Reinforcement Learning

Shan-Hung Wu
shwu@cs.nthu.edu.tw

Department of Computer Science,
National Tsing Hua University, Taiwan

NetDB-ML, Spring 2015

Outline

Outline

Why?

- In supervised learning, we see examples $\boldsymbol{x}^{(t)}$'s that are 1) i.i.d. and 2) given the unambiguous "right" labels $r^{(t)}$'s
- In sequential decision making and control problems, neither holds
- The next example may be the outcome of your "action" to the previous example
- It is very difficult to provide explicit supervision on the "correct" action of an example
  - E.g., if we have just built a four-legged robot and are trying to program it to walk, then initially we have no idea what the "correct" actions ($r^{(t)}$) to take are to make it walk under a certain condition ($\boldsymbol{x}^{(t)}$)

Reinforcement Learning

- In the reinforcement learning framework, we will instead provide only a ***reward function*** for the learning algorithm to maximize
  - In the four-legged walking example, the reward function might give the robot positive rewards for moving forwards, and negative rewards for falling over
- Based on Markov Decision Process

Outline

1. **Preliminaries**
   - Random Process and Markov Process
2. **Markov Decision Process**
   - Definitions
   - Bellman Equations
   - Determining the Best Actions: Value and Policy Iteration
   - Learning a Model for MDP
3. **Continuous State MDPs**

Random Processes (1/2)

Definition

Given a probability space $(\Omega, \mathcal{F}, P)$, a ***random process*** (or ***stochastic process***), denoted by $\{X^{(t)} : t \in \mathcal{T}\}$, is a collection of random variables defined over $(\Omega, \mathcal{F}, P)$ and indexed by elements t of a set $\mathcal{T}$.

- Typically, we think $t = 1, 2, \cdots$ as ***time***
  - t could also be space or position on a DNA string, etc.
- The values of $X^{(t)}$ are called the ***states***
- $X^{(t)}$ is a function of both the outcome ω and time t
  - Fixing the outcome $\omega \in \Omega$, $X^{(t)}$ is a deterministic function of t
  - Fixing t, $X^{(t)}$ is a random variable
- The distribution of $X^{(t)}$, called ***state distribution***, changes with t

Random Processes (2/2)

- A random process is said to be ***discrete*** or ***continuous in time*** depending on whether $\mathcal{T}$ is discrete (finite or infinitely countable) or continuous
- A random process is ***discrete*** or ***continuous in state*** depending on whether X is discrete or continuous
- Random processes vs graphical models (e.g., Bayesian networks and Markov random fields)?

Random Processes (2/2)

- A random process is said to be ***discrete*** or ***continuous in time*** depending on whether $\mathcal{T}$ is discrete (finite or infinitely countable) or continuous
- A random process is ***discrete*** or ***continuous in state*** depending on whether X is discrete or continuous
- Random processes vs graphical models (e.g., Bayesian networks and Markov random fields)?
  - A Bayesian network degenerates into a discrete-state-and-time random process when its random variables are indexed by time

Why Random Process?

- So far, we assume that instances in a dataset are i.i.d.
  - This simplifies the calculation of the likelihood $P[\mathcal{X}|\theta]$
- However, in practice instances may come in order and successive instances may be dependent
  - E.g.,. letters in a word, phonemes in speech utterance, page visits in the Web, etc.
- The sequence can be characterized as being generated by a parametric random process
  - We want to estimate the parameter of a random process and then make predictions

Markov Processes

- A random process is called the ***Markov process*** if it satisfies the ***Markov property***: $P\left[X^{(t_0+t_1)} \leqslant x | X^{(t_0)} = x_0, X^{(t)} = x_t, -\infty < t < t_0\right] = P\left[X^{(t_0+t_1)} \leqslant x | X^{(t_0)} = x_0\right]$
  - We can think t_0 and t_1 be the present and future time respectively
  - This is a mathematical version of the saying "today is the first day of the rest of your life"

	States are fully observable	**States are partially observable**
Transition is autonomous	Markov chains	Hidden Markov models
Transition is controlled	Markov decision processes	Partially observable Markov decision processes

