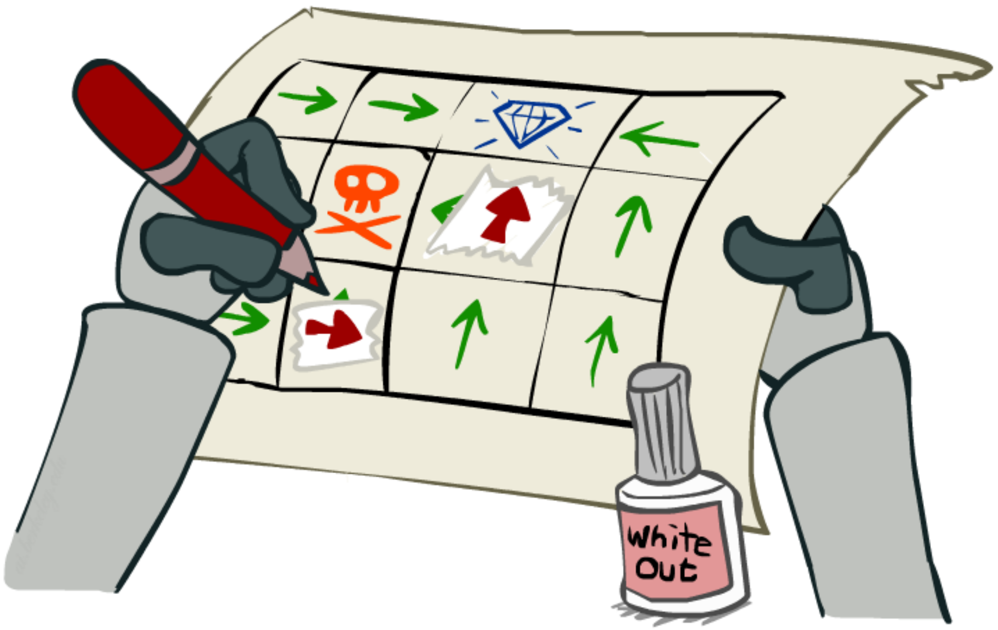
Advanced Topics in AI

Policy Iteration



Instructor: Prof. Dr. techn. Wolfgang Nejdl

Leibniz University Hannover

[These slides were created by Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley. All materials are available at http://ai.berkeley.edu.]

