Agent-Based Modeling and Simulation

Temporal Difference

Dr. Alejandro Guerra-Hernández

Instituto de Investigaciones en Inteligencia Artificial
Universidad Veracruzana
Campus Sur, Calle Paseo Lote II, Sección Segunda No 112,
Nuevo Xalapa, Xalapa, Ver., México 91097

mailto:aguerra@uv.mx
https://www.uv.mx/personal/aguerra/abms

Doctorado en Inteligencia Artificial 2024

Universidad Veracruzana

Credits

- These slides are based on the book of Sutton and Barto [1], chapter 6. Any difference with this source is my responsibility.
-
-

Universidad Veracruzana

Basic Ideas

- If one had to identify one idea as central and novel to RL, it would be undoubtedly be temporal difference (TD) learning.
- TD is a combination of Monte Carlo (MC) and Dynamic Programming (DP) ideas.
- Like MC methods, TD methods can also learn directly from raw experience without a model of the environment's dynamics.
- Like DP methods, TD methods updates estimates based in part on other learned estimates, without waiting for a final outcome (bootstrapping).
- Indeed these methods can be combined in many ways.

Universidad Veracruzana

Prediction and Control Problems

- For the control problem, *i.e.*, finding an optimal policy, DP, TD, and Monte Carlo methods all use some variation of generalized policy iteration (GPI).
- For the prediction problem, *i.e.*, the problem of estimating the value function v_π for a given policy π, they all differ.

Universidad Veracruzana

Constant-α MC

- Given some experience following the policy π, TD and MC update their estimates V of v_π for the nonterminal states S_t occurring in that experience.
- MC methods wait until the return following the visit is known, then use the return as a target for $V(s_t)$.
- A simple every-visit MC method suitable for nonstationary environments is:

$$V(S_t) \leftarrow V(S_t) + \alpha \Big[G_t - V(S_t) \Big] \tag{1}$$

where G_t is the actual return following time t; α is a constant step-size parameter.

Universidad Veracruzana

TD(0)

- Where as MC must wait until the end of the episode to determine the increment to $V(s_t)$, since only then G_t is known, TD methods need to wait only until next time step.
- At time $t+1$ they immediately form a target and make a useful update using the observed reward R_{t+1} and the estimate $V(S_{t+1})$:

$$V(S_t) \leftarrow V(S_t) + \alpha \left[R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \right] \tag{2}$$

- This method is also known as one-step TD.

Universidad Veracruzana

Tabular TD(0) for estimating v_π

Require: π ▷ the policy to be evaluated
Require: $\alpha \in (0, 1]$ ▷ step size
Ensure: $V(s)\ \forall s \in \mathcal{S}^+$ ▷ arbitrarily, except that $V(\mathit{terminal}) = 0$

loop for each episode
    $\mathit{init}(S)$
    loop for each step in the episode
        $A \leftarrow$ the action recommended by π for S.
        $R, S' \leftarrow \mathit{execute}(A)$
        $V(S) \leftarrow V(S) + \alpha \Big[R + \gamma V(S') - V(S)\Big]$
        $S \leftarrow S'$
    end loop
end loop

Universidad Veracruzana

Estimate

- While *MC* uses an estimate, sampling values of:

$$v_\pi \doteq \mathbb{E}_\pi[G_t \mid S_t = s] \tag{3}$$

- *DP* estimates instead:

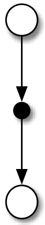
$$v_\pi \doteq \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s] \tag{4}$$

  by adopting the known $V(S_{t+1})$ instead of $v_\pi(S_{t+1})$.
- *TD*(0) is a combination of *MC* sampling with *DP* bootstrapping, *i.e.*, it uses both estimates.
- Remember both equations are equivalent.

Universidad Veracruzana

Backup Diagram

- The value estimate for the state node at the top is updated on the basis of the one sample transition from it to the immediate following state.
- *TD* and *MC* involves looking ahead to a sample successor state, instead of a complete distribution of all possible successors.
- The value of the successor and the reward along the way are used to compute a backed-up value to update the original state.



TD(0)

TD error

- Observe that the difference between the estimated value of S_t and the better estimate $R_{t+1} + \gamma V(S_{t+1})$ is a sort of error, called the TD error:

$$\delta_t \doteq R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \tag{5}$$

- Notice that δ_t is the error in the estimate made at that time, but requires information available one step later.
- If V does not change during the episode, as in MC methods, then:

$$G_{t+1} - V(S_t) = \sum_{k=t}^{T-1} \gamma^{k-t} \delta_k \tag{6}$$

- If this is not the case, as in $TD(0)$, small α values hold this identity approximately.

