Proceedings of Machine Learning Research vol 145:, 2021 2nd Annual Conference on Mathematical and Scientific Machine Learning

Ground States of Quantum Many Body Lattice Models via Reinforcement Learning

Willem Gispen WG265@CANTAB.AC.UK and **Austen Lamacraft** AL200@CAM.AC.UK
TCM Group, Cavendish Laboratory, University of Cambridge, J. J. Thomson Ave., Cambridge CB3 0HE, UK

Editors: Joan Bruna, Jan S Hesthaven, Lenka Zdeborova

Abstract

We introduce reinforcement learning (RL) formulations of the problem of finding the ground state of a many-body quantum mechanical model defined on a lattice. We show that stoquastic Hamiltonians – those without a sign problem – have a natural decomposition into stochastic dynamics and a potential representing a reward function. The mapping to RL is developed for both continuous and discrete time, based on a generalized Feynman–Kac formula in the former case and a stochastic representation of the Schrödinger equation in the latter. We discuss the application of this mapping to the neural representation of quantum states, spelling out the advantages over approaches based on direct representation of the wavefunction of the system.

Keywords: Quantum Mechanics, Feynman–Kac Formula, Optimal Control, Reinforcement Learning

1. Introduction

Finding the ground state of a quantum mechanical system involves finding the lowest eigenvalue of the Hamiltonian operator that describes the system. For many body systems, this is a problem that grows exponentially harder as the number of particles increases, and finding tractable approaches has been the principal computational challenge of quantum mechanics since its inception.

When studying a non-relativistic system of electrons and nuclei, a natural starting point is a realistic Hamiltonian in continuous space with Coulomb interactions. Often, however, *model Hamiltonians* are used instead: for example, the singularity of the Coulomb potential describing the electron-ion attraction is often replaced with a less singular *pseudopotential* in quantum chemistry calculations. Another type of simplification, more common in strongly correlated physics, is to study *lattice models* where particle locations are discretized. Although less realistic, this simplification often allows the treatment of larger systems where qualitative different thermodynamic phases become apparent.

In recent years, progress in deep learning has opened up new fronts to attack the many body problem by parameterizing the wavefunction describing the ground state using neural networks (e.g. Carleo and Troyer (2017); Choo et al. (2019); Pfau et al. (2019)). In this work we introduce a new neural approach to lattice models using an alternative formulation based on Reinforcement Learning (RL), in which the optimal policy matches a stochastic process describing the ground state probability distribution. Barr et al. (2020) applied the same perspective to continuum models using the Feynman–Kac formula and tools of stochastic calculus. We will present the analogous theory for the lattice case, and introduce algorithms to learn the optimal policy in this setting.

Our formulation draws together concepts from many-body physics, control theory, and reinforcement learning. The next Section introduces the necessary background, before Section 3 presents the RL formulation of quantum states. Section 4 describes the application of this formulation to the neural representation of quantum states, before we present our conclusions in Section 5.

2. Theoretical Background

In this Section we will introduce the models to be studied and two stochastic representations: in continuous time using a generalized Feynman–Kac formula, and in discrete time using the Schrödinger equation directly. We then describe the class of linearly solvable Markov decision problems introduced in Todorov (2006), which form the basis of our mapping to RL. The connections between RL and quantum physics are well established in the case of continuous state spaces: see Barr et al. (2020) and references therein.

2.1. Models

For illustration purposes, we restrict ourselves to the simplest family of lattice models describing a system of N spins taking values $Z_i = \pm 1$. [1] The Hamiltonians act on the 2^N-dimensional space $\mathbb{C}^2 \otimes \mathbb{C}^2 \otimes \cdots \mathbb{C}^2$ and are expressed in terms of Pauli spin matrices $\{X_i, Y_i, Z_i\}$

$$X_i = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}_i \qquad Y_i = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}_i \qquad Z_i = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}_i \tag{1}$$

acting on spin i. We consider two examples, the **transverse field Ising model**

$$H_{\text{Ising}} = -J \sum_{\langle i,j \rangle} Z_i Z_j - h \sum_i X_i, \tag{2}$$

and the **XXZ model**

$$H_{XXZ} = -\sum_{\langle i,j \rangle} \left[J Z_i Z_j + J_\perp \left(X_i X_j + Y_i Y_j \right) \right], \tag{3}$$

where $\langle i, j \rangle$ denotes a sum over nearest neighbors on the lattice. The case $J = J_\perp$ is known as the XXX or Heisenberg model (Auerbach, 2012).

2.2. Connection to stochastic dynamics

Particular limits of the above models can be described in terms of stochastic dynamics. We illustrate this connection in the case of the XXX model by observing that in the space of two neighboring spins

$$[Z_i Z_j + X_i X_j + Y_i Y_j] - 1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}_{ij}. \tag{4}$$

This may be interpreted as a transition rate matrix for a Markov process with transitions $\circ\bullet \longleftrightarrow \bullet\circ$, where filled and empty circles represent spin up and spin down states. Apart from an additive

1. The term *lattice gas* is sometimes used, where the interpretation is that sites in the lattice are either occupied or empty.

constant, the operator $-H_{\text{XXX}}$ for $J > 0$ is then a transition rate matrix of the **symmetric simple exclusion process**, describing a lattice gas of diffusing particles (Figure 1).

$$-H_{\text{XXX}} = \overbrace{J\sum_{\langle i,j\rangle}\left[Z_iZ_j + X_iX_j + Y_iY_j - 1\right]}^{\equiv\Gamma_{\text{SEP}}} + JN. \tag{5}$$

A valid transition matrix has columns summing to zero; by symmetry of the Hamiltonian this is also true for the rows. The ground state of H_{XXX} corresponds to the dominant (maximal) eigenvector of the transition matrix with eigenvalue 0. This eigenvector describes a uniform distribution of particle configurations.

When $J \neq J_\perp$ the Hamiltonian no longer has a simple stochastic interpretation, having the form

$$H_{\text{XXZ}} = -\Gamma_{\text{SEP}} + V, \tag{6}$$

where V is diagonal but not constant. Hamiltonians of this form are called *stoquastic*, and we will refer to the Markov process defined by Γ as the *passive dynamics*, for reasons that will become clear. In terms of the matrix elements of the Hamiltonian, the potential is $V_{ss'} = V(s)\delta_{ss'}$, where

$$V(\boldsymbol{s}) = H_{\boldsymbol{ss}} + \sum_{\boldsymbol{s}'\neq\boldsymbol{s}} H_{\boldsymbol{ss}'}. \tag{7}$$

Moreover, note that (6) implies that $H_{\boldsymbol{ss}'} \leq 0$ for $\boldsymbol{s} \neq \boldsymbol{s}'$.

As a second example, consider H_{Ising}, which is stoquastic for $h > 0$ with very simple transition rates

$$\Gamma_{\boldsymbol{s}\to\boldsymbol{s}'} = h, \tag{8}$$

if the two spin configurations s and s' differ at only one site (Figure 1). From Equation (7), the potential is

$$\begin{aligned} V(\boldsymbol{s}) &= -\sum_{\boldsymbol{s}'\neq\boldsymbol{s}} \Gamma_{\boldsymbol{s}\to\boldsymbol{s}'} + H_{ss} \\ &= -hN - J\sum_{\langle i,j\rangle} s_is_j. \end{aligned}$$

A key property of stoquastic Hamiltonians is the following: if the passive dynamics is ergodic, then the ground state wavefunction is positive (Bravyi, 2015). This is a consequence of the Perron-Frobenius theorem, and explains why stoquastic Hamiltonians are 'free of the sign problem': the ground state wavefunction does not change sign.

