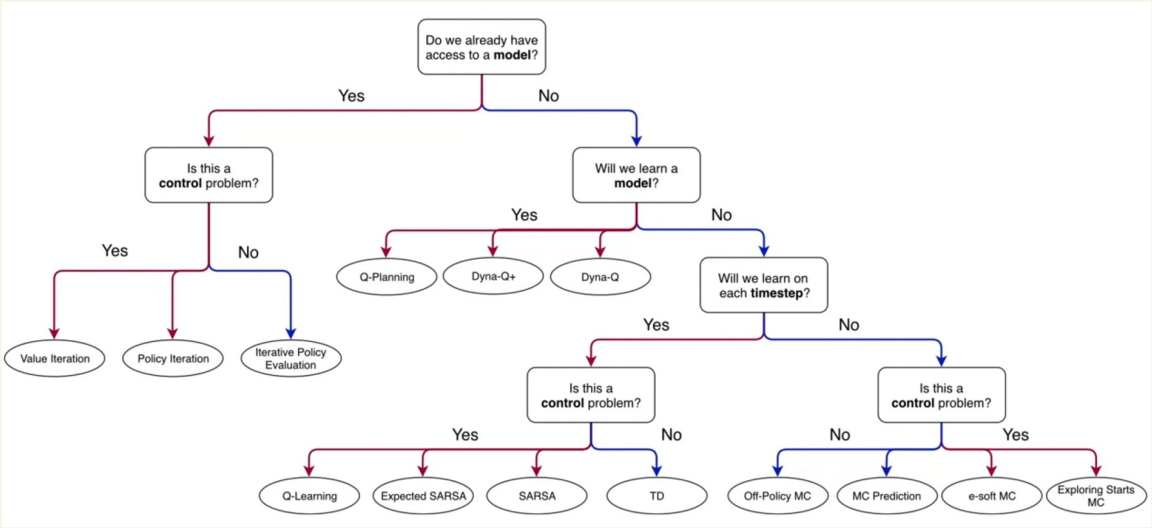
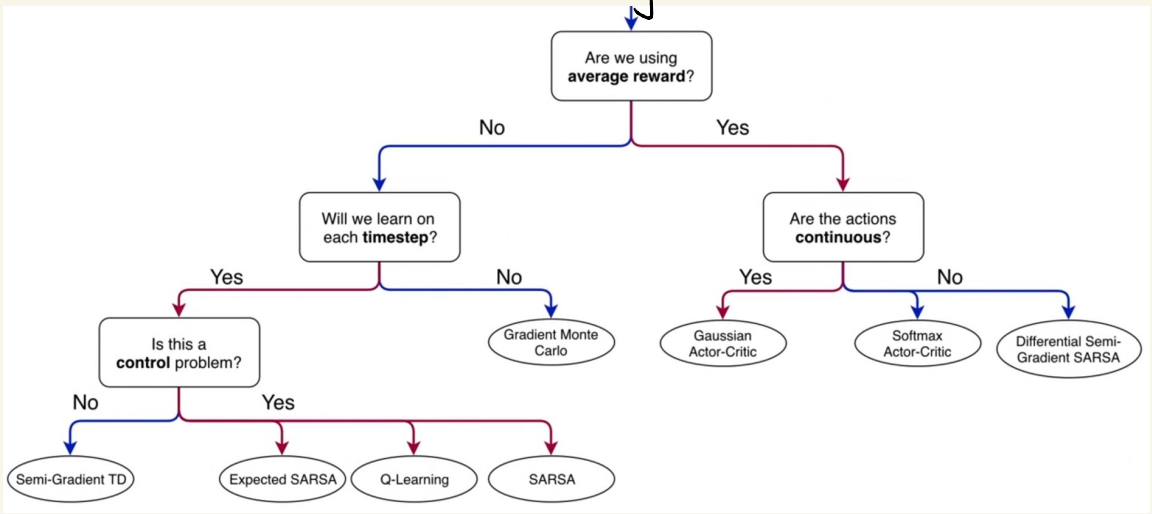
So Far:

Jan 15, 2022



In This Course!

