Meta-Learning

Outline

- Background
- Formulation
- MAML
- MBMPO
- Meta-Critic-Networks
- RL^2
- Conclusion

Outline

- Background
- Formulation
- MAML
- MBMPO
- Meta-Critic-Networks
- RL^2
- Conclusion

Background

- Machine VS Human
  - Machine have surpassed humans at many tasks.
  - However, Machine generally need far more data to reach the same level.

- Human can learn fast with a large amount of prior knowledge encoded in their brains and DNA!
- These prior knowledge are called "meta knowledge"!
- In fact, there is NO UNIFIED DIFINITION about WHAT IS META......

Background

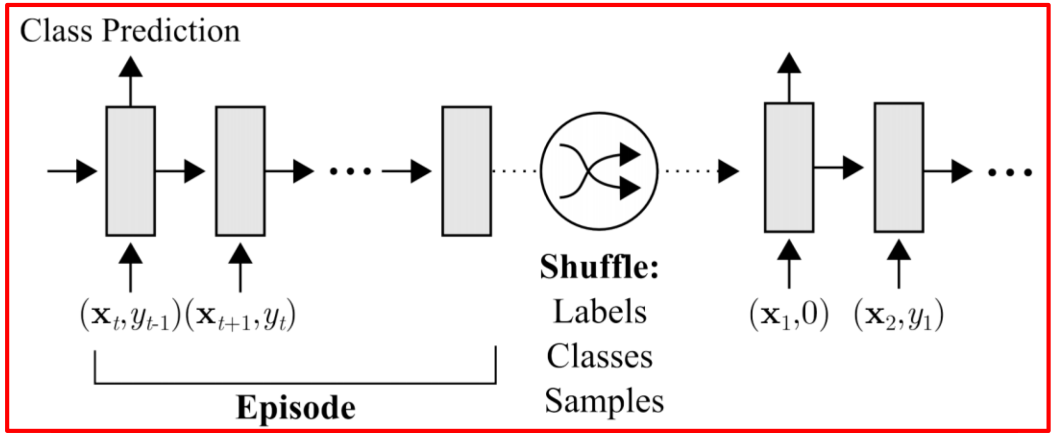
- Meta-Learning——Learning to Learn.
  - When you learn to play FPS, learn to play TCG, learn to play VR game, learn......
    ------>You will know how to learn any game in a short time.
  - That is, You learned how to learn!
- The goal of meta-learning: Train a model on a variety of learning tasks, and solve new learning tasks using few training samples —— that is, few-shot learning.
- Requrement: Meta-learning should be general to the task and the form of computation.(similar to VIN, i.e. learn to plan)

Background

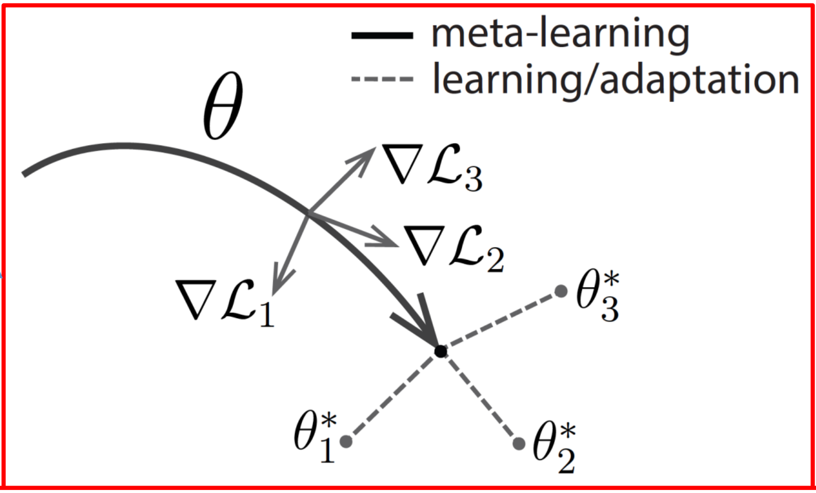
- This process can be viewed as building an internal representation that is broadly suitable for many tasks.
- Furthermore, it also means maximizing the sensitivity of the loss function of new tasks.

