Reinforcement learning via sequence modeling - Beyond Markovian assumption

Presenter: Liangqi Liu

The Chinese University of Hong Kong, Shenzhen, China

July 6, 2021

Outline

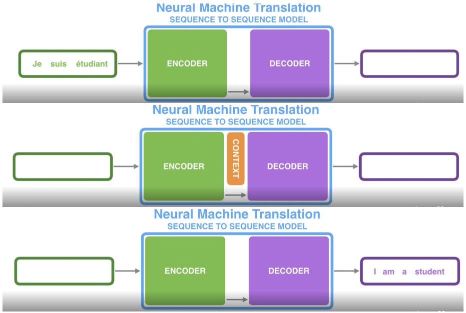
Basic description

- View RL as a sequence modeling problem, with the goal being to predict a sequence of actions that leads to a sequence of high rewards
- Train a single high-capacity sequence model to represent the joint distribution over sequences of states, actions, and rewards
- Produce a simpler method whose effectiveness is determined by the representational capacity of the sequence model rather than algorithmic sophistication
- Demonstrate the flexibility of this approach across long-horizon dynamics prediction, imitation learning, goal-conditioned RL, and offline RL

Outline

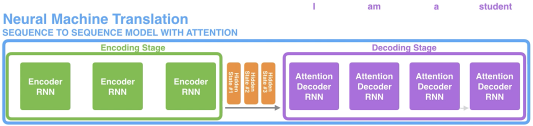
Seq2Seq model

- Sequence-to-Sequence (or Seq2Seq) is a neural net that transforms a given sequence of elements, such as the sequence of words in a sentence, into another sequence
- Seq2Seq models consist of an encoder and a decoder



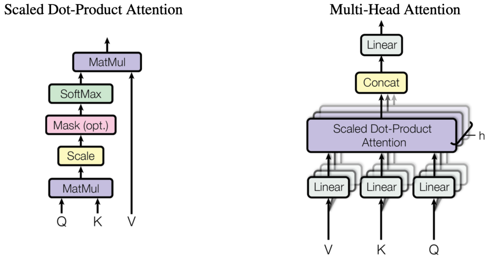
Attention

- The attention-mechanism looks at an input sequence and decides at each step which other parts of the sequence are important
- The Encoder writes down keywords that are important to the semantics of the sentence, and gives them to the Decoder

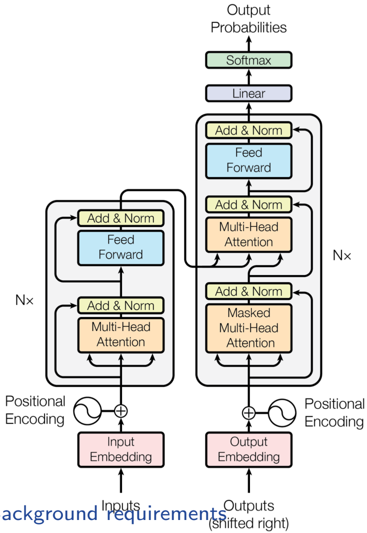


Attention

- Scaled Dot-Product Attention: $\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V$
- Q: query, vector representation of one word in the sequence
- K: key, vector representations of all the words in the sequence
- V: value, vector representations of all the words in the sequence



Transformer



- Use self-attention to boost the speed
- Each encoder consists of two layers: Self-attention and a feed Forward Neural Network
- Self-attention allows the models to associate each word in the input, to other words
- The pointwise feed-forward network is a couple of linear layers with a ReLU activation in between
- The residual connections help the network train, by allowing gradients to flow through the networks directly

Outline

Basic introduction

- the Trajectory Transformer is a substantially more reliable long-horizon predictor than conventional dynamics models, even in Markovian environments for which the standard model parameterization is in principle sufficient
- Trajectory Transformers can attain results on offline reinforcement learning benchmarks that are competitive with state-of-the-art prior methods designed specifically for that setting
- combined with a modified beam search procedure that decodes trajectories with high reward, rather than just high likelihood

Input of the transformer

- A trajectory τ consists of N-dimensional states, M-dimensional actions, and scalar rewards:

$$\tau = \left\{\mathbf{s}_t^0, \mathbf{s}_t^1, \ldots, \mathbf{s}_t^{N-1}, \mathbf{a}_t^0, \mathbf{a}_t^1, \ldots, \mathbf{a}_t^{M-1}, r_t\right\}_{t=0}^{T-1}$$

- each step in the sequence therefore corresponds to a dimension of the state, action, or reward,such that a trajectory with T time steps would correspond to a sequence of length $T * (N + M + 1)$
- For continuous states and actions

$$\bar{\mathbf{s}}_t^i = \left\lfloor V \frac{\mathbf{s}_t^i - \ell^i}{r^i - \ell^i} \right\rfloor + Vi$$

- Use a regular grid with a fixed number of bins per dimension
- Ensure that different state dimensions are represented by disjoint sets of tokens

Training

- Loss function

$$\mathcal{L}(\bar{\tau}) = \sum_{t=0}^{T-1} \left(\sum_{i=0}^{N-1} \log P_\theta \left(\bar{\mathbf{s}}_t^i \mid \bar{\mathbf{s}}_t^{<i}, \bar{\tau}_{<t} \right) + \sum_{j=0}^{M-1} \log P_\theta \left(\bar{\mathbf{a}}_t^j \mid \bar{\mathbf{a}}_t^{<j}, \bar{\mathbf{s}}_t, \bar{\tau}_{<t} \right) + \log P_\theta \left(\bar{r}_t \mid \bar{\mathbf{a}}_t, \bar{\mathbf{s}}_t, \bar{\tau}_{<t} \right) \right)$$