Not Using Table



Chapter 9 - On-Policy Prediction w/ Approximation:

The approximate value function v_π will not be represented as a table, but as a parameterized functional form w/ weight vector $w \in \mathbb{R}^d$!

↳ Write $\hat{v}(s,w) \approx v_\pi(s)$ for the approximate value of state s given weight vector w.

↳ $\hat{v}$ could be a linear function, neural network, decision tree, etc.

↳ $\hat{v}$ is generalizable, meaning changing a weight(s) changes all the states.

9.1 - Value-Function Approximation:

Individual Update Notation:

$s \mapsto u$

s: state updated
u: update target

e.g.
Monte Carlo: $S_t \mapsto G_t$
TD(0): $S_t \mapsto R_{t+1} + \gamma \hat{v}(S_{t+1}, w_t)$

⚠ The estimated value for state s should be more like update target u.

Neural networks and ML methods utilize a static training set for function approximation, but RL needs to occur online w/ incrementally-acquired data.

↳ Also need to learn nonstationary target functions, which change over time (i.e. q_π can change over time).

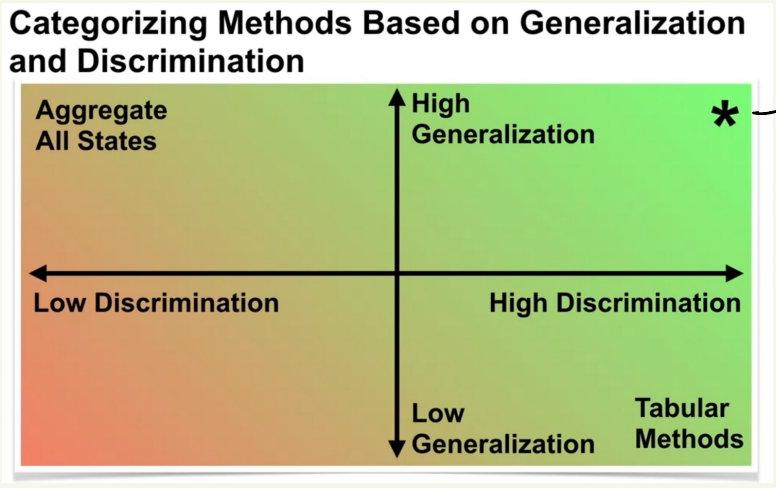
Generalization and Discrimination:

Generalization occurs when updates to the value estimate of one state influence the value of other states.

↳ Can speed up learning by making better use of experience.

↳ No need to update states one at a time.

Discrimination is the ability to make the value of two states different.



→ We want a method that can generalize and discriminate.

9.2 - The Prediction Objective ($\overline{VE}$):

In tabular cases, the algorithms always converge to an ideal learned value function. But for non-tabular methods, we must specify an objective function.

Mean Squared Value Error:

State Distribution: $\mu(s) \geq 0$

$$\sum_s \mu(s) = 1$$

$$\overline{VE}(\vec{w}) \equiv \sum_{s \in S} \mu(s) \left[v_\pi(s) - \hat{v}(s, \vec{w}) \right]^2$$

- Often $\mu(s)$ is chosen as the fraction of time spent in s → weighs states differently.
  - ↳ Results in lower error for more common states, higher error for extremes/outliers.

Ideally, $\overline{VE}$ would reach the global optimum, a weight vector $\vec{w}^*$ for which $\overline{VE}(\vec{w}^*) \leq \overline{VE}(\vec{w})$ for all possible $\vec{w}$.

↳ Often times, it only arrives at a local optimum, where $\overline{VE}(\vec{w}^*) \leq \overline{VE}(\vec{w})$ for all $\vec{w}$ near $\vec{w}^*$.

