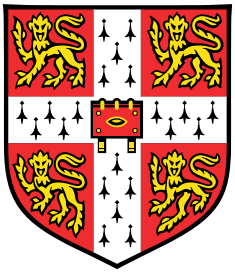
Sampling quantum ground states with maximum entropy reinforcement learning



Willem Gispen

Supervisor: Prof Austen Lamacraft

Department of Physics
University of Cambridge

This dissertation is submitted for the degree of
Master of Philosophy

Corpus Christi College September 2020

Declaration

This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration except where specifically indicated in the text. This dissertation contains fewer than 15,000 words, excluding tables, footnotes, bibliography, and appendices.

Willem Gispen
September 2020

Acknowledgements

I would like to thank Data-Piet Fonds (Prins Bernhard Cultuurfonds) and VSBFonds for their financial support, as well as Corpus Christi College for accommodating me in such a stimulating environment. I thank Prof Austen Lamacraft for his inspiring supervision and Ariel Barr for our fruitful collaboration.

Abstract

Quantum ground states can be computed with deep reinforcement learning. To be more specific, we design a maximum entropy reinforcement learning problem equivalent to the ground state problem for stoquastic Hamiltonians. This is done for both continuous and discrete state spaces and both continuous and discrete time reinforcement learning. Furthermore, we show that deep learning can numerically solve the resultant reinforcement learning problem in practice: we demonstrate our approach using small atomic and molecular systems and spin lattice models.

Table of contents

Chapter 1
Introduction

1.1 Quantum ground states

Physical systems, when cooled to absolute zero temperature, attain a state that is known as the ground state of that system. In a sense, it is the most stable state of that system, and therefore, knowing the ground state gives insight into the behavior of that system at and near absolute zero temperature. The ground state problem is studied by chemists and physicists alike, and comes up, for example, in the study of magnetism, superconductivity and stable configurations of molecules.

In quantum mechanics, the ground state is described by the ground state wavefunction φ_0 that solves the time-independent Schrödinger equation

$$H\varphi_0 = E_0\varphi_0. \tag{1.1}$$

This Schrödinger equation is an eigenequation for the Hamiltonian operator H and the ground state wavefunction φ_0 is the unique[1] eigenfunction for the smallest eigenvalue E_0, the ground state energy.

A sometimes more practical characterization of the ground state wavefunction comes from the imaginary-time Schrödinger equation

$$\partial_t\varphi(t) = -H\varphi(t) \tag{1.2}$$

whose solution $\varphi(t)$ approaches the ground state φ_0 as $t \to \infty$.[17]

As an example of a quantum system we consider a collection of charged particles. Say we have N particles of masses m_i and charges q_i. The Hamiltonian of this system consists of kinetic terms $-\frac{1}{2m_i}\nabla_i^2$ describing the kinetic energy of the particles and Coulomb interaction terms $\frac{q_iq_j}{r_{ij}}$ describing the interaction between the charges when they are at a distance r_{ij}:

$$H = -\sum_i \frac{1}{2m_i}\nabla_i^2 + \sum_{i\neq j}\frac{q_iq_j}{r_{ij}}. \tag{1.3}$$

In this example, the Hamiltonian is a partial differential operator, and the imaginary-time Schrödinger equation becomes a partial differential equation. When solving this numerically on a grid of linear size L, the wavefunction is defined on a grid of size L^{3N}, so that the complexity of the calculation increases exponentially with the number of particles.

[1] We only consider non-degenerate ground states in this thesis.

As another example, we consider the transverse field Ising model. This is a quantum spin lattice model, where the basic degrees of freedom are not positions of particles in continuous space, but rather the spins of particles living on a lattice. In the Ising case, it means that each site is assigned a spin-1/2 particle and that the interactions $-\sigma_i^z \sigma_j^z$ between particles i and j tend to align these spins. In addition, there is a transverse field $-\sigma_i^x$ that tends to flip spins randomly, so that the total Hamiltonian becomes

$$H = -\sum_{\langle i,j \rangle} \sigma_i^z \sigma_j^z - \sum_i \sigma_i^x \tag{1.4}$$

where $\sigma_i^{x,z}$ are Pauli spin matrices. In this example, the Hamiltonian is a matrix, and therefore the imaginary-time Schrödinger equation is an ordinary differential equation describing the evolution of a finite-dimensional vector. However, as the number of particles N increases, the dimension of this vector grows as 2^N, making it quickly intractable to solve the imaginary-time Schrödinger equation.

To go around this 'curse of dimensionality', many numerical techniques have been developed over the last century. Some of these use stochastic methods and are collectively known as Quantum Monte Carlo methods. It turns out that applying stochastic methods is considerably simpler if the sign structure of the ground state wavefunction is known.[5] In this thesis, we will mostly study quantum ground states of 'stoquastic' [11] Hamiltonians, for which the ground state wavefunction can be chosen positive.

1.2 Machine learning quantum ground states

Deep learning methods can address the curse of dimensionality, making it natural to use deep learning to attack the quantum ground state problem. Most modern deep learning approaches to quantum ground states find their origin in variational Monte Carlo. For a review of these 'Neural Quantum States' and their limitations see [53].

In variational Monte Carlo, the ground state wavefunction is parameterized by a list of parameters θ, and a Monte Carlo estimate is made of the variational energy of this parameterized wavefunction. Then this variational energy is minimized with respect to the parameters θ. Although this basic idea of parameterizing the wavefunction and minimizing the variational energy is common to most approaches that we mention here, they differ in their representation and optimization methods.

An early approach [24] is to represent the wavefunction with a Jastrow-Slater ansatz and minimize the variational energy with stochastic gradient descent. Another early approach [34] uses artificial neural networks, but optimizes the wavefunction with the collocation method, i.e. minimizes the mean-squared error of the Schrödinger equation.

More recently, Carleo and Troyer [7] rekindled interest in Neural Quantum States by using Restricted Boltzmann Machines to represent the wavefunction, and stochastic reconfiguration [46] to minimize the variational energy.

Most modern approaches use stochastic reconfiguration (e.g. [8, 37, 38]) or gradient descent (e.g. [23, 25, 30, 43]) to optimize the variational energy and deep artificial neural networks to represent the wavefunction. They mainly differ in the architectures of their deep artificial neural networks. For example, deep convolutional neural networks are appropriate for spin lattice models [8], while physical constraints like anti-symmetry [25, 38] or cusps [23, 25] can be built in for many electron problems.

1.3 Quantum mechanics and stochastic control

As we have seen, most deep learning approaches to quantum ground states use the Schrödinger picture, i.e. they optimize the ground state wavefunction. However, there are formulations of quantum mechanics different from the Schrödinger wave theory.

In particular, there is a (limited) formulation of quantum mechanics in terms of stochastic control theory. Previous literature on this formulation only considered the continuous time case. Additionally, it only considered continuous space single-particle Hamiltonians of the form

$$H = -\frac{1}{2m}\nabla^2 + V(\boldsymbol{s}) \tag{1.5}$$

where $V(\boldsymbol{s})$ is the potential energy.

To start with, Holland [26] showed that the time-independent Schrödinger equation for the ground state of this Hamiltonian can be converted into a Hamilton-Jacobi-Bellman equation. This results in a stochastic optimal control problem, where minimization of the control cost gives the ground state. We will revisit Holland's formulation in Chapter 5 where we show that it is a special case of our continuous time reinforcement learning formulation.[2]

Yasue [54] and Guerra and Morata [18], among others, aimed for a variational formulation of Nelson's stochastic mechanics. Again for the Hamiltonian (1.5), they showed that the time-dependent Schrödinger equation can be reformulated as another stochastic control problem. Theirs differs from Holland's in two ways. Firstly, Yasue's, Guerra and Morata's links are not limited to the ground state or to the time-independent case. Secondly, whereas the ground state follows from minimization of Holland's variational principle, in Yasue's, Guerra and Morata's variational principles, the (excited) states give only extremal points (so not necessarily minimums/maximums). This fact makes theirs less suitable for conversion into a reinforcement learning objective.

[2]Stochastic optimal control and reinforcement are closely related, but reinforcement learning needs less knowledge of its environment. We will only discuss reinforcement learning after this section.

1.4 Outline

It is now possible to state the principle aims of this thesis. Firstly, we aim to recast the ground state problem as a reinforcement learning problem, extending previous work to multiple particles, discrete state spaces and discrete time. Secondly, we show how to solve the resultant reinforcement learning problem efficiently with deep learning methods. This results in a deep learning approach to quantum ground states that is fundamentally different from the variational Monte Carlo techniques of Section 1.2.

The remainder of this thesis is organized as follows. Firstly, we will introduce the relevant theory of Markov processes and Markov Chain Monte Carlo. This is complemented with a presentation of the Feynman–Kac formula for stoquastic Hamiltonians (Chapter 2). Then we review reinforcement learning and its entropy-regularized variant (Chapter 3). We make the connection between the ground state problem and entropy-regularized reinforcement learning in discrete time in Chapter 4, and in continuous time in Chapter 5. Next, we explain the methods we used for neural network representation and loss function computation (Chapter 6). After showing numerical results for a range of quantum systems (Chapter 7), we conclude with a comparison with other deep learning approaches and a discussion of how our approach can be extended and improved (Chapter 8).

As a disclaimer: together with Prof Austen Lamacraft and Ariel Barr, I have published an article [2] that partly overlaps with the contents of this thesis. The overlap is confined to the continuous time approach for continuous state spaces. Although I have tried to give a different perspective whenever possible, I have indicated (usually at the beginning of a section) where this thesis follows the presentation of that article.

Chapter 2
Markov Chain Monte Carlo for quantum ground states

In this chapter we introduce some elementary definitions and properties of Markov processes, mainly to introduce our notation. Then we introduce a few methods of sampling from unnormalized probability distributions using these Markov processes, collectively known as Markov Chain Monte Carlo. Finally, we introduce the Feynman–Kac formula for quantum ground states, which is the basis of many Markov Chain Monte Carlo approaches to finding quantum ground states, including ours.

2.1 Markov processes

Markov processes are a type of random processes satisfying a property known as 'memorylessness'. They are ubiquitous throughout the natural sciences and beyond, describing processes ranging from bacteria movements to financial markets.

For our purposes, a stochastic process is a family of random variables $\{\hat{\boldsymbol{s}}_t\}_{t\in I}$ indexed by a real parameter $t \in I \subset \mathbb{R}$. The random variables take values in a state space S, so that a realization of the stochastic process is a set of states $\{\boldsymbol{s}_t\}_{t\in I} \subset S$, also known as a trajectory through state space.

Now a Markov process is a stochastic process as defined above, but then with memorylessness, also known as the Markov property. Informally, if you pick a time $t \in I$ and the trajectory is known up to that time and reaches a state $\boldsymbol{s}_t$, then the rest of the trajectory, i.e. after time t, only depends on this state $\boldsymbol{s}_t$. To be more precise, the distribution of trajectories $\{\boldsymbol{s}_t\}_{t'>t}$ conditioned on the state $\boldsymbol{s}_t$ is equal to the distribution of trajectories conditioned on the initial part of the trajectory $\{\boldsymbol{s}_t\}_{t'\leq t}$.

We now turn to three concrete examples of Markov processes, and in fact these are the only we will need.

2.1.1 Stochastic differential equations

Stochastic differential equations (SDEs) are used, for example, to describe financial markets and Brownian motion. In physics, the Langevin equation is an example of an SDE.

A stochastic differential equation is an example of a Markov process with a continuous state space. Intuitively, a trajectory describes a particle with position $\boldsymbol{s}_t \in \mathbb{R}^n$ moving through space with a drift velocity $\boldsymbol{v}_t$, disturbed by random displacements $d\boldsymbol{w}_t$:

$$d\boldsymbol{s}_t = \boldsymbol{v}(\boldsymbol{s}_t, t)dt + d\boldsymbol{w}_t. \tag{2.1}$$

Here the random displacements $d\boldsymbol{w}_t$ are given by the Wiener process, also known as standard Brownian motion. [1] Stochastic differential equations are formalized using stochastic integrals following the Ito calculus, see for example [32].

2.1.2 Discrete time Markov chains

Instead of being continuous, the time indexing a Markov process can also be discrete. In fact, now we will give an example of a Markov process with a discrete time index and a general state space.

A discrete time Markov chain describes a system changing state at discrete times $t = 1, 2, \ldots$. At each step, the system randomly transitions following the transition probabilities $P_{\boldsymbol{s}\to\boldsymbol{s}'}$ between states. The set of transition probabilities $P_{\boldsymbol{s}\to\boldsymbol{s}'}$ fully describes the discrete time Markov chain.

2.1.3 Continuous time Markov chains

The same idea of a Markov chain can also be used in continuous time, but then the transition probabilities $P_{\boldsymbol{s}\to\boldsymbol{s}'}$ are replaced by transition rates $\Gamma_{\boldsymbol{s}\to\boldsymbol{s}'}$. The interpretation of the transition rates is that the probability of moving from state $\boldsymbol{s}$ to state $\boldsymbol{s}'$ in a small time step $\Delta t \ll 1$ is

$$P_{\boldsymbol{s}\to\boldsymbol{s}'} = \begin{cases} \Gamma_{\boldsymbol{s}\to\boldsymbol{s}'}\Delta t + \mathcal{O}(\Delta t^2) & \text{if } \boldsymbol{s} \neq \boldsymbol{s}' \\ 1 - \sum_{\boldsymbol{s}'\neq\boldsymbol{s}} \Gamma_{\boldsymbol{s}\to\boldsymbol{s}'}\Delta t + \mathcal{O}(\Delta t^2) & \text{if } \boldsymbol{s} = \boldsymbol{s}' \end{cases} \tag{2.2}$$

Another equivalent definition is that the state of the system is constant for a finite 'holding time' distributed according to an exponential distribution with rate $\sum_{\boldsymbol{s}'\neq\boldsymbol{s}} \Gamma_{\boldsymbol{s}\to\boldsymbol{s}'}$. Then, after this holding time the system transitions into the new state $\boldsymbol{s}'$ with probability

$$P_{\boldsymbol{s}\to\boldsymbol{s}'} = \frac{\Gamma_{\boldsymbol{s}\to\boldsymbol{s}'}}{\sum_{\boldsymbol{s}'\neq\boldsymbol{s}} \Gamma_{\boldsymbol{s}\to\boldsymbol{s}'}}. \tag{2.3}$$

2.2 Markov chain Monte Carlo

One of the most important applications of Markov processes is Markov chain Monte Carlo. It allows efficient sampling from unnormalized distributions over large state spaces.We describe one particular method called the Metropolis-adjusted Langevin algorithm (MALA).

2.2.1 Detailed balance conditions

To sample from a known (but unnormalized) probability distribution ρ, Markov chain Monte Carlo methods construct a Markov process such that its stationary distribution is the desired

[1] Generally, the random displacements can differ in strength and direction, which is modeled with a diffusion coefficient in the SDE.

probability distribution. By simulating many of these Markov processes for long times, one can sample from ρ. To ensure that a Markov process has ρ as its stationary distribution, the detailed balance conditions are normally used.

For a discrete time Markov chain with transition probabilities $P^*_{\boldsymbol{s}\to\boldsymbol{s}'}$ to have $\rho(\boldsymbol{s})$ as its stationary distribution, the detailed balance conditions suffice:

$$P^*_{\boldsymbol{s}\to\boldsymbol{s}'}\rho(\boldsymbol{s}) = P^*_{\boldsymbol{s}'\to\boldsymbol{s}}\rho(\boldsymbol{s}'). \tag{2.4}$$

For a continuous time Markov chain with transition rates $\Gamma^*_{\boldsymbol{s}\to\boldsymbol{s}'}$, the detailed balance conditions are similarly

$$\Gamma^*_{\boldsymbol{s}\to\boldsymbol{s}'}\rho(\boldsymbol{s}) = \Gamma^*_{\boldsymbol{s}'\to\boldsymbol{s}}\rho(\boldsymbol{s}'). \tag{2.5}$$

2.2.2 MALA with a stochastic differential equation

To sample from a continuous probability distribution ρ, we can use a stochastic differential equation with a drift derived from the probability distribution. This idea, together with a correction for discrete time steps, is known as the Metropolis-adjusted Langevin algorithm (MALA). [40, 41]

To be specific, if we use the gradient of the logarithm of the probability distribution

$$\boldsymbol{v} = \frac{1}{2}\nabla \log \rho \tag{2.6}$$

as a drift, then the stationary state of the stochastic differential equation

$$d\boldsymbol{s} = \boldsymbol{v}dt + d\boldsymbol{w} \tag{2.7}$$

is exactly ρ. [41] Therefore, starting from any initial distribution of states, we can run the SDE above for a long time, and eventually, the resultant states will be samples from the probability distribution ρ.

In practice, this SDE is solved with discrete time steps Δt, for example with the Euler–Mayurama algorithm

$$\Delta\boldsymbol{s} = \boldsymbol{v}\Delta t + \Delta\boldsymbol{w}. \tag{2.8}$$

This time discretization introduces a bias, which is corrected with a Metropolis–Hastings acceptance step. The discretization is used to propose a move $\Delta\boldsymbol{s}$ to a new state $\boldsymbol{s}' = \boldsymbol{s} + \Delta\boldsymbol{s}$, which is then accepted with probability

$$\alpha = \min\left\{1, \frac{\rho(\boldsymbol{s}')}{\rho(\boldsymbol{s})}\frac{P_{\boldsymbol{s}'\to\boldsymbol{s}}}{P_{\boldsymbol{s}\to\boldsymbol{s}'}}\right\} \tag{2.9}$$

where $P_{\boldsymbol{s}\to\boldsymbol{s}'}$ is the probability density of proposing a move $\boldsymbol{s}\to\boldsymbol{s}'$. Using the detailed balance conditions, one can check that MALA has ρ as its stationary distribution.[41]

2.2.3 'MALA' with a Markov chain

With a discrete state space, we can also use a Metropolis–Hasting algorithm: use some transition probabilities $P_{\boldsymbol{s}\to\boldsymbol{s}'}$ to propose moves and then accept these moves with probability (2.9). However, in a discrete state space, it is less clear how to choose the transition probabilities $P_{\boldsymbol{s}\to\boldsymbol{s}'}$.

