Lecture 16: Introduction to Reinforcement Learning

- The reinforcement learning problem
- Brief history and example applications
- Markov Decision Processes
- What to learn: policies and value functions

Control Learning

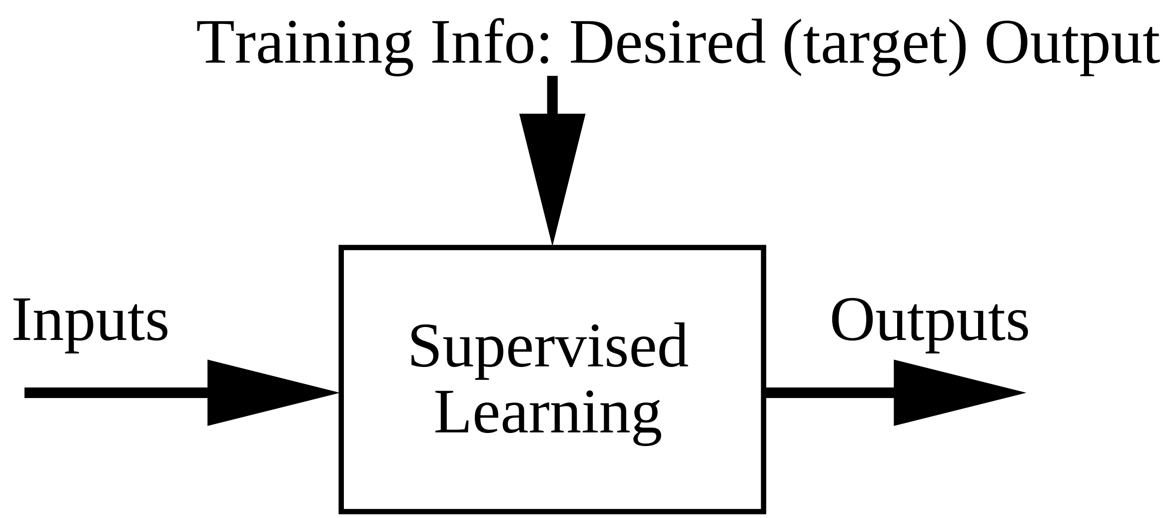
Consider *learning to choose actions*, e.g.,

- Robot learning to dock on battery charger
- Learning to choose actions to optimize factory output
- Learning to play Backgammon

Specific problem characteristics:

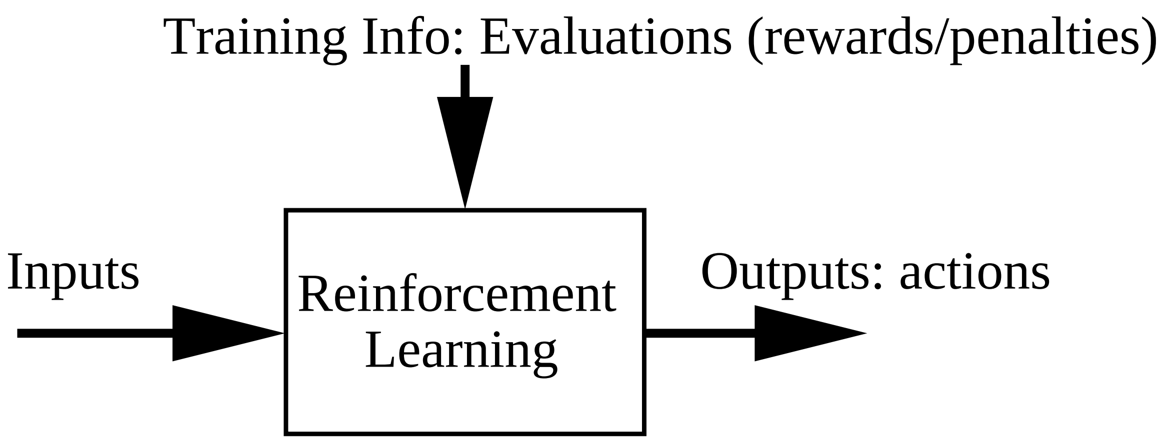
- Delayed reward
- Opportunity for active exploration
- There may not exist an adequate teacher!
- May need to learn multiple tasks using the same sensors/effectors

Supervised Learning



Error = (target output - actual output)

Reinforcement Learning (RL)



Objective: Get as much reward as possible

Key Features of RL

- The learner is not told what actions to take
- It find finds out what to do by trial-and-error search
- Possibility of delayed reward: sacrifice short-term gains for greater long-term gains
- Need to *explore* and *exploit*
- The environment is stochastic and unknown

