Learning in Multi-Agent Systems

27 March, 2017

Janyl Jumadinova
Learning in Humans

- The act / process of acquiring, modify or reinforcing knowledge or skills through synthesizing different types of new or existed information.
Learning in Humans

- The act / process of acquiring, modify or reinforcing knowledge or skills through synthesizing different types of new or existed information.
- Key to human survival.
Learning in Humans

- The act / process of acquiring, modify or reinforcing knowledge or skills through synthesizing different types of new or existed information.
- Key to human survival.
- Progress over time tends to follow learning curves (relatively permanent).
Learning in Computing Systems

- Computational methods using “experience” to improve performance.
Learning in Computing Systems

- Computational methods using “experience” to improve performance.
- Experience – data driven task
Learning in Computing Systems

Machine learning
Learning in Computing Systems

Machine learning

- Computer program(s) with adaptive mechanisms that enable computer / machine to learn from experience / example / analogy / rewards.
Learning in Computing Systems

Machine learning
★ Computer program(s) with adaptive mechanisms that enable computer / machine to learn from experience / example / analogy / rewards.
★ It improves the performance of a system over time (e.g, reducing error rate, improving rewards).
Main Learning Paradigms

- **Supervised learning**
  - input-output relationships
Main Learning Paradigms

- Supervised learning
  - input-output relationships

- Unsupervised learning
  - relationship among inputs
Main Learning Paradigms

- **Supervised learning**
  - input-output relationships

- **Unsupervised learning**
  - relationship among inputs

- **Reinforcement learning**
  - input-action relates to rewards / punishment
Main Learning Paradigms

- **Supervised learning**
  - input-output relationships

- **Unsupervised learning**
  - relationship among inputs

- **Reinforcement learning**
  - input-action relates to rewards / punishment
Reinforcement Learning

▶ Learning approach of getting a computer system to act in the world so as to maximize its rewards.
Reinforcement Learning

- Learning approach of getting a computer system to act in the world so as to maximize its rewards.
- Consider teaching a small child. We cannot tell him/her what to do, but we can reward / punish if he/she does the right/wrong thing.
Reinforcement Learning

- Learning approach of getting a computer system to act in the world so as to maximize its rewards.
- Consider teaching a small child. We cannot tell him/her what to do, but we can reward / punish if he/she does the right/wrong thing.
- Process to determine what it did that made it get the reward / punishment – “credit assignment problem.”
Reinforcement Learning
Reinforcement Learning

Basic idea:

- Receive feedback in the form of rewards.
- Agent’s utility is defined by the reward function.
- Must (learn to) act so as to maximize expected rewards.
Agents can use:

- **model-based learning**: model the other agents and compute optimal action based on this model and knowledge of the reward structure (the agent attempts to learn a model of its environment), or

- **model-free**: directly learn the expected utility (probability \cdot payoff) of actions in a given state.
Model-Based vs. Model-Free: Expected Age

**Goal:** Compute expected age

### Known $P(A)$

$$E[A] = \sum_a P(a) \cdot a = 0.35 \times 20 + \ldots$$

### Without $P(A)$, instead collect samples $[a_1, a_2, \ldots, a_N]$:

#### Unknown $P(A)$: “Model Based”

$$\hat{P}(a) = \frac{\text{num}(a)}{N}$$

$$E[A] \approx \sum_a \hat{P}(a) \cdot a$$

#### Unknown $P(A)$: “Model Free”

$$E[A] \approx \frac{1}{N} \sum_i a_i$$
Model-based learning

If the task environment is deterministic, and consists of only a modest number of discrete states, then a simple approach to model learning is to implement a look up table.
Model-based learning

If the task environment is deterministic, and consists of only a modest number of discrete states, then a simple approach to model learning is to implement a look up table.

- Example: fictitious play
  
  ▶ agents use information about the past actions and payoffs of other players to update their own beliefs about the other players in the next time step,
  
  ▶ then the agents choose optimal action based on those beliefs.
Model-based learning

If the task environment is deterministic, and consists of only a modest number of discrete states, then a simple approach to model learning is to implement a look up table.
- Example: fictitious play
  - agents use information about the past actions and payoffs of other players to update their own beliefs about the other players in the next time step,
  - then the agents choose optimal action based on those beliefs.

For example, agent observes time-average frequency of other players’ action choices, and models:

\[ P(\text{action } a) = \frac{\# \text{times}_a \text{ observed}}{\text{total}_\text{num}_\text{observations}} \]

agent then selects best-response action to this model.
Model-free reinforcement learning

- **Idea**: learn how to act without explicitly learning the transition probabilities $P(s' | s, a)$
Model-free reinforcement learning

- **Idea**: learn how to act without explicitly learning the transition probabilities $P(s'|s, a)$
- **Q-learning**: learn an action-utility function $Q(s, a)$ that tells us the value of doing action $a$ in state $s$
  - $V(s) = \max_a Q(s, a)$
  - $Q(s, a) \leftarrow Q(s, a) + \beta (R(s) + \gamma \max_{a'} Q(s', a'))$, where $\beta$ is the small constant called *learning rate*
  - Selected action: $\pi(s) = \arg\max_a Q(s, a)$
At each step $s$, choose the action $a$ which maximizes the function $Q(s, a)$

- $Q$ is the estimated utility function - it tells us how good an action is given a certain state

$Q(s, a) = \text{immediate reward for making an action} + \text{best utility (Q) for the resulting state}$
Q-learning

- At each step $s$, choose the action $a$ which maximizes the function $Q(s, a)$
  - $Q$ is the estimated utility function - it tells us how good an action is given a certain state

- $Q(s, a) = \text{immediate reward for making an action} + \text{best utility (Q) for the resulting state}$

- Note: recursive definition
Exploration vs. Exploitation

- Exploration: change to a different random strategy
- Exploitation: keep selecting the best strategy so far