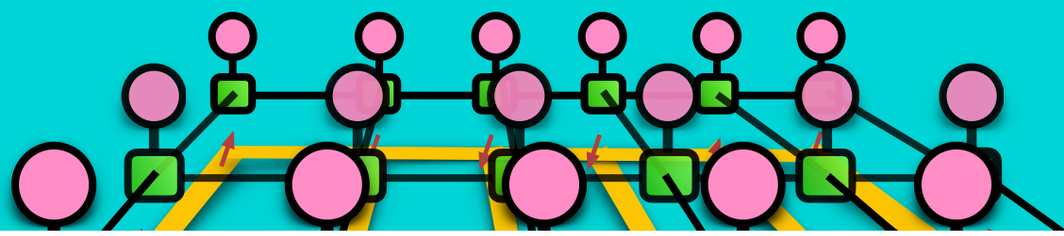
# RECURRENT NEURAL NETWORKS FOR MANY-BODY PHYSICS

**Juan Felipe Carrasquilla Álvarez**
**Vector Institute**

**Machine Learning in Quantum Physics and Chemistry**
**Warsaw**
**August 24, 2021**



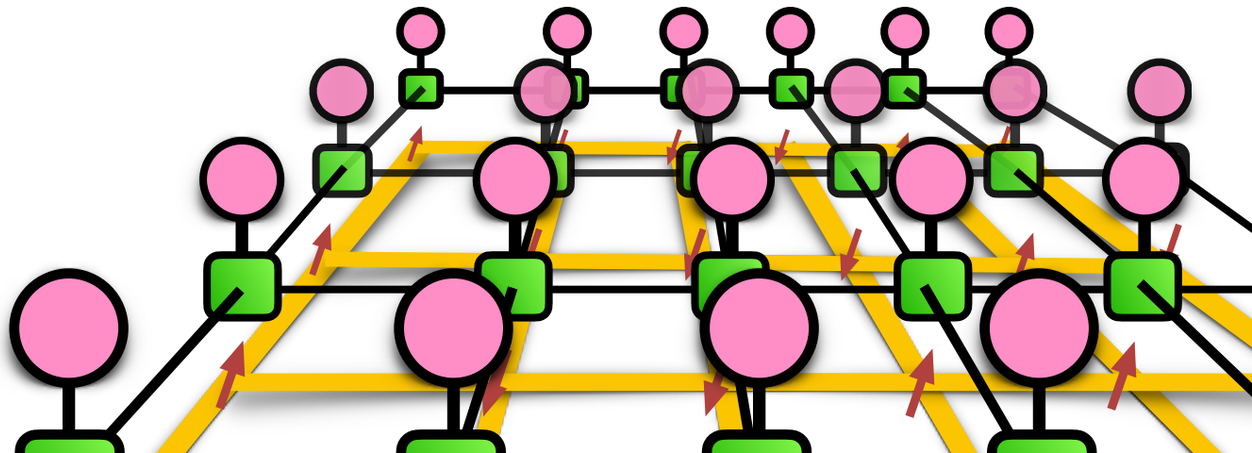
VECTOR INSTITUTE | INSTITUT VECTEUR

UNIVERSITY OF WATERLOO

CIFAR

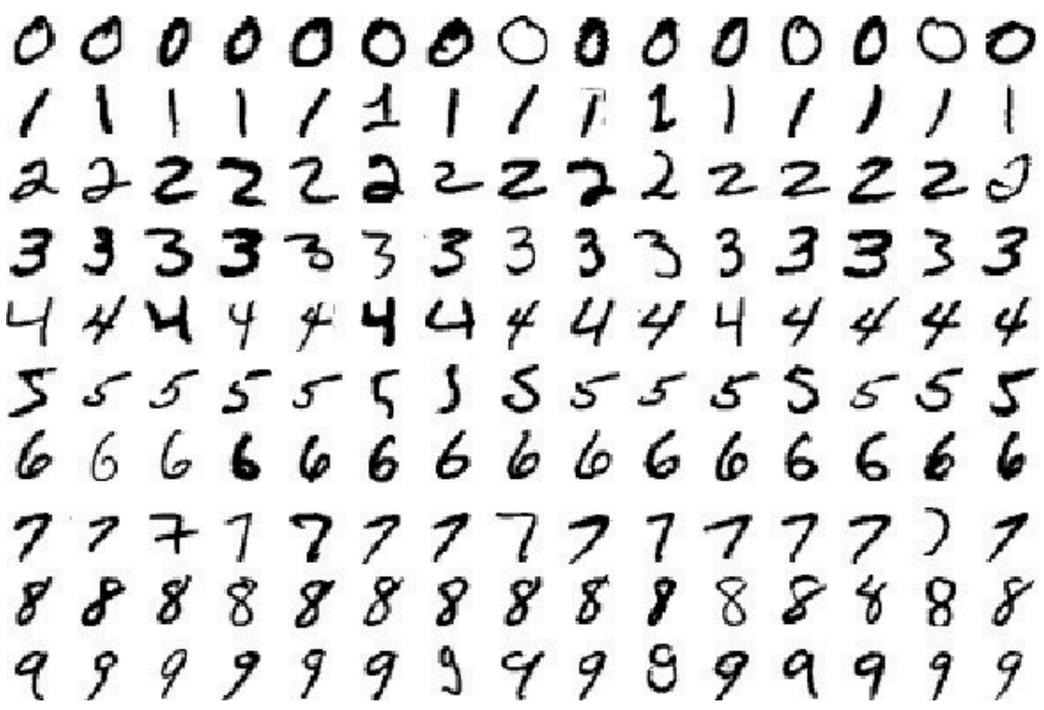
NSERC CRSNG

# PROBABILISTIC MODELS



# PROBABILISTIC MODELS AND INFERENCE

- We assume that there is a probability distribution underlying $x \sim P(x)$
- We don't know what this distribution looks like in most cases
- We can model this distribution $P_{\boldsymbol{\theta}}(x)$
- We can use the observations $x$ to infer $\boldsymbol{\theta}$

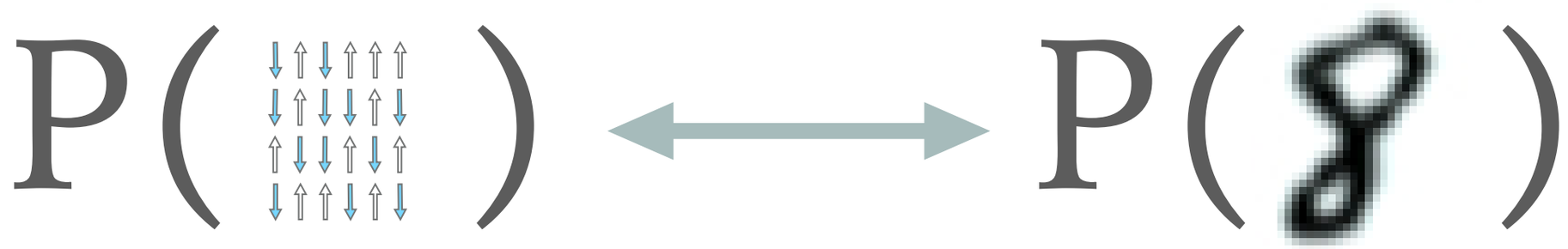


$$P_{\boldsymbol{\theta}}(\text{8}) = ?$$

# IN ML PEOPLE STUDY P(IMAGE), IN STAT MECH ?

- Boltzmann distribution

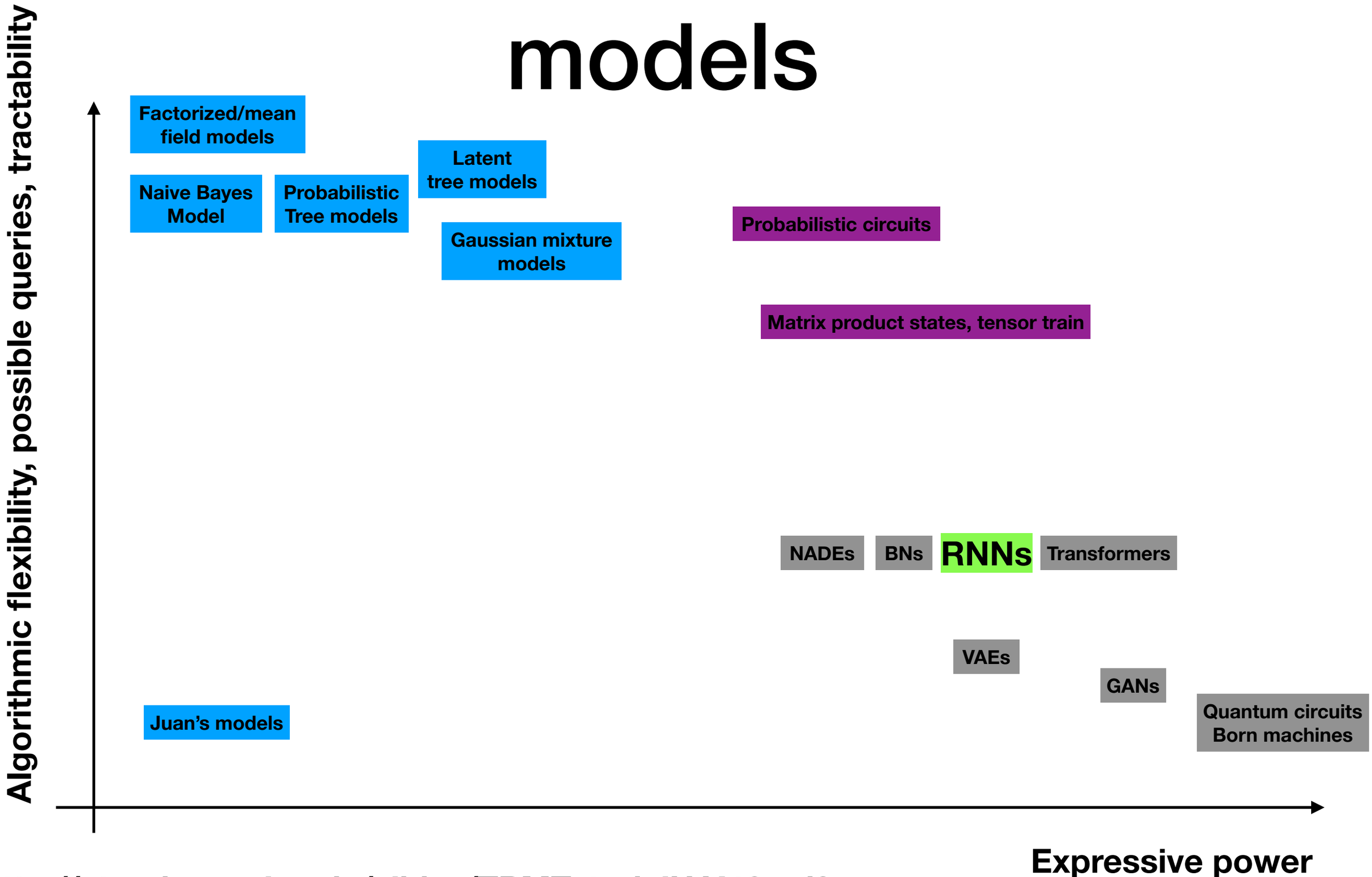
$$P(E) = \frac{e^{-E/k_B T}}{Z} \qquad E = -J \sum_{\langle i,j \rangle} \sigma_i \sigma_j$$




- Wavefunction $\Psi$( )

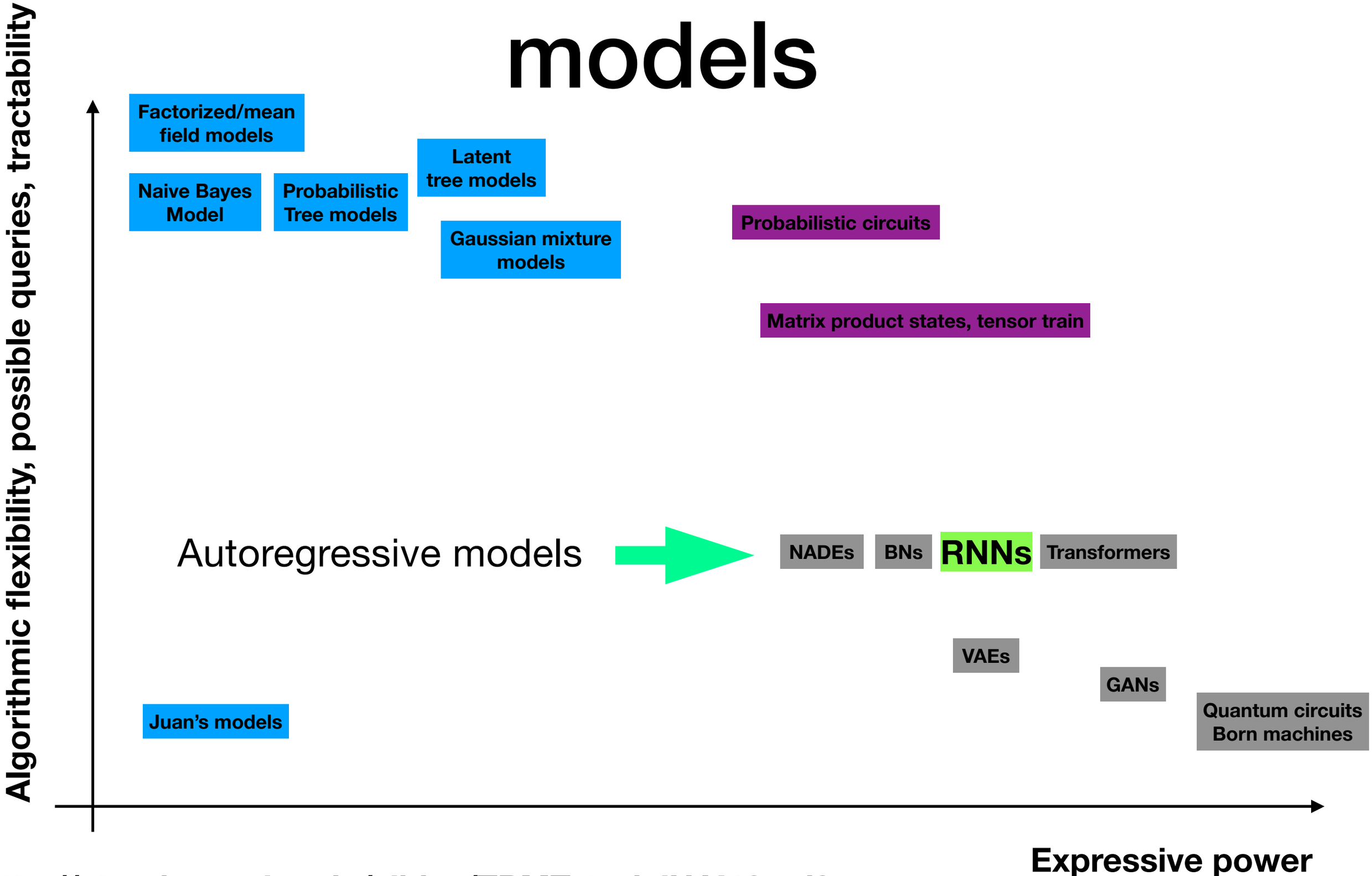
ML and statistical (and quantum) physics are interested in similar high dimensional distributions and wavefunctions

# Landscape of probabilistic models



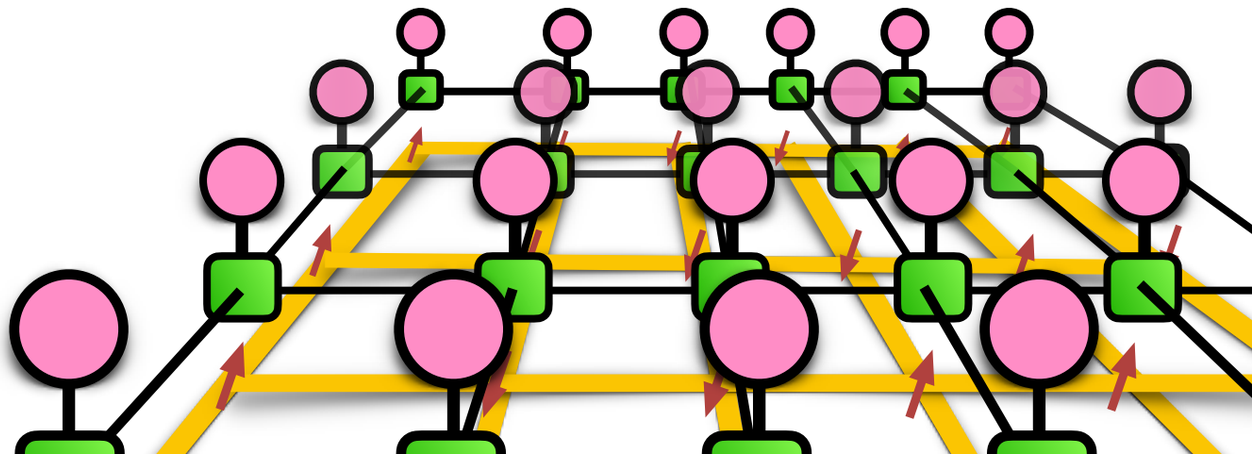

http://starai.cs.ucla.edu/slides/TPMTutorialUAI19.pdf

# Landscape of probabilistic models




http://starai.cs.ucla.edu/slides/TPMTutorialUAI19.pdf

# WHAT ARE AUTOREGRESSIVE MODELS?



# PROBABILISTIC AUTOREGRESSIVE MODELS

- The term *autoregressive* originates from the literature on time-series models where observations from the previous time-steps are used to predict the value at the current time step.
- Consider a probability distribution $P(\boldsymbol{\sigma}) = P(\sigma_1, \sigma_2, \ldots, \sigma_N)$, with $\sigma_i = 0,1,\ldots, d_v - 1$

$$P(\sigma_1, \sigma_2, \ldots, \sigma_N) = P(\sigma_1)P(\sigma_2|\sigma_1)P(\sigma_3|\sigma_1, \sigma_2)\ldots P(\sigma_N|\sigma_1, \sigma_2, \ldots, \sigma_{N-1})$$

