Reinforcement Learning

Bandit Problems, Markov Chains and Markov Decision Processes

Subramanian Ramamoorthy
School of Informatics

20 January 2017

What is Reinforcement Learning(RL)?

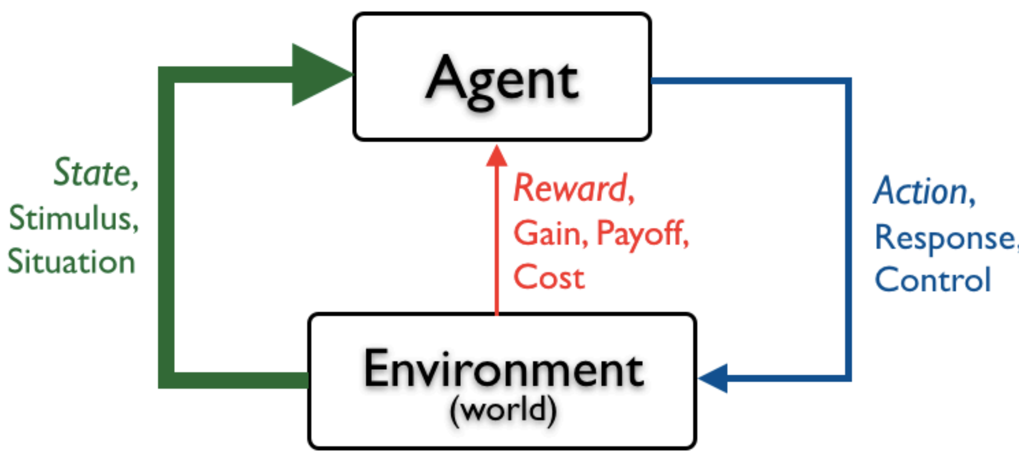
- An approach to Artificial Intelligence
- Learning from **interaction**
- Learning about, from, and while interacting (trial and error) with an external environment
- Goal-oriented learning; implying delayed rewards

- Learning what to do—how to map situations to actions—so as to maximize a numerical reward signal
- Can be thought of as a stochastic optimization over time

Setup for RL

Agent (algorithm) is:

- Temporally situated
- Continual learning and planning
- Objective is to ***affect*** the environment – actions *and* states
- Environment is uncertain, stochastic



Multi-arm Bandits (MAB)

- *N* possible actions
- You can play for some period of time and *__you want to maximize reward__* (expected utility)

Which is the best arm/ machine?

DEMO

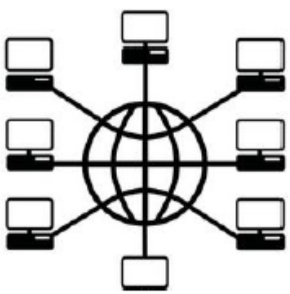
Numerous Applications!

Computer Go

Brain computer interface

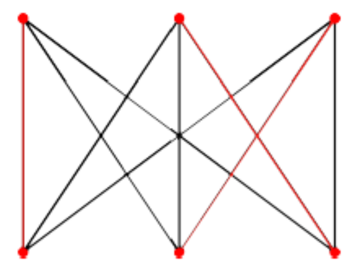
Medical trials

Packets routing

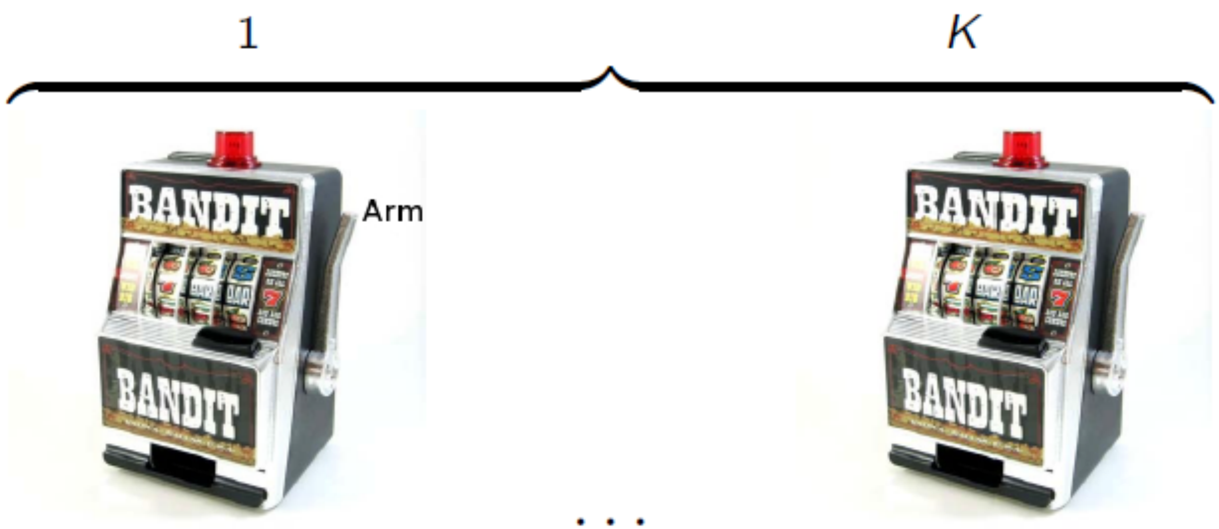


Ads placement

Dynamic allocation



What is the Choice?



t=1	0.3	0.2	0.8	0.4	0.0
t=2	0.7	0.1	0.9	0.5	0.1
t=3	0.5	0.3	0.7	0.3	0.3

...

n-Armed Bandit Problem

- Choose repeatedly from one of *n* actions; each choice is called a *play*
- After each play a_t , you get a reward r_t , where

$$E\{r_t \mid a_t\} = Q^*(a_t)$$

These are unknown ***action values***
Distribution of r_t depends only on a_t

Objective is to maximize the reward in the long term, e.g., over 1000 plays

To solve the *n*-armed bandit problem,
you must **explore** a variety of actions
and **exploit** the best of them

Exploration/Exploitation Dilemma

- Suppose you form estimates

$$Q_t(a) \approx Q^*(a)$$ action value estimates

- The **greedy** action at time t is a_t^*

$$a_t^* = \arg\max_a Q_t(a)$$

$$a_t = a_t^* \Rightarrow \text{exploitation}$$

$$a_t \neq a_t^* \Rightarrow \text{exploration}$$

Why?

- You can't exploit all the time; you can't explore all the time
- You can never stop exploring; but you could reduce exploring.

Action-Value Methods

- Methods that adapt action-value estimates and nothing else, e.g.: suppose by the t-th play, action a had been chosen k_a times, producing rewards $r_1, r_2, \ldots, r_{k_a}$, then

$$Q_t(a) = \frac{r_1 + r_2 + \cdots + r_{k_a}}{k_a}$$

"sample average"

$$\lim_{k_a \to \infty} Q_t(a) = Q^*(a)$$

What is the behaviour with finite samples?

