REINFORCEMENT LEARNING
PART 3

Ivan Bratko
Faculty of Computer and Information Science
University of Ljubljana

ACTIVE LEARNING:

TRY TO FIND A GOOD POLICY

Ideally: Aim at optimal policy

ACTIVE LEARNING

- Active learner attempts to find an optimal policy
- How can ADP be turned into active agent?
  - (1) Agent has to optimise policy, not just execute fixed policy
  - (2) If agent has learned state-transition function delta then optimal policy can be determined by solving constraint problem defined by Bellman equations:

$$U(s) = R(s) + gamma*max_a SUM_{s'}[P(s' \mid s,a)*U(s')]$$

ACTIVE AGENT: EXPLOITATION VS. EXPLORATION

- Active agent’s task is to search among possible policies

- Actions tried by agent serve two goals:

  (1) Achieve high reward (**exploitation**)

  (2) Learning transition model (**exploration**)

MATHEMATICAL PROBLEM FOR STUDYING OPTIMAL EXPLORATION POLICIES

- What is optimal exploration policy?
- A theoretical framework for studying optimal exploration policies: **multi-armed bandit problem**
- n-armed bandit is a gambling machine with n levers (arms - slot machines); gambler inserts a coin, chooses and pulls one of the levers; corresponding machine decides probabilistically about the reward to be paid to the gambler
- Gambler plays many times: what is the best exploration policy (that maximises expected cumulative reward)? Answer in general case is very hard, specially when the machines are not independent
- This is a RL problem

BALANCE BETWEEN EXPLOITATION AND EXPLORATION

- Is there optimal balance between exploitation and exploration? Main idea is roughly: GLIE

- GLIE schemes try to achieve good balance
- GLIE = **G**reedy in the **Li**mit of **E**xploration

  Initially tend to explore, later tend to exploit.

  In the end, method becomes just greedy (choose action that maximizes reward)
- A simple GLIE scheme:

  With probability 1/t, choose random action, otherwise choose highest utility action (t=time; this probability decreases with time).

This does converge, but can be very slow.

TWO MAIN APPROACHES TO ACTIVE RL

Both approaches explore among policies aiming at a good policy, but they differ in what is being learned

1. Utility based learning: Learn optimal state utilities and model delta

2. Q-learning: Learn Q-values

UTILITY BASED ACTIVE RL

- To search for a good policy, agent carries out trials in the domain
- In choosing next action, the agent strives to good compromise between exploration and exploitation
- The agent updates a domain model, i.e. transition probabilities by counting transitions to next state of form (s, a, s')
- During experimentation, the agent also updates approximations to optimal state utilities and optimal policy
- This can be done by solving the ADP problem; a popular method for that is **value iteration algorithm**

VALUE ITERATION

- Value iteration is a simple and practical method for determining optimal state utilities, so it is part of common technology for RL

- Given a (current) model delta (i.e. transition probabilities), problem is to solve equations:

  $U(s) = \max_a (P(s' \mid s,a) * [\ r(s,a,s') + gamma * U(s')]$ (Eq. B)

- Value iteration method roughly consists of:

1. Initialise U(s) for all states s with arbitrary initial values, e.g. U(s)=0
2. Keep updating simultaneously all U(s) according to Eq. B using current U(s) in right hand side of Eq. B, until difference between old U and new U values become sufficiently small

VALUE ITERATION ALGORITHM

U'(s) are current estimates, U(s) are estimates from previous iteration

Initialise for all s: U'(s) = 0;

Repeat

for all s: U(s) $\leftarrow$ U'(s), D $\leftarrow$ 0

for all s do simultaneously:

U'(s) $\leftarrow$ $\max_a$ (P(s' | s,a) * [r(s,a,s') + gamma * U(s')]

if | U'(s) – U(s) | > D then D $\leftarrow$ | U'(s) – U(s) |

until D < eps * (1 – gamma) / gamma

eps is maximum error allowed in utility of any state

The termination condition comes from mathematical result:

if $|| U_{i+1} - U_i || <$ eps*(1-gamma)/gamma then $|| U_{i+1} - U || <$ eps

That is: If successive estimates are close then U_{i+1} is close to true U

Notation $|| U || = \max_s |U(s)|$ (max abs. value in a vector)

