CS-583: Deep Learning Reinforcement Learning - Transfer Learning

Abdul Rafae Khan
Department of Computer Science
Stevens Institute of Technology
akhan4@stevens.edu

November 27, 2023

What's the problem?

this is easy (mostly)

Why?

this is impossible

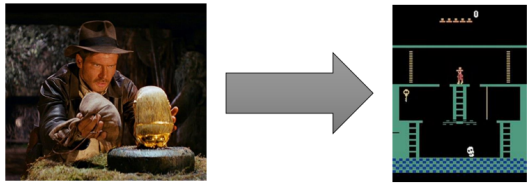
Montezuma's revenge

- 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?



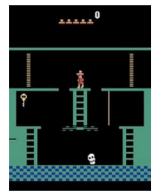
- If we've solved prior tasks, we might acquire useful knowledge for solving a new task
- How is the knowledge stored?
- Q-function: tells us which actions or states are good
- Policy: tells us which actions are potentially useful
  - some actions are never useful!
- Models: what are the laws of physics that govern the world?
- Features/hidden states: provide us with a good representation

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



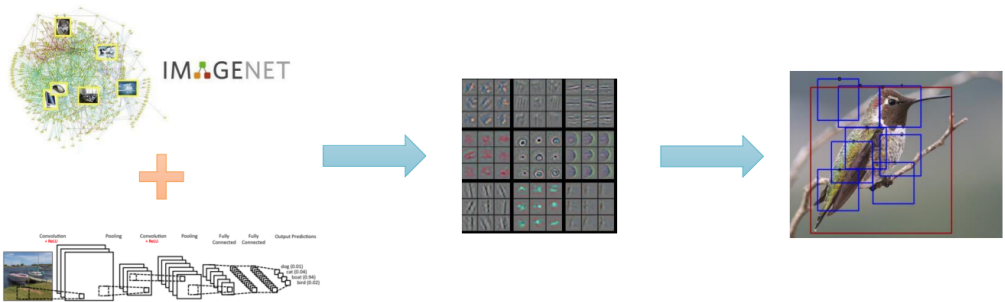
- "shot": number of attempts in the target domain
- 0-shot: just run a policy trained in the source domain
- 1-shot: try the task once
- few shot: try the task a few times

How can we frame transfer learning problems?

1. Forward transfer: learn policies that transfer effectively
   - Train on source task, then run on target task (or finetune)
   - Relies on the tasks being quite similar!
2. Multi-task transfer: train on many tasks, transfer to a new task
   - Sharing representations and layers across tasks in multi-task learning
   - New task needs to be similar to the distribution of training tasks
3. Meta-learning: learn to learn on many tasks
   - 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

train here

z

dog (0.01)
cat (0.04)
boat (0.94)
bird (0.02)

correct answer

reversed gradient

(same network)

can we force this layer to be **invariant** to domain?

$D_\phi(z) = \text{domain}$

domain classifier:
guess domain from z

dog (0.01)
cat (0.04)
boat (0.94)
bird (0.02)

incorrect answer

z

do well here

Is this true?

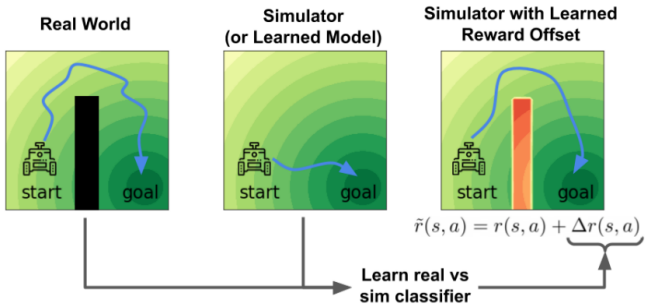
Invariance assumption: everything that is **different** between domains is **irrelevant**

formally:

$p(x)$ is different exists some $z = f(x)$ such that $p(y|z) = p(y|x)$, but $p(z)$ is same

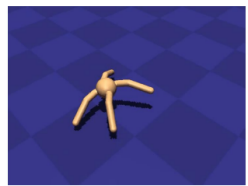
Domain adaptation in RL for dynamics?