ε-Greedy Action Selection

- Greedy action selection:

$$a_t = a_t^* = \arg\max_a Q_t(a)$$

- ε-Greedy:

$$a_t = \begin{cases} a_t^* & \text{with probability } 1-\varepsilon \\ \text{random action} & \text{with probability } \varepsilon \end{cases}$$

. . . the simplest way to balance exploration and exploitation

A simple bandit algorithm

Initialize, for $a = 1$ to k:

$Q(a) \leftarrow 0$
$N(a) \leftarrow 0$

Repeat forever:

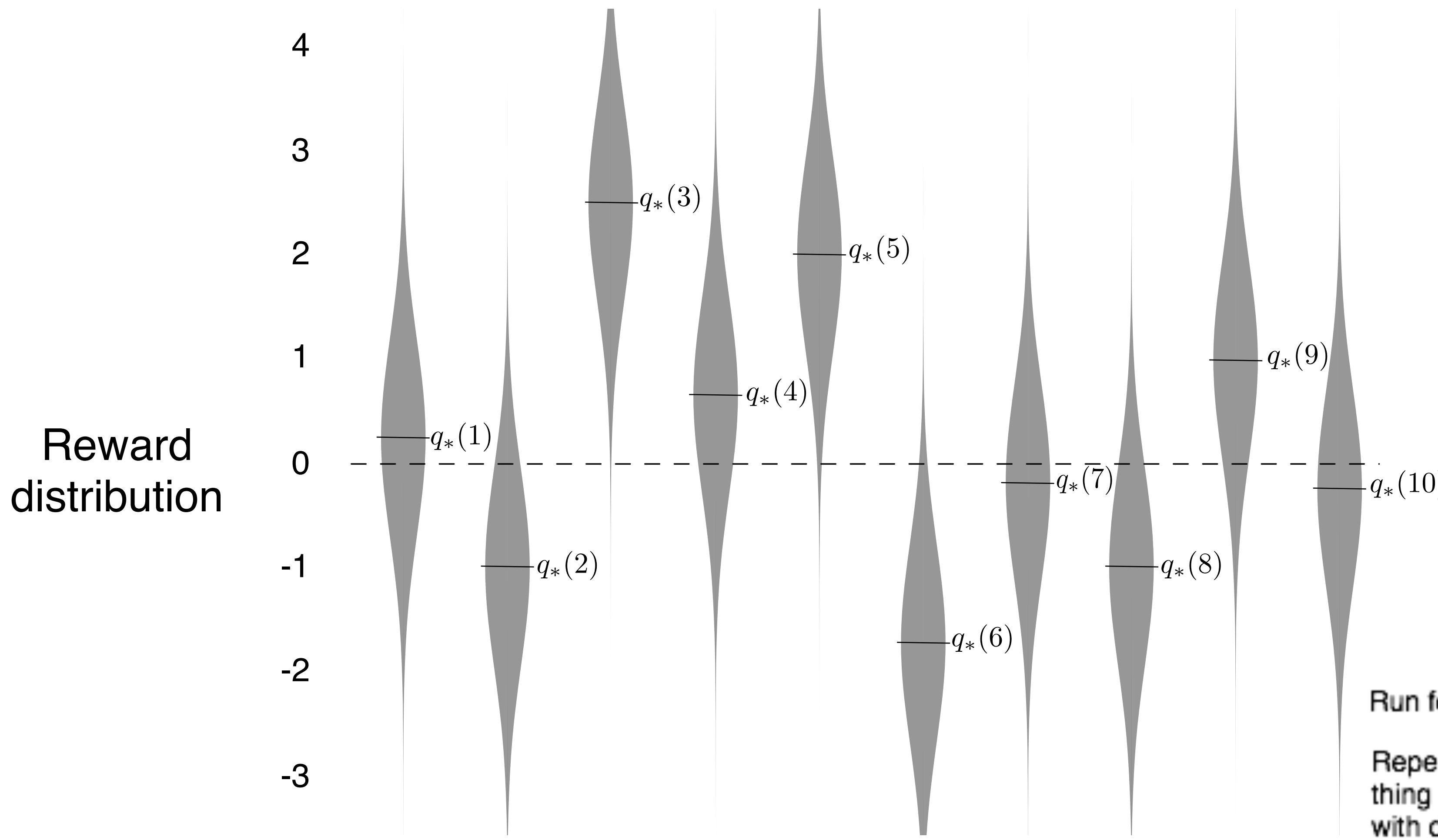
$$A \leftarrow \begin{cases} \arg\max_a Q(a) & \text{with probability } 1-\varepsilon \quad \text{(breaking ties randomly)} \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$$

$R \leftarrow bandit(A)$
$N(A) \leftarrow N(A) + 1$
$Q(A) \leftarrow Q(A) + \frac{1}{N(A)}\big[R - Q(A)\big]$

Worked Example: 10-Armed Testbed

- n = 10 possible actions
- Each $Q^*(a)$ is chosen randomly from a normal distrib.: $N(0,1)$
- Each r_t is also normal: $N(Q^*(a_t),1)$
- 1000 plays, repeat the whole thing 2000 times and average the results

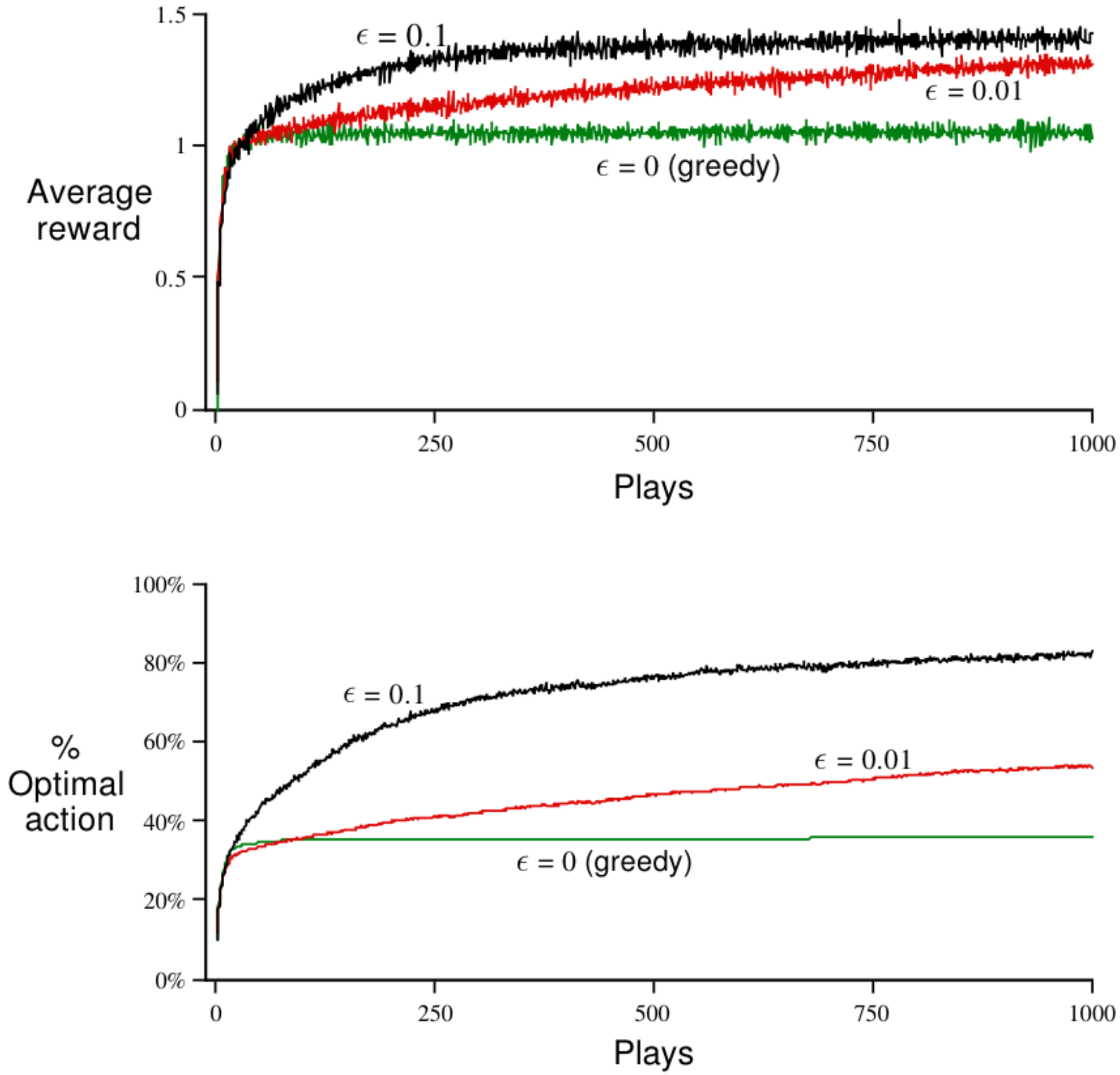
10-Armed Testbed Rewards



Run for 1000 steps

Repeat the whole thing 2000 times with different bandit tasks

ε-Greedy Methods on the 10-Armed Testbed



Incremental Implementation

Sample average estimation method:

The average of the first k rewards is (dropping the dependence on a):

$$Q_k = \frac{r_1 + r_2 + \cdots r_k}{k}$$

How to do this incrementally (without storing all the rewards)?

