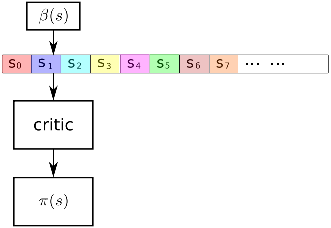
Reinforcement Learning

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

Olivier Sigaud

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

Introducing a replay buffer



- Helps decorrelating the agent trajectory and samples fed to the critic
- Samples can be fed to the critic randomly or through various heuristics
- Introduces sample efficiency discussion

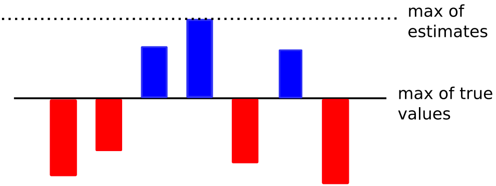
Replay buffer and sample efficiency

- Important intuition: in the discrete deterministic case, one sample from each (state, action) pair in the buffer is enough for Q-LEARNING to converge
- Thus using a replay buffer can be very sample efficient
- In the stochastic case, samples in the replay buffer should reflect the distribution over next state
- This may require a large replay buffer (over $1e^6$ samples)
- In the continuous case, the state (and action) spaces cannot be covered
- But off-policy deep RL algorithms using a replay buffer still benefit from the initial intuition

Maximization in RL

- Two maximization steps:
  - In action selection:
    $$\pi(s) \sim \operatorname*{argmax}_{a \in A} Q(s, a)$$
    might be stochastic or contain some exploration
  - In Q-LEARNING, in the value update rule
    $$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_{a \in A} Q(s_{t+1}, a) - Q(s_t, a_t)]$$

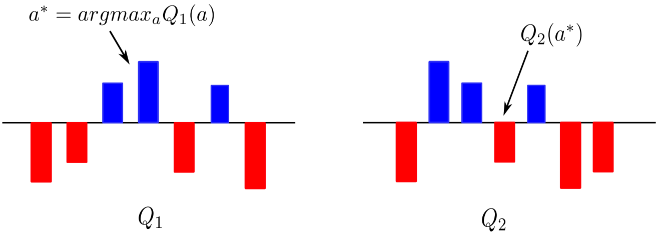
Maximization bias



- In action selection, a maximum over estimated $Q(s, a)$ is performed
- "In these algorithms, a maximum over estimated values is used implicitly as an estimate of the maximum value, which can lead to a significant positive bias."
- Example: imagine all true $Q(s, a)$ values are null

Sutton, R. S. & Barto, A. G. (2018) *Reinforcement Learning: An Introduction (Second edition)*. MIT Press

Double Q-LEARNING



- Solution: using two Q-Tables, one for value estimation and one for action selection
- $a^* = \text{argmax}_a\, Q_1(a)$
- $Q_2(a^*) = Q_2(\text{argmax}_a\, Q_1(a))$ unbiased estimate of $Q(a^*)$
- $a'^* = \text{argmax}_a\, Q_2(a)$
- $Q_1(a'^*) = Q_1(\text{argmax}_a\, Q_2(a))$ unbiased estimate of $Q(a'^*)$
- Randomly select one of each at all steps

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

Double Q-LEARNING: results

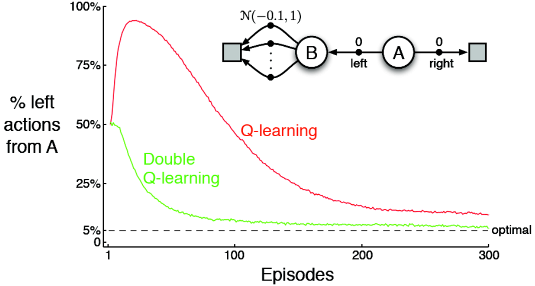
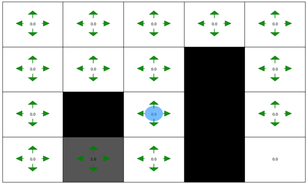


Figure 6.5: Comparison of Q-learning and Double Q-learning on a simple episodic MDP (shown inset). Q-learning initially learns to take the left action much more often than the right action, and always takes it significantly more often than the 5% minimum probability enforced by ε-greedy action selection with $\varepsilon = 0.1$. In contrast, Double Q-learning is essentially unaffected by maximization bias. These data are averaged over 10,000 runs. The initial action-value estimates were zero. Any ties in ε-greedy action selection were broken randomly.

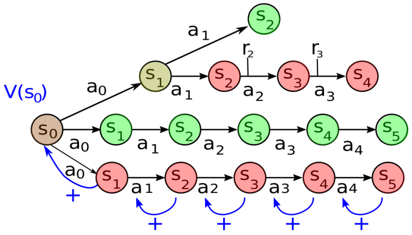
Over-estimation bias propagation



- Some initial bias cannot be prevented due to Q-Table initialization
- In Q-LEARNING, due to the max operator, the maximization bias propagates
- No propagation of under-estimation
- The same holds for DDPG without a max operator!

