Latent Predictive Agents

Dereck Piché

August 10 2024

1 Introduction

Given $Q : s, a \to r$, a good action $\hat{a}$ can be found by applying gradient ascent. Unfortunately, Q has proven to be impossible to learn with naive methods in complex environments. A function Q' which approximates Q well implicitely predicts some aspects of the future states. This paper proposes a method to make this predictive modelling both explicit and efficient. As such, the methods introduced can be considered as part of predictive-Q-learning.

Predicting the state transitions directly is too difficult. An assumption should be made that it is only a small subset of the features of the states which the agent needs to predict in order to model Q accurately. Then if an encoder mapped the observations to a latent space $\mathbb{Z}$, it would not be intractable to model the transitions of the states *projected in the latent space*. In other words, an assumption should be made that useful patterns are easier to notice when they are searched in the transitions of important learned features.

As a concrete illustration, picture a series of photos of pedestrians crossing the street at traffic lights. If one seeks patterns in an abstract representation space (for example, in textual descriptions), it will be evident that pedestrians crossing the street are usually preceded by the pedestrian "Walk" symbol turning on. But in the space of pixels, because of the variance and the higher number of dimensions, countless patterns with equal data support can be found. Therefore, the amount of data necessary to constrict the set of patterns such that the aforementionned one takes precedence is an efficiency bottleneck.

2 Methods

2.1 Latent Space Mapping

The first objective is to train a function f_{enc} which extracts the relevant features of the observation space $\mathbb{O}$ by mapping it to a latent space $\mathbb{Z}$. Therein lies the difficulty: which futures are useful? We assume that usefull features help predict, from observation o and action a, the expected return $Q(a, s)$, some grounding vector $G(s)$ and, most importantly, the mapping of future observations to $\mathbb{Z}$. We also add the requirement that a_t can be infered from z_t, for reasons which will be made clear in following paragraphs.

We therefore define the model f as the set of functions $\{f_{enc}, f_{\text{gdn}}, f_{act}, f_Q, f_{pred}\}$ such that $z_t = f_{enc}(z_{[0:t-1]}, o_t, a_t, z_{[0:t-1]})$, $\hat{g}_t = f_{\text{gdn}}(z_{[0:t]})$, $\hat{a}_t = f_{\text{act}}(z_{[0:t]})$, $\hat{q}_t = f_{\text{Q}}(z_{[0:t]})$, and $\hat{z}_{t+1} = f_{\text{pred}}(z_{[0:t]})$. The objective here is to create a bootstrapping process where each feature added to z adds difficulty in accurately predicting $\hat{z}_{t+1}$, leading to more features being added. The other constraints ensures that the inital features learned are useful for the agent and help with selecting actions.

2.2 Using f to generate actions

Suppose that f was well trained, then it should be expected that

$$\frac{\partial}{\partial a_t} Q(a_t, s_t) \approx \frac{\partial}{\partial a_t} \left[\gamma^k f_Q(\hat{z}_{[0:t+k]}) + \sum_{i=0}^{k-2} \gamma^i \left[f_Q(\hat{z}_{[0:t+i]}) - f_Q(\hat{z}_{[0:t+i+1]}) \right] \right] \tag{1}$$

where $\hat{z}_{t+i} \triangleq f_{\text{pred}}^{[[i]]}(z_{[0:t-1]}, f_{\text{enc}}(z_{[0:t-1]}, o_t, a_t))$ and $\hat{z}_{j<t} \triangleq z_j$. Here, $f^{[[i]]}(x)$ denotes i autoregressive applications of the function f on context x. And so Gradient Ascent can be used to find an action which yields a high expected return, at inference time.

2.2.1 Initial a

At first, the initial action will be given heuristically. However, after a sufficient number of actions have been generated, one can train a function to provide better starting points from which to start performing Gradient Ascent.

2.3 Training f

During the simulation of trajectories, one can store $\{o, a, z, g, r\}$ sets. We suggest performing gradient ascent with the derivative

$$\frac{d}{d\theta} \boldsymbol{L}(x|y;\theta) \triangleq \frac{d}{d\theta} ||V \odot (y(x) - \hat{y})||_2^2 \tag{2}$$

where V is a hyperparameter, $x := \langle o_t, a_t \rangle$, $y := \langle g_t, a_t, q_t, z_{t+1} \rangle$, $z_t := f_{\text{enc}}(z_{[0:t-1]}, o_t, a_t; \theta)$ and

$$\hat{y} := \left\langle f_{\text{gnd}}(z_{[0:t]}; \theta), f_{\text{act}}(z_{[0:t]}; \theta), f_{\text{Q}}(z_{[0:t]}; \theta), f_{\text{pred}}(z_{[0:t]}; \theta), \right\rangle$$

However, there are countless losses than one can try. This is left to future work.