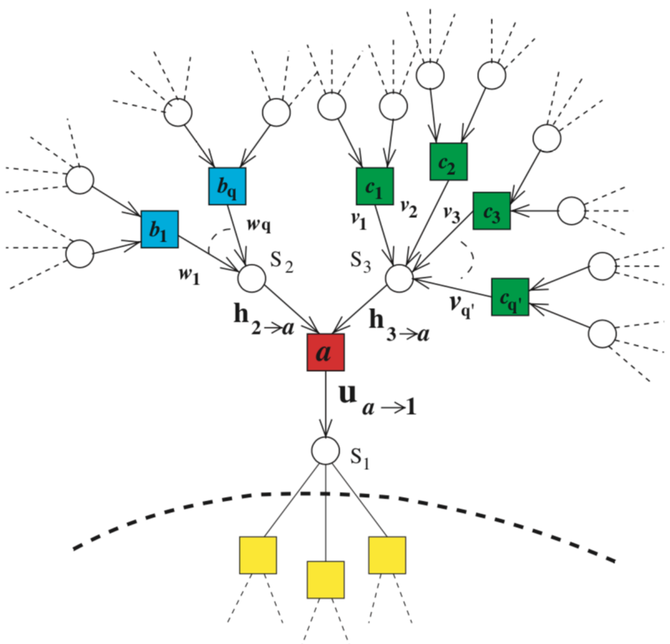
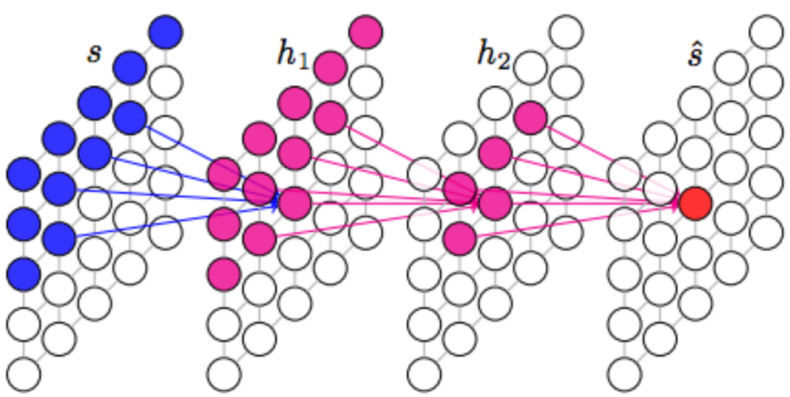
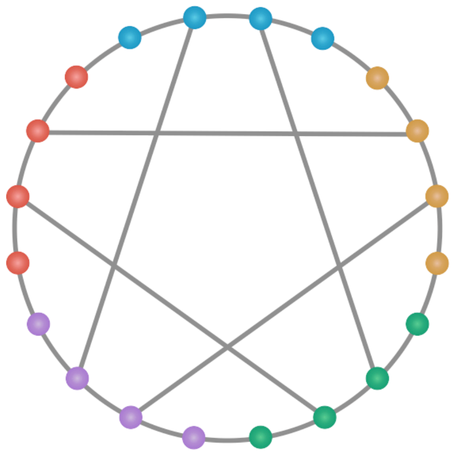
统计力学计算方法：

从 平均场 到 神经网络 再到 张量网络







Mean-field

Neural Networks

Tensor Networks

Pan Zhang
ITP,CAS

华中师大

2021.10.12

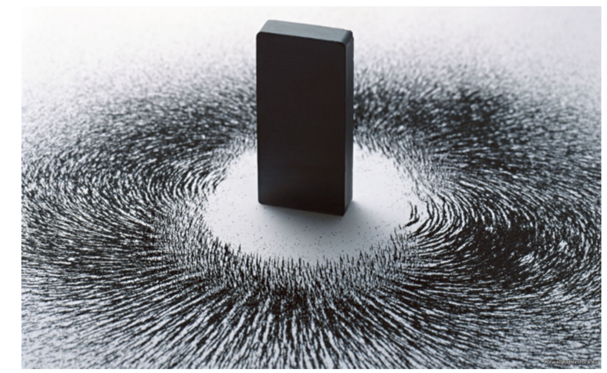
统计物理与复杂系统

微观集体行为，宏观规律涌现：

- 热力学、相变、非平衡
- 量子统计
- 凝聚态物理 ……

物理之外的应用

- 化学、生物科学
- 机器学习、人工智能
- 社会、经济科学 ……



$$P(\mathbf{S}) = \frac{1}{Z} \mathrm{e}^{-\beta E(\mathbf{S})}$$

微观构型分布

$$P(\mathrm{Data})$$

数据变量分布

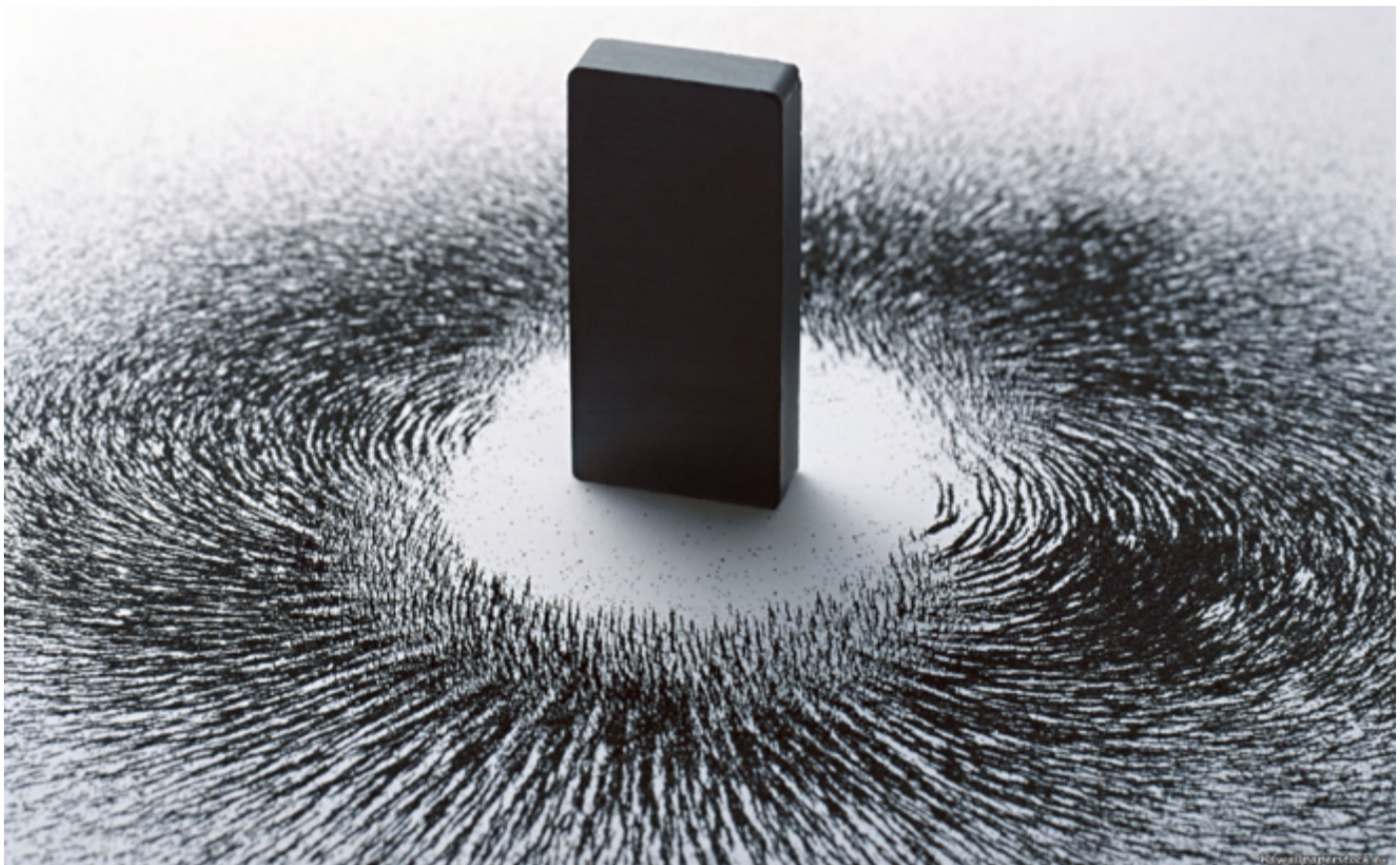
Statistical Mechanics

$$\mathbf{S} = \{+1, -1\}^n$$

$\uparrow \uparrow \uparrow \downarrow \uparrow \downarrow \downarrow \downarrow \downarrow \downarrow \uparrow \uparrow$

$$P(\mathbf{S}) = \frac{1}{Z} \mathrm{e}^{-\beta E(\mathbf{S})}$$

$$Z = \sum_{\mathbf{S}} \mathrm{e}^{-\beta E(\mathbf{S})}$$



Statistical Mechanics

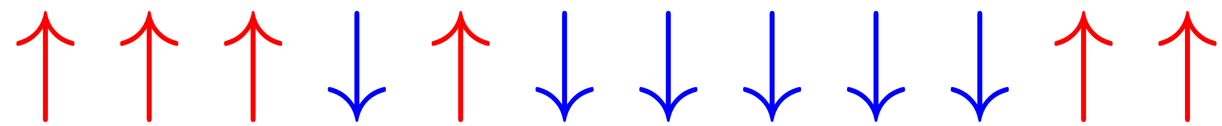
$$\mathbf{S} = \{+1, -1\}^n \qquad \uparrow \uparrow \uparrow \downarrow \uparrow \downarrow \downarrow \downarrow \downarrow \downarrow \uparrow \uparrow$$

$$P(\mathbf{S}) = \frac{1}{Z} \mathrm{e}^{-\beta E(\mathbf{S})} \qquad Z = \sum_{\mathbf{s}} \mathrm{e}^{-\beta E(\mathbf{S})}$$

- Estimating the free energy
- Computing observables
- Sampling

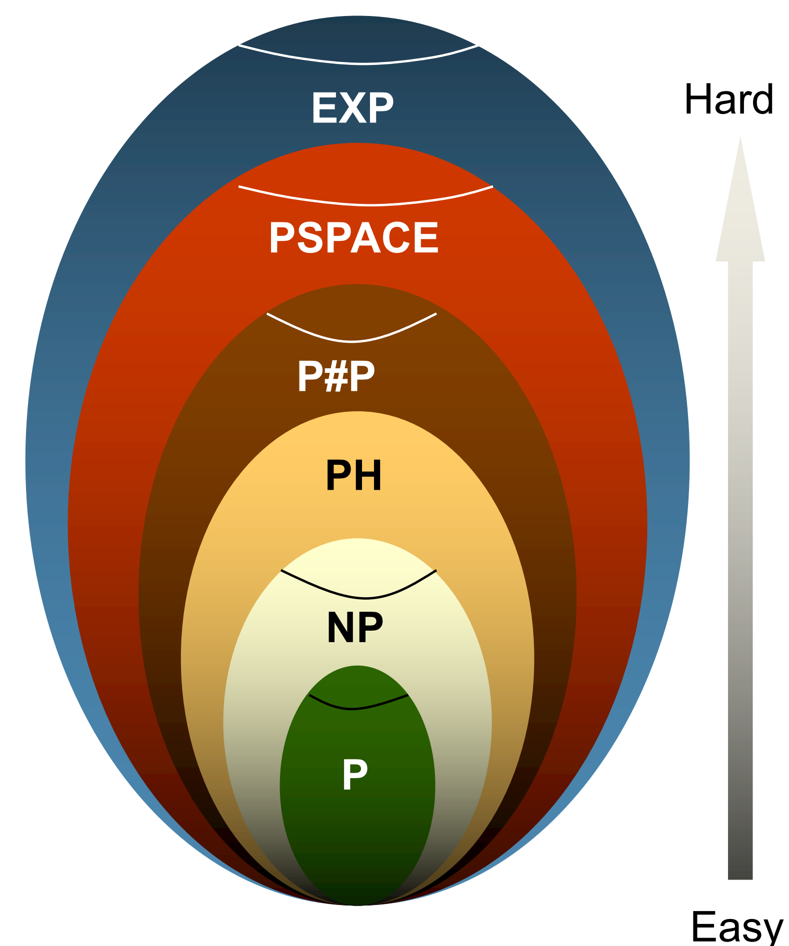
Statistical Mechanics

$$\mathbf{S} = \{+1, -1\}^n$$



$$P(\mathbf{S}) = \frac{1}{Z} \mathrm{e}^{-\beta E(\mathbf{S})} \qquad Z = \sum_{\mathbf{S}} \mathrm{e}^{-\beta E(\mathbf{S})}$$

- Estimating the free energy
- Computing observables
- Sampling

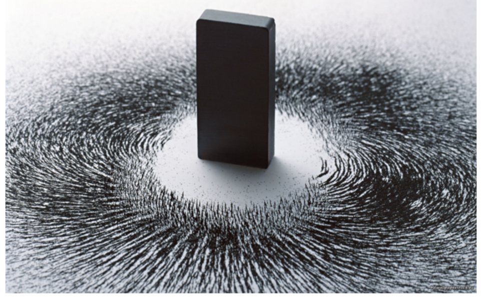


Applications of Statistical Mechanics

- Physics:

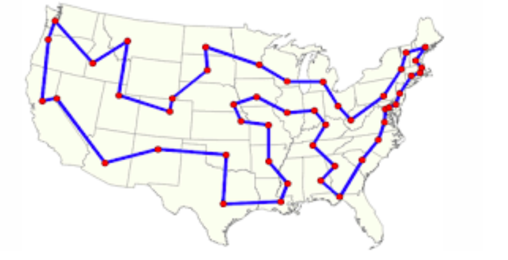
  Thermodynamics,

  Phase transitions …



- Combinatorial Optimization

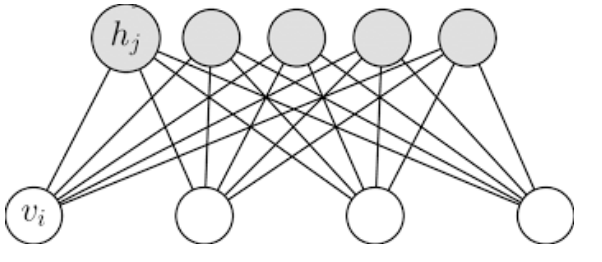
$$P(\mathbf{S}) = \lim_{\beta \to \infty} \frac{1}{Z} e^{-\beta E(\mathbf{S})}$$



- Machine Learning

  Hopfield model,

  Boltzmann machines

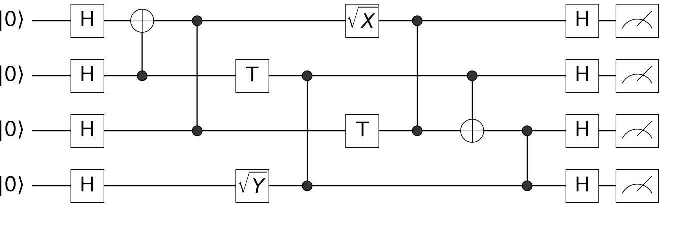


- Statistical Inference

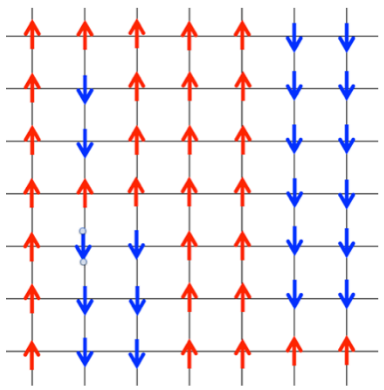
  Bayesian Inference, M.A.P.

- Quantum computation

  Stat. Mech. with complexity interactions



统计物理，机器学习与量子计算

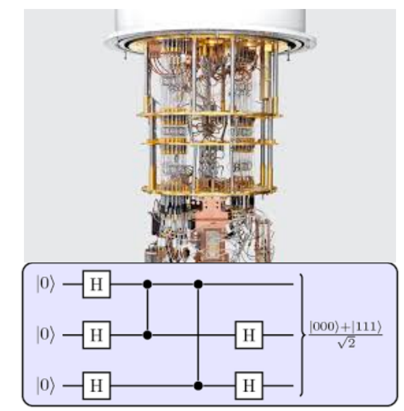


微观构型联合分布概率

$$P(\sigma) = \frac{1}{Z} \exp(-\beta E(\sigma))$$

数据变量联合分布概率

$$P(\text{Data})$$



量子态的操纵

$$\psi(\sigma)$$

指数大的空间

有效的模型

强力的计算能力

例子1：统计力学与统计推断

统计力学

$$\mathbf{S} = \{+1, -1\}^n$$

↑ ↑ ↑ ↓ ↑ ↓ ↓ ↓ ↓ ↓ ↑ ↑

$$P(\mathbf{S}) = \frac{1}{Z} \mathrm{e}^{-\beta E(\mathbf{S})} \qquad Z = \sum_{\mathbf{s}} \mathrm{e}^{-\beta E(\mathbf{S})}$$

配分函数

统计推断

Given data $\mathbf{x}$ and model $p(\mathbf{x}|\mathbf{s})$, find latent variable $\mathbf{s}$

$$p(\mathbf{s}|\mathbf{x}) = \frac{1}{Z} p(\mathbf{x}|\mathbf{s}) p_0(\mathbf{s}) \qquad Z = p(\mathbf{x}) = \sum_{\mathbf{s}} p(\mathbf{x}|\mathbf{s}) p_0(\mathbf{s})$$

模型似然度

↑ ↑ ↑ ↓ ↑ ↓ ↓ ↓ ↓ ↓ ↑ ↑

例子1：统计力学与统计推断

↑ ↑ ↑ ↓ ↑ ↓ ↓ ↓ ↓ ↓ ↑ ↑ **x:** data

Estimate *q*, prob of positive
from *n* events, where *m* of them are positive

$$P(q|x) = \frac{P(x,q)}{\int P(x,q)dq} = \frac{P(x|q)P_0(q)}{\int P(x|q)P_0(q)dq} = \frac{e^{\ln[P(x|q)P_0(q)]}}{\int e^{\ln[P(x|q)P_0(q)]}dq}$$

$$= \frac{e^{\ln\left[q^n(1-q)^{m-n}\right]}}{\int e^{\ln[q^n(1-q)^{m-n}]}dq}$$

$$\widehat{q} = \arg\min_q \ln\left[q^m(1-q)^{n-m}\right]$$

$$q = \frac{m}{n}$$

例子1：统计力学与统计推断的字典

例子1：统计力学与统计推断的字典

Statistical Mechanics	Statistical Inference/Learning

例子1：统计力学与统计推断的字典

Statistical Mechanics		Statistical Inference/Learning	
A configuration	$\mathbf{s}$	x and $\mathbf{s}$	Observable and latent variable

例子1：统计力学与统计推断的字典

Statistical Mechanics		Statistical Inference/Learning	
A configuration	$\mathbf{s}$	x and $\mathbf{s}$	Observable and latent variable
Weight of a configuration	$e^{-\beta E(\mathbf{s})}$	$p(x\|\mathbf{s})$	Likelihood

例子1：统计力学与统计推断的字典

Statistical Mechanics		Statistical Inference/Learning	
A configuration	$\mathbf{s}$	x and $\mathbf{s}$	Observable and latent variable
Weight of a configuration	$e^{-\beta E(\mathbf{s})}$	$p(x\|\mathbf{s})$	Likelihood
Energy	$E(\mathbf{s})$	$-\ln[p(x\|\mathbf{s})]$	Negative log-likelihood

例子1：统计力学与统计推断的字典

Statistical Mechanics		Statistical Inference/Learning	
A configuration	$\mathbf{s}$	x and $\mathbf{s}$	Observable and latent variable
Weight of a configuration	$e^{-\beta E(\mathbf{s})}$	$p(x\|\mathbf{s})$	Likelihood
Energy	$E(\mathbf{s})$	$-\ln[p(x\|\mathbf{s})]$	Negative log-likelihood
Boltzmann distribution	$p(\mathbf{s}) = \frac{1}{Z} e^{-\beta E(\mathbf{s})}$	$p(\mathbf{s}\|x) = \frac{1}{p(x)} p(x\|\mathbf{s}) p_0(\mathbf{s})$	Posterior of Bayesian Inference

例子1：统计力学与统计推断的字典

Statistical Mechanics		Statistical Inference/Learning	
A configuration	$\mathbf{s}$	x and $\mathbf{s}$	Observable and latent variable
Weight of a configuration	$e^{-\beta E(\mathbf{s})}$	$p(x\|\mathbf{s})$	Likelihood
Energy	$E(\mathbf{s})$	$-\ln[p(x\|\mathbf{s})]$	Negative log-likelihood
Boltzmann distribution	$p(\mathbf{s}) = \frac{1}{Z} e^{-\beta E(\mathbf{s})}$	$p(\mathbf{s}\|x) = \frac{1}{p(x)} p(x\|\mathbf{s}) p_0(\mathbf{s})$	Posterior of Bayesian Inference
Partition Function	$Z = \sum_{\mathbf{s}} e^{-\beta E(\mathbf{s})}$	$p(x) = \sum_{\mathbf{s}} p(x\|\mathbf{s}) p_0(\mathbf{s})$	Marginal likelihood

例子1：统计力学与统计推断的字典

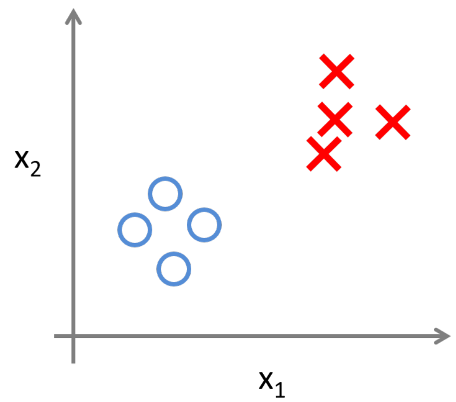
Statistical Mechanics		Statistical Inference/Learning	
A configuration	$\mathbf{s}$	x and $\mathbf{s}$	Observable and latent variable
Weight of a configuration	$e^{-\beta E(\mathbf{s})}$	$p(x\|\mathbf{s})$	Likelihood
Energy	$E(\mathbf{s})$	$-\ln[p(x\|\mathbf{s})]$	Negative log-likelihood
Boltzmann distribution	$p(\mathbf{s}) = \frac{1}{Z} e^{-\beta E(\mathbf{s})}$	$p(\mathbf{s}\|x) = \frac{1}{p(x)} p(x\|\mathbf{s}) p_0(\mathbf{s})$	Posterior of Bayesian Inference
Partition Function	$Z = \sum_{\mathbf{s}} e^{-\beta E(\mathbf{s})}$	$p(x) = \sum_{\mathbf{s}} p(x\|\mathbf{s}) p_0(\mathbf{s})$	Marginal likelihood
Free energy	$F = -\frac{1}{\beta} \ln Z$	$\mathcal{L} = \ln p(x)$	Marginal log-likelihood

例子1：统计力学与统计推断的字典

Statistical Mechanics		Statistical Inference/Learning	
A configuration	$\mathbf{s}$	x and $\mathbf{s}$	Observable and latent variable
Weight of a configuration	$e^{-\beta E(\mathbf{s})}$	$p(x\|\mathbf{s})$	Likelihood
Energy	$E(\mathbf{s})$	$-\ln[p(x\|\mathbf{s})]$	Negative log-likelihood
Boltzmann distribution	$p(\mathbf{s}) = \frac{1}{Z} e^{-\beta E(\mathbf{s})}$	$p(\mathbf{s}\|x) = \frac{1}{p(x)} p(x\|\mathbf{s}) p_0(\mathbf{s})$	Posterior of Bayesian Inference
Partition Function	$Z = \sum_{\mathbf{s}} e^{-\beta E(\mathbf{s})}$	$p(x) = \sum_{\mathbf{s}} p(x\|\mathbf{s}) p_0(\mathbf{s})$	Marginal likelihood
Free energy	$F = -\frac{1}{\beta} \ln Z$	$\mathcal{L} = \ln p(x)$	Marginal log-likelihood
Ground state	$\arg\min_{\mathbf{s}} E(\mathbf{s})$	$\arg\max_{\mathbf{s}} \ln[p(x\|\mathbf{s}) p_0(\mathbf{s})]$	M.A.P. inference

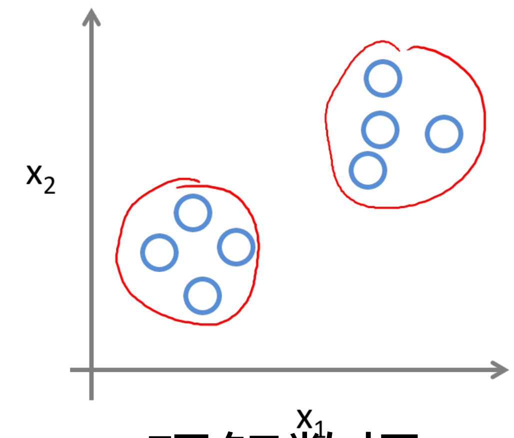
例子2：统计物理与非监督学习

Supervised Learning



预测标签

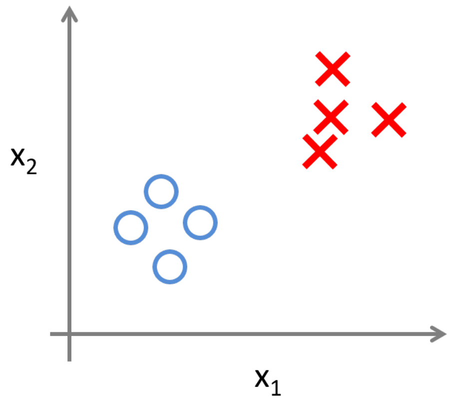
Unsupervised Learning



理解数据

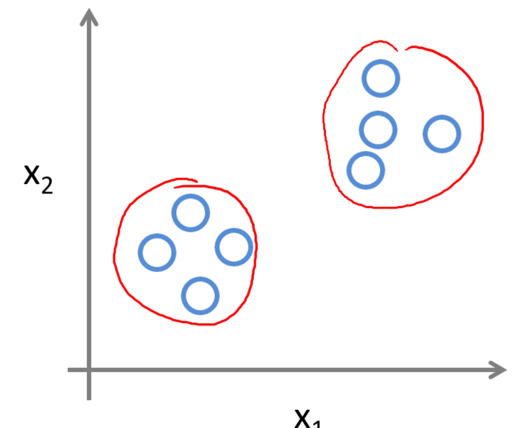
例子2：统计物理与非监督学习

Supervised Learning



预测标签

Unsupervised Learning



理解数据

Discriminative

$$y = f(\boldsymbol{x})$$

or $p(y\,|\,\boldsymbol{x})$

分类器

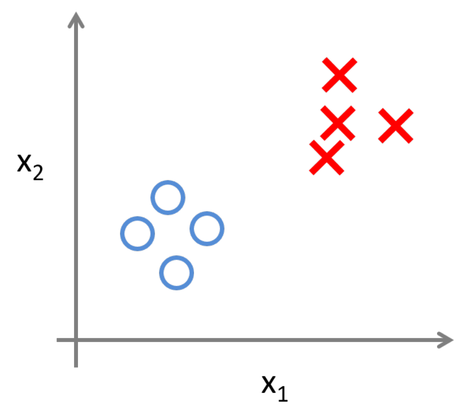
Generative

$$p(\boldsymbol{x}, y)$$

生成新的数据

例子2：统计物理与非监督学习

Supervised Learning



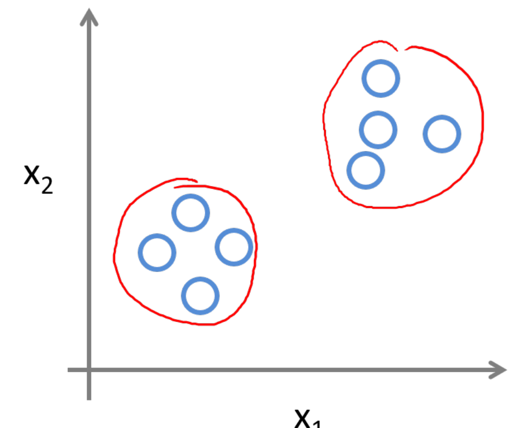
预测标签

Discriminative

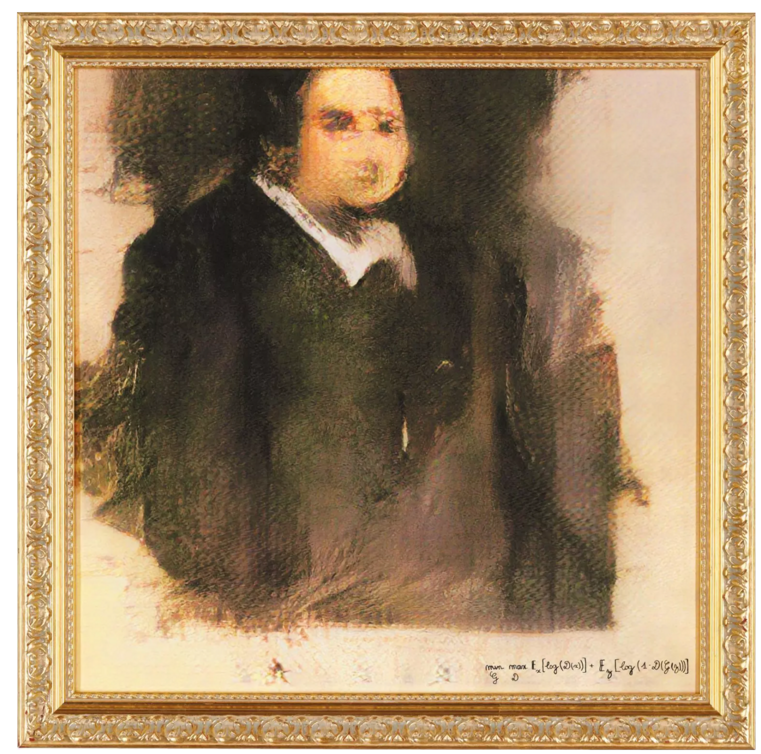
$$y = f(\boldsymbol{x})$$

or $p(y\,|\,\boldsymbol{x})$

Unsupervised Learning



理解数据



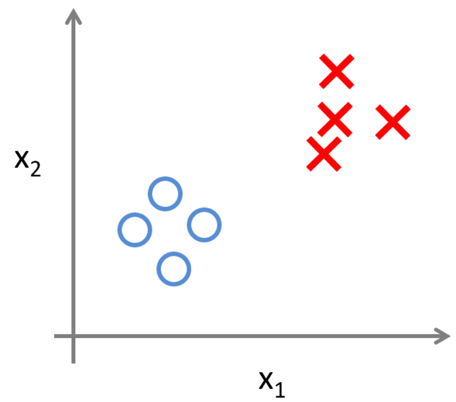
$$p(\boldsymbol{x}, y)$$

A machine learning generated print

sold for $432,500

例子2：统计物理与非监督学习

Supervised Learning



预测标签

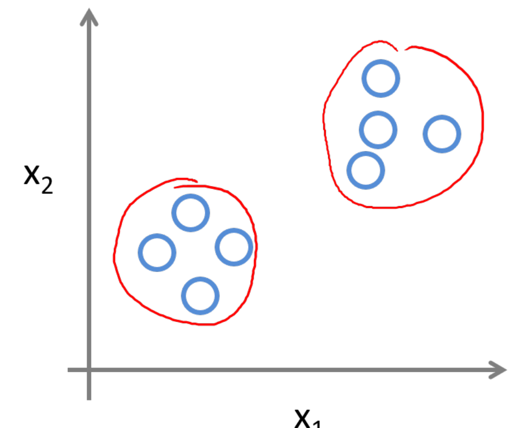
Discriminative

$$y = f(\boldsymbol{x})$$

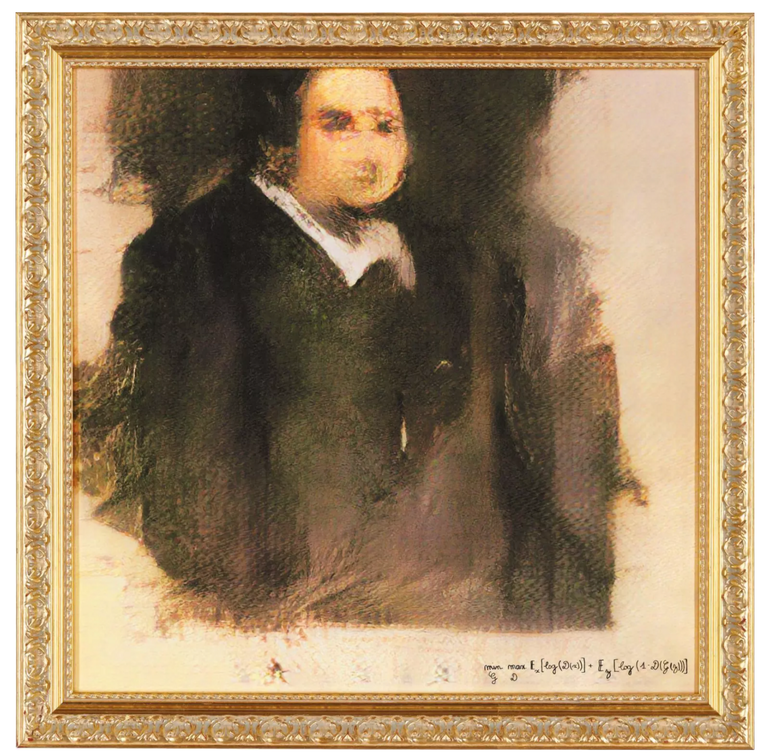
or $p(y\,|\,\boldsymbol{x})$

分类器

Unsupervised Learning



理解数据



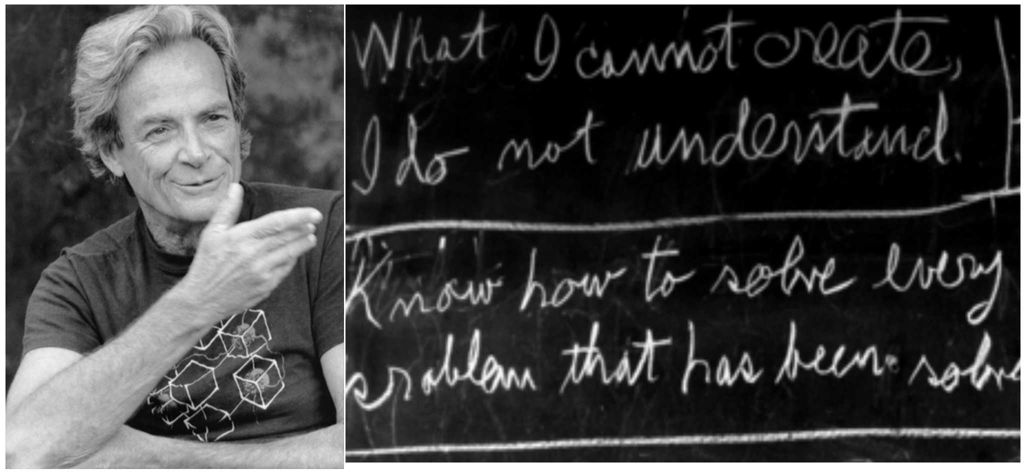
A machine learning generated print

sold for $432,500

$$p(\boldsymbol{x}, y)$$

生成新的数据

例子2：统计物理与非监督学习



"What I can not create,
I do not understand"

Understand： 从高维数据中学习到分布规律
Create： 生成新的数据

生成学习是机器学习的研究前沿

TN models…
Quantum models…

Boltzmann Machines	Variational Autoencoder	Adversarial Networks	Normalizing Flows	Autoregressive Networks	**Born Machines**
1980s	2013	2014	2015	2016	2018

$$\mathbf{P}(\mathbf{x}) = \frac{\mathbf{e}^{-\beta \mathbf{E}(\mathbf{x})}}{\mathcal{Z}}$$

统计物理玻尔兹曼分布表达数据的联合分布

$$\mathbf{P}(\mathbf{x}) = \frac{|\boldsymbol{\Psi}(\mathbf{x})|^2}{\mathcal{N}}$$

- 生成，无偏采样属于#P难问题
- 高维度变量的联合分布难以精确地描述

Z.Han, J. Wang, H. Fan,L. Wang, PZ, Phys. Rev. X. 8, 031012 (2018)

4.4. NEURAL NETWORKS

One wide ranging development, in the statistical physics of neural networks, has been the so-called Gardner approach, namely a statistical analysis in parameter space, i.e. the space of interactions (e.g. synaptic weights). It has been called the inverse problem of statistical mechanics, because in ordinary statistical mechanics the interactions are given and the statistical analysis is done in variable space (e.g. the space of neural activities). At this point,

Gerard Toulouse 1992’

Langevin Prize, Holweck Prize

4.4. NEURAL NETWORKS

One wide ranging development, in the statistical physics of neural networks, has been the so-called Gardner approach, namely a statistical analysis in parameter space, i.e. the space of interactions (e.g. synaptic weights). It has been called the inverse problem of statistical mechanics, because in ordinary statistical mechanics the interactions are given and the statistical analysis is done in variable space (e.g. the space of neural activities). At this point,

Gerard Toulouse 1992'

Langevin Prize, Holweck Prize

4.4. NEURAL NETWORKS

One wide ranging development, in the statistical physics of neural networks, has been the so-called Gardner approach, namely a statistical analysis in parameter space, i.e. the space of interactions (e.g. synaptic weights). It has been called the inverse problem of statistical mechanics, because in ordinary statistical mechanics the interactions are given and the statistical analysis is done in variable space (e.g. the space of neural activities). At this point,

Gerard Toulouse 1992'

Langevin Prize, Holweck Prize

4.4. NEURAL NETWORKS

One wide ranging development, in the statistical physics of neural networks, has been the so-called Gardner approach, namely a statistical analysis in parameter space, i.e. the space of interactions (e.g. synaptic weights). It has been called the inverse problem of statistical mechanics, because in ordinary statistical mechanics the interactions are given and the statistical analysis is done in variable space (e.g. the space of neural activities). At this point,

Gerard Toulouse 1992'

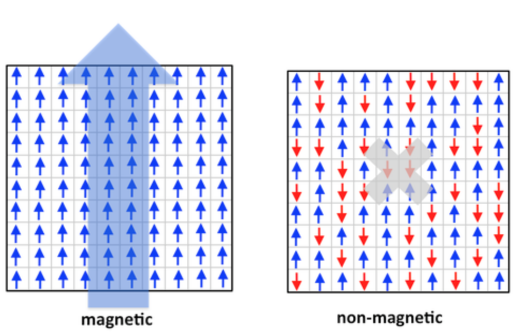
Langevin Prize, Holweck Prize

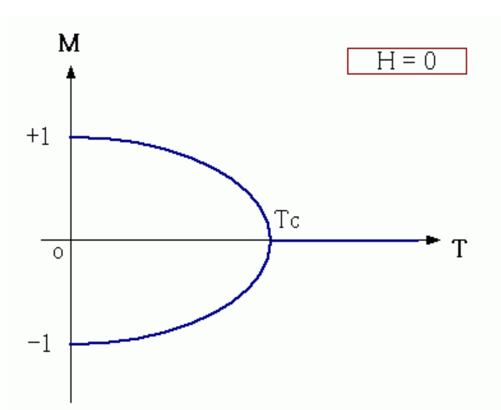
The Ising model
(E. Ising, 1924)

$$x = \{+1, -1\}$$

$$P(x) = \frac{1}{Z} e^{\beta\left(\sum_{(ij)} J_{ij} x_i x_j + \sum_i \theta_i x_i\right)}$$

Ordinary Statistical Mechanics





4.4. NEURAL NETWORKS

One wide ranging development, in the statistical physics of neural networks, has been the so-called Gardner approach, namely a statistical analysis in parameter space, i.e. the space of interactions (e.g. synaptic weights). It has been called the inverse problem of statistical mechanics, because in ordinary statistical mechanics the interactions are given and the statistical analysis is done in variable space (e.g. the space of neural activities). At this point,

Gerard Toulouse 1992'

Langevin Prize, Holweck Prize

The Ising model
(E. Ising, 1924)

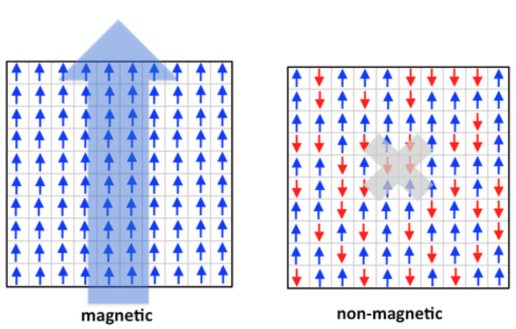
$$x = \{+1, -1\}$$

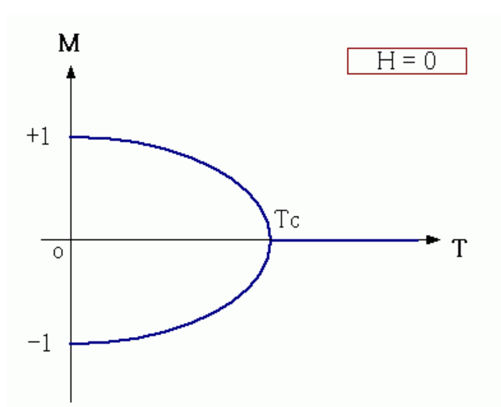
$$P(x) = \frac{1}{Z} e^{\beta(\sum_{(ij)} J_{ij} x_i x_j + \sum_i \theta_i x_i)}$$

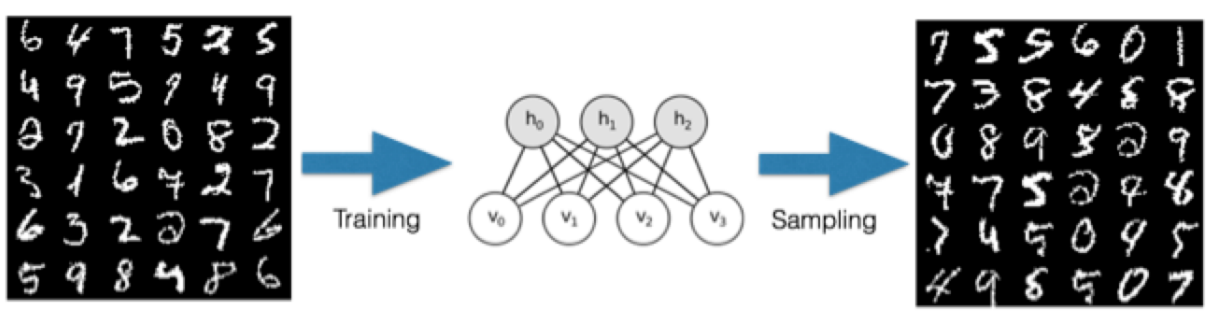
Restricted Boltzmann Machine
(Ackley, Hinton, Sejnowski, 1985)

Ordinary Statistical Mechanics

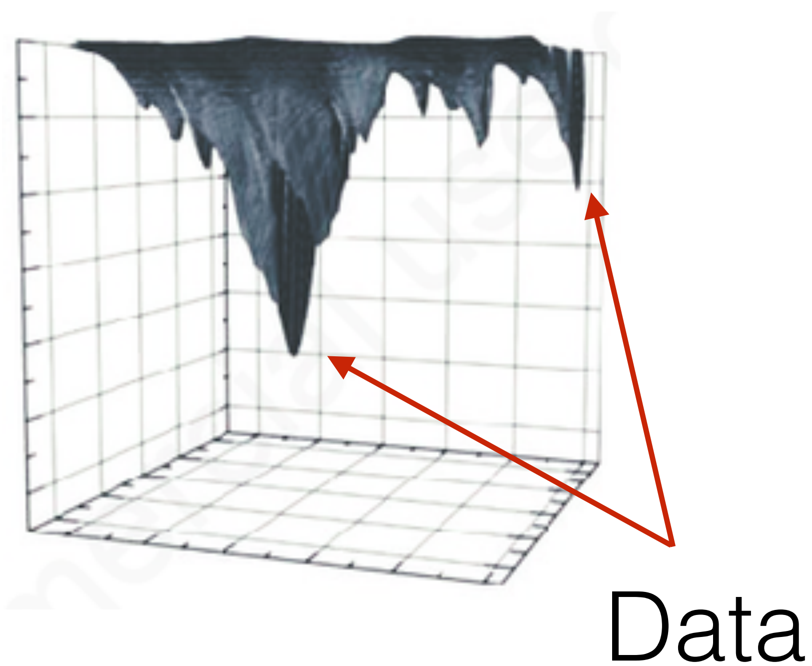
Inverse problem of statistical mechanics



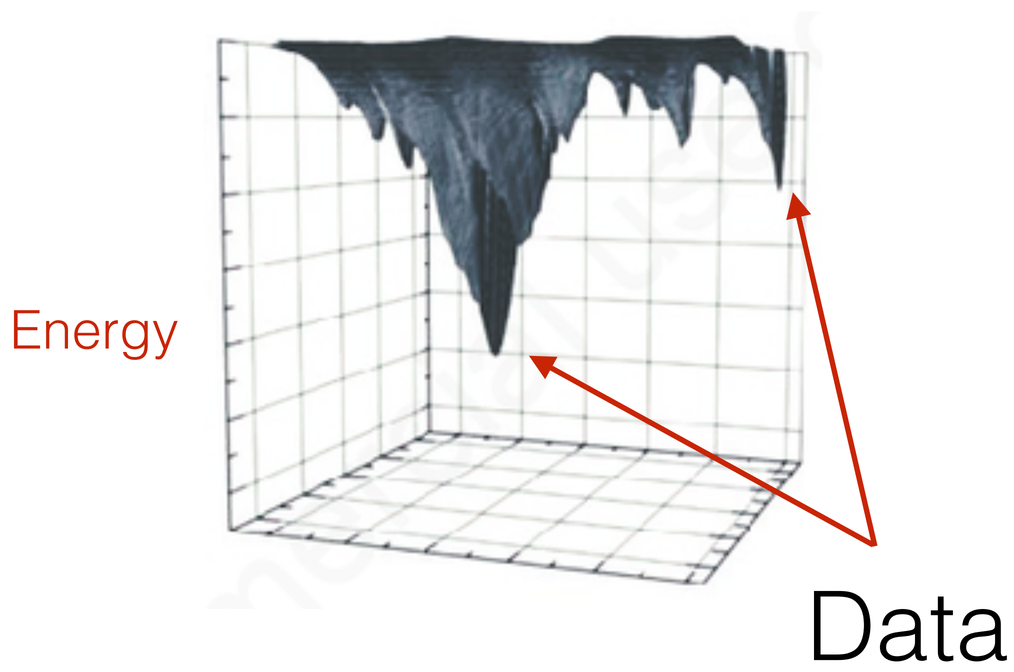




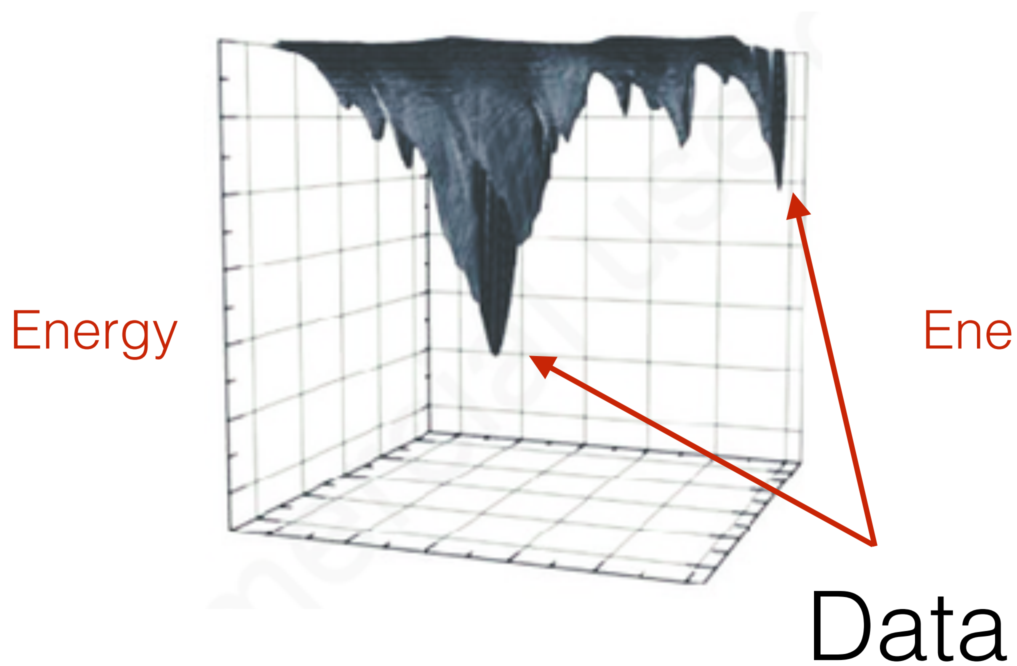
Data live in the corner of the high-dimensional space

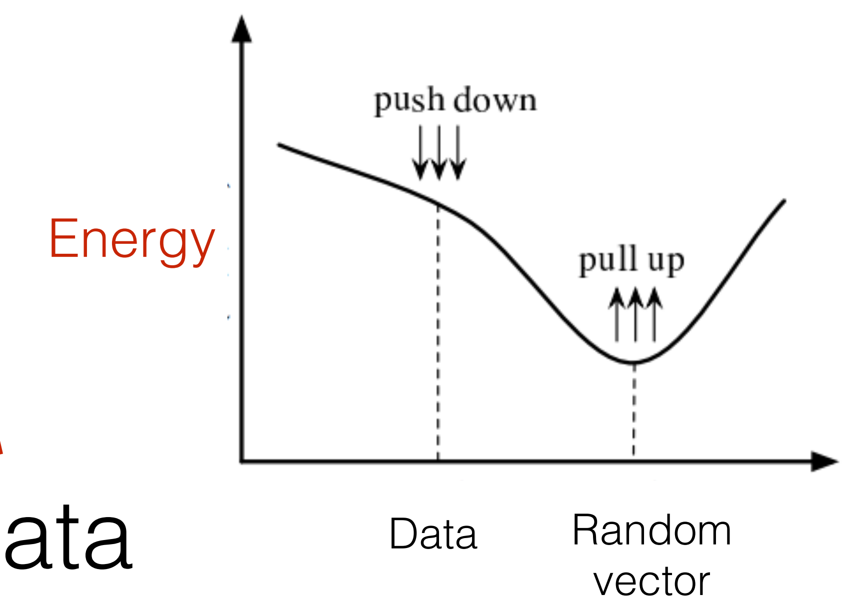


Data live in the corner of the high-dimensional space



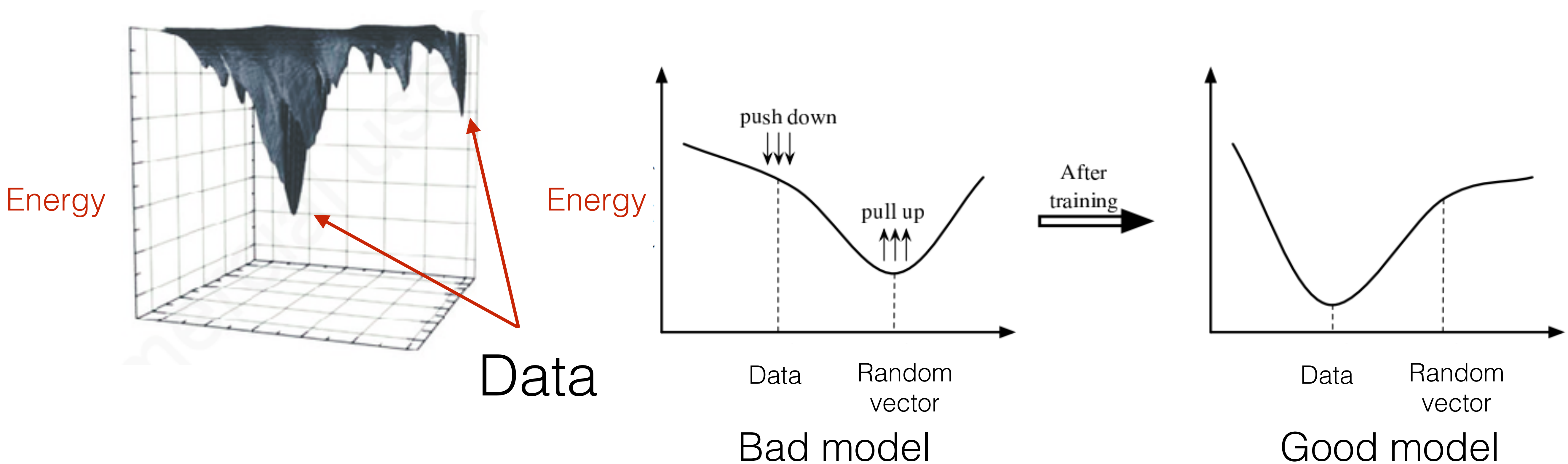
Data live in the corner of the high-dimensional space

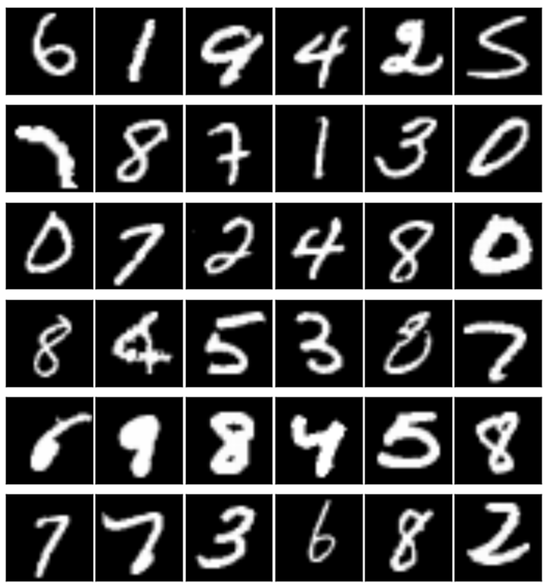




Bad model

Data live in the corner of the high-dimensional space



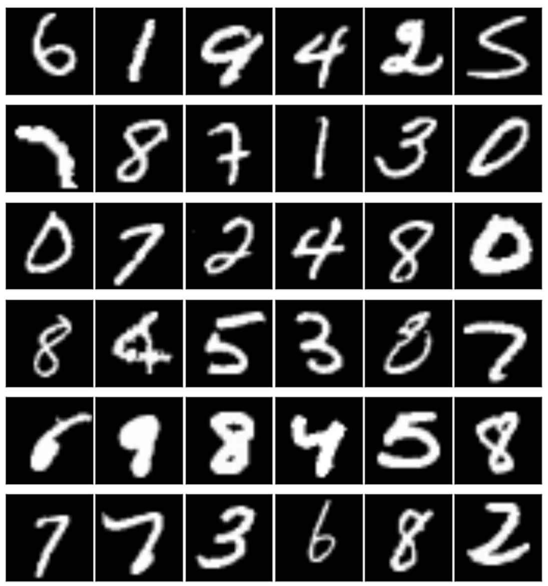


Data: a vector x of dimension n

$$\mathbf{x} \in \{1, 0\}^n$$

In case of the MNIST:

$$\mathbf{x} \in \{1, 0\}^{784}$$



$$P(\mathbf{x}) = \frac{1}{Z}\mathrm{e}^{-\beta E(\mathbf{x})}$$

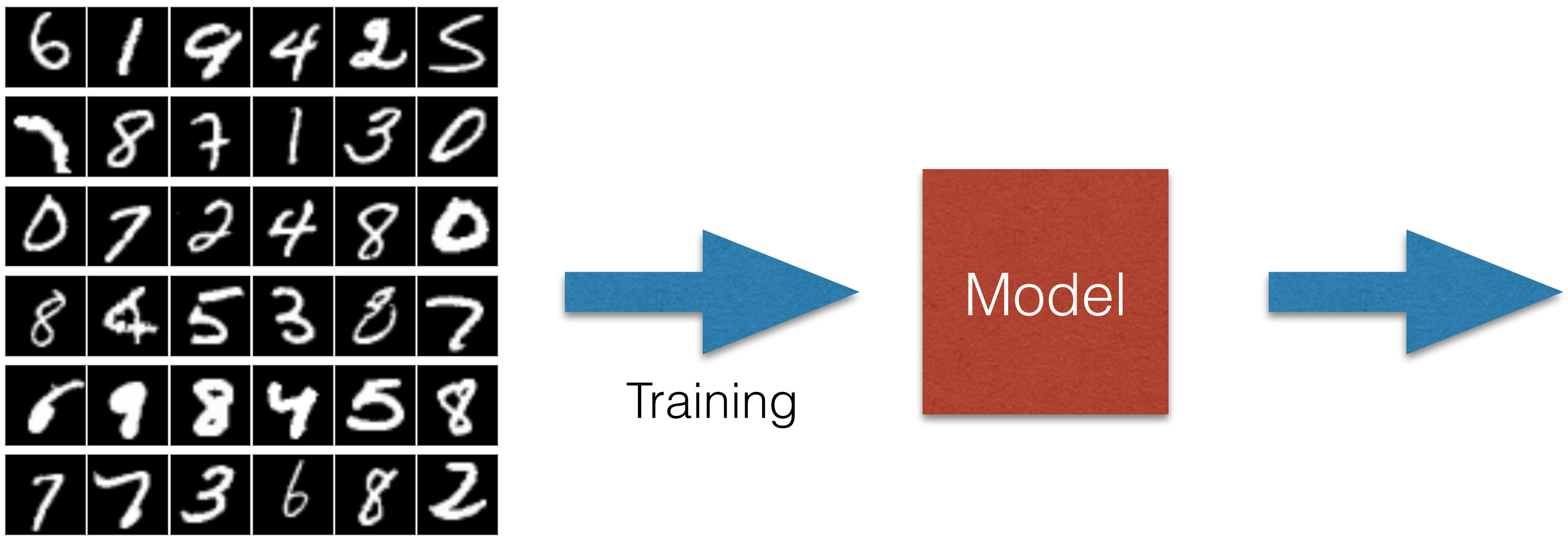
Data: a vector x of dimension n

$$\mathbf{x} \in \{1, 0\}^n$$

$$Z = \sum_{\{\mathbf{x}\}} \mathrm{e}^{-\beta E(\mathbf{x})}$$

In case of the MNIST:

$$\mathbf{x} \in \{1, 0\}^{784}$$



$$P(\mathbf{x}) = \frac{1}{Z} \mathrm{e}^{-\beta E(\mathbf{x})}$$

$$Z = \sum_{\{\mathbf{x}\}} \mathrm{e}^{-\beta E(\mathbf{x})}$$

Data: a vector x of dimension n

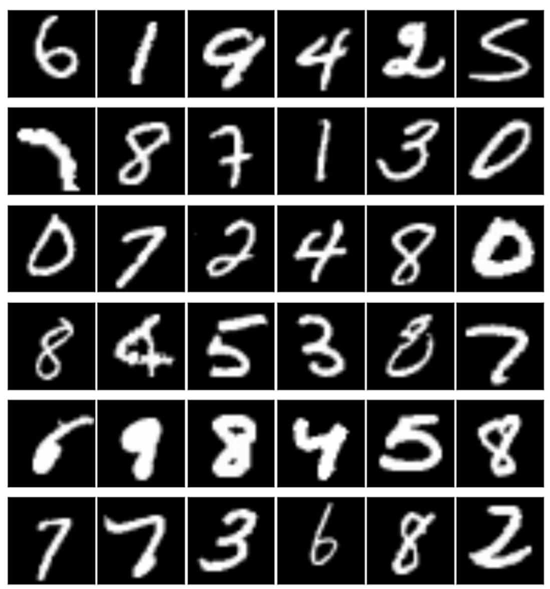
$$\mathbf{x} \in \{1, 0\}^n$$

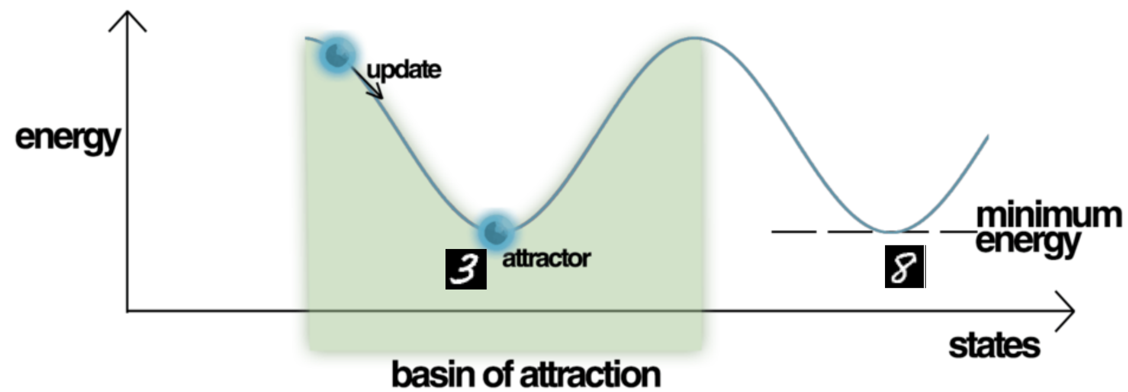
2^n terms

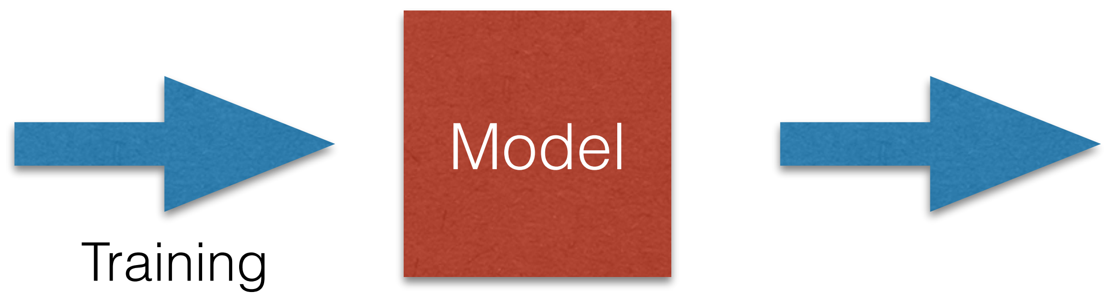
In case of the MNIST:

$$\mathbf{x} \in \{1, 0\}^{784}$$

Partition function







$$P(\mathbf{x}) = \frac{1}{Z} e^{-\beta E(\mathbf{x})}$$

$$Z = \sum_{\{\mathbf{x}\}} e^{-\beta E(\mathbf{x})}$$

2^n terms

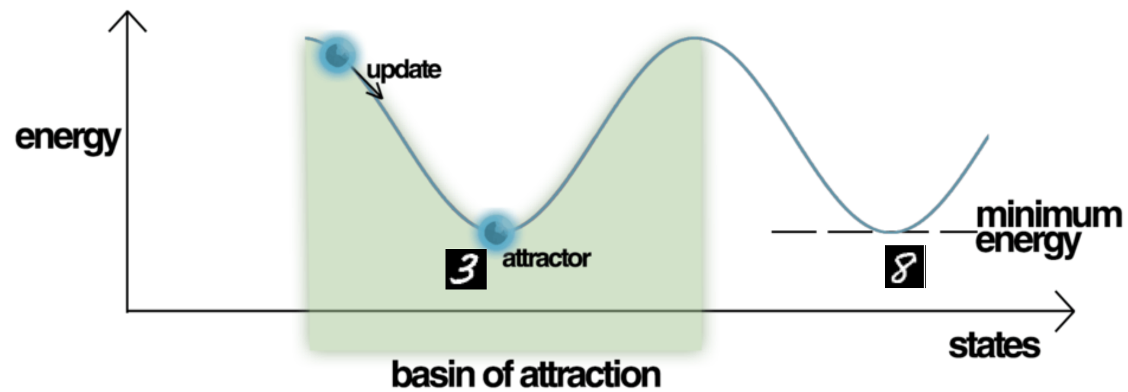
Partition function

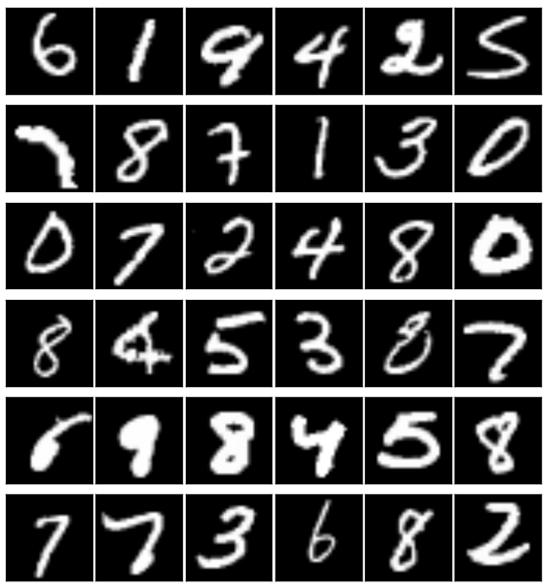
Data: a vector x of dimension n

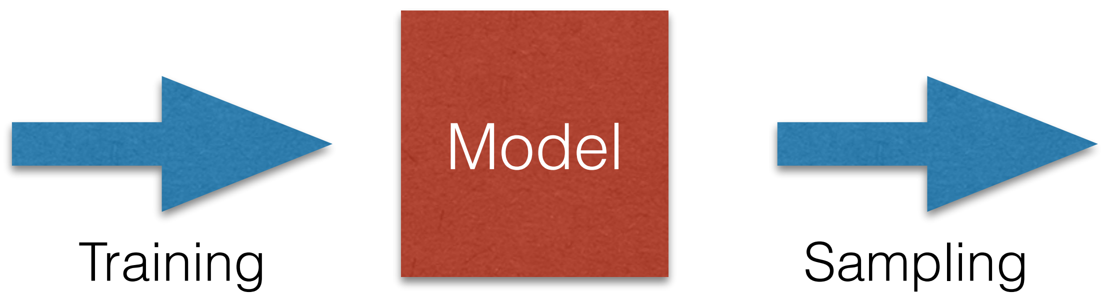
$$\mathbf{x} \in \{1, 0\}^n$$

In case of the MNIST:

$$\mathbf{x} \in \{1, 0\}^{784}$$







$$P(\mathbf{x}) = \frac{1}{Z}\mathrm{e}^{-\beta E(\mathbf{x})}$$

$$Z = \sum_{\{\mathbf{x}\}} \mathrm{e}^{-\beta E(\mathbf{x})}$$

2^n terms

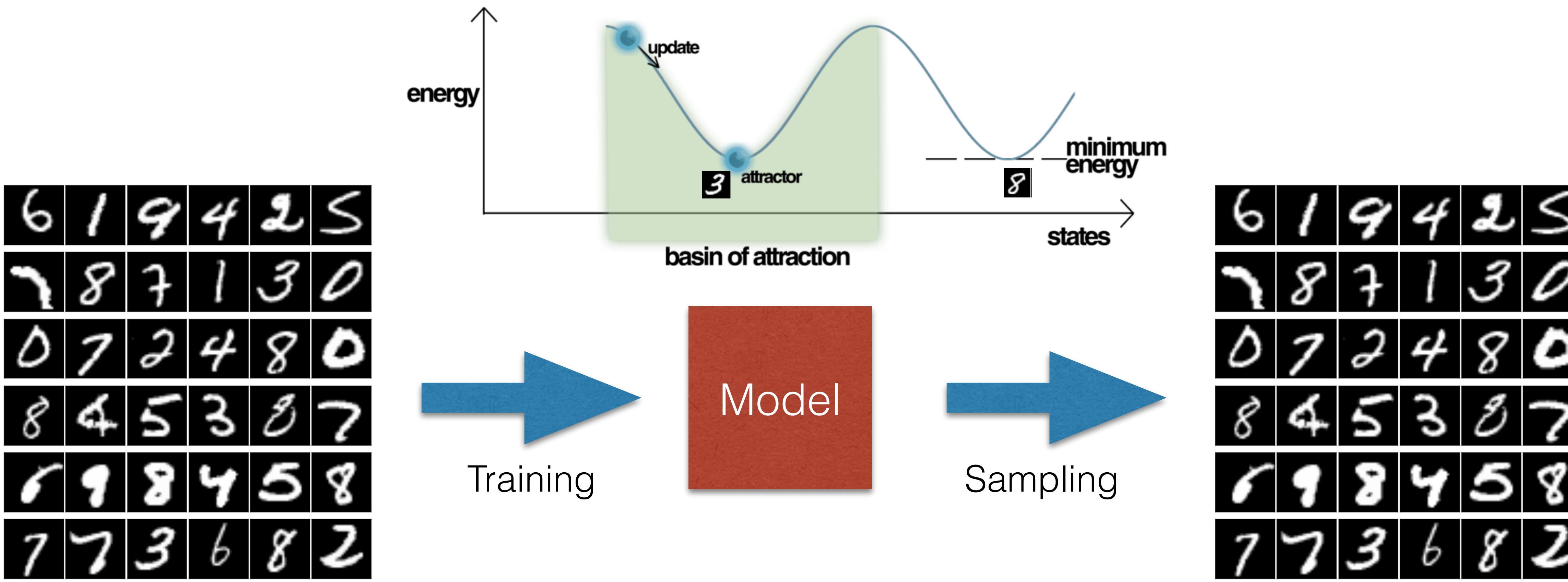
Partition function

Data: a vector x of dimension n

$$\mathbf{x} \in \{1, 0\}^n$$

In case of the MNIST:

$$\mathbf{x} \in \{1, 0\}^{784}$$



$$P(\mathbf{x}) = \frac{1}{Z} \mathrm{e}^{-\beta E(\mathbf{x})}$$

$$Z = \sum_{\{\mathbf{x}\}} \mathrm{e}^{-\beta E(\mathbf{x})}$$

Data: a vector x of dimension n

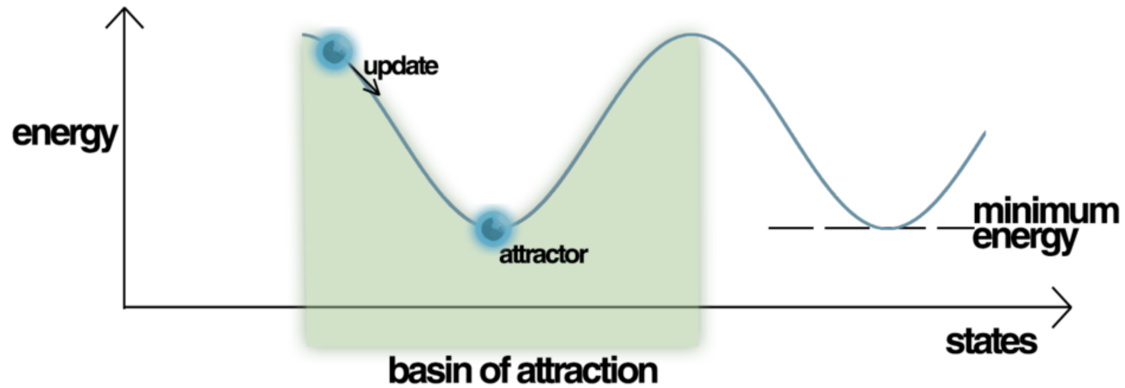
$$\mathbf{x} \in \{1, 0\}^n$$

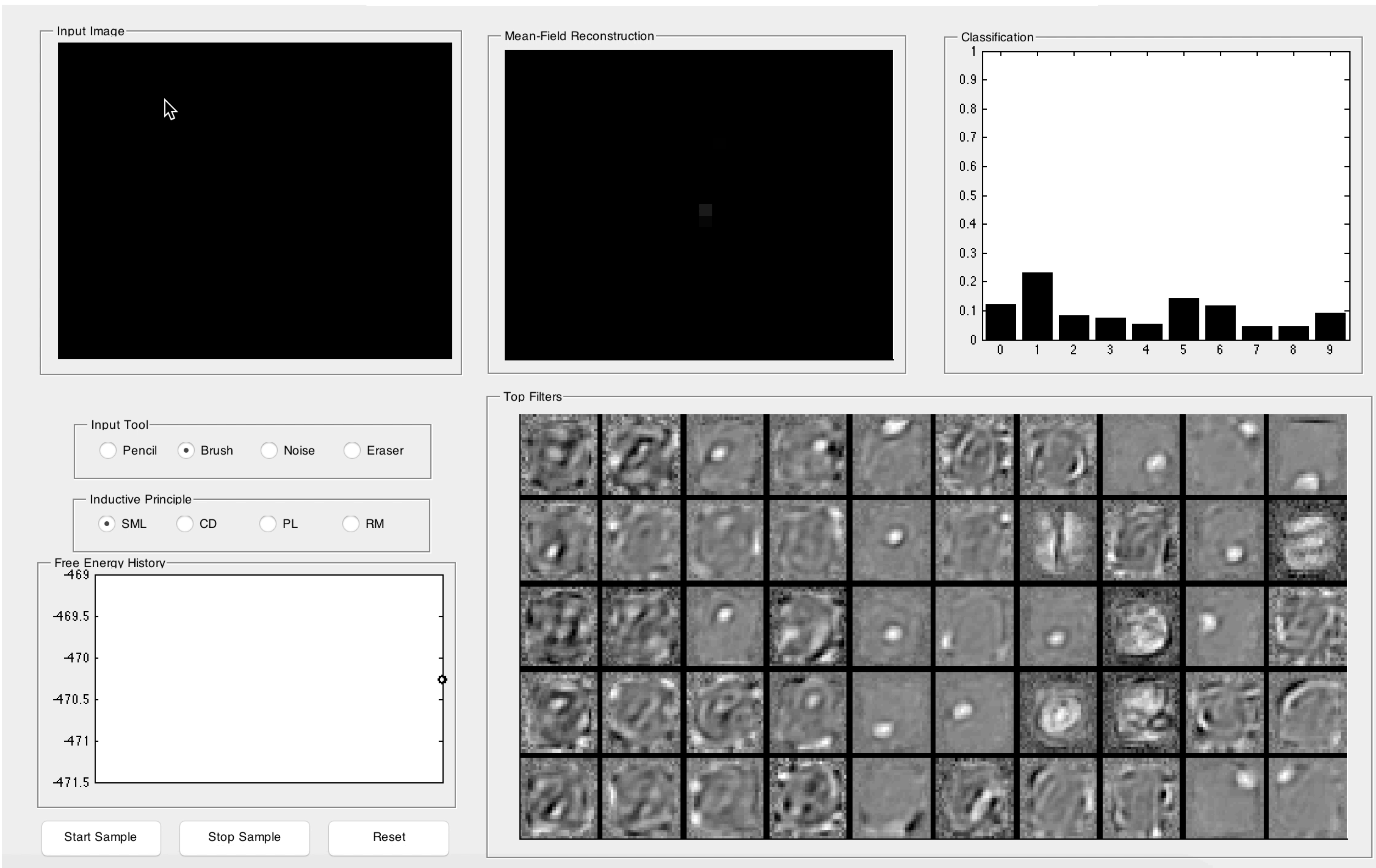
2^n terms

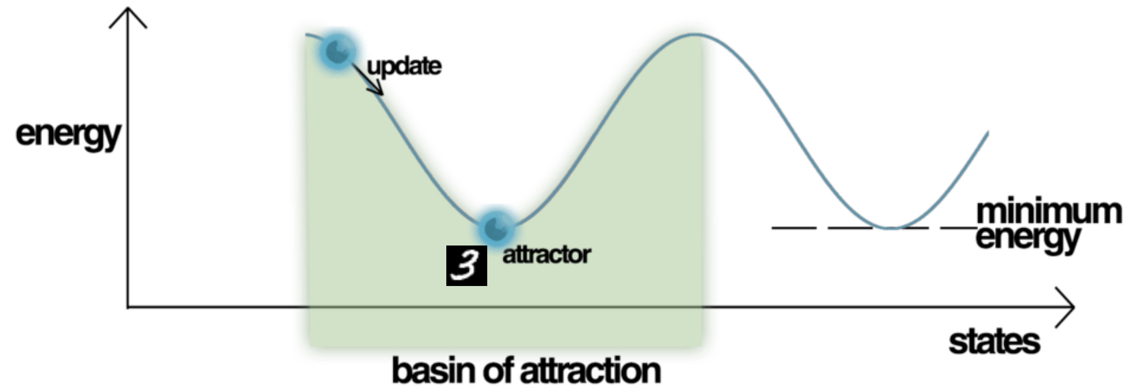
In case of the MNIST:

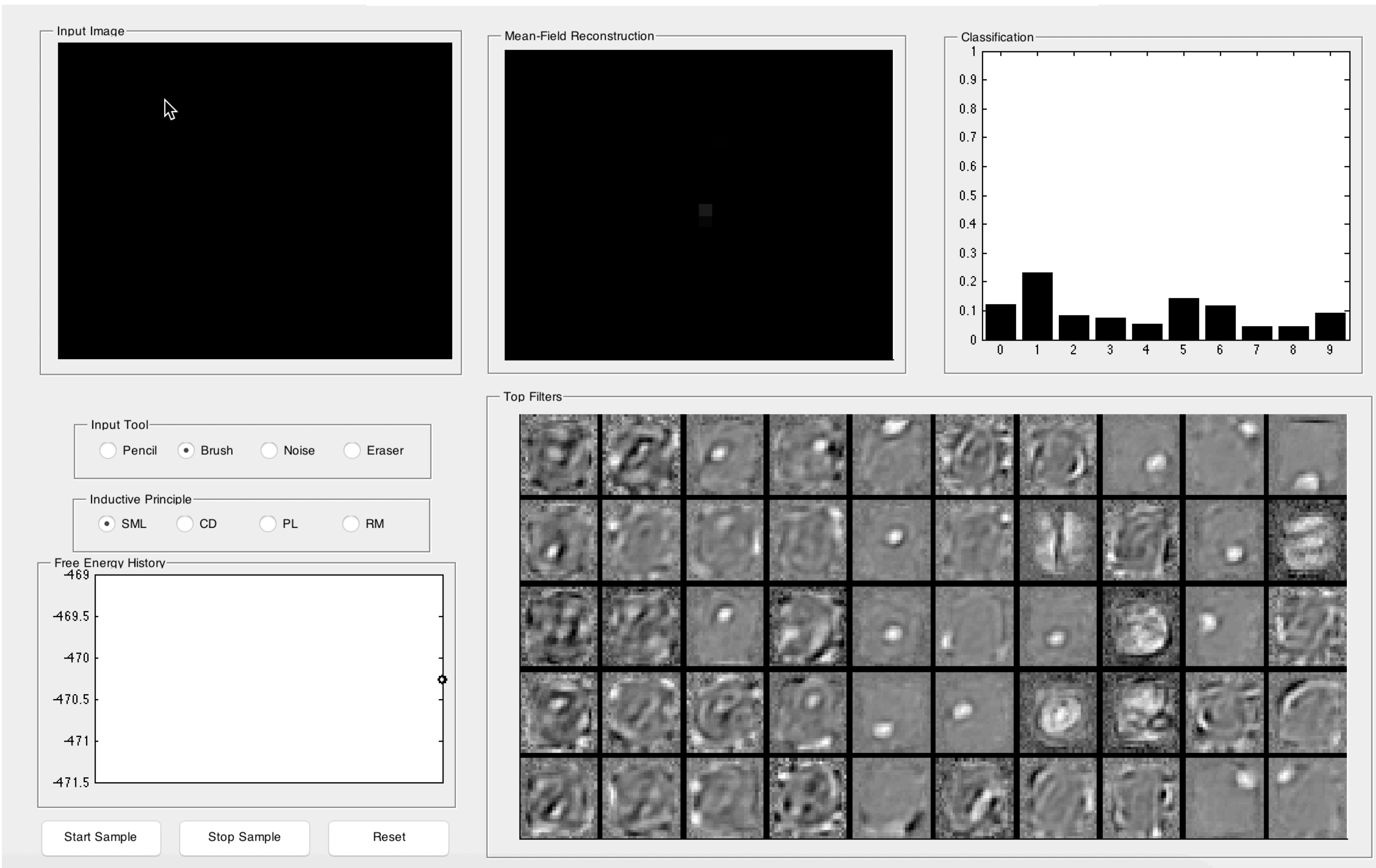
$$\mathbf{x} \in \{1, 0\}^{784}$$

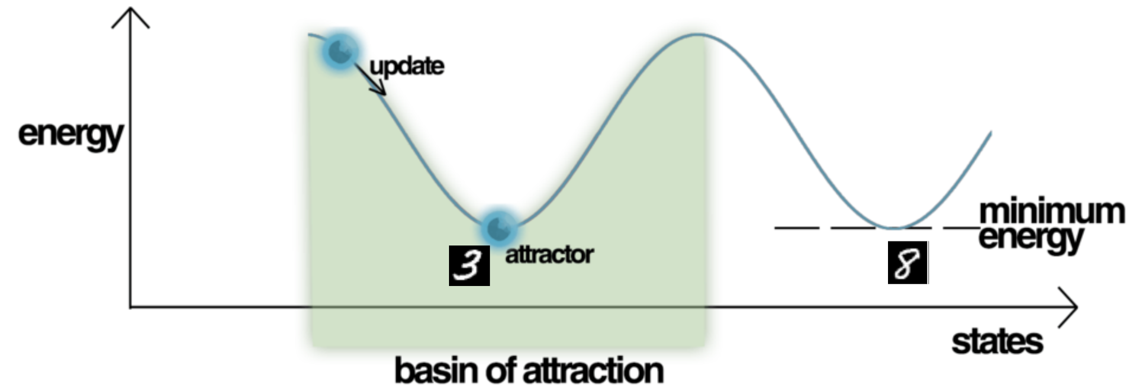
Partition function

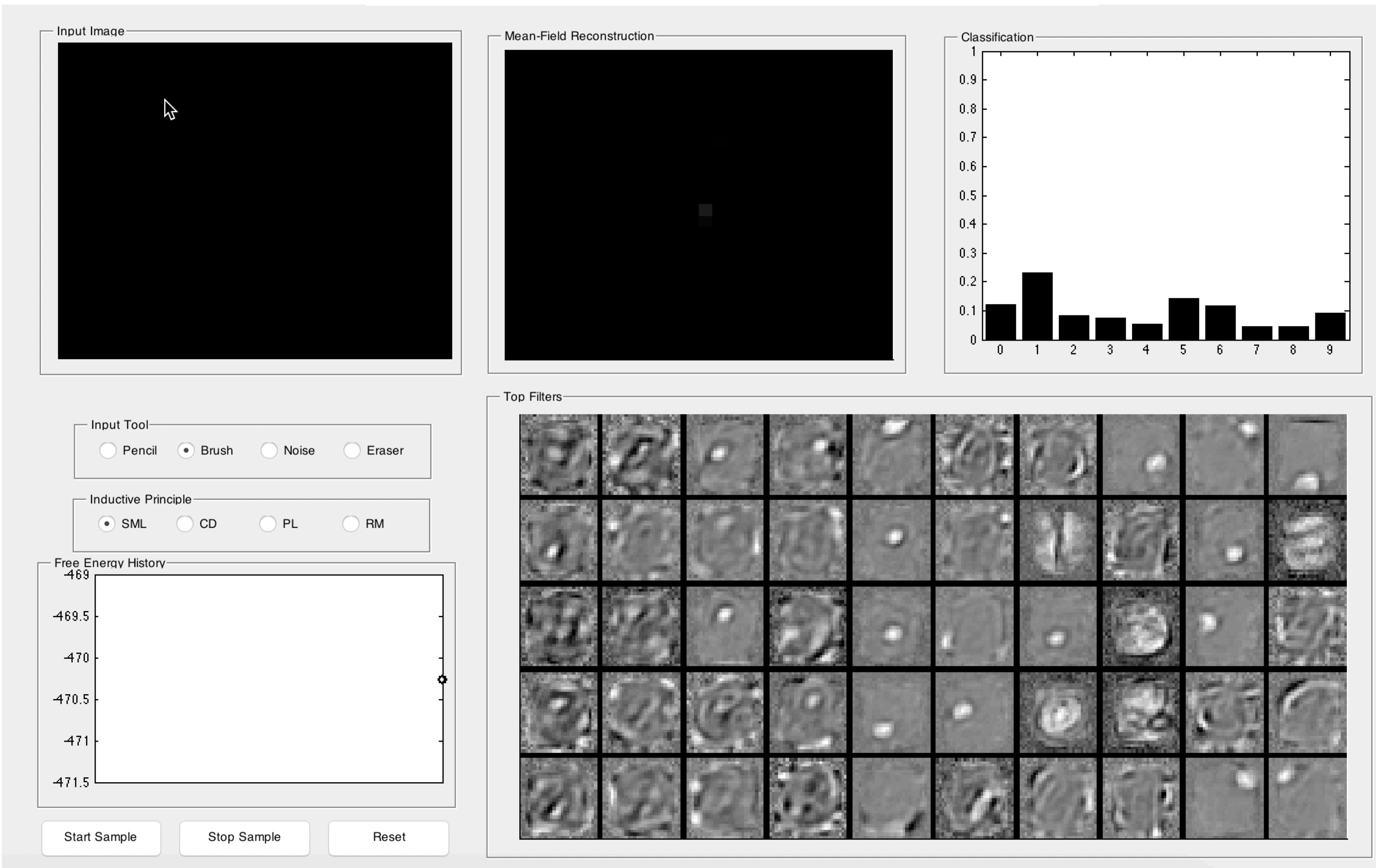

energy
update
attractor
minimum energy
states
basin of attraction

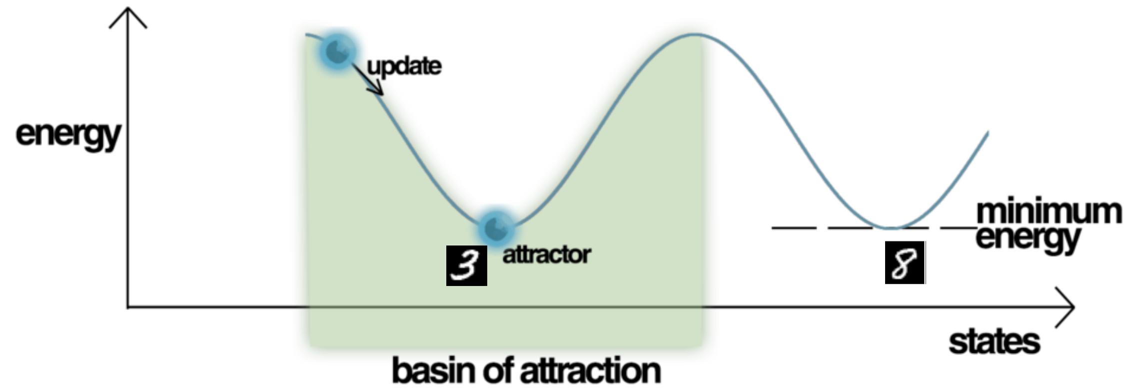

Input Image
Mean-Field Reconstruction
Classification
Top Filters
Input Tool
Pencil
Brush
Noise
Eraser
Inductive Principle
SML
CD
PL
RM
Free Energy History
Start Sample
Stop Sample
Reset

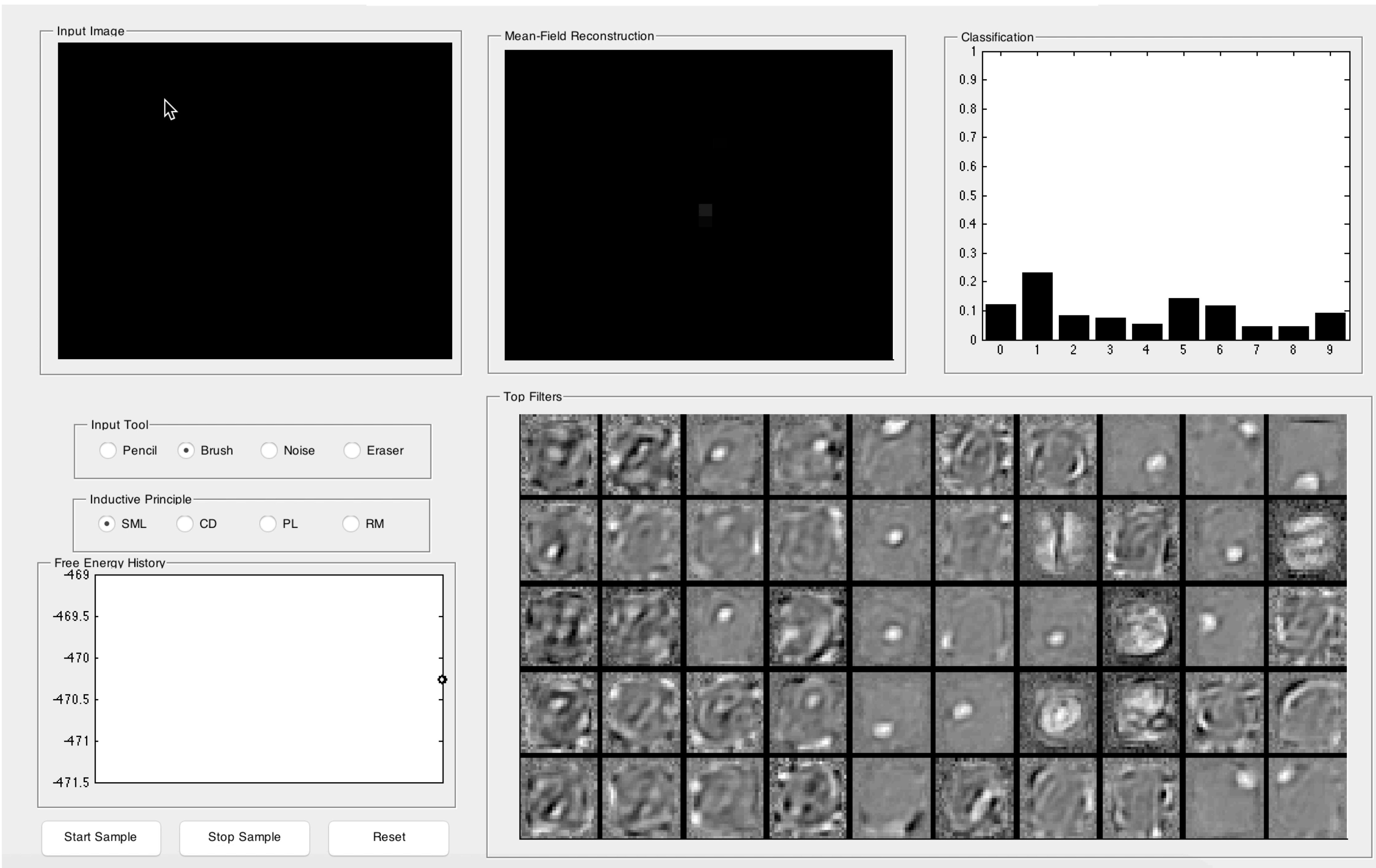

energy
update
attractor
3
minimum energy
states
basin of attraction


Input Image
Mean-Field Reconstruction
Classification
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0 1 2 3 4 5 6 7 8 9
Top Filters
Input Tool
Pencil
Brush
Noise
Eraser
Inductive Principle
SML
CD
PL
RM
Free Energy History
-469
-469.5
-470
-470.5
-471
-471.5
Start Sample
Stop Sample
Reset


energy
update
attractor
minimum energy
states
basin of attraction


Input Image
Mean-Field Reconstruction
Classification
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0 1 2 3 4 5 6 7 8 9
Top Filters
Input Tool
Pencil
Brush
Noise
Eraser
Inductive Principle
SML
CD
PL
RM
Free Energy History
-469
-469.5
-470
-470.5
-471
-471.5
Start Sample
Stop Sample
Reset


energy
update
attractor
minimum energy
states
basin of attraction


Input Image
Mean-Field Reconstruction
Classification
Top Filters
Input Tool
Pencil
Brush
Noise
Eraser
Inductive Principle
SML
CD
PL
RM
Free Energy History
Start Sample
Stop Sample
Reset

