Reinforcement Learning
an introduction

Ann Nowé
Ann.nowe@vub.ac.be
http://como.vub.ac.be

By Sutton and Barto
The elements of reinforcement learning (RL)

- Supervised vs unsupervised
- States, actions, policies
- Goals and reward scheme
- Returns
- Episodic & Continuing tasks + unified view
What is it?
Learning from interaction
Learning about, from, and while interacting with an external environment
Learning what to do—how to map situations to actions—so as to maximize a numerical reward signal
Reinforcement Learning

Key features?
Learner is not told which actions to take
Trial-and-Error search
Possibility of delayed reward
  ◆ Sacrifice short-term gains for greater long-term gains

The need to explore and exploit

Considers the whole problem of a goal-directed agent interacting with an uncertain environment
Supervised learning

Training Info = desired (target) outputs

Error = (target output – actual output)

Objective: get as much reward as possible

Unsupervised learning

Training Info = evaluations ("rewards" / "penalties")

"states"  "actions"
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: \( r_{t+1} \in R \)

and resulting next state: \( s_{t+1} \)
Policy at step $t$, $\pi_t$:

A mapping from states to action probabilities

$$\pi_t(s, a) = \text{probability that } a_t = a \text{ when } s_t = s$$

Reinforcement learning methods specify how the agent changes its policy as a result of experience.

Roughly, the agent’s goal is to get as much reward as it can over the long run.
Example: Tic-Tac-Toe

Assume an imperfect opponent:
— he/she sometimes makes mistakes
An RL Approach to Tic-Tac-Toe

1. Make a table with one entry per state:

<table>
<thead>
<tr>
<th>State</th>
<th>$V(s)$ – estimated probability of winning</th>
</tr>
</thead>
<tbody>
<tr>
<td>✗</td>
<td>.5</td>
</tr>
<tr>
<td>✗</td>
<td>.5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1</td>
<td>win</td>
</tr>
<tr>
<td>0</td>
<td>loss</td>
</tr>
<tr>
<td>0</td>
<td>draw</td>
</tr>
</tbody>
</table>

2. Now play lots of games.
To pick our moves, look ahead one step:

Just pick the next state with the highest estimated prob. of winning — the largest $V(s)$; a **greedy** move.

But 10% of the time pick a move at random; an **exploratory move**.

---

*Current state*  
Various possible next states
RL Learning Rule for Tic-Tac-Toe

We increment each $V(s)$ toward $V(s')$ – a backup:

$$V(s) \leftarrow V(s) + \alpha [V(s') - V(s)]$$

where

- $s$ – the state before our greedy move
- $s'$ – the state after our greedy move

a small positive fraction, e.g., $\alpha = .1$

the step-size parameter
Examples of Reinforcement Learning

- **Robocup Soccer Teams**  Stone & Veloso, Reidmiller et al.  
  World's best player of simulated soccer, 1999; Runner-up 2000

- **Inventory Management**  Van Roy, Bertsekas, Lee & Tsitsiklis  
  10-15% improvement over industry standard methods

- **Dynamic Channel Assignment**  Singh & Bertsekas, Nie & Haykin  
  World's best assigner of radio channels to mobile telephone calls

- **Elevator Control**  Crites & Barto  
  (Probably) world's best down-peak elevator controller

- **Many Robots**  
  navigation, bi-pedal walking, grasping, switching between skills...

- **TD-Gammon and Jellyfish**  Tesauro, Dahl  
  World's best backgammon player
Is a scalar reward signal an adequate notion of a goal?—maybe not, but it is surprisingly flexible.

A goal should specify what we want to achieve, not how we want to achieve it.

A goal must be outside the agent’s direct control—thus outside the agent.

The agent must be able to measure success:
- explicitly;
- frequently during its lifespan.
What’s the objective?

Suppose the sequence of rewards after step $t$ is:

$$r_{t+1}, r_{t+2}, r_{t+3}, \ldots$$

What do we want to maximize?

In general, we want to maximize the expected return, $E\{R_t\}$, for each step $t$.

**Episodic tasks**: interaction breaks naturally into episodes, e.g., plays of a game, trips through a maze.

$$R_t = r_{t+1} + r_{t+2} + \ldots + r_T$$

- Immediate reward
- Long term reward
Continuing tasks: interaction does not have natural episodes.

