

# Learning Algorithms: Illustrative Examples

---

Notes for Economics 308  
(Agent-Based Computational Economics)

Leigh Tesfatsion

Professor of Economics and Mathematics

Iowa State University

Ames, IA 50011-1070

<http://www.econ.iastate.edu/tesfatsi/>

[tesfatsi@iastate.edu](mailto:tesfatsi@iastate.edu)

# References & Acknowledgement

---

## Main References (Linked to Econ 308 Syllabus):

[1] "Notes on Learning"

[www.econ.iastate.edu/classes/econ308/tesfatsi/learning.Econ308.pdf](http://www.econ.iastate.edu/classes/econ308/tesfatsi/learning.Econ308.pdf)

[2] "Constructing Computational Traders with Learning Capabilities"

[www.econ.iastate.edu/classes/econ308/tesfatsi/ConstructLearningAgent.pdf](http://www.econ.iastate.edu/classes/econ308/tesfatsi/ConstructLearningAgent.pdf)

[3] "Learning and the Embodied Mind"

[www.econ.iastate.edu/tesfatsi/aemind.htm](http://www.econ.iastate.edu/tesfatsi/aemind.htm)

## Important Acknowledgement:

Some of the slides below are adopted from the following great on-line slide presentations:

Andrew Barto, *"Searching in the Right Space"*

Bill Smart, *"Reinforcement Learning: A User's Guide"*

Bill Tomlinson, *"Biomorphic Computing"*

Wendy Williams, *"GA Tutorial"*

Nicolas Galoppo von Borries, *"Intro To ANNs"*

# Presentation Outline

---

- 1. Overview
- 2. Reactive Reinforcement Learning (RL)
  - *Example 1:* Deterministic reactive RL (e.g. Derivative-Follower)
  - *Example 2:* Stochastic reactive RL (e.g. Roth-Erev algorithms)
- 3. Belief-Based Learning
  - *Example 1:* Fictitious play
  - *Example 2:* Hybrid forms (e.g. Camerer/Ho EWA algorithm)

# Presentation Outline...Continued

---

- ❑ 4. Anticipatory Learning  
(*Example: Q-Learning*)
- ❑ 5. Evolutionary Learning  
(*Example: Genetic Algorithms - GAs*)
- ❑ 6. Connectionist Learning  
(*Example: Artificial Neural Nets - ANNs*)

# 1. Overview

---

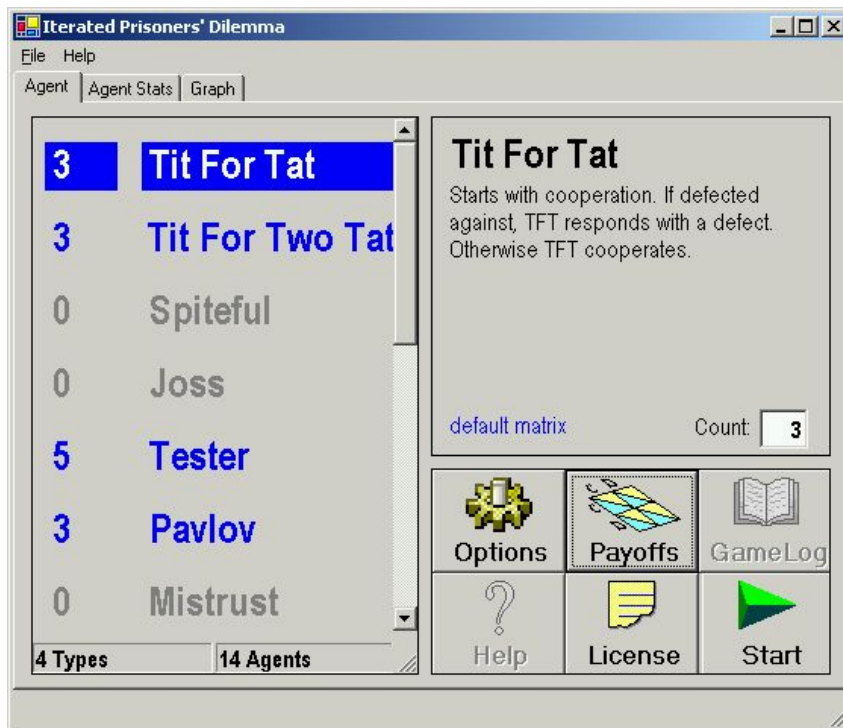
- So far, we have worked with given strategies for very simple one-stage and iterated (multi-stage) games
- The strategies we have seen to date for iterated games have been *adaptive* in the following sense:
  - The action dictated by the strategy at any given time is conditioned on the current (information) state of the player
- But this state conditioning is determined in advance of any actual game play.

**Example:** TFT (start by cooperating, then do whatever your rival did in the previous stage)

# Axelrod Tournament Demo

Basic Tournament by R. Axelrod; Demo developed by C. Cook

<http://www.econ.iastate.edu/tesfatsi/acedemos.htm>



- User-specified strategies for playing a specified type of game (e.g. PD, Chicken, Stag Hunt) are pitted against one another in repeated round-robin play.

- **KEY ISSUE:**

What types of strategies **perform best over time?**

**Will nasty or cooperative types prevail?**

# Overview...Continued

---

- In the next part of the course we will investigate adaptive strategies for more complicated types of iterated **MARKET** games.
- We will also investigate the possibility of **LEARNING** in iterated market games.
- That is, we will want to permit one or more players to **STRUCTURALLY MODIFY** their strategies **DURING** successive game iterations based on sequentially observed events.

# Overview...Continued

---

**LEARNING** means....for example:

- A player starts an iterated game with an initial strategy ("policy")  $\pi$  dictating an action  $a$  to be taken in each state  $s$ :

State  $s \rightarrow$  Action  $a$

- But, after observing the payoff ("reward")  $r$  from using this state-action association, the player eventually decides to **change** this association:

State  $s \rightarrow$  Action  $a^*$



## Caution: Intrinsic Ambiguity in the Distinction between Adaptation and Learning

---

- Suppose an agent is acting in accordance with a particular state-action association  $s \rightarrow a$  in a general environment  $e$ .
- Suppose something happens ( $e$  changes to  $e^*$ ) that convinces the agent to change this association to some other association  $s \rightarrow a^*$ .
- If the definition of "state" is expanded from  $s$  to  $(s,e)$ , the associations  $(s,e) \rightarrow a$  and  $(s,e^*) \rightarrow a^*$  *have not changed*.

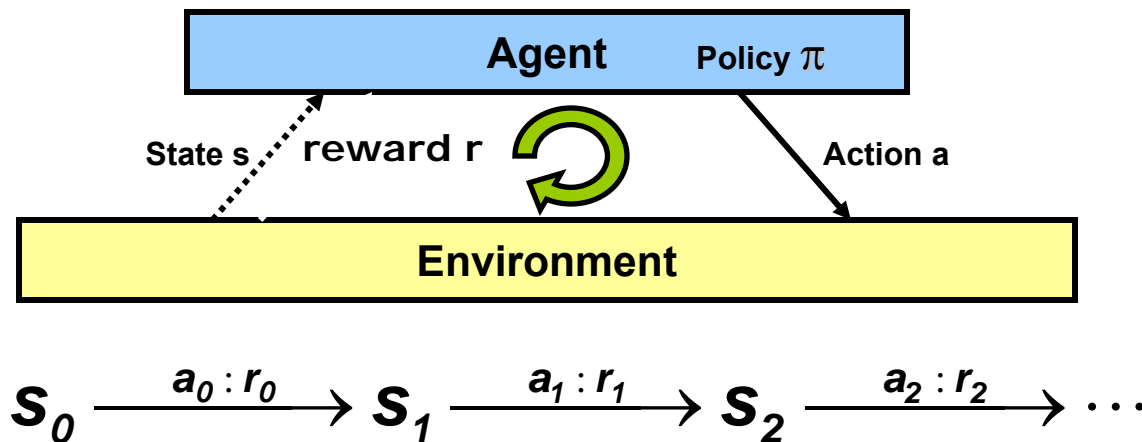
# General Types of Learning

---

- **Unsupervised Learning**
  - Update structure based on intrinsic motivation (curiosity, enjoyment, moral duty, ...)
  
- **Reinforcement Learning (RL)**
  - Update structure in response to successive rewards attained through actions taken
  
- **Supervised Learning**
  - Update structure on basis of examples of desired (or required) state-action associations provided by an expert external supervisor

# Reinforcement Learning (RL)

## □ Elements of traditional RL:

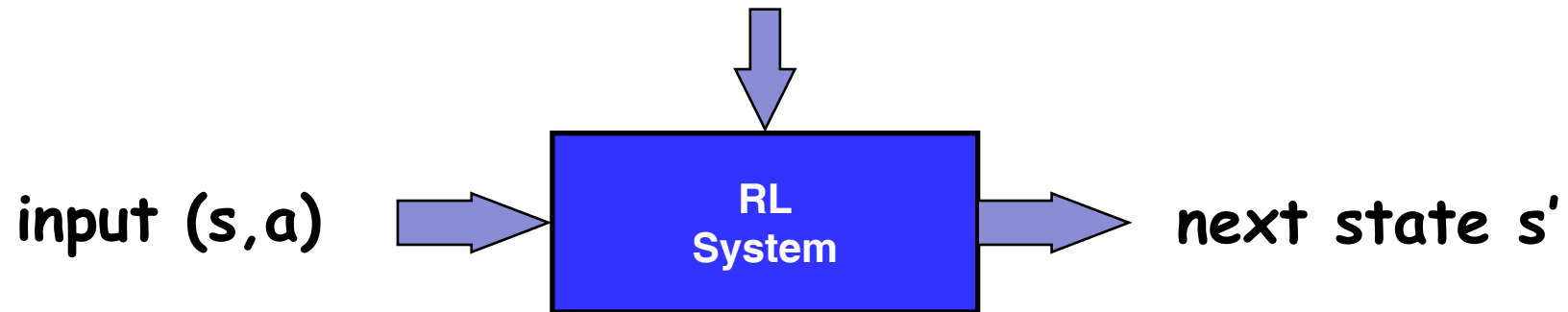


- **Policy  $\pi$** : Maps each state  $s$  to an action choice  $a$
- **Reward  $r$** : Immediate value of state-action pairing
- **Transition model  $T(s,a)=s'$** : Maps current state-action pairing  $(s,a)$  to a next state  $s'$

# Elements of Traditional RL...

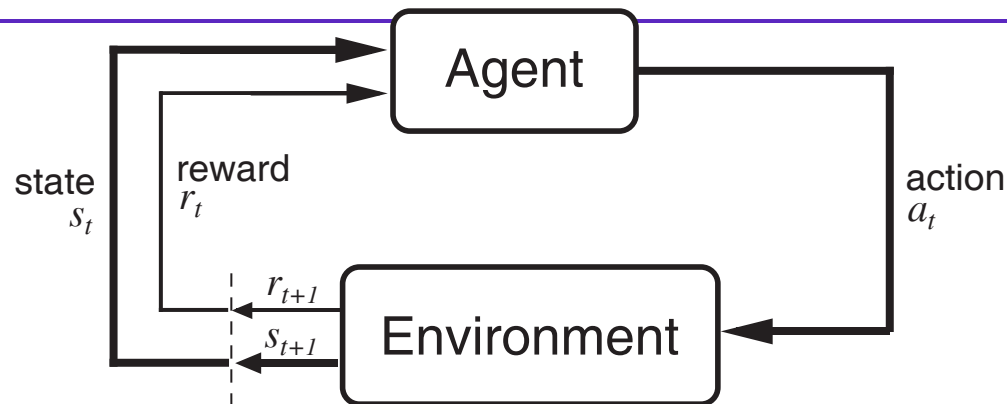
---

reward  $r$  (“utility”, “score”, “payoff”, “penalty”)



**Basic Intuition:** The tendency to take an action  $a$  in state  $s$  should be strengthened (reinforced) if it produces favorable results and weakened if it produces unfavorable results.

