Quantum Tensor Networks for Variational Reinforcement Learning

Xiao-Yang Liu[1*], **Yiming Fang**[2*]
[1]Department of Electrical Engineering, [2]Department of Computer Science,
Columbia University, New York, NY 10025
{xl2427, yf2484}@columbia.edu

Abstract

As a major machine learning paradigm, reinforcement learning has recently been found to have enormous potential in applications across disciplines. However, existing approaches may require exhaustive exploration to converge, e.g., an exponential number of transitions. In this paper, we propose a novel quantum tensor network approach for variational reinforcement learning. First, we derive a variational reinforcement learning framework by reformulating the Bellman equation into the Hamiltonian equation. The task of searching for the optimal policy in the state-action space is equivalent to finding the ground state (lowest-energy state) of a quantum-mechanical system. Then, we propose a quantum tensor network scheme that approximates the policy with matrix product state (MPS), alleviating the curse of dimensionality for searching in a huge state-action space. On the Gridworld puzzle example, we empirically show that our algorithm has better stability and higher inference accuracy from partial observations, compared with Q-learning.

1 Introduction

Reinforcement learning (RL) is one of the most promising and fast-developing field in machine learning. In a generic RL setting, an autonomous agent interacts with an environment, often modeled as a Markov Decision Process (MDP), to learn an optimal set of behaviors by adjusting its behaviors according to the environment's response. Recently, big data and the growth of computational resources have enabled the agent to navigate through more complex environments, helping RL achieve remarkable successes in many tasks. RL has been proved to be of great value for a wide range of applications, such as robotics [10], financial markets [5], operations research [6], game theory [17], etc. One of the best-known example of RL's success is DeepMind's AlphaGo and AlphaZero algorithms [23], which defeated human world-champions of Go, a notoriously difficult chess game.

However, in the development of RL, there are as many challenges as there are successes. First, traditional methods based on dynamic programming are known to suffer from instability during training [1, 14]. Whether the function approximation converges successfully depends heavily on the careful choice of hyper-parameters and reward functions [8]. Second, since the computation and storage complexities for MDP are often combinatorially large, RL faces the curse of dimensionality problem [1]. Third, the reproducibility, reusability, and robustness in DRL are questioned by numerous researchers [9].

In recent years, researchers have explored variational algorithms to improve RL by solving the Hamiltonian equation with the quantum annealing method [4, 22]. It has also been shown that tensor networks are a powerful tool for solving the ground state of quantum spins and lattice models [13].

[*] Equal contribution.

First Workshop on Quantum Tensor Networks in Machine Learning, 34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

Tensor networks have been used to solve finite MDP problems using DMRG approach [7]. There are also numerous works done on the application of tensor networks on supervised learning and neural networks [12, 16].

In this paper, we propose to address the challenges of RL using a variational optimization scheme, with the aid of tensor networks. By demonstrating that there are fundamental similarities between MDP and quantum states, we recast MDP to an energy minimization problem. Then, exploiting tensor networks' ability to model many-body problems with polynomial-order parameters, we propose a joint algorithm based on Alternating Least Squares (ALS) and Stochastic Gradient Descent (SGD) to carry out the estimation of the reward tensor and energy minimization simultaneously, utilizing efficient tensor network operations. We apply this algorithm to the classic task of Gridworld, and show that our algorithm is able to stably obtain the optimal policy.

The remainder of this paper is organized as follows. Section 2 describes the background of RL and tensor networks. In Section 3, we reformulate the Bellman equation into a Hamiltonian equation. Section 4 describes a tensor network solution for the Hamiltonian equation. Section 5 presents performance evaluations. Section 6 concludes the paper.

2 Background and Preliminaries

We describe the background of MDP and tensor networks.

Notations: We denote tensors by calligraphic letters, e.g., $\mathcal{A}$. Let $\otimes$ stand for the tensor (Kronecker) product, $\times_k$ for mode-k product (a.k.a, tensor contraction, Einstein sum), $\circledast$ for the Hadamard (element-wise) product, and $\langle \cdot\ ,\ \cdot \rangle$ for inner product. We use order-N for N-way tensors, and rank[2] for the maximum number of linearly independent vectors in a tensor. Let $[n]$ denote the set $\{1, 2, \cdots, n\}$.

2.1 Markov Decision Process and Bellman Equation

RL algorithms train the agent to interact with an environment, such that it chooses a sequence of actions that promise the maximum expected rewards. This dynamic decision making process is formulated as a MDP denoted by a five-tuple (S, A, P, R, γ), where S stands for the state space, A for the action space, P for the transitional probability, R for reward function, and $\gamma \in (0, 1)$ for a discount factor.

The goal of the agent is to learn the optimal policy $\pi^* : S \times A \to [0, 1]$, which is a probability distribution over the allowed actions at a state, so that the (discounted) expected reward is maximized. Define the Q-value as expected reward of a state-action pair (s, a) under policy π

$$Q(\pi|s, a) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid s_t = s, a_t = a \right], \tag{1}$$

where R_{t+k+1} denotes the direct reward of taking action a_{t+k} at state s_{t+k}.

