kohn theorem

For the ground state of a system, there is a one-to-one relation between the electronic density and the many-body wavefunction (Hohenberg-Kohn theorem)

$$\rho_0 \leftrightarrow |\Psi\rangle$$

The total energy can be written as a functional of the electronic-density

$$E(\rho_0)$$

The ground state energy can be computed if we know the functional (which we do not)

The Kohn-Sham equations

We do not know the “true” functional for density-functional theory

Let us take an “imaginary” non-interacting electron gas, with the same density as the real one

$$F_{\mathrm{HK}}[\rho] = T_{\mathrm{S}}[\rho] + J[\rho] + E_{\mathrm{XC}}[\rho]$$

where

$$T_{\mathrm{S}} = -\frac{1}{2}\sum_{j=1}^{N} \left\langle \psi_j \left| \nabla^2 \right| \psi_j \right\rangle$$

Mean-field electronic interaction

And this is approximated (LDA, GGA, metaGGA, etc)

Problem #1: We do not know the exchange-correlation functional, could it be machine learned?

The Kohn-Sham equations

Given a certain density-functional $F_{KS}(\rho)$

An effective single-particle Hamiltonian must be solved, taking the form

$$H_{KS}|\psi_n\rangle = \epsilon_n|\psi_n\rangle$$

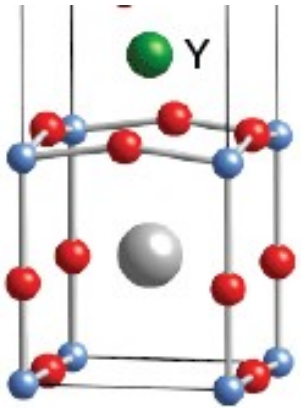
Where the Hamiltonian is obtained as a functional derivative

$$(H_{KS} - \epsilon_n)|\psi_n\rangle = \frac{\delta F_{KS}}{\delta|\psi_n\rangle} = 0$$

The Kohn-Sham equations

Define your crystal
(atomic positions)

$$V(\mathbf{r})$$



Electronic density

$$\rho(\mathbf{r})$$

Hamiltonian

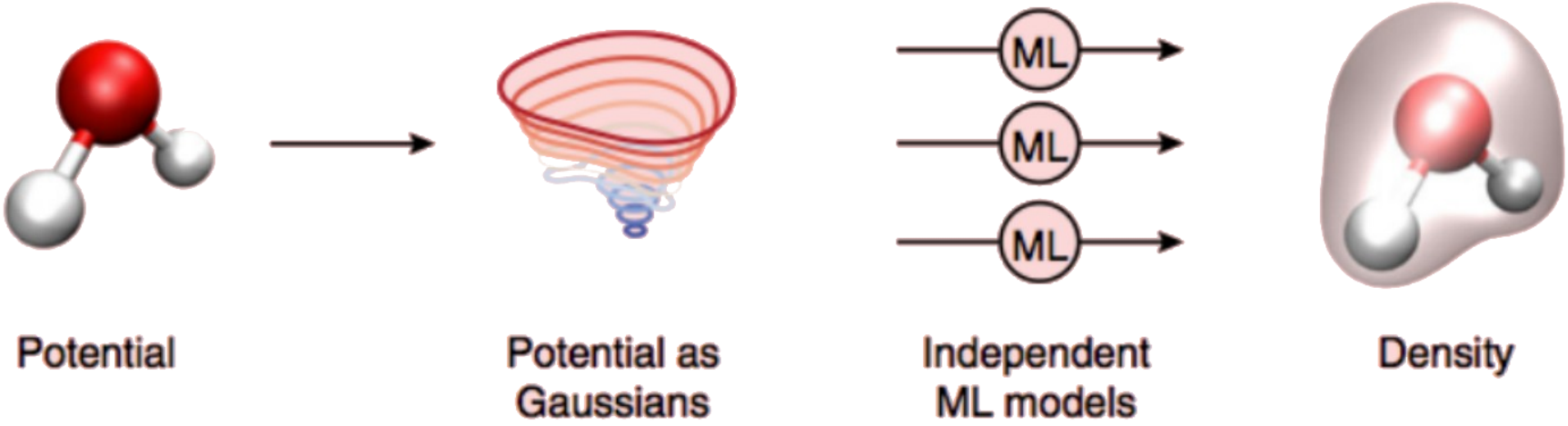
$$H_{KS}(\rho)$$

KS eigenvectors

$$|\psi_n\rangle$$

Problem #2: Solving the Kohn-Sham equations becomes too demanding for large systems

