Lecture 2: Immediate RL, Bandits and the Full RL problem

B. Ravindran

Immediate Reinforcement

- The payoff accrues immediately after an action is chosen
- One key question - the dilemma between exploration and exploitation
- *Bandit problems* encapsulate 'Explore vs Exploit'

The Explore-Exploit Dilemma

- Explore to find profitable actions
- Exploit to act according to the best observations already made
- Always exploiting might not be optimal
- Always exploring might not be optimal either
- Hence, there is an explore-exploit dilemma

Multi-arm Bandits

- n-arm bandit problem is to learn to preferentially select a particular action (arm) from a set of n actions $(1, 2, 3, \ldots, n)$
- Each selection results in Rewards derived from the respective probability distribution
- Arm i has a reward distribution with mean μ_i and

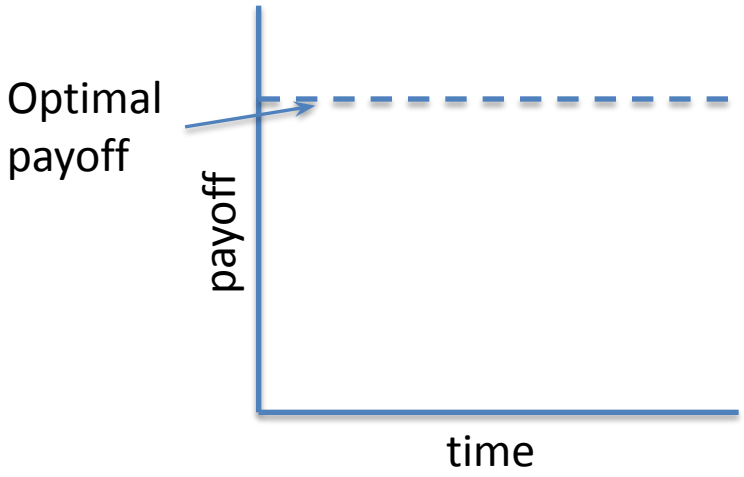
$$\mu^* = \max\{\mu_i\}$$

Objectives

- Identify the *correct* arm eventually
- Maximize the total rewards obtained
  - Minimize regret (= loss) while learning

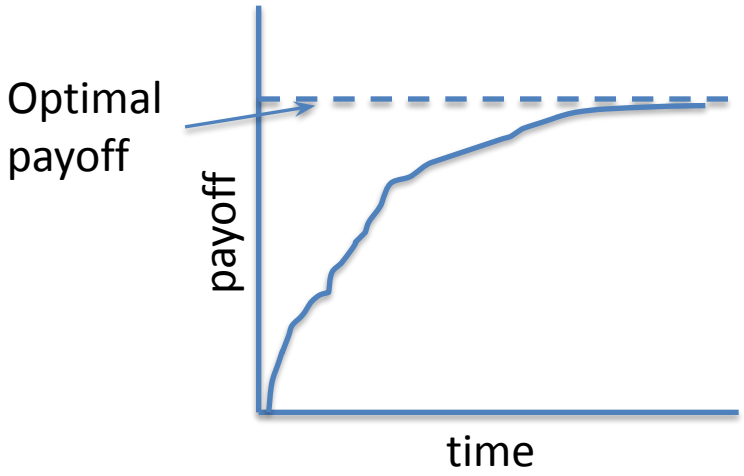
Objectives

- Identify the *correct* arm eventually
- Maximize the total rewards obtained
  - Minimize regret (= loss) while learning



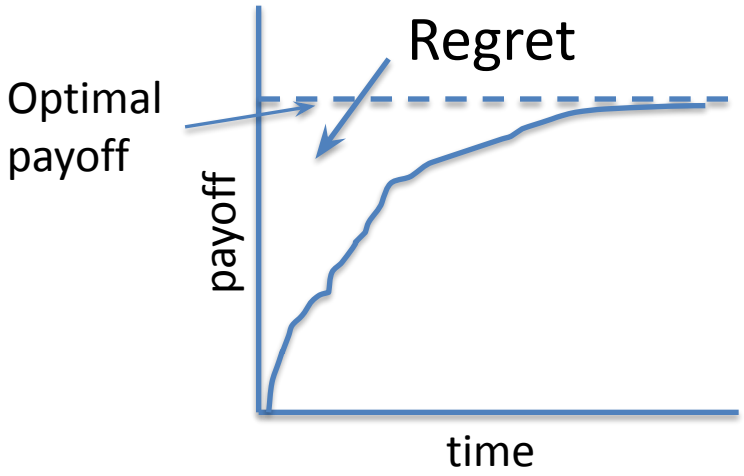
Objectives

- Identify the *correct* arm eventually
- Maximize the total rewards obtained
  - Minimize regret (= loss) while learning



Objectives

- Identify the *correct* arm eventually
- Maximize the total rewards obtained
  - Minimize regret (= loss) while learning



Objectives

- Identify the *correct* arm eventually
- Maximize the total rewards obtained
  - Minimize regret (= loss) while learning
- Probably Approximately Correct (PAC) frameworks
  - Identification of an ε-optimal arm with probability $1 - \delta$
  - ε-Optimal: Mean of the selected arm satisfies $\mu \geq \mu^* - \epsilon$
  - Minimize sample complexity: Order of samples required for such an arm identification

Traditional Approaches

- Let $r_{i,k}$ be the reward sample acquired when i^{th} arm is selected for the k^{th} time
- Define:

$$Q(a_i) = \frac{\sum_k r_{i,k}}{\sum_{\{k:r_{i,k}\}} 1} \qquad Q(a_i^*) = \max_i \{Q(a_i)\}$$

$$Q_{k+1}(a_i) = Q_k(a_i) + \alpha(r_k - Q_k(a_i))$$

- Setting $\alpha = \frac{1}{k_i + 1}$ yields the average.

