01.02.2018

example: continued

$$V^{(k)}(A) = \max_{a=\{\rightarrow,\downarrow\}} \Big(p(s'=B, r \mid s=A, a=\rightarrow) \cdot (r_{A\rightarrow B} + V^{(k-1)}(B)) + p(s'=C, r \mid s=A, a=\rightarrow) \cdot (r_{A\rightarrow C} + V^{(k-1)}(C)),$$

$$p(s'=B, r \mid s=A, a=\downarrow) \cdot (r_{A\rightarrow B} + V^{(k-1)}(B)) + p(s'\rightarrow C, r \mid s=A, a=\downarrow) \cdot (r_{A\rightarrow C} + V^{(k-1)}(C)) \Big)$$

$$= \max\big(-20.8 + 0.8\, V^{(k-1)}(B),\ -80.2 + 0.2\, V^{(k-1)}(B) \big)$$

$$V^{(k)}(B) = \max\big(19.2 + 0.8\, V^{(k-1)}(A),\ 79.8 + 0.2\, V^{(k-1)}(A) \big)$$

k	$V^{(k)}(A)$	$V^{(k)}(B)$
1	-20.8	79.8
3	39.4	88.4
⋮		
∞	51	90

→ here solved non-linear system → didn't know optimal policy

06.02.2019

Bellman equations: for optimal value function V^*, i.e. the expected return of the optimal policy π^*, when starting in state s

$$V^*(s) = \max_a \sum_{s',r} p(s',r|s,a)\,[r + \gamma V^*(s')]$$

over all possible actions

solve by value iteration algorithm, without knowing the optimal policy. ← about optimal policy is needed

- once we know V^*, we get the optimal policy by

policy extraction

$$\forall s: \quad \pi^*(s) = \operatorname{argmax}_a \sum_{s',r} p(s',r|s,a)\,[r + \gamma V^*(s')]$$

Q-function or action-value function

- If is often better to consider value function and policy together

→ action-value function $Q_\pi(s,a)$

state-value function $V_\pi(s)$	action value function $Q_\pi(s,a)$
• expect return if we follow policy π and start in state s $V_\pi(s) = \mathbb{E}_{p,\pi}\left[\sum_{t'=t+1}^{\infty} \gamma^{t'-t-1} R_{t'} \mid S_t = s\right]$ $= \sum_a \pi(a\|s)\, Q_\pi(s,a)$	• more freedom expected return if we start in state s, execute action a in the next move and only then follow policy π $Q_\pi(s,a) = \mathbb{E}_{p,\pi}\left[\sum_{t'=t+1}^{\infty} \gamma^{t'-t-1} R_{t'} \mid S_t = a, a_t = a\right]$ $= \sum_{s',r} p(s',r|s,a)\,[r + \gamma V_\pi(s')]$

can be transformed in each other

- Optimal policy is $\pi^*(s) = \operatorname{argmax}_a Q^*(s,a)$
  with $V^*(s) = \max_a Q^*(s,a)$

$Q_{\pi^*}(s,a)$ where Q^* can be determined by the coresponding Bellman equations without knowing π^*

$$Q^*(s,a) = \sum_{s',r} p(s',r|s,a)\,[r + \gamma \max_a Q^*(s',a')]$$

06.02.2018

Model-free Reinforcement Learning

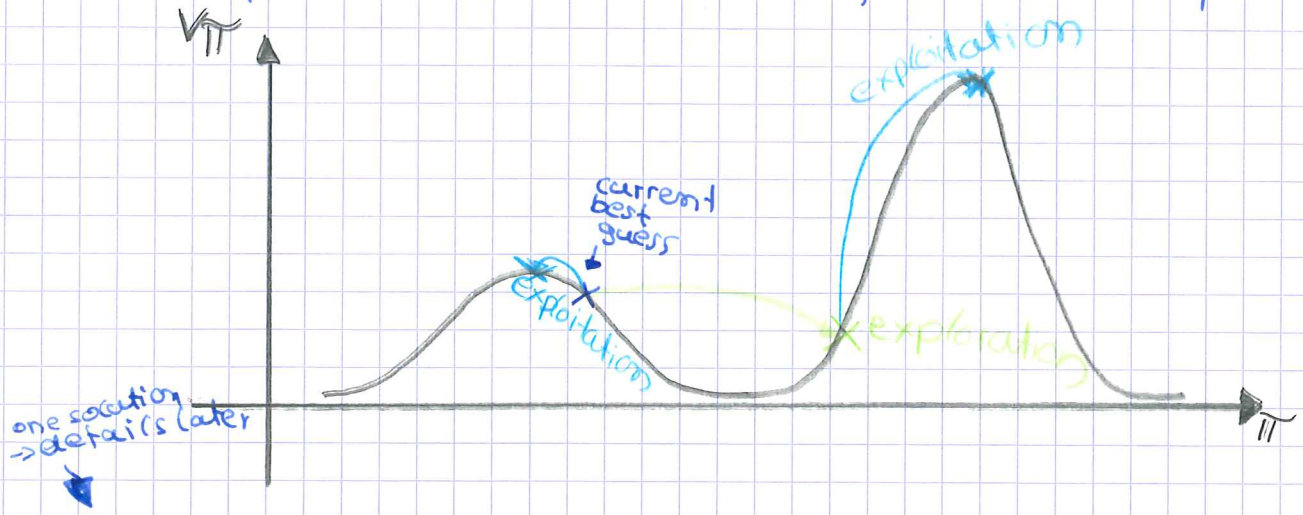
up until know we needed knowledge of probs...

