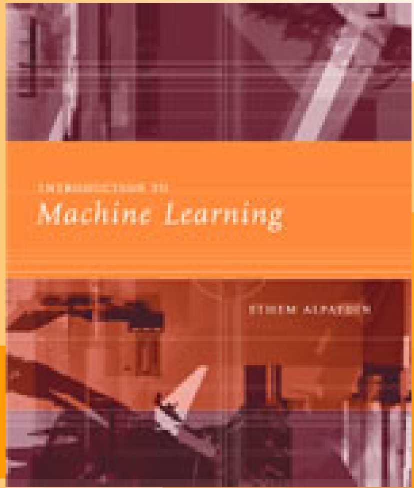
Lecture Slides for

INTRODUCTION TO

Machine Learning



ETHEM ALPAYDIN

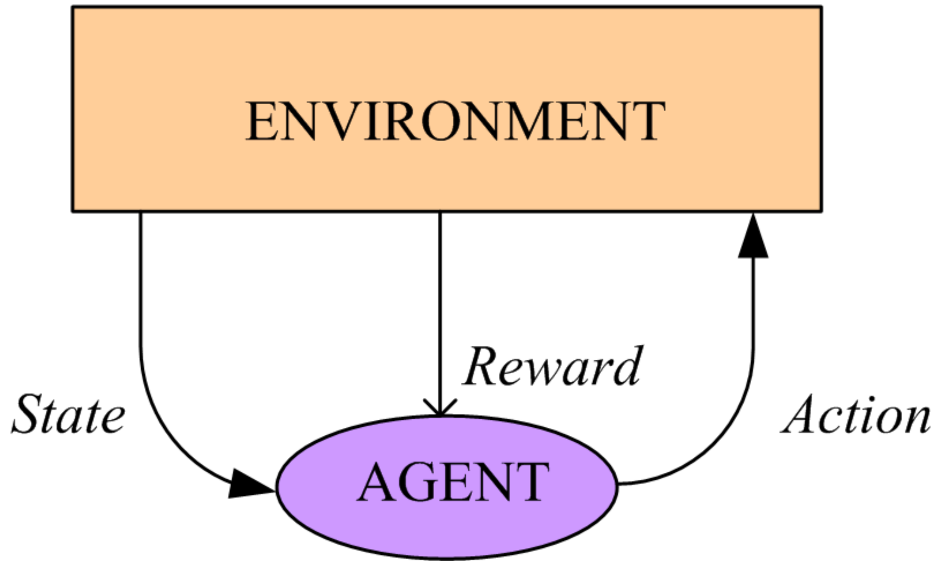
alpaydin@boun.edu.tr
http://www.cmpe.boun.edu.tr/~ethem/i2ml

CHAPTER 16:

Reinforcement Learning

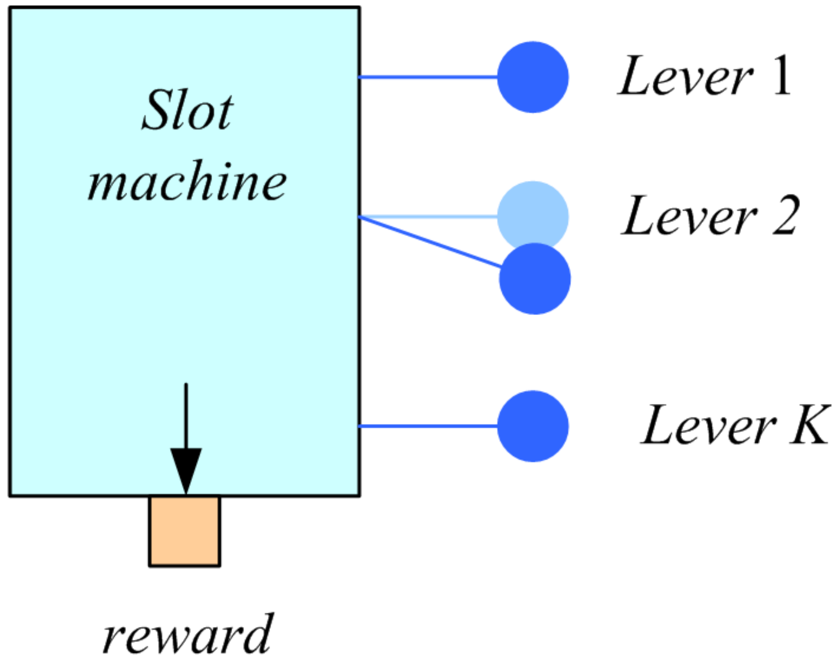
Introduction

- Game-playing: Sequence of moves to win a game
- Robot in a maze: Sequence of actions to find a goal
- Agent has a state in an environment, takes an action and sometimes receives reward and the state changes
- Credit-assignment
- Learn a policy



