Meta Reinforcement Learning

Alina Vereshchaka

CSE4/510 Reinforcement Learning
Fall 2019

avereshc@buffalo.edu

November 21, 2019

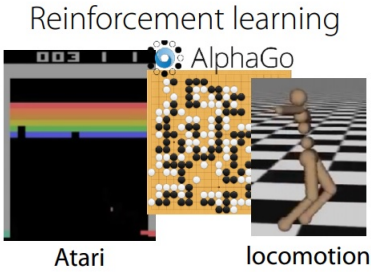
* Slides are based on CS 330: Deep Multi-Task and Meta Learning by Chelsea Finn (Stanford); Meta Reinforcement Learning lecture by Kate Rakelly (UC Berkley); Meta Reinforcement Learning by Lilian Weng

Overview

1. Recap: Reinforcement Learning
2. Meta-learning problem statement
3. Meta-imitation Learning

Table of Contents

Recap: Reinforcement Learning



Learn **one task** in **one environment**, **starting from scratch**
rely on **detailed supervision and guidance**.

Recap: Reinforcement Learning

Reinforcement learning

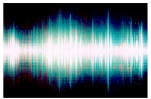
Atari locomotion

Learn **one task** in **one environment**, **starting from scratch**
rely on **detailed supervision and guidance**.

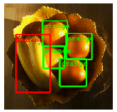
Not just a problem with reinforcement learning & robotics.

specialists
[single task]

machine translation



speech recognition



object detection

More diverse, yet still **one task**, **from scratch**, with **detailed supervision**

Recap: Reinforcement Learning

Source: https://youtu.be/8vNxjwt2AqY

Recap: Reinforcement Learning

vs.

Recap: Reinforcement Learning

What if you need to quickly learn something new?
about a new person, for a new task, about a new environment, etc.

Learning Something New

training data

Braque

Cezanne

test datapoint

By Braque or Cezanne?

Learning Something New

What if you need to quickly learn something new?

about a new person, for a new task, about a new environment, etc.

"few-shot learning"

How did you accomplish this?

by leveraging prior experience!

Learning Something New

The bad news: Different tasks need to share some structure.

If this doesn't hold, you are better off using single-task learning.

The good news: There are many tasks with shared structure!



Even if the tasks are seemingly unrelated:

- The **laws of physics** underly real data.
- People are **all organisms with intentions**.
- The **rules of English** underly English language data.
- Languages all develop for **similar purposes**.

This leads to far greater structure than random tasks.

Table of Contents

Goal: design models that can learn new skills or adapt to new environments rapidly with a few training examples.

Meta Learning

- Good meta-learning model capable of well adapting or generalizing to new tasks and new environments that have never been encountered during training time.

Meta Learning

- Good meta-learning model capable of well adapting or generalizing to new tasks and new environments that have never been encountered during training time.
- The **adaptation process**, essentially a mini learning session, happens during test but with a limited exposure to the new task configurations.

Meta Learning

- Good meta-learning model capable of well adapting or generalizing to new tasks and new environments that have never been encountered during training time.
- The **adaptation process**, essentially a mini learning session, happens during test but with a limited exposure to the new task configurations.
- Meta-learning is also known as **learning to learn**.

Meta Learning

- Good meta-learning model capable of well adapting or generalizing to new tasks and new environments that have never been encountered during training time.
- The **adaptation process**, essentially a mini learning session, happens during test but with a limited exposure to the new task configurations.
- Meta-learning is also known as **learning to learn**.
- Examples:
  - A classifier trained on non-cat images can tell whether a given image contains a cat after seeing a handful of cat pictures.

Meta Learning

- Good meta-learning model capable of well adapting or generalizing to new tasks and new environments that have never been encountered during training time.
- The **adaptation process**, essentially a mini learning session, happens during test but with a limited exposure to the new task configurations.
- Meta-learning is also known as **learning to learn**.
- Examples:
  - A classifier trained on non-cat images can tell whether a given image contains a cat after seeing a handful of cat pictures.
  - A game bot is able to quickly master a new game.

