DAGGER and Friends

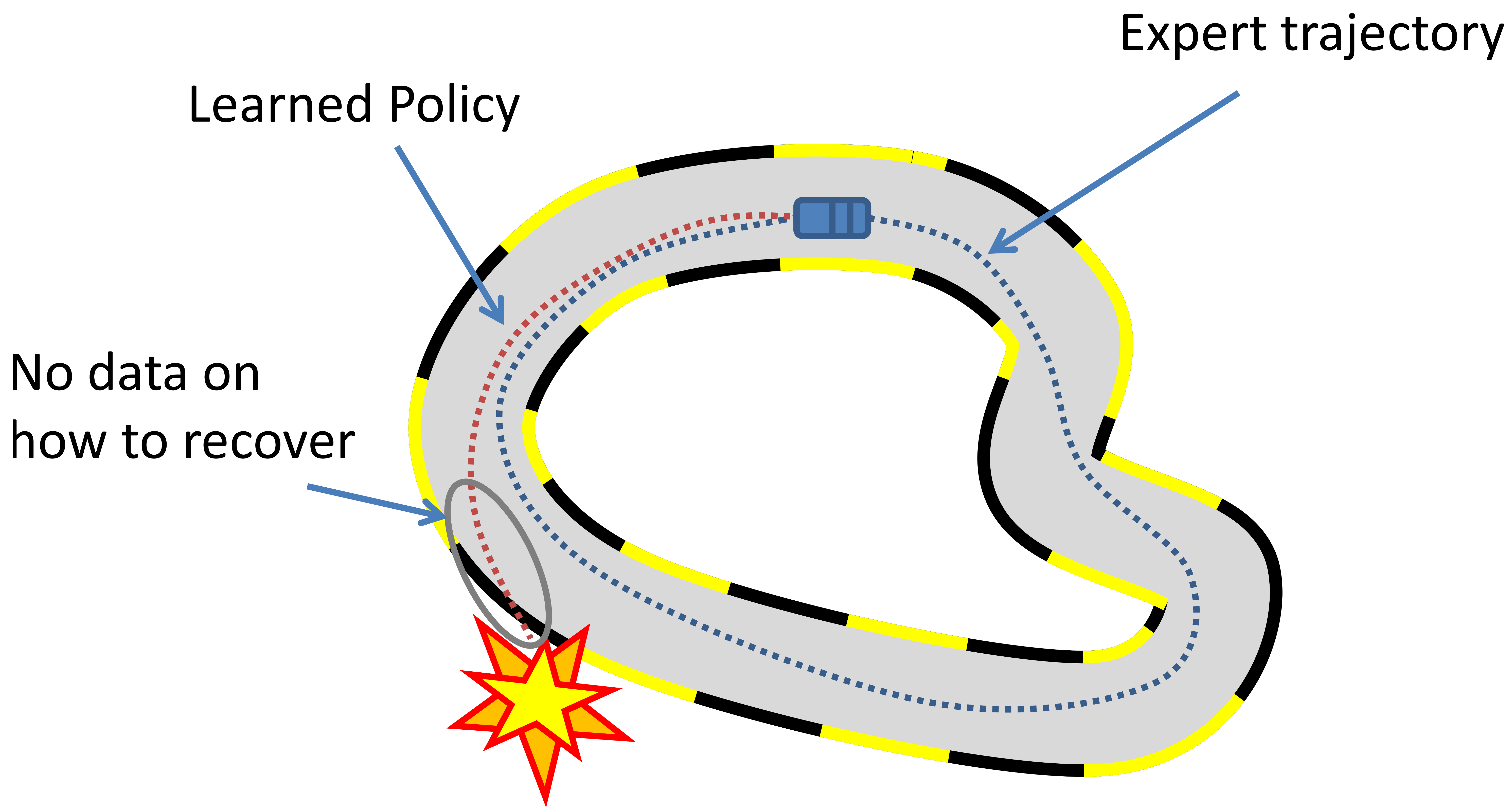
References:

1. **A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning**, Ross, Gordon & Bagnell (2010). *DAGGER algorithm*
2. **Reinforcement and Imitation Learning via Interactive No-Regret Learning** Ross & Bagnell (2014). *AGGREVATE algorithm*
3. **Deep Learning for Real-Time Atari Game Play Using Offline Monte-Carlo Tree Search Planning** Guo et al. (2014)
4. **SEARN in Practice** Daume et al. (2006)

John Schulman

2015/10/5

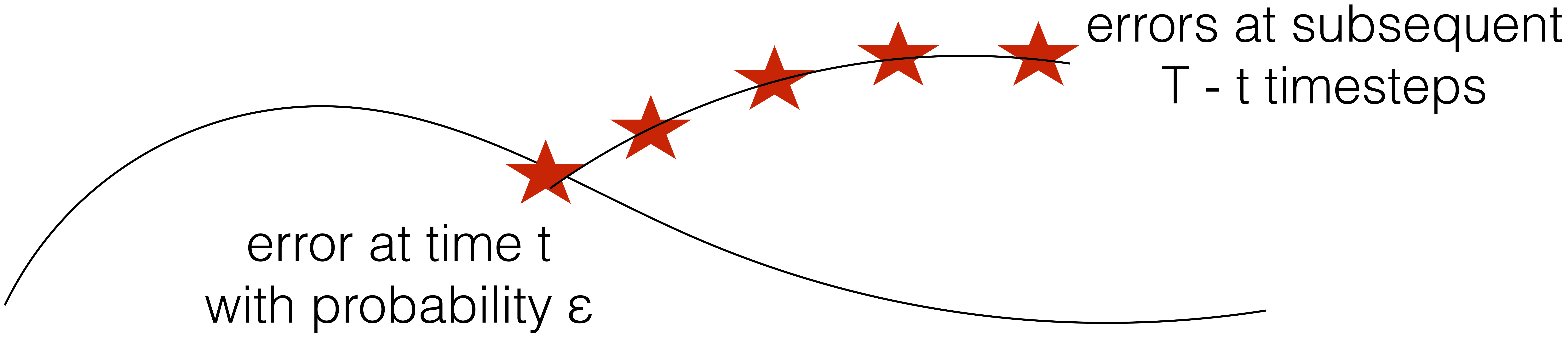
Data Mismatch Problem



Data Mismatch Problem

	supervised learning	supervised learning + control (NAIVE)
train	$(x,y) \sim D$	$s \sim d_{\pi^*}$
test	$(x,y) \sim D$	$s \sim d_{\pi}$