- To specify P in a tabular form requires exponential resources
- Insight from ML—> Parametrize the conditionals

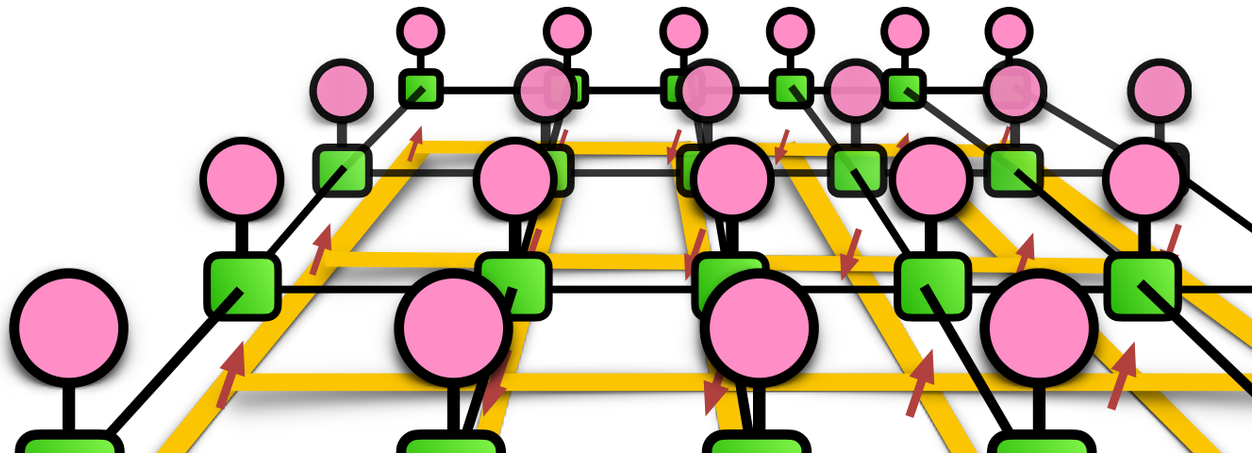
$$P(\sigma_i|\sigma_{<i}) = P_\theta(\sigma_i|\sigma_{<i})$$

# PROBABILISTIC AUTOREGRESSIVE MODELS

- Advantages:
- Can be exactly sampled in polynomial time
- Computing the probability of a configuration $P(\boldsymbol{\sigma}) = P(\sigma_1, \sigma_2, \ldots, \sigma_N)$ is efficient
- Can be defined in any spatial dimension d
- Contain mean-field theory (e.g. Gutzwiller mean-field theory)
- We can impose symmetry and other inductive biases.
- Disadvantages(?):
- Expectation value over P requires sampling (observables, gradients) —> Error bars

These three properties remain true for autoregressive models of the quantum state

# A CANONICAL EXAMPLE: THE RECURRENT NEURAL NETWORK



# Recurrent neural networks

- RNNs parameterize probability distributions specifying the conditionals $P(\sigma_i \mid \sigma_{<i})$

- The elementary building block of an RNN is a recurrent cell— it helps us specify the conditionals

- Recurrent relation $\boldsymbol{h}_n = f\left(W[\boldsymbol{h}_{n-1}; \boldsymbol{\sigma}_{n-1}] + \boldsymbol{b}\right)$

- $\boldsymbol{h}_n \in \mathbb{R}^{d_h}$ is the hidden dimension of the RNN —expressive power.

- $W \in \mathbb{R}^{(d_h+d_v)\times d_h}$, $\boldsymbol{b} \in \mathbb{R}^{d_h}$ are variational parameters.

- $f$ is a non-linear function, tanh for a vanilla RNN.

# Recurrent neural networks

- $\boldsymbol{h}_n = F(\boldsymbol{\sigma}_{n-1}, \boldsymbol{\sigma}_{n-2}, \ldots, \boldsymbol{\sigma}_1, \boldsymbol{h}_0, W, \boldsymbol{b})$ — can model $P(\sigma_n \,|\, \sigma_{n-1}, \ldots, \sigma_2, \sigma_1)$

- $\boldsymbol{h}_0$, $\sigma_0$ are arbitrary. They can be set to zero or be trained.

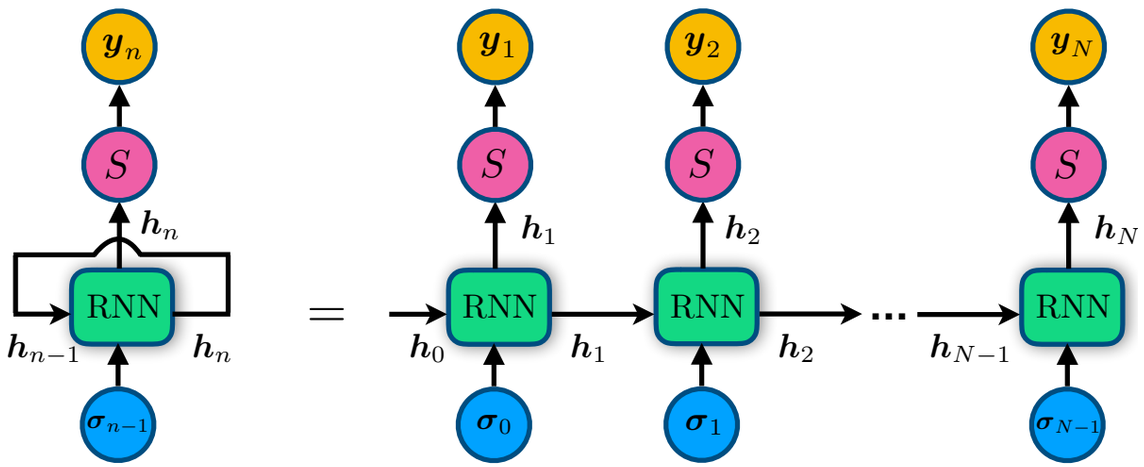
- Notice that the parameters W and b are shared at each call the function— compact representation.

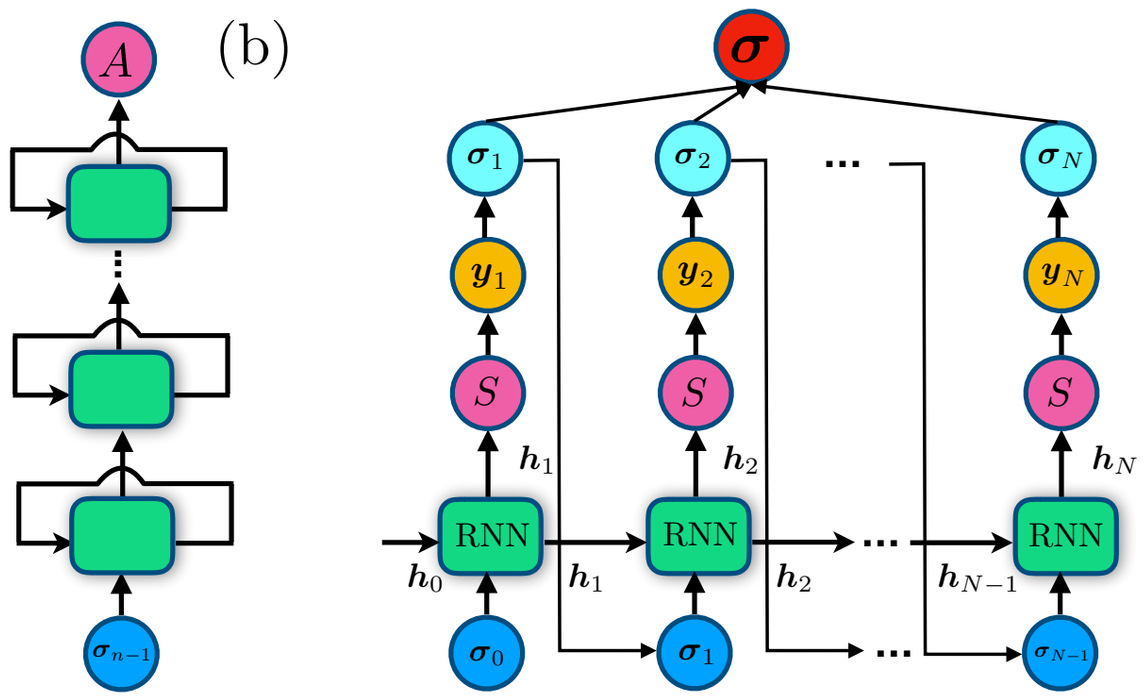
- This can be relaxed and is potentially useful.

- RNNs are universal function approximators. Schäfer and Zimmermann (2006)

# RECURRENT NEURAL NETWORKS (RNN)

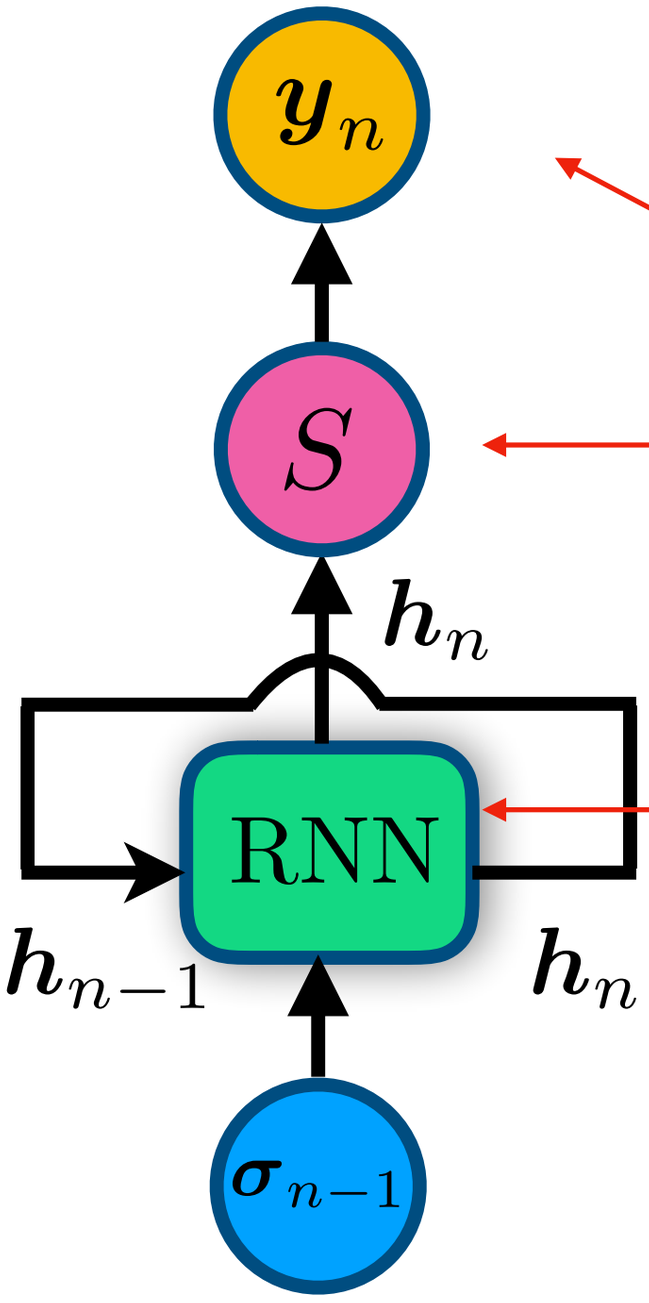
- Building blocks: RNN cell and sampling




$$P(\sigma_1, \sigma_2, \ldots, \sigma_N) = P(\sigma_1)P(\sigma_2|\sigma_1)P(\sigma_3|\sigma_1, \sigma_2)\ldots P(\sigma_N|\sigma_1, \sigma_2, \ldots, \sigma_{N-1})$$

# Recurrent neural networks




- Here, $U \in \mathbb{R}^{d_\nu \times d_h}$ and $\boldsymbol{c} \in \mathbb{R}^{d_\nu}$ are weights and biases of a so-called softmax layer
- $S(v_n) = \dfrac{\exp(v_n)}{\sum_i \exp(v_i)}$ It is a normalized probability

$$\boldsymbol{y}_n \equiv S\left(U\boldsymbol{h}_n + \mathbf{c}\right) = P_\theta(\boldsymbol{\sigma}_n \,|\, \boldsymbol{\sigma}_{n-1}, \ldots, \boldsymbol{\sigma}_1)$$

**RNN model**

$$\boldsymbol{h}_n = f\left(W[\boldsymbol{h}_{n-1}; \boldsymbol{\sigma}_{n-1}] + \boldsymbol{b}\right)$$

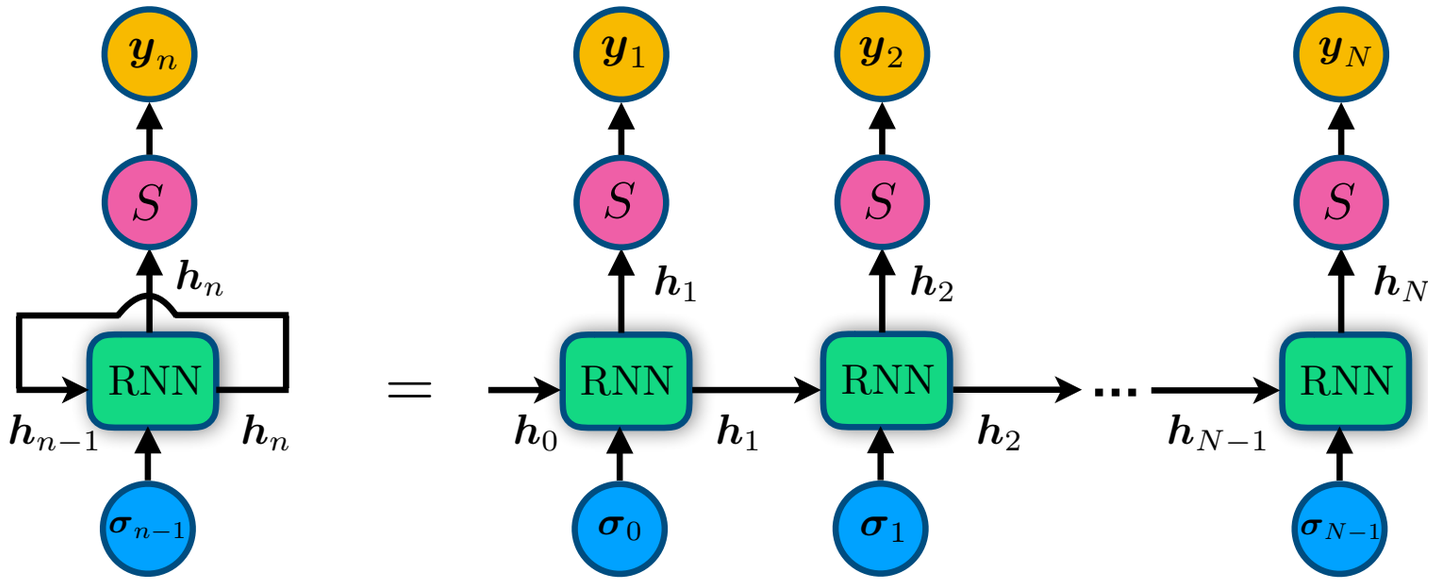
**RNN cell**

$$\boldsymbol{\sigma}_{n-1} = (1,0), (0,1)$$

**Input encoding ( $d_\nu = 2$ )**

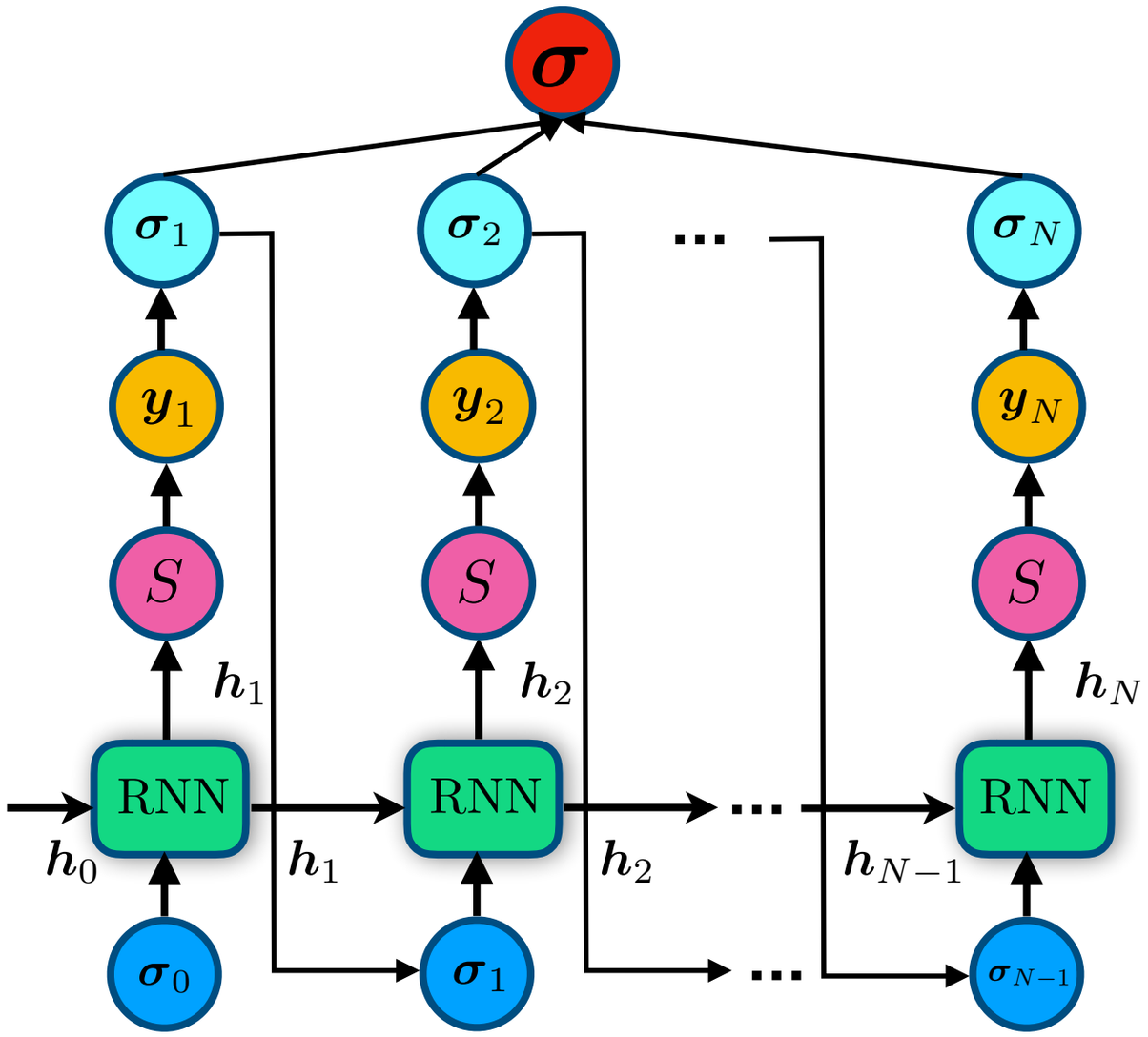
$$P(\sigma_1, \sigma_2, \ldots, \sigma_N) = P(\sigma_1)P(\sigma_2|\sigma_1)P(\sigma_3|\sigma_1, \sigma_2)\ldots P(\sigma_N|\sigma_1, \sigma_2, \ldots, \sigma_{N-1})$$

