MACHINE LEARNING PHASES OF MATTER

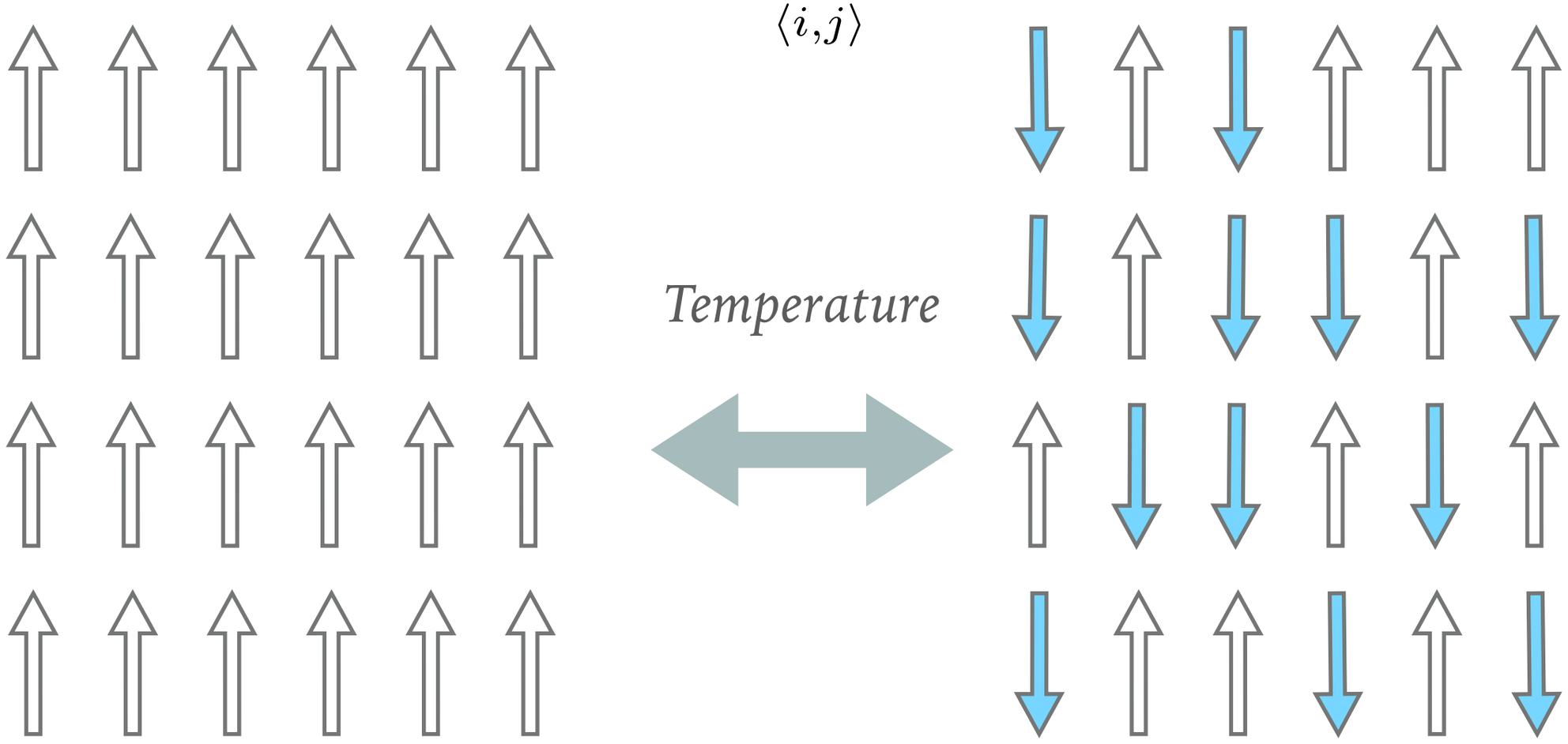
Juan Carrasquilla
D-Wave systems

PHASES, PHASE TRANSITIONS, AND THE ORDER PARAMETER

PHASES, PHASE TRANSITIONS, AND THE ORDER PARAMETER

Ising ferromagnet in two dimensions

$$E = -J \sum_{\langle i,j \rangle} \sigma_i \sigma_j$$

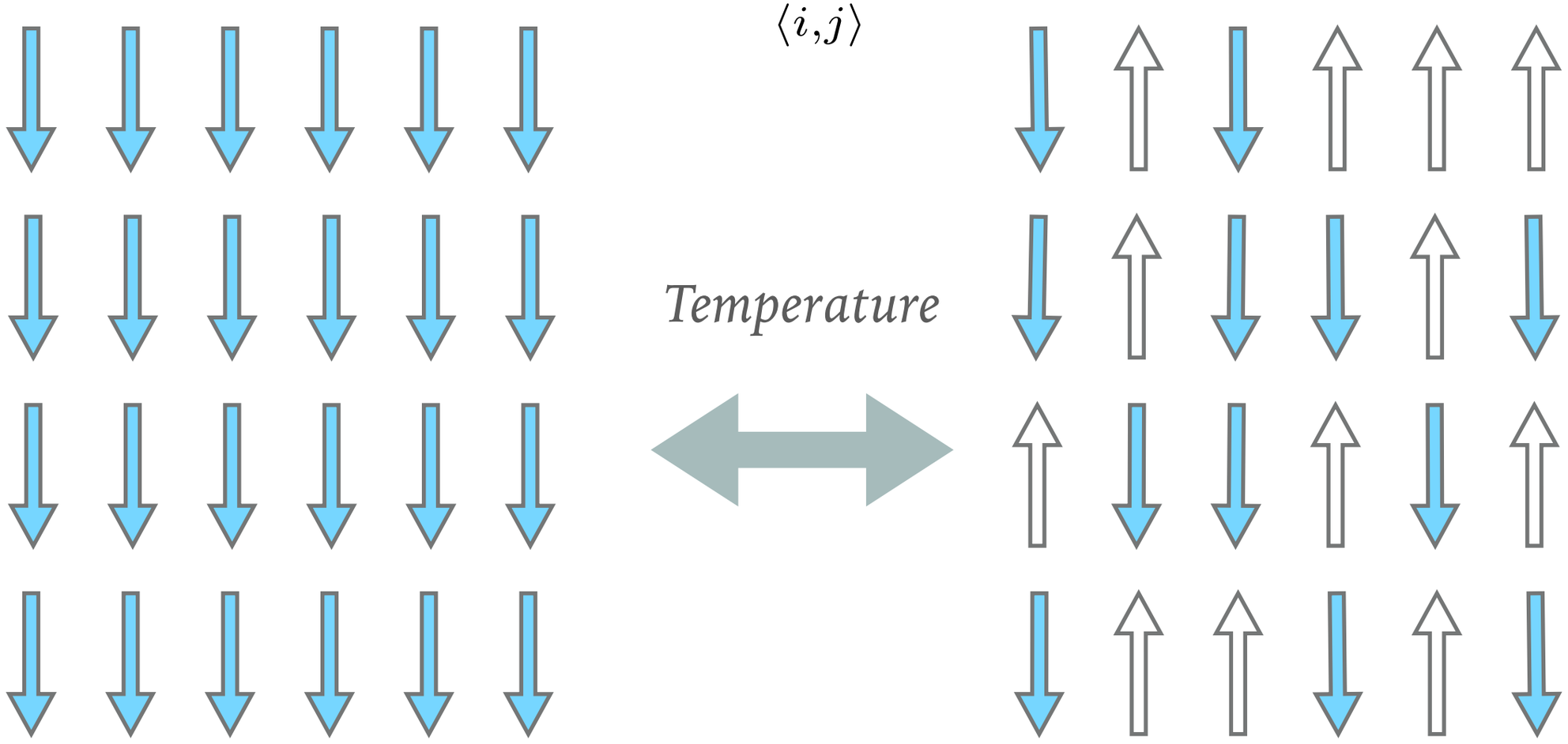


Lars Onsager Phys. Rev. 65, 117

PHASES, PHASE TRANSITIONS, AND THE ORDER PARAMETER

Ising ferromagnet in two dimensions

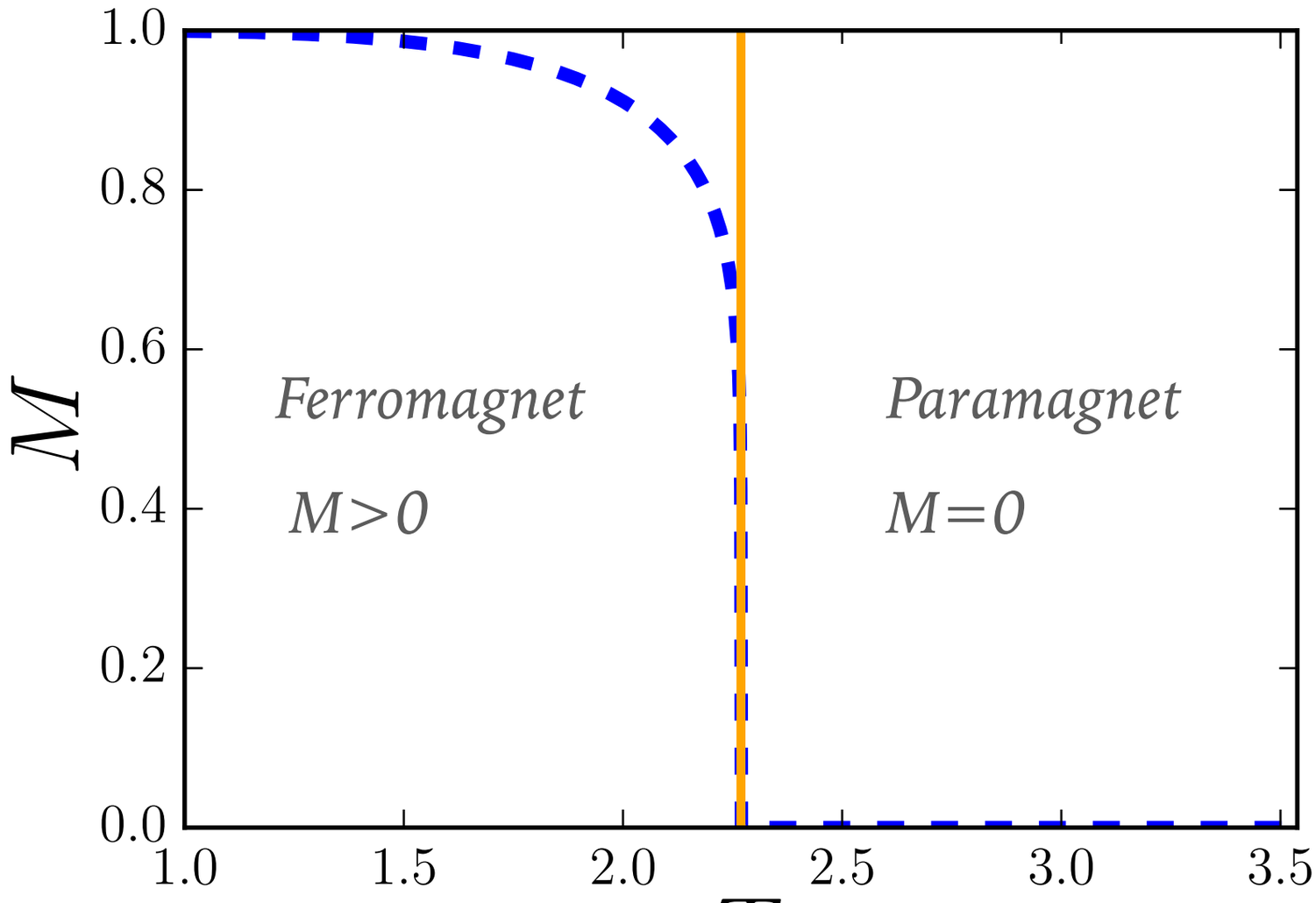
$$E = -J \sum_{\langle i,j \rangle} \sigma_i \sigma_j$$



