Meta Learning for RL

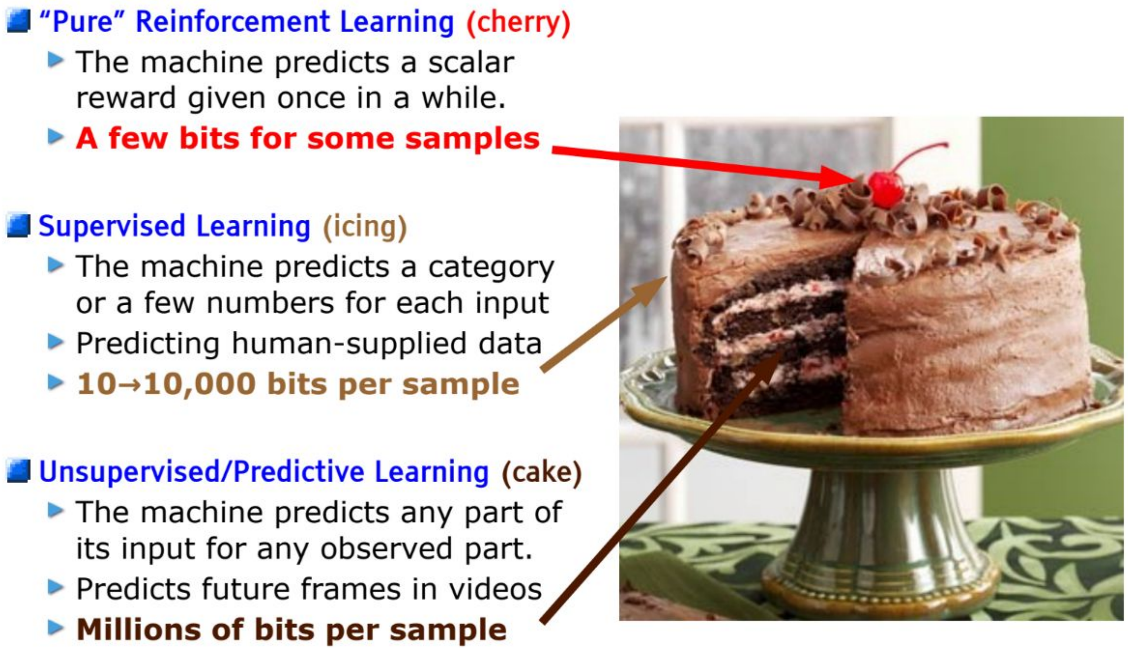
Yunqiu Xu

Contents

Introduction

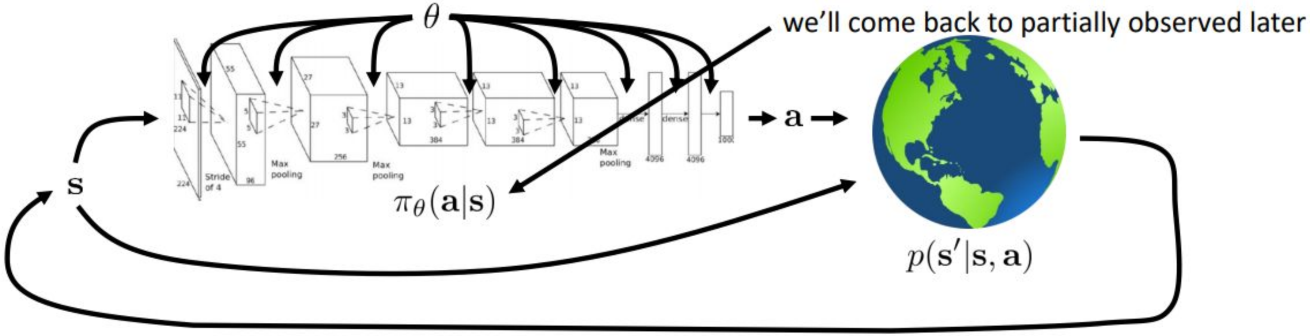
Reinforcement Learning

- Agent is not presented with target outputs, but is given a reward signal, which it aims to maximize



Yann LeCun, NIPS 2016

Reinforcement Learning: Goal



$$\underbrace{p_\theta(\mathbf{s}_1,\mathbf{a}_1,\ldots,\mathbf{s}_T,\mathbf{a}_T)}_{\pi_\theta(\tau)} = p(\mathbf{s}_1)\prod_{t=1}^{T}\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)p(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)$$

$$\theta^\star = \arg\max_\theta E_{\tau\sim p_\theta(\tau)}\left[\sum_t r(\mathbf{s}_t,\mathbf{a}_t)\right]$$

Reinforcement Learning: Recent Study

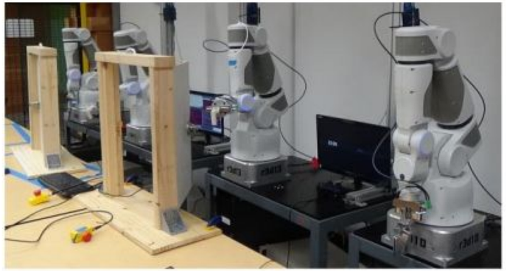
Atari games:

Q-learning:

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, et al. “Playing Atari with Deep Reinforcement Learning”. (2013).

Policy gradients:

J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. “Trust Region Policy Optimization”. (2015).
V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, et al. “Asynchronous methods for deep reinforcement learning”. (2016).



Real-world robots:

Guided policy search:

S. Levine*, C. Finn*, T. Darrell, P. Abbeel. “End-to-end training of deep visuomotor policies”. (2015).

Q-learning:

S. Gu*, E. Holly*, T. Lillicrap, S. Levine. “Deep Reinforcement Learning for Robotic Manipulation with Asynchronous Off-Policy Updates”. (2016).

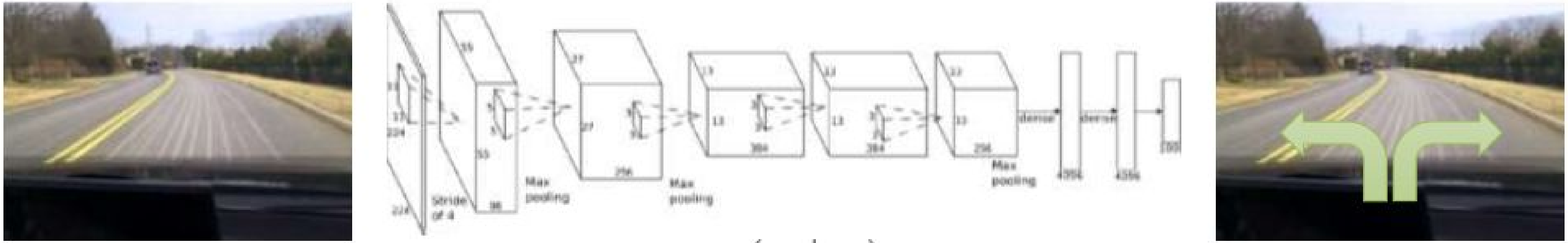
Beating Go champions:

Supervised learning + policy gradients + value functions + Monte Carlo tree search:

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, et al. “Mastering the game of Go with deep neural networks and tree search”. Nature (2016).

