Prospects of quantum computing approach to reinforcement learning

Matteo Conterno

April 29, 2022

Abstract

Reinforcement learning is one of three major techniques that allows a model to learn, especially this technique focus on how to create an optimal agent able to reach an objective by interacting with an environment. This thesis tries to analyze the possible potentials and advantages that would derive by using quantum circuits with neural networks; analyzing and explaining how it is possible to create hybrid algorithms that exploit the advantages of both the classical and quantum algorithm, expanding the applicability not only on simulators, but even on quantum devices such as Rigetti and IonQ's processors through the amazon AWS braket service. Lastly it is analyzed how the quantum noise influence an optimal model obtained through a simulator, by testing it on a quantum processor to understand if it is possible to use the hybrid algorithm on the actual devices.

Contents

1 Introduction to Deep Reinforcement Learning and Quantum Computing

This section of the thesis is created so that it can give an introduction to the fields of Artificial Intelligence(AI) specifically to Deep Reinforcement Learning(DRL) and the basis of Quantum Computing(QC) in order to understand afterward the fusion of these two different fields on Quantum Reinforcement Learning(QRL). If already expert on these subjects fell free to skip at the next section.

1.1 Approaches of learning

Currently to make an algorithm learn the following components are required:

- Data
- Model
- Approach of learning

The data is necessary in every model that uses some learning approach, the amount and quality can heavily influence on the model ability to correctly and efficiently reach his goal. The model is an algorithm that, given some data and a predefined objective, tries to complete his task using some approach of learning. In order to check that the model has correctly learned it will be later tested on unseen data and evaluated to understand if it is able to replicate the performances given a similar dataset.
The approach of learning specify how the model can learn to complete his task. At the moment there are three major ways:

- **Supervised learning** : the dataset given is already labeled, this means that we can define when the model is incorrect or correct.
- **Unspervised learning** : the dataset is not labeled, this means that we can't define easily when the model is correct or incorrect and the model must uncover some pattern.
- **Reinforcement learning** : an agent interact with the environment in order to become the most optimal agent to complete the task.

This thesis will focalize mainly on the reinforcement learning approach and especially on the Deep Reinforcement Learning (DRL) which uses Neural Networks (NN) to define the most optimal agent. In this thesis the focus is on what kind of advantage we can obtain by using a quantum algorithm such as Variational Quantum Algorithm (VQA), this can be used in other context and application. Futhermore it has been demostrated that VQA and NN share some similarities and properties. The main drawback is the fact that when a VQA is trained on a classical device, there is a major overhead of time due to simulation of the quantum circuit implemented and the number of qubits is limited.

1.2 Reinforcement Learning

As said earlier a reinforcement learning approach consist of creating the most optimal agent able to complete a predefined task by interacting with the environment and taking some action. Questions arises about: how do we define correctly what is the correct action? How do we model a dynamic environment? Futhermore how do we define when an action is good or bad? The answer is given by using a staristical model called Markov Decision Process (MDP) .

1.2.1 Markov Decision Process

In order to model a dynamic environment with a model whose action can influence the sistem it is necessary to use the Markov Decision Process (MDP) which is an extension of Markov chains; these are a stochastic model that is able to describe a sequence of possible events that satisfy the Markov propriety, that is each event depends only on the previous event.
It is necessary to note that an MDP is based on Markov Chains that not only model the states, but even the time and this can be defined as continuous or discretized, for this thesis we will focalize on the discrete case. MDP is an extension of Markov Chains because we have an agent which can influence the state of environment and outcome, so we need to have a framework to define his decision making. An MDP is defined as a 4-tuple containing the following elements:

- S : set of states
- A : set of actions
- $P_a(s, s') = Pr(s_{t+1} = s' | s_t = s)$: is the transition probability of going from state s to s' by taking an action a
- $R_a(s, s')$ is the immediate reward obtained by transitioning from state s to s' by action a

The difference with a Markov chains is the missing elements $P_a(s, s')$ and $R_a(s, s')$ which are necessary for the decision process, to see an example graph of and MDP see Figure 1.

