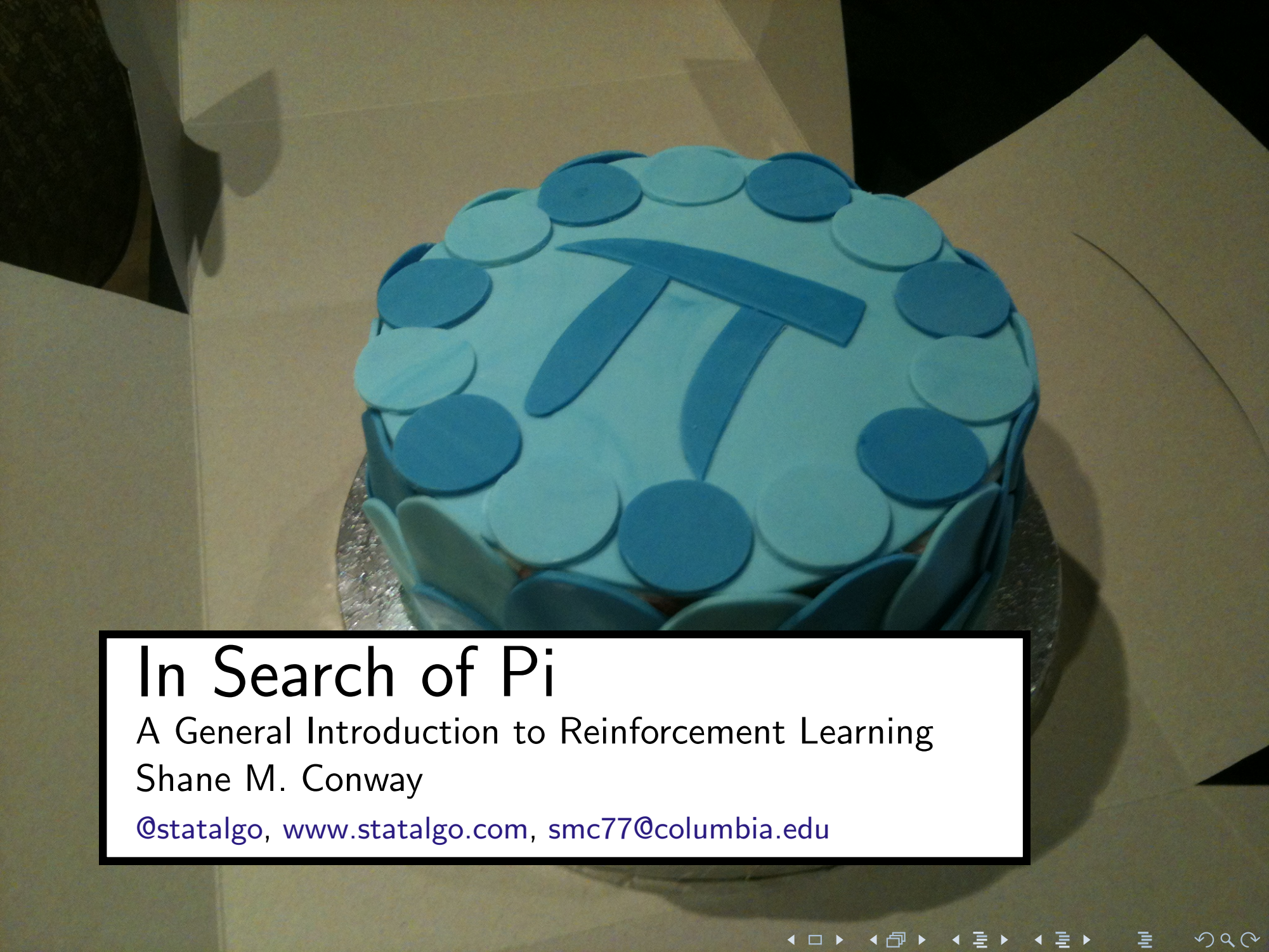


In Search of Pi

A General Introduction to Reinforcement Learning

Shane M. Conway

@statalgo, www.statalgo.com, smc77@columbia.edu

"It would be in vain for one **intelligent Being**, *to set a Rule to the Actions of another, if he had not in his Power, to* **reward** *the compliance with, and* **punish** *deviation from his Rule, by some Good and Evil, that is not the natural product and* **consequence of the action** *itself." (Locke, "Essay", 2.28.6)*

"The use of punishments and rewards can at best be a part of the teaching process. Roughly speaking, if the teacher has no other means of communicating to the pupil, the amount of information which can reach him does not exceed **the total number of rewards and punishments applied**." *(Turing (1950) "Computing Machinery and Intelligence")*

Table of contents

The recipe (some context)

Looking at other pi's (motivating examples)

Prep the ingredients (the simplest example)

Mixing the ingredients (models)

Baking (methods)

Eat your own pi (code)

I ate the whole pi, but I'm still hungry! (references)

Outline

The recipe (some context)

Looking at other pi's (motivating examples)

Prep the ingredients (the simplest example)

Mixing the ingredients (models)

Baking (methods)

Eat your own pi (code)

I ate the whole pi, but I'm still hungry! (references)

What is Reinforcement Learning?

Some context

Why is reinforcement learning so rare here?

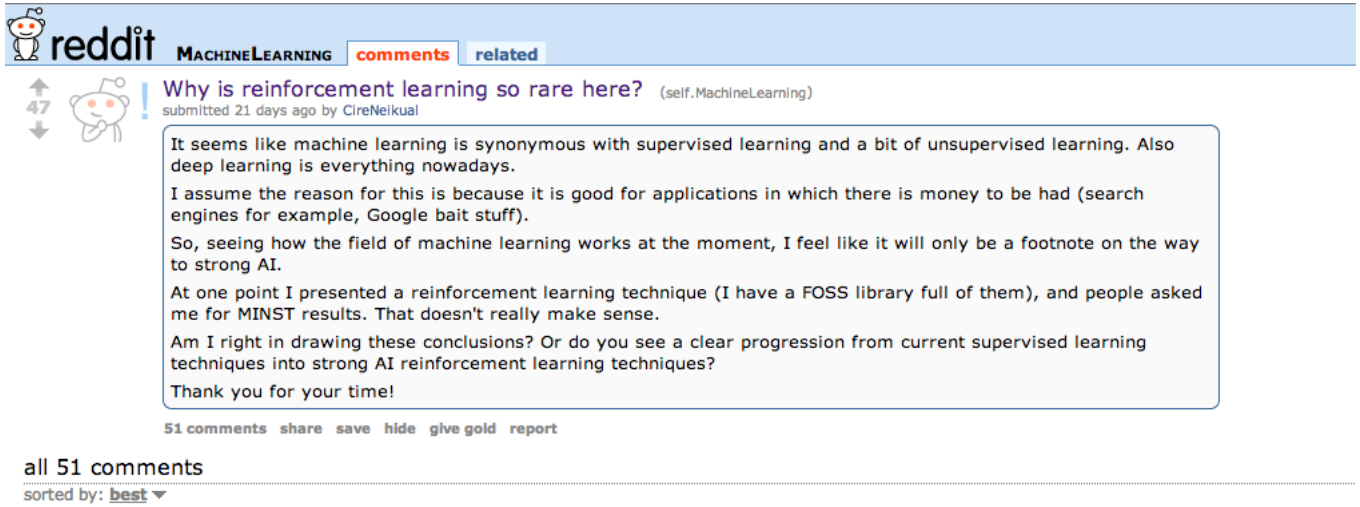


Figure: The machine learning sub-reddit on July 23, 2014.

Why is reinforcement learning so rare here?

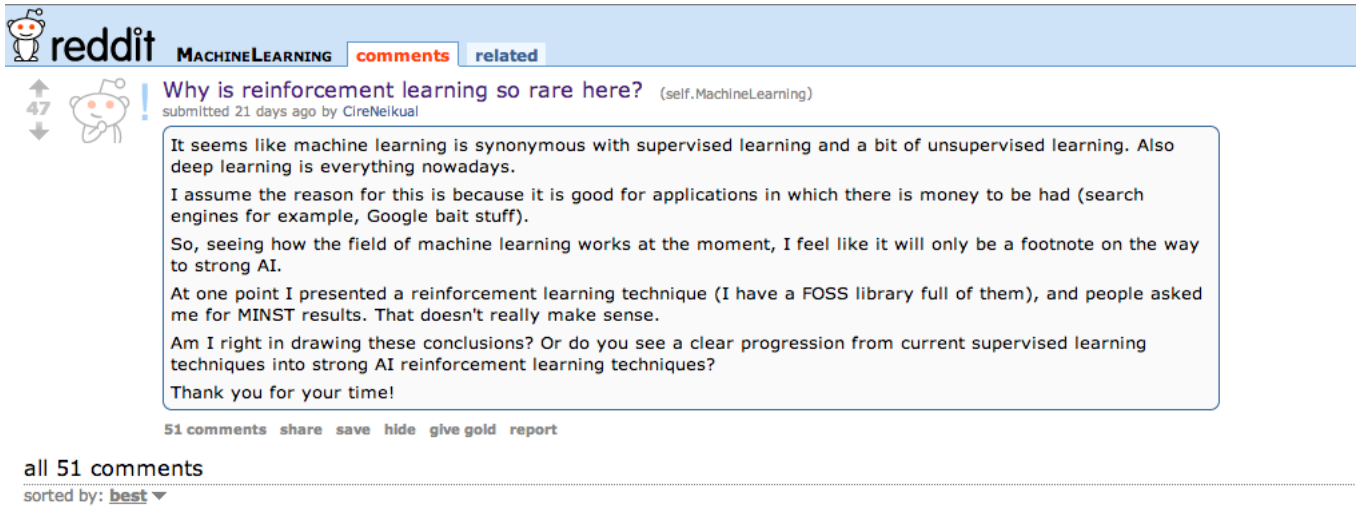


Figure: The machine learning sub-reddit on July 23, 2014.

Reinforcement learning is useful for optimizing the long-run behavior of an agent:

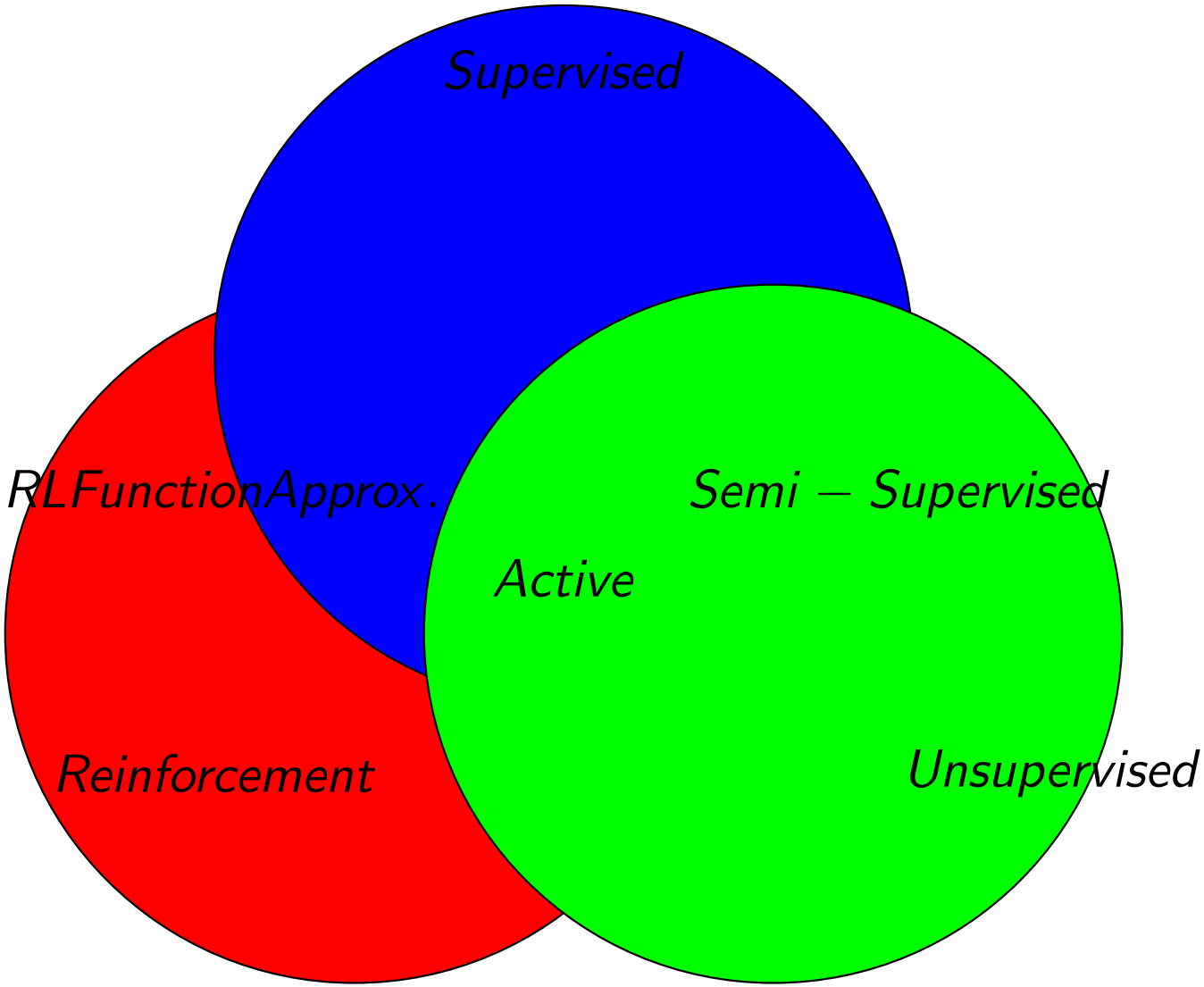
- Handles more complex environments than supervised learning
- Provides a powerful framework for modeling streaming data

