RL 3: Reinforcement Learning

$\mathcal{Q}$-Learning

Michael Herrmann

University of Edinburgh, School of Informatics

21/01/2014

Last time: Multi-Armed Bandits (Points to remember)

- MAB applications do exist (e.g. experimental design)
- Exploration-exploitation dilemma: Both the reward and information about the reward are important. Although dilemma in general not unambiguously solvable, for MAB reasonable solutions exist.
- Estimate parameters of the action-specific reward distributions
- $Q_k = Q_{k-1} + \frac{1}{k}\left(r_k - Q_{k-1}\right)$ gives an exact mean over r_k for $Q_0 = 0$
- $Q_{k+1} = (1-\alpha)\, Q_k + \alpha r_{k+1}$ gives an exponentially weighted average (for non-stationary problems; initialisation usually does not matter)
- regret: $\rho = T\mu^* - \sum_{t=1}^{T} \bar{r}_t \quad$ where $\mu^* = \max_k \mu_k$
- ε-greedy policy has an asymptotic regret (not if ε decays sufficiently slowly)
- Optimistic initialisation of Q often provides a good exploration (not if reality is even better or if you are unlucky)
- Boltzmann action selection $\frac{e^{Q_t(a)/\tau}}{\sum_{b=1}^{N} e^{Q_t(b)/\tau}}$ with temperature τ is an reasonable exploration scheme, but ε-greedy is often not bad, too.

For more details see e.g. ICML 2011 Tutorial Introduction to Bandits: Algorithms and Theory, Jean-Yves Audibert, Remi Munos https://sites.google.com/site/banditstutorial/

The Gittins Index (still for MABs)

- Each arm delivers reward with a probability which may change through time but only when arm is pulled
- Not only adapt estimates of the reward by moving averages, also maximise discounted rewards: exp. discount factor $\gamma < 1$
- γ relates to $1 - \alpha$ (exp. average) and time scale

$$\tau \sim \frac{1}{1-\gamma} = \frac{1}{\alpha}$$

- All you need to do is compute an "index" for each arm and play the one with the highest index:

$$\nu_i = \sup_{T>0} \left\langle \sum_{t=0}^{T} \gamma^t R^i(t) \right\rangle$$

- Often the index is defined with a normalising denominator.

The Gittins Index Theorem

- Gittins index theorem: The optimal policy is to play at each epoch a bandit of greatest Gittins index.
- Gittins index for any given arm is independent on other arms
  - Adding/removing arms does not really change computation
  - Once you have a good arm, keep playing it
  - Arms are not updated unless used (Exploration?)

See e.g.:

R. Weber: On the Gittins index for multiarmed bandits. Annals of Appl. Prob. (1992) 1024-1033.

Beyond the MAB Model: Towards RL

- Previously, we were in a single casino and the only decision is to pull from a set of N arms
  - not more than a single state!

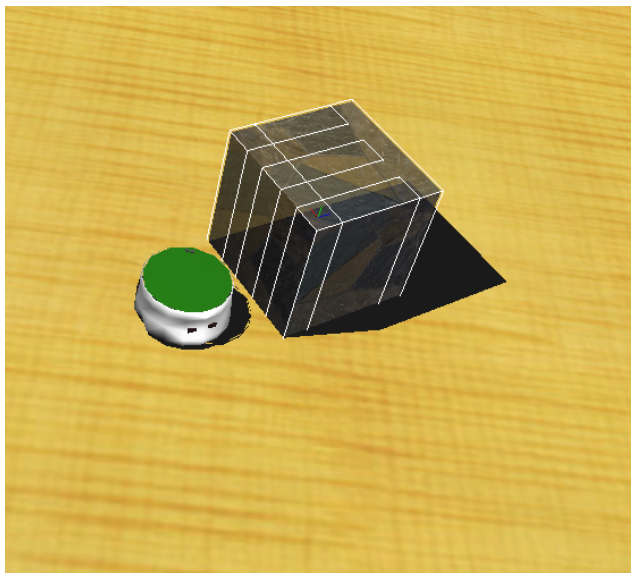
Now,

- What if there is more than one state?
- So, in this state space, what is the effect of the distribution of payout changing based on how you pull arms?
- What happens if you only obtain a net reward corresponding to a long sequence of arm pulls (at the end)?

