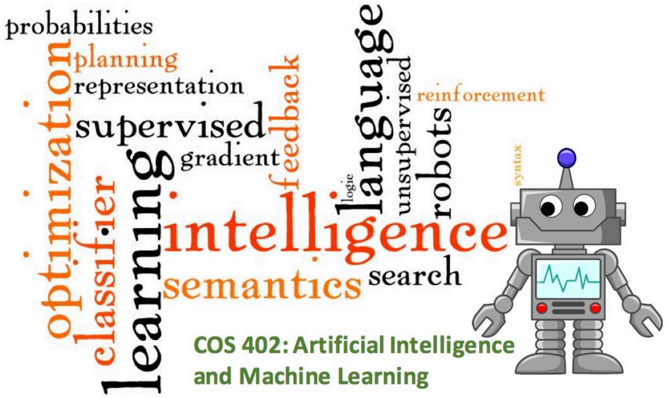


COS 402 – Machine Learning and Artificial Intelligence
Fall 2016

Lecture 19: Reinforcement Learning – part II (RL algorithms)

Sanjeev Arora Elad Hazan

PRINCETON UNIVERSITY

Admin

- (programming) exercise MCMC – due next class
- We will have at least 1 more exercise on RL
- Last lecture of the course: “ask us anything”, Prof. Arora + myself. Exercise: submit a question the lecture before (graded)

Markov Decision Process

Markov Reward Process, definition:

- Tuple (S, P, R, A, γ) where
  - S = states, including start state
  - A = set of possible actions
  - P = transition matrix $P_{ss'}^{a} = \Pr[S_{t+1} = s' | S_t = s, A_t = a]$
  - R = reward function, $R_s^a = E[R_{t+1} | S_t = s, A_t = a]$
  - $\gamma \in [0,1]$ = discount factor

- Return

$$G_t = \sum_{i=1\ to\ \infty} R_{t+i} \gamma^{i-1}$$

- Goal: take actions to maximize expected return

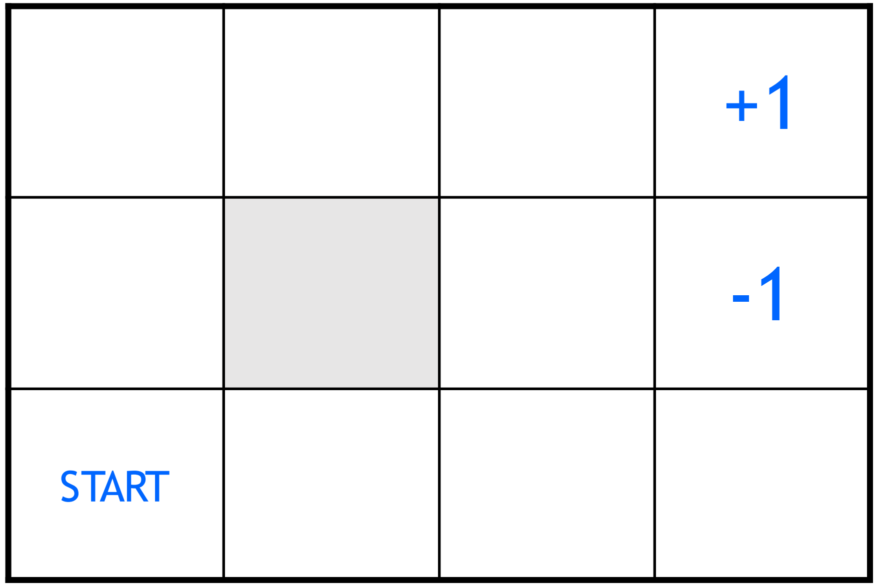
Policies

The Markovian structure ➔ best action depends only on current state!