INTUITION WHY SMALL GAMMA HELPS CONVERGENCE

- Consider effects of utilities between distant states:

S0 → S1 → ..... → Sn →

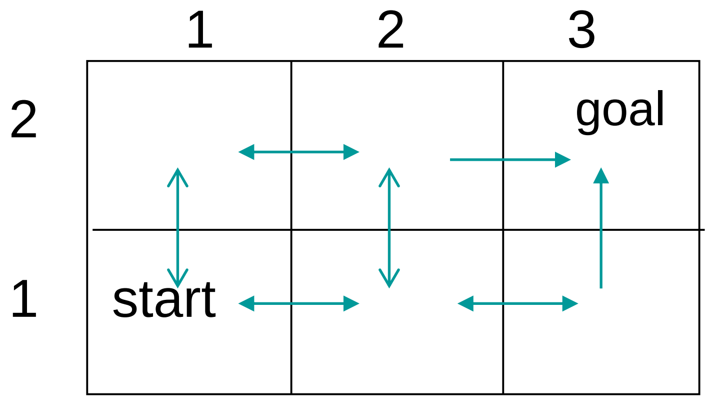
Suppose action in Sn generates large reward, and actions before Sn give low rewards. How does this effect U(S0) if Gamma is low/high?

CONVERGENCE OF VALUE ITERATION

- Value iteration algorithm **always converges** to (unique) solution of Bellman equations B.
- In each iteration, error is reduced at least by factor gamma (exponential convergence). However, this gets slow if gamma is close to 1.
- **Convergence to optimal policy is typically faster than convergence to approximation to optimal U values**
- Why is convergence to optimal policy usually faster? Note that current values U(s) define a policy. When in some iteration U(s) are sufficiently close to U*(s), these U(s) define optimal policy PI*. Often U(s) already define optimal policy even if they are quite far from true values

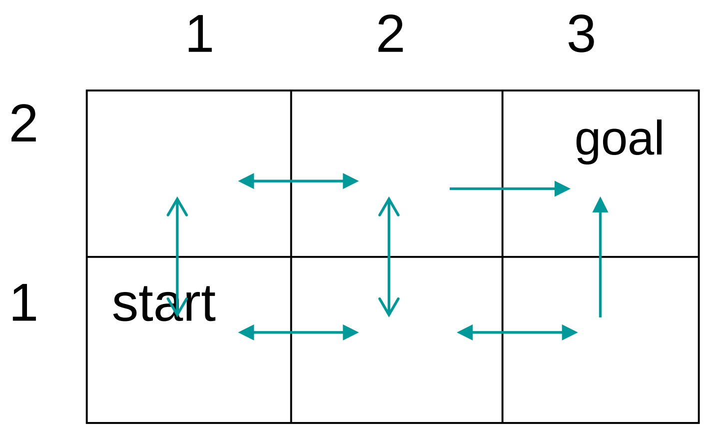
VALUE ITERATION, 2x3 GRID EXAMPLE

- A simple robot world consisting of a 2x3 grid. Robot can move horizontally or vertically between adjacent cells; transition into goal state gives reward 100, other transitions reward 0. Gamma = 0.9.



2x3 GRID EXAMPLE WITH BROKEN ROBOT

- Let transitions 22-right and 11-right be nondeterministic: 50% chance that robot stays where it is. Gamma = 0.9.



	1	2	3
2	0, 45, 62.25, ...	0, 50, 72.5, 82.625 87.18, 89.23, 90.15 90.57, 90.84, 90.88	0
1	0, 0, 0, 81, 81, ...	0, 0, 90, 90	0, 100, 100, ...

