Understand the Properties of Actions and Environment Components from the Experiences in Montezuma Revenge

Lisheng Wu

Research Proposal Draft

1 Montezuma Revenge

Montezuma Revenge is an Atari game with sparse, delayed rewards. The game requires the agent to navigate through different rooms while collecting rewards. We show the first room in Fig.1 where the agent needs to go a long way to collect the key as the first reward(+100) and use the key to open the door to the next room(+400). It's one of the most challenging Atari games where DQN[4] failed to obtain any reward. It's the main environment we will research on.

2 Topic Statement

In an environment like Montezuma Revenge, even without rewards, some environment properties or knowledge can still be learned from the interactions with the environments, through unsupervised learning or self-supervised learning. Those properties could be further used to boost the training process of RL algorithms. The properties can include but not be limited to, what one action changes to the state and how one environment component reacts to the action. An action may not lead to a change in environment, or one component may only react to a limited set of actions. What's more, two actions may cancel out due to their opposite effects.

With those properties, we could eliminate redundant actions. Furthermore, with deeper understandings about those environment components, we could find the minimal set of action series that makes different senses on one component. In Montezuma Revenge, the agent can not move left or right on the ladders and cannot move down on the ground. Also, without rewards or life loss, it is only meaningful to keep moving upwards or downwards on the ladders. These properties could boost exploration efficiency and even be reused in a new room if we can recognise the same components. In total, the topic aims at exploiting the action and environment properties to help efficient RL training.

Though in the real world, we learn about the world from the first-person point of view and the interactions could be more complicated than Montezuma Revenge, we still need to learn structures from the interactions with the real world even without the rewards. In autonomous driving, we could also infer the knowledge from demonstrated video streams by learning or even clustering how the car interacts with other cars, passengers and roads.

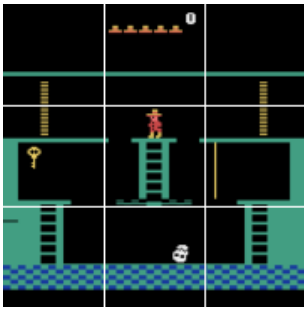

Figure 1: Montezuma Revenge Grids

Then those actions could provide a safe action space for the car to explore. It is promising to apply RL to the real world if we can understand those properties in the future.

3 Research Aims

It is not easy to segment each component from the image input semantically because we don't have data for semantic segmentation training. Instead, dividing the image input into grids could be a good way to segment the environment into different parts as the environment components. This method also has one advantage that each component has the same rectangular shape, and it is convenient for us to input them to a neural network.

The research aim is train four neural networks h, g, π_C, π_G to make use of the properties of actions and environment components to explore more efficiently. (a)h: attention network, learn about the components that we should attend to; (b)g: actions refinement network, provide signals for π_C to learn better interaction strategies; (c) π_C: the local component policy, mainly used to assist the exploration; (d) π_G: the global policy. For details refer to Section.4.

The main contributions of this work will include

(1). Use the properties of action to learn the attention mechanism.

(2). Use the properties of environment components to learn the most efficient ways to interact with each component even without rewards.

(3). We can reuse the learned interactions with each component potentially.

4 Methods

As stated above, we obtain the environment components by using grid lines. For example, reshape the image input into 180x180 and segment it into 3x3 grids as Fig.1.

Understand Action Properties: In our research, the action properties mainly refer to what an action changes when it interacts with each environment component. We choose to use the action properties to train a neural network h for attention. We'll adopt the inverse dynamic model used in [6] to predict the actions with all the environment components as the inputs. If the prediction result for a component is not confident about the actions taken or not taken, then the component doesn't account for the changes. Otherwise, they are the components to which we should attend. Those components form a set $\mathcal{A}$ and we can considered those as the components with which the agent is interacting. It is similar to the way how people noticed what objects they are controlling when they play games.

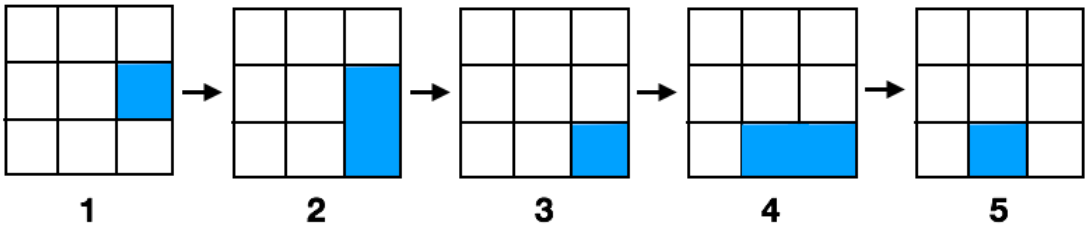

Figure 2: Attention Switch

Understand Components Properties: One reason why DQN[4] failed at Montezuma Revenge is that most of its explorations are limited to a small region. Thus, there are too many redundant or useless actions. For example, the agent cannot move left or right when it is on the ladders.