Reinforcement Learning : Achievements

- Acquire high degree of proficiency in domains governed by simple, known rules
- Learn simple skills with raw sensory inputs, given enough experience
- Learn from imitating enough human-provided expert behavior



$\mathbf{o}_t$ $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ $\mathbf{a}_t$

Reinforcement Learning: Challenges

- Humans can learn incredibly quickly, while deep RL methods are usually slow
- Require a large amount of supervision or experience per task
- Can not reuse experience from previous tasks to more quickly solve new tasks
- Not clear what the reward function should be
- Not clear what the role of prediction should be
- Still hard to handle “real world”: complex, non-stationary environment

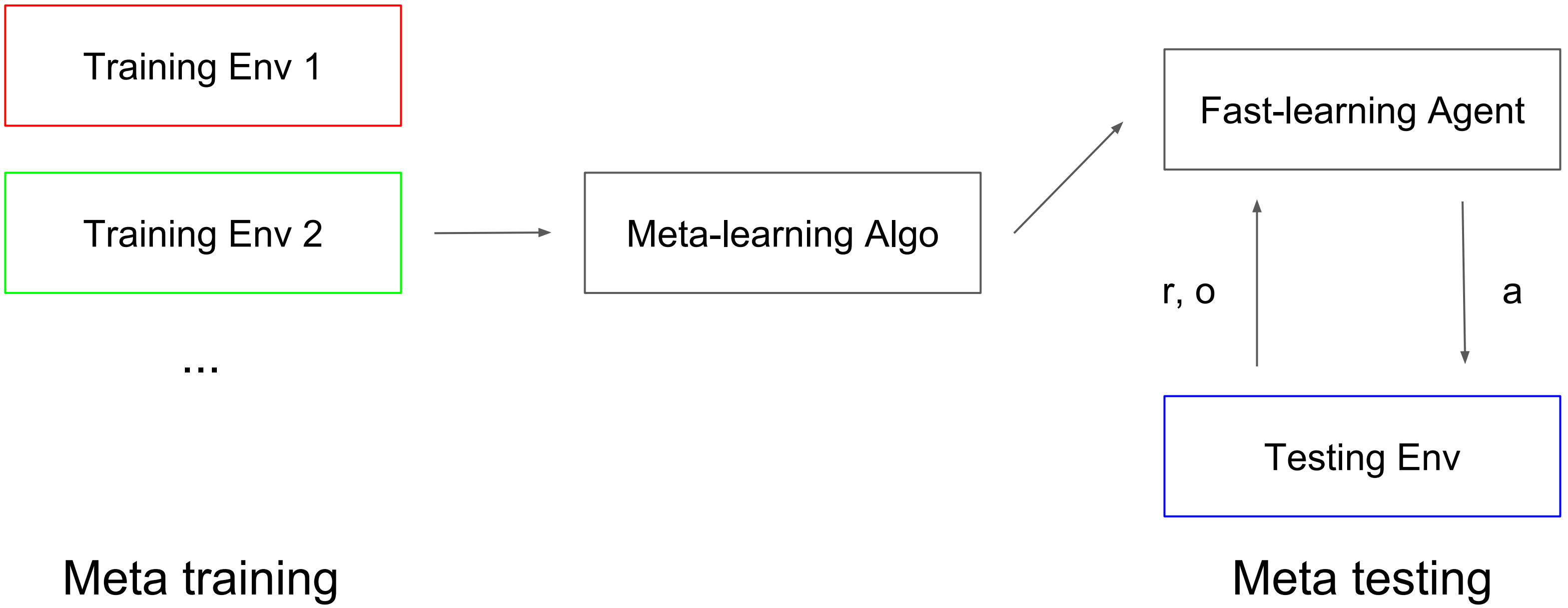
Here we try to use transfer learning to handle some challenges

Transfer Learning in RL

- Standard finetuning with RL is hard
  - Not single input and output → A sequence of trajectory
  - Prone to make catastrophic mistakes from an early deviation
- "Forward" transfer:
  - Train on one task (source domain), transfer to a new task (target domain)
  - From simulation to real world → domain randomization (Tobin et al, 2017)
- Multi-task transfer:
  - Train on many tasks to add diversity, then transfer to a new task
  - Meta learning : transfer learning across tasks, try to reuse past experience

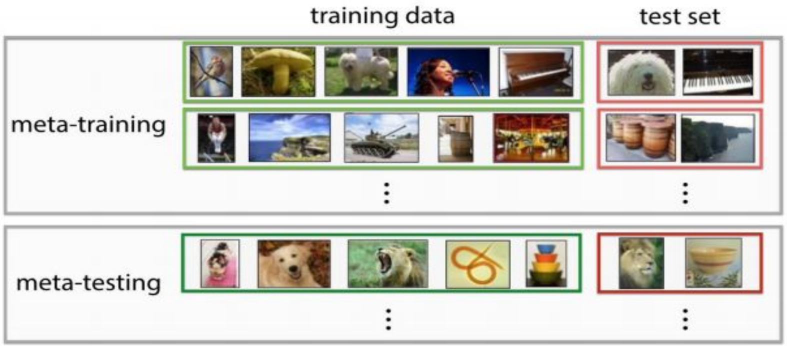
Meta Learning : Learning to Learn

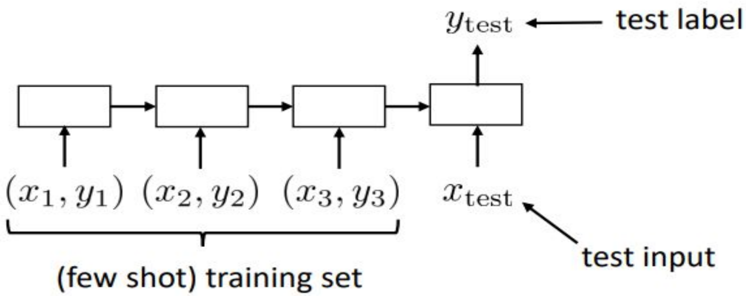
- If you have learned 100 tasks, can you get some insights from them, which can help you to solve a new task?



Pieter Abbeel, NIPS 2017

Meta Learning for Classification





supervised learning: $f(x) \rightarrow y$

input (e.g., image) output (e.g., label)

supervised meta-learning: $f(\mathcal{D}_{\text{train}}, x) \rightarrow y$

training set

- Meta-based LSTM
- Learn update function for both weight initializaton and optimizer
- Introduce additional parameters

Ravi & Larochelle, 2017. Optimization as a Model for Few-Shot Learning

Meta Learning for Optimization

- Wichrowska et al, 2017. Learned Optimizers that Scale and Generalize
- Ke et al, 2017. Learning to Optimize Neural Nets
- Wu et al, 2017. Understanding Short-Horizon Bias in Stochastic Meta-Optimization

Meta Learning for RL

- Schmidhuber et al, 1998. Reinforcement learning with self-modifying policies
- Schmidhuber et al, 1999. A general method for incremental self-improvement and multiagent learning
- Singh et al, 2010. Intrinsically motivated reinforcement learning: An evolutionary perspective
- Niekum et al, 2011. Evolution of reward functions for reinforcement learning
- Duan et al, 2016. RL2: Fast Reinforcement Learning via Slow Reinforcement Learning
- Wang et al, 2016. Learning to reinforcement learn
- Finn et al, 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks
- Finn et al, 2017. One-Shot Visual Imitation Learning via Meta-Learning
- Frans et al, 2017. Meta-learning shared Hierarchies
- Al-Shedivat et al, 2017. Continuous Adaptation via Meta-Learning in Nonstationary and Competitive Environments

Model-Agnostic Meta Learning

MAML: Introduction

- MAML: can be treated as an initialization method to get “robust base model” that is easy to fine-tune to new tasks
- Model agnostic: both supervised learning and reinforcement learning
- Do not need additional parameters
- Recall: Ravi & Larochelle 2017, only classification, need to introduce new parameters

