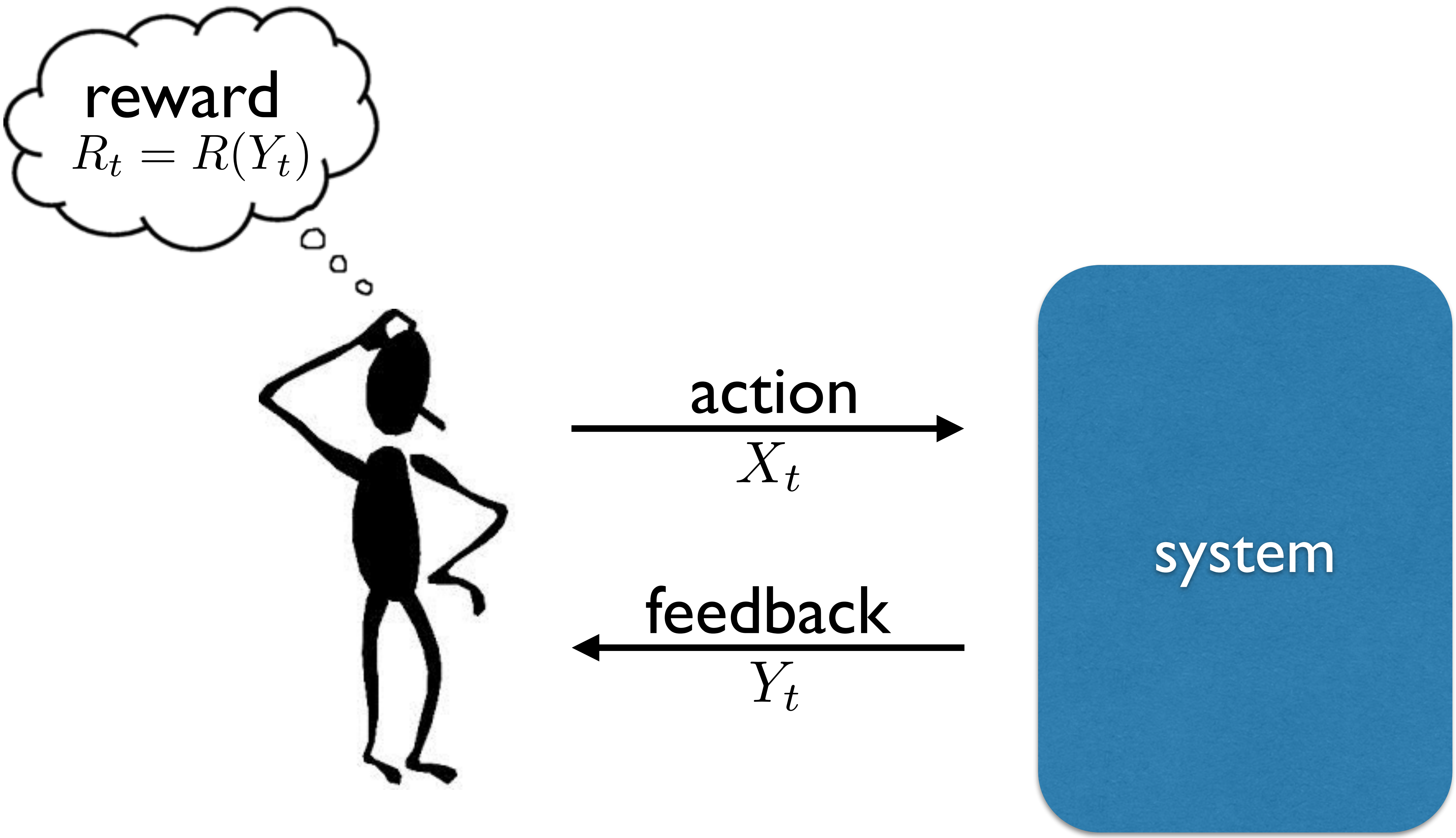
Learning to Optimize

Delayed Consequences

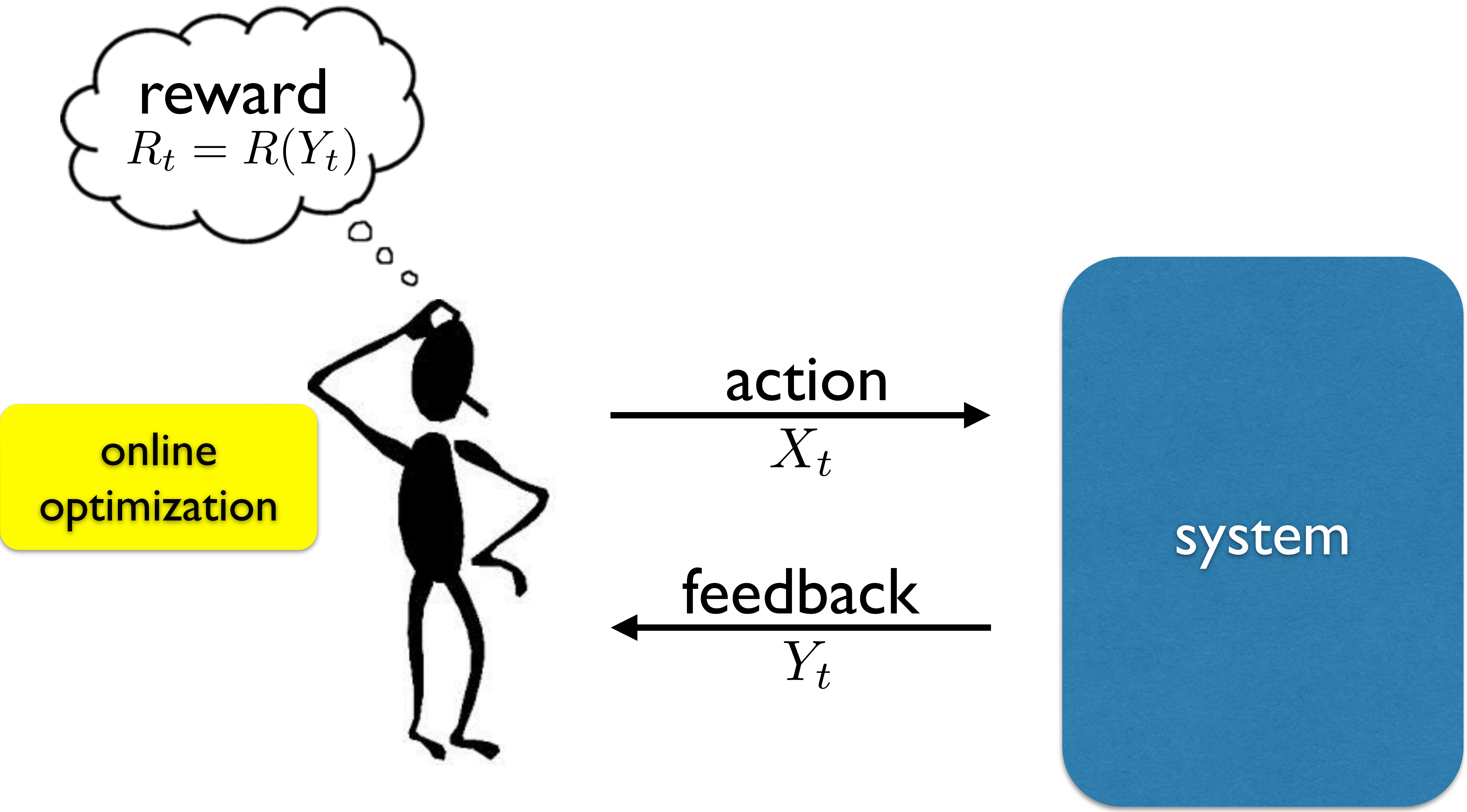
Benjamin Van Roy

work done with Ian Osband, Dan Russo, Zheng Wen

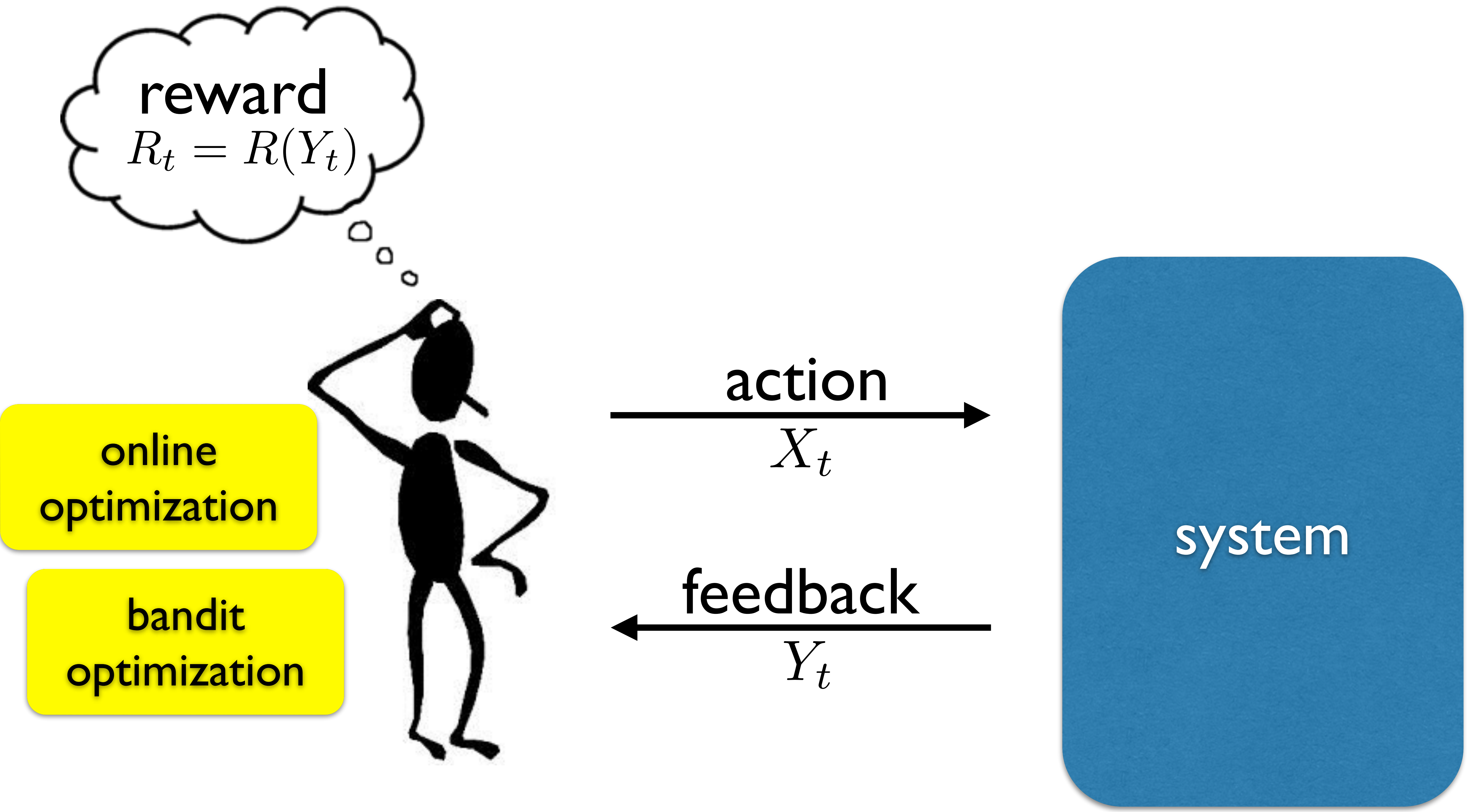
Learning to Optimize



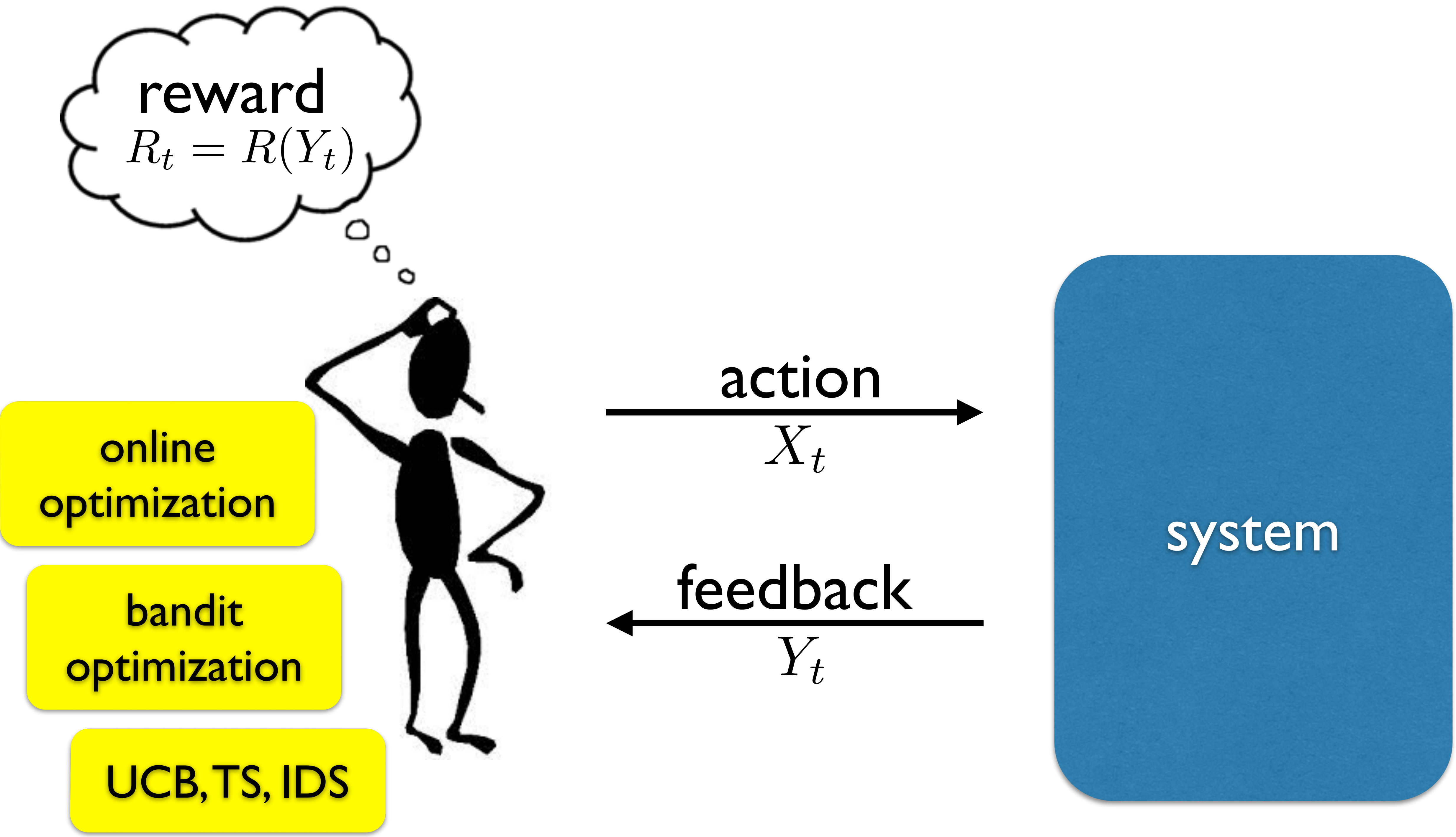
Learning to Optimize



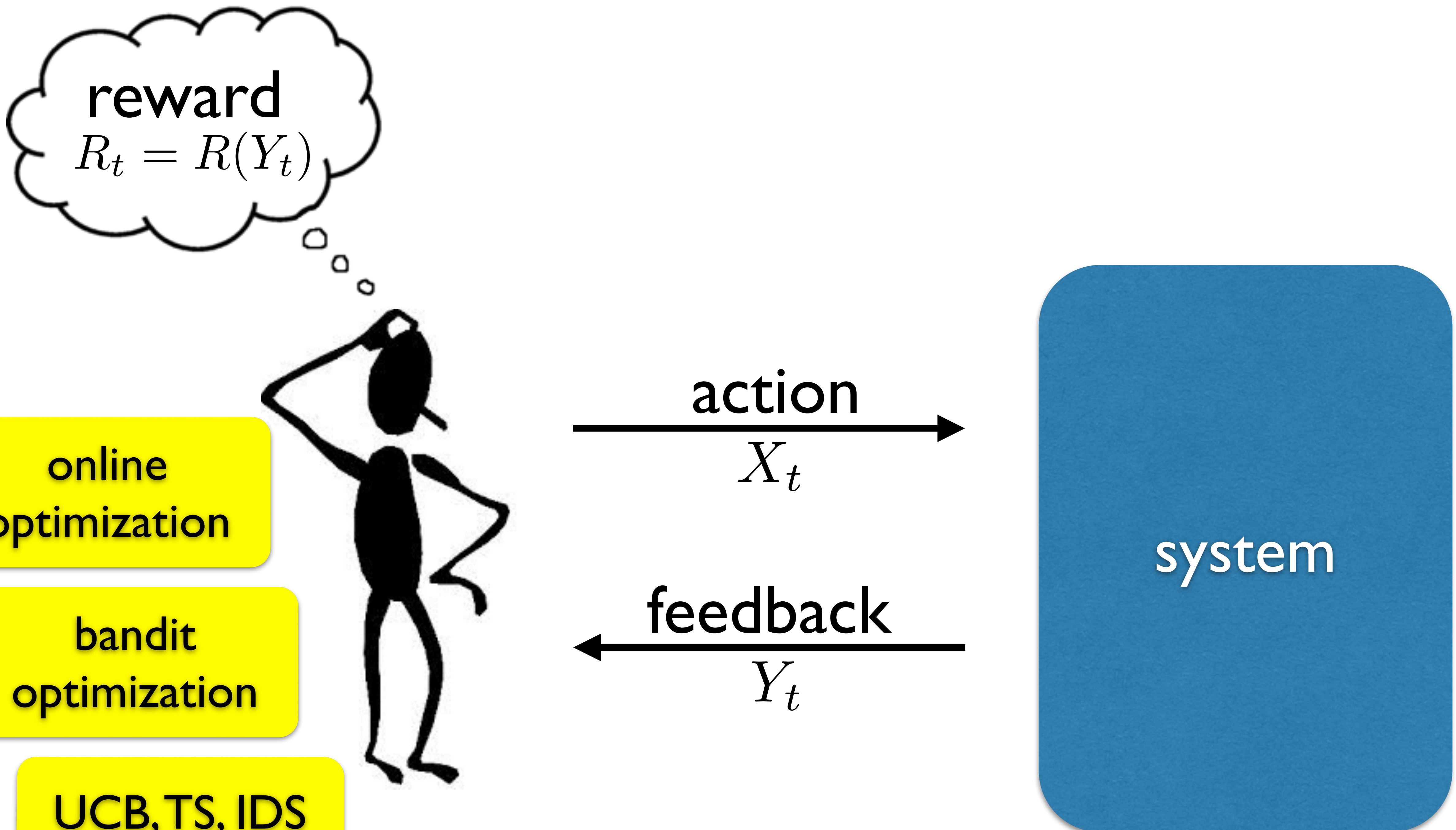
Learning to Optimize



Learning to Optimize



Learning to Optimize



online optimization

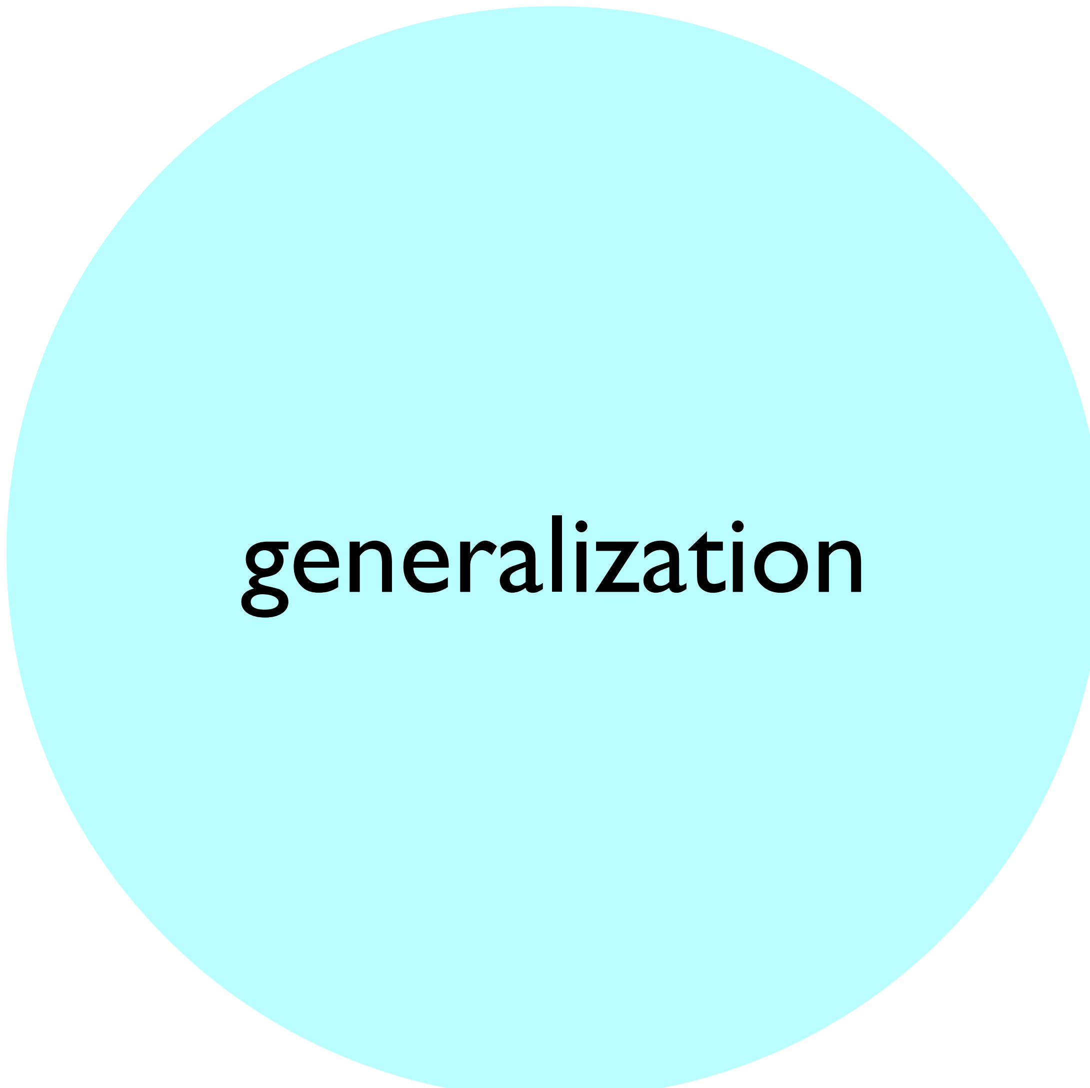