Compounding Errors



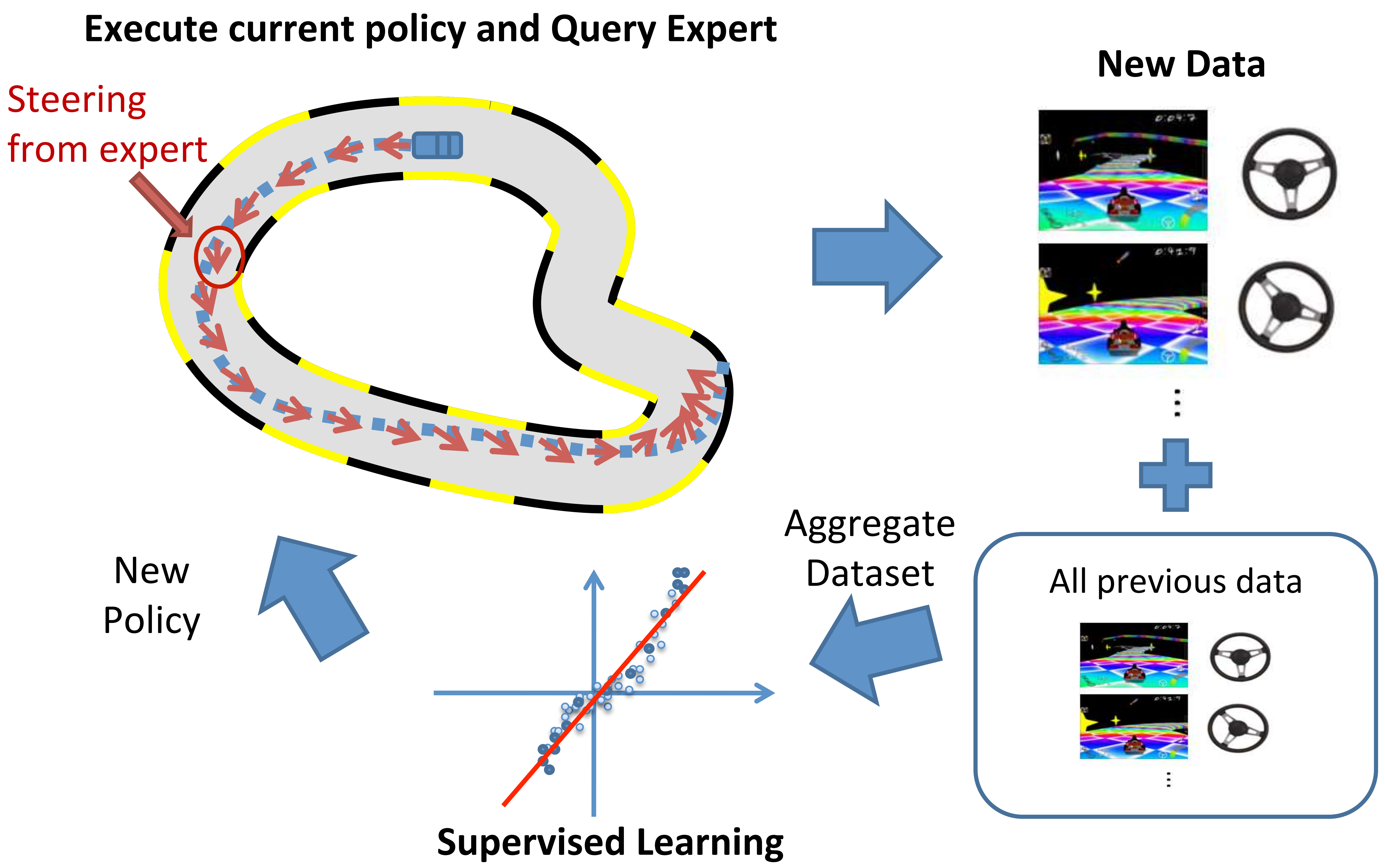
$$E[\text{Total errors}] \lesssim \varepsilon(T + (T-1) + (T-2) + \ldots + 1) \propto \varepsilon T^2$$

Forward Algorithm

Initialize $\pi_1, \pi_2, \ldots, \pi_T$ arbitrarily.
for $t = 1$ **to** T **do**
  Sample multiple t-step trajectories by executing the policies $\pi_1, \pi_2, \ldots, \pi_{t-1}$, starting from initial states drawn from the initial state distribution.
  Query expert for states encountered at time step t.
  Get dataset $\mathcal{D} = \{(s_t, \pi^*(s_t))\}$ of states, actions taken by expert at time step t.
  Train classifier $\pi_t = \arg\min_{\pi \in \Pi} \sum_{(s,a) \in \mathcal{D}} \ell(s, a, \pi)$.
end for
Return non-stationary policy $\hat{\pi}$, such that at time t in state s, $\hat{\pi}(s,t) = \pi_t(s)$

E[total errors] $\lesssim \varepsilon T$

DAGGER


Execute current policy and Query Expert
Steering
from expert
New Data
All previous data
Aggregate
Dataset
Supervised Learning
New
Policy