Background

- Three categories of methods:
  - Recurrent Models
  - Metric Learning
  - Learning Optimizers
    - Parameter Initialization: learn the initialization of network and fine-tune(微调) on the new task, e.g. MAML.



Meta-Learning with Memory-Augmented Neural Networks



Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks

Outline

- Background
- Formulation
- MAML
- MBMPO
- Meta-Critic-Networks
- RL^2
- Conclusion

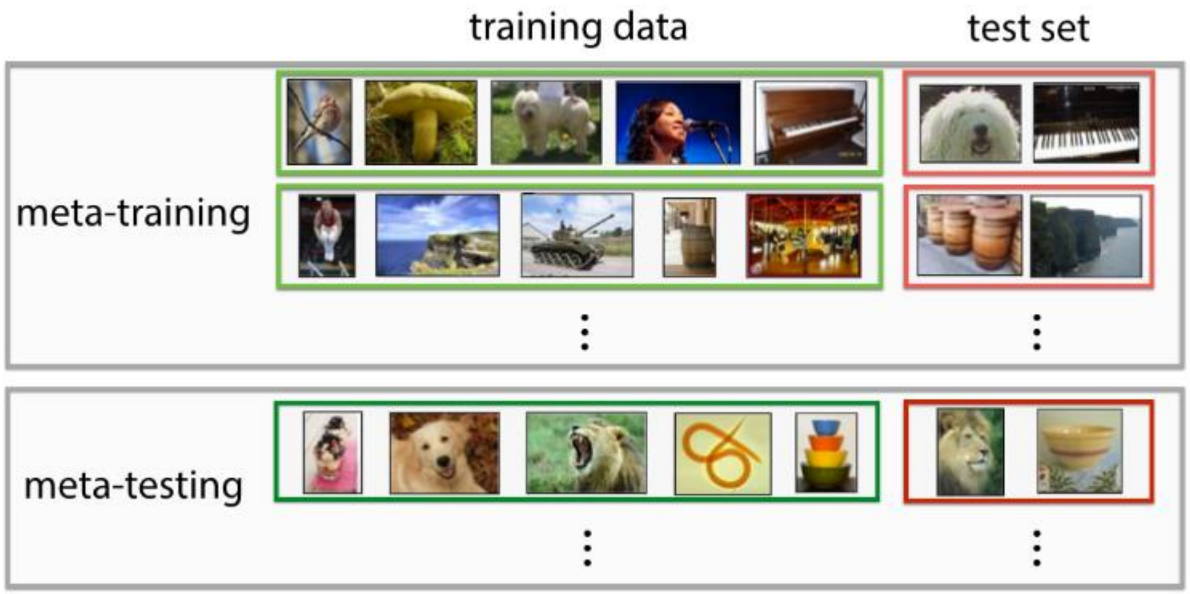
Fomulation

- Task: $\boxed{\mathcal{T} = \{\mathcal{L}(\mathbf{x}_1, \mathbf{a}_1, \ldots, \mathbf{x}_H, \mathbf{a}_H), q(\mathbf{x}_1), q(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{a}_t), H\}}$
- Loss: $\mathcal{L}(\mathbf{x}_1, \mathbf{a}_1, \ldots, \mathbf{x}_H, \mathbf{a}_H)$ (depend on specified task)
- Init Observation: $q(\mathbf{x}_1)$
- Transition Distribution: $q(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{a}_t)$
- Episode Length: H (H=1 for supervised learning)
- Task Distribution: $\boxed{p(\mathcal{T})}$

- Goal: Find a model to be able to adapt to the task distribution $p(\mathcal{T})$

Fomulation

- How does meta-learning work to achieve this goal?
  - Meta-training and meta-testing.
  - Training set and test set.



Outline

- Background
- Formulation
- MAML
- MBMPO
- Meta-Critic-Networks
- RL^2
- Conclusion

MAML

- Model-Agnostic Meta-Learning: a method to learn an initialization.
- Algorithm:

Meta Training

- ①Initialization: meta-parameter θ
- ②Sample Tasks: $\mathcal{T}_i \sim p(\mathcal{T})$
- ③Adaptation: compute adapted-parameter θ_i' using meta-parameter θ and training set, i.e. $\boxed{\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)}$
- ④Optimization: update θ using θ_i' and test set w.r.t. $\boxed{\theta}$, i.e.

$$\boxed{\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})}$$

why????

