# Reinforcement Learning

Lecture 8

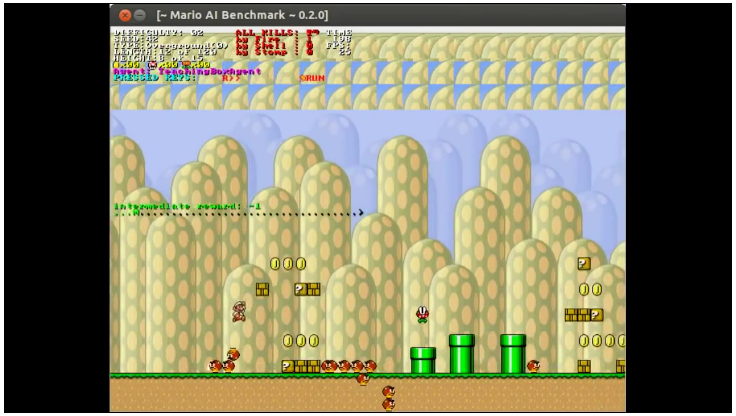
# Outline

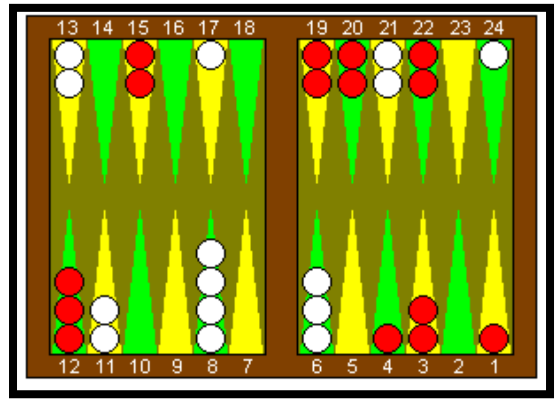
1. Context
2. TD Learning
3. Issues

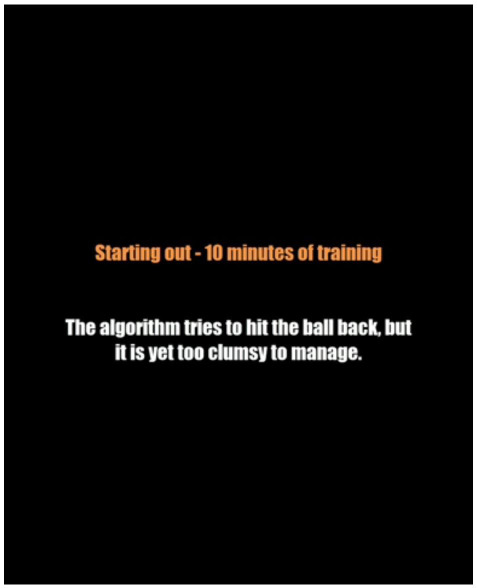
# Machine Learning Tasks

- ***Supervised***
  - Given a ***training set*** and a target variable, ***generalize***; measured over a ***testing set***
- ***Unsupervised***
  - Given a dataset, find "interesting" patterns; potentially no "right" answer
- ***Reinforcement***
  - Learn an optional action ***policy*** over time; given an environment that provides states, affords actions, and provides feedback as numerical ***reward***, maximize the *expected* future reward
    - Never given I/O pairs
    - Focus: online (balancing exploration/exploitation)

