General Principles of Human and Machine Learning

Lecture 8: Common tools for understanding brain and neural networks

Dr Charline Tessereau

https://hmc-lab.com/GPHML.html

THIS WEEK:

- SUMMARY PAST MATERIAL COVERED + FINISHING LAST SESSION’S MATERIAL
- RL TO UNDERSTAND COGNITIVE PROCESSES
- MANIFOLD TO UNDERSTAND NEURAL POPULATION DYNAMICS
- RSA TO UNDERSTAND NEURAL REPRESENTATIONS

RL RESULTS: INVESTIGATING MECHANISMS OF DECISION MAKING AND LEARNING

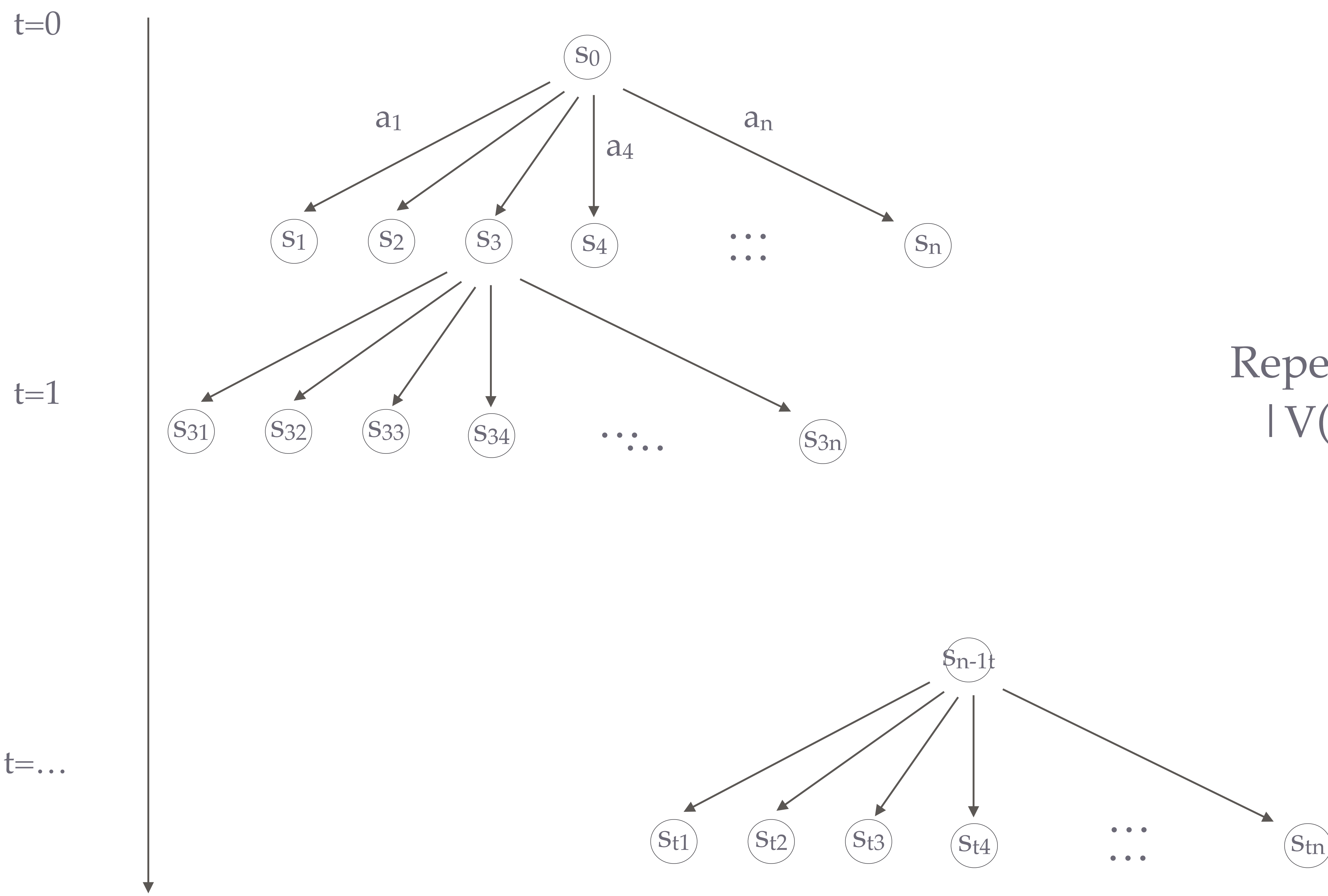
- Value/Q-learning: formalizes operant and pavlovian conditioning
- Policy gradient: formalizes ‘repeat bias’/‘win-stay’ behaviors
- Actor-critic: investigates boundary between values and actions
- Hierarchical RL: investigates how we break-out tasks
- Model-based: how we plan ahead

MODEL-BASED

The transitions between states and the reward vector over the states are known.

Learning the optimal policy using simulated experience
=graph search

Value update



Repeat steps until:
$|V(s_t)\text{-}V(s_{t+1})| < \varepsilon$

MODEL-BASED

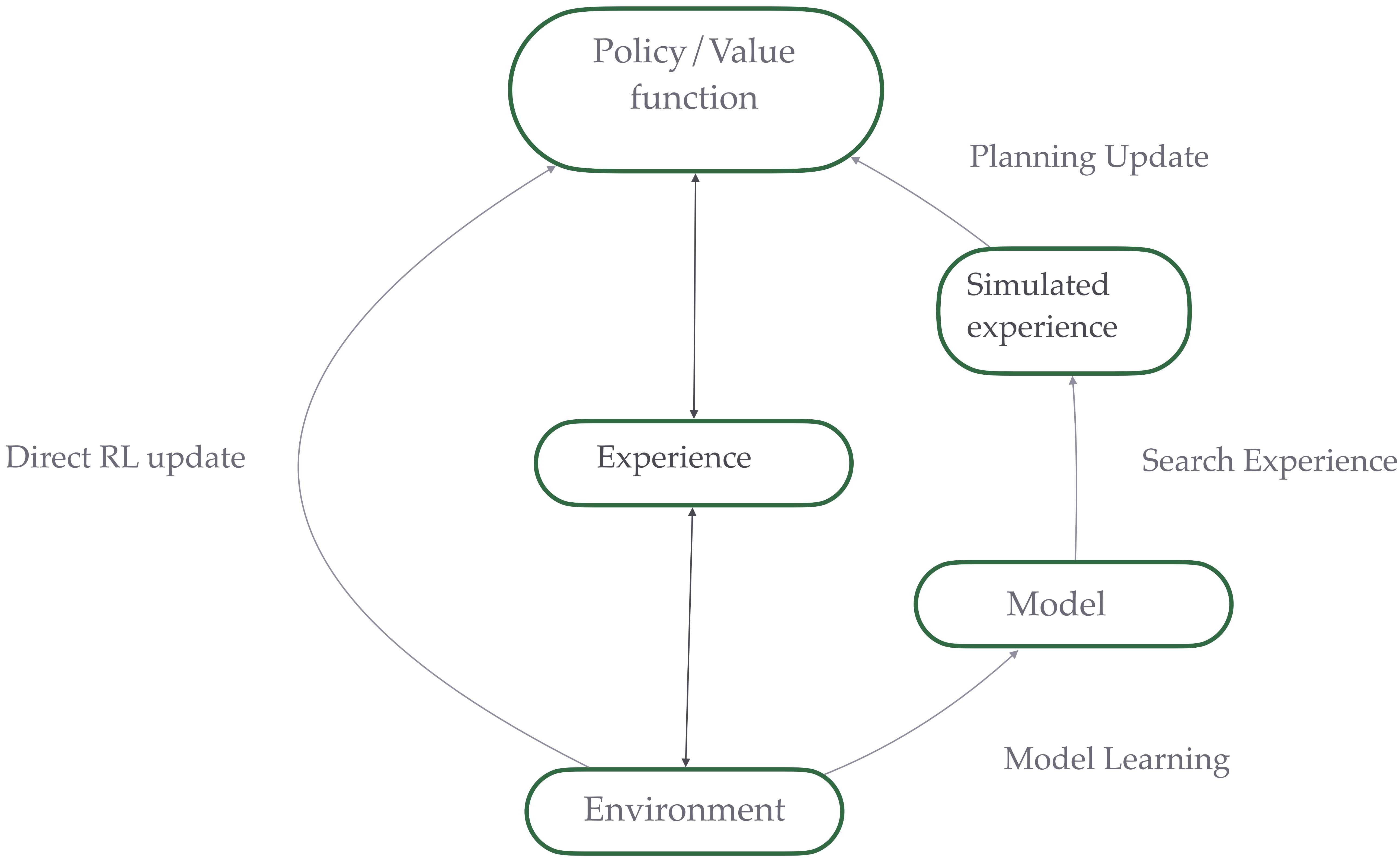
- Requires full knowledge of the model
- Can be very expensive - especially in large state spaces

