15-780 – Reinforcement Learning

J. Zico Kolter

March 26, 2014

Outline

Review of MDPs, challenges for RL

Model-based methods

Model-free methods

Exploration and exploitation

Outline

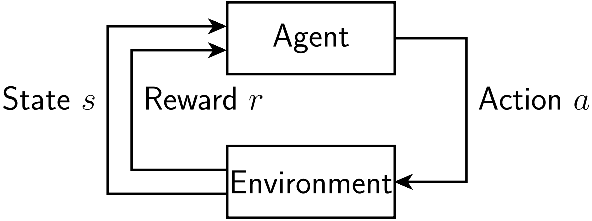
Review of MDPs, challenges for RL

Model-based methods

Model-free methods

Exploration and exploitation

Agent interaction with environment



Markov decision processes

- Recall a (discounted) Markov decision process is defined by:

$$\mathcal{M} = (S, A, T, R)$$

  - S: set of states
  - A: set of actions
  - $T : S \times A \times S \to [0, 1]$: transition distribution,$T(s, a, s')$ is probability of transitions to state s' after taking action a from state s
  - $R : S \to \mathbb{R}$: reward function, where $R(s)$ is reward for state s

- The RL twist: we don't know T or R, or they are too big to enumerate (only have the ability to act in MDP, observe states and rewards)

- Policy $\pi : S \rightarrow A$ is a mapping from states to actions

- Determine value of policy (policy evaluation)

$$
\begin{aligned}
V^{\pi}(s) &= \mathbf{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) | s_0 = s\right] \\
&= R(s) + \gamma \sum_{s' \in S} T(s, \pi(s), s') V^{\pi}(s')
\end{aligned}
$$

accomplished via iteration

$$
\forall s \in S,\ \hat{V}^{\pi}(s) \leftarrow R(s) + \gamma \sum_{s' \in S} T(s, \pi(s), s') \hat{V}^{\pi}(s')
$$

(or just solving linear systems)

- Determine value of optimal policy

$$V^\star(s) = R(s) + \gamma \sum_{s' \in S} T(s, \pi(s), s') V^\star(s')$$

accomplished via value iteration

$$\forall s \in S,\ \hat{V}^\pi(s) \leftarrow R(s) + \max_a \gamma \sum_{s' \in S} T(s, a, s') \hat{V}^\star(s')$$

(optimal policy is then $\pi^\star(s) = \max_a \gamma \sum_{s' \in S} T(s, a, s') \hat{V}^\star(s')$)

- How can we compute these quantities when T and R are unknown?

Outline

Review of MDPs, challenges for RL

Model-based methods

Model-free methods

Exploration and exploitation

Model-based RL

- A simple approach: just learn the MDP from data

- Given samples $(s_i, r_i, a_i, s'_i), i = 1, \ldots, m$ (could be from a single chain of experience)

$$\hat{T}(s, a, s') = \frac{\sum_{i=1}^{m} \mathbf{1}\{s_i = s, a_i = a, s'_i = s'\}}{\sum_{i=1}^{m} \mathbf{1}\{s_i = s, a_i = a\}}$$

$$\hat{R}(s) = \frac{\sum_{i=1}^{m} \mathbf{1}\{s_i = s\} r_i}{\sum_{i=1}^{m} \mathbf{1}\{s_i = s\}}$$

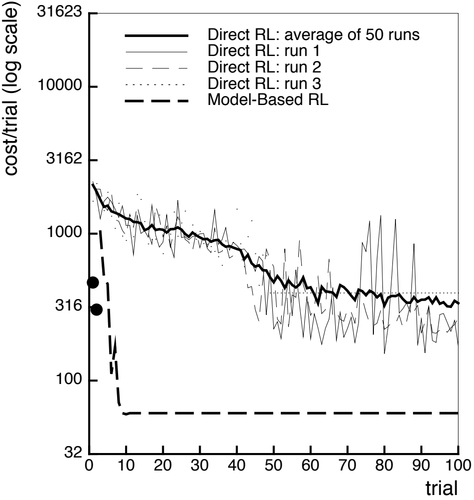
- Now solve the MDP $(S, A, \hat{T}, \hat{R})$