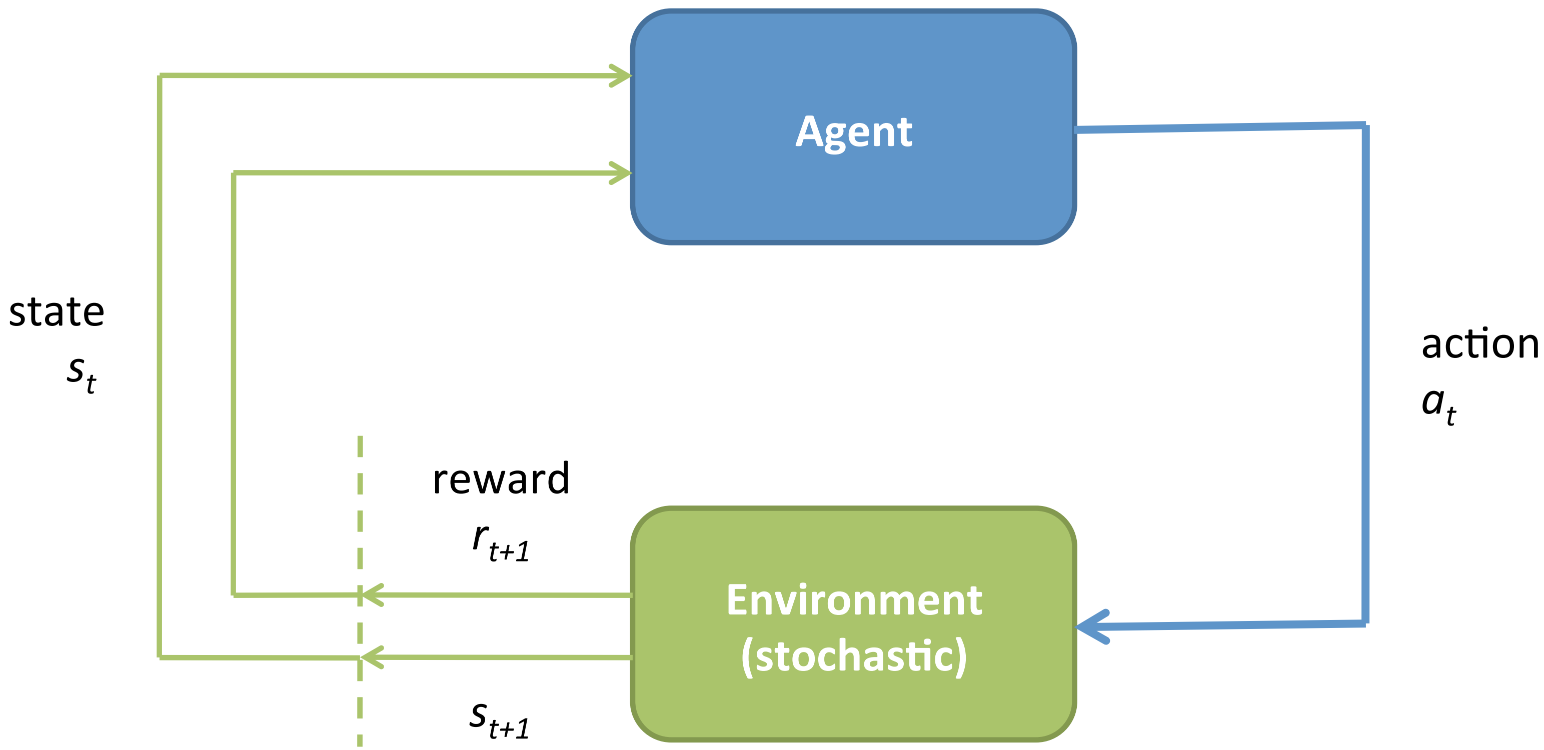
# Success Stories



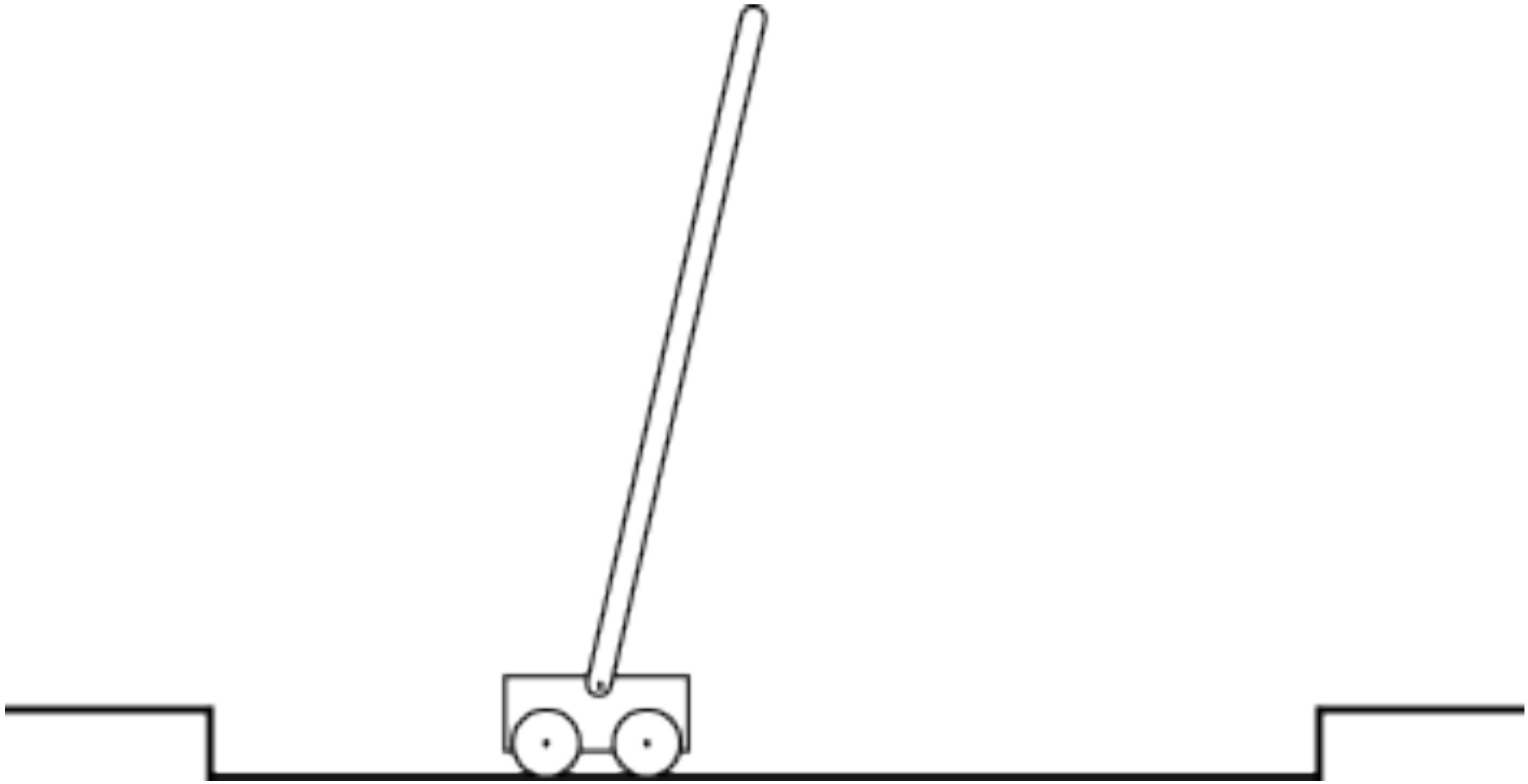




# The Agent-Environment Interface



# Pole Balancing



# Multi-Armed Bandit

# Types of Tasks

- Some tasks are *continuous*, meaning they are an ongoing sequence of decisions

- Some tasks are *episodic*, meaning there exist *terminal* states that reset the problem

# Policies

A ***policy*** is a function that associates a probability with taking a particular action in a particular state

$$\pi(s, a)$$

The goal of RL is to *learn* an “effective” policy for a particular task

# Objective

Select actions so that the sum of the discounted rewards it receives over the future is maximized

- Discount rate: $0 \leq \gamma \leq 1$

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots$$
$$= \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

# Environmental Modeling

- An important issue in RL is state representation
  - Current sensors (observability!)
  - Past history?

- A stochastic process has the ***Markov*** property if the conditional probability distribution of future states of the process depends only upon the present state
  - Given the present, the future does not depend on the past
  - Memoryless, pathless

# Implications of the Markov Property

Often the process is not strictly Markovian, but we can either (i) approximate it as such and yield good results, or (ii) include a fixed window of history as state

Thus we can approximate

$$P(s_{t+1} = s', r_{t+1} = r | s_t, a_t, s_{t-1}, a_{t-1}, \ldots r_1, s_0, a_1)$$

