Welcome back! You may not have noticed, but before the break, we crossed a major threshold for intelligence: utility, or evaluation.

To nail this home, let's think of scale-free intelligence theory.

Allowing ourselves to ask: what rules apply to intelligence at any size?

can a bacterium be intelligent?
colony?

can an ant be intelligent?
colony?

can a tree be intelligent
& ?
forest?

can an animal be intelligent? → flock?

...

can a human be intelligent? ecosystem?

City? country? planet?

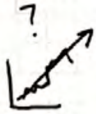
systems

sensation
① can it sense?

actuation
② can it act?

control
③ can it decide?
if... then

detect
④ can it model?



learning!

pure mechanics

bear trap ... thermostat — your robots.

utility
5 can it evaluate? Good vs. bad actions.

networks
Argument: a plant is intelligent by this def.

single
Why? plants demonstrate ~~[illegible]~~ ~~[illegible]~~.

sunflowers

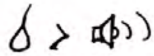
sense ✓
act ✓
control ✓

turn towards sun.

model? evaluate?

grow towards sound.

no moisture



grow towards water

mimosa

close up.

habituation — stimulus → response

classical cond. — stimulus → stimulus

operant — stimulus → reward

utility

A Forest!

- share resources
- communicates info (warnings)
- recognize kin.
- evaluate others + redistribute resources

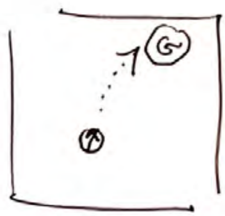
1-4 absolutely ✓✓   5... depends on further research.

our model utility agent: Q-table.

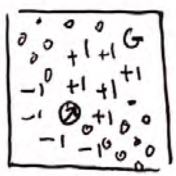
state	action	reward
x, y ...	left, right ...	+1, -1, +100 ...

State

ϕ	x	y	sensor,	action	reward
15°	5	10	30cm	fwd	+1

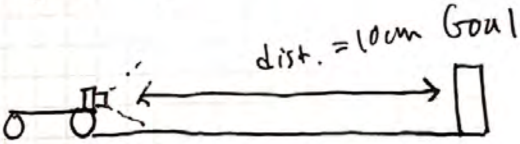


reward(state, action)   1 state is goal.



produce a field like this.

easier in 1D:



★ action:
fwd, bwd, wait

every step is
argmax(reward(state, action))

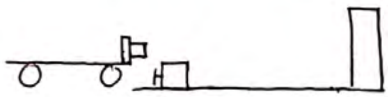
★ design a reward fn. that will bring you to goal.

state	action = fwd	bwd	wait
10	+1	-1	0
9	+1	-1	0
8	+1	-1	0
7	+1	-1	0
6	+1	-1	0
5	+1	-1	0
4	+1	-1	0
3	+1	-1	0
2	+1	-1	0
1	+1	-1	0
0	0	-1	0

conflict

Now: an obstacle pops up (out of fov)

rule: stays up until released.



☆ How do you solve this problem?

- add randomness
- add state
- ?

Now, what if the obstacle is random? spatially + temporally

↳ you need an update function.

Q-table:

state	bwd	fwd	wait
10	-1	+1	-1
9	-1	+1	-1
8	-1	+1	-1
7	-1	+1	-1
6	-1	+1	-1
5	-1	+1	-1
4	-1	+1	-1
3	-1	+1	-1
2	-1	+1	-1
1	-1	+1	-1
0	-1	-1	100

init to perfect world ⋮ or random.

↑ ↑ enforce any policy you want.

$0 \leq \gamma \leq 1$

① Act

every action phase: $\gamma \rightarrow$ act randomly

$1-\gamma \rightarrow$ take max reward.

② sense

update state.

③ evaluate determine reward(state)

④ Learn

update Q-table.

q[s][a] = ~~Reward(state, ...)~~ Reward(s')

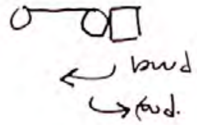
↑ good start... but unstable.

☆ remember bayes... how do we incorporate old info?

$q[s][a] = (1-\alpha)\, q[s][a] + \alpha\, \text{reward}(s')$

learning rate.

problem: not enough future.



$\alpha\left(r(s') + d \max_a Q(s_{t+1}, a)\right)$

best future path.

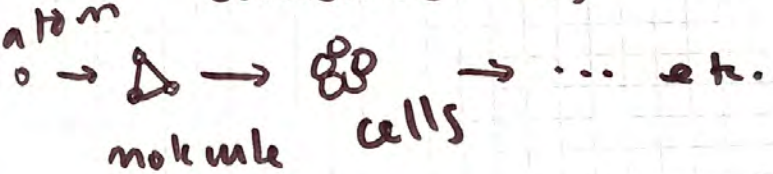
$$\alpha\left(r(s') + \text{discount}\left(\max_a Q(s_{t+1}, \text{action})\right)\right)$$

now we can chain a series of actions + consequences together.

Q: is a forest this smart?

COGS 300 Distribution 03 Nov 18/25

warm up: Draw examples of different scales with the same structure. Eg.



bacterium → colonies
?

sensing ✓
acting ✓ → ? modeling ✓
control ✓

ant? ? → colony

tree? → forest?

evaluation (utility)

2

sensation actuation control

1 2 3

if... then

switch light

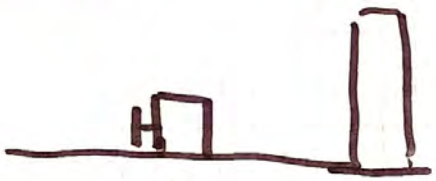
4 detection (model)

thermostat

robots.

5 evaluation of unknown

state	action Fwd	Bwd	wait
10	+1	−1	0
9	+2	−1	0
8	+3	−1	0
7	+4	⋮	⋮
6	+5		
5	+6		
4	+7		
3	+8		
2	+9		
1	+10		
0	−1	−1	0



★ goal

How do you construct a reward + action...

obstacle

robot fwd + bwd for release

explore vs exploit

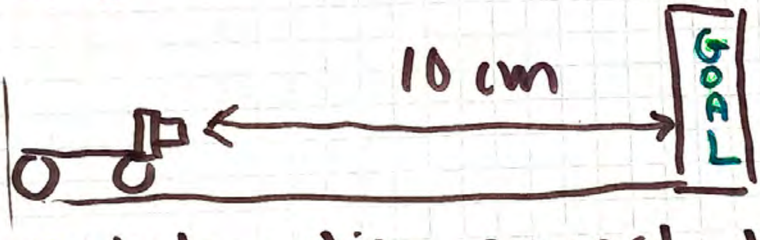
r   1−r

state		action		
x	y	fwd	left	bak
3	5	+1	0	-1
		⋮		

★ what fn could produce a reward schema that promotes movement towards a goal, and punishes opposite?

$$\text{action} = \max(\text{rewards} \ldots)$$

$$\text{reward} = \text{dist}(r, g)$$



★ state, action reward table

☆ How do you update the reward table?

$$q[s][a] = \overline{\text{reward}}(s)$$

$$q[s][a] = (1-\alpha)\, q[s][a] + \alpha \,\text{reward}(s')$$

↓

$$\alpha\Big(r(s') + d\,\big(\max_a Q(s_{t+1}, \text{action}$$

reinforcement learning