The Bellman equation gives the optimality condition for MDP problems

$$Q(\pi|s, a) = R^a_{s,s'} + \gamma \sum_{a'} \pi(s', a') Q(\pi|s', a'), \tag{2}$$

which expresses the expected reward at s as a summation of direct reward $R^{a'}_{s,s'}$ and discounted future reward at s'. The optimal policy is given by

$$\pi^* = \arg\max_{\pi}\ Q(\pi|s, a). \tag{3}$$

2.2 Tensor Networks and Matrix Product State

Tensor network is a powerful tool that enables efficient approximation of quantum and mechanical systems. Tensor networks represent multilinear mapping using interconnected tensors. This operation, often known as tensor contraction, closely correlates with the concept of entanglement and interaction

[2] In quantum physics, it is called bond dimension.

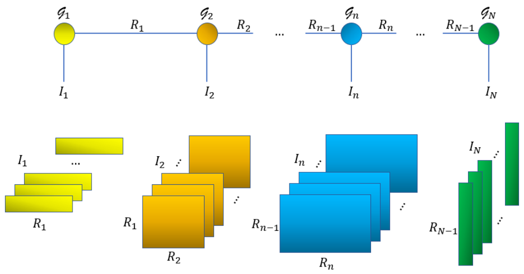

Figure 1: Tensor networks.

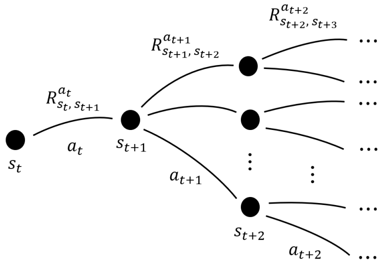

Figure 2: State transition diagram of MDP.

between particles in physical systems. We write the mode-(m, n) contraction between order-N tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and order-M tensor $\mathcal{B} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_M}$ as $\mathcal{C} = \mathcal{A} \times_n^m \mathcal{B}$, where $I_n = J_m$ and $\mathcal{C}_{i_{p \neq n}, j_{q \neq m}} = \sum_{i_n=1}^{I_n} \mathcal{A}_{i_1,\ldots,i_N} \mathcal{B}_{j_1,\ldots,j_M}$. Note that $\mathcal{C}$ is an order-$(M + N - 2)$ tensor. Therefore, it is much more space-efficient to store $\mathcal{A}$ and $\mathcal{B}$ in a connected format than to store $\mathcal{C}$ directly, and contracting them only when needed. Tensor networks have a highly intuitive graph representation [19]. As shown by Fig. 1, the modes are represented as edges. Two nodes connected by a common edge means the corresponding tensors are contracted in that mode. This representation allows us to manipulate tensor networks with complex topology without diving into the mathematical formalism.

There are numerous schemes that decompose a high-order tensor into a tensor network, including Parafac [15], Tucker [21], MPS [20], etc. In this paper, we will resort to the MPS format, as shown in Fig. 1. The tensor cores are connected via contraction into a chain that approximates the original high-order tensor.

3 Problem Formulation for Variational Reinforcement Learning

We reformulate the Bellman equation into a Hamiltonian equation and obtain a variational framework for reinforcement learning.

3.1 Variational Reinforcement Learning as Hamiltonian Equation

We would like to point out that the following derivations are inspired by [4] and (8) is consistent with (10) in [4] after replacing the stochastic transition P to an deterministic indicator function $\mathbb{I}$. Please note that the general connection between the Bellman equation and the Hamiltonian equation was widely known for both the continuous case [27] and the discrete case [18].

For brevity, let $\pi_t = \pi(s_t, a_t)$. We expand the recursion in (2) and obtain

$$Q(\pi|s_t, a_t) = R^{a_t}_{s_t,s_{t+1}} + \gamma \sum_{a_{t+1}} R^{a_{t+1}}_{s_{t+1},s_{t+2}} \pi_{t+1} + \gamma^2 \sum_{a_{t+1}} \sum_{a_{t+2}} R^{a_{t+2}}_{s_{t+2},s_{t+3}} \pi_{t+1}\pi_{t+2} + \cdots , \tag{4}$$

where rewards are accumulated along a trajectory $(s_t, a_t) \to (s_{t+1}, a_{t+1}) \to (s_{t+2}, a_{t+2}) \to \cdots$. This process is illustrated in Fig. 2. Since s_t and a_t are random variables, (4) has implicitly imposed an assumption that a transition $s_t \xrightarrow{a_t} s_{t+1}$ exists on the trajectory. Alternatively, when the environment is deterministic, we insert an indicator function on each transition and express each term of (4) over the domain $S \times A$ as follows