# Traditional RL in More Detail



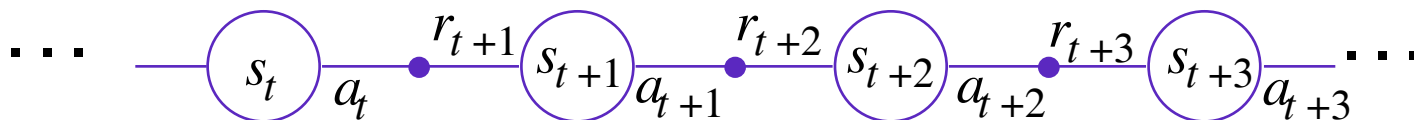
Agent and environment interact at discrete time steps :  $t = 0, 1, 2, \dots$

Agent observes state at step  $t$  :  $s_t \in \mathcal{S}$

produces action at step  $t$  :  $a_t \in A(s_t)$

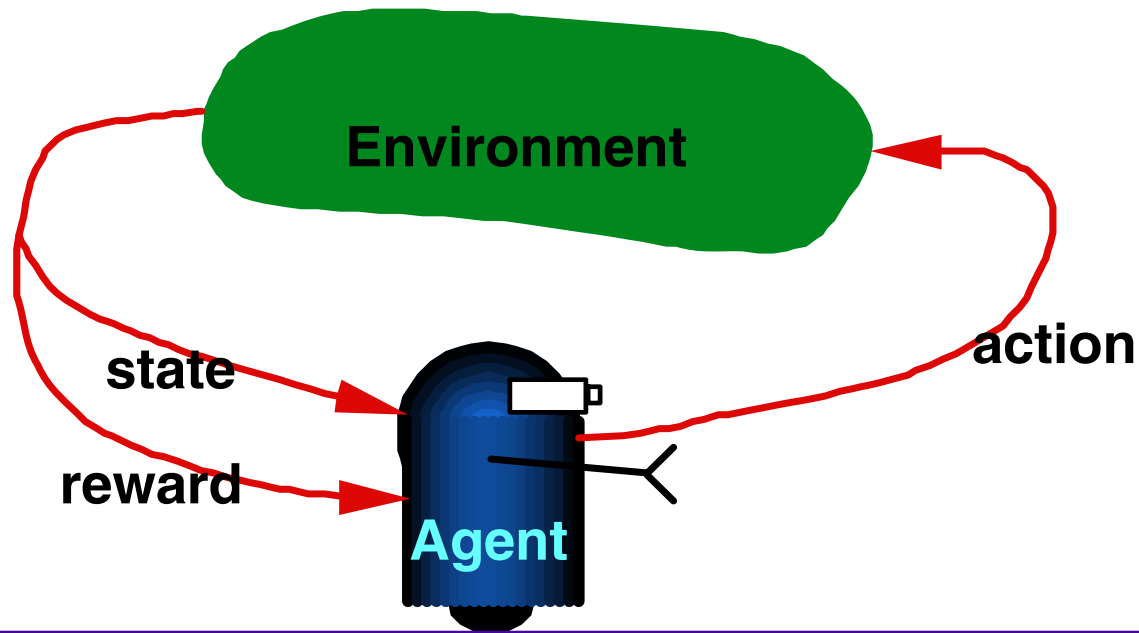
gets resulting reward :  $r_{t+1} \in \mathcal{R}$

and resulting next state :  $s_{t+1}$



# Traditional RL View of Agent Action Choice

---



States and rewards are modeled as external forces determining an agent's choice of actions

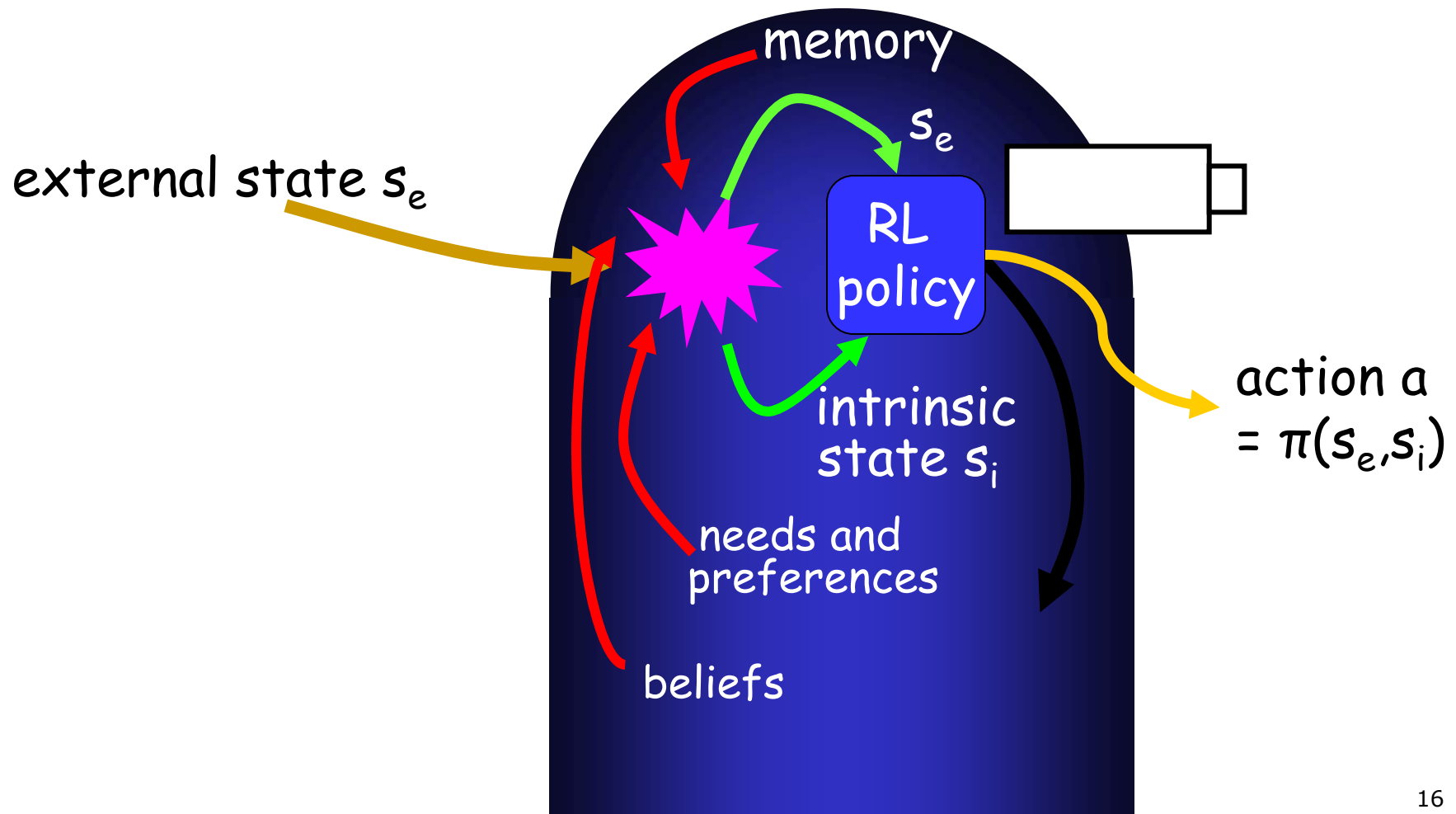
# In Accord with Human Motivation?

---

Factors that energize a person to act and that direct his or her activity:

- **Extrinsic Motivation:** Being moved to act in hopes of receiving some external reward (\$\$, a prize, praise, etc.)
- **Intrinsic Motivation:** Being moved to act because it is perceived to be inherently desirable, enjoyable, moral, ...

# A More Modern Extrinsic/Intrinsic View of Agent Action Choice





# Intrinsic Motivation: Questions

---

- An activity is intrinsically motivated if an agent does it for its own sake rather than to receive specific rewards (or avoid specific penalties)
- Curiosity, exploration, moral duty, . . .
- Can a *computational learning system* be intrinsically motivated?
- Specifically, can a *computational RL agent* be intrinsically motivated?

(Cf. Work by Andrew Barto and Satinder Singh)

## 2. Reactive RL

---

*Asks....*

Given past events, what action should I take now?

# Example 1: Deterministic Reactive RL Derivative-Follower (DF) Adaptation (Greenwald and Kephart, 1999)

---

- ❑ Originally developed as a simple way for computational agents to repeatedly modify a *scalar* decision  $d$ .
- ❑ The DF agent experiments with incremental increases or decreases in  $d$  of a given magnitude  $\Delta d > 0$ .
- ❑ An external reward is attained after each change in  $d$ .
- ❑ The DF agent continues to move  $d$  in the same direction (increases or decreases) until the reward level falls, at which point the direction of movement in  $d$  is reversed.
- ❑ Letting states  $s = \Delta \text{reward}$  and actions  $a = \pm \Delta d$ , the associations  $s \rightarrow a$  are actually fixed in advance.

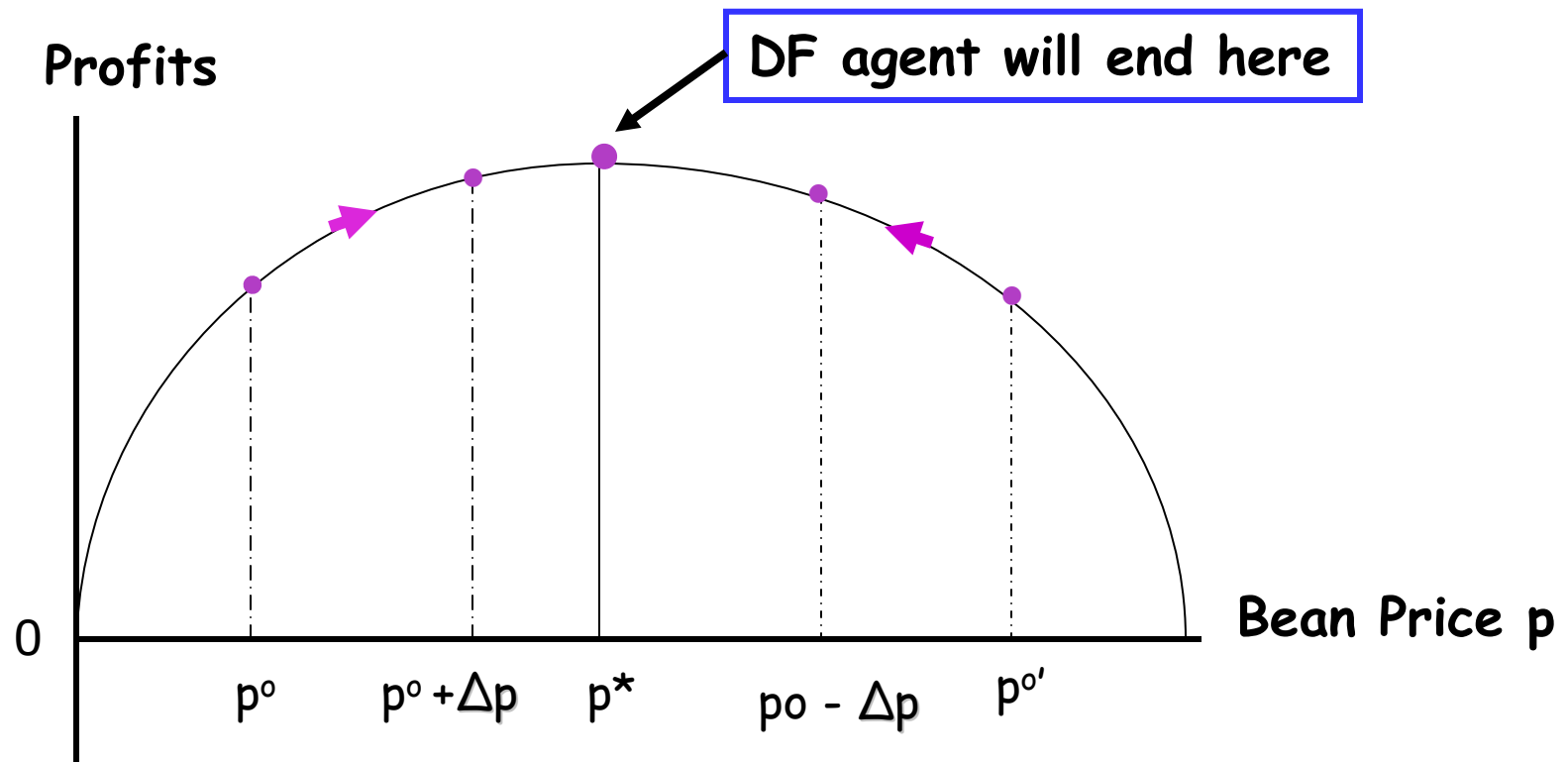
## DF Adaptation: A Simple Market Example

---

- Each day a firm produces  $b^*$  pounds of beans.
- On the first day the firm selects a unit price  $p^0$  (\$'s per pound) at which to sell  $b^*$ .
- The firm then posts successively higher daily prices  $p$  for beans of the form  $p^0 + \Delta p$ ,  $p^0 + 2\Delta p$ , ... with  $\Delta p > 0$  until profits are observed to fall
- The firm then reverses course and starts to decrease  $p$  by step-size  $\Delta p$ . And so on...
- *Question:* When will this work well (if ever)?

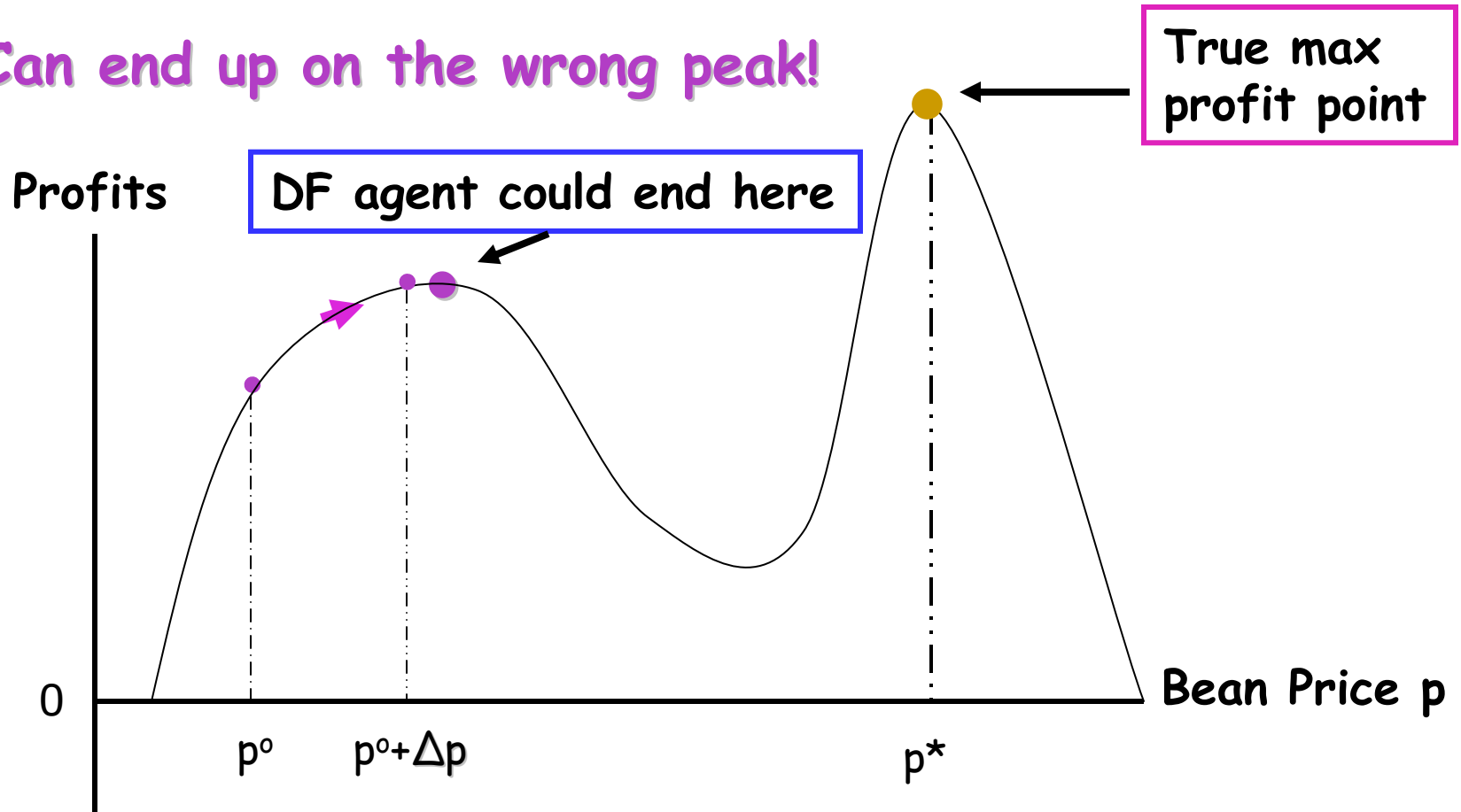
# When will DF adaptation work well (if ever)?