The simplest energy-based model

The simplest energy-based model

The Ising model (E. Ising, 1924)

The simplest energy-based model

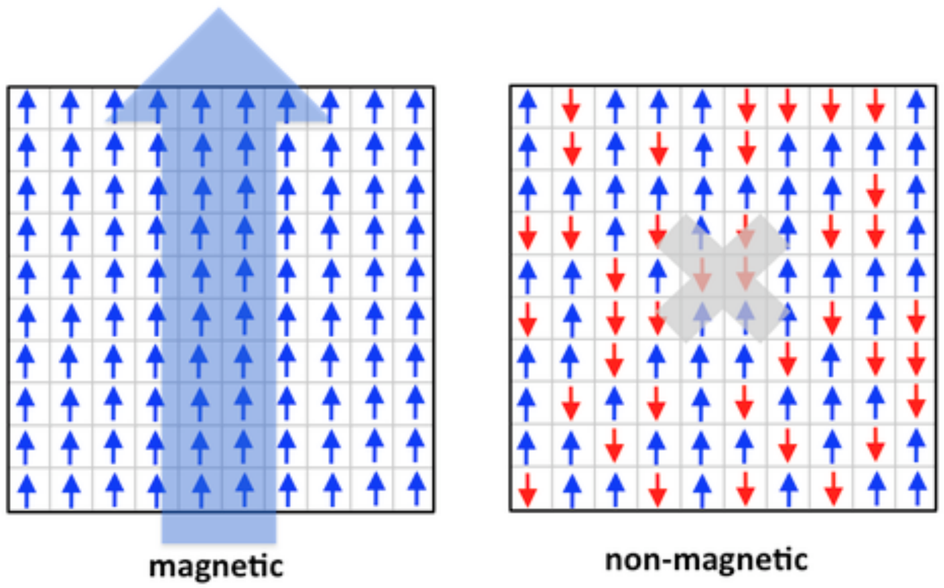
The Ising model (E. Ising, 1924)

$$\mathbf{x} = \{+1, -1\}^n$$

$$P(\mathbf{x}) = \frac{1}{Z} e^{\beta \sum_{(ij)} J_{ij} x_i x_j + \sum_i H_i x_i}$$

$$Z = \sum_{\mathbf{x}} e^{\beta \sum_{(ij)} J_{ij} x_i x_j + \sum_i H_i x_i}$$

The simplest energy-based model



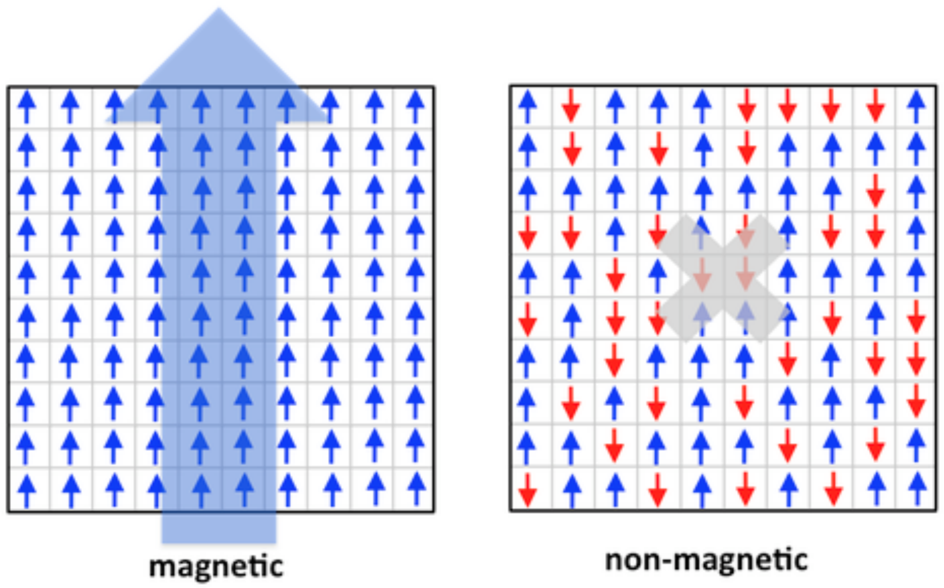
The Ising model (E. Ising, 1924)

$$\mathbf{x} = \{+1, -1\}^n$$

$$P(\mathbf{x}) = \frac{1}{Z} e^{\beta \sum_{(ij)} J_{ij} x_i x_j + \sum_i H_i x_i}$$

$$Z = \sum_{\mathbf{x}} e^{\beta \sum_{(ij)} J_{ij} x_i x_j + \sum_i H_i x_i}$$

The simplest energy-based model



The Ising model (E. Ising, 1924)

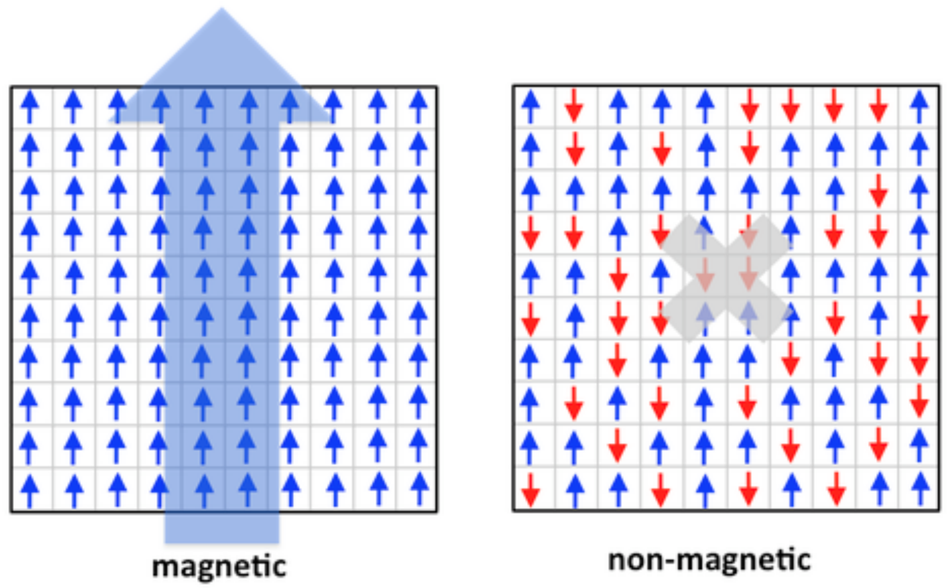
$$\mathbf{x} = \{+1, -1\}^n$$

$$P(\mathbf{x}) = \frac{1}{Z} e^{\beta \sum_{(ij)} J_{ij} x_i x_j + \sum_i H_i x_i}$$

$$Z = \sum_{\mathbf{x}} e^{\beta \sum_{(ij)} J_{ij} x_i x_j + \sum_i H_i x_i}$$

$J_{ij} = J$	Ferromagnet
$T = \frac{1}{\beta}$	Temperature
$M = \frac{1}{n} \sum_i x_i$	Magnetization

The simplest energy-based model

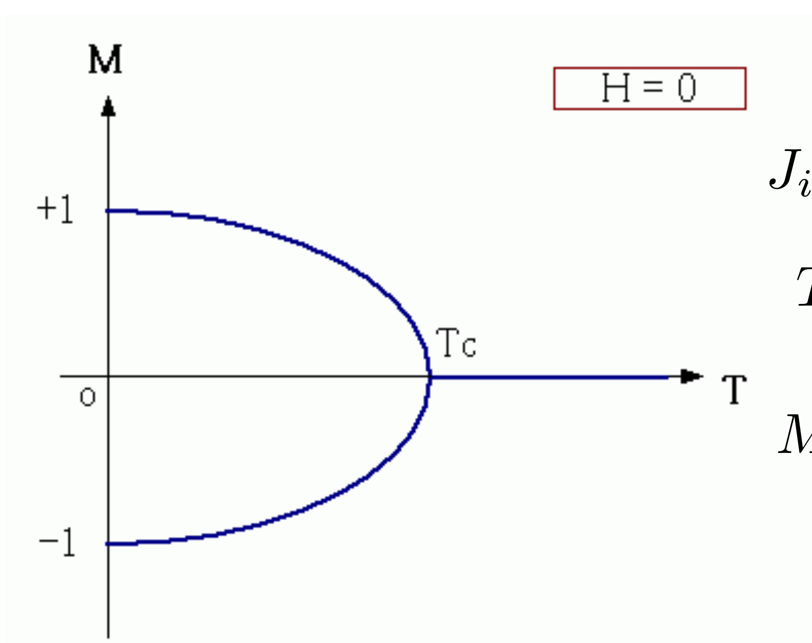


The Ising model (E. Ising, 1924)

$$\mathbf{x} = \{+1, -1\}^n$$

$$P(\mathbf{x}) = \frac{1}{Z} e^{\beta \sum_{(ij)} J_{ij} x_i x_j + \sum_i H_i x_i}$$

$$Z = \sum_{\mathbf{x}} e^{\beta \sum_{(ij)} J_{ij} x_i x_j + \sum_i H_i x_i}$$

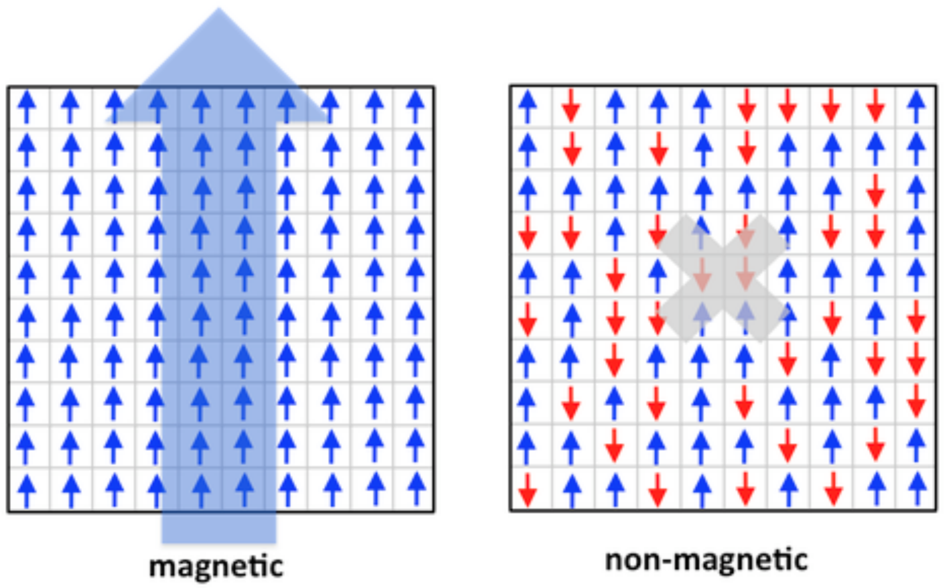


$J_{ij} = J$ Ferromagnet

$T = \frac{1}{\beta}$ Temperature

$M = \frac{1}{n} \sum_i x_i$ Magnetization

The simplest energy-based model

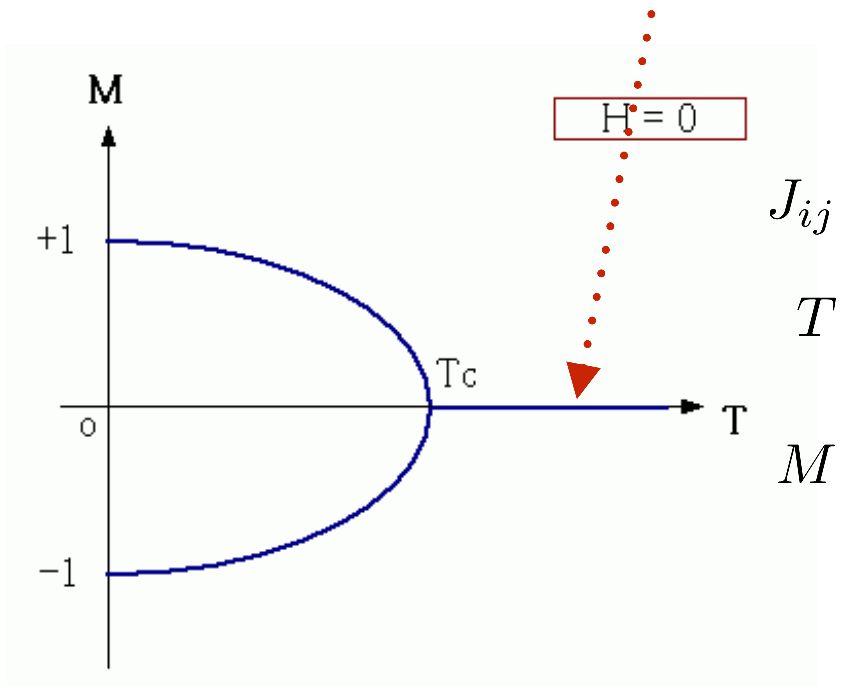


The Ising model (E. Ising, 1924)

$$\mathbf{x} = \{+1, -1\}^n$$

$$P(\mathbf{x}) = \frac{1}{Z} e^{\beta \sum_{(ij)} J_{ij} x_i x_j + \sum_i H_i x_i}$$

$$Z = \sum_{\mathbf{x}} e^{\beta \sum_{(ij)} J_{ij} x_i x_j + \sum_i H_i x_i}$$

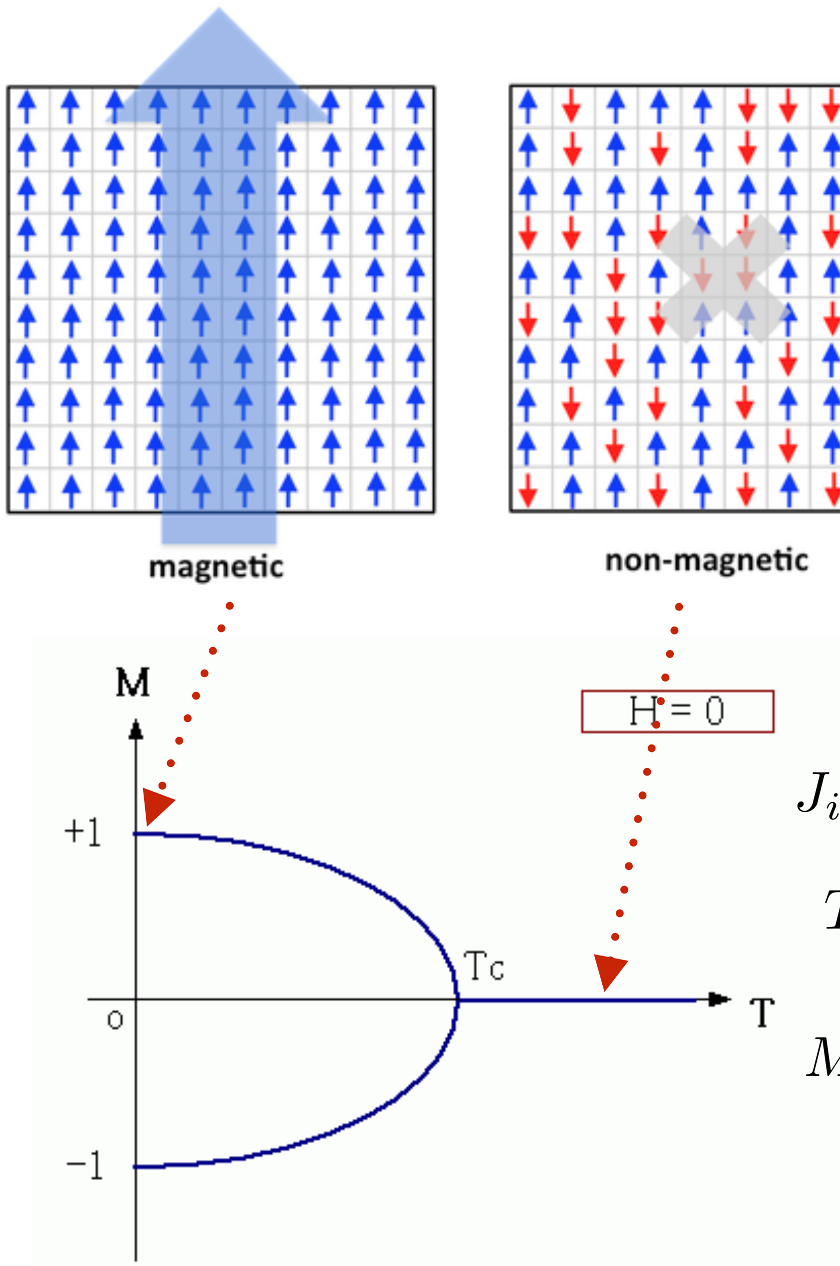


$J_{ij} = J$ Ferromagnet

$T = \frac{1}{\beta}$ Temperature

$M = \frac{1}{n} \sum_i x_i$ Magnetization

The simplest energy-based model



The Ising model (E. Ising, 1924)

$$\mathbf{x} = \{+1, -1\}^n$$

$$P(\mathbf{x}) = \frac{1}{Z} e^{\beta \sum_{(ij)} J_{ij} x_i x_j + \sum_i H_i x_i}$$

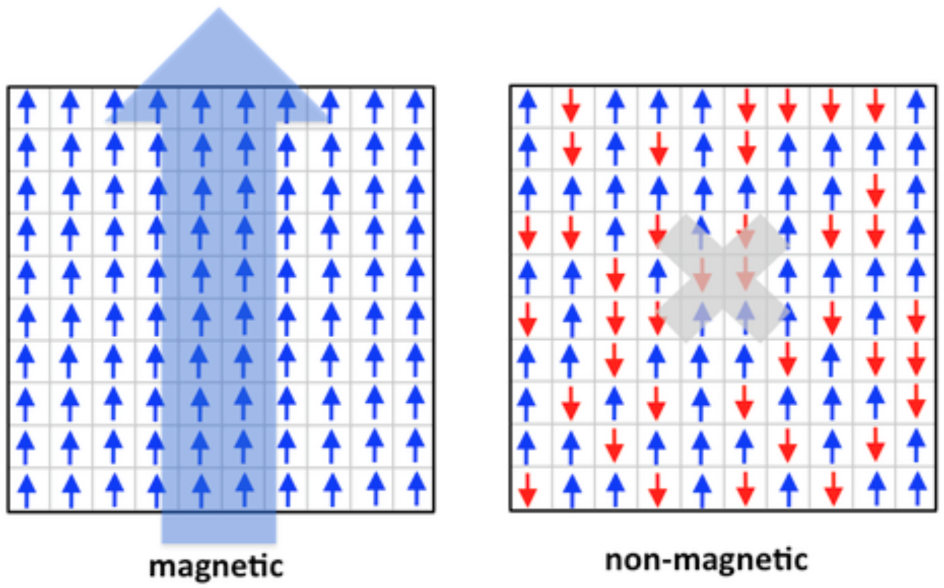
$$Z = \sum_{\mathbf{x}} e^{\beta \sum_{(ij)} J_{ij} x_i x_j + \sum_i H_i x_i}$$

$J_{ij} = J$ Ferromagnet

$T = \frac{1}{\beta}$ Temperature

$M = \frac{1}{n} \sum_i x_i$ Magnetization

The simplest energy-based model

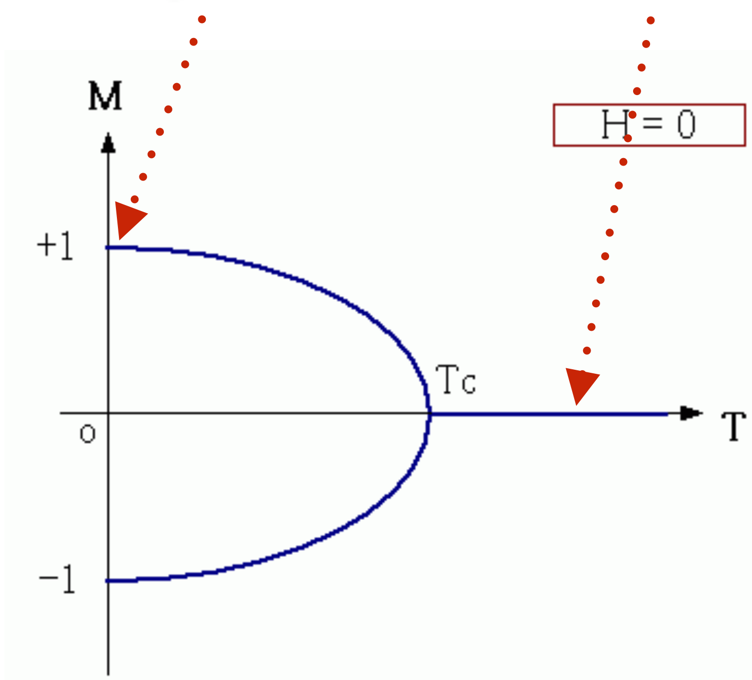


The Ising model (E. Ising, 1924)

$$\mathbf{x} = \{+1, -1\}^n$$

$$P(\mathbf{x}) = \frac{1}{Z} e^{\beta \sum_{(ij)} J_{ij} x_i x_j + \sum_i H_i x_i}$$

$$Z = \sum_{\mathbf{x}} e^{\beta \sum_{(ij)} J_{ij} x_i x_j + \sum_i H_i x_i}$$



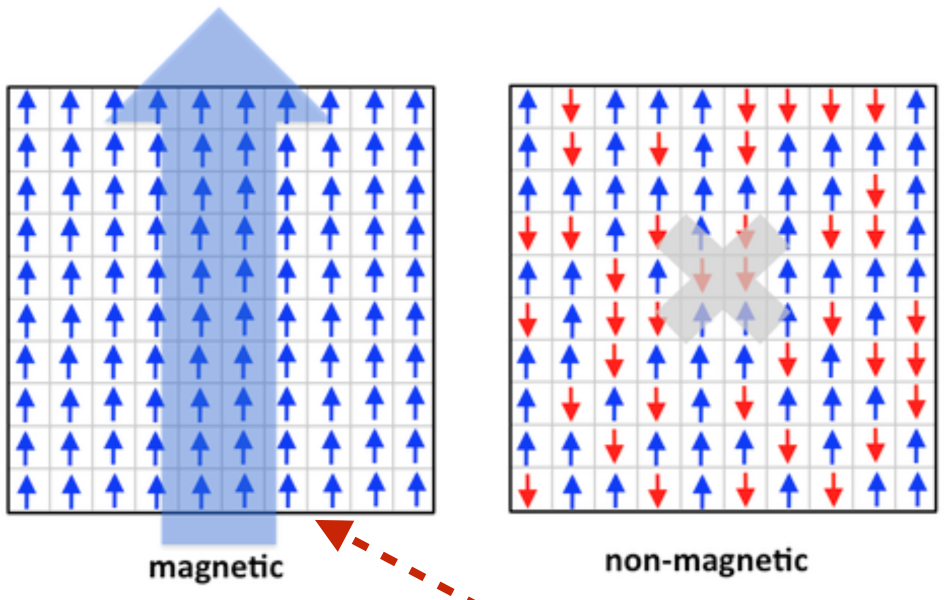
$J_{ij} = J$ Ferromagnet

$T = \frac{1}{\beta}$ Temperature

$M = \frac{1}{n} \sum_i x_i$ Magnetization

★ At a high temperature, the model does not remember any data,
★ At a low temperature, the model successfully remembers one data !

The simplest energy-based model

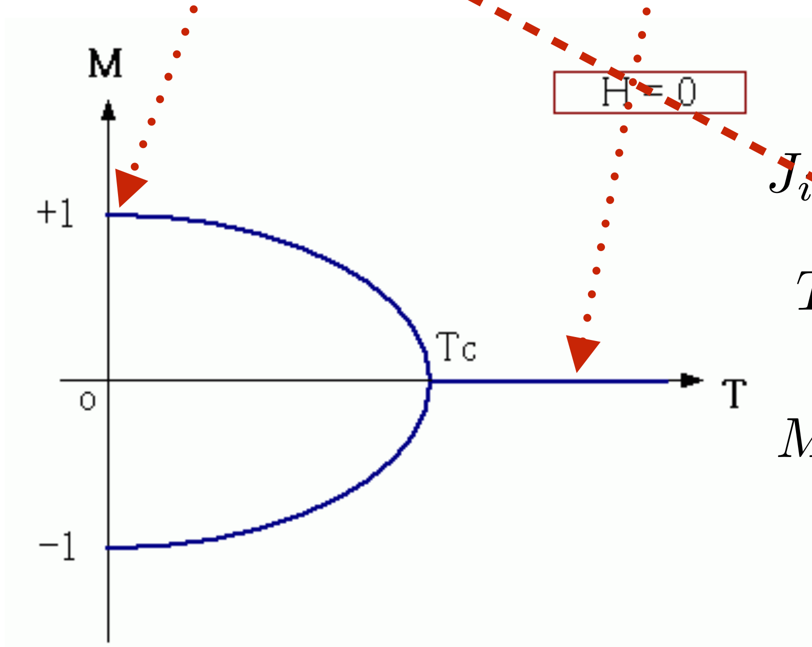


The Ising model (E. Ising, 1924)

$$\mathbf{x} = \{+1, -1\}^n$$

$$P(\mathbf{x}) = \frac{1}{Z} e^{\beta \sum_{(ij)} J_{ij} x_i x_j + \sum_i H_i x_i}$$

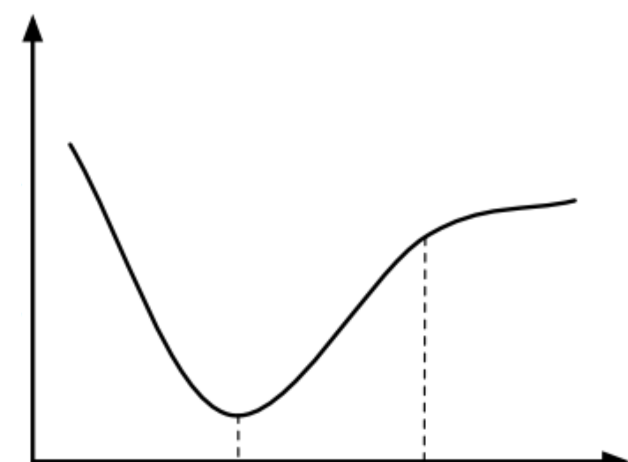
$$Z = \sum_{\mathbf{x}} e^{\beta \sum_{(ij)} J_{ij} x_i x_j + \sum_i H_i x_i}$$



$J_{ij} = J$ Ferromagnet

$T = \frac{1}{\beta}$ Temperature

$M = \frac{1}{n} \sum_i x_i$ Magnetization



★ At a high temperature, the model does not remember any data,
★ At a low temperature, the model successfully remembers one data !

Remember more pattern: The Hopfield model

The Hopfield model (Hofield, 1982, Amari 1977, Little 1974)

Remember more pattern: The Hopfield model

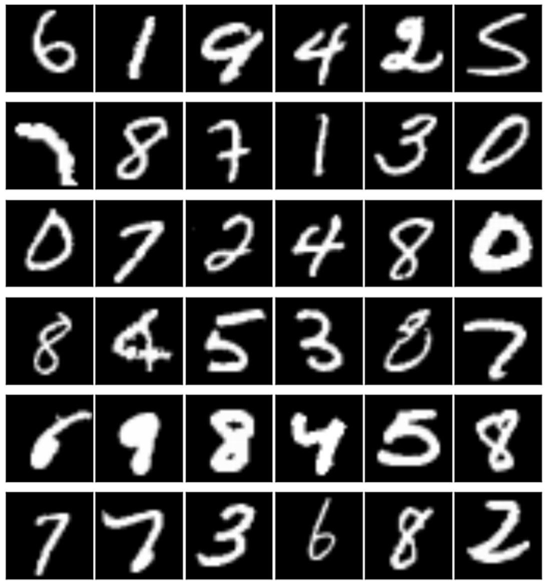
The Hopfield model (Hofield, 1982, Amari 1977, Little 1974)

$$\mathbf{x} = \{+1, -1\}^n$$

$$P(\mathbf{x}) = \frac{1}{Z} e^{\beta \sum_{(ij)} J_{ij} x_i x_j + \sum_i H_i x_i}$$

$$Z = \sum_{\mathbf{x}} e^{\beta \sum_{(ij)} J_{ij} x_i x_j + \sum_i H_i x_i}$$

Remember more pattern: The Hopfield model



The Hopfield model (Hofield, 1982, Amari 1977, Little 1974)

$$\mathbf{x} = \{+1, -1\}^n$$

$$P(\mathbf{x}) = \frac{1}{Z} e^{\beta \sum_{(ij)} J_{ij} x_i x_j + \sum_i H_i x_i}$$

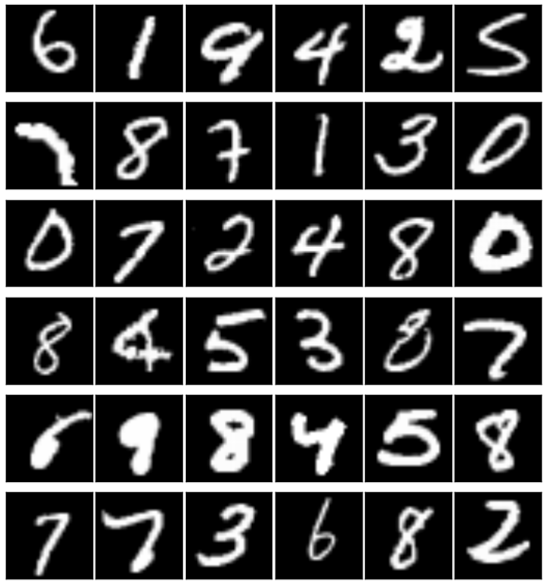
$$Z = \sum_{\mathbf{x}} e^{\beta \sum_{(ij)} J_{ij} x_i x_j + \sum_i H_i x_i}$$

$$\{\xi_i^\mu\} \in \{+1, -1\}^{\alpha n \times n}$$

$$J_{ij} = \frac{1}{n} \sum_{\mu=1}^{\alpha n} \xi_i^\mu \xi_j^\mu$$

Hebb's learning rule
Hebb 1949

Remember more pattern: The Hopfield model



The Hopfield model (Hofield, 1982, Amari 1977, Little 1974)

$$\mathbf{x} = \{+1, -1\}^n$$

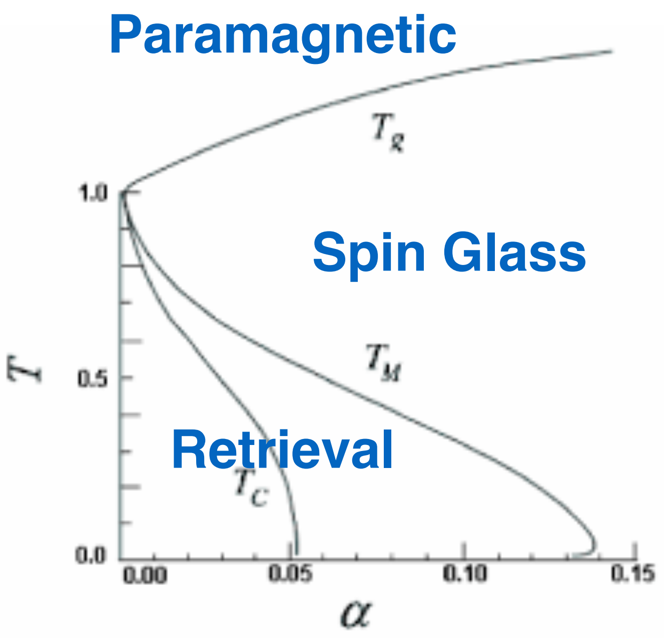
$$P(\mathbf{x}) = \frac{1}{Z} e^{\beta \sum_{(ij)} J_{ij} x_i x_j + \sum_i H_i x_i}$$

$$Z = \sum_{\mathbf{x}} e^{\beta \sum_{(ij)} J_{ij} x_i x_j + \sum_i H_i x_i}$$

$$\{\xi_i^\mu\} \in \{+1, -1\}^{\alpha n \times n}$$

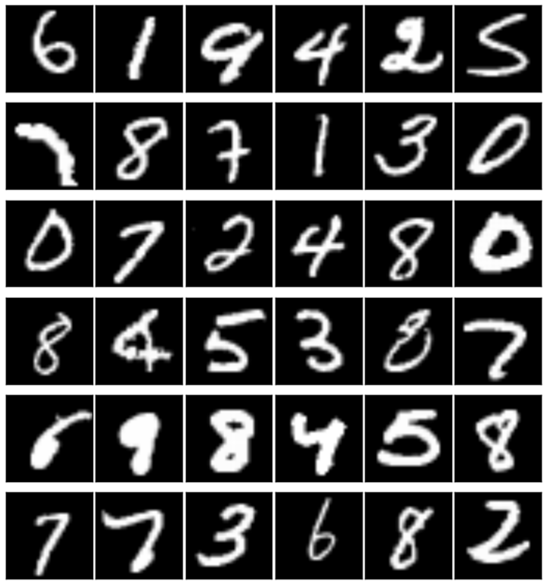
$$J_{ij} = \frac{1}{n} \sum_{\mu=1}^{\alpha n} \xi_i^\mu \xi_j^\mu$$

Hebb's learning rule
Hebb 1949



Phase diagram
Amit et al 1985

Remember more pattern: The Hopfield model



The Hopfield model (Hofield, 1982, Amari 1977, Little 1974)

$$\mathbf{x} = \{+1, -1\}^n$$

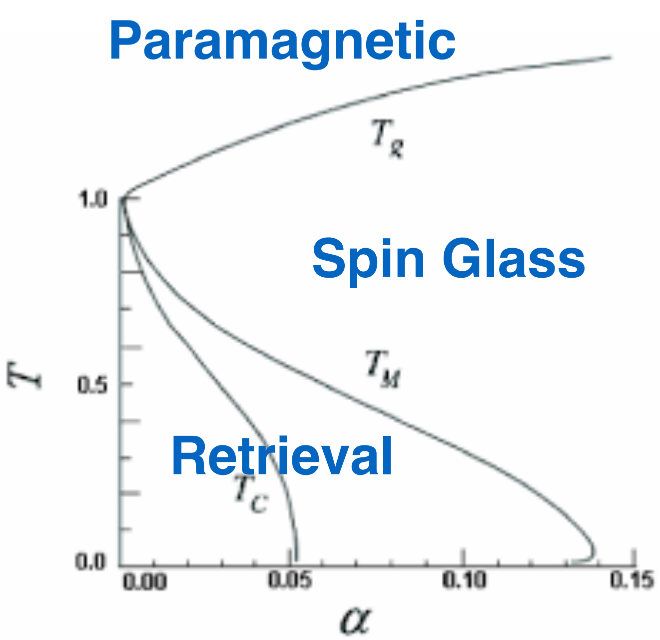
$$P(\mathbf{x}) = \frac{1}{Z} e^{\beta \sum_{(ij)} J_{ij} x_i x_j + \sum_i H_i x_i}$$

$$Z = \sum_{\mathbf{x}} e^{\beta \sum_{(ij)} J_{ij} x_i x_j + \sum_i H_i x_i}$$

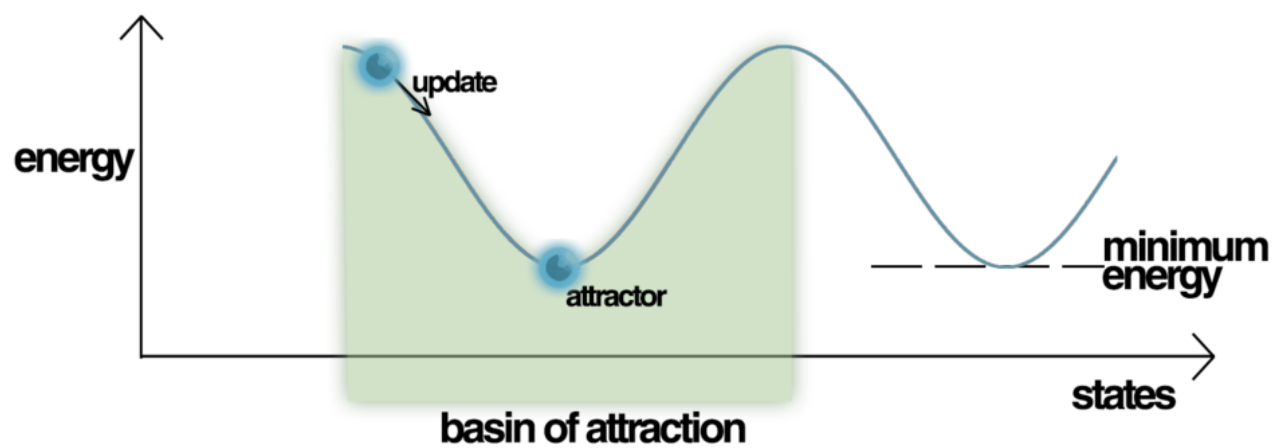
$$\{\xi_i^\mu\} \in \{+1, -1\}^{\alpha n \times n}$$

$$J_{ij} = \frac{1}{n} \sum_{\mu=1}^{\alpha n} \xi_i^\mu \xi_j^\mu$$

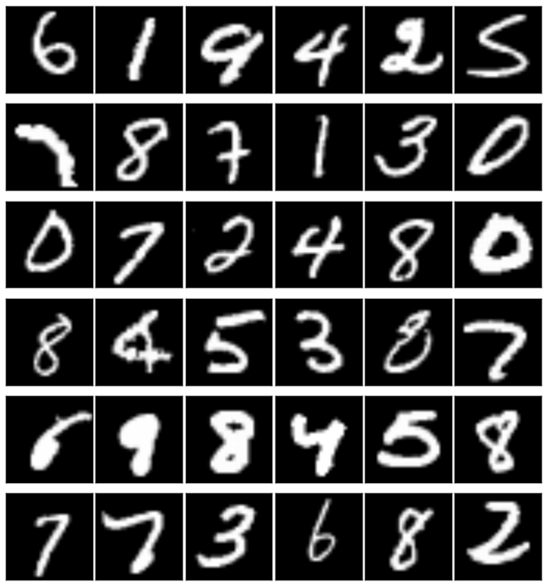
Hebb's learning rule
Hebb 1949



Phase diagram
Amit et al 1985



Remember more pattern: The Hopfield model



The Hopfield model (Hofield, 1982, Amari 1977, Little 1974)

$$\mathbf{x} = \{+1, -1\}^n$$

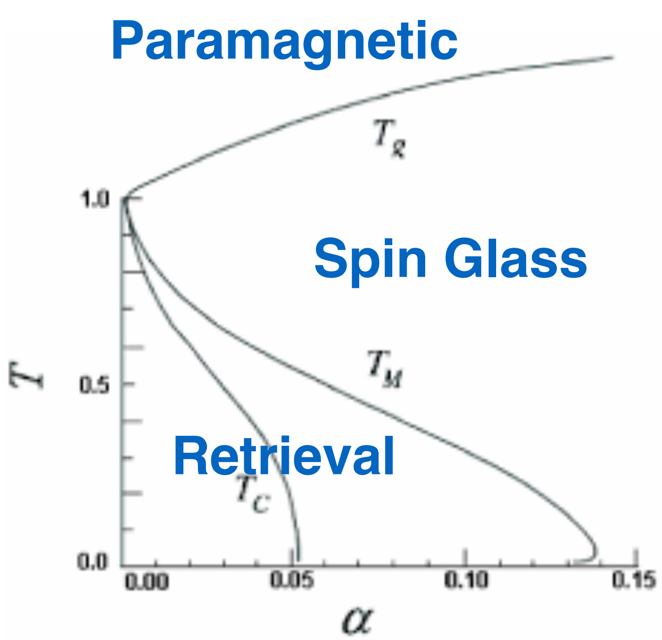
$$P(\mathbf{x}) = \frac{1}{Z} e^{\beta \sum_{(ij)} J_{ij} x_i x_j + \sum_i H_i x_i}$$

$$Z = \sum_{\mathbf{x}} e^{\beta \sum_{(ij)} J_{ij} x_i x_j + \sum_i H_i x_i}$$

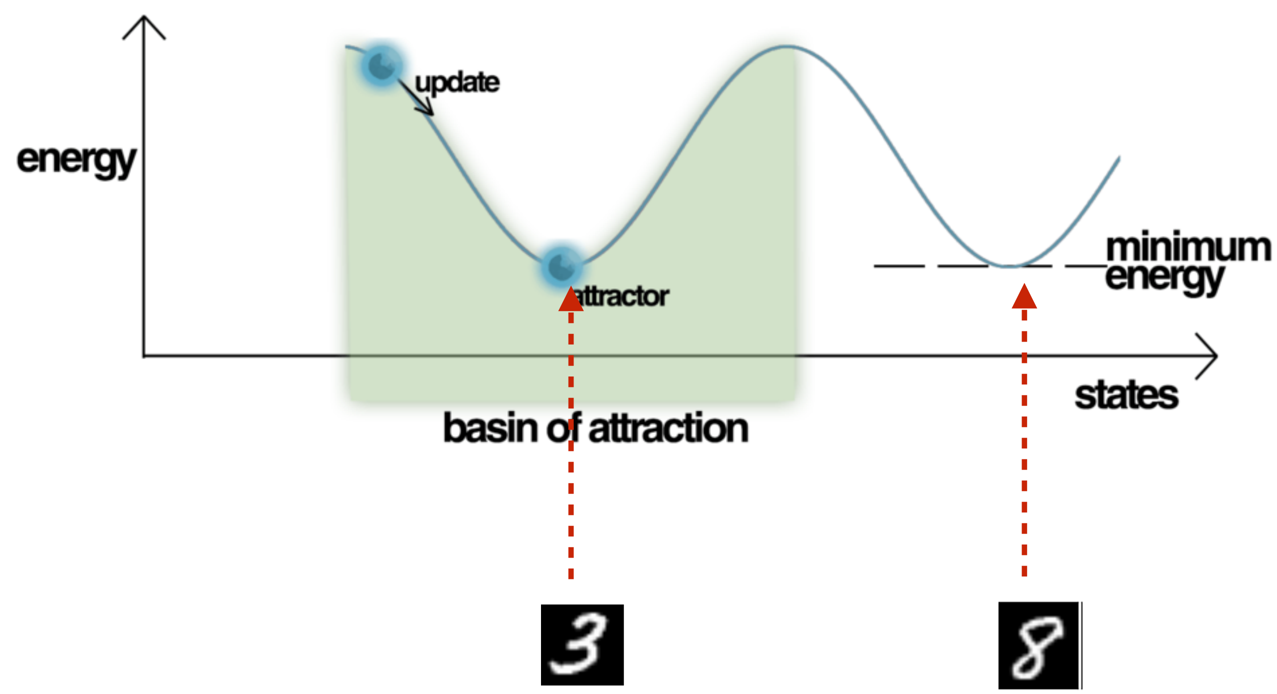
$$\{\xi_i^\mu\} \in \{+1, -1\}^{\alpha n \times n}$$

$$J_{ij} = \frac{1}{n} \sum_{\mu=1}^{\alpha n} \xi_i^\mu \xi_j^\mu$$

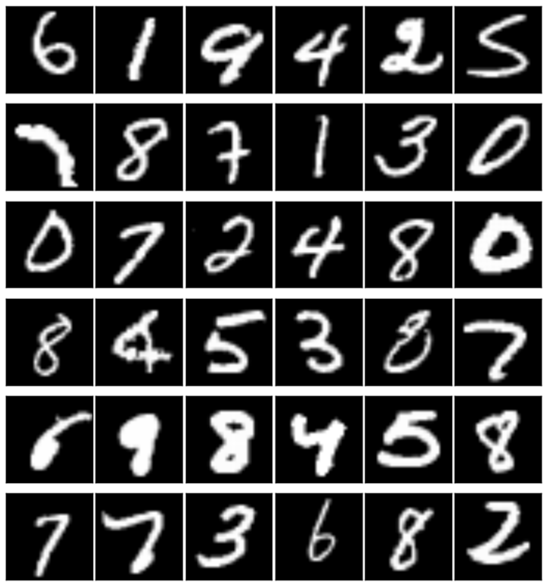
Hebb's learning rule
Hebb 1949



Phase diagram
Amit et al 1985



Remember more pattern: The Hopfield model



The Hopfield model (Hofield, 1982, Amari 1977, Little 1974)

$$\mathbf{x} = \{+1, -1\}^n$$

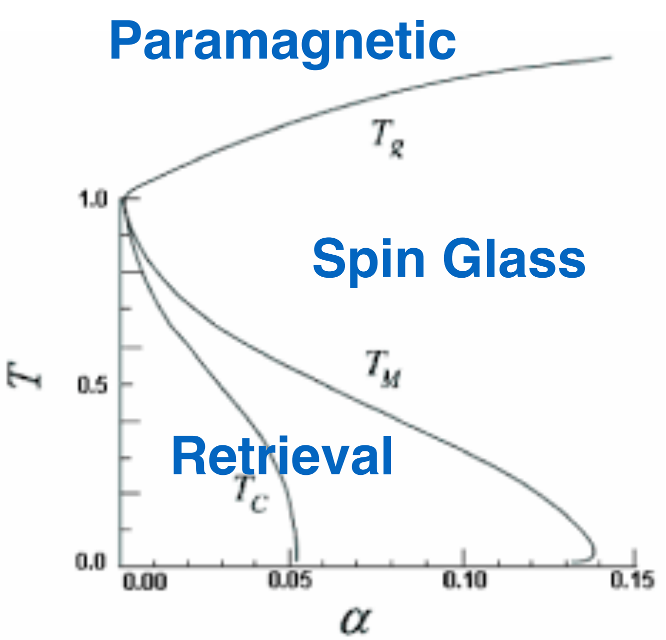
$$P(\mathbf{x}) = \frac{1}{Z} e^{\beta \sum_{(ij)} J_{ij} x_i x_j + \sum_i H_i x_i}$$

$$Z = \sum_{\mathbf{x}} e^{\beta \sum_{(ij)} J_{ij} x_i x_j + \sum_i H_i x_i}$$

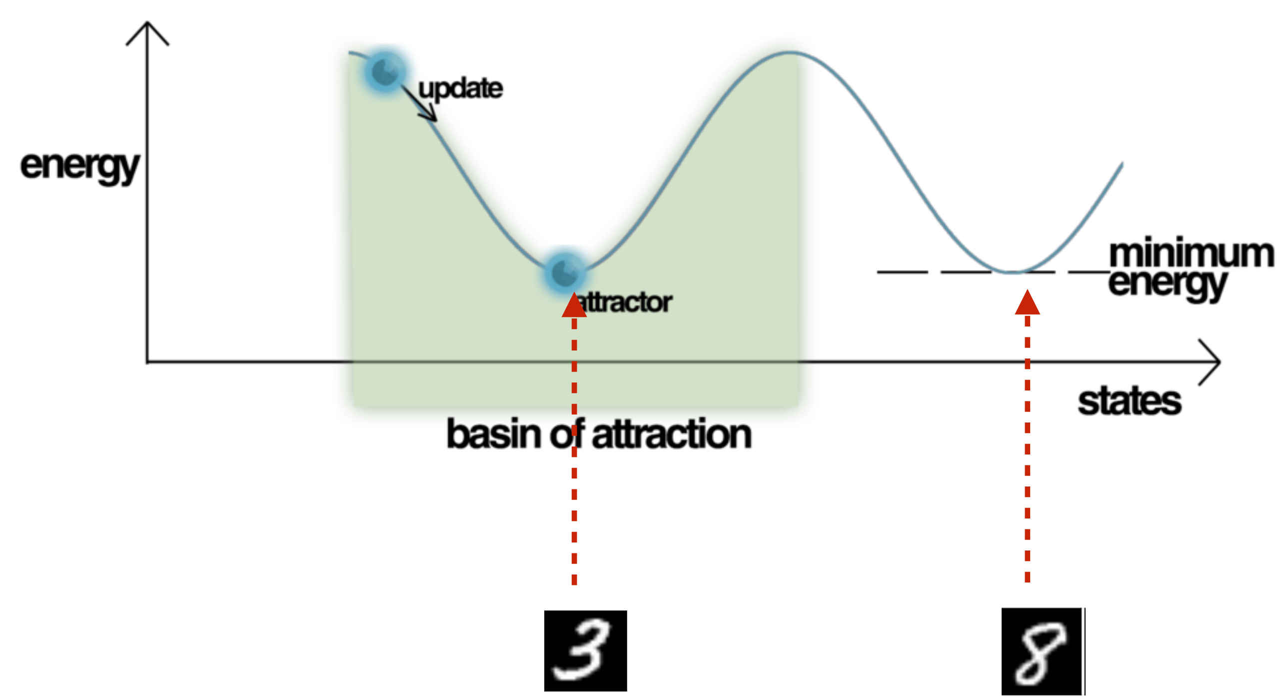
$$\{\xi_i^\mu\} \in \{+1, -1\}^{\alpha n \times n}$$

$$J_{ij} = \frac{1}{n} \sum_{\mu=1}^{\alpha n} \xi_i^\mu \xi_j^\mu$$

Hebb's learning rule
Hebb 1949

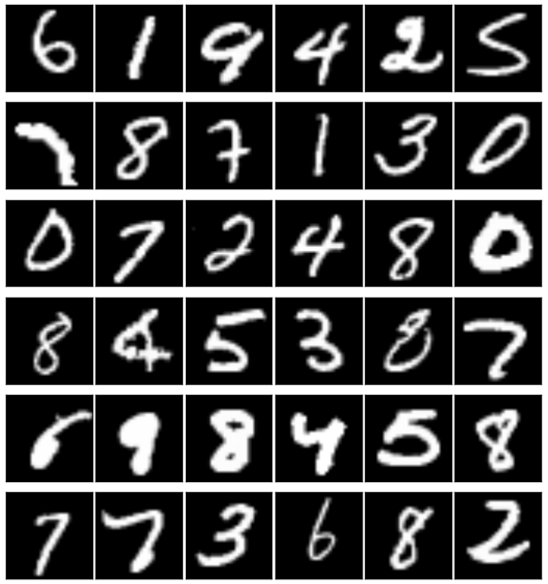


Phase diagram
Amit et al 1985

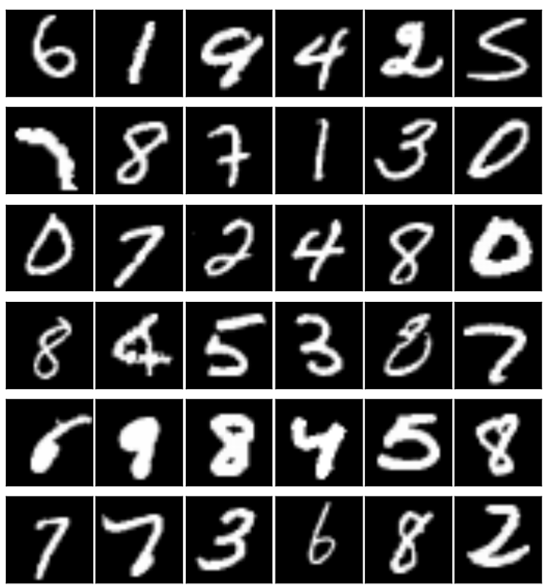


Drawback: orthogonality, linear capacity

The inverse Ising model

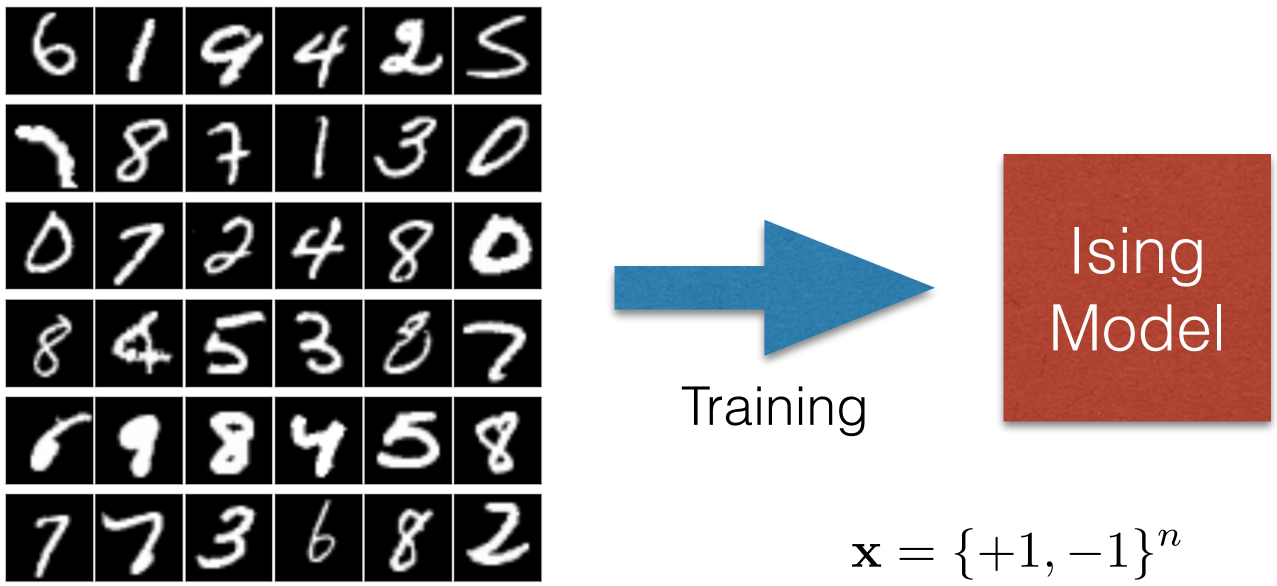


The inverse Ising model



Training

The inverse Ising model

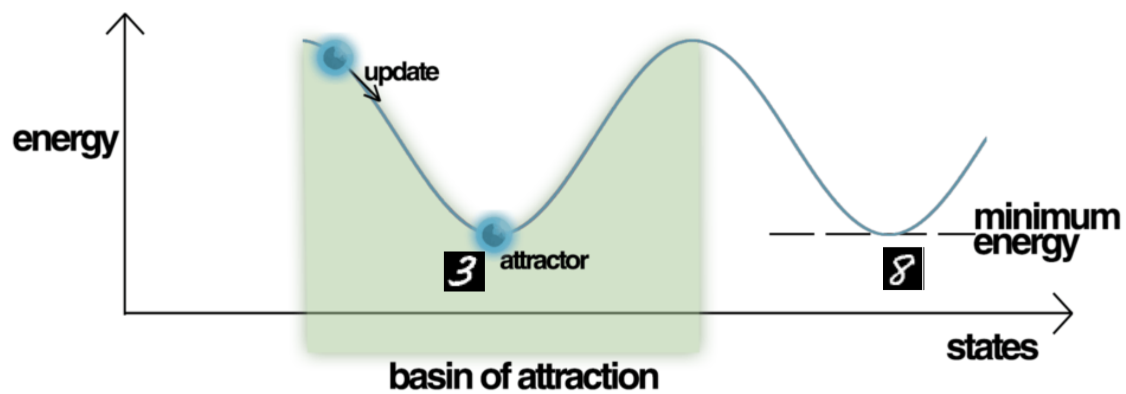


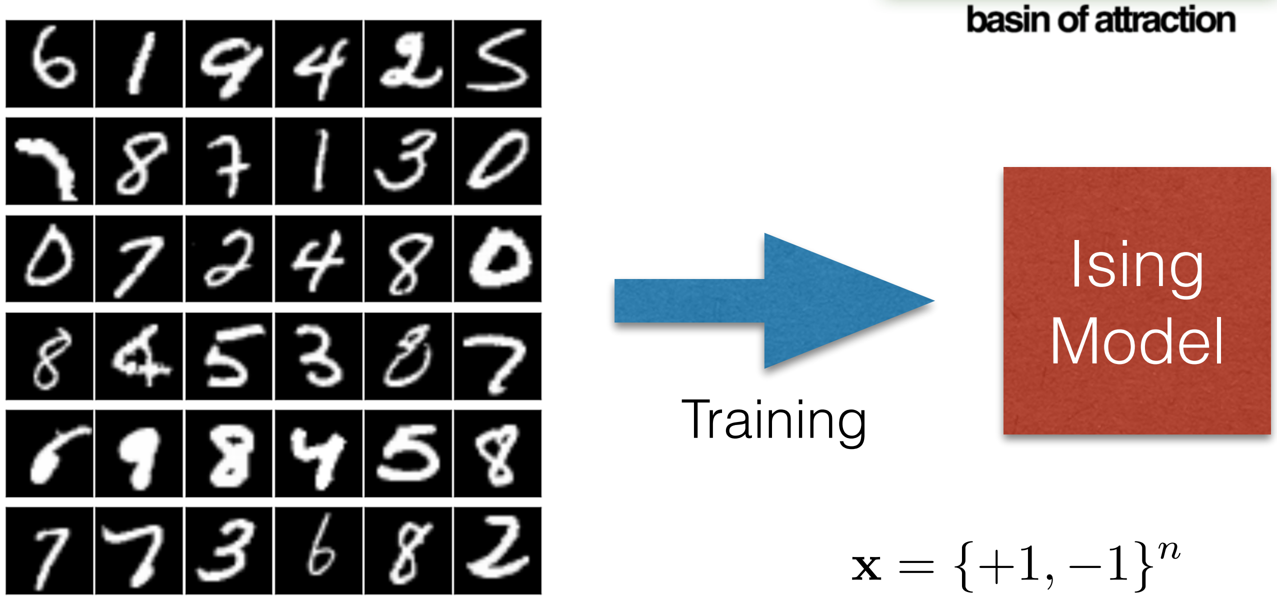
$$\mathbf{x} = \{+1, -1\}^n$$

$$P(\mathbf{x}) = \frac{1}{Z} e^{\beta \sum_{(ij)} J_{ij} x_i x_j + \sum_i H_i x_i}$$

$$Z = \sum_{\mathbf{x}} e^{\beta \sum_{(ij)} J_{ij} x_i x_j + \sum_i H_i x_i}$$

The inverse Ising model



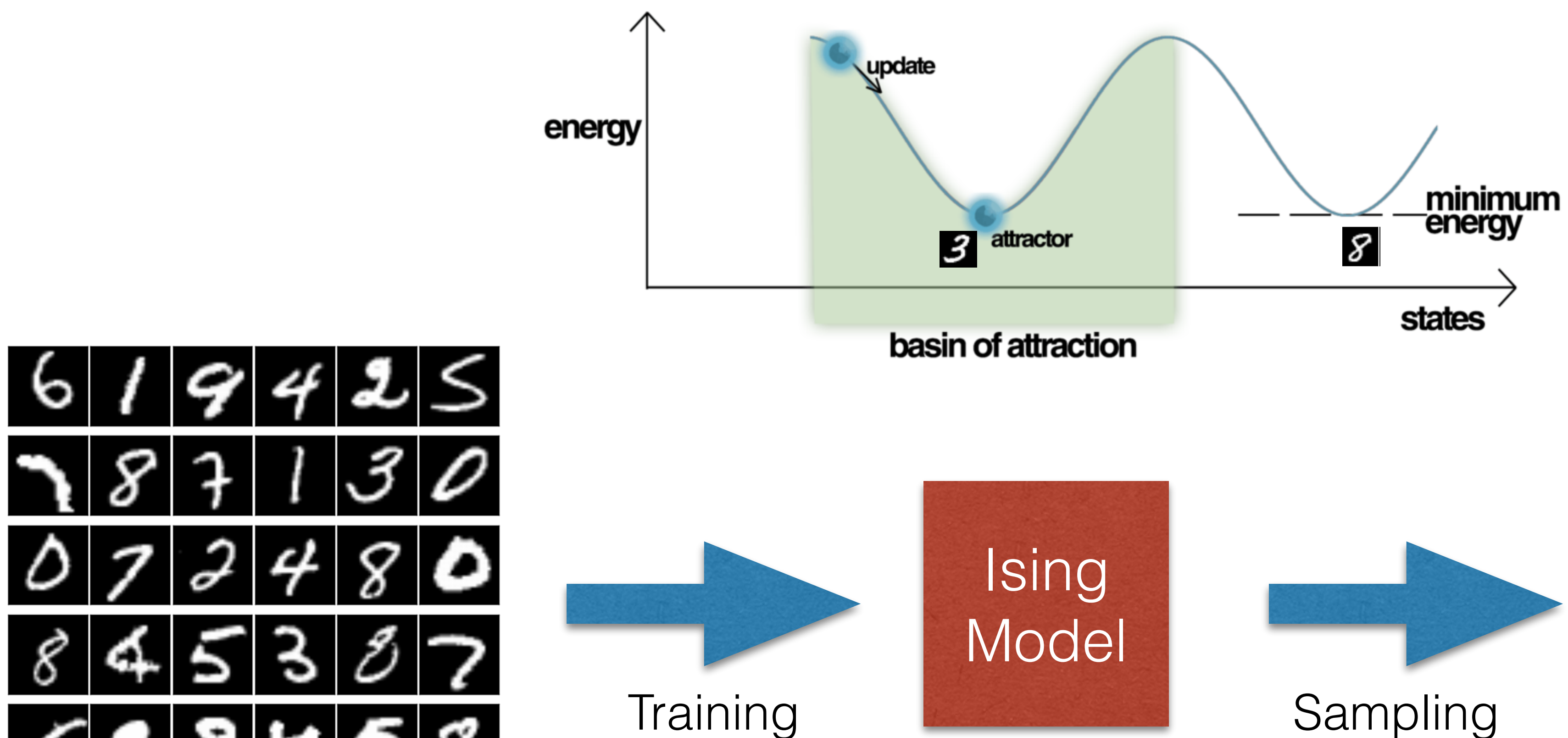


$$\mathbf{x} = \{+1, -1\}^n$$

$$P(\mathbf{x}) = \frac{1}{Z} e^{\beta \sum_{(ij)} J_{ij} x_i x_j + \sum_i H_i x_i}$$

$$Z = \sum_{\mathbf{x}} e^{\beta \sum_{(ij)} J_{ij} x_i x_j + \sum_i H_i x_i}$$

The inverse Ising model

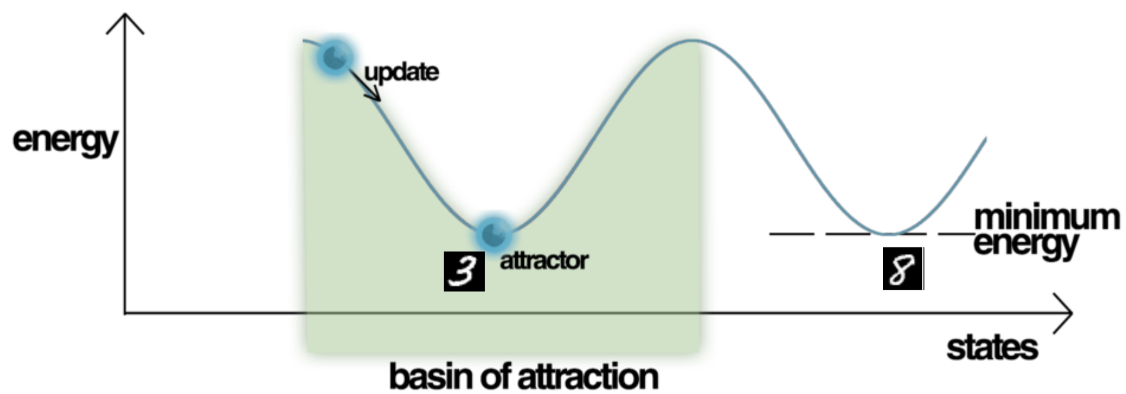


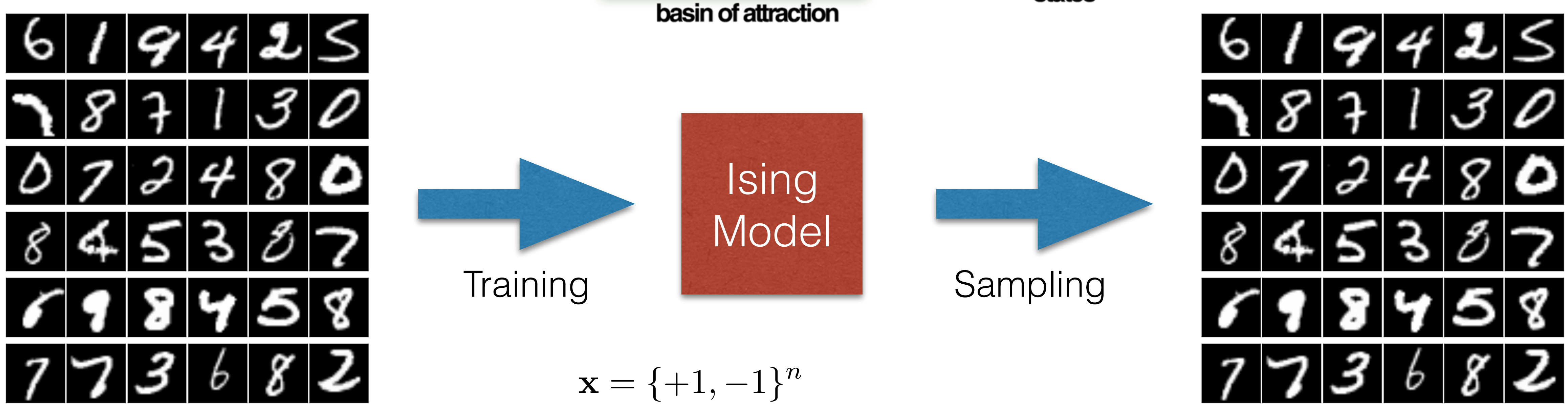
$$\mathbf{x} = \{+1, -1\}^n$$

$$P(\mathbf{x}) = \frac{1}{Z} e^{\beta \sum_{(ij)} J_{ij} x_i x_j + \sum_i H_i x_i}$$

$$Z = \sum_{\mathbf{x}} e^{\beta \sum_{(ij)} J_{ij} x_i x_j + \sum_i H_i x_i}$$

The inverse Ising model





$$\mathbf{x} = \{+1, -1\}^n$$

$$P(\mathbf{x}) = \frac{1}{Z} e^{\beta \sum_{(ij)} J_{ij} x_i x_j + \sum_i H_i x_i}$$

$$Z = \sum_{\mathbf{x}} e^{\beta \sum_{(ij)} J_{ij} x_i x_j + \sum_i H_i x_i}$$

AN INTRODUCTION TO LEARNING AND GENERALISATION

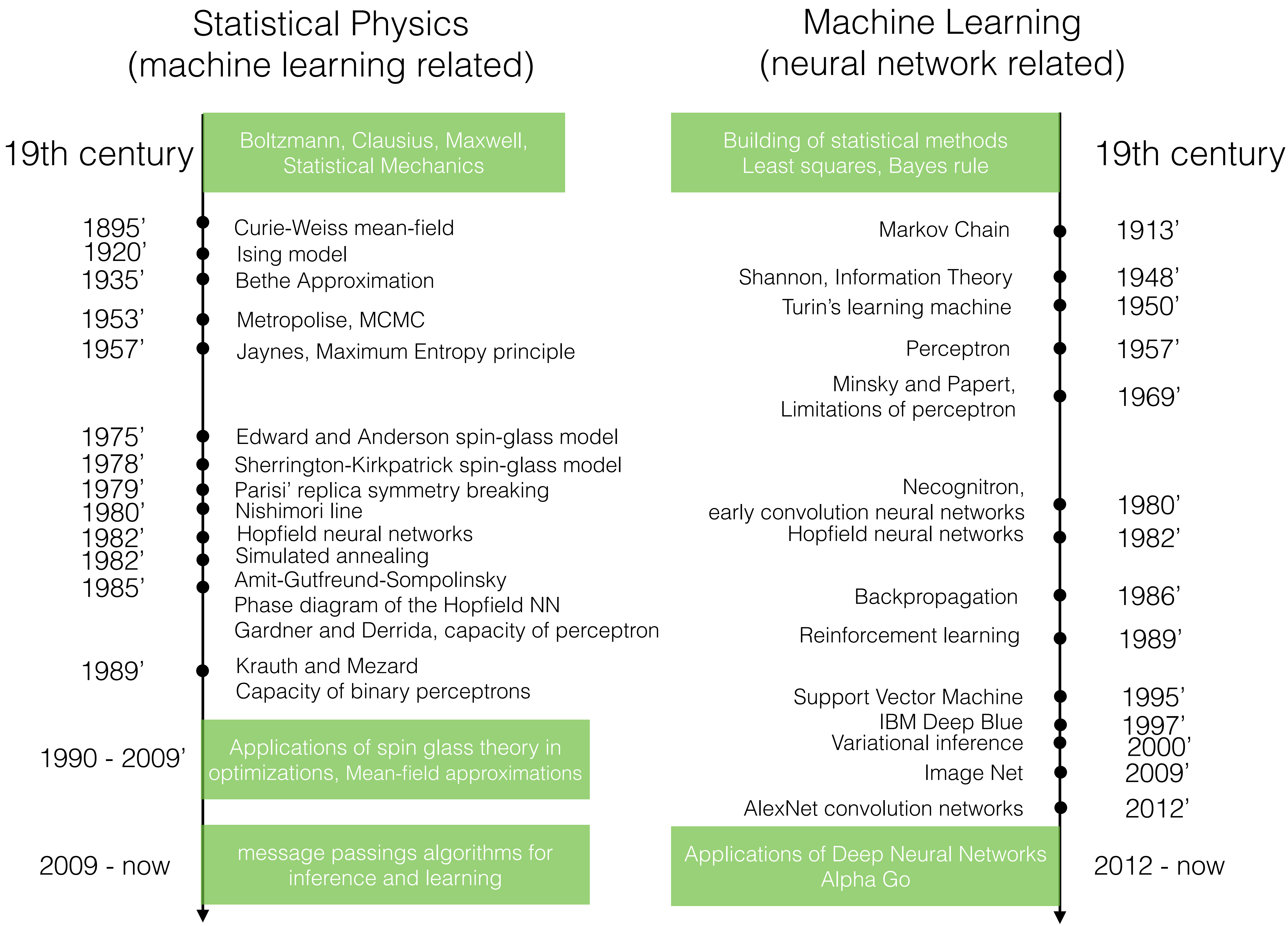
Giorgio Parisi
Dipartimento di Fisica
Piazzale delle Scienze
Roma Italy 00185

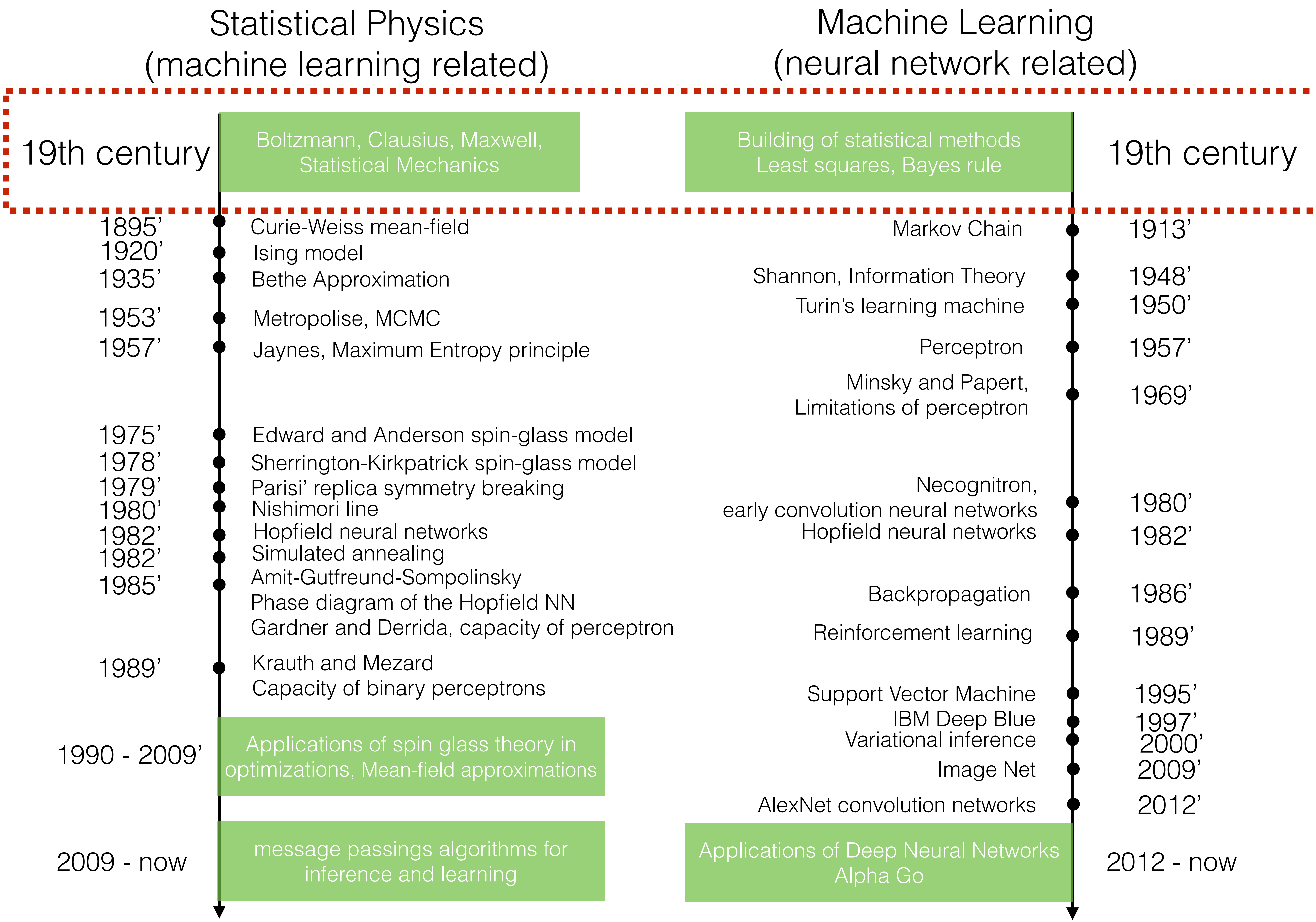
ABSTRACT. In this lecture I will present some basic ideas on how computers may learn rules from examples and how generalisation may be achieved. The general prospective is presented. Some comments are also done on the definition of intelligence.

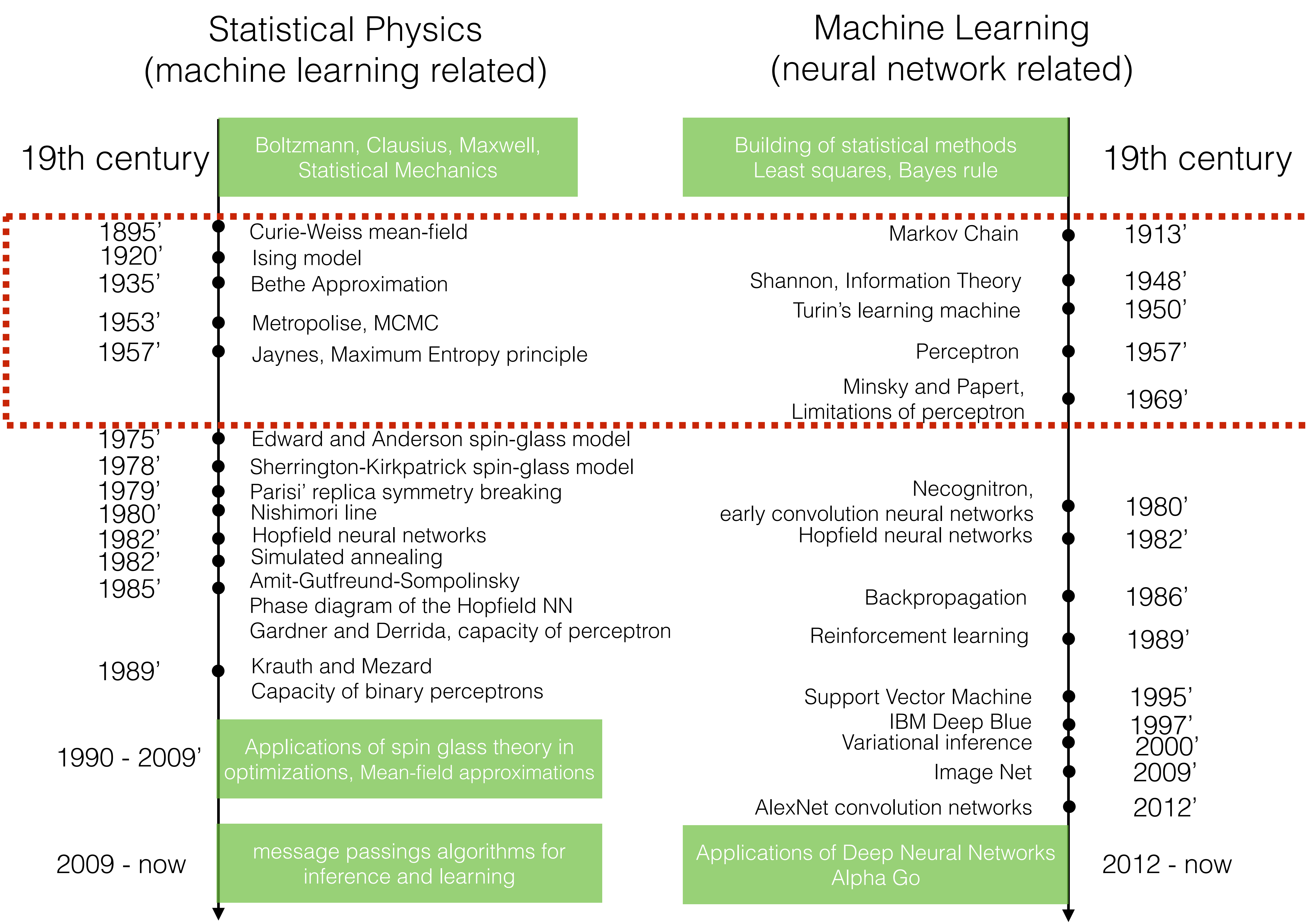
Learning – Generalisation – Intelligence

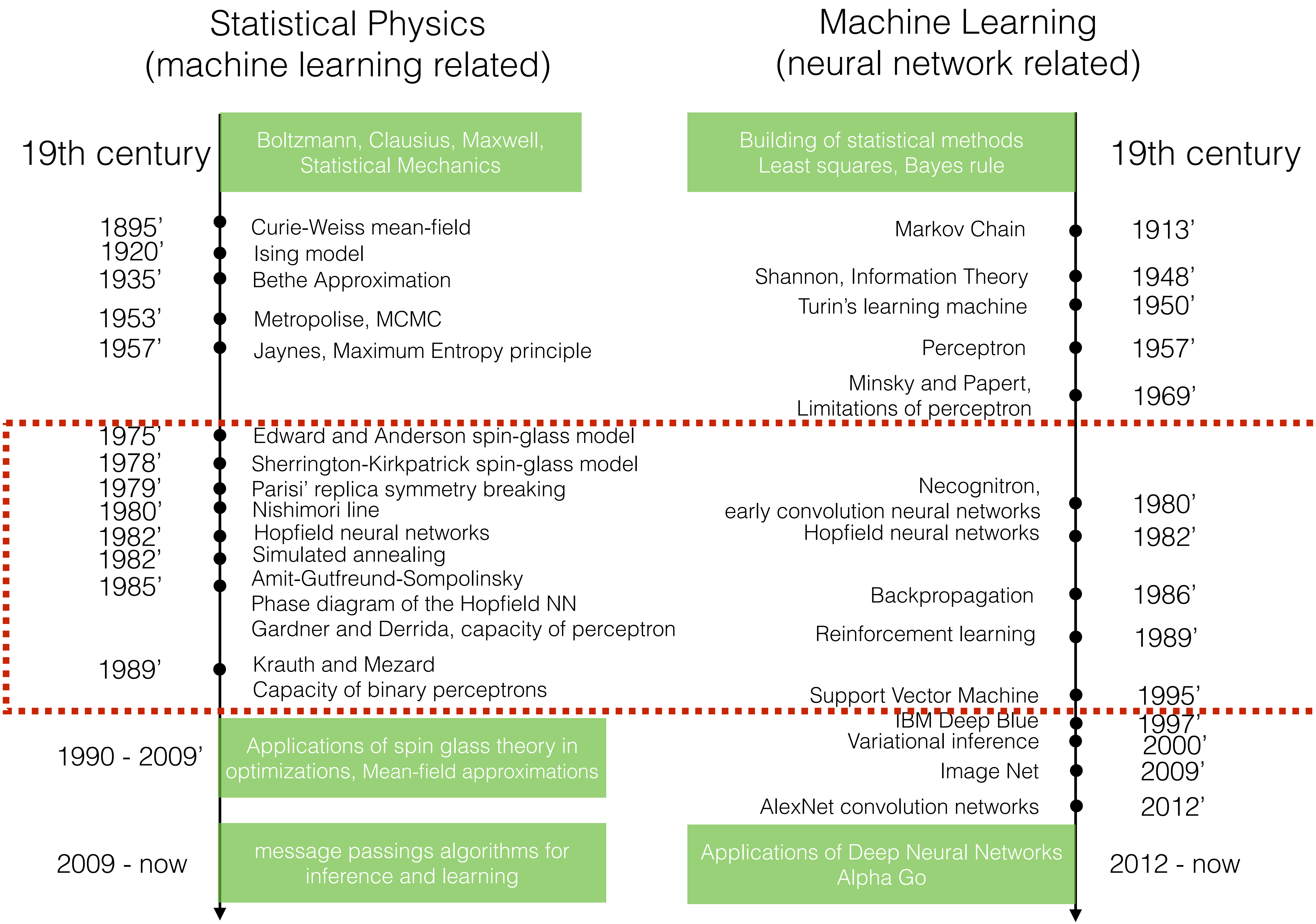
Giorgio Parisi 1992'

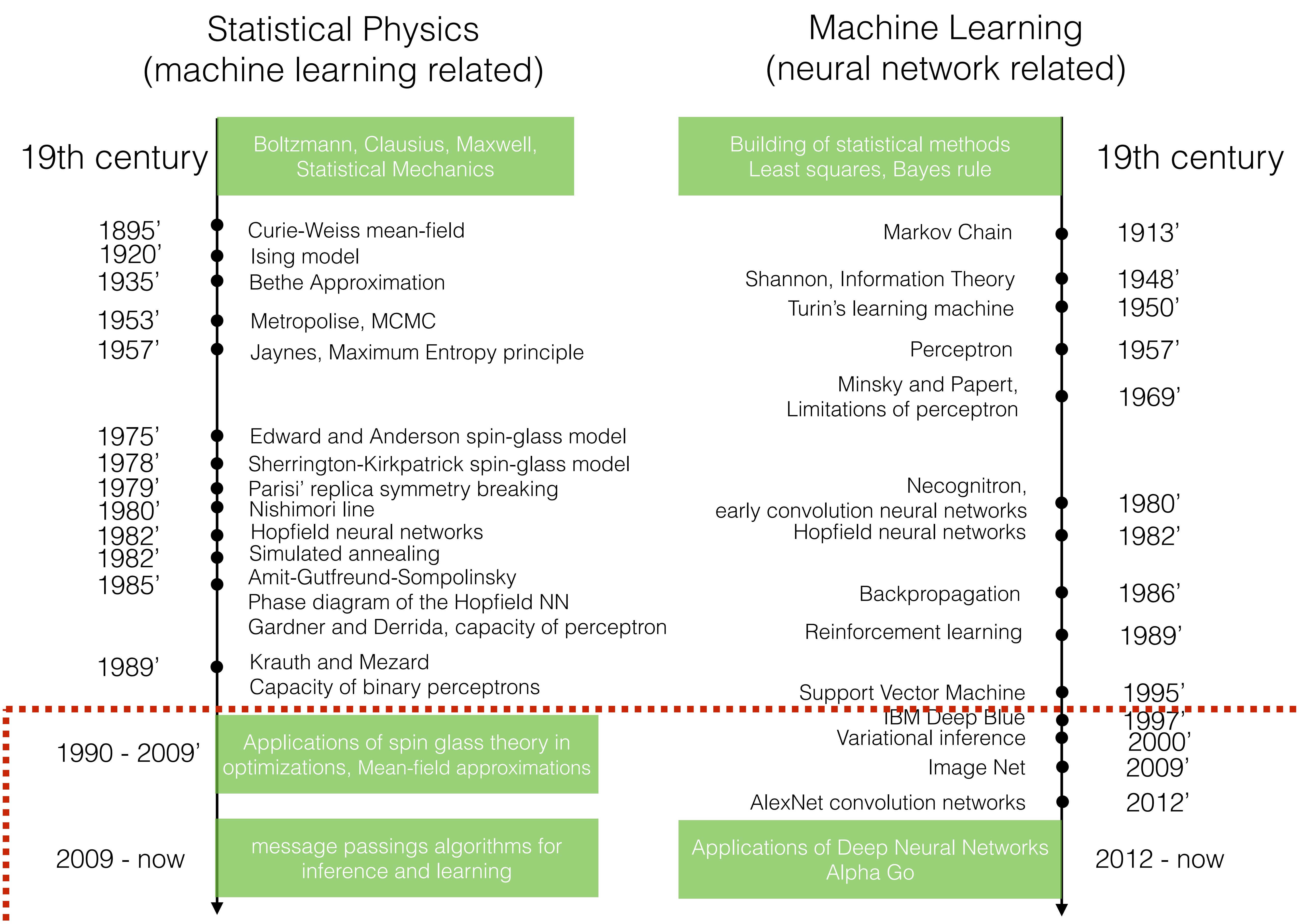
Boltzmann Medal, Lars Onsager Medal, Dirac Medal


Statistical Physics
(machine learning related)
19th century
Boltzmann, Clausius, Maxwell,
Statistical Mechanics
1895' Curie-Weiss mean-field
1920' Ising model
1935' Bethe Approximation
1953' Metropolise, MCMC
1957' Jaynes, Maximum Entropy principle
1975' Edward and Anderson spin-glass model
1978' Sherrington-Kirkpatrick spin-glass model
1979' Parisi' replica symmetry breaking
1980' Nishimori line
1982' Hopfield neural networks
1982' Simulated annealing
1985' Amit-Gutfreund-Sompolinsky
Phase diagram of the Hopfield NN
Gardner and Derrida, capacity of perceptron
1989' Krauth and Mezard
Capacity of binary perceptrons
1990 - 2009' Applications of spin glass theory in optimizations, Mean-field approximations
2009 - now message passings algorithms for inference and learning
Machine Learning
(neural network related)
Building of statistical methods
Least squares, Bayes rule
19th century
Markov Chain 1913'
Shannon, Information Theory 1948'
Turin's learning machine 1950'
Perceptron 1957'
Minsky and Papert,
Limitations of perceptron 1969'
Necognitron,
early convolution neural networks 1980'
Hopfield neural networks 1982'
Backpropagation 1986'
Reinforcement learning 1989'
Support Vector Machine 1995'
IBM Deep Blue 1997'
Variational inference 2000'
Image Net 2009'
AlexNet convolution networks 2012'
Applications of Deep Neural Networks
Alpha Go
2012 - now

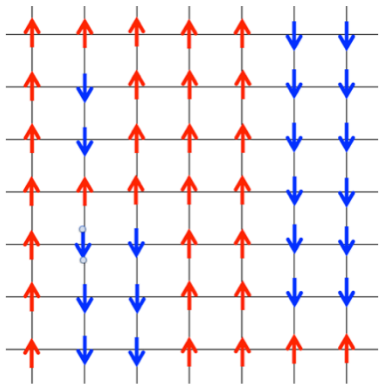

Statistical Physics
(machine learning related)
Machine Learning
(neural network related)
19th century
Boltzmann, Clausius, Maxwell,
Statistical Mechanics
Building of statistical methods
Least squares, Bayes rule
19th century
1895' Curie-Weiss mean-field
1920' Ising model
1935' Bethe Approximation
1953' Metropolise, MCMC
1957' Jaynes, Maximum Entropy principle
1975' Edward and Anderson spin-glass model
1978' Sherrington-Kirkpatrick spin-glass model
1979' Parisi' replica symmetry breaking
1980' Nishimori line
1982' Hopfield neural networks
1982' Simulated annealing
1985' Amit-Gutfreund-Sompolinsky
Phase diagram of the Hopfield NN
Gardner and Derrida, capacity of perceptron
1989' Krauth and Mezard
Capacity of binary perceptrons
1990 - 2009'
Applications of spin glass theory in
optimizations, Mean-field approximations
2009 - now
message passings algorithms for
inference and learning
Markov Chain 1913'
Shannon, Information Theory 1948'
Turin's learning machine 1950'
Perceptron 1957'
Minsky and Papert,
Limitations of perceptron 1969'
Necognitron,
early convolution neural networks 1980'
Hopfield neural networks 1982'
Backpropagation 1986'
Reinforcement learning 1989'
Support Vector Machine 1995'
IBM Deep Blue 1997'
Variational inference 2000'
Image Net 2009'
AlexNet convolution networks 2012'
Applications of Deep Neural Networks
Alpha Go
2012 - now


Statistical Physics
(machine learning related)
Machine Learning
(neural network related)
19th century
Boltzmann, Clausius, Maxwell,
Statistical Mechanics
Building of statistical methods
Least squares, Bayes rule
19th century
1895' Curie-Weiss mean-field
1920' Ising model
1935' Bethe Approximation
1953' Metropolise, MCMC
1957' Jaynes, Maximum Entropy principle
Markov Chain 1913'
Shannon, Information Theory 1948'
Turin's learning machine 1950'
Perceptron 1957'
Minsky and Papert,
Limitations of perceptron 1969'
1975' Edward and Anderson spin-glass model
1978' Sherrington-Kirkpatrick spin-glass model
1979' Parisi' replica symmetry breaking
1980' Nishimori line
1982' Hopfield neural networks
1982' Simulated annealing
1985' Amit-Gutfreund-Sompolinsky
Phase diagram of the Hopfield NN
Gardner and Derrida, capacity of perceptron
1989' Krauth and Mezard
Capacity of binary perceptrons
1990 - 2009'
Applications of spin glass theory in
optimizations, Mean-field approximations
2009 - now
message passings algorithms for
inference and learning
Necognitron,
early convolution neural networks 1980'
Hopfield neural networks 1982'
Backpropagation 1986'
Reinforcement learning 1989'
Support Vector Machine 1995'
IBM Deep Blue 1997'
Variational inference 2000'
Image Net 2009'
AlexNet convolution networks 2012'
Applications of Deep Neural Networks
Alpha Go
2012 - now