2.3. Feynman–Kac formula

For a stoquastic Hamiltonian, a representation of the ground state in terms of the passive dynamics exists, in the form of the Feynman–Kac formula. Although this is more familiar in the context of continuous state spaces where the underlying stochastic dynamics is Brownian motion, we will require a discrete analog.

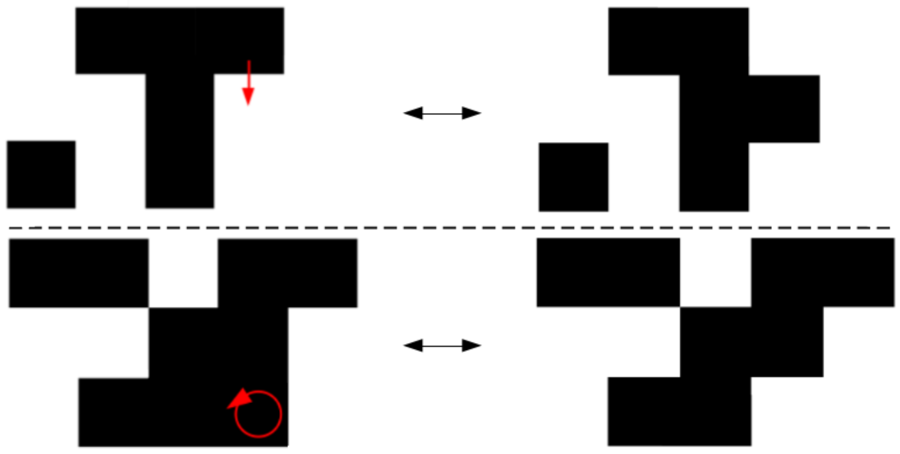

Figure 1: Transitions in the simple exclusion process (above) and the Ising model (below). In the simple exclusion process, a transition is a move of a particle to a non-occupied (white) lattice site. For the Ising model, a transition is a flip of spin up (black) to spin down (white) or the other way around.

First, note that the ground state φ_0 of the Hamiltonian is the asymptotic solution of the imaginary time Schrödinger equation

$$\partial_t \varphi(t) = -H\varphi(t) \tag{9}$$

$\varphi(t) \to e^{-E_0 t}\varphi_0$ as $t \to \infty$, where E_0 is the ground state energy (Goldberg and Schwartz, 1967).

The Feynman–Kac formula gives the solution of the imaginary time Schrödinger equation with a *terminal* condition at some time $t + T$. For a stoquastic Hamiltonian with passive rates Γ and potential V, the Feynman–Kac formula states that the solution at the earlier time t can be expressed as an expectation (Rogers and Williams, 2000)

$$\varphi(\boldsymbol{s}_t, t) = \mathop{\mathbb{E}}_{\mathbb{P}}\left[\exp\left(-\int_t^{t+T} V(\boldsymbol{s}_{t'})dt'\right)\varphi(\boldsymbol{s}_{t+T}, t+T)\right]. \tag{10}$$

Here $\varphi(\boldsymbol{s}_{t+T}, t+T)$ is the terminal condition, and the expectation is over trajectories following the passive dynamics Γ starting at s_t at time t. Since $\varphi(\boldsymbol{s}_t, t) = e^{-E_0 t}\varphi_0(\boldsymbol{s}_t)$ is an exact solution of the imaginary time Schrödinger equation, the Feynman–Kac formula implies

$$\varphi_0(\boldsymbol{s}_t) = \mathop{\mathbb{E}}_{\mathbb{P}}\left[\exp\left(-\int_t^{t+T} (V(\boldsymbol{s}_{t'})+E_0)\, dt'\right)\varphi_0(\boldsymbol{s}_{t+T})\right]. \tag{11}$$

In (11) the role of the potential is to modify the measure over trajectories relative to that of the passive dynamics. It is perhaps not obvious, but nonetheless true, that the modified measure describes the trajectories of a different Markov process. In Section 3.1, we formulate the problem of learning the Markov process that correctly describes the trajectories in the Feynman–Kac formula as a Reinforcement Learning problem.

2.4. Stochastic representations of the Schrödinger equation

The Feynman–Kac formula gives a natural stochastic representation of the ground state in terms of a *continuous* time Markov chain. As discussed in Neirotti and De Oliveira (1996), the ground state of a stoquastic Hamiltonian can also be characterized by the stationary state of a *discrete* time Markov chain. Inspired by this result, we give two stochastic representations of the time-independent Schrödinger equation

$$\sum_{\boldsymbol{s}'} H_{\boldsymbol{s}\boldsymbol{s}'}\varphi_0(\boldsymbol{s}') = E_0\varphi_0(\boldsymbol{s}). \tag{12}$$

For stoquastic Hamiltonians, the ground state wavefunction φ_0 is the unique positive eigenfunction.

For the first representation, fix $C > \max_{\boldsymbol{s}} H_{\boldsymbol{s}\boldsymbol{s}}$. For any such constant, we can rewrite the Schrödinger equation (12) as

$$\varphi_0(\boldsymbol{s}) = \frac{1}{C-E_0}\left[(C-H_{\boldsymbol{s}\boldsymbol{s}})\varphi_0(\boldsymbol{s}) - \sum_{\boldsymbol{s}'\neq\boldsymbol{s}} H_{\boldsymbol{s}\boldsymbol{s}'}\varphi_0(\boldsymbol{s}')\right]. \tag{13}$$

Since $C > \max H_{\boldsymbol{s}\boldsymbol{s}}$ and $H_{\boldsymbol{s}\boldsymbol{s}'} < 0$ for $\boldsymbol{s}' \neq \boldsymbol{s}$, we can turn this into a stochastic representation

$$\varphi_0(\boldsymbol{s}) = \frac{Z_1(\boldsymbol{s})}{C-E_0} \mathop{\mathbb{E}}_{\boldsymbol{s}'\sim p_1(\cdot|\boldsymbol{s})}\left[\varphi_0(\boldsymbol{s}')\right]. \tag{14}$$

Here the transition probabilities $p_1(\boldsymbol{s}'|\boldsymbol{s})$ are

$$p_1(\boldsymbol{s}'|\boldsymbol{s}) = \begin{cases} (C-H_{\boldsymbol{s}\boldsymbol{s}})/Z_1(\boldsymbol{s}) & \boldsymbol{s}'=\boldsymbol{s} \\ -H_{\boldsymbol{s}\boldsymbol{s}'}/Z_1(\boldsymbol{s}) & \boldsymbol{s}'\neq\boldsymbol{s} \end{cases} \tag{15}$$

with $Z_1(\boldsymbol{s}) = C - H_{\boldsymbol{s}\boldsymbol{s}} - \sum_{\boldsymbol{s}'\neq\boldsymbol{s}} H_{\boldsymbol{s}\boldsymbol{s}'}$.

