Learning to Learn for Robotic Control

Pieter Abbeel

Embodied Intelligence  AI for robotic automation
UC Berkeley  AI research
Gradescope  AI for grading homework and exams

Research in this talk was done at OpenAI and UC Berkeley
Humans vs. DDQN

Humans after 15 minutes tend to outperform DDQN after 115 hours

Black dots: human play
Blue curve: mean of human play
Blue dashed line: ‘expert’ human play

Red dashed lines:
DDQN after 10, 25, 200M frames
(~ 46, 115, 920 hours)

[Tsividis, Pouncy, Xu, Tenenbaum, Gershman, 2017]
How to bridge this gap?
environments (how much they measure / incentivise general intelligence)
more multi-agent / non-stationary / real-world-like.

RL

Reality

Digital worlds
(complex multi-agent envs)

MuJoCo/ATARI
/Universe
(~few dozen envs)

Cartpole etc.
(and bandits, gridworld, ...few toy tasks)

BlocksWorld
(SHRDLU etc)

Hard Coded
(LISP programs, no learning)

Value Iteration etc.
(~discrete MDPs, linear function approximators)

DQN, PG
(deep nets, hard-coded various tricks)

RL^2
(Learn the RL algorithm. structure fixed.)

CodeGen
(learn structure and learning algorithm)

agents

(how impressive they are)
more learning.
more compute.

Zone of “not going to happen.”

[Slide adapted from Andrej Karpathy]
environments (how much they measure / incentivise general intelligence)
more multi-agent / non-stationary / real-world-like.

RL

Reality

Digital worlds
(complex multi-agent envs)
(simple multi-agent envs)

MuJoCo/ATARI
/Universe
(~few dozen envs)

Cartpole etc.
(and bandits, gridworld, ...few toy tasks)

BlocksWorld
(SHRDLU etc)

Hard Coded
(LISP programs, no learning)

Value Iteration etc.
(~discrete MDPs, linear function approximators)

DQN, PG
(deep nets, hard-coded various tricks)

RL^2
(Learn the RL algorithm. structure fixed.)

CodeGen
(learn structure and learning algorithm)

agents

2013
2016

Zone of “not going to happen.”

[Slide adapted from Andrej Karpathy]
Meta Learning for Optimization

Task distribution: different neural networks, weight initializations, and/or different loss functions

- Bengio et al., (1990) Learning a synaptic learning rule
- Hochreiter et al., (2001) Learning to learn using gradient descent
- Younger et al., (2001), Meta learning with back propagation
- Andrychowicz et al., (2016) Learning to learn by gradient descent by gradient descent
- Chen et al., (2016) Learning to Learn for Global Optimization of Black Box Functions
- 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 Classification

**Task distribution: different classification datasets (input: images, output: class labels)**

- Hochreiter et al., (2001) Learning to learn using gradient descent
- Younger et al., (2001), Meta learning with back propagation
- Vinyals et al., (2016) Matching networks for one shot learning
- Edwards et al., (2016) Towards a Neural Statistician
- Ravi et al., (2017) Optimization as a model for few-shot learning
- Munkhdalai et al., (2017) Meta Networks
- Snell et al., (2017) Prototypical Networks for Few-shot Learning
- Shyam et al., (2017) Attentive Recurrent Comparators
- Mehrotra et al., (2017) Generative Adversarial Residual Pairwise Networks for One Shot Learning
- Finn and Levine, (2017) Meta-Learning and Universality: Deep Representations and Gradient Descent can Approximate any Learning Algorithm
- Anon@OpenReview, (2017) Recasting Gradient-Based Meta-Learning as Hierarchical Bayes
Meta Learning for Generative Models

Task distribution: different unsupervised datasets (e.g. collection of images)

- Rezende et al., (2016) One-Shot Generalization in Deep Generative Models
- Edwards et al., (2016) Towards a Neural Statistician
- Bartunov et al., (2016) Fast Adaptation in Generative Models with Generative Matching Networks
- Bornschein et al., (2017) Variational Memory Addressing in Generative Models
Meta-Learning for Control

- Learning to Reinforcement Learn
- Learning to Imitate
Reinforcement Learning

[Duan, Schulman, Chen, Bartlett, Sutskever, Abbeel, 2016] also: [Wang et al, 2016]
Reinforcement Learning

Reinforcement Learning

[Duan, Schulman, Chen, Bartlett, Sutskever, Abbeel, 2016] also: [Wang et al, 2016]
Reinforcement Learning

Traditional RL research:
• Human experts develop the RL algorithm
• After many years, still no RL algorithms nearly as good as humans...

