



Lecture Slides for

INTRODUCTION TO

# Machine Learning

2nd Edition

ETHEM ALPAYDIN

© The MIT Press, 2010

*alpaydin@boun.edu.tr*

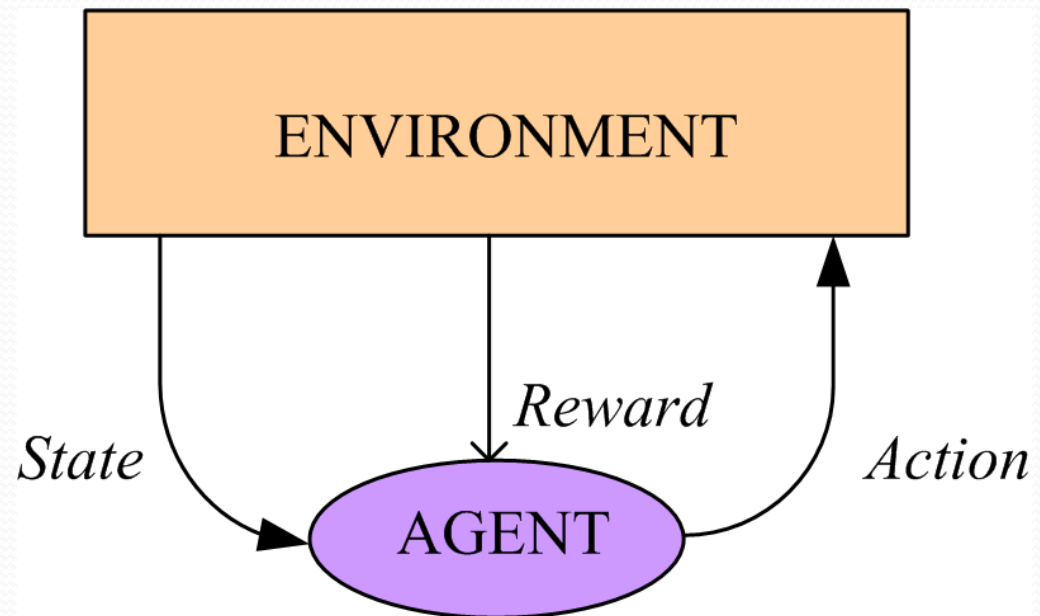
*<http://www.cmpe.boun.edu.tr/~ethem/i2ml2e>*

CHAPTER 18:

# 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

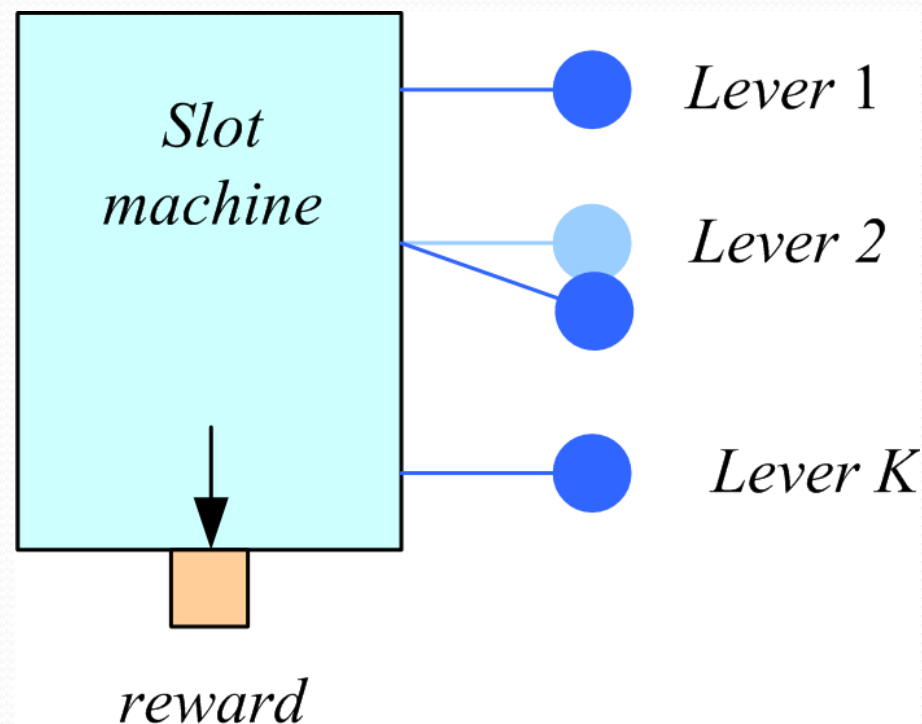
$Q(a)$ : value of action  $a$

Reward is  $r_a$

Set  $Q(a) = r_a$

Choose  $a^*$  if

$$Q(a^*) = \max_a Q(a)$$



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

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

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

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

- Infinite horizon:

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

$0 \leq \gamma < 1$  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

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

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

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

# Model-Based Learning

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

- Solve for

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

- Optimal policy

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



# 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  $\pi$  arbitrarily

Repeat

$$\pi \leftarrow \pi'$$

Compute the values using  $\pi$  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,  $P(s_{t+1} | s_t, a_t)$ ,  $p(r_{t+1} | s_t, a_t)$ , is not known; model-free learning
- There is need for exploration to sample from  $P(s_{t+1} | s_t, a_t)$  and  $p(r_{t+1} | s_t, a_t)$
- 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

- $\epsilon$ -greedy: With pr  $\epsilon$ , choose one action at random uniformly; and choose the best action with pr  $1-\epsilon$

- Probabilistic:

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

- Move smoothly from exploration/exploitation.

- Decrease  $\epsilon$

- Annealing

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

# Deterministic Rewards and Actions

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

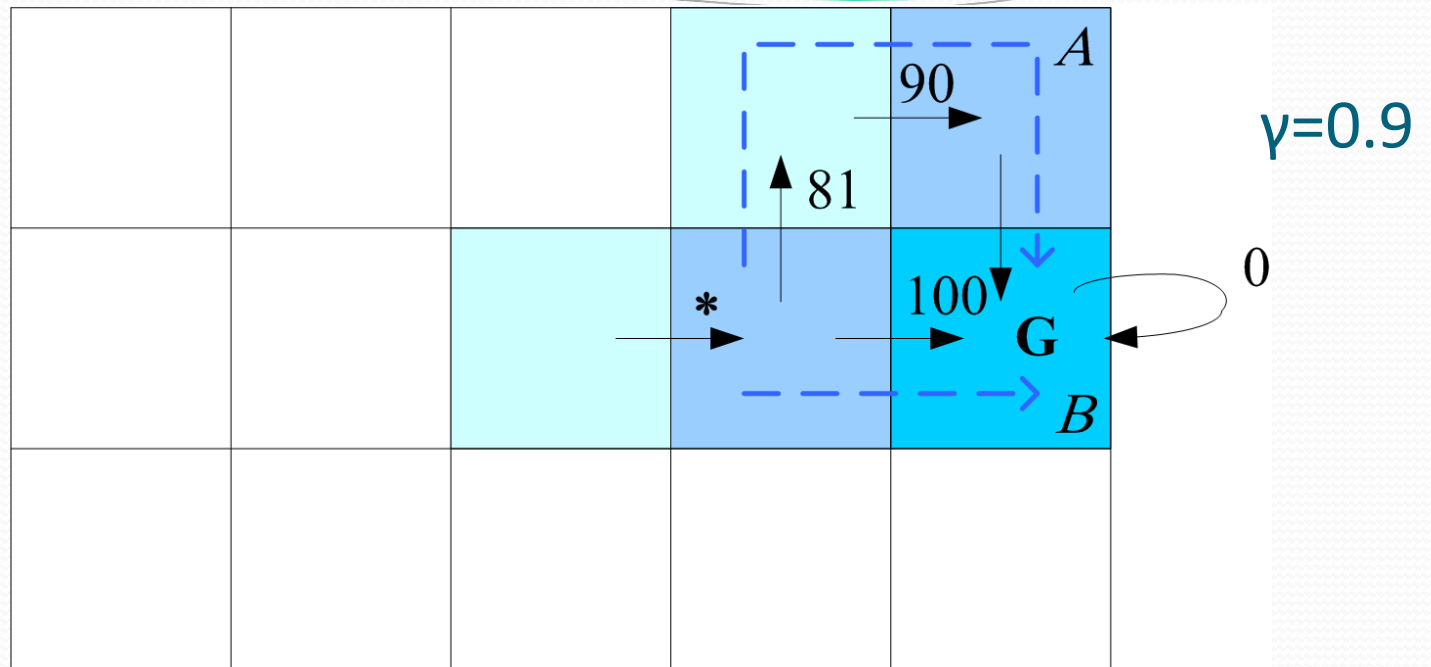
- Deterministic: single possible reward and next state

