Sampling states in statistical physics with neural networks

Tomasz Stebel

Institute of Theoretical Physics,
Jagiellonian University, Kraków

with Piotr Białas and Piotr Korcyl

Based on:
Phys.Rev.E 107 (2023) 1, 015303; Phys.Rev.E 107 (2023) 5, 054127
Comput.Phys.Commun. 281 (2022) 108502;
Phys.Rev.E 108 (2023) 4, 4
e-Print: 2308.13294

NATIONAL SCIENCE CENTRE
POLAND

AIRA, 21.12.2023

Project is supported by
National Science Centre, grant
no. 2019/32/C/ST2/00202.

Plan

- Ising Model
- Variational Autoregressive NN (VAN) and Hierarchical Autoregressive NN (HAN)
- Normalizing Flows and field theory

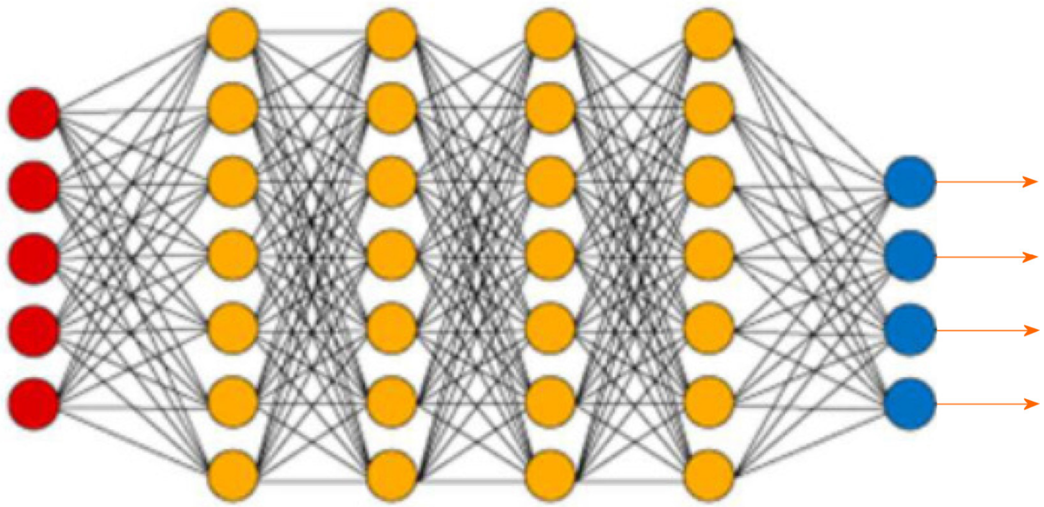
Neural networks

Neural network:

$$f_\theta : R^n \to R^m$$

θ– weights (=parameters of function) – $O(100) - O(10^9)$

$n = 5$



$m = 4$

f_θ is composition of layers:

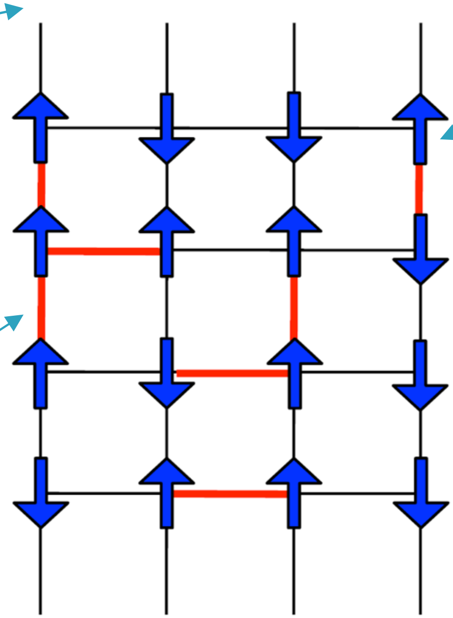
$$f_\theta = f_k \circ f_{k-1} \circ \cdots \circ f_1$$

Layer is e.g. $\vec{f_i} = \sigma(\boldsymbol{W_\theta} \vec{x}_i + \vec{b}_{\boldsymbol{\theta}})$

1. Sampling models with <u>discrete</u> degrees of freedom (d.o.f.)

Ising Model in 2d

Periodic boundary conditions in x and y direction



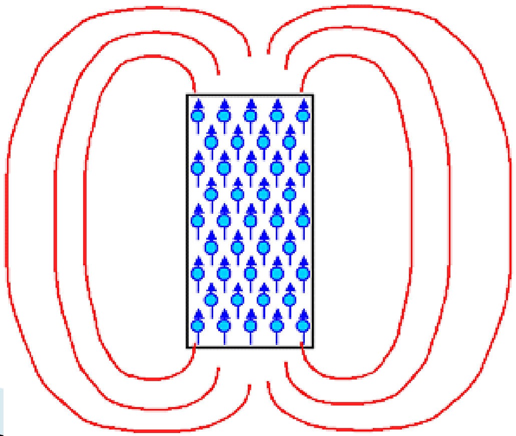
„spin": $s = \pm 1$

Energy:

$$E = -\sum_{<i,j>} s_i \, s_j$$

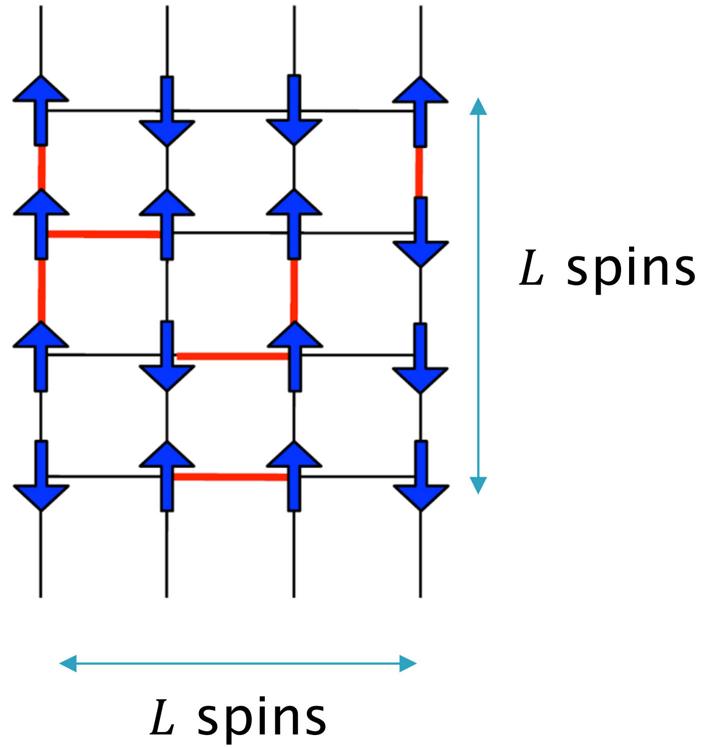
Nearest Neighbors interactions

This is the simplest model of magnet:



… and probably the best-studied model in statistical physics.

