Deep reinforcement learning

Sequential decision making with large state spaces

Cathy Wu

1.041/1.200 Transportation: Foundations and Methods

Readings

1. Morales, Miguel. **Grokking deep reinforcement learning**. 2020. [URL]

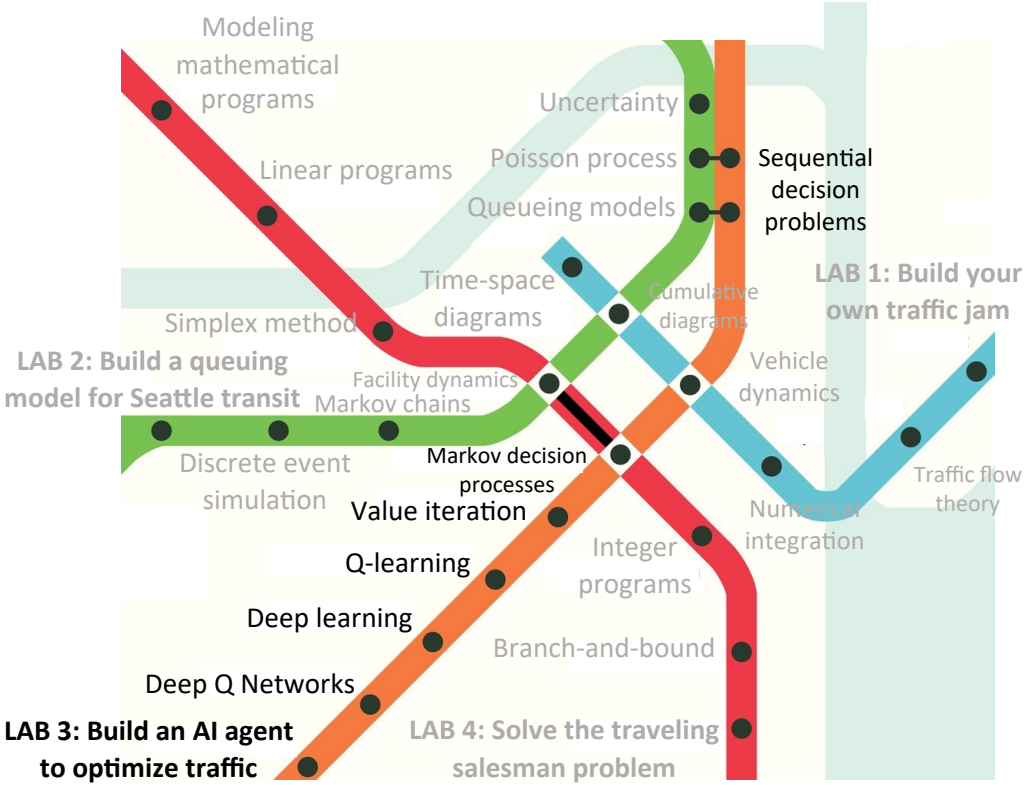
   - Chapter 8: Introduction to value-based deep reinforcement learning.
   - Chapter 9: More stable value-based methods

Unit 3: Machine learning for traffic control

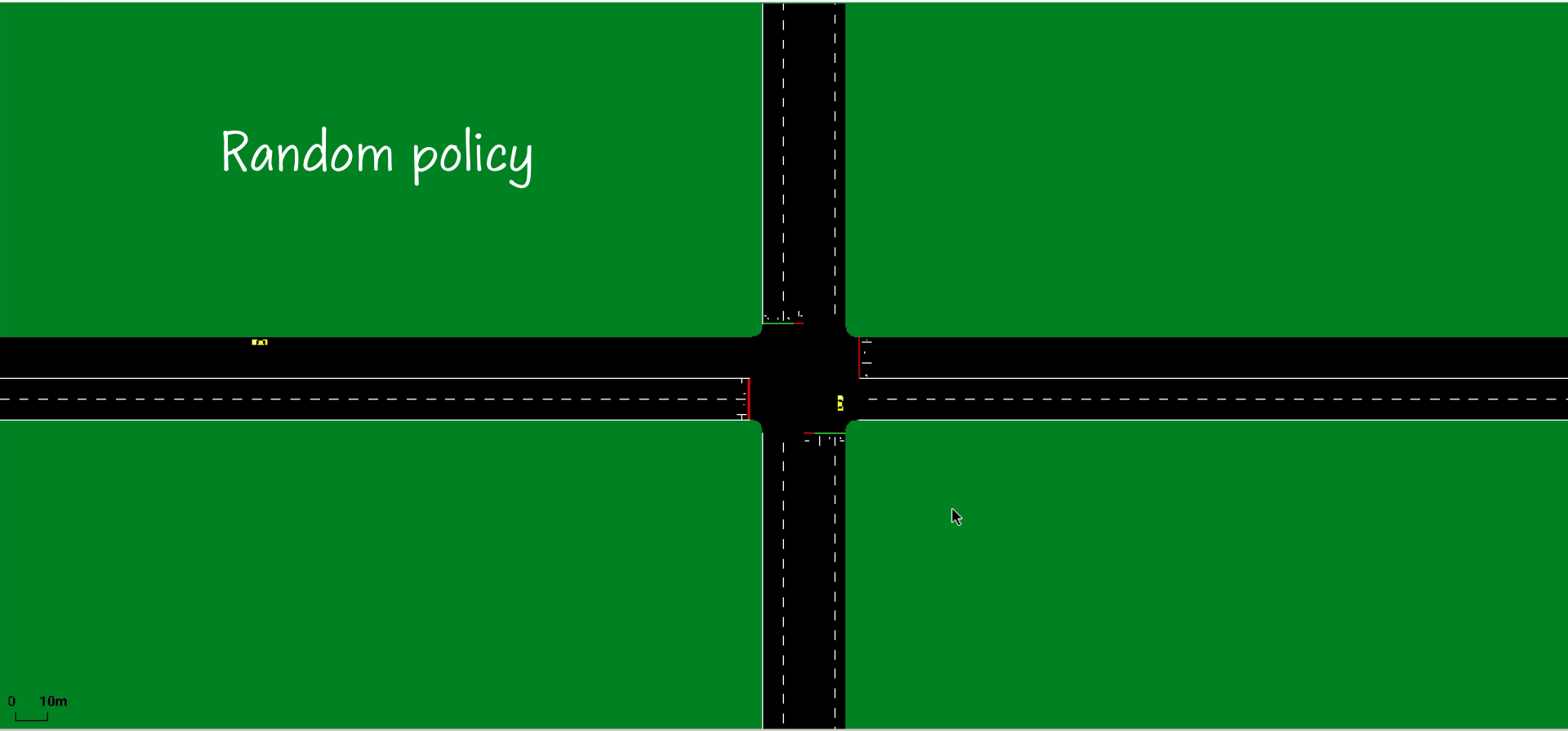
Unit 3

Optimizing

Multi-stage



CL3: Build an AI agent to optimize traffic



DP for traffic signal control: challenges

(Today) ~~Updates all states (even impossible/unlikely)~~

(Lectures 17-18) **Large state space (e.g., $|S| = 2^{80}$)**

(Today) ~~Long horizon (e.g., $T = 5400$)~~

(Today) ~~Reward sparse (often zero)~~

(Today) ~~Check all next states to select next action~~

$V_T(s_T) = r_T(s_T)$
for $t = T-1, \dots, 0$ **do**
  for $s_t \in \mathcal{S}_t$ **do**
    $V_t(s_t) = \max_{a_t \in \mathcal{A}_t(s_t)} r_t(s_t, a_t) + \mathbb{E}_{s_{t+1} \sim P(\cdot|s_t, a_t)}[V_{t+1}(s_{t+1})]$
end for

- Dynamic programming: O(|S|²|A|T) = $2^{80\times 2} \times 4 \times 5400$
  - Not so efficient ☹
- Parts that are (surprisingly) OK
  - DP recursion
  - Action space usually small

Outline

1. Deep Q Networks (DQN)
2. DQN demo
3. Transfer learning

Outline

1. **Deep Q Networks (DQN)**
2. DQN demo
3. Transfer learning

a.k.a. fitted Q-iteration with deep neural networks

Main idea: iteratively minimize an (error) function.

$$\min_\theta f_\theta(x) = \min_\theta \sum_{k=0}^{N} \left(\tilde{Q}_\theta(s_k, a_k) - R_k\right)^2$$

where $x = (s_k, a_k, R_k)_{k \in [N]}$

and $R_k \coloneqq r(s,a) + \gamma \max_{a'} Q_k(s', a')$

via Backprop

$$\nabla_\theta f_\theta(x) = 2 \sum_{k=0}^{N} \left(\tilde{Q}_\theta(s_k, a_k) - R_k\right) \nabla_\theta \tilde{Q}_\theta(s_k, a_k)$$

Gradient descent algorithm:

1. Pick a starting θ_0
2. Repeat:
   1. Compute descent direction: $-\nabla_\theta f_\theta(x)$
   2. Step in the direction: $\theta_{t+1} = \theta_t - \eta \nabla_\theta f_\theta(x)$
   3. Check if we should stop

Recall (L16)

- Q-learning update:

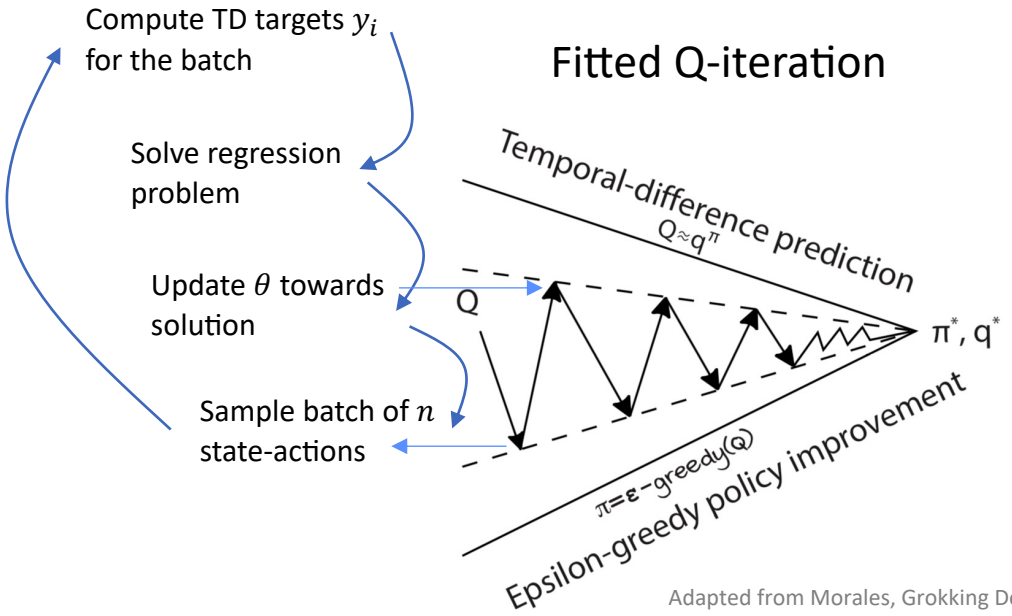
$$Q_{k+1}(s,a) = Q_k(s,a) + \eta_k \left(r(s,a) + \gamma \max_{a'} Q_k(s', a') - Q_k(s,a)\right)$$

Caveat: Loss with deep neural networks is not convex

Implication: (S)GD might get stuck in local optima

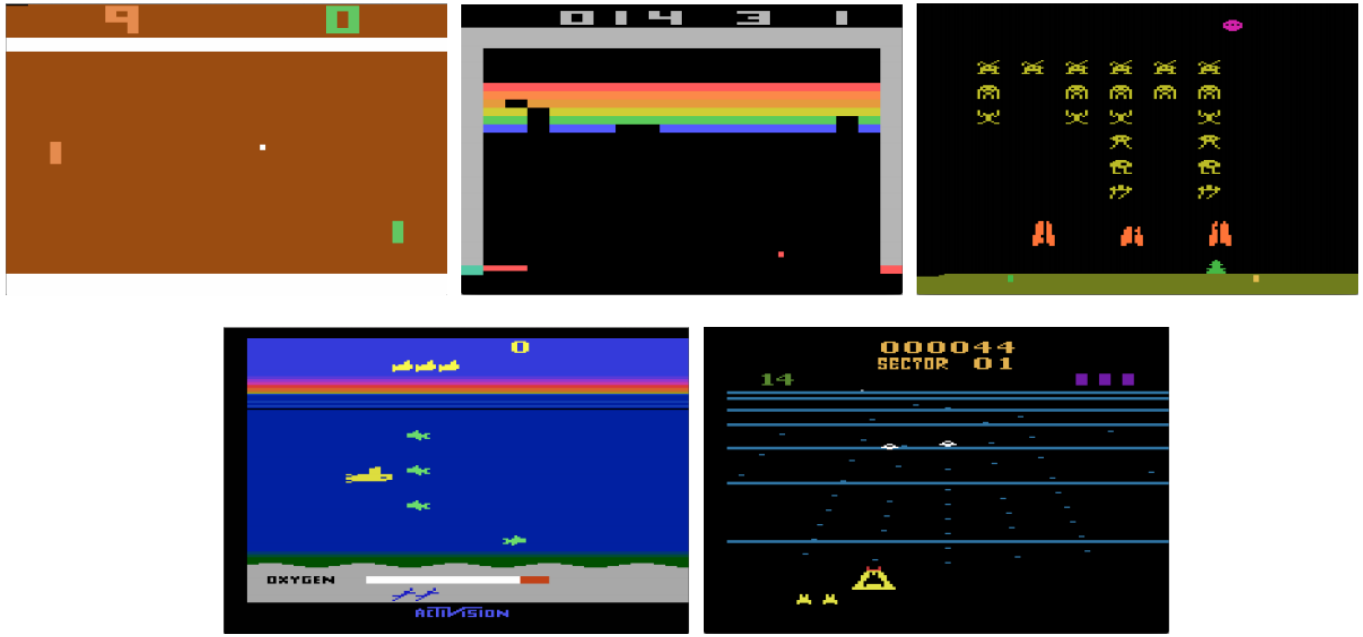
Loss Landscape Explorer | Explore real loss landscapes of deep learning optimization processes