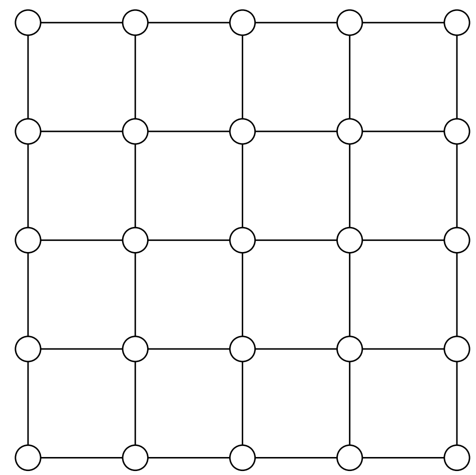

Statistical Physics
(machine learning related)
Machine Learning
(neural network related)
19th century
Boltzmann, Clausius, Maxwell,
Statistical Mechanics
Building of statistical methods
Least squares, Bayes rule
19th century
1895' Curie-Weiss mean-field
1920' Ising model
1935' Bethe Approximation
1953' Metropolise, MCMC
1957' Jaynes, Maximum Entropy principle
1975' Edward and Anderson spin-glass model
1978' Sherrington-Kirkpatrick spin-glass model
1979' Parisi' replica symmetry breaking
1980' Nishimori line
1982' Hopfield neural networks
1982' Simulated annealing
1985' Amit-Gutfreund-Sompolinsky
Phase diagram of the Hopfield NN
Gardner and Derrida, capacity of perceptron
1989' Krauth and Mezard
Capacity of binary perceptrons
1990 - 2009'
Applications of spin glass theory in optimizations, Mean-field approximations
2009 - now
message passings algorithms for inference and learning
Markov Chain 1913'
Shannon, Information Theory 1948'
Turin's learning machine 1950'
Perceptron 1957'
Minsky and Papert,
Limitations of perceptron 1969'
Necognitron,
early convolution neural networks 1980'
Hopfield neural networks 1982'
Backpropagation 1986'
Reinforcement learning 1989'
Support Vector Machine 1995'
IBM Deep Blue 1997'
Variational inference 2000'
Image Net 2009'
AlexNet convolution networks 2012'
Applications of Deep Neural Networks
Alpha Go
2012 - now

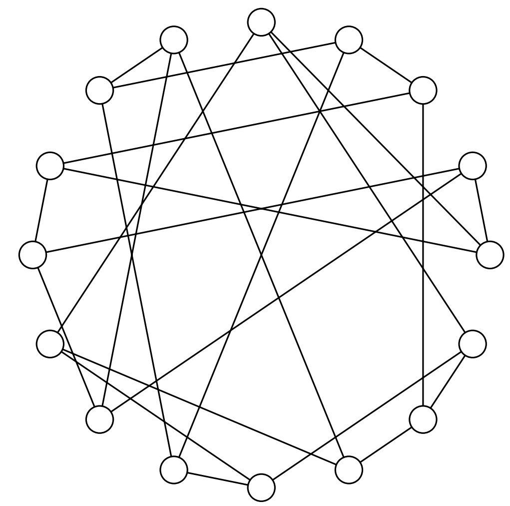

Statistical Physics
(machine learning related)
Machine Learning
(neural network related)
19th century
Boltzmann, Clausius, Maxwell,
Statistical Mechanics
Building of statistical methods
Least squares, Bayes rule
19th century
1895'
Curie-Weiss mean-field
1920'
Ising model
1935'
Bethe Approximation
1953'
Metropolise, MCMC
1957'
Jaynes, Maximum Entropy principle
1975'
Edward and Anderson spin-glass model
1978'
Sherrington-Kirkpatrick spin-glass model
1979'
Parisi' replica symmetry breaking
1980'
Nishimori line
1982'
Hopfield neural networks
1982'
Simulated annealing
1985'
Amit-Gutfreund-Sompolinsky
Phase diagram of the Hopfield NN
Gardner and Derrida, capacity of perceptron
1989'
Krauth and Mezard
Capacity of binary perceptrons
1990 - 2009'
Applications of spin glass theory in
optimizations, Mean-field approximations
2009 - now
message passings algorithms for
inference and learning
Markov Chain
1913'
Shannon, Information Theory
1948'
Turin's learning machine
1950'
Perceptron
1957'
Minsky and Papert,
Limitations of perceptron
1969'
Necognitron,
early convolution neural networks
1980'
Hopfield neural networks
1982'
Backpropagation
1986'
Reinforcement learning
1989'
Support Vector Machine
1995'
IBM Deep Blue
1997'
Variational inference
2000'
Image Net
2009'
AlexNet convolution networks
2012'
Applications of Deep Neural Networks
Alpha Go
2012 - now

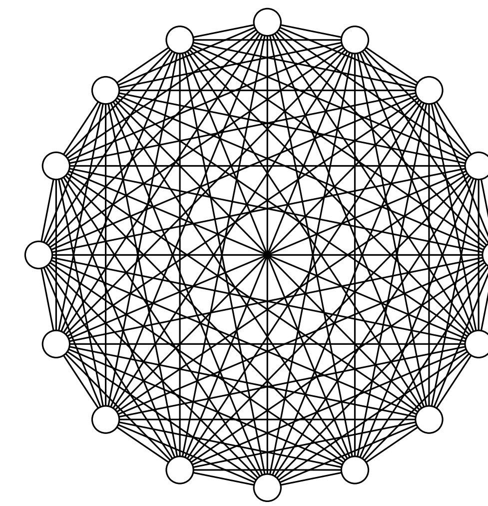
例子3：统计物理与量子计算

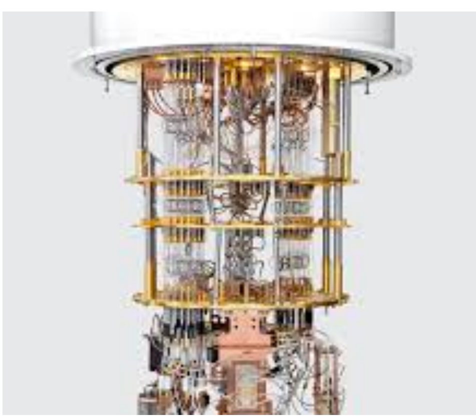


统计物理模型

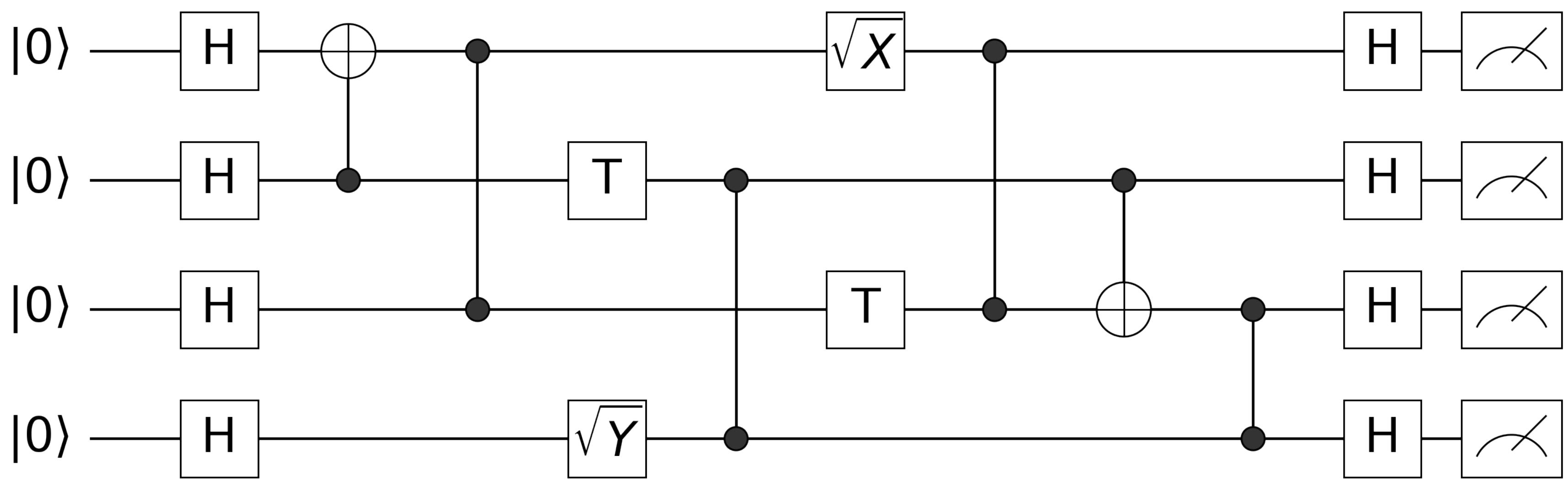






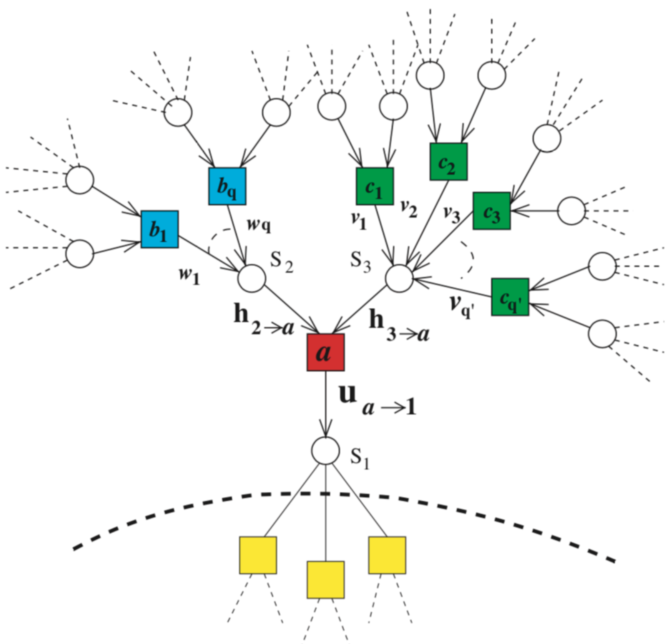


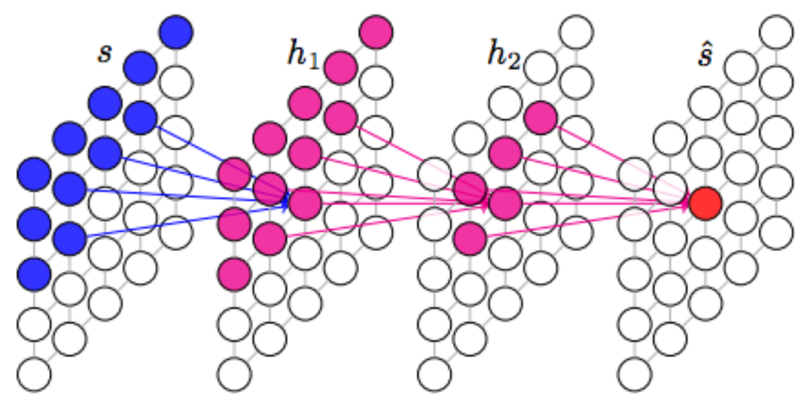
量子线路模拟

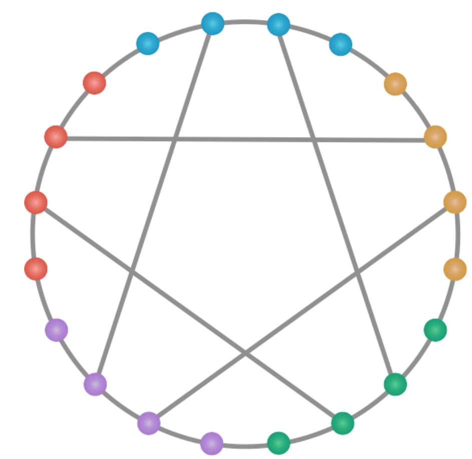


复数温度统计物理模型配分函数 = 量子线路单振幅测量

My research journey on Statistical Mechanics:
From Mean-Field to Neural Networks Then to Tensor Networks







Mean-field

Variational Mean-field
Belief propagation
Replica symmetry breaking
Survey Propagation
Expectation Propagation

Neural Networks

Variational Autoregerssive Networks
Feedback-set VAN

Tensor Networks

CATN
Tropical Tensor Networks

统计力学 Statistical Mechanics

$$\mathbf{S} = \{+1, -1\}^n$$

$$\uparrow\ \uparrow\ \uparrow\ \downarrow\ \uparrow\ \downarrow\ \downarrow\ \downarrow\ \downarrow\ \downarrow\ \uparrow\ \uparrow$$

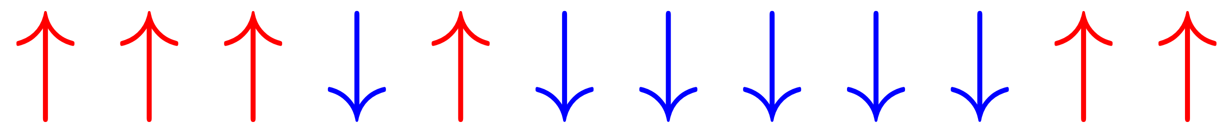
$$P(\mathbf{S}) = \frac{1}{Z} \mathrm{e}^{-\beta E(\mathbf{S})}$$

$$Z = \sum_{\mathbf{S}} \mathrm{e}^{-\beta E(\mathbf{S})}$$

- 自由能
- 计算统计量
- 无偏采样

统计力学 Statistical Mechanics

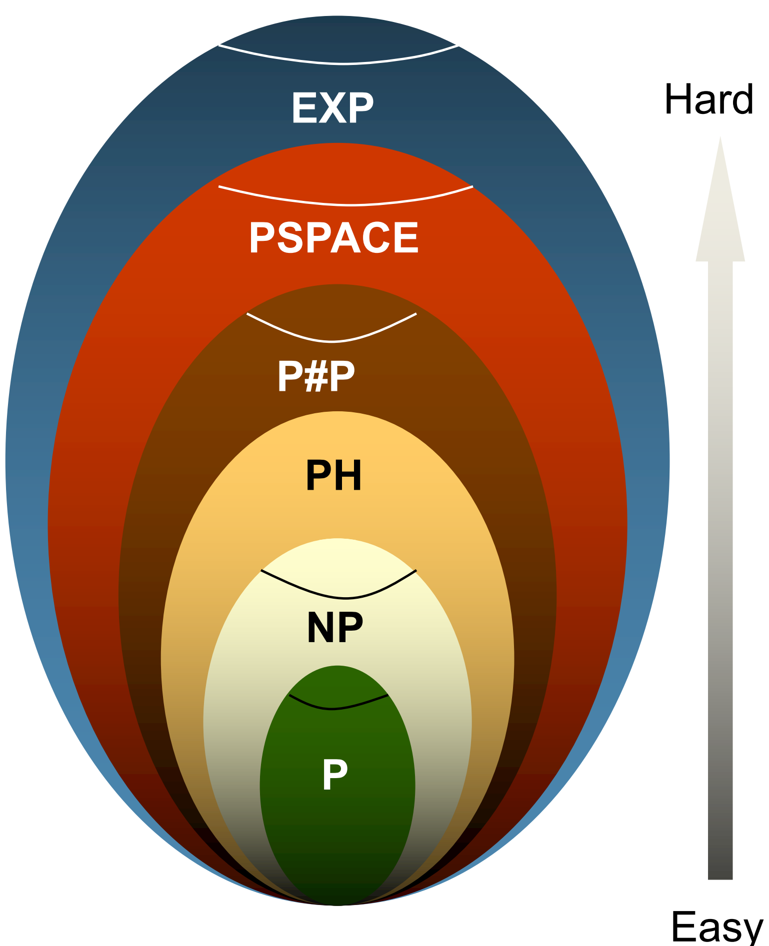
$$\mathbf{S} = \{+1, -1\}^n$$



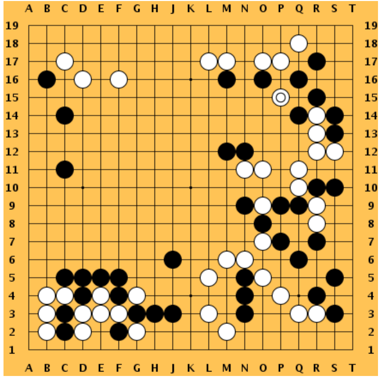
$$P(\mathbf{S}) = \frac{1}{Z} \mathrm{e}^{-\beta E(\mathbf{S})}$$

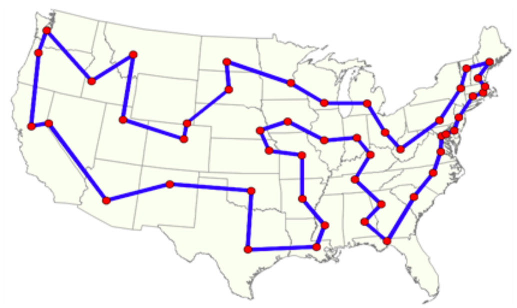
$$Z = \sum_{\mathbf{S}} \mathrm{e}^{-\beta E(\mathbf{S})}$$

- 自由能
- 计算统计量
- 无偏采样

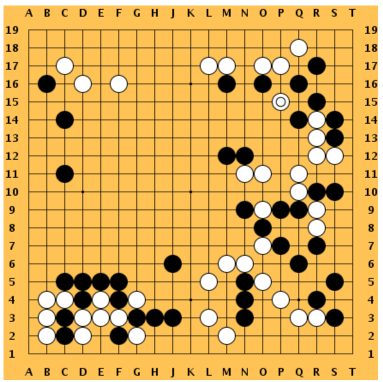


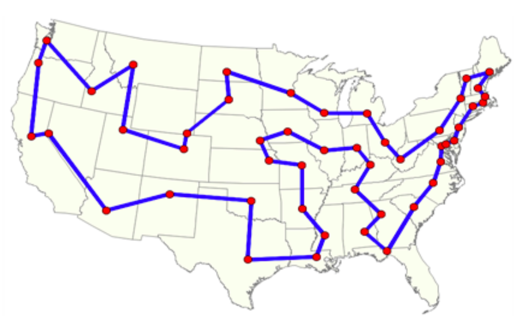
计算的难度

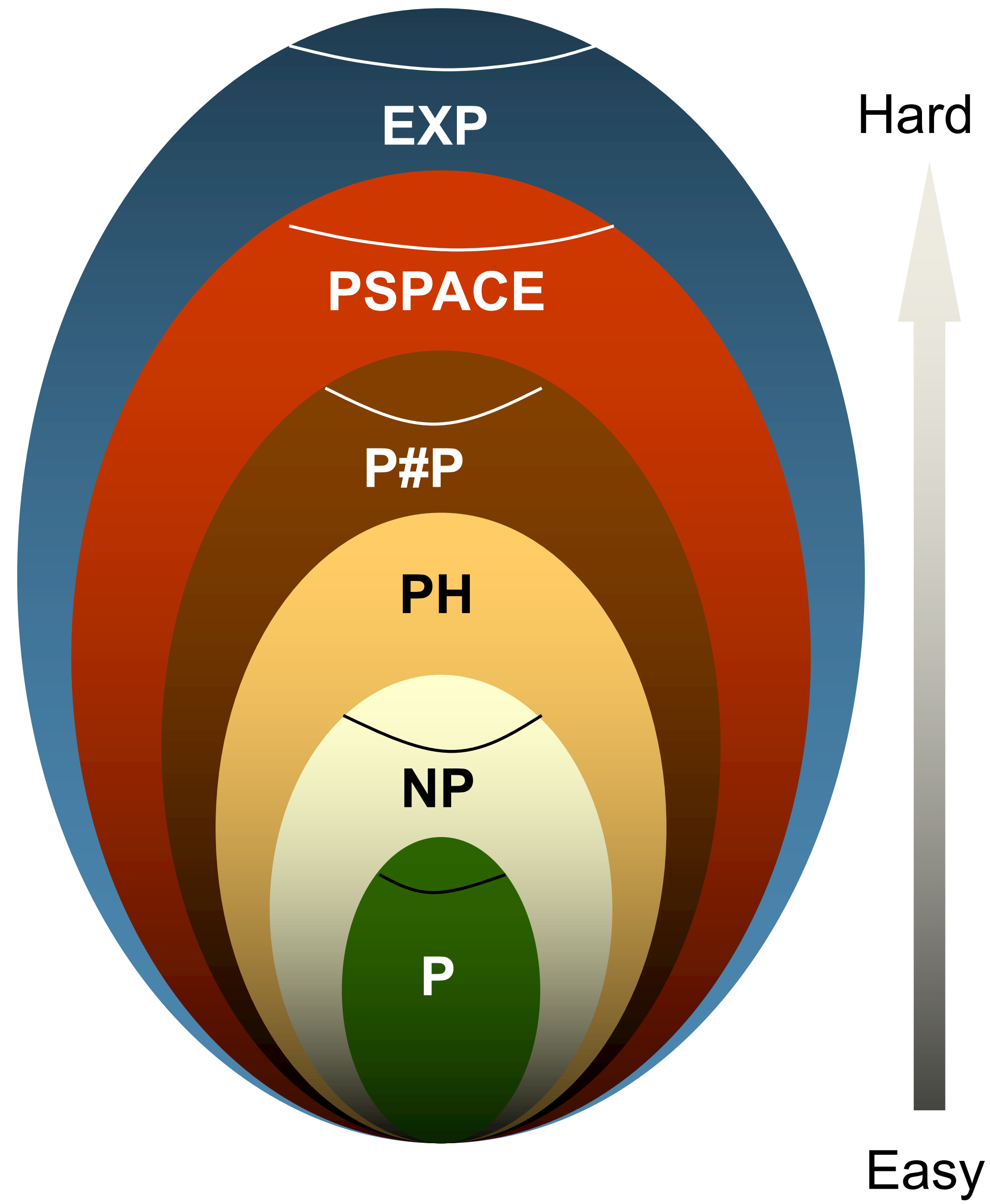




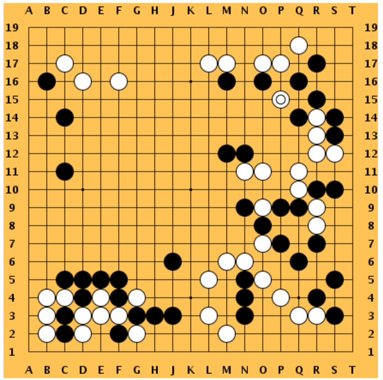
计算的难度

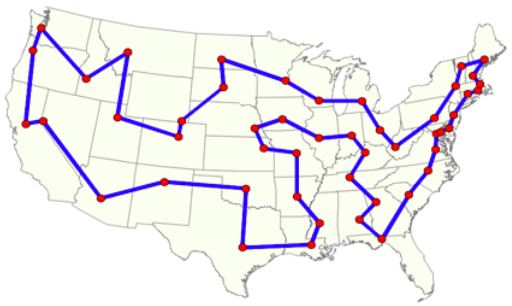




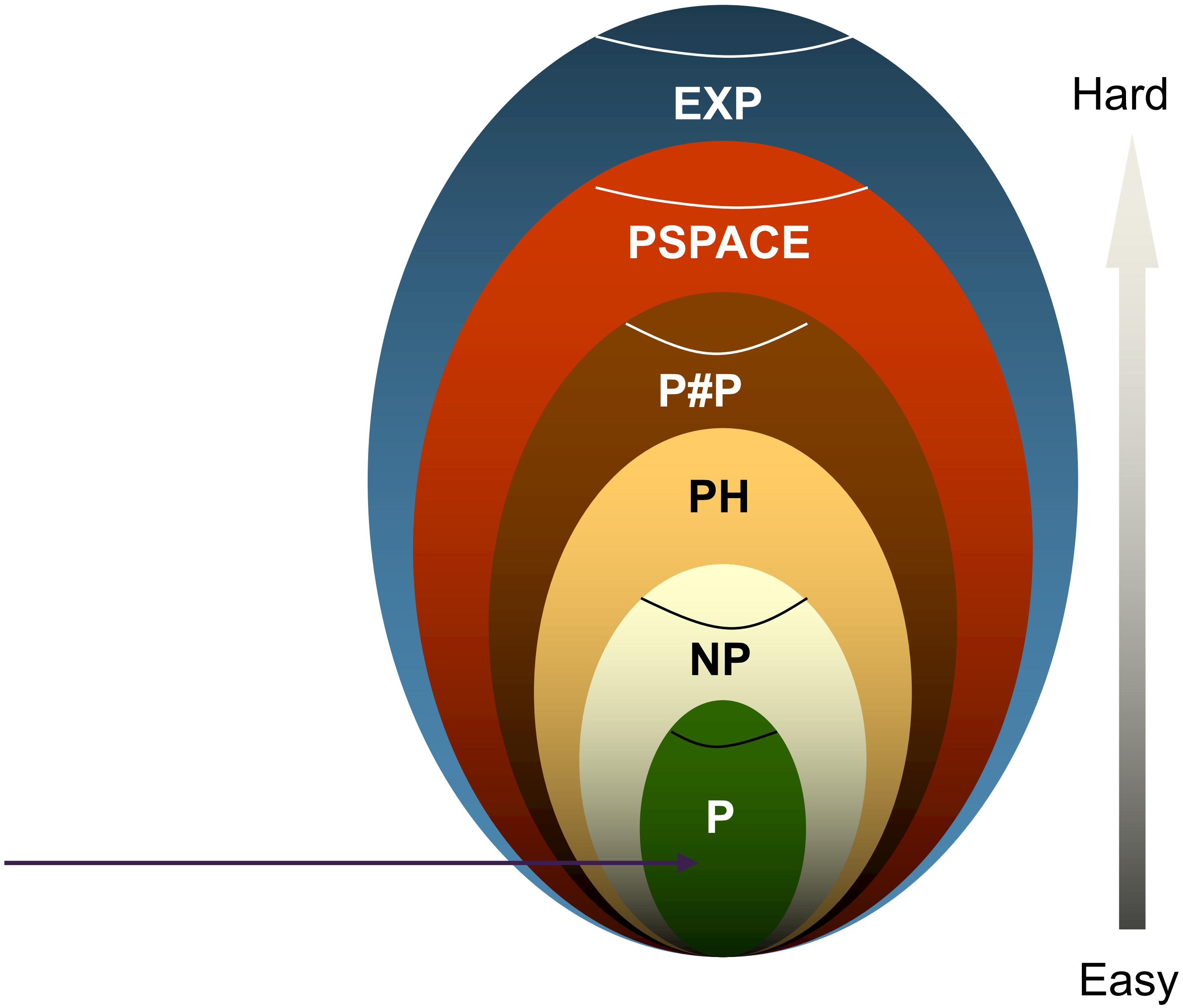


计算的难度

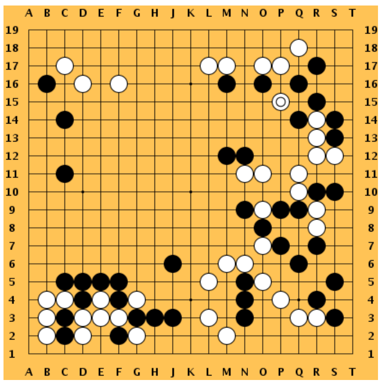


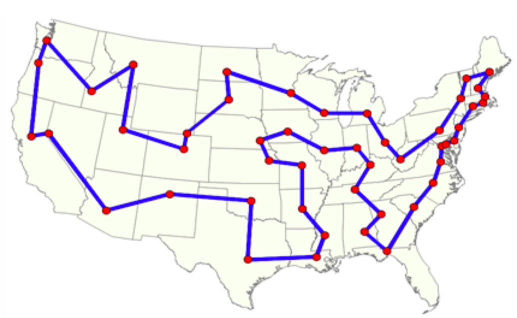


排序



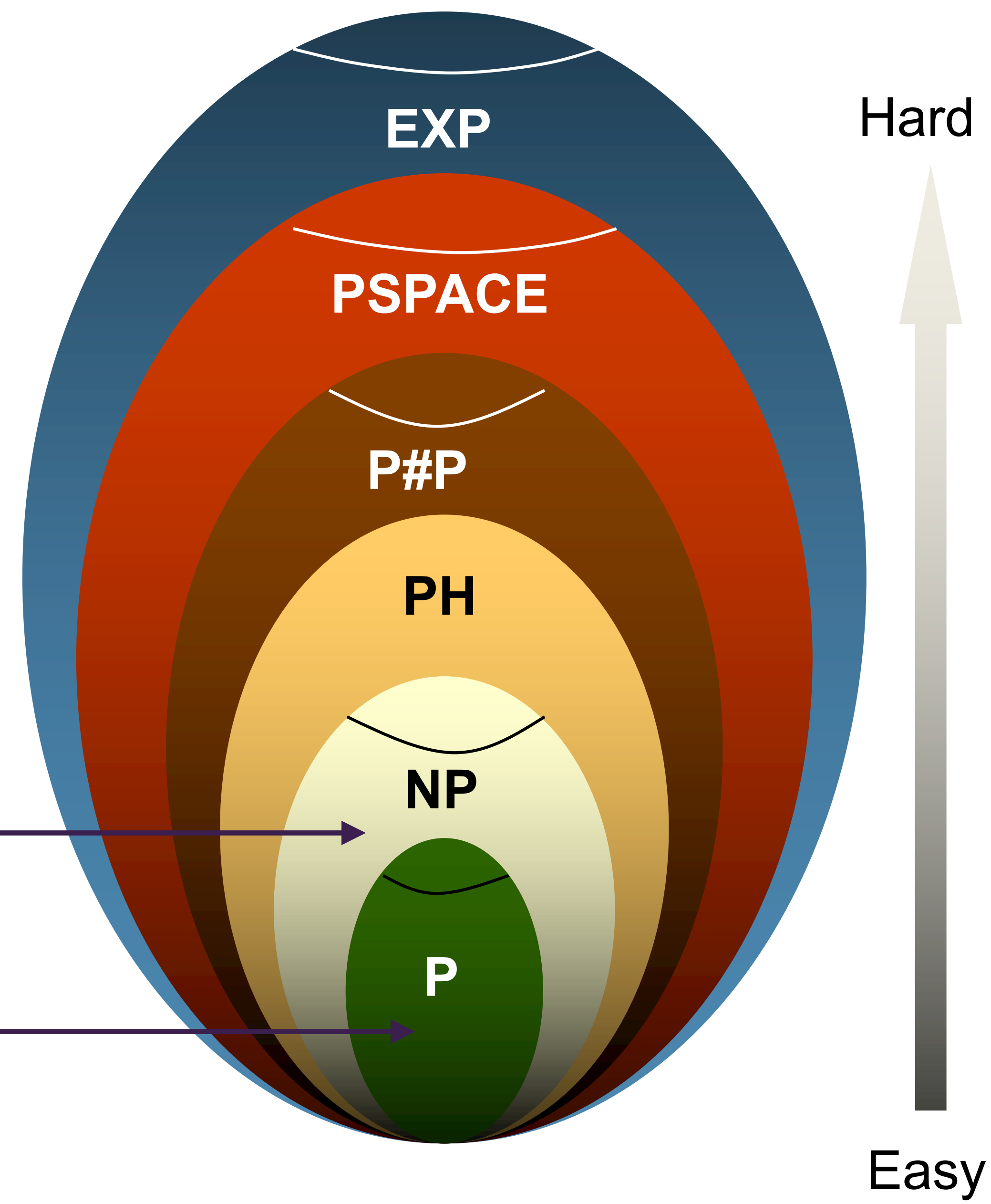
计算的难度



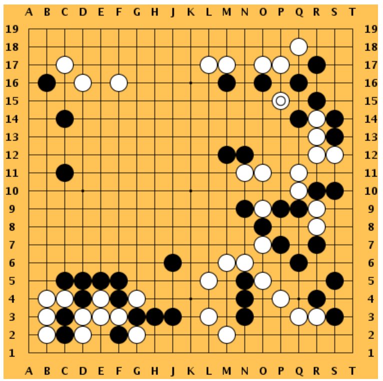


旅行商问题

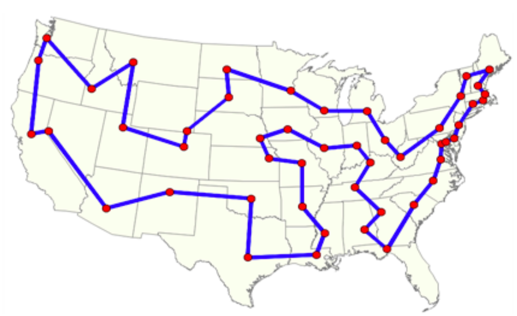
排序



计算的难度

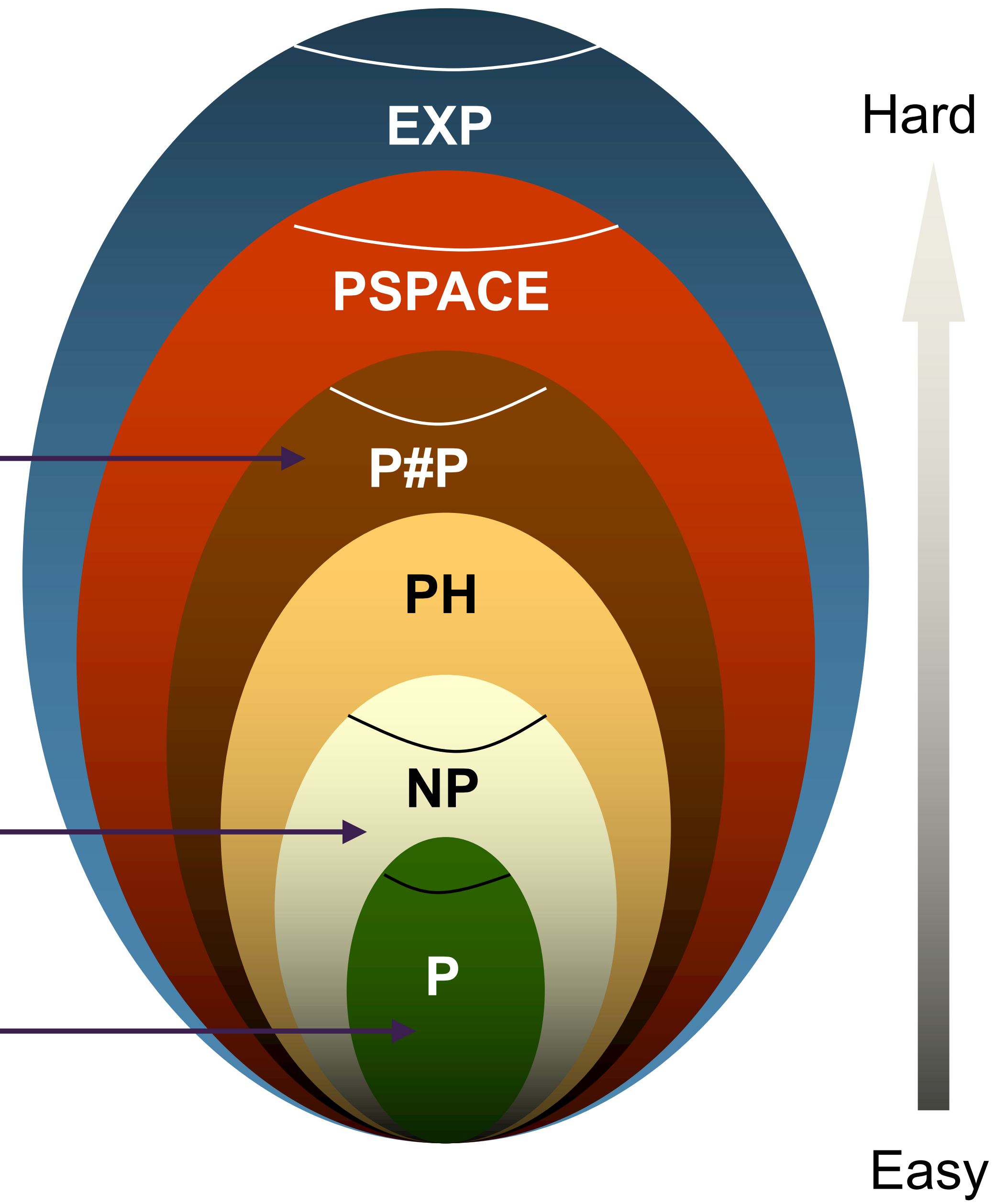


计算联合概率

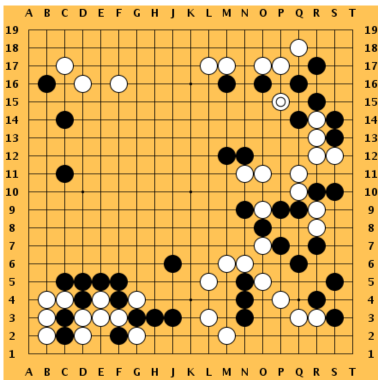


旅行商问题

排序

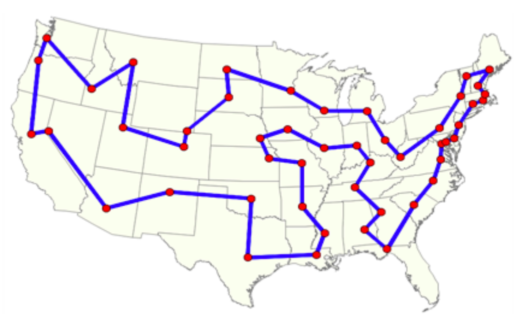


计算的难度



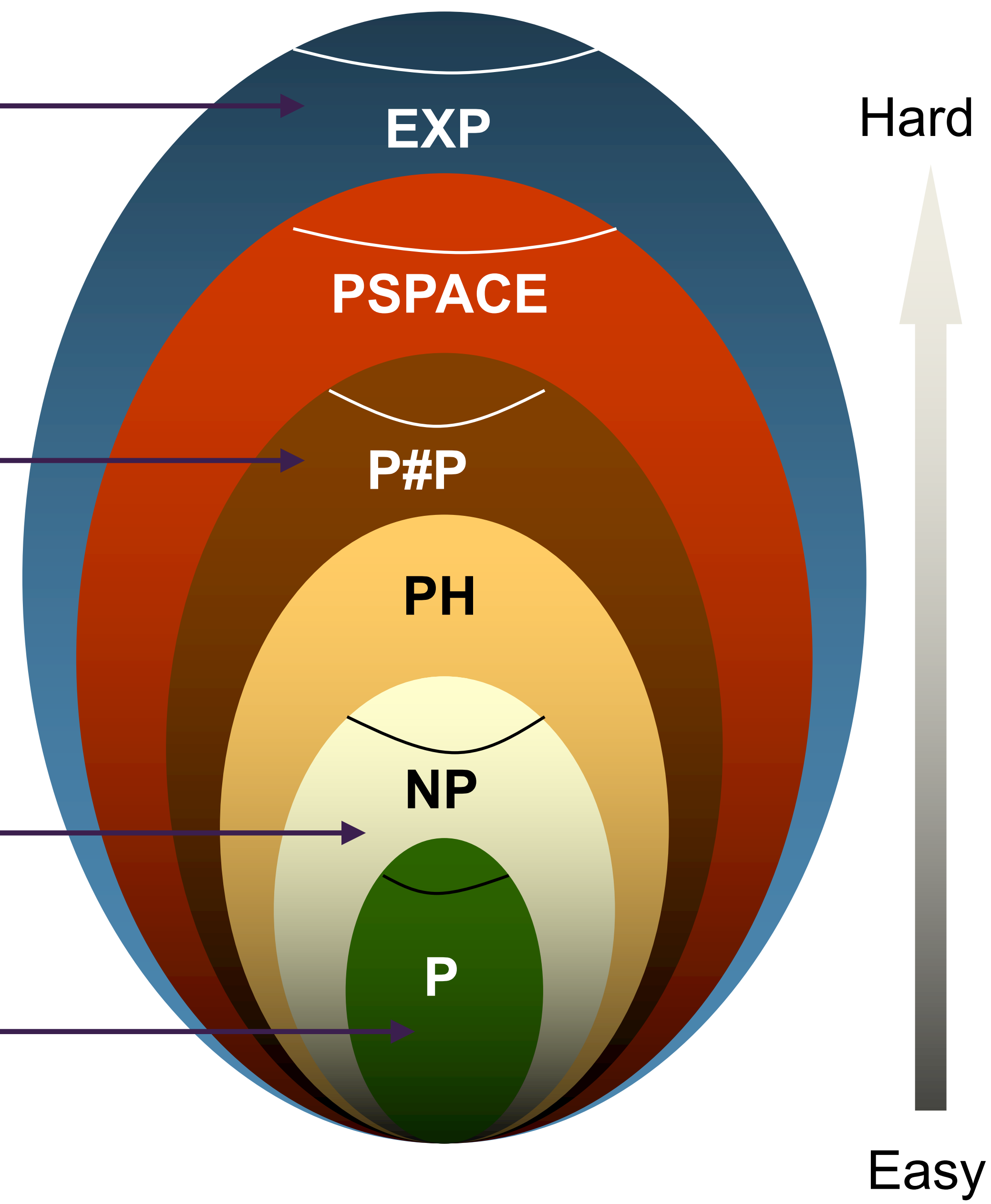
围棋

计算联合概率



旅行商问题

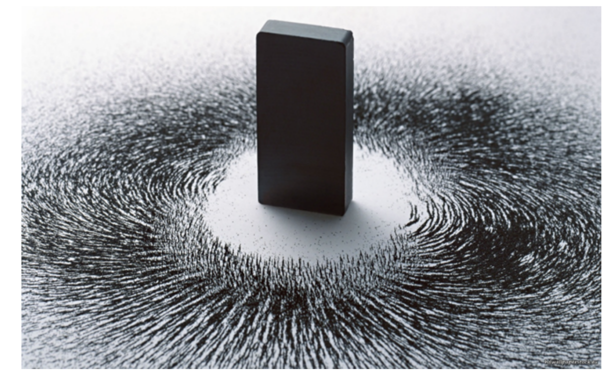
排序



核心方法和困难

统计力学: 自由能，观测量，无偏采样

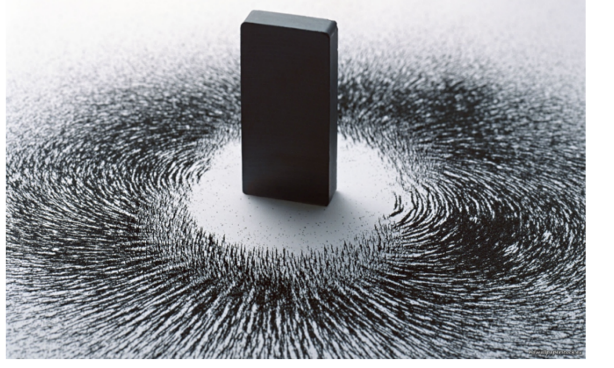
- #P-Hard 问题
- 能量曲面崎岖，优化、采样困难



$$P(\mathbf{S}) = \frac{1}{Z}\mathrm{e}^{-\beta E(\mathbf{S})}$$

核心方法和困难

统计力学: 自由能，观测量，无偏采样



$$P(\mathbf{S}) = \frac{1}{Z} \mathrm{e}^{-\beta E(\mathbf{S})}$$

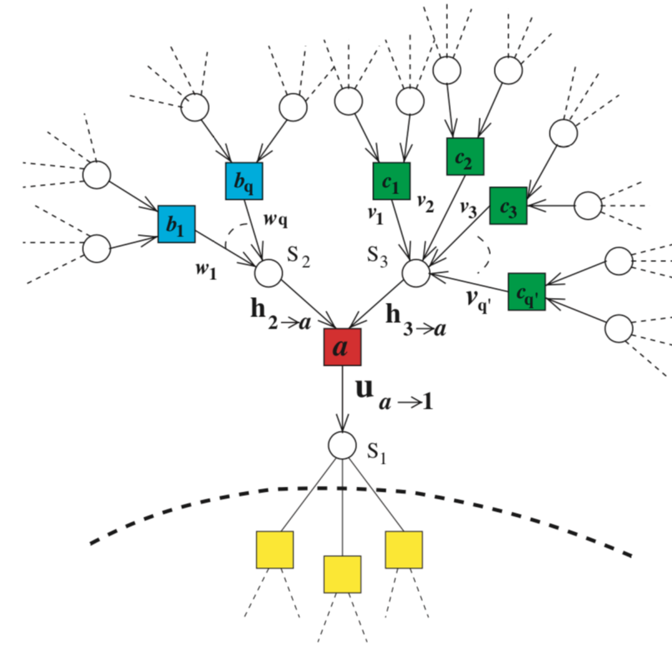
- #P-Hard 问题
- 能量曲面崎岖，优化、采样困难

统计力学经典方法	局限性
重整化群	依赖拓扑结构
蒙特卡洛	自相关问题 难以计算自由能和熵
平均场方法	对假设敏感 变分分布表达能力弱

平均场方法

应用于

- 无序系统
- 凝聚态物理
- 机器学习
- 统计推断 …



困难

- 自旋玻璃
- 组合优化
- 分析神经网络

$$q(\mathbf{s}) = \prod_i q_i(s_i)$$

[Curie, Weiss, 1907']

局限性

- 变分分布表述能力弱

$$q(\mathbf{s}) = \frac{\prod_{(ij)} q_{ij}(s_i, s_j)}{\prod_i q_i(s_i)^{d_i-1}}$$

[Bethe, 1935']

Variational Methods: working with an upper bound

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

Boltzmann distribution

$$-\beta F = \ln Z = \ln \sum_{\mathbf{s}} e^{-\beta E(\mathbf{s})}$$

$$= \ln \sum_{\mathbf{s}} q(\mathbf{s}) \frac{e^{-\beta E(\mathbf{s})}}{q(\mathbf{s})}$$

Introduce a trackable distribution q(s)

$$\geq \sum_{\mathbf{s}} q(\mathbf{s}) \ln \frac{e^{-\beta E(\mathbf{s})}}{q(\mathbf{s})}$$

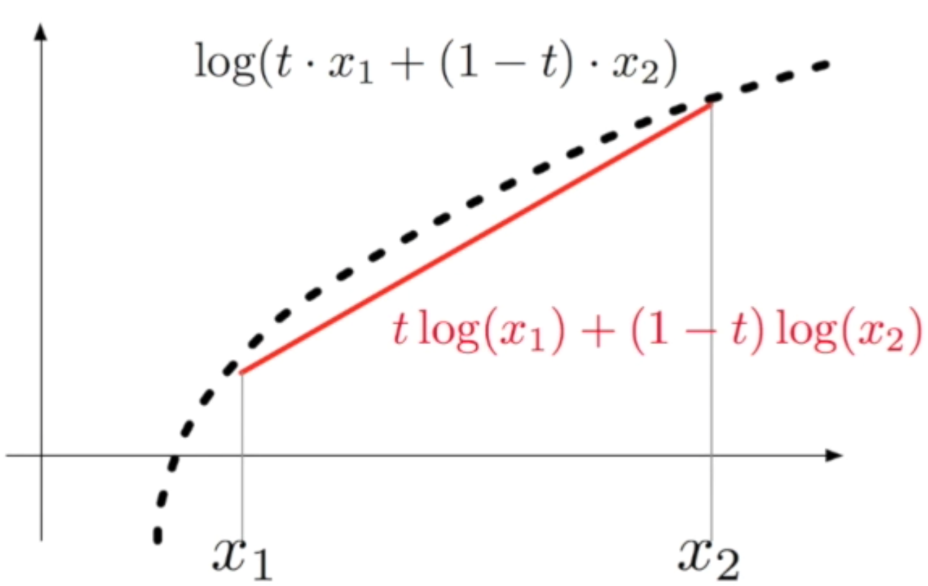
Jenson's inequality

$$= -\beta \sum_{\mathbf{s}} q(\mathbf{s}) E(\mathbf{s}) - \sum_{\mathbf{s}} q(\mathbf{s}) \ln q(\mathbf{s})$$

$$= -\beta F_q$$

$$F_q = \langle E \rangle_q - \frac{1}{\beta} S_q$$

$$= F - \frac{1}{\beta} \mathrm{D_{KL}}(q \| p)$$



Many faces of the variational Free Energy

$$P(\mathbf{s}|\mathbf{x}) = \frac{e^{\ln P(\mathbf{x}|\mathbf{s})P_0(\mathbf{s})}}{e^{\ln P(\mathbf{x})}}$$

$$\ln Z = \ln P(\mathbf{x}) = \ln \sum_{\mathbf{s}} e^{\ln P(\mathbf{x}|\mathbf{s})P_0(\mathbf{s})}$$

$$\ln P(\mathbf{x}) \geq \sum_{\mathbf{s}} Q(\mathbf{s}|\mathbf{x}) \ln[P(x|s)P_0(s)] - \sum_{\mathbf{s}} Q(\mathbf{s}|\mathbf{x}) \ln Q(\mathbf{s}|\mathbf{x})]$$

Variational Free Energy: Energy - Entropy

$$\ln P(\mathbf{x}) \geq \sum_{\mathbf{s}} Q(\mathbf{s}|\mathbf{x}) \ln[P(x|s)] - \mathrm{KL}\left[\ln Q(\mathbf{s}|\mathbf{x}) || P_0(\mathbf{s})\right]$$

Variational autoencoder: reconstruction error - KL regularization

Mean-field methods: Variational Mean-field

$$q(\mathbf{s}) = \prod_i q_i(s_i)$$

n parameters !

$$F_q = \sum_{\mathbf{s}} \left[q(\mathbf{s})E(\mathbf{s}) + \frac{1}{\beta}\sum_i \ln q_i(s_i) \right]$$

$$\nabla_{\{q_i\}} F_q = 0 \Rightarrow \text{Naïve Mean-Field equations}$$

In case of the Ising model with $E(\mathbf{s}) = -\sum_{(ij)} J_{ij} s_i s_j$

$$m_i = 2q_i(s_i = 1) - 1$$

$$m_i = \tanh(\beta \sum_{j \neq i} J_{ij} m_j)$$

$$F_q = \sum_{(ij)} J_{ij} m_i m_j + \frac{1}{\beta} \sum_i \left(\log \frac{1+m_i}{2} + \log \frac{1-m_i}{2} \right)$$

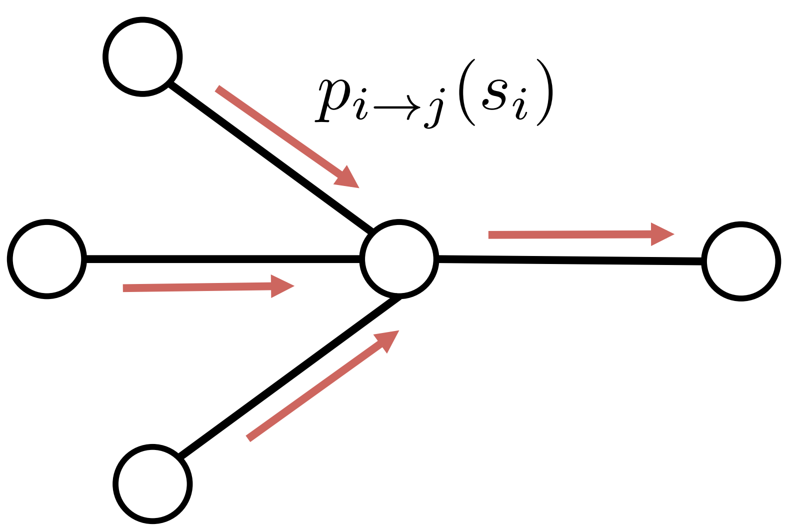
Mean-field methods: Bethe approximation

$$q(\mathbf{s}) = \frac{\prod_{(ij)} q_{ij}(s_i, s_j)}{\prod_i q_i(s_i)^{d_i - 1}}$$

- Exact on a tree
- A good approximation on sparse graphs or dense + weak systems
- However in general q(s) is not normalized on loopy graphs

$$F_q = \sum_{\mathbf{s}} q(\mathbf{s}) E(\mathbf{s}) + \frac{1}{\beta} \sum_{(ij)} \sum_{s_i, s_j} \ln q_{ij}(s_i, s_j) - \frac{1}{\beta} \sum_i (d_i - 1) \sum_{s_i} \ln q_i(s_i)$$

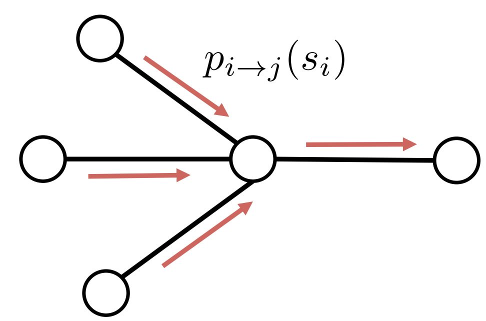
$$\nabla_{\{q_i, q_{ij}\}} F_q = 0 \Rightarrow \text{belief propagation}$$



Pros and Cons of Mean-field

Pros:

- Analytical computation of Free Energy
- Fast Message Passing
- Analysable



Cons:

- Requires certain conditions to hold
- Low expressive power

$$q(\mathbf{s}) = \prod_i q_i(s_i)$$

$$q(\mathbf{s}) = \frac{\prod_{(ij)} q_{ij}(s_i, s_j)}{\prod_i q_i(s_i)^{d_i - 1}}$$

Using a more expressive variational distribution

Variational distributions in Mean-field methods are too weak.

$$q(\mathbf{s}) = \prod_i q_i(s_i)$$

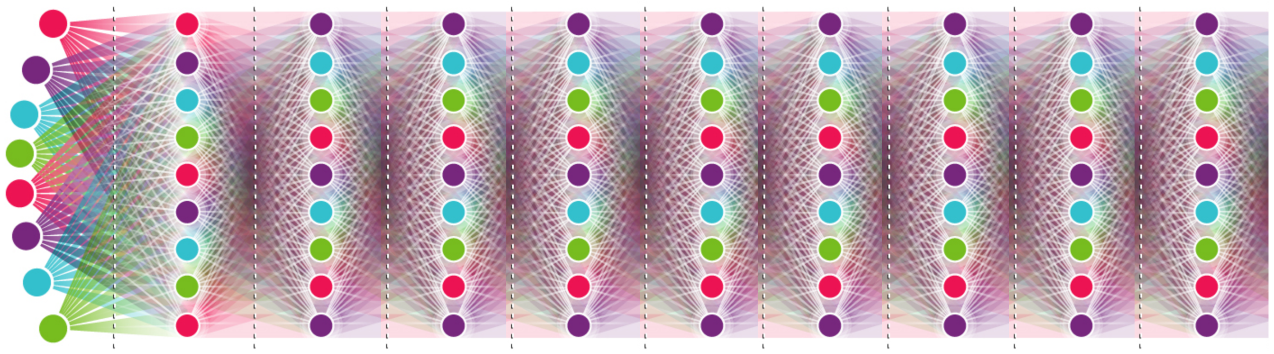
n parameters

$$q(\mathbf{s}) = \frac{\prod_{(ij)} q_{ij}(s_i, s_j)}{\prod_i q_i(s_i)^{d_i - 1}}$$

2m parameters

It is difficult to increase the number of parameters

Machine learning can help: neural network models contain lots of trainable parameters,
giving good representation power in theory.



Using a more expressive variational distribution

Variational distributions in Mean-field methods are too weak.

$$q(\mathbf{s}) = \prod_i q_i(s_i)$$

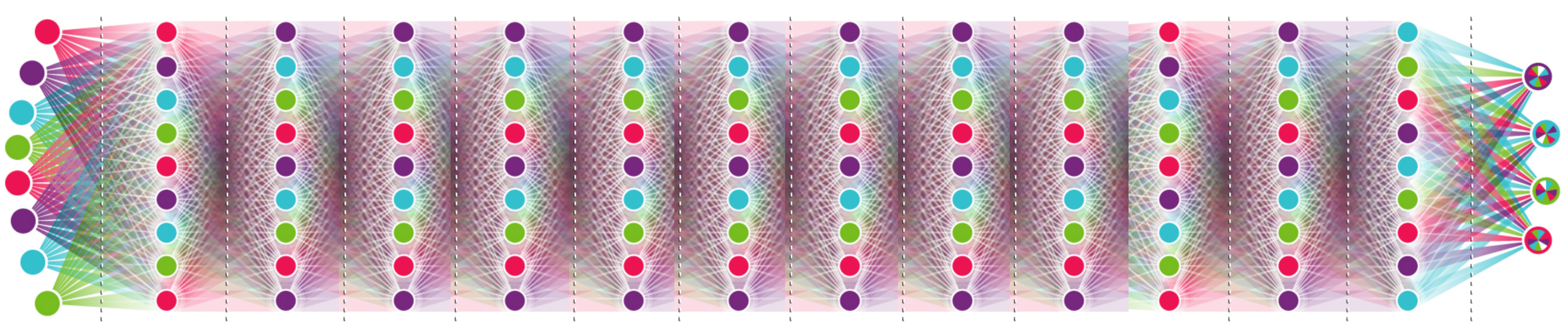
n parameters

$$q(\mathbf{s}) = \frac{\prod_{(ij)} q_{ij}(s_i, s_j)}{\prod_i q_i(s_i)^{d_i - 1}}$$

2m parameters

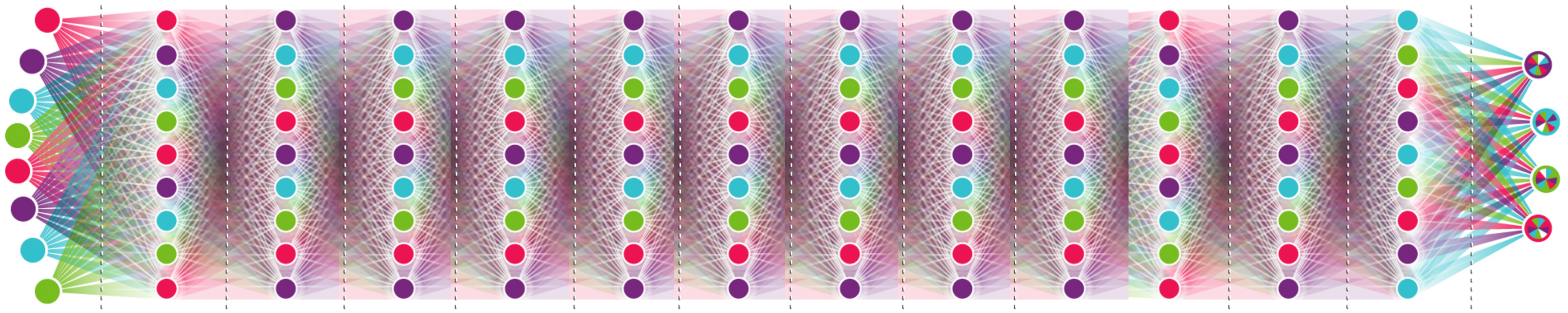
It is difficult to increase the number of parameters

Machine learning can help: neural network models contain lots of trainable parameters,
giving good representation power in theory.



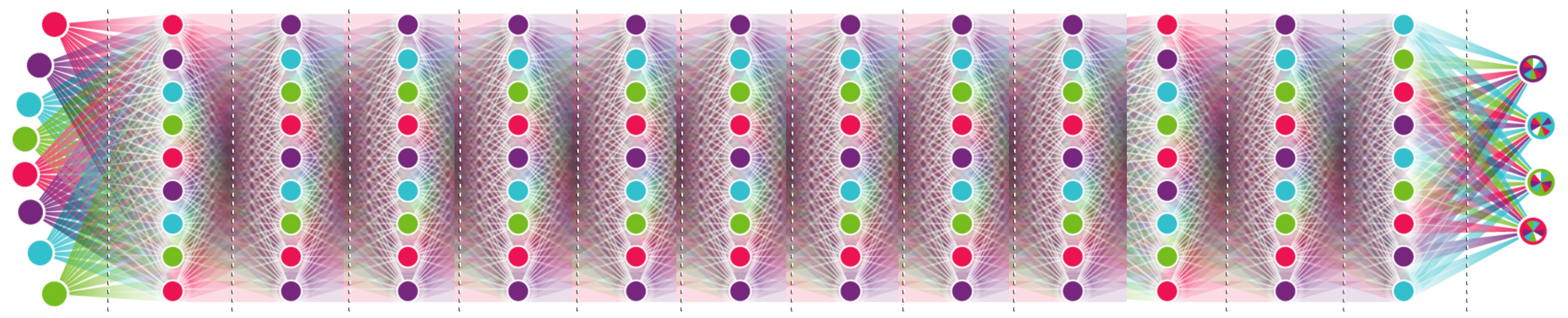
Using a more expressive variational distribution

Neural network models contain lots of trainable parameters, giving good representation power in theory.

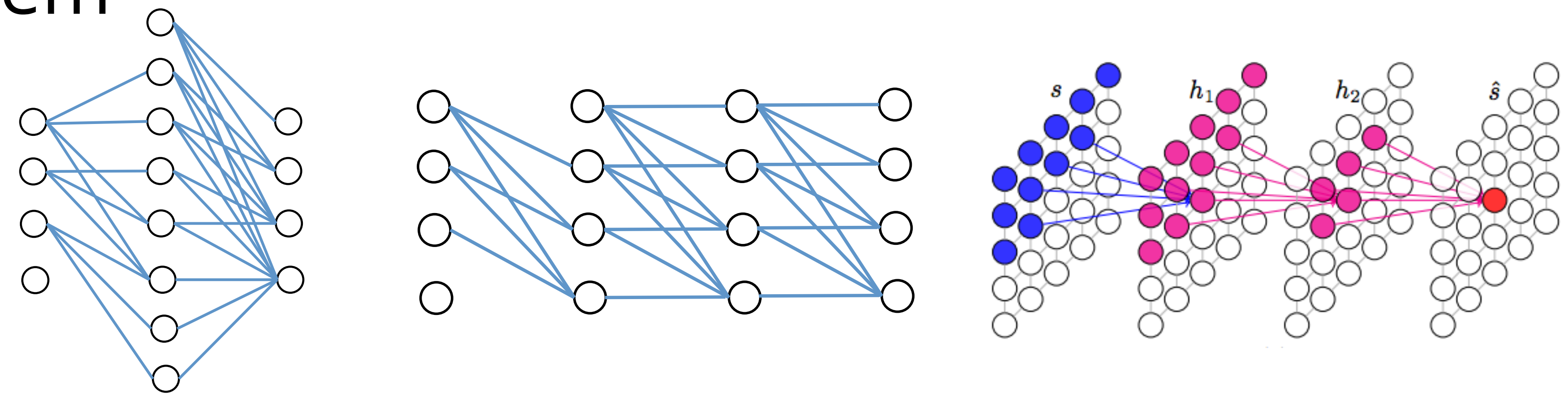


Using a more expressive variational distribution

Neural network models contain lots of trainable parameters, giving good representation power in theory.

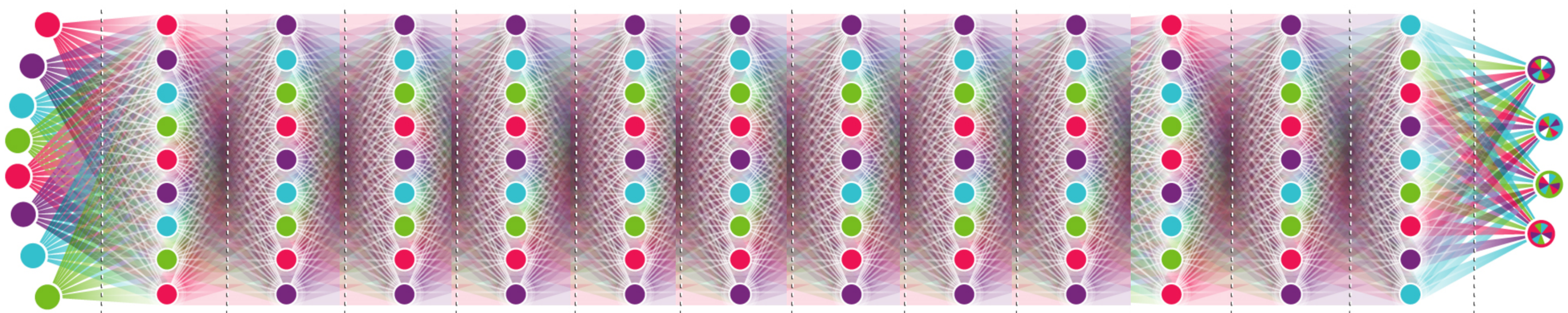


Design Neural network models to use structures of the problem

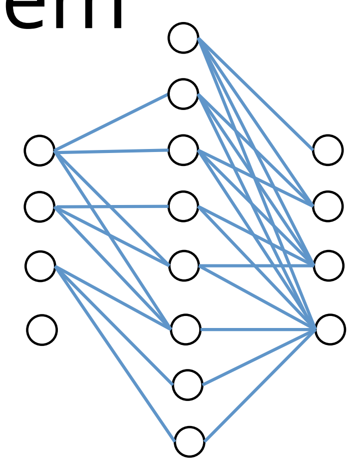


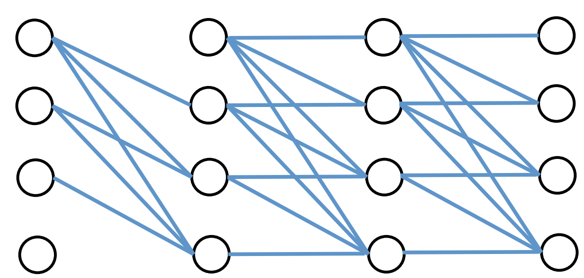
Using a more expressive variational distribution

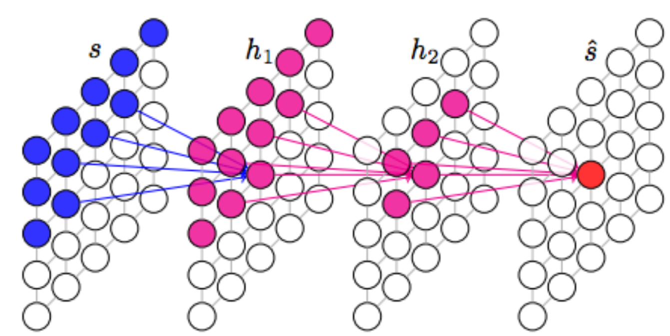
Neural network models contain lots of trainable parameters, giving good representation power in theory.



Design Neural network models to use structures of the problem







VAN: variational methods to Stat. Mech. using autoregressive neural networks.

D Wu, L Wang, PZ, *Phys. Rev. Lett.* 122, 080602 (2019)
O Sharir, Y Levine, N Wies, G Carleo, A Shashua, *Phys. Rev. Lett.* 124, 020503 (2020)

- Expressive power:
  - Using deep neural networks
    - universal approximator
    - convenient programming platforms
    - easy GPU support
- Tractability of variational free energy:
  - Do it by sampling !
  - Constraints on neural networks:
    - represent joint distribution
    - support Direct Sampling
    - explicit probability output

CPU

GPU

→ Autoregressive Networks

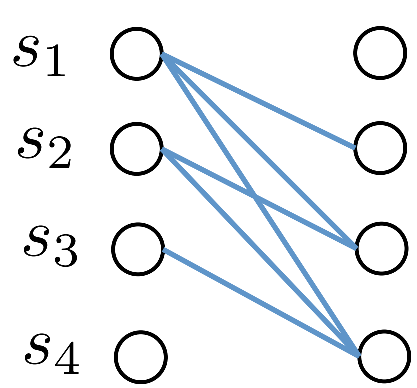
Auto-regressive distribution

- Representing joint distribution using chain rule of conditional probabilities.

$$q(\mathbf{s}) = \prod_i q(s_i|\mathbf{s}_{j<i})$$

$$\begin{aligned} q(s_1, s_2, s_3, s_4) &= q(s_4|s_3, s_2, s_1)q(s_3, s_2, s_1) \\ &= q(s_4|s_3, s_2, s_1)q(s_3|s_2, s_1)q(s_2, s_1) \\ &= q(s_4|s_3, s_2, s_1)q(s_3|s_2, s_1)q(s_2|s_1)q(s_1) \end{aligned}$$

Fully Visible Belief Network
[Frey 1998]



$$\hat{s}_1 = \text{sigmoid}(0) = q(s_1 = 1)$$

$$\hat{s}_2 = \text{sigmoid}(w_{12}s_1) = q(s_2 = 1|s_1)$$

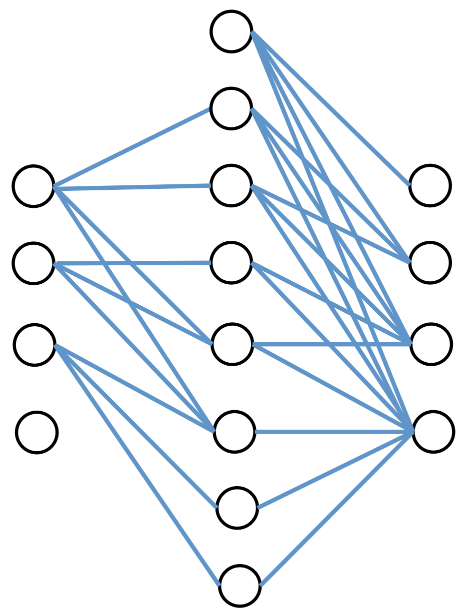
$$\hat{s}_3 = \text{sigmoid}(w_{13}s_1 + w_{23}s_2) = q(s_3 = 1|s_2, s_1)$$

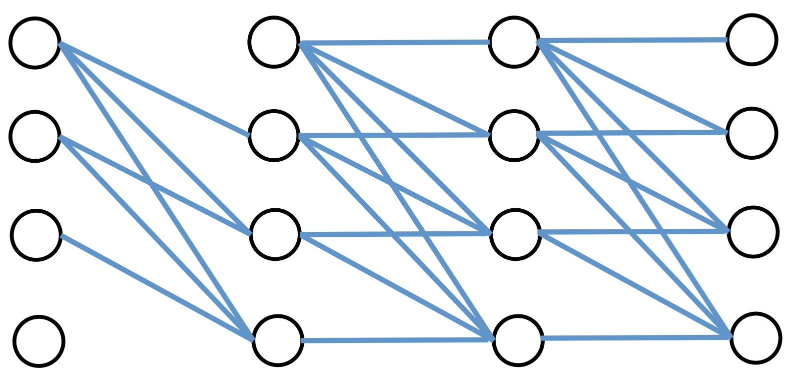
$$\hat{s}_4 = \text{sigmoid}(w_{14}s_1 + w_{24}s_2 + w_{34}s_4) = q(s_4 = 1|s_3, s_2, s_1)$$

$$q(s1, s2, s3, s4) = \hat{s}_1\hat{s}_2\hat{s}_3\hat{s}_4$$

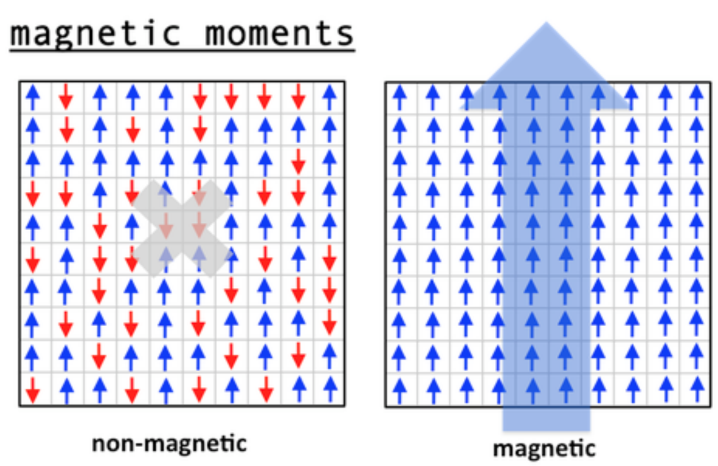
- Directed Sampling is easy because *we have all the conditional probabilities*! This is known as *ancestral sampling* [*Bishop 2006*]

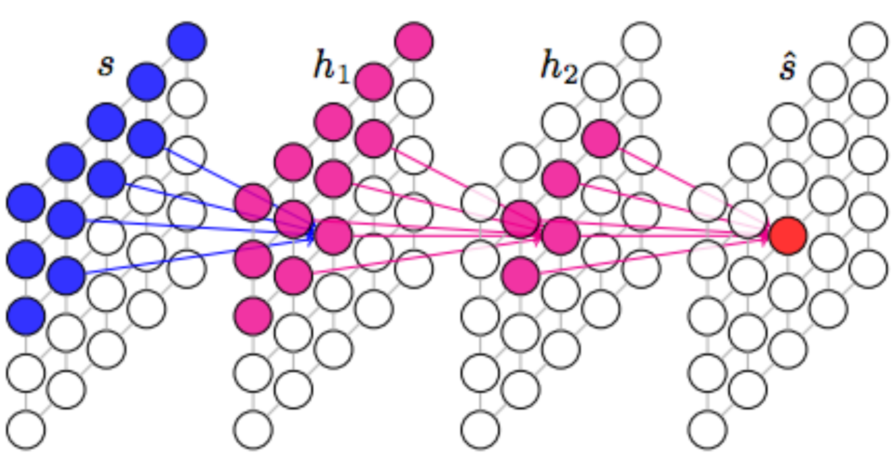
- Extending the expressibility and generalization power by making it deeper and using weights sharing.
[*Bengio/Bengio 2000, Uria/Cote/Gregor/Murray/Larochelle 2016, Germain/Gregor/Murray/Larochelle 2015, Larochelle/Murray 2011, Gregor/Danihelka/Mnih/Blundell/Wierstra 2014*]





- When the system has topology structure, one should respect it !
[*PixelCNN, Van den Oord et. al., 2014*]





Minimizing the variational free energy

- Minimizing the variational free energy is equivalent to minimizing KL divergence

$$\hat{\theta} = \arg\min_{\theta} \mathrm{D}_{\mathrm{KL}}(q_\theta | p) = \arg\min F_q$$

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

- The parameters are continuous, so we can apply the (stochastic) gradient descent

$$\begin{aligned}
\beta \nabla_\theta F_q &= \nabla_\theta \sum_{\mathbf{s}} \left[q_\theta(\mathbf{s}) \cdot (\beta E(\mathbf{s}) + \ln q_\theta(\mathbf{s}))\right] \\
&= \sum_{\mathbf{s}} \left[\nabla_\theta q_\theta(\mathbf{s}) \cdot (\beta E(\mathbf{s}) + \ln q_\theta(\mathbf{s})) + q_\theta(\mathbf{s}) \nabla_\theta \ln q_\theta(\mathbf{s})\right] \\
&= \sum_{\mathbf{s}} \left[q_\theta(\mathbf{s}) \nabla_\theta \log q_\theta(\mathbf{s}) \cdot (\beta E(\mathbf{s}) + \ln q_\theta(\mathbf{s})) + \nabla_\theta q_\theta(\mathbf{s})\right] \\
&= \mathbb{E}_{\mathbf{s} \sim q_\theta(\mathbf{s})} \left[\nabla_\theta \ln q_\theta(\mathbf{s}) \cdot \underbrace{(\beta E(\mathbf{s}) + \ln q_\theta(\mathbf{s}))}_{R(\mathbf{s})}\right]
\end{aligned}$$

Known as the ***REINFORCE*** algorithm **[Williams 1992]**

Variational Autoregressive Networks

Input: energy function and temperature

output: variational free energy, samples, and observables (e.g. correlations)

1. Initialize VAN randomly.

2. Draw many samples from VAN using ancestral sampling, and
   compute their log-probabilities

3. Compute variational free energy

   using samples and their log-probabilities

4. Do gradient-descent via back-propagation

5. Go to step 2, until loss converges, or reaches zero-variance

Stat. Mech. vs. Density Estimation

Stat. Mech. vs. Density Estimation

Statistical Mechanics	Density Estimation [e.g. ***PixelCNN, Van den Oord et. al., 2014***]

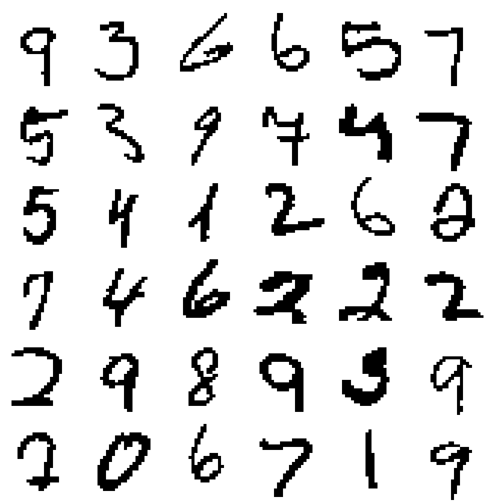
Stat. Mech. vs. Density Estimation

Statistical Mechanics

Given energy function $E(\mathbf{s})$

Density Estimation

[e.g. ***PixelCNN, Van den Oord et. al., 2014***]



Given data

Stat. Mech. vs. Density Estimation

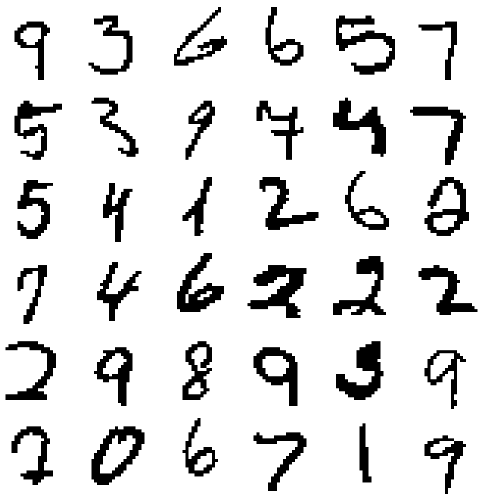
Statistical Mechanics

Given energy function $E(\mathbf{s})$

Boltzmann distribution $p_{\text{Boltzmann}}(\mathbf{s})$

Density Estimation

[e.g. ***PixelCNN, Van den Oord et. al., 2014***]



Given data

$$p_{\text{ data}}(\mathbf{s}) \propto \sum_{i \in \text{ data}} \delta(\mathbf{s} - \mathbf{s}_i)$$

Empirical data distribution

Stat. Mech. vs. Density Estimation

Statistical Mechanics

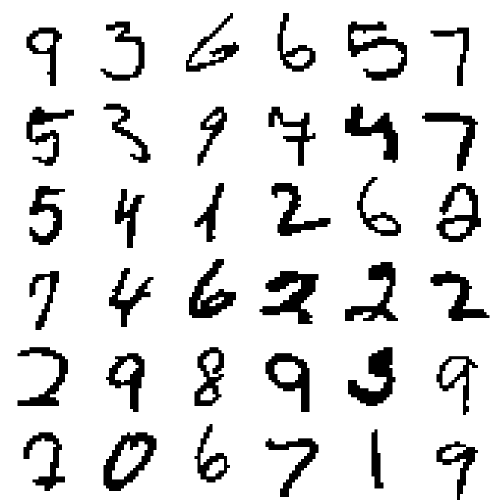
Given energy function $E(\mathbf{s})$

Boltzmann distribution $p_{\text{Boltzmann}}(\mathbf{s})$

Minimizing reverse KL $D_{\text{KL}}(q_\theta \| p_{\text{Boltzmann}})$

Density Estimation

[e.g. ***PixelCNN, Van den Oord et. al., 2014***]

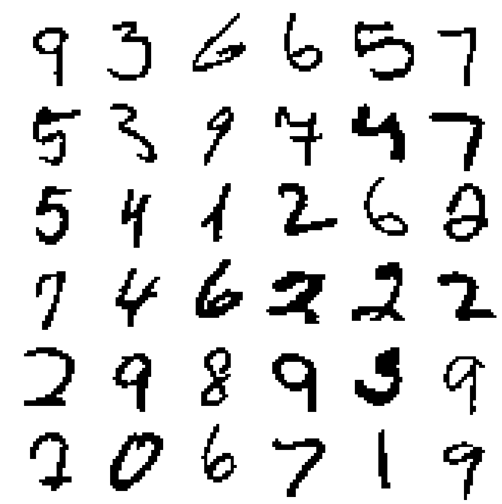


Given data

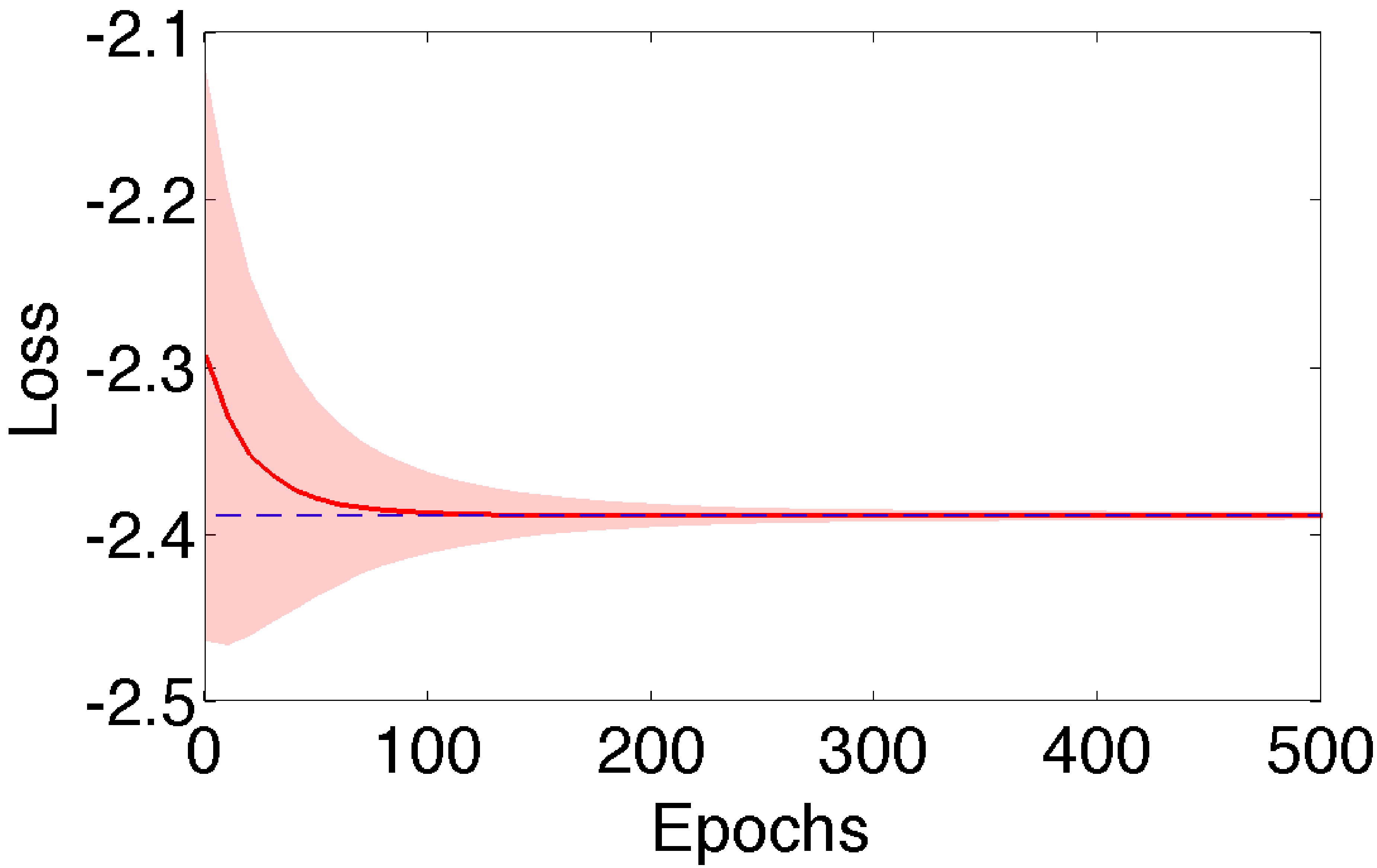
$p_{\text{ data}}(\mathbf{s}) \propto \sum_{i \in \text{ data}} \delta(\mathbf{s} - \mathbf{s}_i)$ Empirical data distribution

$D_{\text{KL}}(p_{\text{data}} \| q_\theta)$ Minimizing KL

Stat. Mech. vs. Density Estimation

Statistical Mechanics		Density Estimation [e.g. ***PixelCNN, Van den Oord et. al., 2014***]	
			
Given energy function	$E(\mathbf{s})$		Given data
Boltzmann distribution	$p_{\text{Boltzmann}}(\mathbf{s})$	$p_{\text{ data }}(\mathbf{s}) \propto \sum_{i \in \text{ data}} \delta(\mathbf{s} - \mathbf{s}_i)$	Empirical data distribution
Minimizing reverse KL	$D_{\text{KL}}(q_\theta \Vert p_{\text{Boltzmann}})$	$D_{\text{KL}}(p_{\text{data}} \Vert q_\theta)$	Minimizing KL
Minimizing variational free energy	$F_q = \langle E \rangle_q - \frac{1}{\beta} S_q$	$-\mathcal{L} = -\log(q(\mathbf{s}_{\text{data}}))$	Minimizing negative log likelihood

Typical Training Process



$$F_q = \mathbb{E}_{\mathbf{s}\sim q_\theta(\mathbf{s})}\left[E(\mathbf{s}) + \frac{1}{\beta}\ln q_\theta(\mathbf{s})\right].$$

Red line: variational free energy, i.e. mean value of loss

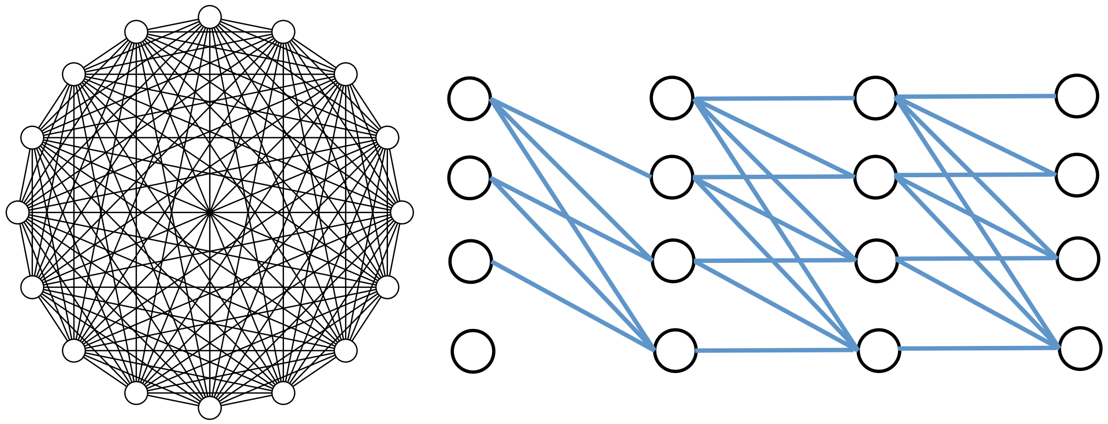
Red area: variance of R(s)

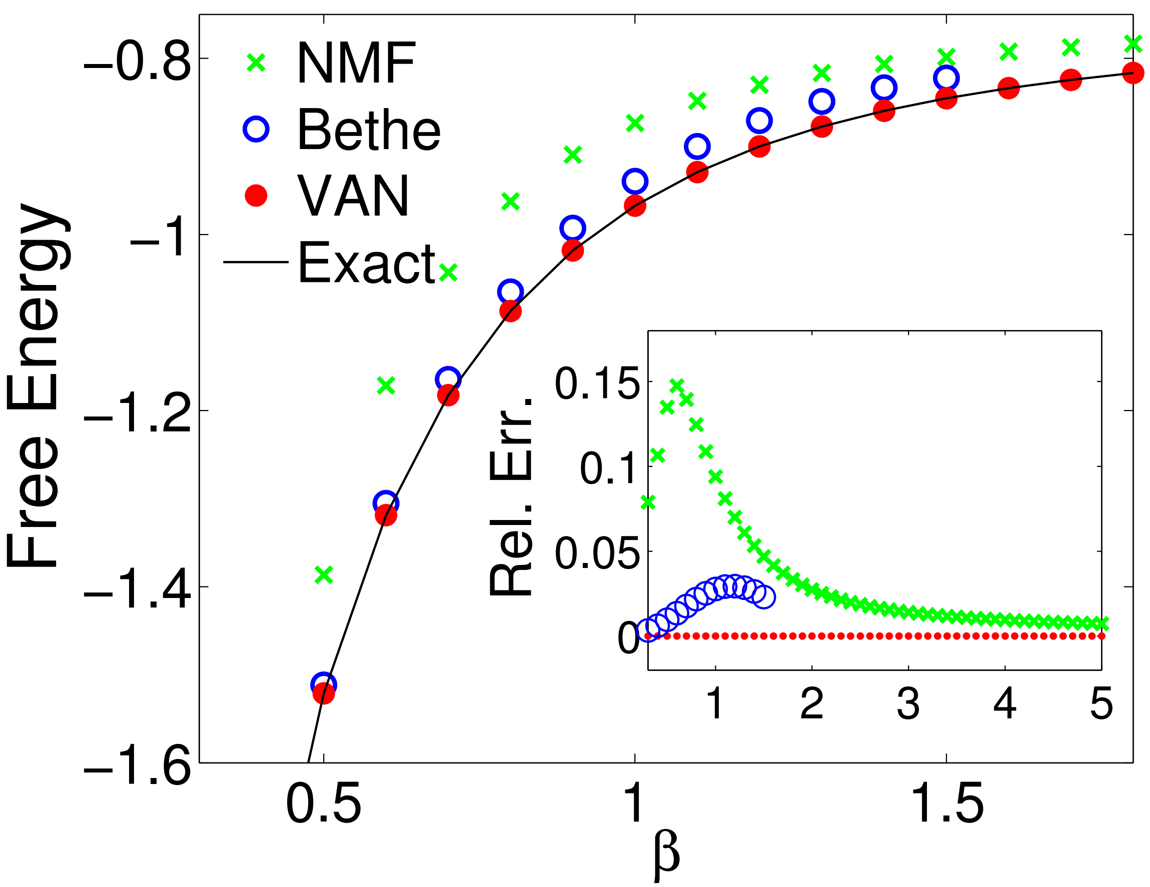
Blue dashed line: exact value obtained by enumerations

D.Wu, L. Wang, PZ, *Phys. Rev. Lett.* 122,080602 (2019)

