REINFORCEMENT LEARNING
PART 2

Ivan Bratko

Faculty of Computer and Information Science

University of Ljubljana

THREE AGENT DESIGNS FOR RL PROBLEM

- **Utility-based agent**: Learns utility function on states and uses it to choose best action in a given state. Note: Agent also needs a model of action effects on states. That is (approximation to) function delta. Agent has to learn this model!

- **Q-learning agent**: learns action-utility function

  Q(S,A) = maximum utility of performing A in state S

  Note: this doesn't need model of delta to choose most promising action

- **Reflex agent**: learns policy PI: States --> Actions

PASSIVE vs. ACTIVE REINFORCEMENT LEARNING

- **Passive agent** assumes fixed policy PI, A = PI(State)
- Goal is to learn how good PI is
- I.e. learn utility $U^{PI}(s)$ of state s given policy PI

- Agent simply executes a number of trials (sequences of state-action pairs) with policy PI

- **Active agent** searches for best (good) policy among possible policies. Therefore it trades off between **exploitation** and **exploration**.

ESTIMATING STATE UTILITIES

This is fundamental problem in RL

Four problems of increasing difficulty of estimating utilities:

- **1. Given** state transition model delta, a reward function, and a policy

  Find utilities of states produced by this policy
- **2. Given** state transition model delta and a reward function

  Find best policy, and utilities of states produced by best policy
- **3. Given** a policy (and not model of delta)

  Learn utilities of states produced by this policy
- **4. Neithe**r delta nor policy given

  Learn optimal policy (which maximises utilities)

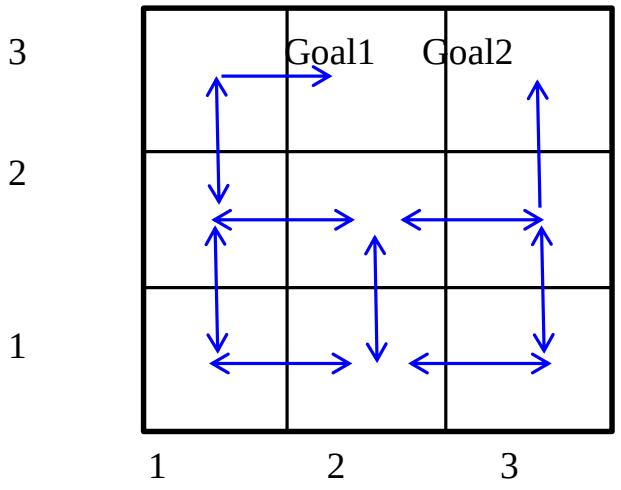
OVERVIEW OF THE FOUR PROBLEMS OF INCREASING DIFFICULTY

	Model delta, rewards	Policy PI	Utilities
(1)	✓	✓	?
(2)	✓	?	?
(3)	?	✓	?
(4)	?	?	?

COMPUTING UTILITIES WHEN REWARDS, MODEL AND POLICY ARE KNOWN

- This can be done directly by using Bellman equations
- For example: an exam problem, 5 September 2019

Exam problem. The 2-dimensional grid 3x3 below specifies a reinforcement learning problem.



The arrows indicate possible actions and corresponding transitions between states. The actions are l, r, d, u (left, right, down, up). As shown in the figure above, not all actions are possible in every state. E.g. in state (2,2), actions r, l and d are possible. The system is deterministic except for the action »up« in state (3,2). Here action »up« causes transition upwards (into goal state (3,3)) with probability p=0.9, or transition to into state (2,2) with probability 0.1. In goal states Goal1 (2,3) and Goal2 (3,3) no transitions are possible. Rewards for all the transitions are equal 0, except for the following:

Transition into any of the goal states is rewarded by 1

Transition from (2,2) to (2,1) is rewarded by 2

The agent is rewarded by the discounted total cumulative reward; the discount factor gamma is 0.5. The agent is initially in the start state (1,2).

Exam problem continued

(a) What is the cumulative reward from the start state (1,2) of action sequence [u,r]?

(b) What is the cumulative reward from the start state (1,2) of infinite action sequence: [r,d,l,u,r,d,l,u,r,d,l,u….]. That is: repeat indefinitely the cycle of four actions [r,d,l,u].

(c) What is approximately the expected utility U_{32} of state (3,2) with the following policy: in state (3,2) do “up”, and in state (2,2) do “right”. (Note that doing “up” in (3,2) may result in transition to (2,2)).

