Meta Reinforcement Learning

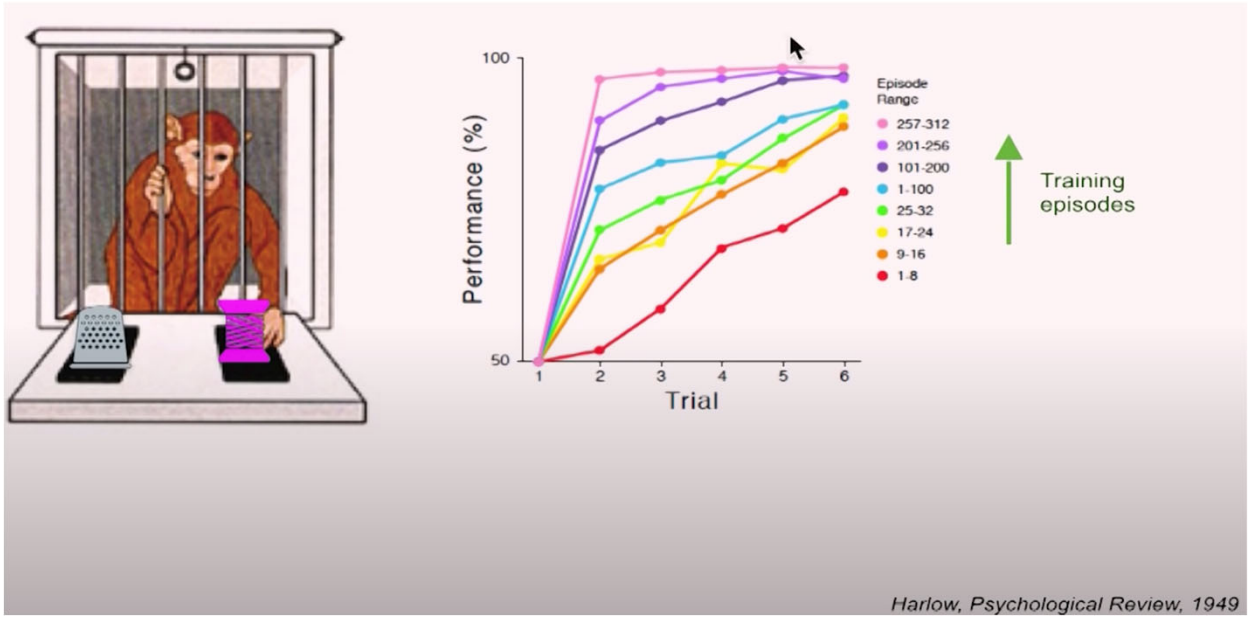
By Harvey Huang

Motivation

Brain, Mind and Markets Laboratory

- Deep RL algorithms: powerful but slow
  - Outperform humans in many tasks.
  - Require large amount of training data.
- The key difference between human vs. deep RL:
  - “Sample efficiency”: the amount of data required for a learning system to attain any chosen target level of performance.
- E.g. How many Go games does the agent have to play to reach a Grand Master level?
  - AlphaGo: 29 million games in 40 days
  - Human: 10 games per day for 60 years = 219,000 games

