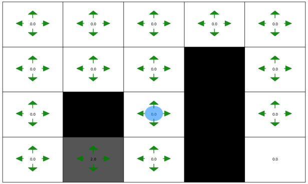
Reinforcement Learning

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

Olivier Sigaud

Sorbonne Université
http://people.isir.upmc.fr/sigaud

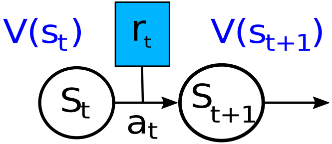
Over-estimation bias



- 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

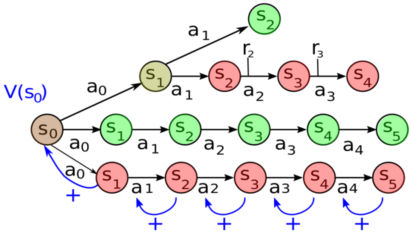
Van Hasselt, H. (2010) Double q-learning. *Advances in Neural Information Processing Systems*, pages 2613–2621

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:

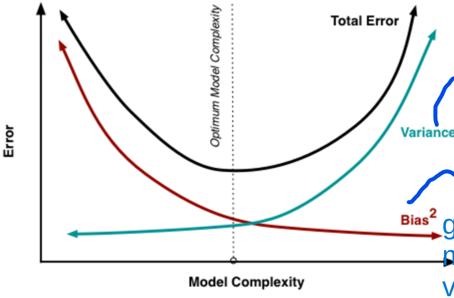
$$V(s_k) \leftarrow V(s_k) + \alpha(s_k)(r_k + r_{k+1} + \cdots + r_N - V(s_k))$$

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 δ_k update

Bias-variance compromize

Schema prof pour explication

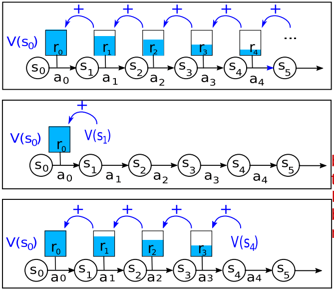


different estimates at different times

get an average over many stockhastic values => it's long

- 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



lot's of variance because of the random values

Monte Carlo

One-step TD

bias because we use the previous values of the value function to estimate the new value. If the estimate (1st one previous) is wrong because of the bias propagation then the next value will be wrong as well

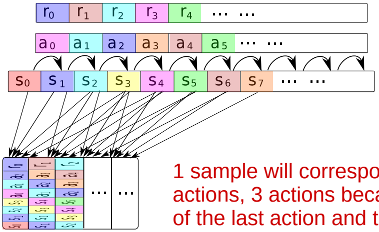
N-step TD

best solution in termes of variance and bias

- MC suffers from variance due to exploration (+ stochastic trajectories)
- MC is on-policy $\rightarrow$ less sample efficient
- One-step TD suffers from bias
- N-step TD: tuning N to control the bias variance compromize

The N-step return in practice

store samples in the replay buffer. The samples correspond to n consecutive steps in the environment



1 sample will correspond here to 4 consecutive actions, 3 actions because we need the next state of the last action and the reward that we get for the first state

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

Schulman, J., Moritz, P., Levine, S., Jordan, M., & Abbeel, P. (2015b) High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*

Sharma, S., Ramesh, S., Ravindran, B., et al. (2017) Learning to mix N-step returns: Generalizing λ-returns for deep reinforcement learning. *arXiv preprint arXiv:1705.07445*

Eligibility traces

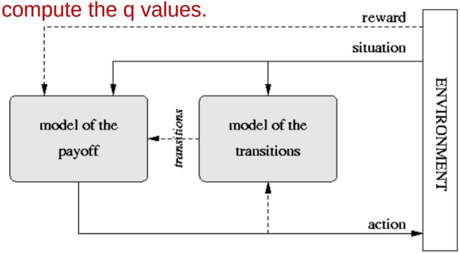
record the trajectory while moving the agent and once you find a source of reward we will propagate values backwards along the trajectory so that you will update the values of all the states that we have visited along that particular trajectory

- 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(λ), SARSA (λ) and Q(λ): more sophisticated approach to deal with infinite horizon trajectories
- A variable $e(s)$ is decayed with a factor λ after s was visited and reinitialized each time s is visited again
- TD(λ): $V(s) \leftarrow V(s) + \alpha\delta e(s)$, (similar for SARSA ($\lambda$) and Q($\lambda$)),
- If $\lambda = 0$, $e(s)$ goes to 0 immediately, thus we get TD(0), SARSA or Q-LEARNING
- TD(1) = Monte-Carlo...

lamba parameter allows us to propagate the values along states. The rest is very good explained in the bullet points

Model-based Reinforcement Learning

Dynamic programming => learning a model of the reward and transition function then propagating he reward to the model to compute the q values.

