Literature review: Reinforcement Learning

Daniel Hernandez

0.1 Markov Decision Processes

subfiles

Bellman (1957) introduced the concept of a Markov Decision Process (MDP) as an extension of the famous idea of Markov chains. Markov decision processes are a standard model for sequential decision making and control problems. An MDP is fully defined by the 5-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}(\cdot|\cdot,\cdot), \mathcal{R}(\cdot,\cdot), \gamma)$. Whereby:

- $\mathcal{S}$ is the set of states $s \in \mathcal{S}$.
- $\mathcal{A}$ is the set of actions $a \in \mathcal{A}$.
- $\mathcal{P}(s'|s,a)$ where $s, s' \in \mathcal{S}$, $a \in \mathcal{A}$ is a transition kernel which states the probability of transitioning to state s' from state s after performig action a. $\mathcal{P} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$. If the environment is stochastic, as opposed to deterministic, the function $\mathcal{P}$ maps a state-action pair to a distribution over states in $\mathcal{S}$.
- $\mathcal{R}(s,a)$ where $s \in \mathcal{S}$, $a \in \mathcal{A}$; is the reward function, which returns the immediate reward (typically in the range $[-1, 1]$) of performing action a in state s. $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{R}$. The reward at time step t can be interchangably written as r_t or $r(s_t, a_t)$.
- $\gamma \in [0, 1]$ is the discount factor, which represent the difference in importance between the current reward and future rewards. This is often used as a variance reduction method, and aids proofs in infinitely running environments. (links?)

From here we can introduce the notion of an agent. (link to control theory?), An agent is an entity that on every state $s_t \in \mathcal{S}$ it can take an action $a_t \in \mathcal{A}$ in an environment transforming the environment from s_t to s_{t+1}. The beheviour of an agent is fully defined by a policy π. A policy π is a mapping from states to actions, $\pi : \mathcal{S} \rightarrow \mathcal{A}$. The agent chooses which action a_t to take in every state s_t by querying its policy such that $a_t = \pi(s_t)$. If the policy is stochastic, π will map an action to a distribution over action $a_t \sim \pi(s_t)$. The objective for an agent is to find an *optimal* policy, which tries to maximize the cumulative sum of possibly discounted rewards.

There are two functions of special relevance in reinforcement learning, the *state value* function $V^\pi(s)$ and the *action value* function $Q^\pi(s,a)$:

- The state value function $V^\pi(s)$ under a policy π, where $s \in \mathcal{S}$, represents the expected sum of rewards obtained by starting in state s and following the policy π until termination. Formally defined as $V^\pi(s) = \mathbb{E}^\pi[\sum_{t=0}^{\infty} r(s_t, a_t)|s_0 = s]$
- The state-action value function $Q^\pi(s,a)$ under a policy π, where $s \in \mathcal{S}$, $a \in \mathcal{A}$, represents the expected sum of rewards obtained by performing action a in state s and then following policy π. Formally defined as: $Q^\pi(s,a) = \mathbb{E}^\pi[r(s_0, a_0) + \sum_{t=1}^{\infty} r(s_t, a_t)|s_0 = s, a_0 = a]$

The Bellman equations are the most straight forward, dynamic programming approach at solving MDPs (Bertsekas, 2007; Bellman, 1957).

0.2 Bellman equations and optimality principle

Note that in general it is not the case that all actions $a \in \mathcal{A}$ can be taken on every state $s_t \in \mathcal{S}$.

The optimality principle, found in Bellman (1957), states the following: An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision. The optimality principle, coupled with the proof of the existance of a deterministic optimal policy for any MDP as outlined in (Borkar, 1988) give rise to the optimal state value function $V^*(s) = \text{argmax}_\pi V^\pi(s) = V^{\pi^*}(s)$ and the optimal action value function $Q^*(s,a) = \text{argmax}_\pi Q^\pi(s,a) = Q^{\pi^*}(s,a)$. The optimal value functions determine the best possible performance in a MDP. An MDP is considered *solved* once the optimal value functions are found.