Ising Model in 2d



$$E = -\sum_{<i,j>} s_i \, s_j$$

Probability of given configuration s:

$$p(\boldsymbol{s}) = \frac{1}{Z} e^{-\beta E(\boldsymbol{s})}$$

$$\beta = \frac{1}{temperature}$$

where partition function $\quad Z = \sum_{\boldsymbol{s}} e^{-\beta E(\boldsymbol{s})}$

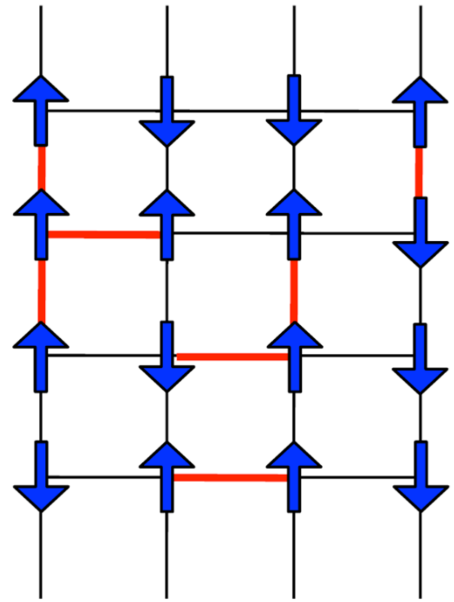
2^N configurations,
$N = L^2 = \#$ spins

Order parameter:

$$|m| = \frac{1}{N} \left| \sum_i s_i \right|$$

absolute magnetization per spin

Ising Model in 2d



Properties:

1) For $\beta = 0$ (infinite temperature):
disordered phase, $|m|$=0

2) For $\beta = \infty$ (zero temperature):
ordered phase, $|m|$=1

3) When $L \to \infty$ we have second-order phase transition at:

$$\beta_c = \frac{1}{2}\ln(1+\sqrt{2}) \approx 0.441$$

Sampling from Boltzman distribution p

2^N configurations

$$\langle O \rangle = \sum_{\boldsymbol{s}} O(\boldsymbol{s}) p(\boldsymbol{s}) \longrightarrow \langle O \rangle \approx \frac{1}{N_{samples}} \sum_{k=1}^{N_{samples}} O(\boldsymbol{s}_k)$$

For some observable O

where $\boldsymbol{s}_k$ sampled from p

Problem: how to sample from p?

$\downarrow$

Usually done by Markov Chain Monte Carlo (e.g. Metropolis algorithm), but:

- correlation between samples
- $Z = \sum_{\boldsymbol{s}} e^{-\beta E(\boldsymbol{s})}$ very hard to calculate (no access to the free energy $F = -\frac{1}{\beta}\log Z$ and entropy)

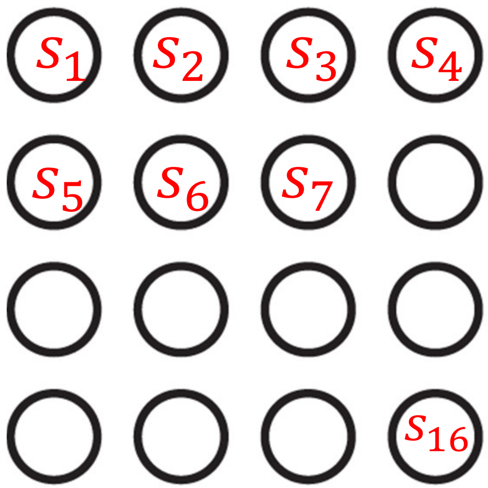
Neural networks can be trained to be a sampler and provides variational estimate of Z

D. Wu, L. Wang, and P. Zhang, Phys. Rev. Lett., vol. 122, p. 080602

Autoregressive networks

We denote probability of configuration $\boldsymbol{s}$ which network provides as:

$$q_\theta(\boldsymbol{s})$$



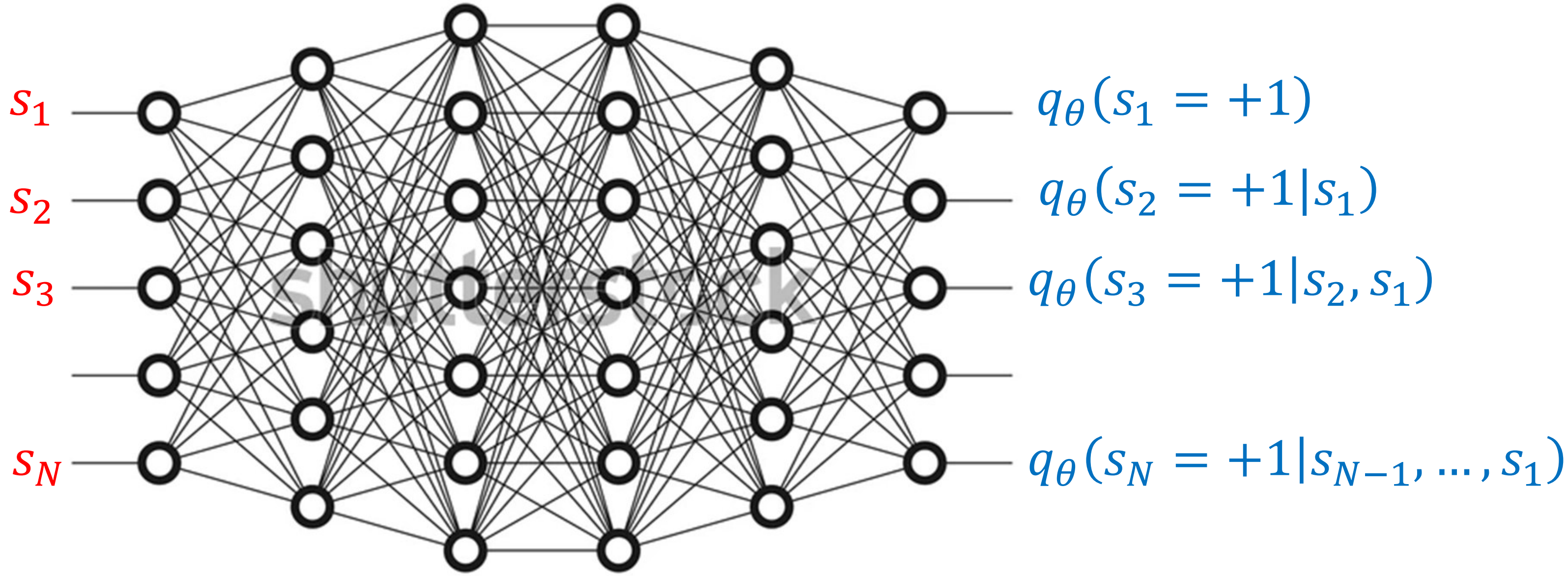
IDEA:

Network learns N (# of spins) conditional probabilities:

$$q_\theta(\boldsymbol{s}) = q_\theta(s_1)\, q_\theta(s_2|s_1)\, q_\theta(s_3|s_2, s_1) \ldots q_\theta(s_N|s_{N-1}, \ldots, s_1)$$

Autoregressive networks

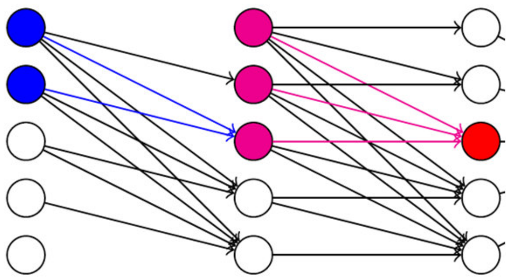
$$q_\theta(\boldsymbol{s}) = q_\theta(s_1)\, q_\theta(s_2|s_1)\, q_\theta(s_3|s_2, s_1) \ldots q_\theta(s_N|s_{N-1}, \ldots, s_1)$$