Consider the example of a quantum spin model. In this case, the Hamiltonian provides a natural description of which states are 'adjacent': states $\boldsymbol{s} \neq \boldsymbol{s}'$ are adjacent iff $H_{\boldsymbol{s}\boldsymbol{s}'} \neq 0$. So in this case it seems natural to consider transitions between adjacent states. A simple technique is to use transition probabilities that are equal for all adjacent states, and zero otherwise.

As an alternative, we can adapt the Metropolis-adjusted Langevin algorithm to a Markov chain with a discrete state space as follows. Define transition rates between adjacent states as

$$\Gamma^*_{\boldsymbol{s}\to\boldsymbol{s}'} = \sqrt{\frac{\rho(\boldsymbol{s}')}{\rho(\boldsymbol{s})}} \tag{2.10}$$

and transition rates between non-adjacent states to be zero. One can easily check that with this choice, the transition rates satisfy the detailed balance conditions, so the continuous time Markov chain has ρ as its stationary distribution. If Γ^* is known exactly, this can be used to sample from ρ without a Metropolis–Hastings acceptance step.

However, an approximation $\Gamma^\theta \approx \Gamma^*$ will create a bias. Again, this can be corrected with a Metropolis–Hasting acceptance step. Similar to the continuous space case, we do this by moving to a discrete time Markov chain. At discrete time steps, the transition rates above are used to propose a move $\boldsymbol{s} \to \boldsymbol{s}' \neq \boldsymbol{s}$ (the holding times are discarded), i.e. the transition probabilities are

$$P_{\boldsymbol{s}\to\boldsymbol{s}'} = \frac{\Gamma^\theta_{\boldsymbol{s}\to\boldsymbol{s}'}}{\sum_{\boldsymbol{s}'\neq\boldsymbol{s}} \Gamma^\theta_{\boldsymbol{s}\to\boldsymbol{s}'}} \tag{2.11}$$

Then the move is accepted with the Metropolis–Hastings probability (2.9), which guarantees that this will sample from ρ.

In this case, the Metropolis–Hasting step is even more important than in the continuous space case: even if Γ^* is exactly known, discarding the holding times creates a bias that must be corrected.

2.3 Stoquastic Hamiltonians

Before we show how to link the previous Markov process with quantum ground states using the Feynman–Kac formula, we introduce the class of Hamiltonians to which the Feynman–Kac formula applies. This is the class of stoquastic Hamiltonians, and apart from defining stoquasticity, we give examples of some stoquastic and some non-stoquastic Hamiltonians.

Our method is primarily meant for stoquastic Hamiltonians, so this section also introduces models we will experiment with.

2.3.1 Spin-lattice models

Stoquasticity is a statement about a Hamiltonian matrix that depends on the basis used for the matrix representation. Given a matrix representation of a Hamiltonian in a certain basis, the Hamiltonian matrix is stoquastic if the Hamiltonian matrix has real and non-positive off-diagonal elements. Nevertheless, we will also call the Hamiltonian operator itself stoquastic if there is a known basis for which the Hamiltonian matrix is stoquastic.

Stoquasticity for a Hamiltonian matrix H implies that we can decompose the Hamiltonian matrix as

$$H = -\Gamma + V \tag{2.12}$$

where Γ is a transition rate matrix and V is diagonal. The interpretation is as follows. Γ is the kinetic term and defines a continuous time Markov chain: the 'passive dynamics'. The second term is the potential V. In terms of the Hamiltonian, the passive transition rates are given by

$$\Gamma_{\boldsymbol{s}\to\boldsymbol{s}'} = \begin{cases} -H_{\boldsymbol{s}\boldsymbol{s}'} & \text{if } \boldsymbol{s} \neq \boldsymbol{s} \\ \sum_{\boldsymbol{s}'\neq\boldsymbol{s}} H_{\boldsymbol{s}\boldsymbol{s}'} & \text{if } \boldsymbol{s} = \boldsymbol{s}' \end{cases} \tag{2.13}$$

and the potential is given by

$$V(\boldsymbol{s}) = H_{\boldsymbol{s}\boldsymbol{s}} + \sum_{\boldsymbol{s}'\neq\boldsymbol{s}} H_{\boldsymbol{s}\boldsymbol{s}'}. \tag{2.14}$$

The passive transition rates (2.13) describe a continuous time Markov chain. The transitions are between states $\boldsymbol{s} \neq \boldsymbol{s}'$ that satisfy $H_{\boldsymbol{s}\boldsymbol{s}'} \neq 0$. We will call such states 'adjacent'. An important property of stoquastic Hamiltonians is the following: if the passive dynamics (2.13) is ergodic, then the ground state wavefunction can be chosen positive in the stoquastic basis [11], i.e.

$$\varphi_0(\boldsymbol{s}) > 0. \tag{2.15}$$

This is a consequence of the Perron-Frobenius theorem.

In the following, we give examples of stoquastic (Ising, XY, AFH) and non-stoquastic (J_1-J_2) spin-lattice model Hamiltonians. The stoquastic Hamiltonians we consider all satisfy this ergodic property so that the ground state wavefunction may be chosen positive. We make use of the z-spin basis, i.e. a basis state $\boldsymbol{s} = (s_1, s_2, \ldots, s_N)$ measures the spin s_i of each site i in the z-direction. The total number of spin sites is denoted by N.

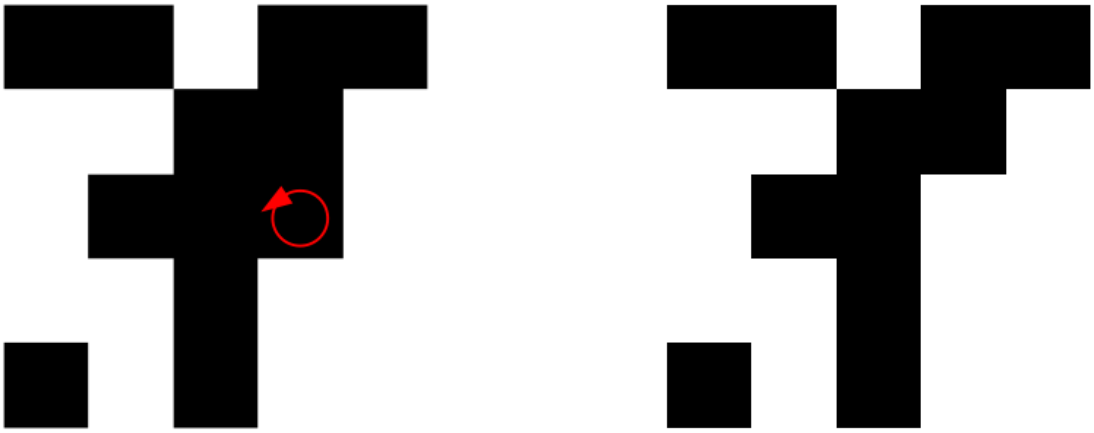


Fig. 2.1 Two adjacent Ising spin states: one spin can be flipped (red icon) to obtain the new state (on the right).

Transverse field Ising model

First consider the transverse field Ising model

$$H = -J \sum_{\langle i,j \rangle} \sigma_i^z \sigma_j^z - h \sum_i \sigma_i^x. \tag{2.16}$$

where the first sum is over all nearest neighbors $\langle i, j \rangle$ and the second sum is over all sites i. In the z-spin basis this becomes

$$\langle \boldsymbol{s}' | H | \boldsymbol{s} \rangle = -J \sum_{\langle i,j \rangle} s_i s_j \langle \boldsymbol{s}' | \boldsymbol{s} \rangle - h \sum_i \langle \boldsymbol{s}' | \sigma_i^+ + \sigma_i^- | \boldsymbol{s} \rangle . \tag{2.17}$$

Therefore, states $\boldsymbol{s}$ and $\boldsymbol{s}'$ are adjacent if $s_i = s_i'$ for each site i except one, which is illustrated in Figure 2.1. The passive transition rates are

$$\Gamma_{\boldsymbol{s} \to \boldsymbol{s}'} = h \tag{2.18}$$

if $\boldsymbol{s}$ and $\boldsymbol{s}'$ are adjacent and zero otherwise. Following (2.14), the potential becomes

$$V(\boldsymbol{s}) = -\sum_{\boldsymbol{s}' \neq \boldsymbol{s}} \Gamma_{\boldsymbol{s} \to \boldsymbol{s}'} + H_{ss} \tag{2.19}$$

$$= -hN - J \sum_{\langle i,j \rangle} s_i s_j. \tag{2.20}$$

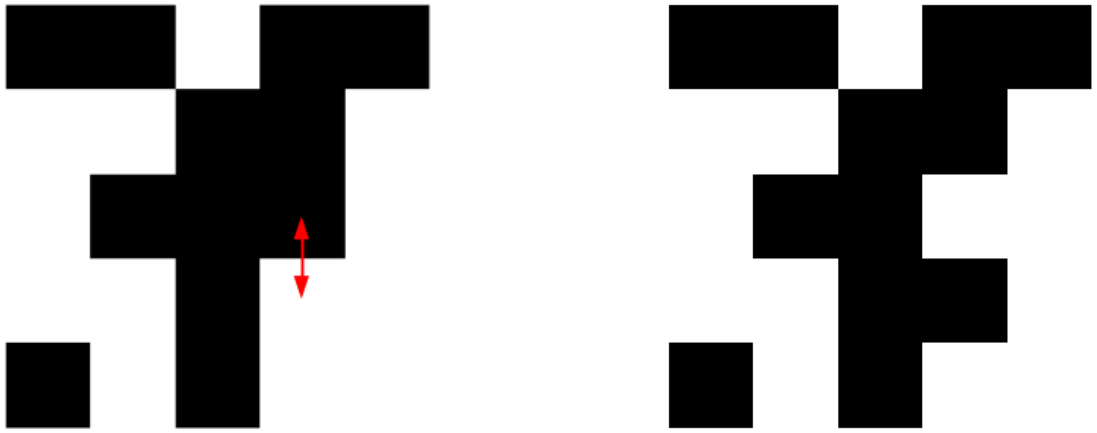


Fig. 2.2 Two adjacent XY spin states: a pair of unequal neighboring spins can be flipped to obtain the new state (on the right).

XY model

As another example, the XY model is described by the Hamiltonian

$$H = -\frac{1}{2}\sum_{\langle i,j\rangle} \sigma_i^x \sigma_j^x + \sigma_i^y \sigma_j^y \,. \tag{2.21}$$

In the z-basis this Hamiltonian becomes

$$\langle \boldsymbol{s}'|H|\boldsymbol{s}\rangle = -\sum_{\langle i,j\rangle} \langle \boldsymbol{s}'|\sigma_i^+\sigma_j^- + \sigma_i^-\sigma_j^+|\boldsymbol{s}\rangle \,. \tag{2.22}$$

Therefore, states $\boldsymbol{s}$ and $\boldsymbol{s}'$ are adjacent if you can obtain $\boldsymbol{s}'$ from $\boldsymbol{s}$ by first picking a pair of nearest neighbors $\langle i,j\rangle$ for which $s_i \neq s_j$, then setting $s_i' = s_j$ and $s_j' = s_i$, i.e. swap the values of s_i and s_j. This is illustrated in Figure 2.2. The transition rates are

$$\Gamma_{\boldsymbol{s}\to\boldsymbol{s}'} = 1 \tag{2.23}$$

if states $\boldsymbol{s}$ and $\boldsymbol{s}'$ are adjacent and zero otherwise. The potential becomes

$$V(\boldsymbol{s}) = -\sum_{\boldsymbol{s}'\neq\boldsymbol{s}} \Gamma_{\boldsymbol{s}\to\boldsymbol{s}'} \tag{2.24}$$

$$= -\sum_{\langle i,j\rangle} (n_i(1-n_j) + n_j(1-n_i)) \tag{2.25}$$

where $n_i = s_i + \frac{1}{2}$.

Antiferromagnetic Heisenberg model

Next consider the antiferromagnetic Heisenberg model

$$H = \frac{J}{2} \sum_{\langle i,j \rangle} \sigma_i^x \sigma_j^x + \sigma_i^y \sigma_j^y + \sigma_i^z \sigma_j^z \ . \tag{2.26}$$

In the z-basis this Hamiltonian becomes

$$\langle \boldsymbol{s}' | H | \boldsymbol{s} \rangle = J \sum_{\langle i,j \rangle} \langle \boldsymbol{s}' | \sigma_i^+ \sigma_j^- + \sigma_i^- \sigma_j^+ | \boldsymbol{s} \rangle + \frac{J}{2} \sum_{\langle i,j \rangle} s_i' s_j \langle \boldsymbol{s}' | \boldsymbol{s} \rangle \tag{2.27}$$

This means that the z-basis does not give a stoquastic representation of the antiferromagnetic Heisenberg model. For a bipartite lattice however, we know the sign structure of the ground state wavefunction. The Marshall sign rule [8, 36] states that if $M_A(\boldsymbol{s})$ is the number of up spins of the state $\boldsymbol{s}$ on the first bipartite sublattice, then the ground state wavefunction satisfies

$$\varphi_0(\boldsymbol{s}) = (-1)^{M_A(\boldsymbol{s})} \tilde{\varphi}_0(\boldsymbol{s}) \tag{2.28}$$

where $\tilde{\varphi}_0$ is positive. Using this sign rule, we can transform the Hamiltonian by bosonization $\sigma_i^\pm = \epsilon_i b_i^\pm$, where ϵ_i equals $+1$ if i is on the first sublattice and -1 if i is on the second sublattice. [51] This gives the transformed Hamiltonian

$$H = E_N - \frac{J}{2} \sum_{\langle i,j \rangle} b_i^+ b_j^- + b_i^- b_j^+ + J \sum_{\langle i,j \rangle} \hat{n}_i \hat{n}_j \tag{2.29}$$

where $E_N = -JNZ/8$ is the energy of the Neel state, Z is the coordination number of the lattice (4 for the square lattice) and $\hat{n}_i = b_i^+ b_i^-$ counts the number of bosons on site i.

In the occupation number basis $|\boldsymbol{n}\rangle$, this becomes

$$\langle \boldsymbol{n} | H | \boldsymbol{n}' \rangle = -\frac{J}{2} \sum_{\langle i,j \rangle} \langle \boldsymbol{n} | b_i^+ b_j^- + b_i^- b_j^+ | \boldsymbol{n}' \rangle + (E_N + J \sum_{\langle i,j \rangle} n_i n_j) \langle \boldsymbol{n} | \boldsymbol{n}' \rangle \tag{2.30}$$

where n_i is the number of bosons on site i. So states $\boldsymbol{n}$ and $\boldsymbol{n}'$ are adjacent in the same way as XY states are adjacent. We get the passive transition rates

$$\Gamma_{\boldsymbol{n} \to \boldsymbol{n}'} = \frac{J}{2} \tag{2.31}$$

if states $\boldsymbol{n}$ and $\boldsymbol{n}'$ are adjacent and zero otherwise. The potential is

$$V(\boldsymbol{n}) = -\sum_{\boldsymbol{n}'\neq\boldsymbol{n}} \Gamma_{\boldsymbol{n}\to\boldsymbol{n}'} + H_{\boldsymbol{nn}} \tag{2.32}$$

$$= E_N - J\sum_{\langle i,j\rangle} n_i(1-n_j) + n_j(1-n_i) + n_i n_j. \tag{2.33}$$

J_1-J_2 model

A generalization of the Heisenberg model is the J_1-J_2 model

$$H = J_1 \sum_{\langle i,j\rangle} \boldsymbol{\sigma}_i \cdot \boldsymbol{\sigma}_j + J_2 \sum_{\langle\langle i,j\rangle\rangle} \boldsymbol{\sigma}_i \cdot \boldsymbol{\sigma}_j \tag{2.34}$$

where the first sum is over nearest neighbors $\langle i,j\rangle$ and the second over next-nearest neighbors $\langle\langle i,j\rangle\rangle$. The J_1-J_2 model includes the antiferromagnetic Heisenberg model when $J_2 = 0$. Similar to the Heisenberg case, in the z-basis the Hamiltonian becomes

$$\langle \boldsymbol{s}'|H|\boldsymbol{s}\rangle = 2J_1 \sum_{\langle i,j\rangle} \langle \boldsymbol{s}'|\sigma_i^+\sigma_j^- + \sigma_i^-\sigma_j^+|\boldsymbol{s}\rangle + 2J_2 \sum_{\langle\langle i,j\rangle\rangle} \langle \boldsymbol{s}'|\sigma_i^+\sigma_j^- + \sigma_i^-\sigma_j^+|\boldsymbol{s}\rangle \tag{2.35}$$

$$+ J_1 \sum_{\langle i,j\rangle} s_i' s_j \langle \boldsymbol{s}'|\boldsymbol{s}\rangle + J_2 \sum_{\langle\langle i,j\rangle\rangle} s_i' s_j \langle \boldsymbol{s}'|\boldsymbol{s}\rangle\,. \tag{2.36}$$

Once again, this is not a stoquastic representation. In contrast with the antiferromagnetic Heisenberg model, for the J_1-J_2 no stoquastic representation is known.

2.3.2 Continuous state space

Although stoquasticity is a statement about Hamiltonian matrices, it can also be given a meaning in continuous state space. A Hamiltonian is stoquastic if the Hamiltonian matrix is stoquastic in a discretized position basis.[11] Therefore, any system of interacting bosons or distinguishable particles (or systems that can be treated as such) is stoquastic in the position basis. [11] Just like in the discrete case (2.15), stoquastic Hamiltonians have ground state wavefunctions that can be chosen positive, i.e.

$$\varphi_0(\boldsymbol{s}) > 0. \tag{2.37}$$

For the examples below and for the rest of this thesis, we use Hartree atomic natural units $m_e = e = \hbar = a_0 = \mathrm{E}_h = k_e = 1$ throughout, as well as the Born–Oppenheimer approximation.

Hydrogen atom

The hydrogen atom consists of a positively charged proton and a negatively charged electron. With $\boldsymbol{s} \in \mathbb{R}^3$ the position of the electron relative to the position of the proton, the Hamiltonian is

$$H = -\frac{1}{2}\nabla^2 + V(\boldsymbol{s}) = -\frac{1}{2}\nabla^2 - \frac{1}{|\boldsymbol{s}|}. \tag{2.38}$$

For the Hydrogen atom, the ground state wavefunction is known exactly to be

$$\varphi_0(\boldsymbol{s}) = \frac{1}{\sqrt{\pi}} e^{-|\boldsymbol{s}|}. \tag{2.39}$$

with ground state energy $E_0 = -\frac{1}{2}$.