MODEL-BASED

- Requires full knowledge of the model
- Can be very expensive - especially in large state spaces

- Some intermediary exists: DYNA
  D - Dynamic
  Y - Immediate
  N - Neighbourhood
  A - Approximation

DYNA


Policy / Value function
Planning Update
Simulated experience
Direct RL update
Experience
Search Experience
Model
Model Learning
Environment

DYNA

- Initialize the model, initialize the Q-values

DYNA

- Initialize the model, initialize the Q-values
- Policy learning:

DYNA

- Initialize the model, initialize the Q-values
- Policy learning:
    - Select an action based on the current policy and observe the resulting reward and next state

DYNA

- Initialize the model, initialize the Q-values
- Policy learning:
    - Select an action based on the current policy and observe the resulting reward and next state
    - Update the Q-values using Q learning

DYNA

- Initialize the model, initialize the Q-values
- Policy learning:
    - Select an action based on the current policy and observe the resulting reward and next state
    - Update the Q-values using Q learning
    - Store the previous state, the action, reward and next state in the ‘model’

DYNA

- Initialize the model, initialize the Q-values
- Policy learning:
    - Select an action based on the current policy and observe the resulting reward and next state
    - Update the Q-values using Q learning
    - Store the previous state, the action, reward and next state in the 'model'
- Planning phase: 'simulate' offline the model to learn the Q-values.

DYNA

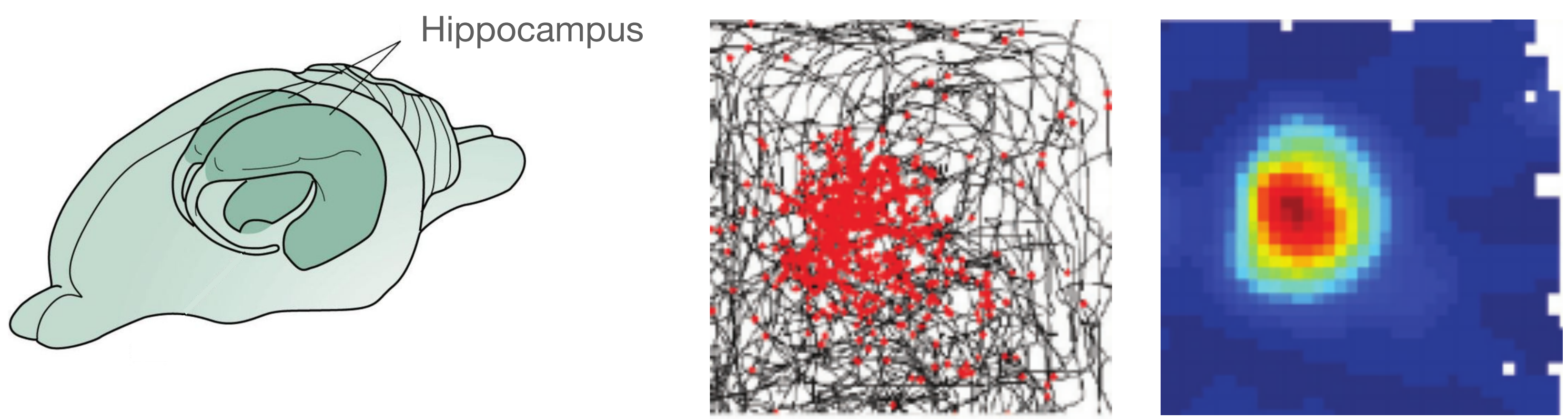
- Initialize the model, initialize the Q-values
- Policy learning:
    - Select an action based on the current policy and observe the resulting reward and next state
    - Update the Q-values using Q learning
    - Store the previous state, the action, reward and next state in the ‘model’
- Planning phase: ‘simulate’ offline the model to learn the Q-values.

-> Replay

- Enables to save ‘real experience’ to have an estimate of the Q function

REPLAY AS A NEURAL MECHANISM

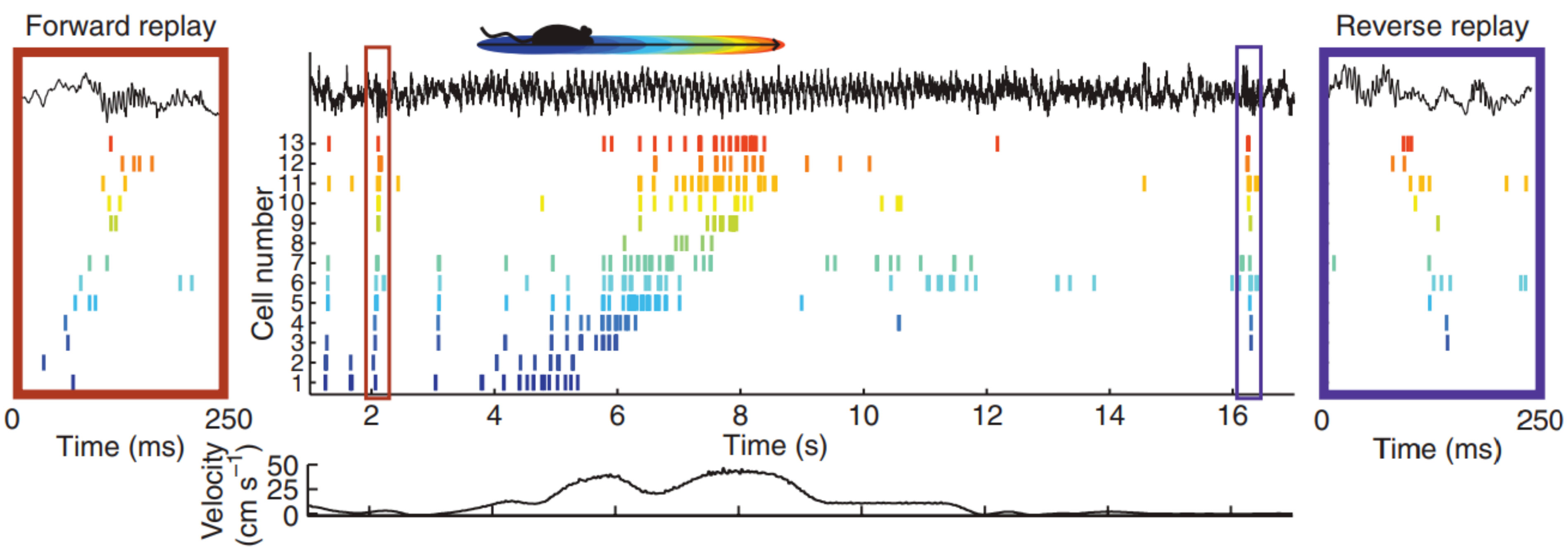
Hippocampal place cells fire more around their preferred location:



Marozzi and Jeffery, *Curr. Biol.*, 2012.