We could keep a running sum and count, or, equivalently:

$$Q_{k+1} = Q_k + \frac{1}{k+1}\left[r_{k+1} - Q_k\right]$$

NewEstimate **=** *OldEstimate* **+** *StepSize* [*Target* **–** *OldEstimate*]

Tracking a Non-stationary Problem

Choosing Q_k to be a sample average is appropriate in a stationary problem,

i.e., when none of the $Q^*(a)$ change over time,

But not in a nonstationary problem.

The better option in the nonstationary case is:

$$Q_{k+1} = Q_k + \alpha\left[r_{k+1} - Q_k\right]$$

for *constant* α, $0 < \alpha \leq 1$

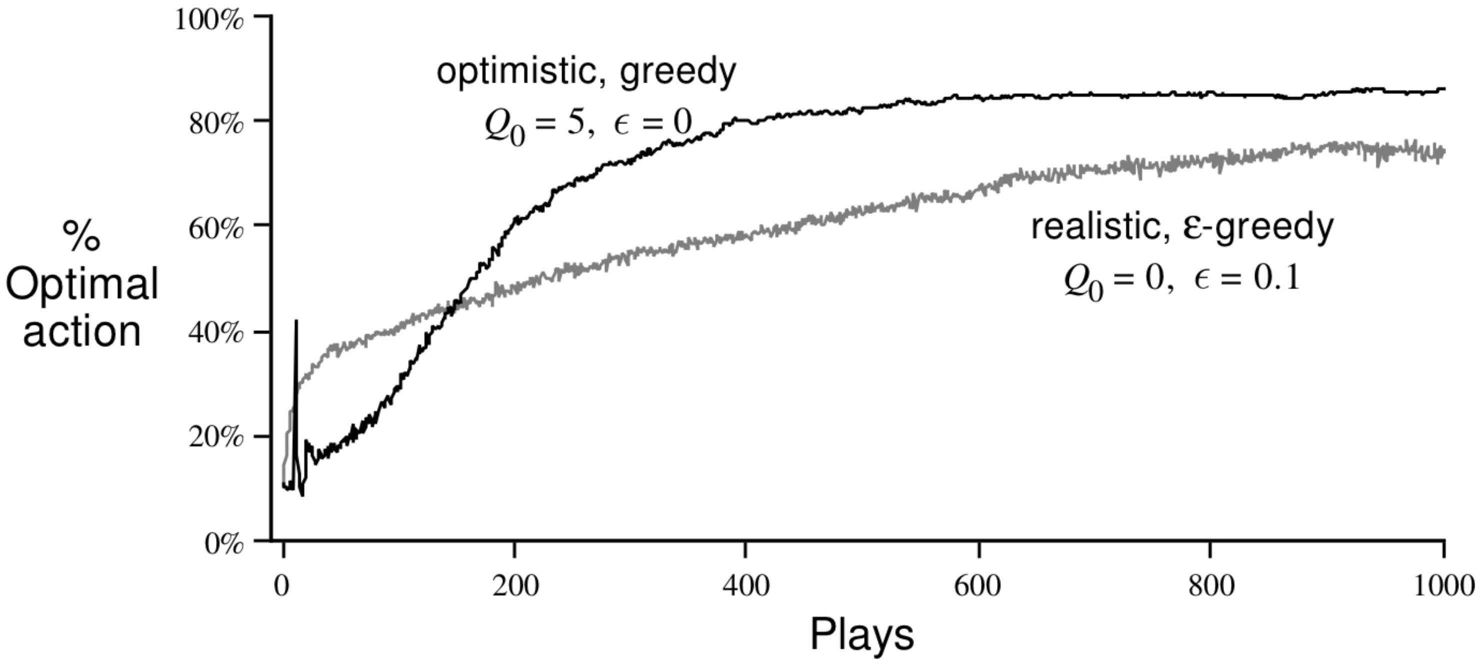
$$= (1-\alpha)^k Q_0 + \sum_{i=1}^{k} \alpha(1-\alpha)^{k-i} r_i$$

exponential, recency-weighted average

Optimistic Initial Values

- All methods so far depend on $Q_0(a)$, i.e., they are *biased*
- Encourage exploration: initialize the action values optimistically,

  i.e., on the 10-armed testbed, use $Q_0(a) = 5 \quad \text{for all } a$



Softmax Action Selection

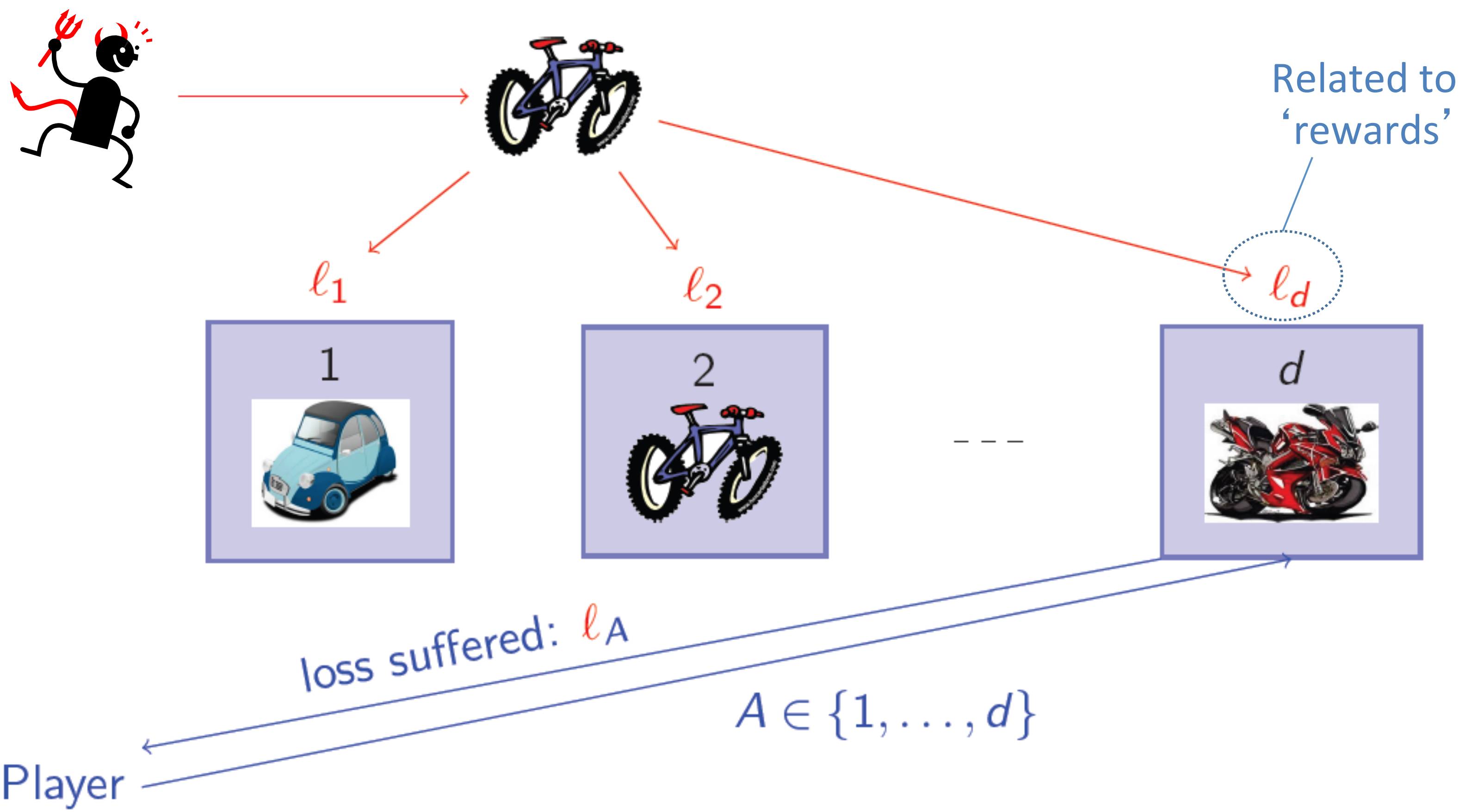
- Softmax action selection methods grade action probabilities by estimated values.
- The most common softmax uses a Gibbs, or Boltzmann, distribution:

Choose action a on play t with probability

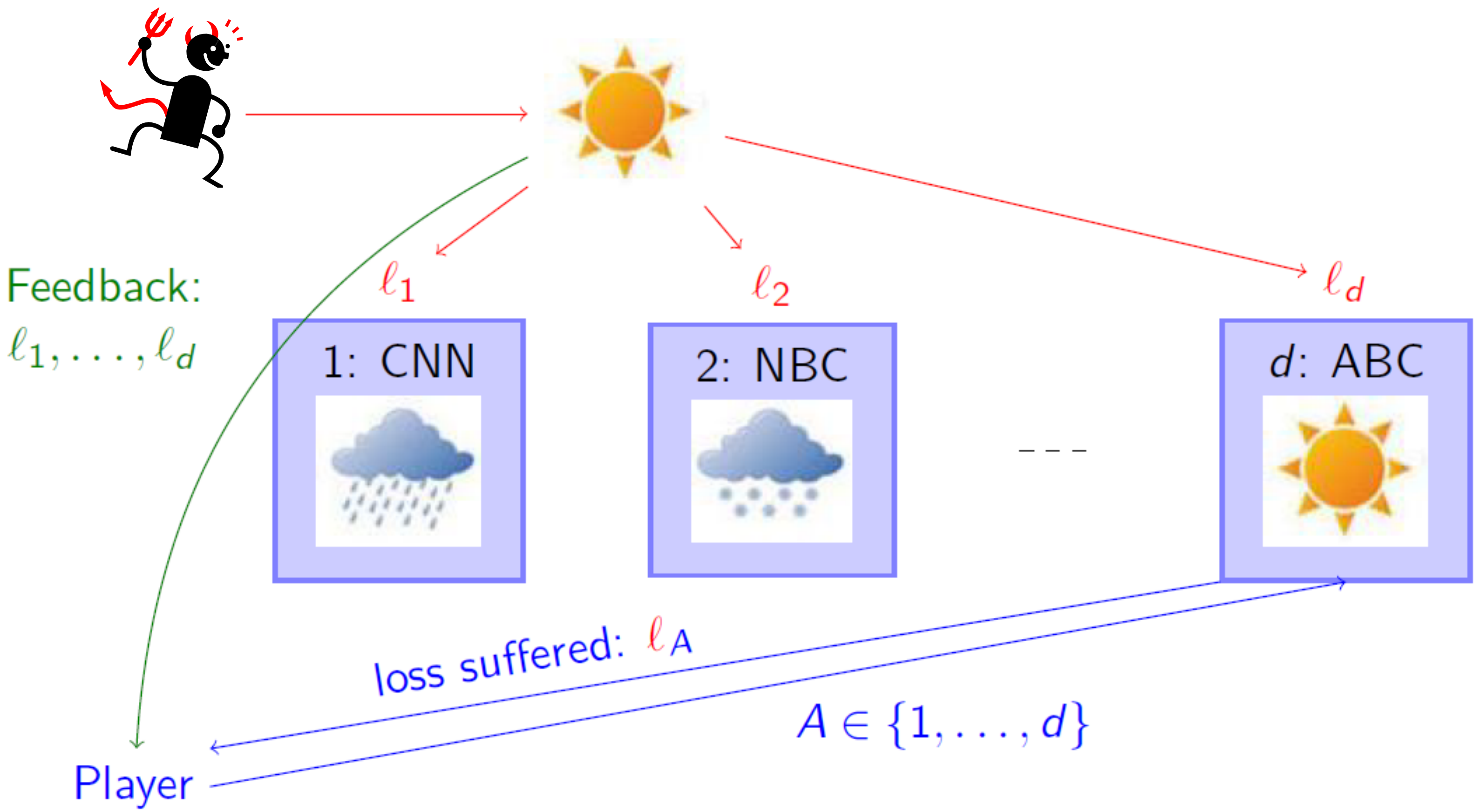
$$\frac{e^{Q_t(a)/\tau}}{\sum_{b=1}^{n} e^{Q_t(b)/\tau}},$$

where τ is a 'computational temperature'

Another Interpretation of MAB Problems



MAB is a Special Case of Online Learning



How to Evaluate Online Alg.: Regret

- After you have played for T rounds, you experience a regret:

  = [Reward sum of optimal strategy] – [Sum of actual collected rewards]

$$\rho = T\mu^* - \sum_{t=1}^{T} \hat{r}_t = T\mu^* - \sum_{t=1}^{T} E\left[r_{i_t}(t)\right]$$

$$\mu^* = \max_k \mu_k$$

Randomness in draw of rewards & Player's strategy

- If the average regret per round goes to zero with probability 1, asymptotically, we say the strategy has **no-regret** property

  ~ guaranteed to converge to an optimal strategy

- ε-greedy is sub-optimal (so has some regret). Why?

Interval Estimation

- Attribute to each arm an "optimistic initial estimate" within a certain confidence interval
- Greedily choose arm with highest optimistic mean (upper bound on confidence interval)

- Infrequently observed arm will have over-valued reward mean, leading to exploration
- Frequent usage pushes optimistic estimate to true values

Interval Estimation Procedure

- Associate to each arm 100(1-α)% reward mean upper band

- Assume, e.g., rewards are normally distributed
- Arm is observed n times to yield empirical mean & std dev
- α-upper bound: $$u_\alpha = \hat{\mu} + \frac{\hat{\sigma}}{\sqrt{n}} c^{-1}(1-\alpha)$$

$$c(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{t} \exp\left(-\frac{x^2}{2}\right) dx$$ Cum. Distribution Function

- If α is carefully controlled, could be made zero-regret strategy
  - In general (i.e., for other distributions), we don't know

Reminder: Chernoff-Hoeffding Bound

Let $X_1, X_2, \ldots, X_n$ be independent random variables in the range $[0, 1]$ with $\mathbb{E}[X_i] = \mu$. Then for $a > 0$,

$$P\left(\frac{1}{n}\sum_{i=1}^{n} X_i \geq \mu + a\right) \leq e^{-2a^2 n}$$

Variant: UCB Strategy

- Again, based on notion of an upper confidence bound but more generally applicable

- Algorithm:
  - Play each arm once
  - At time $t > K$, play arm i_t maximizing

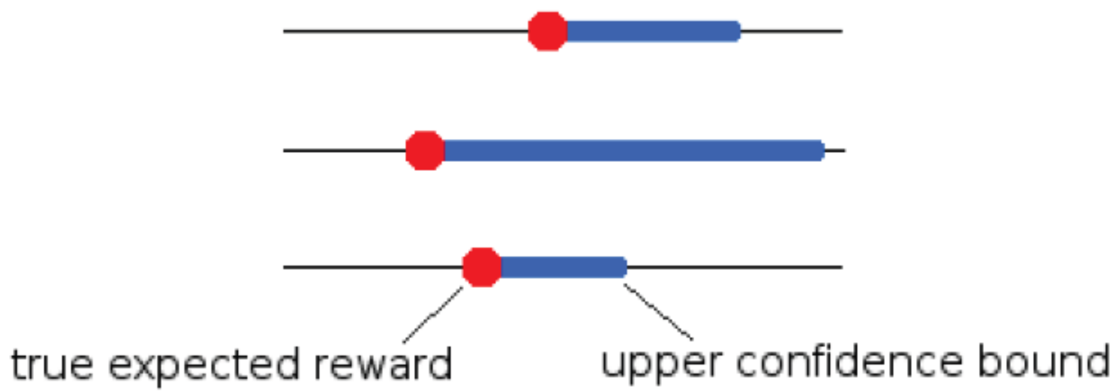
$$\bar{r}_j(t) + \sqrt{\frac{2 \ln t}{T_{j,t}}}$$

$T_{j,t}$: number of times j has been played so far

UCB Strategy

Intuition:

The second term $\sqrt{2 \ln t / T_{j,t}}$ is the the size of the one-sided $(1 - 1/t)$-condifence interval for the average reward (using Chernoff-Hoeffding bounds).