via

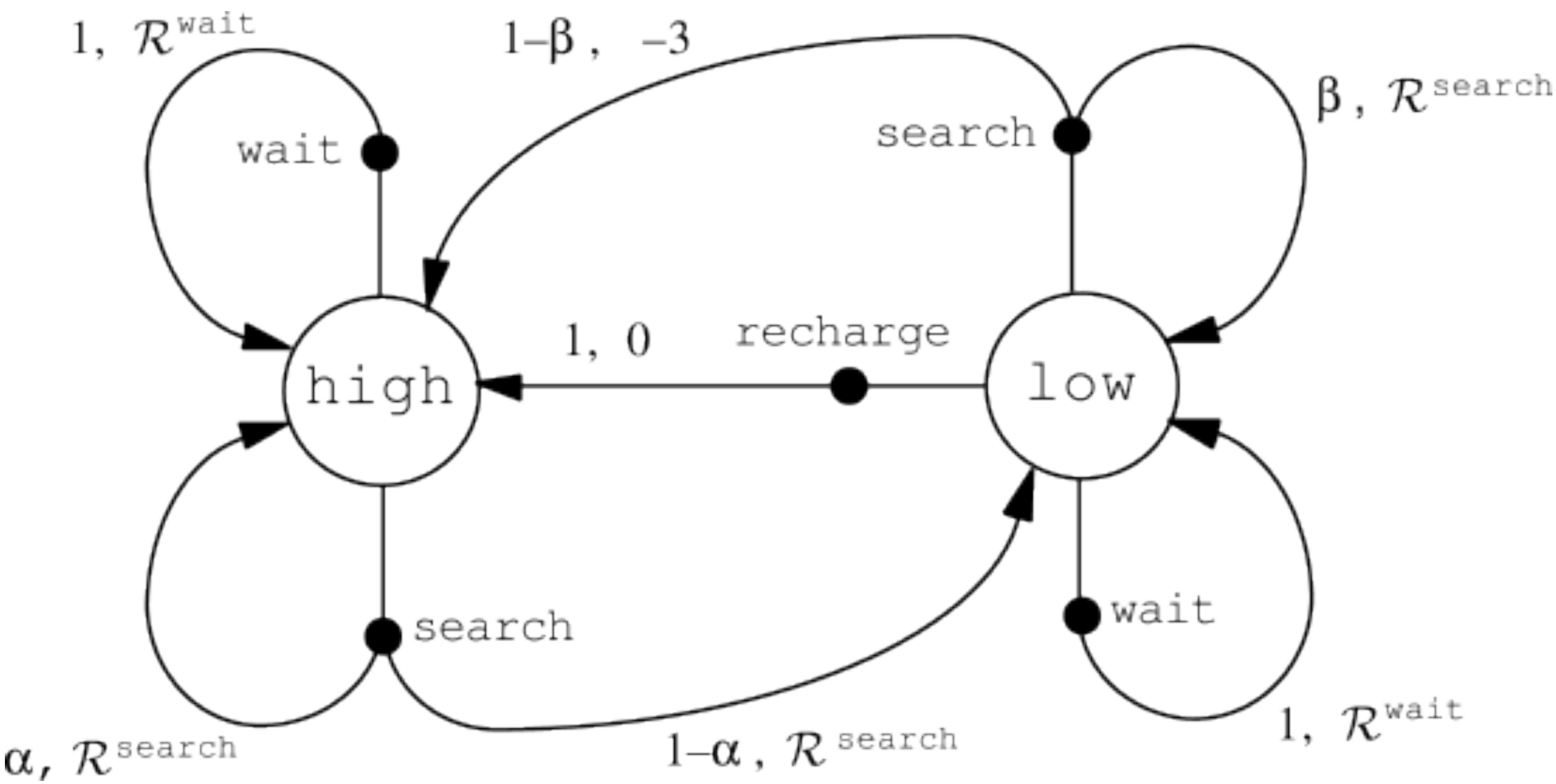
$$P(s_{t+1} = s', r_{t+1} = r | s_t, a_t)$$

# Markov Decision Processes

If a process is Markovian, we can model it as a 5-tuple **MDP**: $(S, A, P(\cdot,\cdot), R(\cdot,\cdot), \gamma)$

- S: set of states
- A: set of actions
- $P_a(s, s')$: transition function
- $R_a(s, s')$: immediate reward

# Recycling Robot MDP



# Value Functions

Almost all RL algorithms are based on estimating *value functions* – functions of states (or of state-action pairs) that estimate how good it is for the agent to be in a given state (or how good it is to perform a given action in a given state)

Value functions are defined with respect to particular policies

# State-Value Function

$$V^{\pi}(s) = E_{\pi}[R_t | s_t = s]$$

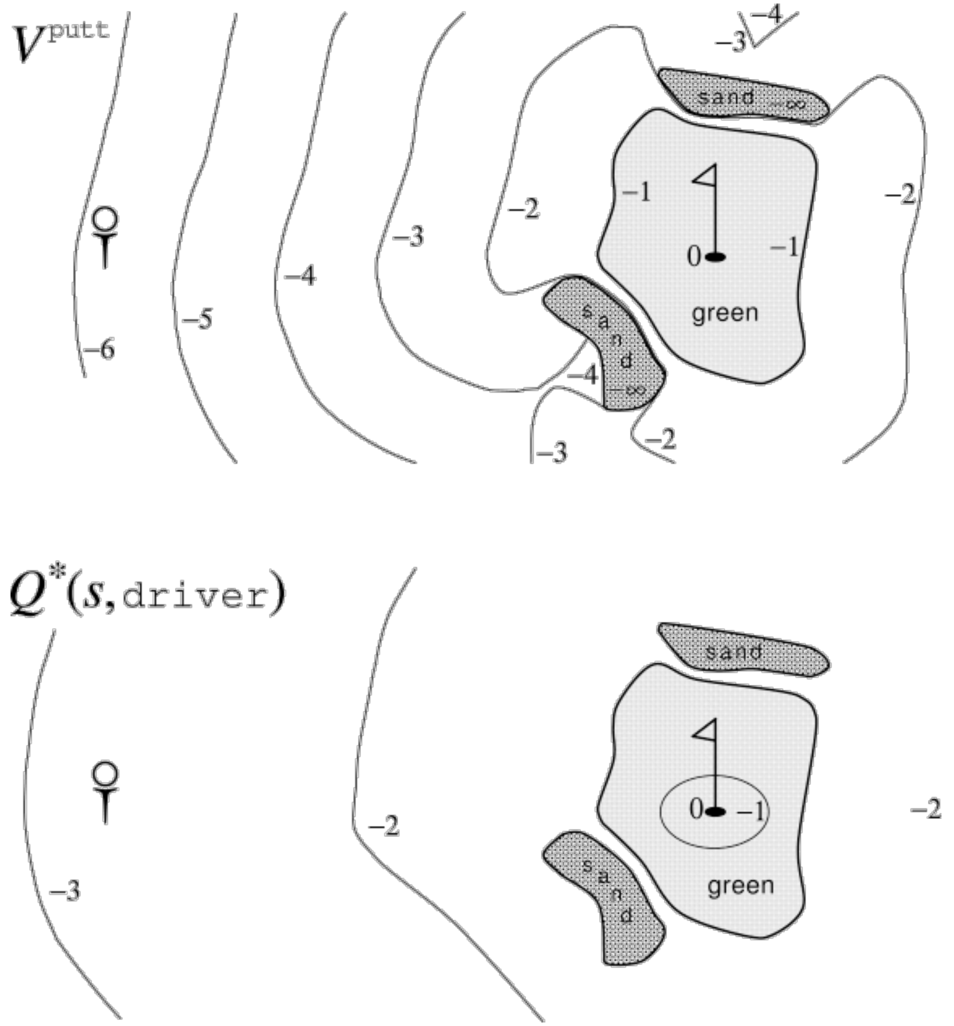
$$= E_{\pi}[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s]$$