$$\begin{aligned} Q(\pi|s_t, a_t) = \sum_{s_{t+1}}^{S} \mathbb{I}^{a_t}_{s_t,s_{t+1}} \Bigg(& R^{a_t}_{s_t,s_{t+1}} + \gamma \sum_{a_{t+1}}^{A} \sum_{s_{t+2}}^{S} \mathbb{I}^{a_{t+1}}_{s_{t+1},s_{t+2}} \Big(R^{a_{t+1}}_{s_{t+1},s_{t+2}} \pi_{t+1} + \\ & \gamma^2 \sum_{a_{t+1}}^{A} \sum_{a_{t+2}}^{A} \sum_{s_{t+3}}^{S} \mathbb{I}^{a_{t+2}}_{s_{t+2},s_{t+3}} \big(R^{a_{t+2}}_{s_{t+2},s_{t+3}} \pi_{t+1}\pi_{t+2} + \cdots \big) \Big) \Bigg), \end{aligned} \tag{5}$$

where $R^{a_{t+q}}_{s_{t+q},s_{t+q+1}} = 0$ if the transition $s_{t+q} \xrightarrow{a_{t+q}} s_{t+q+1}$ is not allowed, and

$$\mathbb{I}^{a_q}_{s_q,s_{q+1}} = \begin{cases} 1, \text{ if } s_q \xrightarrow{a_q} s_{q+1}, \\ 0, \text{ otherwise}, \end{cases} \tag{6}$$

where $\mathbb{I}^{a_q}_{s_q,s_{q+1}}$ can be replaced by a probability $\mathbb{P}^{a_q}_{s_q,s_{q+1}}$ for a probabilistic transition $s_q \xrightarrow{a_q} s_{q+1}$ when the environment is stochastic.

Distribute the summation operation in (5) to each term and obtain

$$\begin{aligned} Q(\pi|s_t, a_t) = \sum_{s_{t+1}}^{S} \mathbb{I}^{a_t}_{s_t,s_{t+1}} R^{a_t}_{s_t,s_{t+1}} + \gamma \sum_{\mu_{t+1}}^{S\times A} \sum_{s_{t+2}}^{S} \mathbb{I}^{a_t}_{s_t,s_{t+1}} \mathbb{I}^{a_{t+1}}_{s_{t+1},s_{t+2}} R^{a_{t+1}}_{s_{t+1},s_{t+2}} \pi(\mu_{t+1}) + \cdots + \\ \gamma^k \sum_{\mu_{t+1}}^{S\times A} \cdots \sum_{\mu_{t+k}}^{S\times A} \sum_{s_{t+k+1}}^{S} \left(\prod_{q=0}^{k} \mathbb{I}^{a_{t+q}}_{s_{t+q},s_{t+q+1}} \right) R^{a_{t+k}}_{s_{t+k},s_{t+k+1}} \pi(\mu_{t+1}) \cdots \pi(\mu_{t+k}), \end{aligned} \tag{7}$$

where $\mu_p = (s_p, a_p)$ for $p = t+1, ..., t+k$, and the look-ahead steps $K = \infty$ for the case of infinity horizon MDP. In practice, we may consider a finite series by setting K to be a small number, e.g. set $K = 3$, since γ^k decays quickly and larger K would incur larger computational cost.

In (3), one aims to find the policy that maximizes Q-values for all state-action pairs, not one particular pair (s, a). Taking a variational approach, we define a summation of Q-values over state-action pairs in $S \times A$, group similar terms in (7), and derive a compact representation

$$\begin{aligned} \sum_{s_t,a_t}^{S\times A} Q(\pi|s_t, a_t) &= \mathcal{C}^{(0)} + \gamma \sum_{\mu_{t+1}}^{S\times A} \mathcal{C}^{(1)}_{\mu_{t+1}} \pi(\mu_{t+1}) + \gamma^2 \sum_{\mu_{t+1}}^{S\times A} \sum_{\mu_{t+2}}^{S\times A} \mathcal{C}^{(2)}_{\mu_{t+1},\mu_{t+2}} \pi(\mu_{t+1})\pi(\mu_{t+2}) + \cdots \\ &= \mathcal{C}^{(0)} + \sum_{k=1}^{K} \sum_{\mu_{t+1},\ldots,\mu_{t+k}}^{S\times A} \mathcal{C}^{(k)}_{\mu_{t+1},\ldots,\mu_{t+k}} \pi(\mu_{t+1})\pi(\mu_{t+2}) \cdots \pi(\mu_{t+k}), \end{aligned} \tag{8}$$

where $\mathcal{C}^{(k)} \in \mathbb{C}^{|S\times A|^k}$ is a reward tensor, and an entry $\mathcal{C}^{(k)}_{\mu_{t+1},\ldots,\mu_{t+k}}$ is a discounted reward at s_{t+k},

$$\mathcal{C}^{(k)}_{\mu_{t+1},\ldots,\mu_{t+k}} = \gamma^k \sum_{s_t,a_t}^{S\times A} \sum_{s_{t+k+1}}^{S} \left(\prod_{q=t}^{t+k} \mathbb{I}^{a_q}_{s_q,s_{q+1}} \right) R^{a_{t+k}}_{s_{t+k},s_{t+k+1}}, \tag{9}$$

where $s_t \xrightarrow{a_t} s_{t+1}$ indicates a transition from state s_t to s_{t+1}, and the discounting is taken backward along a trajectory $(s_1, a_1) \leftarrow (s_2, a_2) \leftarrow \cdots \leftarrow (s_k, a_k)$. Note that $\mathcal{C}^{(0)}$ is a constant offset denoting the sum of immediate rewards following all (s_t, a_t), which is not multiplied with π because it is independent of the policy.

