Meta-Learning & Transfer Learning

CS 285

Instructor: Sergey Levine
UC Berkeley
What’s the problem?

this is easy (mostly)

this is impossible

Why?
Montezuma’s revenge

- Getting key = reward
- Opening door = reward
- Getting killed by skull = bad
Montezuma’s revenge

• We know what to do because we understand what these sprites mean!
• Key: we know it opens doors!
• Ladders: we know we can climb them!
• Skull: we don’t know what it does, but we know it can’t be good!
• Prior understanding of problem structure can help us solve complex tasks quickly!
Can RL use the same prior knowledge as us?

• If we’ve solved prior tasks, we might acquire useful knowledge for solving a new task

• How is the knowledge stored?
  • Q-function: tells us which actions or states are good
  • Policy: tells us which actions are potentially useful
    • some actions are never useful!
  • Models: what are the laws of physics that govern the world?
  • Features/hidden states: provide us with a good representation
    • Don’t underestimate this!
Transfer learning terminology

**transfer learning**: using experience from one set of tasks for faster learning and better performance on a new task.

**in RL, task = MDP!**

- **source domain**
- **target domain**

  - **0-shot**: just run a policy trained in the source domain
  - **1-shot**: try the task once
  - **few shot**: try the task a few times

“shot”: number of attempts in the target domain
How can we frame transfer learning problems?

1. Forward transfer: learn policies that transfer effectively
   a) Train on source task, then run on target task (or finetune)
   b) Relies on the tasks being quite similar!

2. Multi-task transfer: train on many tasks, transfer to a new task
   a) Sharing representations and layers across tasks in multi-task learning
   b) New task needs to be similar to the distribution of training tasks

3. Meta-learning: learn to learn on many tasks
   a) Accounts for the fact that we’ll be adapting to a new task during training!
Pretraining + Finetuning

The most popular transfer learning method in (supervised) deep learning!
What issues are we likely to face?

- **Domain shift:** representations learned in the source domain might not work well in the target domain

- **Difference in the MDP:** some things that are possible to do in the source domain are not possible to do in the target domain

- **Finetuning issues:** if pretraining & finetuning, the finetuning process may still need to explore, but optimal policy during finetuning may be deterministic!
Domain adaptation in computer vision

Invariance assumption: everything that is different between domains is irrelevant

formally:

\( p(x) \) is different \( \exists z = f(x) \) such that \( p(y|z) = p(y|x) \), but \( p(z) \) is same
Domain adaptation in RL for dynamics?

Why is invariance not enough when the dynamics don’t match?

\[
\Delta r(s_t, a_t, s_{t+1}) = \log p_{\text{target}}(s_{t+1} \mid s_t, a_t) - \log p_{\text{source}}(s_{t+1} \mid s_t, a_t)
\]

\[
\Delta r(s_t, a_t, s_{t+1}) = \log p(\text{target} \mid s_t, a_t, s_{t+1}) - \log p(\text{target} \mid s_t, a_t) - \log p(\text{source} \mid s_t, a_t, s_{t+1}) + \log p(\text{source} \mid s_t, a_t)
\]

When might this not work?

Eysenbach et al., “Off-Dynamics Reinforcement Learning: Training for Transfer with Domain Classifiers”
What if we can also finetune?

1. RL tasks are generally much less diverse
   • Features are less general
   • Policies & value functions become overly specialized

2. Optimal policies in fully observed MDPs are deterministic
   • Loss of exploration at convergence
   • Low-entropy policies adapt very slowly to new settings

See “exploration 2” lecture on unsupervised skill discovery and “control as inference” lecture on MaxEnt RL methods!
How to maximize **forward** transfer?

**Basic intuition:** the more varied the training domain is, the more likely we are to generalize in **zero shot** to a slightly different domain.

“**Randomization**” (dynamics/appearance/etc.): widely used for simulation to real world transfer (e.g., in robotics)
EPOpt: randomizing physical parameters

Rajeswaran et al., “EPOpt: Learning robust neural network policies...”
More randomization!


Xue Bin Peng et al., “Sim-to-Real Transfer of Robotic Control with Dynamics Randomization.” 2018

Lee et al., “Learning Quadrupedal Locomotion over Challenging Terrain.” 2020
Some suggested readings

Domain adaptation:

Finetuning:
Andreas et al. Modular multitask reinforcement learning with policy sketches. 2017.
Kumar et al. One Solution is Not All You Need: Few-Shot Extrapolation via Structured MaxEnt RL. 2020

Simulation to real world transfer:
Tobin et al. (2017). Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World.

