META REINFORCEMENT LEARNING

→ Through previous experience (motivation of Meta RL)
→ Data efficiency (with a few examples only)

- How might you get a machine to accomplish this task?
  → Provide model of image formation, etc.
  → Fewer human priors, more data-driven priors: Greater success.

Meta-RL Problem Formulation and Examples

META-LEARNING PROBLEM:

- In Supervised Learning: Inputs: x Outputs: y DATA: $\{(x,y)_i\}$

  $y = f(x;\theta)$

- In Meta Supervised Learning: Inputs: $\mathcal{D}_{train}$ and x_{test} Outputs: y_{test} DATA: $\{\mathcal{D}_i\}$

  $\mathcal{D}_{train}$: $\{(x,y)_{1:k}\}$

  $y_{test} = f(\mathcal{D}_{train}, x_{test}; \theta)$

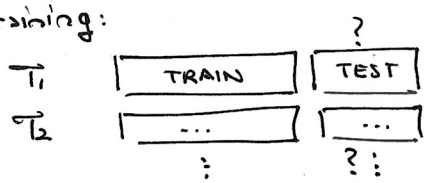
  ↳ diff. datasets (tasks)

  $\mathcal{D}_i$: $\{(x,y)_j\}$

EXAMPLE: FEW-SHOT CLASSIFICATION

- Given 1 example of 5 classes: we want to classify new examples. (x_{test})
  ($\mathcal{D}_{train}$)

- Meta-Training:



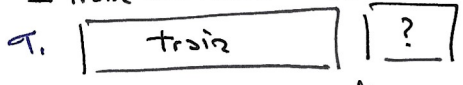
META-TESTING ↑

META-TRAINING. ↓

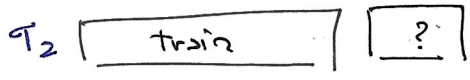
↳ which updates θ function (the parameters)

META-TRAINING

- Train dataset #1: Cat, Bird -

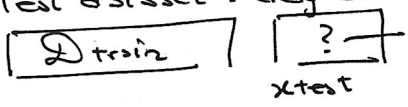


- Train dataset #2: flower, bike -



META-TESTING

- Test dataset: dog, otter -



In one gradient step

- In Reinforcement Learning: Inputs: s_t Outputs: a_t DATA: $\{(s_t, a_t, r_t, s_{t+1})\}$

  $a_t = \pi(s_t;\theta)$

- In Meta RL: Inputs: $\mathcal{D}_{train} \oplus s_t$ Outputs: a_t DATA: $\{\mathcal{D}_i\}$

  k-rollouts from π

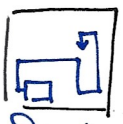
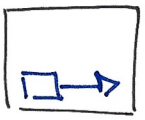
  $a_t = f(\mathcal{D}_{train}, s_t; \theta)$

  dataset of datasets collected for each task.

⇒ Design and Optimization of the f function

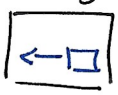
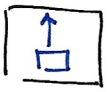
⇒ Collecting Appropriate Data (Learning to Explore - collect good $\mathcal{D}_{train}$)

EXAMPLE: MAZE NAVIGATION

- Given a small amount of experience  $\mathcal{D}_{train}$ ⇒ Learn to solve the task  $s_t \rightarrow a_t$

⇕

By learning how to solve many other tasks... (other diff. mazes)

  ...

- For $\mathcal{D}_{train}$: (episodes)
  - ↳ k rollouts from π ⇒ EPISODIC VARIANT
  - ↳ $1, \dots, k$ timesteps from π ⇒ ONLINE VARIANT.

QUESTIONS.