Traditional Approaches

- Epsilon Greedy: Select arm $a^* = \text{argmax}_i \{Q_k(a_i)\}$ with probability 1- ε and select any arbitrary arm with probability ε
- Softmax: Select arms with probability proportional to the current value estimates

$$\pi_k(a_i) = \frac{\exp(Q_k(a_i)/\tau)}{\sum_j \exp(Q_k(a_j)/\tau)}$$

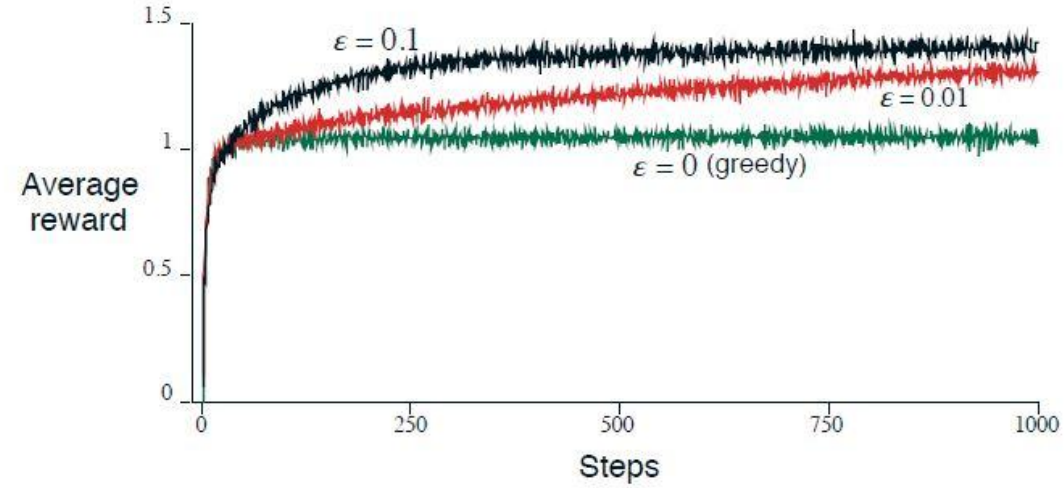
Traditional Approaches

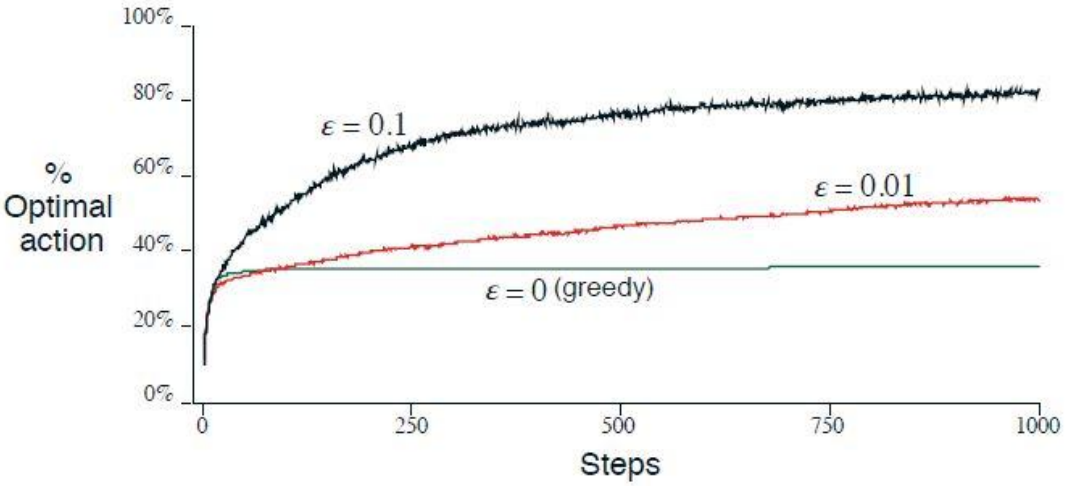
- Epsilon Greedy: Select arm $a^* = \text{argmax}_i \{Q_k(a_i)\}$ with probability 1- ε and select any arbitrary arm with probability ε
- Softmax: Select arms with probability proportional to the current value estimates

$$\pi_k(a_i) = \frac{\exp(Q_k(a_i)/\tau)}{\sum_j \exp(Q_k(a_j)/\tau)}$$

- Asymptotic Convergence guarantees

ε-Greedy Example





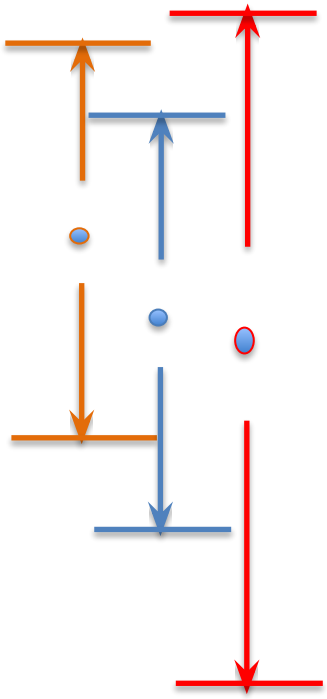
Other Approaches

- Median Elimination (Even-Dar et al., 2006)
- Upper Confidence Bounds (UCB) (Auer et al., 1998, 2010)
- Thompson Sampling (Chappelle & Li, 2001, Agrawal & Goyal, 2012)

UCB

Auer et al, ICML 1998

- Upper Confidence Bounds (UCB): The arm with the best estimate r^* so far serves as a benchmark, and other arms are played only if the upper bound of a suitable confidence interval is at least r^*
- Simplest Approach – Be greedy with respect to upper confidence bounds