...and many many others!
How can we frame transfer learning problems?

1. Forward transfer: learn policies that transfer effectively
   a) Train on source task, then run on target task (or finetune)
   b) Relies on the tasks being quite similar!

2. Multi-task transfer: train on many tasks, transfer to a new task
   a) Sharing representations and layers across tasks in multi-task learning
   b) New task needs to be similar to the *distribution* of training tasks

3. Meta-learning: learn to *learn* on many tasks
   a) Accounts for the fact that we’ll be adapting to a new task during training!
Can we learn faster by learning multiple tasks?

- Accelerate learning of all tasks that are learned together
- Provide better pre-training for down-stream tasks
Can we solve multiple tasks at once?

Multi-task RL corresponds to single-task RL in a **joint MDP**
How does the model know what to do?

- What if the policy can do *multiple* things in the *same* environment?
Contextual policies

standard policy: $\pi_\theta(a|s)$

contextual policy: $\pi_\theta(a|s, \omega)$

e.g., do dishes or laundry

formally, simply defines augmented state space: $\tilde{s} = \begin{bmatrix} s \\ \omega \end{bmatrix}$

$\tilde{S} = S \times \Omega$

$\omega$: stack location

$\omega$: walking direction

$\omega$: where to hit puck
Goal-conditioned policies

\[ \pi_\theta(a|s, g) \]

\[ r(s, a, g) = \delta(s = g) \]

\[ r(s, a, g) = \delta(||s - g|| \leq \varepsilon) \]

- Convenient: no need to manually define rewards for each task
- Can transfer in zero shot to a new task if it’s another goal!
- Often hard to train in practice (see references)
- Not all tasks are goal reaching tasks!

A few relevant papers:
- Kaelbling. Learning to achieve goals.
- Schaul et al. Universal value function approximators.
- Andrychowicz et al. Hindsight experience replay.
Meta-Learning
What is meta-learning?

• If you’ve learned 100 tasks already, can you figure out how to learn more efficiently?
  • Now having multiple tasks is a huge advantage!
• Meta-learning = learning to learn
• In practice, very closely related to multi-task learning
• Many formulations
  • Learning an optimizer
  • Learning an RNN that ingests experience
  • Learning a representation

Image credit: Ke Li
Why is meta-learning a good idea?

• Deep reinforcement learning, especially model-free, requires a huge number of samples
• If we can *meta-learn* a faster reinforcement learner, we can learn new tasks efficiently!
• What can a *meta-learned* learner do differently?
  • Explore more intelligently
  • Avoid trying actions that are know to be useless
  • Acquire the right features more quickly
Meta-learning with supervised learning

image credit: Ravi & Larochelle ‘17
Meta-learning with supervised learning

supervised learning: \( f(x) \rightarrow y \)
input (e.g., image)  output (e.g., label)

supervised meta-learning: \( f(D^{tr}, x) \rightarrow y \)
training set

• How to read in training set?
  • Many options, RNNs can work
  • More on this later
What is being “learned”?

(few shot) training set

\[(x_1, y_1) (x_2, y_2) (x_3, y_3)\]

\[x_{test}\]

\[y_{test}\]

test label

test input

supervised meta-learning: \[f(D_{tr}, x) \rightarrow y\]

---

“Generic” learning:

\[\theta^* = \arg\min_{\theta} \mathcal{L}(\theta, D_{tr})\]

\[= f_{learn}(D_{tr})\]

“Generic” meta-learning:

\[\theta^* = \arg\min_{\theta} \sum_{i=1}^{n} \mathcal{L}(\phi_i, D_{ts}^i)\]

where \[\phi_i = f_{\theta}(D_{tr}^i)\]
What is being “learned”?

“Generic” learning:

\[ \theta^* = \arg \min_\theta \mathcal{L}(\theta, \mathcal{D}^{tr}) \]

\[ = f_{\text{learn}}(\mathcal{D}^{tr}) \]

“Generic” meta-learning:

\[ \theta^* = \arg \min_\theta \sum_{i=1}^{n} \mathcal{L}(\phi_i, \mathcal{D}_i^{ts}) \]

where \( \phi_i = f_{\theta}(\mathcal{D}_i^{tr}) \)

---

![Diagram of RNN with meta-learned weights]
Meta Reinforcement Learning
The meta reinforcement learning problem