- Will converge to correct MDP (and hence correct value function / policy) given enough samples of each state

- How can we ensure we get the "right" samples? (a challenging problem for all methods we present here, stay tuned)

- Advantages (informally): makes "efficient" use of data, each

- Disadvantages: requires we build the the actual MDP models, not much help if state space is too large



(Atkeson and Santamaría, 96)

Outline

Review of MDPs, challenges for RL

Model-based methods

Model-free methods

Exploration and exploitation

Model-free RL

- Temporal difference methods (TD, SARSA, Q-learning): directly learn value function V^π or $V^\star$

- Direct policy search: directly learn optimal policy $\pi^\star$

Temporal difference (TD) methods

- TD algorithm is just a stochastic version of policy evaluation

> **algorithm** $\hat{V}^{\pi} = \text{TD}(\pi, \alpha, \gamma)$
>   // Estimate value function V^{π}
>   **initialize** $\hat{V}^{\pi}(s) \leftarrow 0$
>   **repeat**
>   Observe state s and reward r
>   Take action $a = \pi(s)$, and observe next state s'
> $\hat{V}^{\pi}(s) \leftarrow (1-\alpha)\hat{V}^{\pi}(s) + \alpha(r + \gamma \hat{V}^{\pi}(s'))$
>   return $\hat{V}^{\pi}$

- Will converge to $\hat{V}^{\pi}(s) \rightarrow V^{\pi}(s)$ (for all s visited frequently enough)

- TD lets us learn the value function of a policy π directly, *without* ever constructing the MDP

- But is this really that helpful?

- Consider trying to execute greedy policy w.r.t. estimated $\hat{V}^{\pi}$

$$\pi'(s) = \max_a \sum_{s'} T(s, a, s') \hat{V}^{\pi}(s')$$

we need a model anyway

SARSA and Q-learning

- Q functions are like value functions but defined over state-action pairs

$$Q^\pi(s,a) = R(s) + \sum_{s' \in S} T(s,a,s') Q(s', \pi(s'))$$

$$Q^\star(s,a) = R(s) + \sum_{s' \in S} T(s,a,s') \max_{a'} Q^\star(s',a')$$

- I.e., Q function is value of starting is state s, taking action a, and then acting according to π (or optimally, for $Q^\star$)

- Q function leads to new TD-like methods

- As with TD, observe state s, reward r, take action a (but not necessarily $a = \pi(s)$), observe next state s'

- SARSA: estimate $Q^\pi(s, a)$

$$\hat{Q}^\pi(s,a) \leftarrow (1-\alpha)\hat{Q}^\pi(s,a) + \alpha\left(r + \gamma \hat{Q}^\pi(s', \pi(s'))\right)$$

- Q-learning: estimate $Q^\star(s, a)$

$$\hat{Q}^\star(s,a) \leftarrow (1-\alpha)\hat{Q}^\star(s,a) + \alpha\left(r + \gamma \max_{a'} \hat{Q}^\star(s', a')\right)$$

- Again, these algorithms converge to true Q^π, $Q^\star$ if all state-action pairs seen frequently enough

- The advantage of this approach is that we can now select actions *without* a model of MDP

- SARSA, greedy policy w.r.t. $Q^\pi(s,a)$

$$\pi'(s) = \max_a \hat{Q}^\pi(s,a)$$

- Q-learning, optimal policy

$$\pi^\star(s) = \max_a \hat{Q}^\star(s,a)$$

- So with Q-learning, for instance, we can learn optimal policy without model of MDP

Function approximation

- Something is amiss here: we justified model-free RL approaches to avoid learning MDP, but we still need to keep track of value for each state