Fitted Q-iteration (approximate Q-iteration)

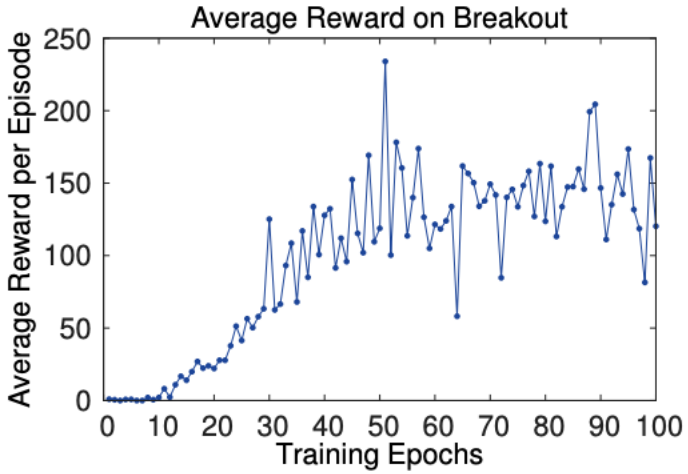


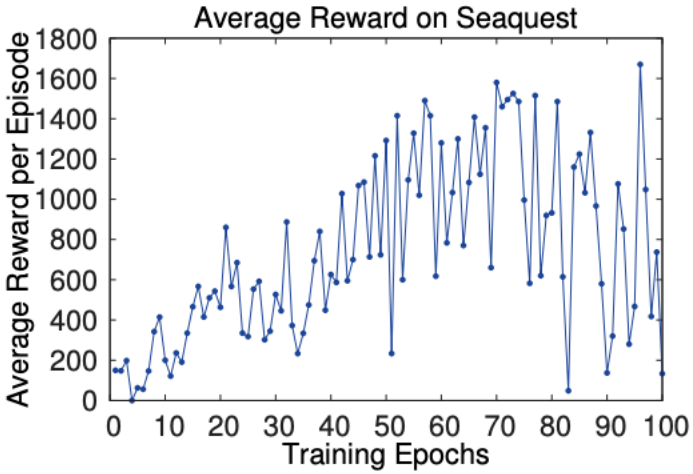
Adapted from Morales, Grokking Deep Reinforcement Learning, 2020.

Fitted Q-iteration applied to ATARI Games



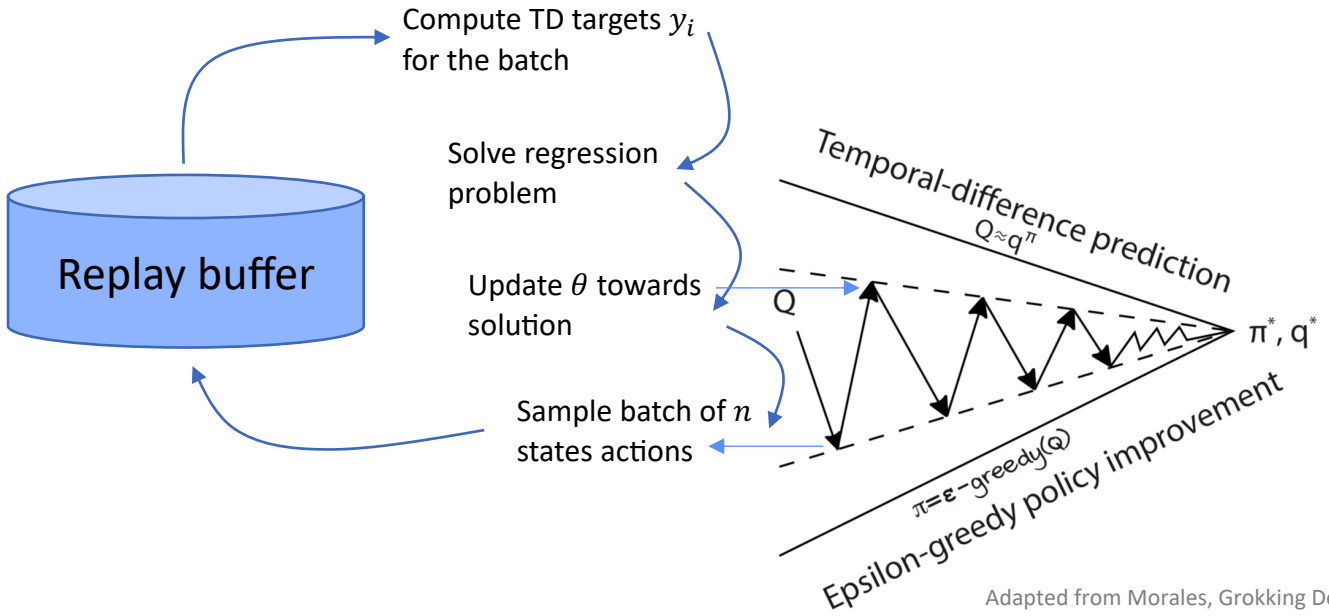
Still doesn't quite work. Why?





- Training is not stable.
- A big reason: the training data is not representative of all state-action pairs.
- Recall: garbage in, garbage out
  - The state-action space is HUGE.
  - The RL agent "collects its own data," wherever it happens to be.
  - If that data is bad, then the result is bad too (and can further worsen future collection of data).

Solution: increase data diversity!



Adapted from Morales, Grokking Deep Reinforcement Learning, 2020.

- De-correlates samples
- Increased diversity in data → less likely that the data overall is bad for learning

Another issue: chasing a moving target

Q-iteration loss using bootstrapped target

- Iteration i

$$\tilde{L}(s_{1:n}, a_{1:n}, y_{1:n}; \theta)$$

$$= \sum_t \left(Q_\theta(s_t, a_t) - r_t - \gamma \max_{a'} Q_{\theta_i}(s_{t+1}, a') \right)^2$$

- Iteration $i+1$

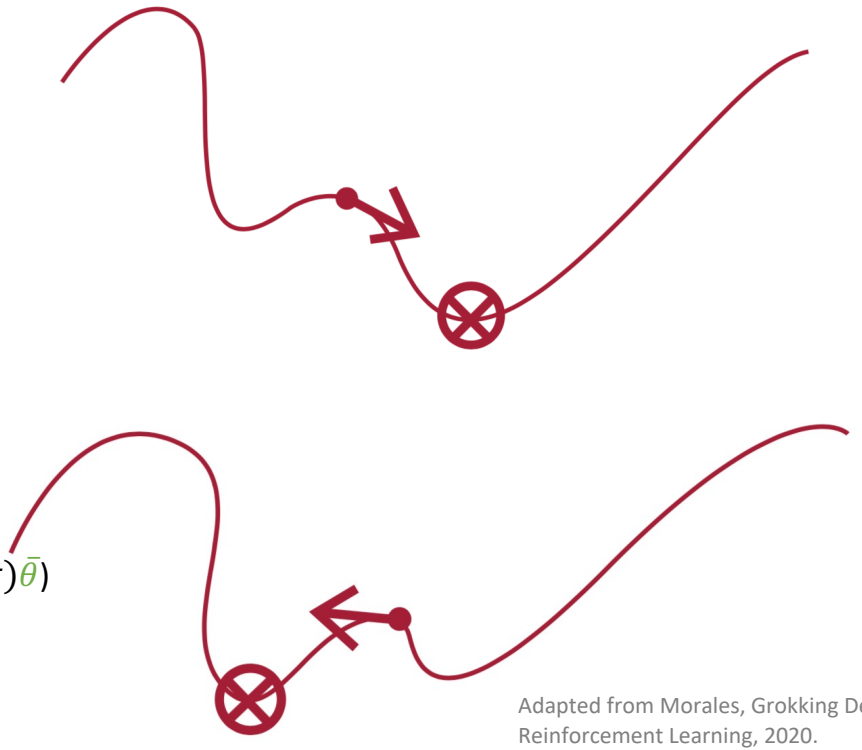
$$\tilde{L}(s_{1:n}, a_{1:n}, y_{1:n}; \theta)$$

$$= \sum_t \left(Q_\theta(s_t, a_t) - r_t - \gamma \max_{a'} Q_{\theta_{i+1}}(s_{t+1}, a') \right)^2$$

- Solution: change the target slowly
(e.g., $\bar{\theta} \leftarrow \theta$ every 1000 steps or $\bar{\theta}' \leftarrow \tau\theta + (1-\tau)\bar{\theta}$)

$$\tilde{L}(s_{1:n}, a_{1:n}, y_{1:n}; \theta)$$

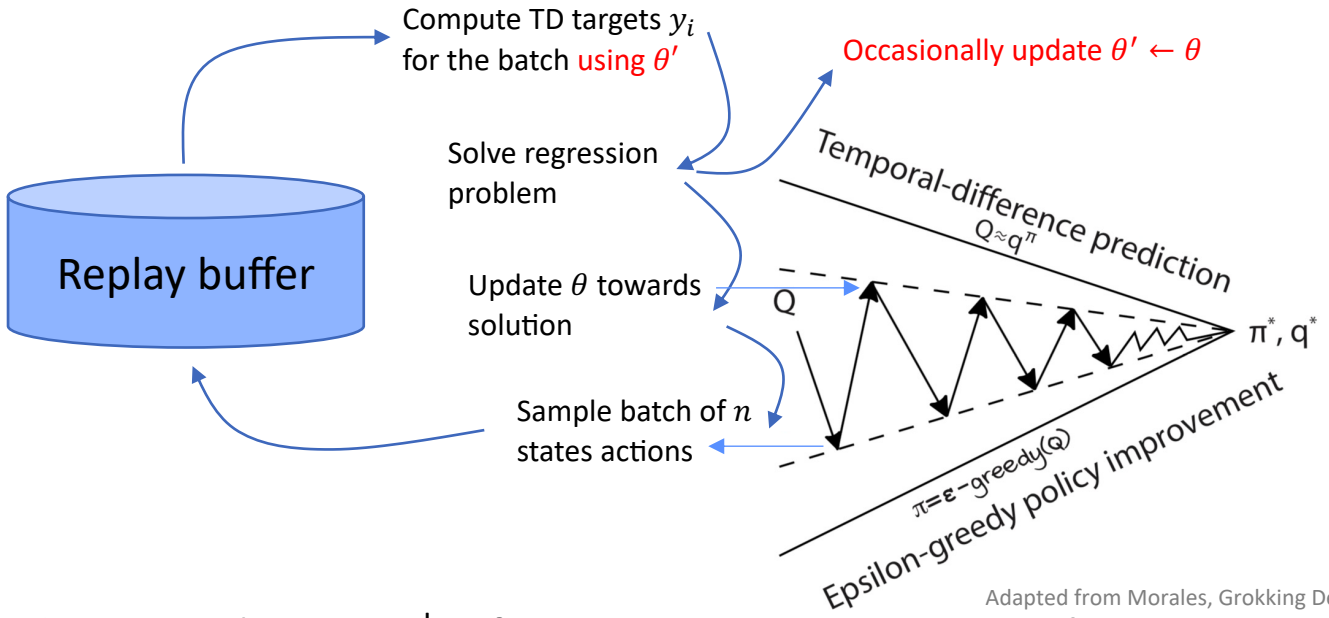
$$= \sum_t \left(Q_\theta(s_t, a_t) - r_t - \gamma \max_{a'} Q_{\bar{\theta}}(s_{t+1}, a') \right)^2$$



Adapted from Morales, Grokking Deep Reinforcement Learning, 2020.

"Target network"

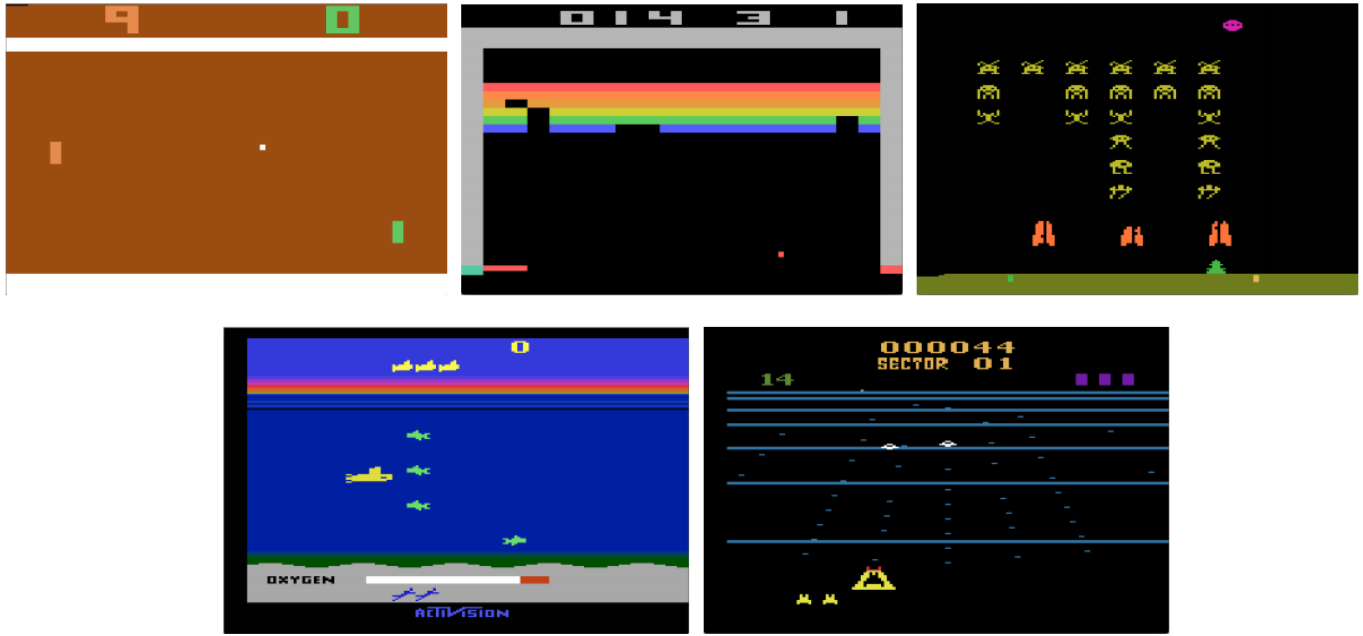
DQN algorithm



Adapted from Morales, Grokking Deep Reinforcement Learning, 2020.