(d) What is the optimal policy that gives maximum utility of state (1,2). Justify your answer, without necessarily calculating this utility.

ANSWERS TO QUESTIONS

(a) What is the cumulative reward from the start state (1,2) of action sequence [u,r]?

ANSW: U((u,r), (1,2)) = 0 + 0.5 * 1 = 0.5

(b) What is the cumulative reward from the start state (1,2) of infinite action sequence: [r,d,l,u,r,d,l,u,r,d,l,u….]. That is: repeat indefinitely the cycle of four actions [r,d,l,u].

ANSW:

U12 = 0 + gamma*U22,

U22 = 2 + gamma*U21, U21 = 0 + gamma*U11, U11 = 0 + gamma*U12

U12 = gamma*(2 + gamma^3 * U12)

U12 = 0.5 * 2 + 0.5^4 * U12

U12 = 1 / (1 – (1/2)^4)

U12 = 16/15 ~ 1.0667

ANSWERS TO QUESTIONS CTD.

(c) What is approximately the expected utility U_{32} of state (3,2) with the following policy: in state (3,2) do “up”, and in state (2,2) do “right”. (Note that doing “up” in (3,2) may result in transition to (2,2)).

ANSW

U32 = 0.9*1 + 0.1*(0 + gamma*U22)

U22 = 0 + gamma*U32

U32 = 0.9 + 0.1 * gamma^2 * U32

U32 = 0.9 / (1 – 0.1 * 0.5 ^ 2) = 0.923

ANSWERS TO QUESTIONS CTD.

(d) What is the optimal policy that gives maximum utility of state (1,2). Justify your answer, without necessarily calculating this utility.

ANSW

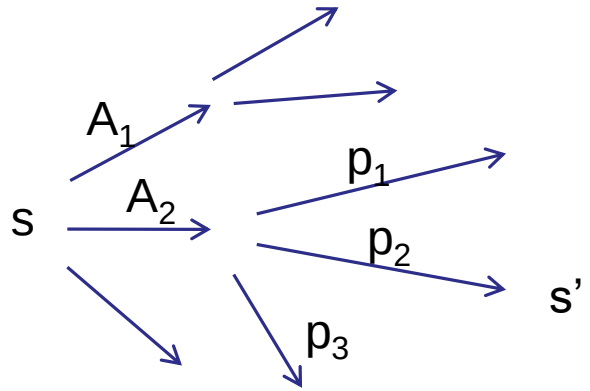
Best U12 is produced by policy (starting from S12): r, d, l, u, r, d, l, u, ...

That is, repeat the cycle: S12, S22, S21, S11, S12, ...

There is an alternative optimal policy also repetitively exploiting reward 2 of transition S22-->S21, starting at S12: r,d,r,u,l,d,r,u,l,...

FINDING BEST POLICY WHEN MODEL DELTA AND REWARDS ARE KNOWN

- Choose among available actions with non-deterministic effects:



- Let PI* be an optimal policy; then:
- $U^{PI*}(s) = MAX_{Ai} [SUM_{s'} P(s'|s,A_i) * (r(s,A_i,s') + gamma * U^{PI*}(s'))]$

SOLVING $U^{Pl*}(s) = MAX_{Ai}$ (.....)

- The equations $U^{Pl*}(s) = MAX_{Ai}$ (.....)

  can be solved by the value iteration algorithm (discussed later)

- Alternatively, CLP(R) can be used directly (Constraint Logic Programming with Real numbers), next slide

CORRESPONDING CONSTRAINT SATISFACTION PROBLEM

- For all states s and all available actions A_i in state s set a constraint

  $U^{PI*}(s) \geq SUM_{s'} P(s'|s,A_i) * [r(s,A_i,s') + gamma * U^{PI*}(s')]$

- This determines lower bounds on utilities U^{PI*} for all states
- The actual attainable utilities cannot exceed max. of these lower bounds. This can be formulated as the optimisation crirerion:

  For all s: minimize($U^{PI*}(s)$)

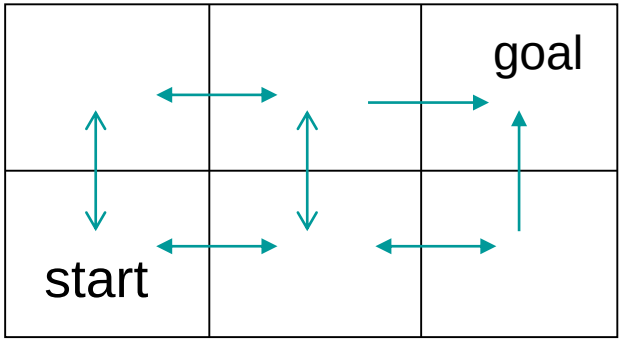
- Explanation of this method by example. E.g. the max. value M among values { 3, 6, 4} can be stated by constraints as:

  $M \geq 3, M \geq 6, M \geq 4$, minimize(M)

  The solution of this constraint problem is M = 6

EXAMPLE

- Consider a simple robot world consisting of a 3x2 grid. Robot can move horizontally or vertically between adjacent cells; the goal is to arrive into the topmost rightmost cell as quickly as possible.



- Reward = 100 for the two possible moves into the top-right cell, and 0 for all other states and actions; gamma is, say, 0.9.

WHAT IS BEST POLICY HERE?

- 3x2 grid with non-deterministic action up in state (3,1): action “up” results in state (3,2) in 80% of cases, and in (3,1) in 20% of cases

- Solve with CLP (see robot on grid 2by3 2.pl)

CLP(R) PROGRAM

```
% Optimal policy for robots on grid 3x2
% All state transitions are deterministic except for state s31 where
% up causes transition up with P=0.8, and no transition with P = 0.2

utilities( U11, U21, U31, U12, U22) :-
  G = 0.9,
  { U11 >= G*U21, U11 >= G*U12,
    U12 >= G*U11, U12 >= G*U22,
    U21 >= G*U11, U21 >= G*U22, U21 >= G*U31,
    U22 >= G*U12, U22 >= G*U21, U22 >= 100,
    U31 >= G*U21,                    % The case of action left
    U31 >= 0.8*100 + 0.2*G*U31 },   % The case of nondeterministic action up
  minimize( U11), minimize(U31).
```

Not for exam!

ESTIMATING STATE UTILITIES FOR FIXED POLICY WHEN DELTA IS NOT KNOWN ("PASSIVE AGENT")

TWO PASSIVE RL METHODS FOR ESTIMATING STATE UTILITIES

1. ADP, Adaptive Dynamic Programming; learns state transition model

2. TD, Temporal Difference learning: keeps adjusting utility estimates of states in direction of “Bellman eq. agreement” between states

Both methods use Bellman equations

ADP (Adaptive Dynamic Programming)

DETERMINE UTILITIES THROUGH LEARNING A MODEL DELTA

- To learn a model means to learn probabilities P(s' | s, PI(s));
  PI(s) is action chosen by policy PI
- These probabilities and rewards can be estimated from experimental trials
- When these probabilities and rewards R are known, then plug these into Bellman equations
- Solve resulting system of linear equations to obtain utilities $U^{PI}(s)$ for all s

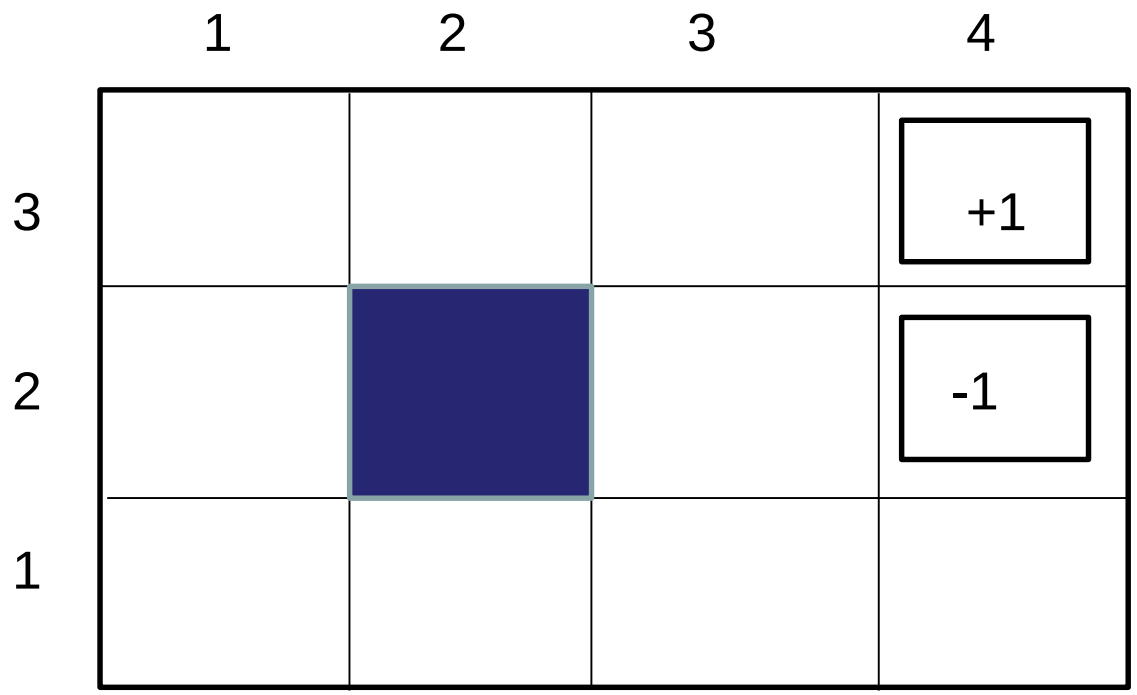
TD LEARNING: EXAMPLE
From Russell&Norvig