- $\bar{\tau}_{<t}$: a shorthand for a tokenized trajectory from timesteps 0 through $t1$
- probabilities are written as conditional on all preceding tokens in a trajectory

Testing: Beam search

Algorithm 1 Beam search

```
1:  Require State s, vocabulary 𝒱
2:  Require Sequence length L, beam width B
3:  Discretize s to s̄ (Equation 1)
4:  Initialize 𝒯_0 = {([s̄], 0)} and 𝒯_{1:L} = ∅
5:  for l ∈ {1, ··· , L} do
6:      for (τ̄_{l−1}, q_{l−1}) ∈ 𝒯_{l−1}, v ∈ 𝒱 do
7:          τ̄_l ← τ̄_{l−1} + [v]
8:          q_l ← q_{l−1} + log P_θ(v | τ̄_{l−1})
9:          𝒯_l ← 𝒯_l ∪ (τ̄_l, q_l)
10:     end for
// Select B most probable sequences
11:     𝒯_l ← arg max_{𝒯⊆𝒯_l,|𝒯|=B} Σ_{(τ̄,q)∈𝒯} {q}
12: end for
13: Return arg max_{τ̄|(τ̄,q)∈𝒯_L} {q}
```

- V: output dictionary
- $\bar{\tau}$: trajectory, q: corresponding probability
- Use beam search to search the B predicted trajectory with the largest likelihood probability, and then select the trajectory with the largest reward and reward-to-go as the final prediction result
- reward-to-go

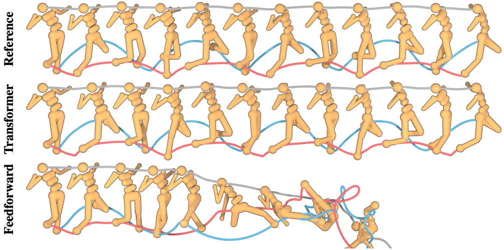
$$R_t = \sum_{t'=t}^{T-1} \gamma^{t'-t} r_{t'}$$

Decision transformer

- Trajectory representation: $\tau = \left(\widehat{R}_1, s_1, a_1, \widehat{R}_2, s_2, a_2, \ldots, \widehat{R}_T, s_T, a_T\right)$
- Instead of feeding the rewards directly, we feed the model with the returns-to-go: $\widehat{R}_t = \sum_{t'=t}^{T} r_{t'}$

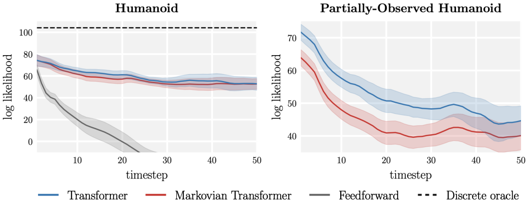
Outline

Experiments: Trajectory predictions



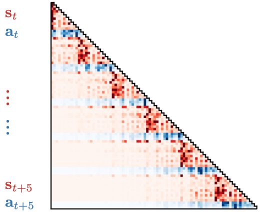
- Predict 100-timestep trajectories from this model after having trained on a dataset collected by a trained humanoid policy
- Feedward: a feedforward Gaussian dynamics model from PETS, a state-of-the-art planning algorithm

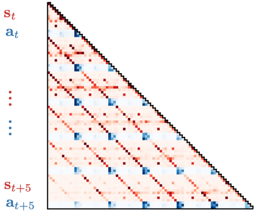
Experiments: Trajectory predictions



- 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
- Markovian transformer: a Markovian variant of our same architecture. This ablation has a truncated context window that prevents it from attending to more than one timestep in the past

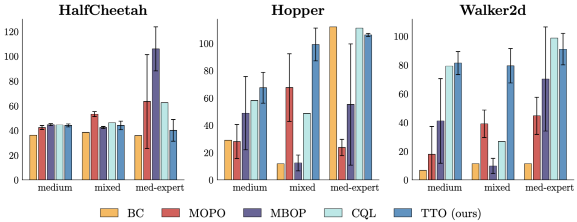
Experiments: Attention patterns





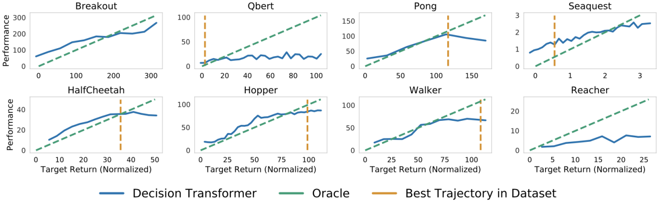
- Produced by a first-layer and third-layer attention head
- In the first, both states and actions are dependent primarily on the immediately preceding transition, corresponding to a model that has learned the Markov property
- In the second, actions depend more on past actions than they do on past states, reminiscent of the action smoothing used in some trajectory optimization algorithms

Experiments: Offline reinforcement learning



- CQL: conservative Q-learning; MOPO: model-based offline policy optimization; MBOP: model-based offline planning; BC: behavior cloning
- MBOP provides a point of comparison for a planning algorithm that uses a single-step dynamics model as opposed to a Transformer

How well does Decision Transformer model the distribution of returns



- The average sampled return accumulated by the agent over the course of the evaluation episode for varying values of target return
- The desired target returns and the true observed returns are highly correlated
- Decision Transformer is sometimes capable of extrapolation

How can Decision Transformer benefit online RL regimes

- Offline RL and the ability to model behaviors has the potential to enable sample-efficient online RL for downstream tasks
- Decision Transformer can meaningfully improve online RL methods by serving as a strong model for behavior generation
- Decision Transformer can serve as a powerful "memorization engine"