- Policy = mapping from state to distribution over actions
  $\pi: S \mapsto \Delta(A),\ \pi(a|s) = \Pr[A_t = a | S_t = s]$

- Given a policy, the MDP reduces to a Markov Reward Process

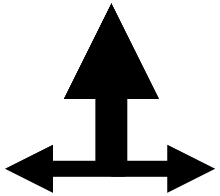
Reminder: MDP1



actions: UP, DOWN, LEFT, RIGHT

UP

80% move UP
10% move LEFT
10% move RIGHT



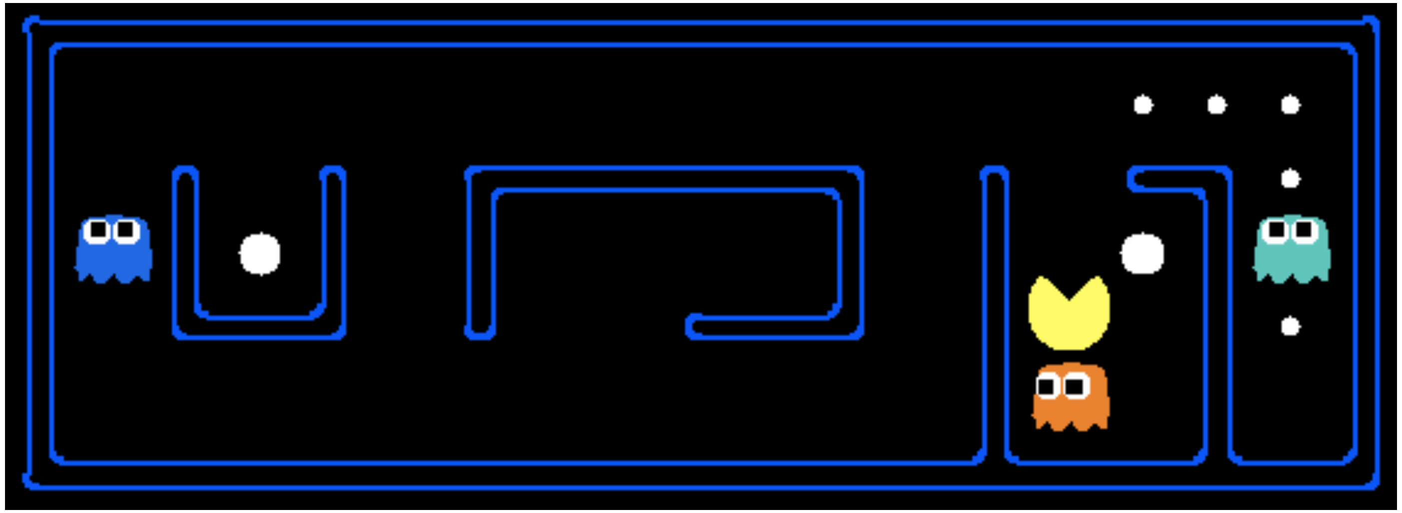
reward +1 at [4,3], -1 at [4,2]
reward -0.04 for each step

