📖

Offline Reinforcement Learning as One Big Sequence Modeling Problem —hoho

论文试图解决什么问题？

如何将RL建模为序列模型，从而简化决策的设计。

这是否是一个新的问题？

本文作者提出Trajectory Transformer，在当时不是一个新问题。跟Decision Transformer一样，虽然也有将序列模型应用与RL的研究，但他们的研究还是基于传统的RL方法，而本文方法是直接将传统的RL流程架构替换为transformer架构。

本文的出现比Decision Transformer要晚。

这篇文章要验证一个什么科学假设？

本文通过三个场景验证用transformer建模是否可以应用与解决RL的某些问题：

1. 模仿学习
2. 目标为导向的强化学习（goal-conditioned）
3. 离线强化学习

有哪些相关研究？如何归类？谁是这一课题在领域内值得关注的研究员？

hoho_todo

论文中提到的解决方案之关键是什么？

将状态的每个维度，和动作的每个维度拆分，模型通过自回归的方式一个维度一个维度来输出，

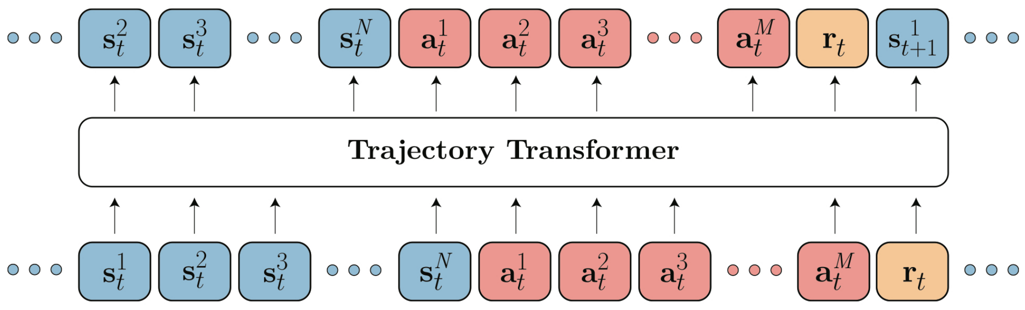

Figure 1 (Architecture) The Trajectory Transformer trains on sequences of (autoregressively discretized) states, actions, and rewards. Planning with the Trajectory Transformer mirrors the sampling procedure used to generate sequences from a language model.

所以，算法的训练目标为：

$$\mathcal{L}(\tau) = \sum_{t=1}^{T} \left(\sum_{i=1}^{N} \log P_\theta\left(s_t^i \mid \mathbf{s}_t^{<i}, \boldsymbol{\tau}_{<t}\right) + \sum_{j=1}^{M} \log P_\theta\left(a_t^j \mid \mathbf{a}_t^{<j}, \mathbf{s}_t, \boldsymbol{\tau}_{<t}\right) + \log P_\theta\left(r_t \mid \mathbf{a}_t, \mathbf{s}_t, \boldsymbol{\tau}_{<t}\right) \right),$$

如果依次生成每一个维度，可能最后得到的动作不会特别好。因此使用了 beam search 来得到较好的整体动作的输出。

Algorithm 1 Beam search

```
1: Require Input sequence $\mathbf{x}$, vocabulary $\mathcal{V}$, sequence length $T$, beam width $B$
2: Initialize $Y_0 = \{\ (\ )\ \}$
3: for $t = 1, \dots, T$ do
4:     $\mathcal{C}_t \leftarrow \{\mathbf{y}_{t-1} \circ y \mid \mathbf{y}_{t-1} \in Y_{t-1} \text{ and } y \in \mathcal{V}\}$    // candidate single-token extensions
5:     $Y_t \leftarrow \underset{Y \subseteq \mathcal{C}_t,\ |Y|=B}{\operatorname{argmax}} \log P_\theta(Y \mid \mathbf{x})$    // $B$ most likely sequences from candidates
6: end for
7: Return $\underset{\mathbf{y} \in Y_T}{\operatorname{argmax}} \log P_\theta(\mathbf{y} \mid \mathbf{x})$
```