- Suppose profits are a concave function of the price  $p$



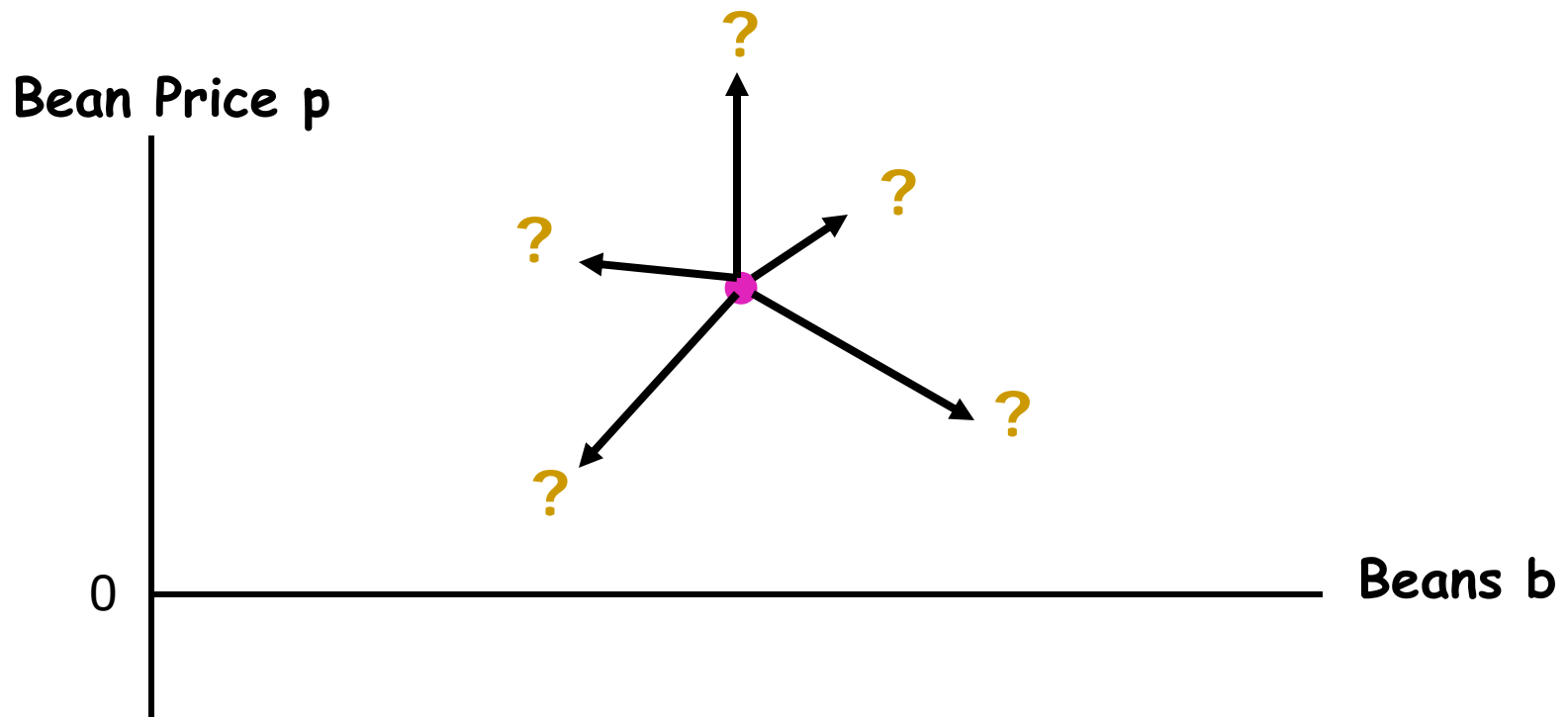
# But suppose profits are NOT a concave function of the price $p$ ?

- Can end up on the wrong peak!



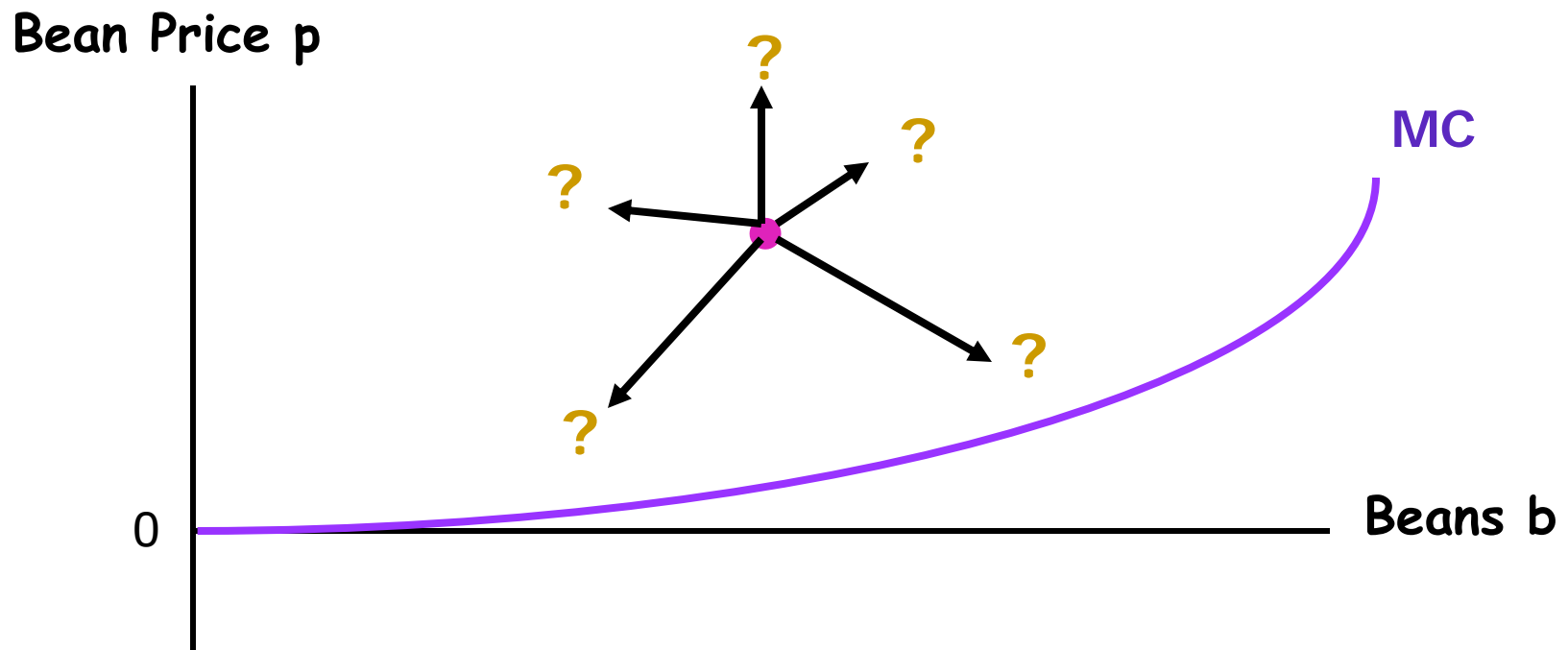
# Or suppose a profit-seeking firm must set **BOTH** price **AND** quantity levels?

- Where to start, which direction to search in, and how far to search in this direction?



A profit-seeking firm should try to stay on or above its marginal production cost function  $MC$

- **KEY ISSUE:** *Correlated  $\Delta p$  and  $\Delta b$  choices needed to stay above  $MC$  & move in desirable directions*





# Example 2: Stochastic Reactive RL

## Roth-Erev Algorithms

---

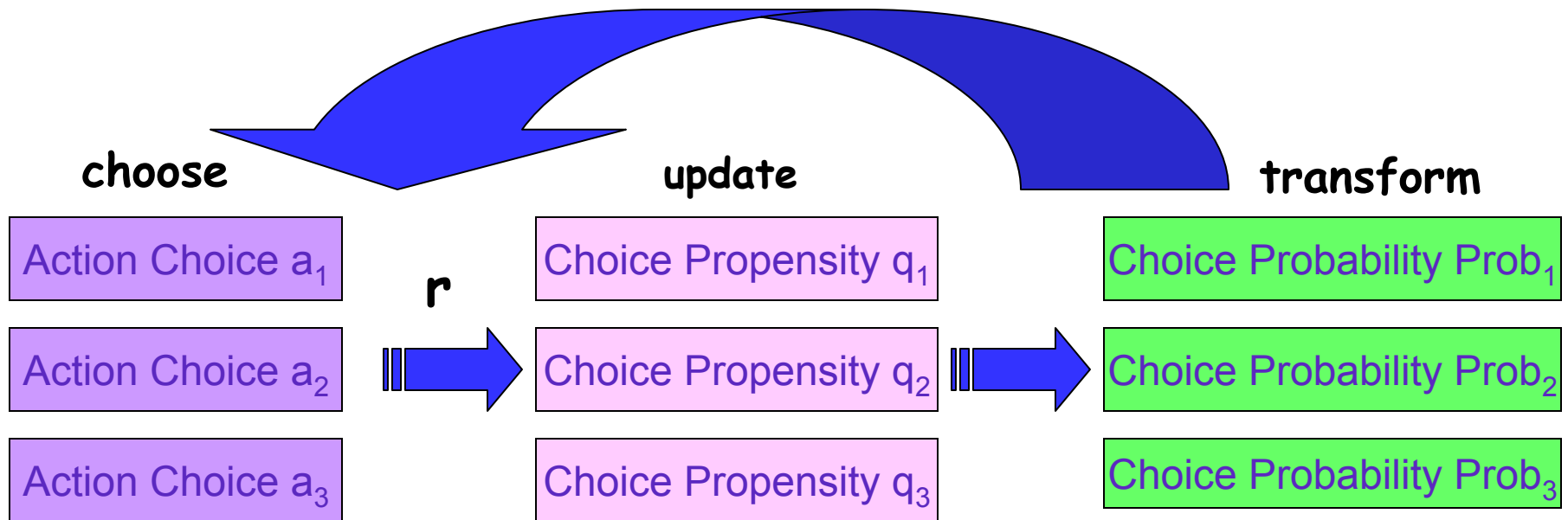
- Developed by Alvin E. Roth and Ido Erev (*Games & Econ Beh.* 1995, *AER* 1998)
  - Based on observations of people's behavior in iterated game play with **multiple** strategically interacting players in various game contexts
  - Two extensions found necessary relative to RL methods developed earlier by psychologists for individuals learning in fixed environments:
    - Need to "forget" rewards received in distant past
    - Need for "spillover" of reward attributions across actions in early game play to encourage experimentation and avoid premature fixation on a suboptimal chosen action.

# Roth-Erev Algorithm: Outline

---

1. **Initialize propensities  $q$**  for choosing actions.
2. **Generate action choice probabilities  $\text{Prob}$**  from current action propensities  $q$ .
3. **Choose an action  $a$**  in accordance with current action choice probabilities  $\text{Prob}$ .
4. **Update action propensity values  $q$**  using the reward  $r$  received after the last chosen action  $a$ .
5. **Repeat** from step 2.

# Roth-Erev Algorithm Structure



- Action choice  $a$  leads to a reward  $r$ , followed by updating of all action choice propensities  $q$  based on this reward, followed by a transformation of these propensities into action choice probabilities Prob.

