Title: Restricted Boltzmann Machines (RBMs)

Speakers: Mohamed Hibat Allah

Collection: Navigating Quantum and AI Career Trajectories: A Beginnerâ€™s Mini-Course on Computational Methods and their Applications

Date: May 21, 2024 - 11:30 AM

URL: https://pirsa.org/24050083

Lecture 1

Restricted Boltzmann Machines (RBMs)

* Outline:

1. Motivation for Generative modeling.
2. Generative models classification.
3. Restricted Boltzmann Machines (RBMs).

Lecture 1

Restricted Boltzmann Machines (RBMs)

* Outline:

1. Motivation for Generative modeling.
2. Generative models classification.
3. Restricted Boltzmann Machines (RBMs).

↳ Machine Learning references:

② Generative models classification.

③ Restricted Boltzmann Machines (RBMs).

↳ Machine Learning references:

↳ Machine Learning for many-body physics, PSI course, pirsa.org/c24027

↳ "Deep learning specialization", Coursera, Andrew Ng

↳ "Neural Networks and Deep Learning", Nielson

* References for this lecture:

↳ "A Practical guide to Training Restricted Boltzmann Machines". G. Hinton

↳ "A high-bias, low-variance introduction to machine learning for physicists", ArXiv : 1803.08823

↳ "Restricted Boltzmann Machines : A short Tutorial" by Perimeter Institute Quantum Intelligence

https://qucumber.readthedocs.io/en/stable/_static/RBM_tutorial.pdf

① Motivation for generative modeling : → very relevant applications to 2Phys

↳ It's a machine learning strategy among many strategies

→ Supervised learning $D = \{(\vec{x}, \vec{y})\}$

$\vec{x}$: Data points, $\vec{y}$: Labels

↳ Learn f such that $f(\vec{x}) \simeq \vec{y}$

↳ Infer $f(\vec{x}_{new})$ ← New datapoints

→ Supervised learning $D = \{(\vec{x}, \vec{y})\}$

Data points — Labels

↳ Learn f such that $f(\vec{x}) \simeq \vec{y}$

↳ Infer $f(\vec{x}_{new})$ ← New datapoints

← Good generalization

→ Unsupervised learning: $D = \{\vec{x}\}$ ← No labels

Dimensionality reduction

Generative modeling

Data points

↳ Learn f such that $f(\vec{x}) \simeq \vec{y}$

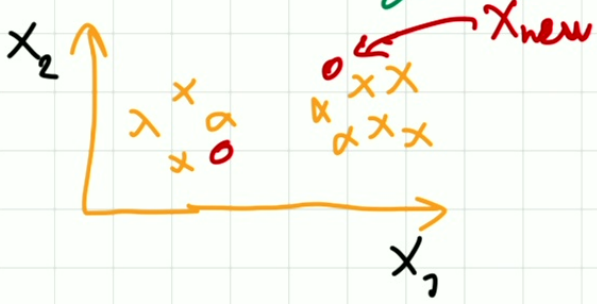
↳ Infer $f(\vec{x}_{new})$ ← New datapoints

← Good generalization

→ Unsupervised learning: $D = \{\vec{x}\}$ ← No labels

Dimensionality reduction

Generative modeling



→ Reinforcement learning:

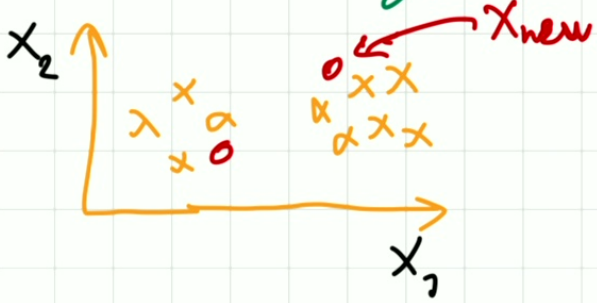
↳ Infer $f(\vec{x}_{new})$ ← New datapoints

← Good generalization

→ Unsupervised learning: $D = \{\vec{x}\}$ ← No labels

Dimensionality reduction

Generative modeling



→ Reinforcement learning:

↳ We aim to use a ML model to maximize a reward r

* Generative modeling:

e.g. GAN (deep fake), diffusion models (DALL-E)

ChatGPT, GPT-4 → cf. Ethics session at 2pm

* Principle:

↳ We have a dataset $\{\vec{x}\}$ sampled from $P_{data}(\vec{x})$, what's $P_{data}(\vec{x})$?

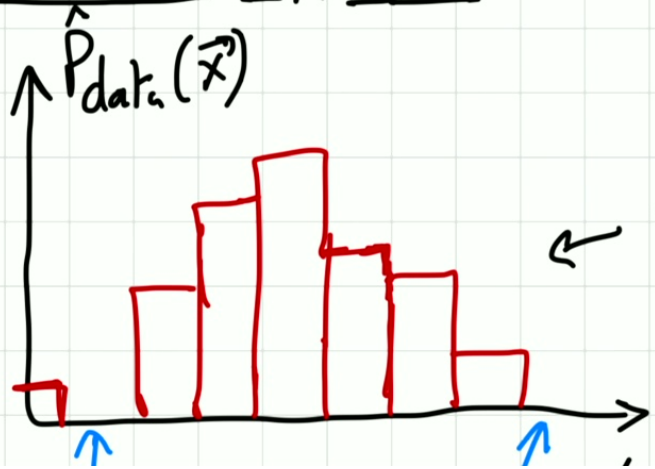
→ Traditional approach: (statistics)

ChatGPT, GPT-4 → cf. Ethics sesion at 2pm

* Principle:

↳ We have a dataset $\{\vec{x}\}$ sampled from $P_{data}(\vec{x})$, what's $P_{data}(\vec{x})$?