bandit optimization

UCB, TS, IDS

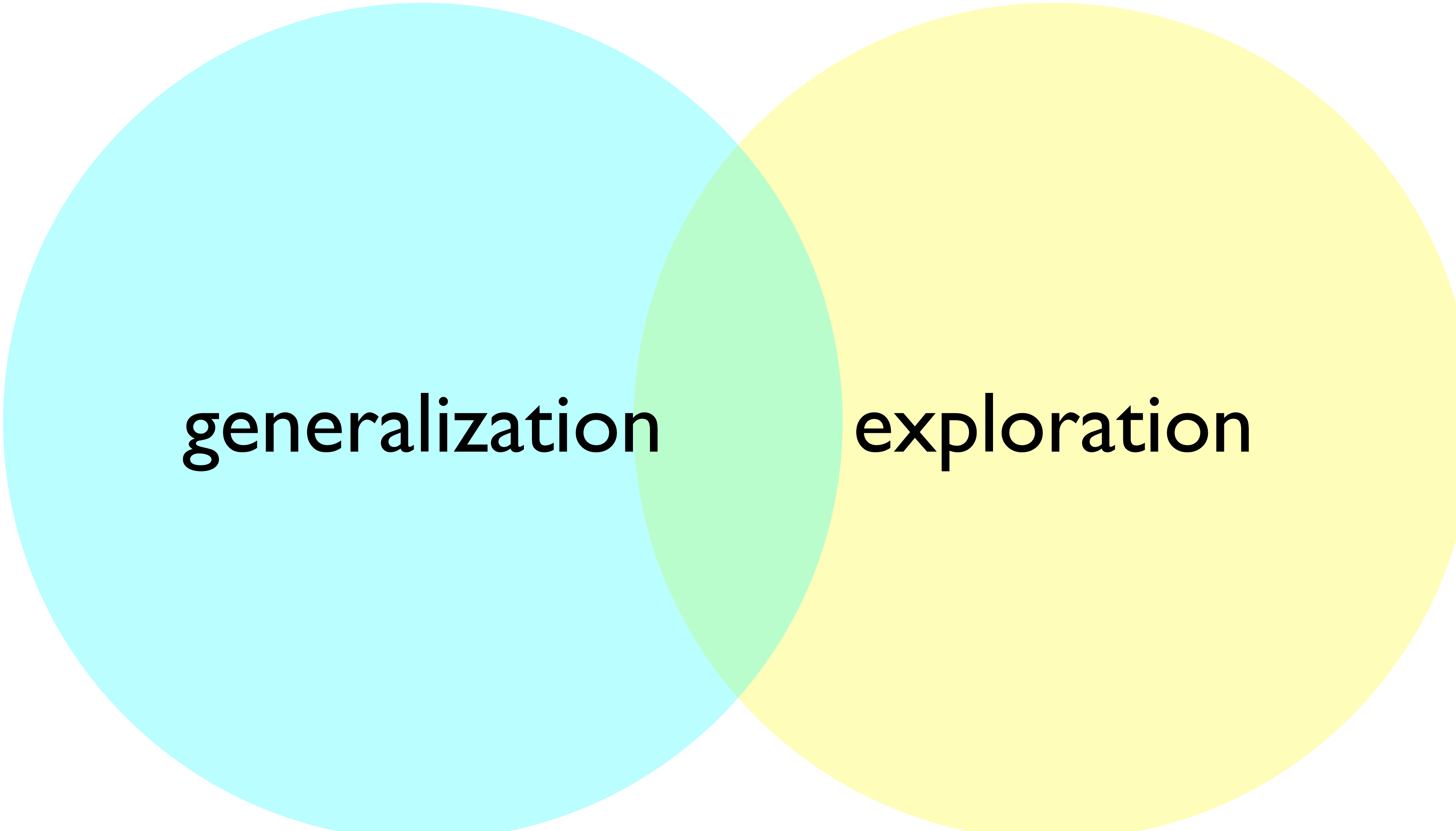
delayed consequences

Elements of Statistical Learning

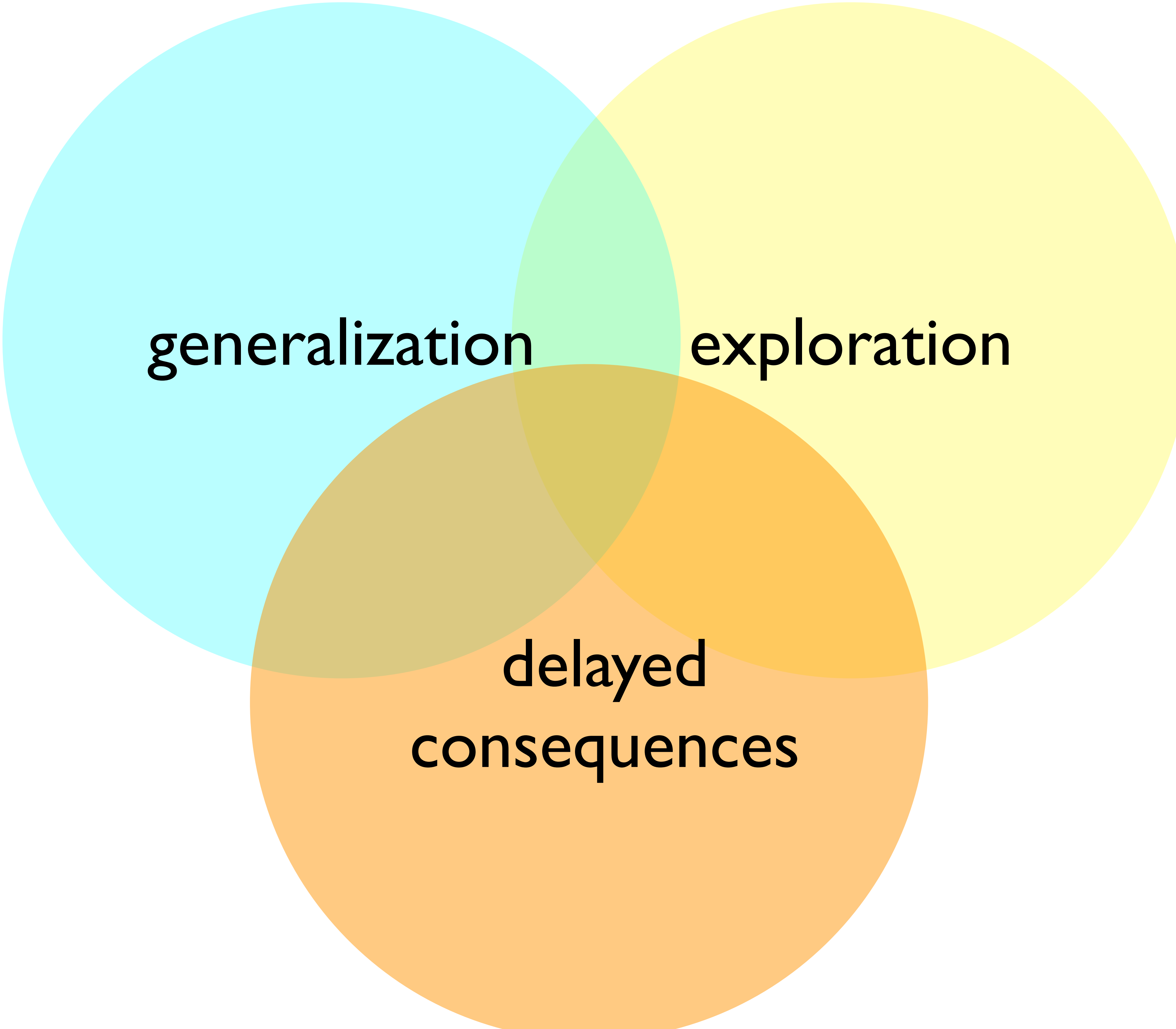
Elements of Statistical Learning



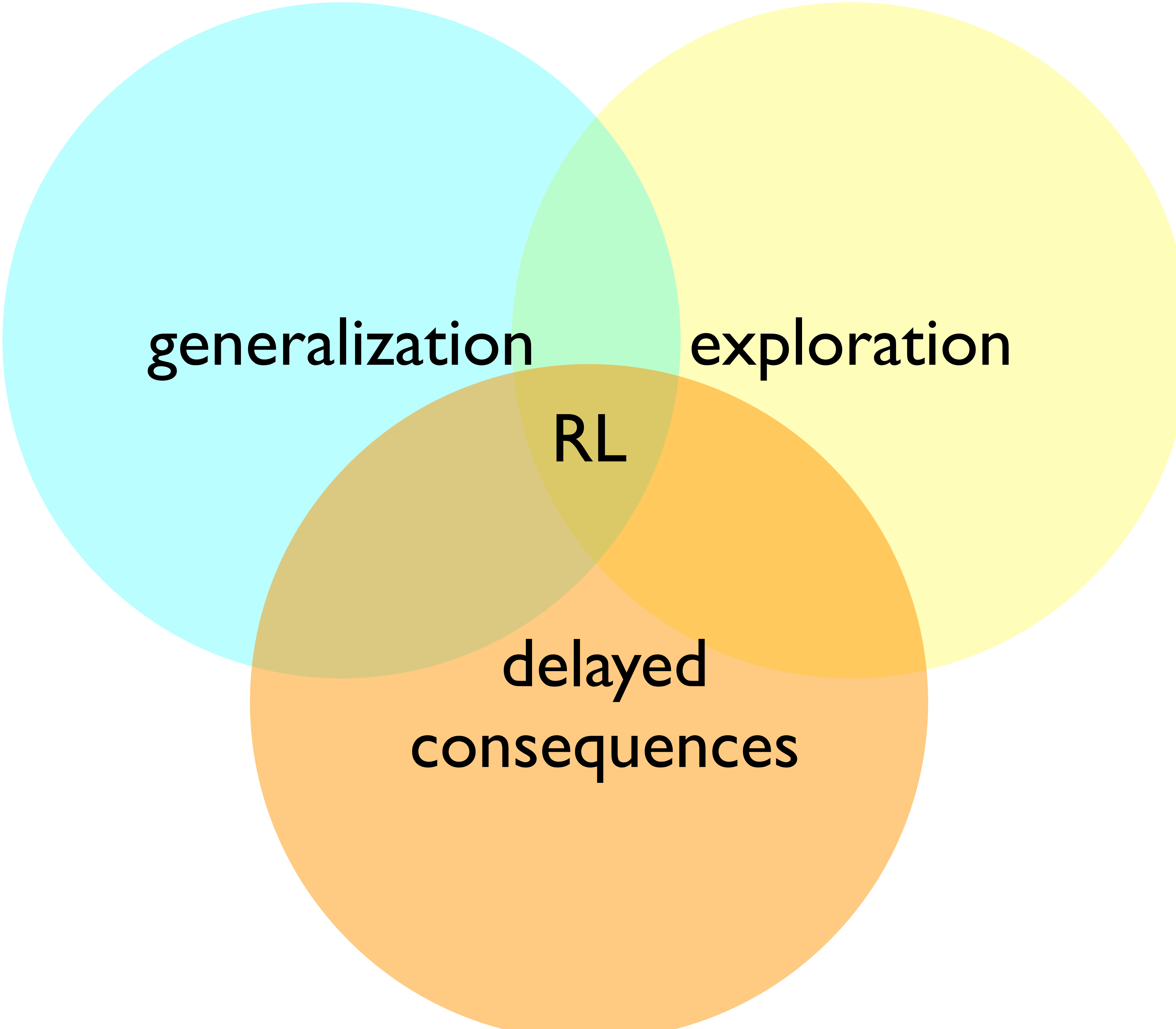
Elements of Statistical Learning



Elements of Statistical Learning



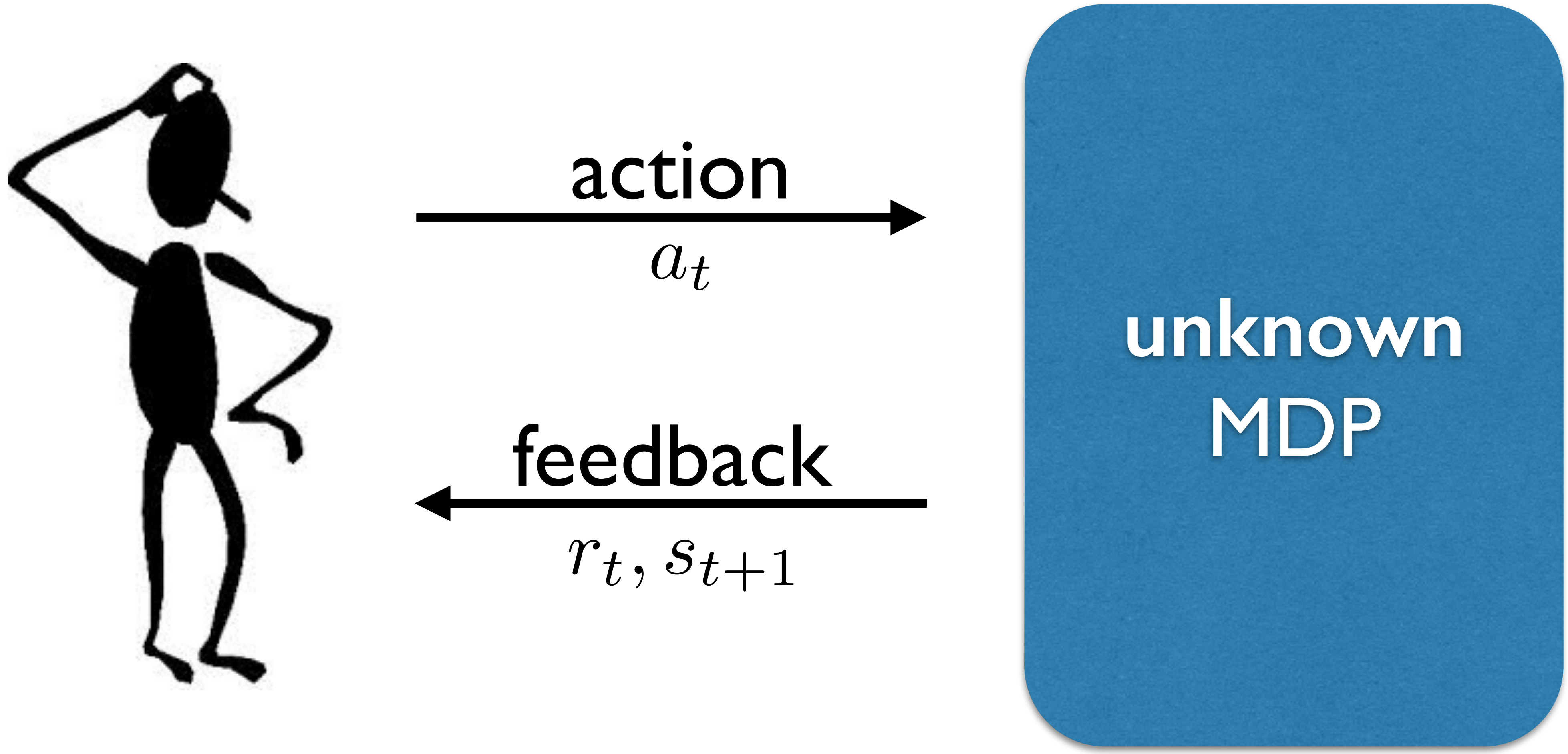
Elements of Statistical Learning



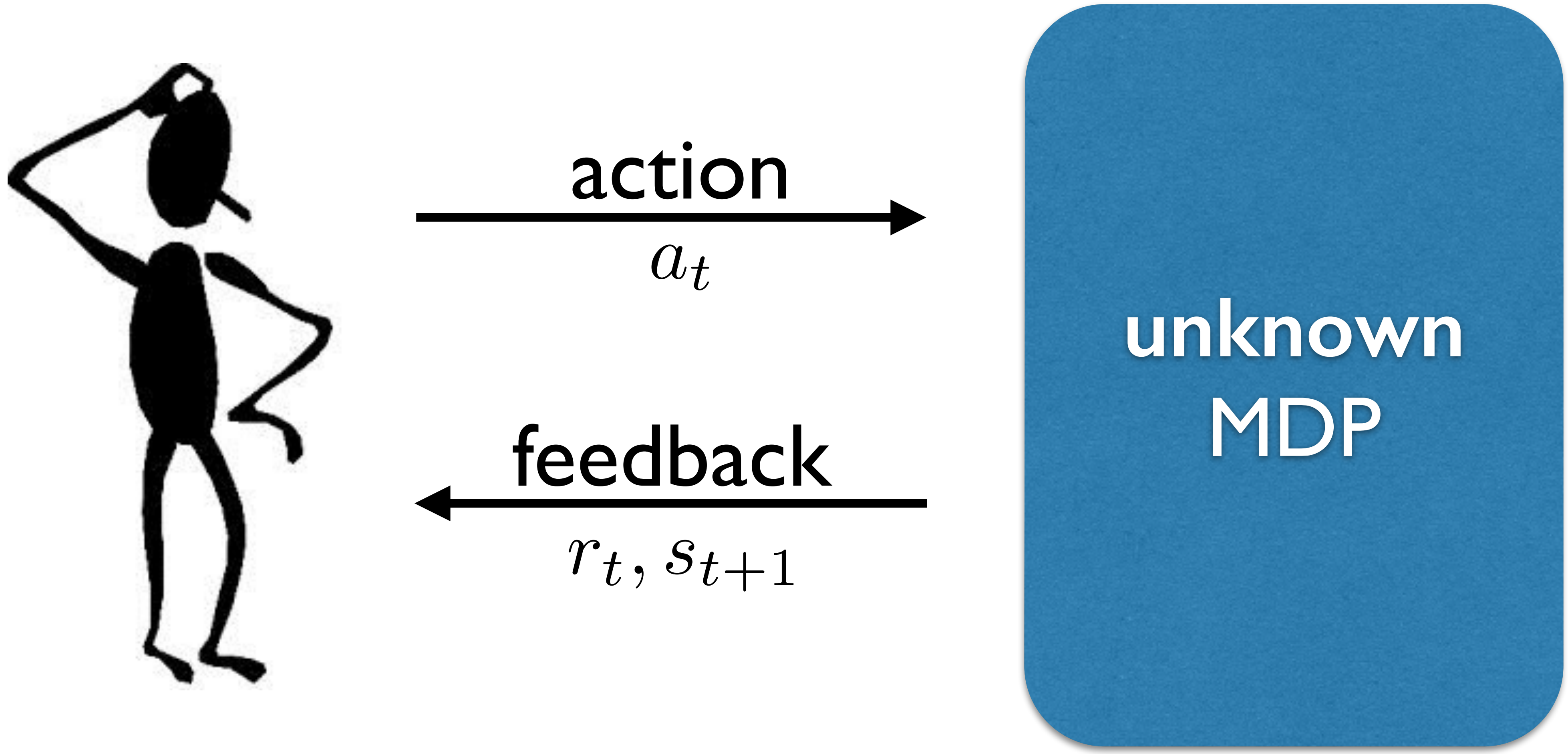
Do delayed consequences matter?