MAML

- Procedure ④ is an optimization problem:

$$\boxed{\min_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'}) = \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}\left(f_{\theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)}\right)}$$

- That is to say, we want to find an optimal θ to make the performance of fine-tuning best! (i.e. maximize the sensitivity)
- However, the update $\boxed{\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})}$ involves a gradient through a gradient.
  - It requires additional computation(of course supported by some DL libraries)
  - Instead, we can only use a first-order approximatiom(FOMAML)

MAML

Algorithm 1 Model-Agnostic Meta-Learning

Require: $p(\mathcal{T})$: distribution over tasks

Require: α, β: step size hyperparameters

1: randomly initialize θ

2: **while** not done **do**

3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$

4: **for all** $\mathcal{T}_i$ **do**

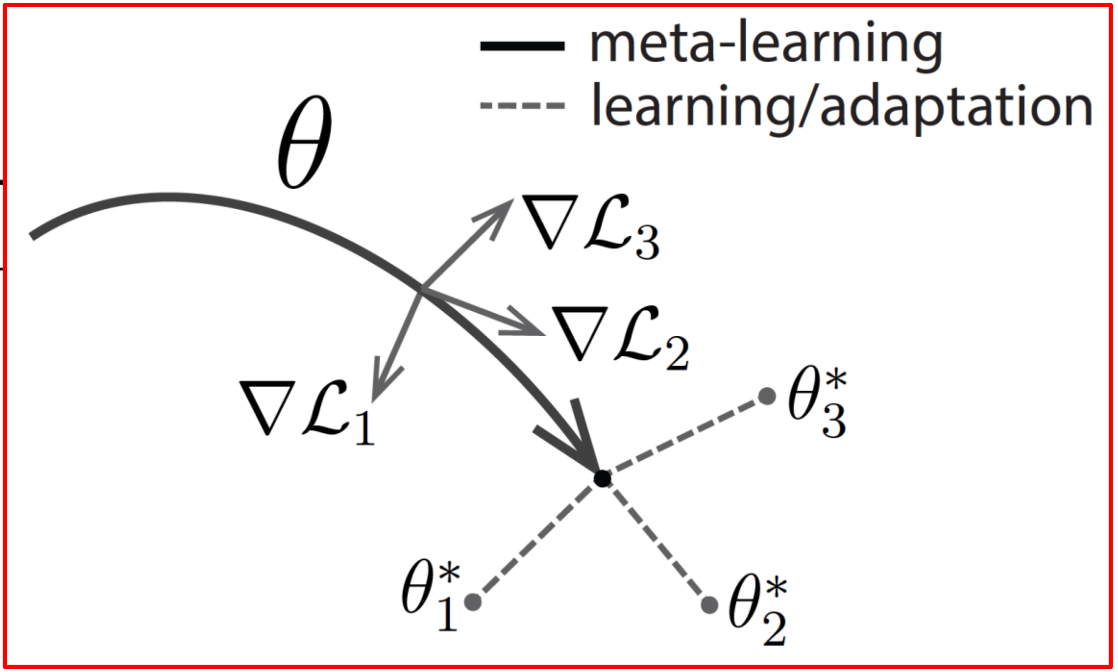
5: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to K examples

6: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$

7: **end for**

8: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$

9: **end while**



MAML

- With Algorithm 1, we can easily apply it to some different domains~~
- Supervised Learning
  - Regression
  - Classification
  - ……
- Reinforcement Learning
  - Policy Gradient
  - Actor Critic
  - ……

MAML

Algorithm 2 MAML for Few-Shot Supervised Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters

1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Sample K datapoints $\mathcal{D} = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from $\mathcal{T}_i$
6: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using $\mathcal{D}$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation (2) or (3)
7: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
8: Sample datapoints $\mathcal{D}_i' = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from $\mathcal{T}_i$ for the meta-update
9: **end for**
10: 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 2 or 3
11: **end while**

MSE

$$\mathcal{L}_{\mathcal{T}_i}(f_\phi) = \sum_{\mathbf{x}^{(j)}, \mathbf{y}^{(j)} \sim \mathcal{T}_i} \| f_\phi(\mathbf{x}^{(j)}) - \mathbf{y}^{(j)} \|_2^2, \qquad (2)$$

