imitation learning

Hal Daumé III | U Maryland / Microsoft Research | me@hal3.name | @haldaume3

setup

1. Learning from reinforcement alone is hard
    1. Exploration is hard
    2. Credit assignment is hard
    3. Designing rewards is hard

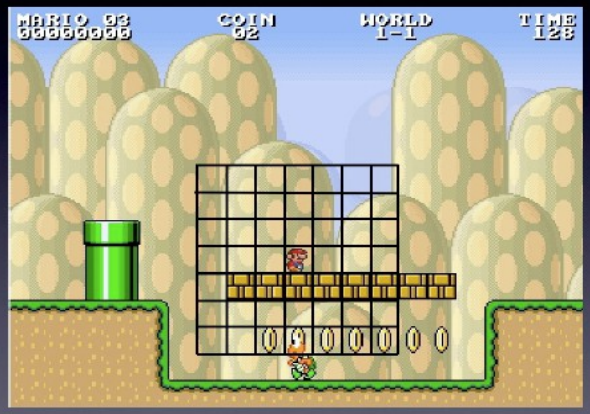
2. Yet people are pretty good at many tasks
3. Perhaps we can use them to help

An example from playing Mario

From Mario AI competition 2009

Input:

Interface



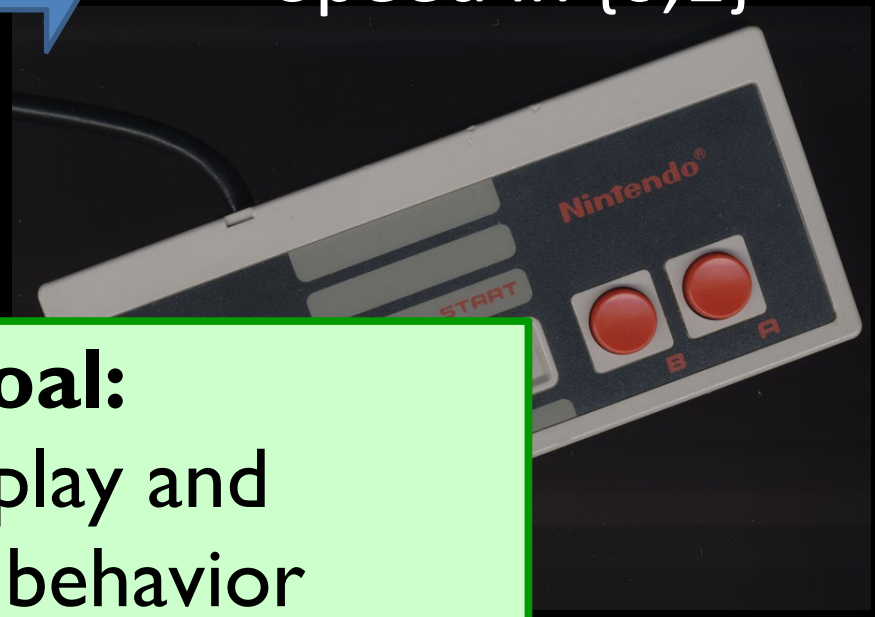
Output:

Jump in {0,1}

Right in {0,1}

Left in {0,1}

Speed in {0,1}



High level goal:

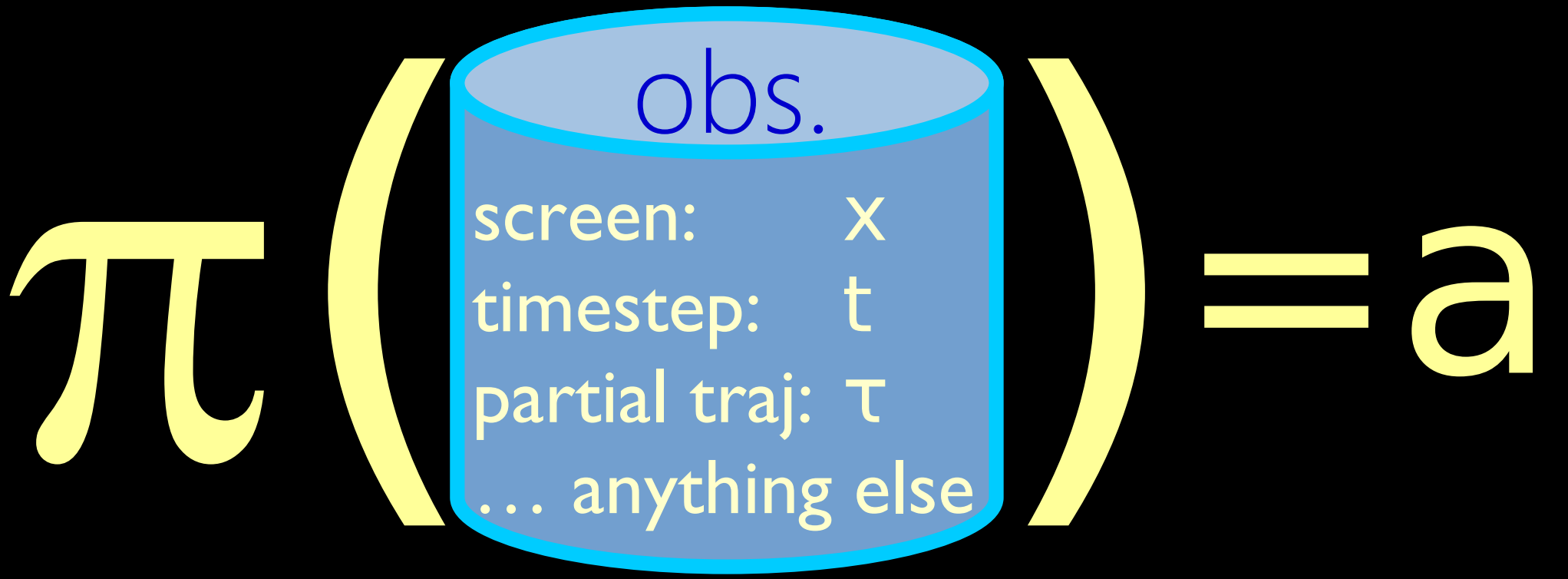
Watch an expert play and learn to mimic her behavior

Training (expert)

Video credit: Stéphane Ross, Geoff Gordon and Drew Bagnell

Policies

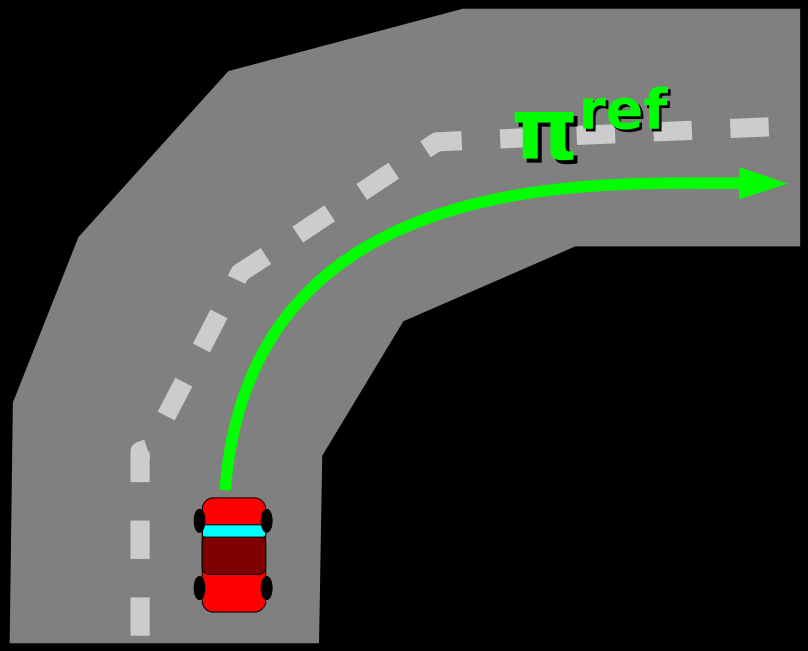
- A policy maps observations to actions



Warm-up: Supervised learning

1. Collect trajectories from expert π^{ref}
2. Store dataset **D** = { (o, π^{ref}(o)) | o ~ π^{ref} }
3. Train classifier π on **D**