Note that (8) has the same form as the Hamiltonian Equation. According to (3), we derive the optimal policy by minimizing a Hamiltonian functional of π as $H(\pi) = -\sum_{s,a} Q(\pi|s, a)$,

$$\pi^* = \arg\min_{\pi}\ H(\pi), \tag{10}$$

which searches for the minimal energy of the quantum system underlying $H(\pi)$. This equality is reached when the variation of $H(\pi)$ with regard to π approaches zero [3],

$$\frac{\delta H(\pi)}{\delta \pi} = \frac{\delta \left(\sum_{s_t,a_t} Q(\pi|s_t, a_t) \right)}{\delta \pi} = \sum_{s_t,a_t} \frac{\delta Q(\pi|s_t, a_t)}{\delta \pi} = 0. \tag{11}$$

The set of reward tensors $\{\mathcal{C}^{(k)}\}_{k=1}^{K}$ contain exponentially many parameters, but we can make a structural assumption that the coefficient tensors are low-rank, and therefore can be accurately estimated using smaller tensor cores. This assumption is plausible because states that are one or few steps away from each other share a large portion of their trajectories, so that from one trajectory, one can infer information about a family of states. Moreover, for most applications, the connectivity between states is sparse, so many entries in the reward tensors are zeros.

3.2 Physical Interpretation

The Bellman equation corresponds to Hamiltonian equation not only in mathematical formalism, but also in physical interpretation It is well-known that the Hamiltonian represents the total energy of a system, i.e., the sum of its potential and kinetic energy [2]. Therefore, the Hamiltonian is often used to find the optimal solutions for systems of motion that obey the principle of least action[3].

The task in finding the optimal policy for a MDP that results in the maximum rewards is analogous to finding an optimal path in space that incurs the least energy penalty. We can interpret a particular state-action as a particle in motion, the immediate reward following it as its momentum, and the future expected rewards down the path as its potential energy. Therefore, the optimal policy dictates a path that obeys the principle of least action by maximizing the (negative) sum of kinetic and potential energies. It may also be observed that the probability constraint in the policy, i.e., $\sum_a \pi(s,a) = 1, \forall s \in S$ corresponds to the condition of unitarity in quantum systems.

In both cases, the solution is a trajectory that best economizes resources offered by the environment. Therefore, it is physically plausible to solve the Bellman Equation as a quantum system using a variational approach.

4 Energy Minimization Algorithm Using Tensor Networks

We propose a novel algorithm using variational tensor networks that searches for the minimal energy of a Hamiltonian Equation. We also analyze the cost of exploration and propose a method for efficient exploration.

4.1 Basic Idea for Algorithm Design

Drop the constant term $-\mathcal{C}^{(0)}$, and we simplify $H'(\pi)$ as follows

$$\begin{aligned} H'(\pi) &= -\sum_{k=1}^{K}\left(\sum_{\mu_{t+k}}^{S\times A}\left(\cdots\left(\sum_{\mu_{t+1}}^{S\times A}\mathcal{C}^{(k)}_{\mu_{t+1},\ldots,\mu_{t+k}}\pi(\mu_{t+1})\right)\cdots\right)\pi(\mu_{t+k})\right) \\ &= -\sum_{k=1}^{K}\left(\left(\cdots\left(\mathcal{C}^{(k)}\times_1\pi\right)\times_2\cdots\right)\times_k\pi\right) = -\sum_{k=1}^{K}\left\langle\mathcal{C}^{(k)},\ \underbrace{(\pi\otimes\pi\otimes\cdots\otimes\pi)}_{k\text{ times}}\right\rangle. \end{aligned} \tag{12}$$

Note that first-order tensors (vectors), the tensor product $\otimes$ is equivalent to the outer product $\circ$, and $\pi \in \mathbb{R}^{|S\times A|}$ defined over $S \times A$ satisfies the probability constraint.

$$H(\pi) = -\sum_{k=1}^{K}\left\langle\mathcal{C}^{(k)},\ \underbrace{(\pi\otimes\pi\otimes\cdots\otimes\pi)}_{k\text{ times}}\right\rangle$$

4.2 Efficient Random Exploration

In practice, it is difficult to obtain the reward tensors $\{\mathcal{C}^{(k)}\}_{k=1}^{K}$ from an unknown environment, which has exponentially many parameters. We can explore the environment to obtain a set of observations $\Omega \subseteq [S \times A]^K$, and define a set of binary indicator tensors $\mathcal{P}^{(k)}_{\Omega} \in \mathbb{R}^{|S\times A|^k}$, for $k = 1, ..., K$, to specify the positions of observed state-action pairs. For all values of k, we construct reward tensor $\mathcal{C}^{(k)}_{\Omega}$ on partial observations Ω and use it to obtain an estimate of the full reward tensor $\widehat{\mathcal{C}^{(k)}}$.

