# Learning to Make Better Decisions: Challenges for the 21st Century

Csaba Szepesvári
University of Alberta
Department of Computing Science

Based on joint work with:
Yasin-Abbasi Yadkori and Dávid Pál

# Making a difference

# Making a difference

- Autonomous cars: Save lives of people dying on the road

# Making a difference

- Autonomous cars: Save lives of people dying on the road
- Voice-user interface systems: Humanizing computer-human interaction

# Making a difference

- Autonomous cars: Save lives of people dying on the road
- Voice-user interface systems: Humanizing computer-human interaction
- Dynamic treatment regimes: Save patients. Maximize treatment efficiency while avoiding ill effects

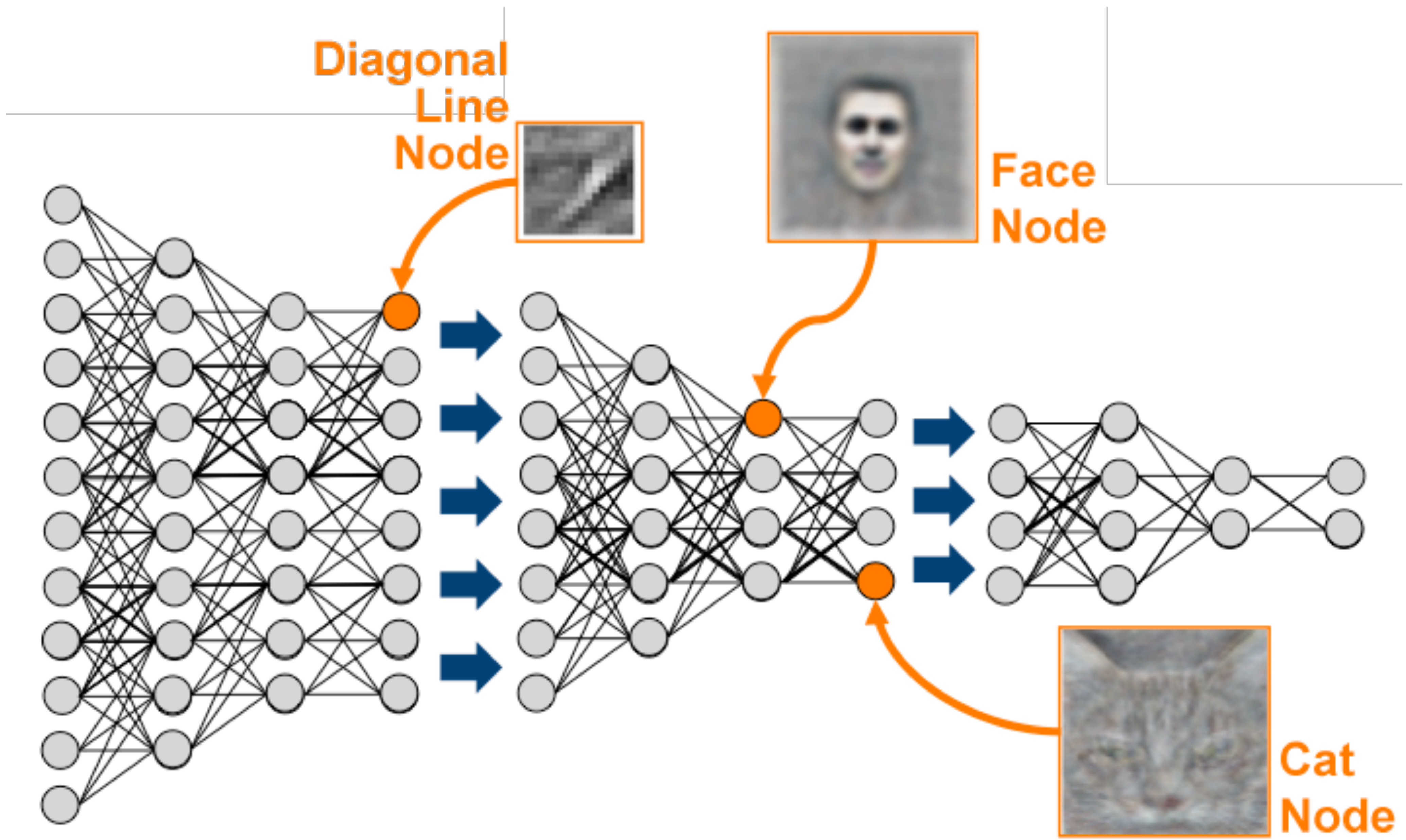
# Making a difference

- Autonomous cars: Save lives of people dying on the road
- Voice-user interface systems: Humanizing computer-human interaction
- Dynamic treatment regimes: Save patients. Maximize treatment efficiency while avoiding ill effects
- Intelligent Tutoring: Bring education to the masses while improving it

How?


# Explosion of data


# Computation

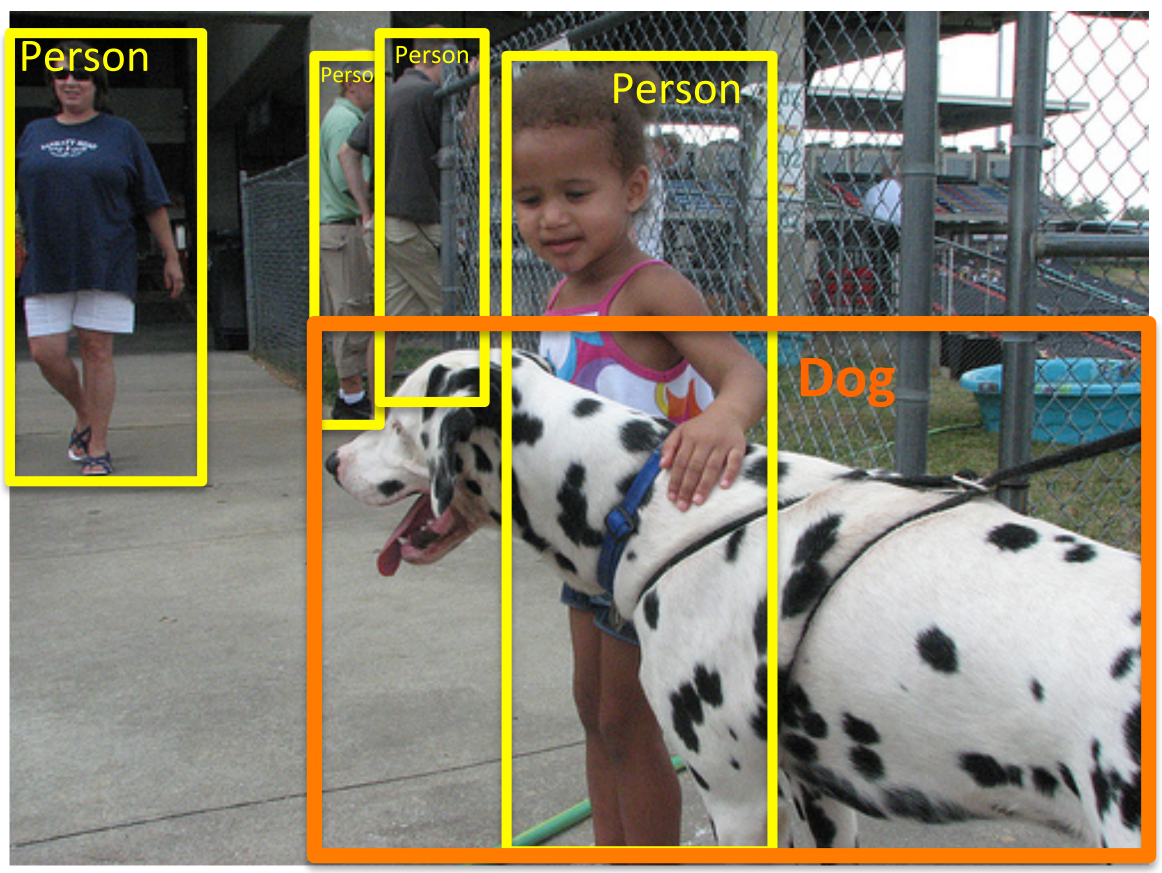



# Improved Learning Methods

How far did we get?

# IMAGENET Large Scale Visual Recognition Challenge (ILSVRC) 2010-2014

**1000 object classes** **1,431,167 images** **CLS-LOC**




http://image-net.org/challenges/LSVRC/

# Evaluation

Steel drum

| **Output:** |
| --- |
| Scale |
| T-shirt |
| <u>Steel drum</u> |
| Drumstick |
| Mud turtle |

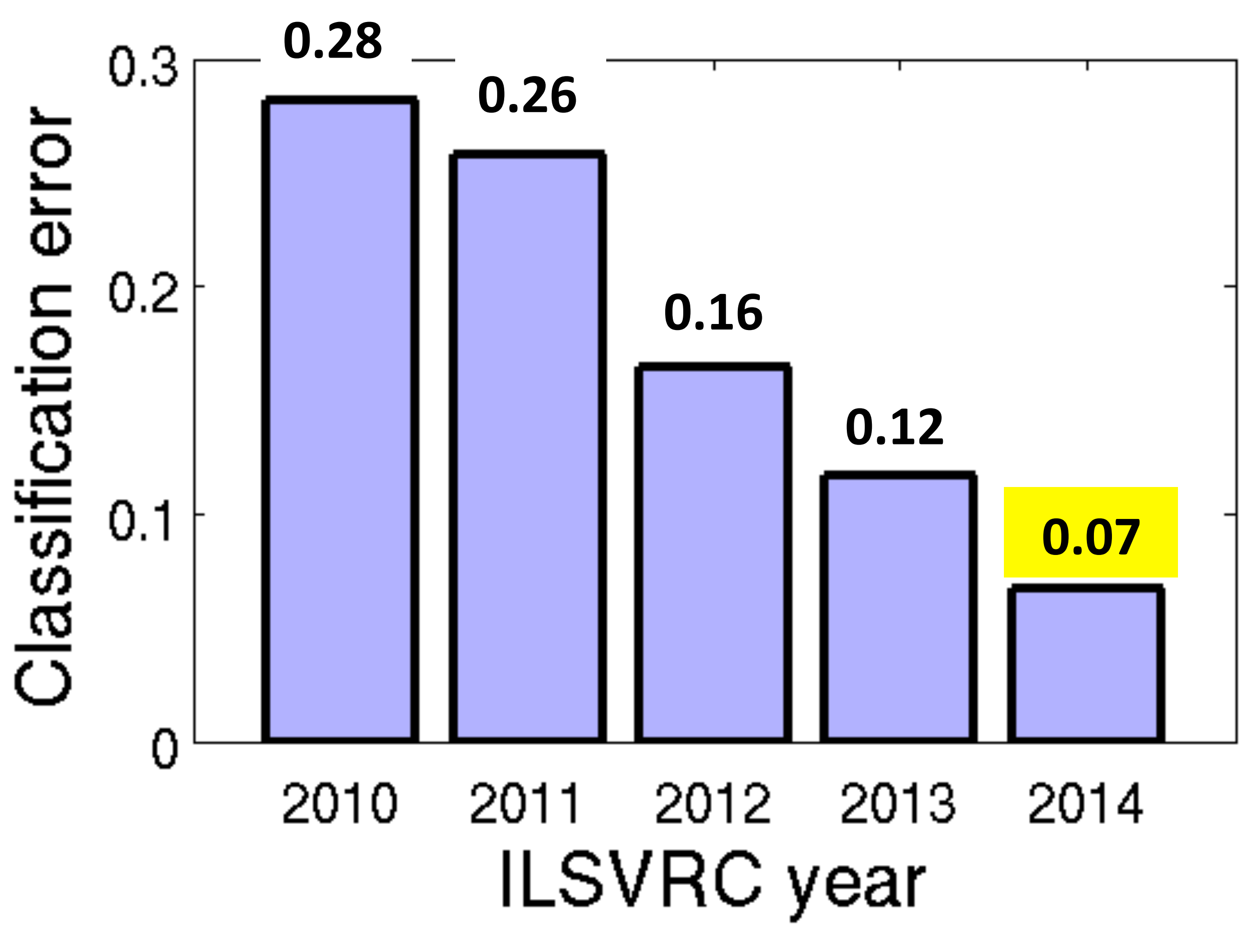
✔

| **Output:** |
| --- |
| Scale |
| T-shirt |
| Giant panda |
| Drumstick |
| Mud turtle |

✗

$$\text{Error} = \frac{1}{100{,}000} \sum_{\substack{100{,}000 \\ \text{images}}} 1[\text{incorrect on image i}]$$

# Progress



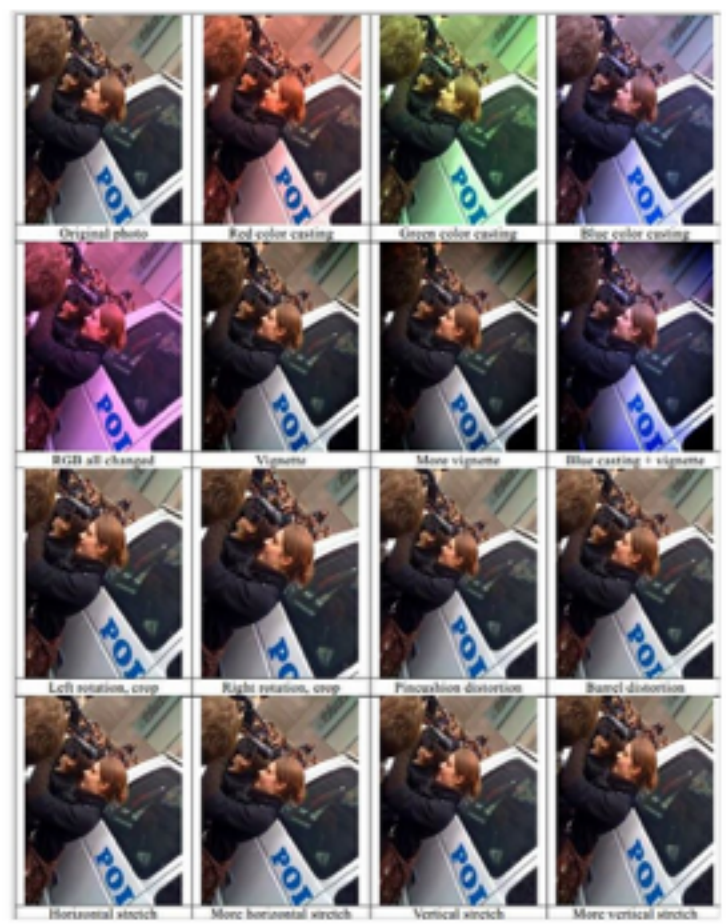

1.7x reduction in classification error since last year

4.2x reduction in classification error since 2010

# And the war goes on..

**Andrew Ng**
52 mins ·

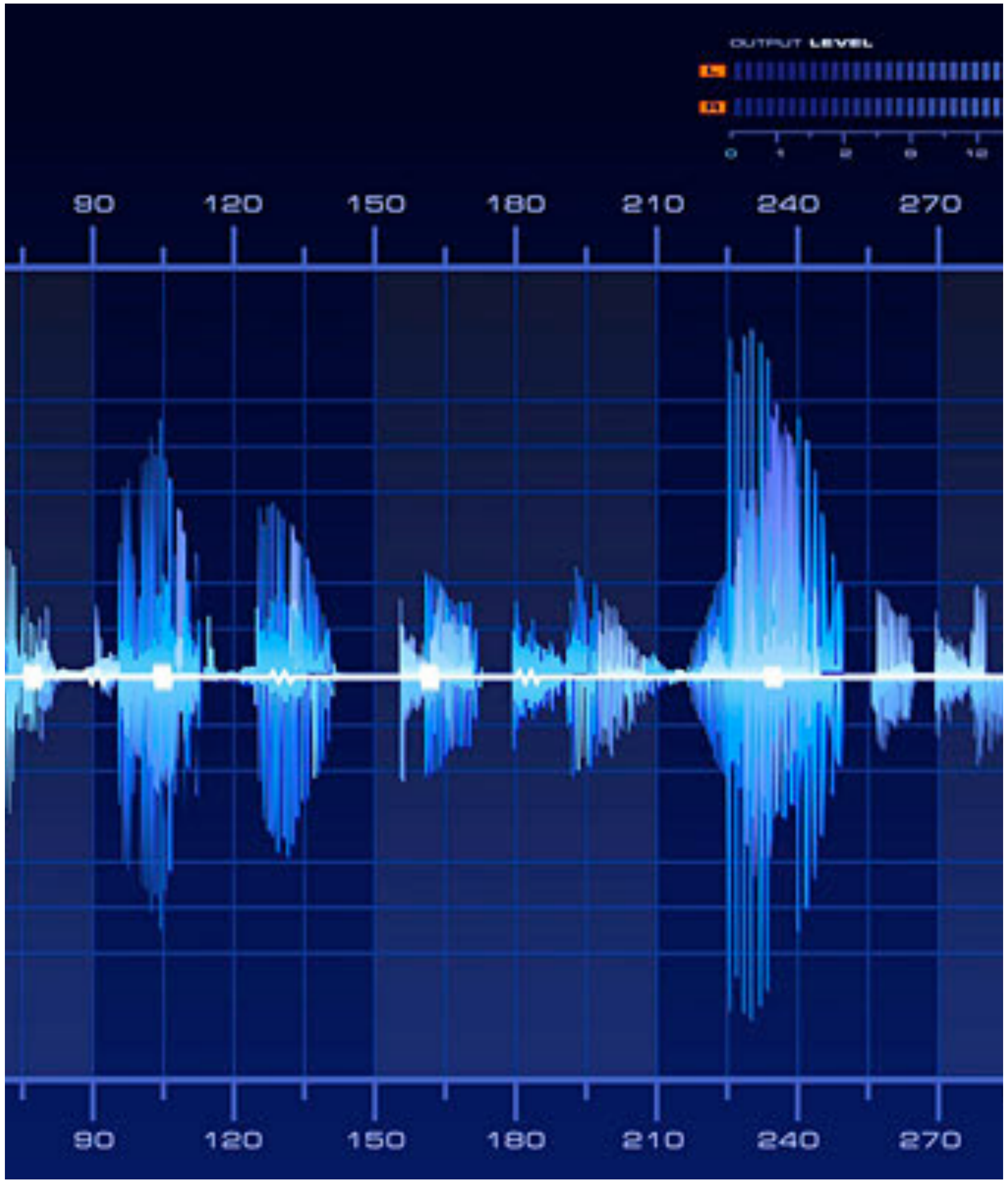
Baidu Research just attained the best computer vision ImageNet classification result 5.98% error (vs. GoogLeNet's 6.66%). The key to this was our multi-GPU deep learning infrastructure, which by using a mix of model-parallelism and data-parallelism, allows us to train our model 24.7x faster than using only a single GPU. This scale also allows us to use higher-resolution images, and absorb more (synthetic) training data. Paper here: bit.ly/deepimage



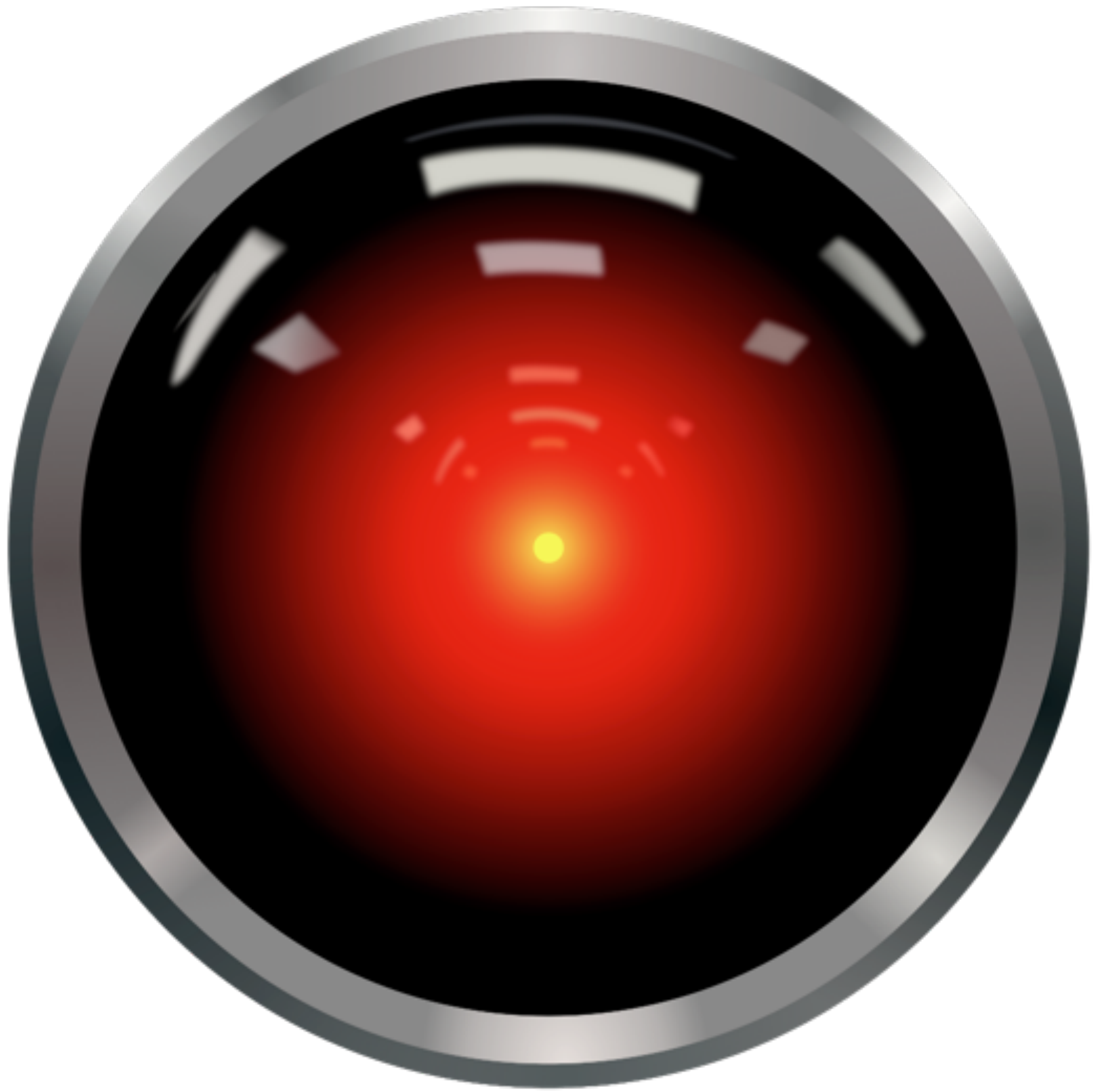

Unlike · Comment · Share

# Speech recognition

- Google
- Apple
- Baidu
- Achievements:
  - Error rates constantly drop since 2009, halved or so..
  - “Speech 2.0”



# Are we done?

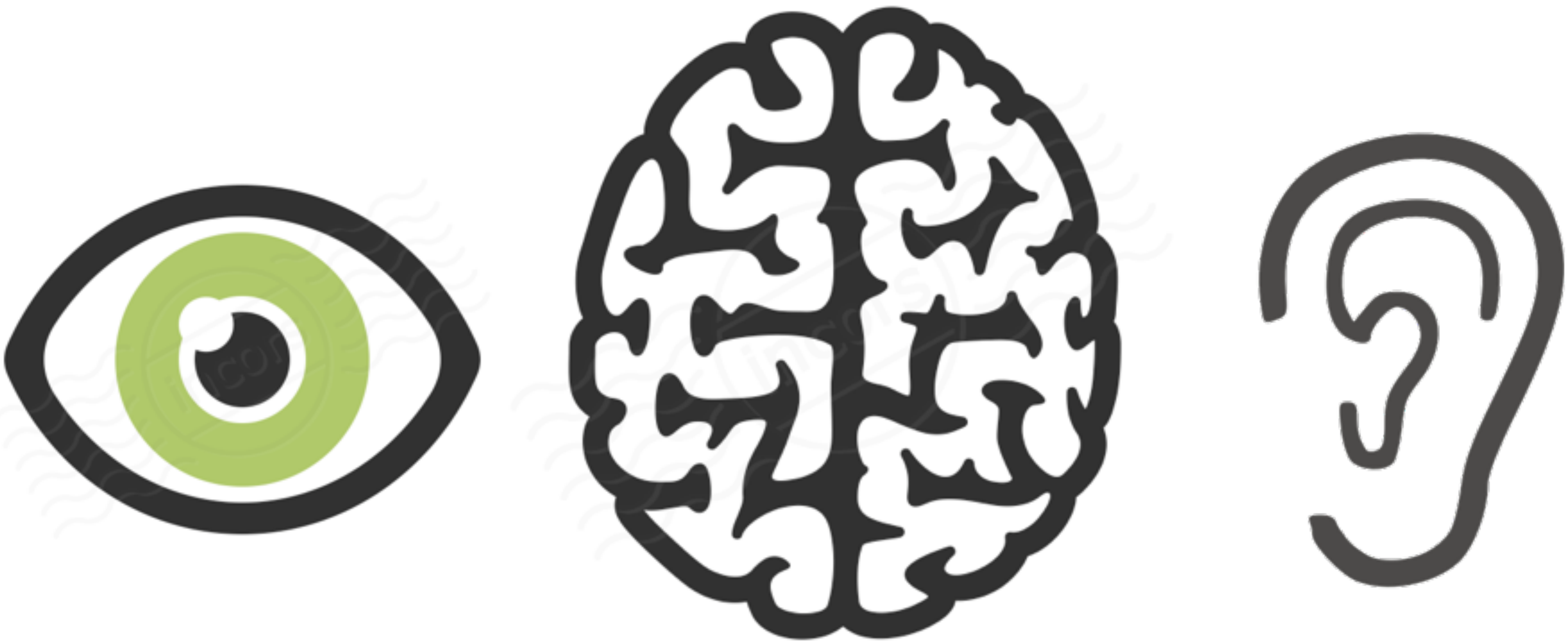


# Are we done?

# Are we done?

# Are we done?

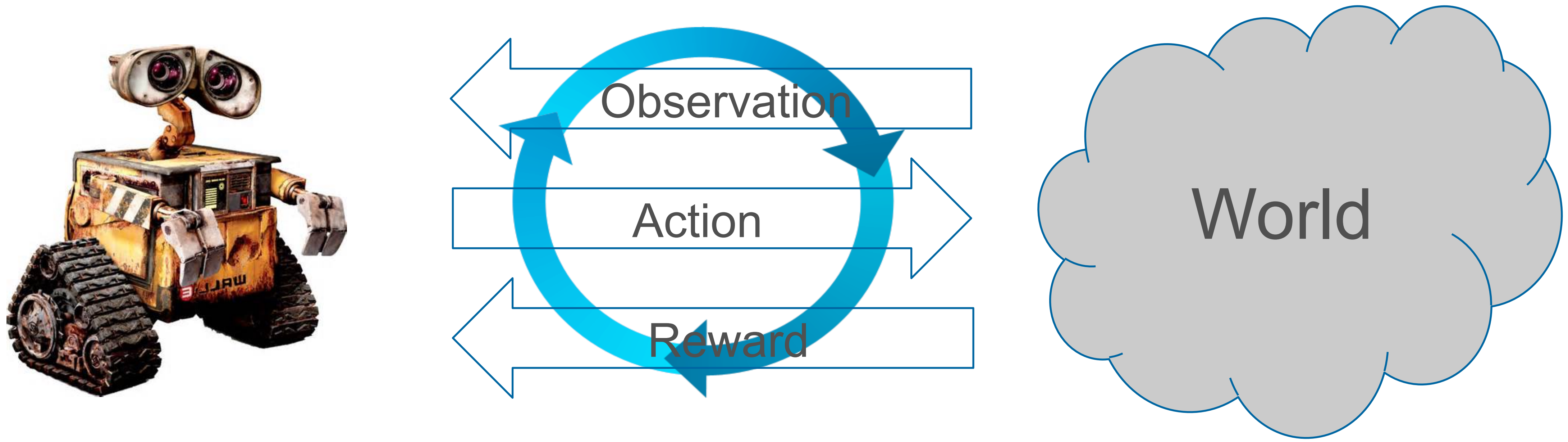
# Are we done?