Motivation

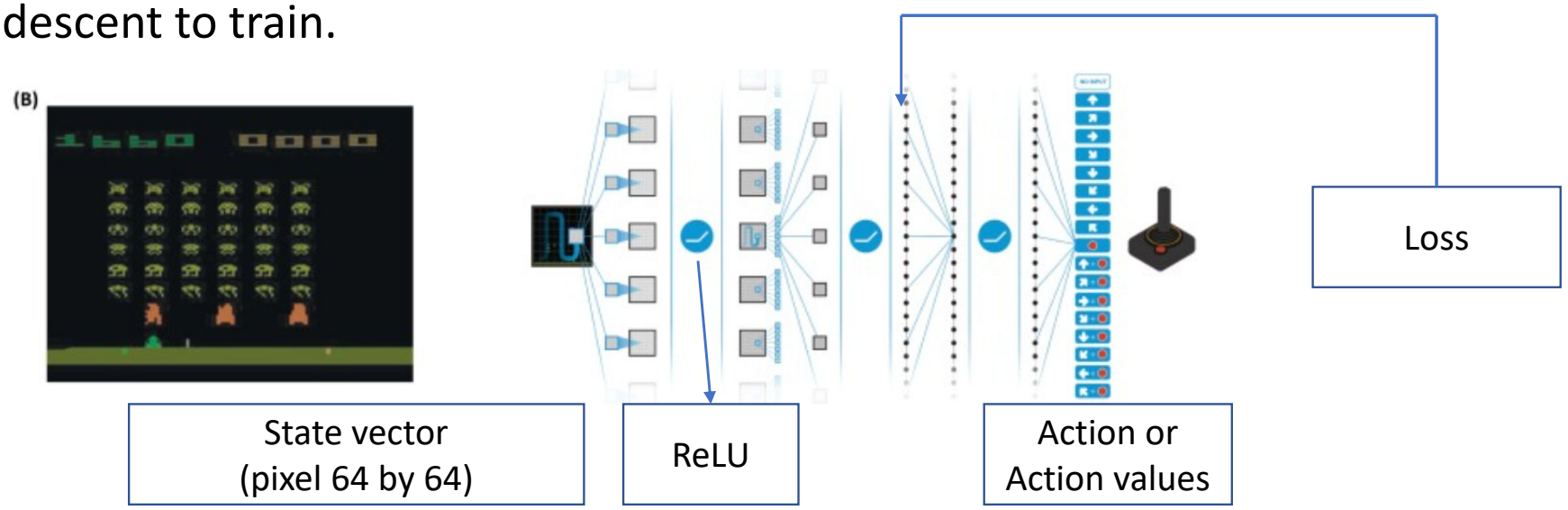


How do we achieve this using a neural network?

Source of slowness?

Brain, Mind and Markets Laboratory

- Source 1: incremental parameter adjustment
  - Vanilla version DQN employed Deep Neural Network and use gradient descent to train.



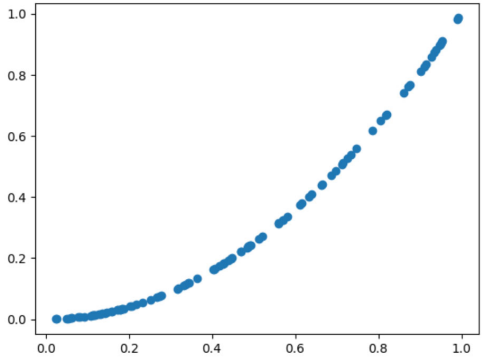
Source of slowness?

- Deep RL training:
  - Gradient descent: broadcast the loss back to every weight (synapse).
- To maximize the ability of generalization,
  - Training has to be slow (small learning rate/step size, e.g. 1e-4).
  - Large learning rate ➔ Catastrophic inference: overwriting the effects of earlier learning.

Source of slowness?

Brain, Mind and Markets Laboratory

- Source 2: weak inductive bias
  - Any learning faces bias-variance tradeoff (of the prediction).
    - A stronger model assumption => high variance but low bias
    - A broader model assumption => low variance but high bias
  - Inductive bias: the level of initial assumptions the learning procedure makes about the patterns to be learned.



Strong inductive bias: Y=ax^2 + b + e
- Perfect fit
- less data required to train
- Hard to generalize to other DGP

Weak inductive bias: neural net
- Slow to train (relatively)
- Need a lot of data
- Can be generalized to other DGP