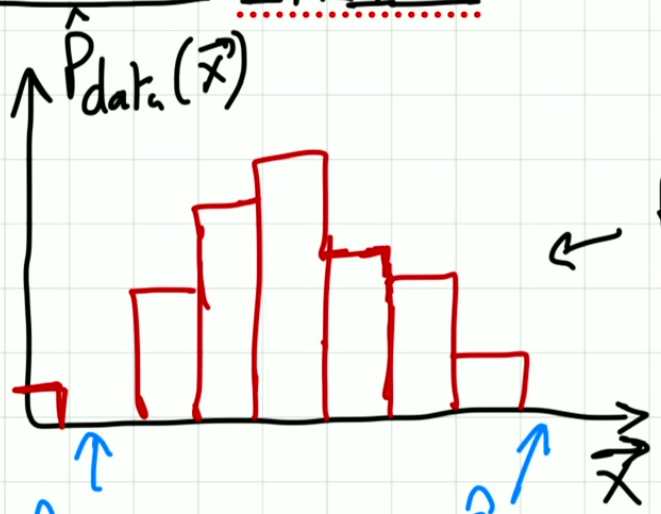
→ Traditional approach: (Statistics)



$$\hat{P}(\vec{x}) \simeq \frac{1}{|D|} \sum_{\vec{x}} \delta(\vec{x} - \vec{x}_k)$$

↳ We have a dataset $\{\vec{x}\}$, what's $P_{data}(\vec{x})$?

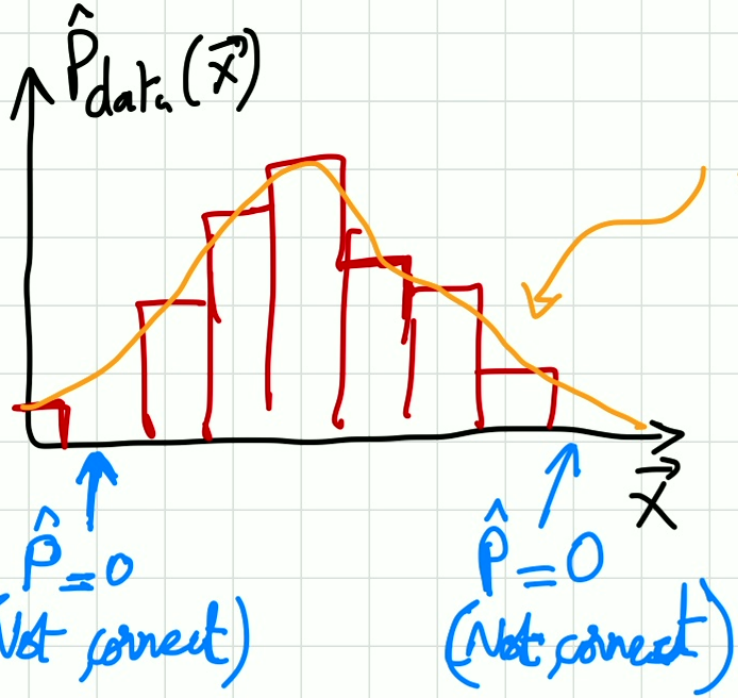
→ Traditional approach: (Statistics)



$$\leftarrow P(\vec{x}) \simeq \frac{1}{|D|} \sum_{\vec{x}} \delta(\vec{x}-\vec{x}_k)$$