- Target network $Q_{\theta'}$: introduce a 2nd Q function
- Update it slowly → balance avoiding the moving target issue and keeping TD targets accurate

DQN applied to ATARI Games



DQN – Atari

Image preprocessing: grey-scale, crop to 84x84

$$\phi\left(\quad\right)$$

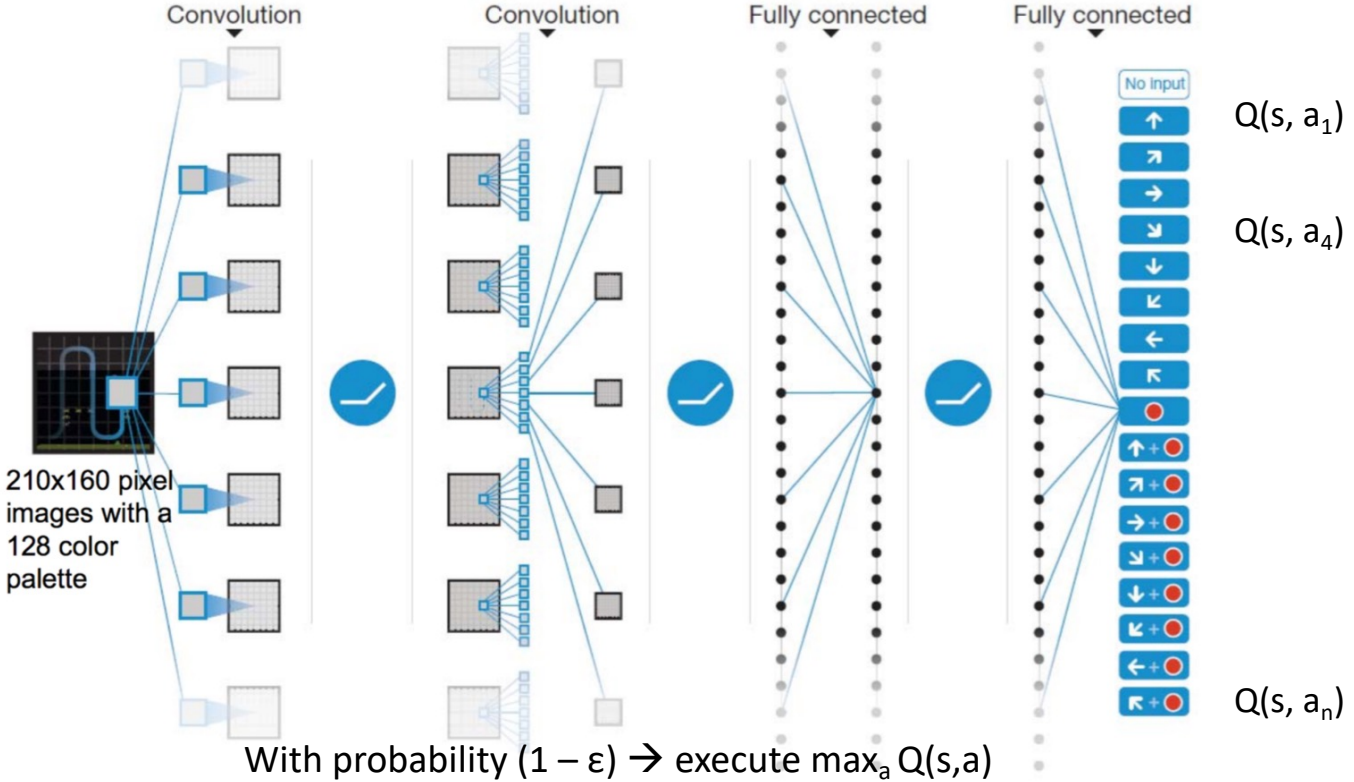
DQN – Atari

State definition: 4 last frames

$$s_t = \left\{ \text{  } \; a_{t-1}, r_{t-1}, \; \ldots \; a_{t-4}, r_{t-4} \right\}$$

DQN – Atari

Action-value function: deepNet with as many heads as actions

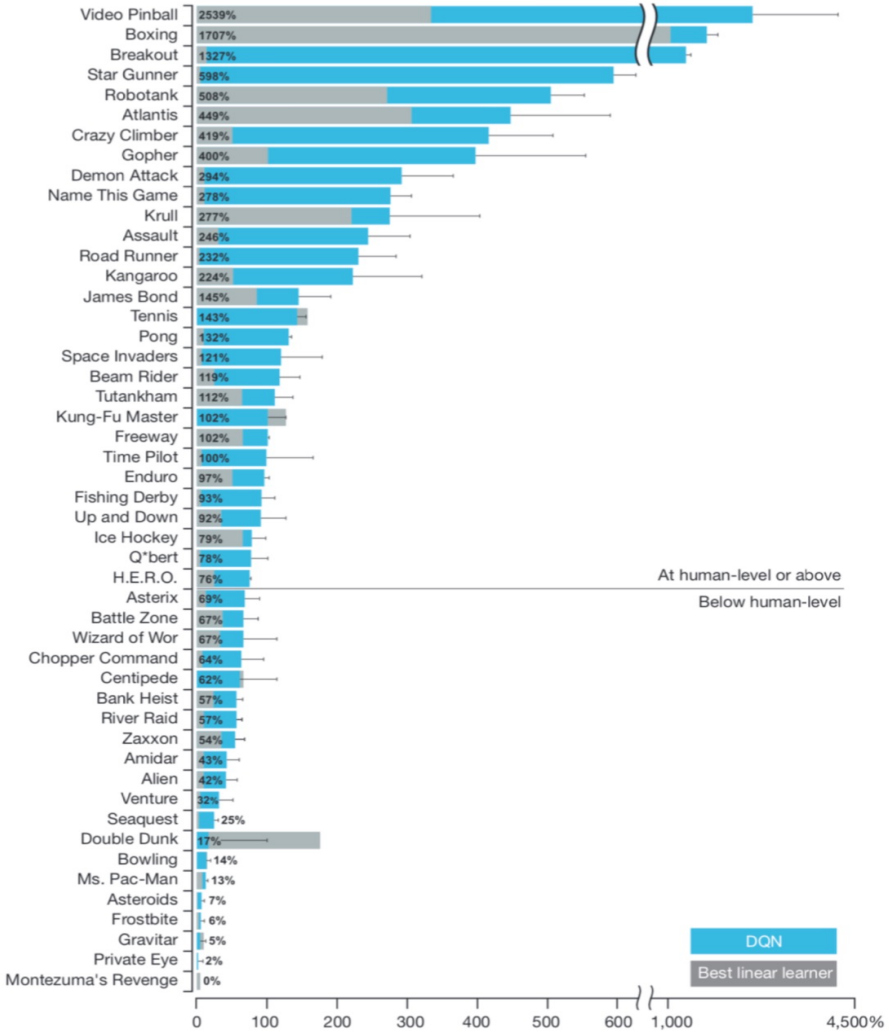


With probability $(1 - \varepsilon)$ → execute $\max_a Q(s,a)$

With probability ε → execute random action

DQN – Atari

Performance



DQN – Atari

Ablation

DQN

Game	With replay, with target Q	With replay, without target Q	Without replay, with target Q	Without replay, without target Q
Breakout	316.8	240.7	10.2	3.2
Enduro	1006.3	831.4	141.9	29.1
River Raid	7446.6	4102.8	2867.7	1453.0
Seaquest	2894.4	822.6	1003.0	275.8
Space Invaders	1088.9	826.3	373.2	302.0

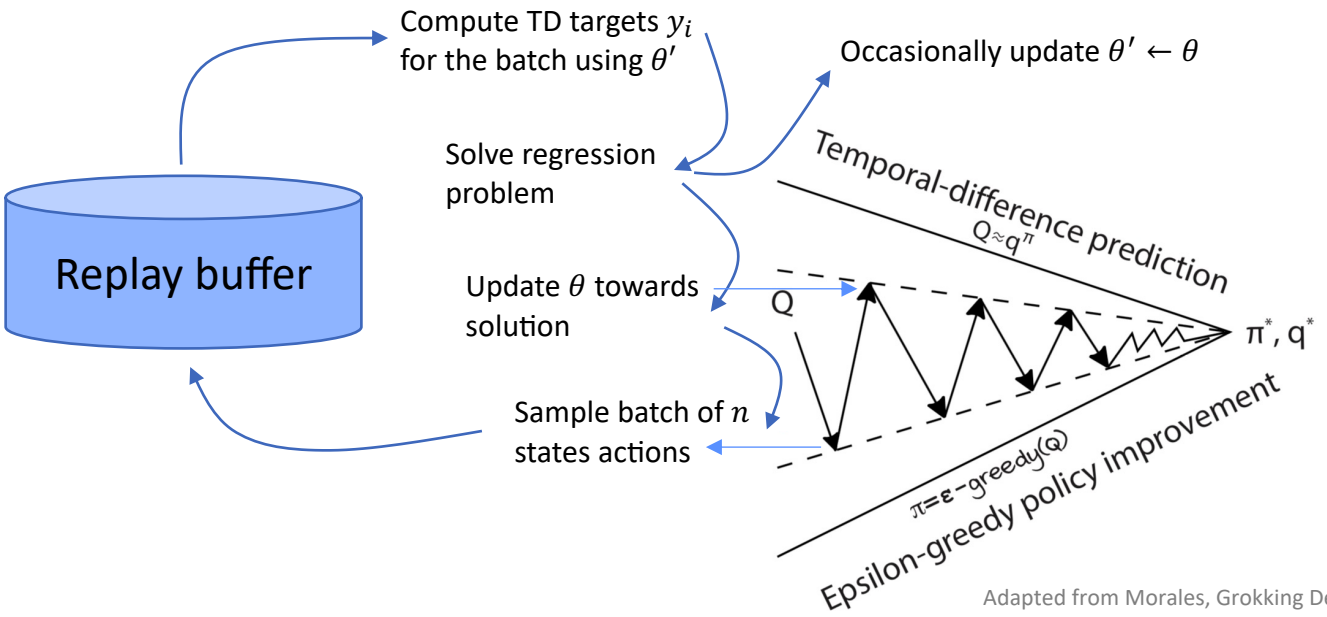
Outline

1. Deep Q Networks (DQN)
2. **DQN demo**
3. Transfer learning

Deep RL Demo

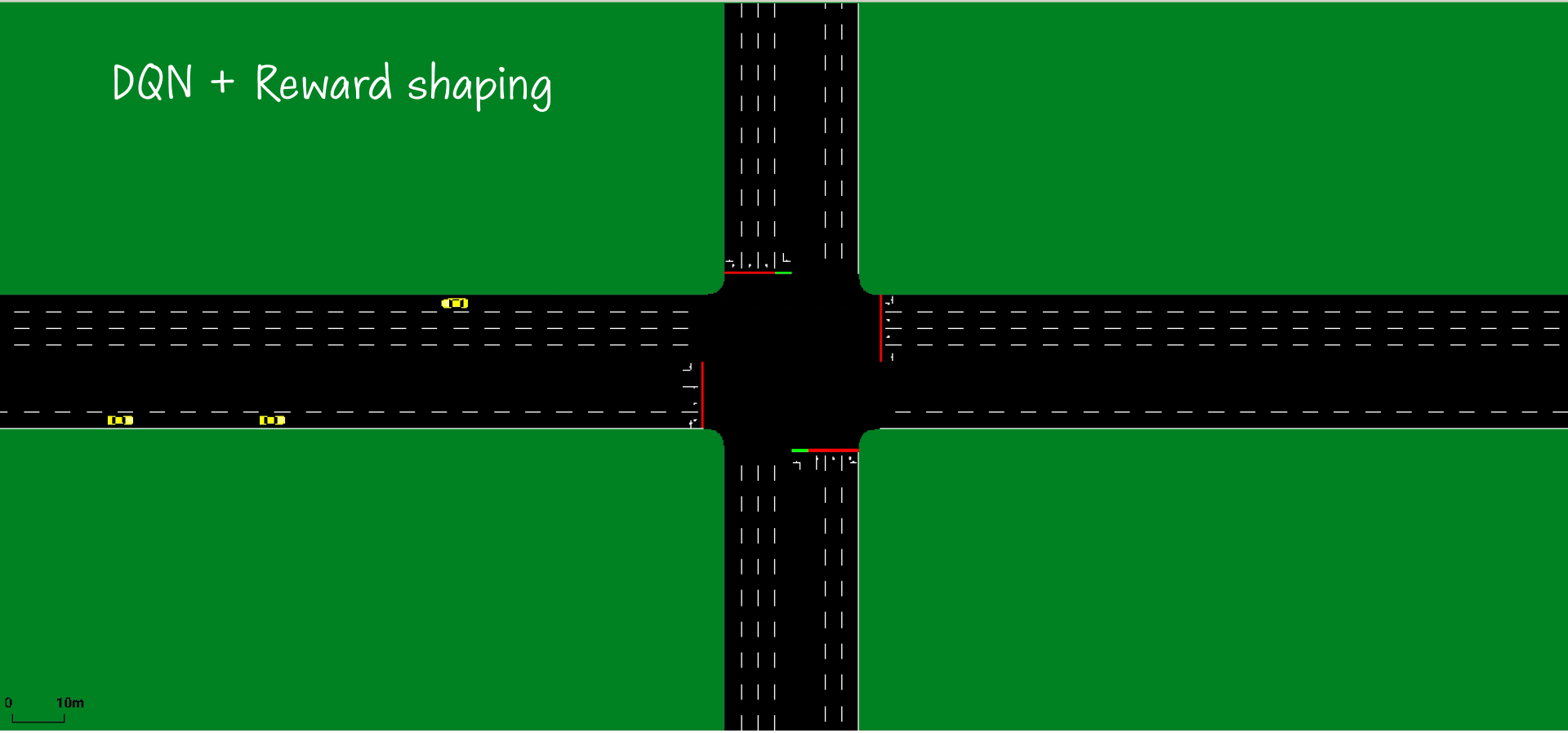
Link

DQN algorithm



Adapted from Morales, Grokking Deep Reinforcement Learning, 2020.