UCB

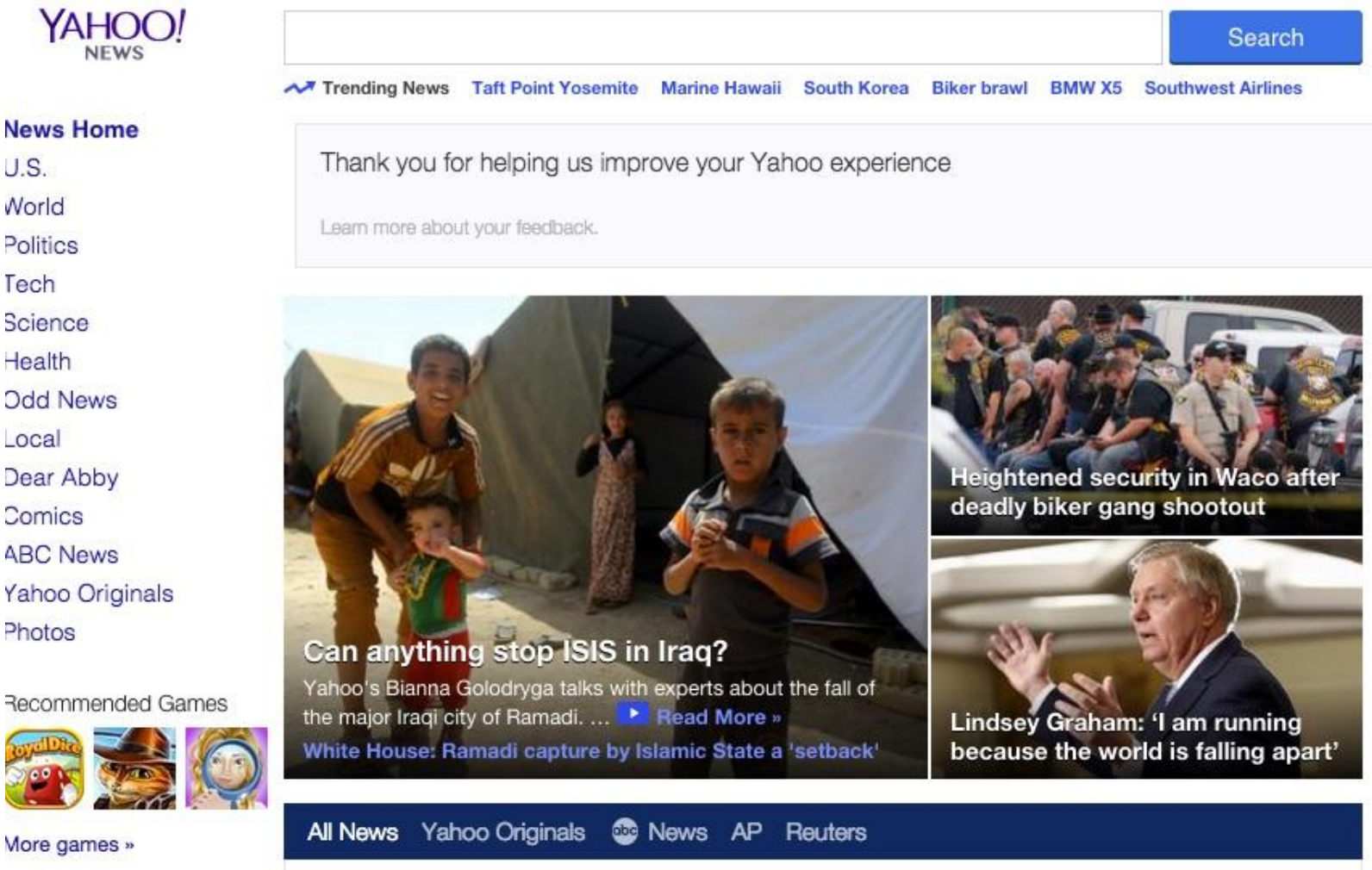
Auer et al, ICML 1998

- Sub-optimal arm *j* played fewer than $(8/\Delta_j)\ ln\ n$ times
  - Further improvements focus on reducing the constants

Deterministic policy: UCB1.
Initialization: Play each machine once.
Loop:

- Play machine j that maximizes $\bar{x}_j + \sqrt{\frac{2 \ln n}{n_j}}$, where $\bar{x}_j$ is the average reward obtained from machine j, n_j is the number of times machine j has been played so far, and n is the overall number of plays done so far.

Customization



Ad Selection

Shop for Florists in Chennai on Google

Sponsored

Online Flowers Delivery
₹1,749
Ferns N Petals

Message In A Bottle with teg
₹349
Ferns N Petals

Classic Bunch - online flower ...
₹499
FlowerAura
Special offer

Online Flower Delivery
₹599
Ferns N Petals

Relish Of Heavenly Treat
₹1,399
Ferns N Petals

Florists In Chennai - Same Day Delivery Within 4 Hrs - floweraura.com
Ad www.floweraura.com/Online-Florist/Chennai
Online **Flowers** & Gifts Delivery @ Rs 399. Best Price, 100% Smile Guaranteed.
Delivery in 4 Hrs · Mid-Night Delivery · No Hidden Cost · Free Shipping · Flowers Starting @ Rs 399
Types: Cakes, Flowers, Gifts, Chocolate

Flowers Delivery in Chennai - Expess Delivery in 2-3 Hrs
Ad www.flowersnfruits.com/Flower_Delivery/Chennai 099300 06747
Order **Flowers** Now For Express Delivery within 2-3 hrs Anywhere in **Chennai**.

Contextual Bandits

- Different ads for different users
  - One bandit for each user!
- Hard to train
  - Need several rounds of experience with same user
- Assume that the parameters of the reward distributions themselves are determined by a set of hyperparameters
  - Typical assumption is a linear parameterization of the expectation

Contextual Bandits

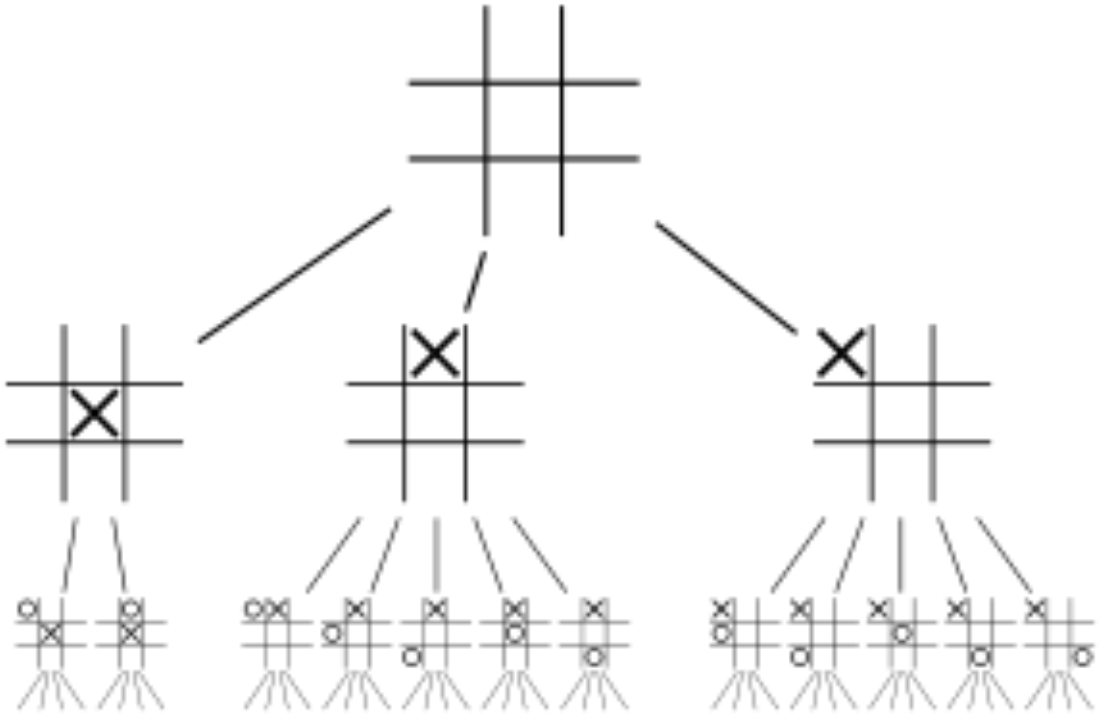
- Assume that each user is represented by a set of features
  - Can be joint features of user and arm
- The "statistic" used for choosing arms is now dependent on these features
- Could correspond to the presence or absence of different signals

