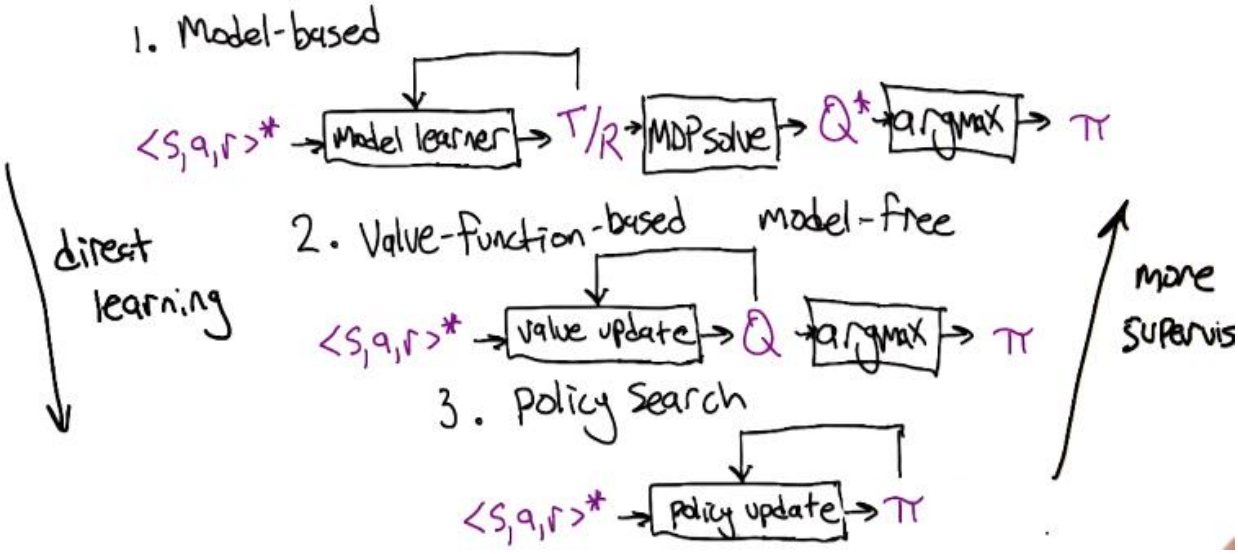
03. Temporal Difference Learning

Scan me

Introduction:

- In Reinforcement Learning, the inputs are transitions (Initial state s, action a, reward r) and the intended output is to "learn" the policy π.
- Types of Reinforcement Learning algorithms:
  - Model-based: Sending the sequence of (state/action/reward) $< s, a, r >^*$ to a Model Learner that learns the transitions and rewards. Then, an MDP solver can be used to infer an optimal value function Q^* form these transitions and rewards. Then, using $argmax$ we can get the policy π. The model learner also takes into consideration the resulting transitions and rewards when producing new transitions and rewards.
  - Value Function based (model-free): Mapping states to values. Learning values from states is quite direct but turning this into a policy might be done in some sense using an argmax.
  - Policy Search: Taking the policy itself and feed it back to a policy update that modifies the policy according to the input $< s, a, r >^*$. The problem with this approach is that the data doesn't reveal what actions need to be takes.



Temporal Difference Learning – TD(λ):

- TD Learning is learning to predict over time. We try to predict the expected sum of discounted rewards from a sequence of states where each transition has an associated reward.
- Computing estimates incrementally:

$$V_T(s_1) = \frac{(T-1)V_{T-1}(s_1)R_T(s_1)}{T}$$
$$= \frac{(T-1)}{T}V_{T-1}(s_1) + \frac{1}{T}R_T(s_1)$$
$$= V_{T-1}(s_1) + \alpha_T(R_T(s_1) - V_{T-1}(s_1)) \qquad where\ \alpha_T = \frac{1}{T}$$

  - This equation means that the update to the value is equal to the difference between the rewards we got at this step and the value we had at the previous step, multiplied by the learning rate α_T. α_T is getting smaller with higher number of episodes.