- states
- actions
- rewards

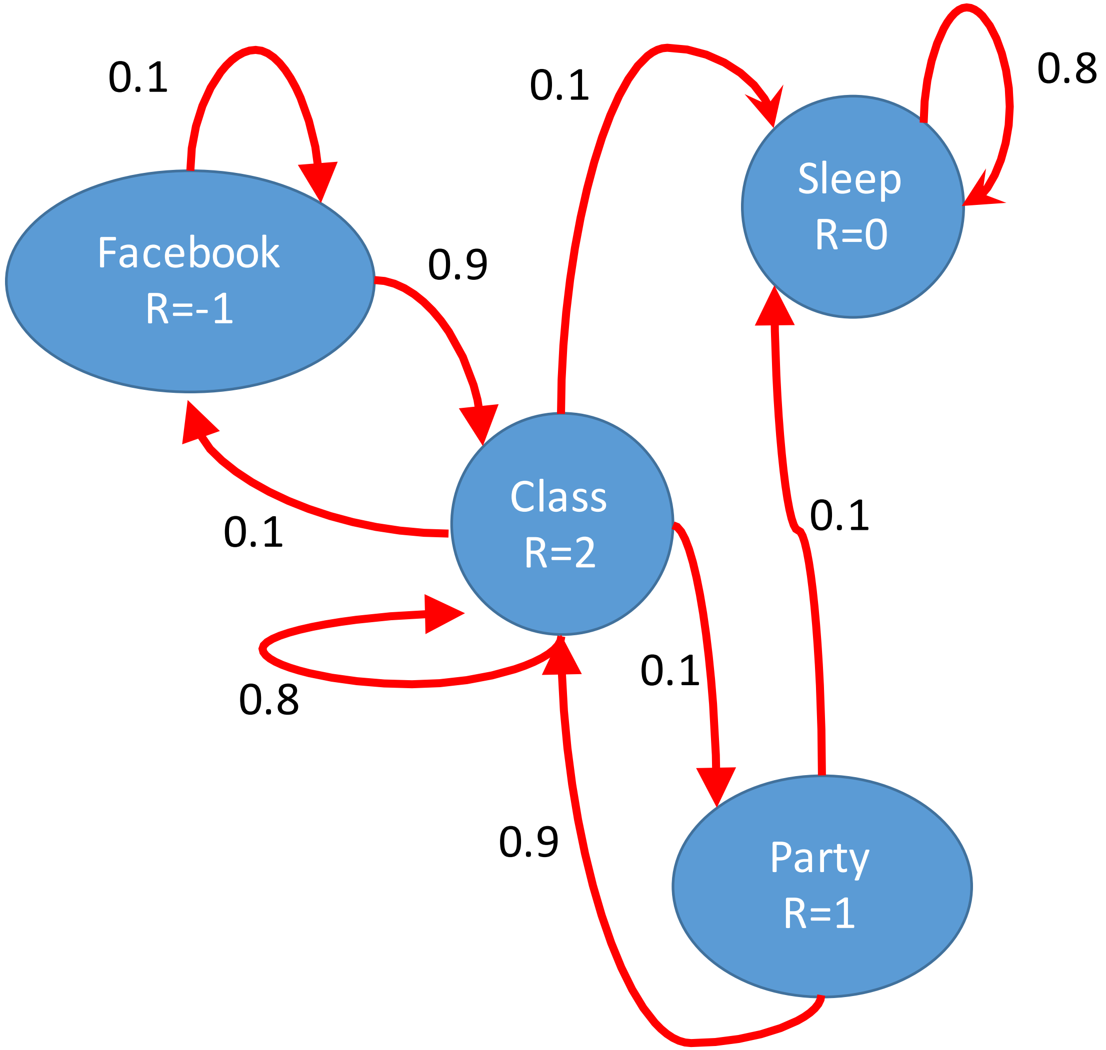
- Policies?