1. Meta-Learning vs Transfer Learning
   - → Optimize for good transfer performance (new task learnt differently) ↳ Meta-Learning.
   - → Similar Motivations: both want to understand how to learn and transfer a representation that will help with learning new skills.
   - → Th: Accomplish this goal without any additional training. ↳ less formal for generalization process.
   - → NL: Explicitly says that you can gather little more data on the new task to update the model (quickly)

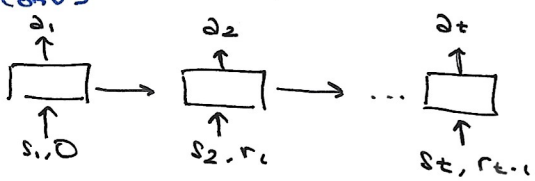
Method Classes

Design of the f function:

BLACK-BOX METHOD $\mathcal{D}_{train}, s_t \rightarrow a_t$

→ Recurrent Network (LSTM, NTM, Conv)

$a_t = f(\mathcal{D}_{train}, s_t; \theta)$



⇒ diff compared to recurrent policy?

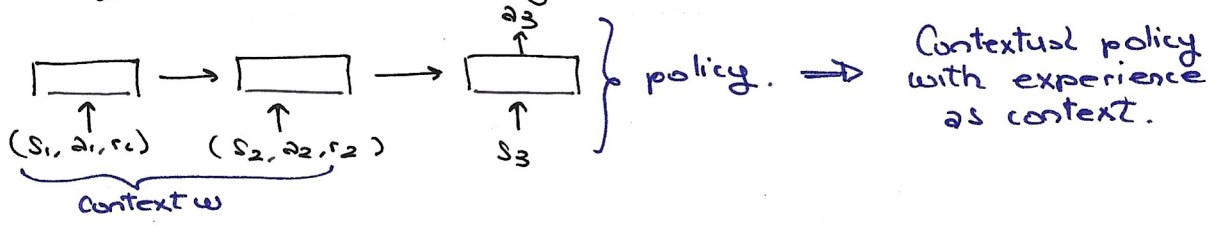
⇒ Hidden state maintained across episodes within a task.

- +: general + expressive
- +: variety of design choices in Architecture
- −: complex model for complex task of learning
- −: Impractical data requirements.

PAPER: Simple Neural Attentive Meta-Learner

- Experiment: learning to visually navigate a maze
  - → train on 1000 small mazes
  - → test on held-out small mazes and large mazes.

- Digression: Connection to Contextual Policies.
  - Contextual policy: $\pi_\theta(a|s,\omega)$
    - ↳ ω: stack location, walking direction, etc. (context)



What about goal-conditioned policies/value functions?

- Rewards: strict generalization of goals.
- Meta-RL: Adapt new tasks (META-LEARNING (k-shot)) vs. generalize new tasks (CLASSIC LEARNING (0-shot))

OPTIMIZATION-BASED APPROACHES

Fine-Tuning [test-time]

$$\phi_j \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}^j_{train}(\theta)$$

(θ: pretrained parameters; $\mathcal{L}^j_{train}$: training data for new task.)

OUR METHOD

$$\min_\theta \sum_{task\,i} \mathcal{L}^i_{test}(\theta - \alpha \nabla_\theta \mathcal{L}^i_{train}(\theta))$$

↳ in your meta-training set

↳ Embedded gradient descent in Meta-Learning

↳ ∴ Train over many tasks, to learn parameter vector θ that transfers.

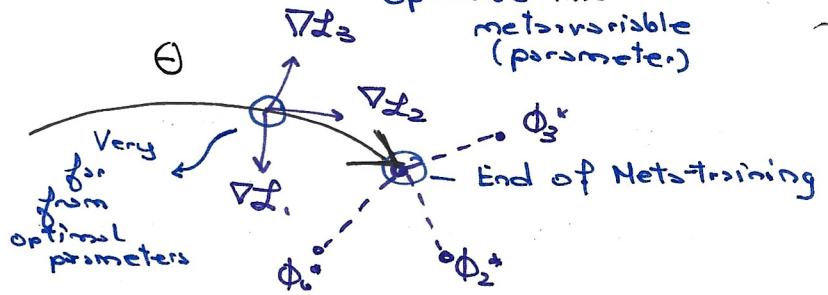
- Suppose that:
  - → θ: parameter vector being meta-learned
  - → ϕ_i^*: optimal parameter vector for task i