Machine learning the potential-density functional



The neural network takes as input the potential, and as output the density

Machine learning the exchange-correlation functional

In the Kohn-Sham equations

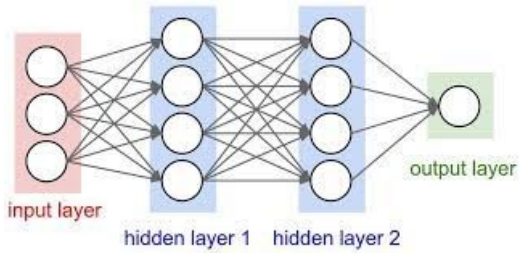
$$F[\rho] = T[\rho] + J[\rho] + E_{XC}[\rho]$$

The exchange-correlation functional is approximated (LDA, GGA, metaGGA, etc)

Instead of using an approximation, the functional can also be encoded as a neural-network

$$E_{XC} \equiv E_{XC}^{NN}$$

$$\frac{\delta E_{XC}}{\delta \rho} = V_{XC} \equiv V_{XC}^{NN}$$



Using machine learning in quantum materials

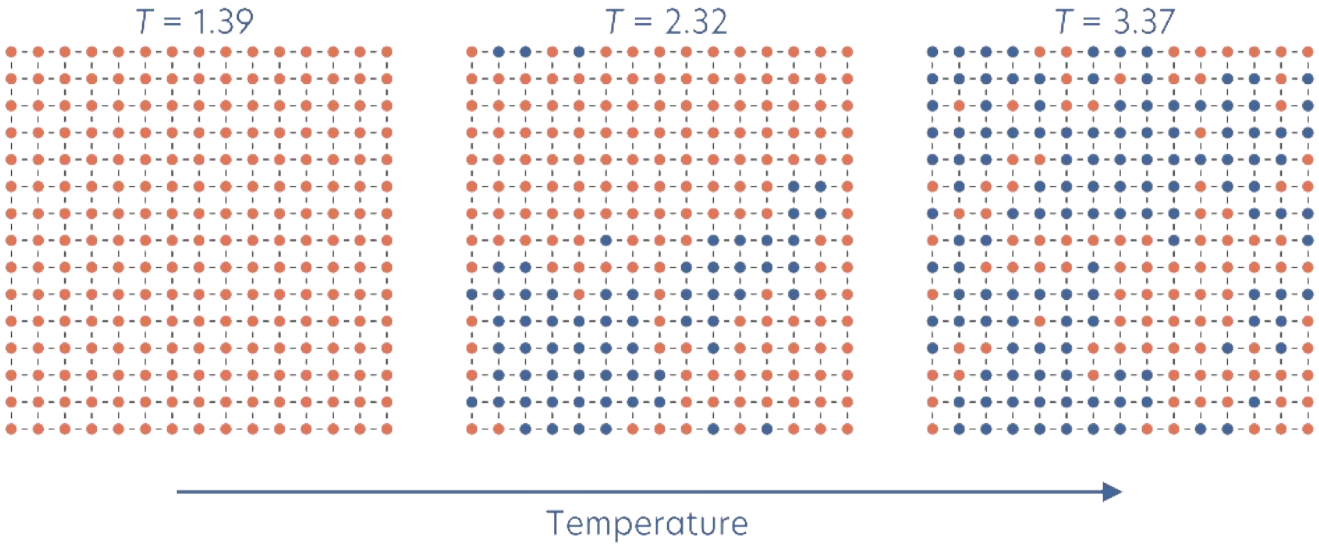
How use machine learning in combination with conventional methods

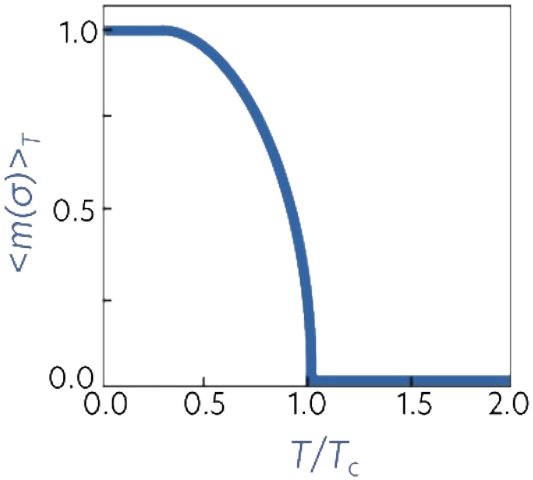
- Phase classification in the Ising model (trivial) and Ising gauge theory (non-trivial)
- Theoretical and experimental Hamiltonian learning