Finn et al, 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks

MAML: Problem Setup

- Goal: train a "meta learning" model on a set of tasks, then this model can adapt to a new task with only a few data / iterations $\rightarrow$ learn **as much as** possible with limited data
- Train a model or policy $f : x \rightarrow a$
  - x : observations
  - a : outputs
  - This model is like a "base model" which will be able to adapt to a lot of new tasks
- A task $T = \{L(x_1, a_1, \ldots, x_H, a_H), q(x_1), q(x_{t+1}|x_t, a_t), H\}$:
  - $L \rightarrow R$: loss function
  - $q(x_1)$: distribution over initial observations
  - $q(x_{t+1}|x_t, a_t)$: transition distribution
  - H : Episode length, for supervised learning, $H = 1$

Finn et al, 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks

MAML: Algorithm

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**

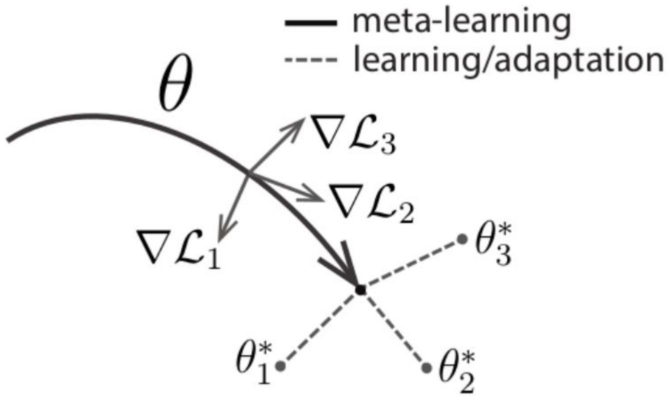
Finn et al, 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks

MAML: Algorithm

- Train model f:
  - Sample a new task T_i from $p(T)$ (training taskset)
  - Learn T_i :
    - Train model with K samples drawn from q_i
    - Get feedback L_{T_i} from T_i
  - Test on new samples from T_i and get test error
  - Improve model f : treat the test error on sampled tasks T_i as the training error of meta-learning process
- Test meta-learning:
  - Sample new task from $p(T)$ (testing taskset), try to adapt f to this new task
  - Learn the model with K samples
  - Treat the performance as "meta-performance"

Finn et al, 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks

MAML: Algorithm



- Adapt \theta to \theta_1, \theta_2, \theta_3 to fit T_1, T_2, and T_3
- Compute the test error of these training tasks to update \theta → a direction that is easier to fine-tune
- Preference : find model parameters that are **sensitive to changes of the task**
- T_1, T_2, and T_3 should be “similar tasks”?

Finn et al, 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks

MAML: different species

$$L_{T_i}(f_\psi) = -E_{x_t, a_t \sim f_\psi, q_{T_i}}\left[\sum_{t=1}^{H} R_i(x_t, a_t)\right] \quad (4)$$

Algorithm 2 MAML for Few-Shot Supervised Learning

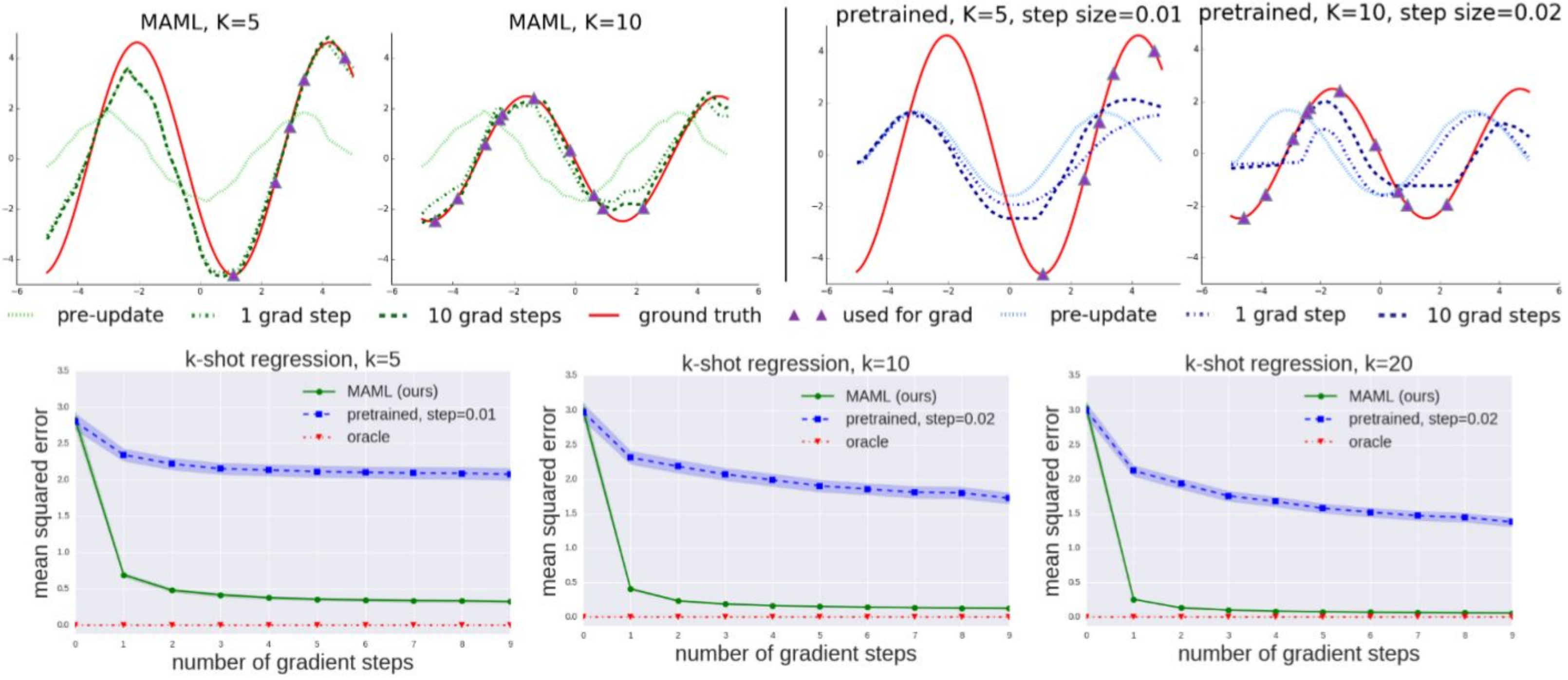
Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Sample K datapoints $\mathcal{D} = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ 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**

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 f_θ 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**

Finn et al, 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks

MAML: Regression Experiment



Finn et al, 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks

MAML: Classification Experiment

Omniglot (Lake et al., 2011)	5-way Accuracy 1-shot	5-way Accuracy 5-shot	20-way Accuracy 1-shot	20-way Accuracy 5-shot
MANN, no conv (Santoro et al., 2016)	82.8%	94.9%	–	–
MAML, no conv (ours)	**89.7 ± 1.1%**	**97.5 ± 0.6%**	–	–
Siamese nets (Koch, 2015)	97.3%	98.4%	88.2%	97.0%
matching nets (Vinyals et al., 2016)	98.1%	98.9%	93.8%	98.5%
neural statistician (Edwards & Storkey, 2017)	98.1%	99.5%	93.2%	98.1%
memory mod. (Kaiser et al., 2017)	98.4%	99.6%	95.0%	98.6%
MAML (ours)	**98.7 ± 0.4%**	**99.9 ± 0.1%**	**95.8 ± 0.3%**	**98.9 ± 0.2%**