“Generic” learning:

\[ \theta^* = \arg \min_\theta \mathcal{L}(\theta, D^{tr}) \]
\[ = f_{\text{learn}}(D^{tr}) \]

Reinforcement learning:

\[ \theta^* = \arg \max_\theta E_{\pi_\theta(\tau)}[R(\tau)] \]
\[ = f_{\text{RL}}(\mathcal{M}) \quad \mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, r\} \]

“Generic” meta-learning:

\[ \theta^* = \arg \min_\theta \sum_{i=1}^{n} \mathcal{L}(\phi_i, D_i^{ts}) \]
\[ \text{where } \phi_i = f_\theta(D_i^{tr}) \]

Meta-reinforcement learning:

\[ \theta^* = \arg \max_\theta \sum_{i=1}^{n} E_{\pi_{\phi_i(\tau)}}[R(\tau)] \]
\[ \text{where } \phi_i = f_\theta(\mathcal{M}_i) \]
The meta reinforcement learning problem

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

where \( \phi_i = f_\theta(M_i) \)

assumption: \( M_i \sim p(M) \)

meta test-time:

sample \( M_{\text{test}} \sim p(M) \), get \( \phi_i = f_\theta(M_{\text{test}}) \)

\( \{M_1, \ldots, M_n\} \)

\( \text{meta-training MDPs} \)
Contextual policies and meta-learning

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

where \( \phi_i = f_{\theta}(M_i) \)

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

\[ \pi_{\theta}(a_t|s_t, s_1, a_1, r_1, \ldots, s_{t-1}, a_{t-1}, r_{t-1}) \]

context used to infer whatever we need to solve \( M_i \) i.e., \( z_t \) or \( \phi_i \) (which are really the same thing)

in meta-RL, the context is inferred from experience from \( M_i \)

in multi-task RL, the context is typically given

\( \phi \): stack location \quad \phi \): walking direction \quad \phi \): where to hit puck

“context”
Meta-RL with recurrent policies

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

where \( \phi_i = f_\theta(M_i) \)

main question: how to implement \( f_\theta(M_i) \)?

what should \( f_\theta(M_i) \) do?

1. improve policy with experience from \( M_i \)
\[ \{(s_1, a_1, s_2, r_1), \ldots, (s_T, a_T, s_{T+1}, r_T)\} \]

2. (new in RL): choose how to interact, i.e. choose \( a_t \)
meta-RL must also choose how to explore!

pick \( a_t \sim \pi_\theta(a_t|s_t) \)

use \( (s_t, a_t, s_{t+1}, r_t) \) to improve \( \pi_\theta \)
Meta-RL with recurrent policies

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

where \( \phi_i = f_\theta(M_i) \)

so... we just train an RNN policy?

yes!

**crucially**, RNN hidden state is **not** reset between episodes!
Why recurrent policies *learn to explore*

1. improve policy with experience from $\mathcal{M}_i$
   \[\{(s_1, a_1, s_2, r_1), \ldots, (s_T, a_T, s_{T+1}, r_T)\}\]
   \[\pi_{\phi_i}^*(a|s)\]

2. (new in RL): choose how to interact, i.e. choose $a_t$
   meta-RL must also *choose how to explore*!

\[
\theta^* = \arg \max_{\theta} E_{\pi_\theta} \left[ \sum_{t=0}^{T} r(s_t, a_t) \right]
\]

optimizing total reward over the entire meta-episode with RNN policy *automatically* learns to explore!
Meta-RL with recurrent policies

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

where \( \phi_i = f_{\theta}(M_i) \)


Architectures for meta-RL

- Standard RNN (LSTM) architecture

- Attention + temporal convolution

- Parallel permutation-invariant context encoder
Gradient-Based Meta-Learning
is pretraining a *type* of meta-learning?
better features = faster learning of new task!
Meta-RL as an optimization problem

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

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

what if \( f_\theta(\mathcal{M}_i) \) is itself an RL algorithm?

\[ f_\theta(\mathcal{M}_i) = \theta + \alpha \nabla_\theta J_i(\theta) \]

requires interacting with \( \mathcal{M}_i \)
to estimate \( \nabla_\theta E_{\pi_\theta}[R(\tau)] \)

1. improve policy with experience from \( \mathcal{M}_i \)
\[ \{(s_1, a_1, s_2, r_1), \ldots, (s_T, a_T, s_{T+1}, r_T)\} \]

standard RL:

\[ \theta^* = \arg \max_{\theta} E_{\pi_\theta(\tau)}[R(\tau)] \]
\[ J(\theta) \]
\[ \theta^{k+1} \leftarrow \theta_k + \alpha \nabla_{\theta^k} J(\theta^k) \]

model-agnostic meta-learning (MAML)

MAML for RL in pictures

\[ \theta \leftarrow \theta + \alpha \nabla_{\theta} J(\theta) \]

\[ \theta \leftarrow \theta + \beta \sum_{i} \nabla_{\theta} J_i[\theta + \alpha \nabla_{\theta} J_i(\theta)] \]
What did we just do??