Meta-Learning ↑ training / Adaptation ↑

$$\min_\theta \sum_{\text{task } i} \mathcal{L}^i_{\text{test}}\left(\theta - \alpha \nabla_\theta \mathcal{L}^i_{\text{train}}(\theta)\right)$$

↳ Optimize this meta-variable (parameter)

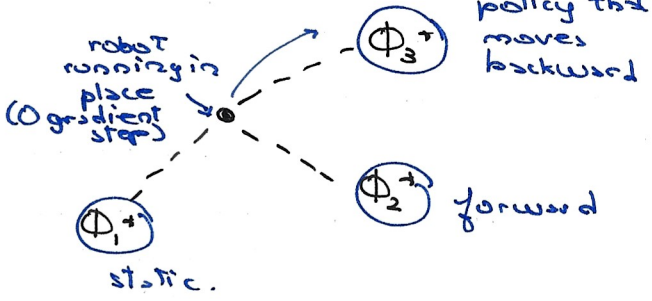
{ ▬ : Meta-Learning. -- : Learning / Adaptation }



↳ Model-Agnostic Meta-Learning ↳

MAML

Example: At the end of meta-training



MAML and Model-Based RL

- Adapt to diff. dynamics in environments.

↳ Ex: on variable terrains

META-TRAIN A PRIOR θ_*

ONLINE ADAPTATION

Store recent history
Last N timesteps
$\{s_{t-N:t}, a_{t-N:t}\}$

(a_t, s_{t+1})

Take Action at via MPC

$\hat{e} \sim \hat{p}_{\theta_*}$

$\hat{p}_{\theta'_*}$

Model Adaptation

$\theta'^* = u_\psi(s_{t-N:t}, a_{t-N:t}, \theta^*)$

One step of gradient descent on learned dynamics model

$\hat{p}_{\theta'^*_*}$ cheetah

$\hat{p}_{\theta_*}$

Ex: META-TRAIN on variable terrains
META-TEST with slope, missing leg, payload, calibration errors.

To summarize...

- Black-Box Method: $a_t = f(\mathcal{D}_{\text{train}}, s_t; \theta)$
- Optimization-Based Method: $a_t = f(s_t; \theta - \alpha \nabla_\theta \mathcal{L}(\mathcal{D}_{\text{train}}))$
  - ↳
    - +: inductive bias of SGD built in
    - +: model-agnostic (plug and play with existing archi)
    - −: RL gradients ∅ informative (policy gradients, Bellman errors)

$\theta \rightarrow \nabla_\theta \mathcal{L} \rightarrow \theta'$

s_t, a_t, r_t, s_{t+1}

Learning to Explore

- How should we collect $\mathcal{D}_{train}$?

↳ Coupling problem ↳ Execution Exploration

#0 Optimize for Exploration + Execution End-to-End w.r.t Reward

⊕ · simple
· leads to optimal strategy in principle

⊖ · Challenging optimization when exploration is hard

Example of hard Meta-RL Problem (Exploration):

- Learned cooking tasks in previous kitchens (META-TRAINING)
- Goal: quickly learn tasks in a new kitchen. (META-TESTING → 1 exploration episode → 1 execution episode)

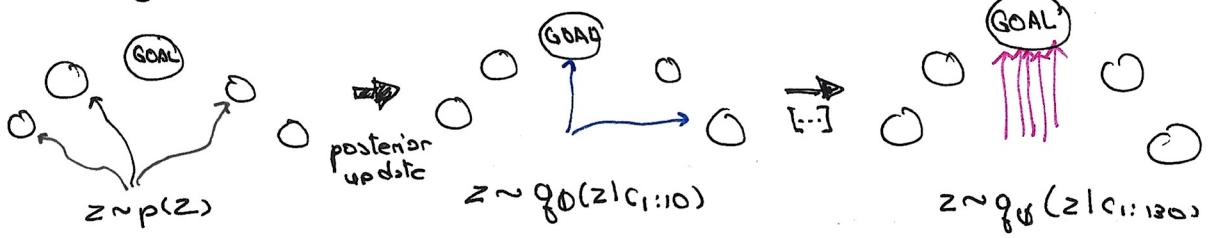
#1 Leverage Alternative Exploration Strategies.

a) Posterior sampling (Thompson sampling)

↳ data you've collected so far.