- See Fig. 21.1 in Russell & Norvig (2010)

- Note: Even if the policy is fixed, the results of actions are not always the same (system may behave non-deterministically). E.g. executing action Right in state (1,3) may result once in state (2,3) and once in state (1,2)

- The task we consider now is learning state utilities from experimental trials. In each trial, the robot starts in state (1,1), executes the given fixed policy, observes actual state transitions and rewards, and ends in a terminal state.

EXAMPLE ROBOT'S WORLD
From Russell and Norvig



- Robot moves in 4x3 grid between adjacent squares
- Square (2,2) (dark square) is an obstacle, squares (4,2) and (4,3) are **terminal states**

EFFECTS OF ROBOT'S ACTIONS

(1) Robot cannot move into wall or obstacle

(2) Robot can move into an adjacent square
(either horizontally or vertically): right, left, up, down

(3) Robot's actions have non-deterministic results, e.g.:
Robot's action is "up", but actual effects can be:
up, with probability 0.8;
left with prob. 0.1 (if cell to the left is clear, else no move);
right with prob. 0.1 (if cell to the right is clear, else no move)

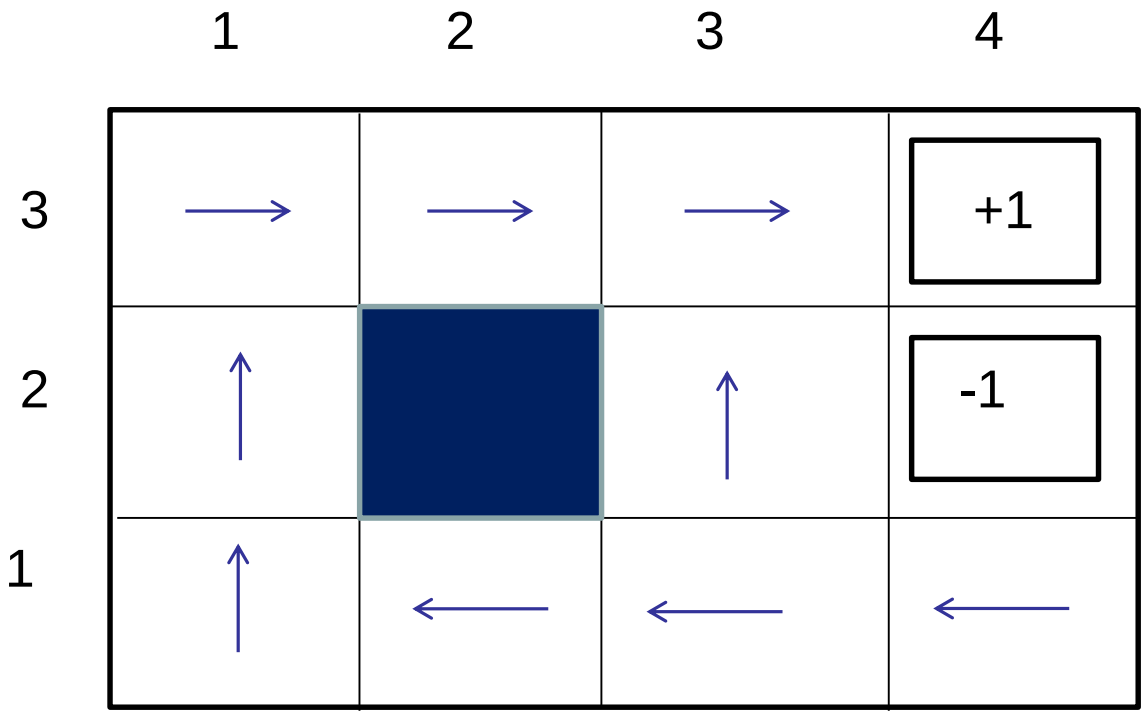
REWARDS

- Rewards are received by the robot when the robot enters a state

- For terminal state (4,3): r((4,3)) = 1
- For terminal state (4,2): r((4,2)) = -1
- For all non-terminal states s: r(s) = -0.04
- Gamma = 1 (no discount)