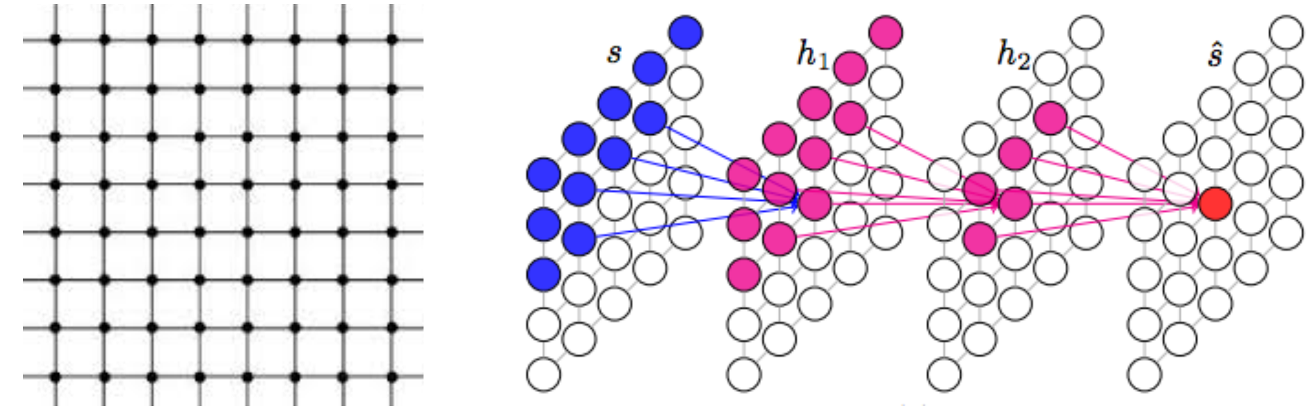
Results

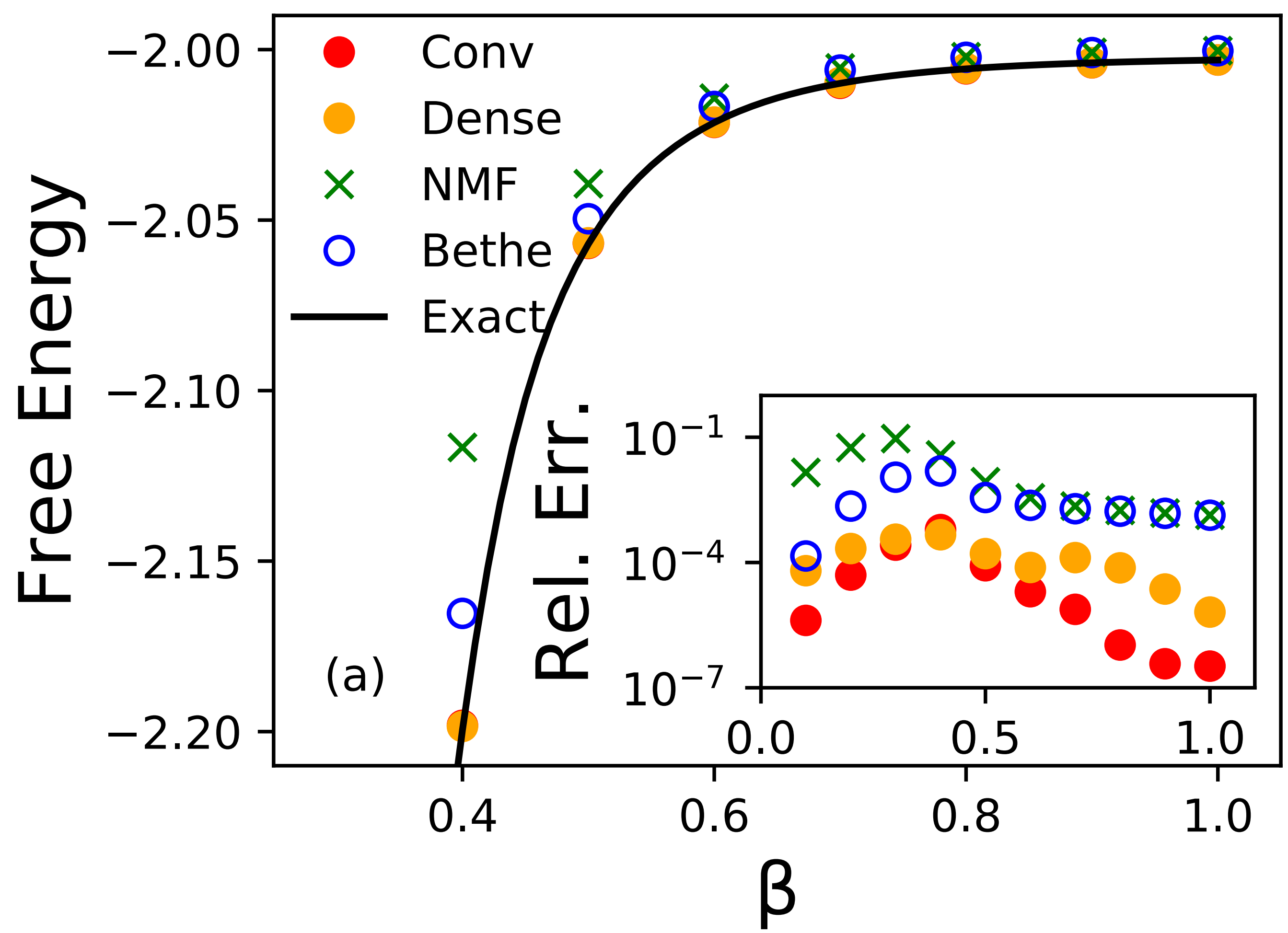
D.Wu, L. Wang, PZ, *Phys. Rev. Lett.* 122,080602 (2019)





Sherrington Kirkpatrik Model
n=20 spins





2D Ising Model
16 x 16

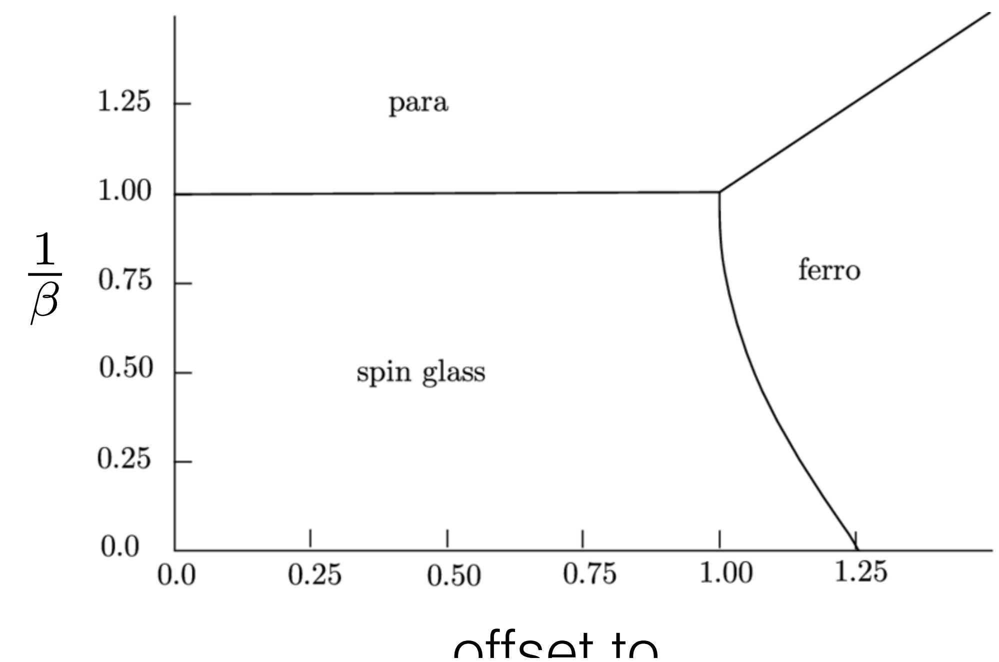
Sherrington-Kirkpatrick model

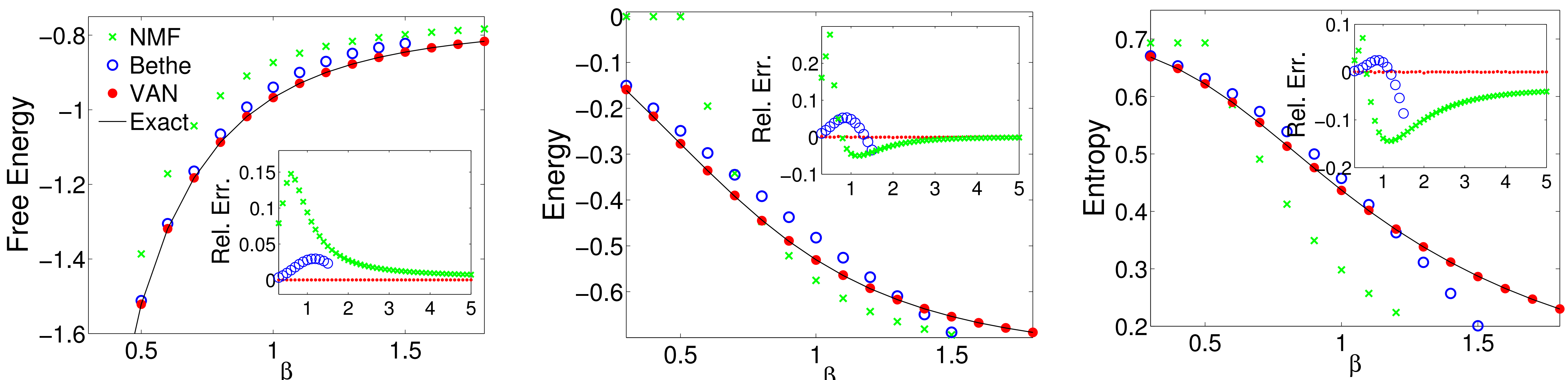
SK model, N=20 spins

[*Sherrington/Kirkpatrick 1975*]

$$E(\mathbf{s}) = -\sum_{(ij)} J_{ij} s_i s_j$$

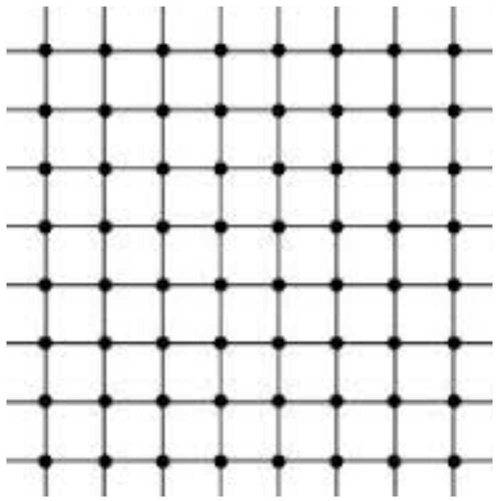
$$J_{ij} \sim \mathcal{N}(0, 1/N)$$

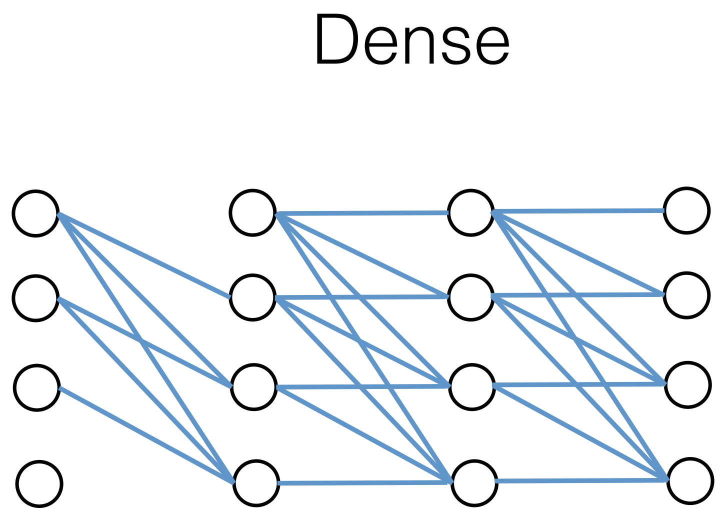


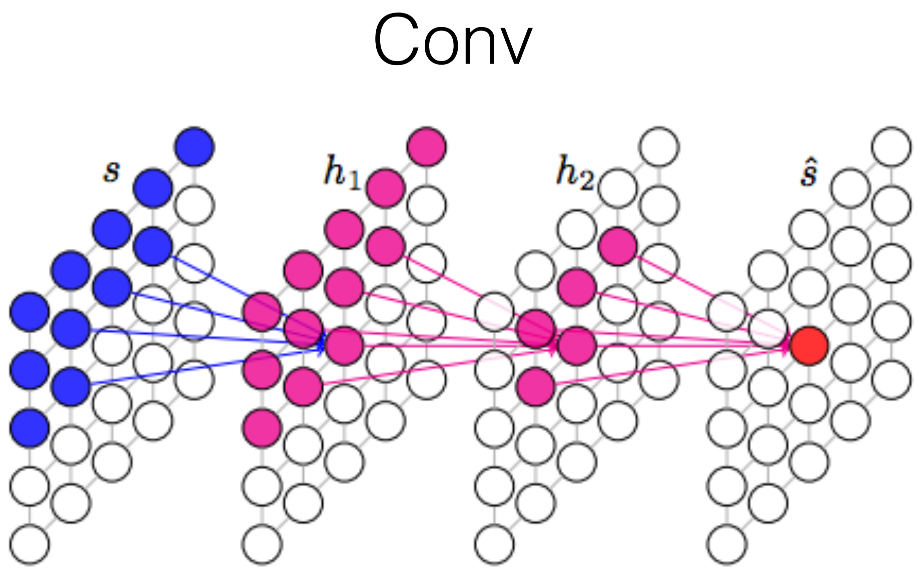


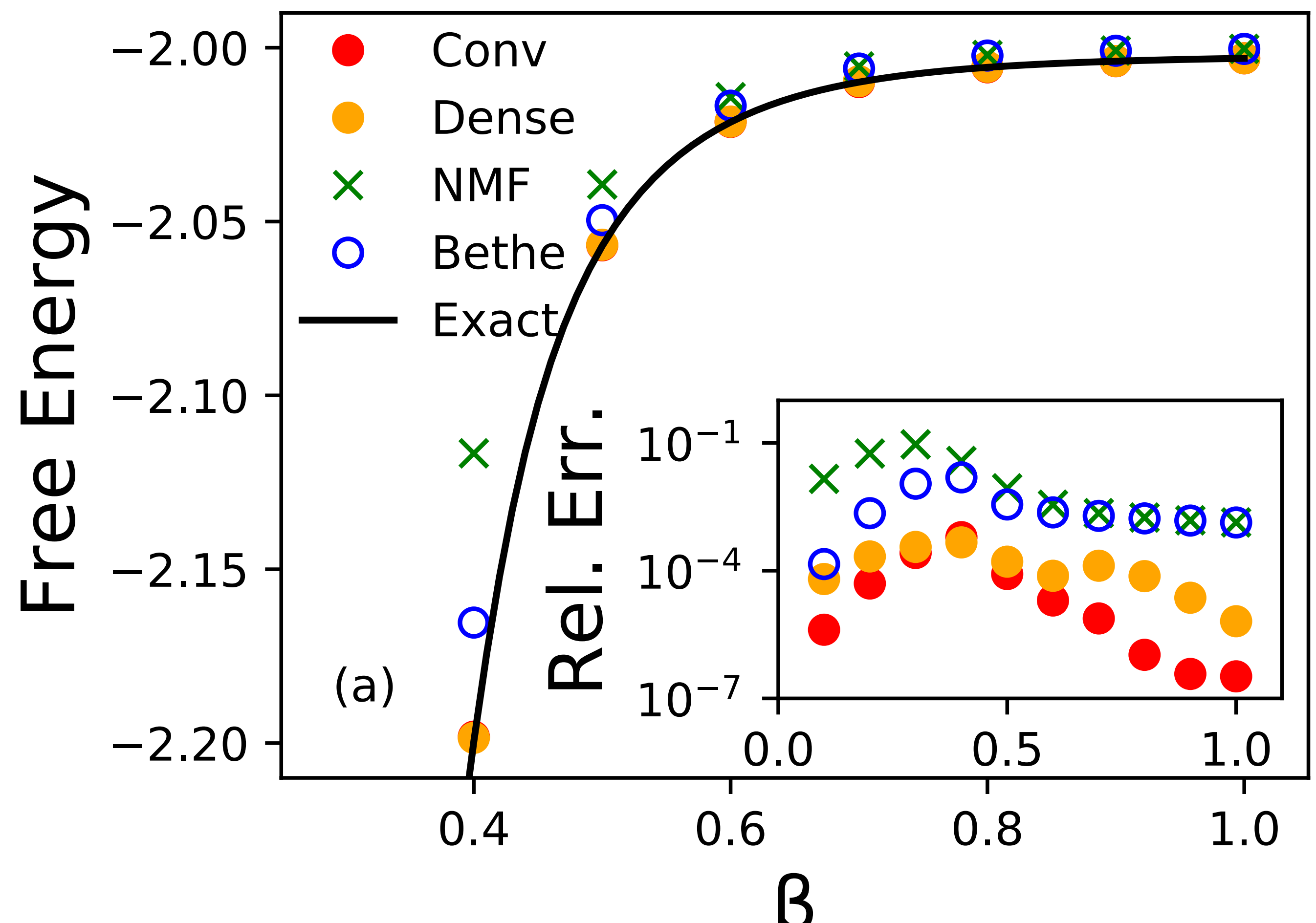
D.Wu, L. Wang, PZ, *Phys. Rev. Lett.* 122, 080602 (2019)

Ising model on 2-D lattice









D.Wu, L. Wang, PZ, *Phys. Rev. Lett.* 122, 080602 (2019)

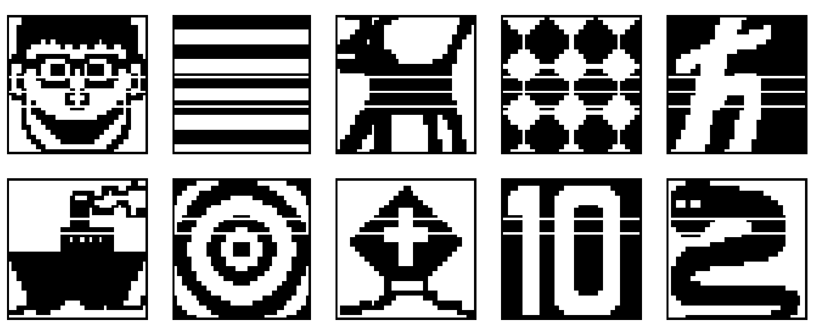
Hopfield Model

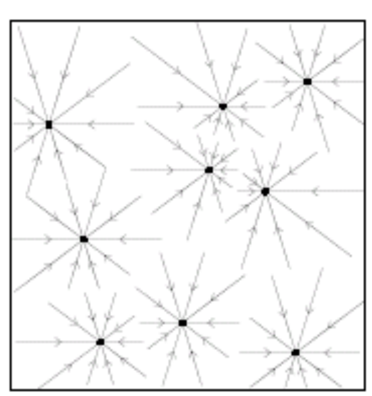
A classic associative memory [Hopfield 1982]

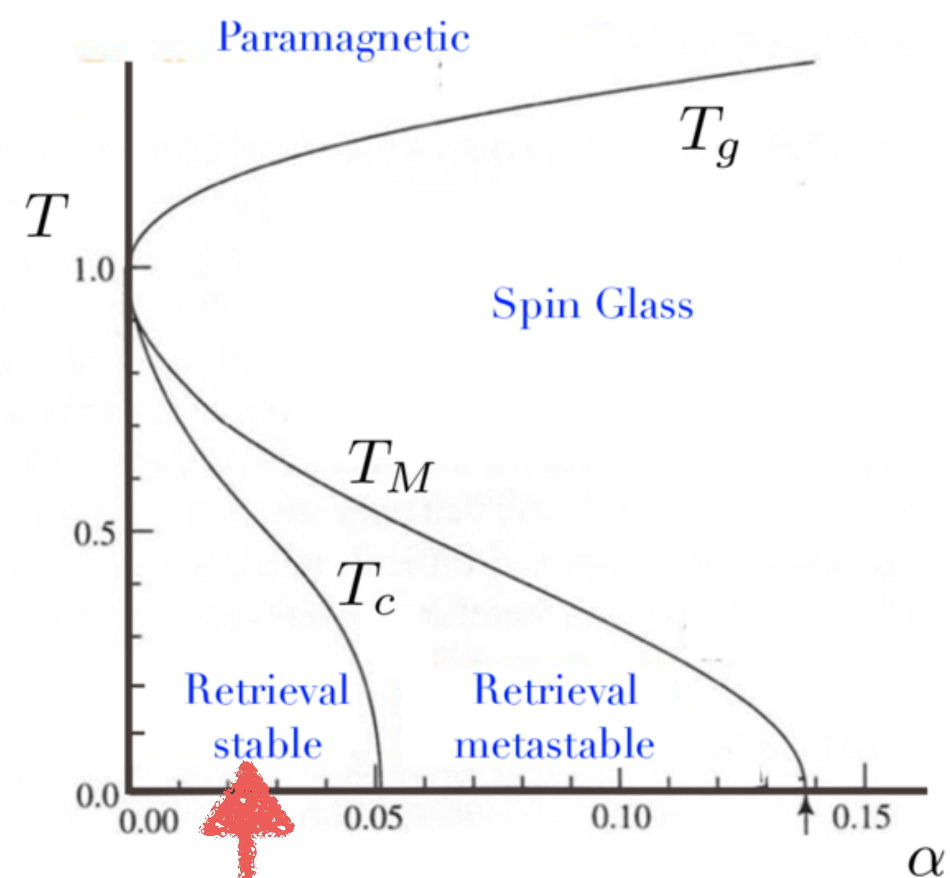
$$E(\mathbf{s}) = -\sum_{(ij)} J_{ij} s_i s_j$$

$$J_{ij} = \frac{1}{N} \sum_{\mu=1}^{\alpha N} \xi_i^\mu \xi_j^\mu$$

$$\xi^\mu \in \{+1, -1\}^N$$



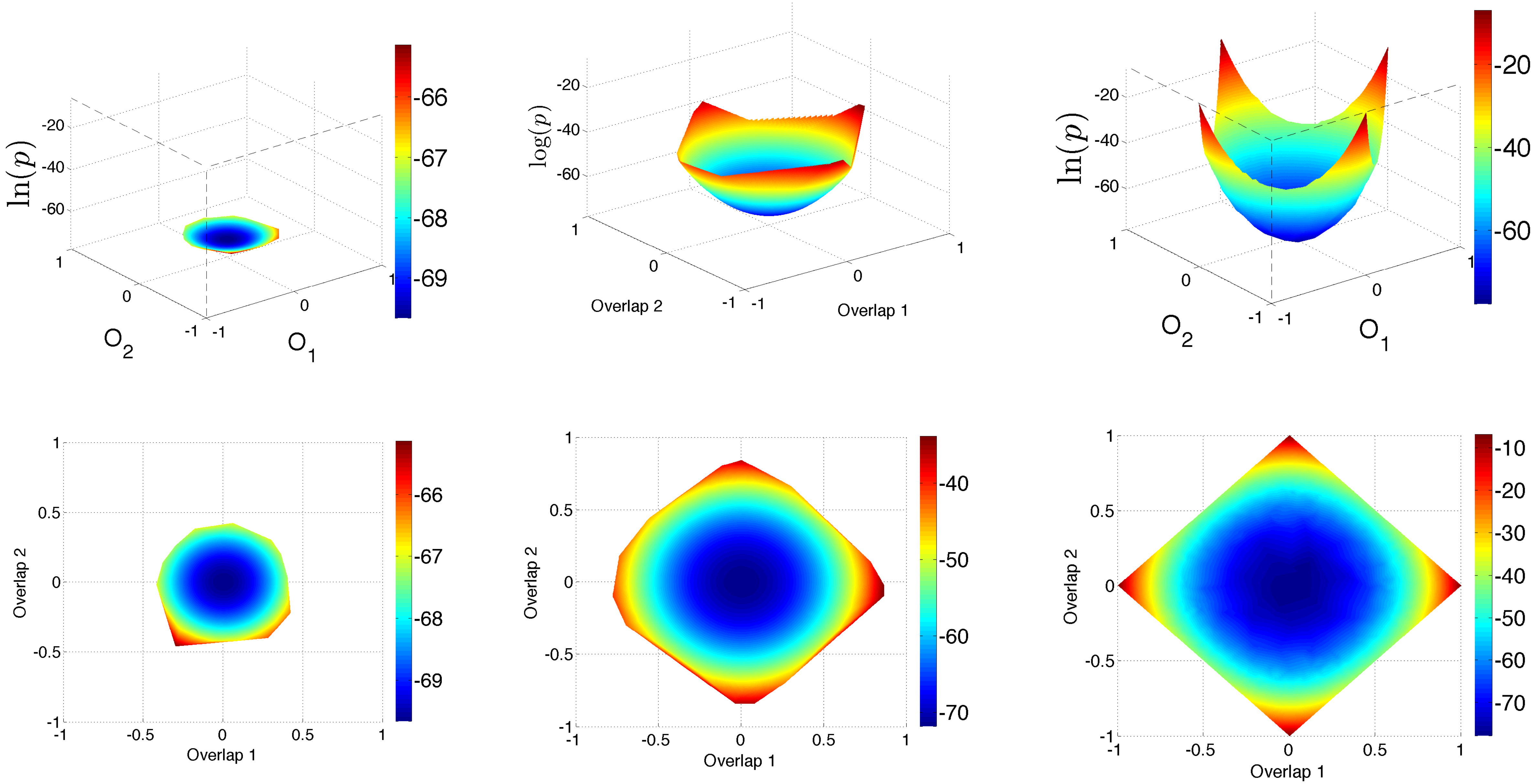




[Amit/Gutfreund/Sompolinsky 1985]

Multi-modal

Sampling



$\beta = 0.3$ $\beta = 1.0$ $\beta = 1.5$

D.Wu, L. Wang, PZ, *Phys. Rev. Lett.* 122,080602 (2019)

Methods for statistical mechanics

MCMC:

- Integrate E over β
- Histogram-based [Wang-Landau]

Variational:

- Mean-field
- Variational Autoregressive Networks

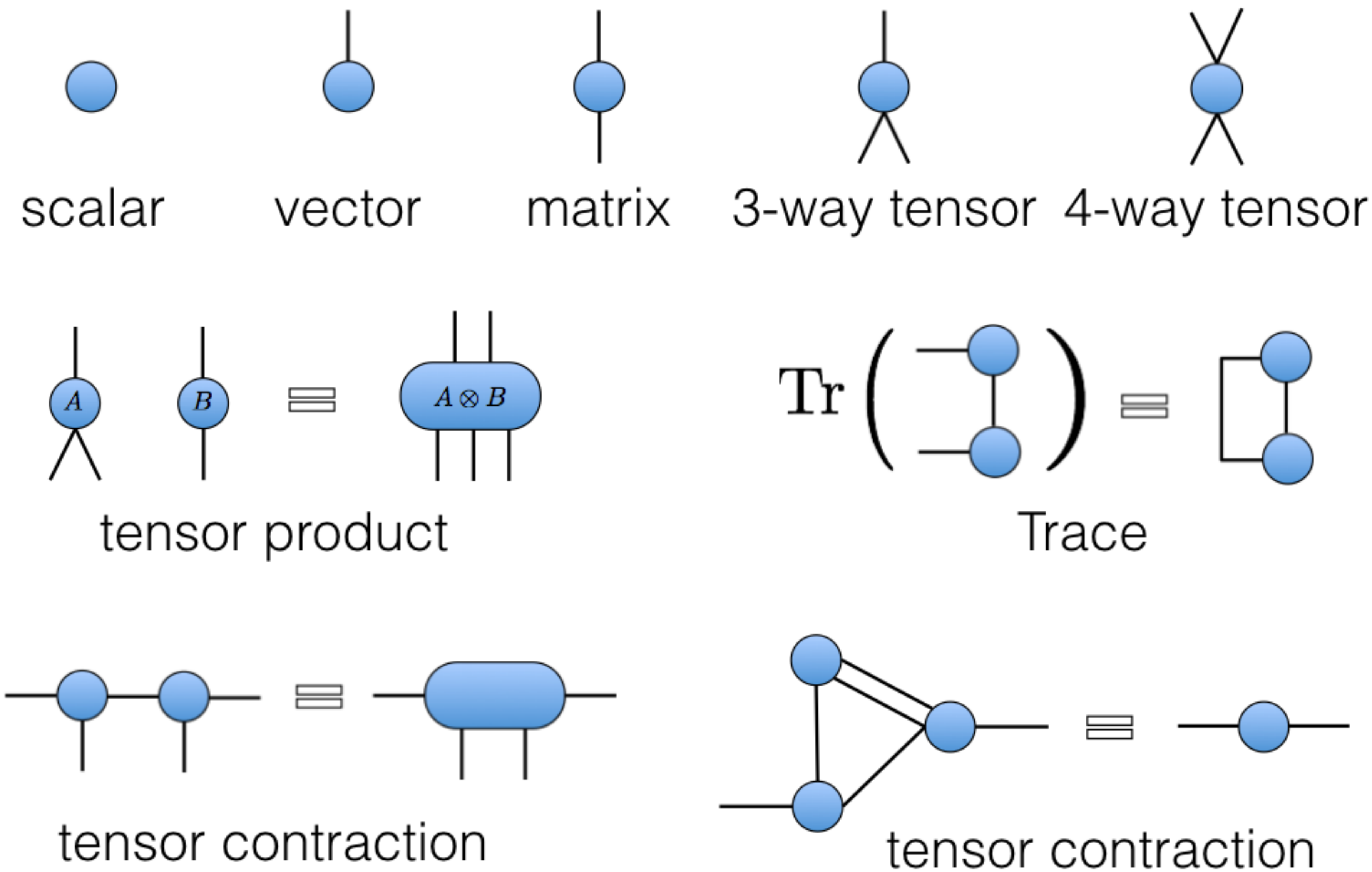
Direct computation:

- RG
- Tensor Networks

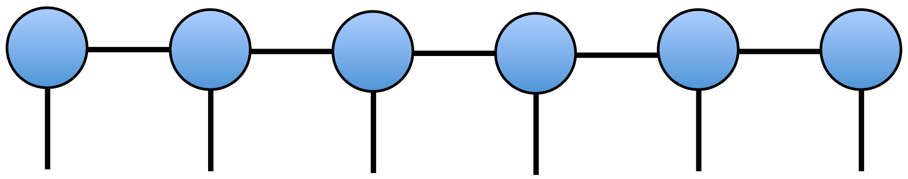
Tensor Networks

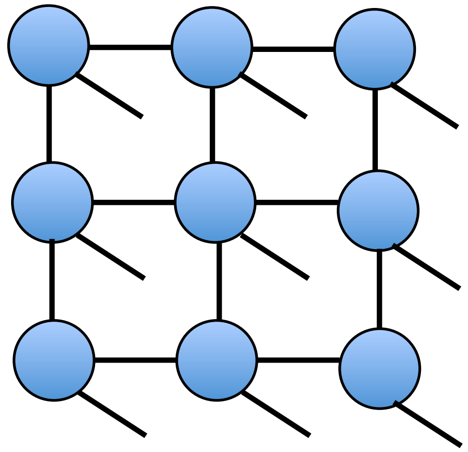
- In physics: wave functions
- Out of physics:
  - Revealing internal low-rank structures (CP, Tucker, TT rank)
  - Compression data, optimization
  - Learning: (kerneled) classification, generative modeling
  - Inference in graphical models
  - Simulating quantum circuits

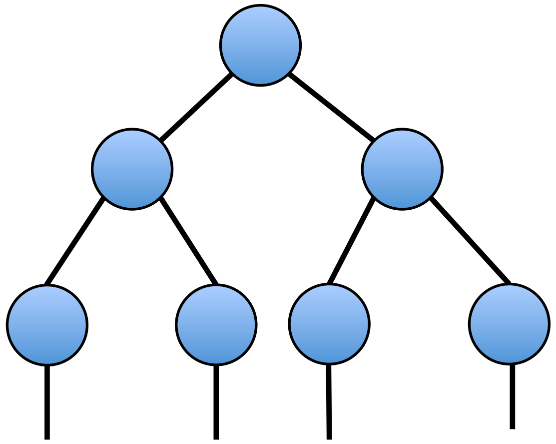
Diagram notations

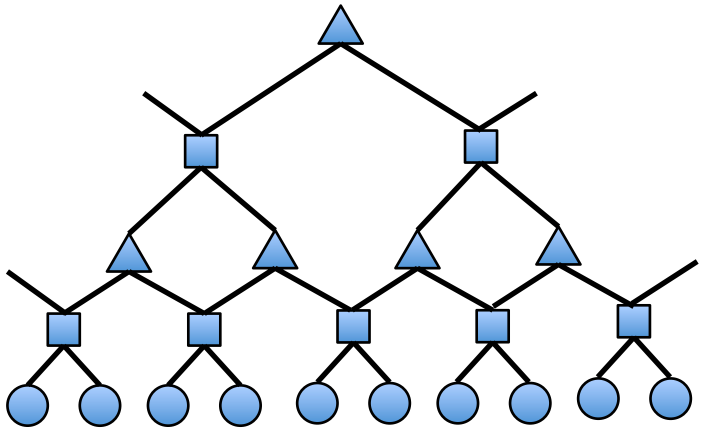


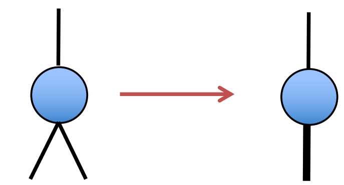
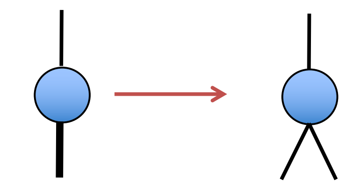
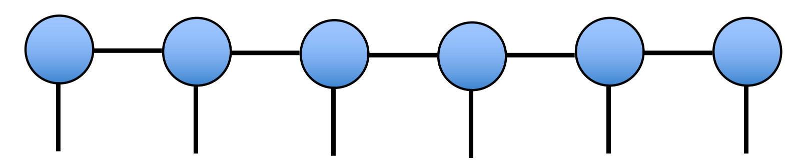
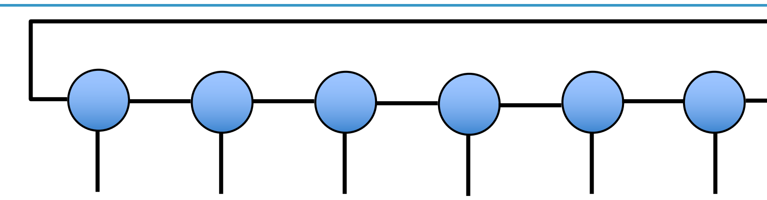
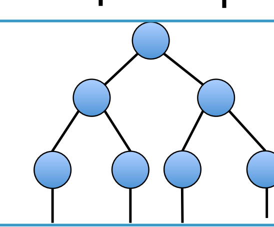
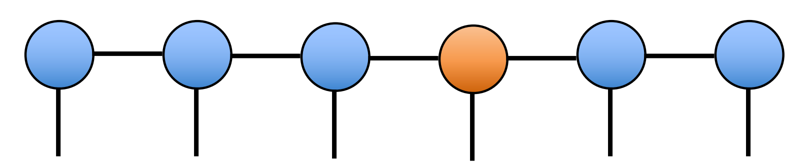
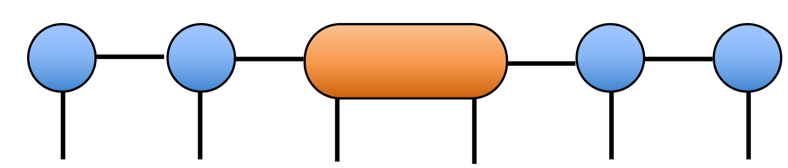
Tensor networks in physics: imposing prior of physical wave functions



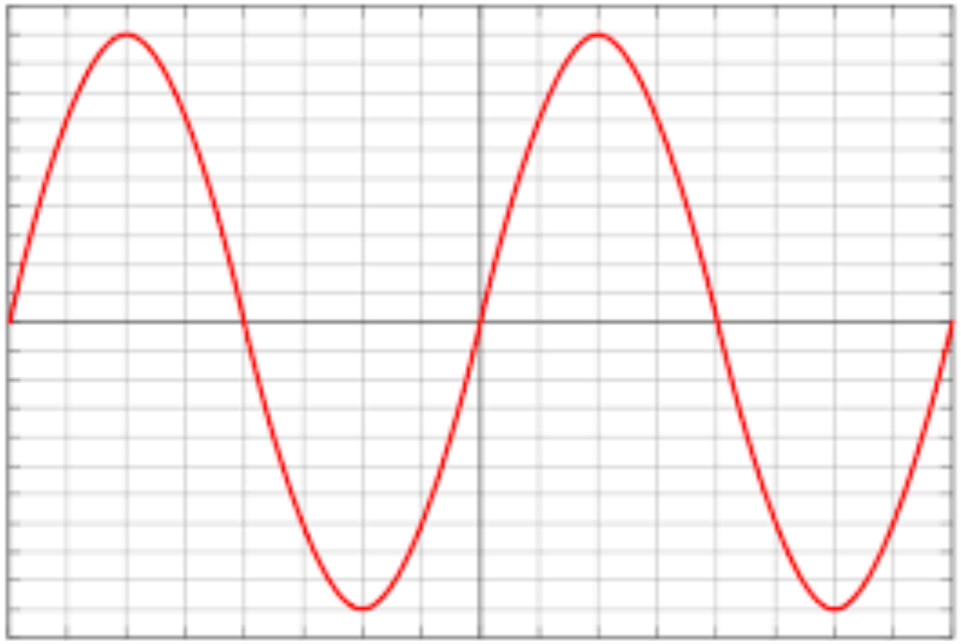






In Physics	Out of Physics	Diagram
grouping of indices	unfolding, matricization	
splitting of indices	tensorizing	
matrix product states	tensor train decomposition	
periodic boundary MPS	tensor chain decomposition	
tree tensor networks	hierarchical Tucker decompostion	
single-site DMRG	alternating least square	
two-site DMRG	modified alternating least square	

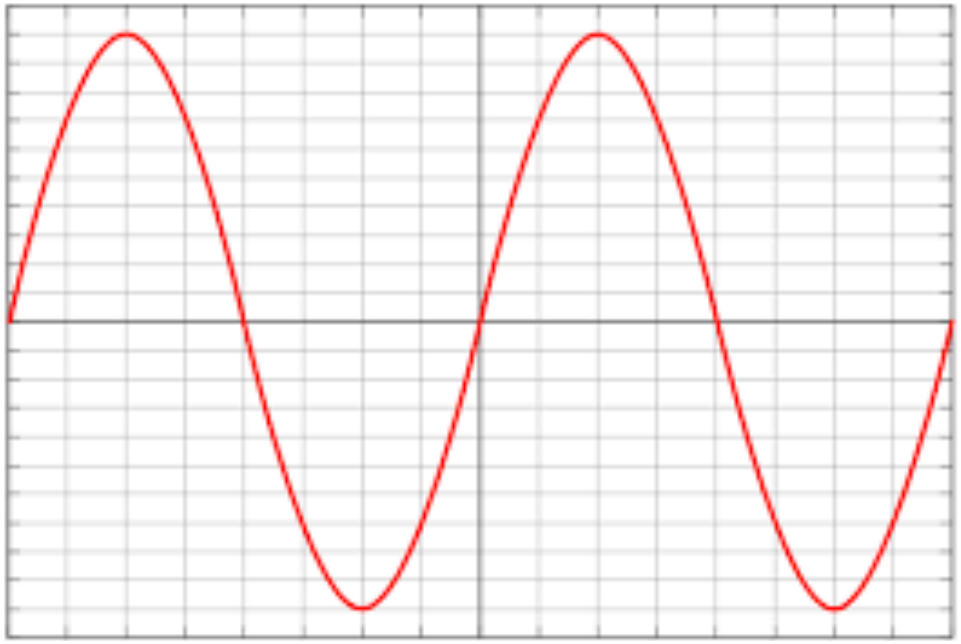
Exploring internal structures in the data



[0.000, 0.100, 0.199, 0.296, 0.389, 0.479, 0.565, 0.644, 0.717, 0.783,
0.841, 0.891, 0.932, 0.964, 0.985, 0.997, 1.000, 0.992, 0.974, 0.946,
0.909, 0.863, 0.808, 0.746, 0.675, 0.598, 0.516, 0.427, 0.335, 0.239,
0.141, 0.042, -0.058, -0.158, -0.256, -0.351, -0.443, -0.530, -0.612,
......
......
-0.688, -0.757, -0.818, -0.872, -0.916, -0.952, -0.978, -0.994, -1.000,
-0.996, -0.982, -0.959, -0.926, -0.883, -0.832, -0.773, -0.706, -0.631,
-0.551, -0.465, -0.374, -0.279, -0.182, -0.083, 0.017, 0.117, 0.215,
0.312, 0.405, 0.494, 0.578, 0.657, 0.729, 0.794, 0.850, 0.899, 0.938]

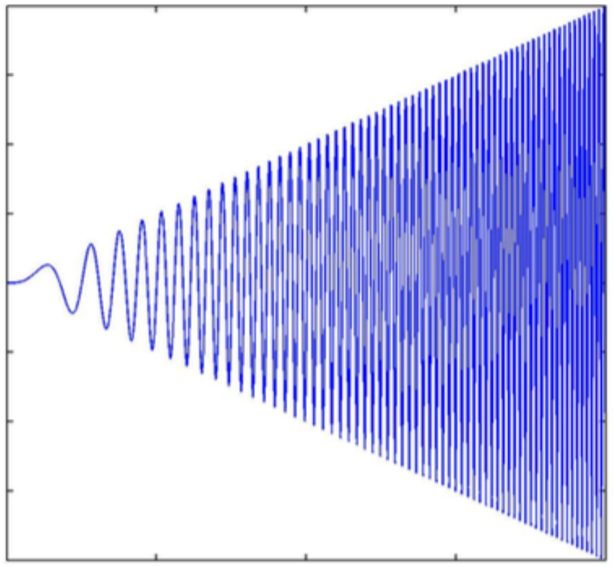
10^6 data points in a vector

Exploring internal structures in the data

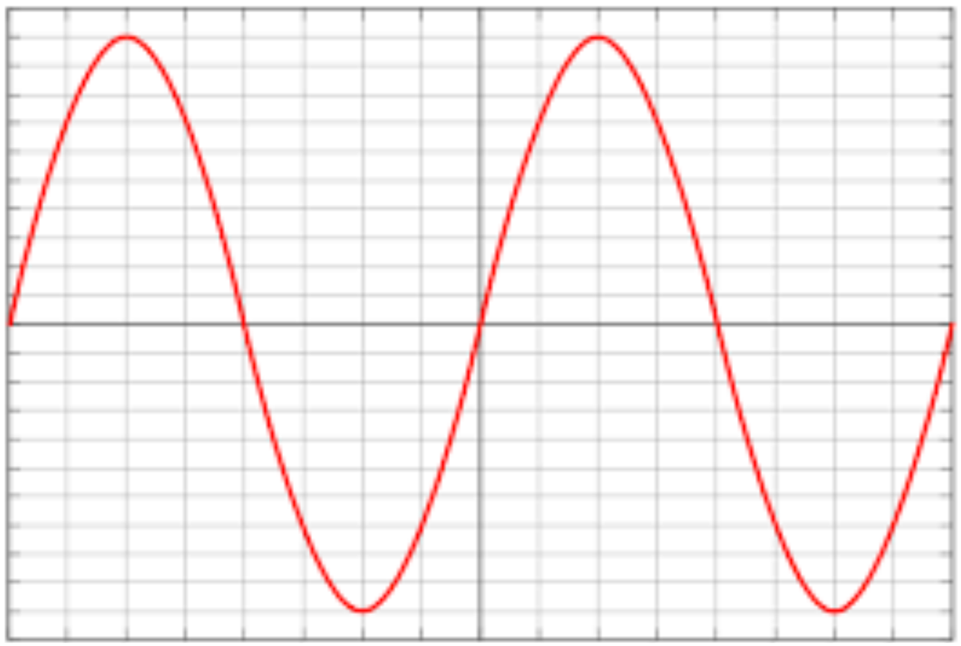


[0.000, 0.100, 0.199, 0.296, 0.389, 0.479, 0.565, 0.644, 0.717, 0.783,
0.841, 0.891, 0.932, 0.964, 0.985, 0.997, 1.000, 0.992, 0.974, 0.946,
0.909, 0.863, 0.808, 0.746, 0.675, 0.598, 0.516, 0.427, 0.335, 0.239,
0.141, 0.042, -0.058, -0.158, -0.256, -0.351, -0.443, -0.530, -0.612,
......
......
-0.688, -0.757, -0.818, -0.872, -0.916, -0.952, -0.978, -0.994, -1.000,
-0.996, -0.982, -0.959, -0.926, -0.883, -0.832, -0.773, -0.706, -0.631,
-0.551, -0.465, -0.374, -0.279, -0.182, -0.083, 0.017, 0.117, 0.215,
0.312, 0.405, 0.494, 0.578, 0.657, 0.729, 0.794, 0.850, 0.899, 0.938]

10^6 data points in a vector

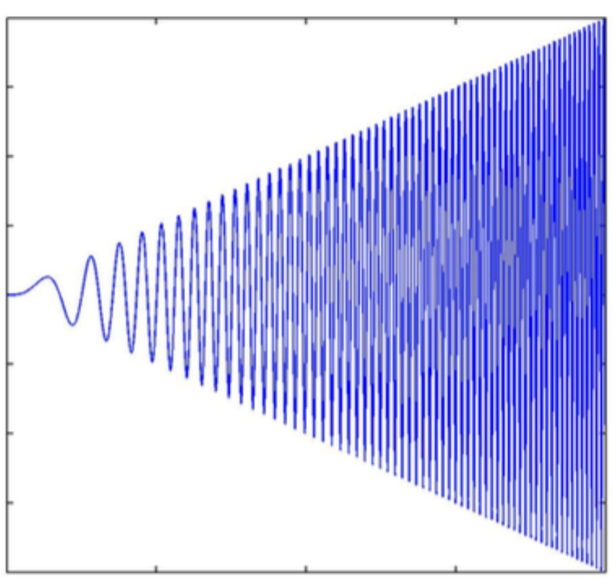


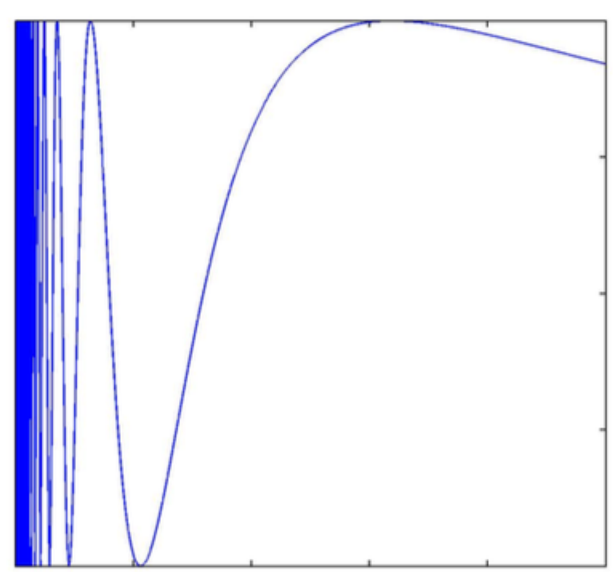
Exploring internal structures in the data



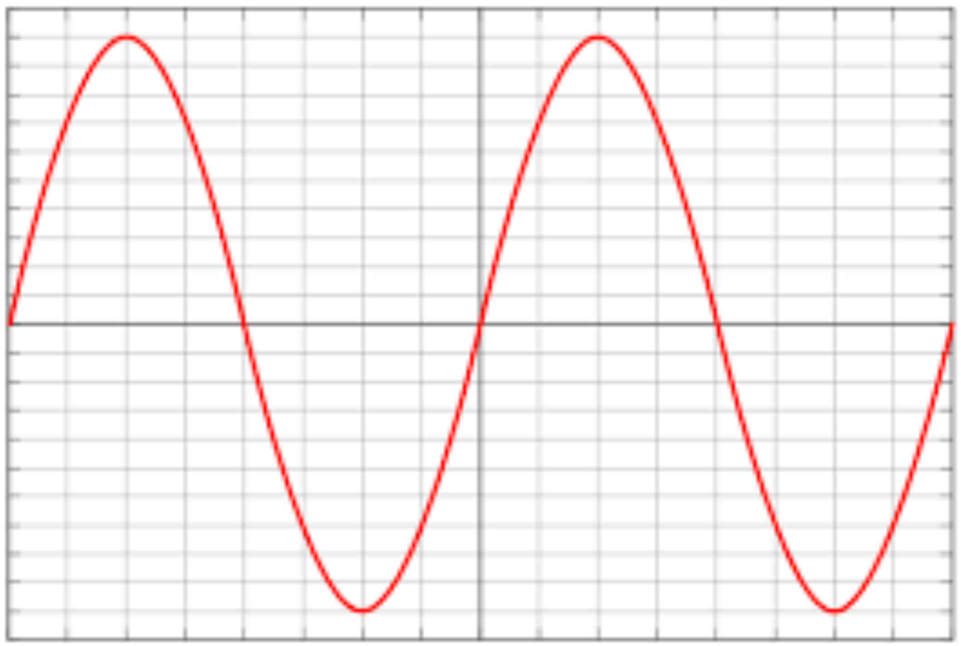
[0.000, 0.100, 0.199, 0.296, 0.389, 0.479, 0.565, 0.644, 0.717, 0.783,
0.841, 0.891, 0.932, 0.964, 0.985, 0.997, 1.000, 0.992, 0.974, 0.946,
0.909, 0.863, 0.808, 0.746, 0.675, 0.598, 0.516, 0.427, 0.335, 0.239,
0.141, 0.042, -0.058, -0.158, -0.256, -0.351, -0.443, -0.530, -0.612,
......
......
-0.688, -0.757, -0.818, -0.872, -0.916, -0.952, -0.978, -0.994, -1.000,
-0.996, -0.982, -0.959, -0.926, -0.883, -0.832, -0.773, -0.706, -0.631,
-0.551, -0.465, -0.374, -0.279, -0.182, -0.083, 0.017, 0.117, 0.215,
0.312, 0.405, 0.494, 0.578, 0.657, 0.729, 0.794, 0.850, 0.899, 0.938]

10^6 data points in a vector



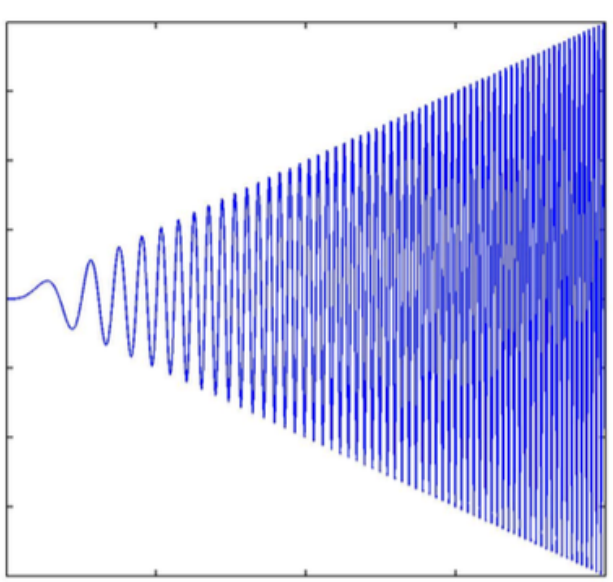


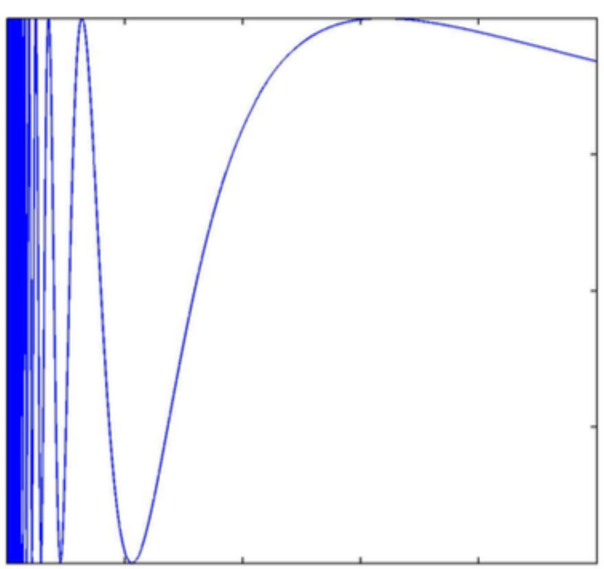
Exploring internal structures in the data

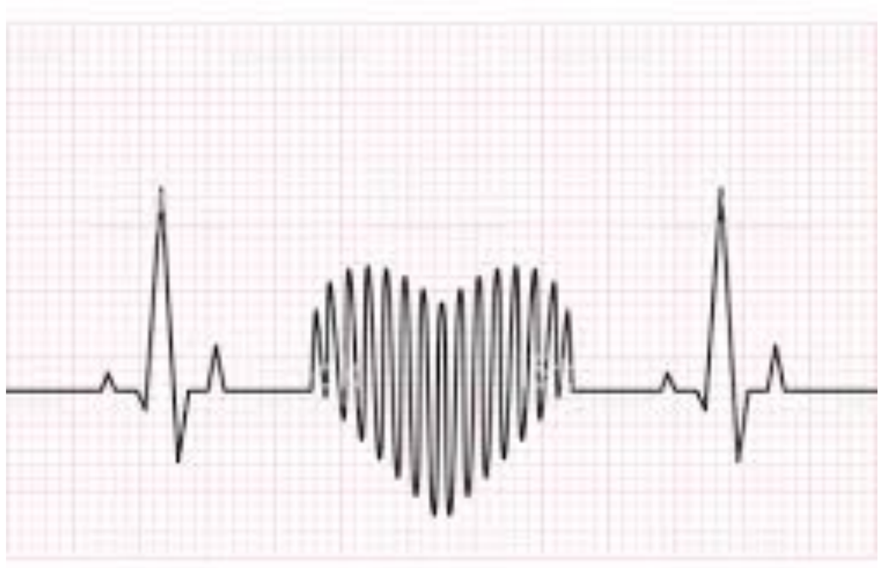


[0.000, 0.100, 0.199, 0.296, 0.389, 0.479, 0.565, 0.644, 0.717, 0.783,
0.841, 0.891, 0.932, 0.964, 0.985, 0.997, 1.000, 0.992, 0.974, 0.946,
0.909, 0.863, 0.808, 0.746, 0.675, 0.598, 0.516, 0.427, 0.335, 0.239,
0.141, 0.042, -0.058, -0.158, -0.256, -0.351, -0.443, -0.530, -0.612,
......
......
-0.688, -0.757, -0.818, -0.872, -0.916, -0.952, -0.978, -0.994, -1.000,
-0.996, -0.982, -0.959, -0.926, -0.883, -0.832, -0.773, -0.706, -0.631,
-0.551, -0.465, -0.374, -0.279, -0.182, -0.083, 0.017, 0.117, 0.215,
0.312, 0.405, 0.494, 0.578, 0.657, 0.729, 0.794, 0.850, 0.899, 0.938]

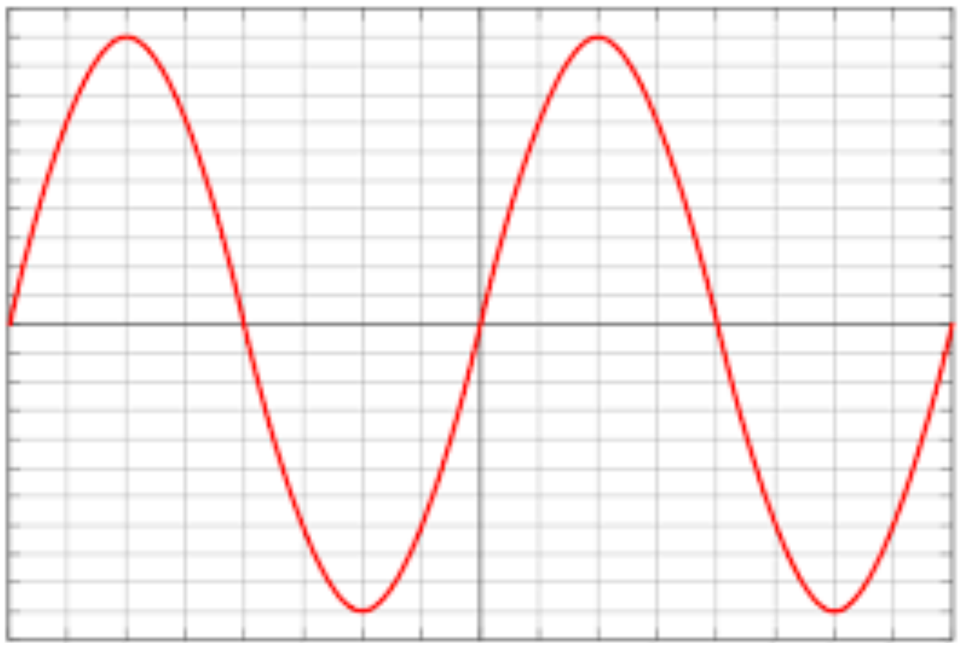
10^6 data points in a vector





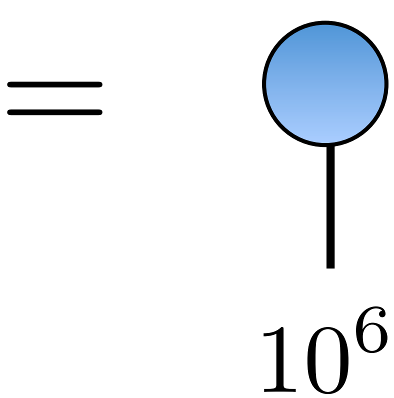


Exploring internal structures in the data

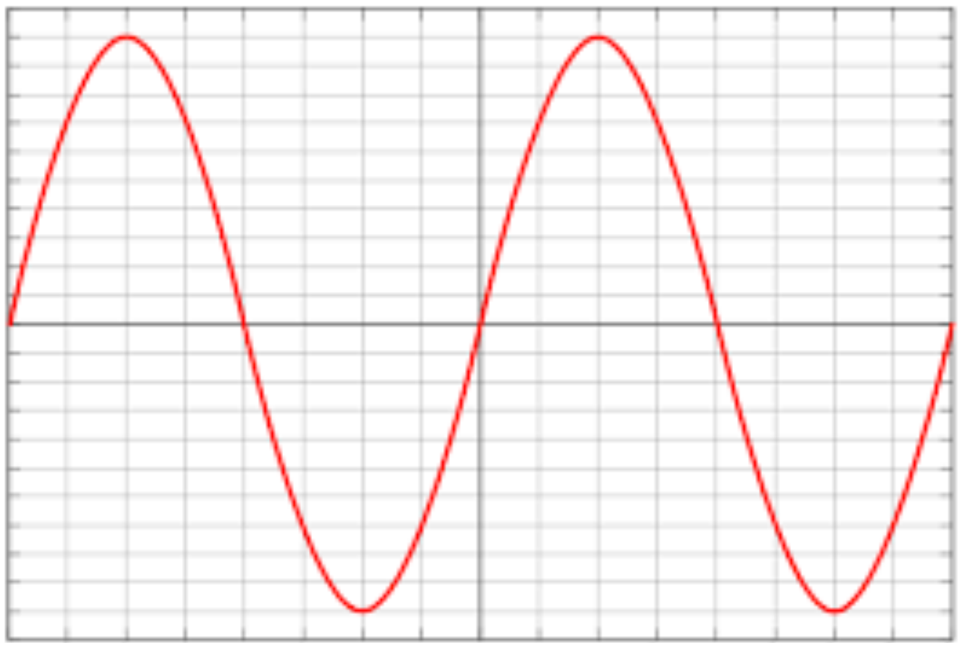


[0.000, 0.100, 0.199, 0.296, 0.389, 0.479, 0.565, 0.644, 0.717, 0.783, 0.841, 0.891, 0.932, 0.964, 0.985, 0.997, 1.000, 0.992, 0.974, 0.946, 0.909, 0.863, 0.808, 0.746, 0.675, 0.598, 0.516, 0.427, 0.335, 0.239, 0.141, 0.042, -0.058, -0.158, -0.256, -0.351, -0.443, -0.530, -0.612,
......
......
-0.688, -0.757, -0.818, -0.872, -0.916, -0.952, -0.978, -0.994, -1.000, -0.996, -0.982, -0.959, -0.926, -0.883, -0.832, -0.773, -0.706, -0.631, -0.551, -0.465, -0.374, -0.279, -0.182, -0.083, 0.017, 0.117, 0.215, 0.312, 0.405, 0.494, 0.578, 0.657, 0.729, 0.794, 0.850, 0.899, 0.938]

10^6 data points in a vector

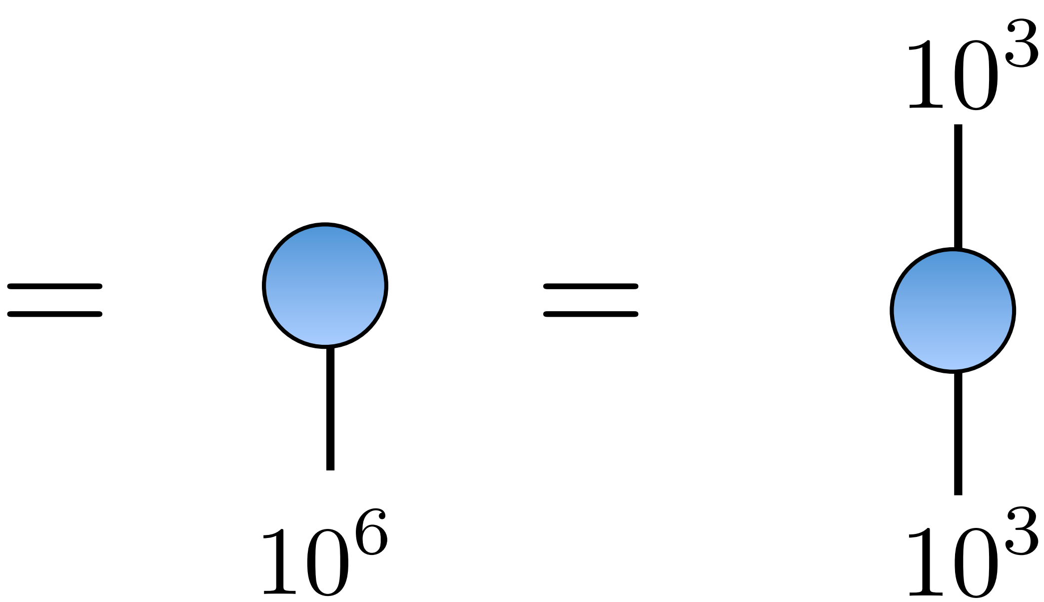


Exploring internal structures in the data

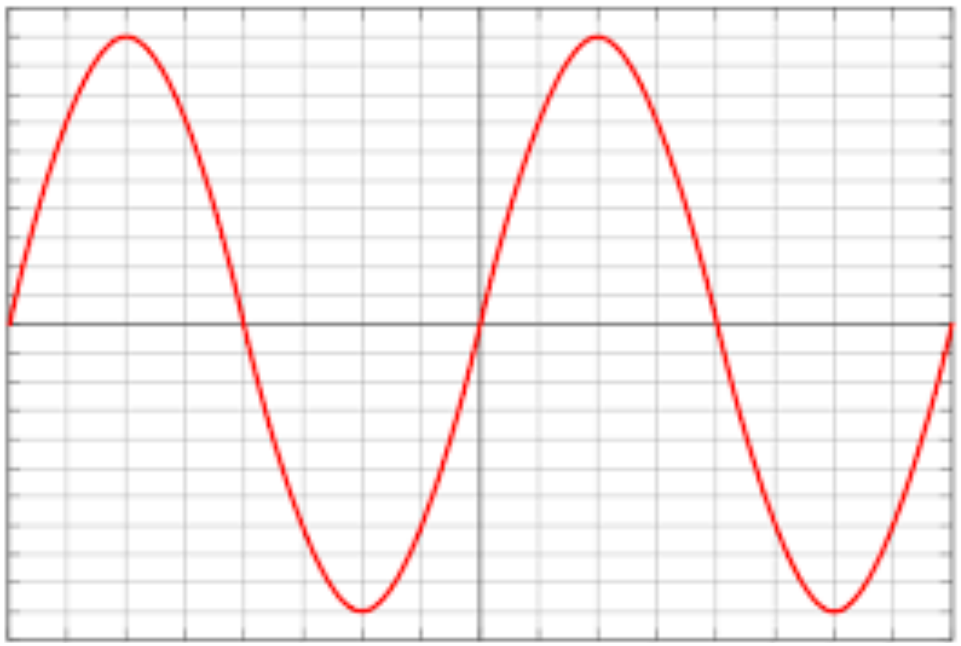


[0.000, 0.100, 0.199, 0.296, 0.389, 0.479, 0.565, 0.644, 0.717, 0.783,
0.841, 0.891, 0.932, 0.964, 0.985, 0.997, 1.000, 0.992, 0.974, 0.946,
0.909, 0.863, 0.808, 0.746, 0.675, 0.598, 0.516, 0.427, 0.335, 0.239,
0.141, 0.042, -0.058, -0.158, -0.256, -0.351, -0.443, -0.530, -0.612,
......
......
-0.688, -0.757, -0.818, -0.872, -0.916, -0.952, -0.978, -0.994, -1.000,
-0.996, -0.982, -0.959, -0.926, -0.883, -0.832, -0.773, -0.706, -0.631,
-0.551, -0.465, -0.374, -0.279, -0.182, -0.083, 0.017, 0.117, 0.215,
0.312, 0.405, 0.494, 0.578, 0.657, 0.729, 0.794, 0.850, 0.899, 0.938]

10^6 data points in a vector

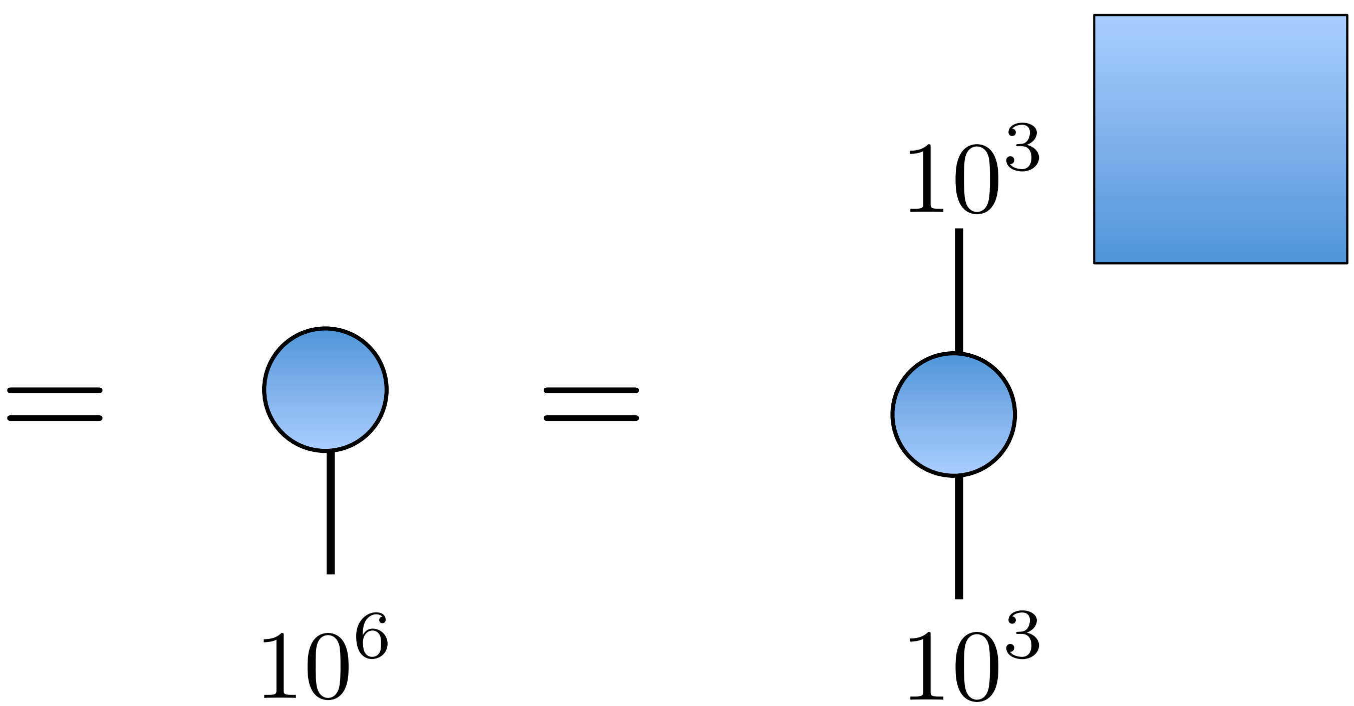


Exploring internal structures in the data

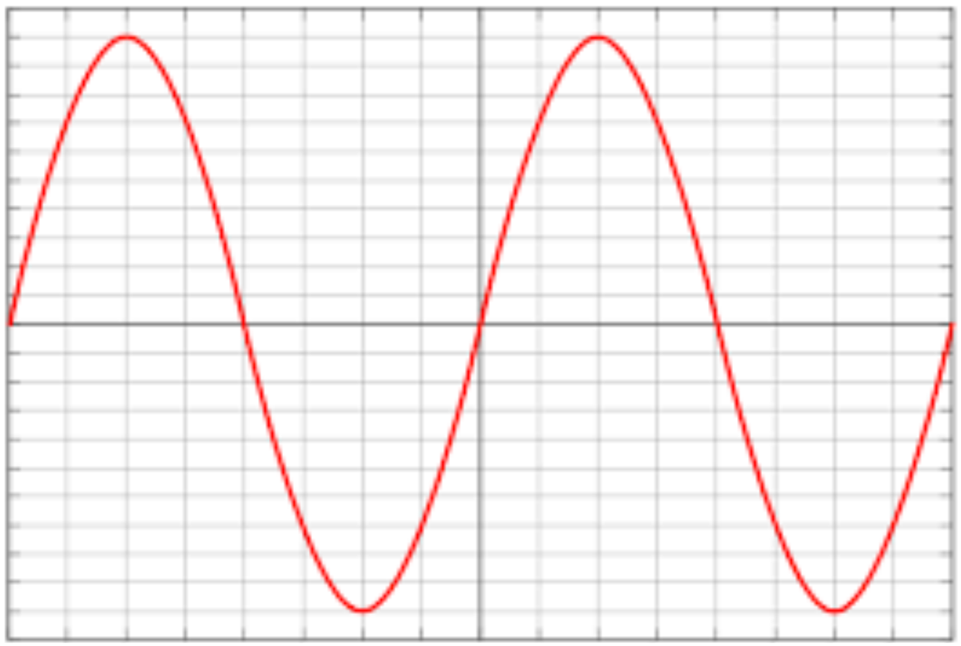


[0.000, 0.100, 0.199, 0.296, 0.389, 0.479, 0.565, 0.644, 0.717, 0.783,
0.841, 0.891, 0.932, 0.964, 0.985, 0.997, 1.000, 0.992, 0.974, 0.946,
0.909, 0.863, 0.808, 0.746, 0.675, 0.598, 0.516, 0.427, 0.335, 0.239,
0.141, 0.042, -0.058, -0.158, -0.256, -0.351, -0.443, -0.530, -0.612,
......
......
-0.688, -0.757, -0.818, -0.872, -0.916, -0.952, -0.978, -0.994, -1.000,
-0.996, -0.982, -0.959, -0.926, -0.883, -0.832, -0.773, -0.706, -0.631,
-0.551, -0.465, -0.374, -0.279, -0.182, -0.083, 0.017, 0.117, 0.215,
0.312, 0.405, 0.494, 0.578, 0.657, 0.729, 0.794, 0.850, 0.899, 0.938]

10^6 data points in a vector

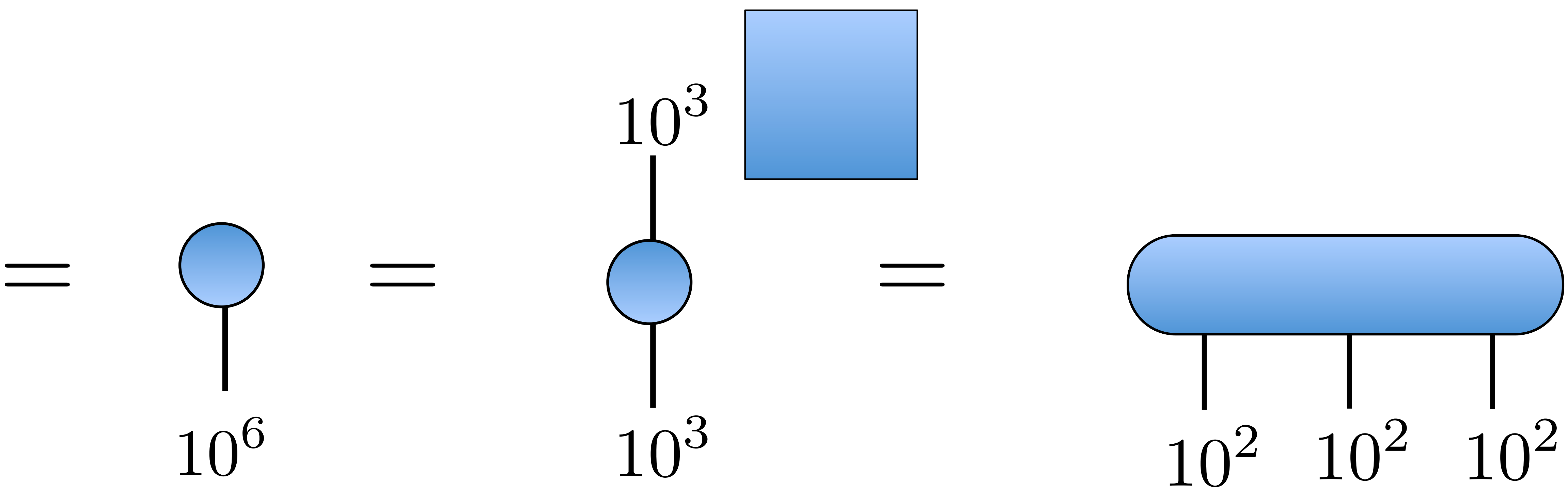


Exploring internal structures in the data

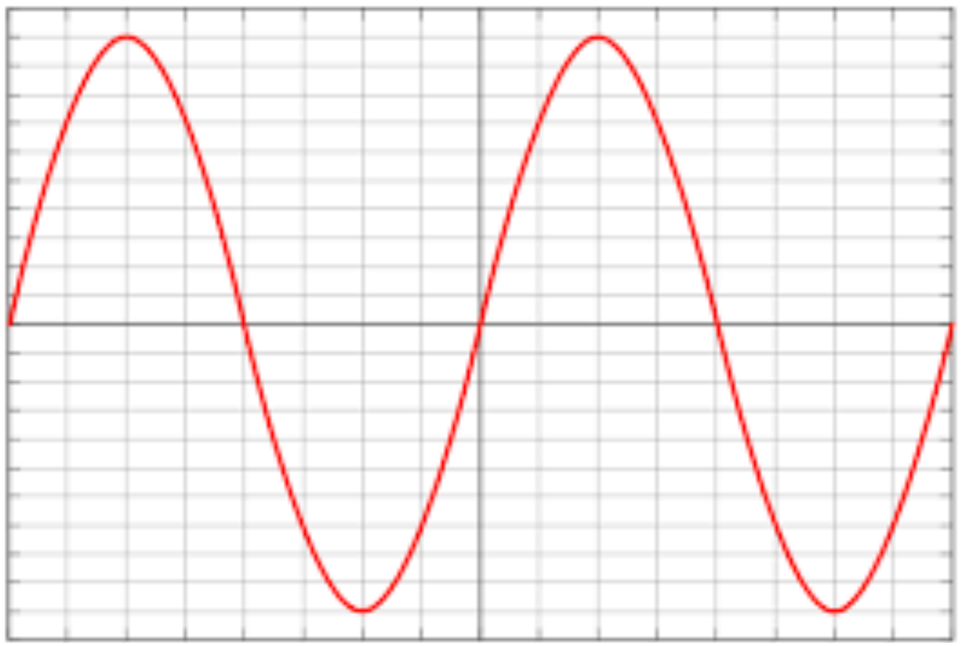


[0.000, 0.100, 0.199, 0.296, 0.389, 0.479, 0.565, 0.644, 0.717, 0.783,
0.841, 0.891, 0.932, 0.964, 0.985, 0.997, 1.000, 0.992, 0.974, 0.946,
0.909, 0.863, 0.808, 0.746, 0.675, 0.598, 0.516, 0.427, 0.335, 0.239,
0.141, 0.042, -0.058, -0.158, -0.256, -0.351, -0.443, -0.530, -0.612,
......
......
-0.688, -0.757, -0.818, -0.872, -0.916, -0.952, -0.978, -0.994, -1.000,
-0.996, -0.982, -0.959, -0.926, -0.883, -0.832, -0.773, -0.706, -0.631,
-0.551, -0.465, -0.374, -0.279, -0.182, -0.083, 0.017, 0.117, 0.215,
0.312, 0.405, 0.494, 0.578, 0.657, 0.729, 0.794, 0.850, 0.899, 0.938]

10^6 data points in a vector

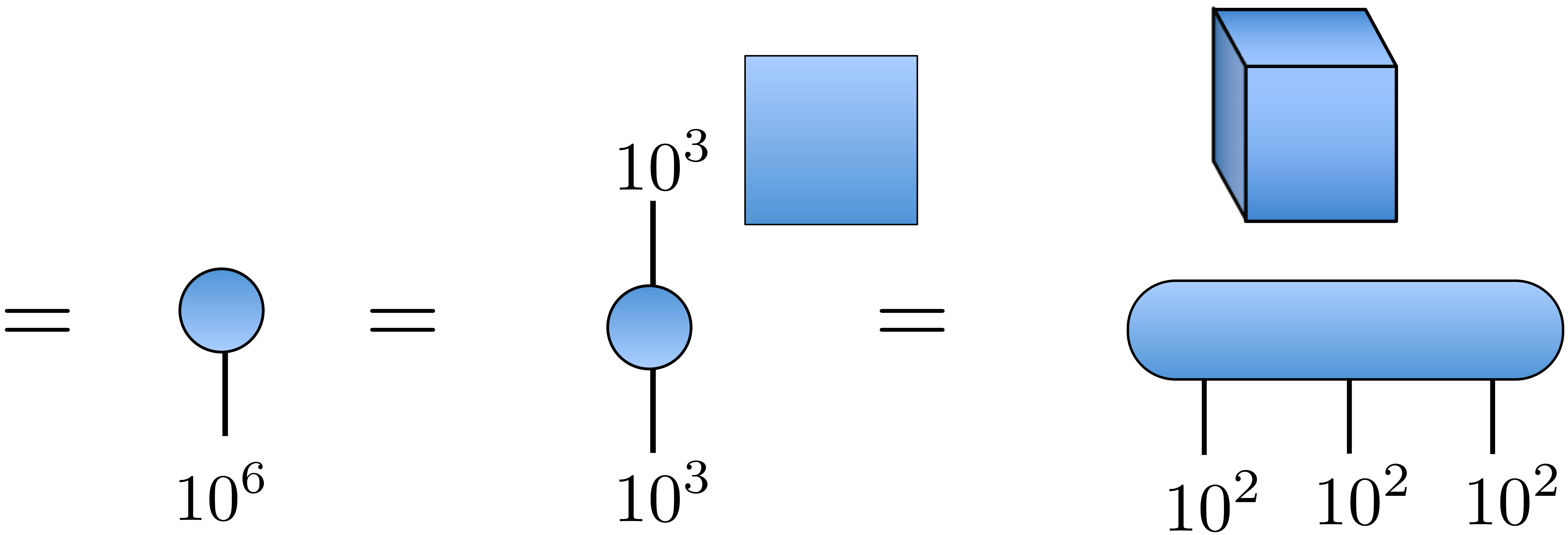


Exploring internal structures in the data

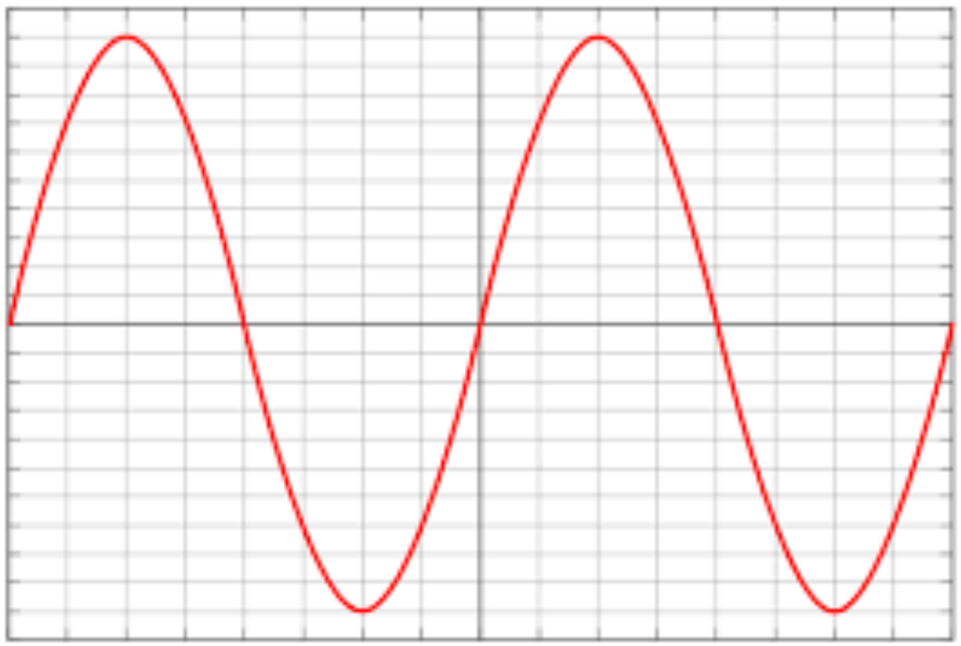


[0.000, 0.100, 0.199, 0.296, 0.389, 0.479, 0.565, 0.644, 0.717, 0.783,
0.841, 0.891, 0.932, 0.964, 0.985, 0.997, 1.000, 0.992, 0.974, 0.946,
0.909, 0.863, 0.808, 0.746, 0.675, 0.598, 0.516, 0.427, 0.335, 0.239,
0.141, 0.042, -0.058, -0.158, -0.256, -0.351, -0.443, -0.530, -0.612,
......
......
-0.688, -0.757, -0.818, -0.872, -0.916, -0.952, -0.978, -0.994, -1.000,
-0.996, -0.982, -0.959, -0.926, -0.883, -0.832, -0.773, -0.706, -0.631,
-0.551, -0.465, -0.374, -0.279, -0.182, -0.083, 0.017, 0.117, 0.215,
0.312, 0.405, 0.494, 0.578, 0.657, 0.729, 0.794, 0.850, 0.899, 0.938]

10^6 data points in a vector

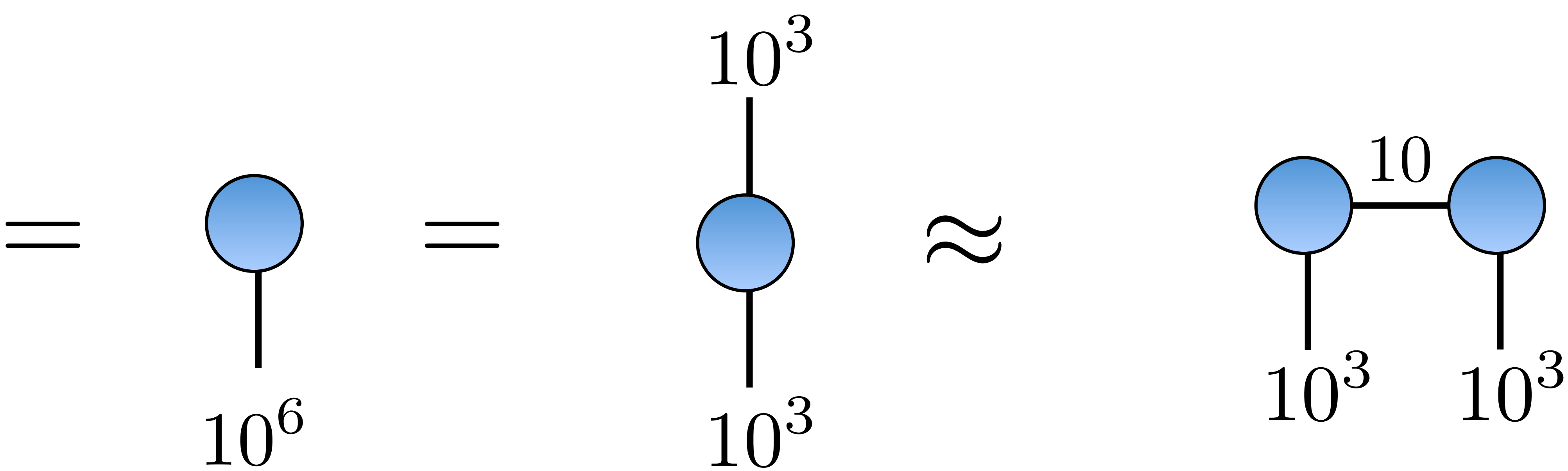


Exploring internal structures in the data



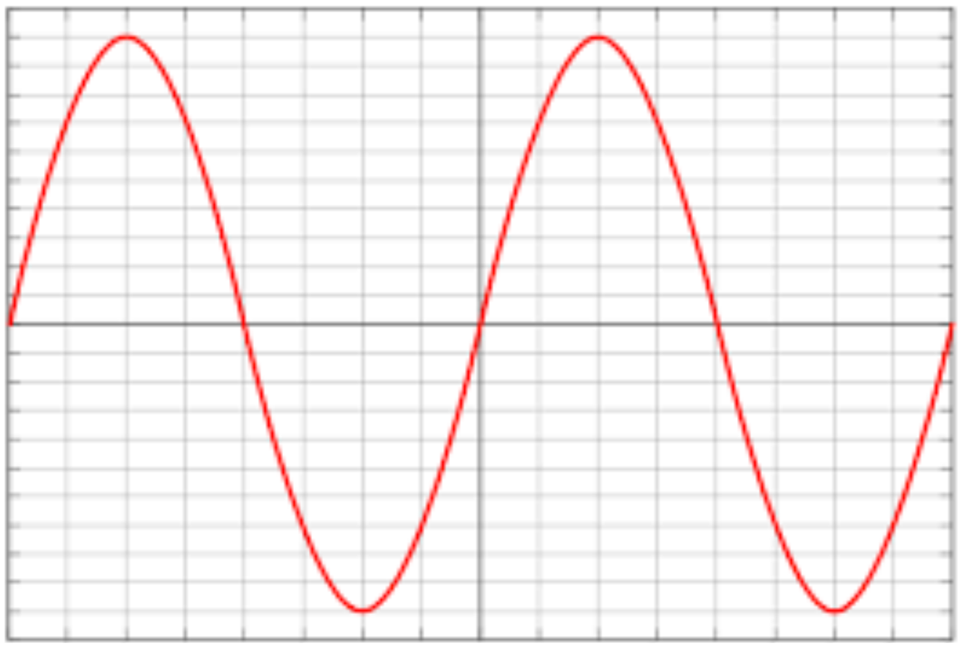
[0.000, 0.100, 0.199, 0.296, 0.389, 0.479, 0.565, 0.644, 0.717, 0.783,
0.841, 0.891, 0.932, 0.964, 0.985, 0.997, 1.000, 0.992, 0.974, 0.946,
0.909, 0.863, 0.808, 0.746, 0.675, 0.598, 0.516, 0.427, 0.335, 0.239,
0.141, 0.042, -0.058, -0.158, -0.256, -0.351, -0.443, -0.530, -0.612,
......
......
-0.688, -0.757, -0.818, -0.872, -0.916, -0.952, -0.978, -0.994, -1.000,
-0.996, -0.982, -0.959, -0.926, -0.883, -0.832, -0.773, -0.706, -0.631,
-0.551, -0.465, -0.374, -0.279, -0.182, -0.083, 0.017, 0.117, 0.215,
0.312, 0.405, 0.494, 0.578, 0.657, 0.729, 0.794, 0.850, 0.899, 0.938]