# Recurrent neural networks
# Unrolled version and forward pass




- Given a configuration $\sigma'_1, \ldots, \sigma'_N$, evaluate $P(\boldsymbol{\sigma}'_1, \ldots, \boldsymbol{\sigma}'_N)$.
- Evaluate $\boldsymbol{y}_1 = P(\boldsymbol{\sigma}_1)$ ( a vector of $d_\nu$ entries). Choose the entry corresponding to $\sigma'_1$
- Compute $\boldsymbol{y}_2 = P(\boldsymbol{\sigma}_2 \,|\, \boldsymbol{\sigma}'_1)$ , choose the entry that corresponds to $\sigma'_2$
- Repeat until $\boldsymbol{\sigma}'_N$
- Compute $P(\boldsymbol{\sigma}'_1, \ldots, \boldsymbol{\sigma}'_N)$ as a product of the conditionals computed above.

# Sampling




- Compute $P(\boldsymbol{\sigma}_1)$ $(\in \mathbb{R}^{d_\nu})$.
  Sample it $\rightarrow \boldsymbol{\sigma}'_1$
- Compute $P(\boldsymbol{\sigma}_2 \,|\, \boldsymbol{\sigma}'_1)$
  Sample it $\rightarrow \boldsymbol{\sigma}'_2$
- Repeat until $\boldsymbol{\sigma}'_N$
- If required, compute $P(\boldsymbol{\sigma}'_1, \ldots, \boldsymbol{\sigma}'_N)$ as a product of the conditionals seen during the sampling

# RNNs as probability

- $P(\boldsymbol{\sigma})$ is already properly normalized and its computation is **efficient** as long as the number of parameters remains a polynomial of the size of the system.

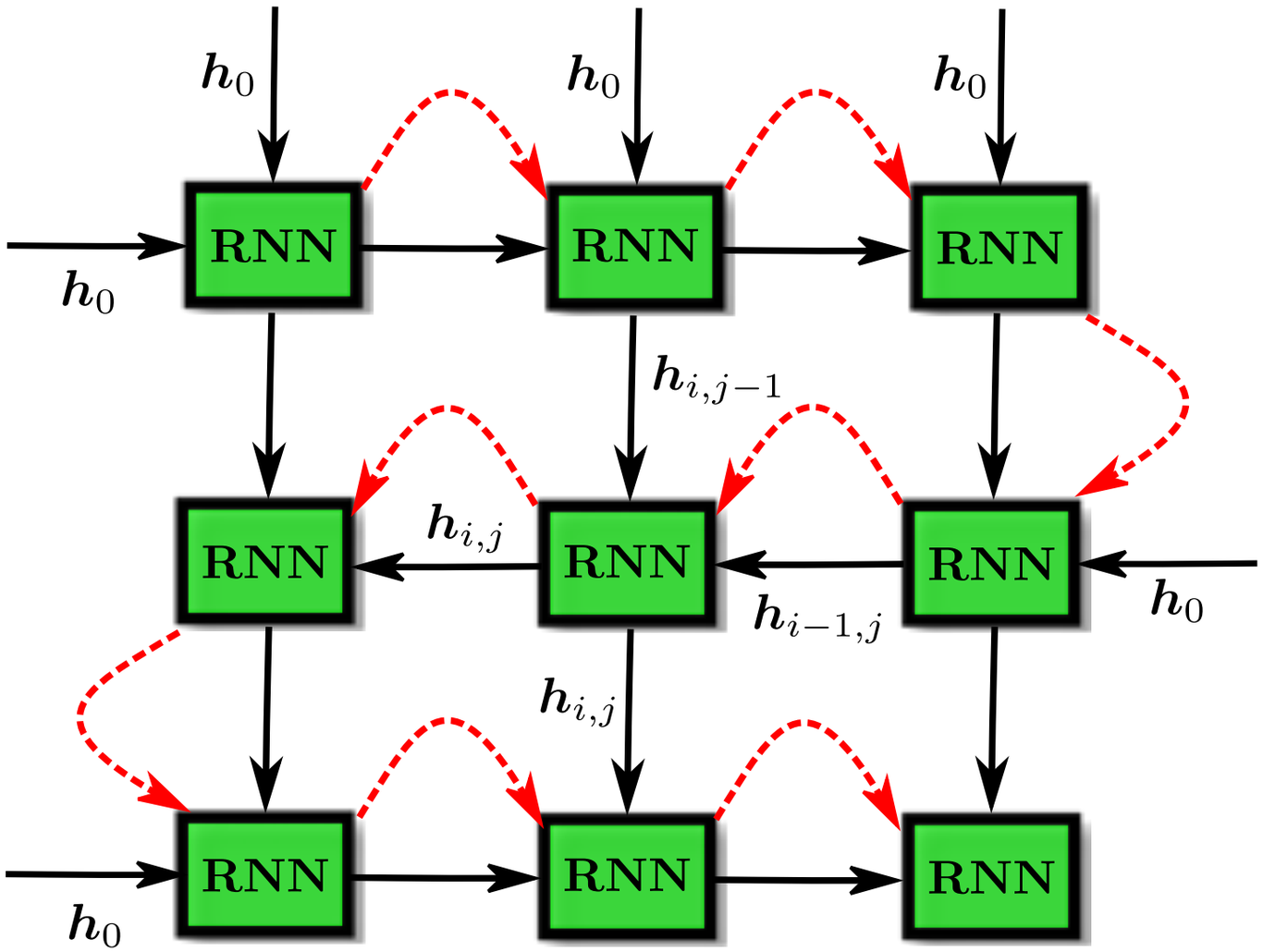
- Sampling from an RNN is **efficient too.**

- These are perfect samples, ie. they are uncorrelated—no need to perform expensive Markov chain Monte Carlo simulations

# BUT THERE ARE MORE AND LIST IS LONG

- Transformers
- Neural autoregressive density estimators
- Autoregressive flows
- Masked Autoencoder for Distribution Estimation
- PixelRNN
- PixelCNN
- Wavenet
- ....

# 2-D RNN

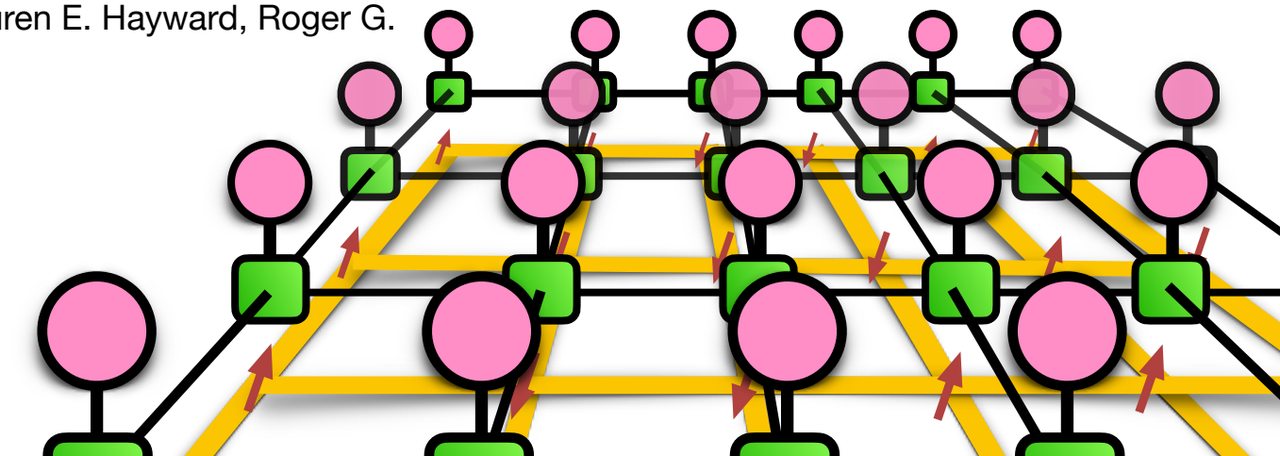



- $\boldsymbol{h}_{i,j} = F\left(W_{i,j}^{(h)}[\boldsymbol{h}_{i-1,j}; \boldsymbol{\sigma}_{i-1,j}] + W_{i,j}^{(v)}[\boldsymbol{h}_{i,j-1}; \boldsymbol{\sigma}_{i,j-1}] + \boldsymbol{b}_{i,j}\right)$

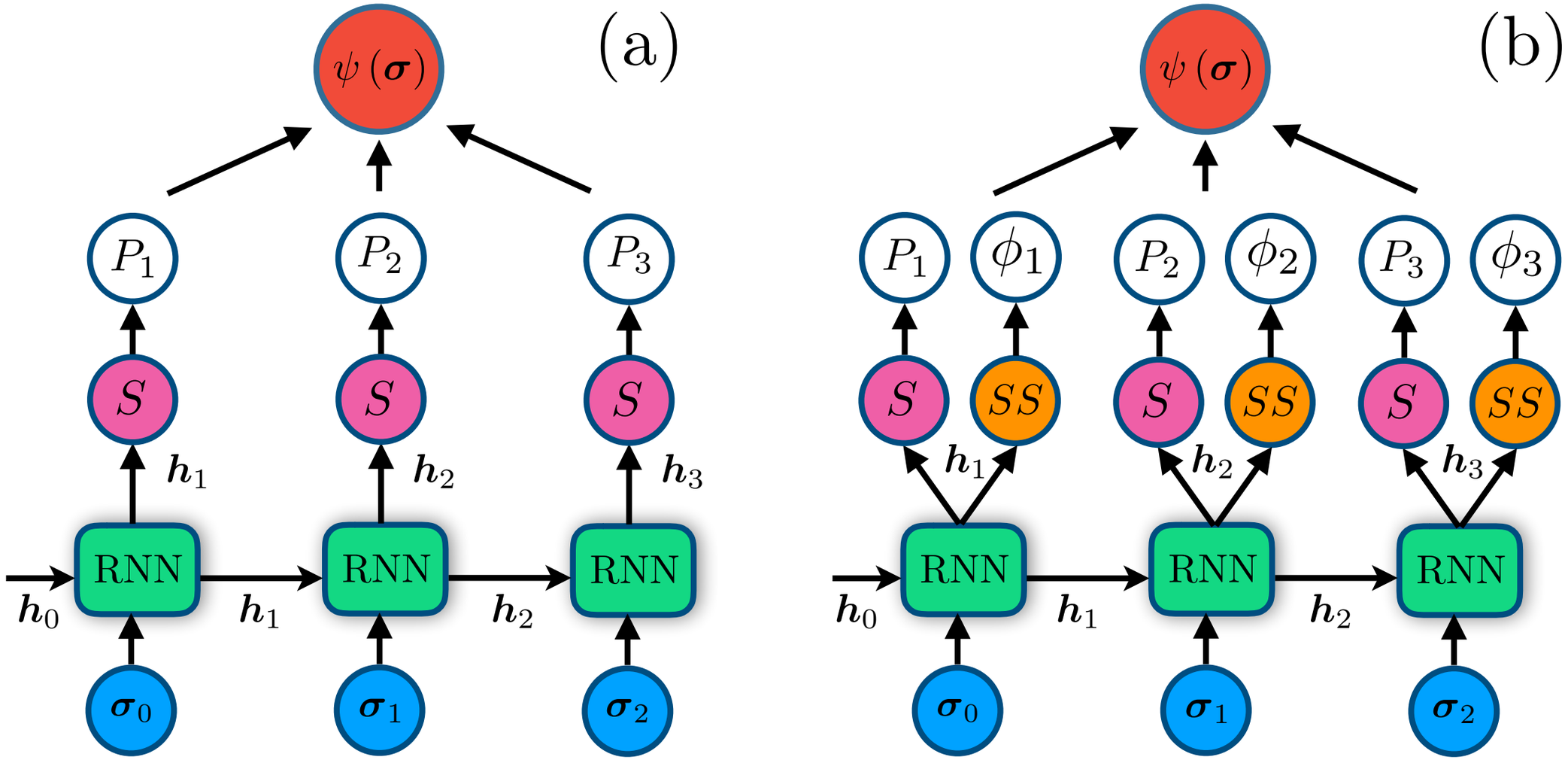
# RECURRENT NEURAL NETWORK WAVEFUNCTIONS

Mohamed Hibat-Allah, Martin Ganahl, Lauren E. Hayward, Roger G. Melko, and Juan Carrasquilla

Phys. Rev. Research **2**, 023358 (2020)



# RNNs wave functions

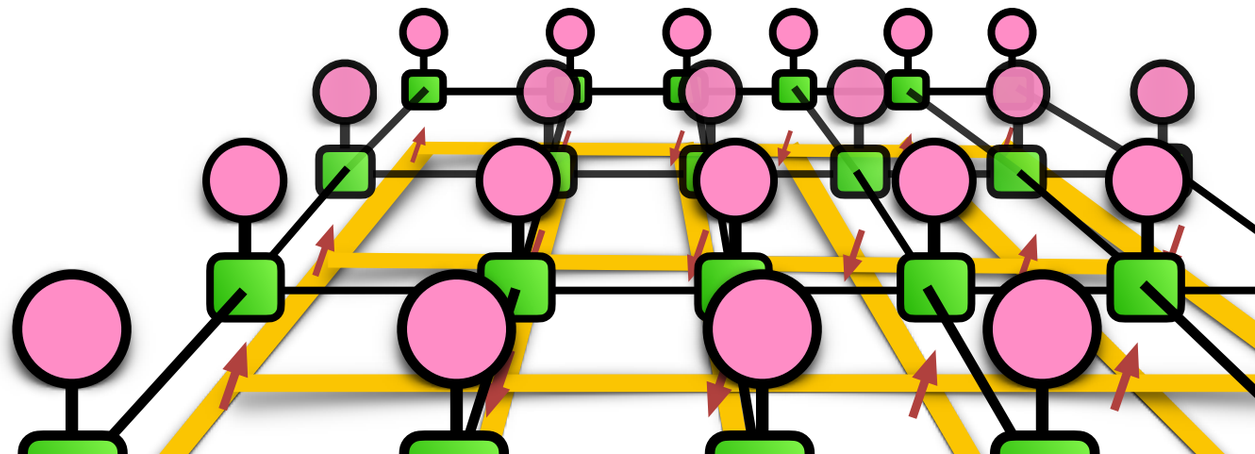



$$|\Psi\rangle = \sum_{\boldsymbol{\sigma}} \psi(\boldsymbol{\sigma})\,|\boldsymbol{\sigma}\rangle = \sum_{\boldsymbol{\sigma}} \sqrt{P(\boldsymbol{\sigma})}\,|\boldsymbol{\sigma}\rangle\,.$$

$$|\Psi\rangle = \sum_{\boldsymbol{\sigma}} e^{\mathrm{i}\phi(\boldsymbol{\sigma})}\sqrt{P(\boldsymbol{\sigma})}\,|\boldsymbol{\sigma}\rangle\,.$$

$$\phi(\boldsymbol{\sigma}) = \sum_{n=1}^{N} \phi_n$$

# TRAINING RNNS



# DATA DRIVEN APPROACH

# MAXIMUM LIKELIHOOD ESTIMATION

- In statistics, **maximum likelihood estimation** (**MLE**) is a method of estimating the parameters of a probability distribution by maximizing a likelihood function, so that under the assumed statistical model the observed data is most probable.
- Makes sense because if we observe an event it is typically a probable event.
- Distribution: $P_{\boldsymbol{\theta}}(\boldsymbol{\sigma})$ given by an RNN with parameters $\theta$. Loss function—likelihood, to be maximized.

https://en.wikipedia.org/wiki/Maximum_likelihood_estimation

# MAXIMUM LIKELIHOOD ESTIMATION

- Given a dataset of i.i.d. observations $\{\boldsymbol{\sigma}^{(i)}\}_{i=1}^{N_s}$
- Likelihood function of the model is: $\mathscr{L}(\boldsymbol{\theta}) = \prod_{i=1}^{N_s} P_{\boldsymbol{\theta}}(\boldsymbol{\sigma}^{(i)})$
- The MLE principle tells us that we should maximize $\mathscr{L}(\boldsymbol{\theta})$
- $\boldsymbol{\theta}_{\text{MLE}} = \arg\max_{\boldsymbol{\theta}} \mathscr{L}(\boldsymbol{\theta})$
- In practice we minimize the negative log-likelihood $\text{NLL} = -\log \mathscr{L}(\boldsymbol{\theta})$
- We use gradient-descent techniques to do this minimization, i.e. $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - \gamma \nabla_{\boldsymbol{\theta}} \text{NNL}$. We have to do backpropagation through the RNN.