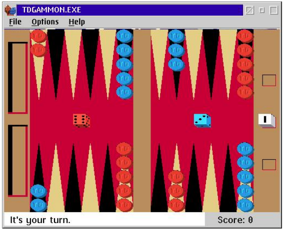
- A major advantage to model-free RL methods is that we can use *function approximation* to represent value function compactly

- Without going into derivations, let $\hat{V}^{\pi}(s) = f_{\theta}(s)$ denote function approximator parameterized by θ, TD update is

$$\theta \leftarrow \theta + \alpha(r + \gamma f_{\theta}(s') - f_{\theta}(s))\nabla_{\theta} f_{\theta}(s)$$

- Similar updates for SARSA, Q-learning

TD Gammon

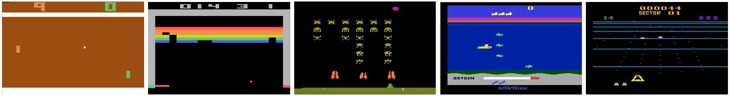


- Developed by Gerald Tesauro at IBM Watson in 1992

- Used TD w/ neural network as function approximator (known model, but much too large to solve as MDP)

- Achieved expert-level play, many world experts changed strategies based upon what AI found

Q-learning for Atari games



- Recent paper by Volodymyr Mnih et al., 2013 at DeepMind

- Q-learning with a deep neural network to learn to play games directly from pixel inputs

- DeepMind acquired by Google in Jan 2014

Direct policy search

- Rather that parameterizing Q function, and selecting $\pi(s) = \max_a Q(s,a)$, we could directly encode policy using a function approximator

$$\pi(s) = f_\theta(s)$$

- An optimization problem: find θ that maximize $V^\pi(s_0)$ for some initial state s_0

- A non-convex problem (even if we can compute it exactly), so we don't typically expect to find optimal policy

- Can't analytically compute gradients, so we need a way to approximately optimize this function only from samples

- A basic machine learning approach:

  1. Run M trials with perturbed parameters $\theta_1, \ldots, \theta_M$ and observe sum of rewards $J_1, \ldots, J_1$, where $J_i = \sum_{t=1}^{\infty} \gamma^t r_t$ when executing policy w/ parameters θ_i

  2. Learn model $J_i \approx g(\theta_i), \;\; \forall i = 1, \ldots, m$ using machine learning method

  3. Update parameters $\theta \leftarrow \theta + \alpha \nabla_\theta g(\theta)$

- This and more involved variants are surprisingly effective in many situations

Outline

Review of MDPs, challenges for RL

Model-based methods

Model-free methods

Exploration and exploitation

Exploration/exploitation problem

- All the methods discussed so far had some condition like "assuming we visit each state enough"

- A fundamental question: if we don't know the system dynamics, should we take exploratory actions that will give us more information, or exploit current knowledge to perform as best we can?

- Example: a model based procedure that does *not* work

  1. Use all past experience to build models $\hat{T}$ and $\hat{R}$ of MDP

  2. Find optimal policy for $(S, A, \hat{T}, \hat{R})$ using e.g. value iteration, act according to this policy

- Issue is that bad initial estimates in the first few cases can drive policy into sub-optimal region, and never explore further

- The procedure *does* work if we add an additional reward of $O(1/\sqrt{n(s,a)})$ to each state-action pair, where $n(s,a)$ denotes the number of times we have taken action a from state s.

  – But, this effectively take every action from every state in the MDP enough times: not a very practical solution

- A large outstanding issue for research: how can we perform guided exploration for large domains,

Take home points

- Reinforcement Learning lets us solve Markov decision problems, but in cases where we do not have a prior model of the system, or it is too large to allow computing an exact solution

- A number of possible approaches: model-based, value function model-free, policy search model-free, each with advantages/disadvantages

- Task of learning good model/value function/policy while simultaneously acting in the domain is still an open problem, except in extremely simple cases