# Action-Value Function

$$Q^{\pi}(s,a) = E_{\pi}[R_t | s_t = s, a_t = a]$$

$$= E_{\pi}[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a]$$

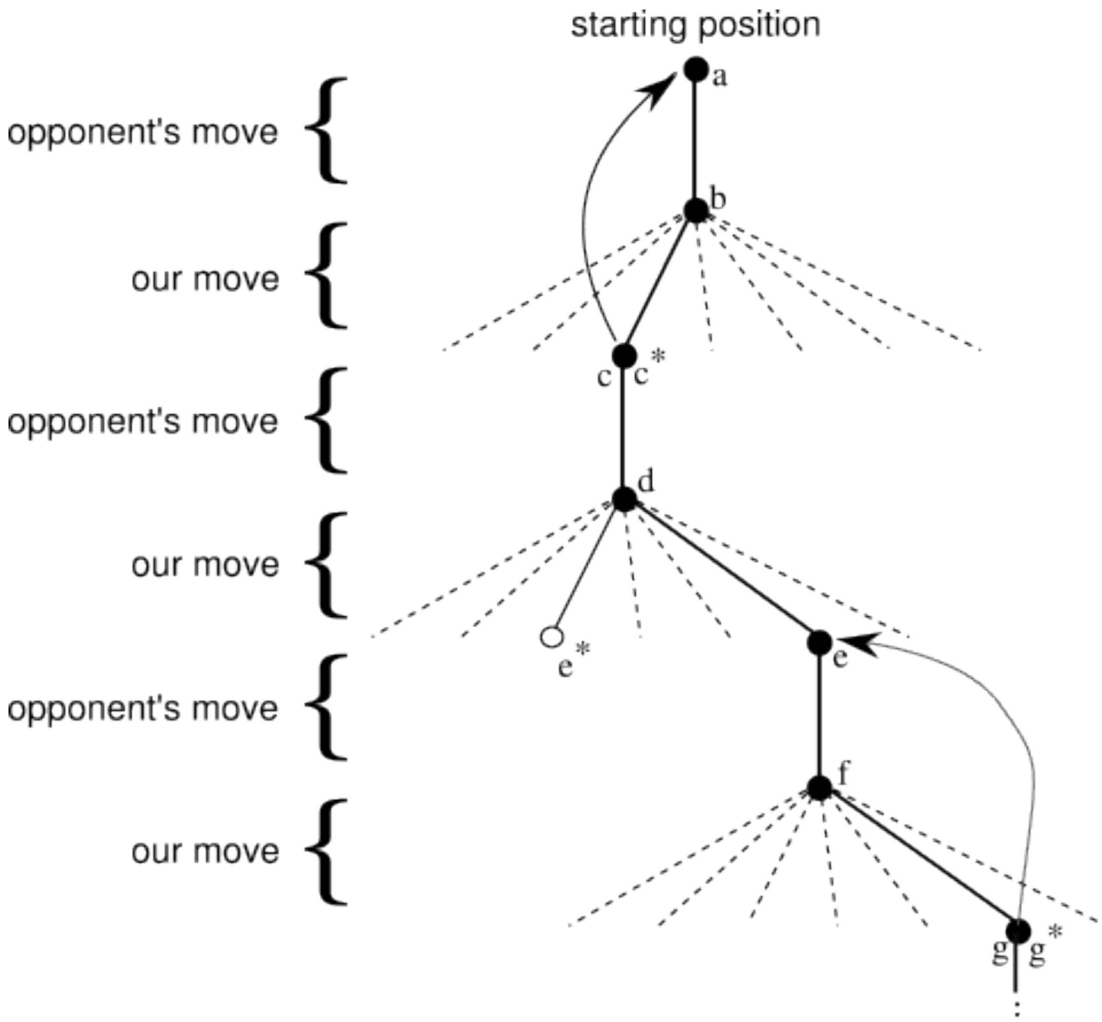
# Example: Golf



# Temporal Difference (TD) Learning

- Combines ideas from Monte Carlo sampling and dynamic programming
- Learns directly from raw experience without a model of environment dynamics
- Update estimates based in part on other learned estimates, without waiting for a final outcome

# Visual TD Learning



# Q-Learning: Off-Policy TD Control

1. Initialize Q(s,a)
   – Random, optimistic, realistic, knowledge
2. Repeat (for each episode):
   a. Initialize s
   b. Repeat (for each step of episode)