- Methods so far require knowledge of the environment behavior $p(s', r | s, a)$ i.e. a model of the environment. In general, such a model is unknown.
- two approaches: 1. model-based: first learn $p(s', r | s, a)$, then derive value functions and policies via Bellman equations
  => true understanding of the world, but very difficult (similiar generative models in classification)

  2. learn values and policies directly, without explicit construction of $p(s', r | s, a)$ (world knowledge is implicit)
  => simpler, but lacks explicit "insight" ← no idea what it is doing (similiar to discriminative modeling in classification)

- model free approaches must balance two goals
  - learn something about the environment -> exploration
  - learn useful actions -> exploitation
  → trade-off

- exploitation: local search to improve the current policy
- exploration: global search to find an entirely new policy



perform the best known action most time, but a random action every once in a while

theory: policy is a valid exploration/exploitation trade-off if after an infinite number of moves, every state is visited and -"- action a/s is tried infinitely often.

goal: solve Bellman equations for V^* or Q^* without knowing transition prob $p(s', r \mid s, a)$

Monte-Carlo (MC) value estimation

- value ≙ expected return → estimate by empirical mean
- simplification: assume that every game $\tau = 1, 2, \dots$ ends after a finite number of moves T_τ
  - each game ≙ one episode
  - each move ≙ experience or event
- define expected return $G_{\tau,t}$ in step t of episode τ
  - return in episode
  - increment

$$G_{\tau,t} = \sum_{t'=t+1}^{T_\tau} \gamma^{t'-t-1} R_{\tau,t'}$$

$R_{\tau,t'}$: immediate rewards after step $t-1$ in episode τ

- run many episode with some exploration/exploitation policy π and compute empirical mean

$$\hat{V}_\pi(s) = \frac{1}{N_s} \sum_{\tau,t\,:\,s_{\tau,t}=s} G_{\tau,t}$$

N_s: total number of visits to state s; $s_{\tau,t}=s$: start in state s and some the experience

- the mean can be computed incrementally

⓪ init $\hat{V}(s) = 0$, $N_s = 0$

① repeat for all τ, t

if $s_t = s$:

$N_s \leftarrow N_s + 1$

$\hat{V}(s) \leftarrow \hat{V}(s) + \frac{1}{N_s}\left(G_{\tau,t} - \hat{V}(s)\right)$

- old guess
- new guess
- correction
- learning rate
- looks like gradient descent

generelized to arbitrary learning rates α

$$\hat{V}(s_{\tau,t}) \leftarrow \hat{V}(s_{\tau,t}) + \alpha\left(G_{\tau,t} - \hat{V}(s_{\tau,t})\right)$$

α: hyper-parameter

→ if α const → old experience will be forgotten

→ make α smaller over time

↳ better in practise

→ stabilize the system

06.02.2018

Temporal Difference (TD) value estimation

- don't calculate $G_{\tau,t}$ exactly, but approximate with current guess of $\hat{V}$

$$G_{\tau,t} = \sum_{t'=t+1}^{T\tau} \gamma^{t'-t-1} R_{t'} = R_{t+1} + \gamma \sum_{t'=t+2}^{T\tau} \gamma^{t'-t-2} R_{t'}$$

$$V_\pi(s_{t+1}) \approx \hat{V}(s_{t+1})$$

$$\approx R_{t+1} + \gamma \hat{V}(s_{t+1})$$

$\Rightarrow$ update rule:

$$\hat{V}(s_{\tau,t}) \leftarrow \hat{V}(s_{\tau,t}) + \alpha \left(R_{\tau,t} + \gamma \hat{V}(s_{\tau,t+1}) - \hat{V}(s_{\tau,t}) \right)$$

