CS 232: Artificial Intelligence

Spring 2024

Prof. Carolyn Anderson
Wellesley College

Reminders

- I have help hours today from 3:30-4:30 and on Monday from 4-5:15
- Midterm is 1 week away (no HW until then)
- Bring midterm questions to class on Tuesday, we'll have a brief review
- Read YLLATAILY Chapters 5-6 for next Tuesday

How To Explore

Passive Reinforcement Learning

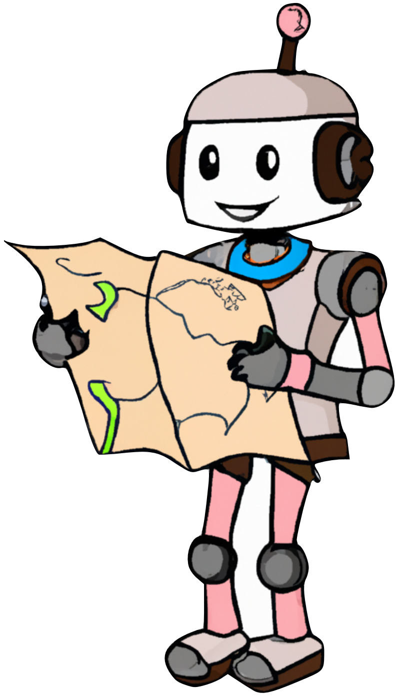
- Take a fixed policy and follow it

Random Exploration

- Pick actions randomly

Balancing Exploration and Exploitation

- Pick actions randomly initially, but gradually switch to mostly doing actions that you have already found to be valuable



How To Evaluate

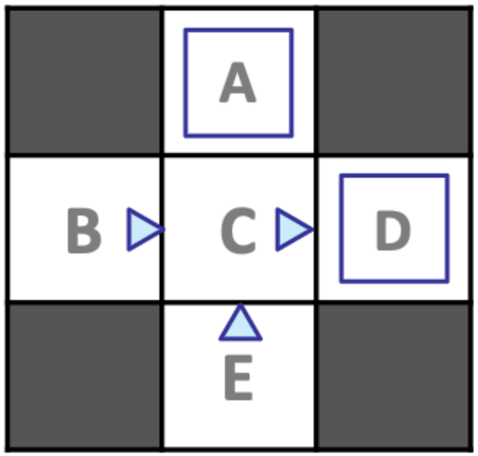
In Direction Evaluation, we repeatedly follow the same policy and observe its rewards.

Direct Evaluation:

- Act according to policy π
- In each state, write down the sum of discounted rewards
- Average samples

Example: Direct Evaluation

Input Policy π



Assume: $\gamma = 1$

Observed Episodes (Training)

