Applications of machine learning to computational physics

Based on arXiv: 1609.09087, 1812.01522, 1712.03893 and works in progress and preliminary
Self-Introduction
Who and what am I?

1. I have been working on lattice gauge theory
   2. U(1) axial anomaly at finite temperature with OV/DW. arXiv:1612.01908, …

2. Machine learning (today’s topic, some of them are on-going)
   3. QCD Spectral function

Deep learning and physics
(Textbook in Japanese)
Now translating
1. Machine learning?
2. Detection of phase transition
3. Configuration generation (work in progress)
4. QCD spectral function (work in progress)
5. Summary
Machine learning?
What is machine learning?
A way of theoretical high-energy physics

“Unknown theory”
generate
“mimic”

Data, input

Prediction (outside of data)

Determine parameters

The standard model

A.I. FOR NUCLEAR PHYSICS

Application of ML to physics

https://www.usqcd.org/computing.html
What is machine learning?
In physics language, modeling and prediction

"Unknown theory"
generate

"mimic"

Data, input

"Prediction" (outside of data)

Determine parameters

Neural network (or some model)

results by style-GAN
**Example: neural network?**

Fit ansatz with multi-nested linear/non-linear func. with parameters

**Input**

\[ \vec{x} = \begin{pmatrix} 0.1 \\ 0.0 \\ 0.5 \\ \vdots \end{pmatrix} \]

**Flatten**

\[ \vec{y}_{\text{ans}} = \begin{pmatrix} 1,0^T \\ 0,1^T \end{pmatrix} \]

**Answers**

Dog, Cat

Regard as a vector

\[ \vec{y}_{\text{ans}} \]

**Neural network is a parametrized non-linear map between two vector space**
What is the neural network?

Fit ansatz with multi-nested linear/non-linear func. with parameters

<table>
<thead>
<tr>
<th>Input</th>
<th>Flatten</th>
<th>$\vec{x} = \begin{pmatrix} 0.1 \ 0.0 \ 0.5 \ \vdots \end{pmatrix}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>vector</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Answers</th>
<th>Dog as a Vector</th>
<th>Dog</th>
<th>Cat</th>
<th>$(1,0)^T$</th>
<th>$(0,1)^T$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$\vec{y}_{\text{ans}}$</td>
<td>vector</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dog</td>
<td>Cat</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Neural network</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\vec{x}$</td>
</tr>
<tr>
<td>$\vec{f}_\theta(\vec{x})$</td>
</tr>
</tbody>
</table>

$\vec{f}_\theta(\vec{x}) = \sigma(W_3\sigma(W_2\sigma(W_1\sigma(W_0 \vec{x} + \vec{b}_0) + \vec{b}_1) + \vec{b}_2) + \vec{b}_3)$

$\sigma(\vec{v})$: element-wise nonlinear function (eg tanh), “activation”
**What is the neural network?**

Fit ansatz with multi-nested linear/non-linear func. with parameters

<table>
<thead>
<tr>
<th>Input</th>
<th>Flatten</th>
<th>$\vec{x} = \begin{pmatrix} 0.1 \ 0.0 \ 0.5 \ \vdots \end{pmatrix}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>vector</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Answers</th>
<th>Dog</th>
<th>Regard as a Vector</th>
<th>Cat</th>
<th>$(1,0)^T$</th>
<th>$(0,1)^T$</th>
<th>$\vec{y}_{ans}$</th>
<th>vector</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Dog</td>
<td></td>
<td>Cat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Neural network**

$\vec{x}$

$\vec{y}_{ans}$

Flow of data

Parameters

$W_i$ (matrix)

$\vec{b}_i$

$\theta$

$\vec{f}_\theta(\vec{x}) = \sigma(W_3\sigma(W_2\sigma(W_1\sigma(W_0\vec{x} + \vec{b}_0) + \vec{b}_1) + \vec{b}_2) + \vec{b}_3)$

$\sigma(\vec{v})$: element-wise nonlinear function (eg tanh), “activation”

“Training” = optimization

Minimize “distance” between $\vec{f}_\theta(\vec{x})$ and $\vec{y}_{ans}$ for data in dataset by tuning $\theta$
1. Detection of phase transition

arXiv: 1609.09087 (w/ A. Tanaka),
1812.01522 (w/ K. Kashiwa, Y. Kikuchi)
Q. Can AI detect phase transition?