Two parameters

See demo on Colab ←

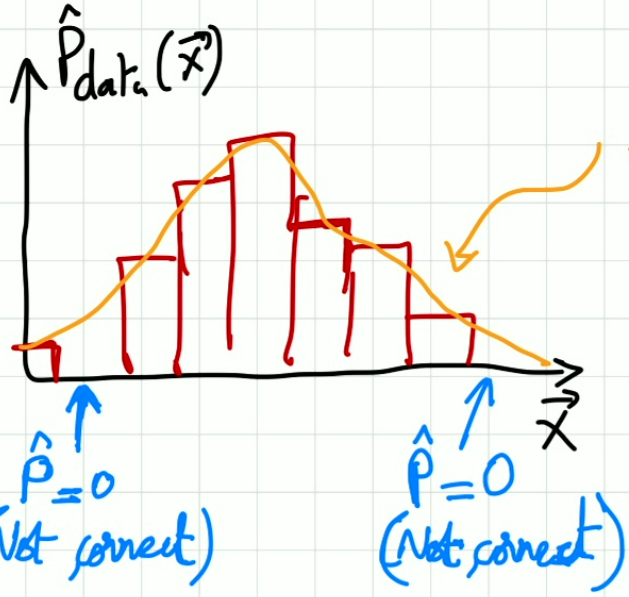


$\vec{\lambda}^* = (\mu^*, \sigma^*)$

→ μ^*, σ^* is obtained using the maximum likelihood principle

$\vec{\lambda}^* = \text{argmax}\ P_{\vec{\lambda}}(D)$

See demo on Colab

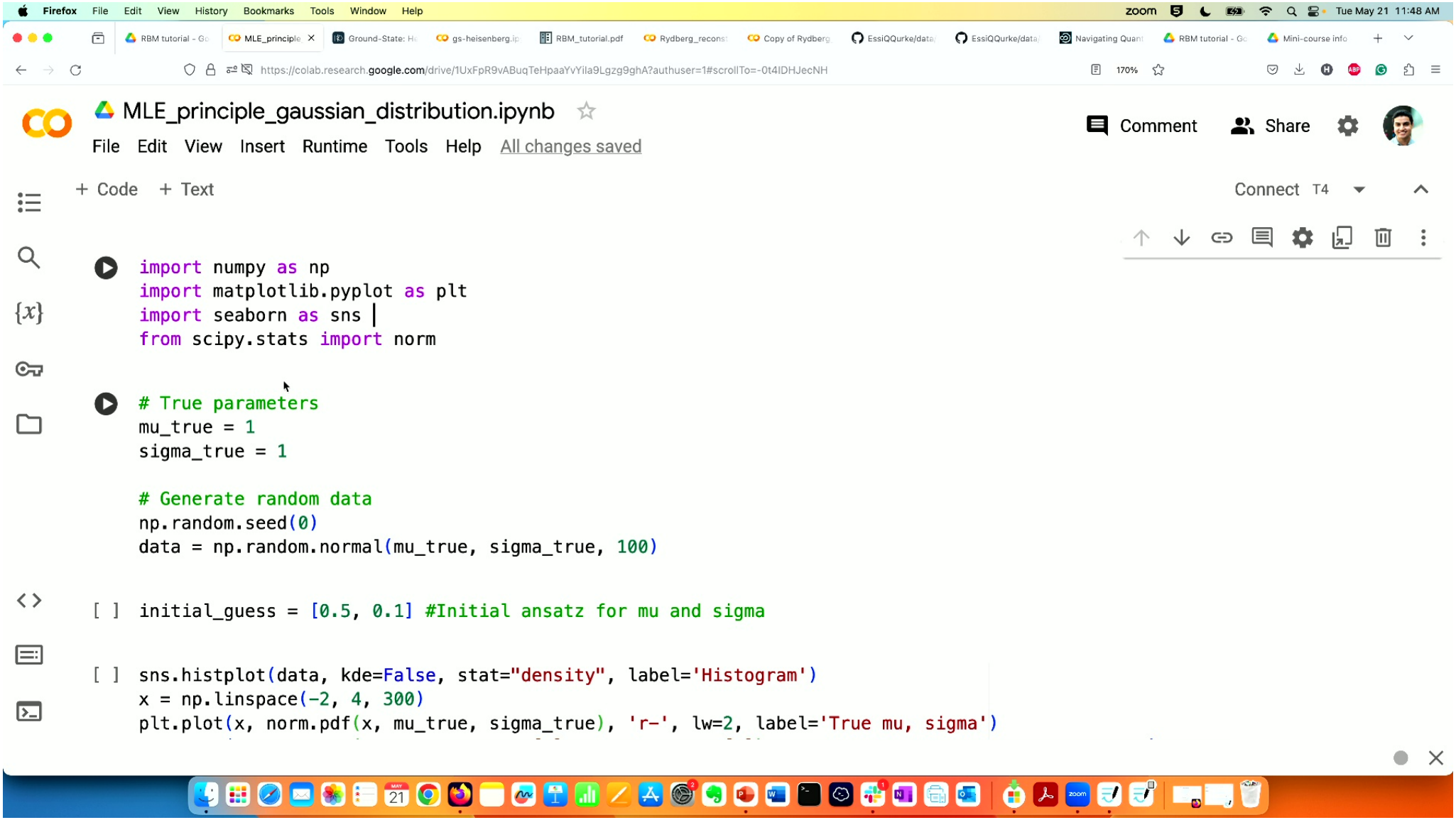


$\vec{\lambda}^* = (\mu^*, \sigma^*)$

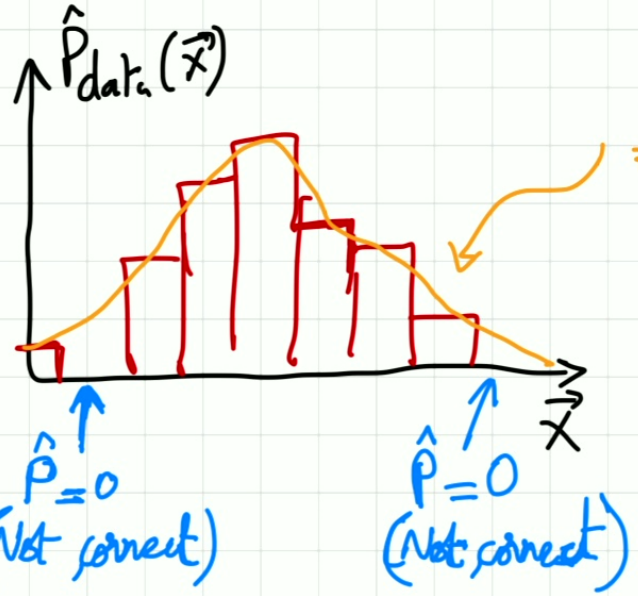
→ μ^*, σ^* is obtained using the maximum likelihood principle

$$\vec{\lambda}^* = \underset{\vec{\lambda}}{\operatorname{argmax}} \underbrace{P_{\vec{\lambda}}(D)}_{\prod_{i=1}^{M} P_{\vec{\lambda}}(\vec{x}_i)}$$

Assuming


MLE_principle_gaussian_distribution.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Connect T4
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import norm
# True parameters
mu_true = 1
sigma_true = 1
# Generate random data
np.random.seed(0)
data = np.random.normal(mu_true, sigma_true, 100)
[ ] initial_guess = [0.5, 0.1] #Initial ansatz for mu and sigma
[ ] sns.histplot(data, kde=False, stat="density", label='Histogram')
x = np.linspace(-2, 4, 300)
plt.plot(x, norm.pdf(x, mu_true, sigma_true), 'r-', lw=2, label='True mu, sigma')

See demo on Colab



$\vec{\lambda}^* = (\mu^*, \sigma^*)$

→ μ^*, σ^* is obtained using the maximum likelihood principle

$$\vec{\lambda}^* = \underset{\vec{\lambda}}{\text{argmax}} \underbrace{P_{\vec{\lambda}}(D)}_{\prod_i^M P_{\vec{\lambda}}(\vec{x}_i)}$$

Assuming

$P=0$ (Not correct) $P=0$ (Not correct)

→ μ^*, σ^* is obtained using the maximum likelihood principle

$$\vec{\lambda}^* = \underset{\vec{\lambda}}{\operatorname{argmax}} \underbrace{P_{\vec{\lambda}}(D)}_{\prod_{i=1}^{M} P_{\vec{\lambda}}(\vec{x}_i)}$$

← Assuming datapoints are independent

* Why do we do this?

↳ Efficient representation of $p_{data}(\vec{x})$

$\prod_{i=1}^{M} p_{\vec{\lambda}}(\vec{x}_i)$ ← Assuming datapoints are independent

* Why do we do this?

↳ Efficient representation of $p_{data}(\vec{x})$

↳ Calculating properties / Observables $\langle O \rangle_{p_{data}(\vec{x})}$

↳ Ability to generalize to unseen data.

* In quantum physics:

↳ Adapt generative models to construct $\psi_\lambda(\vec{x}) = \sqrt{P_\lambda(\vec{x})}$