We describe an exploration scheme as follows. In order to construct $\mathcal{C}^{(k)}_{\Omega}$, the agent starts at a random state-action pair (s_{t+1}, a_{t+1}) and explores a random trajectory of length k, for $k = 1, ..., K$. We denote the number of incoming transitions to s_{t+1} as $N_t = \sum_{s_t, a_t}\mathbb{I}_{s_t, s_{t+1}}$ and calculate the sum of rewards for all possible s_{t+2} as $R_{t+1} = \sum_{s_{t+2}} R^{a_{t+1}}_{s_{t+1}, s_{t+2}}$. We then randomly choose a s_{t+2} as the next state, and repeat the same calculations until the length of the trajectory reaches k. Denote an observed trajectory as $\mu_{t+1} \to \mu_{t+2} \to \cdots \to \mu_{t+k}$, we use the collected information to update the

[3] The principle of least action – or, more accurately, the principle of stationary action – is a variational principle that, when applied to the action of a mechanical system, can be used to obtain the equations of motion.

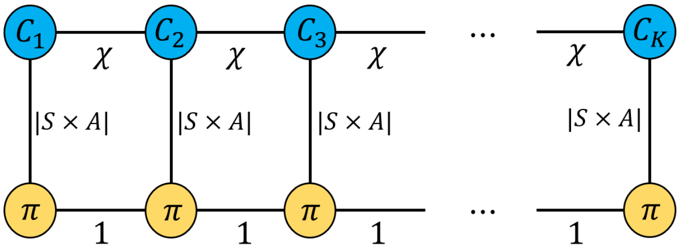

Figure 3: Contraction of two MPS.

partial reward tensor $\mathcal{C}^{(k)}_{\Omega}$ as follows

$$\mathcal{C}^{(k)}_{\mu_{t+1},\ldots,\mu_{t+k}} = \sum_{s_t,a_t} \mathbb{I}_{s_t,s_{t+1}} \left(\gamma^k \sum_{s_{t+k+1}} R^{a_{t+k}}_{s_{t+k},s_{t+k+1}}\right) = \gamma^k N_t R_{t+k}, \tag{13}$$

where $\mu_{t+k} \in S \times A, \forall k$. Note that we can update a parameter in $\mathcal{C}^{(k)}_{\Omega}$ by exploring one trajectory consisted of k steps, and each step in the trajectory costs $|A|$ transitions.

We improve the above scheme by taking L steps in each trajectory, with $L \gg K$. For each explored trajectory, we use all sub-trajectories of length k in a rolling fashion to update $\mathcal{C}^{(k)}_{\Omega}$ and $\mathcal{P}^{(k)}_{\Omega}$, for $k = 1, ..., K$. After obtain the N and R values for every transition, let

$$\mathcal{C}^{(k)}_{\mu_{t+p+1},\ldots,\mu_{t+p+k}} = \gamma^k N_{t+p} R_{t+p+k} \quad , \quad \mathcal{P}^{(k)}_{\mu_{t+p+1},\ldots,\mu_{t+p+k}} = 1, \tag{14}$$

for $p = 0, ..., L - k$ and $\mu_{t+p+k} \in S \times A, \forall p, k$. This modified scheme allows each explored trajectory to be reused $\sum_{k=1}^{K}(L-k)$ times, reducing the cost of updating each entry in $\mathcal{C}^{(k)}_{\Omega}$ to $\sum_{k=1}^{K} \frac{L}{L-k}|A| = O(\frac{|A|}{K})$.

Our proposed exploration scheme requires the environment to be deterministic, because one has to gain knowledge about the transitional probabilities and rewards through exploration. However, it can be extended to stochastic environment by sampling a pool of M transitions to obtain an estimate of the transitional probability. Notice that this operation incurs computational cost, and the complexity of exploration increases to $O(\frac{M|A|}{K})$.

4.3 Tensor Network Representation

According to (12), the Hamiltonian is now representable using tensor networks. The top part of Fig. 3 shows the approximation of reward tensor $\mathcal{C}^{(k)}_{\Omega}$ using a set of cores $\{\mathcal{C}^{(k)}_1, \mathcal{C}^{(k)}_2, ..., \mathcal{C}^{(k)}_k\}$ in MPS format. The bottom part of Fig. 3 shows how the outer product of k copies of π vectors is connected into a symmetric tensor network with MPS structure, which is constructed by adding two dummy dimensions (edges) of dimension 1 to the π vectors to form tensors $\pi \in \mathbb{R}^{1\times|S\times A|\times 1}$ and connecting these tensors along their edges with dimension 1. This operation gives a cascade structure because now each π is a tensor core, whose dangling edge represents the current policy.

Having obtained representations for the reward and policy using tensor networks, we connect the two MPS's on every dangling edge. This connection operation corresponds to the inner product in (12), giving us a tensor network that calculates a scalar output for the total energy $H'(\pi)$ when contracted. One thing to note is the order of contraction is extremely important for the efficiency of the algorithm, and the contraction path needs to be chosen carefully.

