Lecture 11: Some Advanced Research Topics

Zhi Wang & Chunlin Chen

Department of Control and Systems Engineering
Nanjing University

Dec. 10th, 2020

Table of Contents

What's the problem?

this is easy (mostly)

Why?

this is impossible

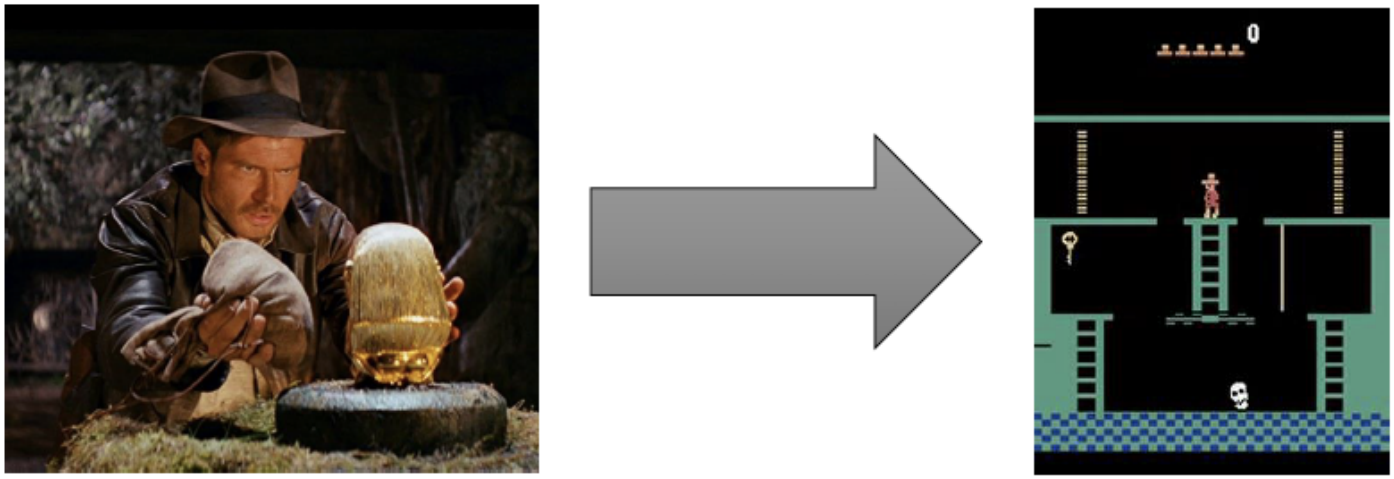
Montezuma's revenge

- Getting key = positive reward
- Opening door = positive 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

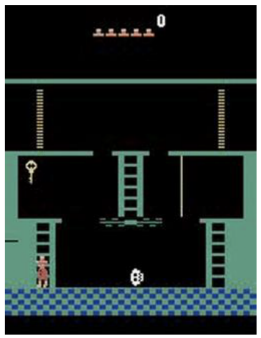
Transfer learning terminology

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

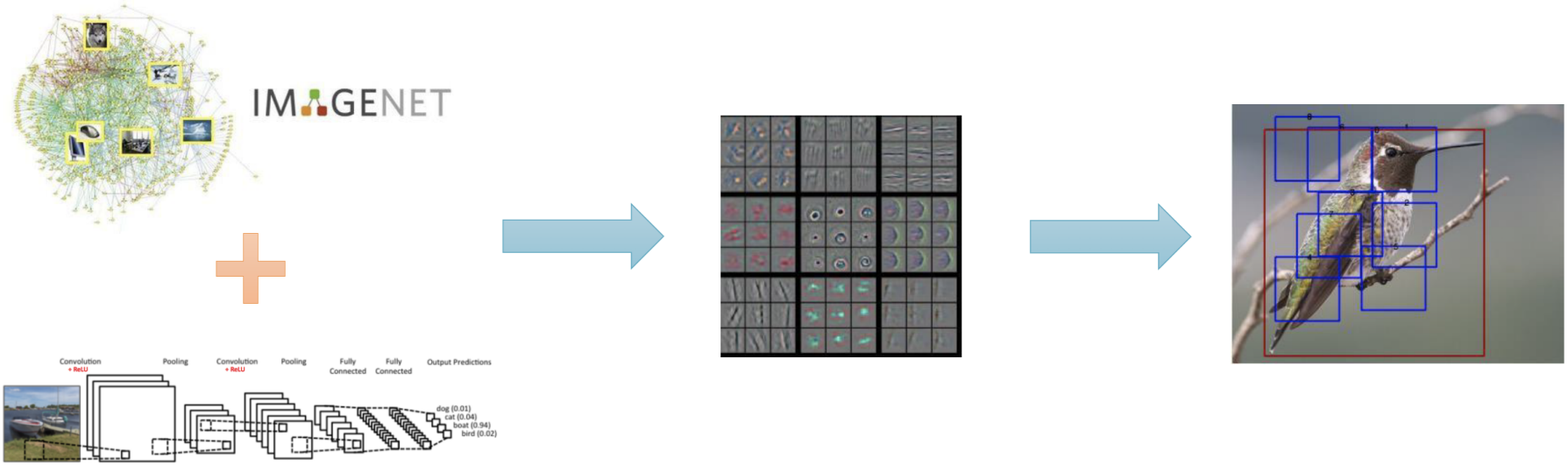
How can we frame transfer learning problems?

No single solution! Survey of various recent research papers?

- Forward transfer: train on one task, transfer to a new task
  - Transferring visual representations & domain adaptation
  - Fine-tune on the new task
  - Domain randomization
- Multi-task transfer: train on many tasks, transfer to a new task
  - Sharing representations and layers across tasks in multi-task learning
  - Contextual policies
  - Optimization challenges for multi-task learning
- Transferring models and value functions
  - Model-based RL as a mechanism for transfer
  - Successor features & representations

Forward transfer I: Pre-training + Fine-tuning

The most popular transfer learning method in (supervised) deep learning!

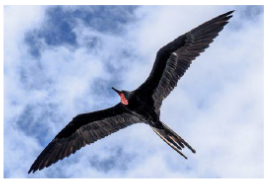


Where are the "ImageNet" features of RL?

