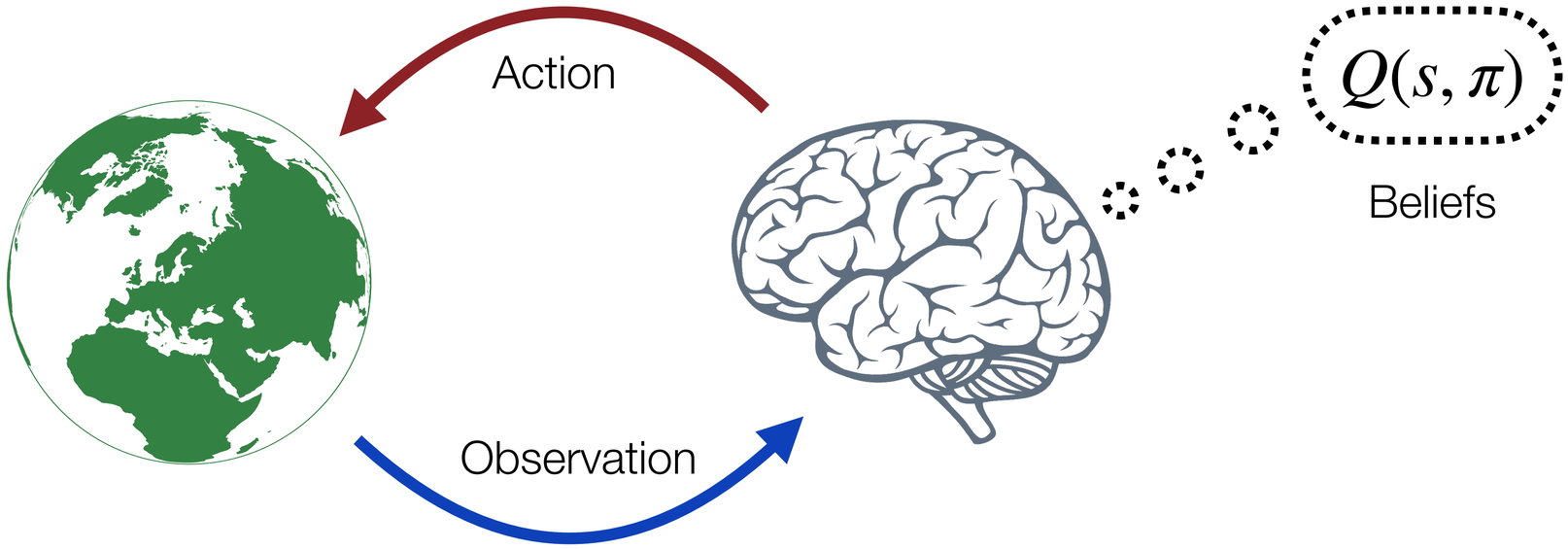
Tutorial B: Part I

Hands-on Active Inference with **pymdp**



Conor Heins and Daphne Demekas

Tutorial B: Part I

Hands-on Active Inference with **`pymdp`**

- Python 

- NumPy 


- Google Colab 

Brief introductions

Conor Heins

International Max Planck Research School for Organismal Biology

NESTED MINDS NETWORK

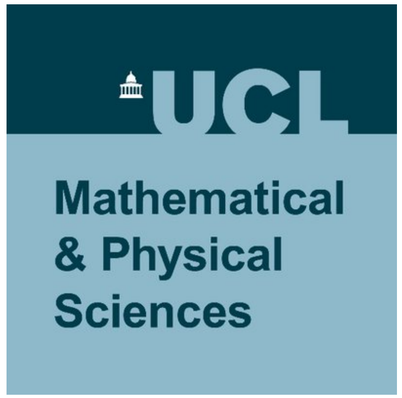
MAX PLANCK INSTITUTE
OF ANIMAL BEHAVIOR

wellcome centre human neuroimaging

UCL

Brief introductions

Daphne Demekas

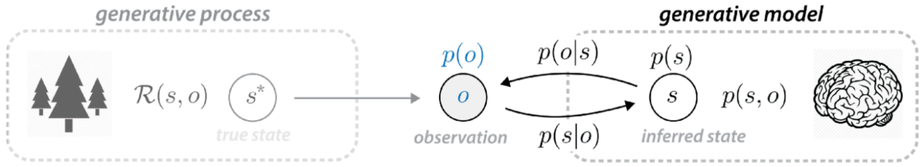


References for active inference material

- *A Step-by-Step Tutorial on Active Inference and its Application to Empirical Data.* **R. Smith, K. Friston, C. Whyte**
- *The free energy principle for action and perception: A mathematical review.* **C. Buckley, C. Sub Kim, S. McGregor, A. Seth**
- *A tutorial on the free energy framework for modelling perception and learning.* **R. Bogacz**

Oleg Solopchuk Sep 4, 2018 · 18 min read

Tutorial on Active Inference

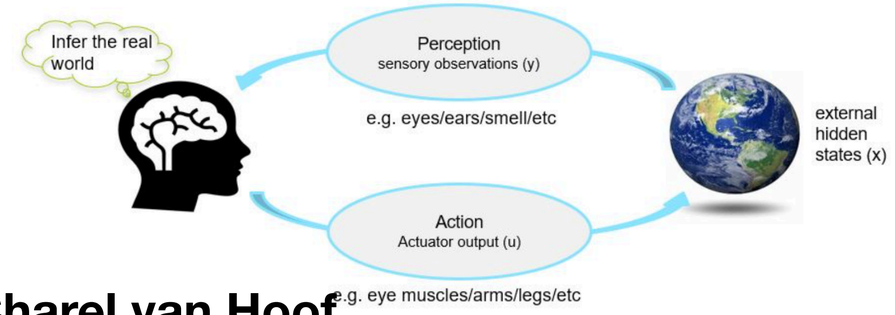


$$p(a|b) = \frac{p(b|a)\,p(a)}{p(b)}$$

Oleg Solopchuk

Learn by example: Active Inference in the brain -1

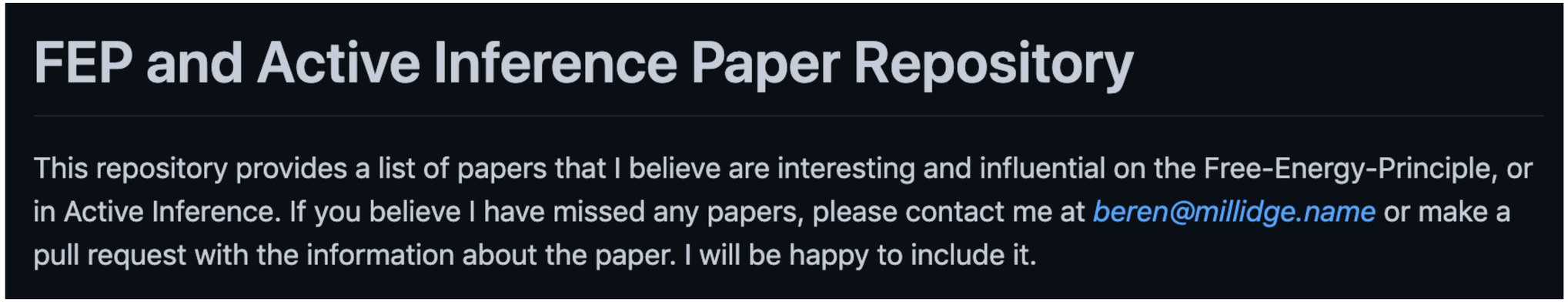
Python notebook using data from no data sources · 6,370 views · 5mo ago · matplotlib, numpy, seaborn, +5 more



Charel van Hoof

References for active inference material

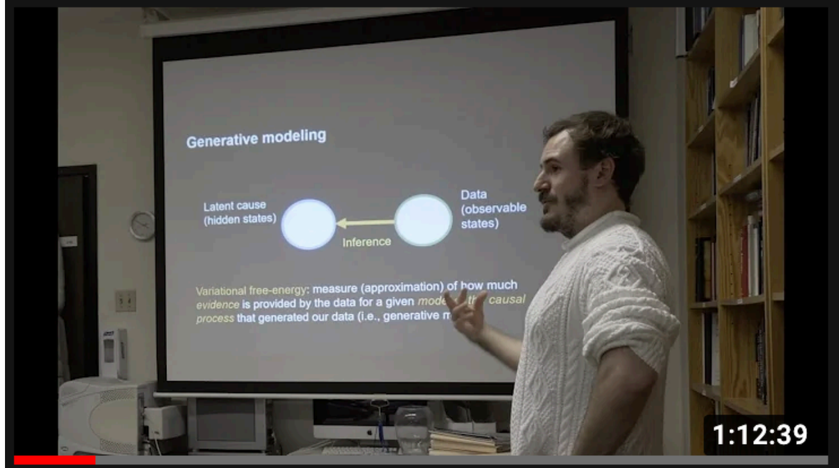
Beren Millidge's Paper Repository



https://github.com/BerenMillidge/FEP_Active_Inference_Papers

References for active inference material

https://www.youtube.com/watch?v=WzFQzFZiwzk&t=10s



Maxwell Ramstead — A tutorial on active inference

4.9K views • 1 year ago

McGill Transcultural Psychiatry

"A **tutorial** on **active inference**: from the predictive brain to socio-cultural regimes of expectations" **Maxwell Ramstead** Culture and ...

References for active inference material

Livestream(s) of R. Smith and C. Whyte discussing their tutorial paper

1. https://www.youtube.com/watch?v=H5AolqFl2Nw&t=4670s
2. https://www.youtube.com/watch?v=bDJbEofbvyk
3. https://www.youtube.com/watch?v=vacR2gYgOxk
4. https://www.youtube.com/watch?v=QRGaGmT-VFM

MATLAB vs. Python


```
function [MDP] = spm_MDP_VB_X(MDP,OPTIONS)
% active inference and learning using variational message passing
```

Different levels of abstraction with pymdp

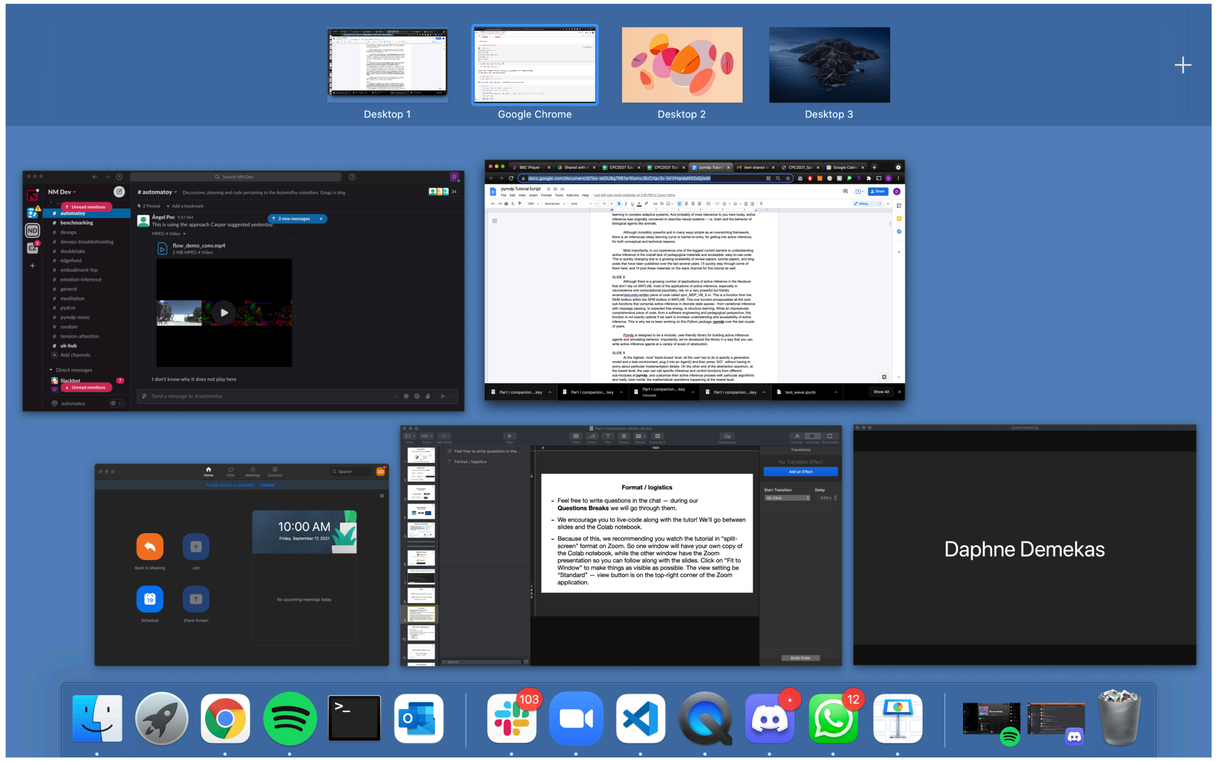
```
from pymdp import control # import control library for expected free energy calculations
from pymdp.algos import run_fpi # important a particular inference algorithm (here, fixed-point iteration)

Q_states = run_fpi(A, observation, num_obs, num_states, prior, num_iter = 10)

for idx, policy in enumerate(policies):

    expected_states = control.get_expected_states(Q_states, B, policy)
    expected_obs = control.get_expected_obs(expected_states, A)

    expected_free_energy[idx] += control.calc_expected_utility(expected_obs, C)

    expected_free_energy[idx] += control.calc_states_info_gain(A, Q_states)

Q_policies = softmax(expected_free_energy * gamma)
```

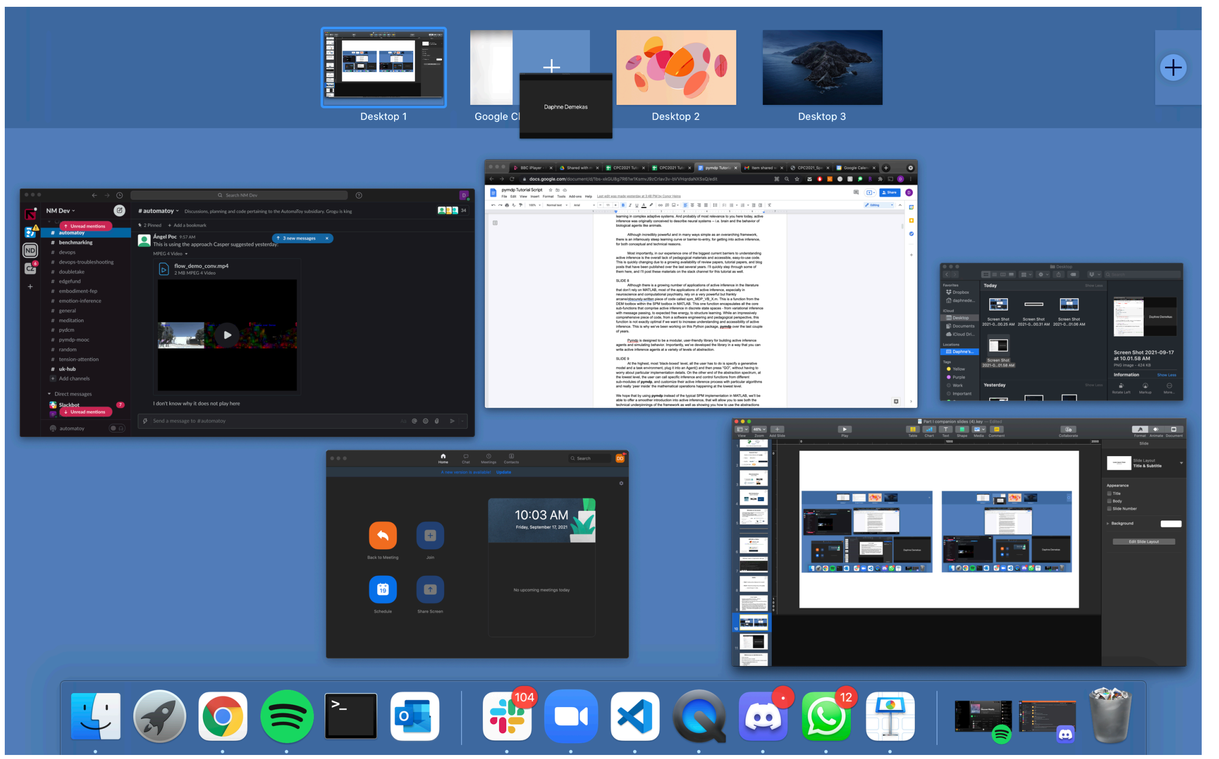
Outline

- **Part I**: Coding active inference from scratch
- **Part II**: Abstracting things away with `pymdp`
    - using the `Agent()` class

Format / logistics

- Feel free to write questions in the chat — during our **Questions Breaks** we will go through them.
- We encourage you to live-code along with the tutor! We'll go between slides and the Colab notebook.
- Because of this, we recommending you watch the tutorial in "split-screen" format on Zoom.
- Let's open the notebook, save our own copy, and run the first cell


Desktop 1
Google Chrome
Desktop 2
Desktop 3
Format / logistics
Feel free to write questions in the chat — during our Questions Breaks we will go through them.
We encourage you to live-code along with the tutor! We'll go between slides and the Colab notebook.
Because of this, we recommending you watch the tutorial in "split-screen" format on Zoom. So one window will have your own copy of the Colab notebook, while the other window have the Zoom presentation so you can follow along with the slides. Click on "Fit to Window" to make things as visible as possible. The view setting be "Standard" — view button is on the top-right corner of the Zoom application.
Daphne Demekas


Desktop 1
Google Ch
Daphne Demekas
Desktop 2
Desktop 3

CPC2021 - Active Inference Liv

colab.research.google.c...

