Reinforcement Learning

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

Department of Computer Science,
National Tsing Hua University, Taiwan

NetDB-ML, Spring 2015

Outline

1. Introduction
2. Markov Decision Process
   - Definitions
   - Bellman Equations
   - Determining the Best Actions
3. Single-Agent RL
   - Difference from MDP
   - Model-based Learning
   - Temporal-Difference Learning (Model-Free)
   - Q-Learning (Model-Free)
   - Exploration Policies
   - SARSA (Model-Free)
4. Multi-Agent RL and Game Theory**

Outline

1. Introduction
2. Markov Decision Process
   - Definitions
   - Bellman Equations
   - Determining the Best Actions
3. Single-Agent RL
   - Difference from MDP
   - Model-based Learning
   - Temporal-Difference Learning (Model-Free)
   - Q-Learning (Model-Free)
   - Exploration Policies
   - SARSA (Model-Free)
4. Multi-Agent RL and Game Theory**

Why?

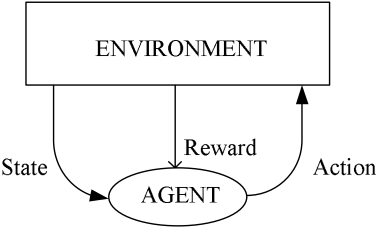
- In supervised learning, we see examples $\boldsymbol{x}^{(t)}$'s that are 1) i.i.d. and 2) given the unambiguous "right" labels $r^{(t)}$'s
- In sequential decision making and control problems, neither holds
- The next example may be the outcome of your "action" to the previous example
    - We've seen how random process helps modeling the dependency
- 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 ($\boldsymbol{x}^{(t)}$)
    - E.g., in the mouse-in-maze problem, we cannot tell the mouse the "correct" path to leave the maze

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
  - In the mouse-in-maze problem, we can define a reward if the mouse has left the maze
- Machine learns the correct from "critics" repeatedly, rather than from correct labels once

Agent Point of View

- The learner is a decision making agent that sees ***states*** of an environment, takes ***actions***, and receives ***reward*** (or penalty) for its actions in trying to solve a problem
  - The state of the environment may be changed due the action
- After a set of trial-and-error runs, it should learn the best ***policy***, which is the sequence of actions that maximizes the total reward



- Assumption: environment does ***not*** change with time

Outline

1. Introduction
2. **Markov Decision Process**
   - Definitions
   - Bellman Equations
   - Determining the Best Actions
3. Single-Agent RL
   - Difference from MDP
   - Model-based Learning
   - Temporal-Difference Learning (Model-Free)
   - Q-Learning (Model-Free)
   - Exploration Policies
   - SARSA (Model-Free)
4. Multi-Agent RL and Game Theory**

Outline

1 Introduction

2 Markov Decision Process
- Definitions
- Bellman Equations
- Determining the Best Actions

3 Single-Agent RL
- Difference from MDP
- Model-based Learning
- Temporal-Difference Learning (Model-Free)
- Q-Learning (Model-Free)
- Exploration Policies
- SARSA (Model-Free)

4 Multi-Agent RL and Game Theory**

Markov Processes Revisited

- A random process is called the ***Markov process*** if it satisfies the ***Markov property***: $P\left[X^{(t_0+t_1)} \leqslant x | X^{(t_0)} = x_0, X^{(t)} = x_t, -\infty < t < t_0\right] = P\left[X^{(t_0+t_1)} \leqslant x | X^{(t_0)} = x_0\right]$

	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

Markov Decision Process

- A Markov decision process $\{X^{(t)}\}_t$ defined over $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ is a Markov process, where
  - $\mathcal{S}$ is the state space
  - $\mathcal{A}$ is the ***action space***
  - $P(X^{(t+1)} = S' | X^{(t)} = S; a)$ (or simply $P(S'|S;a)$) is the ***transition distribution*** that is controlled by the action, but does not change with time t
  - $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ (or simply $R : \mathcal{S}' \to \mathbb{R}$) is the deterministic (expected) ***reward function***
  - $\gamma \in \mathbb{R}$ is the ***discount factor***
- 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 total payoff

$$R(X^{(0)}, a^{(0)}, X^{(1)}) + \gamma R(X^{(1)}, a^{(1)}, X^{(2)}) + \gamma^2 R(X^{(2)}, a^{(2)}, X^{(3)}) + \cdots$$

