CS230

Deep Learning Insights into Many-Body Physics

Chao Wang and Mae Teo
Department of Physics
Stanford University
cwang15@stanford.edu, maehwee@stanford.edu

Abstract

The problem of identification of phases of matter in quantum many-body physics can be formulated as a classification problem and addressed using deep learning techniques. In this project, we explore this task using three architectures: logistic regression, convolutional neural network (CNN), and tensor-train (TT) regression [1]. We consider two types of quantum phase transitions, one type from a valence-bond solid (VBS) to anti-ferromagnet (AF), and another type from a quantum spin liquid (QSL) to AF. For the first type, both CNN and TT regression both work well in identifying the location of the phase transition; for the second type, none of the architectures employed perform satisfactorily, presumably due to the subtle topological nature of QSL (correlations are very non-local).

1 Introduction

Many-body physics is a thriving subfield of physics that aims to study complex systems. Even though each individual constituent in a system is described easily by quantum mechanics, the whole system can exhibit emergent collective behavior that cannot be explained by simply considering the sum of its parts. Because of its complexity, most many-body physics problems do not have analytic solutions. In the past few decades there has been a growth in the use of quantum Monte Carlo (QMC) techniques, which are simulations that can provide a close approximation of the true solution. However, analyzing datasets generated from QMC simulations can be a challenging task - which invites the use of deep learning.

In physics, phases of matter are defined by the concept of adiabatic continuity. We say two systems belong to the same phase of matter if there is a path connecting the two systems in the abstract "system-space" (more precisely, the space of all possible Hamiltonians) and that the path does not include any singular point (non-analyticities). In this project, we will consider a family of systems (Hamiltonians), which consist of strongly interacting electrons that is susceptible to a variety of magnetically ordered phases as well as a phase that does not order at all even at zero-temperature. (By "ordered" we mean broken symmetries.) The system lives on a two-dimensional square lattice.

As we tune the parameters that parametrize the family of Hamiltonians, we could cross the boundary between different phases and encounter a phase transition. One example of such a parameter is h, known as the transverse field.

At certain fixed values of the other parameters, one expects a phase transition between one phase (Anti-Ferromagnetic, AF) to another (Valence Bond Solid, VBS) (see Fig. 1) when the value of h crosses some critical value h_c of h.

At another fixed set values of the other parameters, one encounters a phase transition between AF and a very exotic and subtle phase of matter called quantum spin liquid (QSL). QSL breaks no symmetries

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

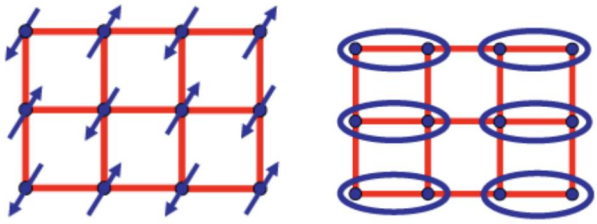

Figure 1: (Left) an Anti-Ferromagnetic State (AFS), where each site's electron alternates in spin direction, (right) a Valence Bond Solid (VBS) state, where each circled pair of electrons have strongly anti-correlated spins. Images from: http://qpt.physics.harvard.edu/peierls.html

of the Hamiltonian even at zero temperature, and moreover, it is a topologically ordered state with fractional excitations (for instance, an electron is fractionalized into a spinon and a chargon).

Our task is to use three different machine-learning architectures to identify different phases and phase transitions of this system, similar to [2]. We obtained our dataset from determinant QMC simulations.

For each fixed Hamiltonian (all parameters fixed), the QMC generates a sequences of Monte Carlo (MC) samples, which taken altogether, constitute an approximation of the true probability distribution of the configurations of the system. Each MC sample will include a matrix of complex numbers, G_{ij} (a complex-valued Green's function that captures some information about the quantum-correlation between sites i and j of the 2-dimensional square lattice).

We formulate the idenfication of phase transtion as a binary classification problem. Fixing all parameters that describe the system except for h, we first consider two extreme values h_1 and h_2 for h, such that we know that the system belongs to two different phases for h_1 and h_2. We then label all MC samples corresponding to h_1 as "0", and all MC samples corresponding to h_2 as "1". These labelled examples will be used for the training of the deep learning architecture, which will then be used to predict the labels on MC samples for many intermediate values of h; then for each intermediate h, we take the average of the labels for its MC samples and use that as a proxy for the phase identificaiton. By looking at how this proxy changes as a function of h we hope to identify the critical value h_c where a phase transition occurs.