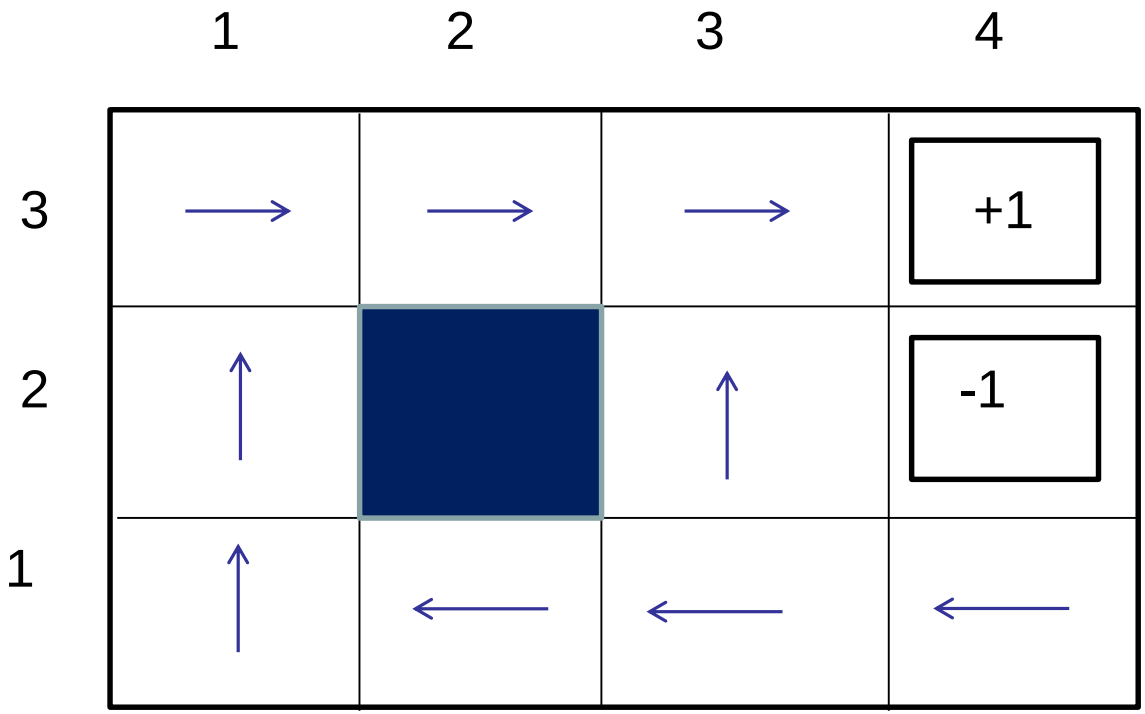
- All this means roughly: Robot should try to reach (4,3) as quickly as possible, and try to avoid (4,2)

OPTIMAL POLICY FOR THIS PROBLEM (shown by arrows)



- Policy PI*; This is optimal for cumulative reward with no discounting
- Question: In state (3,1), why is best robot's action "left"?

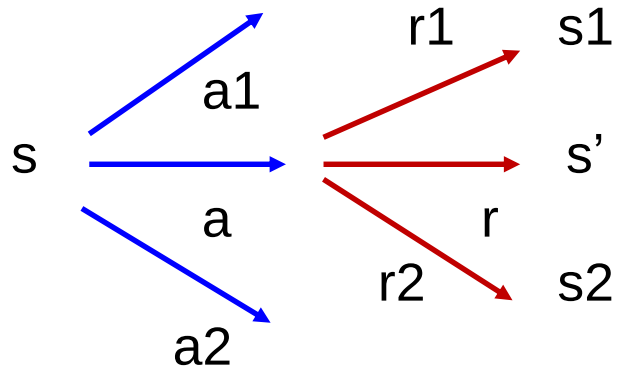
POLICY GIVEN, DELTA NOT KNOWN, WHAT ARE STATE UTILITIES?