Example

- A robot learns to "stop-and-back-up-and-turn" (a_1) in near an obstacle (s_1) and to "go" (a_2) away from an obstacle (s_2) from $r(s_2) = 1$ and $r(s_1) = 0$

- How does the robot find out that a_2 is actually better in s_1 where both actions lead to $r = 0$?
- How can the robot be kept from preferring a_1 in state s_2?
- What can go wrong?



Brute force

1. For each possible policy, sample returns while following it
2. Choose the policy with the largest expected return

If a trial takes T time steps and

Policy assigns to each time $t \leq T$ an action $a \in \mathcal{A}$ by

$$\pi : \mathcal{T} \mapsto \mathcal{A},$$

where $\mathcal{T}$ is the set of decision times, i.e. $|\mathcal{T}| = T$.

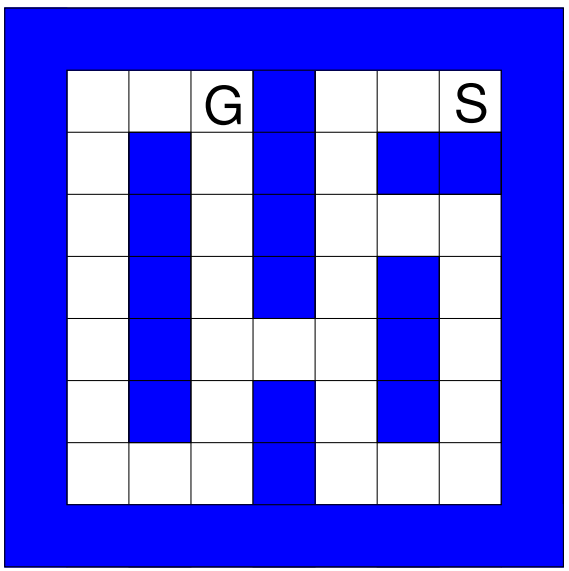
In principle there are thus $|\mathcal{A}|^T$ policies.

(assuming the cardinalities $|\mathcal{A}|$ and T are finite)

For the bandit problem with $T = 1$ this worked well.

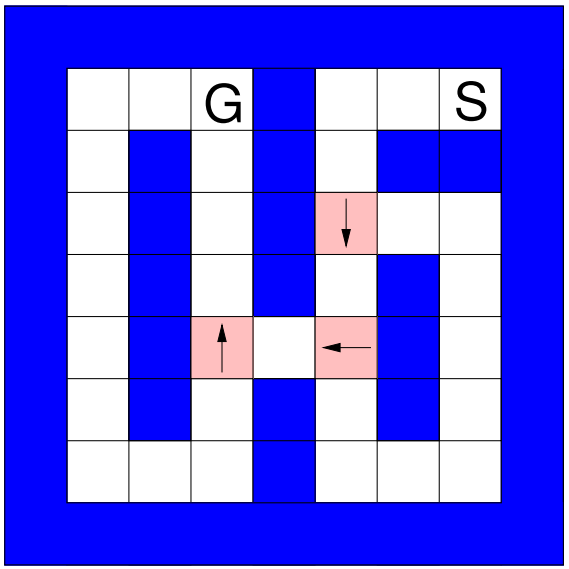
Improving beyond brute force

Consider a maze of 7×7 fields, i.e. $|\mathcal{T}| \leq 49$. There are 4 directions.
So there are at most 4^{49} policies.
Many can be excluded when using the immediate punishment given at bumping into a wall.
Employing in addition an inhibition-of-return principle usually even less actions are admissible.
Many policies differ only in the part after the goal or are equivalent (e.g. "first up then left" may equal "first left then up" if there are no obstacles nearby).



