

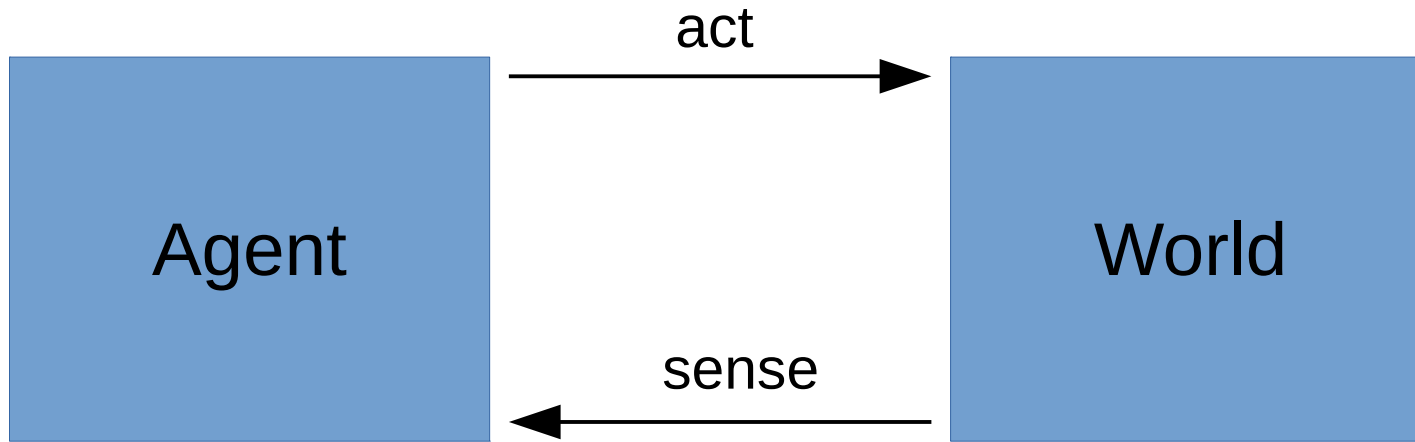
Reinforcement Learning

Robert Platt
Northeastern University

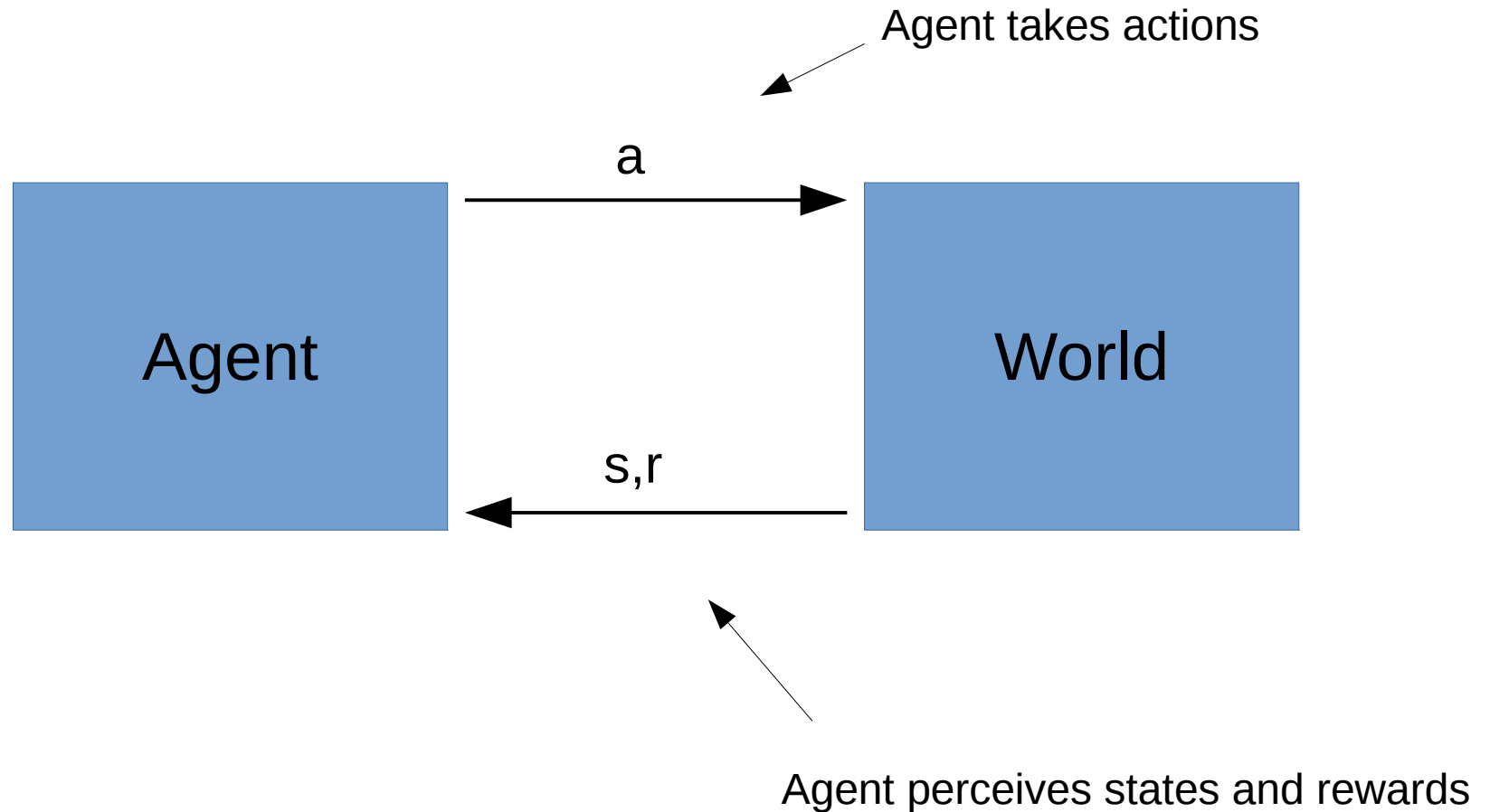
Some images and slides are used from:

1. CS188 UC Berkeley
2. RN, AIMA

Conception of agent

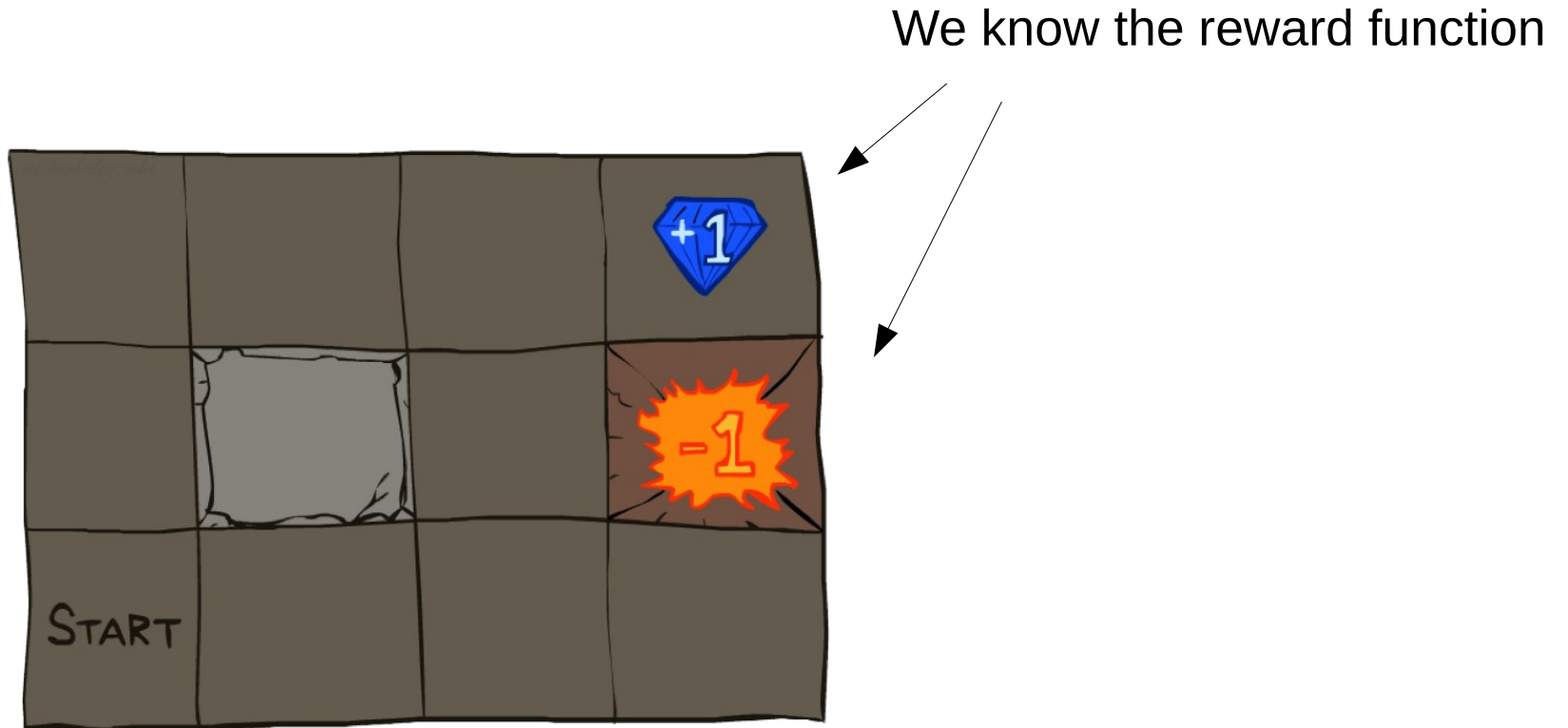


RL conception of agent



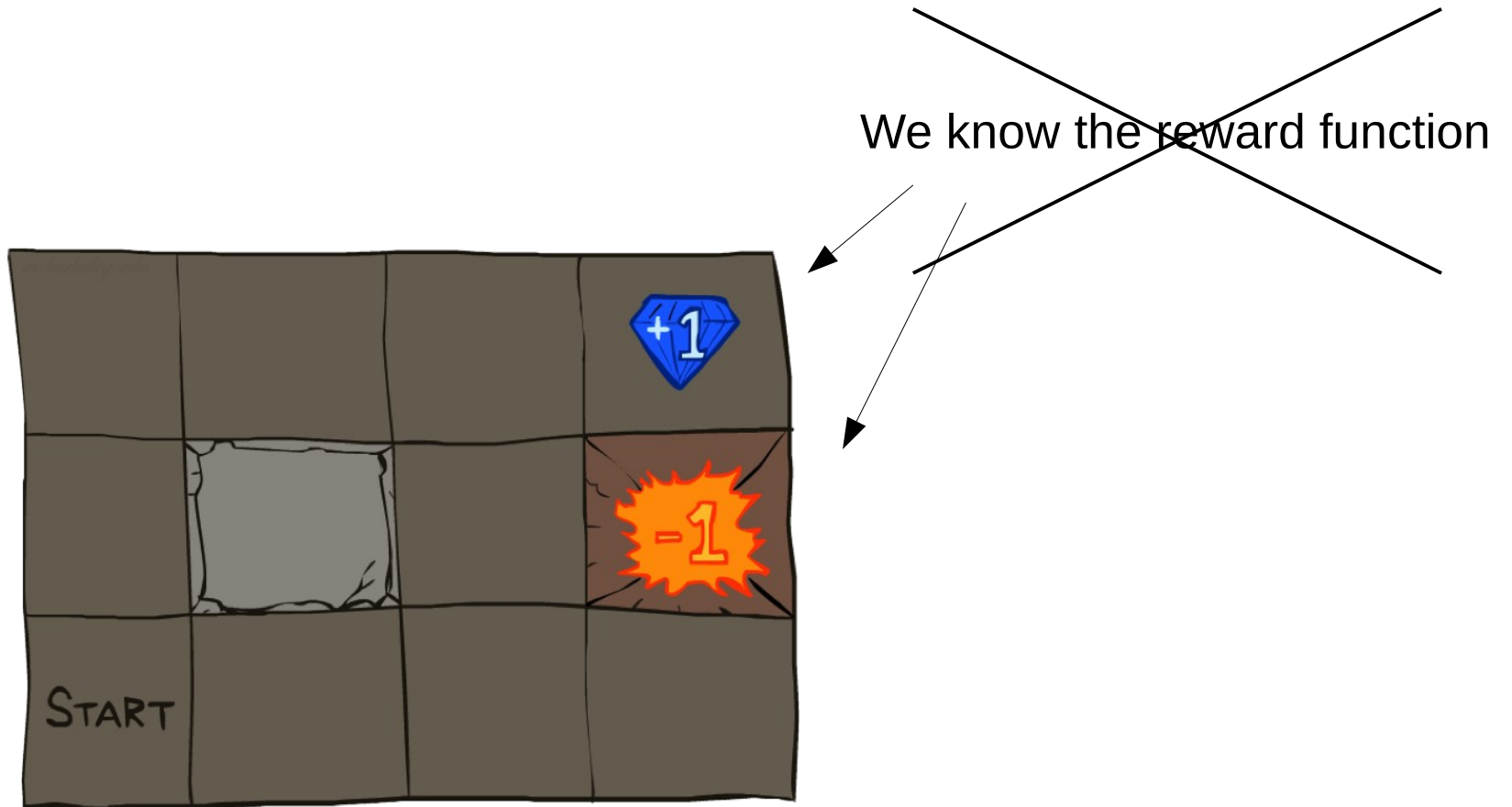
Transition model and reward function are initially unknown to the agent!
– value iteration assumed knowledge of these two things...