Brief History

- Minsky's PhD thesis (1954): Stochastic Neural-Analog Reinforcement Computer
- Samuel's checkers player (1959)
- Ideas about state-action rewards from animal learning and psychology
- Dynamic programming methods developed in operations research (Bellman)
- Died down in the 70s (along with much of the learning research)
- Temporal difference (TD) learning (Sutton, 1988), for prediction
- Q-learning (Watkins, 1989), for control problems
- TD-Gammon (Tesauro, 1992) - the big success story
- Evidence that TD-like updates take place in dopamibne neurons in the brain (W.Schultz et.al, 1996)
- Currently a very active research community, with links to different fields

Success Stories

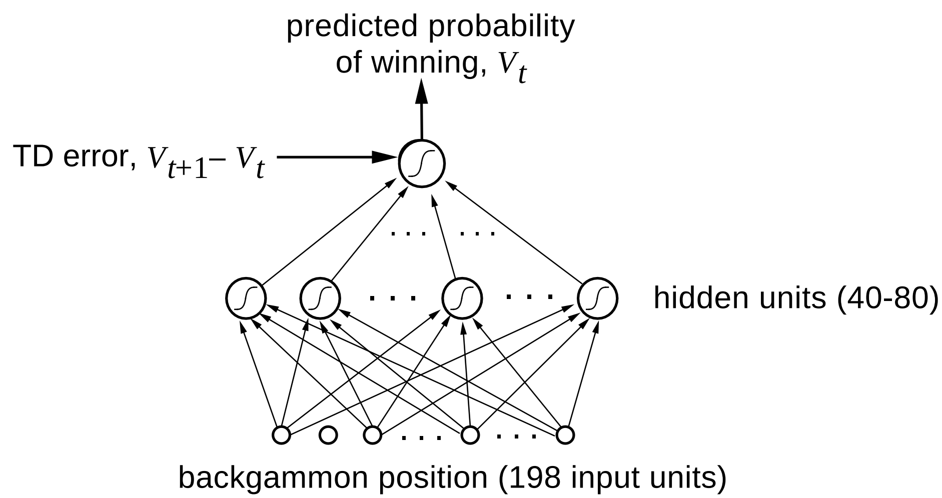
- TD-Gammon (Tesauro, 1992)
- Elevator dispatching (Crites and Barto, 1995): better than industry standard
- Inventory management (Van Roy et. al): 10-15% improvement over industry standards
- Job-shop scheduling for NASA space missions (Zhang and Dietterich, 1997)
- Dynnamic channel assignement in cellular phones (Singh and Bertsekas, 1994)
- Learning walking gaits in a legged robot (Huber and Grupen, 1997)
- Robotic soccer (Stone and Veloso, 1998) - part of the world-champion approach

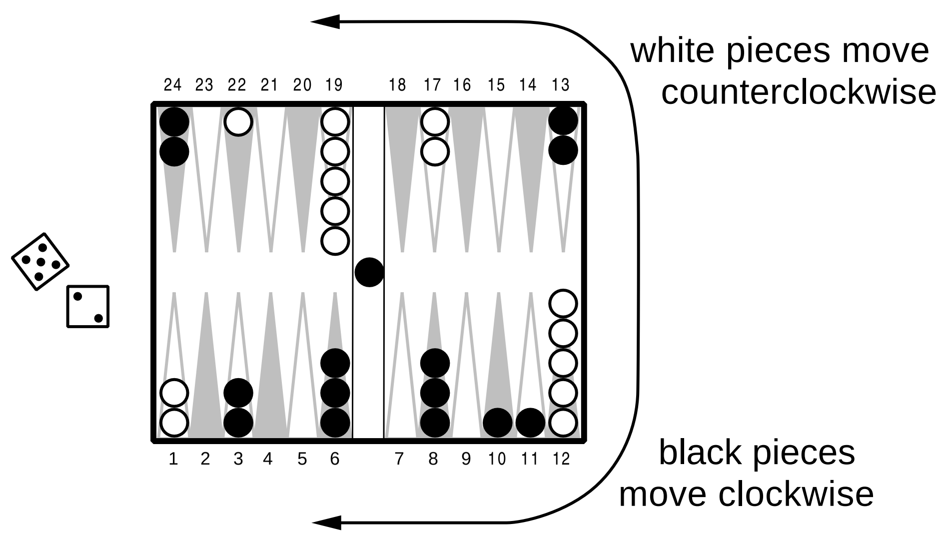
All these are *large, stochastic optimal control problems*:

- Conventional methods require the problem to be simplified
- *RL just finds an approximate solution!*

An approximate solution can be better than a perfect solution to a simplified problem

TD-Gammon (Tesauro, 1992-1995)





