CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Improving sample-efficiency of model-free reinforcement learning algorithms by learning latent space representations

a

Master's thesis in Computer science and engineering

Marko Guberina, Betelhem Dejene Desta

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2022

Master's thesis 2022

Improving sample-efficiency of model-free reinforcement learning algorithms by learning latent space representations

a

Marko Guberina, Betelhem Dejene Desta

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2022

Improving sample-efficiency of model-free reinforcement learning algorithms by learning latent space representations
a
Marko Guberina, Betelhem Dejene Desta

Supervisor: Name, Department
Advisor: Name, Company or Institute (if applicable)
Examiner: Name, Department

Master's Thesis 2022
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Description of the picture on the cover page (if applicable)

Typeset in LaTeX
Gothenburg, Sweden 2022

Improving sample-efficiency of model-free reinforcement learning algorithms by learning latent space representations
a
Marko Guberina
Betelhem Dejene Desta
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

a

Keywords: Computer, science, computer science, engineering, project, thesis.

Acknowledgements

We give thanks to the supervisor and everyone else at Chalmers who made out work possible.

Marko Guberina, Betelhem Dejene Desta, Gothenburg, May 2022

Contents

List of Figures

List of Tables

1
Introduction

In computer science, reinforcement learning is the formalization of trial-and-error learning. While this is not the only legitimate interpretation of the concept, it is the most straightforward one: “trial” implies existance of an agent which observes its environment and interacts with it though its own actions. “Error” implies that the agent has a goal it tries to achieve and that it does not know how to achieve it (in the most effective manner). What it can do is take different actions and appraise them according to how closely they lead the agent toward its goal, thereby observing the quality of those actions. By repeatedly exploring the effects of various sequences of actions, the agent can find, i.e. learn, the sequence of actions which lead to its goal. Here it is important to discuss what a goal is. To formalize the process outlined above, one needs to described in purely mathematical terms, and that includes the goal as well. To achieve that, the notion of a reward function is used: it maps every state of the environment to a number which denotes its value called the reward. The state of the environment to which the highest reward is ascribed is then the goal. A more general description of the goal of reinforcement learning is to maximize the sum of rewards over time. Formalization of the entire process will be carried out later in the text, while here only the most important concepts will be outlined. One of these is the trade-off between “exploration” and “exploitation.” To learn just by trial and error implies learning from experience. This means that the agent can not know how a certain strategy fares unless it collects experiences which come by following said strategy. Thus in order to find a good strategy, usually refered to as a “policy”, the agent needs to produce various different strategies and observe their results until it find a promising one. The process of finding different strategies and experimenting with new random behavior is called exploration. Likewise, the process of repeating a good strategy is called exploitation. Due to the curse of dimensionality, [1] in complex multi-dimensional domains it is impossible to test but a miniscule proportion of all possible strategies. Because of this, the problem of effective exploration and the trade-off between it and exploitation is a fundamental problem in reinforcement learning.
Due to its generality, reinforcement learning is studied in many different disciplines: control theory, game theory, information theory, simulation-based optimization, multi-agent systems etc.. Of these, control theory is of particular importance because it often enables clear analysis of various reinforcement learning algorithms. This foremost concerns the usage of dynamic programming which provides a basis for a large class of reinforcement learning algorithms. Reinforcement learning is also

[1]The curse of dimensionality refers to the exponential rise of possible configurations of the problem with the number of dimensions.

considered to be one of the pillars of modern data-driven machine learning. In the context of machine learning, reinforcement learning can be view as a combination of supervised and unsupervised learning: the “trial” portion of the trial-and-error learning can be interpreted as unsupervised or as self-supervised learning because in it the agent collects its own dataset without any explicit labels to guide its way. This process is refered to as “exploration”. The dataset created by exploration is labelled by the reward function. Thus the agent can learning from “past experience” in a supervised manner. This text will introduce concepts from both control theory and machine learning which are necessary to formalize the reinforcement learning objective and to develop algorithms to achieve it. It will not concern itself with other disciplines.