Machine Learning

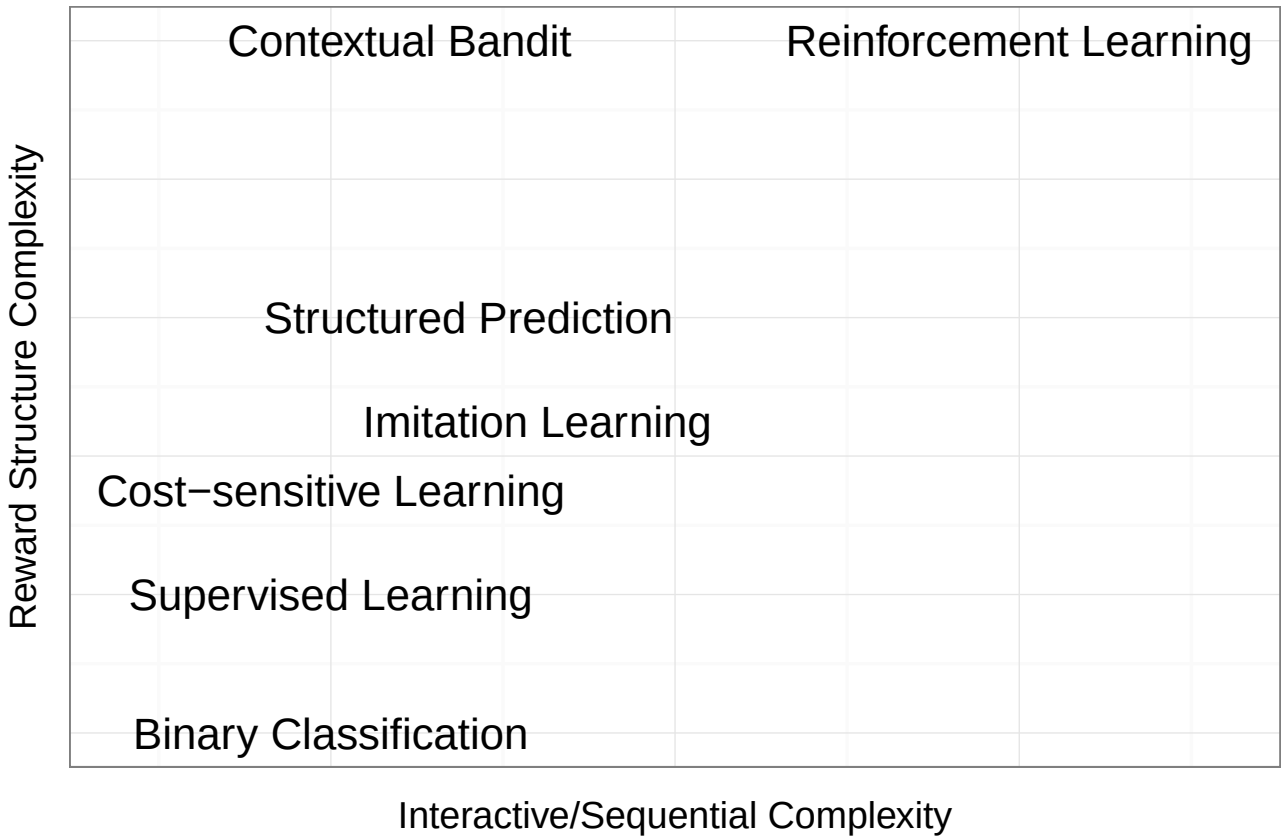
Machine Learning is often introduced as distinct three approaches:

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

Machine Learning (Relationships)



Machine Learning (Complexity and Reductions)



(Langford/Zadrozny 2005)

Reinforcement Learning

...the idea of a learning system that wants something. This was the idea of a **"hedonistic"** *learning system, or, as we would say now, the idea of reinforcement learning.*

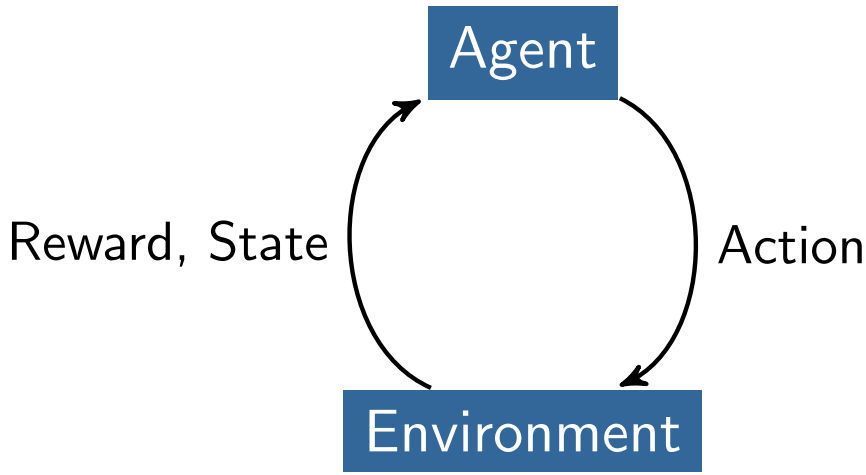
- Barto/Sutton (1998), p.viii

Definition

- Agents take actions in an environment and receive rewards
- Goal is to find the policy π that maximizes rewards
- Inspired by research into psychology and animal learning

RL Model

In a single agent version, we consider two major components: the *agent* and the *environment*.



The agent takes actions, and receives updates in the form of state/reward pairs.

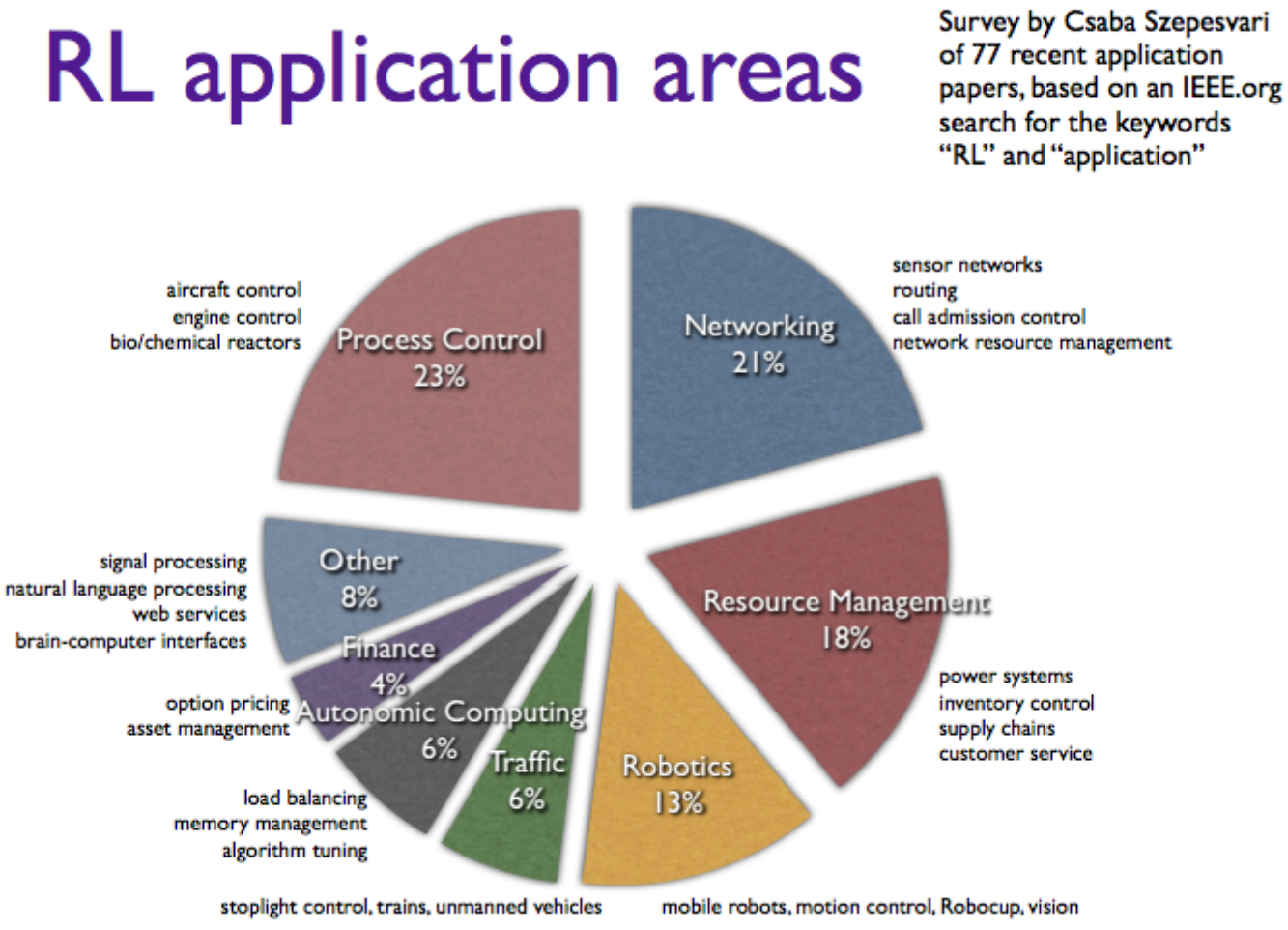
Reinforcement Learning (Fields)

Reinforcement learning gets covered in a number of different fields:

- Artificial intelligence/machine learning
- Control theory/optimal control
- Neuroscience
- Psychology

One primary research area is in *robotics*, although the same methods are applied under *optimal control theory* (often under the subject of *Approximate Dynamic Programming*, or *Sequential Decision Making Under Uncertainty*.)

Reinforcement Learning (Fields)



From "Deconstructing Reinforcement Learning" ICML 2009

Artificial Intelligence

Major goal of Artificial Intelligence: build intelligent agents.

> *"An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators".*
>
> *Russell and Norvig (2003)*

1. Belief Networks (Chp. 14)
2. Dynamic Belief Networks (Chp. 15)
3. Single Decisions (Chp. 16)
4. Sequential Decisions (Chp. 17) (includes MDP, POMDP, and Game Theory)
5. Reinforcement Learning (Chp. 21)

Major Considerations

- Generalization (Learning)
- Sequential Decisions (Planning)
- Exploration vs. Exploitation (Multi-Armed Bandits)
- Convergence (PAC learnability)

Variations

- Type of uncertainty.
- Full vs. partial state observability.
- Single vs. multiple decision-makers.
- Model-based vs. model-free methods.
- Finite vs. infinite state space.
- Discrete vs. continuous time.
- Finite vs. infinite horizon.

Key Ideas

1. Time/life/interaction
2. Reward/value/verification
3. Sampling
4. Bootstrapping

Richard Sutton's list of key ideas for reinforcement learning ("Deconstructing Reinforcement Learning" ICML 2009)

Outline

The recipe (some context)

Looking at other pi's (motivating examples)

Prep the ingredients (the simplest example)

Mixing the ingredients (models)

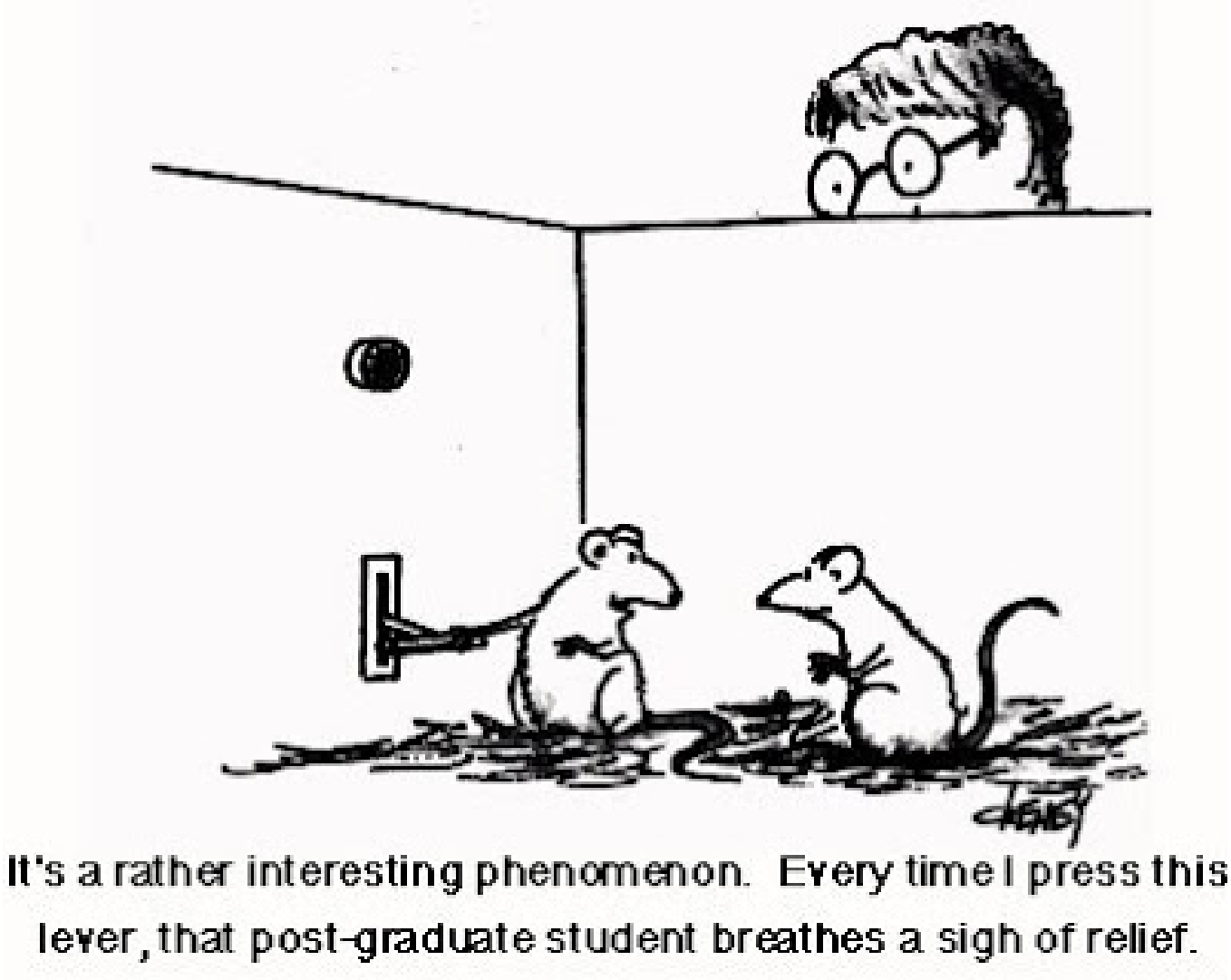
Baking (methods)

Eat your own pi (code)

I ate the whole pi, but I'm still hungry! (references)

How is Reinforcement Learning being used?