- Policy PI*; This is optimal for cumulative reward with no discounting

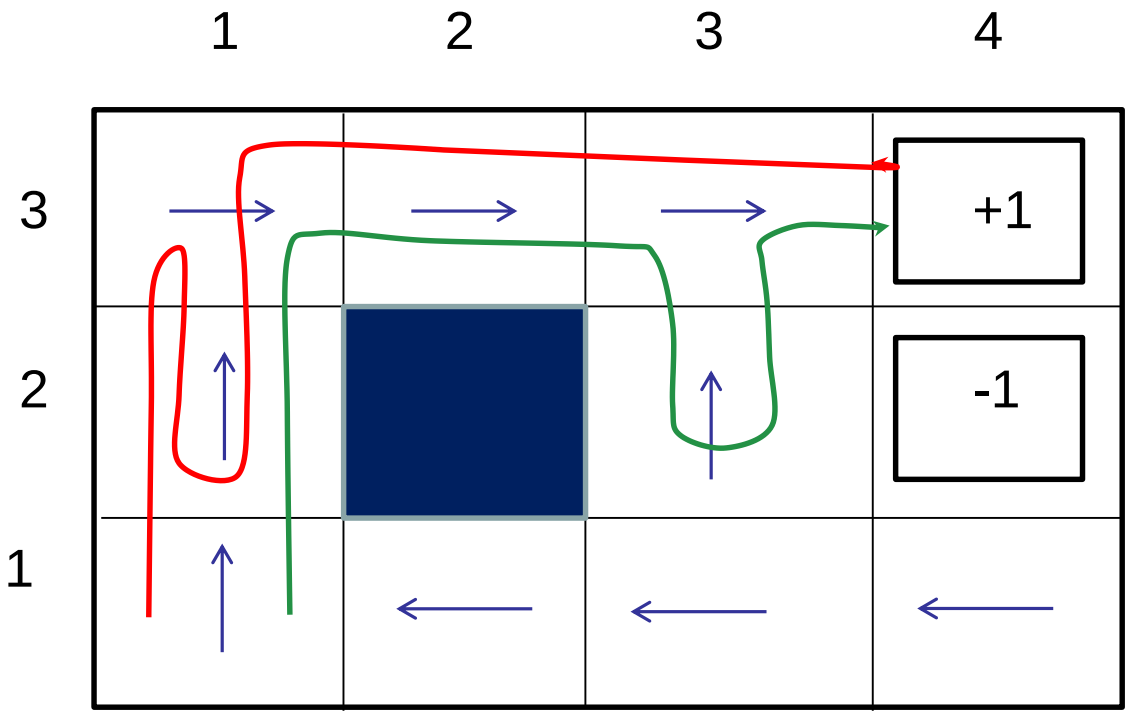
TD-LEARNING (TEMPORAL DIFFERENCE)

- In consecutive trials, fixed policy may produce different state transitions due to non-determinism



- Fixed policy chooses action a;
- Result of a can be s', s1, or s2
- This gives rise to different immediate rewards r1, r, or r2, and different further cumulative rewards
- Different trials produce different cumulative rewards of state s

SOME EXAMPLE TRIALS



SOME EXAMPLE TRIALS

Note: non-deterministic effects of actions

Trial 1:

State utilities: 0.76 0.80 0.84 0.88 0.92 0.96 1.00 0

(1,1) --> (1,2) --> (1,3) --> (1,2) --> (1,3) --> (2,3) --> (3,3) --> (4,3)

Rewards: -0.04 -0.04 ... -0.04 +1

Cumulative reward of state (1,1), no discount, from trial 1 is 0.76

Trial 2: (1,1) --> (1,2) --> (1,3) --> (2,3) --> (3,3) --> (3,2) --> (3,3) --> (4,3)