10^6 data points in a vector



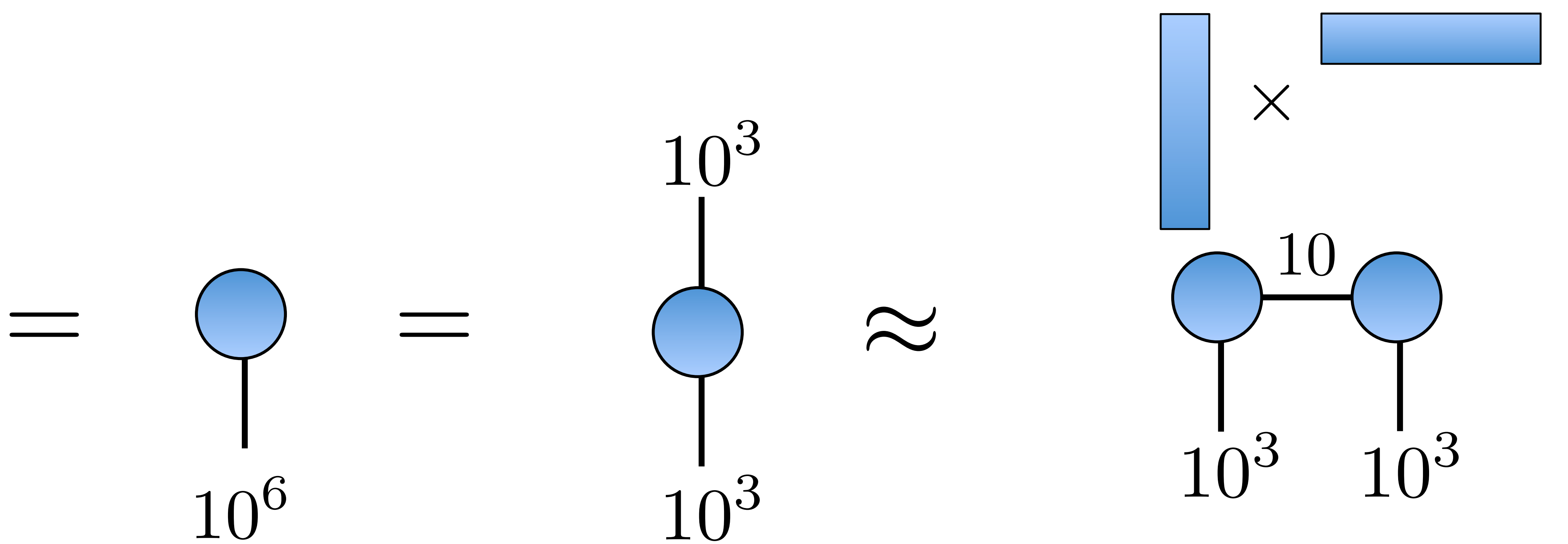
2×10^4 scalars

Exploring internal structures in the data



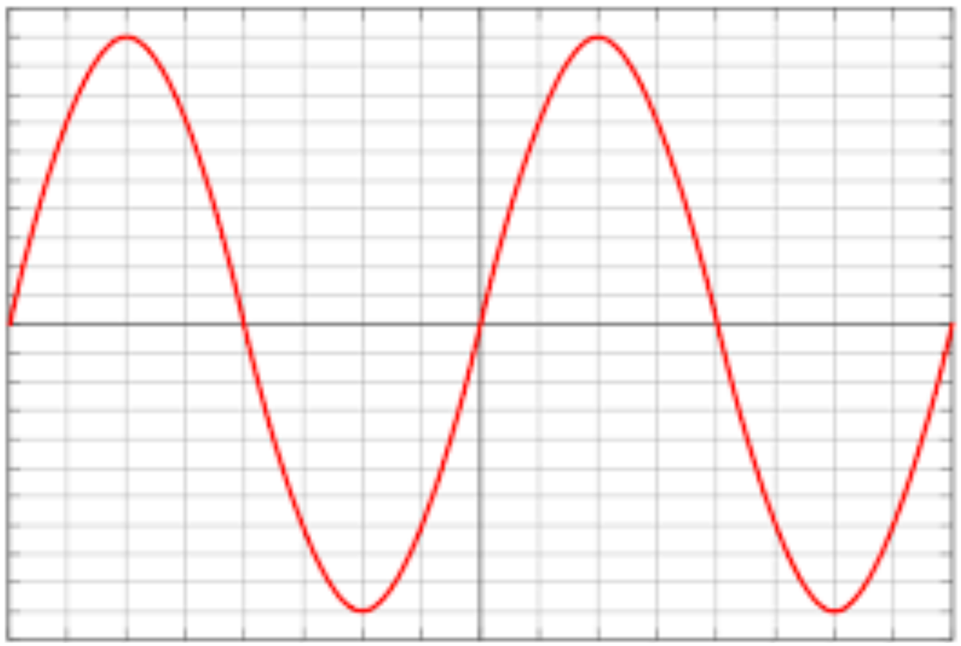
[0.000, 0.100, 0.199, 0.296, 0.389, 0.479, 0.565, 0.644, 0.717, 0.783, 0.841, 0.891, 0.932, 0.964, 0.985, 0.997, 1.000, 0.992, 0.974, 0.946, 0.909, 0.863, 0.808, 0.746, 0.675, 0.598, 0.516, 0.427, 0.335, 0.239, 0.141, 0.042, -0.058, -0.158, -0.256, -0.351, -0.443, -0.530, -0.612,
......
......
-0.688, -0.757, -0.818, -0.872, -0.916, -0.952, -0.978, -0.994, -1.000, -0.996, -0.982, -0.959, -0.926, -0.883, -0.832, -0.773, -0.706, -0.631, -0.551, -0.465, -0.374, -0.279, -0.182, -0.083, 0.017, 0.117, 0.215, 0.312, 0.405, 0.494, 0.578, 0.657, 0.729, 0.794, 0.850, 0.899, 0.938]

10^6 data points in a vector



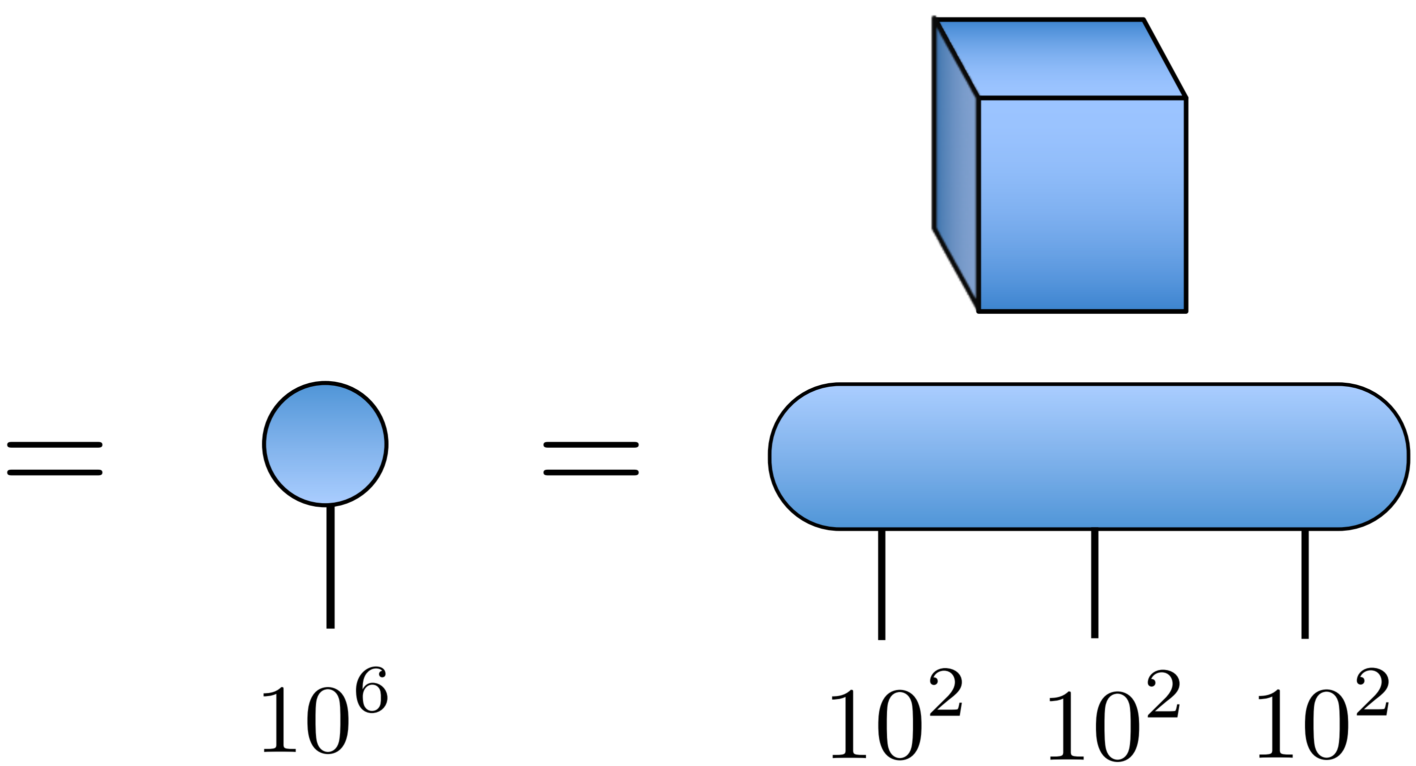
2×10^4 scalars

Exploring internal structures in the data

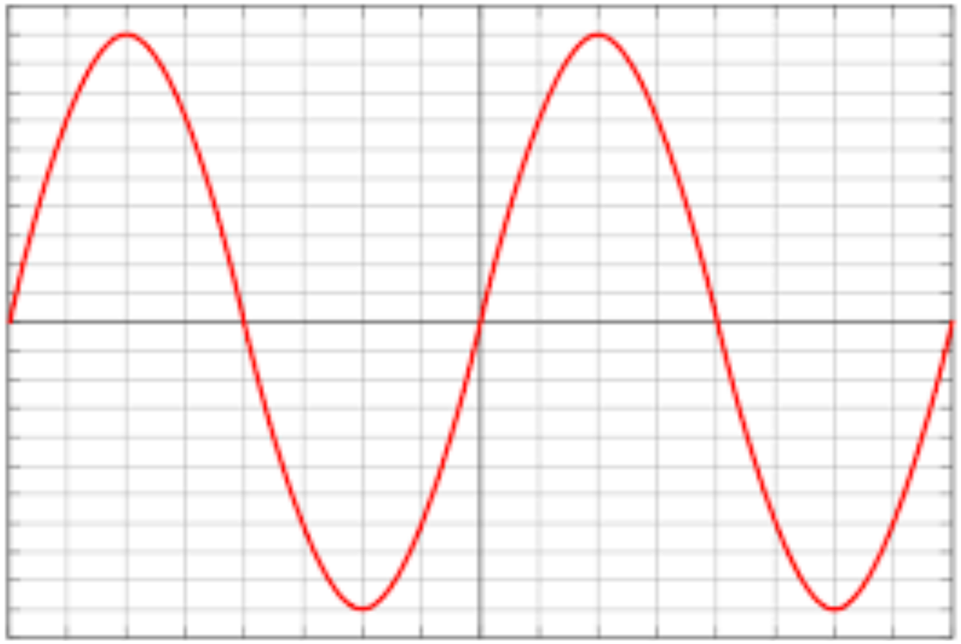


[0.000, 0.100, 0.199, 0.296, 0.389, 0.479, 0.565, 0.644, 0.717, 0.783,
0.841, 0.891, 0.932, 0.964, 0.985, 0.997, 1.000, 0.992, 0.974, 0.946,
0.909, 0.863, 0.808, 0.746, 0.675, 0.598, 0.516, 0.427, 0.335, 0.239,
0.141, 0.042, -0.058, -0.158, -0.256, -0.351, -0.443, -0.530, -0.612,
......
......
-0.688, -0.757, -0.818, -0.872, -0.916, -0.952, -0.978, -0.994, -1.000,
-0.996, -0.982, -0.959, -0.926, -0.883, -0.832, -0.773, -0.706, -0.631,
-0.551, -0.465, -0.374, -0.279, -0.182, -0.083, 0.017, 0.117, 0.215,
0.312, 0.405, 0.494, 0.578, 0.657, 0.729, 0.794, 0.850, 0.899, 0.938]

10^6 data points in a vector

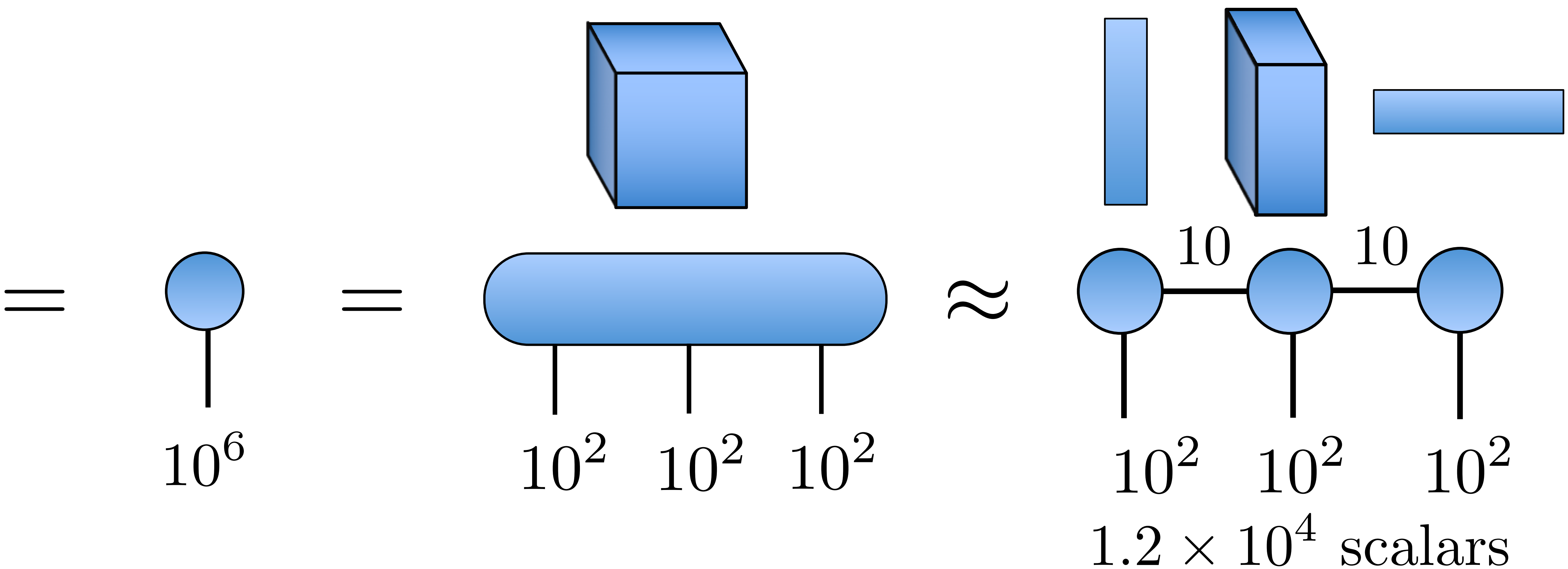


Exploring internal structures in the data

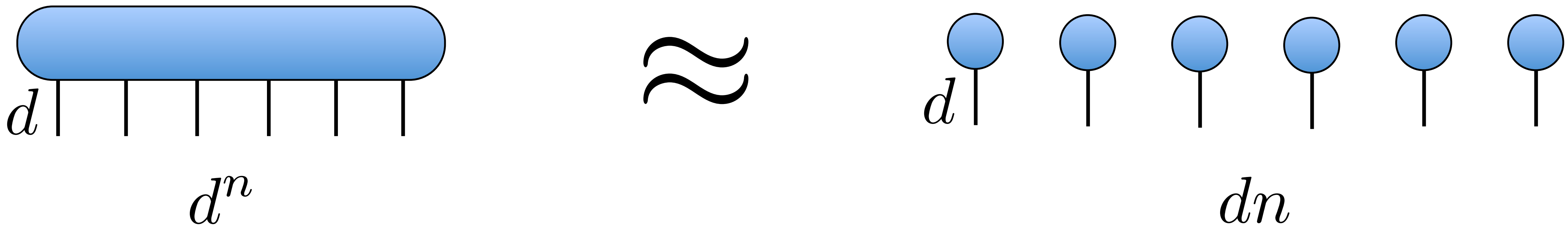


[0.000, 0.100, 0.199, 0.296, 0.389, 0.479, 0.565, 0.644, 0.717, 0.783,
0.841, 0.891, 0.932, 0.964, 0.985, 0.997, 1.000, 0.992, 0.974, 0.946,
0.909, 0.863, 0.808, 0.746, 0.675, 0.598, 0.516, 0.427, 0.335, 0.239,
0.141, 0.042, -0.058, -0.158, -0.256, -0.351, -0.443, -0.530, -0.612,
......
......
-0.688, -0.757, -0.818, -0.872, -0.916, -0.952, -0.978, -0.994, -1.000,
-0.996, -0.982, -0.959, -0.926, -0.883, -0.832, -0.773, -0.706, -0.631,
-0.551, -0.465, -0.374, -0.279, -0.182, -0.083, 0.017, 0.117, 0.215,
0.312, 0.405, 0.494, 0.578, 0.657, 0.729, 0.794, 0.850, 0.899, 0.938]

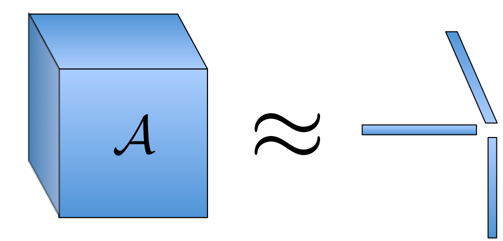
10^6 data points in a vector



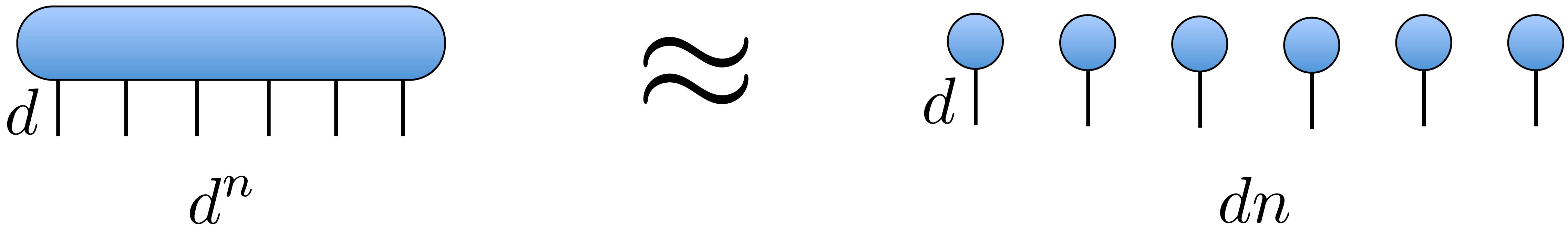
Representation of a large tensor: rank-one



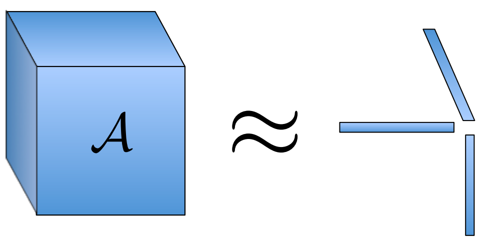
$$\mathcal{A} = \mathbf{a}^{(1)} \circ \mathbf{a}^{(2)} \circ \cdots \mathbf{a}^{(n)}$$



Representation of a large tensor: rank-one

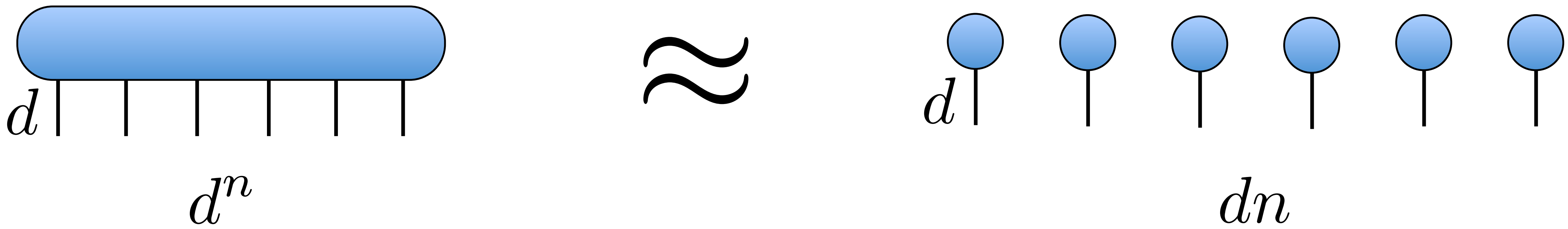


$$\mathcal{A} = \mathbf{a}^{(1)} \circ \mathbf{a}^{(2)} \circ \cdots \mathbf{a}^{(n)}$$

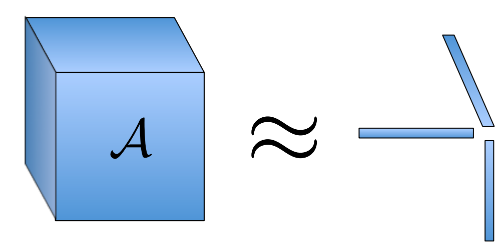


independent modes

Representation of a large tensor: rank-one



$$\mathcal{A} = \mathbf{a}^{(1)} \circ \mathbf{a}^{(2)} \circ \cdots \mathbf{a}^{(n)}$$

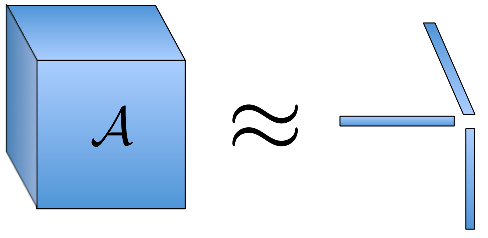


independent modes

factorized pure state

Representation of a large tensor: rank-one

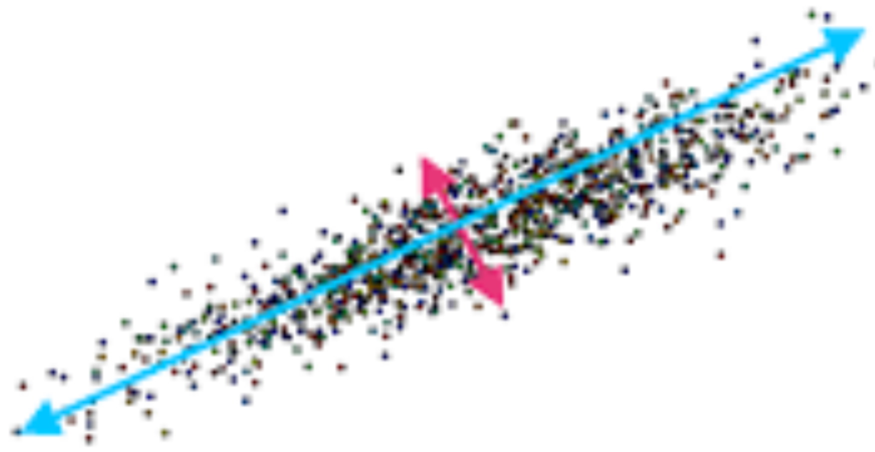
$$\mathcal{A} = \mathbf{a}^{(1)} \circ \mathbf{a}^{(2)} \circ \cdots \mathbf{a}^{(n)}$$



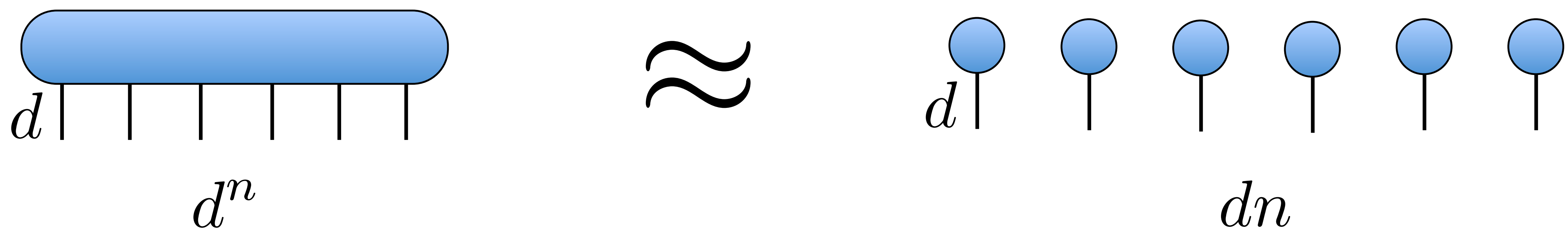
independent modes

factorized pure state

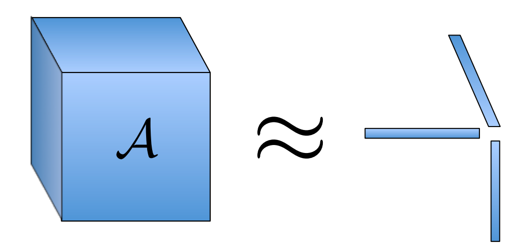
principled component



Representation of a large tensor: rank-one



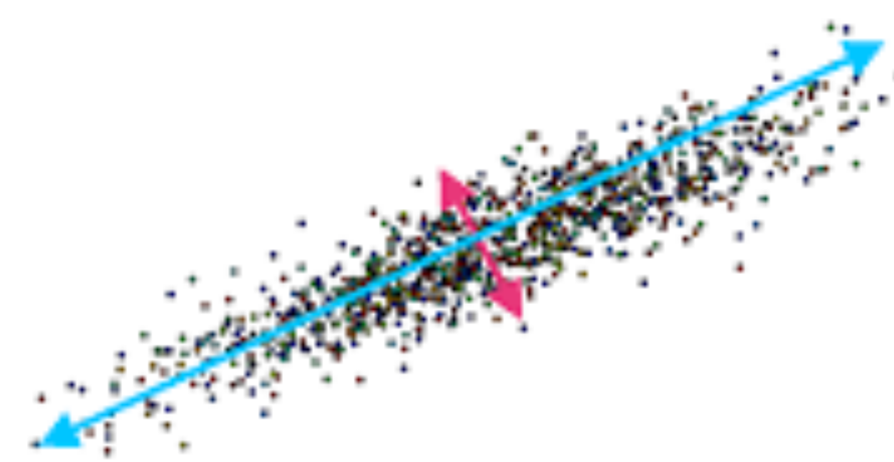
$$\mathcal{A} = \mathbf{a}^{(1)} \circ \mathbf{a}^{(2)} \circ \cdots \mathbf{a}^{(n)}$$



independent modes

factorized pure state

principled component

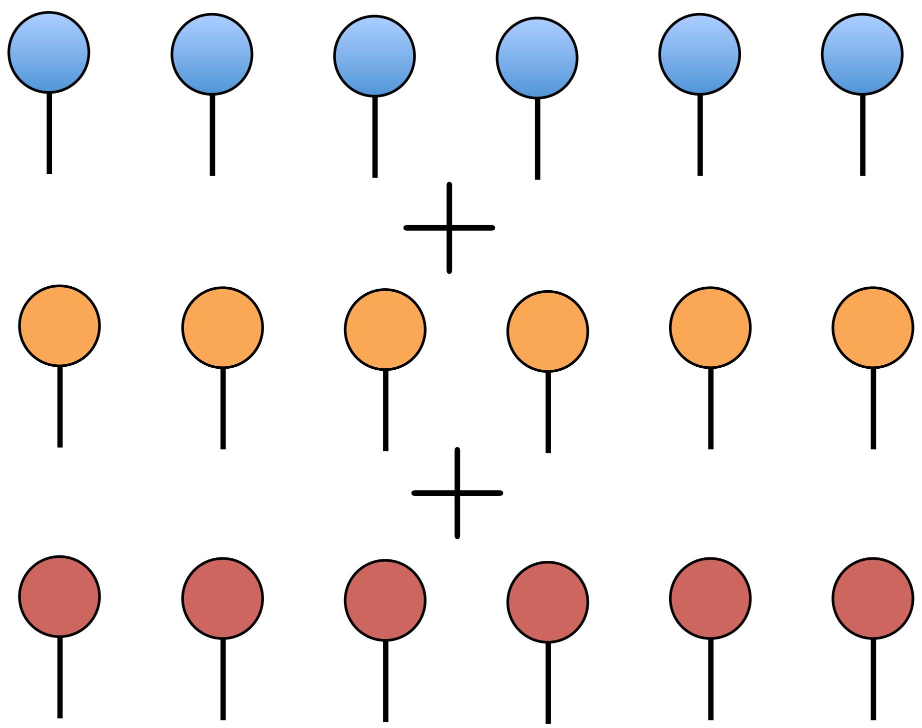


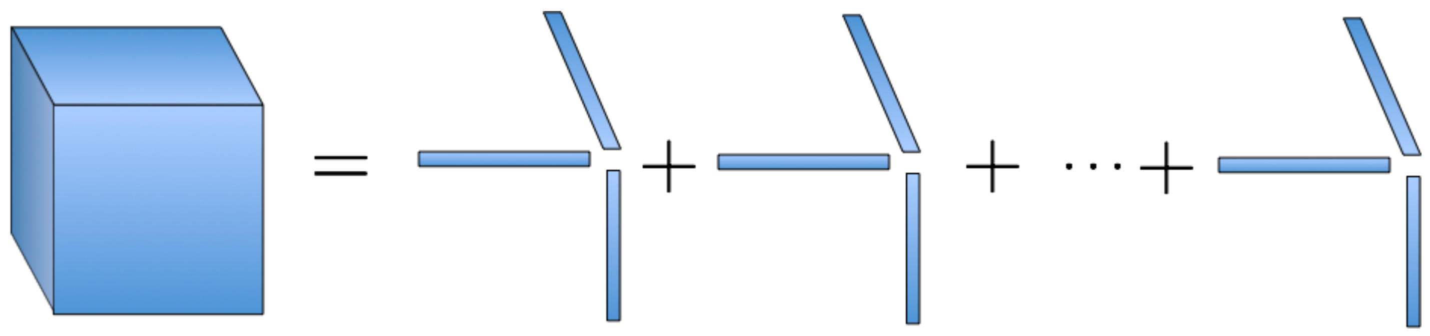
reminiscent of the ***mean-field*** approximation

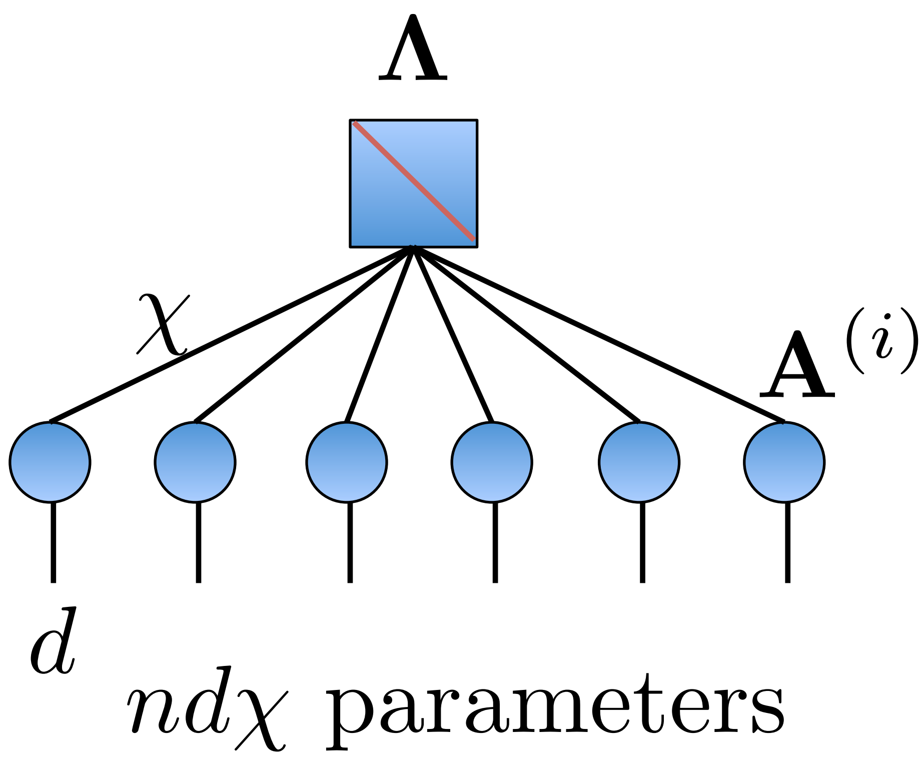
$$p(\{x_1, x_2, ..., x_n\}) = p(x_1)p(x_2)...p(x_n)$$

Canonical Polyadic (CP) decomposition

$$\mathcal{A} = \sum_{r=1}^{\chi} \lambda_r \mathbf{a}_r^{(1)} \circ \mathbf{a}_r^{(2)} \circ \cdots \circ \mathbf{a}_r^{(n)}$$
$$= \mathbf{\Lambda} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \cdots \times_n \mathbf{A}^{(\mathbf{n})}$$
$$= [\mathbf{\Lambda}; \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \cdots, \mathbf{A}^{(\mathbf{n})}]$$

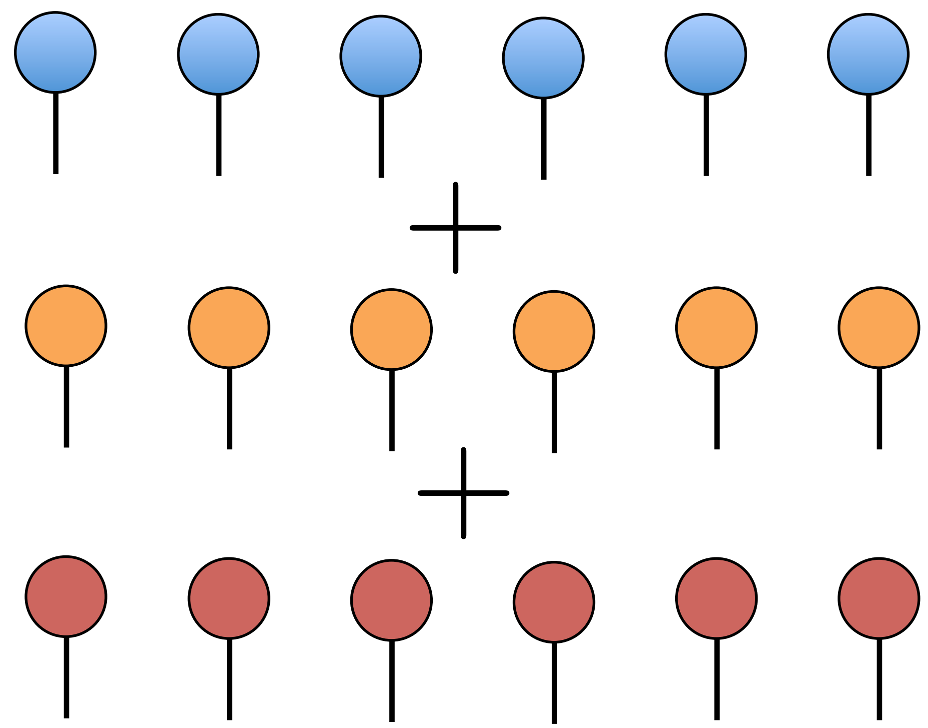






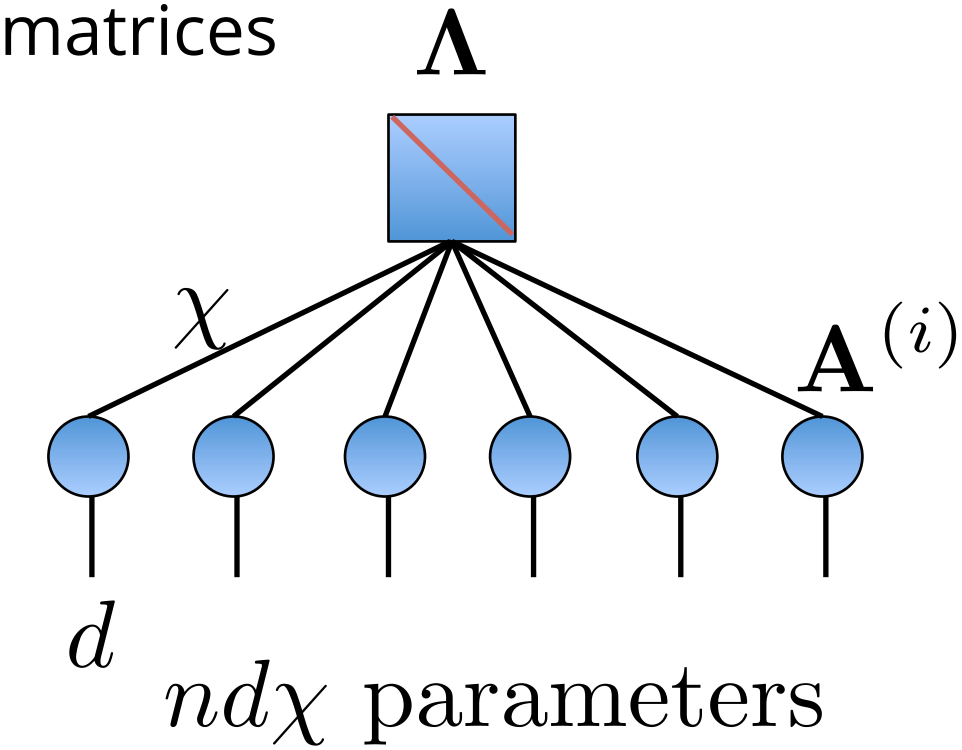
Canonical Polyadic (CP) decomposition

$$\mathcal{A} = \sum_{r=1}^{\chi} \lambda_r \mathbf{a}_r^{(1)} \circ \mathbf{a}_r^{(2)} \circ \cdots \circ \mathbf{a}_r^{(n)}$$
$$= \mathbf{\Lambda} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \cdots \times_n \mathbf{A}^{(\mathbf{n})}$$
$$= [\mathbf{\Lambda}; \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \cdots, \mathbf{A}^{(\mathbf{n})}]$$

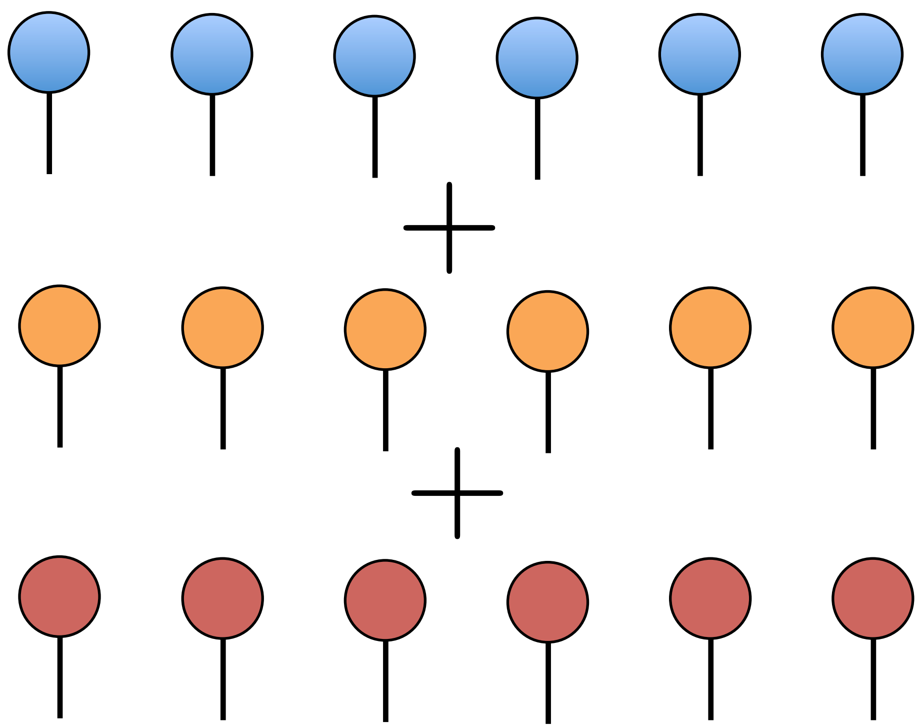


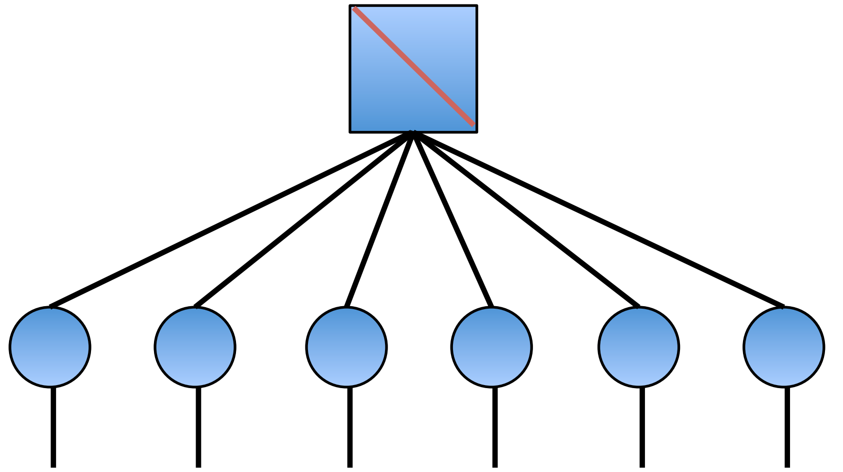
Convenient representations of CP using unfolded matrices

$$\mathbf{A}_{(i)} = \mathbf{A}^{(i)} \mathbf{\Lambda} \left(\mathbf{A}^{(1)} \odot \mathbf{A}^{(2)} \odot \cdots \odot \mathbf{A}^{(n)} \right)$$



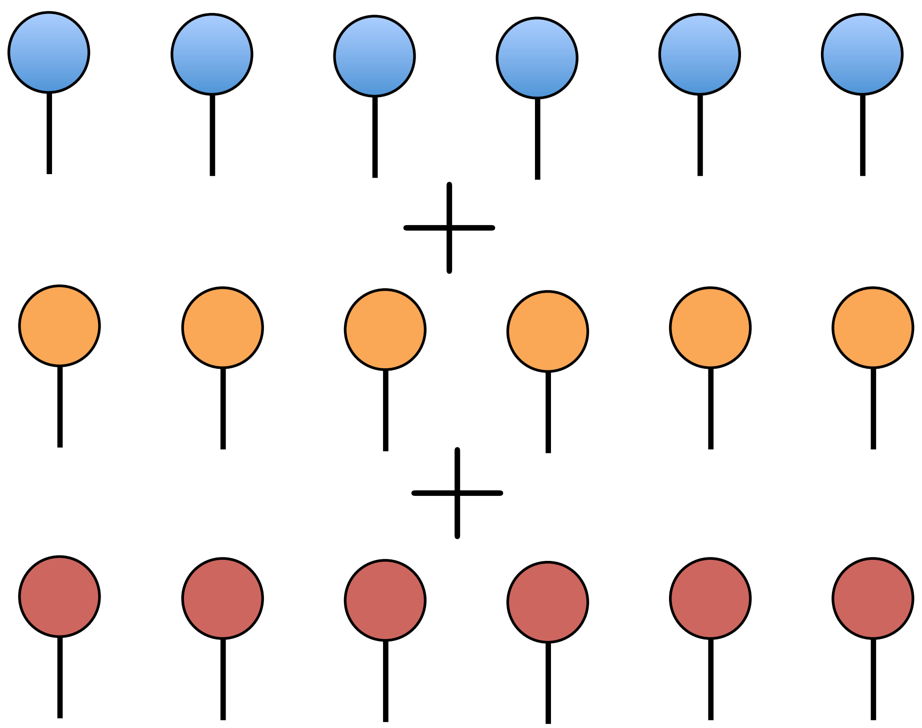
Canonical Polyadic (CP) decomposition

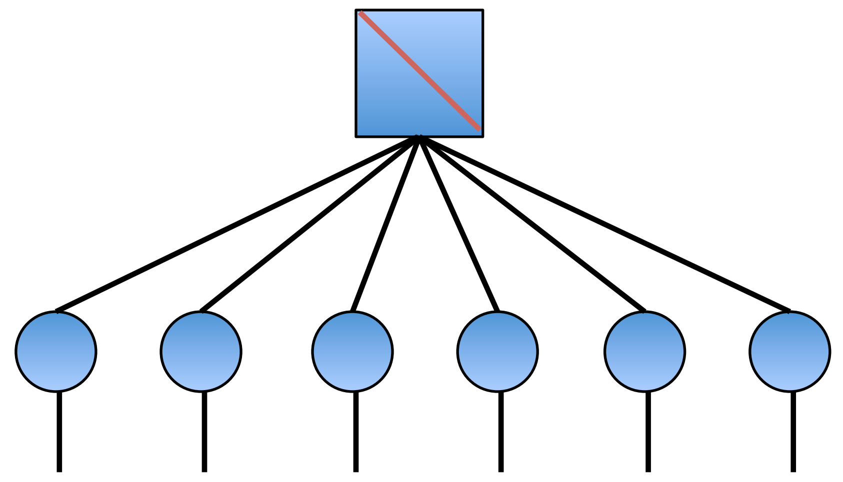




Canonical Polyadic (CP) decomposition

Also known as ***Polyadic***, ***PARAFAC***, ***CANDECOMP***, and ***Topographic***.

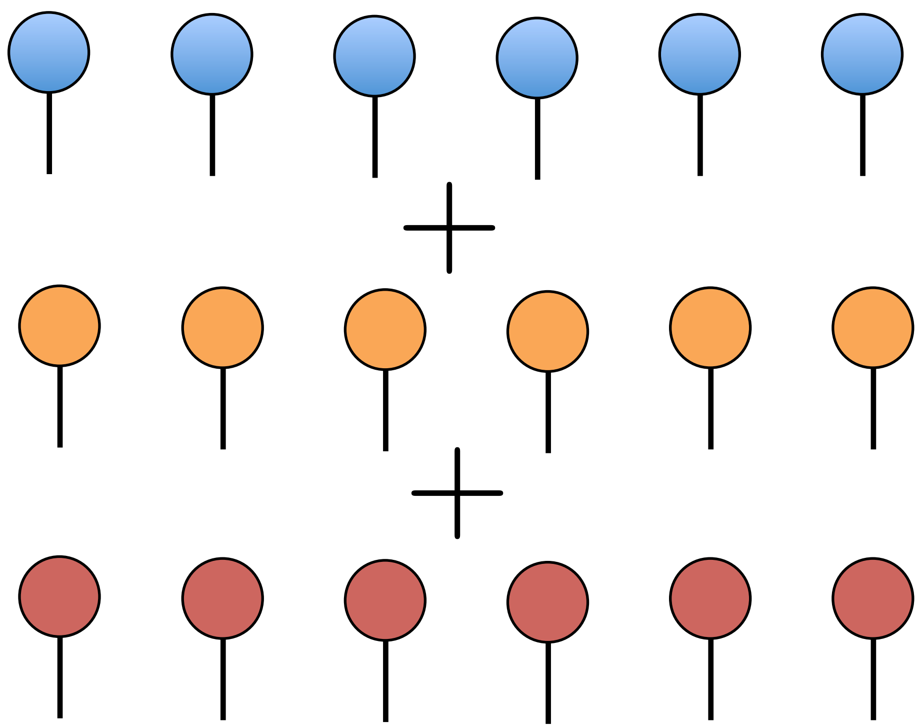


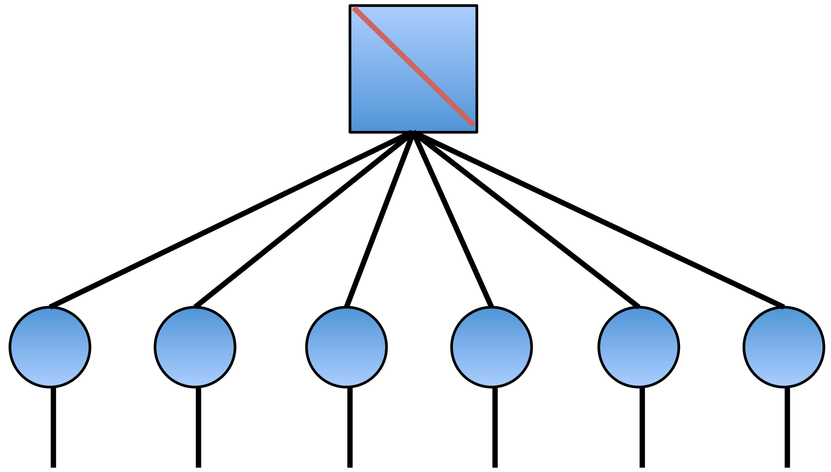


Canonical Polyadic (CP) decomposition

Also known as ***Polyadic***, ***PARAFAC***, ***CANDECOMP***, and ***Topographic***.

First proposed in 1927 by ***Hitchcock,*** then became popular in 1970's in psychometric and in 1980's in chemometrics.



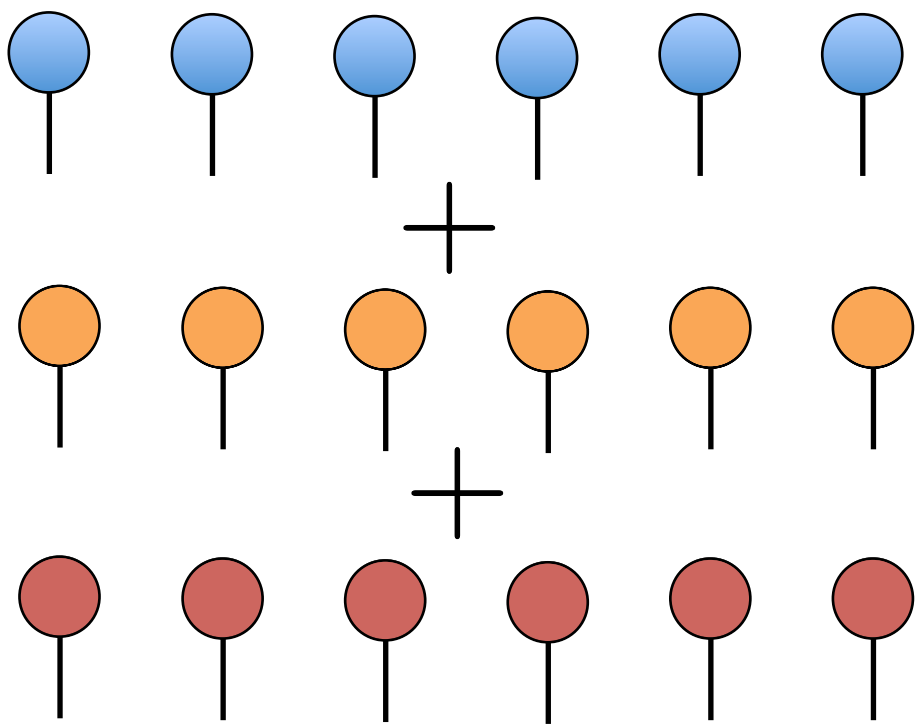


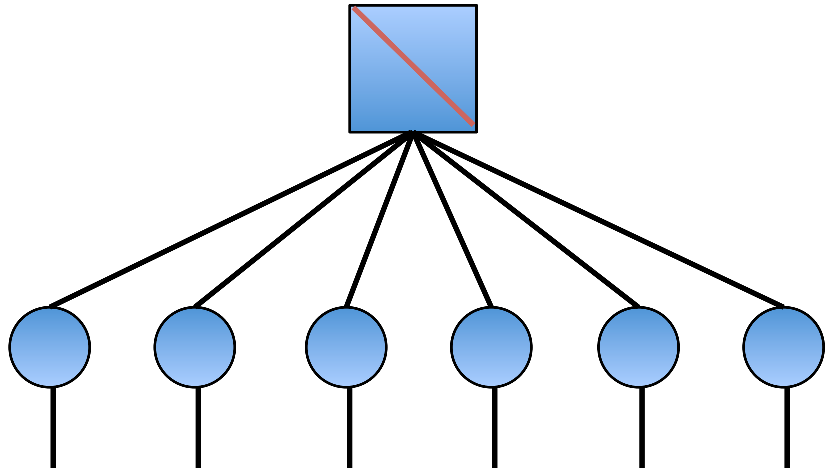
Canonical Polyadic (CP) decomposition

Also known as ***Polyadic***, ***PARAFAC***, ***CANDECOMP***, and ***Topographic***.

First proposed in 1927 by ***Hitchcock,*** then became popular in 1970's in psychometric and in 1980's in chemometrics.

Found applications in signal processing, neural sciences, signal processing in 1990's, and image compression and classification from 2005.





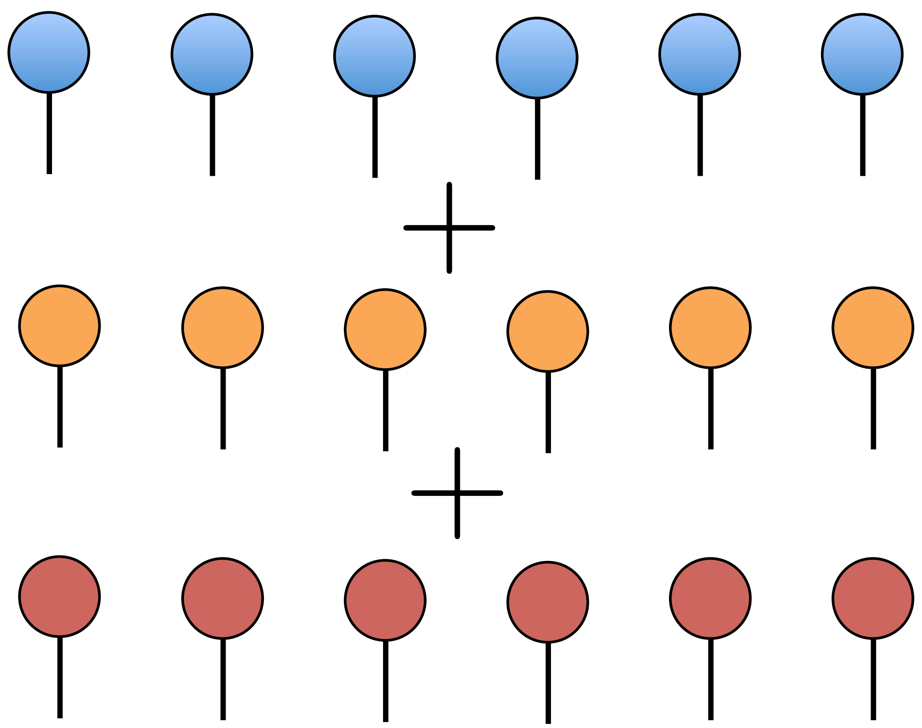
Canonical Polyadic (CP) decomposition

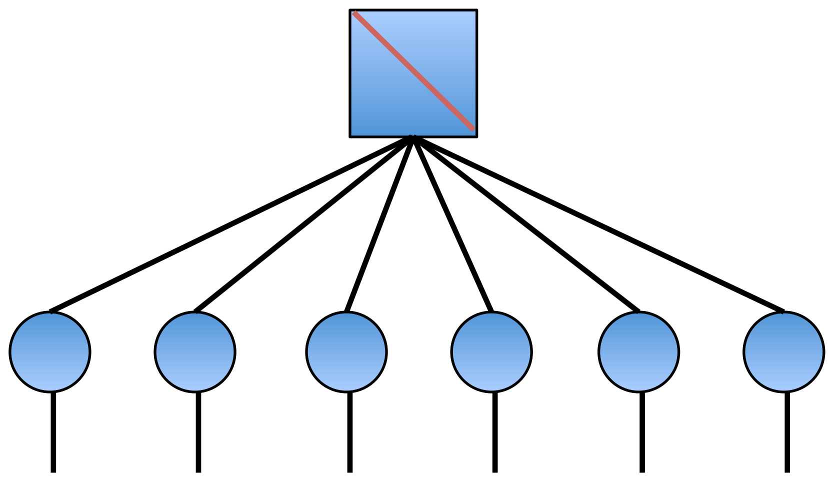
Also known as ***Polyadic***, ***PARAFAC***, ***CANDECOMP***, and ***Topographic***.

First proposed in 1927 by ***Hitchcock,*** then became popular in 1970's in psychometric and in 1980's in chemometrics.

Found applications in signal processing, neural sciences, signal processing in 1990's, and image compression and classification from 2005.

Any tensor can be expressed as ***finite*** sum of rank-1 tensors.





Canonical Polyadic (CP) decomposition

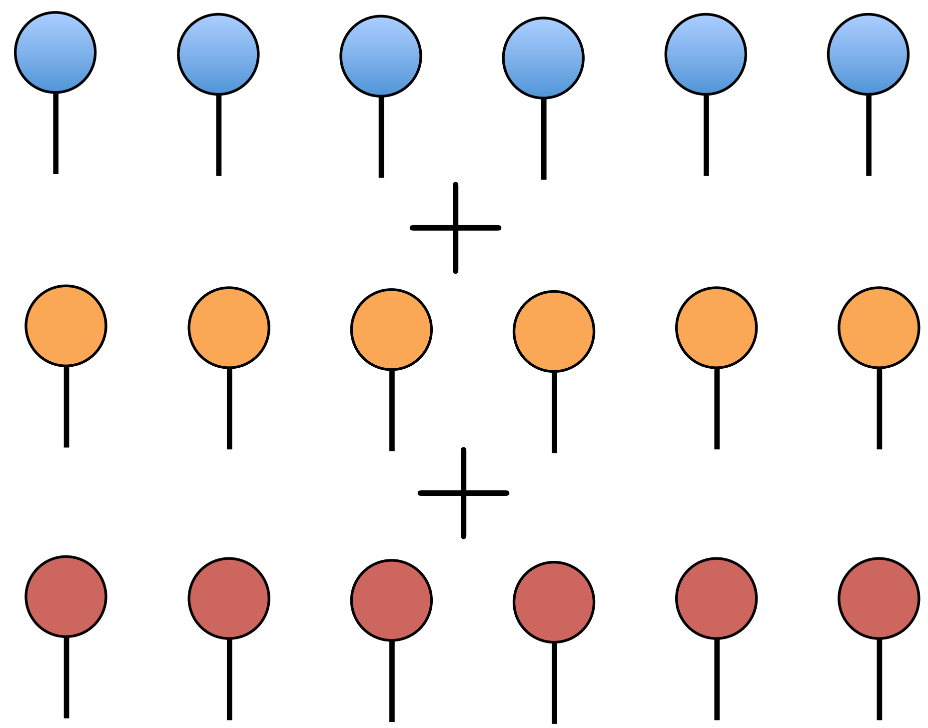
Also known as ***Polyadic***, ***PARAFAC***, ***CANDECOMP***, and ***Topographic***.

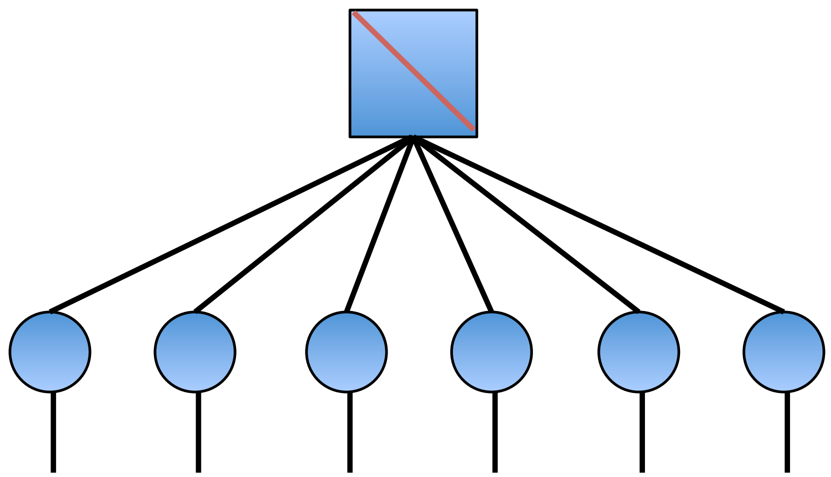
First proposed in 1927 by ***Hitchcock,*** then became popular in 1970's in psychometric and in 1980's in chemometrics.

Found applications in signal processing, neural sciences, signal processing in 1990's, and image compression and classification from 2005.

Any tensor can be expressed as ***finite*** sum of rank-1 tensors.

Tensor rank, or CP-rank, is defined as a ***minimum number of rank-1 tensors*** in the exact CP decomposition.





Canonical Polyadic (CP) decomposition

Also known as ***Polyadic***, ***PARAFAC***, ***CANDECOMP***, and ***Topographic***.

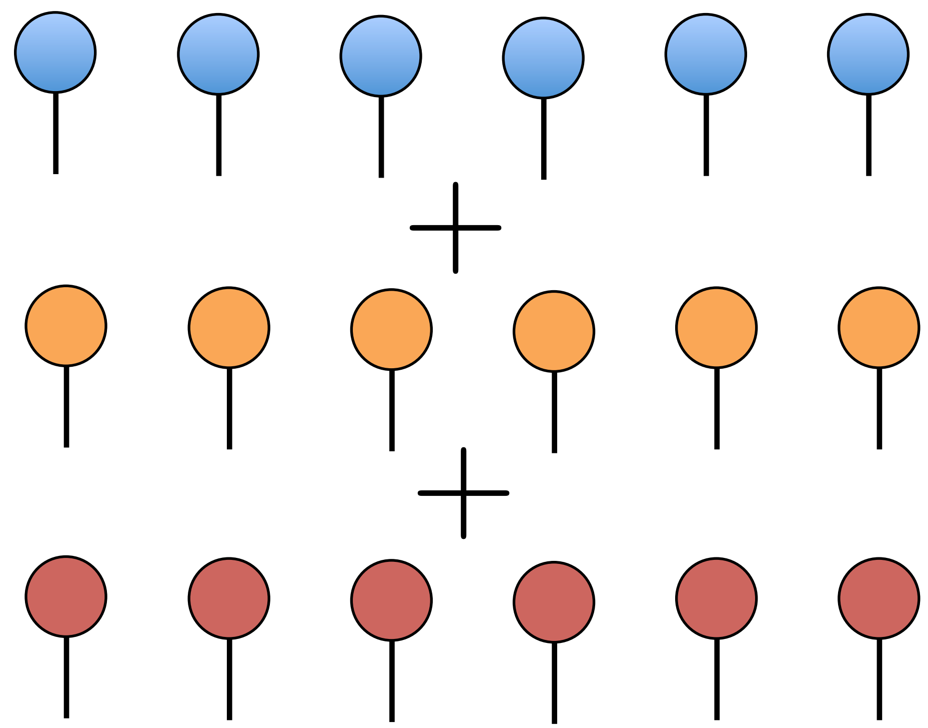
First proposed in 1927 by ***Hitchcock,*** then became popular in 1970's in psychometric and in 1980's in chemometrics.

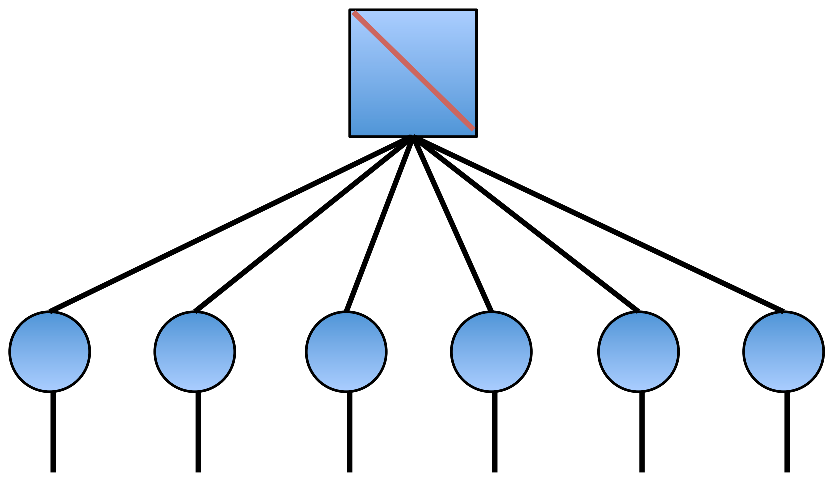
Found applications in signal processing, neural sciences, signal processing in 1990's, and image compression and classification from 2005.

Any tensor can be expressed as ***finite*** sum of rank-1 tensors.

Tensor rank, or CP-rank, is defined as a ***minimum number of rank-1 tensors*** in the exact CP decomposition.

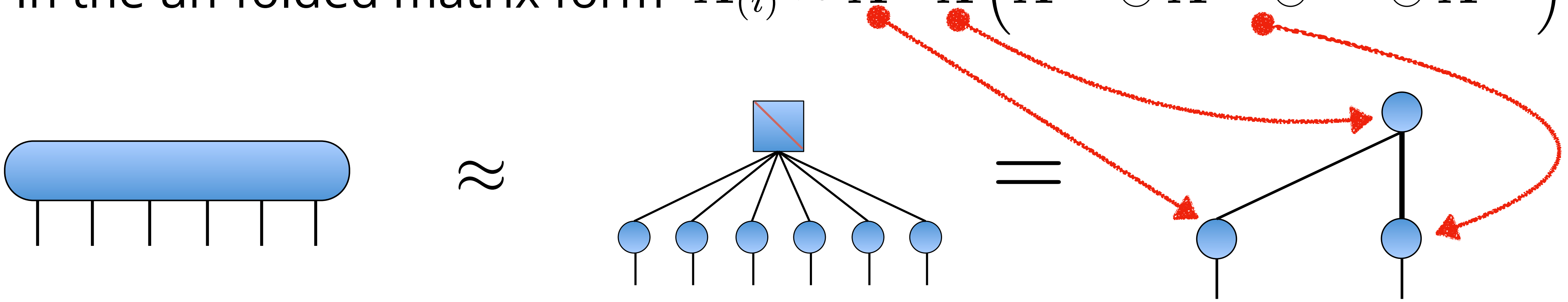
However, determining the CP-rank is ***NP-hard*** problem, finding the best CP decomposition is ***hard***, yielding many local minimums.





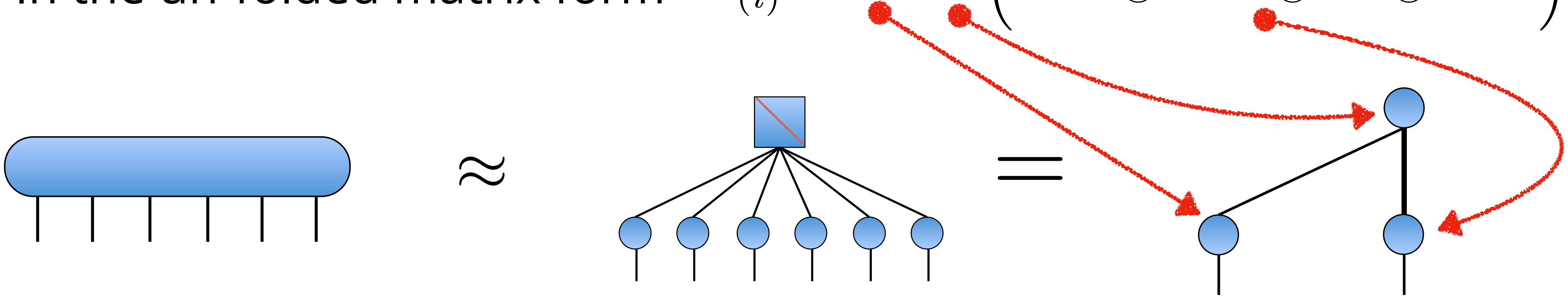
The workhorse algorithm: Alternating Least Square

In the un-folded matrix form $\mathbf{A}_{(i)} \approx \mathbf{A}^{(i)} \mathbf{\Lambda} \left(\mathbf{A}^{(1)} \odot \mathbf{A}^{(2)} \odot \cdots \odot \mathbf{A}^{(n)} \right)$

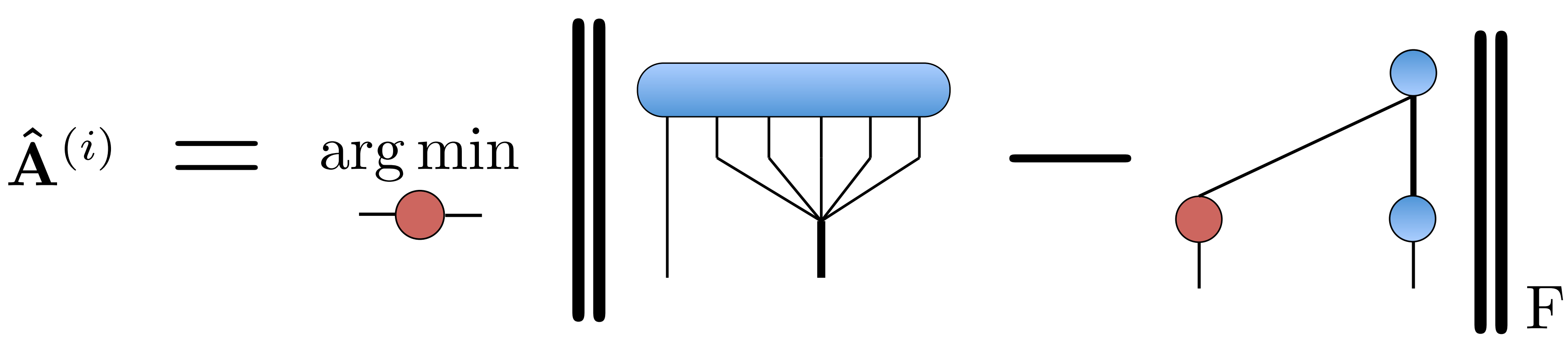


The workhorse algorithm: Alternating Least Square

In the un-folded matrix form $\mathbf{A}_{(i)} \approx \mathbf{A}^{(i)} \mathbf{\Lambda} \left(\mathbf{A}^{(1)} \odot \mathbf{A}^{(2)} \odot \cdots \odot \mathbf{A}^{(n)} \right)$

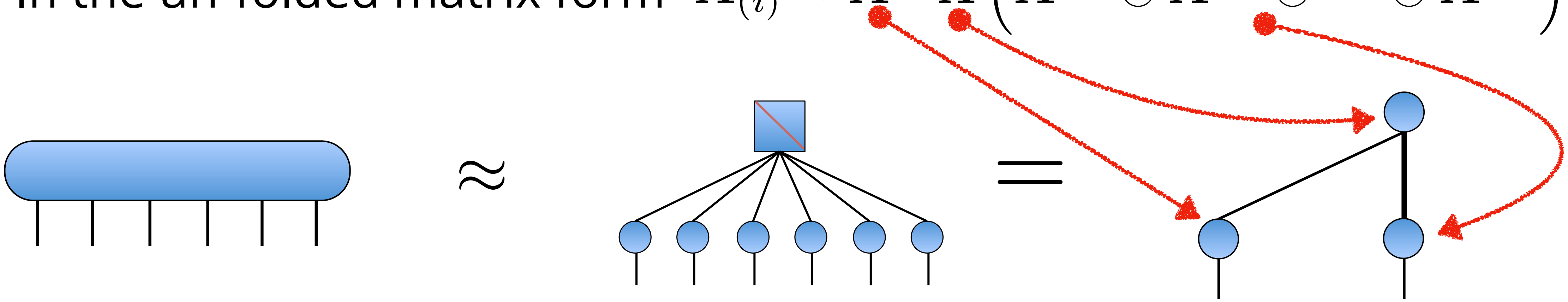


The ALS fixes all except one matrix for optimizing, the sub-problems is a least-square problem

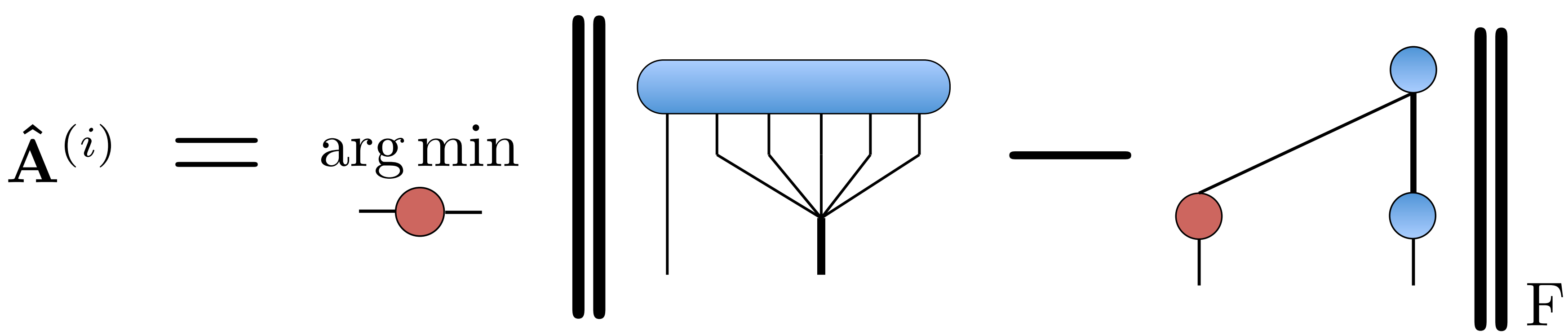


The workhorse algorithm: Alternating Least Square

In the un-folded matrix form $\mathbf{A}_{(i)} \approx \mathbf{A}^{(i)} \mathbf{\Lambda} \left(\mathbf{A}^{(1)} \odot \mathbf{A}^{(2)} \odot \cdots \odot \mathbf{A}^{(n)} \right)$



The ALS fixes all except one matrix for optimizing, the sub-problems is a least-square problem



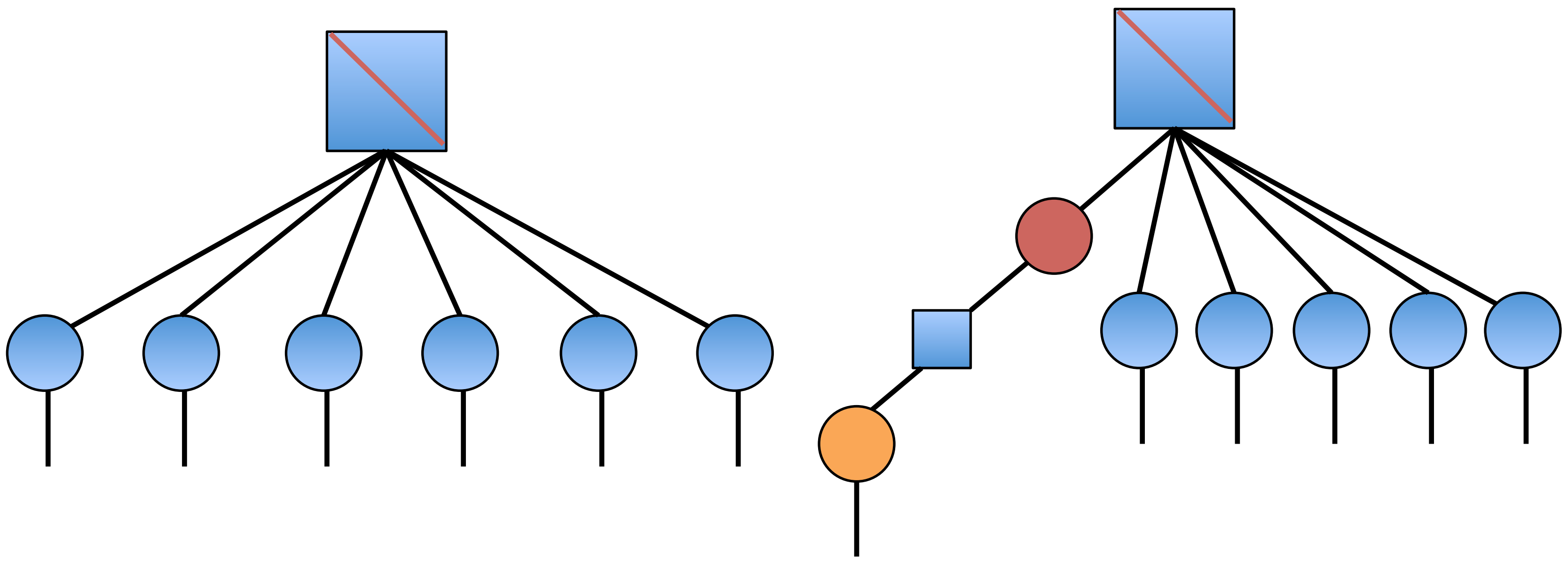
Solved simply using pseudo-inverse

$$\hat{\mathbf{A}}^{(\mathbf{i})} \leftarrow \mathbf{A}^{(i)} \left(\mathbf{\Lambda} \mathbf{A}^{(1)} \odot \mathbf{A}^{(2)} \odot \cdots \odot \mathbf{A}^{(i-1)} \odot \mathbf{A}^{(i+1)} \cdots \odot \mathbf{A}^{(n)} \right)^{+}$$

From CP to Tucker

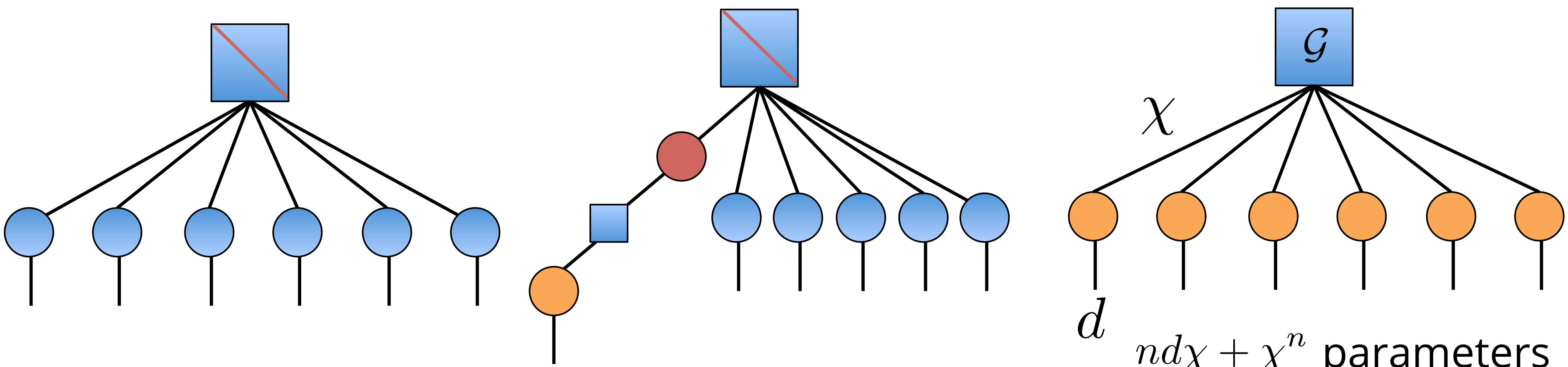
The CP formats is lack of canonical forms and orthogonalization

by imposing orthogonalization, the core tensor is no more diagonal



From CP to Tucker

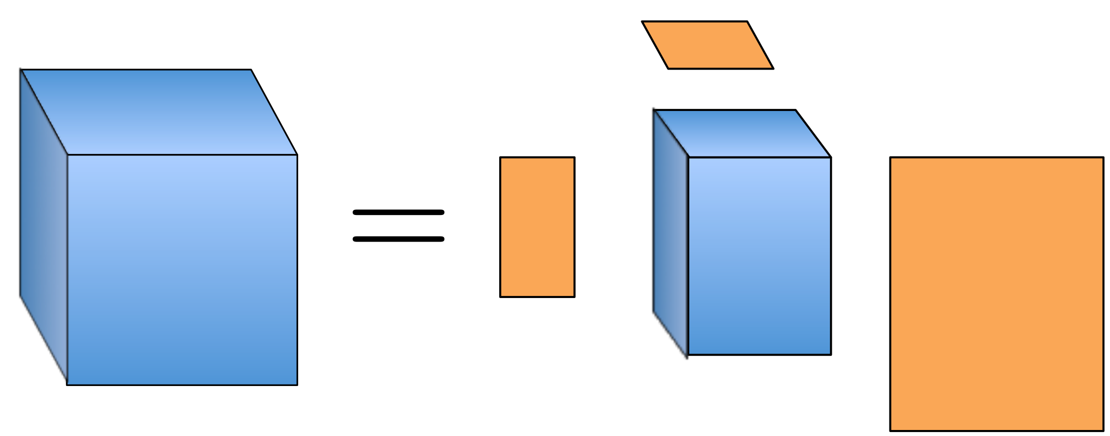
The CP formats is lack of canonical forms and orthogonalization
by imposing orthogonalization, the core tensor is no more diagonal



$nd\chi + \chi^n$ parameters

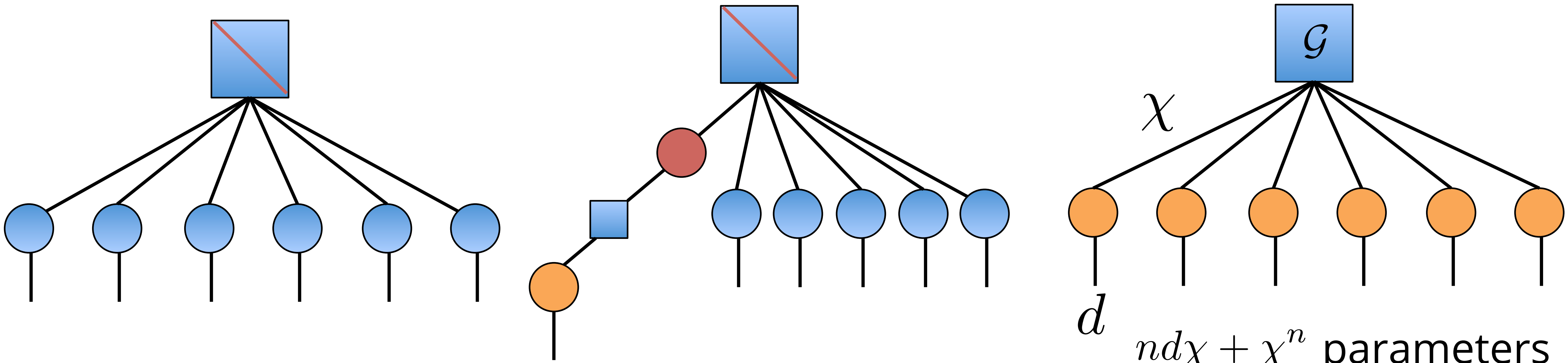
this leads to the ***Tucker Decomposition***

$$\mathcal{A} = [\mathcal{G}; \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \cdots, \mathbf{A}^{(\mathbf{n})}]$$
$$= \mathcal{G} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times_3 \cdots \times_n \mathbf{A}^{(n)}$$



From CP to Tucker

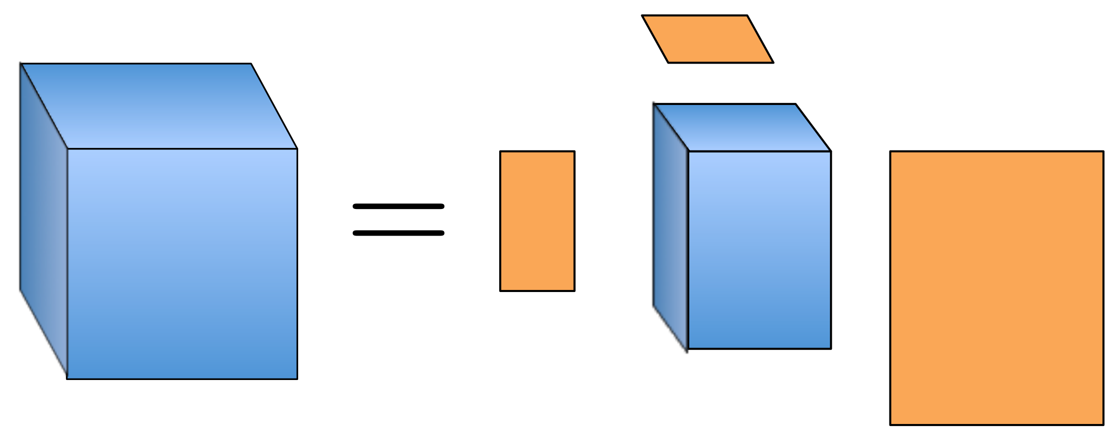
The CP formats is lack of canonical forms and orthogonalization
by imposing orthogonalization, the core tensor is no more diagonal



$nd\chi + \chi^n$ parameters

this leads to the ***Tucker Decomposition***

$$
\begin{aligned}
\mathcal{A} &= [\mathcal{G}; \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \cdots, \mathbf{A}^{(\mathbf{n})}] \\
&= \mathcal{G} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times_3 \cdots \times_n \mathbf{A}^{(n)}
\end{aligned}
$$

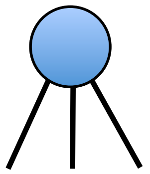


First introduced by Tucker in ***1963,*** known as ***N-mode PCA/SVD/Factor analysis.***
Usually treated as a multilinear extension of PCA

High order singular value decomposition

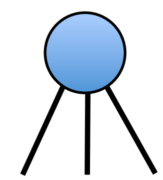
High order singular value decomposition

given a raw tensor $\mathcal{A}$

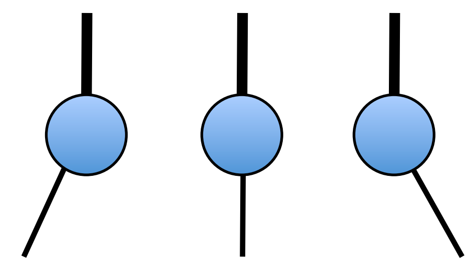


High order singular value decomposition

given a raw tensor $\mathcal{A}$

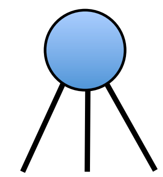


unfold to matrix at each mode

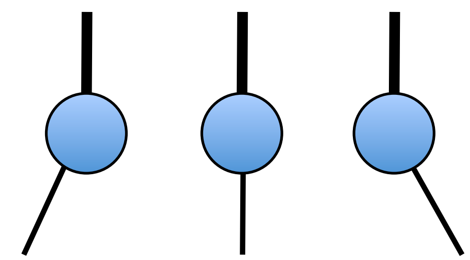


High order singular value decomposition

given a raw tensor $\mathcal{A}$

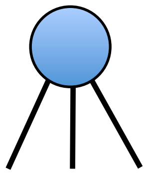


unfold to matrix at each mode

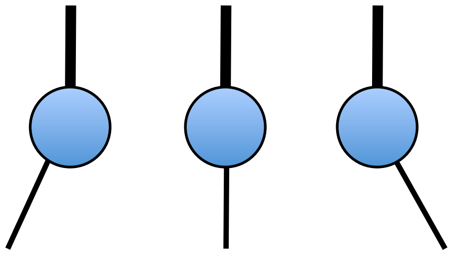


High order singular value decomposition

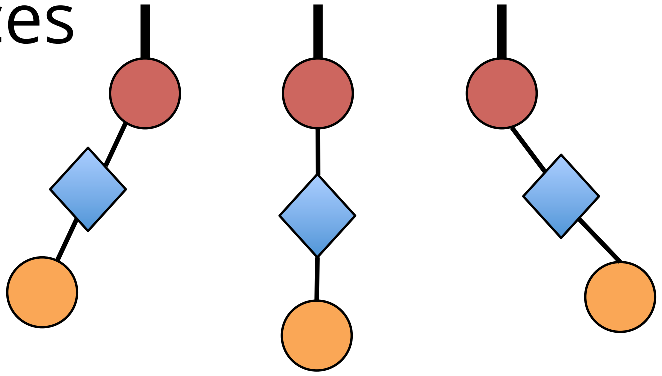
given a raw tensor $\mathcal{A}$



unfold to matrix at each mode

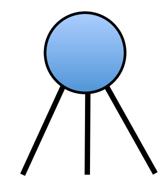


do SVD on obtained matrices

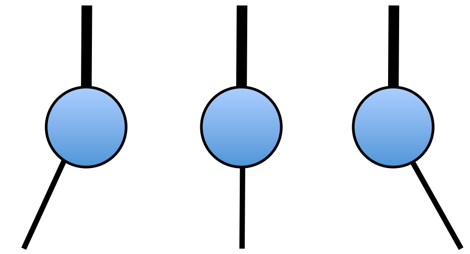


High order singular value decomposition

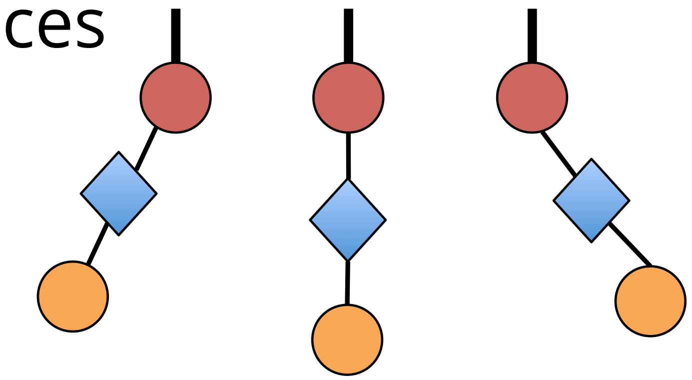
given a raw tensor $\mathcal{A}$



unfold to matrix at each mode



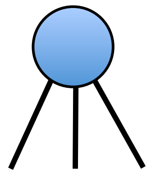
do SVD on obtained matrices



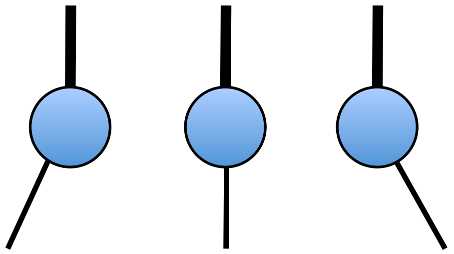
contract A to the inverse of the obtained column orthogonal matrices, to obtain the core $\mathcal{G}$.

High order singular value decomposition

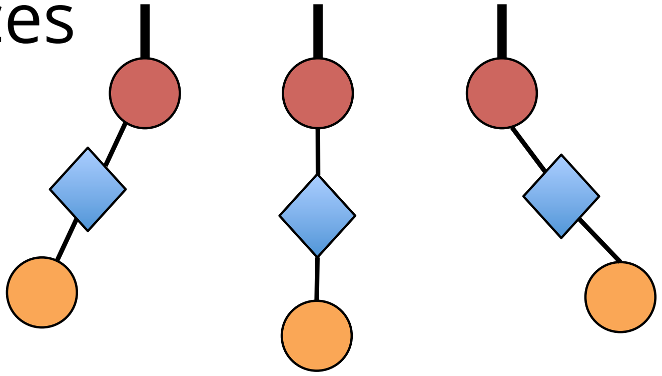
given a raw tensor $\mathcal{A}$



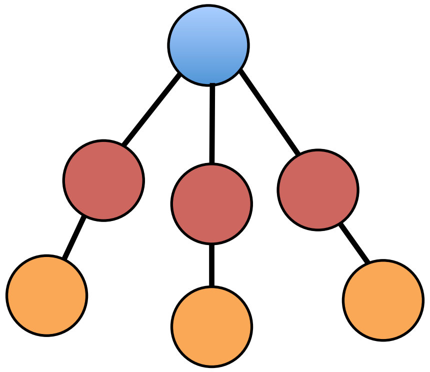
unfold to matrix at each mode



do SVD on obtained matrices

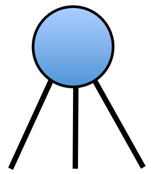


contract A to the inverse of the obtained column orthogonal matrices,
to obtain the core $\mathcal{G}$.

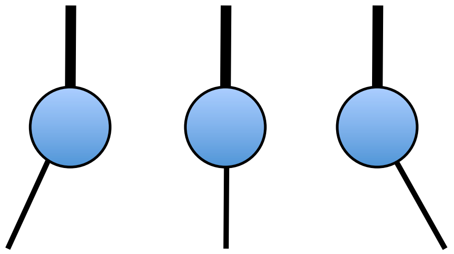


High order singular value decomposition

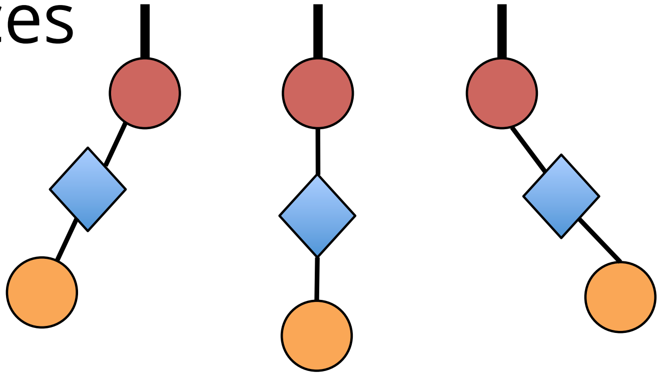
given a raw tensor $\mathcal{A}$



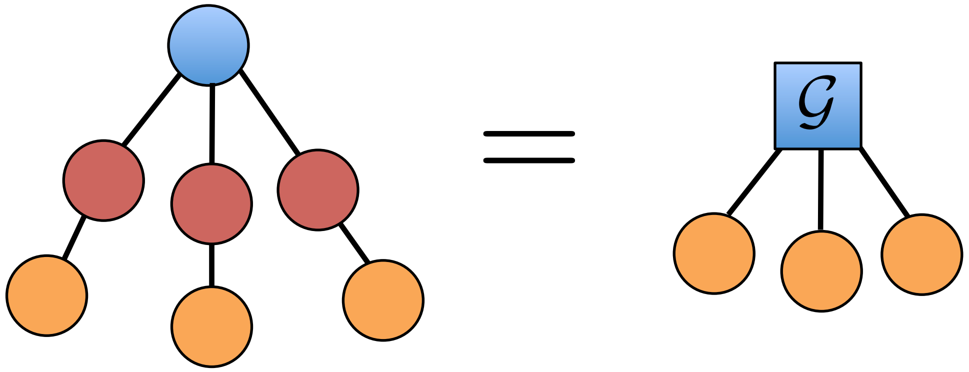
unfold to matrix at each mode



do SVD on obtained matrices



contract A to the inverse of the obtained column orthogonal matrices, to obtain the core $\mathcal{G}$.