For the second representation, we rewrite the Schrödinger equation slightly differently as

$$\varphi_0(\boldsymbol{s}) = -\frac{1}{H_{\boldsymbol{s}\boldsymbol{s}}-E_0}\sum_{\boldsymbol{s}'\neq\boldsymbol{s}} H_{\boldsymbol{s}\boldsymbol{s}'}\varphi_0(\boldsymbol{s}') \tag{16}$$

This leads to the stochastic representation

$$\varphi_0(\boldsymbol{s}) = \frac{Z_2(\boldsymbol{s})}{H_{\boldsymbol{s}\boldsymbol{s}}-E_0} \mathop{\mathbb{E}}_{\boldsymbol{s}'\sim p_2(\cdot|\boldsymbol{s})}\left[\varphi_0(\boldsymbol{s}')\right] \tag{17}$$

$$p_2(\boldsymbol{s}'|\boldsymbol{s}) = \begin{cases} 0 & \boldsymbol{s}'=\boldsymbol{s} \\ -H_{\boldsymbol{s}\boldsymbol{s}'}/Z_2(\boldsymbol{s}) & \boldsymbol{s}'\neq\boldsymbol{s} \end{cases} \tag{18}$$

with $Z_2(\boldsymbol{s}) = -\sum_{\boldsymbol{s}'\neq\boldsymbol{s}} H_{\boldsymbol{s}\boldsymbol{s}'}$. Note that in this second formulation the system always changes state at each step of the process. In both cases, the formulation requires knowledge of the ground state energy E_0.

2.5. Linearly solvable reinforcement learning problems

The stochastic formulation of the Schrödinger equation given in the previous section may be reinterpreted as a linearly solvable Markov decision problem of the type introduced in Todorov (2006).

Todorov considers the maximum entropy reinforcement learning paradigm, where stochastic policies are encouraged by complementing the reward function $r(\boldsymbol{s}, \boldsymbol{a})$ with the entropy of the policy relative to some reference policy p.[2] As is usual in RL, a problem setting can be summarized and studied by its Bellman equation. Two examples of (soft) Bellman equations are

$$U^*(\boldsymbol{s}) = R^* + \log \mathop{\mathbb{E}}_{\boldsymbol{a}\sim p(\cdot|\boldsymbol{s})} \left[\exp\left(r(\boldsymbol{s}, \boldsymbol{a}) + U^*(\boldsymbol{a}(\boldsymbol{s}))\right)\right] \tag{19a}$$

$$U^*(\boldsymbol{s}) = \log \mathop{\mathbb{E}}_{\boldsymbol{a}\sim p(\cdot|\boldsymbol{s})} \left[\exp\left(r(\boldsymbol{s}, \boldsymbol{a}) + U^*(\boldsymbol{a}(\boldsymbol{s}))\right)\right]. \tag{19b}$$

Here $r(\boldsymbol{s}, \boldsymbol{a})$ is the reward following the action $\boldsymbol{a}$ in the state $\boldsymbol{s}$ and R^* is a constant describing the average reward of the optimal policy. U^* is the optimal state-value function and p is the reference policy. Information implicitly contained in these soft Bellman equations is the objective function and time horizon. While (19a) describes an infinite time horizon, (19b) describes a problem with terminal states. Both consider an undiscounted setting where the objective function is simply the sum (or average) of received rewards.

Todorov (2009) showed that for action-independent rewards, both soft Bellman optimality equations (19a) and (19b) can be transformed into linear equations. Consider action-independent rewards $r(\boldsymbol{s}, \boldsymbol{a}) = r(\boldsymbol{s})$. The key is an exponential transformation to the 'desirability' z of a state

$$z(\boldsymbol{s}) = \exp(U^*(\boldsymbol{s})). \tag{20}$$

By simply exponentiating the soft Bellman optimality equations (19a) and (19b), we obtain

$$z(\boldsymbol{s}) = e^{r(\boldsymbol{s})+R^*} \mathop{\mathbb{E}}_{\boldsymbol{a}\sim p(\cdot|\boldsymbol{s})} \left[z(\boldsymbol{a}(\boldsymbol{s}))\right], \tag{21a}$$

$$z(\boldsymbol{s}) = e^{r(\boldsymbol{s})} \mathop{\mathbb{E}}_{\boldsymbol{a}\sim p(\cdot|\boldsymbol{s})} \left[z(\boldsymbol{a}(\boldsymbol{s}))\right], \tag{21b}$$

which are indeed linear equations for the desirability z. Note that the introduction of discounting results in nonlinear equations, breaking the connection with the Schrödinger equation.

3. Reinforcement learning for quantum ground states

Our first contribution is to show how to transform the stochastic representations of Sections 2.3 and 2.4 to reinforcement learning problems. The key step is a logarithmic transformation that transforms the stochastic representation into a Bellman equation, and represents the inverse of Todorov's approach.

2. For details see Appendix A.

3.1. Transforming the Feynman-Kac representation

The Feynman–Kac representation of the ground state Equation (11) may be formulated as a continuous time RL problem. In this section, we give an idea of the proof, see Appendix B for details.

Consider a small time step Δt for the Feynman–Kac characterization of the ground state (11)

$$\varphi_0(\boldsymbol{s}_t) = e^{-V(\boldsymbol{s})\Delta t + E_0 \Delta t} \mathop{\mathbb{E}}_{\mathbb{P}} \left[\varphi_0(\boldsymbol{s}_{t+\Delta t}))\right] + \mathcal{O}(\Delta t^2). \tag{22}$$

Comparison with (21a) gives a reward $r(\boldsymbol{s}) = -V(\boldsymbol{s})\Delta t$. Furthermore, the ground state wavefunction takes the role of the desirability z, so that we can make the identification

$$\log \varphi_0(\boldsymbol{s}) = U^*(\boldsymbol{s}). \tag{23}$$

This identification is visualized in Figure 2 as a Bellman backup diagram. Taking the continuous time limit, this leads to the following maximum entropy RL problem. The control is provided by transition rates Γ^θ that drive a continuous time Markov chain for the state $\boldsymbol{s}_t$ of the system. We seek the optimal rates Γ^*

$$\Gamma^* = \underset{\Gamma^\theta}{\operatorname{argmax}}\, R \tag{24}$$

$$R = -\lim_{T\to\infty} \frac{1}{T} \mathop{\mathbb{E}}_{s_{0:T}} \left[\int_0^T \left(V(\boldsymbol{s}_t) + \mathcal{H}(\boldsymbol{s}_t) \right) dt \right], \tag{25}$$

where the entropy term $\mathcal{H}$ is given by

$$\mathcal{H}(\boldsymbol{s}_t) = \sum_{\boldsymbol{s}' \neq \boldsymbol{s}_t} \left(\Gamma_{\boldsymbol{s}_t \to \boldsymbol{s}'} - \Gamma^\theta_{\boldsymbol{s}_t \to \boldsymbol{s}'} + \Gamma^\theta_{\boldsymbol{s}_t \to \boldsymbol{s}'} \log \left(\frac{\Gamma^\theta_{\boldsymbol{s}_t \to \boldsymbol{s}'}}{\Gamma_{\boldsymbol{s}_t \to \boldsymbol{s}'}} \right) \right), \tag{26}$$

and $\Gamma_{s\to s'}$ denotes the passive dynamics of the system

$$\Gamma_{\boldsymbol{s}\to\boldsymbol{s}'} = \begin{cases} -H_{\boldsymbol{s}\boldsymbol{s}'} & \boldsymbol{s} \neq \boldsymbol{s} \\ \sum_{\boldsymbol{s}'\neq\boldsymbol{s}} H_{\boldsymbol{s}\boldsymbol{s}'} & \boldsymbol{s} = \boldsymbol{s}' \end{cases} \tag{27}$$

The expectation in the objective (25) is over trajectories $s_{0:T}$ following the parameterized dynamics Γ^θ.

