Recap ①

Planning Problems: Problems in which a model of the environment is available and thus, no learning is required. Goal of such problems is to find an optimal policy, as opposed to learning an optimal policy.

What we saw in chapter 3 was planning and technically not RL: Policy iteration with model of environment available.

1) Policy evaluation:

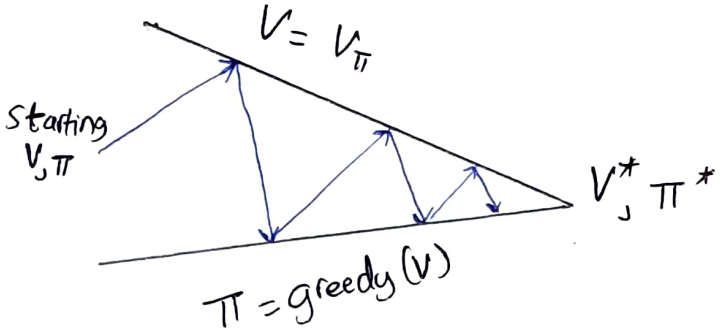
Given policy π,

$$V_{k+1}(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\left[r + \gamma V_k(s')\right]$$

2) Policy improvement:

$$\pi'(s) = \text{greedy } V_\pi(s)$$

i.e $$\pi'(s) = \underset{a}{\text{argmax}} \sum_{s',r} p(s',r|s,a)\left[r + \gamma V_\pi(s')\right]$$



Learning Problems: Refers to problems in which learning from samples is required, as there is no model of the environment available.

First, Generalised Policy Iteration:

→ As before, the goal of GPI is to obtain an optimal policy.

2 processes repeat for this:

1) Policy Evaluation - estimate V_π (using any policy evaluation algo)
2) Policy Improvement - generate $\pi' \geq \pi$ (any policy improvement algo)

→ In chapter 5, we saw, Monte Carlo, TD and TD(1), but only saw them as techniques for Policy Evaluation.

1) Monte Carlo:

$$V(S_t) \leftarrow V(S_t) + \alpha \left(G_t - V(S_t)\right)$$

2) TD:

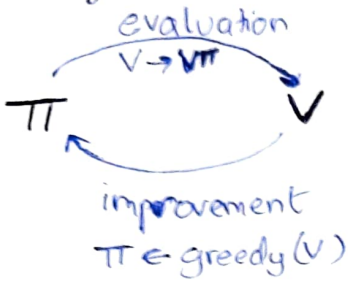
$$V(S_t) \leftarrow V(S_t) + \alpha \left(\underbrace{R_{t+1} + \gamma V(S_{t+1})}_{\text{TD target}} - V(S_t)\right)$$

→ Bootstrapping is involved

Chapter 6: Improving Agents' Behaviours

Monte-Carlo Control: Episode-wise Policy improval

First, the GPI framework:



2 steps: 1) Policy Evaluation
2) Policy improvement.

With MDP available to the agent, the Bellman Equation can be used for iterative evaluation:

$$V_{k+1}(s) = \sum_{a} \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma V_k(s')]$$

Without the MDP available to the agent, we could conduct policy evaluation by Monte Carlo prediction, but how do we improve our policy?

Can we use the standard greedy improvement scheme that we have seen in Policy-Iteration?

No! By greedily selecting our policy every time, we are leaving the agent no room for exploration, so it is very likely our agent does not see many states.

So, ~~that~~ the changes we will make are :

1) Estimate $Q(s,a)$ instead of $V(s)$.

Why? : When we try to take the best action, given $V(s)$, we require the transition function, i.e

$$\pi'(s) = \underset{a \in A}{\arg\max}\ R_s^a + P_{ss'}^a V(s')$$, but with the Q-function we have a model-free greedy policy improvement :

$$\pi'(s) = \underset{a \in A}{\arg\max}\ Q(s,a).$$

2) Make sure our agent also explores.

Why? Since we are estimating Q from samples, we only get good values of $Q(s,a)$ for those s,a pairs the agent has experienced many times. It is possible the best states were not visited enough.

So, in Monte-Carlo control, we will

1) Evaluate a policy π by estimating Q_π using Monte Carlo prediction

2) (Decaying) ε-greedy strategy for improvement of π.

ε - Greedy Exploration:

At the k^{th} iteration, suppose we are given $0 < \varepsilon < 1$, we conduct policy improvement as follows:

$$\pi_{k+1}(a|s) = \begin{cases} \varepsilon/m + (1-\varepsilon) & \text{if } a = \underset{a \in A}{\operatorname{argmax}}\, Q(s,a) \\ \varepsilon/m & \text{otherwise.} \end{cases}$$

where m is the number of available actions in state s.

SARSA: Improving policies after each step

Recall:

→ TD has many advantages over MC -

- Online
- Can be applied to incomplete experience sequences
- Can be applied to non-episodic tasks also

So, a natural idea arises:

→ Use TD instead of MC to estimate $Q(S,A)$

→ Continue ε-greedy policy improvement

→ Update every time-step

$Q(S,A) \leftarrow Q(S,A) + \alpha(R + \gamma Q(S',A') - Q(S,A))$, is the update that can be performed after each time step.

