Reinforcement Learning

6. Monte Carlo, Bias-Variance and Model-Based RL

Olivier Sigaud

Sorbonne Université
http://people.isir.upmc.fr/sigaud
Over-estimation bias

- In Q-LEARNING, due to the max operator, if some Q-value is over-estimated, this over-estimation propagates
- This is not the case of under-estimation
- Over-estimation propagation cannot be prevented due to Q-Table initialization
- Solution: using two Q-Tables, one for value estimation and one for value propagation

Reminder: TD error

- The goal of TD methods is to estimate the value function $V(s)$
- If estimations $V(s_t)$ and $V(s_{t+1})$ were exact, we would get $V(s_t) = r_t + \gamma V(s_{t+1})$
- $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$ measures the error between $V(s_t)$ and the value it should have given $r_t$
Monte Carlo (MC) methods

- Much used in games (Go...) to evaluate a state
- It uses the average estimation method $E_{k+1}(s) = E_k(s) + \alpha[r_{k+1} - E_k(s)]$
- Generate a lot of trajectories: $s_0, s_1, \ldots, s_N$ with observed rewards $r_0, r_1, \ldots, r_N$
- Update state values $V(s_k)$, $k = 0, \ldots, N - 1$ with:
  $$V(s_k) \leftarrow V(s_k) + \alpha(s_k)(r_k + r_{k+1} + \cdots + r_N - V(s_k))$$
TD vs MC

- Temporal Difference (TD) methods combine the properties of DP methods and Monte Carlo methods:
- In Monte Carlo, $T$ and $r$ are unknown, but the value update is global along full trajectories
- In DP, $T$ and $r$ are known, but the value update is local
- TD: as in DP, $V(s_t)$ is updated locally given an estimate of $V(s_{t+1})$ and $T$ and $r$ are unknown
- Note: Monte Carlo can be reformulated incrementally using the temporal difference $\delta_k$ update
Bias-variance compromise

- A bigger model may have more variance, and less bias
- Trajectories are a large model of value, a Q-Table is a smaller model.
Monte Carlo, One-step TD and N-step return

- MC suffers from variance due to exploration (+ stochastic trajectories)
- MC is on-policy → less sample efficient
- One-step TD suffers from bias
- N-step TD: tuning N to control the bias variance compromise
The N-step return in practice

▶ How do we store into the replay buffer?
▶ **N-step Q-LEARNING** is more efficient than **Q-LEARNING**


Eligibility traces

- To improve over Q-learning
- Naive approach: store all \((s, a)\) pair and back-propagate values
- Limited to finite horizon trajectories
- Speed/memory trade-off
- \(TD(\lambda), SARS(\lambda)\) and \(Q(\lambda)\): more sophisticated approach to deal with infinite horizon trajectories
- A variable \(e(s)\) is decayed with a factor \(\lambda\) after \(s\) was visited and reinitialized each time \(s\) is visited again
- \(TD(\lambda): V(s) \leftarrow V(s) + \alpha \delta e(s), (\text{similar for SARS}(\lambda)\text{ and } Q(\lambda)),\)
- If \(\lambda = 0\), \(e(s)\) goes to 0 immediately, thus we get \(TD(0), SARS\) or \(Q\)-LEARNING
- \(TD(1) = \text{Monte-Carlo...}\)
Model-based Reinforcement Learning

- General idea: planning with a learnt model of \( T \) and \( r \) is performing back-ups "in the agent’s head" ([Sutton, 1990, Sutton, 1991])
- Learning \( T \) and \( r \) is an incremental self-supervised learning problem
- Several approaches:
  - Draw random transition in the model and apply TD back-ups
  - Dyna-PI, Dyna-Q, Dyna-AC
  - Better propagation: Prioritized Sweeping

Dyna architecture and generalization

- Thanks to the model of transitions, Dyna can propagate values more often
- Problem: in the stochastic case, the model of transitions is in $\text{card}(S) \times \text{card}(S) \times \text{card}(A)$
- Usefulness of compact models
- MACS: Dyna with generalisation (Learning Classifier Systems)
- SPITI: Dyna with generalisation (Factored MDPs)


Any question?

Send mail to: Olivier.Sigaud@upmc.fr
Learning the Structure of Factored Markov Decision Processes in Reinforcement Learning Problems.

Combining latent learning with dynamic programming in MACS.

Prioritized sweeping: Reinforcement learning with less data and less real time.

High-dimensional continuous control using generalized advantage estimation.

Learning to mix n-step returns: Generalizing lambda-returns for deep reinforcement learning.

Integrating architectures for learning, planning, and reacting based on approximating dynamic programming.

DYNA, an integrated architecture for learning, planning and reacting.

Double q-learning.