Input: spin configuration (value of each spin) $(\pm 1, \ldots \pm 1)$

Output: conditional probabilities

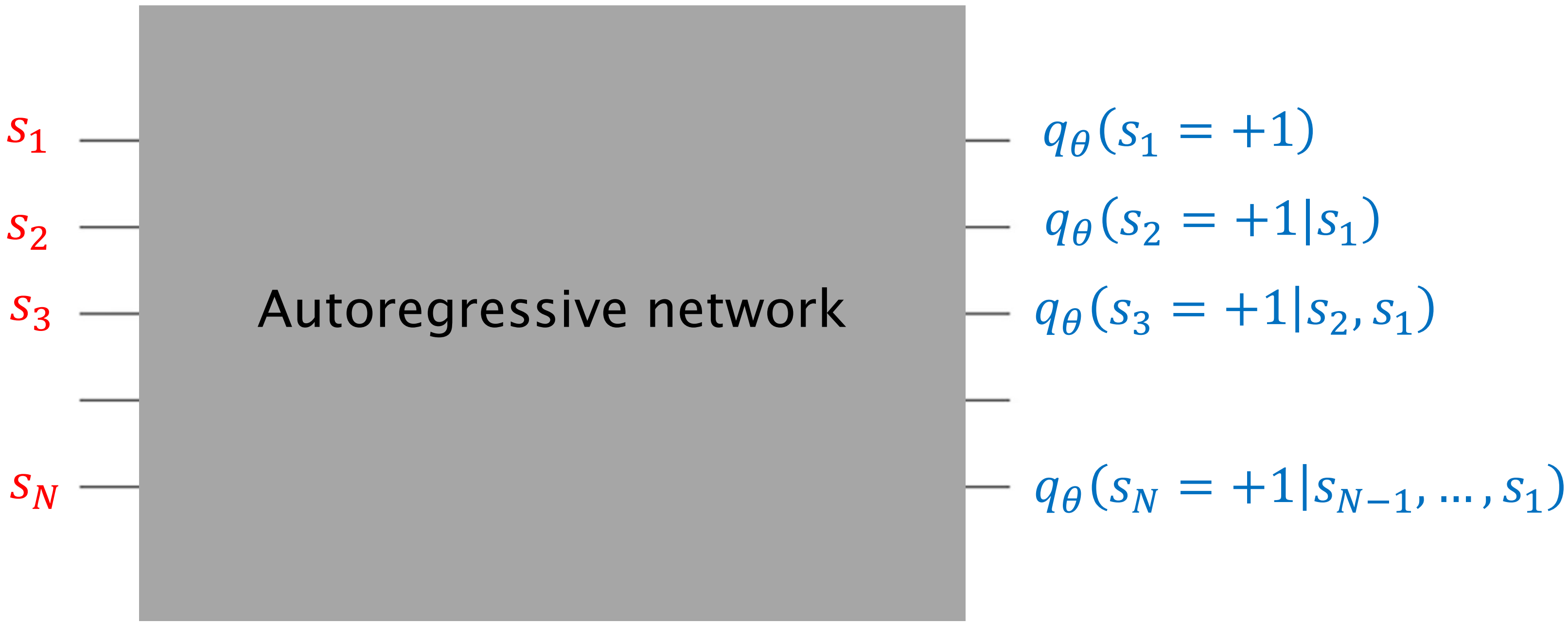
Autoregressive networks:



Half of the connections removed.

Loss function and training: recipe

Generate a spin configuration using network:



Run network N (=# spins) times – after k-th run we obtain:
$q_\theta(s_k = +1|s_{k-1}, \dots, s_1)$ and knowing it we DRAW s_k value.

Loss function and training

Training = adjust network parameters θ such that $q_\theta(\boldsymbol{s})$ is as close to $p(\boldsymbol{s}) = Z^{-1}e^{-\beta E(\boldsymbol{s})}$ as possible.

Kullback–Leibler (KL) divergence

$$D_{\mathrm{KL}}(q_\theta \parallel p) = \sum_{\mathrm{s}} q_\theta(\mathrm{s}) \ln\left(\frac{q_\theta(\mathrm{s})}{p(\mathrm{s})}\right)$$

can measure difference between two distributions.

$$D_{KL} \geq 0,$$
$$D_{KL}(q||p) = 0 \leftrightarrow q = p$$

We rewrite it using form $p(\boldsymbol{s}) = \frac{1}{Z}e^{-\beta E(\boldsymbol{s})}$:

$$D_{\mathrm{KL}}(q_\theta \parallel p) = \sum_{\mathrm{s}} q_\theta(\mathrm{s}) \ln\left(\frac{q_\theta(\mathrm{s})}{p(\mathrm{s})}\right) = \beta(F_q - F),$$

where

$$F = -\frac{1}{\beta}\log Z \qquad F_q = \frac{1}{\beta}\sum_{\mathrm{s}} q_\theta(\mathrm{s})\left[\beta E(\mathrm{s}) + \ln q_\theta(\mathrm{s})\right]$$

Variational free energy!

Loss function and training

$$D_{\mathrm{KL}}(q_\theta \parallel p) = \sum_{\mathrm{s}} q_\theta(\mathrm{s}) \ln\left(\frac{q_\theta(\mathrm{s})}{p(\mathrm{s})}\right) = \beta(F_q - F),$$

where

$$F_q = \frac{1}{\beta} \sum_{\mathrm{s}} q_\theta(\mathrm{s}) \left[\beta E(\mathrm{s}) + \ln q_\theta(\mathrm{s})\right]$$

Training = minimizing F_q

Note: here we have sum over all states.

Instead:

$$\sum_{s} q_\theta(s)[\beta E + \ln q_\theta(s)] \to \frac{1}{N_{Batch}} \sum_{n=1}^{N_{Batch}} [\beta E + \ln q_\theta(s)] \equiv \hat{F}_q$$

We estimate F_q on relatively small (~1000) batch of configurations generated using probability $q_\theta(s)$. $\hat{F}_q$ is our loss function.

Loss function and training: recipe

1) Generate a spin configuration using network.
2) Repeat point 1 N_{Batch} times to get batch of spin configurations.
3) Calculate:

$$\hat{F}_q = \frac{1}{N_{Batch}} \sum_{n=1}^{N_{Batch}} [\beta E + \ln q_\theta(\boldsymbol{s})]$$

4) Change parameters θ so that $\hat{F}_q$ is smaller – standard backward propagation in NN: calculate gradient of loss function w.r.t. parameters θ.

Points 1)-4) are called epoch.

5) Train your network for ~10000 epoch.

Results for L=16 (256 spins)

Note: the 2D Ising model we consider has analitical solution for $Z(\beta, L)$:

$$Z = \frac{1}{2}(2\sinh(2\beta))^{L^2/2}\sum_{i=i}^{4} Z_i\,, \qquad \text{(A1)}$$

where we have used the definitions

$$Z_1 = \prod_{r=0}^{L-1} 2\cosh(\tfrac{1}{2}L\gamma_{2r+1})\,, \quad Z_2 = \prod_{r=0}^{L-1} 2\sinh(\tfrac{1}{2}L\gamma_{2r+1})\,,$$

$$Z_3 = \prod_{r=0}^{L-1} 2\cosh(\tfrac{1}{2}L\gamma_{2r})\,, \qquad Z_4 = \prod_{r=0}^{L-1} 2\sinh(\tfrac{1}{2}L\gamma_{2r})\,,$$