The Ising model

Take the Ising model and evaluate realizations at different temperatures

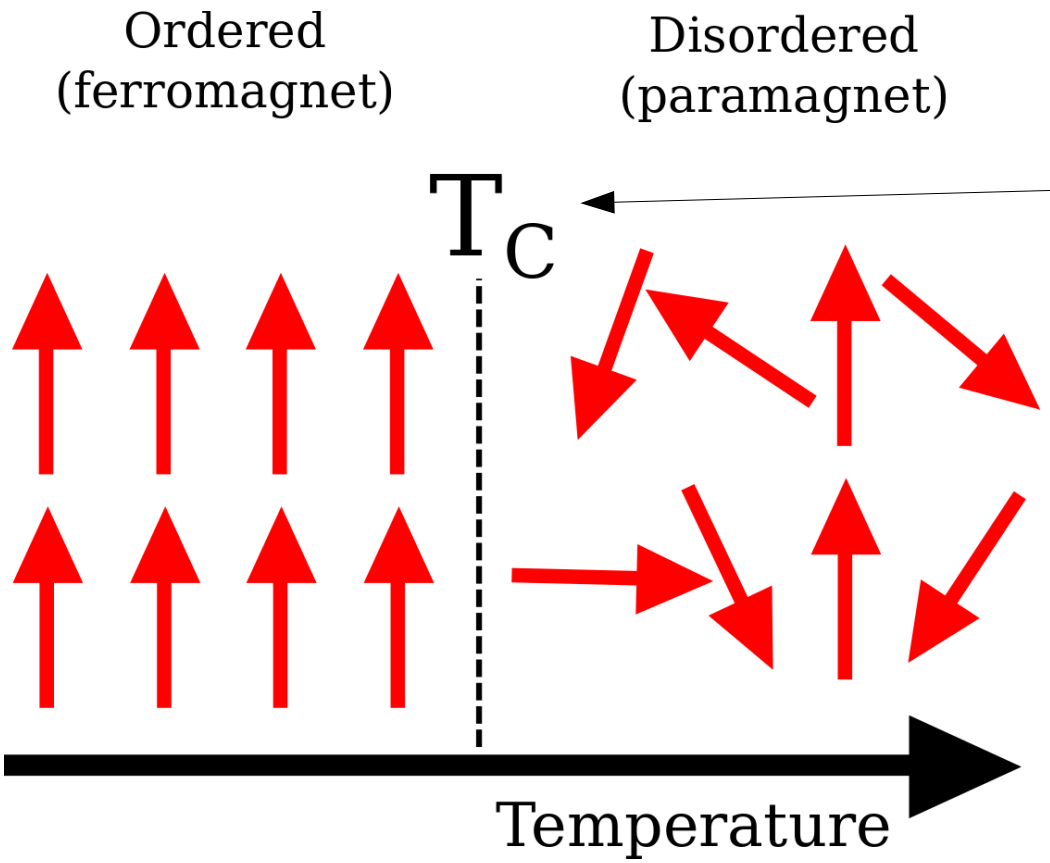
$$H(\sigma) = -J \sum_{\langle i,j \rangle} \sigma_i \sigma_j$$





(This is of course a very simple problem)