Ferromagnet *Paramagnet*

PHASES, PHASE TRANSITIONS, AND THE ORDER PARAMETER

Ferromagnetic transition: order parameter



It is a measure of the degree of order in the system

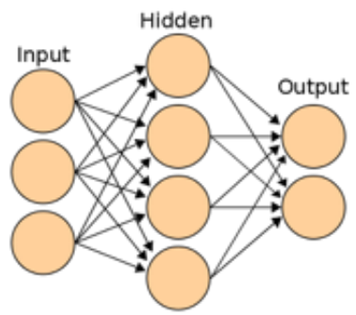
$$M = \frac{1}{N}\sum_i \langle \sigma_i \rangle, \qquad \sigma_i = \pm 1$$

Lars Onsager Phys. Rev. 65, 117

LEARNING PHASES OF MATTER: INSPIRATION FROM THE FLUCTUATIONS IN HANDWRITTEN DIGITS AND SUPERVISED LEARNING

INSPIRATION: FLUCTUATIONS HANDWRITTEN DIGITS (MNIST)

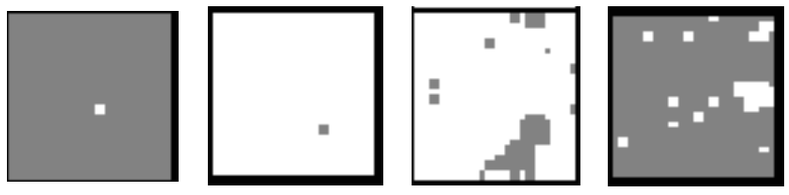
$5 = 5 + \textit{Fluctuations}$



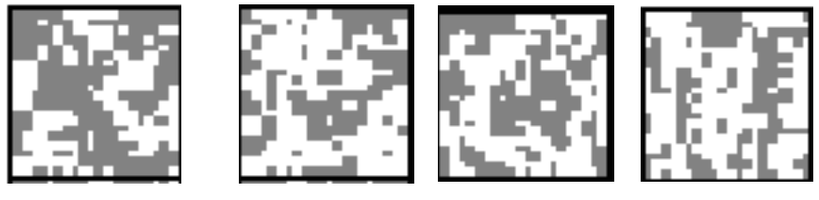
5

ML community has developed powerful learning algorithms based on artificial neural networks

FM phase

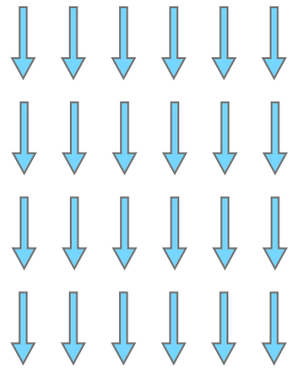


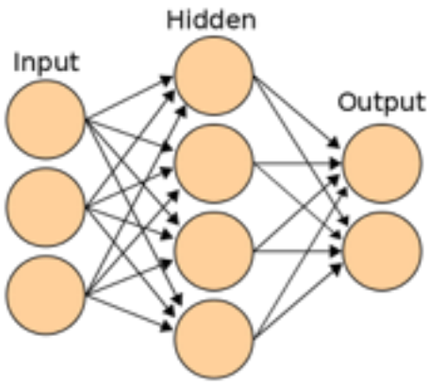
High T phase



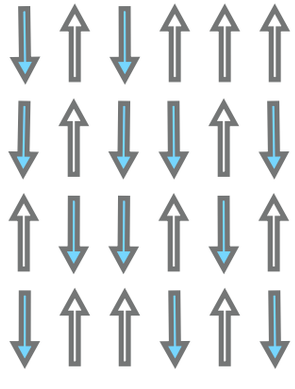
gray = spin up

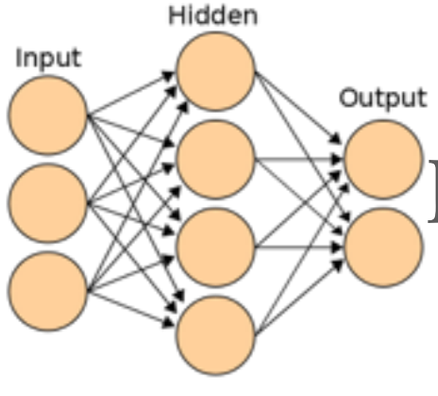
white = spin down





FM [0,1]



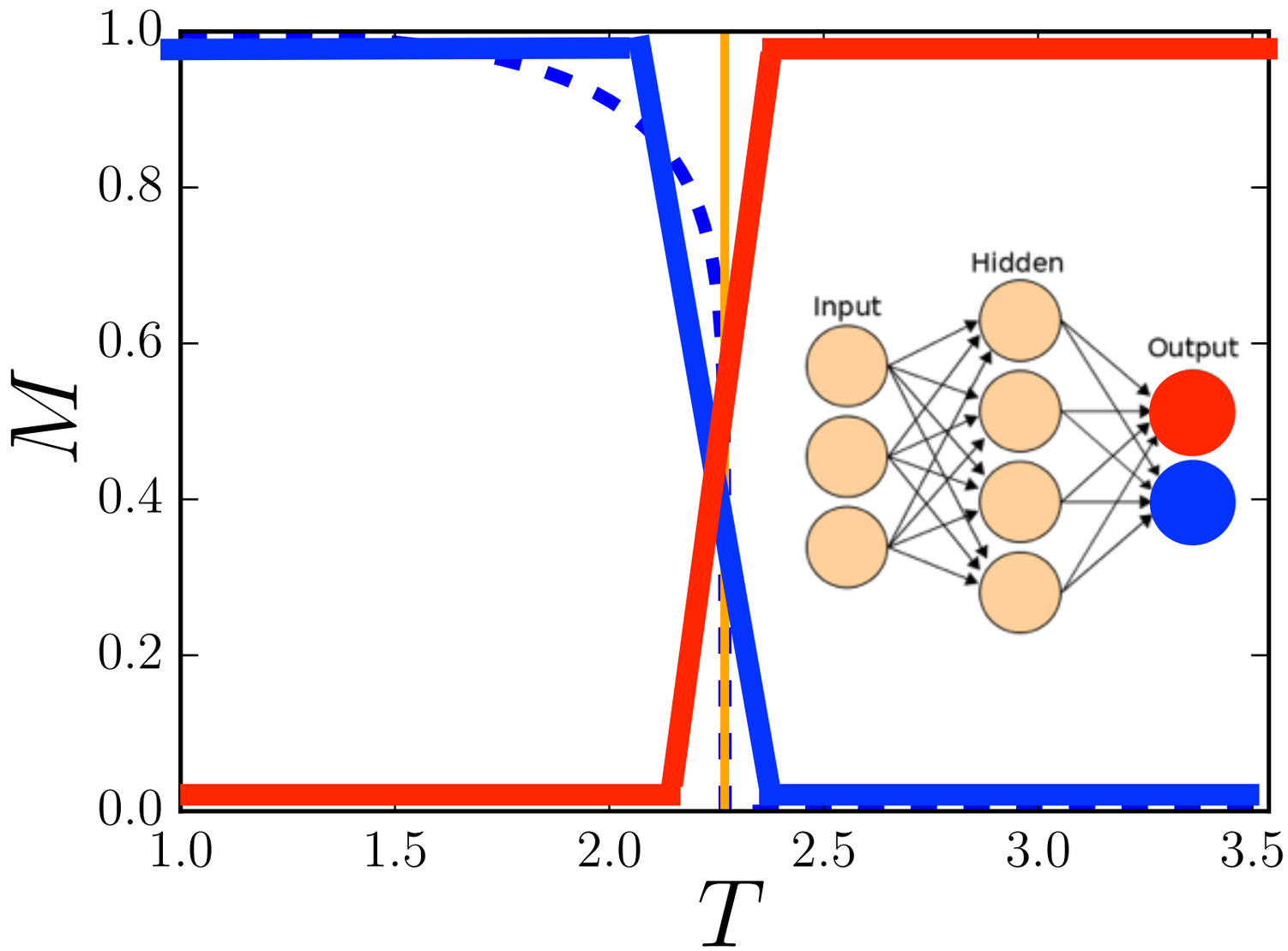


PM [1,0]

1605.01735

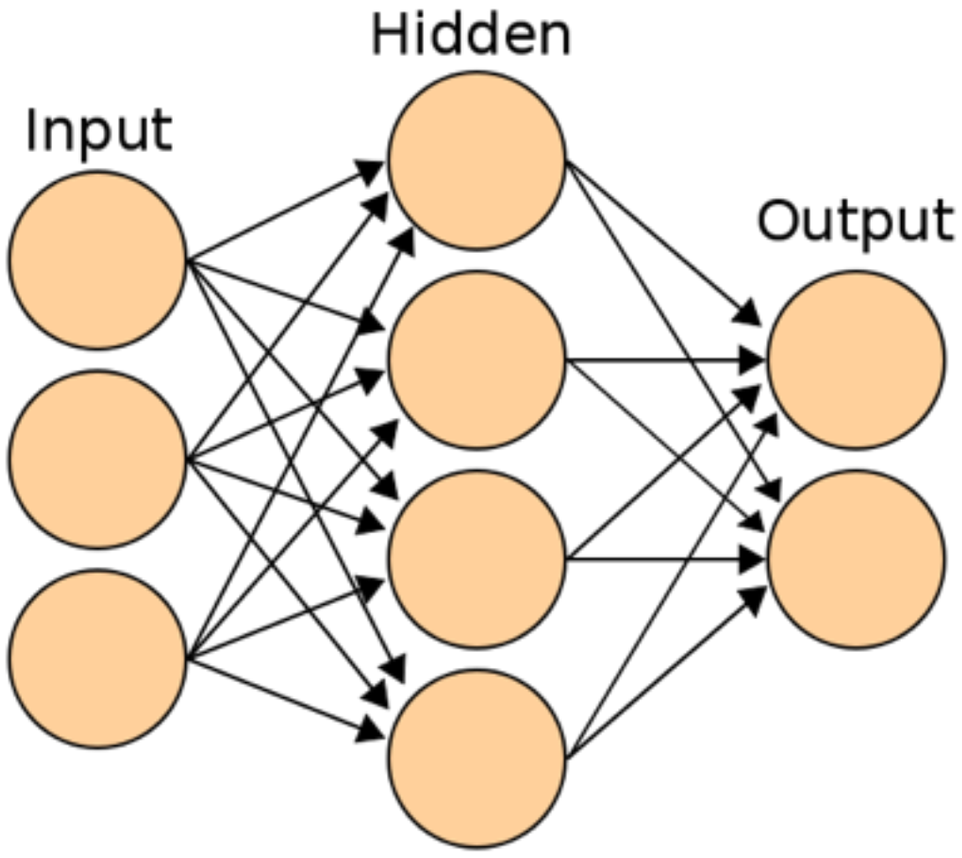
PHASES, PHASE TRANSITIONS, AND THE ORDER PARAMETER

We want 2 functions cold and hot such that



Artificial neural networks

Artificial neural networks are a family of models used to approximate functions that can depend on a large number of inputs. Artificial neural networks are generally presented as systems of interconnected "neurons" which exchange messages between each other