Reminder 2

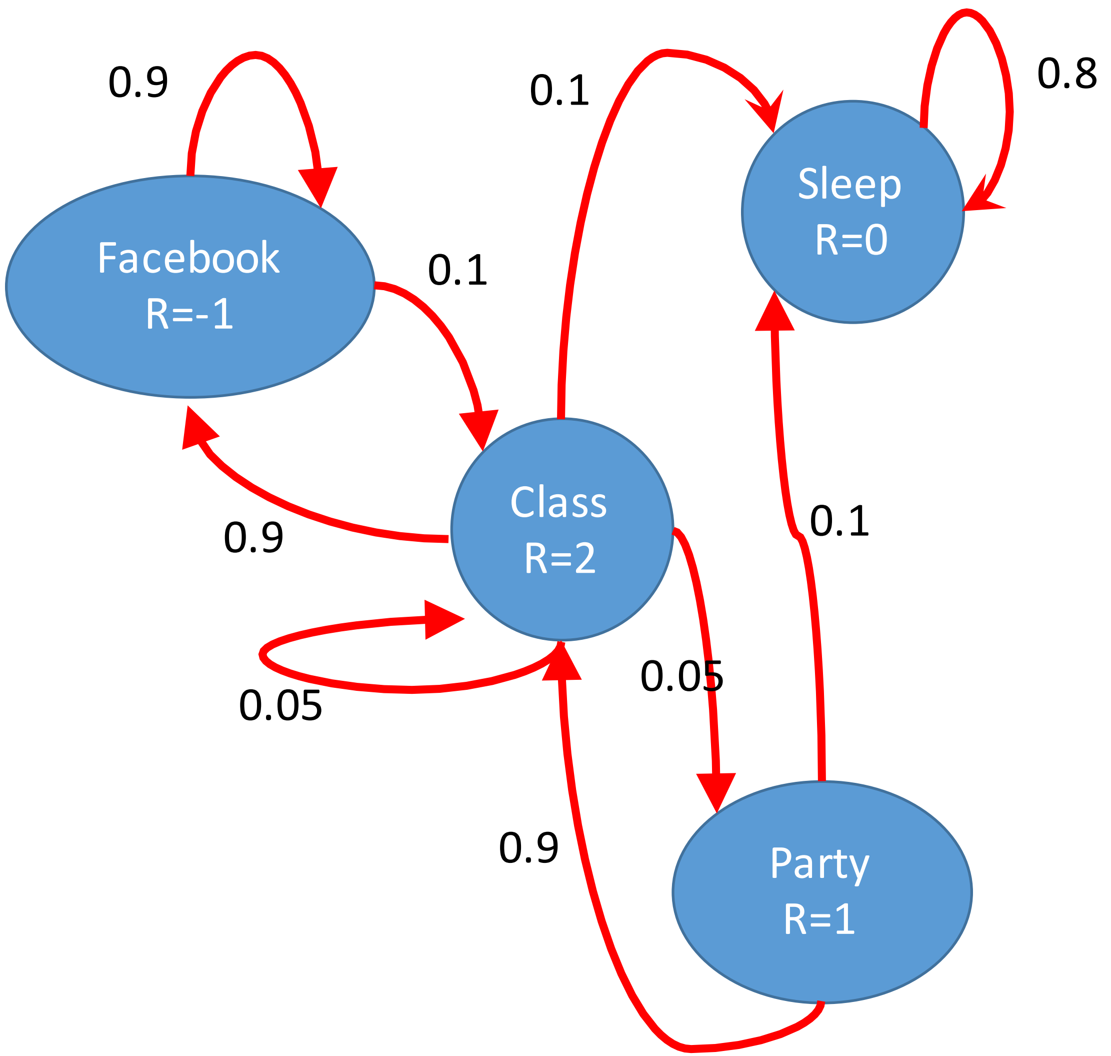
- State? Actions? Rewards? Policy?



Policies: action = study



Policies: action = facebook



Fixed policy → Markov Reward process

- Given a policy, the MDP reduces to a Markov Reward Process

- $P^{\pi}_{ss'} = \sum_{a \in A} \pi(a|s)\, P^{a}_{ss'}$
- $R^{\pi}_{s} = \sum_{a \in A} \pi(a|s)\, R^{a}_{s}$
- Value function for policy: $v_{\pi}(s) = E_{\pi}[G_t | S_t = s]$
- Action-Value function for policy: $q_{\pi}(s, a) = E_{\pi}[G_t | S_t = s, A_t = a]$

- How to compute the best policy?

The Bellman equation

- Policies satisfy the Bellman equation:

$$v_\pi(s) = E_\pi[R_{t+1} + \gamma\ v_\pi(S_{t+1})|\ S_t = s] = R_s^\pi + \gamma \sum_s P_{ss'}^\pi v_\pi(s')$$

- And similarly for value-action function:

$$v_\pi(s) = \sum_{a \in A} \pi(a|s) q_\pi(s, a)$$

- Optimal value function, and value-action function
- $v_*(s) = \max_\pi \{v_\pi(s)\}$ $\qquad q_*(s, a) = \max_\pi \{q_\pi(s, a)\}$

- Important: $v_*(s) = \max_a q_*(s, a)$, why?

Bellman optimality equations

- There exists an optimal policy π_* (it is deterministic!)
- All optimal policy achieve the same optimal value $v_*(s)$ at every state, and the same optimal value-action function $q_*(s,a)$ at every state and for every action.

