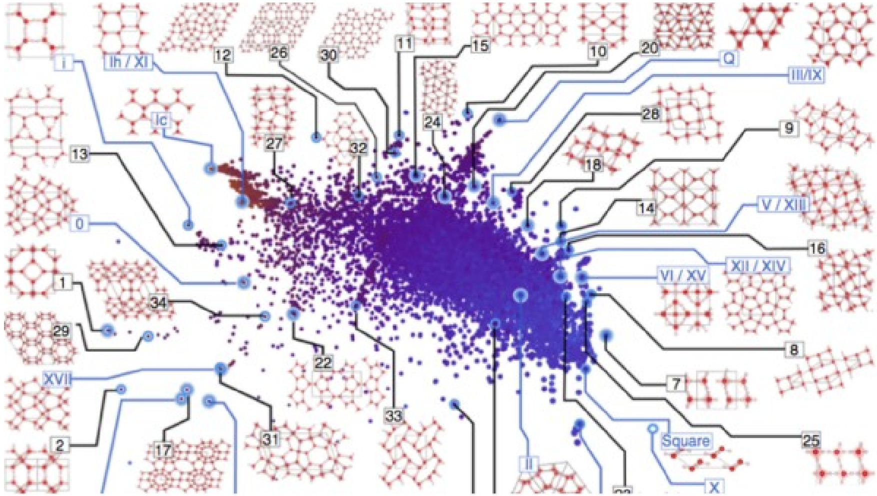
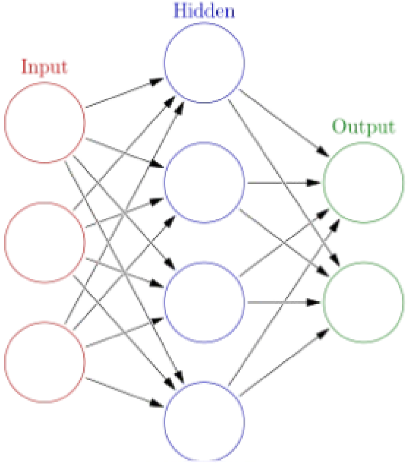
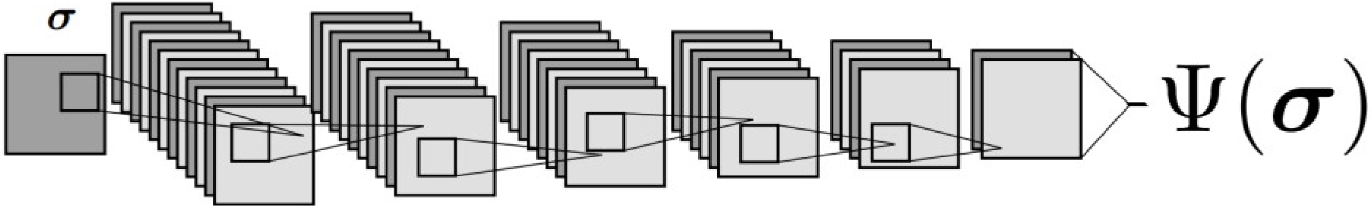
Machine learning in quantum materials







May 24th 2021

Today’s learning outcomes

- Some problems in quantum matter can be rephrased so that they can be tackled with AI
- Neural-networks can be used as generic functional approximators

Some paradigmatic examples of machine learning

Supervised learning

“Dog”