Cross-Entropy Loss

$$\mathcal{L}_{\mathcal{T}_i}(f_\phi) = \sum_{\mathbf{x}^{(j)}, \mathbf{y}^{(j)} \sim \mathcal{T}_i} \mathbf{y}^{(j)} \log f_\phi(\mathbf{x}^{(j)}) + (1 - \mathbf{y}^{(j)}) \log(1 - f_\phi(\mathbf{x}^{(j)})) \qquad (3)$$

MAML

Algorithm 3 MAML for Reinforcement Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Sample K trajectories $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{a}_1, ...\mathbf{x}_H)\}$ using $\boxed{f_\theta}$ in $\mathcal{T}_i$
6: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using $\mathcal{D}$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation 4
7: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
8: Sample trajectories $\mathcal{D}_i' = \{(\mathbf{x}_1, \mathbf{a}_1, ...\mathbf{x}_H)\}$ using $f_{\theta_i'}$ in $\mathcal{T}_i$
9: **end for**
10: 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
11: **end while**

$$\boxed{\pi_\theta(a \mid x_t)}$$

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

MAML

- Benefits of MAML:
- 1.It is a simple framework.
- 2.It can be combined with any model representation / architecture and any differentiable objective. (MAML merely produces a parameter initialization)
- 3.Adaptation can be performed with any amount of data and any number of gradient step.

- As you can see, MAML is quite a common framework!

Outline

- Background
- Formulation
- MAML
- MBMPO
- Meta-Critic-Networks
- RL^2
- Conclusion

MBMPO

- Model-Based Meta-Policy Optimization.
- Key Point:
  - Model-based approach with ensemble.
  - Meta-policy for fast adaptation.

- Benefits:
  - Lower sample complexity(model-based)
  - Alleviate model bias to achieve better performance(ensemble)
  - Faster convergence(meta-policy)

MBMPO

- Model Learning:

$$\min_{\phi_k} \frac{1}{|\mathcal{D}_k|} \sum_{(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}) \in \mathcal{D}_k} \|\boldsymbol{s}_{t+1} - \underbrace{\boxed{\hat{f}_{\phi_k}(\boldsymbol{s}_t, \boldsymbol{a}_t)}}_{\text{models}}\|_2^2$$

- Meta-Reinforcement Learning:

$$\max_{\boldsymbol{\theta}} \underbrace{\boxed{\frac{1}{K} \sum_{k=0}^{K} J_k(\boldsymbol{\theta}'_k)}}_{\text{meta-objective}} \quad \text{s.t.:} \quad \boldsymbol{\theta}'_k = \boldsymbol{\theta} + \alpha \nabla_{\boldsymbol{\theta}} \underbrace{\boxed{J_k(\boldsymbol{\theta})}}_{\text{adaptation objective}}$$

- with the objective function: $J_k(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{a}_t \sim \pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t|\boldsymbol{s}_t)} \left[\sum_{t=0}^{H-1} r(\boldsymbol{s}_t, \boldsymbol{a}_t) \middle| \boldsymbol{s}_{t+1} = \hat{f}_{\phi_k}(\boldsymbol{s}_t, \boldsymbol{a}_t) \right]$

MBMPO

Algorithm 1 MB-MPO

Require: Inner and outer step size α, β

Initialize the policy $\pi_{\boldsymbol{\theta}}$, the models $\hat{f}_{\phi_1}, \hat{f}_{\phi_2}, ..., \hat{f}_{\phi_K}$ and $\mathcal{D} \leftarrow \emptyset$

repeat

  Sample trajectories from the real environment with the adapted policies $\boxed{\pi_{\boldsymbol{\theta}'_1}, ..., \pi_{\boldsymbol{\theta}'_K}}$. Add them to $\mathcal{D}$.