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

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):

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

- Off-policy vs on-policy (Sarsa)
- Learning  $V$  (TD-learning: Sutton, 1988) <sup>backup</sup>

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

# Q-learning

Initialize all  $Q(s, a)$  arbitrarily

For all episodes

    Initialize  $s$

    Repeat

        Choose  $a$  using policy derived from  $Q$ , e.g.,  $\epsilon$ -greedy

        Take action  $a$ , observe  $r$  and  $s'$

        Update  $Q(s, a)$ :

$$Q(s, a) \leftarrow Q(s, a) + \eta(r + \gamma \max_{a'} Q(s', a') - Q(s, a))$$
$$s \leftarrow s'$$

    Until  $s$  is terminal state



# Sarsa

Initialize all  $Q(s, a)$  arbitrarily

For all episodes

    Initialize  $s$

    Choose  $a$  using policy derived from  $Q$ , e.g.,  $\epsilon$ -greedy

    Repeat

        Take action  $a$ , observe  $r$  and  $s'$

        Choose  $a'$  using policy derived from  $Q$ , e.g.,  $\epsilon$ -greedy

        Update  $Q(s, a)$ :

$$Q(s, a) \leftarrow Q(s, a) + \eta(r + \gamma Q(s', a') - Q(s, a))$$

$$s \leftarrow s', \quad a \leftarrow a'$$

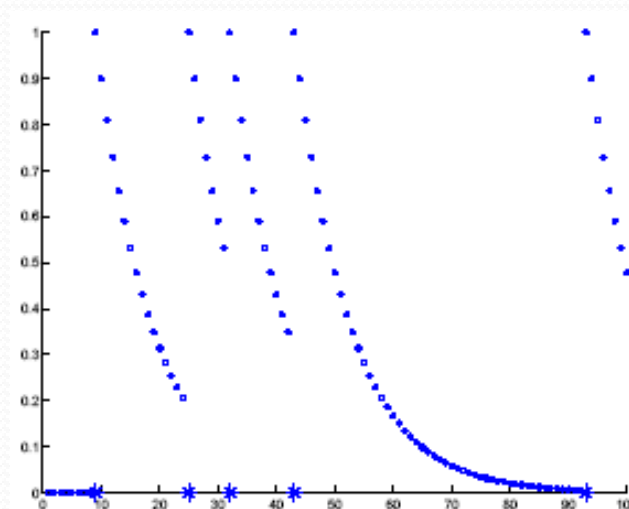
    Until  $s$  is terminal state

# 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 ( $\lambda$ )

```
Initialize all  $Q(s, a)$  arbitrarily,  $e(s, a) \leftarrow 0, \forall s, a$ 
For all episodes
  Initialize  $s$ 
  Choose  $a$  using policy derived from  $Q$ , e.g.,  $\epsilon$ -greedy
  Repeat
    Take action  $a$ , observe  $r$  and  $s'$ 
    Choose  $a'$  using policy derived from  $Q$ , e.g.,  $\epsilon$ -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  $V(s)$

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

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

Eligibility

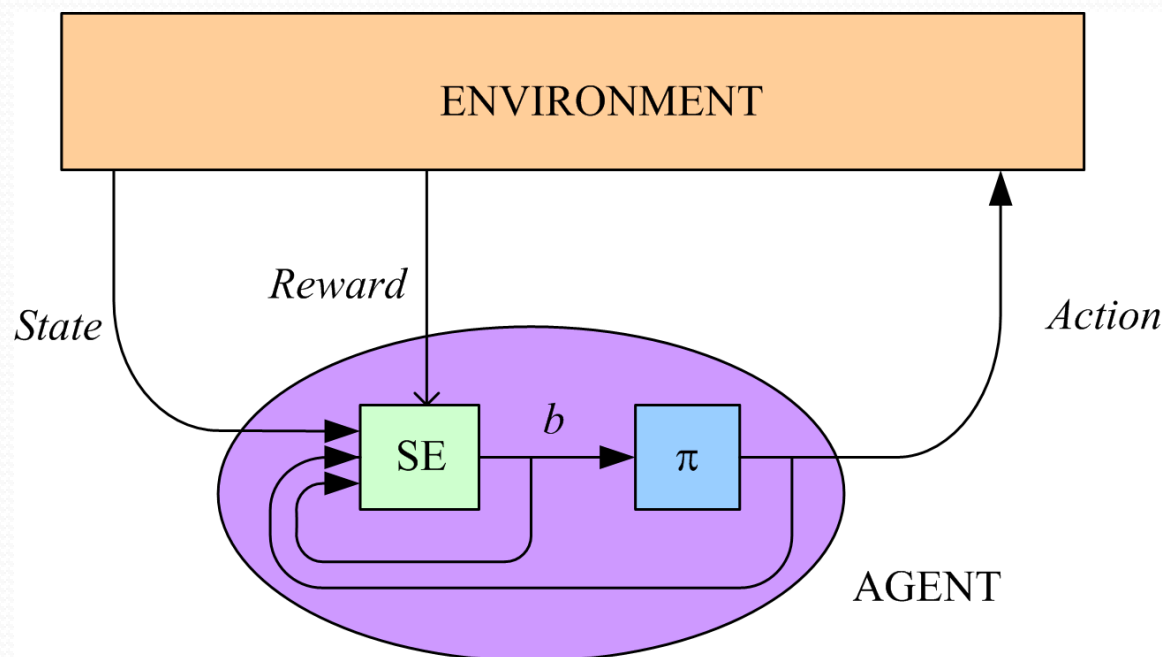
$$\Delta \boldsymbol{\theta} = \eta \delta_t \mathbf{e}_t$$

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

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