Universidad Veracruzana

Bootstrapping

- TD methods update their estimates based in part on other estimates. They learn guess from a guess –they bootstrap.
- What advantages do TD methods over MC and DP?
- Clue. The answer is in the rest of the book of Sutton and Barto [1].

Universidad Veracruzana

Advantages

- TD methods do not require a model of the environment, of its reward and next-state probability distributions; as DP methods do.
- TD methods are naturally implemented in an online, fully incremental fashion. They don't have to wait until the end of the episode to learn, as MC.
- Surprisingly often, this turns out to be a critical consideration. *e.g.*, when facing very long episodes or continuous tasks.
- TD methods learn from each transition regardless of what subsequent actions are taken, they are faster than some MC methods that ignore exploring actions.

Universidad Veracruzana

Soundness

- Certainly it is convenient to learn one guess from the next, without waiting for an actual outcome, but can we still guarantee convergence to the correct answer?
- Yes. For any fixed policy π, $TD(0)$ has been proved to converge to v_π, in the mean for a constant step-size parameter if it is sufficiently small, and with probability 1 if α decreases accordingly to the usual approximation conditions.
- Most convergence proofs apply only to the table-based case, but some also apply to the general linear function approximation.

Universidad Veracruzana

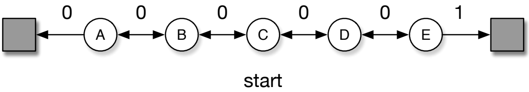
Speed

- If both TD and MC methods converge asymptotically to the correct predictions, the which gets there first?
- At the current time this is an open question in the sense that no one has been able to prove mathematically that one method converges faster than the other.
- In fact, it is not even clear what is the most appropriate formal way to phrase this question.
- In practice TD methods have usually been found to converge faster than constant-α MC methods on stochastic tasks.

Universidad Veracruzana

Example: Random Walk

- Consider the following Markov Reward Process (an MDP without actions):

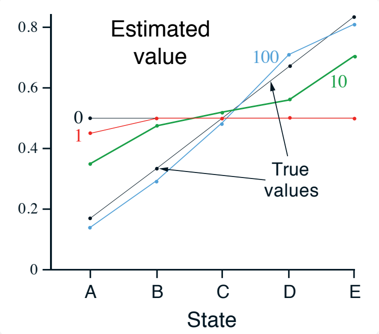


- All episodes start at C, then proceed either left or right by one state on each step, with equal probability.
- Example. C,0,B,0,C,0,D,0,E,1
- The true value of each state is the probability of terminating on the right if starting at that state, *e.g.*, $v_\pi(C) = 0.5$
- True value from A to E are $\frac{1}{6}, \frac{2}{6}, \frac{3}{6}, \frac{4}{6}$, and $\frac{5}{6}$

Universidad Veracruzana

Values Learned by TD(0), $\alpha = 0.1$

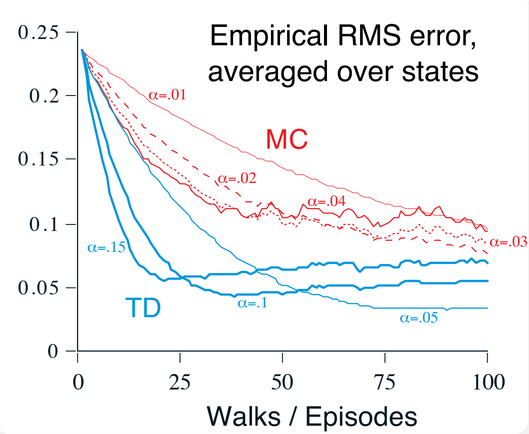
- The graph shows the values learned after various number of episodes on a single run of $TD(0)$.



- 100 episodes (blue) with $\alpha = 0.1$ are quite good.

Universidad Veracruzana

Empirical RMS error



Batch Updating

- Suppose there is available only a finite amount of experience, say 10 episodes or 100 time steps.
- A common approach with incremental learning is to present the experience repeatedly until the method converges upon an answer.
- Given an approximate value function V, the increments specified in eqs. 1 and 2 are computed for every time step t at which a nonterminal state is visited, but the value function is changed only once, by the sum of all the increments.
- Then all the available experience is processed again with the new value function to produce a new overall increment, an so on, until the value function converges.