Value iteration



We know the probabilities of moving in each direction when an action is executed

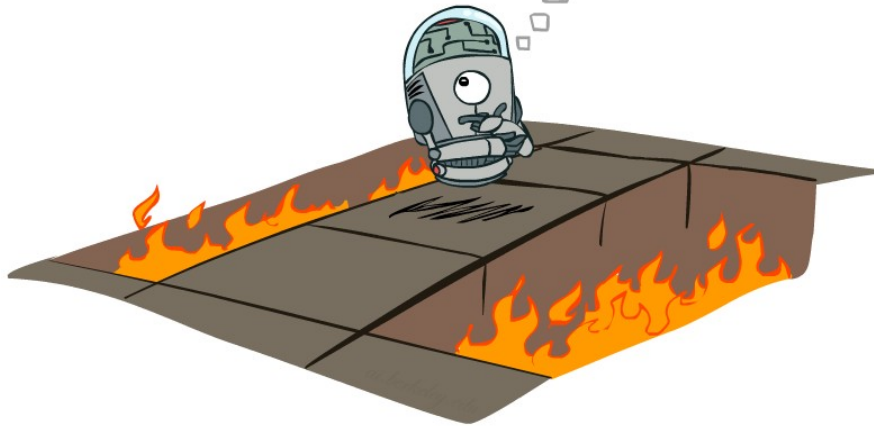
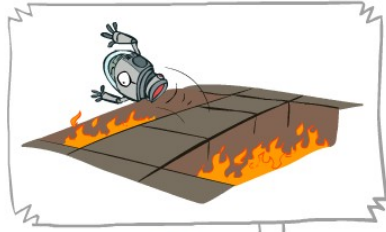
Reinforcement Learning



~~We know the reward function~~

~~We know the probabilities of moving in each direction when an action is executed~~

The different between RL and value iteration

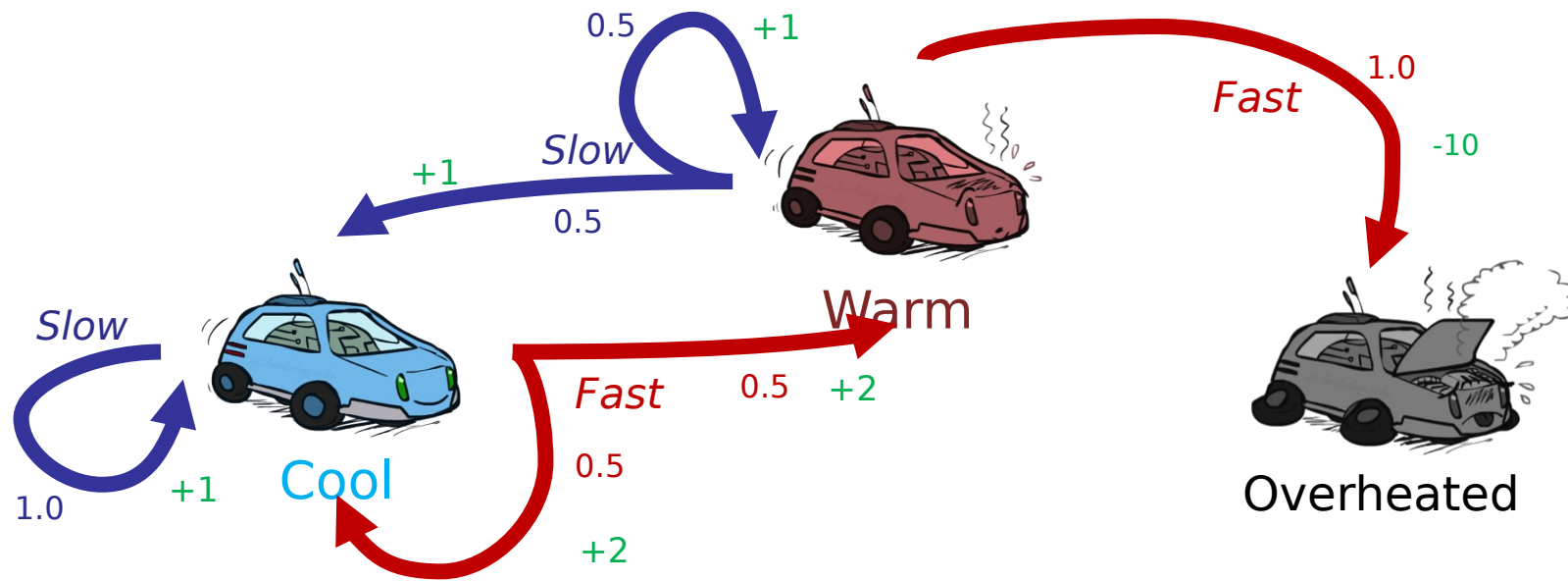


Offline Solution
(value iteration)



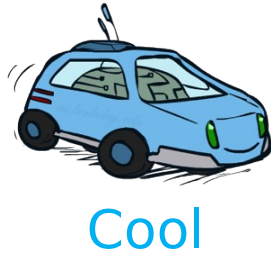
Online Learning
(RL)

Value iteration vs RL



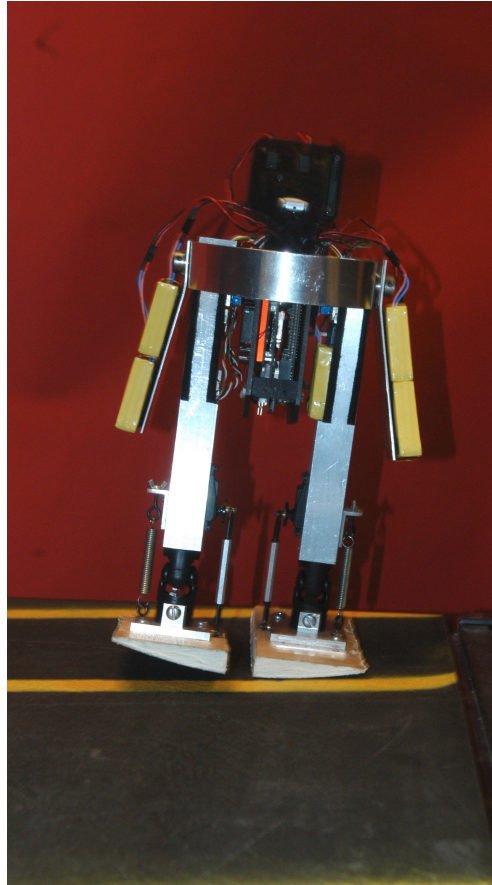
RL still assumes that we have an MDP

Value iteration vs RL



RL still assumes that we have an MDP
– but, we assume we don't know T or R

RL example

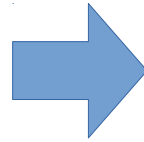


<https://www.youtube.com/watch?v=goqWX7bC-ZY>

Model-based RL

1. estimate T , R by averaging experiences

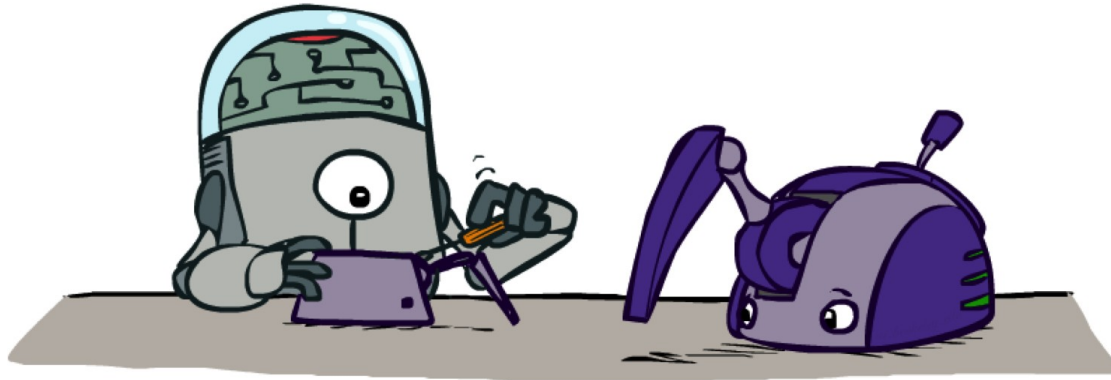
2. solve for policy using value iteration



a. choose an exploration policy
– policy that enables agent to explore all relevant states