# 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



# The Tiger Problem

- Two doors, behind one of which there is a tiger
- $p$ : prob that tiger is behind the left door

| $r(A, Z)$  | Tiger left | Tiger right |
|------------|------------|-------------|
| Open left  | -100       | +80         |
| Open right | +90        | -100        |

- $R(a_L) = -100p + 80(1-p)$ ,  $R(a_R) = 90p - 100(1-p)$
- We can sense with a reward of  $R(a_S) = -1$
- We have unreliable sensors

$$\begin{array}{ll} P(o_L|z_L) = 0.7 & P(o_L|z_R) = 0.3 \\ P(o_R|z_L) = 0.3 & P(o_R|z_R) = 0.7 \end{array}$$

- If we sense  $o_L$ , *our belief in tiger's position changes*

$$p' = P(z_L | o_L) = \frac{P(o_L | z_L)P(z_L)}{P(o_L)} = \frac{0.7p}{0.7p + 0.3(1-p)}$$

$$\begin{aligned} R(a_L | o_L) &= r(a_L, z_L)P(z_L | o_L) + r(a_L, z_R)P(z_R | o_L) \\ &= -100p' + 80(1-p') \end{aligned}$$

$$= -100 \frac{0.7p}{P(o_L)} + 80 \frac{0.3(1-p)}{P(o_L)}$$

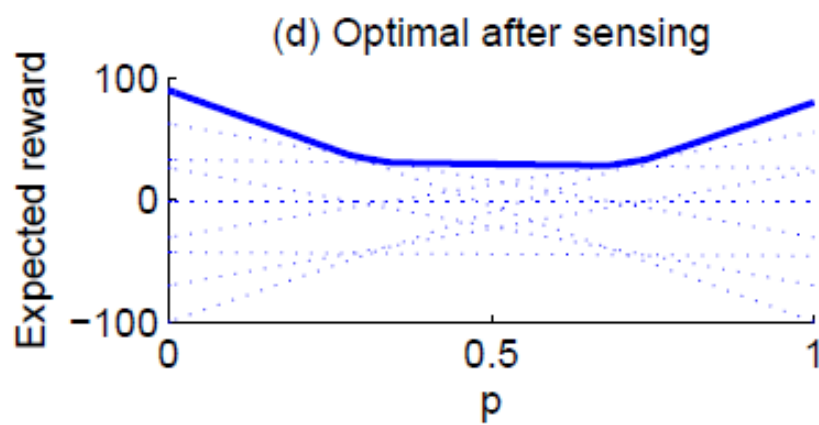
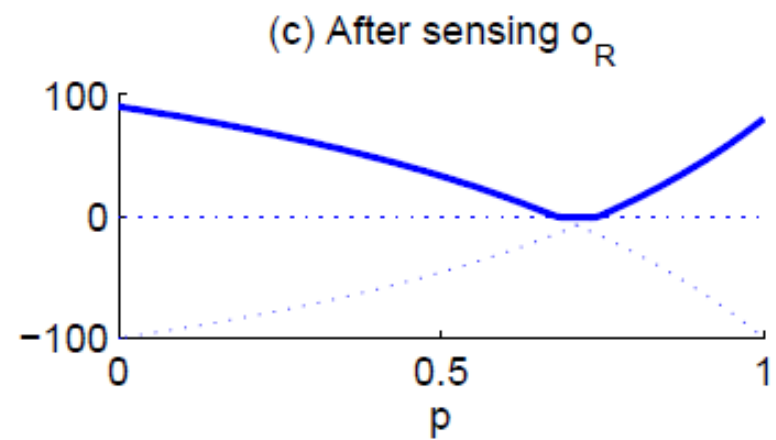
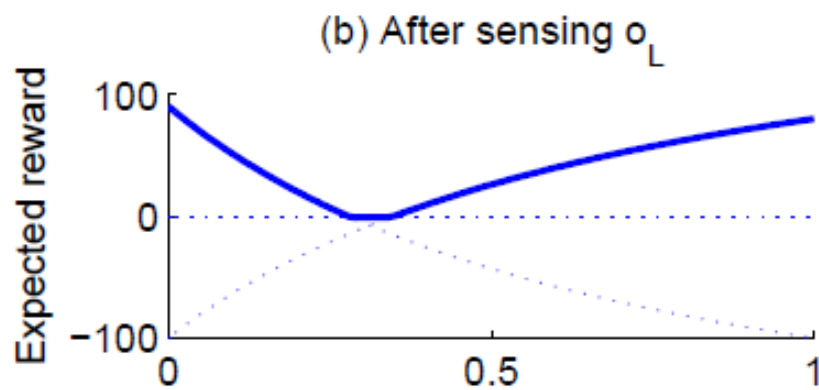