Most of the field of reinforcement learning research focuses on approximating these two equations (Tamar et al., 2017) (Watkins and Dayan, 1992) (Mnih et al., 2013). (cite many more)

Bellman (1957) outlined two analytical equations for the state value and action value function:

$$V^\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) * (r(s,a) + \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a) * V^\pi(s')) \tag{1}$$

$$Q^\pi(s,a) = r(s,a) + \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a) * (\sum_{a' \in \mathcal{A}} \pi(a'|s') Q^\pi(s', a')) \tag{2}$$

Most RL algorithms can be devided into the following categories: Policy based Value based actor critic

A further categorization of algorithms is the notion of *model free* and *model based* algorithms. Consider a *model* of an environment to be the transition function $\mathcal{P}$ and reward function $\mathcal{R}$. Model free algorithms aim to approximate an optimal policy without them. Model based algorithms are either given a prior model that they can use for planning (Browne et al., 2012; Soemers, 2014), or they learn a representation via their own interaction with the environment (Sutton, 1991; Guzdial et al., 2017). Note that an advantage of learning your own model is that you can choose a representation of the environment that is relevant to the agent's actions, which can have the advantage of modelling uninteresting (but perhaps complicated) environment behaviour (Pathak et al., 2017).

0.3 Q-learning

subfiles

The Q-learning algorithm was first introduced by Watkins (1989), and is arguably one of the most famous and widely implemented methods in the entire field. Given an MDP, Q-learning aims to calculate the corresponding optimal action value function Q^*, hence the name. From there, a deterministic greedy policy can be calculated via $\pi(s) = \text{argmax}_{a \in \mathcal{A}_s} Q(s, a)$. Q-learning has been proven to converge to the optimal solution for an MDP under some assumptions. These assumptions being that the MDP is episodic, meaning that a terminal state is eventually reached, and that each state-action pair is visited an infinite number of times. Another necessary condition for convergence is that the sequence of updates of Q-values has to be monotonically increasing $Q(s_i, a_i) \leq Q(s_{i+1}, a_{i+1})$.

Q-learning is not a perfect algorithm, Watkins and Dayan (1992) tells us that the algorithm converges to optimality with probability 1 if each state-action pair is represented discretely. If there is a function approximator[1] in place, Thrun and Schwartz (1993) shows that if the approximation error is greater than a threshold which depends on the discount factor γ and episode length, then a systematic overestimation effect, which happens mainly due to the joint effort of function approximation methods and the max operator. There has been some research focused on overcoming this overestimation (Kaisers and Tuyls, 2010). Most notably, Abdallah et al. (2016)

References

Abdallah, S., Org, S., Kaisers, M., and Nl, K. (2016). Addressing Environment Non-Stationarity by Repeating Q-learning Updates. *Journal of Machine Learning Research*, 17:1–31.

Bellman, R. (1957). *Dynamic Programming*, volume 70.

Bertsekas, D. P. (2007). *Dynamic Programming and Optimal Control, Vol. II*, volume 2.

[1] With neural networks being the most famous function approximators in reinforcement learning at the time of writing.

Borkar, V. S. (1988). A convex analytic approach to Markov decision processes. *Probability Theory and Related Fields*, 78(4):583–602.

Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., and Colton, S. (2012). A survey of {Monte Carlo Tree Search} methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43.

Guzdial, M., Li, B., and Riedl, M. O. (2017). Game Engine Learning from Video. *International Conference on Artificial Intelligence (IJCAI).*

Kaisers, M. and Tuyls, K. (2010). Frequency Adjusted Multi-agent Q-learning. *Learning*, pages 309–316.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing Atari with Deep Reinforcement Learning. pages 1–9.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. (2017). Curiosity-Driven Exploration by Self-Supervised Prediction. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2017-July:488–489.

Soemers, D. (2014). Tactical Planning Using MCTS in the Game of StarCraft. page 12.

Sutton, R. S. (1991). Dyna, an integrated architecture for learning, planning, and reacting. *ACM SIGART Bulletin*, 2(4):160–163.

Tamar, A., Wu, Y., Thomas, G., Levine, S., and Abbeel, P. (2017). Value iteration networks. *IJCAI International Joint Conference on Artificial Intelligence*, (Nips):4949–4953.

Thrun, S. and Schwartz, A. (1993). Issues in Using Function Approximation for Reinforcement Learning. *Proceedings of the 4th Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum*, pages 1–9.

Watkins, C. J. (1989). *Learning from delayed rewards*. PhD thesis, King's College.

Watkins, C. J. and Dayan, P. (1992). Technical Note: Q-Learning. *Machine Learning*, 8(3):279–292.