USING OPTIMISTIC UTILITY ESTIMATES

- This is an idea to promote exploration in utility based RL
- Assign higher utilities to states reached by under-explored actions
- Such optimistic utility estimates can be used in value iteration algorithm
- Such optimistic utility estimates promote exploration
- Optimistic estimates can be viewed as reducing desire to explore (curiosity) to (a kind of) greed; high utility estimates reflect underexplored states (novelty of states). A simple greedy policy w.r.t. optimistic estimates will drive the agent to underexplored areas

OPTIMISTIC ESTIMATES

- $U^+(s)$ = optimistic utility estimate of s (higher than realistic)
- $U^+(s) \leftarrow \max_a (R(s,a) + \text{gamma } f(\text{SUM}_{s'} P(s' \mid s,a) * U^+(s'), N(s,a))$
- N(s,a) = # times the pair (s,a) has occurred so far

  i.e. # times a has been tried in s
- f(u, n) is called **exploration function**; it trades between greed and curiosity
- To stimulate exploitation, f(u,n) should increase with u
- To stimulate exploration, f(u,n) should decrease with n

AN EXPLORATION FUNCTION

$$f(u,n) = \begin{cases} R^{+} & \text{if } n < N_e \\ u & \text{otherwise} \end{cases}$$

R^{+} is max. possible cumulative reward obtainable in any state, in practice unrealistically high

N_e is fixed parameter, a threshold for optimism

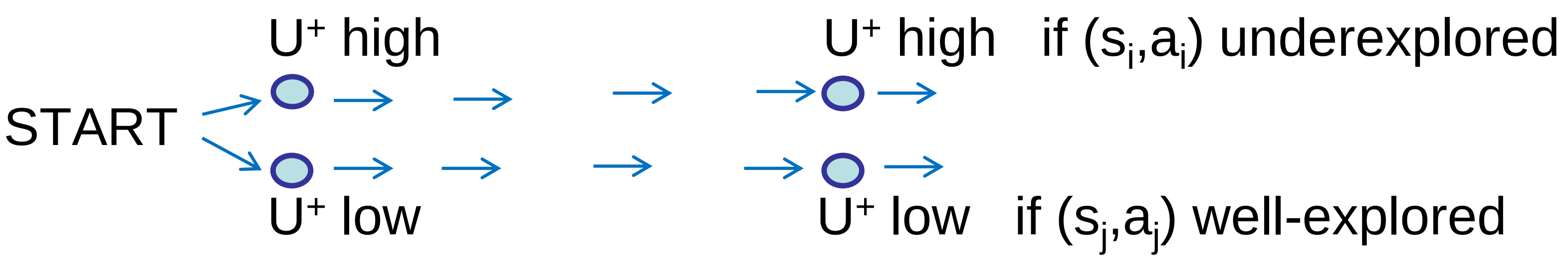
This f makes the agent try each action at least N_e times

DEFINITION OF U^+

- Note important detail in definition of U^+

- $U^+(s)$ <-- $\max_a$ (R(s,a) + gamma * f(SUM P(...) $U^+(s')$, ...)

