Learning Latent Dynamics for Planning from Pixels

Danijar Hafner [1 2] Timothy Lillicrap [3] Ian Fischer [4] Ruben Villegas [1 5]
David Ha [1] Honglak Lee [1 5] James Davidson [1]

【强化学习 40】PlaNet

张楚珩
清华大学 交叉信息院博士在读

Google Brain和DeepMind合作的一个model-based强化学习算法——Deep Planning Network（PlaNet）。

原文传送门

Hafner, Danijar, et al. "Learning Latent Dynamics for Planning from Pixels." arXiv preprint arXiv:1811.04551 (2018).

特色

一种基于模型（model-based）的强化学习算法，能够直接图像作为输入来控制机器人；规划（planning）的过程在隐空间（latent space）上进行，速度较快；采取了一些措施避免了长距离规划中状态估计不准确的问题。

过程

1. 总体流程

既然是基于模型的方法，那么其两大问题主要就是***如何学习模型***和***如何利用模型来做规划***。

首先，模型包含些什么？

这里考虑一个基于图像的输入，这是一个高维度的输入，因此我们认为这是一个POMDP。图像输入是观测量 o_t ，状态是一个隐变量 s_t 。基于模型的方法中的“模型”指的是transition model $p(s_t|s_{t-1},a_{t-1})$ 和 reward model $p(r_t|s_t)$ ，POMDP还带来了observation model $p(o_t|s_t)$ 。另外我们还需要一个encoder，快速地利用可观测量 $o_{\leq t},a_{<t}$ 来估计隐变量 s_t ， $p(s_t|o_{\leq t},a_{<t})$ 。

总结一下，要学习以下四个模型的参数 transition model, reward model, observation model, encoder。

规划的流程如下，使用学习到的模型来针对不同的行动预测收益，并选取最好的下一步的行动。

$$o_{\leq t},a_{<t} \rightarrow \text{encoder} \rightarrow s_t \rightarrow \text{transition model} \rightarrow s_{t+1},s_{t+2},\cdots \rightarrow \text{reward model} \rightarrow R \rightarrow \text{CEM} \rightarrow a_t$$

模型学习的流程如下

$a_{1:T}, o_{1:T} \rightarrow$ SGD on variational bound $\rightarrow$ obervation and transition model, encoder

$a_{1:T}, o_{1:T} \rightarrow$ encoder $\rightarrow s_{1:T}(+r_{1:T}) \rightarrow$ reward model

整体算法如下

Algorithm 1: Deep Planning Network (PlaNet)

Input :

R	Action repeat	$p(s_t \mid s_{t-1}, a_{t-1})$	Transition model
S	Seed episodes	$p(o_t \mid s_t)$	Observation model
C	Collect interval	$p(r_t \mid s_t)$	Reward model
B	Batch size	$q(s_t \mid o_{\leq t}, a_{<t})$	Encoder
L	Chunk length	$p(\epsilon)$	Exploration noise
α	Learning rate		

Initialize dataset $\mathcal{D}$ with S random seed episodes.
Initialize model parameters θ randomly.
while *not converged* **do**
  `// Model fitting`
  for *update step* $s = 1..C$ **do**
    Draw sequence chunks $\{(o_t, a_t, r_t)_{t=k}^{L+k}\}_{i=1}^{B} \sim \mathcal{D}$ uniformly at random from the dataset.
    Compute loss $\mathcal{L}(\theta)$ from Equation 7.
    Update model parameters $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta)$.
  `// Data collection`
  $o_1 \leftarrow$ `env.reset()`
  for *time step* $t = 1..\lceil \frac{T}{R} \rceil$ **do**
    Infer belief over current state $q(s_t \mid o_{\leq t}, a_{<t})$ from the history.
    $a_t \leftarrow$ `planner`$(q(s_t \mid o_{\leq t}, a_{<t}), p)$, see Algorithm 2 in the appendix for details.
    Add exploration noise $\epsilon \sim p(\epsilon)$ to the action.
    for *action repeat* $k = 1..R$ **do**
      $r_t^k, o_{t+1}^k \leftarrow$ `env.step`(a_t)
    $r_t, o_{t+1} \leftarrow \sum_{k=1}^{R} r_t^k, o_{t+1}^R$
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(o_t, a_t, r_t)_{t=1}^{T}\}$