Fujimoto, S., van Hoof, H., & Meger, D. (2018) Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*

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

Eligibility traces

- Goal: 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...
- Eligibility traces can be seen as a combination of N-step returns for all N

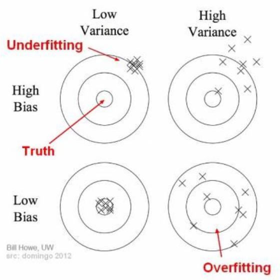
Schulman, J., Moritz, P., Levine, S., Jordan, M., & Abbeel, P. (2015b) High-dimensional continuous control using Generalized Advantage Estimation. *arXiv preprint arXiv:1506.02438*

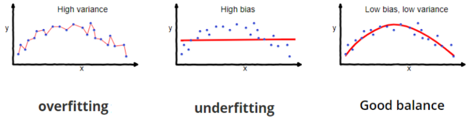
Bias versus variance

- If you compute an expectation over infinitely many samples, you get the same expectation each time
- But if you compute it over a finite set of samples, you get a different expectation each time
- This is known as variance
- Given a large variance, you need many samples to get an accurate estimate of the mean
- If you update an expectation based on a previous (wrong) expectation estimate, the expectation estimate you get provided infinitely many samples is wrong
- This is known as bias

Geman, S., Bienenstock, E., & Doursat, R. (1992) Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1–58

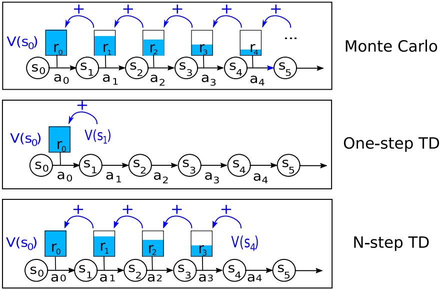
Bias, variance, overfitting and underfitting





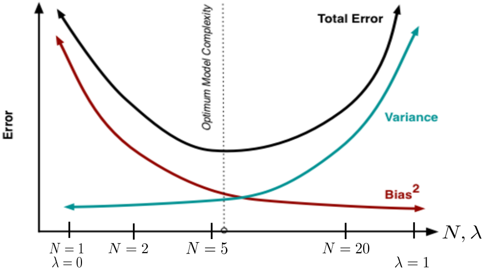
- With high bias, the risk is underfitting
- With high variance, the risk is overfitting
- You need low bias and low variance

Monte Carlo, One-step TD and N-step return



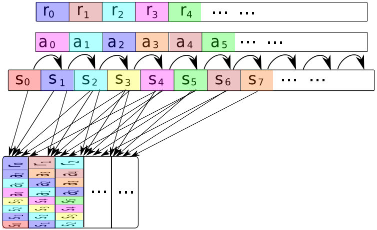
- One-step TD suffers from bias
- MC suffers from variance due to exploration (+ stochastic trajectories)
- MC is on-policy $\rightarrow$ less sample efficient
- N-step TD: tuning N to control the bias-variance compromize

Bias-variance compromize



- Total error = bias^2 + variance + irreducible error
- A more complex model (e.g. bigger network) generally has more variance, but less bias
- Tuning N in the N-step return or λ in an eligilibity trace method helps finding the right compromize.

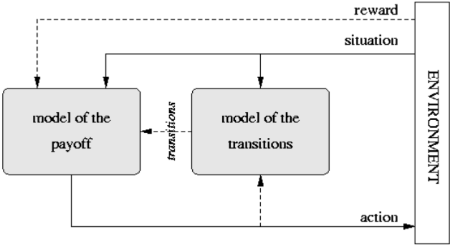
The N-step return in practice



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

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*

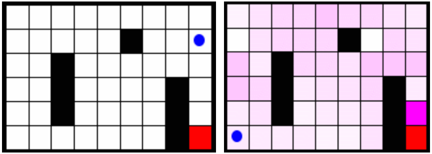
Model-based Reinforcement Learning



- 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

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

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)

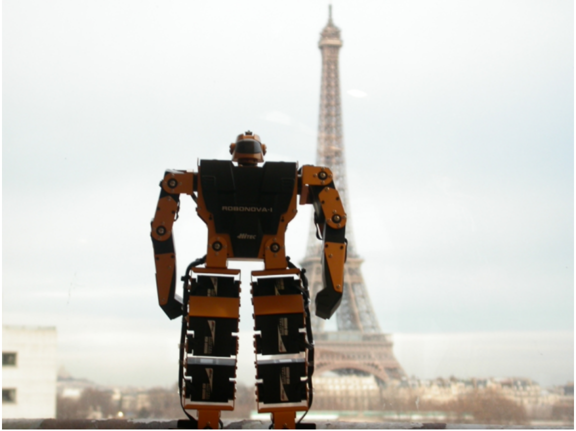
Gérard, P., Meyer, J.-A., & Sigaud, O. (2005) Combining latent learning with dynamic programming in MACS. *European Journal of Operational Research*, 160:614–637.

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

Corresponding labs

- See `https://github.com/osigaud/rl_labs_notebooks`
- One notebook about N-step return
- One notebook about model-based RL, based on RTDP

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.

Fujimoto, S., van Hoof, H., & Meger, D. (2018).
Addressing function approximation error in actor-critic methods.
arXiv preprint arXiv:1802.09477.

Geman, S., Bienenstock, E., & Doursat, R. (1992).
Neural networks and the bias/variance dilemma.
Neural computation, 4(1):1–58.

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.

Sutton, R. S. & Barto, A. G. (2018).
Reinforcement Learning: An Introduction (Second edition).
MIT Press.

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