4.4 Algorithm for Joint Tensor Completion and Energy Minimization

We use an joint ALS and SGD algorithm to perform tensor completion and energy minimization simultaneously, as given in Alg. 1.

The algorithm first updates each core in the tensor network given in Fig. 3. We randomly initialize K tensor networks in MPS format. For $k = 1, ..., K$, the k-th tensor network contain cores

Algorithm 1 Variational reinforcement learning using tensor networks

```
1: Input: $S \times A$, $\gamma$, $K$, $\chi$, $\alpha$, and $\lambda$.
2: Output: An approximately optimal policy $\pi \in \mathbb{R}^{|S\times A|}$.
3: Randomly initialize a matrix $\Pi \in \mathbb{R}^{|S|\times|A|}$.
4: Perform softmax for each row, i.e., $\Pi_{ij} = \exp(\Pi_{ij}) \Big/ \sum_{j\in[|A|]} \exp(\Pi_{ij}),\ i \in [|S|],\ j \in [|A|]$.
5: Vectorize $\Pi$ into a vector $\pi$ and add two dummy dimension of 1 to obtain $\pi \in \mathbb{R}^{1\times|S\times A|\times 1}$.
6: Explore the environment by making a random trajectory of length $L$ by taking $L|A|$ transitions.
7: for $k = 1, ..., K$ do
8:     for each explored step in the trajectory do
9:         Update tensor $\mathcal{C}_\Omega^{(k)} \in \mathbb{R}^{|S\times A|^k}$ and binary tensor $\mathcal{P}_\Omega^{(k)} \in \mathbb{R}^{|S\times A|^k}$ via (14).
10:    end
11:    Randomly initialize a set of tensor cores $\{\mathcal{C}_1^{(k)}, \mathcal{C}_2^{(k)}, ..., \mathcal{C}_k^{(k)}\}$ as in (15).
12:    Connect $k$ copies of $\pi$ into MPS, and connect $\{\mathcal{C}_1^{(k)}, \mathcal{C}_2^{(k)}, ..., \mathcal{C}_k^{(k)}\}$ into MPS, as in Fig 3.
13: end
14: while not converged do
15:    for $k = 1, ..., K$ do
16:        Update $\mathcal{C}_j^{(k)}$ via (16), for $j = 1, ..., k$.
17:        Estimate $\widehat{\mathcal{C}^{(k)}} \leftarrow \left(\times_3^1\right)_{j=1}^k \mathcal{C}_j^{(k)}$.
18:    end
19:    Compute $\bar{H}(\pi)$ via (17).
20:    Update $\pi$ via (18).
21:    Organize $\pi$ into a matrix $\Pi \in \mathbb{R}^{|S|\times|A}$. Repeat Lines 4 and 5 to normalize $\pi$.
22: end
23: Return $\pi$
```

$\{\mathcal{C}_1^{(k)}, \mathcal{C}_2^{(k)}, ..., \mathcal{C}_k^{(k)}\}$ with

$$\mathcal{C}_1^{(k)} \in \mathbb{R}^{1\times|S\times A|\times\chi},\ \mathcal{C}_k^{(k)} \in \mathbb{R}^{\chi\times|S\times A|\times 1},\ \mathcal{C}_j^{(k)} \in \mathbb{R}^{\chi\times|S\times A|\times\chi},\ \text{for } j = 2, ..., k-1, \tag{15}$$

where $\chi \in \mathbb{Z}^+$ is the internal dimension in each tensor core. We apply tensor completion algorithm [25] to this tensor network and obtain an estimate $\widehat{\mathcal{C}^{(k)}}$ using ALS algorithm. For the core tensor $\mathcal{C}_j^{(k)}$, we fix all other cores and update $\mathcal{C}_j^{(k)}$ by

$$\mathcal{C}_j^{(k)} \leftarrow \arg\min_{\mathcal{C}_j^{(k)}} \left\| \mathcal{P}_\Omega^{(k)} \circledast \left(\times_3^1\right)_{i=1}^k \mathcal{C}_i^{(k)} - \mathcal{C}_\Omega^{(k)} \right\|_F^2, \tag{16}$$

where we obtain the updated $\mathcal{C}_j^{(k)}$ by solving a least square problem [11].

Then, we optimize the policy π by minimizing the energy $H'(\pi)$ of the quantum system using SGD. To ensure that the probability constraint holds during training, we combine the energy function with a regularization term to obtain a new energy function $\bar{H}(\pi)$ as follows

$$\bar{H}(\pi) = -\sum_{k=1}^{K} \left\langle \mathcal{C}^{(k)},\ \underbrace{(\pi \otimes \pi \otimes \cdots \otimes \pi)}_{k \text{ times}} \right\rangle + \lambda \sum_{s=1}^{|S|} \left(1 - \sum_{a=1}^{|A|} \pi(s,a)\right)^2, \tag{17}$$

where $\lambda \in \mathbb{R}^+$ is the regularization constant, a hyper-parameter that controls the strength of regularization during training.