Meta Learning

- Good meta-learning model capable of well adapting or generalizing to new tasks and new environments that have never been encountered during training time.
- The **adaptation process**, essentially a mini learning session, happens during test but with a limited exposure to the new task configurations.
- Meta-learning is also known as **learning to learn**.
- Examples:
  - A classifier trained on non-cat images can tell whether a given image contains a cat after seeing a handful of cat pictures.
  - A game bot is able to quickly master a new game.
  - A mini robot completes the desired task on an uphill surface during test even through it was only trained in a flat surface environment.

Meta-learning Problem Statement

supervised learning

Meta-learning Problem Statement

supervised learning

reinforcement learning

Meta-learning Problem Statement

supervised learning

reinforcement learning

Robot art by Matt Spangler, *mattspangler.com*

Meta-learning Problem Statement

supervised learning

"Dalmation" "German shepherd" "Pug"

corgi ???

reinforcement learning

Robot art by Matt Spangler, *mattspangler.com*

RL Problem Statement

Regular RL: learn policy for single task

$$\theta^\star = \arg\max_\theta E_{\pi_\theta(\tau)}[R(\tau)]$$

$$= f_{\mathrm{RL}}(\mathcal{M})$$

MDP

Meta-learning Problem Statement

Regular RL: learn policy for single task

$$\theta^\star = \arg\max_\theta E_{\pi_\theta(\tau)}[R(\tau)]$$

$$= f_{\text{RL}}(\mathcal{M})$$

MDP

Meta-RL: learn adaptation rule

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

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

MDP for task i

$\mathcal{M}_1$ $\mathcal{M}_2$ $\mathcal{M}_3$ $\mathcal{M}_{test}$

Meta-learning Problem Statement

Regular RL: learn policy for single task

$$\theta^\star = \arg\max_\theta E_{\pi_\theta(\tau)}[R(\tau)]$$

$$= f_{\text{RL}}(\mathcal{M})$$

MDP

Meta-RL: learn adaptation rule

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

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

MDP for task i

Meta-training / Outer loop

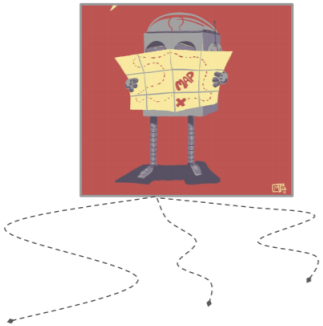
Adaptation / Inner loop

$\mathcal{M}_1$ $\mathcal{M}_2$ $\mathcal{M}_3$ $\mathcal{M}_{test}$

Meta-learning Problem Statement

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

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



What should the adaptation procedure do?

- **Explore**: Collect the most informative data
- **Adapt**: Use that data to obtain the optimal policy

Meta Learning

- The goal of **few-shot meta-learning** is to train a model that can quickly adapt to a new task using only a few datapoints and training iterations

Meta Learning

- The goal of **few-shot meta-learning** is to train a model that can quickly adapt to a new task using only a few datapoints and training iterations
- To do this, the model is trained during a meta-learning phase on a set of tasks, such that the trained model can quickly adapt to new tasks using only a small number of examples or trials.

Meta Learning

- The goal of **few-shot meta-learning** is to train a model that can quickly adapt to a new task using only a few datapoints and training iterations
- To do this, the model is trained during a meta-learning phase on a set of tasks, such that the trained model can quickly adapt to new tasks using only a small number of examples or trials.
- A meta-RL model is trained over a distribution of MDPs ($M_i \in \mathcal{M}$), and at test time, it is able to learn to solve a new task quickly.

Meta RL: Training Procedure

The training procedure:

1. Sample a new MDP, $M_i \sim \mathcal{M}$
2. Reset the hidden state of the model;
3. Collect multiple trajectories and update the model weights;
4. Repeat from step 1.