(or $R(X^{(1)}) + \gamma R(X^{(2)}) + \gamma^2 R(X^{(3)}) + \cdots$)

Goal

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

$$E_{\{X^{(t)}\}_t}[R(X^{(0)},a^{(0)},X^{(1)})+\gamma R(X^{(1)},a^{(1)},X^{(2)})+\gamma^2 R(X^{(2)},a^{(2)},X^{(3)})+\cdots$$

  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***
  - Because, e.g., the agent may be powered by battery of limited capacity
- 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. Introduction
2. Markov Decision Process
    - Definitions
    - Bellman Equations
    - Determining the Best Actions
3. Single-Agent RL
    - Difference from MDP
    - Model-based Learning
    - Temporal-Difference Learning (Model-Free)
    - Q-Learning (Model-Free)
    - Exploration Policies
    - SARSA (Model-Free)
4. Multi-Agent RL and Game Theory**

Policy and Value Function

- A ***policy*** is a function $\pi : \mathcal{S} \rightarrow \mathcal{A}$
  - 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

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

Bellman Equations

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

$$V_\pi(S) = \sum_{S' \in \mathcal{S}} P(S'|S;\pi(S)) \left[R(S,\pi(S),S') + \gamma V_\pi(S')\right]$$

  for all S'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
  - Time complexity: $O(|\mathcal{S}|^3)$

Outline

1. Introduction
2. Markov Decision Process
   - Definitions
   - Bellman Equations
   - Determining the Best Actions
3. Single-Agent RL
   - Difference from MDP
   - Model-based Learning
   - Temporal-Difference Learning (Model-Free)
   - Q-Learning (Model-Free)
   - Exploration Policies
   - SARSA (Model-Free)
4. Multi-Agent RL and Game Theory**

Determining the Best Actions

- Optimal value function:

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

- By Bellman equations:

$$V^*(S) = \max_{a\in\mathcal{A}} \sum_{S'\in\mathcal{S}} P(S'|S;a)\left[R(S,a,S') + \gamma V^*(S')\right]$$

- Define the optimal policy $\pi^* : \mathcal{S} \to \mathcal{A}$ as

$$\pi^*(S) := \arg\max_{a\in\mathcal{A}} \sum_{S'\in\mathcal{S}} P(S'|S;a)\left[R(S,a,S') + \gamma V^*(S')\right]$$

- Memoryless property: π^* is independent with $X^{(0)}$
  - We can use the same policy π^* no matter what the initial state of our MDP is to maximize value

Value Iteration (1/2)

- $\pi^*(S) := \arg\max_{a \in \mathcal{A}} \gamma \sum_{S' \in \mathcal{S}} P(S'|S;\pi(S)) V^*(S')$ can be easily solved in $O(|\mathcal{S}||\mathcal{A}|)$ time, if we already have $V^*(S)$'s
  - $O(|\mathcal{S}|^2|\mathcal{A}|)$ for the optimal policy for ***all*** states S's
- Idea: guess $V(S)$'s first, and iteratively improve them

Value Iteration (2/2)

Input: MDP $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$
Output: $\pi(S)$'s for all S's

For each state S, initialize $V(S) \leftarrow 0$;
repeat
  foreach S **do**
    $V(S) \leftarrow \max_{a \in \mathcal{A}} \sum_{S' \in \mathcal{S}} P(S'|S;a)\,[R(S,a,S') + \gamma V(S')]$;
  end
until $V(S)$*'s converge*;
foreach S **do**
  $\pi(S) \leftarrow \arg\max_{a \in \mathcal{A}} \sum_{S' \in \mathcal{S}} P(S'|S;a)\,[R(S,a,S') + \gamma V(S')]$;
end

Algorithm 1: Value Iteration.