Single State: K-armed Bandit

- Among K levers, choose the one that pays best
  - $Q(a)$: value of action a
  - Reward is r_a
  - Set $Q(a) = r_a$
  - Choose a^* if
    - $Q(a^*)=\max_a Q(a)$



- Rewards stochastic (keep an *expected* reward):

$$Q_{t+1}(a) \leftarrow Q_t(a) + \eta\left[r_{t+1}(a) - Q_t(a)\right]$$

Elements of RL (Markov Decision Processes)

- s_t : State of agent at time t
- a_t: Action taken at time t
- In s_t, action a_t is taken, clock ticks and reward r_{t+1} is received and state changes to s_{t+1}
- Next state prob: $P(s_{t+1} \mid s_t, a_t)$
- Reward prob: $p(r_{t+1} \mid s_t, a_t)$
- Initial state(s), goal state(s)
- Episode (trial) of actions from initial state to goal
- (Sutton and Barto, 1998; Kaelbling et al., 1996)

Policy and Cumulative Reward

- Policy, $\pi : \mathcal{S} \rightarrow \mathcal{A} \quad a_t = \pi(s_t)$
- Value of a policy, $V^{\pi}(s_t)$
- Finite-horizon:

$$V^{\pi}(s_t) = E[r_{t+1} + r_{t+2} + \cdots + r_{t+T}] = E\left[\sum_{i=1}^{T} r_{t+i}\right]$$

- Infinite horizon:

$$V^{\pi}(s_t) = E[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots] = E\left[\sum_{i=1}^{\infty} \gamma^{i-1} r_{t+i}\right]$$

$0 \le \gamma < 1$ is the discount rate

$$V^*(s_t) = \max_\pi V^\pi(s_t), \forall s_t$$

$$= \max_{a_t} E\left[\sum_{i=1}^{\infty} \gamma^{i-1} r_{t+i}\right]$$

$$= \max_{a_t} E\left[r_{t+1} + \gamma \sum_{i=1}^{\infty} \gamma^{i-1} r_{t+i+1}\right]$$

$$= \max_{a_t} E\left[r_{t+1} + \gamma V^*(s_{t+1})\right]$$

Bellman's equation

$$V^*(s_t) = \max_{a_t}\left(E[r_{t+1}] + \gamma \sum_{s_{t+1}} P(s_{t+1} \mid s_t, a_t) V^*(s_{t+1})\right)$$

$$V^*(s_t) = \max_{a_t} Q^*(s_t, a_t) \quad \text{Value of } a_t \text{ in } s_t$$

$$Q^*(s_t, a_t) = E[r_{t+1}] + \gamma \sum_{s_{t+1}} P(s_{t+1} \mid s_t, a_t) \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1})$$

Model-Based Learning

- Environment, $P(s_{t+1} \mid s_t, a_t)$, $p(r_{t+1} \mid s_t, a_t)$, is known
- There is no need for exploration
- Can be solved using dynamic programming
- Solve for

$$V^*(s_t) = \max_{a_t}\left(E[r_{t+1}] + \gamma \sum_{s_{t+1}} P(s_{t+1} \mid s_t, a_t) V^*(s_{t+1})\right)$$

- Optimal policy

$$\pi^*(s_t) = \arg\max_{a_t}\left(E[r_{t+1} \mid s_t, a_t] + \gamma \sum_{s_{t+1}} P(s_{t+1} \mid s_t, a_t) V^*(s_{t+1})\right)$$

Value Iteration

Initialize $V(s)$ to arbitrary values

Repeat

  For all $s \in \mathcal{S}$

    For all $a \in \mathcal{A}$

      $Q(s,a) \leftarrow E[r|s,a] + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a)V(s')$

    $V(s) \leftarrow \max_a Q(s,a)$

Until $V(s)$ converge

Policy Iteration

Initialize a policy π arbitrarily

Repeat

$\pi \leftarrow \pi'$

Compute the values using π by

solving the linear equations

$$V^{\pi}(s) = E[r|s, \pi(s)] + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, \pi(s)) V^{\pi}(s')$$

Improve the policy at each state

$$\pi'(s) \leftarrow \arg\max_a (E[r|s,a] + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) V^{\pi}(s'))$$