The optimal policy strikes a balance between steering towards states $\boldsymbol{s}$ with low potential energy $V(\boldsymbol{s})$ and staying close to the passive dynamics Γ of Equation (27). In physical terms, the objective (25) minimizes the sum of potential energy $V(\boldsymbol{s})$ and kinetic energy $\mathcal{H}$. The optimal rates Γ^* reproduce the Feynman–Kac measure on the space of trajectories . Note that (25) involves an expectation over the parameterized Markov process. In the case of continuous state spaces, optimizing this expectation is straightforward using the reparameterization trick. The discrete case requires a different approach as described below.

3.2. Transforming the Schrodinger representations

In a similar way, the stochastic representations of the Schrödinger equation from Section 2.4 can be mapped to RL in discrete time. The linear equations for the desirability Equations (21a) and (21b)

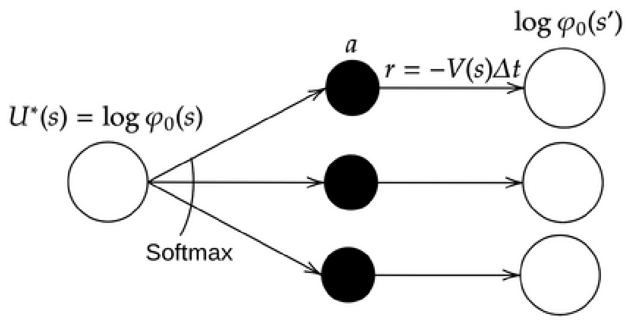

Figure 2: The Feynman–Kac formula seen as a Bellman backup diagram.

will be identified with the stochastic representations Equations (14) and (17) of the Schrödinger equation.

The first representation Equation (14) maps to the soft Bellman equation Equation (19a) i.e.

$$U^*(\boldsymbol{s}) = R^* + \log \mathop{\mathbb{E}}_{\boldsymbol{a}\sim p_1(\cdot|\boldsymbol{s})} \left[\exp\left(r(\boldsymbol{s}) + U^*(\boldsymbol{a}(\boldsymbol{s}))\right)\right]. \tag{28}$$

The identification is $\log \varphi_0(\boldsymbol{s}) = U^*(\boldsymbol{s})$ as before, but now with reward function

$$r(\boldsymbol{s}) = \log Z_1(\boldsymbol{s}) \tag{29}$$

and $R^* = -\log(C - E_0)$. Therefore, the ground state problem is also equivalent to a discrete time RL problem with an infinite time horizon.

The second representation (17) maps to the soft Bellman equation (19b)

$$U^*(\boldsymbol{s}) = \log \mathop{\mathbb{E}}_{\boldsymbol{a}\sim p_2(\cdot|\boldsymbol{s})} \left[\exp\left(r(\boldsymbol{s}) + U^*(\boldsymbol{a}(\boldsymbol{s}))\right)\right] \tag{30}$$

with reward

$$r(\boldsymbol{s}) = \log\left(\frac{Z_2(\boldsymbol{s})}{H_{\boldsymbol{s}\boldsymbol{s}} - E_0}\right). \tag{31}$$

This Bellman equation describes a RL problem with terminal states. In this type of RL problem, there is no fixed time horizon, but exploration stops when one of terminal states is reached. The reward is the total received until the terminal state is reached, which means terminal states necessarily have state-value $U^* = 0$.

We are free to choose the terminal states. Choosing a terminal state, i.e. setting $U^*(\boldsymbol{s})$ to zero, is equivalent to forcing $\varphi_0(\boldsymbol{s}) = 1$ because of the logarithmic transformation $U^* = \log \varphi_0$. Therefore, a choice of terminal state entails a choice of normalization for the wavefunction. A natural candidate for the terminal state is the classical ground state, i.e. the state $\boldsymbol{s}$ for which $H_{\boldsymbol{s}\boldsymbol{s}}$ is minimal. Another choice that could be advantageous computationally is a state $\boldsymbol{s}$ with minimum symmetry. This is because states $\boldsymbol{s}$ and $\boldsymbol{s}'$ that are related by a symmetry (such as translation) must obey $\varphi_0(\boldsymbol{s}) = \varphi_0(\boldsymbol{s}')$, so that choosing a minimum symmetry state gives you many other terminal states for free.

4. Application: neural quantum states

In this section, we describe how our mapping of the ground state problem to RL leads to new optimization techniques for neural quantum states. Usually, the quantum wavefunction is parameterized by a neural network $\varphi(\boldsymbol{s}) = \mathsf{NN}(\boldsymbol{s})$. In contrast, almost all deep reinforcement learning algorithms obtain a neural approximation of either the state-value function $U^*(\boldsymbol{s})$ or the action-value function $Q^*(\boldsymbol{s}, \boldsymbol{a})$. Given a neural approximation of the state-value function $U^*(\boldsymbol{s})$, we immediately obtain a neural approximation of the ground state wavefunction

$$\varphi_0(\boldsymbol{s}) = \exp(U^*(\boldsymbol{s})) \tag{32}$$

because this is the transformation (23) we used to derive the reinforcement learning formulations.

Given a neural approximation of the action-value function $Q^*(\boldsymbol{s}, \boldsymbol{a})$, we can also obtain a neural quantum state. For this we use the following relation between the state-value function U^* and the action-value function Q^*

$$U^*(\boldsymbol{s}) = \log \mathop{\mathbb{E}}_{\boldsymbol{a}\sim p(\cdot|\boldsymbol{s})} \left[\exp\left(Q^*(\boldsymbol{s}, \boldsymbol{a})\right)\right]. \tag{33}$$

Therefore, a neural approximation of Q^* gives a neural approximation of the ground state wavefunction

$$\varphi_0(\boldsymbol{s}) = \mathop{\mathbb{E}}_{\boldsymbol{a}\sim p(\cdot|\boldsymbol{s})} \left[\exp\left(Q^*(\boldsymbol{s}, \boldsymbol{a})\right)\right]. \tag{34}$$

4.1. Advantages of reinforcement learning neural quantum states

Most papers on Neural Quantum States use either Variational Monte Caro (e.g. Han et al. (2019); Hermann et al. (2020); Kessler et al. (2019); Saito (2017)) or Stochastic Reconfiguration (e.g. Choo et al. (2019); Nomura et al. (2017); Pfau et al. (2019)) to optimize neural quantum states. Our optimization method using reinforcement learning could be preferable for the following reasons.

Firstly, our approach needs less data for one update step. For example, Variational Monte Carlo needs to calculate the variational energy for each update step. This means that, given a parametrized wavefunction φ, samples $\boldsymbol{s} \propto \varphi^2$ need to be produced and the so-called local energy

$$\frac{H\varphi}{\varphi}(\boldsymbol{s}) = \frac{1}{\varphi(\boldsymbol{s})} \sum_{\boldsymbol{s}' \neq \boldsymbol{s}} H_{\boldsymbol{s}\boldsymbol{s}'} \varphi(\boldsymbol{s}') \tag{35}$$

needs to be averaged over these samples. Therefore, for each update step, new samples need to be generated, and for each sample, the neural wavefunction φ needs to be evaluated $\mathcal{O}(N)$ times, where N is the number of spins in the system. In contrast, in a Q-learning approach, the action-value function Q can be optimized using off-policy pairs of states $\boldsymbol{s}$ and actions $\boldsymbol{a}$. The fact that only pairs of states and actions are needed means that the neural action-value function Q only needs to evaluated $\mathcal{O}(1)$ times per sample $\boldsymbol{s}$. The fact that off-policy data can be used means that generated samples can be reused for different update steps. So one update step can be completed more efficiently using our reinforcement learning approach.