Improving beyond brute force

- Branch and bound
- Define policy based on states $s \in \mathcal{S}$
  - Some states are inaccessible (try not to represent what is never experienced)
  - Avoid reaching at state repeatedly in one run (e.g. by intrinsic reward/punishment)
  - But: states need to incorporate all the relevant information*



An efficient solution.

- Introduce a *geometry* on the set of states
- Allow samples generated from one policy to influence the estimates made for another.

*E.g. in the pendulum swing-up task states typically include position *and* velocity.

Preliminary considerations

- If much information about the task is available, it should be used
- Configurations of different actions may be equivalent $a_1 = E$ and $a_3 = N$ is as good as $a_1 = S$ and $a_3 = S$ (and $a_2 = W$)
- or even misleading: $a_1 = E$ and $a_2 = W$ is better than $a_1 = S$ and $a_2 = S$, although the globally best policy has $a_1 = S$ (and $a_2 = W$, $a_3 = N$)
- Earlier decisions may influence later decisions (consider agent-centered code: L, R, S, B)
- Be that as it may, do we have a chance to deal with stochastic problems of this type?
- There are a number of approach, we choose one (and consider later others)

Off-policy learning: $\mathcal{Q}$-Learning

Consider a person moving randomly through Edinburgh in search for a nice place to have dinner. She eventually finds a place and enjoys a great dinner.

Later the person doesn't remember the location of the place but she knows that it was near a statue of a dog.

Later the agents stumbles into the statue of the dog. She does not remember how she got there, but she remembers that it was near some elephant plate in shopping window.

Later she finds the elephant image again after having been at a library.

Later she bumps into the dog again, and remember that she was that time in front of a museum before.

Much later ... ... all sites of Edinburgh are connected by some "nearness" relation. And the agent knows that different sites are in different "nearness" steps from that restaurant.

Off-policy learning: $\mathcal{Q}$-Learning

Assume that the cardinalities $|\mathcal{A}|$ and $|\mathcal{S}|$ are finite, and $\mathcal{T} \subseteq \mathbb{N}_0$.

By definition, states contain the relevant information about the system, i.e. the value of an action can depend only on the state:

$$\mathcal{Q}(s, a)$$

Intuitively, the $\mathcal{Q}$ function expresses the expected reward when being in state s and applying a.

However, often there is no reward ($r = 0$). Then, differently from MAB, it is still an option to move to a "better" state.

The value $V(s_1)$ of a state s_1 is defined as the value of the best action that can be applied in this state

$$V(s_1) = \max_{a \in \mathcal{A}} (Q(s_1, a))$$

We have to solve a MAB for each state!

Off-policy learning: $\mathcal{Q}$-Learning

Taken together, if s_1 is reached from s due to a, then (preliminarily)

$$\mathcal{Q}(s,a) = r(s,a) + V(s_1)$$

Immediate reward (if there is any)

plus

Value of the next state (assuming best possible action)

s_1 reached from s by action a

How can this work?
An artificial example

It could happen that $s_0 = s_1$ (taking a_0). If also $r(s_0, a_0) = 0$, then

$$\mathcal{Q}(s_0, a_0) = V(s_0)$$

If in addition

$$a_0 = \arg\max_{a \in \mathcal{A}} (\mathcal{Q}(s_0, a))$$

then the value of $\mathcal{Q}$ for will depend on the initialisation.

(Recall what we have said about exploration and initialisation in MABs)

- General reachability
- Possibly via reinitialisation (upon reaching goal/timeout/loss)

Iterative learning rule

$$\rho_t = r\left(s_{t-1}, a_{t-1}\right) + V\left(s_t\right)$$

Combined reward = immediate reward + value of new state

$$\mathcal{Q}_t\left(s_{t-1}, a_{t-1}\right) = \mathcal{Q}_{t-1}\left(s_{t-1}, a_{t-1}\right) + \eta\left(\rho_t - \mathcal{Q}_{t-1}\left(s_{t-1}, a_{t-1}\right)\right)$$

When observing a state transition from s_{t-1} to s_t under action a_{t-1}, then use a sliding average to estimate the combined reward.