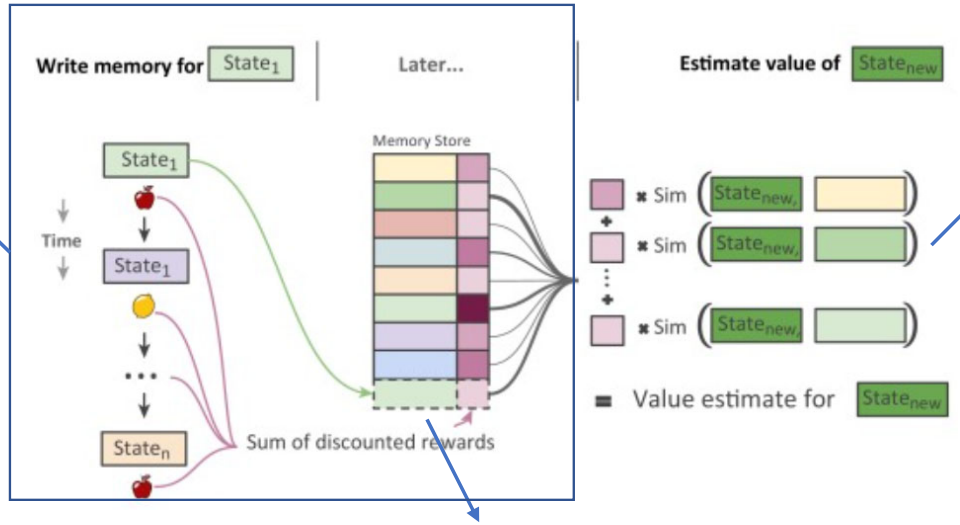
Tackle the slowness

- Incremental parameter adjustment:
  - Episodic RL
- Weak inductive bias:
  - Meta RL: deep RL + learning to learn
- The methods can be combined.

Episodic RL

- Idea: keep a record of past events, use the record directly as a reference point in making new decisions (i.e. episodic memory) so that you don't have to train agents "from scratch").

Pre-trained (recorded) episodic memory



When a new state:

- Weight = similarity("state x", "new state")
- Value of new state = weighted avg of all values

{"state1": discounted reward (state value), "state2": discounted reward}

Meta-RL

- Fast learning requires the learner to go in with a reasonably sized set of hypothesis on patterns of data to be learned.
- The question is how?
- Natural answer: past experience.
  - Meta-learning (machine learning) / learning-to-learn (psychology)

Meta-RL

- There are different ways to inject prior knowledge:
  - E.g. in computer vision, attach a model with prior image information: Geometry, fine-tuned ImageNet features (basically trained neural nets)
  - But this is more data-driven, rather than human driven.
  - So the question is can agents explicitly learn priors from past experience?

The meta-learning problem

Brain, Mind and Markets Laboratory

Supervised Learning:

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

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

Meta Supervised Learning:

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

$$\{(\mathbf{x}, \mathbf{y})_{1:K}\}$$

$$\mathbf{y}_{\text{test}} = f(\mathcal{D}_{\text{train}}, \mathbf{x}_{\text{test}}; \theta)$$

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

Different Dataset batches For different Tasks

Why is this view useful?

Reduces the problem to the design & optimization of f.

The meta-RL problem

Brain, Mind and Markets Laboratory

Reinforcement Learning:

Inputs: ~~$\mathbf{x}$~~ $\mathbf{s}_t$ Outputs: ~~$\mathbf{y}$~~ $\mathbf{a}_t$

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

$\mathbf{a}_t = \pi(\mathbf{s}_t;\theta)$

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

$\{(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})\}$

Meta Reinforcement Learning:

Inputs: $\mathcal{D}_{\text{train}}$ $\mathbf{s}_t$ Outputs: $\mathbf{a}_t$

k rollouts from π

$\mathbf{a}_t = f(\mathcal{D}_{\text{train}}, \mathbf{s}_t;\theta)$

Data: $\{\mathcal{D}_i\}$

dataset of datasets collected for each task

Design & optimization of *f* *and* collecting appropriate data (learning to explore)