What issues are we likely to face?

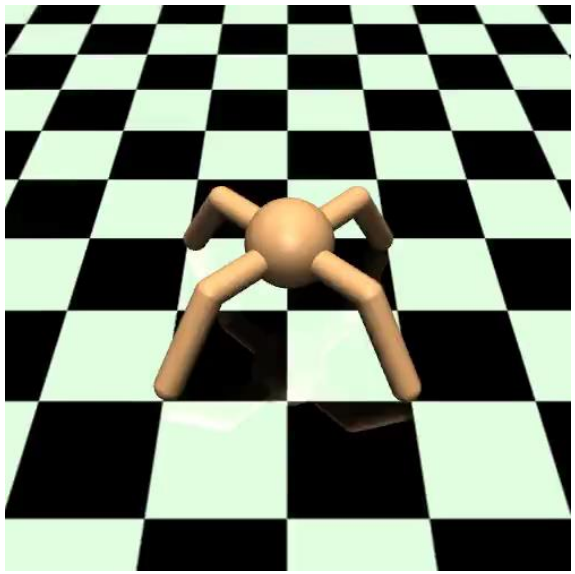
- **Domain shift**: representations learned in the source domain might not work well in the target domain


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



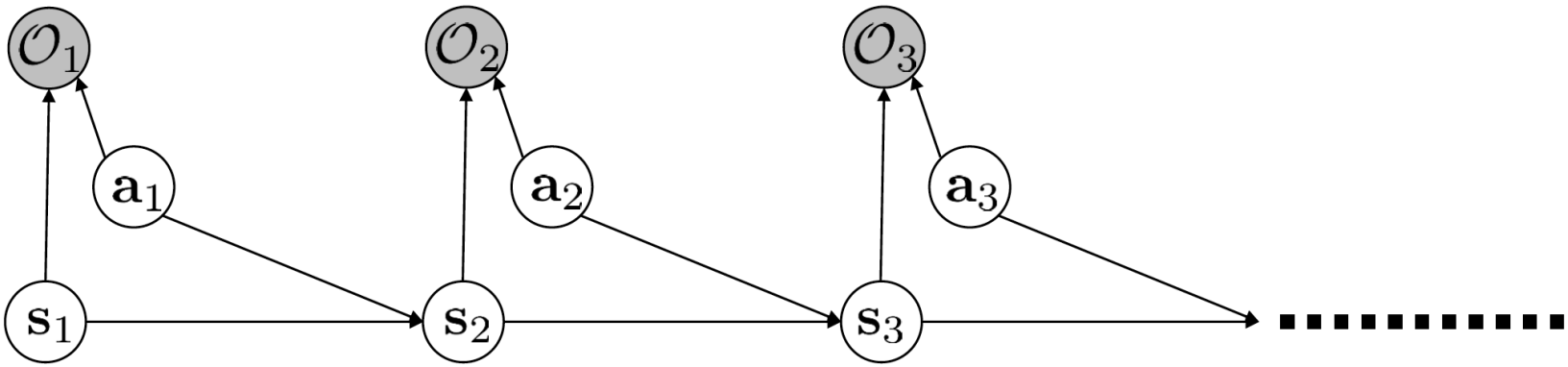

Challenges of fine-tuning in RL

- RL tasks are generally much less diverse
  - Features are less general
  - Policies & value functions become overly specialized
- Optimal policies in fully observed MDPs are deterministic
  - Loss of exploration at convergence
  - Low-entropy policies adapt very slowly to new settings



Fine-tuning with maximum-entropy policies

How can we increase diversity and entropy?

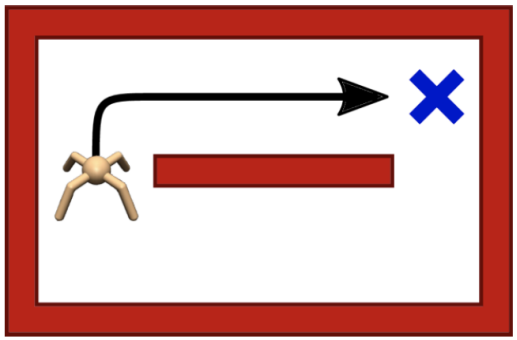


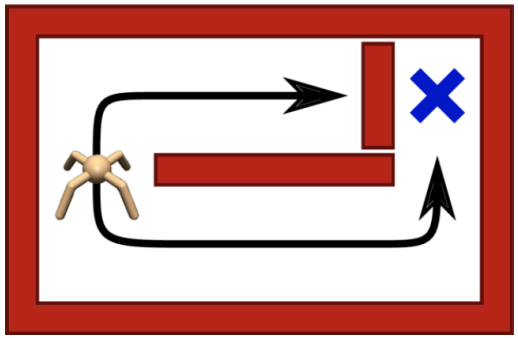
$$\pi(\mathbf{a}|\mathbf{s}) = \exp(Q_\phi(\mathbf{s},\mathbf{a}) - V(\mathbf{s})) \text{ optimizes } \sum_t E_{\pi(\mathbf{s}_t,\mathbf{a}_t)}[r(\mathbf{s}_t,\mathbf{a}_t)] + E_{\pi(\mathbf{s}_t)}[\mathcal{H}(\pi(\mathbf{a}_t|\mathbf{s}_t))]$$

policy entropy

Act **as randomly as possible** while collecting high rewards!

Pre-training for diversity/robustness





Learning to solve a task **in all possible ways** provides for more robust transfer!

Fine-tuning in RL: suggested readings

- Haarnoja et al., **Reinforcement learning with deep energy-based policies**, 2017.
- Andreas et al., **Modular multitask reinforcement learning with policy sketches**, 2017.
- Florensa et al., **Stochastic neural networks for hierarchical reinforcement learning**, 2017.
- Kumar et al., **One solution is not all you need: Few-shot extrapolation via structured MaxEnt RL**, 2020.

Forward transfer II: Domain randomization

What if we can manipulate the source domain?