Classification

Unsupervised learning

Clustering & generation

Reinforcement learning

Decision making

Unsupervised learning

Generating new faces of humans

https://thispersondoesnotexist.com/

Unsupervised learning

Generating new faces of humans



https://thispersondoesnotexist.com/

Generating new faces of cats



https://thiscatdoesnotexist.com/

Reinforcement learning

Dancing robots https://www.youtube.com/watch?v=fn3KWM1kuAw

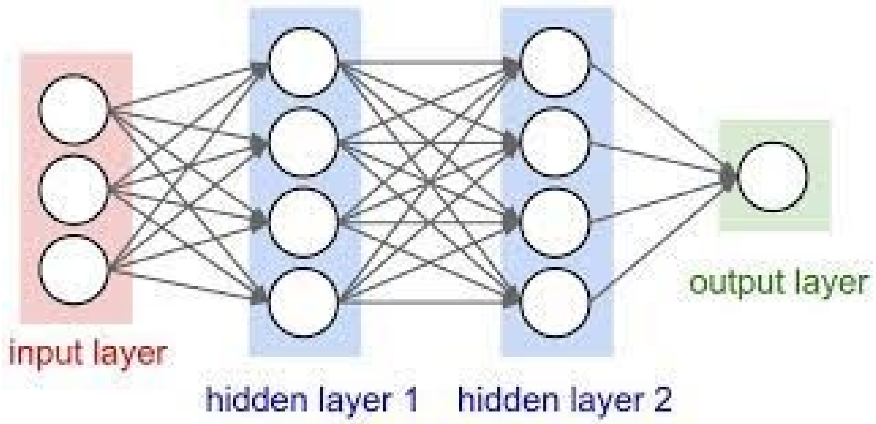
Today’s plan

- ML for spontaneous symmetry breaking
- ML for density functional theory
- ML for quantum many-body problems
- Many-body methods for ML

Neural networks

The basics of deep neural networks

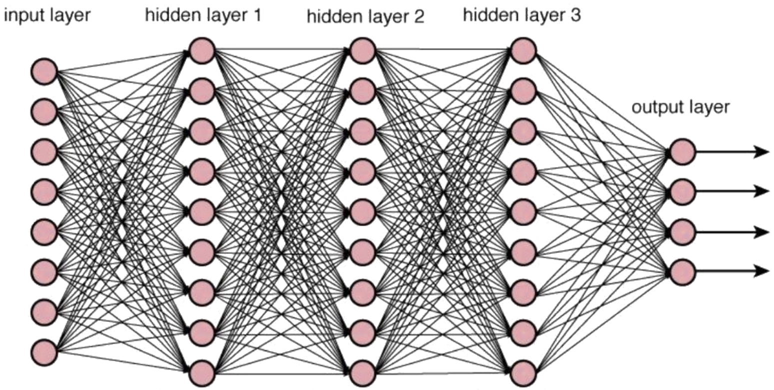
Deep neural networks are “general function approximators”



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

The basics of deep neural networks

Neural networks are a family of parametric functions (bias and weights)



The parameters are optimized to minimize a certain functional

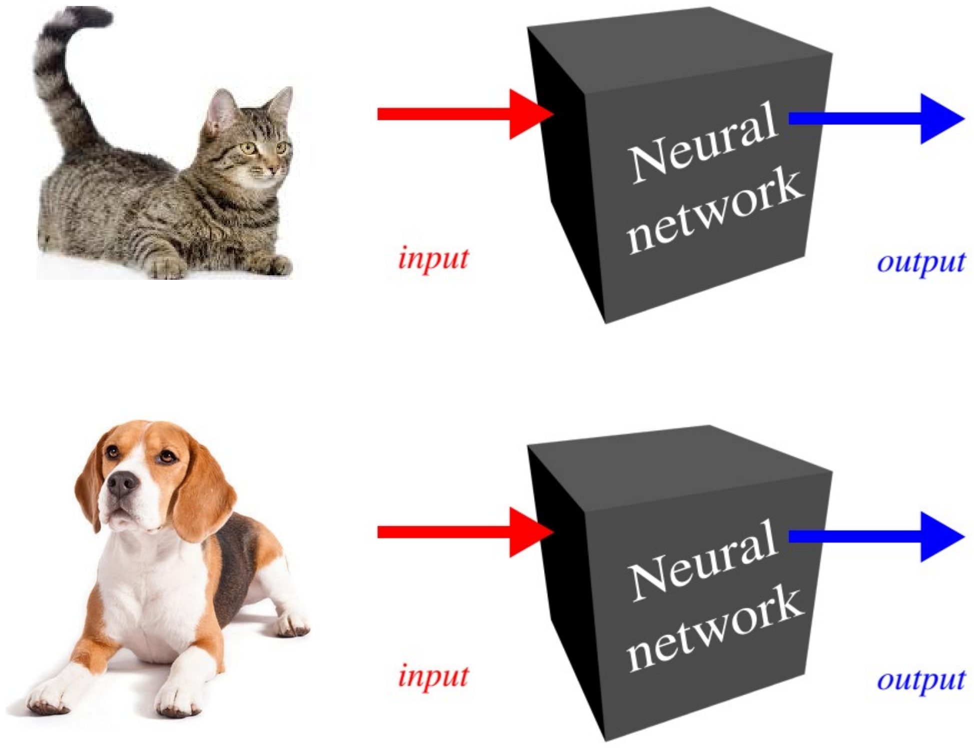
$$\chi = \mathrm{LOSS}[\vec{y}_{\mathrm{real}} - \vec{y}_{\mathrm{predicted}}] = \mathrm{LOSS}[\vec{y}_{\mathrm{real}} - f(\vec{x}_{\mathrm{real}})]$$