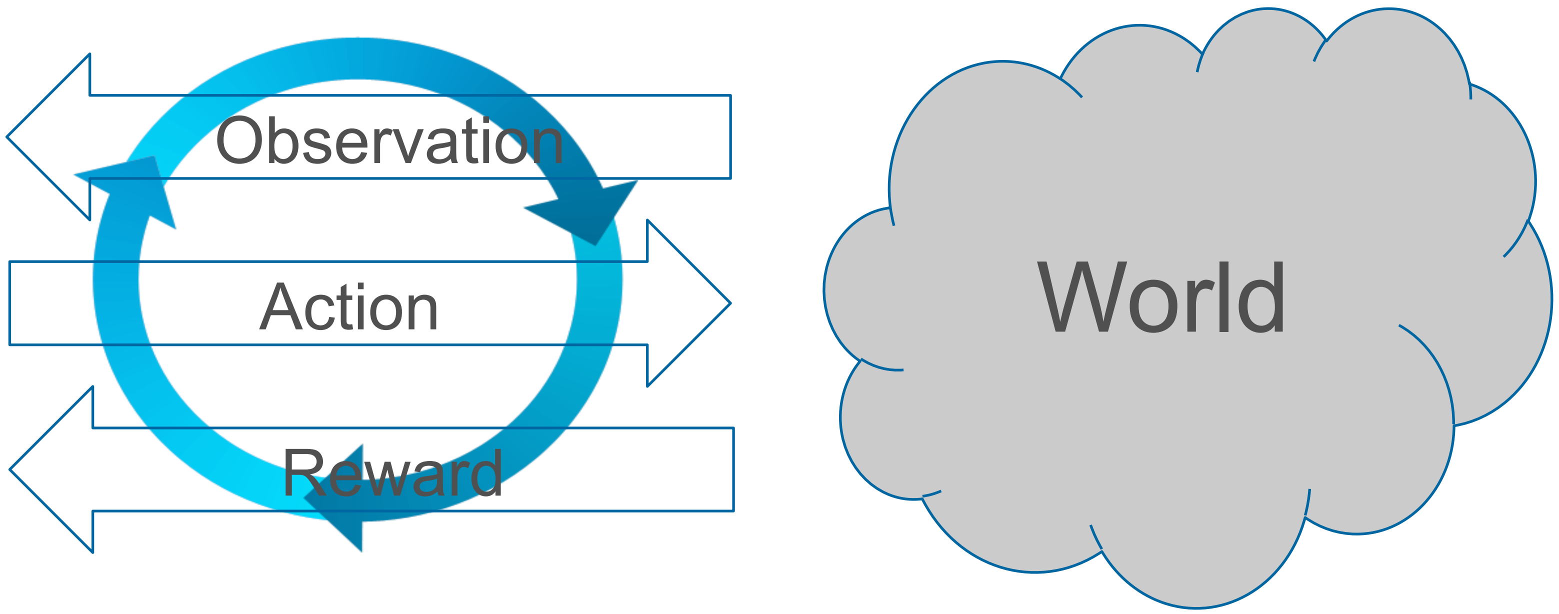
# Are we done?

Need to make decisions!

# RL to the Rescue

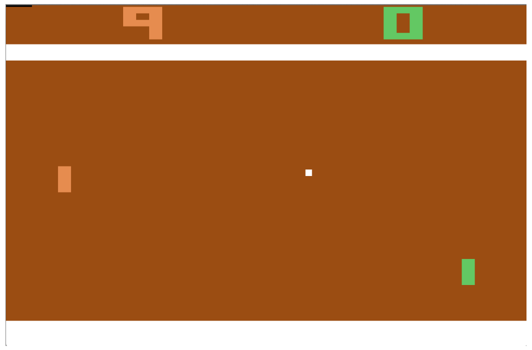


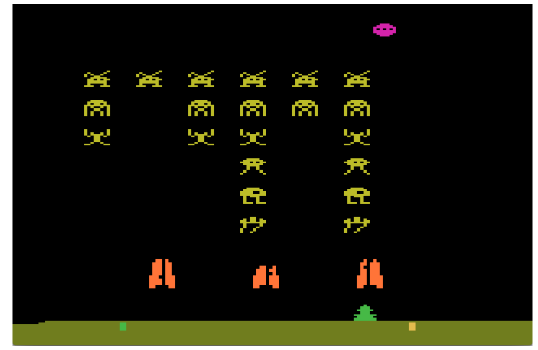
# RL to the Rescue




Goal: Maximize the total reward collected

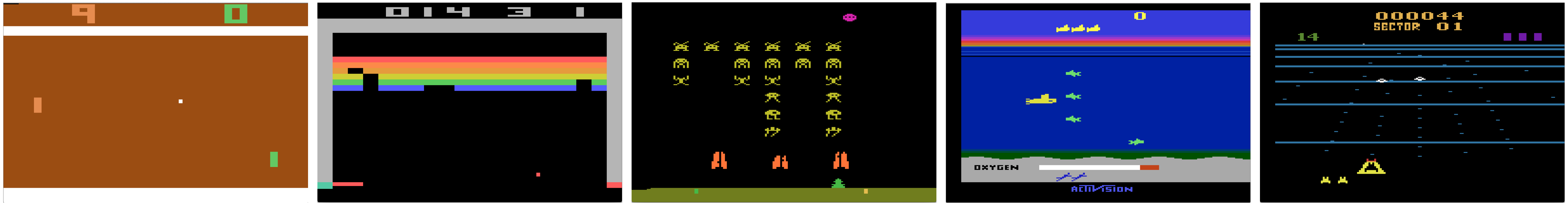
# Google DeepMind: RL meets Deep Learning and Big Data







# Google DeepMind:
# RL meets Deep Learning and Big Data




| | B. Rider | Breakout | Enduro | Pong | Q*bert | Seaquest | S. Invaders |
|---|---|---|---|---|---|---|---|
| **Random** | 354 | 1.2 | 0 | −20.4 | 157 | 110 | 179 |
| **Sarsa** [3] | 996 | 5.2 | 129 | −19 | 614 | 665 | 271 |
| **Contingency** [4] | 1743 | 6 | 159 | −17 | 960 | 723 | 268 |
| **DQN** | **4092** | **168** | **470** | **20** | **1952** | **1705** | **581** |
| **Human** | 7456 | 31 | 368 | −3 | 18900 | 28010 | 3690 |
| **HNeat Best** [8] | 3616 | 52 | 106 | 19 | 1800 | 920 | **1720** |
| **HNeat Pixel** [8] | 1332 | 4 | 91 | −16 | 1325 | 800 | 1145 |
| **DQN Best** | **5184** | **225** | **661** | **21** | **4500** | **1740** | 1075 |

Table 1: The upper table compares average total reward for various learning methods by running an $\epsilon$-greedy policy with $\epsilon = 0.05$ for a fixed number of steps. The lower table reports results of the single best performing episode for HNeat and DQN. HNeat produces deterministic policies that always get the same score while DQN used an $\epsilon$-greedy policy with $\epsilon = 0.05$.

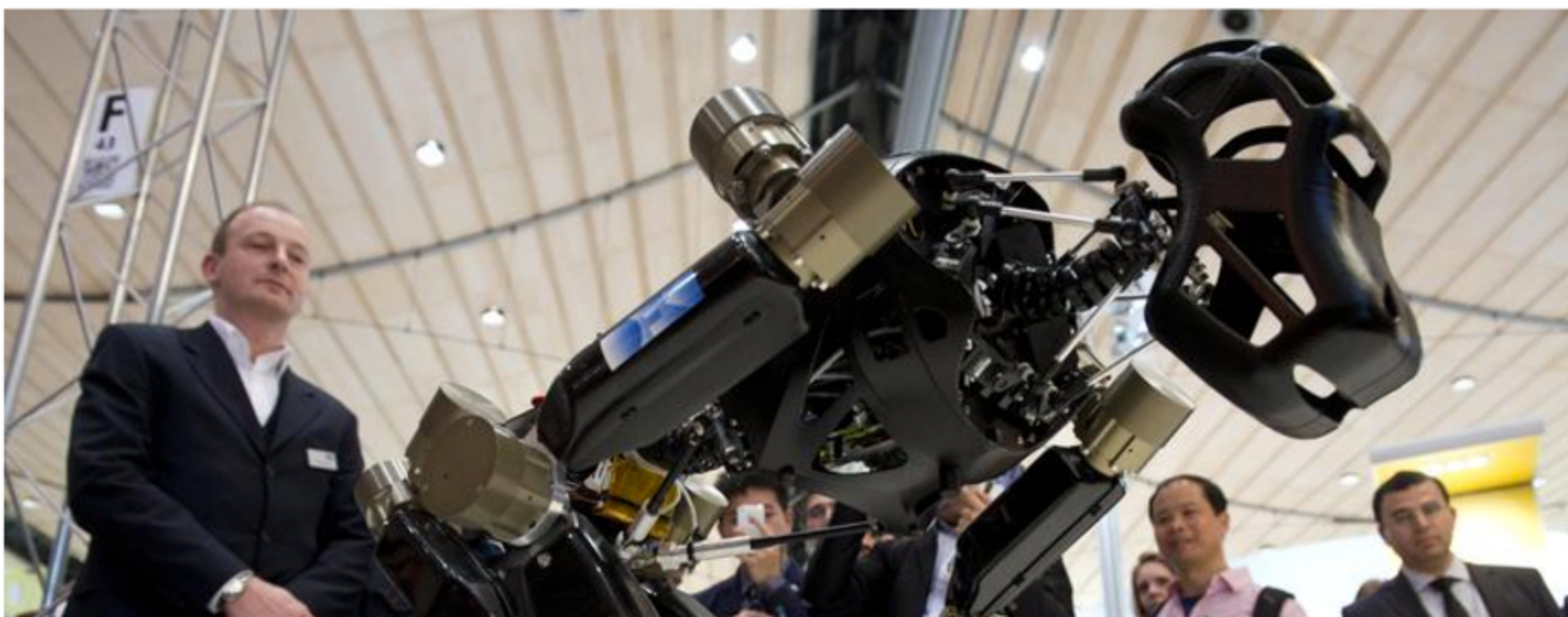
# Google DeepMind:
# RL meets Deep Learning and Big Data

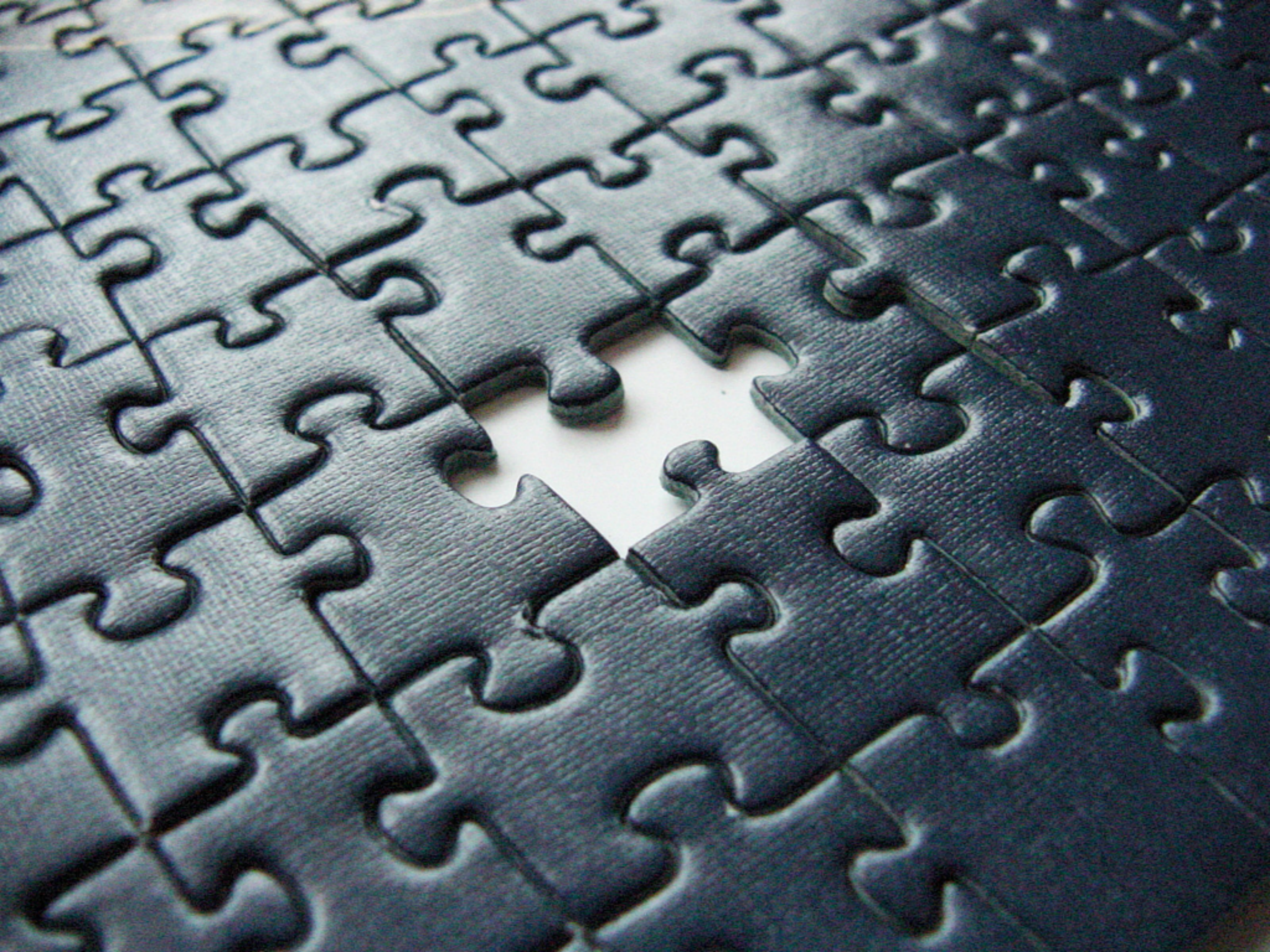
# Artificial intelligence experts sign open letter to protect mankind from machines

The Future of Life Institute wants humanity to tread lightly while developing really smart machines.

by Nick Statt @nickstatt / 12 January 2015 12:10 am GMT

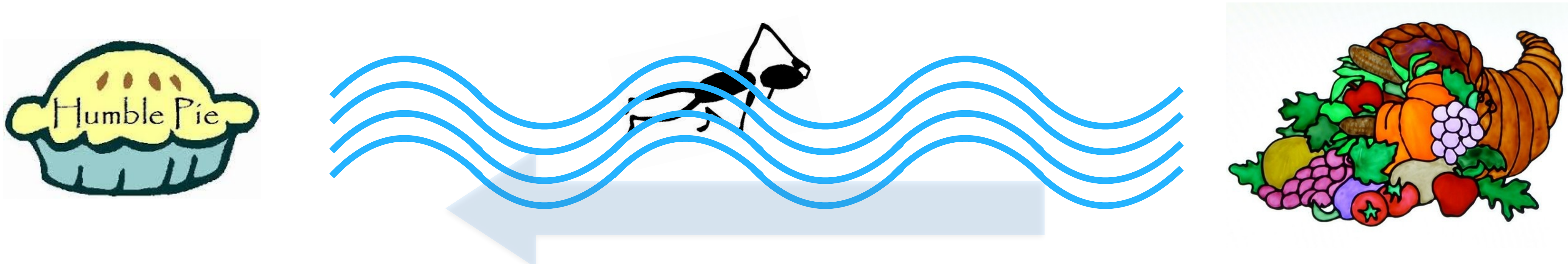
73 / 1.2K / 1.1K / 249 / / more +





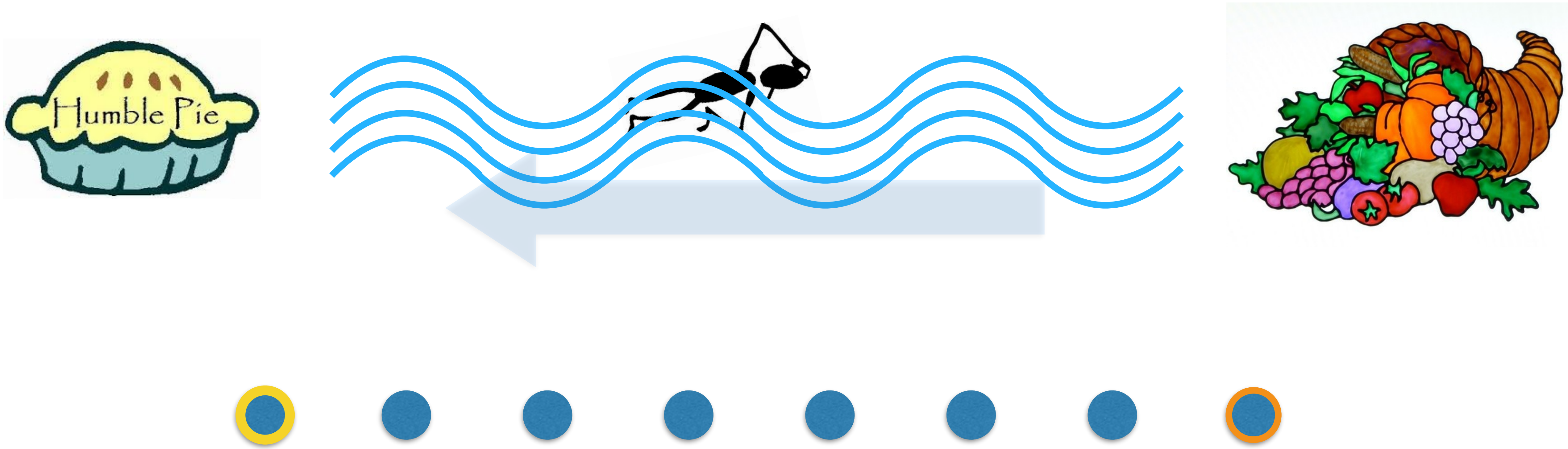
# On Data Collection

# A Swimming Lesson



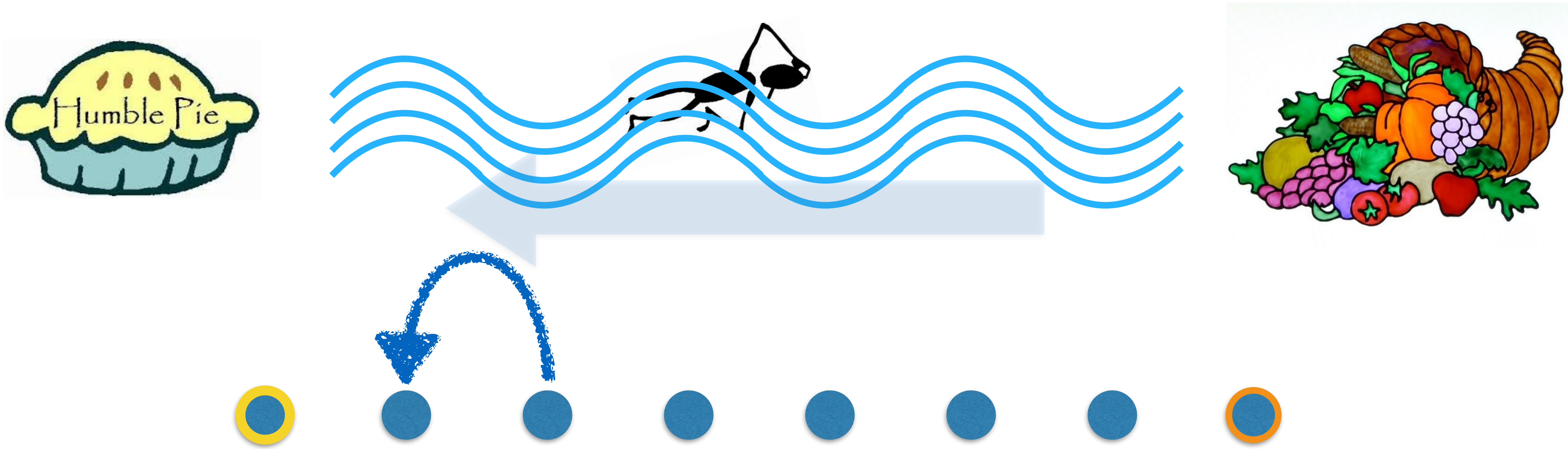

Slide graphics courtesy of Ben van Roy.
Problem due to [Strehl-Littman,'08]

# A Swimming Lesson



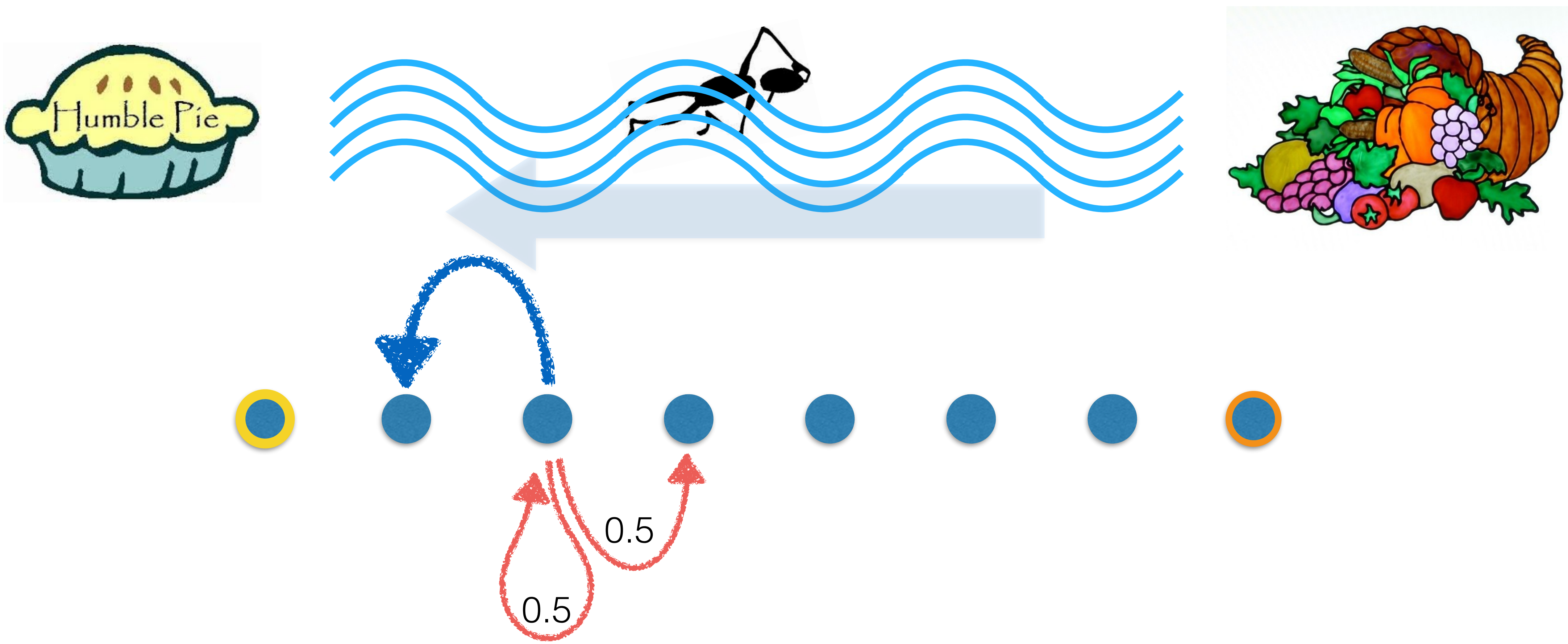

Slide graphics courtesy of Ben van Roy.
Problem due to [Strehl-Littman,'08]

# A Swimming Lesson



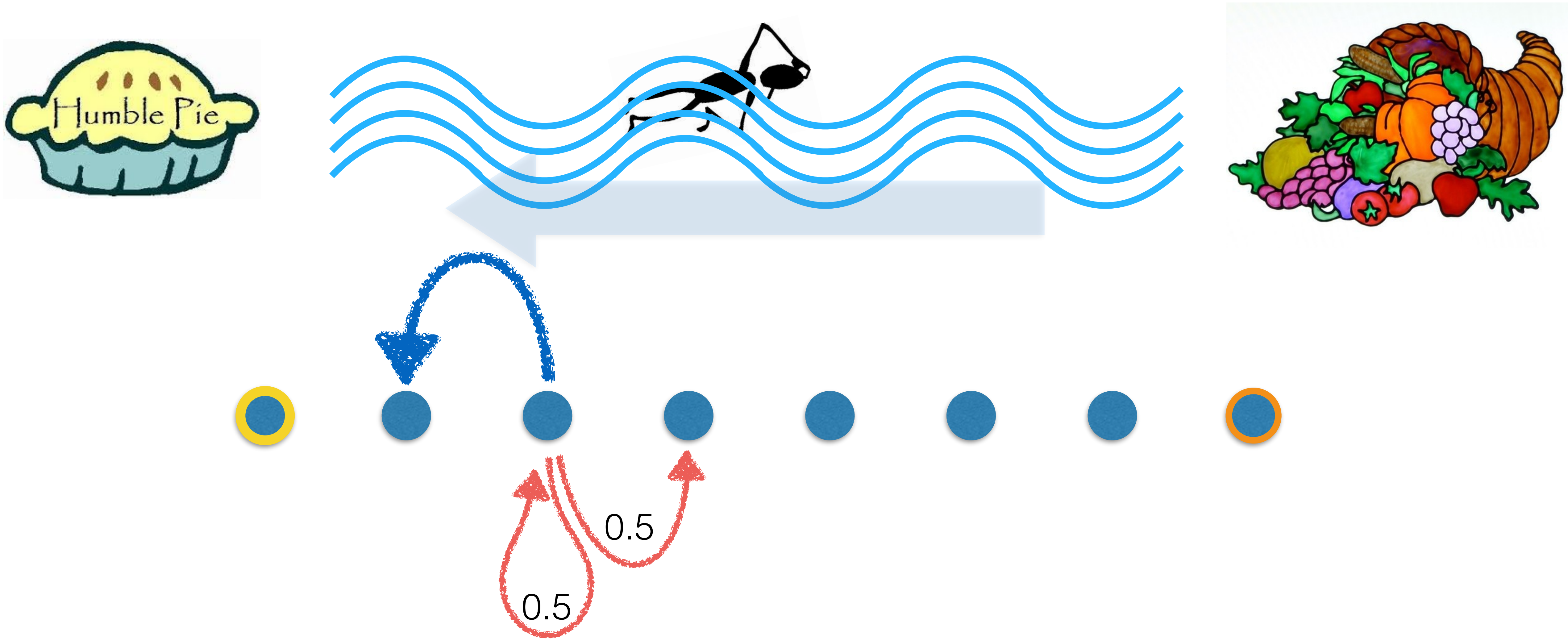

Slide graphics courtesy of Ben van Roy.
Problem due to [Strehl-Littman,'08]

# A Swimming Lesson




Slide graphics courtesy of Ben van Roy.
Problem due to [Strehl-Littman,'08]

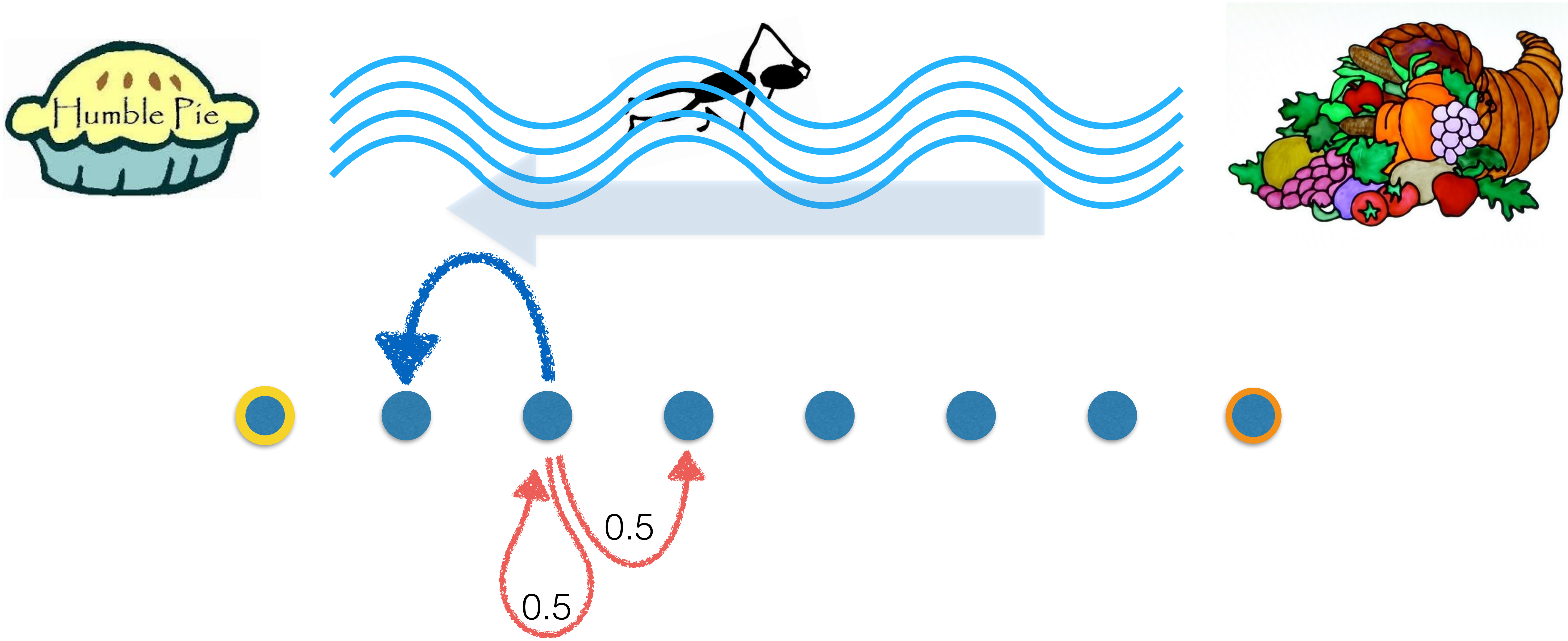
# A Swimming Lesson




- **Reckless** data collection: Choose the actions ***uniformly at random***!