With fewer information...

\[ \beta = \frac{1}{T} \]

configurations, temperature

A. YES!
Can “AI” detect phase transition?
Neural net as a thermometer (Classification problem)

Input = Ising configurations (by MCMC) with inverse temperature \( \beta \in (\beta_{\text{min}} < \beta_{\text{cr}} < \beta_{\text{max}}) \)

Output = A class of temperature (Discretized inverse temperature)

NN is trained as a “thermometer”
Input = Ising configurations (by MCMC) with inverse temperature \( \beta \in (\beta_{\text{min}} < \beta_{\text{cr}} < \beta_{\text{max}}) \)

Output = A class of temperature (Discretized inverse temperature)

NN is trained as a “thermometer”

\[
E(y_K) \propto \sum_{j \in \text{Conf}} \delta_{\beta, \beta_{j_{\text{ans}}}} \times (-\log y_K(\{\sigma\}_j)) \quad 0 < y_K < 1
\]

Minimize \( E \), NN becomes a thermometer but we focus on \( W_1 \)
After training, $W_1$ gets some pattern, especially there is a border around critical temp. From detail analysis, output of $W_0$ is correlated to magnetization.
After training neural networks as a thermometer, it captures phase boundary.

Output of first layer is correlated to magnetization, so second layer gets a pattern.

This framework actually works also for 3-states Potts model (skipped).

If make it deeper with convolution layers to improve the temperature prediction, but the pattern of weights becomes blurred.

Applicability for gauge system? How can we input data?

Cf.
P Shanahan, D Trewartha, W Detmold 1801.05784
S Wetzel, M Scherzer 1705.05582
2. Configuration generation
for gauge theory

arXiv: 1712.03893 (w/ A. Tanaka)
and work in progress (w/ A. Tanaka, Y. Nagai)
Markov chain Monte-Carlo
It enables us to calculate observables

- Quantum fired theories, lattice QCD, are written by very high dimensional integral

\[
\langle O[\phi] \rangle = \frac{1}{Z} \int D\phi e^{-S[\phi]} O[\phi]
\]
Markov chain Monte-Carlo
It enables us to calculate observables

- Quantum fired theories, lattice QCD, are written by very high dimensional integral

\[
\langle O[\phi] \rangle = \frac{1}{Z} \int D\phi e^{-S[\phi]} O[\phi]
\]

\[
D\phi = \prod_{i \in \{\mathbb{Z}/L\}^4} d\phi_i : \text{Very high dimensional integral} \quad \sim 10^{4 \text{ dim}}
\]

Markov chain with

\[
P[\phi] = \frac{1}{Z} e^{-S[\phi]}
\]

We can calculate expectation values by using Markov chain Monte-Carlo!
Markov chain Monte-Carlo
It enables us to calculate observables

- Quantum fired theories, lattice QCD, are written by very high dimensional integral

\[
\langle O[\phi] \rangle = \frac{1}{Z} \int \mathcal{D}\phi e^{-S[\phi]} O[\phi]
\]

\[
\mathcal{D}\phi = \prod_{i \in \{\mathbb{Z}/L\}^4} d\phi_i
\]

\[
= \frac{1}{N} \sum_{k}^{N} O[\phi_k] \pm O\left(\frac{1}{\sqrt{N}}\right)
\]

Markov chain Monte-Carlo with \( P[\phi] = \frac{1}{Z} e^{-S[\phi]} \)

- It is difficult to estimate expectation values using simple numerical integral like the trapezoid method.
- **Markov chain** Monte-Carlo can do it, independent to the dimensionality!
- **If** a system has fermions, cost becomes expensive…
- We make this cheaper via “self-learning algorithm” in lattice gauge theory.
**Exact algorithm is needed**

Self-learning Monte Carlo (SLMC) is exact

SLMC for spin systems

\[
P(S_{k'} | S_k) = \min \left( 1, \frac{e^{-\beta(H[S_{k'}] - H_{\text{eff}}[S_{k'}])}}{e^{-\beta(H[S_k] - H_{\text{eff}}[S_k])}} \right) Q_{\text{eff}}^\theta(S_{k'} | S_k)
\]

Accept/Reject

Corrected by modified Metropolis test

Proposing part

\[\theta: \text{tunable parameter = coupling}\]

Update using effective model

this must satisfy detailed balance

This is an exact algorithm: It gives correct configurations and if the effective model is far from the target system, acceptance is zero.

Intuitively, Self-learning MC = Metropolis + reweighting on-fly + update with tunable param $\theta$.