Compute the variation of the energy function with regard to π, $\delta\bar{H}(\pi)/\delta\pi$, and update π by SGD optimization scheme, using variation instead of gradient as follows

$$\pi \leftarrow \pi - \alpha \frac{\delta \bar{H}(\pi)}{\delta\pi}, \tag{18}$$

where $\alpha \in \mathbb{R}^+$ is the learning rate that controls the amount of update during each iteration.

The above steps describe one iteration of the joint algorithm for tensor completion and energy minimization, which is repeated until the error between the true tensor and the estimation is smaller than ϵ_1 and $\delta\bar{H}(\pi)/\delta\pi < \epsilon_2$, where $\epsilon_1, \epsilon_2 \in \mathbb{R}$ are small threshold values.

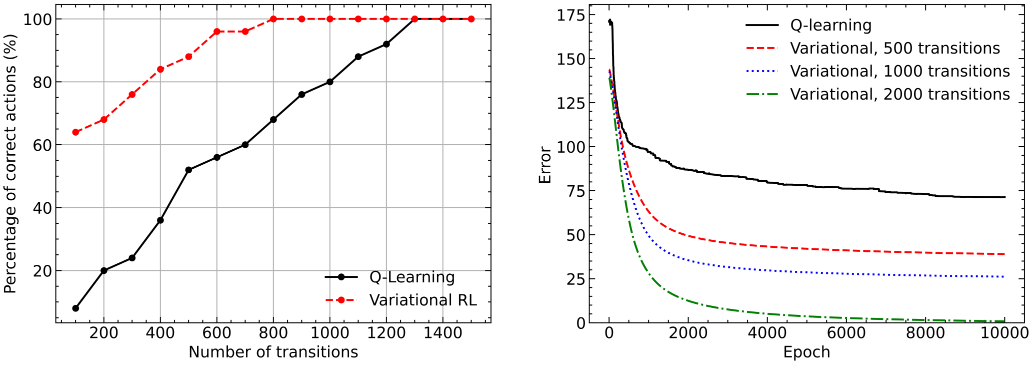

Figure 4: Comparison of variational RL and Q-Learning on Gridworld.

4.5 Advantages

The advantage of variational RL over traditional methods is twofold. First, the optimization process has greater stability and faster convergence because the objective function is evaluated over the global state-action space, rather than over a few particular state-action pairs as in dynamic programming based algorithms. Second, the variational optimization error is bounded by $\gamma^{K+1}\bar{H}(\pi^*)$ as time goes to infinity, and the convergence rate is exponential. This rate is astonishingly quick, since most algorithms converge at linear or quadratic rates. Third, the algorithm requires only a small number of observations for the solution of the optimal policy, allowing inexpensive exploration.

5 Performance Evaluation

5.1 Experiment Settings

Consider the illustrative Gridworld problem [24], in a 5×5 grid where each cell is a state, there are 4 possible actions, $A = \{\leftarrow, \rightarrow, \uparrow, \downarrow\}$. Actions that take the agent out of the grid cause the agent to remain in the same state and receive a reward of -1; other actions have a reward of 0, except actions taken at special states X and Y. All actions taken at X and Y result in a reward of 10 and 5 and a subsequent move to X' and Y', respectively.

We run the variational RL algorithm and Q-learning [26] on the Gridworld problem. The tensor operations are implemented using Google's open source library $TensorNetwork$ [13]. The number of observed transitions, $|\Omega|$, is varied to test to what extent the model's performance is compromised by the lack of complete knowledge about the environment, and other hyper-parameters are fixed: $\gamma = 0.6$, $K = 3$, $\chi = 20$, $\alpha = 10^{-4}$, and $\lambda = 1$.

5.2 Results

We obtain the optimal policy π^* and minimum value for $H(\pi^*) = -\sum_{s_t,a_t} Q(\pi^*|s_t, a_t)$ for the Gridworld problem, and compare the two methods on how well they approximate both.

As shown on the left of Fig. 4, variational RL is able to find the optimal policy after exploring only 800 transitions, whereas Q-learning requires at least 1300 transitions to converge. Also, when the number of observation is around 100, the policy found by Q-learning falters while policy found by our algorithm retains above 60% of correct actions.

The right side of Fig. 4 shows how much the approximated $H(\pi)$ value deviates from the true value $H(\pi^*)$ during the optimization process. After 10000 epochs, variational RL is able to closely approach the true value with 2000 explored transitions. With 500 and 1000 explored transitions, our algorithm still outperforms Q-learning in minimizing the error. Note that for Q-learning, the number of explored transitions equals the number of epochs, so variational RL performs better minimization of $H(\pi)$ even with only 500 transitions, over Q-learning with 10000 observations.

6 Conclusions

In this paper, we have presented a novel quantum tensor network approach for variational reinforcement learning. We mathematical demonstrate the equivalence between the Bellman equation and the Hamiltonian equation, allowing us to transform the reward maximization problem into energy minimization of a quantum mechanical system. Then, we apply ALS tensor completion to estimate the reward tensor using MPS, and optimize the policy using a SGD-based algorithm on tensor networks. We also proposed an efficient exploration algorithm. Our experimental results show that our algorithm has notable advantage in terms of training stability and data efficiency over traditional methods.

