Reinforcement Learning Lab

A brief review of things you may find useful

Computational cognitive modeling

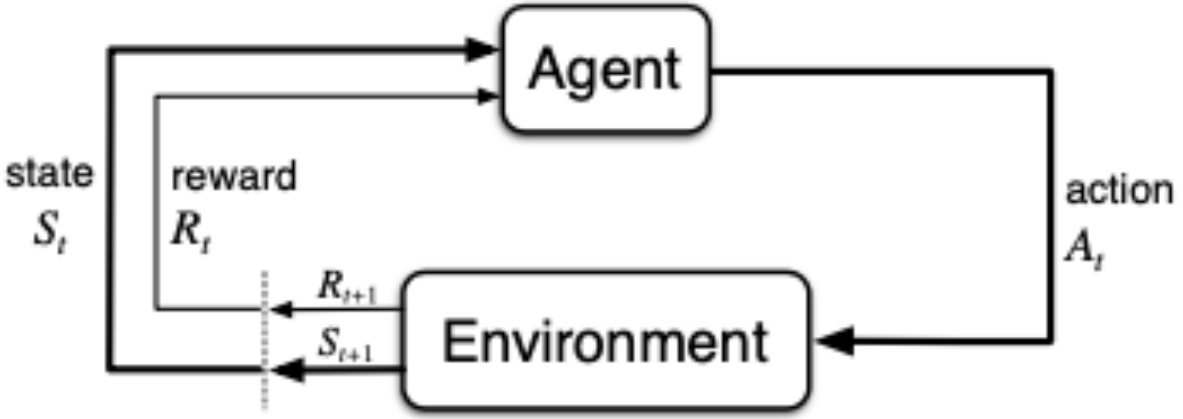
PSYCH-GA 3405.002 / DS-GS 3001.006

2020-03-02

What's different about RL?

- How is it different from supervised learning? Unsupervised learning?
- Two temporal/sequential implications
  - Reward not necessarily instantaneous
  - The order of the data matters (non-i.i.d.)
- Interaction with the environment
  - The data you see depends on the actions you take
- Reward hypothesis
  - *All* goals can be described as maximizing some expected reward function

Important Terms



- Reward and return
  - R_t vs. $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$, $0 \leq \gamma \leq 1$
  - What happens as $\gamma \to 0$? As $\gamma \to 1$?
- Agent-environment loop
  - At time t, agent receives state $S_t \in \mathcal{S}$ (sometimes also O_t for observation), takes action $A_t \in \mathcal{A}$, and then receives reward $R_{t+1} \in \mathcal{R}$ and S_{t+1}.
- Markov Decision Process formalism
  - $< \mathcal{S}, \mathcal{A}, T, R, \gamma >$
  - T: environment transition function: $T(s, a, s') = P(S_{t+1} = s' | S_t = s, A_t = a)$
  - R: environment reward function: …
  - Dynamics function: p$(s'r, | \, a, s) = P(S_{t+1} = s', R_{t+1} = r | S_t = s, A_t = a)$

Markov Assumption

- The next state depends only on the action and the current state
- Formally, $P(S_{t+1}|S_t) = P(S_{t+1}|S_0, \cdots, S_t)$
- When does it hold? When doesn't it hold?
- Can be viewed as restriction on the environment, or on the state.

RL Agent Components

- Policy: decision-making, mapping from state to action
  - Deterministic: $a = \pi(s)$
  - Stochastic: $\pi(a|s) = P(A_t = a|S_t = s)$
  - Does an agent have to have a policy?
- Value function: how good is a state over time?
  - Formally, $v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots | S_t = s] = \mathbb{E}_\pi[G_t | S_t = s]$
  - Q-value, state-action value: $q_\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a]$
  - Why do these depend on the policy?
  - Does an agent have to have a value function?
- Model: transition function estimation; a way to simulate experience
  - Does an agent have to have a model?

Types of RL Agents: Value-based

- TD-learning (or TD(0)):
  - $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - \mathrm{V(S_t)}$
  - $V(S_t) \leftarrow V(S_t) + \alpha\delta_t$
- SARSA: learn Q-value based on each transition experienced:
  - $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$
  - Why is it called SARSA? State-Action-Reward-State-Action
- Q-learning: similar, but different:
  - $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$
  - What's the difference?
- Modern version: DQN, deep Q-networks
- On-policy vs. off-policy

Types of RL Agents: Policy-based

- Policy gradient: map states directly to actions:
  - Construct feature vectors $\boldsymbol{x}(s,a) \in \mathbb{R}^d$ (hand-crafted or learned)
  - Learn policy parameters $\boldsymbol{\theta} \in \mathbb{R}^d$
  - Compute state preferences : $h(s,a;\boldsymbol{\theta}) = \boldsymbol{\theta}^{\boldsymbol{T}}\boldsymbol{x}(s,a)$ in the linear case
  - Softmax policy: $\pi(a|s;\theta) = \frac{\exp(\beta h(s,a;\boldsymbol{\theta}))}{\sum_{a\prime}\exp(\beta h(s,a\prime;\boldsymbol{\theta}))}$, β is the inverse temperature
  - Learning: policy gradient [Sutton & Barto 13.2, pp. 325]
- State preferences $h(s,a;\boldsymbol{\theta})$ can be arbitrarily complicated (DNNs)
- Can you combine policy-based and value-based methods?
- Yes! Actor-critic (learn both policy and value estimate, use value estimate to estimate the 1-step return, use that to update the policy)
  - Modern methods: advantage actor critic (A2C), asynchronous A2C (A3C).