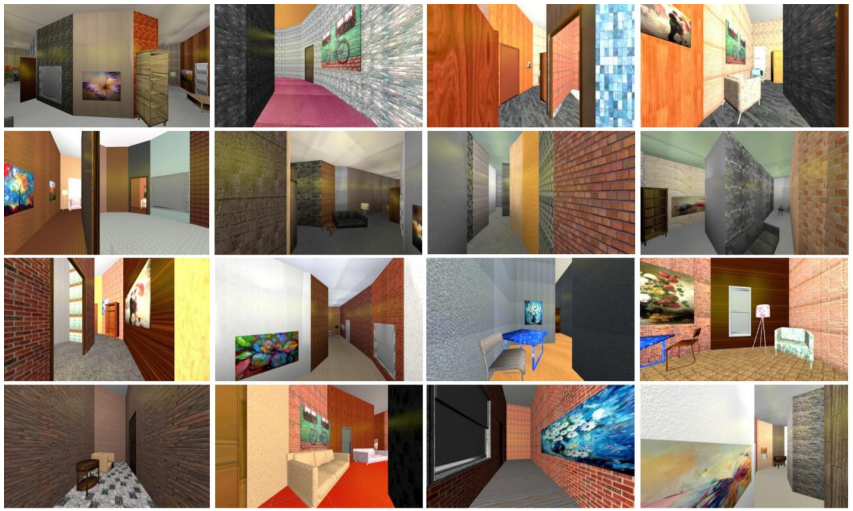
- So far: source domain and target domain are fixed
- What if we can **design** the source domain, and we have a **difficult** target domain?
  - Often the case for simulation to real world transfer, i.e., sim-to-real transfer

Domain randomization for sim-to-real transfer

- Instead of training a model on a single simulated environment, we randomize the simulator to expose the model to a wide range of environments at training time.
- Hypothesis: if the variability in simulation is significant enough, models trained in simulation will generalize to the real world with no additional training.

Josh Tobin, et al., "Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World," IROS 2017.

Domain randomization for sim-to-real transfer



- Fereshteh Sadeghi and Sergey Levine, "CAD2RL: Real Single-Image Flight without a Single Real Image," Robotics: Science and Systems Conference, 2017.

Table of Contents

Multi-task transfer

- Can we learn **faster** by learning multiple tasks?





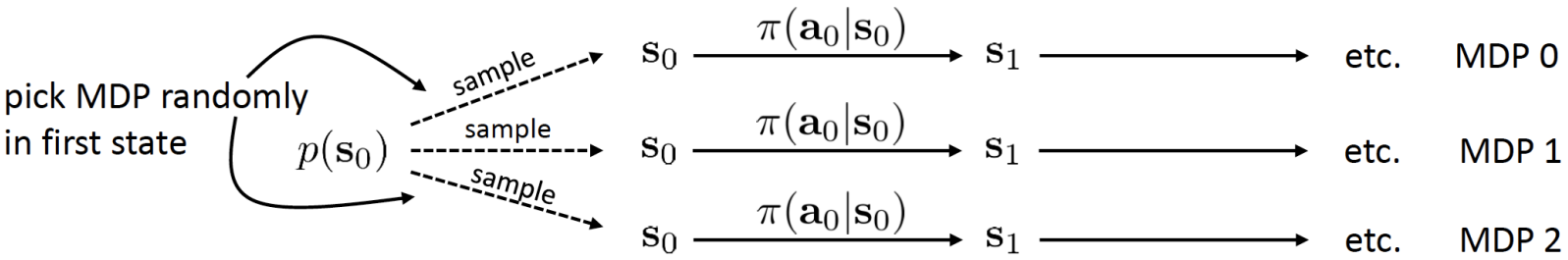
learn learn learn learn learn

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**

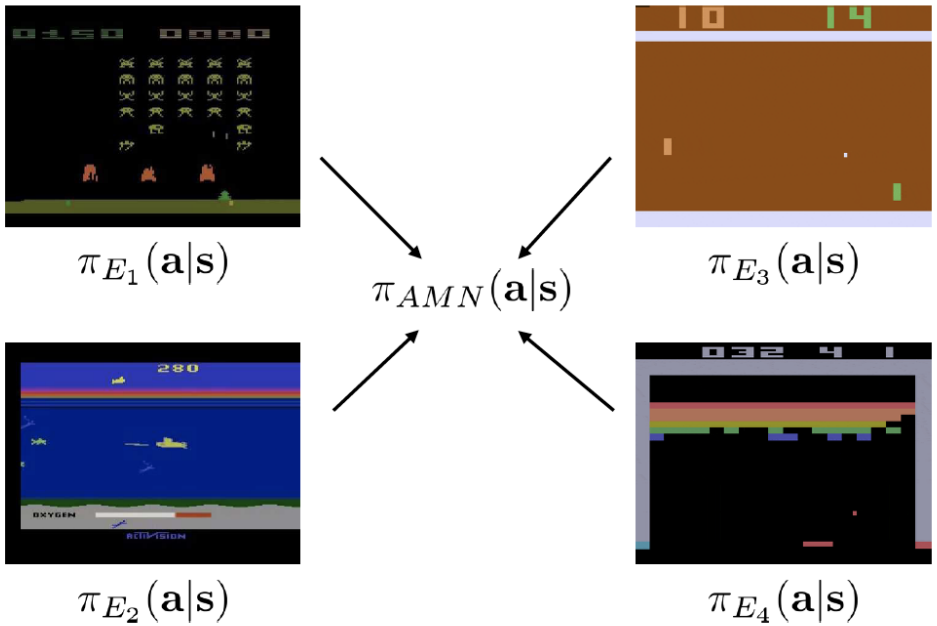


What is difficult about this?