$\propto \exp(i\phi_\lambda(\vec{x}))$

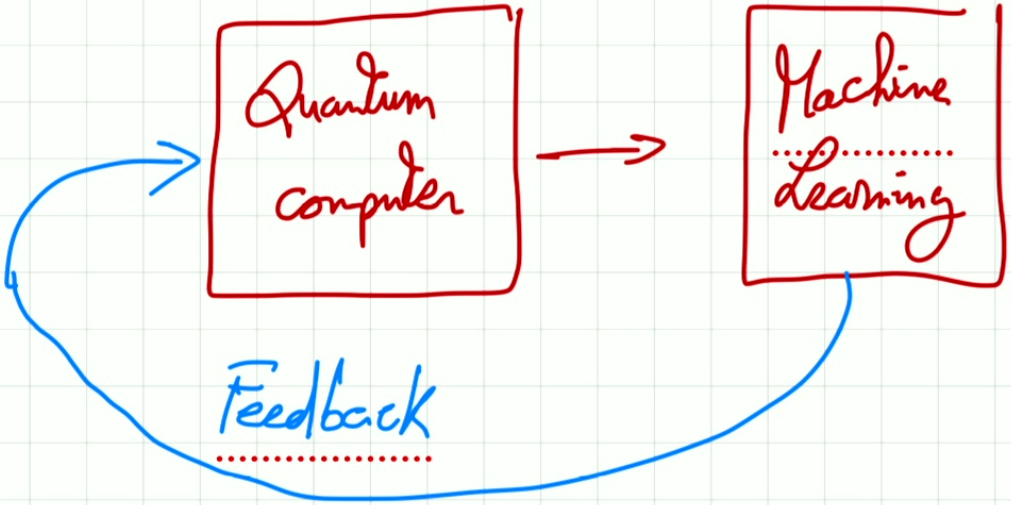
* Goal: find an ML model in $D > 1$ better than SOTA.

→ Tensor Networks are very effective in $D = 1$.

→ Tensor Networks are very effective in $D=1$.

Another Goal:

↳ Advance the development of Quantum Computers.



(2) Classification of generative models:

Explicit density

Direct parametrization

- **Tractable**
  - « Recurrent Neural Networks (RNNs) » → Thurs
  - « Transformers » → Friday
- **Intractable**
  - « Restricted Boltzmann Machines (RBMs) » → Today

Implicit density

No direct parametrization

(Generative Adversarial Network)

③ Restricted Boltzmann machines (RBM)

↳ Belongs to the class of energy-based models

$$P_{\vec{\lambda}}(\vec{x}) = \frac{\exp\left(-E_{\vec{\lambda}}(\vec{x})\right)}{Z_{\vec{\lambda}}}$$

← Partition function

where $Z_{\vec{\lambda}} = \sum_{\vec{x}} \exp\left(-E_{\vec{\lambda}}(\vec{x})\right)$

$\vec{x} = (x_1, \dots, x_N)$

$0, 1$

$Z_{\vec{\lambda}}$ ← Partition function

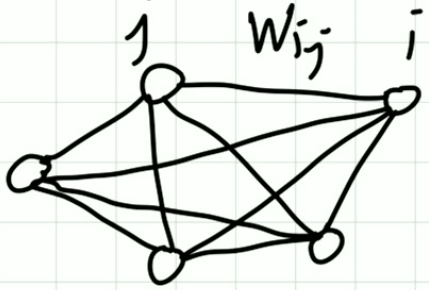
where $Z_{\vec{\lambda}} = \sum_{\vec{x}} \exp\left(-E_{\vec{\lambda}}(\vec{x})\right)$

$\vec{x}$: 2^N

↳ J. Hopfield (1982): "± 1"

$$E_{\vec{\lambda}}(\vec{x}) = \sum_{ij} W_{ij} X_i X_j$$

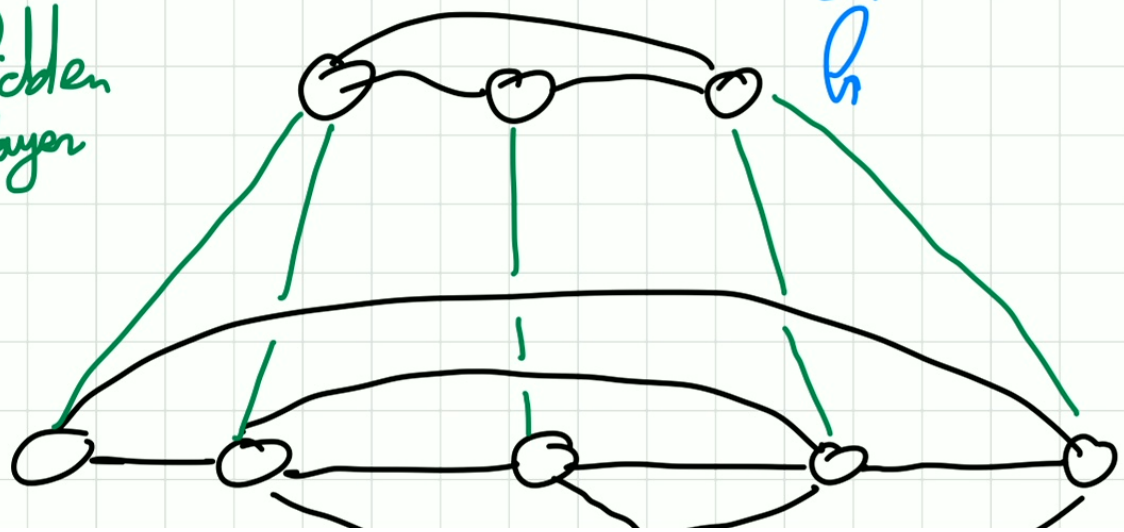
W_{ij} ← Parameters $\in \mathbb{R}$



* Disadvantages:

This motivates → Boltzmann machine (1985): Hinton, Ackley, Sejnowsk
"A learning algorithm for Boltzmann Machines"