  Train all models using $\mathcal{D}$.

  for all models $\hat{f}_{\phi_k}$ **do**

    Sample imaginary trajectories $\mathcal{T}_k$ from $\hat{f}_{\phi_k}$ using $\boxed{\pi_{\boldsymbol{\theta}}}$

    Compute adapted parameters $\boxed{\boldsymbol{\theta}'_k = \boldsymbol{\theta} + \alpha \nabla_{\boldsymbol{\theta}} J_k(\boldsymbol{\theta})}$ using trajectories $\mathcal{T}_k$

    Sample imaginary trajectories $\mathcal{T}'_k$ from $\hat{f}_{\phi_k}$ using the adapted policy $\boxed{\pi_{\boldsymbol{\theta}'_k}}$

  end for

  Update $\boxed{\boldsymbol{\theta} \rightarrow \boldsymbol{\theta} - \beta \frac{1}{K} \sum_k \nabla_{\boldsymbol{\theta}} J_k(\boldsymbol{\theta}'_k)}$ using the trajectories $\mathcal{T}'_k$

until the policy performs well in the real environment

return Optimal pre-update parameters $\boldsymbol{\theta}^*$

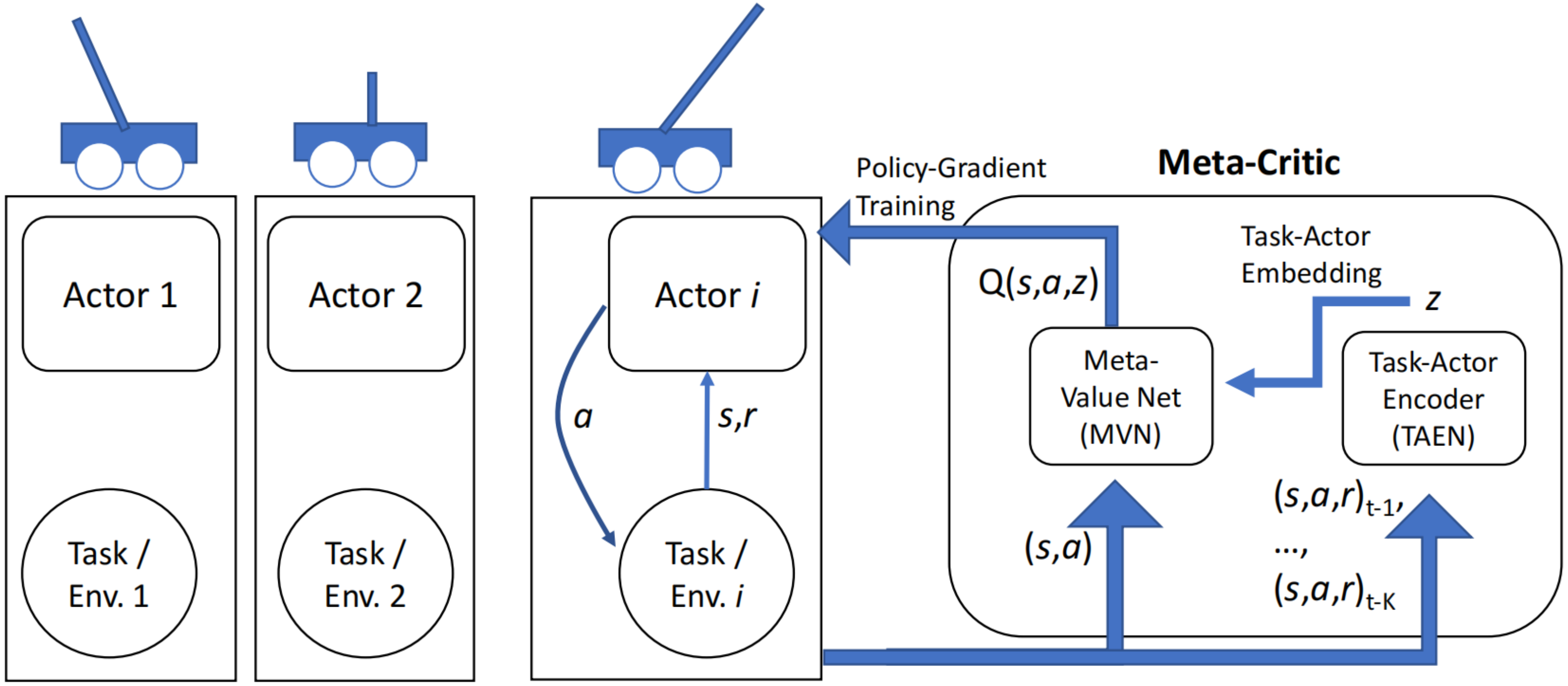
PG method, e.g.TRPO,VPG......