# OTHER DATA DRIVEN APPROACHES FOR RNNS

- Adversarial training
- Moment matching
- Variational inference
- Maximum mean discrepancy
- And more…

# “PHYSICAL LAW” OR HAMILTONIAN DRIVEN APPROACH

# Classical stat. mech.

- We train the model $P_{\boldsymbol{\theta}}(\boldsymbol{\sigma})$ so that it mimics the Boltzmann distribution of a system described by a Hamiltonian H.

- Use variational principle and optimize model's free energy $F_{\boldsymbol{\theta}} = \langle H \rangle_{\boldsymbol{\theta}} - T\,S(P_{\boldsymbol{\theta}}) \geq F$

- $F$ is the true free energy of the system at temperature T.

- $S(P_{\boldsymbol{\theta}}) = -\sum_{\boldsymbol{\sigma}} P_{\boldsymbol{\theta}}(\boldsymbol{\sigma})\, \log P_{\boldsymbol{\theta}}(\boldsymbol{\sigma})$ is the entropy of the model $P_{\boldsymbol{\theta}}(\boldsymbol{\sigma})$

Mohamed Hibat-Allah, Estelle M. Inack, Roeland Wiersema, Roger G. Melko, Juan Carrasquilla. Variational neural annealing. arXiv:2101.10154.
Dian Wu, Lei Wang, Pan Zhang. Solving Statistical Mechanics Using Variational Autoregressive Networks. Phys. Rev. Lett. 122, 080602 (2019)

# Classical stat. mech.

- The free energy can be estimated from samples:

$$F_{\boldsymbol{\theta}}(T) \approx \frac{1}{N_s} \sum_{i=1}^{N_s} F_{\mathrm{loc}}(\boldsymbol{\sigma}^{(i)}), \text{ where}$$

$$F_{\mathrm{loc}}(\boldsymbol{\sigma}) = H_{\mathrm{target}}(\boldsymbol{\sigma}) + T \log \left( P_{\boldsymbol{\theta}}(\boldsymbol{\sigma}) \right)$$

- Requires access to samples $\boldsymbol{\sigma}^{(i)} \sim P_{\boldsymbol{\theta}}$ and fast access to $P_{\boldsymbol{\theta}}(\boldsymbol{\sigma})$ (without expensive calculations of the partition function Z).

Mohamed Hibat-Allah, Estelle M. Inack, Roeland Wiersema, Roger G. Melko, Juan Carrasquilla. Variational neural annealing. arXiv:2101.10154.
Dian Wu, Lei Wang, Pan Zhang. Solving Statistical Mechanics Using Variational Autoregressive Networks. Phys. Rev. Lett. 122, 080602 (2019)

# Classical stat. mech.

- Likewise, the gradients of $F_{\boldsymbol{\theta}}(T)$ are given by

$$\partial_{\boldsymbol{\theta}} F_{\boldsymbol{\theta}}(T) \approx \frac{1}{N_s} \sum_{i=1}^{N_s} \partial_{\boldsymbol{\theta}} \log \left( P_{\boldsymbol{\theta}} \left( \boldsymbol{\sigma}^{(i)} \right) \right) \times \left( F_{\mathrm{loc}}(\boldsymbol{\sigma}^{(i)}) - F_{\boldsymbol{\theta}}(T) \right)$$

- $\partial_{\boldsymbol{\theta}} \log \left( P_{\boldsymbol{\theta}} \left( \boldsymbol{\sigma}^{(i)} \right) \right)$ backprop through time/ automatic diff.

- Simplest parameter update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \partial_{\boldsymbol{\theta}} F_{\boldsymbol{\theta}}$ until convergence

- Note that there is no data involved.

Mohamed Hibat-Allah, Estelle M. Inack, Roeland Wiersema, Roger G. Melko, Juan Carrasquilla. Variational neural annealing. arXiv:2101.10154.
Dian Wu, Lei Wang, Pan Zhang. Solving Statistical Mechanics Using Variational Autoregressive Networks. Phys. Rev. Lett. 122, 080602 (2019)

# Quantum systems

- We can extended this idea to ground states of **quantum many-body systems**

- Promote RNN to a quantum state: $P_{\boldsymbol{\theta}}(\boldsymbol{\sigma}) \rightarrow \Psi_{\boldsymbol{\theta}}(\boldsymbol{\sigma})$

- Modify the objective function:

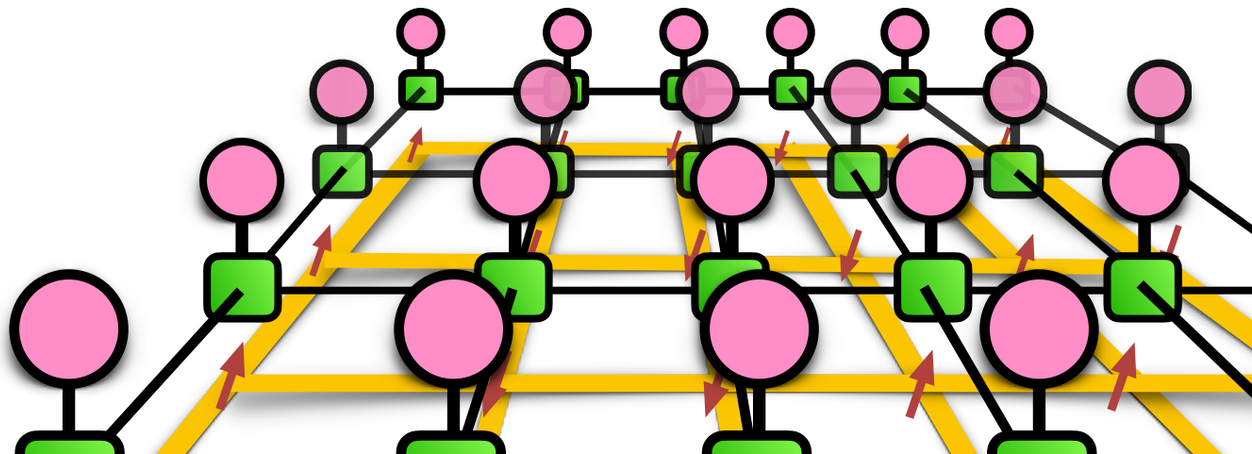
- $E_{\boldsymbol{\theta}} = \langle \Psi_{\boldsymbol{\theta}} | \hat{H} | \Psi_{\boldsymbol{\theta}} \rangle$, $\hat{H}$ is a Hamiltonian whose ground state we want to approximate.

- Both $E_{\boldsymbol{\theta}}$ and its gradients available through sampling.

# Quantum systems

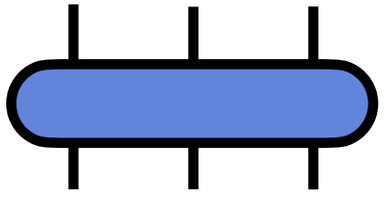
- Both $E_{\boldsymbol{\theta}}$ and its gradients available through sampling.
- $$E = \langle \Psi_\theta | \hat{H} | \Psi_\theta \rangle = \sum_{\boldsymbol{\sigma}} |\psi_\theta(\boldsymbol{\sigma})|^2 \sum_{\boldsymbol{\sigma'}} H_{\boldsymbol{\sigma}\boldsymbol{\sigma'}} \frac{\psi_\theta(\boldsymbol{\sigma'})}{\psi_\theta(\boldsymbol{\sigma})}$$
- $$\equiv \sum_{\boldsymbol{\sigma}} |\psi_\theta(\boldsymbol{\sigma})|^2 E_{loc}(\boldsymbol{\sigma}) \approx \frac{1}{N_S} \sum_{\boldsymbol{\sigma} \sim |\psi_\theta(\boldsymbol{\sigma})|^2} E_{loc}(\boldsymbol{\sigma})$$
- Gradients
- $$\partial_{\theta_j} E = \sum_{\boldsymbol{\sigma}} |\psi_\theta(\boldsymbol{\sigma})|^2 \frac{\partial_{\theta_j} \psi^*_\theta(\boldsymbol{\sigma})}{\psi^*_\theta(\boldsymbol{\sigma})} E_{loc}(\boldsymbol{\sigma}) + \text{c.c}$$
- $$\partial_{\theta_j} E \approx \frac{2}{N_S} \mathfrak{Re} \left( \sum_{i=1}^{N_S} \frac{\partial_{\theta_j} \psi^*_\theta(\boldsymbol{\sigma^{(i)}})}{\psi^*_\theta(\boldsymbol{\sigma^{(i)}})} E_{loc}(\boldsymbol{\sigma^{(i)}}) \right)$$

# LEARNING QUANTUM STATES WITH RNNS

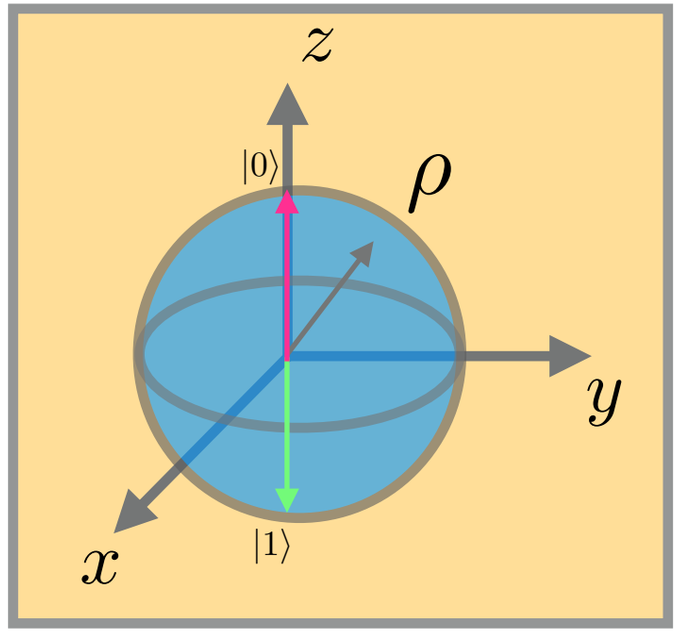


# HOW IS A QUANTUM STATE TRADITIONALLY DESCRIBED?

- A **density matrix** describes the statistical state of a system in quantum mechanics. Everything we can possibly know about a quantum system is encoded in the density matrix.
- A quantum state is a positive semidefinite, Hermitian operator of trace 1 acting on the state space.
- The family of quantum states forms a convex set. For one qubit: Bloch sphere.



$\rho$



# ALTERNATIVE: QUANTUM MECHANICS WITH PROBABILITY

# MEASUREMENTS: POSITIVE OPERATOR-VALUED MEASURE (POVM)

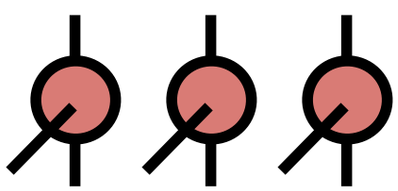
- POVM elements $\mathbf{M} = \{M^{(a)} \mid a \in \{1, ..., m\}\}$
- Positive semidefinite operators $\sum_i M^{(a)} = \mathbb{1}$
- Born Rule $P(\mathbf{a}) = \mathrm{Tr}\, \rho M^{\mathbf{a}}$ quantum theory ⟷ experiment

# INFORMATIONALLY COMPLETE POVM

- The measurement statistics $P(\mathbf{a})$ contains all of the information about the state.
- Relation between $\rho$ and distribution $P(\mathbf{a})$ can be inverted.

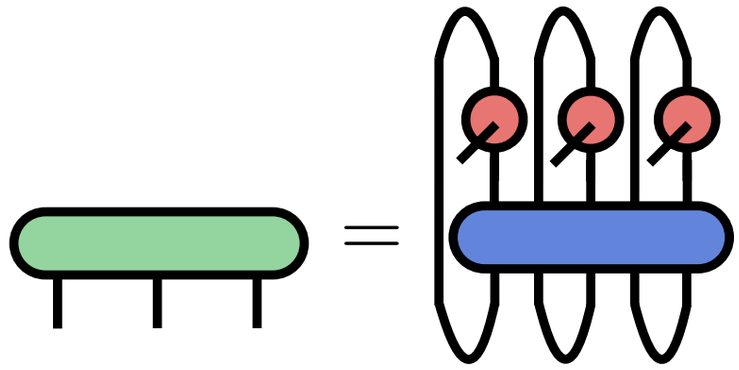
# TAKE A SINGLE QUBIT POVM AND MAKE A TENSOR PRODUCT

$$M = \{M^{(a_1)} \otimes M^{(a_2)} \otimes \ldots M^{(a_N)}\}_{a_1, \ldots a_N}$$



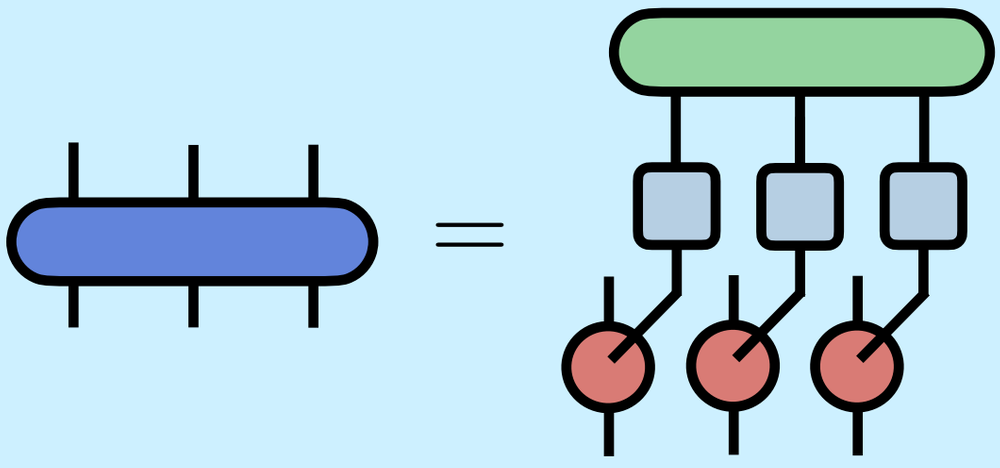
# GRAPHICAL NOTATION AND INVERSE

**BORN RULE** $P(\mathbf{a}) = \mathrm{Tr}\,\rho M^{\mathbf{a}}$



**INFORMATIONALLY COMPLETENESS ——>THIS RELATION CAN BE INVERTED**

$$\rho = \sum_{a,a'} T^{-1}_{a,a'} P(a') M^{(a)}$$



- Unitary evolution
- Schrodinger equation
- Linblad equation
- Measurements

$$i\frac{\partial \rho}{\partial t} = [\mathcal{H}, \rho] \quad \text{BORN RULE} \quad i\frac{\partial P(\boldsymbol{a}'', t)}{\partial t} = \sum_{\boldsymbol{a},\boldsymbol{a}'} \mathrm{Tr}\left(\left[\mathcal{H}, M^{(\boldsymbol{a})}\right] M^{(\boldsymbol{a}'')}\right) T^{-1}_{\boldsymbol{a},\boldsymbol{a}'} P(\boldsymbol{a}', t)$$

# INSIGHT: PARAMETRIZE STATISTICS OF MEASUREMENTS AND INVERT

$$P(\mathbf{a}) = \mathrm{Tr}\,\rho M^{\mathbf{a}}$$

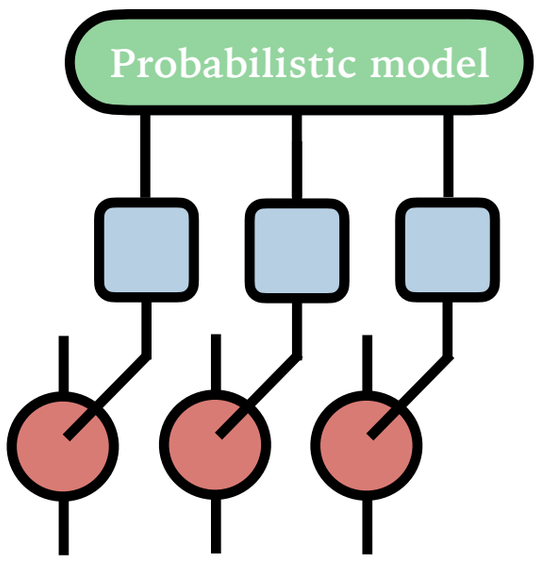
=> Create an autoregressive model of P(a)