以下分别从三个场景来说明具体的做法：

1. 对于模仿学习：

模型训练好后，用 beam search 算法直接使用就可以达到模仿学习的效果。其中，把 y 当做 actions来预测，把 x 当做state 和之前的轨迹。

2. 对于目标为导向的强化学习：

一般我们希望把最后的状态s_T作为模型达到的目标，只需要把最后的状态放在轨迹的前面即可：

$$\{\mathbf{s}_T, \mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_{T-1}\}.$$

所以就有：

$$P_\theta(s_t^i \mid \mathbf{s}_t^{<i}, \boldsymbol{\tau}_{<t}, \mathbf{s}_T)$$

3. 对于离线强化学习：

原本的 beam search 是挑选 log probability 最大的动作，这里改成 先挑选log probability 较大的动作，然后在这些动作对应的回合中挑选相应 reward-to-go 预测值最大的那些动作。

论文中的实验是如何设计的？

在一个物理模拟人动作实验上，让普通的前向预测模型和本文方法对对比。

并且将tranformer的注意力模式可视化。

用于定量评估的数据集是什么？代码有没有开源？

不用数据集。

代码：https://trajectory-transformer.github.io/

论文中的实验及结果有没有很好地支持需要验证的科学假设？

发现transformer与参考模型几乎一样，而使用普通的前向预测模型效果则很差

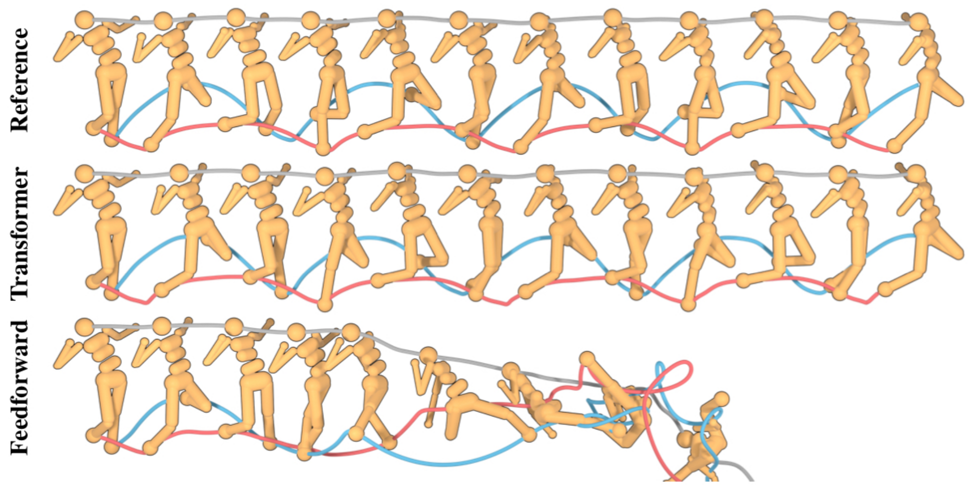

Figure 2 (Prediction visualization) A qualitative comparison of length-100 trajectories generated by the Trajectory Transformer and a feedforward Gaussian dynamics model from PETS, a state-of-the-art planning algorithm Chua et al. (2018). Both models were trained on trajectories collected by a single policy, for which a true trajectory is shown for reference. Compounding errors in the single-step model lead to physically implausible predictions, whereas the Transformer-generated trajectory is visually indistinguishable from those produced by the policy acting in the actual environment. The paths of the feet and head are traced through space for depiction of the movement between rendered frames.