Universidad Veracruzana

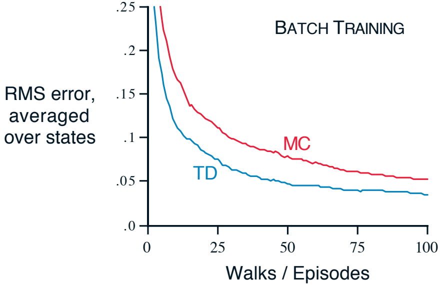
Convergence

- Under batch updating, $TD(0)$ converges deterministically to a single answer independent of the step-size parameter, as long as α is chosen to be sufficiently small.
- Constant-α MC method also converges deterministically under the same conditions, but to a different answer!.

Universidad Veracruzana

Example: Random Walk under Batch Updating

- After each new episode, all episodes seen so far were treated as a batch –they were repeatedly presented to the algorithm with an α sufficiently small to converge.



- MC is optimal only in a limited way, TD is optimal in a way that is more relevant to predict returns.

Universidad Veracruzana

Example: You are the Predictor

- Suppose you observe the following eight episodes:

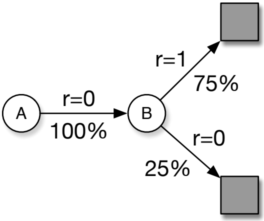
A,0,B,0	B,1
B,1	B,1
B,1	B,1
B,1	B,0

- Given this batch data, what would you say are the optimal predictions, the best values for estimates $V(A)$ and $V(B)$?
- Everybody would agree that the optimal value for $V(B)$ is $\frac{3}{4}$ because six out of eight times in the state B the process terminated immediately with return 1.
- What about A?

Universidad Veracruzana

Answer 1

- Observe that 100% of the times the process was in state A it traverses immediately to B (with a reward of 0); and
- Because we have already decided that B has value $\frac{3}{4}$, therefore A must have value $\frac{3}{4}$ as well.
- Modelling the MDP enables the computation of this value. This is the answer that batch $TD(0)$ also gives.



Answer 2

- We have seen A only once and the return that followed was 0; therefore $V(A) = 0$.
- This is the answer that batch MC methods give.
- Notice that it is also the answer that gives minimum square error on the trining data.
- If the process is Markov, we expect that answer 1 will produce lower error on future data; while answer 2 is better on existing data.

Universidad Veracruzana

Maximum Likelihood Estimate

- Batch MC methods always find the estimate that minimize the mean-squared error on the training set, whereas batch $TD(0)$ always find the estimate that would be exactly correct for the maximum likelihood model of the Markov process.
- The maximum likelihood estimate of a parameter is the parameter value whose probability of generating the data is greatest.
- Given the MDP model we can compute the estimate of the value function that would be exactly correct if the model were exactly correct.
- This is called the certainty-equivalence estimate. In general, batch $TD(0)$ converges to it.

Universidad Veracruzana

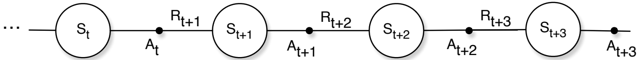
Certainty-Equivalence

- Although the certainty-equivalence estimate is in some sense an optimal solution, it is almost never feasible to compute it directly.
- If $n = |\mathcal{S}|$ is the number of states, then just forming the maximum-likelihood estimate may require the order of n^2 memory, and computing the corresponding function requires on the order of n^3 steps.
- It is striking that TD methods can approximate the same solution with no more than order n repeated computations over the training set.
- On tasks with large state spaces, TD methods may be the only feasible way of approximating this solution.

Universidad Veracruzana

Action-value Function

- The first step is to learn an action-value function rather than a state-value function.
- In particular, for an on-policy method we must estimate $q_\pi(s, a)$ for the current behavior policy π and for all states s and actions a.
- This can be donde using essentially the same *TD* method described before for learning v_π.
- Recall that an episode consists of an alternating sequence of states and state-action pairs:



Universidad Veracruzana

Transitions I

- Consider now transitions from state-action pair to state-action pair to learn the value of state-action-pairs.
- Formally both cases are identical, *i.e.*, they are both Markov chains with a reward process.
- The theorems assuring convergence of state values under $TD(0)$ also apply to the corresponding algorithm for action values:

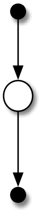
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t) \right] \quad (7)$$

- This update is done after every transition from a nonterminal state S_t.