LinUCB

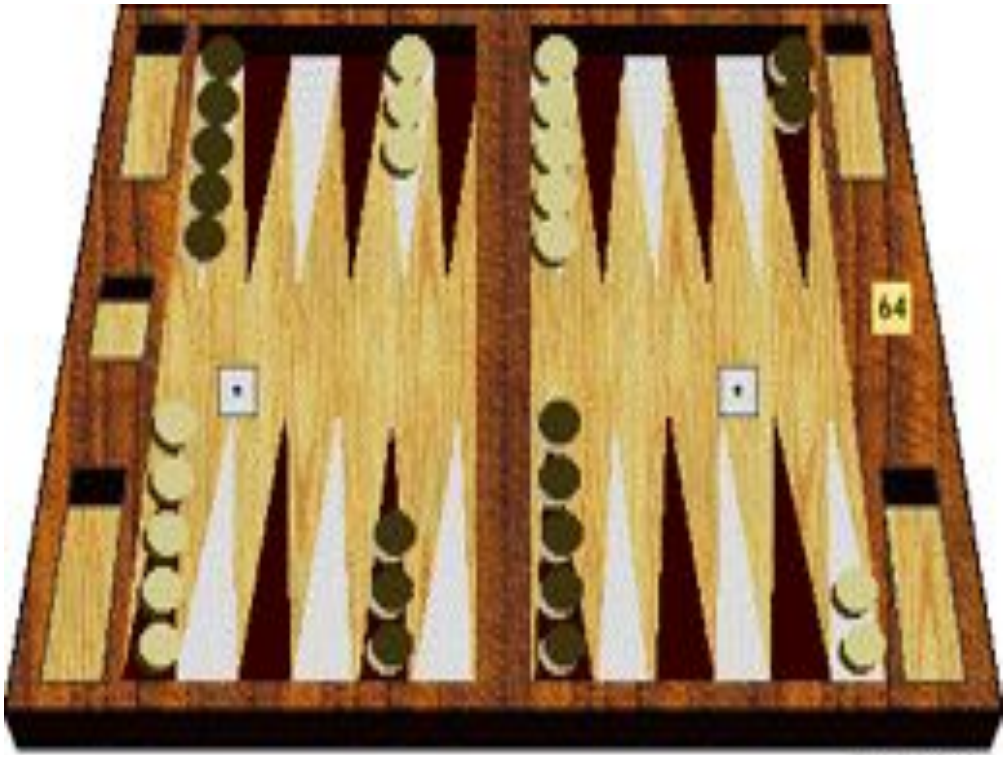
Li et al., WWW10

- One of the more popular contextual bandit algorithms
- *Predicted expected reward* assumed to be a linear function of the features
  - Use ridge regression to fit parameters
  - Can derive upper confidence bounds for the regression fit
  - Use UCB like action selection
  - Gives better performance with lesser “training” data

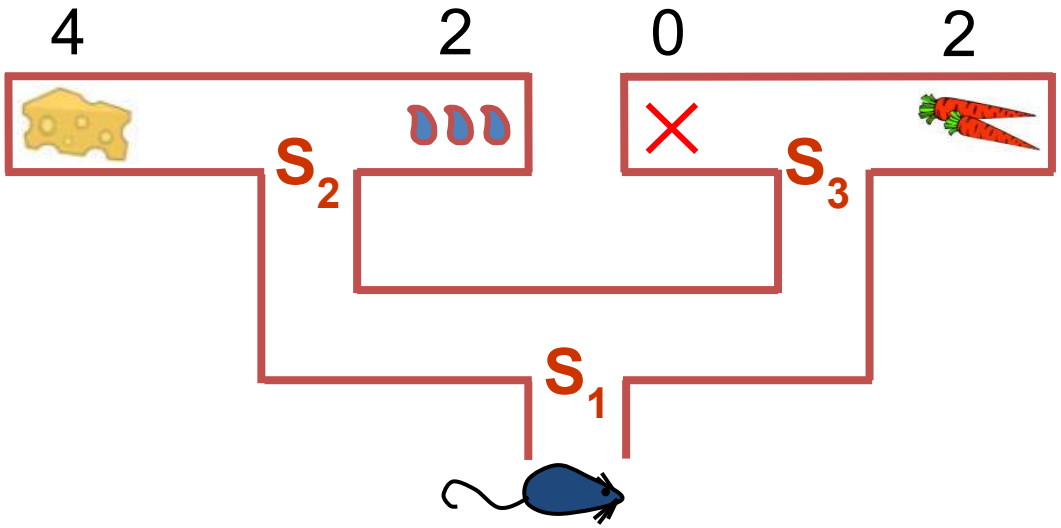
What about Tic-Tac-Toe?



What about Backgammon?