Improving HOSVD

HOSVD only gives a quasi-best approximation

$$\left\| \mathcal{A} - [\mathcal{G}; \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \cdots, \mathbf{A}^{(n)}] \right\|_F \leq \sqrt{n} \, \|\mathcal{A} - \mathcal{A}_{\text{best}}\|_F$$

High Order Orthogonal Iteration (HOOI) interactively refines factor matrices using SVDs.

Elden and Savas's method contains factor matrices to a Grassmann manifold which defines an equivalent class of orthogonal matrices, then optimize them using Newton's method.

Lathauwer, Moor, Vandewalle. SIAM J. Matrix Anal. Appl. 21, 1253 (2000)
Elden, Savas, SIAM J. Matrix Anal. Appl. 31, 248 (2009)

Improving HOSVD

HOSVD only gives a quasi-best approximation

$$\left\| \mathcal{A} - [\mathcal{G}; \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \cdots, \mathbf{A}^{(n)}] \right\|_F \leq \sqrt{n} \, \| \mathcal{A} - \mathcal{A}_{\text{best}} \|_F$$

High Order Orthogonal Iteration (HOOI) interactively refines factor matrices using SVDs.

Elden and Savas's method contains factor matrices to a Grassmann manifold which defines an equivalent class of orthogonal matrices, then optimize them using Newton's method.

limitation:

$nd\chi + \chi^n$ parameters, ***not parameter efficient for n large***

Lathauwer, Moor, Vandewalle. SIAM J. Matrix Anal. Appl. 21, 1253 (2000)
Elden, Savas, SIAM J. Matrix Anal. Appl. 31, 248 (2009)

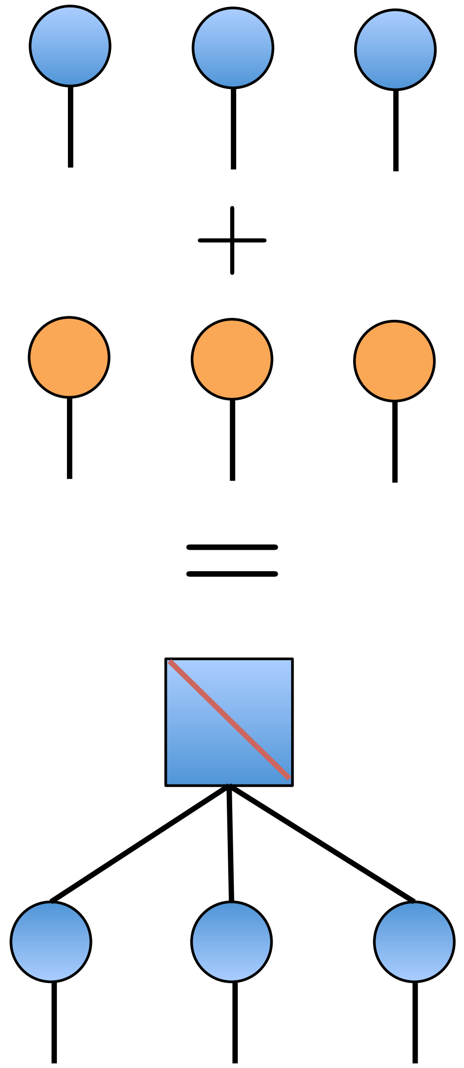
From CP to MPS

Summing two rank-1 tensors

$$\mathcal{A} = \mathbf{a}^{(1)} \circ \mathbf{a}^{(2)} \circ \mathbf{a}^{(3)}$$

$$\mathcal{B} = \mathbf{b}^{(1)} \circ \mathbf{b}^{(2)} \circ \mathbf{b}^{(3)}$$

$$\mathcal{C} = \mathcal{A} + \mathcal{B}$$



From CP to MPS

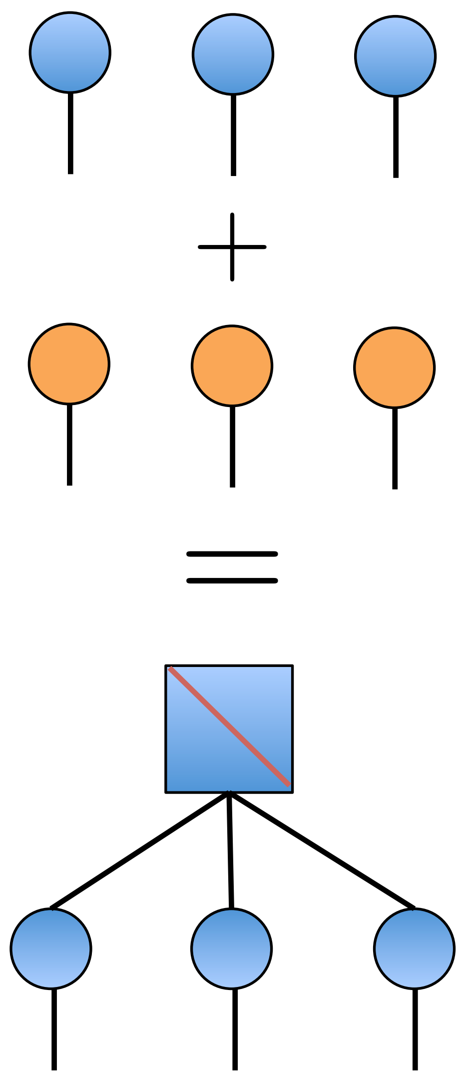
Summing two rank-1 tensors

$$\mathcal{A} = \mathbf{a}^{(1)} \circ \mathbf{a}^{(2)} \circ \mathbf{a}^{(3)}$$

$$\mathcal{B} = \mathbf{b}^{(1)} \circ \mathbf{b}^{(2)} \circ \mathbf{b}^{(3)}$$

$$\mathcal{C} = \mathcal{A} + \mathcal{B}$$

$$\begin{aligned} c_{s_1,s_2,s_3} &= a^{(1)}_{s_1} a^{(2)}_{s_2} a^{(3)}_{s_3} + b^{(1)}_{s_1} b^{(2)}_{s_2} b^{(3)}_{s_3} \\ &= \begin{pmatrix} a^{(1)}_{s_1} & b^{(1)}_{s_1} \end{pmatrix} \begin{pmatrix} a^{(2)}_{s_2} & 0 \\ 0 & b^{(2)}_{s_2} \end{pmatrix} \begin{pmatrix} a^{(3)}_{s_3} \\ b^{(3)}_{s_3} \end{pmatrix} \end{aligned}$$



From CP to MPS

Summing two rank-1 tensors

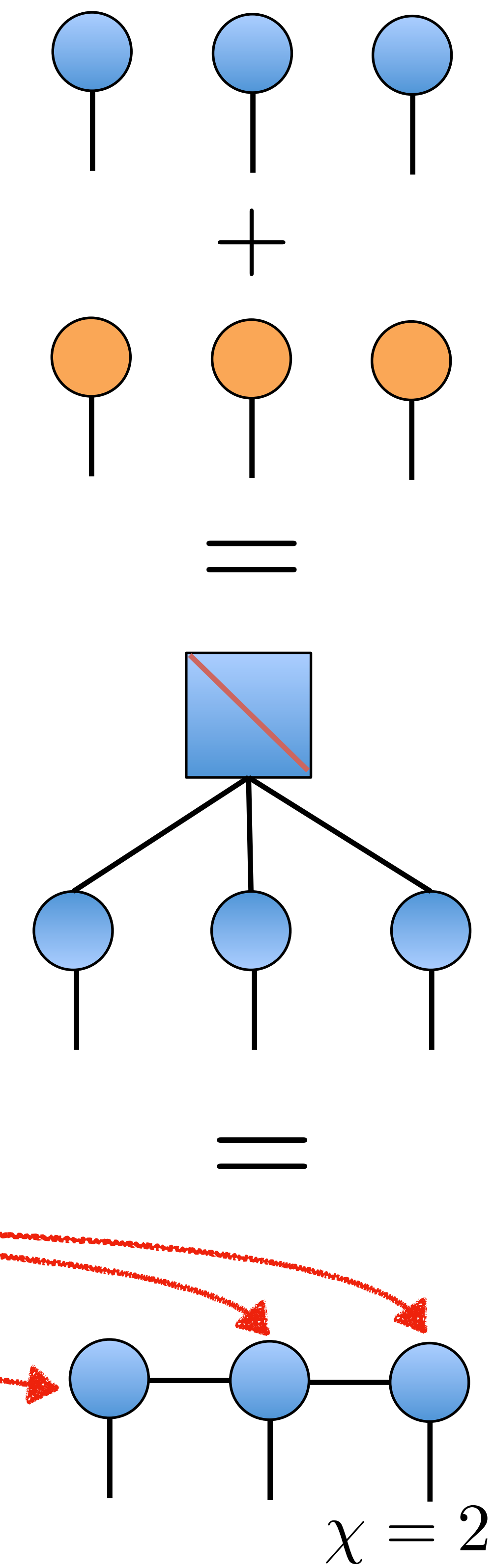
$$\mathcal{A} = \mathbf{a}^{(1)} \circ \mathbf{a}^{(2)} \circ \mathbf{a}^{(3)}$$

$$\mathcal{B} = \mathbf{b}^{(1)} \circ \mathbf{b}^{(2)} \circ \mathbf{b}^{(3)}$$

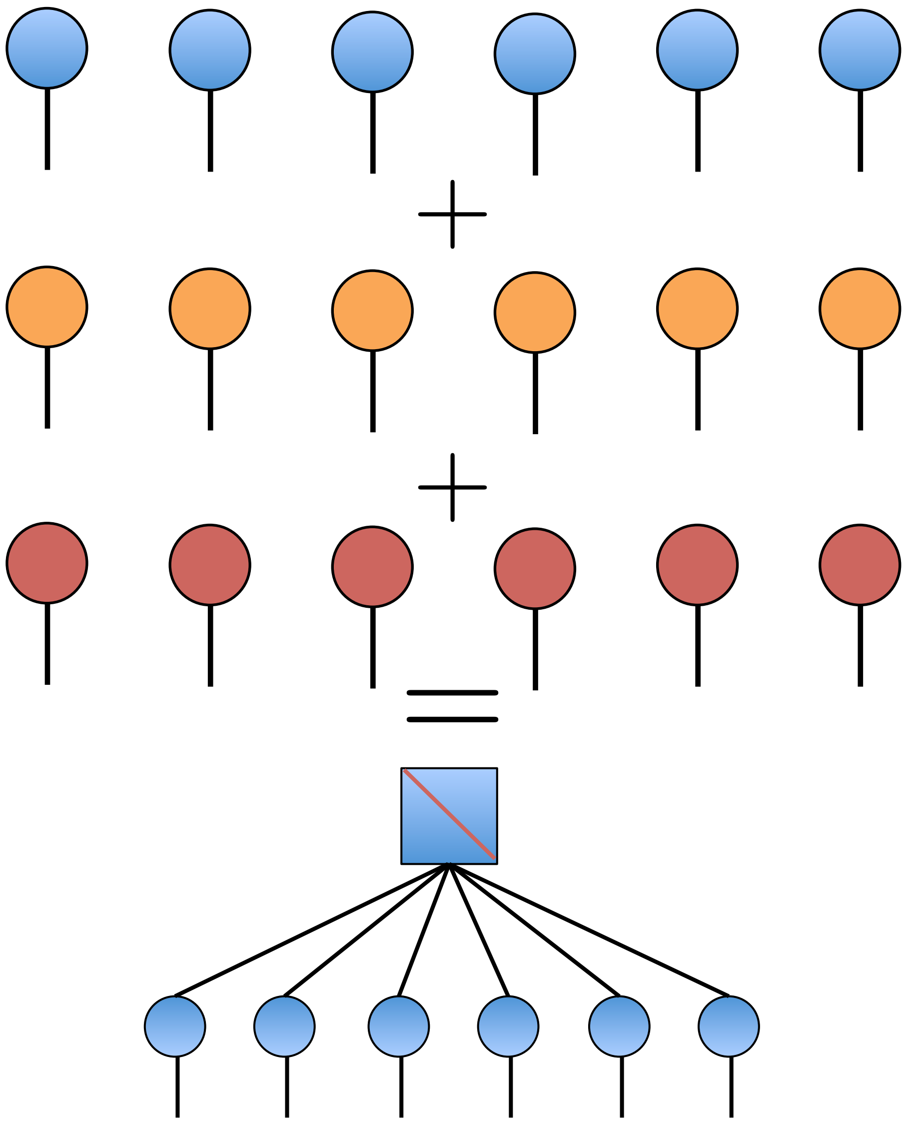
$$\mathcal{C} = \mathcal{A} + \mathcal{B}$$

$$\begin{aligned} c_{s_1,s_2,s_3} &= a^{(1)}_{s_1} a^{(2)}_{s_2} a^{(3)}_{s_3} + b^{(1)}_{s_1} b^{(2)}_{s_2} b^{(3)}_{s_3} \\ &= \begin{pmatrix} a^{(1)}_{s_1} & b^{(1)}_{s_1} \end{pmatrix} \begin{pmatrix} a^{(2)}_{s_2} & 0 \\ 0 & b^{(2)}_{s_2} \end{pmatrix} \begin{pmatrix} a^{(3)}_{s_3} \\ b^{(3)}_{s_3} \end{pmatrix} \end{aligned}$$

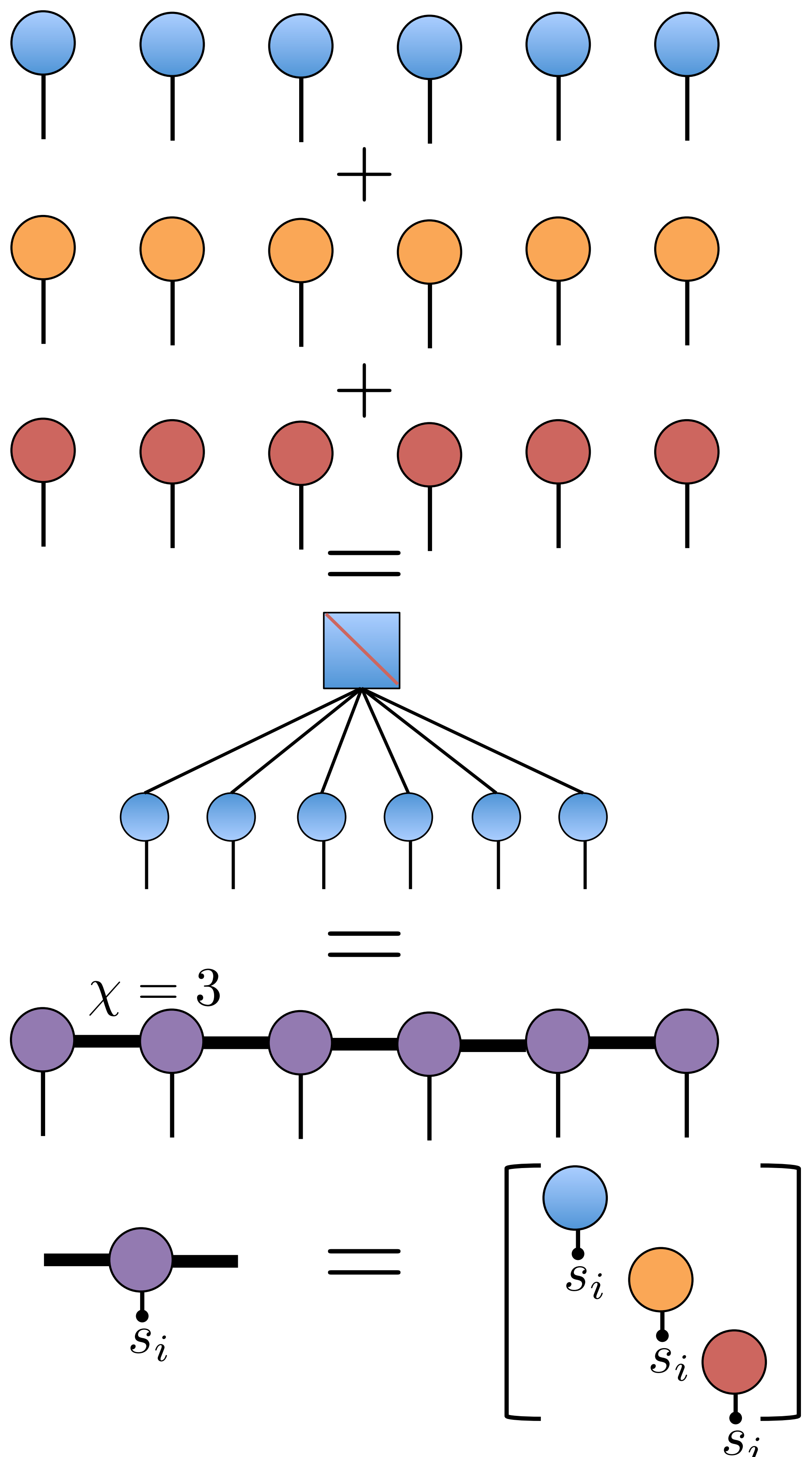
$$\mathbf{C}^{(1)}(s_1, :) \qquad \mathcal{C}^{(2)}(:, s_2, :) \qquad \mathbf{C}^{(3)}(:, s_3)$$



From CP to MPS



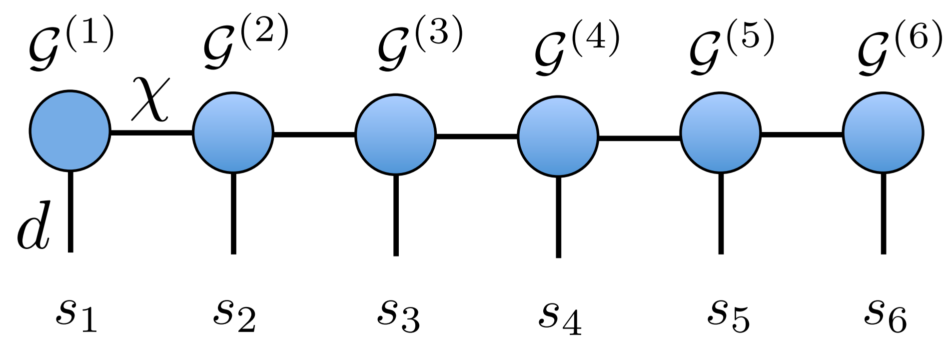
From CP to MPS



Matrix Product States and Matrix Product Operator

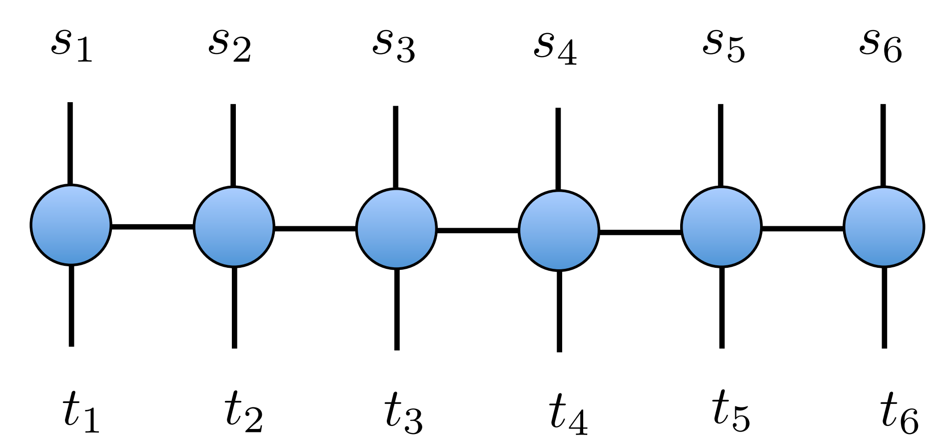
$$\mathcal{A} = \langle \mathcal{G}^{(1)}, \mathcal{G}^{(2)}, \cdots, \mathcal{G}^{(n)} \rangle$$
$$= \mathcal{G}^{(1)} \times_1 \mathcal{G}^{(2)} \times_1 \cdots \times_1 \mathcal{G}^{(n)}$$

$$a_{s_1, s_2, \cdots, s_n} = \mathbf{G}^{(1)}_{s_1} \mathbf{G}^{(2)}_{s_2} \cdots \mathbf{G}^{(n)}_{s_n}$$



$\approx nd\chi^2$ parameters

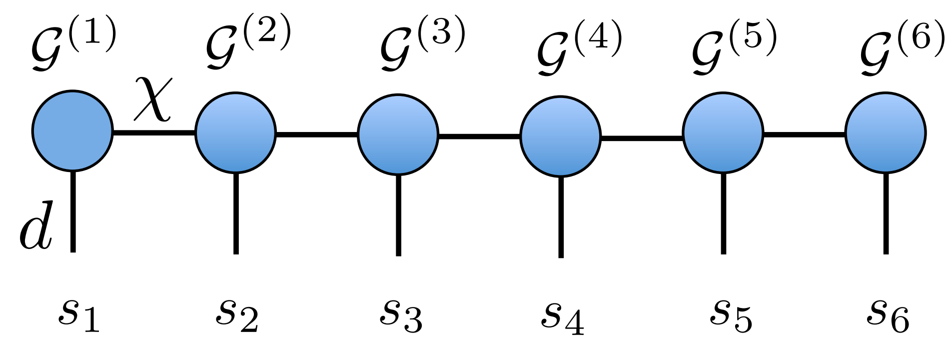
$$a_{s_1, s_2, \cdots, s_n} = \mathbf{G}^{(1)}_{s_1, t_1} \mathbf{G}^{(2)}_{s_2, t_2} \cdots \mathbf{G}^{(n)}_{s_n, t_n}$$



Matrix Product States and Matrix Product Operator

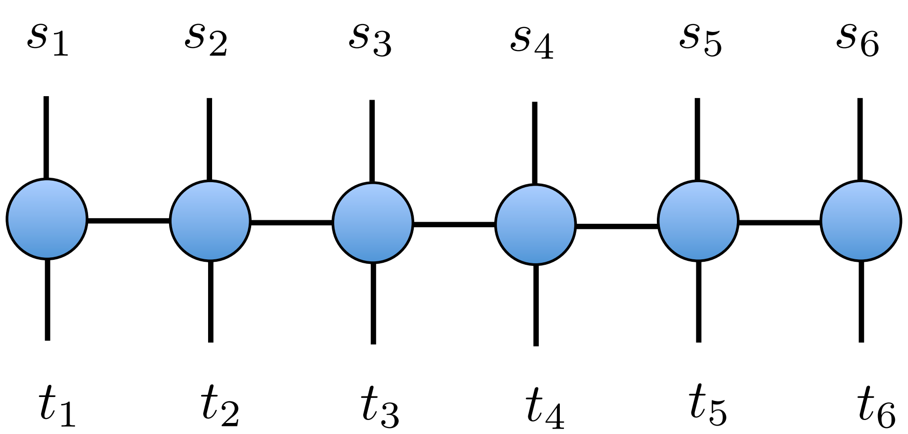
$$\mathcal{A} = \langle \mathcal{G}^{(1)}, \mathcal{G}^{(2)}, \cdots, \mathcal{G}^{(n)} \rangle$$
$$= \mathcal{G}^{(1)} \times_1 \mathcal{G}^{(2)} \times_1 \cdots \times_1 \mathcal{G}^{(n)}$$

$$a_{s_1, s_2, \cdots, s_n} = \mathbf{G}^{(1)}_{s_1} \mathbf{G}^{(2)}_{s_2} \cdots \mathbf{G}^{(n)}_{s_n}$$



$\approx n d \chi^2$ parameters

$$a_{s_1, s_2, \cdots, s_n} = \mathbf{G}^{(1)}_{s_1, t_1} \mathbf{G}^{(2)}_{s_2, t_2} \cdots \mathbf{G}^{(n)}_{s_n, t_n}$$

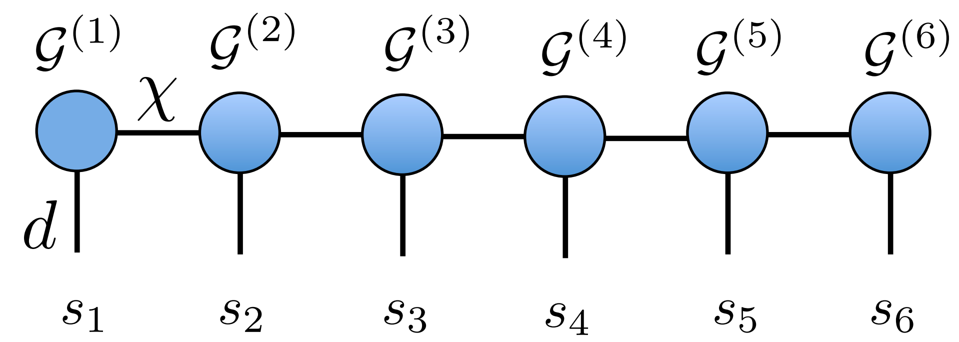


with χ=1, MPS is a rank-one tensor

Matrix Product States and Matrix Product Operator

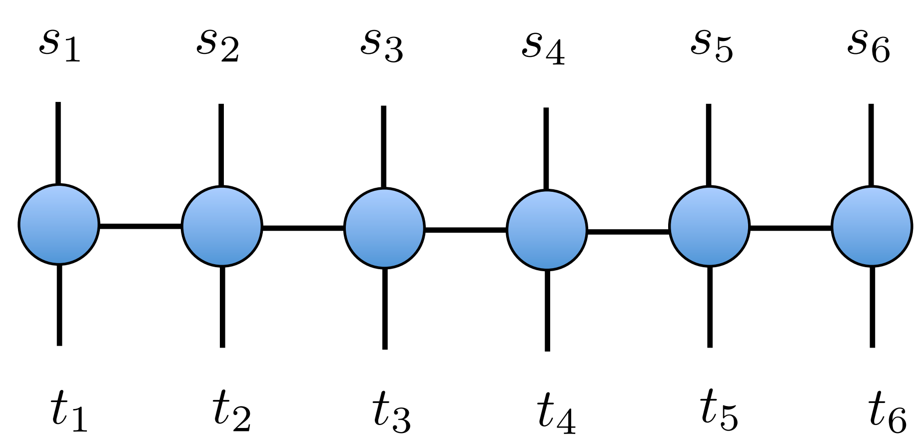
$$\mathcal{A} = \langle \mathcal{G}^{(1)}, \mathcal{G}^{(2)}, \cdots, \mathcal{G}^{(n)} \rangle$$
$$= \mathcal{G}^{(1)} \times_1 \mathcal{G}^{(2)} \times_1 \cdots \times_1 \mathcal{G}^{(n)}$$

$$a_{s_1,s_2,\cdots,s_n} = \mathbf{G}^{(1)}_{s_1} \mathbf{G}^{(2)}_{s_2} \cdots \mathbf{G}^{(n)}_{s_n}$$



$\approx nd\chi^2$ parameters

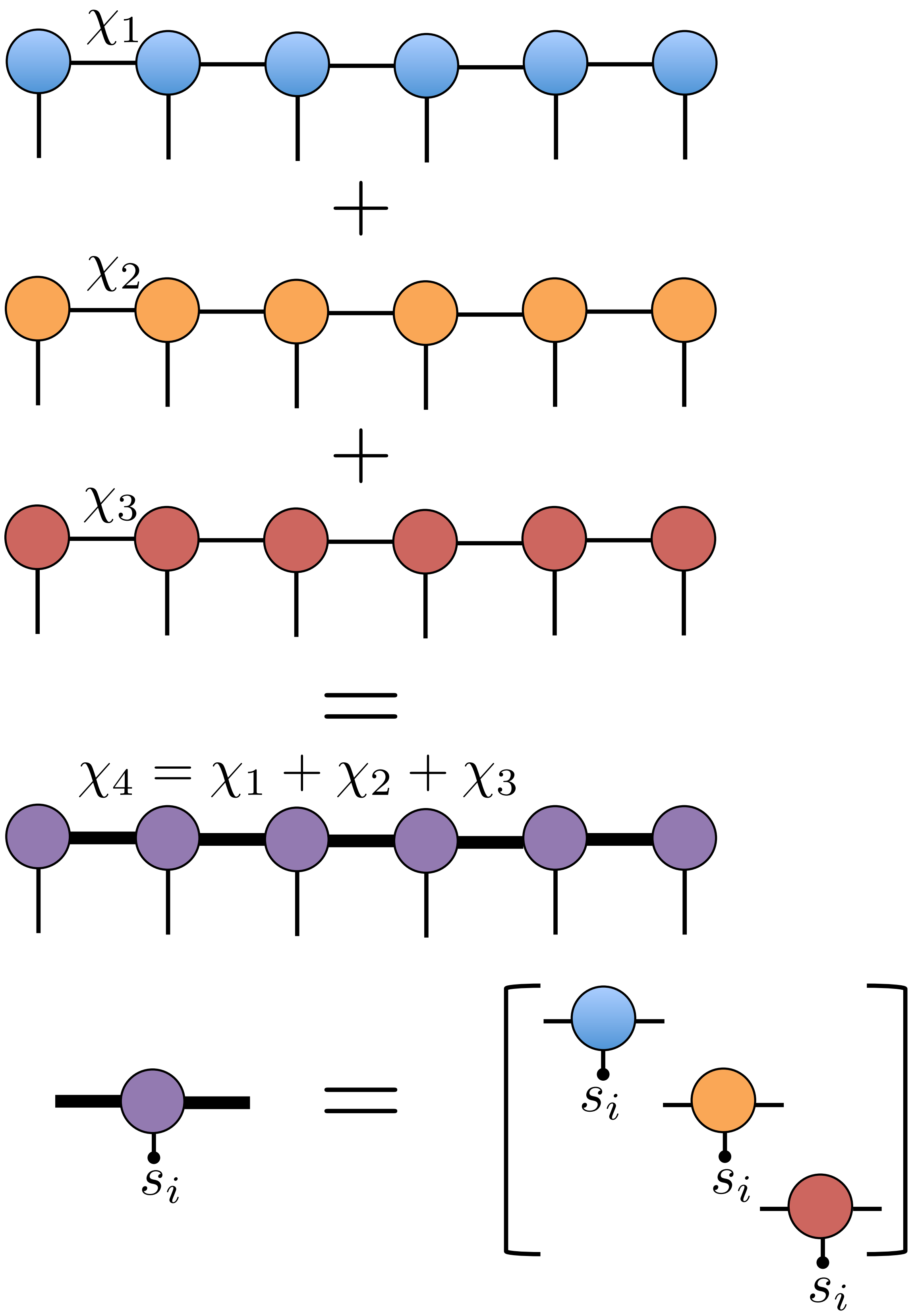
$$a_{s_1,s_2,\cdots,s_n} = \mathbf{G}^{(1)}_{s_1,t_1} \mathbf{G}^{(2)}_{s_2,t_2} \cdots \mathbf{G}^{(n)}_{s_n,t_n}$$



with χ=1, MPS is a rank-one tensor

besides CP, MPS is another generalization of rank-one tensors

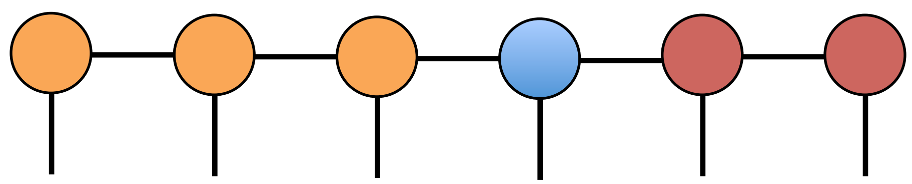
Summing MPSes

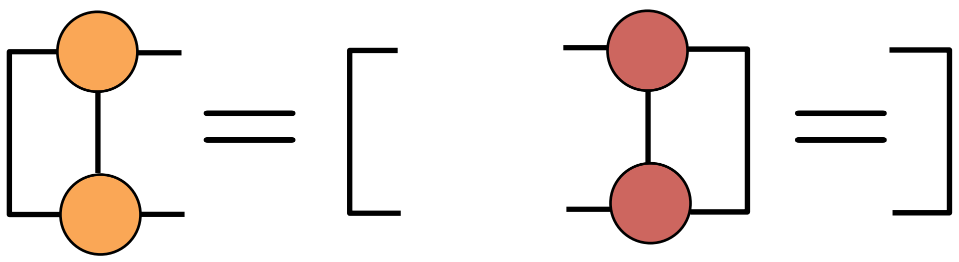


Canonical forms of MPS

Canonical forms of MPS

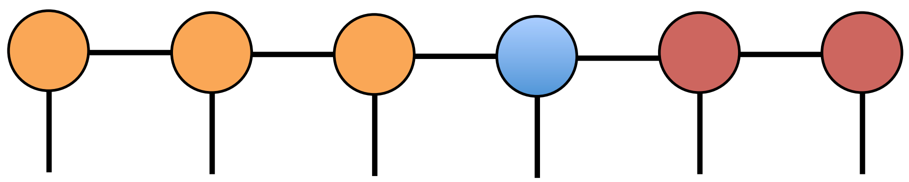
Analogous to the Tucker decomposition and HOSVDs, MPS has the benefits of orthogonality.

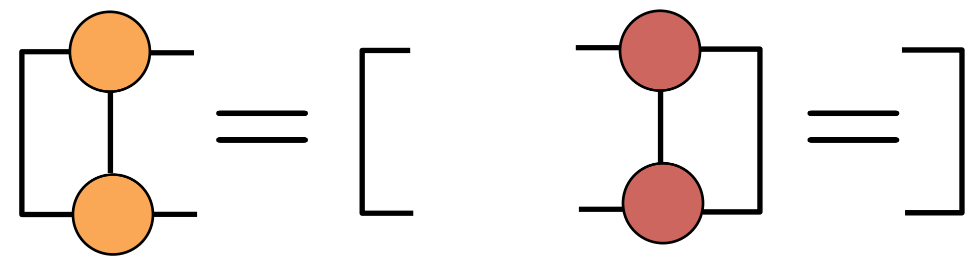




Canonical forms of MPS

Analogous to the Tucker decomposition and HOSVDs, MPS has the benefits of orthogonality.

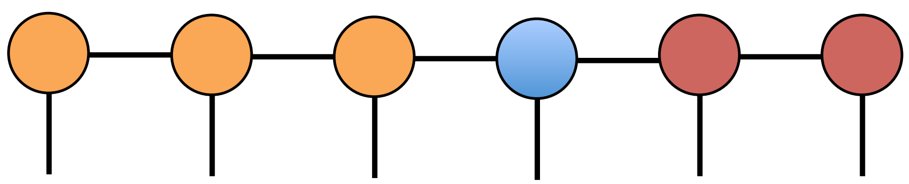


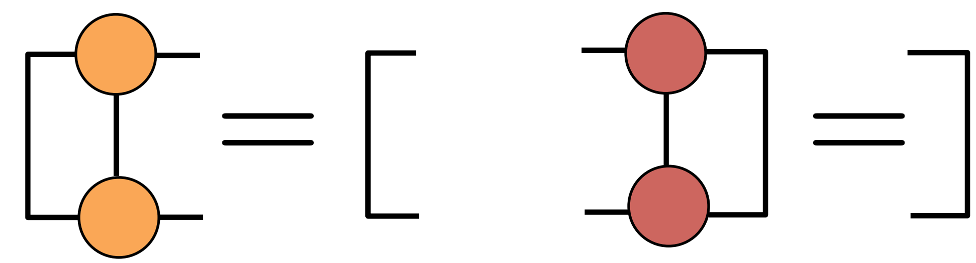


Benefits

Canonical forms of MPS

Analogous to the Tucker decomposition and HOSVDs, MPS has the benefits of orthogonality.



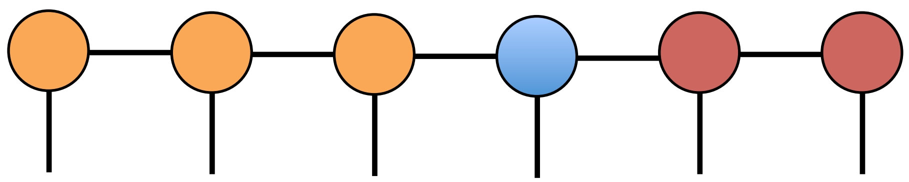


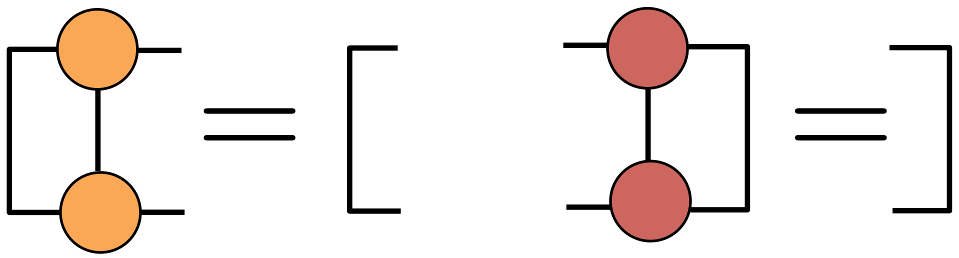
Benefits

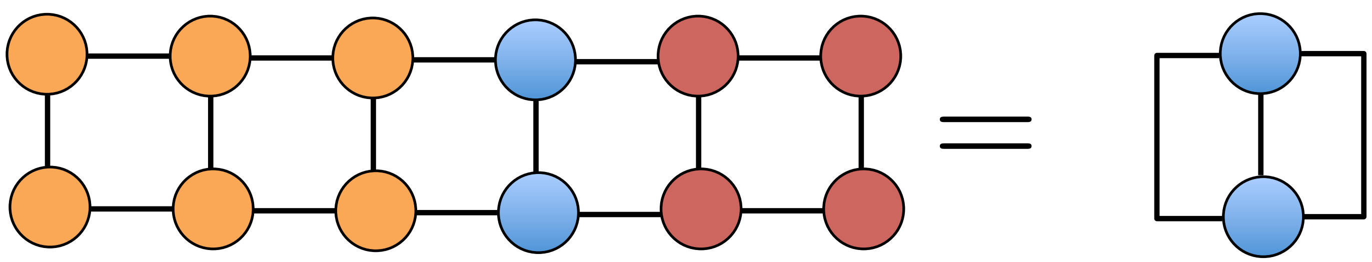
- Fixed gauge, no ambiguity

Canonical forms of MPS

Analogous to the Tucker decomposition and HOSVDs, MPS has the benefits of orthogonality.





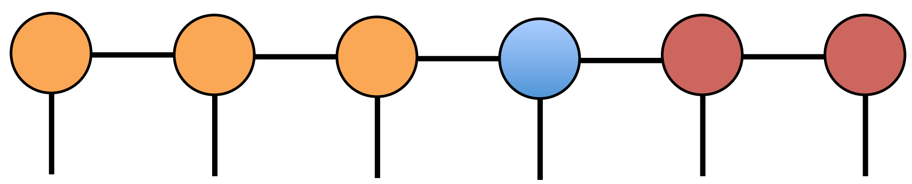


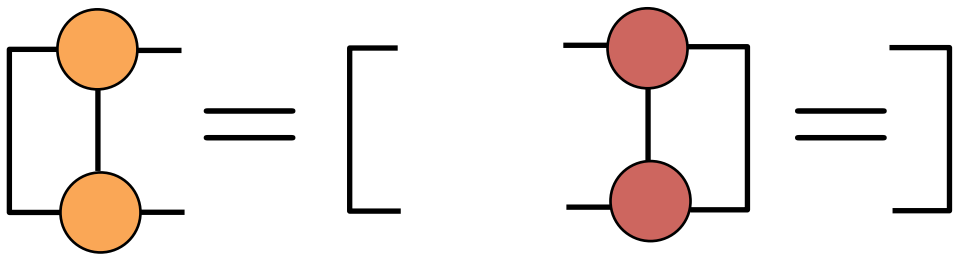
Benefits

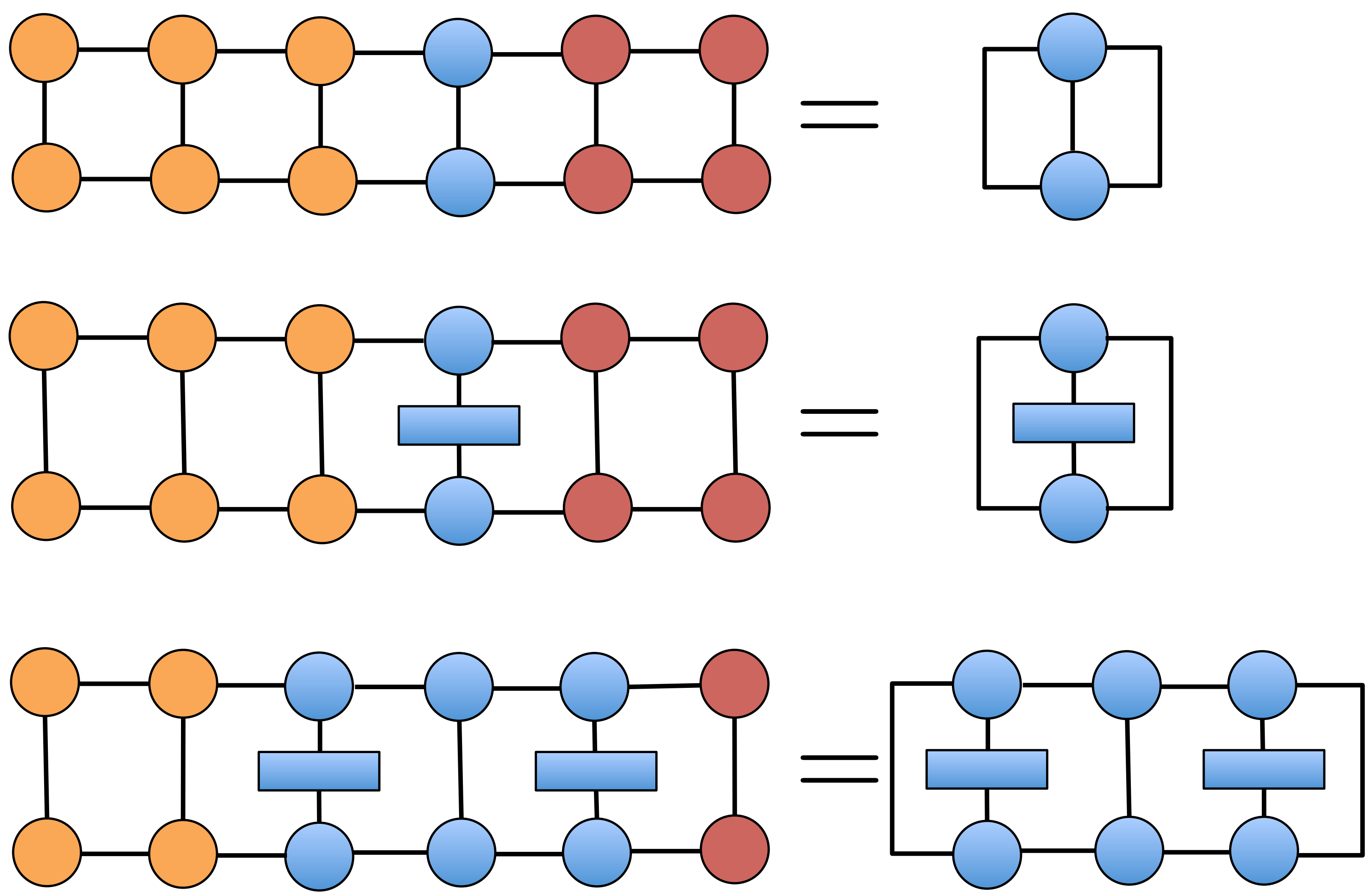
- Fixed gauge, no ambiguity
- Easy norm computation

Canonical forms of MPS

Analogous to the Tucker decomposition and HOSVDs, MPS has the benefits of orthogonality.





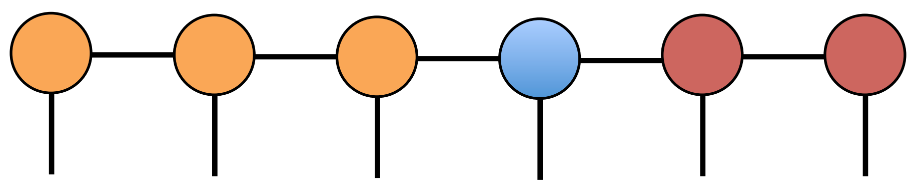


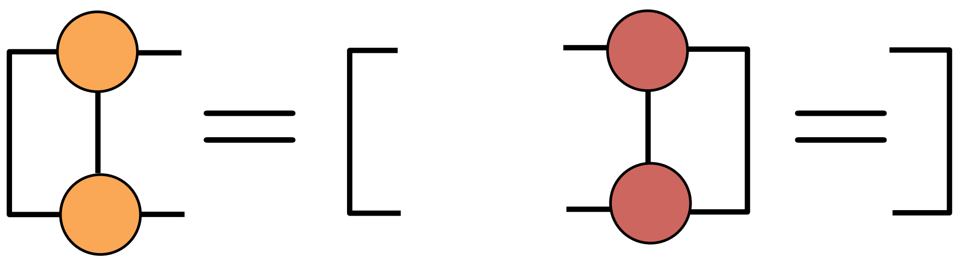
Benefits

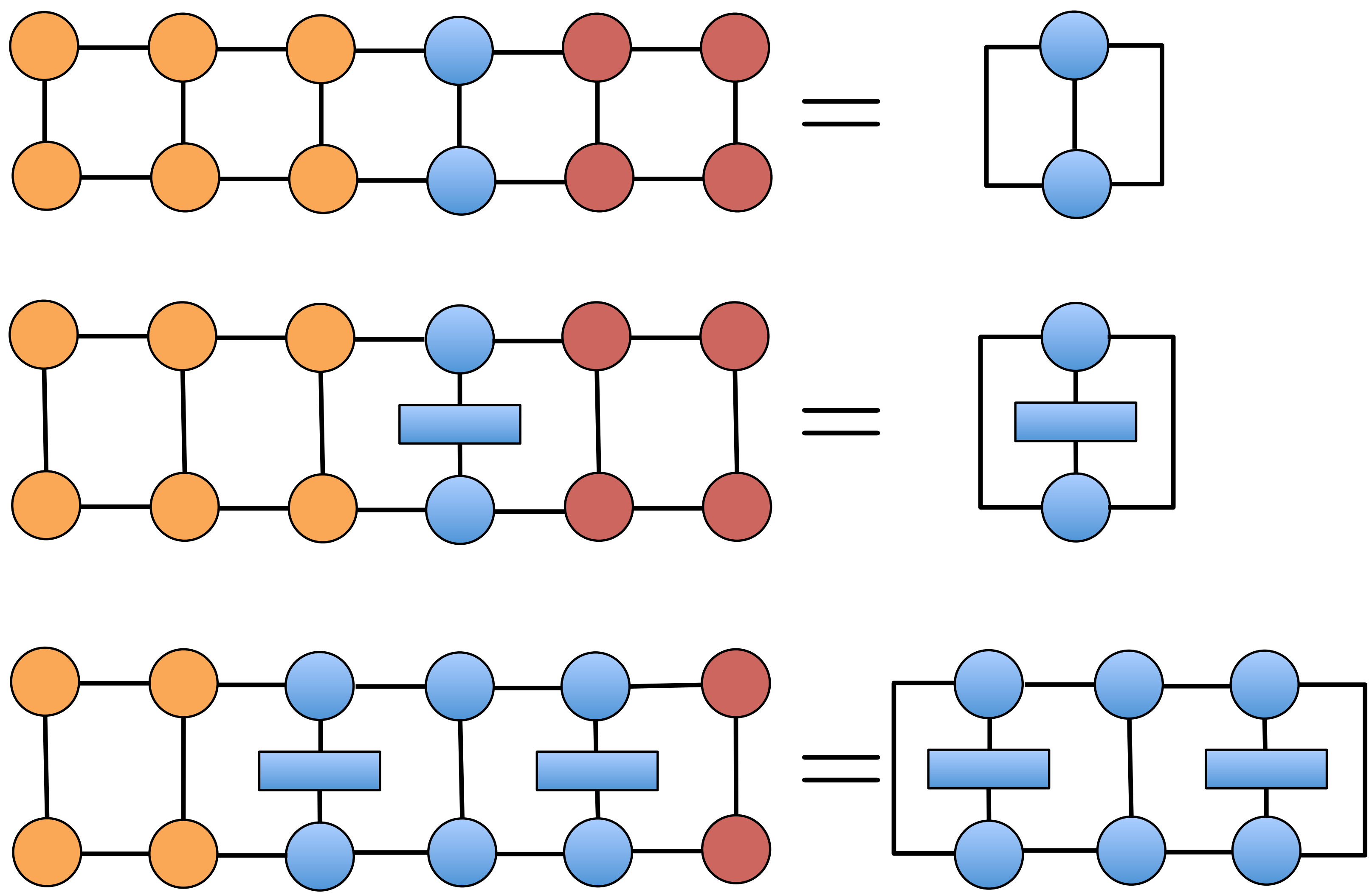
- Fixed gauge, no ambiguity
- Easy norm computation
- Easy expectation/ correlation computation

Canonical forms of MPS

Analogous to the Tucker decomposition and HOSVDs, MPS has the benefits of orthogonality.







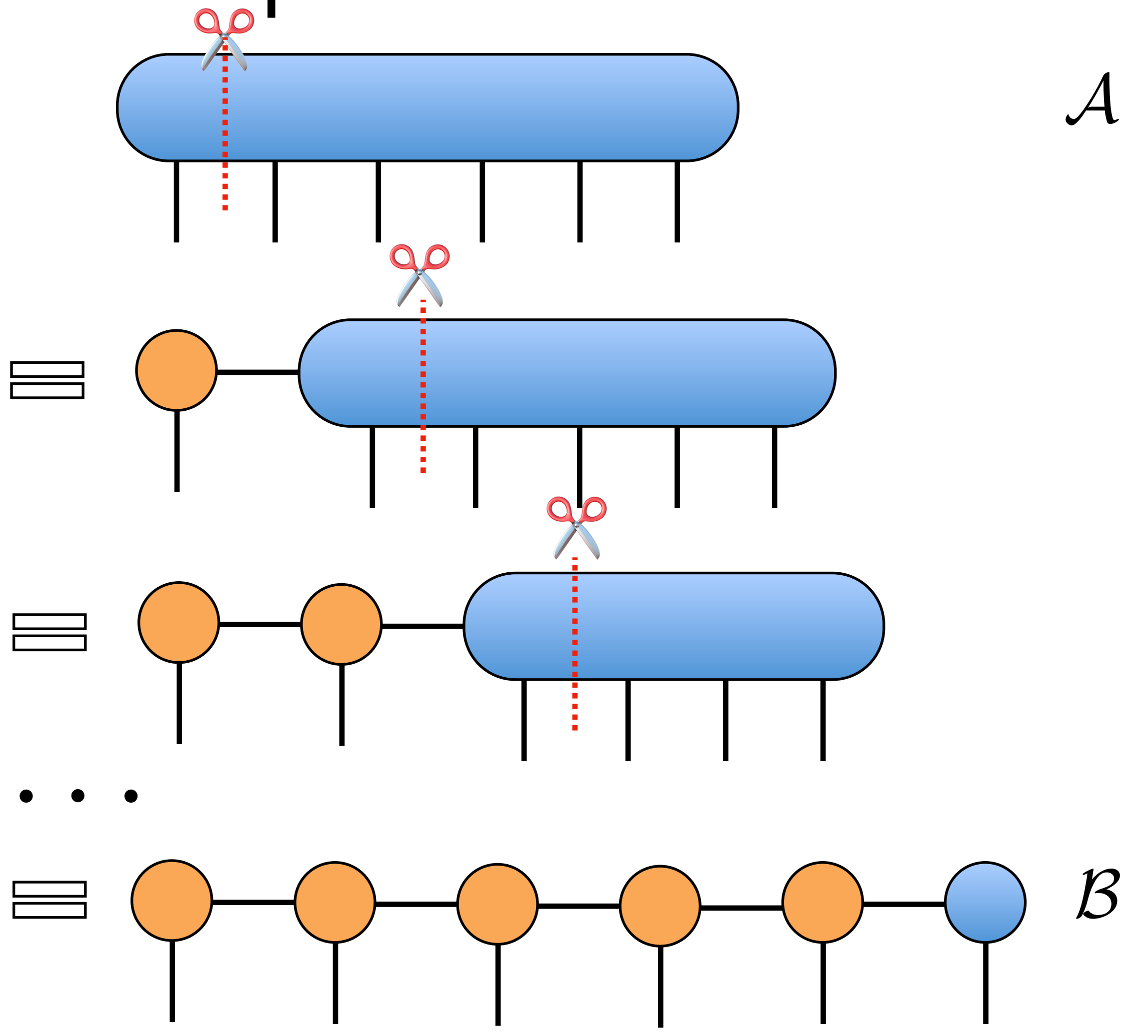
Benefits

- Fixed gauge, no ambiguity
- Easy norm computation
- Easy expectation/ correlation computation
- Always good conditioned

Convert a raw tensor to a MPS: sequential SVDs

Convert a raw tensor to a MPS: sequential SVDs

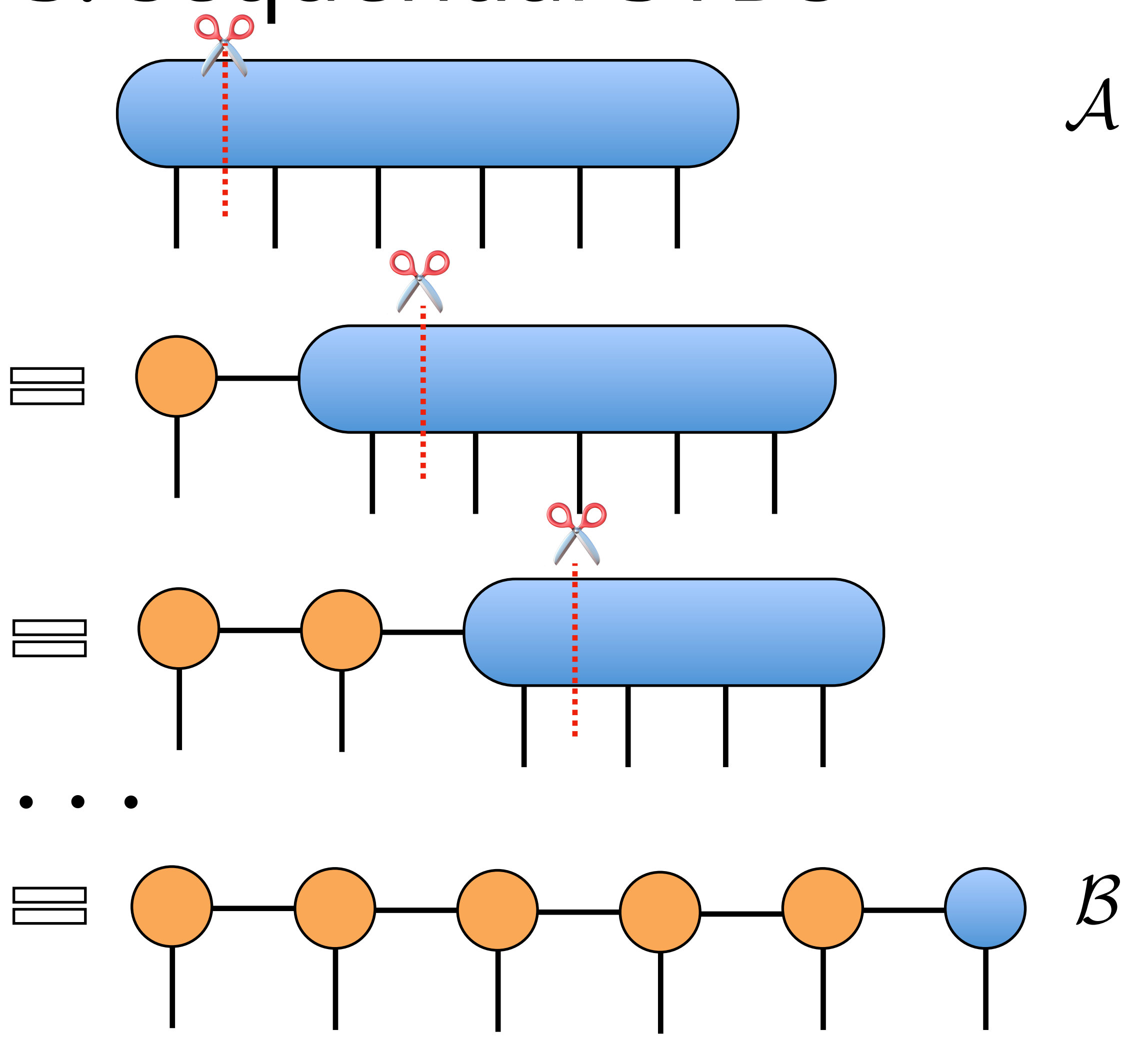
analogous to HOSVD, but processes every mode ***one by one***



Convert a raw tensor to a MPS: sequential SVDs

analogous to HOSVD, but processes every mode ***one by one***

matrices encountered during SVDs are larger than HOSVD



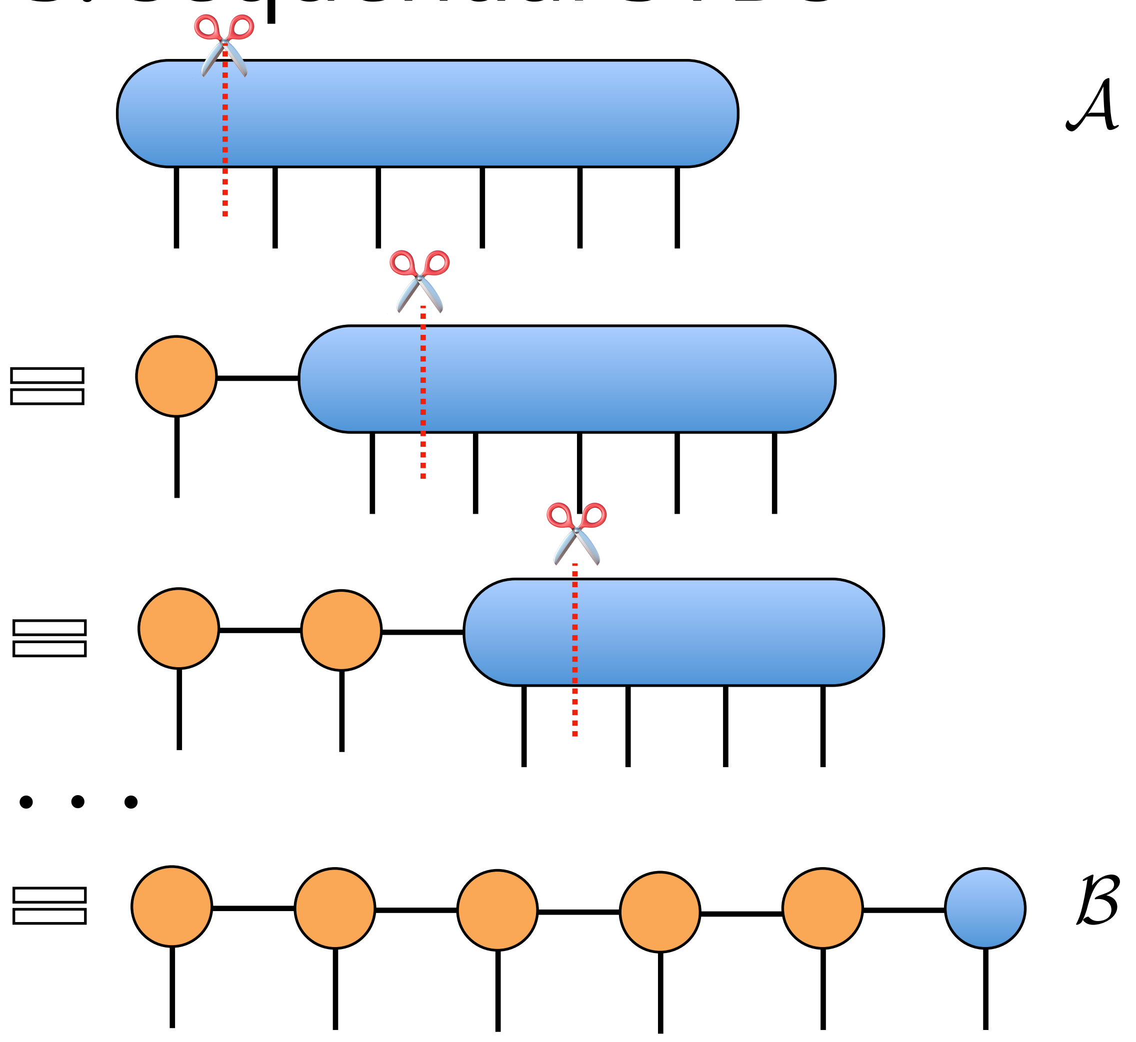
Convert a raw tensor to a MPS: sequential SVDs

analogous to HOSVD, but processes every mode ***one by one***

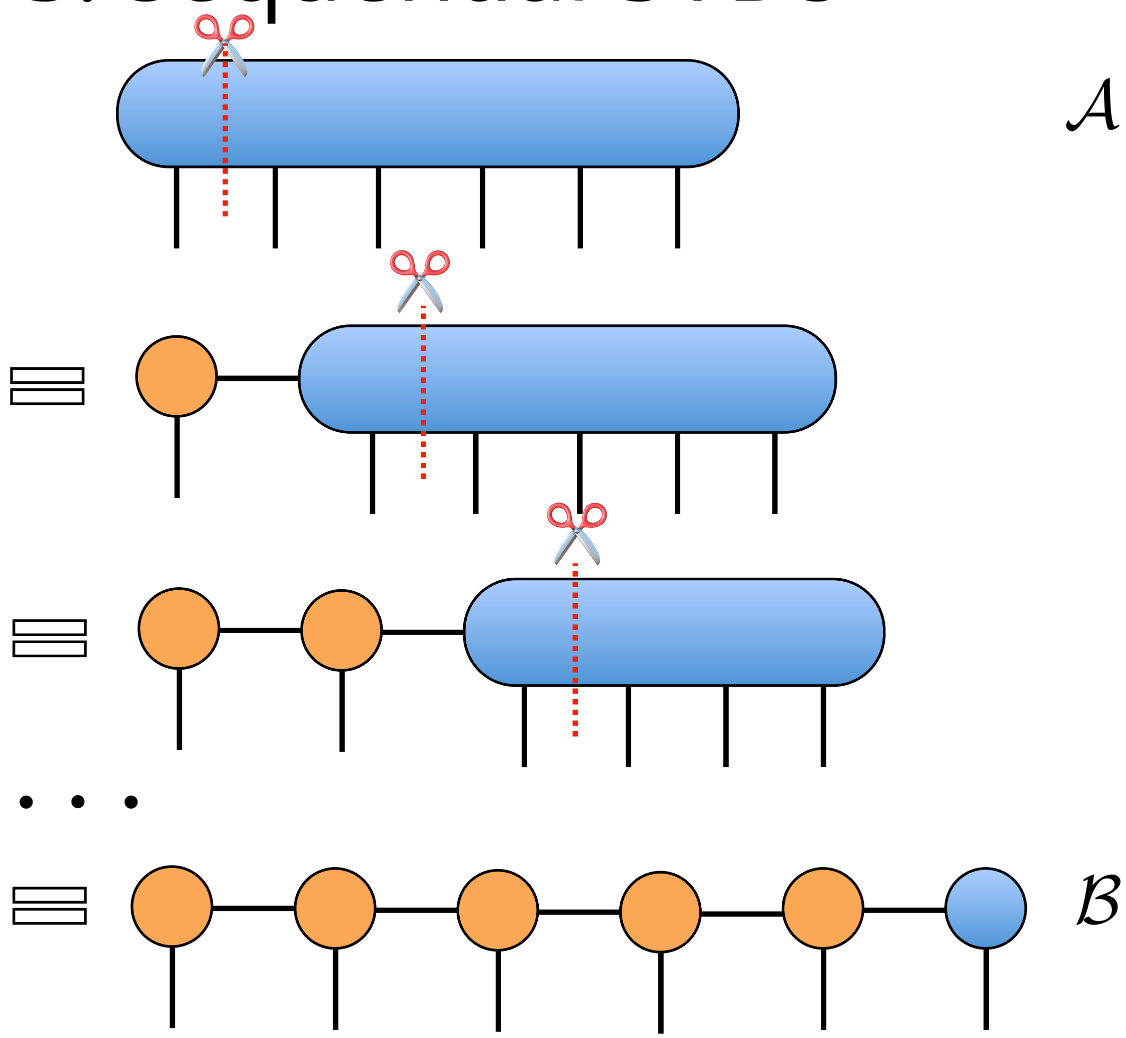
matrices encountered during SVDs are larger than HOSVD

error made in compression

$$\|\mathcal{A} - \mathcal{B}\|_F \leq \sqrt{\sum_{i=1}^{n} \epsilon_i^2}$$



Convert a raw tensor to a MPS: sequential SVDs



analogous to HOSVD, but processes every mode ***one by one***

matrices encountered during SVDs are larger than HOSVD

error made in compression

$$\|\mathcal{A} - \mathcal{B}\|_F \leq \sqrt{\sum_{i=1}^{n} \epsilon_i^2}$$

sequential SVD is quasi-optimal

$$\|\mathcal{A} - \mathcal{B}\|_F \leq \sqrt{n}\, \|\mathcal{A} - \mathcal{B}_{\text{best}}\|_F$$

Oseledets, SIAM J. Sci. Comput. 33, 2295 (2011)

Convert a raw tensor to a MPS: sequential SVDs

analogous to HOSVD, but processes every mode ***one by one***

matrices encountered during SVDs are larger than HOSVD

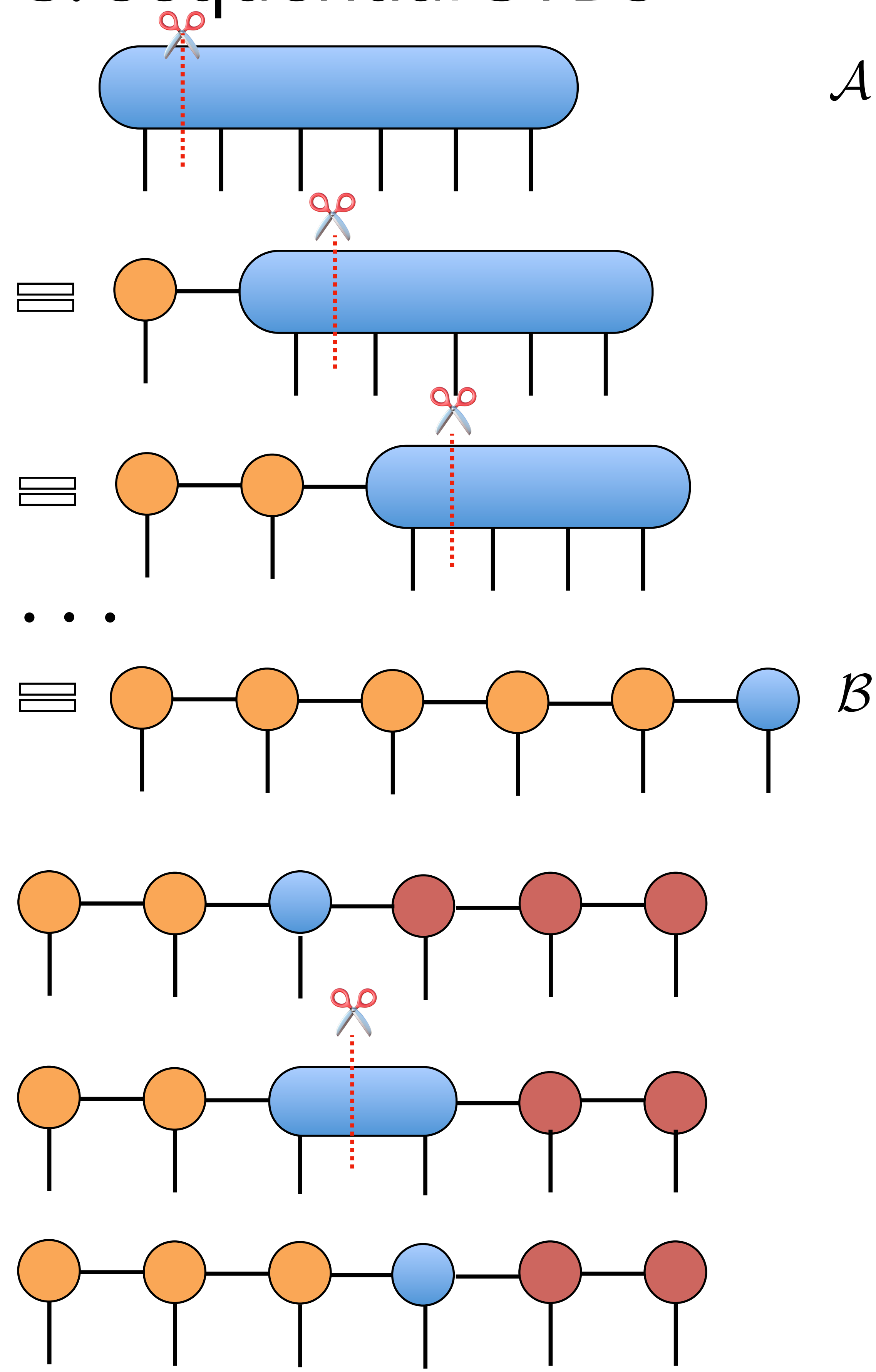
error made in compression

$$\|\mathcal{A}-\mathcal{B}\|_F \leq \sqrt{\sum_{i=1}^{n} \epsilon_i^2}$$

sequential SVD is quasi-optimal

$$\|\mathcal{A}-\mathcal{B}\|_F \leq \sqrt{n}\,\|\mathcal{A}-\mathcal{B}_{\text{best}}\|_F$$

can be refined iteratively (known as recompression, rounding)



Oseledets, SIAM J. Sci. Comput. 33, 2295 (2011)

Application: compressing neural networks

Application: compressing neural networks

Deep neural networks are most popular machine learning models.

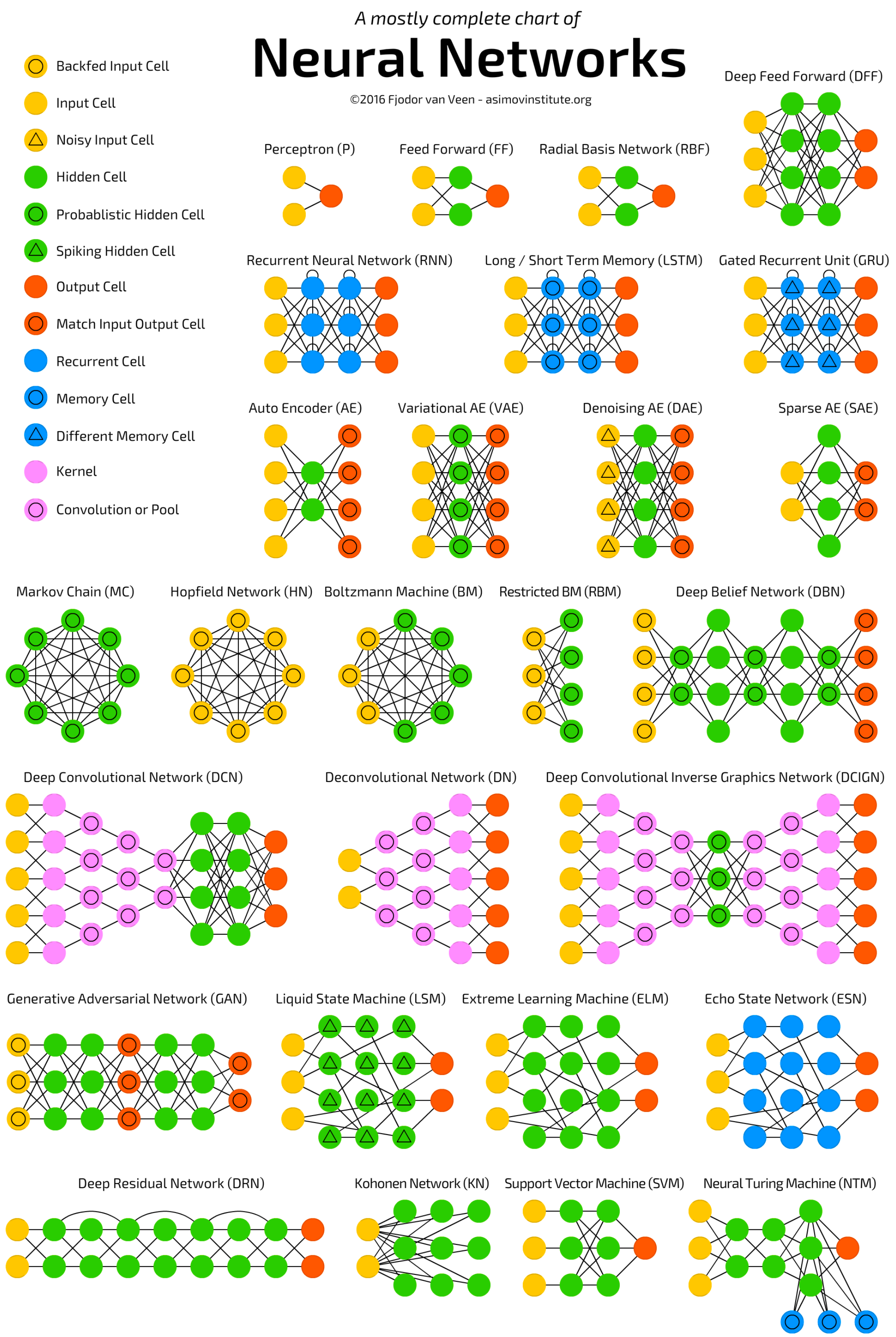
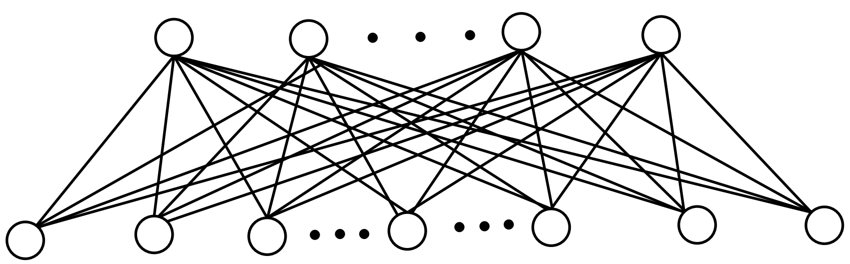


Image courtesy: Fjodor van Veen

Application: compressing neural networks

Deep neural networks are most popular machine learning models.

Most of the parameters are contained in fully-connected layers, which are ***dense matrices***.



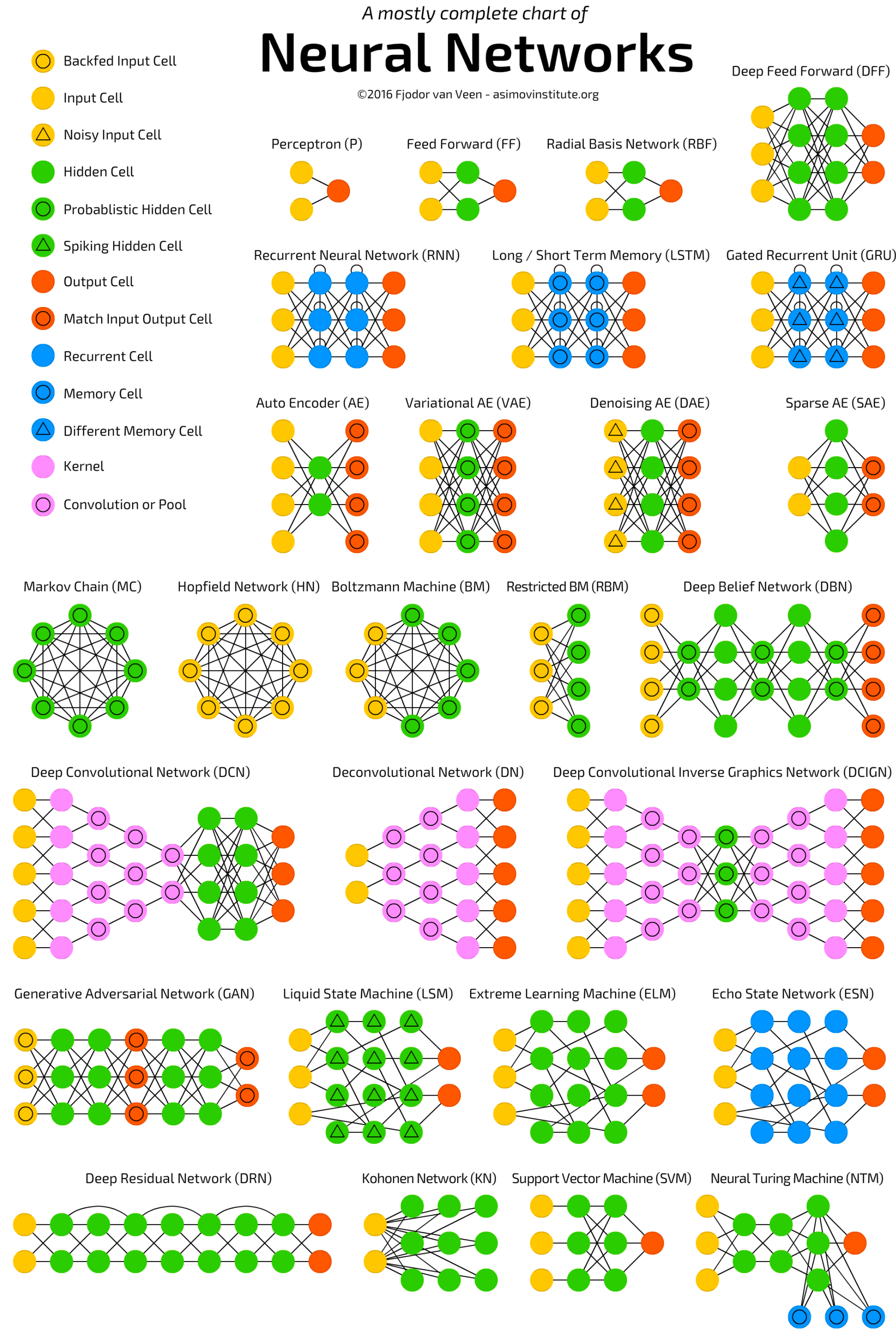
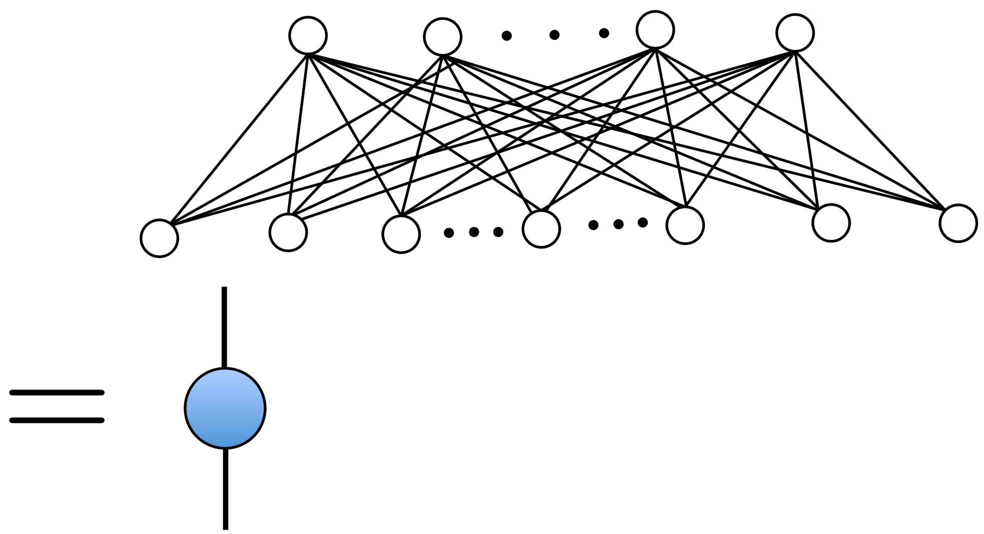


Image courtesy: Fjodor van Veen

Application: compressing neural networks

Deep neural networks are most popular machine learning models.

Most of the parameters are contained in fully-connected layers, which are ***dense matrices***.



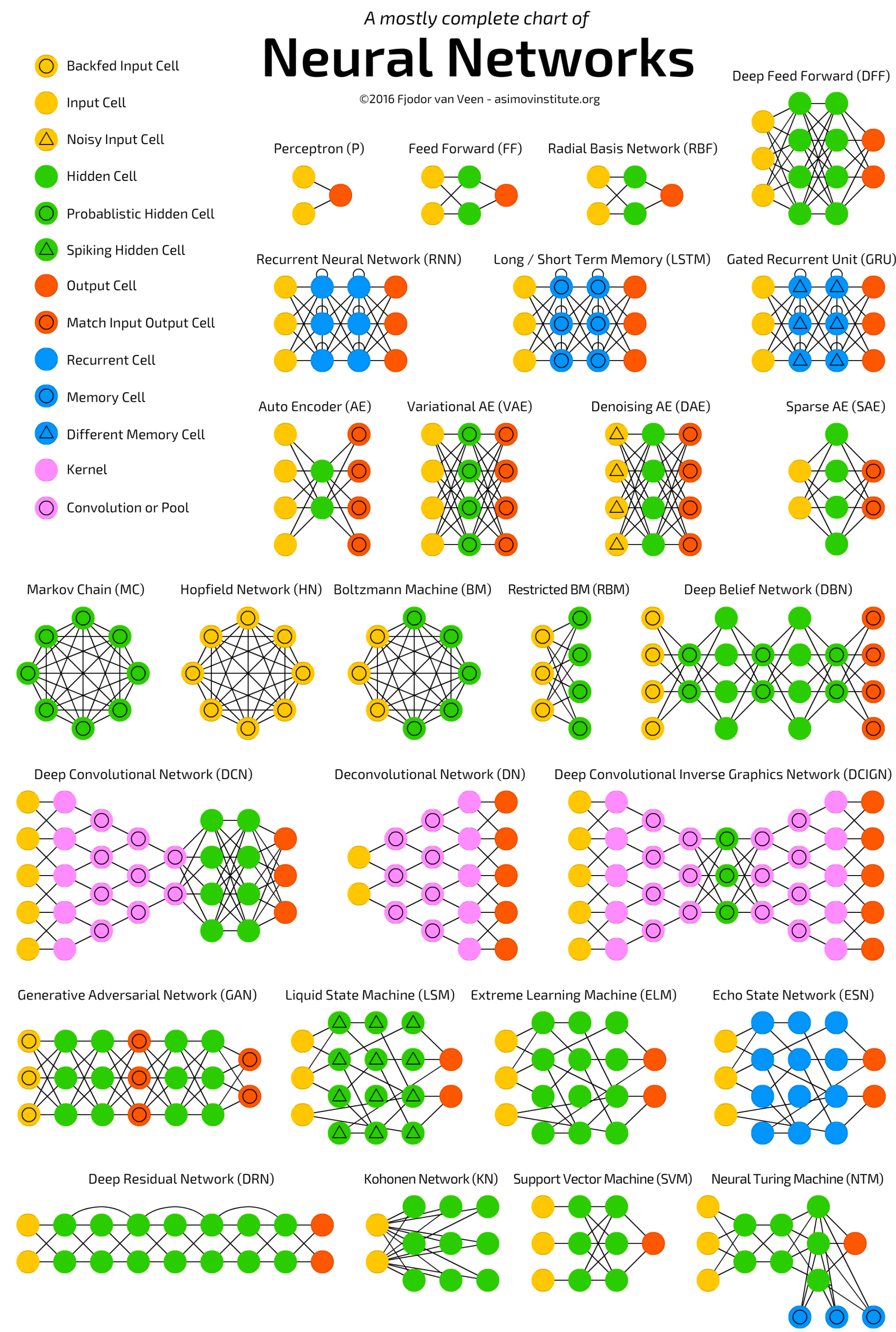
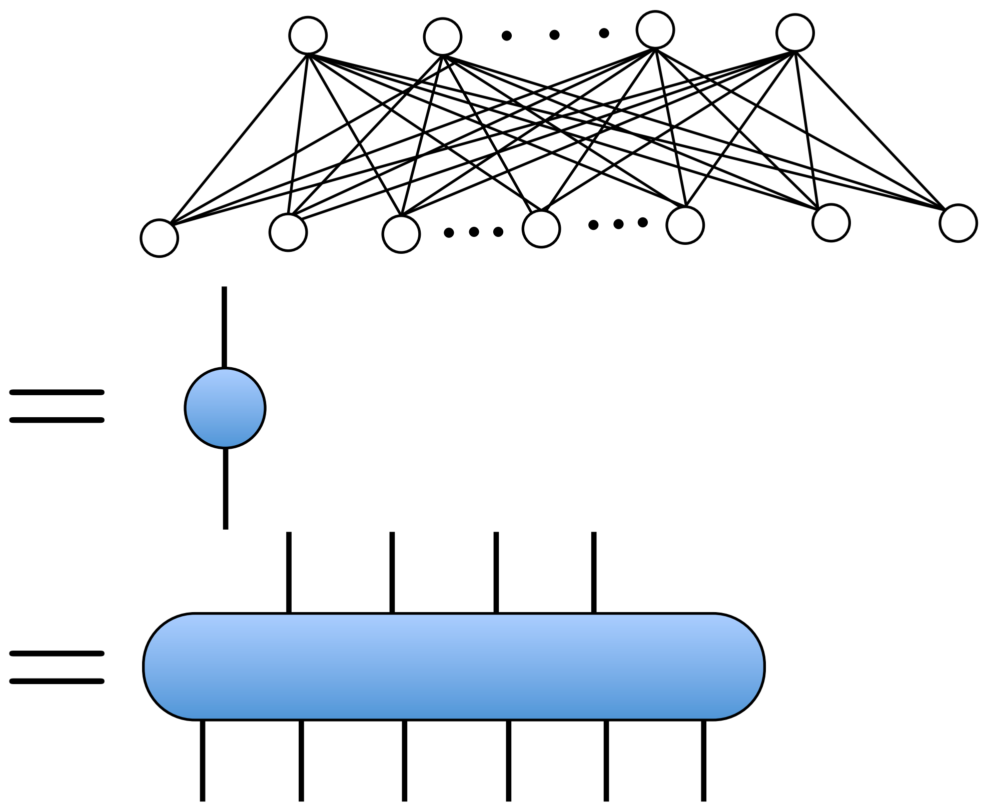


Image courtesy: Fjodor van Veen

Application: compressing neural networks

Deep neural networks are most popular machine learning models.

Most of the parameters are contained in fully-connected layers, which are ***dense matrices***.