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

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

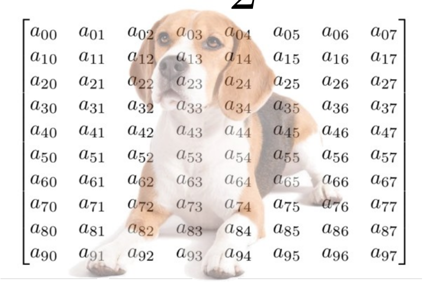
$$\vec{x}_1$$

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

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

“Cat”

$$\vec{x}_2$$



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

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

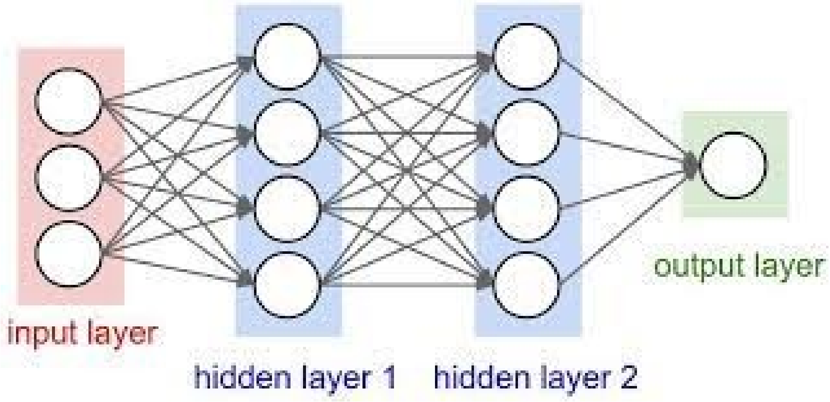
“Dog”