2 Related work

The idea of using machine learning to study the problem of phase transitions is very new, but we are certainly not the ones who first proposed it. The case of classical (non-quantum) phase transitions was first by [3]. However, the problem of quantum phase recognition is by its physical nature much harder, especially for problems involving fermions. [4, 2] provided the first examples of using machine learning to study the interacting fermion problem using determinant quantum Monte Carlo dataset, and we have adopted their approaches of using CNNs in this project.

In addition to CNNs, we have also considered the tensor train (TT) architecture, for two reasons: first, TT is a powerful data compression architecture; second, TT, known as matrix product states (MPS) in quantum mechanics, provides a good class of variational wavefunctions for certain quantum many-body problems. [5, 1, 6] provided similar models for the purpose of image recognition or physics phase recognition problems, which we have made attempts implementing in this project.

3 Dataset and Features

To study the VBS-AF transition, we produced 6000 examples for each value of h shown in Fig. 2, and for each lattice site size of 8×8 and 10×10. For 12×12 lattice site size, we produced about 3000 examples for each h. To study the QSL-AF transition, we produced 4000 examples for $h = 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.4, 0.5, 0.6$ with lattice size 8×8.

For example, for the VBS-AF study, there are 6000 examples for $h = 1.5$ and lattice size 8×8. Each example has shape $(64, 64, 2)$. Because earlier tests have determined that including the second

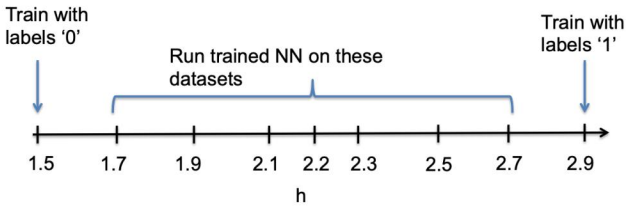

Figure 2: We expect a phase transition at an intermediate value of h.

(imaginary) component of the dataset makes no difference to training results, we only keep the first (real) component, such that each example has shape $(64, 64, 1)$.

Furthermore, we used data augmentation to increase the volume of data four-fold. As shown in Fig.3, each training example was rotated 90, 180 and 270 degrees in real space (i.e. not a rotation of the 64×64 array, but a more subtle transformation that relies on the periodicity of the lattice). Prior tests have shown that training does not work as well without utilizing this rotational symmetry due to idiosyncrasies of each dataset. Furthermore, other tests have shown that normalizing each dataset has no effect on training, so we do not implement it thereafter.

For the tensor train architecture, we further augmented the data by applying a random translation to the real-space lattice, due to the fact that our tensor train regression implementation has not taken into account of any translational symmetries by means of parameter sharing. Clearly the implementation of such parameter sharing would give rise to a much better architecture for what we have used here, and this task we will leave for future studies.

We used 500 examples each from the largest and smallest value of h for validation, and the rest for training.

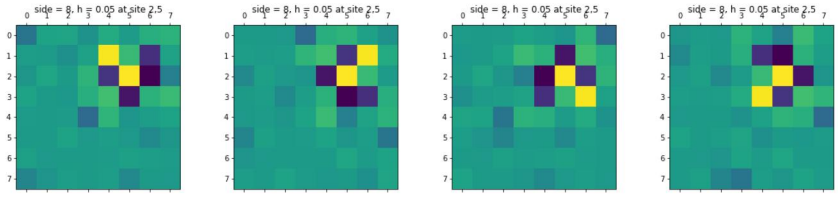

Figure 3: Data augmentation by rotation in the real space. When the data is reshaped to $(8, 8, 8, 8)$, we can visualize the correlation with respect to some chosen lattice point, say $(2, 5)$ by plotting $(2, 5, :, :)$. Here we see that the data augmentation has the physical interpretation of rotating the lattice.