其中第4~7行表示的是模型训练的部分，第11行表示的是模型的规划部分，第8~16行表示的模型的规划连同行动和采样的部分。

2. 如何利用模型来做规划？

先假设我们前面提到的各个模型的参数已经学习到了，那么如何来利用这些模型来做规划呢？这里使用了一个最为常用的框架，叫做model-predictive control（MPC）。简单说来就是每一步都往后面模拟很多步，然后选取一个收益最高的方案，然后只采取一步的行动；等到下一步的时候再重新规划并且选取新的一步；即并不是规划出一串行动之后开环地做着一串，而是规划一串只走一步。

回顾刚刚提到的流程框架

$$o_{\leq t}, a_{<t} \rightarrow \text{encoder} \rightarrow s_t \rightarrow \text{transition model} \rightarrow s_{t+1}, s_{t+2}, \cdots \rightarrow \text{reward model} \rightarrow R \rightarrow \text{CEM} \rightarrow a_t$$

我们先基于先前的观察使用encoder得到当前状态的估计，然后对于未来的行动序列进行采样，并结合transition model来预测未来的状态序列，同时使用reward model来估计获得奖励的期望值。这样对于一个给定的未来行动序列就能够得到一个期望收益了，把这一套当做一个黑盒子我们就能利用CEM来规划出一个好的未来行动序列了。取这个行动序列的第一个行动作为下一步我们选择的行动即可。具体算法如下。

Algorithm 2: Latent planning with CEM

Input : H Planning horizon distance; I Optimization iterations; J Candidates per iteration; K Number of top candidates to fit; $q(s_t \mid o_{\leq t}, a_{<t})$ Current state belief; $p(s_t \mid s_{t-1}, a_{t-1})$ Transition model; $p(r_t \mid s_t)$ Reward model

Initialize factorized belief over action sequences $q(a_{t:t+H}) \leftarrow \text{Normal}(0, \mathbb{I})$.
for *optimization iteration* $i = 1..I$ **do**
  // Evaluate J action sequences from the current belief.
  for *candidate action sequence* $j = 1..J$ **do**
    $a^{(j)}_{t:t+H} \sim q(a_{t:t+H})$
    $s^{(j)}_{t:t+H+1} \sim q(s_t \mid o_{1:t}, a_{1:t-1}) \prod_{\tau=t+1}^{t+H+1} p(s_\tau \mid s_{\tau-1}, a^{(j)}_{\tau-1})$
    $R^{(j)} = \sum_{\tau=t+1}^{t+H+1} \mathrm{E}[p(r_\tau \mid s^{(j)}_\tau)]$
  // Re-fit belief to the K best action sequences.
  $\mathcal{K} \leftarrow \text{argsort}(\{R^{(j)}\}_{j=1}^{J})_{1:K}$
  $\mu_{t:t+H} = \frac{1}{K} \sum_{k \in \mathcal{K}} a^{(k)}_{t:t+H}, \quad \sigma_{t:t+H} = \frac{1}{K-1} \sum_{k \in \mathcal{K}} |a^{(k)}_{t:t+H} - \mu_{t:t+H}|.$
  $q(a_{t:t+H}) \leftarrow \text{Normal}(\mu_{t:t+H}, \sigma^2_{t:t+H}\, \mathbb{I})$
return *first action mean* μ_t.

3. 如何学习模型？

通过采样得到的数据是观察到的图像 $o_{1:T}$ 和 $a_{1:T}$ ，因此我们需要同时训练多个模型使得出现该数据的概率最大（maximum log-likelihood）。

One-step predictive distribution The variational bound for latent dynamics models $p(o_{1:T}, s_{1:T} \mid a_{1:T}) = \prod_t p(s_t \mid s_{t-1}, a_{t-1}) p(o_t \mid s_t)$ and a variational posterior $q(s_{1:T} \mid o_{1:T}, a_{1:T}) = \prod_t q(s_t \mid o_{\leq t}, a_{<t})$ follows from importance weighting and Jensen's inequality as shown,