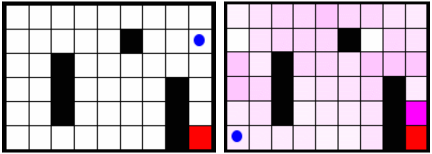


- General idea: planning with a learnt model of T and r is performing back-ups "in the agent's head" ([Sutton, 1990, Sutton, 1991])
- Learning T and r is an incremental self-supervised learning problem
- Several approaches:
  - Draw random transition in the model and apply TD back-ups
  - Dyna-PI, Dyna-Q,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

Moore, A. W. & Atkeson, C. (1993). Prioritized sweeping: Reinforcement learning with less data and less real time. *Machine Learning*, 13:103–130.

the agent(blue dot) needs time to explore the good path that he has to follow. With time he learns the path, even when he didn't find the reward.

Dyna architecture and generalization

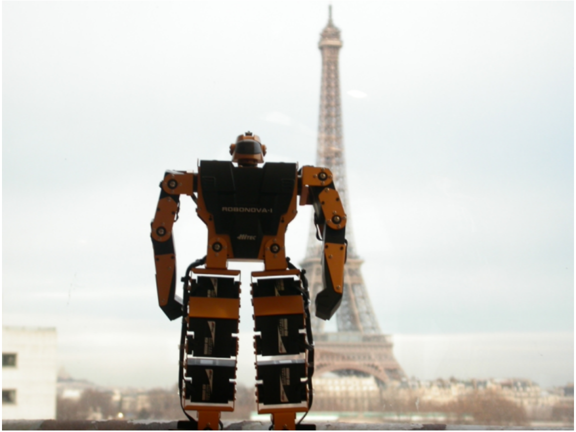


- Thanks to the model of transitions, Dyna can propagate values more often
- Problem: in the stochastic case, the model of transitions is in $card(S) \times card(S) \times card(A)$
- Usefulness of compact models
- MACS: Dyna with generalisation (Learning Classifier Systems)
- SPITI: Dyna with generalisation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

Gérard, P., Meyer, J.-A., & Sigaud, O. (2005) Combining latent learning with dynamic programming in MACS. *European Journal of Operational Research*, 160:614–637.

Degris, T., Sigaud, O., & Wuillemin, P.-H. (2006) Learning the Structure of Factored Markov Decision Processes in Reinforcement Learning Problems. *Proceedings of the 23rd International Conference on Machine Learning (ICML'2006)*, pages 257–264

Any question?



Send mail to: `Olivier.Sigaud@upmc.fr`

Degris, T., Sigaud, O., & Wuillemin, P.-H. (2006).
Learning the Structure of Factored Markov Decision Processes in Reinforcement Learning Problems.
Edité dans *Proceedings of the 23rd International Conference on Machine Learning*, pages 257–264, CMU, Pennsylvania.

Gérard, P., Meyer, J.-A., & Sigaud, O. (2005).
Combining latent learning with dynamic programming in MACS.
European Journal of Operational Research, 160:614–637.

Moore, A. W. & Atkeson, C. (1993).
Prioritized sweeping: Reinforcement learning with less data and less real time.
Machine Learning, 13:103–130.

Schulman, J., Moritz, P., Levine, S., Jordan, M. I., & Abbeel, P. (2015).
High-dimensional continuous control using generalized advantage estimation.
arXiv preprint arXiv:1506.02438.

Sharma, S., Ramesh, S., Ravindran, B., et al. (2017).
Learning to mix n-step returns: Generalizing lambda-returns for deep reinforcement learning.
arXiv preprint arXiv:1705.07445.

Sutton, R. S. (1990).
Integrating architectures for learning, planning, and reacting based on approximating dynamic programming.
Edité dans *Proceedings of the Seventh International Conference on Machine Learning*, pages 216–224, San Mateo, CA. Morgan Kaufmann.

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

Van Hasselt, H. (2010).
Double q-learning.
Edité dans *Advances in Neural Information Processing Systems*, pages 2613–2621.