Intro to RL: Notes

Aga Slowik

July 2020

1 Lecture 1

Link: TD-Gammon

Link: Summary of Chapter 2 from the RL book. The transition from multi-armed bandits to full RL through contextual bandits, ways of balancing exploration and exploitation: ϵ-greedy, UCB

Action-Value Methods: e.g. sample average $Q_t(a)$

The sample average converges to the optimal value for an action a if a has been taken an infinite number of times.

Standard form for the learning/update rules: $NewEstimate = OldEstimate + StepSize[Reward - OldEstimate]$

In a non-stationary env (non-stationary bandit): exponential, recency-weighted average

$Q_{n+1} = (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i$

$\alpha \in (0, 1]$

Q function - estimate value of (s, a) under the policy π at the timestep t. (future return starting at $t+1$) (*action-value function*)

V - *state-value function*

(V, Q - random variables)

We are interested in maximizing expected (future) return starting from the timestep t.

In episodic task, this is usually a sum of the future rewards in an episode.

In continuing tasks (no natural episodes), use a discount factor $\gamma \in [0, 1]$ (typically $\gamma = 0.9$ which means we are rather farsighted).

Mean Square Value Error in RL ($\mu(s)$ - the fraction of timesteps spent in the state s/distribution). Similar to regression but: the IID input assumption does not work (returns and inputs are correlated as they lie on the same trajectory). Gradient Monte Carlo algorithm. State aggregation.

2 Lecture 2

Markov property:

$P(S_{t+1}|S_t, A_t) = P(S_{t+1}|S_1, A_1, S_2, A_2 ... S_t, A_t)$

A finite discrete-time MDP $< S, A, R, P, \gamma >$

One-step models of the environment:

one-step state transition probabilities $p(s'|s,a) = P$

one-step expected reward $r(s,a) = R$

value function $v_\pi(s)$

values (state-value and action-value) can be written in terms of the successor values: Bellman equations

The optimal (state/action) value function: the maximum value function over all policies

Any policy that is greedy with respect to the optimal state-value function (v_*) is an optimal policy

Bellman optimality equations

Use Bellman equations in update rules

Iterative policy evaluation

Policy iteration: finite number of steps, no local optima

General policy iteration: 1. estimate value function 2. generate better policy

From policy-evalutation to value-iteration

Dynamic programming: we know the model. The number of states grows exponentially with the number of state variables (also branching factor etc).

Key challenges in RL:

To solve large problems, we need to approximate the iterations (sampling, ..) and generalize the value function to unseen states.

Solutions:

Online learning: adjust the value function and policy based on experience

Monte Carlo: sample a trajectory instead of expectation over all trajectories

But in MC we need to terminate (episodic environment).

Temporal-difference learning: look at one (or more) timesteps instead of the entire rollout. Estimate the reward (in TD, this will be a biased estimate).

MC: high variance, no bias

TD: low variance, some bias

every-visit MC?

MC coverges to the solutin with MSE

TD(0) - max likelihood Markovian model

Bootstrapping: update involves an estimate (DP, TD)

Sampling: update samples an expectation (MC, TD)

Unified view of RL (check it out).

On-policy and off-policy (check it out).

3 Lecture 3

Notes after the meeting -

2-player case?

Comparison of RL and DP