Interest in reinforcement learning has grown tremendously over the past decade. It has been fueled by successes of deep machine learning in fields such as computer vision. The subsequent utilization of neural networks in reinforcement learning, dubbed deep reinforcement learning, led to impressive results such as achieving better-than-human perfomance on Atari games, in the game of go and in many others. These recent success were kick-started by Deep Q-Networks (DQN) algorithm which, by utilizing convolutional neural network, crucially enabled the agents to successfully learn from raw pixels. Learning from pixels is incredibly important for most practical applications, such as those in robotics where it is usually impossible to get full access to the state of the environment. The state then needs to be inferred from obsevations such as those from cameras. [2] Due to these incredible results in simulated environments, reinforcement learning holds the promise of solving many incredibly important engineering problems, for example robotic manipulation and grasping. Having that said, there exists a large gap between simple simulated environments and the real world, and many improvements to the current state-of-the-art algorithms are required to bridge that gap. To explain the approach investigated in this thesis, a bit more context is needed.

An important classification of reinforcement learning algorithm is the one between model-based and model-free algorithms. As the name suggests, model-free algorithms do not form an explicit model of the environment. Instead, they function as black-box optimization algorithms, simply finding actions which maximize reward without other concerns such as predicting the states resulting from those actions. In other words, they only predict the reward of actions in given states. Model-based algorithms on the other hand learn an explicit model of the environment and use it to plan their actions. They thus learn the dynamics of the environment and use that knowledge to choose actions which lead the agent to states with high reward. Both classes have their benefits and their drawbacks. Since model-free algorithms do not require any knowledge of environment dynamics to operate, they are more widely applicable and usually achieve better performance. But the fact that they can not leverage environment dynamics to create plans implies a harder learning problem:

[2]Here the state refers to the underlying physical parameters of the environment: the positions and velocities of objects, the friction coefficients and so on. Observations from sensors such as cameras do not explicitly provide such information. However, since humans and animals are able to utilize such observations to achieve their goals, we know that they implicitely hold enough information about the true state of the world for successful goal completetion.

they need to implicitely learn those dynamics while only being provided the reward signal. This makes them much less sample-efficient. By contrast, model-based algorithms are of course more sample-efficient. Furthermore, the plan generated from the learned model can be utilized to interpret the agent's actions which in turn leads to many further benefits such as the ability to guarantee outcomes in safety-critical operations. Unfortunatelly, the twin learning objective of learning the best action-choosing policy to maximize the reward over time, and the learning of the model results in fundamental training instabilities which usually result in worse final performance. All this will be further discussed in a later chapter.

2
Background

Markove Decision Processes(MDPs):

Markov decision processes model decision making in discrete, stochastic, sequential environments[9].The formal definition of Markov decision process can be summarised in the tuple $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{P}(\mathcal{S}_{\prime}), \mathcal{T}, r\}$[8]. The goal of the model is that an agent inhabits a stochastic environment in a response to action choice made by the agent[9].

The environment consists of the the Transition function, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to (\mathcal{P}\mathcal{S})$.
and the reward function $(r(s_t, a_t))$, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$.In MDPs the transition probability and reward only depends on the current state and the action chosen by the agent but not the past state and action.
The agent acts in environment according to a policy $\pi : \mathcal{S} \to (\mathcal{P}\mathcal{S})$.policy learned can be off policy where the behaviour policy is different from the policy used for action selection one common example is Q-learning or on-policy methods which attempts to evaluate or improve the policy that is used to make decision.
The former state-action value function (Q function for short)can be computes recursively with dynamic programming:

$$Q^{\pi}(s, a) = E_s'[r + \gamma E_a' \ \pi(s')[Q^{\pi}(s', a')]|s, a, \pi]$$

(2.1)

optimal q value under deterministic policy

$$a = argmax_a' \epsilon A Q^*(s, a') \tag{2.2}$$

$$Q^*(s, a) = \max_{\pi} Q^{\pi}(s, a)$$

(2.3)

if the optimal function satisfies the bellman equation:

$$Q^*(s, a) = E_s'[r + \gamma max(a')Q^*(s', a')|s, a, \pi] \tag{2.4}$$

The advantage function describes "how good" the action a is, compared to the expected return when following direct policy pi.