      i. Choose action via Q
      ii. Take action, observe r, s'
      iii. $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
      iv. s = s'

   until s is terminal

# Choosing Actions

- Given a Q function, a common approach to selecting action is ε-greedy
  1. Select a random value in [0,1]
     - If > ε, take action with highest estimated value
     - Else, select randomly

- In the limit, every action will be sampled an infinite number of times

# Function Representation

- Given large state-action spaces, there is a practical problem of how to sample the space, and how to represent it

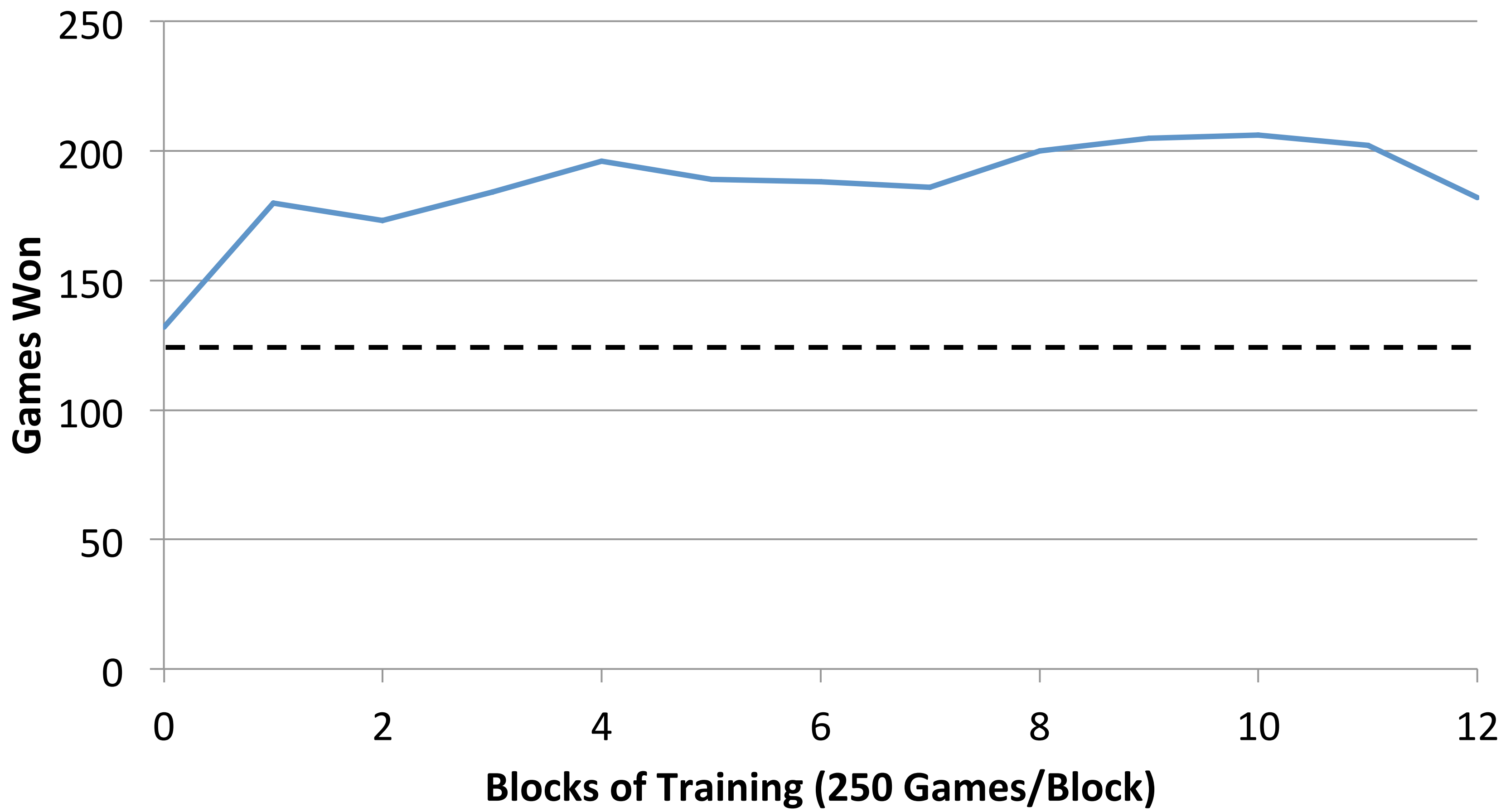
- Modern approaches include hierarchical methods and neural networks