REPLAY AS A NEURAL MECHANISM

Hippocampal place cells activities display 'replay' patterns:



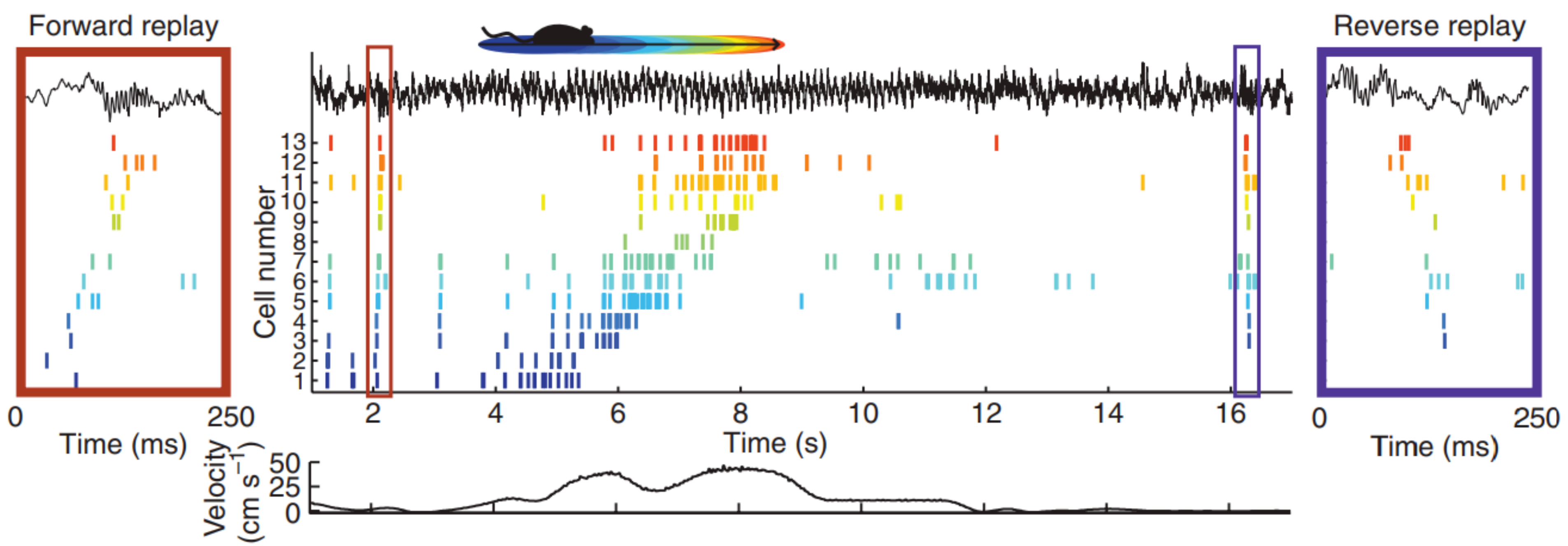
Carr, Jadhav, and Frank. *Nat. Neuro.* (2018)

DYNA

- Initialize the model, initialize the Q-values
- Policy learning:
    - Select an action based on the current policy and observe the resulting reward and next state
    - Update the Q-values using Q learning
    - Store the previous state, the action, reward and next state in the ‘model’
- Planning phase: ‘simulate’ offline the model to learn the Q-values.
- Enables to save ‘real experience’ to have an estimate of the Q function
- Problem: really dependent on your model!

REPLAY AS A NEURAL MECHANISM

Hippocampal place cells activities display 'replay' patterns:



- Scaled with performance
- Influenced by new rewards, novelty

Carr, Jadhav, and Frank. *Nat. Neuro.* (2018)

DYNA

- Initialize the model, initialize the Q-values
- Policy learning:
    - Select an action based on the current policy and observe the resulting reward and next state
    - Update the Q-values using Q learning
    - Store the previous state, the action, reward and next state in the 'model'
- Planning phase: 'simulate' offline the model to learn the Q-values.
- Enables to save 'real experience' to have an estimate of the Q function
- Problem: really dependent on your model!

DYNA

- "Simulating the model” requires a good model!
  - Important (ongoing) questions about:
    - how, what and when storing events
    - how, what and when retrieving/learning from past

DYNA

- Examples of models:
  - (s,a,r,s’) - random selection among past experiences

DYNA

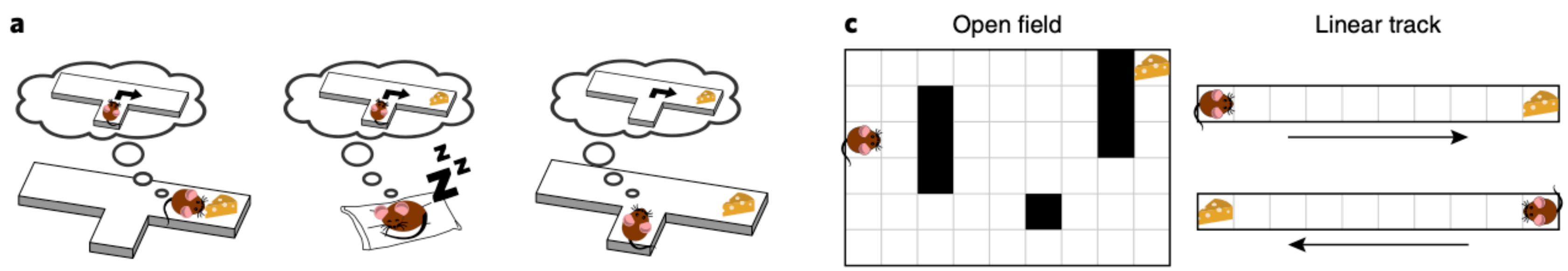
- Examples of models:
    - (s,a,r,s') - random selection among past experiences
    - Importance sampling: based on the probability distribution of the target policy compared to the behavior policy

DYNA

- Examples of models:
    - (s,a,r,s’) - random selection among past experiences
    - Importance sampling: based on the probability distribution of the target policy compared to the behavior policy
    - Prioritized replay: assigns priorities to the experiences based on their importance or learning potential - typically based on the TD error

DYNA

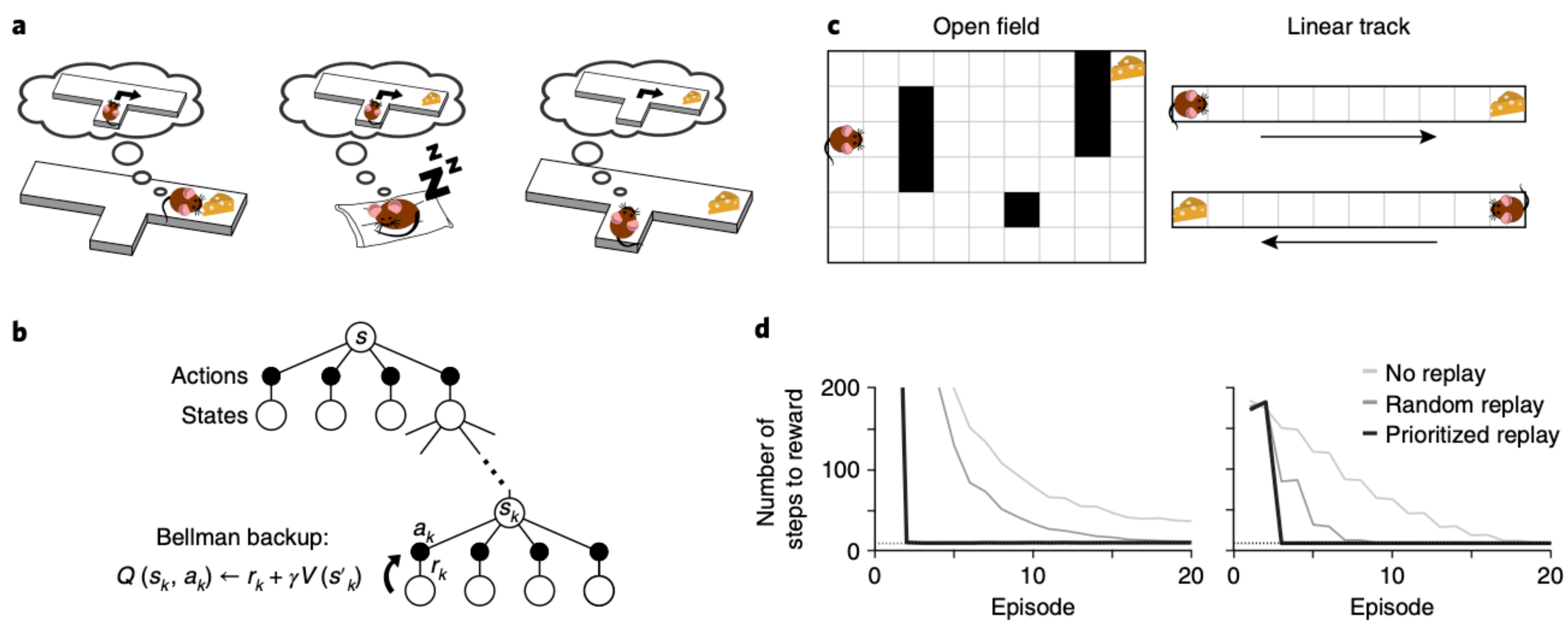
Prioritised replay: assigns priorities to the experiences based on their importance or learning potential.



