Integrated Domain-Independent Learning

Gal A. Kaminka

Integrated Learning and Action Selection

Where Does Learning Take Place?

```
1 W knowledge base, g goal, B behaviors/actions
2
3 while g not satisfied:
4    PERCEIVE() to update W
5    CHOOSE() action b (from B)
6    EXECUTE() action b
```

Two opportunities for learning

- Step 5: Learn choice, given W, B
- Step 6: Learn effects of action b (in step 4)
  - step 4 of time $t+1$ shows effects of step 6, time t

Learning CHOICES

```
W knowledge base, g goal, B behaviors/actions

while g not satisfied:
   PERCEIVE() to update W
   CHOOSE() action b (from B)
   EXECUTE() action b
```

CHOOSE()

- CHOOSE() can be very complex procedure
  - Call planner, use previous cases, ask someone
- Worthwhile to learn the results
- This is sometimes called *speed-up learning*
- More generally, it learns what action to take given selection

Learning ACTIONS (Action Models)

Effects of EXECUTE at time t PERCEIVE'd at time $t+1$ (or later)

```
1 W knowledge base, g goal, B behaviors/actions
2
3 while g not satisfied:
4    PERCEIVE() to update W
5    CHOOSE() action b (from B)
6    EXECUTE() action b
```

Action Model (*Forward Model*)

- Step 6: Execute action b
- Step 4: Find out its effects
    - Also find out whether it was as predicted

Integrating Learning with Acting

Deterministic Worlds

- Rote Learning: Cache/memorize (propositional)
  - Store decisions reached
  - Store effects of action
- Explanation-based learning: Generalize (relational)
  - Learn general rule for CHOOSING
  - Learn general effects of actions

Non-Deterministic Worlds

- Reinforcement Learning
  - Learn what action to choose (roughly: model-free)
  - Learn effect of action (roughly: model-based)

Rote Learning

CHOICE learning

- Given W, $b =$CHOOSE(), remember rule $W \Rightarrow b$
- Problem: Effects not always deterministic
- Problem: No generalization of state W

Rote Learning

CHOICE learning

- Given W, $b = $CHOOSE(), remember rule $W \Rightarrow b$
- Problem: Effects not always deterministic
- Problem: No generalization of state W

ACTION learning

- Given W, b, new W', remember effects(b)$=W'/W$
  - What beliefs $\langle k, v \rangle$ changed (same key, new value)
- Problem: Confused by extemporaneous effects
  - Not everything happens because of actions

Explanation-Based Learning (Briefly)

Use knowledge of *variables* used

CHOICE learning

- Given W, $b =$CHOOSE(), remember rule $W' \Rightarrow b$
  - Where W are beliefs $< k, v >$ *used* in CHOOSE
  - e.g., when CHOOSE uses a planner, focus on conditions tested by planner
  - Requires transparent CHOOSE procedure
- This "shortcuts" the decision the next time it is encountered
- Generalizes, recursively creating rules which short-cut other rules

ACTION model learning

- Similarly to ROTE learning, and with similar problems

Reinforcement Learning (Bird's Eye View)

- Assume problem is an MDP (there are more general models)
  - Actions lead to resulting states probabilistically
    - Classic planning is a special deterministic case
  - Goal is encoded in utility/reward
- We can plan using the MDP as domain knowledge
  - Through Value Iteration or Policy Iteration
- But: We do not know the MDP parameters
  - **the action forward model**
    - The transition probabilities
    - The rewards
- Reinforcement Learning comes to address this

Reinforcement Learning (Bird's Eye View)

- Assume problem is an MDP (there are more general models)
  - Actions lead to resulting states probabilistically
    - Classic planning is a special deterministic case
  - **Goal is encoded in utility/reward** $\Rightarrow$ Revisit later
- We can plan using the MDP as domain knowledge
  - Through Value Iteration or Policy Iteration
- But: We do not know the MDP parameters
  - **the action forward model**
    - The transition probabilities
    - The rewards
- Reinforcement Learning comes to address this

Markov Decision Processes (MDPs)

MDP is a tuple $\langle S, A, T, R \rangle$

S finite set of states

A finite set of actions

T stochastic transition function $T(s, a, s') = Pr(s'|s, a)$

R instant scalar reward for taking transition $R(s, a, s')$

Markov Decision Processes (MDPs)