The on-policy distribution in episodic tasks

In an episodic task, the on-policy distribution is a little different in that it depends on how the initial states of episodes are chosen. Let $h(s)$ denote the probability that an episode begins in each state s, and let $\eta(s)$ denote the number of time steps spent, on average, in state s in a single episode. Time is spent in a state s if episodes start in s, or if transitions are made into s from a preceding state $\bar{s}$ in which time is spent:

$$\eta(s) = h(s) + \sum_{\bar{s}} \eta(\bar{s}) \sum_{a} \pi(a|\bar{s}) p(s|\bar{s}, a), \quad \text{for all } s \in \mathcal{S}. \tag{9.2}$$

This system of equations can be solved for the expected number of visits $\eta(s)$. The on-policy distribution is then the fraction of time spent in each state normalized to sum to one:

$$\mu(s) = \frac{\eta(s)}{\sum_{s'} \eta(s')}, \quad \text{for all } s \in \mathcal{S}. \tag{9.3}$$

This is the natural choice without discounting. If there is discounting ($\gamma < 1$) it should be treated as a form of termination, which can be done simply by including a factor of γ in the second term of (9.2).

Gradient Descent for Policy Evaluation!

Note! $\hat{v}(s, w) \equiv \langle \vec{w}, \vec{x}(s) \rangle$

$$\nabla \hat{v}(s, \vec{w}) = \vec{x}(s)$$

$$\nabla \sum_{s \in \mathcal{S}} \mu(s) [v_\pi(s) - \hat{v}(s, \vec{w})]^2 \quad \text{\& Gradient of MSE}$$

$$= \sum_{s \in \mathcal{S}} \mu(s) 2 [v_\pi(s) - \hat{v}(s, \vec{w})] \nabla \hat{v}(s, \vec{w})$$

↳ But not feasible to calculate over all states.

∴ Use batches!

9.3 - Stochastic-Gradient and Semi-Gradient Methods:

Stochastic Gradient Descent:

Batches of states: $(S_1, v_\pi(S_1)), (S_2, v_\pi(S_2)) \ldots$

$$\vec{w}_2 \equiv \vec{w}_1 + \alpha[v_\pi(S_1) - \hat{v}(S_1, \vec{w}_1)] \nabla \hat{v}(S_1, \vec{w}_1)$$

$$\vec{w}_3 \equiv \vec{w}_2 + \alpha[v_\pi(S_2) - \hat{v}(S_2, \vec{w}_2)] \nabla \hat{v}(S_2, \vec{w}_2)$$

$\vdots$

↳ Policy will slowly approach optimum in steps.

✱ But we still don't have v_π!

Gradient Monte Carlo:

Recall: $v_\pi(s) \equiv E_\pi[G_t | S_t = s]$

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha[v_\pi(S_t) - \hat{v}(S_t, \mathbf{w}_t)] \nabla \hat{v}(S_t, \mathbf{w}_t)$$

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha[\ G_t\ - \hat{v}(S_t, \mathbf{w}_t)] \nabla \hat{v}(s, \mathbf{w}_t)$$

Gradient Monte Carlo Algorithm for Estimating $\hat{v} \approx v_\pi$

Input: the policy π to be evaluated
Input: a differentiable function $\hat{v} : \mathcal{S} \times \mathbb{R}^d \to \mathbb{R}$
Algorithm parameter: step size $\alpha > 0$
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)

Loop forever (for each episode):
  Generate an episode $S_0, A_0, R_1, S_1, A_1, \ldots, R_T, S_T$ using π
  Loop for each step of episode, $t = 0, 1, \ldots, T-1$:
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha[G_t - \hat{v}(S_t, \mathbf{w})] \nabla \hat{v}(S_t, \mathbf{w})$

Semi-Gradient TD!

We can replace G_t w/ $U_t \equiv R_{t+1} + \gamma \hat{v}(S_{t+1}, \vec{w})$, like in the TD(0) algorithm.