CPC2021 - Active Inference Live Coding, Part II - Filled Out

File Edit View Insert Runtime Tools Help Last saved at 9:27 AM

Comment Share

+ Code + Text

Connect Editing

The $\mathbf{D}$ vectors

```
[ ] D = utils.obj_array(num_factors)
```

```
""" Fill this out """

D_context = np.array([0.5,0.5])

D[0] = D_context

D_choice = np.zeros(len(choice_names))

D_choice[0] = 1.0

D[1] = D_choice
```

```
[ ] print(f'Beliefs about which arm is better: {D[0]}')
    print(f'Beliefs about starting location: {D[1]}')

    Beliefs about which arm is better: [0.5 0.5]
    Beliefs about starting location: [1. 0. 0. 0.]
```

Now let's take advantage of the `Agent` class in `pymdp` to wrap this all into an Agent instance that we can use to do active inference in a few lines.

```
[ ] from pymdp.agent import Agent

    my_agent = Agent(A = A, B = B, C = C, D = D)
```

Define a class for the 2-armed bandit environment (AKA the *generative process*)

```
[ ] class TwoArmedBandit(object):

      def __init__(self, context = None, p_hint = 1.0, p_reward = 0.8):

        self.context_names = ["Left-Better", "Right-Better"]

        if context == None:
          self.context = self.context_names[utils.sample(np.array([0.5, 0.5]))] # randomly sample which bandit
        else:
          self.context = context

        self.p_hint = p_hint
        self.p_reward = p_reward
```

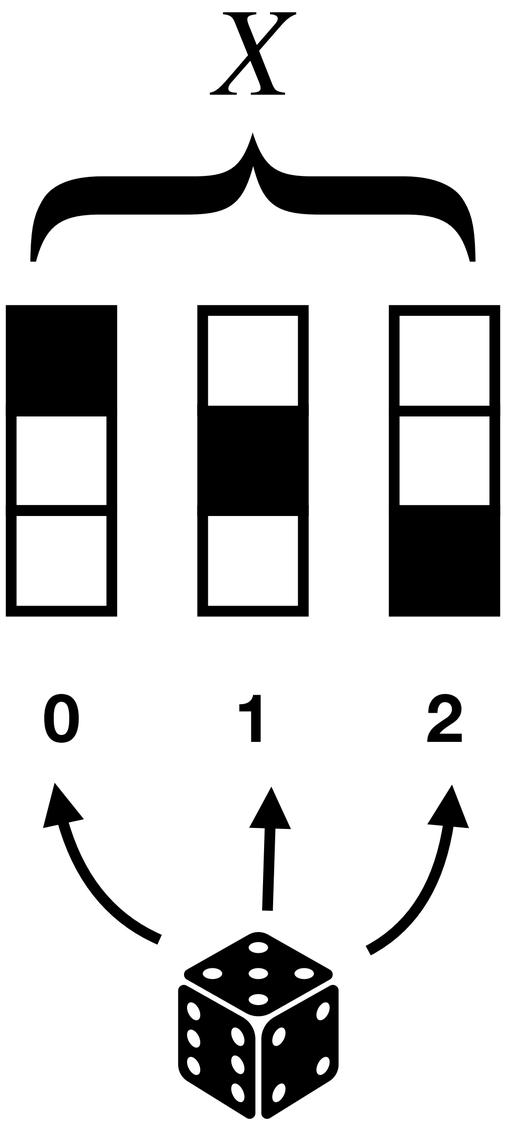
Daphne Demekas

Before we go on, big thank you to…

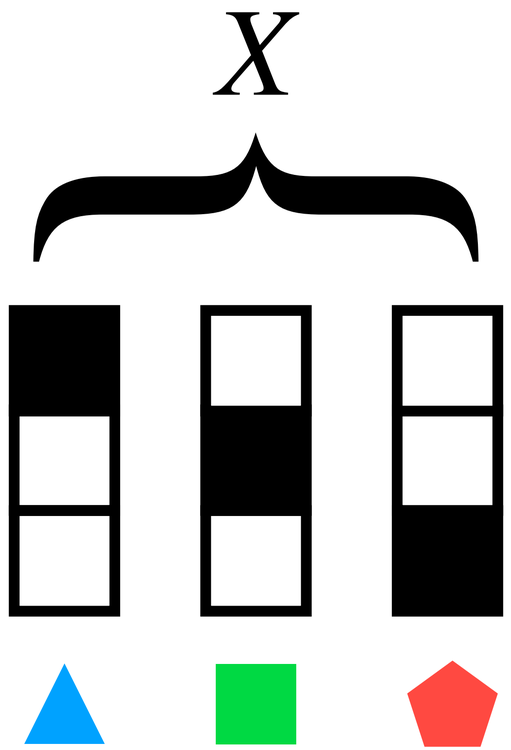
- Daphne Demekas
- Alec Tschantz (co-creator of `pymdp`)
- Beren Millidge (contributor to `pymdp`, also author of tutorial material)
- Support from Nested Minds Solutions, Ltd.
  - Alice Ou, Riddhi Jain, Uri Cohen, Mahault Albaraccin, Dimitrije Markovic, Alex Kiefer, Maxwell Ramstead

Representing Categorical distributions in `NumPy`

The Categorical distribution



The Categorical distribution



The Categorical distribution

$$P(X) = \left\{ P(\ \blacksquare\ ),\ P(\ \blacksquare\ ),\ P(\ \blacksquare\ ) \right\}$$

The Categorical distribution

$$P(X) = \left\{ P(X=0) \, , \; P(X=1) \, , \; P(X=2) \right\}$$

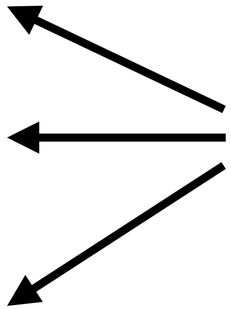
The Categorical distribution

$$P(X) = \begin{bmatrix} 0.60 \\ 0.15 \\ 0.25 \end{bmatrix}$$

one probability per outcome

The Categorical distribution

$$P(X) = \begin{bmatrix} 0.60 \\ 0.15 \\ 0.25 \end{bmatrix}$$



one probability per outcome

```
prob_x = np.array( [0.6, 0.15, 0.25] )
```

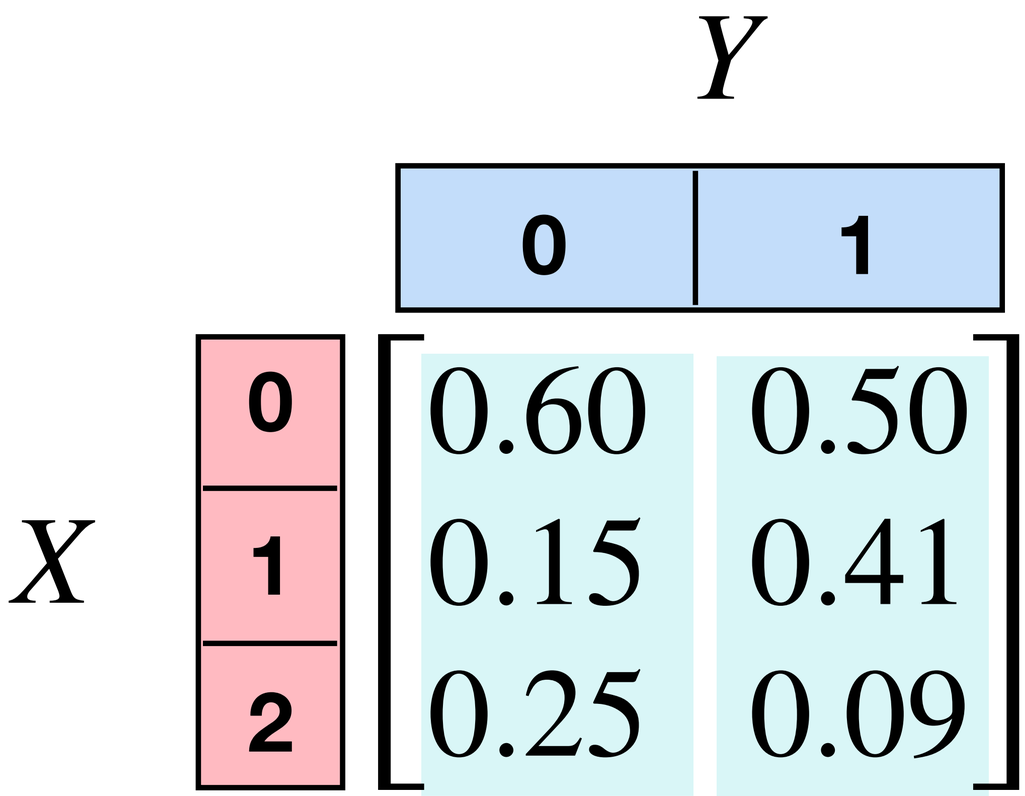
Let’s try that out in code…

!!!! Reminder about MATLAB vs. Python indexing !!!!

`array[1] vs. array[0]`

Conditional distributions as matrices

$$P(X \mid Y) = \begin{bmatrix} 0.60 & 0.50 \\ 0.15 & 0.41 \\ 0.25 & 0.09 \end{bmatrix}$$



$$P(X \mid Y = 0)$$

$$
\begin{array}{c|c|cc}
 & & Y & \\
 & & \mathbf{0} & \mathbf{1} \\
\hline
X & \mathbf{0} & 0.60 & 0.50 \\
 & \mathbf{1} & 0.15 & 0.41 \\
 & \mathbf{2} & 0.25 & 0.09 \\
\hline
 & & +1.00 & +1.00
\end{array}
$$

$$\sum_{X} P(X \mid Y = i) = 1.0$$

$$\mathbf{A} = \begin{bmatrix} 0.60 & 0.50 \\ 0.15 & 0.41 \\ 0.25 & 0.09 \end{bmatrix}$$

$$\mathbf{A}[\texttt{i,j}] = P(X = i \mid Y = j)$$

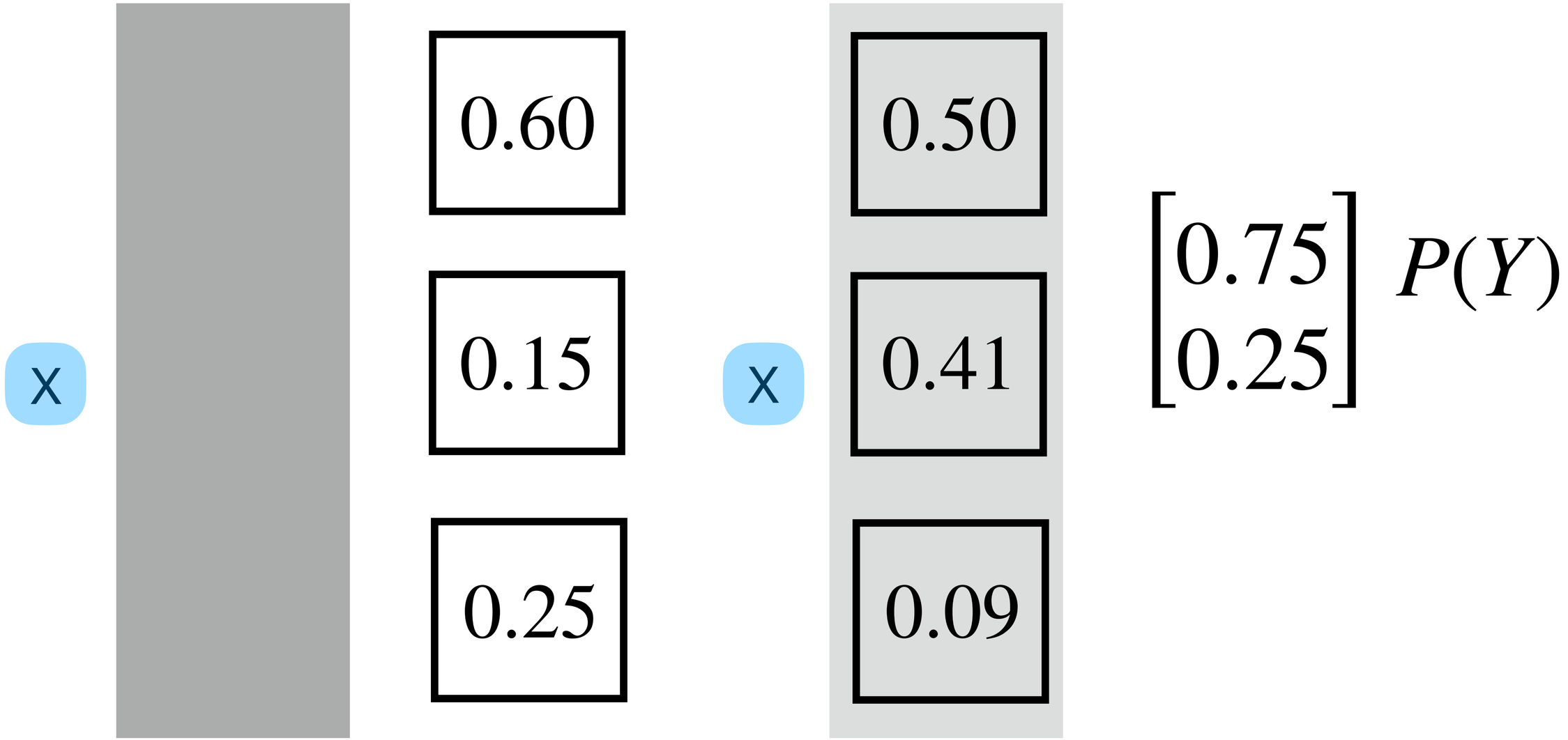
Back to Colab...