Cumulative reward of state (1,1) from trial 2 is again 0.76

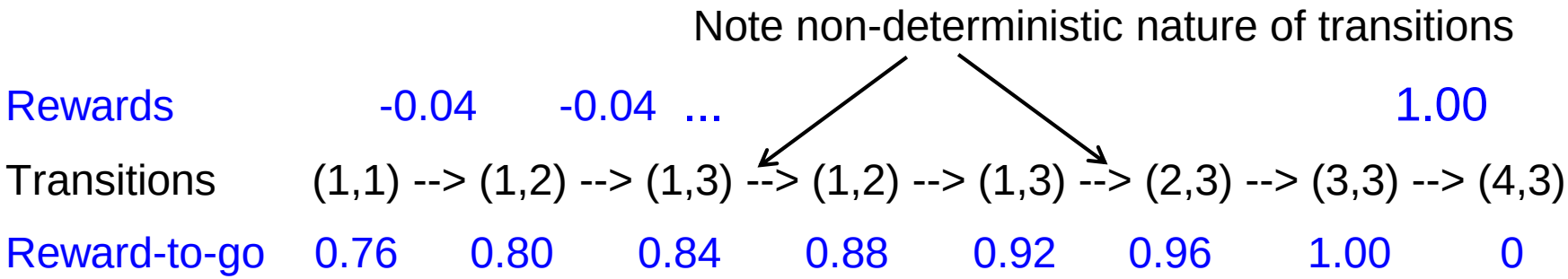
Trial 3: (1,1) --> (1,2) --> (1,3) --> (2,3) --> (3,3) --> (4,3)

Cumulative reward of state (1,1) from trial 3 is 0.84

TD-LEARNING

- Exploits Bellman eq. constraints
- Example: Two trials in 4x3 world of Russell & Norvig

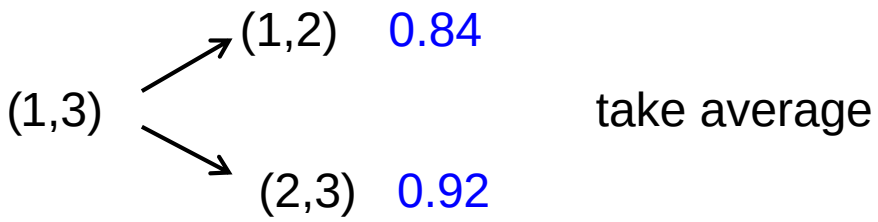
Trial 1



Some utility estimates from Trial 1:

$U^{PI}(2,3) = 0.96$

$U^{PI}(1,3) = (0.84 + 0.92)/2 = 0.88$



Trial 2

Rewards -0.04 -0.04 -0.04

Transitions (1,1) --> (1,2) --> (1,3) --> (2,3) --> …

Current utility estimates: 0.88 0.96

Consider (1,3) --> (2,3) in Trial 2; Note at this point we don't know yet rewards-to-go of (1,3) and (2,3) for Trial 2

Estimates from Trial 1: $U^{PI}(1,3) = 0.88$, $U^{PI}(2,3) = 0.96$

If transition (1,3) --> (2,3) occurred always then according to Bellman:

$U^{PI}(1,3) = -0.04 + U^{PI}(2,3)$

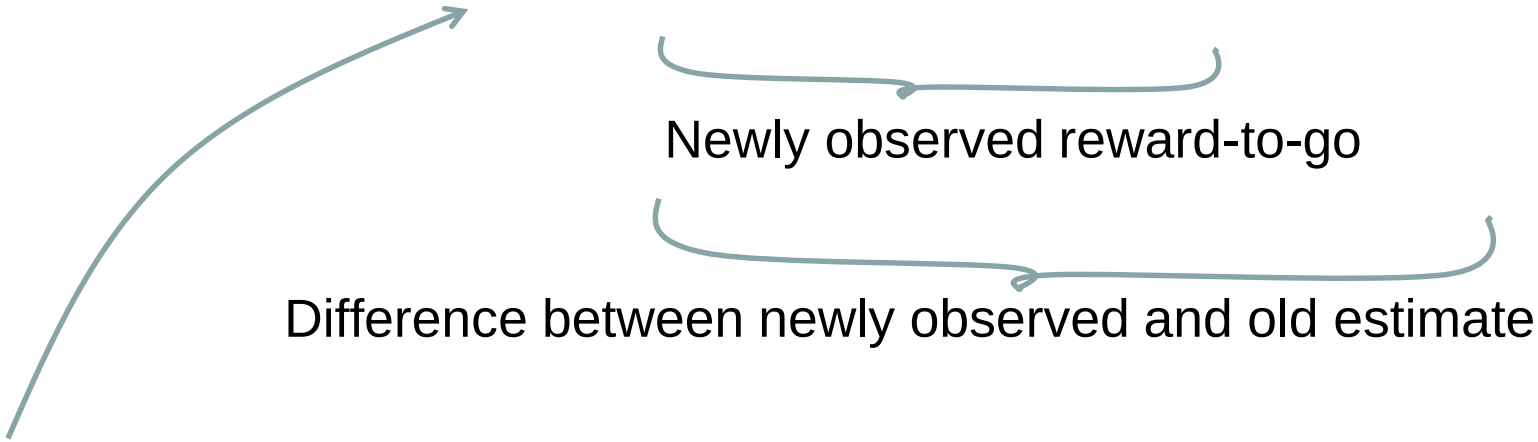
$U^{PI}(1,3) = 0.92$ would be consistent with this according to trial 2,