↳ Since U_t depends on current value estimate, it may not converge to a local optimum.

↳ But it learns faster.

✠ It is semi-gradient cuz:

$$\nabla \frac{1}{2}\left[U_t - \hat{v}(S_t, \vec{w})\right]^2$$

$$= \left(U_t - \hat{v}(S_t, \vec{w})\right)\left(\nabla U_t - \nabla \hat{v}(S_t, \vec{w})\right)$$

$$\neq -\left(U_t - \hat{v}(S_t, \vec{w})\right) \nabla \hat{v}(S_t, \vec{w})$$

TD Update

Semi-gradient TD(0) for estimating $\hat{v} \approx v_\pi$

Input: the policy π to be evaluated
Input: a differentiable function $\hat{v} : \mathcal{S}^+ \times \mathbb{R}^d \to \mathbb{R}$ such that $\hat{v}(\text{terminal},\cdot) = 0$
Algorithm parameter: step size $\alpha > 0$
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose $A \sim \pi(\cdot|S)$
    Take action A, observe R, S'
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha\left[R + \gamma \hat{v}(S', \mathbf{w}) - \hat{v}(S, \mathbf{w})\right] \nabla \hat{v}(S, \mathbf{w})$
    $S \leftarrow S'$
  until S is terminal

9.4 - Linear Methods:

Linear methods approximate state-value functions by:

$$\hat{v}(s,\vec{w}) \equiv \vec{w}^T \vec{x}(s) \equiv \sum_{i=1}^{d} w_i x_i(s)$$

Gradient: $\nabla \hat{v}(s,\vec{w}) = \vec{x}(s)$

SGD: $\vec{w}_{t+1} \equiv \vec{w}_t + \alpha [U_t - \hat{v}(s,\vec{w}_t)] \vec{x}(S_t)$

Key Equations:

$$\vec{w}_{t+n} \equiv \vec{w}_{t+n-1} + \alpha [G_{t:t+n} - \hat{v}(S_t, \vec{w}_{t+n-1})] \nabla \hat{v}(S_t, \vec{w}_{t+n-1})$$

$$G_{t:t+n} \equiv R_{t+1} + \gamma R_{t+2} \dots \gamma^{n-1} R_{t+n} + \gamma^n \hat{v}(S_{t+n}, \vec{w}_{t+n-1})$$

n-step semi-gradient TD for estimating $\hat{v} \approx v_\pi$

Input: the policy π to be evaluated
Input: a differentiable function $\hat{v} : \mathcal{S}^+ \times \mathbb{R}^d \to \mathbb{R}$ such that $\hat{v}(\text{terminal},\cdot) = 0$
Algorithm parameters: step size $\alpha > 0$, a positive integer n
Initialize value-function weights $\mathbf{w}$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)
All store and access operations (S_t and R_t) can take their index mod $n+1$

Loop for each episode:
  Initialize and store $S_0 \neq$ terminal
  $T \leftarrow \infty$
  Loop for $t = 0, 1, 2, \dots$:
  | If $t < T$, then:
  | Take an action according to $\pi(\cdot|S_t)$
  | Observe and store the next reward as R_{t+1} and the next state as S_{t+1}
  | If S_{t+1} is terminal, then $T \leftarrow t+1$
  | $\tau \leftarrow t - n + 1$ (τ is the time whose state's estimate is being updated)
  | If $\tau \geq 0$:
  | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
  | If $\tau + n < T$, then: $G \leftarrow G + \gamma^n \hat{v}(S_{\tau+n},\mathbf{w})$ $(G_{\tau:\tau+n})$
  | $\mathbf{w} \leftarrow \mathbf{w} + \alpha [G - \hat{v}(S_\tau,\mathbf{w})] \nabla \hat{v}(S_\tau,\mathbf{w})$
  Until $\tau = T - 1$