- n-step TD update: only use current guess after n steps of the sum

$$G_{\tau,t} = R_{\tau,t+1} + R_{\tau,t+2} + \dots + \gamma^{n-1} R_{\tau,t+n-1} + \gamma^n \sum_{t'=t+n}^{} \gamma^{t'-t-n} R_{t'}$$

$$G_{\tau,t} \approx \sum_{t'=t+1}^{t+n-1} \gamma^{t'-t-1} R_{t'} + \gamma^n \hat{V}(s_{t+n})$$

update rule:

$$\hat{V}(s_{\tau,t}) \leftarrow \hat{V}(s_{\tau,t}) + \alpha \left(\sum_{t'=t+1}^{t+n-1} \gamma^{t'-t-1} R_{\tau,t} + \gamma^n \hat{V}(s_{\tau,t+n}) - \hat{V}(s_{\tau,t}) \right)$$

n: hyperparameter, typical: $n = 3, \dots, 10$)
→ reduces the variance of our estimates for $G_{\tau,t}$, each $R_{\tau,t}$ is used n times

In general: TD updates have lower variance than MC updates, but corrections are slightly biased due to the use of $\hat{V}(s_{\tau,t+1})$
bias-variance trade-off

Q-learning: learn Q^* instead of V^*

best for the game

- we can always improve the current policy by updating

$\forall s: \hat{\pi}(s) \leftarrow \underset{a}{\operatorname{argmax}}\ \hat{Q}(s,a)$

$\hat{\pi}(s)$: new guess; $\hat{Q}(s,a)$: current guess

Theorem: if everything is discrete ($\hat{\pi}$, $\hat{Q}$ are stored as tables), update will never make $\hat{\pi}$ worse.

Q-learning algorithm

0. choose initial guess for $\hat{Q}(s,a)$ (good guesses → later)

1. for episodes $\tau = 1, 2, \dots$

a) create episode $s_{\tau,1}, a_{\tau,1}, r_{\tau,2}, s_{\tau,2}, a_{\tau,2}, r_{\tau,3}, \dots$, termination

using an exploration/exploitation policy (→ later)

b) for $t = 1, \dots, T_\tau - 1$

$$\hat{Q}(s_{\tau,t}, a_{\tau,t}) \leftarrow \hat{Q}(s_{\tau,t}, a_{\tau,t}) + \alpha \left[r_{\tau,t+1} + \gamma \hat{V}(s_{\tau,t+1}) - \hat{Q}(s_{\tau,t}, a_{\tau,t}) \right]$$

(correction)

two variants: 1. Q-learning:

$$\hat{V}(s_{\tau,t+1}) = \max_{a'} \hat{Q}(s_{\tau,t+1}, a')$$

2. SARSA

$$\hat{V}(s_{\tau,t+1}) = \hat{Q}(s_{\tau,t+1}, a_{\tau,t+1})$$

2. $\forall s \quad \hat{\pi}(s) = \underset{a}{\operatorname{argmax}}\ \hat{Q}(s,a)$

variants:
- update $\hat{Q}$ after every move, not after entire episode
  => rest of the episode will use the updated policy already
- n-step updates (equal to n-step TD)

corrections:

$$\sum_{t'=t+1}^{t+n-1} \gamma^{t'-t-1} r_{\tau,t'+1} + \gamma^{n-1} \hat{V}(s_{\tau,t+n}) - \hat{Q}(s_{\tau,t}, a_{\tau,t})$$

06.02.2019

- exploration / exploitation policy:
  - ϵ-greedy policy:

$$\pi(a|s) = \begin{cases} \mathbb{1}[a = a'] \text{ with } a' = \arg\max_a Q(s,a) \text{ with prob } 1-\epsilon \\ \text{uniform}(a) \text{ with prob } \epsilon \end{cases}$$

ϵ : hyperparameter, anneal overtime

$$\epsilon = \frac{\epsilon_0}{f(\tau, t)} \leftarrow \text{increases with } \tau \text{ and } t$$

- softmax policy at temperature ρ

$$\pi(a|s) \sim \frac{e^{Q(s,a)/\rho}}{\sum_{a'} e^{Q(s,a')/\rho}}$$

if $\rho \to 0$: $a' = \arg\max_a Q(s,a)$ → greedy policy

if $\rho \to \infty$: $a \sim \text{uniform}(a')$

anneal ρ over time $\rho = \frac{\rho_0}{f(\tau,t)}$ ↗ increasing with τ and t

- good initial guess:
  - required that $Q(s,a) = 0$ if s is a terminal state
- other Q are arbitrary ≥ 0, but good choice speed-up convergence
- "reward shaping": if rewards are very sparse ($r_{a,t} = 0$ most of the time), convergence is slow

  ⇒ use hand-crafted auxiliary rewards to guide algorithm towards good solutions, but without changing the optimal $\hat{\pi} \approx \pi^*$:

  $Q^{(0)}(s,a) = \phi(s)$ ("potential"): does not prefer any action a

  $\phi(s) \simeq V^*(s)$ start close to true values