Deep learning in Lattice 1+1d Scalar Field Theory

Kai Zhou 周凯 (FIAS, Frankfurt)

In collaboration with:
Gergely Endrődi (ITP, Frankfurt)
LongGang Pang (UC Berkeley, CCNU)
Horst Stoecker (ITP, FIAS, Frankfurt)

Phys. Rev. D100 (2019) no.1, 011501
Introduction

- Convolutional Neural Network has proved to be extremely powerful in Pattern Recognition, Image Classification

- Discriminative learning (prediction): Classification, Regression
- Generative modelling (generation): RBM, VAE, GAN
1+1d $\lambda \phi^4$ field (prepare training set)

regularization of continuum **Action:**

$$S^{\text{lat}} = \sum_x \left\{ (4 + m^2) \phi^*(x) \phi(x) + \lambda [\phi^*(x) \phi(x)]^2 - \sum_{\nu=1,2} \left[ e^{\mu \delta_{\nu,2} \phi^*(x) + \nu} + e^{-\mu \delta_{\nu,2} \phi^*(x) \phi(x - \nu)} \right] \right\}$$

**Partition sum:**

$$Z = \int D[\phi] \exp \left( -S^{\text{lat}} [\phi] \right)$$

**Dualization approach:**

$$Z = \sum_{\{k,\ell\}} \prod_n \left\{ e^{\mu k_{\ell}(n)} \cdot W[s(n)] \cdot \delta[\nabla \cdot k(n)] \cdot \prod_{\nu} A[k_{\nu}(x), \ell_{\nu}(x)] \right\}$$

$$W[s(n)] = \int_0^\infty dr \, r^{s(n)+1} e^{-(4+m^2)r^2 - \lambda r^4}$$

$$s(n) = \sum_{\nu} \left[ |k_{\nu}(n)| + |k_{\nu}(n - \nu)| + 2(\ell_{\nu}(n) + \ell_{\nu}(n - \nu)) \right]$$

$$A[k_{\nu}(x), \ell_{\nu}(x)] = \frac{1}{(\ell_{\nu}(n) + |k_{\nu}(n)|)! \ell_{\nu}(n)!}$$


*O. Orasch* and *C. Gattringer*, Int. J.Mod.Phys.A33(2016) no.01,1650010,
Configurations - Dualization approach

Configurations - 4 integer-valued variables: $k_t$, $k_x$, $l_t$, $l_x$

$N_t = 200$
$N_x = 10$
$m = 0.1$
$\lambda = 1.0$

Divergence constraint:

$$\nabla \cdot k(n) = \sum [k_{\nu}(n) - k_{\nu}(n - \nu)] = 0$$
Observables: $n$ and $|\phi|^2$

**Grand canonical ensemble**

$$\langle n \rangle = \frac{T}{L} \frac{\partial \log Z}{\partial \mu}$$

$$n = \frac{1}{N_x N_t a} \sum_n k_t(n)$$

$$\langle |\phi|^2 \rangle = \frac{T}{L} \frac{\partial \log Z}{\partial (m^2)}$$

$$|\phi|^2 = \frac{1}{N_x N_t} \sum_n \frac{W[s(n) + 2]}{W[s(n)]}$$

**Condensation sets in at**

$$\mu_{th} \sim m_{phys} \sim 0.94$$
(1) Classification: detect 'phase transition' status based on configs (identify \textit{order parameter})

(2) Regression: physical observables regression (identify \textit{thermodynamics})

(3) GAN(generate): Learn to generate new configurations
Generate configs with proper \textit{distribution} (identify \textit{partition function})
Training set consists of two ensembles of configurations @

\[ \mu = 0.91 \quad \text{with label} \quad y = (0, 1) \]

and

\[ \mu = 1.05 \quad \text{with label} \quad y = (1, 0) \]
Testing set consists of different ensembles of configurations @ different chemical potential

\[ 0.91 < \mu < 1.05 \]
Strong correlation between $P_{\text{cond}}$ and observables: $n$, squared field.
Ensemble average cond-probability

Classifier of the phases: \( \langle n \rangle = 0 \) \text{ and } \langle n \rangle \neq 0

\[ \mu_{th} (\langle P_{cond} \rangle > 0) \sim \mu_{th} (\langle n \rangle > 0) \]
Ensemble average cond-probability

Classifier of the phases: \( \langle n \rangle = 0 \) and \( \langle n \rangle \neq 0 \)

\[
\mu_{\text{th}}(\langle P_{\text{cond}} \rangle > 0) \sim \mu_{\text{th}}(\langle n \rangle > 0)
\]

\[
n = \frac{1}{N_x N_t a} \sum_{n} k_t(n)
\]
Discard $kt$ information

There's finite correlation between $kt$ and $\ell$ variable, But, No correlation between $kt$ and $k_x$
The same transition point came out, even use only $k_x$!