Key challenges

1. Network design and training
2. How to collect Dtrain because this is also in the control of meta-RL.

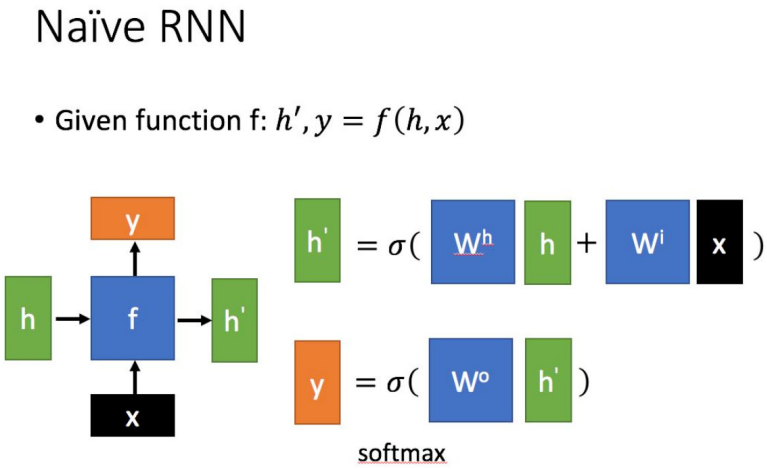
Meta-RL: model/implementation

- Blackbox/contextual-based approach
- Optimization-based approach

Blackbox approach

Brain, Mind and Markets Laboratory

- A neural network structure with “memory of states/events”.
- e.g. RNN/LSTM.
- We use RNN as an example:
  - X: state info (vector)
  - h: hidden state (h’: next hidden state)
  - Y: values/action (depends)
- Harvey:
  - Trained weights emphasize how each neuron should react to signals.
  - Hidden states are like information interpreted and memorized by the agents.



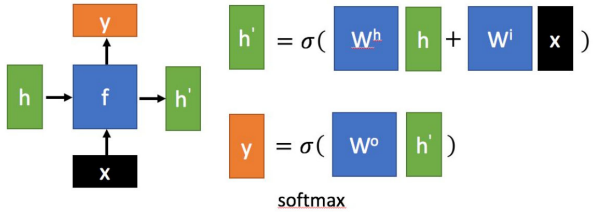
Blackbox approach

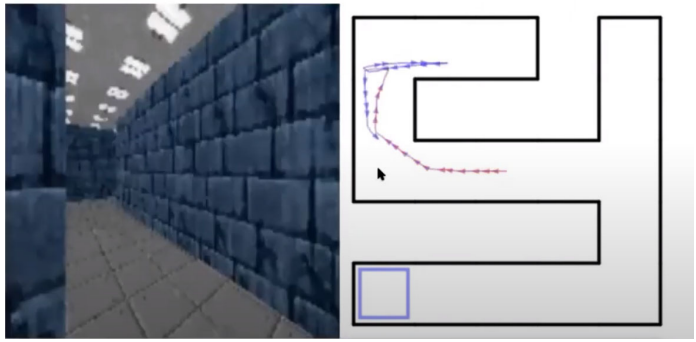
Brain, Mind and Markets Laboratory

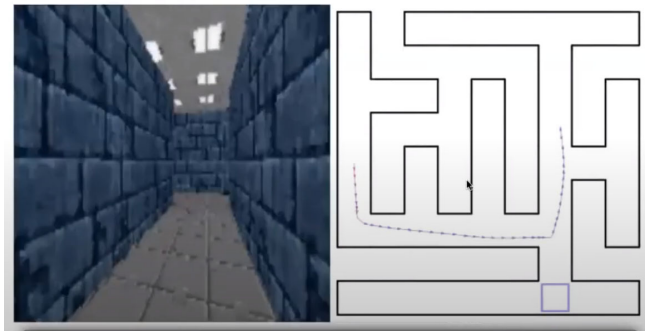
- Say you train this on a maze task.
- You will get all hidden (cell) states h and weights w.
- You want to look at the performance on similar tasks.