Behaviorism



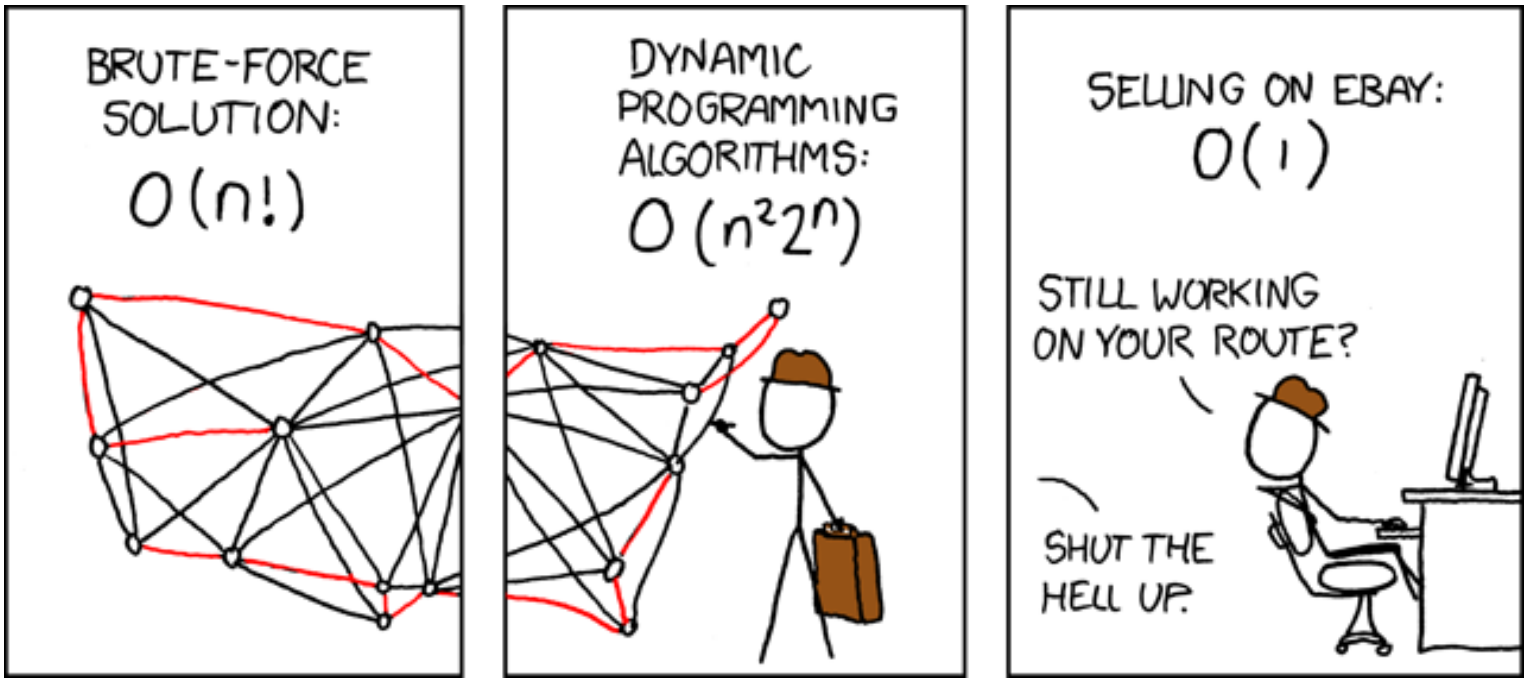
It's a rather interesting phenomenon. Every time I press this lever, that post-graduate student breathes a sigh of relief.

Human Trials

Figure: "How Pavlok Works: Earn Rewards when you Succeed. Face Penalties if you Fail. Choose your level of commitment. Pavlok can reward you when you achieve your goals. Earn prizes and even money when you complete your daily task. But be warned: if you fail, you'll face penalties. Pay a fine, lose access to your phone, or even suffer an electric shock...at the hands of your friends."

Shortest Path, Travelling Salesman Problem

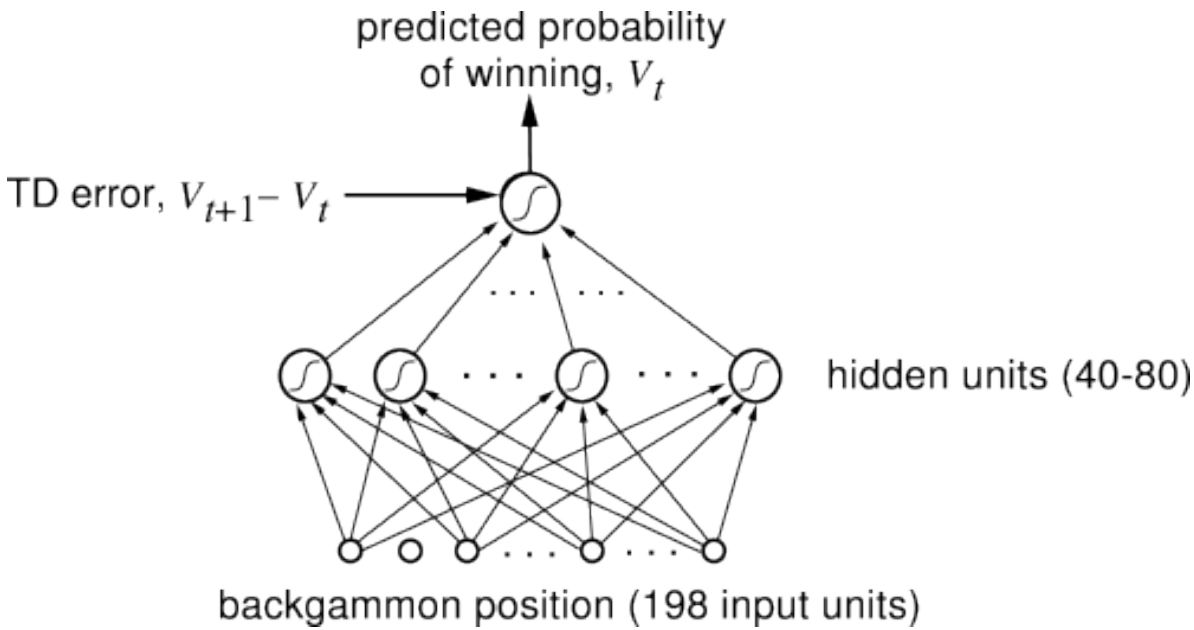
Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?



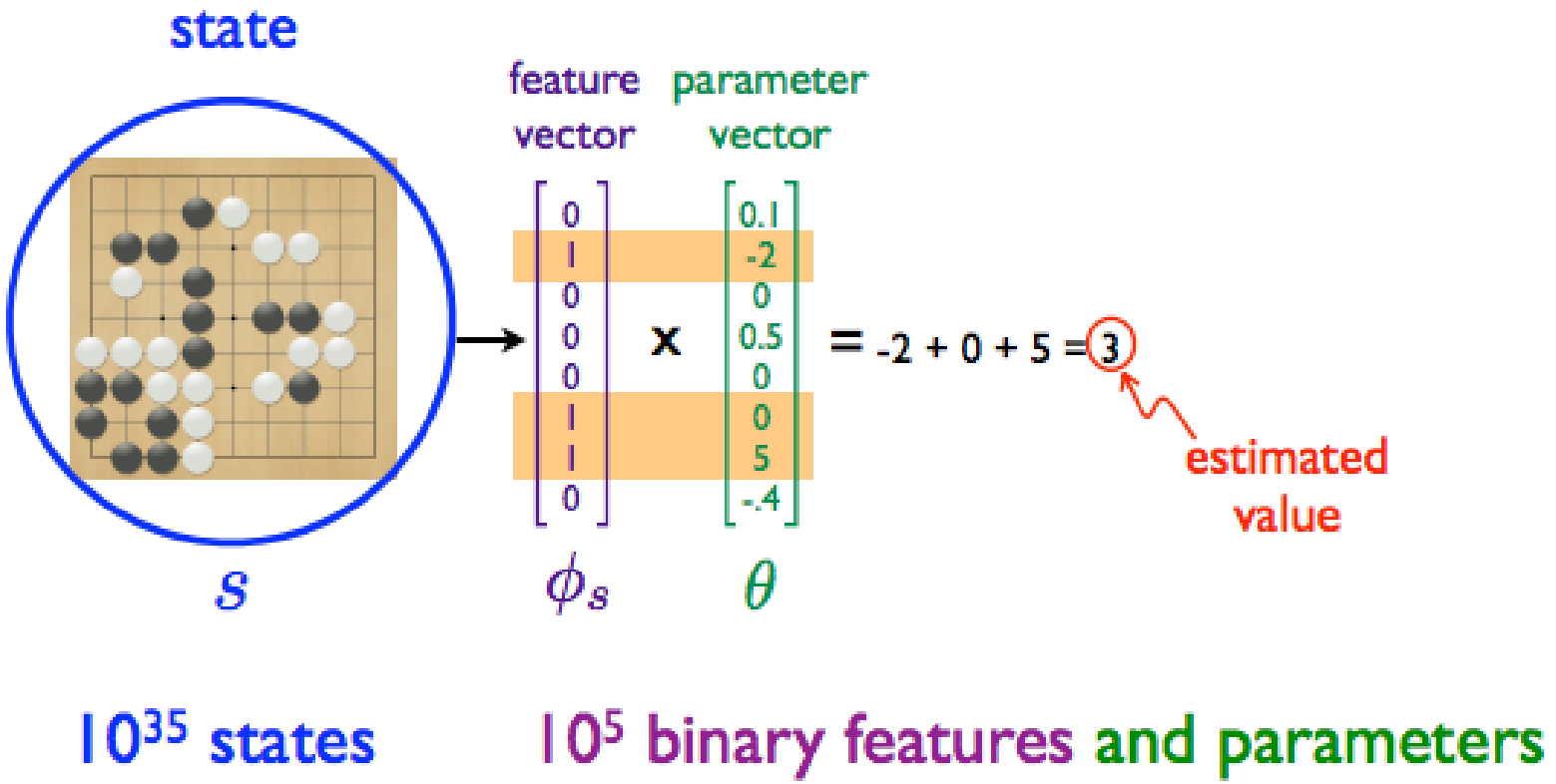
- Bellman, R. (1962), "Dynamic Programming Treatment of the Travelling Salesman Problem"
- Example in python from Mariano Chouza

TD-Gammon

Tesauro (1995) "Temporal Difference Learning and TD-Gammon" may be the most famous success story for RL, using a combination of the TD(λ) algorithm and nonlinear function approximation using a multilayer neural network trained by backpropagating TD errors.

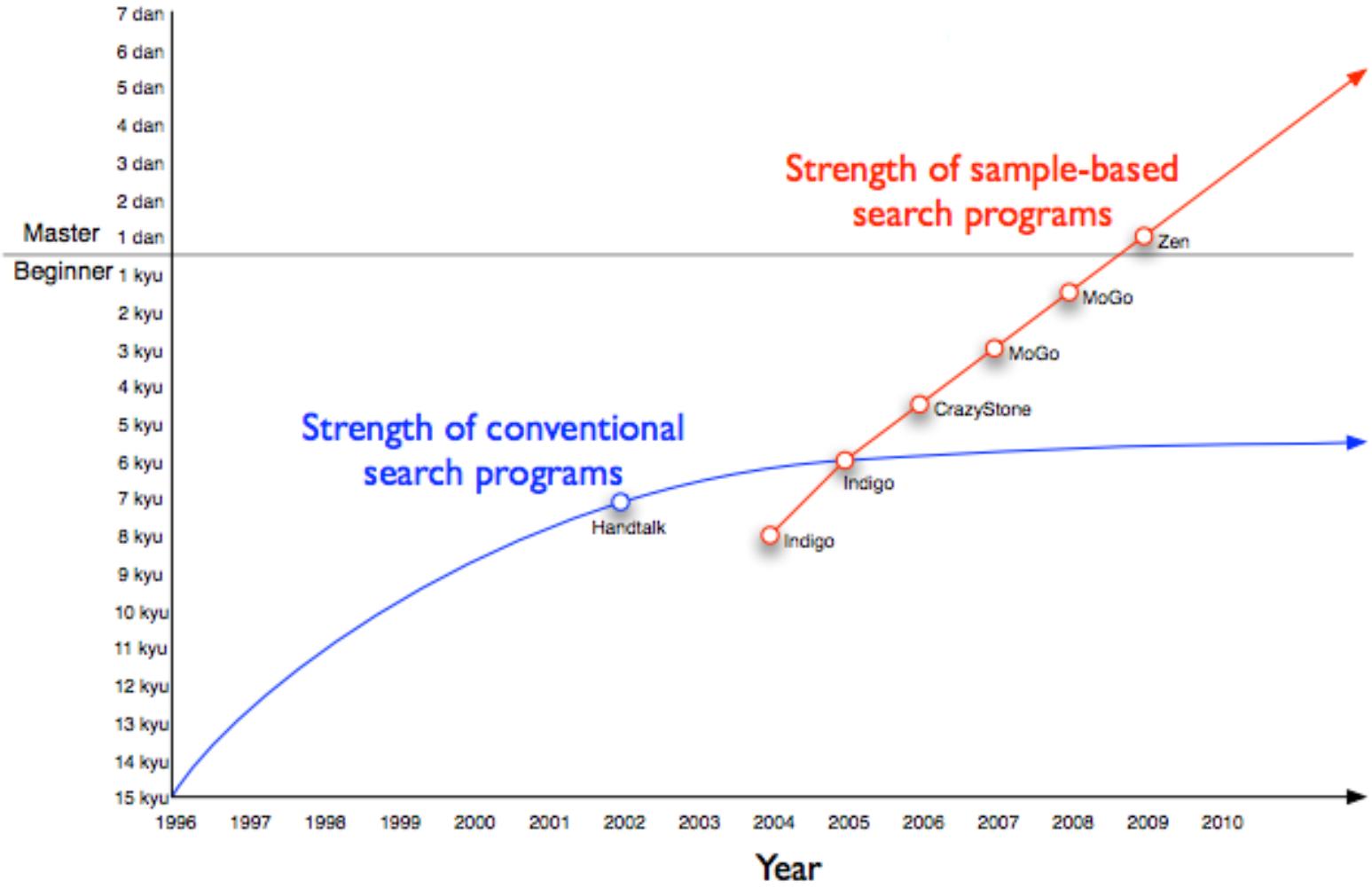


Go



From Sutton (2009) "Deconstructing Reinforcement Learning" ICML

Go



From Sutton (2009) "Fast Gradient-Descent Methods for Temporal-Difference Learning with Linear Function Approximation" ICML

Andrew Ng's Helicopters

https://www.youtube.com/watch?v=Idn10JBsA3Q

Outline

The recipe (some context)

Looking at other pi's (motivating examples)

Prep the ingredients (the simplest example)

Mixing the ingredients (models)

Baking (methods)

Eat your own pi (code)

I ate the whole pi, but I'm still hungry! (references)

Multi-Armed Bandits

Single-state reinforcement learning problems.