Classical phase transitions



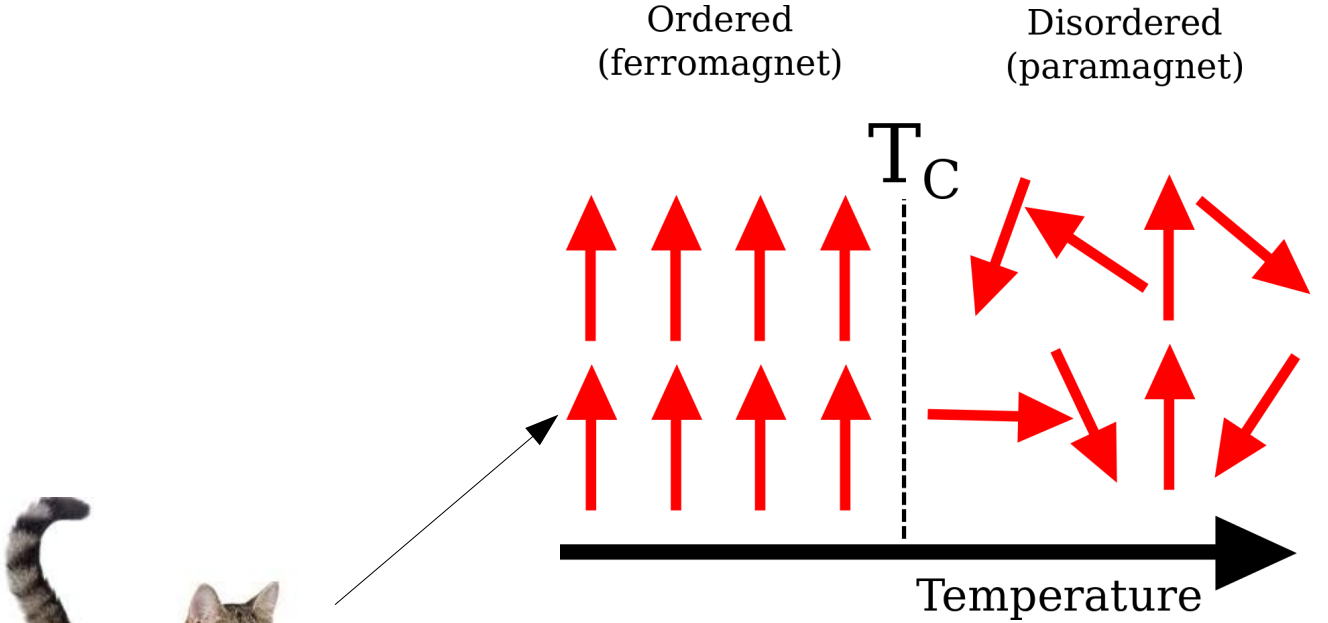
Critical temperature

Can we use machine learning to find phase transitions?

2D Ising model

$$H = -J \sum_{\langle ij \rangle} s_i s_j$$

Classical phase transitions

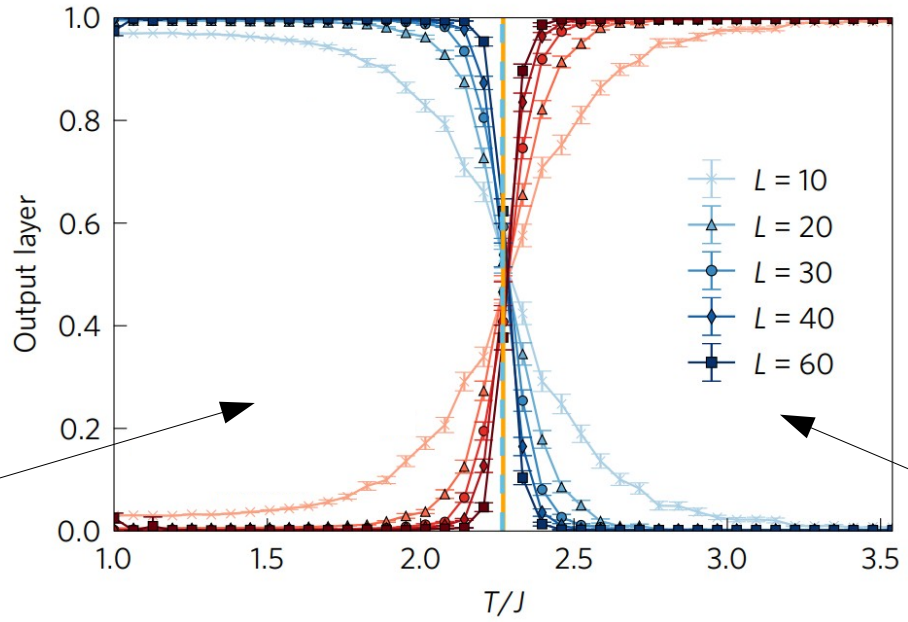


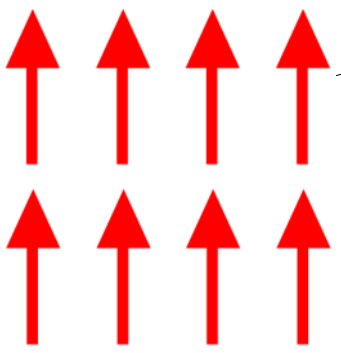
$$H = -J \sum_{\langle ij \rangle} s_i s_j$$