Helium atom

The Helium atom consists of a nucleus of charge $+2$ and two electrons of charge -1. The Hamiltonian is

$$H = -\frac{1}{2}\left(\nabla^2_{\boldsymbol{r}_1} + \nabla^2_{\boldsymbol{r}_2}\right) - \frac{2}{|\boldsymbol{r}_1|} - \frac{2}{|\boldsymbol{r}_2|} + \frac{1}{|\boldsymbol{r}_1 - \boldsymbol{r}_2|} \tag{2.40}$$

where $\boldsymbol{r}_1$, $\boldsymbol{r}_2$ measure the distance of the electrons from the nucleus. Although this is a system of two electrons (so fermions not bosons), it is known that that the ground spin state is anti-symmetric (parahelium), so that the spatial ground state wavefunction is symmetric. Therefore, to find the spatial ground state wavefunctions, they may be treated as bosons, so that this may be treated as a stoquastic Hamiltonian.

Hydrogen molecule

The Hydrogen molecule consists of two nuclei of charge $+1$ each and two electrons of charge -1. The nuclei are separated by a distance R, and for definiteness we place the nuclei at the positions $\pm R\hat{\boldsymbol{z}}/2$. The Hamiltonian is

$$H = -\frac{1}{2}\left(\nabla^2_{\boldsymbol{r}_1} + \nabla^2_{\boldsymbol{r}_2}\right) + \frac{1}{R} + \frac{1}{|\boldsymbol{r}_1 - \boldsymbol{r}_2|} - \frac{1}{|\boldsymbol{r}_1 - R\hat{\boldsymbol{z}}/2|} - \frac{1}{|\boldsymbol{r}_1 + R\hat{\boldsymbol{z}}/2|} - \frac{1}{|\boldsymbol{r}_2 - R\hat{\boldsymbol{z}}/2|} - \frac{1}{|\boldsymbol{r}_2 + R\hat{\boldsymbol{z}}/2|} \tag{2.41}$$

where $\boldsymbol{s}_1$, $\boldsymbol{s}_2$ are the positions of the electrons with respect to the origin. This is another case where the ground spin state is anti-symmetric (parahydrogen), so that it may be treated as a stoquastic Hamiltonian.

Notation for multi-particle systems

In this thesis, we generally denote the state of a system with $\boldsymbol{s}$. So for the Helium atom and Hydrogen molecule, the state $\boldsymbol{s}$ is a 6-dimensional vector $\boldsymbol{s} = (\boldsymbol{r}_1, \boldsymbol{r}_2)$. In general, the state for N particles is a $3N$-dimensional vector $\boldsymbol{s} = (\boldsymbol{r}_1, \ldots, \boldsymbol{r}_N)$. This situation should be kept in

mind at all times: for example differential operators such as $\nabla_{\boldsymbol{s}}^2$ and vector fields $\boldsymbol{v}(\boldsymbol{s})$ are $3N$-dimensional. This allows us to present statements and proofs that are simultaneously valid for single- and multi-particle systems.

2.4 Feynman–Kac formula

Now we return to the quantum ground state problem. The Markov chain Monte Carlo algorithms discussed earlier apply to sampling from the ground state probability density: if we set $\rho = |\varphi_0|^2$ we can use Markov chain Monte Carlo to sample from $|\varphi_0|^2$. However, the ground state wavefunction φ_0 is the unknown in the ground state problem. The Feynman–Kac formula is a method of solving for φ_0 using Markov processes. To be more specific, it is a stochastic solution of the imaginary time Schrödinger equation (1.2). In turn, the Feynman–Kac formula is the basis of Quantum Monte Carlo methods such as path-integral Monte Carlo.

We present the Feynman–Kac formula for the case of stoquastic Hamiltonians only. In this case, it does not suffer from the so-called 'sign problem'.

2.4.1 Feynman–Kac formula for continuous state space

We want to solve the imaginary time Schrödinger equation (1.2) with some terminal condition $\varphi(\boldsymbol{s}_{t+T}, t+T)$. The Feynman–Kac formula states that the solution at the earlier time t can be expressed as an expectation

$$\varphi(\boldsymbol{s}_t, t) = \mathop{\mathbb{E}}_{\mathbb{P}}\left[\exp\left(-\int_t^{t+T} V\left(\boldsymbol{s}_{t'}\right) dt'\right) \varphi(\boldsymbol{s}_{t+T}, t+T)\right] \tag{2.42}$$

over the measure $\mathbb{P}$ of Brownian trajectories starting at $\boldsymbol{s}_t$ at time t.

Although this may not seem immediately useful because it goes backwards in time, it also implies another characterization of the ground state wavefunction. Because we know that $\varphi(\boldsymbol{s}_t, t) = e^{-E_0 t}\varphi_0(\boldsymbol{s}_t)$ is an exact solution of the imaginary time Schrödinger equation, the Feynman–Kac formula implies

$$\varphi_0(\boldsymbol{s}_t) = \mathop{\mathbb{E}}_{\mathbb{P}}\left[\exp\left(-\int_t^{t+T} \left(V(\boldsymbol{s}_{t'}) - E_0\right) dt'\right) \varphi_0(\boldsymbol{s}_{t+T})\right], \tag{2.43}$$

where again the expectation is over Brownian trajectories starting at $\boldsymbol{s}_t$ at time t. Reversely, if there is a positive wavefunction φ and a constant E satisfying (2.43), then $\varphi = \varphi_0$ and $E = E_0$. The reason is that (2.43) implies (via the imaginary time Schrödinger equation) that φ is a positive eigenfunction of H, and φ_0 is the only such eigenfunction.

2.4.2 Feynman–Kac formula for discrete state space

The Feynman–Kac formula has a less well known extension to discrete spaces. For a stoquastic Hamiltonian with passive rates Γ and potential V, the Feynman–Kac solution of the imaginary time Schrödinger equation with terminal condition $\varphi(\boldsymbol{s}_{t+T}, t+T)$ becomes [42]

$$\varphi(\boldsymbol{s}_t, t) = \mathop{\mathbb{E}}_{\mathbb{P}}\left[\exp\left(-\int_t^{t+T} V(\boldsymbol{s}_{t'})dt'\right)\varphi(\boldsymbol{s}_{t+T}, t+T)\right] \tag{2.44}$$

where the expectation is over trajectories following the passive dynamics Γ starting at s_t at time t. So the Feynman–Kac formula takes the exact same form as in the continuous case (2.42), with the only difference being the measure $\mathbb{P}$: in the continuous case this is the measure of Brownian dynamics, whereas in the discrete case it is the measure of trajectories following passive dynamics given by the transition rates (2.13). Therefore, we also get the same characterization of the ground state (2.43) for the discrete case.

Chapter 3
Maximum entropy reinforcement learning

Informally, in reinforcement learning an agent tries to make optimal decisions. Each decision is judged with a certain reward, and the goal of the agent is to maximize the total reward over its lifetime. Reinforcement learning algorithms can be used to solve problems ranging from lift management, reactor control and computer games.

In this chapter, we start with the most common formulation of reinforcement learning. Afterwards, we summarize maximum entropy reinforcement learning. This is a common regularization strategy in modern reinforcement learning algorithms. Lastly, we review work by Todorov on linearly solvable maximum entropy reinforcement learning problems.

3.1 Reinforcement learning

This section follows Sutton and Barton. [47]

Reinforcement learning is usually formulated in terms of a Markov decision process, which is an extension of a discrete time Markov process. In a discrete time Markov process, the state $\boldsymbol{s}_t$ at time t changes according to some transition probabilities $P_{\boldsymbol{s}_t \to \boldsymbol{s}'}$. The discrete time Markov process is extended in two ways: there is a notion of control and a notion of reward. In a Markov decision process, an agent, after observing the state $\boldsymbol{s}$, chooses an action $\boldsymbol{a}$ (control). Following that action, the state changes to state $\boldsymbol{s}'$ with transition probability $P_{\boldsymbol{s} \to \boldsymbol{s}'}(\boldsymbol{a})$. Finally, the agent is rewarded for its action with a reward $r(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')$. Generally, the goal of the agent is to choose its actions optimally so that it maximizes the total reward it receives.

There are multiple variants of reinforcement learning problems, but we will focus on the case of a deterministic environment and an infinite time horizon with an objective of averaged reward. In this case, the next state depends deterministically on the action chosen: $\boldsymbol{s}' = \boldsymbol{a}(\boldsymbol{s})$ (deterministic environment), so that the reward $r(\boldsymbol{s}, \boldsymbol{a})$ also only depends on the current state and action. Finally, the objective is to maximize the average reward R over infinite time. That is, the performance of a policy π that takes actions $\boldsymbol{a}_t$ in state $\boldsymbol{s}_t$ is measured by the objective function

$$R^\pi = \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T} r(\boldsymbol{s}_t, \boldsymbol{a}_t), \quad \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_t = \pi(\boldsymbol{s}_t) \tag{3.1}$$

for any initial state $\boldsymbol{s}_0$. Note that this independent of the chosen initial state $\boldsymbol{s}_0$. The policy that maximizes the objective function is known as the optimal policy π^* i.e.

$$R^* = R^{\pi^*} = \max_\pi \{R^\pi\}. \tag{3.2}$$

A common strategy for solving reinforcement leaning problems is to estimate the optimal value $U^*(\boldsymbol{s})$ of a state $\boldsymbol{s}$. The optimal value $U^*(\boldsymbol{s})$ of $\boldsymbol{s}$ is basically the total reward relative to R^* over infinite time starting from the state $\boldsymbol{s}$ and following the optimal policy π^*

$$U^*(\boldsymbol{s}) = \lim_{T\to\infty} \sum_{t=0}^{T} \left(r(\boldsymbol{s}_t, \boldsymbol{a}_t) - R^*\right), \quad \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_t = \pi^*(\boldsymbol{s}_t). \tag{3.3}$$

The idea is that if you know which actions lead to which states, then you simply select the action that leads to the state with the highest value. Since the optimal policy is not known, it cannot be used to compute the value function U^*. Instead, the value function U^* satisfies the Bellman optimality equation

$$U^*(\boldsymbol{s}) = -R^* + \max_{\boldsymbol{a}}\{r(\boldsymbol{s}, \boldsymbol{a}) + U^*(\boldsymbol{a}(\boldsymbol{s}))\} \tag{3.4}$$

which can be solved for U^*.

If it is not known which actions lead to which states, then the action-value function $Q(\boldsymbol{s}, \boldsymbol{a})$ is used instead. This is defined as the total reward relative to R^* over infinite time starting from the state $\boldsymbol{s}$ and the action $\boldsymbol{a}$, and then following the optimal policy:

$$Q^*(\boldsymbol{s}, \boldsymbol{a}) = \lim_{T\to\infty} \sum_{t=0}^{T} \left(r(\boldsymbol{s}_t, \boldsymbol{a}_t) - R^*\right), \quad \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}, \boldsymbol{a}_t = \pi^*(\boldsymbol{s}_t). \tag{3.5}$$

Similar to the value function U^*, the action-value function Q^* satisfies a Bellman optimality equation

$$Q^*(\boldsymbol{s}, \boldsymbol{a}) = -R^* + r(\boldsymbol{s}, \boldsymbol{a}) + \max_{\boldsymbol{a}'}\{Q^*(\boldsymbol{a}(\boldsymbol{s}), \boldsymbol{a}')\} \tag{3.6}$$

which can be solved for Q^*. If, somehow, a solution Q^* has been obtained, the optimal policy follows from Q^* by taking the action that has the largest action-value, i.e.

$$\boldsymbol{a}^*(\boldsymbol{s}) = \pi^*(\boldsymbol{s}) = \operatorname{argmax}\{Q^*(\boldsymbol{s}, \boldsymbol{a})\}. \tag{3.7}$$

3.2 Maximum entropy reinforcement learning

This section is based on [16, 19, 47, 50].

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.[35]

We consider the same setting as before, a deterministic environment with infinite time averaged rewards, but now we allow for stochastic policies. This means that an agent, observing state $\boldsymbol{s}$, chooses action $\boldsymbol{a}$ with probability $\pi(\boldsymbol{a}|\boldsymbol{s})$. The entropy regularization is ensured by the replacement

$$r(\boldsymbol{s},\boldsymbol{a}) \to r(\boldsymbol{s},\boldsymbol{a}) + \mathcal{H}(\pi(\cdot|\boldsymbol{s})||p(\cdot|\boldsymbol{s})) \tag{3.8}$$

in the objective R^π where $\mathcal{H}(\pi(\cdot|\boldsymbol{s})||p(\cdot|\boldsymbol{s}))$ is the relative entropy[1] of the policy π with respect to some reference policy p

$$\mathcal{H}(\pi(\cdot|\boldsymbol{s})||p(\cdot|\boldsymbol{s})) = -\mathop{\mathbb{E}}_{\boldsymbol{a}\sim p(\cdot|\boldsymbol{s})}\left[\frac{d\pi}{dp}\log\left(\frac{d\pi}{dp}\right)(\boldsymbol{a}|\boldsymbol{s})\right] \tag{3.9}$$

and $\frac{d\pi(\cdot|\boldsymbol{s})}{dp(\cdot|\boldsymbol{s})}$ is the Radon–Nikodym derivative of π with respect to p. So the entropy-regularized objective is to maximize

$$R^\pi = \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T} r(\boldsymbol{s}_t,\boldsymbol{a}_t) + \mathcal{H}(\pi(\cdot|\boldsymbol{s}_t)||p(\cdot|\boldsymbol{s}_t)). \tag{3.10}$$

The value function U^* and the action-value function Q^* are changed accordingly. Therefore, the Bellman optimality equations are replaced by their 'soft' counterparts: the soft Bellman optimality equations

$$U^*(\boldsymbol{s}) = -R^* + \log\mathop{\mathbb{E}}_{\boldsymbol{a}\sim p(\cdot|\boldsymbol{s})}\left[\exp\left(r(\boldsymbol{s},\boldsymbol{a}) + U^*(\boldsymbol{a}(\boldsymbol{s}))\right)\right] \tag{3.11a}$$

$$Q^*(\boldsymbol{s},\boldsymbol{a}) = -R^* + r(\boldsymbol{s},\boldsymbol{a}) + \log\mathop{\mathbb{E}}_{\boldsymbol{a}'\sim p(\cdot|\boldsymbol{a}(\boldsymbol{s}))}\left[\exp\left(Q^*(\boldsymbol{a}(\boldsymbol{s}),\boldsymbol{a}')\right)\right] \tag{3.11b}$$

where the expectations are according to the reference policy p.

When there are a finite number of actions and the reference policy is uniform, it is easier to see how these are the 'soft' version of the earlier Bellman optimality equations (3.4) and (3.6): in this case we recognize the LogSumExp function which can be interpreted as a soft version of the maximum function.

Before, the optimal policy was derived from Q^* using the argument of the maximum, but in maximum entropy reinforcement learning, the optimal policy follows instead from the soft argument of the maximum, i.e.

$$\pi^*(\boldsymbol{a}|\boldsymbol{s}) = \frac{\exp(Q^*(\boldsymbol{s},\boldsymbol{a}))}{\mathbb{E}_{\boldsymbol{a}'\sim p(\cdot|\boldsymbol{s})}\left[\exp\left(Q^*(\boldsymbol{s},\boldsymbol{a}')\right)\right]}. \tag{3.12}$$

[1]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.

3.3 Linearly solvable maximum entropy reinforcement learning problems

Todorov [50] showed that for action-independent rewards, the soft Bellman optimality equation (3.11a) can be transformed into a linear equation.

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

$$z(\boldsymbol{s}) = \exp(U^*(\boldsymbol{s})). \tag{3.13}$$

By simply exponentiating the soft Bellman optimality equation (3.11a), we obtain

$$z(\boldsymbol{s}) = e^{r(\boldsymbol{s})+R^*} \mathop{\mathbb{E}}_{\boldsymbol{a}\sim p(\cdot|\boldsymbol{s})} \left[z(\boldsymbol{a}(\boldsymbol{s}))\right], \tag{3.14}$$

which is indeed a linear equation for the desirability z.

It is worth stressing again that our reinforcement learning formulation is only one of many. Notably, the objective function (3.10) may be adapted to continuous time, finite time horizons and discounted rewards. The linearization of the soft Bellman optimality equation is possible in many of these cases. For example, see [28] and [48] for continuous time and space, or [14] and [50] and for a more general overview.

Chapter 4

Quantum ground states from discrete-time reinforcement learning

In this chapter, we show how to reformulate the quantum ground state problem as a soft Bellman optimality equation. As we have seen in Section 3.3, Todorov [50] proved that in a special case the soft Bellman optimality equation can be converted into a linear equation. In this chapter, we effectively use the opposite direction: we transform linear equations – the Feynman–Kac formula and the Schrödinger equation – into a soft Bellman optimality equation (this idea is from [2]). We do this for stoquastic Hamiltonians only. In the non-stoquastic case, we show that the Schrödinger equation is instead equivalent to a complex-valued soft Bellman equation.

To informally motivate the developments of the coming sections, we consider the logarithm of the Feynman–Kac formula

$$\log \varphi(\boldsymbol{s}_t, t) = \log \mathop{\mathbb{E}}_{\mathbb{P}} \left[\exp \left(- \int_t^{t+T} V(\boldsymbol{s}_{t'}) dt' + \log \left(\varphi(\boldsymbol{s}_{t+T}, t+T) \right) \right) \right]. \tag{4.1}$$

This is very similar to a soft Bellman equation (3.11a), particularly if you define the state-value function $U(\boldsymbol{s}, t) = \log \varphi(\boldsymbol{s}, t)$ and reward $r = -\int V(\boldsymbol{s}_{t'}) dt'$. Then you get

$$U(\boldsymbol{s}, t) = \log \mathop{\mathbb{E}}_{\mathbb{P}} \left[\exp \left(r + U(\boldsymbol{s}_{t+T}, t+T) \right) \right] \tag{4.2}$$

which is almost a soft Bellman optimality equation.