UCB Strategy – Behaviour

We will not prove the following result, but I quote the theorem to explain the benefit of UCB – regret is bounded.

Theorem

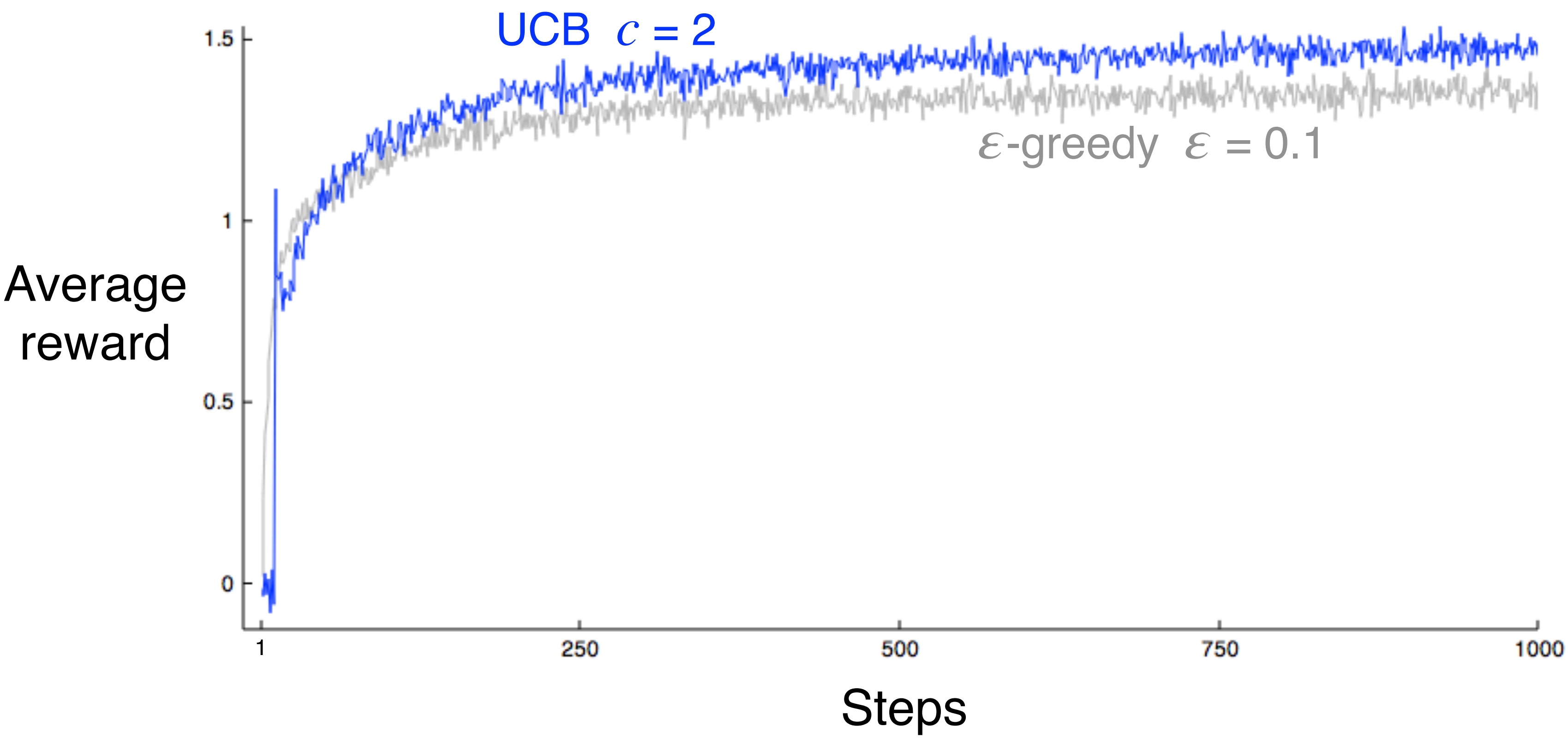
(Auer, Cesa-Bianchi, Fisher) At time T, the regret of the UCB policy is at most

$$\frac{8K}{\Delta^*} \ln T + 5K,$$

K = number of arms

where $\Delta^* = \mu^* - \max_{i:\mu_i<\mu^*} \mu_i$ (the gap between the best expected reward and the expected reward of the runner up).

Empirical Behaviour: UCB



Variation on SoftMax: $\frac{e^{Q_t(a)/\tau}}{\sum_{b=1}^{n} e^{Q_t(b)/\tau}}$

- It is possible to drive regret down by annealing τ
- Exp3 : Exponential weight alg. for exploration and exploitation
- Probability of choosing arm k at time t is

$$P_k(t) = (1-\gamma)\frac{w_k(t)}{\sum_{j=1}^{k} w_j(t)} + \frac{\gamma}{K}$$

γ is a user defined open parameter

$$w_j(t+1) = \begin{cases} w_j(t)\exp\left(\gamma \frac{r_j(t)}{P_j(t)K}\right) & \text{if arm } j \text{ is pulled at } t \\ w_j(t) & \textit{otherwise} \end{cases}$$

$$\text{Regret} \approx O\left(\sqrt{KT\log(K)}\right)$$

The Gittins Index

- Each arm delivers reward with a probability
- This probability may *change* through time but only when arm is pulled
- Goal is to maximize discounted rewards – future is discounted by an exponential discount factor $\delta < 1$
- The structure of the problem is such that, all you need to do is compute an "index" for each arm and play the one with the highest index
- Index is of the form:

$$\nu_i = \sup_{T>0} \frac{\left\langle \sum_{t=0}^{T} \delta^t R^i(t) \right\rangle}{\left\langle \sum_{t=0}^{T} \delta^t \right\rangle}$$

Gittins Index – Intuition

- We will not give a proof of its optimality now, and will return to that issue later in the course.
- Analysis is based on stopping time: the point where you should 'terminate' a bandit arm

- Nice Property: Gittins index for any given bandit is independent of expected outcome of all other bandits
  - Once you have a good arm, keep playing until there is a better one
  - If you add/remove machines, computation doesn't really change

BUT:

  - hard to compute, even when you know the distributions
  - Exploration issues; arms aren't updated unless used (restless bandits?)

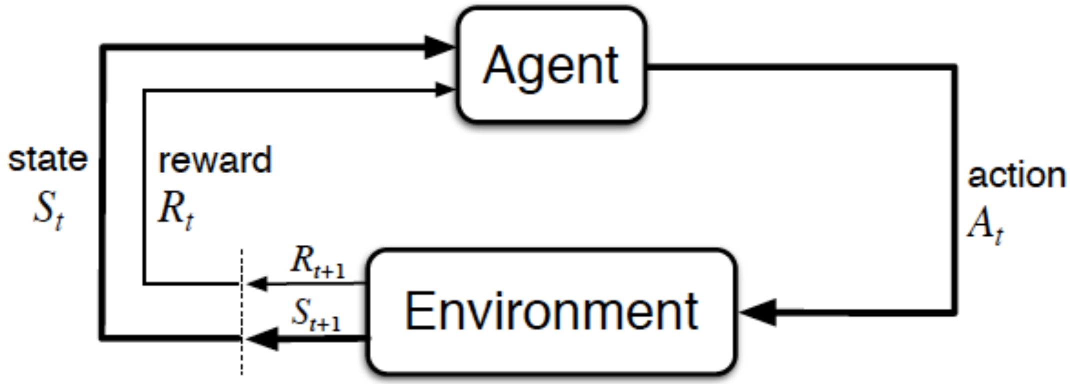
What About State Changes?