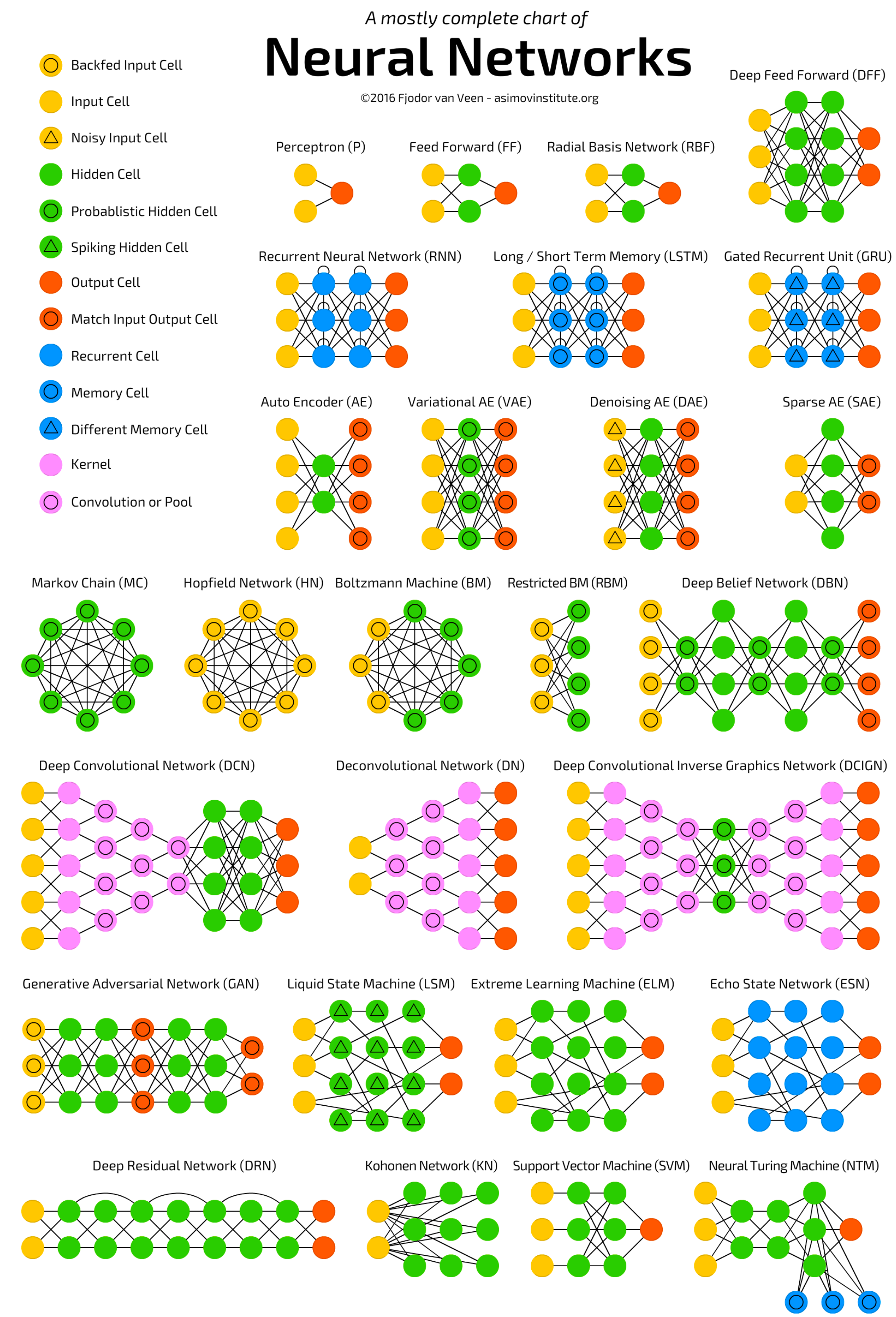
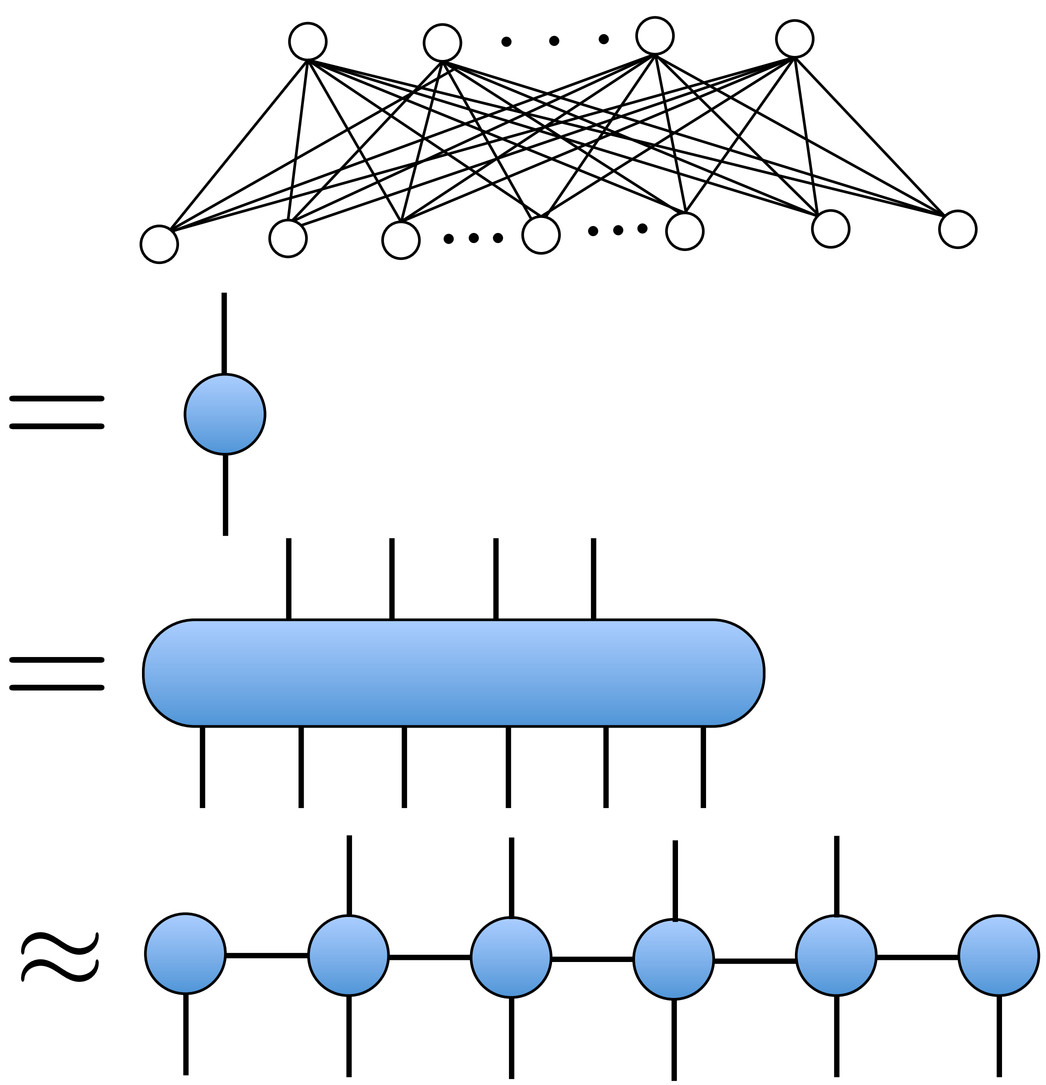


Image courtesy: Fjodor van Veen

Application: compressing neural networks

Deep neural networks are most popular machine learning models.

Most of the parameters are contained in fully-connected layers, which are ***dense matrices***.



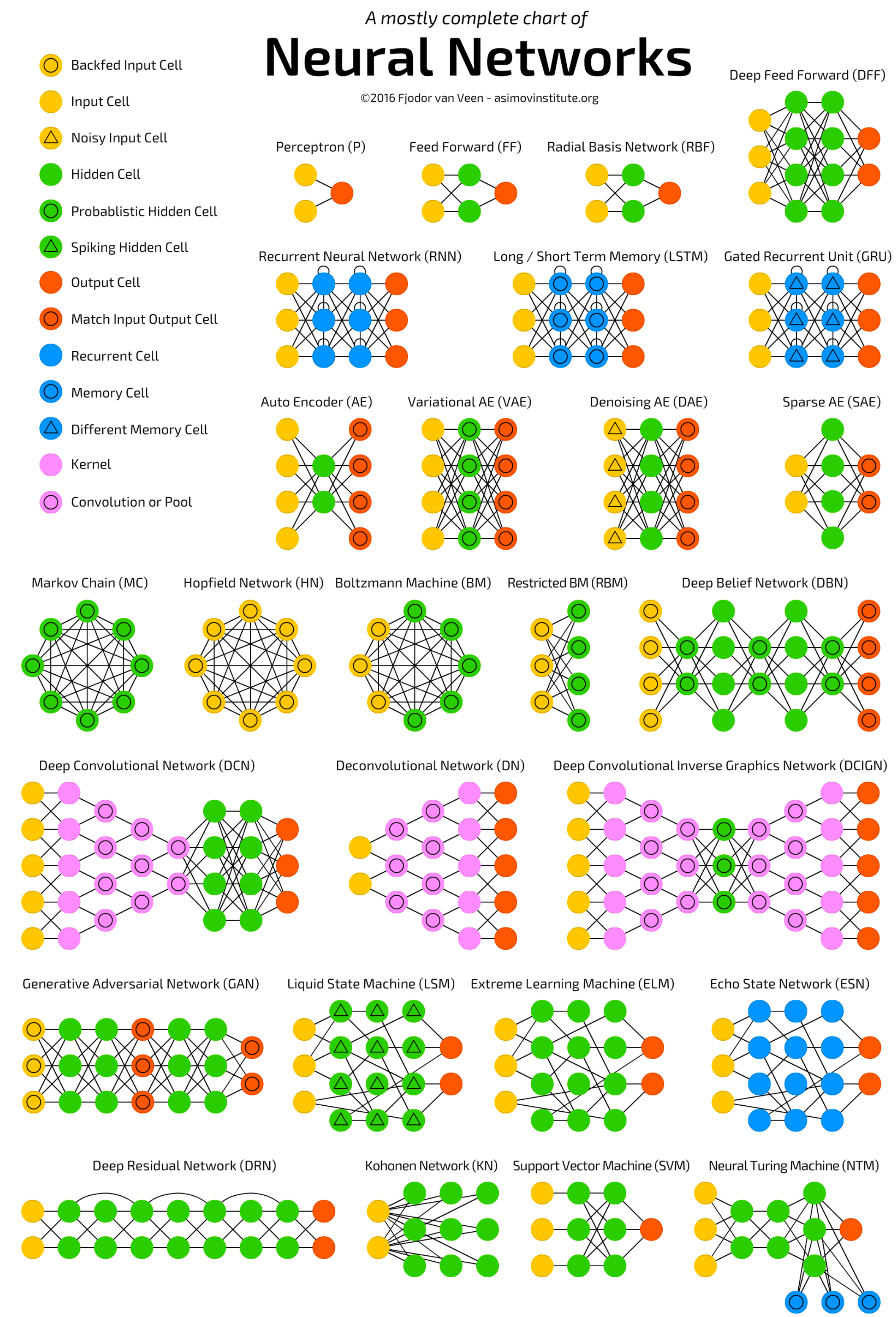
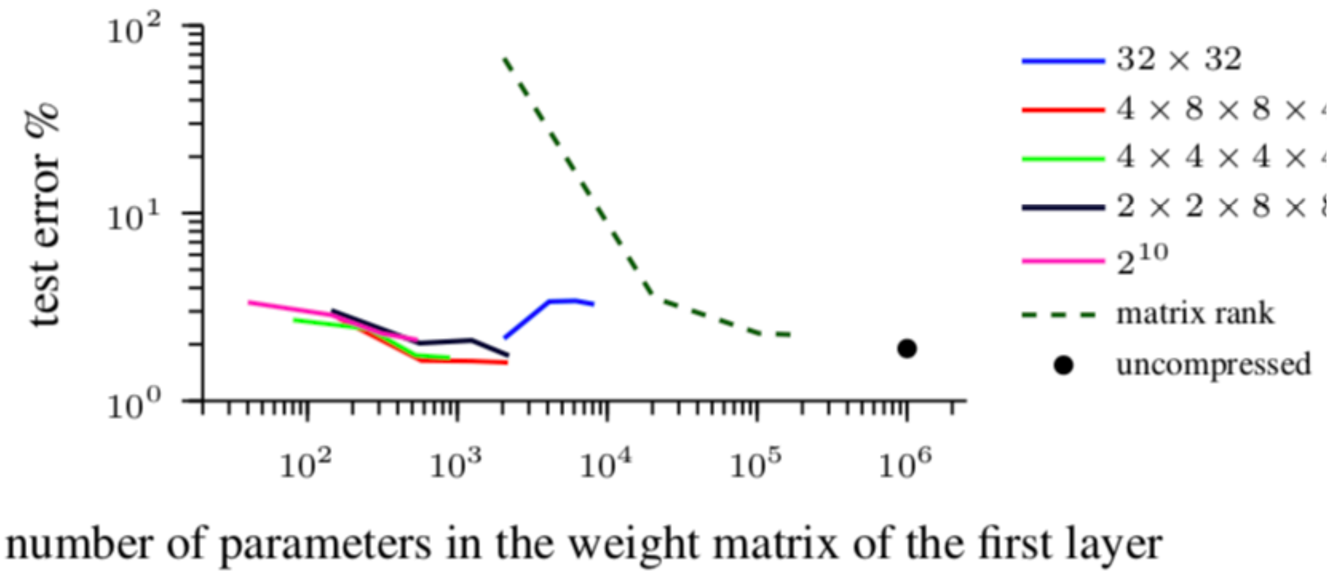
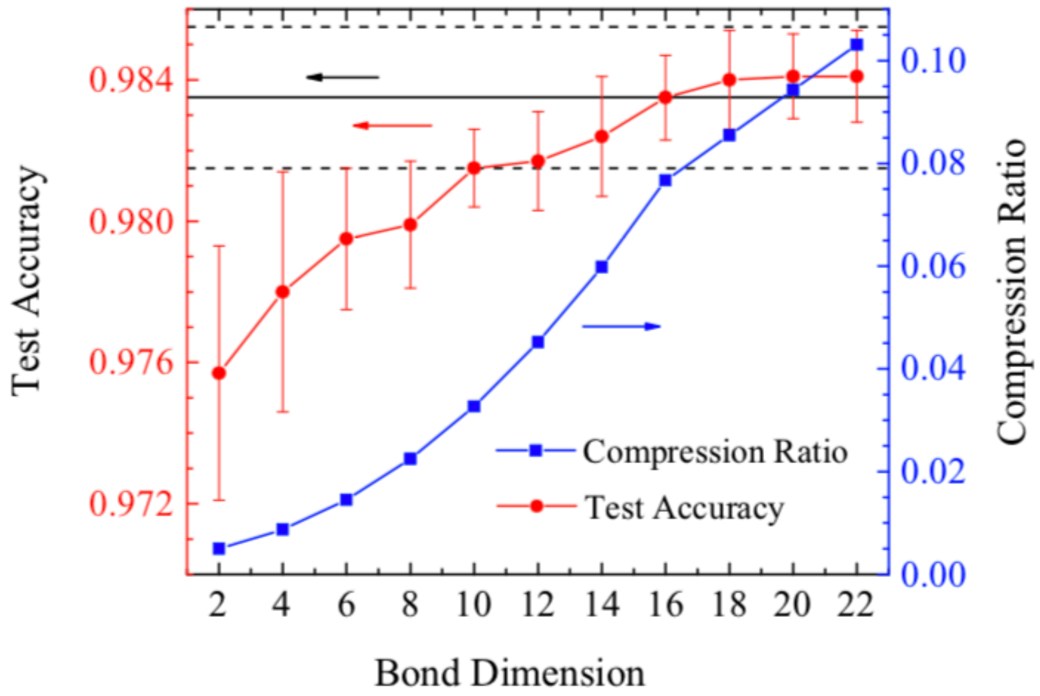


Image courtesy: Fjodor van Veen

Application: compressing neural networks

On 2-layer perceptron, MNIST



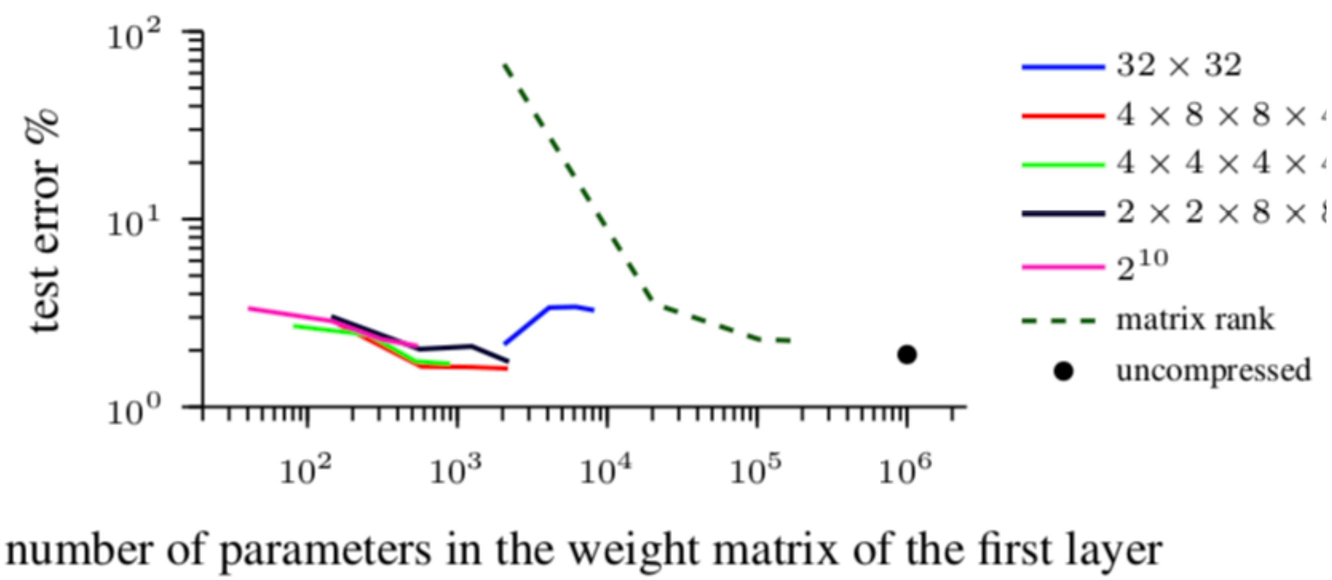


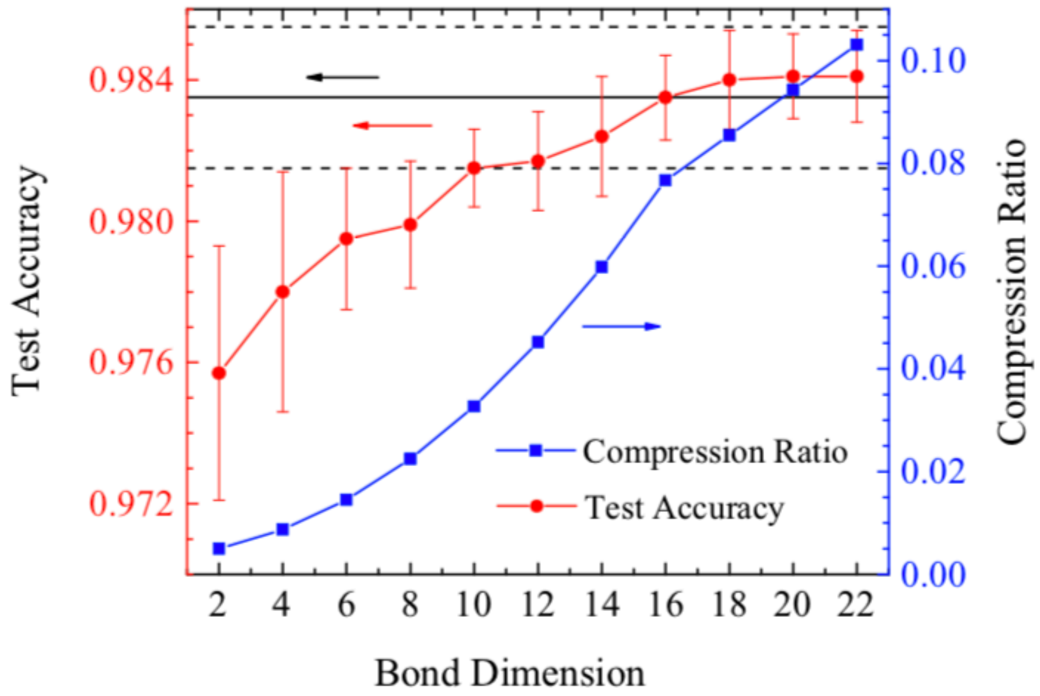
Novikov,Podoprikhin, Osokin, Vetrov, NIPS 2015

Gao et al, arXiv:1904.06194

Application: compressing neural networks

On 2-layer perceptron, MNIST





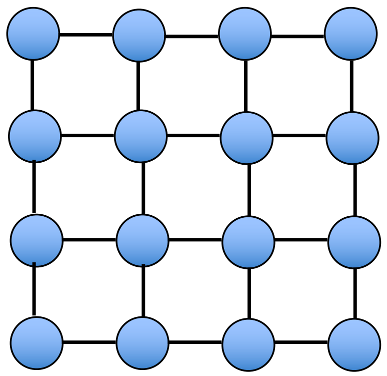
Novikov,Podoprikhin, Osokin, Vetrov, NIPS 2015

Gao et al, arXiv:1904.06194

Similar ideas have been applied to RBM as
“***Matrix product operator restricted Boltzmann Machine***” in
Chen, Batselier, Ko, Wong arXiv:1811.04608

TN contraction for computing the partition function

$\mathbf{S} = \{+1, -1\}^n$ $\quad \uparrow\uparrow\uparrow\downarrow\uparrow\downarrow\downarrow\downarrow\downarrow\downarrow\uparrow\uparrow$



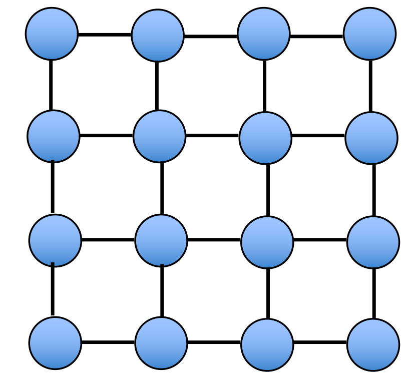
Computing normalization of a discrete probability distribution

$$P(\mathbf{S}) = \frac{1}{Z}\widetilde{P} = \frac{1}{Z}e^{-\beta E(\mathbf{S})}$$

TN contraction for computing the partition function

$\mathbf{S} = \{+1, -1\}^n$

$\uparrow\uparrow\uparrow\downarrow\uparrow\downarrow\downarrow\downarrow\downarrow\downarrow\uparrow\uparrow$



Computing normalization of a discrete probability distribution

$$P(\mathbf{S}) = \frac{1}{Z}\widetilde{P} = \frac{1}{Z}e^{-\beta E(\mathbf{S})}$$

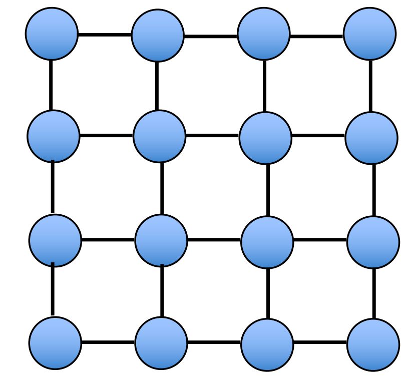
Any discrete probability distribution is a tensor,

decomposed using tensor networks.

TN contraction for computing the partition function

$\mathbf{S} = \{+1, -1\}^n$

$\uparrow\uparrow\uparrow\downarrow\uparrow\downarrow\downarrow\downarrow\downarrow\downarrow\uparrow\uparrow$



Computing normalization of a discrete probability distribution

$$P(\mathbf{S}) = \frac{1}{Z}\widetilde{P} = \frac{1}{Z}e^{-\beta E(\mathbf{S})}$$

Any discrete probability distribution is a tensor,

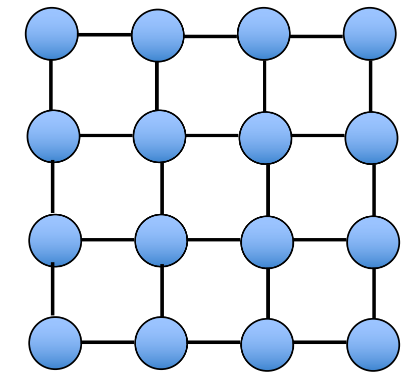
decomposed using tensor networks.

$$Z = \left\|\widetilde{P}\right\|_1 = \widetilde{P} \cdot \mathbf{1}_{2^n}^\top = \widetilde{P} \cdot \underbrace{\begin{pmatrix} 1 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 1 \end{pmatrix} \otimes \cdots \otimes \begin{pmatrix} 1 \\ 1 \end{pmatrix}}_{n},$$

TN contraction for computing the partition function

$\mathbf{S} = \{+1, -1\}^n$

↑ ↑ ↑ ↓ ↑ ↓ ↓ ↓ ↓ ↓ ↑ ↑



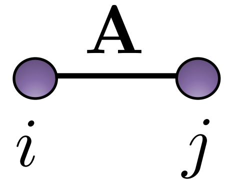
Computing normalization of a discrete probability distribution

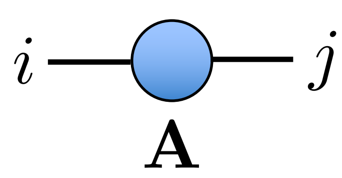
$$P(\mathbf{S}) = \frac{1}{Z}\widetilde{P} = \frac{1}{Z}e^{-\beta E(\mathbf{S})}$$

Any discrete probability distribution is a tensor,

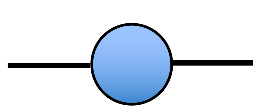
decomposed using tensor networks.

$$Z = \left\|\widetilde{P}\right\|_1 = \widetilde{P} \cdot \mathbf{1}_{2^n}^{\top} = \widetilde{P} \cdot \underbrace{\begin{pmatrix} 1 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 1 \end{pmatrix} \otimes \cdots \otimes \begin{pmatrix} 1 \\ 1 \end{pmatrix}}_{n},$$



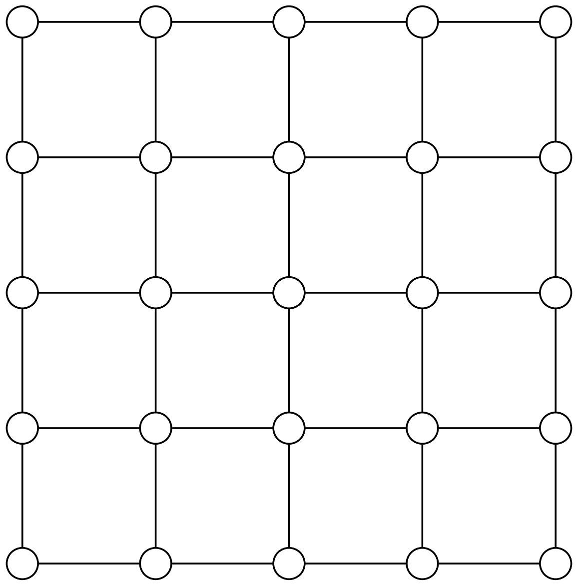


$$Z = \sum_{i,j} A_{ij} = \mathbf{1}^T A \mathbf{1} = [1, 1]$$



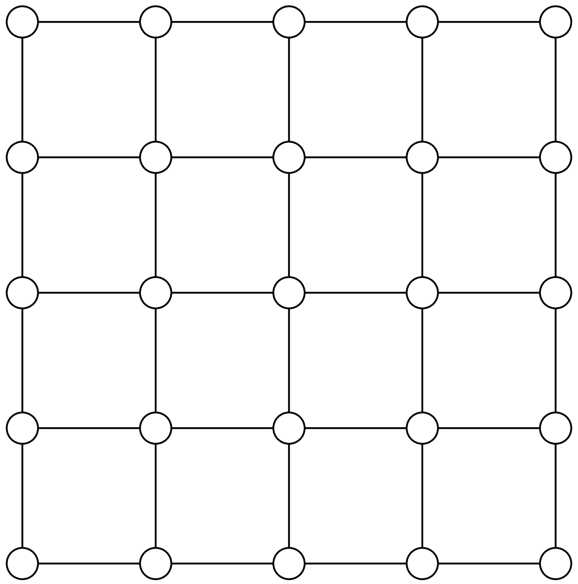
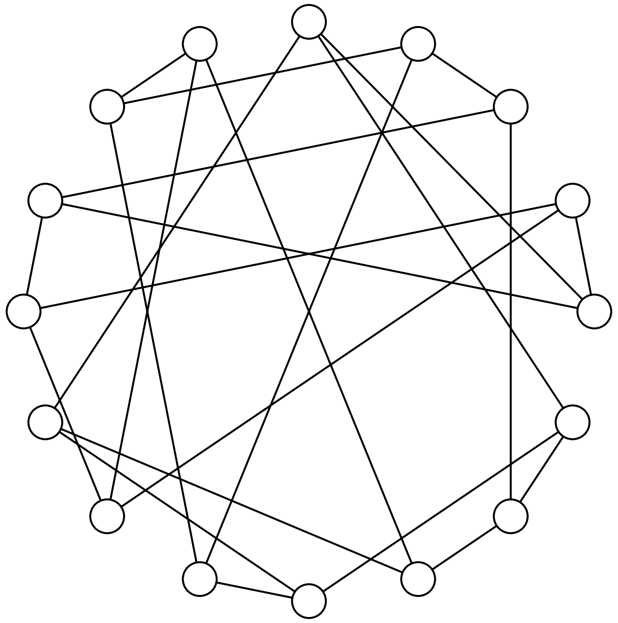
$$\begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

TN Contraction $\longrightarrow$ $Z(\beta)$



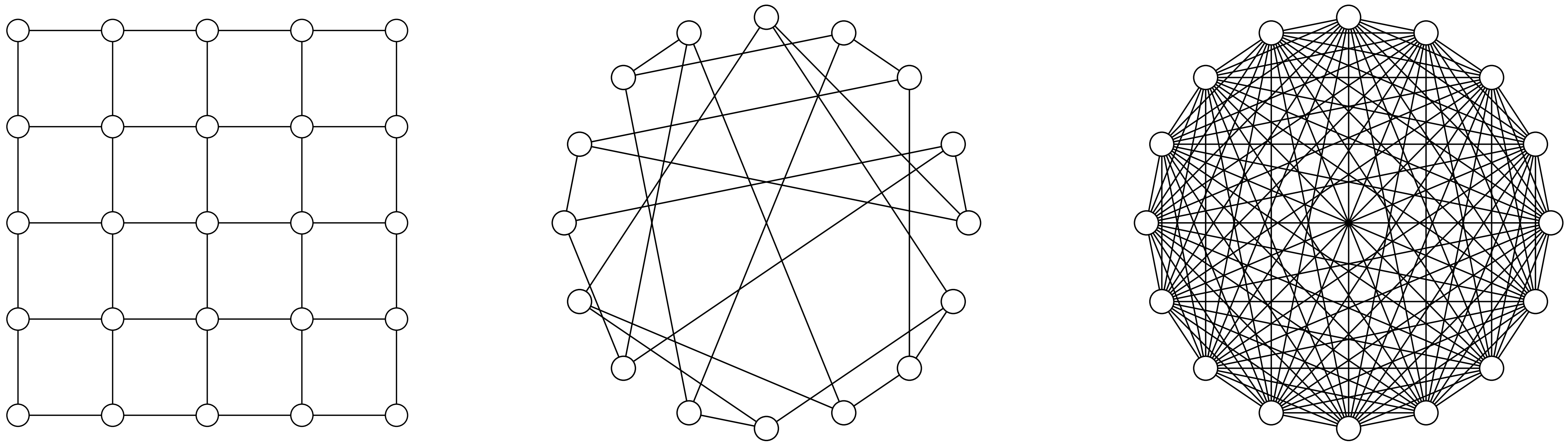
S. Li, P. Zhou, F. Pan, PZ, arXiv:2105.04130 (2021)

TN Contraction $\longrightarrow$ $Z(\beta)$

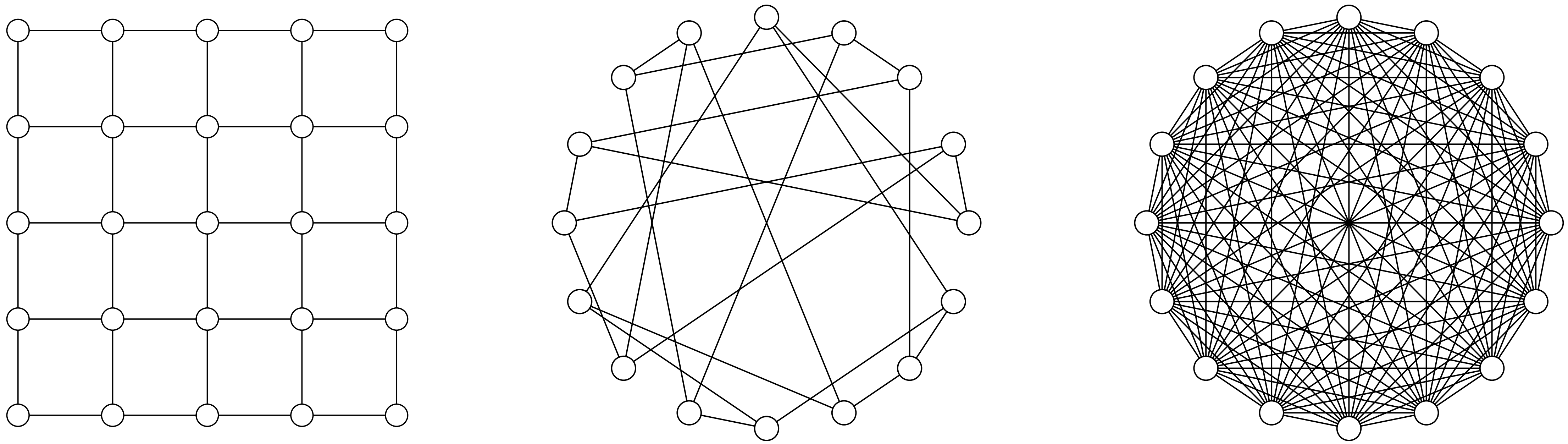
S. Li, P. Zhou, F. Pan, PZ, arXiv:2105.04130 (2021)

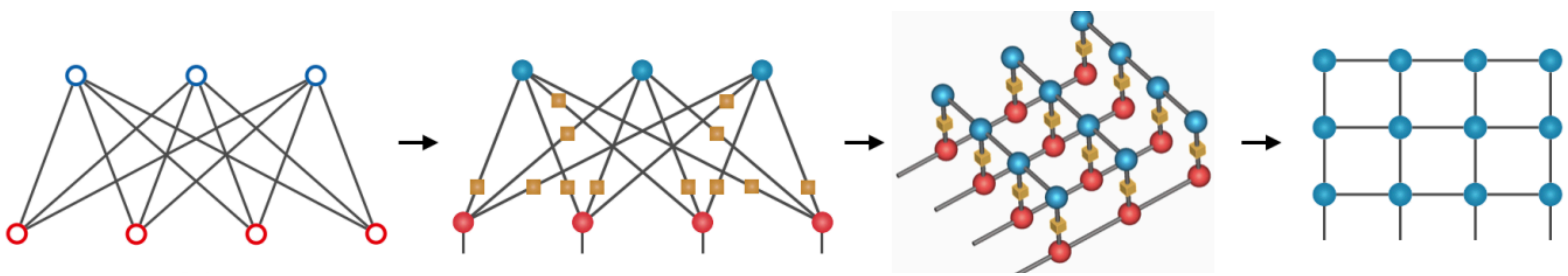
TN Contraction $\longrightarrow$ $Z(\beta)$



S. Li, P. Zhou, F. Pan, PZ, arXiv:2105.04130 (2021)

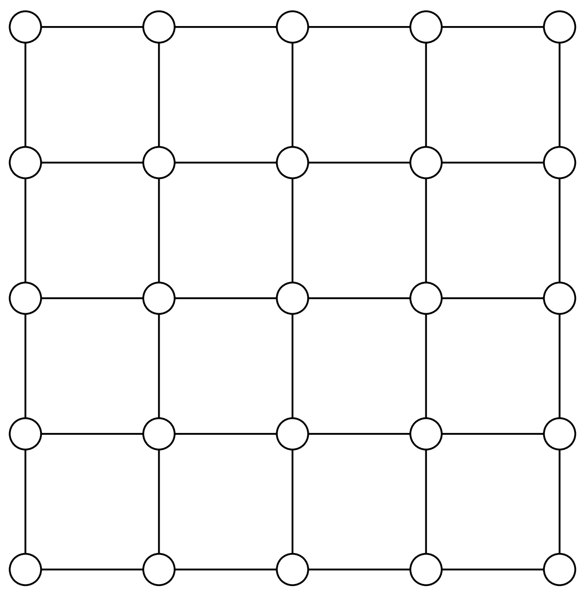
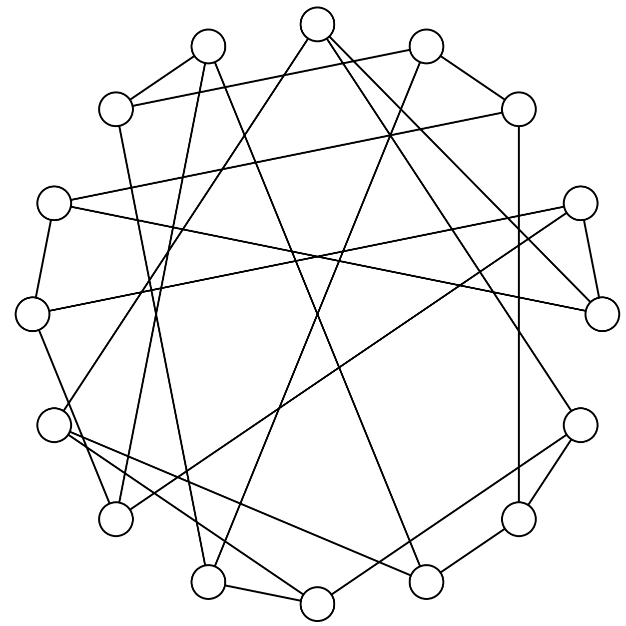
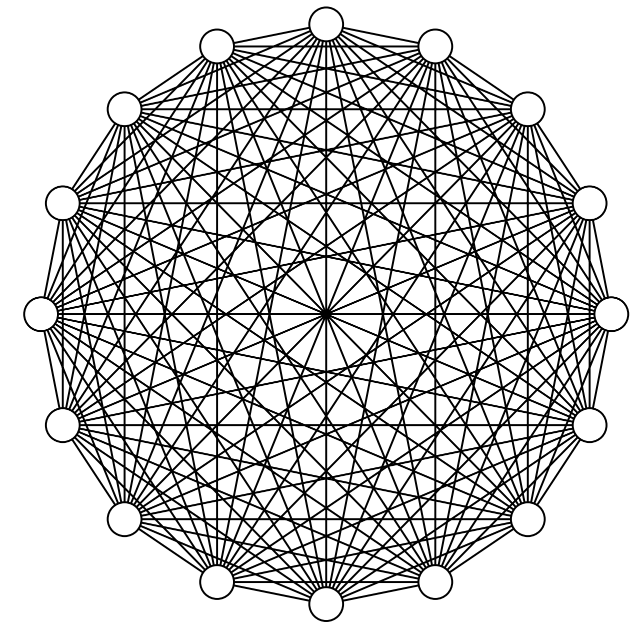
TN Contraction $\longrightarrow$ $Z(\beta)$

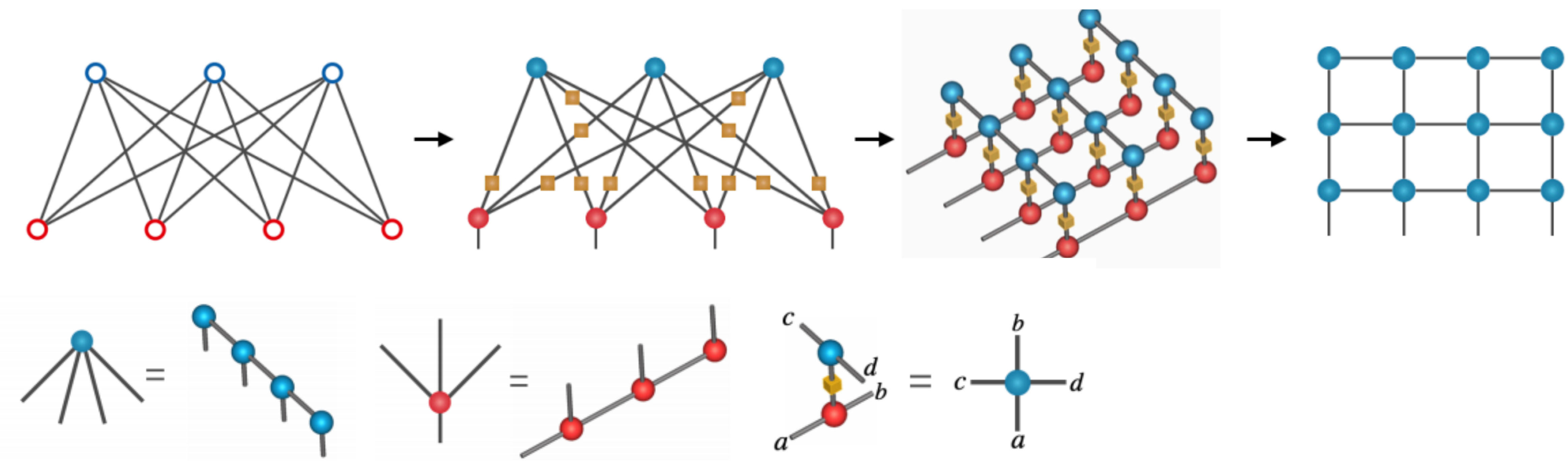




S. Li, P. Zhou, F. Pan, PZ, arXiv:2105.04130 (2021)

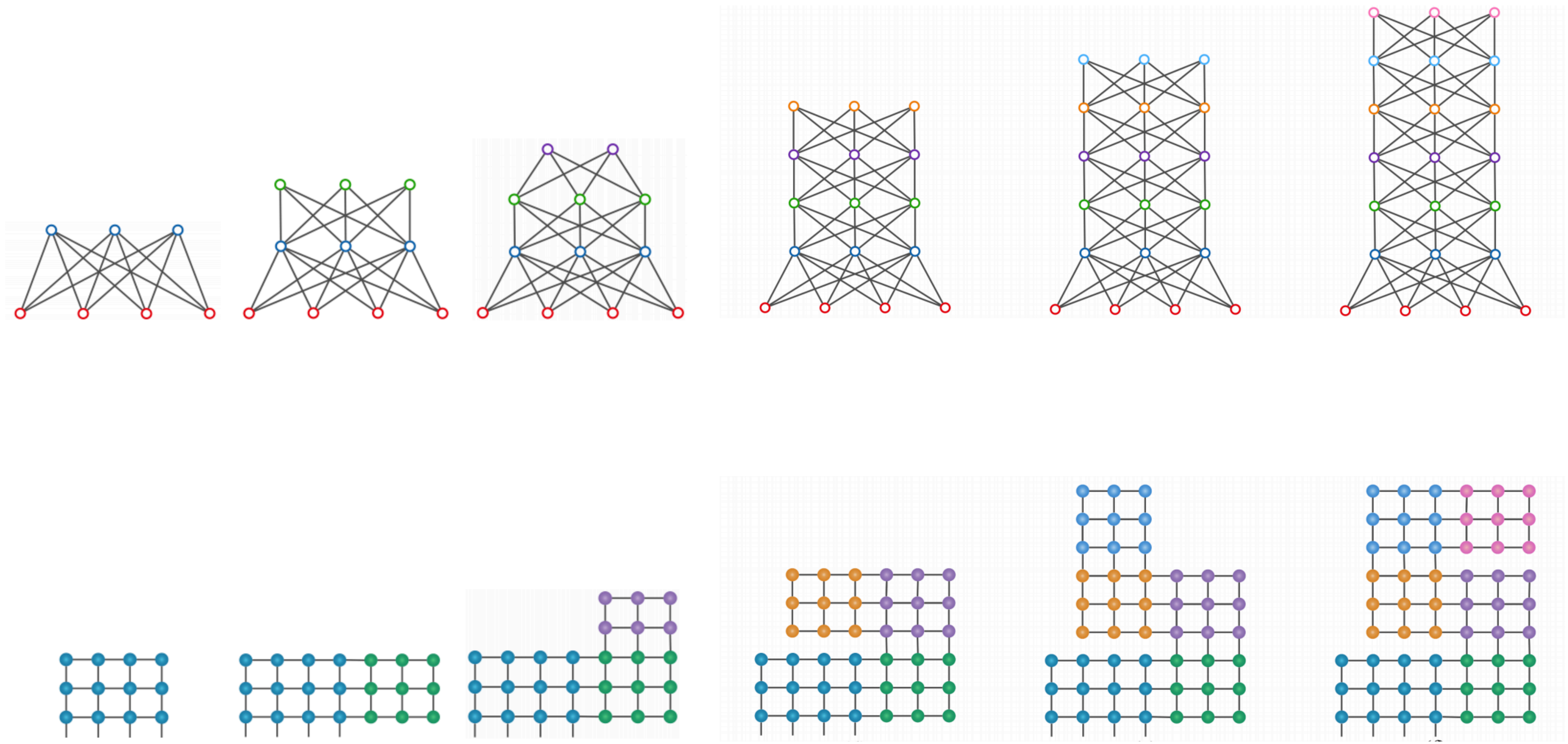
TN Contraction $\longrightarrow$ $Z(\beta)$



S. Li, P. Zhou, F. Pan, PZ, arXiv:2105.04130 (2021)

TN Contraction $\longrightarrow$ $Z(\beta)$

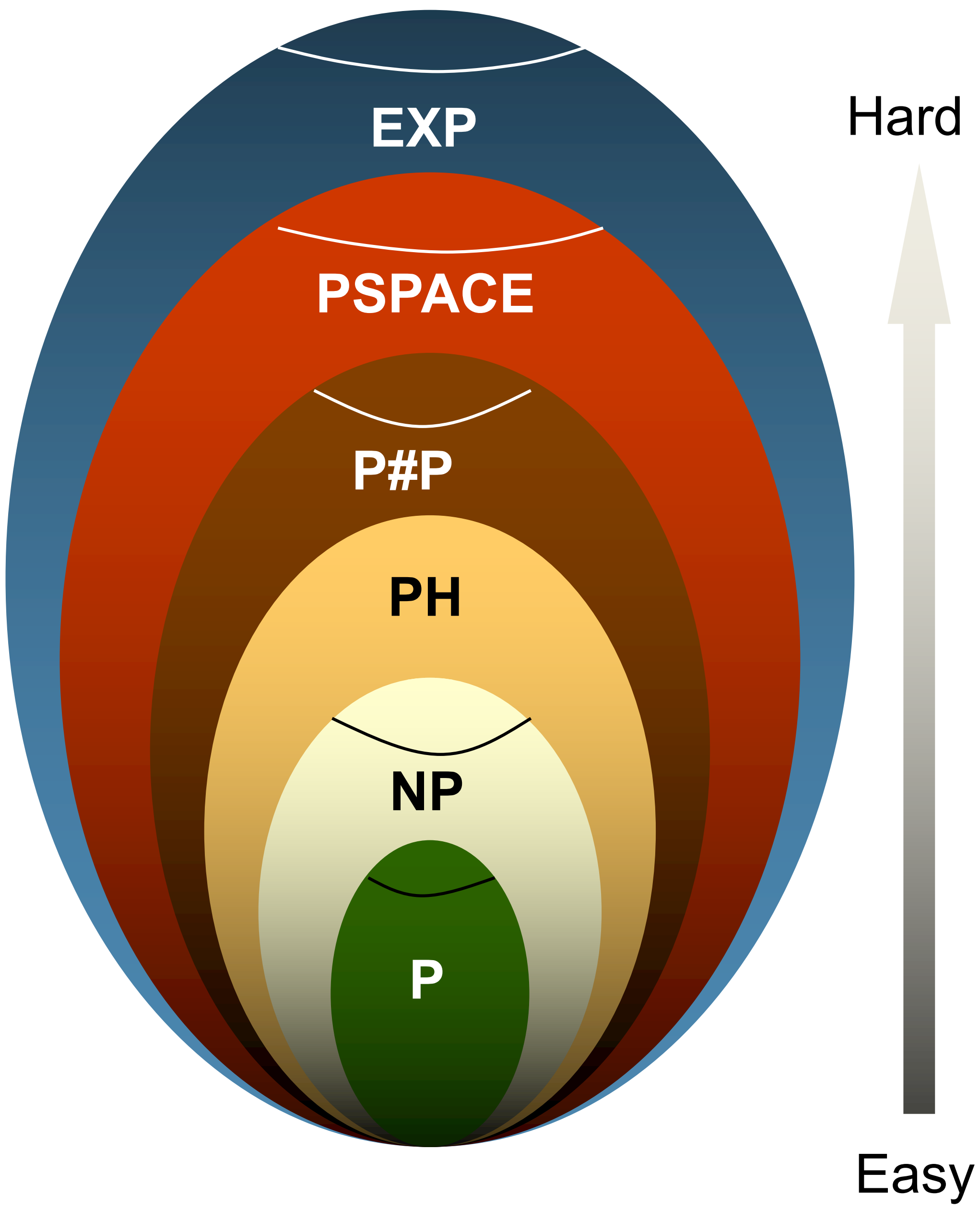


Deep Boltzmann Machines **2D Tensor Network**

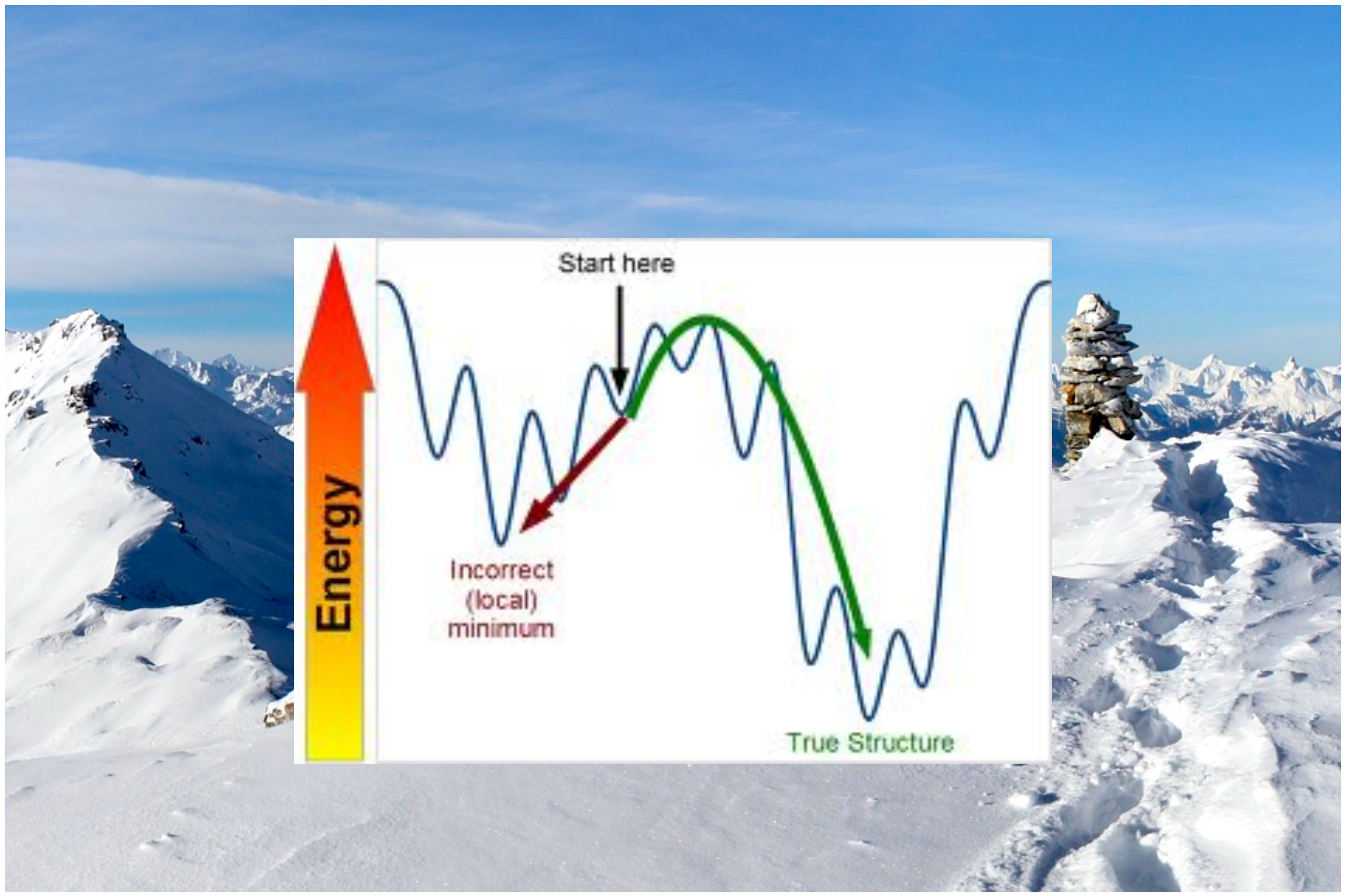
S. Li, P. Zhou, F. Pan, PZ, arXiv:2105.04130 (2021)

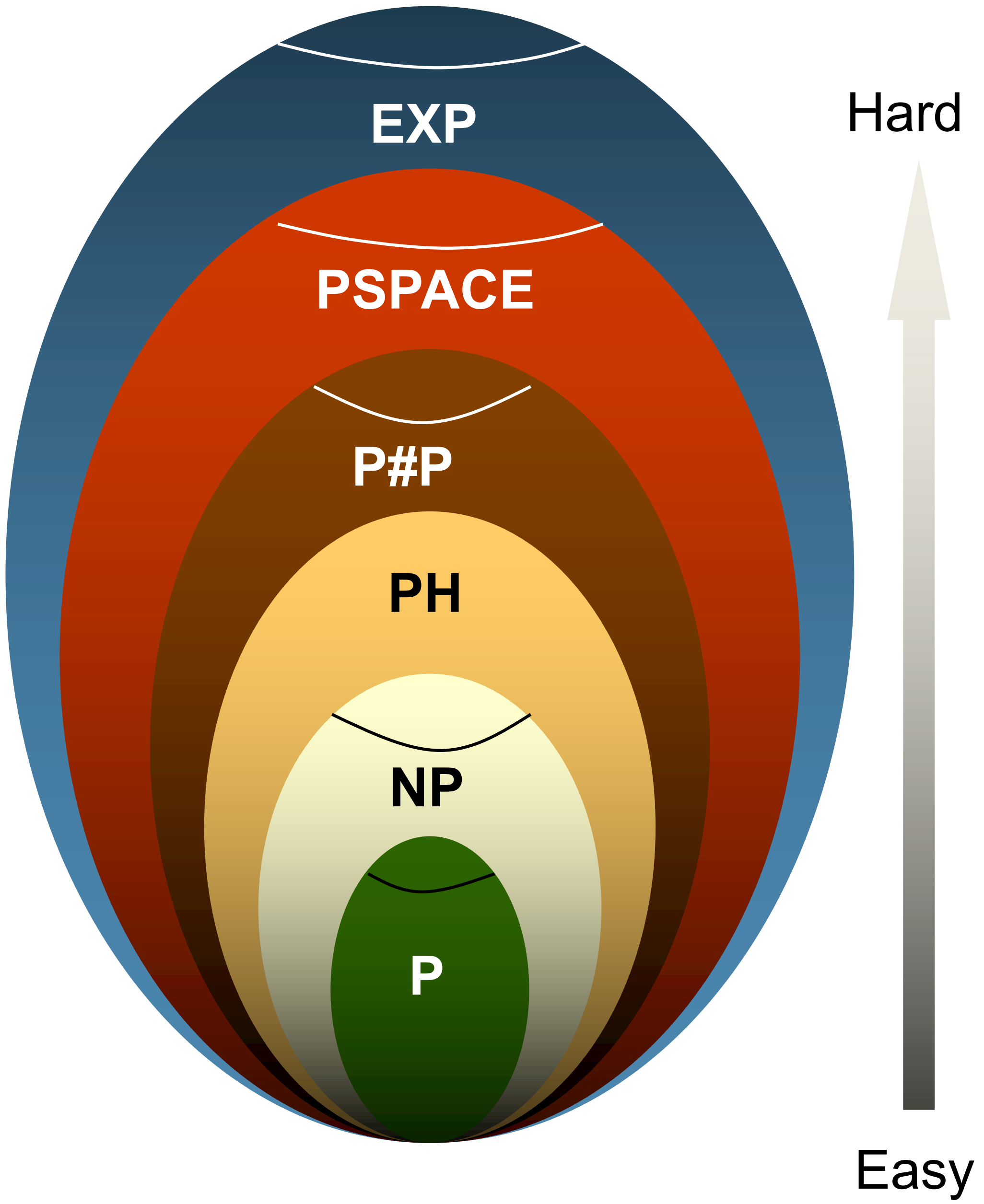
Towards $T \to 0$: Ground States, combinatorial optimization

Towards $T \to 0$: Ground States, combinatorial optimization

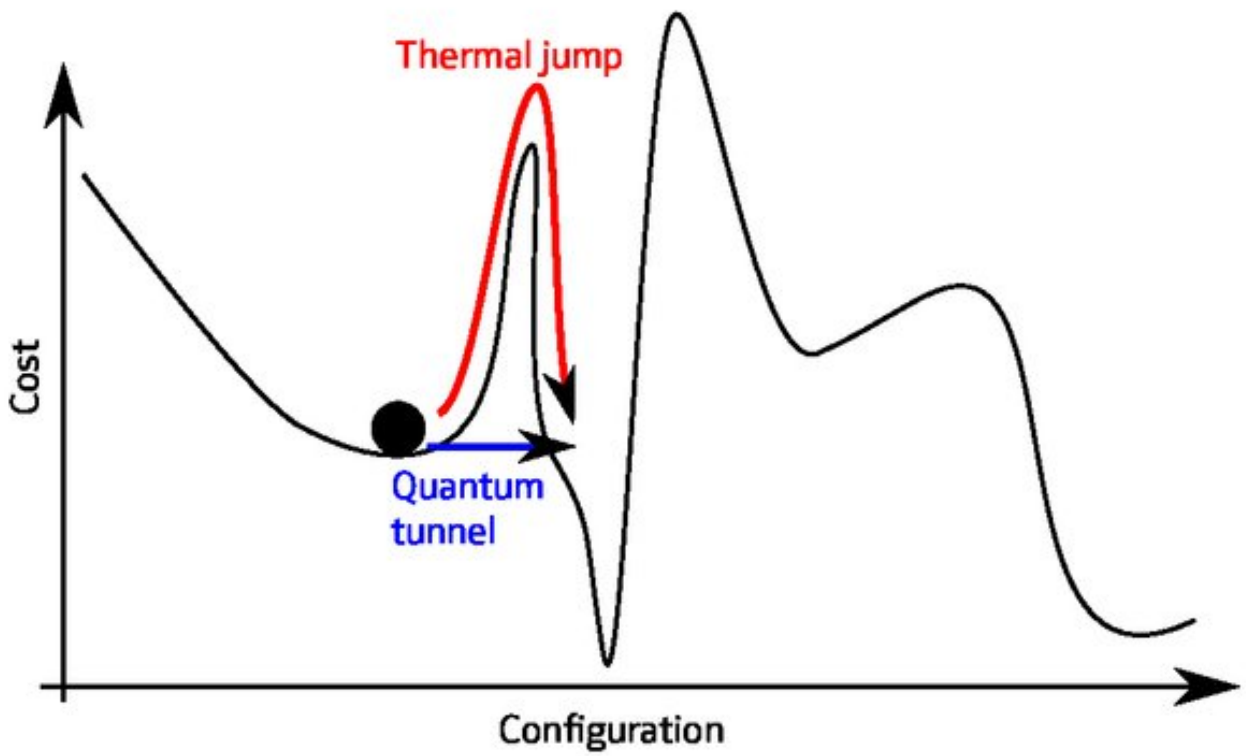


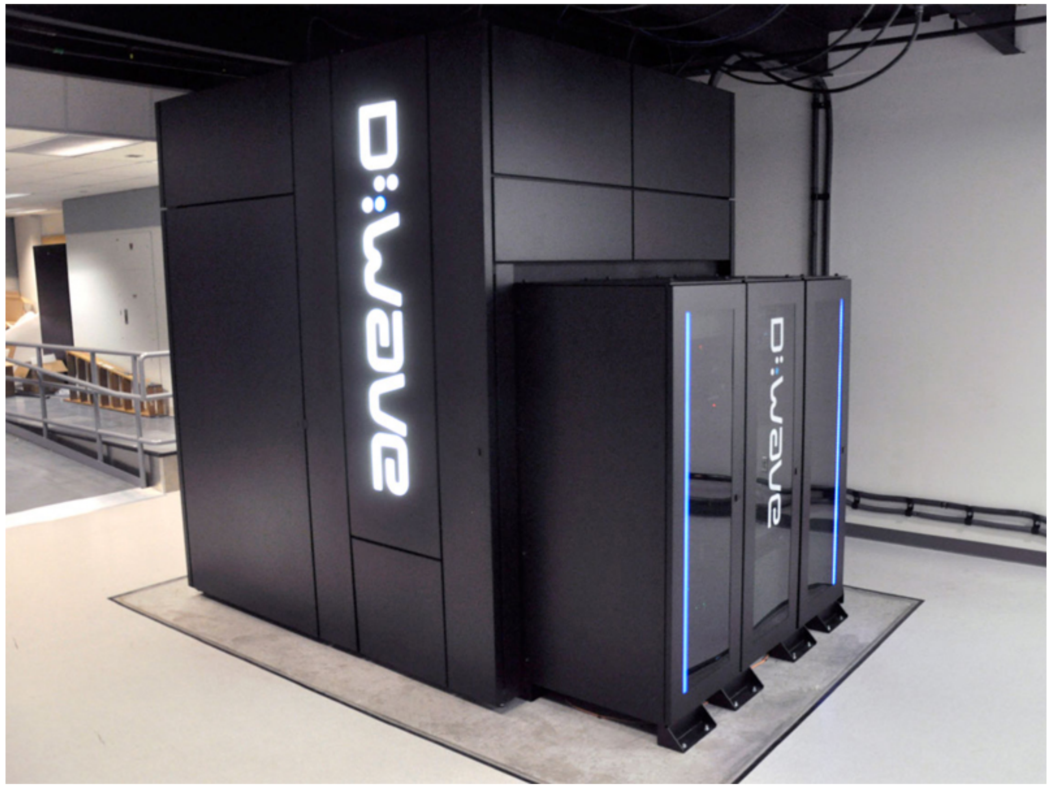
Towards $T \to 0$: Ground States, combinatorial optimization

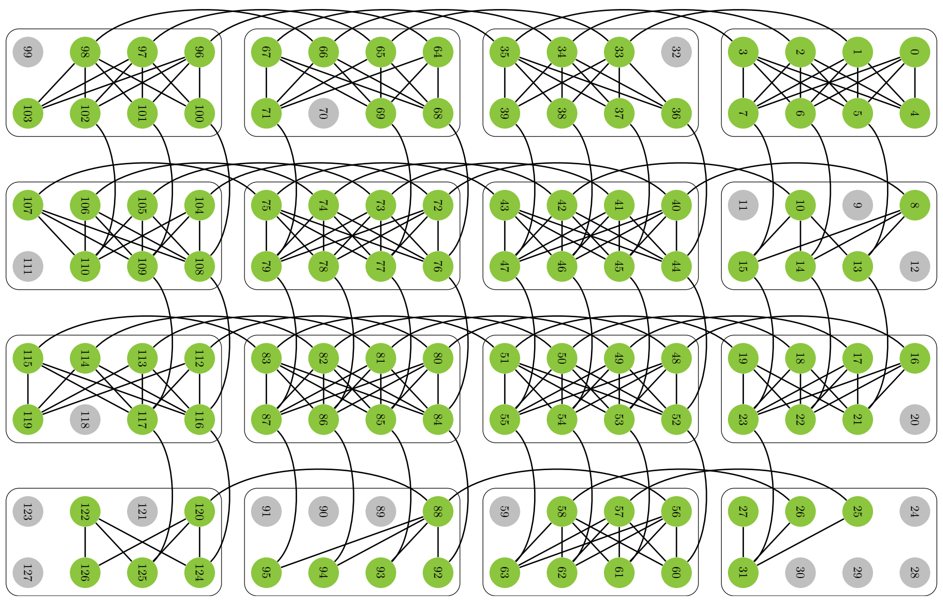




Simulated and quantum annealing







Slides Courtesy: Lei Wang

Tropical Tensor Network

Replace $(\times, +)$ operations in linear algebra with $(\otimes, \oplus)$

$$x \otimes y = x + y \qquad x \oplus y = \min(x, y)$$

$$x \oplus y = \min(x, y) \qquad x \odot y = x + y$$

Addition table

$\oplus$	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	1	1	1	1	1	1	1
2	1	2	2	2	2	2	2
3	1	2	3	3	3	3	3
4	1	2	3	4	4	4	4
5	1	2	3	4	5	5	5
6	1	2	3	4	5	6	6
7	1	2	3	4	5	6	7

Multiplication table

$\odot$	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	2	3	4	5	6	7	8
2	3	4	5	6	7	8	9
3	4	5	6	7	8	9	10
4	5	6	7	8	9	10	11
5	6	7	8	9	10	11	12
6	7	8	9	10	11	12	13
7	8	9	10	11	12	13	14

Tropical Tensor Network

Replace $(\times, +)$ **operations in linear algebra with** $(\otimes, \oplus)$

$$x \otimes y = x + y \qquad x \oplus y = \min(x, y)$$

J. Liu, L. Wang, PZ, PRL 126, 090506 (2021)

Tropical Tensor Network

Replace $(\times, +)$ **operations in linear algebra with** $(\otimes, \oplus)$

$$x \otimes y = x + y \qquad x \oplus y = \min(x, y)$$

$$\begin{aligned} F_0 &= \lim_{\beta\to\infty} -\frac{1}{\beta} \ln \sum_{\mathbf{s}} e^{-\beta E(\mathbf{s})} \\ &= \lim_{\beta\to\infty} \left[E_{\min} - \frac{1}{\beta} S(E_{\min}) \right] \end{aligned}$$

J. Liu, L. Wang, PZ, PRL 126, 090506 (2021)

Tropical Tensor Network

Replace $(\times, +)$ **operations in linear algebra with** $(\otimes, \oplus)$

$$x \otimes y = x + y \qquad\qquad x \oplus y = \min(x, y)$$

$$F_0 = \lim_{\beta\to\infty} -\frac{1}{\beta} \ln \sum_{\mathbf{s}} e^{-\beta E(\mathbf{s})}$$

$$= \lim_{\beta\to\infty} \left[E_{\min} - \frac{1}{\beta} S(E_{\min}) \right]$$

$$\lim_{\beta\to\infty} -\frac{1}{\beta} \ln(e^{-\beta x} \times e^{-\beta y}) = x \otimes y$$

$$\lim_{\beta\to\infty} -\frac{1}{\beta} \ln(e^{-\beta x} + e^{-\beta y}) = x \oplus y$$

J. Liu, L. Wang, PZ, PRL 126, 090506 (2021)

Tropical Tensor Network

Replace $(\times, +)$ **operations in linear algebra with** $(\otimes, \oplus)$

$$x \otimes y = x + y \qquad\qquad x \oplus y = \min(x, y)$$

$$F_0 = \lim_{\beta\to\infty} -\frac{1}{\beta} \ln \sum_{\mathbf{s}} e^{-\beta E(\mathbf{s})}$$
$$= \lim_{\beta\to\infty} \left[E_{\min} - \frac{1}{\beta} S(E_{\min})\right]$$

$$\lim_{\beta\to\infty} -\frac{1}{\beta} \ln(e^{-\beta x} \times e^{-\beta y}) = x \otimes y$$
$$\lim_{\beta\to\infty} -\frac{1}{\beta} \ln(e^{-\beta x} + e^{-\beta y}) = x \oplus y$$

ground state energy
$\longrightarrow$ Tropical tensor network contraction

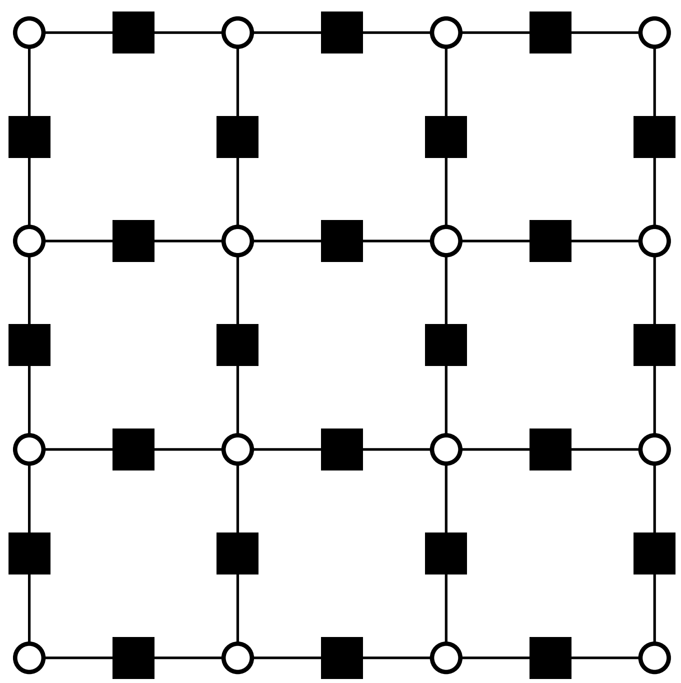
ground state configuration
$\longrightarrow$ Gradient with respect to the field

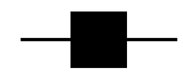
ground state degeneracy
$\longrightarrow$ Mix tropical with ordinary algebra

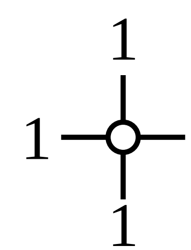
J. Liu, L. Wang, PZ, PRL 126, 090506 (2021)

Tropical tensor networks for the Ising spin glasses

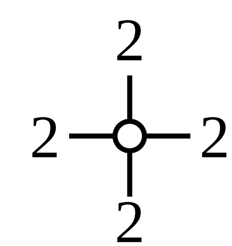
$$E(\{\sigma\}) = \sum_{i<j} J_{ij}\sigma_i\sigma_j + \sum_i h_i\sigma_i$$

$E^* =$ 

 $= \begin{pmatrix} J_{ij} & -J_{ij} \\ -J_{ij} & J_{ij} \end{pmatrix}$



$= h_i$

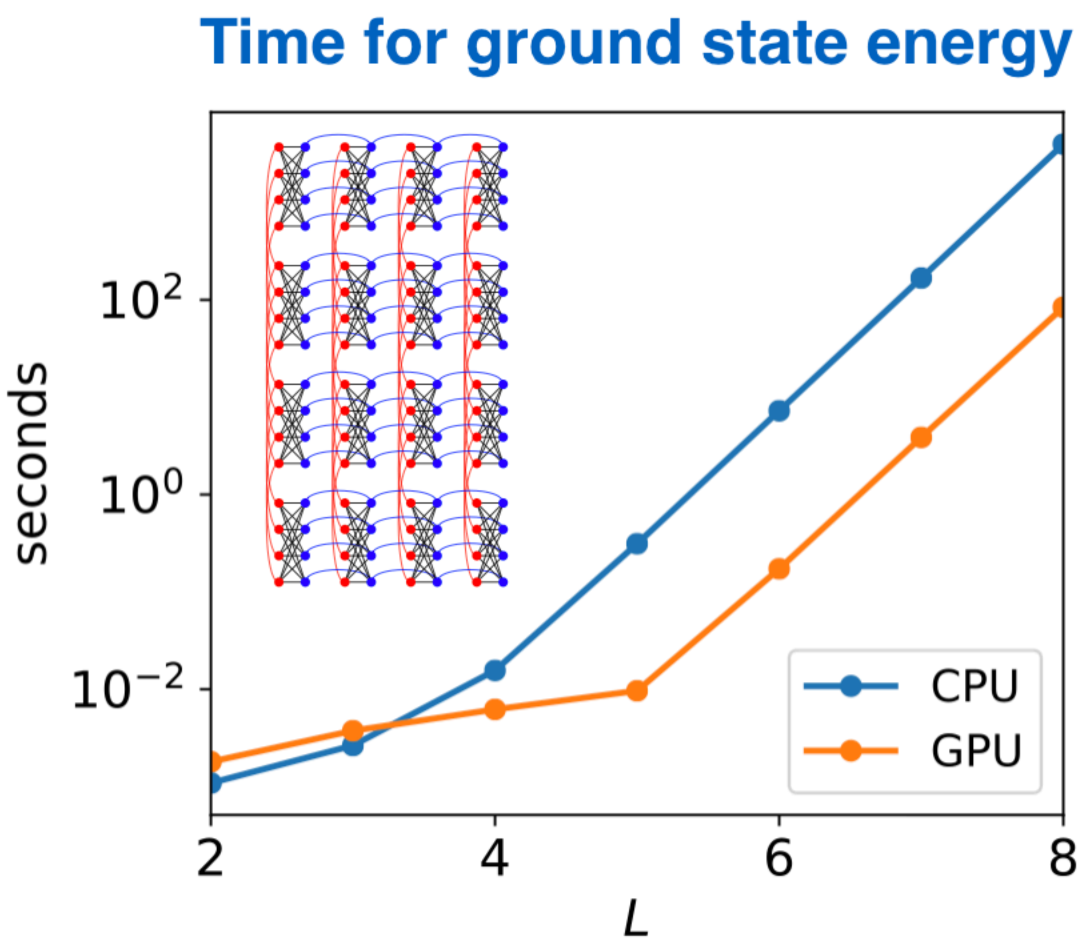


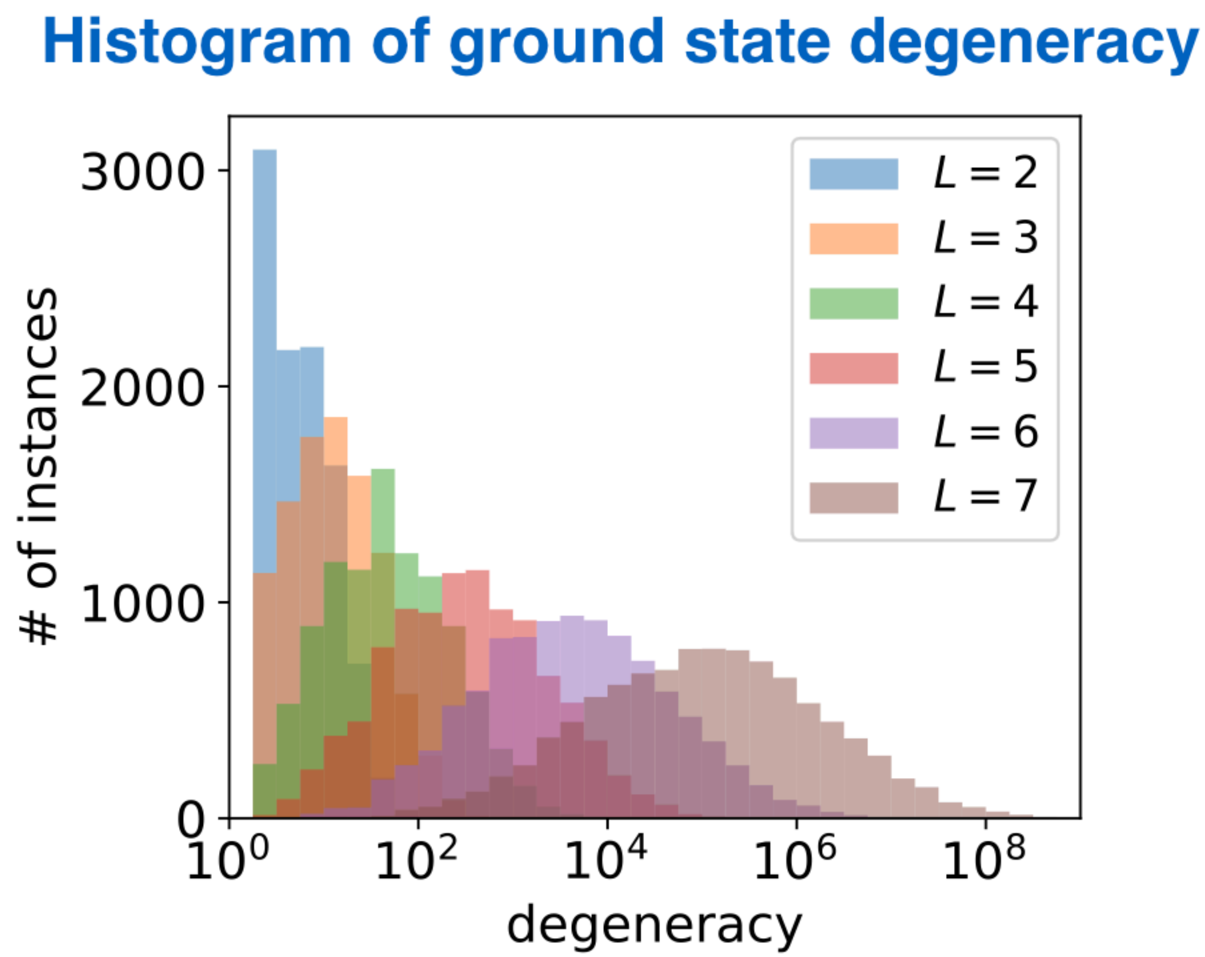
$= -h_i$

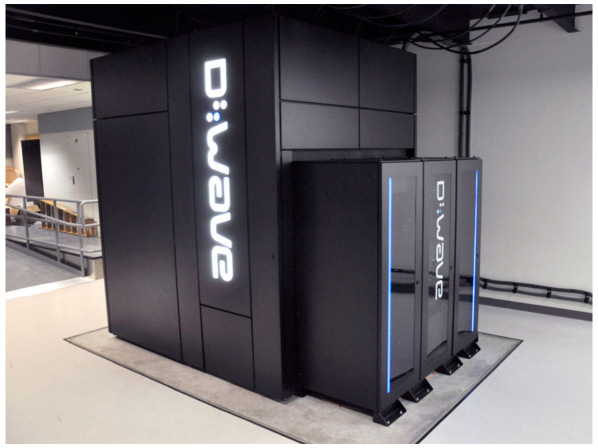
all other elements are ∞

Slides Courtesy: Lei Wang

Results on Spin glasses on the Chimera graphs





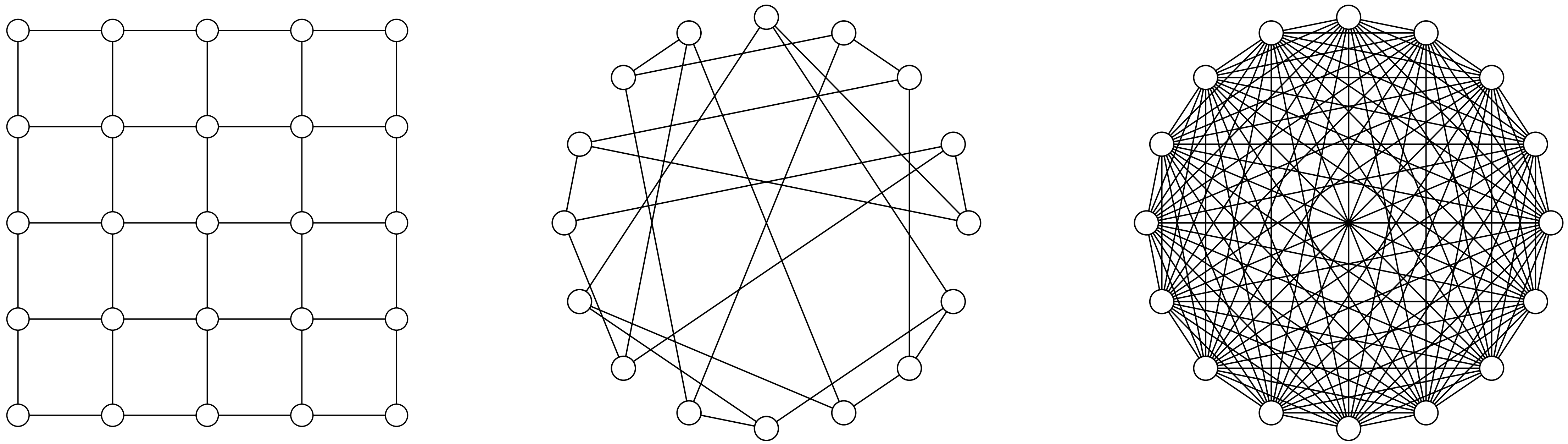


J. Liu, L. Wang, PZ, PRL 126, 090506 (2021)

Exact ground-state energy and degeneracies

	Branch-and-bound / cut	**Our method**
Ising Gaussian 2D lattice	100x100	32x32
Ising pm J 2D lattice GS Energy	50x50	32x32
Ising pm J 2D lattice GS Entropy	8x8	32x32
Ising pm J 3D lattice GS Energy	4x4x4	6x6x6
Ising pm J 3D lattice GS Entropy	*	6x6x6
3-state-Potts 2D Lattice GS Energy	9x9	18x18
3-state-Potts 2D Lattice GS Energy	*	18x18

J. Liu, L. Wang, PZ, *PRL* 126, 090506 (2021)



For large TNs: we need approximate contractions

Challenges:

1. Exponential space complexity

---

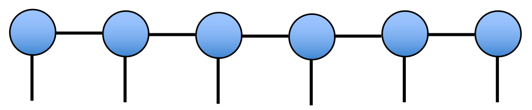
2. Global approximations

---

3. Scalability

F. Pan, P. Zhou, S. Li , PZ, *PRL* 125,060503 (2020)

Challenges:

1. Exponential space complexity **Compressed using MPS** 

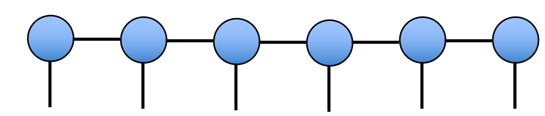
2. Global approximations

3. Scalability

F. Pan, P. Zhou, S. Li , PZ, *PRL* 125,060503 (2020)

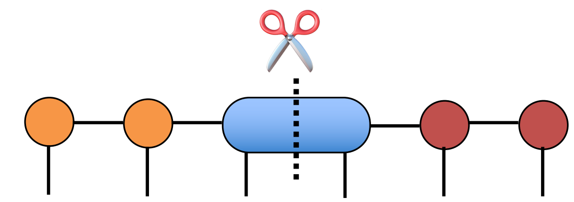
Challenges:

1. Exponential space complexity **Compressed using MPS**



---

2. Global approximations **Canonical form, DMRG**



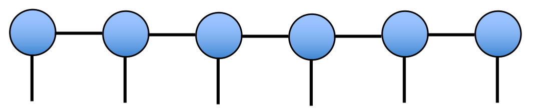
---

3. Scalability

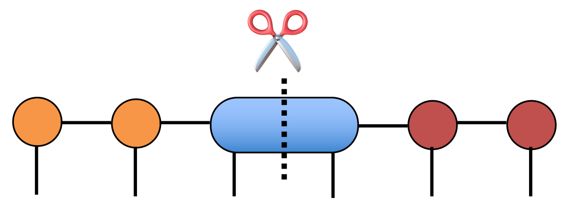
F. Pan, P. Zhou, S. Li , PZ, *PRL* 125,060503 (2020)

Challenges:

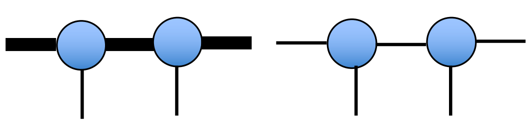
1. Exponential space complexity — **Compressed using MPS**



2. Global approximations — **Canonical form, DMRG**



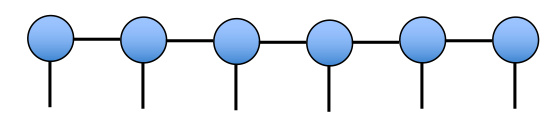
3. Scalability — **Tunable bond dimension**



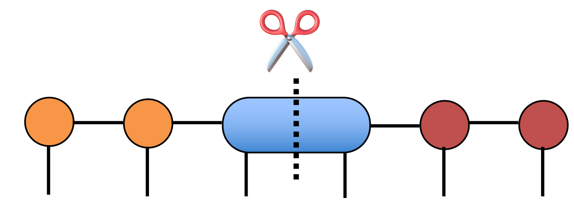
F. Pan, P. Zhou, S. Li , PZ, *PRL* 125,060503 (2020)

Challenges:

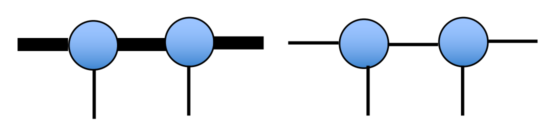
1. Exponential space complexity **Compressed using MPS**



2. Global approximations **Canonical form, DMRG**



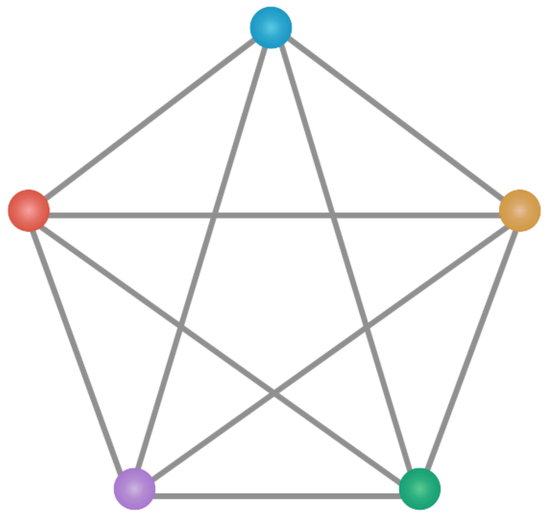
3. Scalability **Tunable bond dimension**



→ **CATN**

F. Pan, P. Zhou, S. Li , PZ, *PRL* 125,060503 (2020)

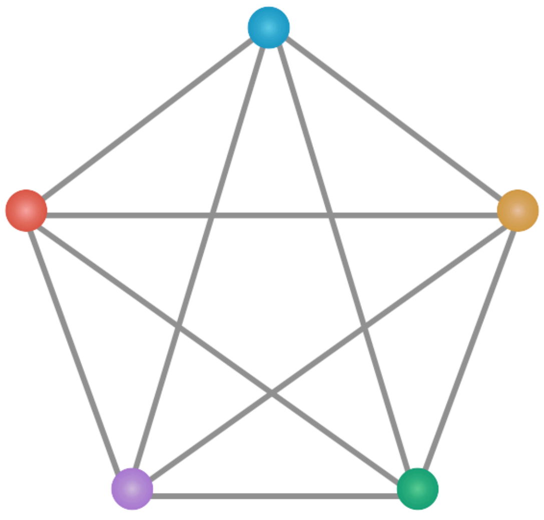
CATN: Contracting Arbitrary Tensor Networks



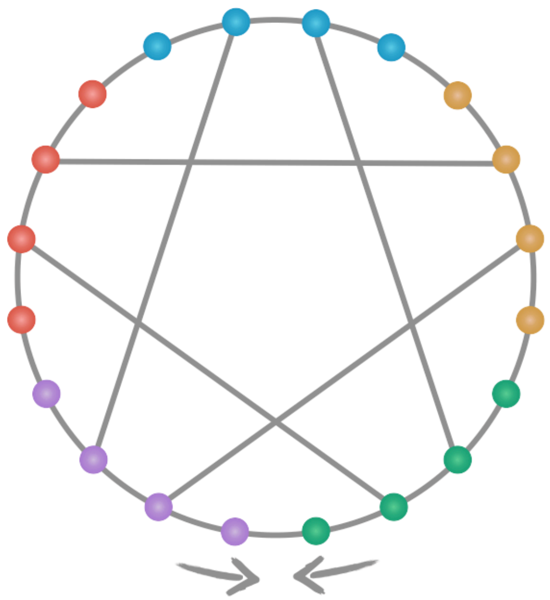
(1)

F. Pan, P. Zhou, S. Li , PZ, *PRL* 125,060503 (2020)

CATN: Contracting Arbitrary Tensor Networks



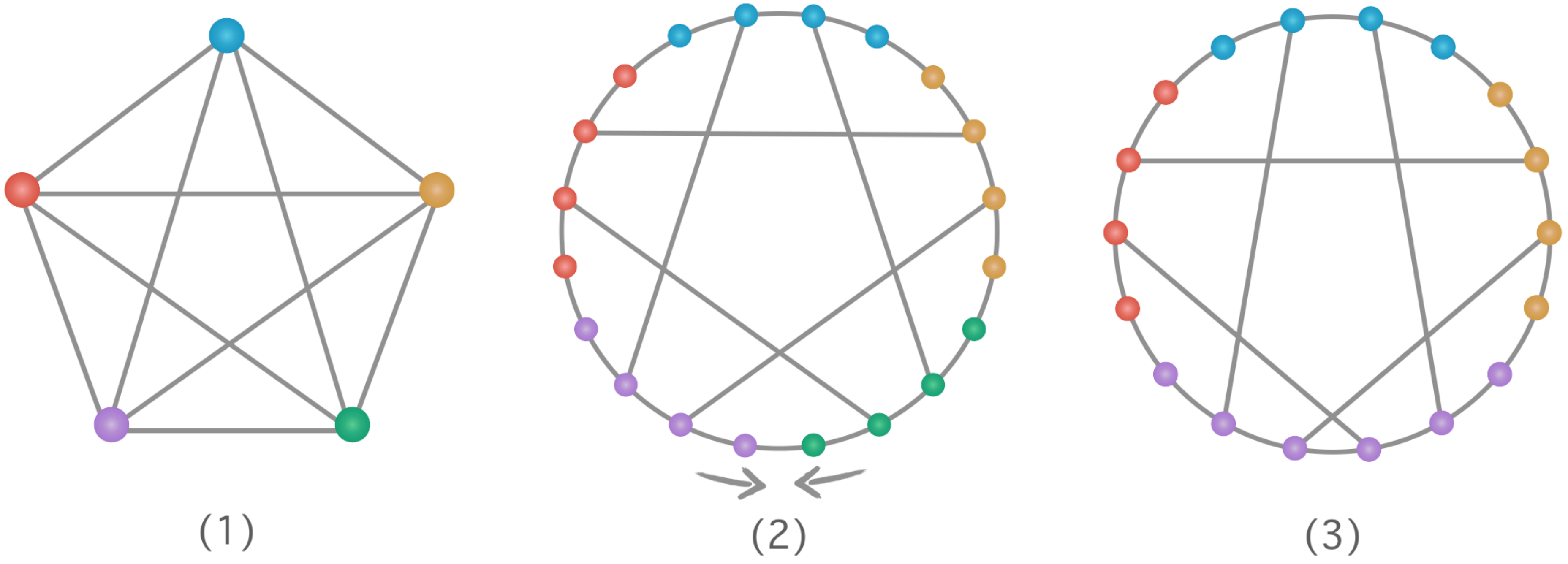
(1)



(2)

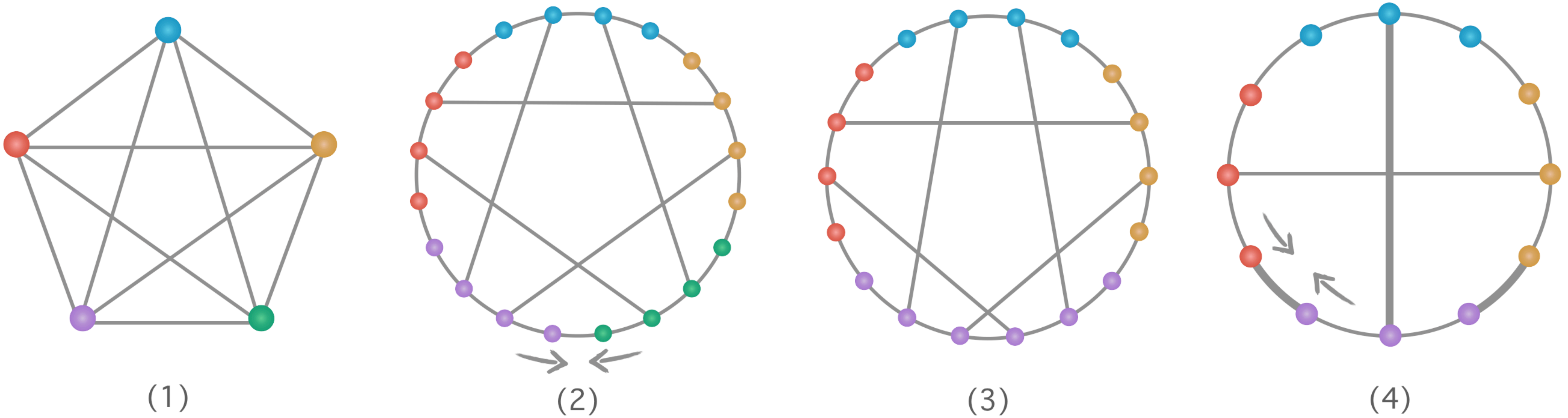
F. Pan, P. Zhou, S. Li , PZ, *PRL* 125,060503 (2020)