One way to think about the problems could be maximising the entropy of next state distribution conditioned on the current state. $p(s'|s,a)$ is the transition probability, $p(a|s,s')$ is the action distribution predicted by inverse dynamic model. Assume the inverse dynamic model here is independent of the policy and considers all possible actions with equal probabilities. Then, when we maximise the entropy of $p_\pi(s'|s)$, we could find a much more balanced action distribution for exploration. However, it is not easy to generalise to state transition after multiple actions and compute the entropy for continuous state space.

$$p_\pi(s'|s) = \frac{p(s'|s,a)p_\pi(a|s)}{p(a|s,s')} \tag{1}$$

Instead, we encourage the agent to interact with a component with fewer actions. That means if there exists a way to discard some redundant actions, the agent will get a reward for such an experience. By this way, we could eliminate those redundant actions indirectly. We identify the start and end of the interactions with a component C_i as the attention's change predicted by the attention network. For each component C_i, the start of the interaction begins when C_i is involved in the interactions with the agent($C_i \in \mathcal{A}$) and ends when C_i exits the interactions. We note the component state of C_i as s_{C_i}, the start interaction state of component C_i is noted as $s^S_{C_i}$, and the end state is noted for $s^E_{C_i}$. In Fig.2, for the right bottom component C_{rb}, the first component state at step 2 is $s^S_{C_{rb}}$ and the last state at step 4 is the $s^E_{C_{rb}}$. The component state here can be choosed as a stack of frames like the setting in DQN[4].

$s^S_{C_i}$ and $s^E_{C_i}$ are taken as inputs into the neural network g for actions refinement which predicts the number of steps between the two states as a continuous variable $n_{s^S_{C_i},s^E_{C_i}}$. The network g is trained by supervised learning on the real number of steps $N_{s^S_{C_i},s^E_{C_i}}$ between the start state and the end state.

$$n_{s^S_{C_i},s^E_{C_i}} = g(s^S_{C_i}, s^E_{C_i}) \tag{2}$$

We can consider the network g as a value function. If the number of interaction steps with a component $N_{s^S_{C_i},s^E_{C_i}}$ is lower than the value predicted then $n_{s^S_{C_i},s^E_{C_i}}$ the agent would be rewarded for the more efficient interactions. The reward can be calculated as:

$$r_g(C_i) = n_{s^S_{C_i},s^E_{C_i}} - N_{s^S_{C_i},s^E_{C_i}} \tag{3}$$

The policy network π_C which takes component states as inputs and we train π_C with the rewards $r_g(C_i)$ only from g. The reason is that if we also train π_C with environment rewards, it would make it more difficult to generalise to a new room. However, with only π_C, we lose the global information, and we can't control $s^S_{C_i}$, $s^E_{C_i}$. Thus, we cannot use π_C as a policy, and we

need to introduce a global policy π_G, which takes the whole state as the input and we train π_G with the environment rewards. In contrast, π_C is used to combine with $\epsilon - greedy$ algorithm for more efficient exploration. When the agent does the exploration during the interactions with elements, the exploration policy π_e could be:

$$\pi_e(s) = (1-\epsilon)\pi_G(s) + \epsilon \ (\frac{(1-\alpha)\sum_{C_i \in \mathcal{A}} \pi_C(s_{C_i})}{||\mathcal{A}||} + \alpha \ Uniform(s)) \tag{4}$$

$\alpha \in (0, 1)$ is used to balance the learned component policies and uniform policy and we expect to give the component policies higher weights(i.e. $\alpha > 0.5$).

5 Literature Review

The papers working on sparse reward problem like Montezuma Revenge could be mainly classified into two classes: **hierarchical reinforcement learning(HRL)** which has typically high-level actions or sub-goals, and **curiosity-driven** method which uses intrinsic rewards measured by the novelties of states.

Hierarchical DQN[3] introduced predefined sub-goals to assist the agents to explore, which relies on prior knowledge from humans heavily. Though FeUdal Networks[7] could learn a sub-goal as an embedding and potentially generalise to different rooms, the sub-goal is hard to be understood, and the agent still needs to explore from scratch in a new room. In our method, if the agent interacts with one component for a long time, our method could find the most efficient ways to interact with it. What is more, if the agent meets the same components in a new room, our method could reuse the prior learned strategies to interact with them. That means we do not need to explore from scratch in a new room.

In contrast to HRL, curiosity-driven methods are more robust, and the agent is encouraged to explore the unfamiliar states. [1, 5] utilised pseudo rewards as intrinsic rewards. The new work Go-Explore[2] by Uber can solve the entire game by storing those novel states and choose states from them as stepping-stones. Another work[6] on Super Mario can learn to pass the levels without rewards from the environments. They also face similar problems to HRL's while combining them with our method can potentially address the problems. Besides, being motivated by novelties can introduce redundancy to the actions, like lingering around some particular states. Our method could find the most efficient ways to interact with each component, which could also be considered as high-level actions, to reduce the redundancy during exploration.

6 About me

I am interested in searching for additional signals or structures besides rewards to guide the training of reinforcement learning. For example, my master thesis[8] is about associate a Pong environment and its transformed environment by sharing the same dynamics between two environments. Though two environments can have entirely different state space, the actions have the same underlying physical meanings between them. By observing how the environments change with the actions, we could unify the representations of two worlds with the same dynamics applied to the representations. The research does not solve significant challenges, but it shows that investigating the environments from the dynamics of the interactions between agent and environment is meaningful.

References

[1] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 1471–1479, 2016.

[2] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Montezuma's revenge solved by go-explore, a new algorithm for hard-exploration problems (sets records on pitfall, too).

[3] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in neural information processing systems*, pages 3675–3683, 2016.

[4] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[5] Georg Ostrovski, Marc G Bellemare, Aaron van den Oord, and Rémi Munos. Count-based exploration with neural density models. *arXiv preprint arXiv:1703.01310*, 2017.

[6] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning (ICML)*, volume 2017, 2017.

[7] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1703.01161*, 2017.

[8] Lisheng Wu, Minne Li, and Jun Wang. Learning shared dynamics with meta-world models. *arXiv preprint arXiv:1811.01741*, 2018.