Episode 1

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 2

B, east, C, -1
C, east, D, -1
D, exit, x, +10

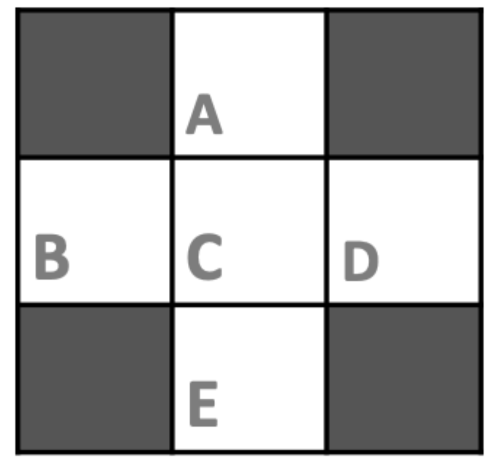
Episode 3

E, north, C, -1
C, east, D, -1
D, exit, x, +10

Episode 4

E, north, C, -1
C, east, A, -1
A, exit, x, -10

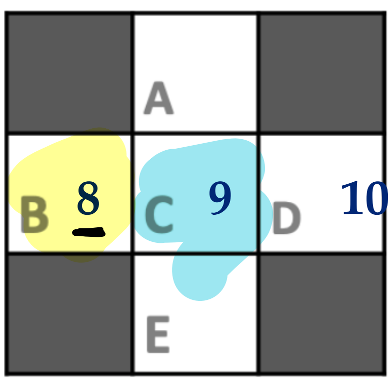
Output Values

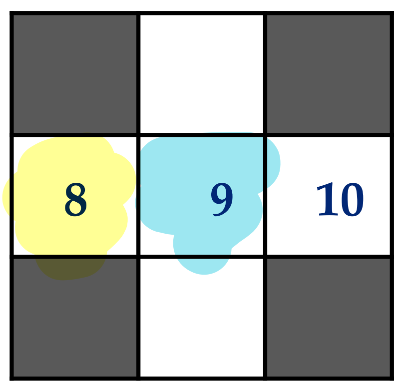


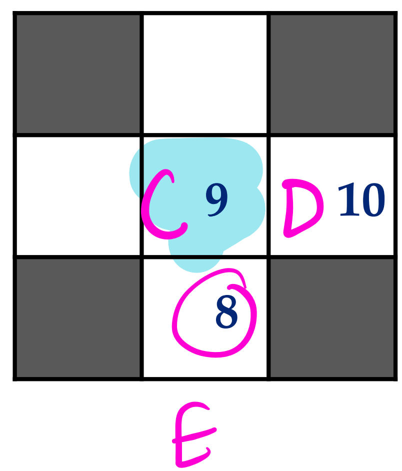
Example from CS188 Intro to AI at UC Berkeley

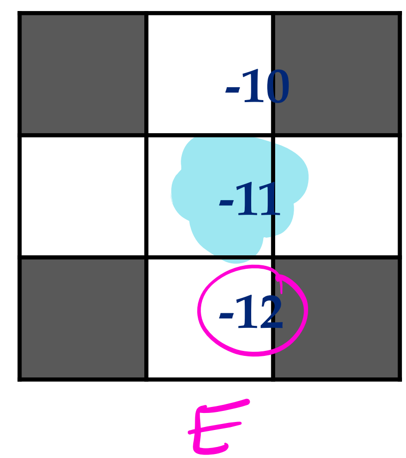
Example: Direct Evaluation

Episode 1	Episode 2	Episode 3	Episode 4
B, east, C, -1	B, east, C, -1	E, north, C, -1	E, north, C, -1
C, east, D, -1	C, east, D, -1	C, east, D, -1	C, east, A, -1
D, exit, x, +10	D, exit, x, +10	D, exit, x, +10	A, exit, x, -10
8	8	8	-12





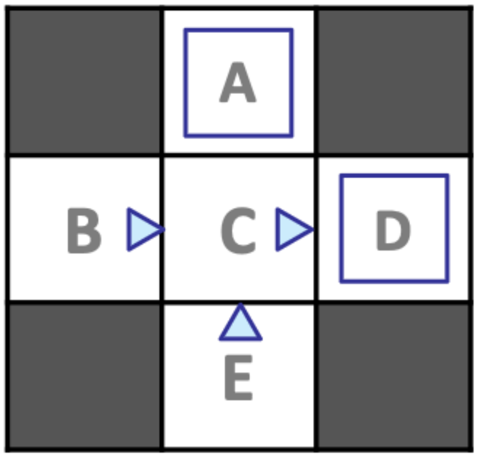




$\gamma = 1$

Example: Direct Evaluation

Input Policy π



Assume: $\gamma = 1$

Observed Episodes (Training)