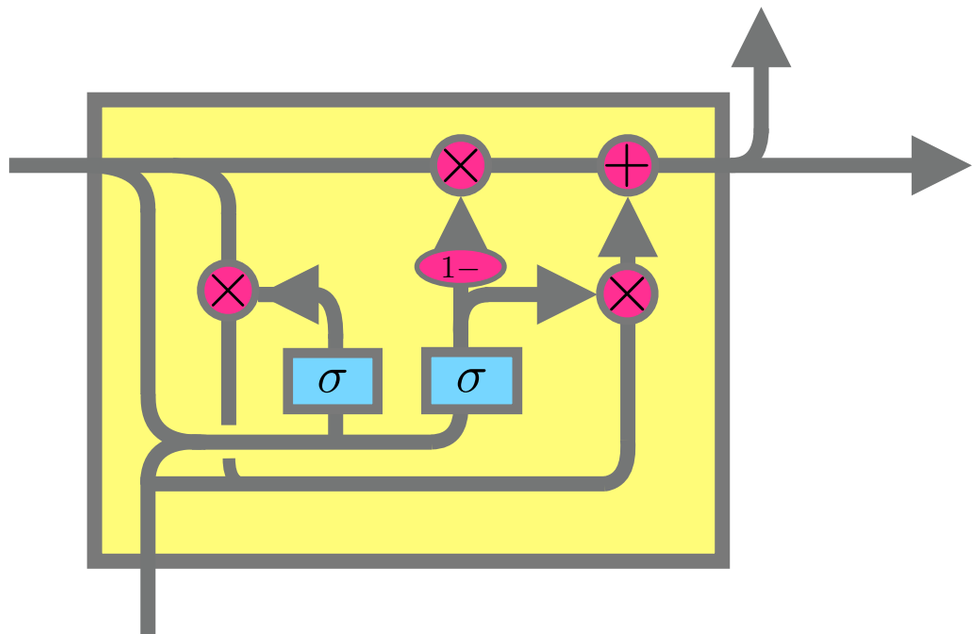
Autoregressive models (RNNs and transformer)

1. Allow for exact sampling
2. Tractable density $P_{\mathrm{model}}(\mathbf{a})$
3. Use MLE to learn it.

$P_{\mathrm{model}}(\mathbf{a})$ ⟶

$\rho_{\mathrm{model}} =$



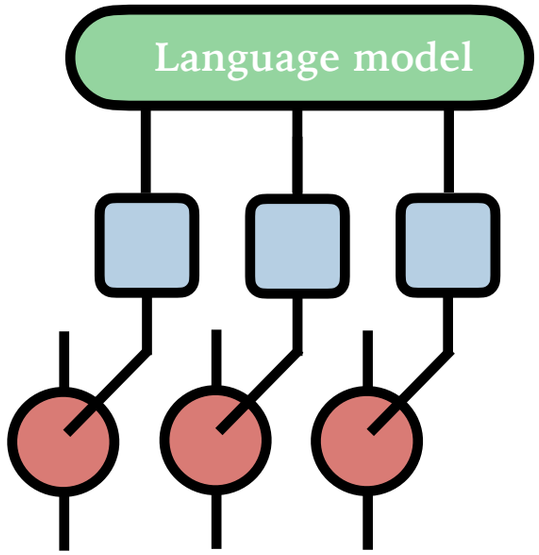
# EXAMPLE: LEARN A QUANTUM STATE FROM MEASUREMENTS

# NEED TO GO BEYOND STANDARD QUANTUM STATE TOMOGRAPHY

- Prepare an unknown quantum state
- Apply a measurement that probes enough information about the quantum state
- Repeat and collect the statistics of the measurement
- Infer a reconstruction of the state consistent with the measurement outcomes

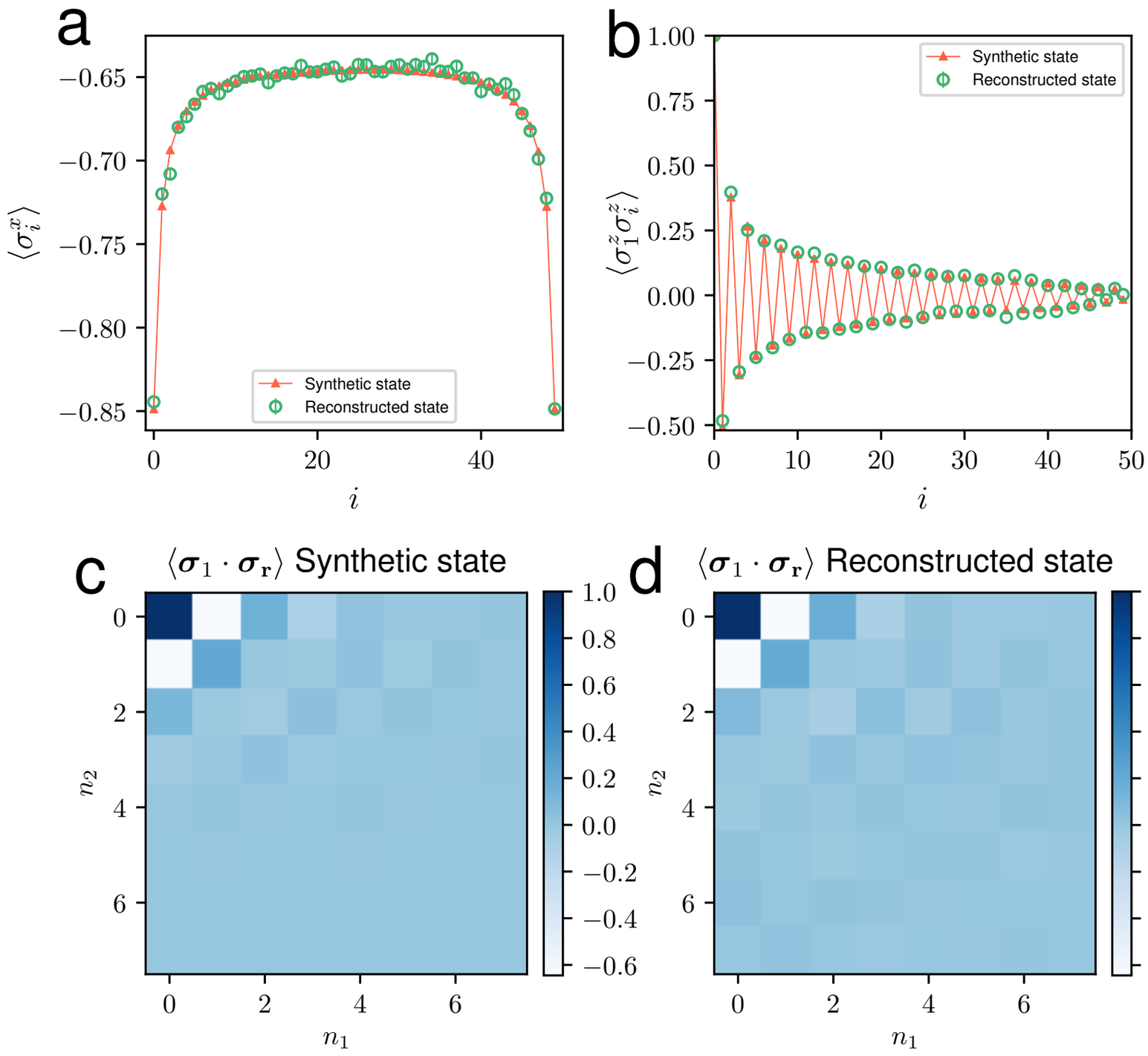


$\rho^* =$




Carrasquilla, Torlai, Melko, Aolita. Nature Machine Intelligence 1, 200 (2019)

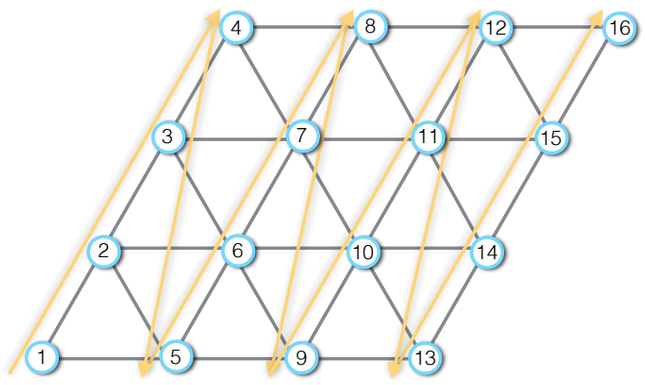
# LEARNING GROUND STATES OF LOCAL HAMILTONIANS FROM DATA (RNN)




$$\mathcal{H} = J \sum_{ij} \sigma_i^z \sigma_j^z + h \sum_i \sigma_i^x$$

N=50 spins. P(a) is a deep (3 layer GRU) recurrent neural network language model.

$$H = J \sum_{i,j} \boldsymbol{\sigma}_i . \boldsymbol{\sigma}_j$$

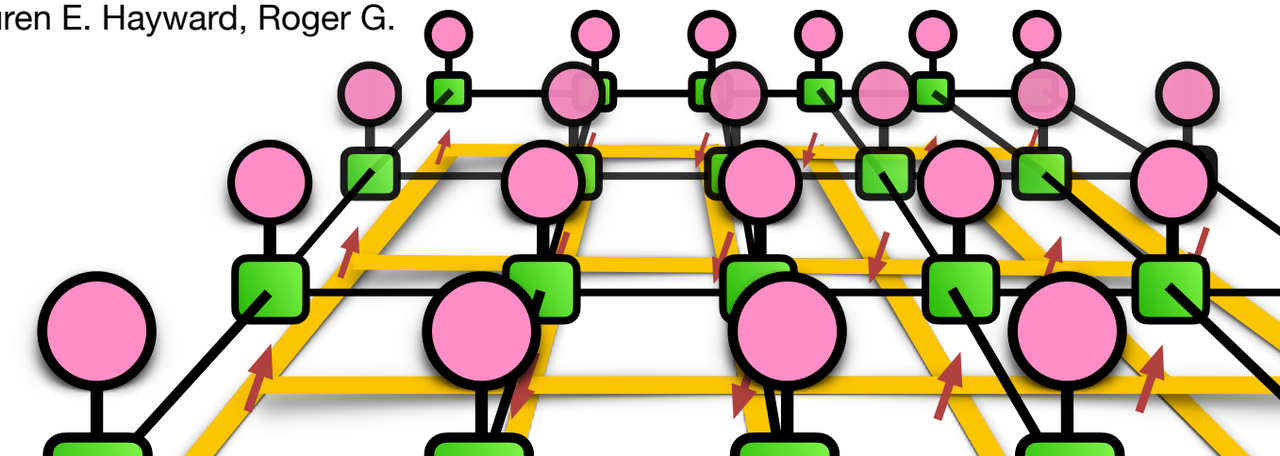



Carrasquilla, Torlai, Melko, Aolita. Nature Machine Intelligence 1, 200 (2019)

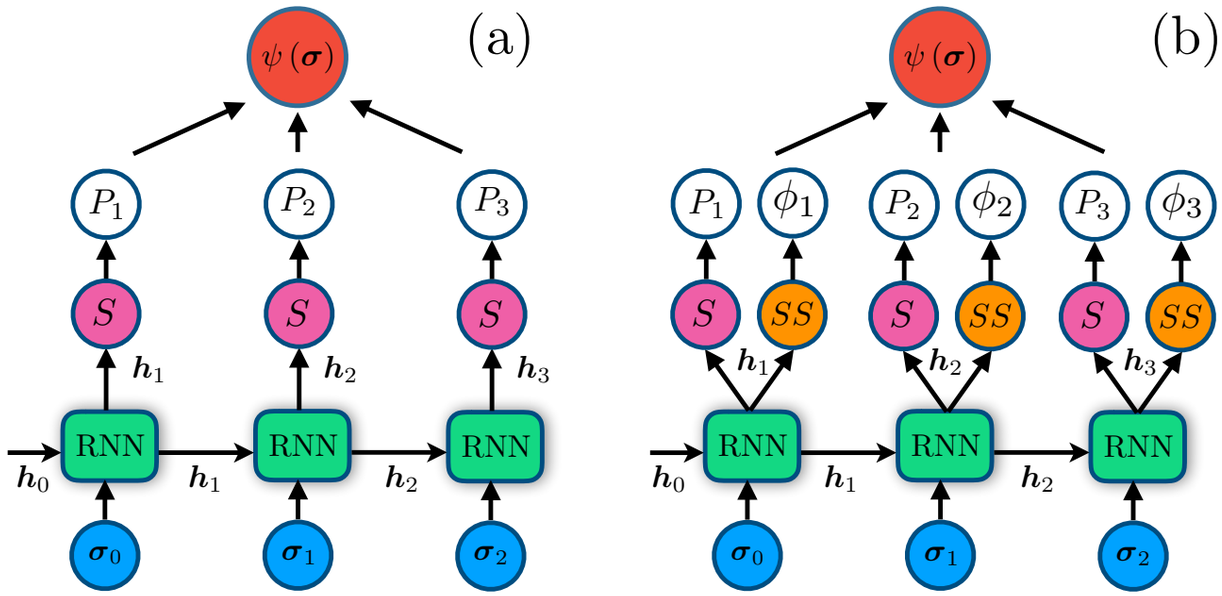
# RECURRENT NEURAL NETWORK WAVEFUNCTIONS

Mohamed Hibat-Allah, Martin Ganahl, Lauren E. Hayward, Roger G. Melko, and Juan Carrasquilla

Phys. Rev. Research **2**, 023358 (2020)

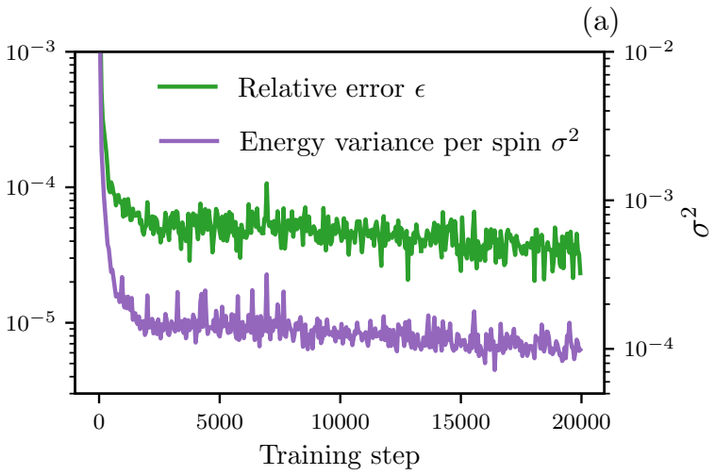


# RECURRENT NEURAL NETWORK WAVEFUNCTIONS

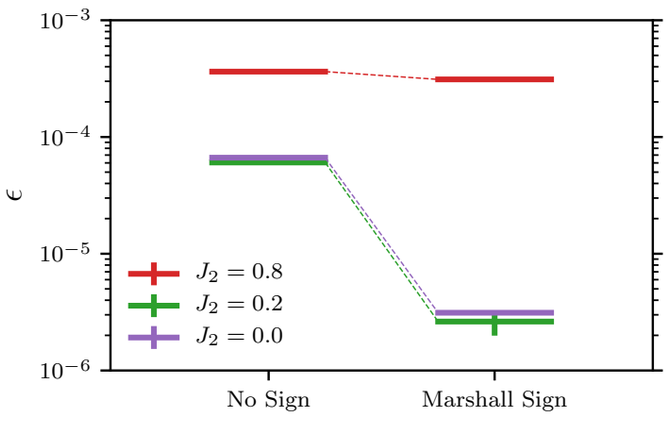



$$|\Psi\rangle = \sum_{\boldsymbol{\sigma}} \psi(\boldsymbol{\sigma})\,|\boldsymbol{\sigma}\rangle = \sum_{\boldsymbol{\sigma}} \sqrt{P(\boldsymbol{\sigma})}\,|\boldsymbol{\sigma}\rangle$$

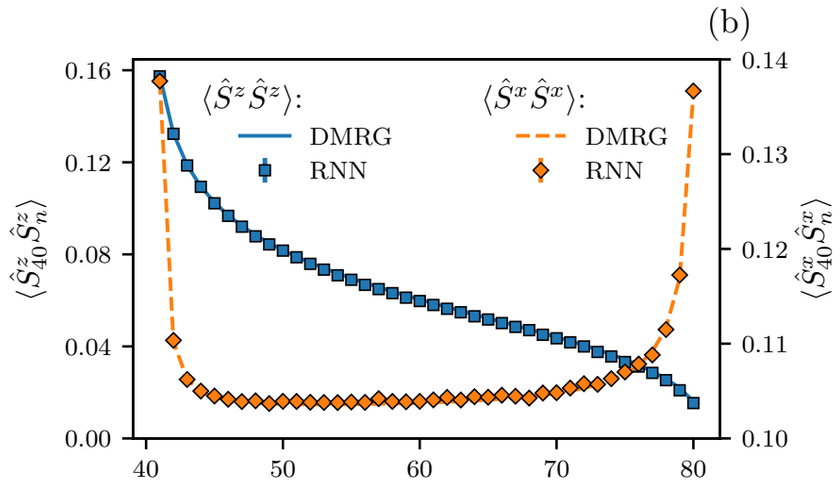
$$|\Psi\rangle = \sum_{\boldsymbol{\sigma}} \exp(\mathrm{i}\phi(\boldsymbol{\sigma}))\sqrt{P(\boldsymbol{\sigma})}\,|\boldsymbol{\sigma}\rangle$$