DAGGER

Initialize $\mathcal{D} \leftarrow \emptyset$.
Initialize $\hat{\pi}_1$ to any policy in Π.
for $i = 1$ **to** N **do**
  Let $\pi_i = \beta_i \pi^* + (1 - \beta_i)\hat{\pi}_i$.
  Sample T-step trajectories using π_i.
  Get dataset $\mathcal{D}_i = \{(s, \pi^*(s))\}$ of visited states by π_i and actions given by expert.
  Aggregate datasets: $\mathcal{D} \leftarrow \mathcal{D} \bigcup \mathcal{D}_i$.
  Train classifier $\hat{\pi}_{i+1}$ on $\mathcal{D}$ (or use online learner to get $\hat{\pi}_{i+1}$ given new data $\mathcal{D}_i$).
end for
Return best $\hat{\pi}_i$ on validation.

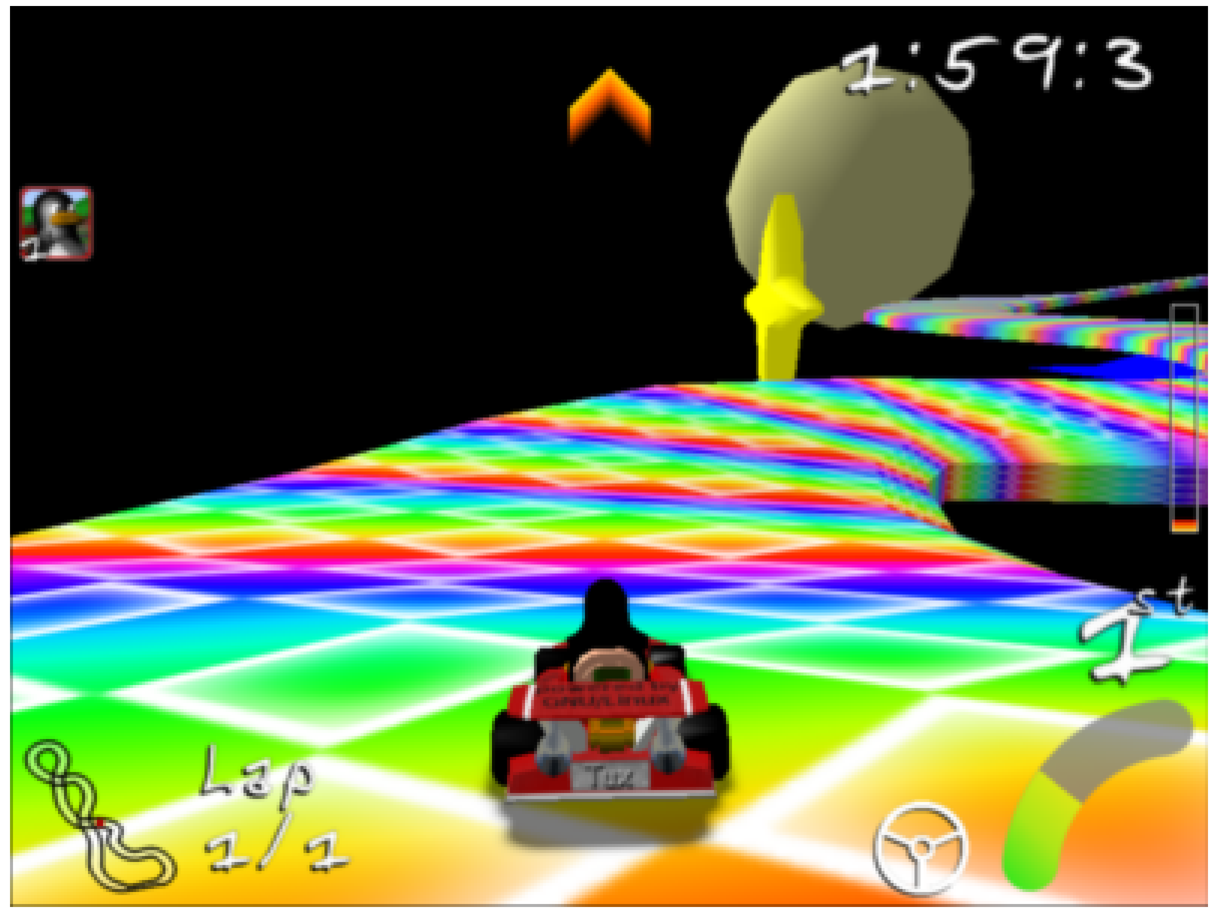
