Reinforcement Learning, yet another introduction. Part 3/3: Control problems

Emmanuel Rachelson (ISAE - SUPAERO)

The madhatter's casino

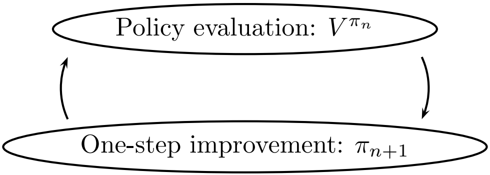
The madhatter's casino

States	4 rooms
Actions	3 slot machines (stacks of cards)
Transitions	From room to room, following the Mad hatter's will!
Rewards	Cups of tea

The madhatter's casino

Brainstorming

Reminder: Policy Iteration



Asynchronous Policy Iteration

Pb of DP methods: long sweeps, lots of useless backups.

Two types of backups:

Update Q: $Q(s,a) \leftarrow r(s,a) + \sum_{s'} p(s'|s,a) Q(s', \pi(s'))$

Improve π: $\pi(s) \leftarrow \underset{a}{\operatorname{argmax}}\, Q(s,a)$

Asynchronous Policy Iteration

Pb of DP methods: long sweeps, lots of useless backups.

Two types of backups:

Update Q: $Q(s,a) \leftarrow r(s,a) + \sum_{s'} p(s'|s,a) Q(s', \pi(s'))$

Improve π: $\pi(s) \leftarrow \operatorname*{argmax}_a Q(s,a)$

Value Iteration

In each state, one update of Q and one improvement of π.

Asynchronous Policy Iteration

Pb of DP methods: long sweeps, lots of useless backups.

Two types of backups:

Update Q: $Q(s,a) \leftarrow r(s,a) + \sum_{s'} p(s'|s,a) Q(s', \pi(s'))$

Improve π: $\pi(s) \leftarrow \operatorname{argmax}_a Q(s,a)$

Policy Iteration

Update Q in all states until convergence, then update π in all states.

Asynchronous Policy Iteration

Pb of DP methods: long sweeps, lots of useless backups.

Two types of backups:

Update Q: $Q(s,a) \leftarrow r(s,a) + \sum_{s'} p(s'|s,a) Q(s', \pi(s'))$

Improve π: $\pi(s) \leftarrow \operatorname{argmax}_a Q(s,a)$

Asynchronous Dynamic Programming

As long as every state is visited infinitely often for Bellman backups on V or π, the sequences of V_n and π_n converge to V^* and π^*.

DP converges whatever the ordering of the backups!

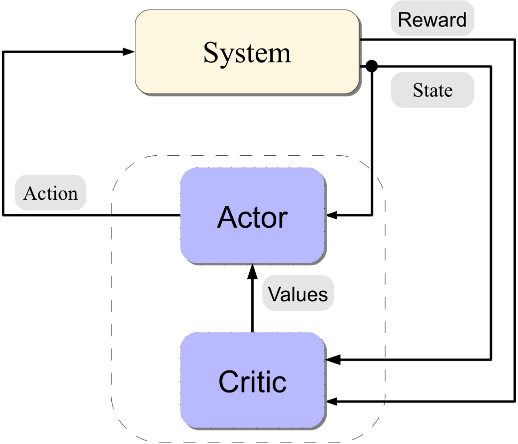
Generalized Policy Iteration

Two interacting processes: policy evaluation and policy improvement.

Not from the model anymore, but from samples.

Generalized Policy Iteration

The bigger picture: actor-critic architectures



Almost all RL algorithms fall into an A-C architecture.
Let's look at several ones.