Meta RL: Schema

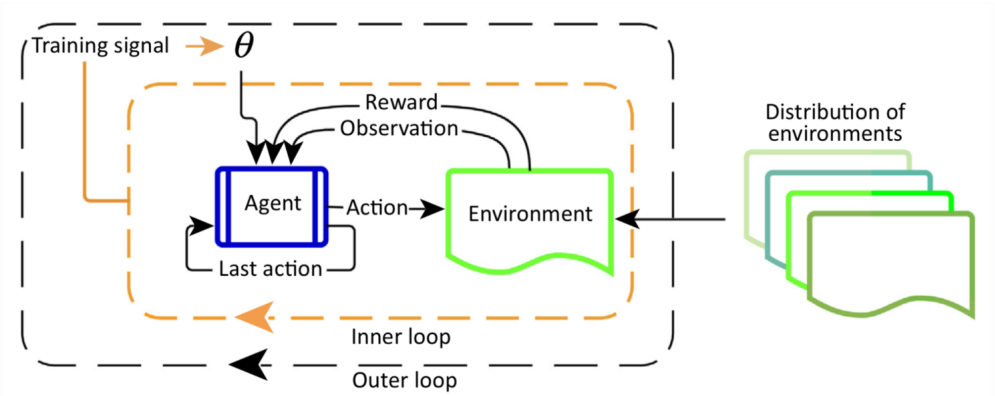


Fig. 2. Illustration of meta-RL, containing two optimization loops. The outer loop samples a new environment in every iteration and adjusts parameters that determine the agent's behavior. In the inner loop, the agent interacts with the environment and optimizes for the maximal reward. (Image source: Botvinick, et al. 2019

Key Components in Meta-RL

- **A Model with Memory**
  A recurrent neural network maintains a hidden state. Thus, it could acquire and memorize the knowledge about the current task by updating the hidden state during rollouts. Without memory, meta-RL would not work.

Key Components in Meta-RL

- **A Model with Memory**
  A recurrent neural network maintains a hidden state. Thus, it could acquire and memorize the knowledge about the current task by updating the hidden state during rollouts. Without memory, meta-RL would not work.
- **Meta-learning Algorithm**
  A meta-learning algorithm refers to how we can update the model weights to optimize for the purpose of solving an unseen task fast at test time.

Key Components in Meta-RL

- **A Model with Memory**
  A recurrent neural network maintains a hidden state. Thus, it could acquire and memorize the knowledge about the current task by updating the hidden state during rollouts. Without memory, meta-RL would not work.
- **Meta-learning Algorithm**
  A meta-learning algorithm refers to how we can update the model weights to optimize for the purpose of solving an unseen task fast at test time.
- **A Distribution of MDPs**
  While the agent is exposed to a variety of environments and tasks during training, it has to learn how to adapt to different MDPs.

Main Differences from RL

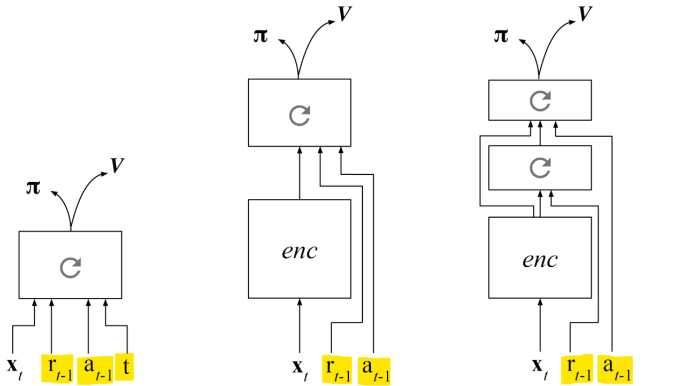
In RL: $\pi_\theta(s_t) \rightarrow$ a distribution over $\mathcal{A}$

Main Differences from RL

In RL: $\pi_\theta(s_t) \rightarrow$ a distribution over $\mathcal{A}$

In meta-RL: $\pi_\theta(a_{t-1}, r_{t-1}, s_t) \rightarrow$ a distribution over $\mathcal{A}$

Advantage actor-critic with recurrence



(a) LSTM A2C (b) LSTM A3C (c) Stacked-LSTM A3C

Figure: Different actor-critic architectures all use a recurrent model. Last reward and last action are additional inputs. The observation is fed into the LSTM either as a one-hot vector or as an embedding vector after passed through an encoder model. "Learning to Reinforcement Learn" by JX Wang et al.