Secondly, our approach can more efficiently produce samples from the neural ground state. To see this, note that such samples are usually generated using the Metropolis–Hastings algorithm. In the Metropolis–Hastings algorithm, the proposal distribution of moves greatly influences its speed of convergence. All reinforcement learning algorithms produce a policy that prefers states of high state-value. In our formulation, that translates to a policy that points towards states of high ground state probability. In the next section, we show how this leads to a more efficient proposal distribution.

4.2. Experiments

To demonstrate the application to neural quantum states, we consider the two-dimensional transverse Ising model (Equation (2)). Training details can be found in Appendix C and the code at https://github.com/WillemGispen/Lattice-QuaRL .

We use Soft Q-learning (Haarnoja et al., 2018) with discrete space adaptations (Christodoulou, 2019) to approximate the optimal action-value function Q^* with a convolutional neural network. As described before, this leads via (34) to a neural representation of the ground state wavefunction. One advantage of Christodoulou's discrete space adaptations is that all action-values $Q^*(\boldsymbol{s}, \boldsymbol{a})$ for a specific state $\boldsymbol{s}$ are obtained with one neural network call, so that the neural quantum state (34) may be calculated simply by weighting the sum of this output.

For square lattices up to 6×6, we can compare the variational energy of our optimized wavefunctions with exact diagonalization (Hamer, 2000). The numerically exact ground state energy of the 6×6 square Ising model, at $h = 1$ and $J = 0.32758$, is $E_0 = -1.06375$ (Hamer, 2000). All our methods come within 0.1% of this value, obtaining $-1.0628(0.09\%)$, $-1.0632(0.05\%)$, and $-1.0633(0.04\%)$ for the continuous, non-terminal, and terminal states formulations, respectively. These errors should not be viewed as the limits of each method, but purely as a demonstration that each method can obtain the ground state with reasonable accuracy.

To produce samples from the ground state, we use Metropolis–Hastings with a variety of proposal distributions. The standard proposal distribution is uniform: a single random spin flip is proposed per Metropolis–Hastings step. Using our learned action-values, we can instead propose single spin flips according to $\exp(Q(\boldsymbol{s}, \boldsymbol{a}))$. Finally, we can sample multiple spin flips from $\exp(Q(\boldsymbol{s}, \boldsymbol{a}))$ and propose to flip those all at once.

To quantify how efficiently the different proposal distributions sample our optimized 6×6 Ising wavefunctions, we measure the autocorrelation of the potential energy. This autocorrelation decays approximately exponentially $\propto \exp(-t/\tau)$. Sampling single spin flips from $\exp(Q(\boldsymbol{s}, \boldsymbol{a}))$ instead of uniformly approximately halves τ. Sampling 6 spin flips from $\exp(Q(\boldsymbol{s}, \boldsymbol{a}))$ results in a $6\times$ speedup of τ. Sampling 10 spin flips from $\exp(Q(\boldsymbol{s}, \boldsymbol{a}))$ for a 10×10 Ising model (trained with the terminal states method) results in a $10\times$ speedup of τ. These 'speedups' refer to comparisons with a single uniformly generated spin flip and are comparable for all our methods. Although our measurements of τ are only rough indications for the true autocorrelation or mixing times, they suggest the following: sampling $\sqrt{N}$ spin flips from $\exp(Q(\boldsymbol{s}, \boldsymbol{a}))$ for an Ising system with N spins may reduce autocorrelation/mixing times from $\mathcal{O}(N)$ to $\mathcal{O}(\sqrt{N})$. Anyhow, it seems clear that using $\exp(Q(\boldsymbol{s}, \boldsymbol{a}))$ to propose spin flips can signicantly reduce autocorrelation/mixing times. This could be particularly beneficial when autocorrelation/mixing times are long, such as for large systems or when random actions have very low acceptance rates.

5. Conclusion

In this paper we have introduced three different reinforcement learning formulations of the problem of finding the ground state of a quantum lattice model, as summarized in Table 1.

Even before doing extensive experiments, we can note some relative advantages and disadvantages of the different RL formulations. Firstly, discrete time RL settings are more commonly studied

Basis	Time	Time horizon	Reward
Feynman-Kac formula	Continuous	Infinite	$V(s)$
Schrödinger equation	Discrete	Infinite	$\log Z_1(\boldsymbol{s})$
Schrödinger equation	Discrete	Terminal states	$\log\left(\frac{Z_2(\boldsymbol{s})}{H_{\boldsymbol{ss}}-E_0}\right)$

Table 1: Overview of the three different reinforcement learning formulations of the ground state problem.

than continuous time ones, so it is easier to find efficient RL algorithms to solve the discrete time formulations. A disadvantage of the third formulation is that the reward contains the ground state energy E_0. Still, an estimate for the ground state energy may be obtained from the variational energy, which must be calculated anyway for validation purposes. Moreover, sparse terminal states are challenging, although they can be dealt with (Agostinelli et al., 2019).

Future experiments will explore the performance of our three approaches relative to each other and existing neural representations of quantum states. As well as being the basis of methods in their own right, the learned process described by the rates Γ_θ could be used for importance sampling of the Feynman–Kac formula, which will be optimal (i.e. with zero variance) for $\Gamma_\theta = \Gamma^*$.

Acknowledgments

We thank Ariel Barr for early discussions of this work and collaboration on a related project. AL acknowledges support from EPSRC Grant No. EP/P034616/1.

References

Forest Agostinelli, Stephen McAleer, Alexander Shmakov, and Pierre Baldi. Solving the rubik's cube with deep reinforcement learning and search. *Nature Machine Intelligence*, 1(8):356–363, 2019.

Assa Auerbach. *Interacting electrons and quantum magnetism*. Springer Science & Business Media, 2012.

Ariel Barr, Willem Gispen, and Austen Lamacraft. Quantum ground states from reinforcement learning. In *Mathematical and Scientific Machine Learning*, pages 635–653. PMLR, 2020.

Sergey Bravyi. Monte Carlo simulation of stoquastic Hamiltonians. *arXiv:1402.2295 [quant-ph]*, January 2015. URL http://arxiv.org/abs/1402.2295. arXiv: 1402.2295.

Giuseppe Carleo and Matthias Troyer. Solving the quantum many-body problem with artificial neural networks. *Science*, 355(6325):602–606, February 2017. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.aag2302. URL https://science.sciencemag.org/content/355/6325/602. Publisher: American Association for the Advancement of Science Section: Research Articles.

Kenny Choo, Titus Neupert, and Giuseppe Carleo. Two-dimensional frustrated J1-J2 model studied with neural network quantum states. *Physical Review B*, 100(12):125124, September 2019.

doi: 10.1103/PhysRevB.100.125124. URL https://link.aps.org/doi/10.1103/PhysRevB.100.125124. Publisher: American Physical Society.

Petros Christodoulou. Soft Actor-Critic for Discrete Action Settings. *arXiv:1910.07207 [cs, stat]*, October 2019. URL http://arxiv.org/abs/1910.07207. arXiv: 1910.07207.