# Updating of Action Propensities

## Parameters:

- $q_j(0)$  Initial propensity
- $\epsilon$  Experimentation
- $\phi$  Recency (forgetting)

## Variables:

- $a_j$  Current action choice
- $q_j$  Propensity for action  $a_j$
- $a_k$  Last action chosen
- $r_k$  Reward for action  $a_k$
- $t$  Current time step
- $N$  Number of actions

$$q_j(t + 1) = [1 - \phi]q_j(t) + E_j(\epsilon, N, k, t)$$

$$E_j(\epsilon, N, k, t) = \begin{cases} r_k(t)[1 - \epsilon] & \text{if } j = k \\ r_k(t) \frac{\epsilon}{N-1} & \text{if } j \neq k \end{cases}$$

# From Propensities to Probabilities

---

- **Example A:** Probability of choosing action  $j$  at time  $t$  = Relative propensity for action  $j$

$$\text{Prob}_j(t) = \frac{q_j(t)}{\sum_{n=1}^N [q_n(t)]}$$

## Example B: Gibbs-Boltzmann Probability

---

- Handles negative propensity values  $q_j(t)$
- $T$  = Temperature (“cooling”) parameter
- $T$  affects dynamic shaping of Prob distributions

$$Prob_j(t) = \frac{e^{q_j(t)/T}}{\sum_{n=1}^N e^{q_n(t)/T}}$$

# More on the Updating of Action Propensities - 1

---

- Specification of the **initial propensity levels**  $q_j(0)$  for an agent's feasible action choices  $a_j$ ,  $j = 1, \dots, N$ 
  - Initial propensity levels act as "aspiration levels"
  - High initial propensity levels → Agent is disappointed with the rewards resulting from his early chosen actions, which encourages continued experimentation.
  - Low initial propensity levels → Agent is happy with the rewards resulting from his early chosen actions, which encourages premature fixation on one of these actions

## More on the Updating of Action Propensities - 2

---

- Might want to “forget” rewards  $r$  received in the distant past in time-changing environments:
  - Controlled by **recency (forgetting) parameter**  $\varphi$  lying between 0 and 1
  - As  $\varphi$  approaches 1, heaviest weight placed on most recently received rewards  $r$
  - As  $\varphi$  approaches 0, approximately equal weight placed on *all* rewards  $r$  received to date  
(exactly equal weight when  $\varphi = 0$  and  $\varepsilon = 0$  )



## More on the Updating of Action Propensities - 3

---

- Need for “spillover” of reward attributions across actions in early game play to encourage experimentation and avoid premature fixation on a suboptimal chosen action  $a_k$ .
  - Controlled by **experimentation parameter**  $\epsilon$  lying between 0 and 1
  - As  $\epsilon$  increases, more “spillover” of reward resulting from chosen action  $a_k$  to other actions  $a_j$ , resulting in smaller divergencies among propensities  $q_k$  and  $q_j$
  - As  $\epsilon$  approaches 0, reward resulting from chosen action  $a_k$  is attributed only to  $a_k$ , implying only  $a_k$ 's propensity  $q_k$  is updated

# Modified Roth-Erev RL

- Nicolaisen, Petrov and Tesfatsion (IEEE TEC, 2001) modified the response function  $E_j$  so propensity updating occurs even with zero-valued rewards  $r$ , as follows: Letting  $a_j$  = any feasible action choice and  $a_k$  = currently chosen action,

$$E_j(\epsilon, N, k, t) = \begin{cases} r_k(t)[1 - \epsilon] & \text{if } j = k \\ q_j(t) \frac{\epsilon}{N-1} & \text{if } j \neq k \end{cases}$$

- The NPT electricity traders typically achieved 90% or higher market efficiency using Modified Roth-Erev RL.

# Modified Roth-Erev RL

---

□ NPT\* electricity traders typically achieved market efficiency levels  $\geq 90\%$  using *Modified* Roth-Erev RL and much lower market efficiency levels (e.g. 20%) using *Original* Roth-Erev RL.

\* Nicolaisen, J., Petrov, V., and Tesfatsion, L., "Market Power and Efficiency in a Computational Electricity Market with Discriminatory Double-Auction Pricing". *IEEE Transactions on Evolutionary Computing* 5, 5 (October 2001), 504-523.

□ See also Mridul Pentapalli, "A Comparative Study of Roth-Erev and Modified Roth-Erev Reinforcement Learning Algorithms for Uniform-Price Double Auctions," M.S. Thesis Talk, March 2008

[www.econ.iastate.edu/tesfatsi/MridulPentapalli.MSThesisTalk2008.pdf](http://www.econ.iastate.edu/tesfatsi/MridulPentapalli.MSThesisTalk2008.pdf)

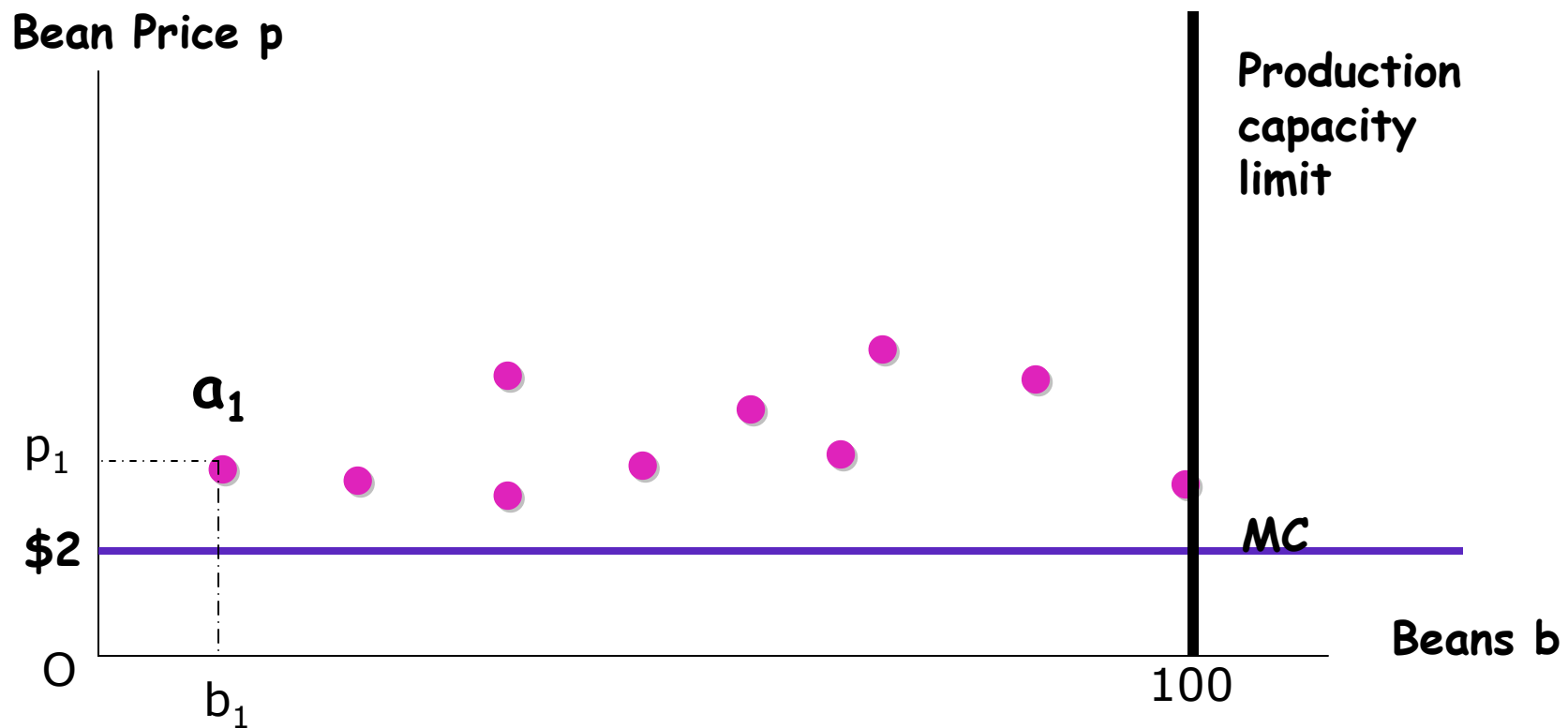
# Illustration: A Bean Firm in a Computational Market Economy

---

- Strategic learning agent
- Gains profit by producing and selling beans  $b$ , a perishable (nonstorable) good measured in lbs
- Adjusts bean production and price level in each trading period  $t$  using Modified Roth-Erev RL
  - Period  $t$  Action = Choice of supply offer of form  
(Production Level  $b$ , Unit Price  $p$ )
- Marginal cost of production = \$2 per lb
- Production limit in each period  $t$ : 100 lbs

# Bean Firm: Structural Conditions

- Action Domain AD: Set of  $N=10$  feasible action choices  $\{(b_1, p_1), \dots, (b_N, p_N)\} = \{a_1, \dots, a_{10}\}$



## Bean Firm Learning Method: Modified Roth-Erev RL - Step 1

---

- Initial propensity levels for actions  $a_1, \dots, a_{10}$ :  
 $q_j(0) = 20, \quad j = 1, \dots, 10$
- Initial probability distribution for choosing among the feasible actions  $a_1, \dots, a_{10}$ :

$$\begin{aligned} \text{Prob}_j(0) &= \exp(q_j(0)/T) / \sum_{n=1}^{10} \exp(q_n(0)/T) \\ &= 1/10, \quad j = 1, 2, \dots, 10 \end{aligned}$$

## Bean Firm Learning Method: Modified Roth-Erev RL - Step 2

---

- Recency (forgetting) parameter:  $\psi = 0.04$
- Experimental (spillover) parameter:  $\varepsilon = 0.96$
- Reward  $r_k(t)$  in trade period  $t \geq 0$  consists of profits (+ or -) resulting from **chosen action**  $a_k(t) = (b_k(t), p_k(t))$  and **actual bean sales**  $b(t)$ :

$$r_k(t) = [ p_k(t) \circ b(t) ] - [ \$2 \circ b_k(t) ]$$

Actual revenues  
from sale of  $b(t)$

Actual costs of  
producing  $b_k(t)$

# Bean Firm Learning Method: Modified Roth-Erev RL - Step 3

---

Updating of propensities after receipt of reward  $r_k(t)$  in period  $t \geq 0$ :

$$q_j(t+1) = [1 - \phi]q_j(t) + E_j(\epsilon, N, k, t)$$

$$E_j(\epsilon, N, k, t) = \begin{cases} r_k(t)[1 - \epsilon] & \text{if } j = k \\ q_j(t) \frac{\epsilon}{N-1} & \text{if } j \neq k \end{cases}$$



# From Propensities to Probabilities for the Bean Firm - Step 4

---

The probability of choosing an action  $j$  is an increasing function of its current propensity value, all else equal:

$$Prob_j(t) = \frac{e^{q_j(t)/T}}{\sum_{n=1}^{10} e^{q_n(t)/T}}$$

Probability of choosing action  
 $j$  at time  $t$ ,  $j=1, \dots, 10$

## Illustrative Pseudo-Code Implementation of Action Choice in Accordance with Action Probabilities

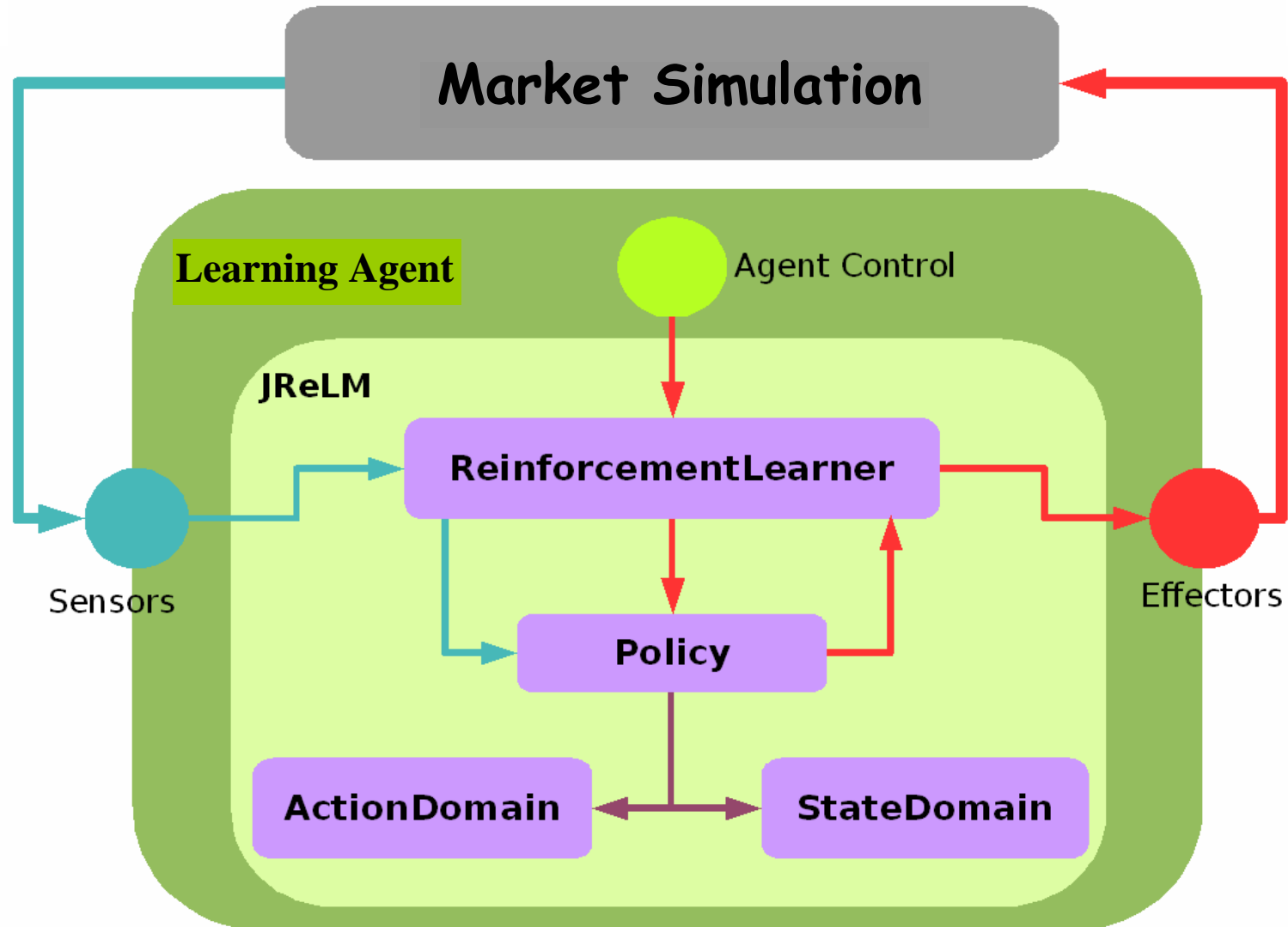
---

```
for j = 1,...,10:  
    pj = probability of choosing action j (previous formula);  
p = Random.nextDouble(); //uniformly distributed double value  
    between 0.0 and 1.0 (Java). For NetLogo: "let p random-float 1.0"  
sum = 0.0;  
for j = 1,...,9:  
    sum = sum + pj ; // form cdf assessment sum=[p1+...+pj]  
    If p ≤ sum, return j; // returned j is index of action choice  
return 10; //returns action choice 10 if no previous return activated
```

**NOTE:** Then  $p_j \cong$  probability that "return j" is activated,  $j = 1, \dots, 10$

# JReLM: Java Reinforcement Learning Module (Includes MRE Reinforcement Learning)

(Charles J. Gieseler, S05 Econ 308 Student, CS M.S. Thesis, 2005)



# 3. Belief-Based Learning

---

Asks ...