Mattar and Daw. *Nature*. (2018)

DYNA

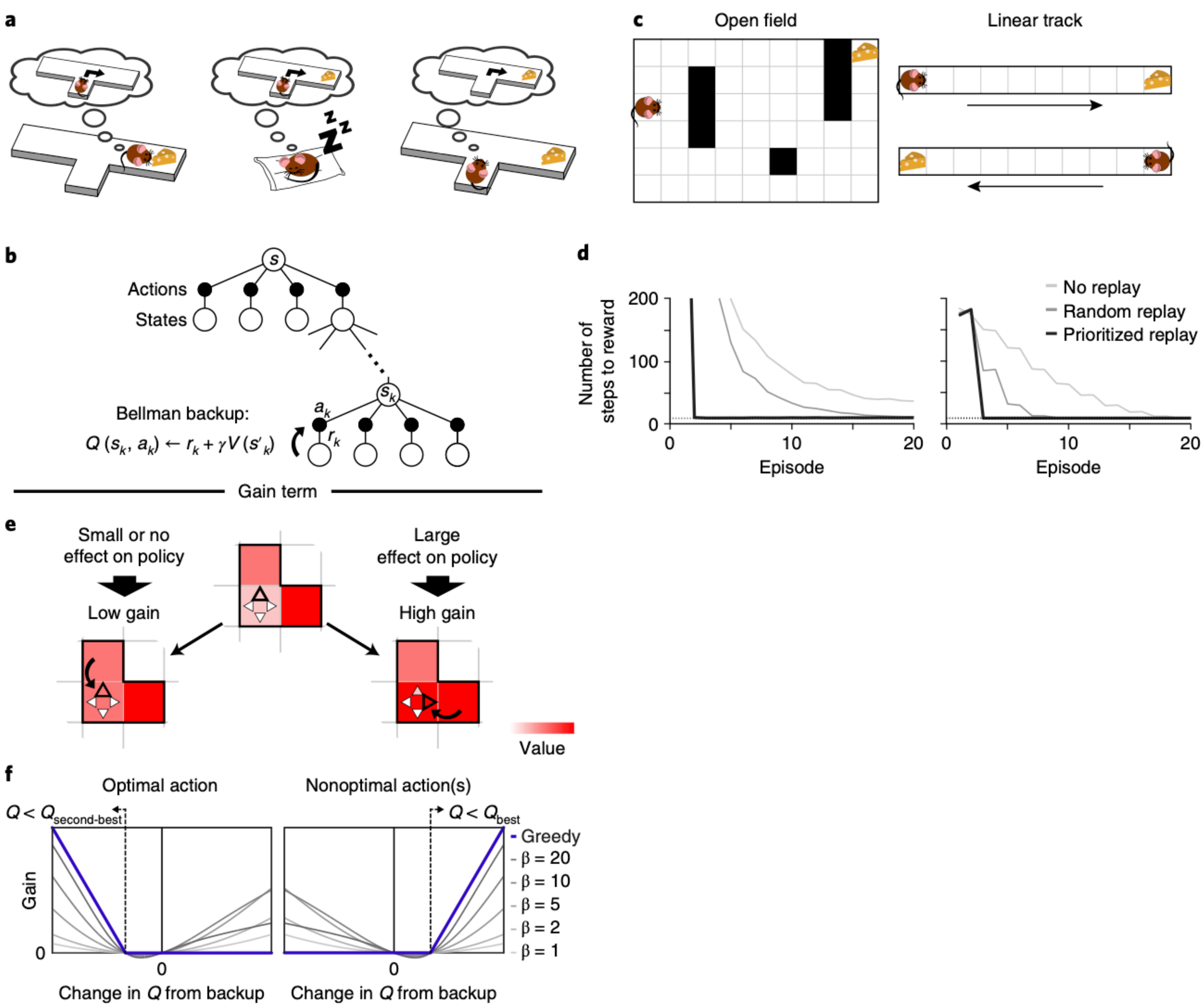
Prioritised replay: assigns priorities to the experiences based on their importance or learning potential.



Mattar and Daw. *Nature*. (2018)

DYNA

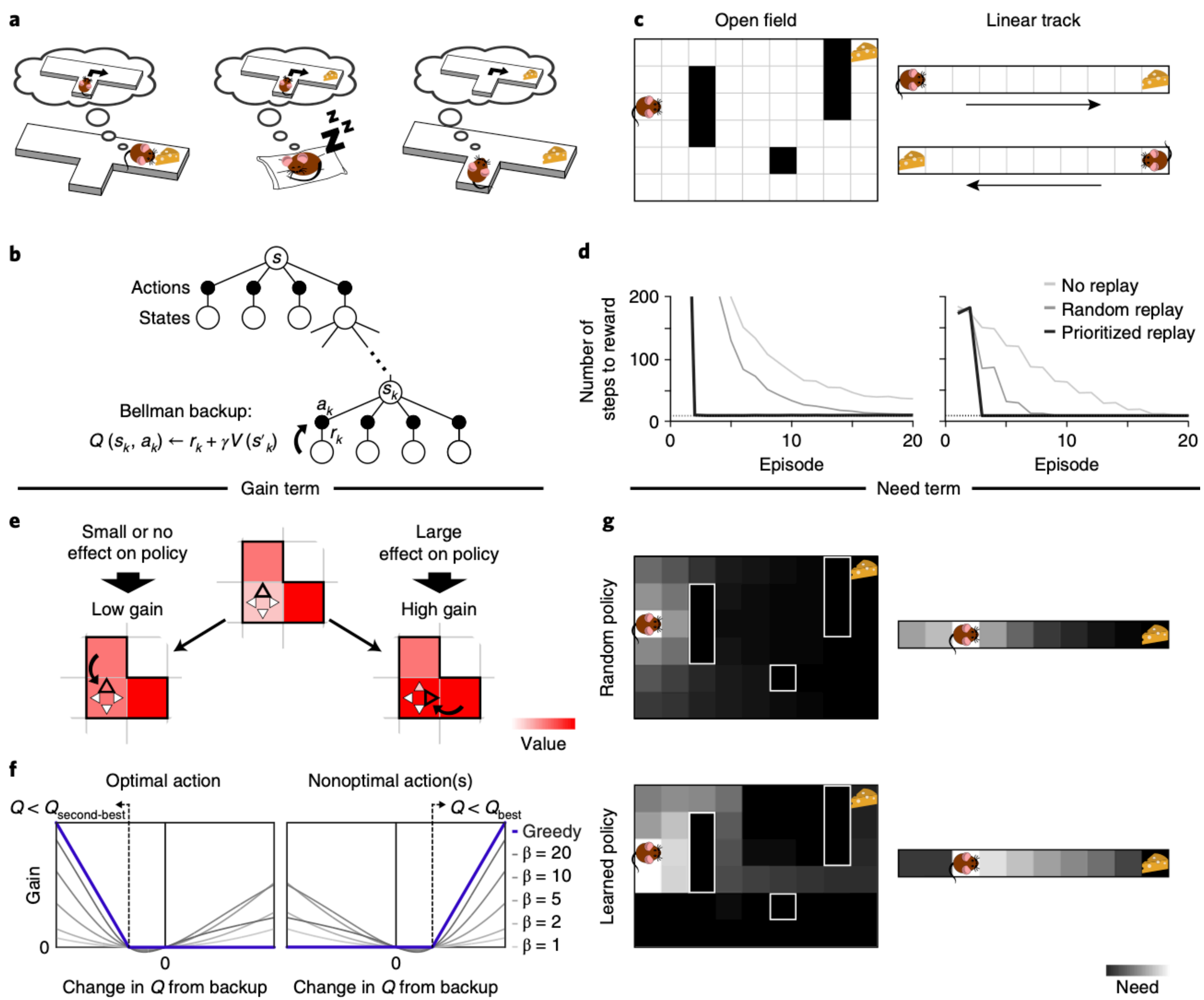
Prioritised replay: assigns priorities to the experiences based on their importance or learning potential.