could also be ± 1

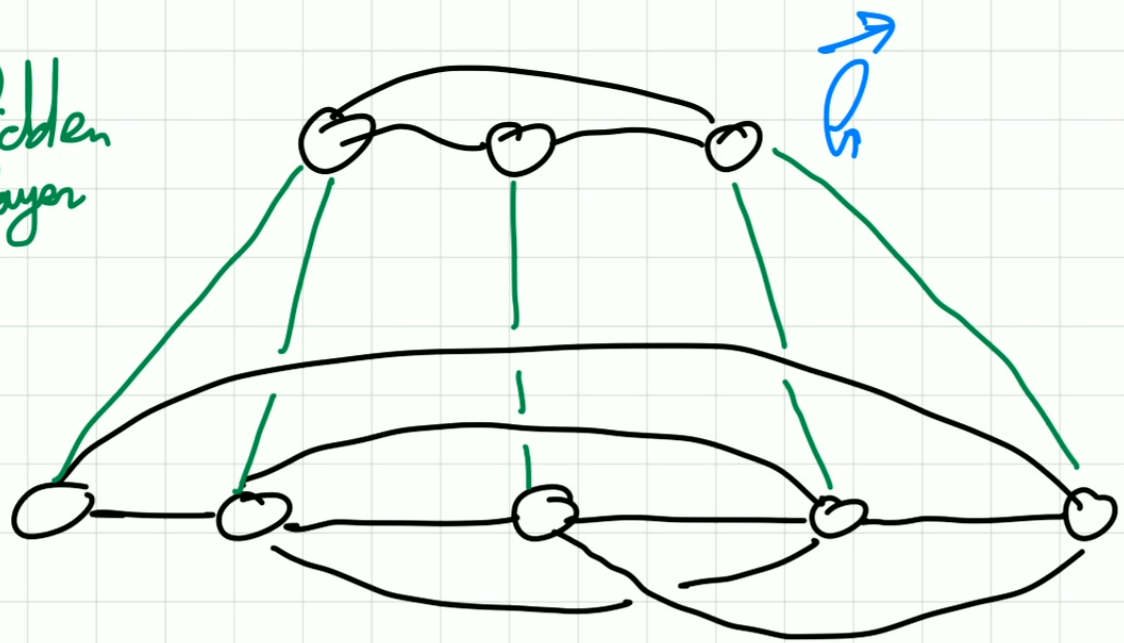


$\vec{h}$

$h_i = 0, 1$

$x_i = 0, 1$

$\vec{x}$



$\vec{h}$

$h_i = 0, 1$

$x_i = 0, 1$

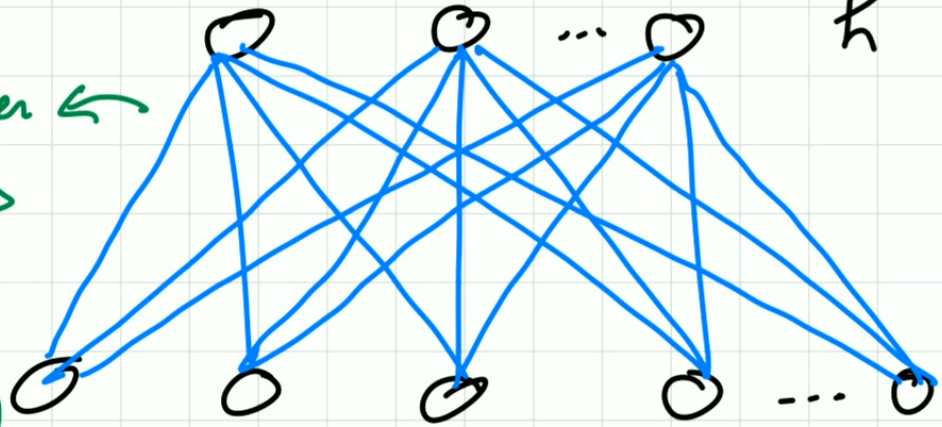
$\vec{x}$

↳ BM is systematically improvable with the increase of the number of hidden variables "$\vec{h}$"

number of hidden variables "$\vec{h}$"

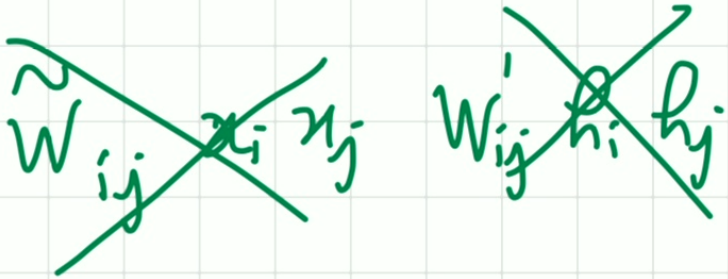
↳ Restricted Boltzmann Machine (RBM) → Also pioneered by Hinton and collaborators

Remove intra-layer connections ↓ For Easier analytical computations



The hidden variables h_1, h_2, h_3.

→ In more details:

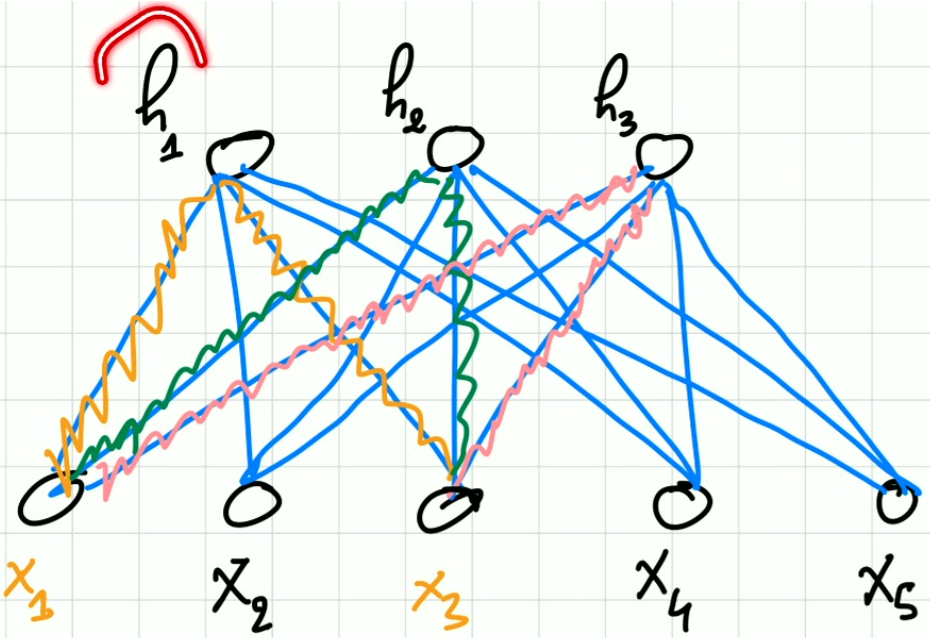


$$E_{\vec{\lambda}}(\vec{x}, \vec{h}) = -\sum_{ij} W_{ij}\, x_i\, h_j - \sum_{i=1}^{N} b_i\, x_i - \sum_{j=1}^{H} c_j\, h_j$$

Hyperparameter $\in \mathbb{R}^+$

and $H = \lfloor \alpha N \rfloor$ (Number of hidden variables)

Main idea:

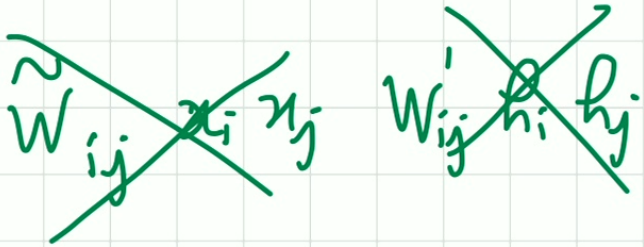


For instance:

↪ Correlation between x_1 and x_3 is introduced through the hidden variables h_1, h_2, h_3.

the hidden variables h_1, h_2, h_3.