$$\text{(A2)}$$

with the coefficients

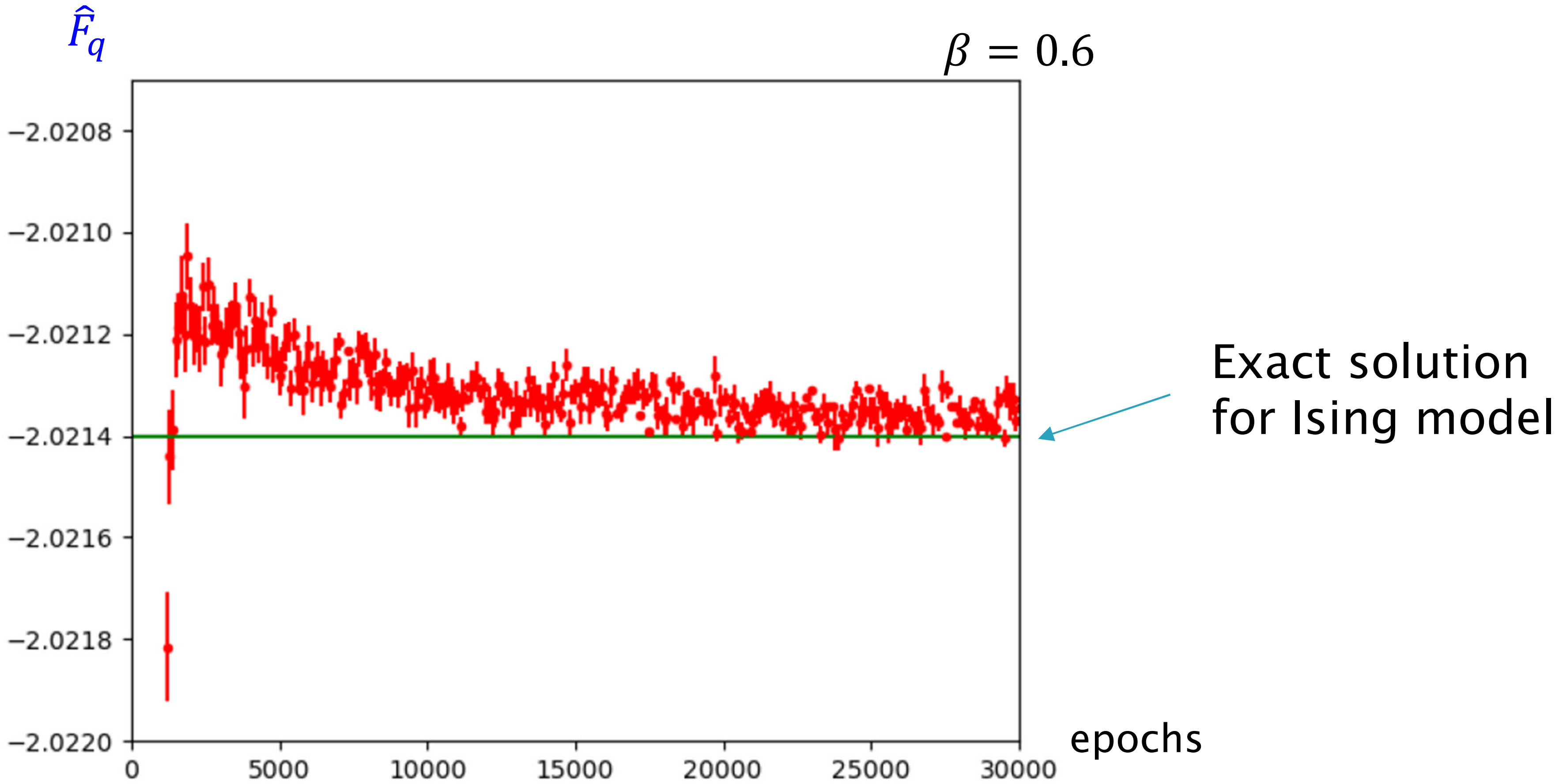
$$\gamma_0 = 2\beta + \ln\tanh\beta\,,$$

$$\gamma_r = \ln(c_r + \sqrt{c_r^2 - 1}) \qquad \text{for} \qquad r > 0\,, \qquad \text{(A3)}$$

and $c_r = \cosh 2\beta \coth 2\beta - \cos(r\pi/L)$. From this ex-

Provides benchmark for NN.

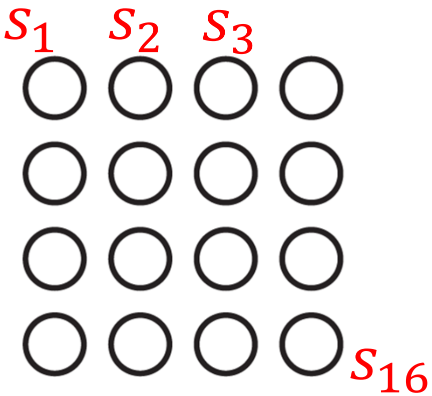
Results for L=16 (256 spins)



Difference between $q_\theta(\boldsymbol{s})$ and $p(\boldsymbol{s})$ is very small.

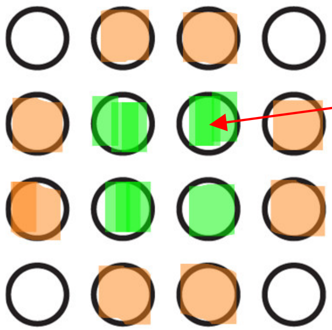
Białas, Korcyl, TS, Comput.Phys.Commun. 281 (2022) 108502

Hierarchical autoregressive networks



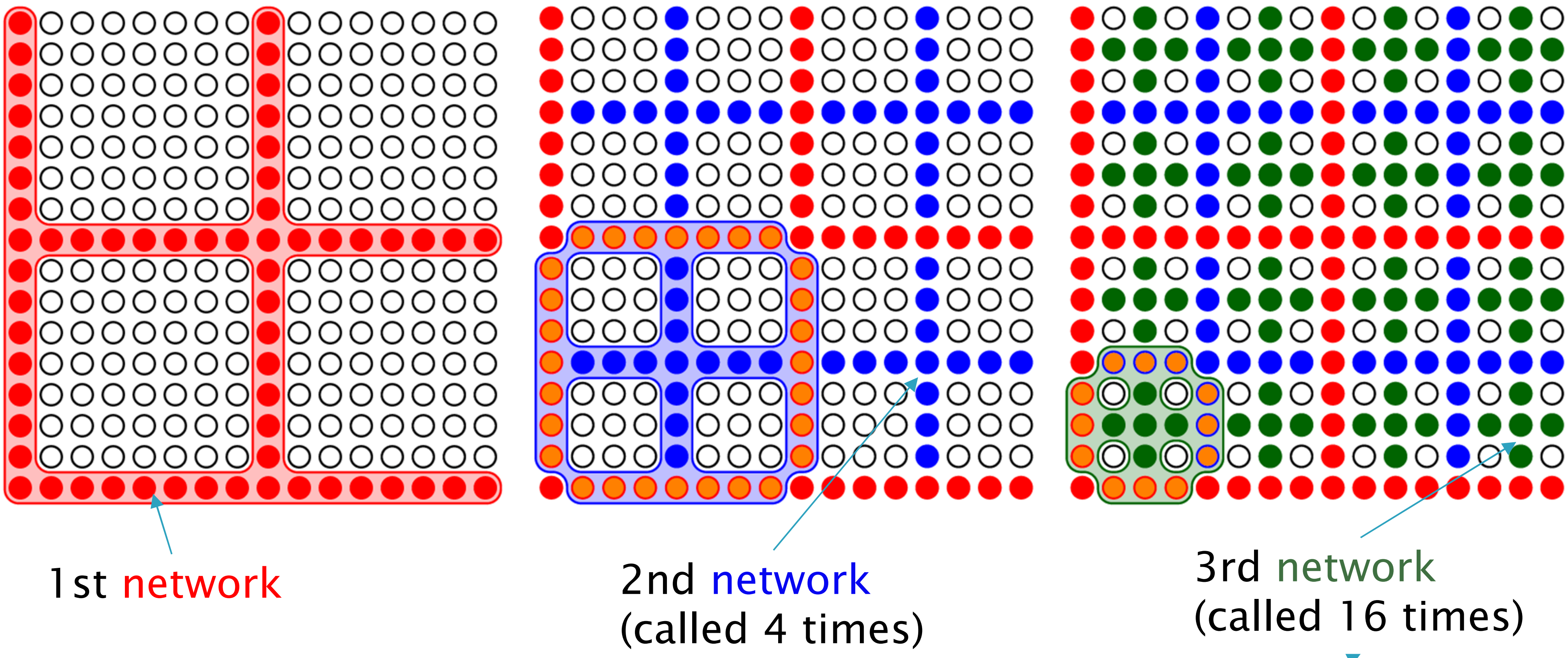
- It is there a better way to numerate the spins?