- **Let π play the game!**



sometimes called “behavioral cloning”

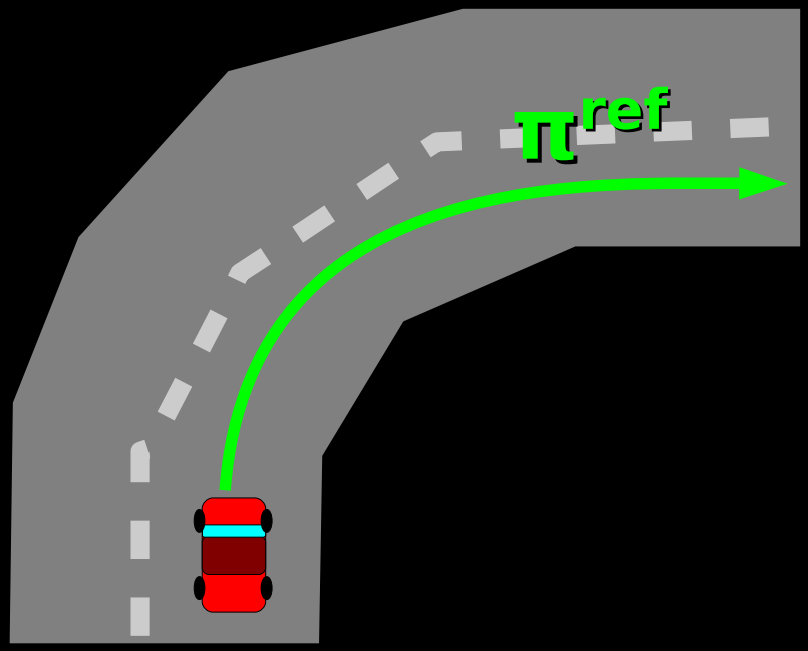
Test-time execution (sup. learning)

Video credit: Stéphane Ross, Geoff Gordon and Drew Bagnell

What's the (biggest) failure mode?

The expert never gets stuck next to pipes

$\Rightarrow$ Classifier doesn't learn to recover!



Outline

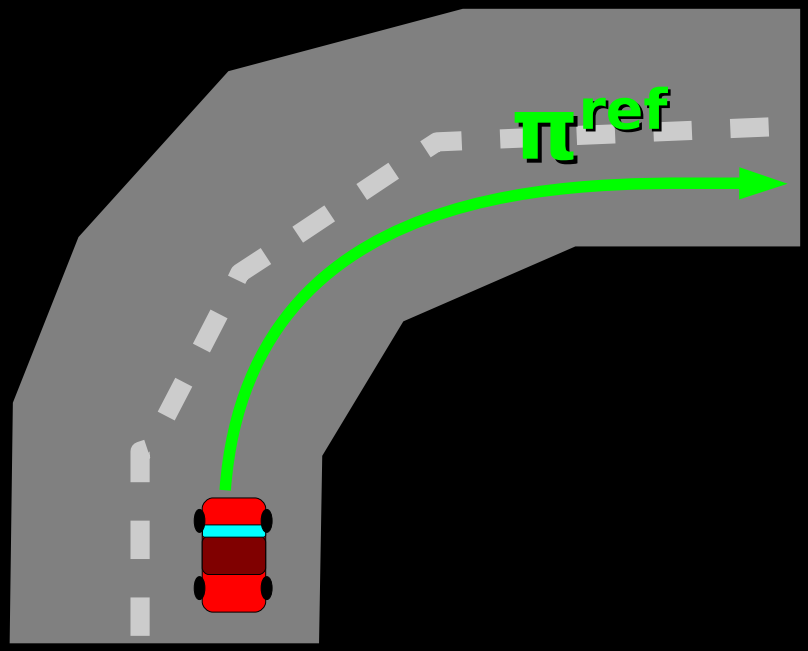
- From demonstrations → expert decisions
- From expert decisions → expert costs
- Whence the expert?
- Combining experts and reward

What's the (biggest) failure mode?

The expert never gets stuck next to pipes

$\Rightarrow$ Classifier doesn't learn to recover!

- We'd like to train the policy on all states
- Can't do that
- Let's train it *where it visits*



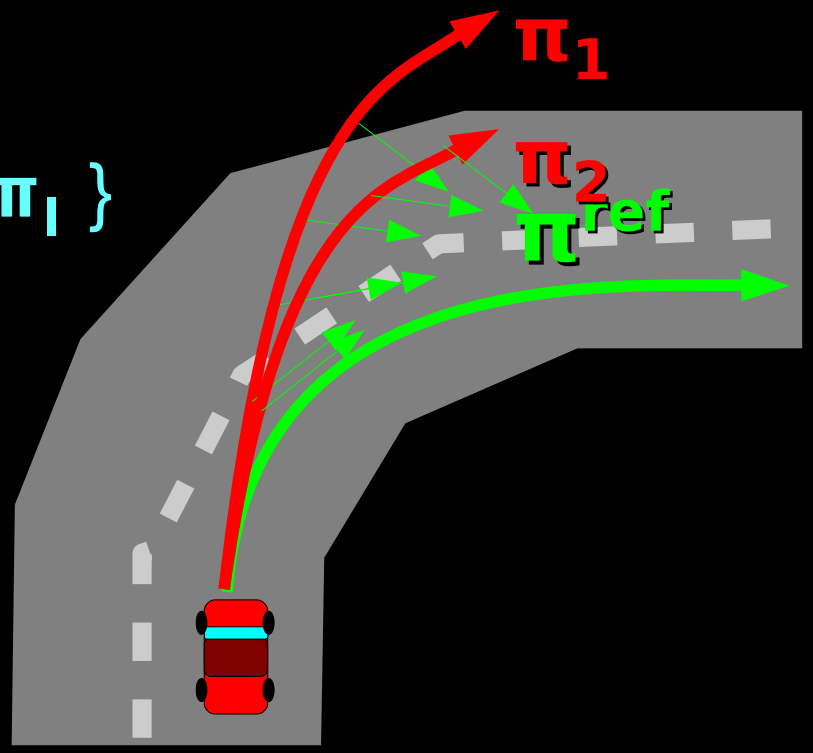
Learning from an expert: DAgger

1. Collect trajectories from expert π^{ref}
2. Dataset $D_0 = \{ (o, \pi^{ref}(o,y)) \mid o \sim \pi^{ref} \}$
3. Train π_1 on D_0
4. Collect new trajectories from π_1
   - But let the *expert* steer!
5. Dataset $D_1 = \{ (o, \pi^{ref}(o,y)) \mid o \sim \pi_1 \}$
6. Train π_2 on $D_0 \cup D_1$