MiniImagenet (Ravi & Larochelle, 2017)	5-way Accuracy 1-shot	5-way Accuracy 5-shot
fine-tuning baseline	28.86 ± 0.54%	49.79 ± 0.79%
nearest neighbor baseline	41.08 ± 0.70%	51.04 ± 0.65%
matching nets (Vinyals et al., 2016)	43.56 ± 0.84%	55.31 ± 0.73%
meta-learner LSTM (Ravi & Larochelle, 2017)	43.44 ± 0.77%	60.60 ± 0.71%
MAML, first order approx. (ours)	**48.07 ± 1.75%**	**63.15 ± 0.91%**
MAML (ours)	**48.70 ± 1.84%**	**63.11 ± 0.92%**

Note that current SOTA is TCML (Mishra et al 2017) :

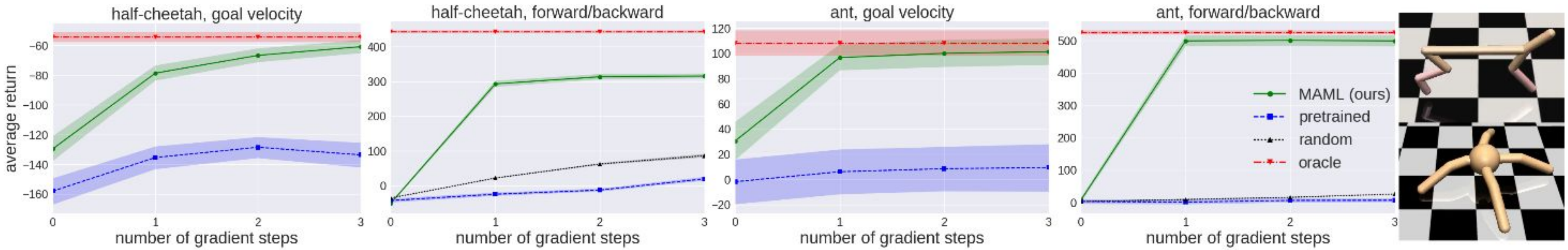
TCML, Ours	**98.96% ± 0.20%**	**99.75% ± 0.11%**	**97.64% ± 0.30%**	**99.36% ± 0.18%**

TCML, Ours	**55.71% ± 0.99%**	**68.88% ± 0.92%**

Finn et al, 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks

MAML: Reinforcement Learning Experiment

- Performance of MuJoCo simulation
- https://sites.google.com/view/maml
- Adapt to new goal velocities and directions substantially faster than conventional pretraining or random initialization
- Achieve good performance in just two or three gradient steps



Finn et al, 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks

MAML: Summary

- Contribution
  - Simple and no additional parameters
  - Adapt to new tasks with few-shot
  - Model Agnostic
- Limitation
  - Tough optimization problem
  - Hard to design task distribution
  - Sensitive to task distribution

Finn et al, 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks

Meta Learning with Imitation Learning

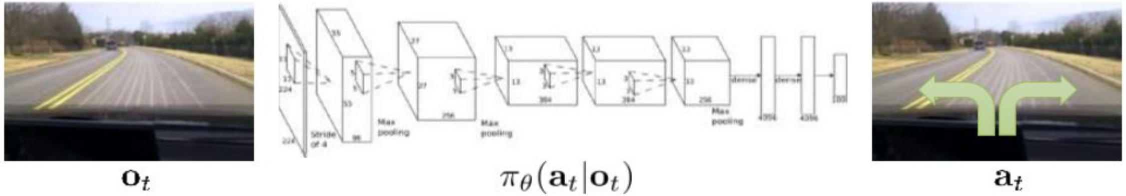
MIL: Introduction

- An application of MAML
- Combine imitation learning with meta-learning for one-shot learning from visual demonstrations
- Reuse past experience to train the "base model", then adapt it to new task with only a single demonstration

If a robot can pick up an apple, a pear, …, then it can learn to pick an orange fastly!

Finn et al, 2017. One-Shot Visual Imitation Learning via Meta-Learning

Imitation Learning



Challenges:

- Behavioral cloning: requires a large number of demonstrations for each task
- Inverse RL or GANs: hard to evaluatereward in high dimension (images), GAN’s problems
- On-policy methods: human’s feedback

MIL: Problem Setup

- Goal : learn a policy that can quickly adapt to new tasks from a single demonstration of that task
- Each imitation task $T_i = \{\tau = \{o_1, a_1, \ldots, o_T, a_T\} \sim \pi_i^*, L(a_{1:T}, \hat{a}_{1:T}), T\}$
- o_t is the observation at time t, i.e. an image, while a_t is the action
- For demonstration trajectory τ, we use MSE to compute loss:

$$L_{T_i}(f_\phi) = \sum_{\tau_j \sim T_i} \sum_t ||f_\phi(o_t^{(j)}) - a_t^{(j)}||_2^2 \qquad (2)$$

Finn et al, 2017. One-Shot Visual Imitation Learning via Meta-Learning

MIL: Algorithm

Algorithm 1 Meta-Imitation Learning with MAML

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 demonstration $\tau = \{\mathbf{o}_1, \mathbf{a}_1, ...\mathbf{o}_T, \mathbf{a}_T\}$ from $\mathcal{T}_i$
        Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using τ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation (2)
        Compute adapted parameters with gradient descent: $\theta'_i = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
        Sample demonstration $\tau'_i = \{\mathbf{o}'_1, \mathbf{a}'_1, ...\mathbf{o}'_T, \mathbf{a}'_T\}$ from $\mathcal{T}_i$ for the meta-update
    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 τ'_i and $\mathcal{L}_{\mathcal{T}_i}$ in Equation 2
end while
return parameters θ that can be quickly adapted to new tasks through imitation.