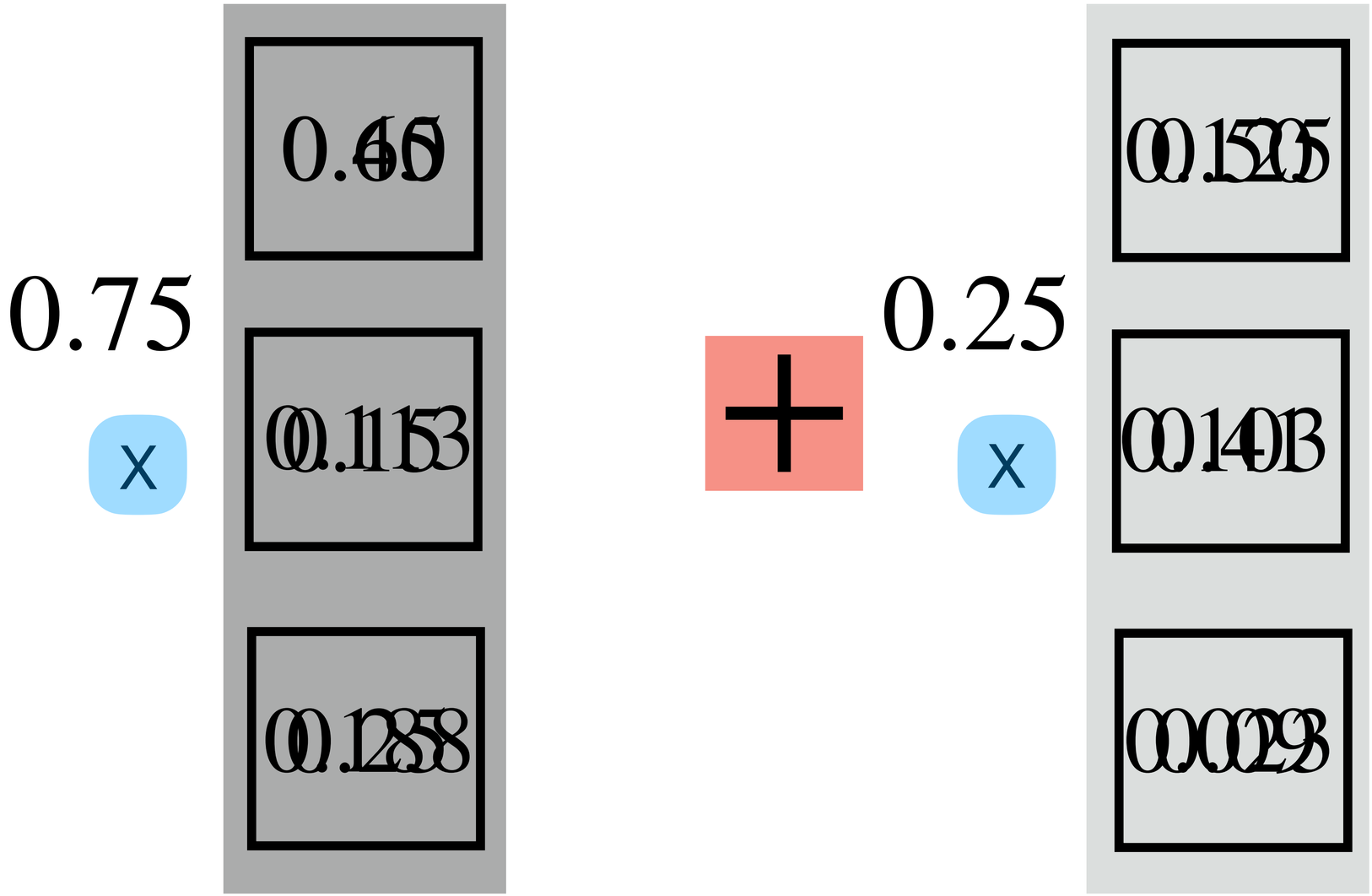
Conditional expectation

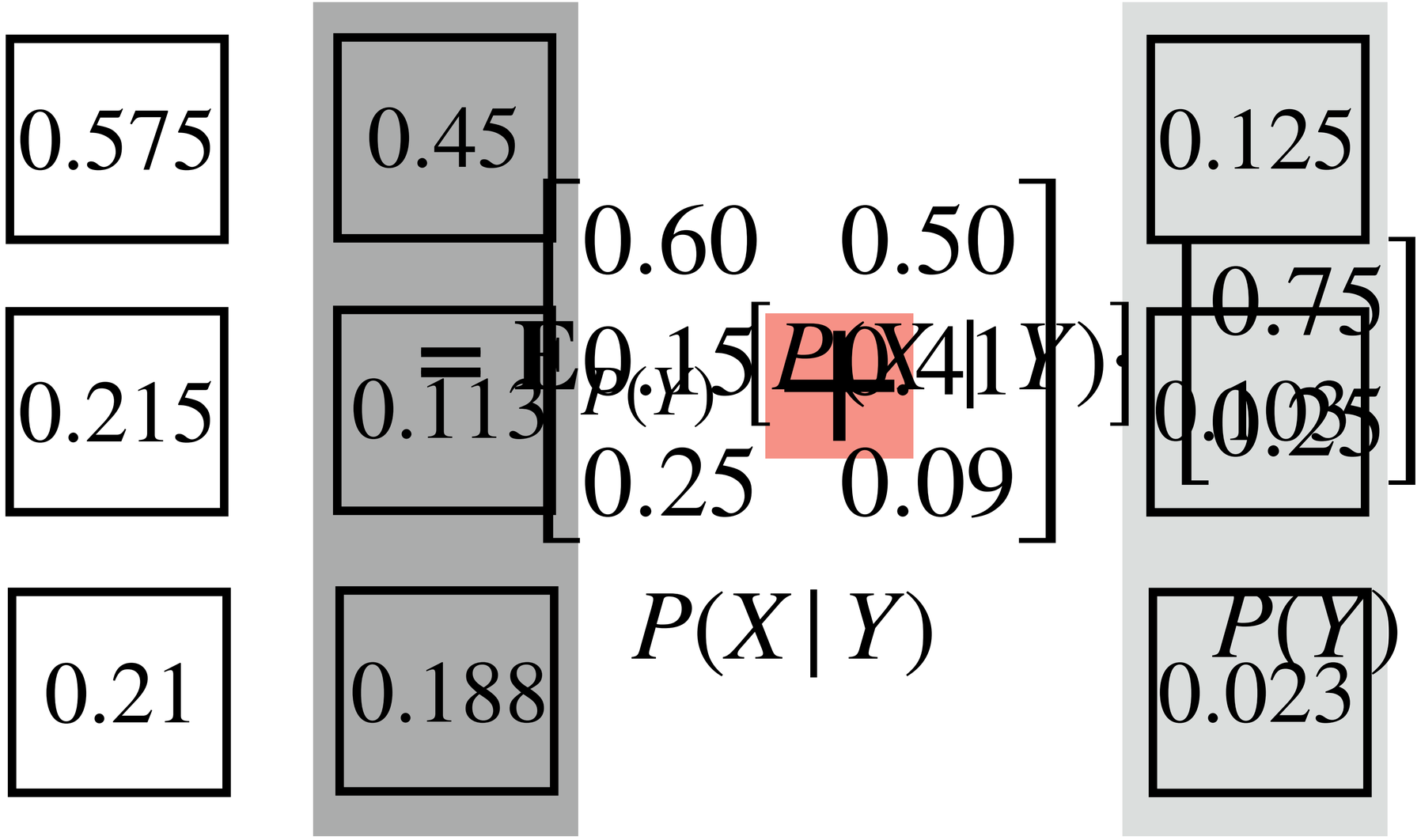
$$\mathbf{E}_{P(Y)}[P(X \mid Y)] = \sum_i P(X \mid Y = i) P(Y = i)$$

$$P(X \mid Y)$$

$P(X \mid Y = 0)$ $P(X \mid Y = 1)$




0.75
X
0.25
X


0.575
0.215
0.21
0.45
0.113
0.188
=
0.60 0.50
0.15 0.40
0.25 0.09
P(X|Y)
P(Y)
0.125
0.75
0.25
0.023
P(X|Y)
P(Y)

```
expected_x = np.dot(p_x_given_y, p_y)
```

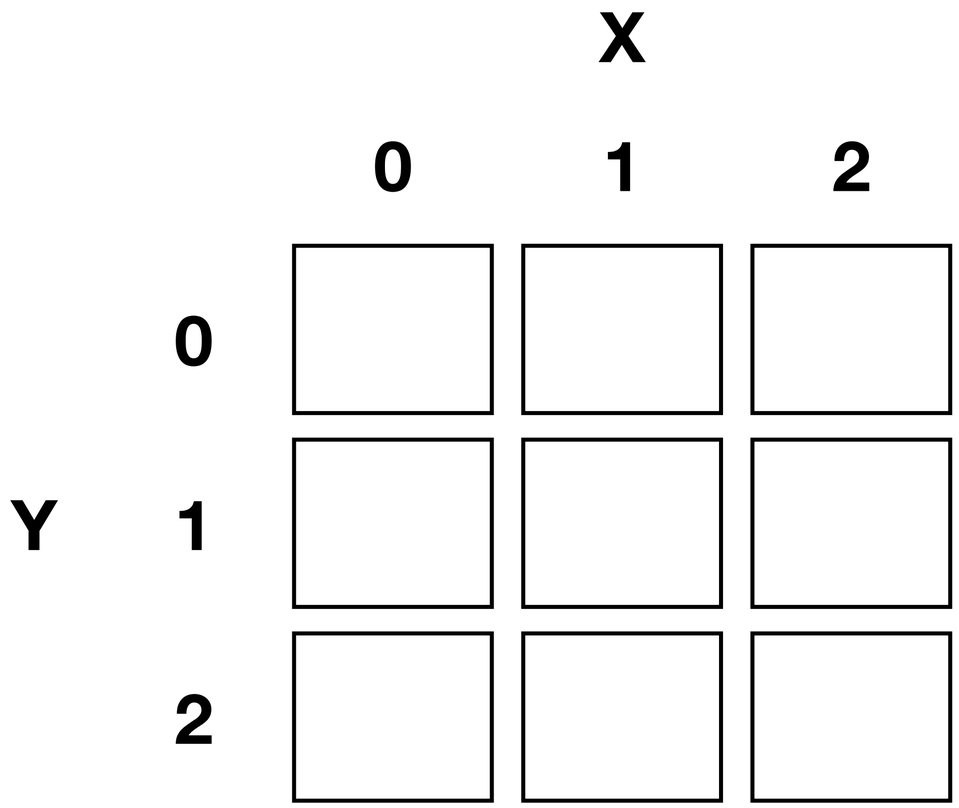
Back to Colab...

Questions on Categoricals / Conditionals?

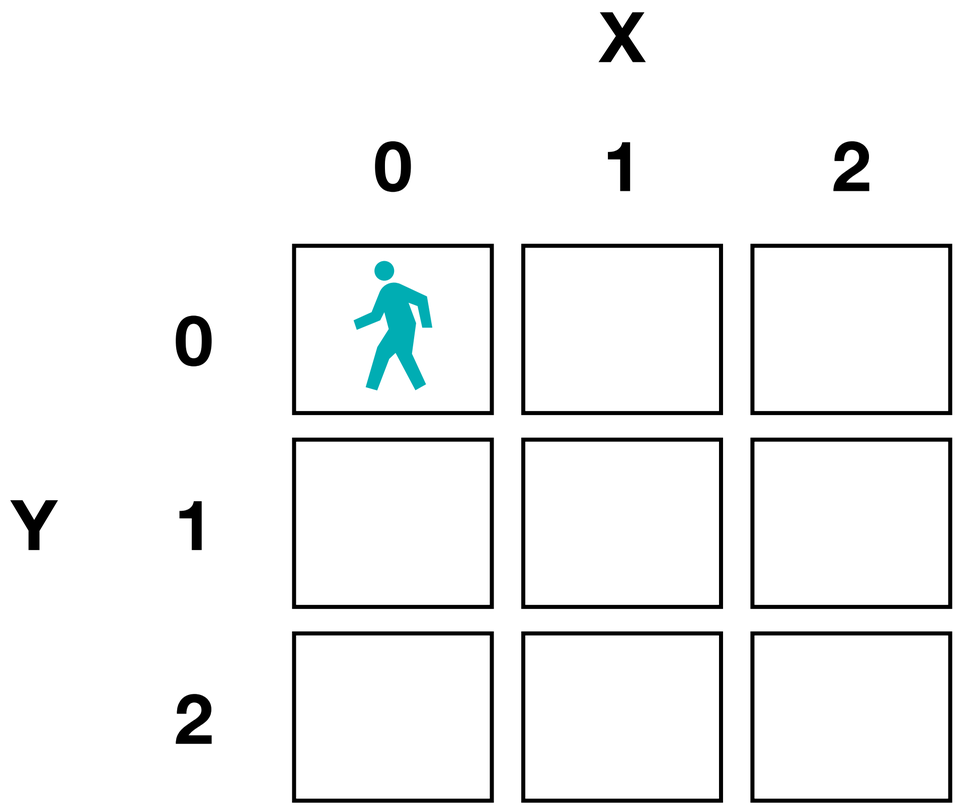
Setting up the task:

Grid World

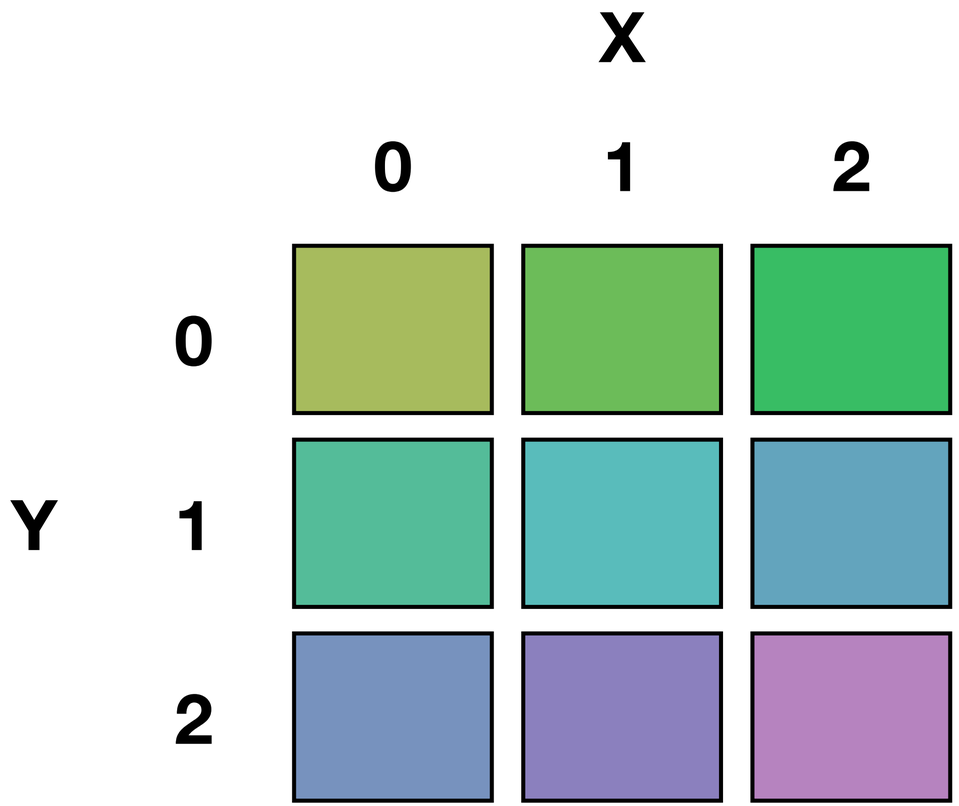
Grid World



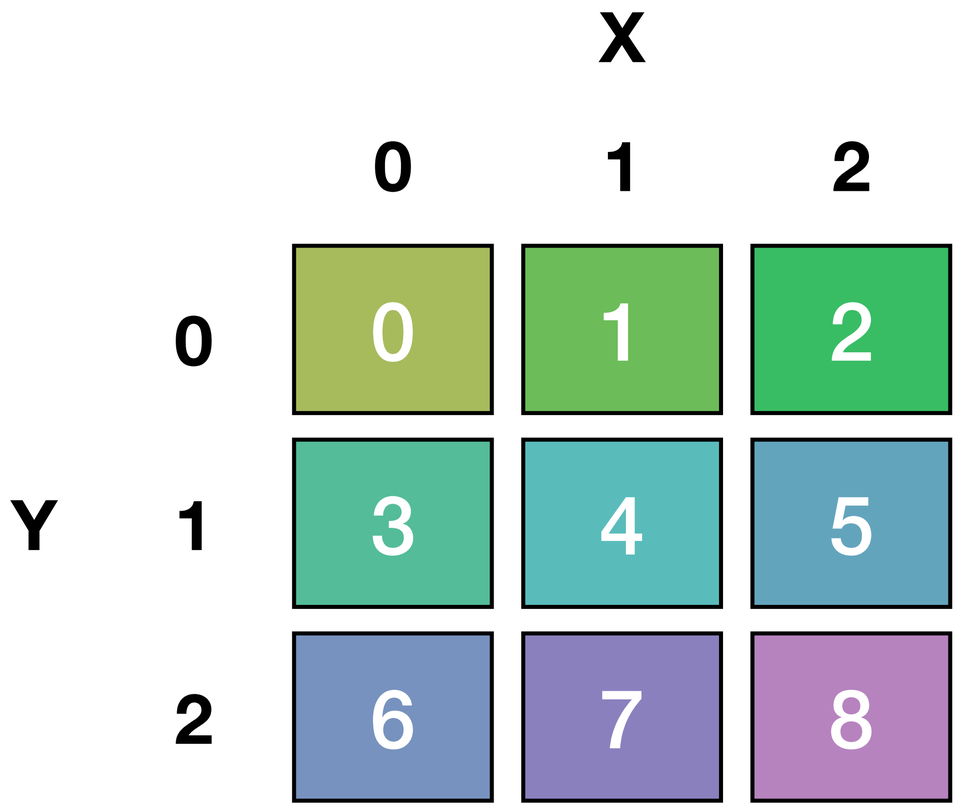
Grid World



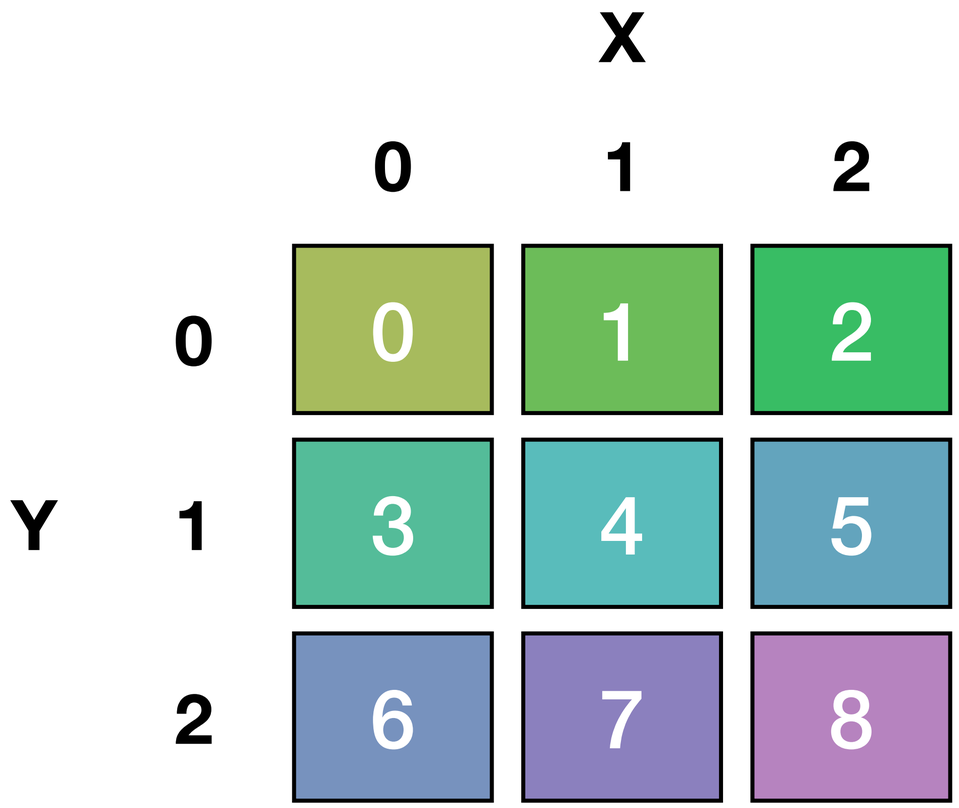
Grid World



Grid World



Grid World



(Y, X)

0: (0, 0)

1: (0, 1)

2: (0, 2)

3: (1, 0)

4: (1, 1)

5: (1, 2)

6: (2, 0)

7: (2, 1)

8: (2, 2)

Back to Colab...

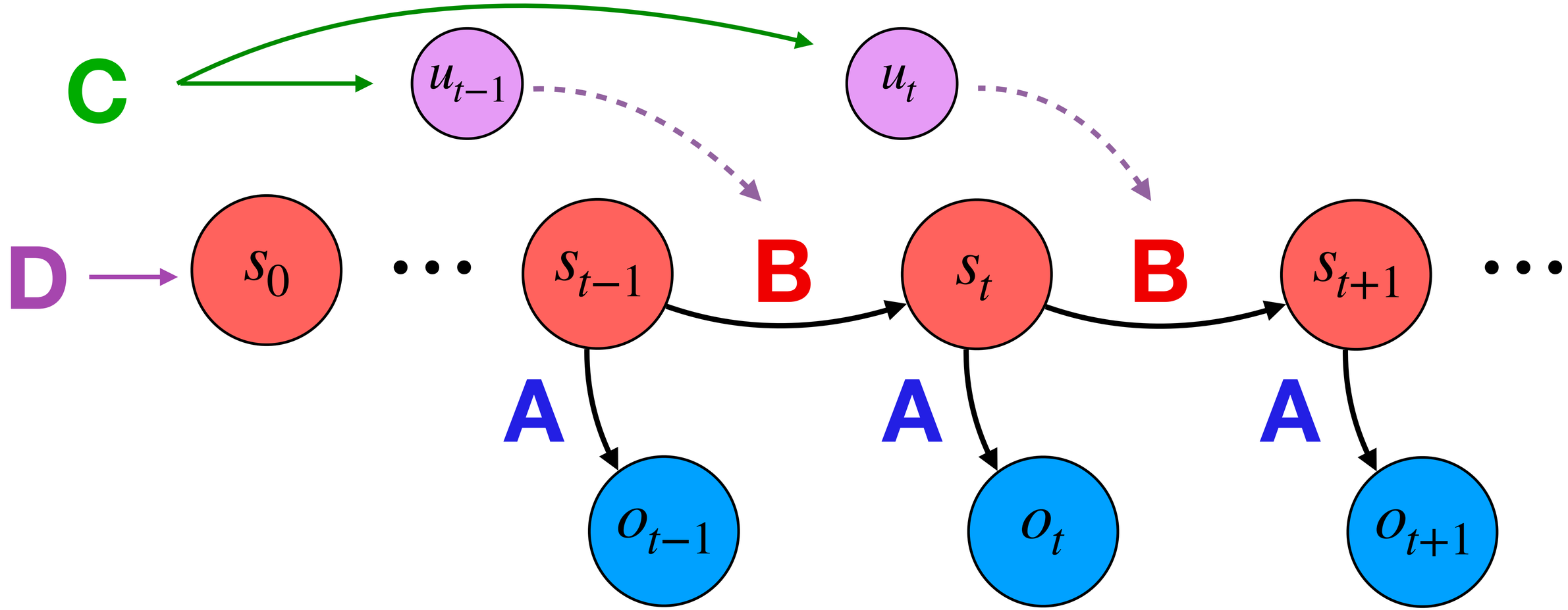
Generative model overview

$$P(o_{0:T}, s_{0:T}, u_{0:T}) = P(s_0) \prod_{t=1}^{T} P(o_t \mid s_t) P(s_t \mid s_{t-1}, u_{t-1}) P(u_{t-1})$$

$P(o_t \mid s_t)$ "Observation model" "Sensory likelihood mapping"	**A**
$P(s_t \mid s_{t-1}, u_{t-1})$ "Transition or dynamics model" "Dynamical mapping"	**B**
$P(o \mid \mathbf{C})$ "Prior over observations" "Prior preferences"	**C**
$P(s_0)$ Prior over initial hidden states	**D**

Generative model overview

$$P(o_{0:T}, s_{0:T}, u_{0:T}) = P(s_0)\prod_{t=1}^{T} P(o_t \mid s_t)P(s_t \mid s_{t-1}, u_{t-1})P(u_{t-1})$$



Setting up the generative model:

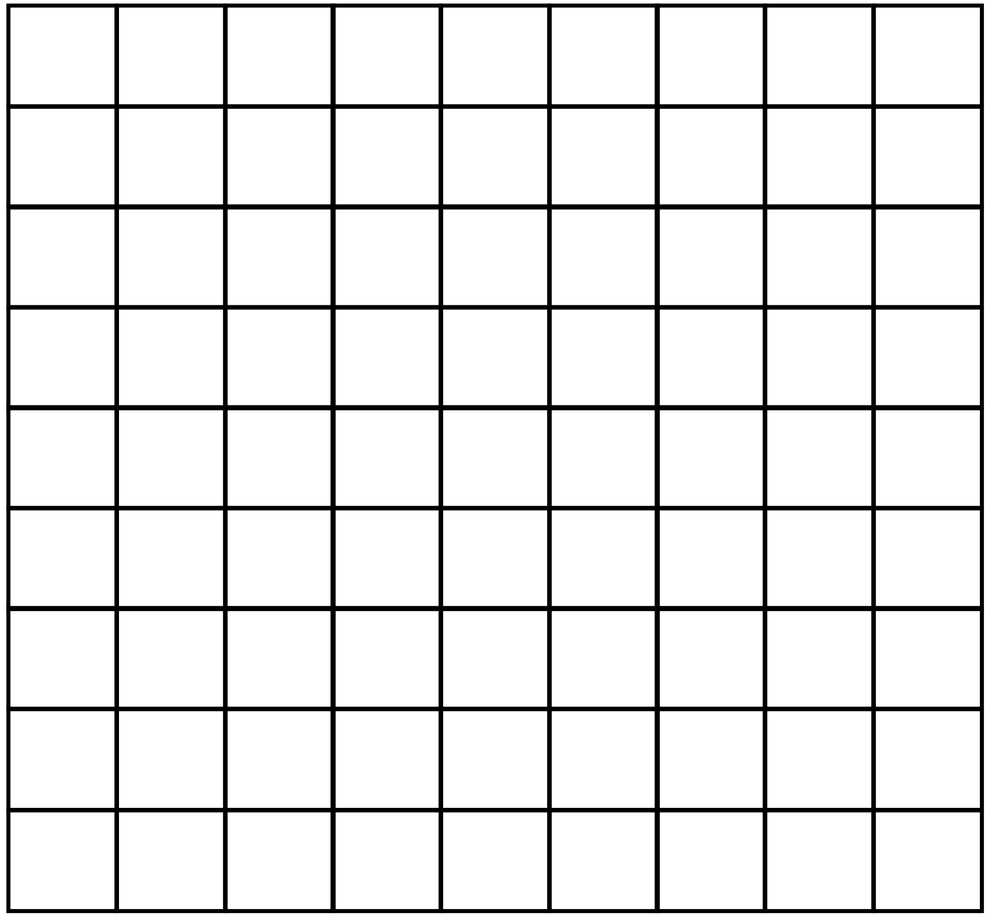
The A Matrix

The A Matrix

AKA $P(o \mid s)$

S

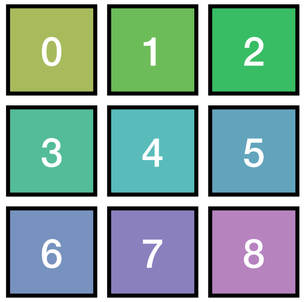
O

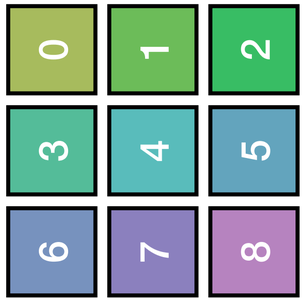


The A Matrix

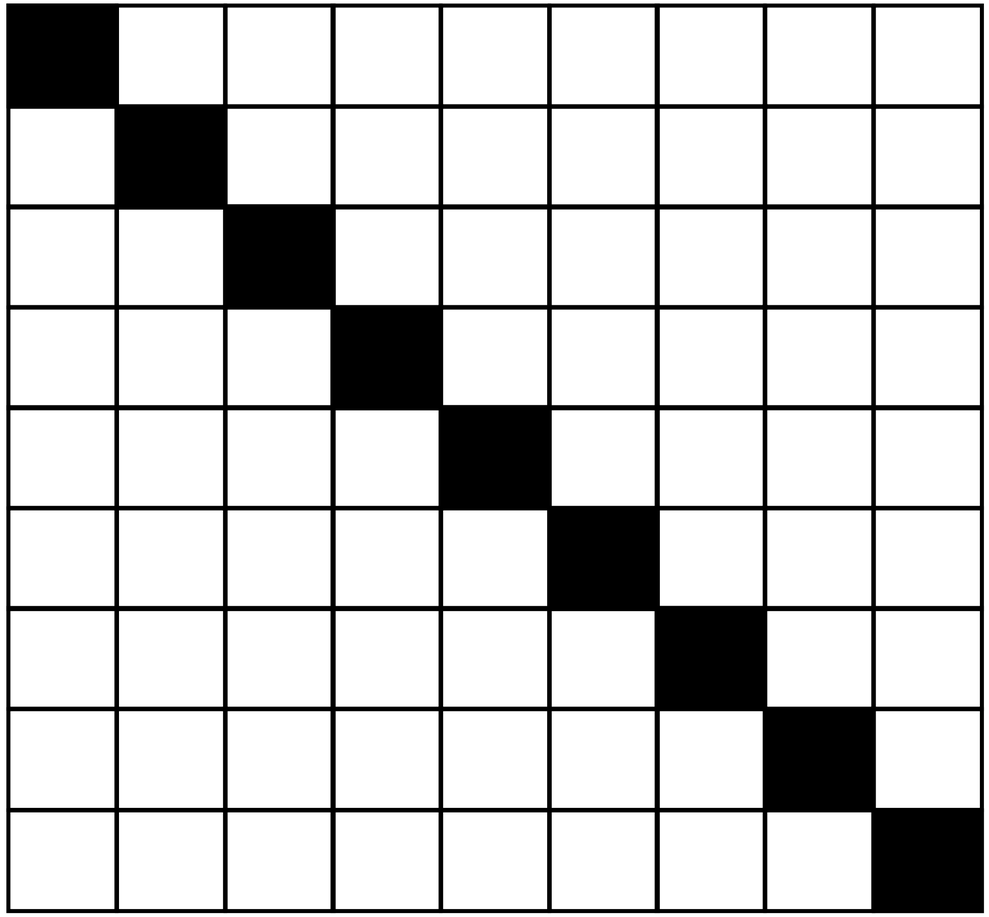
AKA $P(o \mid s)$

s





o

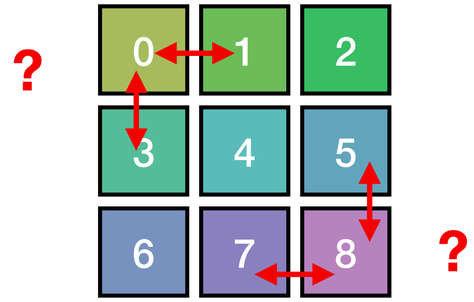


The mapping from ‘causes’ (hidden states s) to ‘consequences’ (observations o) is certain / unambiguous