- Robotics
- Web site content optimization
- Online education
- Medical treatments

A Reinforcement Learning Problem



A Reinforcement Learning Problem



different from the dynamic programming problem

MDP Formulation

MDP Formulation

- Unknown MDP
  - Finite horizon, state space, action space
  - Initial state s_0
  - Time-inhomogenous transition and rewards distributions
  - Rewards in [0,1]

MDP Formulation

- Unknown MDP
  - Finite horizon, state space, action space
  - Initial state s_0
  - Time-inhomogenous transition and rewards distributions
  - Rewards in [0,1]

- Policy $\pi = (\pi_0, \ldots, \pi_{H-1})$

MDP Formulation

- Unknown MDP
  - Finite horizon, state space, action space
  - Initial state s_0
  - Time-inhomogenous transition and rewards distributions
  - Rewards in [0,1]

- Policy $\pi = (\pi_0, \ldots, \pi_{H-1})$

$a_h = \pi_h(s_h)$

- Objective $V_0^\pi(s_0) = \mathbb{E}_\pi \left[\sum_{h=0}^{H-1} r_h \right]$

MDP Formulation

- Unknown MDP
  - Finite horizon, state space, action space
  - Initial state s_0
  - Time-inhomogenous transition and rewards distributions
  - Rewards in [0,1]

- Policy $\pi = (\pi_0, \ldots, \pi_{H-1})$

$$a_h = \pi_h(s_h)$$

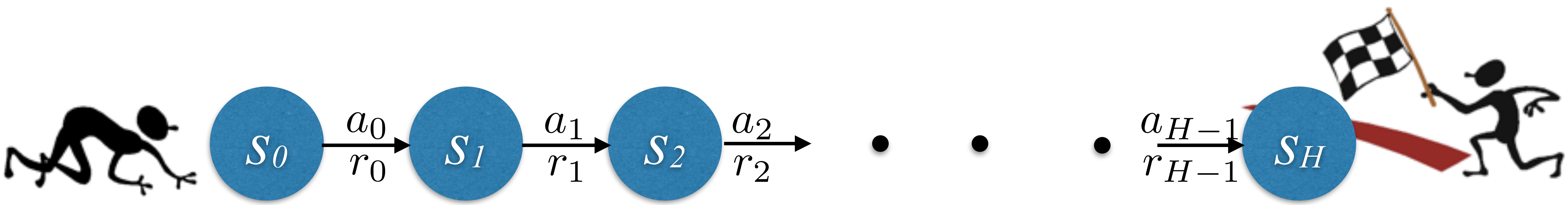
- Objective $V_0^\pi(s_0) = \mathbb{E}_\pi \left[\sum_{h=0}^{H-1} r_h \right]$

- Optimal value $V_0^*(s_0) = \max_\pi V_0^\pi(s_0)$

Reinforcement Learning Algorithms

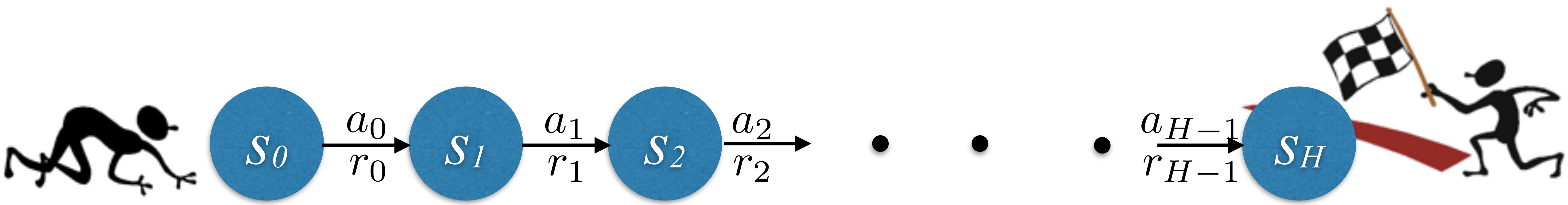
Reinforcement Learning Algorithms

- Episodic learning



Reinforcement Learning Algorithms

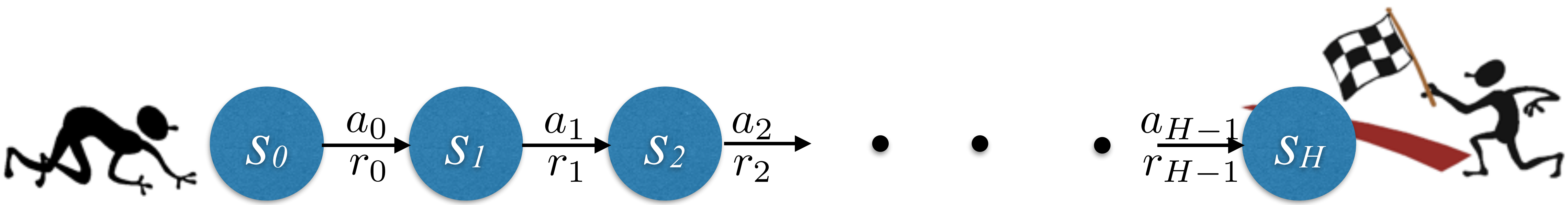
- Episodic learning



- Reinforcement learning algorithm

Reinforcement Learning Algorithms

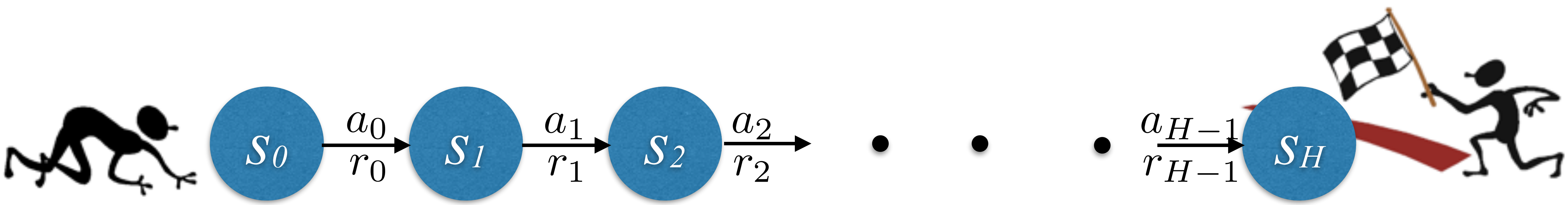
- Episodic learning



- Reinforcement learning algorithm
  - Given observations made through episode $\ell - 1$

Reinforcement Learning Algorithms

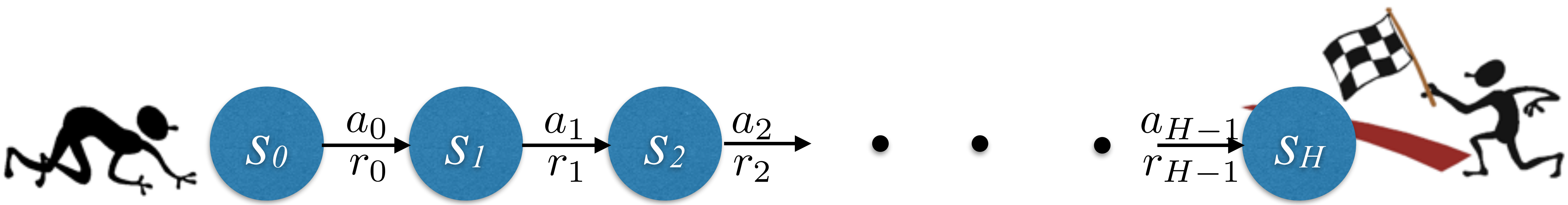
- Episodic learning



- Reinforcement learning algorithm
  - Given observations made through episode $\ell - 1$
  - Select policy $\pi^\ell = (\pi_0^\ell, \ldots, \pi_{H-1}^\ell)$

Reinforcement Learning Algorithms

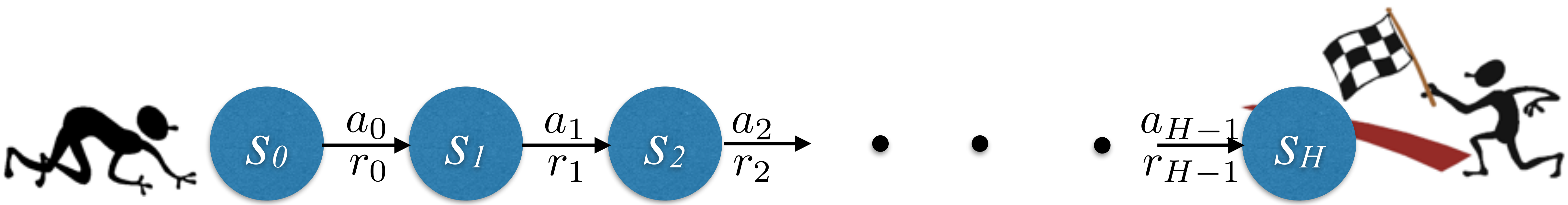
- Episodic learning



- Reinforcement learning algorithm
  - Given observations made through episode $\ell - 1$
  - Select policy $\pi^\ell = (\pi_0^\ell, \ldots, \pi_{H-1}^\ell)$
  - Apply actions $a_h^\ell = \pi_h^\ell(s_h^\ell)$

Reinforcement Learning Algorithms

- Episodic learning



- Reinforcement learning algorithm
  - Given observations made through episode $\ell - 1$
  - Select policy $\pi^\ell = (\pi_0^\ell, \ldots, \pi_{H-1}^\ell)$
  - Apply actions $a_h^\ell = \pi_h^\ell(s_h^\ell)$

- Regret

$$\text{Regret}(L) = \sum_{\ell=1}^{L} \left(V_0^*(s_0) - V^{\pi^\ell}(s_0) \right)$$

Exploration in Theory and Practice

Exploration in Theory and Practice

- Theory of “efficient RL exploration”

Exploration in Theory and Practice

- Theory of "efficient RL exploration"
  - Focus on *tabula rasa* case (no generalization)

Exploration in Theory and Practice

- Theory of “efficient RL exploration”
  - Focus on *tabula rasa* case (no generalization)
  - Focus on regret bounds, not empirical performance

Exploration in Theory and Practice

- Theory of "efficient RL exploration"
  - Focus on *tabula rasa* case (no generalization)
  - Focus on regret bounds, not empirical performance
  - Algorithms serve as constructive proofs

Exploration in Theory and Practice

- Theory of “efficient RL exploration”
  - Focus on *tabula rasa* case (no generalization)
  - Focus on regret bounds, not empirical performance
  - Algorithms serve as constructive proofs
  - Interesting insights

Exploration in Theory and Practice

- Theory of “efficient RL exploration”
  - Focus on *tabula rasa* case (no generalization)
  - Focus on regret bounds, not empirical performance
  - Algorithms serve as constructive proofs
  - Interesting insights
- Exploration in practice

Exploration in Theory and Practice

- Theory of "efficient RL exploration"
  - Focus on *tabula rasa* case (no generalization)
  - Focus on regret bounds, not empirical performance
  - Algorithms serve as constructive proofs
  - Interesting insights

- Exploration in practice
  - Generalization is critical

Exploration in Theory and Practice

- Theory of "efficient RL exploration"
  - Focus on *tabula rasa* case (no generalization)
  - Focus on regret bounds, not empirical performance
  - Algorithms serve as constructive proofs
  - Interesting insights

- Exploration in practice
  - Generalization is critical
    - Parameterized value functions or policies

Exploration in Theory and Practice

- Theory of "efficient RL exploration"
  - Focus on *tabula rasa* case (no generalization)
  - Focus on regret bounds, not empirical performance
  - Algorithms serve as constructive proofs
  - Interesting insights

- Exploration in practice
  - Generalization is critical
    - Parameterized value functions or policies
  - Inefficient exploration schemes

Exploration in Theory and Practice

- Theory of "efficient RL exploration"
  - Focus on *tabula rasa* case (no generalization)
  - Focus on regret bounds, not empirical performance
  - Algorithms serve as constructive proofs
  - Interesting insights

- Exploration in practice
  - Generalization is critical
    - Parameterized value functions or policies
  - Inefficient exploration schemes
    - Dithering: Boltzmann, ϵ-greedy

Exploration in Theory and Practice

- Theory of "efficient RL exploration"
  - Focus on *tabula rasa* case (no generalization)
  - Focus on regret bounds, not empirical performance
  - Algorithms serve as constructive proofs
  - Interesting insights

- Exploration in practice
  - Generalization is critical
    - Parameterized value functions or policies
  - Inefficient exploration schemes
    - Dithering: Boltzmann, ϵ-greedy

open issue: how to explore efficiently alongside effective generalization methods

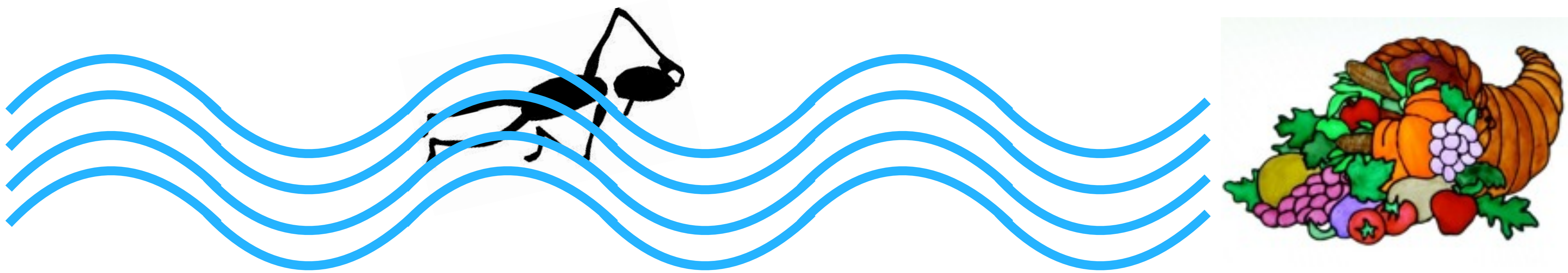
Key Insight from Theory of "Efficient RL"

Key Insight from Theory of “Efficient RL”

[Strehl-Littman, 2008]

Key Insight from Theory of "Efficient RL"

[Strehl-Littman, 2008]



Key Insight from Theory of “Efficient RL”

[Strehl-Littman, 2008]

Key Insight from Theory of "Efficient RL"

[Strehl-Littman, 2008]

planning to learn is critical

Key Insight from Theory of “Efficient RL”

[Strehl-Littman, 2008]

planning to learn is critical

dithering

Key Insight from Theory of "Efficient RL"

[Strehl-Littman, 2008]

planning to learn is critical

dithering

Key Insight from Theory of “Efficient RL”

[Strehl-Littman, 2008]

planning to learn is critical

dithering

myopia

Key Insight from Theory of "Efficient RL"

[Strehl-Littman, 2008]

planning to learn is critical

UCRL2

[Jaksch-Ortner-Auer, 2010]

UCRL2

[Jaksch-Ortner-Auer, 2010]

- To select π^{ℓ}

UCRL2

[Jaksch-Ortner-Auer, 2010]

- To select π^ℓ
  - At each *(s,a)*, construct confidence sets for

UCRL2

[Jaksch-Ortner-Auer, 2010]

- To select π^ℓ
  - At each *(s,a)*, construct confidence sets for
    - transition probability vector

UCRL2

[Jaksch-Ortner-Auer, 2010]

- To select π^ℓ
  - At each *(s,a)*, construct confidence sets for
    - transition probability vector
    - mean reward

UCRL2

[Jaksch-Ortner-Auer, 2010]

- To select π^ℓ
  - At each *(s,a)*, construct confidence sets for
    - transition probability vector
    - mean reward
  - Solve optimistic MDP $\max_{\pi,\hat{\mathcal{M}}} \mathbb{E}_{\pi,\hat{\mathcal{M}}}\left[\sum_{h=0}^{H-1} r_h\right]$

UCRL2

[Jaksch-Ortner-Auer, 2010]

- To select π^ℓ
  - At each *(s,a)*, construct confidence sets for
    - transition probability vector
    - mean reward
  - Solve optimistic MDP $\max_{\pi,\hat{\mathcal{M}}} \mathbb{E}_{\pi,\hat{\mathcal{M}}}\left[\sum_{h=0}^{H-1} r_h\right]$
- Regret bound $\mathrm{Regret}(L) = \tilde{O}\left(HS\sqrt{AHL}\right)$