Universidad Veracruzana

Transitions II

- If S_{t+1} is terminal, then $Q(S_{t+1}, A_{t+1})$ is defined as zero.
- This rule uses every element of the quintuple of events, $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$ (can you see the name of the method?).
- Backup diagram:



Sarsa

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Require: $\alpha \in (0, 1]$ ▷ the step size
Require: Small $\epsilon > 0$ ▷ probability of exploration
Ensure: Initialize $Q(s, a)\ \forall s \in \mathcal{S}^+, a \in \mathcal{A}$ ▷ arbitrarily except that $Q(terminal, .) = 0$.

```
loop for each episode
    Initialize S
    loop for each step of the episode
        Take action A, observe R, S′
        Choose A′ from S′ using policy derived from Q (e.g., ε-greedy).
        Q(S, A) ← Q(S, A) + α [R + γQ(S′, A′) − Q(S, A)]
        S ← S′; A ← A′;
    end loop until S in terminal
end loop
```

$$Q(S, A) \leftarrow Q(S, A) + \alpha \left[R + \gamma Q(S', A') - Q(S, A) \right]$$

Universidad Veracruzana

Q-Learning

- One of the early breakthroughs in RL, defined by Watkins and Dayan [2] as:

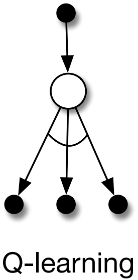
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} \gamma \max_a Q(S_{t+1}) - Q(S_t, A_t) \Big] \quad (8)$$

- The learned action-value function Q, directly approximates q_*, the optimal action-value function, independent of the policy followed.
- All what is required for correct convergence is that all pairs continue to be updated.
- Observe this is a minimal requirement, *i.e.*, any method that guarantees to find optimal behavior requires it.

Universidad Veracruzana

Backup Diagram

- What does the backup diagram of Q-learning look like?



Expected Sarsa

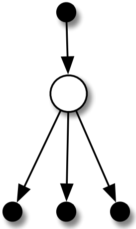
- Consider an algorithm that is just like Q-learning, except that instead of the maximum over next state-action pairs it uses the expected value, taking into account how likely each action is under current policy:

$$
\begin{aligned}
Q(S_t, A_t) &\leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma \mathbb{E}_\pi [Q(S_{t+1}, A_{t+1}) \mid S_{t+1}] - Q(S_t, A_t) \Big] \\
&\leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma \sum_a \pi(a \mid S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t) \Big]
\end{aligned}
\tag{9}
$$

- Given S_{t+1} it moves deterministically in the same direction that Sarsa moves in expectation.

Universidad Veracruzana

Backup Diagram



Expected Sarsa

Advantages

- Expected Sarsa is more complex computationally than Sarsa, but in return, it eliminates the variance due to the random selection of A_{t+1}.
- Given the same amount of experience we might expect it to perform slightly better than Sarsa, and indeed it generally does.

Universidad Veracruzana

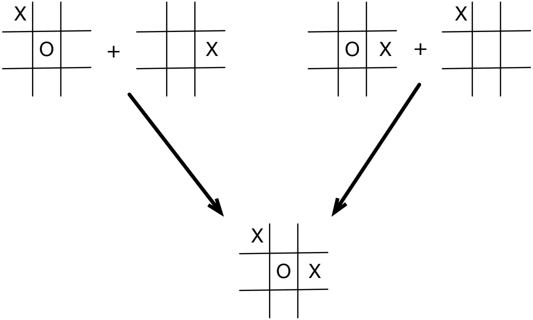
Afterstates

- Our general approach involves learning an action-value function.
- But then we reviewed a TD method for learning to play tic-tac-toe based on something closer to a state-value function.
- However, conventional state-value functions evaluates states in which the agent has the option of selecting an action; while the tic-tac-toe evaluates board positions after the agent has made its move.
- Afterstates are useful when we have knowledge of an initial part of the environment's dynamics but not necessarily the full dynamics.
- Thereby they produce a more efficient learning method.

Universidad Veracruzana

Behind Efficiency

- Many position-move pairs produce the same resulting position:



- Thus must have the same value.

Universidad Veracruzana

Referencias I

[1] R Sutton and AG Barto. *Reinforcement Learning: An Introduction*. 2nd. Cambridge, MA, USA: The MIT Press, 2018.

[2] CJCH Watkins and P Dayan. "Q-Learning". In: *Machine Learning* 8.1992 (1992), pp. 279–292.

Universidad Veracruzana