MDP is a tuple $\langle S, A, T, R \rangle$

S finite set of states

A finite set of actions

T stochastic transition function $T(s, a, s') = Pr(s'|s, a)$

R instant scalar reward for taking transition $R(s, a, s')$

Standard problem:

- Find optimal policy π^*, mapping $S \rightarrow A$
- π^* maximizes expected value (sum of rewards)

Model-Based vs Model-Free

Given MDP with unknown T, R

Two approaches to RL[1]

- Model-free (also *direct*): Learn $s \rightarrow a$
  - Directly, ignoring the MDP
  - Know what to do, without the model behind it
  - This is akin to **learning CHOOSE**
- Model-based (also *indirect*): Learn MDP, use it
  - Learn the MDP parameters
  - Use MDP planning to generate next action
  - This involves both **learning CHOOSE, action model**

[1] Survey of Model-Based Reinforcement Learning: Applications on Robotics, Athanasios S. Polydoros and L. Nalpantidis, in Journal of Intelligent and Robotic Systems, 86(2):153–173, 2017.

Side-note: RL In Human Brains

- Studies show animals/humans have both model-free and model-based RL mechanisms[2]
- Not clear when one is used, and when the other
  - Looks like it is done parallel

[2] Reinforcement learning: The Good, The Bad and The Ugly, Peter Dayan and Yael Niv, Current Opinion in Neurobiology 2008, 18:1–12}

Model-Free approaches

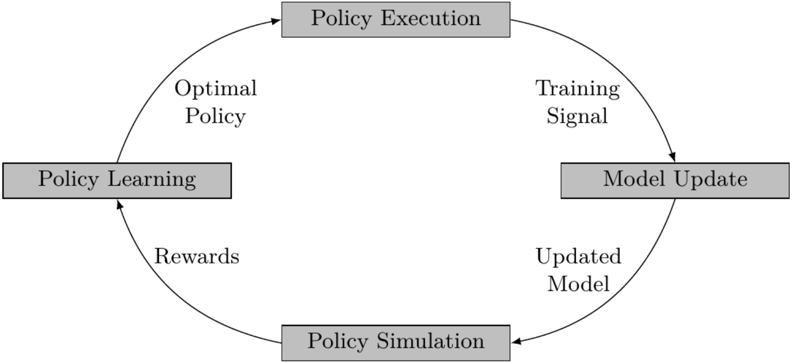
- Q-Learning is the most famous (others exist, e.g., SARSA)
  - Incrementally approximates optimal Q^* value
  - This is used to compute the best V^* value of policy
- Q-learning is a value-function learner
  - Searches through space of Q values
- Can also have policy learners
- Or mixed (Actor-Critic learns both value and policy)

Completely avoids learning the transitions and rewards

Model-Based Reinforcement Learning (MBRL)

- Learn transition probability function $T(s, a, s^{new})$
- Learn reward function $R(s, a, s')$
- This corresponds to learning the action model
- But MBRL also use them to predict next best action
  - thus also learn CHOOSE

Basic MBRL Approach



[3]

[3]Survey of Model-Based Reinforcement Learning: Applications on Robotics, Athanasios S. Polydoros and L. Nalpantidis, in Journal of Intelligent and Robotic Systems, 86(2):153–173, 2017.}

General Approach to MBRL

1. Start with empty model (random/guess)
2. Use MDP planning to generate a policy π
   - Value or policy iteration
3. Execute policy, noting actual transitions and rewards
4. Update the model
5. Goto step 2

When does it stop?

---

[4] Ronen I. Brafman and Moshe Tennenholtz, R-MAX—A General Polynomial Time Algorithm for Near-Optimal Reinforcement Learning, Journal of Machine Learning Research 3:213–231, 2002

[5] Ian Osband, Benjamin Van Roy, Daniel Russo, (More) Efficient Reinforcement Learning via Posterior Sampling, Arxiv, 2013}

General Approach to MBRL

1. Start with empty model (random/guess)
2. Use MDP planning to generate a policy π
   - Value or policy iteration
3. Execute policy, noting actual transitions and rewards
4. Update the model
5. Goto step 2

When does it stop?

Several Variants: R-MAX[4], PSRL[5], ...