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



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

$$\begin{aligned}\Delta r(s_t, a_t, s_{t+1}) = &\log p(\text{target} \mid s_t, a_t, s_{t+1}) - \log p(\text{target} \mid s_t, a_t) \\ &- \log p(\text{source} \mid s_t, a_t, s_{t+1}) + \log p(\text{source} \mid s_t, a_t)\end{aligned}$$



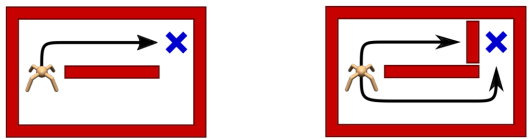
When might this **not** work?

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

Finetuning for diversity



How can we increase diversity?

- Act as randomly as possible while collecting high rewards
- Learning to solve a task in all possible ways provides for more robust transfer
- Manipulate the source domain
  - So far, source domain (e.g., empty room) and target domain (e.g., corridor) are fixed
  - We can **design** the source domain, to handle a **difficult** target domain
    - Often the case for simulation to real world transfer

Can we learn faster by learning multiple tasks?





learn learn learn learn learn

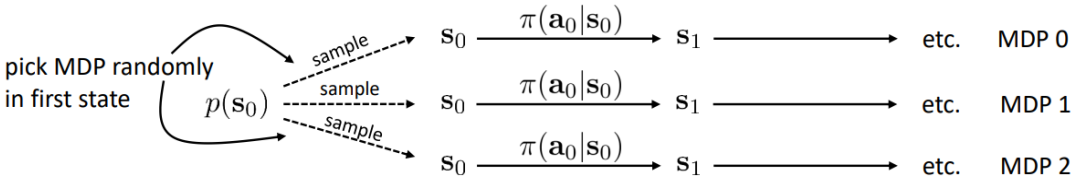
learn

Multi-task learning can:

- Accelerate learning of all tasks that are learned together
- Provide better pre-training for down-stream tasks

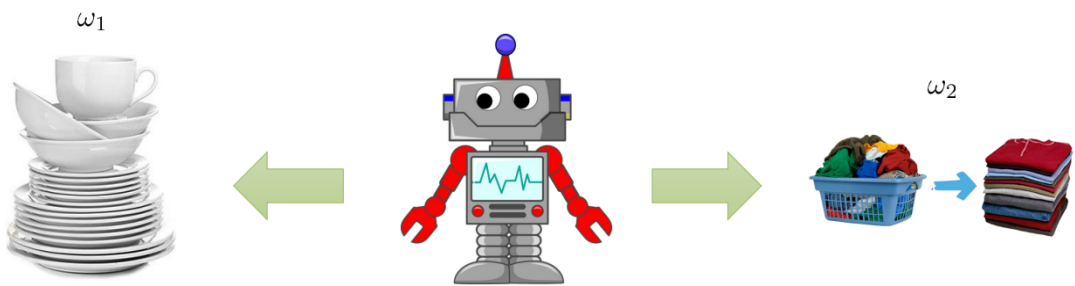
Can we solve multiple tasks at once?

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?

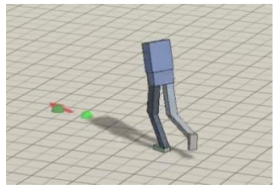


Contextual policies

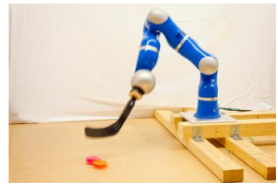
standard policy: $\pi_\theta(a|s)$
contextual policy: $\pi_\theta(a|s, \omega)$
ω is the task, e.g. doing dishes vs laundry

formally, simply defines augmented state space: $\tilde{s} = \begin{bmatrix} s \\ \omega \end{bmatrix}$ $\quad \tilde{\mathcal{S}} = \mathcal{S} \times \Omega$

ω: stack location



ω: walking direction



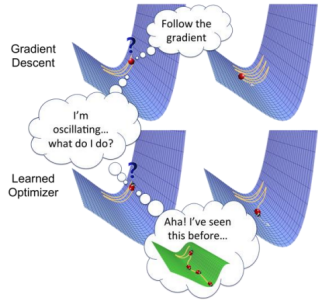
ω: where to hit puck

images: Peng, van de Panne, Peters

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



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

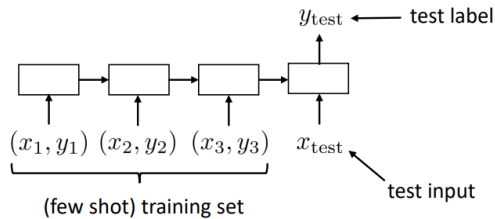
Meta-learning with supervised learning

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

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

- How to read in training set?
  - Many options, e.g. RNNs

What is being "learned"?