Notes:

- Not all actions $a \in \mathcal{A}$ need to be admissible in all states $s \in \mathcal{S}$.
- For $\mathcal{Q}_t$ the parameter t refers to update steps, for s_t and a_t is refers to the physical time of the system. Often it is reasonable to keep the two parameters in sync.
- Temporally a_{t-1} is in between s_{t-1} and s_t, we denote it by $t-1$ by convention

Assume again that $s_0 = s_1$, but now with a large $r(s_0, a_0)$, then

$$\mathcal{Q}_t(s_0, a_0) = r(s_0, a_0) + V(s_0)$$

$$\mathcal{Q}_t(s_0, a_0) = \mathcal{Q}_{t-1}(s_0, a_0) + \eta\left(r(s_0, a_0) + V(s_0) - \mathcal{Q}_{t-1}(s_0, a_0)\right)$$

If in addition

$$a_0 = \arg\max_{a \in \mathcal{A}} \left(\mathcal{Q}(s_0, a)\right)$$

and

$$V(s_0) = \max_{a \in \mathcal{A}} \left(\mathcal{Q}(s_0, a)\right)$$

then $\mathcal{Q}(s_0, a_0)$ will grow without bounds.

If, in a different setting, $r(a_t, s_t) = 0$ for many or few steps before finally $r > 0$ is achieved, $\mathcal{Q}$ would be the same.

$\Longrightarrow$ Value function should express discounted reward

Value functions

The value of a state is the total future (discounted) reward that is expected when choosing the presently known best action for this state continue with the policy that is currently considered optimal.

$$\begin{aligned} V(s_t) &= r(s_t, a^*(s_t)) + r(s_{t+1}, a^*(s_{t+1})) + r(s_{t+2}, a^*(s_{t+2})) + \cdots \\ &= \sum_{k=t}^{\infty} r(s_k, a^*(s_k)) \end{aligned}$$

Discounted version $\gamma \leq 1$:

$$\begin{aligned} V(s_t) &= r(s_t, a^*(s_t)) + \gamma r(s_{t+1}, a^*(s_{t+1})) + \gamma^2 r(s_{t+2}, a^*(s_{t+2})) + \cdots \\ &= \sum_{k=t}^{\infty} \gamma^{k-t} r(s_k, a^*(s_k)) \end{aligned}$$

Discounted update

$$
\begin{aligned}
V(s_t) &= \sum_{k=t}^{\infty} \gamma^{k-t} r(s_k, a^*(s_k)) \\
&= r(s_k, a^*(s_k)) + \sum_{k=t+1}^{\infty} \gamma^{k-t} r(s_k, a^*(s_k)) \\
&= r(s_k, a^*(s_k)) + \gamma \sum_{k=t+1}^{\infty} \gamma^{k-(t+1)} r(s_k, a^*(s_k)) \\
& r(s_k, a^*(s_k)) + \gamma V(s_{t+1})
\end{aligned}
$$

The $\mathcal{Q}$-function is updated analogously, but for an arbitrary action

$$\mathcal{Q}_{t+1}(s_t, a_t) = \mathcal{Q}_t(s_t, a_t) + \eta(r(s_t, a_t) + \gamma V(s_{t+1}) - \mathcal{Q}_t(s_t, a_t))$$

$\mathcal{Q}$-learning was first introduced by C. J. C. H. Watkins (1989)

How does this work?
A final artificial example

Assume again that $s_0 = s_1$, but now with $r(s_0, a_0) = r_{\max}$, then

$$\Delta Q_t(s_0, a_0) = \eta\left(r(s_0, a_0) + \gamma V(s_0) - Q_t(s_0, a_0)\right)$$

Choose always

$$a_0 = \arg\max_{a \in \mathcal{A}}\left(Q(s_0, a)\right)$$

i.e.

$$V(s_0) = \max_{a \in \mathcal{A}}\left(Q(s_0, a)\right) = Q(s_0, a_0)$$

stationarity at

$$0 = \eta\left(r_{\max} + \gamma V(s_0) - Q_t(s_0, a_0)\right)$$

$$\frac{r_{\max}}{1-\gamma} = Q_t(s_0, a_0)$$

finite for $\gamma < 1$ (remember that γ is typically close to 1)

If $r_{\max}$ is known, this can be useful for optimistic initialisation.

