Notes on: “Quantum Tensor Networks for Variational Reinforcement Learning”

1 <u>Introduction</u>

Reinforcement Learning (RL) is a novel field in machine learning. Some recent examples of its success are seen in DeepMind’s AlphaGo and AlphaZero, which beat the top players in both Go and Chess. This was achieved through the growth of computational resources and big data.
However, there are still many challenges for RL:

1. Traditional dynamic programming suffers from instability
2. Computation and storage complexities for Markov Decision Process/RL are combinatorially large
3. Reproducibility, reusability, and robustness is not clear, according to many researchers.

Using a variational optimization scheme, with the aid of tensor networks the paper will address these challenges.

2 <u>Background and Preliminaries</u>

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 $\{1, 2, \cdots, n\}$.

RL algorithms train the agent to interact with an environment, such that it chooses the sequence of actions that have the maximum expected rewards.

The equation for reward:

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

R(t+k+1) denotes the direct reward for 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

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

Tensors are algebraic objects that describe a multilinear relationship between sets of algebraic objects related to a vector space.

Tensor networks are useful for RL as they represent multilinear mapping using interconnected tensors, which allows for greater efficiency. This operation is called tensor contraction.

For tensors $\mathcal{C} = \mathcal{A} \times_n^m \mathcal{B}$, storing A and B instead of C is much more efficient for storage. Also, if two nodes are connected, that means the corresponding tensors are contracted, allowing for efficiency in that facet

3 <u>Problem Formulation for Variational Reinforcement Learning</u>

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

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

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

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

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

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 $s_q \xrightarrow{u_q} s_{q+1}$ when the environment is stochastic.

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

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

$$\sum_{s_t,a_t}^{S\times A} Q(\pi|s_t,a_t) = \mathcal{C}^{(0)} + \gamma \sum_{\mu_{t+1}}^{S\times A} \mathcal{C}^{(1)}_{\mu_{t+1}} \pi(\mu_{t+1}) + \gamma^2 \sum_{\mu_{t+1}}^{S\times A} \sum_{\mu_{t+2}}^{S\times A} \mathcal{C}^{(2)}_{\mu_{t+1},\mu_{t+2}} \pi(\mu_{t+1})\pi(\mu_{t+2}) + \cdots$$

$$= \mathcal{C}^{(0)} + \sum_{k=1}^{K} \sum_{\mu_{t+1},\ldots,\mu_{t+k}}^{S\times A} \mathcal{C}^{(k)}_{\mu_{t+1},\ldots,\mu_{t+k}} \pi(\mu_{t+1})\pi(\mu_{t+2})\cdots\pi(\mu_{t+k}), \tag{8}$$

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 Hamiltonian is useful for finding optimal solutions for systems with the notion of least action, or efficiency, which is a distinctive characteristic of the products of algorithms, especially Machine Learning algorithms.

The Hamiltonian equation finds the total energy of a system (the sum of potential and kinetic energy). 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. So, for finding the most optimal path, we would need to minimize any actions that would incur the most penalty (losing potential energy).

4 <u>Energy Minimization Algorithm Using Tensor Networks</u>

This section is about an algorithm that uses Variational Tensor Networks to search for the minimum energy of a Hamiltonian Equation, and also analyzes the cost of exploration as well as proposing a method for efficient exploration.

Tensor Products:
If x, y are vectors of length M and N, respectively, their tensor product x⊗y is defined as the M ×N-matrix defined by (x ⊗ y)ij = xiyj

Algorithm Design:

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

It is difficult to get the reward tensors $\{\mathcal{C}^{(k)}\}_{k=1}^{K}$ from an exponential environment, however we can obtain a set of observations for each k from (1 to K, as shown in the above Hamiltonian) to ultimately obtain an estimate of the full reward tensor.

Exploration is as follows: start at a random state-action pair (s_{t+1}, a_{t+1}) and explores a random trajectory of length k for k = 1, …, K. at each increment, we calculate the sum of all possible rewards, and then randomly choose the next state; repeat. Using the collected information, we update the reward tensor.

To improve efficiency, the agent explores trajectories of length L (where L≫k) and reuses sub-trajectories of length k. This means that one trajectory of length L can be used to update multiple entries in the reward tensor, reducing the overall cost of exploration.

Order of contraction is important for the efficiency of algorithm, and the contraction path needs be chosen carefully.

Algorithm for joint tensor completion and energy minimization is ALS and SGD

Variational RL has advantages over traditional methods in two ways: optimization process has greater stability and faster convergence because the objective function is evaluated over the global state-action space. Convergence rate is exponential, rather than linear or polynomial as in traditional methods, and algorithm only requires a small amount of observations for the solution.