Continuous Control: Basics and Beyond

阴钰

2019.11.12

Outline

- Background
- Problem and Modeling
- Algorithms for Continuous Control
- Applications
- Conclusion
- References

Background

- Continuous control

Background

- Characteristics
  - Continuous action space
  - (Mostly) discrete time
  - Long-term feedback

Outline

- Background
- Problem and Modeling
- Algorithms for Continuous Control
- Applications
- Conclusion
- References

Problem Definition

- Modeling control problem: MDP
  - General case: stochastic environment, arbitrary action space
  - State: $\mathcal{S}$
  - Action: $\mathcal{A}$
  - State transition: $P(s_{t+1} | s_t, a_t)$
  - Reward: $r(s_t, a_t)$
  - Discount factor: γ

Problem Definition

- Agent
  - General case: stochastic policy

$$\pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$$

  - Deterministic policy

$$\pi : \mathcal{S} \to \mathcal{A}$$

Problem Definition

- Objective
  - Trajectory under policy π

$$s_1 \xrightarrow{a_1 \sim \pi, s_2 \sim P(s_2|s_1,a_1)} s_2 \xrightarrow{a_2 \sim \pi, s_3 \sim P(s_3|s_2,a_2)} s_3 \ldots$$

  - Return (from time t)

$$G_t = \sum_{i=t}^{T} \gamma^{i-t} r(s_i, a_i)$$

  - Expected return (objective)

Random variable

$$J = \mathbb{E}_{\underline{(s_t,a_t)\sim\rho_\pi}} [\underline{G_t}]$$

Following policy trajectory distribution

Problem Definition

- Value functions
  - Commonly used expectations
  - State value function

$$V^{\pi}(s_t) = \mathbb{E}_{\pi}[G_t | s_t]$$

  - Action value function

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{\pi}[G_t | s_t, a_t]$$

Basic Methods

- Bellman Equation
  - Action-value function

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{\pi}[G_t | s_t, a_t]$$

  - **Bellman equation** (unfold one term)

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{s_{t+1} \sim P(s_{t+1}|s_t, a_t)}\left[r(s_t, a_t) + \gamma \mathbb{E}_{a_{t+1} \sim \pi}[Q^{\pi}(s_{t+1}, a_{t+1})]\right]$$

  - Deterministic case

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{s_{t+1} \sim P(s_{t+1}|s_t, a_t)}\left[r(s_t, a_t) + \gamma Q^{\pi}(s_{t+1}, \pi(s_{t+1}))\right]$$

Basic Methods

- **TD** (Temporal Difference) Learning
  - For deterministic policies
  - Approximate Q^π using $Q(s, a; \theta_Q)$

$$Q^\pi(s_t, a_t) = \mathbb{E}_{s_{t+1} \sim P(s_{t+1}|s_t, a_t)}\left[r(s_t, a_t) + \gamma Q^\pi(s_{t+1}, \pi(s_{t+1}))\right]$$

$$L(\theta_Q) = \mathbb{E}_{\underbrace{(s_t, a_t) \sim \rho_\beta}_{\text{arbitrary policy}}, s_{t+1} \sim E}\left[\mathcal{L}[Q(s_t, a_t; \theta^Q), y_t]\right]$$

$$y_t = r(s_t, a_t) + \gamma Q(s_{t+1}, \pi(s_{t+1}); \underline{\theta_Q})$$

often omitted or replaced
by target network

Basic Methods

- Discrete case: Q-Learning
  - Approximate action function Q^* for best deterministic policy π^*
  - With Q^*, π^* can be expressed as: $\pi^*(s) = \arg\max_{a \in \mathcal{A}} Q^*(s, a)$
  - Bellman equation:

$$Q^*(s_t, a_t) = \mathbb{E}_{s_{t+1} \sim E}\left[r(s_t, a_t) + \gamma \max_{a_{t+1} \in \mathcal{A}} Q^*(s_{t+1}, a_{t+1})\right]$$