If $N = T \log T$,

$L(\pi_n) < T \epsilon_N + O(1)$

for some n

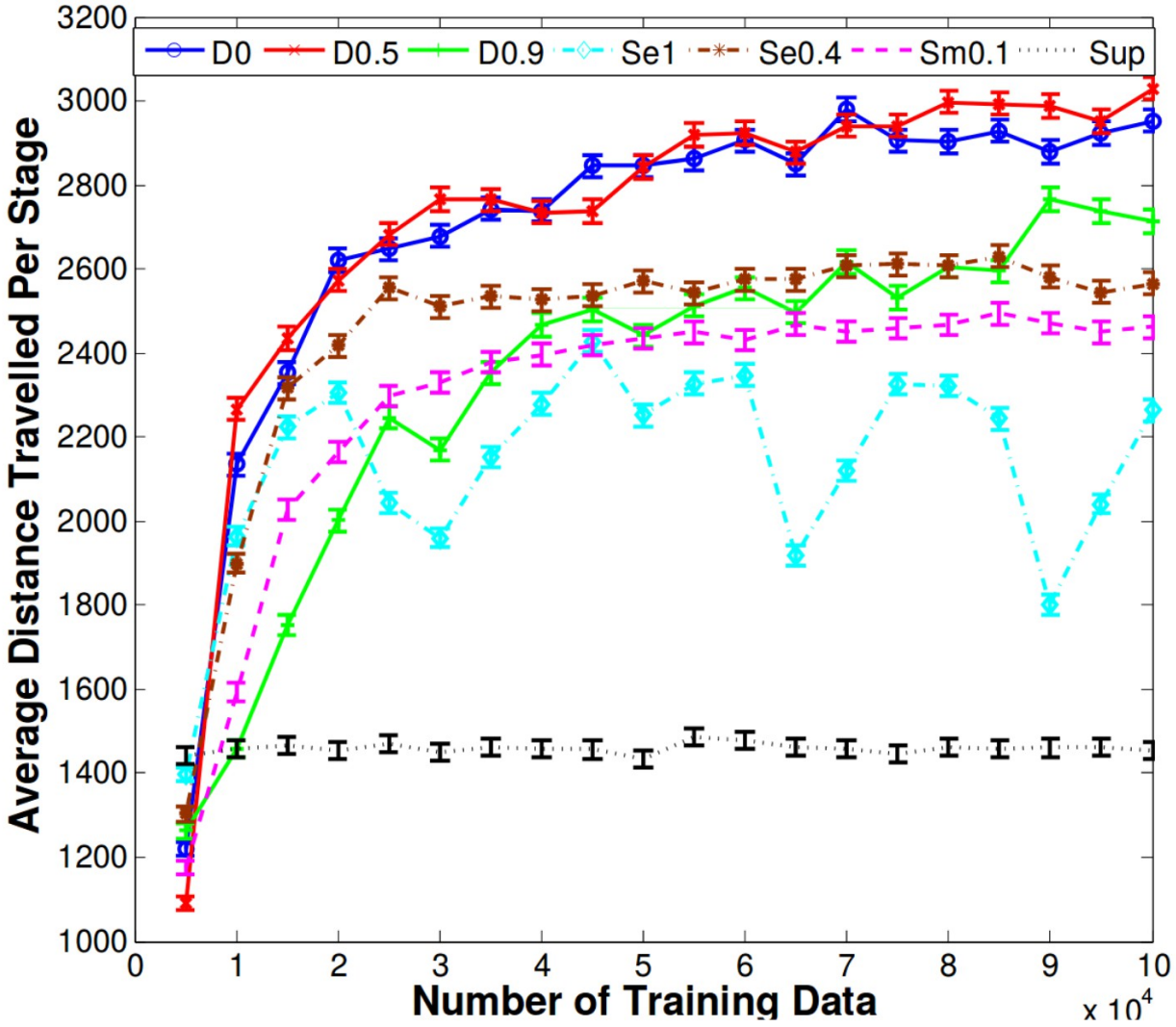


- In general:
  - $D_n = \{ (o, \pi^{ref}(o,y)) \mid o \sim \pi_n \}$

Test-time execution (DAgger)

Video credit: Stéphane Ross, Geoff Gordon and Drew Bagnell

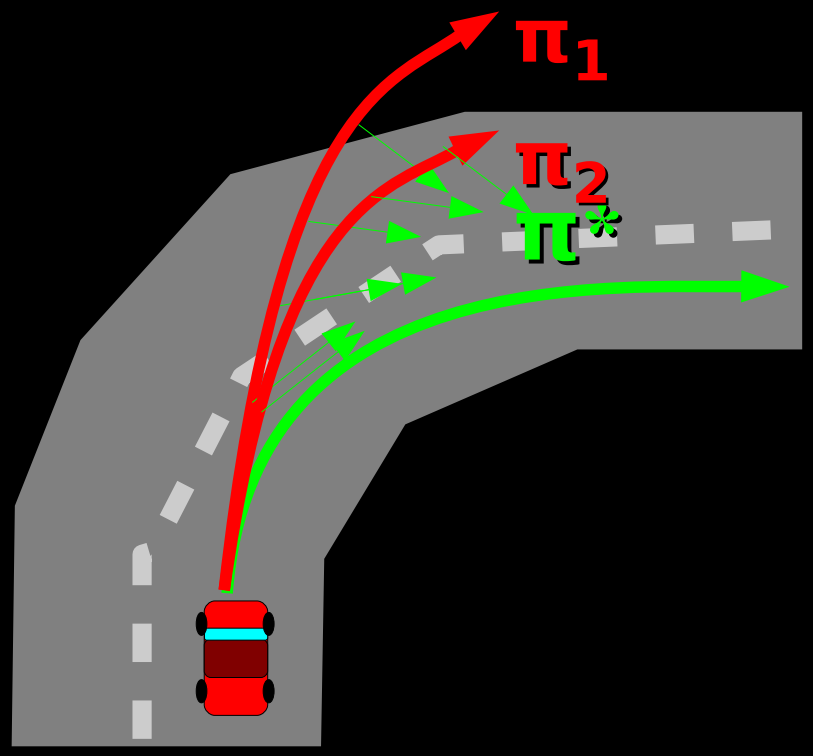
how well does this strategy work?



Ross+Bagnell, AIStats'10

What's the biggest failure mode?