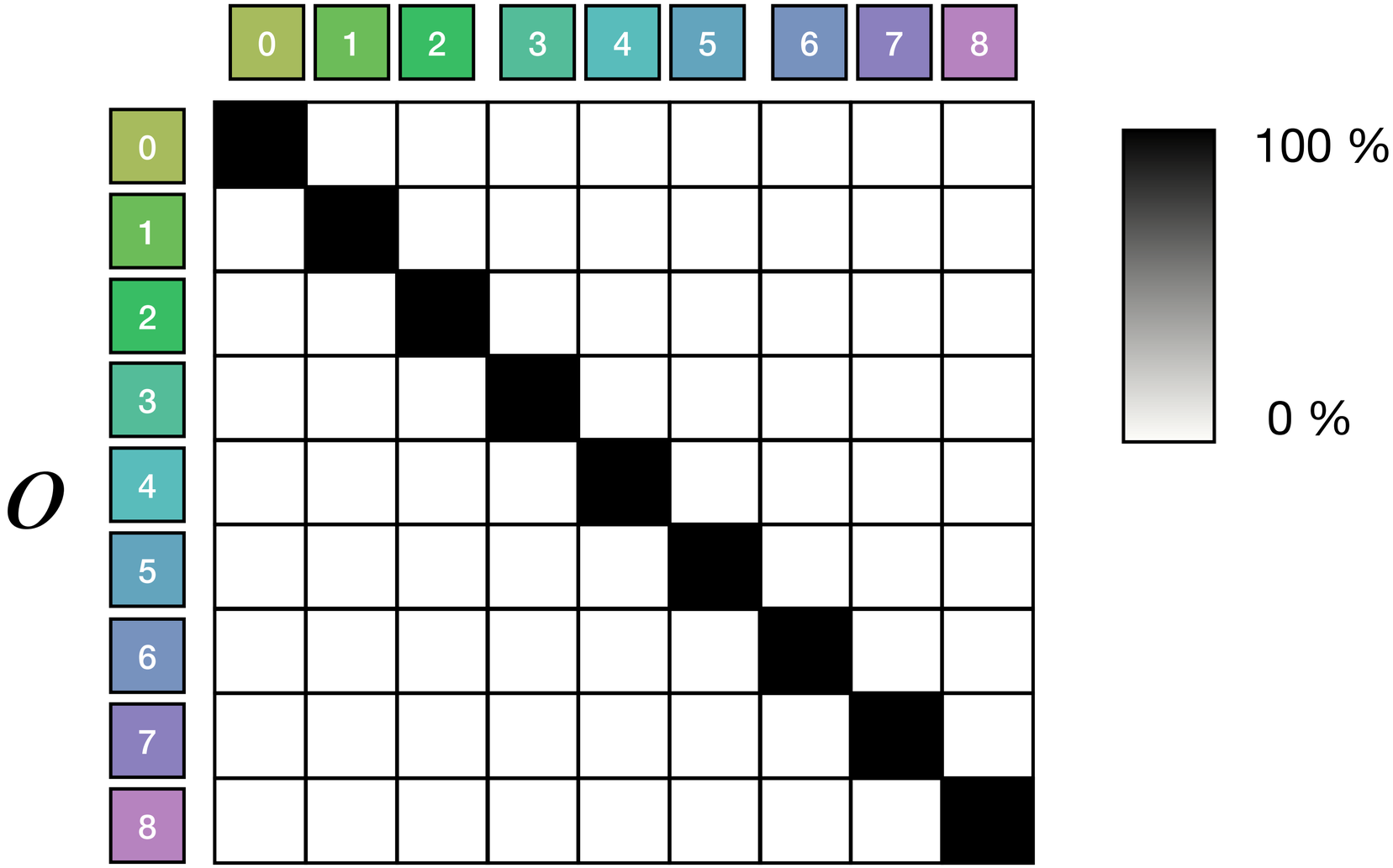
Back to Colab...

The A Matrix

AKA $P(o \mid s)$

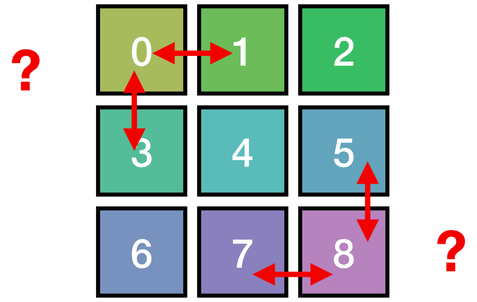


S

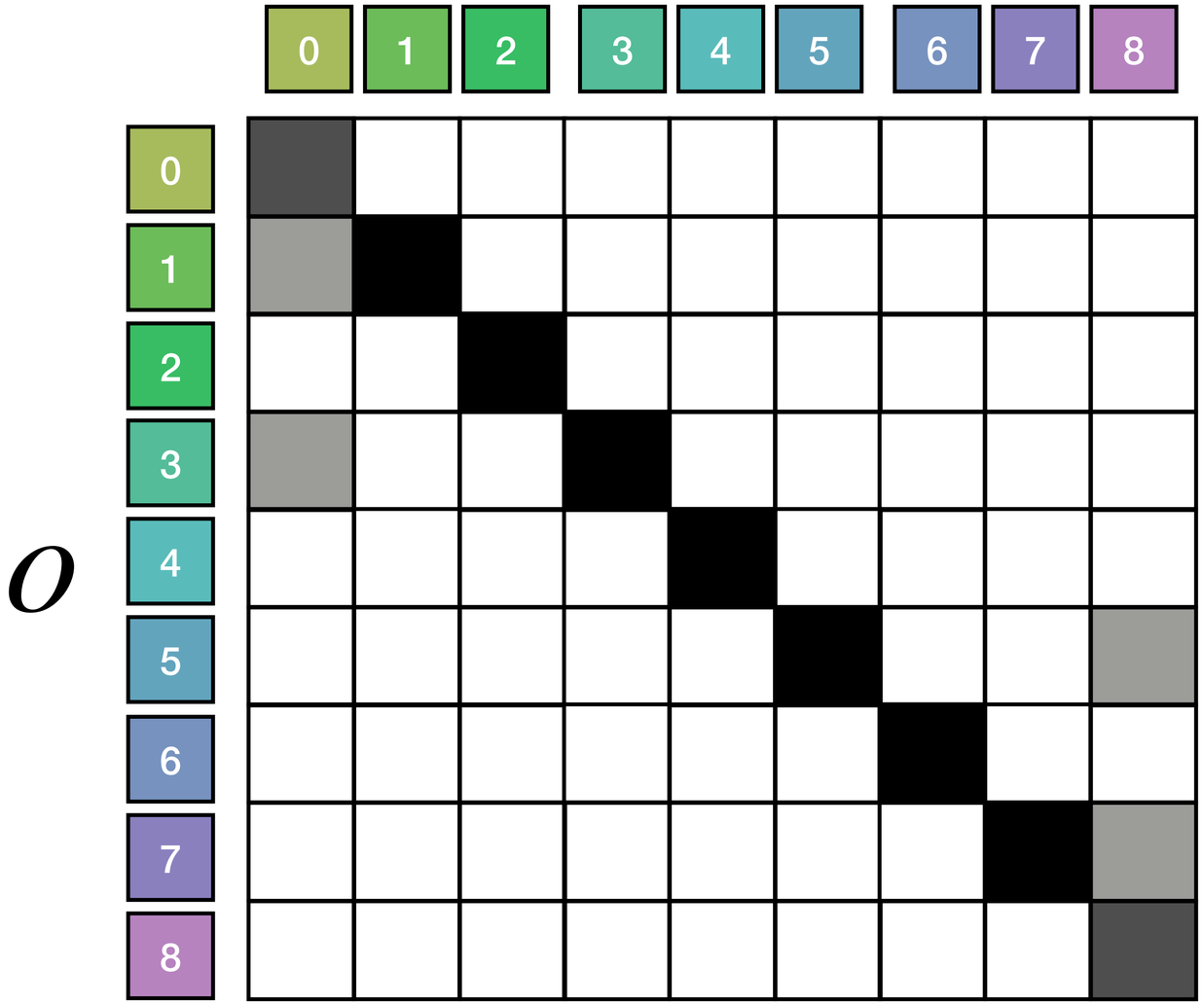


The A Matrix

AKA $P(o \mid s)$



S



100 %

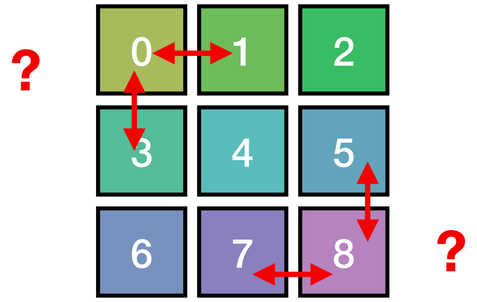
0 %

Uncertainty is represented via degeneracy in the A matrix (multiple possible observations, given a hidden state)

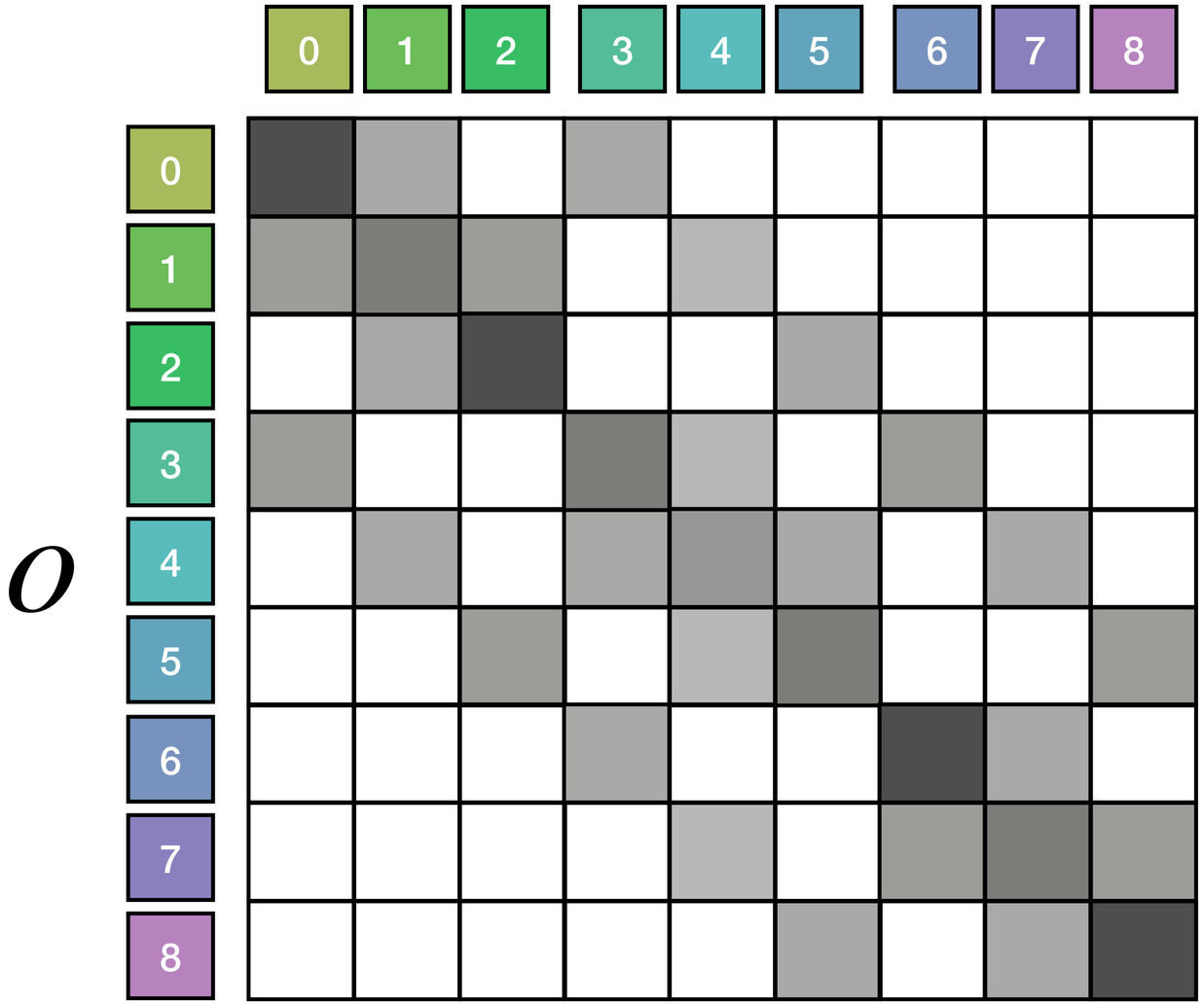
Remember: adjacent rows/ columns don't necessarily map to 'closer' locations in grid space

The A Matrix

AKA $P(o \mid s)$



s



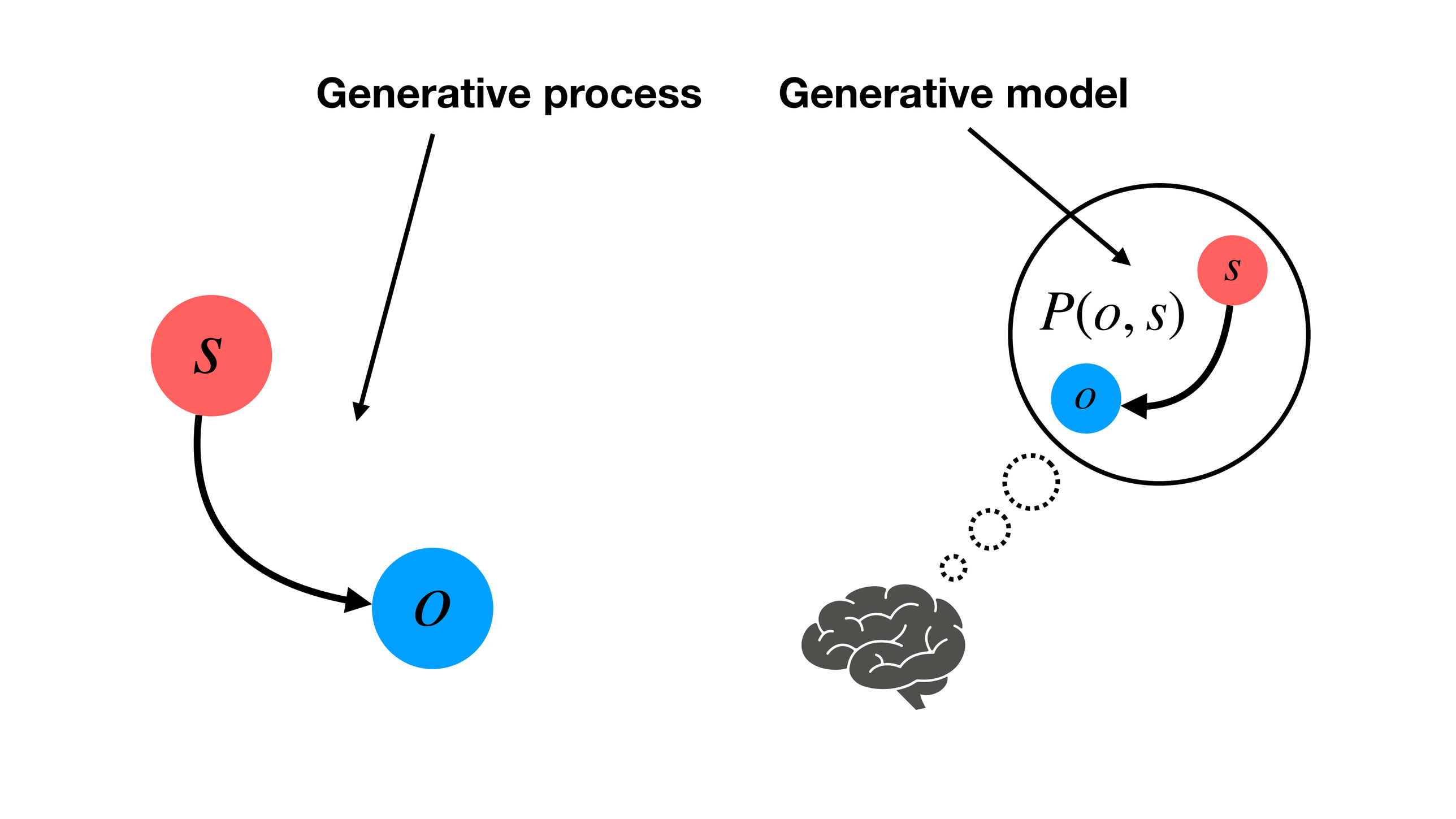
100 %

0 %

Uncertainty is represented via degeneracy in the A matrix (multiple possible observations, given a hidden state)

Remember: adjacent rows/ columns don't necessarily map to 'closer' locations in grid space

Back to Colab...