Policy Iteration (1/2)

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

Policy Iteration (2/2)

Input: MDP $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$
Output: $\pi(S)$'s for all S's

For each state S, initialize $\pi(S)$ randomly;
repeat
  Solve $V(S)$'s from the system of Bellman equations;
  foreach S **do**
    $\pi(S) \leftarrow \arg\max_{a \in \mathcal{A}} \sum_{S' \in \mathcal{S}} P(S'|S;a)\,[R(S,a,S') + \gamma V(S')]$;
  end
until $\pi(S)$*'s converge*;

Algorithm 2: Policy Iteration.

Value vs. Policy Iteration

- Which one is better?

Value vs. Policy Iteration

- Which one is better?
- Time complexity for each iteration:
  - Value iteration: $O(|\mathcal{S}|^2|\mathcal{A}|)$
  - Policy iteration: $O(|\mathcal{S}|^2|\mathcal{A}|+|\mathcal{S}|^3)$
- 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 ($O(|\mathcal{S}|^3)$). Value iteration is preferred

Outline

1. Introduction
2. Markov Decision Process
   - Definitions
   - Bellman Equations
   - Determining the Best Actions
3. **Single-Agent RL**
   - Difference from MDP
   - Model-based Learning
   - Temporal-Difference Learning (Model-Free)
   - Q-Learning (Model-Free)
   - Exploration Policies
   - SARSA (Model-Free)
4. Multi-Agent RL and Game Theory**

Outline

1. Introduction
2. Markov Decision Process
   - Definitions
   - Bellman Equations
   - Determining the Best Actions
3. Single-Agent RL
   - Difference from MDP
   - Model-based Learning
   - Temporal-Difference Learning (Model-Free)
   - Q-Learning (Model-Free)
   - Exploration Policies
   - SARSA (Model-Free)
4. Multi-Agent RL and Game Theory**

Episode

- In some applications, there could be terminal/absorbing states
  - An absorbing state transit to itself with probability 1
- E.g., in the mouse-in-maze problem, the "leaving the maze" is an absorbing state
- A sequence of actions from starting to terminal states is called an ***episode*** or ***trial***
- The agent can perform many trails, and the goal would be to learn the best policy for an episode

Example: k-Armed Bandit

- Action: pull a lever
- Goal: maximizes the total reward

The MDP Point of View

- A single state MDP
  - As pulling a lever (an action) does not change anything in the bandit machine
- However,
  - The rewards received when action a takes state S to state S' may be generated by following some unknown distribution
  - The expected reward $R(S, a, S')$ is unknown
- For the robot walking problem, the state transition distribution $P(S'|S; a)$ is unknown

Example: Data Partitioning and Replication for the Cloud Database Systems

- Each state represents (current workload, a particular placement of data chunks) on the machines
- An action: splitting hot data chunks, merging cold data chunks, or replicating chunks, etc.
- R: system throughput, which is ***unknown***
  - It is generally hard to predict the performance given a particular workload and data consolidation
- $P(S'|S;a)$ is ***unknown*** too since the workload from clients is unpredictable

Exploration vs. Exploitation

- What would you do if you were the agent?

Exploration vs. Exploitation

- What would you do if you were the agent?
- To perform actions to ***explore*** $R(S,a,S')$ and $P(S'|S;a)$ first
  - Learn $R(S,a,S')$ and $P(S'|S;a)$ from their samples "queried" from the environment
- Then, to perform actions to ***exploit*** the learned $P(r|S;a)$ and $P(S'|S;a)$ to maximize the total rewards
  - The best policy can be computed locally by the agent itself (thanks to MDP)

Outline

1. Introduction
2. Markov Decision Process
   - Definitions
   - Bellman Equations
   - Determining the Best Actions
3. **Single-Agent RL**
   - Difference from MDP
   - **Model-based Learning**
   - Temporal-Difference Learning (Model-Free)
   - Q-Learning (Model-Free)
   - Exploration Policies
   - SARSA (Model-Free)
4. Multi-Agent RL and Game Theory**

How to Learn $R(S,a,S')$ and $P(S'|S;a)$?