# Application: Michigan Liar's Dice

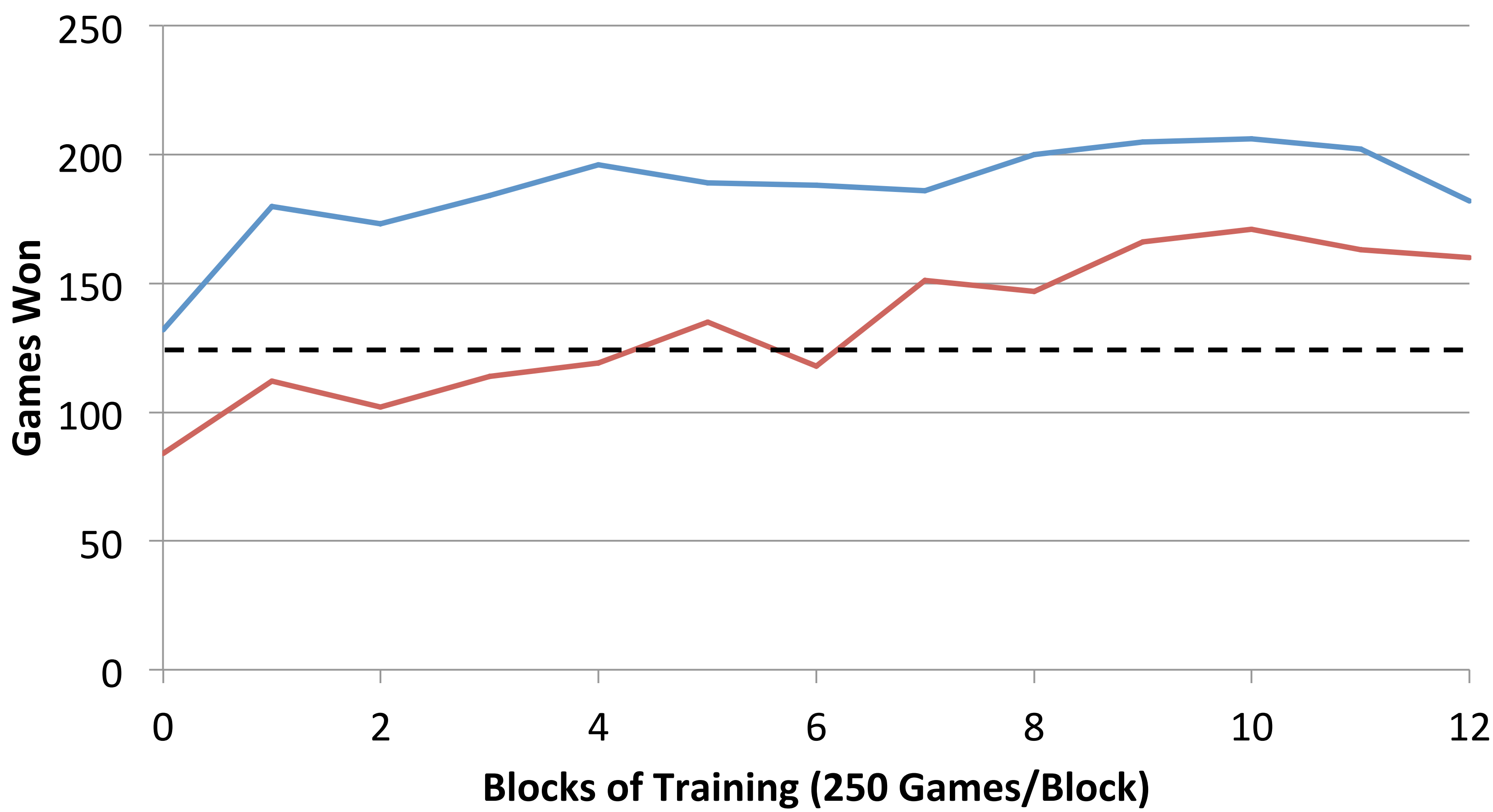
- Multi-agent opponents
- Varying degrees of background knowledge
  - Opponent modeling
  - Probabilistic calculation