Episode 1

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 2

B, east, C, -1
C, east, D, -1
D, exit, x, +10

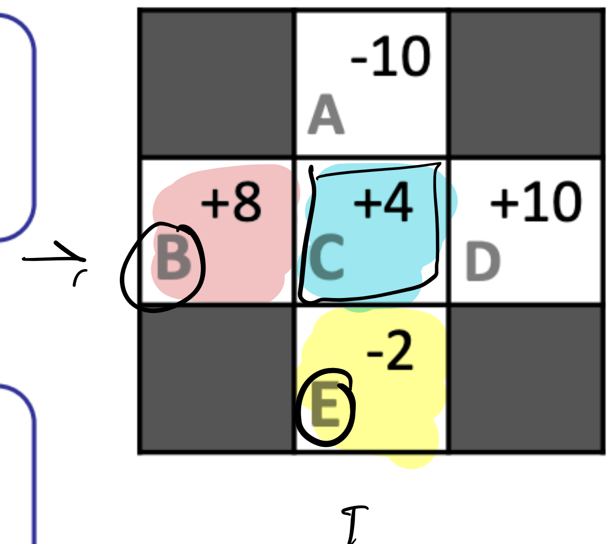
Episode 3

E, north, C, -1
C, east, D, -1
D, exit, x, +10

Episode 4

E, north, C, -1
C, east, A, -1
A, exit, x, -10

Output Values



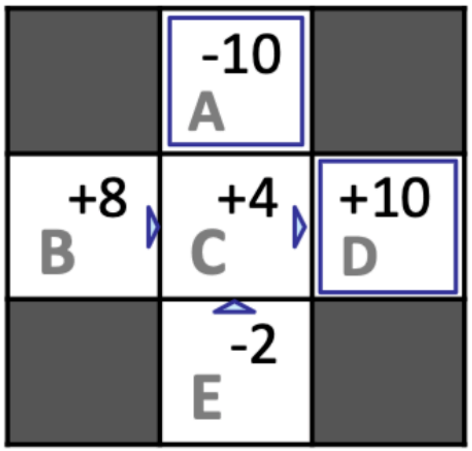
Weaknesses of Direct Evaluation

Direct Evaluation works.

But, we had to learn each state separately.

That isn't very efficient!

Output Values



If B and E both go to C under this policy, how can their values be different?

Temporal Difference Learning

Temporal Difference Learning

Main Idea: learn from every experience

α : learning rate

"How willing are we to believe new evidence"

TD Learning

- Update V(s) every time we make a transition T(s,a,s',r)
- Most likely outcomes (s') will contribute more updates

Reward for leaving s to reach s' by policy $\pi(s)$

Value of state reached (s')

Sample of V(s): $sample = R(s, \pi(s), s') + \gamma V^{\pi}(s')$

discount

Value of state s

Update to V(s): $V^{\pi}(s) \leftarrow (1-\alpha)\underbrace{V^{\pi}(s)}_{\text{old estimate}} + (\alpha)\underbrace{sample}_{\text{new info}}$

old est.

Same update: $V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha(sample - V^{\pi}(s))$

new info

TD: A Moving Average

In TD Learning, we average our observations in a way that weights more recent samples highly.

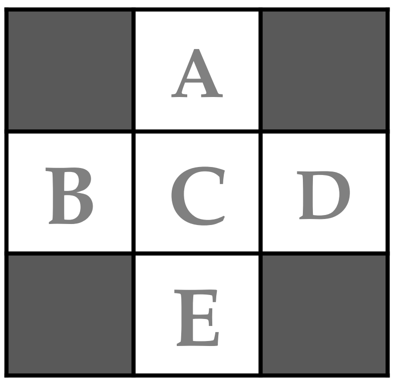
Over time, we forget our initial estimates (which probably weren't very good!)

$$\bar{x}_n = \frac{x_n + (1-\alpha)\cdot x_{n-1} + (1-\alpha)^2 \cdot x_{n-2} + \dots}{1 + (1-\alpha) + (1-\alpha)^2 + \dots}$$

We decrease the **learning rate** α over time.

Example: Temporal Difference Learning

States



Assume:

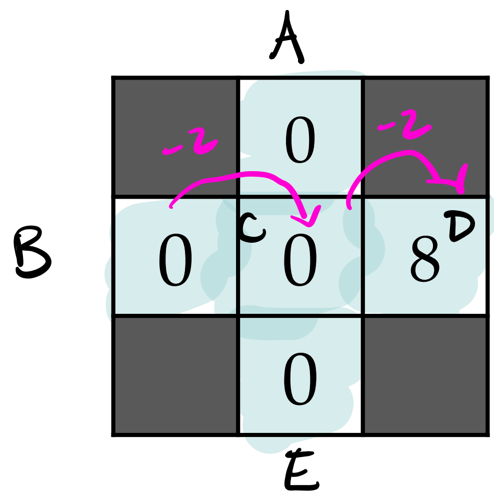
$\gamma = 1,$

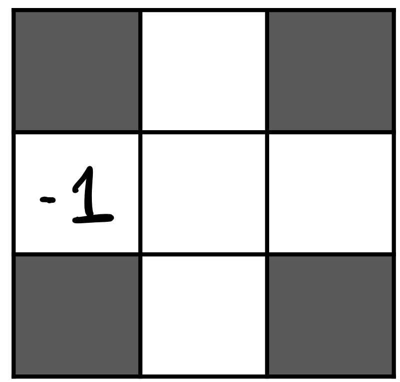
$\alpha = 1/2$

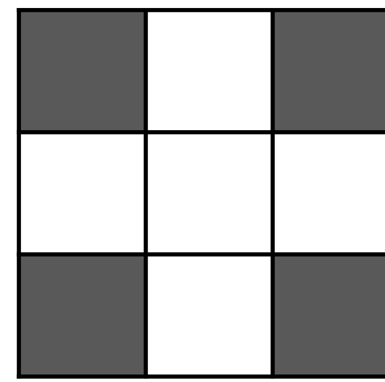
Observed Transitions

B, east, C, -2

C, east, D, -2







$$V^\pi(s) \leftarrow (1-\alpha)V^\pi(s) + \alpha\left[R(s,\pi(s),s') + \gamma V^\pi(s')\right]$$

$$V(B) = \underbrace{(1-0.5)\cdot 0 + 0.5[-2 + \underbrace{1\cdot 0}]}_{}$$

$$= -1$$

Example from CS188 Intro to AI at UC Berkeley

Example: Temporal Difference Learning

States

	A	
B	C	D
	E	

Assume:

$\gamma = 1,$

$\alpha = 1/2$

Observed Transitions

B, east, C, -2

C, east, D, -2

	0	
0	0	8
	0	

	0	
-1	3	8
	0	

$$V^{\pi}(s) \leftarrow (1-\alpha)V^{\pi}(s) + \alpha\left[R(s,\pi(s),s') + \gamma V^{\pi}(s')\right]$$

$$V(C) \leftarrow \underbrace{(1-0.5)\,0}_{} + 0.5[-2 + 1\cdot 8]$$

$$= 3$$

Example from CS188 Intro to AI at UC Berkeley

Example: Temporal Difference Learning

States

	A	
B	C	D
	E	

Assume:

$\gamma = 1$,

$\alpha = 1/2$

Observed Transition:

B, east, C, -2

	0	
0	0	8
	0	

$$V^{\pi}(s) \leftarrow (1-\alpha)V^{\pi}(s) + \alpha\left[R(s,\pi(s),s') + \gamma V^{\pi}(s')\right]$$

Example: Temporal Difference Learning

States

	A	
B	C	D
	E	

Assume:

$\gamma = 1,$

$\alpha = 1/2$

Observed Transition:

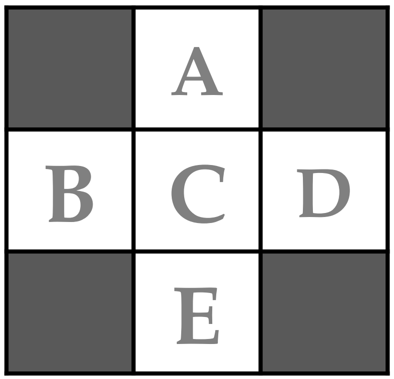
C, east, D, -2

	0	
0	0	8
	0	

$$V^{\pi}(s) \leftarrow (1-\alpha)V^{\pi}(s) + \alpha\left[R(s,\pi(s),s') + \gamma V^{\pi}(s')\right]$$

Example: Temporal Difference Learning

States



Assume:

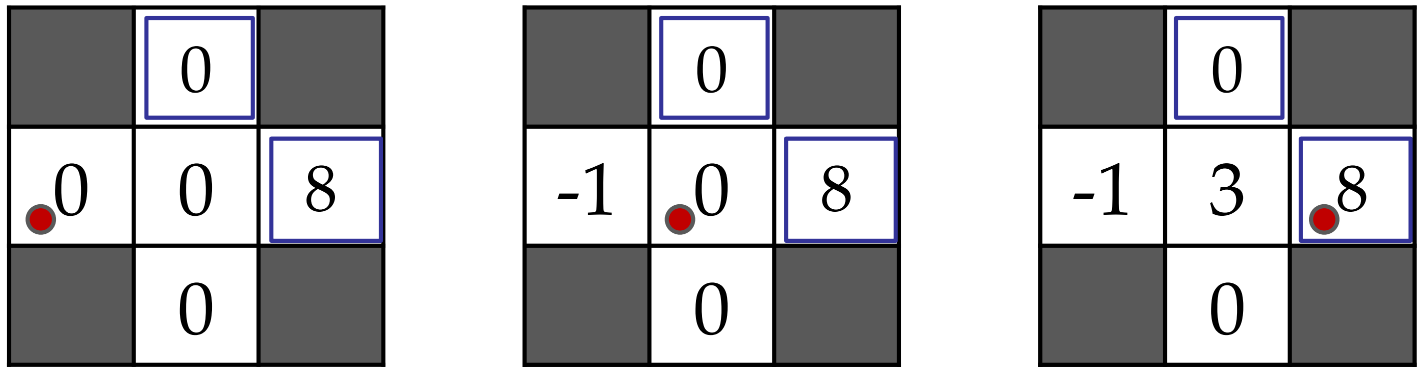
$\gamma = 1,$

$\alpha = 1/2$

Observed Transitions

B, east, C, -2

C, east, D, -2

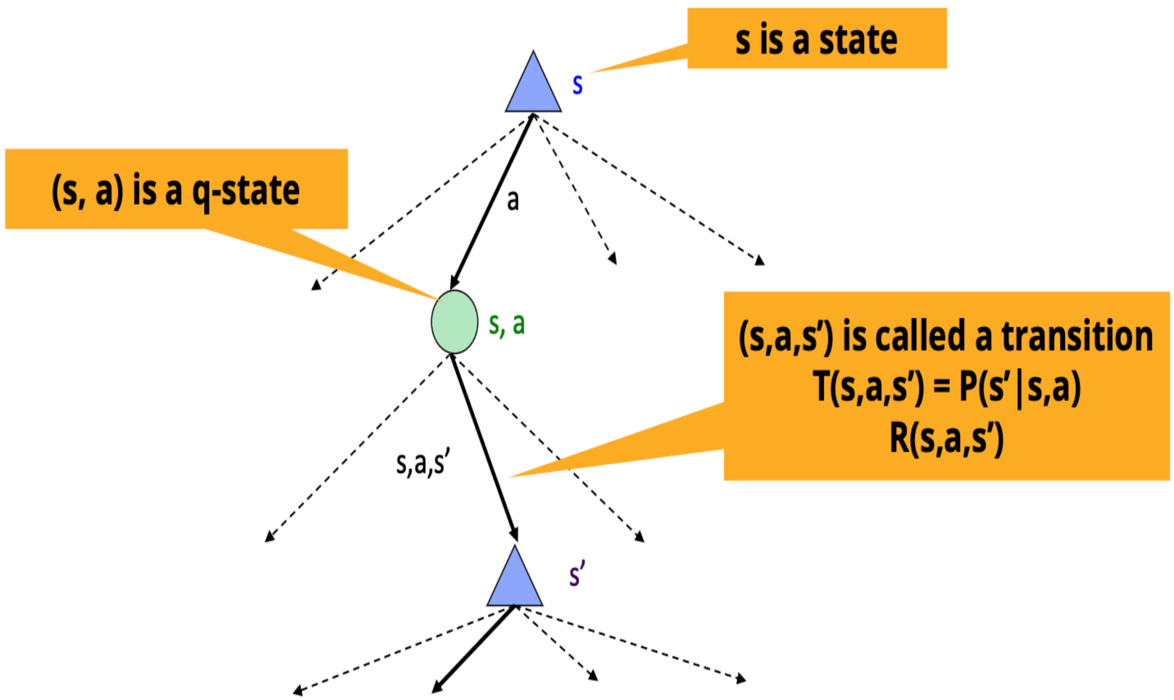