- We can use a property of Nearest Neighbour interactions:



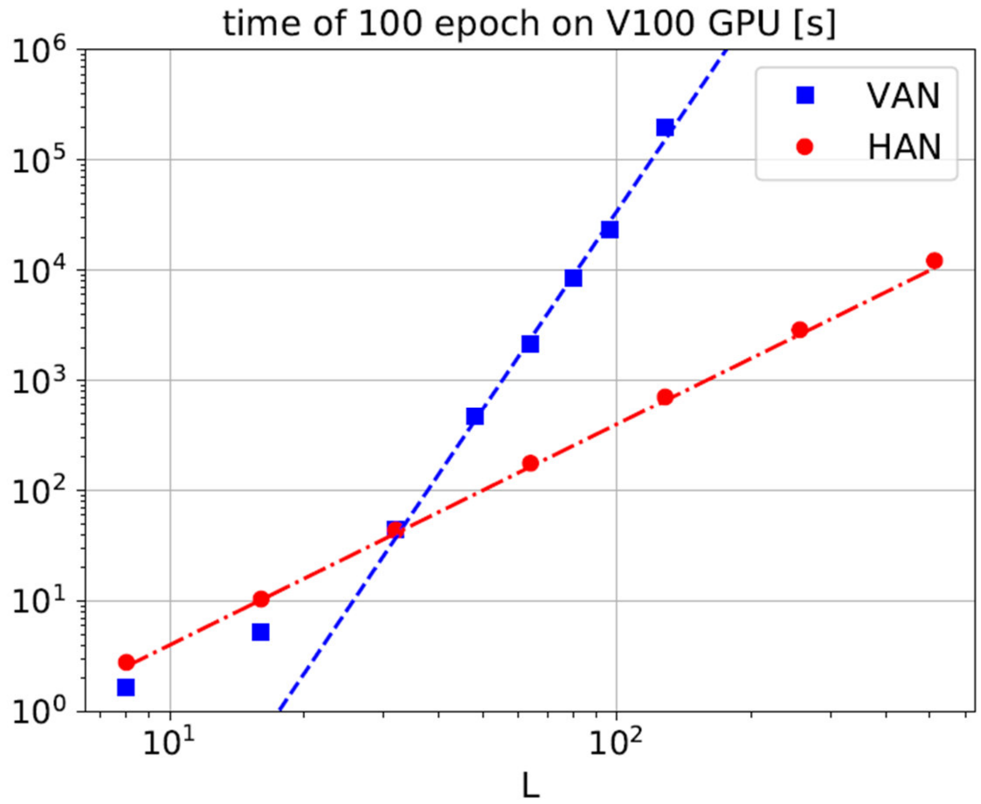
Probability of green interior depends only on orange boundary (Hammersley-Clifford theorem)

Hierarical autoregressive networks



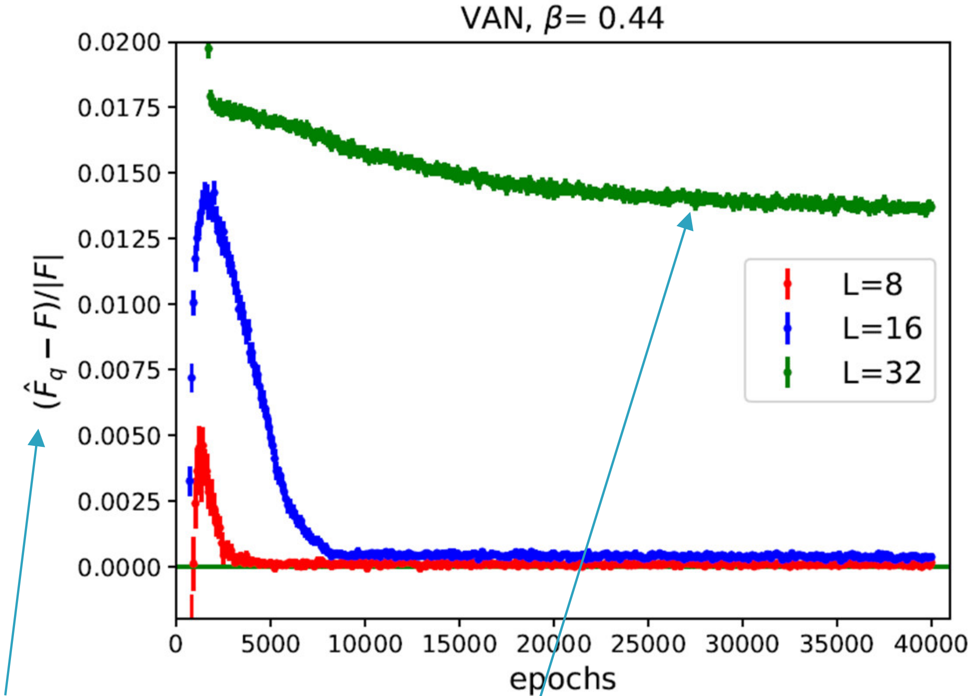
These networks have additional dependence on boundary surrounding them

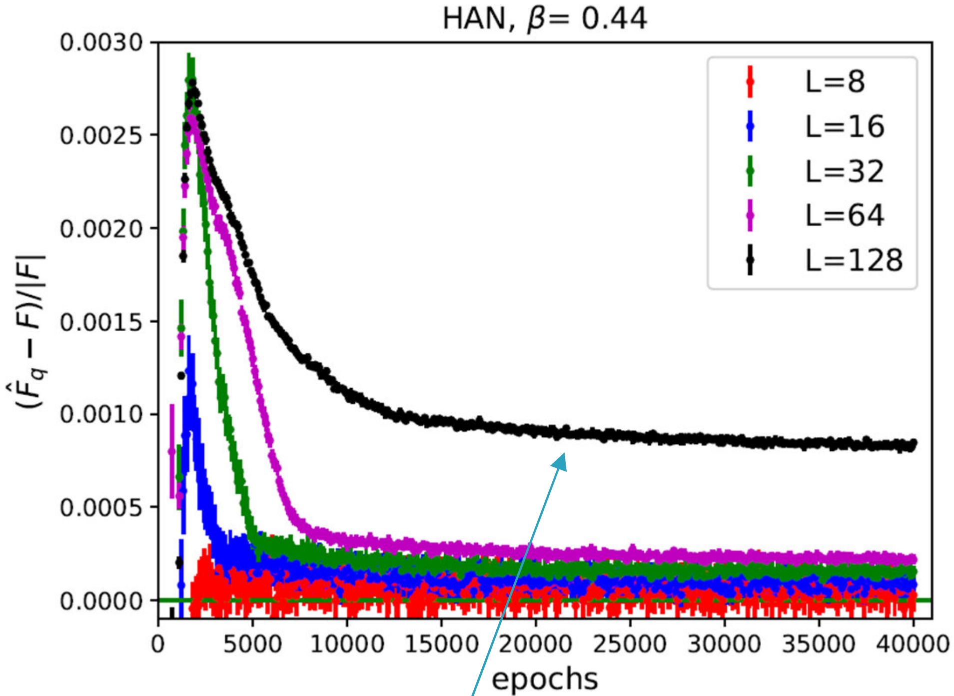
Hierarchical autoregressive networks



Much better scaling of numerical cost with system size L than original algorithm

Hierarchical autoregressive networks





Deviation from exact value

VAN: ~1.5% error for L=32

HAN: ~0.1% error for L=128

K. A. Nicoli, S. Nakajima, N. Strodtho, W. Samek, K.-R. Muller, and P. Kessel, Phys. Rev. E, vol. 101, p. 023304,

Imperfection of training

- NN cannot learn $p(s)$ perfectly. We can however correct it. There are two ways to do this:

1) Neural Importance Sampling (NIS):

Reweighting observables

$$\langle \mathcal{O}(s) \rangle_p \approx \sum_i w_i \mathcal{O}(s_i)$$

$$\text{where } w_i = \frac{\hat{w}_i}{\sum_i \hat{w}_i} \text{ for } \hat{w}_i = \frac{e^{-\beta H(s_i)}}{q(s_i)}$$

2) Neural Markov Chain Monte Carlo (NMCMC)

Here we focus on 2).