Naïve RNN

- Given function f: $h', y = f(h, x)$





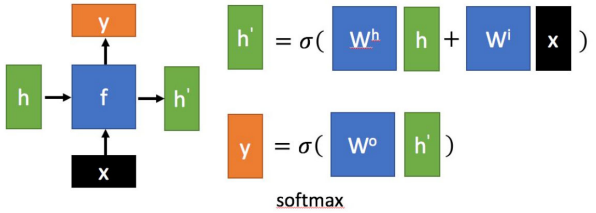


Blackbox approach

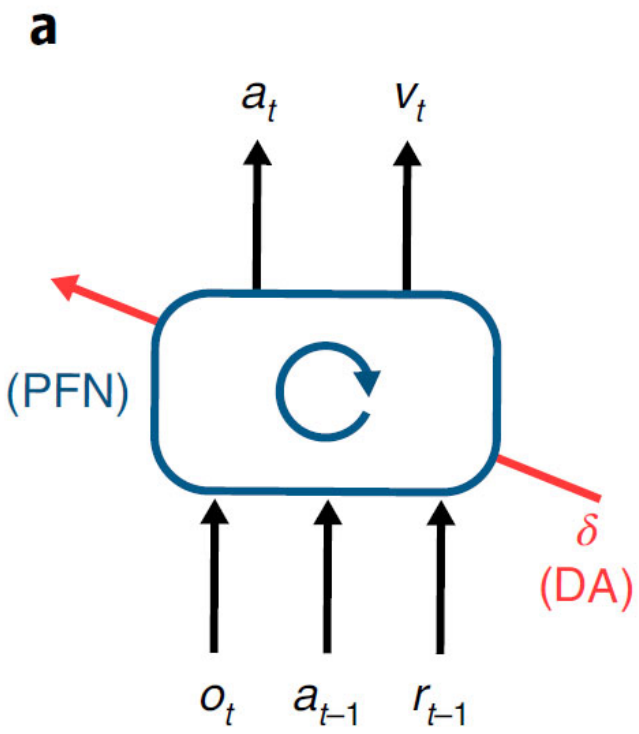
- Let's fixed the weights.
  - No more updates on synapse.
  - No more prediction error update.
- Reset the hidden state h.
- Train on the new task (similar task).
- Results show that the system is learning.
  - Learning happens exclusively in the activation dynamics within these recurrent networks.

Naïve RNN

- Given function f: $h', y = f(h, x)$



Meta-RL architecture



The prefrontal network (PFN), including sectors of the basal ganglia and the thalamus that connect directly with PFC, is modeled as a recurrent neural network (LSTM), with synaptic weights adjusted through an RL algorithm driven by phasic dopamine (DA)

Blackbox approach

Brain, Mind and Markets Laboratory

- We can fix the weights, reset hidden state.
- Alternatively, we can also store hidden states.
  - Just like humans: “Ah, I think I’ve seen this (task) before, looks similar.”
    - Recall and update “what”: we can map the new task on the old task.
    - Recall and update “how”: we know how to deal with the new task by leveraging from the skills we have learned before and perform slight adjustments on them to the new task.

Combined with episodic memory

- Episodic (memory) variant
  - {“state1”: discounted reward (state value), “state2”: discounted reward}
  - Customize the key-value pairs.
    - Keys: pixel matrix (or some task embeddings).
    - Values: network cell states “h” (hidden information/agent interpretation).
  - Retrieve the values based on similarity measure.
    - And then reinstate (basically a smart way of plugin) the information into the new network.

Optimization based approach

Brain, Mind and Markets Laboratory

- Suppose we have some pre-trained neural network f.
- We can attach new layers (depends on the specific task), and “fine tune” the new layers (as well as the pre-trained layers) with the tasks.
- The concept of “re-use” ⇔ built-in inductive bias.

Fine-tuning
[test-time]