AGGREVATE

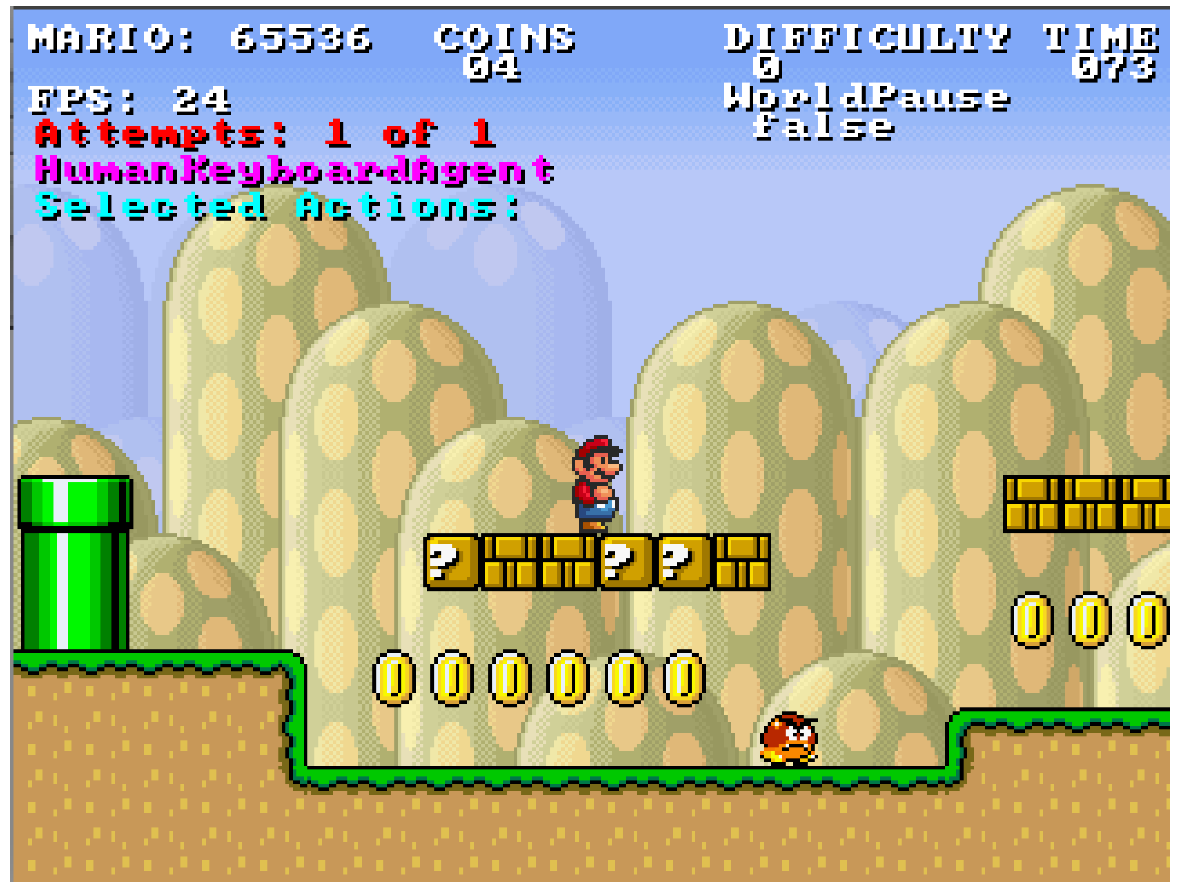
Initialize $\mathcal{D} \leftarrow \emptyset$.
Initialize $\hat{\pi}_1$ to any policy in Π.
for $i = 1$ **to** N **do**
  Let $\pi_i = \beta_i \pi^* + (1 - \beta_i)\hat{\pi}_i$.
  Sample T-step trajectories using π_i.
  Get dataset $\mathcal{D}_i = \{(s, \pi^*(s))\}$ of visited states by π_i and actions given by expert.
  Aggregate datasets: $\mathcal{D} \leftarrow \mathcal{D} \bigcup \mathcal{D}_i$.
  Train classifier $\hat{\pi}_{i+1}$ on $\mathcal{D}$ (or use online learner to get $\hat{\pi}_{i+1}$ given new data $\mathcal{D}_i$).
end for
Return best $\hat{\pi}_i$ on validation.