TD-Gammon: Training Procedure

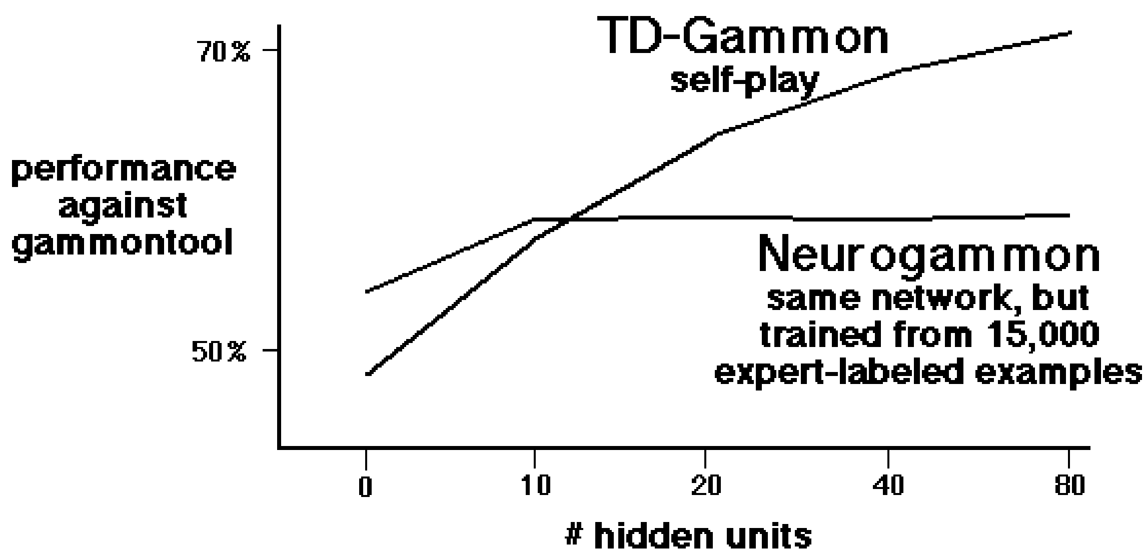
Immediate reward:

- +100 if win
- -100 if lose
- 0 for all other states

Trained by playing 1.5 million games *against itself*

Now approximately equal to best human player

The Power of Learning from Experience

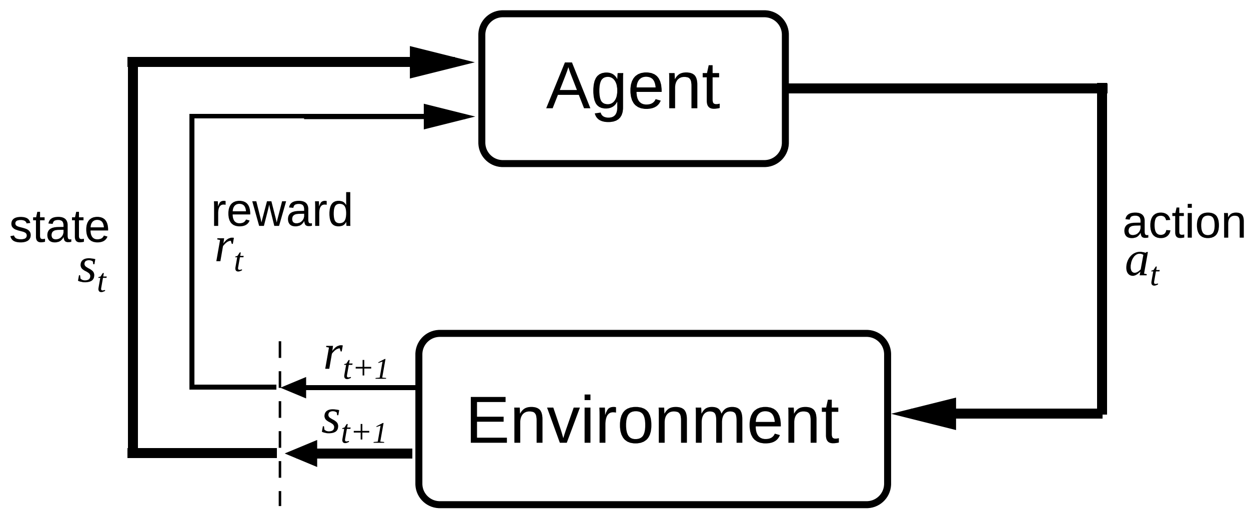


Tesauro, 1992

Expert examples are expensive and scarce

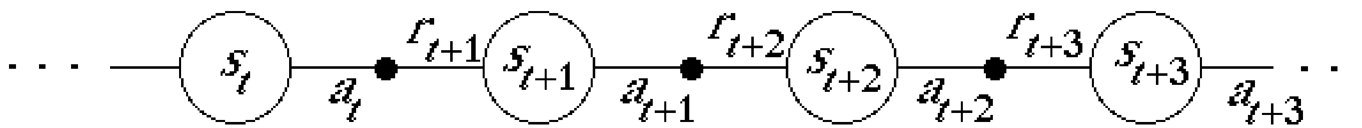
Experience is cheap and plentiful!