Other possibility: FLOW based model (M. S. Albergo et al.1904.12072)
Exact algorithm is needed
Self-learning Monte Carlo (SLMC)

SLMC for spin systems

\[ P(S_k' | S_k) = \min \left( 1, \frac{e^{-\beta(H(S_k') - H_{eff}^{\theta}[S_k'])}}{e^{-\beta(H(S_k) - H_{eff}^{\theta}[S_k])}} \right) \]

Accept/Reject

Proposing part

\[ \theta : \text{tunable parameter} = \text{coupling} \]

Corrected by modified Metropolis test

Update using effective model
this must satisfy detailed balance

This is an exact algorithm:
if the effective model far from the system, acceptance is zero.

Testcase

\[ H = -J \sum_{\langle ij \rangle} S_i S_j - K \sum_{ijkl} S_i S_j S_k S_l, \]

\[ H_{eff} = E_0 - \tilde{J}_1 \sum_{\langle ij \rangle_1} S_i S_j, \]

\[ S_i = \pm 1 \]

No “efficient” update because of 2nd term

Ising model with parameter \( \tilde{J}_1 \)
, which is determined by fitting!
(no fancy ML is needed!)

This has effective update
Exact algorithm is needed
Self-learning Monte Carlo (SLMC)

SLMC for spin systems

\[
P(S'_k | S_k) = \min \left( 1, \frac{e^{-\beta(H[S'_k] - H^\theta_{\text{eff}}[S'_k])}}{e^{-\beta(H[S_k] - H^\theta_{\text{eff}}[S_k])}} \right) Q^\theta_{\text{eff}}(S'_k | S_k)
\]

This is an exact algorithm.

Testcase

\[
H = -J \sum_{\langle ij \rangle} S_i S_j - K \sum_{ijkl \in \square} S_i S_j S_k S_l, \quad H_{\text{eff}} = E_0 - \tilde{J}_1 \sum_{\langle ij \rangle_1} S_i S_j \quad S_i = \pm 1
\]

J.Liu, Y.Qi, Z.Meng, L.Fu (arXiv:1610.03137)

Accept/Reject

Proposing part

\[ \theta : \text{tunable parameter} = \text{coupling} \]

Corrected by modified Metropolis test

Update using effective model this must satisfy detailed balance

Autocorrelation function

Dynamic Critical exponent

24 time efficient

Very mild scaling

A.I. FOR NUCLEAR PHYSICS

Application of ML to physics
Exact algorithm is needed
QCD with Self-learning Monte Carlo

work in progress

SLMC for lattice QCD

$$P(U_k' \mid U_k) = \min \left( 1, \frac{e^{-(S[U_k] - S_{\text{eff}}^\theta[U_k'])}}{e^{-(S[U_k] - S_{\text{eff}}^\theta[U_k])}} \right) Q_{\text{eff}}^\theta(U_k' \mid U_k)$$

Setup: SU(2) plaquette action + staggered quarks with $m = 0.5$
Effective action = hopping parameter expanded action = pure-gluonic, heatbath

$$S[U] = \beta_{\text{pl}} \begin{array}{c} \square \end{array} + \bar{\psi}(D_{\text{stag}} + m)\psi$$

Our choice: $$S_{\text{eff}}[U] = \tilde{\beta}_{\text{pl}} \begin{array}{c} \square \end{array} + \tilde{\beta}_{\text{rec}} \begin{array}{c} \square \end{array} + \tilde{\beta}_* \begin{array}{c} \triangle \end{array} + \cdots$$

Parameters determined by HMC with linear regression
or we can use SLMC (“self-learning” way of use)

Collaborate with
Akinori Tanaka (Riken AIP/ iTHENS)
Yuki Nagai (JAEA/ RIKEN AIP)
Preliminary result
QCD with Self-learning Monte Carlo

work in progress

SLMC for lattice QCD

Collaborate with
Akinori Tanaka (Riken AIP/ iTHENS)
Yuki Nagai (JAEA/ RIKEN AIP)

\[
P(U_{k'} | U_k) = \min \left( 1, \frac{e^{-(S[U_k'] - S_{\text{eff}}^\theta[U_k'])}}{e^{-(S[U_k] - S_{\text{eff}}^\theta[U_k])}} \right) Q_{\text{eff}}^\theta(U_{k'} | U_k)
\]

\[\theta : \text{tunable parameter} = \text{coupling}\]

Setup: SU(2) plaquette action + staggered quarks with ma = 0.5
Effective action = hopping parameter expanded action = pure-gluonic, heatbath