4.1 Discrete time Feynman–Kac and soft Bellman

In this section, we consider only stoquastic Hamiltonians. In the stoquastic case, we know that the ground state wavefunction is strictly positive. Moreover, we deal with continuous and discrete state spaces simultaneously. For both cases, we have an identical Feynman–Kac characterization of the ground state (2.43). The only difference is in the meaning of the measure of the passive dynamics $\mathbb{P}$: for continuous state spaces, this corresponds to the dynamics of Brownian motion, whereas for discrete state spaces, it corresponds to the passive dynamics of the continuous time Markov chain (2.13).

For clarity, we repeat the Feynman–Kac characterization of the ground state (2.43)

$$\varphi_0(\boldsymbol{s}_t) = \mathop{\mathbb{E}}_{\mathbb{P}} \left[\exp \left(- \int_t^{t+T} \left(V(\boldsymbol{s}_{t'}) - E_0 \right) dt' \right) \varphi_0(\boldsymbol{s}_{t+T}) \right]. \tag{4.3}$$

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

$$\varphi_0(\boldsymbol{s}) = \mathop{\mathbb{E}}_{\mathbb{P}}\left[\exp(-(V(\boldsymbol{s}) - E_0)\Delta t + \mathcal{O}(\Delta t^2))\varphi_0(\boldsymbol{s}_{t+\Delta t})\right] \tag{4.4}$$

$$= e^{r(\boldsymbol{s})+E_0\Delta t}\mathop{\mathbb{E}}_{\mathbb{P}}\left[\varphi_0(\boldsymbol{s}_{t+\Delta t}))\right] + \mathcal{O}(\Delta t^2) \tag{4.5}$$

where the reward is defined as

$$r(\boldsymbol{s}) \equiv -V(\boldsymbol{s})\Delta t. \tag{4.6}$$

Definition of the action space

Although (4.5) already looks very similar to Todorov's linear equation for the desirability (3.14), 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 $\mathbb{P}$ and is therefore different for continuous and discrete state spaces.

In the continuous case, the actions are displacements $\boldsymbol{a} = \Delta\boldsymbol{s}$ and they act on a state by addition $\boldsymbol{a}(\boldsymbol{s}) = \boldsymbol{s} + \Delta\boldsymbol{s}$. Since the passive dynamics is Brownian, the passive policy is a normal distribution with zero mean and variance Δt:

$$p_{\Delta t}(\cdot|\boldsymbol{s}) = N(0, \Delta t) \tag{4.7}$$

In the discrete case, an action is either a move to an adjacent state, 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 (2.13), the passive policy becomes

$$p_{\Delta t}(\boldsymbol{a}|\boldsymbol{s}) = \begin{cases} \Gamma_{\boldsymbol{s}\to\boldsymbol{a}(\boldsymbol{s})}\Delta t & \text{if } \boldsymbol{a} \neq \boldsymbol{a}_0 \\ 1 - \sum \Gamma_{\boldsymbol{s}\to\boldsymbol{a}(\boldsymbol{s})}\Delta t & \text{if } \boldsymbol{a} = \boldsymbol{a}_0. \end{cases} \tag{4.8}$$

With these definitions of the passive policy $p_{\Delta t}$, we can further rewrite (4.5) as

$$\varphi_0(\boldsymbol{s}) = e^{r(\boldsymbol{s})+E_0\Delta t}\mathop{\mathbb{E}}_{\boldsymbol{a}\sim p_{\Delta t}(\cdot|\boldsymbol{s})}\left[\varphi_0(\boldsymbol{a}(\boldsymbol{s}))\right] + \mathcal{O}(\Delta t^2). \tag{4.9}$$

Soft Bellman optimality equations

We recognize this as the same equation as Todorov's linear equation for the desirability (3.14). Therefore, if we define the state-value function $U^*(\boldsymbol{s}) = \log\varphi_0(\boldsymbol{s})$, we get the soft Bellman optimality equation (3.11a)

$$U^*(\boldsymbol{s}) = E_0\Delta t + \log\mathop{\mathbb{E}}_{\boldsymbol{a}\sim p_{\Delta t}(\cdot|\boldsymbol{s})}\left[\exp\left(r(\boldsymbol{s}) + U^*(\boldsymbol{a}(\boldsymbol{s}))\right)\right]. \tag{4.10}$$

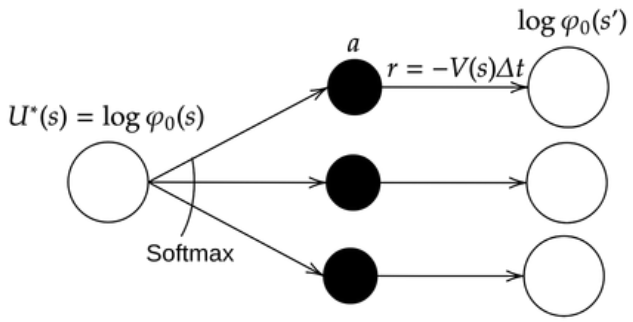


Fig. 4.1 The soft Bellman equation (4.10) visualized as a Bellman backup diagram.

Because generally $Q^*(\boldsymbol{s}, \boldsymbol{a}) = r(\boldsymbol{s}, \boldsymbol{a}) - R^* + U^*(\boldsymbol{a}(\boldsymbol{s}))$ (this follows from the definitions (3.3), (3.5)), this also implies that the action-value function

$$Q^*(\boldsymbol{s}, \boldsymbol{a}) = r(\boldsymbol{s}) + E_0 \Delta t + \log \varphi_0(\boldsymbol{a}(\boldsymbol{s})) \tag{4.11}$$

satisfies the soft Bellman optimality equation (3.11b)

$$Q^*(\boldsymbol{s}, \boldsymbol{a}) = r(\boldsymbol{s}) + E_0 \Delta t + \log \mathop{\mathbb{E}}_{\boldsymbol{a}' \sim p_{\Delta t}(\cdot|\boldsymbol{a}(\boldsymbol{s}))} \left[\exp\left(Q^*(\boldsymbol{a}(\boldsymbol{s}), \boldsymbol{a}') \right) \right]. \tag{4.12}$$

Obtain the wavefunction

Finally, we show how to obtain the ground state wavefunction φ_0 from a solution Q of the soft Bellman optimality equation (4.12). Since E_0 is a priori unknown, by a solution we mean a function Q and a constant E such that

$$Q(\boldsymbol{s}, \boldsymbol{a}) = r(\boldsymbol{s}) + E \Delta t + \log \mathop{\mathbb{E}}_{\boldsymbol{a}' \sim p_{\Delta t}(\cdot|\boldsymbol{a}(\boldsymbol{s}))} \left[\exp\left(Q(\boldsymbol{a}(\boldsymbol{s}), \boldsymbol{a}') \right) \right]. \tag{4.13}$$

If you have such a solution Q, we define the wavefunction

$$\varphi(\boldsymbol{s}) = \mathop{\mathbb{E}}_{\boldsymbol{a} \sim p_{\Delta t}(\cdot|\boldsymbol{s})} \left[\exp(Q(\boldsymbol{s}, \boldsymbol{a})) \right]. \tag{4.14}$$

With this definition, (4.13) implies that $\varphi = \varphi_0$, $E = E_0$ and $Q = Q^*$.

Firstly, (4.13) implies that φ satisfies

$$\varphi(\boldsymbol{s}) = e^{r(\boldsymbol{s}) + E \Delta t} \mathop{\mathbb{E}}_{\boldsymbol{a} \sim p_{\Delta t}(\cdot|\boldsymbol{s})} \left[\varphi_0(\boldsymbol{a}(\boldsymbol{s})) \right]. \tag{4.15}$$

Following previous arguments in the opposite direction, this implies that

$$\varphi(\boldsymbol{s}_t) = \mathop{\mathbb{E}}_{\mathbb{P}}\left[\exp\left(-\int_t^{t+T}(V(\boldsymbol{s}_{t'}) - E)\,dt'\right)\varphi(\boldsymbol{s}_{t+T})\right], \tag{4.16}$$

i.e. φ_0 satisfies the Feynman–Kac characterization of the ground state wavefunction. Therefore $\varphi = \varphi_0$ and $E = E_0$, giving $Q = Q^*$ as well.

4.2 Schrödinger and (complex) soft Bellman

We begin this section by giving another proof of (4.9). Although this proof is less intuitive and only valid for finite state spaces, it can be adapted to non-stoquastic Hamiltonians.

With the non-stoquastic case in mind, consider a Hamiltonian H. The time-independent Schrodinger equation can be written as

$$(H_{\boldsymbol{ss}} - E)\varphi(\boldsymbol{s}) = -\sum H_{\boldsymbol{s},\boldsymbol{a}(\boldsymbol{s})}\varphi(\boldsymbol{a}(\boldsymbol{s})). \tag{4.17}$$

Introducing a time step $\Delta t > 0$, we can rewrite this as

$$(1 + (V(\boldsymbol{s}) - E)\Delta t)\varphi(\boldsymbol{s}) = -\Delta t\sum H_{\boldsymbol{s},\boldsymbol{a}(\boldsymbol{s})}\varphi(\boldsymbol{a}(\boldsymbol{s})) + \left(1 + \Delta t\sum H_{\boldsymbol{s},\boldsymbol{a}(\boldsymbol{s})}\right)\varphi(\boldsymbol{s}) \tag{4.18}$$

where $V(\boldsymbol{s}) \equiv H_{\boldsymbol{ss}} + \sum H_{\boldsymbol{s},\boldsymbol{a}(\boldsymbol{s})}$. Now for $\Delta t \ll 1$ this is equivalent to

$$\varphi(\boldsymbol{s}) = \frac{1}{1 + (V(\boldsymbol{s}) - E)\Delta t}\left[-\Delta t\sum H_{\boldsymbol{s},\boldsymbol{a}(\boldsymbol{s})}\varphi(\boldsymbol{a}(\boldsymbol{s})) + \left(1 + \Delta t\sum H_{\boldsymbol{s},\boldsymbol{a}(\boldsymbol{s})}\right)\varphi(\boldsymbol{s})\right] \tag{4.19}$$

$$= e^{-V(\boldsymbol{s})\Delta t + E\Delta t}\left[-\Delta t\sum H_{\boldsymbol{s},\boldsymbol{a}(\boldsymbol{s})}\varphi(\boldsymbol{a}(\boldsymbol{s})) + \left(1 + \Delta t\sum H_{\boldsymbol{s},\boldsymbol{a}(\boldsymbol{s})}\right)\varphi(\boldsymbol{s})\right] + \mathcal{O}(\Delta t^2) \tag{4.20}$$

Stoquastic case

Now if H is stoquastic, i.e. $H = -\Gamma + V$, this implies

$$\varphi(\boldsymbol{s}) = e^{-V(\boldsymbol{s})\Delta t + E\Delta t}\mathop{\mathbb{E}}_{\boldsymbol{a}\sim p_{\Delta t}(\cdot|\boldsymbol{s})}[\varphi(\boldsymbol{a}(\boldsymbol{s}))] + \mathcal{O}(\Delta t^2) \tag{4.21}$$

where the passive policy $p_{\Delta t}$ is given by (4.8). So in the stoquastic case we have recovered (4.9) as desired.

4.2.1 Complex Bellman equation for non-stoquastic Hamiltonians

If the Hamiltonian is not stoquastic, we can still transform (4.20) into something similar to a soft Bellman optimality equation.

To do this, we introduce, for any complex-valued function $f : \boldsymbol{a} \mapsto f(\boldsymbol{a})$

$$\tilde{\mathbb{E}}_{\boldsymbol{a}\sim p_{\Delta t}(\cdot|\boldsymbol{s})}\left[f(\boldsymbol{a})\right] \equiv -\Delta t \sum_{\boldsymbol{a}\neq\boldsymbol{a}_0} H_{\boldsymbol{s},\boldsymbol{a}(\boldsymbol{s})} f(\boldsymbol{a}) + \left(1 + \Delta t \sum_{\boldsymbol{a}\neq\boldsymbol{a}_0} H_{\boldsymbol{s},\boldsymbol{a}(\boldsymbol{s})}\right) f(\boldsymbol{a}_0). \tag{4.22}$$

This is not really an expectation, because if H is non-stoquastic, it would contain negative 'probabilities'. Nevertheless, this notation is useful because it allows us to translate the developments of Section 4.1 to non-stoquastic Hamiltonians.

To start with, with this notation, (4.20) becomes

$$\varphi(\boldsymbol{s}) = e^{-V(\boldsymbol{s})\Delta t + E\Delta t}\tilde{\mathbb{E}}_{\boldsymbol{a}\sim p_{\Delta t}(\cdot|\boldsymbol{s})}\left[\varphi(\boldsymbol{a}(\boldsymbol{s}))\right] + \mathcal{O}(\Delta t^2) \tag{4.23}$$

which is the analog of (4.9). Therefore, the complex-valued action-value function

$$Q^*(\boldsymbol{s},\boldsymbol{a}) = r(\boldsymbol{s}) + E_0\Delta t + \log\varphi_0(\boldsymbol{a}(\boldsymbol{s})). \tag{4.24}$$

satisfies the analog of the soft Bellman optimality equation (3.11b)

$$Q^*(\boldsymbol{s},\boldsymbol{a}) = r(\boldsymbol{s}) + E_0\Delta t + \log\tilde{\mathbb{E}}_{\boldsymbol{a}'\sim p_{\Delta t}(\cdot|\boldsymbol{a}(\boldsymbol{s}))}\left[\exp\left(Q^*(\boldsymbol{a}(\boldsymbol{s}),\boldsymbol{a}')\right)\right], \tag{4.25}$$

where the reward is $r(\boldsymbol{s}) = -V(\boldsymbol{s})\Delta t$ as before. In the same vein, we can recover the wavefunction from a solution Q of this equation by

$$\varphi(\boldsymbol{s}) = \tilde{\mathbb{E}}_{\boldsymbol{a}\sim p_{\Delta t}(\cdot|\boldsymbol{s})}\left[\exp(Q(\boldsymbol{s},\boldsymbol{a})\right]. \tag{4.26}$$

The complex soft Bellman equation (4.25) differs in two important ways from the usual soft Bellman equation (3.11b). Firstly, as stated before, the term containing $\tilde{\mathbb{E}}$ is not an expectation. Therefore, it should be evaluated using its definition (4.22). Secondly, the action-value function Q^* is complex valued, whereas it is usually positive.

The second point means that care should be taken with the logarithm. Since ground states can generally be taken real-valued, a convenient branch for the logarithm is the one that excludes the negative imaginary line.

Although (4.25) differs from the standard soft Bellman optimality equation in these two ways, we will see that we can still use standard reinforcement learning methods to solve it.

Chapter 5

Quantum ground states from continuous time reinforcement learning

Usually, reinforcement learning is studied in discrete time and this is also what we presented in Chapter 3. In the previous chapter, we have shown that the ground state problem is equivalent to such a discrete time reinforcement learning problem. This equivalence holds up to $\mathcal{O}(\Delta t^2)$ where Δt is the discrete time step. Therefore, it is natural to investigate the limit $\Delta t \to 0$.

Doing this, we show that the quantum ground state problem is equivalent to a continuous time reinforcement learning problem. Then we reinterpret this result using the Feynman–Kac measure for continuous time trajectories. Finally, this results in a variational principle similar to, but more general than, that of Holland [26].

5.1 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 a drift or with transition rates in continuous and discrete state spaces, respectively. In other words, we move from discrete time Markov chains to continuous time Markov chains and stochastic differential equations.

The following reinforcement learning problems arise naturally from the results of Chapter 4 using a continuous time limit. In the continuous space case, we are looking to optimize a drift $\boldsymbol{v}^\theta$ with respect to the objective

$$R = -\lim_{T\to\infty} \frac{1}{T} \int_0^T \left(V(\boldsymbol{s}_t) + \frac{1}{2} |\boldsymbol{v}^\theta(\boldsymbol{s}_t)|^2 \right) dt, \tag{5.1}$$

where $\boldsymbol{s}_t$ follows the drift $\boldsymbol{v}^\theta$. In the discrete case, we are looking to optimize transition rates Γ^θ with respect to the objective

$$R = -\lim_{T\to\infty} \frac{1}{T} \int_0^T \left(V(\boldsymbol{s}_t) + \sum_{\boldsymbol{s}' \neq \boldsymbol{s}_t} \left(\Gamma_{\boldsymbol{s}_t \to \boldsymbol{s}'} - \Gamma^\theta_{\boldsymbol{s}_t \to \boldsymbol{s}'} + \Gamma^\theta_{\boldsymbol{s}_t \to \boldsymbol{s}'} \log \left(\frac{\Gamma^\theta_{\boldsymbol{s}_t \to \boldsymbol{s}'}}{\Gamma_{\boldsymbol{s}_t \to \boldsymbol{s}'}} \right) \right) \right) dt, \tag{5.2}$$

where $\boldsymbol{s}_t$ follows the transition rates Γ^θ.

Proof: continuous state space

Following Todorov [50], we first show how a continuous time reinforcement learning problem arises from the discrete time formulation. In continuous state space, the control is specified

by a drift $\boldsymbol{v}^\theta(\boldsymbol{s})$, and the dynamics by the stochastic differential equation

$$d\boldsymbol{s}_t = \boldsymbol{v}^\theta(\boldsymbol{s}_t)dt + d\boldsymbol{w}_t. \tag{5.3}$$

This makes sense, because in Section 2.2.2 we have seen that with the right choice of $\boldsymbol{v}^\theta$ this can sample from the ground state. Discretizing time gives

$$\Delta\boldsymbol{s}_t = \boldsymbol{v}^\theta(\boldsymbol{s}_t)\Delta t + \Delta\boldsymbol{w}_t, \tag{5.4}$$

where $\Delta\boldsymbol{w}_t \sim N(0, \Delta t)$. Therefore, the policy $\pi(\cdot|\boldsymbol{s})$ corresponding to a drift $\boldsymbol{v}^\theta(\boldsymbol{s})$ is Gaussian with mean $\boldsymbol{v}^\theta(\boldsymbol{s})\Delta t$ and variance Δt. Comparing (4.7), we see that the passive policy corresponds to $\boldsymbol{v}^\theta = 0$. Furthermore, to compute the entropy-regularized reward (3.10), we first claim that

$$\mathcal{H}(\pi(\cdot|\boldsymbol{s})||p_{\Delta t}(\cdot|\boldsymbol{s})) = -\frac{1}{2}|\boldsymbol{v}^\theta(\boldsymbol{s})|^2\Delta t \tag{5.5}$$

which gives the entropy-regularized reward

$$R^\pi = \lim_{T\to\infty}\frac{1}{T}\sum_t\left(\frac{r(\boldsymbol{s}_t)}{\Delta t} - \frac{1}{2}|\boldsymbol{v}^\theta(\boldsymbol{s}_t)|^2\right)\Delta t. \tag{5.6}$$

Taking the limit $\Delta t \to 0$, we get the objective

$$R^\theta = \lim_{T\to\infty}\frac{1}{T}\int_{t=0}^{T}\left(\dot{r}(\boldsymbol{s}_t) - \frac{1}{2}|\boldsymbol{v}^\theta(\boldsymbol{s}_t)|^2\right)dt \tag{5.7}$$

where $\dot{r}(\boldsymbol{s}) = \lim_{\Delta t\to 0}(r(\boldsymbol{s})/\Delta t)$.