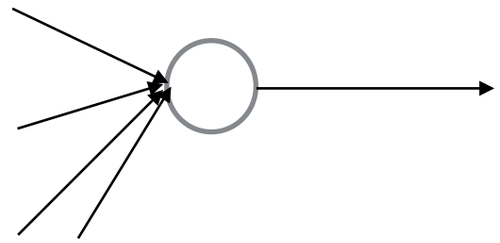


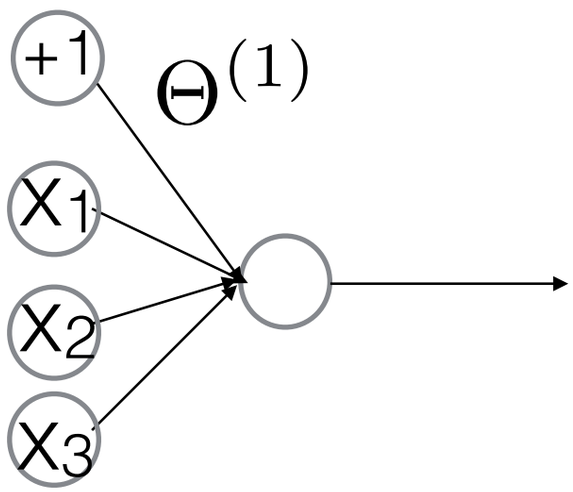
Connections= sets of adaptive weights, i.e. numerical parameters that are tuned by a learning algorithm

$$f : \mathbf{R}^n \to \mathbf{R}^m$$

Wikipedia

A neuron:





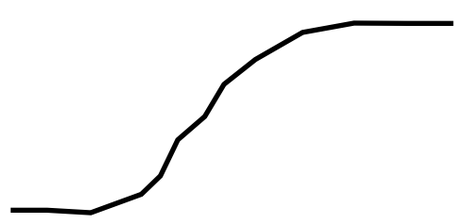
Sigmoid neuron

$$\beta = 1$$

$$h_\Theta(x) = \frac{1}{1 + e^{-\beta \Theta^T x}}$$

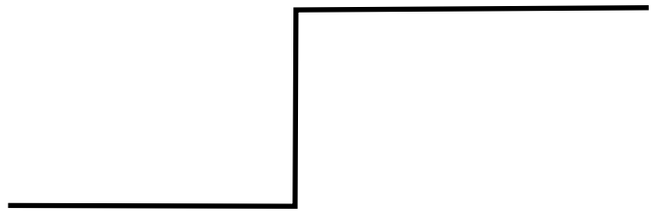
$$\Theta = (\theta_0\, \theta_1\, \theta_2\, \theta_3)$$

$$x = (1\ x_1\, x_2\, x_3)$$



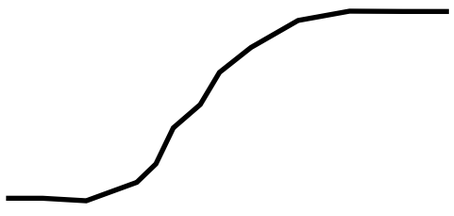
Perceptron:

$$\beta \to \infty$$

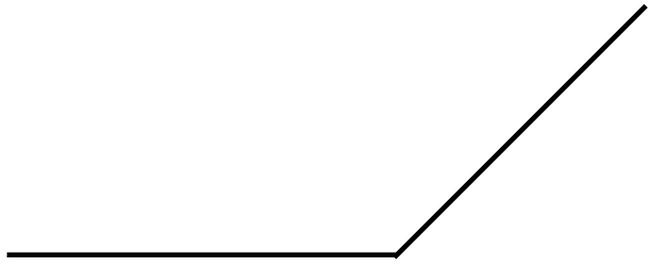


Where Θ's are the parameters you fiddle with

more neurons



$$h_\Theta(x) = \tanh(\Theta^T x)$$



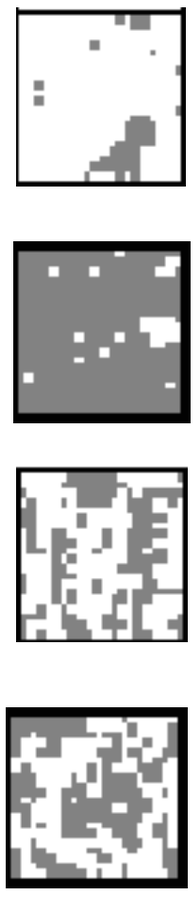
$$h_\Theta(x) = max(0, \Theta^T x)$$

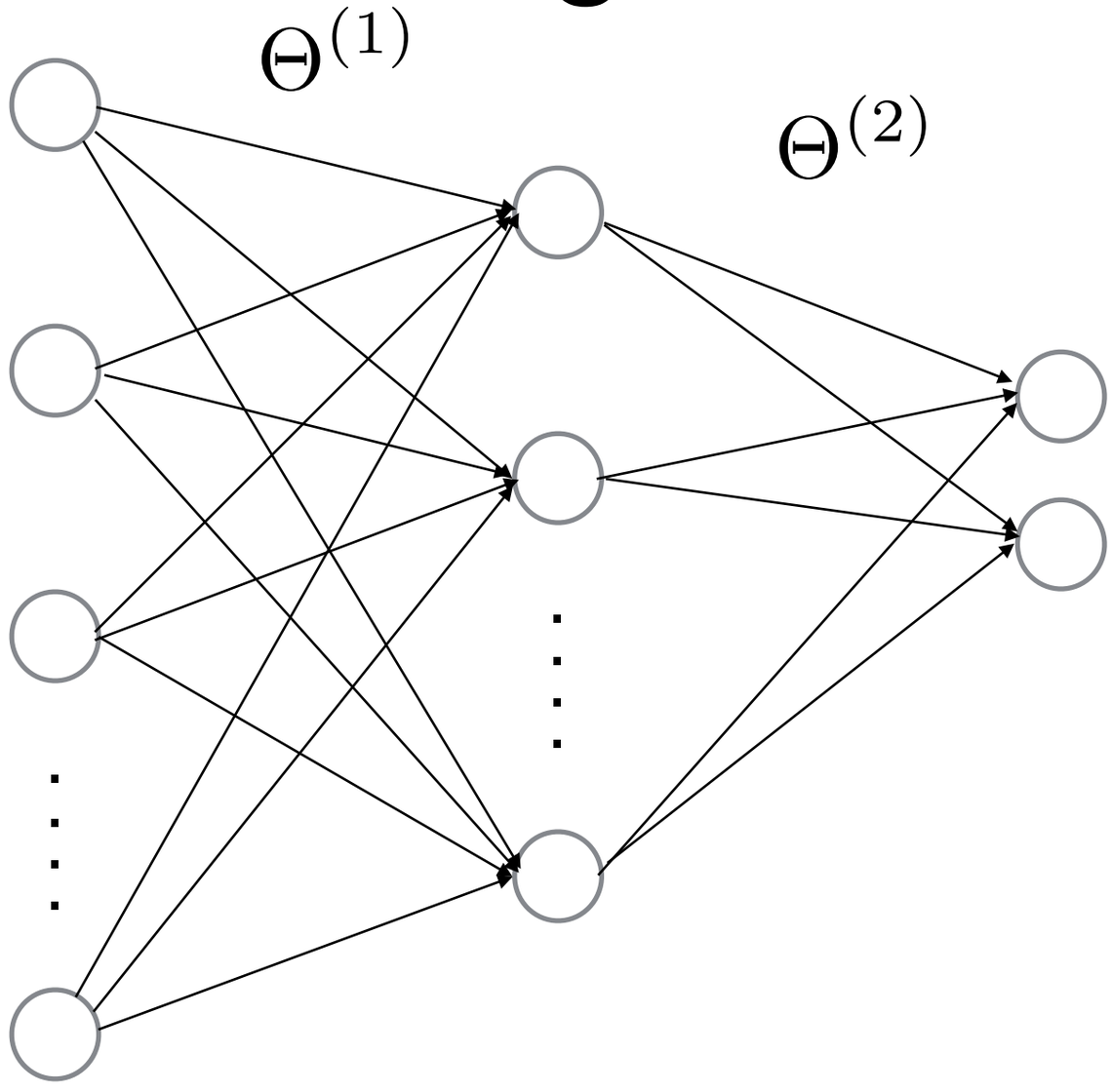
Rectified linear unit or ReLu

A neural net is a composition of these simpler nonlinear functions

One architecture for phase recognition

Input





output

FM [0,1]

PM [1,0]

0

1

Input layer hidden layer output layer

The theta matrices parametrize our function

How do we specify our function to do what we want? ($\Theta^{(1)}\Theta^{(2)}$)

Train the function with a big bunch of known (m) images called training set:

$$X = \begin{bmatrix} — (x^{(1)})^T — \\ — (x^{(2)})^T — \\ \vdots \\ — (x^{(m)})^T — \end{bmatrix} \qquad y = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

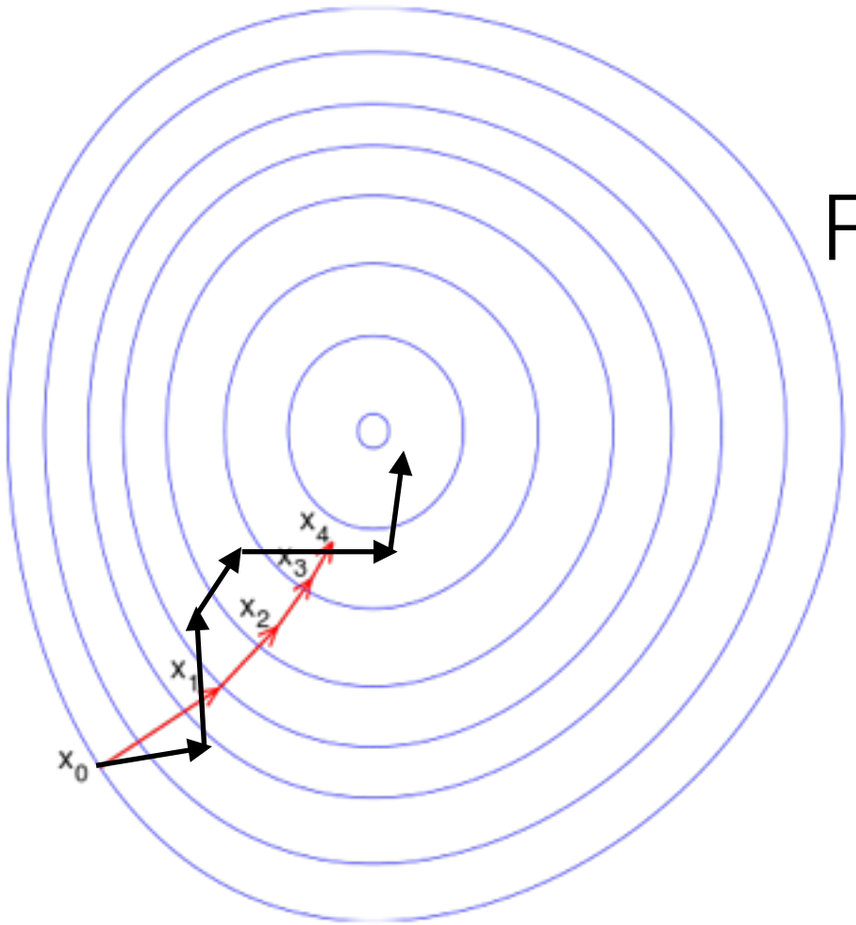
Define a cost function to be minimized (recall least squares cost function) Cross entropy

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \sum_{k=1}^{K} \left[-y_k^{(i)} \log((h_\theta(x^{(i)}))_k) - (1 - y_k^{(i)}) \log(1 - (h_\theta(x^{(i)}))_k) \right],$$

Learning algorithm

Learning: it sometimes just means fitting data to a function (the neural net) used to make predictions for unknown data.