Observables (■=SLMC, □=HMC, ◯=quenched)

ma=0.5, L=4

So far so good
3. QCD spectral function via sparse modeling

Work in progress, very preliminary

A. Tomiya
QCD spectral function?
It contains everything, but we cannot obtain

Two point functions $G(\tau)$ can be calculate on the lattice,

$$G(\tau) = \langle O(\tau) O^\dagger(0) \rangle.$$ 

Corresponding QCD spectral function $\rho(\omega)$ contains every information of QCD for that channel,

$$G(\tau) = \int d\omega K(\tau, \omega) \rho(\omega) \quad \bar{K}(\tau, \omega) \sim \cosh: \text{kernel}$$

Practically, we can not obtain $\rho$ because, 

Discretize 

$$G_\tau = \sum_\omega K_{\tau, \omega} \rho_\omega$$

Lack of information to determine $\rho$ from $G$ (“ill-posed problem”)

Maxima entropy method (MEM; Asakawa et al.) has been used = Bayesian analysis
Sparse Modeling?

Fitting with L1 regulator (=LASSO, least absolute shrinkage and selection operator)

Apply the singular value decomposition (also used in MEM),

\[
K = USV^\top \\
U, V \quad \text{Orthogonal (unitary) mat.} \\
S \quad \text{Rectangular diagonal matrix}
\]

\[
\begin{cases}
\tilde{G} = U^\top G \\
\tilde{\rho} = V^\top \rho
\end{cases}
\]

\[
\tilde{G}_\tau = S_{\tau,\omega} \tilde{\rho}_\omega
\]

\[
\chi^2(\tilde{\rho}) = |\tilde{G} - S\tilde{\rho}|_2^2
\]

 Modify

\[
L(\tilde{\rho}; \lambda) = |\tilde{G} - S\tilde{\rho}|_2^2 + \lambda |\tilde{\rho}|_1
\]

And minimize this L.

Naive chi square

Overfit the noisy data

Chi square with L1 regulator (LASSO) using Lagrange mult.

A.I. FOR NUCLEAR PHYSICS

Application of ML to physics
**Why L1 regulator works?**

L1 regulator can kill ambiguities well = sparseness

**Example:**

Given Eq.

\[(a_1\ a_2)\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \text{constant}\]

w/ L2 constraint

\[
\min_{x_i} \| \vec{x} \|_2
\]

Definition

\[
\| \vec{x} \|_2 = \sqrt{x_1^2 + x_2^2}
\]

Solution:

x1\neq0, x2\neq0

w/ L1 constraint

\[
\min_{x_i} \| \vec{x} \|_1
\]

\[
\| \vec{x} \|_1 = |x_1| + |x_2|
\]

Even if the equation has statistical noise, a solution with L1 constraint is robust (insensitive to noise)
**Original Problem:**

We want to determine $O(1000)$ points of $\rho$ from $O(10)$ data.

This means, $O(10)$ points of $\rho$ in some basis can be determined because we don’t have information.

BUT, in SVD basis, the spectral function is sparse!

We can model the data as:

$$ L(\tilde{\rho}; \lambda) = |\tilde{G} - S\tilde{\rho}|^2 + \lambda |\tilde{\rho}| $$

and,

$$ \rho = V\tilde{\rho} $$

(In practice, we add positivity constraint for $\rho$ to $L$.)
Mock data (vector ch.) from PRD65, 014501(CP-PACS), noise level from Asakawa et al.

\[
\rho_{in}(\omega) = \frac{2}{\pi} \frac{\Gamma_\rho(\omega)m_\rho}{\rho(\omega^2 - m_\rho^2) + \Gamma_\rho(\omega)m_\rho^2} + \frac{1}{8\pi} \left( 1 + \frac{\alpha_s}{\pi} \right) \frac{1}{1 + e^{(\omega_0 - \omega)/\delta}}.
\]

\[
\Gamma_\rho(\omega) = \frac{1}{48\pi} \frac{m_\rho^3}{F_\rho^2} \left( 1 - \frac{4m_\pi^2}{\omega_0^2} \right)^{3/2} \theta(\omega - 2m_\pi).
\]

\( m_\rho = 0.77, \quad m_\pi = 0.14, \quad F_\rho = 0.142, \quad \omega_0 = 1.3, \quad \delta = 0.2, \quad \alpha_s = 0.3, \)

\[
|\rho_{rec}(\omega) - \rho_{mock}(\omega)|_2 = 0.1071
\]
Summary

Machine learning provides us new techniques

1. Neural network can detect phase transition in classical spin chain

2. SLMC can generate meaningful gauge configurations

3. Sparse modeling can reconstruct QCD spectral function (for mock data though)

Todo:

5. Application of SLMC to physical system; right top corner of the Columbia plot

6. Improve SLMC by adding more and more terms, neural net may help + extend to SU(3)

7. More test on sparse modeling and apply to real lattice QCD data

Comment:  
- Our community should discuss systematic error from ML techniques, if it is not exact (Benefit of LQCD is quantitativity).
- How can we control or evaluate error?