Until $\pi = \pi'$

Temporal Difference Learning

- Environment, $P(s_{t+1} \mid s_t, a_t)$, $p(r_{t+1} \mid s_t, a_t)$, is not known; model-free learning
- There is need for exploration to sample from $P(s_{t+1} \mid s_t, a_t)$ and $p(r_{t+1} \mid s_t, a_t)$
- Use the reward received in the next time step to update the value of current state (action)
- The temporal difference between the value of the current action and the value discounted from the next state

Exploration Strategies

- ε-greedy: With pr ε,choose one action at random uniformly; and choose the best action with pr 1-ε
- Probabilistic:

$$P(a \mid s) = \frac{\exp Q(s,a)}{\sum_{b=1}^{\mathcal{A}} \exp Q(s,b)}$$

- Move smoothly from exploration/exploitation.
- Decrease ε
- Annealing

$$P(a \mid s) = \frac{\exp[Q(s,a)/T]}{\sum_{b=1}^{\mathcal{A}} \exp[Q(s,b)/T]}$$

Deterministic Rewards and Actions

$$Q^*(s_t, a_t) = E[r_{t+1}] + \gamma \sum_{s_{t+1}} P(s_{t+1} \mid s_t, a_t) \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1})$$

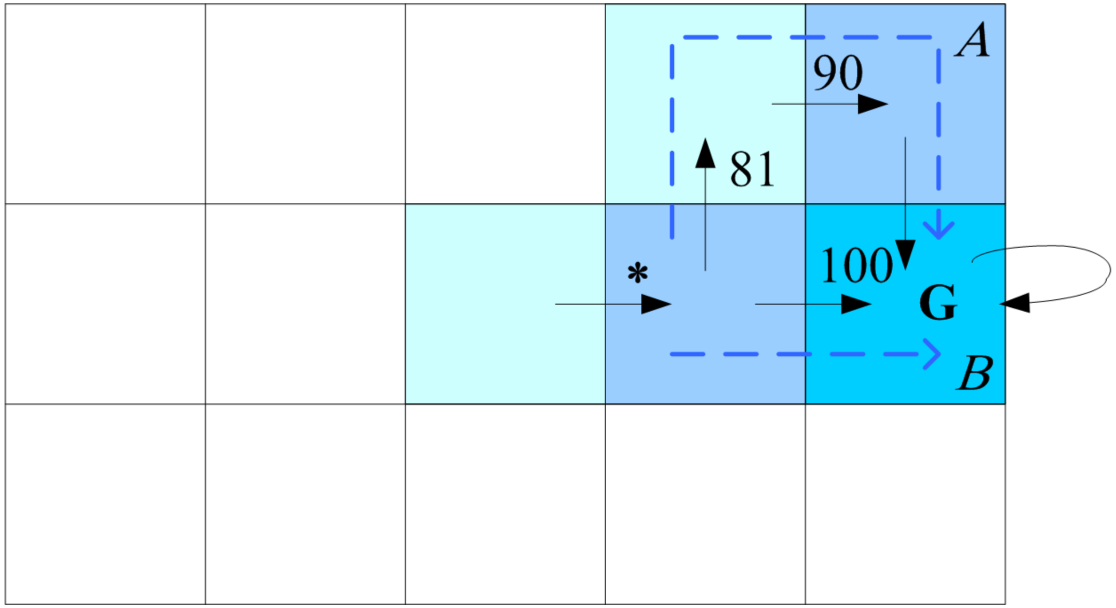
- Deterministic: single possible reward and next state

$$Q(s_t, a_t) = r_{t+1} + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})$$

used as an update rule (backup)

$$\hat{Q}(s_t, a_t) \leftarrow r_{t+1} + \gamma \max_{a_{t+1}} \hat{Q}(s_{t+1}, a_{t+1})$$

Starting at zero, Q values increase, never decrease



Consider the value of action marked by '*':

If path A is seen first, Q(*)=0.9*max(0,81)=73

Then B is seen, Q(*)=0.9*max(100,81)=90

Or,

If path B is seen first, Q(*)=0.9*max(100,0)=90

Then A is seen, Q(*)=0.9*max(100,81)=90

Q values increase but never decrease

Nondeterministic Rewards and Actions

- When next states and rewards are nondeterministic (there is an opponent or randomness in the environment), we keep averages (expected values) instead as assignments
- Q-learning (Watkins and Dayan, 1992):

$$\hat{Q}(s_t,a_t) \leftarrow \hat{Q}(s_t,a_t) + \eta\left(\boxed{r_{t+1} + \gamma \max_{a_{t+1}} \hat{Q}(s_{t+1},a_{t+1})} - \hat{Q}(s_t,a_t)\right)$$

backup

- Off-policy vs on-policy (Sarsa)
- Learning V (TD-learning: Sutton, 1988)

$$V(s_t) \leftarrow V(s_t) + \eta\left(\boxed{r_{t+1} + \gamma V(s_{t+1})} - V(s_t)\right)$$