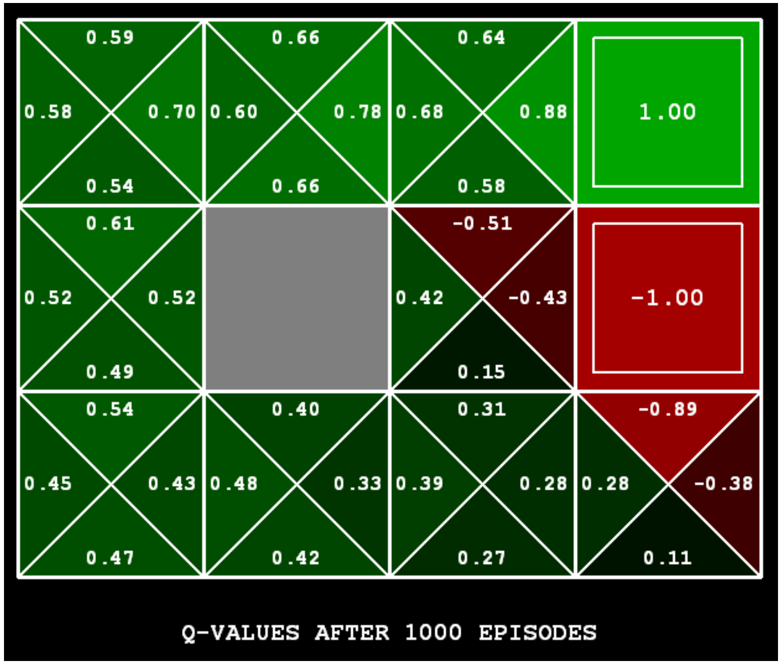
$$V^\pi(s) \leftarrow (1-\alpha)V^\pi(s) + \alpha\left[R(s,\pi(s),s') + \gamma V^\pi(s')\right]$$

