Meta Reinforcement Learning

Yun Hua

Outline

- Motivation
- Basic Formulation
- Main Works：
  - 1. Optimizing Model Weights for Meta-learning(MAML)
  - 2. Meta-learning Hyperparameters(Meta Gradient RL)
  - 3. Meta-learning the Exploration Strategies(MAESN)
  - 4. Meta-Learning the Task Embedding(RL^2)
- Shortages
- More works and summary

Motivation

Regular RL: learn policy for single task

$$\theta^\star = \arg\max_\theta E_{\pi_\theta(\tau)}[R(\tau)]$$

$$= f_{\mathrm{RL}}(\mathcal{M})$$

MDP

Meta-training / Outer loop

Adaptation / Inner loop

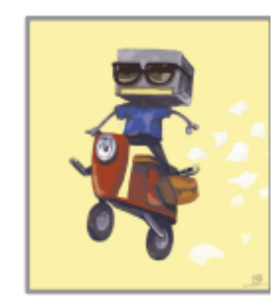
Meta-RL: learn adaptation rule

$$\theta^\star = \arg\max_\theta \sum_{i=1}^{n} E_{\pi_{\phi_i}(\tau)}[R(\tau)]$$

$$\text{where } \phi_i = f_\theta(\mathcal{M}_i)$$

MDP for task i

$\mathcal{M}_1$ $\mathcal{M}_2$ $\mathcal{M}_3$

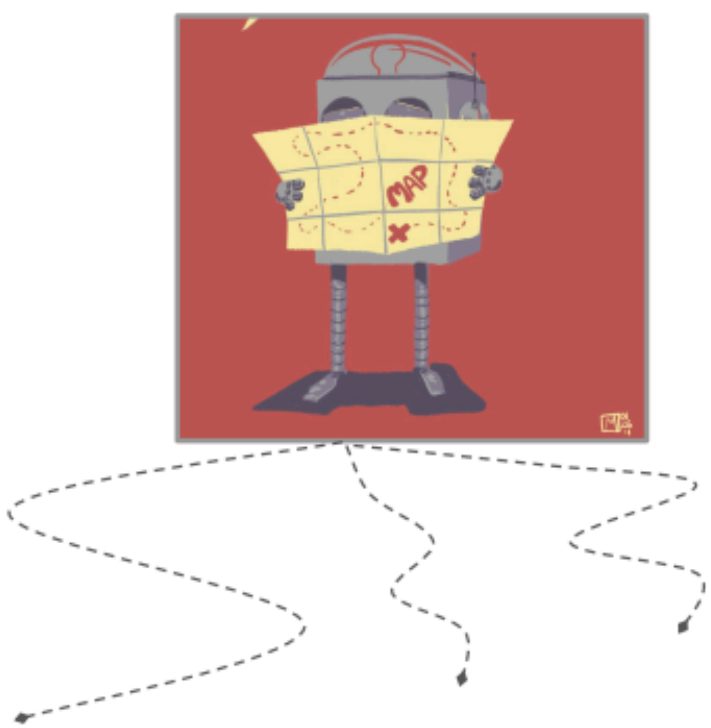


$\mathcal{M}_{test}$

Basic Formulation

$$\theta^\star = \arg\max_\theta \sum_{i=1}^{n} E_{\pi_{\phi_i}(\tau)}[R(\tau)]$$

$$\text{where } \phi_i = f_\theta(\mathcal{M}_i)$$



What should the adaptation procedure do?

- **Explore**: Collect the most informative data
- **Adapt**: Use that data to obtain the optimal policy

Basic Formulation

while training:

1. sample task i, collect data $\mathcal{D}_i$
2. adapt policy by computing $\phi_i = f(\theta, \mathcal{D}_i)$
3. collect data $\mathcal{D}'_i$ with adapted policy π_{ϕ_i}
4. update θ according to $\mathcal{L}(\mathcal{D}'_i, \phi_i)$

Can do more than one round of adaptation

In practice, compute update across a batch of tasks

Different algorithms:

- Choice of function f
- Choice of loss function L

MAML

Algorithm 3 MAML for Reinforcement Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
randomly initialize θ
while not done **do**
  Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
  for all $\mathcal{T}_i$ **do**
    Sample K trajectories $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{a}_1, ...\mathbf{x}_H)\}$ using f_θ in $\mathcal{T}_i$
    Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using $\mathcal{D}$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation 4
    Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
    Sample trajectories $\mathcal{D}_i' = \{(\mathbf{x}_1, \mathbf{a}_1, ...\mathbf{x}_H)\}$ using $f_{\theta_i'}$ in $\mathcal{T}_i$
  end for
  Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$ using each $\mathcal{D}_i'$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation 4
end while

$$\mathcal{L}_{\mathcal{T}_i}(f_\phi) = -\mathbb{E}_{\mathbf{x}_t, \mathbf{a}_t \sim f_\phi, q_{\mathcal{T}_i}} \left[\sum_{t=1}^{H} R_i(\mathbf{x}_t, \mathbf{a}_t) \right]. \quad (4)$$

Key point: using the prior infos to learning an Efficient initialization parameter for the test tasks.

Meta Gradient RL

- The core update function is:

$$\theta' = \theta + f(\tau, \theta, \eta) \qquad \eta = (\lambda, \gamma)$$

Experience trace **Parameter** **Meta Parameter**

$$G_\eta^{(n)}(\tau_t) = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n v_\theta(s_{t+n}) \qquad \text{; n-step return}$$

$$G_\eta^{\lambda}(\tau_t) = (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_\eta^{(n)} \qquad \text{; λ-return, mixture of n-step returns}$$

Meta Gradient RL

- The work process of the algorithm:
  - 1. The algorithm starts with parameters θ , and applies the update function to the first sample(s) τ , resulting in new parameters θ'
  - 2. The gradient $d\theta' / d\eta$ of these update indicates how the meta-parameters affected these new parameters.
  - 3. Measuring the performance of the new parameters θ' on a subsequent, independent sample τ' , utilizing a differentiable meta-objective $J'(\tau', \theta', \eta')$

Meta-Reinforcement Learning of Structured Exploration Strategies

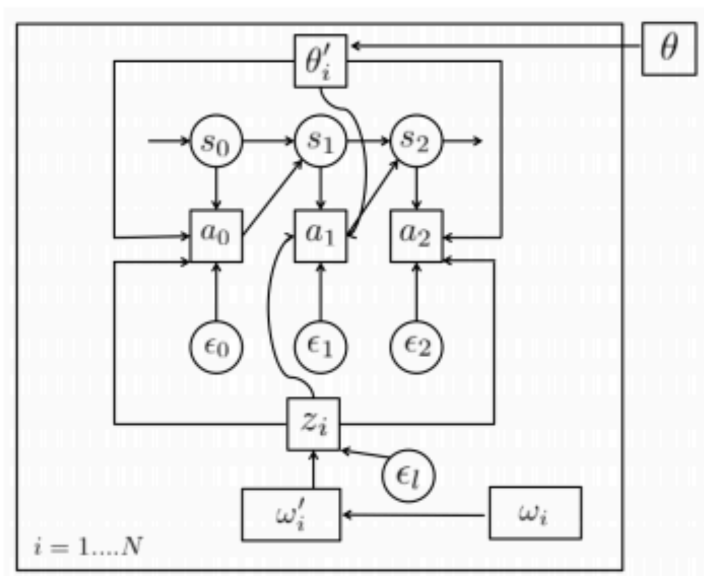


Figure 1: Computation graph for MAESN. Meta-learn pre-update latent parameters ω_i, and policy parameters θ, such that after a gradient step, the post-update latent parameters ω'_i, policy parameters θ', are optimal for the task. The sampling procedure introduces time correlated noise.