Reinforcement Learning Problem



- At each discrete time t, the agent observes state $s_t \in S$ and chooses action $a_t \in A$
- Then it receives an immediate reward r_{t+1} and the state changes to s_{t+1}

Markov Decision Processes (MDPs)



Assume:

- Finite set of states S (we will lift this later)
- Finite set of actions $A(s)$ available in each state s
- γ = discount factor for later rewards (between 0 and 1, usually close to 1)
- *Markov assumption:* s_{t+1} and r_{t+1} depend only on s_t, a_t and not on anything that happened before t

Models for MDPs

- r_s^a = expected value of the immediate reward if the agent is in s and does action a

$$r_s^a = E_{r_{t+1}}\{s_t = s, a_t = a\}$$

- $p_{ss'}^a$ = probability of going from s to s' when doing action a

$$p_{ss'}^a = E_{s_{t+1}=s'}\{s_t = s, a_t = a\}$$

These form the *model* of the environment, and are *usually unknown*

Agent's Learning Task

Execute actions in environment, observe results, and **learn policy** $\pi : S \times A \rightarrow [0, 1]$,

$$\pi(s, a) = \text{Pr}\{a_T = a\} s_t = s$$

- Note that the target function is $\pi : S \rightarrow A$ but we have **no training examples** of form $\langle s, a \rangle$
  Training examples are of form $\langle \langle s, a \rangle, r \ldots \rangle$
- Reinforcement learning methods specify how the agent should change the policy as a function of the rewards received over time
- Roughly speaking, the agent's goal is to get as much reward as possible *in the long run*

Returns

Suppose the sequence of rewards received after time step t is $r_{t+1}, r_{t+2} \ldots$. We want to maximize the **expected return** $E\{R_t\}$ for every time step t

- Episodic tasks: the interaction with the environment takes place in episodes (e.g. games, trips through a maze etc)

$$R_t = r_{t+1} + r_{t+2} + \cdots + r_T$$

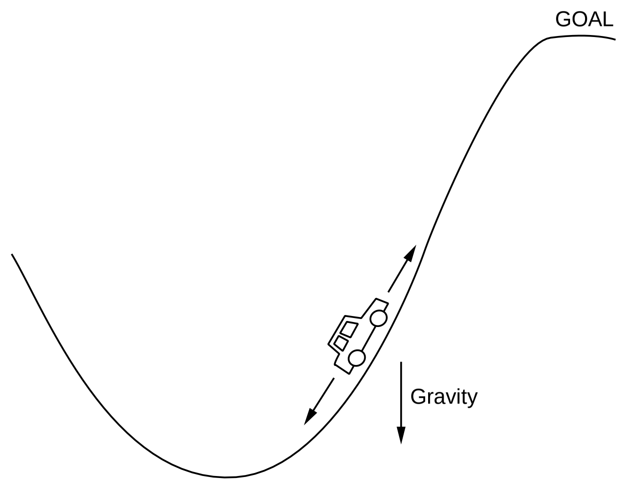
  where T is the time when a terminal state is reached

- Continuing tasks:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = \sum_{k=1}^{\infty} \gamma^{t+k-1} r_{t+k}$$

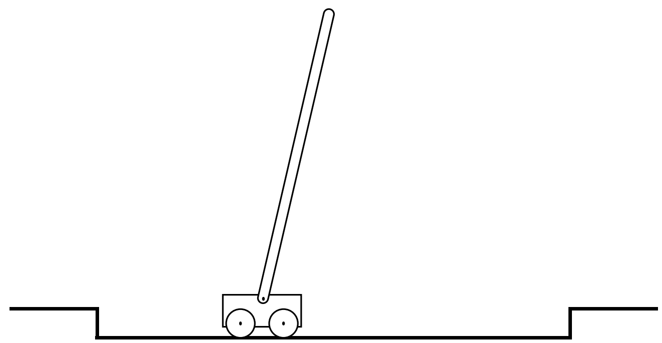
  where $0 \leq \gamma < 1$ is the discount factor for future rewards

Example: Mountain-Car



- States: position and velocity
- Actions: accelerate forward, accelerate backward, coast
- Rewards:
  - reward = -1 for every time step, until car reaches the top
  - reward = 1 at the top, 0 otherwise $\gamma < 1$
- Return is maximized by minimizing the number of steps to the top of the hill

Example: Pole Balancing



Avoid failure: pole falling beyond a given angle, or cart hitting the end of the track

- Episodic task formulation: reward = +1 for each step before failure

  $\Rightarrow$ return = number of steps before failure

- Continuing task formulation: reward = -1 upon failure, 0 otherwise, $\gamma < 1$

  $\Rightarrow$ return = $-\gamma^k$ if there are k steps before failure