  - Loss function for basic Q-Learning:

$$L(\theta_Q) = \mathbb{E}_{s_t, a_t, s_{t+1}, r}\left[\mathcal{L}\left[Q(s_t, a_t; \theta_Q), r + \gamma \max_{a \in \mathcal{A}} Q(s_{t+1}, a; \underline{\theta_Q})\right]\right]$$

often omitted or replaced by target network

Basic Methods

- Stochastic policies: policy gradient
  - Given policy as parameterized function: $P(a|s) = \pi(s, a; \theta^{\pi})$
  - Objective: maximize expected return using gradient descent

$$J = \mathbb{E}_{\pi}[V^{\pi}(s_t)] = \mathbb{E}_{\pi}[Q^{\pi}(s_t, a_t)]$$

  - Policy Gradient Theorem:

$$\begin{aligned}\nabla_{\theta_\pi} J &= \nabla_{\theta_\pi} \mathbb{E}_{\pi}[Q^{\pi}(s_t, a_t)] \\ &\propto \mathbb{E}_{\pi}[Q^{\pi}(s_t, a_t) \nabla_{\theta_\pi} \log \pi(s_t, a_t; \theta_\pi)]\end{aligned}$$

Basic Methods

- REINFORCE
  - Approximate expected return directly from sampling

$$
\begin{aligned}
\nabla_{\theta_\pi} J &\propto \mathbb{E}_\pi[Q^\pi(s_t, a_t)\nabla_{\theta_\pi} \log \pi(s_t, a_t; \theta_\pi)] \\
&= \mathbb{E}_\pi[\underline{G_t}\nabla_{\theta_\pi} \log \pi(s_t, a_t; \theta_\pi)]
\end{aligned}
$$

estimate through sample trajectory

Basic Methods

- Actor-Critic
  - Actor: policy
  - Critic: value function approximator
- A3C/A2C
  - Approximate state value with $V(s_t;\theta_V)$
  - Policy gradient

$$\nabla_{\theta_\pi} J \propto \mathbb{E}_\pi[(Q^\pi(s_t,a_t) - V^\pi(s_t))\nabla_{\theta_\pi} \log \pi(s_t,a_t;\theta_\pi)]$$
$$\approx \mathbb{E}_\pi[(\underbrace{G_t - V(s_t;\theta_V)}_{\text{advantage}})\nabla_{\theta_\pi} \log \pi(s_t,a_t;\theta_\pi)]$$

  - Critic loss: $\mathcal{L}(G_t, V(s_t;\theta_V))$

On-policy and Off-policy

- Definition and comparison
  - **On-policy**: training samples collected from the target policy
    - Example: REINFORCE

$$\nabla_{\theta_\pi} J \propto \mathbb{E}_\pi [G_t \nabla_{\theta_\pi} \log \pi(s_t, a_t; \theta_\pi)]$$

    - G_t must be sampled from current policy
  - **Off-policy**: training samples can come from different policy
    - Enables replay buffer: reuse history for better sample efficiency
    - Better exploration: may take arbitrary policy when exploring
    - Example: Q-Learning

$$L(\theta_Q) = \mathbb{E}_{s_t, a_t, s_{t+1}, r} \left[\mathcal{L} \left[Q(s_t, a_t; \theta_Q), r + \gamma \max_{a \in \mathcal{A}} Q(s_{t+1}, a; \theta_Q) \right] \right]$$

Outline

- Background
- Problem and Modeling
- **Algorithms for Continuous Control**
- Applications
- Conclusion
- References

Continuous Case

- Challenges
  - Policy representation: continuous
  - Adapting Q-Learning: not straightforward
  - Stability issues

DPG

- PG for deterministic policy:
  - Deterministic policy: $a = \mu(s)$
  - Policy Gradient Theorem for deterministic policy:

$$\nabla_{\theta_\mu} J = \underbrace{\mathbb{E}_{s \sim \rho^\beta}}_{\text{off-policy}} [\underbrace{\nabla_a Q^\mu(s, a) \nabla_{\theta_\mu} \mu(s; \theta_\mu)}_{\text{chain rule}}|_{a=\mu(s;\theta_\mu)}]$$

DDPG

- DDPG: Deep Deterministic Policy Gradient
  - Actor-critic framework
  - Value function approximator: $Q(s, a; \theta_Q)$
  - Policy gradient (actor loss):

$$\nabla_{\theta_\mu} J = \mathbb{E}_{s \sim \rho^\beta}[\nabla_a Q(s, a; \theta_Q)|_{s=s_t, a=\mu(s_t;\theta_\mu)} \nabla_{\theta_\mu} \mu(s; \theta_\mu)|_{s=s_t}]$$

off-policy

  - Critic loss: Q Learning (on deterministic policy)

$$L(\theta_Q) = \mathbb{E}_{s_t, a_t, s_{t+1}, r}\left[\mathcal{L}[Q(s_t, a_t; \theta_Q), r + \gamma Q(s_{t+1}, \mu(s_{t+1}); \theta_Q)]\right]$$

