

Reinforcement Learning

- Ole-Christoffer Granmo

University of Agder

ole.granmo@uia.no

September 5, 2007

Outline

- Reinforcement Learning
 - The Tsetlin Automaton
 - Game Theory and Automata Games
 - Applications
 - Exercise

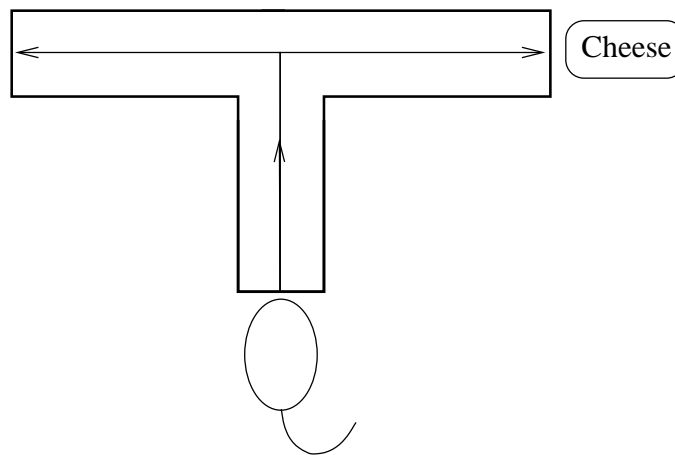
Part IV:

Reinforcement Learning

Reinforcement Learning

- **Reinforcement learning** involves an agent that *explores* an *environment*
- **The agent:** *perceives* its current state and takes *actions*
- **The environment:** provides a *reward* or a *penalty*
- Reinforcement learning algorithms attempt to find a policy for maximizing cumulative reward for the agent over the course of the problem

The T-maze Problem



- A hungry rat is placed at the lower end of the middle limb of a T-maze
- The rat can move along the limb and turn to the right or to the left
- Food is kept at the end of the right arm with probability 0.7 and at the end of the left arm with probability 0.3
 - How does the rat behave over successive trials?
 - Can the rat behavior be modeled mathematically?

Learning Automata Characteristics

- Learning Automata (LA) are adaptive decision making devices that can operate in:
 - **Unknown Environments:** They do not need information about the effect of their actions at start of operation
 - **Random Environments:** An action does not necessarily produce the same response each time it is performed
- A powerful property of LA is that they progressively improve their performance by the means of a learning process
 - combine rapid and accurate convergence with low computational complexity

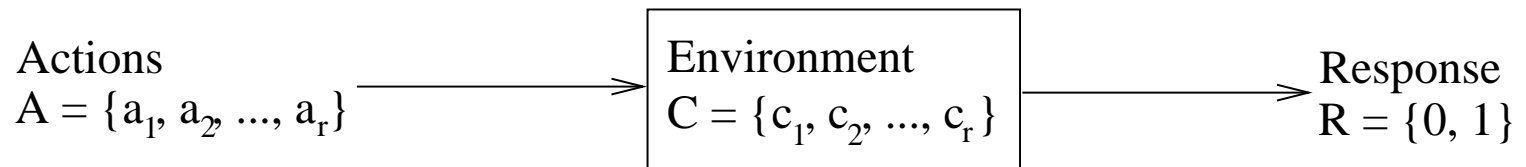
Learning Automata Make Decisions in an Environment

- Example decisions:
 - Which of two alternate routes to choose in network routing
 - Choosing between air travel and car travel to a neighbor city
- Outcome of choice random:
 - The time to reach a destination using one route is assumed to be a random variable depending on traffic conditions

Learning Automata Applications

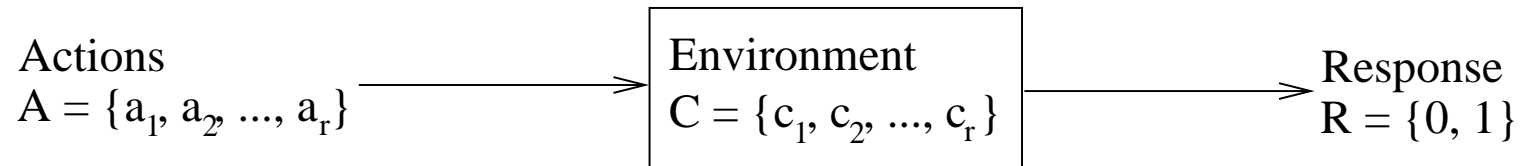
- Potential Learning Automata applications include:
 - Process control
 - Pattern recognition
 - Control of service activity
 - Task scheduling
 - Optimization problems
 - Image processing
 - Diagnosis
 - Computer vision
 - Concept learning
 - Routing and bandwidth allocation in computer communication networks