In our case $r(\boldsymbol{s}) = -V(\boldsymbol{s})\Delta t$, which means that the ground state problem is equivalent to a continuous time reinforcement learning problem, where the goal is to find the drift $\boldsymbol{v}^\theta$ that maximizes the objective (5.1).

Proof: discrete state space

Now we extend Todorov's argument to discrete state spaces. In this case, the control is specified by transition rates Γ^θ. Discretizing time gives

$$\pi^\theta(\boldsymbol{a}|\boldsymbol{s}) = \begin{cases}\Gamma^\theta_{\boldsymbol{s}\to\boldsymbol{a}(\boldsymbol{s})}\Delta t & \text{if } \boldsymbol{a}\neq\boldsymbol{a}_0,\\ 1-\sum\Gamma^\theta_{\boldsymbol{s}\to\boldsymbol{a}(\boldsymbol{s})}\Delta t & \text{if } \boldsymbol{a}=\boldsymbol{a}_0.\end{cases} \tag{5.8}$$

Comparing with (4.8), we see that the passive policy of course corresponds to $\Gamma^\theta = \Gamma$.

Next, we compute the relative entropy

$$\mathcal{H}(\pi^\theta(\cdot|\boldsymbol{s})||p_{\Delta t}(\cdot|\boldsymbol{s})) = \Delta t\sum_{\boldsymbol{s}'\neq\boldsymbol{s}}\left(\Gamma_{\boldsymbol{s}\to\boldsymbol{s}'} - \Gamma^\theta_{\boldsymbol{s}\to\boldsymbol{s}'} + \Gamma^\theta_{\boldsymbol{s}\to\boldsymbol{s}'}\log\left(\frac{\Gamma^\theta_{\boldsymbol{s}\to\boldsymbol{s}'}}{\Gamma_{\boldsymbol{s}\to\boldsymbol{s}'}}\right)\right). \tag{5.9}$$

Again taking the limit $\Delta t \to 0$ of the entropy-regularized objective (3.10), we arrive at a continuous time reinforcement learning problem, where the goal is to find the transition rates Γ^θ that maximize the objective (5.2).

5.2 Feynman–Kac measure

This section partly follows [2] for the continuous state space case. The interpretation using the ground state drift/rates is novel, although a closely related variant with a time-dependent drift has been published before.[15]

We reinterpret the previous results using the Feynman–Kac measure. The Feynman–Kac characterization of the ground state (2.43) suggests a transformed measure $\mathbb{P}_{\mathrm{FK}}$ – on the space of trajectories $\{\boldsymbol{s}_t\}$ – defined by the Radon–Nikodym derivative

$$\log\left(\frac{d\mathbb{P}_{\mathrm{FK}}}{d\mathbb{P}}\right) \equiv -\int_0^T (V(\boldsymbol{s}_t) - E_0)dt + \log\left(\frac{\varphi_0(\boldsymbol{s}_T)}{\varphi_0(\boldsymbol{s}_0)}\right). \tag{5.10}$$

The Feynman–Kac characterization of the ground state is basically equivalent to the statement that $\mathbb{P}_{\mathrm{FK}}$ as defined above is a normalized measure. In the following, we prove an interpretation of $\mathbb{P}_{\mathrm{FK}}$ using the ground state drift

$$\boldsymbol{v}^* \equiv \nabla \log \varphi_0 \tag{5.11}$$

and the ground state transition rates

$$\Gamma^*_{\boldsymbol{s}\to\boldsymbol{s}'} \equiv \Gamma_{\boldsymbol{s}\to\boldsymbol{s}'}\frac{\varphi_0(\boldsymbol{s}')}{\varphi_0(\boldsymbol{s})}. \tag{5.12}$$

These are the MALA drift (2.6) and transition rates (2.10) for the ground state probability density $\rho = \varphi_0^2$. Essentially, the interpretation is that $\mathbb{P}_{\mathrm{FK}}$ measures the probability density that a trajectory $\{\boldsymbol{s}_t\}$ has been generated by the ground state drift/transition rates.

In order to later make the connection with the reinforcement learning problem, we also compute the Radon–Nikodym derivative between the Feynman–Kac measure and $\mathbb{P}_\theta$. In the continuous case, $\mathbb{P}_\theta$ measures the probability that a trajectory has been generated by the drift $\boldsymbol{v}^\theta$, and the Radon–Nikodym derivative becomes

$$\log\left(\frac{d\mathbb{P}_\theta}{d\mathbb{P}_{\mathrm{FK}}}\right) = \int\left(V(\boldsymbol{s}_t) - E_0 - \frac{1}{2}|\boldsymbol{v}(\boldsymbol{s}_t)|^2\right)dt + \int \boldsymbol{v}(\boldsymbol{s}_t)\cdot d\boldsymbol{s}_t - \log\left(\frac{\varphi_0(\boldsymbol{s}_T)}{\varphi_0(\boldsymbol{s}_0)}\right). \tag{5.13}$$

In the discrete case, $\mathbb{P}_\theta$ measures the probability that a trajectory has been generated by the transition rates Γ^θ, and the Radon–Nikodym derivative becomes

$$\log\left(\frac{d\mathbb{P}_\theta}{d\mathbb{P}_{\mathrm{FK}}}(\boldsymbol{s}_t)\right) = \int \left(V(\boldsymbol{s}_t) - E_0 + \sum_{\boldsymbol{s}'\neq \boldsymbol{s}_t} (\Gamma_{\boldsymbol{s}_t\to\boldsymbol{s}'} - \Gamma^\theta_{\boldsymbol{s}_t\to\boldsymbol{s}'})\right) dt \tag{5.14}$$
$$+ \sum_n \log\left(\frac{\Gamma^\theta_{\boldsymbol{s}_n\to\boldsymbol{s}_{n+1}}}{\Gamma_{\boldsymbol{s}_n\to\boldsymbol{s}_{n+1}}}\right) - \log\left(\frac{\varphi_0(\boldsymbol{s}_T)}{\varphi_0(\boldsymbol{s}_0)}\right).$$

Here $\boldsymbol{s}_t$ denotes the state at time t, $\boldsymbol{s}_0$ the state at time $t=0$, $\boldsymbol{s}_T$ at time $t=T$, and $\boldsymbol{s}_n$ denotes the state after n transitions have occurred.

Both (5.13) and (5.14) are relatively trivial consequences of (5.10). The proof of (5.10) is the same in both cases: first we rewrite the time-independent Schrödinger equation in terms of the ground state drift/rates, then we use Girsanov's theorem (and its analog) to compute the Radon–Nikodym derivatives.

Proof: continuous state space

Rewriting the time-independent Schrödinger equation in terms of $\boldsymbol{v}^*$ gives

$$V(\boldsymbol{s}) - E_0 = \frac{1}{2}\left(\nabla\cdot\boldsymbol{v}^*(\boldsymbol{s}) + |\boldsymbol{v}^*(\boldsymbol{s})|^2\right). \tag{5.15}$$

We can use this together with Girsanov's theorem [32]

$$\log\left(\frac{d\mathbb{P}_\theta}{d\mathbb{P}}\right) = -\frac{1}{2}\int |\boldsymbol{v}^\theta|^2 dt + \int \boldsymbol{v}^\theta\cdot d\boldsymbol{s}_t \tag{5.16}$$

to compute the RN-derivative

$$\log\left(\frac{d\mathbb{P}_{\mathrm{FK}}}{d\mathbb{P}}\right) = -\frac{1}{2}\int |\boldsymbol{v}^*|^2 dt + \int \boldsymbol{v}^* d\boldsymbol{s} \tag{5.17}$$
$$= -\frac{1}{2}\int |\boldsymbol{v}^*|^2 dt - \frac{1}{2}\int (\nabla\cdot\boldsymbol{v}^*) dt + \int \boldsymbol{v}^*\circ d\boldsymbol{s} \tag{5.18}$$
$$= -\int (V - E_0) dt + \log\left(\frac{\varphi_0(\boldsymbol{s}_T)}{\varphi_0(\boldsymbol{s}_0)}\right) \tag{5.19}$$

where we used Girsanov's theorem and Ito calculus. This proves (5.10) in the continuous case and using Girsanov's theorem again gives (5.13).

Proof: discrete state space

Rewriting the time-independent Schrödinger equation in terms of Γ^* gives

$$\sum_{\boldsymbol{s}'\neq\boldsymbol{s}} \Gamma^*_{\boldsymbol{s}\to\boldsymbol{s}'} = H_{\boldsymbol{s}\boldsymbol{s}} - E_0. \tag{5.20}$$

We can use this with the analog of Girsanov's theorem for continuous time Markov chains

$$\log\left(\frac{d\mathbb{P}_\theta}{d\mathbb{P}}(\boldsymbol{s}_t)\right) = \int \sum_{\boldsymbol{s}'\neq \boldsymbol{s}_t} \left(\Gamma_{\boldsymbol{s}_t\to\boldsymbol{s}'} - \Gamma^\theta_{\boldsymbol{s}_t\to\boldsymbol{s}'}\right) dt + \sum_n \log\left(\frac{\Gamma^\theta_{\boldsymbol{s}_n\to\boldsymbol{s}_{n+1}}}{\Gamma_{\boldsymbol{s}_n\to\boldsymbol{s}_{n+1}}}\right) \tag{5.21}$$

to compute the RN–derivative

$$\log\left(\frac{d\mathbb{P}_{\mathrm{FK}}}{d\mathbb{P}}\right) = \sum_n \left(-\sum_{\boldsymbol{s}'\neq \boldsymbol{s}_n} \left(\Gamma^*_{\boldsymbol{s}_n\to\boldsymbol{s}'} - \Gamma_{\boldsymbol{s}_n\to\boldsymbol{s}'}\right)\Delta t_n + \log\left(\frac{\Gamma^*_{\boldsymbol{s}_n\to\boldsymbol{s}_{n+1}}}{\Gamma_{\boldsymbol{s}_n\to\boldsymbol{s}_{n+1}}}\right)\right) \tag{5.22}$$

$$= -\sum_n \left(\left(V\left(\boldsymbol{s}_n\right) - E_0\right)\Delta t_n + \log\left(\frac{\varphi_0(\boldsymbol{s}_{n+1})}{\varphi_0(\boldsymbol{s}_n)}\right)\right) \tag{5.23}$$

$$= -\sum_n \left(V\left(\boldsymbol{s}_n\right) - E_0\right)\Delta t_n + \log\left(\frac{\varphi_0(\boldsymbol{s}_T)}{\varphi_0(\boldsymbol{s}_0)}\right). \tag{5.24}$$

Here we used the definition (2.14) of the potential $V(\boldsymbol{s}) = H_{\boldsymbol{ss}} - \sum_{\boldsymbol{s}'\neq\boldsymbol{s}} \Gamma_{\boldsymbol{ss}'}$. This proves (5.10) in the discrete case and using (5.21) again gives (5.14).

5.3 Generalized Holland's variational principle

This section follows [2] for the continuous state space case.

In this section we use (5.13) and (5.14) to derive a variational principle for the ground state drift/rates. In the infinite time limit, this results in another proof of the reinforcement learning objectives (5.1) and (5.2).

5.3.1 Finite time variational principle

The idea is to formulate a variational principle based on the Kullback–Leibler (KL) divergence between the Feynman–Kac measure and $\mathbb{P}_\theta$:

$$D_{\mathrm{KL}}(\mathbb{P}_\theta||\mathbb{P}_{\mathrm{FK}}) = \mathop{\mathbb{E}}_{P_\theta}\left[\log\left(\frac{d\mathbb{P}_\theta}{d\mathbb{P}_{\mathrm{FK}}}\right)\right]. \tag{5.25}$$

We have already computed the necessary Radon–Nikodym derivatives (5.13) and (5.14).

However, there are two unknowns in these expressions. The first is the ground state energy E_0. This can simply be dropped: because it is constant it does not affect the maximizer. Dropping the ground state energy, the function to be maximized in the continuous case is

$$R_T = -\frac{1}{T}\mathop{\mathbb{E}}_{P_\theta}\left[\int_0^T \left(V(\boldsymbol{s}_t) - \frac{1}{2}|\boldsymbol{v}^\theta(\boldsymbol{s}_t)|^2\right)dt + \int_0^T \boldsymbol{v}^\theta(\boldsymbol{s}_t)\cdot d\boldsymbol{s}_t - \log\left(\frac{\varphi_0(\boldsymbol{s}_T)}{\varphi_0(\boldsymbol{s}_0)}\right)\right]. \tag{5.26}$$

where the expectation is over trajectories following the drift $\boldsymbol{v}^\theta$. In the discrete case, the function to be maximized is

$$R_T = -\frac{1}{T}\mathop{\mathbb{E}}_{P_\theta}\left[\int V(\boldsymbol{s}_t) + \sum_{\boldsymbol{s}'\neq\boldsymbol{s}_t}(\Gamma_{\boldsymbol{s}_t\to\boldsymbol{s}'} - \Gamma^\theta_{\boldsymbol{s}_t\to\boldsymbol{s}'})\,dt \right. \\ \left. + \sum_n \log\left(\frac{\Gamma^\theta_{\boldsymbol{s}_n\to\boldsymbol{s}_{n+1}}}{\Gamma_{\boldsymbol{s}_n\to\boldsymbol{s}_{n+1}}}\right) - \log\left(\frac{\varphi_0(\boldsymbol{s}_T)}{\varphi_0(\boldsymbol{s}_0)}\right)\right] \tag{5.27}$$

where the expectation is over trajectories following the rates Γ^θ. Because E_0 was dropped, both (5.26) and (5.27) have a maximum value of $-E_0$.

The other unknown in this expression is the ground state wavefunction φ_0. If the initial states are drawn from the stationary distribution of $\boldsymbol{v}^\theta$ / Γ^θ, then the term involving the ground state wavefunction may also be dropped from (5.26) and (5.27). This leaves no unknowns so that (5.26) and (5.27) provide usable variational principles.

If a drift $\boldsymbol{v}^\theta$ or rates Γ^θ has been found that attains the maximal value $-E_0$, then these must be the ground state drift/rates. If a drift/rates maximizes R_T, then it also minimizes the KL–divergence $D_{\mathrm{KL}}(\mathbb{P}_\theta||\mathbb{P}_{\mathrm{FK}})$. Therefore, the measures $\mathbb{P}_\theta$ and $\mathbb{P}_{\mathrm{FK}}$ must coincide. Since $\mathbb{P}_{\mathrm{FK}}$ is the measure associated with the ground state drift/rates, the minimizers $\boldsymbol{v}^\theta$ and Γ^θ must be equal to the ground state drift/rates.

We end this section with the remark that since the expectation is over trajectories following the drift $\boldsymbol{v}^\theta$, we can also use $d\boldsymbol{s}_t = \boldsymbol{v}^\theta(\boldsymbol{s}_t)dt + d\boldsymbol{w}_t$ to rewrite R_T as

$$R_T = -\frac{1}{T}\mathop{\mathbb{E}}_{P_\theta}\left[\int_0^T \left(V(\boldsymbol{s}_t) + \frac{1}{2}|\boldsymbol{v}^\theta(\boldsymbol{s}_t)|^2\right)dt + \int_0^T \boldsymbol{v}^\theta(\boldsymbol{s}_t)\cdot d\boldsymbol{w}_t\right]. \tag{5.28}$$

5.3.2 Holland's variational principle

Now we can revisit the infinite time averaged objectives R (5.1) and (5.2). The continuous space objective (5.1) is the same as Holland's [26]. As we described in the introduction, Holland's paper is the only one explicitly relating the quantum ground state problem to optimal control/reinforcement learning. Holland proved the variational principle (5.1) by transforming the Schrödinger equation into a Hamilton–Jacobi–Bellman equation with a logarithmic transformation. We have already given an alternative proof using the discrete time connection of the previous chapter. Another route presents itself now: the infinite time limit of R_T (equations (5.26) and (5.27)) yields the objective R (we will prove this statement shortly) .

So the relation of our work with Holland's is as follows. Firstly, we have given two alternative proofs of Holland's variational principle. Secondly, we have extended Holland's variational principle to discrete state spaces and to finite time. In that sense, the previous section describes a generalization of Holland's variational principle.

Proof: continuous state space

Starting from (5.26), we see that in the limit $T \to \infty$, the term involving the ground state wavefunction φ_0 can be dropped so that

$$\lim_{T\to\infty} R_T = -\lim_{T\to\infty} \frac{1}{T} \mathop{\mathbb{E}}_{P_\theta} \left[\int_0^T \left(V(\boldsymbol{s}_t) - \frac{1}{2} |\boldsymbol{v}^\theta(\boldsymbol{s}_t)|^2 \right) dt + \int_0^T \boldsymbol{v}^\theta(\boldsymbol{s}_t) \cdot d\boldsymbol{s}_t \right] \tag{5.29}$$

even if the initial states are not drawn from the stationary distribution of $\boldsymbol{v}^\theta$. Furthermore, using the remark at the end of Section 5.3.1 and the fact that the Brownian increments $d\boldsymbol{w}_t$ have zero mean, we get

$$\lim_{T\to\infty} R_T = -\lim_{T\to\infty} \frac{1}{T} \mathop{\mathbb{E}}_{P_\theta} \left[-\int_0^T \left(V(\boldsymbol{s}_t) + \frac{1}{2} |\boldsymbol{v}^\theta(\boldsymbol{s}_t)|^2 \right) dt \right] \tag{5.30}$$

which is precisely the objective R (5.1).