2 applications in statistical physics

Potts model with Q=12

- Potts model with 12 states:

$$H(\mathbf{s}) = -\sum_{\langle i,j \rangle} \delta_{s^i,s^j}, \qquad s^i = 1, \dots, 12$$

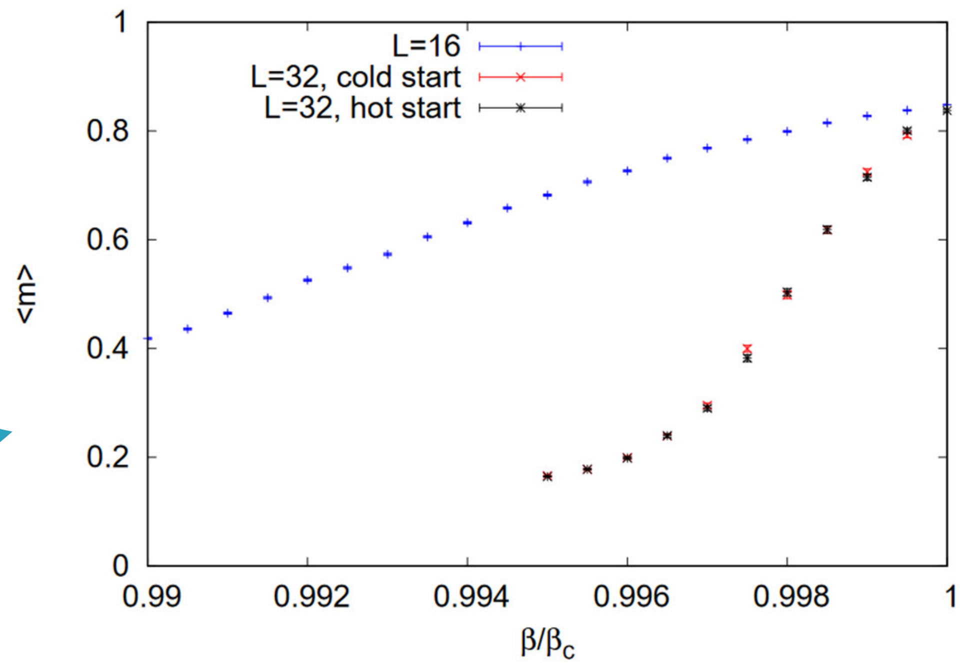
generalization of Ising model (2-state Potts model ≡ Ising model)

For Q>4 undergoes 1st order phase transition at:

$$\beta_c(Q) = \log(1+\sqrt{Q}).$$

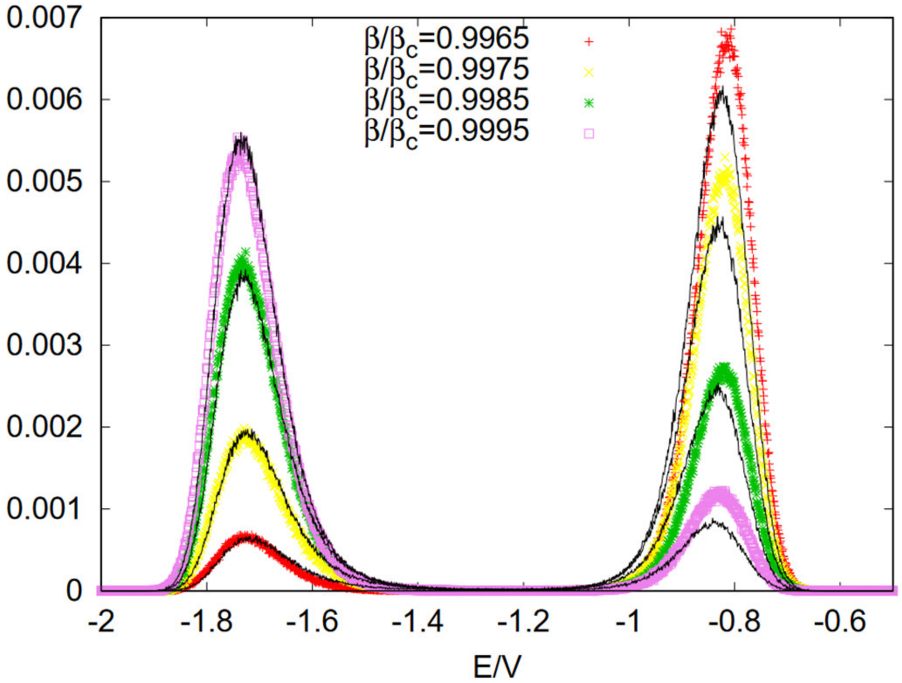
Wolff cluster algorithm

U. Wolff, Physics Letters B, vol. 228, no. 3, pp. 379–382, 1989



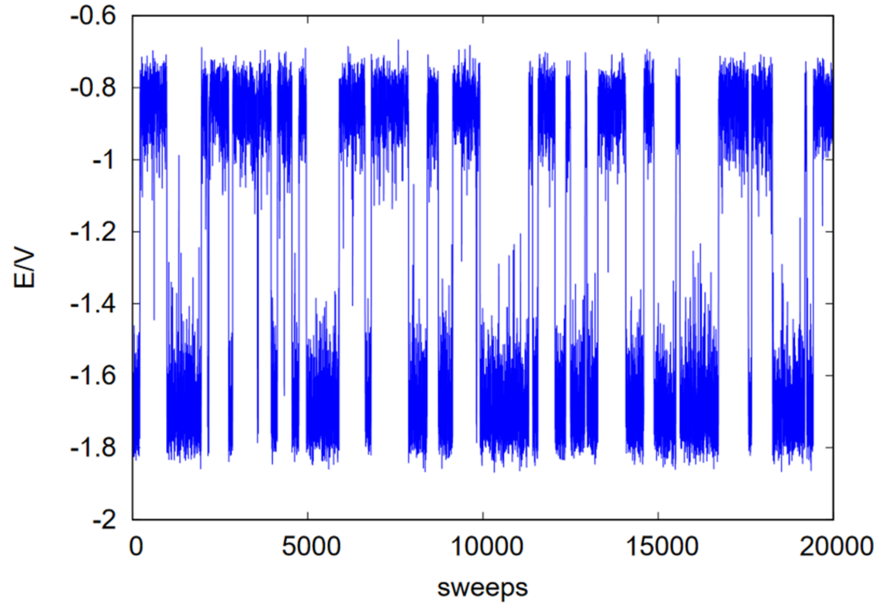
Potts model with Q=12 for L=32

Energy density histogram:



Color: HAN (neural network)

Black: Wolff algorithm



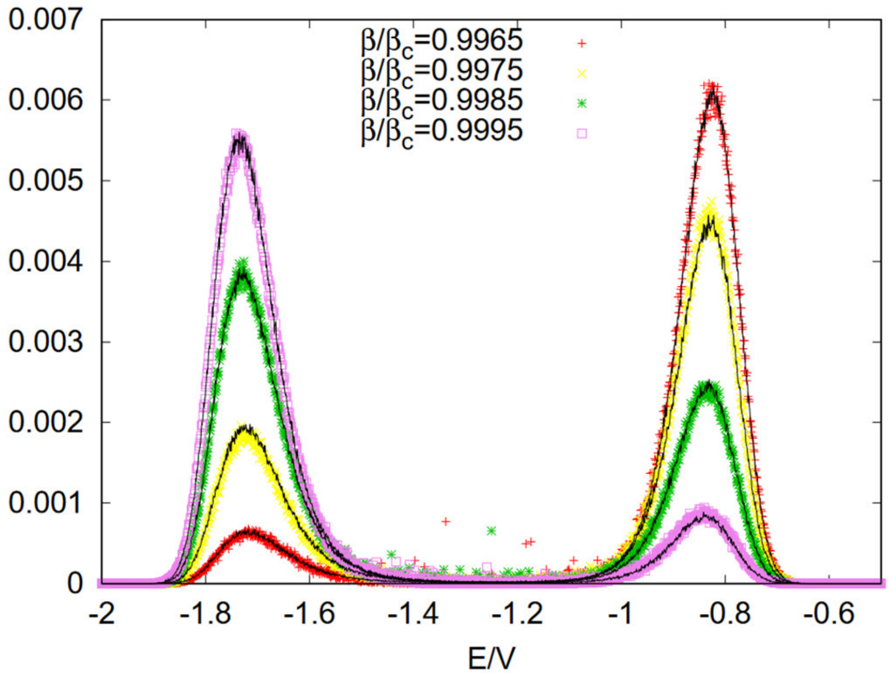
Wolff cluster algorithm leads to autocorrelation.

