Unifying value learning and model learning

Carles Gelada

January 13, 2017

Abstract

Learning a model of the world, that focuses on the relevant information, is a fundamental challenge of artificial intelligence. In this work I present a method of integrating model based and model free reinforcement learning in a simple, end-to-end manner. Although the proposed method is general and can be applied to both, actor-critic and value based methods, I test a simple extension that integrates with DQN, providing it with the ability predict future rewards and values. Once trained, planning algorithms can be used together with the learned model to train on simulated experience and to select better actions. The presented architecture has a minimal computational overhead compared to DQN.

1 Introduction

The ability to "imagine" what would happen if an action is taken seems a fundamental property of intelligence. Considering different situations, and reasoning about them is a useful cognitive tool. Giving neural networks the ability to perform such tasks is therefore a promising research path. Although imagination is a much broader concept, here I am only going to discuss a constrained form of it, model based planning.

Usually, model based reinforcement learning (MBRL) is approached by breaking it down into two tasks, learning a model of the world and training a normal RL agent. In cases where the state has high dimensionality, learning a transition function can be very hard and can have high computational complexity. Furthermore, a lot of information in the state is irrelevant for the task at hand. We would like to be able to learn a transition function of an inner representation of the agent that only focuses on the information necessary to predict the current policy and or value. This inner representation should also encode everything relevant to predicting future values.

In highly structured environments, where the transition function follows highly regular patterns, learning such transition function could be much easier than directly learning an optimal policy. Planning could then be used to generate simulated experience and update the policy and values or to select better actions.

2 Previous Work

DYNA is the classic example of MBRS. Works in the tabular setting.

The possibilities of merging video prediction with RL were explored by [4]. Similarly, [1] tried video prediction and also introduced the idea of value prediction.

VINs [6] show how a RL policy can learn to use recurrent convnets to do a kind of planning.

This work shares many similarities with "The predictron" [5].

3 The model

Consider an agent interacting with and MDP, consisting of states $s \in S$, actions $a \in A$, reward function $R(s, a)$ and transition probability function $P(s'|s, a)$. At each time step t with state s_t, the agent samples an action a_t according to the policy $\pi(s_t)$ and then observes the a reward $r_t \sim R(s_t, a_t)$ and the next state sampled from $P(s_{t+1}|s_t, a_t)$.

The agent's goal is to maximize the value $V^\pi(s) = \mathbb{E}[\sum_{i \geq 0} \gamma^i R(s_i, a_i)]$, where the expectation is taken over the trajectories generated by the policy π and state transition P and where $\gamma \in (0, 1)$ is a discount factor that trades-off the importance of long and short term rewards.

RL value methods try to solve this problem by estimating $Q_\theta(s)$ in some form or another. On-policy methods estimate the value of each state after following policy pi. Off-policy methods, instead, try to directly estimate the optimal value.

The main contribution of this work is the idea that MBRL can also be approached by trying to find a transition function f_t that, instead of predicting future states, predicts future hidden states with a similar value than the real future state.

A parametrized function $\psi(s)$ is introduced to find a hidden representation useful to plan and estimate the Q value. The new $Q(\psi(s))$ can be trained with any already existing algorithm and the functions $\hat{R}(\psi(s), a)$, $\hat{T}(\psi(s), a)$ are trained to predict the reward and the probability of the next state being terminal. Finally, $\hat{P}(\psi(s), a)$ is trained to output a next hidden state that has a value function equal to the value function of the real next sate.

There are several reasons learning to plan could be easier than learning the value function. Even in MDP with highly regular transition dynamics (e.g. following the laws of physics), the value function might require a lot of samples to be estimated. Learning a model of the (right representation of the) MDP might simply be easier and generalize more quickly than learning a value function.