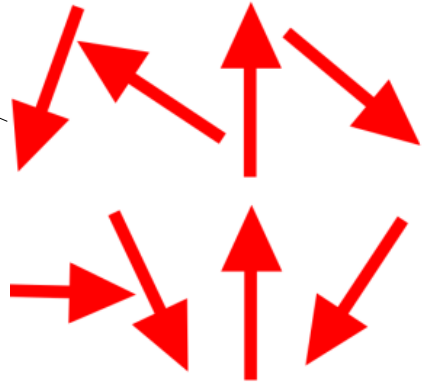
We can use a “classification algorithm” as did before

Classical phase transitions

$$H = -J \sum_{\langle ij \rangle} s_i s_j$$







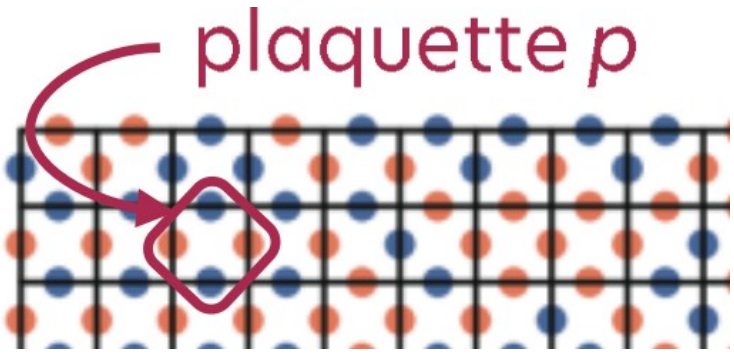
A neural-network learns to classify the two phases

A more challenging case: Topological phase transitions

Topological phase transitions do not have a local order parameter

Ising gauge theory $$H(\boldsymbol{\sigma}) = -J \sum_{p} \prod_{i \in p} \sigma_i$$

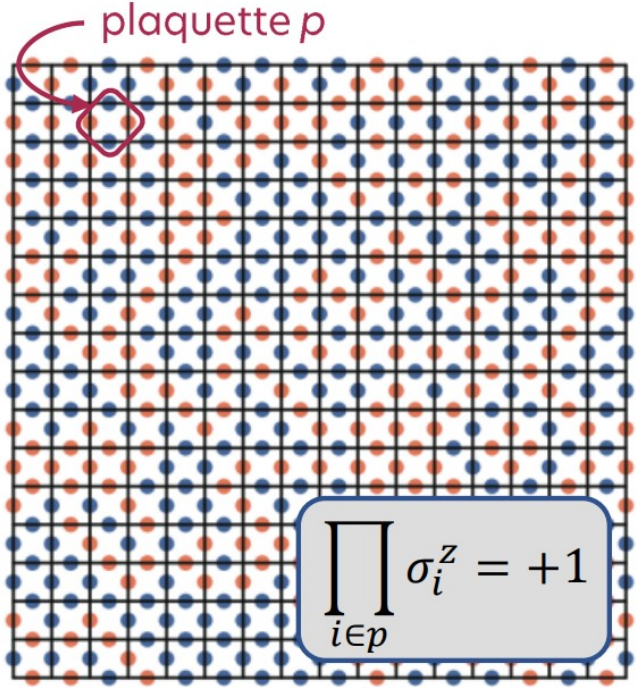
At low temperatures, the ground state fufills $$\prod_{i \in p} \sigma_i = 1$$



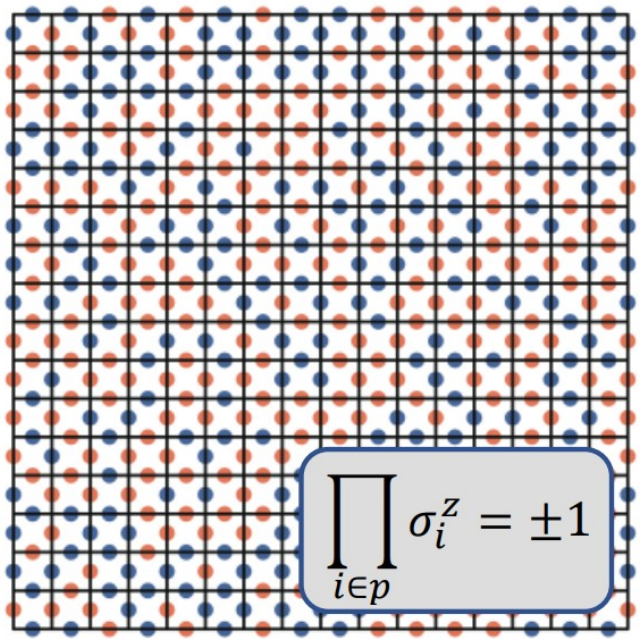
At high temperatures, the constraint is not fufilled