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

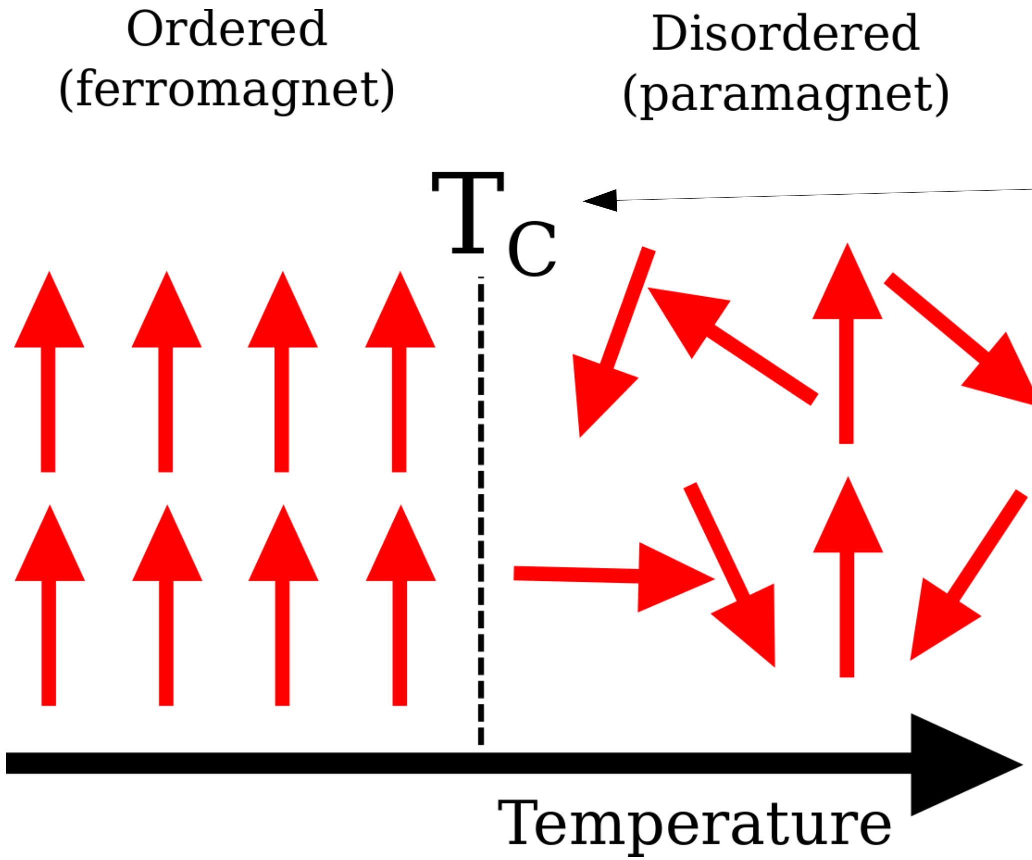


Take a few examples and minimize $$\chi = \mathrm{LOSS}[\vec{y}_{\mathrm{real}} - f(\vec{x}_{\mathrm{real}})]$$