CL3: Build an AI agent to optimize traffic

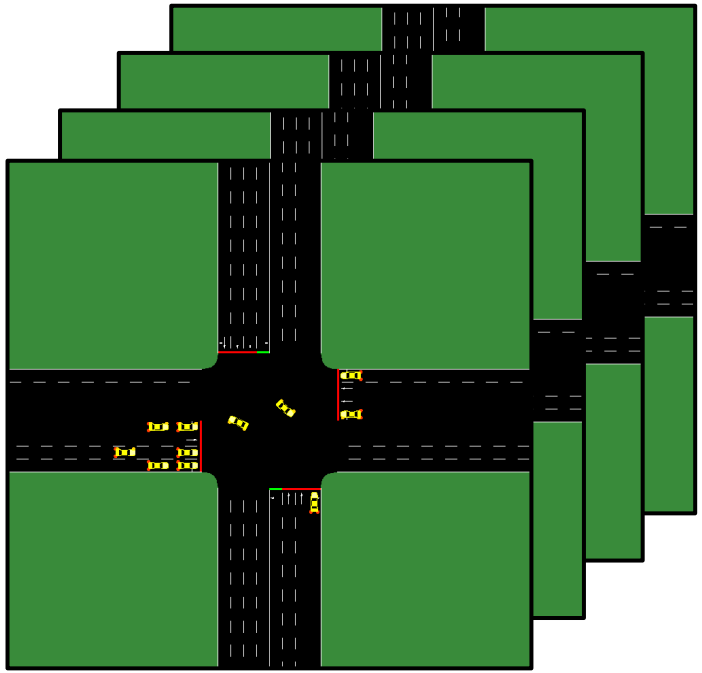


Outline

1. Deep Q Networks (DQN)
2. DQN demo
3. **Transfer learning**

Generalization in RL

- Now consider a set of intersections, which may vary across different characteristics
- Ex. Speed limit, approach length

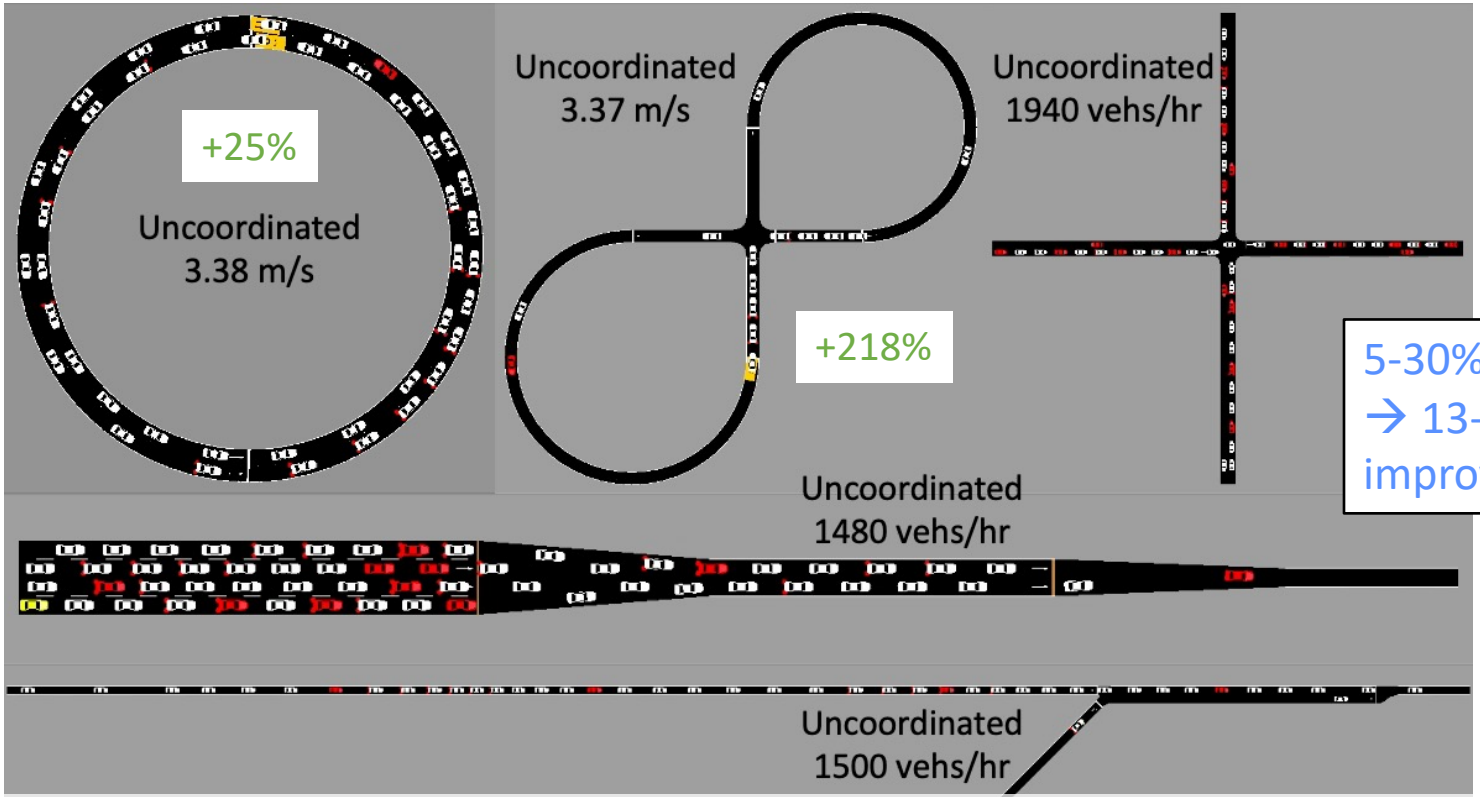


Question 4: Generalization with zero-shot transfer learning [25 Points]

For this last part, we consider how reinforcement learning will be used in practice in transportation. A beautiful but also frustrating aspect of this field is just how complex and varied traffic systems are. It is believed that while reinforcement learning works well in specific settings, they do not necessarily generalize to settings even if they seem quite similar (see our paper on this unfortunate result). Given the vast diversity of traffic settings, it is impractical (and intractable) to train an agent for every setting. This is where *zero-shot transfer learning* comes in. In this problem, we consider a simplified version of this challenge by focusing on just one way in which traffic systems can vary: different speed limits.

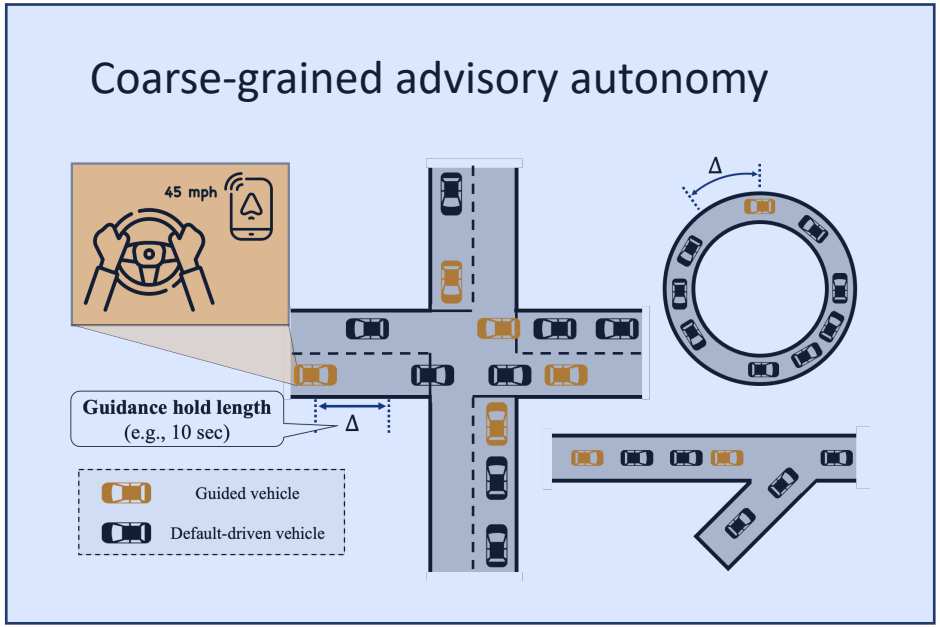
Problem setting. Consider now that you are an engineer at a transportation authority vested with the responsibility to regulate traffic signals across the city. Different intersections have different speed limits (suppose for simplicity that all other aspects of the intersections are the same). Training separate models for each potential speed limit is computationally expensive and impractical. Instead, you consider leveraging zero-shot transfer learning to employ a small number of models trained on specific speed limits to achieve good performance across the full range of speed limits.

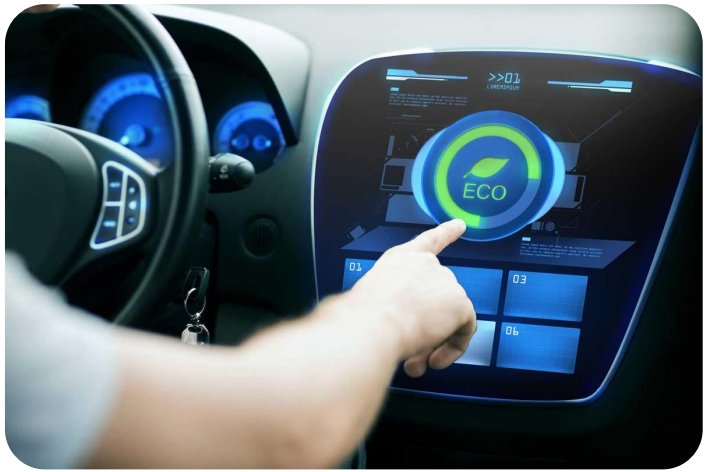
Mixed autonomy can substantially boost traffic flow (2016–2022)



Wu, Kreidieh, Vinitsky, Bayen, **Emergent behaviors in mixed-autonomy traffic**, in *1st Annual Conference on Robot Learning (CoRL)*, PMLR, 2017.
Wu, Kreidieh, Parvate, Vinitsky, Bayen, **Flow: A modular learning framework for mixed autonomy traffic**, *IEEE Transactions on Robotics (T-RO)*, 2021.
Vinitsky, et al. Wu, Bayen. **Benchmarks for reinforcement learning in mixed-autonomy traffic**, in *2nd Annual Conference on Robot Learning (CoRL)*, PMLR, 2018.
Yan, Kreidieh, Vinitsky, Bayen, Wu, **Unified automatic control of vehicular systems with reinforcement learning**, *IEEE Transactions on Automation Science and Engineering (T-ASE)*, 2022.