Alexandre Galashov, Siddhant M. Jayakumar, Leonard Hasenclever, Dhruva Tirumala, Jonathan Schwarz, Guillaume Desjardins, Wojciech M. Czarnecki, Yee Whye Teh, Razvan Pascanu, and Nicolas Heess. Information asymmetry in KL-regularized RL. *arXiv:1905.01240 [cs, stat]*, May 2019. URL http://arxiv.org/abs/1905.01240. arXiv: 1905.01240.

Abraham Goldberg and Judah L Schwartz. Integration of the Schrödinger equation in imaginary time. *Journal of Computational Physics*, 1(3):433–447, February 1967. ISSN 0021-9991. doi: 10.1016/0021-9991(67)90049-6. URL http://www.sciencedirect.com/science/article/pii/0021999167900496.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement Learning with Deep Energy-Based Policies. In *International Conference on Machine Learning*, pages 1352–1361, July 2017. URL http://proceedings.mlr.press/v70/haarnoja17a.html. ISSN: 1938-7228 Section: Machine Learning.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. *arXiv:1801.01290 [cs, stat]*, August 2018. URL http://arxiv.org/abs/1801.01290. arXiv: 1801.01290.

C. J. Hamer. Finite-size scaling in the transverse Ising model on a square lattice. *Journal of Physics A: Mathematical and General*, 33(38):6683–6698, September 2000. ISSN 0305-4470. doi: 10.1088/0305-4470/33/38/303. URL https://doi.org/10.1088%2F0305-4470%2F33%2F38%2F303. Publisher: IOP Publishing.

Jiequn Han, Linfeng Zhang, and Weinan E. Solving Many-Electron Schrödinger Equation Using Deep Neural Networks. *Journal of Computational Physics*, 399:108929, December 2019. ISSN 00219991. doi: 10.1016/j.jcp.2019.108929. URL http://arxiv.org/abs/1807.07014. arXiv: 1807.07014.

Jan Hermann, Zeno Schätzle, and Frank Noé. Deep neural network solution of the electronic Schrödinger equation. *arXiv:1909.08423 [physics, stat]*, July 2020. URL http://arxiv.org/abs/1909.08423. arXiv: 1909.08423.

Jan Kessler, Francesco Calcavecchia, and Thomas D. Kühne. Artificial Neural Networks as Trial Wave Functions for Quantum Monte Carlo. *arXiv:1904.10251 [physics]*, May 2019. URL http://arxiv.org/abs/1904.10251. arXiv: 1904.10251.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, January 2017. URL http://arxiv.org/abs/1412.6980. arXiv: 1412.6980.

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

JP Neirotti and MJ De Oliveira. Monte carlo method for obtaining the ground-state properties of quantum spin systems. *Physical Review B*, 53(2):668, 1996.

Yusuke Nomura, Andrew S. Darmawan, Youhei Yamaji, and Masatoshi Imada. Restricted Boltzmann machine learning for solving strongly correlated quantum systems. *Physical Review B*, 96(20):205152, November 2017. doi: 10.1103/PhysRevB.96.205152. URL https://link.aps.org/doi/10.1103/PhysRevB.96.205152. Publisher: American Physical Society.

David Pfau, James S. Spencer, Alexander G. de G. Matthews, and W. M. C. Foulkes. Ab-Initio Solution of the Many-Electron Schrödinger Equation with Deep Neural Networks. *arXiv e-prints*, 1909:arXiv:1909.02487, September 2019. URL http://adsabs.harvard.edu/abs/2019arXiv190902487P.

L. C. G. Rogers and David Williams. *Diffusions, Markov Processes, and Martingales: Volume 1: Foundations*, volume 1 of *Cambridge Mathematical Library*. Cambridge University Press, Cambridge, 2 edition, 2000. ISBN 978-0-521-77594-6. doi: 10.1017/CBO9781107590120. URL https://www.cambridge.org/core/books/diffusions-markov-processes-and-martingales/188B6A2BAABAF735E61796C3CD18114B.

Hiroki Saito. Solving the Bose–Hubbard Model with Machine Learning. *Journal of the Physical Society of Japan*, 86(9):093001, July 2017. ISSN 0031-9015. doi: 10.7566/JPSJ.86.093001. URL https://journals.jps.jp/doi/10.7566/JPSJ.86.093001. Publisher: The Physical Society of Japan.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2 edition, 2018.

Emanuel Todorov. Linearly-solvable markov decision problems. *Advances in neural information processing systems*, 19:1369–1376, 2006.

Emanuel Todorov. Efficient computation of optimal actions. *Proceedings of the National Academy of Sciences*, 106(28):11478, July 2009. doi: 10.1073/pnas.0710743106. URL http://www.pnas.org/content/106/28/11478.abstract.

Appendix A. Maximum entropy reinforcement learning

This section is based on Galashov et al. (2019); Haarnoja et al. (2017); Sutton and Barto (2018); Todorov (2009).

The usual reinforcement learning objective of maximizing average reward leads to deterministic policies. This can lead to instabilities in reinforcement learning algorithms. One attempt to alleviate this is to extend the reward with a term that promotes policies that are as random as possible. This strategy leads to the theory of maximum entropy reinforcement learning. For reinforcement learning problems, the advantages range from improved stability, exploration, and transfer learning (Levine, 2018).

Consider an reinforcement learning agent using a stochastic policy π. This means that the agent, observing state $\boldsymbol{s}$, chooses action $\boldsymbol{a}$ with probability $\pi(\boldsymbol{a}|\boldsymbol{s})$. The entropy regularization is ensured by complementing the reward $r(\boldsymbol{s}, \boldsymbol{a})$ with an entropy term

$$r(\boldsymbol{s}, \boldsymbol{a}) \to r(\boldsymbol{s}, \boldsymbol{a}) + \alpha \mathcal{H}(\pi(\cdot|\boldsymbol{s})||p(\cdot|\boldsymbol{s})). \tag{36}$$

Here α is called the 'temperature': it controls the importance of the entropy term. $\mathcal{H}(\pi(\cdot|\boldsymbol{s})||p(\cdot|\boldsymbol{s}))$ is the relative entropy[3] of the policy π with respect to some reference policy p

$$\mathcal{H}(\pi(\cdot|\boldsymbol{s})||p(\cdot|\boldsymbol{s})) = - \underset{\boldsymbol{a}\sim p(\cdot|\boldsymbol{s})}{\mathbb{E}} \left[\frac{d\pi}{dp} \log \left(\frac{d\pi}{dp} \right) (\boldsymbol{a}|\boldsymbol{s}) \right] \tag{37}$$

and $\frac{d\pi(\cdot|\boldsymbol{s})}{dp(\cdot|\boldsymbol{s})}$ is the Radon–Nikodym derivative of π with respect to p. Although the temperature α is an important parameter in maximum entropy reinforcement learning, $\alpha = 1$ is fixed by our requirement that the Bellman equation maps to a linear Schrödinger equation (Todorov (2009)). Thus, for an infinite time horizon, the entropy-regularized objective is to maximize

$$R^{\pi} = \lim_{T\to\infty} \frac{1}{T} \underset{\boldsymbol{s}_{0:T}}{\mathbb{E}} \left[\sum_{t=0}^{T} r(\boldsymbol{s}_t, \boldsymbol{a}_t) + \mathcal{H}(\pi(\cdot|\boldsymbol{s}_t)||p(\cdot|\boldsymbol{s}_t)) \right] \tag{38}$$

where the expectation is over trajectories $\boldsymbol{s}_{0:T}$ of the policy π.