Q-Learning

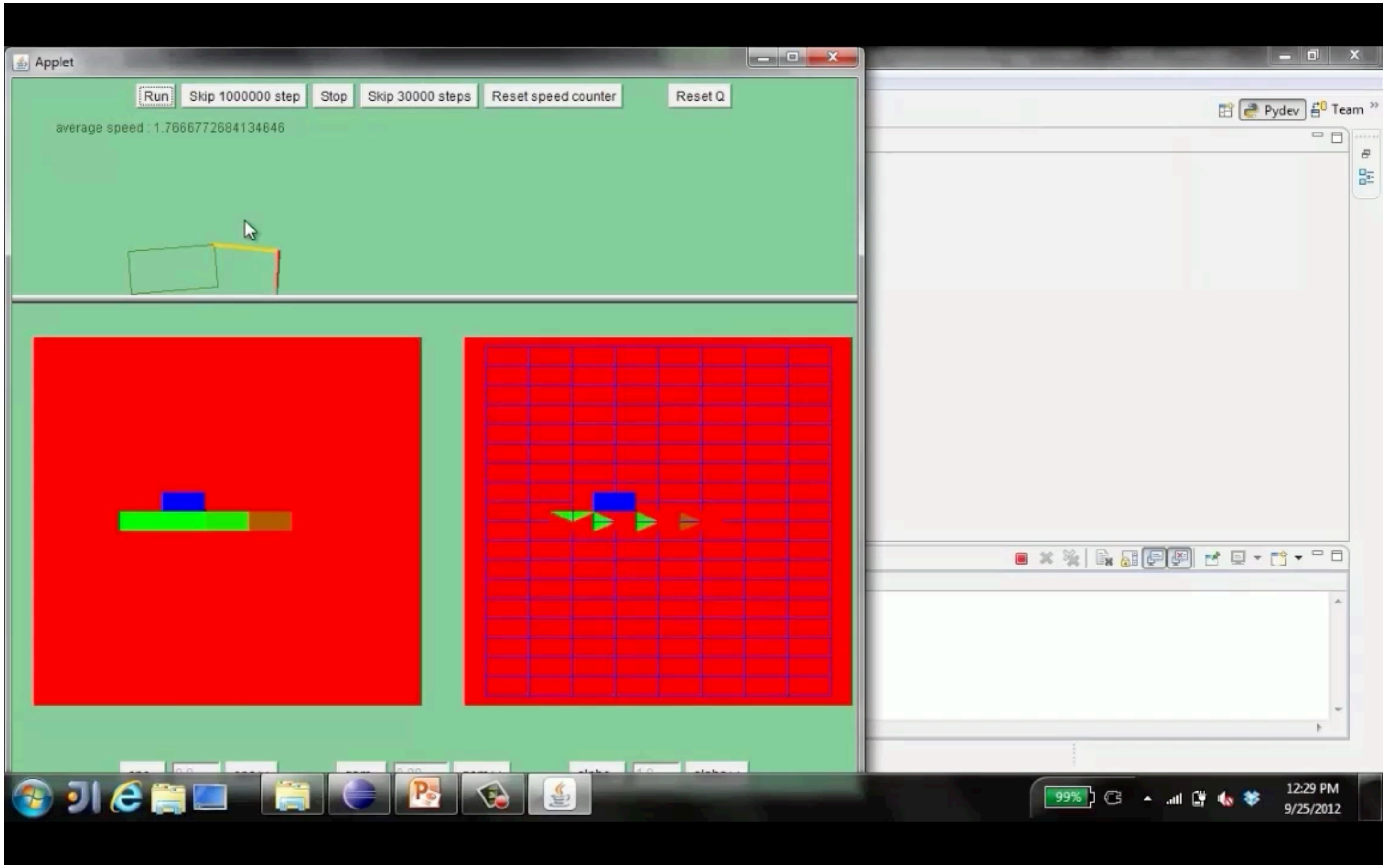
In practice, we want to know the value of state-action pairs, not just states.

This version of TD Learning is called **Q-Learning**, because the value of a state-action pair is called a **q-value**.





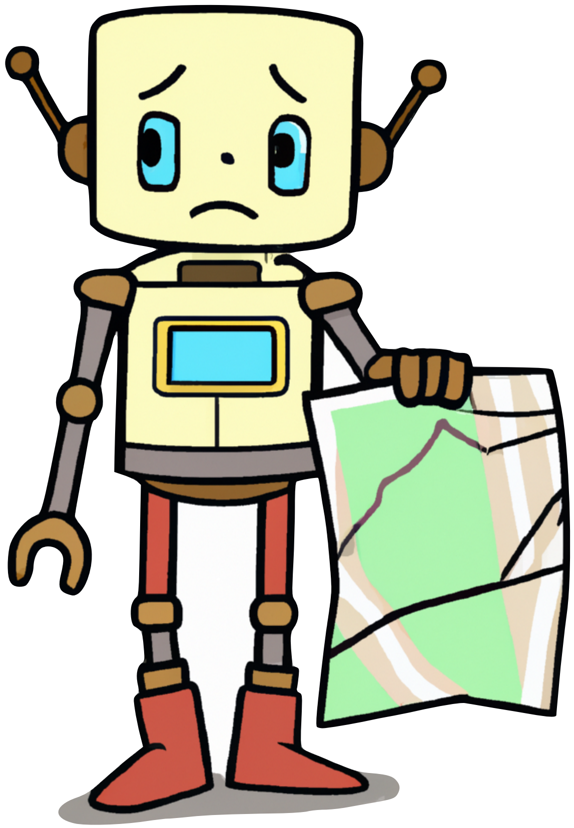
Demo of Q-Learning



Regret

Even if our Reinforcement Learning agent learns an optimal policy eventually, it will still take sub-par actions along the way.

Regret is the total cost of all of those mistakes: the differences between optimal expected rewards and the agent's actual rewards.