- **Gradient interference**: becoming better on one task can make you worse on another
- **Winner-take-all problem**: imagine one task starts getting good - algorithm is likely to prioritize that task (to increase average expected reward) at the expensive of others

- In practice, this kind of multi-task RL is very challenging

Policy distillation for multi-task transfer



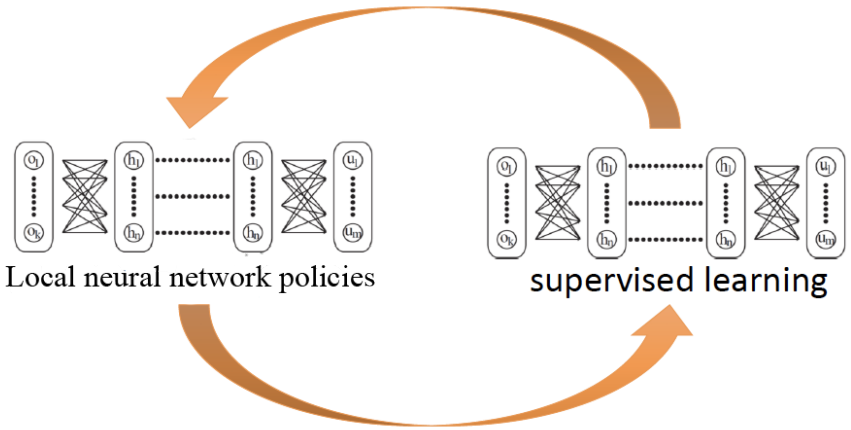
$$\mathcal{L} = \sum_{\mathbf{a}} \pi_{E_i}(\mathbf{a}|\mathbf{s}) \log \pi_{AMN}(\mathbf{a}|\mathbf{s})$$

(just supervised learning/distillation)

analogous to guided policy search, but for multi-task learning

- Parisotto et al., "Actor-mimic: Deep multitask and transfer reinforcement learning," ICLR 2016.

Combining weak policies into a strong policy



Divide and Conquer Reinforcement Learning

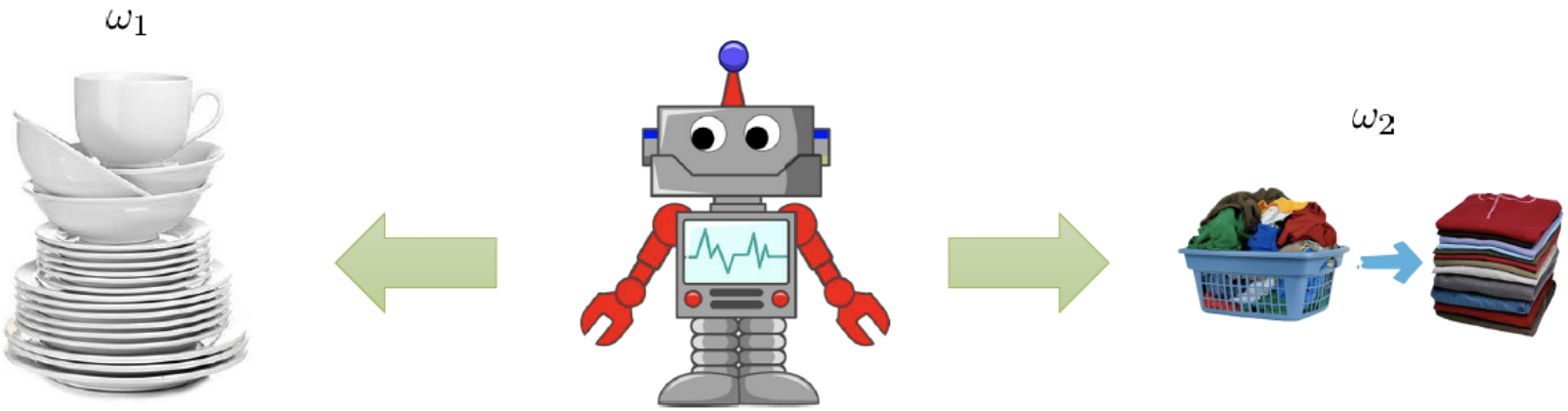
Divide and conquer reinforcement learning algorithm sketch:

1. optimize each local policy $\pi_{\theta_i}(\mathbf{a}_t|\mathbf{s}_t)$ on initial state $\mathbf{s}_{0,i}$ w.r.t. $\tilde{r}_{k,i}(\mathbf{s}_t, \mathbf{a}_t)$
2. use samples from step (1) to train $\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)$ to mimic each $\pi_{\theta_i}(\mathbf{u}_t|\mathbf{x}_t)$
3. update reward function $\tilde{r}_{k+1,i}(\mathbf{x}_t, \mathbf{u}_t) = r(\mathbf{x}_t, \mathbf{u}_t) + \lambda_{k+1,i} \log \pi_\theta(\mathbf{u}_t|\mathbf{x}_t)$

- Ghosh et al., "Divide and conquer reinforcement learning," ICLR 2018.

How does the model know what to do?

- So far: what to do is apparent from the input (e.g., which game is being played)
- What if the policy can do multiple things in the same environment?

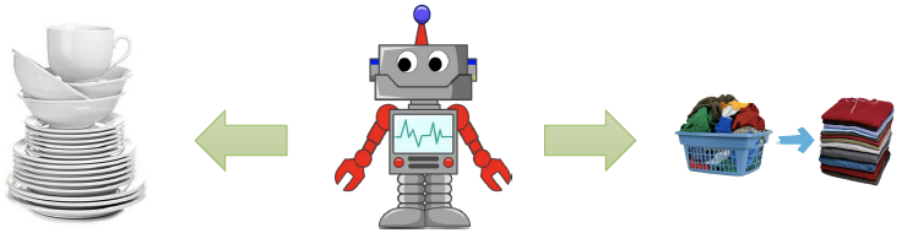


Contextual policies

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

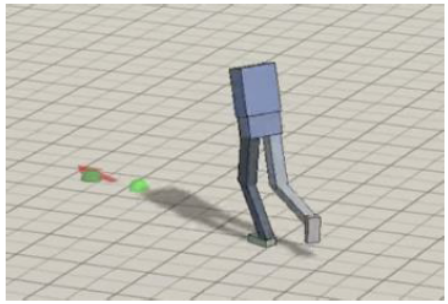
contextual policy: $\pi_\theta(\mathbf{a}|\mathbf{s}, \omega)$

e.g., do dishes or laundry

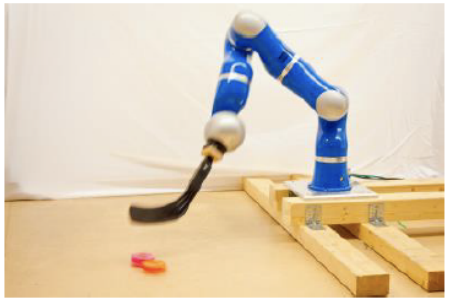


formally, simply defines augmented state space: $\tilde{\mathbf{s}} = \begin{bmatrix} \mathbf{s} \\ \omega \end{bmatrix}$ $\tilde{\mathcal{S}} = \mathcal{S} \times \Omega$