Co-financed by the Connecting Europe Facility of the European Union

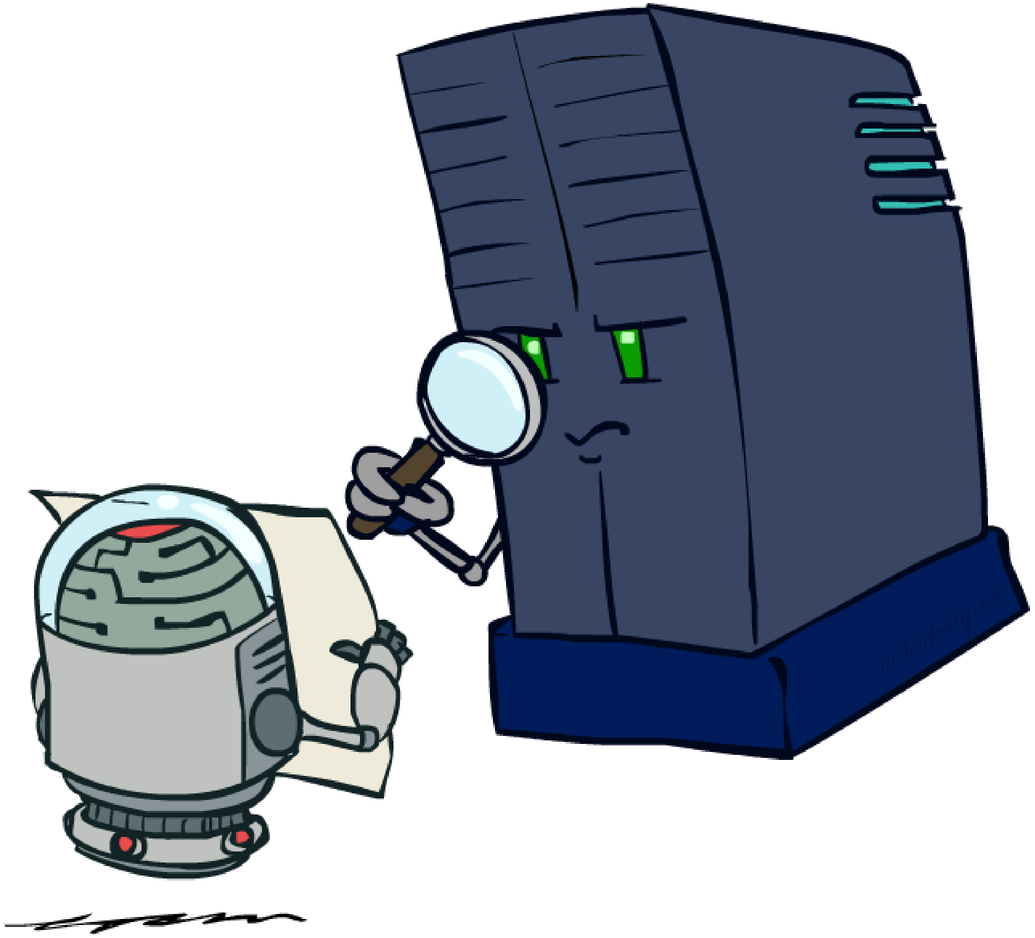
Policy Iteration

- Alternative approach for optimal values:
  - Step 1: Policy Evaluation: calculate utilities for some fixed policy (not optimal utilities!) until convergence
  - Step 2: Policy Improvement: update policy using one-step look-ahead with resulting converged (but not optimal!) utilities as future values
  - Repeat steps until policy converges

- This is Policy Iteration
  - It's still optimal!
  - Can converge (much) faster under some conditions

Co-financed by the Connecting Europe Facility of the European Union

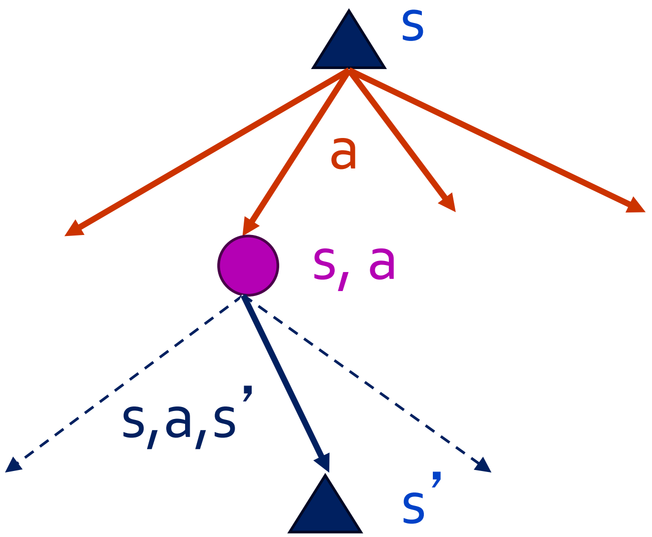
Policy Evaluation



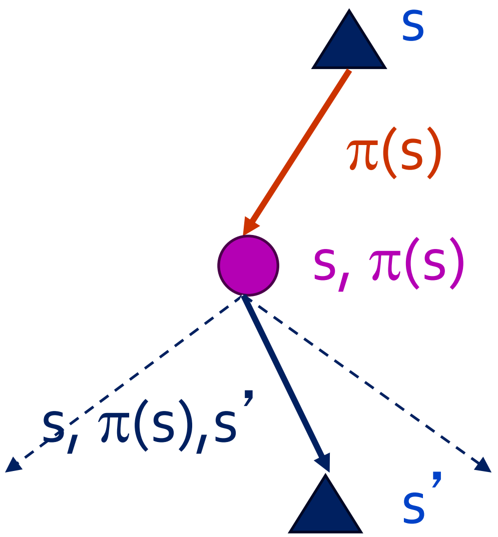
Co-financed by the Connecting Europe Facility of the European Union