How to Learn $R(S,a,S')$ and $P(S'|S;a)$?

- Why not use their sample means?
- Given the trials

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

- Example estimation of (discrete) $P(S'|S;a)$:
  - $\widetilde{P}(S'|S;a) = \dfrac{\#\text{ times the action } a \text{ takes states } s \text{ to state } s'}{\#\text{ times action } a \text{ is taken in states } s}$

Learning $R(S, a, S')$

- E.g., average all reward values, r's, received when action a takes state S to state S'
- But this requires memory to store ***all*** r's
- Cheaper solution?

The Moving Average

- The ***exponential moving average***:
  - $\bar{x}_n = \frac{x^{(n)}+(1-\eta)x^{(n-1)}+(1-\eta)^2x^{(n-2)}+\cdots}{1+(1-\eta)+(1-\eta)^2+\cdots}$, where η is a small constant
  - Recent samples are more important
  - $\bar{x}_n = \eta x^{(n)} + (1-\eta)\bar{x}_{n-1}$ [Proof]
- To learn $R(S,a,S')$, keep the current estimator $\widetilde{R}(S,a,S')$ and η
  - Every time when a new value r is seen, update $\widetilde{R}(S,a,S') = (1-\eta)\widetilde{R}(S,a,S') + \eta r = \widetilde{R}(S,a,S') + \eta(r - \widetilde{R}(S,a,S'))$
- η is similar to the learning rate we've seen in the perception classifier

Problems of Model-based Learning

- $R(S, a, S')$ and $P(S'|S; a)$ can only be estimated if we have samples
- If $P(S'|S; a)$ is small, there may be too few samples to have a good estimate
- Low $P(S'|S; a)$ may also lead to a poor estimate of $R(S, a, S')$

Outline

1. Introduction
2. Markov Decision Process
   - Definitions
   - Bellman Equations
   - Determining the Best Actions
3. **Single-Agent RL**
   - Difference from MDP
   - Model-based Learning
   - **Temporal-Difference Learning (Model-Free)**
   - Q-Learning (Model-Free)
   - Exploration Policies
   - SARSA (Model-Free)
4. Multi-Agent RL and Game Theory**

Temporal-Difference Learning (1/2)

- We explore/estimate $R(S, a, S')$ and $P(S'|S; a)$ in order to compute $V^*(S)$
- Why not estimate $V^*(S)$ directly?

Temporal-Difference Learning (1/2)

- We explore/estimate $R(S,a,S')$ and $P(S'|S;a)$ in order to compute $V^*(S)$
- Why not estimate $V^*(S)$ directly?
- Recall that $V^*(S)$ is an expectation, let's compute the moving average estimator $\widetilde{V}^*(S)$ again
- Given an exploration policy π, we can compute a sample of $V^*(S)$ by $sample = R(S,\pi(S),S') + \gamma\widetilde{V}^*(S')$ after each action
    - Based on the Bellman's equation
    - The unknown $V^*(S')$ is replaced by $\widetilde{V}^*(S')$
- So, we can update the estimator $\widetilde{V}^*(S) = \widetilde{V}^*(S) + \eta(sample - \widetilde{V}^*(S))$ after each action

Temporal-Difference Learning (2/2)

Input: $\mathcal{S}$, $\mathcal{A}$, and γ of an MDP, a policy π, and η
Output: $V^*(S)$'s for all S's

For each state S, initialize $V_\pi(S)$ arbitrarily;
foreach *episode* **do**
  Initialize S;
  repeat
    Choose action $a \leftarrow \pi(S)$;
    Take action a, observe S' and reward $R(S,a,S')$;
    $V_\pi(S) \leftarrow V_\pi(S) + \eta\,[(R(S,a,S') + \gamma V_\pi(S')) - V_\pi(S)]$;
    $S \leftarrow S'$;
  until S *is terminal state*;
end

Algorithm 3: Temporal-Difference (TD) Learning.