General Meta-RL Algorithm Outline

Algorithm Outline

While training:

1. Sample task i (new MDP), collect data $\mathcal{D}_i$
2. Adapt policy by computing $\phi_i = f(\theta, \mathcal{D}_i)$
3. Collect data $\mathcal{D}'_i$, with adapted policy π_{ϕ_i}
4. Update θ according to $\mathcal{L}(\mathcal{D}'_i, \phi_i)$

General Meta-RL Algorithm Outline

Algorithm Outline

While training:

1. Sample task i (new MDP), collect data $\mathcal{D}_i$
2. Adapt policy by computing $\phi_i = f(\theta, \mathcal{D}_i)$
3. Collect data $\mathcal{D}_i'$, with adapted policy π_{ϕ_i}
4. Update θ according to $\mathcal{L}(\mathcal{D}_i', \phi_i)$

Different algorithms:

- Choice of f
- Choice of loss function $\mathcal{L}$

Solution 1: Meta RL with Recurrence

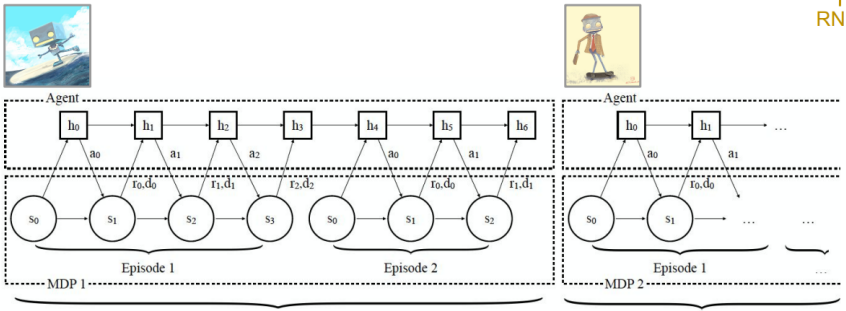
$$\theta^\star = \arg\max_\theta \sum_{i=1}^{n} E_{\pi_{\phi_i}(\tau)}[R(\tau)]$$

PG

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

RNN

Implement the policy as a recurrent network, train across a set of tasks



Duan et al. 2016, Wang et al. 2016. Heess et al. 2015. Fig adapted from Duan et al. 2016

Solution 1: Meta RL with Recurrence

while training:
  for i in tasks:
    initialize hidden state $\mathbf{h_0} = 0$
    for t in timesteps:
      1. sample 1 transition $\mathcal{D}_i = \mathcal{D}_i \cup \{(s_t, a_t, s_{t+1}, r_t)\}$ from π_{h_t}
      2. update policy hidden state $\mathbf{h_{t+1}} = f_\theta(\mathbf{h_t}, s_t, a_t, s_{t+1}, r_t)$

  update policy parameters $\theta \leftarrow \theta - \nabla_\theta \sum_i \mathcal{L}_i(\mathcal{D}_i, \pi_{\mathbf{h}})$

Solution 2: Model-Agnostic Meta-Learning (MAML)

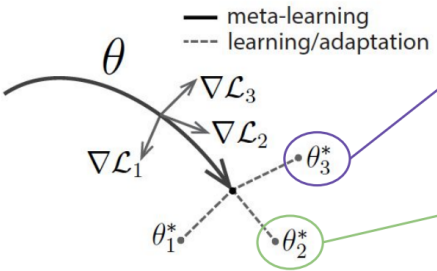
$$\theta^\star = \arg\max_\theta \sum_{i=1}^{n} E_{\pi_{\phi_i}(\tau)}[R(\tau)]$$

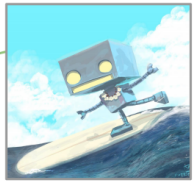
PG

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

PG

Learn a parameter initialization from which fine-tuning for a new task works!





Solution 2: Model-Agnostic Meta-Learning (MAML)

while training:
for i in tasks:

1. sample k episodes $\mathcal{D}_i = \{(s, a, s', r)\}_{1:k}$ from π_θ
2. compute adapted parameters $\phi_i = \theta - \alpha \nabla_\theta \mathcal{L}_i(\pi_\theta, \mathcal{D}_i)$
3. sample k episodes $\mathcal{D}'_i = \{(s, a, s', r)_{1:k}\}$ from π_ϕ

update policy parameters $\theta \leftarrow \theta - \nabla_\theta \sum_i \mathcal{L}_i(\mathcal{D}'_i, \pi_{\phi_i})$

Requires second order derivatives!