Fixed Policies

Do the optimal action



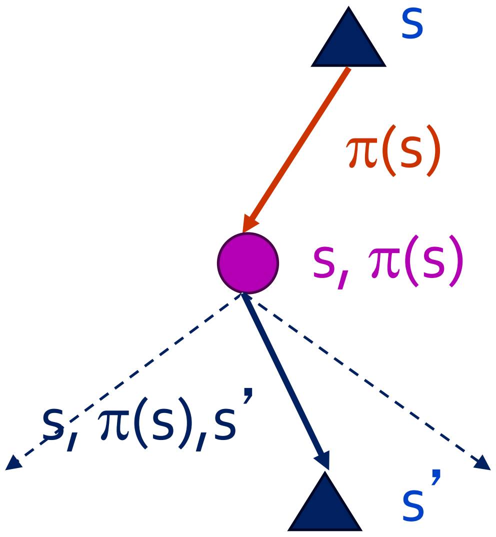
Do what π says to do



- Expectimax trees max over all actions to compute the optimal values
- If we fixed some policy $\pi(s)$, then the tree would be simpler – only one action per state
  - … though the tree's value would depend on which policy we fixed

Co-financed by the Connecting Europe Facility of the European Union

Utilities for a Fixed Policy

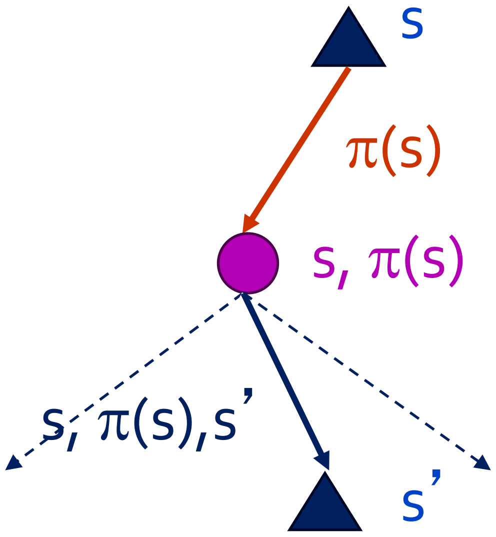


- Define the utility of a state s, under a fixed policy π :
  - $V^{\pi}(s)$ = expected total discounted rewards starting in s and following π
- What is the recursive relation (one-step look-ahead / Bellman equation)?
  - Hint: recall Bellman equation for optimal policy:

$$V^*(s) = \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$$

Co-financed by the Connecting Europe Facility of the European Union

Utilities for a Fixed Policy



- Define the utility of a state s, under a fixed policy π :
  - $V^\pi(s)$ = expected total discounted rewards starting in s and following π
- What is the recursive relation (one-step look-ahead / Bellman equation)?
  - Hint: recall Bellman equation for optimal policy:

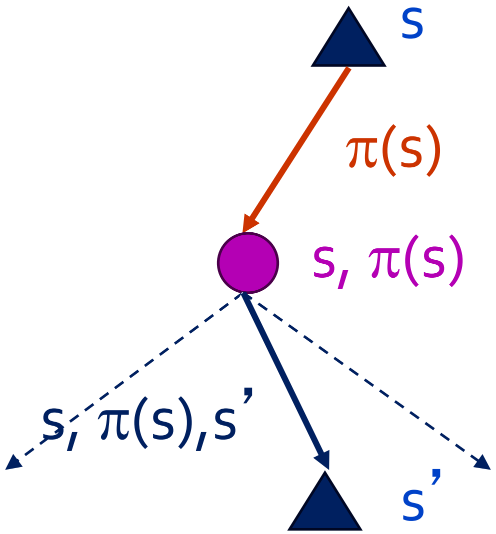
$$V^*(s) = \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$$