Consider a variant of mixed autonomy





Dynamic eco-driving at signalized intersections

[1] Reducing fuel emissions with innovative tech, 2018. https://www.eastpoint.co.uk/case-studies/glosa/

A cheap way to coordinate vehicles is to give drivers an app

Level 0
Fully manual

Level 1
Driver assistance

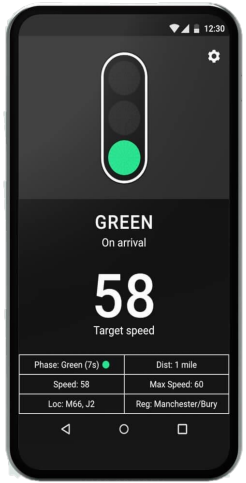
…

Level 4
High automation

Level 5
Fully autonomous

Driver advisories: the lowest cost, near-term form of autonomy

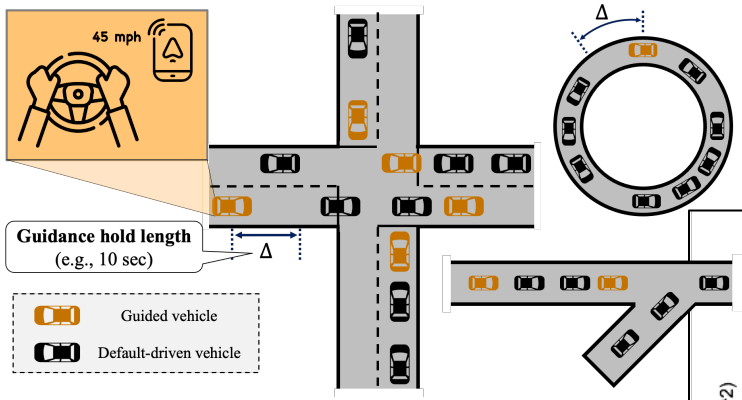
"Level 0.5"



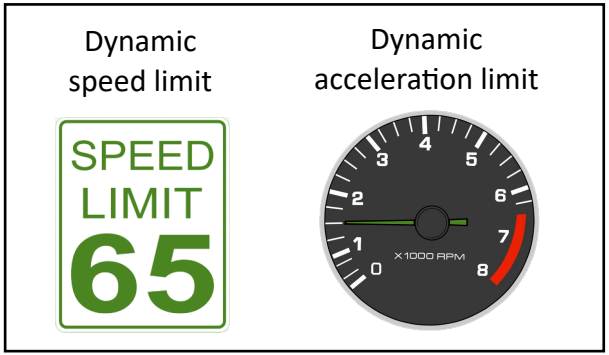
GLOSA Demo app [1]

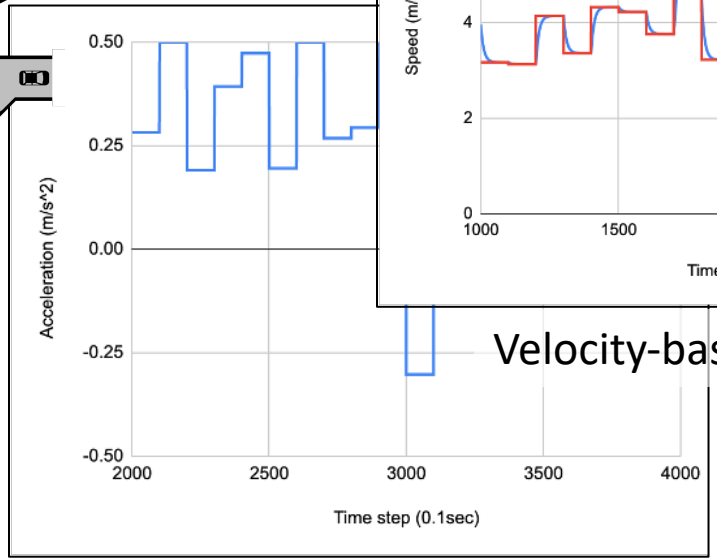
[1] Reducing fuel emissions with innovative tech, 2018. https://www.eastpoint.co.uk/case-studies/glosa/

Traffic optimization with coarse-grained advisory autonomy

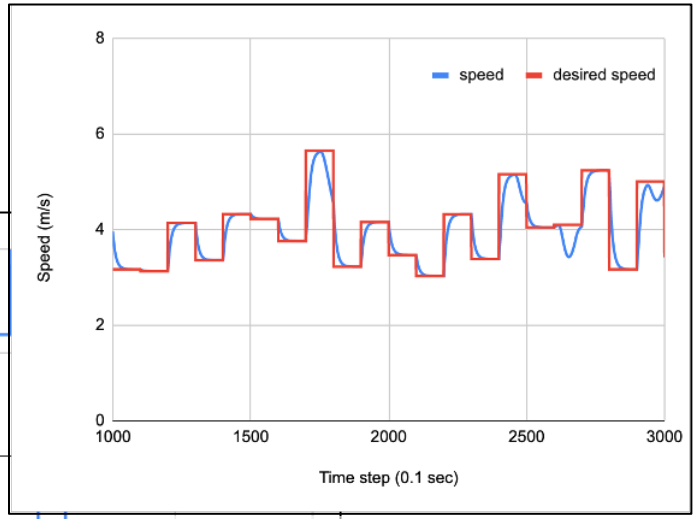


Advisory type:





Acceleration-based advisories

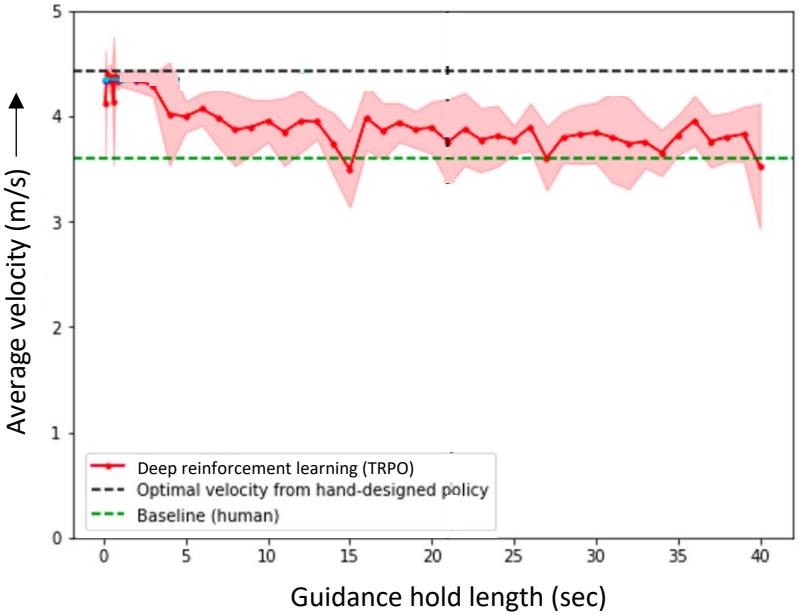


Velocity-based advisories

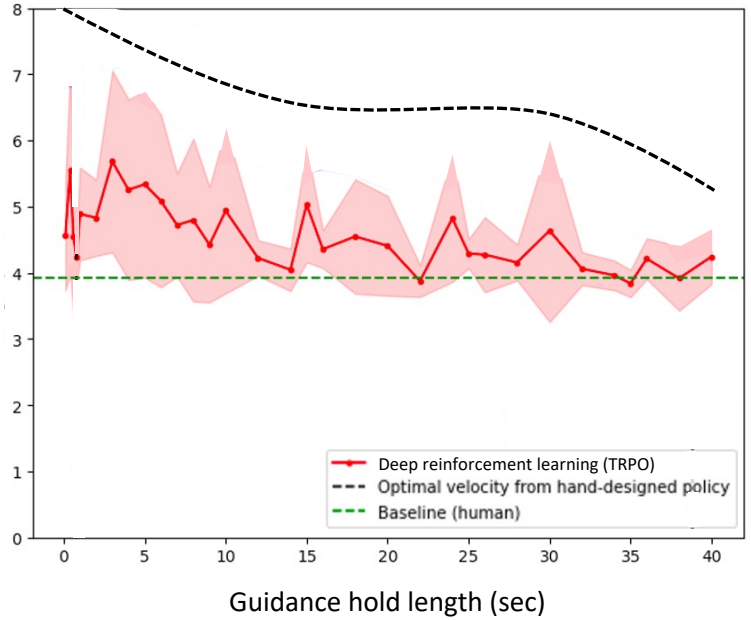
Cho, Li, Kim, Wu. **Temporal Transfer Learning for Traffic Optimization with Coarse-grained Advisory Autonomy**. In preparation for T-RO.

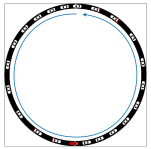
Non-robustness in coarse-grained advisory autonomy

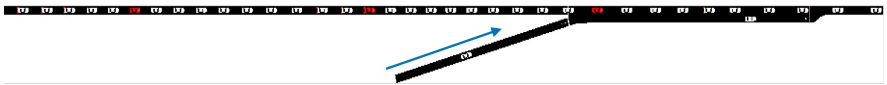
Model highway (acceleration guidance)



Highway ramp (velocity guidance)







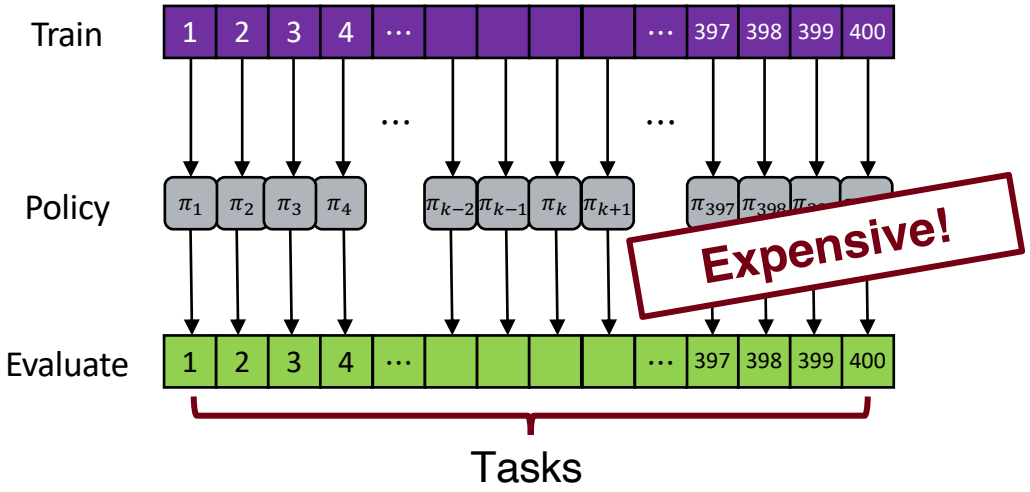
Cho, Li, Kim, Wu. **Temporal Transfer Learning for Traffic Optimization with Coarse-grained Advisory Autonomy**. In preparation for T-RO.

Motivation

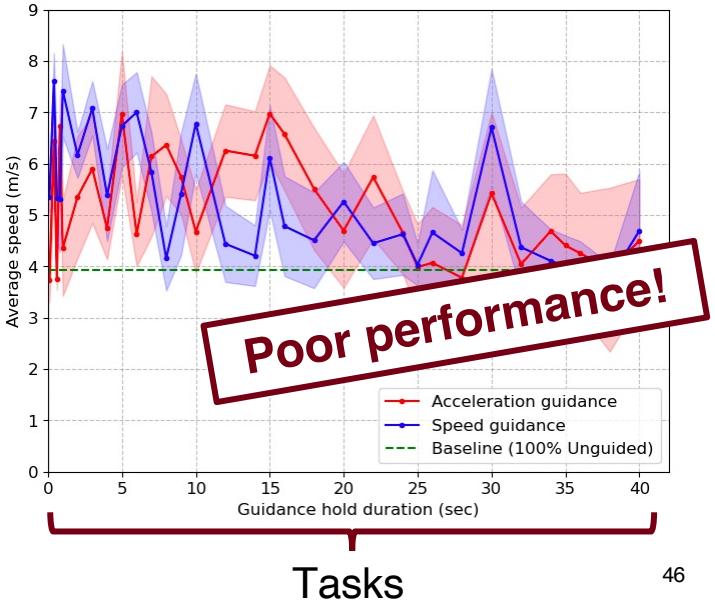
Deep reinforcement learning is fragile to small changes in the setting.

Setting: multiple related tasks

- Training RL agents for each setting is computationally expensive and unreliable.



Expensive!



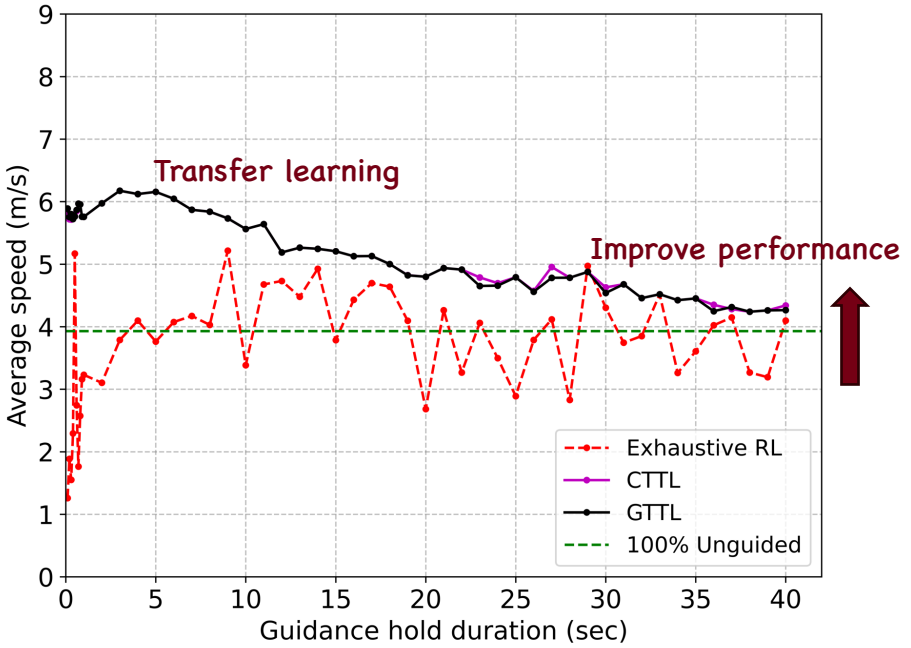
Poor performance!

Key idea

Training is expensive, but evaluation is cheap

Learning for Generalization

- Key observation: RL agents trained on some settings perform remarkably well on related settings.
- This is called **(Zero-shot) Transfer Learning!**
- To perform well across all settings, how can we **strategically** select good settings to train on?