$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s) \tag{2.5}$$

The value function V measures the how good it is to be in a particular state s. The Q function, however, measures the the value of choosing a particular action when in this state[6].

Deep Reinforcement Learning and DQN

(paraphrase)After deep learning has shown remarkable results in learning from raw pixel data in computer vision it's application has been adopted to solve many classical learning problems.The goal of Deep Reinforcement Learning is to connect reinforcement learning algorithm to deep neural network which operates directly on RGB images and efficiently process training data by using stochastic gradient updates[2].

The use of label data and the assumption the distribution of data to be identical and independent through out training process in deep learning makes it complex to use it directly into reinforcement learning algorithms which must be able to learn from sparse ,noisy and delayed reward and highly correlated data.To address this issues and successfully apply deep learning to reinforcement learning [2] used experience replay mechanism[15] which randomly samples previous transitions, and thereby smooths the training distribution over many past behaviors.Convolutional neural network was used to learn successful control policies from raw video data in complex reinforcement learning environment.The network is trained with a variant of Q-learning algorithm,with stochastic gradient descent to update the weights.

This architecture was based on Tesauro's TD-Gammon architecture [17]which updates the parameters of the network that estimates the value function directly from on-policy samples of experience. Similar to this approach the online network in DQN utilize a technique known as experience replay [18] where the agent's experiences at each time-step, $\rceil_t = (s_t, a_t, r_t, s_t + 1)$ is stored in a data set $\mathcal{D} = e1, ..., e_N$ pooled over many episodes into a replay memory.By drawing random samples from this pool of stored experiences the Q-learning is updated.After performing experience replay, the agent selects and executes an action according to an -greedy policy.The use of experience replay and target networks enables relatively stable learning of Q values, and led to super human performance on several Atari games.

The advantage of using Deep Q-learning over Q-learning includes allowing to have greater sample efficiency,reduced variance by randomising the sample bias and avoiding being stuck in local minimum.Drawback DQNs are it only handle discrete,low-dimensional action space.

2.0.1 Extension of DQN

Several Studies were performed to increase the performance of DQNs(Paraphrase).

2.0.1.1 Double Deep Q-networks: DDQN

DQN suffer from overestimation bias due to due to the maximization step in optimisation function in Q-learning.Q-learning can overestimate actions that have been tried often and the estimations can be higher than any realistic optimistic estimate .Double Q-learning [19], addresses this overestimation by decoupling, in the maximization performed for the bootstrap target, the selection of the action from its evaluation.Double Q-learning stores two Q-functions,The average of the two Q values for each action and then performed ϵ-greedy exploration with the resulting average Q values.It was successfully combined with DQN to reduce overestimations. **TODOS: DDQN target equation**

2.0.1.2 Prioritized replay

The main use of replay buffer is to sample transitions with maximum probability.Both DQN and DDQN samples experiences uniformly.Prioritized replay [20] samples transitions using the maximum priority ,providing a bias towards recent transitions and stochastic transitions even when there is little left to learn about them.

2.0.1.3 Dueling Network

Dueling Network was designed for value based learning,this architecture separates the representation of sate-value and state-dependent action advantages without supervision[6].its consists of two streams that represents the value and advantage functions,while sharing a common convolutional feature learning module.This network has a single Q-learning network with two streams that replace DQN architecture[3].

$$Q(s, a; \theta, \alpha, \beta) = V(s, \theta, \beta) + A(s, a; \theta, \alpha) \tag{2.6}$$

if needed TODO ADD EQUATION:factorization of action values

2.0.1.4 Multi-step learning

Previously stated extension of DQN have indicated that the use deep learning has enhanced the learning capability of Q-learning.The performance of Q-learning is still limited by greedy action selection after accumulating a single reward.An alternative approach was multi-step targets:

$$y_{j,t} = \sum_{t'=t}^{t+N-1} \gamma^{t-t'} r_{j,t'} + \gamma^N \max_{a_{j,t+N}} Q_{\phi'}(s_{j,t+N}, a_{j,t+N}) \tag{2.7}$$