- How can we find it? Bellman equation: $v_*(s) = \max_a \{q_*(s,a)\}$ implies **Bellman optimality equations**: (why?)

$$q_*(s,a) = R_s^a + \gamma \sum_{s'} P_{ss'}^a \max_{a'} \{q_*(s',a')\}$$

$$v_*(s) = \max_a \left\{ R_s^a + \gamma \sum_{s'} P_{ss'}^a v_*(s') \right\}$$

Non-linear! Solution → optimal policy. why? (later)

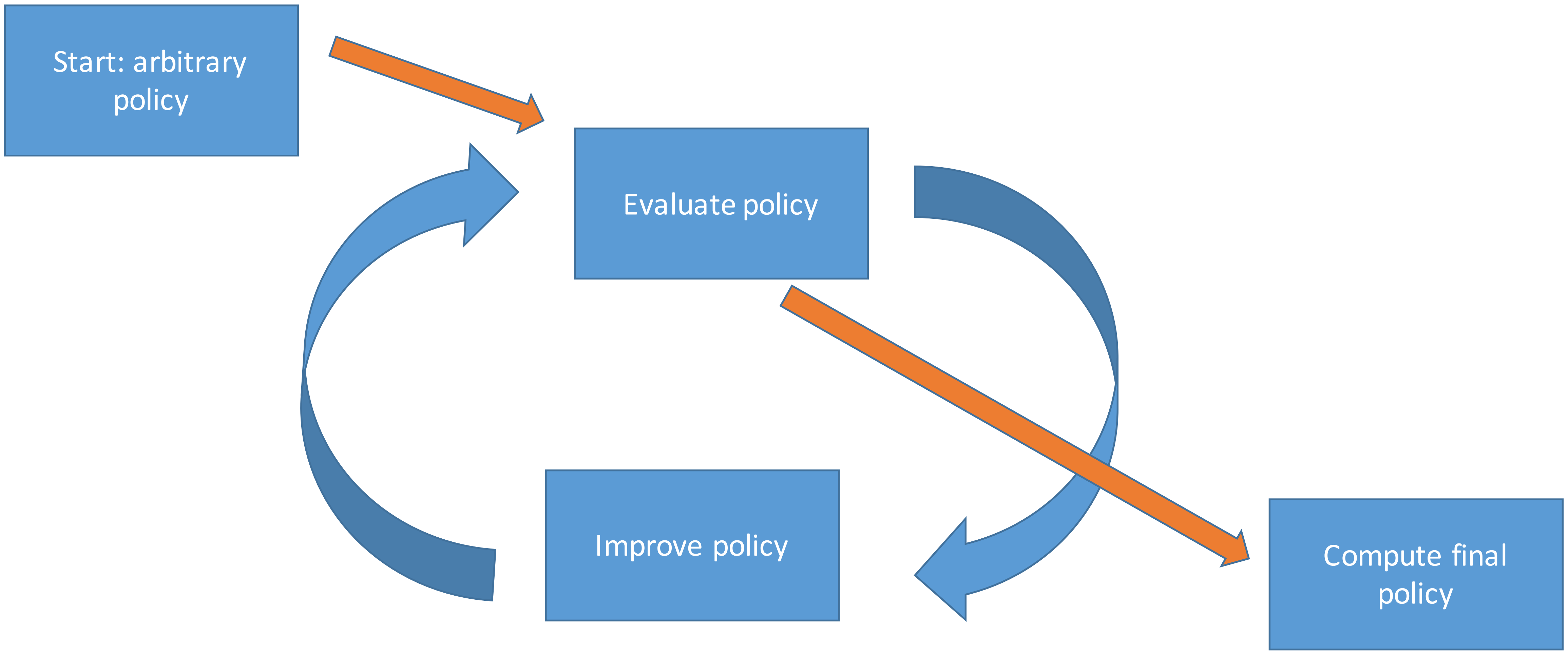
Algorithms – finding an optimal policy

$$q_*(s,a) = R_s^a + \gamma \sum_{s'} P_{ss'}^a \max_{a'} \{q_*(s',a')\}$$

$$v_*(s) = \max_a \left\{ R_s^a + \gamma \sum_{s'} P_{ss'}^a v_*(s') \right\}$$

- Iterative methods based on the Bellman equations: dynamic programming
  - Policy iteration
  - Value iteration

Policy iteration



Policy iteration

- Start with arbitrary policy $\pi_0 : S \mapsto A$
- While (not converged to optimality) do:
  - Evaluate current policy

$$v_k = R^{\pi_k} + \gamma\, P^{\pi_k} v_k$$

  - Improve policy by greedy step (from some or all states)

$$\pi_{k+1}(s) = \underset{a \in A}{\operatorname{argmax}} \left\{ R_s^a + \gamma \sum_{s'} P_{ss'}^a v_k(s') \right\}$$