Stochastic gradient descent

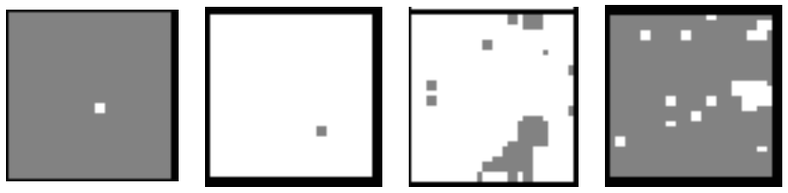


Relies on the analytical expression for the derivatives of the cost function with respect to the weights. For the neural nets this still OK:

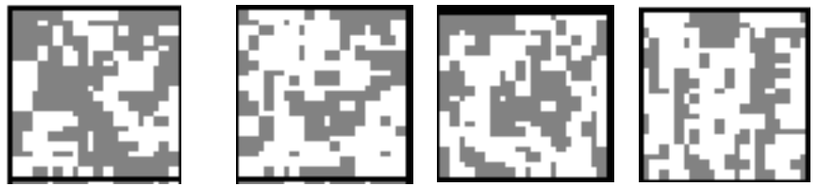
NN=composition of functions, so that derivatives can be computed using the chain rule.

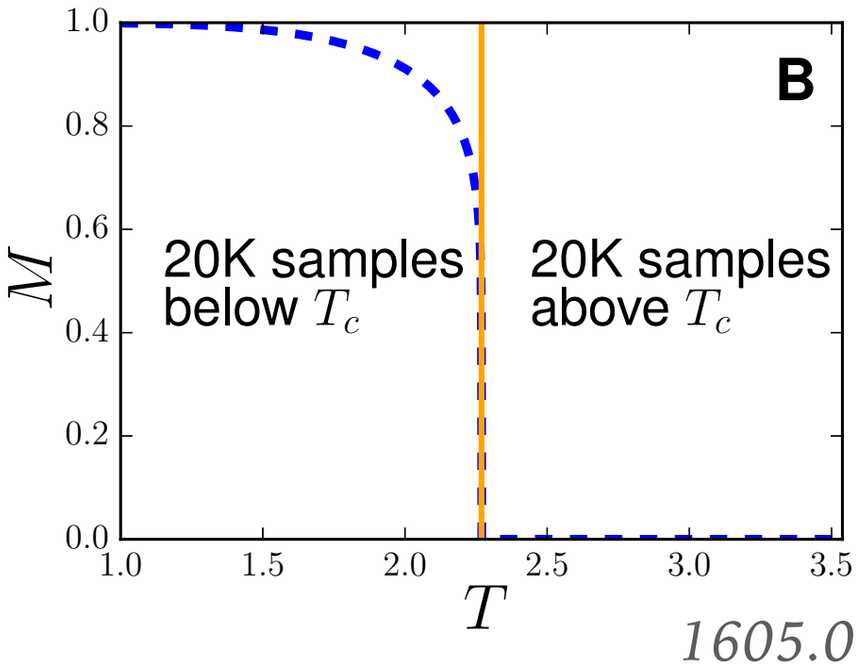
COLLECTING THE TRAINING/TESTING DATA: MC SAMPLING ISING MODEL AND LABELS

2D Ising model in the ordered phase



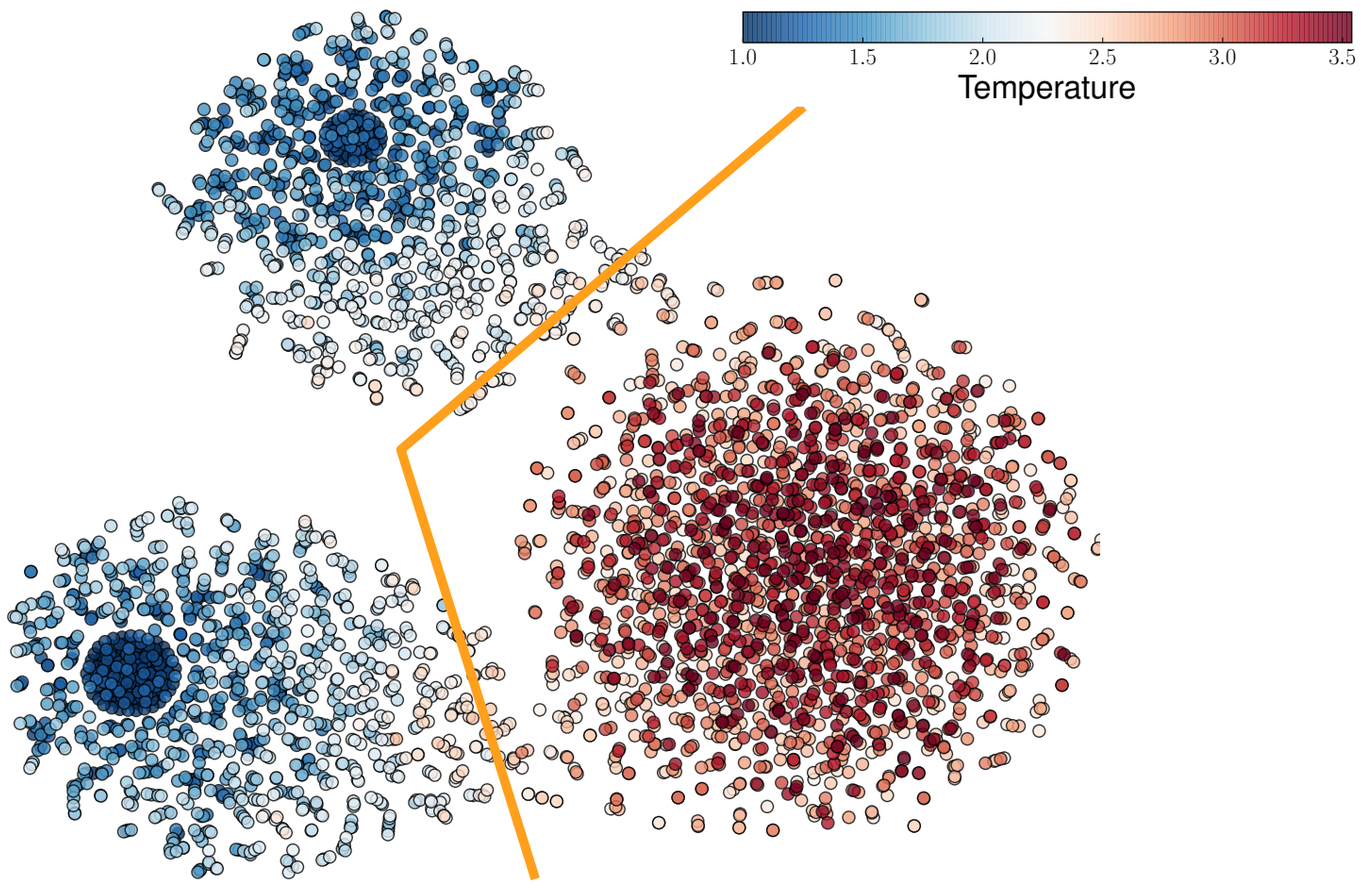
2D Ising model in the disordered phase





Training/testing data is drawn from the Boltzmann distribution

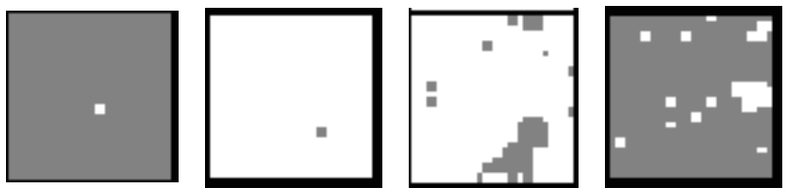
$$p(\sigma_1, \sigma_2, ..., \sigma_N) = \frac{e^{-\beta E(\sigma_1, \sigma_2, ..., \sigma_N)}}{Z(\beta)}$$