Finn et al. 2017. Fig adapted from Finn et al. 2017

Table of Contents

Meta-imitation Learning

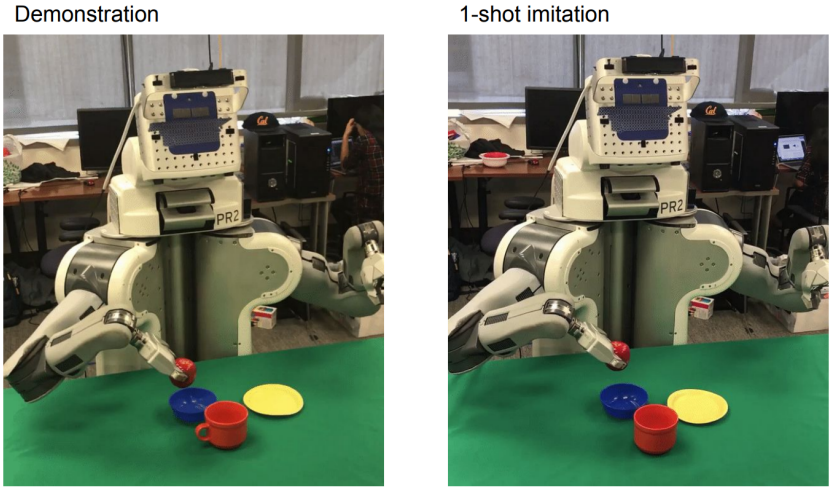


Figure: One-Shot Imitation from Watching Videos by BAIR (More details)

Meta-imitation learning

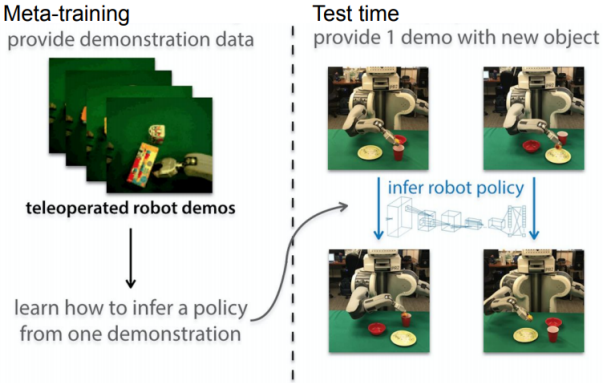
$$\theta^\star = \arg\max_\theta \sum_{i=1}^{n} E_{\pi_{\phi_i}(\tau)}[R(\tau)]$$

PG

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

Behavior cloning

Test: perform task given single **robot demo**
Training: run **behavior cloning** for adaptation



$$\phi_i = \theta - \alpha \nabla_\theta \sum_t ||\pi_\theta(o_t) - a_t^*||^2$$

Meta-imitation learning from **humans**

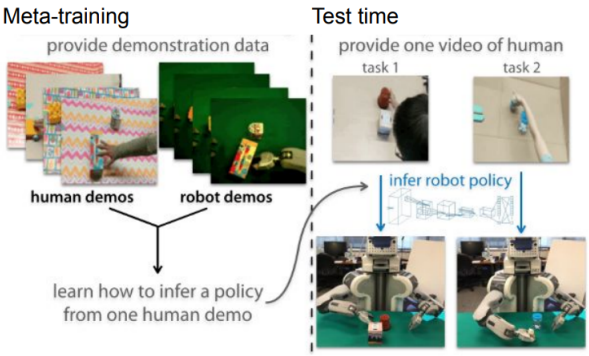
$$\theta^\star = \arg\max_\theta \sum_{i=1}^{n} E_{\pi_{\phi_i}(\tau)}[R(\tau)]$$

PG

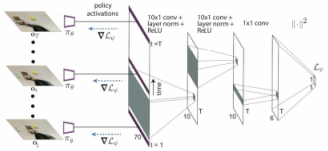
$$\text{where } \phi_i = f_\theta(\mathcal{M}_i)$$

Learned loss

Test: perform task given single **human demo**
Training: **learn a loss function** that adapts policy



$$\phi = \theta - \alpha \nabla_\theta \mathcal{L}_\psi(\theta, \mathbf{d}^h)$$



Supervised by **paired robot-human demos** only during meta-training!