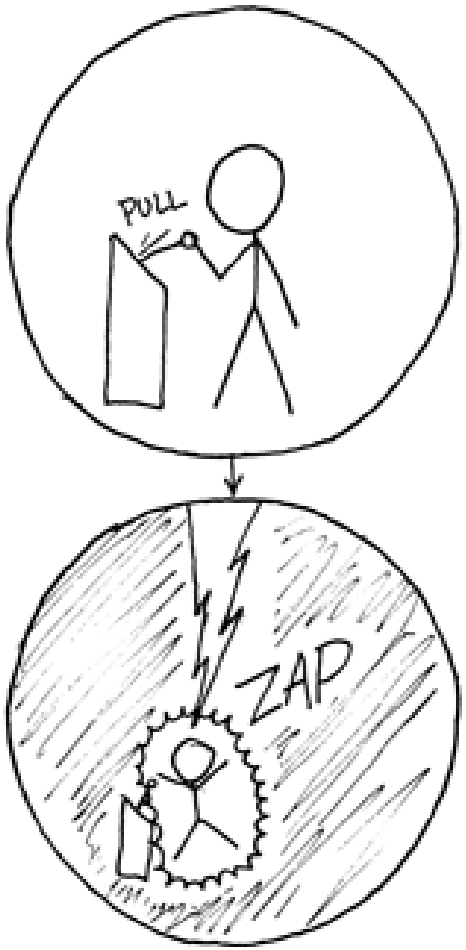
Multi-Armed Bandits

A simple introduction to the reinforcement learning problem is the case when there is only one state, also called a *multi-armed bandit*. This was named after the slot machines (one-armed bandits).

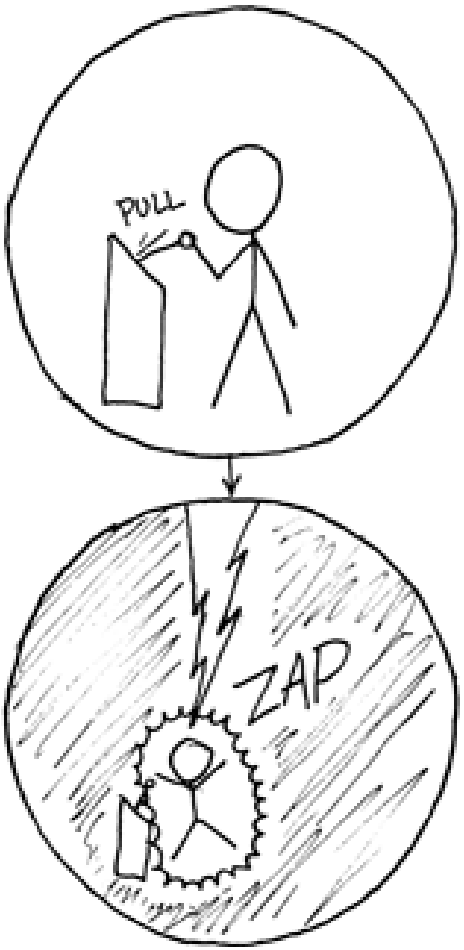
Definition

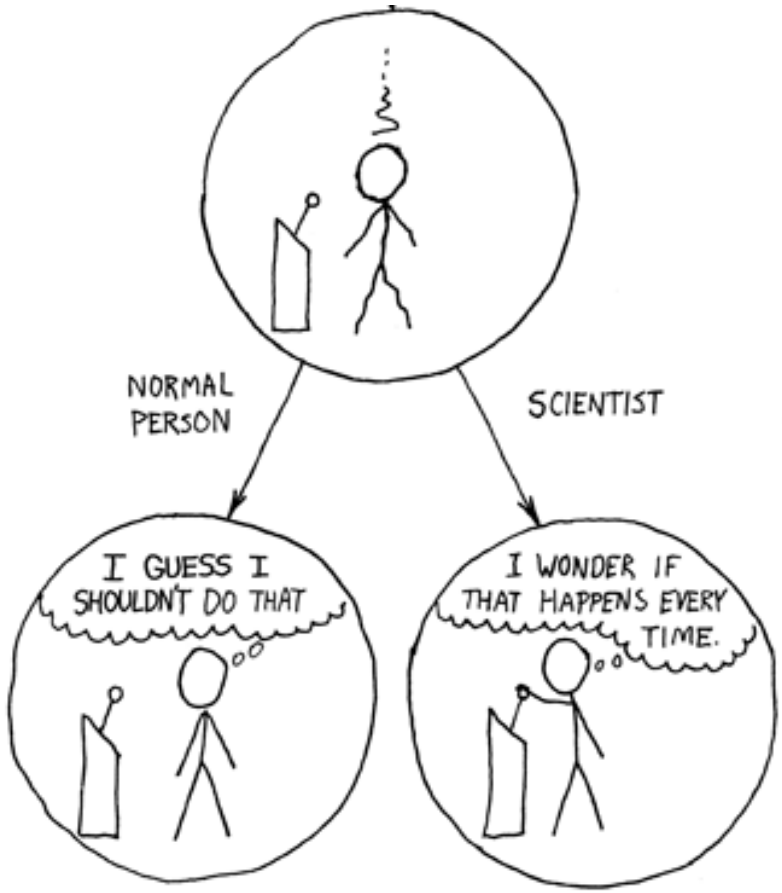
- Set of actions $A = 1, ..., n$
- Each action gives you a random reward with distribution $P(r_t|a_t = i)$
- The value (or utility) is $V = \sum_t r_t$

Exploration vs. Exploitation



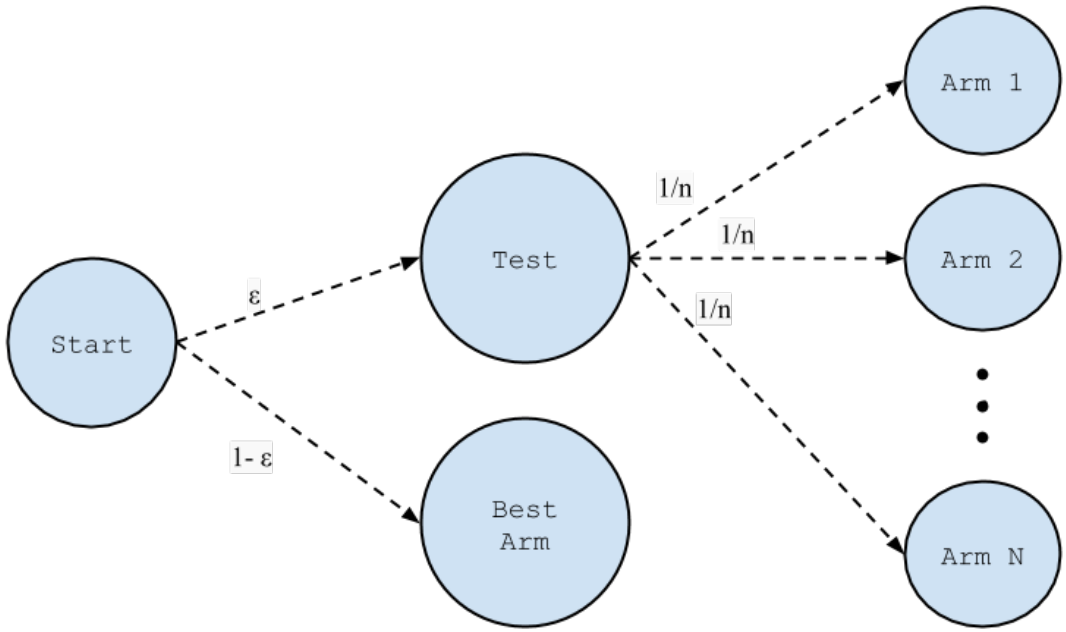
Exploration vs. Exploitation





ϵ-Greedy

The ϵ-Greedy algorithm is one of the simplest and yet most popular approaches to solving the exploration/exploitation dilemma.



(picture courtesy of "Python Multi-armed Bandits" by Eric Chiang, yhat)

Outline

The recipe (some context)

Looking at other pi's (motivating examples)

Prep the ingredients (the simplest example)

Mixing the ingredients (models)

Baking (methods)

Eat your own pi (code)

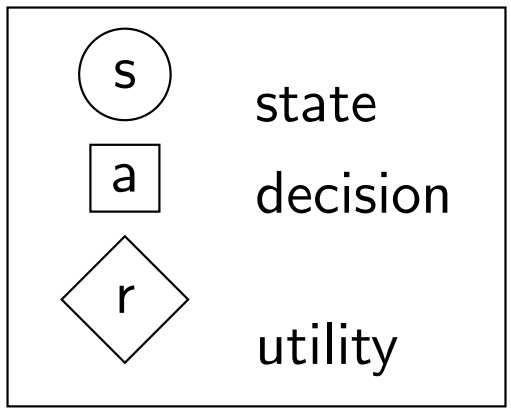
I ate the whole pi, but I'm still hungry! (references)

Reinforcement Learning Models

Especially Markov Decision Processes.

Dynamic Decision Networks

Bayesian networks are a popular method for characterizing probabilistic models. These can be extended as a Dynamic Decision Network (DDN) with the addition of decision (action) and utility (value) nodes.



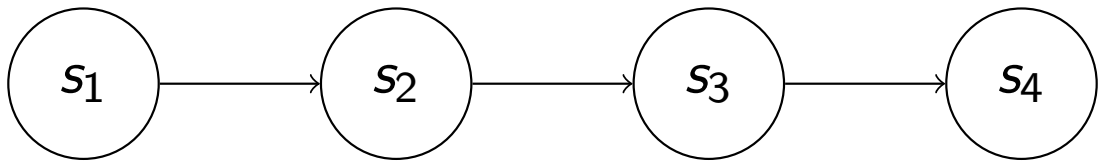
Markov Models

We can extend the markov process to study other models with the same the property.

Markov Models	Are States Observable?	Control Over Transitions?
Markov Chains	Yes	No
MDP	Yes	Yes
HMM	No	No
POMDP	No	Yes

Markov Processes

Markov Processes are very elementary in time series analysis.



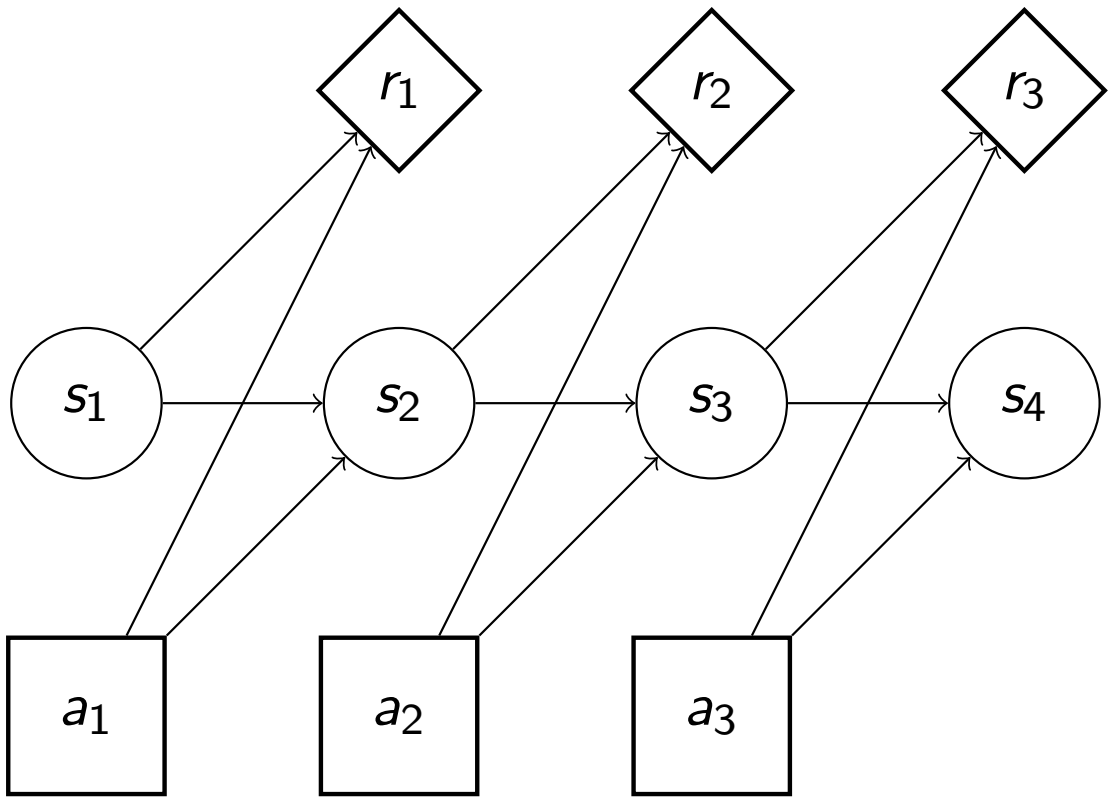
Definition

$$P(s_{t+1}|s_t, ..., s_1) = P(s_{t+1}|s_t) \tag{1}$$

- s_t is the state of the markov process at time t.