The Environment



- Environment \longrightarrow large class of general unknown media in which an automaton or a group of automata can operate
 - **Input:** Action from Set of Actions
 - **Output:** *Reward* or *Penalty*
 - **Penalty Probabilities:** When performing an action i , there is a certain probability that the *Environment* responds with a *Penalty*
 - * $P(\text{Penalty} | \text{Action} = a_i) = c_i, 1 \leq i \leq r$
 - **Remark:** If the Environment does not respond with a *Penalty*, it responds with a *Reward* instead

Static and Dynamic Environments



- An environment can be
 - **Static:** Penalty probabilities do not change
 - **Dynamic:** Penalty probabilities change

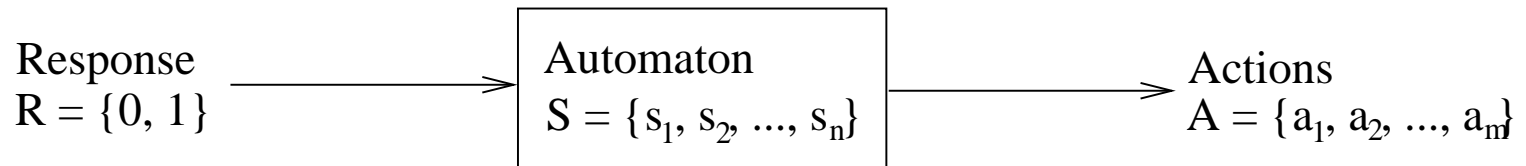
Simulation of Environment

Simulation of Two Action Environment:

```
class Environment:
    def __init__(self, c_1, c_2):
        self.c_1 = c_1
        self.c_2 = c_2

    def penalty(self, action):
        if action == 1:
            if random.random() <= self.c_1:
                return True
            else:
                return False
        elif action == 2:
            if random.random() <= self.c_2:
                return True
            else:
                return False
```