Policy iteration

- Start with arbitrary policy $\pi_0 : S \mapsto A$
- While (not converged to optimality) do:
  - Evaluate current policy

$$v_k = R^{\pi_k} + \gamma\, P^{\pi_k} v_k$$

  - Improve policy by greedy step (from some or all states)

$$\pi_{k+1}(s) = \operatorname*{argmax}_{a \in A} \left\{ R_s^a + \gamma \sum_{s'} P_{ss'}^a v_k(s') \right\}$$

$$= \operatorname*{argmax}_{a \in A} \{ q^\pi(s,a) \}$$

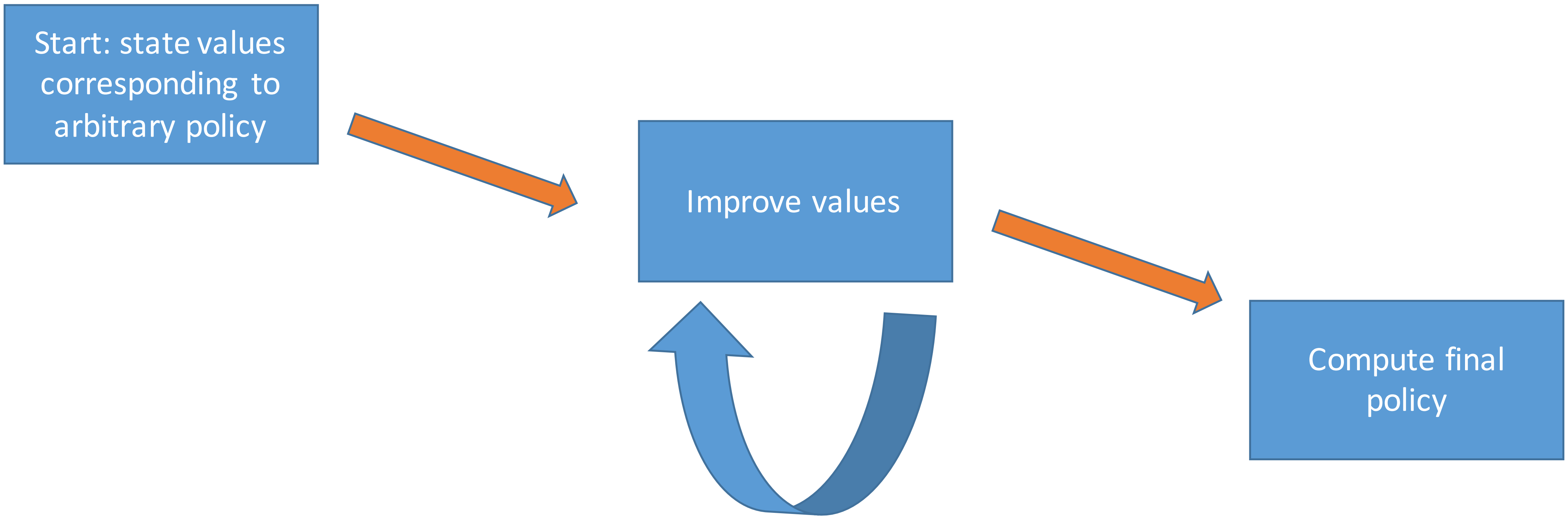
(compute $q^\pi(s,a)$ from $v^\pi(s)$)

  - Measure of distance to optimality, e.g., $|v_{k+1} - v_k|_\infty$

Policy iteration

- Intuitive
- Provably converging (prove?)
- Computationally somewhat expensive
- Better method with easier convergence analysis next…

Value iteration



Value iteration

$$v_{k+1}(s) = \max_{a \in A} \left\{ R_s^a + \gamma \sum_{s'} P_{ss'}^a v_k(s') \right\}$$