Alternative:
• Could we learn a better RL algorithm?
• Or even learn a better entire agent?

[Ref. Duan, Schulman, Chen, Bartlett, Sutskever, Abbeel, 2016] also: [Wang et al, 2016]

[Revised by Pieter Abbeel -- embody.ai / UC Berkeley / Gradescope]
Meta-Reinforcement Learning

Meta-training environments

Environment A

Environment B

→ Meta RL Algorithm

→ "Fast" RL Agent

[Duan, Schulman, Chen, Bartlett, Sutskever, Abbeel, 2016] also: [Wang et al, 2016]

Pieter Abbeel -- embody.ai / UC Berkeley / Gradescope
Meta-Reinforcement Learning

Meta-training environments

Environment A

Environment B

…

Meta RL Algorithm

"Fast" RL Agent

Environment F

Testing environments

[Duan, Schulman, Chen, Bartlett, Sutskever, Abbeel, 2016] also: [Wang et al, 2016]

Pieter Abbeel -- embody.ai / UC Berkeley / Gradescope
Meta-Reinforcement Learning

Meta-training environments

Environment A

Environment B

Meta RL Algorithm

"Fast" RL Agent

Environment G

Testing environments

[Duan, Schulman, Chen, Bartlett, Sutskever, Abbeel, 2016] also: [Wang et al, 2016]
Meta-Reinforcement Learning

Meta-training environments

Environment A

Environment B

Meta RL Algorithm

"Fast" RL Agent

Environment H

Testing environments

Formalizing Learning to Reinforcement Learn

$$\max_{\theta} \mathbb{E}_M \mathbb{E}_{\tau^{(k)}_M} \left[ \sum_{k=1}^{K} R(\tau^{(k)}_M) \mid \text{RLagent}_\theta \right]$$

$M$ : sample environment

$\tau^{(k)}_M$ : $k'$th episode in environment $M$
Formalizing Learning to Reinforcement Learn

\[
\max_{\theta} \mathbb{E}_M \mathbb{E}_{\tau_M^{(k)}} \left[ \sum_{k=1}^{K} R(\tau_M^{(k)}) \mid \text{RLagent}_\theta \right] \\
M : \text{sample MDP}
\]

\[
\tau_M^{(k)} : k\text{'th trajectory in MDP } M
\]

Meta-train:

\[
\max_{\theta} \sum_{M \in M_{\text{train}}} \mathbb{E}_{\tau_M^{(k)}} \left[ \sum_{k=1}^{K} R(\tau_M^{(k)}) \mid \text{RLagent}_\theta \right] \\
\]

[Duang, Schulman, Chen, Bartlett, Sutskever, Abbeel, 2016] also: [Wang et al, 2016]
Representing $\text{RLAgent}_\theta$: RL2

$$\max_\theta \sum_{M \in M_{\text{train}}} \mathbb{E}_{\tau_M^{(k)}} \left[ \sum_{k=1}^{K} R(\tau_M^{(k)}) \right] \mid \text{RLAgent}_\theta$$

- $\text{RLAgent} = \text{RNN} = \text{generic computation architecture}$
  - different weights in the RNN means different RL algorithm and prior
  - different activations in the RNN means different current policy
  - meta-train objective can be optimized with an existing (slow) RL algorithm

[Duan, Schulman, Chen, Bartlett, Sutskever, Abbeel, 2016] also: [Wang et al, 2016]
Representing $RL_{agent_\theta}$: SNAIL

- Like RL2 but:
  - replace the LSTM with dilated temporal convolution (like wavenet)
  - + attention

[Wavenet: van den Oord et al, 2016]
[Attention-is-all-you-need: Vaswani et al, 2017]
Representing $RLagent_{\theta}$: MAML

**Key idea:** End-to-end learning of parameter vector $\theta$ that is good init for fine-tuning for many tasks

MAML test time: fine-tuning:  
\[ \theta' \leftarrow \theta - \alpha \nabla_{\theta} \mathcal{L}_{tr}(\theta) \]

MAML training:  
\[ \min_{\theta} \sum_{\text{tasks}} \mathcal{L}_{val}(\theta - \alpha \nabla_{\theta} \mathcal{L}_{\text{train}}(\theta)) \]

[Fin, Abbeel, Levine ICML 2017]
Evaluation: Multi-Armed Bandits

- Multi-Armed Bandits setting
  - Each bandit has its own distribution over pay-outs
  - Each episode = choose 1 bandit
  - Good RL agent should explore bandits sufficiently, yet also exploit the good/best ones

- Provably (asymptotically) optimal RL algorithms have been invented by humans: Gittins index, UCB1, Thompson sampling, ...