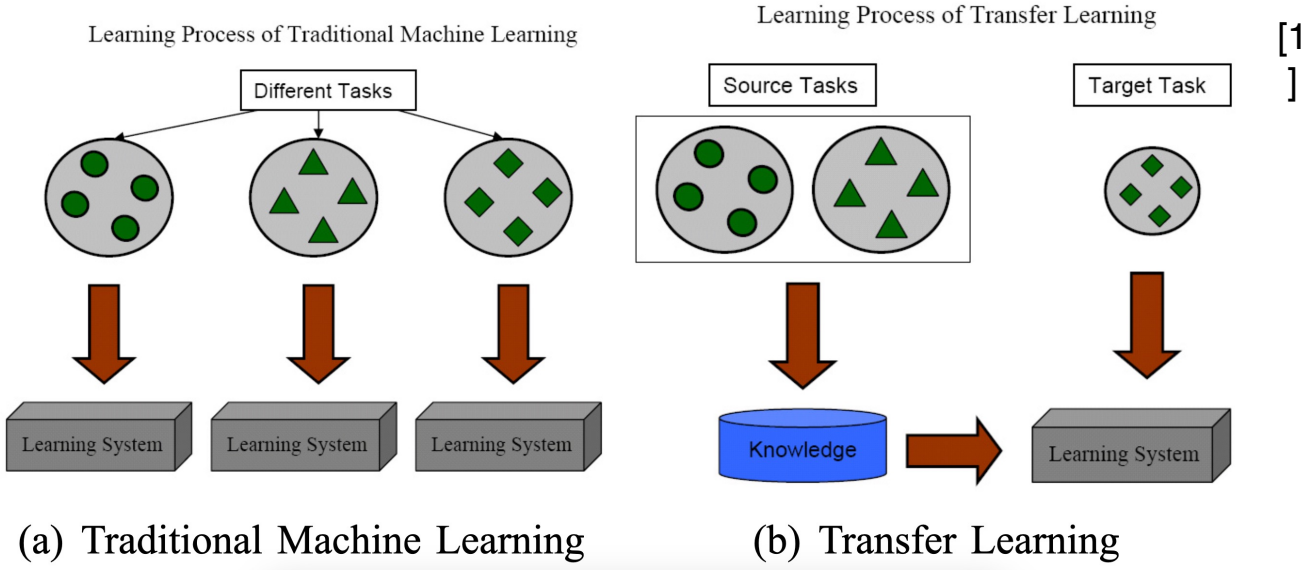


Transfer learning

Use experience from source task when solving a new task

Transfer Learning

- Use pre-trained knowledge from the **source task** to solve a new task (**target task**)



[1]

[1] S. J. Pan and Q. Yang, “A Survey on Transfer Learning,” IEEE Trans. Knowl. Data Eng., vol. 22, no. 10, pp. 1345–1359, Oct. 2010, doi: 10.1109/TKDE.2009.191.

Transfer learning

Zero-shot transfer – Easy-to-implement and best-performing transfer method

Terminology

- **"Shot"**: The number of attempts in the **target task**
- **Zero-shot**: Just run a policy trained in the **source task**
- **1-shot**: Try the task once
- **Few-shot**: Try the task a few times
- **Fine-tuning**: Train further in the target task

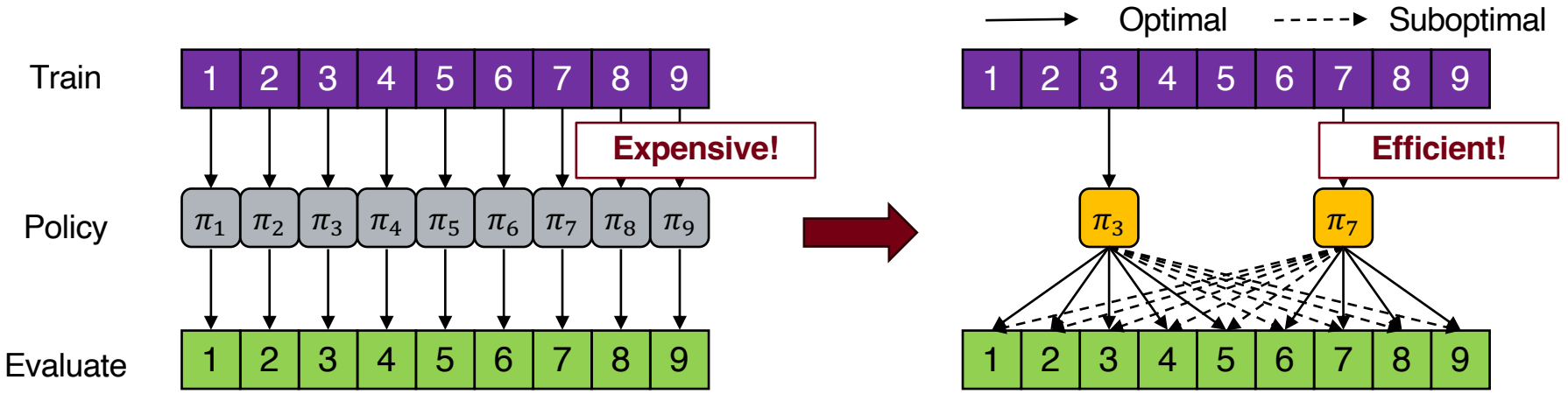
Our focus!

Transfer learning

Source tasks selection problem

Source tasks selection problem

- Transfer learning is more efficient than training exhaustively.
- Then, how do you determine which task to be the source and target task?



Exhaustive RL (Train each model)

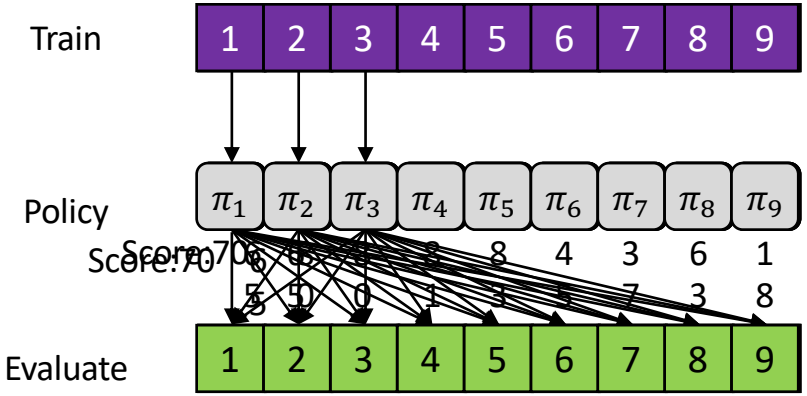
Transfer Learning

Transfer learning

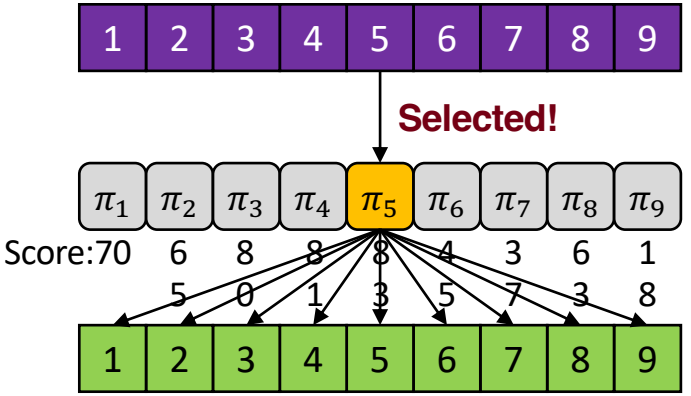
Sequential source tasks selection problem

Sequential source tasks selection problem

- Aim to sequentially choose the optimal source task that maximizes the estimated aggregate performance for multiple target tasks.



Estimation of expected performance



Source task selection

Possible strategies

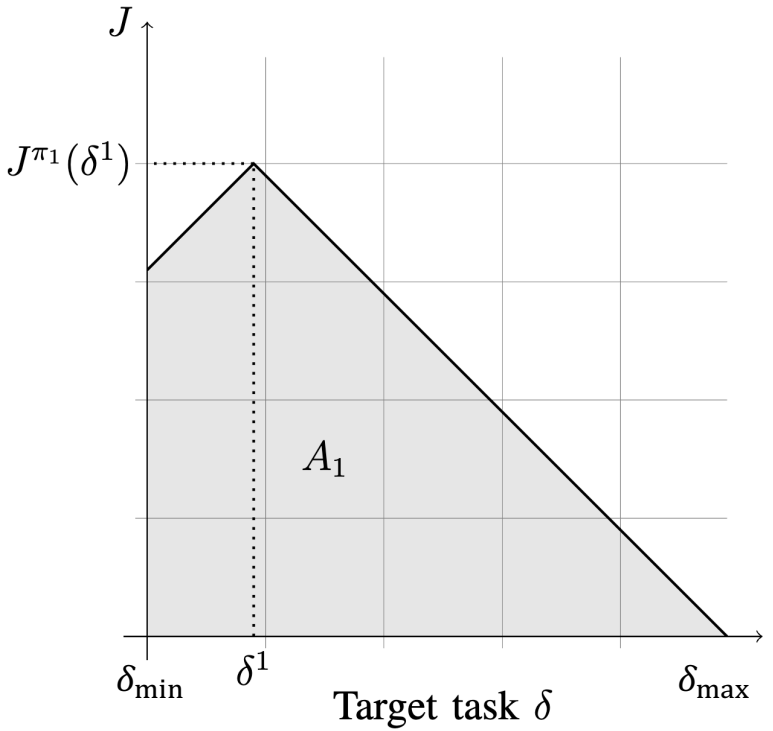
- Suppose you have a training budget of, e.g., $K = 5$ settings

- **Random strategy**: A basic strategy is to choose randomly!
  - Simple, worth trying

- What's the best strategy?
  - Can't know without making some assumptions

Problem Definitions

Guidance hold duration and performance

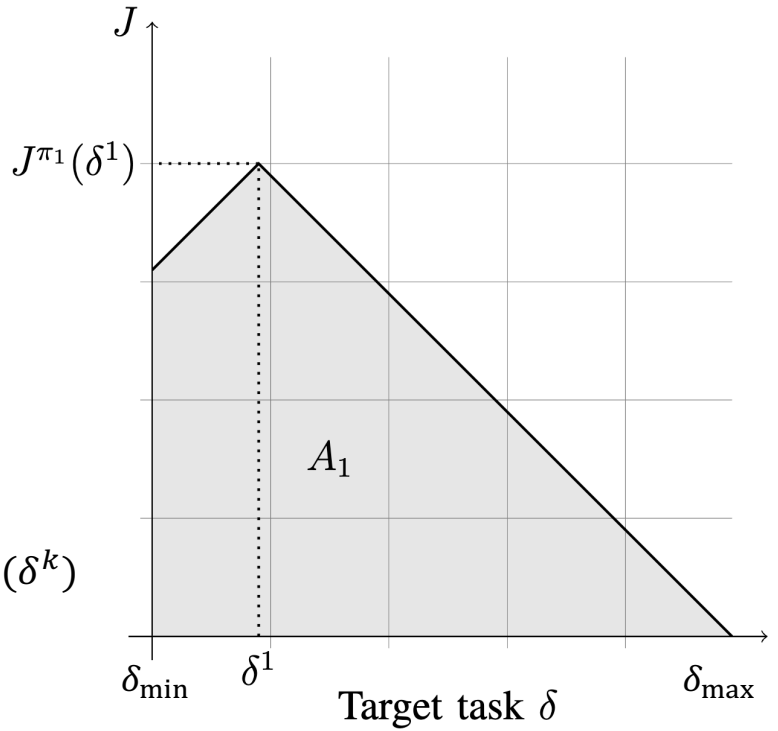


Problem Definitions

Guidance hold duration and performance

Guidance Hold Duration and Performance

- Guidance Hold Duration δ
- Multiple target tasks $[\delta_{\min}, \delta_{\max}]$
- Performance $J(\delta)$
- Aggregate performance $A(\delta_{\min}, \delta_{\max})$ or A
- Source task selected at kth transfer step: δ^k
- Transfer budget: K
- The policy trained at the task at δ^k: π_k
- The performance of policy π_k evaluated at δ^k: $J^{\pi_k}(\delta^k)$



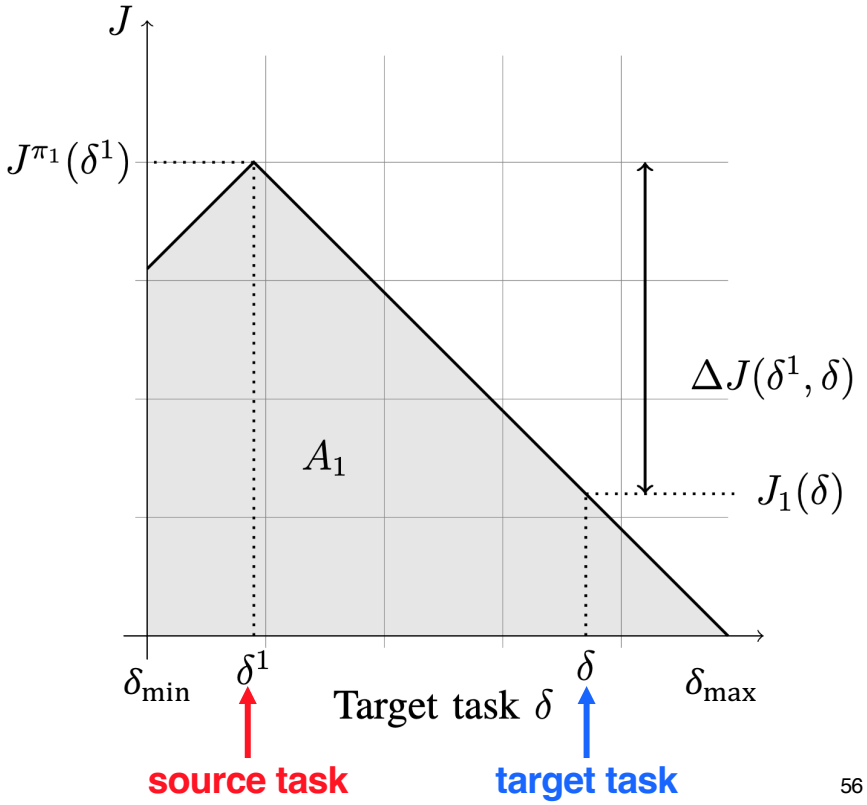
Problem Definitions

Generalization gap and Estimated performance

Generalization Gap

- The generalization gap quantifies the expected reduction in performance when a policy trained at source task is transferred to a target task