DDPG

- Stability tricks
  - Replay buffer: for i.i.d. samples in critic learning
  - Target network for both actor and critic in critic learning
  - Soft update
  - Batch normalization: automatic scaling
  - Actor noise: better exploration

SAC

- SAC: Soft Actor-Critic
  - Stochastic action representation: $\pi(\mathbf{a}_t|\mathbf{s}_t)$
  - Entropy term in objective function:

maximize entropy

$$J = \sum_{t=1}^{T} \mathbb{E}_\pi[r(s_t, a_t) + \underline{\alpha\mathcal{H}(\pi(\cdot|s_t}))]$$

    - Acts as randomly as possible while still able to succeed at the task
  - Learned functions
    - Policy: parameterized distribution $\pi(\cdot\,;\theta_\pi)$
    - Soft action value (update through Bellman equation):

$$Q(s_t, a_t; \theta_Q) \to r(s_t, s_{t+1}) + \gamma\mathbb{E}_{s_{t+1}\sim\rho_\pi(s)}[V(s_{t+1})]$$

    - Soft state value (update by definition):

$$V(s_t; \theta_V) \to \mathbb{E}_{a_t\sim\pi}[Q(s_t, a_t) - \alpha \log \pi(a_t|s_t)]$$

SAC

- SAC: Soft Actor-Critic
  - Policy update: minimize KL-divergence with Q
  - Gradient:

$$
\begin{aligned}
\nabla_{\theta_\pi} J &= \nabla_{\theta_\pi} D_{\mathrm{KL}}\Big(\pi(\cdot|s_t;\theta_\pi)\Big\|\frac{\exp(Q(s_t,\cdot))}{Z(s_t)}\Big)\Big) \\
&= \nabla_{\theta_\pi}\mathbb{E}_{a_t\sim\pi}\Big[-\log\Big(\frac{\exp(Q(s_t,a_t)-\log Z(s_t))}{\pi(a_t|s_t;\theta_\pi)}\Big)\Big] \\
&= \mathbb{E}_{a_t\sim\pi}\nabla_{\theta_\pi}\log\pi(a_t|s_t;\theta_\pi)[\log\pi(a_t|s_t;\theta_\pi)-Q(s_t,a_t)] \\
&= \mathbb{E}_{a_t\sim\pi}\nabla_{\theta_\pi}\log\pi(a_t|s_t;\theta_\pi)[\log\pi(a_t|s_t;\theta_\pi)-(Q(s_t,a_t)-V(s_t))]
\end{aligned}
$$