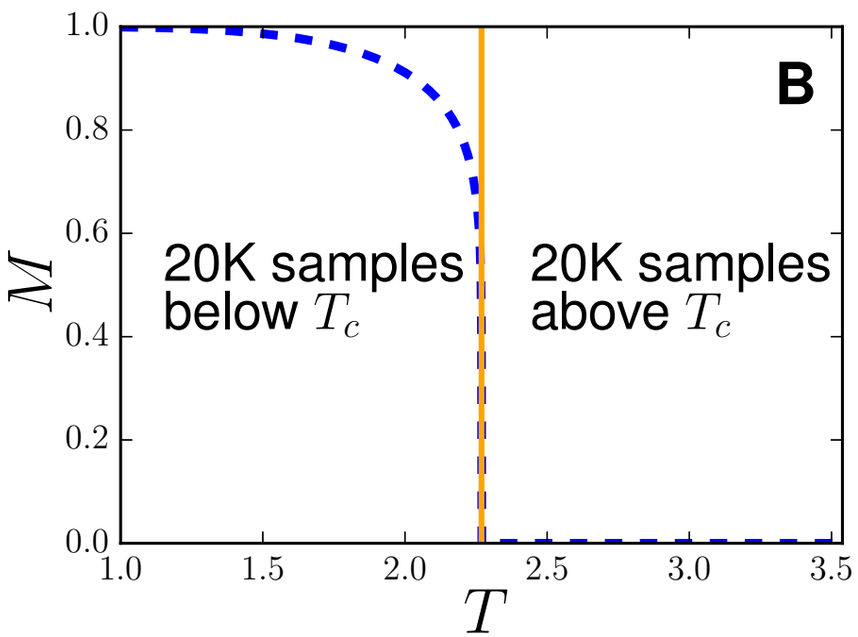
1605.01735

COLLECTING THE TRAINING/TESTING DATA: MC SAMPLING ISING MODEL AND LABELS

2D Ising model in the ordered phase

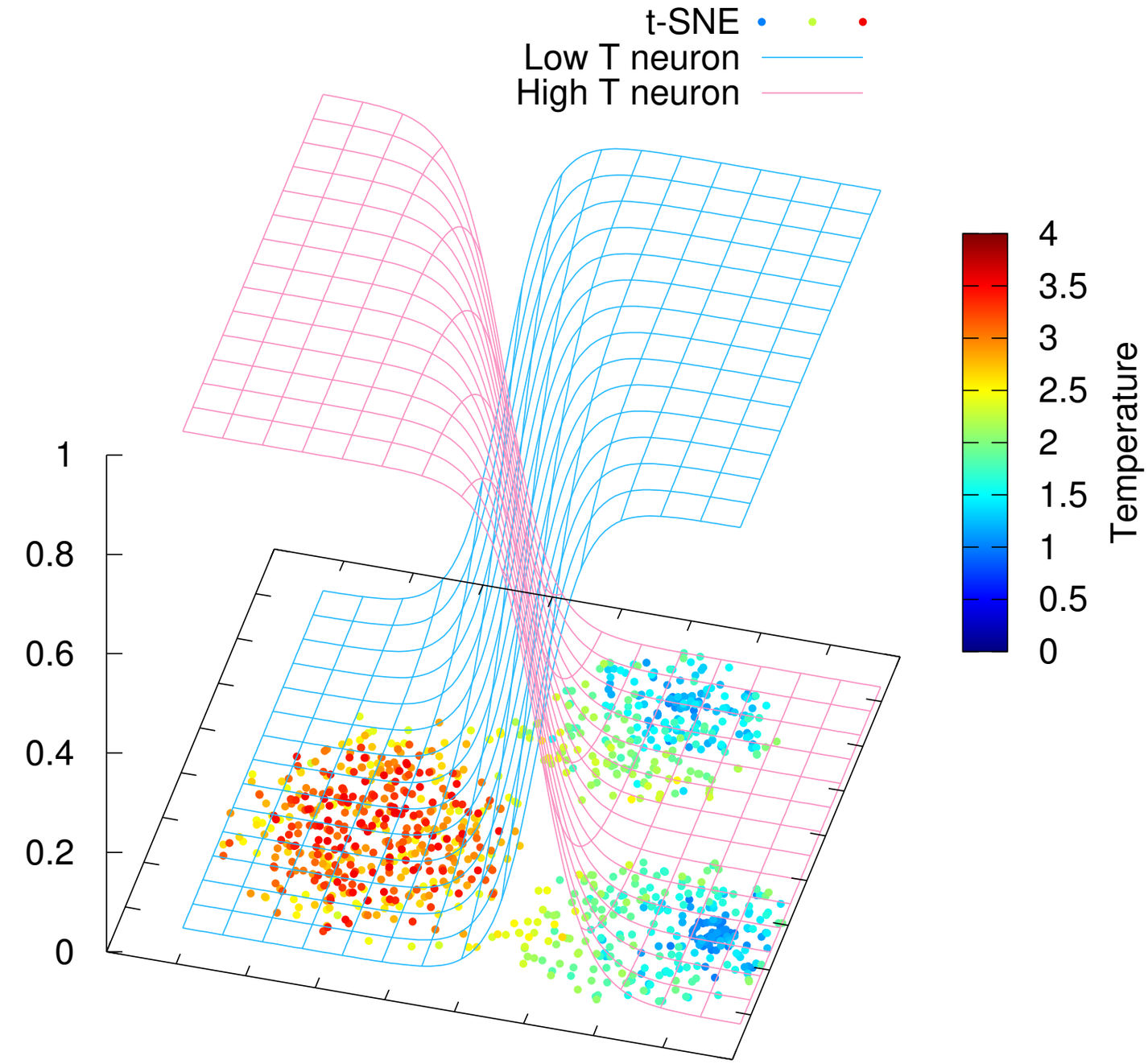


2D Ising model in the disordered phase



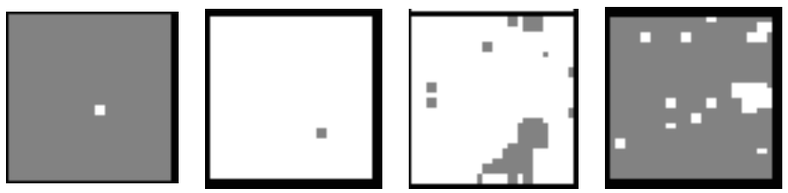
Successful training amounts to finding functions

$F_{\text{High } T}(\sigma_1, ..., \sigma_N)$ $F_{\text{Low } T}(\sigma_1, ..., \sigma_N)$

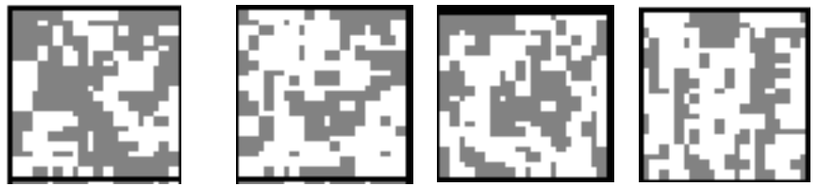


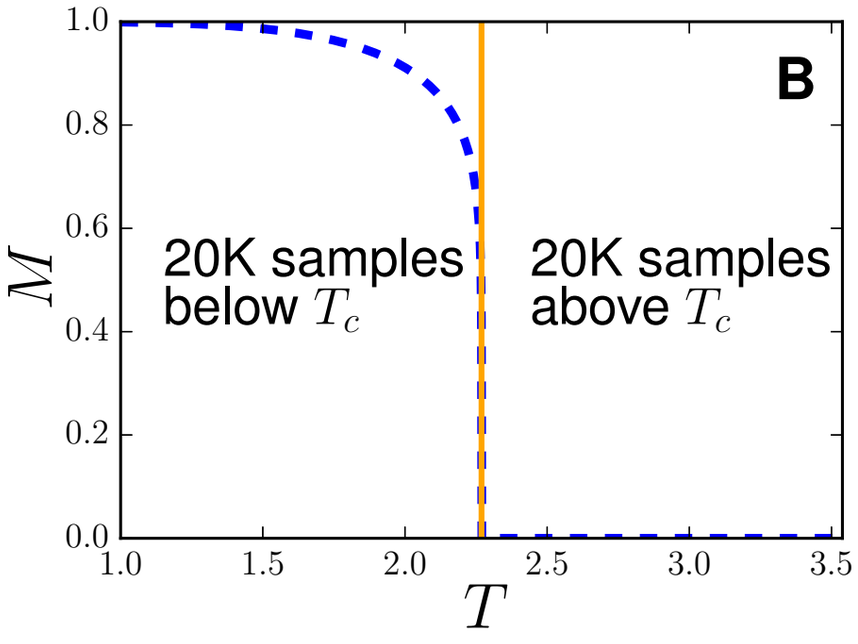
RESULTS: SQUARE LATTICE ISING MODEL (TEST SETS)