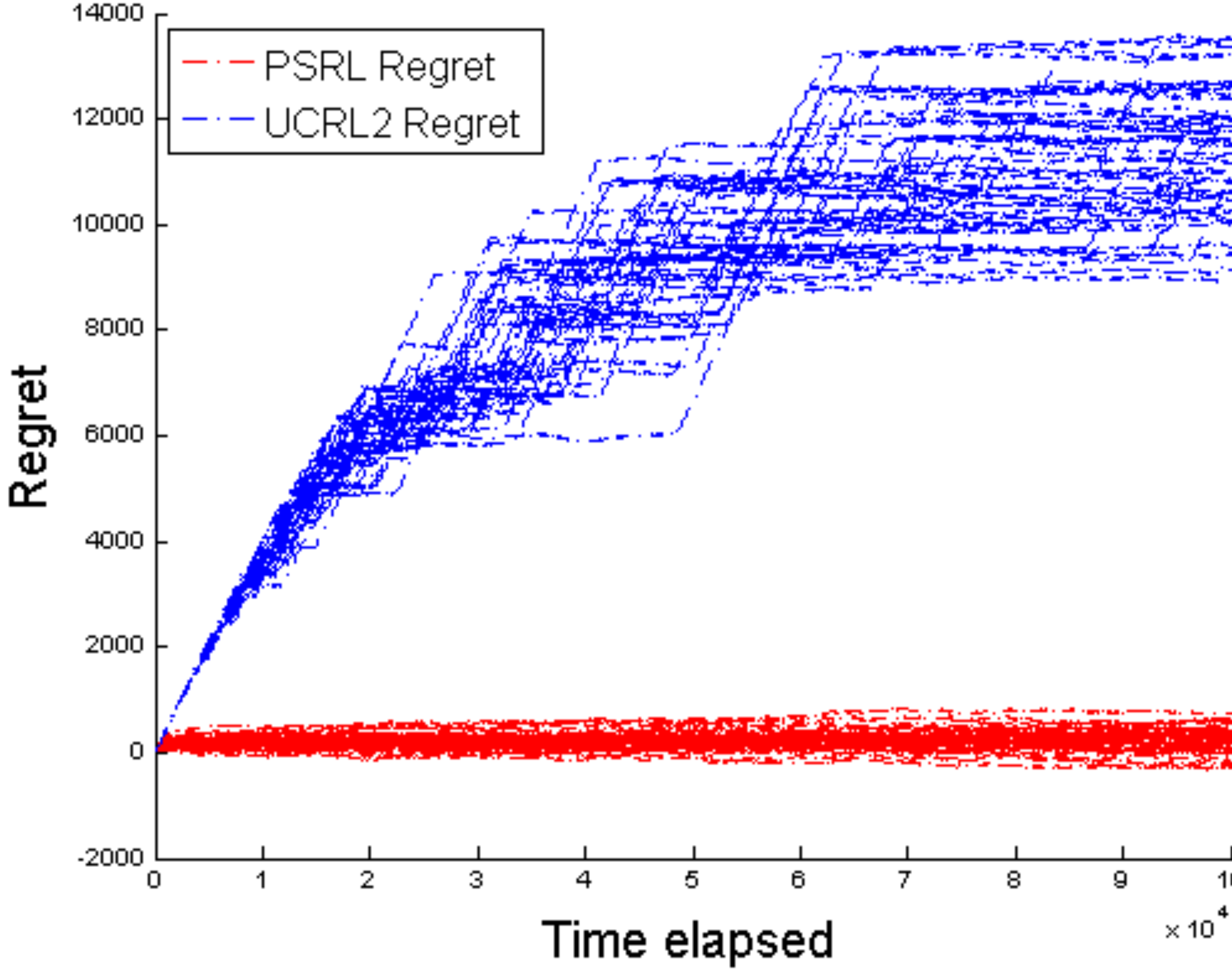
UCRL2

[Jaksch-Ortner-Auer, 2010]

- To select π^ℓ
  - At each *(s,a)*, construct confidence sets for
    - transition probability vector
    - mean reward
  - Solve optimistic MDP $\max_{\pi,\hat{\mathcal{M}}} \mathbb{E}_{\pi,\hat{\mathcal{M}}}\left[\sum_{h=0}^{H-1} r_h\right]$
- Regret bound $\mathrm{Regret}(L) = \tilde{O}\left(HS\sqrt{AHL}\right)$
- "Near-optimal reinforcement learning"

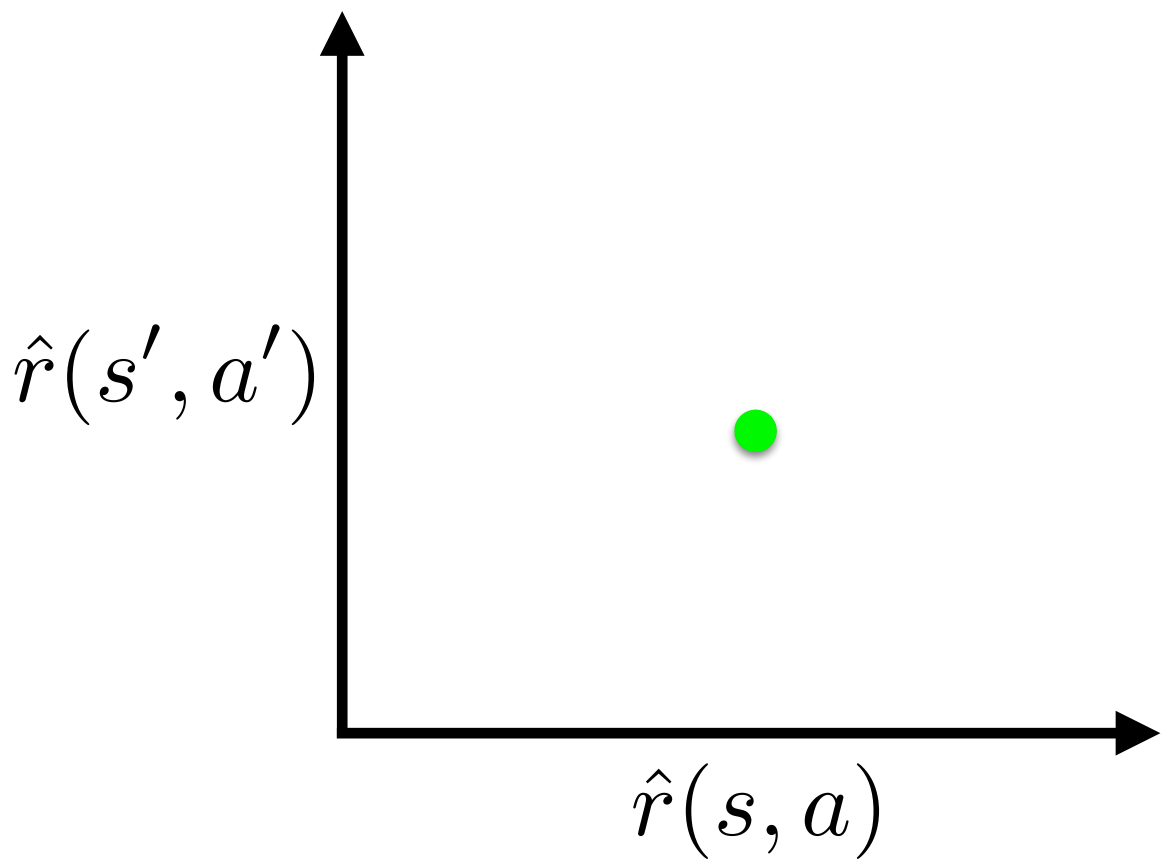
UCRL2 versus PSRL

[Osband-Russo-VanRoy, 2014]

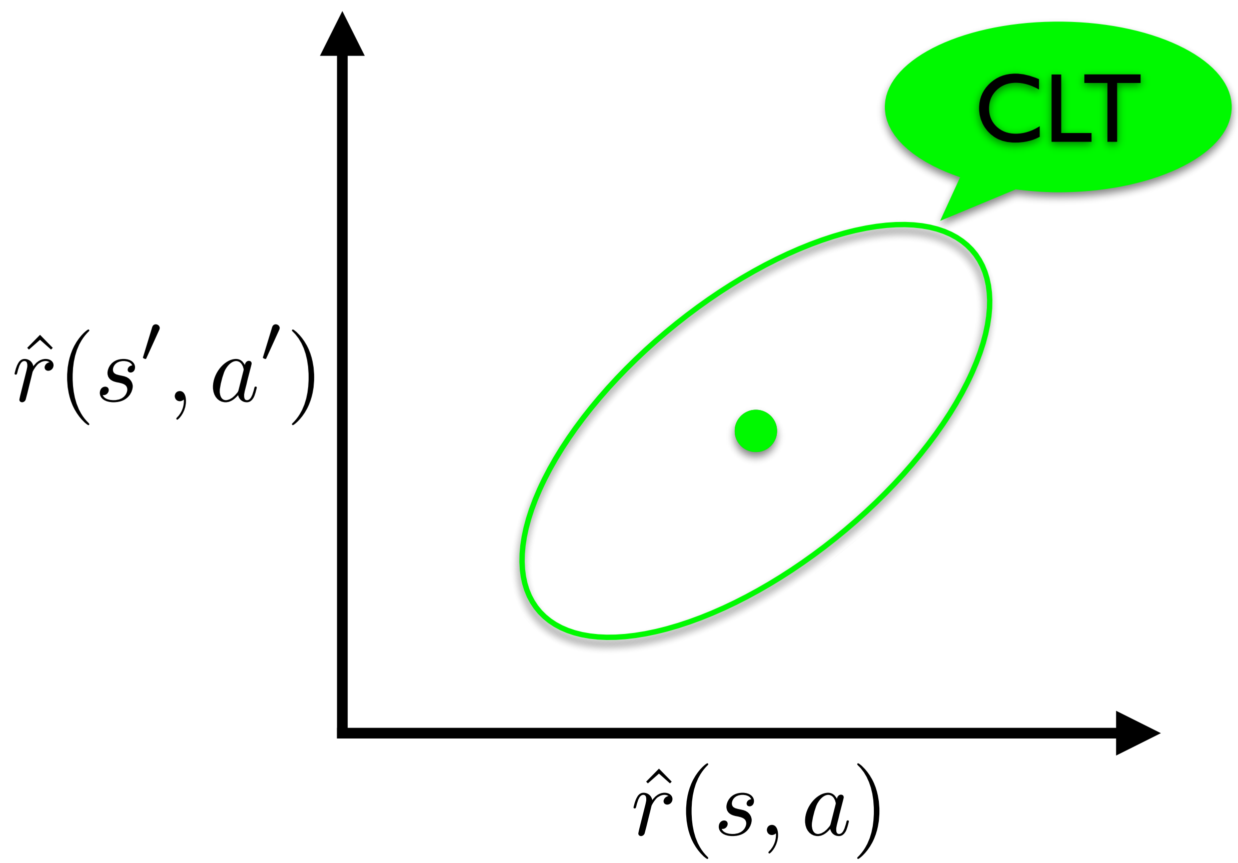


Why does UCRL2 perform so poorly?

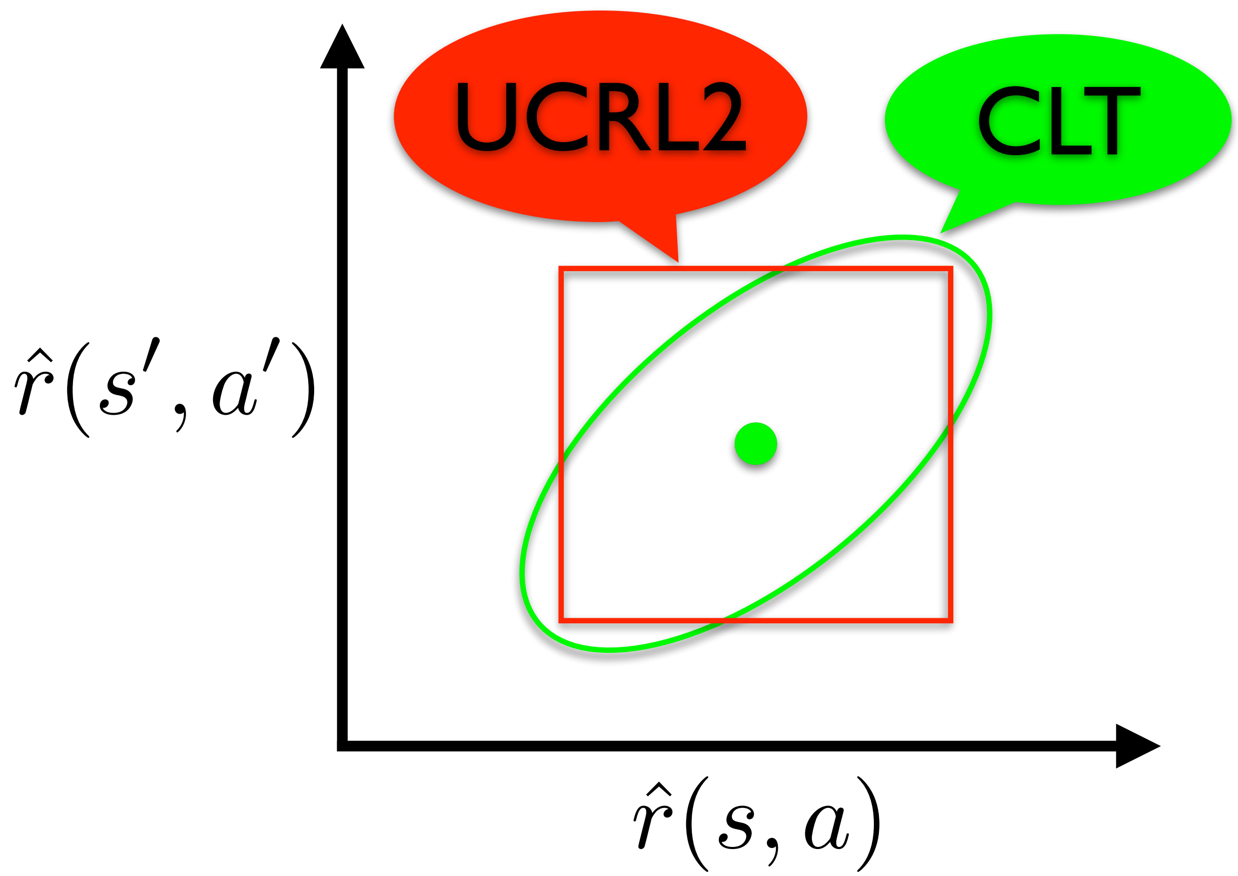
Why does UCRL2 perform so poorly?



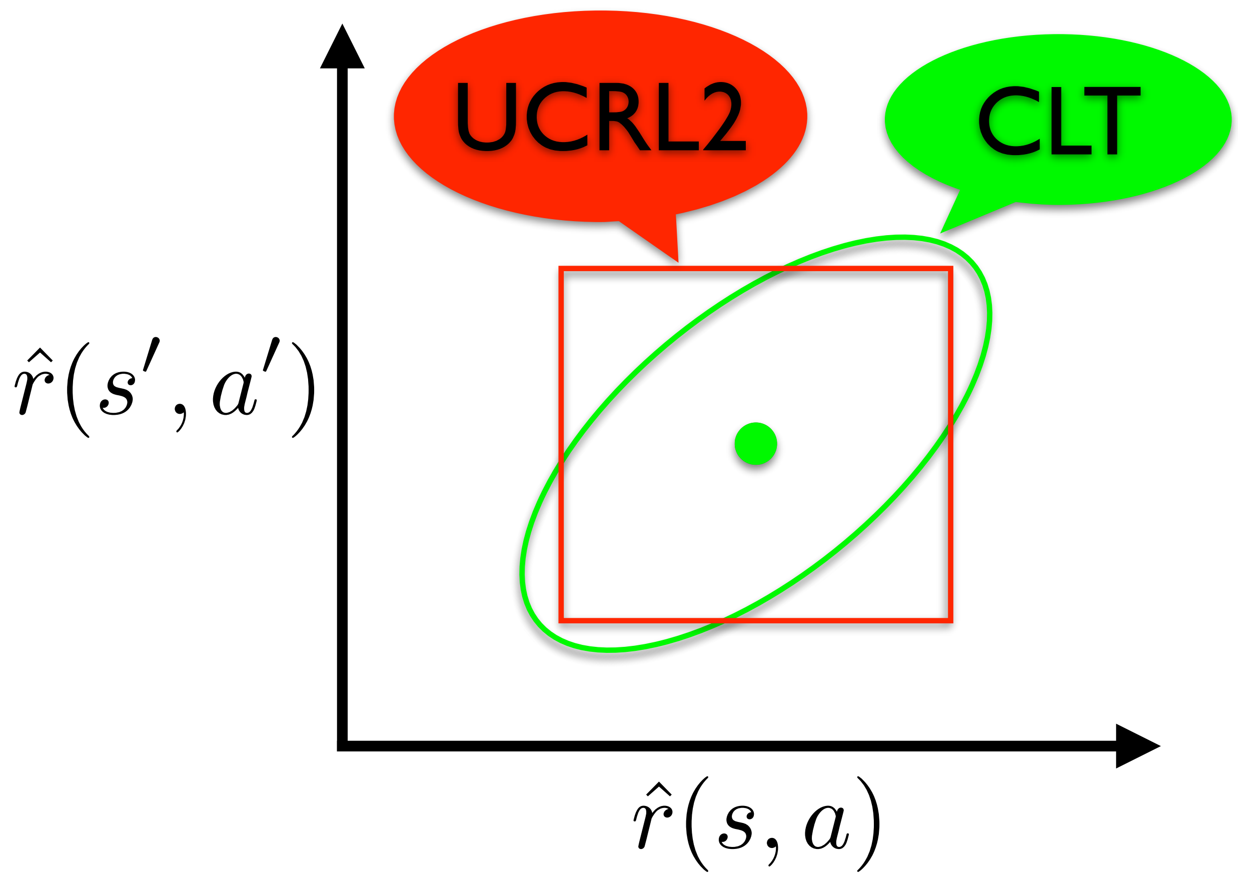
Why does UCRL2 perform so poorly?



Why does UCRL2 perform so poorly?

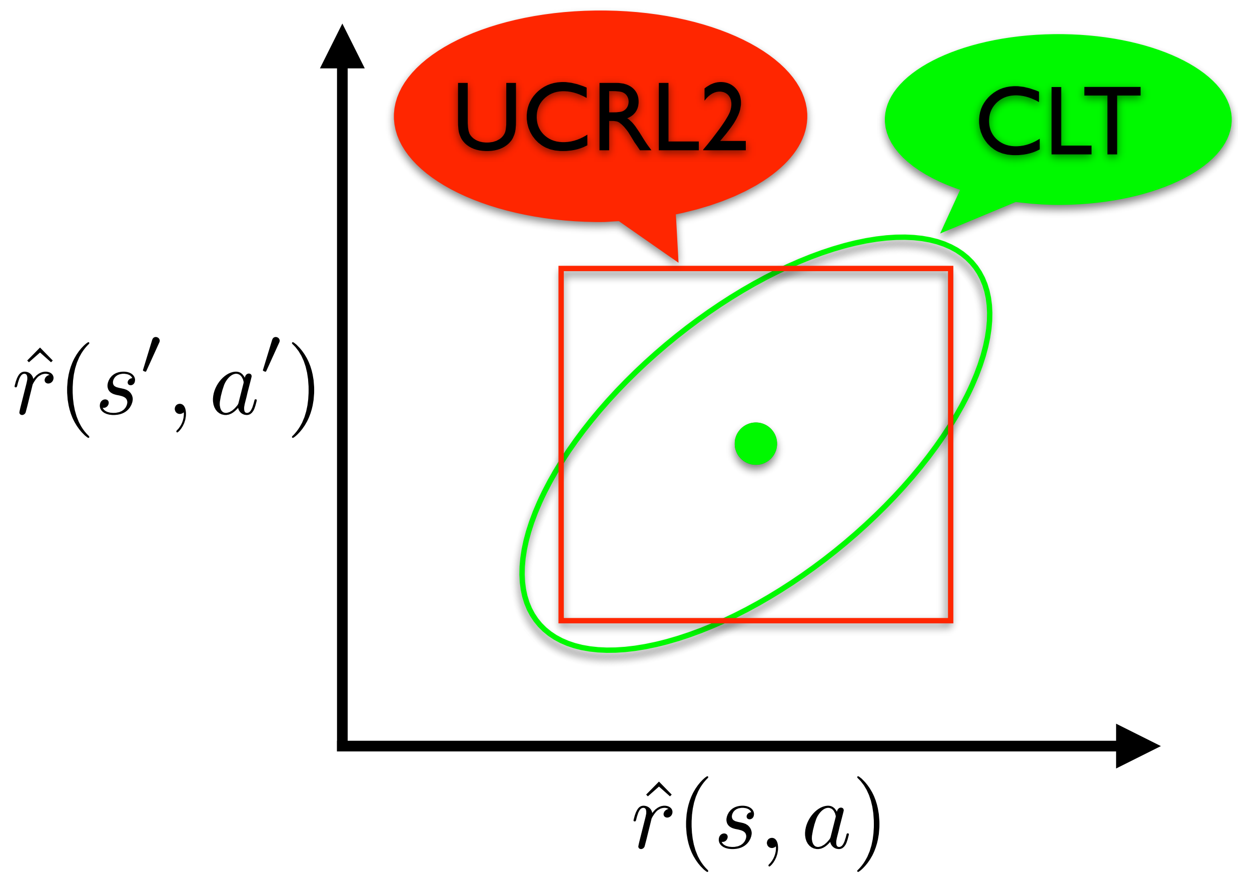


Why does UCRL2 perform so poorly?