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

Phase classification

Classical phase transitions



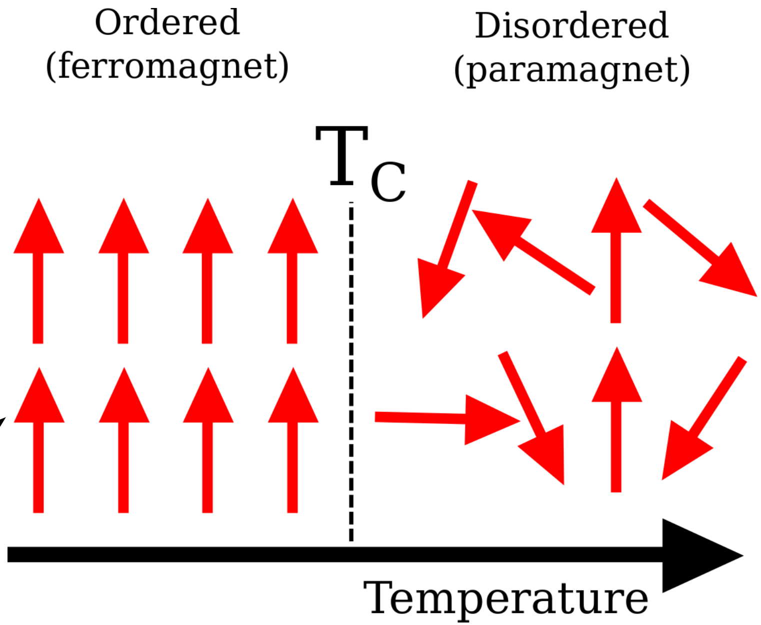
Critical temperature

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

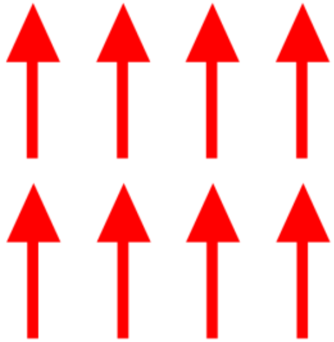


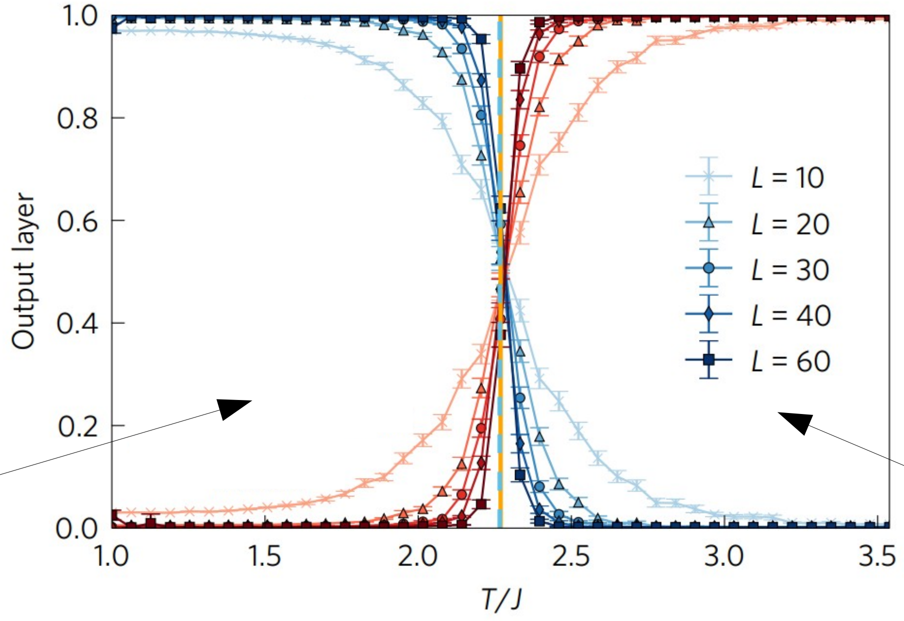
$$H = -J \sum_{\langle ij \rangle} s_i s_j$$

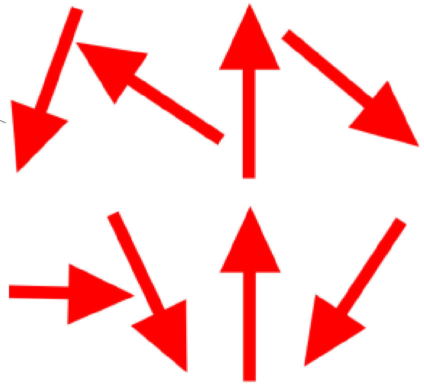
We can use a “classification algorithm” as did before

Classical phase transitions

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







The neural-network learns to classify the two phases

Machine learning in density functional theory

The many-electron problem

The Hamiltonian for electrons in a solid

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

Has an associated electronic density

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

How do we compute the density and energy without computing the many-body wavefunction?

Rminder: The Hohenberg-Kohn theorem

For the ground state of a system, there is a one-to-one relation between the electronic density and the external electrostatic potential