2D Ising model in the ordered phase

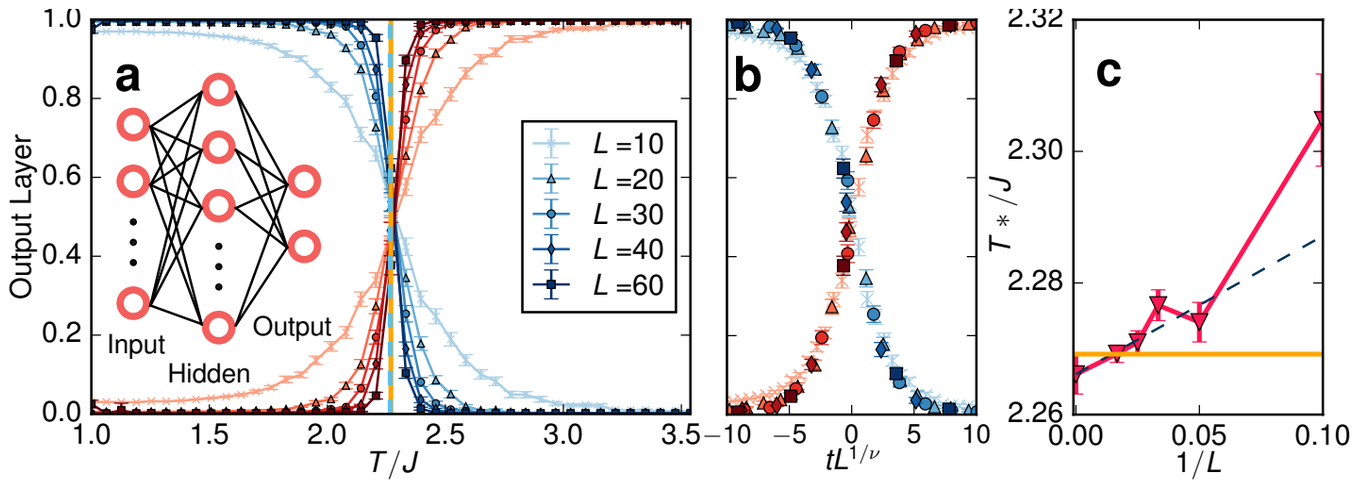


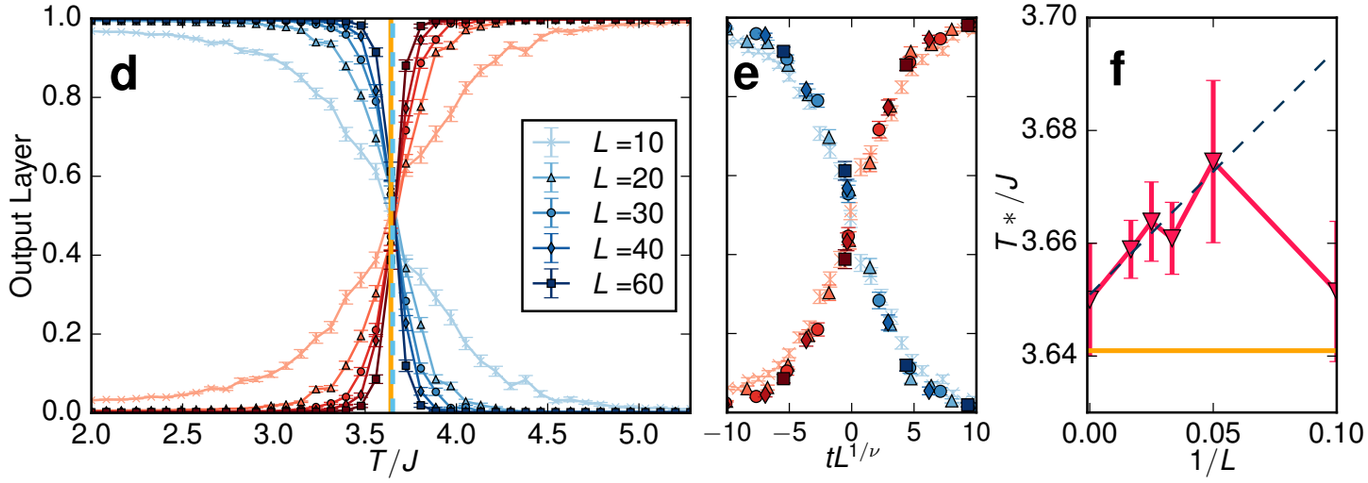
2D Ising model in the disordered phase





$T_c/J = 2/\ln\left(1+\sqrt{2}\right)$





Critical

1605.01735, Nature Physics

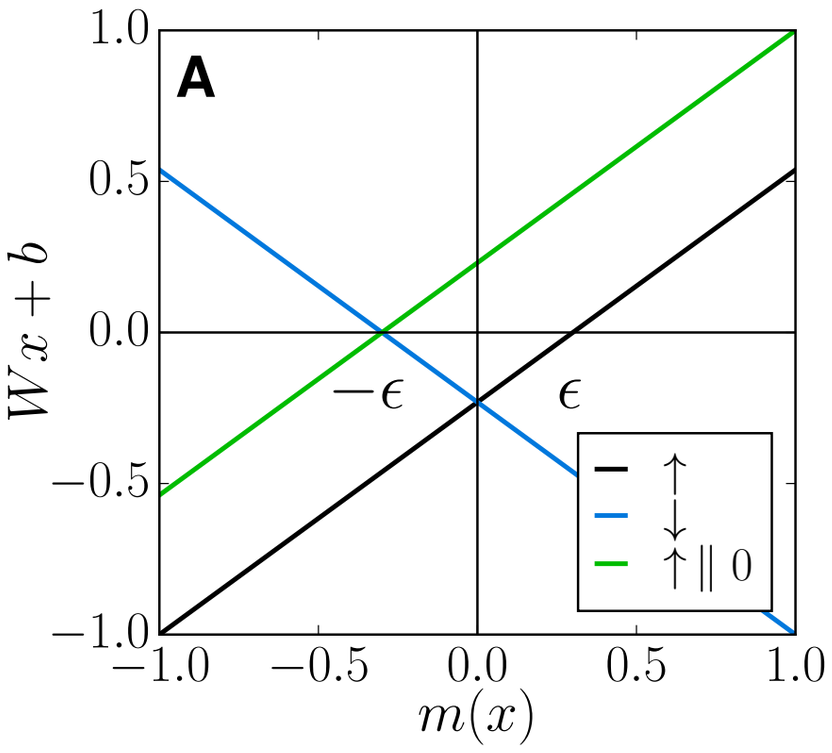
Tc of Triangular within <1%

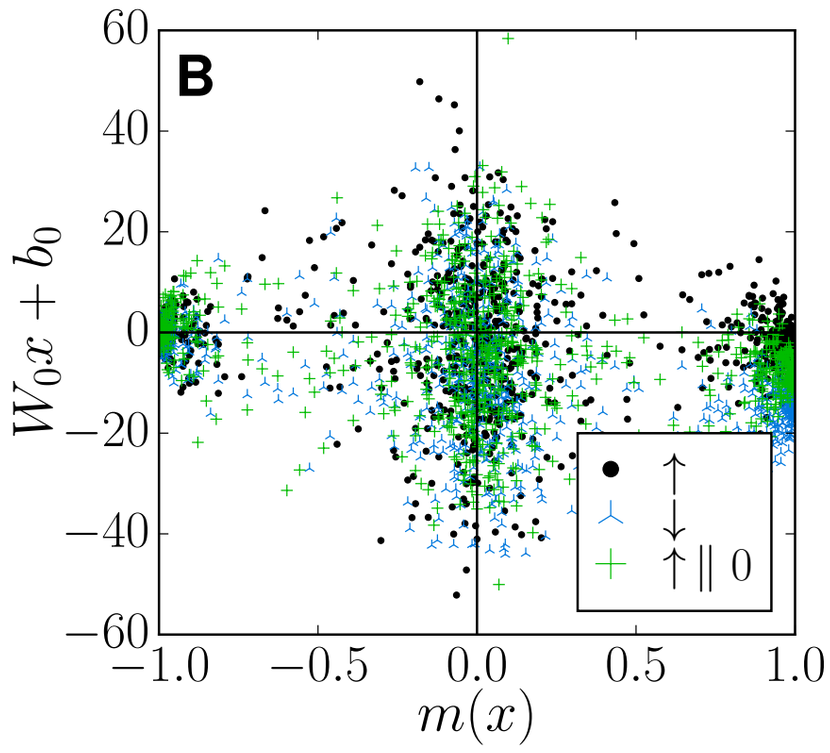
NN knows about criticality

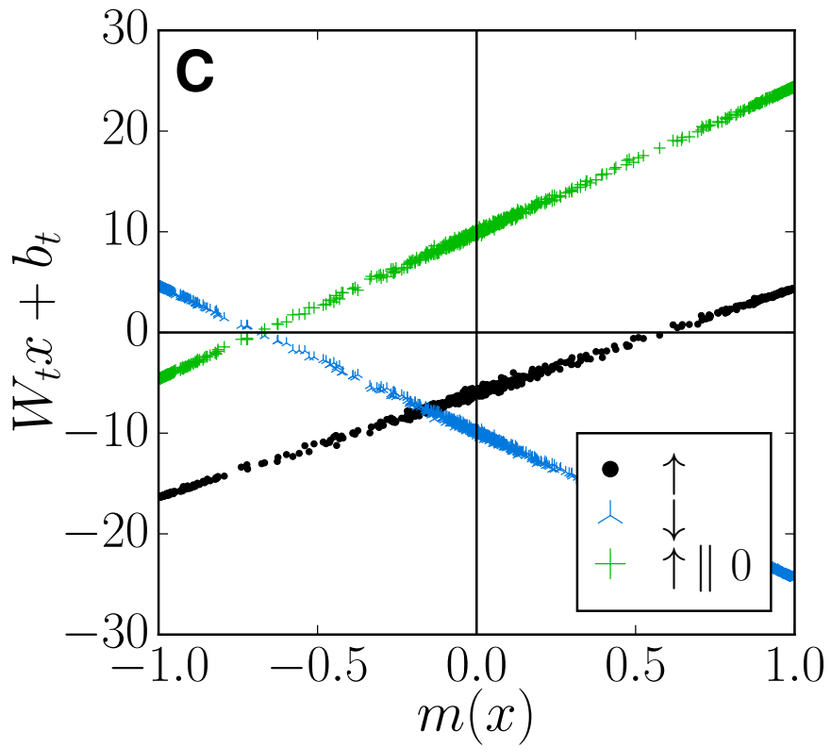
$\nu \approx 1$

ANALYTICAL UNDERSTANDING

Investigating the argument of the hidden layer during the training





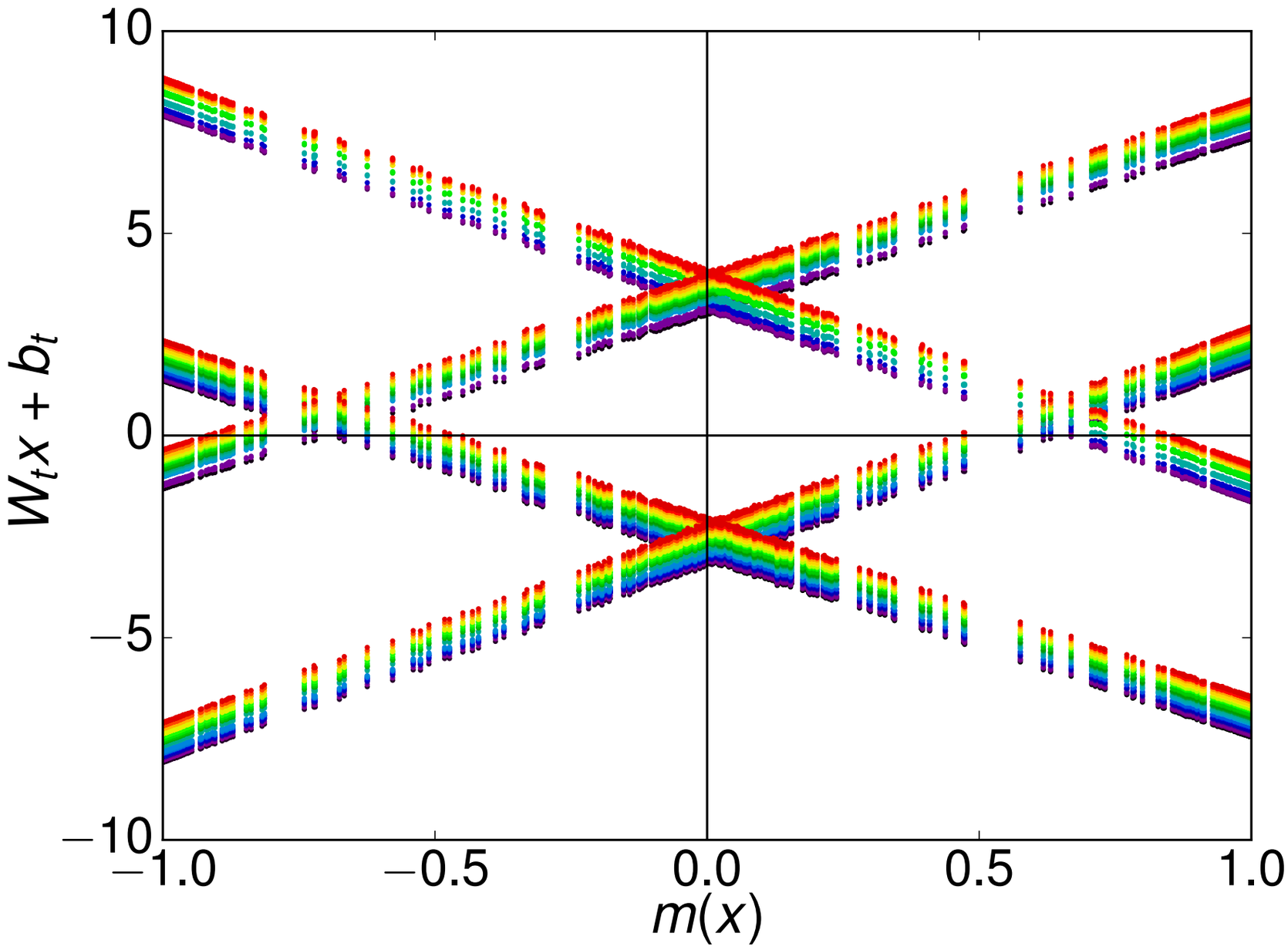


$$W = \frac{1}{N\,(1+\epsilon)} \begin{pmatrix} 1 & 1 & \cdots & 1 \\ -1 & -1 & \cdots & -1 \\ 1 & 1 & \cdots & 1 \end{pmatrix}, \text{ and } b = \frac{\epsilon}{(1+\epsilon)} \begin{pmatrix} -1 \\ -1 \\ 1 \end{pmatrix}, \qquad Wx + b = \frac{1}{(1+\epsilon)} \begin{pmatrix} m(x) - \epsilon \\ -m(x) - \epsilon \\ m(x) + \epsilon \end{pmatrix},$$

$$x = [\sigma_1\sigma_2, ..., \sigma_N]^{\mathrm{T}} \qquad m(x) = \tfrac{1}{N}\sum_{i=1}^{N} \sigma_i$$

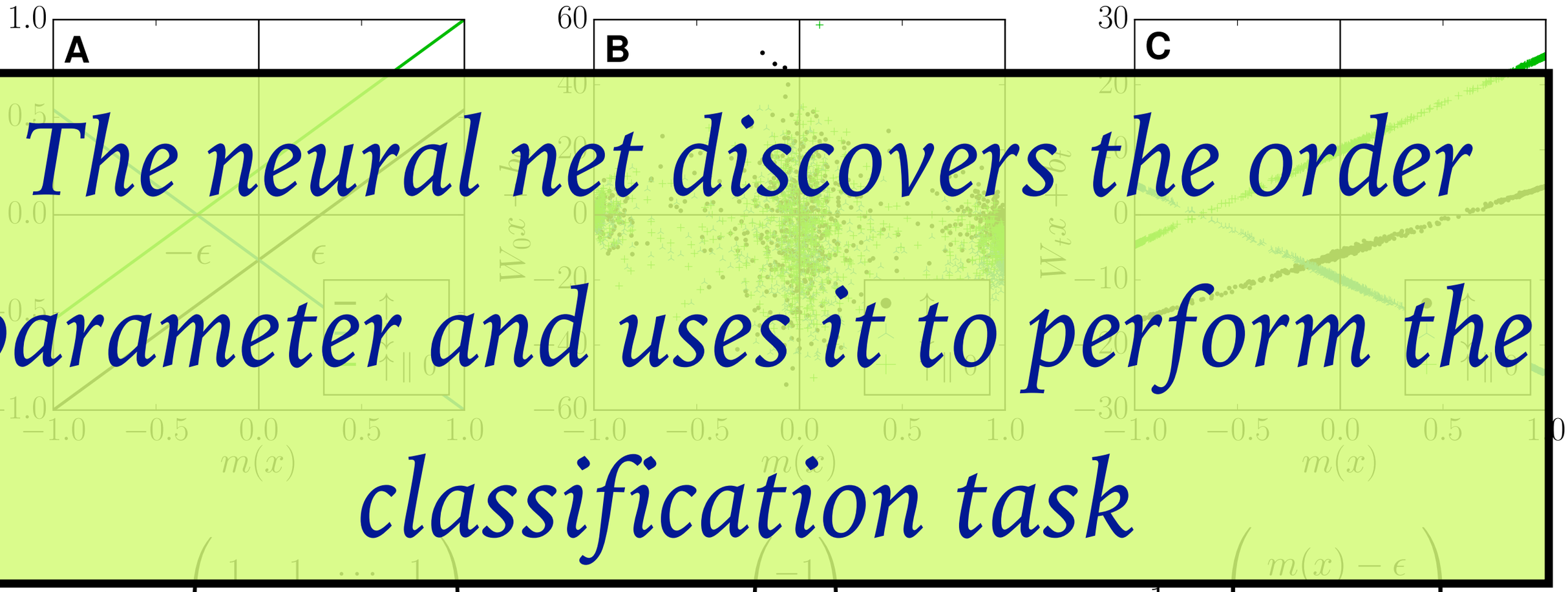
1605.01735, Nature Physics

FOR A REAL NN THIS REMAINS TRUE



ANALYTICAL UNDERSTANDING

Investigating the argument of the hidden layer during the training



The neural net discovers the order parameter and uses it to perform the classification task