Answer:

$$V^\pi(s) = \sum_{s'} T(s,\pi(s),s')[R(s,\pi(s),s') + \gamma V^\pi(s')]$$

Co-financed by the Connecting Europe Facility of the European Union

Policy Evaluation



- How do we calculate the V's for a fixed policy π?
- Idea 1: Turn recursive Bellman equations into updates (like value iteration)

$$V_0^\pi(s) = 0$$

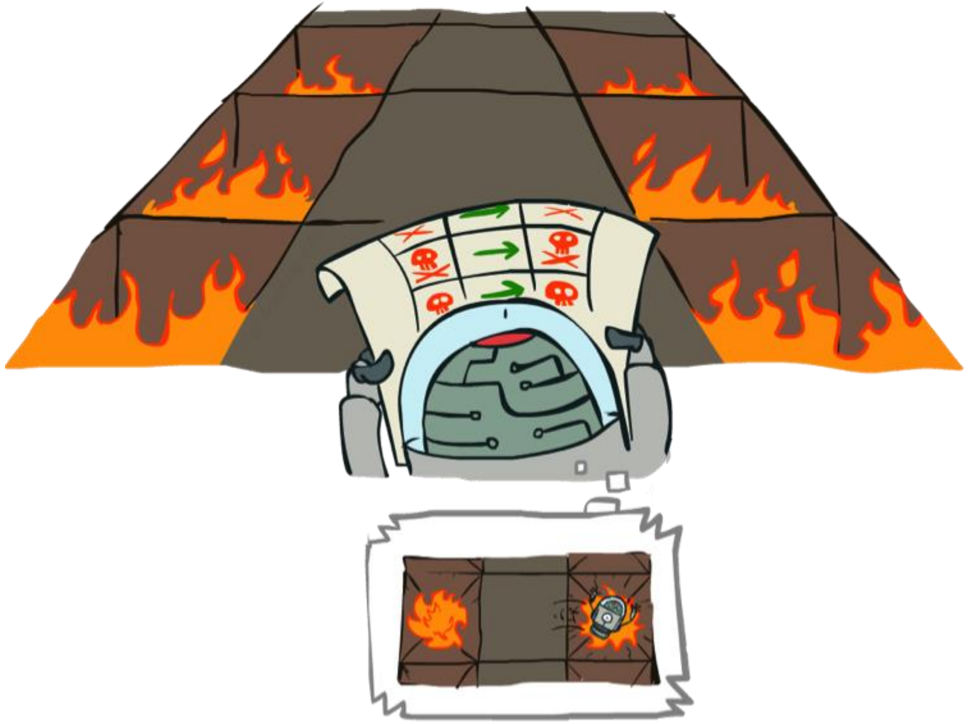
$$V_{k+1}^\pi(s) \leftarrow \sum_{s'} T(s,\pi(s),s')[R(s,\pi(s),s') + \gamma V_k^\pi(s')]$$

- Efficiency: $O(S^2)$ per iteration
- Idea 2: Without the maxes, the Bellman equations are just a linear system
  - Solve with Matlab (or your favorite linear system solver)