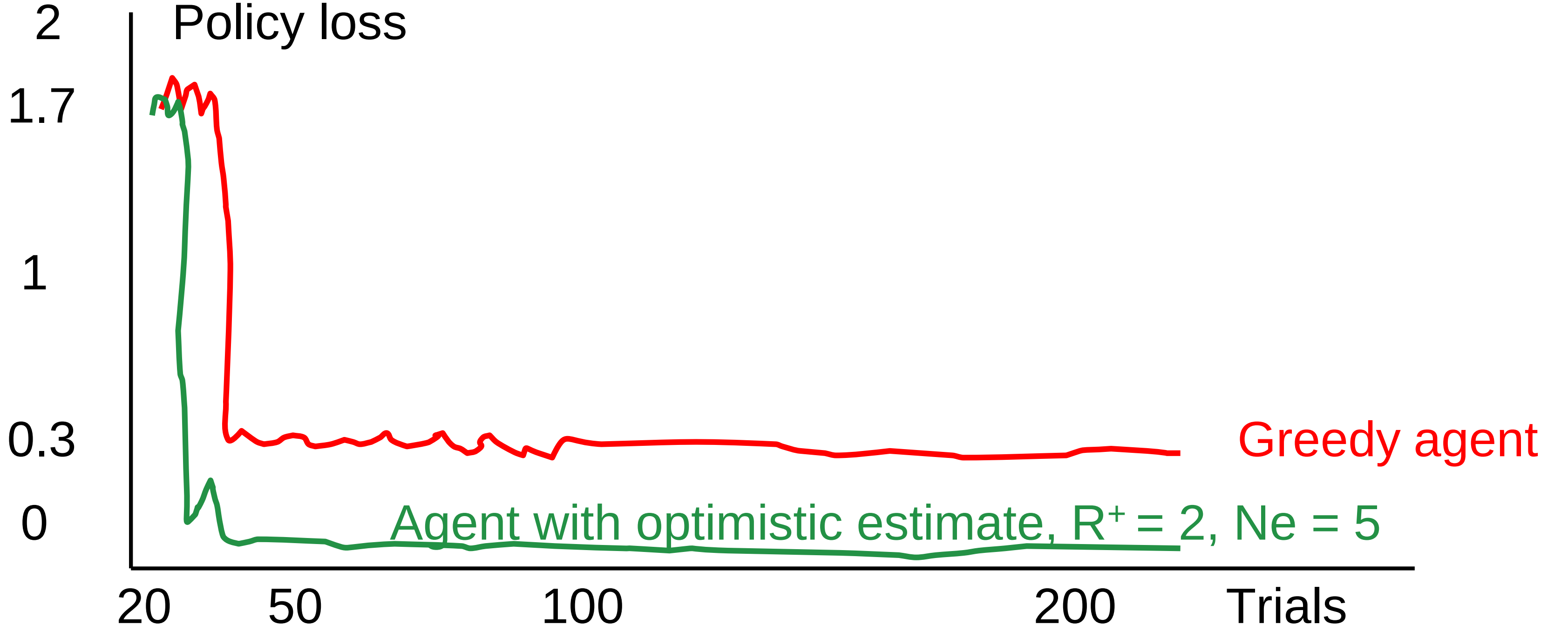
- Very important to have U^+ here and not just U
- This drives agent even in states close to START towards remote states in under-explored regions



Close to START, well explored Far from START, likely under-explored

RUSSELL & NORVIG 4x3 EXAMPLE

- Convergence of learned policy



Note: Re-drawn approximately from Russell&Norvig

Q-LEARNING

Q FUNCTION

- In Q-learning, agent learns function Q. Q(s,a) is defined as the maximal cumulative reward achievable by action a in state s:

  $$Q(s,a) = r(s,a) + \text{gamma } U^{*}(\text{ delta}(s,a)) \quad \text{(for deterministic case)}$$

- In utility-based RL, agent has to learn reward function r and state transition function delta. This suffices to determine U* function, and that determines the optimal policy: (for deterministic case):

  $$PI(s) = \text{argmax}_a \, [\, r(s,a) + \text{gamma } U^{*}(\text{ delta}(s,a))\,]$$

LEARNING Q FUNCTION

- A TD agent that learns Q function does not need a model of delta
- Therefore: Q-learning is said to be a model-free method
- Complexity comparison of value-based RL and Q-learning: in both cases the domain of functions to be learned are of order |S| x |A|

LEARNING Q FUNCTION

- Using identity $U^*(s) = \max_{a'} Q(s,a')$ gives recursive definition of Q:

$$Q(s,a) = r(s,a) + \text{gamma} \max_{a'} Q(s',a')$$

- One way to learn Q function is through iterative approximations

- $Q^\wedge(s,a)$ is current approximation of $Q(s,a)$

- After each action a in state s, approximation is updated by the TD rule:

$$Q^\wedge(s,a) \leftarrow Q^\wedge(s,a) + \text{alpha}*[\, r(s,a) + \text{gamma} \max_{a'} Q^\wedge(s',a') - Q^\wedge(s,a)\,]$$