ω: stack location



ω: walking direction



ω: where to hit puck

Table of Contents

Real-world behavior is hierarchical

- Humans plan modularly at different granularities of understanding
- Going out of one room is similar to going out of another room

1. pour coffee
2. add sugar
3. add milk
4. stir

1. set water temp
   - too cold → add hot
   - too hot → add cold
   - change → wait 5sec
   - just right → success
2. get wet
3. shampoo
4. soap
5. turn off water
6. dry off

Real-world behavior is hierarchical

- In the field of cognitive science, research has long suggested that human and animal behavior is based on a hierarchical structure

- Botvinick, Niv and Barto. **Hierarchically organized behavior and its neural foundations: a reinforcement learning perspective**. *Cognition*, 2008.
- Badre, Kayser, and D'Esposito. **Frontal cortex and the discovery of abstract action rules**, *Neuron*, 2010.

Hierarchical RL (HRL)

- RL problems suffer from serious scaling issues. HRL is a computational approach intended to address these issues by learning to operate on different levels of temporal abstraction

- **Long-term credit assignment**: faster learning and better generalization
- **Structured exploration**: explore with sub-policies rather than primitive actions
- **Transfer learning**: different levels of hierarchy can encompass different knowledge and allow for better transfer

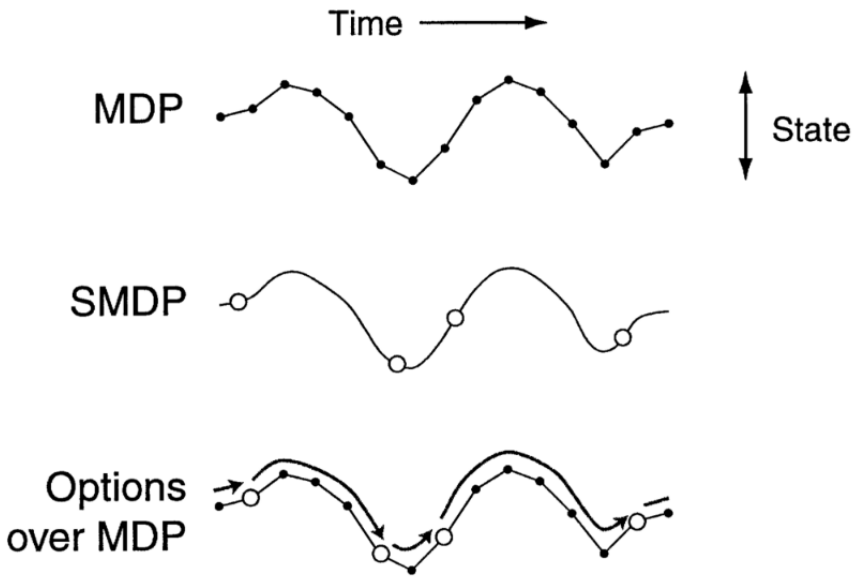
The "Option" framework

Bottom level is a sub-policy

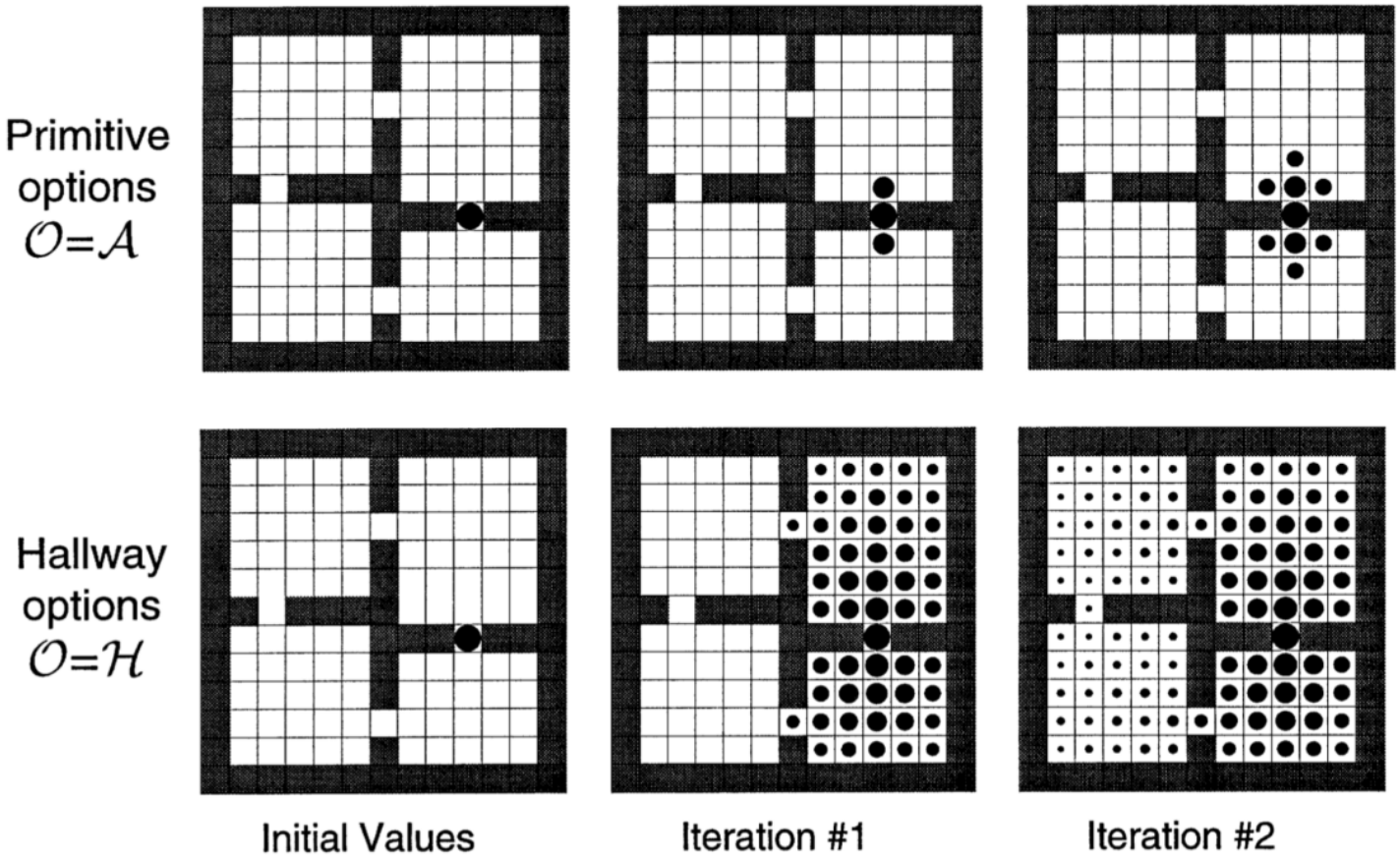
- takes environment observations
- outputs actions
- runs until termination