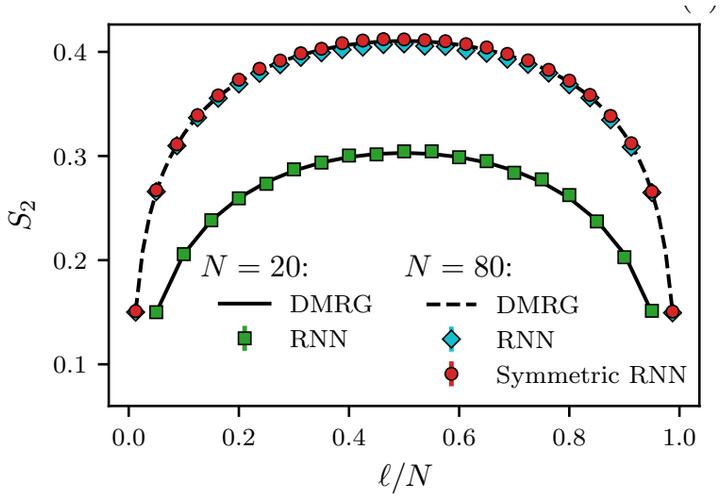

TFIM 1-D N=1000




J1-J2 model in 1-D




TFIM 1-D N=80

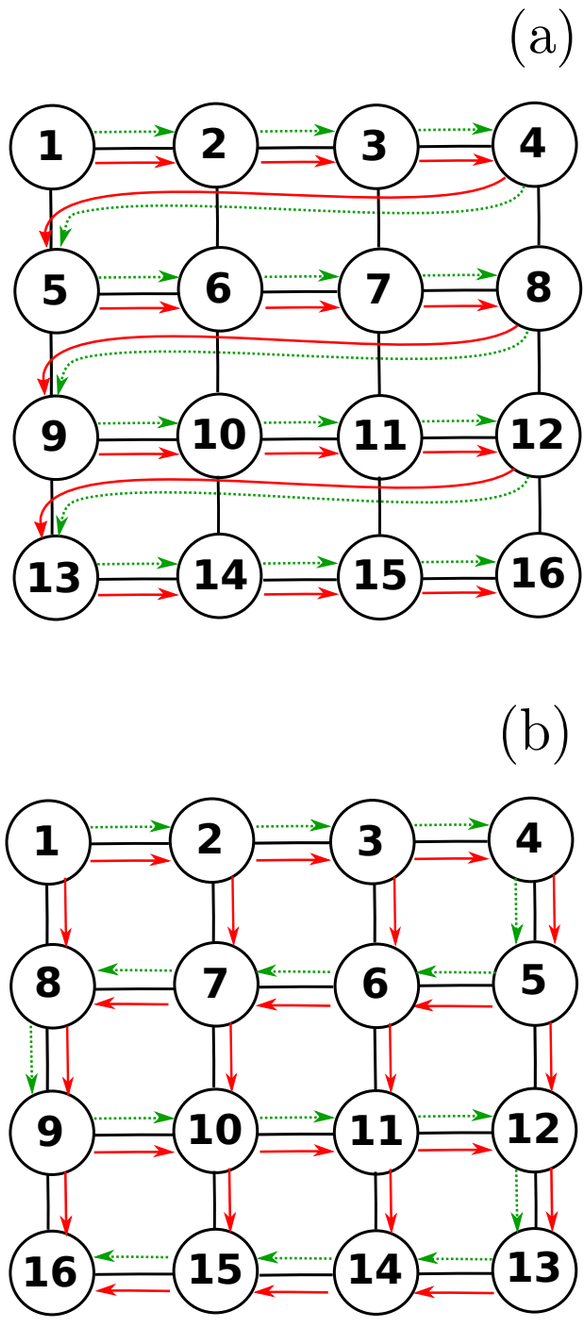


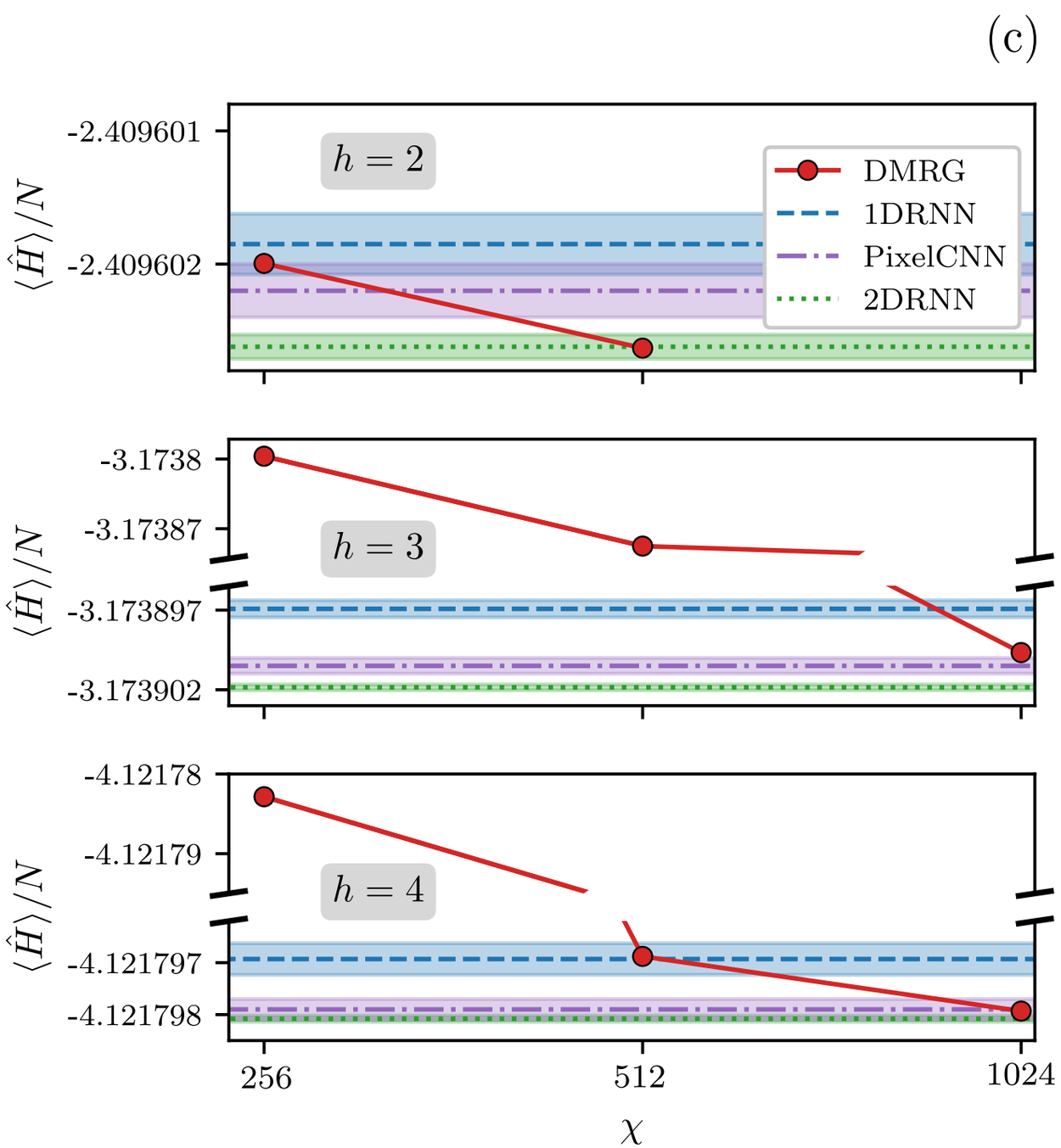

TFIM 1-D N=80

Symmetries: Spin inversion, mirror reflection, Sz. Sign: different Marshall signs for the J1-J2 model

Mohamed Hibat-Allah, Martin Ganahl, Lauren E. Hayward, Roger G. Melko, and Juan Carrasquilla Phys. Rev. Research **2**, 023358 (2020)

# RECURRENT NEURAL NETWORK WAVEFUNCTIONS IN 2 DIMENSIONS




Mohamed Hibat-Allah, Martin Ganahl, Lauren E. Hayward, Roger G. Melko, and Juan Carrasquilla. Phys. Rev. Research **2**, 023358 (2020)

# Combinatorial optimization

- Many important challenges in science and technology can be cast as optimization problems.

- Areas include artificial intelligence, machine learning, auction theory, software engineering, applied mathematics and theoretical computer science

- Traveling salesman problem, nurse scheduling problem, Vehicle routing problem, factoring, chip placement, etc

# Combinatorial optimization by simulated annealing

- Deep connection between materials science and optimization. S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, “Optimization by simulated annealing,” Science 220, 671–680 (1983).

- Annealing in materials and metallurgy: a crystalline solid is heated, kept at high temperature for a while, and then slowly cooled down to its lowest energy state— enables **control** of the properties of the material.

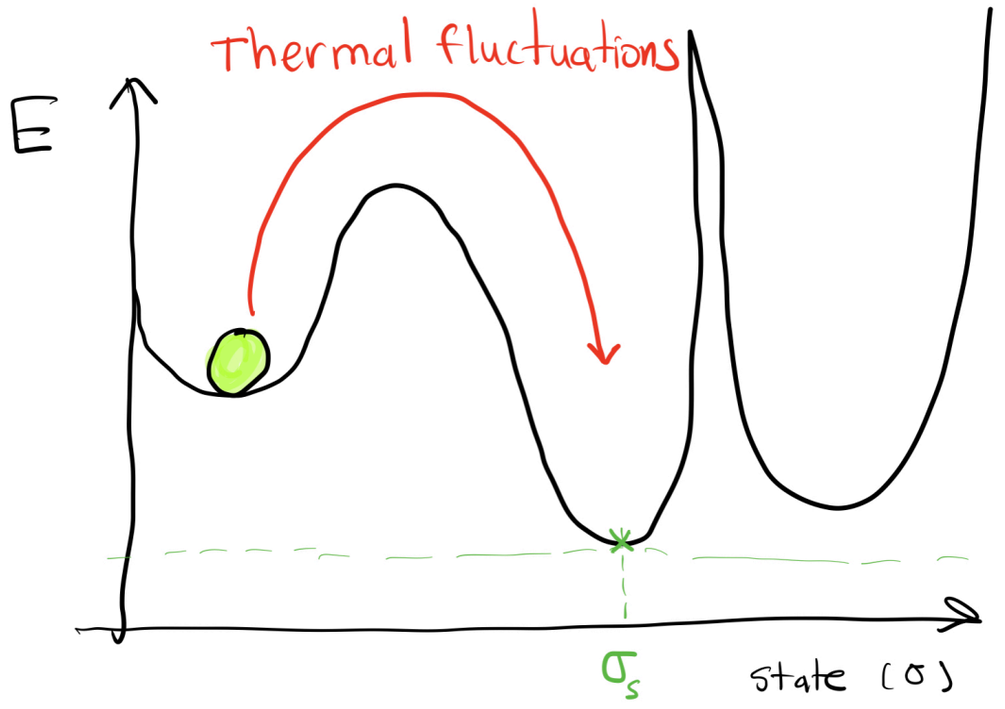
- “We show how the Metropolis algorithm for approximate numerical simulation of many-body system at a finite temperature provides a natural tool for bringing techniques of statistical mechanics to bear on optimization”.

# Combinatorial optimization by simulated annealing

- SA mirrors the analogous annealing process in materials science and metallurgy

- The SA algorithm explores an optimization problem's energy landscape via a gradual decrease in thermal fluctuations generated by the Metropolis-Hastings algorithm.

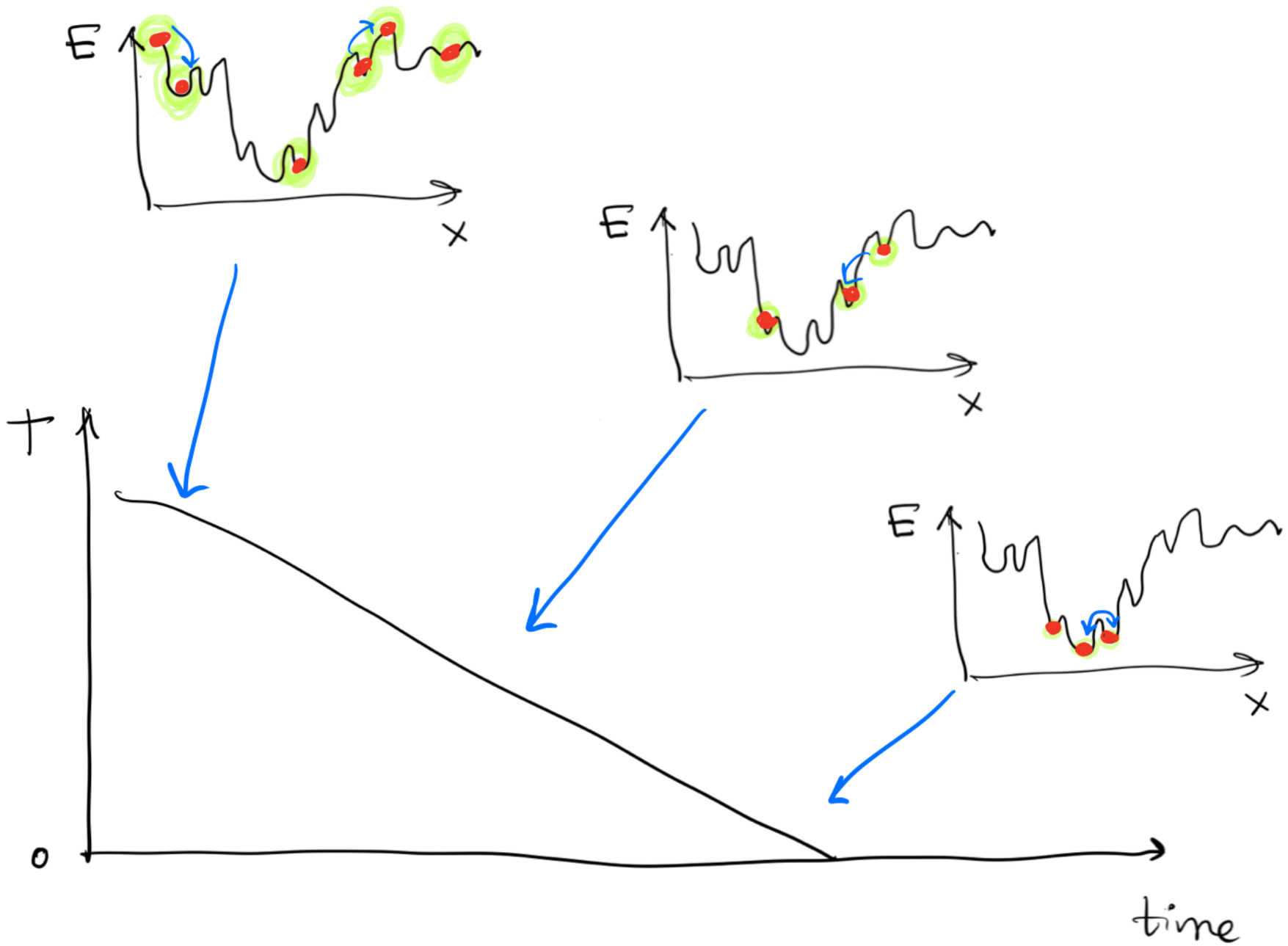
- Temperature is reduced slowly according to some user-defined schedule.



# Simulated annealing

- Real space cartoon

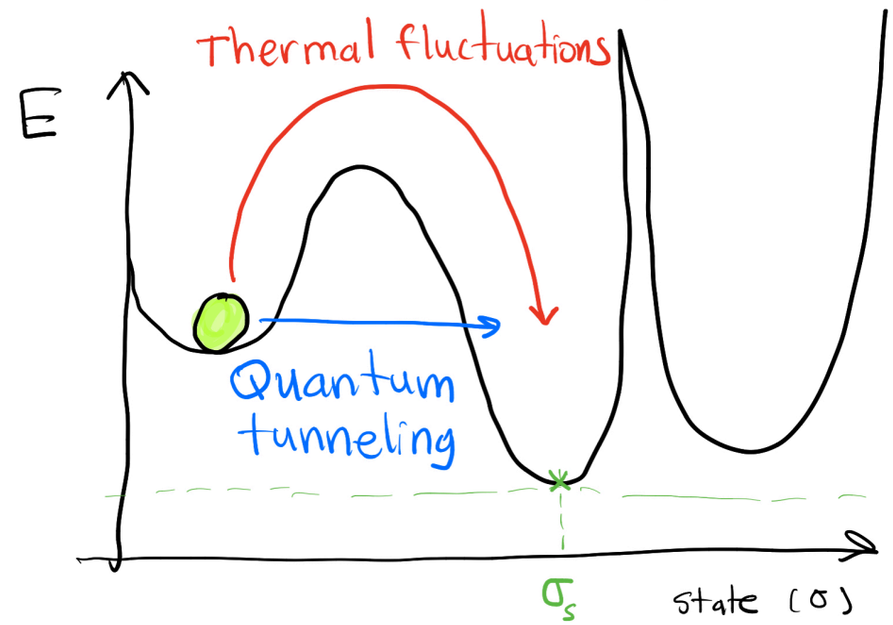
$$x = \downarrow\uparrow\uparrow\downarrow\downarrow\uparrow\uparrow$$

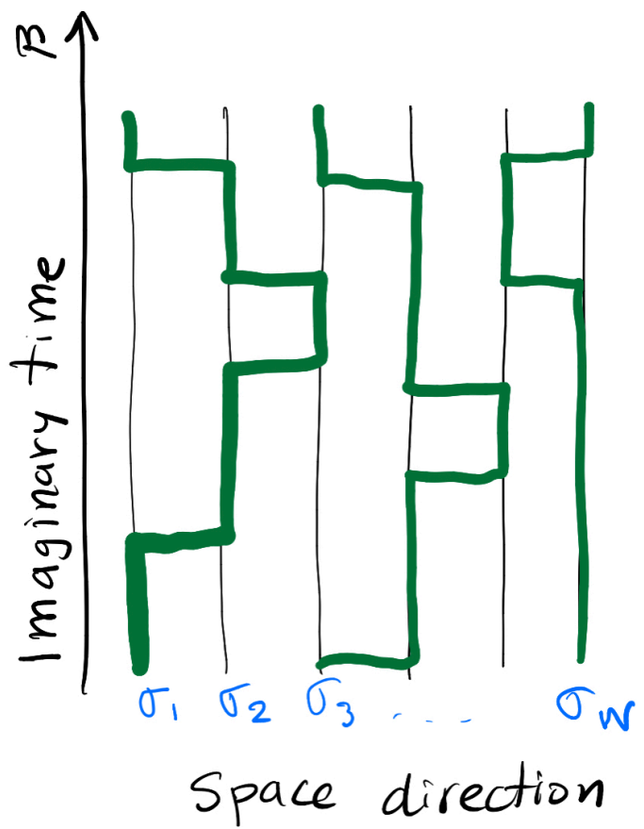


# Combinatorial optimization by quantum annealing

- Quantum annealing: Solve an optimization problem using quantum effects:
  $\hat{H}(t) = \hat{H}_{\text{target}} + f(t)\hat{H}_D$