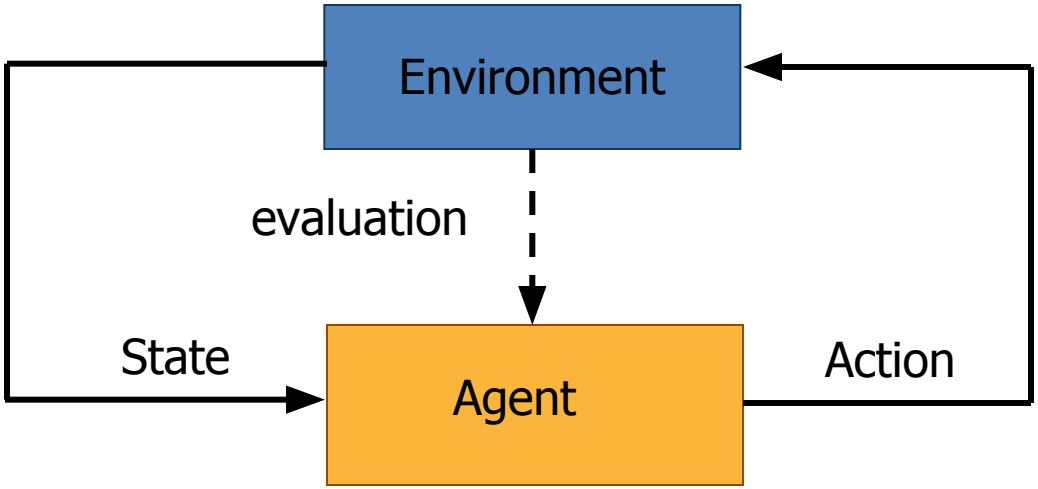


Action at a (Temporal) Distance



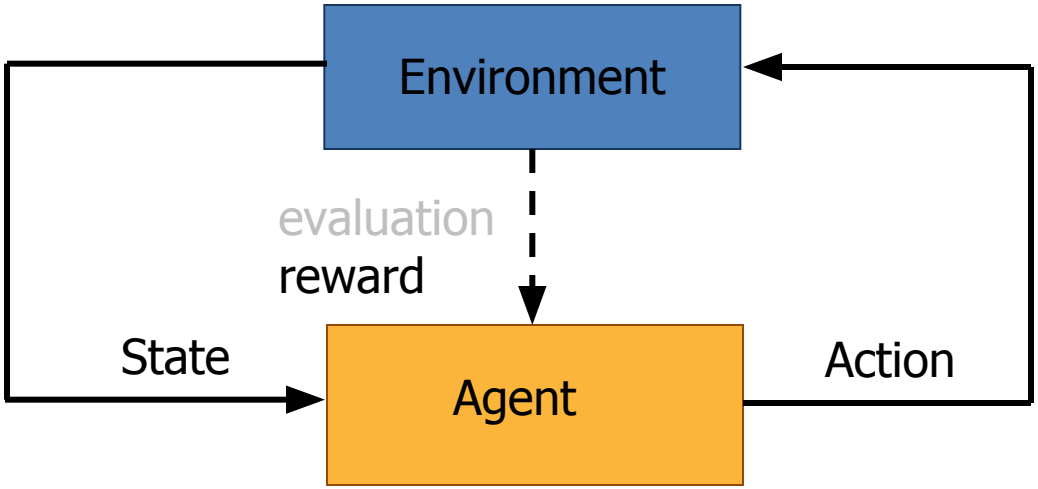
- learning an appropriate action at S_1:
  - –depends on the actions at S_2 and S_3
  - –gains no immediate feedback

Full RL Framework



- Learn from close interaction
- ...with a stochastic environment
- ...having noisy delayed scalar evaluation
- ...with a goal to maximize a measure of long term performance

Full RL Framework

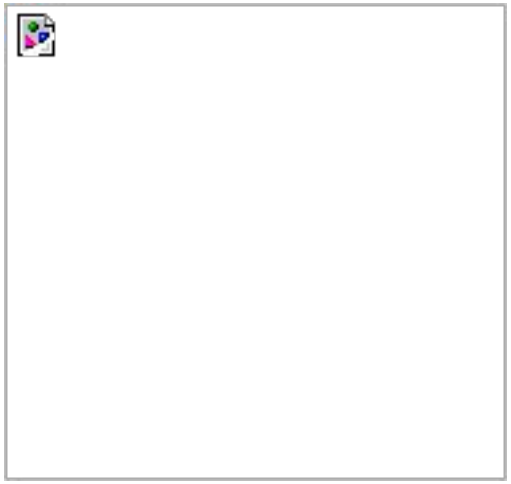


- Learn from close interaction
- ...with a stochastic environment
- ...having noisy delayed scalar evaluation
- ...with a goal to maximize a measure of long term performance

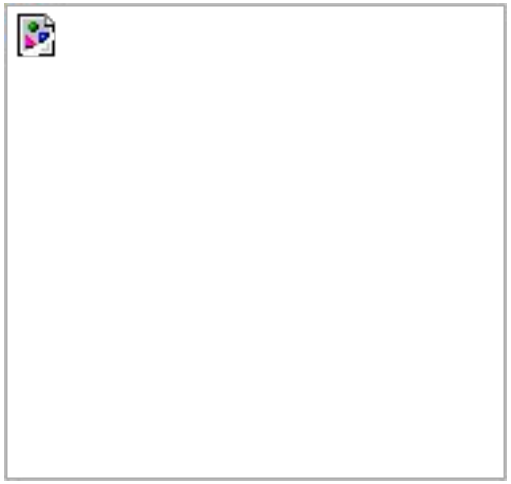
Designing an RL solution

- States
  - Enough information to take decisions
  - Raw inputs often not sufficient
- Actions
  - The control variables
  - Discrete – items to recommend, moves in a game
  - Continuous – torque to a motor, rate of mixing
- Rewards
  - Define the *goal* of the problem