Mattar and Daw. *Nature*. (2018)

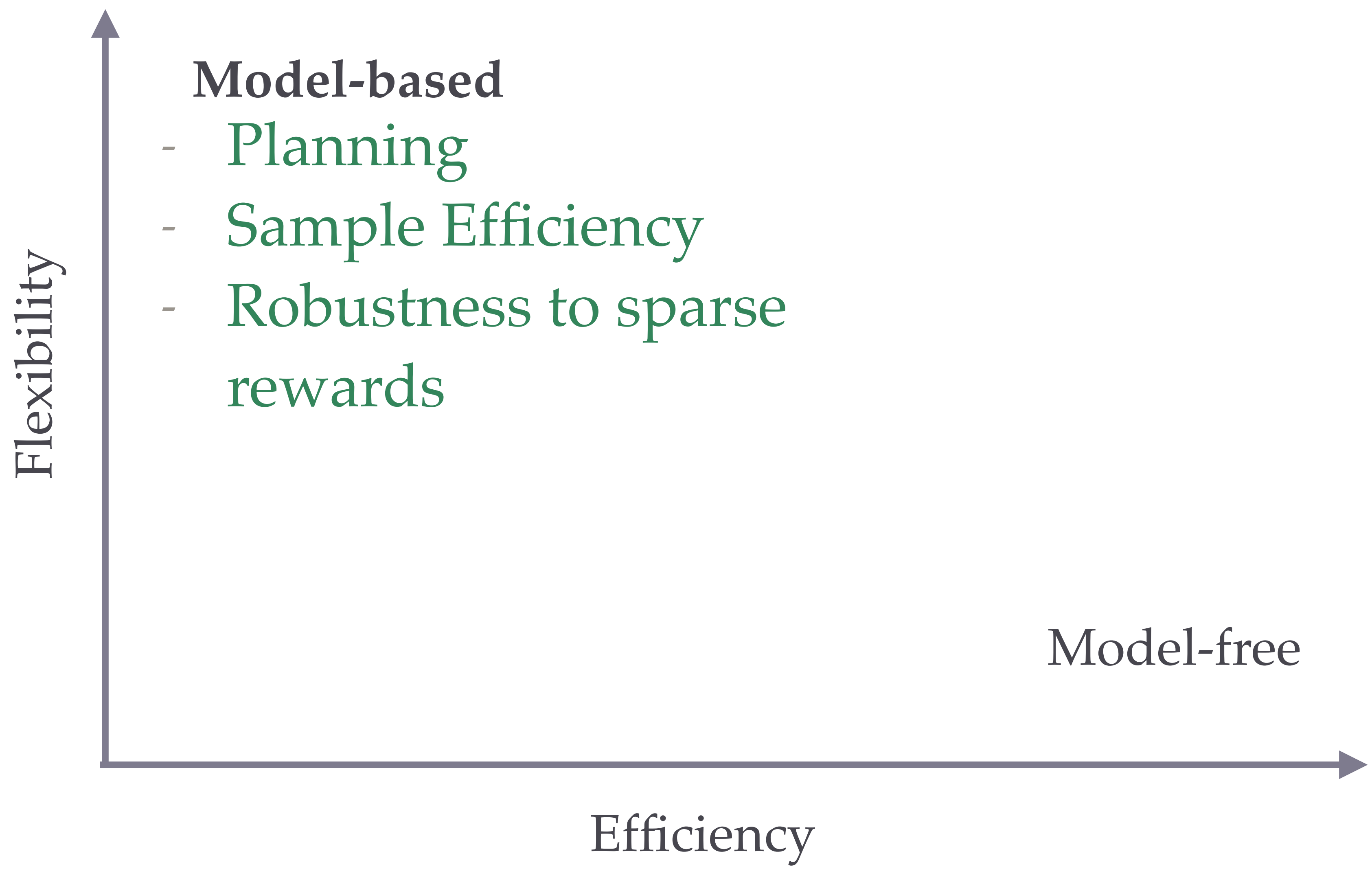
DYNA

Prioritised replay: assigns priorities to the experiences based on their importance or learning potential.

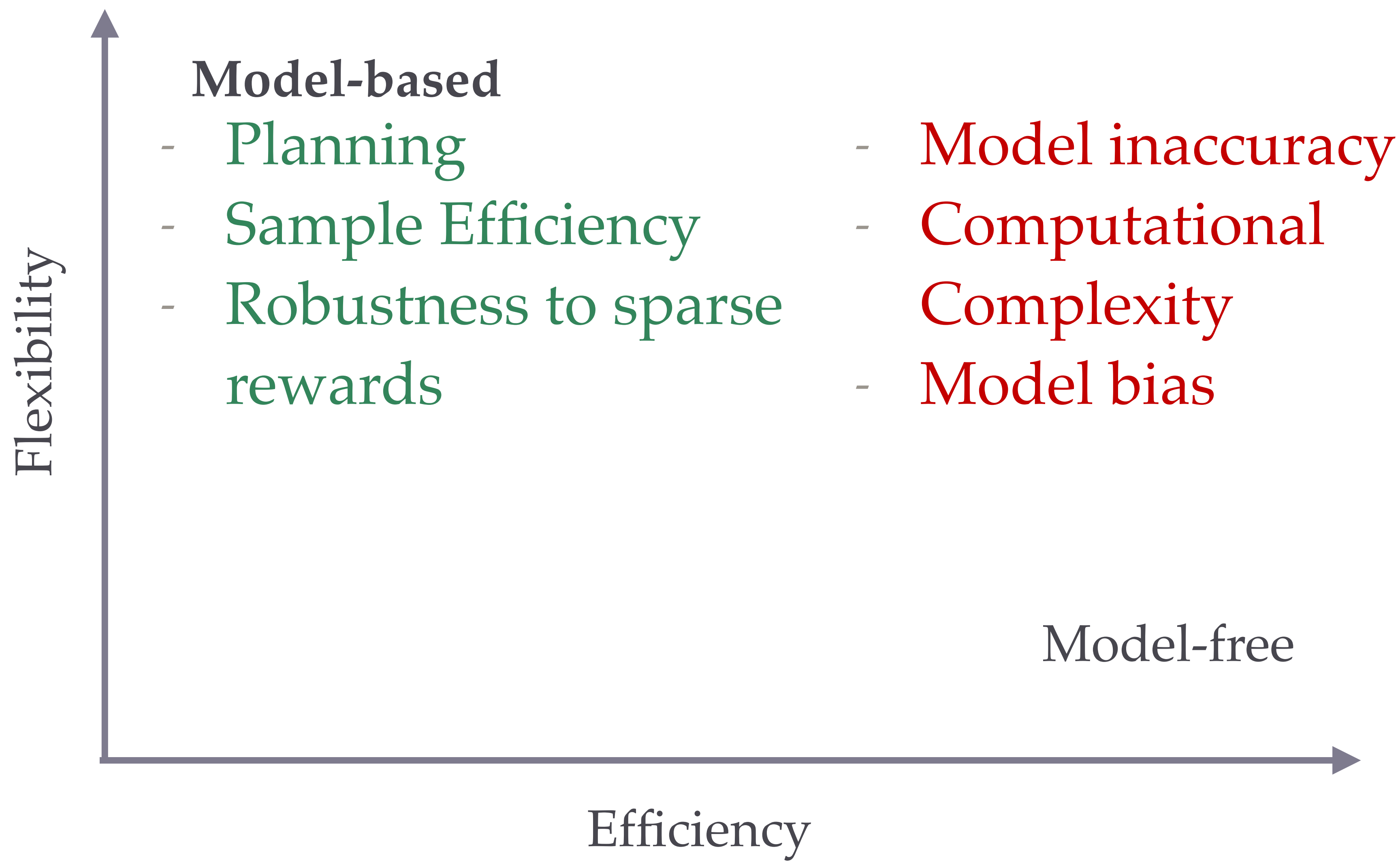


Mattar and Daw. *Nature*. (2018)