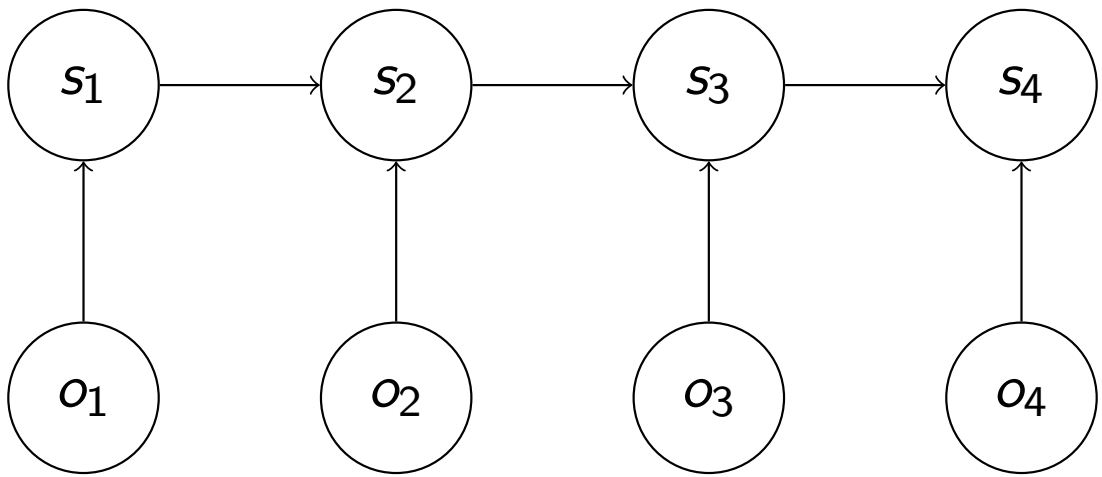
Markov Decision Process (MDP)

A Markov Decision Process (MDP) adds some further structure to the problem.



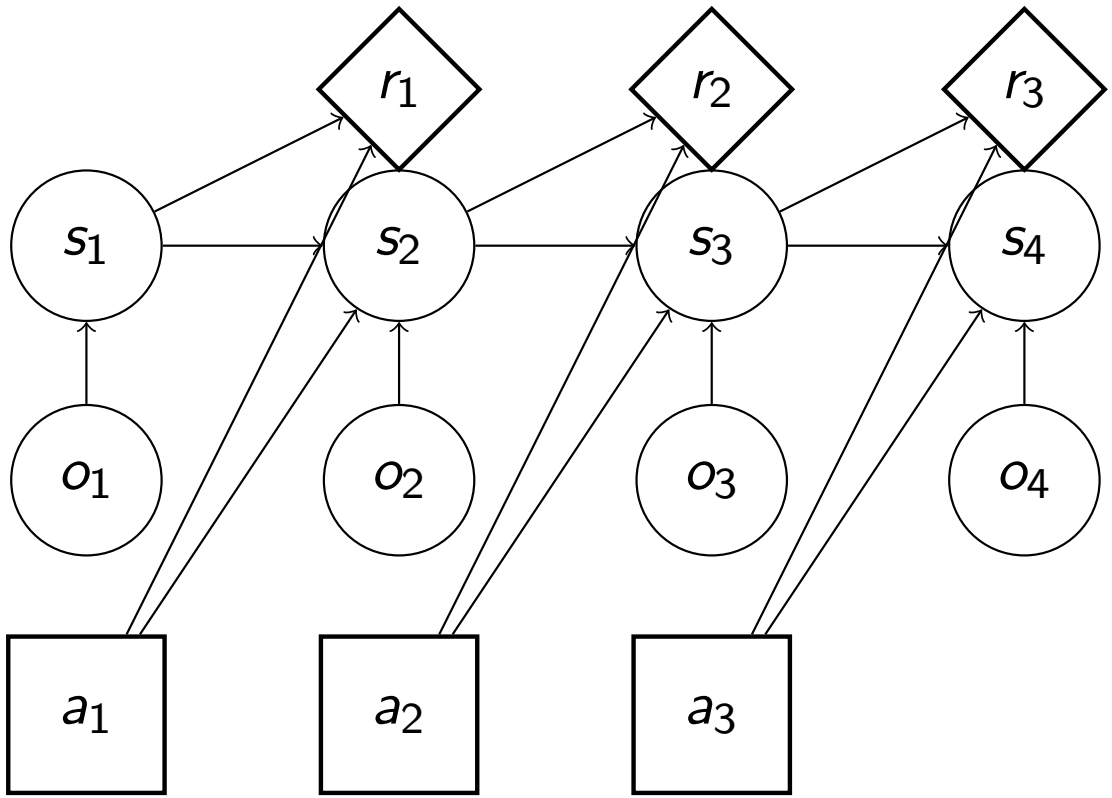
Hidden Markov Model (HMM)

Hidden Markov Models (HMM) provide a mechanism for modeling a hidden (i.e. unobserved) stochastic process by observing a related observed process. HMM have grown increasingly popular following their success in NLP.



Partially Observable Markov Decision Processes (POMDP)

A Partially Observable Markov Decision Processes (POMDP) extends the MDP by assuming partial observability of the states, where the current state is a probability model (a *belief state*).



RL Model

An MDP tranisitons from state s to state s' following an action a, and receiving a reward r as a result of each transition:

$$s_0 \xrightarrow[r_0]{a_0} s_1 \xrightarrow[r_1]{a_1} s_2 \ldots \tag{2}$$

MDP Components

- S is a set of states
- A is set of actions
- $R(s)$ is a reward function

In addition we define:

- $T(s'|s, a)$ is a probability transition function
- γ as a discount factor (from 0 to 1)

Policy

The objective is to find a policy π that maps actions to states, and will maximize the rewards over time:

$$\pi(s) \rightarrow a$$

RL Model

We define a *value function* to maximize the expected return:

$$V^{\pi}(s) = E[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots | s_0 = s, \pi]$$

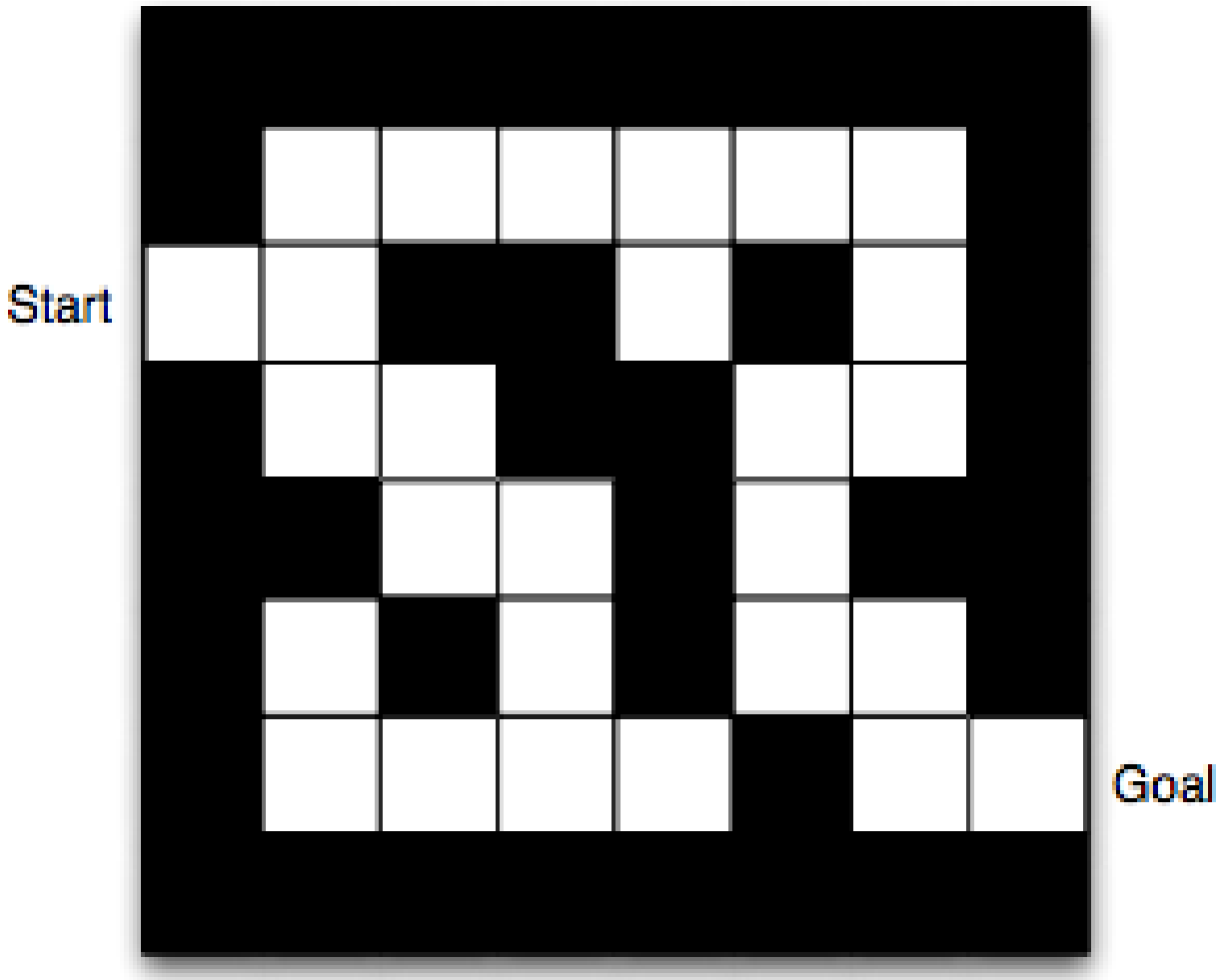
We can rewrite this as a recurrence relation, which is known as the **Bellman Equation**:

$$V^{\pi}(s) = R(s) + \gamma \sum_{s' \in S} T(s') V^{\pi}(s')$$

$$Q^{\pi}(s, a) = R(s) + \gamma \sum_{s' \in S} T(s') max_a Q^{\pi}(s', a')$$

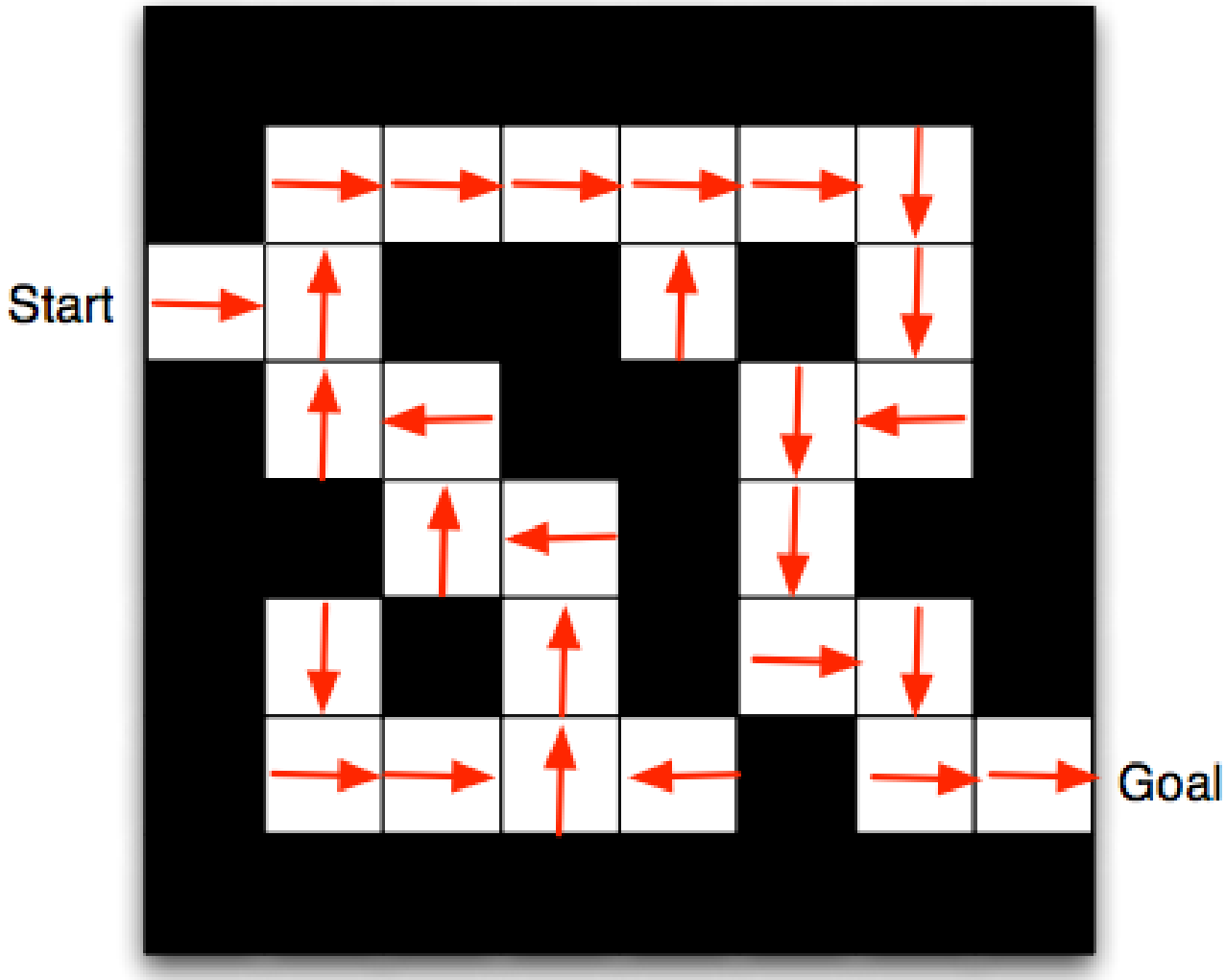
Grid World

http://www0.cs.ucl.ac.uk/staff/D.Silver/web/Teaching$_f$*iles*/*intro*$_R$*L.pdf*
Grid world is a canonical example used in reinforcement learning.