What *different* rewards might I have received in the past if I had acted differently?

And how can I use these "*opportunity cost*" assessments to help choose a better action now?

# Belief-Based Learning ...

---

- ❑ In belief-based learning, the presence of other decision making agents in the learning environment is explicitly taken into account.
- ❑ Variants of belief-based learning currently in use by economists include:
  - **Cournot (naïve) belief learning** - the belief that rivals will act today in the same way they acted in the immediate past
  - **Fictitious play** - the belief that rivals will act today in accordance with the historical frequencies of all their past action choices.
  - **Experience-weighted attraction learning** (Camerer/Ho 1999) - hybrid of reactive RL and fictitious play learning

# Belief-Based Learning: Example 1

## Fictitious Play Learning (FPL)

---

- An agent  $A$  assumes each other agent in its choice environment chooses its actions in accordance with an unknown but time-invariant “probability distribution function (PDF)”.
- Agent  $A$  estimates these PDFs based on the historical frequencies with which other agents have been observed to choose different actions.
- At each time  $t$ , Agent  $A$  chooses a “best response” action conditional on its current PDF estimates for other agents.

# Concrete FPL Illustration: Matching Pennies Game

---

		Player 2	
		Heads	Tails
Player 1	Heads	$(1, -1)$	$(-1, 1)$
	Tails	$(-1, 1)$	$(1, -1)$

# Concrete FPL Illustration: Matching Pennies...Continued (1)

---

- The one-shot matching pennies game has NO Nash equilibrium in “pure strategies”.
- That is, none of the four feasible action pairs (H,H), (H,T), (T,H), or (T,T) is a Nash equilibrium.
- However, suppose Player 1 is choosing its actions H and T in accordance with a *mixed strategy*, i.e., a probability distribution function (PDF) over the action domain {H,T} of the form  $[\text{Prob}^1(H), \text{Prob}^1(T)]$ .
- Then Player 2 can calculate a “best response” mixed strategy  $[\text{Prob}^2(H), \text{Prob}^2(T)]$  to Player 1's mixed strategy that maximizes Player 2's *expected* payoff.



# Concrete FPL Illustration: Matching Pennies...Continued (2)

---

- Player 2 is said to engage in *Fictitious Play Learning (FPL)* in the matching pennies game if the following conditions hold:
  - The game is played in successive periods  $t=1,2,\dots$ , and Player 2 in each period  $t > 1$  knows the actions that have been chosen by Player 1 in all *past* periods.
  - In each period  $t > 1$ , Player 2 forms an estimate of the mixed strategy (PDF) it thinks is being used by Player 1 based on the frequencies with which Player 1 has been observed to choose H and T in past game plays.
  - In each period  $t > 1$ , Player 2 chooses a “best response” mixed strategy for its own action choice conditional on its current estimate for the mixed strategy being used by Player 1.

## Concrete FPL Illustration: Matching Pennies...Continued (3)

---

- **EXAMPLE:** Suppose Player 1 has selected H and T with the following frequencies over the PAST ten periods  $t=1, \dots, 10$ 
  - Action H: 5 times
  - Action T: 5 times
- Then Player 2's CURRENT ( $t=11$ ) estimate for the mixed strategy (PDF) being used by Player 1 to choose an action is
  - $\text{Prob}^1(H) = 5/10 = 1/2$
  - $\text{Prob}^1(T) = 5/10 = 1/2$
- Player 2's best response to the estimated PDF  $(1/2, 1/2)$  for Player 1 is the mixed strategy  $\text{Prob}^2(H) = 1/2, \text{Prob}^2(T) = 1/2$ .
- **NOTE:** It can be shown that this pair of *mixed* strategies is actually *the unique Nash equilibrium for the one-shot matching pennies game*.

## Open Issues for FPL

---

- Calculation of estimated PDFs (frequencies) for the action choices of other players is straightforward if all past action choices are observed.
- But how, practically, to calculate a “best response” PDF (mixed strategy) in each time period, given realistic time and cost constraints?
- And what happens if other players are NOT using time-invariant PDFs to choose their action choices?

## Example 2: Experience-Weighted Attraction (EWA) Algorithm (Camerer and Ho, *Econometrica*, 1999)

---

- ❑ Reactive RL assumes agents only take into account actual past rewards, ignoring foregone rewards that might have been obtained had different actions been taken (opportunity costs)
- ❑ FPL assumes agents form opportunity cost estimates to select best-response mixed strategies.
- ❑ EWA is a hybrid form that combines Reactive RL and FPL.

# EWA Algorithm...

- The EWA Algorithm assumes propensities ("attractions") and probabilities ("logit responses") for (mixed) strategy choices are sequentially generated as follows:

$N(t) = \rho N(t-1) + 1$ ,  $N$  is experience weight,  $\rho$  is a discount factor

$$A_i^j(t) = \frac{\phi N(t-1) A_i^j(t-1) + [\delta + (1-\delta) I(s_i^j, s_i(t))] \pi_i(s_i^j, s_{-i}(t))}{N(t)},$$

$A_i^j(t)$  is  $i$ 's attraction for strategy  $j$  at time  $t$ ,  $\phi$  is a decay rate,

$I(s_i^j, s_i(t))$  is an indicator function = 1 if chosen strategy  $s_i(t) = s_i^j$ ,

0 otherwise.  $\pi_i(s_i^j, s_{-i}(t))$  is the payoff from playing  $j$  at time  $t$ .

$\delta$  is the weight on hypothetical payoffs and  $1 - \delta$  is the weight on

actual payoffs. Logit response :  $P_i^j(t+1) = \exp[\lambda A_i^j(t)] / \sum_{k=1}^m \exp[\lambda A_i^k(t)]$ .

$\delta = 0$ ,  $N(0) = 1$ , reinforcement learning;  $\delta = 1$ , weighted fictitious play.

# 4. Anticipatory Learning

---

*Asks....*

If I take this action now, what might happen in the future?

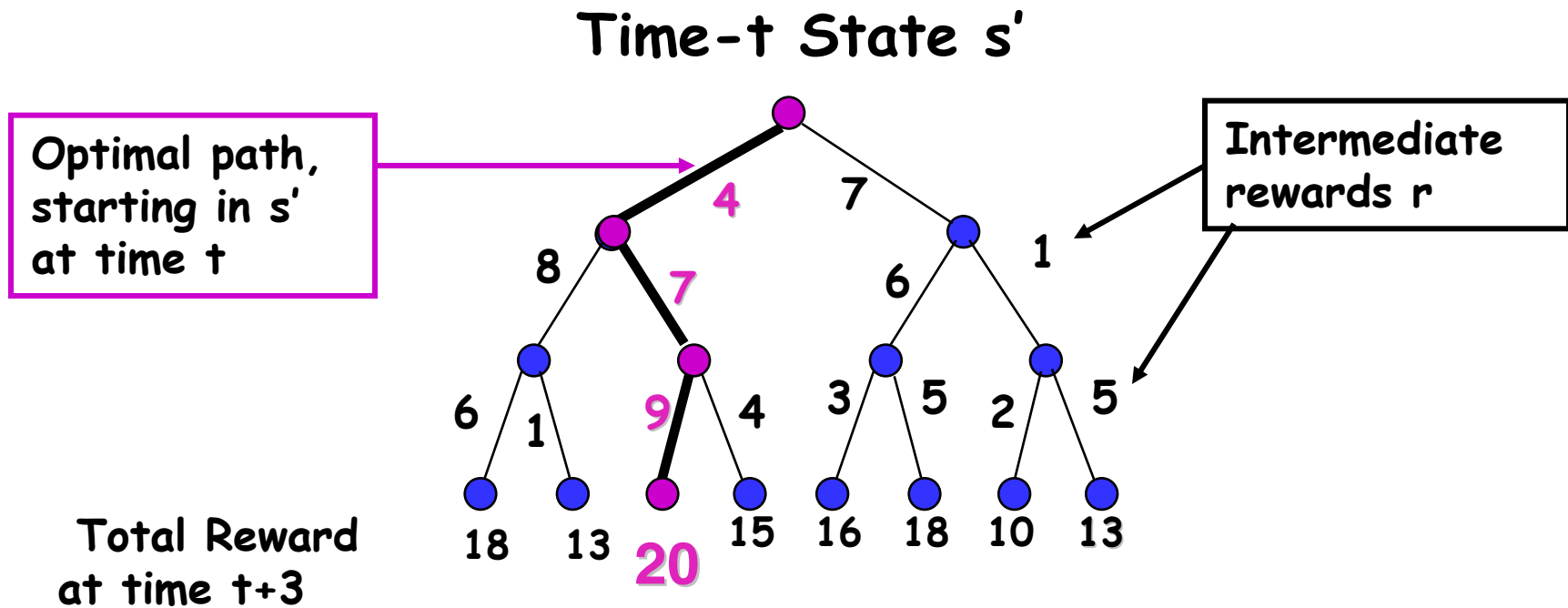
# Key AL Concept: Value Function

---

Let the optimum total reward that can be obtained by an agent starting at time  $t$  in some state  $s'$  be denoted by

$$V_t(s')$$

# Value Function Illustration



Value Function  $V_t$  Gives  $V_t(s') = 20$   
If the Decision Tree Ends at  $[t+3]$   
(Total Reward = Sum of All Intermediate Rewards  $r$ )



# Key Idea: Recursive Relationship Among Value Functions

---

- Suppose I'm currently in state  $s'$  at time  $t$ .
- Suppose I take an action  $a'$ , get a reward  $r' = R(s', a')$ , and transit to a new state  $s'' = T(s', a')$ .
- Then the best I can do starting from time  $t+1$  is

$$V_{t+1}(s'')$$

- Consequently, the best I can do *starting from time  $t$*  is

$$V_t(s') = \max_a [ R(s', a) + V_{t+1}(T(s', a)) ]$$

## More Formally Stated: Richard Bellman's Famous Principle of Optimality (Dynamic Programming, 1950s)

---

- Let  $t$  denote the "current time" and let  $S = \{s, s', \dots\}$  denote the collection of all possible states of the world at time  $t$ .
- For each state  $s$  in  $S$ , let  $A(s) = \{a, a', \dots\}$  denote the collection of all feasible actions that an agent can take in state  $s$  at time  $t$ .
- For each state  $s$  in  $S$ , let  $W$  denote the collection of all possible total rewards  $w$  an agent can attain over current and future times  $t, \dots, T_{\text{Max}}$ .
- Let the **value function**  $V_t: S \rightarrow W$  be defined as follows: For each  $s$  in  $S$ ,  $V_t(s)$  gives the optimum total reward  $w$  in  $W$  that can be attained by the agent over current and future times  $t, \dots, T_{\text{Max}}$  starting in state  $s$  at time  $t$ .

# Principle of Optimality...Continued

---

- Let  $\pi^*$  denote the **optimal policy function** giving the optimal action  $a'$  as a function  $a'=\pi^*(t,s')$  of the current time  $t$  and state  $s'$ .
- Let  $T$  denote the **transition function** that determines the next state  $s''$  as a function  $s''=T(s',a')$  of the current state  $s'$  and the current action choice  $a'$ .
- Let  $R$  denote the **intermediate return function** that determines the immediate reward  $r''$  as a function  $r''=R(s',a')$  of the current state  $s'$  and current action choice  $a''$ .
- Then for each state  $s'$  in  $S$ :

$$\begin{aligned} V_t(s') &= R(s',\pi^*(t,s')) + V_{t+1}( T(s',\pi^*(t,s')) ) \\ &= \text{Max}_a [ R(s',a) + V_{t+1}(T(s',a)) ] \end{aligned}$$

# Practical Difficulties

---

- ❑ How practically to compute the Optimal Policy Function  $\pi^*$  ?
- ❑ What if the Transition Function  $T$  is not known?  
And what if state transitions depend on actions chosen by **MANY** agents, not just by me?
- ❑ What if the Return Function  $R$  is not known?
- ❑ How practically to compute the Value Function  $V$ ?