Q-learning

Initialize all $Q(s,a)$ arbitrarily
For all episodes
  Initalize s
  Repeat
    Choose a using policy derived from Q, e.g., ϵ-greedy
    Take action a, observe r and s'
    Update $Q(s,a)$:
$$Q(s,a) \leftarrow Q(s,a) + \eta(r + \gamma \boxed{\max_{a'} Q(s',a')} - Q(s,a))$$
    $s \leftarrow s'$
  Until s is terminal state

Sarsa

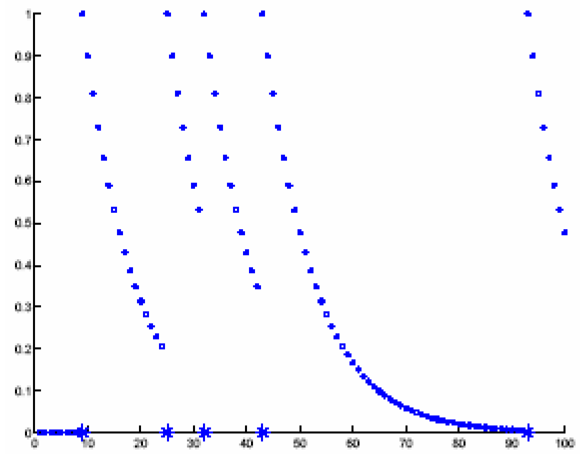
Initialize all $Q(s,a)$ arbitrarily
For all episodes
    Initalize s
    Choose a using policy derived from Q, e.g., ϵ-greedy
    Repeat
        Take action a, observe r and s'
        Choose a' using policy derived from Q, e.g., ϵ-greedy
        Update $Q(s,a)$:
            $Q(s,a) \leftarrow Q(s,a) + \eta(r + \gamma \boxed{Q(s',a')} - Q(s,a))$
        $s \leftarrow s'$, $a \leftarrow a'$
    Until s is terminal state

Eligibility Traces

- Keep a record of previously visited states (actions)

$$e_t(s,a) = \begin{cases} 1 & \text{if } s = s_t \text{ and } a = a_t \\ \gamma\lambda e_{t-1}(s,a) & \text{otherwise} \end{cases}$$

$$\delta_t = r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)$$
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \eta \delta_t e_t(s,a), \forall s, a$$



Sarsa (λ)

Initialize all $Q(s,a)$ arbitrarily, $e(s,a) \leftarrow 0, \forall s, a$
For all episodes
    Initalize s
    Choose a using policy derived from Q, e.g., ϵ-greedy
    Repeat
        Take action a, observe r and s'
        Choose a' using policy derived from Q, e.g., ϵ-greedy
        $\delta \leftarrow r + \gamma Q(s',a') - Q(s,a)$
        $e(s,a) \leftarrow 1$
        For all s, a:
            $Q(s,a) \leftarrow Q(s,a) + \eta \delta e(s,a)$
            $e(s,a) \leftarrow \gamma \lambda e(s,a)$
        $s \leftarrow s', \ a \leftarrow a'$
    Until s is terminal state

Generalization

- Tabular: $Q(s, a)$ or $V(s)$ stored in a table
- Regressor: Use a learner to estimate $Q(s, a)$ or $V(s)$

$$E^t(\boldsymbol{\theta}) = [r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]^2$$

$$\Delta\boldsymbol{\theta} = \eta[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]\nabla_{\theta_t} Q(s_t, a_t)$$

Eligibility

$$\Delta\boldsymbol{\theta} = \eta\delta_t \boldsymbol{e}_t$$

$$\delta_t = r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)$$

$$\boldsymbol{e}_t = \gamma\lambda\boldsymbol{e}_{t-1} + \nabla_{\theta_t} Q(s_t, a_t) \text{ with } \boldsymbol{e}_0 \text{ all zeros}$$

Partially Observable States

- The agent does not know its state but receives an observation $p(o_{t+1}|s_t, a_t)$ which can be used to infer a belief about states
- Partially observable MDP