A*(s,a) for all actions a

i.e., minimize $\Sigma_n A^*(s_n, \pi(s_n))$

Empirical Demonstrations





Online Learning + Regret

- Learn from a stream of data, might be non-stationary or adversarial
- At n^{th} step, algorithm chooses π_n, receives loss $L_n(\pi_n)$
- Want to minimize $\Sigma_n L_n(\pi_n)$
- Regret: $\Sigma_n L_n(\pi_n) - \min_\pi \Sigma_n L_n(\pi)$
  - e.g. for convex L with online gradient descent, one can show that total regret ~ $\sqrt{T}$

Great review: Shalev-Shwartz, Shai, and Yoram Singer. "Online learning: Theory, algorithms, and applications." (2007).

AGGREVATE: Theory

$$\eta(\pi) - \eta(\pi^*) = \mathbb{E}_{\tau;\pi}\left[\sum_{t=1}^{T} A^{\pi}(s_t, a_t)\right]$$

$$L_n(\pi) := \mathbb{E}_{\tau;\pi_n}\left[\sum_{t=1}^{T} A^{\pi^*}(s_t, \pi(s_t))\right]$$

Suboptimality of n[th] policy: $L_n(\pi_n)$

AGGREVATE (β=0)

- At n^{th} step, sample trajectories using π_n
  - suboptimality is $L_n(\pi_n)$
- Update policy based on new data to get π_{n+1}
  - e.g., take $\pi_{n+1} = \text{argmin}_\pi \Sigma_n L_n(\pi)$

AGGREVATE (β=0)

- Now, consider π, obtained by randomly sampling n in {1,2,…,N}

$$\eta(\bar{\pi}) - \eta(\pi^*) = \frac{1}{N}\sum_{n=1}^{N} L_n(\pi_n)$$

- => Suboptimality is bounded by regret of learning algorithm

AGGREVATE (β=0)

- Sample trajectories

Application to Atari

Deep Learning for Real-Time Atari Game Play Using Offline Monte-Carlo Tree Search Planning