Finn et al, 2017. One-Shot Visual Imitation Learning via Meta-Learning

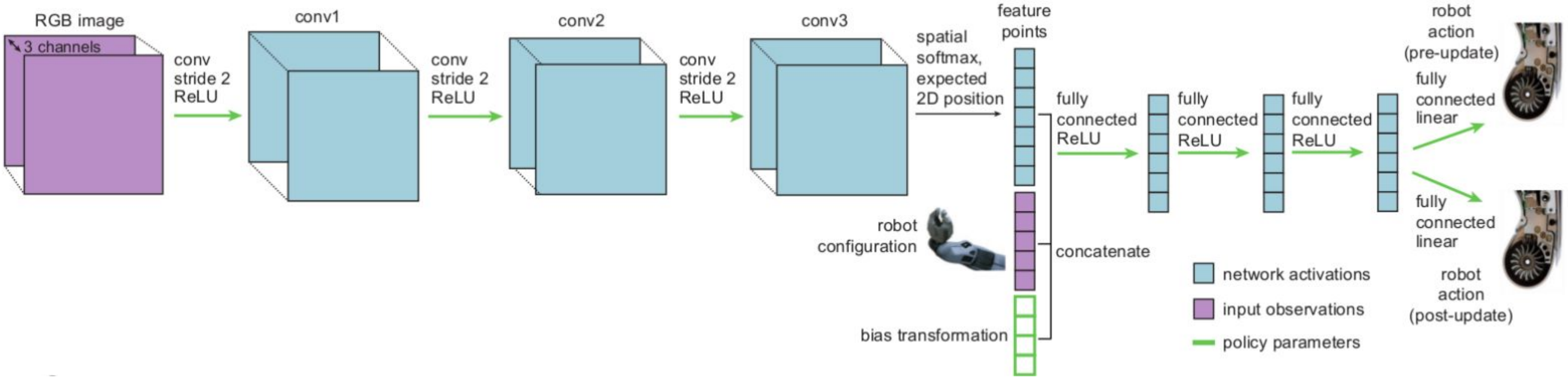
MIL: Algorithm

- Meta-training:
  - Assume each training task has at least 2 demonstrations, thus we can sample a set of tasks with **two demonstrations per task**
  - For each task T_i, train θ_i' with its one demonstration τ_i $\rightarrow$ inner loop of meta-learning
  - Use another demonstration τ_i' to "test" θ_i' , i.e. check the mse of predicted actions and demonstration actions
  - Then we can update θ according to the gradient of meta-objective
  - As we get a series of $\theta_i' s$ and their testing error, we can update θ
  - Finally we can get trained parameters θ for meta-learner
- Meta-testing:
  - Sample a new task T and its one demonstration
  - This task can involve new goals or manipulating new, previously unseen objects
  - Then we can adapt θ to this task

Finn et al, 2017. One-Shot Visual Imitation Learning via Meta-Learning

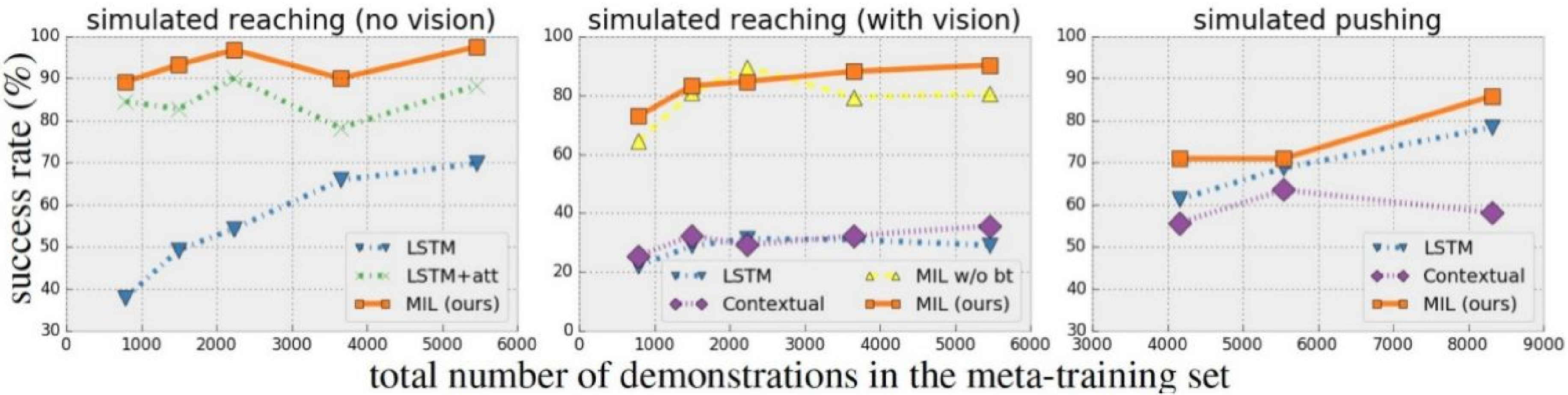
MIL: Some other modifications

- Two head structure: more flexibility during adapting
- Learn to imitate without expert actions: **just mentioned, maybe future work**
- Layer normalization: data within a demonstration trajectory is highly correlated across time, thus BN is not effective
- Bias transformation: increase the representational power of the gradient



MIL: Experiment

- Comparision: MIL, random policy, contextual policy, LSTM, LSTM with attention
- Task: simulated reaching/pushing, real-world placing
- https://sites.google.com/view/one-shot-imitation/



Finn et al, 2017. One-Shot Visual Imitation Learning via Meta-Learning

MIL: Experiment

subset of training objects | test objects | subset of training objects | test objects

Task:

Evaluate how well a real robot (PR2) can learn to interact with new unknown objects from a single visual demonstration.

Success: the held object landed in or on the target container after the gripper is opened

method	test performance
LSTM	25%
contextual	25%
MIL	**90%**
MIL, video only	**68.33%**

Table 2: One-shot success rate of placing a held item into the correct container, with a real PR2 robot, using 29 held-out test objects. Meta-training used a dataset with $\sim$100 objects. MIL, using video only receives the only video part of the demonstration and not the arm trajectory or actions.

Finn et al, 2

MIL: Summary

- Contribution:
  - Reuse prior experience when learning in new settings
  - Effective one-shot imitation learning
- Limitation:
  - Problems of MAML
  - How to define and collect “similar tasks / demonstrations”?
    - For demonstrations, is L2-distance sufficient for evaluating similarity?
    - For tasks, we need to find a quantify method instead of assigning manually
  - Can we learn from “third-person perspective”? Some related work:
    - Stadie et al, 2017. Third-Person Imitation Learning
    - Liu et al, 2017. Imitation from Observation- Learning to Imitate Behaviors from Raw Video via Context Translation

Meta Learning with Hierarchical RL

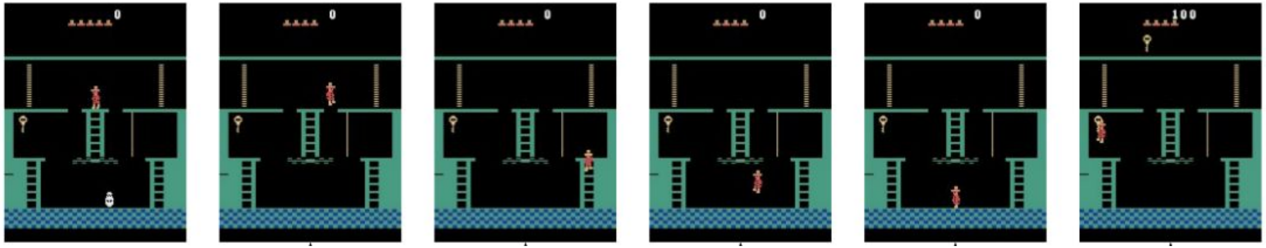
MLSH: Introduction

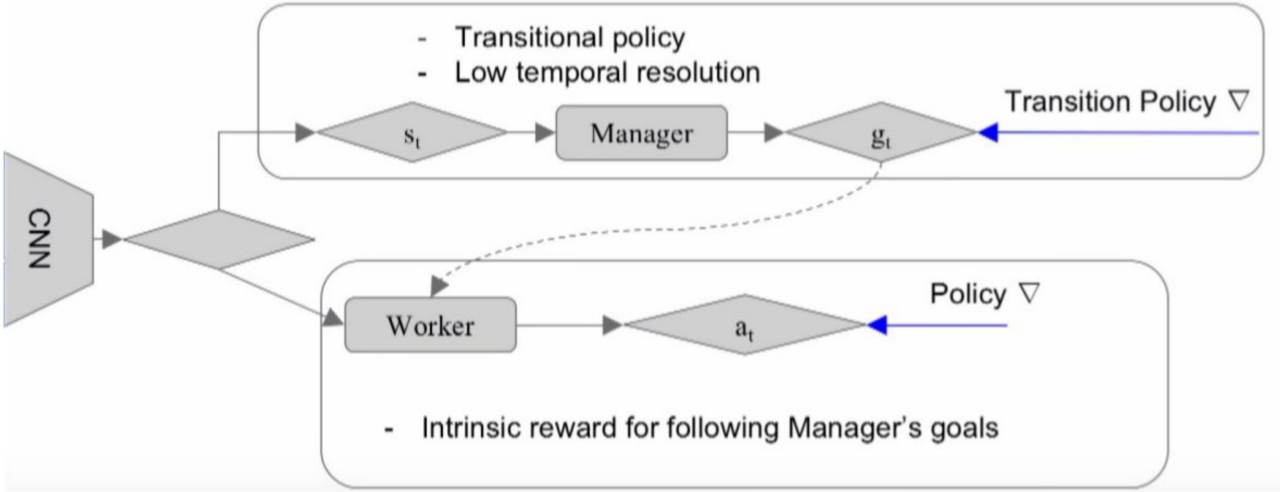
- Recall that MAML try to learn as much as possible with limited data
- Goal of MLSH: try to learn faster
- Hierarchical architecture: master policy and sub-policies
  - Sub-policies (primitives) are shared within a distribution of tasks
  - Task-specific master policy is used to switch sub-policies
- Meta-learn: learn sub-policies as base models
- For a new task, we just need to learn how to choose them correctly (i.e. master policy)