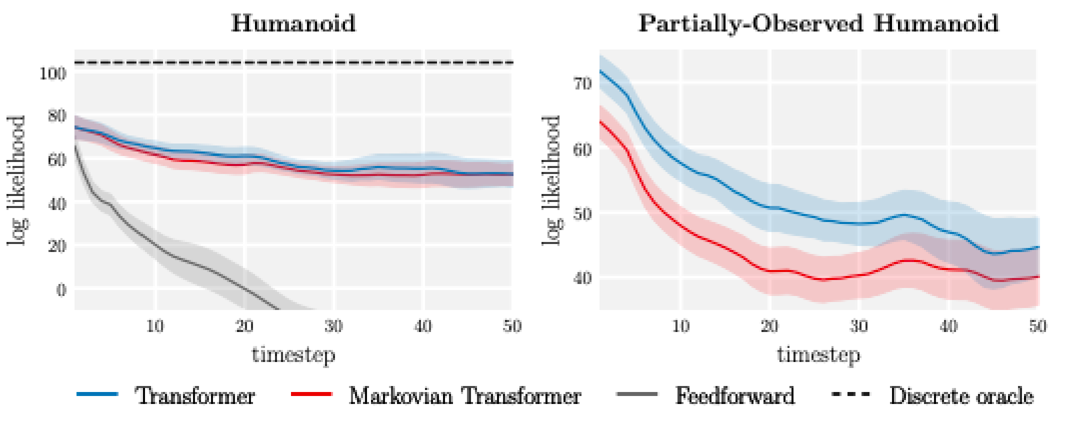

Figure 3 (Compounding model errors) We compare the accuracy of the Trajectory Transformer (with uniform discretization) to that of the probabilistic feedforward model ensemble (Chua et al., 2018) over the course of a planning horizon in the humanoid environment, corresponding to the trajectories visualized in Figure 2. The Trajectory Transformer has substantially better error compounding with respect to prediction horizon than the feedforward model. The discrete oracle is the maximum log likelihood attainable given the discretization size; see Appendix ?? for a discussion.

通过将注意力可视化

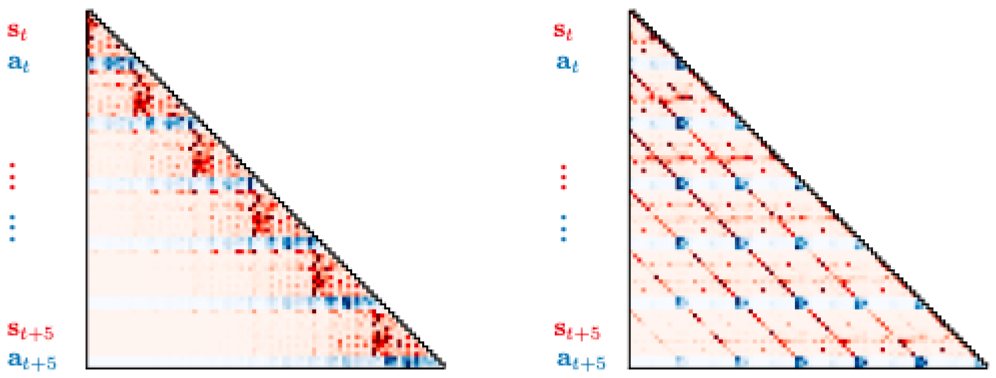

Figure 4 (Attention patterns) We observe two distinct types of attention masks during trajectory prediction. In the first, both states and actions are dependent primarily on the immediately preceding transition, corresponding to a model that has learned the Markov property. The second strategy has a striated appearance, with state dimensions depending most strongly on the same dimension of multiple previous timesteps. Surprisingly, actions depend more on past actions than they do on past states, reminiscent of the action smoothing used in some trajectory optimization algorithms (Nagabandi et al., 2019). The above masks are produced by a first- and third-layer attention head during sequence prediction on the hopper benchmark; reward dimensions are omitted for this visualization.[1]

这篇论文到底有什么贡献？

hoho_todo

下一步呢？有什么工作可以继续深入？

hoho_todo