Also, while the targets for the value function are non static, the ones for the prediction of reward and termination probabilities are. It is likely the learning of $\hat{R}(\psi(s), a)$ and $\hat{T}(\psi(s), a)$ will converge much quicker and need a lot less samples. Predicting deeper can be interpreted as reducing dependence on non static targets, likely making the value estimates much more accurate. This more accurate value estimates could be used as better targets for training or to select better actions.

It might be that the forcing the inner representations to be able to predict the reward, termination probabilities and next values acts as a regularizer, driving the representations to contain information that will likely be useful to predict the current value. This is some resemblance to the work on auxiliary tasks by [3] and [2] that has recently shown significantly useful in RL.

3.1 Planning DQN

Since applying a transition function trained to predict over one step would likely lead to unstable behaviours, training it to predict over multiple time steps is important. If we are looking to extend with the ability to plan value methods, n-step Q learning is the natural way to do it. In n-step Q learning the target to $Q(s_t, a_t)$ is constructed as $\sum_{i=0}^{n-1} \gamma^i r_{t+i} + \gamma^n V(s_{t+n})$.

Let $\psi_k(s_t, a_{t:k})$ be the predicted hidden state resultant after applying $\hat{P}$ recursively k times. And let $Q_k(s_t, a_{t:t+k}) = Q(\psi_k(s_t, a_{t:t+k-1}), a_{t+k})$ be the k-step value predictions, $\hat{R}_k(s_t, a_{t:t+k}) = \hat{R}(\psi_k(s_t, a_{t:t+k-1}), a_{t+k})$ and $\hat{T}_k(s_t, a_{t:t+k}) = \hat{T}(\psi_k(s_t, a_{t:t+k-1}), a_{t+k})$. Then, each $V_k(s_t, a_{t:t+k})$ can be trained with the target $\sum_{i=k}^{n-1} \gamma^i r_{t+i} + \gamma^n V(s_{t+n})$, each $\hat{R}_k(s_t, a_{t:t+k})$ with the target r_{t+k} and each $\hat{T}_k(s_t, a_{t:t+k})$ with the target 1 if s_{t+k} was a terminal state and 0 otherwise.

4 Experiments

I evaluate PDQN on the atari domain...

5 Discussion

Here are some of the new challenges and opportunities arising from the integration of planning with deep RL.

5.1 Planning algorithms

X explore different planning algorithms X This is equivalent to the adaptive computation in the predictrion.

5.2 Stochastic transition functions

X learn stochastic transition functions. They might learn more natural representations. Finding an abstraction of a stochastic world where the transition function is deterministic might be very hard or impossible. X

5.3 Continuous action spaces

Although learning to predict values in continuous spaces does not require any modification to the algorithm, it is unclear how planning would be done. Maybe similar methods to the ones used in continuous RL could be used to learn search policies.

6 Conclusions

I have presented a simple, yet powerful architecture capable of learning the transition function. It can be readily integrated with any planning algorithm. The reference implementation is released at ... with the aim of facilitating further development in this area.

References

[1] Justin Fu and Irving Hsu. Model-Based Reinforcement Learning for Playing Atari Games. Technical report, 2016.

[2] X. Li, L. Li, J. Gao, X. He, J. Chen, L. Deng, and J. He. Reinforcement Learning with Unsupervised Auxiliary Tasks. pages 1–11, 2016.

[3] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, and K. Kavukcuoglu. Learning to Navigate in Complex Environments. (2), 2017.

[4] J. Oh, X. Guo, H. Lee, R. Lewis, and S. Singh. Action-Conditional Video Prediction using Deep Networks in Atari Games. *Nips*, page 9, 2015.

[5] D. Silver, H. van Hasselt, M. Hessel, T. Schaul, A. Guez, T. Harley, G. Dulac-Arnold, D. Reichert, N. Rabinowitz, A. Barreto, and T. Degris DeepMind. THE PREDICTRON: END-TO-END LEARNING AND PLANNING.

[6] A. Tamar, S. Levine, and P. Abbeel. Value Iteration Networks. *arXiv*, (Nips):1–14, 2016.