Behavior Learning

1. run base policy $\pi_0(\mathbf{a}_t|\mathbf{s}_t)$ (e.g., random policy) to collect $\mathcal{D} = \{(\mathbf{s}, \mathbf{a}, \mathbf{s}')_i\}$
2. learn dynamics model $f(\mathbf{s}, \mathbf{a})$ to minimize $\sum_i \|f(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i\|^2$
3. plan through $f(\mathbf{s}, \mathbf{a})$ to choose actions

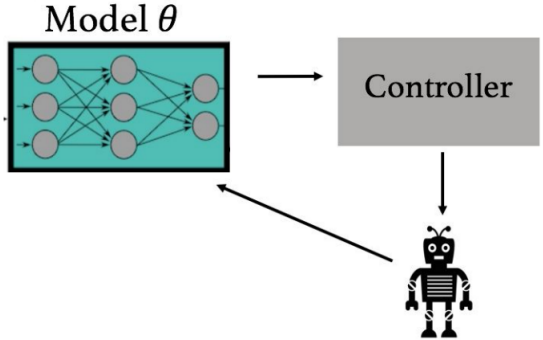
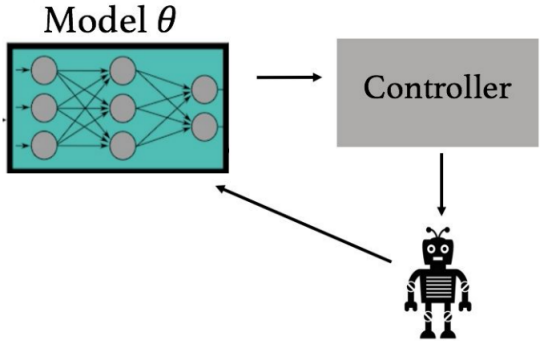


Figure adapted from Anusha Nagabandi

Meta-imitation Learning

1. run base policy $\pi_0(\mathbf{a}_t|\mathbf{s}_t)$ (e.g., random policy) to collect $\mathcal{D} = \{(\mathbf{s}, \mathbf{a}, \mathbf{s}')_i\}$
2. learn dynamics model $f(\mathbf{s}, \mathbf{a})$ to minimize $\sum_i \|f(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i\|^2$
3. plan through $f(\mathbf{s}, \mathbf{a})$ to choose actions



What if the system dynamics change?

- Low battery
- Malfunction
- Different terrain

Re-train model? :(

Figure adapted from Anusha Nagabandi

Meta-imitation Learning

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

MPC

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

Supervised model learning

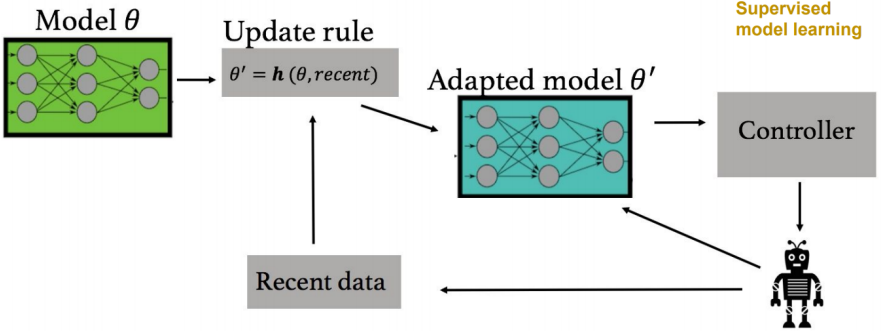


Figure adapted from Anusha Nagabandi

Summary

- The idea of meta-learning is to learn the learning process

Summary

- The idea of meta-learning is to learn the learning process
- We start from skills learned earlier in related tasks, reuse approaches that worked well before. With every skill learned, learning new skills becomes easier, requiring fewer examples and less trial-and-error.