→ In more details:



$$E_{\vec{\lambda}}(\vec{x}, \vec{h}) = -\sum_{ij} W_{ij} x_i h_j - \sum_{i=1}^{N} b_i x_i - \sum_{j=1}^{H} c_j h_j$$

and $H = \lfloor \alpha N \rfloor$ (α: Hyperparameter $\in \mathbb{R}^+$)

(Number of hidden variables)

$$P(\vec{x}) = \sum P(\vec{x}, \vec{h})$$

$$E_\lambda(\vec{x}, \vec{h}) = -\sum_{ij} W_{ij} x_i h_j - \sum_{i=1}^{N} b_i x_i - \sum_{j=1}^{H} c_j h_j$$

Hyperparameter $\in \mathbb{R}^+$

and $H = \lfloor \alpha N \rfloor$ (Number of hidden variables)

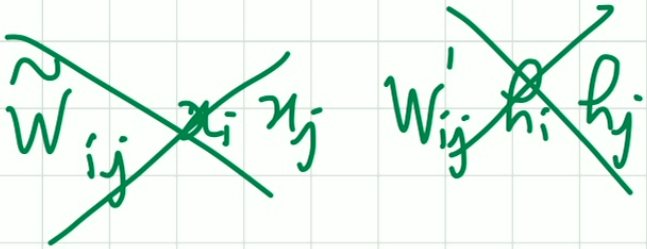
$$P_\lambda(\vec{x}) = \sum_{\vec{h}} P(\vec{x}, \vec{h})$$

Marginalization

$$= \frac{\exp(-E_\lambda(\vec{x}, \vec{h}))}{Z_\lambda}$$

the hidden variables h_1, h_2, h_3.

→ In more details:



$$E_{\vec{\lambda}}(\vec{x}, \vec{h}) = -\sum_{ij} W_{ij}\, x_i\, h_j - \sum_{i=1}^{N} b_i\, x_i - \sum_{j=1}^{H} c_j\, h_j$$

and $H = \lfloor \alpha N \rfloor$ (α: Hyperparameter $\in \mathbb{R}^+$)

(Number of hidden variables)

$Q(\vec{\cdot}) \quad \propto \quad Q(\vec{\cdot}\,|\,\vec{\cdot})$

* Industry applications

→ Recommendation engines (collaborative filtering ~ 2000)

→ Netflix competition (2006)

→ Unsupervised pre-training for deep learning (Pre-2012) AlexNet

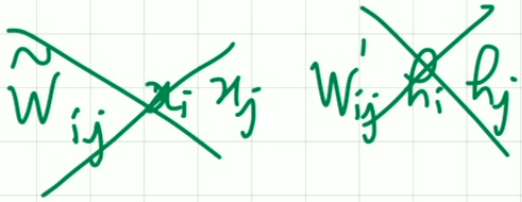
* Many-body Physics applications:

* 1st application: Train RBM on data drawn from $P(\vec{x})$ ← Unknown

So that: $P_{RBM}(\vec{x}) \simeq P(\vec{x})$

↪ After training, we can sample RBM to compute observables "Q"

→ In more details:



$$E_\lambda(\vec{x}, \vec{h}) = -\sum_{ij} W_{ij} x_i h_j - \sum_{i=1}^{N} b_i x_i - \sum_{j=1}^{H} c_j h_j$$

and $H = \lfloor \alpha N \rfloor$ (α: Hyperparameter $\in \mathbb{R}^+$) (Number of hidden variables)

$$P_\lambda(\vec{x}) = \sum_{\vec{h}} P(\vec{x}, \vec{h})$$

Marginalization

$$= \frac{\exp(-E_\lambda(\vec{x}, \vec{h}))}{Z_\lambda}$$

1st application: Train RBM on data drawn from $P(\vec{x})$ (unknown)

so that: $P_{RBM}(\vec{x}) \simeq P(\vec{x})$

↪ After training, we can sample RBM to compute observables "Q"

$$\langle Q \rangle = \sum_{\vec{x}} P_{RBM}(\vec{x})\, Q(\vec{x})$$

$$\langle Q \rangle = \frac{1}{Z_{\vec{x}}} \sum_{\vec{x}} \sum_{\vec{h}} \exp\left(-E_{\vec{x}}(\vec{x},\vec{h})\right) Q(\vec{x})$$

Kullback-Liebler Divergence

$$D_{KL}\left(P \,\|\, P_{RBM}\right) = + \sum_{\vec{x}} P(\vec{x}) \log\left(\frac{P(\vec{x})}{P_{RBM}(\vec{x})}\right)$$

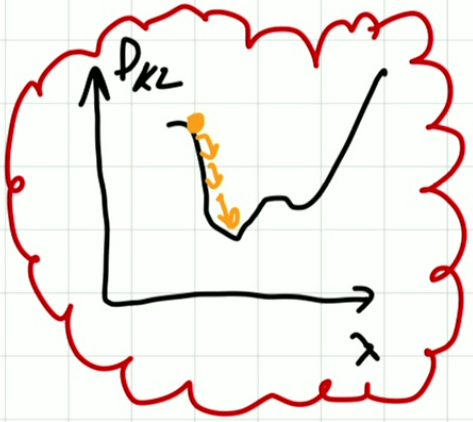
Cost function

→ Property:

$$D_{KL} = 0 \quad \text{iff} \quad P = P_{RBM} \quad \text{and} \quad D_{KL} > 0 \quad \text{(otherwise)}$$

↳ We can perform gradient descent:

$$\lambda \leftarrow \lambda - \eta \partial_\lambda D_{KL}$$



$$D_{KL}\left(P \,\|\, P_{RBM}\right) = + \sum_{\vec{x}} P(\vec{x}) \log\left(\frac{P(\vec{x})}{P_{RBM}(\vec{x})}\right)$$