$$\Delta J(\delta_S, \delta_T)$$



Problem Definitions

Generalization gap and Estimated performance

Generalization Gap

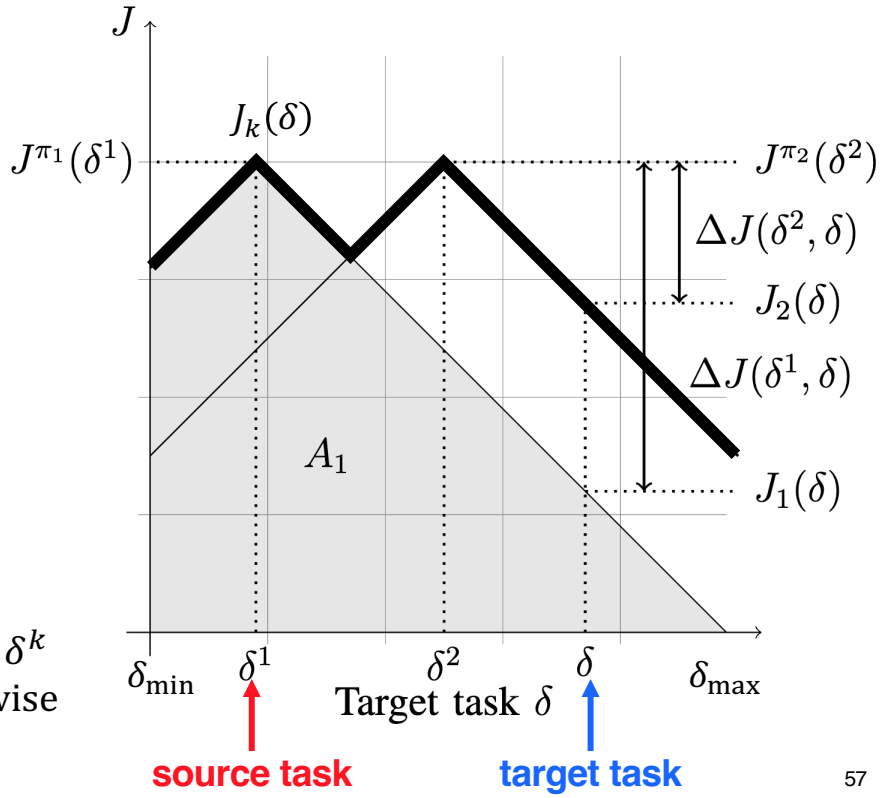
- The generalization gap quantifies the expected reduction in performance when a policy trained at source task is transferred to a target task

$$\Delta J(\delta_S, \delta_T)$$

Updating estimated performance

- At each iteration k, the estimated performance $J_k(\delta)$ is updated with the best performing policies

$$J_k(\delta) = \begin{cases} J^{\pi_k}(\delta^k) & \text{if } \delta = \delta^k \\ \max(J_{k-1}(\delta), J^{\pi_k}(\delta^k) - \Delta J(\delta^k, \delta)) & \text{otherwise} \end{cases}$$



Problem Definitions

Sequential source tasks selection problem

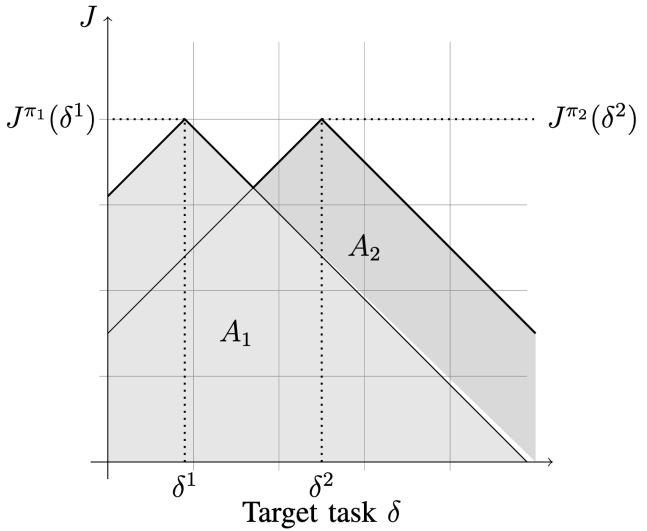
Sequential source tasks selection problem

- Aim to sequentially choose the optimal guidance hold duration δ^k that maximizes the estimated performance for multiple target tasks

- For the initial step ($k = 1$)
  - δ^1 is selected randomly from the specified range of tasks
  - (or select the task that maximizes the A_1)
- For each subsequent steps ($k = 2, \ldots, K$)
  - δ^k is chosen to maximize the aggregated estimated performance A_k

$$\delta^k = \arg\max_{\delta} A_k$$

  - where $A_k = \int_{\delta_{\min}}^{\delta_{\max}} J_k(\delta) \mathrm{d}\delta$.



Modeling Assumptions

All tasks have same upper bound performance and deterministic training.

Assumption 1: Constant upper bound performance

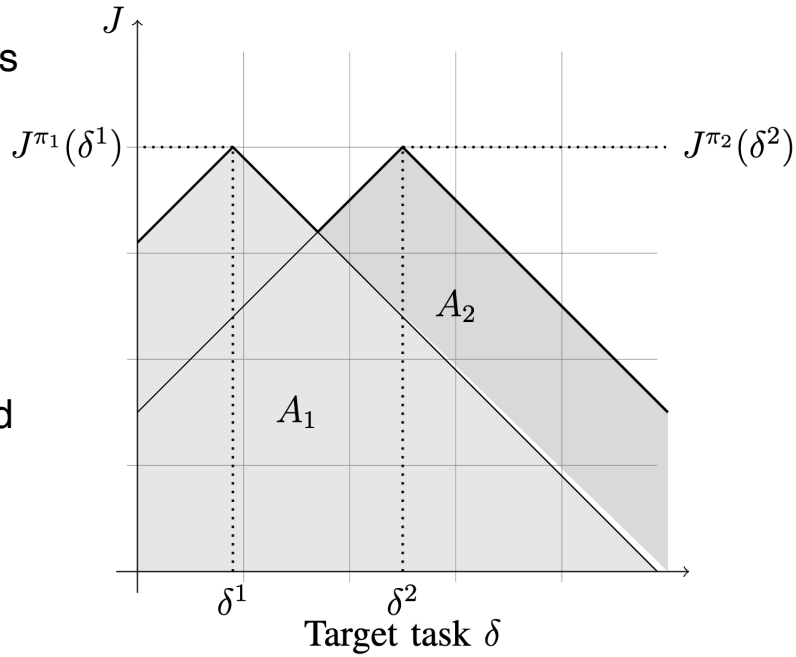
- The upper bound performance remains constant across all guidance hold duration tasks

$$J^{\pi_k}(\delta) = J^* \qquad \forall \delta \in [\delta_{\min}, \delta_{\max}]$$

Assumption 2: Deterministic training performance

- For any task trained, it always achieve the upper bound performance

$$J^{\pi_k}(\delta^k) = J^*(\delta^k)$$



Modeling Assumptions

Linear and identical generalization gap

Assumption 3: Linear generalization gap

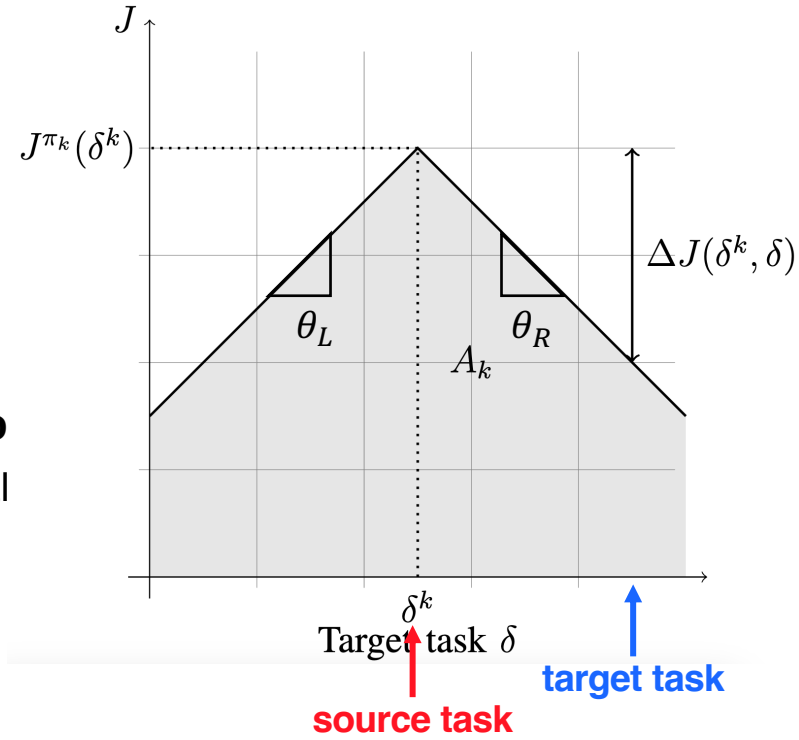
- The generalization gap $\Delta J(\delta_S, \delta_T)$ between source task and target task is linearly related to the absolute difference between δ_S and δ_T.

$$\Delta J(\delta_S, \delta_T) = \begin{cases} \theta_L(\delta_S - \delta_T), & \text{if } \delta_S \geq \delta_T \\ \theta_R(\delta_T - \delta_S), & \text{otherwise} \end{cases}$$

Assumption 4: Identical slope of generalization gap

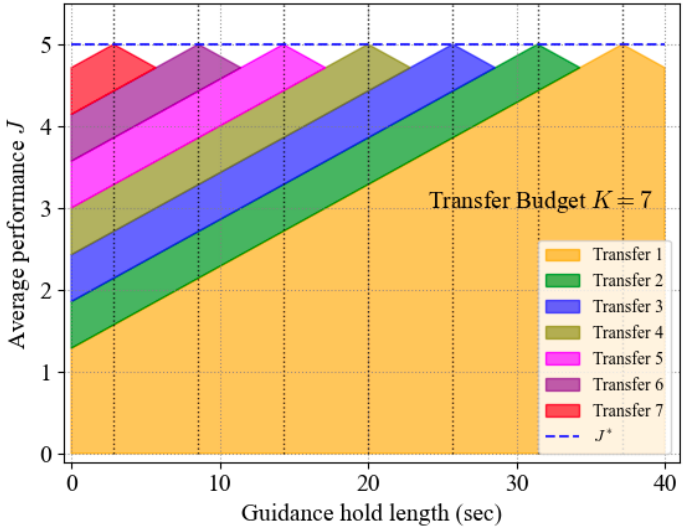
- The slope of the generalization gap function is identical for whether transferring either direction

$$\theta_L = \theta_R$$



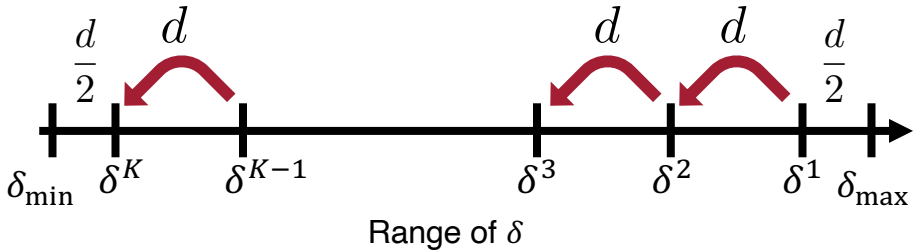
Possible strategies

- Suppose you have a training budget of, e.g., $K = 5$ settings
- **Equidistant strategy**: Choose evenly spaced settings



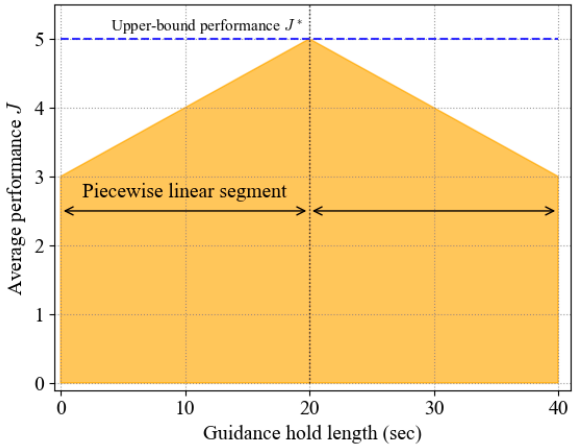
$$\delta^1 = \delta_{\max} - \frac{\delta_{\max} - \delta_{\min}}{2K}$$

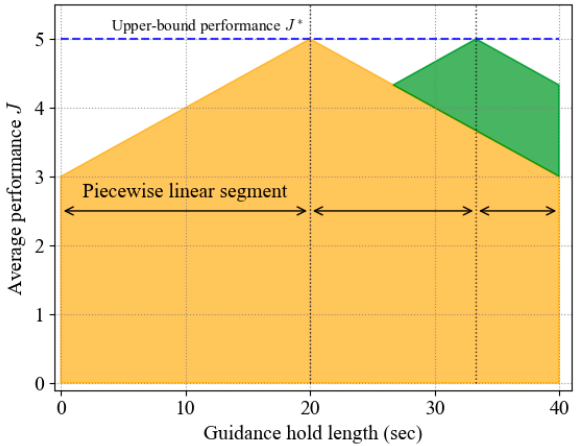
$$d = \frac{\delta_{\max} - \delta_{\min}}{K}$$