Frans et al, 2017. Meta-learning shared Hierarchies

Hierarchical RL

- Why hierarchical?
  - Long-term credit assignment
  - Sparse reward
- An example -- FeUdal Net



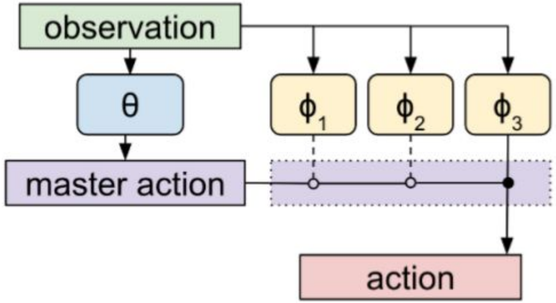


Vezhnevets et al, 2017. FeUdal Networks for Hierarchical Reinforcement Learning

MLSH: Problem Setup

- Define a policy $\pi_{\phi,\theta}(a|s)$
  - ϕ :
    - A set of parameters **shared between all tasks**
    - $\phi = \{\phi_1, \phi_2, \ldots, \phi_K\}$
    - Each $\phi_k \rightarrow$ the parameters of a sub-policy $\pi_{\phi_k}(a|s)$
  - θ :
    - The parameters of master policy
    - Task-specific $\rightarrow$ zero or random initialized at the beginning
    - Choose a sub-task to activate for given timestep
- For a task M sampled from P_M
  - Randomly initialized θ and shared ϕ
  - Goal: learn θ , note that this is just the **objective for current task**
- The objective of meta-learning:
  - By learning training tasks, try to **find shared parameter** psi which can be generalize to a new MDP
  - Then for a new task, only learn θ

$$maximize_{\phi} E_{M \sim P_M, t=0,\ldots,T-1}[R]$$



Why faster?
For a new task we just need to learn \theta
Treat the problem as 1/N times as long

Frans et al, 2017. Meta-learning shared Hierarchies

MLSH: Algorithm

Algorithm 1 Meta Learning Shared Hierarchies

Initialize ϕ
repeat
  Initialize θ
  Sample task $M \sim P_M$
  for $w = 0, 1, ...W$ (warmup period) **do**
    Collect D timesteps of experience using $\pi_{\phi,\theta}$
    Update θ to maximize expected return from $1/N$ timescale viewpoint
  end for
  for $u = 0, 1, ...U$ (joint update period) **do**
    Collect D timesteps of experience using $\pi_{\phi,\theta}$
    Update θ to maximize expected return from $1/N$ timescale viewpoint
    Update ϕ to maximize expected return from full timescale viewpoint
  end for
until convergence

Frans et al, 2017. Meta-learning shared Hierarchies

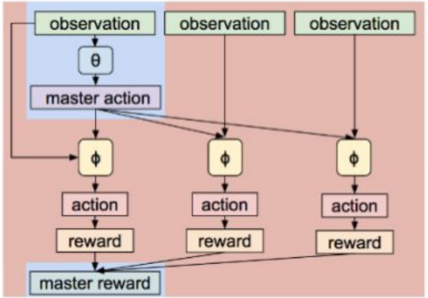
MLSH: Algorithm

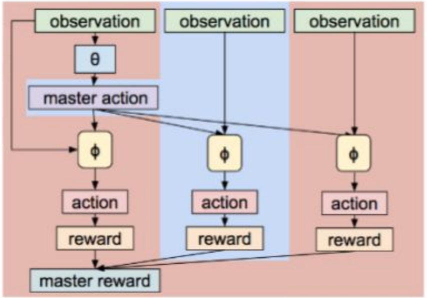
- Warm-up period:
  - **Goal : try to optimize θ to nearly optimal**
  - In this period, we hold ϕ fixed
  - For each iteration sample D timesteps of experience
  - For each 1/N timescale, consider a sub-policy as an "action"
- Joint update period:
  - **Both θ and ϕ are updated**
  - For each iteration, collect experience and optimize $\theta \rightarrow$ **same as warm-up**
  - Update ϕ : reuse these D samples, but viewed via sub-policy
  - Treat the master policy as an extension of the environment $\rightarrow$ a discrete portion of observation
  - For each N-timestep slice of experience, we only update the parameters of the sub-policy that had been activated by master policy

Frans et al, 2017. Meta-learning shared Hierarchies

MLSH: Algorithm

- Left : warm-up, update master policy
  - Here we hold sub-policies fixed, the available action is to choose one of them at each time slice
- Right : train a sub-policy
  - During joint-updating we also need to update $\boldsymbol{\theta}$ first
  - Then we use the same data to update $\boldsymbol{\phi}$
  - Note that currently only the blue sub-policy is chosen by master $\rightarrow$ only update this sub-policy

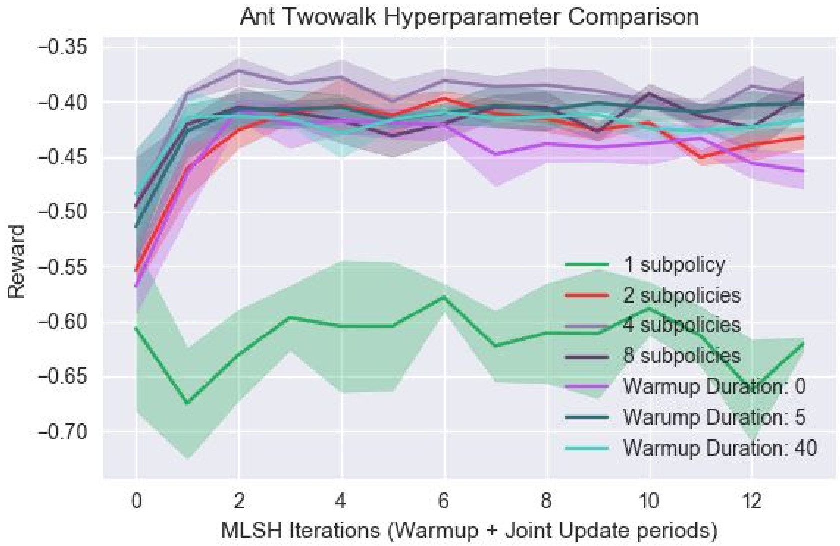




Frans et al, 2017. Meta-learning shared Hierarchies

MLSH: Experiment

- https://sites.google.com/site/mlshsupplementals
- Task 1: 2D moving bandits
- Task 2: simulated walk
- Additional experiment: the number of sub-policies, warm-up duration