The value function U^* and the action-value function Q^* are changed accordingly. Therefore, the usual Bellman optimality equations are replaced by their 'soft' counterparts: the soft Bellman optimality equations (again for the example of an infinite time horizon)

$$U^*(\boldsymbol{s}) = R^* + \log \underset{\boldsymbol{a}\sim p(\cdot|\boldsymbol{s})}{\mathbb{E}} \left[\exp \left(r(\boldsymbol{s}, \boldsymbol{a}) + U^*(\boldsymbol{a}(\boldsymbol{s})) \right) \right] \tag{39a}$$

$$Q^*(\boldsymbol{s}, \boldsymbol{a}) = R^* + r(\boldsymbol{s}, \boldsymbol{a}) + \log \underset{\boldsymbol{a}'\sim p(\cdot|\boldsymbol{a}(\boldsymbol{s}))}{\mathbb{E}} \left[\exp \left(Q^*(\boldsymbol{a}(\boldsymbol{s}), \boldsymbol{a}') \right) \right] \tag{39b}$$

where the expectations are according to the reference policy p. The constant R^* describes the average reward of the optimal policy. When there are terminal states, these soft Bellman optimality equations become

$$U^*(\boldsymbol{s}) = \log \underset{\boldsymbol{a}\sim p(\cdot|\boldsymbol{s})}{\mathbb{E}} \left[\exp \left(r(\boldsymbol{s}, \boldsymbol{a}) + U^*(\boldsymbol{a}(\boldsymbol{s})) \right) \right] \tag{40a}$$

$$Q^*(\boldsymbol{s}, \boldsymbol{a}) = r(\boldsymbol{s}, \boldsymbol{a}) + \log \underset{\boldsymbol{a}'\sim p(\cdot|\boldsymbol{a}(\boldsymbol{s}))}{\mathbb{E}} \left[\exp \left(Q^*(\boldsymbol{a}(\boldsymbol{s}), \boldsymbol{a}') \right) \right] \tag{40b}$$

When the reference policy is uniform, it is easier to see how these are the 'soft' version of the usual Bellman optimality equations: in this case we recognize the LogSumExp function which can be interpreted as a soft version of the maximum function.

Finally, it is instructive to see the relation between the optimal action-value function Q^* and the optimal policy π^*. Usually, the action is selected that has the maximum action-value, i.e. the argument of the maximum is used. In maximum entropy reinforcement learning, the optimal policy follows instead from the soft argument of the maximum, i.e.

$$\pi^*(\boldsymbol{a}|\boldsymbol{s}) = \frac{\exp(Q^*(\boldsymbol{s}, \boldsymbol{a}))}{\mathbb{E}_{\boldsymbol{a}'\sim p(\cdot|\boldsymbol{s})} \left[\exp \left(Q^*(\boldsymbol{s}, \boldsymbol{a}') \right) \right]}. \tag{41}$$

3. The negative relative entropy is also called the Kullback-Leibler (KL) divergence, and this regularization strategy is therefore sometimes called KL-regularization. If the reference policy p is uniform, then the relative entropy reduces to the Shannon entropy.

Appendix B. Proof of Reinforcement Learning Interpretation of Feynman–Kac Formula

In this section, we prove the assertions in Section 3.1. For clarity, we begin by repeating the Feynman–Kac representation of the ground state (11)

$$\varphi_0(\boldsymbol{s}_t) = \mathop{\mathbb{E}}_{\mathbb{P}}\left[\exp\left(-\int_t^{t+T} \left(V(\boldsymbol{s}_{t'}) - E_0\right) dt'\right) \varphi_0(\boldsymbol{s}_{t+T})\right]. \tag{42}$$

For a small time step $T = \Delta t$, this implies

$$\begin{aligned}\varphi_0(\boldsymbol{s}) &= \mathop{\mathbb{E}}_{\mathbb{P}}\left[\exp(-(V(\boldsymbol{s}) - E_0)\Delta t + \mathcal{O}(\Delta t^2))\varphi_0(\boldsymbol{s}_{t+\Delta t})\right] &(43)\\ &= e^{r(\boldsymbol{s})+E_0\Delta t} \mathop{\mathbb{E}}_{\mathbb{P}}\left[\varphi_0(\boldsymbol{s}_{t+\Delta t}))\right] + \mathcal{O}(\Delta t^2) &(44)\end{aligned}$$

where the reward is defined as

$$r(\boldsymbol{s}) \equiv -V(\boldsymbol{s})\Delta t. \tag{45}$$

B.1. Definition of the action space

Although (44) already looks very similar to Todorov's linear equation for the desirability (21a), we still have to rewrite the expectation over paths of length Δt in terms of actions. That is, we replace the expectation over trajectories $\mathbb{P}$ with an expectation with respect to a 'passive policy' $p_{\Delta t}(\cdot|\boldsymbol{s})$. This passive policy is a discrete time approximation of the path measure $\mathbb{P}$.

To define the policy, we first clarify what the available actions are. An action is either a move to another state $\boldsymbol{s}' \neq \boldsymbol{s}$ with $H_{\boldsymbol{s}\boldsymbol{s}'} \neq 0$, or the trivial action $\boldsymbol{a}_0$ of doing nothing, i.e. $\boldsymbol{a}_0(\boldsymbol{s}) = \boldsymbol{s}$. Since $\mathbb{P}$ is characterized by the transition rates Γ (27), the passive policy becomes

$$p_{\Delta t}(\boldsymbol{a}|\boldsymbol{s}) = \begin{cases}\Gamma_{\boldsymbol{s}\to\boldsymbol{a}(\boldsymbol{s})}\Delta t & \boldsymbol{a} \neq \boldsymbol{a}_0\\ 1 - \sum \Gamma_{\boldsymbol{s}\to\boldsymbol{a}(\boldsymbol{s})}\Delta t & \boldsymbol{a} = \boldsymbol{a}_0.\end{cases} \tag{46}$$

With this definition of the passive policy $p_{\Delta t}$, we can further rewrite (44) as

$$\varphi_0(\boldsymbol{s}) = e^{r(\boldsymbol{s})+E_0\Delta t} \mathop{\mathbb{E}}_{\boldsymbol{a}\sim p_{\Delta t}(\cdot|\boldsymbol{s})}\left[\varphi_0(\boldsymbol{a}(\boldsymbol{s}))\right] + \mathcal{O}(\Delta t^2). \tag{47}$$

B.2. Soft Bellman optimality equation

We recognize this as the same equation as Todorov's linear equation for the desirability (21a). Therefore, if we define the state-value function $U^*(\boldsymbol{s}) = \log \varphi_0(\boldsymbol{s})$, we get the soft Bellman optimality equation (19a)

$$U^*(\boldsymbol{s}) = R^* + \log \mathop{\mathbb{E}}_{\boldsymbol{a}\sim p_{\Delta t}(\cdot|\boldsymbol{s})}\left[\exp\left(r(\boldsymbol{s}) + U^*(\boldsymbol{a}(\boldsymbol{s}))\right)\right] \tag{48}$$

with $R^* = E_0\Delta t$, which is illustrated in Figure 2. Therefore, using a small time step Δt, the Feynman–Kac formula is approximately equivalent to a reinforcement learning problem with discrete time steps Δt, an infinite time horizon and reward function $r(\boldsymbol{s}) = -V(\boldsymbol{s})\Delta t$.