Topologically different phase in Ising gauge theory

$T \to 0$



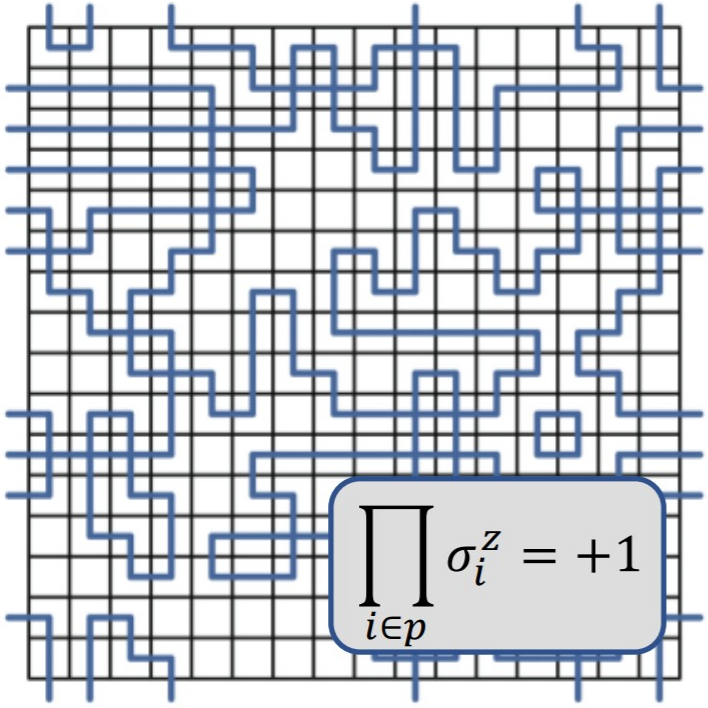
$T \to \infty$



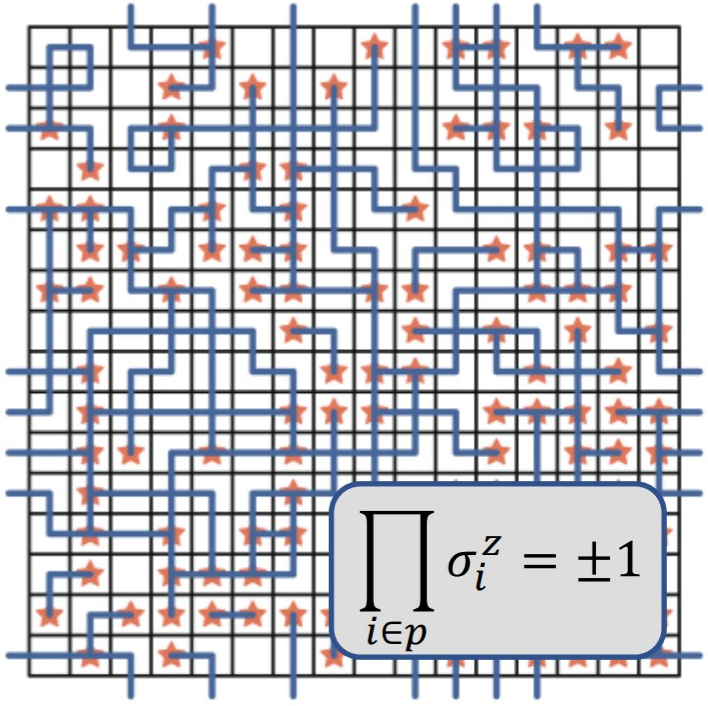
Low temperature and high temperature are topologically different, but the difference is not obvious

Dual representation of the Ising gauge theory

$T \to 0$



$T \to \infty$



Link sites with the same orientation. The constraint leads to all loops closed.

The problem of Hamiltonian learning

How can we go from observables to the Hamiltonian?