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

supervised meta-learning: $f(D^{tr}, x) \rightarrow y$

model-agnostic meta-learning: $f_{\text{MAML}}(D^{tr}, x) \rightarrow y$

$f_{\text{MAML}}(D^{tr}, x) = f_{\theta'}(x)$

$\theta' = \theta - \alpha \sum_{(x, y) \in D^{tr}} \nabla_{\theta} \mathcal{L}(f_{\theta}(x), y)$

Just another computation graph...
Can implement with any autodiff package (e.g., TensorFlow)
But has favorable inductive bias...
MAML for RL in videos

after MAML training

after 1 gradient step
(forward reward)

after 1 gradient step
(backward reward)
More on MAML/gradient-based meta-learning for RL

MAML meta-policy gradient estimators:
• Foerster, Farquhar, Al-Shedivat, Rocktaschel, Xing, Whiteson. DiCE: The Infinitely Differentiable Monte Carlo Estimator.

Improving exploration:

Hybrid algorithms (not necessarily gradient-based):
• Houthooft, Chen, Isola, Stadie, Wolski, Ho, Abbeel. Evolved Policy Gradients.
Meta-RL as a POMDP
Meta-RL as... partially observed RL?

$\mathcal{M} = \{S, A, O, P, E, r\}$

$O$ – observation space, observations $o \in O$ (discrete or continuous)

$E$ – emission probability $p(o_t|s_t)$

Policy must act on observations $o_t$!

$\pi_\theta(a|o)$

Typically requires either:

- Explicit state estimation, i.e. to estimate $p(s_t|o_{1:t})$
- Policies with memory
Meta-RL as... partially observed RL?

\[ \tilde{s} \]
\[ \pi_{\theta}(a | s, z) \]

encapsulates information policy
needs to solve current task

learning a task = inferring \( z \)
from context \((s_1, a_1, s_2, r_1), (s_2, a_2, s_3, r_2), \ldots\)

this is just a POMDP!

before: \( \mathcal{M} = \{S, A, \mathcal{P}, r\} \)

now: \( \tilde{\mathcal{M}} = \{\tilde{S}, A, \tilde{\mathcal{O}}, \tilde{\mathcal{P}}, \mathcal{E}, r\} \)

\[ \tilde{S} = S \times \mathcal{Z} \quad \tilde{s} = (s, z) \]
\[ \tilde{\mathcal{O}} = S \quad \tilde{o} = s \]

**key idea:** solving the POMDP \( \tilde{\mathcal{M}} \) is equivalent to meta-learning!
Meta-RL as... partially observed RL?

\[ \pi_\theta(a|s, z) \]

encapsulates information policy needs to solve current task

learning a task = inferring \( z \)

from context \((s_1, a_1, s_2, r_1), (s_2, a_2, s_3, r_2), \ldots\)

exploring via posterior sampling with latent context

1. sample \( z \sim \hat{p}(z_t|s_{1:t}, a_{1:t}, r_{1:t}) \)
2. act according to \( \pi_\theta(a|s, z) \) to collect more data

act as though \( z \) was correct!

this is just a POMDP!

typically requires \textit{either}:

explicit state estimation, i.e. to estimate \( p(s_t|o_{1:t}) \)
policies with memory

need to estimate \( p(z_t|s_{1:t}, a_{1:t}, r_{1:t}) \)

this is \textit{not} optimal!

why?

but it’s pretty good, both in theory and in practice!