[Duan, Schulman, Chen, Bartlett, Sutskever, Abbeel, 2016]
## Bandits

<table>
<thead>
<tr>
<th>Setup $(N, K)$</th>
<th>Gittins (optimal as $N \to \infty$)</th>
<th>Method</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Random</td>
<td>$RL^2$</td>
<td>MAML</td>
<td>SNAIL (ours)</td>
</tr>
<tr>
<td>10, 5</td>
<td>6.6</td>
<td>5.0</td>
<td>6.7</td>
<td>6.5 ± 0.1</td>
<td>6.6 ± 0.1</td>
</tr>
<tr>
<td>10, 10</td>
<td>6.6</td>
<td>5.0</td>
<td>6.7</td>
<td>6.6 ± 0.1</td>
<td>6.7 ± 0.1</td>
</tr>
<tr>
<td>10, 50</td>
<td>6.5</td>
<td>5.1</td>
<td>6.8</td>
<td>6.6 ± 0.1</td>
<td>6.7 ± 0.1</td>
</tr>
<tr>
<td>100, 5</td>
<td>78.3</td>
<td>49.9</td>
<td>78.7</td>
<td>67.1 ± 1.1</td>
<td>79.1 ± 1.0</td>
</tr>
<tr>
<td>100, 10</td>
<td>82.8</td>
<td>49.9</td>
<td>83.5</td>
<td>70.1 ± 0.6</td>
<td>83.5 ± 0.8</td>
</tr>
<tr>
<td>100, 50</td>
<td>85.2</td>
<td>49.8</td>
<td>84.9</td>
<td>70.3 ± 0.4</td>
<td>85.1 ± 0.6</td>
</tr>
<tr>
<td>500, 5</td>
<td>405.8</td>
<td>249.8</td>
<td>401.5</td>
<td>–</td>
<td>408.1 ± 4.9</td>
</tr>
<tr>
<td>500, 10</td>
<td>437.8</td>
<td>249.0</td>
<td>432.5</td>
<td>–</td>
<td>432.4 ± 3.5</td>
</tr>
<tr>
<td>500, 50</td>
<td>463.7</td>
<td>249.6</td>
<td>438.9</td>
<td>–</td>
<td>442.6 ± 2.5</td>
</tr>
<tr>
<td>1000, 50</td>
<td>944.1</td>
<td>499.8</td>
<td>847.43</td>
<td>–</td>
<td>889.8 ± 5.6</td>
</tr>
</tbody>
</table>

Evaluation: Locomotion – Half Cheetah

- Task – reward based on target running direction + speed

[Duan, Schulman, Chen, Bartlett, Sutskever, Abbeel, 2016]
Evaluation: Locomotion – Half Cheetah

- Task – reward based on target running direction + speed

- Result of meta-training = a single agent (the “fast RL agent”), which masters each task almost instantly within 1st episode
Task – reward based on target running direction + speed
Evaluation: Locomotion – Ant

- Task – reward based on target running direction + speed

- Result of meta-training = a single agent (the “fast RL agent”), which masters each task almost instantly within 1st episode
**Evaluation: Visual Navigation**

**Agent input:** current image

**Agent action:** straight / 2 degrees left / 2 degrees right

*Map just shown for our purposes, but not available to agent*

Agent Dropped in New Maze

Meta-Learning Shared Hierarchies

Goal: find subpolicies that enable fast learning of master policy $\theta$

[Frans, Ho, Chen, Abbeel, Schulman, 2017]
Meta-Learning Shared Hierarchies

**RL2 Meta-Learning Objective:**

\[
\max_{\theta} \mathbb{E}_M \mathbb{E}_{\tau_M^{(k)}} \left[ \sum_{k=1}^{K} R(\tau_M^{(k)}) \right] | \text{RLAgent}_\theta \\
\]