Top level is a policy-over-options:

- takes environment observations
- outputs sub-policies
- runs until termination



Option framework

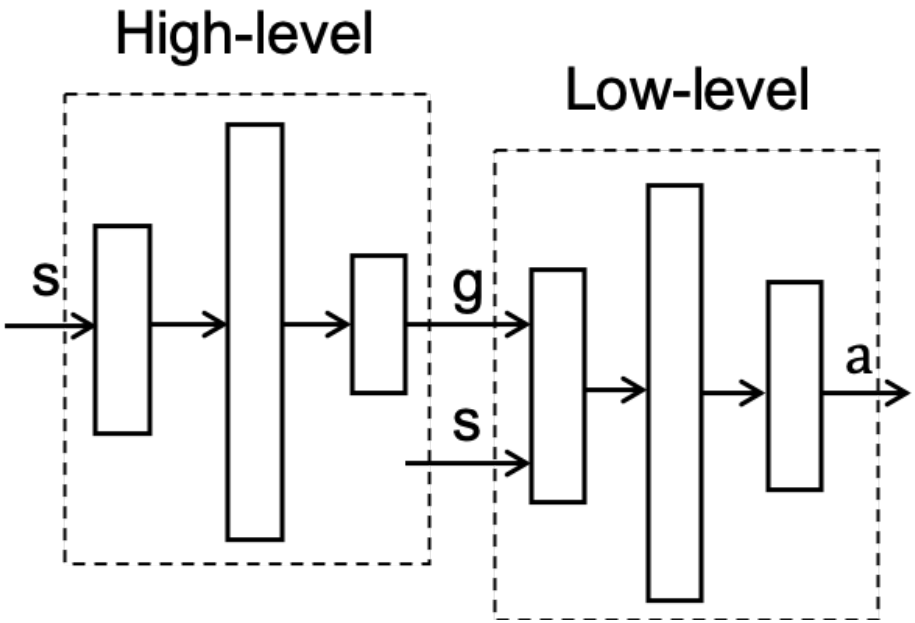


HRL frameworks

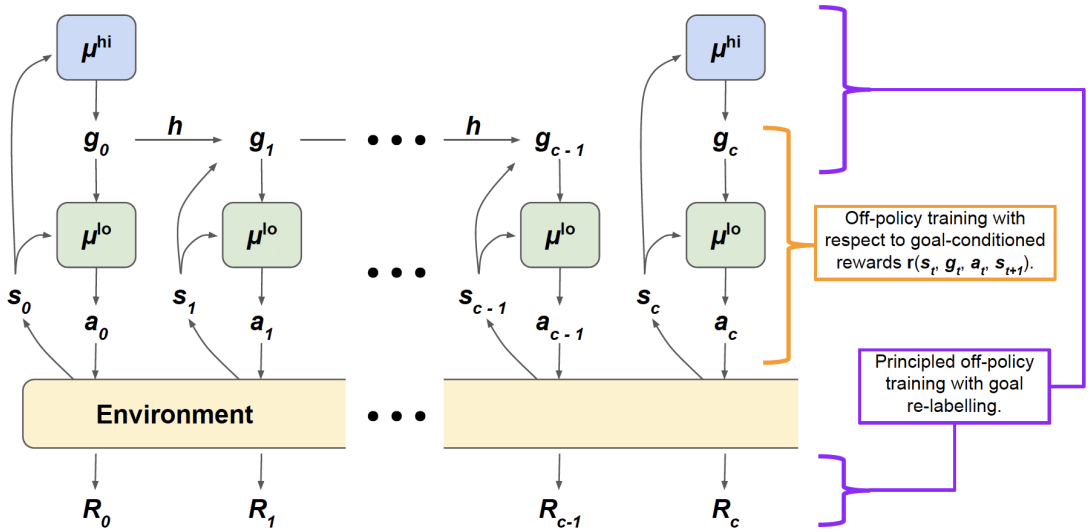
	Temporal abstraction	State abstraction	Sub-tasks *fixed policy provided by the programmer*	Sub-tasks *non-deterministic finite-state controller*	Sub-tasks *termination predicate and a local reward function*	Algorithm with proven convergence	Optimal policy	Domain knowledge requirements
Feudal	✓	✓			✓		hierarchical	++++
Options	✓		✓			✓	hierarchical	+++
HAM	✓			✓		✓	hierarchical	++
MAXQ	✓	✓			✓	✓	recursive	+

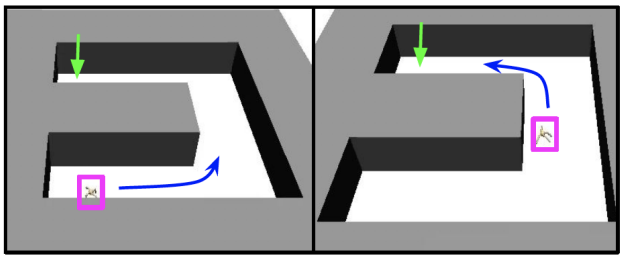
Goal-conditioned HRL

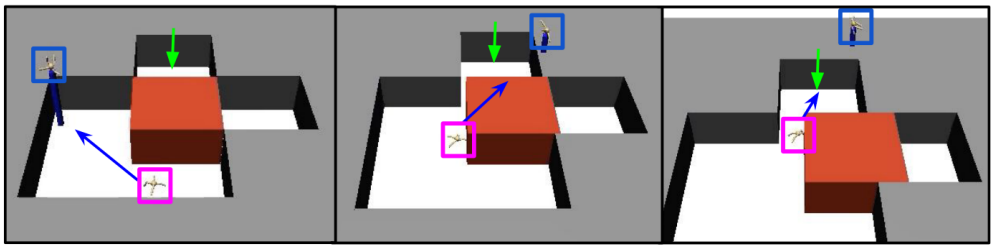
- **High-level**: proposes goal states g
- **Low-level**: is trained to real these goal states



Goal-conditioned HRL







- O. Nachum, et al., **Data-Efficient Hierarchical Reinforcement Learning**, NIPS 2018.