Problem of TD Learning

- The given (exploration) policy π is ***not*** the optimal (exploitation) policy π^* that generates $V*(S)$'s
- We need to solve $\pi^*(S)$'s from $V*(S)$'s, as did in the value iteration algorithm
- But, without knowing/estimating $R(S,a,S')$ and $P(S'|S;a)$, we cannot do that now!

Outline

1. Introduction
2. Markov Decision Process
   - Definitions
   - Bellman Equations
   - Determining the Best Actions
3. Single-Agent RL
   - Difference from MDP
   - Model-based Learning
   - Temporal-Difference Learning (Model-Free)
   - Q-Learning (Model-Free)
   - Exploration Policies
   - SARSA (Model-Free)
4. Multi-Agent RL and Game Theory**

The Q Function

- We need something that
  - we can estimate in spite of low transition probability; and
  - the estimated value helps computing $V^*(S)$'s ***and*** π^*
- Define a function $Q^* : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ by letting $Q^*(S,a)$ be the maximum expected cumulative reward that an agent will receive when starting from state S and action a and then obeying the optimal policy afterward
- We have $V^*(S) = \max_a Q^*(S,a)$
- Similar to the Bellman's equations

$$V^*(S) = \sum_{S' \in \mathcal{S}} P(S'|S;\pi^*(S)) \left[R(S,\pi^*(S),S') + \gamma V^*(S')\right],$$

now we have

$$\begin{aligned} Q^*(S,a) &= \sum_{S' \in \mathcal{S}} P(S'|S;a) \left[R(S,a,S') + \gamma V^*(S')\right] \\ &= \sum_{S' \in \mathcal{S}} P(S'|S;a) \left[R(S,a,S') + \gamma \max_a Q^*(S,a)\right] \end{aligned}$$

Getting the Best Policy

- $Q^*(S,a)$ is an expectation, so we can estimate it using the moving average, as did in the TD learning
- Even better, we can derive $\pi^*(S)$'s ***directly*** from $Q^*(S,a)$'s
  - By definition of Q^*, we have $\pi^*(S) = \arg\max_a Q^*(S,a)$
  - No need for $R(S,a,S')$ and $P(S'|S;a)$

Q-Learning (1/2)

Input: $\mathcal{S}$, $\mathcal{A}$, and γ of an MDP, and η
Output: $\pi^*(S)$'s for all S's

For each state S and a, initialize $Q(S,a)$ arbitrarily;
foreach *episode* **do**
  Initialize S;
  repeat
    Choose action a using ***some exploration policy***;
    Take action a, observe S' and reward $R(S,a,S')$;
    $Q(S,a) \leftarrow Q(S,a) + \eta\,[(R(S,a,S') + \gamma \max_b Q(S',b)) - Q(S,a)]$;
    $S \leftarrow S'$;
  until S *is terminal state*;
end
foreach S **do**
  $\pi^*(S) = \arg\max_{a'} Q(S,a')$;
end

Algorithm 4: Q-learning.

Q-Learning (2/2)

- Amazing results: Q-learning converges to the optimal policy!
  - If you explore enough
  - If η is small enough and does not decrease too quickly
  - Does ***not*** matter how you select exploration actions!
- The exploration policy (a's) is ***not*** the exploitation policy (b's) used to update $Q(S,a)$'s
  - Q-learning is an ***off-policy*** RL method

Outline

1. Introduction
2. Markov Decision Process
   - Definitions
   - Bellman Equations
   - Determining the Best Actions
3. Single-Agent RL
   - Difference from MDP
   - Model-based Learning
   - Temporal-Difference Learning (Model-Free)
   - Q-Learning (Model-Free)
   - Exploration Policies
   - SARSA (Model-Free)
4. Multi-Agent RL and Game Theory**

ϵ-Greedy Policy

- To actually maximize total rewards, an agent needs to gradually move from exploration to exploitation
  - How?

ϵ-Greedy Policy

- To actually maximize total rewards, an agent needs to gradually move from exploration to exploitation
    - How?
- Simplest: the ϵ-greedy strategy
- At every time step, flip a coin
    - With probability ϵ, act randomly (explore)
    - With probability $(1-\epsilon)$, compute/update the best policy and act accordingly (exploit)
- Gradually decrease ϵ over time
- Any other idea?