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

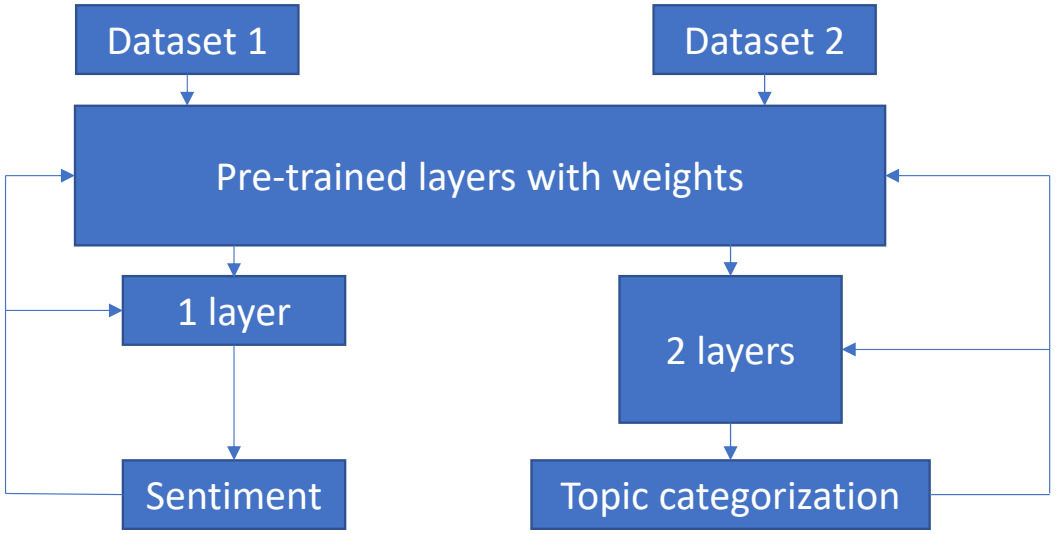
pretrained parameters

training data for new task

Optimization based approach

Brain, Mind and Markets Laboratory

- The concept of “pre-train” is quite popular in CV and NLP.
- E.g. GPT-3, BERT models.
- Fine-tune training takes much less data.

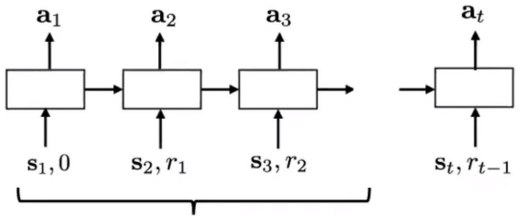


Pros and Cons

Brain, Mind and Markets Laboratory

Blackbox approach

$$\mathbf{a}_t = f(\mathcal{D}_{\text{train}}, \mathbf{s}_t; \theta)$$



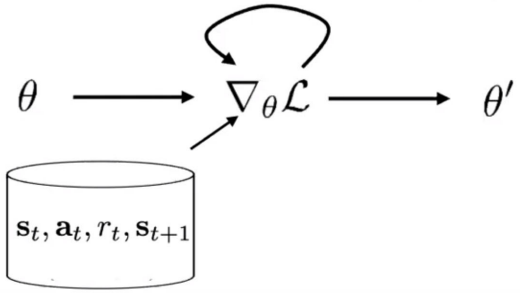
network implements the "learned learning procedure"

+ general & expressive
+ a variety of design choices in architecture

-- complex model for complex task of learning
-- impractical data requirements

Optimization-based approach

$$\mathbf{a}_t = f(\mathbf{s}_t; \theta - \alpha \nabla_\theta \mathcal{L}(\mathcal{D}_{\text{train}}))$$



+ inductive bias of SGD built in
+ model-agnostic (plug & play with your existing architecture)

-- RL gradients often not very informative (e.g. policy gradients, Bellman error)

Resources

- Reinforcement learning: Fast and slow by Matthew Botvinick
  - https://www.youtube.com/watch?v=b0LddBiF5jM
- Matthew Botvinick, DeepMind: Meta-learning in brains and machines
  - https://www.youtube.com/watch?v=otjR6iGp4yU&t=1247s
- DLRL summer school 2020 meta-RL by Chelsea Finn:
  - https://www.youtube.com/watch?v=c0vSwglRY4w
- Learning to learn with gradients by Chelsea Finn:
  - https://ai.stanford.edu/~cbfinn/_files/dissertation.pdf
- Prefrontal cortex as a meta-reinforcement learning system
  - https://www.nature.com/articles/s41593-018-0147-8
- Reinforcement Learning, Fast and Slow
  - https://www.cell.com/trends/cognitive-sciences/fulltext/S1364-6613(19)30061-0
- A collection of meta-RL papers with code:
  - https://github.com/metarl/awesome-metarl