advantage

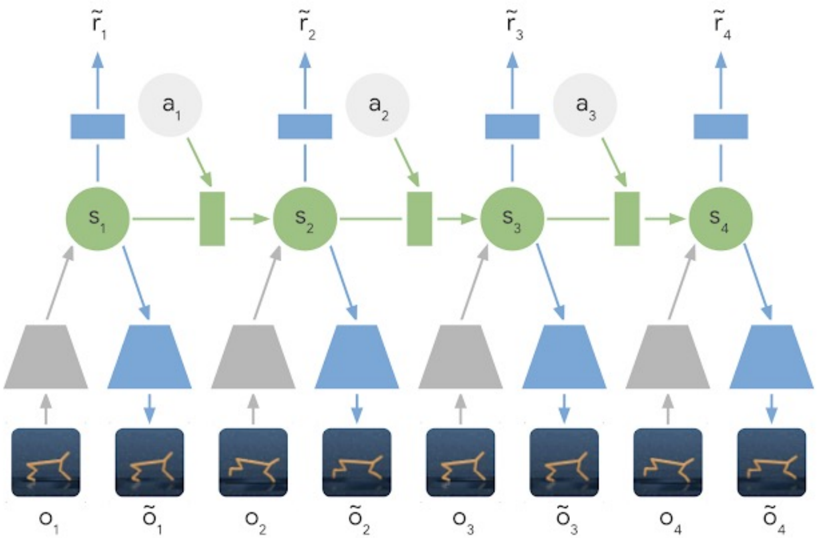
  - Proved to converge in the paper

SAC

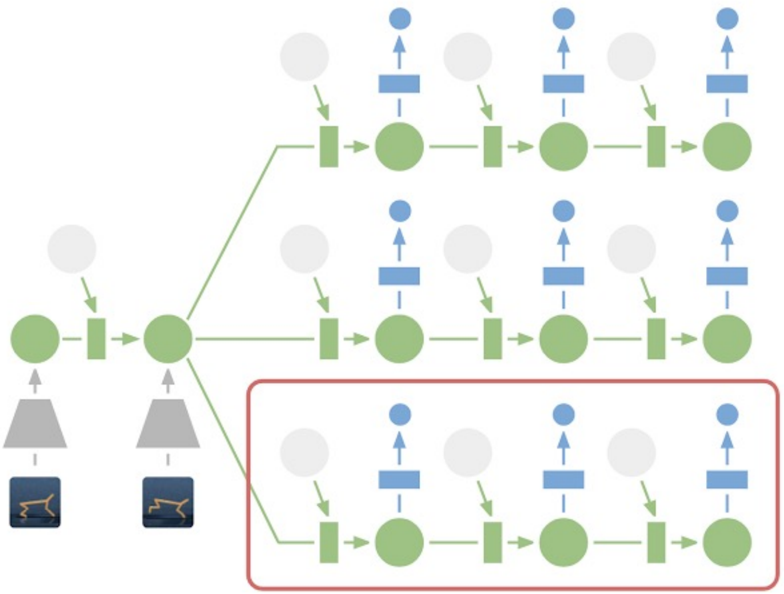
- SAC: Soft Actor-Critic
  - Tricks applied
    - Replay buffer
    - Target value network
    - Soft update

Planet

- Planet: Deep Planning Network
  - Model based
    - Learn how environment behaves
    - Planning over learned dynamics



learning latent dynamics



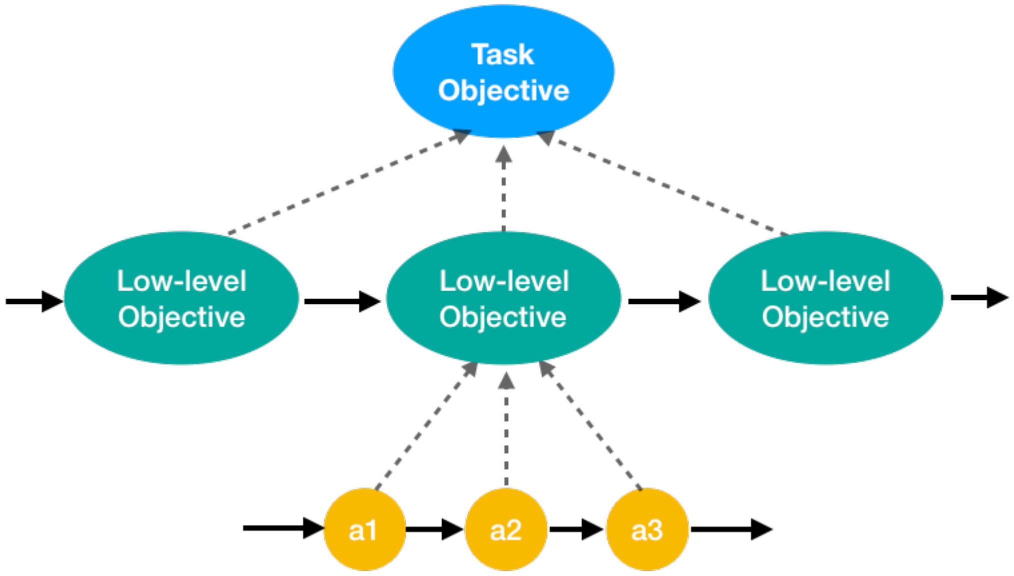
planning in latent space

Outline

- Background
- Problem and Modeling
- Algorithms for Continuous Control
- **Applications**
- Conclusion
- References