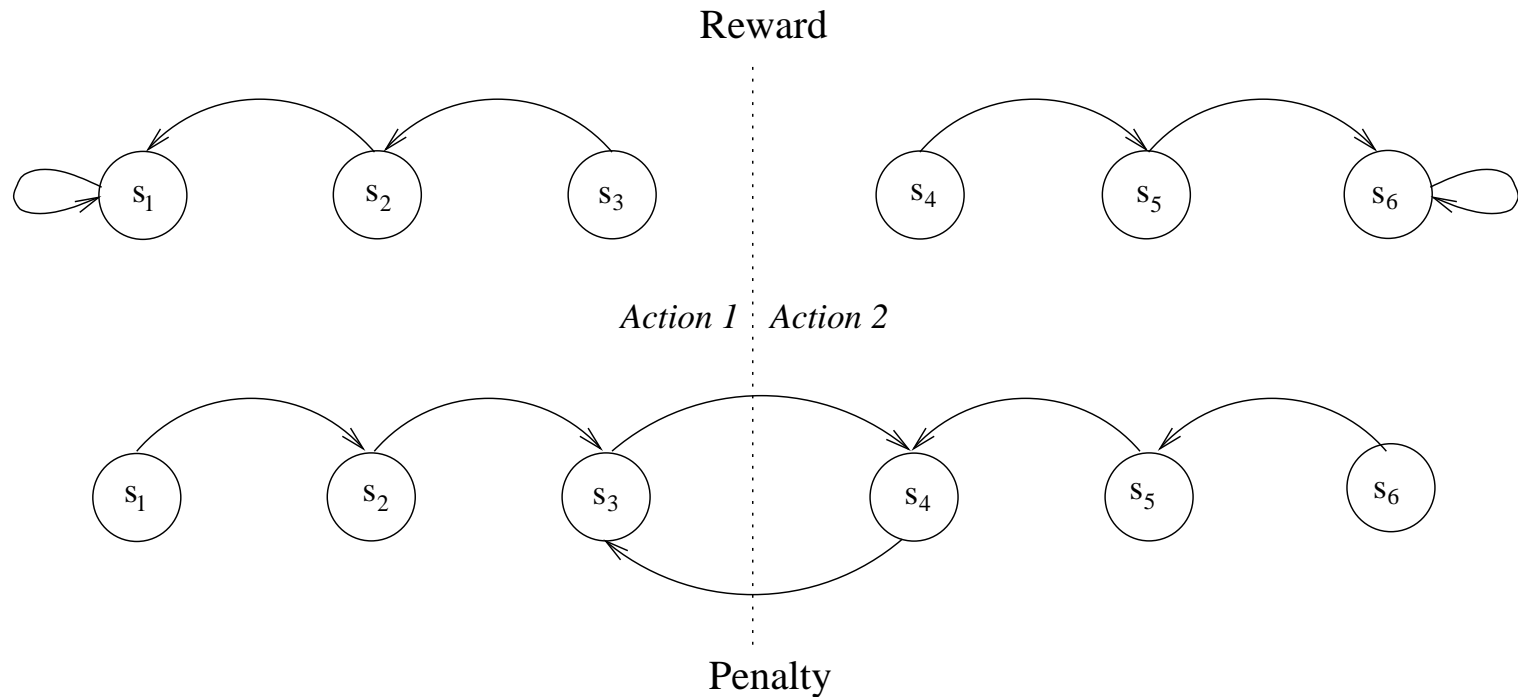
The Operation of an Automaton



- An automaton remembers which actions are “good” by maintaining a state $s_t \in \{s_1, \dots, s_n\}$
- **Operation** – an automaton:
 1. Selects and *outputs* an action based on its present state
 2. Takes a response from the environment as *input*
 3. Changes its *state* based on (a) the *response* and (b) the *action* performed
- An automaton can be said to *learn* if it reduces the number of penalties received as a result of interacting with the environment

Two-Action Tsetlin Automaton I

Tsetlin Automaton with 3 states per action:



- A Tsetlin-automaton can learn the optimal action if the lowest penalty probability is less than 0.5
- Number of automaton states determines “learning accuracy” and “learning speed”

Two-Action Tsetlin Automaton II

```
class Tsetlin:
    def __init__(self, n):
        # 'n' is the number of states per action
        self.n = n
        # Initial state selected randomly
        self.state = random.choice([self.n, self.n+1])

    def reward(self):
        if self.state <= self.n and self.state > 1:
            self.state -= 1
        elif self.state > self.n and self.state < 2*self.n:
            self.state += 1

    def penalize(self):
        if self.state <= self.n:
            self.state += 1
        elif self.state > self.n:
            self.state -= 1

    def makeDecision(self):
        if self.state <= self.n:
            return 1
        else:
            return 2
```

Learning Automata and Games I

- **Decentralization** is a common and often necessary feature of complex natural and man-made systems
 - Arises from the reality that the complete information exchange needed for centralized decision making may not be feasible
- **Formidable Problem:** Coordination of decentralized decision makers