See, e.g. Russo, Roy. \textit{Learning to Optimize via Posterior Sampling}. 
Variational inference for meta-RL

policy: $\pi_\theta(a_t|s_t, z_t)$

inference network: $q_\phi(z_t|s_1, a_1, r_1, \ldots, s_t, a_t, r_t)$

$$(\theta, \phi) = \arg\max_{\theta, \phi} \frac{1}{N} \sum_{i=1}^{n} E_{z \sim q_\phi, \tau \sim \pi_\theta} [R_i(\tau) - D_{KL}(q(z|\ldots)||p(z))]$$

maximize post-update reward (same as standard meta-RL)

stay close to prior

conceptually very similar to RNN meta-RL, but with stochastic $z$

stochastic $z$ enables exploration via posterior sampling

---

Specific instantiation: PEARL

policy: \( \pi_\theta(a_t|s_t, z_t) \)

inference network: \( q_\phi(z_t|s_1, a_1, r_1, \ldots, s_t, a_t, r_t) \)

\[
(\theta, \phi) = \arg \max_{\theta, \phi} \frac{1}{N} \sum_{i=1}^{n} E_{z \sim q_\phi, \tau \sim \pi_\theta} [R_i(\tau) - D_{KL}(q(z|\ldots)||p(z))] 
\]

perform maximization using soft actor-critic (SAC), state-of-the-art off-policy RL algorithm

References on meta-RL, inference, and POMDPs


• Zintgraf, Igl, Shiarlis, Mahajan, Hofmann, Whiteson. **Variational Task Embeddings for Fast Adaptation in Deep Reinforcement Learning.**

• Humplik, Galashov, Hasenclever, Ortega, Teh, Heess. **Meta reinforcement learning as task inference.**
The three perspectives on meta-RL

Perspective 1: just RNN it

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

where \( \phi_i = f_\theta(M_i) \)

what should \( f_\theta(M_i) \) do?

1. improve policy with experience from \( M_i \)
   \[ \{(s_1, a_1, s_2, r_1), \ldots, (s_T, a_T, s_{T+1}, r_T)\} \]

2. (new in RL): choose how to interact, i.e. choose \( a_t \)

meta-RL must also choose how to explore!

Perspective 2: bi-level optimization

\[ f_\theta(M_i) = \theta + \alpha \nabla_\theta J_i(\theta) \]

MAML for RL

Perspective 3: it’s an inference problem!

\[ \pi_\theta(a|s, z) \quad z_t \sim p(z_t|s_{1:t}, a_{1:t}, r_{1:t}) \]

everything needed to solve task
The three perspectives on meta-RL

Perspective 1: just RNN it

\[ \theta^* \]

\[ \{ s_1, a_1, s_2, r_1 \} \rightarrow \{ s_2, a_2, s_3, r_2 \} \rightarrow \{ s_3, a_3, r_3 \} \]

+ conceptually simple
+ relatively easy to apply
- vulnerable to meta-overfitting
- challenging to optimize in practice

Perspective 2: bi-level optimization

\[ f_\theta(M_i) = \theta + \alpha \nabla_\theta J_i(\theta) \]

MAML for RL

+ good extrapolation ("consistent")
+ conceptually elegant
- complex, requires many samples

Perspective 3: it’s an inference problem!

\[ \pi_\theta(a|s,z) \]

\[ \pi_\theta(z_t|s_{1:t}, a_{1:t}, r_{1:t}) \]

everything needed to solve task

+ simple, effective exploration via posterior sampling
+ elegant reduction to solving a special POMDP
- vulnerable to meta-overfitting
- challenging to optimize in practice
But they’re not that different!

Perspective 1: just RNN it

\[ \theta^* \]

\[ (s_1, a_1, s_2, r_1), (s_2, a_2, s_3, r_2), (s_3, a_3, s_4, r_3) \]

\[ h_t \]

\[ a \]

\[ s \]

just perspective 1, but with stochastic hidden variables!

i.e., \( \phi = z \)

Perspective 2: bi-level optimization

\[ f_\theta(M_i) = \theta + \alpha \nabla_\theta J_i(\theta) \]

MAML for RL

Perspective 3: it’s an inference problem!

\[ \pi_\theta(a | s, z) \]

\[ z_t \sim p(z_t | s_{1:t}, a_{1:t}, r_{1:t}) \]

everything needed to solve task

just a particular architecture choice for these
Humans and animals seemingly learn behaviors in a variety of ways:

- Highly efficient but (apparently) model-free RL
- Episodic recall
- Model-based RL
- Causal inference
- etc.

Perhaps each of these is a separate “algorithm” in the brain

But maybe these are all emergent phenomena resulting from meta-RL?

**Meta-RL and emergent phenomena**

- **meta-RL gives rise to episodic learning**

  ![Diagram of episodic learning](image1)


- **model-free meta-RL gives rise to model-based adaptation**

  ![Diagram of model-based adaptation](image2)


- **meta-RL gives rise to causal reasoning (!)**

  ![Diagram of causal reasoning](image3)