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

"Generic" learning:

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

"Generic" meta-learning:

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

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

What is being "learned"?

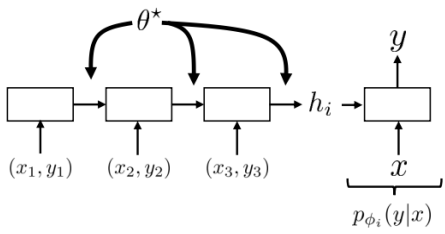
"Generic" learning:

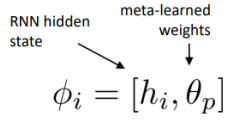
$$\theta^* = \arg\min_\theta \mathcal{L}(\theta, \mathcal{D}^{tr})$$
$$= f_{learn}(\mathcal{D}^{tr})$$

"Generic" meta-learning:

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

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





Meta Reinforcement Learning

"Generic" learning:

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

"Generic" meta-learning:

$$\theta^* = \arg\min_\theta \sum_{i=1}^{n} \mathcal{L}(\phi_i, \mathcal{D}_i^{ts})$$
where $\phi_i = f_\theta(\mathcal{D}_i^{tr})$

Reinforcement learning:

$$\theta^* = \arg\max_\theta E_{\pi_\theta(\tau)}[R(\tau)]$$
$$= f_{RL}(\mathcal{M})$$

$\mathcal{M}$ = MDP

Meta-reinforcement learning:

$$\theta^* = \arg\max_\theta \sum_{i=1}^{n} E_{\pi_{\phi_i}(\tau)}[R(\tau)])$$
where $\phi_i = f_\theta(\mathcal{M}_i)$

$\mathcal{M}_i$ = MDP for task i

Meta Reinforcement Learning

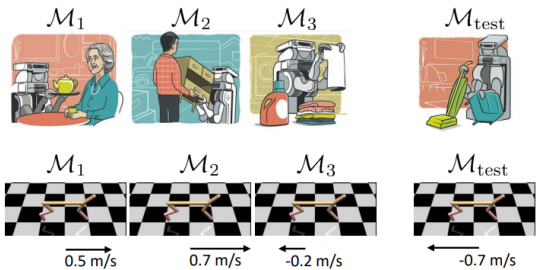
$\theta^* = \arg\max_\theta \sum_{i=1}^{n} E_{\pi_{\phi_i}(\tau)}[R(\tau)])$
where $\phi_i = f_\theta(\mathcal{M}_i)$

assumption: $\mathcal{M}_i \sim p(\mathcal{M})$

meta-test time:
sample $\mathcal{M}_{test} \sim p(\mathcal{M})$, get $\phi_i = f_\theta(\mathcal{M}_{test})$

meta-training MPDs: $\{\mathcal{M}_1, \cdots, \mathcal{M}_n\}$

Some examples:

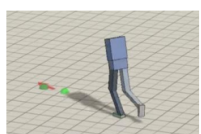


Contextual policies and meta-learning

In meta-RL, the context is inferred from the experience from $\mathcal{M}_i$ $\pi_\theta(a_t|s_t, \phi_i)$
ϕ_i is the 'context'

In multi-task RL, the context is typically given

ϕ: stack location

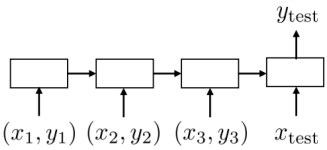


ϕ: walking direction

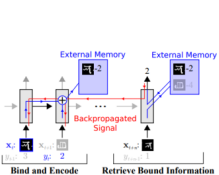
ϕ: where to hit puck

Meta-Learning Methods

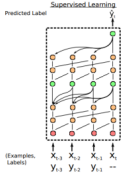
black-box meta-learning



some kind of network that can read in an entire (few-shot) training set

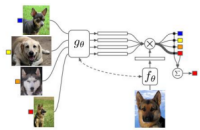


Santoro et al. **Meta-Learning with Memory-Augmented Neural Networks**. 2016.