Classifier only sees *right* versus *not-right*



Aside: cost-sensitive classification

Classifier: $h : x \to [K]$

Multiclass classification

- $(x,y) \in X \times [K]$
- $\min_h \Pr(h(x) \neq y)$

Cost-sensitive classification

- $(x,c) \in X \times [0,\infty)^K$
- $\min_h E_{(x,c)} [c_{h(x)}]$

Easy solution to cost-sensitive cl

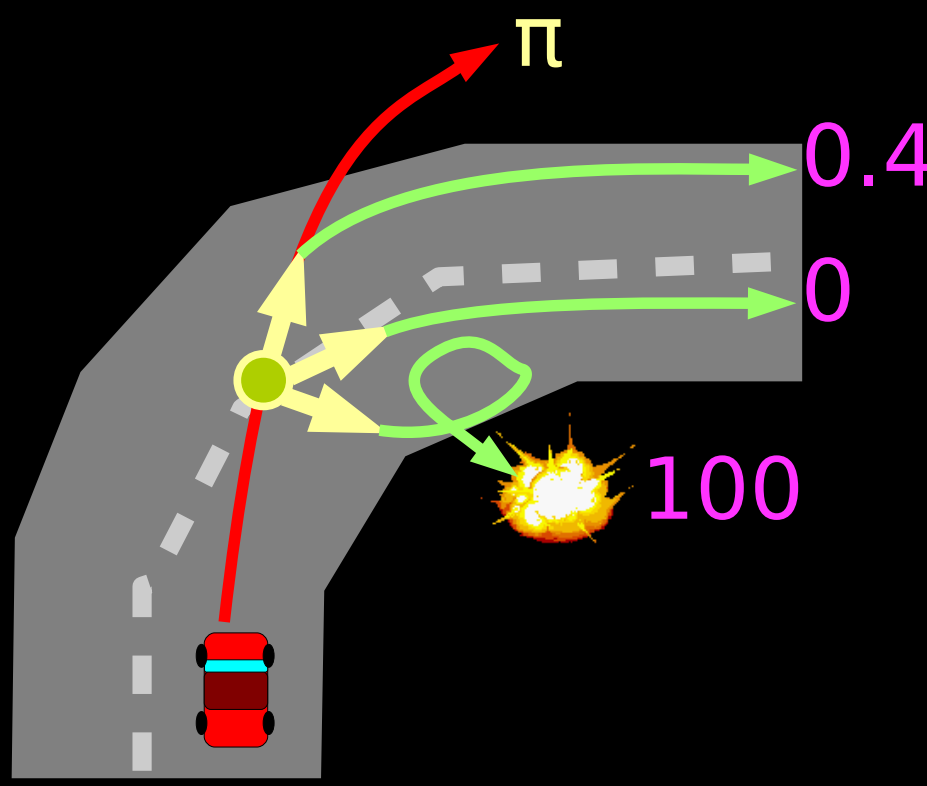
Classifier: $h : x \rightarrow [K]$

- $(x,c) \in X \times [0,\infty)^K$
- $\min_h E_{(x,c)} [\, c_{h(x)} \,]$

Solution learn a K-dimensional regressor on costs; pick minimal cost

Learning to search: AggraVaTe

1. Let learned policy π drive for t timesteps to obs. o
2. For each possible action a:
   - Take action a, and let expert π^{ref} drive the rest
   - Record the overall loss, c_a
3. Update π based on example:
   $(o, \langle c_1, c_2, \ldots, c_K \rangle)$
4. Goto (1)



Outline

- From demonstrations → expert decisions
- From expert decisions → expert costs
- Whence the expert?
- Combining experts and reward

what is the expert's API?

- **Behavioral cloning**
  - **Input:** World
  - **Output:** Set of trajectories

- **Dagger**
  - **Input:** observation
  - **Output:** optimal(ish) action

- **Aggrevate:**
  - **Input:** observation
  - **Output:** long term costs for all actions

where does an expert come from?

- Option 1: An actual real life human being
- Option 2: Simulation

reference policy

Given partial traj. $a_1, a_2, \ldots, a_{t-1}$

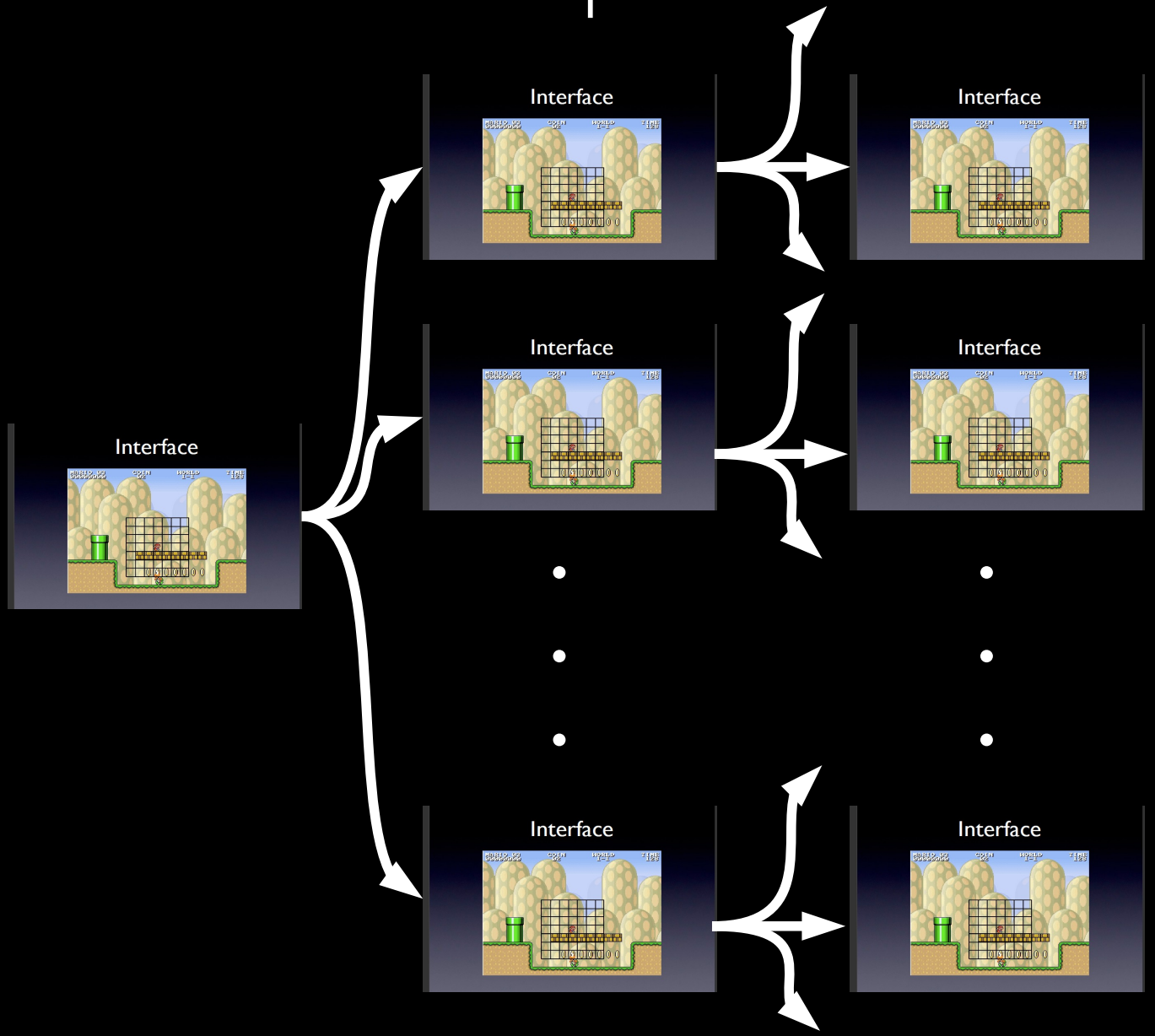
The *(expected) minimum achievable loss* is:

$$\min_{(a_t, a_{t+1}, \ldots)} \mathbb{E}\ \text{loss}(\vec{a})$$

The *optimal action* (DAgger) is corresponding a_t

The *optimal Q values* (Aggrevate) are all the mins for each a_t

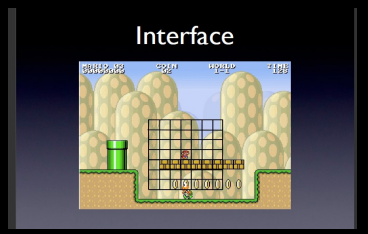
simulated experts



Rollout to some depth

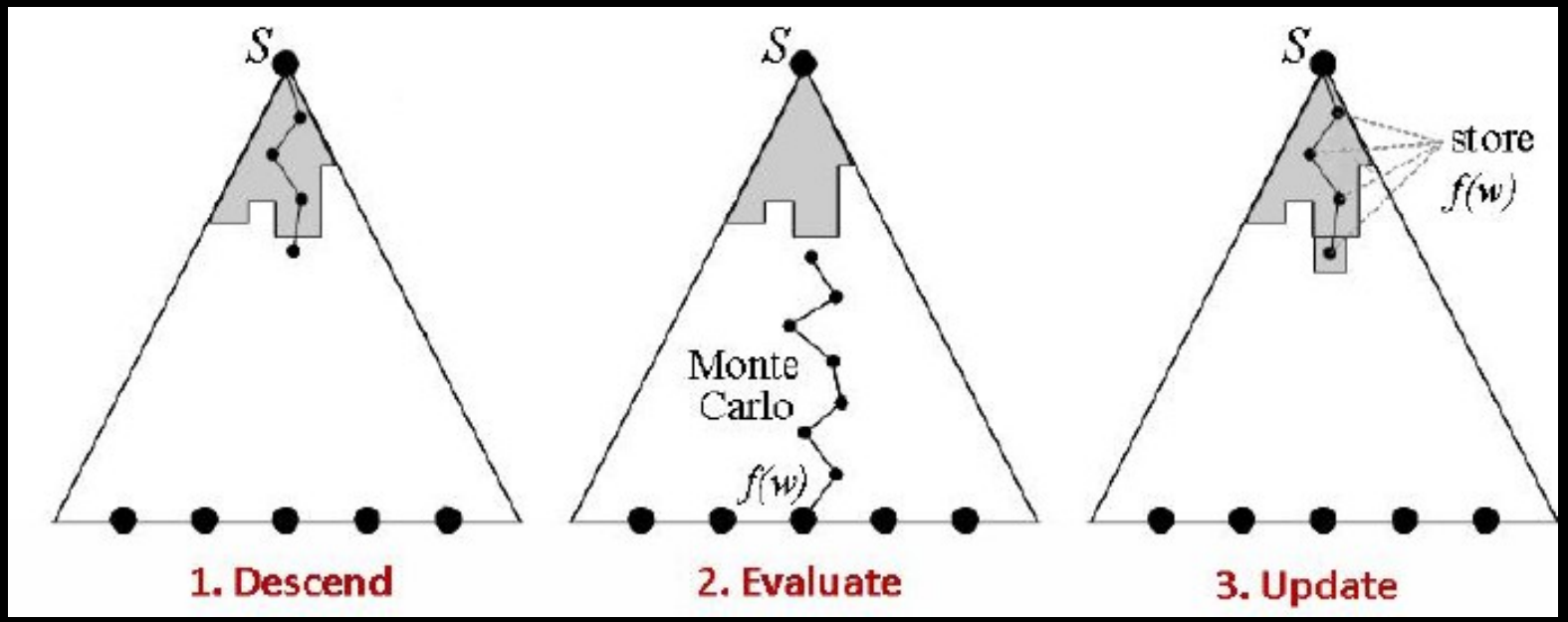
→ Intermediate loss

→ Return best start



$x2^{16}$

beyond naive search: any planner



e.g., Monte Carlo
Tree Search

Image credit: Michele Sebag and DeepMind

also effective for structured prediction