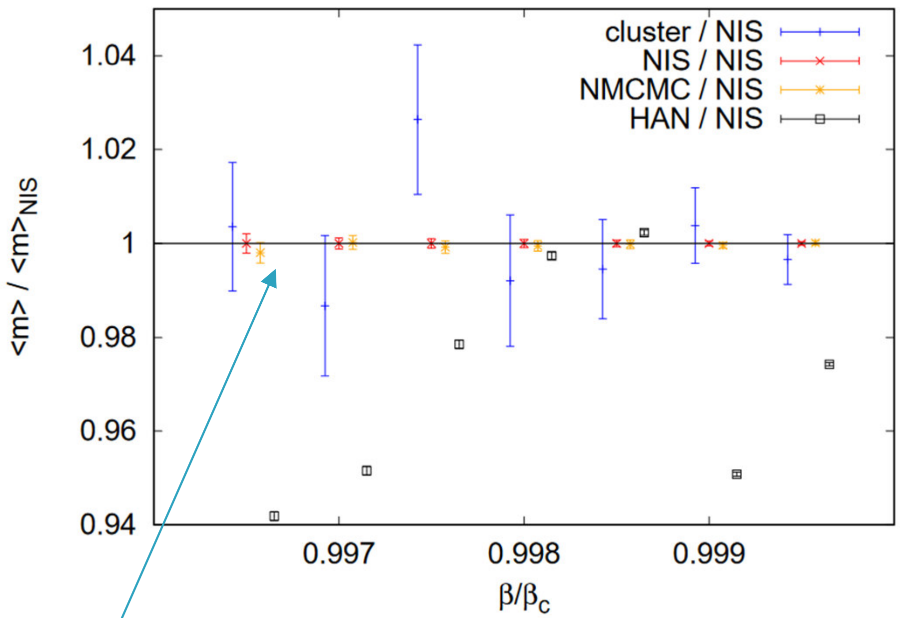
Potts model with Q=12

The difference between $q_\theta(\boldsymbol{s})$ and $p(\boldsymbol{s})$ can be canceled applying reweighting, Neural Importance Sampling (NIS):

$$\langle \mathcal{O} \rangle = \frac{1}{N\hat{Z}} \sum_{i=1}^{N} \hat{\bar{w}}(\mathbf{s}_i)\mathcal{O}(\mathbf{s}_i), \qquad \hat{\bar{w}}(\mathbf{s}) = \frac{e^{-\beta H(\mathbf{s})}}{\bar{q}_\theta(\mathbf{s})}.$$

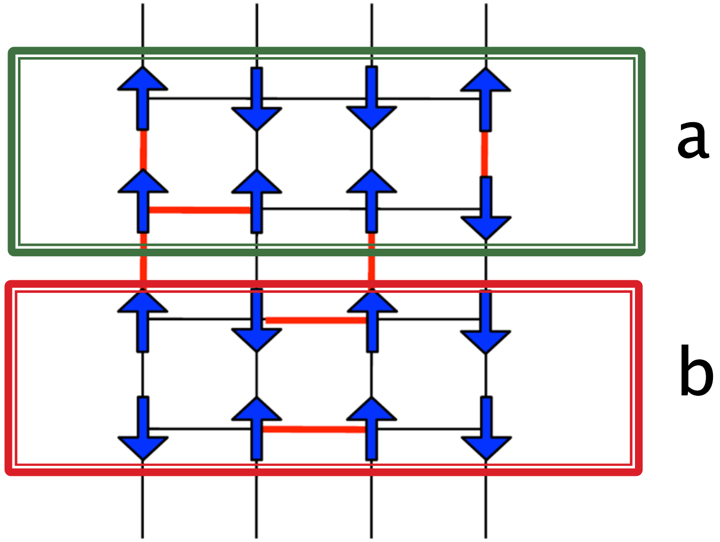
Nicoli et al., Phys. Rev. E, vol. 101, p. 023304





10x smaller errors of HAN compared to Wolff algorithm (at the same time of sampling)

Classical mutual information in Ising model



$$I = \sum_{\mathbf{a}\in A, \mathbf{b}\in B} p(\mathbf{a},\mathbf{b}) \log \frac{p(\mathbf{a},\mathbf{b})}{p(\mathbf{a})p(\mathbf{b})}$$

where

$$p(\mathbf{a},\mathbf{b}) = \frac{1}{Z} e^{-\beta E(\mathbf{a},\mathbf{b})}, \quad Z = \sum_{\mathbf{a}\in A, \mathbf{b}\in B} e^{-\beta E(\mathbf{a},\mathbf{b})}$$

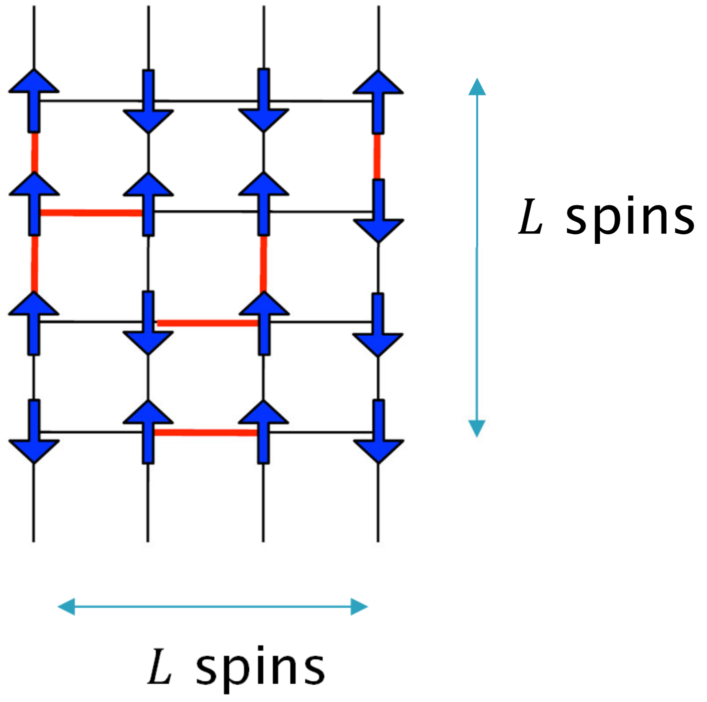
and

$$p(\mathbf{a}) = \sum_{\mathbf{b}\in B} p(\mathbf{a},\mathbf{b}), \quad p(\mathbf{b}) = \sum_{\mathbf{a}\in A} p(\mathbf{a},\mathbf{b})$$