Slide graphics courtesy of Ben van Roy.
Problem due to [Strehl-Littman,'08]

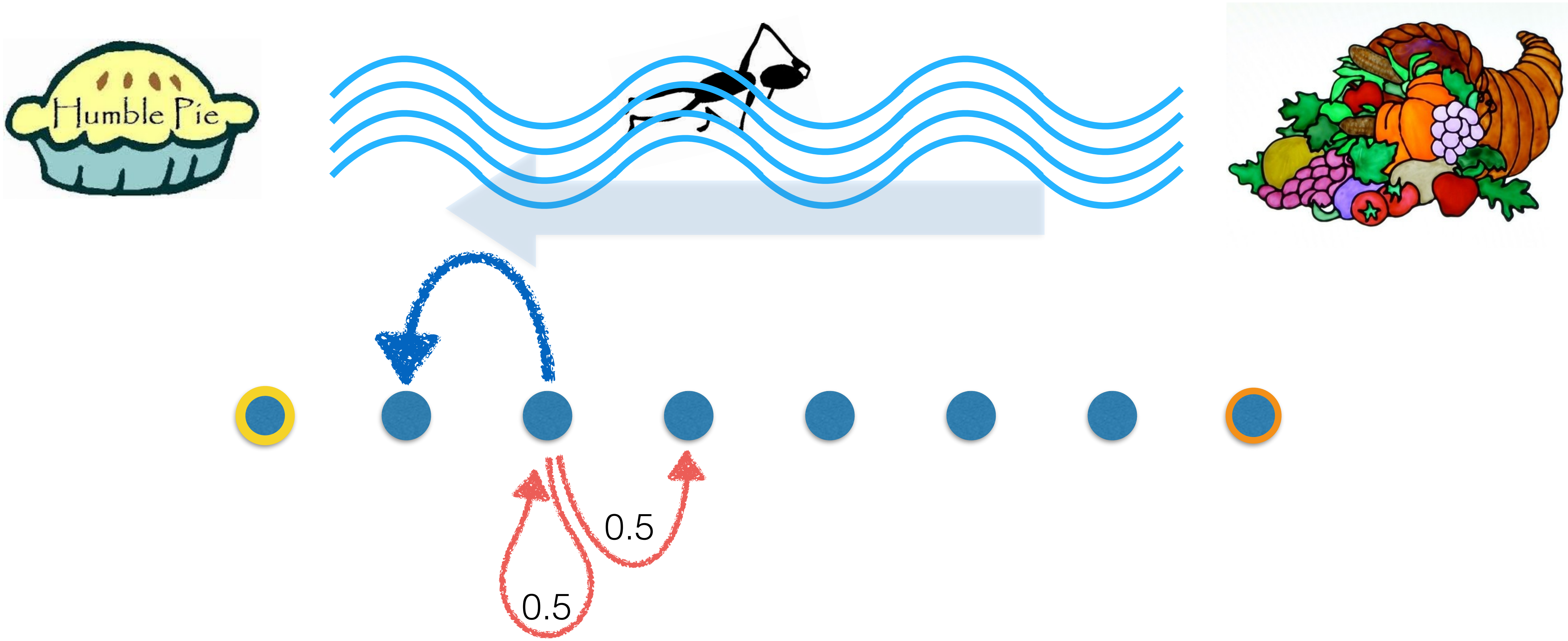
# A Swimming Lesson




- **Reckless** data collection: Choose the actions ***uniformly at random***!
- **How much data** do we need to collect before we see the bounty for the first time, starting from the middle?

Slide graphics courtesy of Ben van Roy.
Problem due to [Strehl-Littman,'08]

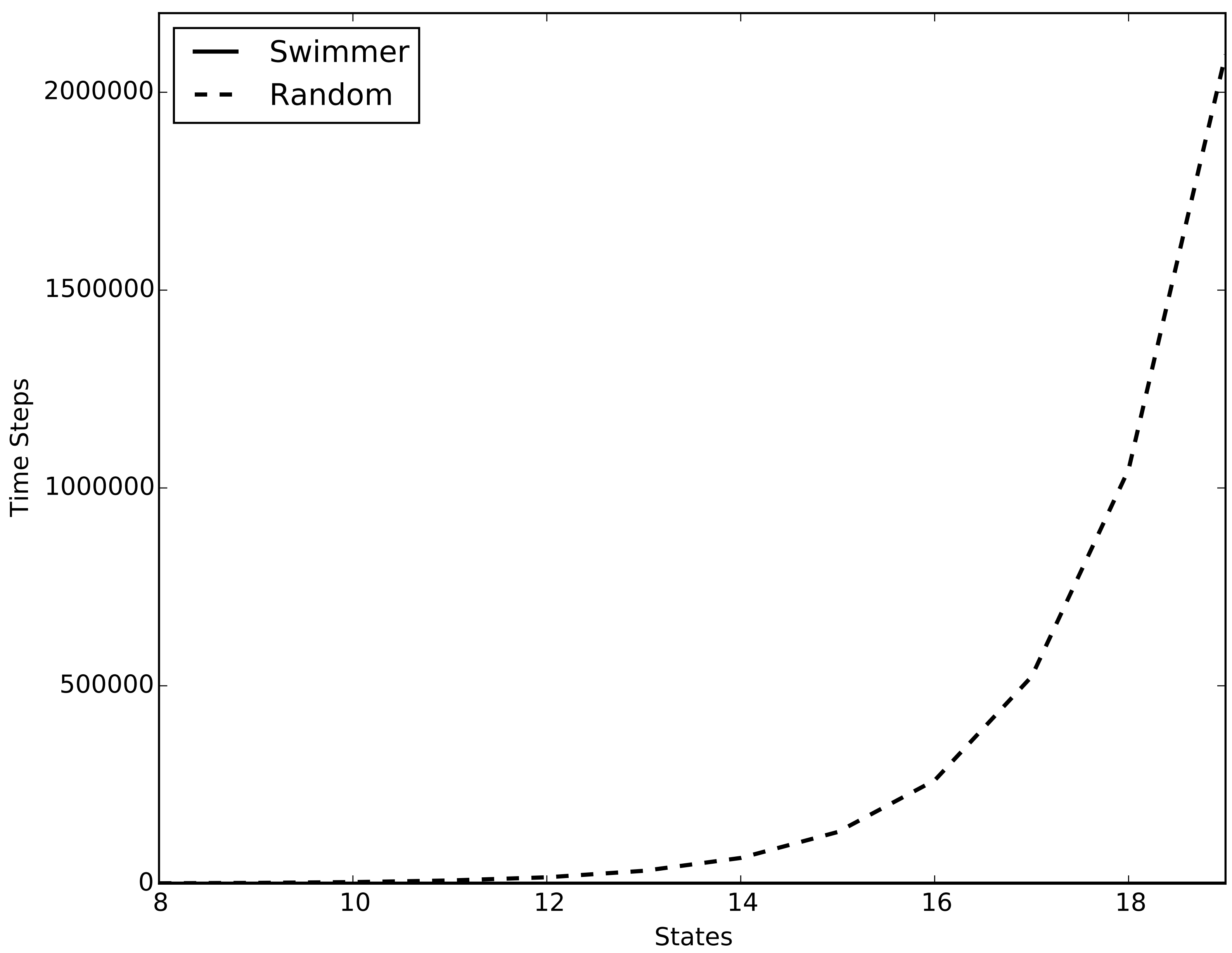
# A Swimming Lesson

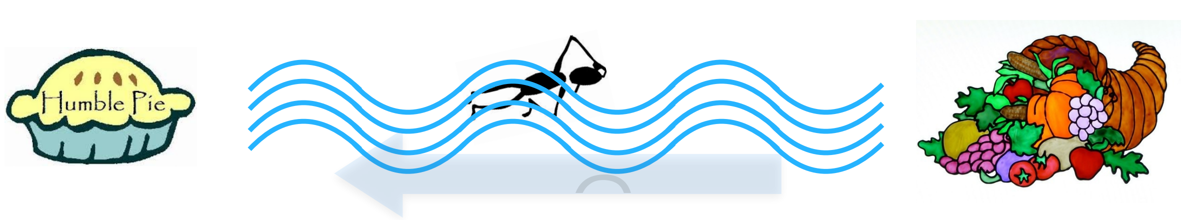



- **Reckless** data collection: Choose the actions ***uniformly at random***!
- **How much data** do we need to collect before we see the bounty for the first time, starting from the middle?
- How does this depend on the number of states?

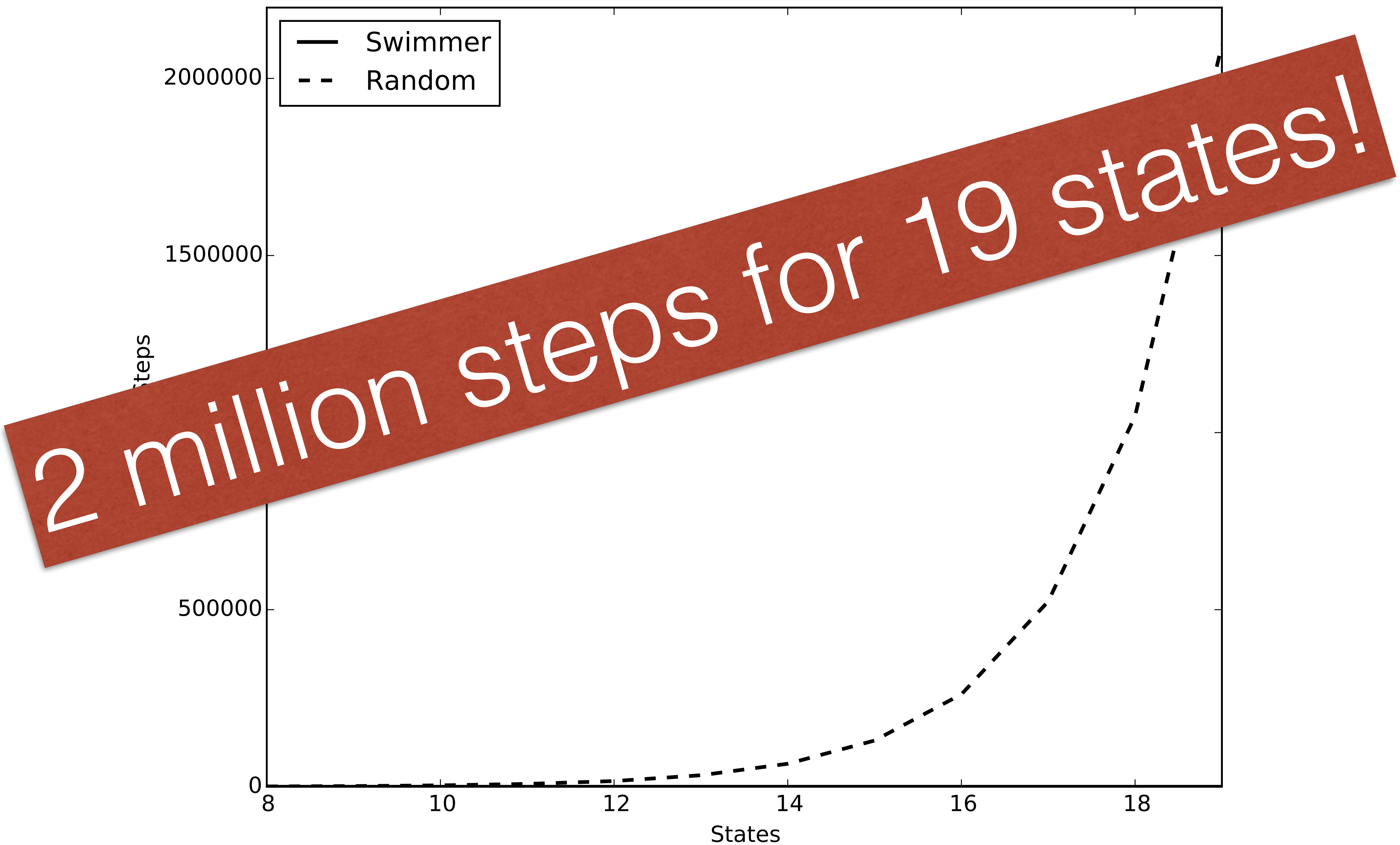
Slide graphics courtesy of Ben van Roy.
Problem due to [Strehl-Littman,'08]

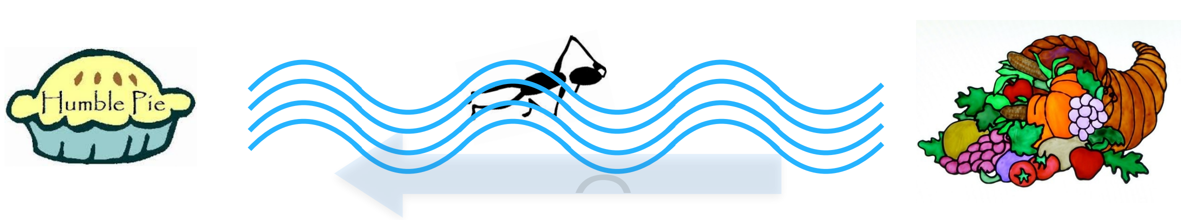
# Time before bounty is found



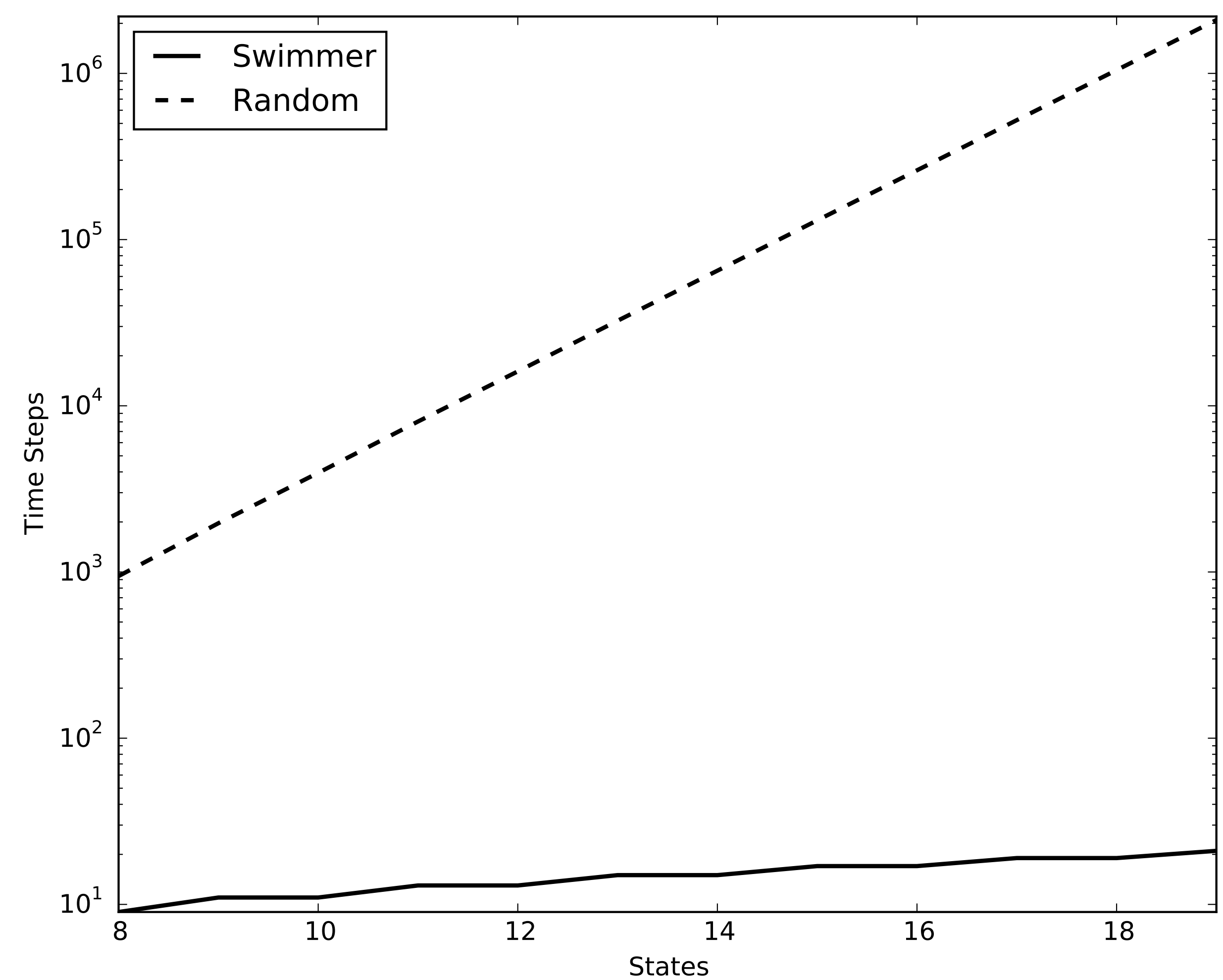



# Time before bounty is found



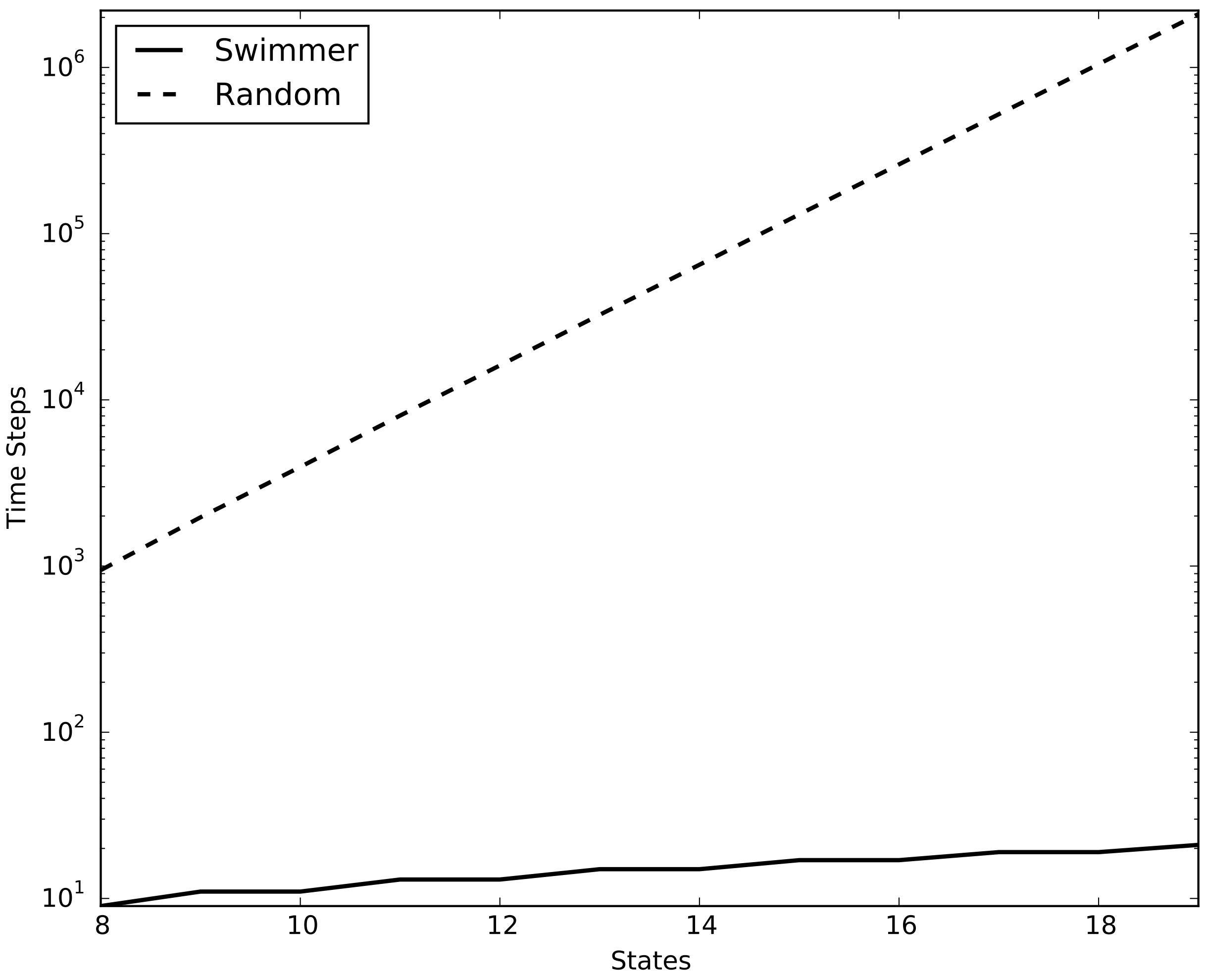
# Time before bounty is found



# Time before bounty is found

- Hitting time for random policy:

$$\Theta(2^n)$$



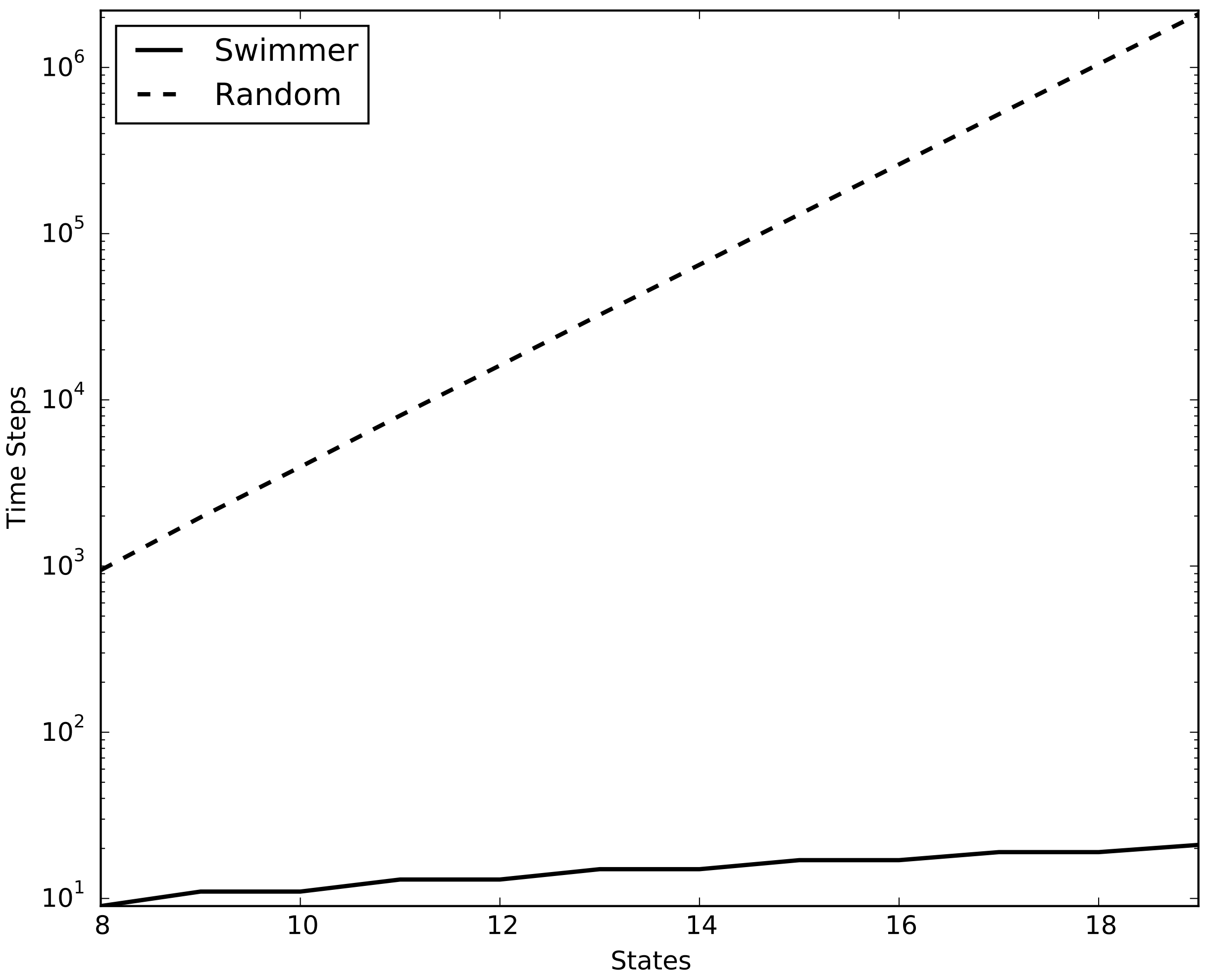
# Time before bounty is found

- Hitting time for random policy:

$$\Theta(2^n)$$

- Hitting time for "swimming policy":

$$\Theta(n)$$



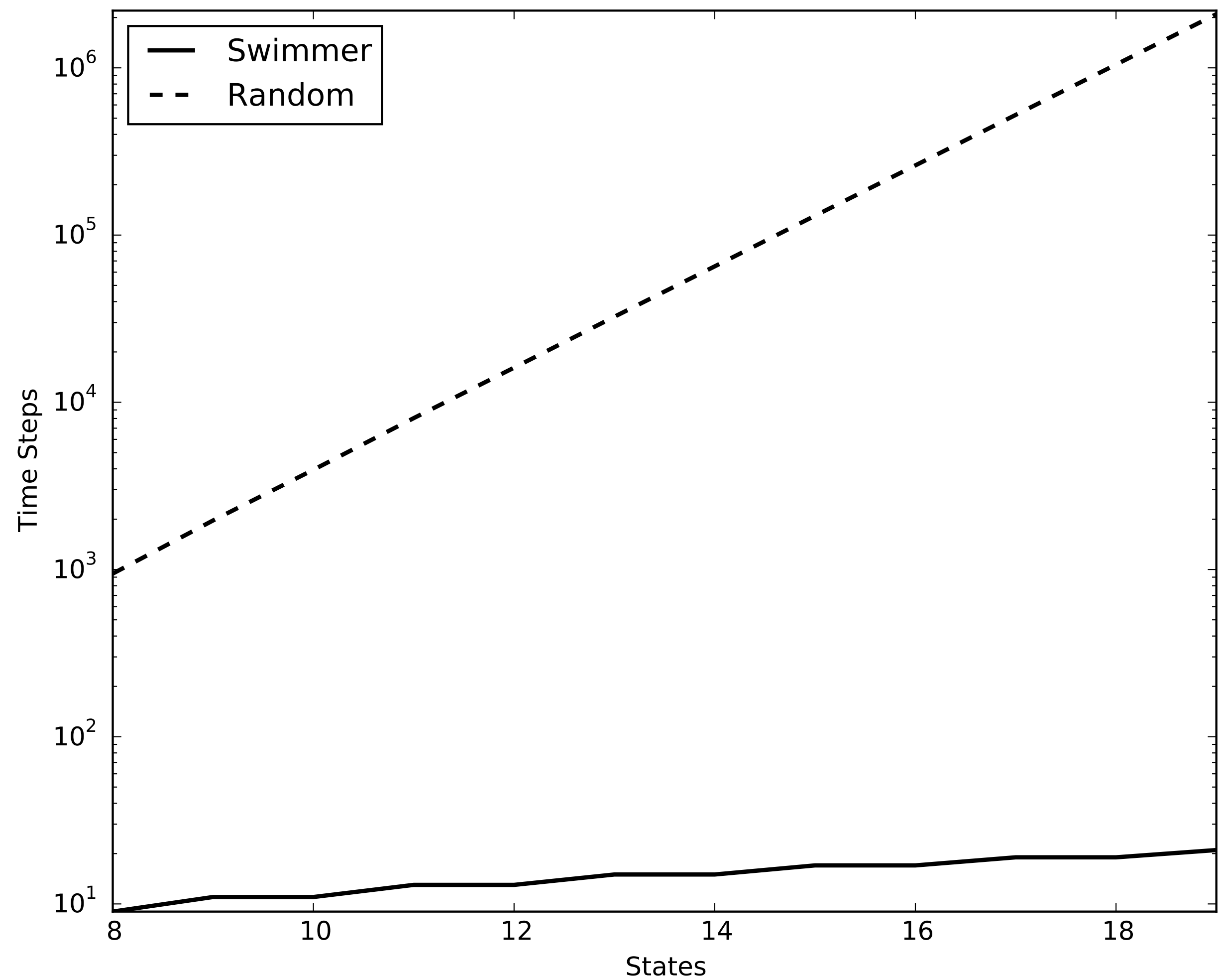
# Time before bounty is found

- Hitting time for random policy:

$$\Theta(2^n)$$

- Hitting time for "swimming policy":

$$\Theta(n)$$




- Exponential gap on a very simple example!
..could be ***much*** worse on a real problem