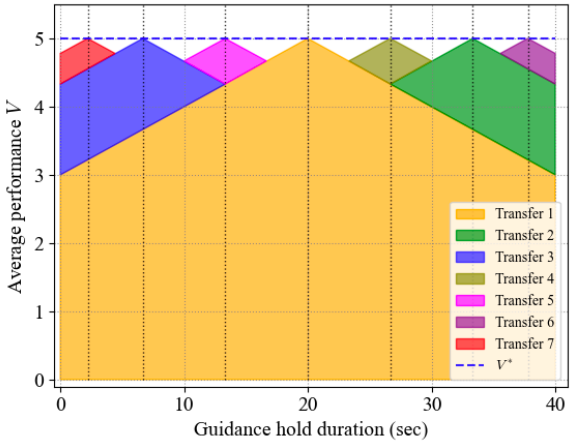


Advanced strategies

- Suppose you don’t know your training budget (”just don’t blow the whole lab budget”)
- **Bisection strategy**: Choose the next setting that would increase the generalization performance area





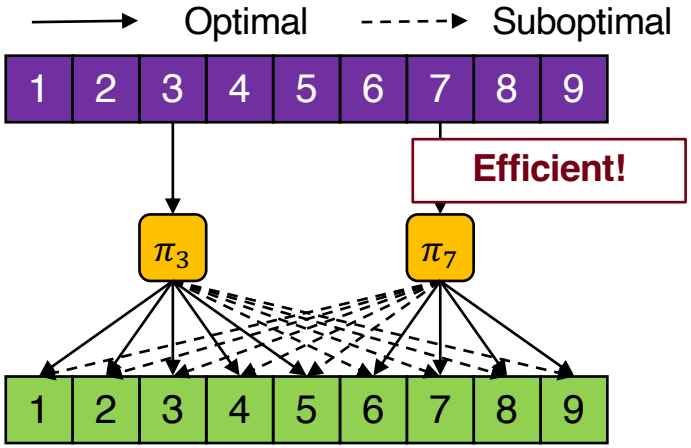


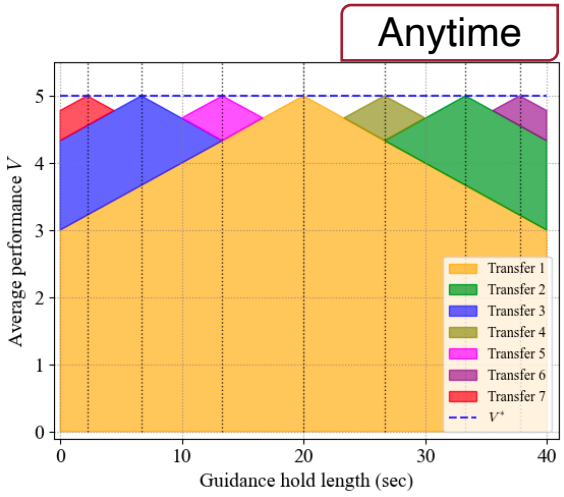
Temporal Transfer Learning Algorithms

Comparison of equidistant and bisection strategies

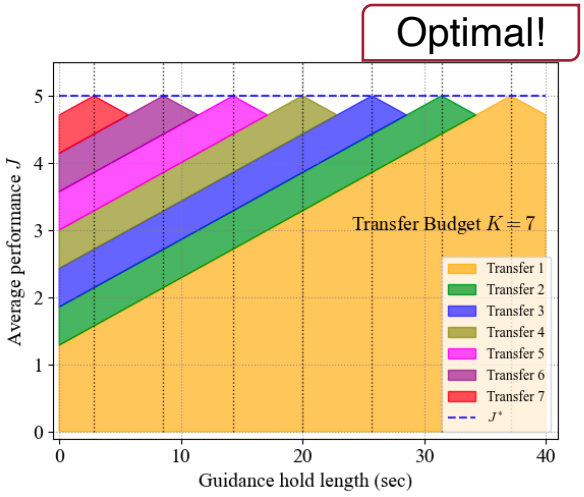
Summary

- **Equidistant strategy** is optimal (given a fixed budget)
- **Bisection strategy** is gives valid solutions for any budget





Bisection strategy



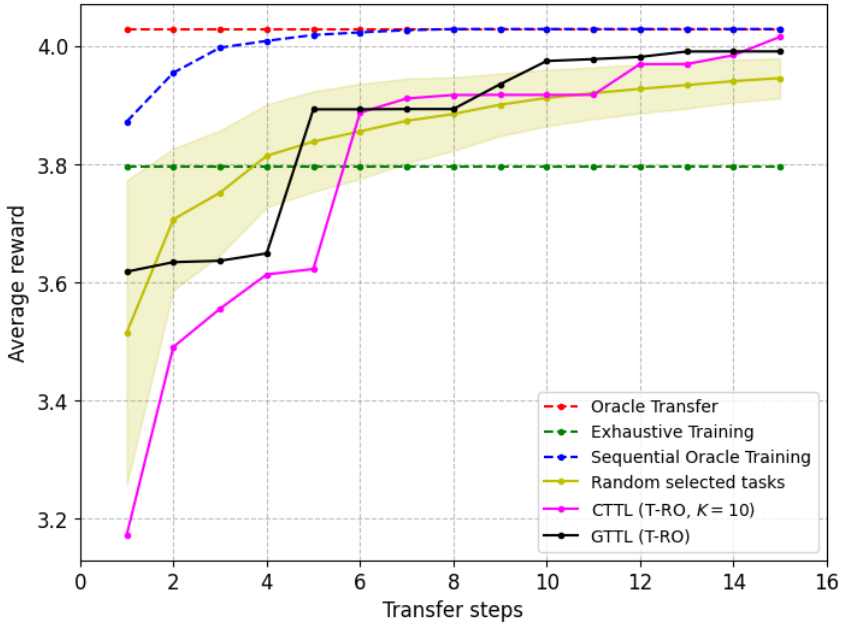
Equidistant strategy

Results

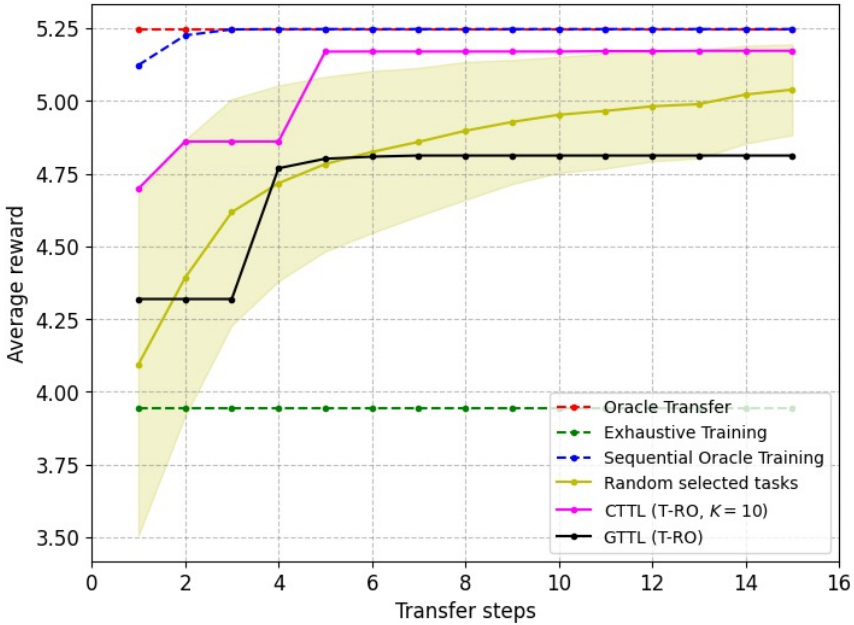
System performance of transfer learning algorithms

Demo
(Google Colab)

Ring with acceleration guidance



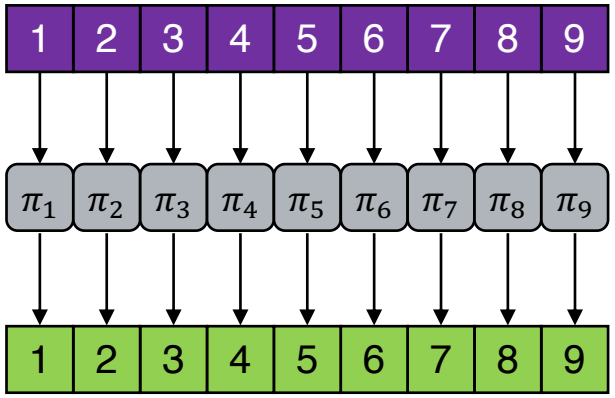
Highway ramp with acceleration guidance



Summary

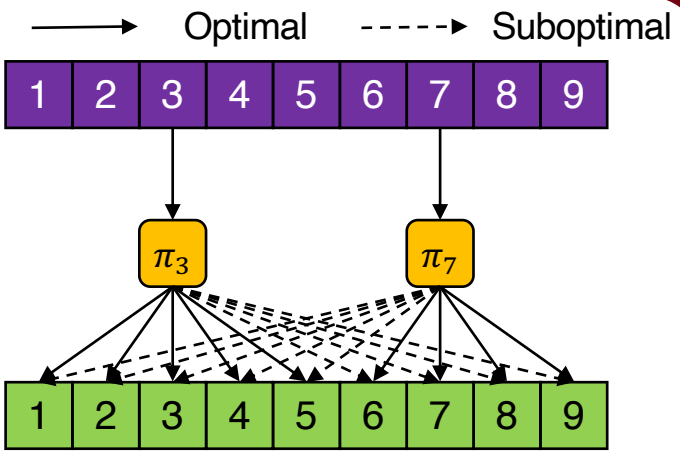
Experimental results of TTL

If you were to solve multiple tasks, what would you choose?



Exhaustive RL

- Large sample complexity
- Uncertain training performance
- Ignore task-relatedness



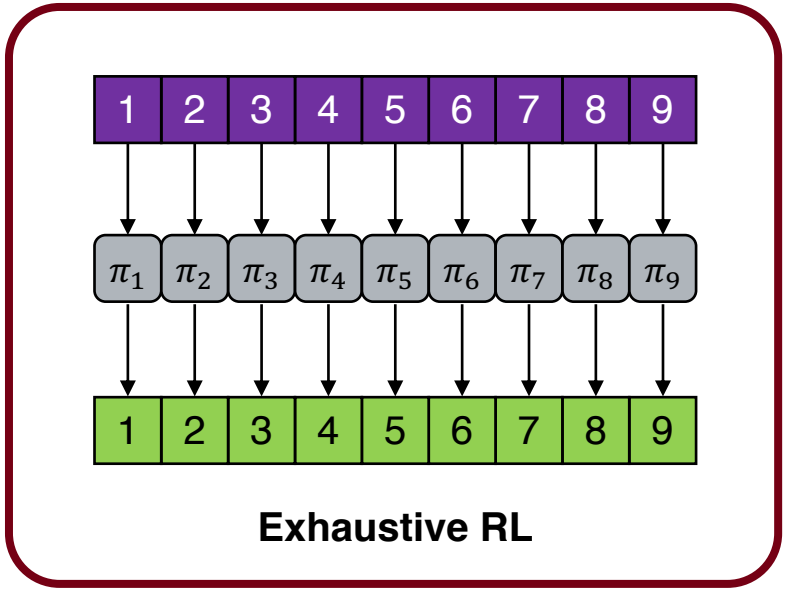
Transfer Learning

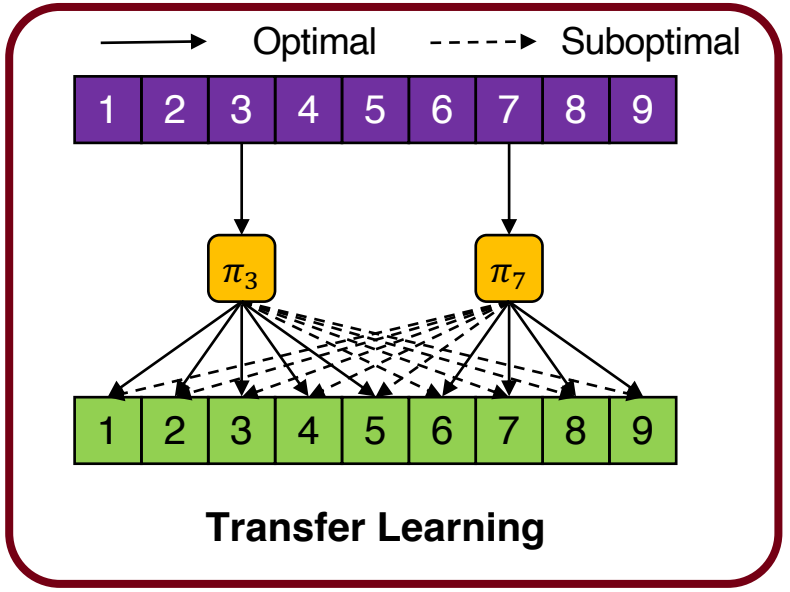
- Efficient computation
- Improved average-case performance
- Consider task-relatedness

Summary

Experimental results of transfer learning algorithms

If you were to solve multiple tasks, what would you choose?





You only need a few strategically selected settings to train + zero-shot transfer on the rest!

References

References:

1. Readings: Chapters 8-9. Grokking deep reinforcement learning.
2. Deep Learning by “Goodfellow, Bengio, Courville”
3. http://neuralnetworksanddeeplearning.com/index.html

With slides adapted from:

1. Jung Hoon Cho, Alessandro Lazaric and Matteo Pirotta
2. Eugene Vinitsky (Berkeley EE290O)
3. Professor Zico Kolter (CMU 15-780)
4. John Schulman (Berkeley CS294-112)