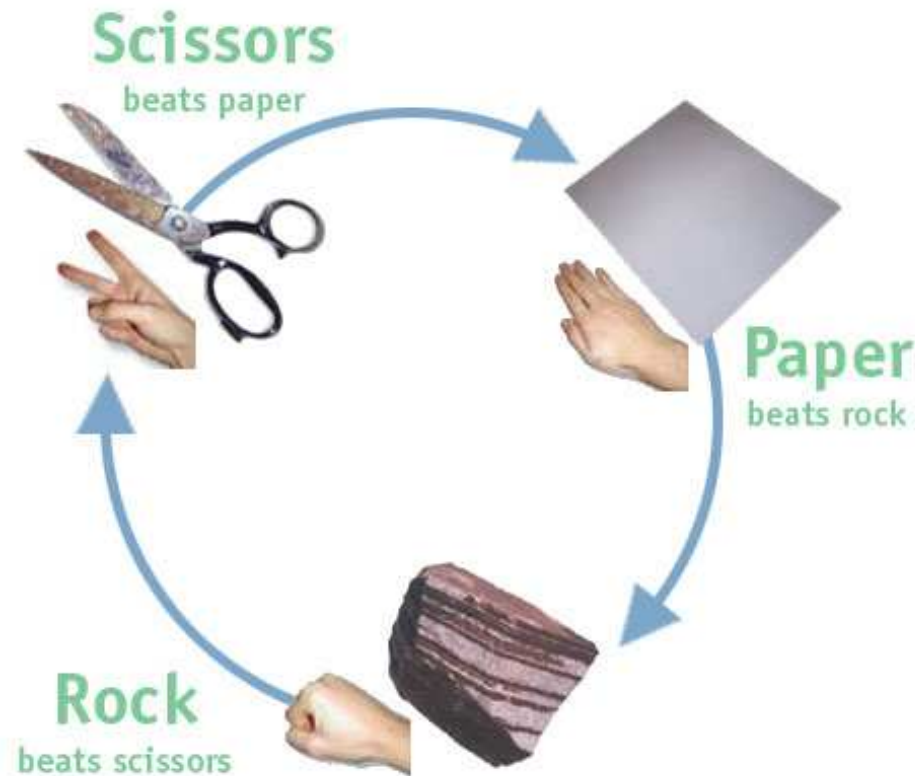
Example: QoS Control in Sensor Networks

- Consider a basic sensor network that consists of an *unknown* number of *sensors* and a single *base station*
 - Each sensor can be *powered-down* or *powered-up*
 - The base station receives packets from powered-up sensors
- **Performance goals:**
 - Maximize the life time of the sensor network by having sensors periodically power-down to conserve their battery energy
 - Have sufficient sensors powered-up and sending packets to the base stations so that enough data is collected
- **Problem:** How can the base station control the sensors so that only Q sensors are powered up when:
 1. The base station is only able to broadcast information, and cannot address the sensors individually
 2. The base station does not know the number of sensors in the network

Learning Automata and Games II

- Decentralized decision making can be formulated as a game
- **Game:** Metaphor for a much wider range of human interactions
 - Outcomes depend on the *interactive strategies* of two or more persons
 - Persons have *opposed* or at best *mixed motives*

Zero-Sum Games — Rock, Paper, Scissors



- In zero-sum games the total benefit to all players in the game, for every combination of strategies, always adds to zero
- **Remark:** A player benefits only at the expense of others

Non-Zero-Sum Games — Prisoner's Dilemma I

	B Remains Silent	B Betrays
A Remains Silent	Both six months	A ten years; B free
A Betrays	A free; B ten years	Both two years

The classical prisoner's dilemma (PD) can be described as follows:

- Two suspects, A and B, are arrested by the police
- The police have insufficient evidence for a conviction, and, having separated both prisoners, visit each of them to offer the same deal:
 1. If one testifies for the prosecution against the other and the other remains silent, the betrayer goes free and the silent accomplice receives the full 10-year sentence
 2. If both stay silent, the police can sentence both prisoners to only six months in jail for a minor charge
 3. If each betrays the other, each will receive a two-year sentence
- Each prisoner must make the choice of whether to betray the other or to remain silent
- However, neither prisoner knows for sure what choice the other prisoner will make

Non-Zero-Sum Games — Prisoner's Dilemma II

Example - Cycling Races:

- Consider two cyclists halfway in a race, with the peloton (larger group) at great distance behind them
- The two cyclists often work together (mutual cooperation) by sharing the tough load of the front position, where there is no shelter from the wind
- If neither of the cyclists makes an effort to stay ahead, the peloton will soon catch up (mutual defection)
- An often-seen scenario is one cyclist doing the hard work alone (cooperating), keeping the two ahead of the peloton. In the end, this will likely lead to a victory for the second cyclist (defecting) who has an easy ride in the first cyclist's slipstream.