$$
\begin{aligned}
\ln p(o_{1:T} \mid a_{1:T}) &\triangleq \ln \mathrm{E}_{p(s_{1:T} \mid a_{1:T})} \left[\prod_{t=1}^{T} p(o_t \mid s_t) \right] \\
&= \ln \mathrm{E}_{q(s_{1:T} \mid o_{1:T}, a_{1:T})} \left[\prod_{t=1}^{T} p(o_t \mid s_t) p(s_t \mid s_{t-1}, a_{t-1}) / q(s_t \mid o_{\leq t}, a_{<t}) \right] \\
&\geq \mathrm{E}_{q(s_{1:T} \mid o_{1:T}, a_{1:T})} \left[\sum_{t=1}^{T} \ln p(o_t \mid s_t) + \ln p(s_t \mid s_{t-1}, a_{t-1}) - \ln q(s_t \mid o_{\leq t}, a_{<t}) \right] \\
&= \sum_{t=1}^{T} \Big(\underbrace{\mathrm{E}_{q(s_t \mid o_{\leq t}, a_{<t})} [\ln p(o_t \mid s_t)]}_{\text{reconstruction}} - \underbrace{\mathrm{E}_{q(s_{t-1} \mid o_{\leq t-1}, a_{<a-1})} \left[\mathrm{KL}[q(s_t \mid o_{\leq t}, a_{<t}) \parallel p(s_t \mid s_{t-1}, a_{t-1})] \right]}_{\text{complexity}} \Big).
\end{aligned}
\tag{8}
$$

这里完全是按照ELBO（evidence lower bound）的推导。注意，不等式的右边包含encoder $q(s_t|o_{\leq t}, a_{<t})$ ， observation model $p(o_t|s_t)$ 和transition model $p(s_t|s_{t-1}, a_{t-1})$ 。通过对于右式做随机梯度上升即可学到这三个模型。

Reward model的学习就比较简单了，类似于普通的有监督学习。

至此，大致的框架已经讲完了，但是本文的创新点是提出了以下两种技术，使得planning过程中状态的估计更为准确。

4. Recurrent State Space Model （RSSM）

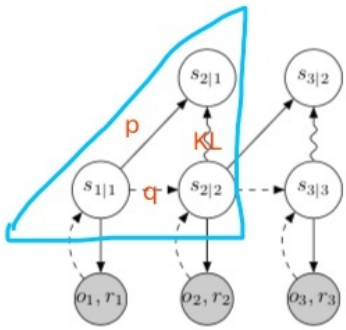
前面我们提到状态的转移模型 transition model 是表示为 $s_t \sim p(s_t|s_{t-1}, a_{t-1})$ ，文中都是使用的diagonal Gaussian distribution，即由神经网络生成下一步状态的均值和方差，并从高斯模型中进行采样。如果每一个单步的transition model都是完美的，那么逐步估计也会比较准确，但是单步的模型并不完美，它受限于不充足的训练以及较弱的模型表示能力。每一步都完全是随机采样，在经过很多步之后，前面信息就会丢失掉。因此，这里作者还引入了确定性（非随机）的部分。（当然了，如果神经网络自己学出来某一部分的方差就为0，那么就等效于加上确定性部分了，不过不能期望神经网络能够自己学出来）

通过在状态隐变量中加上确定性的部分，就得到了RSSM。transition model变为了 $h_t = f(h_{t-1}, s_{t-1}, a_{t-1}), s_t \sim p(s_t|h_t)$ ，即原来的 s_t 拆成了 (h_t, s_t) 两个部分。相应地，encoder也进行了一些改变， $q(s_{1:T}|o_{1:T}, a_{1:T}) = \prod_t q(s_t|o_{\leq t}, a_{<t}) \rightarrow q(s_{1:T}|o_{1:T}, a_{1:T}) = \prod_t q(s_t|h_t, o_t) = \prod_t q(s_t|f(h_{t-1}, s_{t-1}, a_{t-1}), o_t)$ 。

这样的改变只是改变了网络的结构，训练的目标和方式没有大的变化。

5. Latent Overshooting

前面推导到的下界（lower bound）是优化的目标，transition model的学习主要是和encoder $q(s_t|o_{\leq t}, a_{<t})$ 的比较得来的，它们之间的比较是一步一步地比较。如下图所示，蓝色框中的虚线代表encoder一步步的推断，实线代表transition model的生成过程，波浪线代表需要对两者生成的 s_t 分布进行比较，并且最小化它们之间的KL散度。