# One Approach: Replace V-values by Q-values

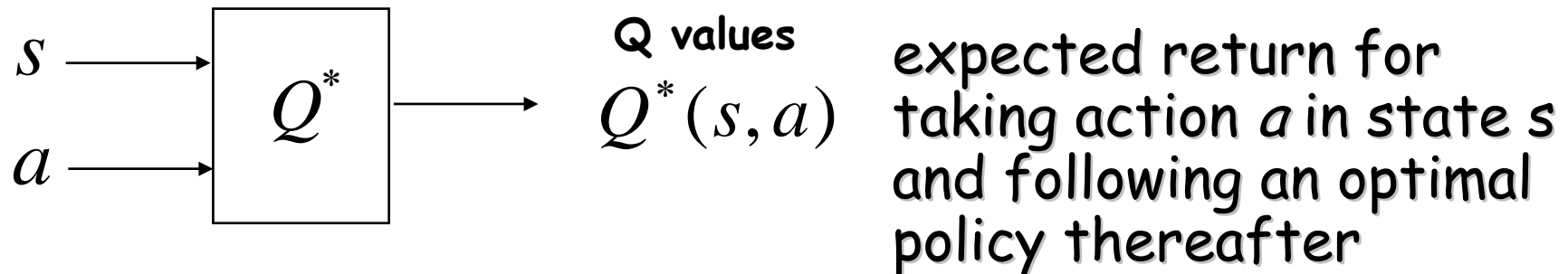
---

- ✿ Suppose the final time TMax is infinite and suppose that  $\pi^*$ , T, R, and V are independent of time t (strong assumption)
- ✿ For each s in S and each a in A(s), define

$$Q^*(s,a) = [ R(s,a) + V(T(s,a)) ]$$

- ✿ If these Q\*-values can be learned, the Optimal Policy Function  $\pi^*$  can be found without knowing the T, R, and V functions, as follows: For any s' in S,  
 $\pi^*(s') = \text{action } a' \text{ that maximizes } Q^*(s',a) \text{ over } a \text{ in } A(s')$
- ✿ But will  $\pi^*$  result in good action choices if state/reward outcomes actually depend on actions of multiple agents?

## Q-Learning in More Detail (Watkins 1989; see also criterion filtering, [www.econ.iastate.edu/tesfatsi/cfhome.htm](http://www.econ.iastate.edu/tesfatsi/cfhome.htm))



For any state  $s$ , any action  $a^*$  that maximizes  $Q^*(s, a)$  is called an **optimal action**:

$$a^* = [\text{optimal action in state } s] = \arg \max_a Q^*(s, a)$$

Let  $Q(s, a) =$  current estimate of  $Q^*(s, a)$

# Q-Learning ...

---

Q-learning in simplest form iteratively determines estimates  $Q(s,a)$  for  $Q^*(s,a)$  conditional on a user-specified learning rate  $\alpha$  ( $0 \leq \alpha \leq 1$ ).

- Q-learning avoids direct calculation of  $T, R, V$
- The Q-value estimates  $Q(s,a)$  are stored in a table
- The Q-value estimates are updated after each new observation is obtained.
- The Q-value estimates depend on observation history but not directly on the particular method used to generate action choices.

# Basic Q-Learning Algorithm

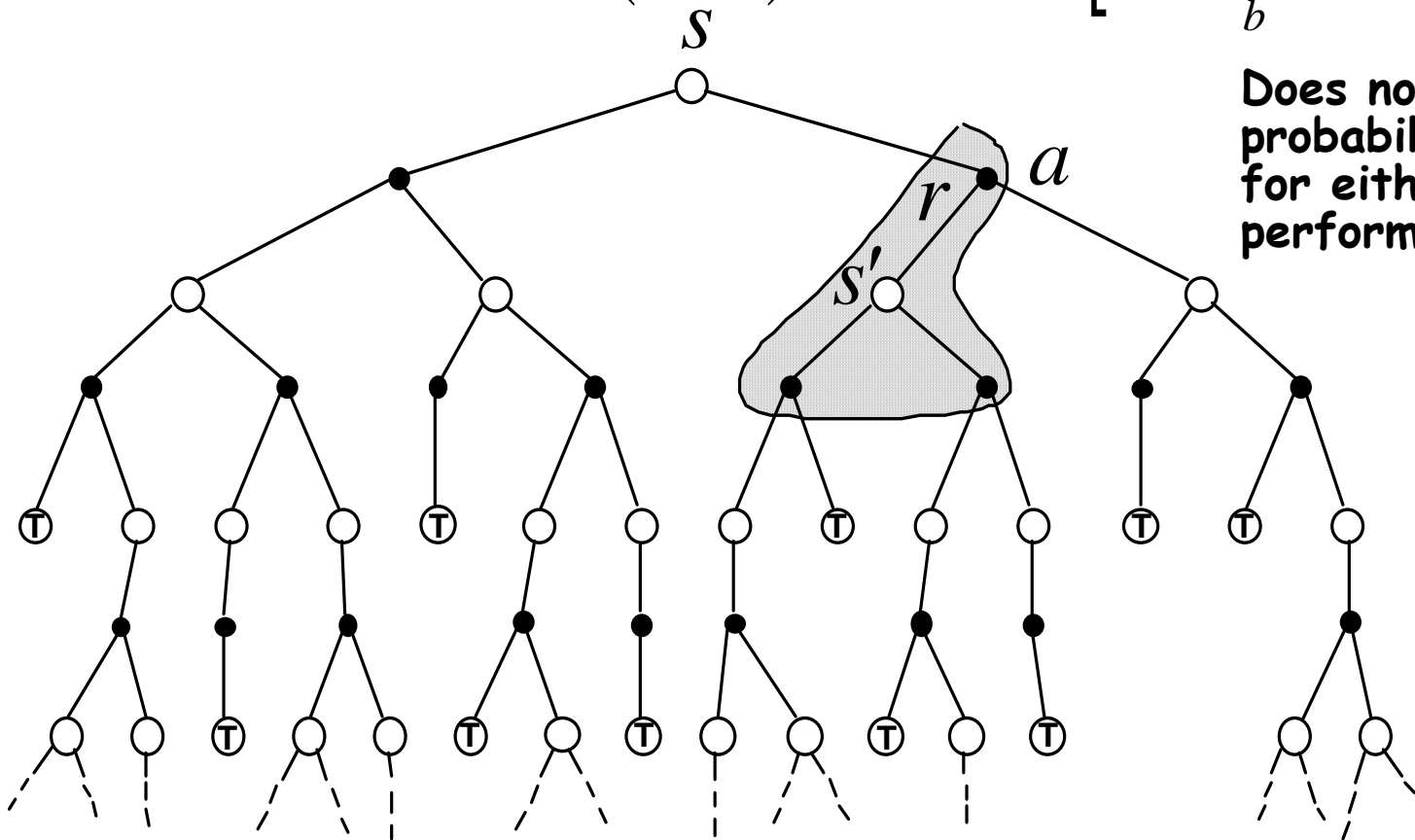
---

1. Initialize  $Q(s,a)$  to a random value for each state  $s$  in  $S$  and each action  $a$  in  $A(s)$ .
2. Observe actual state  $s'$ .
3. Pick an action  $a'$  in  $A(s')$  and implement it.
4. Observe next state  $s''$  and next reward  $r''$ .
5. Update  $Q(s',a')$  value as follows:  
$$Q(s',a') \leftarrow [1 - \alpha]Q(s',a') + \alpha[ r'' + \max_a Q(s'',a) ]$$
- 6 . Loop back to step 2



# Q-Learning Update Process

$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha \left[ r + \max_b Q(s', b) \right]$$



Does not need a probability model for either learning or performance evaluation

# Picking Actions for Q-Learning

---

- Just as in reactive RL, an agent might want to pick “optimal” actions most of the time but also do some exploration.
  - An agent can exploit its *current* information state  $s$  to choose a “greedy” action  $a$  in  $A(s)$  that *currently* appears to be optimal.
  - But, the agent might also choose an action for exploratory purposes, to learn more about its choice environment.
  - Exploring might permit the agent to learn a better policy  $\pi: s \rightarrow a(s)$  for determining *future* action choices.
  - This is called the *exploration/exploitation problem*

# Picking Actions for Q-Learning ...

---

## □ $\epsilon$ -Greedy Approach

- Given state  $s$ , choose an action  $a$  in  $A(s)$  with the highest value  $Q(s,a)$  with probability  $1-\epsilon$  and explore (pick a random action) with probability  $\epsilon$

## □ Gibbs-Boltzmann (soft-max) approach

- Given state  $s$ , pick action  $a$  in  $A(s)$  with probability

$$P(a|s) = \frac{e^{\left(\frac{Q(s,a)}{\tau}\right)}}{\sum_{a'} e^{\left(\frac{Q(s,a')}{\tau}\right)}} \quad , \quad \text{where } \tau = \text{"temperature"}$$

# 5. Evolutionary Learning

---

## Asks...

Given all the actions that have been taken to date by myself (and possibly by others), together with observations on the rewards that have resulted, what **NEW** actions might I devise to try to do better?

# Evolutionary Learning Algorithms

---

## EXAMPLES...

- Genetic Algorithm (GA) - John Holland 1970s
- Genetic Programming (GP) - John Koza 1990s
- Evolutionary Strategy (ES) - Rechenberg 1970s
- Evolutionary Program (EP) ... Etc.

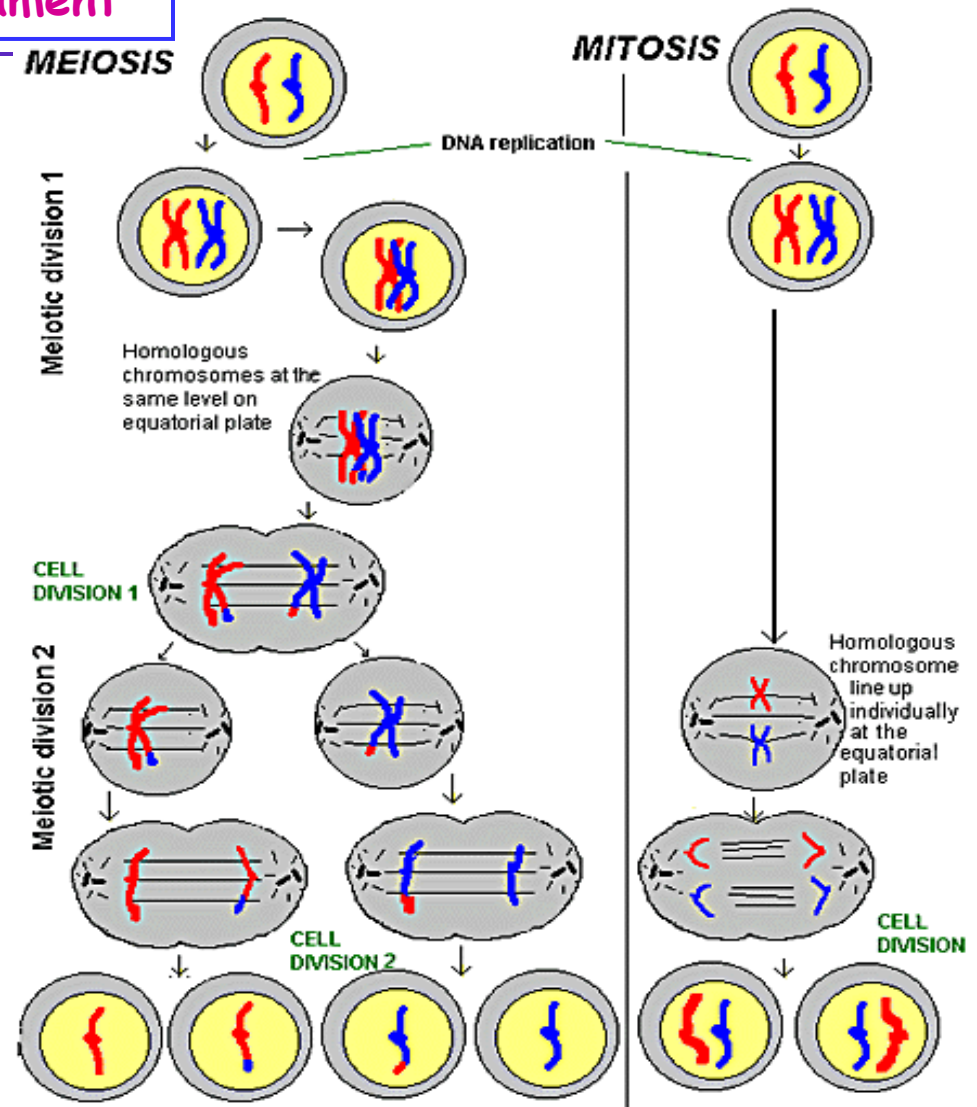
***Basic Idea:*** Devise learning algorithms for complex environments that mimic effective adaptive and evolutionary processes found in nature.

# Evolutionary Processes in Nature: Mitosis vs. Meiosis

## Replication as in Axelrod Evol Tournament

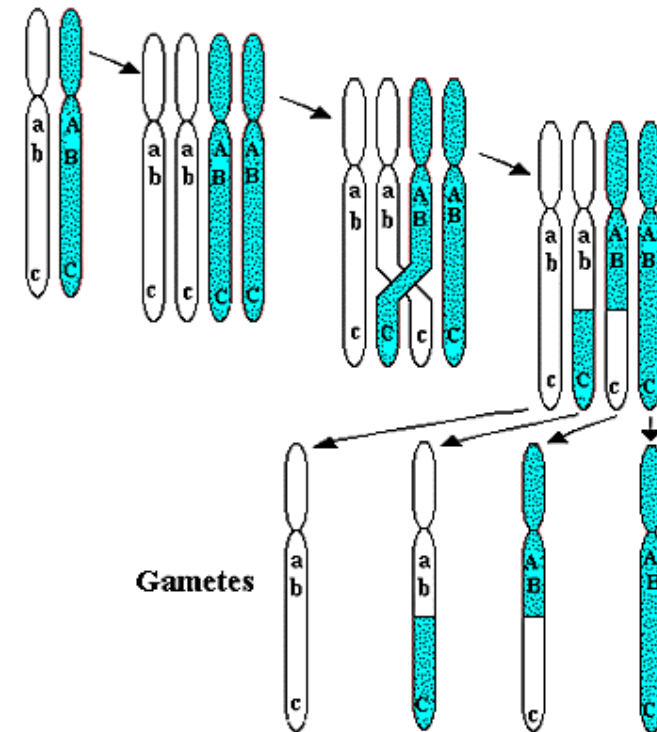
- Mitosis: one cell becomes two cells with the same DNA (cloning)
- Meiosis: one cell becomes four cells with one strand each (basis for sexual reproduction)

Permits "Genetic Evolution"!