Frans et al, 2017. Meta-learning shared Hierarchies

MLSH: Summary

- Contribution:
  - Compared with FeUdal Net: handle multi-task
  - Compared with MAML: try to be faster
- Limitation:
  - The description is still in high-level
  - Assumption: \theta can be trained to be optimal during warm-up even at the very beginning
  - How to define sub-policies, manually or automatically?
  - Interesting work, but still need to improve

Frans et al, 2017. Meta-learning shared Hierarchies

Meta-learning for Non-stationary Environments

MAML for Non-stationary: Introduction

- Challenges: real world is non-stationary
  - Dynamics and objectives change over life-long time
  - Multiple learning agents
- This paper:
  - Treat non-stationary task as a sequence of stationary tasks
  - Modify MAML for multi-task, then extend to dynamically changing tasks
  - Specifically, Find the dependence between consecutive tasks

Al-Shedivat et al, 2017. Continuous Adaptation via Meta-Learning in Nonstationary and Competitive Environments

Original MAML

- A task is defined as

$$T = L_T, P_T(x), P_T(x_{t+1}|x_t, a_t), H \quad (1)$$

- Inner loop:
  - In original MAML, ϕ is called θ'
  - In original MAML, α is call β in inner loop

$$\phi := \theta - \alpha \nabla_\theta \mathcal{L}_T \left(\boldsymbol{\tau}_\theta^{1:K} \right), \text{ where } \mathcal{L}_T \left(\boldsymbol{\tau}_\theta^{1:K} \right) := \frac{1}{K} \sum_{k=1}^{K} \mathcal{L}_T(\boldsymbol{\tau}_\theta^k), \text{ and } \boldsymbol{\tau}_\theta^k \sim P_T(\boldsymbol{\tau} \mid \theta) \quad (2)$$

- Meta objective:

$$\min_\theta \mathbb{E}_{T \sim \mathcal{D}(T)} \left[\mathcal{R}_T(\theta) \right], \text{ where } \mathcal{R}_T(\theta) := \mathbb{E}_{\boldsymbol{\tau}_\theta^{1:K} \sim P_T(\boldsymbol{\tau}|\theta)} \left[\mathbb{E}_{\boldsymbol{\tau}_\phi \sim P_T(\boldsymbol{\tau}|\phi)} \left[\mathcal{L}_T(\boldsymbol{\tau}_\phi) \mid \boldsymbol{\tau}_\theta^{1:K}, \theta \right] \right] \quad (3)$$

where $\boldsymbol{\tau}_\theta$ and $\boldsymbol{\tau}_\phi$ are trajectories obtained under π_θ and π_ϕ, respectively.

Al-Shedivat et al, 2017. Continuous Adaptation via Meta-Learning in Nonstationary and Competitive Environments

Probabilistic View of MAML

- Probabilistic view:
  - Task T , trajectories τ and policies π_θ are random variables, phi is generated from conditional distribution $P_T(\phi|\theta, \tau_{1:k})$
  - Inner loop update: equivalent to assuming the delta distribution

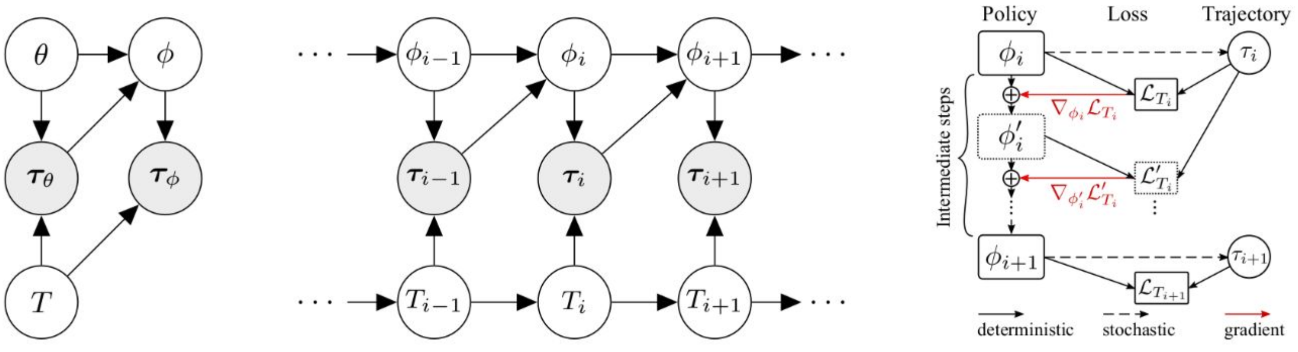
$$P_T(\phi|\theta, \tau_{1:k}) := \delta(\theta - \alpha \nabla_\theta \frac{1}{K} \sum_{k=1}^{K} L(\tau_k))$$

  - Optimize meta-objective: PG where the gradient of $R_T(\theta)$

$$\nabla_\theta \mathcal{R}_T(\theta) = \mathbb{E}_{\substack{\boldsymbol{\tau}_\theta^{1:K} \sim P_T(\boldsymbol{\tau}|\theta) \\ \boldsymbol{\tau}_\phi \sim P_T(\boldsymbol{\tau}|\phi)}} \left[\mathcal{L}_T(\boldsymbol{\tau}_\phi) \left[\nabla_\theta \log \pi_\phi(\boldsymbol{\tau}_\phi) + \nabla_\theta \sum_{k=1}^{K} \log \pi_\theta(\boldsymbol{\tau}_\theta^k) \right] \right] \tag{4}$$

Al-Shedivat et al, 2017. Continuous Adaptation via Meta-Learning in Nonstationary and Competitive Environments

Probabilistic View of MAML



(a) (b) (c)

- Probabilistic graph of MAML
  - (a) MAML in a multi-task RL setting
  - (b) Extend to continuous adaptation
    - Policy and trajectories at a previous step are used to construct a new policy for the current step, i.e. $\phi_i, \tau_i \rightarrow \phi_{i+1}$
  - (c) Computation graph for the meta-update from ϕ_i to ϕ_{i+1}
    - The model is optimized via truncated backpropagation through time starting from $L_{T_{i+1}}$

Al-Shedivat et al, 2017. Continuous Adaptation via Meta-Learning in Nonstationary and Competitive Environments

Meta-learning for Continuous Adaptation

- $D(T)$ is defined by the environment changes, and the tasks become sequentially dependent
- Goal:
  - Find the dependence between consecutive tasks
  - Meta-learn a rule to minimize total expected loss during interacting
  - An example, if our enemy changes action, we need to do some adjustment as well

$$\mathcal{R}_{T_i,T_{i+1}}(\theta) := \mathbb{E}_{\boldsymbol{\tau}_{i,\theta}^{1:K} \sim P_{T_i}(\boldsymbol{\tau}|\theta)} \left[\mathbb{E}_{\boldsymbol{\tau}_{i+1,\phi} \sim P_{T_{i+1}}(\boldsymbol{\tau}|\phi)} \left[\mathcal{L}_{T_{i+1}}(\boldsymbol{\tau}_{i+1,\phi}) \mid \boldsymbol{\tau}_{i,\theta}^{1:K}, \theta \right] \right] \tag{6}$$

The principal difference between the loss in ($\boxed{3}$) and ($\boxed{6}$) is that trajectories $\boldsymbol{\tau}_{i,\theta}^{1:K}$ come from the current task, T_i, and are used to construct a policy, π_ϕ, that is good for the upcoming task, T_{i+1}.