$$H = T + V$$

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

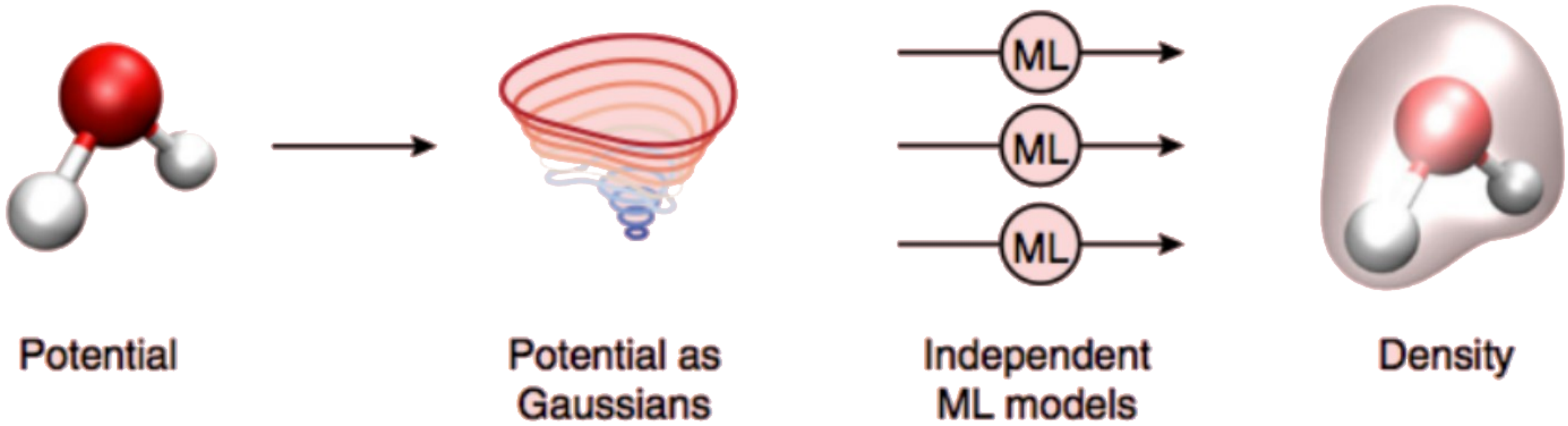
ρ_0 Ground state electronic density

$E[V] \leftrightarrow E[\rho]$ However, we do not know what is the functional

Can we use neural networks to parametrize the functional?

- Machine learning potentials
- Machine learning exchange correlation functional

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

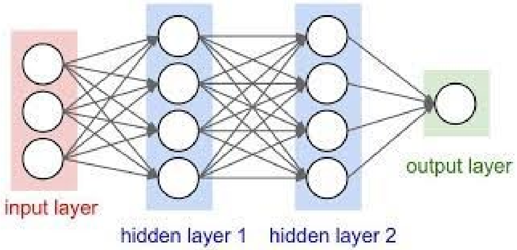
$$F[\rho] = T[\rho] + J[\rho] + E_{XC}[\rho]$$

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

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

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



Neural network quantum states

The quantum many-body problem

Let us go back to a simple many-body problem

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

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

What is the dimension of the Hilbert space?

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

What is the dimension of the Hilbert space?

$$d = 2^L$$

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

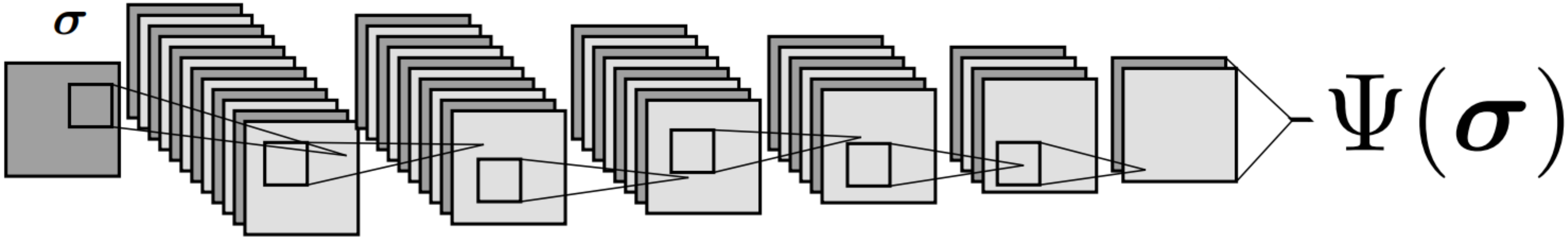
Is there an efficient way of storing so many coefficients?