Outline

- Background
- Formulation
- MAML
- MBMPO
- Meta-Critic-Networks
- RL^2
- Conclusion

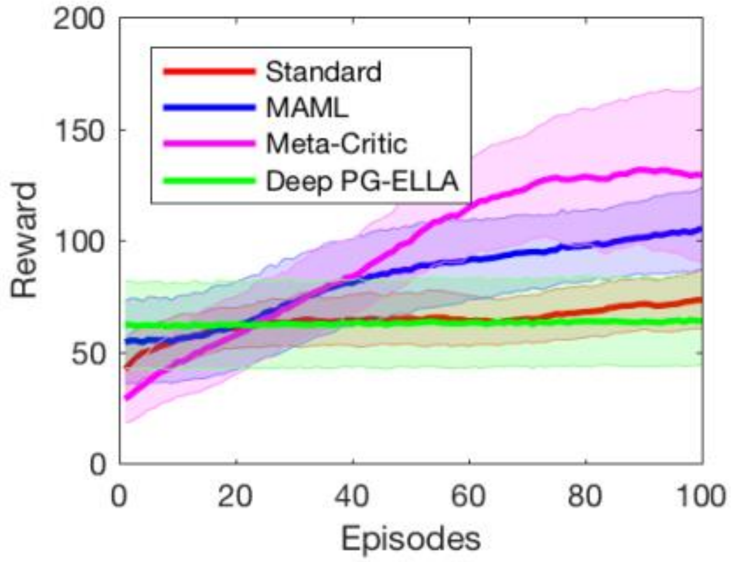
Meta-Critic-Networks

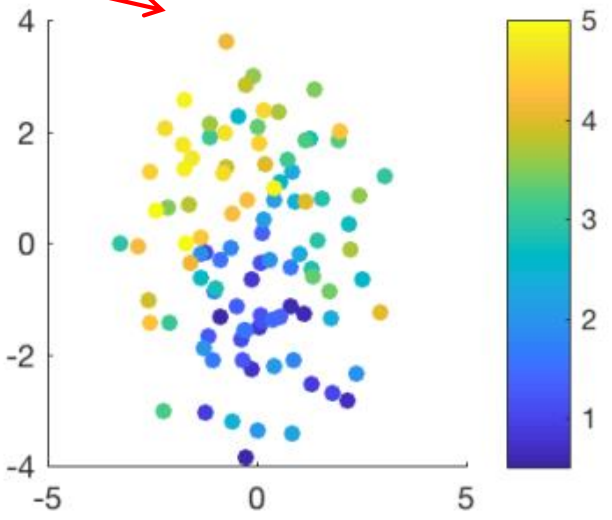
- Idea: Instead of learning a policy, Meta-Critic-Networks learn a Meta Value Network(MVN) and a Task-Actor Encoder Network(TAEN).
- In their words, that is a “核心价值观”

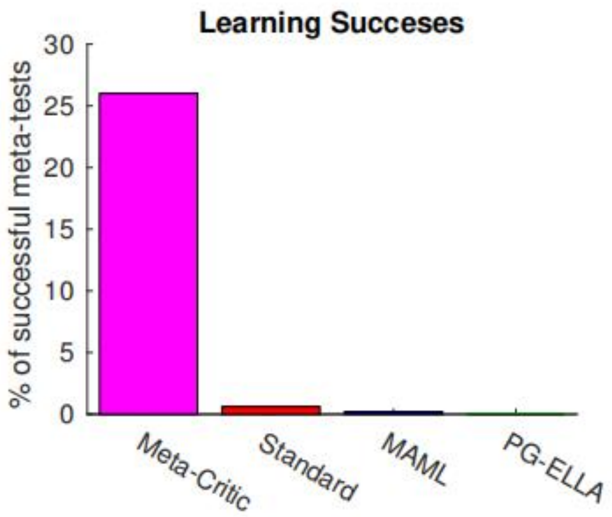


Meta-Critic-Networks

- Experiment: Cartpole
- What does TAEN encode?