Cost function

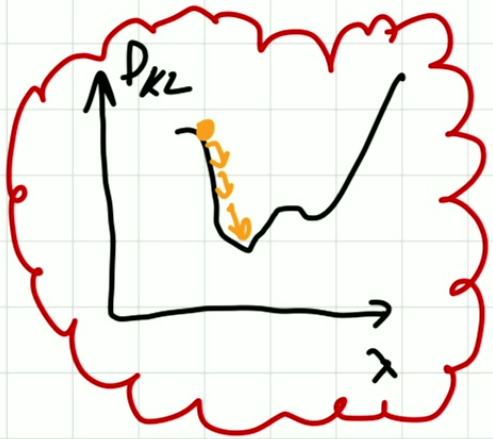
→ Property:

$D_{KL} = 0$ iff $P = P_{RBM}$ and $D_{KL} > 0$ (otherwise)

↳ We can perform gradient descent:

"Adam" optimizer

$$\lambda \leftarrow \lambda - \eta \, \partial_\lambda D_{KL}$$



* Question: Is D_{KL} tractable?

$$D_{KL}(P \| P_{RBM}) = + \sum_{\vec{x}} P(\vec{x}) \log\left(\frac{P(\vec{x})}{P_{RBM}(\vec{x})}\right)$$

Cost function

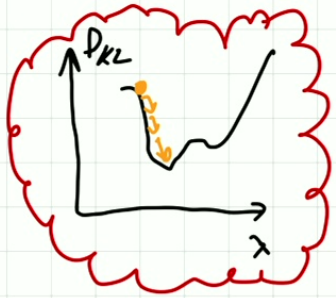
→ Property:

$D_{KL} = 0$ iff $P = P_{RBM}$ and $D_{KL} > 0$ (otherwise)

↳ We can perform gradient descent:

"Adam optimizer"

$$\lambda \leftarrow \lambda - \eta \partial_\lambda D_{KL}$$



* Question: Is D_{KL} tractable?

$$D_{KL}(P \| P_{RBM}) = \underbrace{\sum_{\vec{x}} P(\vec{x}) \log(P(\vec{x}))}_{-H_{data} = \text{- entropy of data}} - \sum P(\vec{x}) \log(P_{RBM}(\vec{x}))$$

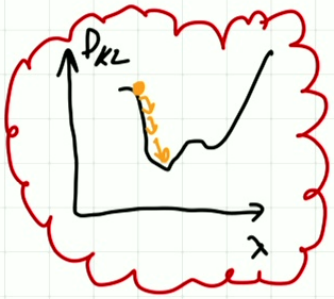
Cost function

$D_{KL} = 0$ iff $P = P_{RBM}$ and $D_{KL} > 0$ (otherwise)

↳ We can perform gradient descent:

"Adam optimizer"

$$\lambda \leftarrow \lambda - \eta \partial_\lambda D_{KL}$$



* Question: Is D_{KL} tractable?

$$D_{KL}(P \| P_{RBM}) = \underbrace{\sum_{\vec{x}} P(\vec{x}) \log(P(\vec{x}))}_{-H_{data} = -\text{entropy of data}}$$

$$- \sum_{\vec{x}} P(\vec{x}) \log(P_{RBM}(\vec{x}))$$

Independent of parameters $\vec{\lambda}$

$$\simeq -H_{data} - \frac{1}{|D|} \sum_{\vec{x} \in D} \log(P_{RBM}(\vec{x}))$$

→ Recommendation engines (Collaborative filtering ~ 2000)

→ Netflix competition (2006)

→ Unsupervised pre-training for deep learning (Pre-2012) AlexNet

* Many-body Physics applications:

*1st application: Train RBM on data drawn from $P(\vec{x})$ (unknown)

So that: $P_{RBM}(\vec{x}) \simeq P(\vec{x})$

↪ After training, we can sample RBM to compute observables "Q"

$$\langle Q \rangle = \sum_{\vec{x}} P_{RBM}(\vec{x})\, Q(\vec{x})$$

$$\langle Q \rangle = \frac{1}{Z_{\vec{x}}} \sum_{\vec{x}} \sum_{\vec{h}} \exp\left(-E_{\lambda}(\vec{x},\vec{h})\right) Q(\vec{x})$$

$p_{RBM}(x)$

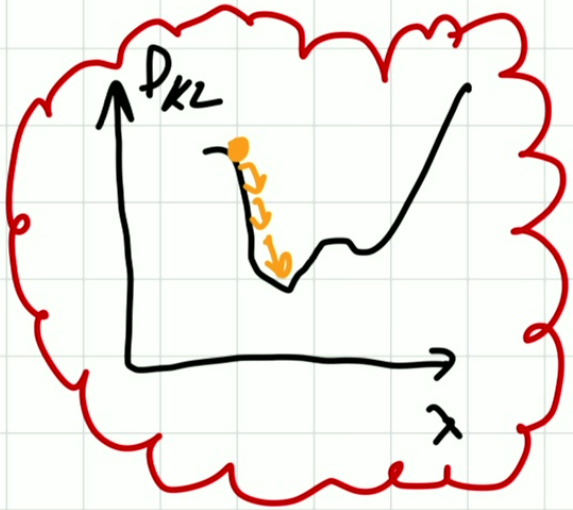
→ Property :

$D_{KL} = 0$ iff $p = p_{RBM}$ and $D_{KL} > 0$ (otherwise)

We can perform gradient descent :

Adam optimizer

$$\lambda \leftarrow \lambda - \eta \partial_\lambda D_{KL}$$



Question : Is D_{KL} tractable ?

- HIL - - Entropy S

$$+ \frac{1}{|D|} \sum_{\vec{x} \in D} \partial_{\cdot} \vec{x} \vec{z}_{\vec{x}}$$

Intractable → Contrastive Divergence (CD)

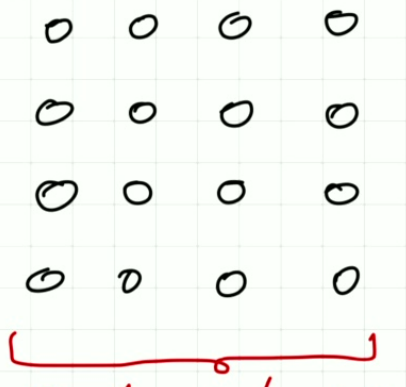
(See additional Notes)

Negative phase

Demo:

→ See tutorial tomorrow (Qucumber for Rydberg atoms)

Example:



Quantum system $|\psi\rangle$ (e.g. Rydberg atoms)