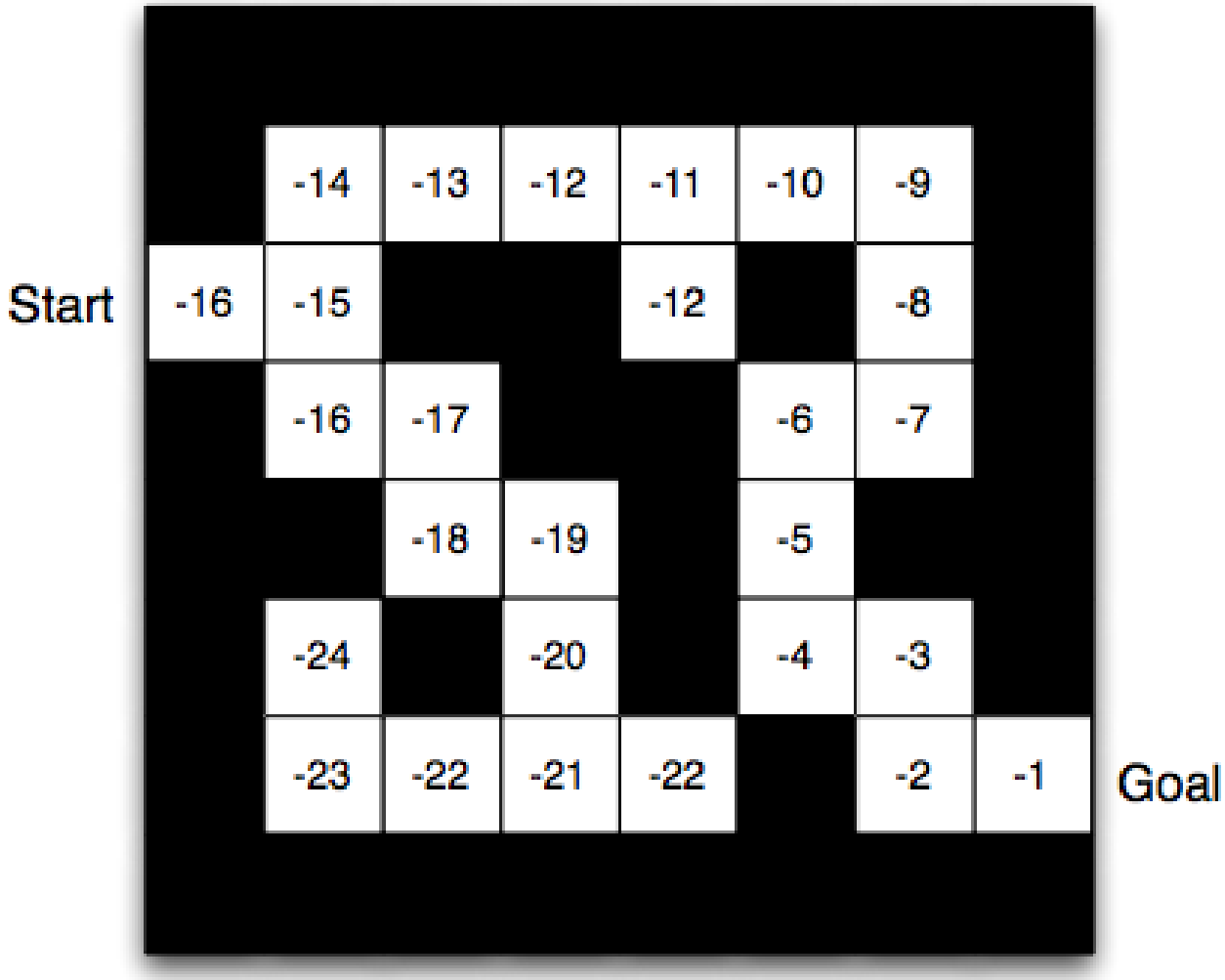
Grid World

Grid world is a canonical example used in reinforcement learning.



Grid World

Grid world is a canonical example used in reinforcement learning.



Model-based vs. Model-free

- Model-free: Learn a controller without learning a model.
- Model-based: Learn a model, and use it to derive a controller.

Notation Comment

I am using a fairly standard notation throughout this talk, which focuses on *maximization of a utility*; an alternative version uses *minimization of cost*, where the cost is the negative value of the reward:

Here	Alternative
action a	control u
reward R	cost g
value V	cost-to-go J
policy π	policy μ
discounting factor γ	discounting factor α
transition probability $P_a(s, s')$	transition probability $p_{ss'}(a)$

Outline

The recipe (some context)

Looking at other pi's (motivating examples)

Prep the ingredients (the simplest example)

Mixing the ingredients (models)

Baking (methods)

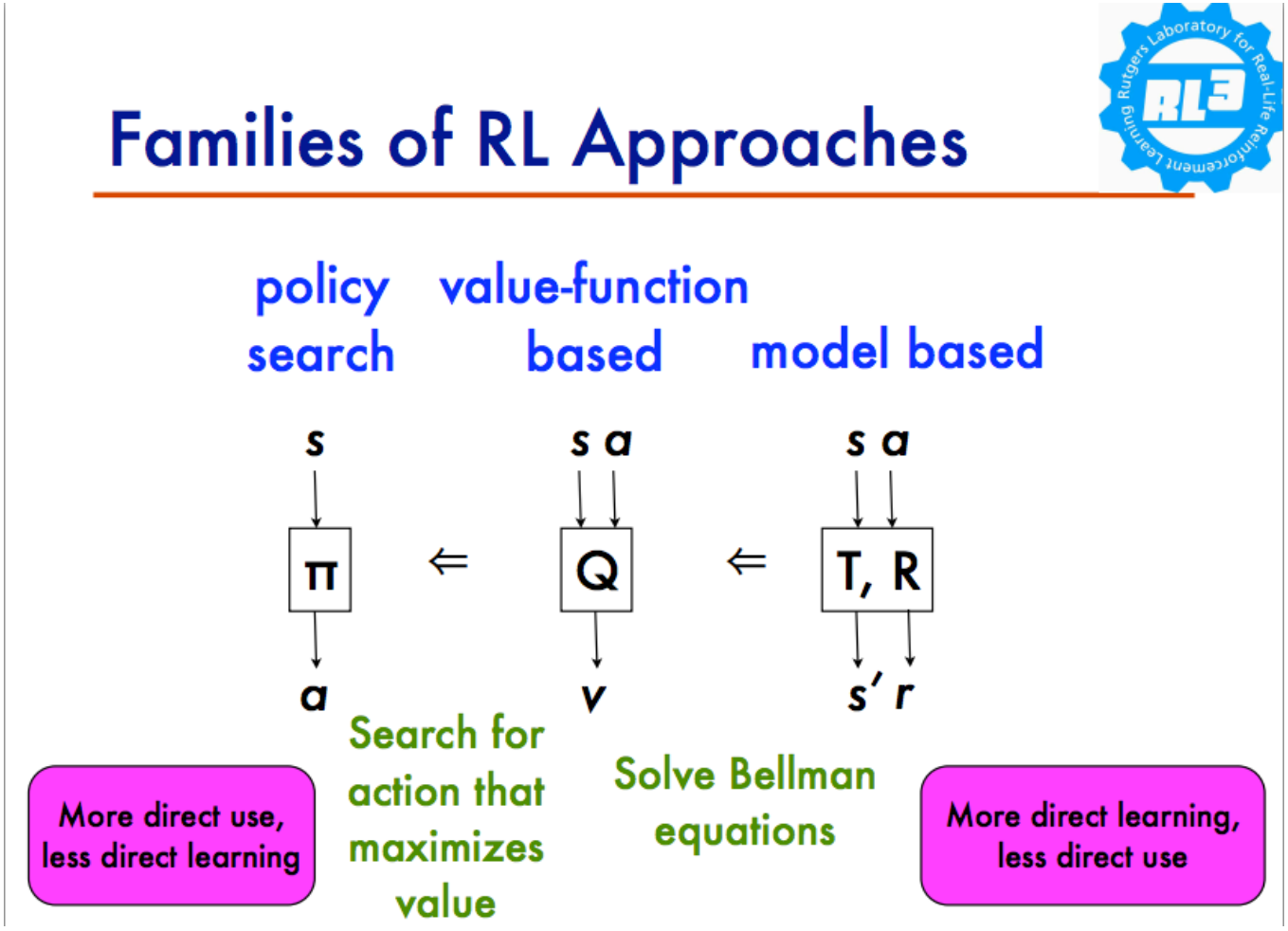
Eat your own pi (code)

I ate the whole pi, but I'm still hungry! (references)

How to Solve an MDP

The basics, from dynamic programming to TD(λ).

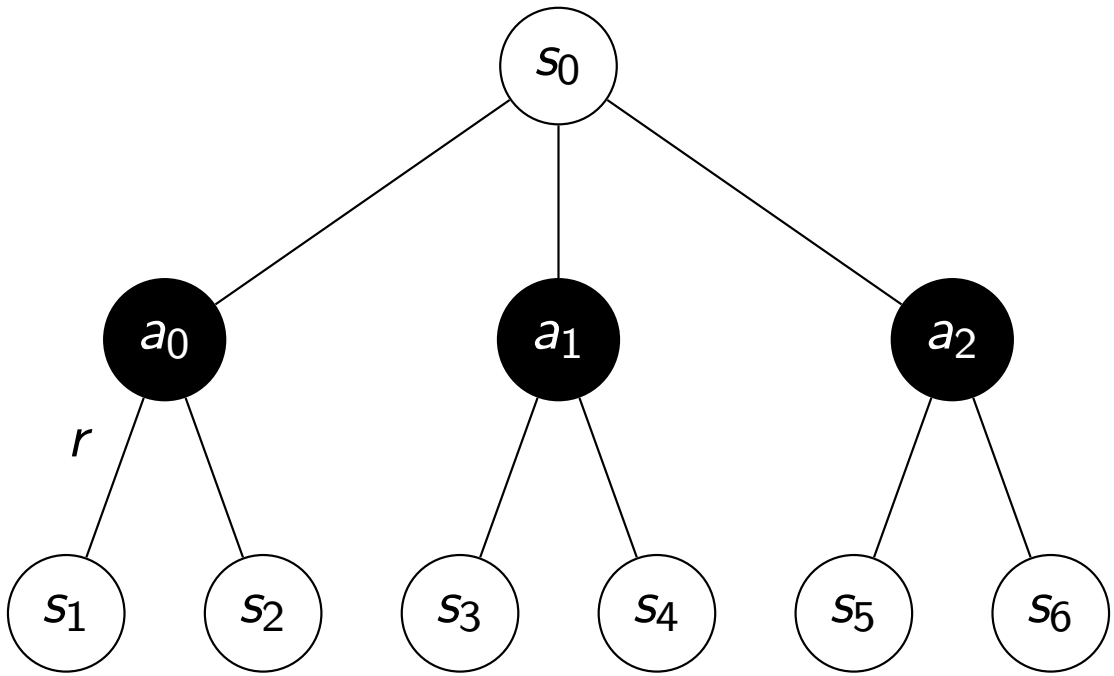
Families of Approaches



The approaches to RL can be summarized based on what they learn (from Littman (2009) talk at NIP)

Backup Diagrams

Backup diagrams provide a mechanism for summarizing how different methods operate by showing how information is backed up to a state.



Dynamic Programming

Dynamic programming is one of the widely known methods for multi-period optimization.

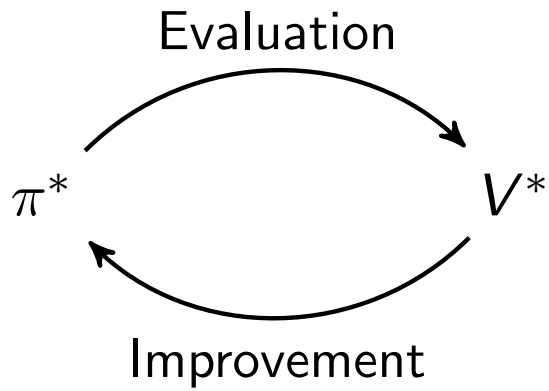
Dynamic Programming Methods

Dynamic programming methods require full knowledge of the environment: T (probability transition function) and R (the reward function).

- Value iteration: Bellman (1957) introduced this method, which finds the value of each state, which can then be used to compute a policy.
- Policy Iteration: Howard (1960) updates the value once, then finds the optimal policy, repeatedly until the policy does not change.