Full RL Problem

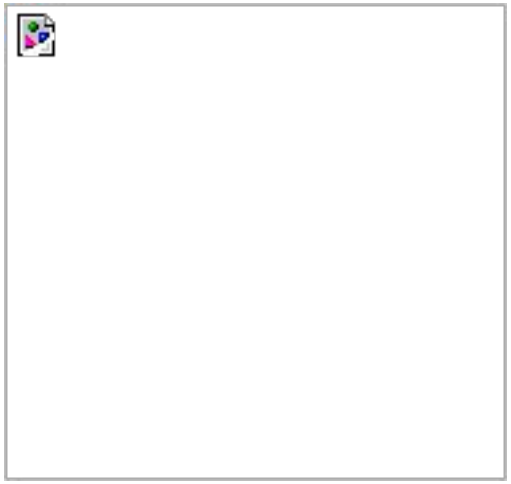


Full RL Problem



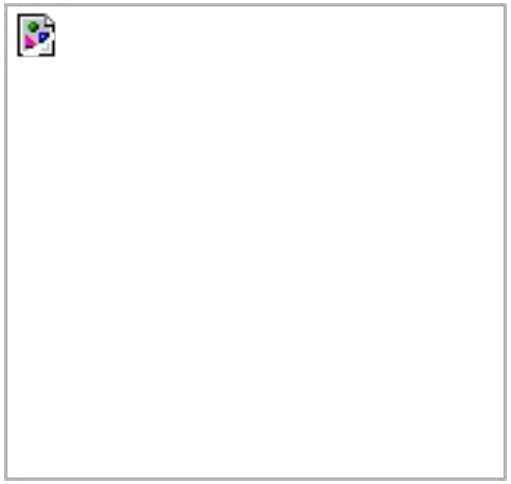
$$Q(a_i^*) = \max_i \{Q(a_i)\}$$

Full RL Problem



$$Q(a_i^*) = \max_i \{Q(a_i)\}$$

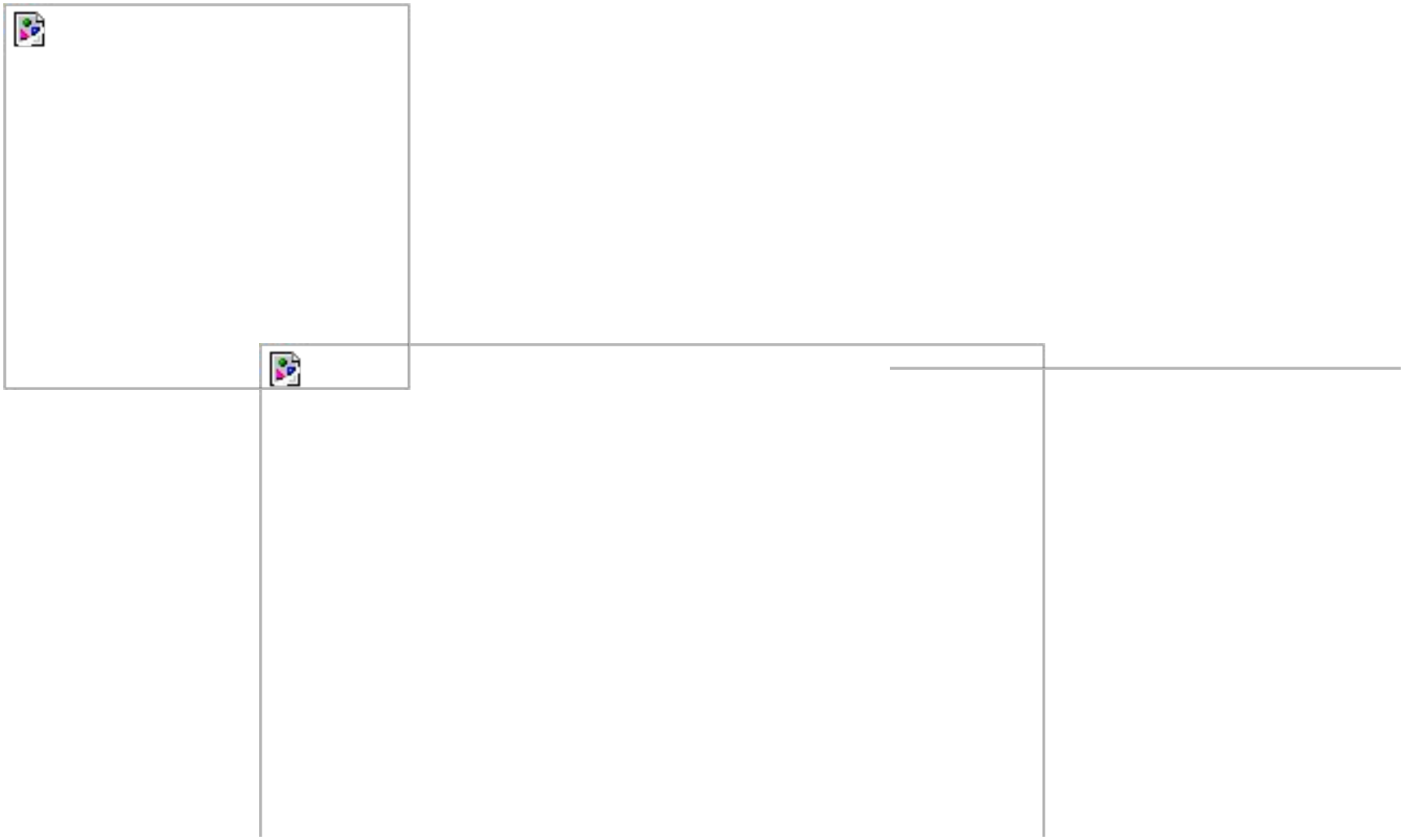
Full RL Problem



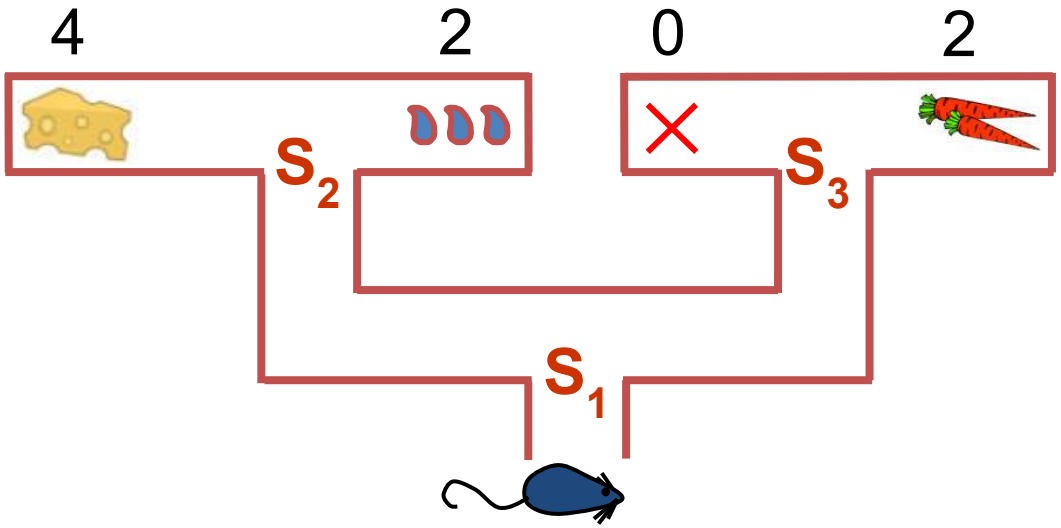
$$Q(a_i^*) = \max_i \{Q(a_i)\}$$

