Advanced Topics on Artificial Intelligence

Alban Grastien

The Australian National University

Second Semester, 2020

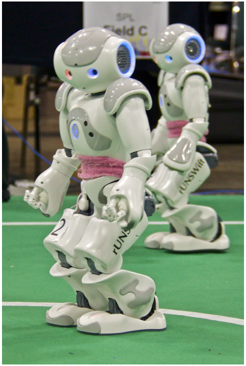
Limits of the Previous Approaches

- The approaches we looked at until now are all *model-based*.
  - What if we don't have a model?
  - How to model complex interactions in robotics?
- What if the model evolves?
- What if the state space is so large that it cannot be explored?

Examples

How do you play if you don't know how the elements interact?

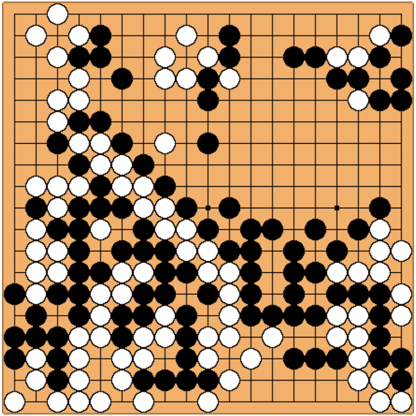
Examples



Team rUNSWift competing in the Standard Platform League at RoboCup 2010 in Singapore

Image source: wikipedia

Examples



How do you play this game if you cannot explore all the state space?

Examples

How do you ride a bike if the parameters (stiffness of the brakes, pressure and friction of the tires, etc.) are unknown and keep changing?

RL and Machine Learning

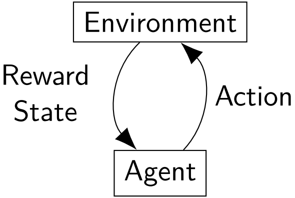
Three types of machine learning framework:

Unsupervised ML Search for correlation in data

Supervised ML Given data points on $X \times Y$, find a function that best approximates $X \to Y$

Reinforcement Learning Given an interaction between the environment and the learner, learn the course of action that maximises the reward

Reinforcement Learning Diagram



Different Properties

The framework presented in this course applies to a large variety of systems, but different systems have different properties, and some techniques are not useable when some properties hold:

- Markov vs pseudo-Markov systems
  - The approaches here sometimes work for non-Markov systems, but this is not always true
- Real vs virtual systems (can be started from scratch)
  - Virtual systems allow us to test any state many times
- Expensive vs expandable systems
  - Expandable systems allow us to take risks
- Systems with static or varying dynamics
  - Systems with varying dynamics should forget obsolete information

Bandit

Problem description

- There is only one state
- There are k actions available, $A = \{a_1, \ldots, a_k\}$
- Each action a_i has a stochastic reward $r(a_i) : Prob(\mathbb{N})$ with expected reward $e(a_i)$
- You have to repeatedly choose one action
- Your goal is to maximise reward

Examples

- Drug prescriptions
- Project resource allocation
- Adaptive routing
- Financial portfolio design

Greedy Strategy

- Select the best action so far.
- Learn the reward associated with the best action.

Greedy Strategy

Greedy-Strategy

- $R := \{\}$ // cumulative reward of each action
- $N := \{\}$ // number of times each action was used
- **for all** action $a \in A$
  - *reward* $:= execute(a)$
  - $R[a] :=$ *reward*
  - $N[a] := 1$
- **repeat**
  - Select action a that maximises $R[a]/N[a]$
  - *reward* $:= execute(a)$
  - $R[a] \mathrel{+}=$ *reward*
  - $N[a] \mathrel{+}= 1$

Explaining Greedy Strategy

Idea:

- First get an estimate of the value of each action
- Select the action that looks best
- Switch to a different action when the current one seems sub-optimal

Property:

- If the underlying dynamics is stationary, this algorithm converges to a single action

Example

Example with three actions

- r_1: uniform in $[3, 5]$ (expected 4)
- r_2: uniform in $[0, 10]$ (expected 5)
- r_3: uniform in $[2, 4]$ (expected 3)

Execution:

- First executions: $a_1 \to 4.7$, $a_2 \to 4.2$, $a_3 \to 3.8$
- Next executions:

$R/N[a_1]$	$R/N[a_2]$	$R/N[a_3]$	next action	next reward
4.7	4.2	3.8	a_1	3.9
4.3	4.2	3.8	a_1	3.4
4	4.2	3.8	a_2	7.4
4	5.8	3.8	a_2	3.1
4	4.9	3.8	a_2	...

Problem

What is wrong with this method?

Problem

What is wrong with this method?