---

[4] Ronen I. Brafman and Moshe Tennenholtz, R-MAX—A General Polynomial Time Algorithm for Near-Optimal Reinforcement Learning, Journal of Machine Learning Research 3:213–231, 2002

[5] Ian Osband, Benjamin Van Roy, Daniel Russo, (More) Efficient Reinforcement Learning via Posterior Sampling, Arxiv, 2013}

Specific MBRL: R-MAX Algorithm

Key idea: Assume what is unknown is maximally rewarding

R-MAX for MDPs:

1. Initialize *optimistic* model
   - Maximum rewards, all transitions possible
2. Repeat
   - 2.1 Compute an optimal D-step policy for current state
   - 2.2 Execute D steps, or until new transition taken (new state discovered)
   - 2.3 Observe: Let a be action taken in state s_i
   - 2.4 Update $T(s_i, a, s_{i+1})$, $R(s_i, a, s_{i+1})$

This has bias towards exploration early, switching to exploitation

R-MAX MDP Initialization

Initial MDP $M = \langle S, A, T, R \rangle$:

- $S \leftarrow s_0, \ldots, s_n$ (s_0 is special)
- $A \leftarrow a_1, \ldots, a_k$
- $\forall s, t \in S, a \in A \qquad T(s, a, t) \leftarrow 1$
- $\forall s, t \in S, a \in A \qquad R(s, a, t) \leftarrow R_{max}$

Assume: Known n, k, R_{max}

For each $s \in S$:

- Mark s as *unknown*
- $\forall s, t \in S$, set $count(s, a, t) \leftarrow 0$
- $\forall s, t \in S, a \in A$, note that $R(s, a, t)$ was not observed

Updating

For all actions a in the D-step policy:

- Assume a applied in state $s \in S$, and resulted in state $t \in S$
- If a was taken for the first time, record $R(s,a,t)$
- Update $count(s,a,t)$
- If the a was taken K_1 times in s
  - mark s as *known*
  - $\forall t \in S$, update $T(s,a,t)$ according to $count(s,a,t)$
  - $K_1 \leftarrow 1 + max(\lceil(\frac{4nDR_{max}}{\epsilon})^3\rceil, \lceil -6ln^3(\frac{\delta}{6nk^2})\rceil)$
    - ϵ error bound, δ failure probability

Comparing Model-Based and Model-Free RL

Model-Free

- + Much easier to implement
- + Less assumptions on model underlying the domain
- + $O(1)$ runtime, $O(S \times A)$ memory (naive)
- - Large # of interactions, dangerous and slow
- - Strictly reward-dependent

Comparing Model-Based and Model-Free RL

Model-Free

- + Much easier to implement
- + Less assumptions on model underlying the domain
- + $O(1)$ runtime, $O(S \times A)$ memory (naive)
- - Large # of interactions, dangerous and slow
- - Strictly reward-dependent

Model-Based

- + Smaller # of interactions, data efficient
- + Fast convergence
- + Can be reward independent
- - High run-time complexity
- - $O(S \times A \times S)$ memory
- - Highly-dependent on model

Main Issues

- Domain-dependent vs domain-independent $\Rightarrow$ **Big Issue**
- Exploration vs Exploitation

Reward is domain dependent, task dependent

Domain Independent Rewards

Rewards in MDPs

- Rewards are part of MDPs
- Naively, tied to particular goal state
- But can be learned/used independently from transition probabilities

Types of Rewards

- *Intrinsic* or *Task-Dependent*[6]
- Originating from *external* or *internal* signals[7]
  - e.g., from perception of environment (external)
  - e.g., from perception of battery or clock (internal)

[6] {Sing, S., Barto, A. G., Chentanez, N. {*Intrinsically Motivated Reinforcement Learning*}, *NIPS 2004*}

[7] {Oudeyer, P.-Y., and Kaplan, F. {*How can we define intrinsic motivation?*}, *Proc. Of the 8th Conf. On Epigenetic Robotics, 2008*}

Example of Rewards

- Heuristic distance to goal
- Effectiveness Index (EI), 1-1/EI
- Surprise
- Empowerment
- ...

Are these intrinsic or task-dependent? Internal or External?

Exploration vs Exploitation ⇒ **Big Issue**

Examples:

- Epsilon-greedy exploration
- Boltzman exploration
- Win or Lose Fast (WOLF)
- . . .