UCRL2 decouples confidence sets

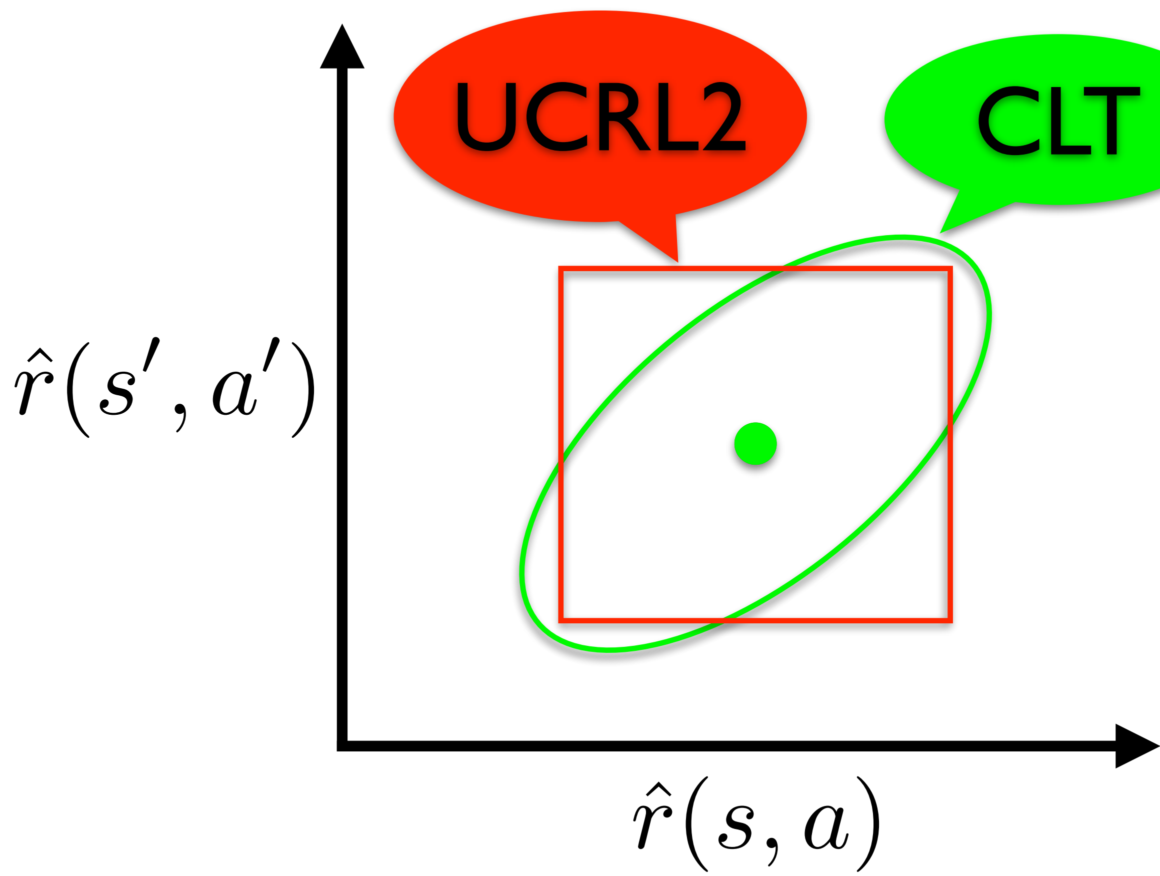
Why does UCRL2 perform so poorly?



UCRL2 decouples confidence sets

- Theory: "Thompson sampling approximates Bayes-UCB"

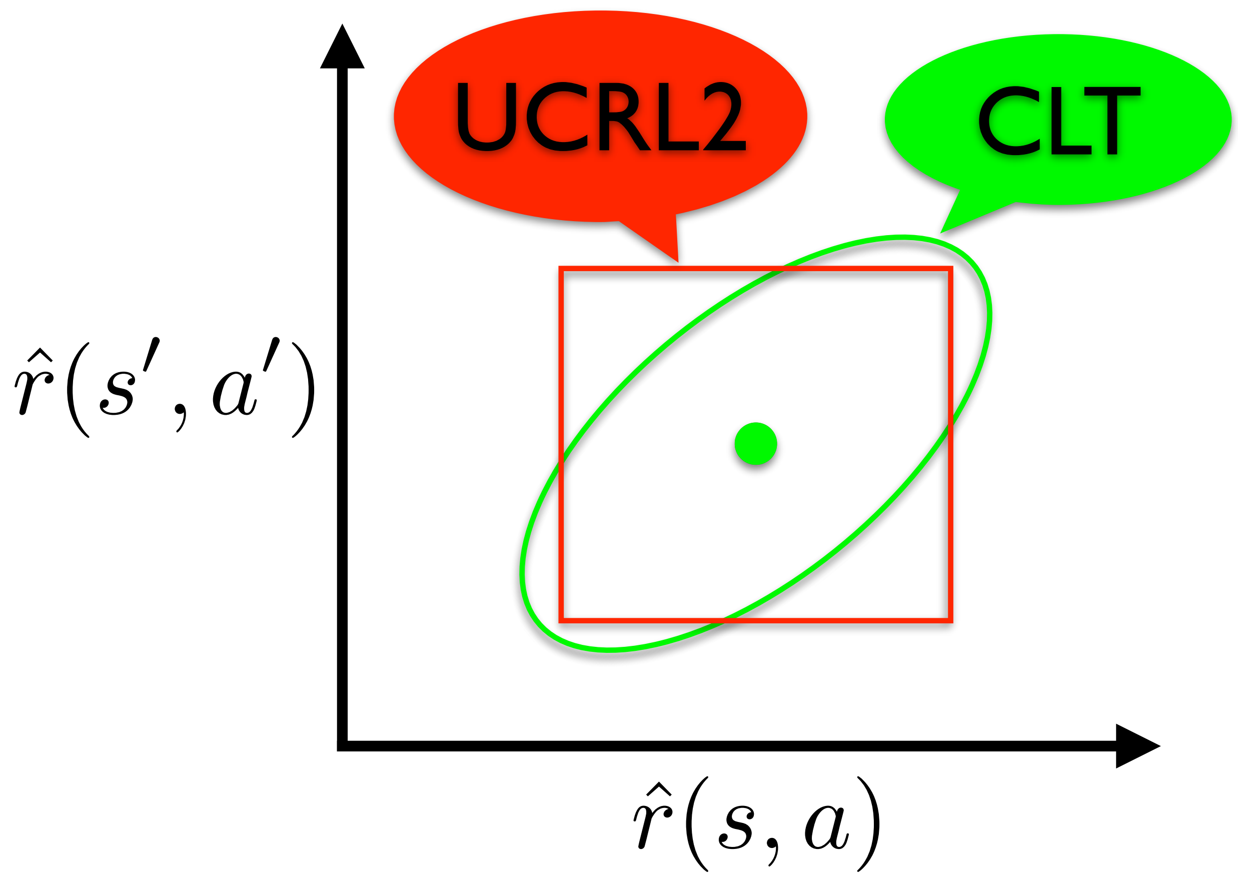
Why does UCRL2 perform so poorly?



UCRL2 decouples confidence sets

- Theory: "Thompson sampling approximates Bayes-UCB"
- "Bayes-UCRL2" should work much better than UCRL2

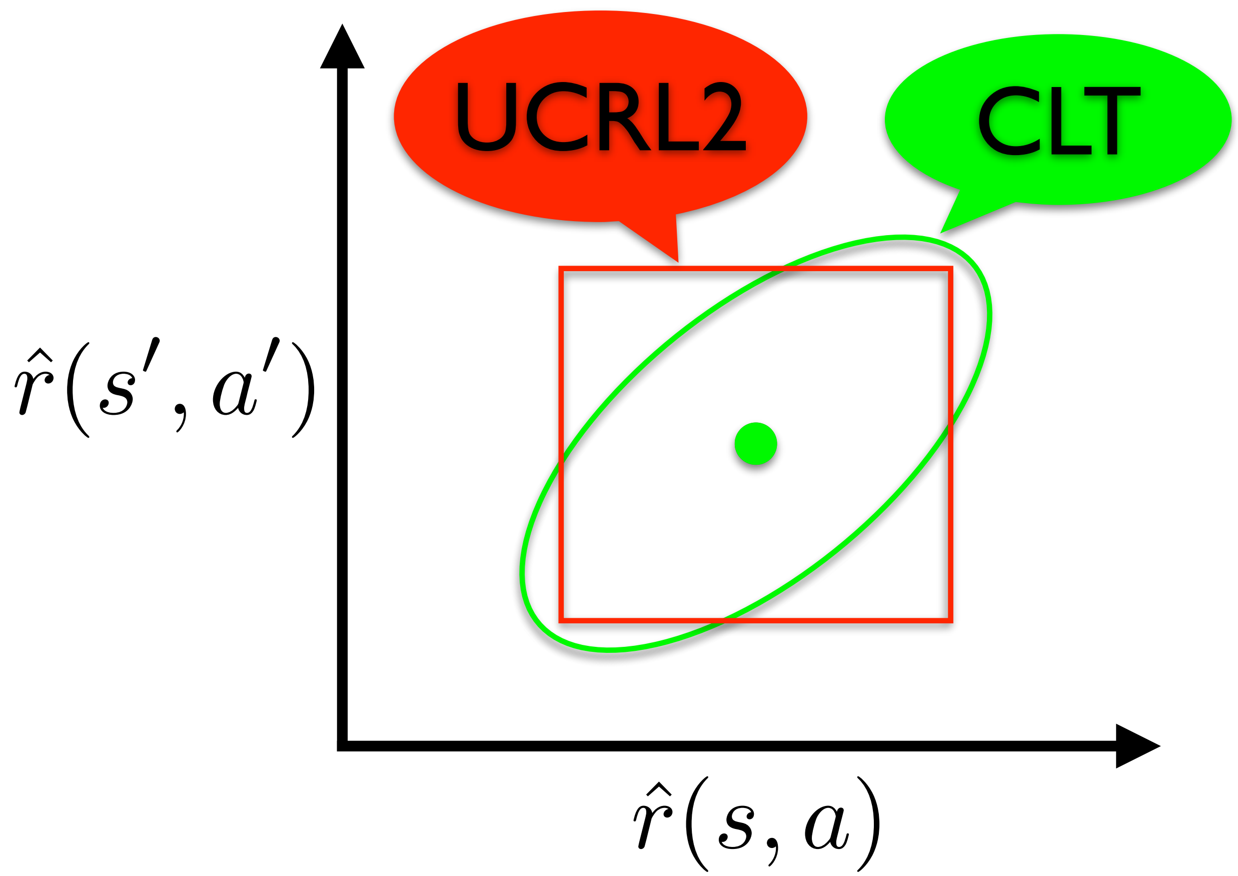
Why does UCRL2 perform so poorly?



UCRL2 decouples confidence sets

- Theory: "Thompson sampling approximates Bayes-UCB"
- "Bayes-UCRL2" should work much better than UCRL2
  - But this is computationally intractable

Why does UCRL2 perform so poorly?



UCRL2 decouples confidence sets

- Theory: "Thompson sampling approximates Bayes-UCB"
- "Bayes-UCRL2" should work much better than UCRL2
  - But this is computationally intractable
  - PSRL approximates this

PSRL

[Strens, 2000]

PSRL

[Strens, 2000]

- To select π^{ℓ}

PSRL

[Strens, 2000]

- To select π^ℓ
  - Sample a statistically plausible MDP $\hat{\mathcal{M}}$

PSRL

[Strens, 2000]

- To select π^ℓ
  - Sample a statistically plausible MDP $\hat{\mathcal{M}}$
  - Optimize sampled MDP $\max_{\pi} \mathbb{E}_{\pi,\hat{\mathcal{M}}} \left[\sum_{h=0}^{H-1} r_h \right]$

PSRL

[Strens, 2000]

start with uninformative prior

- To select π^{ℓ}
  - Sample a statistically plausible MDP $\hat{\mathcal{M}}$
  - Optimize sampled MDP $\max_{\pi} \mathbb{E}_{\pi,\hat{\mathcal{M}}} \left[\sum_{h=0}^{H-1} r_h \right]$

PSRL

[Strens, 2000]

start with uninformative prior

- To select π^ℓ
  - Sample a statistically plausible MDP $\hat{\mathcal{M}}$

sample from posterior

  - Optimize sampled MDP $\max_\pi \mathbb{E}_{\pi,\hat{\mathcal{M}}}\left[\sum_{h=0}^{H-1} r_h\right]$

PSRL

[Strens, 2000]

start with uninformative prior

- To select π^ℓ
  - Sample a statistically plausible MDP $\hat{\mathcal{M}}$

sample from posterior

  - Optimize sampled MDP $\max_{\pi} \mathbb{E}_{\pi,\hat{\mathcal{M}}}\left[\sum_{h=0}^{H-1} r_h\right]$

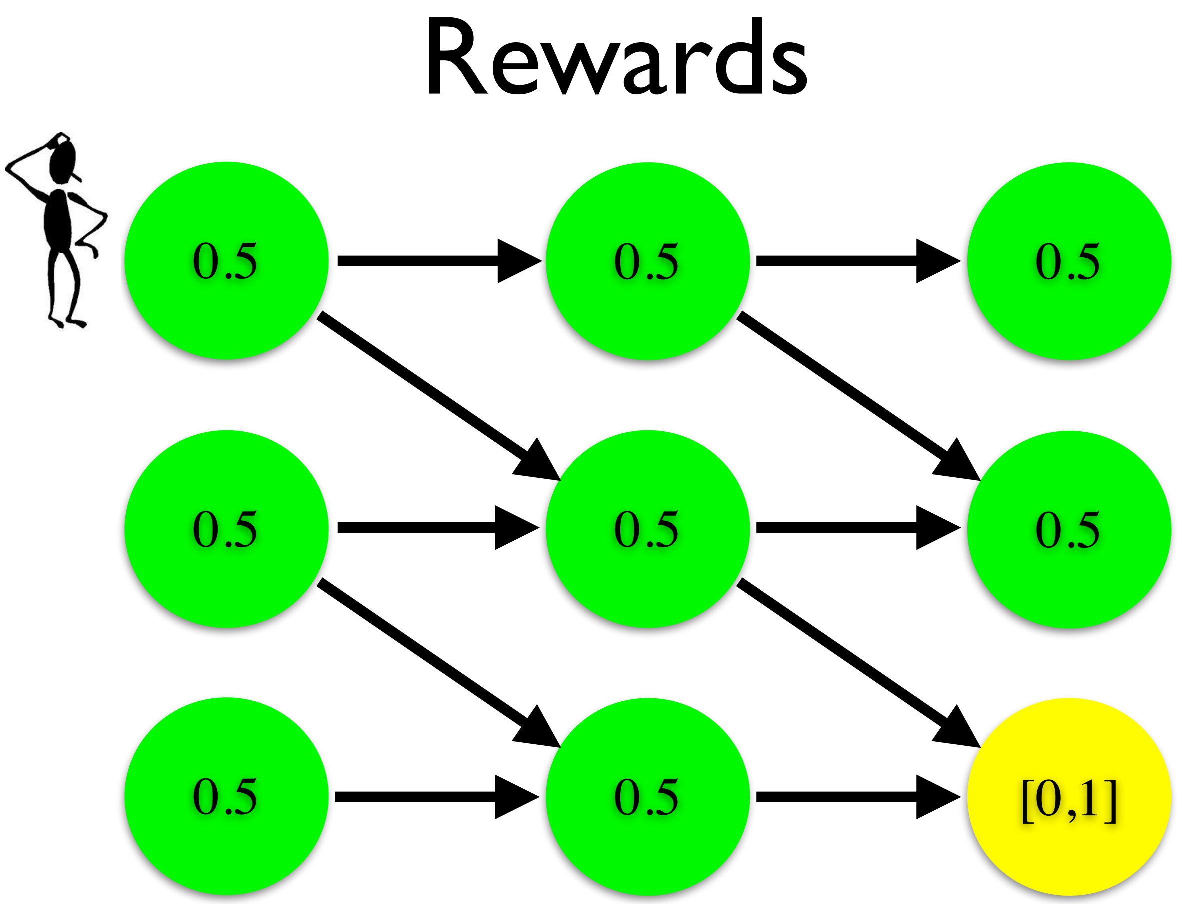
- Regret bound [Osband-Russo-VanRoy, 2014]

$$\mathbb{E}\left[\text{Regret}(L)\right] = \tilde{O}\left(HS\sqrt{AHL}\right)$$

How does PSRL plan to learn?

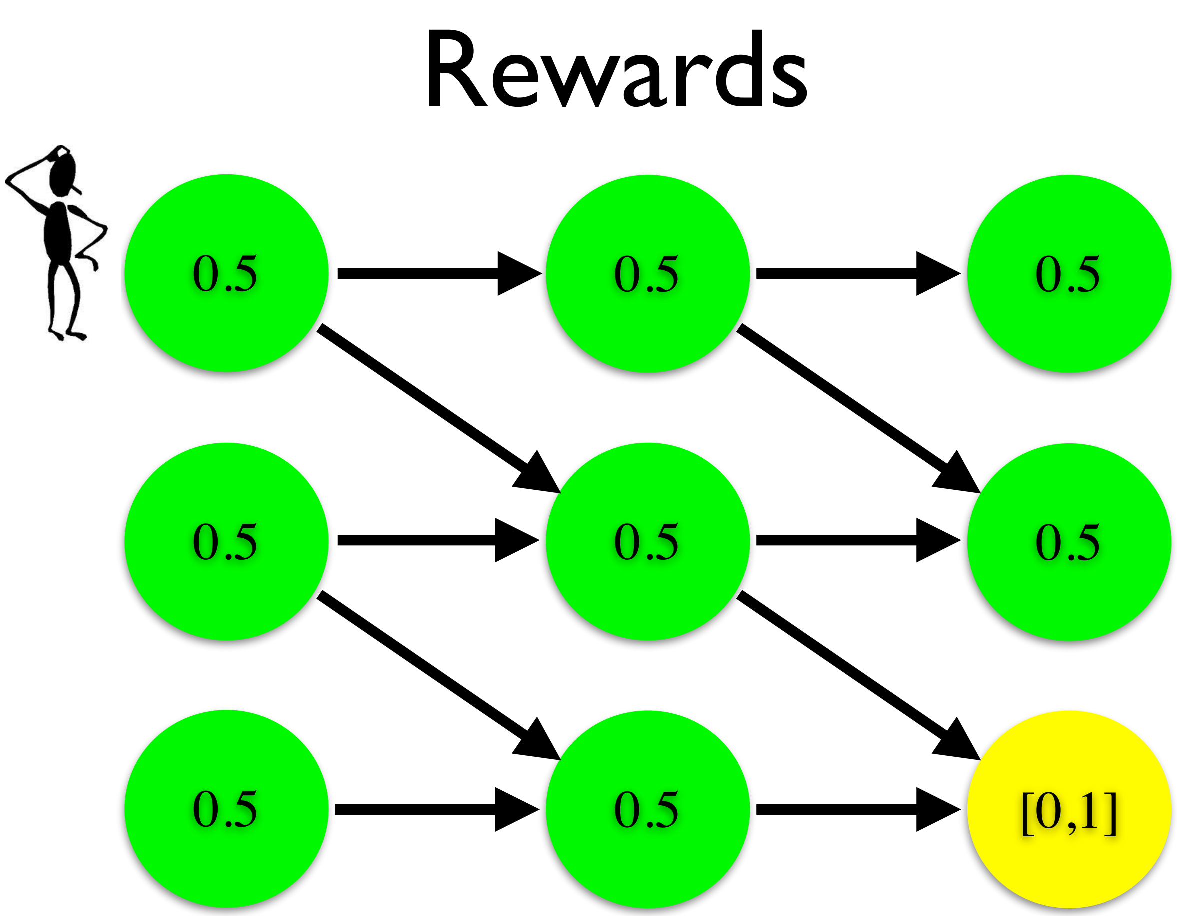
How does PSRL plan to learn?