Proof: discrete state space

The term involving the ground state wavefunction can again be dropped, giving

$$\lim_{T\to\infty} R_T = \lim_{T\to\infty} \frac{1}{T} \mathop{\mathbb{E}}_{P_\theta} \left[\int_0^T V(\boldsymbol{s}_t) + \sum_{\boldsymbol{s}' \neq \boldsymbol{s}_t} \left(\Gamma_{\boldsymbol{s}_t \to \boldsymbol{s}'} - \Gamma^\theta_{\boldsymbol{s}_t \to \boldsymbol{s}'} \right) dt + \sum_n \log \left(\frac{\Gamma^\theta_{\boldsymbol{s}_n \to \boldsymbol{s}_{n+1}}}{\Gamma_{\boldsymbol{s}_n \to \boldsymbol{s}_{n+1}}} \right) \right] \tag{5.31}$$

Moreover, one can show that

$$\mathop{\mathbb{E}}_{P_\theta} \left[\sum_n \log \left(\frac{\Gamma^\theta_{\boldsymbol{s}_n \to \boldsymbol{s}_{n+1}}}{\Gamma_{\boldsymbol{s}_n \to \boldsymbol{s}_{n+1}}} \right) \right] = \mathop{\mathbb{E}}_{P_\theta} \left[\sum_{\boldsymbol{s}' \neq \boldsymbol{s}_n} \Gamma^\theta_{\boldsymbol{s}_n \to \boldsymbol{s}'} \log \left(\frac{\Gamma^\theta_{\boldsymbol{s}_n \to \boldsymbol{s}'}}{\Gamma_{\boldsymbol{s}_n \to \boldsymbol{s}'}} \right) \Delta t_n \right]. \tag{5.32}$$

Together, these two equations imply that $\lim_{T\to\infty} R_T$ is equal to the objective R (5.2).

Chapter 6
Methods

To start with, we give an overview of the methods we will use for our experiments. Afterwards, the algorithms used to generate SDE trajectories and calculate the continuous space objective function is discussed in Section 6.2. The rest of the chapter deals with the neural network representations, optimization, and validation.

6.1 Overview

Everything was implemented using PyTorch. The code was developed in PyTorch collaboration with Prof Austen Lamacraft and Ariel Barr, and is available – for the continuum case – at https://github.com/AustenLamacraft/QuaRL. This is the code accompanying our paper [2].

In continuous space, we use the generalized Holland's variational principle of Section 5.3.1. This means we optimize a drift function $\boldsymbol{v}^\theta$ with respect to the objective function (5.26). To do this, we represent the drift $\boldsymbol{v}^\theta$ with a neural network and update it with stochastic gradient descent.

In discrete space, we use the discrete time reinforcement learning formulation of Chapter 4. We optimize an action-value function Q^θ to solve the soft Bellman optimality equation (4.12). To do this, we represent the action-value function with a neural network and optimize it with soft Q-learning. For the J_1-J_2 model, we do the same but then solve the complex soft Bellman optimality equation (4.25).

6.2 SDE schemes and errors

In order to calculate the objective function (5.26), we need to generate trajectories of duration T of the stochastic differential equation

$$d\boldsymbol{s}_t = \boldsymbol{v}^\theta(\boldsymbol{s}_t)dt + d\boldsymbol{w}_t. \tag{6.1}$$

The simplest algorithm to solve this SDE is the Euler–Maruyama algorithm

$$\boldsymbol{s}_{t+\Delta t} = \boldsymbol{s}_t + \boldsymbol{v}^\theta(\boldsymbol{s}_t)\Delta t + \Delta\boldsymbol{w}_t. \tag{6.2}$$

where $\Delta\boldsymbol{w}_t$ are i.i.d. Gaussian random variables with zero mean and a variance of Δt. To generate a trajectory of duration T, this step needs to be repeated $T/\Delta t$ times. On the other hand, using discrete time steps Δt introduces a discretization error that vanishes only

as $\Delta t \to 0$. Therefore, there is a careful balance between accuracy and efficiency when generating these trajectories.

There are two convenient properties of our problem. The first is that the SDE (6.1) has additive noise: there is no space-dependent factor in front of the Gaussian noise $d\boldsymbol{w}$ [32]. The second is that, to calculate (5.26), we only need expectations with respect to the trajectories, not the trajectories themselves. That is, we only need weak convergence [32].

These two convenient properties mean we can make use of algorithms more powerful than the Euler–Maruyama algorithm to generate the trajectories. In our experiments, we use the SOSRA algorithm [39]. Given a batch of B of these trajectories, we make a Monte-Carlo estimate of the objective function (5.26) as [2]

$$\hat{R}_T\left[\boldsymbol{v}^\theta\right] = \frac{1}{BT}\sum_{b,n}\left[\left(V\left(\boldsymbol{s}_n^{(b)}\right) + \frac{1}{2}\left|\boldsymbol{v}^\theta\left(\boldsymbol{s}_n^{(b)}\right)\right|^2\right)\Delta t + \boldsymbol{v}^\theta\left(\boldsymbol{s}_n^{(b)}\right)\cdot\Delta\boldsymbol{w}_n^{(b)}\right], \tag{6.3}$$

where $\boldsymbol{s}_n^{(b)}$ is the state after n discrete time steps of trajectory b. The initial states $\boldsymbol{s}_0^{(b)}$ are drawn from the approximate stationary distribution of the SOSRA discretization of the SDE with drift $\boldsymbol{v}^\theta$.

6.2.1 Bias and variance

In this section, we discuss the bias and variance of the estimator (6.3). This is useful for two reasons. Firstly, since (6.3) will be used as an estimator for the ground state energy, its bias must be known in order to report error estimates. Secondly, since (6.3) will be used as an objective function, knowledge of its bias and variance will help to inform hyperparameter tuning.

The bias of the estimator (6.3) has size $\mathcal{O}(\Delta t^p)$ (see proof below). Here p is the so-called weak order, which is $p = 1$ for Euler-Maruyama and $p = 2$ for SOSRA. There is an additional bias term if the trajectories are not started from the stationary distribution, but the discrete time process converges to its stationary distribution exponentially fast (see [41] for more details), so that this should not be a problem in practice.

The variance of (5.26) vanishes as $\boldsymbol{v}^\theta \to \boldsymbol{v}^*$ (see proof below), so we expect the variance of the estimator (6.3) to decrease significantly as $\boldsymbol{v}^\theta \to \boldsymbol{v}^*$, but not vanish completely because of discretization errors. Furthermore, the variance obviously decreases as $1/B$. Lastly, in practice the variance is independent of Δt if T is fixed, but decreases as $1/T$ for large T, and approaches a constant as $T \to 0$.

Proof of bias

We prove that the bias of the estimator (6.3) has size $\mathcal{O}(\Delta t^p)$ if the trajectories have been generated by an SDE scheme of weak order p. We assume that the initial states follow the stationary distribution of the SDE discretization.

So we need to show

$$\mathbb{E}\left[\hat{R}_T\left[\boldsymbol{v}^\theta\right]\right] = R_T\left[\boldsymbol{v}^\theta\right] + \mathcal{O}(\Delta t^p). \tag{6.4}$$

Consider the stationary distribution ρ_θ of the drift $\boldsymbol{v}^\theta$ and the stationary distribution $\hat{\rho}_\theta$ of the SDE scheme with the drift $\boldsymbol{v}^\theta$. By definition of the weak order p, expectations with respect to $\hat{\rho}_\theta$ have an error

$$\mathop{\mathbb{E}}_{\hat{\rho}_\theta}[g(\boldsymbol{s})] = \mathop{\mathbb{E}}_{\rho_\theta}[g(\boldsymbol{s})] + \mathcal{O}(\Delta t^p) \tag{6.5}$$

for any function g [32]. In particular, we have

$$\mathop{\mathbb{E}}_{\hat{\rho}_\theta}\left[\left(V(\boldsymbol{s}) + \frac{1}{2}\left|\boldsymbol{v}^\theta(\boldsymbol{s})\right|^2\right)\right] = \mathop{\mathbb{E}}_{\rho_\theta}\left[V(\boldsymbol{s}) + \frac{1}{2}\left|\boldsymbol{v}^\theta(\boldsymbol{s})\right|^2\right] + \mathcal{O}(\Delta t^p). \tag{6.6}$$

Because the initial states follow the stationary distributions, each of the states $\boldsymbol{s}_t$ of the trajectory also follow the stationary distributions, so that this is equivalent to

$$\mathbb{E}\left[\frac{1}{T}\sum_n\left[\left(V(\boldsymbol{s}_n) + \frac{1}{2}\left|\boldsymbol{v}^\theta(\boldsymbol{s}_n)\right|^2\right)\Delta t\right]\right] = \mathop{\mathbb{E}}_{P_\theta}\left[\frac{1}{T}\int_0^T\left(V(\boldsymbol{s}_t) + \frac{1}{2}\left|\boldsymbol{v}^\theta(\boldsymbol{s}_t)\right|^2\right)dt\right] + \mathcal{O}(\Delta t^p). \tag{6.7}$$

Because the Gaussian increments $\Delta\boldsymbol{w}$ and $d\boldsymbol{w}$ have zero mean, this is equivalent to what we have to prove.

Proof of vanishing variance

The variance of (5.26) vanishes as $\boldsymbol{v}^\theta \to \boldsymbol{v}^*$, because it involves the Radon–Nikodym derivative between the measures of $\boldsymbol{v}^\theta$ and $\boldsymbol{v}^*$. To be specific, as $\boldsymbol{v}^\theta \to \boldsymbol{v}^*$, the Radon–Nikodym derivative approaches the constant 1, so that the variance vanishes.

6.3 Neural network representation

6.3.1 Continuum space: PairDrift

This section follows [2].

In continuous space, we do experiments on indistinguishable particles obeying bosonic symmetry: i.e. the ground state wavefunction is invariant under a permutation of the indistinguishable particles. For the ground state drift, this implies equivariance with respect to any permutation P:

$$\boldsymbol{v}_i^\theta(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_N) = \boldsymbol{v}_{P(i)}^\theta(\boldsymbol{r}_{P(1)}, \ldots, \boldsymbol{r}_{P(N)}). \tag{6.8}$$

To achieve this equivariance, we represent the drift with a permutation equivariant neural network, PairDrift, adapted from [38]. Initially, single-particle and pairwise features are

formed according to $\mathbf{h}_i^{(0)} = \boldsymbol{r}_i$ and $\mathbf{h}_{ij}^{(0)} = \boldsymbol{r}_i - \boldsymbol{r}_j$. These features are then transformed using M layers of the form

$$\mathbf{h}_i^{(m+1)} = \mathrm{HardTanh}\left(\boldsymbol{\sigma}^{(m)}\left(\mathbf{h}_i^{(m)}\right) + \sum_j \boldsymbol{\pi}^{(m)}\left(\mathbf{h}_{ij}^{(m)}\right)\right) \tag{6.9}$$

$$\mathbf{h}_{ij}^{(m+1)} = \mathrm{HardTanh}\left(\boldsymbol{\Pi}^{(m)}\left(\mathbf{h}_{ij}^{(m)}\right)\right), \tag{6.10}$$

where $\boldsymbol{\sigma}^{(m)}$, $\boldsymbol{\pi}^{(m)}$ and $\boldsymbol{\Pi}^{(m)}$ are affine. The number of hidden nodes is fixed across the neural network. Furthermore, the HardTanh activation is omitted from the final layer. The final single-particle features give the drift i.e. $\mathbf{h}_i^{(M)} = \boldsymbol{v}_i$.

This architecture has the necessary equivariance because the affine transformations are the same for each particle.

Initialization

The affine transformations of the final layer are initialized to zero, after which a skip connection ensures a desirable initial drift of simple form. For example, for the Hydrogen atom we use a linear skip connection $-\mathbf{1}$. This makes sure the electron stays close to the nucleus.

For the Helium and Hydrogen molecule systems, we use Kato's cusp conditions [29] for the skip connection. These cusp conditions are originally meant for wavefunctions, but they translate straightforwardly into conditions for the ground state drifts: when the separation between two particles – interacting with a Coulomb interaction – goes to zero, the drift is proportional to $\hat{\boldsymbol{r}}$. When the two interacting particles are electrons, the drift is repulsive with a proportionality factor $\frac{1}{2}$, whereas for interactions between nuclei and electrons the drift should be attractive with a proportionality factor equal to the charge of the nucleus. Therefore, we use a sum of these terms $\hat{\boldsymbol{r}}$ as a skip connection, where each Coulomb interaction contributes one term for the sum.

6.3.2 Lattice models: PeriodicCNN

In discrete space, we will do experiments on spin models defined on a square lattice with periodic boundary conditions. Following [7, 8], we interpret a spin configuration $\boldsymbol{s}$ on such a lattice as a square grayscale image. Therefore, it is natural to consider convolutional neural networks (CNN), which are commonly used in deep learning applications dealing with image data.

The most important advantage of CNNs is that they are equivariant with respect to translations of the input. In image classification, for example, this means that the output label does not change if the object is on another part of the image. Many quantum spin models with periodic boundary conditions also have this feature: the ground state wavefunction is invariant with respect to translations of the lattice.

Fig. 6.1 The output $Q(\cdot|\boldsymbol{s})$ (right) of PeriodicCNN has the same shape as its input $\boldsymbol{s}$ (left).

We use a CNN to represent the action-value function Q^θ. In general, the action-value function takes a state $\boldsymbol{s}$ and an action $\boldsymbol{a}$ and maps it to a number $Q(\boldsymbol{s}, \boldsymbol{a})$. However, in discrete action settings it is more efficient to output the action-value $Q(\boldsymbol{s}, \boldsymbol{a})$ for all actions $\boldsymbol{a}$ given a state $\boldsymbol{s}$.[9] Thus our CNN has a spin configuration (a grayscale image) as its input, and needs to output a vector of values, one for each available action.

We use the transverse field Ising model as an example. In the z-basis, a spin configuration $\boldsymbol{s} = \{s_{ij}\}$ specifies simply whether the spin at site (i, j) is up (with value $s_{ij} = +1$) or down (with value $s_{ij} = -1$). This can be represented as a grayscale image with grayscale 1.0 for up and 0.0 for down.

What are the available actions? In the Ising model, all adjacent spin configurations $\boldsymbol{s}'$ can be reached by flipping one spin. So the available actions consist of flipping the spin at a site (i, j) or doing nothing (the trivial action). Ignoring the trivial action for now, this means that the output of our CNN has the same shape as its input: the output at site (i, j) is the action-value of flipping the spin at that site. This is illustrated in Figure 6.1.

We now describe PeriodicCNN: the simplest CNN architecture that fulfills the requirements of (1) preserving the shape, (2) being fully translationally equivariant and (3) outputs unbounded real values. PeriodicCNN consists of a variable number of layers, each of which in turn consists of a convolution and the ReLU activation function:

$$\text{CR-CR-}\ldots\text{-CR-C} \tag{6.11}$$

Each convolution uses 'circular half-padding' which makes sure that the convolution fulfills both requirements (1) and (2). Since the input is a grayscale image, there is a single in-going

channel. The number of hidden channels is kept constant across the CNN. The final layer only has a convolution, so no activation function, which makes sure the output values are unrestricted. The number of out-going channels is variable. Additionally, the last convolution is initialized to zero.

The variables of PeriodicCNN are (1) the number of layers and the number of hidden channels, controlling the expressiveness of the CNN, (2) the kernel size of the convolutional filters, and (3) the number of out-going channels. The number of out-going channels is fixed by the spin model under consideration. In the Ising case, there is a single out-going channel, and its value corresponds to the action of flipping the spin at that site. For the other models we experiment with, the number of out-going channels is either two or four.

The trivial action-value

Up to now, we have ignored the action-value of the trivial action, i.e. $Q(\boldsymbol{s}, \boldsymbol{a}_0)$. We have seen that this allows for a natural representation in terms of a shape-preserving CNN. Next, we show that the trivial action-value follows from the others, so that it does not need to be learned separately.

Observe that for the optimal action-value function Q^* (4.11), we have

$$\exp\left(Q^*(\boldsymbol{s}, \boldsymbol{a}_0)\right) = e^{r(\boldsymbol{s})+E_0\Delta t}\varphi_0(\boldsymbol{s}) \tag{6.12}$$

$$= \frac{e^{r(\boldsymbol{s})+E_0\Delta t}}{H_{\boldsymbol{ss}} - E_0}\left(\sum_{\boldsymbol{a}\neq\boldsymbol{a}_0} H_{\boldsymbol{sa}(\boldsymbol{s})}\varphi_0(\boldsymbol{a}(\boldsymbol{s}))\right) \tag{6.13}$$

$$= \frac{1}{H_{\boldsymbol{ss}} - E_0}\left(\sum_{\boldsymbol{a}\neq\boldsymbol{a}_0} H_{\boldsymbol{sa}(\boldsymbol{s})}\exp\left(Q^*(\boldsymbol{s}, \boldsymbol{a})\right)\right), \tag{6.14}$$

where we used (4.11) and the time-independent Schrödinger equation. We use this to complete our representation: if $Q^\theta(\boldsymbol{s}, \boldsymbol{a})$ is an approximation of the non-trivial action-values and E is an approximation of the ground state energy, then the approximation of the trivial action-value is defined by

$$\exp\left(Q^\theta(\boldsymbol{s}, \boldsymbol{a}_0)\right) \equiv \frac{1}{H_{\boldsymbol{ss}} - E}\left(\sum_{\boldsymbol{a}\neq\boldsymbol{a}_0} H_{\boldsymbol{sa}(\boldsymbol{s})}\exp\left(Q^\theta(\boldsymbol{s}, \boldsymbol{a})\right)\right). \tag{6.15}$$

This hinges on having an accurate estimate E of the ground state energy. This estimate is discussed in Section 6.4.2.

Marshall fixed sign structure for J_1-J_2

Following [8], we impose the Marshall fixed sign structure (2.28) for the J_1-J_2 model. Because the ground state of the J_1-J_2 does not exactly have this sign structure for $J_2 > 0$, this

introduces a fixed node error. Nevertheless, for small J_2, this error is small, and optimization becomes very challenging without this fixed sign structure.

In practice, we impose the fixed sign structure as follows. We use PeriodicCNN to predict the (positive) amplitude of the action-value function $Q(\boldsymbol{s}, \boldsymbol{a})$. For this step we only need real parameters. Next we add an imaginary part to $Q(\boldsymbol{s}, \boldsymbol{a})$: 0 if the Marshall sign of $\varphi_0(\boldsymbol{a}(\boldsymbol{s}))$ is $+1$ and $i\pi$ if the Marshall sign is -1. Then (4.24) ensures that this indeed fixes the Marshall sign structure.

