Künstliche Intelligenz

18. Reinforcement Learning

Jun.-Prof. Dr.-Ing. Stefan Lüdtke
Institut für Visual & Analytic Computing
Universität Rostock

Learning Objectives - Reinforcement Learning

At the end of the class you should be able to:

- Explain the relationship between decision-theoretic planning (MDPs) and reinforcement learning

Learning Objectives - Reinforcement Learning

At the end of the class you should be able to:

- Explain the relationship between decision-theoretic planning (MDPs) and reinforcement learning
- Implement basic state-based reinforcement learning algorithms: Q-learning

Reinforcement Learning

What should an agent do given:

- Prior knowledge
- Observations
- Goal

Reinforcement Learning

What should an agent do given:

- Prior knowledge — possible states of the world
  possible actions
- Observations
- Goal

Reinforcement Learning

What should an agent do given:

- Prior knowledge — possible states of the world
  possible actions
- Observations — current state of world
  immediate reward / punishment
- Goal

Reinforcement Learning

What should an agent do given:

- Prior knowledge — possible states of the world
  possible actions
- Observations — current state of world
  immediate reward / punishment
- Goal — act to maximize accumulated (discounted) reward

Reinforcement Learning

What should an agent do given:

- Prior knowledge — possible states of the world
  possible actions
- Observations — current state of world
  immediate reward / punishment
- Goal — act to maximize accumulated (discounted) reward
- Like decision-theoretic planning, except model of dynamics and model of reward not given.

Reinforcement Learning Examples

- Game -

Reinforcement Learning Examples

- Game - reward winning, punish losing

Reinforcement Learning Examples

- Game - reward winning, punish losing
- Dog -

Reinforcement Learning Examples

- Game - reward winning, punish losing
- Dog - reward obedience, punish destructive behavior

Reinforcement Learning Examples

- Game - reward winning, punish losing
- Dog - reward obedience, punish destructive behavior
- Robot -

Reinforcement Learning Examples

- Game - reward winning, punish losing
- Dog - reward obedience, punish destructive behavior
- Robot - reward task completion, punish dangerous behavior

Experiences

- We assume there is a sequence of experiences:

  $$\mathit{state}, \mathit{action}, \mathit{reward}, \mathit{state}, \mathit{action}, \mathit{reward}, \ldots.$$

Experiences

- We assume there is a sequence of experiences:

  $$\mathit{state}, \mathit{action}, \mathit{reward}, \mathit{state}, \mathit{action}, \mathit{reward}, \ldots.$$

- The sequence of experiences up to the time the agent has to choose its action is it history
- The agent has to choose its action as a function of its history.

Experiences

- We assume there is a sequence of experiences:

  $$\mathit{state}, \mathit{action}, \mathit{reward}, \mathit{state}, \mathit{action}, \mathit{reward}, \ldots.$$

- The sequence of experiences up to the time the agent has to choose its action is it history
- The agent has to choose its action as a function of its history.
- At any time it must decide whether to

Experiences

- We assume there is a sequence of experiences:

  $$\mathit{state}, \mathit{action}, \mathit{reward}, \mathit{state}, \mathit{action}, \mathit{reward}, \ldots.$$

- The sequence of experiences up to the time the agent has to choose its action is it history
- The agent has to choose its action as a function of its history.
- At any time it must decide whether to
  - explore to gain more knowledge
  - exploit knowledge it has already discovered

Why is reinforcement learning hard?

- What actions are responsible for a reward may have occurred a long time before the reward was received.
  - The dog is expected to determine that eating the shoe at the start of the day is what was resposible for it being scolded at the end of the day.

Why is reinforcement learning hard?

- What actions are responsible for a reward may have occurred a long time before the reward was received.
  - The dog is expected to determine that eating the shoe at the start of the day is what was resposible for it being scolded at the end of the day.
- The long-term effect of an action depend on what the agent will do in the future.
  - It might be okay for a robot to create a mess as long as it cleans up after itself.

Why is reinforcement learning hard?