- Simple case: deterministic with known transitions



How does PSRL plan to learn?

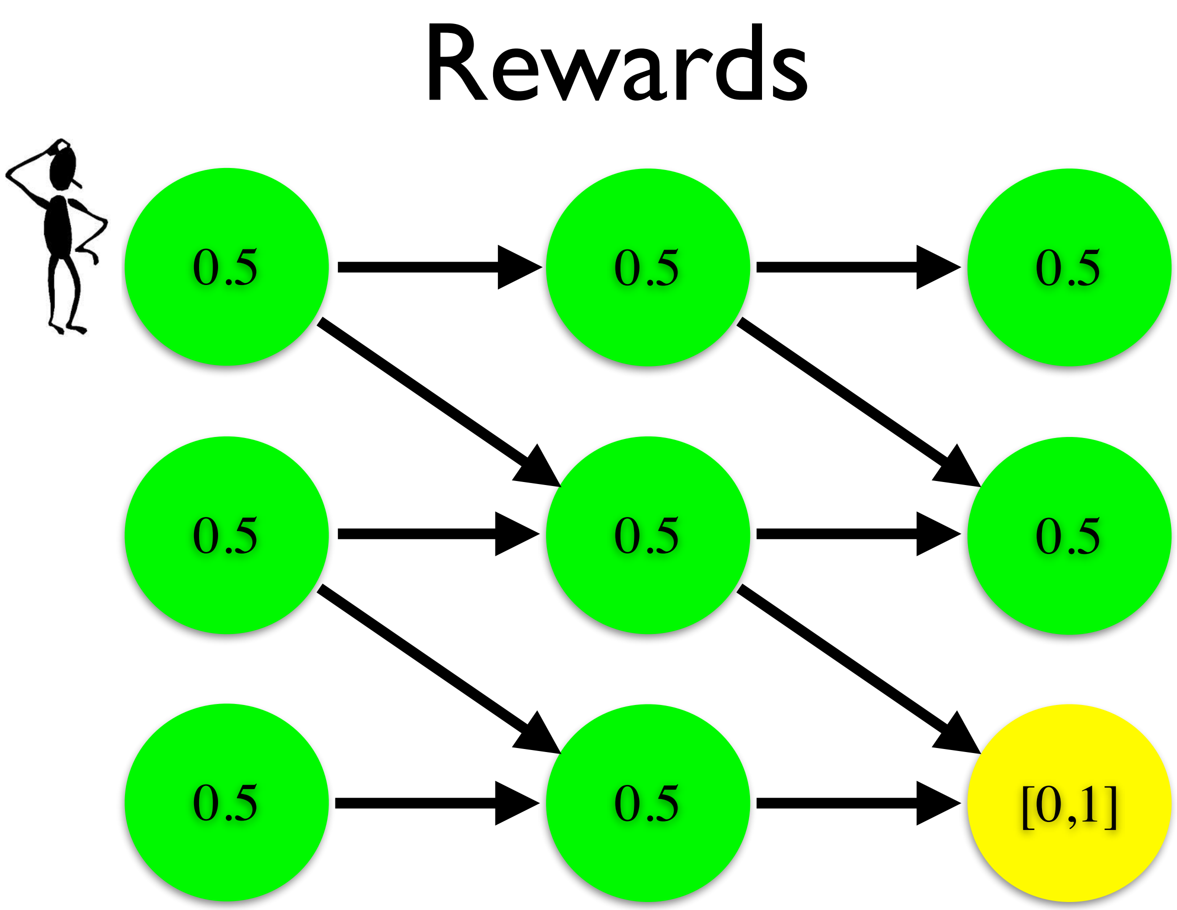
- Simple case: deterministic with known transitions
- How does UCRL2 plan to learn?

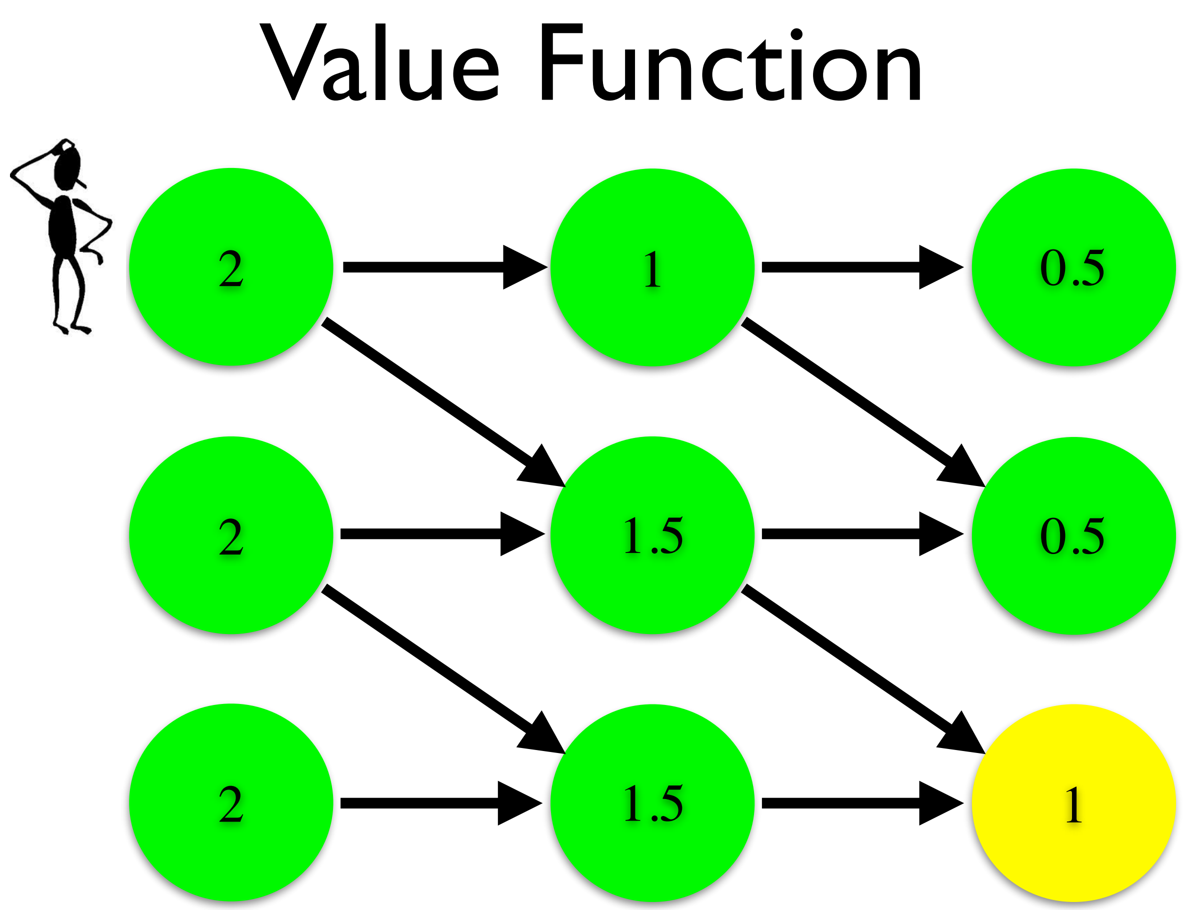


How does PSRL plan to learn?

- Simple case: deterministic with known transitions

- How does UCRL2 plan to learn?



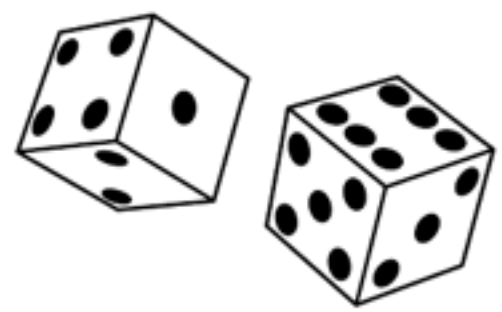


How does PSRL plan to learn?

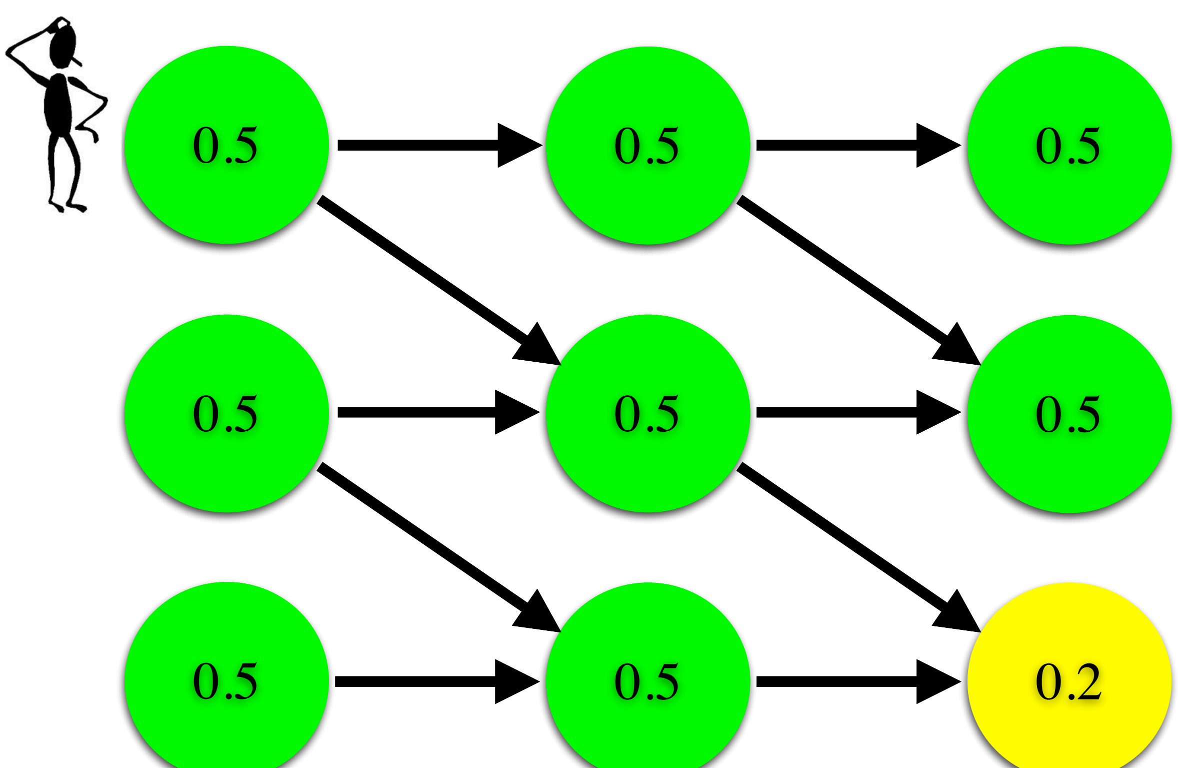
- Simple case: deterministic with known transitions

- How does UCRL2 plan to learn?

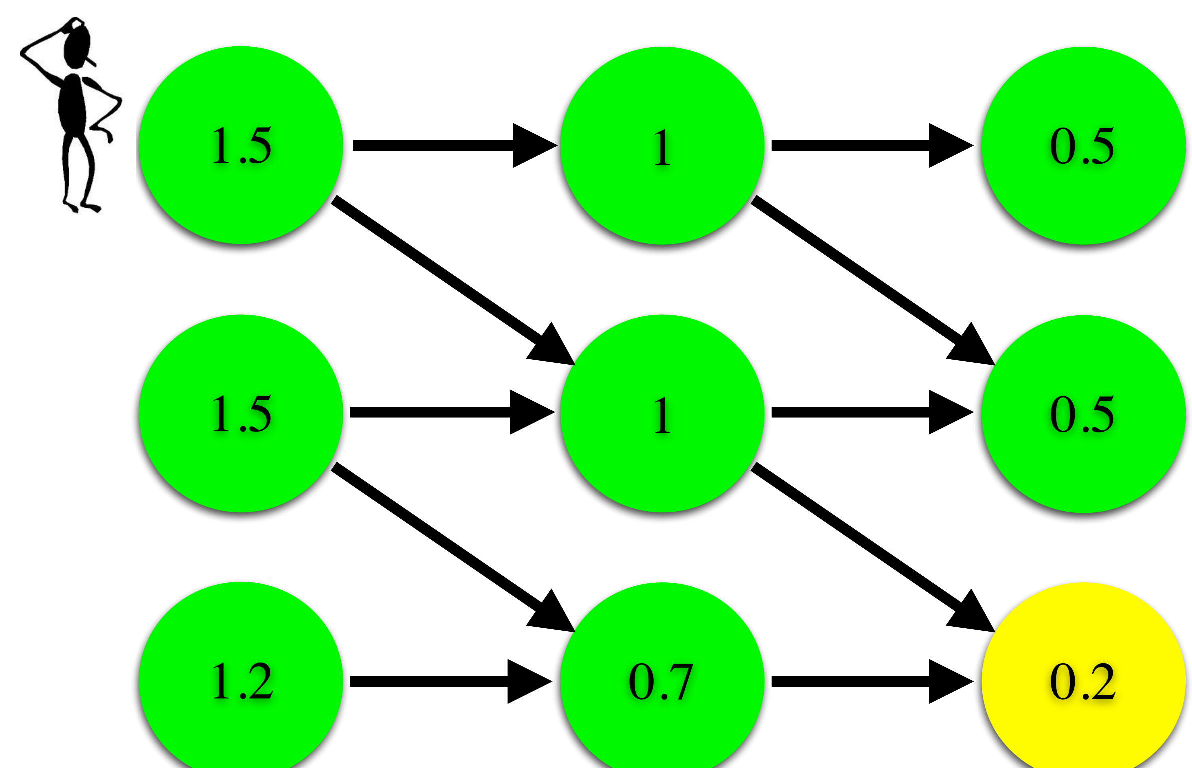
- How does PSRL?



Rewards

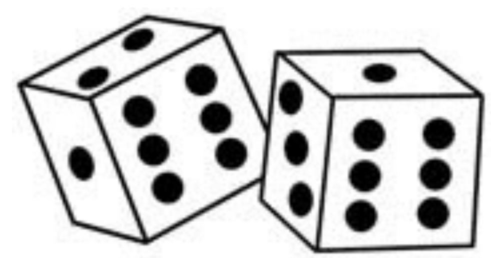


Value Function

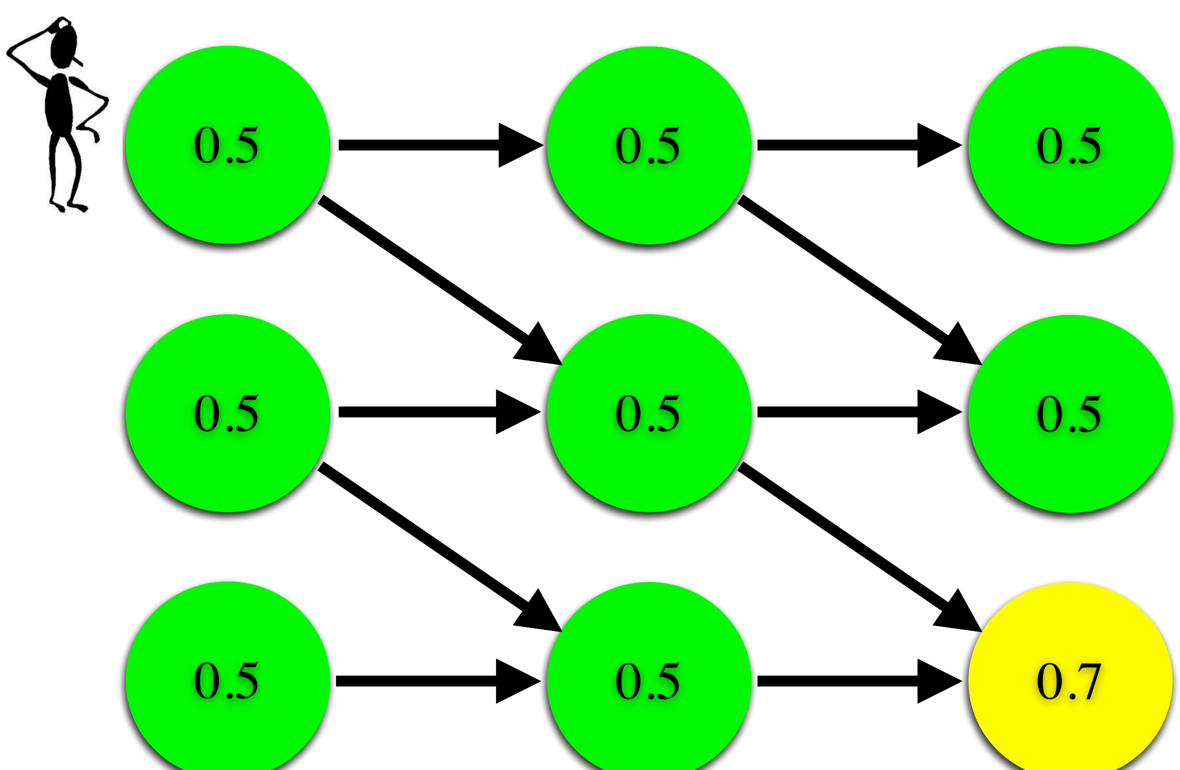


How does PSRL plan to learn?

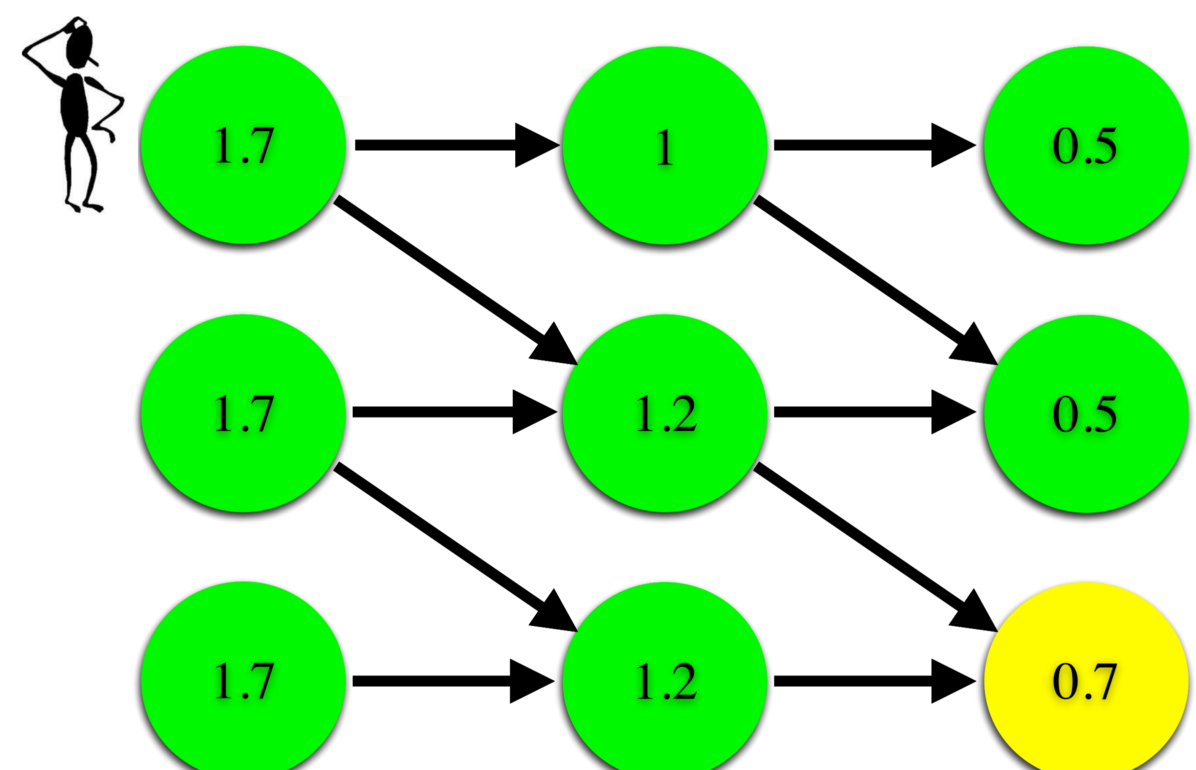
- Simple case: deterministic with known transitions
- How does UCRL2 plan to learn?
- How does PSRL?