Outline

1. Preliminaries
   - Random Process and Markov Process
2. **Markov Decision Process**
   - Definitions
   - Bellman Equations
   - Determining the Best Actions: Value and Policy Iteration
   - Learning a Model for MDP
3. Continuous State MDPs

Outline

Markov Decision Process

- A Markov decision process $\{X^{(t)}\}_t$ defined over $(\mathcal{S}, \mathcal{A}, \gamma, R)$ is a Markov process, where
  - $\mathcal{S}$ is the state space
  - $\mathcal{A}$ is the ***action space***
  - γ is the ***discount factor***
  - $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ (or simply $R : \mathcal{S} \to \mathbb{R}$) is the ***reward function***
- The ***transition distribution***, $P_{S,a}$, where $P_{S,a}(S') = P\left[X^{(t+1)} = S' | X^{(t)} = S\right]$, is controlled by the action, but does not change with time t
- An MDP proceeds as follows:

$$X^{(0)} \xrightarrow{a^{(0)}} X^{(1)} \xrightarrow{a^{(1)}} X^{(2)} \xrightarrow{a^{(2)}} \cdots,$$

with the total payoff

$$R(X^{(0)}, a^{(0)}) + \gamma R(X^{(1)}, a^{(1)}) + \gamma^2 R(X^{(2)}, a^{(2)}) + \cdots$$

(or $R(X^{(0)}) + \gamma R(X^{(1)}) + \gamma^2 R(X^{(2)}) + \cdots$)

Goal of Reinforcement Learning

- Determine the actions over time such that the expected total payoff

$$E_{\{X^{(t)}\}_t}[R(X^{(0)},a^{(0)})+\gamma R(X^{(1)},a^{(1)})+\gamma^2 R(X^{(2)},a^{(2)})+\cdots]$$

  is maximized
- Note that the reward at time t is discounted by a factor of γ
- To make this expectation large, we would like to accrue positive rewards ***as soon as possible***
- E.g., in economic applications where $R(\cdot)$ is the amount of money made, γ has a natural interpretation in terms of the interest rate (where a dollar today is worth more than a dollar tomorrow)

Outline

1. Preliminaries
   - Random Process and Markov Process
2. Markov Decision Process
   - Definitions
   - Bellman Equations
   - Determining the Best Actions: Value and Policy Iteration
   - Learning a Model for MDP
3. Continuous State MDPs

Policy and Value Function

- A ***policy*** is a function $\pi : \mathcal{S} \to \mathcal{A}$
  - We say that we are executing some policy π if, whenever we are in state S, we take action $a = \pi(S)$
- We can also define the ***value function*** for a policy π by

$$V_\pi(S) = E[R(X^{(0)}) + \gamma R(X^{(1)}) + \gamma^2 R(X^{(2)}) + \cdots | X^{(0)} = S, \pi]$$

[1]

[1] We abuse the notation here as π is not a random variable

Bellman Equations

- Given a fixed policy π, the values of V_π satisfy the Bellman equations:

$$V_\pi(S) = R(S) + \gamma \sum_{S' \in \mathcal{S}} P_{S,\pi(S)}(S') V_\pi(S')$$

- In a finite-state MDP ($|\mathcal{S}| < \infty$), Bellman equations can be used to efficiently solve for the values of V_π
  - $|\mathcal{S}|$ linear equations in $|\mathcal{S}|$ variables
  - Complexity: $O(|\mathcal{S}|^3)$

Outline

Determining the Best Actions

- Optimal value function:

$$V^*(S) := \max_{\pi} V_{\pi}(S)$$

- By Bellman equations:

$$V^*(S) = R(S) + \max_{a \in \mathcal{A}} \gamma \sum_{S' \in \mathcal{S}} P_{S,a}(S') V^*(S')$$

- We can also define the optimal policy $\pi^* : \mathcal{S} \rightarrow \mathcal{A}$ as

$$\pi^*(S) := \arg\max_{a \in \mathcal{A}} \gamma \sum_{S' \in \mathcal{S}} P_{S,a}(S') V^*(S')$$