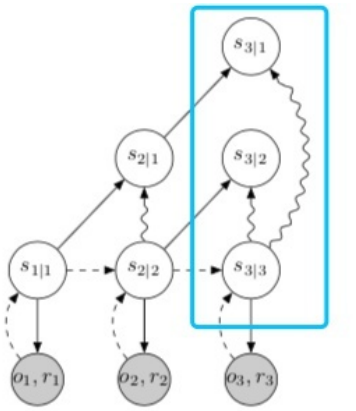
(a) Standard variational bound

图：原变分下界对应的模型。虚线代表推断，实线代表生成过程，波浪线代表KL散度约束。

可以观察到，在这样的做法中，transition model始终是一步步分离开的，并没有像之后使用的时候那样多步串在一起去预测一个未来较长时间内的状态分布。少了多步串联起来的训练，得到的transition model在多步串联的时候会产生较大的误差，因此文章采用了以下latent overshooting的方法来改善。

Latent overshooting的做法如下图所示。以 s_3 为例，不仅仅是一步转移产生的状态 $s_{3|2}$ 要和encoder产生的状态 $s_{3|3}$ 作比较并减少差距，而且多步转移（这里是两步）产生的状态 $s_{3|1}$ 也要和 $s_{3|3}$ 作比较并最小化KL散度。这相当于不仅要求transition model在走一步的情况下要和encoder对的上，而且

要求它在串联工作的时候也要和encoder对的上。



(c) Latent overshooting

图：Latent overshooting对应的模型。虚线代表推断，实线代表生成过程，波浪线代表KL散度约束。

考虑了latent overshooting之后的变分下界写作

$$\frac{1}{D}\sum_{d=1}^{D}\ln p_d(o_{1:T}) \geq \sum_{t=1}^{T}\Bigg(\underbrace{\mathrm{E}_{q(s_t|o_{\leq t})}[\ln p(o_t \mid s_t)]}_{\text{reconstruction}} \hookleftarrow \underbrace{-\frac{1}{D}\sum_{d=1}^{D}\beta_d \mathrm{E}_{p(s_{t-1}|s_{t-d})q(s_{t-d}|o_{\leq t-d})}\left[\mathrm{KL}[q(s_t \mid o_{\leq t}) \parallel p(s_t \mid s_{t-1})]\right]}_{\text{latent overshooting}}\Bigg). \quad (7)$$

其中， D 表示transition model最多串联的步数，可以看到，KL散度项被替换为了多对约束的加权平均，权重通过 β_d 来调节。

实验结果

这个工作跑了如下六组实验，这六组实验都有各自的特点，比如有些具有比较复杂的动力学（与环境有接触）、有些只能观察到部分的信息、有些具有稀疏的奖励结构。同时，该工作着重说明了RSSM和Latent overshooting在其中发挥的作用。

此外，该工作还训练了一个模型，使得该模型能够在所有的环境中运行。

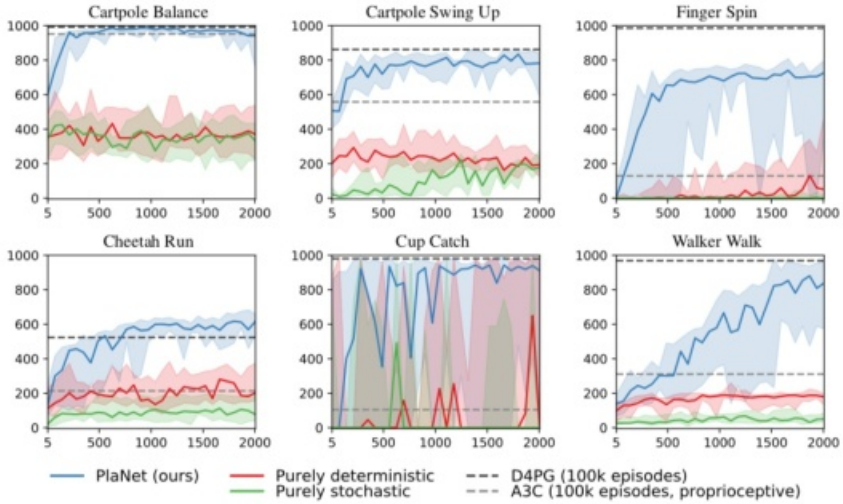


Figure 4: Comparison of PlaNet to model-free algorithms and other model designs. Plots show test performance for the number of collected episodes. We compare PlaNet using our RSSM (Section 3) to purely deterministic (RNN) and purely stochastic models (SSM). The RNN does not use latent overshooting, as it does not have stochastic latents. The lines show medians and the areas show percentiles 5 to 95 over 4 seeds and 10 rollouts.

文章来源：导师推荐

发布于 2019-02-23

强化学习 (Reinforcement Learning)

文章被以下专栏收录