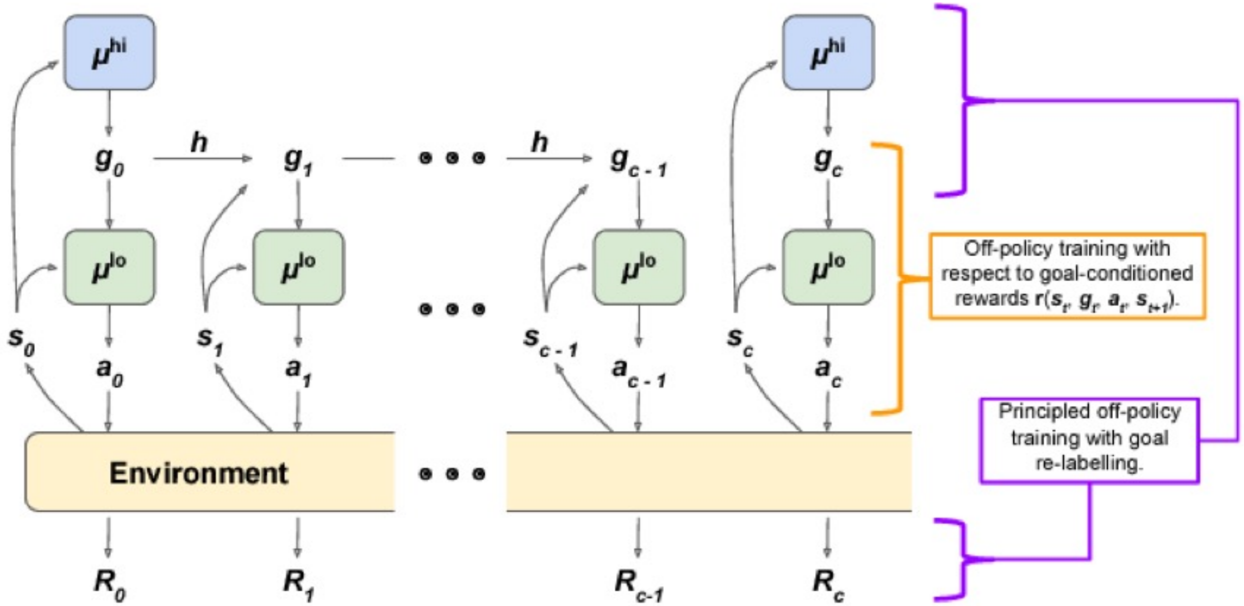
Application: HRL

- Hierarchical Reinforcement Learning
  - For complex tasks with long-term reward signal
  - Hierarchical objectives:
    - **Higher level**: construct low level objective for real objective
    - **Lower level**: take environment action towards low level objective



Application: HRL

- HIRO (Hierarchical Reinforcement Learning with Off-policy Correction)
  - Higher level: produces goal g_t indicating $s_{t+c} \to s_t + g_t$
  - Lower level: take goal accomplishment as reward

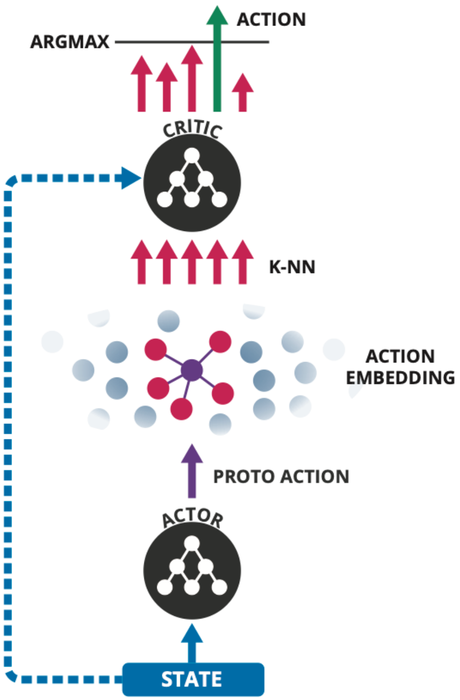


Application: HRL

- HIRO
  - Training: TD3 (DDPG variant)
  - Off-policy correction for high level training
    - Action g_t in high level replay memory no longer take to s_{t+1}
    - To reuse memory, HIRO modify goal to preserve low level action