6.4 Validation

If we have an approximation of the ground state drift or optimal action-value function, we need to have a manner of judging its quality. We do this validation by computing a variational energy from the drift/action-value approximation. A lower variational energy corresponds to a better drift/action-value approximation. Additionally, this variational energy serves as our estimate of the ground state energy E_0.

6.4.1 Continuous state space

In continuous space, following [2], we use the Monte-Carlo estimator (6.3) of (5.26) to validate a drift function $\boldsymbol{v}^\theta$. Before this estimator is calculated, there is a burn-in phase to make sure that the initial states are drawn from the stationary distribution of the drift $\boldsymbol{v}^\theta$. As discussed in Section 5.3.1, (5.26) has a maximum value of $-E_0$.

6.4.2 Discrete state space

In discrete space, we calculate the variational energy using the local energy. Recall that an approximation Q^θ of the action-value function defines a variational wavefunction φ^θ by (4.14) or (4.26). Following [8], we then calculate the variational energy as

$$\langle H \rangle = \mathop{\mathbb{E}}_{\boldsymbol{s} \sim |\varphi^\theta|^2} \left[\sum_{\boldsymbol{s}'} H_{\boldsymbol{s}\boldsymbol{s}'} \frac{\varphi^\theta(\boldsymbol{s}')}{\varphi^\theta(\boldsymbol{s})} \right] \tag{6.16}$$

and sample states $\boldsymbol{s} \sim |\varphi^\theta|^2$ using a Metropolis algorithm. In contrast with [8], we use the MALA algorithm described in Section (2.2.3) using the proposal policy

$$\pi^\theta(\boldsymbol{a}|\boldsymbol{s}) \propto \begin{cases} \left| H_{\boldsymbol{s}\boldsymbol{a}(\boldsymbol{s})} \exp\left(Q^\theta(\boldsymbol{s}, \boldsymbol{a}) \right) \right| & \text{if } \boldsymbol{a} \neq \boldsymbol{a}_0 \\ 0 & \text{if } \boldsymbol{a} = \boldsymbol{a}_0. \end{cases} \tag{6.17}$$

A Monte-Carlo estimate of (6.16) then gives an approximation of the variational energy, and by extension, of the ground state energy.

6.5 Optimization

6.5.1 Continuous space: policy gradient

This section follows [2].

The objective function (5.1) is approximated with the Monte Carlo estimator (6.3). This is analogous to the reparameterization trick: the expectation (5.1) with respect to $\mathbb{P}_\theta$ is written in terms of an expectation with respect to Gaussian random variables $\Delta \boldsymbol{w}$.[52] Therefore, the gradient of the Monte Carlo estimator (6.3) with respect to the parameters θ of the drift neural network $\boldsymbol{v}^\theta$ can be computed directly using backpropagation. Note that normally, policy gradients are calculated indirectly, e.g. using the REINFORCE estimator.

The Monte Carlo estimator (6.3) does not include the boundary term $\log\left(\frac{\varphi_0(\boldsymbol{s}_T)}{\varphi_0(\boldsymbol{s}_0)}\right)$ which is part of (5.26). Although this term vanishes from the expectation if the initial points are drawn from the stationary distribution, the gradient of this term does not. To deal with this, we use the approximation

$$\nabla_\theta \log\left(\frac{\varphi_0(\boldsymbol{s}_T)}{\varphi_0(\boldsymbol{s}_0)}\right) = \nabla_\theta \boldsymbol{s}_T \cdot \boldsymbol{v}^*(\boldsymbol{s}_T) \approx \nabla_\theta \boldsymbol{s}_T \cdot \boldsymbol{v}^\theta(\boldsymbol{s}_T). \tag{6.18}$$

Therefore, the final approximation for the policy gradient is

$$\nabla_\theta R_T = \nabla_\theta \hat{R}_T - \nabla_\theta \boldsymbol{s}_T \cdot \boldsymbol{v}^\theta(\boldsymbol{s}_T) \tag{6.19}$$

where $\hat{R}_T$ is the Monte Carlo estimator (6.3). The bias of this approximation vanishes as $\Delta t \to 0$ and $\boldsymbol{v}^\theta \to \boldsymbol{v}^*$.

With this final policy gradient approximation, the parameters θ of the drift $\boldsymbol{v}^\theta$ are updated using ADAM [31].

6.5.2 Discrete space: soft Q-learning

In discrete space, we optimize the action-value function Q^θ to solve the soft Bellman equation (3.11b). Optimization is done using the soft Q-learning algorithm [19] with a few adjustments. In short, the adjustments are:

1. Changes for finite action spaces
   Sampling actions and calculating expectations can be done exactly in finite action spaces, so there is no need for a separate actor network.

2. The variational energy parameter E
   Our Q-representation has the additional parameter E, which is updated using (5.27).

3. Different sampling of states and actions
   States are sampled $\sim |\varphi^\theta|^2$ as described in Section 6.4.2, actions from (6.17).

4. Variance instead of mean squared error
   We use the variance instead of the MSE in the loss function (6.23). This is appropriate for our infinite horizon averaged objective (3.10). [47]

A summary of the resultant procedure is given in Algorithm 1. In the following, we discuss it in more detail. Apart from the listed adjustments, this algorithm is due to [19]. Moreover, we will discuss states and actions in the singular, but in practice, a batch of states and actions is processed in parallel.

Algorithm 1: Adjusted soft Q-learning

Initialize θ, $\bar{\theta}$, $\boldsymbol{s}_0$, E
for *each episode* **do**
  Get experience
    $\boldsymbol{s} \sim \left|\varphi^{\theta}\right|^2 (\cdot|\boldsymbol{s}_0)$
    $\boldsymbol{s}_0 \leftarrow \boldsymbol{s}$
    $\boldsymbol{a} \sim \pi_\theta(\cdot|\boldsymbol{s})$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{\boldsymbol{s}, \boldsymbol{a}, r(\boldsymbol{s}), \boldsymbol{a}(\boldsymbol{s})\}$
  Update E
    $E \leftarrow \langle H \rangle$
    $E \leftarrow \min\{E, H_{\boldsymbol{ss}}\}$
  Update θ
    $\{\boldsymbol{s}, \boldsymbol{a}, r(\boldsymbol{s}), \boldsymbol{a}(\boldsymbol{s})\} \sim \mathcal{D}$
    $\theta \leftarrow \theta + \alpha \nabla_\theta J$
    Every M **episodes do**
      $\bar{\theta} \leftarrow \theta$
end

Initialization

The adjusted soft Q-learning algorithm learns the parameters θ, $\bar{\theta}$ and E_0. As in the standard soft Q-learning algorithm, θ and $\bar{\theta}$ parameterize action-value functions Q^θ and $Q^{\bar{\theta}}$. Here $Q^{\bar{\theta}}$ is known as the target network. The parameters θ are initialized using the standard initialization procedures of PyTorch, except for the last layer which is initialized to zero. The parameters of the target network $\bar{\theta}$ are then initialized to equal θ.

Additionally, the algorithm keeps track of a parameter E, which is the current variational energy. This parameter should be initialized to any value larger than the ground state energy.

Finally, the state $\boldsymbol{s}_0$ is initialized according to a uniform distribution. In the systems we consider, it is known which spin configurations $\boldsymbol{s}$ satisfy $\varphi_0(\boldsymbol{s}) \neq 0$. The initialization distribution is chosen to be uniform over all such spin configurations.

Experience

The soft-Q learning algorithm says that experience can be sampled in any way, as long as every possible state and action has a change of being sampled. They then propose a sampling method with an additional actor network. As we have a discrete action space, we do it as follows.

The state $\boldsymbol{s}$ is obtained using a fixed number N of MALA steps as described in Section 6.4.2, starting from the initial state $\boldsymbol{s}_0$. In this way, $\boldsymbol{s}$ approximately samples from $|\varphi^\theta|^2$. The initial state $\boldsymbol{s}_0$ is then updated to $\boldsymbol{s}$, so that eventually also $\boldsymbol{s}_0$ approximately samples from $|\varphi^\theta|^2$.

The actions are chosen following the policy (6.17). Sampling from this policy is easy because we have a finite action space. The trivial action $\boldsymbol{a}_0$ never needs to be selected because its action-value follows from the others, as described in Section 6.3.2.

As is standard, the experience, consisting of a state $\boldsymbol{s}$, an action $\boldsymbol{a}$, the obtained reward $r(\boldsymbol{s})$ and the new state $\boldsymbol{a}(\boldsymbol{s})$ are saved in the experience replay $\mathcal{D}$. We discard the oldest experiences if the size of the experience replay $\mathcal{D}$ grows beyond a certain threshold.

Update E

The state $\boldsymbol{s}_0$ gets used to update the variational energy E. Because this approximately samples from $|\varphi^\theta|^2$, it can be used to calculate $\langle H\rangle$ with (6.16). Then E is updated to $\langle H\rangle$.

Additionally, if any state has $H_{\boldsymbol{ss}} < E$, then E is decreased below $H_{\boldsymbol{ss}}$, so that it always satisfies

$$E < \min_{\boldsymbol{s}\in\mathcal{D}}\{H_{\boldsymbol{ss}}\}. \tag{6.20}$$

Update Q

Firstly, an experience is sampled from the experience replay $\mathcal{D}$. The experience is then used to update θ as follows. Recall the soft Bellman optimality equation (4.12)

$$Q^*(\boldsymbol{s},\boldsymbol{a}) = r(\boldsymbol{s}) + E_0\Delta t + \log \mathop{\mathbb{E}}_{\boldsymbol{a}'\sim p_{\Delta t}(\cdot|\boldsymbol{a}(\boldsymbol{s}))} \left[\exp\left(Q^*(\boldsymbol{a}(\boldsymbol{s}),\boldsymbol{a}')\right)\right]. \tag{6.21}$$

Because E_0 is constant, we can rewrite this as

$$\mathrm{Var}_{\boldsymbol{s},\boldsymbol{a}}\left[Q^*(\boldsymbol{s},\boldsymbol{a}) - r(\boldsymbol{s}) - \log \mathop{\mathbb{E}}_{\boldsymbol{a}'\sim p_{\Delta t}(\cdot|\boldsymbol{a}(\boldsymbol{s}))} \left[\exp\left(Q^*(\boldsymbol{a}(\boldsymbol{s}),\boldsymbol{a}')\right)\right]\right] = 0. \tag{6.22}$$

In soft Q-learning, this is turned into the loss function

$$J(\theta) = \mathrm{Var}_{\boldsymbol{s},\boldsymbol{a}\sim\mathcal{D}}\left[Q^\theta(\boldsymbol{s},\boldsymbol{a}) - r(\boldsymbol{s}) - \log \mathop{\mathbb{E}}_{\boldsymbol{a}'\sim p_{\Delta t}(\cdot|\boldsymbol{a}(\boldsymbol{s}))} \left[\exp\left(Q^{\bar\theta}(\boldsymbol{a}(\boldsymbol{s}),\boldsymbol{a}')\right)\right]\right]. \tag{6.23}$$

Note that the target network $Q^{\bar{\theta}}$ appears in the expectation. Furthermore, the expectation can be computed exactly [9] because we have a finite action space:

$$\mathop{\mathbb{E}}_{\boldsymbol{a}' \sim p_{\Delta t}(\cdot|\boldsymbol{a}(\boldsymbol{s}))} \left[\exp\left(Q^{\bar{\theta}}(\boldsymbol{a}(\boldsymbol{s}), \boldsymbol{a}') \right) \right] = \sum_{\boldsymbol{a}'} p_{\Delta t}(\boldsymbol{a}'|\boldsymbol{a}(\boldsymbol{s})) \exp\left(Q^{\bar{\theta}}(\boldsymbol{a}(\boldsymbol{s}), \boldsymbol{a}') \right) \tag{6.24}$$

So the sampled experience is used to calculate the loss function (6.23) and its gradient with respect to θ. This gradient is then used to update θ with ADAM. Finally, the target network $Q^{\bar{\theta}}$ is updated every M episodes to match Q^{θ}.

For the complex soft Bellman optimality equation (4.25), we obtain the same loss function (6.23), but the expectation $\mathbb{E}$ is replaced with $\tilde{\mathbb{E}}$ as defined by (4.22).

Chapter 7
Results

We now present our results. Our main benchmark is the comparison of our variational energies with numerically exact ground state energies. This gives a general indication of the quality of the ground state approximation.

7.1 Hydrogen and Helium atom, Hydrogen molecule

This section follows [2].

All energy results are summarized in Table 7.1, where we continue to use Hartree atomic units $\mathrm{E}_h = 1$. We compare ours with numerically exact and Hartree–Fock values (if available). The difference between the latter two is commonly called the correlation energy. We recover around 90% of this correlation energy and are always within $0.006\,\mathrm{E}_h$ of the numerically exact ground state energy.

Recall that the hydrogen atom has ground state energy $E_0 = -\frac{1}{2}$ and ground state drift $\boldsymbol{v}(\boldsymbol{s}) = -\hat{\boldsymbol{s}}$. In Figure 7.1 our optimized drift $\boldsymbol{v}^\theta$ is plotted in the $x - y$ plane. It closely resembles the ground state drift, as it has an almost uniform magnitude and is directed towards the nucleus. This drift has a variational energy of -0.497 so is 0.6% from the exact ground state energy.

For the Helium atom, which has ground state energy -2.904, [3] our optimized drift recovers 86% of the correlation energy.

Finally, the ground state of the Hydrogen molecule has internuclear distance $R = 1.401$ and ground state energy $E_0 = -1.174$. [33] At this separation, we recover 88% of the correlation energy. If we fix the internuclear distance at $R = 2.800$, i.e. approximately twice the equilibrium separation, then we get within $3\mathrm{mE}_h$ of the numerically exact ground state energy. In Figure 7.1 we plot the electron density distribution at separation $R = 2.800$ resulting from our optimized drift. To be more precise, it shows a few thousand states, each resulting from running the stochastic differential equation with our optimized drift. In this way, Figure 7.1 illustrates how we can sample from the ground state.

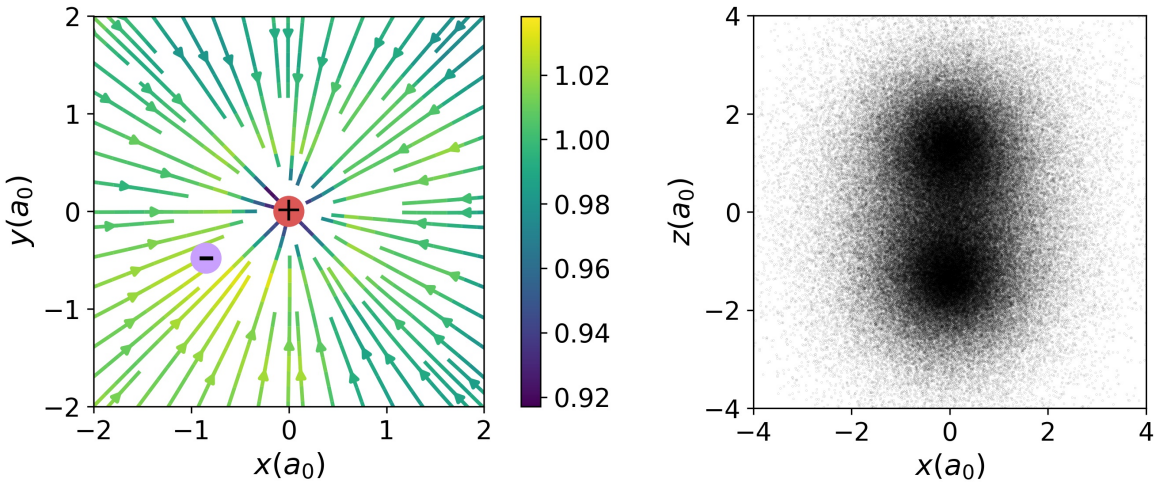


Fig. 7.1 (Left) The learned drift field for the Hydrogen atom in the $x-y$ plane, colorscaled by its magnitude. (Right) The electron density distribution in the stretched H_2 molecule at proton separation $R = 2.8$. Both figures are reprinted from [2].

	H atom	He atom	H_2 molecule	H_2 molecule ($R = 2.8$)
Numerically exact	-0.5	-2.904 [3]	-1.174 [33]	-1.071 [33]
Hartree-Fock	N/A	-2.862 [3]	-1.134 [12]	
Ours	-0.497 (0.6%)	-2.898 (0.3%)	-1.169 (0.4%)	-1.068 (0.3%)
% corr	N/A	86%	88%	

Table 7.1 Comparison of our approach with Hartree-Fock and numerically exact values for the ground state energies. The percentages within parentheses are the relative errors of our energy estimate compared to the numerically exact value. The last row shows how much of the correlation energy is recovered by our optimized drift.

	Ising	XY	AFH	J_1-J_2
Numerically exact	-1.0638 [21]	-1.10539 [22]	-0.6789 [45]	-0.5990 [45]
Ours	-1.0628	-1.10530	-0.6750	-0.5896
Relative error	0.09%	0.008%	0.6%	1.6%

Table 7.2 Comparison of our approach with numerically exact values for the ground state energies. All lattices have size 6×6. The Ising model is at the pseudo-critical point $J \approx 0.327583$, the J_1-J_2 model is evaluated at $J_2 = 0.2$.

7.2 Spin lattice models

We present results for square lattices of size 6×6 for the transverse field Ising model (2.16), the XY model (2.21), the antiferromagnetic Heisenberg model (2.26) and the J_1-J_2 model (2.34), all with periodic boundary conditions. All energy results are summarized and compared with numerically exact values in Table 7.2. The relative error is between 0.008% for the XY model and 1.6% for the J_1-J_2 model.