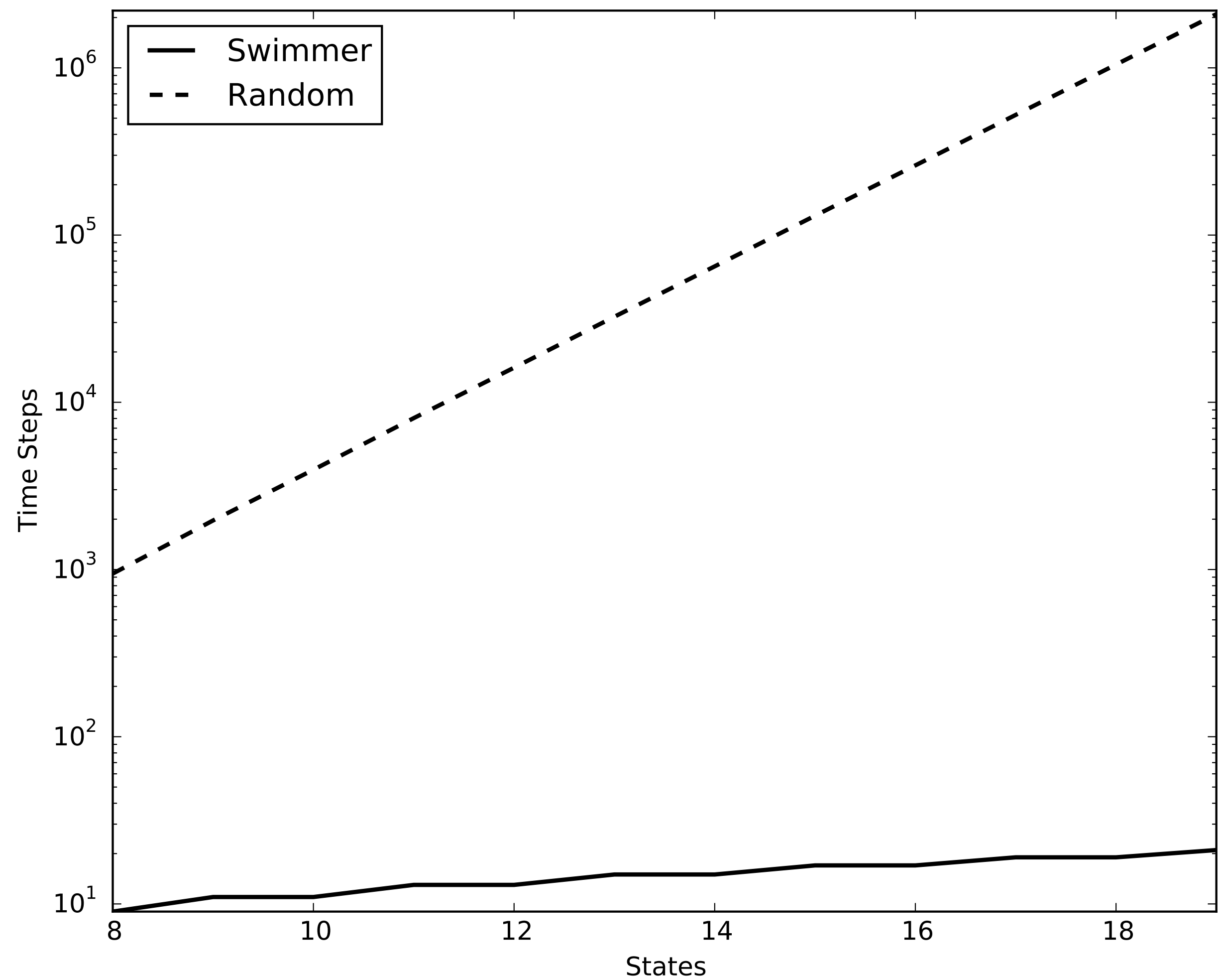
# Time before bounty is found

- Hitting time for random policy:

$$\Theta(2^n)$$

- Hitting time for "swimming policy":

$$\Theta(n)$$




- Exponential gap on a very simple example!
  ..could be ***much*** worse on a real problem
- How "big" is big enough?

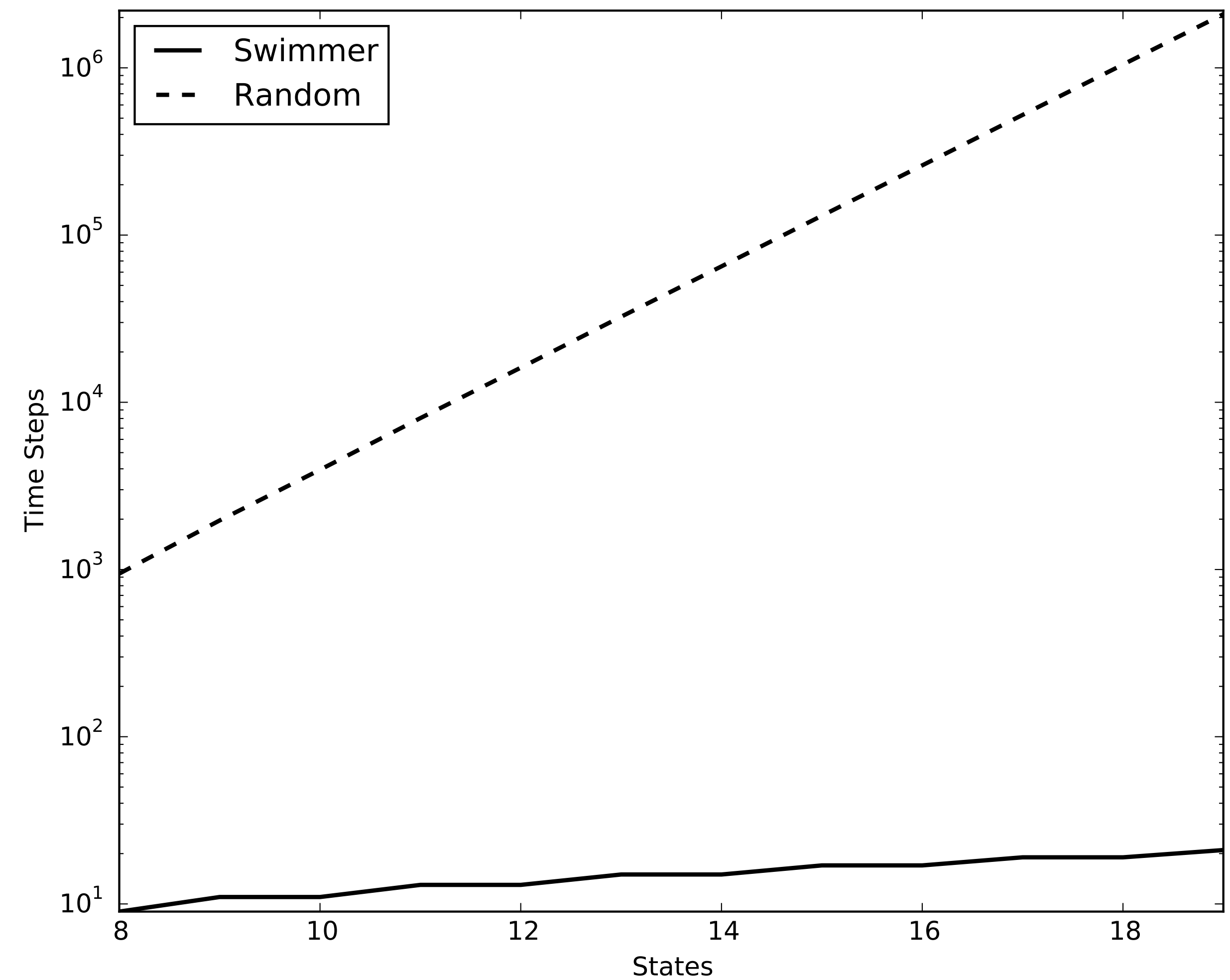
# Time before bounty is found

- Hitting time for random policy:

$$\Theta(2^n)$$

- Hitting time for "swimming policy":

$$\Theta(n)$$




- Exponential gap on a very simple example! ..could be ***much*** worse on a real problem
- How "big" is big enough?
- Will we ever have enough data? Can we do better?

# Changing the game...

# Changing the game..

- Allow data to be collected by a policy we select

# Changing the game..

- Allow data to be collected by a policy we select

- Can we design more efficient data collection policies?

# Standard RL Approach

# Standard RL Approach

- Repeat:

# Standard RL Approach

- Repeat:
  - Learn a “good” policy

# Standard RL Approach

- Repeat:
  - Learn a “good” policy
  - Add randomness to induce exploration

# Standard RL Approach

- Repeat:
  - Learn a “good” policy
  - Add randomness to induce exploration
  - Collect more data (multiple episodes)

# Standard RL Approach

- Repeat:
  - Learn a “good” policy
  - Add randomness to induce exploration
  - Collect more data (multiple episodes)
- “epsilon-greedy”, “Boltzmann exploration”

# Standard RL Approach

- Repeat:
  - Learn a “good” policy
  - Add randomness to induce exploration
  - Collect more data (multiple episodes)
- “epsilon-greedy”, “Boltzmann exploration”
- “Dithering”

# What happens with dithering in RiverSwim?

# What happens with dithering in RiverSwim?


What is the policy learned initially?
How long do we need to wait until the bounty is first collected?

# What happens with dithering in RiverSwim?

Dithering is NOT sufficient
Need smart exploration methods

What is the policy learned initially?
How long do we need to wait until the bounty is first collected?

# How do we evaluate a data collection strategy?

# How do we evaluate a data collection strategy?

- How much data is needed to find a good policy?

# How do we evaluate a data collection strategy?

- How much data is needed to find a good policy? ..reward collected/lost during data collection does not matter: **“pure exploration” problem**

# How do we evaluate a data collection strategy?

- How much data is needed to find a good policy? ..reward collected/lost during data collection does not matter: **“pure exploration” problem**

- How much reward is incurred during data collection? **“exploitation” problem**

# How do we evaluate a data collection strategy?

- How much data is needed to find a good policy?
  ..reward collected/lost during data collection does not matter: **"pure exploration" problem**

- How much reward is incurred during data collection? **"exploitation" problem**
  Must optimize *while* learning. Explore or exploit?
  Metric: **Regret**.

# The Exploitation Problem

# Optimism in the Face of Uncertainty

Lai and Robbins (1985), Burnetas and Katehakis (1996),
Auer, Cesa-Bianchi and Fischer UCB1 (2002), and many others

# Optimism in the Face of Uncertainty

Repeat:

Lai and Robbins (1985), Burnetas and Katehakis (1996),
Auer, Cesa-Bianchi and Fischer UCB1 (2002), and many others

# Optimism in the Face of Uncertainty

Repeat:

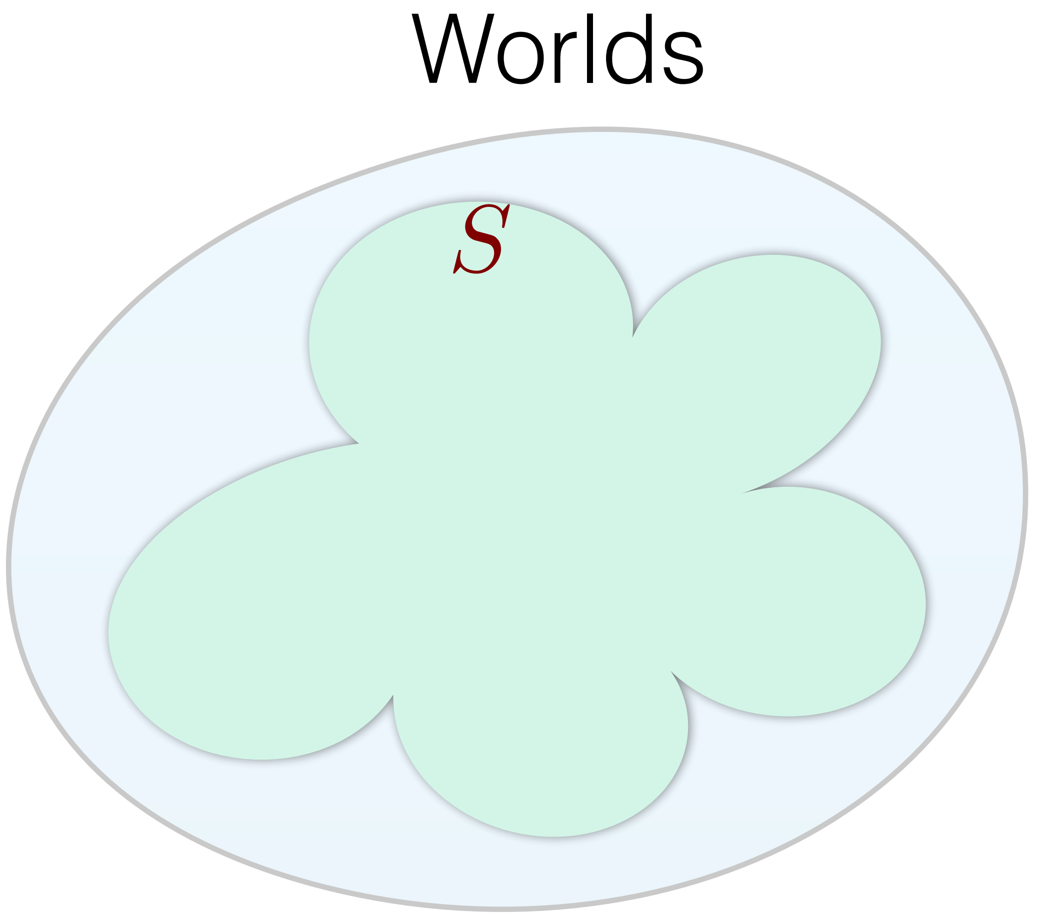
1. Find the set ***S*** of likely "worlds" given the observations so far

Lai and Robbins (1985), Burnetas and Katehakis (1996),
Auer, Cesa-Bianchi and Fischer UCB1 (2002), and many others

# Optimism in the Face of Uncertainty

Repeat:

1. Find the set ***S*** of likely "worlds" given the observations so far



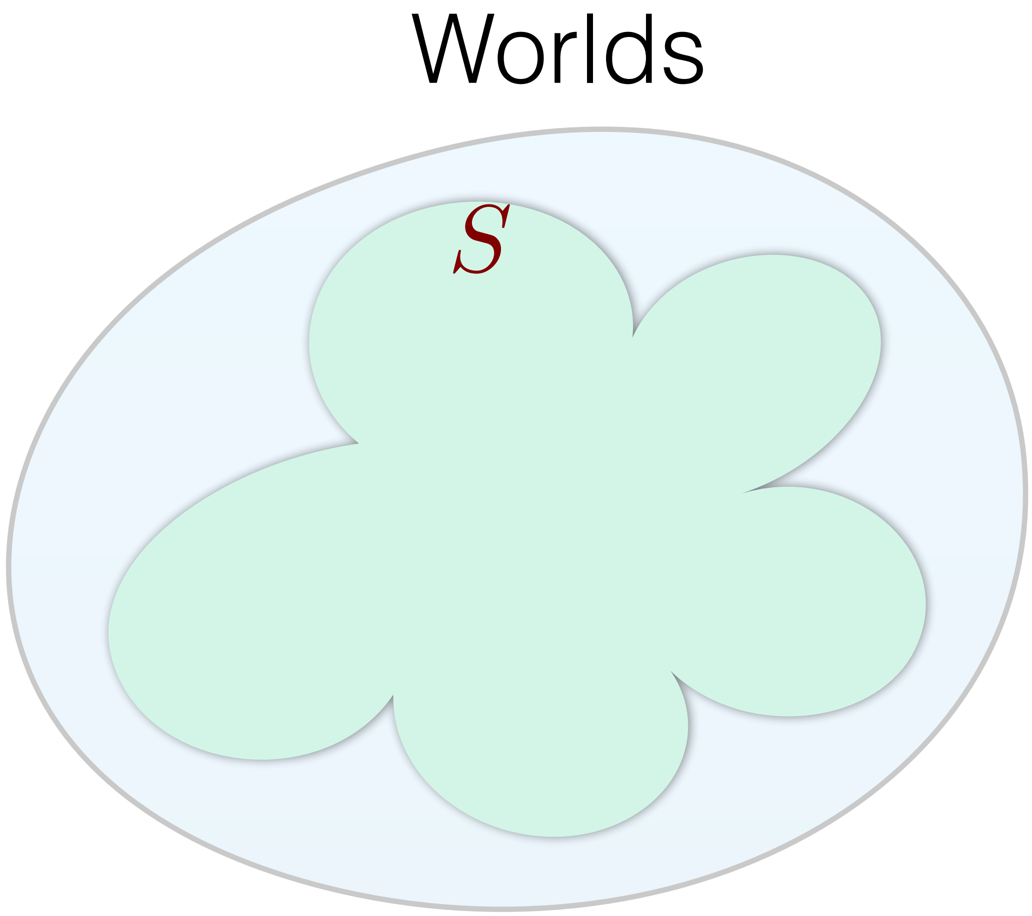

Lai and Robbins (1985), Burnetas and Katehakis (1996),
Auer, Cesa-Bianchi and Fischer UCB1 (2002), and many others

# Optimism in the Face of Uncertainty

Repeat:

1. Find the set ***S*** of likely “worlds” given the observations so far
2. Find the world ***W*** in ***S*** with the maximum payoff:

$$W = \operatorname*{argmax}_{W \in S} J(W)$$



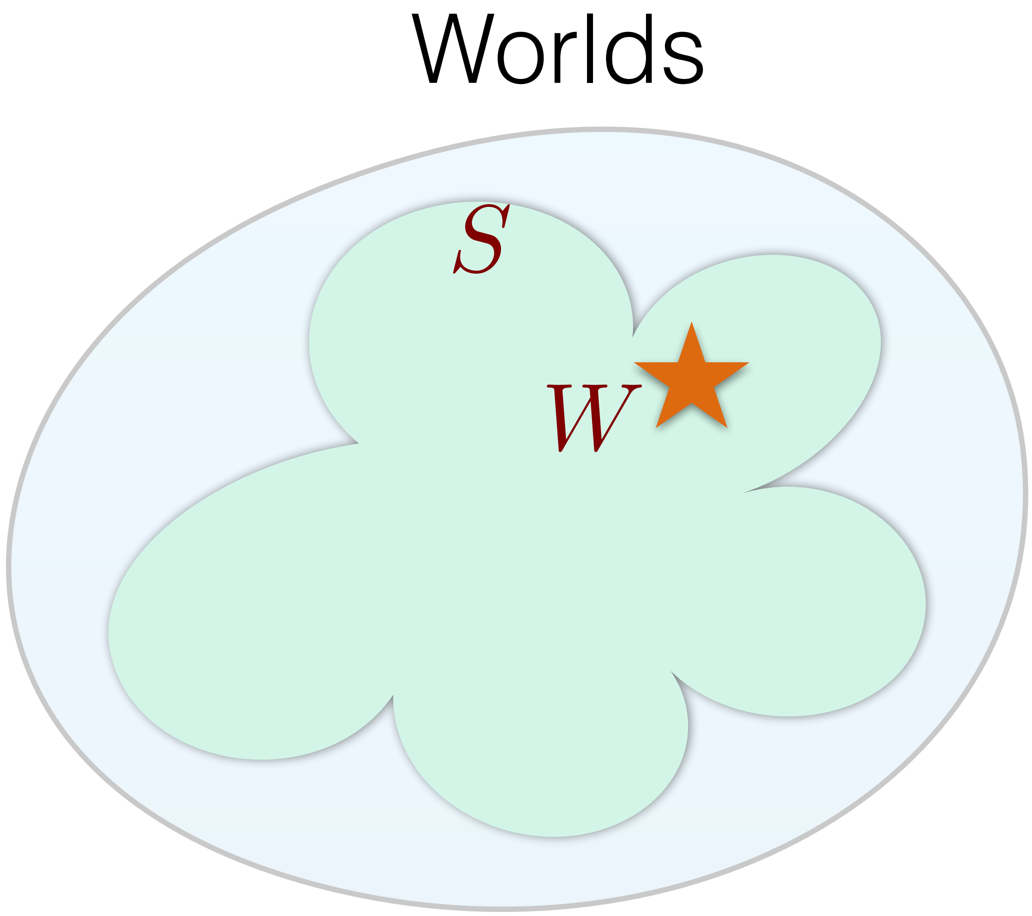

Lai and Robbins (1985), Burnetas and Katehakis (1996),
Auer, Cesa-Bianchi and Fischer UCB1 (2002), and many others

# Optimism in the Face of Uncertainty

Repeat:

1. Find the set ***S*** of likely “worlds” given the observations so far
2. Find the world ***W*** in ***S*** with the maximum payoff:

$$W = \operatorname*{argmax}_{W \in S} J(W)$$




Lai and Robbins (1985), Burnetas and Katehakis (1996),
Auer, Cesa-Bianchi and Fischer UCB1 (2002), and many others

# Optimism in the Face of Uncertainty

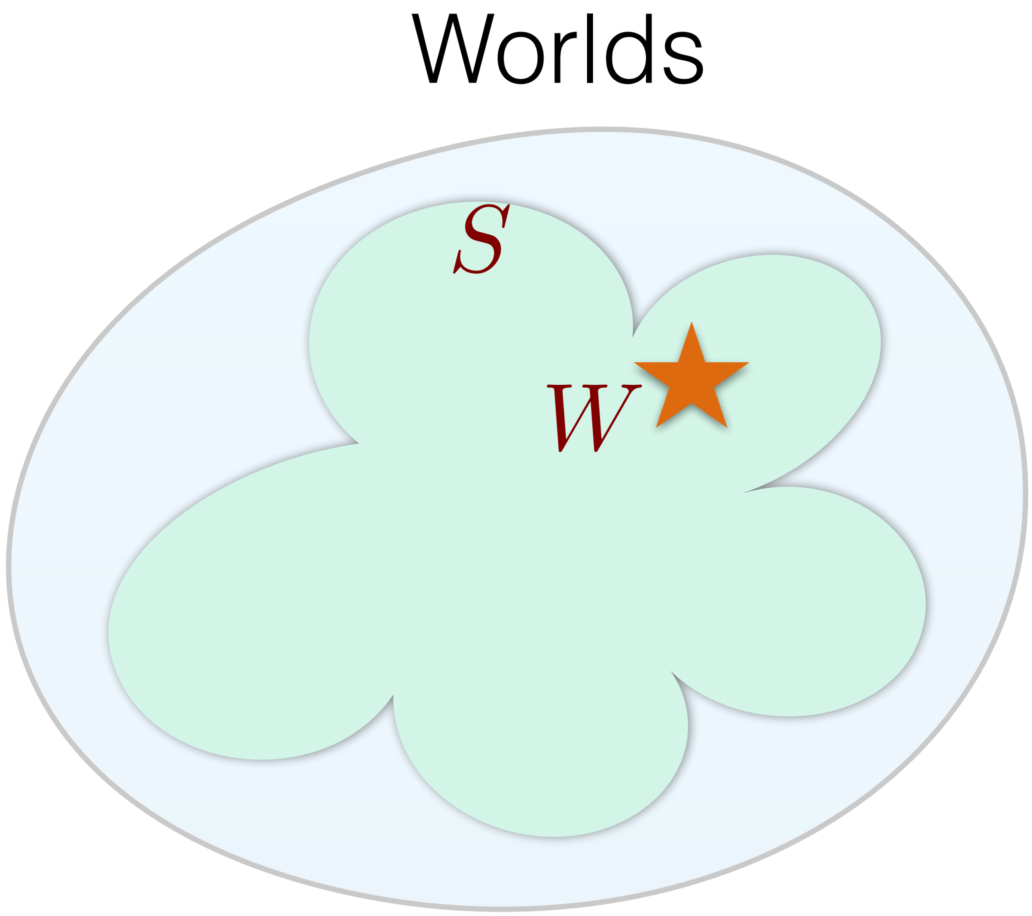
Repeat:

1. Find the set ***S*** of likely "worlds" given the observations so far
2. Find the world ***W*** in ***S*** with the maximum payoff:

$$W = \underset{W \in S}{\operatorname{argmax}}\, J(W)$$

3. Find the optimal policy for this world:

$$\pi = \underset{\xi}{\operatorname{argmax}}\, J(W, \xi)$$




Lai and Robbins (1985), Burnetas and Katehakis (1996),
Auer, Cesa-Bianchi and Fischer UCB1 (2002), and many others

# Optimism in the Face of Uncertainty

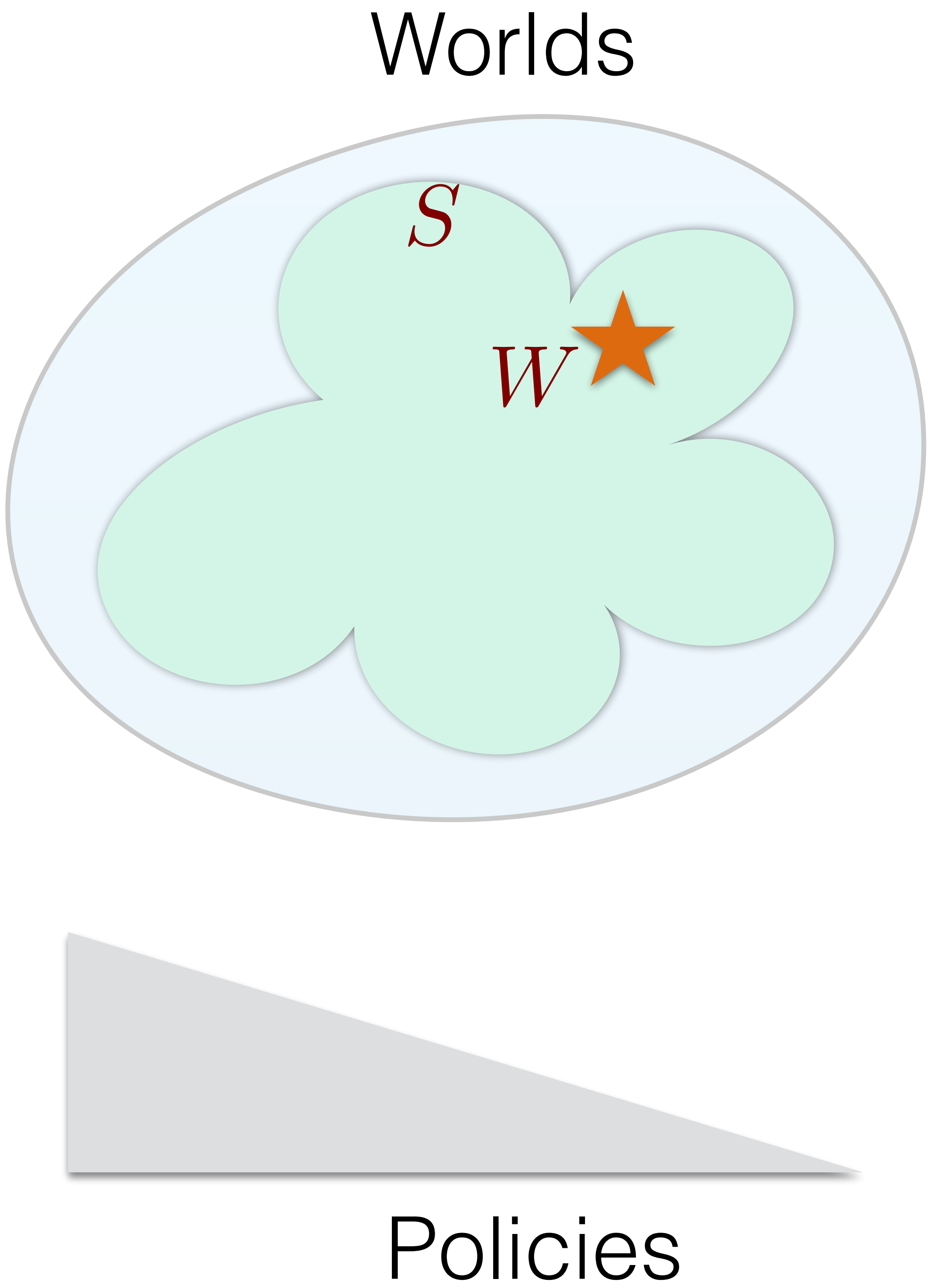
Repeat:

1. Find the set ***S*** of likely “worlds” given the observations so far
2. Find the world ***W*** in ***S*** with the maximum payoff:

$$W = \operatorname*{argmax}_{W \in S} J(W)$$

3. Find the optimal policy for this world:

$$\pi = \operatorname*{argmax}_{\xi} J(W, \xi)$$




Lai and Robbins (1985), Burnetas and Katehakis (1996),
Auer, Cesa-Bianchi and Fischer UCB1 (2002), and many others

# Optimism in the Face of Uncertainty

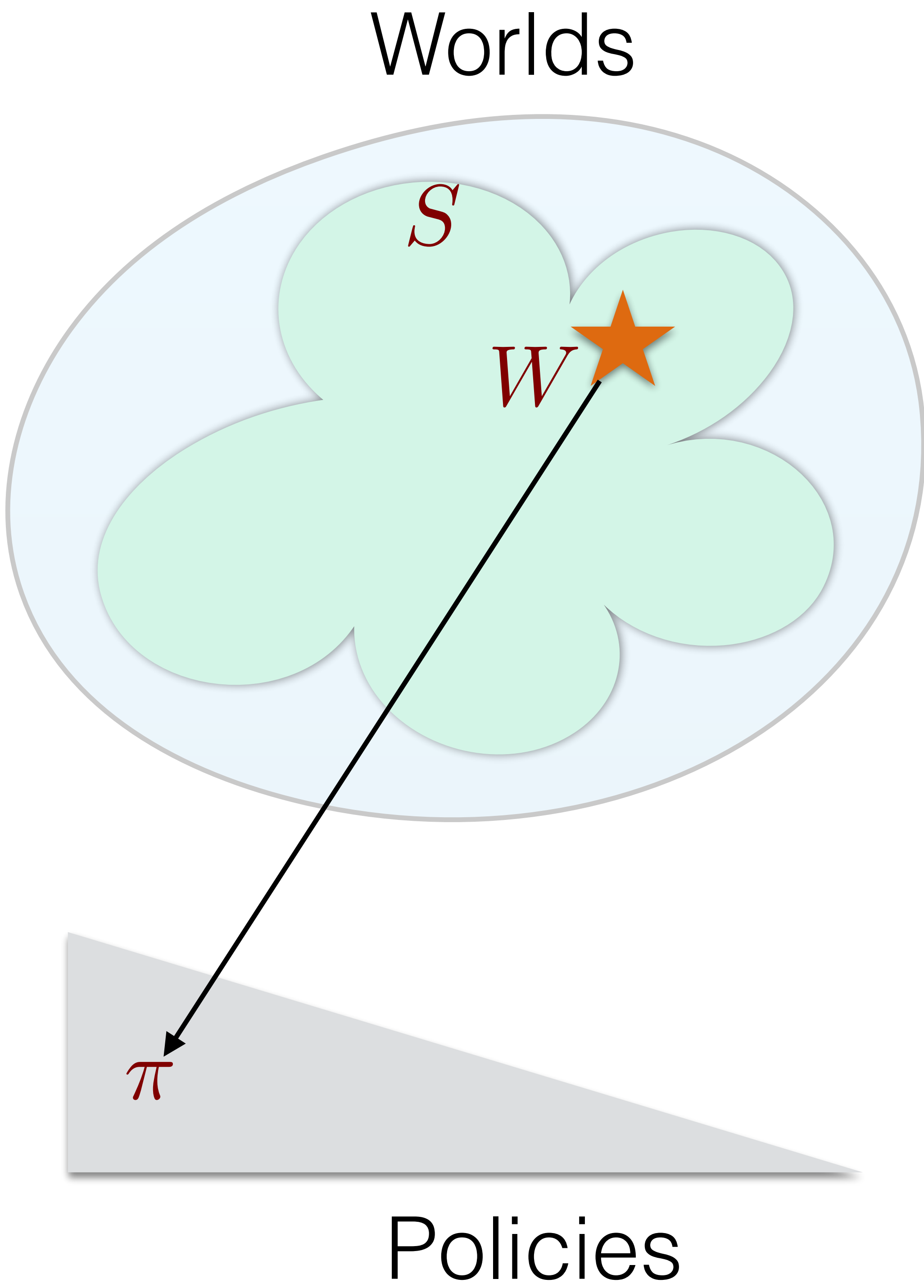
Repeat:

1. Find the set ***S*** of likely "worlds" given the observations so far
2. Find the world ***W*** in ***S*** with the maximum payoff:

$$W = \operatorname*{argmax}_{W \in S} J(W)$$

3. Find the optimal policy for this world:

$$\pi = \operatorname*{argmax}_{\xi} J(W, \xi)$$




Lai and Robbins (1985), Burnetas and Katehakis (1996),
Auer, Cesa-Bianchi and Fischer UCB1 (2002), and many others

# Optimism in the Face of Uncertainty

OFU

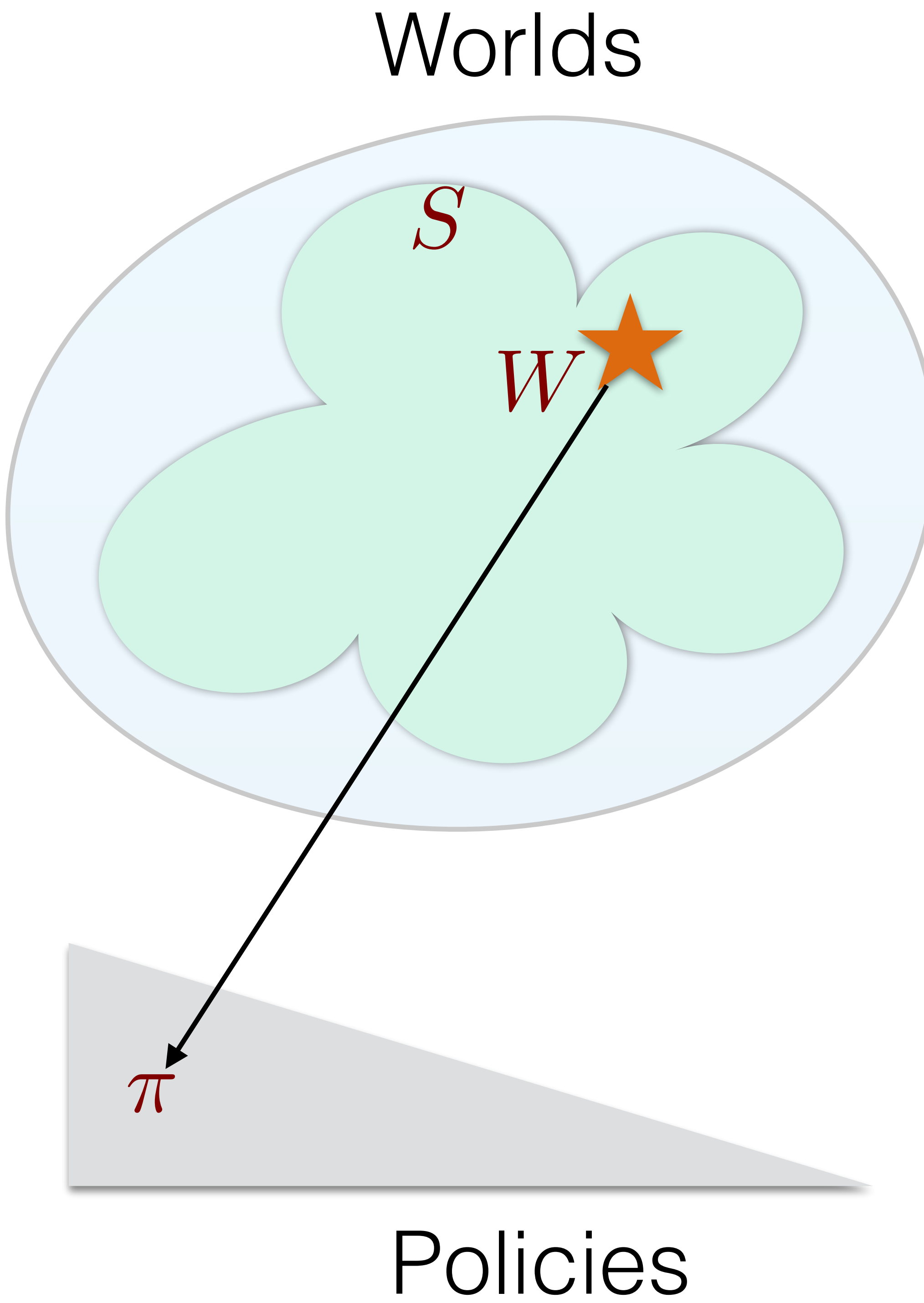
Repeat:

1. Find the set ***S*** of likely "worlds" given the observations so far
2. Find the world ***W*** in ***S*** with the maximum payoff:

$$W = \operatorname*{argmax}_{W \in S} J(W)$$

3. Find the optimal policy for this world:

$$\pi = \operatorname*{argmax}_{\xi} J(W, \xi)$$




Lai and Robbins (1985), Burnetas and Katehakis (1996),
Auer, Cesa-Bianchi and Fischer UCB1 (2002), and many others

# Optimism in the Face of Uncertainty

OFU

Repeat:

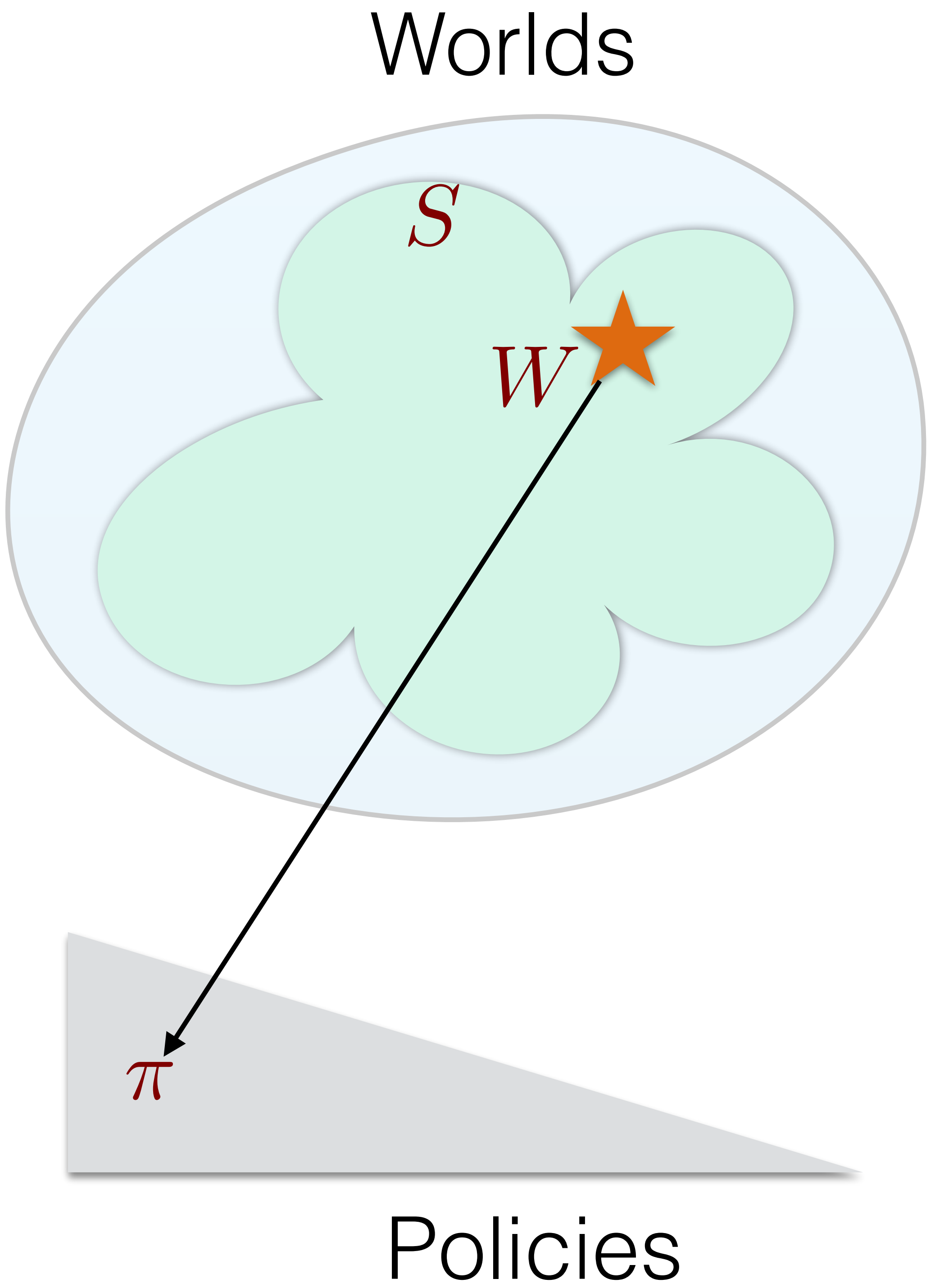
1. Find the set ***S*** of likely “worlds” given the observations so far
2. Find the world ***W*** in ***S*** with the maximum payoff:

$$W = \operatorname*{argmax}_{W \in S} J(W)$$

3. Find the optimal policy for this world:

$$\pi = \operatorname*{argmax}_{\xi} J(W, \xi)$$

4. Use this policy until ***S*** significantly shrinks




Lai and Robbins (1985), Burnetas and Katehakis (1996),
Auer, Cesa-Bianchi and Fischer UCB1 (2002), and many others

[Jaksch-Ortner-Auer,’10]

# How good is OFU?

[Jaksch-Ortner-Auer,'10]

# How good is OFU?

***S*** states, ***A*** actions, rewards in [0,1].

[Jaksch-Ortner-Auer,'10]

# How good is OFU?

***S*** states, ***A*** actions, rewards in [0,1].

OFU for finite problems: UCRL2

[Jaksch-Ortner-Auer,'10]

# How good is OFU?

***S*** states, ***A*** actions, rewards in [0,1].

**<u>Definition:</u>** Diameter := maximum of best travel times between pairs of states. River swim: ***D = S***

OFU for finite problems: UCRL2

[Jaksch-Ortner-Auer,'10]

# How good is OFU?

***S*** states, ***A*** actions, rewards in [0,1].

**Definition:** Diameter := maximum of best travel times between pairs of states. River swim: ***D = S***

- **Theorem:** The regret of an OFU learner satisfies

$$R_T = \tilde{O}(DS\sqrt{AT})$$

OFU for finite problems: UCRL2

[Jaksch-Ortner-Auer,'10]

# How good is OFU?

***S*** states, ***A*** actions, rewards in [0,1].

<u>**Definition:**</u> Diameter := maximum of best travel times between pairs of states. River swim: ***D = S***

- <u>**Theorem:**</u> The regret of an OFU learner satisfies

$$R_T = \tilde{O}(DS\sqrt{AT})$$

- <u>**Theorem:**</u> For any algorithm,

$$R_T = \Omega(\sqrt{DSAT})$$

OFU for finite problems: UCRL2

# Posterior Sampling Reinforcement Learning

[Thompson, 1933(!), Strens '00]

# Posterior Sampling Reinforcement Learning

A Bayesian start:

[Thompson, 1933(!), Strens '00]

# Posterior Sampling Reinforcement Learning

A Bayesian start:

- Prior over the worlds

[Thompson, 1933(!), Strens '00]

# Posterior Sampling Reinforcement Learning

A Bayesian start:

- Prior over the worlds
- Likelihood model

[Thompson, 1933(!), Strens '00]

# Posterior Sampling Reinforcement Learning

A Bayesian start:

- Prior over the worlds
- Likelihood model
- Posterior: $p(W|D) \propto p_W(W)p(D|W)$

[Thompson, 1933(!), Strens '00]

# Posterior Sampling Reinforcement Learning

A Bayesian start:

- Prior over the worlds
- Likelihood model
- Posterior: $p(W|D) \propto p_W(W)p(D|W)$

Repeat:

[Thompson, 1933(!), Strens '00]

# Posterior Sampling Reinforcement Learning

A Bayesian start:

- Prior over the worlds
- Likelihood model
- Posterior: $p(W|D) \propto p_W(W)p(D|W)$

Repeat:

1. Sample a world ***W*** from the posterior:

$$W \sim P(W = \cdot|D)$$

[Thompson, 1933(!), Strens '00]

# Posterior Sampling Reinforcement Learning

A Bayesian start:

- Prior over the worlds
- Likelihood model
- Posterior: $p(W|D) \propto p_W(W)p(D|W)$

Repeat:

1. Sample a world ***W*** from the posterior:

$$W \sim P(W = \cdot|D)$$

Worlds

[Thompson, 1933(!), Strens '00]

# Posterior Sampling Reinforcement Learning

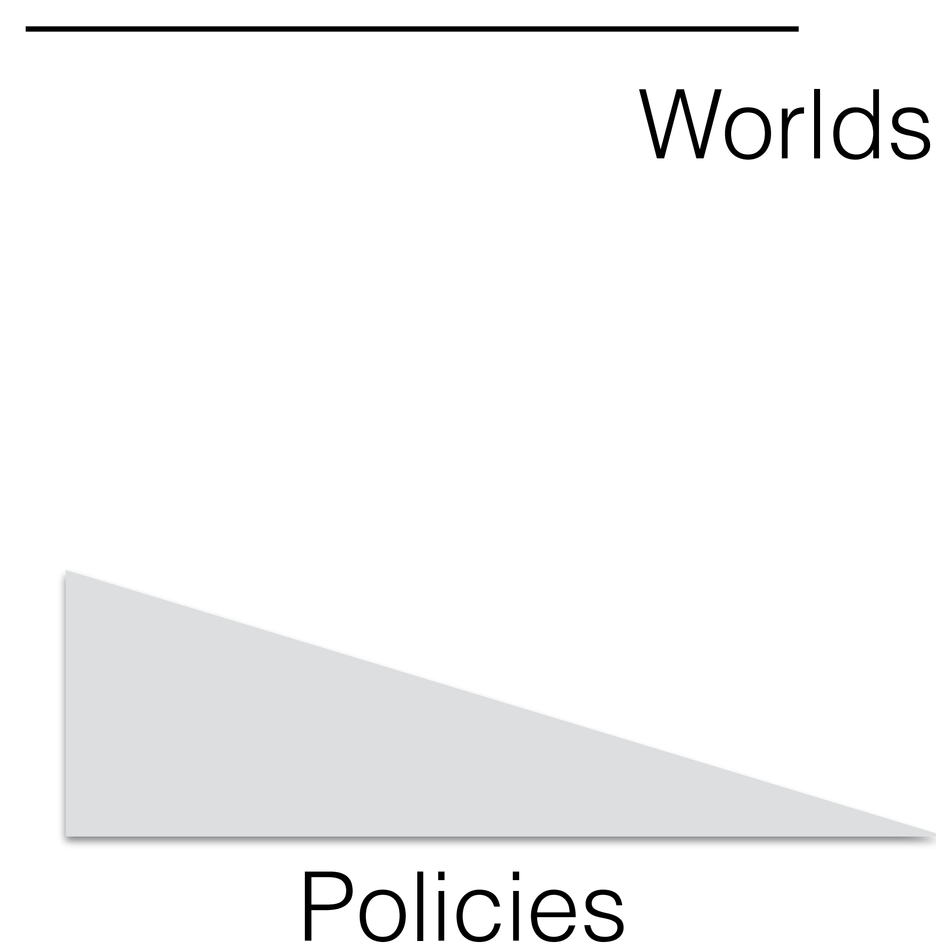
A Bayesian start:

- Prior over the worlds
- Likelihood model
- Posterior: $p(W|D) \propto p_W(W)p(D|W)$

Repeat:

1. Sample a world ***W*** from the posterior:

$$W \sim P(W = \cdot|D)$$




[Thompson, 1933(!), Strens '00]

# Posterior Sampling Reinforcement Learning

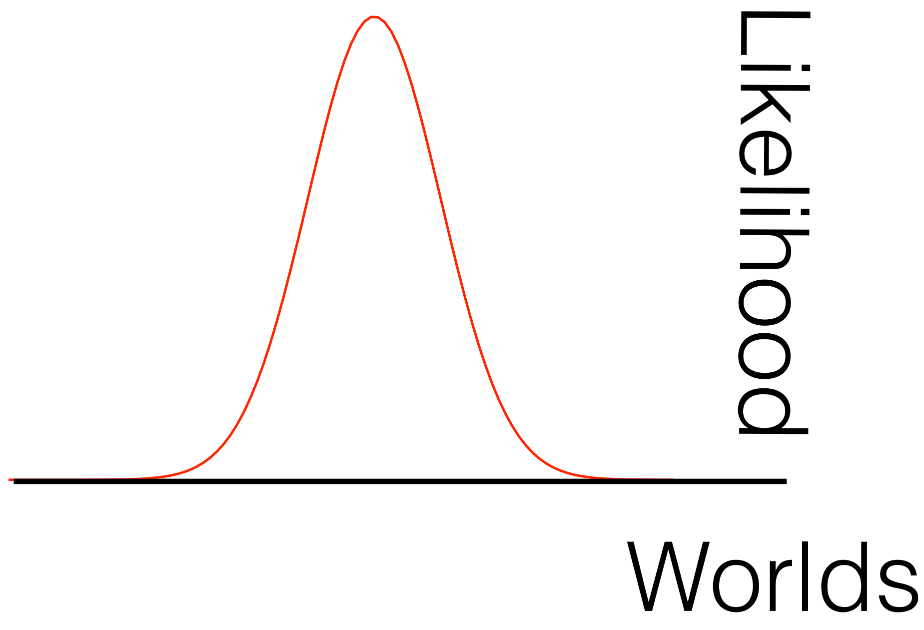
A Bayesian start:

- Prior over the worlds
- Likelihood model
- Posterior: $p(W|D) \propto p_W(W)p(D|W)$

Repeat:

1. Sample a world ***W*** from the posterior:

$$W \sim P(W = \cdot|D)$$





[Thompson, 1933(!), Strens '00]

# Posterior Sampling Reinforcement Learning

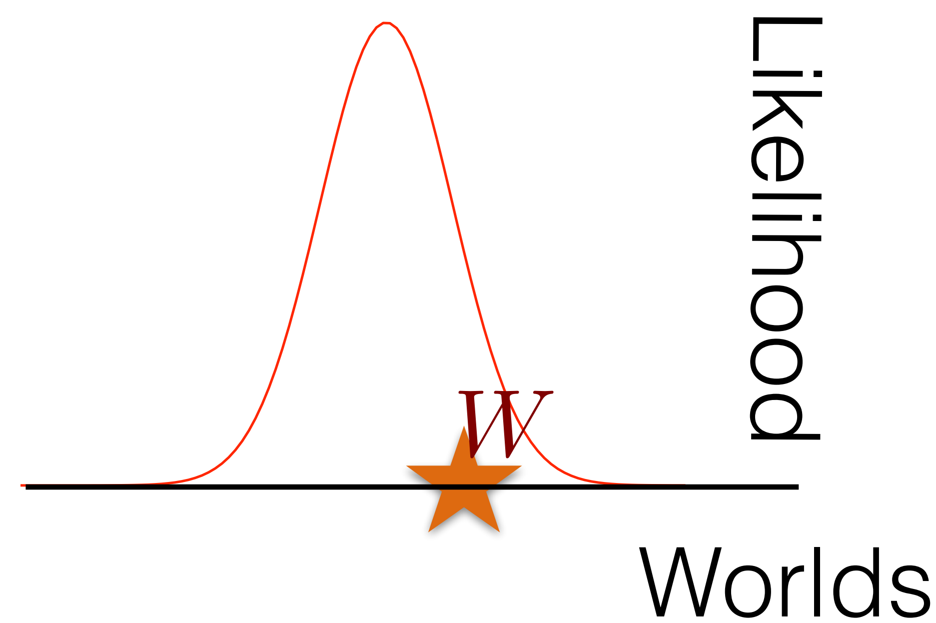
A Bayesian start:

- Prior over the worlds
- Likelihood model
- Posterior: $p(W|D) \propto p_W(W)p(D|W)$

Repeat:

1. Sample a world ***W*** from the posterior:

$$W \sim P(W = \cdot|D)$$





[Thompson, 1933(!), Strens ’00]

# Posterior Sampling Reinforcement Learning

A Bayesian start:

- Prior over the worlds
- Likelihood model
- Posterior: $p(W|D) \propto p_W(W)p(D|W)$

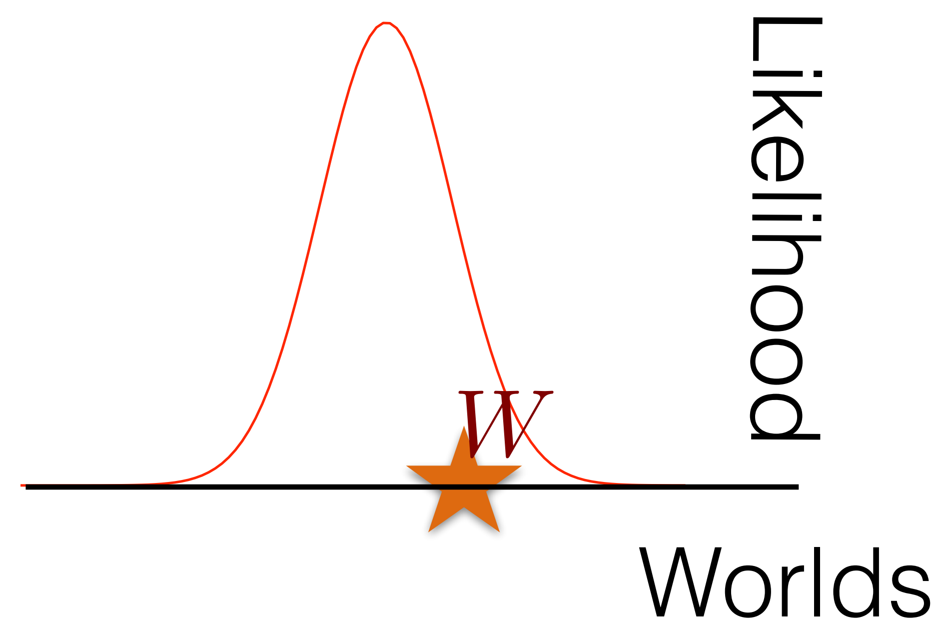
Repeat:

1. Sample a world ***W*** from the posterior:

$$W \sim P(W = \cdot|D)$$

2. Find the optimal policy for this world:

$$\pi = \operatorname*{argmax}_{\xi} J(W, \xi)$$





[Thompson, 1933(!), Strens '00]

# Posterior Sampling Reinforcement Learning

A Bayesian start:

- Prior over the worlds
- Likelihood model
- Posterior: $p(W|D) \propto p_W(W)p(D|W)$