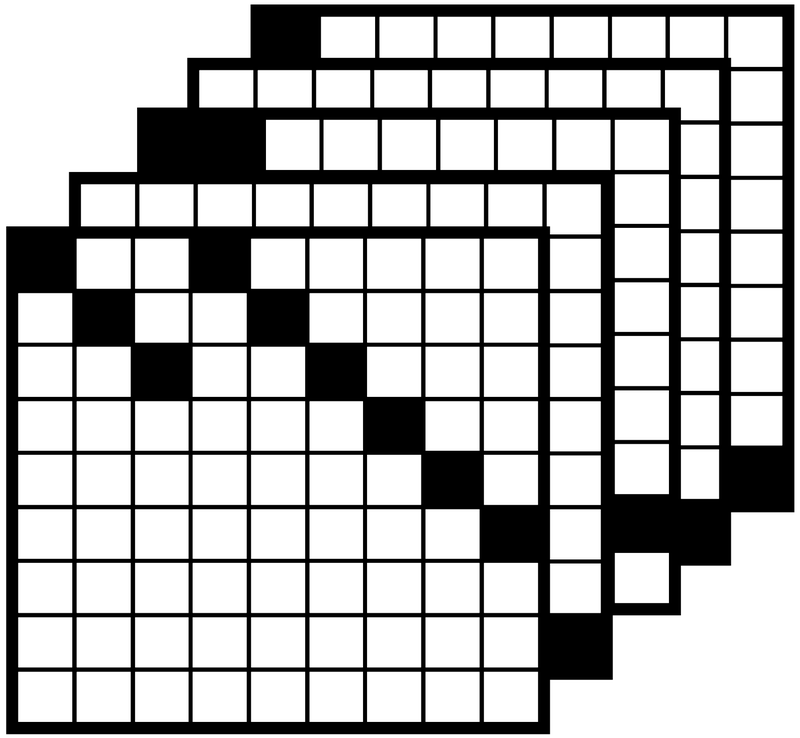

Generative process
Generative model
s
o
$P(o, s)$
s
o

Setting up the generative model:

The B Matrix

The B Matrix

AKA $P(s_t \mid s_{t-1}, u_{t-1})$



s_{t-1}

s_t

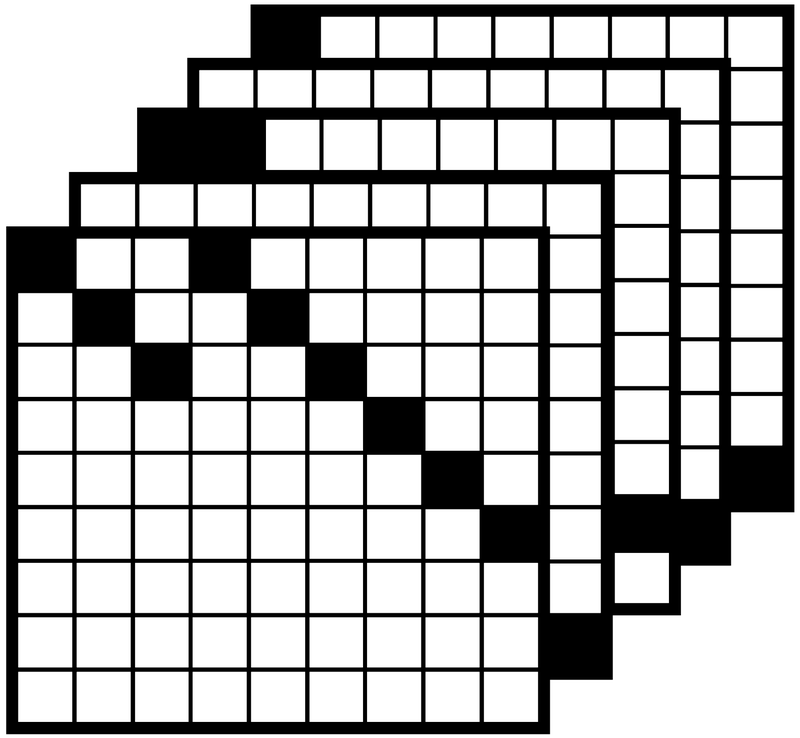
$P(s_t \mid s_{t-1}, u_{t-1} = 0)$

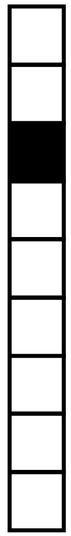
“MOST LIKELY”

The **B** Matrix

Expected states, given action, as dot product

AKA $P(s_t \mid s_{t-1}, u_{t-1})$



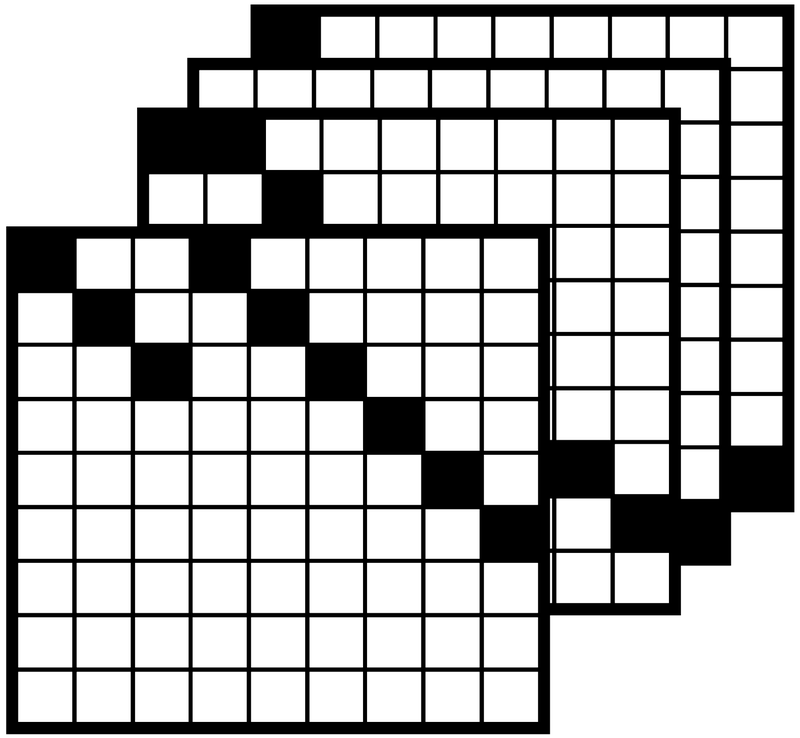


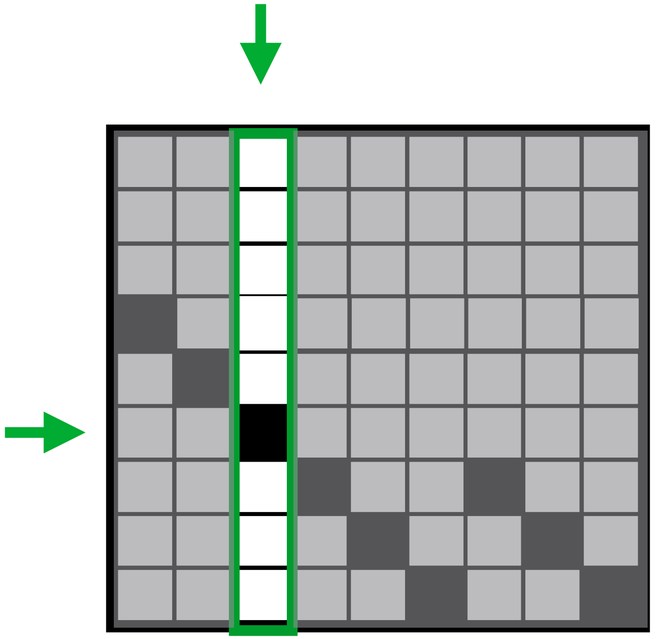
$P(s_{t-1})$

The **B** Matrix

Expected states, given action, as dot product

AKA $P(s_t \mid s_{t-1}, u_{t-1})$



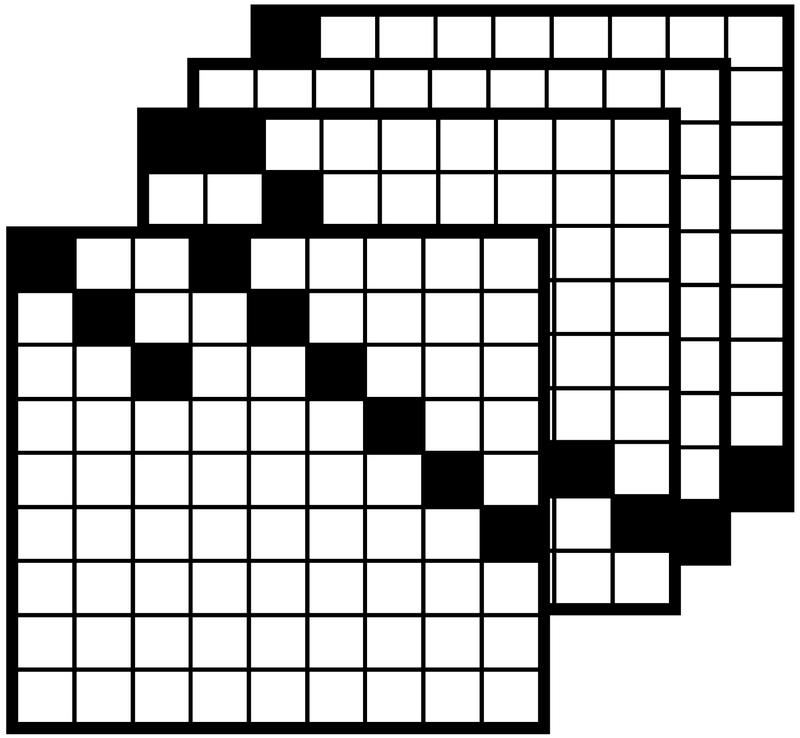


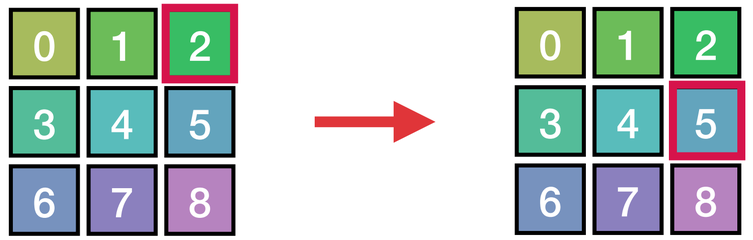
$P(s_{t-1})$

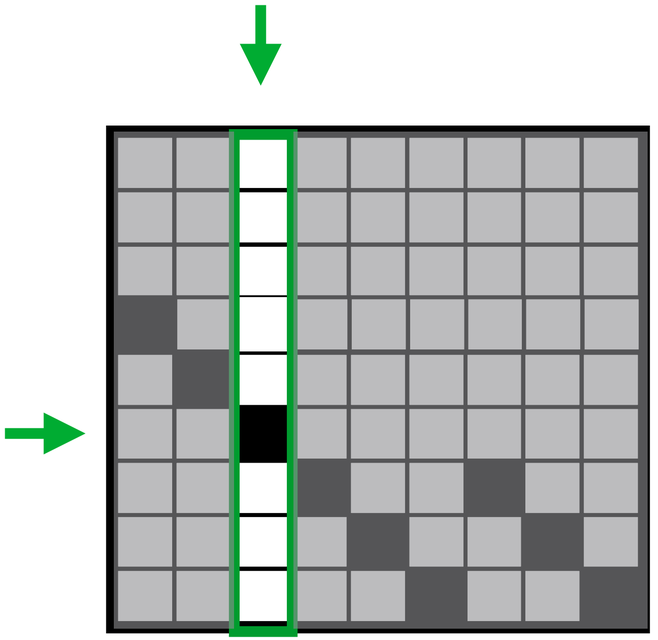
The **B** Matrix

AKA $P(s_t \mid s_{t-1}, u_{t-1})$

Expected states, given action, as dot product







$P(s_{t-1})$

$u = 1 \implies$ MOVE DOWN

The **B** Matrix

Expected states, given action, as dot product

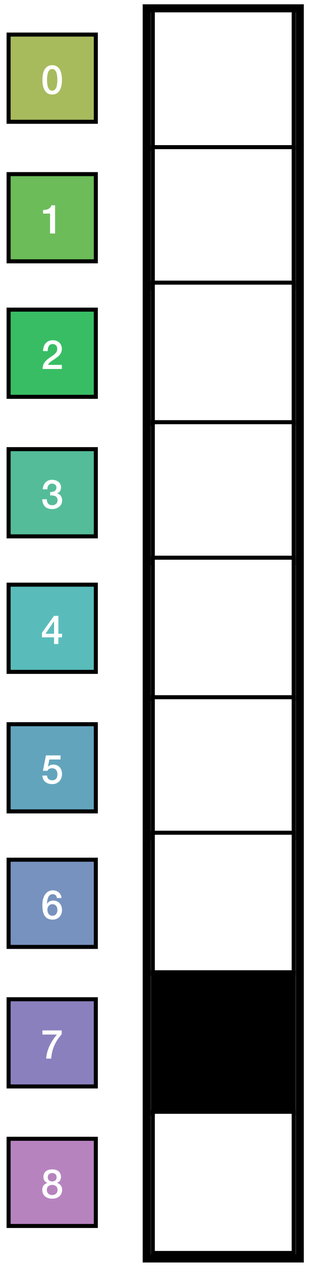
AKA $P(s_t \mid s_{t-1}, u_{t-1})$

$$\mathbf{E}_{P(s_{t-1})}\left[P(s_t \mid s_{t-1}, u_{t-1} = u_t)\right] = \texttt{B[:, :, u].dot(p_s)}$$

Back to Colab...