Properties of Learning Rates:

$$V_T(s_1) = V_{T-1}(s_1) + \alpha_T(R_T(s_1) - V_{T-1}(s_1)) \qquad where\ \alpha_T = \frac{1}{T}$$

- If the number of episodes T is big enough ($\sim\infty$), the estimated value $V_T(s)$ will go to the true value $V(s)$:

$$\lim_{T\to\infty} V_T(s) = V(s)$$

- So, we have two conditions over the learning rate sequence:

$$\sum_T \alpha_T = \infty$$
$$\sum_T {\alpha_T}^2 < \infty$$

- If the learning rate doesn't get smaller over time, we'll never converge. This is why α_T should always be a value associated with T.

TD(1) Update Rule:

- Pseudocode for TD(1):

  Episode T

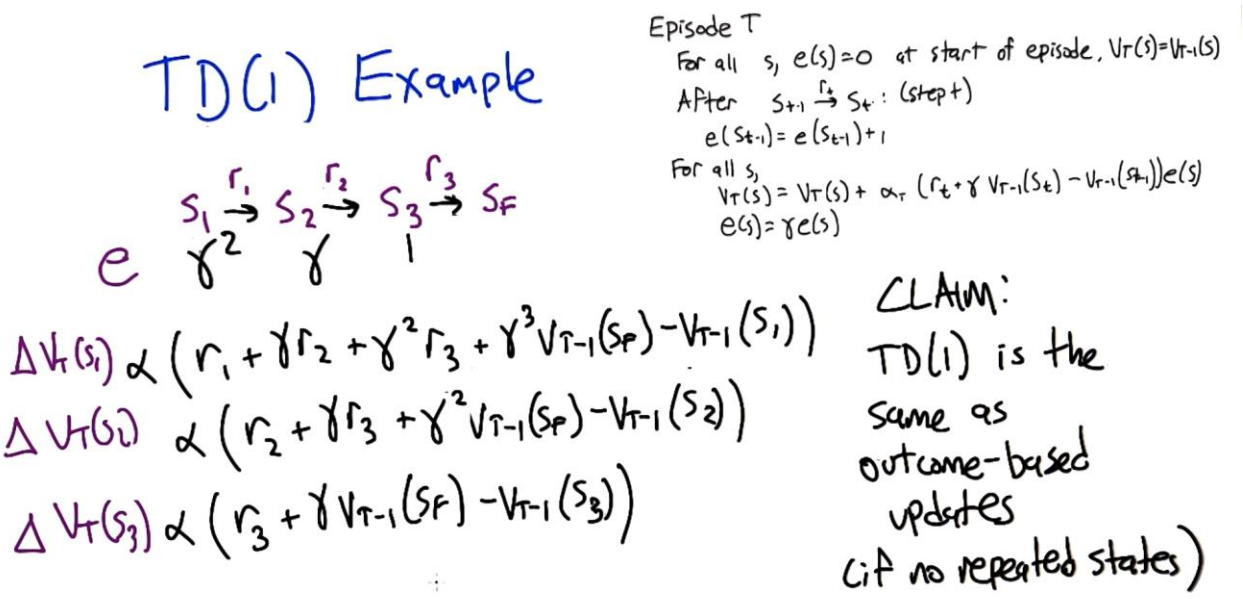
  For all s, $e(s) = 0$ at the beginning of the episode, $V_T(s_1) = V_{T-1}(s_1)$

  After $s_{t-1} \xrightarrow{r_t} s_t$: (step t)

$$e(s_{t-1}) = e(s_{t-1}) + 1$$

  For all s,

$$V_T(s) = V_T(s) + \alpha_T(r_t + \gamma V_{T-1}(s_t) - V_{T-1}(s_{t-1}))e(s)$$
$$e(s) = \gamma e(s)$$



- The problem with TD(1) is that it's using individual episodes as opposed to using all the available data, and this can result in an estimate that is really far of the true value.
- To solve this issue, we investigate the TD(0) rule.