Thanks!
Backup
Output of first layer ~ Magnetization

Scatter plot for output of 1st layer and magnetization

-> x is correlated to the magnetization
(automatically captured)

This means, $W_1$ represents correlation between temperature and magnetization!
Markov chain Monte-Carlo

It has inefficiency from correlation between samples

\[
\langle O[\phi] \rangle = \frac{1}{N} \sum_{k}^{N} O[\phi_k] \pm O\left(\frac{1}{\sqrt{N_{\text{indep}}}}\right)
\]

\[
N_{\text{indep}} = \frac{N}{2\tau_{ac}}
\]

\[
\bar{\Gamma}(t) = \frac{1}{N-t} \sum_{k} (O[\phi_{k+t}] - \bar{O})(O[\phi_k] - \bar{O}) \sim e^{-t/\tau_{ac}}
\]

\[\phi_1 \rightarrow \phi_2 \rightarrow \phi_3 \rightarrow \ldots\]

\[\Gamma \text{ or } \tau_{ac} \text{ measures similarity of configurations}\]
Effect of long autocorrelation

Autocorrelation makes signals/noise ratio bad

Data from
Nf=3, standard staggered
with magnetic field
$L^3 \times N_t = 16^3 \times 4$
$ma = 0.03$

$\langle O[\phi] \rangle = \frac{1}{N_{\text{conf}}} \sum_{k}^{N_{\text{conf}}} O[\phi_k] \pm O(\frac{1}{\sqrt{N_{\text{indep}}}})$

$\tau_{ac} \sim \xi^z \sim L^z$

$z$: Dynamic critical exponent

If we find an algorithm with smaller $z$ (or shorter $\tau_{ac}$),
it enables us precise/large scale research around the critical regime!
A key concept is the detailed balance condition:

If an update algorithm $P(\cdot|\cdot)$ satisfies

$$P(\phi_{k'} | \phi_k) e^{-S[\phi_k]} = P(\phi_k | \phi_{k'}) e^{-S[\phi_{k'}]}$$

it will give configurations with a desired distribution (skip proof)

$$P_{eq}(\phi) = \frac{1}{\int \mathcal{D}\phi' e^{-S[\phi']} e^{-S[\phi]}}$$
We have to fight for autocorrelation!
Machine Learning techniques could reduce autocorrelation

\[
\langle O[\phi] \rangle = \frac{1}{N} \sum_{k}^{N} O[\phi_k] \pm O\left(\frac{1}{\sqrt{N_{\text{indep}}}}\right)
\]

\[
N_{\text{indep}} = \frac{N}{2\tau_{ac}}
\]

\[
\bar{\Gamma}(t) = \frac{1}{N - t} \sum_{k} (O[\phi_{k+t}] - \bar{O})(O[\phi_k] - \bar{O}) \sim e^{-t/\tau_{ac}}
\]

\[
\tau_{ac} \text{ is given by an update algorithm (N. Madras et. al 1988)}
\]

• Correlation between generated configurations are estimated by autocorrelation time \(\tau_{ac}\)

• Autocorrelation time \(\tau_{ac}\) depends on an update algorithm

• If \(\tau_{ac}\) becomes half, statistics becomes effectively double in same cost in time!

I attempt to generate configurations using machine learning!
Exact algorithm is needed
QCD with Self-learning Monte Carlo

work in progress

Collaborate with
Akinori Tanaka (Riken AIP/ iThems)
Yuki Nagai (JAEA/ RIKEN AIP)

\[
P(U_{k'} | U_k) = \min \left( 1, \frac{e^{-(S[U_{k'}] - S_{\text{eff}}[U_{k'}])}}{e^{-(S[U_k] - S_{\text{eff}}[U_k])}} \right) Q_{\text{eff}}(U_{k'} | U_k)
\]

Setup: SU(2) plaquette action + staggered quarks
Effective action = hopping parameter expanded action, heatbath

Autocorrelation (□=HMC, □=SLMC)