therefore current estimate $U^{PI}(1,3) = 0.88$ looks too low from the viewpoint of observed transition (1,3) --> (2,3) in Trial 2 and should be increased to "agree" with this transition and Bellman

Such estimate corrections are implemented through **TD-update rule**, which is applied after **each newly observed transition**

TD-update rule, for observed transition (s, s')

- $U^{PI}(s)$ <-- $U^{PI}(s)$ + alpha * (r(s,s') + gamma * $U^{PI}(s')$ - $U^{PI}(s)$)



Learning rate parameter: higher alpha, more vigorous adjustment

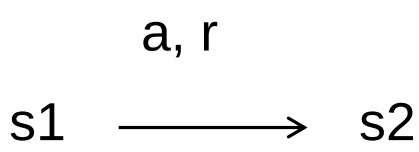
- This rule uses difference in utilities between successive states s and s'
- Therefore it is called Temporal Difference update rule (TD rule)
- In our example, $U^{PI}(1,3)$ update after trans. (1,3) → (2,3) in Trial 2:

$U^{PI}(1,3)$ <-- 0.88 + alpha * (-0.04 + 1*0.96 - 0.88) = 0.88 + alpha*0.04

$0 \leq$ alpha ≤ 1 ("discounted adjustment", some way towards new diff.)

USE TD-UPDATE AFTER EACH OBSERVED TRANSITION

- Observed transition after action a:



- Current U-estimates: U(s1), U(s2)
- After this transition, observed utility U'(s1) = r + gamma*U(s2)
- Difference between current estimate and new observation:

  D = U'(s1) - U(s1) = r + gamma*U(s2) - U(s1)

- TD-update:

  U(s1) ← U(s1) + alpha * D, $0 < \text{alpha} < 1$

TD-UPDATE RULE, INTUITITIVE EXPLANATION: ESTIMATING AVERAGE OVER SEQUENCES

- Illustration of the basic idea of TD-learning
- Consider: Agent is receiving a sequence of numbers v_1, v_2, ..., coming one by one
- After having received k numbers, agent wants to predict next number
- Agents predicts next number as the average A_k of v_1, ..., v_k
- $A_k = (v_1 + ... + v_k) / k$
- $k * A_k = v_1 + ... + v_{k-1} + v_k = (k – 1) * A_{k-1} + v_k$
- Agent therefore keeps updating current average estimate by using recursive relation:

  $A_k = (1- 1/k) * A_{k-1} + v_k/k$
- Let $\alpha_k = 1/k$, then $A_k = A_{k-1} + \alpha_k * (v_k – A_{k-1})$ (TD-update rule)

CONVERGENCE

- Parameter alpha may vary with # times state s has been visited: alpha should decrease with # visits

- **Convergence: If alpha so decreases then $U^{PI}(s)$ will converge to its correct value**

COMPARISON ADP vs TD

- ADP converges faster with # trials than TD towards correct U^{PI}
- But: TD requires less computation per observation
- TD does not need a state transition model to perform its updates. Instead, environment provides corrections between neighbor states in the form of observed transitions

SIMILARITIES BETWEEN TD AND ADP

- They both make local adjustments to state utilities to make them "agree" with their successor states
- TD adjusts a state to better agree with its (*single*) observed successor
- ADP adjusts a state to agree with *all* of its successors (weighted by their probabilities)
- This difference disappears when TD has processed large # observed transitions

COMPARISON CTD.

- For each new observation:

  (1) TD makes local adjustment, whereas

  (2) ADP makes many adjustments (for possibly all states) between U and current model P. Although the observed transition mainly affects probability of this transition, it may (to a lesser extent) also affect other transitions

- Thus: TD can be viewed as a crude, but efficient first approximation to ADP

ACTIVE LEARNING:

TRY TO FIND A GOOD POLICY

Ideally: Aim at optimal policy

See Reinforcement learning Part 3