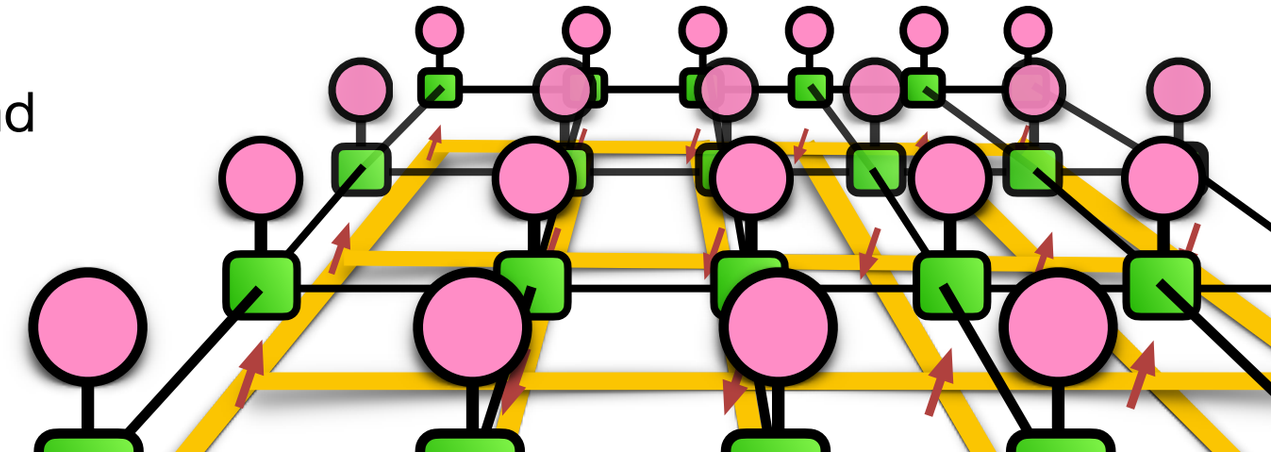
- It is possible to emulate quantum annealing using quantum Monte Carlo—simulated quantum annealing



# CAN WE SIMULATE THESE OPTIMIZATION TECHNIQUES VARIATIONALLY?

Mohamed Hibat-Allah, Estelle M. Inack, Roeland Wiersema, Roger G. Melko, Juan Carrasquilla. Variational neural annealing. arXiv:2101.10154



# Variational classical annealing

- We train a model $P_{\boldsymbol{\theta}}(\boldsymbol{\sigma})$ so that it mimics the annealing of the **Boltzmann distribution**. We model $P_{\boldsymbol{\theta}}(\boldsymbol{\sigma})$ with an RNN

- Use **variational principle** and optimize model's free energy $F_{\boldsymbol{\theta}}(t) = \langle H_{\text{target}} \rangle_{\boldsymbol{\theta}} - T(t)\, S(P_{\boldsymbol{\theta}}) \geq F(t)$ using gradient descent

- $F(t)$ is the true free energy of the system at temperature T(t).

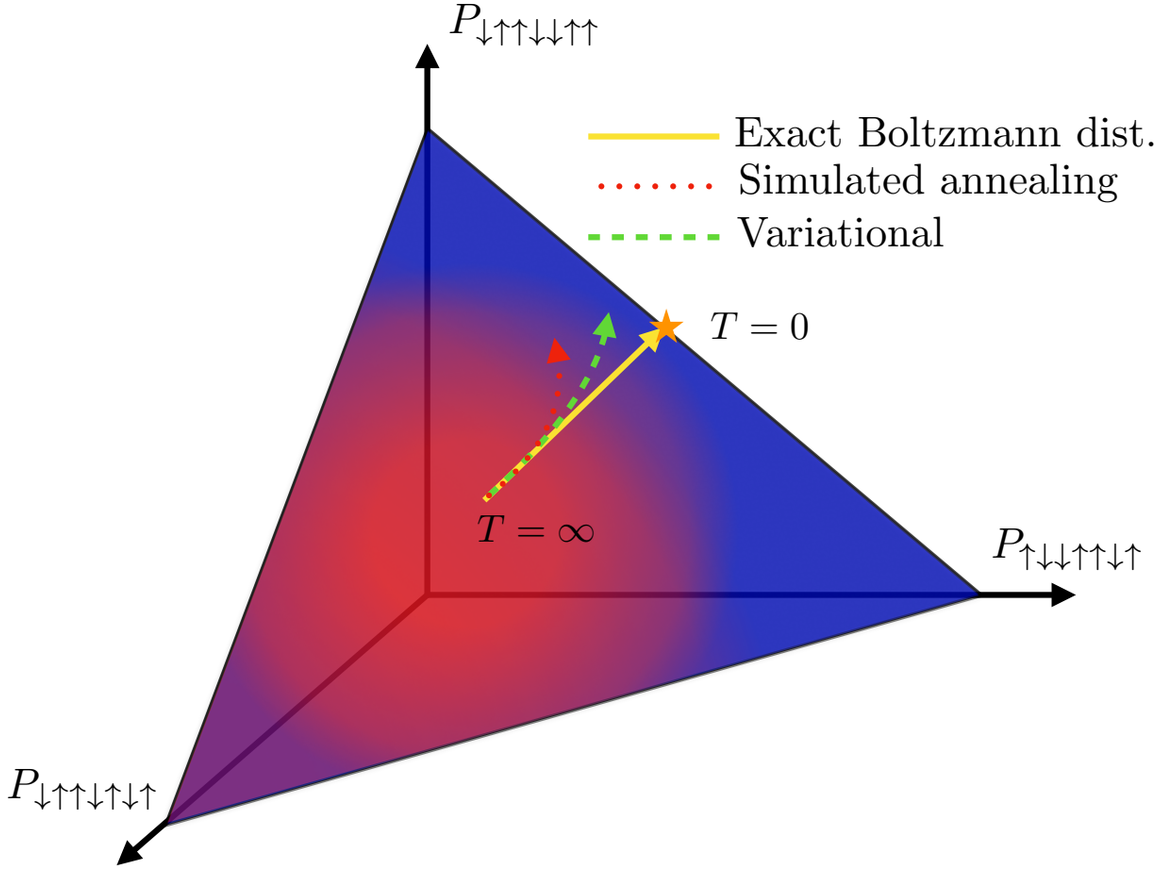
- $S(P_{\boldsymbol{\theta}})$ is the entropy of the model $P_{\boldsymbol{\theta}}(\boldsymbol{\sigma})$

- As in SA, temperature is decreased from an initial value $T_0$ to $0$ using a linear schedule function $T(t) = T_0(1-t)$, where $t \in [0,1]$

Mohamed Hibat-Allah, Estelle M. Inack, Roeland Wiersema, Roger G. Melko, Juan Carrasquilla. Variational neural annealing. arXiv:2101.10154.
Dian Wu, Lei Wang, Pan Zhang. Solving Statistical Mechanics Using Variational Autoregressive Networks. Phys. Rev. Lett. 122, 080602 (2019)

# Variational classical annealing

| | SA | Variational SA |
|---|---|---|
| Distribution | Boltzmann dist. ✓ | Model $P_{\boldsymbol{\theta}}(\boldsymbol{\sigma})$ ✗ |
| Sampling | Markov Chain MC ✗ | Direct sampling ✓ |




We swap wrong sampling of exact distribution with exact sampling the wrong distribution

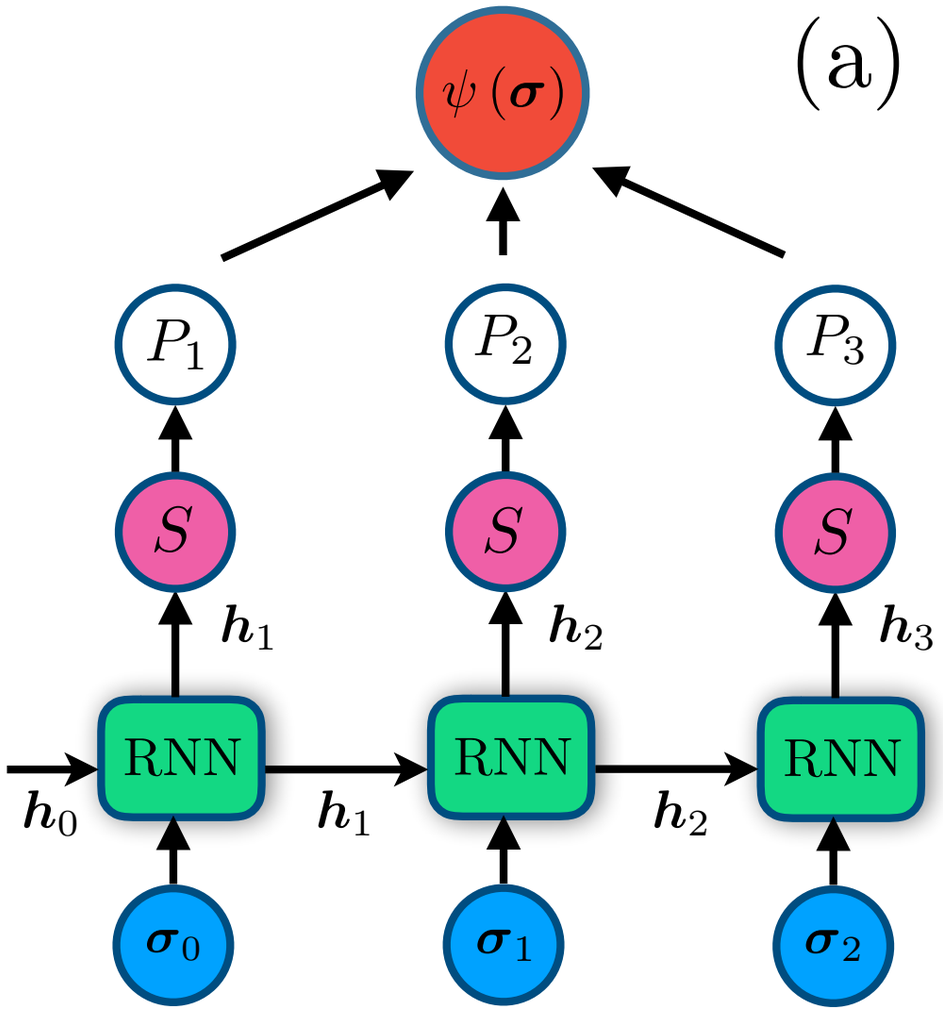
Does this produce anything useful?

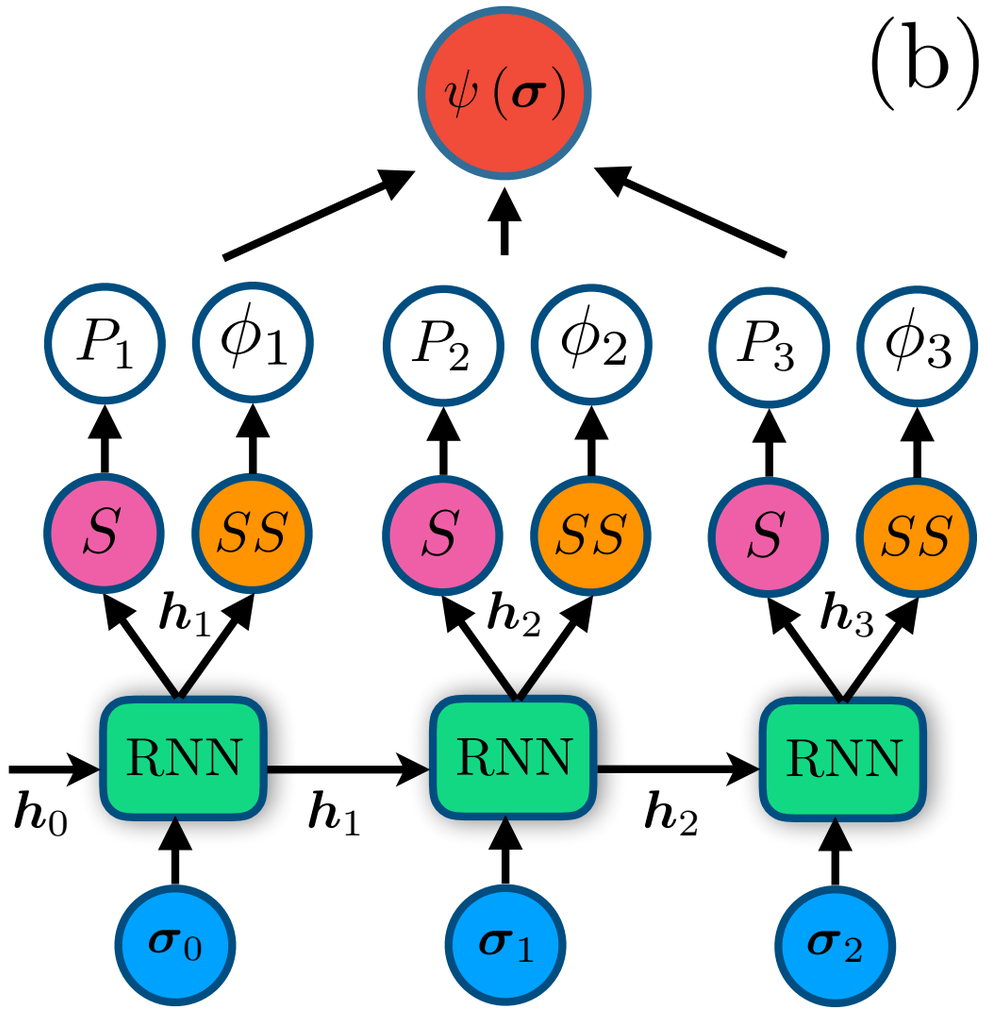
Mohamed Hibat-Allah, Estelle M. Inack, Roeland Wiersema, Roger G. Melko, Juan Carrasquilla. Variational neural annealing. arXiv:2101.10154

# Variational quantum annealing

- We can extended this idea to simulated **quantum** annealing
- Promote RNN to a quantum state: $P_{\boldsymbol{\theta}}(\boldsymbol{\sigma}) \rightarrow \Psi_{\boldsymbol{\theta}}(\boldsymbol{\sigma})$
- Modify the cost function:
- $F_{\boldsymbol{\theta}}(t) = \langle H_{\text{target}} \rangle_{\boldsymbol{\theta}} - T(t)\, S(P_{\boldsymbol{\theta}}) \rightarrow E_{\theta} = \langle \Psi_{\boldsymbol{\theta}} | \hat{H}_{\text{target}} | \Psi_{\boldsymbol{\theta}} \rangle - \Gamma(t) \langle \Psi_{\boldsymbol{\theta}} | \hat{H}_{\text{driver}} | \Psi_{\boldsymbol{\theta}} \rangle$
- $\hat{H}_{\text{driver}} = \sum_i \hat{\sigma}_i^x$ typical choice in quantum annealing
- Optimize $E_{\theta}$ using gradient descent
- Slowly decrease quantum tunnelling $\Gamma(t) = \Gamma_0(1 - t)$, where $t \in [0,1]$

# RNNs wave functions




$$|\Psi\rangle = \sum_{\boldsymbol{\sigma}} \psi(\boldsymbol{\sigma})\,|\boldsymbol{\sigma}\rangle = \sum_{\boldsymbol{\sigma}} \sqrt{P(\boldsymbol{\sigma})}\,|\boldsymbol{\sigma}\rangle\,.$$

$$|\Psi\rangle = \sum_{\boldsymbol{\sigma}} e^{\mathrm{i}\phi(\boldsymbol{\sigma})}\sqrt{P(\boldsymbol{\sigma})}\,|\boldsymbol{\sigma}\rangle\,.$$