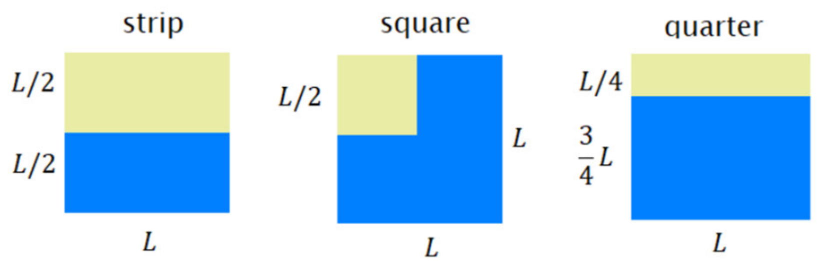
$$p(\mathbf{a},\mathbf{b}) =$$
$$= \underbrace{p(s_1)p(s_2|s_1)p(s_3|s_2,s_1)\ldots p(s_{\frac{N}{2}}|s_{\frac{N}{2}-1},\ldots,s_1)}_{p(\mathbf{a})} \underbrace{p(s_{\frac{N}{2}+1}|s_{N/2},\ldots,s_1)\ldots p(s_N|s_{N-1},\ldots,s_1)}_{p(\mathbf{b}|\mathbf{a})}$$

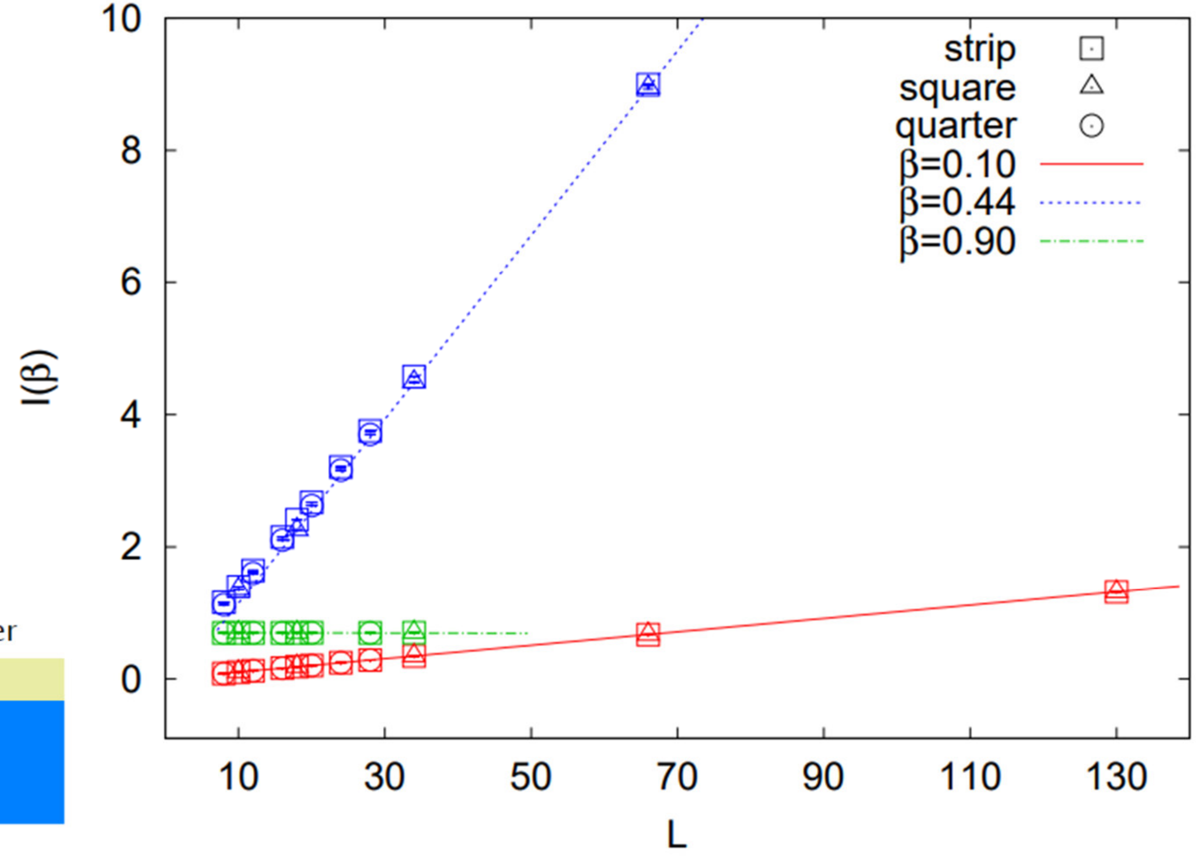
Autregressive networks allow to calculate I

Classical mutual information in Ising model



$$I = \sum_{\mathbf{a}\in A, \mathbf{b}\in B} p(\mathbf{a}, \mathbf{b}) \log \frac{p(\mathbf{a}, \mathbf{b})}{p(\mathbf{a})p(\mathbf{b})}$$

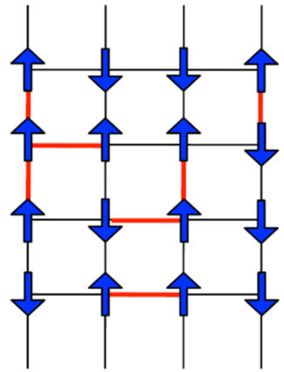




Area law

2. Sampling models with continuous d.o.f.

ϕ^4 field theory on 2d lattice



Now we have real number in each site

- After discretization:

$$S^E_{\mathrm{latt}}(\phi) = \sum_{\vec{n}} \phi(\vec{n}) \left[\sum_{\mu \in \{1,2\}} 2\phi(\vec{n}) - \phi(\vec{n}+\hat{\mu}) - \phi(\vec{n}-\hat{\mu}) \right] + m^2\phi(\vec{n})^2 + \lambda\phi(\vec{n})^4$$

$$p(\phi) = \frac{1}{Z} e^{-S(\phi)}, \quad Z \equiv \int \prod_{\vec{n}} d\phi(\vec{n})\; e^{-S(\phi)},$$

Idea behind Normalizing Flows: toy example

- Problem: how to sample from 2d Gaussian distribution?

1) We draw to random numbers U_1 and U_2 from uniform distribution at (0,1):

$$r(U_1, U_2) = 1 \quad for\ U_1, U_2 \in (0,1)$$

r is called prior distribution

2) We apply transformation:

$$Z_1 = \sqrt{-2 \ln U_1} \cos(2\pi U_2) \quad \text{and} \quad Z_2 = \sqrt{-2 \ln U_1} \sin(2\pi U_2)$$

3) Then Z_1 and Z_2 are distributed according to:

$$q(Z_1, Z_2) = r(U_1, U_2) \left| \det_{kl} \frac{\partial Z_k(U_1, U_2)}{\partial U_l} \right|^{-1}$$

$$= \frac{1}{2\pi} e^{-(Z_1^2 + Z_2^2)/2}$$

Normalizing Flows

- Our goal:

find distribution $q_\theta(\phi)$ as close as possible to Boltzman distribution $p(\phi)$.

Normalizing Flows:

Use some simple prior distribution q_{pr} and some transformation $f: R^n \to R^n$ to construct q_θ.

1) Sample variable z from $q_{pr}(z)$,
2) Field configuration is:

$$\phi = f(\mathrm{z})$$

3) Probability of configuration:

$$q_\theta(\phi) = q_{pr}(z) \left| \det \frac{\partial f}{\partial z} \right|^{-1}$$

f must be bijective and Jacobian should be easy to compute.

Normalizing Flows

For ϕ^4 theory we don't know the form of q_{pr} and f.

In Normalizing Flows one uses simple form of q_{pr} (e.g. uniform or Gaussian) and neural network plays a role of f.

Kullback-Leibler (KL) divergence:

$$D_{\mathrm{KL}}(q_\theta \,\|\, p) = \beta(F_q - F),$$

where variational free energy:

$$F_q = \int dz\, q_{pr}(z)\left[\log q_\theta\big(f(z)\big) + S_E\big(f(z)\big)\right]$$

As previously, this is our loss function for network training.

Summary

- Neural networks can learn probability distributions.
- For discrete d.o.f. one can use autoregressive networks (conditional probabilities training).
- For continuous d.o.f. one uses Normalizing Flows („change of variables”)
- Both approaches not only gives configurations but also probabilities.
- We observe fast progress in this field!

Thank you