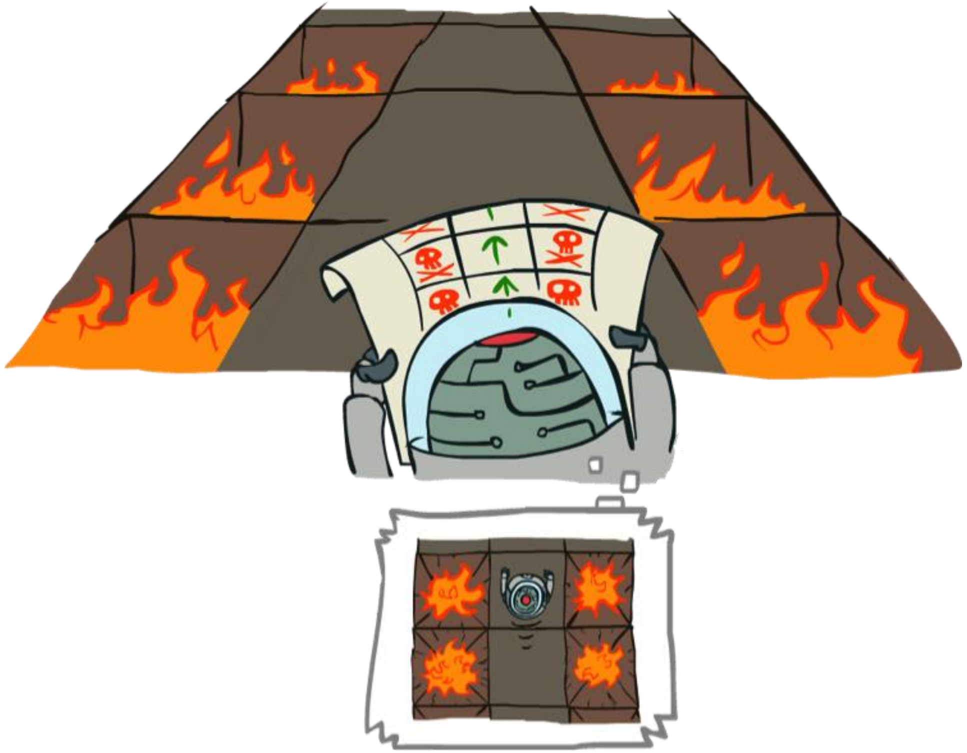
Co-financed by the Connecting Europe Facility of the European Union