Xiaoxiao Guo
Computer Science and Eng.
University of Michigan
guoxiao@umich.edu

Satinder Singh
Computer Science and Eng.
University of Michigan
baveja@umich.edu

Honglak Lee
Computer Science and Eng.
University of Michigan
honglak@umich.edu

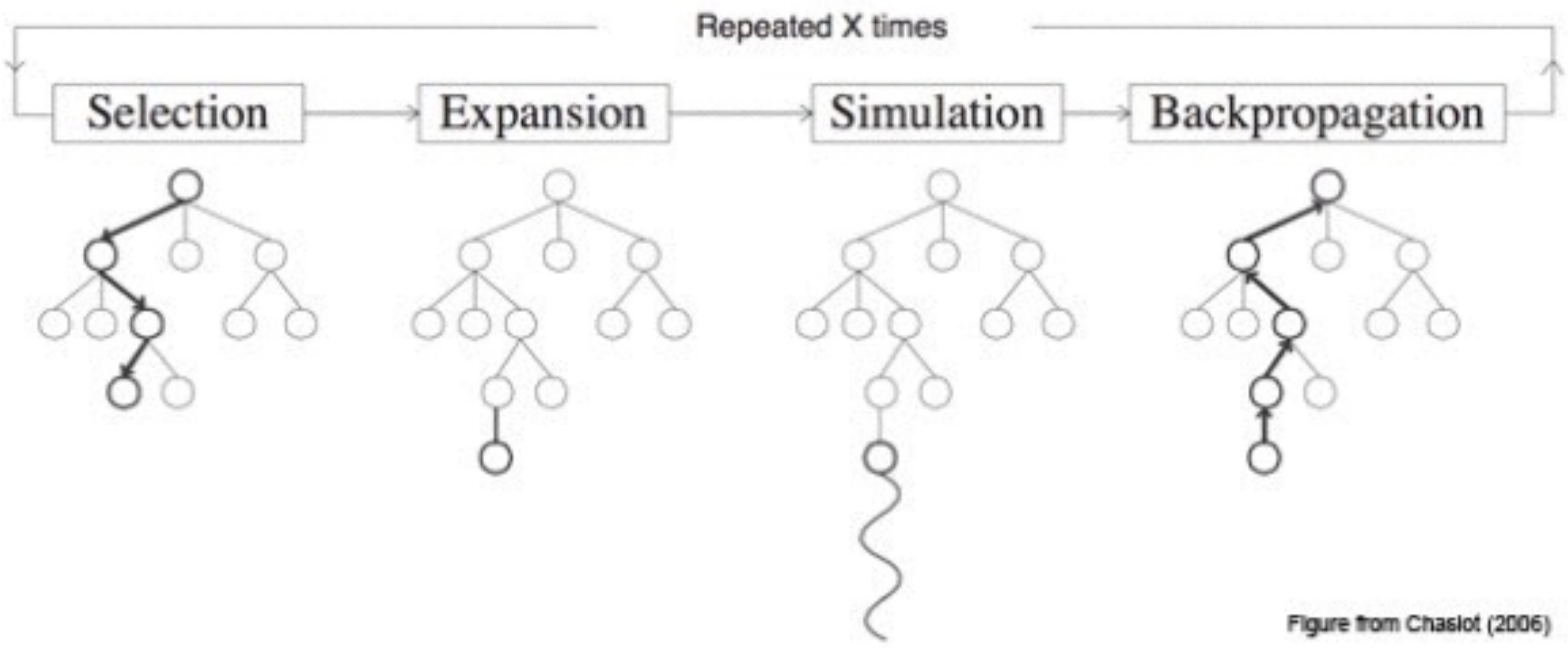
Richard Lewis
Department of Psychology
University of Michigan
rickl@umich.edu

