So far, we've talked about systems that "just" react. Even with systems that have memory, we can't say they're learning because they never change their policies. Even Bayesian networks don't really "learn" as much as process sensory data with w.r.t. conditions.

That's because we need some way of assigning value to an action. Was this the right or wrong choice? Does this work or not work? This is called UTILITY:

Utility, as a concept, comes from economics. Often, it's talked about abstractly, ie. it has no unit. But concretely it's often operationalized as price.

you have chocolate bar — I want it → we negotiate. → Value = \$price = \$1.00

So we can say the chocolate bar is worth \$1.00 and must therefore give me \$1.00 of utility.

I want to point out a divergence between this model and reality.

1. Humans can predict, recall, learn. $1 ≠ $1.00 if your $1/$1000 vs. $1/1000000.
2. Most things don't have $ value. "Money can't buy you love" is literally true.
3. Even within strict preference models, price shifts. E.g. Kahneman.

But utility is (potentially) computable. So, it is the basis of our modelling.

Blacksmith in Dist. Kingdom.

<u>Purse</u> inv. prod. raw. Labour theory of value
10 coins

prod. = +1 to value and inventory.
1 unit of raw = $1.

King Bezos. mon. has set up a distribution network.
Each day, a cart will stop. You can:

1) ~~grace~~ sell what you've make.
2) Buy a raw. ~~product~~. material.
3) Wait and do nothing
4) Spend the day producing something.

You tell the cart man every day for tomorrow.

Buy	sell	wait	produce.
-1	+2	∅	∅

obviously:

sell > wait = produce > buy = prefs

If your policy is strictly argmax (prefs), you would never make or buy anything!

Instead:

coin?	prod?	raw?	action.
Y	N	N	buy

☆ fill out the rest.

Let's say the cart is now true random.
You probably need a more complex policy. e.g.

☆ coin < 2 coin > 10 ... etc.

and you need to track whether it "worked"!!

Now let's say you need to schedule ppl.
produce now is -10 and needs advance commitment. But Sell is now +40 because of efficiency.
cart still random.

☆ schedule policy.

Now let's say that the cards "look" random, but there actually is a pattern. The pattern is mostly consistent but not perfectly. How do you learn the best policy?

Introduce Q-Learning, a type of reinforcement learning.

$$Q : S \times A \to \mathbb{R}$$

S → state

A → action buy, sell, wait, produce.

$\mathbb{R}$ → reward

coin inv. raw inv. prod.

The ideas is to make a big table of time reward so:

$$\frac{c}{10} \quad \frac{raw}{0} \quad \frac{prod}{1} \times \text{sell} \to +\$40$$

May be true in a moment, but might be long-run costly if faced with a chance to buy raw. Why?

$$\frac{c}{9} \quad \frac{r}{0} \quad \frac{prod}{0} \quad \text{sell} \to \$0$$

So abstract utility s.t.

$$\frac{c}{10} \quad \frac{r}{0} \quad \frac{prod}{1} \times \text{sell} < \frac{c}{10} \quad \frac{r}{0} \quad \frac{prod}{1} \times \text{buy}$$

★ Time table so that ~~that~~ a 5-day week can be "planned" using <u>just</u> utility values.

This is <u>hard</u>. Essentially impossible to design by hand w/o very restricted circumstances.

Translate to the robot. How could ★ you reward + punish your robot?

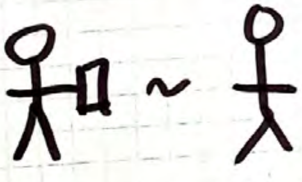
COGS 300 Distribution 02 Nov 6/25 ①

Warm up: Draw networks with "islands" ... what does it take to make an island into a... "penninsula"?

Meet the profs 2nite @ 5pm Koerner's.

utility = value

u(action) = value



$1.00

 = $1.00 = ~20

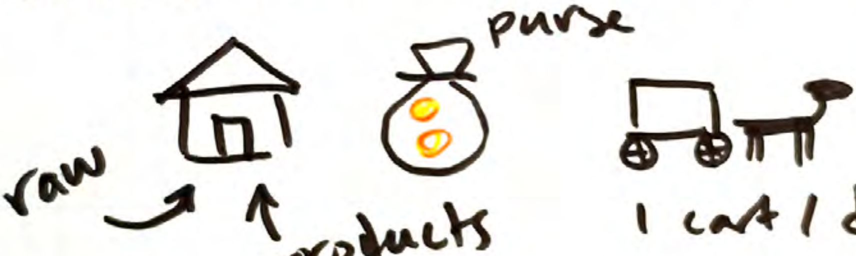
equal pref.

1. $1/1000 $1/1 000 000

2. "money can't buy you love."

3. Kahneman

Blacksmith   iron → products



-1	+2	0	0
buy	sell	wait	produce

actions

sell > wait = produce > buy

✗ argmax(actions) → sell

coin?	prod?	raw?	→ action ✓
Y	N	N	buy
Y	Y	N	→ sell
Y	N	Y	→ produce
N	Y	N	→ sell
N	N	N	→ wait

policy

what if cart is random?

$c \leq 2$ vs. $c \geq 10$

make a new policy table.

$c<2$ → sell
$p? = y$

$c > 10$
$p > 20$ → prod.
-5^{th} prod

for the next week, you hire labourers. 1 iron → 10 prod.

-5 because you're paying to produce but you get +10 products.

			c	raw	prod
sell	+10	X	10	5	20
buy	-1				
produce	-5				
wait	0		100	0	0

↳ what else can you track if carts aren't actually random but just appear so?

day	c	raw	prod.	utility	action
7th Nov	10	5	25	$10	sell

⋮

Reinforcement Learning

Q-learning

8

$Q : S \times A \rightarrow \mathbb{R}$

State

coin	raw	prod.	actions	→ Reward
1	0	0	sell	
0	1	0	buy	
0	0	1		

buy → -1 Reward

buy, produce, sell → +10

1. State tracking?
2. Actions?
3. Reward?