Answer:

- If the first execution of the optimal action gives a small reward (smaller than the expected reward of some other action), this action will be ignored forever.

Example

Example with three actions

- r_1: uniform in $[3, 5]$ (expected 4)
- r_2: uniform in $[0, 10]$ (expected 5)
- r_3: uniform in $[2, 4]$ (expected 3)

Execution:

- First executions: $a_1 \to 4.7$, $a_2 \to 3.7$, $a_3 \to 3.8$
- Next executions:

$R/N[a_1]$	$R/N[a_2]$	$R/N[a_3]$	next action	next reward
4.7	3.7	3.8	a_1	3.9
4.3	3.7	3.8	a_1	3.4
4	3.7	3.8	a_1	3.2
3.8	3.7	3.8	a_1	4.3
3.9	3.7	3.8	a_1	...

It is possible (non-zero probability) that a_2 will never be considered again.

Solution (1)

- Start with an optimistic estimate $R[a]$ for every action a
- Essentially forces each action to be performed a minimal number of times
- Reduces the chances that unlucky executions give us poor estimates

Solution (2)

- Perform seemingly sub-optimal actions every now and then, to check whether the estimate is correct.
- How often?
- This question is the exploration / exploitation dilemma:
  - **Exploration** try sub-optimal actions in order to discover better actions
  - **Exploitation** perform action that currently seems the best

Epsilon-Greedy Strategy

Epsilon-Greedy Strategy

- $R := \{\}$ // cumulative reward of each action
- $N := \{\}$ // number of times each action was used
- **for all** action $a \in A$
  - *reward* $:= execute(a)$
  - $R[a] :=$ *reward*
  - $N[a] := 1$
- **repeat**
  - With probability ε, choose an action a at random
  - Otherwise, select action a that maximises $R[a]/N[a]$
  - *reward* $:= execute(a)$
  - $R[a] += $ *reward*
  - $N[a] += 1$

Eventually, the estimate of each action is accurate.

Nonstationary Problems

- How do we handle problems in which the reward of the actions evolves?
- Previous expected reward was $e(a)$, now is $e'(a)$
- We do not want the old (obsolete) executions of an action to weight as much as the last one.
- But we do not want to remember all the history of rewards.

$\rightarrow$ Use incremental implementation instead.

Incremental Strategy (Temporal Difference)

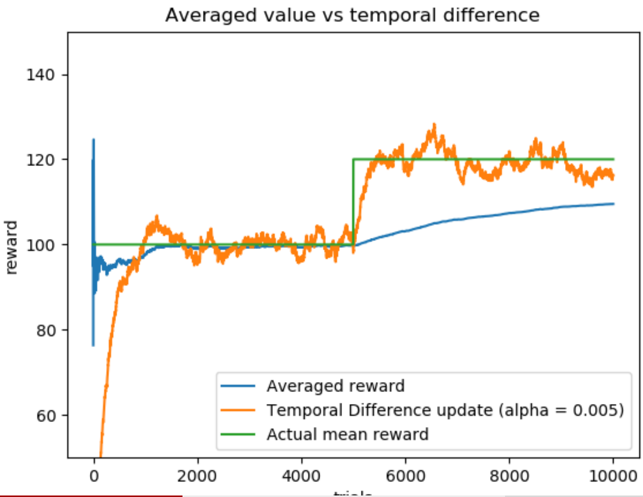
Incremental Strategy

- $E := \{\}$ // estimate of each action
- **for all** action $a \in A$
  - $E[a] := execute(a)$
- **repeat**
  - With probability ε, choose an action a at random
  - Otherwise, select action a that maximises $E[a]$
  - $r := execute(a)$
  - $\Delta := r - E[a]$
  - $E[a] := E[a] + \alpha_t \times \Delta$

α_t is the step size, and it indicates how much the new observation matters.

Example of a Nonstationary Problem

(Python experiment)



How to Choose α_t

- If α_t is a constant, the estimate $E(a)$ becomes unstable.
- If α_t decreases too quickly, $E(a)$ does not reach $e(a)$.

Convergence is guaranteed if

$$\Sigma_{i=1}^{\infty}\ \alpha_t = \infty \quad \text{and} \quad \Sigma_{i=1}^{\infty}\ \alpha_t^2 < \infty$$

If $E[a]$ is the average reward, $\alpha_t = 1/t$

Summary

- Exploration vs exploitation:
    - Often perform best options, to get max rewards
    - try seemingly suboptimal options to get a better estimate

  $\rightarrow$ Epsilon-greedy policy

- Update the estimate of the payoff for each action
    - Either average or temporal difference
    - In non-stationary scenarios, give last observations a bigger weight