Fig. 4 shows a few training examples from $h = 1.5$ (top row) and $h = 2.9$ (bottom row), where lighter color indicates larger values of the matrix. The leftmost image is the array averaged over all training examples, and the three others are individual training examples chosen at random. Fig.5 correspond to the same examples in Fig. 4, but with data reshaped to $(8, 8, 8, 8)$. We plot $(2, 5, :, :)$ in each case to visualize the correlation with respect to the chosen point $(2, 5)$.

4 Methods

To find the critical value h_c, for the VBS-AF study, we trained $h = 1.5$ with labels '0' and $h = 2.9$ with labels '1'. We used 3 architectures:

1. Logistic regression
2. Convolution Neural Net: One ConvNet layer (5 filters, size 5×5, ReLU) fed to a sigmoid [1]
3. Tensor-Train

[1] Tests done on more elaborate architectures or more filters did not significantly change training results.

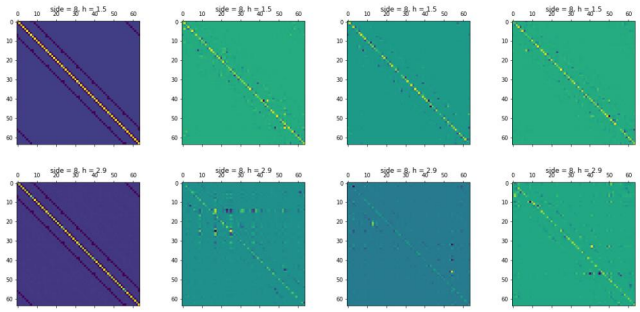

Figure 4: VBS training examples ($h = 1.5$, top row) and AF training examples, ($h = 2.9$, bottom row), leftmost image is the array averaged over training examples in each dataset.

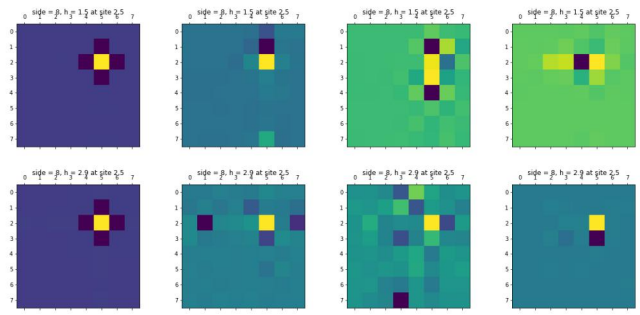

Figure 5: Same as Fig.4 but each array has been reshaped, so as to look at the correlation with respect to the chosen point $(2, 5)$.

In the first two architectures, we used a learning rate of $\alpha = 0.001$, and in all cases used the binary cross-entropy loss.

The tensor-train (TT) regression method considers a somewhat different model [1]. For each Monte Carlo sample, we extract a feature vector $\mathbf{x}$ simply by only keeping the correlations in G_{ij} matrix between sites that are nearest or next-nearest neighbors of each other. The model for TT regression is

$$\hat{y}(\mathbf{x}) = \sum_{i_1=0}^{1} \cdots \sum_{i_d=0}^{1} \mathcal{W}_{i_1 \cdots i_d} \prod_{k=1}^{d} x_k^{i_k} \tag{1}$$

,

where $\hat{y}$ is the prediction, $\mathcal{W}$ is a d-dimensional tensor, and all i_k are either 0 or 1. W has exponentially many entries, and the TT method can be used for compressing it into a form that supports linear in d complexities for both training and evaluation operations. Formally, we say $\mathcal{W}$ is a TT if

$$\mathcal{W}_{i_1 \cdots i_d} = G_1[i_1] \cdots G_d[i_d], \tag{2}$$

for some set of 1-by-r vectors $G_1[i_1]$, r-by-r matrices $G_j[i_j]$ for $j = 2, \cdots, d-1$, and r-by-1 vectors $G_d[i_d]$. r is called the rank of the TT.

This model aims to model interactions within features in any subset of features. The learning of the model corresponds to minimizing loss under the TT-rank constraint [1]:

$$\min_{\mathcal{W}} \mathcal{L}(\mathcal{W}) \tag{3}$$

$$\text{subject to TT-rank}(\mathcal{W}) = r_0, \tag{4}$$

where we have taken the TT-rank $r_0 = 4$, and the loss is given by binary-cross-entropy and a $L-2$ regularization on $\mathcal{W}$.