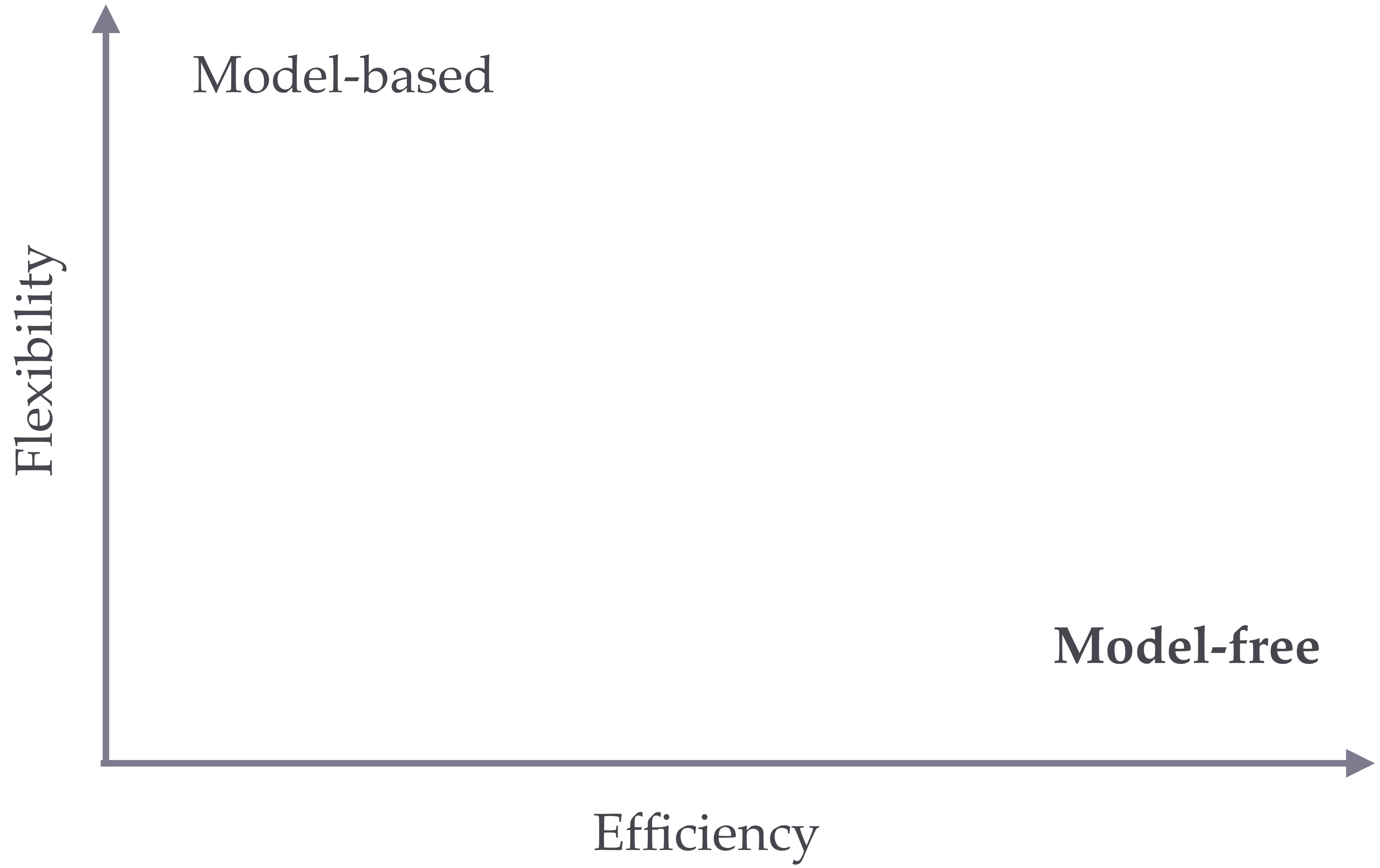
SUMMARIZED SUMMARY



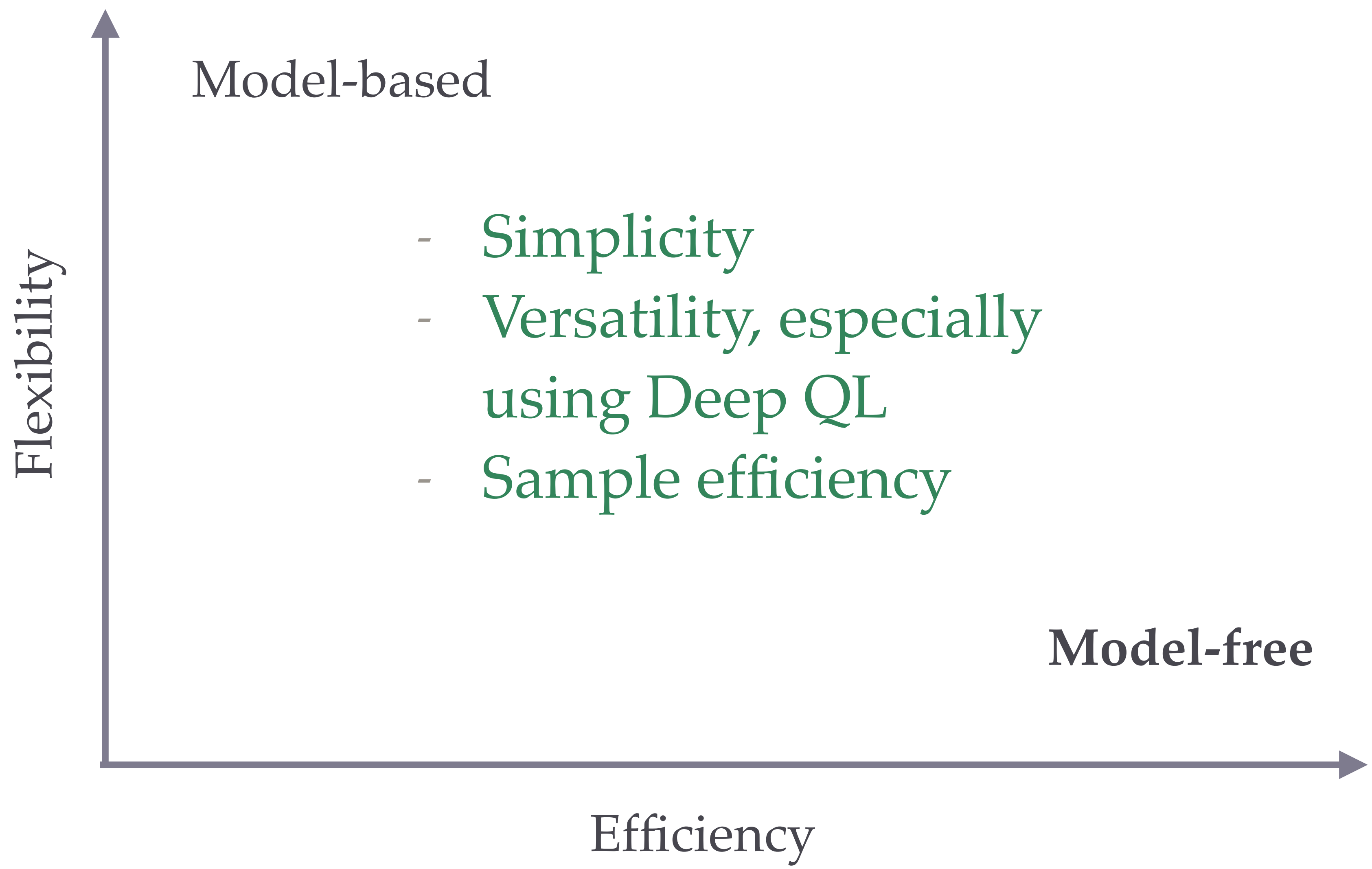
SUMMARIZED SUMMARY



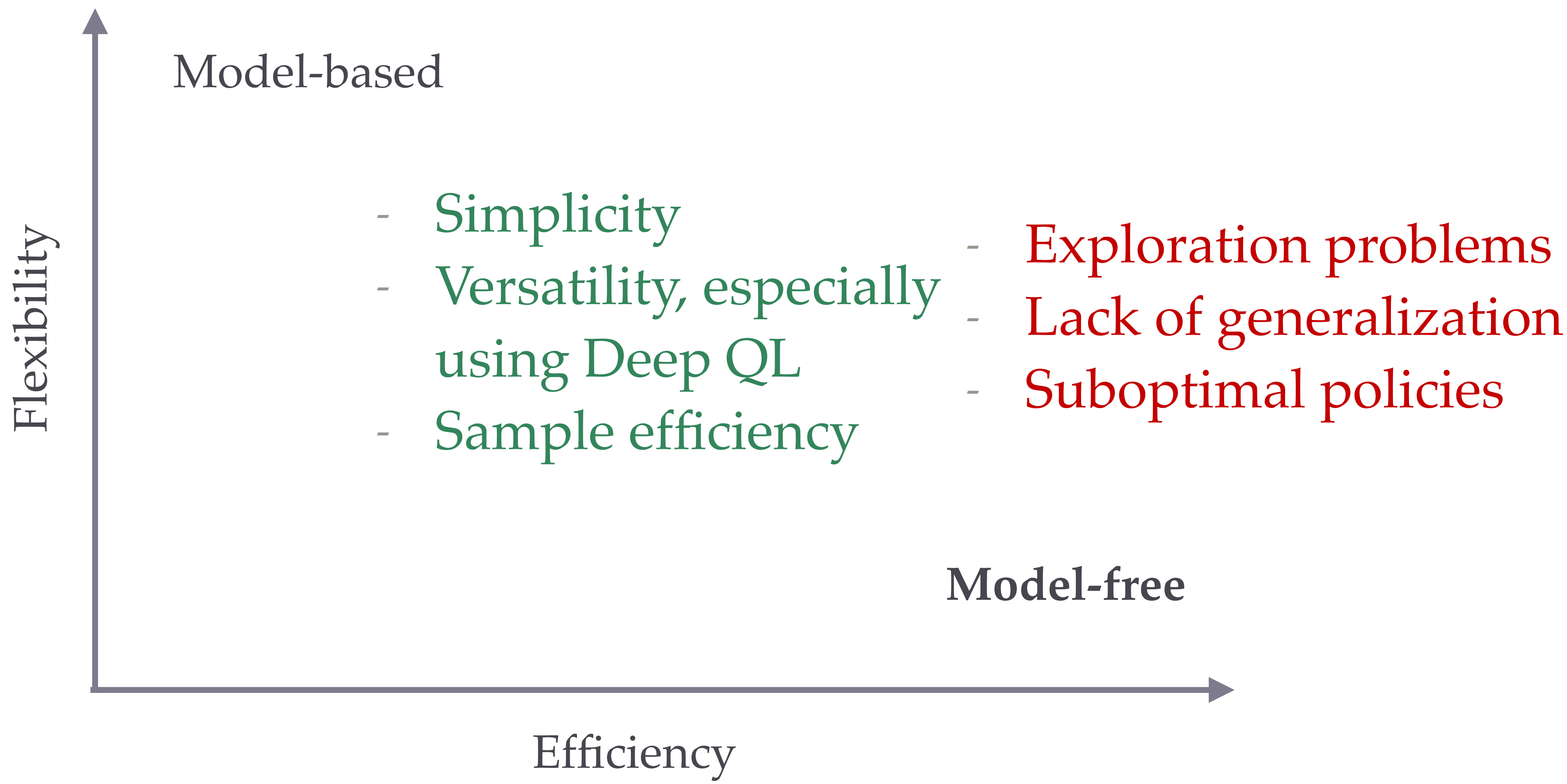
SUMMARIZED SUMMARY



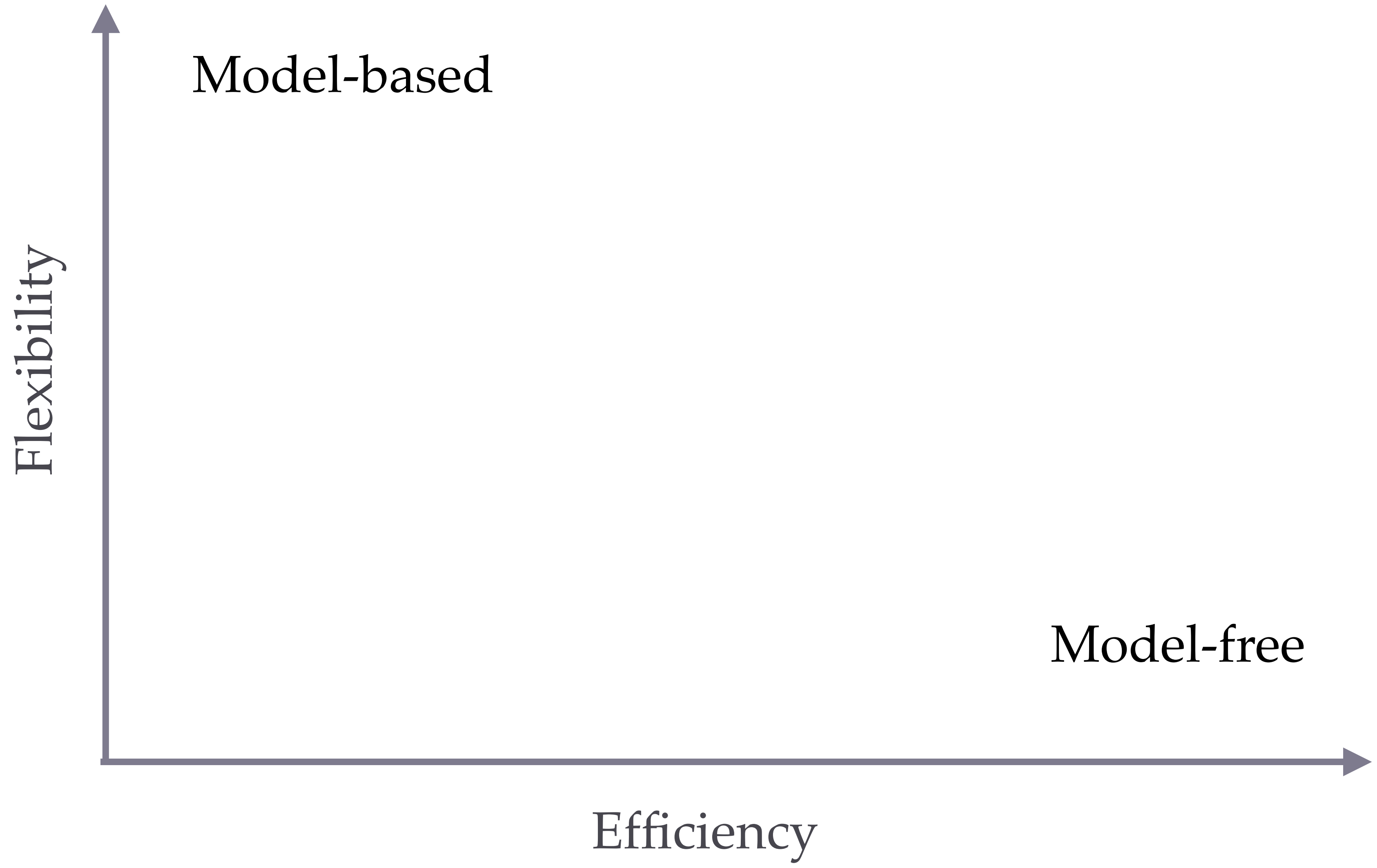
SUMMARIZED SUMMARY



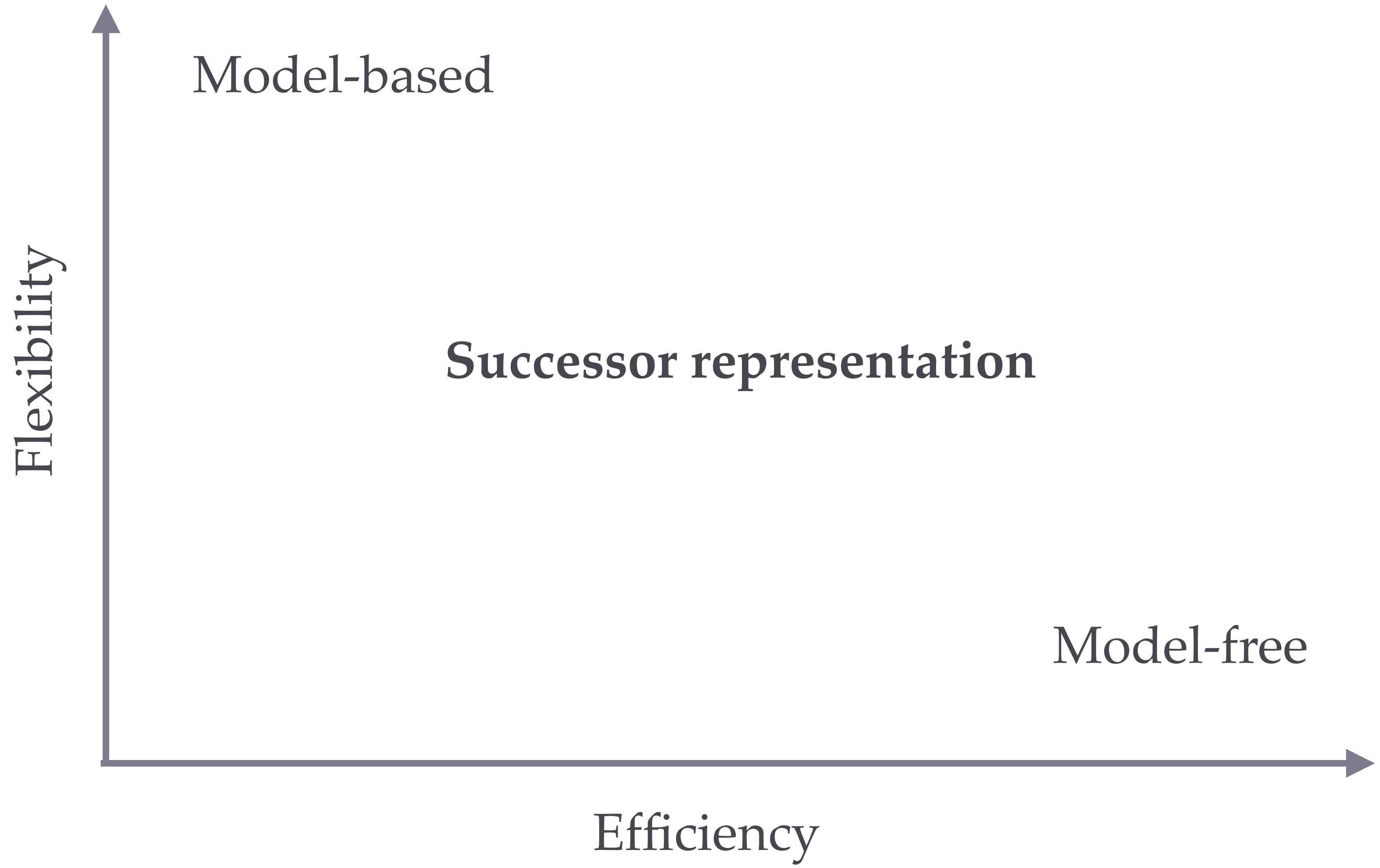
SUMMARIZED SUMMARY



SUMMARIZED SUMMARY



THERE IS SOMETHING IN BETWEEN



SUCCESSOR REPRESENTATION: PREDICTED DISCOUNTED SUM OF STATE OCCUPANCY

$$V_P(s) = \sum_{t=0}^{\infty} \gamma^t P_\pi(s|s_t) r(s_t).$$

$$V_P = \sum_{t=0}^{\infty} \gamma^t P_\pi^t r = (\mathbb{1}_N - \gamma P_\pi)^{-1} r.$$

$$M = \sum_{k=0}^{\infty} \gamma^k P^k$$

M: Successor representation

$$M = (\mathbb{1}_N - \gamma P_\pi)^{-1}$$