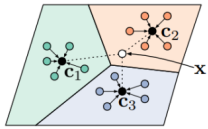


Mishra et al. **A Simple Neural Attentive Meta-Learner**. 2018.

non-parametric meta-learning

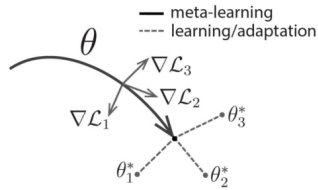


Vinyals et al. **Matching Networks for One Shot Learning**. 2017.



Snell et al. **Prototypical Networks for Few-shot Learning**. 2018.

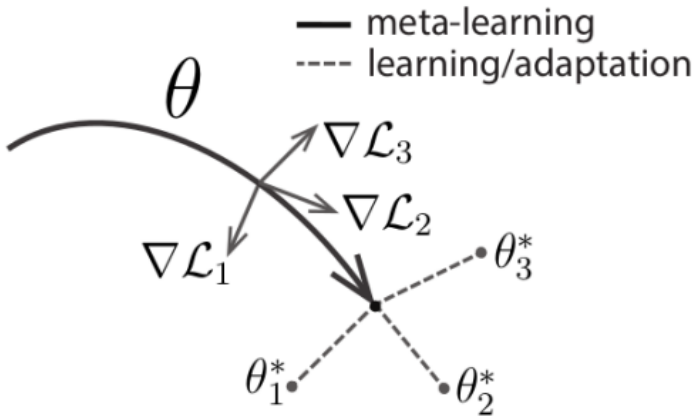
gradient-based meta-learning



Finn et al. **Model-Agnostic Meta-Learning**. 2018.

Model-Agnostic Meta-Learning (MAML)

Find model parameters that are sensitive to changes in the task
Gradient-based Meta-Learning



MAML

- How to use pretrained model:
  - Fine-tune (by gradient descent)

$$\theta \leftarrow \theta - \alpha \nabla_\theta L_{train}(\theta)$$

- What is our goal:
  - Easy to fine-tune for any tasks
  - A meta loss function

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

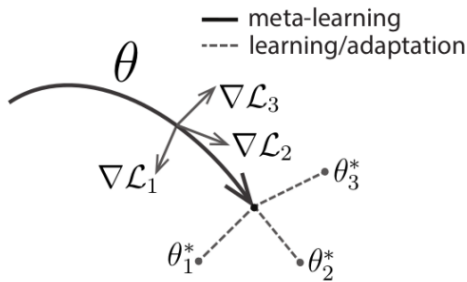
MAML

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**

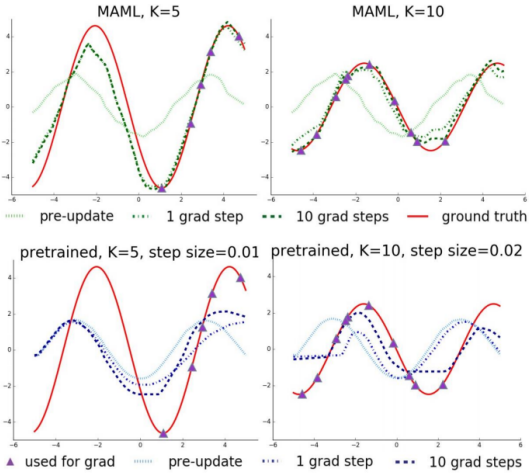
MAML

Inner Loop: Update the model for a task from an initialization
Outer Loop: Optimize for the performance of all inner loop models on all tasks
Intuition: We want achieve a low loss after only a few updates on a task



MAML (Regression)

The regressor is a neural network model
with 2 hidden layers of size 40 with ReLU non-linearities



MAML (Handwriting Recognition)

20 instances of 1623 characters from 50 different alphabets

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%**

The End