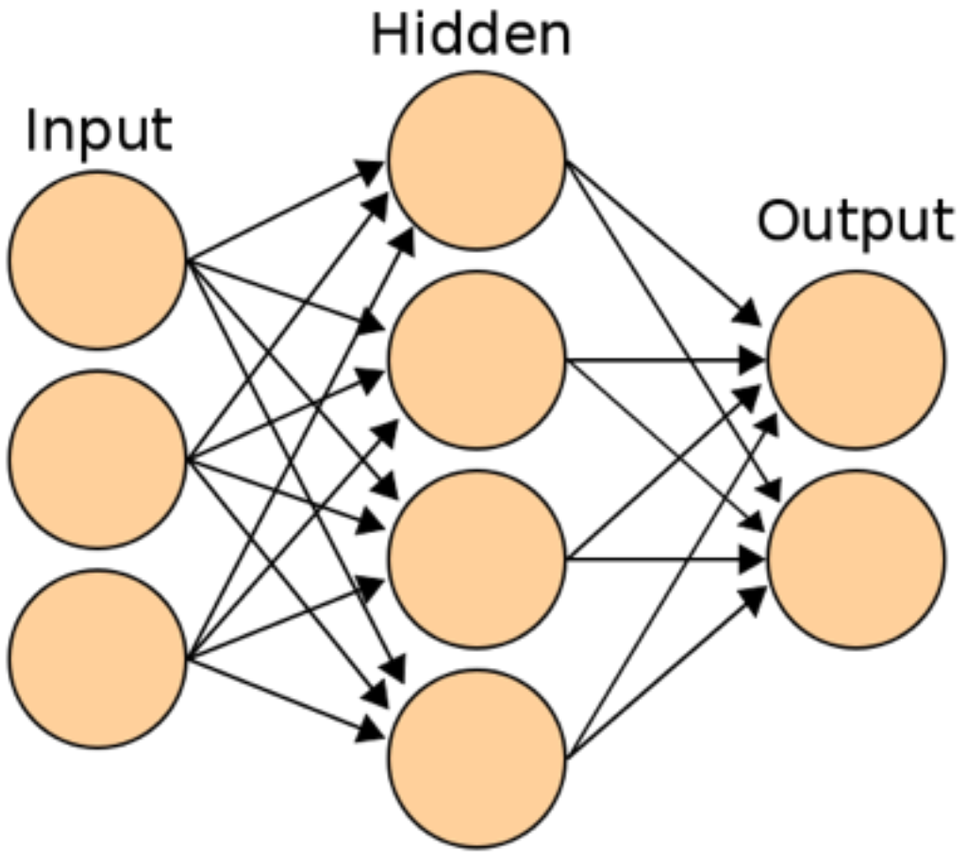
$$W = \frac{1}{N(1+\epsilon)}\begin{pmatrix} -1 & -1 & \cdots & -1 \\ 1 & 1 & \cdots & 1 \end{pmatrix}, \text{ and } b = \frac{\epsilon}{(1+\epsilon)}\begin{pmatrix} -1 \\ 1 \end{pmatrix}, \qquad Wx+b = \frac{1}{(1+\epsilon)}\begin{pmatrix} m(x)-\epsilon \\ -m(x)-\epsilon \\ m(x)+\epsilon \end{pmatrix},$$

$$x = [\sigma_1\sigma_2, ..., \sigma_N]^{\mathrm{T}} \qquad m(x) = \tfrac{1}{N}\sum_{i=1}^{N}\sigma_i$$

TUTORIAL

Artificial neural networks

Artificial neural networks are a family of models used to approximate functions that can depend on a large number of inputs. Artificial neural networks are generally presented as systems of interconnected "neurons" which exchange messages between each other

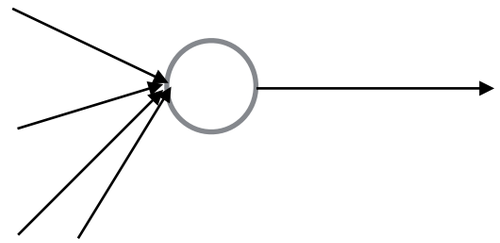


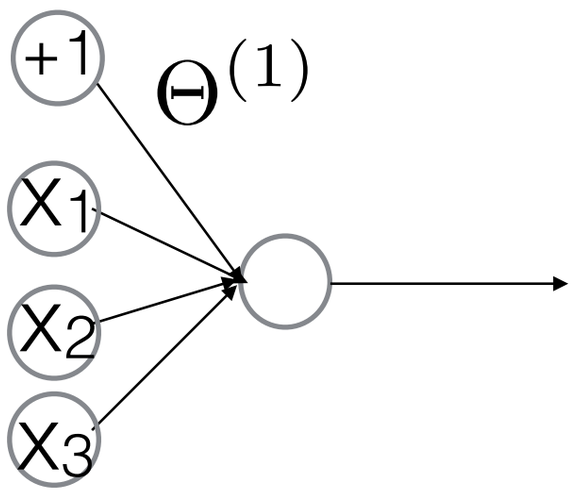
Connections= sets of adaptive weights, i.e. numerical parameters that are tuned by a learning algorithm

$$f : \mathbf{R}^n \to \mathbf{R}^m$$

Wikipedia

A neuron:





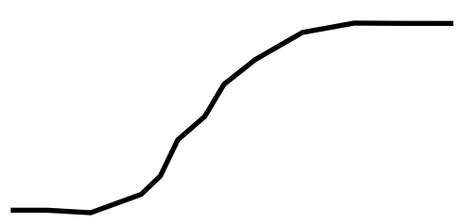
Sigmoid neuron

$$\beta = 1$$

$$h_\Theta(x) = \frac{1}{1 + e^{-\beta \Theta^T x}}$$

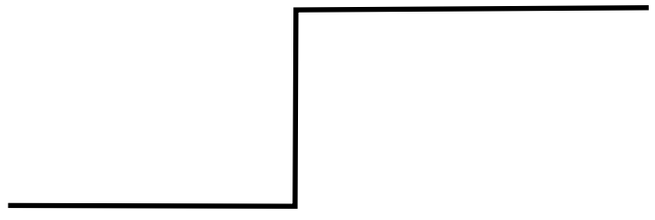
$$\Theta = (\theta_0\, \theta_1\, \theta_2\, \theta_3)$$

$$x = (1\ x_1\, x_2\, x_3)$$



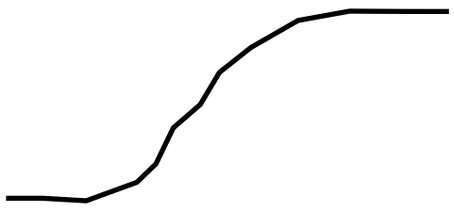
Perceptron:

$$\beta \to \infty$$

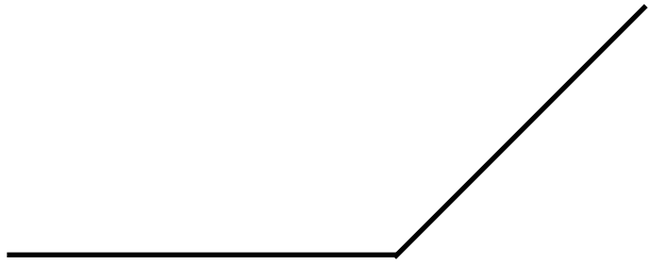


Where Θ's are the parameters you fiddle with

more neurons



$$h_\Theta(x) = \tanh(\Theta^T x)$$



$$h_\Theta(x) = max(0, \Theta^T x)$$

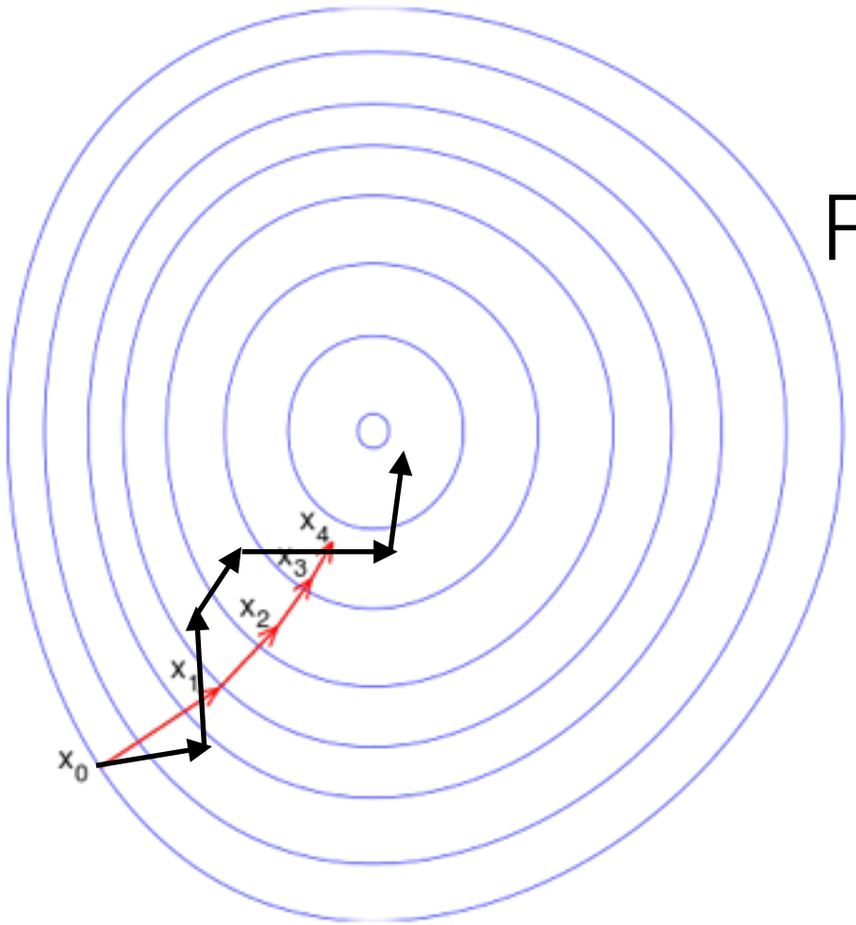
Rectified linear unit or ReLu

A neural net is a composition of these simpler nonlinear functions

Learning algorithm

Learning: it sometimes just means fitting data to a function (the neural net) used to make predictions for unknown data.

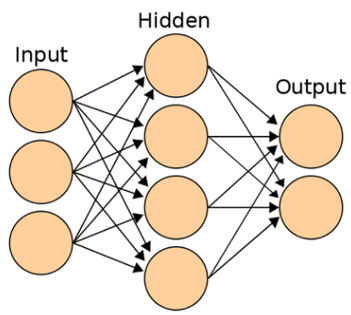
Stochastic gradient descent



Relies on the analytical expression for the derivatives of the cost function with respect to the weights. For the neural nets this still OK:

NN=composition of functions, so that derivatives can be computed using the chain rule.