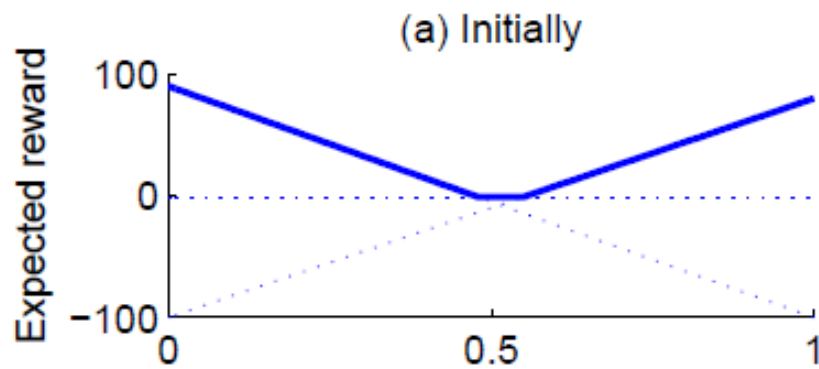
$$\begin{aligned} R(a_R | o_L) &= r(a_R, z_L)P(z_L | o_L) + r(a_R, z_R)P(z_R | o_L) \\ &= 90p' - 100(1-p') \end{aligned}$$

$$= 90 \frac{0.7p}{P(o_L)} - 100 \frac{0.3(1-p)}{P(o_L)}$$

$$R(a_S | o_L) = -1$$

$$\begin{aligned}
 V' &= \sum_j \left[ \max_i R(a_i | o_j) \right] P(o_j) \\
 &= \max(R(a_L | o_L), R(a_R | o_L), R(a_S | o_L)) P(o_L) + \max(R(a_L | o_R), R(a_R | o_R), R(a_S | o_R)) P(o_R) \\
 &= \max \begin{pmatrix} -100p & +80(1-p) \\ -43p & -46(1-p) \\ 33p & +26(1-p) \\ 90p & -100(1-p) \end{pmatrix}
 \end{aligned}$$

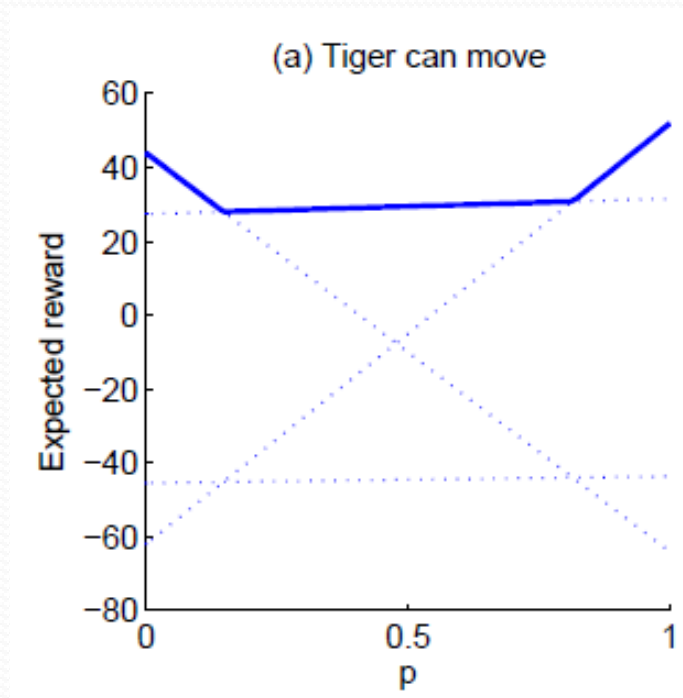




- Let us say the tiger can move from one room to the other with prob 0.8

$$p' = 0.2p + 0.8(1 - p)$$

$$V' = \max \begin{pmatrix} -100p' & +80(1-p') \\ 33p & +26(1-p') \\ 90p & -100(1-p') \end{pmatrix}$$



- When planning for episodes of two, we can take  $a_L$ ,  $a_R$ , or sense and wait:

$$V_2 = \max \begin{pmatrix} -100p & +80(1-p) \\ 90p & -100(1-p) \\ \max V' & -1 \end{pmatrix}$$