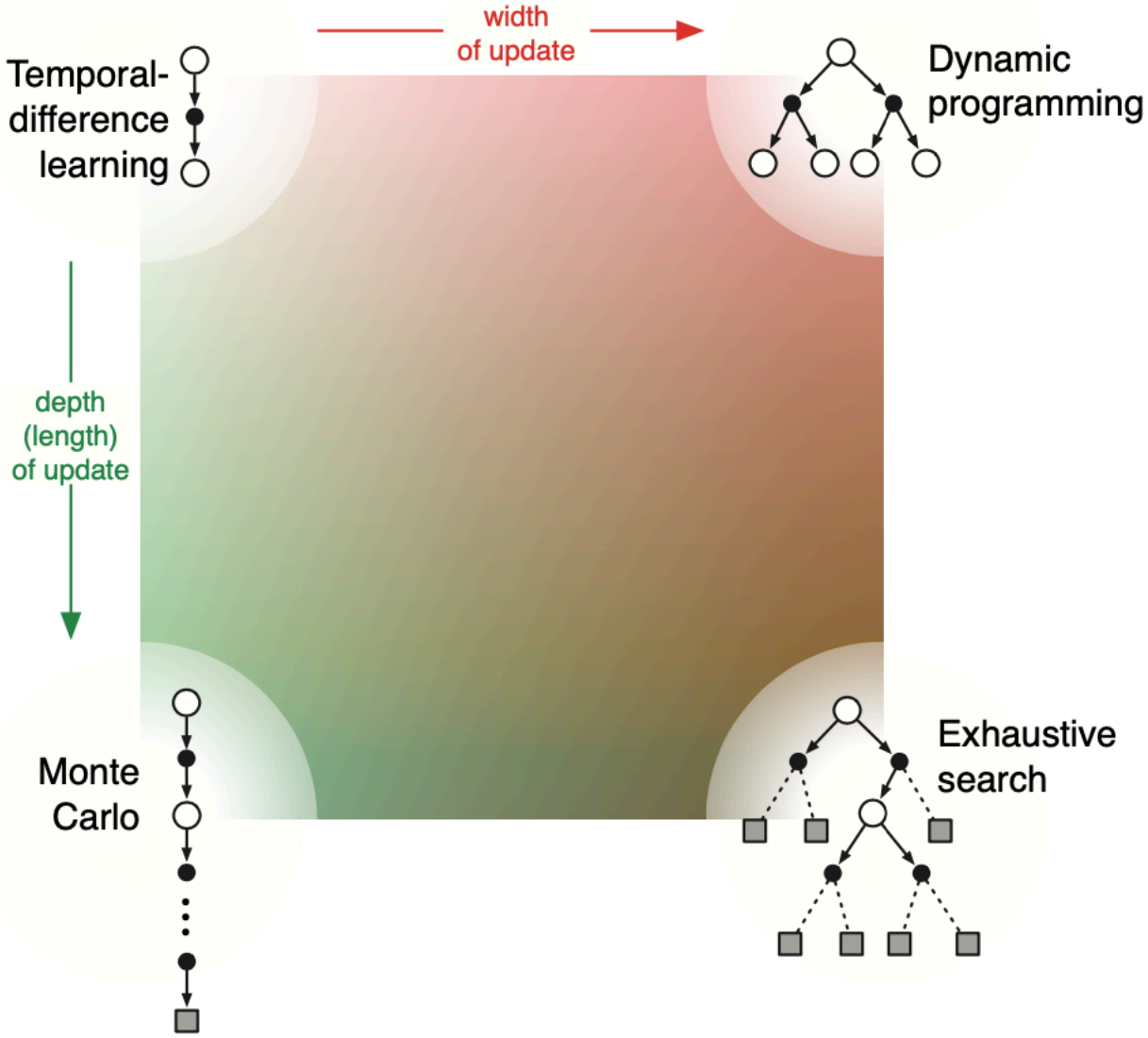
Types of RL Agents: Model-based

- Previous methods only used direct experience to improve estimates
- Model based methods use a model to simulate experience, estimate values from simulated experience, which informs the policy
  - Used to collect more experience, which allows improving the model…
- If the model is good: improved sample efficiency and generalization
- If the model is bad: learning is made much harder
- Relevant algorithms: Dyna, Monte-Carlo Tree Search

Update width vs. depth

- Width: sample updates (single trajectory) vs. expected updates (distribution or best-known)
- Depth: estimate for single timestep vs. arbitrary depth vs. full return (until termination)
- Interpolations in the middle possible

Sutton & Barto, ch. 8.13, p. 190



Other questions to ask about your RL method

- **Exploration:** how do you trade-off exploration and exploitation?
- **Updates:** do you update all states simultaneously? Or one state at a time? Which states do you update at each pass?
- **Real vs. simulated data:** do you use both? If so, how do you trade off?
- **Update timing:** do you update while interacting with the environment? Or after each episode?

Other questions to ask when applying RL

- Do I need to be using RL, or is there a supervised formulation?
- What is my reward? Does it actually capture the behavior I imagine?
- Is there a natural episode structure/boundary?
- Are my states Markov?
- Is my state space continuous or discrete?
- Is my action space continuous or discrete?
- Are there similar problems with a literature of trying RL on? Can their approaches serve as a good baseline for me?

Helpful resources

- David Silver's course: https://deepmind.com/learning-resources/-introduction-reinforcement-learning-david-silver
- Sutton & Barto's Introduction to Reinforcement Learning: http://incompleteideas.net/book/the-book.html
- Shorter but technical: Szepesvari's Algorithms of Reinforcement Learning: https://sites.ualberta.ca/~szepesva/rlbook.html
- At NYU: Lerrel Pinto's Deep RL course: https://docs.google.com/document/d/1yr3IeReI0fc5uiXpsBIL7r29R-tXpLoccMAOc3SBL_w/edit