- What actions are responsible for a reward may have occurred a long time before the reward was received.
  - The dog is expected to determine that eating the shoe at the start of the day is what was resposible for it being scolded at the end of the day.
- The long-term effect of an action depend on what the agent will do in the future.
  - It might be okay for a robot to create a mess as long as it cleans up after itself.
- The explore-exploit dilemma: at each time should the agent be greedy or inquisitive?

Reinforcement learning: main approaches

- search through a space of policies (controllers)

Reinforcement learning: main approaches

- search through a space of policies (controllers)
- learn a model consisting of state transition function $P(s'|a, s)$ and reward function $R(s, a)$; solve this an an MDP.

Reinforcement learning: main approaches

- search through a space of policies (controllers)
- learn a model consisting of state transition function $P(s'|a, s)$ and reward function $R(s, a)$; solve this an an MDP.
- learn $Q^*(s, a)$, use this to guide action.

Batch Learning

Assume that there is a sequence of transition samples $(s, a, r, s')_{1:N}$. A transition sample is also called *experience*. Clearly, we can use a sequence of experiences to create an empirical model:

$$\hat{\mathcal{S}} := \{s_i\} \cup \{s'_i\} \tag{1}$$

$$\hat{\mathcal{A}} := \{a_i\} \tag{2}$$

$$\hat{\mathcal{P}}^a_{ss'} := \frac{N^a_{ss'}}{N^a_s} \tag{3}$$

$$\hat{\mathcal{R}}^a_{ss'} := \frac{1}{N^a_{ss'}} \sum_{i=1}^{N} r_i \left[s_i = s \wedge a_i = a \wedge s'_i = s'\right] \tag{4}$$

$$\text{where } N^a_{ss'} := \sum_{i=1}^{N} [s_i = s \wedge a_i = a \wedge s'_i = s'] \tag{5}$$

$$N^a_s := \sum_{s' \in \hat{\mathcal{S}}} N^a_{ss'} \tag{6}$$

We can then use this empirical model for value or policy iteration.

Batch Learning

- The samples $(s, a, r, s')_{1:N}$ can be generated by an arbitrary policy, as long as this policy guarantees that, as $N \to \infty$, every action will be tried infinitely often in any state.
- Note that for trivial problems where transition model and reward structure are deterministic, only one sample is required for each combination of a and s (each sample representing an edge in the transition graph), and value iteration becomes the Bellman-Ford algorithm.

Q Learning

Consider the Bellman optimality equation for $Q^\star$:

$$Q^\star(s,a) = \sum_{s'} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma \max_{a'} Q^\star(s',a') \right)$$

Now consider an experience $(s, a, r, \tilde{s}')$. Using this tuple, we can first build very coarse approximations of $\mathcal{P}^a_{ss'} = p(s' \mid s, a)$ and $\mathcal{R}^a_{ss'}$ by defining

$$\mathcal{P}^a_{ss'} \approx [s' = \tilde{s}'] \tag{9}$$

$$\mathcal{R}^a_{s\tilde{s}'} \approx r \tag{10}$$

entering this into the equations above gives

$$Q^\star(s,a) \approx \sum_{s'} [s' = \tilde{s}'] \big(r + \gamma \max_{a'} Q^\star(s',a') \big) \tag{11}$$

Q Learning

$$Q^\star(s,a) \approx \sum_{s'} [s' = \tilde{s}'] \big(r + \gamma \max_{a'} Q^\star(s', a')\big)$$
$$= r + \gamma \max_{a'} Q^\star(\tilde{s}', a') \tag{12}$$

Where this quantity is sometimes known as *target*.
Due to approximations, this target value may contain errors. But imagine, $Q_{i-1}(s,a)$ is a previous approximation. From this approximation and the target created from experience (s,a,r,s') we can create an updated approximation by using a weighted average of both values:

$$Q_{i+1}(s,a) \leftarrow (1-\alpha) Q_i(s,a) + \alpha \big(r + \gamma \max_{a'} Q_i(s', a')\big) \tag{13}$$