- In MAB, we were in a single casino and the only decision is to pull from a set of n arms
  - Some change, perhaps, if an adversary were introduced…

Next,

- What if there is **more than one** state?
- So, in this state space, what is the effect of the distribution of payout changing based on how you pull arms?
- What happens if you only obtain a net reward corresponding to a long sequence of arm pulls (at the end)?

Decision Making Agent-Environment Interface



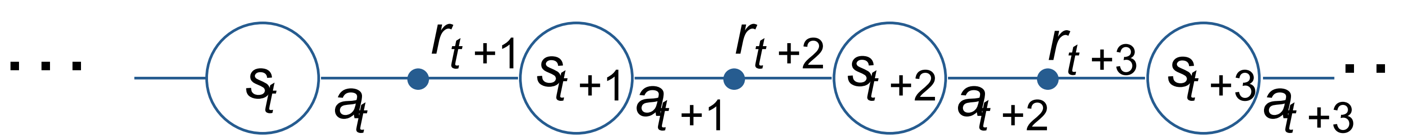
Agent and environment interact at discrete time steps: $t = 0, 1, 2, \ldots$

Agent observes state at step t: $s_t \in S$

produces action at step t : $a_t \in A(s_t)$

gets resulting reward: $r_{t+1} \in \Re$

and resulting next state: s_{t+1}



Markov Decision Processes

- A model of the agent-environment system
- ***Markov*** property = history doesn't matter, only current state
- If state and action sets are finite, it is a **finite MDP**.
- To define a finite MDP, you need to give:
  - **state and action sets**
  - one-step "dynamics" defined by **transition probabilities**:

$$\mathbf{P}_{ss'}^{a} = \Pr\left\{ s_{t+1} = s' \mid s_t = s, a_t = a \right\} \quad \text{for all } s, s' \in S, a \in A(s).$$

  - **reward probabilities**:

$$\mathbf{R}_{ss'}^{a} = E\left\{ r_{t+1} \mid s_t = s, a_t = a, s_{t+1} = s' \right\} \quad \text{for all } s, s' \in S, a \in A(s).$$

An Example Finite MDP

Recycling Robot

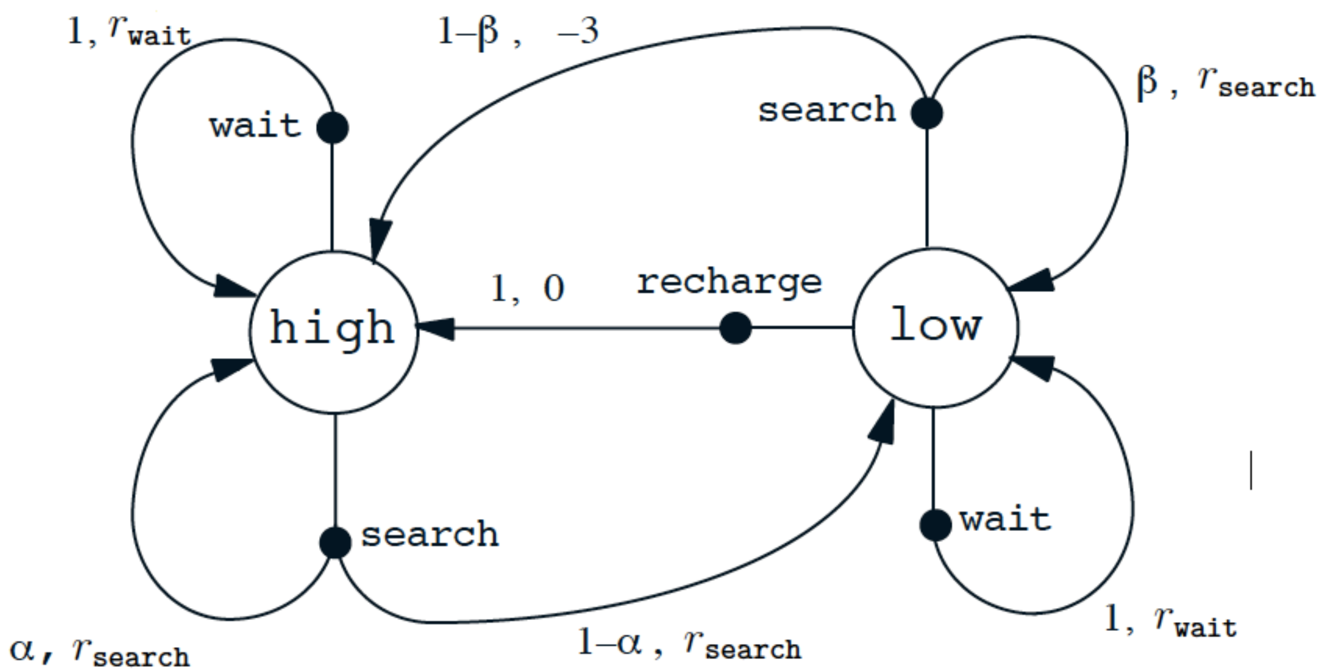
- At each step, robot has to decide whether it should (1) actively search for a can, (2) wait for someone to bring it a can, or (3) go to home base and recharge.
- Searching is better but runs down the battery; if it runs out of power while searching then it has to be rescued (which is bad).
- Decisions made on the basis of current energy level: **`high`**, **`low`**.
- <u>Reward</u> = number of cans collected

Recycling Robot MDP

$$\mathcal{A}(\texttt{high}) \doteq \{\texttt{search}, \texttt{wait}\}$$
$$\mathcal{A}(\texttt{low}) \doteq \{\texttt{search}, \texttt{wait}, \texttt{recharge}\}.$$

Rewards while searching/waiting :

$$r_{search} > r_{wait}$$



Enumerated In Tabular Form

s	s'	a	$p(s'\|s,a)$	$r(s,a,s')$
high	high	search	α	$r_{\texttt{search}}$
high	low	search	$1-\alpha$	$r_{\texttt{search}}$
low	high	search	$1-\beta$	-3
low	low	search	β	$r_{\texttt{search}}$
high	high	wait	1	$r_{\texttt{wait}}$
high	low	wait	0	$r_{\texttt{wait}}$
low	high	wait	0	$r_{\texttt{wait}}$
low	low	wait	1	$r_{\texttt{wait}}$
low	high	recharge	1	0
low	low	recharge	0	$0.$

If you were given this much, what can you say about the behaviour (over time) of the system?

Very Brief Primer on Markov Chains

Stochastic Processes

- A *stochastic process* is an indexed collection of random variables $\{X_t\}$
  - e.g., collection of weekly demands for a product
- One type: At a particular time t, labelled by integers, system is found in exactly one of a finite number of mutually exclusive and exhaustive categories or **states**, labelled by integers too
- Process could be *embedded* in that time points correspond to occurrence of specific events (or time may be equi-spaced)
- Random variables may depend on others, e.g.,

$$X_{t+1} = \{ \begin{array}{c} \max\{(3 - D_{t+1}), 0\}, if X_t < 0 \\ \max\{(X_t - D_{t+1}), 0\}, if X_t \geq 0 \end{array}$$

Markov Chains

- The stochastic process is said to have a **Markovian** property if

$$P\{X_{t+1} = j | X_0 = k_0, X_1 = k_1, ..., X_{t-1} = k_{t-1}, X_t = i\} = P\{X_{t+1} = j | X_t = i\}$$

for $t = 0, 1, ...$ and every sequence $i, j, k_0, ..., k_{t-1}$.