The C Vector

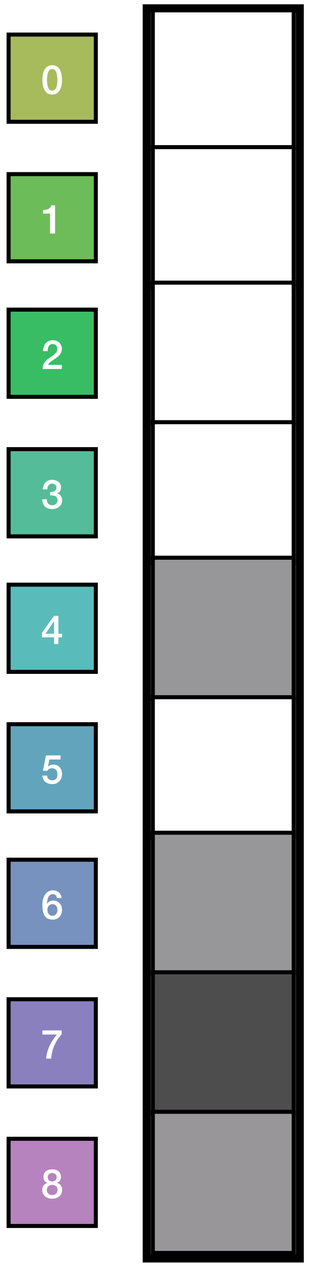
AKA $P(o \mid C)$



Y \ X	0	1	2
0	0	1	2
1	3	4	5
2	6	7	8

The C Vector

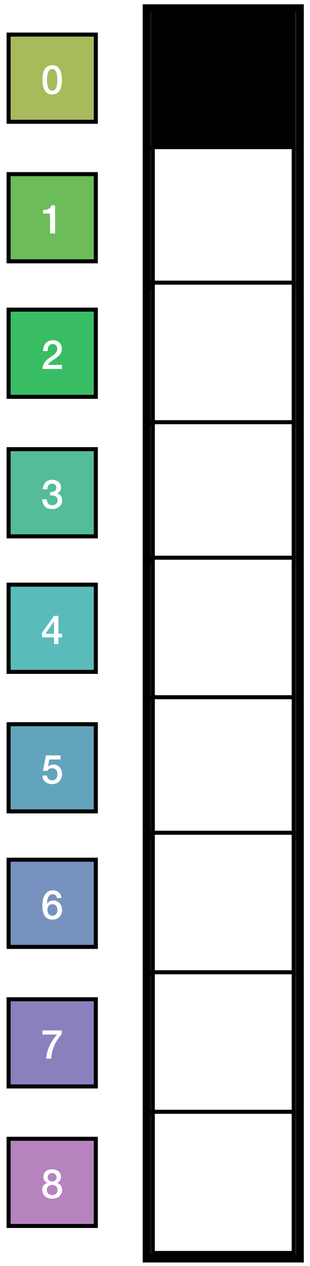
AKA $P(o \mid C)$

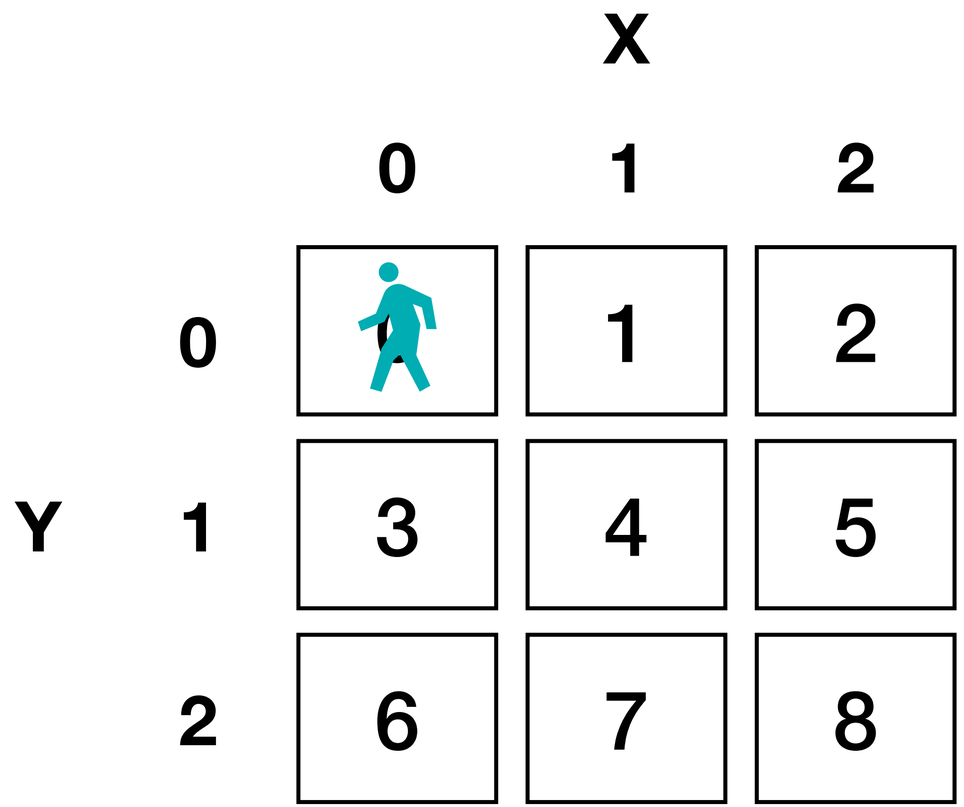


Y \ X	0	1	2
0	0	1	2
1	3	4	5
2	6	7	8

The D Vector

AKA $P(s_0)$





Back to Colab...

Generative model overview

$$P(o_{0:T}, s_{0:T}, u_{0:T}) = P(s_0)\prod_{t=1}^{T} P(o_t \mid s_t)P(s_t \mid s_{t-1}, u_{t-1})P(u_{t-1})$$

$P(o_t \mid s_t)$ "Observation model" "Sensory likelihood mapping"	**A**
$P(s_t \mid s_{t-1}, u_{t-1})$ "Transition or dynamics model" "Dynamical mapping"	**B**
$P(o \mid C)$ "Prior over observations" "Prior preferences"	**C**
$P(s_0)$ Prior over initial hidden states	**D**

First step of Active Inference

Hidden state inference

$$q(s) \approx p(s \mid o)$$

Approximate posterior True posterior

Recap:

Recall the free energy, the objective function for variational inference:

$$\mathscr{F} = \mathrm{D}_{KL}\left(q(s) \parallel P(s \mid o)\right) - \ln p(o)$$

Hidden state inference becomes the problem of finding the setting of $q(s)$ that minimises $\mathscr{F}$

Expand variational free energy

$$
\begin{aligned}
\mathscr{F} &= \mathrm{D}_{KL}\left(q(s) \parallel P(s \mid o)\right) - \ln P(o) \\
&= \mathrm{D}_{KL}\left(q(s) \parallel P(o, s)\right) \\
&= \sum_{s} q(s)\left(\ln q(s) - \ln P(o, s)\right) \\
&= \sum_{s} q(s)\left[\ln q(s) - \ln P(o \mid s) - \ln P(s)\right]
\end{aligned}
$$

Set derivative of $\mathscr{F}$ to 0 and solve for $q(s)$

$$\mathscr{F} = \sum_s q(s)\Big[\ln q(s) - \ln P(o \mid s) - \ln P(s)\Big]$$

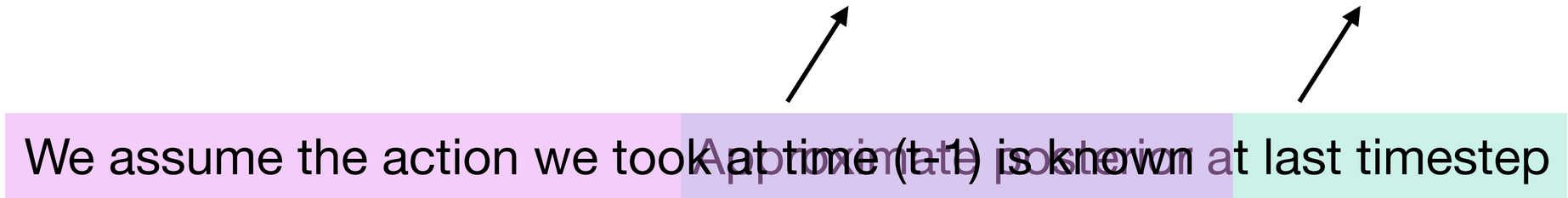
$$\frac{\partial \mathscr{F}}{\partial q} = \ln q(s) - \ln P(o \mid s) - \ln P(s) - \mathbf{1}$$

$$0 = \ln q(s) - \ln P(o \mid s) - \ln P(s) - \mathbf{1}$$

$$\implies \quad q(s) = \sigma\left(\ln P(o \mid s) + \ln P(s)\right)$$

$$P(s_t) = \mathbf{E}_{q(s_{t-1})}\left[P(s_t \mid s_{t-1}, u_{t-1})\right]$$

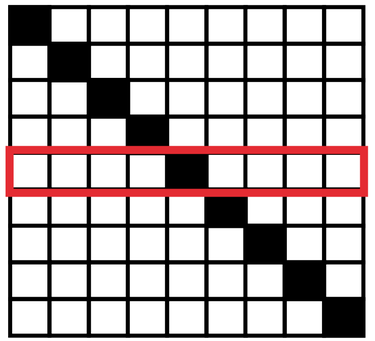
```
ps_current = B[:,:,u_past].dot(qs_past)
```



“Yesterday’s posterior becomes today’s prior”

We want to find the approximate posterior that minimises free energy, given the current observation

$$q(s) = \sigma\left(\ln P(o = o_t \,|\, s) + \ln P(s)\right)$$



```
log_likelihood = np.log( A[obs, :] )

log_prior = np.log( B[:, :,u].dot(qs_past) )

qs_new = softmax(log_likelihood + log_prior)
```

Back to Colab...

Picking up from Part I….

Action selection and expected free energy

Recall our generative model…

?

$$P(o_{0:T}, s_{0:T}, u_{0:T}) = P(s_0)\prod_{t=1}^{T} P(o_t \mid s_t)P(s_t \mid s_{t-1}, u_{t-1})P(u_{t-1})$$

- Active inference casts action-selection as a process of inference.
- Agents therefore maintain beliefs about control states u (which determine actions a).
- The space of control states is $\{u^0, u^1, u^2, u^3, u^4\}$, where each control state maps to a corresponding action $\{a^0, a^1, a^2, a^3, a^4\}$, which here are the allowable moves on the grid world, i.e. **MOVE UP, MOVE DOWN, MOVE LEFT, MOVE RIGHT, STAY**
- If u^0 is more likely in the action posterior $q(u_t)$, the agent is more likely to perform a^0

Recall our generative model…

$$P(o_{0:T}, s_{0:T}, u_{0:T}) = P(s_0)\prod_{t=1}^{T} P(o_t \mid s_t)P(s_t \mid s_{t-1}, u_{t-1})P(u_{t-1})$$

- You can calculate the posterior over control states by solving for approximate posterior $Q(u_t)$ that minimizes the *expected* free energy, which is like the free energy but its defined into the future:

$$\begin{aligned}\mathbf{G} &= \mathbf{E}_{Q(o_{0:T}, s_{0:T}, u_{0:T})}\left[\ln Q(s_{0:T}, u_{0:T}) - P(o_{0:T}, s_{0:T}, u_{0:T})\right] \\ &= \mathrm{D}_{KL}\left(Q(u_{0:T}) \parallel e^{-\sum_{t=0}^{T-1}\mathbf{G}_{t+1}(u_t)}\right)\end{aligned}$$

$$Q^*(u_t) = \sigma\left(-\mathbf{G}_{t+1}(u_t)\right)$$

Recall our generative model…

$$P(o_{0:T}, s_{0:T}, u_{0:T}) = P(s_0)\prod_{t=1}^{T} P(o_t \mid s_t)P(s_t \mid s_{t-1}, u_{t-1})P(u_{t-1})$$

- Where $\mathbf{G}$ is a vector of expected free energies, with one expected free energy (scalar) per control state.

$$-\mathbf{G} = \left[-G_{t+1}(u_t^0), -G_{t+1}(u_t^1), -G_{t+1}(u_t^2), \ldots, -G_{t+1}(u_t^N)\right]$$

- Where $G_{t+1}(u_t^0)$ is the free energy that is expected to accrue at time $t+1$ following taking action u^0 at time t

Now what actually is the expected free energy?

$$\mathbf{G}_{t+1}(u_t) = \mathbf{E}_{Q(s_{t+1},o_{t+1}|u_t)}[\ln Q(s_{t+1} \mid u_t) - \ln P(o_{t+1}, s_{t+1} \mid u_t)]$$

$$= \underbrace{\mathbf{E}_{Q(s_{t+1}|u_t)}\left[\mathbf{H}[P(o_{t+1} \mid s_{t+1})]\right]}_{\text{Predicted uncertainty}} + \underbrace{\mathrm{D}_{KL}\left(Q(o_{t+1} \mid u_t) \parallel \ln P(o_{t+1} \mid C)\right)}_{\text{Predicted divergence}}$$

- Predicted uncertainty: choose actions that are expected to result in hidden states that have an unambiguous (low entropy) mapping to observations

- Predicted divergence: choose actions that are expected to result in observations that are consistent with prior beliefs $P(o_{t+1} \mid C)$

- Remember that $P(o_{t+1} \mid C)$ specifies goals, desires and preferences

Now what actually is the expected free energy?

$$\mathbf{G}_{t+1}(u_t) = \mathbf{E}_{Q(s_{t+1},o_{t+1}|u_t)}[\ln Q(s_{t+1}\,|\,u_t) - \ln P(o_{t+1}, s_{t+1}\,|\,u_t)]$$

$$= \underbrace{\mathbf{E}_{Q(s_{t+1}|u_t)}\left[\mathbf{H}[P(o_{t+1}\,|\,s_{t+1})]\right]}_{\text{Predicted uncertainty}} + \underbrace{\mathrm{D}_{KL}\left(Q(o_{t+1}\,|\,u_t) \parallel \ln P(o_{t+1}\,|\,C)\right)}_{\text{Predicted divergence}}$$

$$= \underbrace{-\,\mathbf{E}_{Q(s_{t+1},o_{t+1}|u_t)} \ln \left[\frac{Q(s_{t+1}\,|\,o_{t+1}, u_t)}{Q(s_{t+1}\,|\,u_t)}\right]}_{\text{(-ve) epistemic value}} \underbrace{-\,\mathbf{E}_{Q(o_{t+1}|u_t)}\left[\ln P(o_{t+1}\,|\,C)\right]}_{\text{(-ve) expected utility}}$$

