COMP90054 – Week 9 tutorial

Last updated: 1 May 2023

Multi-armed Bandits (MABs)

Definition

- A stochastic multi-armed bandit is a set of random variables $X_{i,t}$ where each variable represents a reward from playing arm i at time t
    - $i \in \{1, \cdots, N\}$ is the arm of the bandit
    - $k \in \{1, \cdots, K\}$ represents the index of play
- Successive plays are assumed to be independent from each other
- Each arm provides a reward drawn from an unknown probability distribution
- MAB in RL Context: At state s, the agent has a range of "arms" (a set of actions $A(s)$) that it can choose from, but the reward it receives from playing an arm is uncertain

MAB Strategies

- Aim: To learn $\pi(k)$, i.e. learn which arm to play at the k-th play to maximise the sum of the rewards collected over all rounds
- Problem: want to only play good actions, but don't know which ones are good
- Aim: To learn $\pi(k)$, i.e. learn which arm to play at the k-th play to maximise the sum of the rewards collected over all rounds
- Problem: want to only play good actions, but don't know which ones are good
- Intuition: find balance between
    - exploitation: play the arms deemed as the best so far
    - exploration: play other arms to potentially identify better arm

Ideas of some strategies

- Epsilon-greedy: Explore with probability of ϵ, exploit with probability of $1 - \epsilon$
- Epsilon-decreasing: Explore less often as we play more rounds (ϵ decays)
    - Softmax: Probability of choosing a depends on the empirical expected reward of a, can be controlled by the temperature parameter τ
- Upper Confidence Bounds (UCB1): An asymptotically optimal approach that considers both the empirical mean reward of each arm and the uncertainty about the true mean reward

Temporal difference reinforcement learning

Model-free reinforcement learning

- Idea: To learn a *policy* from the simulated *episodes* of the MDP problem
  - e.g., learn $Q(s, a)$ for all state-action pairs, then extract policy
- During each *episode*, starting from initial state s_0, a simulator generates a sequence of actions, rewards, and subsequent states.
  - At each state s, an action a is chosen, and the reward r and the new state s' are observed
  - Model-free RL techniques *reinforce* the Q estimates of applying a in s with the new observation
- Terminate the process if:
  - we run out of training time
  - we think our policy has converged to the optimal policy (for each new episode we see no improvement)
  - our policy is 'good enough' (for each new episode we see minimal improvement)

Model-based RL vs Model-free RL

- Similarity: both work with MDPs
- Difference: whether they have/use knowledge on $P_a(s'|s)$ and $r(s, a, s')$
  - Model-based RL uses this knowledge, e.g. value iteration
  - Model-free RL does not, and they rely on a simulator and learn through experience
- Given s and a, the simulator provides observations of s' according to $P_a(s'|s)$ and $r(s, a, s')$
- Model-free RL can use (but doesn't have to) use MAB strategies to select actions

Temporal difference (TD) methods

- TD methods use bootstrapping to alleviate the high variance issue in learning $Q(s, a)$ via Monte-Carlo RL
- Bootstrapping: update value function with agent's estimates rather than waiting for actual discounted rewards to be realised
- TD methods update $Q(s, a)$ by
- $$Q(s,a) \leftarrow \underbrace{Q(s,a)}_{\text{old value}} + \overbrace{\alpha}^{\text{learning rate}} \cdot [\underbrace{\overbrace{r}^{\text{reward}} + \overbrace{\gamma}^{\text{discount factor}} \cdot V(s')}_{\text{TD target}} - \overbrace{Q(s,a)}^{\text{do not count extra } Q(s,a)}]$$
  - New value of $Q(s, a)$ is a weighted average between TD target and the old value weighted by learning rate α
- Q-learning and SARSA are two examples of TD methods

Q-learning

- Idea: Estimate $V(s')$ with $\max_{a' \in A(s')} Q(s', a')$
    - Execute a, Observe reward r and new state s'
    - $Q(s,a) \leftarrow Q(s,a) + \alpha \cdot \left[r + \gamma \cdot \max_{a' \in A(s')} Q(s', a') - Q(s,a)\right]$

SARSA

- State-Action-Reward-State-Action
    - $s - a - r - s' - a'$
- Idea: Estimate $V(s')$ with the actual $Q(s', a')$ obtained in simulation
    - Execute a, observe reward r and new state s'
    - Select a' to apply in s' (thereby determining both state and action for next step)
    - $Q(s,a) \leftarrow Q(s,a) + \alpha \cdot [r + \gamma \cdot Q(s', a') - Q(s,a)]$

Comparing Q-learning and SARSA

- Q-learning is off-policy
    - it uses the greedy reward estimate for next state in its update rather than following the policy
    - it will converge to the optimal policy (assuming that all state-action pairs are sampled infinitely often), irrespective of the policy followed
- SARSA is on-policy
    - it chooses its action using the same policy used to choose the previous action, and then uses this difference to update its Q-function
    - its learning is dependent on the policy used