- Markovian property means that the *conditional probability* of a future event given any past events and current state, is *independent* of past states and depends only on present
- The conditional probabilities are **transition probabilities**,

$$P\{X_{t+1} = j | X_t = i\}$$

- These are stationary if time invariant, denote p_{ij},

$$P\{X_{t+1} = j | X_t = i\} = P\{X_1 = j | X_0 = i\}, \forall t = 0, 1, ...$$

Markov Chains

- Looking forward in time, n-step **transition probabilities**, $p_{ij}^{(n)}$

$$P\{X_{t+n} = j | X_t = i\} = P\{X_n = j | X_0 = i\}, \forall t = 0, 1, ...$$

- One can write a transition matrix,

$$\mathbf{P}^{(n)} = \begin{bmatrix} p_{00}^{(n)} & \cdots & p_{0M}^{(n)} \\ \vdots & & \\ p_{M0}^{(n)} & \cdots & p_{MM}^{(n)} \end{bmatrix}$$

- A stochastic process is a finite-state Markov chain if it has,
  – Finite number of states
  – Markovian property
  – Stationary transition probabilities
  – A set of initial probabilities $P\{X_0 = i\}$ for all i

Markov Chains

- n-step transition probabilities can be obtained from 1-step transition probabilities recursively (Chapman-Kolmogorov)

$$p_{ij}^{(n)} = \sum_{k=0}^{M} p_{ik}^{(v)} p_{kj}^{(n-v)}, \forall i, j, n; 0 \leq v \leq n$$

- We can get this via the matrix too

$$P^{(n)} = P.P \ldots P = P^n = PP^{n-1} = P^{n-1}P$$

- **First Passage Time**: number of transitions to go from i to j for the first time
  - If $i = j$, this is the **recurrence time**
  - In general, this itself is a random variable

Markov Chains

- n-step recursive relationship for first passage time

$$
\begin{aligned}
f_{ij}^{(1)} = p_{ij}^{(1)} &= p_{ij}, \\
f_{ij}^{(2)} = p_{ij}^{(2)} - f_{ij}^{(1)} &p_{jj}, \\
&\vdots \\
f_{ij}^{(n)} = p_{ij}^{(n)} - f_{ij}^{(1)} p_{jj}^{(n-1)} - f_{ij}^{(2)} p_{jj}^{(n-2)} \dots - f_{ij}^{(n-1)} &p_{jj}
\end{aligned}
$$

- For fixed i and j, these $f_{ij}^{(n)}$ are nonnegative numbers so that

$$\sum_{n=1}^{\infty} f_{ij}^{(n)} \leq 1$$

What does <1 signify?

- If $\sum_{n=1}^{\infty} f_{ii}^{(n)} = 1$,state is **recurrent**; if so for n=1 then state i is **absorbing**

Markov Chains: Long-Run Properties

- Consider this transition matrix of an inventory process:

$$P^{(1)} = P = \begin{bmatrix} 0.08 & 0.184 & 0.368 & 0.368 \\ 0.632 & 0.368 & 0 & 0 \\ 0.264 & 0.368 & 0.368 & 0 \\ 0.08 & 0.184 & 0.368 & 0.368 \end{bmatrix}$$

- This captures the evolution of inventory levels in a store
  - What do the 0 values mean?
  - Other properties of this matrix?

Markov Chains: Long-Run Properties

The corresponding 8-step transition matrix becomes:

$$P^{(8)} = P^8 = \begin{bmatrix} 0.286 & 0.285 & 0.264 & 0.166 \\ 0.286 & 0.285 & 0.264 & 0.166 \\ 0.286 & 0.285 & 0.264 & 0.166 \\ 0.286 & 0.285 & 0.264 & 0.166 \end{bmatrix}$$

Interesting property: probability of being in state j after 8 weeks appears independent of *initial* level of inventory.

- For an irreducible ergodic Markov chain, one has limiting probability

$$\lim_{n \to \infty} p_{ij}^{(n)} = \pi_j$$

Reciprocal gives you recurrence time

$$\pi_j = \sum_{i=0}^{M} \pi_i p_{ij}, \forall j = 0, ..., M$$

Markov **Decision** Model

- Consider the following application: machine maintenance
- A factory has a machine that deteriorates rapidly in quality and output and is inspected periodically, e.g., daily
- Inspection declares the machine to be in four possible states:
  - 0: Good as new
  - 1: Operable, minor deterioration
  - 2: Operable, major deterioration
  - 3: Inoperable
- Let X_t denote this observed state
  - evolves according to some "law of motion", it is a stochastic *process*
  - Furthermore, assume it is a finite state Markov chain

Markov **Decision** Model

- Transition matrix is based on the following:

States	0	1	2	3
0	0	7/8	1/16	1/16
1	0	3/4	1/8	1/8
2	0	0	1/2	1/2
3	0	0	0	1

- Once the machine goes inoperable, it stays there until repairs
  - If no repairs, eventually, it reaches this state which is absorbing!
- Repair is an **action** – a very simple maintenance **policy**.
  - e.g., machine from from state 3 to state 0

Markov **Decision** Model

- There are costs as system evolves:
  - State 0: cost 0
  - State 1: cost 1000
  - State 2: cost 3000
- Replacement cost, taking state 3 to 0, is 4000 (and lost production of 2000), so cost = 6000
- The modified transition probabilities are:

States	0	1	2	3
0	0	7/8	1/16	1/16
1	0	3/4	1/8	1/8
2	0	0	1/2	1/2
3	1	0	0	0

Markov **Decision** Model

- Simple question (a behavioural property):
  What is the average cost of this maintenance <u>policy</u>?

- Compute the steady state probabilities:

$$\pi_0 = \frac{2}{13}; \pi_1 = \frac{7}{13}; \pi_2 = \frac{2}{13}; \pi_3 = \frac{2}{13}$$

How?

- (Long run) expected average cost per day,

$$0\pi_0 + 1000\pi_1 + 3000\pi_2 + 6000\pi_3 = \frac{25000}{13} = 1923.08$$

Markov **Decision** Model

- Consider a slightly more elaborate policy:
  - Replace when inoperable but if only needing major repairs, overhaul
- Transition matrix now changes a little bit
- Permit one more thing: overhaul
  - Go back to minor repairs state (1) for the next time step
  - Not possible if truly inoperable, but can go from major to minor
- Key point about the system behaviour. It evolves according to
  - "Laws of motion"
  - Sequence of decisions made (actions from {1: none,2:overhaul,3: replace})
- Stochastic process is now defined in terms of $\{X_t\}$ and $\{\Delta_t\}$
  - Policy, R, is a rule for making decisions
    - Could use all history, although popular choice is (current) state-based

Markov **Decision** Model

- There is a space of potential policies, e.g.,

Policies	$d_0(R)$	$d_1(R)$	$d_2(R)$	$d_3(R)$
R_a	1	1	1	3
R_b	1	1	2	3
R_c	1	1	3	3
R_d	1	3	3	3

- Each policy defines a transition matrix, e.g., for R_b

States	0	1	2	3
0	0	7/8	1/16	1/16
1	0	3/4	1/8	1/8
2	0	1	0	0
3	1	0	0	0

Which policy is best?
Need costs….

Markov **Decision** Model

- C_{ik} = expected cost incurred during next transition if system is in state i and decision k is made

State	Dec.	1	2	3
0		0	4	6
1		1	4	6
2		3	4	6
3		∞	∞	6

- The long run average expected cost for each policy may be computed using,

$$E(C) = \sum_{i=0}^{M} C_{ik} \pi_i$$

R_b is best:
Work out details at home.