# Crossover (Recombination)

- Meiosis -> production of germ cells
- Parts of two chromosomes get swapped.
- Also called recombination



Crossing-over and recombination during meiosis

# Mutation

---

- ❑ Occasional misfiring of the replication process.
- ❑ Almost always harmful.
- ❑ On occasion, leads to "fitter" entity.





# Differential Survival

---

- Once there is variability (through sexual reproduction, crossover and mutation) in a population, the environment culls some members of the population while others survive.
- This process is termed *Natural Selection*.

# Evolutionary Learning Algorithm Example: Genetic Algorithms (GAs)

---

- Directed search algorithm based on the mechanics of biological evolution
- Developed by John Holland, University of Michigan (1970's)
- **Original Goal:**
  - To use adaptive and evolutionary processes found in natural systems as a metaphor for the design of *effective search algorithms* suitable for complex environments

# Basic Steps of a Simple GA

---

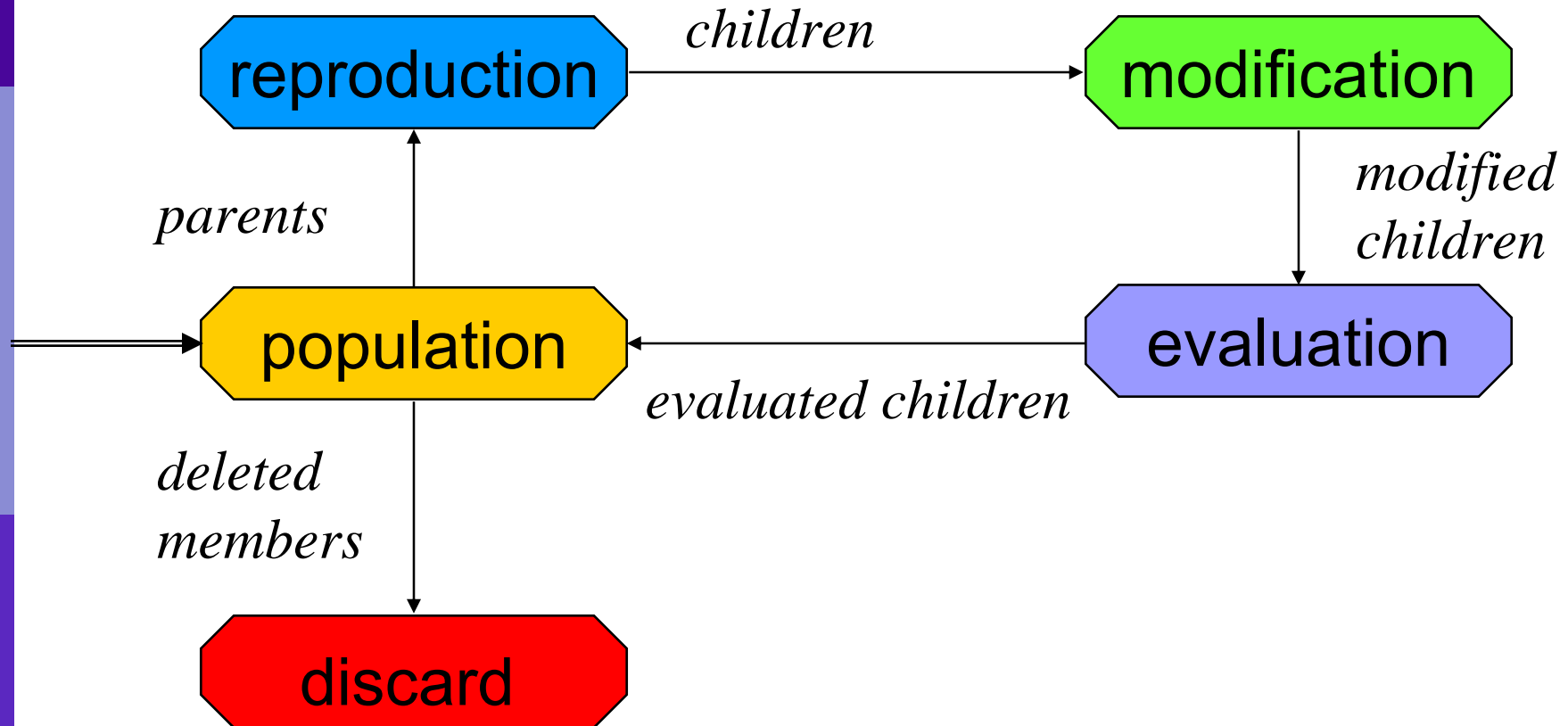
**Step 0:** Construct/configure an initial population of members (agents, strategies, candidate solutions to a problem, ...).

**Step 1:** Evaluate the "fitness" of each member of the current population, and discard least fit members.

**Step 2:** Apply "genetic operations" (e.g. "mutation," "recombination,"...) to the remaining (parent) population to generate a new (child) population to replace discarded least-fit population members.

**Step 3:** Loop back to Step 1 and repeat.

# The GA Cycle of Reproduction



# What Might "Fitness" Mean?

---

## EXAMPLES...

- ❖ The ability to solve a particular problem (e.g. a given math problem)
- ❖ The ability to repeatedly perform some task (e.g., facial recognition)
- ❖ The ability to survive and prosper in some real or computational environment

# Representation of Population Members

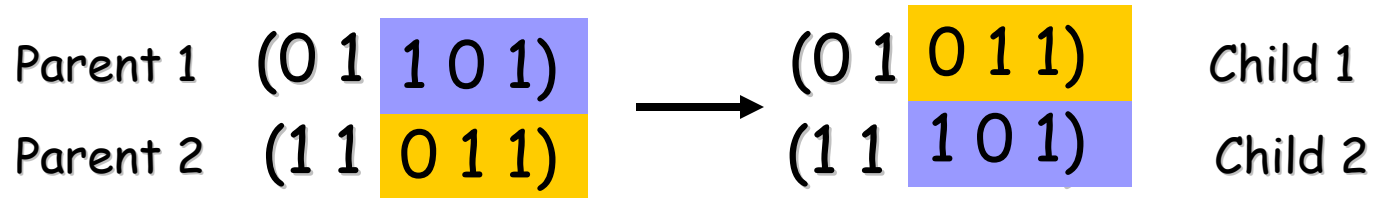
---

**EXAMPLE:** Bit-String Representation (String of 0's & 1's)

- Population Members = PD Game Strategies
- One Possible Strategy  $\mathcal{S}$ 
  - State = (My last play, Rival's last play)
  - Two Possible Actions: Cooperate=1, Defect=0
  - Four Possible States: 1=(1,1), 2=(1,0), 3=(0,1), 4=(0,0)
  - Strategy  $\mathcal{S}$  = TFT:
    - Start by choosing Action 1
    - If State 1, then choose Action 1
    - If State 2, then choose Action 0
    - IF State 3, then choose Action 1
    - IF State 4, then choose Action 0
- Bit-string representation of Strategy  $\mathcal{S}$ : (1 | 1 | 0 | 1 | 0)

# Crossover (Recombination)

---



Crossover is a potentially critical feature of *GAs*:

- It can greatly accelerate search early in the evolution of a population
- It can lead to discovery and retention of effective combinations (blocks, schemas,...) of  $S \rightarrow A$  associations

# Mutation of Population Members

## Example: String Mutations

Before: (1 0 1 1 0 )

After: (1 0 1 0 0 )

Before: (1.38 -69.4 326.44 0.1)

After: (1.38 -67.5 326.44 0.1)

- Causes local or global movement in search space
- Can restore lost information to the population



# Issues for GA Practitioners

---

## □ Basic implementation issues

- Representation of population members
- Population size, mutation rate, ...
- Selection, deletion policies
- Crossover, mutation operators

## □ Termination criteria

- When is a solution good enough?

## □ Fitness Function Specification

- "Solution" depends heavily on the fitness function (specification of "fitness" often the hardest part)

# Types of GA Applications

<b>Domain</b>	<b>Application Types</b>
<b>Control</b>	gas pipeline, pole balancing, missile evasion, pursuit
<b>Design</b>	semiconductor layout, aircraft design, keyboard configuration, communication networks
<b>Scheduling</b>	manufacturing, facility scheduling, resource allocation
<b>Robotics</b>	trajectory planning
<b>Machine Learning</b>	designing neural networks, improving classification algorithms, classifier systems
<b>Signal Processing</b>	filter design
<b>Game Playing</b>	poker, checkers, prisoner's dilemma
<b>Combinatorial Optimization</b>	set covering, travelling salesman, routing, bin packing, graph colouring and partitioning

# 6. Connectionist Learning

---

## Asks...

Does the learning of state-act associations  $s \rightarrow a$  ("if  $s$ , then  $a$ ") require a centralized information processor, or can it proceed through some form of decentralized information processor?

And can the appropriate specification of the conditioning states  $s$  be learned along with the appropriate specification of the associations  $s \rightarrow a$  ?

# Connectionist Learning Example

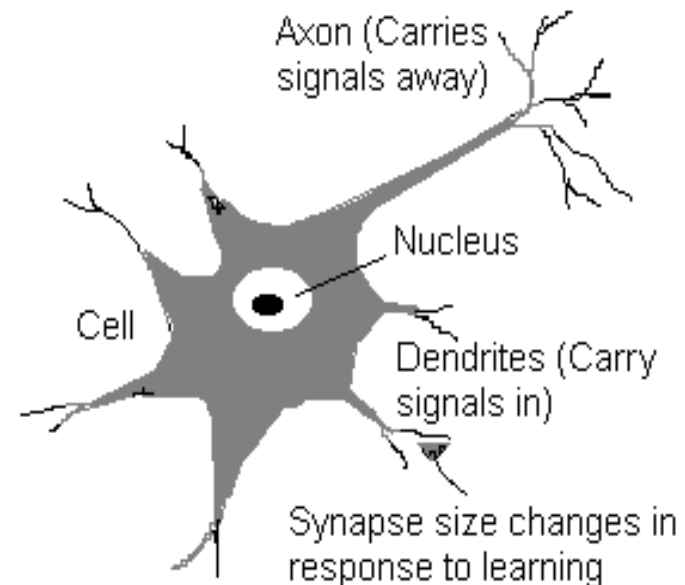
---

## Artificial Neural Networks (ANNs):

Decentralized information processing paradigm inspired by biological nervous systems, such as the human brain

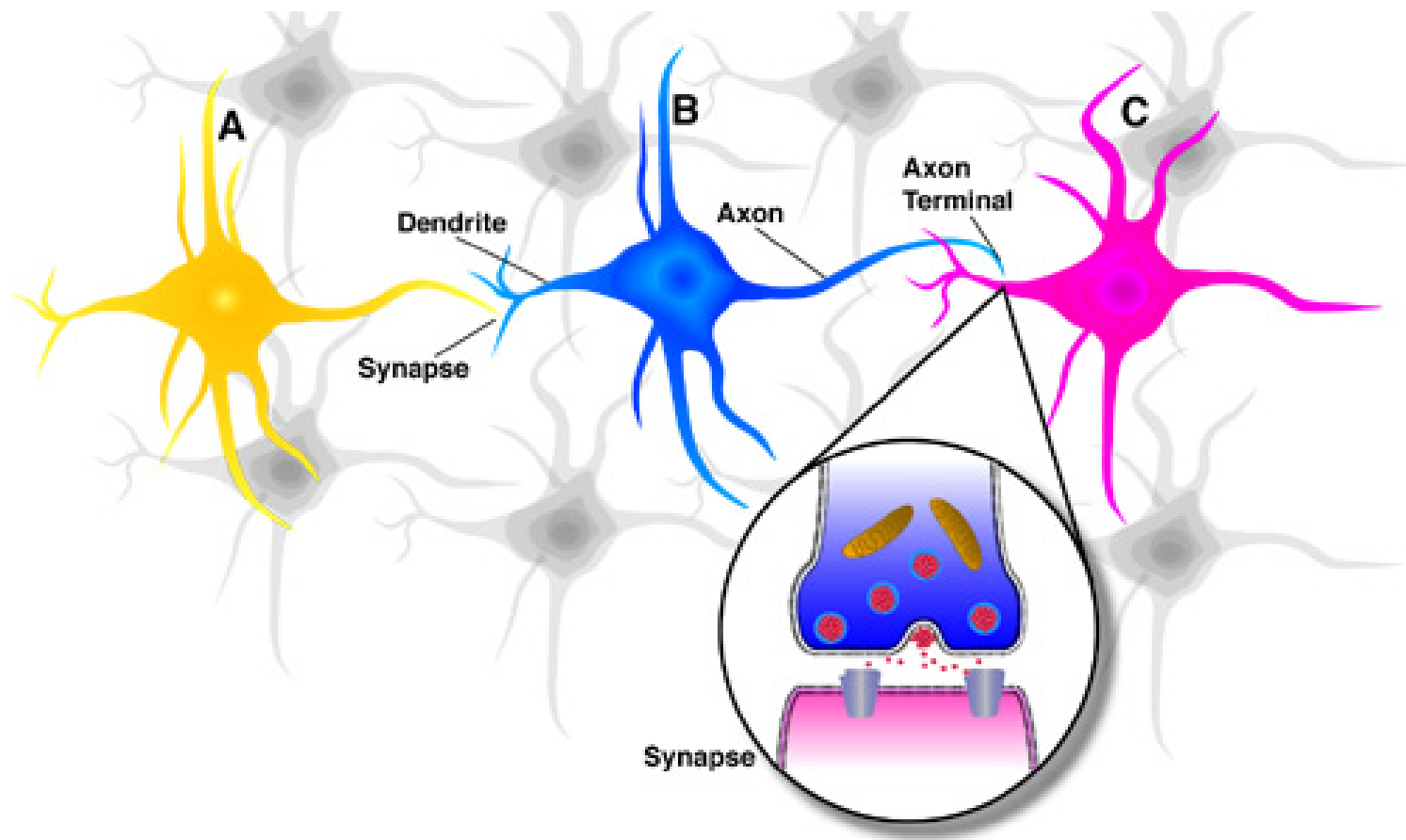
# Inspiration from Neurobiology

- **Neuron** : A many-inputs/one-output unit forming basis of human central nervous system
- Output can be *excited* or *not excited*
- Incoming signals from other neurons determine if the neuron shall *excite* ("fire")
- Output subject to attenuation in the **synapses** (small gaps) that separate a neuron from other neurons at the juncture of its axon with their dendrites