Discounted return:

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

where \( \gamma, 0 \leq \gamma \leq 1 \), is the discount rate.

shortsighted \( 0 \leftarrow \gamma \rightarrow 1 \) farsighted
Avoid **failure**: the pole falling beyond a critical angle or the cart hitting end of track.

As an **episodic task** where episode ends upon failure:
- reward = +1 for each step before failure
- $\Rightarrow$ return = number of steps before failure

As a **continuing task** with discounted return:
- reward = $-1$ upon failure; 0 otherwise
- $\Rightarrow$ return = $-\gamma^k$, for $k$ steps before failure

In either case, return is maximized by avoiding failure for as long as possible.
Get to the top of the hill as quickly as possible.

\[
\text{reward} = -1 \text{ for each step where not at top of hill}
\]

\[
\Rightarrow \text{ return} = -\text{number of steps before reaching top of hill}
\]

Return is maximized by minimizing number of steps reach the top of the hill.
Think of each episode as ending in an absorbing state that always produces reward of zero:

We can cover all cases by writing

\[ R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \]

\( \gamma \) can be 1 only if a zero reward absorbing state is always reached.
Time steps need not refer to fixed intervals of real time.

Actions can be low level (voltages to motors), or high level (accept job offer), “mental” (shift focus of attention), etc.

States can be low-level “sensations”, or abstract, symbolic, based on memory, or subjective (“surprised” or “lost”).

The environment is not necessarily unknown to the agent, only incompletely controllable.
Time steps need not refer to fixed intervals of real time.

Actions can be low level (voltages to motors), or high level (accept job offer), “mental” (shift focus of attention), etc.

States can be low-level “sensations”, or abstract, symbolic, based on memory, or subjective (“surprised” or “lost”).

The environment is not necessarily unknown, only incompletely controllable.

http://probo.vub.ac.be/
Single state RL

**Evaluative feedback**
- AN N-armed bandit problem
- Action value methods
  Greedy, greedy, softmax

**RL and Normal form games**
Evaluative Feedback
(see Chp 2 Sutton & Barto)

Evaluating actions vs. instructing by giving correct actions

Pure evaluative feedback depends totally on the action taken.
Pure instructive feedback depends not at all on the action taken.

Supervised learning is instructive; RL is evaluative

Associative vs. Nonassociative:
- Associative: inputs mapped to outputs; learn the best output for each input
- Nonassociative: “learn” (find) one best output

$n$-armed bandit (at least how we treat it) is:
- Nonassociative
- Evaluative feedback
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.
Suppose you form estimates

\[ Q_t(a) \approx Q^*(a) \]

action value estimates

The greedy action at \( t \) is \( a_t \)

\[ a_t^* = \arg \max_a Q_t(a) \]

\[ a_t = a_t^* \Rightarrow \text{exploitation} \]

\[ a_t \neq a_t^* \Rightarrow \text{exploration} \]

You can’t exploit all the time; you can’t explore all the time

You can never stop exploring; but you should always reduce exploring. Maybe.
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

$$Q_t(a) = \frac{r_1 + r_2 + \cdots + r_{k_a}}{k_a} \quad \text{“sample average”}$$

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

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

ε-Greedy:

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

... the simplest way to balance exploration and exploitation
$n = 10$ possible actions
Each $Q^*(a)$ is chosen randomly from a normal distribution: $\eta(0, 1)$
each $r_t$ is also normal: $\eta(Q^*(a_t), 1)$
1000 plays
repeat the whole thing 2000 times and average the results
ε-Greedy Methods on the 10-Armed Testbed

![Graph showing average reward and optimal action percentage over plays for different ε values.]
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 $\tau$ is the “computational temperature”
Recall the sample average estimation method:

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

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

Can we do this incrementally (without storing all the rewards)?

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

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

This is a common form for update rules:

$$NewEstimate = OldEstimate + StepSize[Target - OldEstimate]$$
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.

Better in the nonstationary case is:

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

for constant $\alpha$, $0 < \alpha \leq 1$

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

exponential, recency-weighted average
All methods so far depend on $Q_0(a)$, i.e., they are biased. Suppose instead we initialize the action values optimistically, i.e., on the 10-armed testbed, use

$$Q_0(a) = 5 \quad \text{for all } a$$
These are all very simple methods

- But very often used in practice

- More formal approaches, see lecture on regret minimisation.
N-Armed bandit problem is single stage or state less

An RL problem usually involves a sequention of decisions (see next lecture)

First RL in normal game settings