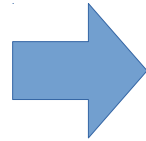
b. follow policy for a while

c. estimate T and R



Model-based RL

1. estimate T , R by averaging experiences



2. solve for policy using value iteration

a. choose an exploration policy
– policy that enables agent to explore all relevant states

b. follow policy for a while

c. estimate T and R

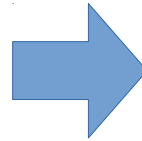
$N_{s,a,s'} \equiv$ Number of times agent reached s' by taking a from s

$R_{s,a,s'} \equiv$ Set of rewards obtained when reaching s' by taking a from s

$$T(s, a, s') \approx \frac{N_{s,a,s'}}{\sum_{s'} N_{s,a,s'}} \quad R(s, a, s') \approx \frac{1}{N_{s,a,s'}} R_{s,a,s'}$$

Model-based RL

1. estimate T, R by averaging experiences



a. choose an exploration policy
– policy that enables agent to explore all relevant states

2. solve for policy via value iteration

What's wrong w/ this approach?

$$N_{s,a,s'} \equiv$$

$$R_{s,a,s'} \equiv$$

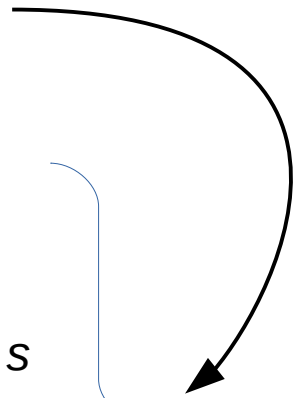
$$T(s, a, s') \approx \frac{N_{s,a,s'}}{\sum_{s'} N_{s,a,s'}}$$

$$R(s, a, s') \approx \frac{1}{N_{s,a,s'}} R_{s,a,s'}$$

a while

R

a from s



Model-based vs Model-free learning

Goal: Compute expected age of students in this class

Known

$$E[A] = \sum_a P(a) \cdot a \stackrel{P(A)}{=} 0.35 \times 20 + \dots$$

Without $P(A)$, instead collect samples $[a_1, a_2, \dots, a_N]$

Unknown $P(A)$: "Model Based"

Why does this work? Because eventually you learn the right model.

$$\hat{P}(a) = \frac{\text{num}(a)}{N}$$
$$E[A] \approx \sum_a \hat{P}(a) \cdot a$$

Unknown $P(A)$: "Model Free"

$$E[A] \approx \frac{1}{N} \sum_i a_i$$

Why does this work? Because samples appear with the right frequencies.

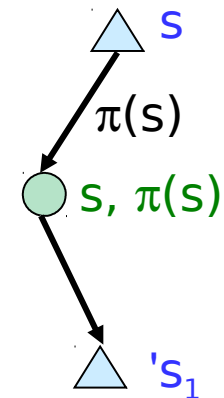
RL: model-free learning approach to estimating the value function

- We want to improve our estimate of V by computing these averages:

$$V_{k+1}^{\pi}(s) \leftarrow \sum_{s'} T(s, \pi(s), s') [R(s, \pi(s), s') + \gamma V_k^{\pi}(s')]$$

- Idea: Take samples of outcomes s' (by doing the action!) and average

$$\begin{aligned} \text{sample}_1 &= R(s, \pi(s), s'_1) + \gamma V_k^{\pi}(s'_1) \\ &\dots \end{aligned}$$



RL: model-free learning approach to estimating the value function

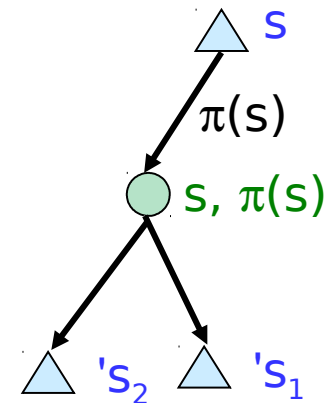
- We want to improve our estimate of V by computing these averages:

$$V_{k+1}^{\pi}(s) \leftarrow \sum_{s'} T(s, \pi(s), s') [R(s, \pi(s), s') + \gamma V_k^{\pi}(s')]$$

- Idea: Take samples of outcomes s' (by doing the action!) and average

$$\text{sample}_1 = R(s, \pi(s), s'_1) + \gamma V_k^{\pi}(s'_1)$$

$$\text{sample}_2 = R(s, \pi(s), s'_2) + \gamma V_k^{\pi}(s'_2)$$



RL: model-free learning approach to estimating the value function

- We want to improve our estimate of V by computing these averages:

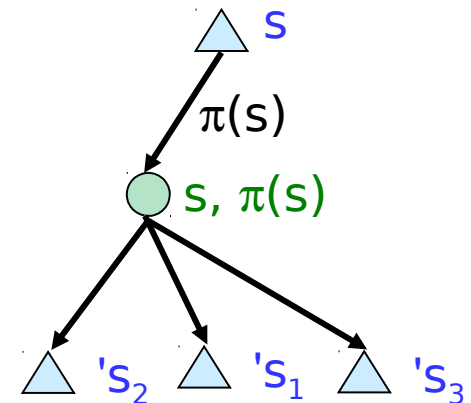
$$V_{k+1}^{\pi}(s) \leftarrow \sum_{s'} T(s, \pi(s), s') [R(s, \pi(s), s') + \gamma V_k^{\pi}(s')]$$

- Idea: Take samples of outcomes s' (by doing the action!) and average

$$\text{sample}_1 = R(s, \pi(s), s'_1) + \gamma V_k^{\pi}(s'_1)$$

$$\text{sample}_2 = R(s, \pi(s), s'_2) + \gamma V_k^{\pi}(s'_2)$$

$$\text{sample}_n = R(s, \pi(s), s'_n) + \gamma V_k^{\pi}(s'_n)$$



RL: model-free learning approach to estimating the value function

- We want to improve our estimate of V by computing these averages:

$$V_{k+1}^{\pi}(s) \leftarrow \sum_{s'} T(s, \pi(s), s') [R(s, \pi(s), s') + \gamma V_k^{\pi}(s')]$$

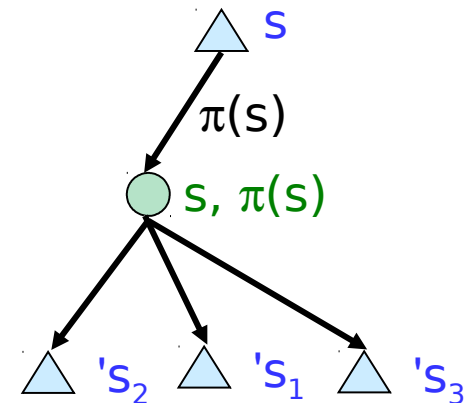
- Idea: Take samples of outcomes s' (by doing the action!) and average

$$sample_1 = R(s, \pi(s), s'_1) + \gamma V_k^{\pi}(s'_1)$$

$$sample_2 = R(s, \pi(s), s'_2) + \gamma V_k^{\pi}(s'_2)$$

$$sample_n = R(s, \pi(s), s'_n) + \gamma V_k^{\pi}(s'_n)$$

$$V_{k+1}^{\pi}(s) \leftarrow \frac{1}{n} \sum_i sample_i$$



Sidebar: exponential moving average

- Exponential moving average

- The running interpolation update:

$$\bar{x}_n = (1 - \alpha) \cdot \bar{x}_{n-1} + \alpha \cdot x_n$$

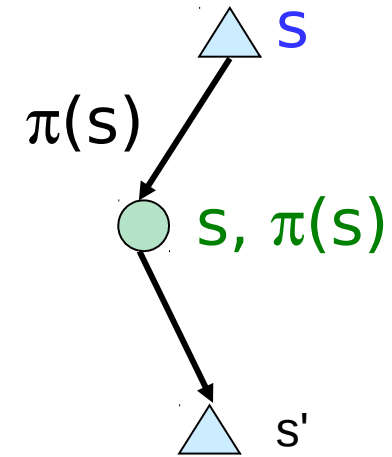
- Makes recent samples more important:

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

- Forgets about the past (distant past values were wrong anyway)

TD Value Learning

- Big idea: learn from every experience!
 - Update $V(s)$ each time we experience a transition (s, a, s', r)
 - Likely outcomes s' will contribute updates more often
- Temporal difference learning of values
 - Policy still fixed, still doing evaluation!
 - Move values toward value of whatever successor occurs: running average



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

Update to $V(s)$: $V^\pi(s) \leftarrow (1 - \alpha)V^\pi(s) + (\alpha)sample$

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

TD Value Learning: example

States

Observed
Transitions

	A	
B	C	D
	E	

	0	
0	0	8
	0	

Assume: $\gamma = 1$,
 $\alpha = 1/2$

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

TD Value Learning: example

States

	A	
B	C	D
	E	

Observed reward Observed Transitions

B, east, C, -2

	0	
0	0	8
	0	

	0	
-1	0	8
	0	

Assume: $\gamma = 1$,
 $\alpha = 1/2$

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

TD Value Learning: example

States

	A	
B	C	D
	E	

Observed reward

Observed Transitions

B, east, C, -2

C, east, D, -2

	0	
0	0	8
	0	

	0	
-1	0	8
	0	

	0	
-1	3	8
	0	

Assume: $\gamma = 1$,
 $\alpha = 1/2$

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

What's the problem w/ TD Value Learning?

What's the problem w/ TD Value Learning?

Can't turn the estimated value function into a policy!

This is how we did it when we were using value iteration:

$$\pi^*(s) = \arg \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^*(s')]$$

Why can't we do this now?

What's the problem w/ TD Value Learning?

Can't turn the estimated value function into a policy!

This is how we did it when we were using value iteration:

$$\pi^*(s) = \arg \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^*(s')]$$

Why can't we do this now?

Solution: Use TD value learning to estimate Q^* , not V^*

Detour: Q-Value Iteration

- Value iteration: find successive (depth-limited) values
 - Start with $V_0(s) = 0$, which we know is right
 - Given V_k , calculate the depth $k+1$ values for all states:

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V_k(s')]$$

- But Q-values are more useful, so compute them instead
 - Start with $Q_0(s,a) = 0$, which we know is right
 - Given Q_k , calculate the depth $k+1$ q-values for all q-states:

$$Q_{k+1}(s, a) \leftarrow \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma \max_{a'} Q_k(s', a')]$$

Q-Learning

- Q-Learning: sample-based Q-value iteration

$$Q_{k+1}(s, a) \leftarrow \sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma \max_{a'} Q_k(s', a') \right]$$

- Learn $Q(s,a)$ values as you go

- Receive a sample (s,a,s',r)
- Consider your old estimate:
- Consider your new sample estimate: $Q(s, a)$

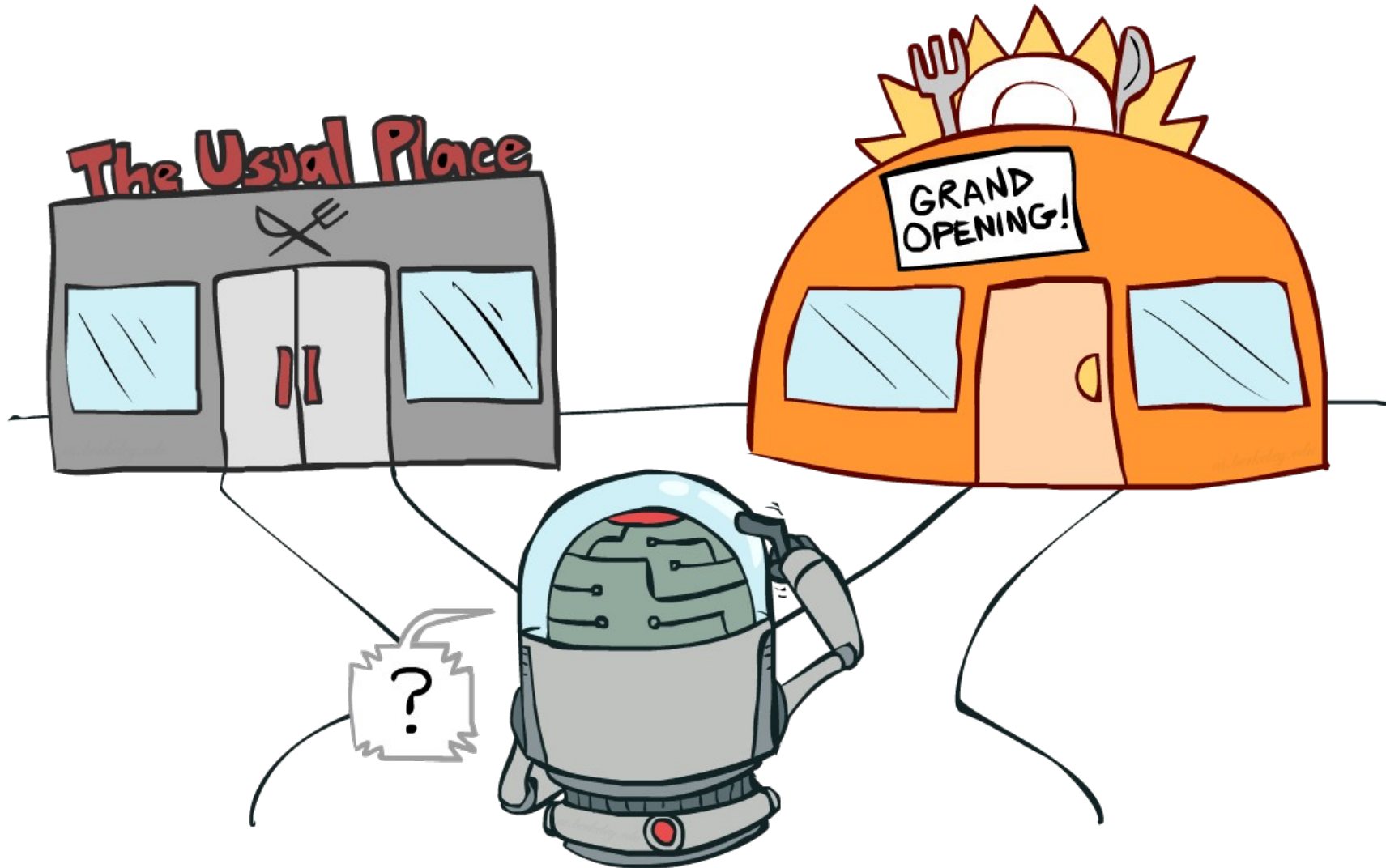
$$sample = R(s, a, s') + \gamma \max_{a'} Q(s', a')$$



- Incorporate the new estimate into a running average:

$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + (\alpha) [sample]$$

Exploration v exploitation



Exploration v exploitation: ϵ -greedy action selection

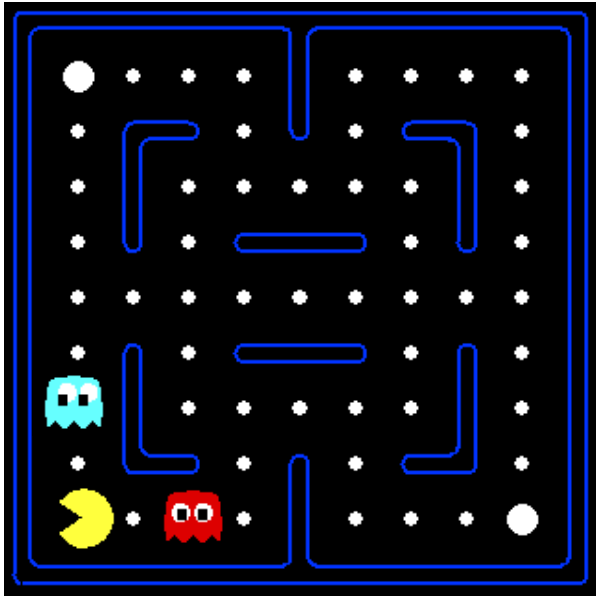
- Several schemes for forcing exploration
 - Simplest: random actions (ϵ -greedy)
 - Every time step, flip a coin
 - With (small) probability ϵ , act randomly
 - With (large) probability $1-\epsilon$, act on current policy
 - Problems with random actions?
 - You do eventually explore the space, but keep thrashing around once learning is done
 - One solution: lower ϵ over time
 - Another solution: exploration functions

Generalizing across states

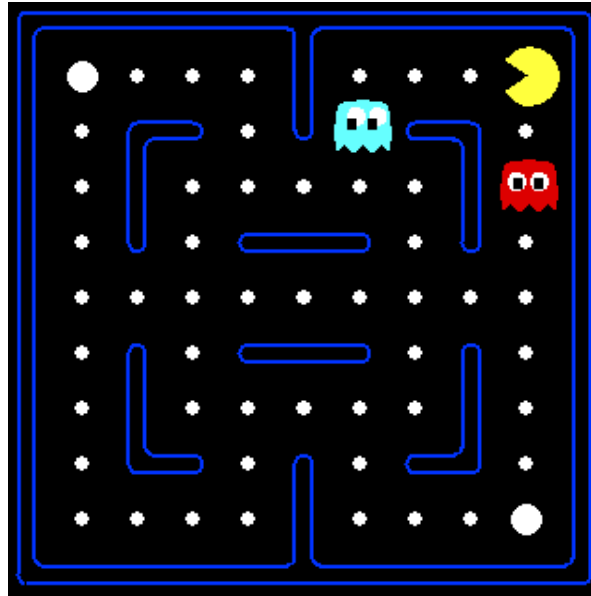
- Basic Q-Learning keeps a table of all q-values
- In realistic situations, we cannot possibly learn about every single state!
 - Too many states to visit them all in training
 - Too many states to hold the q-tables in memory
- Instead, we want to generalize:
 - Learn about some small number of training states from experience
 - Generalize that experience to new, similar situations
 - This is a fundamental idea in machine learning, and we'll see it over and over again

Generalizing across states

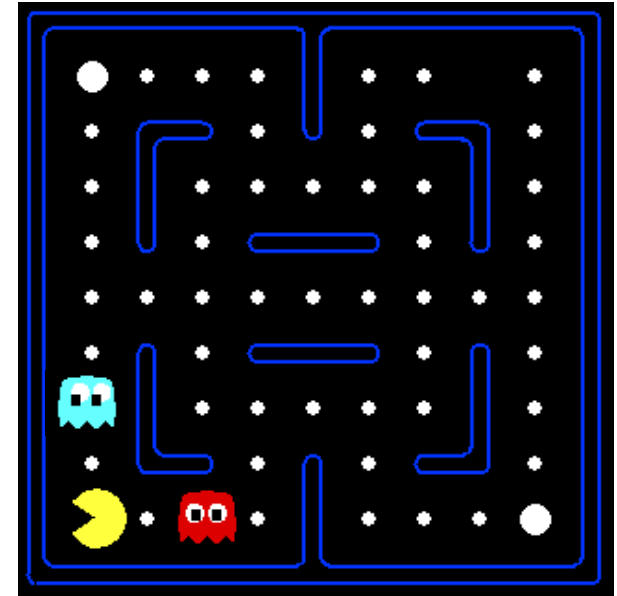
Let's say we discover through experience that this state is bad:



In naïve q-learning, we know nothing about this state:

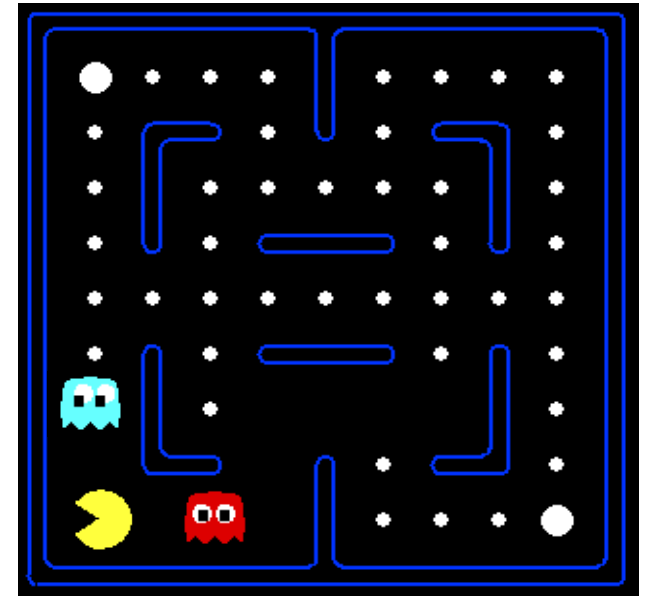


Or even this one!



Feature-based representations

- Solution: describe a state using a vector of features (properties)
 - Features are functions from states to real numbers (often 0/1) that capture important properties of the state
 - Example features:
 - Distance to closest ghost
 - Distance to closest dot
 - Number of ghosts
 - $1 / (\text{dist to dot})^2$
 - Is Pacman in a tunnel? (0/1)
 - etc.
 - Is it the exact state on this slide?
 - Can also describe a q-state (s, a) with features (e.g. action moves closer to food)



Linear value functions

- Using a feature representation, we can write a q function (or value function) for any state using a few weights:

$$V(s) = w_1 f_1(s) + w_2 f_2(s) + \dots + w_n f_n(s)$$

$$Q(s, a) = w_1 f_1(s, a) + w_2 f_2(s, a) + \dots + w_n f_n(s, a)$$

- Advantage: our experience is summed up in a few powerful numbers
- Disadvantage: states may share features but actually be very different in value!