Example: Handwritten digits

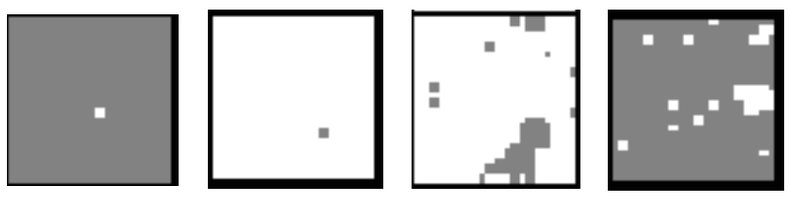


5

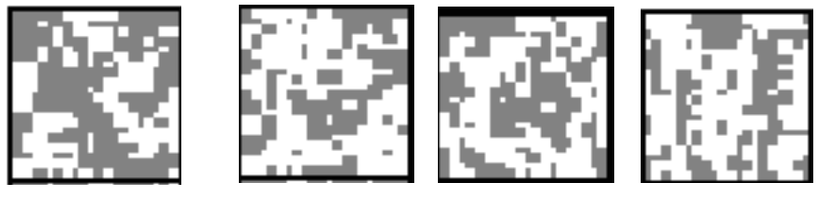
ML community has developed powerful supervised learning algorithms

 $= 5 +$ *Fluctuations*

FM phase

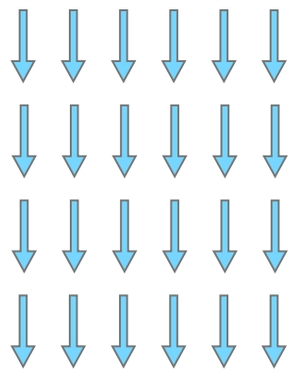


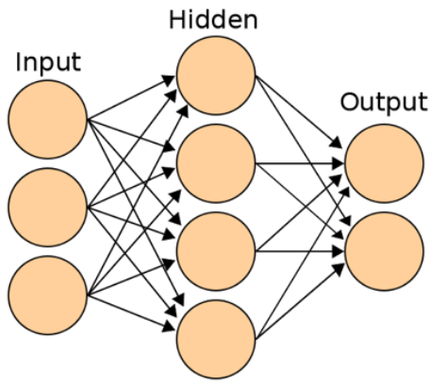
High T phase



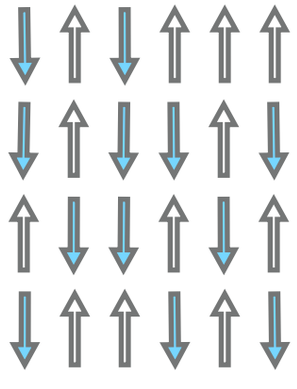
gray = spin up

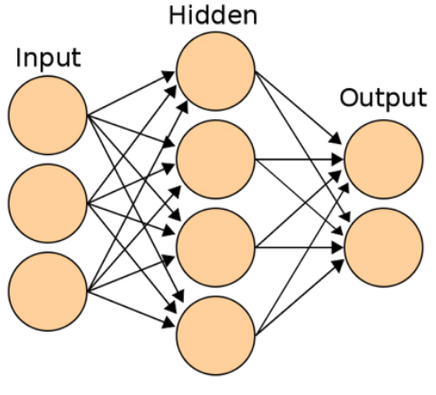
white = spin down





FM (0)

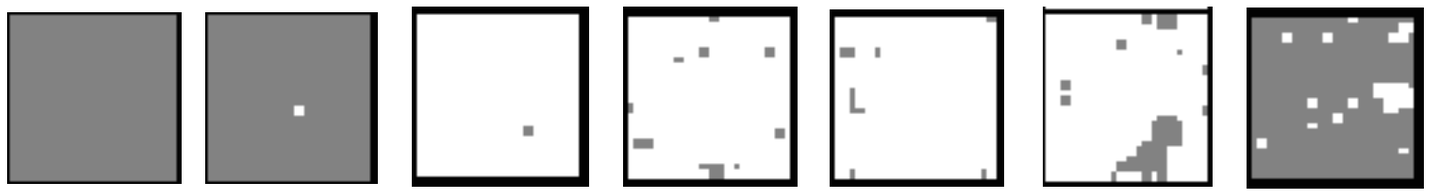




PM (1)

Input data

2D Ising model in the ordered phase

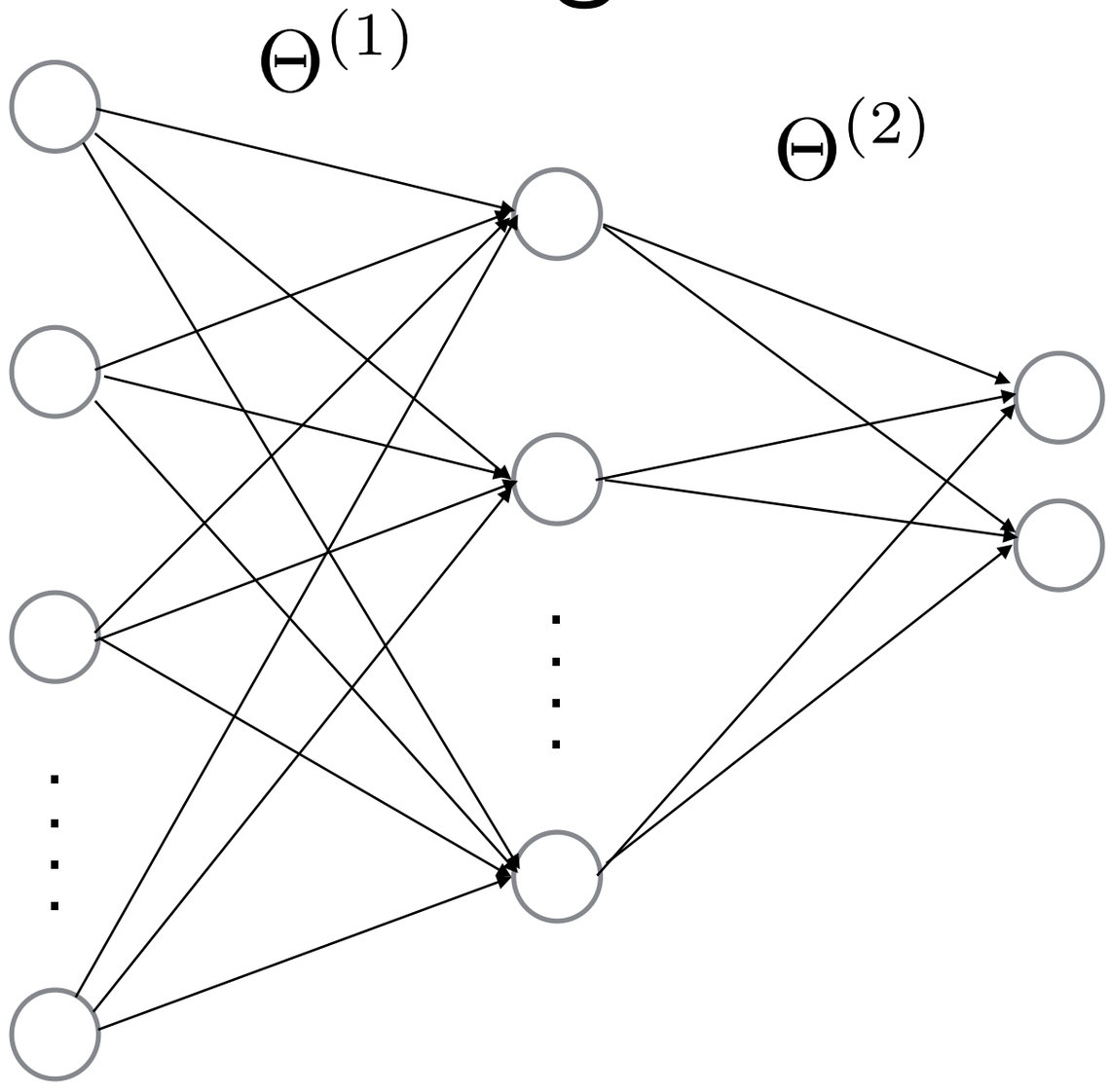


2D Ising model in the disordered phase

Critical

One architecture for phase recognition

Input



output

0

1

FM

FM

HT

HT

Input layer hidden layer output layer

The theta matrices parametrize our function

How do we specify our function to do what we want? ($\Theta^{(1)}\Theta^{(2)}$)

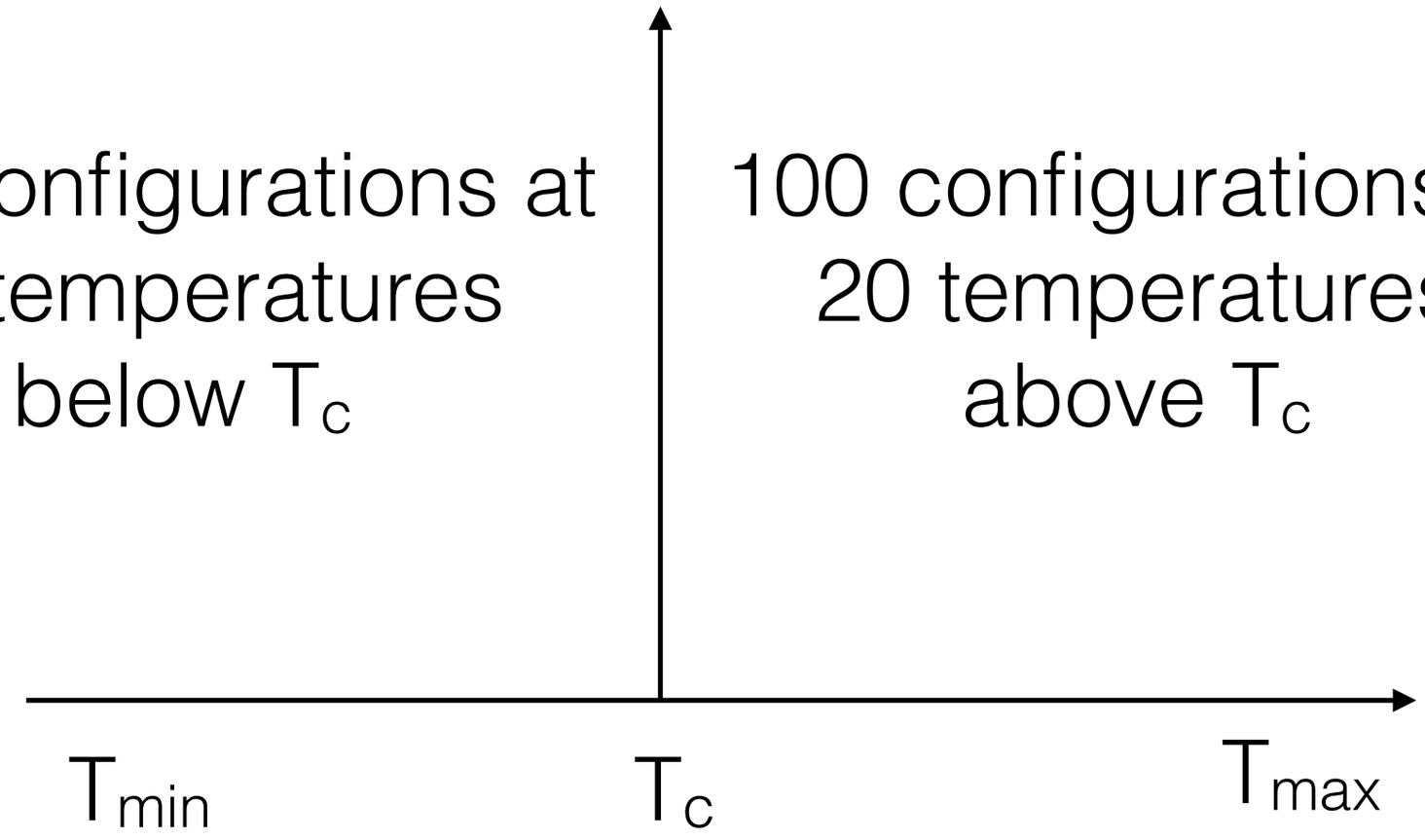
Train the function with a big bunch of known (m) images called training set:

$$X = \begin{bmatrix} — (x^{(1)})^T — \\ — (x^{(2)})^T — \\ \vdots \\ — (x^{(m)})^T — \end{bmatrix} \qquad y = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Define a cost function to be minimized (recall least squares cost function)

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \sum_{k=1}^{K} \left[-y_k^{(i)} \log((h_\theta(x^{(i)}))_k) - (1 - y_k^{(i)}) \log(1 - (h_\theta(x^{(i)}))_k) \right],$$

2d Ising training/test sets: uncorrelated classical Monte Carlo configurations



T_{min}=1.0 T_c=2.2691 T_{max}=3.538

Overall accuracy: 98%!