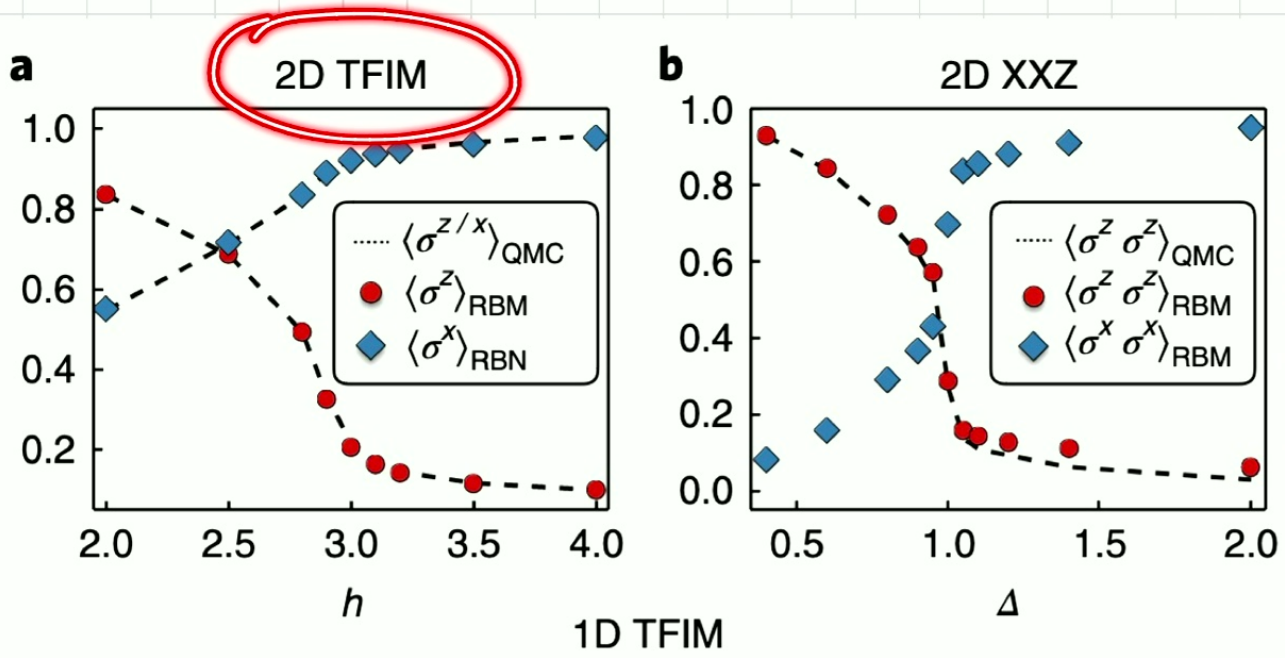
Multiple measurements →

Drawn from $P = |\psi|^2$

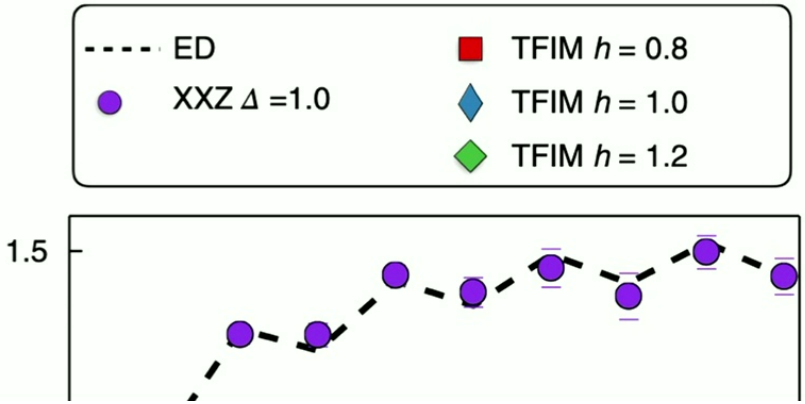
$D = \{\vec{x}\}$

Train an RBM

$$\Psi_{RBM} \simeq \Psi = \sqrt{P} e^{i\phi}$$



See: "Neural Network quantum state tomography",



* 2nd application: (Variational Monte Carlo - VMC)

→ Given a quantum system described by Hamiltonian $\hat{H}$ (e.g. $\hat{H} = -\sum_i \hat{\sigma}_i^z \hat{\sigma}_{i+1}^z - B_x \sum_i \hat{\sigma}_i^x$)

* We want to approximate the ground state: $|\Psi_G\rangle = \begin{pmatrix} \vdots \\ \vdots \\ \vdots \end{pmatrix} \in \mathbb{C}^{2^N}$

$$\hat{H}|\Psi_G\rangle = E_G|\Psi_G\rangle$$

↳ We can use an RBM with

one could use complex-valued parameters

Tractable but not normalized ←

$$\Psi_{RBM}(\vec{x}) = \sum_{\vec{h}} e^{-E_{\vec{x}}(\vec{x},\vec{h})}$$

$$\langle \Psi | \hat{H} | \Psi \rangle$$

↳ We can use an RBM with

$$\psi_{RBM}(\vec{x}) = \sum_{\vec{h}} e^{-E_{\vec{x}}(\vec{x},\vec{h})}$$

Tractable but not normalized

One could use complex-valued parameters

↳ Train RBM using $\frac{\langle \psi_{RBM} | \hat{H} | \psi_{RBM} \rangle}{\langle \psi_{RBM} | \psi_{RBM} \rangle}$

as a cost function.

See Carleo & Troyer, Science 2017

→ See tutorial with NetKet

https://netket.readthedocs.io/en/v3.12.0/tutorials/gs-heisenberg.html

↳ Open quantum states : Vicentini et al. 2019

arXiv : 1902.10104

↳ "Restricted Boltzmann Machines in Quantum Physics", Nature 2019

* Main limitations : (Motivating RNNs & Transformers)

* Block-Gibbs sampling is not perfect.

Tractable

$$* \quad \partial_{\vec{\lambda}} D_{KL} \simeq -\frac{1}{|D|} \sum_{\vec{x} \in D} \log \left(\sum_{\vec{h}} \exp\left(-E_{\vec{\lambda}}(\vec{x}, \vec{h})\right) \right)$$

Gradients estimation is challenging

$$+ \frac{1}{|D|} \sum_{\vec{x} \in D} \partial_{\vec{\lambda}} \mathcal{Z}_{\vec{\lambda}}$$

Intractable → Contrastive Divergence (CD)