Our usual workflow during the course

Solve the Hamiltonian

Compute an observable

$$H \longrightarrow \langle \Psi | O | \Psi \rangle$$

The problem we often find experimentally

Measure an observable

Determine its Hamiltonian

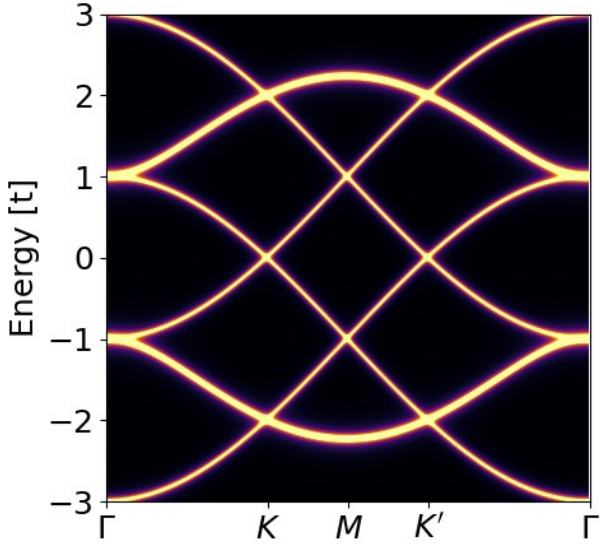
$$\langle \Psi | O | \Psi \rangle \longrightarrow H$$

How can we perform this task?

A computational example

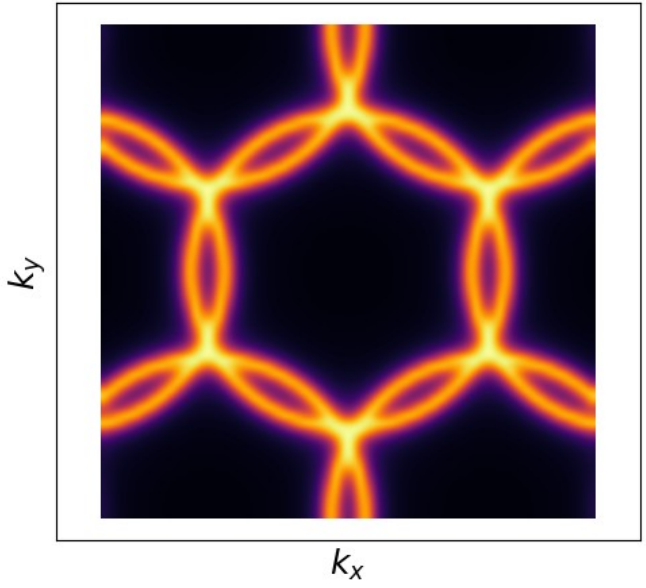
How can we obtain a tight binding model from observables like

The band structure



$$H = \sum_{ij} t_{ij} c_i^\dagger c_j$$

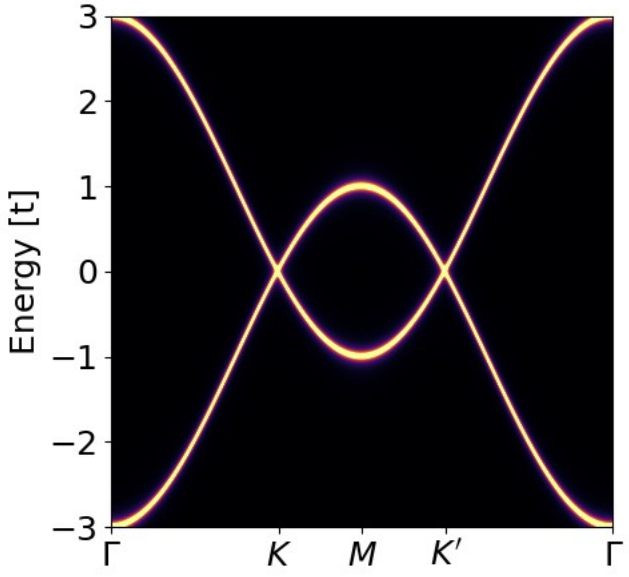
The Fermi surface at E=-2t



A computational example

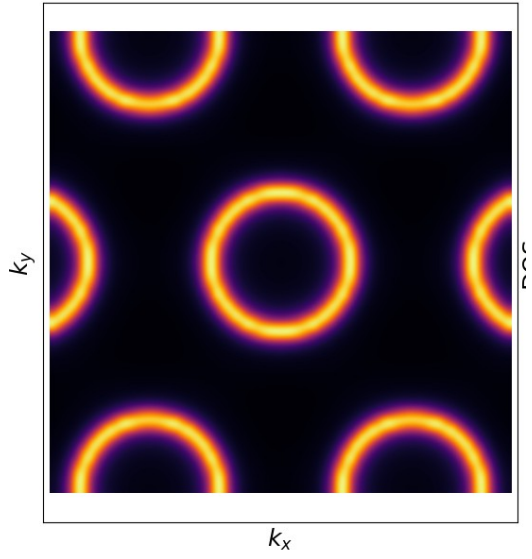
How can we obtain a tight binding model from observables like