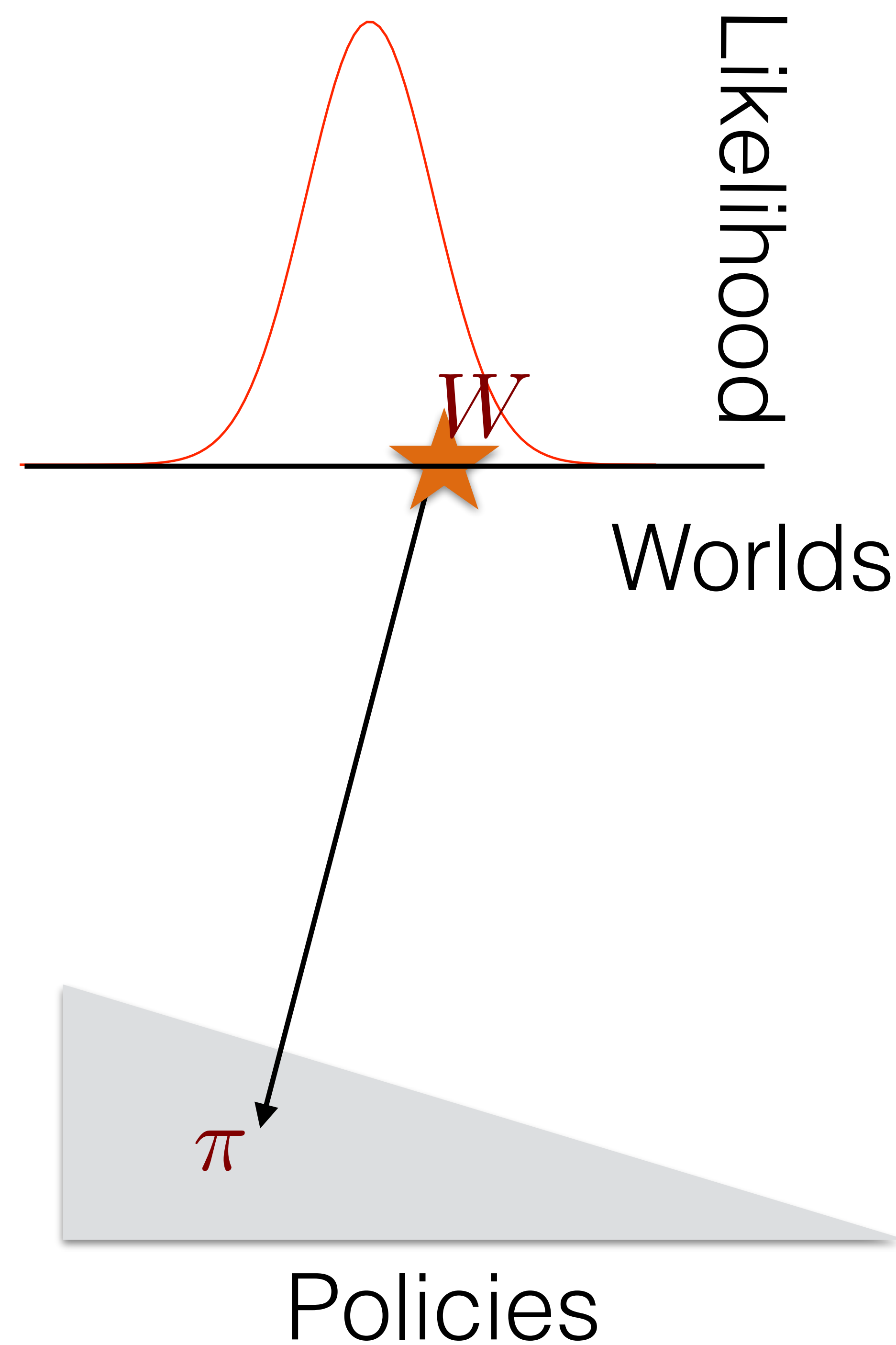
Repeat:

1. Sample a world ***W*** from the posterior:

$$W \sim P(W = \cdot|D)$$

2. Find the optimal policy for this world:

$$\pi = \underset{\xi}{\operatorname{argmax}} J(W, \xi)$$




[Thompson, 1933(!), Strens '00]

# Posterior Sampling Reinforcement Learning

PSRL

A Bayesian start:

- Prior over the worlds
- Likelihood model
- Posterior: $p(W|D) \propto p_W(W)p(D|W)$

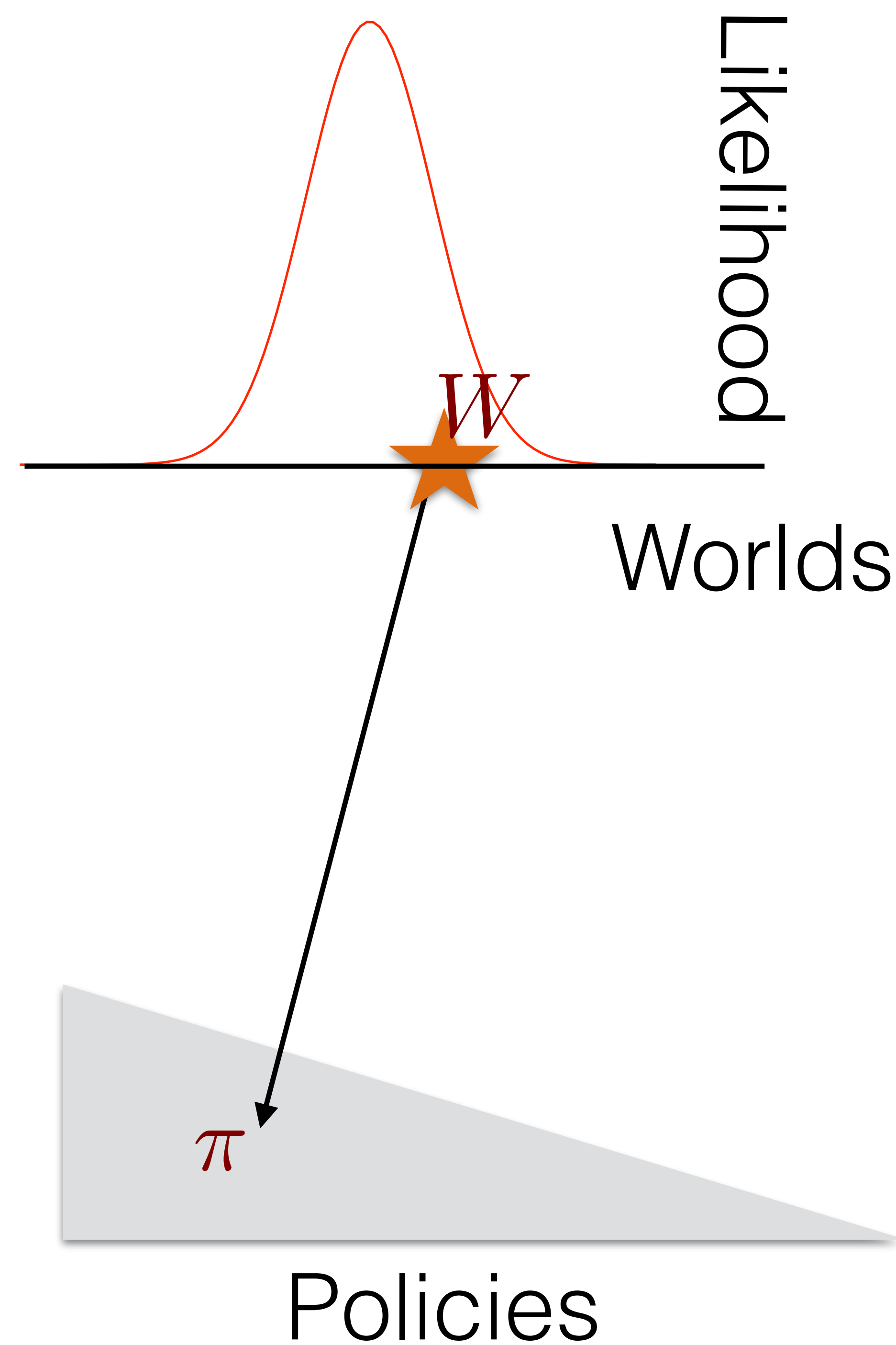
Repeat:

1. Sample a world ***W*** from the posterior:

$$W \sim P(W = \cdot|D)$$

2. Find the optimal policy for this world:

$$\pi = \underset{\xi}{\operatorname{argmax}}\, J(W, \xi)$$




[Thompson, 1933(!), Strens '00]

# Posterior Sampling Reinforcement Learning

PSRL

A Bayesian start:

- Prior over the worlds
- Likelihood model
- Posterior: $p(W|D) \propto p_W(W)p(D|W)$

Repeat:

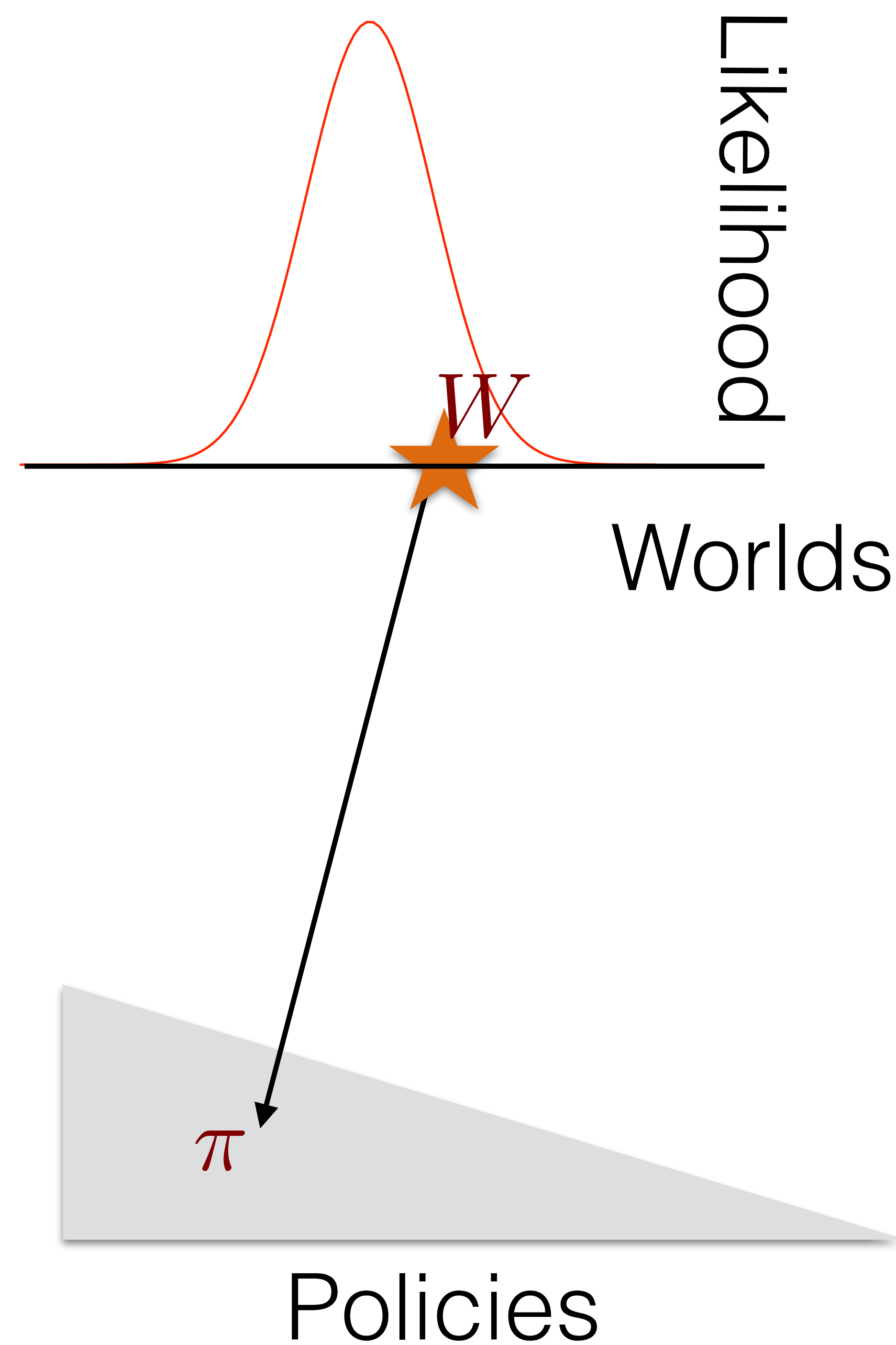
1. Sample a world ***W*** from the posterior:

$$W \sim P(W = \cdot|D)$$

2. Find the optimal policy for this world:

$$\pi = \operatorname*{argmax}_{\xi} J(W, \xi)$$

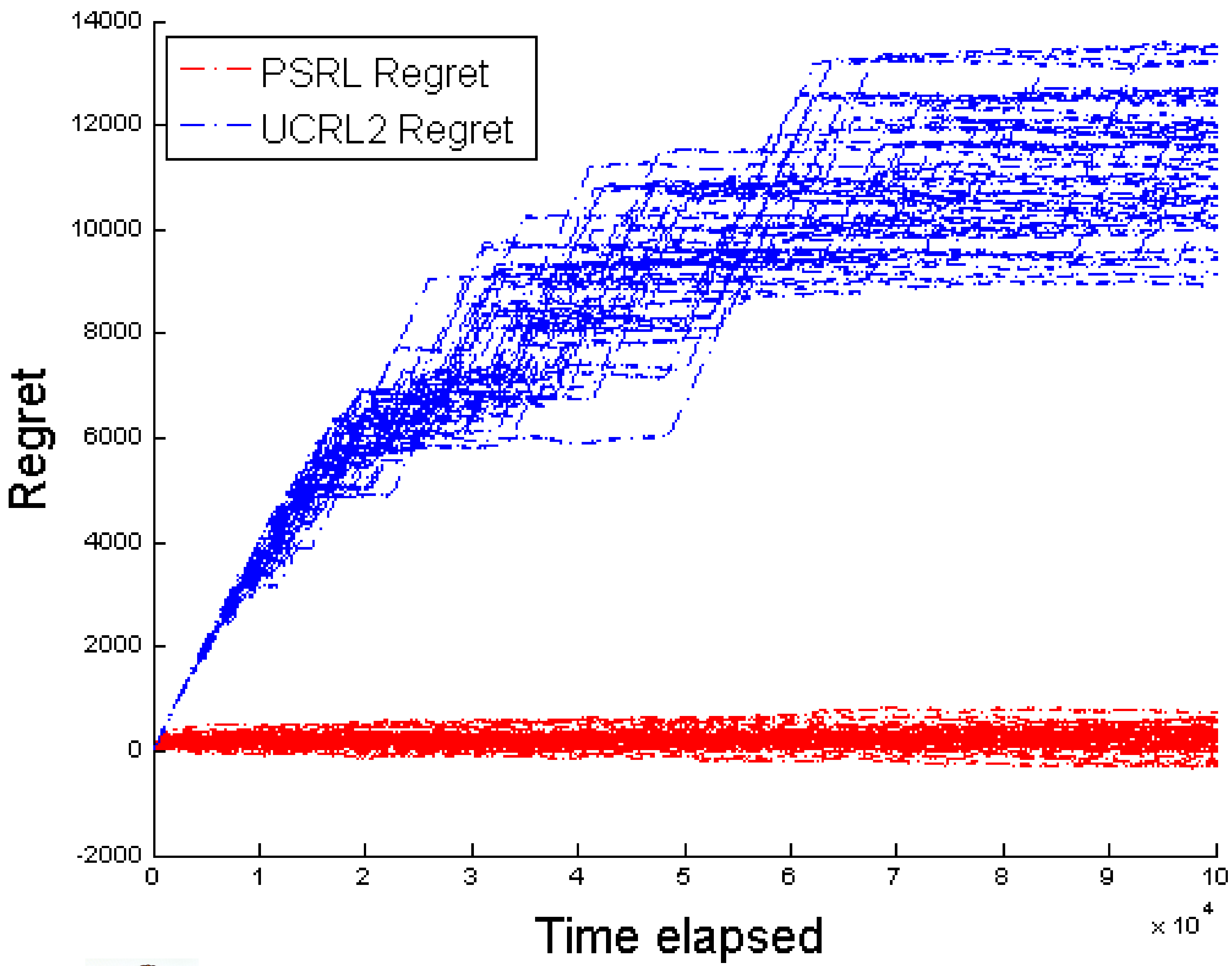
3. Use this policy a “little while”

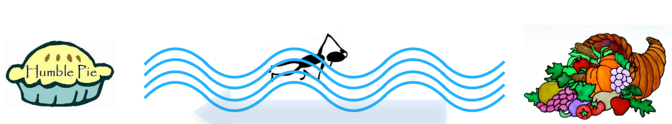



[Thompson, 1933(!), Strens ’00]

[Osband,Van Roy, Russo '13]

# Beating a near-optimal algorithm



# Scaling up

# Scaling up

- **Large** state-action spaces:
  need to **generalize** across states and actions

# Scaling up

- **Large** state-action spaces:
  need to **generalize** across states and actions

- Model based approach:

# Scaling up

- **Large** state-action spaces:
  need to **generalize** across states and actions

- Model based approach:

$$x_{t+1} = f(x_t, a_t, \theta_*, z_{t+1})$$

# Scaling up

- **Large** state-action spaces:
  need to **generalize** across states and actions

- Model based approach:

$$x_{t+1} = f(x_t, a_t, \theta_*, z_{t+1})$$

next
state

# Scaling up

- **Large** state-action spaces:
  need to **generalize** across states and actions

- Model based approach:

$$x_{t+1} = f(x_t, a_t, \theta_*, z_{t+1})$$

next state

current state

# Scaling up

- **Large** state-action spaces:
  need to **generalize** across states and actions

- Model based approach:

$$x_{t+1} = f(x_t, a_t, \theta_*, z_{t+1})$$

next state

current state

action

# Scaling up

- **Large** state-action spaces:
  need to **generalize** across states and actions

- Model based approach:

$$x_{t+1} = f(x_t, a_t, \theta_*, z_{t+1})$$

next state

current state

action

unknown parameter

# Scaling up

- **Large** state-action spaces:
  need to **generalize** across states and actions

- Model based approach:

$$x_{t+1} = f(x_t, a_t, \theta_*, z_{t+1})$$

next state

current state

action

unknown parameter

noise

# Linear Quadratic Regulation

# Linear Quadratic Regulation

$$x_{t+1} = Ax_t + Ba_t + z_{t+1}$$

$$c_{t+1} = x_t^\top Q x_t + a_t^\top R a_t$$

# Linear Quadratic Regulation

$$x_{t+1} = Ax_t + Ba_t + z_{t+1}$$

$$c_{t+1} = x_t^\top Q x_t + a_t^\top R a_t$$

$\theta_* = (A, B)$
is unknown

# Linear Quadratic Regulation

$$x_{t+1} = Ax_t + Ba_t + z_{t+1}$$

$$c_{t+1} = x_t^\top Q x_t + a_t^\top R a_t$$

$\theta_* = (A, B)$
is unknown

- **Theorem [Abbasi-Sz 2011]**: For reachable and controllable systems, the regret of OFU satisfies

# Linear Quadratic Regulation

$$x_{t+1} = Ax_t + Ba_t + z_{t+1}$$

$$c_{t+1} = x_t^\top Q x_t + a_t^\top R a_t$$

$\theta_* = (A, B)$
is unknown

- **Theorem [Abbasi-Sz 2011]**: For reachable and controllable systems, the regret of OFU satisfies

$$R_T = \tilde{O}(\sqrt{T})$$

# Linear Quadratic Regulation

$$x_{t+1} = Ax_t + Ba_t + z_{t+1}$$

$$c_{t+1} = x_t^\top Q x_t + a_t^\top R a_t$$

$\theta_* = (A, B)$
is unknown

- **Theorem [Abbasi-Sz 2011]**: For reachable and controllable systems, the regret of OFU satisfies

$$R_T = \tilde{O}(\sqrt{T})$$

- Key idea: Estimate the unknown parameter using l2 regularized least-squares, develop tight confidence sets

# Nonlinear systems?

# Nonlinear systems?

- Smoothness:

$$y = f(x, a, \theta, z), y' = f(x, a, \theta', z)$$
$$\Rightarrow$$
$$\mathbb{E}\left[\|y - y'\|\right] \leq \|\theta - \theta'\|_{M(x,a)}$$

# Nonlinear systems?

- Smoothness:

$$y = f(x, a, \theta, z), y' = f(x, a, \theta', z)$$

$$\Rightarrow$$

$$\mathbb{E}\left[\|y - y'\|\right] \leq \|\theta - \theta'\|_{M(x,a)}$$

- **Theorem [Abbasi-Sz]**: For smooth, “bounded” systems, if the posterior is “concentrating”, the Bayes regret of PSRL is bounded by

$$R_T = \tilde{O}(\sqrt{T})$$

# Nonlinear systems?

- Smoothness:

$$y = f(x, a, \theta, z), y' = f(x, a, \theta', z)$$

$$\Rightarrow$$

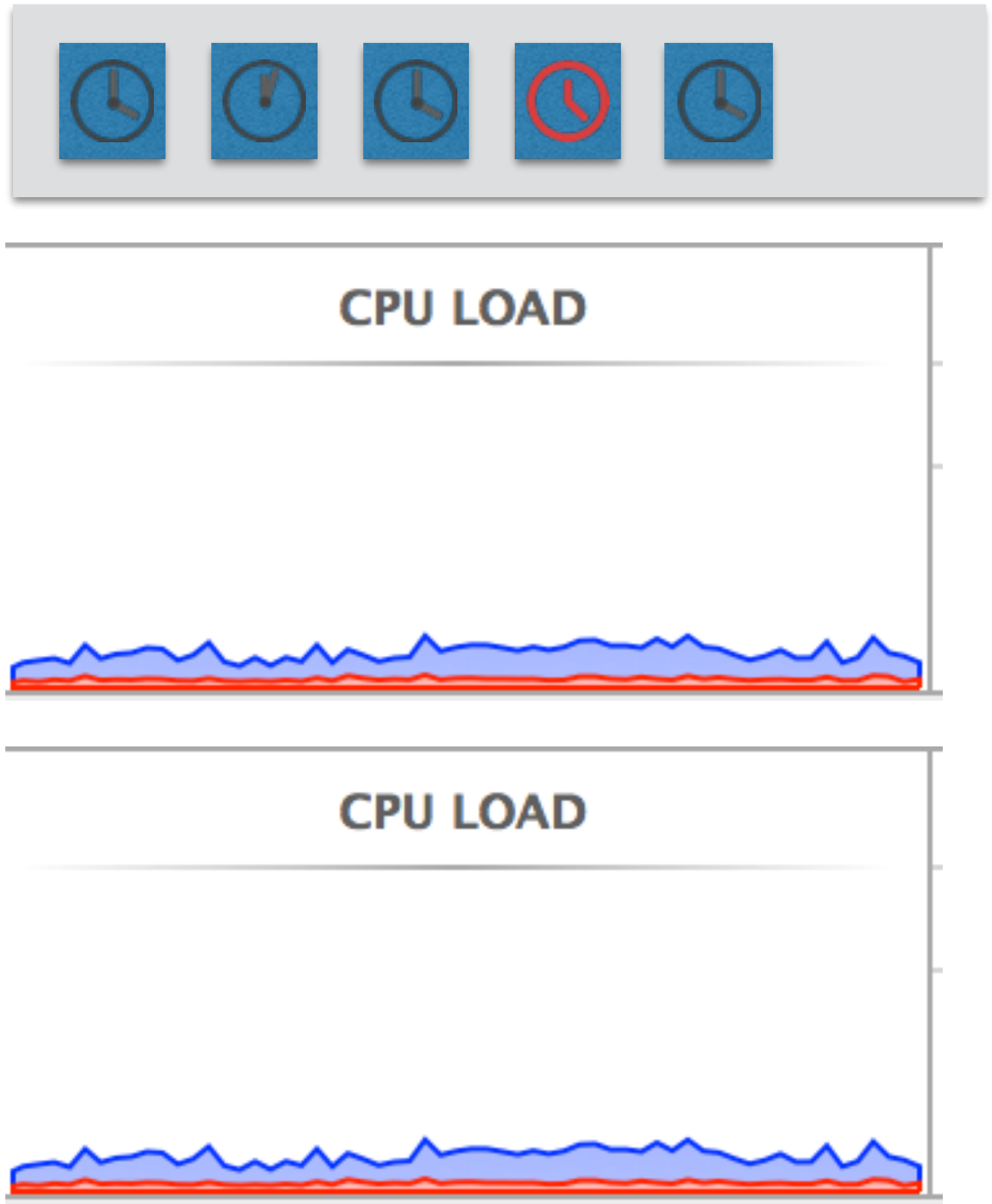
$$\mathbb{E}\left[\|y - y'\|\right] \leq \|\theta - \theta'\|_{M(x,a)}$$

- **Theorem [Abbasi-Sz]**: For smooth, “bounded” systems, if the posterior is “concentrating”, the Bayes regret of PSRL is bounded by

$$R_T = \tilde{O}(\sqrt{T})$$

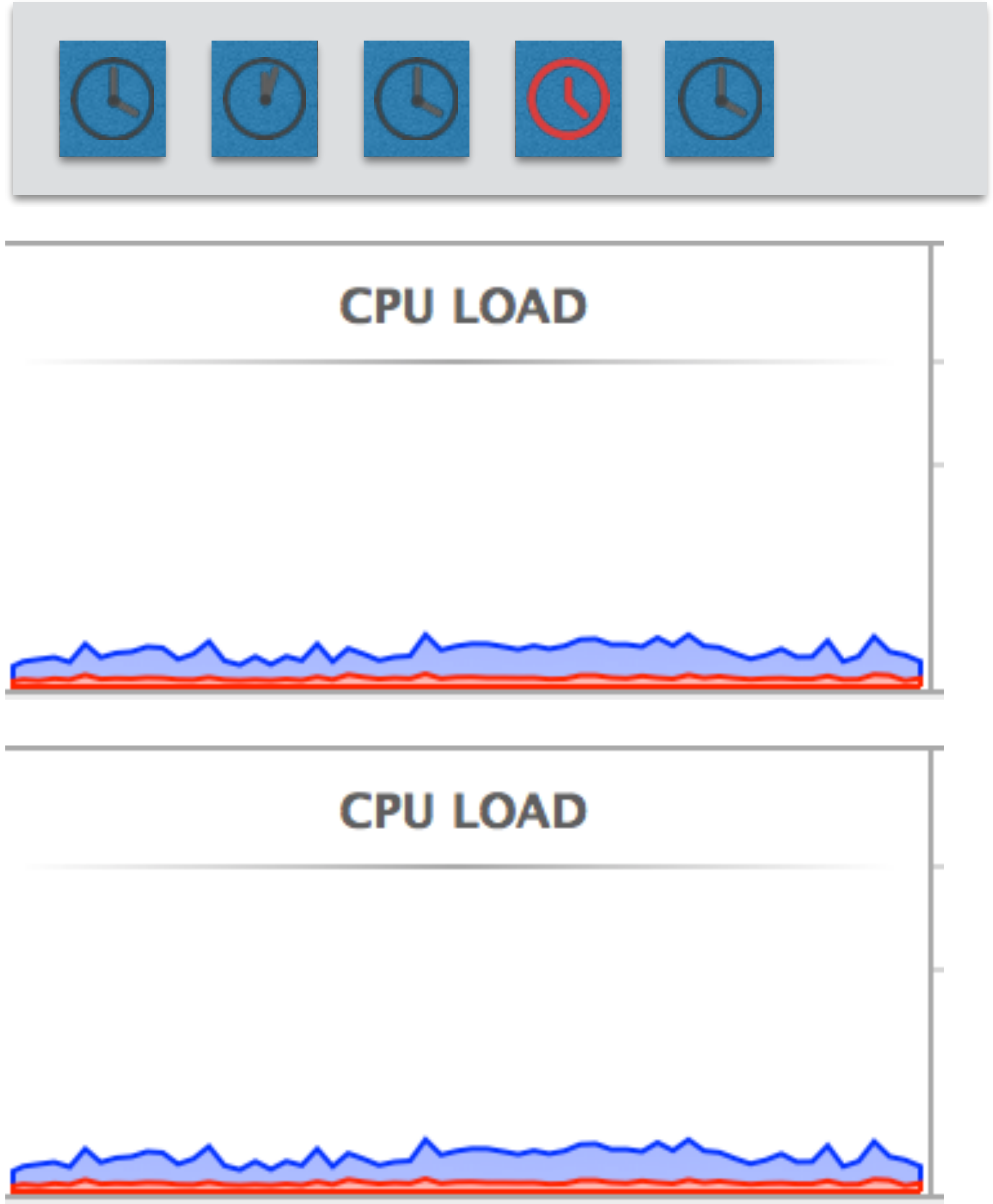
- Key idea: Use $M(x, a)$ to measure information.