Linear value functions

$$Q(s, a) = w_1 f_1(s, a) + w_2 f_2(s, a) + \dots + w_n f_n(s, a)$$

- Q-learning with linear Q-functions:

$$\text{transition} = (s, a, r, s')$$

$$\text{difference} = \left[r + \gamma \max_{a'} Q(s', a') \right] - Q(s, a)$$

$$Q(s, a) \leftarrow Q(s, a) + \alpha [\text{difference}]$$

Exact Q's

$$w_i \leftarrow w_i + \alpha [\text{difference}] f_i(s, a)$$

Approximate Q's

- Intuitive interpretation:

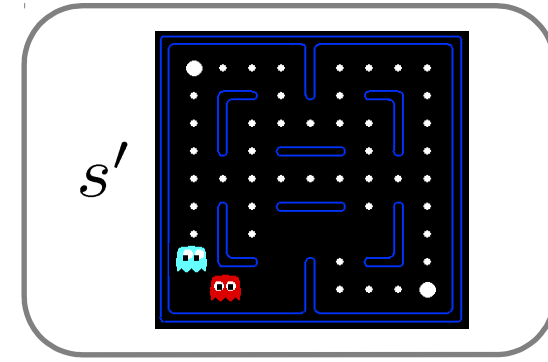
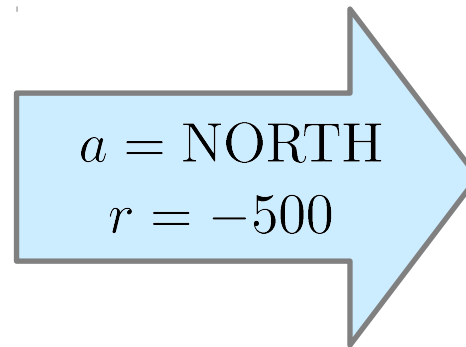
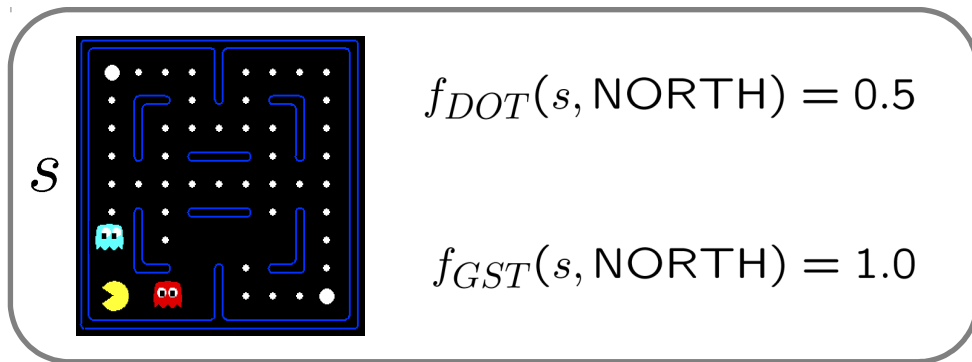
- Adjust weights of active features

- E.g., if something unexpectedly bad happens, blame the features that were on: disprefer all states with that state's features

- Formal justification: online least squares

Example: Q-Pacman

$$Q(s, a) = 4.0 f_{DOT}(s, a) - 1.0 f_{GST}(s, a)$$



$$Q(s, \text{NORTH}) = +1$$

$$r + \gamma \max_{a'} Q(s', a') = -500 + 0$$

difference = -501

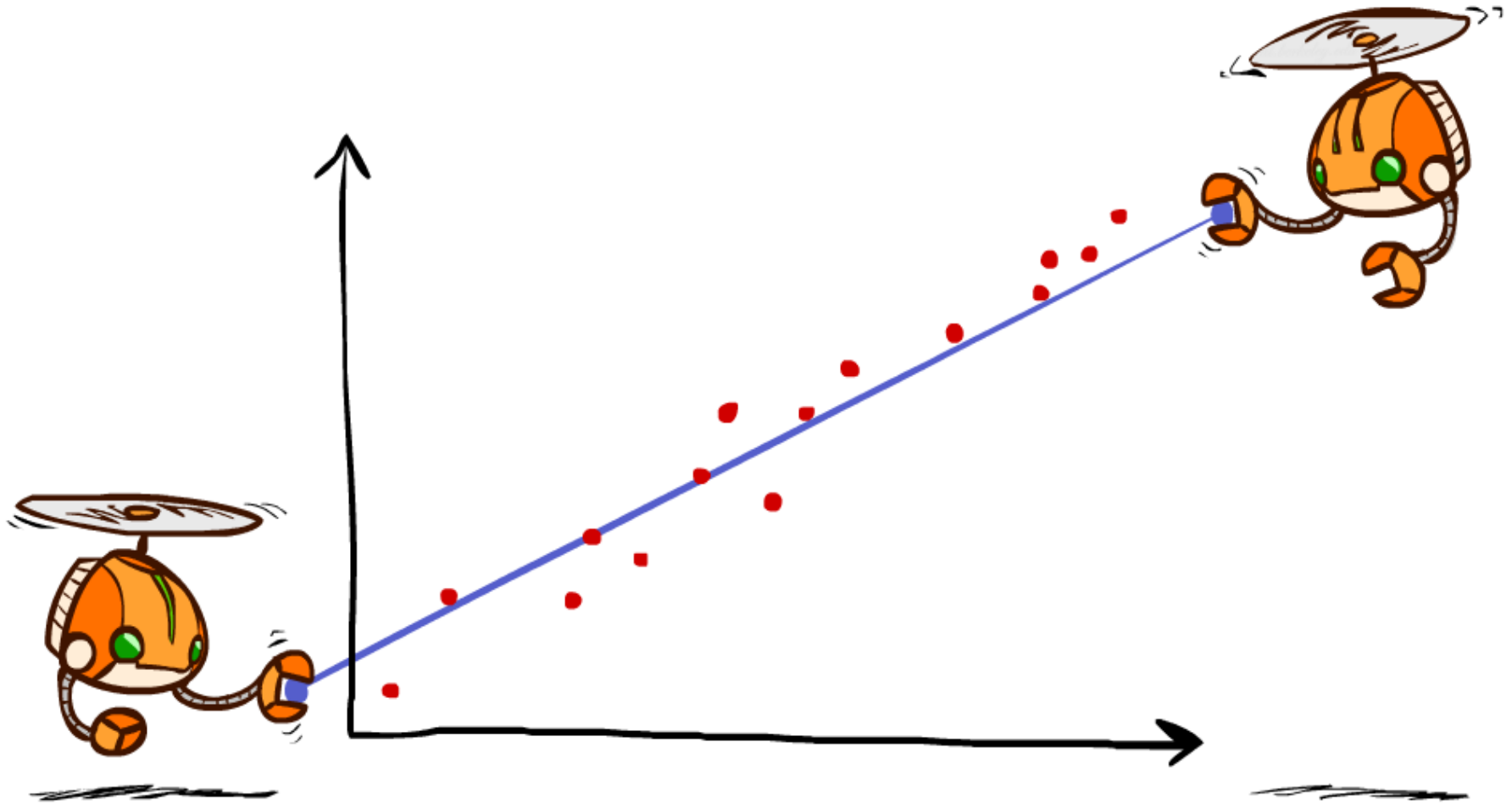


$$w_{DOT} \leftarrow 4.0 + \alpha [-501] 0.5$$

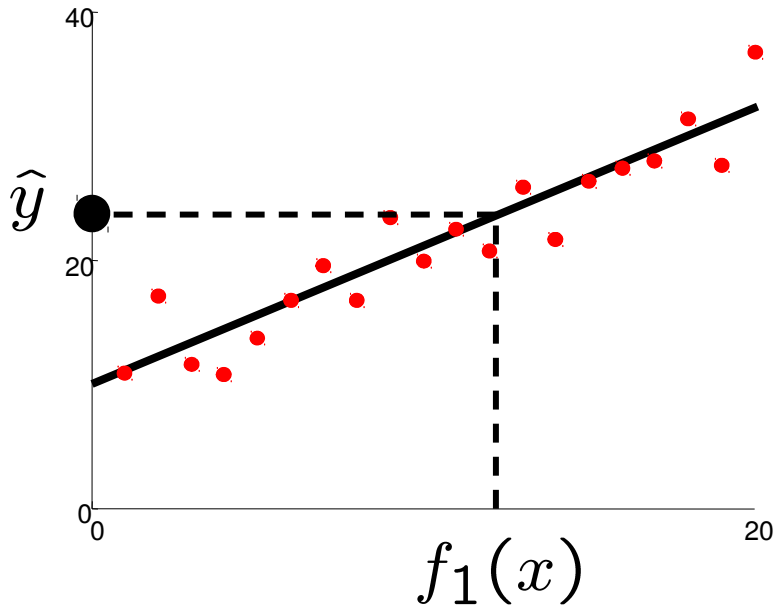
$$w_{GST} \leftarrow -1.0 + \alpha [-501] 1.0$$