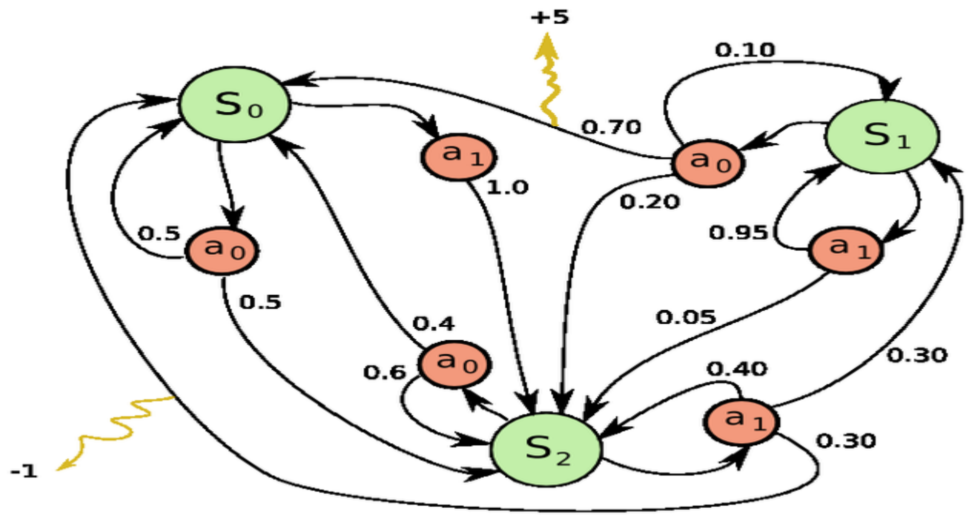
So, What is a Policy?

- A “program”
- Map from states (or situations in the decision problem) to actions that could be taken
  - e.g., if in ‘level 2’ state, call contractor for overhaul
  - If less than 3 DVDs of a film, place an order for 2 more

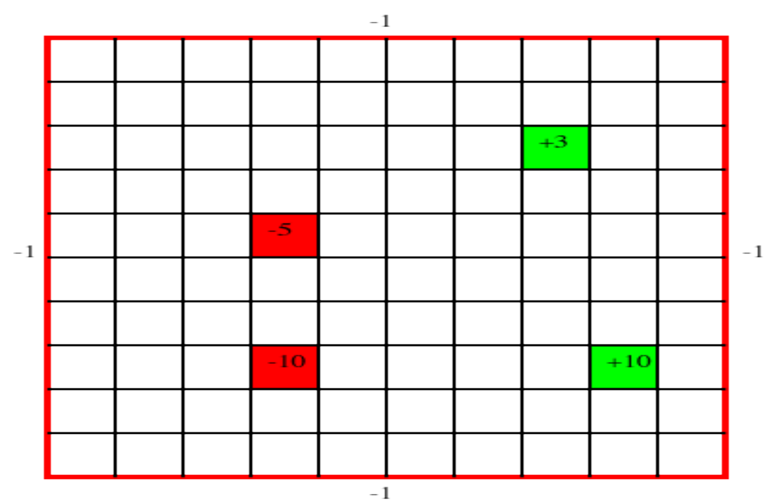
- A probability distribution $\pi(x,a)$
  - A joint probability distribution over states and actions
  - If in a state x_1, then with probability defined by π, take action a_1

Markov Decision Processes

- ‘Static’ view:

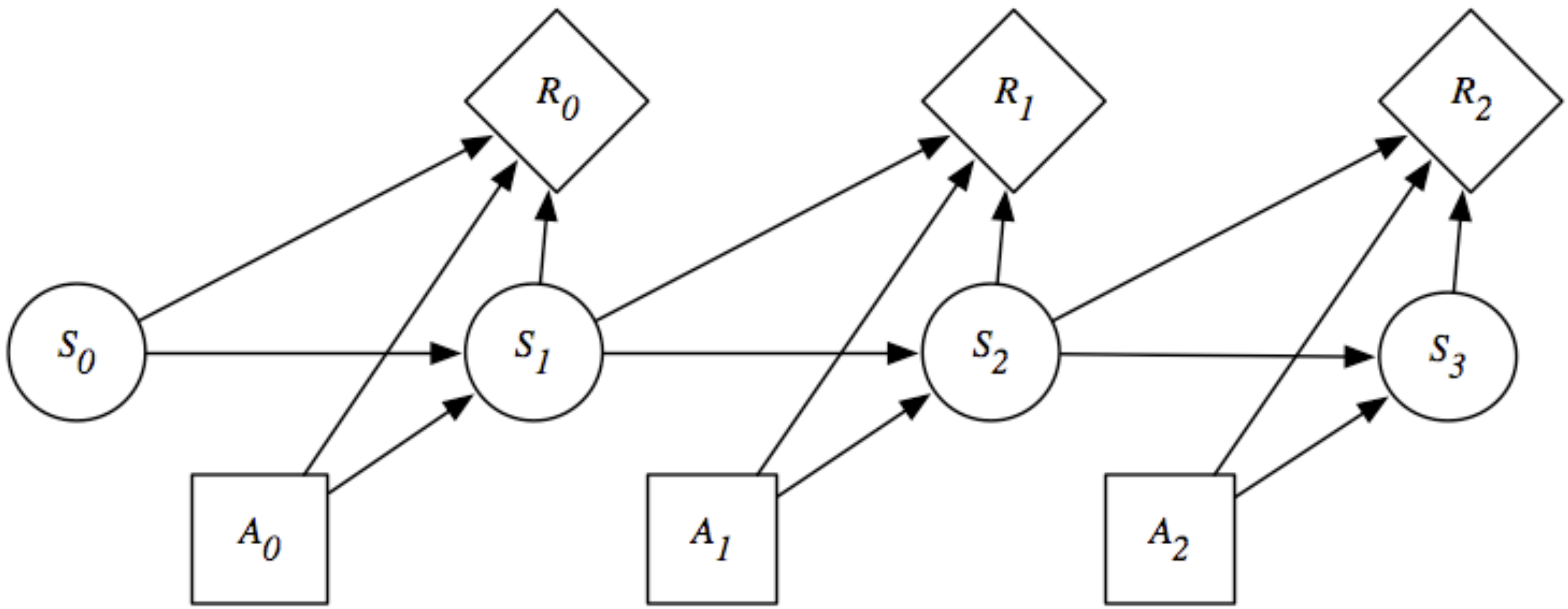


- Example:



Notation:
State ⇔ s/x

MDPs as Bayesian Networks



A Decision Criterion

- The general approach, that computationally implements the previous calculations with simultaneous equations over probabilities is linear programming
- Another approach to dealing with MDPs is via 'learning'
  - Often, treating the discounted, episodic setting
- What is the criterion for adaptation (i.e., learning)?

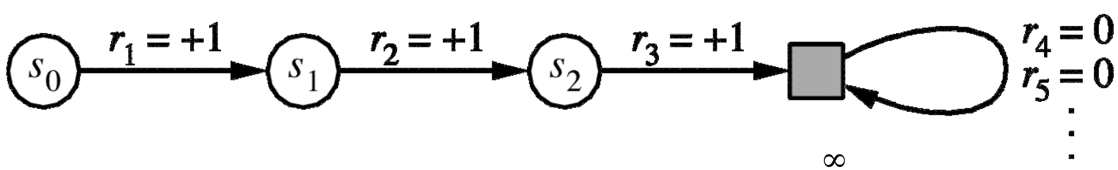
$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1},$$

where $\gamma, 0 \leq \gamma \leq 1,$ is the **discount rate**.

Effect of changing γ?

Episodic vs. Infinite: A Unified Notation

- In (discrete) episodic tasks, we could number the time steps of each episode starting from zero.
- We usually do not have to distinguish between episodes, so we write s_t instead of $s_{t,j}$ for the state at step *t* of episode *j*.
- Think of each episode as ending in an absorbing state that always produces reward of zero:



- We can cover <u>all</u> cases by writing $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$,

where γ can be 1 only if a zero reward absorbing state is always reached.

Acknowledgements

- The Markov Chains and MDP formulation slides are adapted from chapters in F.S. Hillier & G.J. Lieberman, Operations Research, 2nd ed.
- Initial slides on MAB and some later slides on reinforcement learning formulation are adapted from web resources associated with Sutton and Barto's book.