5 Experiments/Results/Discussion

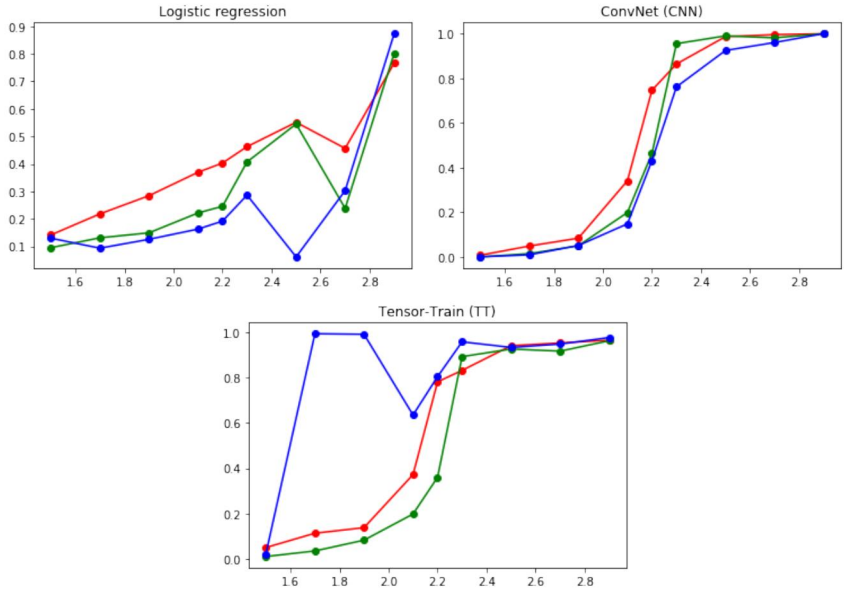

Figure 6: Results: x-axis is h, y-axis is average predicted label. Colors: lattice size 8 (red), 10 (green), 12 (blue).

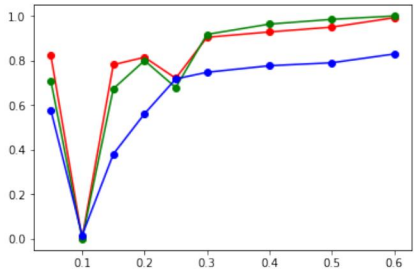

Figure 7: Results: x-axis is h, y-axis is average predicted label. Colors: logistic regression (red), CNN (green), TT (blue).

6 Conclusion and Future Work

Unlike logistic regression, the Convolutional Neural Net and Tensor-Train architecture could (for the most part) predict the location of the VBS-AF phase transition. This demonstrates their usefulness as a tool for extracting features that are not apparent in the dataset. The phase recognition task for the QSL-AF transition fails for all three architectures, likely because QSL is a subtle topological phase with no local correlations, and so identificaiton of topological phases remains an open research question. For tensor train, we should employ parameter sharing to exploit all translational symmetries, which would reduce the number of parameters in the model drastically,

7 Contributions

The codes for our project can be found at https://github.com/maehwee/CS230. Chao obtained all data by determinant QMC simulations, and did all the Tensor-Train tests. Mae obtained the data visualization plots and did the CNN and logistic regression tests, as well as the data augmentation for those tests.

References

[1] Alexander Novikov, Mikhail Trofimov, and Ivan Oseledets. Exponential machines. *arXiv preprint arXiv:1605.03795*, 2016.

[2] Peter Broecker, Fakher F Assaad, and Simon Trebst. Quantum phase recognition via unsupervised machine learning. *arXiv preprint arXiv:1707.00663*, 2017.

[3] Juan Carrasquilla and Roger G Melko. Machine learning phases of matter. *Nature Physics*, 13(5):431, 2017.

[4] Peter Broecker, Juan Carrasquilla, Roger G Melko, and Simon Trebst. Machine learning quantum phases of matter beyond the fermion sign problem. *Scientific reports*, 7(1):8823, 2017.

[5] Alexander Novikov, Dmitrii Podoprikhin, Anton Osokin, and Dmitry P Vetrov. Tensorizing neural networks. In *Advances in neural information processing systems*, pages 442–450, 2015.

[6] Edwin Stoudenmire and David J Schwab. Supervised learning with tensor networks. In *Advances in Neural Information Processing Systems*, pages 4799–4807, 2016.