Also good
Exact algorithm is needed
QCD with Self-learning Monte Carlo

work in progress

Collaborate with
Akinori Tanaka (Riken AIP/ iThems)
Yuki Nagai (JAEA/ RIKEN AIP)

\[ P(U_{k'} | U_k) = \min \left( 1, \frac{e^{-(S[U_{k'}]-S_{\text{eff}}[U_{k'}])}}{e^{-(S[U_k]-S_{\text{eff}}[U_k])}} \right) Q_{\text{eff}}(U_{k'} | U_k) \]

Setup: SU(2) plaquette action + staggered quarks
Effective action = hopping parameter expanded action, heatbath

Acceptance

HMC : 90%  26
SLMC : 60%  1

Bad news: now it works only for \( m > 1 \)
Stay tuned
QCD spectral function?
QCD with Self-learning Monte Carlo

\[ G(\tau) = \int d\omega \tilde{K}(\tau, \omega) \rho(\omega) \]

\[ \tilde{K}(\tau, \omega) = \omega^2 (e^{-\tau \omega} + e^{-(N\tau - \tau) \omega}) \]

\[ G_\tau = \sum_\omega \tilde{K}_{\tau, \omega} \rho_\omega \]

\[ O(10) \]

\[ \tilde{K}_{\tau, \omega} \cdot \rho_\omega \]

\[ O(10^3) \]
Introduction

What does “Machine learning” give us?

What supervised learning does (~ deep learning, neural nets):
What supervised learning does (~ deep learning, neural nets):

"White box" = "model"
- Parametrized function (neural nets), a fit function

\[ f_\theta(x) \]
\[ \theta = \{ \text{parameters} \} \]
Introduction

What does “Machine learning” give us?

What supervised learning does (~ deep learning, neural nets):

“White box” = “model”
- Parametrized function (neural nets), a fit function

\[ f_\theta(x) \]

\[ \theta = \{ \text{parameters} \} \]

\[ D_{KL}(\theta) \geq 0 \] : “Distance between correct answer to current output”
(cf: chi square in fitting, \( \theta \) = a set of parameters)

- Machine learning (\( \geq \) deep learning) basically, make a “map” between data and output
- It could be deterministic (neural nets) or stochastic (generative models, later)
Unsupervised learning in a nutshell

Information comes from a probability distribution

\[ \partial_\theta D_\theta(P \parallel P_\theta) = -\int \mathcal{D}x P(x) \partial_\theta \log(P_\theta(x)) \]

\[ P_\theta[x] = \frac{1}{Z_\theta} e^{-H_\theta[x]} \]

\[ = -\int \mathcal{D}x P(x) \partial_\theta (-H_\theta[x] - \log Z_\theta) \]

\[ = \int \mathcal{D}x P(x) (\partial_\theta H_\theta[x] + \frac{1}{Z_\theta} (\int \mathcal{D}y \partial_\theta e^{-H_\theta[y]})) \]

\[ = \int \mathcal{D}x P(x) \partial_\theta H_\theta[x] - \frac{1}{Z_\theta} \int \mathcal{D}y H_\theta[y] e^{-H_\theta[y]} \]
Unsupervised learning in a nutshell
Inverse of Ising model ~ Boltzmann machine

Eg. Generalized Ising model
   (spin glass)

\[ H_{\theta}[^{\sigma}] = \beta \sum K_{ij}^{\sigma_i \sigma_j} \]

\[ \sigma_i: \text{spin} \]
\[ K_{ij}: \text{coupling} \]

What physicists want to do

Boltzmann machine:
   guess a set of parameters \( \theta \)

\[ \langle O[^{\sigma}] \rangle_{\theta} = \frac{1}{Z_{\theta}} \sum_{\{\sigma\}} e^{-H_{\theta}[\sigma]} \]

\[ \theta = \{K_{ij}, \beta\} \text{ parameters} \]
Unsupervised learning in a nutshell

Unsupervised

Eg. Generalized Ising model (spin glass)

\[ P_\theta[\sigma] = \frac{1}{Z_\theta} \exp(-H_\theta[\sigma]) \]

\( P_\theta[\sigma] \): probability of spin configuration

\( Z_\theta \): partition function

\( H_\theta[\sigma] \): Hamiltonian

\( \sigma_i \): spin

\( K_{ij} \): parameters

Generative models fit (guess) the coupling

Once determine a coupling \( K (= \text{training}) \), we can use it.

Sampling from parametrized distribution is needed.