Available on the App Store

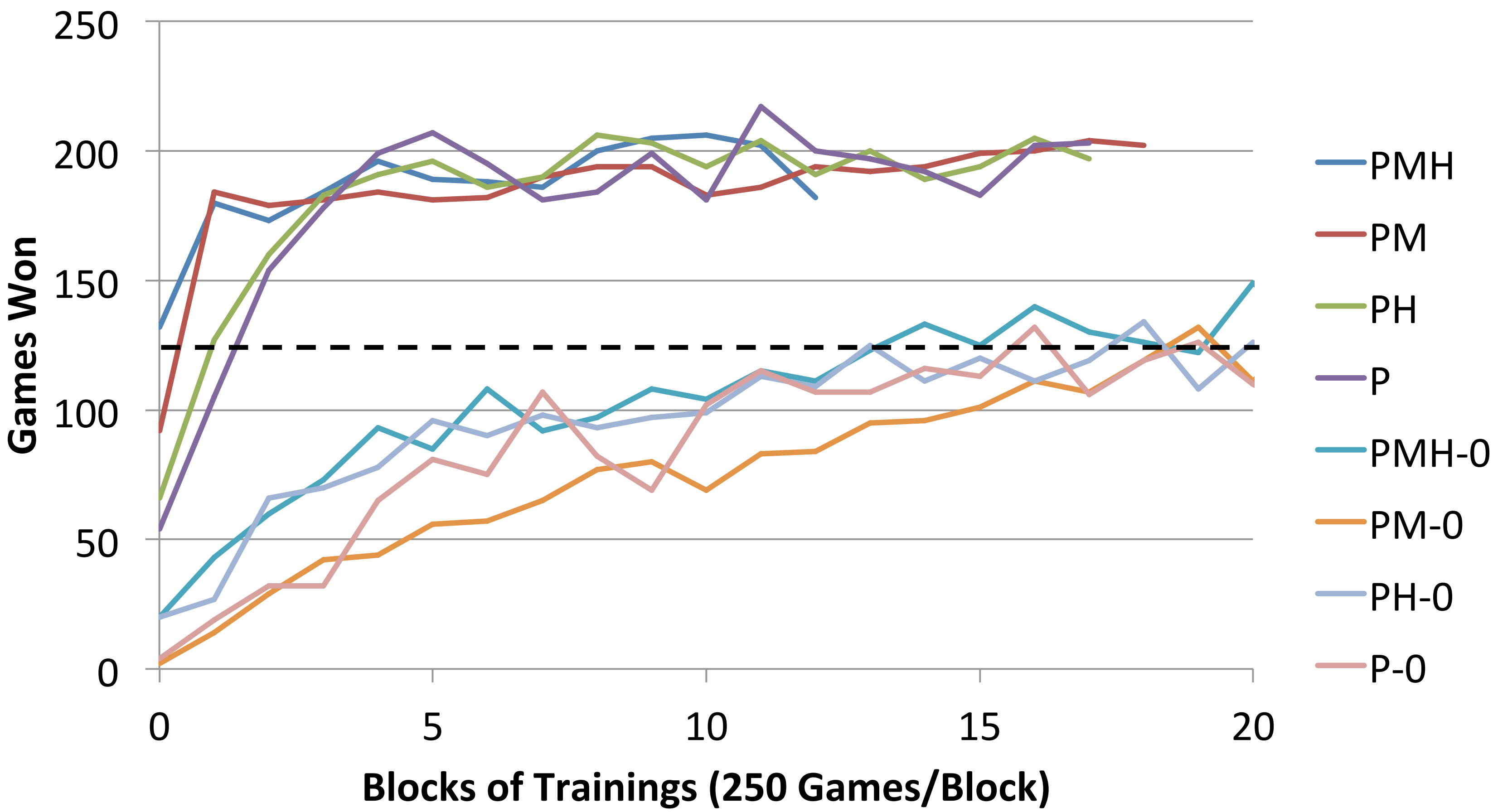
# Evaluation: Learning vs. Static



# Evaluation: Learning vs. Learned



# Evaluation: Value-Function Initialization



# Summary

- Reinforcement Learning (RL) is the problem of learning an effective action policy for obtaining reward

- Most RL algorithms model the task as a Markov Decision Process (MDP) and estimate the value of states/state-actions in a value function

- Temporal-Difference (TD) Learning is one effective method that is online and model-free