Connecting risk and ambiguity to epistemic value and utility

$$
\begin{aligned}
\mathbf{G}_{t+1}(u_t) &= \mathbf{E}_{Q(s_{t+1}|u_t)}\left[\mathbf{H}[P(o_{t+1}\,|\,s_{t+1})]\right] + \mathrm{D}_{KL}\left(Q(o_{t+1}\,|\,u_t) \parallel \ln \tilde{P}(o_{t+1})\right) \\
&= -\mathbf{E}_{Q(s_{t+1}|u_t)}\left[P(o_{t+1}\,|\,s_{t+1}) \ln P(o_{t+1}\,|\,s_{t+1})\right] + \mathbf{H}\left[Q(o_{t+1}\,|\,u_t)\right] - \mathbf{E}_{Q(o_{t+1}|u_t)}\left[\ln \tilde{P}(o_{t+1})\right] \\
&= -\mathbf{E}_{Q(s_{t+1}|u_t)}\left[\mathbf{E}_{P(o_{t+1}|s_{t+1})}[\ln P(o_{t+1}\,|\,s_{t+1})]\right] + \mathbf{E}_{Q(o_{t+1}|u_t)}\left[\ln Q(o_{t+1}\,|\,u_t)\right] - \mathbf{E}_{Q(o_{t+1}|u_t)}\left[\ln \tilde{P}(o_{t+1})\right] \\
&= -\mathbf{E}_{Q(s_{t+1}|u_t)}\left[\mathbf{E}_{P(o_{t+1}|s_{t+1})}[\ln P(o_{t+1}\,|\,s_{t+1})]\right] + \mathbf{E}_{Q(o_{t+1}|u_t)}\left[\ln \frac{{}^{*}P(o_{t+1}\,|\,s_{t+1})Q(s_{t+1}\,|\,u_t)}{Q(s_{t+1}\,|\,o_{t+1}, u_t)}\right] - \mathbf{E}_{Q(o_{t+1}|u_t)}\left[\ln \tilde{P}(o_{t+1})\right] \\
&= -\mathbf{E}_{Q(s_{t+1},o_{t+1}|u_t)}\left[\ln \frac{Q(s_{t+1}\,|\,o_{t+1}, u_t)}{Q(s_{t+1}\,|\,u_t)}\right] - \mathbf{E}_{Q(o_{t+1}|u_t)}[\ln \tilde{P}(o_{t+1})]
\end{aligned}
$$

* via Bayes rule

$$
Q(o_{t+1}\,|\,u_t) = \frac{P(o_{t+1}\,|\,s_{t+1})Q(s_{t+1}\,|\,u_t)}{Q(s_{t+1}\,|\,o_{t+1}, u_t)}
$$

How to compute $\mathbf{G}(u_t)$ in practice

$$\mathbf{G}_{t+1}(u_t) = \mathbf{E}_{Q(s_{t+1}|u_t)}\left[\mathbf{H}[P(o_{t+1}\,|\,s_{t+1})]\right] + \mathrm{D}_{KL}(Q(o_{t+1}\,|\,u_t) \parallel \ln P(o_{t+1}\,|\,C))$$

1. Compute the states expected under the action: $Q(s_{t+1}\,|\,u_t)$
2. Compute the entropy of $\mathbf{A}$, i.e. $\mathbf{H}\left[P(o\,|\,s)\right]$, expected under $Q(s_{t+1}\,|\,u_t)$

+

3. Compute the observations expected under the action: $Q(o_{t+1}\,|\,u_t)$
4. Compute the KL divergence between $Q(o_{t+1}\,|\,u_t)$ and $\mathbf{C}$, i.e. the prior preferences

5. Add the expected entropy to the KL divergence

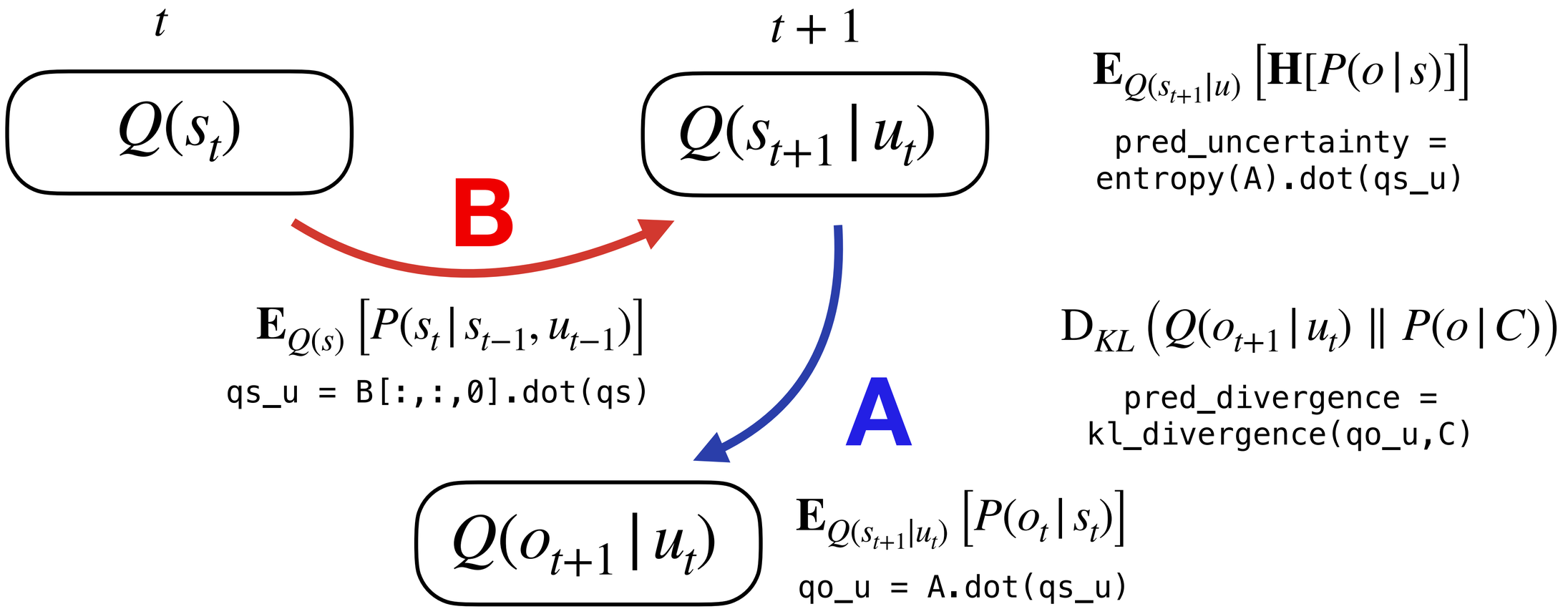
Action Selection

$$Q(u_t) = \sigma(-\mathbf{G})$$

Actions are then sampled as $a_t \sim Q(u_t)$

How to compute $\mathbf{G}(u_t)$ in practice

$$\mathbf{G}_{t+1}(u_t) = \mathbf{E}_{Q(s_{t+1}|u_t)}\left[\mathbf{H}[P(o_{t+1}\,|\,s_{t+1})]\right] + \mathrm{D}_{KL}(Q(o_{t+1}\,|\,u_t) \parallel \ln \tilde{P}(o_{t+1}))$$



Back to Colab...

Planning as inference

$$P(o_{0:T}, s_{0:T}, \pi) = P(s_0)P(\pi)\prod_{t=1}^{T} P(o_t \mid s_t)P(s_t \mid s_{t-1}, u_{t-1})P(u_{t-1} \mid \pi)$$

$$\pi = \{u_1, u_2, \ldots, u_H\}$$

- Now we have introduced a latent variable in our generative model: policies π, which are sequences of control states, where the length of the sequence is referred to as the "planning horizon" H
- Doing inference over *policies*, rather than actions, actually reduces to quite a simple algorithm for planning….

Planning as inference

- The prior (and posterior) probability of a policy is proportional to the negative expected free energy for that policy, **integrated over time**

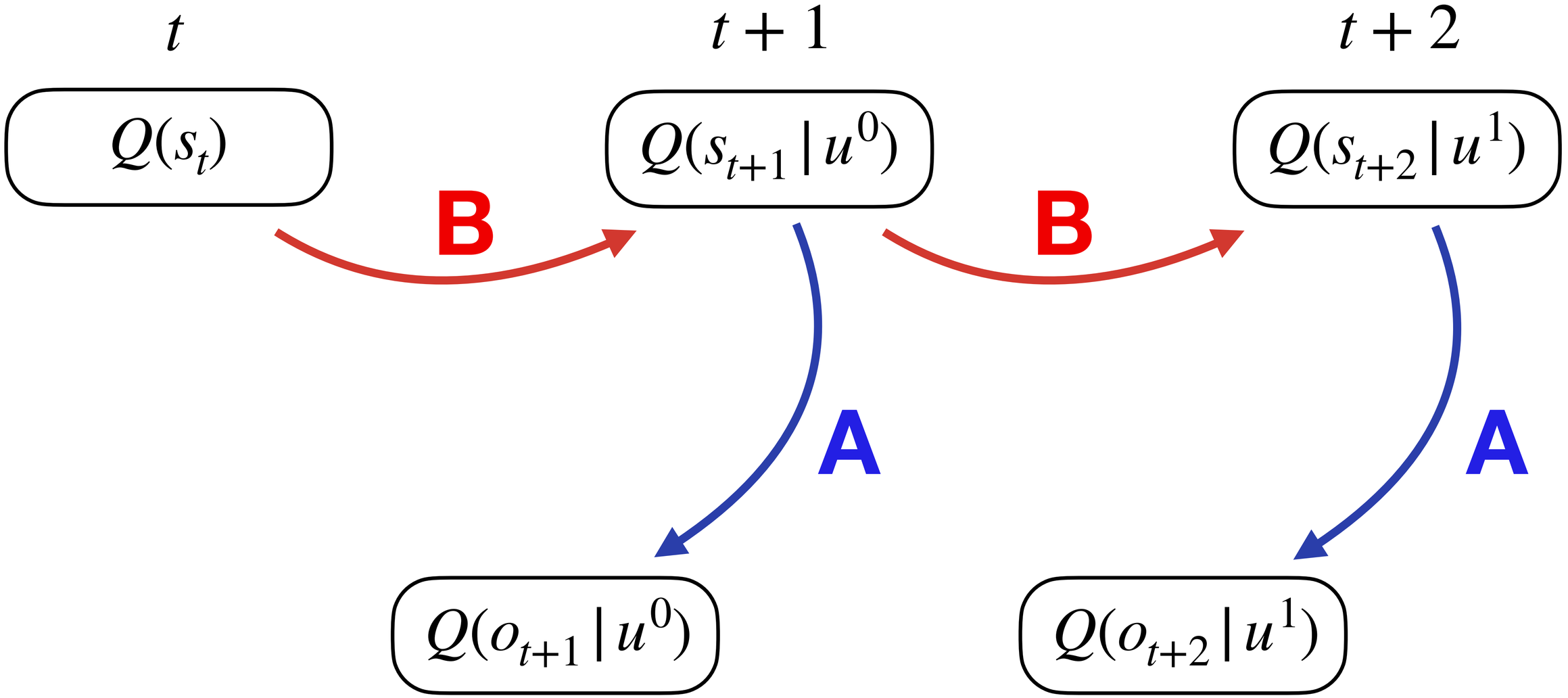
$$P(\pi) = \sigma\left(-\sum_{\tau=1}^{H} \mathbf{G}_{\tau}(\pi)\right)$$

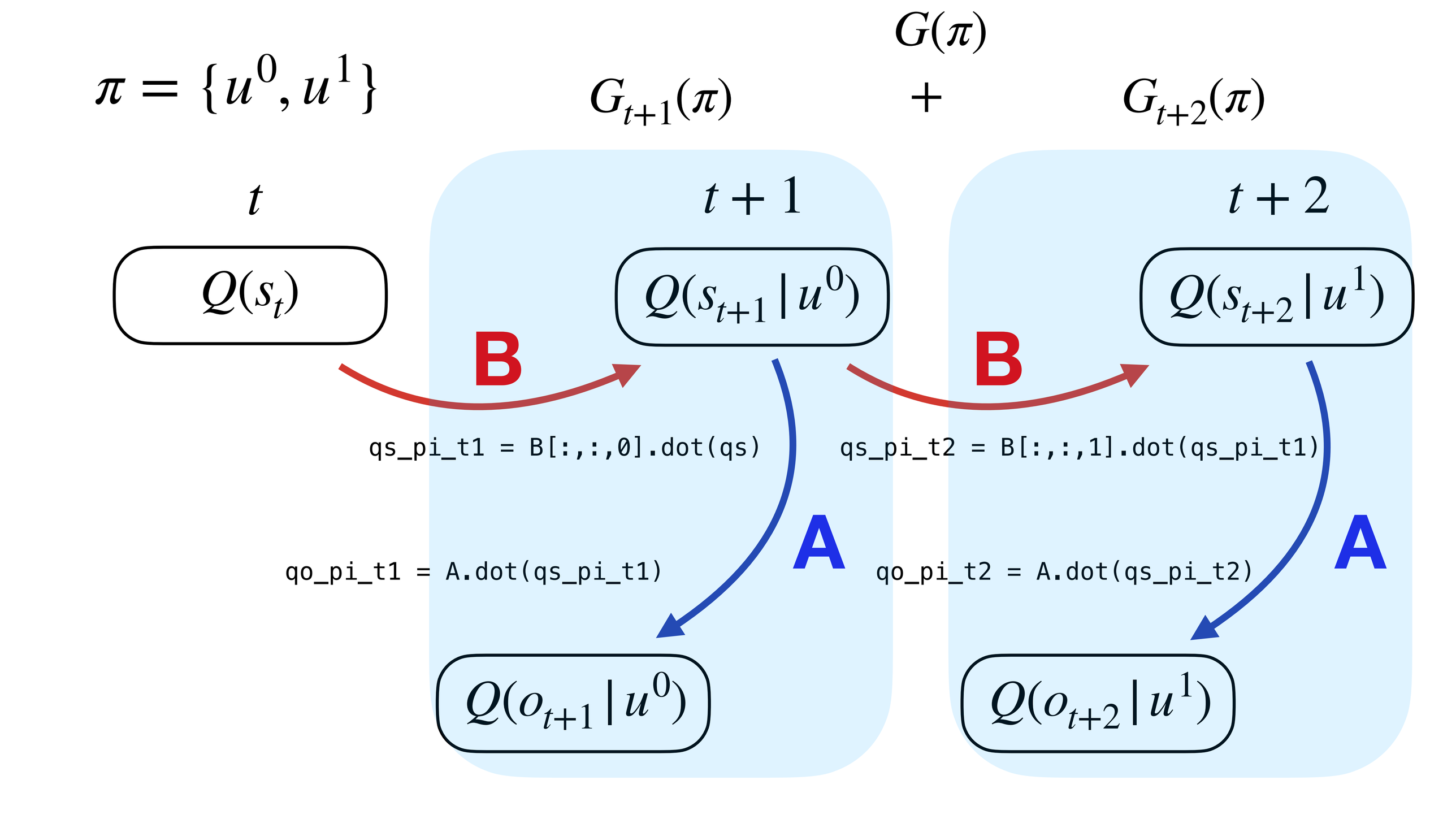
- Where the free energy for a particular policy at a particular time point is:

$$\mathbf{G}_{\tau}(\pi) = \mathbf{G}(u \sim \pi(\tau))$$

where the notation $u \sim \pi(\tau)$ indicates the control state entailed by policy π at time τ

$$\pi = \{u^0, u^1\}$$




$G(\pi)$
$\pi = \{u^0, u^1\}$
$G_{t+1}(\pi)$
+
$G_{t+2}(\pi)$
t
$t+1$
$t+2$
$Q(s_t)$
$Q(s_{t+1}|u^0)$
$Q(s_{t+2}|u^1)$
B
B
qs_pi_t1 = B[:,:,0].dot(qs)
qs_pi_t2 = B[:,:,1].dot(qs_pi_t1)
A
A
qo_pi_t1 = A.dot(qs_pi_t1)
qo_pi_t2 = A.dot(qs_pi_t2)
$Q(o_{t+1}|u^0)$
$Q(o_{t+2}|u^1)$

Back to Colab...

Action selection

- Once we've computed the posterior over policies $Q(\pi) = \sigma(-\mathbf{G})$, we can then compute the posterior probability of each control state, using the action entailed by policy i at time t: $\pi_i(t)$, and the probability of each policy:

$$Q(u_t) = \sum_i \pi_i(t) Q(\pi_i)$$

- Finally, we can sample an action by sampling from this posterior over control states $Q(u_t)$:

$$a_t \sim Q(u_t)$$

$$a_t = \text{argmax}_u \, Q(u_t)$$

Back to Colab...