SARSA — the idea

Let's try to evaluate the current policy's value $\rightarrow Q$,
...while updating π as Q-greedy.
What happens?
Convergence to π^*

SARSA — the TD update

Remember TD(0): $\delta = r + \gamma V(s') - V(s)$

$V(s') = Q(s', \pi(s'))$

Evaluate the current π: $\delta = r + \gamma Q(s', a') - Q(s, a)$

SARSA — the algorithm

In s, choose (*actor*) a using Q, then repeat:

1. Observe r, s'
2. Choose a' (*actor*) using Q
3. $\delta = r + \gamma Q(s', a') - Q(s, a)$
4. $Q(s, a) \leftarrow Q(s, a) + \alpha\delta$
5. $s \leftarrow s'$, $a \leftarrow a'$

SARSA — convergence

Convergence of SARSA

If, in the limit,

1. all (s,a) are visited infinitely often,
2. the actor converges to the Q-greedy policy,

   Greedy in the limit of infinite exploration (GLIE)

then the actor converges to π^*.

To insure (1), necessary exploration!

Implementing an actor:

- ε-soft, ε-greedy: $\pi\left(a \neq \underset{a'}{\operatorname{argmax}}\, Q(s,a')|s\right) = \varepsilon$
- Boltzmann policies $\pi(a|s) = \frac{e^{\frac{Q(s,a)}{\tau}}}{\sum_{a'} e^{\frac{Q(s,a')}{\tau}}}$

SARSA — On-policy critic

SARSA constantly evaluates the current π...
...that shifts towards π^*

When the critic evaluates the current actor's policy,
one talks of *on-policy* algorithms.

Example of *off-policy* method: Q-learning.

Q-learning — the idea

The critic tries to approximate Q^*,
independently of the actions taken by the actor.

Then, as the actor gets Q-greedy, it converges to π^*.

Q-learning — the TD update

Remember TD(0): $\delta = r + \gamma V(s') - V(s)$

$$V(s') = Q(s', \pi(s'))$$

Evaluate the current π: $\delta = r + \gamma \max_{a'} Q(s', a') - Q(s, a)$

Q-learning — the algorithm

In s,

1. Choose a (*actor*) using Q
2. Observe r, s'
3. $\delta = r + \gamma \max_{a'} Q(s', a') - Q(s, a)$
4. $Q(s, a) \leftarrow Q(s, a) + \alpha\delta$
5. $s \leftarrow s'$ and repeat

Q-learning — convergence

Convergence of Q-learning

As for SARSA, if, in the limit,

1. all (s, a) are visited infinitely often,
2. the actor converges to the Q-greedy policy,

then the actor converges to π^*.

Again, to insure (1), necessary exploration!

Implementing an actor:

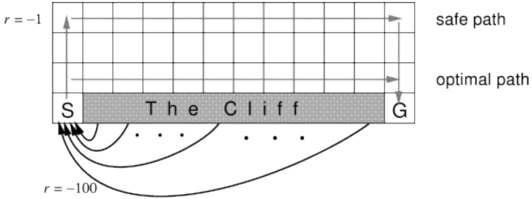
- ε-soft, ε-greedy: $\pi\left(a \neq \underset{a'}{\operatorname{argmax}}\, Q(s,a')|s\right) = \varepsilon$
- Boltzmann policies $\pi\,(a|s) = \frac{e^{\frac{Q(s,a)}{\tau}}}{\sum_{a'} e^{\frac{Q(s,a')}{\tau}}}$

Q-learning — Off-policy critic

Q-learning evaluates the optimal Q^*
and not the current π.

It is an *off-policy* algorithm.

Funny comparison: The cliff



States	grid positions
Actions	N, S, E, W
Transitions	deterministic
Rewards	-100 for falling, -1 otherwise.

- What is the optimal policy?
- With a fixed $\varepsilon = 0.1$ for ε-greedy π, what do you think will happen?
- What if ε goes to zero?

1. The mad hatter's casino
2. Back to Policy Iteration: Generalized PI and Actor-Critic methods
3. Online problems, the exploration vs. exploitation dilemma
   - On-policy TD control: SARSA
   - Off-policy control: Q-learning
   - Funny comparison
4. Offline problems, focussing on the critic alone
   - Fitted *Q*-iteration
   - Least Squares Policy Iteration
5. An overview of control learning problems

Offline problems

No interaction with the environment.
Pre-acquired data: $\mathscr{D} = \{(s_i, a_i, r_i, s'_i\}_{i \in [1,N]}$

No exploration vs. exploitation dilemma.
Can usually tackle larger problems.