# Connections Among Neurons

---

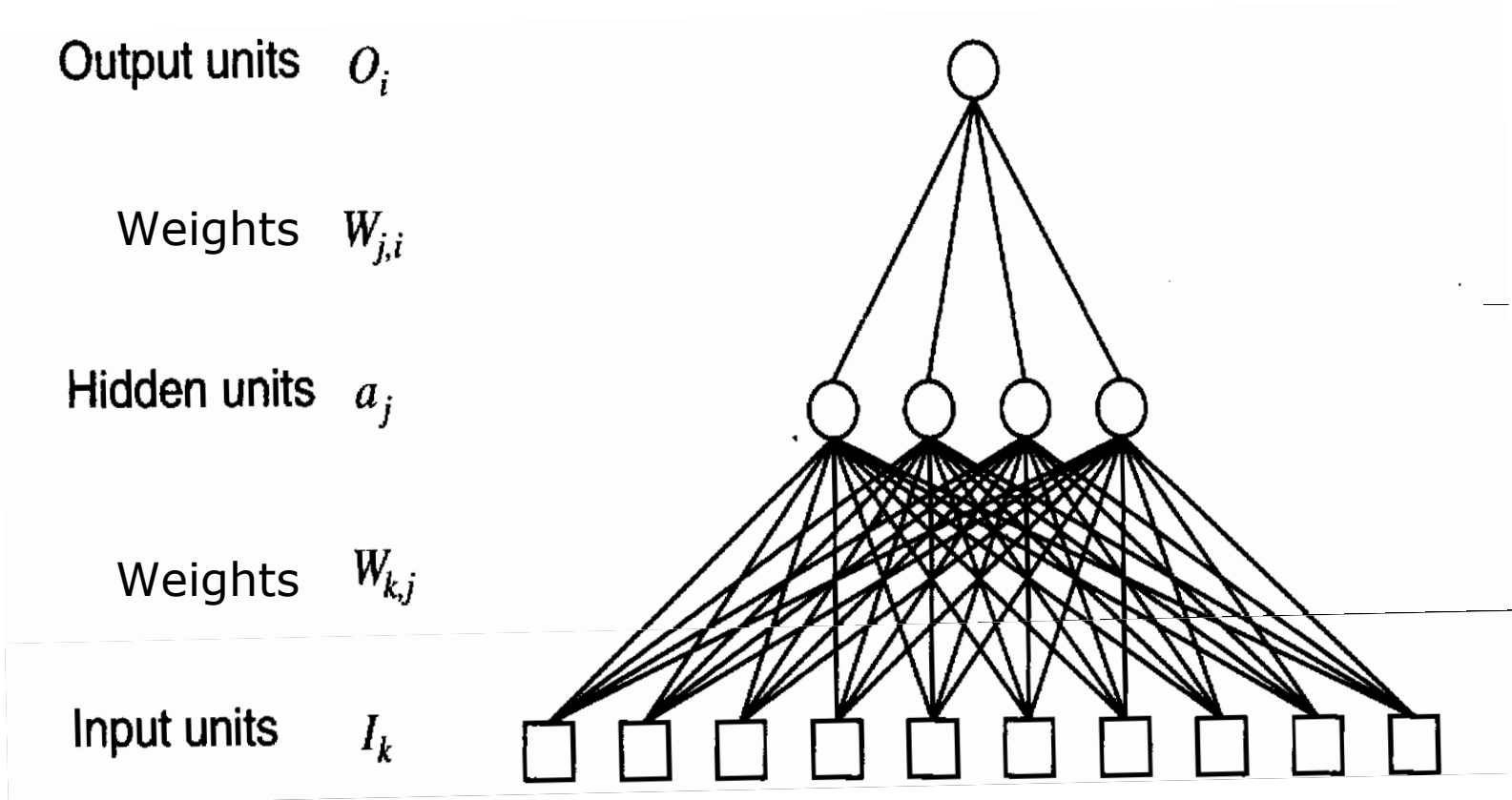


# Structure of ANNs

---

- Collection of interconnected processing units working together
- **Structure** = (1) Unit configuration (numbers of input units, hidden units, and output units); (2) Unit connections; & (3) Connection weights
- Structure can be updated via unsupervised learning, RL, or supervised learning

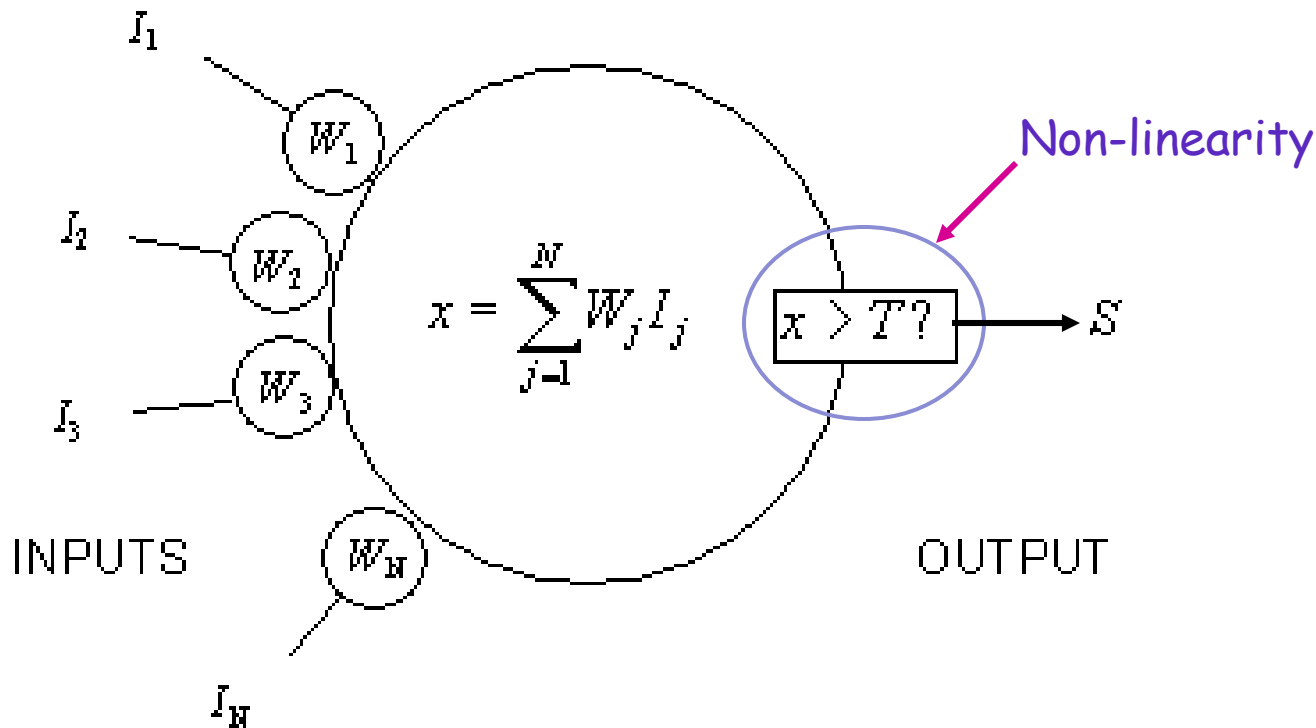
# Example: Feedforward ANN (No recurrent loops)





# Hidden Unit Representation

**Example:** The hidden unit depicted below calculates a weighted sum  $x$  of inputs  $I_j$  and compares it to a threshold  $T$ . If  $x$  is higher than the threshold  $T$ , the output  $S$  is set to 1, otherwise to -1.

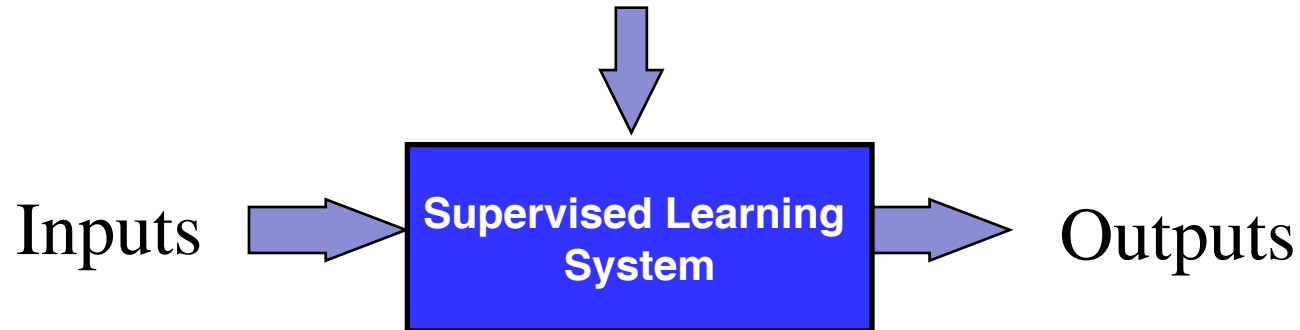


# ANN Supervised Learning

(Learn from a set of examples via **error-correction**)

---

Training Examples = Desired Input-Output Associations



**Error** = [Desired Output - Actual Output]

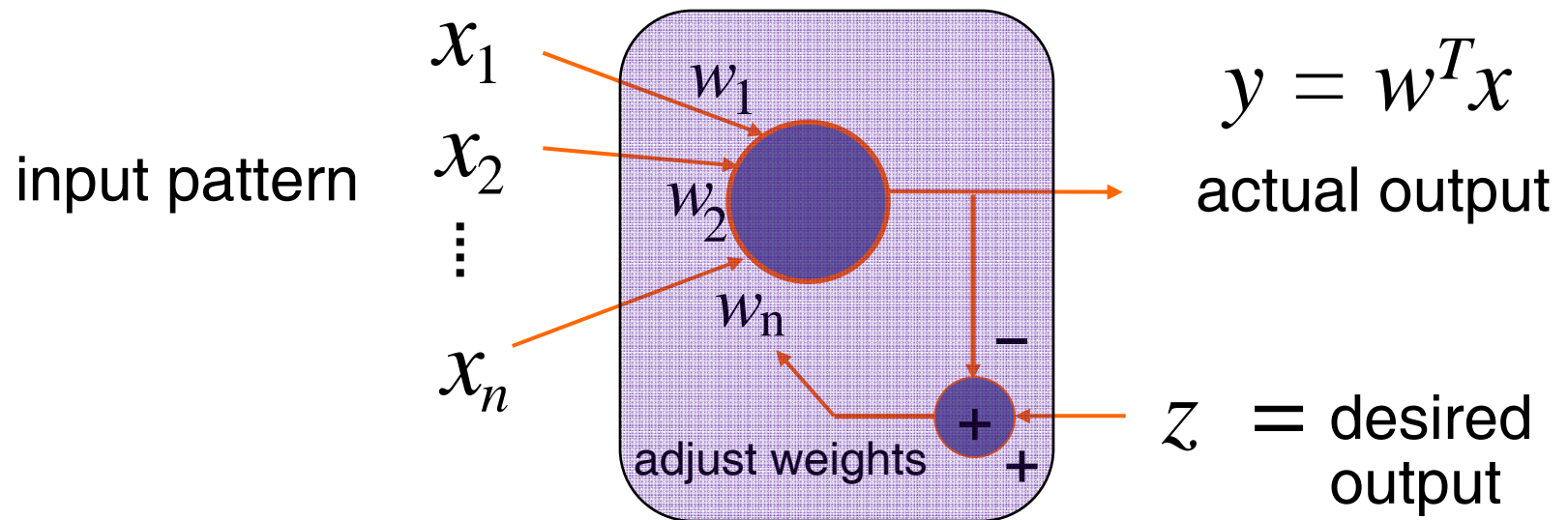
# ANN Supervised Learning via "Back Propagation"

---

- Desired input-output associations provided by supervisor through training examples
- **Error** = Difference between desired and actual output for any given input
- Weights updated relative to error size
- Start by calculating output layer error and weight correction, then "propagate back" through previous layers

# Example: "Adaline" Learning Rule

Widrow and Hoff, 1960



$$\Delta w_i = \alpha [z - y] x_i$$

# Illustrative ANN Applications

---

- **Prediction:** Learning from past experience
  - *pick the best stocks in the market*
  - *predict weather*
  - *identify people with cancer risk*
  
- **Classification**
  - *Image processing*
  - *Predict bankruptcy for credit card companies*
  - *Risk assessment*

# ANN Applications...Continued

---

## □ Recognition

- *Pattern recognition: SNOOPE (bomb detector in U.S. airports)*
- *Character recognition*
- *Handwriting recognition (processing checks)*

## □ Data Association

- *Identify scanned characters AND detect if scanner is working properly*

# ANN Applications...Continued

---

## □ Data Conceptualization

- *infer grouping relationships*  
e.g. extract from a database the names of those most likely to buy a particular product.

## □ Data Filtering

*e.g. take the noise out of a telephone signal*

## □ Planning

- *Evolve "best" decisions for unknown environments*
- *Evolve "best" decisions for highly complex environments*
- *Evolve "best" decisions given highly noisy input data*