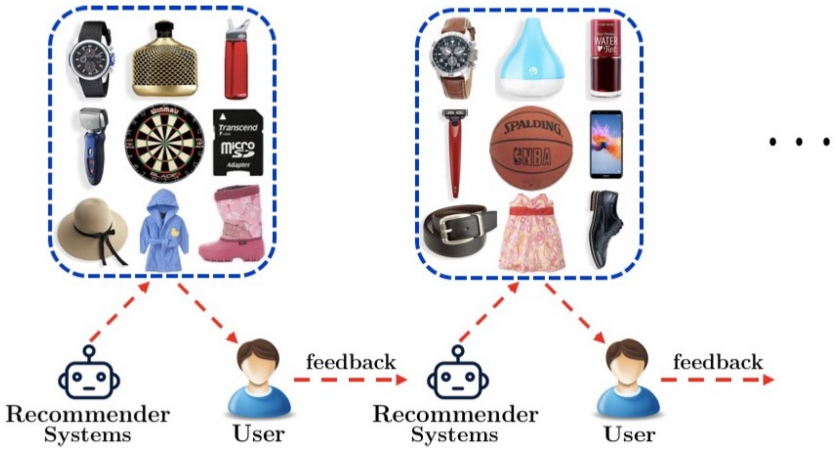
Application: Recommendation

- Prerequisites
  - Deep RL in Large Discrete Action Spaces
    - Action embedding: continuous action
    - Full policy:
      - Take proto action: generate embedding
      - Take k nearest neighbors
      - Take best of k with largest action value
    - Training: DDPG
      - Policy gradient from proto action
      - Critic update from actual action



Application: Recommendation

- DeepPage
  - Page-wise recommendation on E-commerce platforms (e.g. JD)



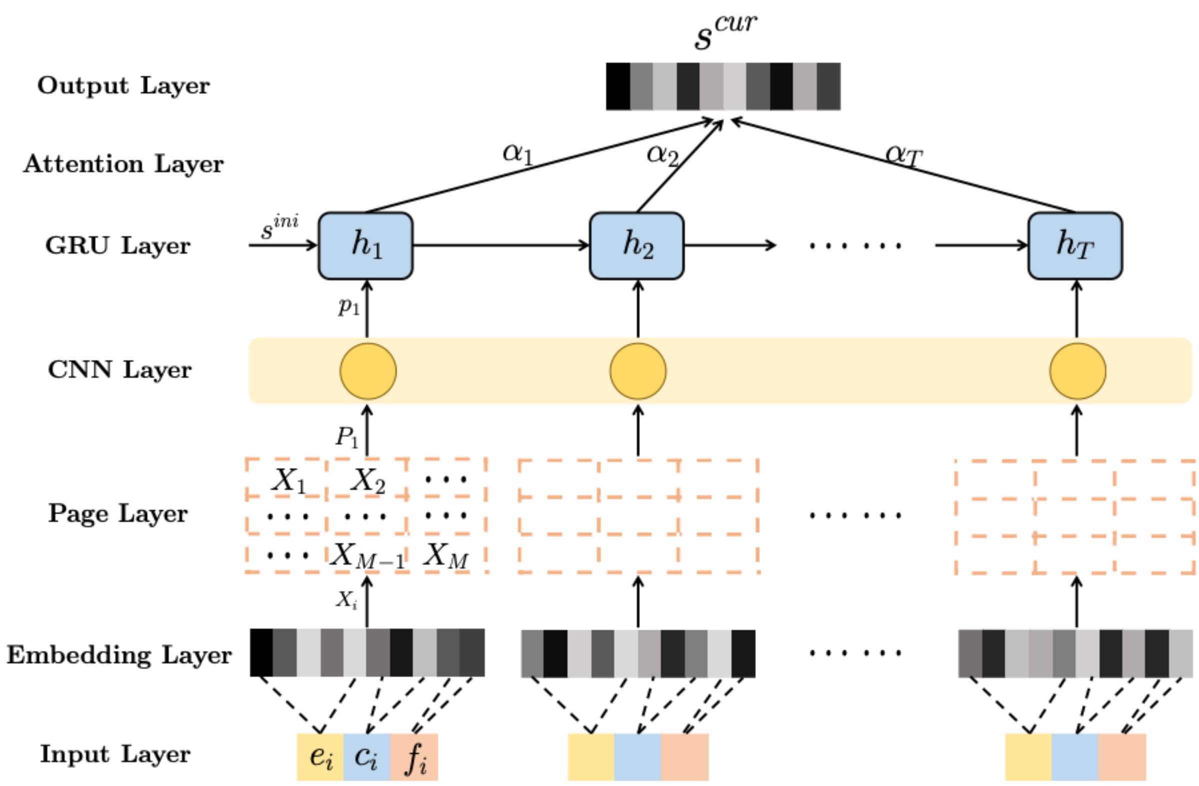
  - MDP definition
    - State: user preference (based on browsing history)
    - Action: a page of items
    - Reward: user feedback (skip, click, purchase)