$$Q(j, a_i^*) = \max_i \{Q(j, a_i)\}$$

Full RL Problem

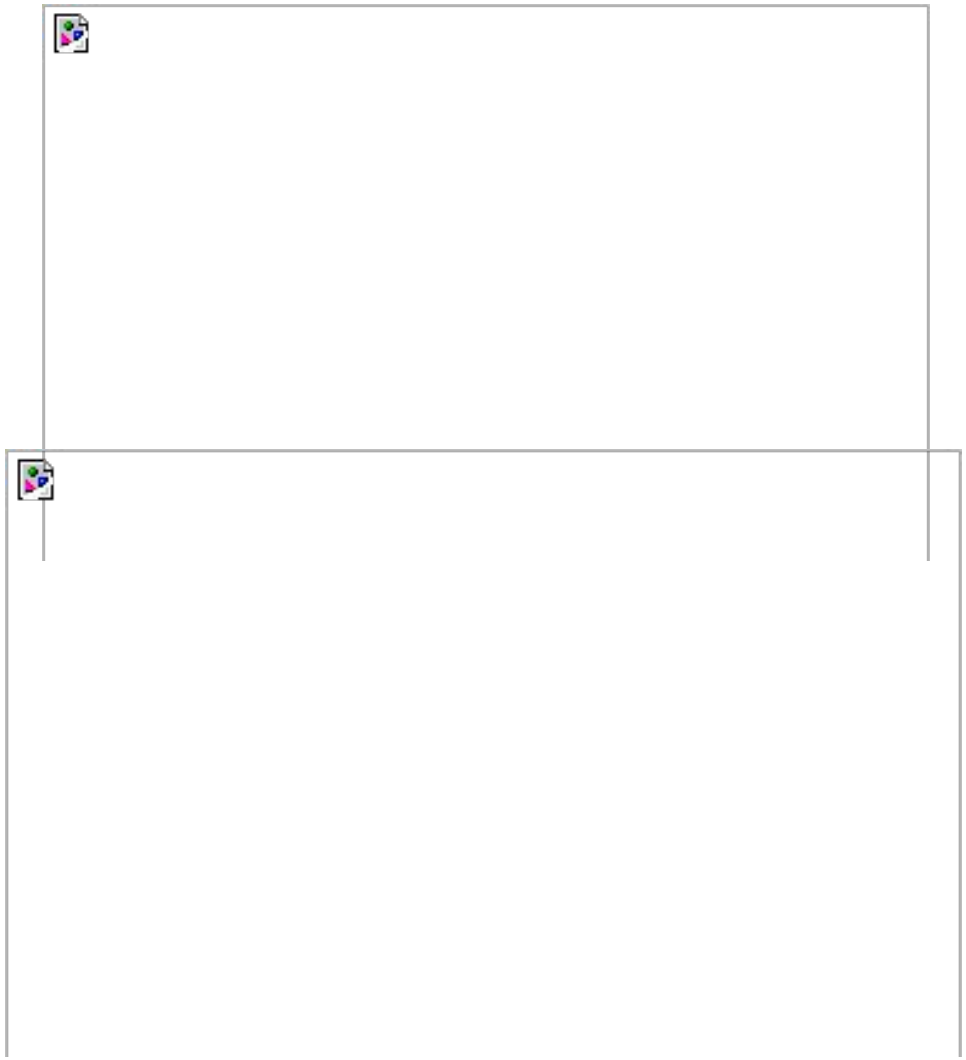


Recall: Action at a (Temporal) Distance

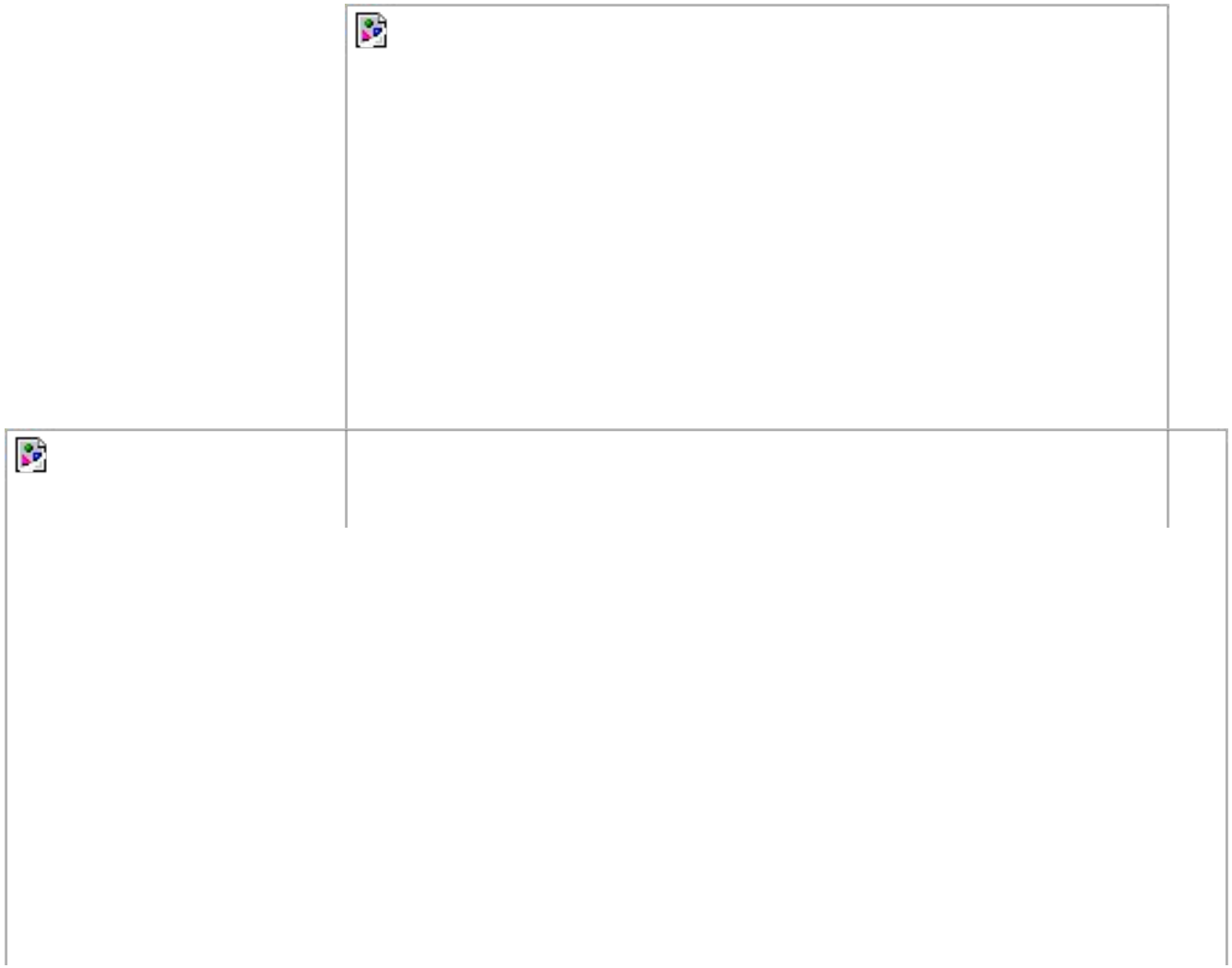


- learning an appropriate action at S_1:
  - depends on the actions at S_2 and S_3
  - gains no immediate feedback
- Idea: use prediction as surrogate feedback