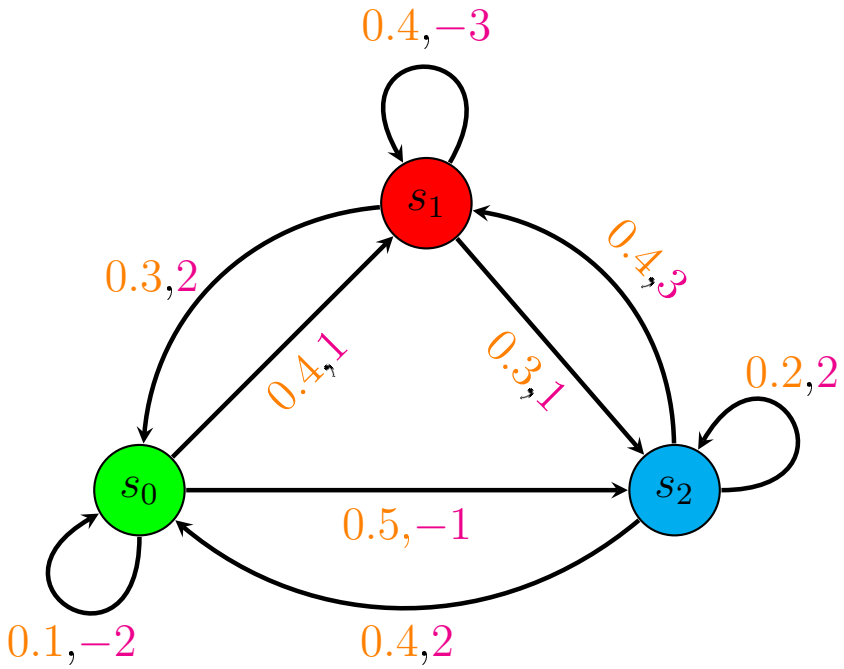

Figure 1: Markov decision process graph, transition probability is in orange and reward in magenta.

The interaction between agent and environment can be defined by time, using a discrete step which implies view it as distinct separate points in time uniquely definied and to which a single state value can be associated. The sequence of observation over times form a chain of states that is called **history**, this will be particularly important because it will be used later to define the transition probability and use it later to model the interaction with environment.
In order to include the reward element of an MDP accumulated from present and future a new quantity need to be defined called **return**:

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k} \tag{1}$$

The γ is a variable called discount factore with values limited inside the range of 0 and 1 with extremes included, i.e. $\gamma \in [0, 1]$. The scope of this variable is to limit the horizon of **return**, in case γ is equal to 0 only the immediate reward will be counted and when it is 1 infinite future steps will be considered.Usually the literature use $\gamma \in [0.9, 0.999]$ in order to gradually consider less relevant further time steps rewards thanks to the k-power of γ inside 1. The RL learning approach has the objective to maximize the **return** quantity, but this form is not useful for the agent due to the fact that it considers every possible chain that can be observed using the Markov reward process. This means that it can vary widely even for the same state, however it is possible to calculate the expectation of return from any state by averaging a large number of chains and obtain a new quantity called **value of state**:

$$V(s) = \mathbb{E}[G|S_t = s] = \mathbb{E}[\sum_{t=0}^{\infty} r_t \gamma^t] \tag{2}$$

These formulas considers the reward and state but they are not sufficient to model envrinoment and agent, so it is necessary to define a set of rules to control the agent behaviour with the objective of RL that is to maximize the return, for these reasons a new quantity must be defined **policy**.It is formally determined as probability distribution over actions for every possible state, i.e.:

$$\pi(a|s) = P[A_t = a|S_t = s] \tag{3}$$

The policy is defined as a probability to introduce randomness of the agent that will be useful during the training phase, furthermore if policy is fixed the transition and reward matrixes can be reduced using policy's probabilities reducing the action dimension.

1.2.2 Value iteration

As stated before the objective of RL is to maximize the return, problem is how to approximate the best optimal policy and values state in order to define the correct actions for given state. Fortunately the **Bellman optimality equation** is able to estimate approximately the best action to take on deterministic and statistical case.The equation for a general case is defined as:

$$V(s) = \max_{a \in A} \mathbb{E}_{s' \sim S}[(s, a) + \gamma V(s')] = \max_{a \in A} \sum_{s' \in S} p_{a, s \to s'}(r(s, a) + \gamma V(s')) \tag{4}$$

The interpretation of this formula is that the optimal value state is equal to the action which gives the maximum possible expected immediate reward, plus the discounted long-term reward of next state. Futhermore this definition is recursive because the value state is determined from values of immediate reachable states. This formula not only gives the best reward that can be obtained, but it gives the best policy to obtain that reward. In order to simplify this formula it is possible to define other quantities, such as **value of action**:

$$Q(s, a) = \mathbb{E}_{s' \sim S}[r(s, a) + \gamma V(s')] = \sum_{s' \in S} p_{a, s \to s'}(r(s, a) + \gamma V(s')) \tag{5}$$

This quantity allow to define a pair of state and action, it is particulary important because it defines a category of learning called **Q-learning** which will be the focus of this thesis. As you can see using this new definition 4 becomes:

$$V(s) = \max_{a \in A} Q(s, a)$$

Thanks to this, the 5 can be even defined recursively and this will be particulary useful later to use it along the deep learning approach:

$$Q(s, a) = r(s, a) + \gamma \max_{a \in A} Q(s', a') \tag{6}$$