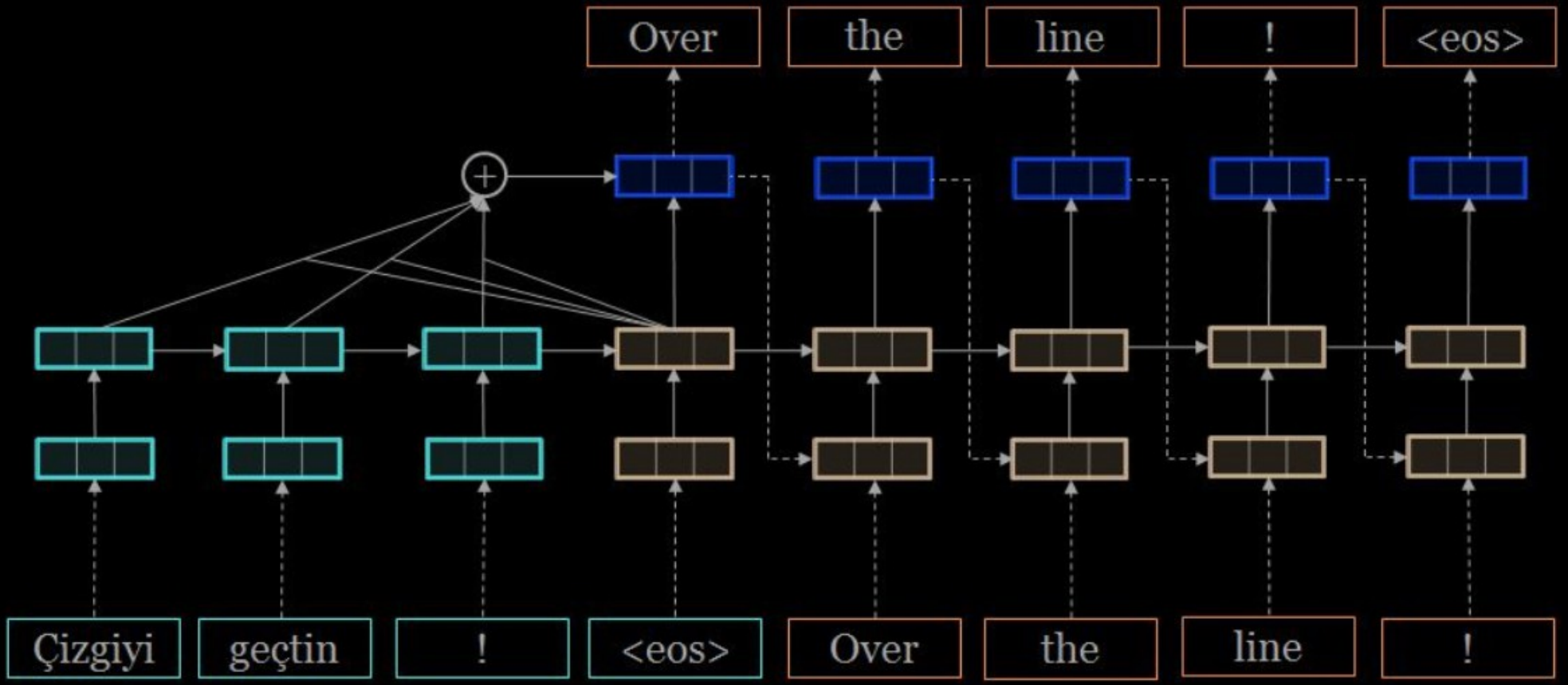
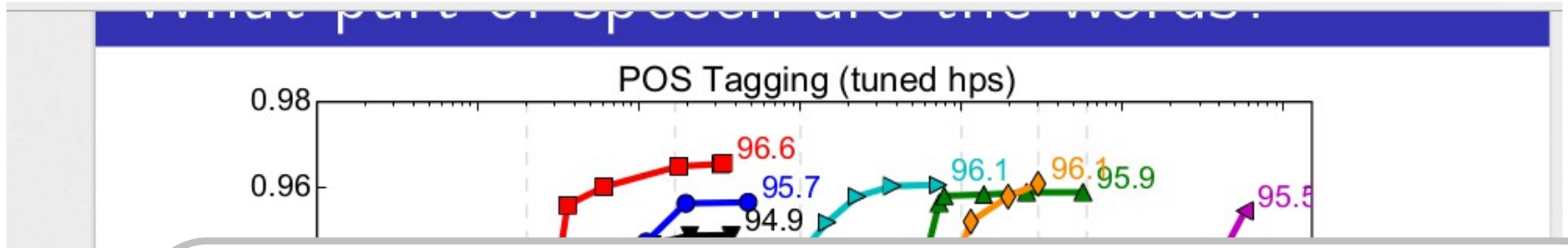
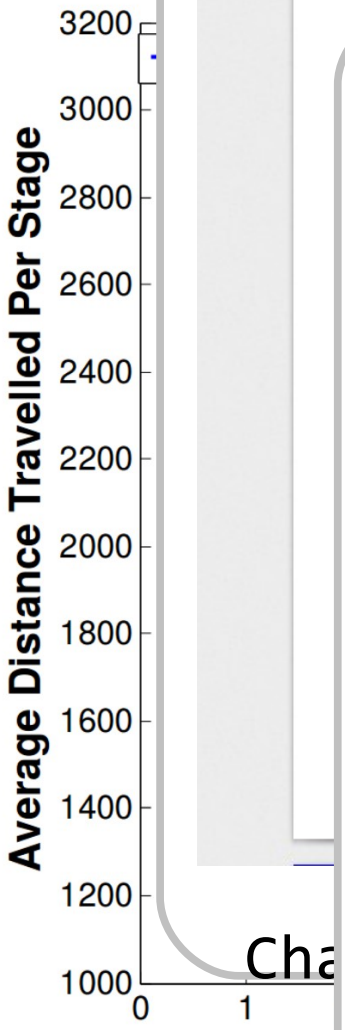
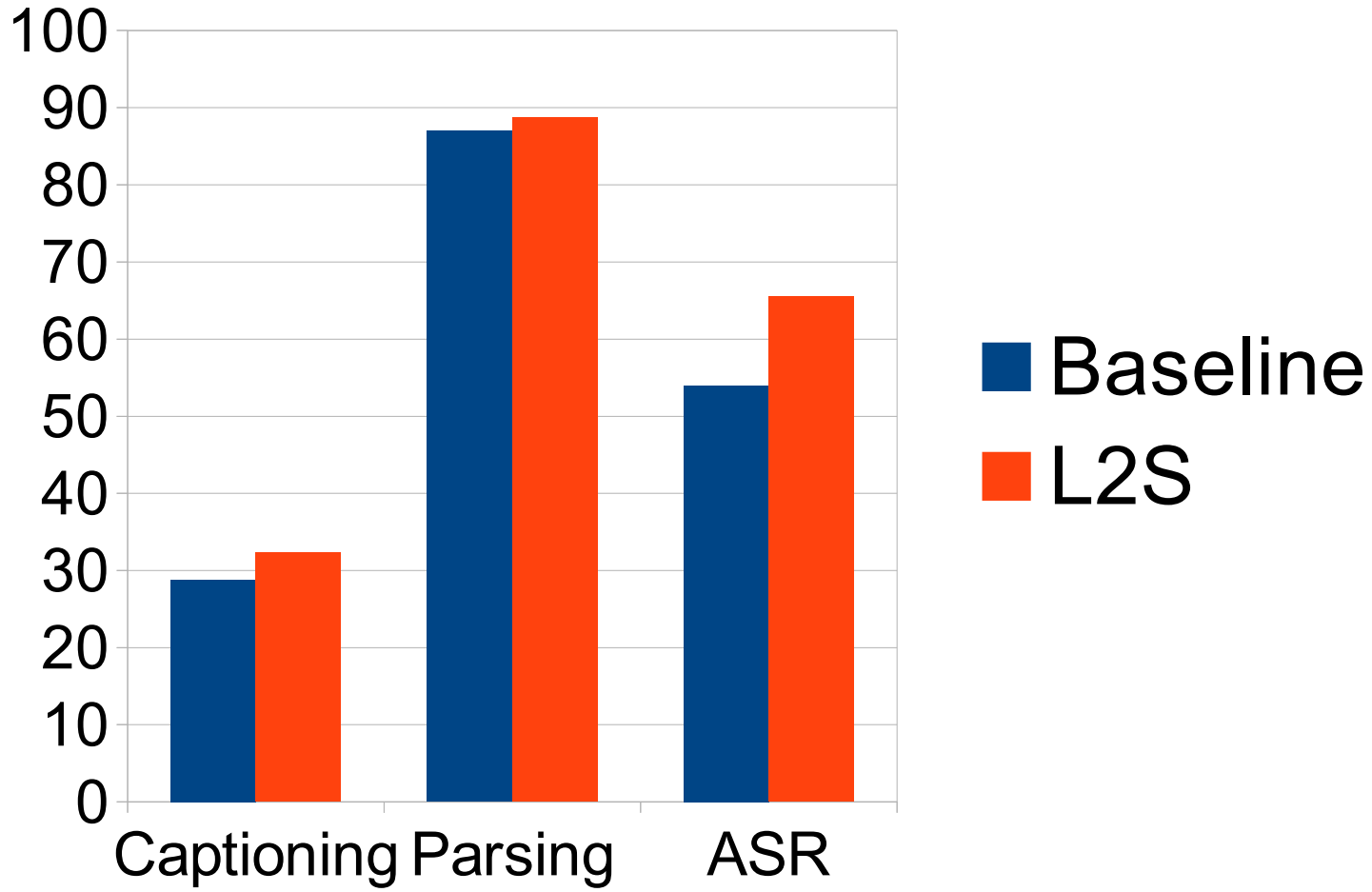


Image credit: Klein et al., 2017

how







Ross+Bagne

Bengio+Vinyals+Navdeep+Shazeer, NIPS'16

Outline

- From demonstrations → expert decisions
- From expert decisions → expert costs
- Whence the expert?
- Combining experts and reward

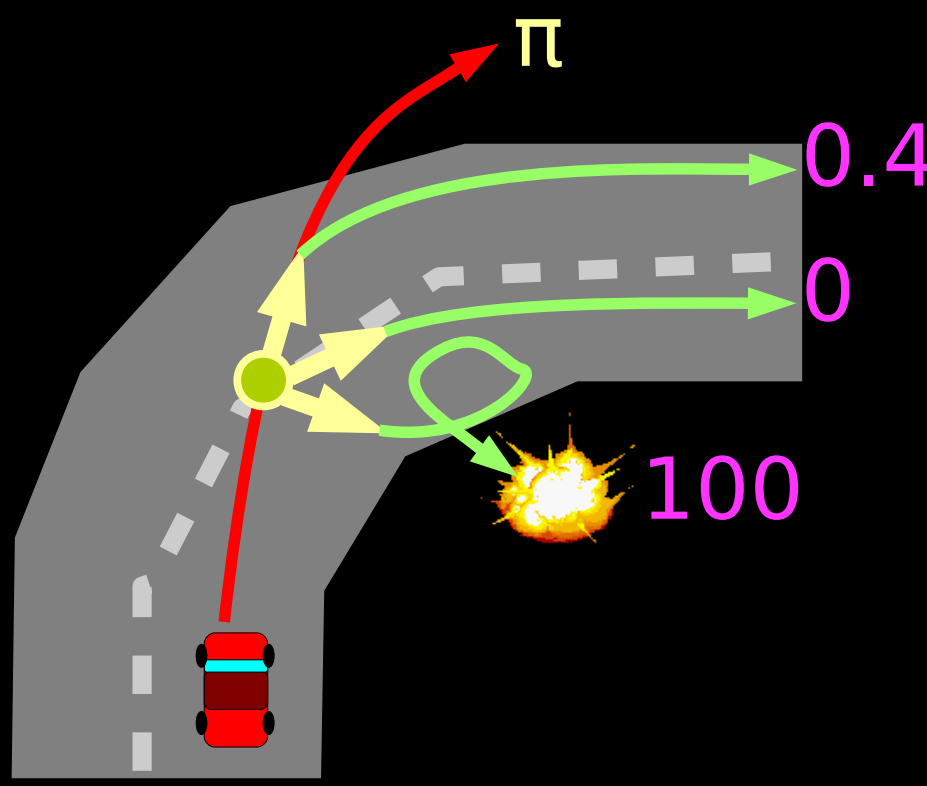
combining experts & rewards

- General solution... joint loss:
  optimize E[reward] + z E[imitation loss]