In matrix form:

$$v_{k+1} = \max_{\vec{a}} \left\{ R + \gamma \, P^{\vec{a}} v_k \right\}$$

- Initialize, $v_0(s) = 1$
- While $|v_{k+1} - v_k|_\infty > \epsilon$ do:
  - Update for all states $v_{k+1}(s)$ from $v_k(s)$
- Compute (near) optimal policy $\pi(s) = \underset{a \in A}{\operatorname{argmax}} \left\{ R_s^a + \gamma \sum_{s'} P_{ss'}^a v_k(s') \right\}$

Value iteration

- Faster computationally (no policy till the end)
- Easier convergence analysis:

Theorem: v^* is unique, and value Iteration converges geometrically:

$$|v_{k+1} - v^*|_\infty \leq \gamma^k |v_1 - v^*|_\infty$$

Value iteration - proof

Define the operator T as

$$T(v) = \max_{\vec{a}}\{R + \gamma\, P^{\vec{a}} v\}$$

Then,

$$T(v^*) = v^*$$

And,

$$|v_{k+1} - v^*|_\infty = |T(v_k) - T(v^*)|_\infty$$

$$= \left|\max_{\vec{a}}\{R + \gamma\, P^{\vec{a}} v_k\} - \max_{\vec{a}}\{R + \gamma\, P^{\vec{a}} v^*\}\right|_\infty$$

$$\leq \max_{\vec{a}}\left|\{R + \gamma\, P^{\vec{a}} v_k\} - \{R + \gamma\, P^{\vec{a}} v^*\}\right|_\infty$$

$$= \gamma\left|P^{\vec{a}}(v_k - v^*)\right|_\infty \leq \ \gamma|v_k - v^*|_\infty$$

|max$_x$ f(x) – max$_y$ g(y)|
<= max$_z$ |(f-g)(z)|

P is a tranition matrix

Value iteration - proof

Thus,

$$|v_{k+1} - v^*|_\infty \leq \gamma |v_k - v^*|_\infty$$

And recursively

$$|v_{k+1} - v^*|_\infty \leq \gamma^k |v_1 - v^*|_\infty$$

Uniqueness of v* ?

Value and policy iteration

Exercise: give running time bounds in terms of # of states, actions

Exact computation of optimal policy/values?

$$v_*(s) = \max_a \left\{ R_s^a + \gamma \sum_{s'} P_{ss'}^a v_*(s') \right\}$$

Equivalent to:

$$\forall a \in A \ . \ v_* \geq R^a + \gamma \, P^a v_*$$

Linear program!

Reminder: linear programming

formulation:

$$A \in R^{m \times n} \;, x \in R^n, b \in R^m \;, m \geq n$$
$$Ax \geq b$$

Special case of convex programming/optimization. Admits polynomial time algorithms (roughly $O(\sqrt{m}\; n^3)$), and practical algorithms such as the simplex.

Admits approximation algorithms that run in linear time, or even sublinear time $O\left(\frac{m+n}{\epsilon^2}\right)$.

MDP LP:

$$\forall a \in A \;.\;\; v_* \geq R^a + \gamma\; P^a v_*$$

So for us, n=|S|, m=|A|*|S|

If $\gamma = 1$?

Model-free RL

Thus far: assumed we know transition matrices, rewards, states, and they are not too large.

What if transitions/rewards are:

1. unknown
2. too many to keep in memory / compute over

"model free" = we do not have the "model" = transition matrix P and reward vector R

Model-free RL

Monte-carlo policy evaluation: instead of computing, estimate $v_\pi(s) = E_\pi[G_t|S_t = s]$ by random walk:

- The first time state s is visited, update counter N(s) (increment every time it's visited again)
- Keep track of all rewards from this point onwards
- Estimate of G_t is sum of rewards / N(s).
- Claim: this estimator has expectation $v_\pi(s)$, and converges to it by law of large numbers
- Similarly can estimate value-action function $q_\pi(s,a) = E_\pi[G_t|S_t = s, A_t = a]$

- What do we do with estimated values?
  - policy iteration requires rewards+transitions
  - Model-free policy improvement:

$$\pi(s) = \arg\max_a \{q_\pi(s,a)\}$$

Summary

- Algorithms for solving MDPs based on dynamic programming (Bellman equation)
  - Value iteration, policy iteration
- Proved convergence, convergence rate
- Linear programming view
- Partial observation – estimation of the state-action function (model free)
- Next: function approximation