A multi-step variant of DQN is then defined by minimizing the alternative loss[16],

$$R_t^{(n)} + \gamma_t^{(n)} \max_{a'} q_{\theta}^{-}(S_{t+n}, a') - q_\theta(S_t, A_t) \tag{2.8}$$

2.0.1.5 Noisy Nets

The one limitation of ϵ-greedy policy is many actions must be executed to collect the first reward.Noisy Nets proposed a noisy linear layer that combines a deterministic and noisy stream.Depending on the learning rate the network ignores to learn the noisy stream.

2.0.1.6 Integrated Agent:Rainbow

In the Rainbow architecture [16] tries combine the above six method stated above all together.Distributional loss (3) was replaced with a multi-step variant.The target distribution was constructed by contracting the value distribution in St+n according to the cumulative discount, and shifting it by the truncated n-step discounted return. multi-step distributional loss with double Q-learning by using the greedy action in St+n selected according to the online network as the bootstrap action at+n, and evaluating such action using the target network.
TODO ADD EQUATION:target distribution

3
Methods

In this section, we will explain the architecture of our auto-encoder and reinforcement learning algorithm. This includes a description of the environment and preprocessing in Section 3.0.1., the collection of training data for the auto-encoder and model architecture in section 3.0.2. and the training of the RL agent in Section 3.0.3.

3.0.1 Enviroment and Preprocessing

[!ADD MORE HERE AFTER NAP]
We perform a comprehensive evaluation of our proposed method on the Arcade Learning Environment (Bellemare et al., 2013), which is composed of 57 Atari games. The challenge is to deploy a single algorithm and architecture, with a fixed set of hyper-parameters, to learn to play all the games given embedded latent space representation of the environment from auto encoder and game rewards. This environment is very demanding because it is both comprised of a large number of highly diverse games and the observations are high-dimensional.

Working with raw Atari frames, which are 210 x 160 pixel pictures with a 128 color palette, is computationally expensive, therefore we do a basic preprocessing step to reduce the input dimensionality. The raw frames are preprocessed by down sampling to a 110 x 84 picture and transforming their RGB representation to grayscale.Cropping an 84 x 84 rectangle of the image that nearly captures the playing area yields the final input representation to encoder part of the auto encoder.

3.0.2 Deep Auto-encoder and Model Architecture

[!ADD MORE HERE AFTER NAP]
The first step in the process is to collect data to train the auto-encoder. we run a data collection module to generate the 100000 frame for each stated under the result section.The raw images are transformed to tensors and then trained a variational autoencoder with the objective of re-constructing the original image fed to the network. The auto-encoder was trained for maximum 100 epochs.When reconstructing an image with a network bottleneck, the encoder is forced to compress the original image to a smaller dimensional vector in the latent space.
[By compressing the raw pixels environment to smaller dimensional vector in the latent space we aim to improve the training time it takes for the integrated Rl agent developed by [16] and the shift in the latent space representation and its impact

on the RL agent learning performance.We show this in more detail in the following sections.]

3.0.3 Training the RL Agent

[!ADD MORE HERE AFTER NAP]
In this paper we integrate auto encoder with integrated agent called Rainbow[12],the selection of this architecture was based it's ability to out perform all the previous architecture.our main focus was to experiment with the latent space representation of the environment.To address this we set up two experiment architectures one two step training and parallel training.

Two step Training: First,we train the auto encoder with the prepossessed data and the weights of the encoder are saved in file for training the agent.In this set up the auto encoder is not updated such that we have a static representation of the environment.
second,we train the integrated agent with the results from the auto encoder.

Parallel Training: In this set up the agent is trained using the dynamic state representation from the encoder as we train both the auto encoder and the integrated agent in parallel.This introduces stochasticity of the environment to the training and in real life control system we believe that this will set a new paradigm on the use RL.