New problem:
Samples only in a subset of $S \times A$,
need to generalize and approximate Q or π.

Fitted Q-iteration — the idea

Generalization of Value Iteration.

Reminder (VI): $V_{n+1}(s) = \max_a \left[r(s,a) + \gamma \sum_{s'} P(s'|s,a) V_n(s') \right]$

Q-iteration: $Q_{n+1}(s,a) = r(s,a) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q_n(s',a')$

online $\rightarrow$ Q-learning

offline $\rightarrow$ Fitted Q-iteration

Fitted Q-iteration — the algorithm

The exact case:

- For each (s, a)
  - $\mathscr{D}_{s,a}$ = subset of $\mathscr{D}$ starting with (s, a)
  - $Q_{n+1}(s,a) = \frac{1}{|\mathscr{D}_{s,a}|} \sum_{(s,a,r,s') \in \mathscr{D}_{s,a}} r + \gamma \max_{a'} Q_n(s', a')$
- Repeat until convergence of Q

With black-box function approximation:
$\hat{Q}_0(s,a) = 0$,

- Build $\mathscr{T} = \left\{ (s_i, a_i), r_i + \gamma \max_{a'} \hat{Q}_n(s'_i, a') \right\}_{i \in [1,N]}$
- Train regressor $\hat{Q}_{n+1}(s,a)$ from $\mathscr{T}$
- Repeat until convergence of Q

Fitteed Q-iteration — properties

Offline
Model-free
Off-policy
Batch

Converges under conditions on the regressor,
to a neighbourhood of Q^*.
Might diverge.
Simple and efficient.

Least Squares Policy Iteration — the idea

Suppose $Q^{\pi}(s,a) = w^T \phi(s,a)$

$Q^{\pi} = r^{\pi} + \gamma P^{\pi} Q^{\pi}$ becomes:

$$w_{\pi}^T \Phi = r^{\pi} + \gamma P^{\pi} w_{\pi}^T \Phi$$

And ... $w_{\pi} = \left[\Phi^T \left(\Phi - \gamma P^{\pi} \Phi\right)\right]^{-1} \Phi^T r^{\pi}$

... which can be approximated by summing over the elements of $\mathscr{D}$

LSTD-Q

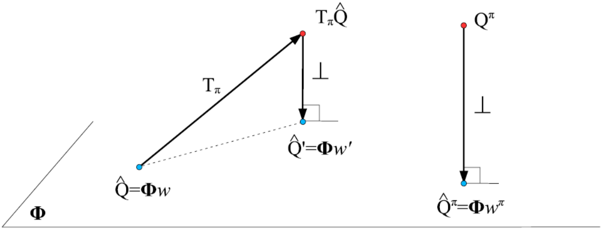
$$A = \Phi^T(\Phi - \gamma P^\pi \Phi) \approx \frac{1}{N}\sum_{i=1}^{N}\left[\phi(s_i,a_i)\left(\phi(s_i,a_i) - \gamma\phi(s_i',\pi(s_i'))\right)^T\right]$$

$$b = \Phi^T r^\pi \approx \frac{1}{N}\sum_{i=1}^{N}[\phi(s_i,a_i)r_i]$$

$$w_\pi = A^{-1}b$$
$$\pi' = Q\text{-greedy}$$

Repeat.

LSPI — graphical explanation



LSPI — properties

Offline
Model-free
Off-policy
Batch

Always converges.
But to what?
Maximal use of $\mathscr{D}$.
Difficulty: choose $\phi(s, a)$.

Let's take a step back

So far, we can classify our algorithms / problems as:

- Model-based vs. Model-free
- On-policy vs. Off-policy
- Online vs. Episodic vs. Offline
- Incremental vs. Batch

Challenges

- Large, continuous, hybrid state and/or action spaces
- Exploration vs. exploitation
- Finite sample convergence bounds
- Lots of applications in control systems, finance, games, etc. and more and more successes

A lot of related approaches and methods in the literature!