Algorithm 1 MAESN meta-RL algorithm

Initialize variational parameters ω_i for each training task τ_i
for iteration $k \in \{1, \dots, K\}$ **do**
    Sample a batch of N training tasks from $p(\tau)$
    for task $\tau_i \in \{1, \dots, N\}$ **do**
        Gather data using the latent conditioned policy θ, (ω_i)
        Compute inner policy gradient on variational parameters via Equation (4) (optionally (5))
    end for
    Compute meta update on both latents and policy parameters by optimizing (3) with TRPO
end for

$$\omega'_i = \omega_i + \alpha_\omega \circ \nabla_{\omega_i} E_{\substack{a_t \sim \pi(a_t|s_t;\theta,z_i) \\ z_i \sim q_{\omega_i}(.)}} \left[\sum_t R_i(s_t) \right] \tag{3}$$

$$\theta'_i = \theta + \alpha_\theta \circ \nabla_\theta E_{\substack{a_t \sim \pi(a_t|s_t;\theta,z_i) \\ z_i \sim q_{\omega_i}(.)}} \left[\sum_t R_i(s_t) \right] \tag{4}$$

Key point: using a prior info to learn the distribution of the exploration.

RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning

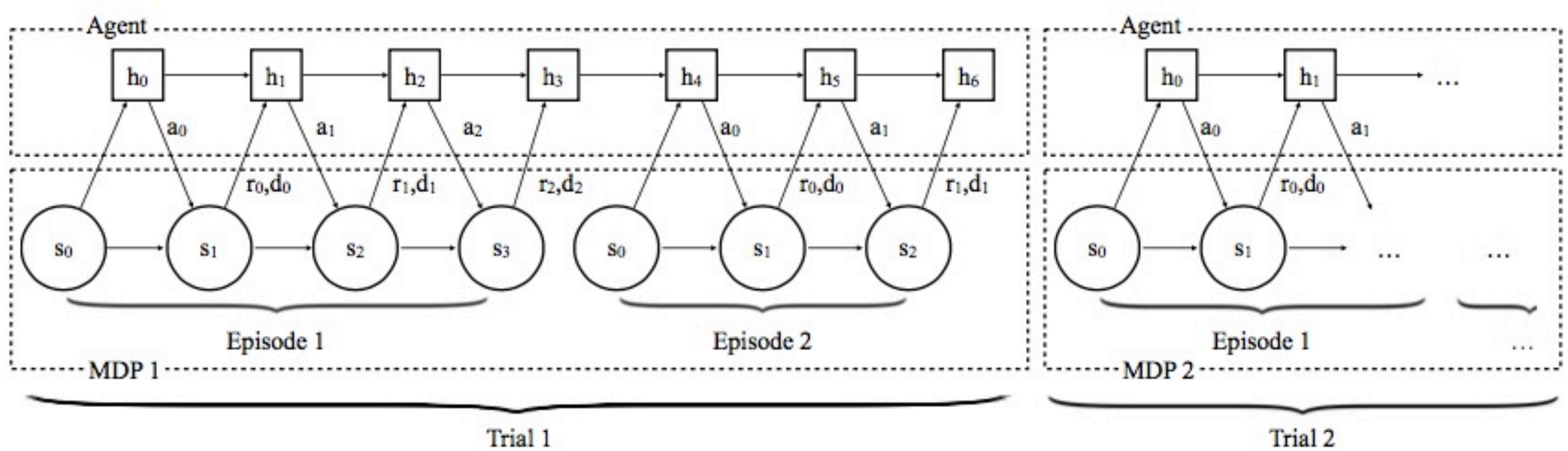


Figure 1: Procedure of agent-environment interaction

Key points:

1. **Using the prior trajectories to learn a task embedding(*meta parameter* θ)**
2. **Adapt the task embedding to the new tasks**

Shortage of recent meta reinforcement learning

- In the sparse reward environment, these basic methods always doesn’t works well.
- In adaptation, the test tasks always in the same distribution with the training tasks.

Summary

- Every thing can be meta-learned !!!!!(trajectories, hyperparamter, gradient, loss function, return function, initialization, reward, advantage, etc)

Further works

- Meta learning in Gradient:
    - Evolved policy gradients.(NIPS2018)
- Meta learning in loss function:
    - High-Dimensional Continuous Control Using Generalized Advantage Estimation(ICLR2016)
- Meta Learning in Reward:
    - Learning a Prior over Intent via Meta-Inverse Reinforcement Learning
- Meta Learning in Advantage:
    - NoRML: No-Reward Meta Learning(AAMAS 2019)