Application: Recommendation

- DeepPage
  - Main concerns:
    - How to capture user preferences: state representation
    - How to generate recommendation: deep RL algorithms
  - Challenges applying deep RL directly
    - Large and dynamic action space
    - Computational cost selecting optimal action (a page of items)
  - Solution: **continuous action** (item embedding)

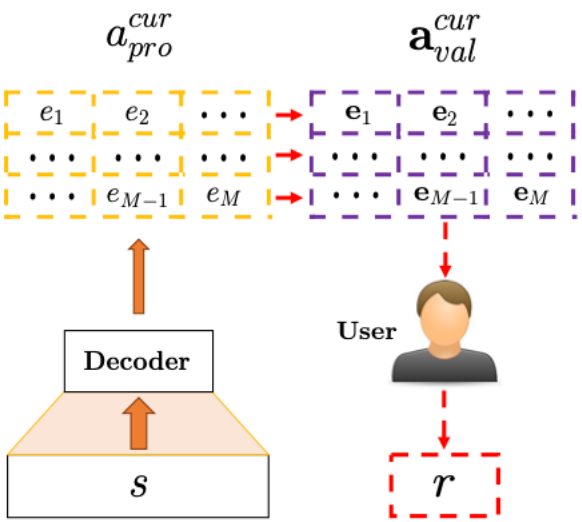
Application: Recommendation

- DeepPage
  - State representation



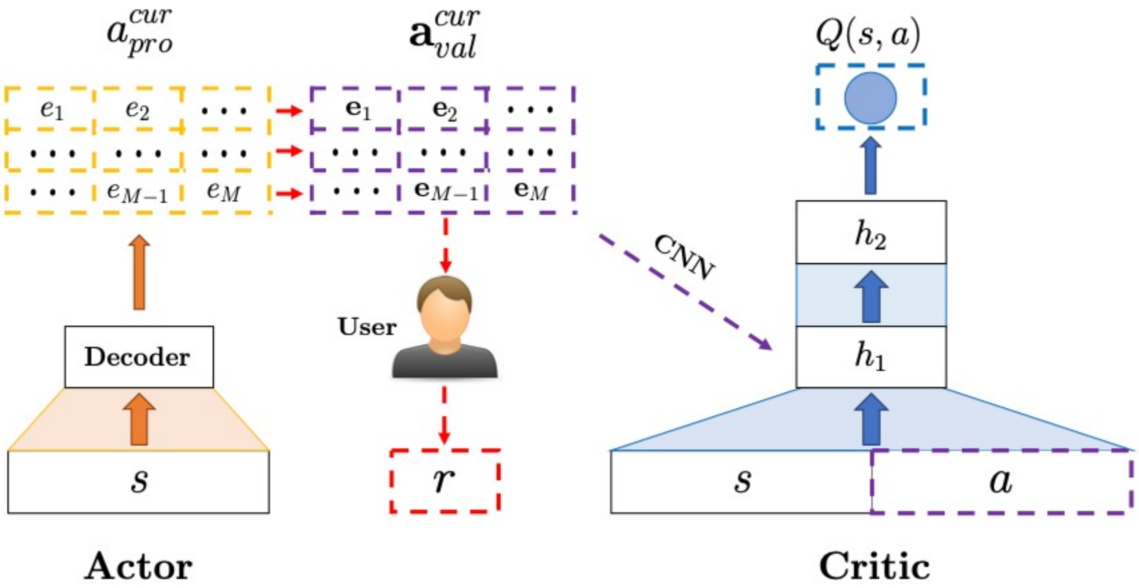
Application: Recommendation

- DeepPage
  - Action generation
    - Decode action (desired embeddings) using deconvolutional layer
    - Embeddings may not correspond to real item
      - Select most similar item without replacement