where s' is result of executing a in s

alpha = update rate: higher alpha, more vigorous update; alpha < 1

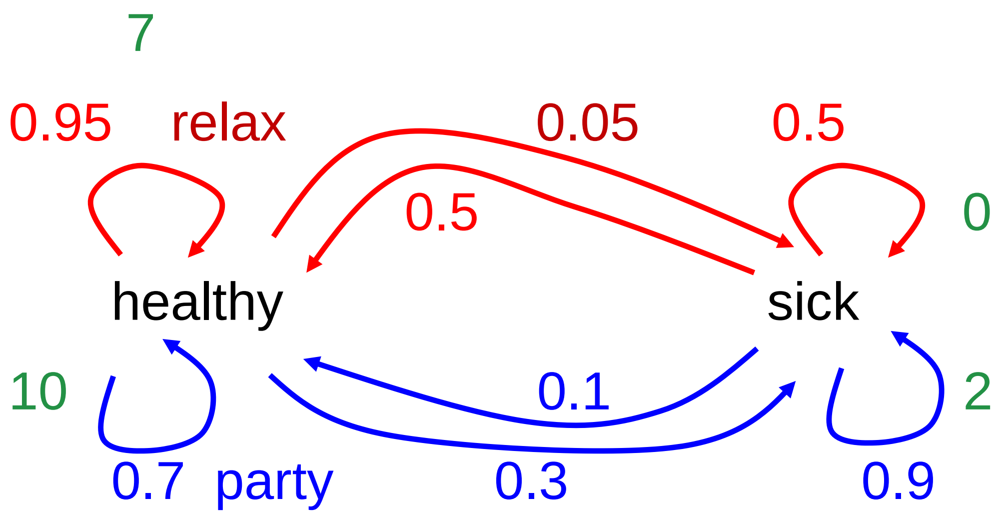
Q LEARNING ALGORITHM

- For each s and a, initialise Q^(s,a) <-- 0
- Let s be currently observed state
- Do “forever”:
  - Select an action a and execute it
  - Receive immediate reward r
  - Observe next state s’
  - Update table entry Q^(s,a):

    Q^(s,a) <-- ... (TD update rule for Q^(s,a))
  - s <-- s’

EXAMPLE FROM POOLE & MACKWORTH 2017

- Sam makes informed decision between partying and relaxing



Green numbers are rewards from actions performed in states

Gamma = 0.8

WHAT ARE Q-VALUES AFTER THE TRIAL BELOW?

s a r s'	(state,action)	Q(state,action)
he re 7 he	he, re	Q^(he,re) = 0.7*0 + 0.3*(7+1) = 2.1
he re 7 he	he, re	Q^(he,re) = 0.7*2.1 + 0.3*(7+0.8*2.1) = 4.07
he pa 10 he	he, pa	Q^(he,pa) = 3.98
he pa 10 si	he, pa	Q^(he,pa) = 5.79
si pa 2 si	si, pa	...
si re 0 si	si, re	
si, re, 0, he	si, re	

gamma = 0.8, alpha = 0.3

BEST POLICY FOR SAM

```
?- utilities(U), q(S,A,Q,U).

U = [healthy/35.714, sick/23.80952],

S = healthy, A = relax, Q = 35.095 ;
S = healthy, A = party, Q = 35.714 ;
S = sick, A = relax, Q = 23.80952 ;
S = sick, A = party, Q = 22.000 ;
```

SAM'S Q-LEARNING

After 100 steps (in this domain a trial never ends):

healthy: [relax:34.0675,party:35.4022]

sick: [relax:26.2121,party:22.02939]

After 1000 steps:

healthy: [relax:36.04827,party:35.31005]

sick: [relax:26.5521,party:21.0226]

After 5.000 steps:

healthy: [relax:34.9970,party:36.1772]

sick: [relax:23.5344,party:21.4785]

After 10.000 steps:

healthy: [relax:35.1979,party:36.0068]

sick: [relax:23.7620,party:21.7365]

RUSSELL & NORVIG ROBOT 4x3

Optimal policy:

In s11: Best action is up, Q(s11,up) = 0.7453

In s31: Best action is left: Q(s31,left) = 0.6514

Q(s31,up) = 0.6325

Q-learning, a trial ends in terminal state, or after 100 steps

#trials	Q(s11,up)	Best policy
50	0.7989	no
100	0.7788	yes
500	0.7582	yes
1000	0.7716	yes
10000	0.7428	yes

EXPLORATION STRATEGIES

- Which action to select next?
- Trade-off between exploitation and exploration
- Extreme exploitation: maximise$_a$ $Q^\wedge(s,a)$ (greedy policy)
- This may never discover policies that are even more profitable than the policy already known (also, cf. convergence theorem that requires visiting all (s,a) pairs many times)

- Epsilon-greedy policy: Choose a random action with probability ε, and action with maximum utility (according to current estimates) with probability 1- ε. Parameter ε may appropriately decrease with time.

MIXTURE OF EXPLOITATION/EXPLORATION, SOFTMAX

- Make probabilistic choices, higher $Q^\wedge$ for action a_i, higher the chance for a_i to be selected ("softmax")
- Let $P(a_i \mid s)$ be prob. of randomly selecting action a_i in state s. Then e.g.

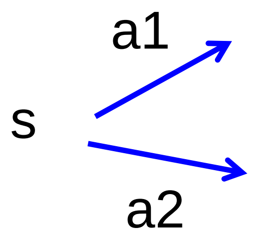
$$P(a_i \mid s) = k^{Q^\wedge(s,ai)} / \text{SUMj } k^{Q^\wedge(s,aj)}$$

  k > 0

  higher k ---> tend to exploit

  lower k ----> tend to explore (k ≥ 1)

  E.g. Q(s,a1)=4, Q(s,a2)=1

  k=1: P(a1)= 1/(1+1) = 0.5, P(a2) = 0.5

  k=2: P(a1)= $2^4/(2^4 + 2^1)$ = 16/18, P(a2)=2/18

- It is appropriate that k varies with time: start with desire to explore (small k), later prefer to exploit (increase k)

CONVERGENCE OF Q LEARNING

- Convergence theorem for Q learning for deterministic MDPs
  - Let all rewards be bounded: for all (s,a), $|(r(s,a)| \leq c$
  - Let agent initialise Q^ table to arbitrary finite values
  - Let agent use Q learning update rule and discount factor gamma s.t. $0 < \text{gamma} \leq 1$
  - Let for all a and s, $Q\hat{}_n(s,a)$ denote values of Q^ after n-th update
  - Then, if each pair (s,a) is visited infinitely many times, then $Q\hat{}_n(s,a)$ converges to Q(s,a) as n --> infinity, for all (s,a).

GENERALISATION OF Q ESTIMATES

- In our methods so far, Q estimates are updated for the visited state-action pairs; these are kept in tabular form, and unvisited values stay unchanged
- This is very limiting
- Therefore other methods determine Q estimates for unvisited state-action pairs through function approximation using ML methods. The estimates for the visited pairs are taken as examples for such ML methods
- Traditionally, neural networks are usually used in this context for function approximation (DRL stands for Deep Reinf. Learning)

CONVERGENCE OF Q LEARNING IN NONDETERMINISTIC CASE

- Convergence theorem for non-deterministic Q learning
  - Let all rewards be bounded: for all (s,a), $|r(s,a)| \leq c$
  - Let the agent initialise Q^ values to arbitrary values, and use the **training rule for nondeterministic MDPs** with discount factor gamma s.t. $0 \leq$ gamma < 1
  - Let n(i,s,a) be the iteration in which action a was for the i-th time applied to state s.
  - If each state-action pair is visited infinitely often, $0 \leq \text{alpha}_n < 1$, and $\text{SUM}_{[i=1..inf]}\ \text{alpha}_{n(i,s,a)} = \text{inf}$ and $\text{SUM}_{[i=1..inf]}\ \text{alpha}^2_{n(i,s,a)} < \text{inf}$, then all $Q^\wedge$ values converge to Q as n --> inf, with probability 1.