→ Learn dist. over latent task variable $p(z)$, $q(z|\mathcal{D}_{train})$ and corresponding task policies $\pi(a|s,z)$.

→ Sample z from current "posterior" and sample from policy $\pi(a|s,z)$



⇒ Posterior sampling is bad when...

⊖ Goals for sway and signs on wall that tells you the correct goal.

b) Use instrinsic rewards

c) Tasks dynamics and reward prediction

→ Train model $f(s', r|s, a, \mathcal{D}_{train})$

→ Collect $\mathcal{D}_{train}$ so that model is accurate.

Bad when...

⊖ Lots of distractors, or complex, high-dim state dynamics.

⊕ · Easy to optimize
· Many based on principled strategies

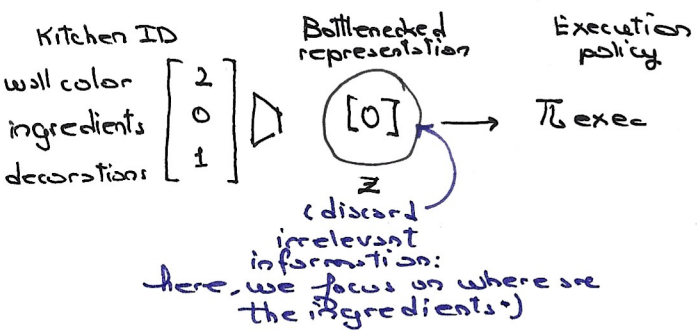
⊖ · suboptimal by arbitrarily large amount in some environments.

#2: Decouple by acquiring representation of task relevant information

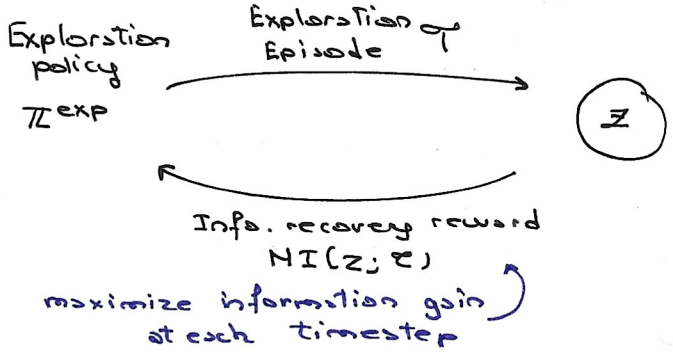
⊕: · Leads to optimal strategy in principle
· Easy to optimize in practice
⊖: · Requires task identifier (sign)

META-TRAINING:

1) Learn execution ⊕ identify key information.



2) Learn to explore by recovering that information.



META-TESTING:

$$\{\pi^{exp} \xrightarrow{\text{Exploration Ep. } \tau} \pi^{exec}\}$$

↳ Decoupled Reward-free ExplorAtion and Execution in Meta-RL (DREAN) ↵

⇒ Consistent with End-To-End Approach.

Example: Sparse Reward 3D Visual Navigation Problem

→ Task: go to the (key / block / ball) of color specified by the sign
→ Agent starts on other side of barrier, must walk around to read the sign
→ Pixel observations (80×60 RGB)
→ Sparse Binary Reward

To summarize...

→ End-To-End
→ Alternative Strategies
→ Decoupled Exp. and Exec (DREAM)

Challenges and Latest Developments.

1. Adopting to entirely new tasks. (meta-world benchmark?)
2. Robustness to out of distribution tasks.
3. Where do the tasks even come from?
4. Better RL Algorithms.