References

[1] Lucian Busoniu, Tim de Bruin, Domagoj Tolić, Jens Kober, and Ivana Palunko. Reinforcement learning for control: performance, stability, and deep approximators. *Annual Reviews in Control*, 10 2018.

[2] Bijoy K. Dey, Marek R. Janicki, and Paul W. Ayers. Hamilton-jacobi equation for the least-action/least-time dynamical path based on fast marching method. *The Journal of Chemical Physics*, 121(14):6667–6679, 2004.

[3] Stuart E Dreyfus. Dynamic programming and the calculus of variations. *Journal of Mathematical Analysis and Applications*, 1(2):228 – 239, 1960.

[4] Eliot Kapit Wesley Jones Eric B. Jones, Peter Graf. K-spin hamiltonian for quantum-resolvable markov decision processes. *ArXiv*, abs/2003.11881, 2020.

[5] Thomas G. Fischer. Reinforcement learning in financial markets - a survey. FAU Discussion Papers in Economics 12/2018, Friedrich-Alexander University Erlangen-Nuremberg, Institute for Economics, 2018.

[6] Thomas Gabel and Martin Riedmiller. On a successful application of multi-agent reinforcement learning to operations research benchmarks. *IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pages 68–75, 2007.

[7] Edward Gillman, Dominic C. Rose, and Juan P. Garrahan. A tensor network approach to finite markov decision processes, 2020.

[8] Georey J. Gordon. Reinforcement learning with function approximation converges to a region. In *NIPS*, 2000.

[9] R. Islam, Peter Henderson, M. Gomrokchi, and Doina Precup. Reproducibility of benchmarked deep reinforcement learning tasks for continuous control. *ICML Workshop on Reproducibility in Machine Learning*, 2017.

[10] Jens Kober, J. Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32:1238 – 1274, 2013.

[11] Lieven De Lathauwer and Dimitri Nion. Decompositions of a higher-order tensor in block terms - part iii: Alternating least squares algorithms. *SIAM J. Matrix Anal. Appl.*, 30:1067–1083, 2008.

[12] Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. In *3rd International Conference on Learning Representations, ICLR 2015-Conference Track Proceedings*, 2015.

[13] Ashley Milsted, Martin Ganahl, Stefan Leichenauer, Jack Hidary, and Guifre Vidal. TensorNetwork on TensorFlow: A spin chain application using tree tensor networks, 2019.

[14] E. Nikishin, Pavel Izmailov, Ben Athiwaratkun, D. Podoprikhin, T. Garipov, Pavel Shvechikov, D. Vetrov, and A. Wilson. Improving stability in deep reinforcement learning with weight averaging. 2018.

[15] Dimitr Nion and Nicholas D. Sidiropoulos. Adaptive algorithms to track the parafac decomposition of a third-order tensor. *IEEE Transactions on Signal Processing*, 57:2299–2310, 2009.

[16] Alexander Novikov, Dmitrii Podoprikhin, Anton Osokin, and Dmitry P Vetrov. Tensorizing neural networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 442–450. Curran Associates, Inc., 2015.

[17] Ann Nowé, Peter Vrancx, and Y. D. Hauwere. Game theory and multi-agent reinforcement learning. In *Reinforcement Learning*, 2012.

[18] Tomoki Ohsawa, Anthony M Bloch, and Melvin Leok. Discrete hamilton–jacobi theory. *SIAM Journal on Control and Optimization*, 49(4):1829–1856, 2011.

[19] Roman Orus. A practical introduction to tensor networks: Matrix product states and projected entangled pair states. *Annals of Physics*, 349:117–158, 2014.

[20] I. V. Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–23, 2011.

[21] Stephan Rabanser, O. Shchur, and Stephan Günnemann. Introduction to tensor decompositions and their applications in machine learning. *ArXiv*, abs/1711.10781, 2017.

[22] A. Ramezanpour. Optimization by a quantum reinforcement algorithm. *Physical Review A*, 96(5), 2017.

[23] D. Silver, Julian Schrittwieser, K. Simonyan, Ioannis Antonoglou, Aja Huang, A. Guez, T. Hubert, L. Baker, Matthew Lai, A. Bolton, Yutian Chen, T. Lillicrap, F. Hui, L. Sifre, George van den Driessche, T. Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550:354–359, 2017.

[24] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[25] Wenqi Wang, Vaneet Aggarwal, and Shuchin Aeron. Tensor completion by alternating minimization under the tensor train (TT) model. *CoRR*, abs/1609.05587, 2016.

[26] Christopher John Cornish Hellaby Watkins. Learning from delayed rewards. 1989.

[27] Jiongmin Yong and Xun Yu Zhou. *Stochastic controls: Hamiltonian systems and HJB equations*, volume 43. Springer Science & Business Media, 1999.