**MLSH Meta-Learning Objective:**

\[
\max_{\phi_{\theta_0}} \mathbb{E}_{\theta_0} \mathbb{E}_M \mathbb{E}_{\tau_M^{(k)}} \left[ \sum_{k=1}^{K} R(\tau_M^{(k)}) \right] | \phi, \text{RLAgent}_{\theta_0} \\
\]

= find a set of subpolicies that enable fast learning of the master policy
MLSH -- Experiment 1: Moving Bandits

Hope for
• Learned subpolicies: low level control for each of the targets
• High level policy: standard bandit problem

Episode Duration = 50, Subpolicy Duration = 10
Episode duration = 1000

Subpolicy duration = 200

Experiment 2: Maze Navigation
MLSH agent was trained on nine separate mazes. It discovered sub-policies for upwards, rightwards, and downwards movement.
Task distribution: different environments

- Wiering, Schmidhuber. Solving POMDPs with Levin search and EIRA. (1996)
- Zhao, Schmidhuber. Solving a complex prisoner’s dilemma with self-modifying policies. (1998)
- Duan et al., (2016) RL2: Fast Reinforcement Learning via Slow Reinforcement Learning
- Wang et al., (2016) Learning to Reinforcement Learn
- Mishra, Rohinenjad et al., (2017) Simple Neural AttentIve meta-Learner
Meta-Learning for Control

- Learning to Reinforcement Learn
- Learning to Imitate
Imitation Learning in Robotics

[Abbeel et al. 2008]  [Kolter et al. 2008]  [Ziebart et al. 2008]

[Schulman et al. 2013]  [Finn et al. 2016]
Imitation Learning

Task A
e.g. assemble a chair

Task B
e.g. assemble a table
One-Shot Imitation Learning

Many demonstrations for task A

Many demonstrations for task B

Meta Learning Algorithm

One-Shot Imitator (Neural network)

[Duan, Andrychowicz, Stadie, Ho, Schneider, Sutskever, Abbeel, Zaremba, 2017]
One-Shot Imitation Learning

[Image showing a diagram of the One-Shot Imitation Learning process.]

[References: Duan, Andrychowicz, Stadie, Ho, Schneider, Sutskever, Abbeel, Zaremba, 2017]

Pieter Abbeel -- embody.ai / UC Berkeley / Gradescope
One-Shot Imitation Learning

Many demonstrations for task A

Many demonstrations for task B

Meta Learning Algorithm

Policy

Single demonstration for task G

One-Shot Imitator (Neural network)

Environment

action

obs

[Duan, Andrychowicz, Stadie, Ho, Schneider, Sutskever, Abbeel, Zaremba, 2017]
Learning a One-Shot Imitator

[Figure credit: Bradly Stadie]
Each task is specified by a desired final layout
- Example: abcd
  - “Place c on top of d, place b on top of c, place a on top of b.”
Evaluation

Duan, Andrychowicz, Stadie, Ho, Schneider, Sutskever, Abbeel, Zaremba, 2017
Learning a One-Shot Imitator with MAML

- Meta-learning loss:

\[
\min_\theta \sum_{\text{tasks}} L_{\text{val}} (\theta - \alpha \nabla_\theta L_{\text{train}}(\theta))
\]

- Task loss = behavioral cloning loss:  
  \[
  L(\theta) = \sum_t \| \pi_\theta(o_t) - a^*_t \|^2
  \]

[Finn*, Yu*, Zhang, Abbeel, Levine, 2017]
Robot Experiments: Learning to Place

- Meta-training targets / objects
- Meta-testing targets / objects

1,300 demonstrations for meta-training

[Finn*, Yu*, Zhang, Abbeel, Levine, 2017]
Robot Experiments: Learning to Place

1 demo

imitation

Succes rate: 90%

[Finn*, Yu*, Zhang, Abbeel, Levine, 2017]

Pieter Abbeel -- embody.ai / UC Berkeley / Gradescope
Robot Experiments: Learning to Place

1 demo

imitation

Succes rate: 90%

[Finn*, Yu*, Zhang, Abbeel, Levine, 2017]
Current Directions

- Architectures for meta RL and imitation agents
  - Neural
  - Code
- Lifelong Learning
  - Non-stationary environments
  - Competition