# Web Server Control



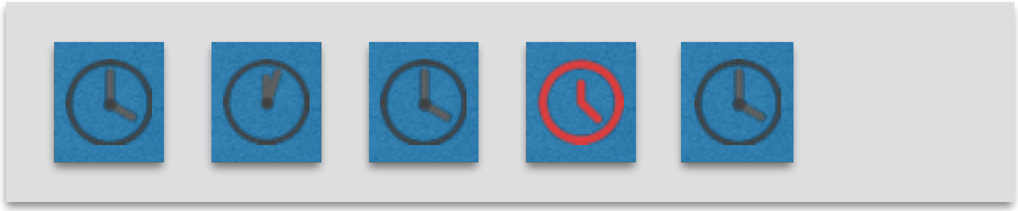
# Web Server Control

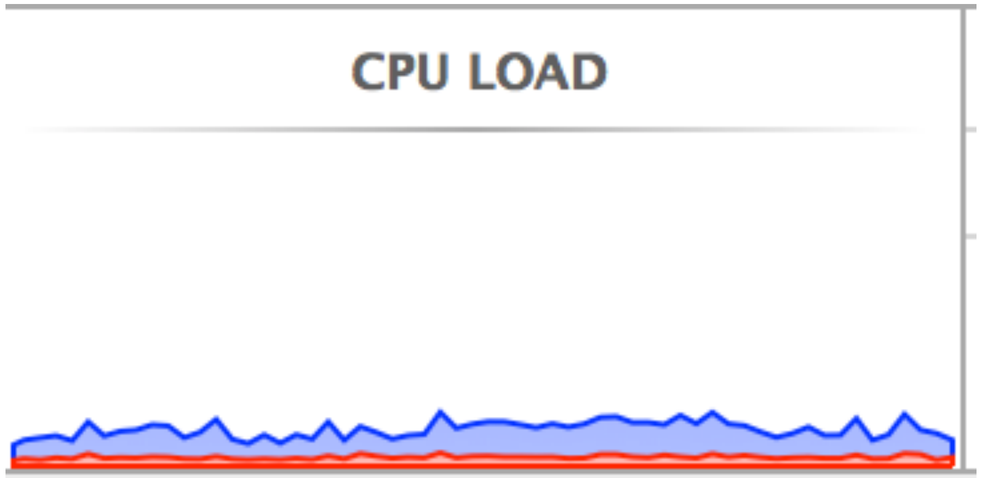
- Control variables:

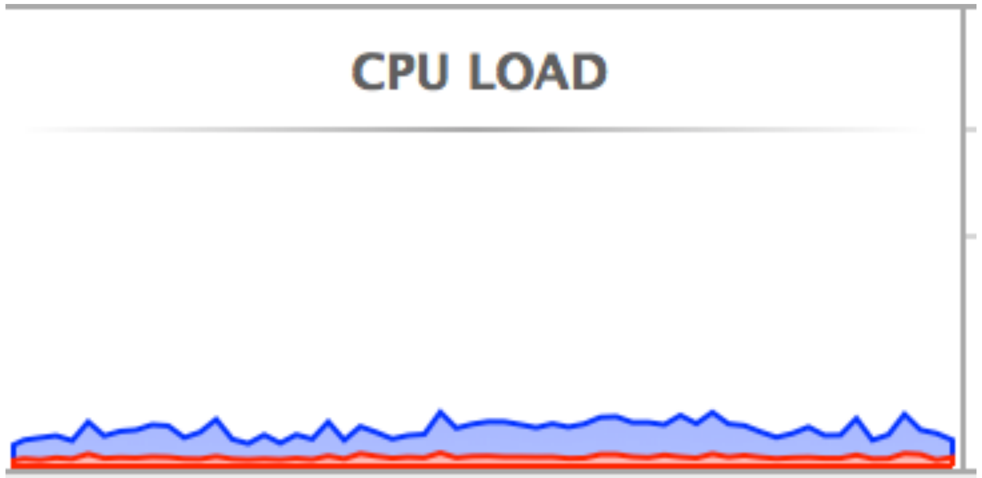


# Web Server Control

- Control variables:
  - How long to keep alive a connection without traffic on it

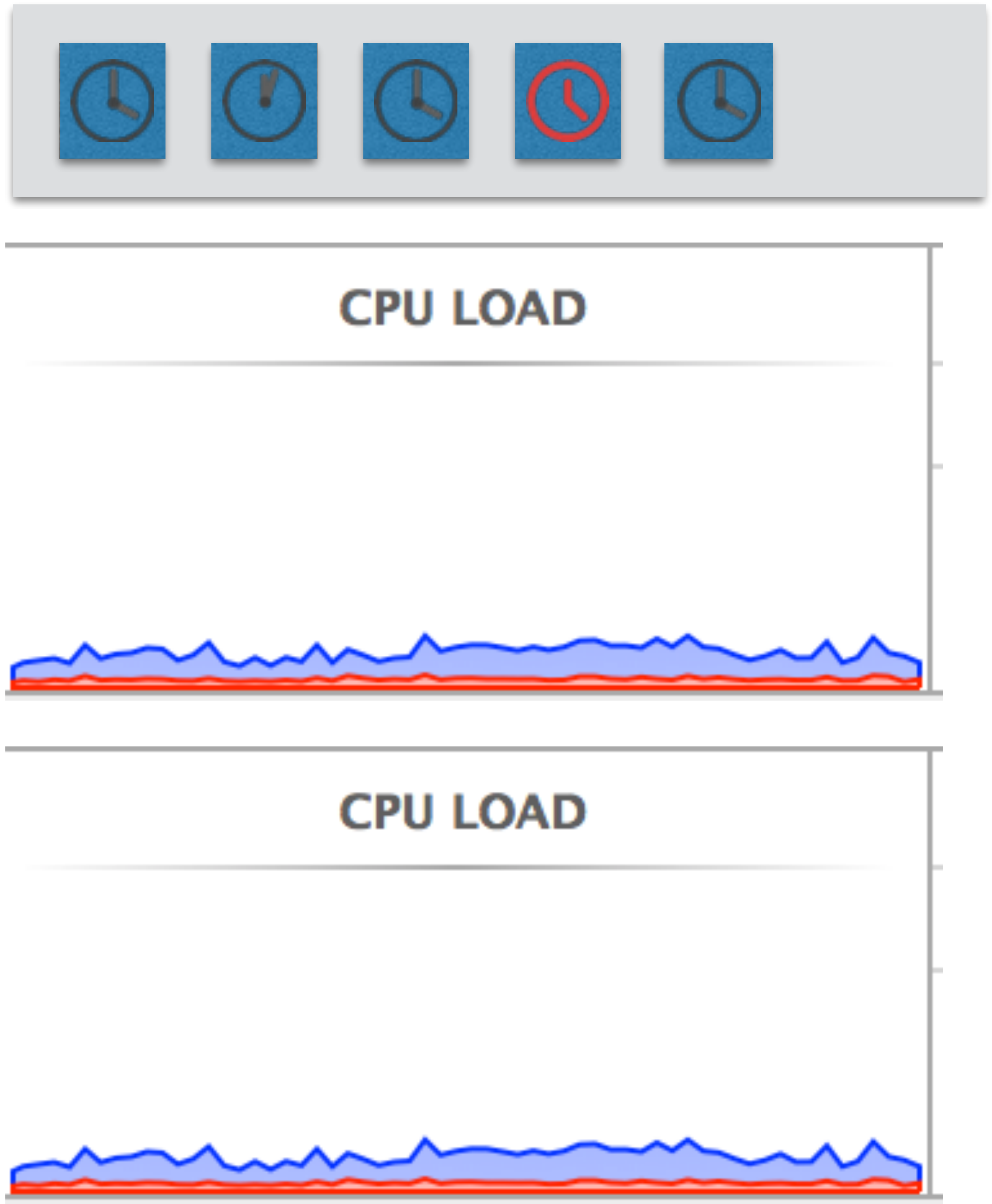




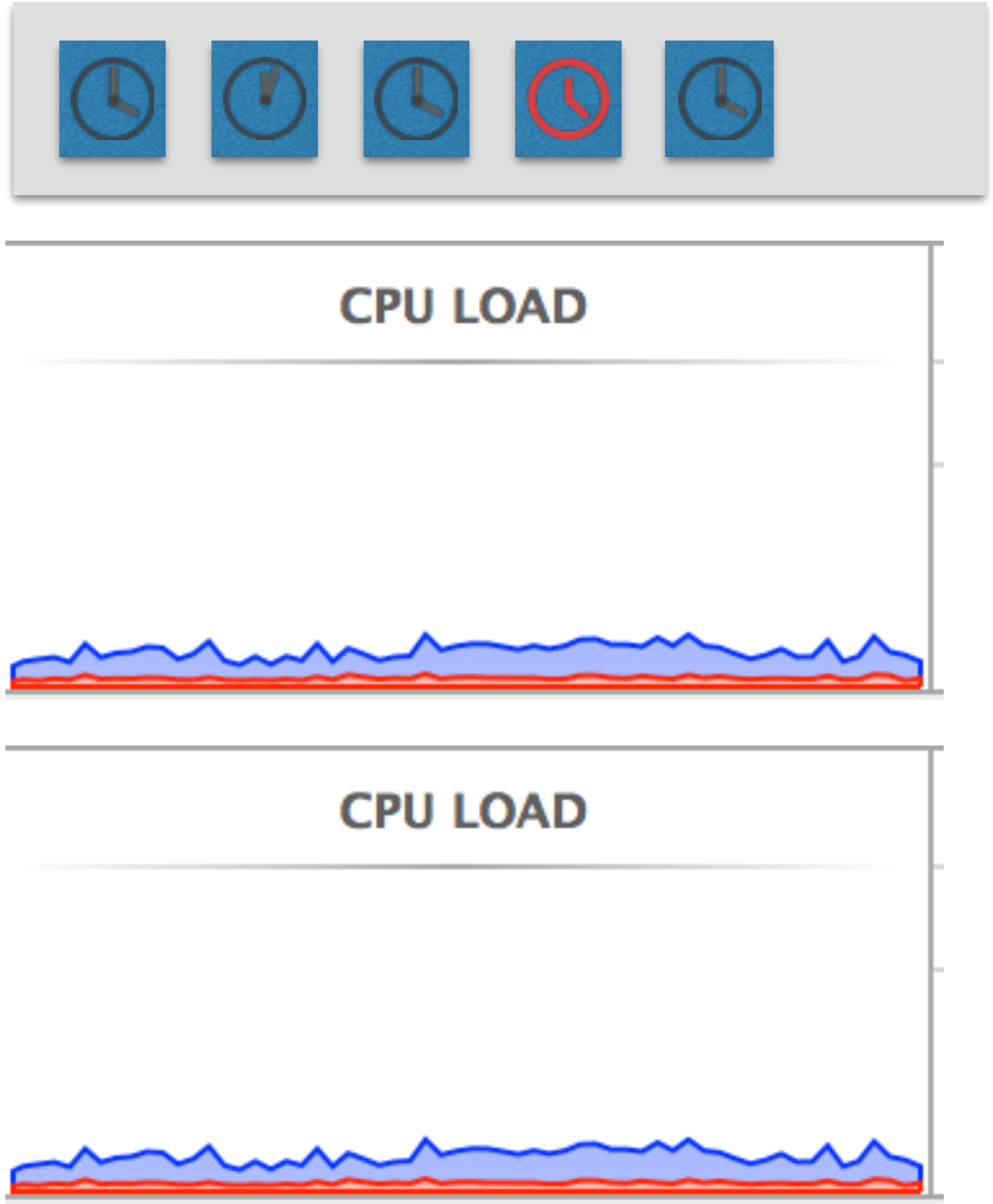
# Web Server Control

- Control variables:
    - How long to keep alive a connection without traffic on it
    - Maximum number of clients that can be served



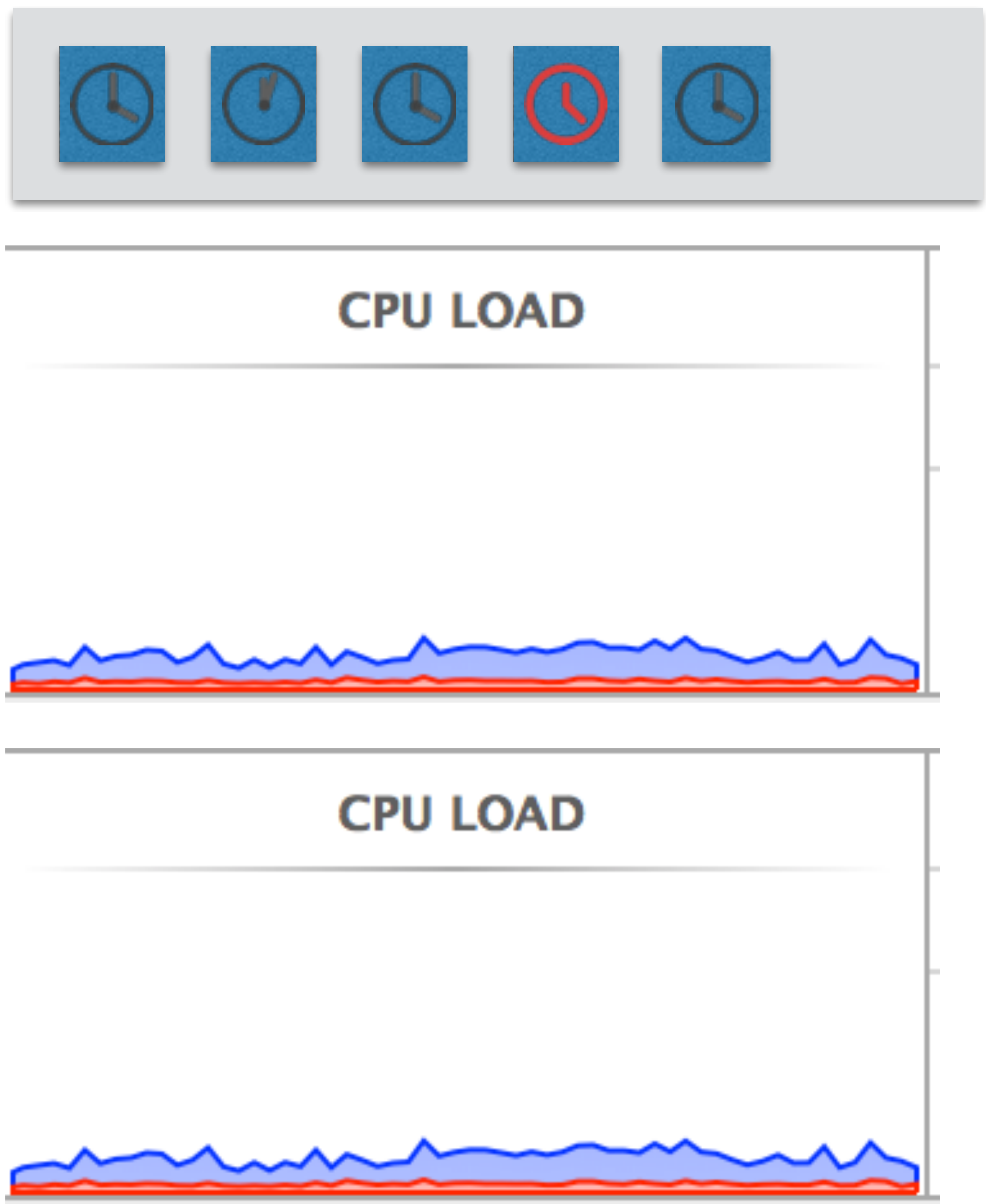
# Web Server Control

- Control variables:
  - How long to keep alive a connection without traffic on it
  - Maximum number of clients that can be served
- State variables:



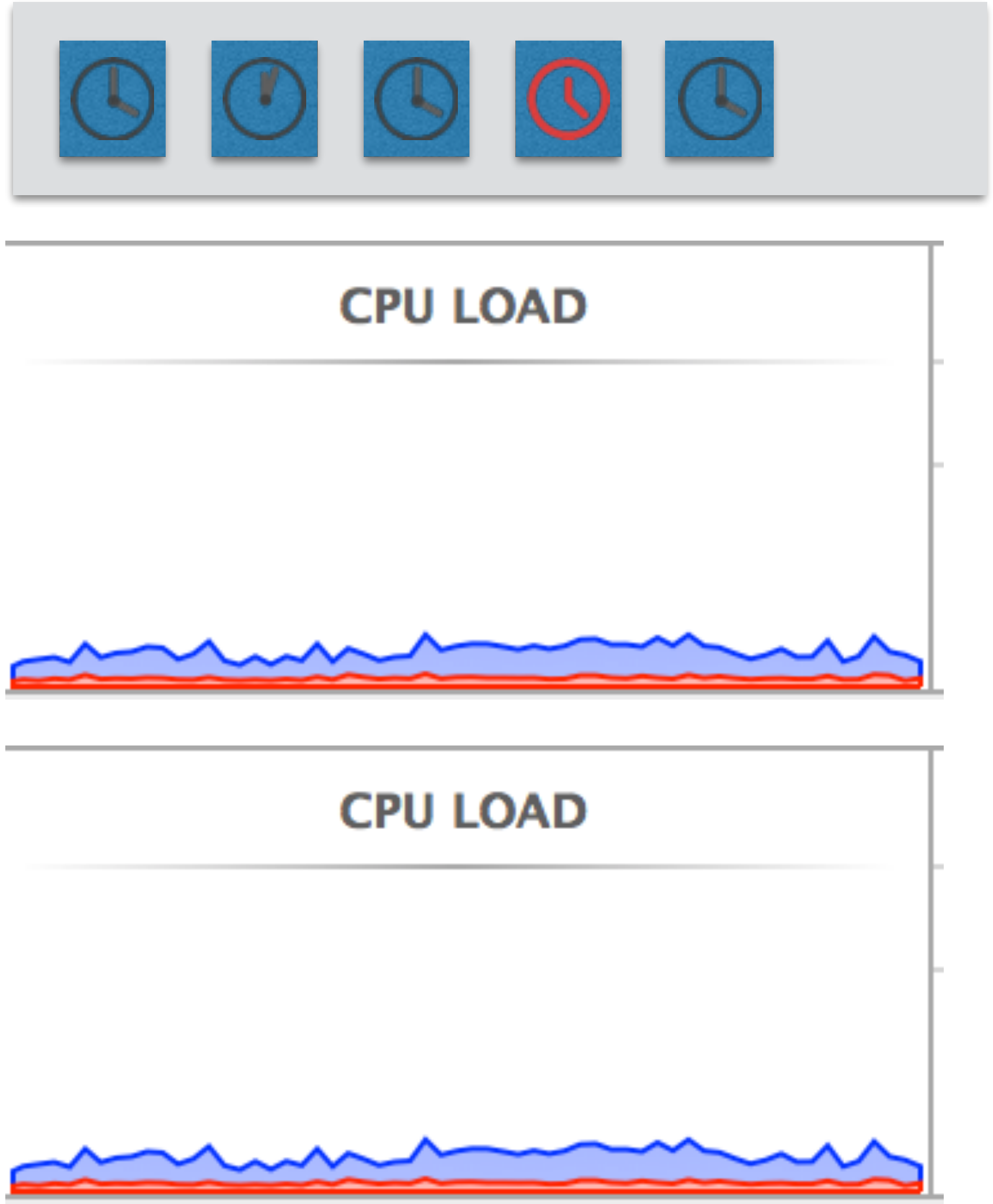
# Web Server Control

- Control variables:
  - How long to keep alive a connection without traffic on it
  - Maximum number of clients that can be served
- State variables:
  - Processor load relative to ideal processor load



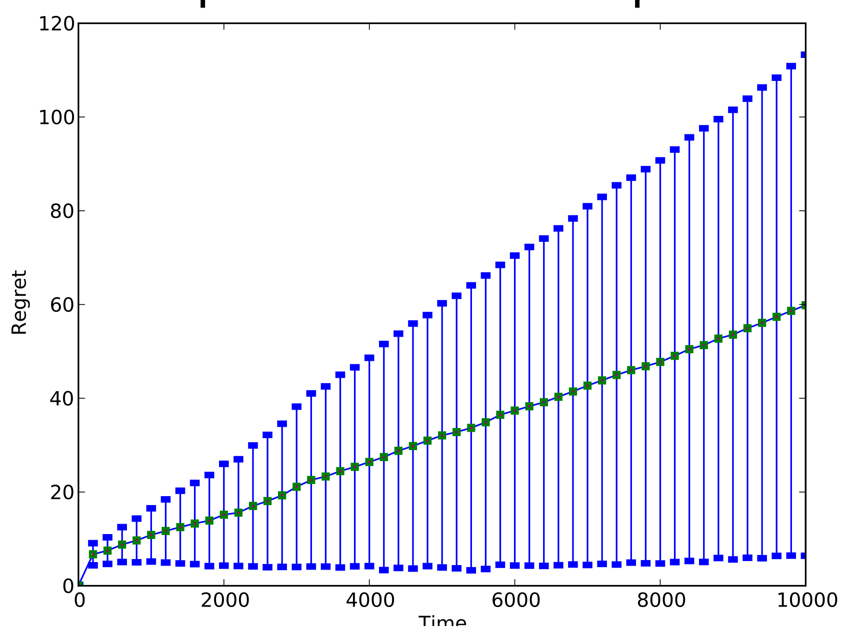
# Web Server Control

- Control variables:
  - How long to keep alive a connection without traffic on it
  - Maximum number of clients that can be served
- State variables:
  - Processor load relative to ideal processor load
  - Memory usage relative to ideal memory usage