Generalized Policy Iteration

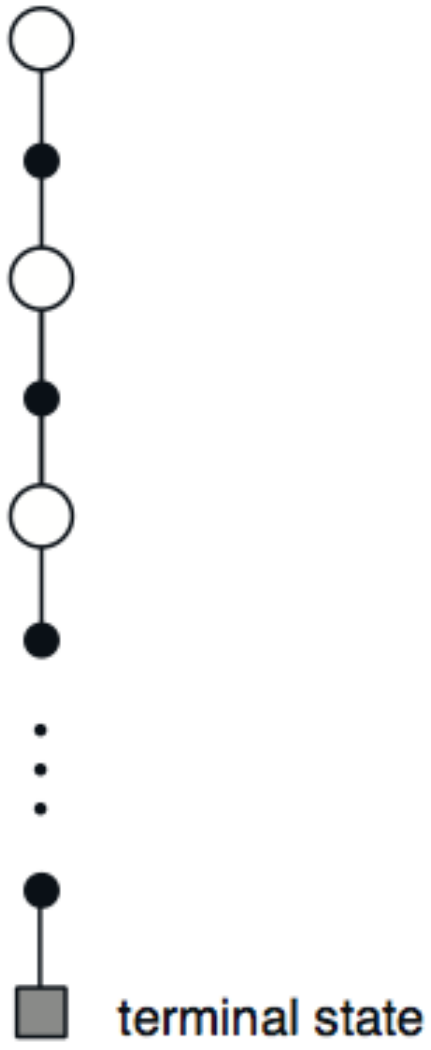
Almost all reinforcement learning methods can be described by the general idea of *generalized policy iteration* (GPI), which breaks the optimization into two processes: policy evaluation and policy improvement.



Monte Carlo

Monte Carlo Methods

Monte carlo methods learn from online, simulated experience, and require no prior knowledge of the environment's dynamics.



Temporal Difference

Temporal Difference (TD) learning was formally introduced in Sutton (1984, 1988). Also used in Samuel (1946).

TD(0) Updates

TD learning computes the temporal difference error, and adds this to the current estimate based on the learning rate α.

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

Q-Learning

Q-learning (Watkins 1989) is a model-free method, and is one of the most important methods in reinforcement learning as it was one of the first to show convergence. Rather than learning the optimal value function, Q-learning learns the Q function.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$$

Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
  Initialize s
  Repeat (for each step of episode):
    Choose a from s using policy derived from Q (e.g., ε-greedy)
    Take action a, observe r, s'
    $Q(s, a) \leftarrow Q(s, a) + \alpha\left[r + \gamma \max_{a'} Q(s', a') - Q(s, a)\right]$
    $s \leftarrow s'$;
  until s is terminal

SARSA

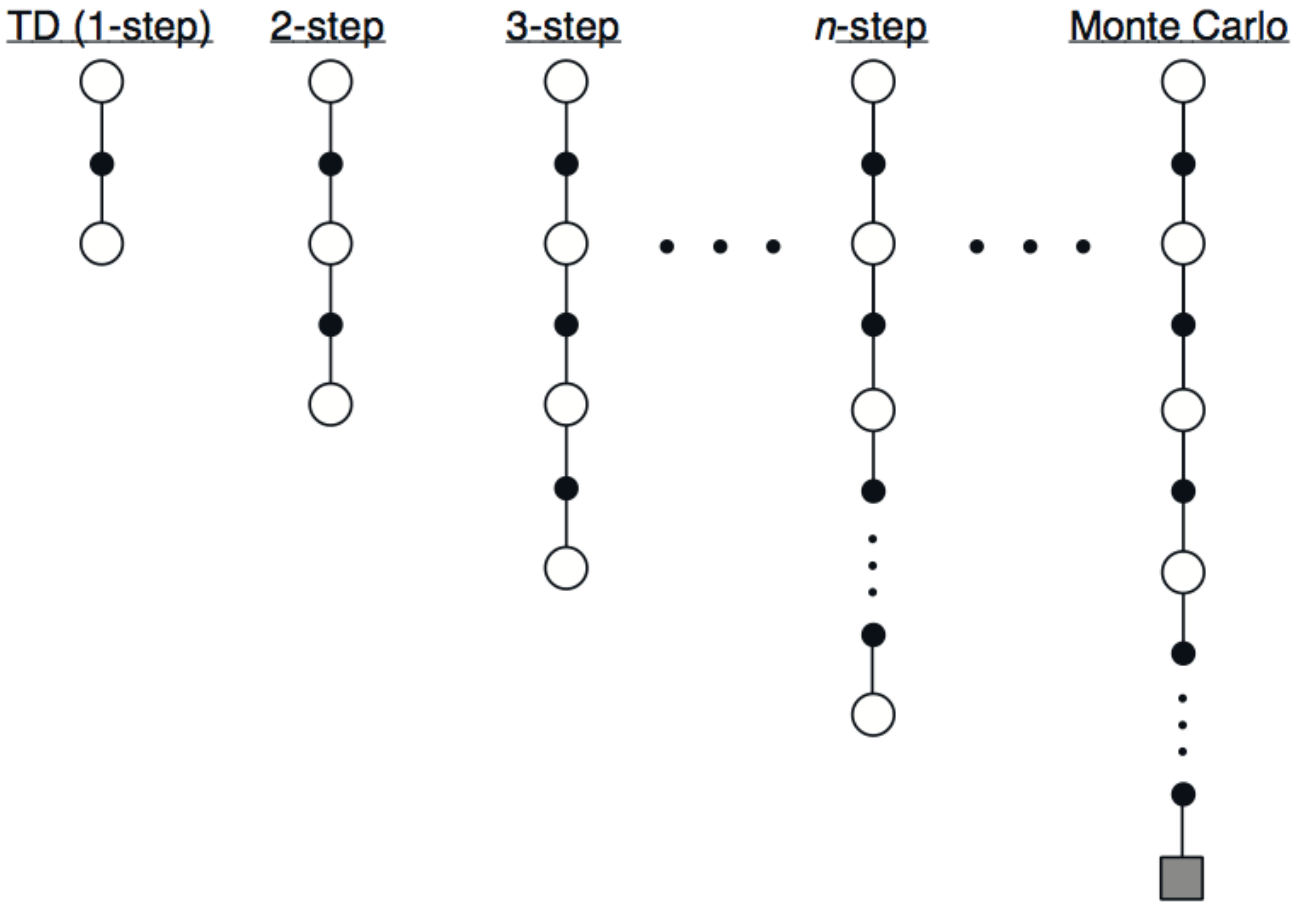
SARSA (Rummery and Niranjan 1994, who called it modified Q-learning) is an on-policy temporal difference learning method.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, a) - Q(S_t, A_t)]$$

Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
  Initialize s
  Choose a from s using policy derived from Q (e.g., ε-greedy)
  Repeat (for each step of episode):
    Take action a, observe r, s'
    Choose a' from s' using policy derived from Q (e.g., ε-greedy)
    $Q(s, a) \leftarrow Q(s, a) + \alpha\left[r + \gamma Q(s', a') - Q(s, a)\right]$
    $s \leftarrow s'$; $a \leftarrow a'$;
  until s is terminal

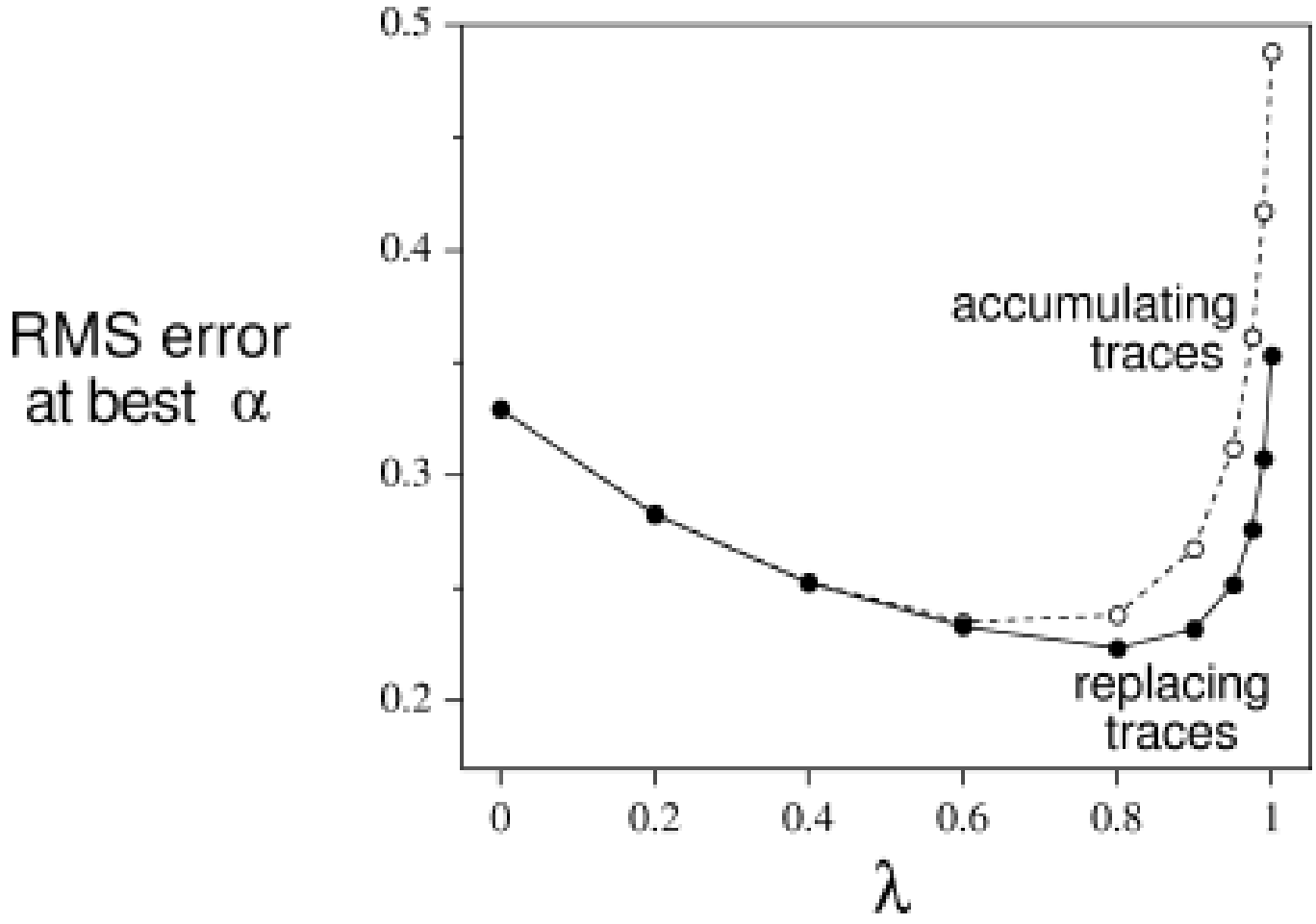
Eligibility Traces

Eligibility Traces provide a mechanism for assigning credit more quickly.

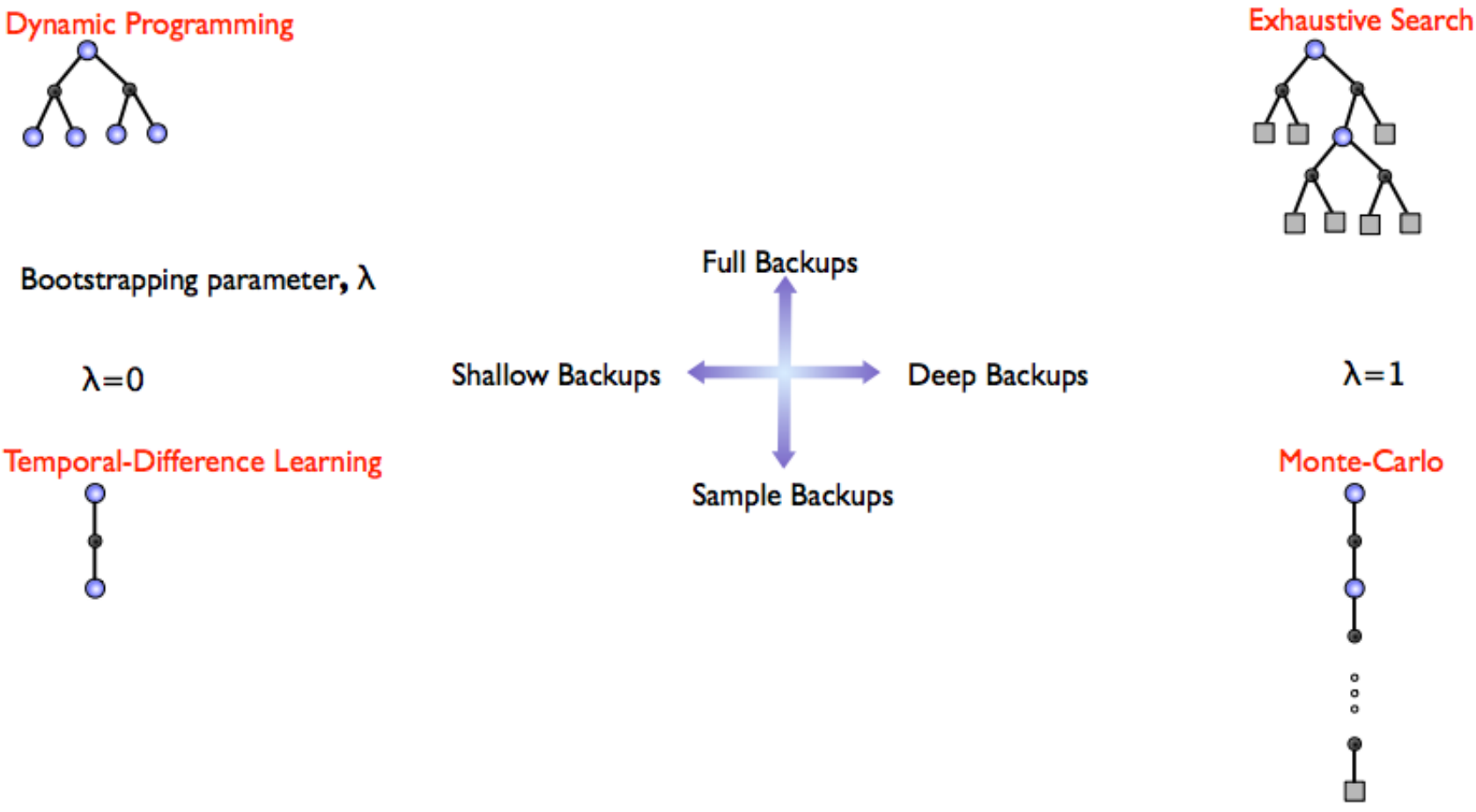


Eligibility Traces

Eligibility Traces provide a mechanism for assigning credit more quickly, and can improve learning.