CATN: Contracting Arbitrary Tensor Networks



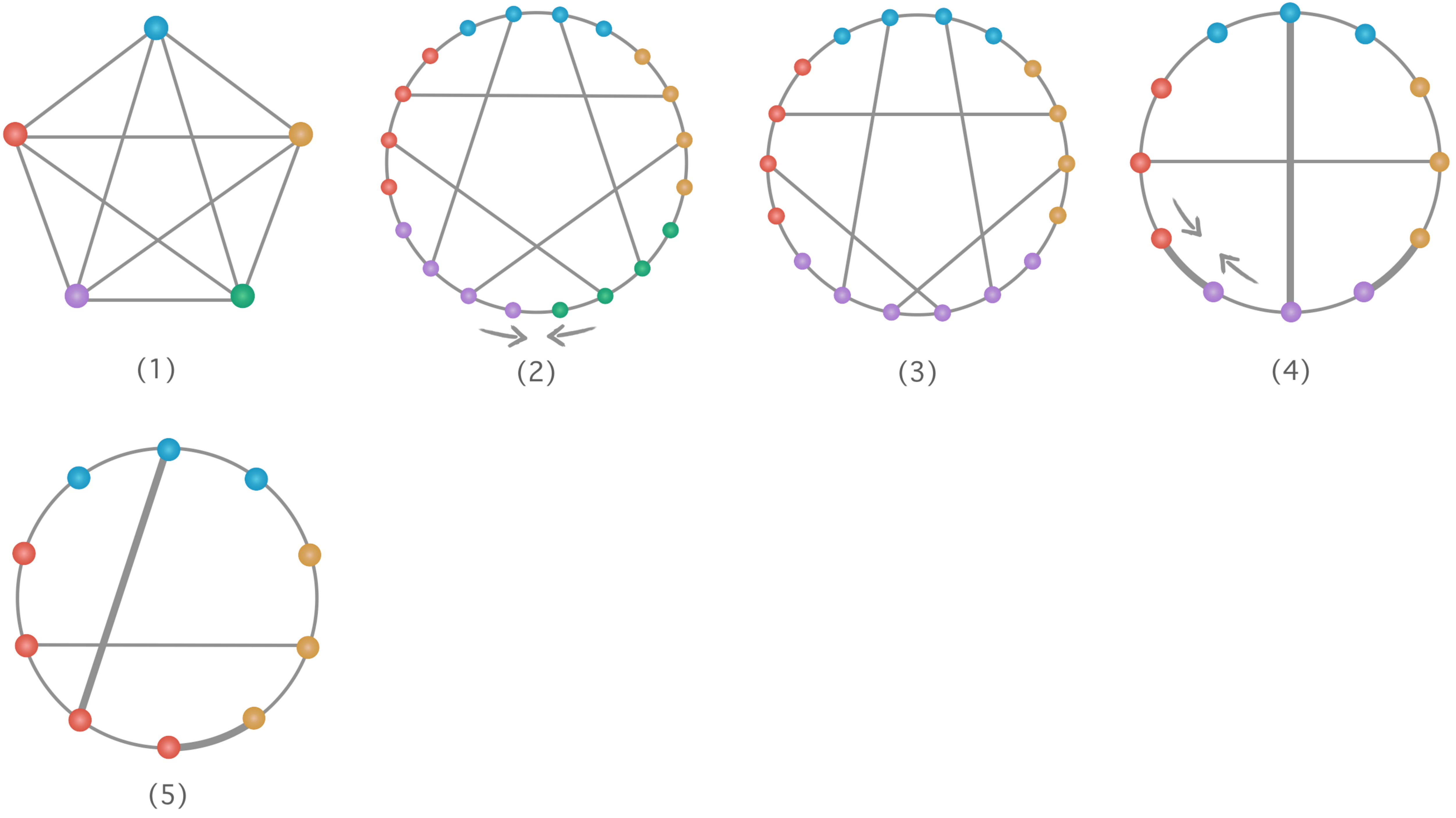
F. Pan, P. Zhou, S. Li , PZ, *PRL* 125,060503 (2020)

CATN: Contracting Arbitrary Tensor Networks



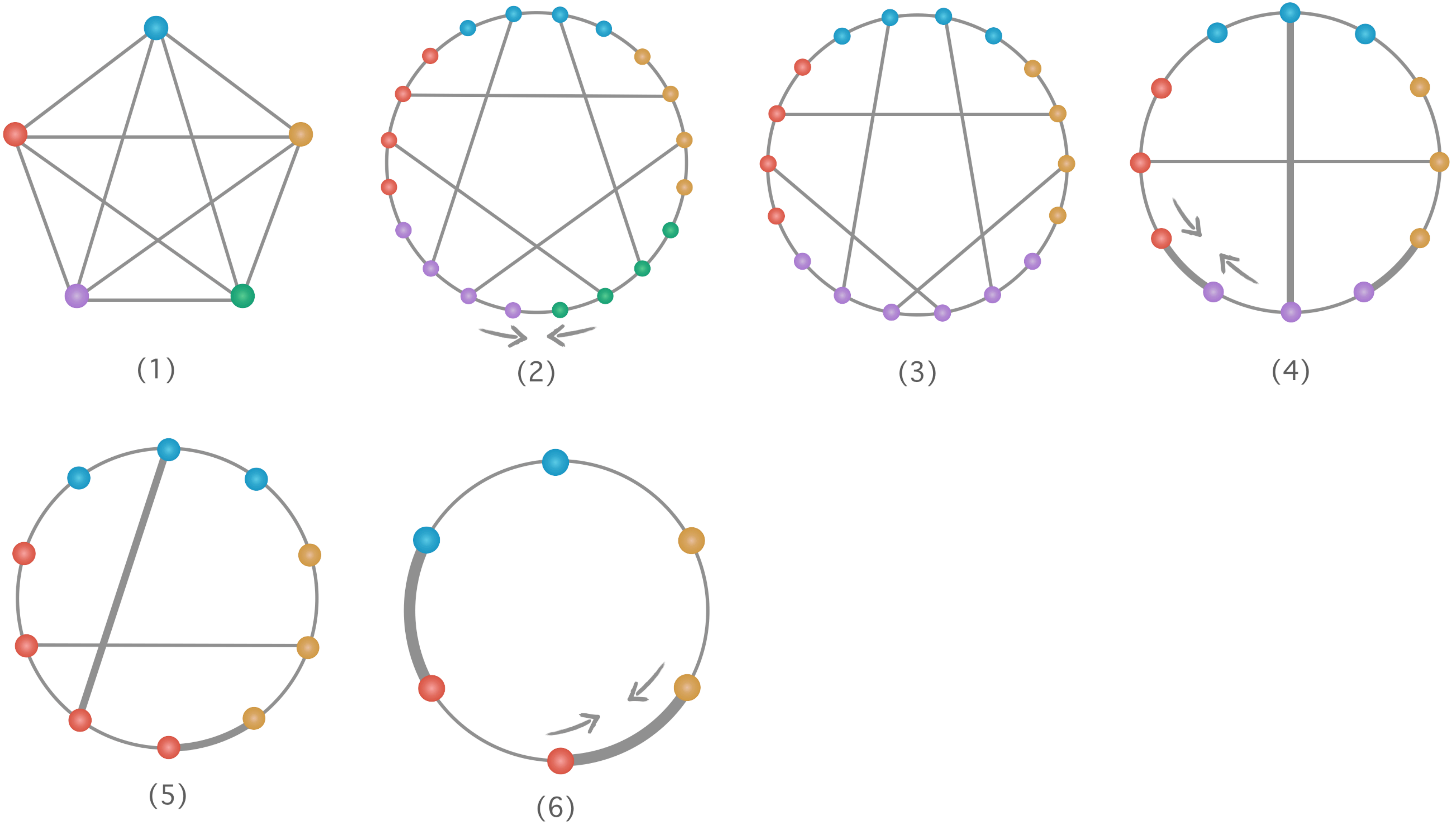
F. Pan, P. Zhou, S. Li , PZ, *PRL* 125,060503 (2020)

CATN: Contracting Arbitrary Tensor Networks



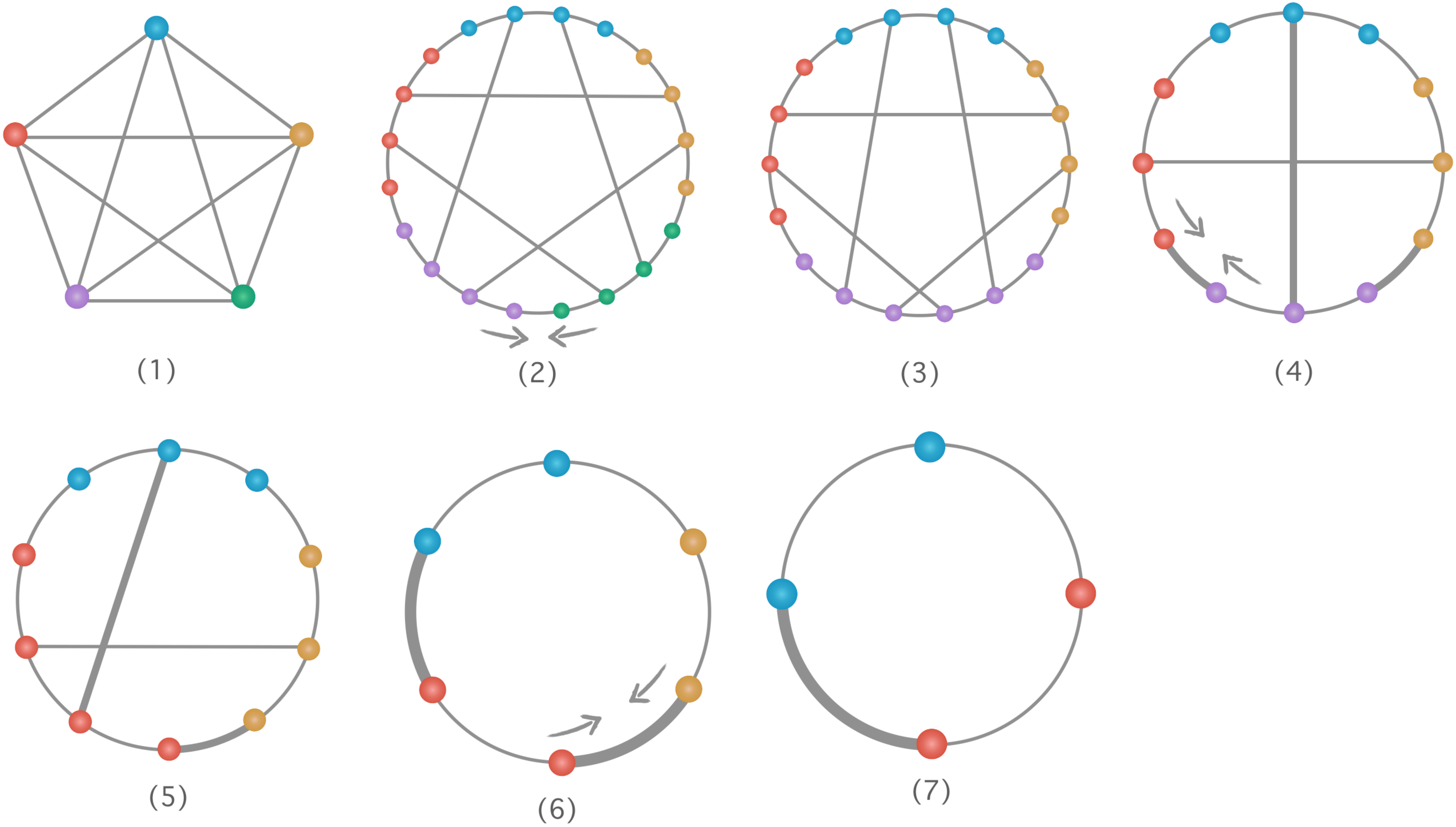
F. Pan, P. Zhou, S. Li , PZ, *PRL* 125,060503 (2020)

CATN: Contracting Arbitrary Tensor Networks



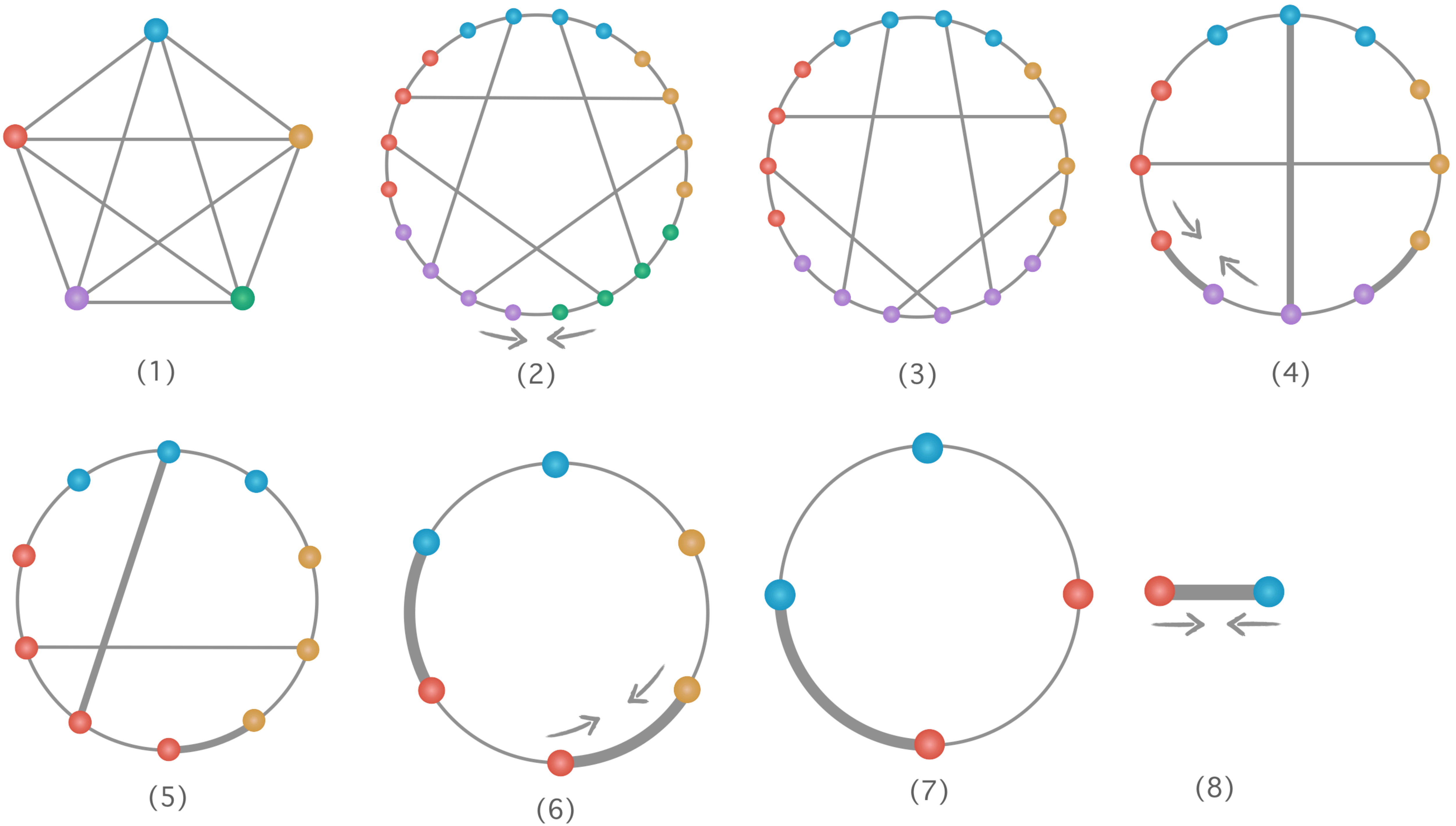
F. Pan, P. Zhou, S. Li , PZ, *PRL* 125,060503 (2020)

CATN: Contracting Arbitrary Tensor Networks



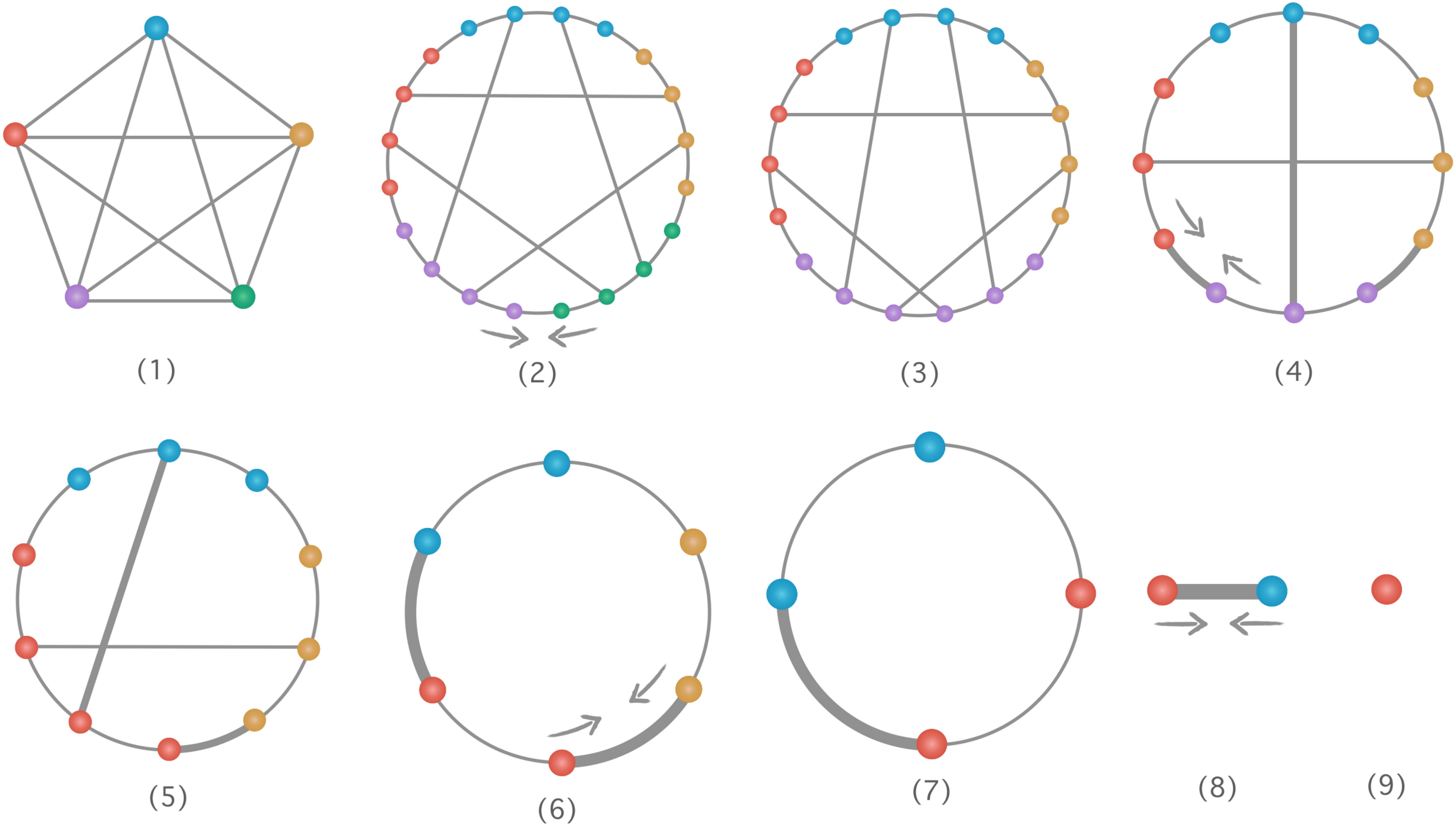
F. Pan, P. Zhou, S. Li , PZ, *PRL* 125,060503 (2020)

CATN: Contracting Arbitrary Tensor Networks



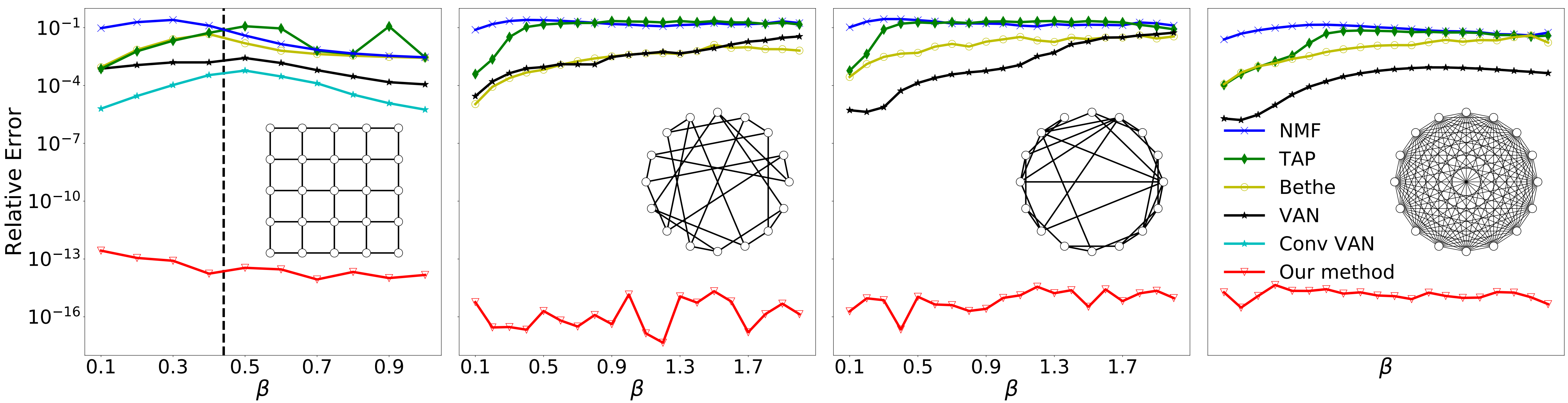
F. Pan, P. Zhou, S. Li , PZ, *PRL* 125,060503 (2020)

CATN: Contracting Arbitrary Tensor Networks



F. Pan, P. Zhou, S. Li , PZ, *PRL* 125,060503 (2020)

Computing free energy of spin glasses



16x16 2D lattice

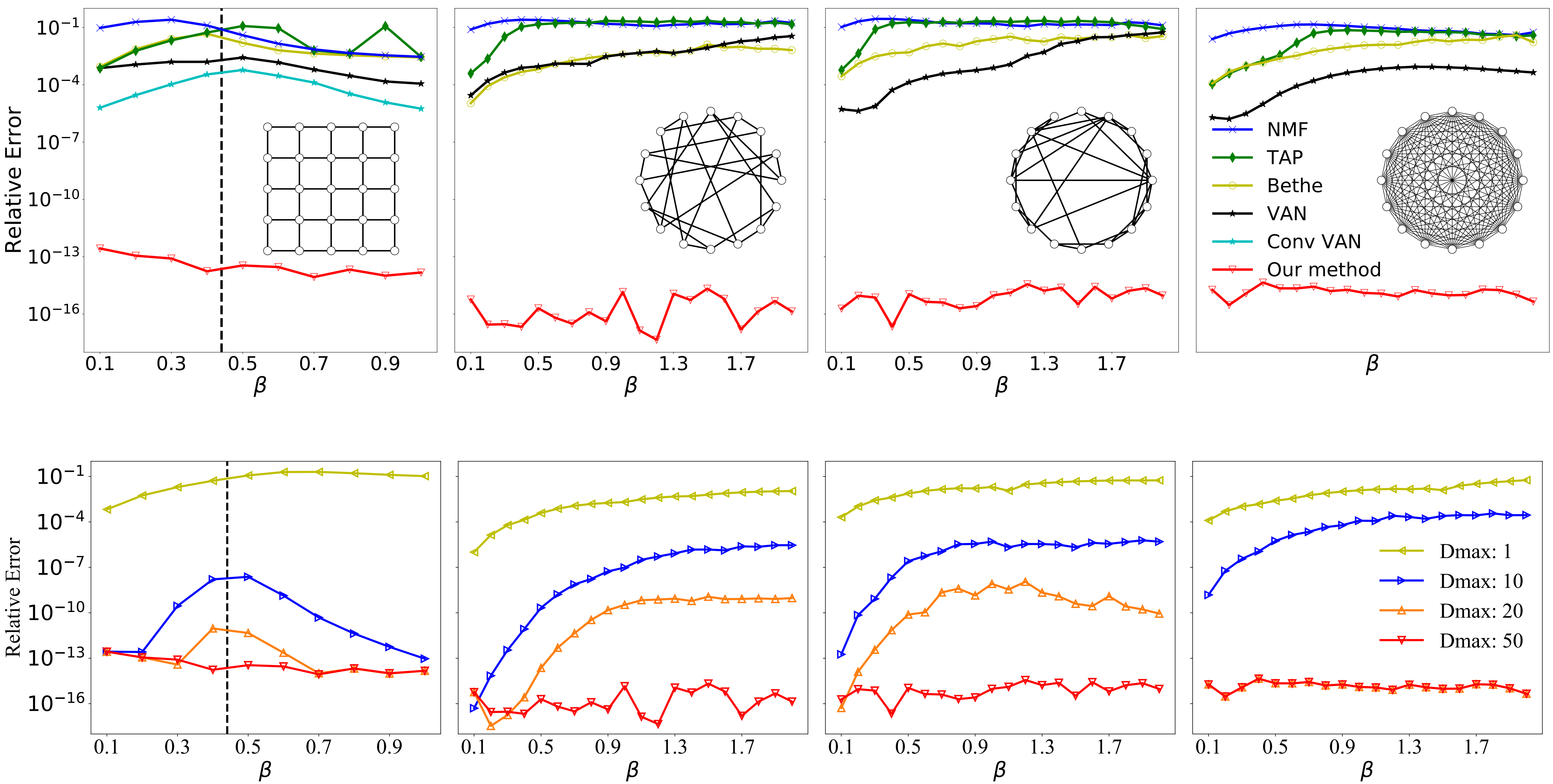
RRG graphs n=80, k=3

Small-world n=70, c=4

SK model n=20

F. Pan, P. Zhou, S. Li , PZ, *PRL* 125,060503 (2020)

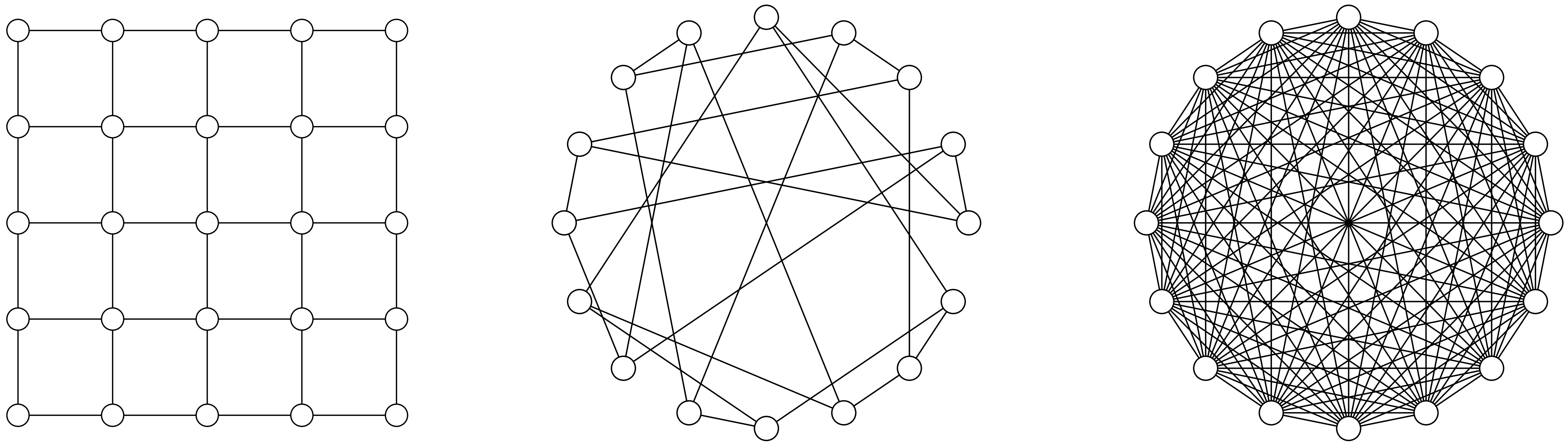
Computing free energy of spin glasses



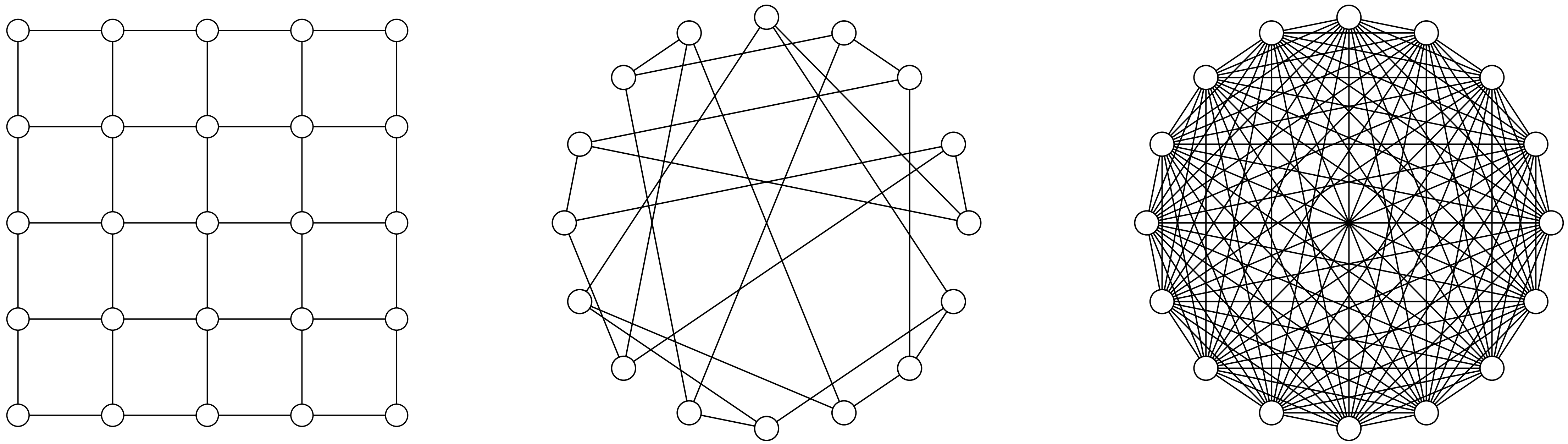
16x16 2D lattice | **Regular Random graphs n=80, k=3** | **Small-world n=70, c=4** | **SK model n=20**

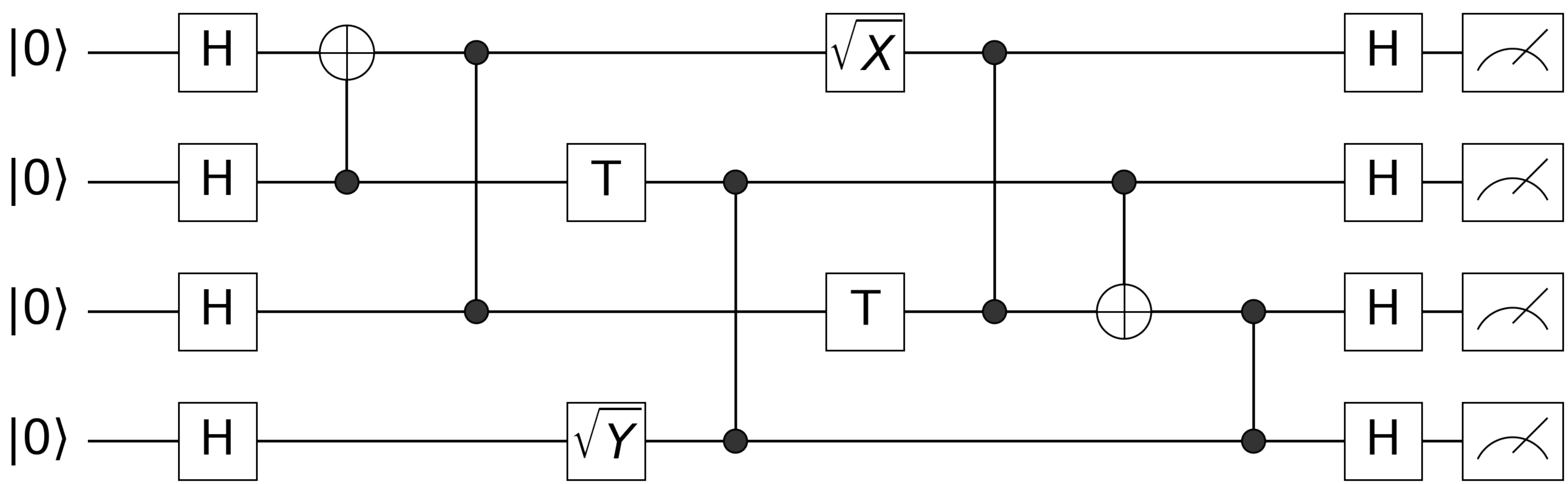
F. Pan, P. Zhou, S. Li , PZ, *PRL* 125,060503 (2020)

From partition function to quantum circuit simulation

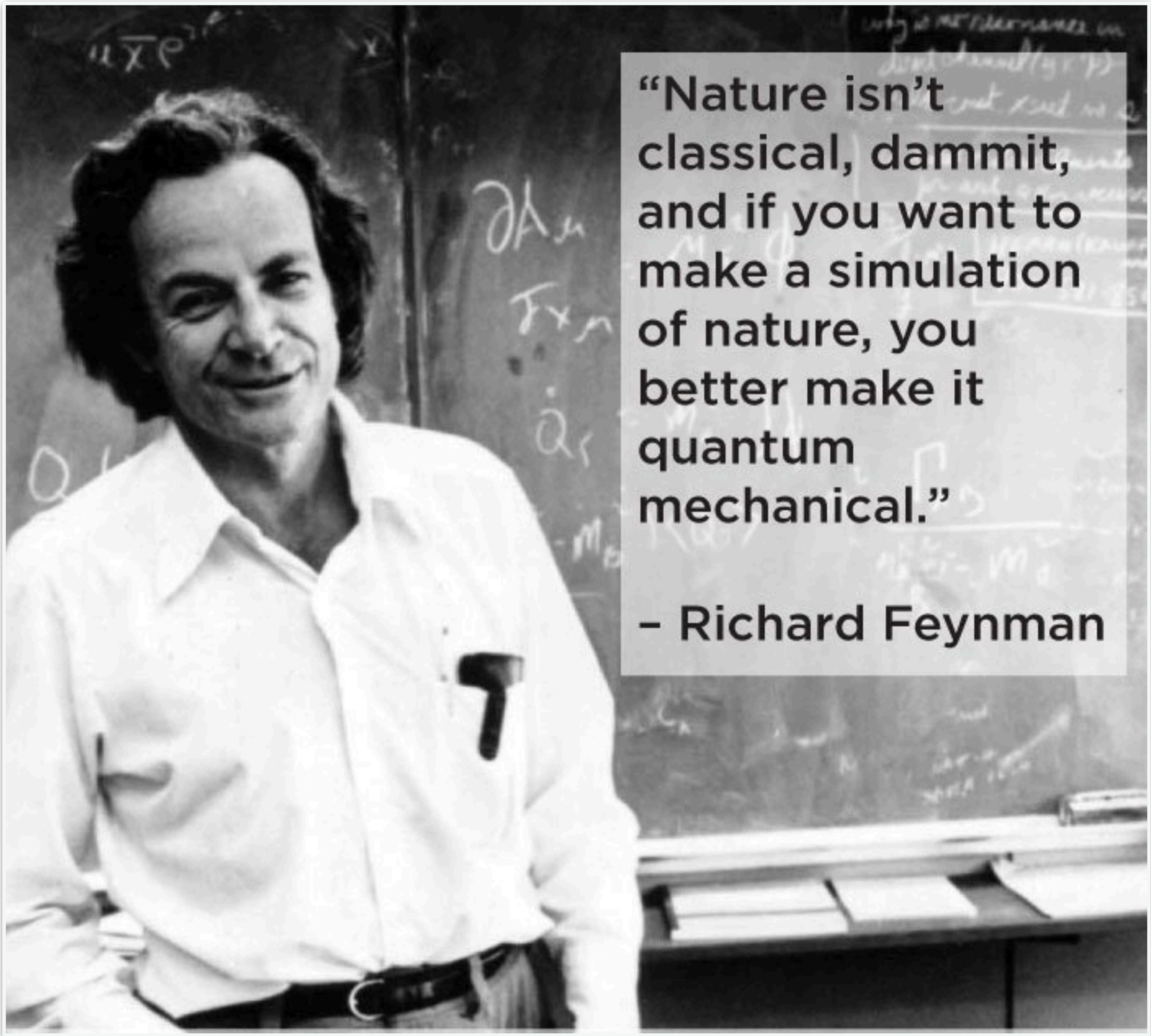


From partition function to quantum circuit simulation





量子计算机

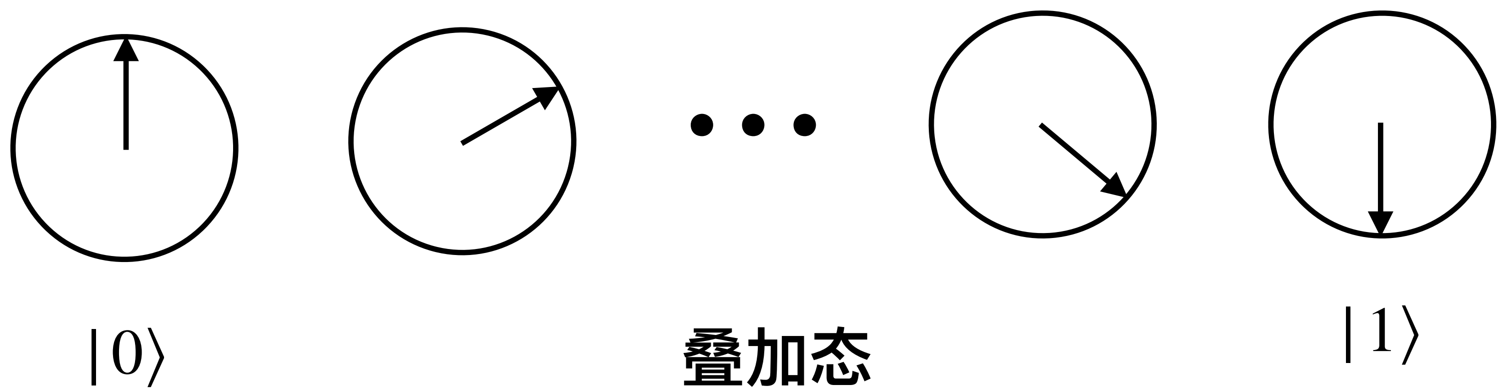


"利用量子力学做计算" —费因曼1981'

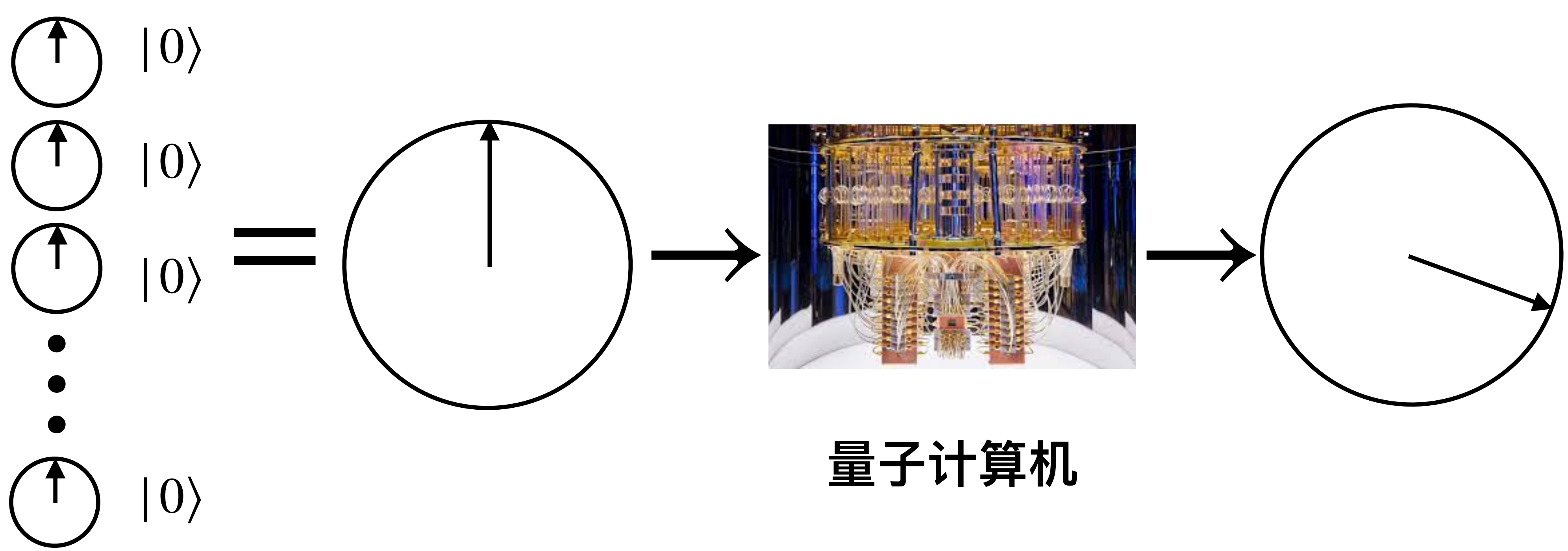
量子信息，量子计算

经典比特

量子比特

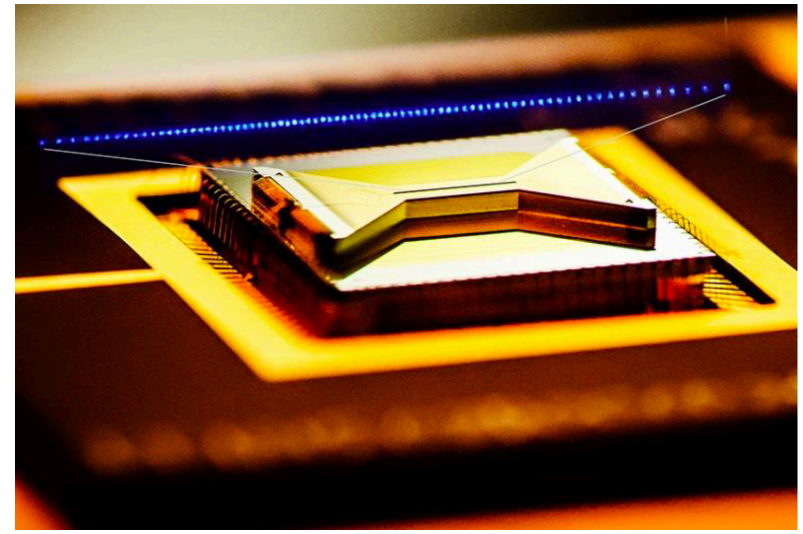


多个量子比特



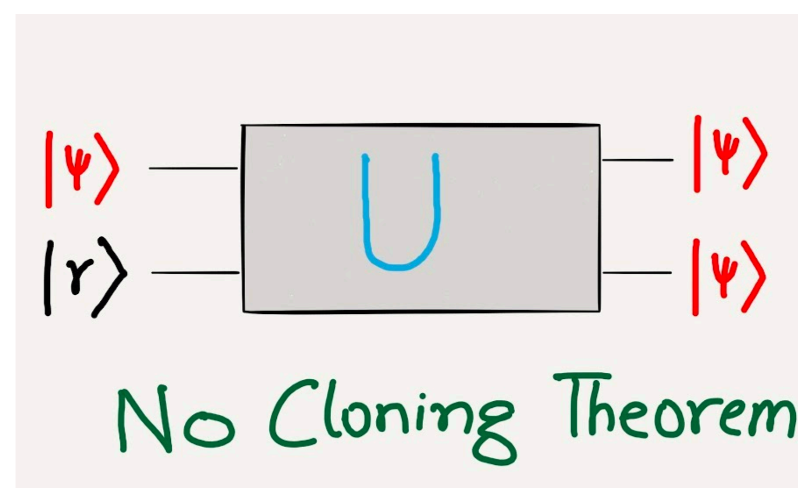
经典信息和量子信息

经典信息可以保存很长时间



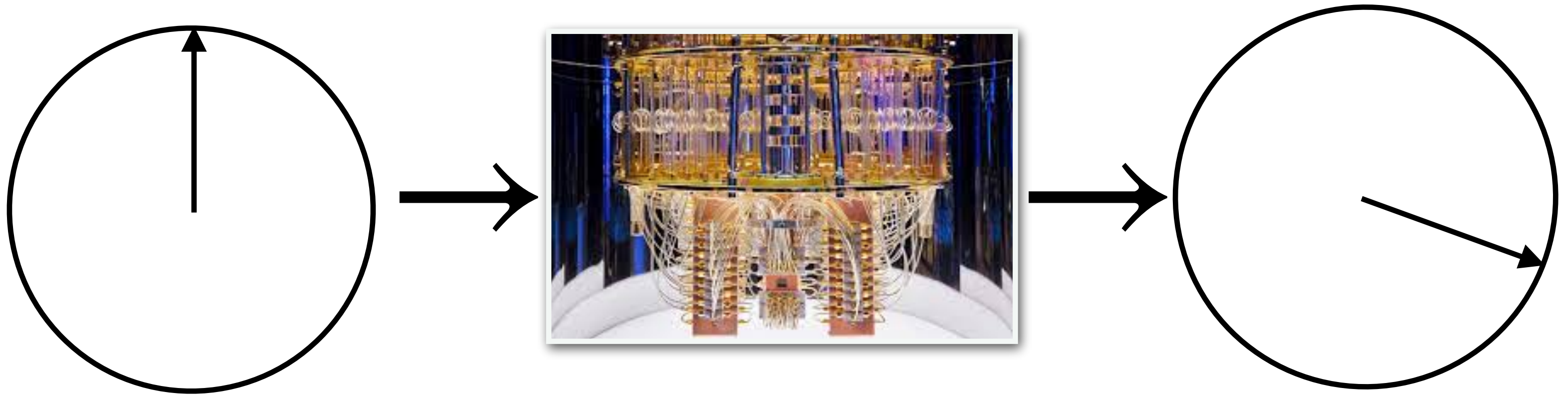
目前量子信息的保存时间很短
没有纠错

经典信息容易复制



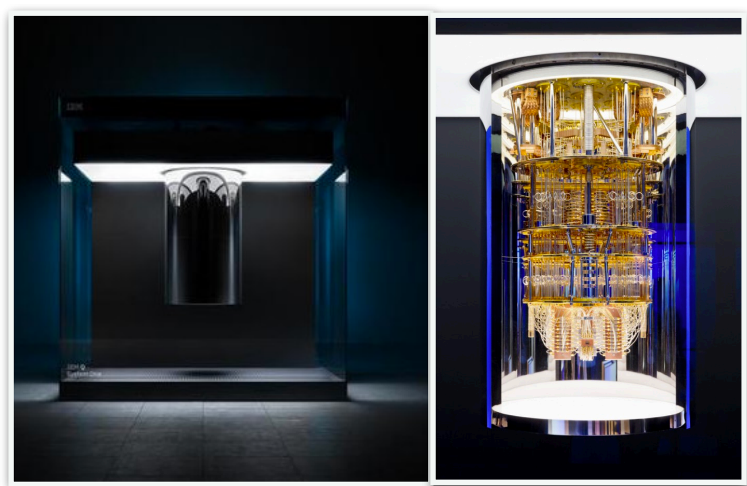
量子信息无法复制

量子计算机

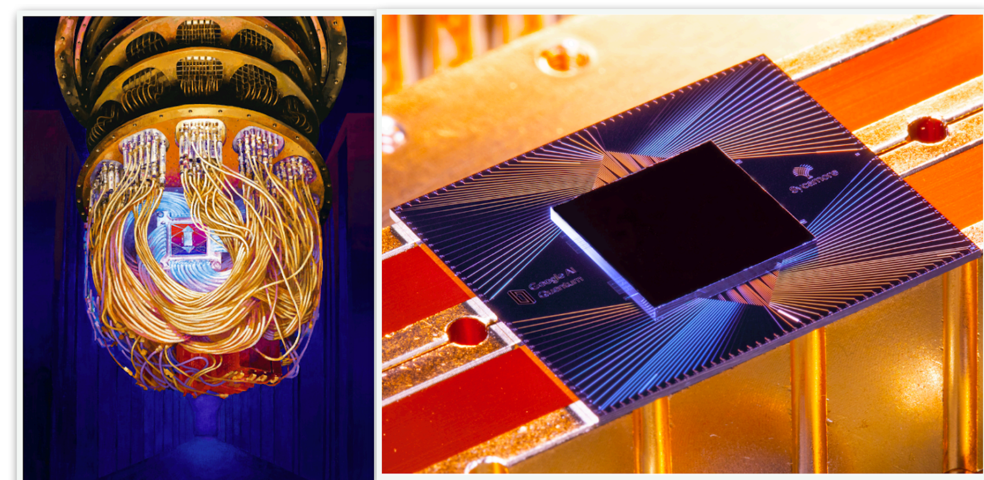


量子计算机

2016年
首个量子计算机在线平台
5个量子比特



2019年
IBM Q System One
首个商用量子计算机
20个量子比特



2019年
Google Sycamore
53个量子比特
宣称实现了量子霸权

量子计算的发展

量子计算原型机

- 几个量子比特，演示作用

↓

含噪音的量子计算机

- 几十个到上百个量子比特
- 有噪音，无纠错
- 演示量子优越性（量子霸权）

↓

通用量子计算机

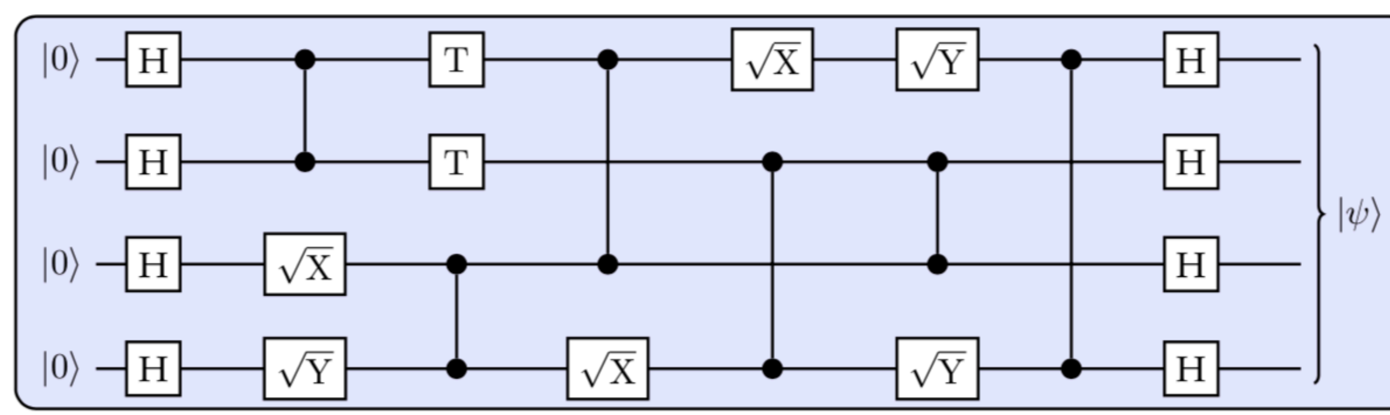
- 上百万量子比特
- 纠错
- 通用，Shor算法…

Google的量子霸权

量子计算机操纵
例如量子比特的量子数据

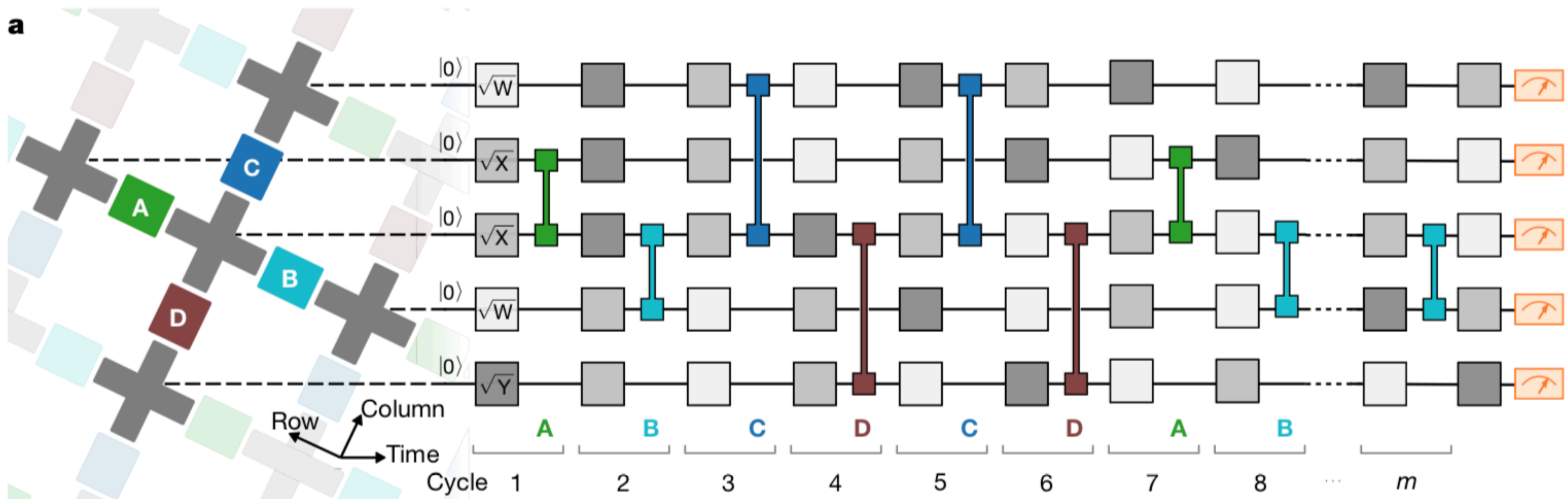
序列地作用量子门

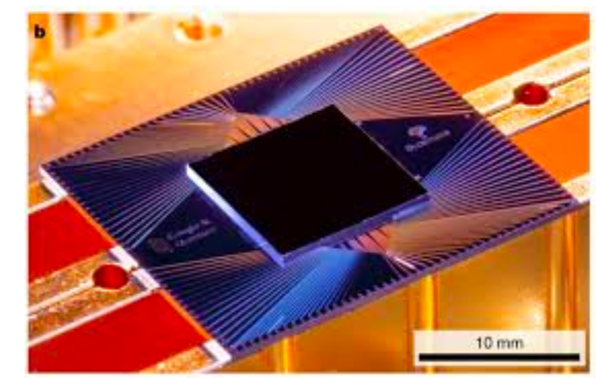
可以表达任意幺正变换

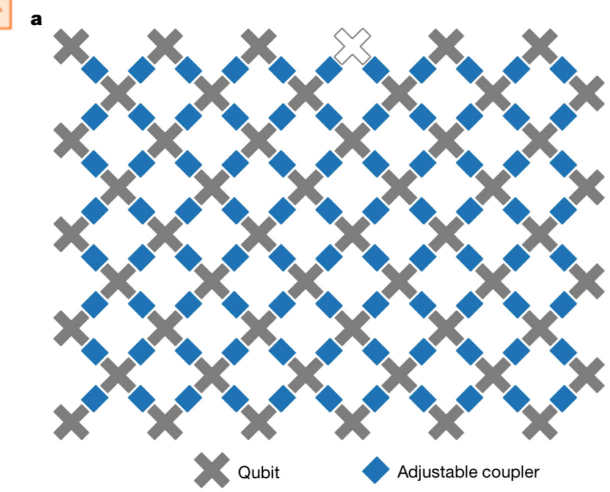


Arute et al, Nature 574 505 (2019)

Google's Quantum Supremacy experiments







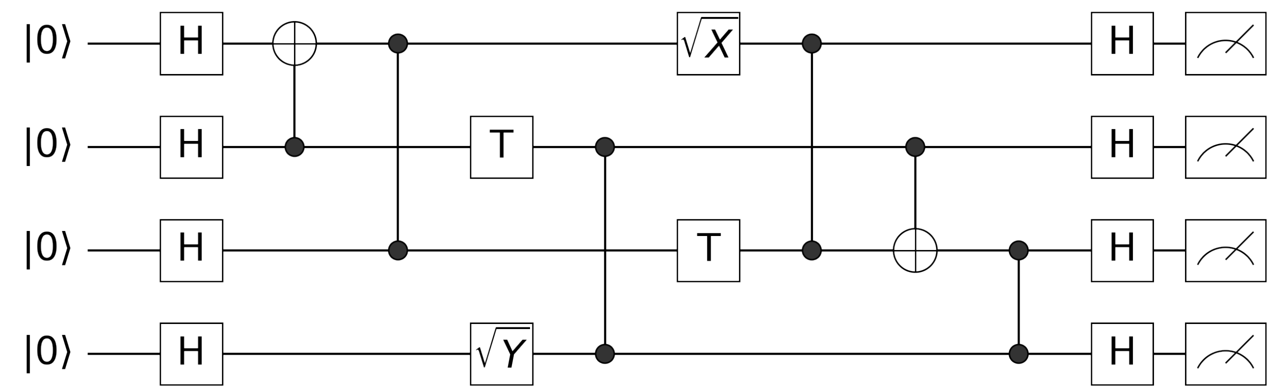
Arute et al, Nature 574 505 (2019)

- **53 qubits, 20 cycles of unitary operations**
- **1 million samples within 200 seconds**
- **Linear Cross Entropy Fidelity (XEB) = 0.002, given by extrapolations**
- **Google's (Shrödinger-Feynmann) classic algorithm requires 10,000 years on Summit**

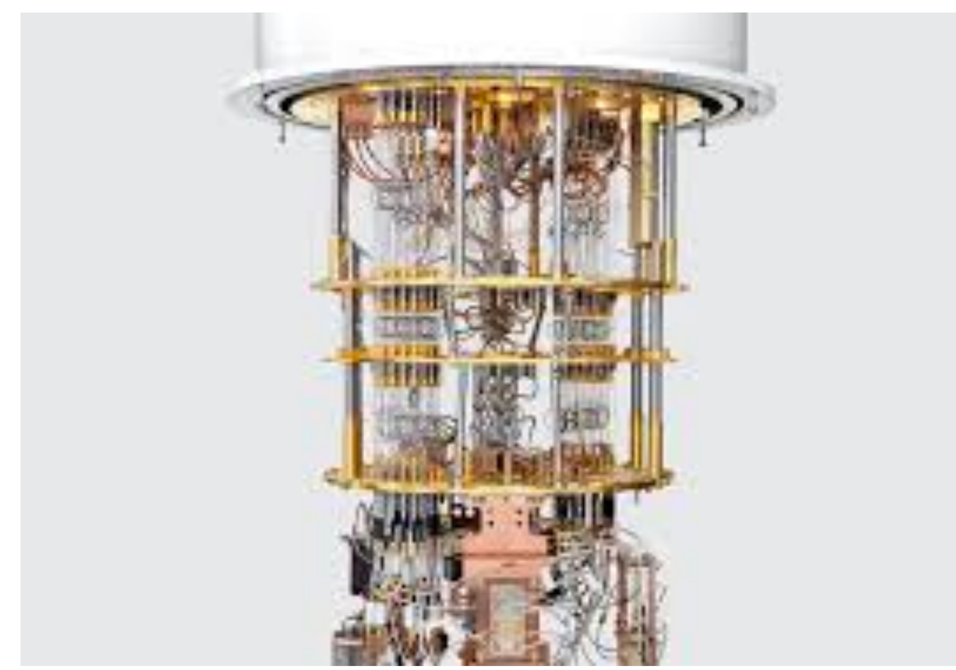
Simulation methods

Full amplitude:

- Storing full state-vector [**Yao.jl**, Qiskit, Qulacs, Cirq...]
- Schrödinger-Feynmann
- MPS / Group MPS



Single/batch amplitude:



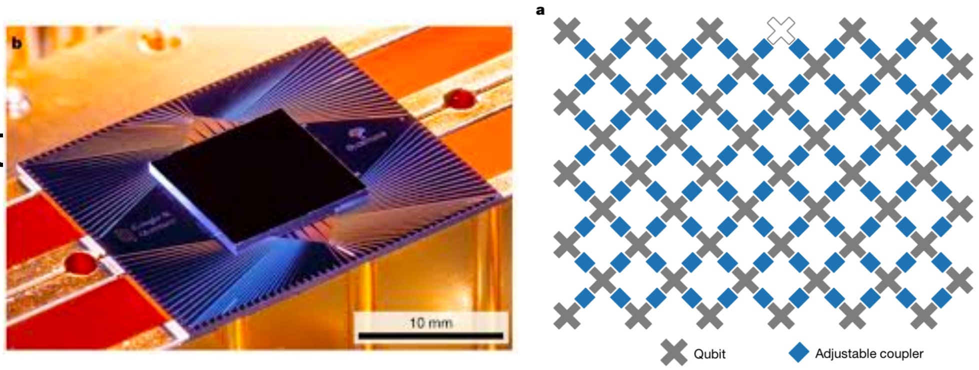
- PEPS based (single amplitude)
- Cotengra (single amplitude)
- Alibaba ACQDP (64-amplitude batch)
- Our method (big-batch)

Classical simulation of Sycamore

Classical simulation of Sycamore

- Google's original estimate [**Arute et. al. 2019**]

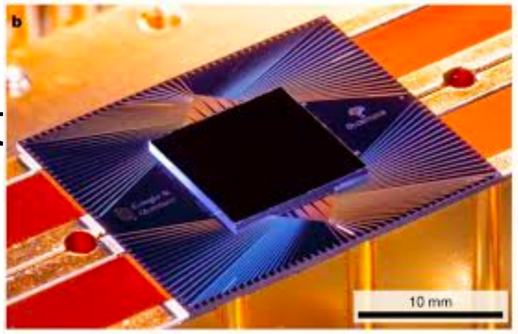
    - 10,000 years for simulating the Sycamore circuit with 53 qubits and 20 cycles (using Summit)

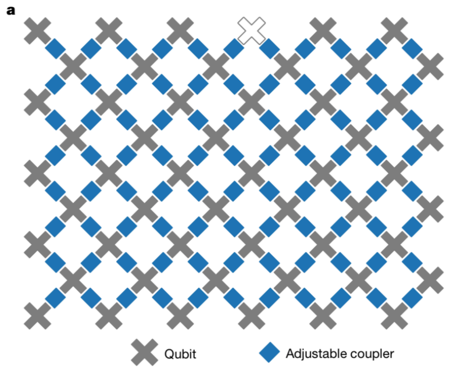


Images from Arute et. al. Nature 2019

Classical simulation of Sycamore

- Google's original estimate [**Arute et. al. 2019**]
  - 10,000 years for simulating the Sycamore circuit with 53 qubits and 20 cycles (using Summit)



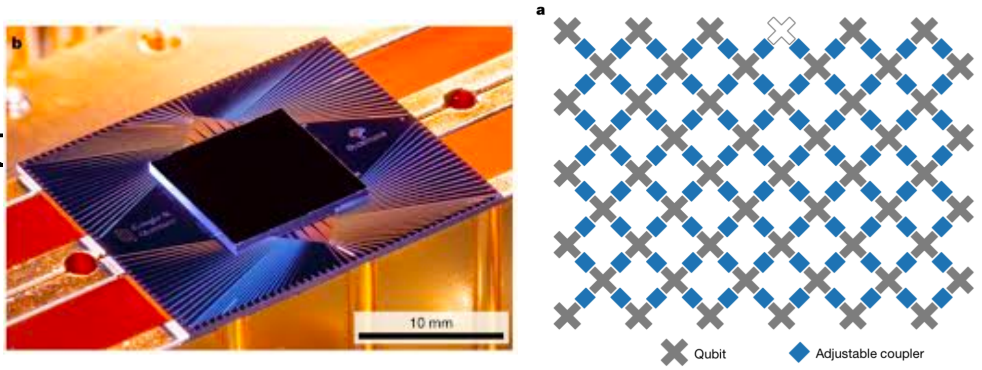


Images from Arute et. al. Nature 2019

- IBM's estimate [**Pednault et al 2019**]:
  - 2.5 days (with 250PB memory, all memory and hard disks of Summit)

Classical simulation of Sycamore

- Google's original estimate [**Arute et. al. 2019**]
  - 10,000 years for simulating the Sycamore circuit with 53 qubits and 20 cycles (using Summit)

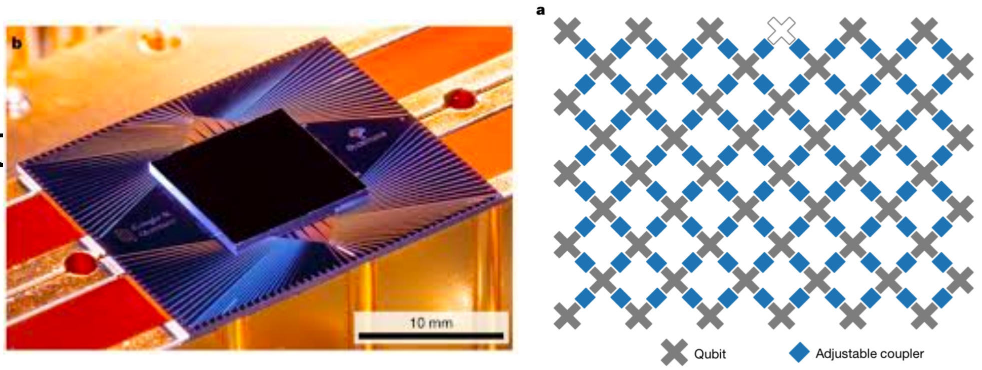


Images from Arute et. al. Nature 2019

- IBM's estimate [**Pednault et al 2019**]:
  - 2.5 days (with 250PB memory, all memory and hard disks of Summit)
- Cotengra [**Gray/Kourtis 2020**]
  - Balanced Partitioning (`Kahypar`) + Greedy/optimal + Slicing
  - 3000 years for single amplitude (Single GPU)

Classical simulation of Sycamore

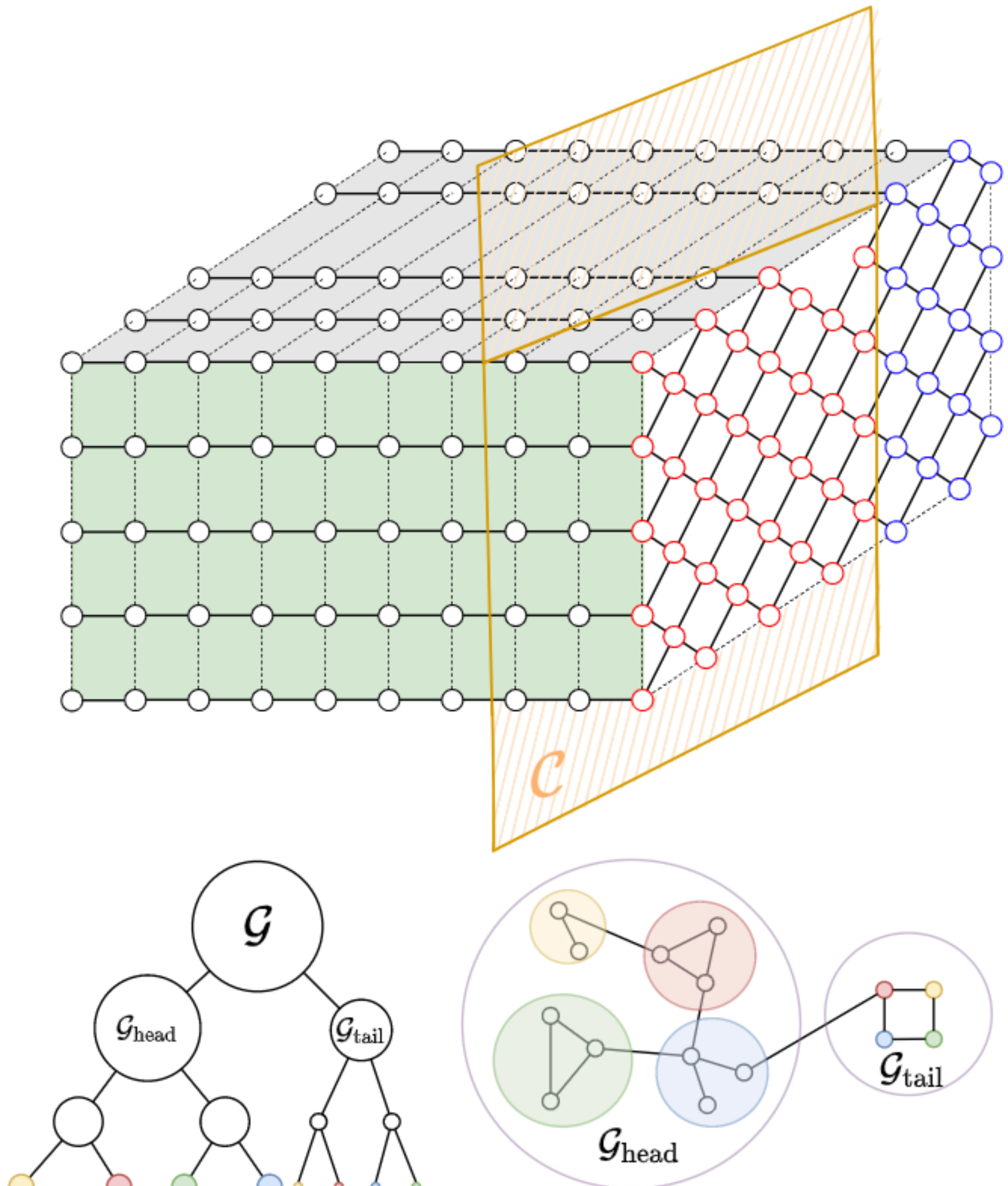
- Google's original estimate [**Arute et. al. 2019**]
  - 10,000 years for simulating the Sycamore circuit with 53 qubits and 20 cycles (using Summit)

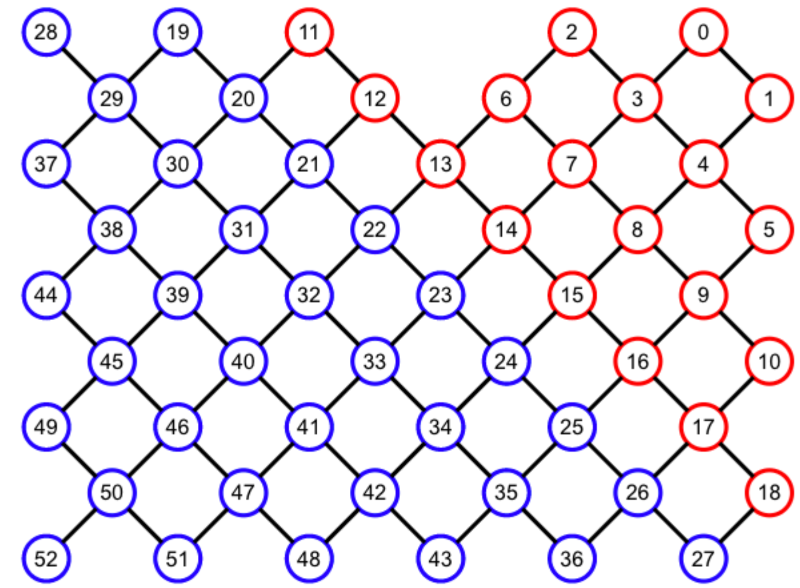


Images from Arute et. al. Nature 2019

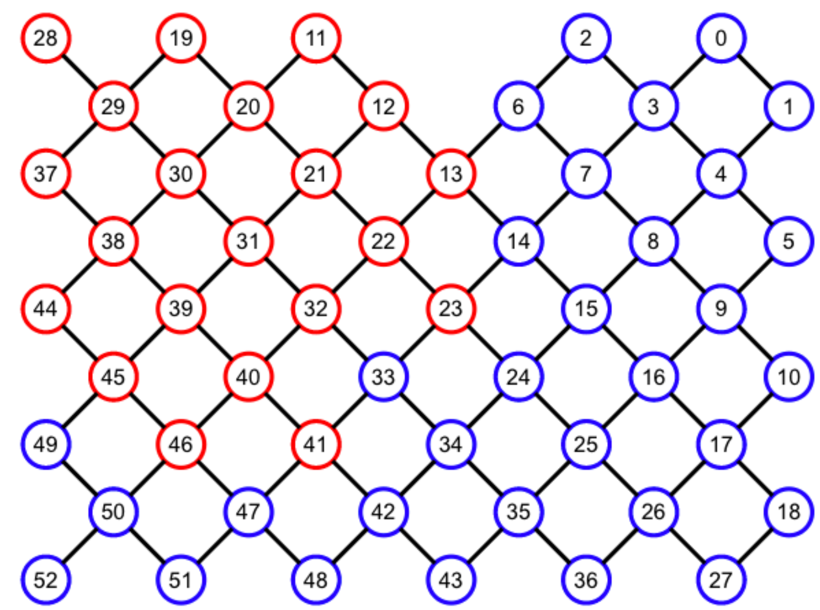
- IBM's estimate [**Pednault et al 2019**]:
  - 2.5 days (with 250PB memory, all memory and hard disks of Summit)
- Cotengra [**Gray/Kourtis 2020**]
  - Balanced Partitioning (`Kahypar`) + Greedy/optimal + Slicing
  - 3000 years for single amplitude (Single GPU)
- Alibaba's simulator [**Huang et. al. 2020**]
  - Hierarchical partitioning (`Kahypar`) + Greedy/optimal + Slicing + sampling
  - 20 days for 1 million samples (with Summit-compatible supercomputer)

Our approach: Big-Head tensor network method





14 cycles



20 cycles

arXiv:2013.03074

Computational complexity

	# bitstrings	Time complexity	Space complexity	Computational time	Computational hardware
Google [1]	10^6	—	—	10,000 years	Summit supercomputer
Cotengra [12]	1	3.10×10^{22}	2^{27}	3,088 years	One NVIDIA Quadro P2000
Alibaba [18]	64	6.66×10^{18}	2^{29}	267 days	One V100 GPU
Ours	**2097152**	4.51×10^{18}	2^{30}	149 Days	One A100 GPU

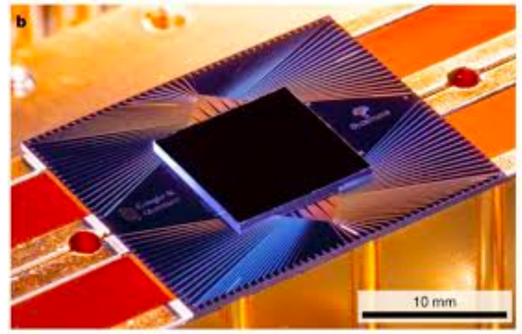
- **The computational cost of our algorithm in obtaining 2 million amplitudes is smaller than obtaining 64 amplitudes using Alibaba's method.**
- **Google, Cotengra, and Alibaba's results are estimations**
- **We did the computations for the first time.**

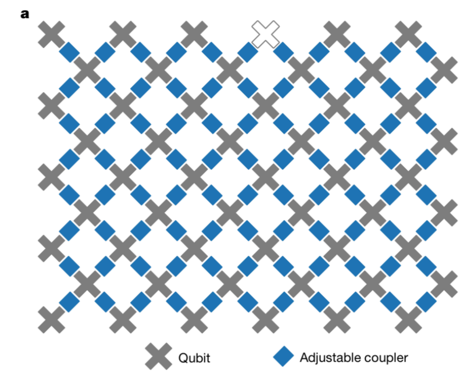
arXiv:2013.03074

Simulating Sycamore with 53 qubits, 20 cycles

	Computation hardward	**Time**
Google [Arute et. al., 2019] (**Estiamte**)	Summit Super Computer	10,000 Years
IBM [Pednault et. al., 2019] (**Estimate**)	Summit Super Computer (all disks)	2.5 days
Alibaba [Huang et. al., 2020] (**Estimate**)	Summit Super Computer (compatible)	20 days
Ours [arXiv:2103.03074] (**Computation**)	60 GPUs	5 days

Simulating Sycamore with 53 qubits, 20 cycles





	Computation hardward	**Time**
Google [Arute et. al., 2019] (**Estiamte**)	Summit Super Computer	10,000 Years
IBM [Pednault et. al., 2019] (**Estimate**)	Summit Super Computer (all disks)	2.5 days
Alibaba [Huang et. al., 2020] (**Estimate**)	Summit Super Computer (compatible)	20 days
Ours [arXiv:2103.03074] (**Computation**)	60 GPUs	5 days

Main results

$$F_{\mathrm{XEB}} = \frac{2^n}{L} \sum_{i=1}^{L} P_U(\mathbf{s}_i) - 1$$

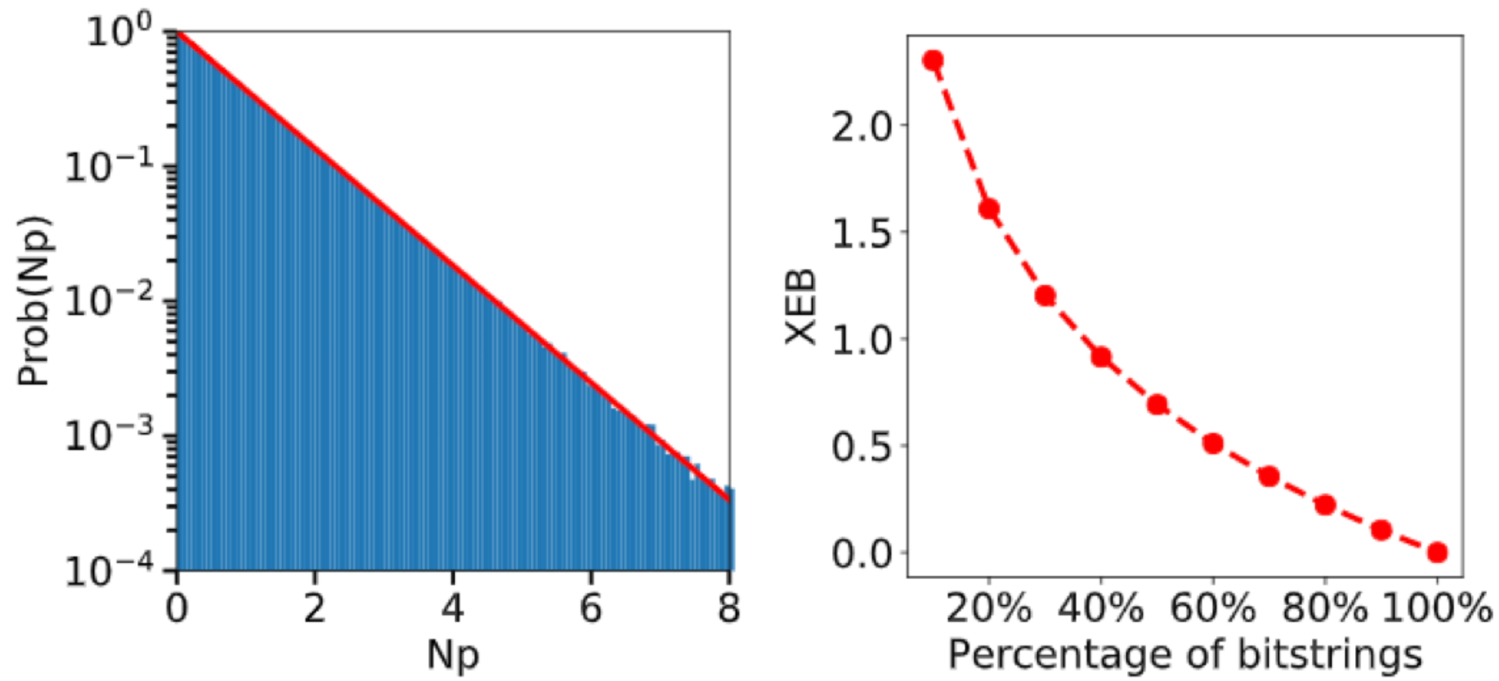


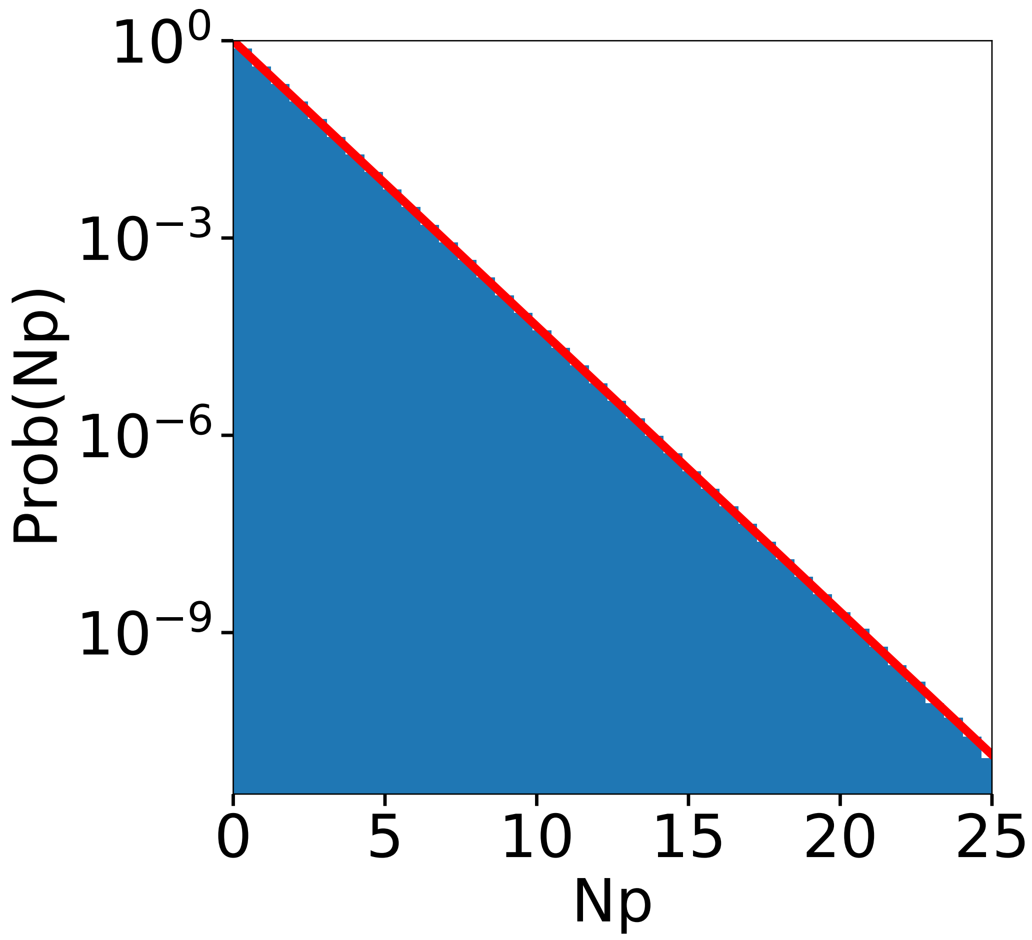
FIG. 2. (Left): Histogram of bitstring probabilities $P_U(\mathbf{s}) = P_U(\mathbf{s}_1; \mathbf{s}_2)$ for $l = 2^{21}$ bitstrings obtained from the Sycamore circuit with $n = 53$ qubits, $m = 20$ cycles, sequence ABCDCDAB, seed 0, and the assignment of partial bitstring $\mathbf{s}_1$ are fixed to $\underbrace{0, 0, 0, \cdots, 0}_{32}$.

- **Obtained 2 million amplitudes and probabilities, following the Porter-Thomas distribution**

- **Sampled 1 million bitstrings, XEB fidelity=0.739, larger than Google's XEB**

arXiv:2013.03074

Full amplitude computations

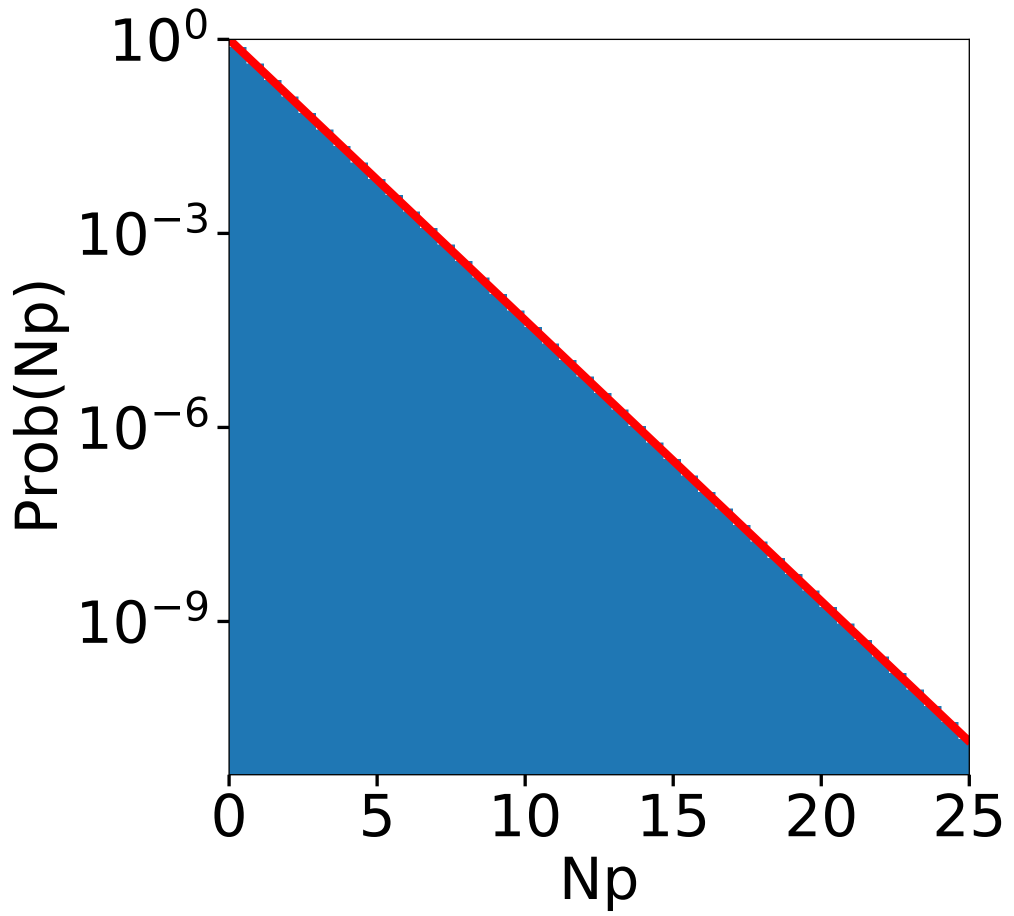


43 qubits, 14 cycles, EFGH sequence

Google: the Jülich supercomputer

(with 100,000 cores, 250 terabytes)

Arute et al, Nature 574, 505 (2019).



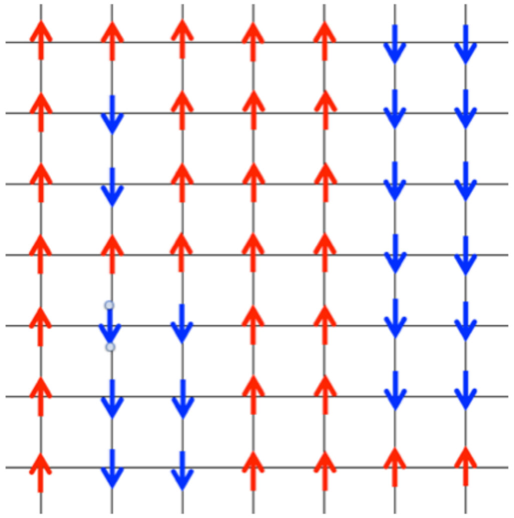
50 qubits, 14 cycles, EFGH sequence

The previous record was 49 quits

(Using the Sunway Taihulight super computer)

R. Li, B. Wu, M. Ying, X. Sun, and G. Yang, IEEE Transactions on PDS 31, 805 (2019).

Computation with tensor networks

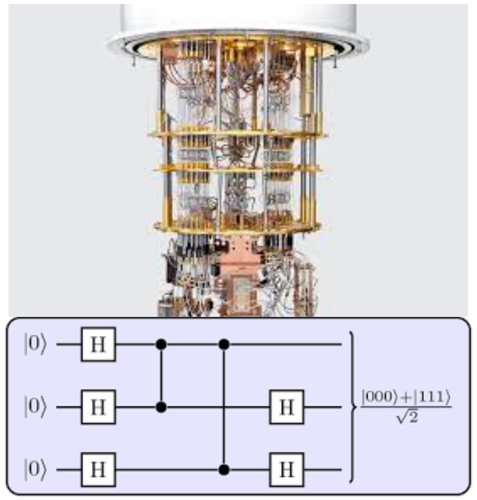


Joint probability distribution of microscopic configurations

$$P(\sigma) = \frac{1}{Z}\exp(-\beta E(\sigma))$$

Joint probability distribution of data variables

$$P(\mathrm{Data})$$



Control of quantum states

$$\psi(\sigma)$$

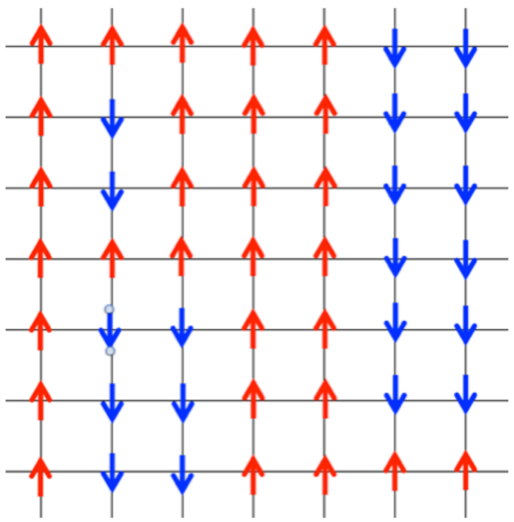
Exponential space

Effective models

Computational power

Tensor Network as a bridge !

Computation with tensor networks

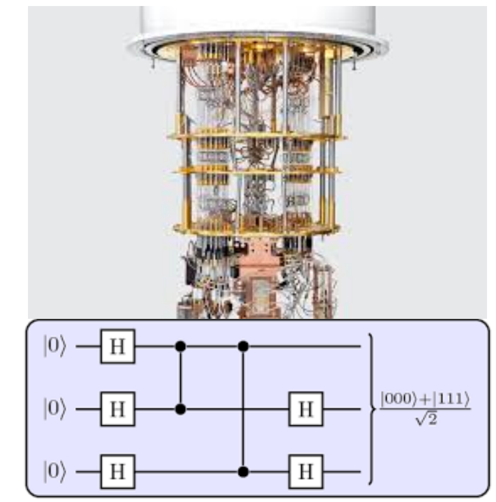


Joint probability distribution of microscopic configurations

$$P(\sigma) = \frac{1}{Z}\exp(-\beta E(\sigma))$$

Joint probability distribution of data variables

$$P(\mathrm{Data})$$



Control of quantum states

$$\psi(\sigma)$$

Exponential space
Effective models
Computational power
Tensor Network as a bridge !

Reference：
D.Wu, L. Wang, PZ, *PRL* 122, 080602 (2019)
F. Pan, P. Zhou, S. Li , PZ, *PRL* 125, 060503 (2020)
J. Liu, L. Wang, PZ, *PRL* 126, 090506 (2021)
F. Pan, PZ, arXiv:2103.03074 (2021)
S. Li, P. Zhou, F. Pan, PZ, arXiv:2105.04130 (2021)