Example: Policy Evaluation

Always Go Right



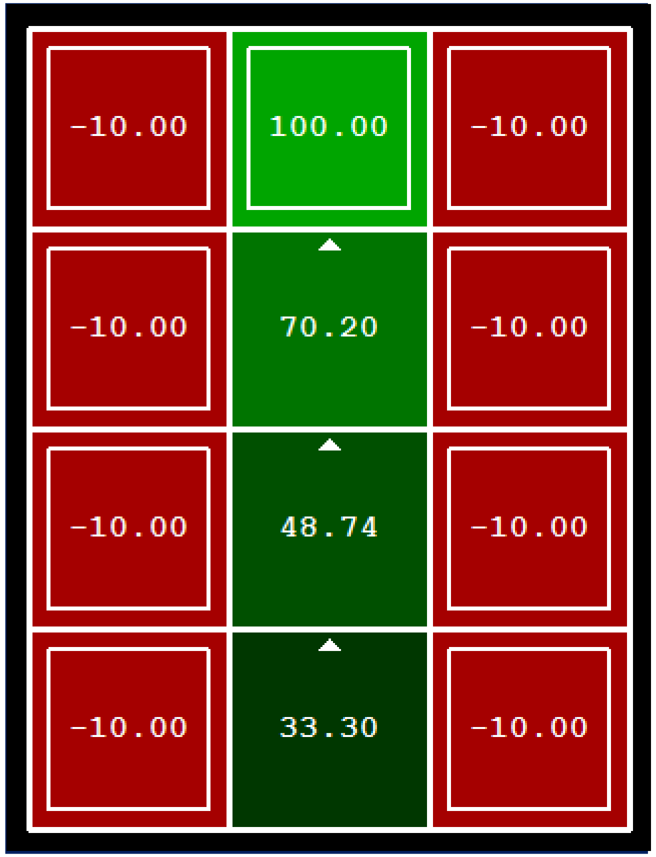
Always Go Forward



Example: Policy Evaluation

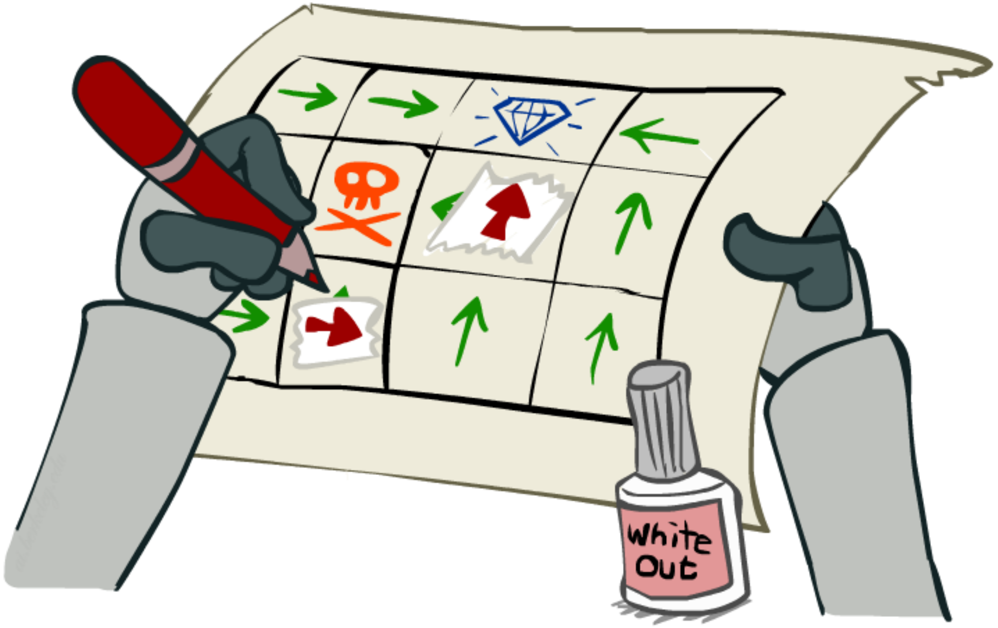
Always Go Right

Always Go Forward



Co-financed by the Connecting Europe Facility of the European Union

Policy Iteration



Co-financed by the Connecting Europe Facility of the European Union

Policy Iteration

- Evaluation: For fixed current policy π, find values with policy evaluation:
  - Iterate until values converge:

$$V_{k+1}^{\pi_i}(s) \leftarrow \sum_{s'} T(s, \pi_i(s), s')[R(s, \pi_i(s), s') + \gamma V_k^{\pi_i}(s')]$$

- Improvement: For fixed values, get a better policy using policy extraction
  - One-step look-ahead:

$$\pi_{i+1}(s) = \arg\max_a \sum_{s'} T(s, a, s')[R(s, a, s') + \gamma V^{\pi_i}(s')]$$

- Repeat steps until policy converges

Co-financed by the Connecting Europe Facility of the European Union

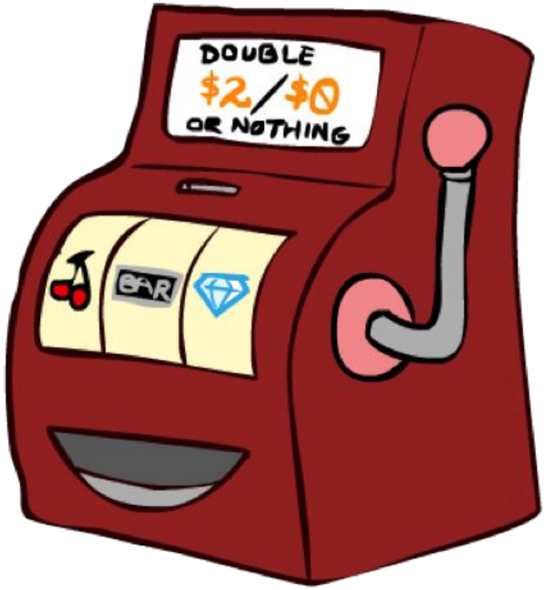
Comparison

- Both value iteration and policy iteration compute the same thing (all optimal values)
- In value iteration:
  - Every iteration updates both the values and (implicitly) the policy
  - We don’t track the policy, but taking the max over actions implicitly recomputes it
- In policy iteration:
  - We do several passes that update utilities with fixed policy (each pass is fast because we consider only one action, not all of them)
  - After the policy is evaluated, a new policy is chosen (slow like a value iteration pass)
  - The new policy will be better (or we’re done)
- Both are dynamic programs for solving MDPs

Co-financed by the Connecting Europe Facility of the European Union

Advanced Topics in AI

Next: Summary and Outlook



Instructor: Prof. Dr. techn. Wolfgang Nejdl

Leibniz University Hannover

Co-financed by the Connecting Europe Facility of the European Union

[These slides were created by Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley. All materials are available at http://ai.berkeley.edu.]