# Variational neural annealing

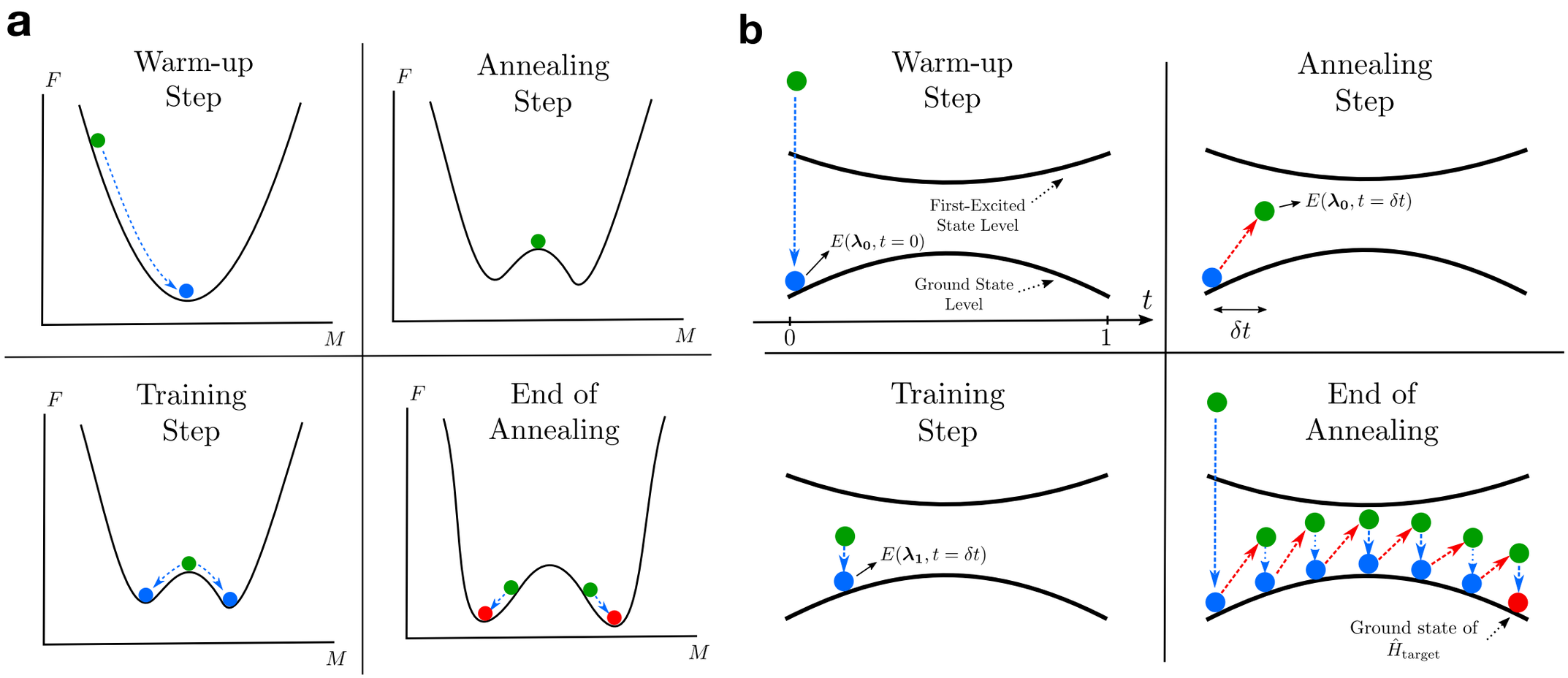



Figure 2. Variational neural annealing protocols. (a) The variational classical annealing (VCA) algorithm steps. A warm-up step brings the initialized variational state (green dot) close to the minimum of the free energy (cyan dot) at a given value of the order parameter $M$. This step is followed by an annealing and a training step that brings the variational state back to the new free energy minimum. Repeating the last two steps until $T(t=1)=0$ (red dots) produces approximate solutions to $H_{\text{target}}$ if the protocol is conducted slowly enough. This schematic illustration corresponds to annealing through a continuous phase transition with an order parameter $M$. (b) Variational quantum annealing (VQA). VQA includes a warm-up step, followed by an annealing and a training step, which brings the variational energy (green dot) closer to the new a ground state energy (cyan dot). We loop over the previous two steps until reaching the target ground state of $\hat{H}_{\text{target}}$ (red dot) if annealing is performed slowly enough.

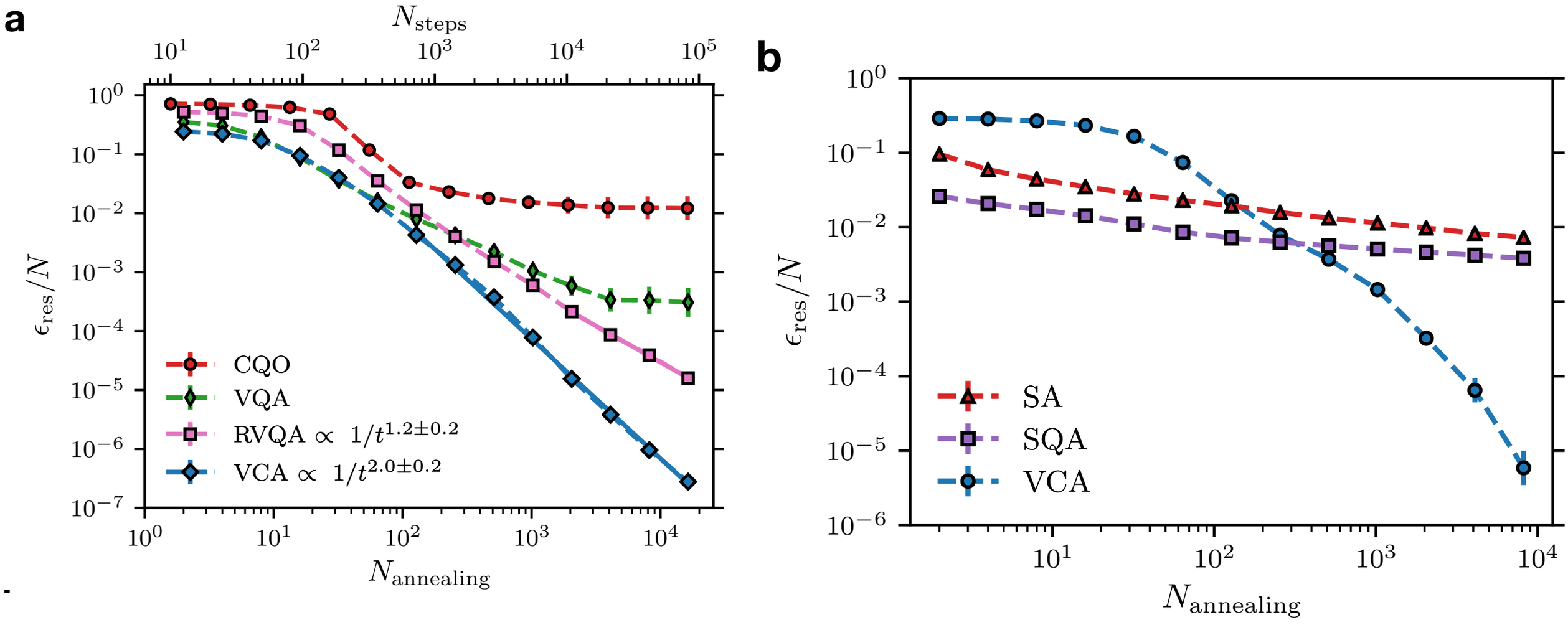
# Edwards-Anderson Model

- $H_{\text{target}} = -\sum_{\langle i,j \rangle} J_{ij}\sigma_i\sigma_j$ in a 2-dimensional square lattice

- $J_{ij} \in [-1,1)$

- We test no annealing (CQO), VCA, VQA and an entropy regularized VQA: $\tilde{F}_{\boldsymbol{\theta}}(t) = \langle \hat{H}(t) \rangle_{\boldsymbol{\theta}} - T(t) S_{\text{pseudo}}(|\Psi_{\boldsymbol{\theta}}|^2)$ where $\hat{H}(t) = \hat{H}_{\text{target}} - \Gamma(t)\sum_i \hat{\sigma}_i^x$

# Edwards-Anderson Model




**Benchmarking the two-dimensional Edwards-Anderson spin glass. (a) A comparison between VCA, VQA, RVQA, and CQO on a $10 \times 10$ lattice by plotting the residual energy per site vs $N_{\text{annealing}}$. For CQO, we report the residual energy per site vs the number of optimization steps $N_{\text{steps}}$. (b) Comparison between SA, SQA with $P = 20$ trotter slices, and VCA on a $40 \times 40$ lattice. The annealing speed is the same for SA, SQA and VCA.**

# Variational annealing on fully connected spin glasses

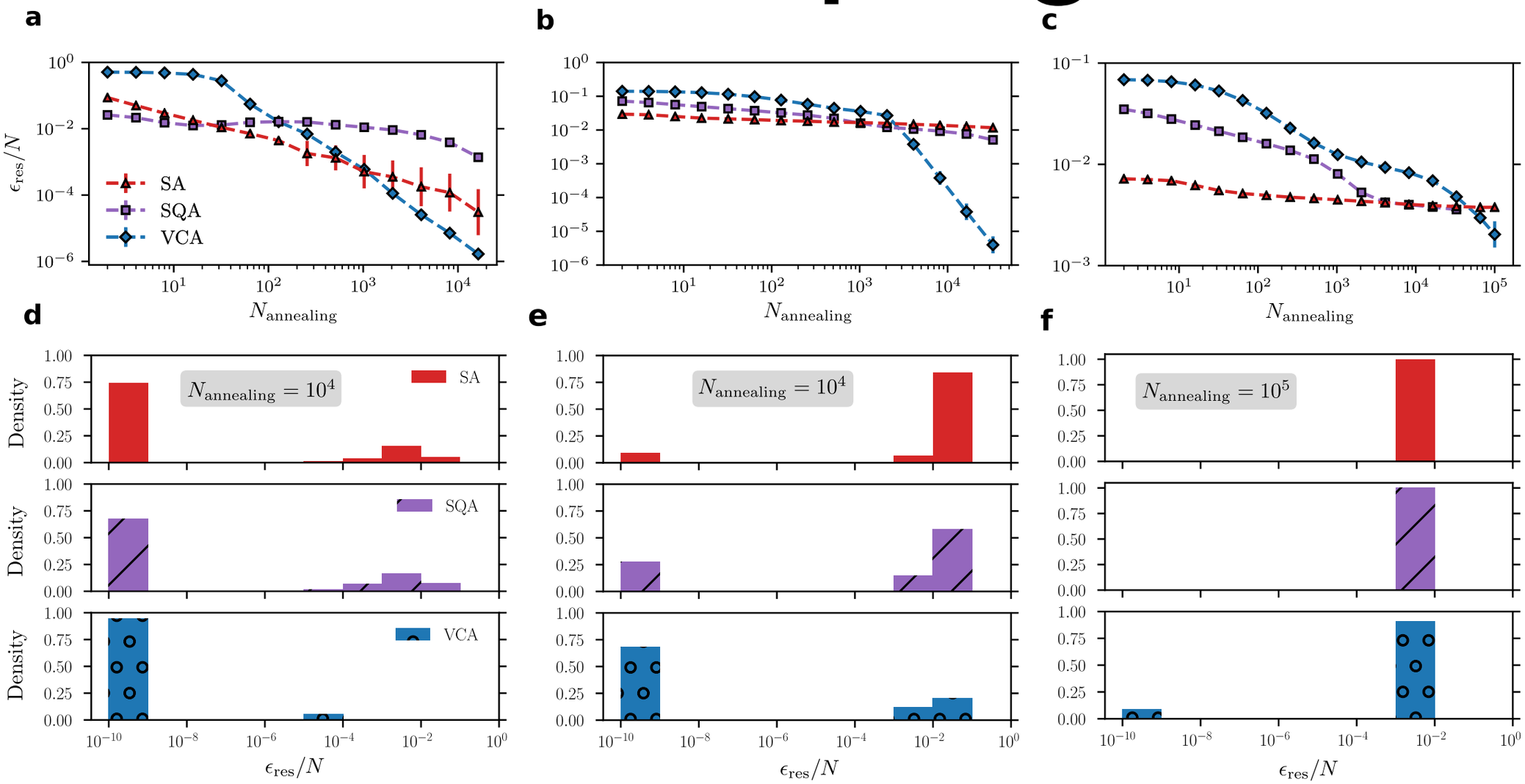


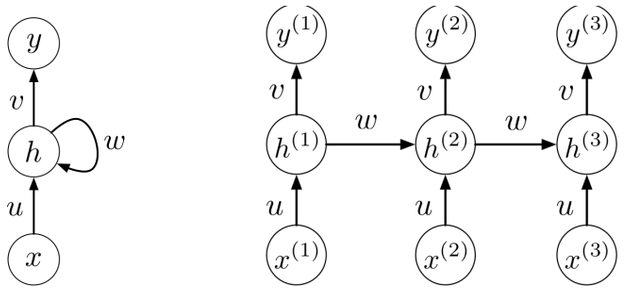

Figure 5. Benchmarking SA, SQA ($P = 100$ trotter slices) and VCA on the Sherrington-Kirkpatrick (SK) model and the Wishart planted ensemble (WPE). Panels (a),(b), and (c) display the residual energy per site as a function of $N_{\rm annealing}$. (a) The SK model with $N = 100$ spins. (b) WPE with $N = 32$ spins and $\alpha = 0.5$. (c) WPE with $N = 32$ spins and $\alpha = 0.25$. Panels (d), (e) and (f) display the residual energy histogram for each of the different techniques and models in panels (a),(b), and (c), respectively. The histograms use 25000 data points for each method. Note that we choose a minimum threshold of $10^{-10}$ for $\epsilon_{\rm res}/N$, which is within our numerical accuracy.

Mohamed Hibat-Allah, Estelle M. Inack, Roeland Wiersema, Roger G. Melko, Juan Carrasquilla. Variational neural annealing. arXiv:2101.10154

# Conclusion

- Introduced RNNs and showed examples of applications in quantum many-body physics. Lots of applications:
- Quantum state reconstruction with RNNs (Nature Machine Intelligence, vol. 1, 155-161 (2019)) and Transformers  (arXiv:2006.12469)
- Simulation of quantum circuits (arXiv:1912.11052)
- Recurrent neural network wavefunctions — extremely accurate ground states, very compact representation in 1d and 2d (Phys. Rev. Research 2, 023358 (2020) )
- Variational neural annealing: produces very accurate solutions to challenging spin glass problems beyond SA and SQA (arXiv:2101.10154)
- Neural Error Mitigation of Near-Term Quantum Simulations (arXiv:2105.08086) (transformer)
- Simulation of open system dynamics (arXiv:2009.05580) (transformer)
- Transfer learning based on physical principles (arXiv:2003.02647) (RNN)

# BACKPROPAGATION THROUGH TIME




It performs the following computations in the forward pass:

$$z^{(t)} = ux^{(t)} + wh^{(t-1)} \tag{3}$$
$$h^{(t)} = \phi(z^{(t)}) \tag{4}$$
$$r^{(t)} = vh^{(t)} \tag{5}$$
$$y^{(t)} = \phi(r^{(t)}). \tag{6}$$

Figure 5 shows the unrolled computation graph. Note the weight sharing. Now we just need to do backprop on this graph, which is hopefully a completely mechanical procedure by now:

$$\overline{\mathcal{L}} = 1 \tag{7}$$
$$\overline{y^{(t)}} = \overline{\mathcal{L}} \frac{\partial \mathcal{L}}{\partial y^{(t)}} \tag{8}$$
$$\overline{r^{(t)}} = \overline{y^{(t)}}\, \phi'(r^{(t)}) \tag{9}$$
$$\overline{h^{(t)}} = \overline{r^{(t)}}\, v + \overline{z^{(t+1)}}\, w \tag{10}$$
$$\overline{z^{(t)}} = \overline{h^{(t)}}\, \phi'(z^{(t)}) \tag{11}$$
$$\overline{u} = \sum_{t=1}^{T} \overline{z^{(t)}}\, x^{(t)} \tag{12}$$
$$\overline{v} = \sum_{t=1}^{T} \overline{r^{(t)}}\, h^{(t)} \tag{13}$$
$$\overline{w} = \sum_{t=1}^{T-1} \overline{z^{(t+1)}}\, h^{(t)} \tag{14}$$

These update rules are basically like the ones for an MLP, except that the weight updates are summed over all time steps.

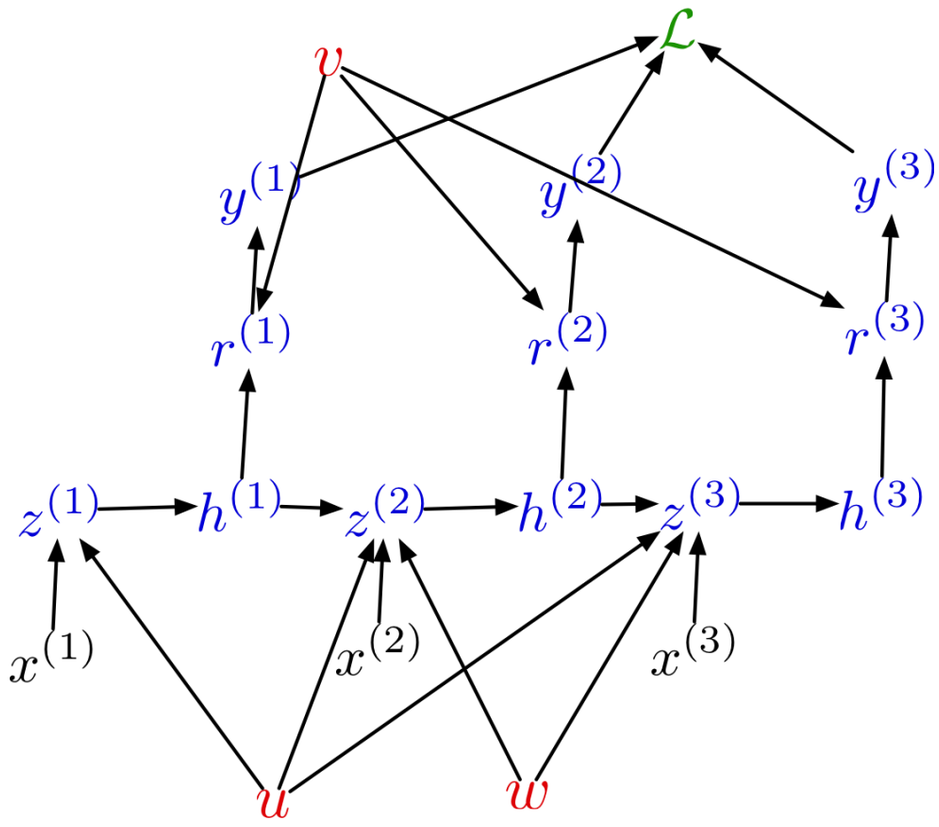



Figure 5: The unrolled computation graph.

See Roger Grosse's CSC421 course

"Neural Networks and Deep Learning" (2019)