Neural-network quantum states

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

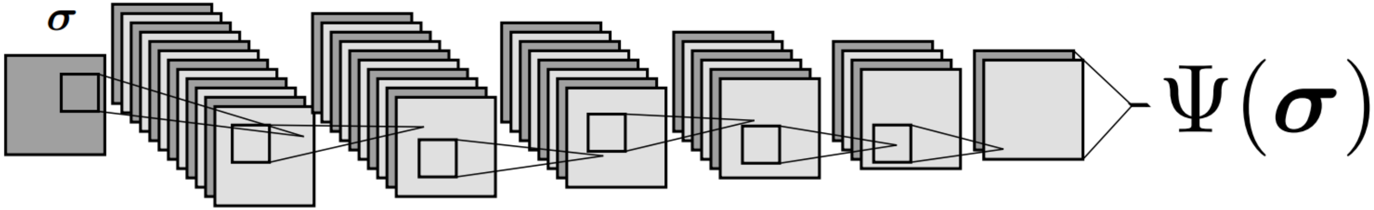
$$c_{s_1, s_2, \ldots, s_L} = f(s_1, s_2, \ldots, s_L)$$

Deep neural network



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

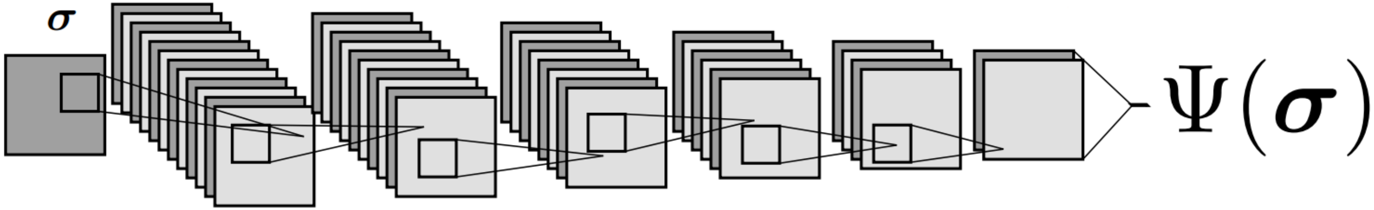
Neural-network quantum states



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

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

Neural-network quantum states



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

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

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

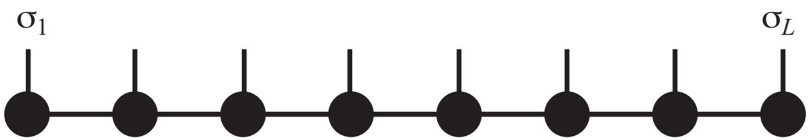
Advantages of neural-network quantum-states

- Not bounded by the area-law (suitable for 2D)
- Can potentially harvest all the power of deep learning
- Can outperform any other quantum many-body method
- Certain architectures are equivalent to tensor-networks