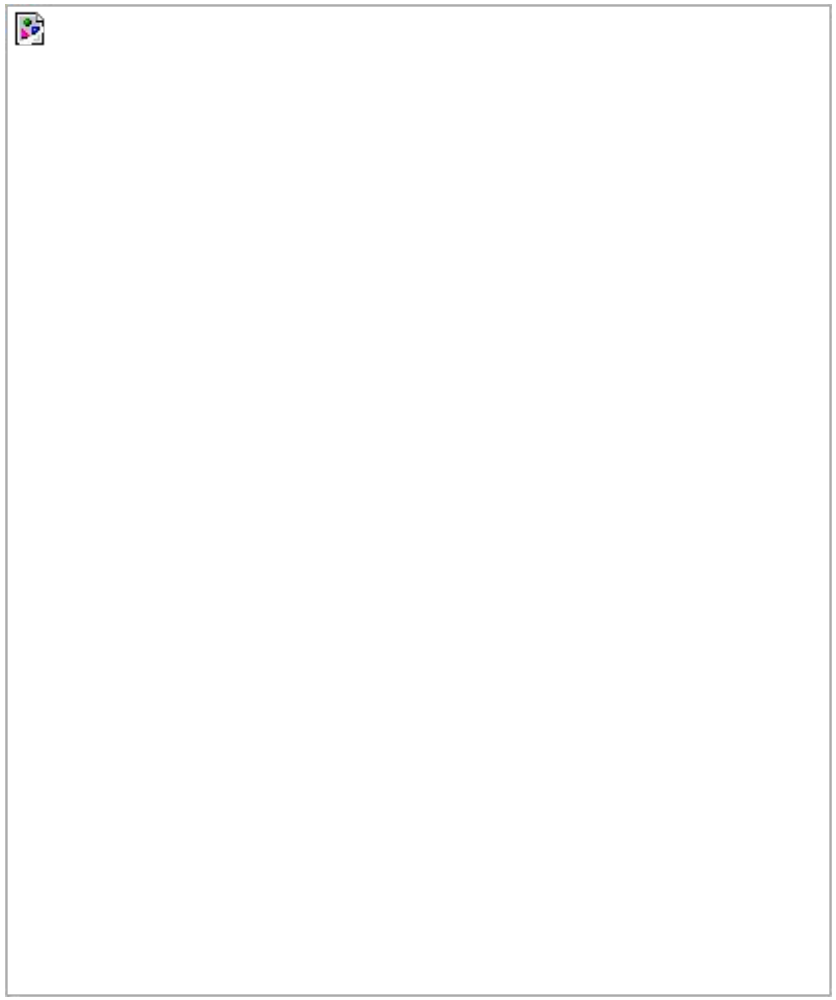
Full RL Problem



Full RL Problem



Full RL Problem



The Markov Property

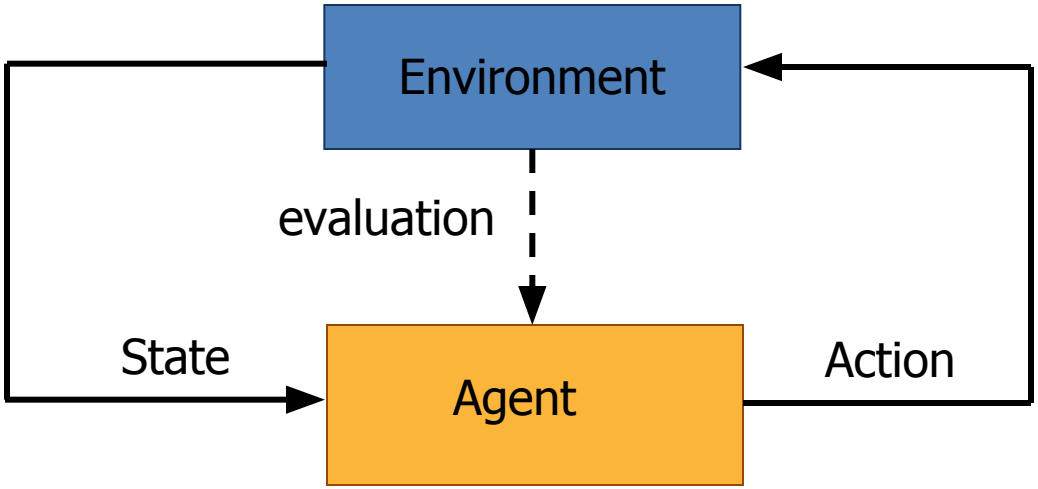
- "the state" at step *t*, means whatever information is available to the agent at step *t* about its environment.
- The state can include immediate "sensations", highly processed sensations, and structures built up over time from sequences of sensations.
- Ideally, a state should summarize past sensations so as to retain all "essential" information, i.e., it should have the **Markov Property**:

$$\Pr\left\{ s_{t+1} = s', r_{t+1} = r \,\middle|\, s_t, a_t, r_t, s_{t-1}, a_{t-1}, \ldots, r_1, s_0, a_0 \right\} =$$

$$\Pr\left\{ s_{t+1} = s', r_{t+1} = r \,\middle|\, s_t, a_t \right\}$$

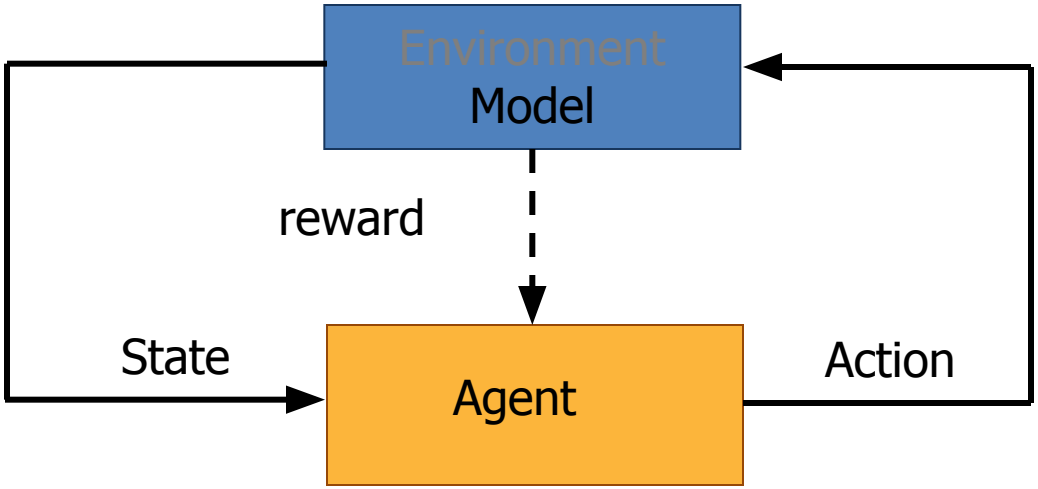
for all s', r, and histories $s_t, a_t, r_t, s_{t-1}, a_{t-1}, \ldots, r_1, s_0, a_0$.

Full RL Framework



- Learn from close interaction
- ...with a stochastic environment
- ...having noisy delayed scalar evaluation
- ...with a goal to maximize a measure of long term performance

RL vs Model Predictive Control



- MPC uses the model to plan
  - perhaps to a limited horizon, execute the plan for a few (typically one) steps and then replan
- Approximate the model to make it analytical
- Approximate the solution
  - Sometimes find open loop policies
- Concerned with stability and correctness of controller

RL vs Model Predictive Control

- RL learns from trajectory data
  - No access to the system model
  - Can use a "sample model"
  - Typically solve for infinite horizon problems
  - Almost always solve for closed loop policies
- Necessarily approximate due to the sample size
- Further approximation for generalization
- Concerned with learning speed, convergence, and scaling
  - Correctness is a challenge!

RL vs MPC

Reinforcement Learning

- Uses trajectories sampled from system (or model)
- Convergence rate, stability of learning process, scaling
- Approximate due to sample size and generalization
- Proving stability/correctness a challenge
  - Especially during the solution process

Model Predictive Control

- Uses (analytic) system model
- Stability of controllers, correctness
- Approximate due to model errors and optimizers
- Incorporating data/measurements is a challenge