Application: Recommendation

- DeepPage
  - Critic (action value estimation)

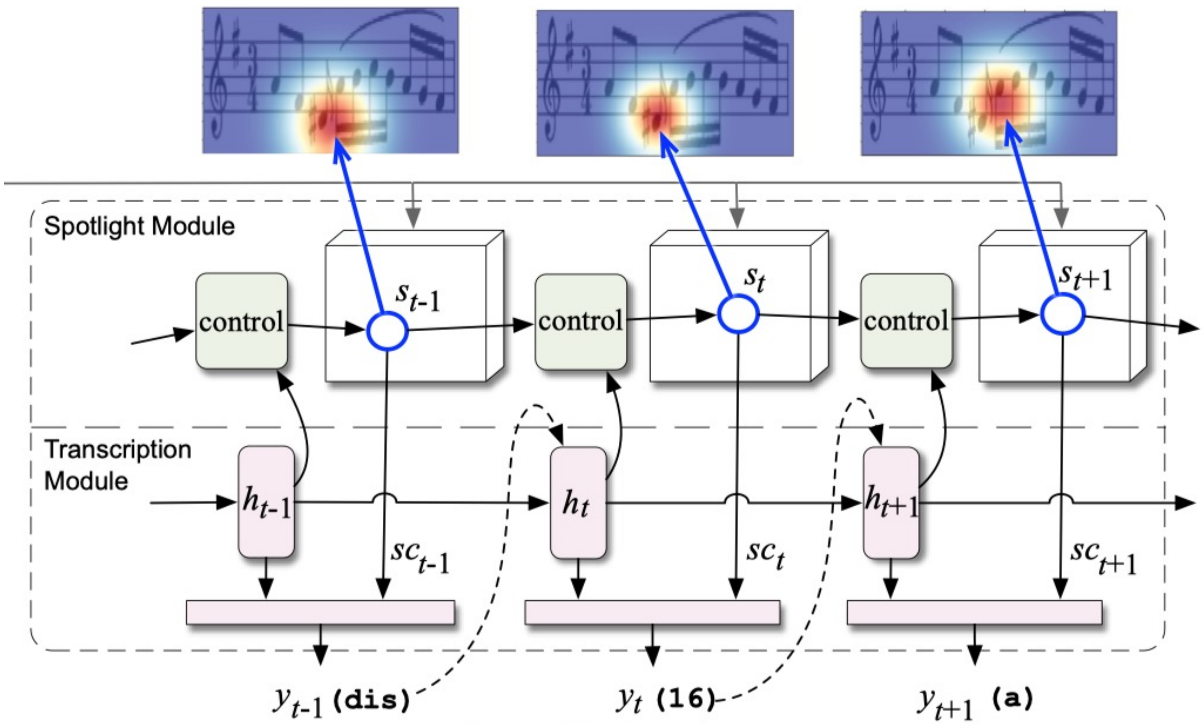


Application: Recommendation

- DeepPage
  - Training
    - Online training: collected real-time data samples
    - Offline training: off-policy data samples

Application: CV

- Spotlight
  - Spotlight mechanism



Application: CV

- Spotlight
  - Spotlight control
    - Formulation:
      - State: image, along with action and output history

$$s_t = \{I, a_1, \ldots, a_{t-1}, y_1, \ldots, y_{t-1}\}$$

      - Action: spotlight center and radius (continuous)

$$a_t = \{\mu_t, \boldsymbol{\Sigma}_t\}$$

      - Reward: NLL at current step / sequence-wise metric score
    - Enables fine-grained control strategy

Outline

Conclusion

- Conclusions
  - Provides better interpretability inside black-box neural networks
  - Good for huge/dynamic output space
  - Ability to fine-tune control strategy

Q&A

Reference

- https://lilianweng.github.io/lil-log/2018/02/19/a-long-peek-into-reinforcement-learning.html
- https://lilianweng.github.io/lil-log/2018/04/08/policy-gradient-algorithms.html
- Continuous control with deep reinforcement learning
- Soft Actor Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor
- Learning Latent Dynamics for Planning from Pixels (planet)
- Data-Efficient Hierarchical Reinforcement Learning
- Deep reinforcement learning for page-wise recommendations