Machine learning with quantum-many body methods

Using a tensor-network to classify images

Tensor-networks allow to parametrice high-dimensional functions

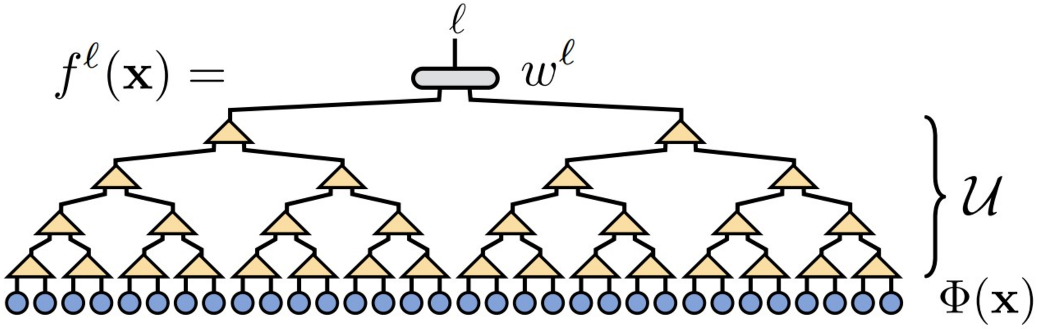


$$c_{s_1,s_2,\dots,s_L} = M_1^{s_1} M_2^{s_2} \dots M_L^{s_3}$$

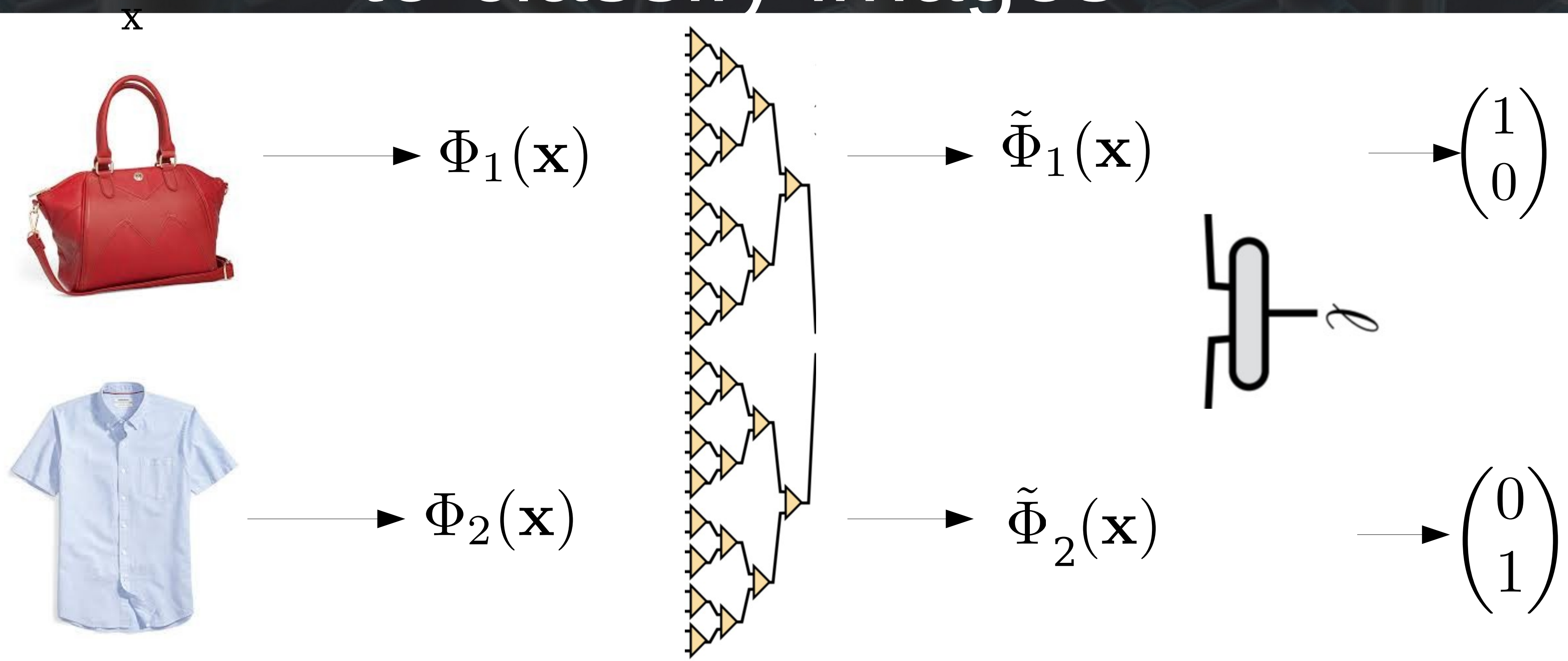
$$|\Psi\rangle = \sum c_{s_1,s_2,\dots,s_L} |s_1, s_2, \dots s_L\rangle$$

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

Take home

- Machine learning methods can be used to solve some problems in quantum matter
- Neural-networks can be used to parametrize functionals used in quantum physics

Reading material

- Machine learning and the physical sciences, Carleo et at, pages 22-35
- A very interesting course on ML for scientists in

https://ml-lectures.org/docs/