Coordination Games — The Goore Game

- Imagine a large room containing N cubicles and a raised platform
- One person (voter) sits in each cubicle and a Referee stands on the platform
- The Referee conducts a series of voting rounds as follows:
 1. On each round the voters vote “Yes” or “No” simultaneously and independently
 2. The Referee counts the number λ of “Yes” votes
 3. The Referee has a uni-modal performance criterion $G(\lambda)$, which is optimized when the number of “Yes” votes is exactly λ^*
 4. The current voting round ends with the Referee awarding a dollar with probability $G(\lambda)$ and assessing a dollar with probability $1 - G(\lambda)$ to every voter independently
 5. On the basis of their individual gains and losses, the voters then decide, again independently, how to cast their votes on the next round

Normal Form Representation of Games

Normal Form Representation of Prisoner's Dilemma:

	B Strategy 1	B Strategy 2
A Strategy 1	3, 3	0, 5
A Strategy 2	5, 0	1, 1

- The **normal form representation of a game** is a matrix which shows *players*, *strategies*, and *payoffs*
- In the example, there are two players:
 - One chooses the *row* and the other chooses the *column*
 - Each player has two strategies, which are specified by the number of rows and the number of columns
- The payoffs are provided in the interior
 - The first number is the payoff received by the row player
 - The second is the payoff for the column player
- **Example:** Suppose that *Player A* plays *Strategy 2* and that *Player B* plays *Strategy 1*. Then *Player A* gets 5, and *Player B* gets 0.

Some Definitions

- **Dominant Strategy:** Let an individual player in a game evaluate separately each of the strategy combinations he may face, and, for each combination, choose from his own strategies the one that gives the best payoff. If the same strategy is chosen for each of the different combinations of strategies the player might face, that strategy is called a "dominant strategy" for that player in that game
- **Nash Equilibrium:** If there is a set of strategies with the property that no player can benefit by changing her strategy while the other players keep their strategies unchanged, then that set of strategies and the corresponding payoffs constitute the Nash Equilibrium
- **Pareto Optimality:** An outcome of a game is Pareto optimal if there is no other outcome that makes every player at least as well off and at least one player strictly better off.
 - *Remark:* A Pareto Optimal outcome cannot be improved upon without hurting at least one player

Nash Equilibrium and Pareto Optimality — Example I

Normal Form Representation of Prisoner's Dilemma:

	B Strategy 1	B Strategy 2
A Strategy 1	3, 3	0, 5
A Strategy 2	5, 0	1, 1

- Find dominant strategies in the above game!
- Find the *Nash Equilibria* of the above game!
- Find the *Pareto Optimal* outcomes of the above game!

Nash Equilibrium and Pareto Optimality — Example II

Normal Form Representation of Two-Player Goore Game:

	B No	B Yes
A No	0.3, 0.3	0.7, 0.7
A Yes	0.7, 0.7	0.5, 0.5

- Find dominant strategies in the above game!
- Find the *Nash Equilibria* of the above game!
- Find the *Pareto Optimal* outcomes of the above game!

Application

- The interaction between an attacker and the administrator of a computer network can be seen as a *two-player stochastic game*
- A strategy pair causes the computer network to move from one state to another in a probabilistic manner
 - When a network operates normally, the attacker may for instance choose between {*Inactivity, Attack httpd, Attack ftpd*}
 - The administrator are mainly taking preventive or restorative actions: {*“Remove compromised account restart httpd”, “Install sniffer detector”, “Remove compromised account restart ftpd”*}
- **Remark:** It turns out that the Nash Equilibria of such game models of security can give an administrator an idea of potential attacker strategies, as well as a plan for what to do in the event of an attack

Exercise

Download and install **Python** from <http://www.python.org>

Implement the following program:

1. Create 3 Tsetlin Automata with actions “No” and “Yes”
2. Count the number of Tsetlin Automata that outputs a “Yes”-action
3. If the number of “Yes”-actions is:
 - **0:** Give each Automaton a reward with probability 0.3, otherwise give it a penalty
 - **1:** Give each Automaton a reward with probability 0.5, otherwise give it a penalty
 - **2:** Give each Automaton a reward with probability 0.7, otherwise give it a penalty
 - **3:** Give each Automaton a reward with probability 0.4, otherwise give it a penalty
4. Goto 2

Remark: Generate the rewards independently for each automaton