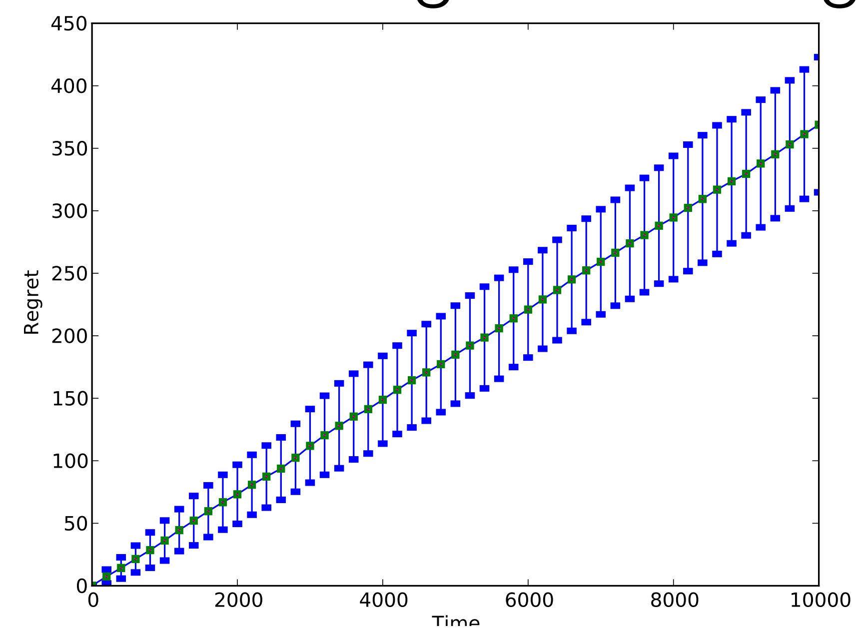


# Results

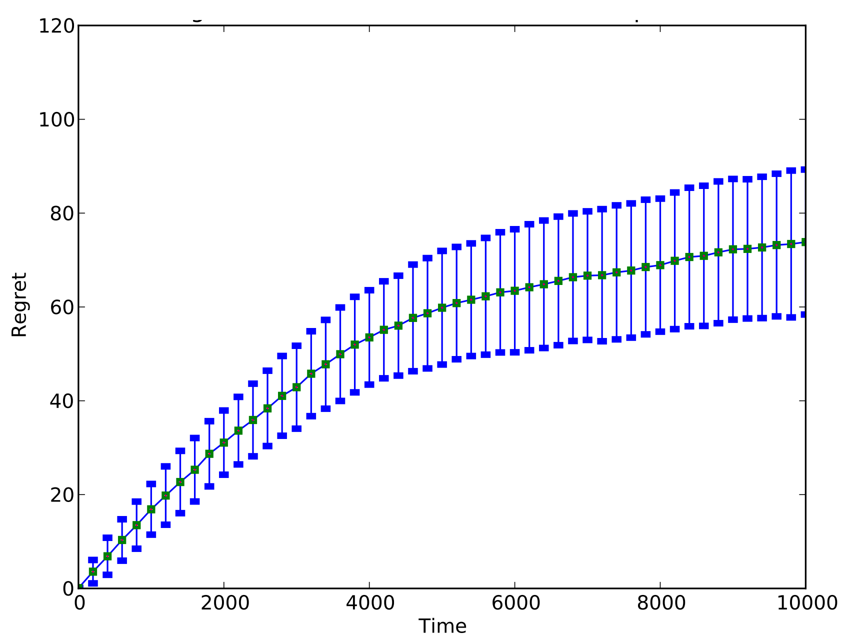
Explore then exploit



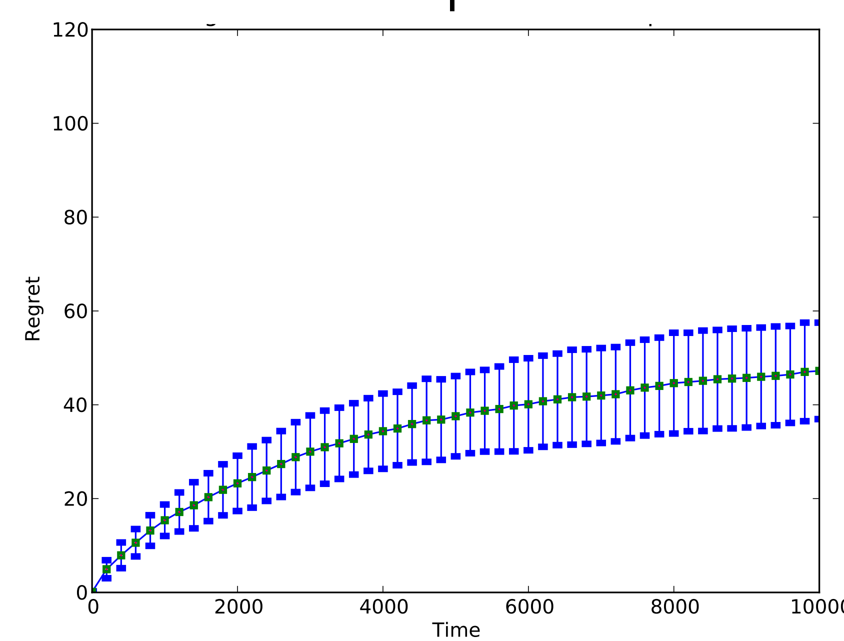

Q-learning w. dithering




OFULQ

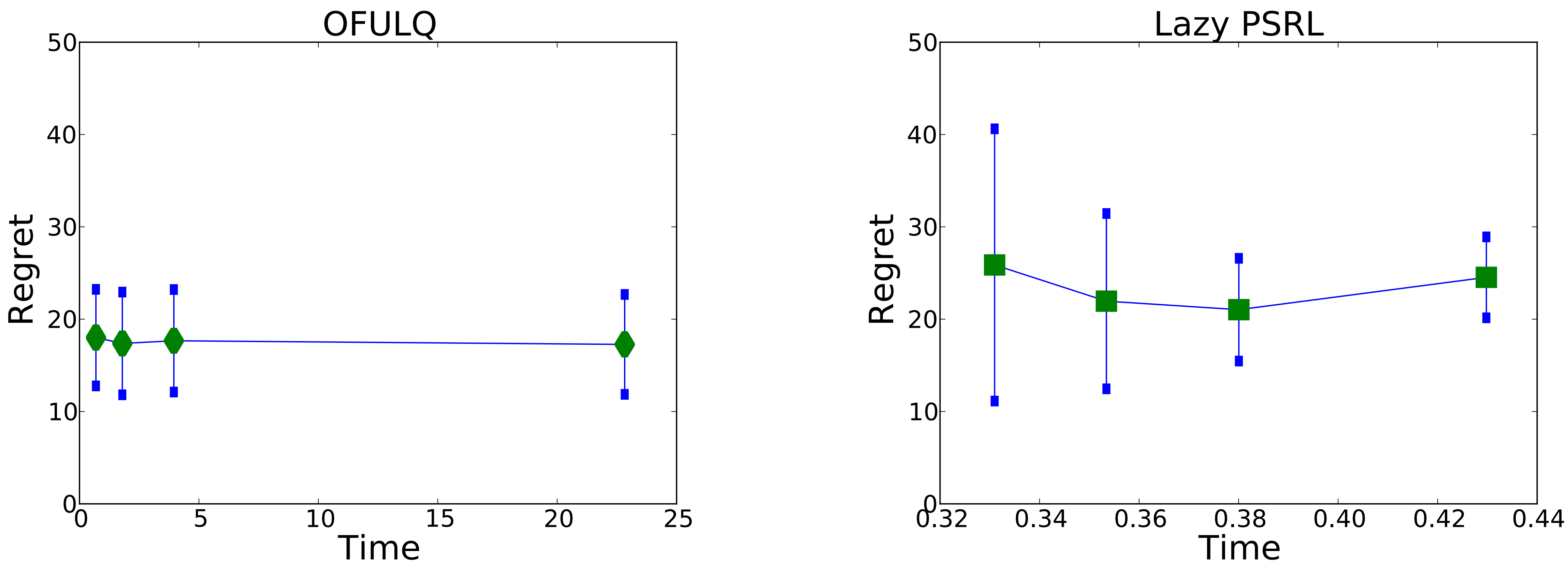



OFULQ prefetch



# OFULQ vs. PSRL

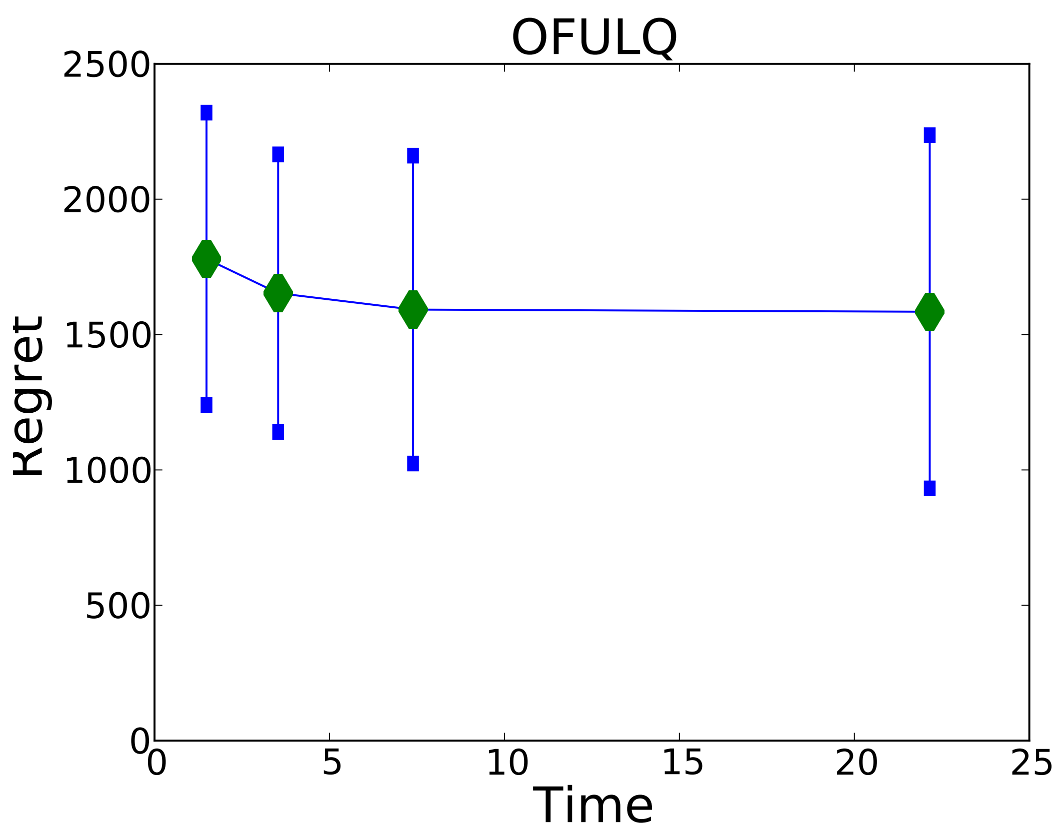
The frequency of policy switches is controlled by a parameter, which ultimate controls the computation time

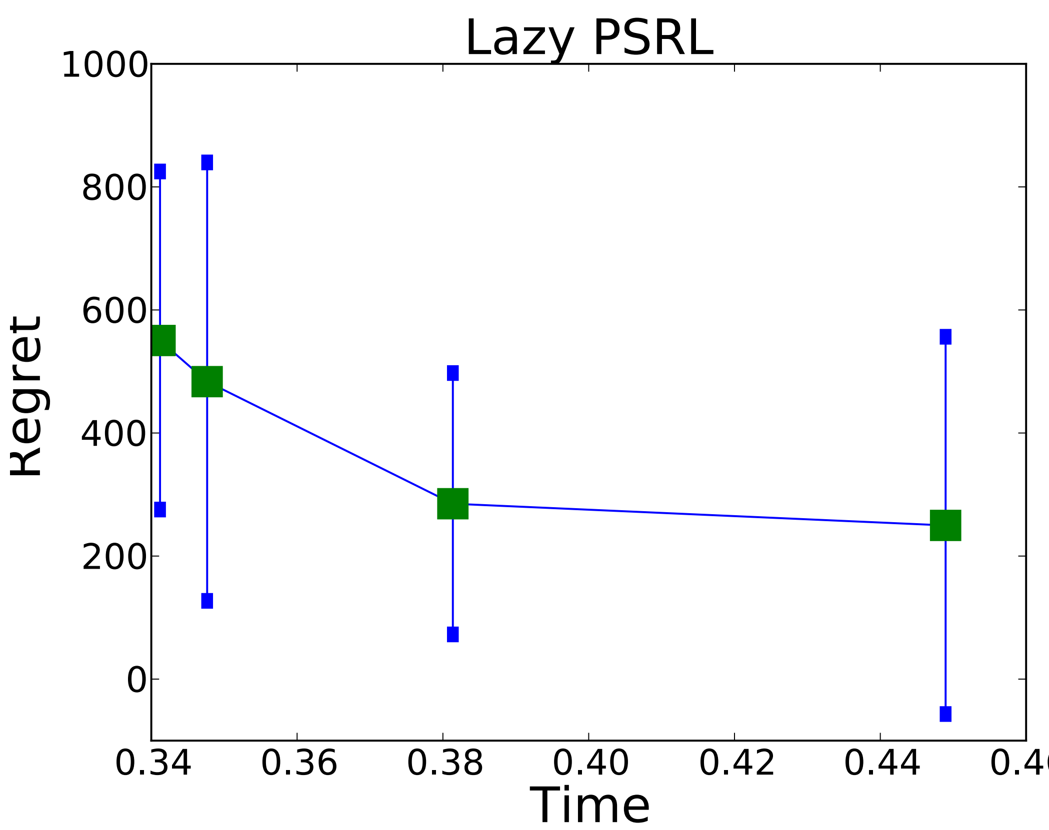



OFULQ = OFU on LQR

Lazy PSRL = PSRL that switches to new policy based on $M(x,a)$

# Higher noise




OFULQ = OFU on LQR

Lazy PSRL = PSRL that switches to new policy based on $M(x, a)$

# High dimensional bandits

# Bandit Problems


**Lever 1**
**Known payout**
**$0.25 bet**
**$0.30 win!**

**Lever 2**
**Unknown payout**
**$0.25 bet**
**$? win**

**EXPLOITATION**

**EXPLORATION**

Goal: maximize the total reward incurred

# Linear Bandits

# Linear Bandits

# Linear Bandits

- Actions are elements of a vector space:

$$a \in \mathcal{A} \subset \mathbb{R}^d$$

- Reward: $$R_t = \langle A_t, \theta_* \rangle + Z_t$$

# Linear Bandits

- Actions are elements of a vector space:

$$a \in \mathcal{A} \subset \mathbb{R}^d$$

- Reward: $R_t = \langle A_t, \theta_* \rangle + Z_t$

- L2 problem: $\|\theta\|_2 \leq 1, \|a\|_2 \leq 1$

# Linear Bandits

- Actions are elements of a vector space:

$$a \in \mathcal{A} \subset \mathbb{R}^d$$

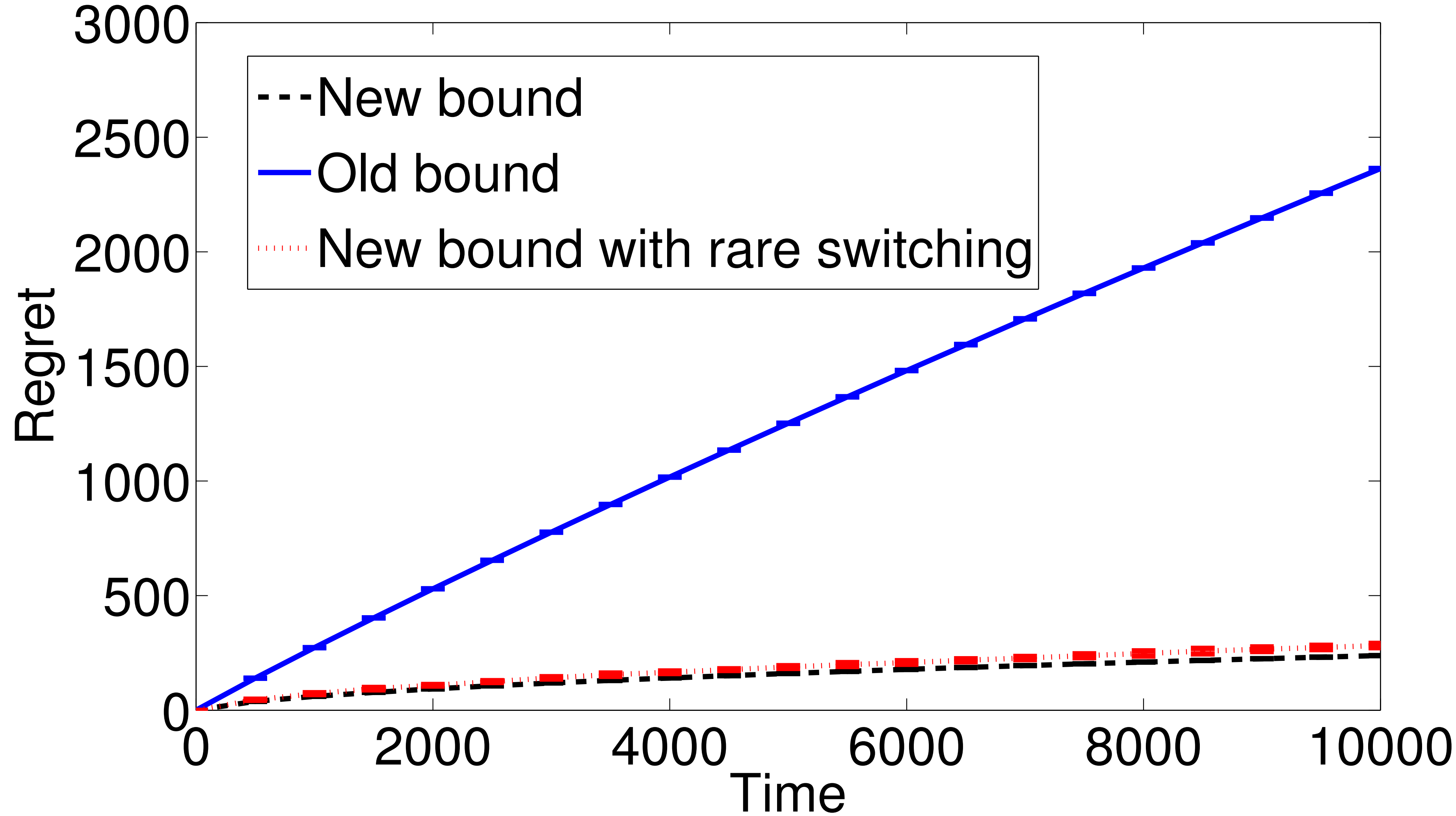
- Reward: $R_t = \langle A_t, \theta_* \rangle + Z_t$

- L2 problem: $\|\theta\|_2 \leq 1, \|a\|_2 \leq 1$

- **<u>Theorem [Dani et al '08]:</u>** For subgaussian noise, OFU's regret for the L2 problem is

$$R_T = \tilde{O}(d\sqrt{T})$$

# Confidence sets matter




- “New bound”: Abbasi-Pal-Sz’11
- “Old bound”: Dani-Hayes-Kakade ‘08

# The challenge

# The challenge

- Linear estimation problem

# The challenge

- Linear estimation problem

  The observations are $R_1, A_1, \ldots, R_t, A_t$, where

$$\ldots, R_t = \langle A_t, \theta_* \rangle + Z_t, \ldots$$

# The challenge

- Linear estimation problem

  The observations are $R_1, A_1, \ldots, R_t, A_t$, where

$$\ldots, R_t = \langle A_t, \theta_* \rangle + Z_t, \ldots$$

- Given $0 \le \delta \le 1$, find a set

$$C_t = C_t(\delta, R_1, A_1, \ldots, R_t, A_t) \subset \mathbb{R}^d$$

  such that $\mathbb{P}\left(\theta_* \in C_t\right) \ge 1 - \delta$

# The challenge

- Linear estimation problem

  The observations are $R_1, A_1, \ldots, R_t, A_t$, where

$$\ldots, R_t = \langle A_t, \theta_* \rangle + Z_t, \ldots$$

- Given $0 \le \delta \le 1$, find a set

$$C_t = C_t(\delta, R_1, A_1, \ldots, R_t, A_t) \subset \mathbb{R}^d$$

  such that $\mathbb{P}\left(\theta_* \in C_t\right) \ge 1 - \delta$

- The covariates, $A_t \in \mathbb{R}^d$, are chosen by a bandit algorithm, they are **far from independent**!
- We need a **honest** confidence set!

# The challenge

- Linear estimation problem

  The observations are $R_1, A_1, \dots, R_t, A_t$, where

$$\dots, R_t = \langle A_t, \theta_* \rangle + Z_t, \dots$$

- Given $0 \le \delta \le 1$, find a set

$$C_t = C_t(\delta, R_1, A_1, \dots, R_t, A_t) \subset \mathbb{R}^d$$

  such that $\mathbb{P}\left(\theta_* \in C_t\right) \ge 1 - \delta$

- The covariates, $A_t \in \mathbb{R}^d$, are chosen by a bandit algorithm, they are **far from independent**!
- We need a **honest** confidence set!
- How to exploit sparsity of $\theta_*$?

# A general solution

# A general solution

- If we have a good predictor for an adversarial linear regression problem with small regret, the predictions $\hat{R}_1, \ldots, \hat{R}_t$ and the regret bound $B_t$ should give us a honest, tight confidence set.

# A general solution

- If we have a good predictor for an adversarial linear regression problem with small regret, the predictions $\hat{R}_1, \ldots, \hat{R}_t$ and the regret bound $B_t$ should give us a honest, tight confidence set.

- **<u>Theorem [Abbasi-Pal-Sz ’12]</u>**: With probability $1-\delta,\ \theta_* \in C_n$ holds for all $n \geq 1$, where

$$C_n = \left\{ \theta \in \mathbb{R}^d \ :\ \sum_{t=1}^{n} (\hat{R}_t - \langle A_t, \theta \rangle)^2 \right.$$

$$\left. \leq 1 + 2B_n + 32\gamma^2 \ln\left( \frac{\gamma\sqrt{8} + \sqrt{1+B_n}}{\delta} \right) \right\}$$

# Sparse Linear Bandits

# Sparse Linear Bandits

- **Theorem [YPSz '12]:** The regret of OFUL enjoys

$$R_T = \tilde{O}(\sqrt{dTB_T})$$

# Sparse Linear Bandits

- **<u>Theorem [YPSz '12]:</u>** The regret of OFUL enjoys

$$R_T = \tilde{O}(\sqrt{dTB_T})$$

- **<u>Theorem [Gerchinowitz '11]:</u>** There exist an algorithm that achieves

$$B_T = O(p \log(dT))$$

for linear regression with ***p***-sparse parameter vectors belonging to the hypercube.

# Sparse Linear Bandits

- **Theorem [YPSz '12]:** The regret of OFUL enjoys

$$R_T = \tilde{O}(\sqrt{dTB_T})$$

- **Theorem [Gerchinowitz '11]:** There exist an algorithm that achieves

$$B_T = O(p\log(dT))$$

  for linear regression with ***p***-sparse parameter vectors belonging to the hypercube.

- **Corollary [YPSz '12]:** For such problems,

$$R_T = \tilde{O}(\sqrt{dpT})$$

# Sparse Linear Bandits

- **<u>Theorem [YPSz '12]:</u>** The regret of OFUL enjoys

$$R_T = \tilde{O}(\sqrt{dTB_T})$$

- **<u>Theorem [Gerchinowitz '11]:</u>** There exist an algorithm that achieves

$$B_T = O(p \log(dT))$$

for linear regression with ***p***-sparse parameter vectors belonging to the hypercube.

- **<u>Corollary [YPSz '12]:</u>** For such problems,

$$R_T = \tilde{O}(\sqrt{dpT})$$

- **<u>Theorem [YPSz'12]:</u>** For all algorithms,

$$R_T = \Omega(\sqrt{dT})$$

# Why?

# Why?

- Prediction problems (Candes, Tao 2006 and Bickel, Ritov, Tsybakov 2009), under RIP for LASSO:

$$\left\|\hat{\theta}_n - \theta_*\right\|_2 \sim \sqrt{p \log(d)/n}$$

# Why?

- Prediction problems (Candes, Tao 2006 and Bickel, Ritov, Tsybakov 2009), under RIP for LASSO:

$$\left\|\hat{\theta}_n - \theta_*\right\|_2 \sim \sqrt{p \log(d)/n}$$

- What is the difference?

# Why?

- Prediction problems (Candes, Tao 2006 and Bickel, Ritov, Tsybakov 2009), under RIP for LASSO:

$$\left\|\hat{\theta}_n - \theta_*\right\|_2 \sim \sqrt{p \log(d)/n}$$

- What is the difference?

  - Good algorithms select good actions frequently ==> No RIP

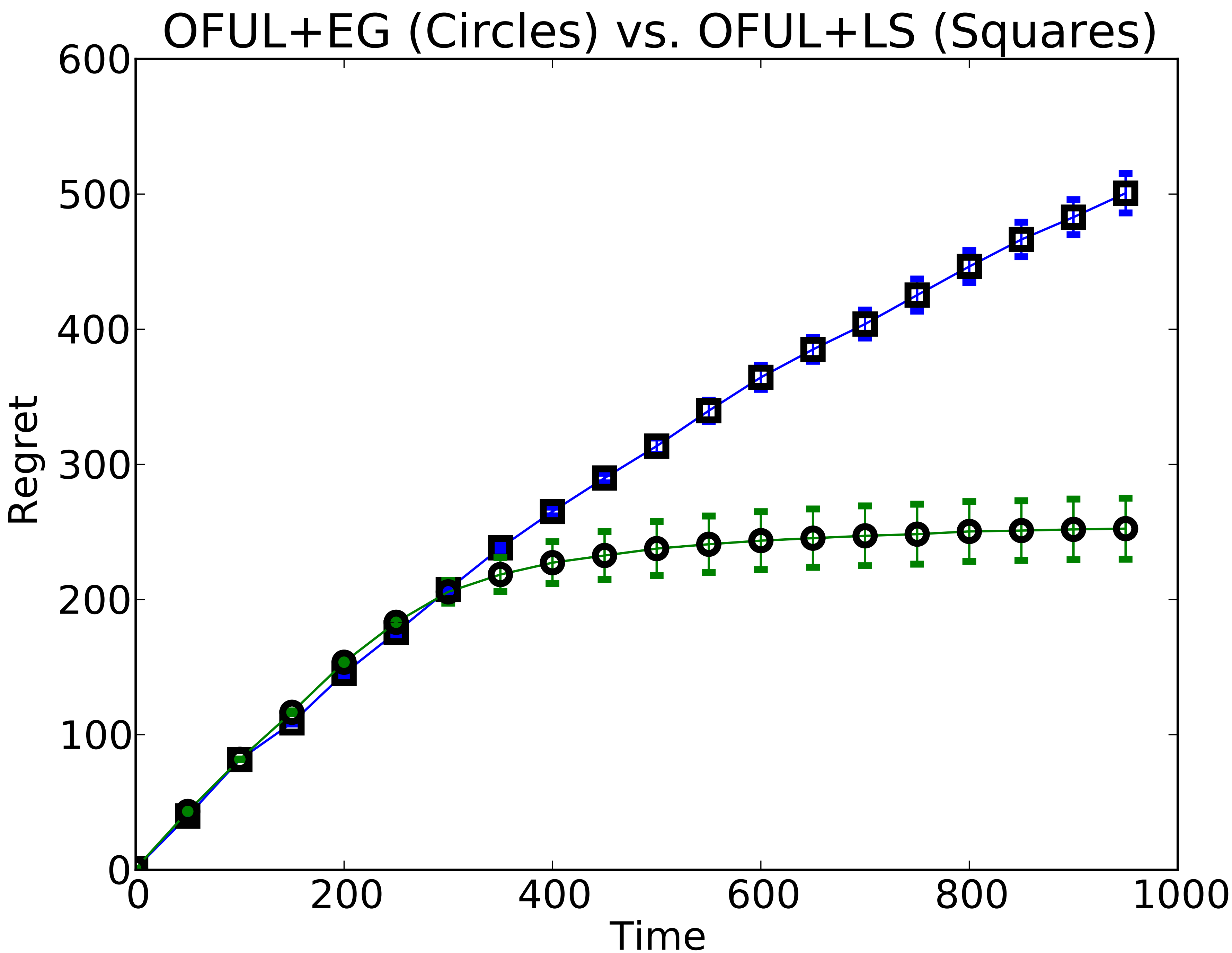
# Why?

- Prediction problems (Candes, Tao 2006 and Bickel, Ritov, Tsybakov 2009), under RIP for LASSO:

$$\left\|\hat{\theta}_n - \theta_*\right\|_2 \sim \sqrt{p \log(d)/n}$$

- What is the difference?
    - Good algorithms select good actions frequently ==> No RIP
    - Covariates are highly correlated

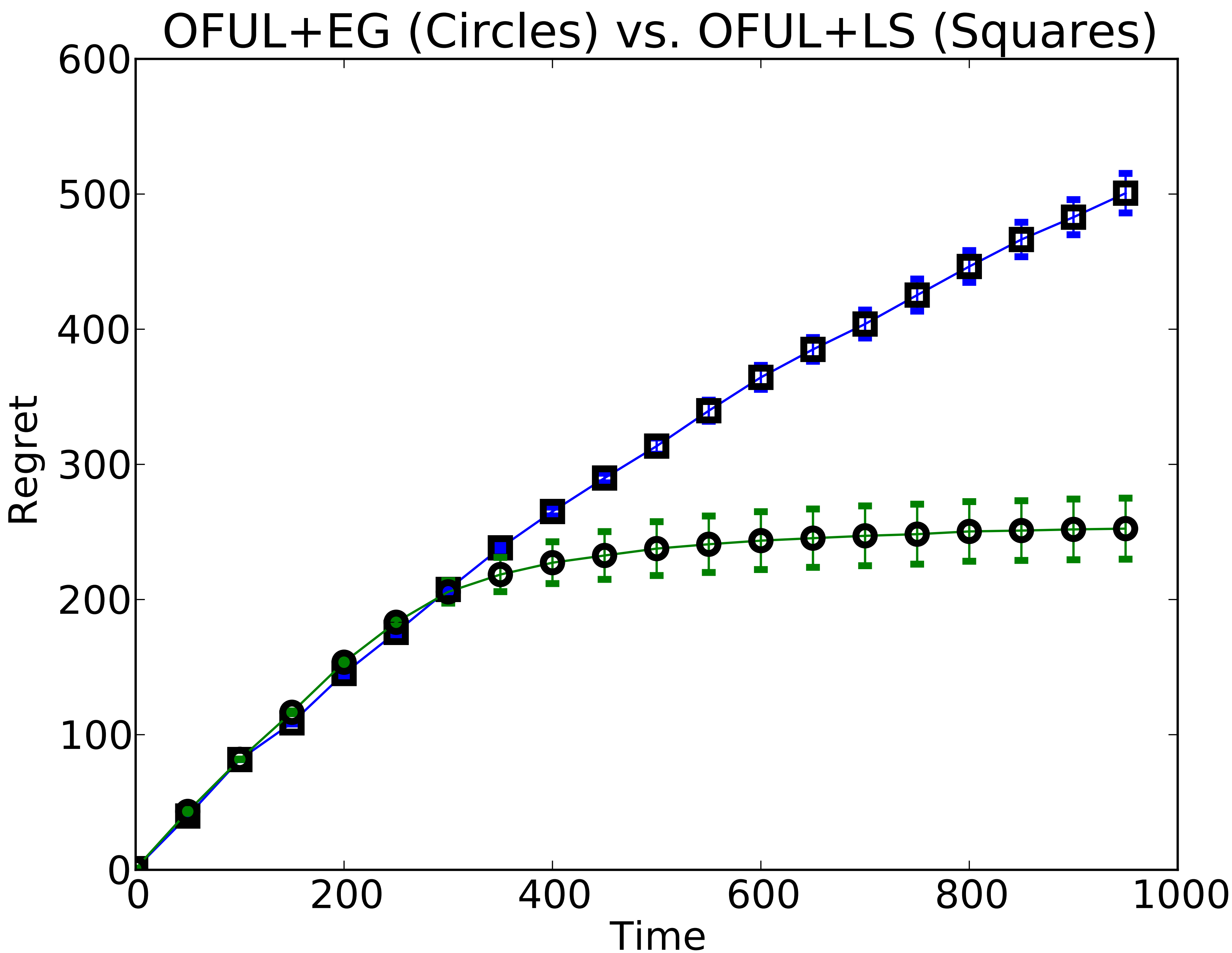
# Still.. does it work?




d = 100, p = 10

# Still.. does it work?

Yes, it does!




d = 100, p = 10

# Summary

# Summary

- To make impact, we need to solve **decision problems**

# Summary

- To make impact, we need to solve **decision problems**
- This makes a **BIG** difference

# Summary

- To make impact, we need to solve **decision problems**
- This makes a **BIG** difference
  - Passive data collection can be extremely ineffective: **small "big data" !?**

# Summary

- To make impact, we need to solve **decision problems**
- This makes a **BIG** difference
  - Passive data collection can be extremely ineffective: **small "big data" !?**
- Need **smart algorithms for learning and control**

# Summary

- To make impact, we need to solve **decision problems**
- This makes a **BIG** difference
  - Passive data collection can be extremely ineffective: **small “big data” !?**
- Need **smart algorithms for learning and control**
  - Planning to learn is critical

# Summary

- To make impact, we need to solve **decision problems**
- This makes a **BIG** difference
  - Passive data collection can be extremely ineffective: **small "big data" !?**
- Need **smart algorithms for learning and control**
  - Planning to learn is critical
  - OFU or PSRL: Competing designs

# Summary

- To make impact, we need to solve **decision problems**
- This makes a **BIG** difference
  - Passive data collection can be extremely ineffective: **small “big data” !?**
- Need **smart algorithms for learning and control**
  - Planning to learn is critical
  - OFU or PSRL: Competing designs
- Current research: **Scaling up!**

# Summary

- To make impact, we need to solve **decision problems**
- This makes a **BIG** difference
  - Passive data collection can be extremely ineffective: **small “big data” !?**
- Need **smart algorithms for learning and control**
  - Planning to learn is critical
  - OFU or PSRL: Competing designs
- Current research: **Scaling up!**
  - S p a r s i t y..?

# Summary

- To make impact, we need to solve **decision problems**
- This makes a **BIG** difference
  - Passive data collection can be extremely ineffective: **small “big data” !?**
- Need **smart algorithms for learning and control**
  - Planning to learn is critical
  - OFU or PSRL: Competing designs
- Current research: **Scaling up!**
  - S p a r s i t y..?

Significant computational, algorithmic and statistical challenges remain. Much to be done!!

Thanks for being here!
Questions?