Q Learning

Definition 1 (Q Learning)

$$Q_{i+1}(s,a) \leftarrow (1-\alpha)Q_i(s,a) + \alpha\big(r + \gamma \max_{a'} Q_i(s',a')\big) \tag{14}$$

Sometimes, this is written as

$$Q_{i+1}(s,a) \leftarrow Q_i(s,a) + \alpha\big(r + \gamma \max_{a'} Q_i(s',a') - Q_i(s,a)\big) \tag{15}$$

This version emphasizes the idea that the new action value is the old value plus a move of step size α in the direction of the *difference* between the *old* value and the target created by the *new experience* (i.e., we move the action value gradually to the target using the temporal difference).

Q Learning

Definition 2 (Learning rate)

In equations (14) and (15), the value α, where $0 \leq \alpha \leq 1$ is the *learning rate*.

Proposition 3 (Convergence of Q learning)

If α is appropriately adjusted over time such that $\lim_{t \to \infty} \alpha = 0$, the Q learning algorithm will converge to $Q^\star$.

Note 4

Finding the right adjustment schedule is problem dependent. Fortunately, $\pi^\star$ will (usually) be discovered long before $Q^\star$ has been reached.

Action Selection

- Q-learning has the interesting property that it will converge to the true optimal policy (and the true optimal action value function), regardless of policy, as long as the policy will try every action infinitely often in any state.
- However, in order to optimize reward, it seems advisable to start using actions with high value as soon as possible. This is know as the *exploration–exploitation dilemma*: if one starts to greedily use the currently optimal moves, action options leading to maybe even better results will never be explored.

Epsilon-greedy Action Selection

One simple solution is to *always* reserve a probability of ϵ (e.g., $\epsilon = 0.01$) for experimentally trying a non-optimal action. $\pi(s, a)$ can in this case be defined by:

$$\pi(s,a) = \begin{cases} 1-\epsilon & \text{if } a \text{ optimal} \\ \epsilon/(n-1) & \text{otherwise} \end{cases} \tag{16}$$

Where $n > 1$ is the number of actions available in state s (if $n = 1$ there is obviously nothing to explore).

Boltzmann action selection

A more involved action selection strategy uses the definition

$$\pi(s,a) = \frac{\exp\left(\lambda Q(s,a)\right)}{\sum_{a'} \exp\left(\lambda Q(s,a')\right)} \tag{17}$$

$\lambda > 0$ (alternatively: $1/\tau$, where τ is known as "temperature") determines the influence of the Q values on action selection.

- If λ is small (resp. if the temperature τ is high), all actions are almost equiprobable.
- If λ is large (low temperature), small differences in Q will create large differences in probability (preferring actions with larger Q values).

The probability in (17) is known as Gibbs or Boltzmann distribution; this is an example of a so called *softmax* action selection strategy.

Problems with Q-learning

- It does one backup between each experience.
  - Is this appropriate for a robot interacting with the real world?
  - An agent can make better use of the data by
    — remember previous experiences and use these to update model (action replay)
    — building a model, and using MDP methods to determine optimal policy.
    — doing multi-step backups
- It learns separately for each state.

Other RL Algorithms

- Explicitly maintaining the Q-Function as a table becomes infeasible for problems with large (or continuous) state or action spaces
  - Can use a discretization of states and/or actions
  - Or, use a function approximator for Q, e.g. a deep neural network ("deep Q learning")
- Q-Learning is just one, simple example of reinforcement learning algorithms, other algorithms differ w.r.t.
  - Number of steps taken before updating Q
  - Policy used for generating experiences ("on-policy" vs. "off-policy" RL)
  - Learning Q vs. directly learning π
  - ...

Summary

- RL algorithms allow an agent to learn taking actions in an unknown environment by learning from experience
- Challenging due to large state and action spaces, sparse rewards
- Active and exciting field of research
- We regularly offer team projects and master theses on applying RL to solve real-world problems