Beyond conventional analysis:

Indicating existence of novel correlation between $k_x$ and $n$.

$\mu_{th}(\langle P_{\text{cond}} \rangle > 0) \sim \mu_{th}(\langle n \rangle > 0)$
Regression for particle density

Note, for training, only used \( \mu = 0.91 \) and \( \mu = 1.05 \)

\[
n = \frac{1}{N_x N_t a} \sum_n k_t(n)
\]

\( RMSE < 0.003 \)
Regression for squared field $|\phi|^2$

Note, for training, only used $\mu = 0.91$ and $\mu = 1.05$

$$|\phi|^2 = \frac{1}{N_x N_t} \sum_n \frac{W[s(n) + 2]}{W[s(n)]}$$

$$W[s(n)] = \int_0^\infty dr r^{s(n)+1} e^{-(4+m^2)r^2-\lambda r^4}$$

$$s(n) = \sum_\nu \left[ |k_\nu(n)| + |k_\nu(n - \hat{\nu})| + 2(\ell_\nu(n) + \ell_\nu(n - \hat{\nu})$$

$$RMSE < 0.005$$
“What I can not create, I do not understand”
GAN - generate proper configurations

The divergence condition get learned automatically:

'Physical' configs can be generated

\[ \nabla \cdot k(n) = \sum_{\nu} [k_{\nu}(n) - k_{\nu}(n - \hat{\nu})] = 0 \]

Automatically capture the **implicit physical constraint**!
Distribution for observables

Histogram distribution with 1k configurations (after 6k training epochs):

\[
\langle n \rangle_{MC} = 0.580 \\
\langle n \rangle_{GAN} = 0.578 \\
\langle \phi^2 \rangle_{MC} = 0.447 \\
\langle \phi^2 \rangle_{GAN} = 0.449
\]
make GAN conditional on particle density $n$:

We train GAN using one ensemble with $\mu = 1.05$ labeled as well with $n$ (including $n=0.4, 0.5, 0.6, 0.7$)

Once trained, we specify different $n$ values in generation stage
Conditional GAN – control $n$

mean value for $n$ and squared field of generated ensemble is controlled by condition in c-GAN.
Summary

(1) Classification:  *pin down phase 'transition' point*  
   (identify *order parameter*)

(2) Regression:  *learn physical observable*  (*non-linear regression*)  
   (identify *thermodynamics*)

(3) GAN(generative): *use GAN to generate*  new physical configuration  
   Capture/reproduce data distribution  (*use for storage*)  
   limited grand canonical  -->  canonical ensemble  
   limited configs.  -->  explore phase diagram  
   (identify *partition function*)

*Needs more exploration!*
Thanks!
DCNN Architecture - Classification

\[ \mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)] + \lambda \|\theta\|_2^2 \]

\[ \alpha_{lr} = 0.0001 \quad \text{With} \quad \text{AdaMax optimization scheme} \]
Condensation probability from CNN

Deleting one CNN layer off gives *slightly worse* distinguish ability
GAN - Nash Equilibrium

Zero-sum game - Nash equilibrium

\[ G^* = \arg \min_G \max_D (-\mathcal{L}_D(G, D)) \]

\[ \mathcal{L}_D = -\mathbb{E}_{\hat{x} \sim p_r(\hat{x})} [\log(D(\hat{x}))] - \mathbb{E}_{z \sim p(z)} [\log(1 - D(G(z)))] \]

\[ \mathcal{L}_G = \mathbb{E}_{z \sim p(z)} [\log(1 - D(G(z)))] \]

\[ D^*(\hat{x}) = \frac{p_r(\hat{x})}{p_r(\hat{x}) + p_g(\hat{x})} \]
GAN - distribution

```
0.4 0.5 0.6 0.7 0.8 0.9 1.0 1.1
```

```
0.4 0.45 0.5 0.55 0.6 0.65
```

GAN results
'ss.dat' u 1:2

QM 2019, Wuhan

05 Nov, Kai Zhou