4

Results

state the Experiments

4.0.1 Two Step Training

4.0.2 Parallel Training

5 Conclusion

You may consider to instead divide this chapter into discussion of the results and a summary.

5.1 Discussion

5.2 Conclusion

Bibliography

[1] [1] Sutton, R. S., Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

[2] Mnih, Volodymyr and Kavukcuoglu, Koray and Silver, David and Graves, Alex and Antonoglou, Ioannis and Wierstra, Daan and Riedmiller, Martin.(2013). Playing Atari with Deep Reinforcement Learning.

[3] Mnih, V., Kavukcuoglu, K., Silver, D. et al.(2015). Human-level control through deep reinforcement learning. Nature 518, 529–533.

[4] Hado van Hasselt and Arthur Guez and David Silver.(2015).Deep Reinforcement Learning with Double Q-learning.

[5] van Hasselt.(2010).Double Q-learning. Advances in Neural Information Processing Systems, 23:2613–2621.

[6] Wang, Ziyu and Schaul, Tom and Hessel, Matteo and van Hasselt, Hado and Lanctot, Marc and de Freitas, Nando.2015.Dueling Network Architectures for Deep Reinforcement Learning.

[7] Moerland, Thomas M. and Broekens, Joost and Plaat, Aske and Jonker, Catholijn M.Model-based Reinforcement Learning: A Survey.(2020).

[8] Puterman, M. L. (2014). Markov Decision Processes.: Discrete Stochastic Dynamic Pro- gramming. John Wiley Sons.

[9] M.L. Littman, in International Encyclopedia of the Social Behavioral Sciences, 2001.

[10]Kidambi et al., 2020
[11] Yu et al., 2021
[12]Ignasi Clavera and Jonas Rothfuss and John Schulman and Yasuhiro Fujita and Tamim Asfour and Pieter Abbeel, Model-Based Reinforcement Learning via Meta-Policy Optimization,2018
[13]Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel. RLˆ2: Fast Reinforcement Learning via Slow Reinforcement Learning. 11 2016.

[14]. Sung, L. Zhang, T. Xiang, T. M. Hospedales, and Y. Yang. Learning to learn: Meta-critic networks for sample efficient learning. CoRR, abs/1706.09529, 2017.
[15]
[16]Hessel, Matteo and Modayil, Joseph and van Hasselt, Hado and Schaul, Tom and Ostrovski, Georg and Dabney, Will and Horgan, Dan and Piot, Bilal and Azar, Mohammad and Silver, David.2017.Rainbow: Combining Improvements in Deep Reinforcement Learning.
[17]Gerald Tesauro. Temporal difference learning and td-gammon. Communications of the ACM, 38(3):58–68, 1995.
[18]Long-Ji Lin. Reinforcement learning for robots using neural networks. Technical report, DTIC Document, 1993.
[19]Double DQN
[20]Prioritized replay [21]On the use of Deep Auto-encoders for Efficient Embedded Reinforcement Learning

A

Appendix 1

(UNFINISED)
//UPDATE THE FORMAT LATER

1. Agent: It is an assumed entity which performs actions in an environment to gain some reward.
2. Environment (e): A scenario that an agent has to face.anything the agent cannot change arbitrarily is considered to be part of the environment.
3. Reward (R): An immediate return given to an agent when he or she performs specific action or task.
4. State (s): State refers to the current situation returned by the environment.
5. Policy (): It is a strategy which applies by the agent to decide the next action based on the current state.
6. Value (V): It is expected long-term return with discount, as compared to the short-term reward.
7. Value Function: It specifies the value of a state that is the total amount of reward. It is an agent which should be expected beginning from that state.
8. Model of the environment: This mimics the behavior of the environment. It helps you to make inferences to be made and also determine how the environment will behave.
9. Model based methods: It is a method for solving reinforcement learning problems which use model-based methods.
10. Q value or action value (Q): Q value is quite similar to value. The only difference between the two is that it takes an additional parameter as a current action.