Xiaoshi Wang
Computer Science and Eng.
University of Michigan
xiaoshiw@umich.edu

NIPS 2014

Monte Carlo Tree Search (UCT) + ConvNet Policy/Classifier

MONTE CARLO TREE SEARCH

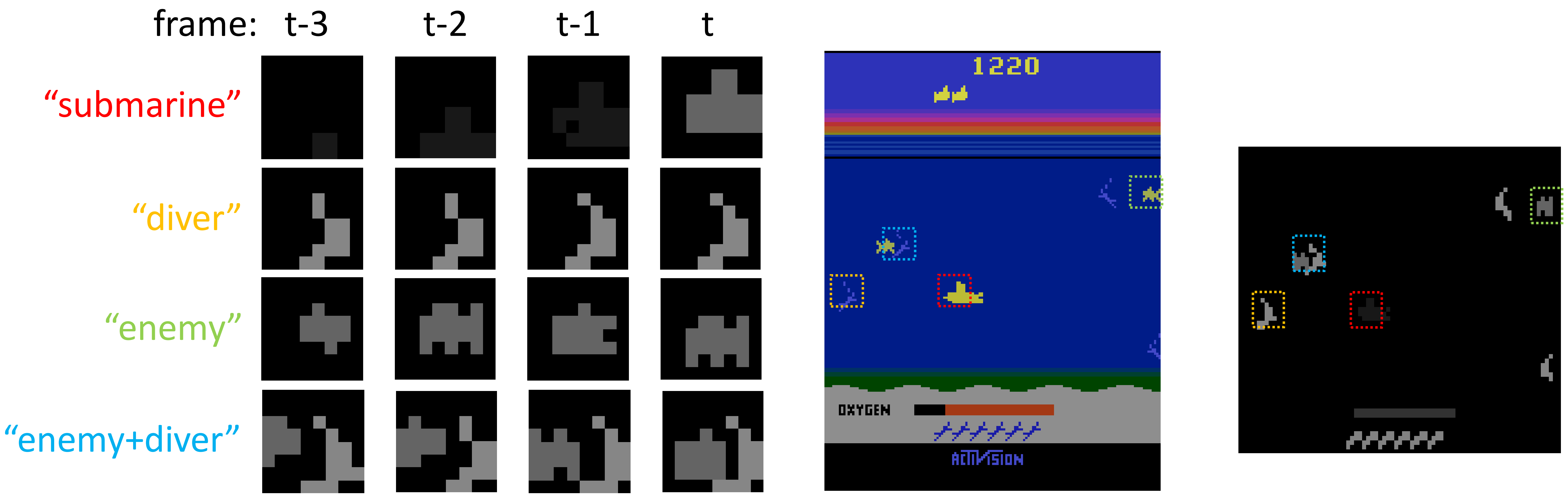


Coulom, Rémi. "Efficient selectivity and backup operators in Monte-Carlo tree search." Computers and games. Springer Berlin Heidelberg, 2007. 72-83.

Kocsis, Levente, and Csaba Szepesvári. "Bandit based monte-carlo planning." Machine Learning: ECML 2006. Springer Berlin Heidelberg, 2006. 282-293. **(UCT Algorithm)**

Kearns, Michael, Yishay Mansour, and Andrew Y. Ng. "A sparse sampling algorithm for near-optimal planning in large Markov decision processes." Machine Learning 49.2-3 (2002): 193-208.

Application to Atari



cool finding — low level filters show game objects

Application to Atari

Agent	*B.Rider*	*Breakout*	*Enduro*	*Pong*	*Q*bert*	*Seaquest*	*S.Invaders*
DQN	4092	168	470	20	1952	1705	581
-best	5184	225	661	21	4500	1740	1075
UCC	5342 (20)	175 (5.63)	558 (14)	19 (0.3)	11574(44)	2273 (23)	672 (5.3)
-best	10514	351	942	21	29725	5100	1200
-greedy	5676	269	692	21	19890	2760	680
UCC-I	5388 (4.6)	215 (6.69)	601 (11)	19 (0.14)	13189 (35.3)	2701 (6.09)	670 (4.24)
-best	10732	413	1026	21	29900	6100	910
-greedy	5702	380	741	21	20025	2995	692
UCR	2405 (12)	143 (6.7)	566 (10.2)	19 (0.3)	12755 (40.7)	1024 (13.8)	441 (8.1)

Table 2: Performance (game scores) of the off-line UCT game playing agent.

Agent	*B.Rider*	*Breakout*	*Enduro*	*Pong*	*Q*bert*	*Seaquest*	*S.Invaders*
UCT	7233	406	788	21	18850	3257	2354

but… 800 games * 1000 actions/game * 10000 rollouts/action * 300 steps/rollout = 2.4e12 steps