B.3. The continuous time limit

In continuous time, the actions taken by an agent cannot be specified by a policy. The reason is simple: the probability of taking an action in infinitesimal time is zero. Therefore, the control of an agent is specified with transition rates instead. Todorov (2009) shows how to perform the continuous time limit in continuous state spaces. We extend this argument to discrete state spaces.

In continuous time, the control is specified by transition rates Γ^θ instead of a policy. For a small discrete time step Δt, the transition rates and policy are related by

$$\pi^\theta(\boldsymbol{a}|\boldsymbol{s}) = \begin{cases} \Gamma^\theta_{\boldsymbol{s}\to\boldsymbol{a}(\boldsymbol{s})}\Delta t + \mathcal{O}(\Delta t^2) & \text{if } \boldsymbol{a} \neq \boldsymbol{a}_0, \\ 1 - \sum \Gamma^\theta_{\boldsymbol{s}\to\boldsymbol{a}(\boldsymbol{s})}\Delta t + \mathcal{O}(\Delta t^2) & \text{if } \boldsymbol{a} = \boldsymbol{a}_0. \end{cases} \tag{49}$$

Comparing with (46), we see that the passive policy of course corresponds to $\Gamma^\theta = \Gamma$.

Next, we compute the relative entropy

$$\mathcal{H}(\pi^\theta(\cdot|\boldsymbol{s})||p_{\Delta t}(\cdot|\boldsymbol{s})) = \Delta t \sum_{\boldsymbol{s}'\neq\boldsymbol{s}} \left(\Gamma_{\boldsymbol{s}\to\boldsymbol{s}'} - \Gamma^\theta_{\boldsymbol{s}\to\boldsymbol{s}'} + \Gamma^\theta_{\boldsymbol{s}\to\boldsymbol{s}'} \log\left(\frac{\Gamma^\theta_{\boldsymbol{s}\to\boldsymbol{s}'}}{\Gamma_{\boldsymbol{s}\to\boldsymbol{s}'}} \right) \right) + \mathcal{O}(\Delta t^2). \tag{50}$$

Now we can take the limit $\Delta t \to 0$. We arrive at a continuous time reinforcement learning problem, where the goal is to find the transition rates Γ^θ that maximize the objective (25).

Appendix C. Training details

Table 2 lists the hyperparameters we used in all our experiments. Training follows the Soft Q-learning algorithm (Haarnoja et al., 2017) with adaptions for discrete action spaces by Christodoulou (2019). In contrast with Christodoulou (2019), for simplicity we use only two neural networks: $Q = NN(\theta)$ and a target $\bar{Q} = NN(\bar{\theta})$. The policy is derived from Q^* following (41). For the continuous time formulation, we use the discrete time approximation described in Appendix B.2 with a timestep $\Delta t = 10^{-4}$. This is because Soft Q-learning is not directly suitable for continuous time reinforcement learning.

To keep the architecture of the neural networks as simple as possible, we used a convolutional neural network with half-padding, so that the shape of the input is preserved. For a 6x6 Ising model, this means that the output of the neural network is also a 6x6 array of real numbers, representing the action-value of flipping the spin at that lattice site. For the continuous and non-terminal formulations, however, we also need the action-value of the trivial action $\boldsymbol{a}_0$, i.e. not flipping any spin $\boldsymbol{a}_0(\boldsymbol{s}) = \boldsymbol{s}$. From the relation $Q^*(\boldsymbol{s}, \boldsymbol{a}) = R^* + r(\boldsymbol{s}) + \log \varphi_0(\boldsymbol{a}(\boldsymbol{s}))$ and the Schrödinger equation (17), it follows that

$$\exp\left(Q^*(\boldsymbol{s}, \boldsymbol{a}_0)\right) = -\frac{1}{H_{\boldsymbol{s}\boldsymbol{s}} - E_0} \sum_{\boldsymbol{a}\neq\boldsymbol{a}_0} H_{\boldsymbol{s}\boldsymbol{s}'} \exp(Q^*(\boldsymbol{s}, \boldsymbol{a}))$$

so that we do not need to represent $Q(\boldsymbol{s}, \boldsymbol{a}_0)$ separately. For this trick, we need an estimate of the ground state energy, and this is obtained from the energy validation of the wavefunction following from Q. This does not result in extra computation: the energy validation (done every 20 episodes) needs to be done to keep track of training progress anyway.

Initial states are generated by randomly selecting half the spin sites and designating those spin up, the others spin down. Training starts after the replay buffer has been filled completely with

Parameter	Value
optimizer	Adam (Kingma and Ba, 2017)
learning rate	10^{-3}
learning rate decay	$\times 0.99$ every 10 episodes
batch size	4096
replay buffer size	65536
target update interval	20
hidden layers	3
convolutional channels	64
convolutional kernel size	3
convolutional stride	1
convolutional padding	circular, 1
nonlinearity	ReLU

Table 2: Hyperparameters

experiences (states, actions, rewards, new states) following a uniform policy (i.e. every action has equal probability). For the terminal states formulation, we choose the entirely magnetized states to be terminal. Although these states are then terminal in the sense that $U^* = 0$, regeneration of initial states is unnecessary.

Every training episode, one batch of experiences is generated and saved to the replay buffer, after which one batch of experiences is sampled from the replay buffer $\mathcal{D}$. Then that batch is used to do one gradient step. The loss used in the gradient step derives from the soft Bellman optimality equation for Q^*. For the continuous time and non-terminal states formulation, the loss is the variance of the soft Bellman residue (Sutton and Barto, 2018)

$$Var_{\boldsymbol{s},\boldsymbol{a}\sim\mathcal{D}}\left[Q(\boldsymbol{s},\boldsymbol{a}) - r(\boldsymbol{s}) - \log \mathop{\mathbb{E}}_{\boldsymbol{a}'\sim p(\cdot|\boldsymbol{a}(\boldsymbol{s}))}\left[\exp\left(\bar{Q}(\boldsymbol{a}(\boldsymbol{s}),\boldsymbol{a}')\right)\right]\right],$$

i.e. the variance of the soft Bellman equation (39b). Note that the target network $\bar{Q}$ is used in the exponential, and that the constant R^* disappears. For the terminal states formulation, it is the mean squared error (Sutton and Barto, 2018)

$$\mathop{\mathbb{E}}_{\boldsymbol{s},\boldsymbol{a}\sim\mathcal{D}}\left[\left(Q(\boldsymbol{s},\boldsymbol{a}) - r(\boldsymbol{s}) - \log \mathop{\mathbb{E}}_{\boldsymbol{a}'\sim p(\cdot|\boldsymbol{a}(\boldsymbol{s}))}\left[\exp\left(\bar{Q}(\boldsymbol{a}(\boldsymbol{s}),\boldsymbol{a}')\right)\right]\right)^2\right],$$

i.e. the mean squared error of the soft Bellman equation (40b). The subscripts $\boldsymbol{s},\boldsymbol{a}\sim\mathcal{D}$ indicate that these losses are computed using batches sampled from the replay buffer $\mathcal{D}$. In contrast, the expectations inside the logarithm can be computed exactly, since one neural network call outputs $\bar{Q}(\boldsymbol{a}(\boldsymbol{s}),\boldsymbol{a}')$ for all $\boldsymbol{a}'$ and the transition probabilities $p(\cdot|\boldsymbol{a}(\boldsymbol{s}))$ are known.

Training is done for 2000/4500/4500 episodes for the continuous/non-terminal/terminal formulations respectively, taking around 20-40 minutes on a 12 GB GPU.