Softmax Policy

- Could we have a "soft" policy between the two extremes (exploration & exploitation) at each time step?

Softmax Policy

- Could we have a "soft" policy between the two extremes (exploration & exploitation) at each time step?
- Idea: perform an action a more often if $Q(S,a)$ is larger
- Choose a based on the softmax function that converts $Q(S,a)$'s to probabilities:

$$P(a|S) = \frac{\exp\left[Q(S,a)/t\right]}{\sum_{a'}\left[\exp Q(S,a')/t\right]}$$

- t starts at a large value (exploration), and decreases over time

Exploration Function

- Idea: to explore areas with fewest samples
  - E.g., in Q-learning, we can define an exploration function $f(q,n) = q + k/n$, where q is an estimated Q-value, n is the number of samples for the estimate, and k is some positive constant
- Instead of the update rule:

$$Q(S,a) \leftarrow Q(S,a) + \eta \left[(R(S,a,S') + \gamma \max_b Q(S',b)) - Q(S,a) \right]$$

- Use f when updating $Q(S,a)$:

$$Q(S,a) \leftarrow Q(S,a) + \eta \left\{ \left[R(S,a,S') + \gamma \max_b f(Q(S',b), N(S',b)) \right] - Q(S,a) \right\}$$

Outline

1. Introduction
2. Markov Decision Process
   - Definitions
   - Bellman Equations
   - Determining the Best Actions
3. Single-Agent RL
   - Difference from MDP
   - Model-based Learning
   - Temporal-Difference Learning (Model-Free)
   - Q-Learning (Model-Free)
   - Exploration Policies
   - SARSA (Model-Free)
4. Multi-Agent RL and Game Theory**

SARSA (1/2)

- There is an ***on-policy*** variant of Q-learning, called SARSA (State-Action-Reward-State-Action)
  - That is, the exploration policy (a's) is used as the exploitation policy (b's) when updating $Q(S,a)$'s
- Still converges with probability 1 to the optimal policy, if a GLIE (Greedy in the Limit with Infinite Exploration) policy is employed:
  - All (S,a) pairs are visited an infinite number of times
  - The policy converges (in the limit) to the exploitation/greedy policy
  - E.g., ϵ-greedy policy

SARSA (2/2)

Input: $\mathcal{S}$, $\mathcal{A}$, and γ of an MDP, and η
Output: $\pi^*(S)$'s for all S's

For each state S and a, initialize $Q(S,a)$ arbitrarily;
foreach *episode* **do**
    Initialize S;
    repeat
        Choose action a using ***some GLIE policy*** derived from Q;
        Take action a, observe S' and reward $R(S,a,S')$;
        Choose action b using ***the same GLIE policy***;
        $Q(S,a) \leftarrow Q(S,a) + \eta\,[(R(S,a,S') + \gamma Q(S',b)) - Q(S,a)]$;
        $S \leftarrow S'$;
    until S *is terminal state*;
end
foreach S **do**
    $\pi^*(S) = \arg\max_{a'} Q(S,a')$;
end

Algorithm 5: SARSA algorithm.

Q-Learning vs. SARSA

- Which one is better? (Why on-policy algorithms?)

Q-Learning vs. SARSA

- Which one is better? (Why on-policy algorithms?)
- Q-Learning tends to converge a little slower, but has the capability to continue learning while changing the exploration policy
- SARSA has the capability to ***avoid the mistakes due to exploration***
- See the live demo of the mouse-in-maze problem: `https://studywolf.wordpress.com/2013/07/01/reinforcement-learning-sarsa-vs-q-learning/`

Outline

1. Introduction
2. Markov Decision Process
   - Definitions
   - Bellman Equations
   - Determining the Best Actions
3. Single-Agent RL
   - Difference from MDP
   - Model-based Learning
   - Temporal-Difference Learning (Model-Free)
   - Q-Learning (Model-Free)
   - Exploration Policies
   - SARSA (Model-Free)
4. **Multi-Agent RL and Game Theory****

Multi-Agent RL and Game Theory

- TBA