Rewards



Value Function



Beyond *Tabula Rasa* RL

Beyond *Tabula Rasa* RL

- Curse of dimensionality

Beyond *Tabula Rasa* RL

- Curse of dimensionality
- Need to generalize
  - Parameterized policies
  - Parameterized value functions

Beyond *Tabula Rasa* RL

- Curse of dimensionality
- Need to generalize
  - Parameterized policies
  - Parameterized value functions
- Simple case: linear combination of features

$$\tilde{Q}_h^{\theta_h}(s, a) = \sum_{k=1}^{K} \theta_{hk} \phi_{hk}(s, a)$$

Value Function Randomization

Value Function Randomization

- To select π^ℓ
  - Sample statistically plausible parameters θ
  - Use greedy policy

Value Function Randomization

- To select π^{ℓ}
  - Sample statistically plausible parameters θ
  - Use greedy policy
- How to sample?

Value Function Randomization

- To select π^ℓ
  - Sample statistically plausible parameters θ
  - Use greedy policy

- How to sample?
  - **Randomized least-squares value iteration**

RLSVI

RLSVI

- Consider least-squares value iteration

$$\min_{\hat{\theta}_h} \left(\frac{1}{\sigma^2} \sum_{\ell=1}^{L} \left(\tilde{Q}_h^{\hat{\theta}_h}(s_h^\ell, a_h^\ell) - \left(r_h^\ell + \max_\alpha \tilde{Q}_{h+1}^{\hat{\theta}_{h+1}}(s_{h+1}^\ell, \alpha) \right) \right)^2 + \lambda \|\hat{\theta}_h\|_2^2 \right)$$

RLSVI

- Consider least-squares value iteration

$$\hat{\theta}_h \leftarrow \frac{1}{\sigma^2}\left(\frac{1}{\sigma^2}A^\top A + \lambda I\right)^{-1} A^\top b$$

RLSVI

- Consider least-squares value iteration

$$\hat{\theta}_h \leftarrow \frac{1}{\sigma^2}\left(\frac{1}{\sigma^2}A^\top A + \lambda I\right)^{-1} A^\top b$$

$$\Sigma_h \leftarrow \left(\frac{1}{\sigma^2}A^\top A + \lambda I\right)^{-1}$$

RLSVI

- Consider least-squares value iteration

$$\hat{\theta}_h \leftarrow \frac{1}{\sigma^2}\left(\frac{1}{\sigma^2}A^\top A + \lambda I\right)^{-1} A^\top b$$

$$\Sigma_h \leftarrow \left(\frac{1}{\sigma^2}A^\top A + \lambda I\right)^{-1}$$

- Randomized least-squares value iteration

RLSVI

- Consider least-squares value iteration

$$\hat{\theta}_h \leftarrow \frac{1}{\sigma^2}\left(\frac{1}{\sigma^2}A^\top A + \lambda I\right)^{-1} A^\top b$$

$$\Sigma_h \leftarrow \left(\frac{1}{\sigma^2}A^\top A + \lambda I\right)^{-1}$$

- Randomized least-squares value iteration

$$\overline{\theta}_h \leftarrow \frac{1}{\sigma^2}\left(\frac{1}{\sigma^2}A^\top A + \lambda I\right)^{-1} A^\top b$$

RLSVI

- Consider least-squares value iteration

$$\hat{\theta}_h \leftarrow \frac{1}{\sigma^2}\left(\frac{1}{\sigma^2}A^\top A + \lambda I\right)^{-1} A^\top b$$

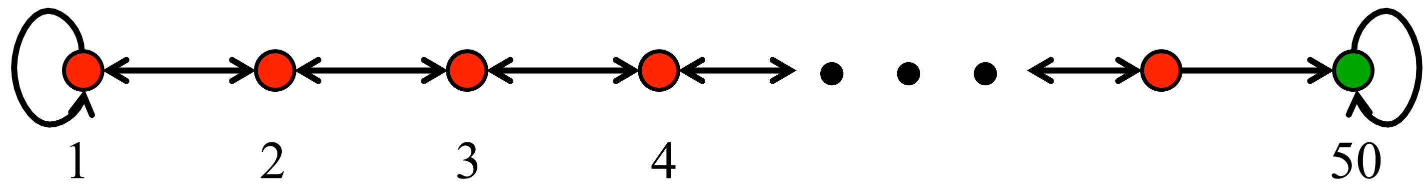
$$\Sigma_h \leftarrow \left(\frac{1}{\sigma^2}A^\top A + \lambda I\right)^{-1}$$

- Randomized least-squares value iteration

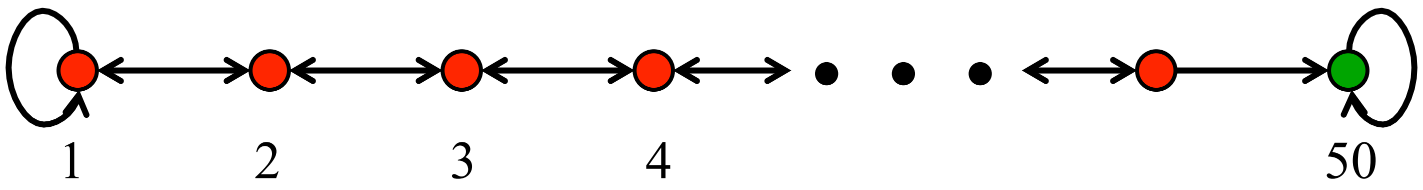
$$\overline{\theta}_h \leftarrow \frac{1}{\sigma^2}\left(\frac{1}{\sigma^2}A^\top A + \lambda I\right)^{-1} A^\top b$$

$$\hat{\theta}_h \sim \mathrm{N}(\overline{\theta}_h, \Sigma_h)$$

A Simple Example

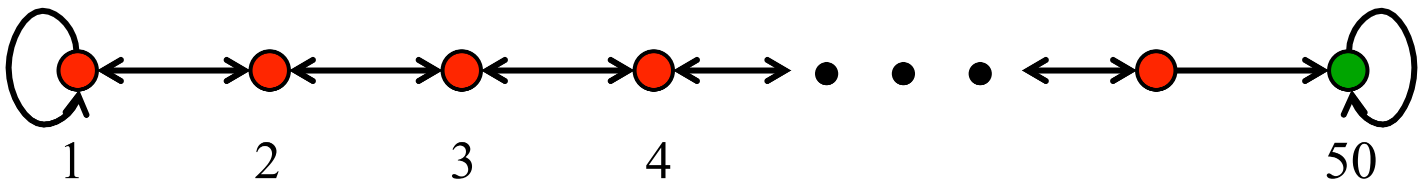


A Simple Example



- Deterministic MDP
  - Start at state 1
  - Actions: left or right
  - Horizon = 50 periods
  - Receive reward 1 only if at state 50

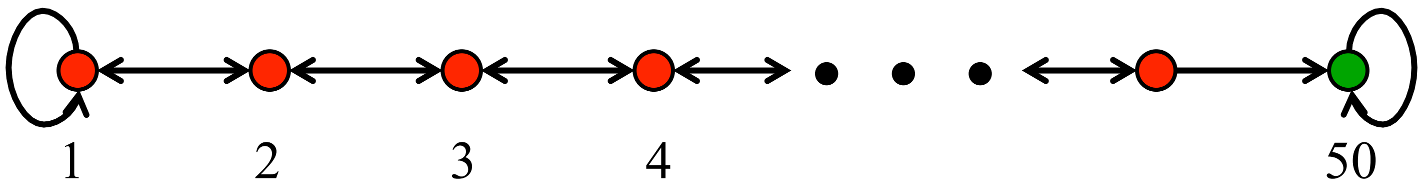
A Simple Example



- Deterministic MDP
  - Start at state 1
  - Actions: left or right
  - Horizon = 50 periods
  - Receive reward 1 only if at state 50

- What is the optimal strategy?

A Simple Example

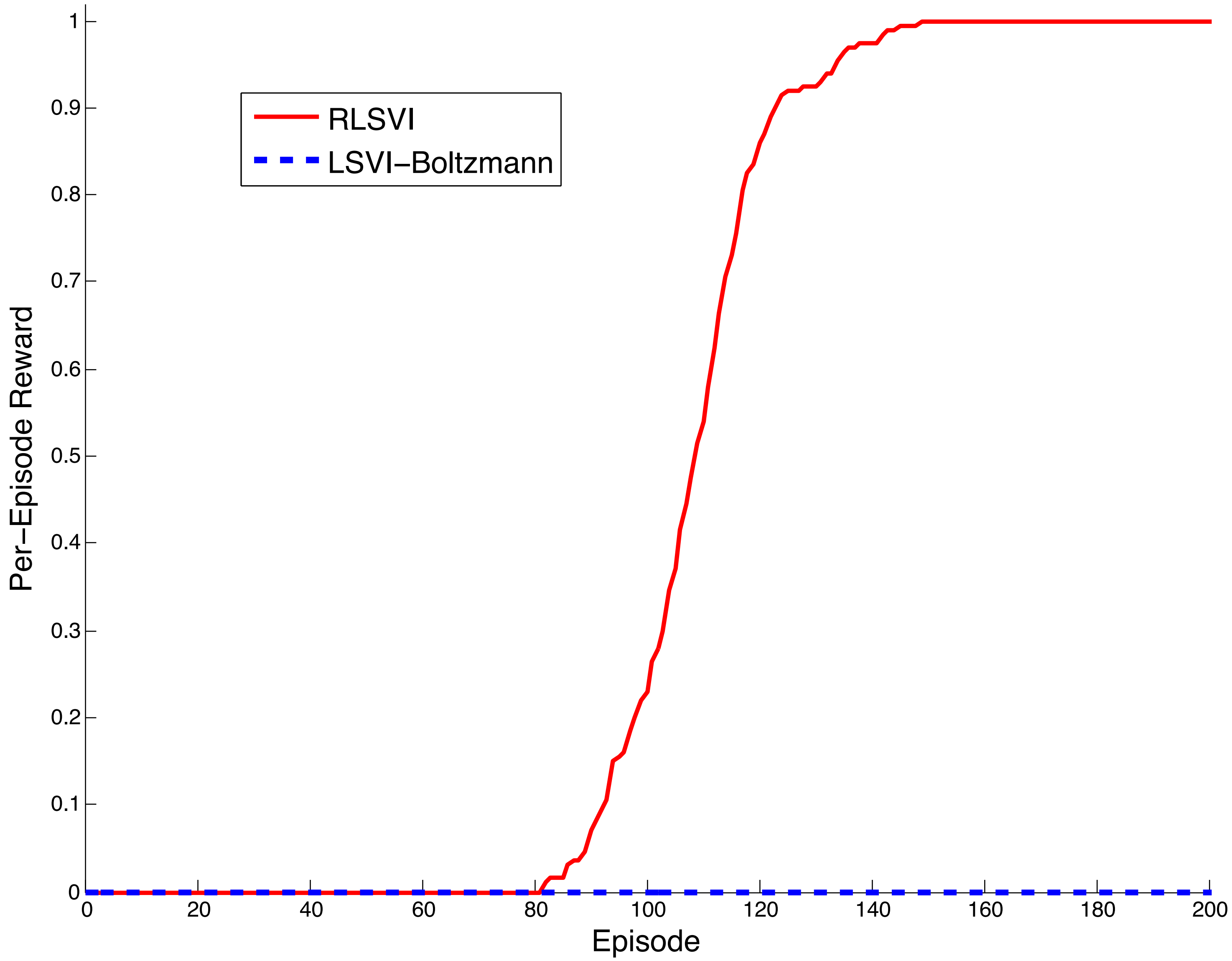


- Deterministic MDP
  - Start at state 1
  - Actions: left or right
  - Horizon = 50 periods
  - Receive reward 1 only if at state 50

- What is the optimal strategy?

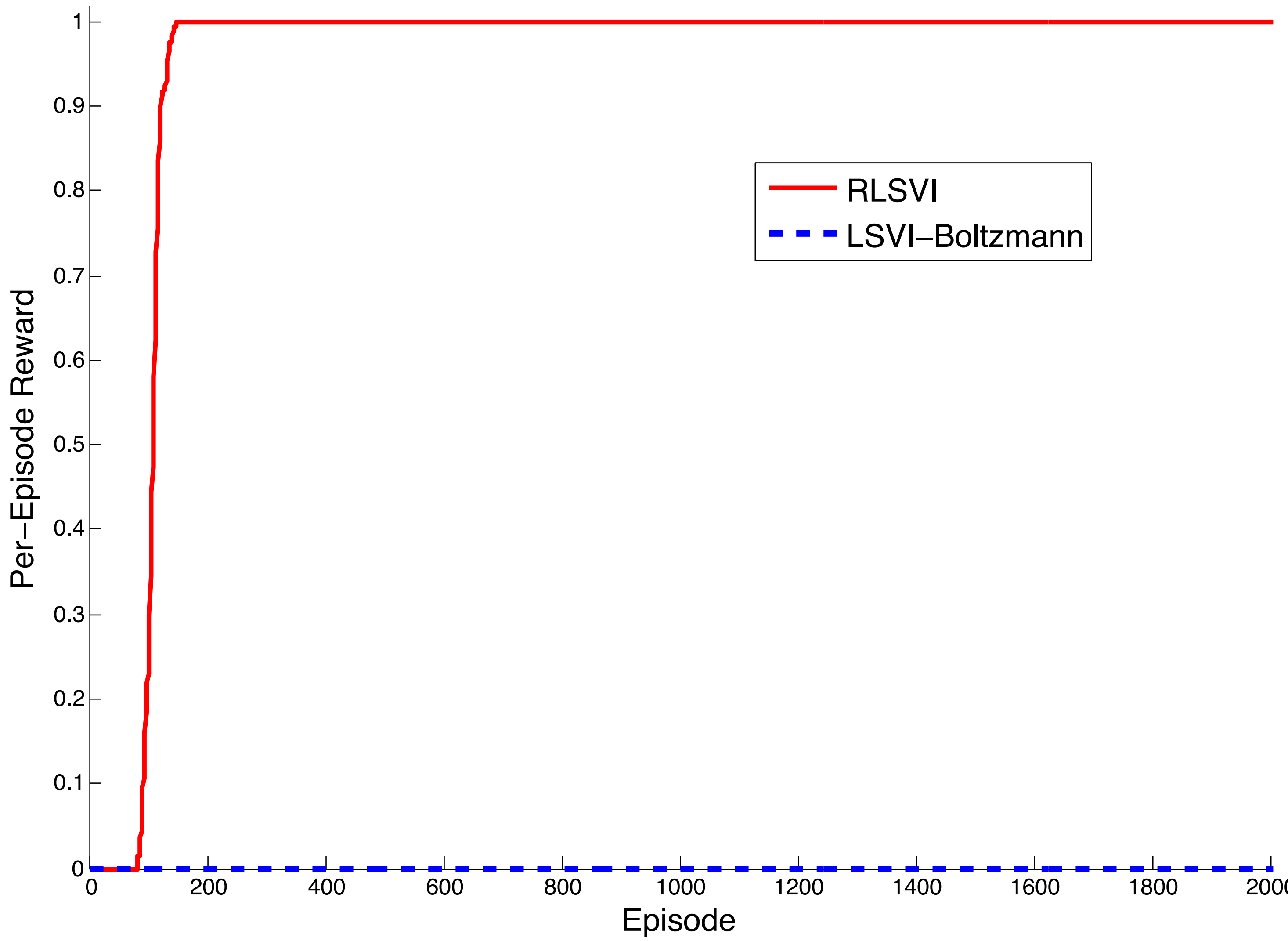
- $\tilde{Q}_h^{\theta_h}$ spans a random affine subspace that contains Q_h^* and constant functions