Al-Shedivat et al, 2017. Continuous Adaptation via Meta-Learning in Nonstationary and Competitive Environments

MAML for Non-stationary: Training

Algorithm 1 Meta-learning at training time.

```
input  Distribution over pairs of tasks, P(T_i, T_{i+1}),
       learning rate, β.
Randomly initialize θ and α.
repeat
    Sample a batch of task pairs, {(T_i, T_{i+1})}_{i=1}^n.
    for all task pairs (T_i, T_{i+1}) in the batch do
        Sample traj. τ_{1:K} from T_i using π_θ.
        Compute φ = φ(τ_{1:K}, θ, α) as given in (7).
        Sample traj. τ from T_{i+1} using π_φ.
    end for
    Construct ∇_θ R_T(θ, α) and ∇_α R_T(θ, α) using τ_{1:K} and τ as given in (8).
    Update θ ← θ + β∇_θ R_T(θ, α).
    Update α ← α + β∇_α R_T(θ, α).
until Convergence
output  Optimal θ* and α*.
```

$$
\begin{aligned}
\phi_i^0 &:= \theta, \quad \boldsymbol{\tau}_\theta^{1:K} \sim P_{T_i}(\boldsymbol{\tau} \mid \theta), \\
\phi_i^m &:= \phi_i^{m-1} - \alpha_m \nabla_{\phi_i^{m-1}} \mathcal{L}_{T_i}\left(\boldsymbol{\tau}_{i,\phi_i^{m-1}}^{1:K}\right), \quad m = 1, \ldots, M-1, \\
\phi_{i+1} &:= \phi_i^{M-1} - \alpha_M \nabla_{\phi_i^{M-1}} \mathcal{L}_{T_i}\left(\boldsymbol{\tau}_{i,\phi_i^{M-1}}^{1:K}\right)
\end{aligned}
\tag{7}
$$

$$
\nabla_{\theta,\alpha} \mathcal{R}_{T_i,T_{i+1}}(\theta,\alpha) = \mathop{\mathbb{E}}_{\substack{\boldsymbol{\tau}_{i,\theta}^{1:K} \sim P_{T_i}(\boldsymbol{\tau}|\theta) \\ \boldsymbol{\tau}_{i+1,\phi} \sim P_{T_{i+1}}(\boldsymbol{\tau}|\phi)}} \left[\mathcal{L}_{T_{i+1}}(\boldsymbol{\tau}_{i+1,\phi}) \left[\nabla_{\theta,\alpha} \log \pi_\phi(\boldsymbol{\tau}_{i+1,\phi}) + \nabla_\theta \sum_{k=1}^{K} \log \pi_\theta(\boldsymbol{\tau}_{i,\theta}^k) \right] \right] \tag{8}
$$

Al-Shedivat et al, 2017. Continuous Adaptation via Meta-Learning in Nonstationary and Competitive Environments

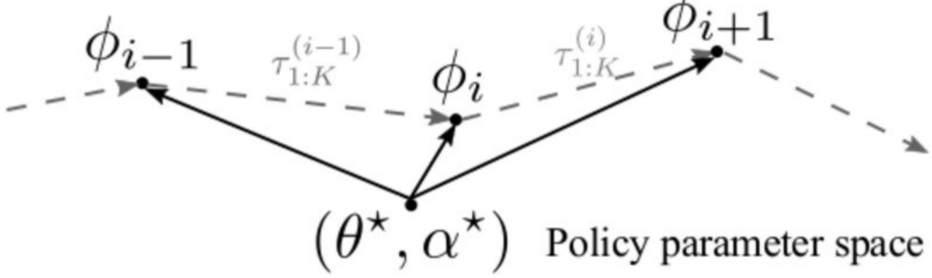
MAML for Non-stationary: Testing

- Nonstationary $\rightarrow$ can not access to same task multiple times
- How to handle : keep acting according to π_ϕ and re-use past experience to for computing updates of ϕ for each new incoming task
- Importance weight correction : past experience obtained by π_ϕ is different from π_θ, we need to make some adjustment

$$\phi_i := \theta - \alpha \frac{1}{K} \sum_{k=1}^{K} \left(\frac{\pi_\theta(\boldsymbol{\tau}^k)}{\pi_{\phi_{i-1}}(\boldsymbol{\tau}^k)} \right) \nabla_\theta \mathcal{L}(\boldsymbol{\tau}^k), \quad \boldsymbol{\tau}^{1:K} \sim P_{T_{i-1}}(\boldsymbol{\tau} \mid \phi_{i-1}), \tag{9}$$

Algorithm 2 Adaptation at execution time.

input A stream of tasks, $T_1, T_2, T_3, \ldots$.
1: Initialize $\phi = \theta$.
2: **while** there are new incoming tasks **do**
3: Get a new task, T_i, from the stream.
4: Solve T_i using π_ϕ policy.
5: While solving T_i, collect trajectories, $\boldsymbol{\tau}_{1:K}^{(i)}$.
6: Update $\phi \leftarrow \phi(\boldsymbol{\tau}_{1:K}^{(i)}, \theta^*, \alpha^*)$ using importance-corrected meta-update as in $\boxed{9}$.
7: **end while**



Al-Shedivat et al, 2017. Continuous Adaptation via Meta-Learning in Nonstationary and Competitive Environments

MAML for Non-stationary: Experiment

- New environment : Robosumo
- https://blog.openai.com/meta-learning-for-wrestling/

Al-Shedivat et al, 2017. Continuous Adaptation via Meta-Learning in Nonstationary and Competitive Environments

MAML for Non-stationary: Summary

- Exploration on non-stationary environment
- Assumption: non-stationary task can be seen as a sequences of correlated stationary tasks
- Train agents to exploit the dependencies between consecutive tasks
- Limitations:
  - Real environment is more complex
  - Can not handle sparse reward → meta-updates use policy gradients and heavily rely on the reward signal
  - Meta-update still requires second order derivates

Al-Shedivat et al, 2017. Continuous Adaptation via Meta-Learning in Nonstationary and Competitive Environments

Summary

What is Meta-learning

- Meta-learning = learning to learn
- Transfer learning across tasks, and is related to multi-task learning
- Get knowledge from past experiences, then adapt to new task fastly / with limited training data

Why Meta-learning?

- Deep RL, especially model free, requires huge number of samples
- Meta-learning makes it easier to learn a new task
- Avoid trying actions that are known to be useless
- More similar to “human thinking”

Open Problems of Meta-learning

- Do we need to achieve model-agnostic, or just focus on RL?
  - Can we make MAML easier to optimize?
- Design appropriate task / demonstration set → Also question for imitation learning
  - How to define and collect “similar task”?
  - Can we derive model for new task that is not so similar to old ones (like “zero-shot”)
  - The more diversity, the better? → Be aware of overfitting!
  - Learn from third-person perspective
- How to combine mata-learning with hierarchical RL?
  - Meta-learn sub-policies? How to handle the very beginning?
  - How to set sub-policies → Also question for hierarchical RL
- How to handle non-stationary environment? → Big problem for RL
  - More robust assumption?
  - Combined with life-long learning (keep learning even during testing)?

Other Reference

- UCB CS 294 2017 Fall
- Deep Learning for Robotics, NIPS 2017 Keynotes
- More paper notes can be found in my github : https://github.com/YunqiuXu/Readings
  - 1703.03400
  - 1709.04905
  - 1710.09767
  - 1710.03641