The best RL agent is one that **minimizes regret**: learns optimally!

Semester Road Map

Almost all AI tasks can be grouped into one of three main categories:

- Search
- Classification
- Generation

Classification Methods

Classification Methods: Supervised Machine Learning

Lots of kinds!

- Naïve Bayes
- Logistic regression
- Neural networks
- k-Nearest Neighbors
- random forests
- …

Classification Methods: Supervised Machine Learning

Input:

- a input d
- a fixed set of classes $C = \{c_1, c_2 \dots c_j\}$
- A training set of *m* hand-labeled examples
  $(d_1, c_1), \dots (d_m, c_m)$

Output:

- a learned classifier $y: d \to c$

Components of a probabilistic machine learning classifier

Given *m* input/output pairs $(x^{(i)}, y^{(i)})$:

1. A **feature representation** of the input

   For each input x^i, a vector of features $[x_1, x_2 \dots x_n]$

   Feature j for x^i as x_j or x_j^i

2. A **classification function** that takes x and compute $\hat{y}$, the estimated class label for x by estimating $p(y|x)$

3. An objective function for learning, like **cross-entropy loss**

4. An algorithm for optimizing the objective function: **stochastic gradient descent**.

Break: Classifiers and Features

Classifiers & Features

ReCaptcha

Resume screening

Document classification

Spam/junk mail detection.

Photo recognition

Plant recognition

Reverse image search

Fraud detection

Recommendation systems

Animal behavior detection

Dating apps

Medical tests

Logistic Regression Classifiers

Is this spam?

Subject: **Important notice!**
From: Stanford University <newsforum@stanford.edu>
Date: October 28, 2011 12:34:16 PM PDT
To: undisclosed-recipients:;

Greats News!

You can now access the latest news by using the link below to login to Stanford University News Forum.

http://www.123contactform.com/contact-form-StanfordNew1-236335.html

Click on the above link to login for more information about this new exciting forum. You can also copy the above link to your browser bar and login for more information about the new services.

© Stanford University. All Rights Reserved.

Who wrote which Federalist papers?

Anonymous essays try to convince New York to ratify U.S Constitution. Authorship of 12 of the letters in dispute.

Solved by Mosteller and Wallace (1963) using Bayesian methods

What is the subject of this research article?

MEDLINE Article

?

Antogonists and Inhibitors
Blood Supply
Chemistry
Drug Therapy
Embryology
Epidemiology
…

Text Classification: definition

Input:

- a document d
- a fixed set of classes $C = \{c_1, c_2, \ldots c_j\}$

Output: a predicted class $\hat{y} \in C$

Binary Classification in Logistic Regression

Given a series of input/output pairs:

(x^i, y^i)

For each observation $x^{(i)}$

- We represent $x^{(i)}$ by a feature vector $[x_1, x_2 \dots x_n]$
- We compute an output $\hat{y}^i \in \{0,1\}$

Features in logistic regression

For feature x_i, weight w_i tells is how important is x_i

- x_i ="review contains '`marvelous`'": $w_1 = +10$
- x_j ="review contains '`awful`'": $w_j = -5$
- x_k ="review contains '`disappoint`'": $w_k = -2$

Logistic Regression for one observation x

Input observation: $x = [x_1, x_2 \dots x_n]$

Weights (one per feature): $W = [w_1, w_2 \dots w_n]$

$\theta = [\theta_1, \theta_2 \dots \theta_3]$

Output: a predicted class $\hat{y} \in \{0, 1\}$

Neg Pos

How to do classification

For each feature x_i, weight w_i tells us importance of x_i

We'll sum up all the weighted features and the bias:

$$z = \left(\sum_{i=1}^{n} w_i x_i \right) + b$$

b : bias

$$z = wx + b$$

slope

intercept

If $z > 0$: we say 1 (+)

Otherwise : we vote 0 (-)

But we want a probabilistic classifier

We need to formalize "sum is high".

We want a model that can tell us:

$$p(y=1 \mid x; w)$$

$$p(y=0 \mid x; w)$$

The very useful sigmoid or logistic function