Boltzmann versus VF Randomization



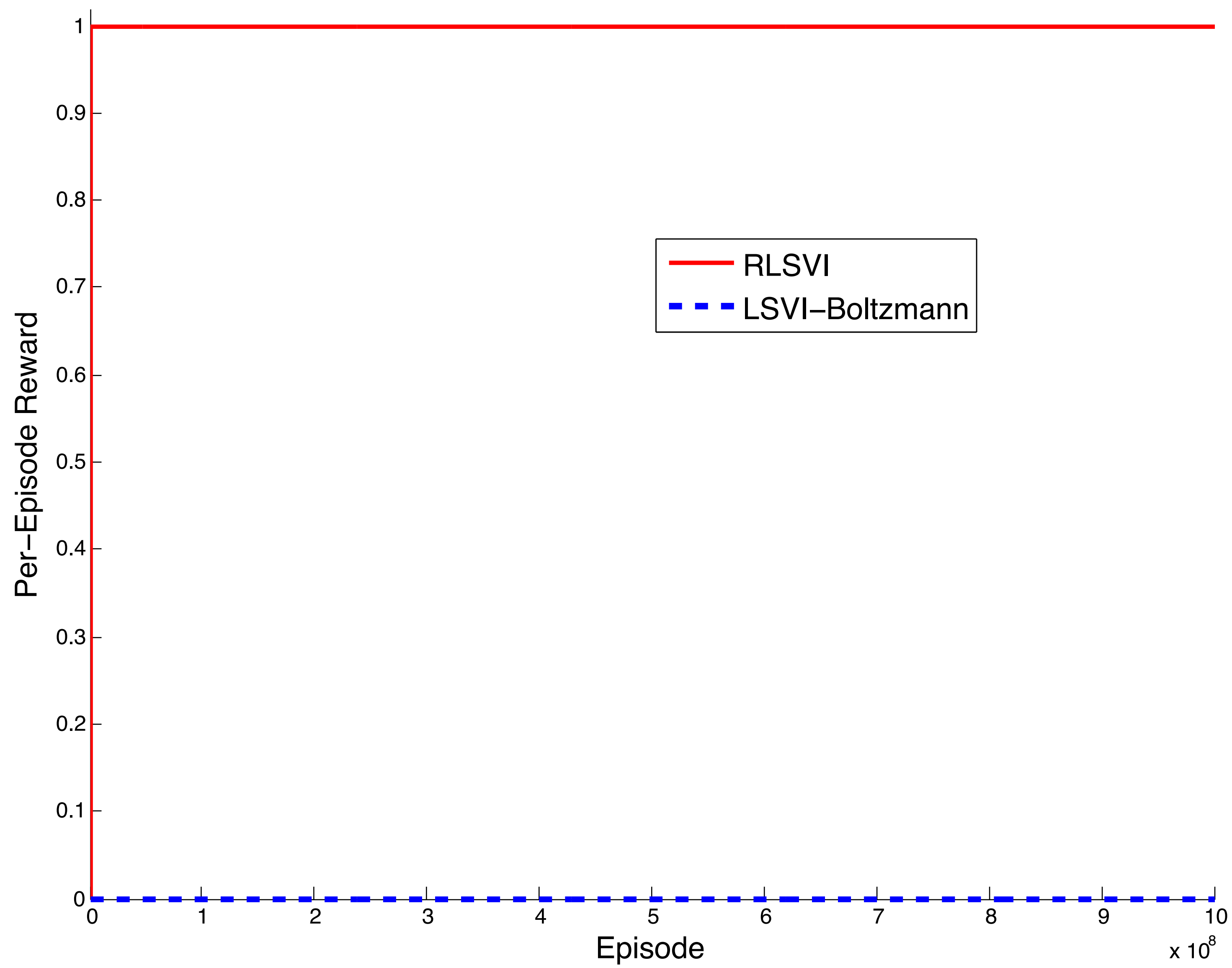
$K = 10$ features

Boltzmann versus VF Randomization



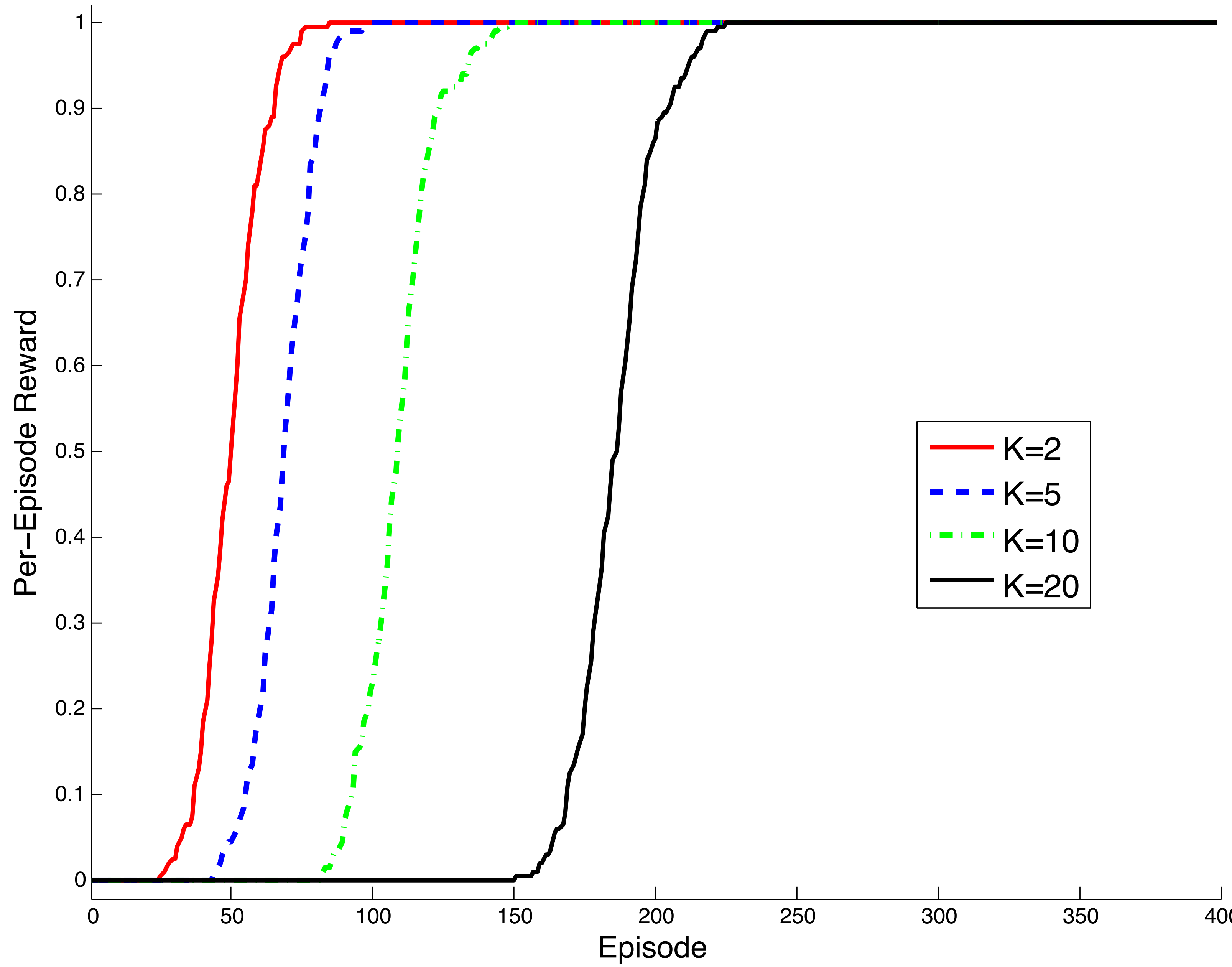
$K = 10$ features

Boltzmann versus VF Randomization

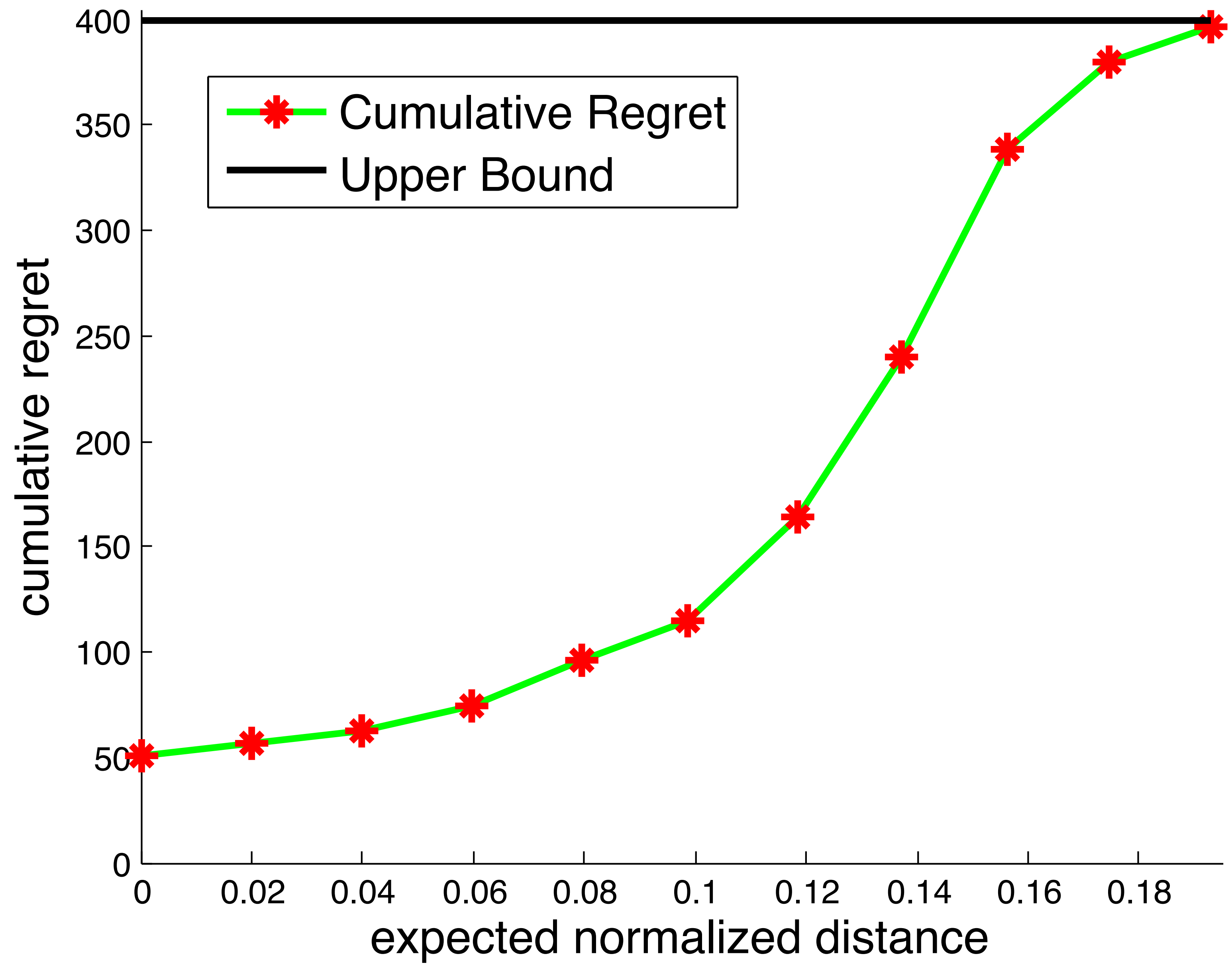


$K = 10$ features

Varying the Number of Features



Agnostic Learning



$K = 11$ features

Understanding RLSVI: *Tabula Rasa* Case

Understanding RLSVI: *Tabula Rasa* Case

$$\text{RLSVI} \approx \text{PSRL}$$

Understanding RLSVI: *Tabula Rasa* Case

RLSVI $\approx$ PSRL

- Assume unknown deterministic rewards $r_h(x, a)$

Understanding RLSVI: *Tabula Rasa* Case

RLSVI $\approx$ PSRL

- Assume unknown deterministic rewards $r_h(x, a)$
- PSRL value computation

$$\hat{Q}_h(s, a) \leftarrow \tilde{r}_h(s, a) + \sum_{s'} \tilde{p}_{ss'}(a) \max_{\alpha} \hat{Q}_{h+1}(s', \alpha)$$

Understanding RLSVI: *Tabula Rasa* Case

RLSVI $\approx$ PSRL

- Assume unknown deterministic rewards $r_h(x, a)$
- PSRL value computation

$$\hat{Q}_h(s,a) \leftarrow \overline{r}_h(s,a) + \sum_{s'} \overline{p}_{ss'}(a) \max_{\alpha} \hat{Q}_{h+1}(s', \alpha) + \text{noise}$$

Understanding RLSVI: *Tabula Rasa* Case

RLSVI $\approx$ PSRL

- Assume unknown deterministic rewards $r_h(x, a)$
- PSRL value computation

$$\hat{Q}_h(s, a) \leftarrow \overline{r}_h(s, a) + \sum_{s'} \overline{p}_{ss'}(a) \max_{\alpha} \hat{Q}_{h+1}(s', \alpha) + \text{noise}$$

- RLSVI value computation with $\lambda = 0$

$$\hat{Q}_h(s, a) \leftarrow \frac{1}{n_h(s, a)} \sum_{\ell} \left(r_h^\ell + \max_{\alpha} \hat{Q}_{h+1}(s_{h+1}^\ell, \alpha) \right) + \text{noise}'$$

Understanding RLSVI: *Tabula Rasa* Case

RLSVI $\approx$ PSRL

- Assume unknown deterministic rewards $r_h(x, a)$
- PSRL value computation

$$\hat{Q}_h(s,a) \leftarrow \overline{r}_h(s,a) + \sum_{s'} \overline{p}_{ss'}(a) \max_{\alpha} \hat{Q}_{h+1}(s', \alpha) + \text{noise}$$

- RLSVI value computation with $\lambda = 0$

$$\hat{Q}_h(s,a) \leftarrow \overline{r}'_h(s,a) + \sum_{s'} \overline{p}'_{ss'}(a) \max_{\alpha} \hat{Q}_{h+1}(s', \alpha) + \text{noise}'$$

Understanding RLSVI: *Tabula Rasa* Case

RLSVI $\approx$ PSRL

- Assume unknown deterministic rewards $r_h(x, a)$
- PSRL value computation

$$\hat{Q}_h(s,a) \leftarrow \overline{r}_h(s,a) + \sum_{s'} \overline{p}_{ss'}(a) \max_{\alpha} \hat{Q}_{h+1}(s',\alpha) + \text{noise}$$

- RLSVI value computation with $\lambda = 0$

$$\hat{Q}_h(s,a) \leftarrow \overline{r}'_h(s,a) + \sum_{s'} \overline{p}'_{ss'}(a) \max_{\alpha} \hat{Q}_{h+1}(s',\alpha) + \text{noise}'$$

- For PSRL with uniform prior

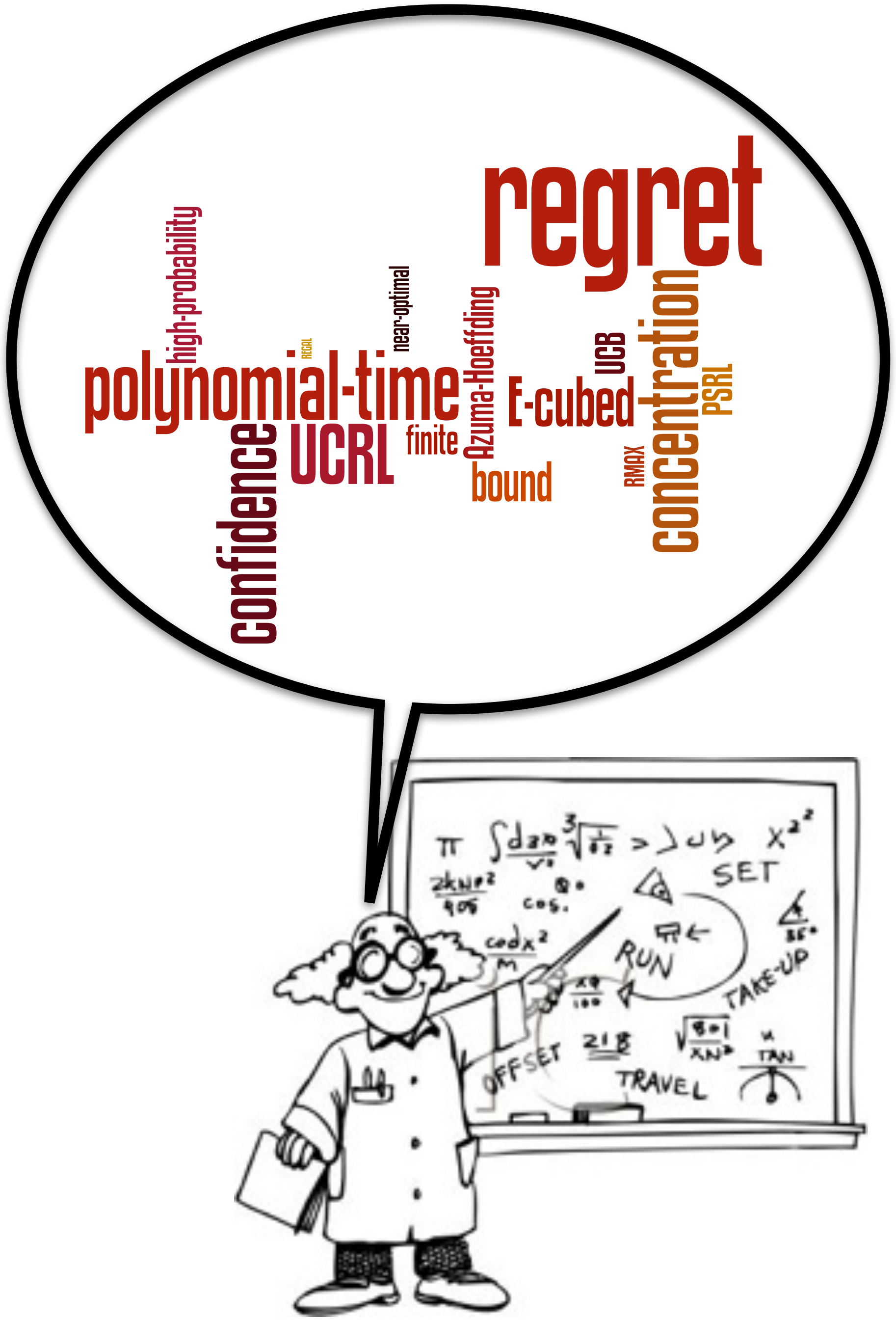
$$\overline{r}_h(s,a) = \overline{r}'_h(s,a)$$

$$\overline{p}_h(s,a) = \left(1 - \frac{1}{n_h(s,a)+1}\right)\overline{p}'_h(s,a) + \frac{1}{n_h(s,a)+1}\frac{1}{S} \rightarrow \overline{p}'_h(s,a)$$

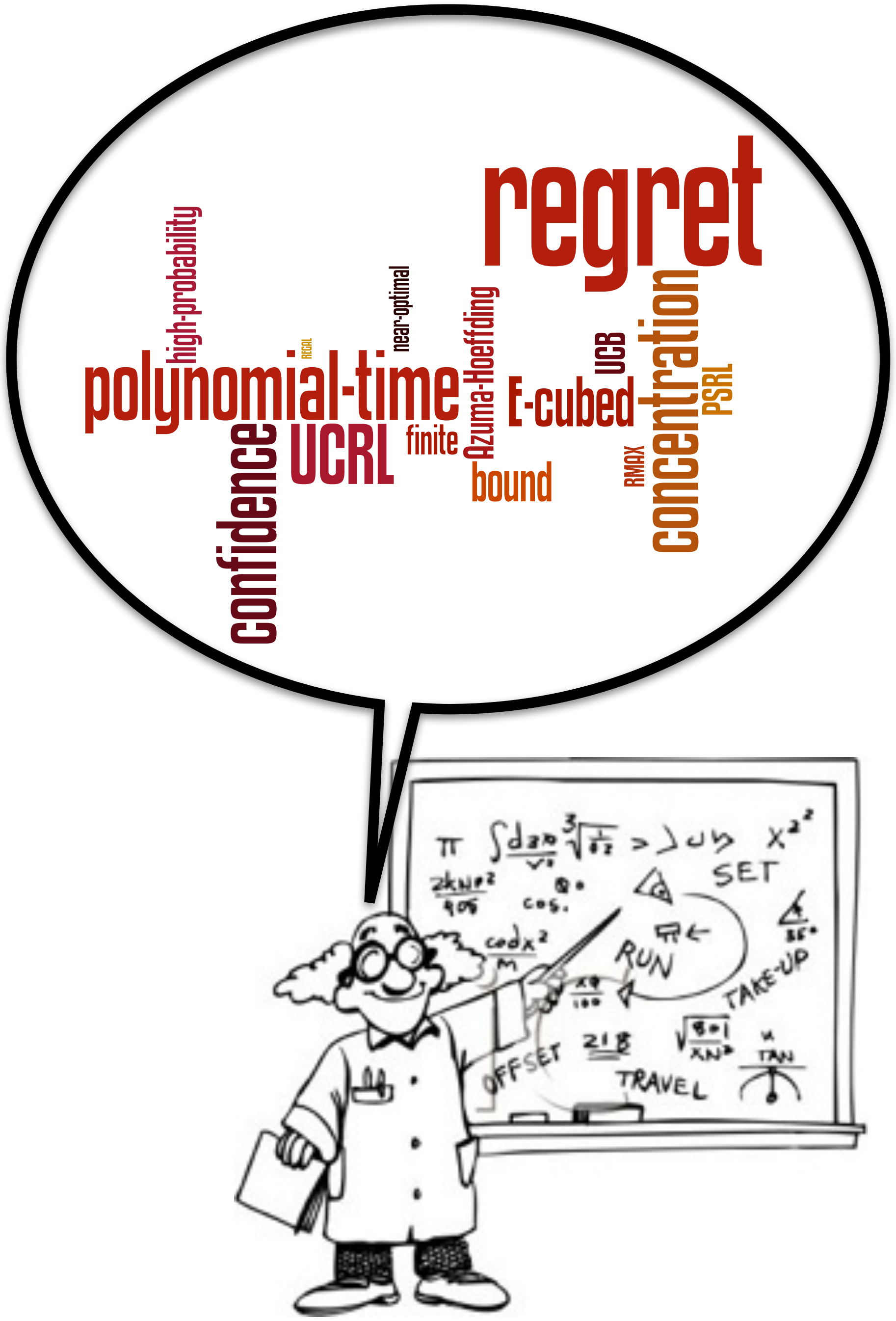

SET
RUN
TAKE-UP
OFFSET
TRAVEL
TAN


SET
cos.
RUN
TAKE-UP
OFFSET
218
TRAVEL
TAN


COLA


regret
polynomial-time
confidence
UCRL
high-probability
near-optimal
Azuma-Hoeffding
finite
bound
E-cubed
UCB
RMAX
concentration
PSRL
SET
RUN
TAKE-UP
OFFSET
TRAVEL


SARSA
greedy
gradient
divergence
projection
convergence
Boltzmann
approximate
feature
LSPI
Q-learning
on-policy
off-policy
contraction
TD
stepsize
exploration
generalization
LSTD
basis
approximation
function


regret
polynomial-time
confidence
UCRL
concentration
E-cubed
bound
Azuma-Hoeffding
finite
UCB
PSRL
high-probability
near-optimal
RMAX


SARSA
greedy
gradient
divergence
projection
convergence
Boltzmann
approximate
TD
feature
LSPI
Q-learning
on-policy
off-policy
contraction
stepsize
exploration
generalization
LSTD
basis
approximation
function