Methods, the Unified View



The space of RL methods (from Maei (2011) "Gradient Temporal-Difference Learning Algorithms")

Outline

The recipe (some context)

Looking at other pi's (motivating examples)

Prep the ingredients (the simplest example)

Mixing the ingredients (models)

Baking (methods)

Eat your own pi (code)

I ate the whole pi, but I'm still hungry! (references)

From Theory to Practice

A tour of reinforcement learning software.

The State of Open Source RL

There are a number of projects that provide RL algorithms:

- RL-Glue/RL-Library (Multi-Language)
- RLPark (Java)
- PyBrain (Python)
- RLPy (Python)
- RLToolkit (Python, paper)
- RL Toolbox (C++) (Master's Thesis)

RL-Glue

RL-Glue is a fundamental library for the RL community.

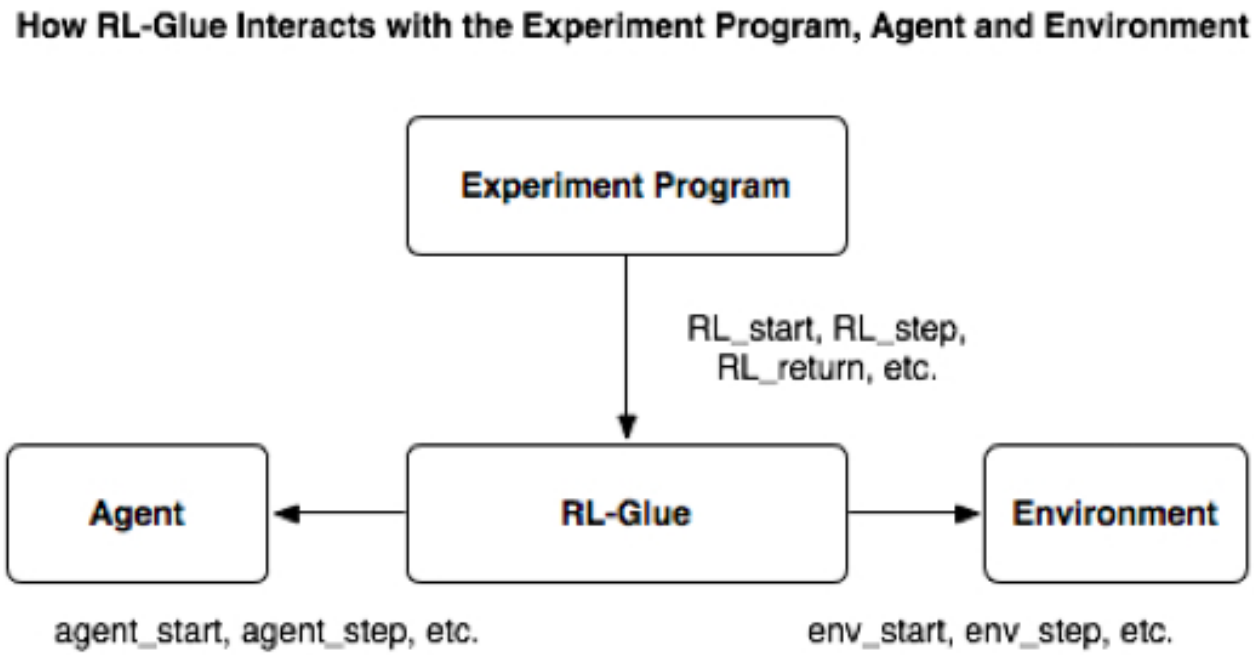


Figure: RL-Glue standard

RL-Glue: Codecs

RL-Glue currently offers codecs for multiple languages (see RL-Glue Extensions):

- C/C++
- Lisp
- Java
- Matlab
- Python

We recently created the *rlglue* R package:

```
library(devtools); install_github("smc77/rlglue")
```

RL R Package

The RL package in R is intended for three things:

- **Clear** RL algorithms for education
- **Generic**, reusable models that can be applied to any dataset
- Sophisticated, cutting edge methods

It also includes features such as ensemble methods.

RL R Package (Roadmap)

- On-policy prediction: TD(λ)
- Off-policy prediction: GTD(λ), GQ(λ)
- On-policy control: Q(λ)
- Off-policy control: SARSA(λ)
- Acting: softmax, greedy, ϵ-greedy

Approach

The RL package in R follows a basic routine:

1. Define an *agent*
   - Specify a model (e.g. MDP, POMDP)
   - Choose a learning method (e.g. Value iteration, Q-Learning)
   - Choose a planning method (e.g. ϵ-greedy, UCB, bayesian)
2. Define an environment (a dataset or simulator, terminal state)
3. Run an experiment (number of episodes, specify ϵ)

The result of running a simulation is an RLModel object, which can hold several different utilities, including the optimal policy. The package also includes a number of examples (grid world, pole balancing).

Simulation

Similar to RLinterface in RLToolkit.
Methods:
step, steps, episode, episodes

Outline

The recipe (some context)

Looking at other pi's (motivating examples)

Prep the ingredients (the simplest example)

Mixing the ingredients (models)

Baking (methods)

Eat your own pi (code)

I ate the whole pi, but I'm still hungry! (references)

Try this at home!

All the source code from this talk is available at:
https://github.com/smc77/rl

Other open source software:

- RL-Glue/RL-Library (Multi-Language)
- RLPark (Java)
- PyBrain (Python)
- RLPy (Python)
- RLToolkit (Python, paper)
- RL Toolbox (C++) (Master's Thesis)

Community

- https://groups.google.com/forum/!forum/rl-list
- glue.rl-community.org/
- http://www.rl-competition.org/

Papers

Surveys:

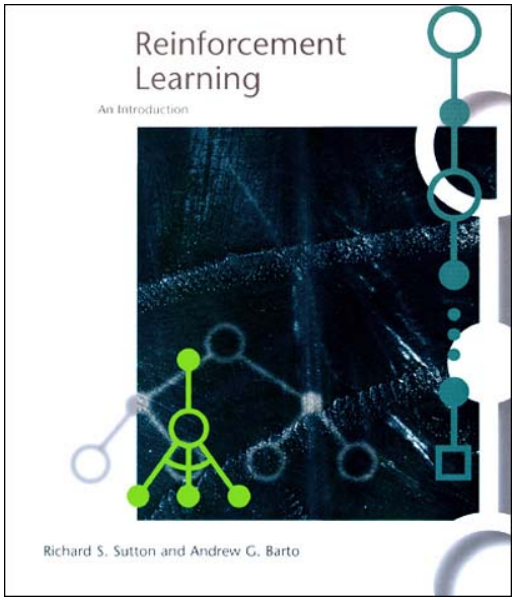
- Kaelbling, Littman, and Moore (1996) "Reinforcement Learning: A Survey"
- Littman (1996) "Algorithms for Sequential Decision Making"
- Kober, Bagnell, and Peters (2013) "Reinforcement Learning in Robotics: A Survey"

Books (AI/ML/Robotics/Planning)

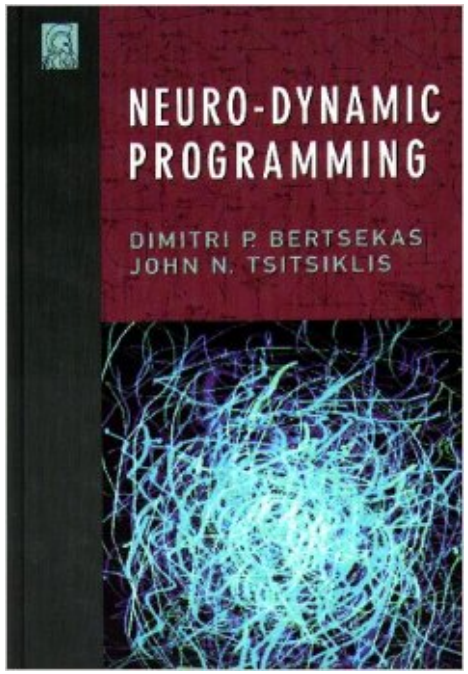
These are general textbooks that provide a good overview of reinforcement learning.

- Russell and Norvig (2010) "Artifical Intelligence: A Modern Approach"
- Ghallab, Nau, and Traverso "Automated Planning: Theory Practice"
- Thurn "Probabilistic Robotics"
- Poole and Mackworth (2010) "Artificial Intelligence: Foundations of Computational Agents"
- Mitchell (1997) "Machine Learning"
- Marsland (2009) "Machine Learning: An Algorithmic Perspective"

Books (RL)



Sutton and Barto (1998)
"Reinforcement Learning: An Introduction"

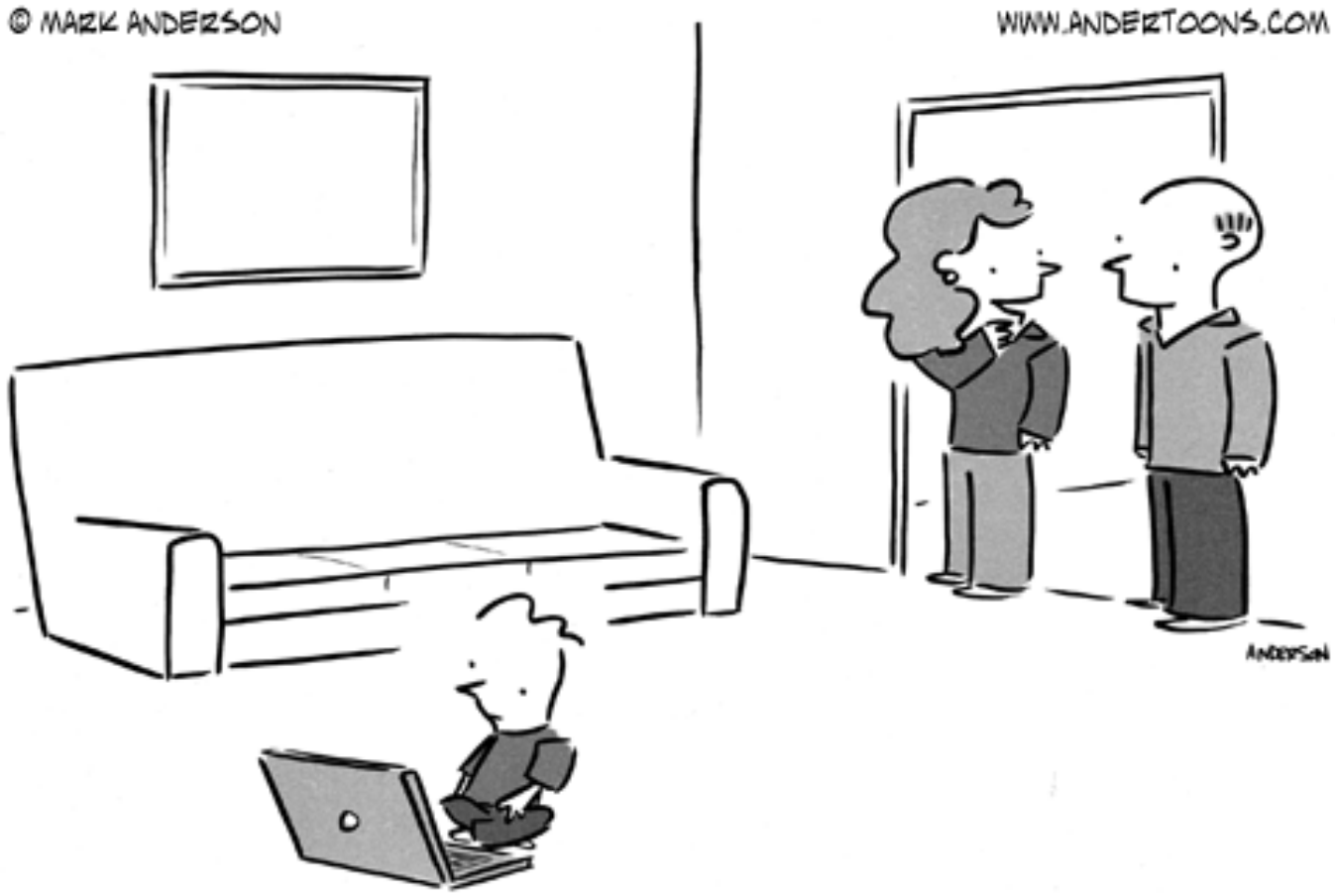


Bertsekas and Tsitsiklis (1996)
"Neuro-Dynamic Programming"

- "Reinforcement Learning: State-of-the-Art"
- Csaba Szepesvari (2009) "Algorithms for Reinforcement Learning"

People

- **Richard Sutton** (Alberta) - http://webdocs.cs.ualberta.ca/ sutton/
- Andrew Barto (UMass) - http://www-anw.cs.umass.edu/ barto/
- Michael Littman (Brown) - http://cs.brown.edu/ mlittman/
- Benjamin Van Roy (Stanford) - http://web.stanford.edu/ bvr/
- Leslie Kaelbling (MIT) - http://people.csail.mit.edu/lpk/
- Emma Brunskill (CMU) - http://www.cs.cmu.edu/ ebrun/
- Dimitri Bertsekas (MIT) - http://www.mit.edu/ dimitrib/home.html
- Csaba SzepesvÃąri - http://www.ualberta.ca/ szepesva/
- Chris Watkins - http://www.cs.rhul.ac.uk/home/chrisw/
- Lihong Li (Microsoft) - http://www.research.rutgers.edu/ lihong/
- John Langford (Microsoft) - http://hunch.net/ jl/
- Hado von Hasselt - http://webdocs.cs.ualberta.ca/ vanhasse/



"Honey, look! He's already reinforcing his ideologies!"

Questions?