The band structure



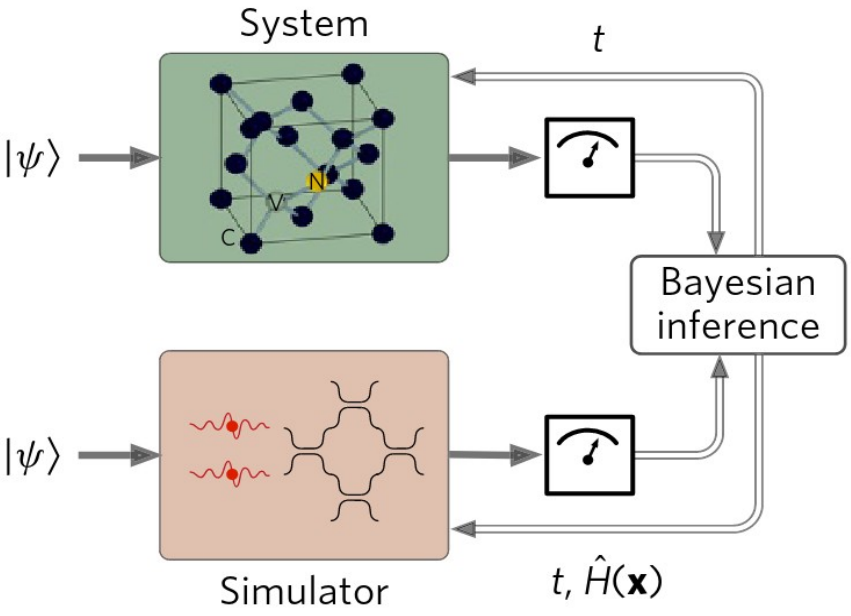
$$H = \sum_{ij} t_{ij} c_i^\dagger c_j$$

The Fermi surface at E=-2t

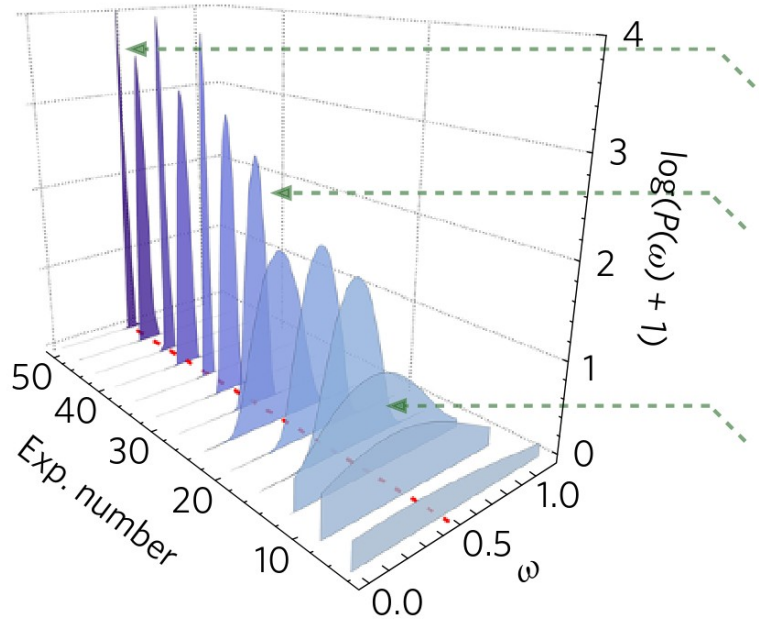


Experimental Hamiltonian learning

Theory and experiment workflow



Learning of the spin Hamiltonian



Dynamics of an NV-center

Take home

- Methods to parametrize wavefunctions can be used in machine learning
- Neural-networks allow parametrizing many-body wavefunctions
- Machine learning methods can be combined with conventional quantum materials methodologies