- Big question: how to set z?

LOLS: change rollout strategy

1. Let learned policy π^{in} drive for t timesteps to obs. o
2. For each possible action a:
   - Take action a, and let something π^{out} drive the rest
   - Record the overall loss, c_a
3. Update π based on example:
   $(o, \langle c_1, c_2, \ldots, c_K \rangle)$
4. Goto (1)



effect of roll-in & roll-out strategies

roll-out → / ↓ roll-in	**Reference**	**Mixture**	**Learned**
Reference	Inconsistent		
Learned	Not locally opt.	Good	RL

guarantees

roll-out → / ↓ roll-in	**Reference**	**Mixture**	**Learned**
Reference	Inconsistent		
Learned	Not locally opt.	Good	RL

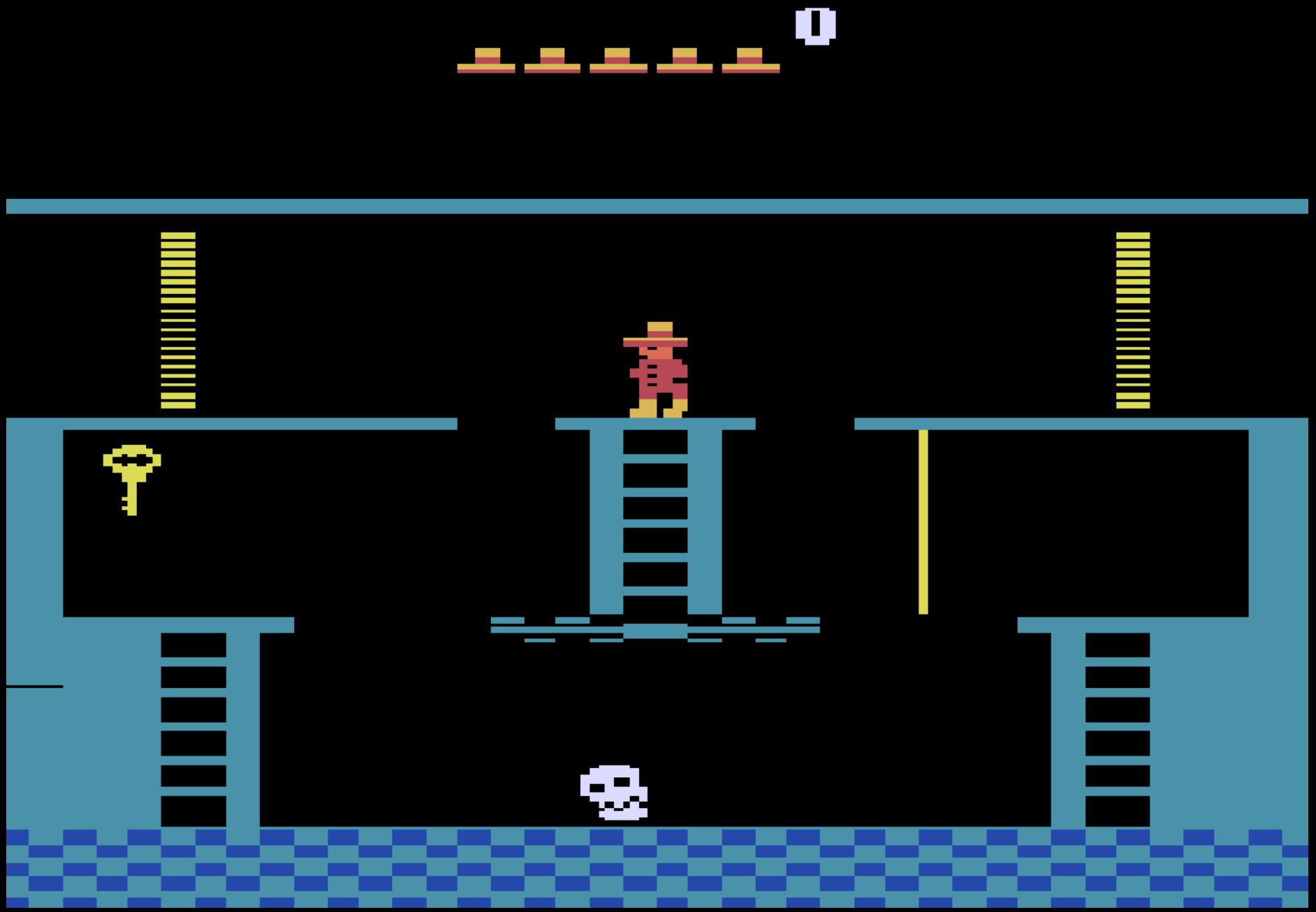
In "Good" setting, can prove that:

-
-
  -
  -

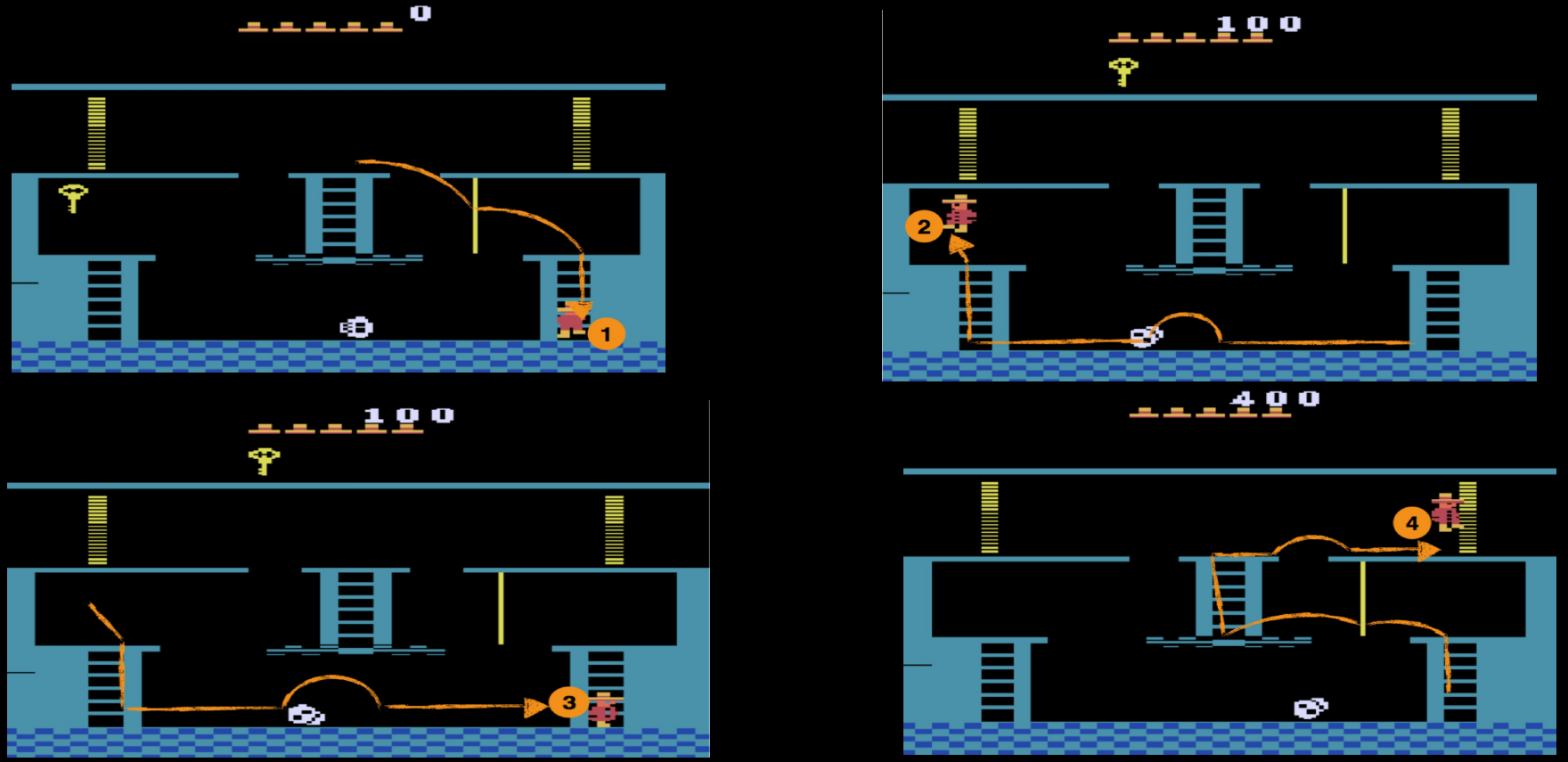
Furthermore, whp:

$$Regret = O\left((KT)^{2/3}\sqrt[3]{\frac{\log(N|\Pi|)}{N}} + T\delta_{class}\right)$$

combination via hierarchies



combination via hierarchies

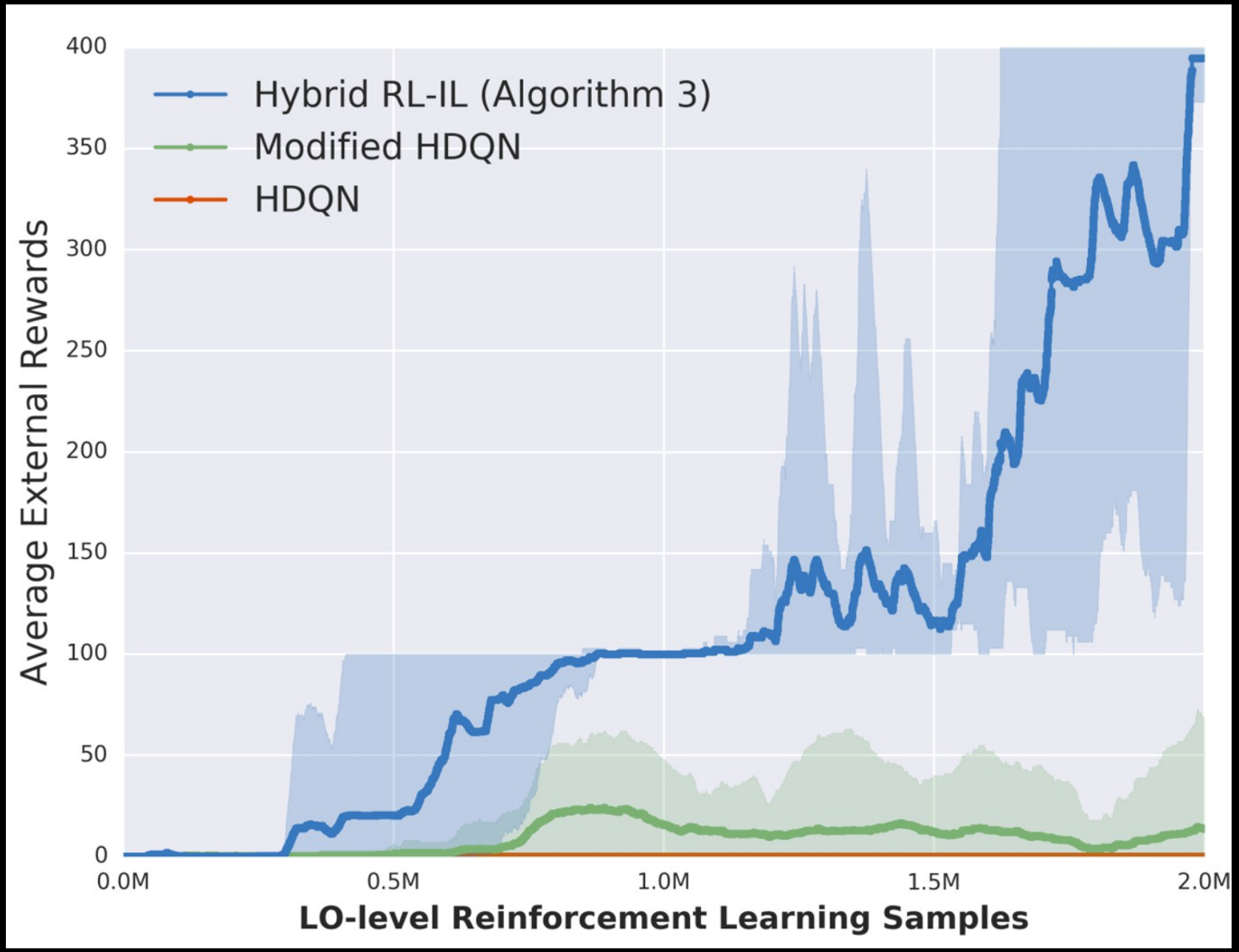


Key insight: **verifying if low-level trajectory is successful is cheaper than labeling low-level trajectory**

→ labeling effort = high-level horizon + low-level horizon
only a fraction of the full horizon (as low as sqrt of the full horizon)

→ subpolicies are only learnt in the relevant part of the state space

combination via hierarchies



refs

- Dagger: A reduction of imitation and structured prediction to no-regret online learning; Ross, Gordon, Bagnell; ICML 2011
- Aggrevate: Ross & Bagnell, Reinforcement and Imitation Learning via No-Regret Learning; Arxiv 2013
- LOLS: Chang, Krishnamurthy, Agarwal, D, Langford; Learning to search better than your teacher; ICML 2015 (Follow-up by D, L, Sharaf, ICLR 2018)
- MCTree: Browne et al., A Survey of MC Tree Search Methods, 2012
- Hierarchies: Le, Jiang, …, Hierarchical Imitation

imitation learning summary

successes:

- if all you have are demonstrations, life can be difficult
- if you have expert, iterate to get right state distribution
- experts come from people or planning
- several ways to combine experts and reinforcement

open problems:

- what is the right way to incorporate experts?
- can we learn from observations of experts?
- how to reduce # of expert samples needed?

Thank you! Queries!