Of course we can also measure quantities other than the ground state energy. To illustrate, we measure the modulus of the magnetization and the spin-spin correlation for the transverse field Ising model. We calculated the modulus of the magnetization per spin

$$m = \frac{2}{N} \left| \sum_i s_i \right| \tag{7.1}$$

averaged over our optimized wavefunction for the 4×4 Ising model for a range of interactions J near the pseudo-critical point $J_{c,4} = 0.324$. We also calculated the spin-spin correlation

$$S^2(j_x, j_y) = \sum_i s_i s_{(i_x + j_x, i_y + j_y)} \tag{7.2}$$

averaged over our optimized wavefunction for the 6×6 transverse Ising model at the pseudo-critical point $J_{c,6} = 0.328$. Here a site i is identified with its coordinates (i_x, i_y) so that $S^2(j_x, j_y)$ measures the correlation between sites that are separated by a displacement (j_x, j_y).

We plotted both the modulus of the magnetization and the spin-spin correlation in Figure 7.2. For the magnetization, you see that it increases with the interaction J, although for this small lattice you do not see a true phase transition. For the spin-spin correlation, you can see that it is periodic and invariant to swapping the 'x' and 'y' axes, as it should be. It also decreases with distance, as it should.

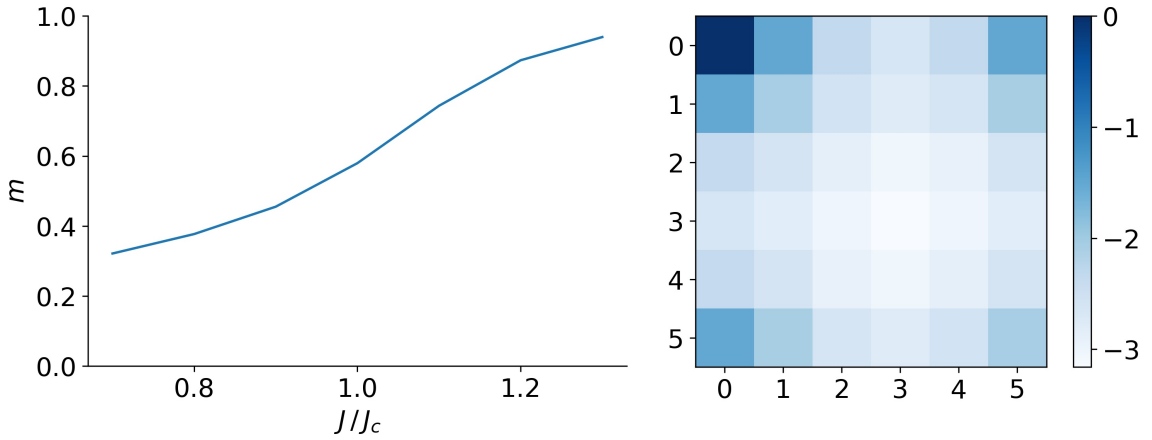


Fig. 7.2 (Left) The modulus of the magnetization per spin (7.1) of a 4×4 Ising model near the pseudo-critical point. (Right) The log of the spin-spin correlation (7.2) of a 6×6 Ising model at the pseudo-critical point.

Chapter 8
Discussion

Before making comparisons with other approaches and providing an outlook for future work, we summarize the main achievements of this work. Firstly, we made the theoretical contribution of reformulating the quantum ground state problem as a reinforcement learning problem, for both continuous and discrete state spaces and in both continuous and discrete time. Previous work only did this in continuous space and time. Like previous work, our approach is mainly applicable to stoquastic Hamiltonians. Secondly, we conducted proof of principle experiments to show that deep reinforcement learning algorithms can indeed solve this reformulated quantum ground state problem.

8.1 Comparison with other methods

In Section 1.2, we reviewed deep learning approaches to the quantum ground state problem. In this section we compare our approach for discrete state spaces with the stochastic reconfiguration method [8].

The downsides of our method are as follows. Firstly, reinforcement learning algorithms are known for relatively unstable, data-hungry training, although soft off-policy methods suffer less from these issues. Secondly, image-to-image networks such as ours intuitively need to be larger, with more parameters, than image-to-scalar networks as used by stochastic configuration. Lastly, the state of the art in stochastic reconfiguration has obtained more accurate ground state energies so far, although this could have many reasons.

On the other hand, our approach learns from extremely local and off-policy data. To improve the action-value function, only pairs of adjacent spin configurations are needed, and these pairs in theory do not need to be generated in any special way. Stochastic reconfiguration, however, needs to calculate a Monte Carlo average over its parameterized wavefunction for every update step.

Another advantage of our method comes up at inference time: our method can sample more efficiently from the ground state. The reason is that our 'MALA' algorithm – only feasible when you have access to an action-value function – is more efficient than Metropolis–Hastings with a uniform proposal distribution (e.g. for the XY model acceptance probabilities increased from 75% to 99%).

8.2 Comparison between our continuous and discrete time methods

We have shown that the quantum ground state problem is equivalent to both a discrete and a continuous time reinforcement learning problem. The discrete time perspective leads naturally to a Q-learning method, the continuous time perspective to a policy gradient method. How do they compare?

As discussed before, the Q-learning method learns from local and off-policy data. In contrast, the policy gradient method learns from on-policy trajectories. This makes the Q-learning method more sample-efficient and less memory intensive.

Whereas the Q-learning method obtains the ground state wavefunction, the policy gradient method as we presented it does not. This means that the policy gradient method can only calculate expectations of on-diagonal observables. If the policy gradient method is complemented with value estimation, however, this issue would vanish.

A standard advantage of policy gradient is that it is more principled. Policy gradient directly optimizes the objective function, making it more stable and easier to understand and work with.

Finally, in continuous space both methods have issues in the limit of small time steps $\Delta t \to 0$. The action-value function Q contains no information as $\Delta t \to 0$ because it is the same for each action. Advantage learning for example may alleviate this problem. On the other hand, as discussed in Section 6.2.1, the variance of the policy gradient increases as $\Delta t \to 0$, slowing down convergence.

8.3 Outlook

We finish with an outlook for future work.

Reinforcement learning algorithms

There are many types of reinforcement learning algorithms so many ways to improve upon our relatively simple soft Q-learning and policy gradient techniques. Model-based methods such as MuZero [44] could boost performance, considering we have a perfect model of our environment. For small discrete time steps, advantage learning [1] could resolve the issue of continuous space discussed above.

For continuous time, value-based algorithms explicitly designed for the continuous time setting such as those for semi-Markov decision processes [4] may be explored. There are also policy rollout methods that promise better performance than policy gradient methods.[49]

In continuous space, state-value estimation would give access to the ground state wavefunction. This would allow the calculation of non-diagonal observables and enable the MALA

algorithm of Section 2.2.2. A suitable candidate is the soft actor-critic algorithm [20], which includes state-value estimation and is a state-of-the-art method for continuous action spaces.

Neural network representation

Next, we discuss possible improvements for the PeriodicCNN architecture that we used for the discrete state space experiments. As we have noted before, the action-value function Q can be seen as an image-to-image transformation. Therefore, the architectures used in image-to-image problems, such as encoder-decoder networks [27] may be particularly beneficial.

Moreover, quantum spin models often have symmetries beyond translation equivariance. In fact, all models we have considered obey dihedral symmetry, which may be built into a CNN [10, 13]. Spin-flip equivariance may be built in by computing bond variables in the first layer of the CNN.

Other extensions

We end with somewhat speculative suggestions for extensions beyond optimization and representation. Firstly, there already exist many quantum Monte Carlo methods that in principle obtain the quantum ground state (properties) with arbitrary accuracy. It would be interesting to investigate whether our method can be combined with any of those existing methods to make them faster.

Secondly, our method is mostly confined to stoquastic Hamiltonians. Ideally, our method would be extended to deal with ground states of general Hamiltonians. As a steppingstone towards that goal, a fixed node version of our method may possibly be obtained from the fixed-node Feynman–Kac formula [6].

Lastly, our methods use continuous time or small discrete time steps, whereas most modern reinforcement learning algorithms are designed for discrete time steps. A proper discrete time formulation would therefore be highly desirable. An initial exploration of this direction suggests that a proper discrete time formulation is indeed possible for discrete state spaces.

References

[1]L. Baird, "Reinforcement learning in continuous time: advantage updating", in Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94), Vol. 4 (June 1994), 2448–2453 vol.4.

[2]A. Barr, W. Gispen, and A. Lamacraft, "Quantum ground states from reinforcement learning", in Mathematical and scientific machine learning (PMLR, 2020), pp. 635–653.

[3]A. L. Baskerville, A. W. King, and H. Cox, "Electron correlation in Li+, He, H and the critical nuclear charge system ZC: energies, densities and Coulomb holes", Royal Society Open Science **6** (2019).

[4]S. J. Bradtke and M. O. Duff, "Reinforcement learning methods for continuous-time Markov decision problems", in Proceedings of the 7th International Conference on Neural Information Processing Systems, NIPS'94 (Jan. 1994), pp. 393–400.

[5]S. Bravyi, "Monte Carlo simulation of stoquastic Hamiltonians", arXiv:1402.2295 [quant-ph] (2015).

[6]M. Caffarel, "Talking Across Fields: A Physicist's Presentation of some Mathematical Aspects of Quantum Monte Carlo Methods", Annales de la Faculté des sciences de Toulouse : Mathématiques **24**, 949–972 (2015).

[7]G. Carleo and M. Troyer, "Solving the quantum many-body problem with artificial neural networks", Science **355**, 602–606 (2017).

[8]K. Choo, T. Neupert, and G. Carleo, "Two-dimensional frustrated J1-J2 model studied with neural network quantum states", Physical Review B **100**, 125124 (2019).

[9]P. Christodoulou, "Soft Actor-Critic for Discrete Action Settings", arXiv:1910.07207 [cs, stat] (2019).

[10]T. S. Cohen and M. Welling, "Group Equivariant Convolutional Networks", arXiv:1602.07576 [cs, stat] (2016).

[11]E. Crosson, *Discussion on stoquastic Hamiltonians*, Nov. 2017.

[12]E. R. Davidson and L. L. Jones, "Natural Expansion of Exact Wavefunctions. II. The Hydrogen-Molecule Ground State", The Journal of Chemical Physics **37**, 2966–2971 (1962).

[13]S. Dieleman, J. De Fauw, and K. Kavukcuoglu, "Exploiting Cyclic Symmetry in Convolutional Neural Networks", arXiv:1602.02660 [cs] (2016).

[14]K. Dvijotham and E. Todorov, "A unifying framework for Linearly Solvable control", in Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence, UAI'11 (July 2011), pp. 179–186.

[15]H. Ezawa, J. R. Klauder, and L. A. Shepp, "A path space picture for Feynman-Kac averages", Annals of Physics **88**, 588–620 (1974).

[16]A. Galashov, S. M. Jayakumar, L. Hasenclever, D. Tirumala, J. Schwarz, G. Desjardins, W. M. Czarnecki, Y. W. Teh, R. Pascanu, and N. Heess, "Information asymmetry in KL-regularized RL", arXiv:1905.01240 [cs, stat] (2019).

[17]A. Goldberg and J. L. Schwartz, "Integration of the Schrödinger equation in imaginary time", Journal of Computational Physics **1**, 433–447 (1967).

[18]F. Guerra and L. M. Morato, "Quantization of dynamical systems and stochastic control theory", Physical Review D **27**, 1774–1786 (1983).

[19]T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, "Reinforcement Learning with Deep Energy-Based Policies", in International Conference on Machine Learning (July 2017), pp. 1352–1361.

[20]T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor", arXiv:1801.01290 [cs, stat] (2018).

[21]C. J. Hamer, "Finite-size scaling in the transverse Ising model on a square lattice", Journal of Physics A: Mathematical and General **33**, 6683–6698 (2000).

[22]C. J. Hamer, T. Hövelborn, and M. Bachhuber, "Finite-size scaling in the spin- model on a square lattice", Journal of Physics A: Mathematical and General **32**, 51–59 (1999).

[23]J. Han, L. Zhang, and W. E, "Solving Many-Electron Schrödinger Equation Using Deep Neural Networks", Journal of Computational Physics **399**, 108929 (2019).

[24]A. Harju, B. Barbiellini, S. Siljamäki, R. M. Nieminen, and G. Ortiz, "Stochastic Gradient Approximation: An Efficient Method to Optimize Many-Body Wave Functions", Physical Review Letters **79**, 1173–1177 (1997).

[25]J. Hermann, Z. Schätzle, and F. Noé, "Deep neural network solution of the electronic Schrödinger equation", arXiv:1909.08423 [physics, stat] (2020).

[26]C. J. Holland, "A new energy characterization of the smallest eigenvalue of the Schroedinger equation", Communications in Pure Applied Mathematics **30**, 755–765 (1977).

[27]P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-Image Translation with Conditional Adversarial Networks", arXiv:1611.07004 [cs] (2018).

[28]H. J. Kappen, "Path integrals and symmetry breaking for optimal control theory", Journal of Statistical Mechanics: Theory and Experiment **2005**, P11011–P11011 (2005).

[29]T. Kato, "On the eigenfunctions of many-particle systems in quantum mechanics", Communications on Pure and Applied Mathematics **10**, 151–177 (1957).

[30]J. Kessler, F. Calcavecchia, and T. D. Kühne, "Artificial Neural Networks as Trial Wave Functions for Quantum Monte Carlo", arXiv:1904.10251 [physics] (2019).

[31]D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization", arXiv:1412.6980 [cs] (2017).

[32]P. E. Kloeden and E. Platen, *Numerical Solution of Stochastic Differential Equations*, Stochastic Modelling and Applied Probability (Springer-Verlag, Berlin Heidelberg, 1992).

[33]W. Kolos and L. Wolniewicz, "Potential-Energy Curves for the X 1g+, b3u+, and C 1u States of the Hydrogen Molecule", The Journal of Chemical Physics **43**, 2429–2441 (1965).

[34]I. E. Lagaris, A. Likas, and D. I. Fotiadis, "Artificial neural network methods in quantum mechanics", Computer Physics Communications **104**, 1–14 (1997).

[35]S. Levine, "Reinforcement learning and control as probabilistic inference: tutorial and review", arXiv preprint arXiv:1805.00909 (2018).

[36]W. Marshall and R. E. Peierls, "Antiferromagnetism", Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences **232**, 48–68 (1955).

[37]Y. Nomura, A. S. Darmawan, Y. Yamaji, and M. Imada, "Restricted Boltzmann machine learning for solving strongly correlated quantum systems", Physical Review B **96**, 205152 (2017).

[38]D. Pfau, J. S. Spencer, A. G. d. 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 (2019).

[39]C. Rackauckas and Q. Nie, "Stability-Optimized High Order Methods and Stiffness Detection for Pathwise Stiff Stochastic Differential Equations", arXiv:1804.04344 [math] (2018).

[40]G. O. Roberts and J. S. Rosenthal, "Optimal scaling of discrete approximations to Langevin diffusions", Journal of the Royal Statistical Society: Series B (Statistical Methodology) **60**, 255–268 (1998).

[41]G. O. Roberts and R. L. Tweedie, "Exponential Convergence of Langevin Distributions and Their Discrete Approximations", Bernoulli **2**, 341–363 (1996).

[42]L. C. G. Rogers and D. Williams, *Diffusions, Markov Processes, and Martingales: Volume 1: Foundations*, 2nd ed., Vol. 1, Cambridge Mathematical Library (Cambridge University Press, Cambridge, 2000).

[43]H. Saito, "Solving the Bose–Hubbard Model with Machine Learning", Journal of the Physical Society of Japan **86**, 093001 (2017).

[44]J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, and D. Silver, "Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model", arXiv:1911.08265 [cs, stat] (2020).

[45]H. J. Schulz, T. A. L. Ziman, and D. Poilblanc, "Magnetic order and disorder in the frustrated quantum Heisenberg antiferromagnet in two dimensions", Journal de Physique I **6**, 675–703 (1996).

[46]S. Sorella, M. Casula, and D. Rocca, "Weak binding between two aromatic rings: Feeling the van der Waals attraction by quantum Monte Carlo methods", The Journal of Chemical Physics **127**, 014105 (2007).

[47]R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. (MIT Press, 2018).

[48]E. Theodorou, J. Buchli, and S. Schaal, "A Generalized Path Integral Control Approach to Reinforcement Learning", Journal of Machine Learning Research **11**, 3137–3181 (2010).

[49]E. Theodorou, J. Buchli, and S. Schaal, "Learning Policy Improvements with Path Integrals", in Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (Mar. 2010), pp. 828–835.

[50]E. Todorov, "Efficient computation of optimal actions", Proceedings of the National Academy of Sciences **106**, 11478 (2009).

[51]N. Trivedi and D. M. Ceperley, "Green-function Monte Carlo study of quantum antiferromagnets", Physical Review B **40**, 2737–2740 (1989).

[52]B. Tzen and M. Raginsky, "Neural Stochastic Differential Equations: Deep Latent Gaussian Models in the Diffusion Limit", arXiv:1905.09883 [cs, stat] (2019).

[53]T. Westerhout, N. Astrakhantsev, K. S. Tikhonov, M. I. Katsnelson, and A. A. Bagrov, "Generalization properties of neural network approximations to frustrated magnet ground states", Nature Communications **11**, 1593 (2020).

[54]K. Yasue, "Quantum mechanics and stochastic control theory", Journal of Mathematical Physics **22**, 1010–1020 (1981).

Appendix A

Hyperparameters

For the continuous state space experiments, we used an exponential decay of the learning rate: the learning rate was multiplied by a factor 0.95 every 10 training steps. Trajectories had 2^{10} time steps of size $\Delta t = 0.01$. The initial learning rates, as well as other training hyperparameters can be found in Table A.1.

For the discrete state space experiments, we used an initial learning rate of 10^{-3}. For the XY model, this was exponentially decayed by a factor 0.99 every 20 episodes, for the other models it was kept constant. Batches had size 2^{10}, the replay memory had size 2^{16}. Between episodes, 64 Metropolis–Hastings steps were taken. The target network was updated every 20 episodes. The time step Δt had size 10^{-4}. The neural network architecture differed a bit per model, see Table A.2.

	H atom	He atom	H_2 molecule	H_2 molecule ($R = 2.8$)
Batch size	2^{10}	2^{10}	2^{10}	2^{10}
Initial learning rate	10^{-2}	10^{-3}	5×10^{-4}	10^{-2}
Width of hidden layer	256	64	64	64

Table A.1 Hyperparameters used in continuous state space experiments.

	Ising	XY	AFH	J_1-J_2
Kernel size	3	3	3	5
Number of layers	3	3	5	3
Width of hidden layer	32	32	64	32

Table A.2 Neural network architecture used in discrete state space experiments.