Summary of SARSA Algo:

Init $Q(s,a)\ \forall\ s \in S,\ a \in A(s)$ arbitrarily, and $Q(\text{terminal state}, \cdot) = 0$

Repeat (for each episode):

Init S (i.e pick an initial state)

→ Take action A, observe R, S'

Choose A from S using policy derived from Q

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

$S \leftarrow S',\ A \leftarrow A';$

} Repeat for each time step

Until S is terminal.

On-Policy vs Off-Policy:

On Policy refers to methods that attempt to evaluate or improve the policy used to make decisions. Straightforward: single policy generates behaviour and agent evaluates this behaviour. This same policy is improved upon

Off Policy methods attempt to evaluate or improve a policy different from the one used to generate data. 2 policies involved: one is responsible for behaviour, and the other one is being learnt.

Q-Learning:

→ A model-free off policy method, that bootstraps and directly approximates the optimal policy, despite the policy generating experiences.

→ Next action is chosen using behaviour policy $A_{t+1} \sim \mu(\cdot|S_t)$

→ But we consider an alternative successor action $A' \sim \pi(\cdot|S_t)$, and update $Q(S_t, A_t)$ towards value of alternative action, i.e

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left(R_{t+1} + \gamma Q(S_{t+1}, A') - Q(S_t, A_t)\right)$$

Off-Policy Control with Q-Learning:

→ Allow both behaviour and target policies to improve.

→ Target policy π is greedy w.r.t $Q(s,a)$.

$$\pi(S_{t+1}) = \underset{a'}{\operatorname{argmax}}\ Q(S_{t+1}, a')$$

→ Behaviour policy μ is ε-greedy w.r.t $Q(s,a)$

→ update equation (target) becomes:

$$R_{t+1} + \gamma Q(S_{t+1}, A')$$

$$= R_{t+1} + \gamma Q\left(S_{t+1}, \underset{a'}{\operatorname{argmax}}\ Q(S_{t+1}, a')\right)$$

$$= R_{t+1} + \max_{a'} \gamma Q(S_{t+1}, a')$$

Recap ②:

Model-Free RL: Refers to algos that don't use models of the environment, but are still able to produce a policy.

Trial and error learning

ex: SARSA, MC, Q-learning.

Model-Based RL: Refers to algos that can learn, but don't require a model of the environment. They attempt to learn the model through interaction with the environment

N-Step TD:

$$G_t^{(1)} = R_{t+1} + \gamma V(S_{t+1}) \quad - \text{TD}$$

$$G_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 V(S_{t+2})$$

$$\vdots$$

$$G_t^{(\infty)} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-1} R_T \quad - \text{MC}.$$

So, define n-step return $G_t^{(n)} := R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n})$

So n-step TD update equation is:

$$V(S_t) \leftarrow V(S_t) + \alpha \left(G_t^{(n)} - V(S_t) \right)$$

Problem arises: What is a good 'n'?

Natural attempt to fix this: Weighted combination of all n-step returns.

Forward View TD(λ):

λ-return G_t^λ combines all n-step returns $G_t^{(n)}$

Weight of $(1-\lambda)\lambda^{n-1}$ is used for $G_t^{(n)}$, so

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

Forward view TD(λ) update eqn:

$$V(S_t) \leftarrow V(S_t) + \alpha\left(G_t^\lambda - V(S_t)\right).$$

But, obvious issue with this:

it is just like MC, i.e it can only be computed after full episodes.

How do we tackle this?

Backward View TD(λ):

Eligibility Traces: Memory vector that keeps track of recently visited states. In other words, it assigns credit to states based on

1) Frequency Heuristic: Assign credit to most frequent states
2) Recency Heuristic: Assign credit to most recent states

We will combine both.

$$E_0(s) = 0$$
$$E_t(s) = \gamma \lambda E_{t-1}(s) + \mathbb{1}(S_t = s).$$

So, in backward view TD:

→ Keep an eligibility trace for every state s

→ Update value $V(s)$ $\forall s$, in proportion to TD-error δ_t and eligibility trace $E_t(s)$, where

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t).$$

$$V(s) \leftarrow V(s) + \alpha \delta_t E_t(s).$$

Observe: $\lambda = 0$ is same as 1-step TD.

$E_t(s) = \mathbb{1}(S_t = s)$ so $V(s) \leftarrow V(s) + \alpha \delta_t E_t(s)$

$= V(s) \leftarrow V(s) + \alpha \delta_t$.

→ Sofar, we extended both Monte-Carlo and TD learning to solve the control problem.

Next: Extending TD(1) to control (and not just evaluation)

Before that: TD(1) (Backward) algo:

Init $V(s)$ arbitrarily (but set to 0 if s is terminal)

Repeat (for each episode):

Init $E(s) = 0$, $\forall s \in S$

Init S (choose start state)

Repeat (for each time step):

$A \leftarrow$ action given by π for S,

Take action A, observe reward R and next state, S'

$\delta \leftarrow R + \gamma V(S') - V(S)$.

$E(S) \leftarrow E(S) + 1$.

$\forall s \in S$

$V(s) \leftarrow V(s) + \alpha \delta E(s)$

$E(s) \leftarrow \gamma \lambda E(s)$

$S \leftarrow S'$

Until S is terminal.