TD(0) Update Rule:

- Pseudocode for TD(0):
  Episode T
  For all s, at the beginning of the episode, $V_T(s_1) = V_{T-1}(s_1)$
  After $s_{t-1} \xrightarrow{r_t} s_t$: (step t)

$$V_T(s_{t-1}) = V_T(s_{t-1}) + \alpha_T(r_t + \gamma V_{T-1}(s_t) - V_{T-1}(s_{t-1}))$$

$$V_T(s_{t-1}) = \mathbb{E}_{s_t}[r + \gamma V_T(s_t)]$$

TD(λ) Update Rule:

- This is a general rule of TD(1) and TD(0) where $0 \leq \lambda \geq 1$
- Pseudocode for TD(λ):
  Episode T
  For all s, $e(s) = 0$ at the beginning of the episode, $V_T(s_1) = V_{T-1}(s_1)$
  After $s_{t-1} \xrightarrow{r_t} s_t$: (step t)

$$e(s_{t-1}) = e(s_{t-1}) + 1$$

  For all s,

$$V_T(s) = V_T(s) + \alpha_T(r_t + \gamma V_{T-1}(s_t) - V_{T-1}(s_{t-1}))e(s)$$
$$e(s) = \lambda\gamma e(s)$$

k-step Estimators:

- It is useful to learn TD(λ) by studying things that are not TD(λ), like k-step estimators.
- We estimate the value of a state s_t as a function of what happens next.
- E_1: Move it a little in the direction of the immediate reward plus the discounted estimated value of the state that we landed in.
  TD(0) → One-step estimator.
- E_2: Move it a little in the direction of the immediate reward we received, plus the discounted reward that we received next, plus the double discounted value of the state we landed in two steps from now. Two-step estimator.
- E_∞ looks like TD(1)
- TD(λ) is a weighted combination of all these estimators.

$$E_1 \quad V(S_t) = V(S_t) + \alpha_T\left(r_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right) \quad \text{TD(0)}$$

$$E_2 \quad V(S_t) = V(S_t) + \alpha_T\left(r_{t+1} + \gamma r_{t+2} + \gamma^2 V(S_{t+2}) - V(S_t)\right)$$

$$E_3 \quad V(S_t) = V(S_t) + \alpha_T\left(r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 V(S_{t+3}) - V(S_t)\right)$$

$$E_K \quad V(S_t) = V(S_t) + \alpha_T\left(r_{t+1} + \cdots + \gamma^{K-1} r_{t+K} + \gamma^K V(S_{t+K}) - V(S_t)\right)$$

$$E_\infty \quad V(S_t) = V(S_t) + \alpha_T\left(r_{t+1} + \cdots + \gamma^{K-1} r_{t+K} + \cdots - V(S_t)\right) \quad \text{TD(1)}$$

k-step Estimators and TD(λ):

- We will use weighted combination of all the estimators for any one of the TD(λ) algorithms.
- When we change the value of a state, we are going to change it by all the different estimators. We will use a weighted combination of all of them each time that we do an update for a value of the state.
- We need to make sure that all the values coming from the estimators add up to one:

$$\sum_{k=1}^{\infty} \lambda^{(k-1)}(1-\lambda)$$

$$= (1-\lambda)\sum_{k=1}^{\infty} \lambda^{(k-1)} = (1-\lambda)\frac{1}{(1-\lambda)} = 1$$

$E_1 \quad 1-\lambda$

$E_2 \quad \lambda(1-\lambda)$

$E_3 \quad \lambda^2(1-\lambda)$

$E_K \quad \lambda^{K-1}(1-\lambda)$

$\cdots$

$E_\infty \quad \lambda^\infty$

- If $\lambda = 1$, we put all our weights on the last estimator, so we get TD(1).
- If $\lambda = 0$, we put all our weights on the first estimator, so we get TD(0).
- If $0 < \lambda > 1$, we get weights for all the estimators, and we combine all of them together.

TD(λ) Empirically:

- What happens if TD(λ) is run as an update algorithm on some finite amount of data?
- We run with different possible values of λ.
- Usually the error goes down as λ goes to some intermediate value and then starts to shoot up again and eventually hit error for TD(1) as λ approaches one.
- This is the justification on why we want to consider TD(λ), it gives you benefit of both TD(0) and TD(1). We are rapidly propagating information and not getting a biased estimate because of the way we are using 1 step information.