  - How?
- Memoryless property: π^* is the optimal policy for ***all*** states S's
  - This means that we can use the same policy π^* no matter what the initial state of our MDP is

Value Iteration

- To have $\pi^*(S) := \arg\max_{a \in \mathcal{A}} \gamma \sum_{S' \in \mathcal{S}} P_{S,a}(S')V^*(S')$, we need to determine V^* first
- Idea: iteratively improve V

For each state S, initialize $V(S) := 0$;
Repeat until convergence {
  For each state S, update $V(S) := R(S) + \max_{a \in \mathcal{A}} \gamma \sum_{S' \in \mathcal{S}} P_{S,a}(S')V(S')$;
  For each state S, solve $\pi(S) := \arg\max_{a \in \mathcal{A}} \gamma \sum_{S' \in \mathcal{S}} P_{S,a}(S')V(S')$;
}

Policy Iteration

- Recall that given any π, we can solve V_π by the system of Bellman equations
- Idea: iteratively improve π

Initialize π randomly;
Repeat until convergence {
  Solve V_π from the system of Bellman equations;
  For each state S, let $\pi(S) := \arg\max_{a \in \mathcal{A}} \gamma \sum_{S' \in \mathcal{S}} P_{S,a}(S') V_\pi(S')$;
}

Value vs. Policy Iteration

- Which one is better?

Value vs. Policy Iteration

- Which one is better?
- For MDPs with small state spaces, policy iteration is often very fast and converges with very few iterations
- However, for large MDPs, solving for V_π explicitly is time consuming ($O(|\mathbb{S}|^3)$). Value iteration is preferred

Outline

Modeling MDP

- MDP is a Markov process $\{X^{(t)}\}_t$ defined over $(\mathcal{S}, \mathcal{A}, \gamma, R)$
  - The transition distribution, $P_{S,a}$, where $P_{S,a}(S') = P\left[X^{(t+1)} = S' | X^{(t)} = S\right]$, is controlled by the action
- In practice, the $P_{S,a}$ (and/or the reward R) may not be known
- We need to estimate them from ***trials***

$$
\begin{aligned}
&X^{(1,0)} \xrightarrow{a^{(1,0)}} X^{(1,1)} \xrightarrow{a^{(1,1)}} X^{(1,2)} \xrightarrow{a^{(1,2)}} \cdots \\
&X^{(2,0)} \xrightarrow{a^{(2,0)}} X^{(2,1)} \xrightarrow{a^{(2,1)}} X^{(2,2)} \xrightarrow{a^{(2,2)}} \cdots \\
&\cdots
\end{aligned}
$$

- Example estimation of $P_{S,a}$:

$$P_{S,a}(S') = \frac{\#\text{ times the action } a \text{ takes states } s \text{ to state } s'}{\#\text{ times action } a \text{ is taken in states } s}$$

Value Iteration with Unknown Transition Distribution

- Idea: improve both V and the estimated transition probabilities iteratively

Initialize π randomly;
For each state S, initialize $V(S) := 0$;
Repeat until convergence {
  1. Execute π in the MDP for some number of trials;
  2. Update our estimates for $P_{S,a}$ (and R, if applicable) using the experience from trials;
  3. Apply value iteration with the estimated state transition probabilities and rewards to get a new estimated value function V;
  4. Update π with respect to V;
}

- Can speed up if Step 3 starts from the V obtained in the previous iteration of this algorithm

Example: Data Partitioning and Replication for the Cloud Database Systems

- How?

Example: Data Partitioning and Replication for the Cloud Database Systems

- How?
- A state: a particular placement of data chunks on the machines
- An action: splitting hot data chunks, merging cold data chunks, or replicating chunks, etc.
- γ: the cost of data migration
- R: system throughput, which is ***unknown*** due to the changing capabilities of machines
- $P_{S,a}$ is ***unknown*** since the workload is changing (a chunk may become hotter or colder)

Outline

Continuous State MDPs

- TBA