Toy Example: 1D maze
Problem set up

- Track $\mathcal{S}$: $|\mathcal{S}| = N$ spaces numbered $1, \ldots, N$
- Actions: $\mathcal{A} = \{\text{left, right}\} \equiv \{-1, +1\}$
- Reward: $r(s,a) = r(s) = \begin{cases} 1 & \text{if } s = 1 \text{ (goal)} \\ 0 & \text{otherwise} \end{cases}$
- Initialisation $\mathcal{Q}(s,a) = 0 \ \forall s, a$ (not really optimistic)
- Exploration: ε-greedy with (initially) $\varepsilon = 1$, i.e. random moves later $\varepsilon \to 0$ and $a_t = \arg\max_a Q(s_t, a)$ for non-random moves
- Starting at random point $s_1 = k$
- Restart after reaching goal or if $t = |\mathcal{T}|$ with e.g. $|\mathcal{T}| = N^2$ (?)

Toy Example: 1D maze
Solution (for $\eta = 1$)

- If $s_1 = 1$ make a move (go right) and update $\mathcal{Q}(1, \text{right}) = 1$, restart
- If $s_1 > 1$ then perform a random move leading to s_2
  - if $s_2 = 1$ then update $\mathcal{Q}(2, \text{left}) = 0 + \gamma V(1)$, restart
  - if $s_2 > 1$ then $\mathcal{Q}(s_2, a)$ remains 0

After ε decayed, we arrive at the value function

for $N > k > 1$: $\mathcal{Q}(k, \text{left}) = \gamma^{k-1}$, $\mathcal{Q}(k, \text{right}) = \gamma^{k-3}$

for the ends of the track: $\mathcal{Q}(1, \text{right}) = 1$, $\mathcal{Q}(1, \text{left}) = 0$, $\mathcal{Q}(N, \text{left}) = \gamma^{N-1}$, $\mathcal{Q}(N, \text{right}) = 0$

Homework: Write a simulation for this example

Discussion

- *Off-policy* algorithm: Learning the value of state-action pairs independently of their position in a policy, i.e. learning generalises across policies
  - Policies are not factorisable in general
  - If the algorithm contains exploration it is not performing optimally and therefore is does not know the value of the currently optimal policy
- Look-up table representation of the $\mathcal{Q}$-function ($Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$) is not very efficient. We will later use function approximation
- Other exploration schemes are clearly possible
- Convergence (Watkins and Dayan, 1992): We need a solid theoretical basis for this
- Complexity: [will take time]

Preliminary discussion of the relevant time scales

1. Behavioural time scale $1/(1-\gamma)$ (discount factor)
2. Sampling in the estimation of the $\mathcal{Q}$-function η (learning rate)
3. Exploration ε (e.g. for ε-greedy strategy)

$$1-\gamma \gg \eta \gg \varepsilon$$

Adaptation of time scales: Initially $1-\gamma \approx \eta \approx \varepsilon$ is possible, then decrease ε faster than η to reach the separation of time scales asymptotically.

Practically, choose number of trials $M < \infty$ and set $\eta \sim 1 - m/M$ and $\varepsilon \sim (1-m/M)^2$, $m = 1, \ldots, M$. Not theoretically justified.

γ may be moved a bit towards 1, i.e. explore first short time scales, then longer ones (assuming there is some reward in both cases)

Examples and Applications

- Higher dimensional mazes, grid worlds, search trees etc.
- Games, decision making, modelling the control of behaviour
- Cart-pole, pendulum swing up, multiple pendula, mountain-car
- Chaos control, robot control
- Industrial control, production control, automotive control, autonomous vehicles control, logistics, telecommunication networks, sensor networks, ambient intelligence, robotics, finance (s. Real World Applications of Reinforcement Learning at International Joint Conference on Neural Networks 2012)

Outlook to next lectures: Will establish a theoretical basis including

- Markov decision problems
- Dynamic programming
- Monte Carlo methods