Hierarchical RL: suggested readings

- M. Ghavamzadeh and S. Mahadevan, **Hierarchical Average Reward Reinforcement Learning**, JMLR 2007.
- TG Dietterich, **Hierarchical reinforcement learning with the MAXQ value function decomposition**, JAIR 2010.
- T. D. Kulkarni, et al., **Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation**, NIPS 2016.
- A. S. Vezhnevets, et al., **FeUdal networks for hierarchical reinforcement learning**, ICML 2017.
- O. Nachum, et al., **Data-Efficient Hierarchical Reinforcement Learning**, NIPS 2018.
- O. Nachum, et al., **Why does hierarchy (sometimes) work so well in reinforcement learning?** 2019.

Table of Contents

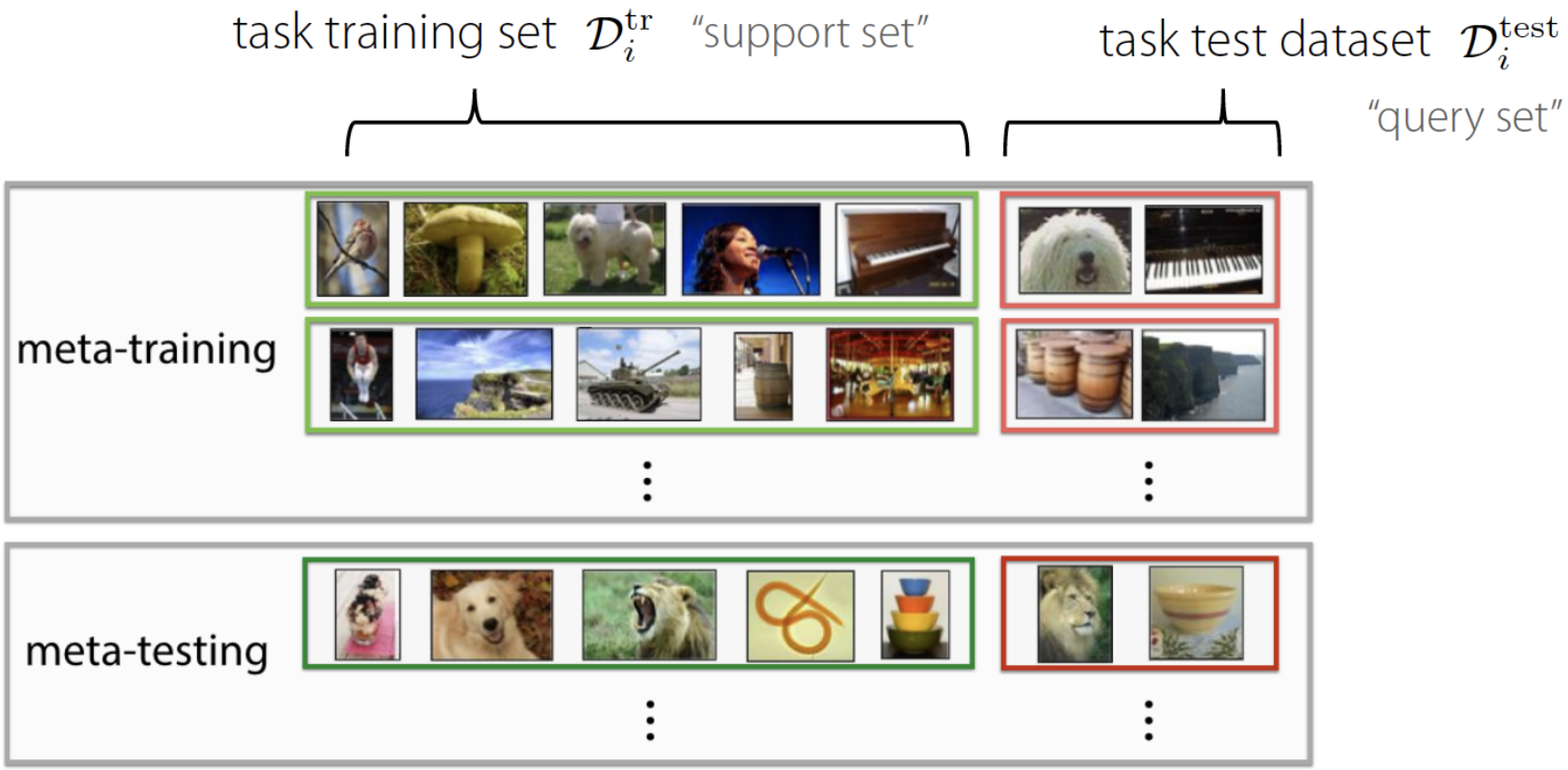
The meta-learning problem

- Given data from $\mathcal{T}_1, ..., \mathcal{T}_n$, quickly solve a new task $\mathcal{T}_{test}$
- **Key assumption**: meta-training tasks and meta-test task drawn i.i.d. from the same task distribution
  - Tasks must share structure

$$\mathcal{T}_1, ..., \mathcal{T}_n \sim p(\mathcal{T}), \qquad \mathcal{T}_{test} \sim p(\mathcal{T})$$

- What do the tasks correspond to?
  - recognizing handwritten digits from different languages
  - spam filter for different users
  - classifying species in different regions of the world
  - a robot performing different tasks

Meta-learning terminology



- k-shot learning: learning with k examples per class
- N-way classification: choosing between N classes

Meta-learning terminology

Supervised Learning:

Inputs: $\mathbf{x}$ Outputs: $\mathbf{y}$ Data: $\{(\mathbf{x}, \mathbf{y})_i\}$

$$\mathbf{y} = f(\mathbf{x}; \theta)$$

Meta Supervised Learning:

Inputs: $\mathcal{D}^{\text{tr}}$ $\mathbf{x}_{\text{test}}$ Outputs: $\mathbf{y}_{\text{test}}$ Data: $\{\mathcal{D}_i\}$

$$\{(\mathbf{x}, \mathbf{y})_{1:K}\} \quad \mathbf{y}_{\text{test}} = h(\mathcal{D}^{\text{tr}}, \mathbf{x}_{\text{test}}; \theta)$$

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

- Why is this view useful?
  - Reduces the meta-learning problem to the design and optimization of hypothesis $h(\cdot)$

Model-agnostic meta-learning (MAML)

Fine-tuning [test-time]

$$\phi \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}^{\text{tr}})$$

pre-trained parameters

training data for new task

Meta-learning

$$\min_\theta \sum_{\text{task } i} \mathcal{L}(\theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}_i^{\text{tr}}), \mathcal{D}_i^{\text{ts}})$$

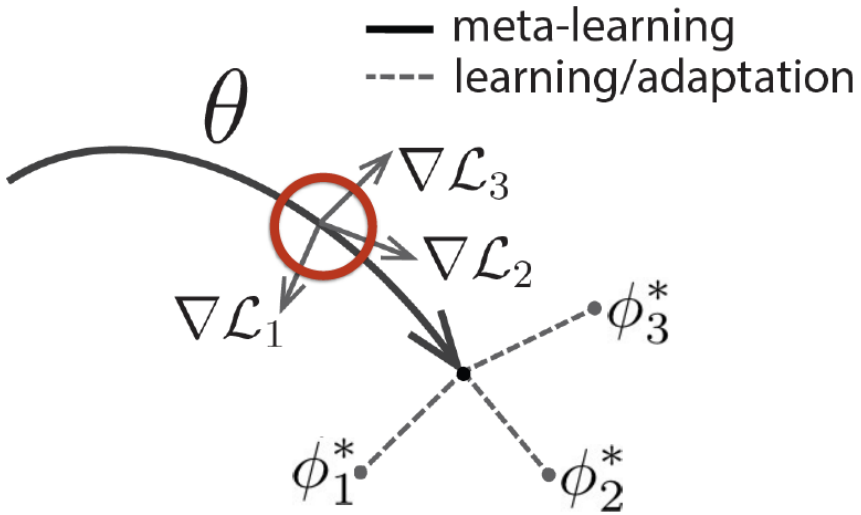
- **Key idea**: Over many tasks, learn parameter vector θ that transfers via fine-tuning
  - a small number of gradient steps with a small amount of training data from a new task will produce good generalization performance on that task

Optimization-based adaptation

$$\min_{\theta} \sum_{\text{task } i} \mathcal{L}(\theta - \alpha \nabla_{\theta} \mathcal{L}(\theta, \mathcal{D}_i^{\text{tr}}), \mathcal{D}_i^{\text{ts}})$$

θ parameter vector being meta-learned

ϕ_i^* optimal parameter vector for task i



Model-Agnostic Meta-Learning

- Finn et al., Model-agnostic meta-learning, ICML 2017.

Chelsea Finn

Chelsea Finn

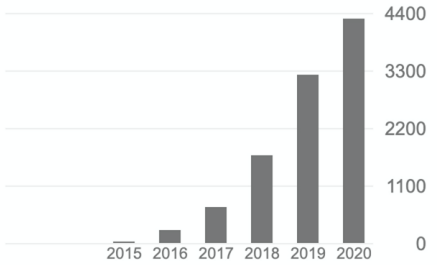
FOLLOWING

Stanford University, Google
Verified email at cs.stanford.edu - Homepage
machine learning robotics reinforcement learning

TITLE	CITED BY	YEAR
Model-agnostic meta-learning for fast adaptation of deep networks C Finn, P Abbeel, S Levine International Conference on Machine Learning (ICML), 1126-1135	2532	2017
End-to-end training of deep visuomotor policies S Levine, C Finn, T Darrell, P Abbeel The Journal of Machine Learning Research 17 (1), 1334-1373	2070	2016
Unsupervised learning for physical interaction through video prediction C Finn, I Goodfellow, S Levine arXiv preprint arXiv:1605.07157	650	2016
Guided cost learning: Deep inverse optimal control via policy optimization C Finn, S Levine, P Abbeel International Conference on Machine Learning (ICML), 49-58	467	2016
Deep visual foresight for planning robot motion C Finn, S Levine 2017 IEEE International Conference on Robotics and Automation (ICRA), 2786-2793	378	2017

Cited by

	All	Since 2015
Citations	10339	10308
h-index	37	37
i10-index	59	59



Co-authors VIEW ALL


Sergey Levine
UC Berkeley, Google


Pieter Abbeel
UC Berkeley | Covariant.AI

- CS 330 at Stanford, Deep Multi-Task and Meta Learning
  - `http://web.stanford.edu/class/cs330/`

Review of syllabus

Lecture Slides

• Sept. 17th, 2020	• Lecture 0: Preliminaries of Machine Learning
• Sept. 24th, 2020	• Lecture 1: Introduction to Reinforcement Learning
• Oct. 10th, 2020	• Lecture 2: Dynamic Programming
• Oct. 15th, 2020	• Lecture 3: Monte Carlo Methods and Temporal-Difference Learning
• Oct. 22nd, 2020	• Lecture 4: Introduction to Deep Reinforcement Learning
• Oct. 29th, 2020	• Lecture 5: Policy Gradients
• Nov. 5th, 2020	• Lecture 6: Advanced Policy Gradients
• Nov. 12th, 2020	• Lecture 7: Actor-Critic Algorithms
• Nov. 19th, 2020	• Lecture 8: Value Function Methods
• Nov. 26th, 2020	• Lecture 9: Deep Q-learning
• Dec. 3rd, 2020	• Lecture 10: Model-Based Reinforcement Learning
• Dec. 10th, 2020	• Lecture 11: Some Advanced Research Topics

THE END