$$Q(s, a) = 3.0 f_{DOT}(s, a) - 3.0 f_{GST}(s, a)$$

Q-Learning and Least Squares

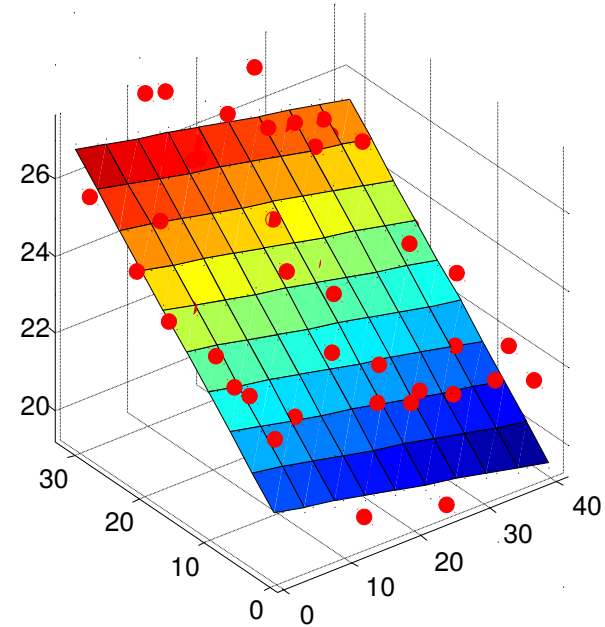


Q-Learning and Least Squares



Prediction:

$$\hat{y} = w_0 + w_1 f_1(x)$$

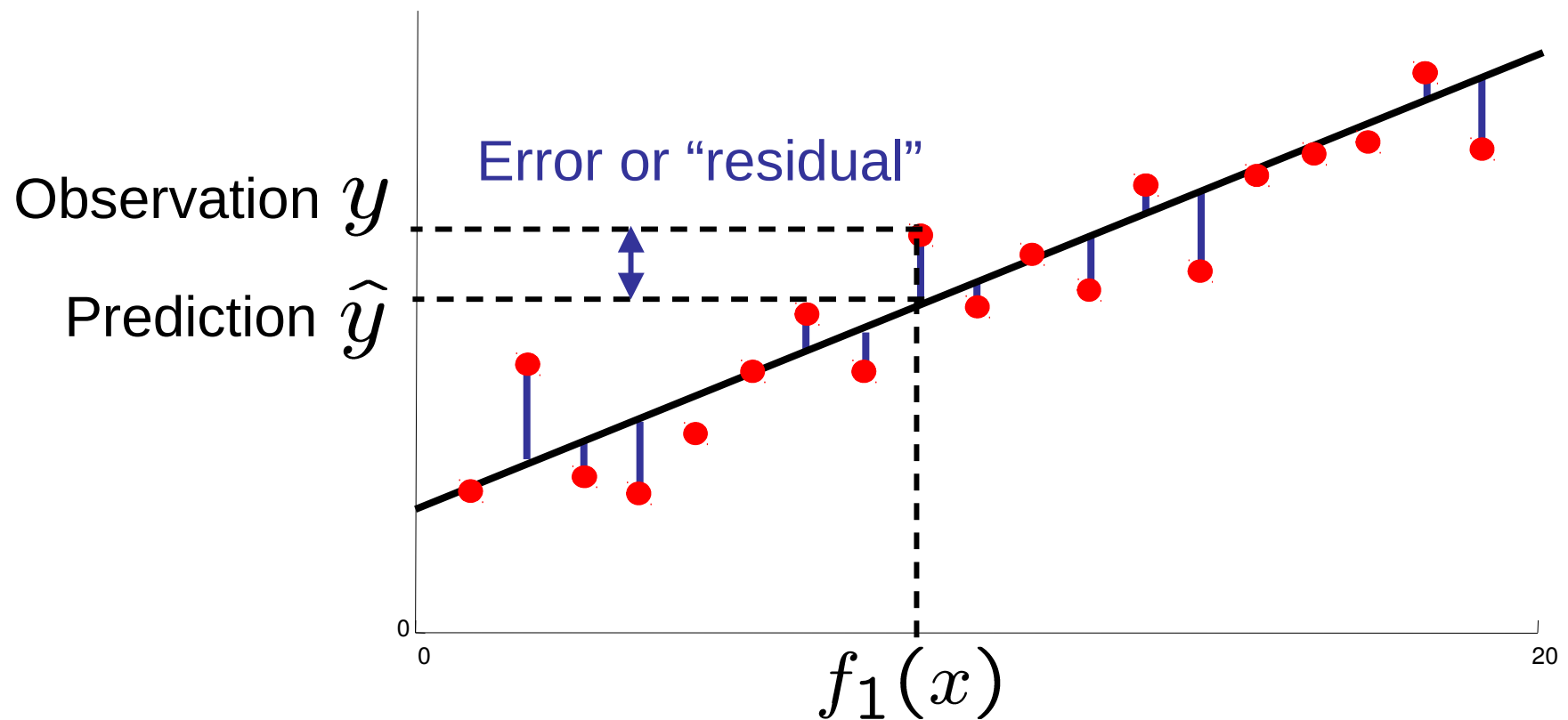


Prediction:

$$\hat{y}_i = w_0 + w_1 f_1(x) + w_2 f_2(x)$$

Optimization: Least Squares

$$\text{total error} = \sum_i (y_i - \hat{y}_i)^2 = \sum_i \left(y_i - \sum_k w_k f_k(x_i) \right)^2$$



Minimizing Error

Imagine we had only one point x , with features $f(x)$, target value y , and weights w :

$$\begin{aligned} \text{error}(w) &= \frac{1}{2} \left(y - \sum_k w_k f_k(x) \right)^2 \\ \frac{\partial \text{error}(w)}{\partial w_m} &= - \left(y - \sum_k w_k f_k(x) \right) f_m(x) \\ w_m &\leftarrow w_m + \alpha \left(y - \sum_k w_k f_k(x) \right) f_m(x) \end{aligned} \quad \left. \vphantom{\begin{aligned} \text{error}(w) \\ \frac{\partial \text{error}(w)}{\partial w_m} \\ w_m \end{aligned}} \right\} \text{Gradient descent}$$

Approximate q update

explained: $w_m \leftarrow w_m + \alpha \left[\underbrace{r + \gamma \max_a Q(s', a')}_{\text{“target”}} - \underbrace{Q(s, a)}_{\text{“prediction”}} \right] f_m(s, a)$