Meta-Critic-Networks

Algorithm 1: Meta-Learning Stage

```
Input: Task generator T
Output: Trained task and value net
Init: task and value net;
for episode = 1 to max episode do
    Generate M tasks from T;
    Init M policy nets (actors);
    for step = 1 to max steps do
        Sample mini-batch of tasks;
        foreach task in mini-batch do
            Sample training data from task;
            Train task-specific actor;
        end
        Train value network;
        Train task network;
    end
end
```

Meta-Critic-Networks

Algorithm 2: Meta-Testing Stage

Input: An unseen task
Input: Trained task and value nets
Output: Trained policy network

Init: one policy network (actor);
for *step = 1 to max step* **do**
    Sample train data from task;
    Train actor;
end

Outline

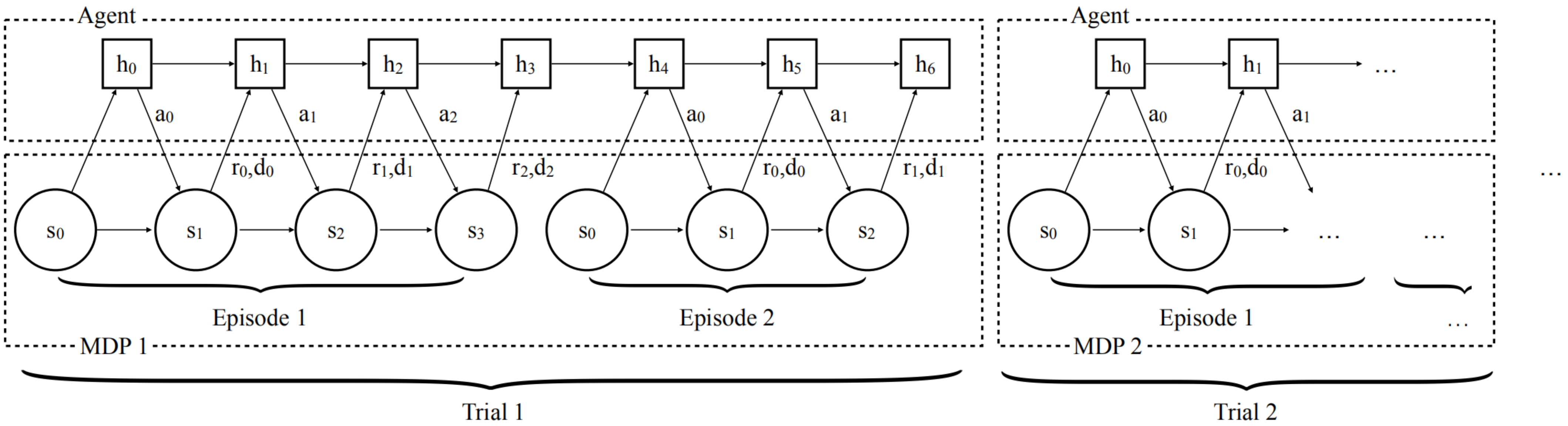
- Background
- Formulation
- MAML
- MBMPO
- Meta-Critic-Networks
- RL^2
- Conclusion

RL^2

- RL^2: Casts learning an RL algorithm as a reinforcement learning problem
- Idea:
  - View the learning process of the agent itself as an objective.
  - Structure the agent as a RNN, which receives past rewards, actions, and termination flags as inputs in addition to the normally received observations.

RL^2

- Illustration



- When MDP changes, the agent must act differently according to its belief over which MDP it is currently in. Hence, the agent is forced to integrate all the information and adapt its strategy continually.

Outline

- Background
- Formulation
- MAML
- MBMPO
- Meta-Critic-Networks
- RL^2
- Conclusion

Conclusion

- What is meta? What is meta-learning?
  - No unified difinition......
  - However, we still can make progress on it.
- MAML is a kind of common framwork to produce a parameter initialization.
- Meta-Critic-Network produce a novel idea for meta-learning.
- RL^2 use RL to learn a RL progcess.
- ....

Reference

- https://bair.berkeley.edu/blog/2017/07/18/learning-to-learn/
- Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks
- Model-Based Reinforcement Learning via Meta-Policy Optimization
- On First-Order Meta-Learning Algorithms
- Learning to learn: Meta-critic networks for sample efficient learning
- RL^2: FAST REINFORCEMENT LEARNING VIA SLOW REINFORCEMENT LEARNING