PLANNING WITH THE SR IS MORE EFFICIENT

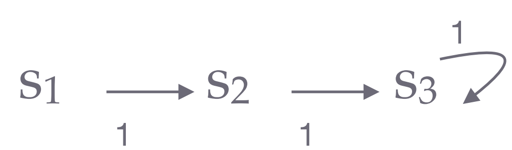
$$M = \sum_{k=0}^{\infty} \gamma^k P^k$$

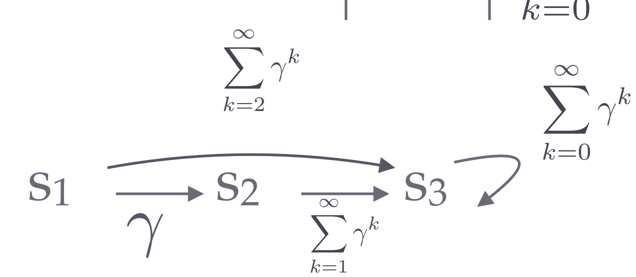
Model-based

Successor Representation

$$P \quad \begin{array}{c|ccc} & s_1 & s_2 & s_3 \\ \hline s_1 & 0 & 1 & 0 \\ s_2 & 0 & 0 & 1 \\ s_3 & 0 & 0 & 1 \end{array}$$

$$M \quad \begin{array}{c|ccc} & s_1 & s_2 & s_3 \\ \hline s_1 & 1 & \gamma & \sum_{k=2}^{\infty} \gamma^k \\ s_2 & 0 & 1 & \sum_{k=1}^{\infty} \gamma^k \\ s_3 & 0 & 0 & \sum_{k=0}^{\infty} \gamma^k \end{array}$$





PLANNING WITH THE SR IS MORE EFFICIENT

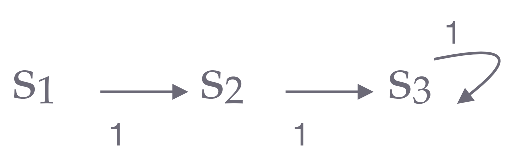
$$M = \sum_{k=0}^{\infty} \gamma^k P^k$$

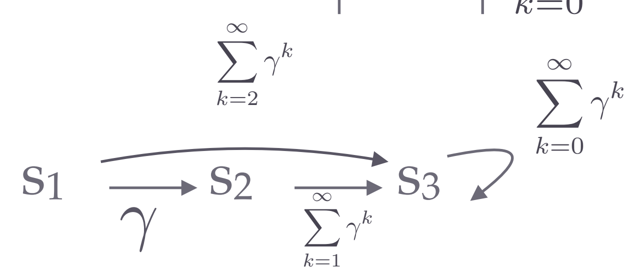
Model-based

Successor Representation

$$P \quad \begin{array}{c|ccc} & s_1 & s_2 & s_3 \\ \hline s_1 & 0 & 1 & 0 \\ s_2 & 0 & 0 & 1 \\ s_3 & 0 & 0 & 1 \end{array}$$

$$M \quad \begin{array}{c|ccc} & s_1 & s_2 & s_3 \\ \hline s_1 & 1 & \gamma & \sum_{k=2}^{\infty} \gamma^k \\ s_2 & 0 & 1 & \sum_{k=1}^{\infty} \gamma^k \\ s_3 & 0 & 0 & \sum_{k=0}^{\infty} \gamma^k \end{array}$$





- Link one state to another according to “how many discounted times the agent can expect to visit state 2 from its current state”

PLANNING WITH THE SR IS MORE EFFICIENT

$$M = \sum_{k=0}^{\infty} \gamma^k P^k$$

Model-based → Successor Representation

$$P \quad \begin{array}{c|ccc} & s_1 & s_2 & s_3 \\ \hline s_1 & 0 & 1 & 0 \\ s_2 & 0 & 0 & 1 \\ s_3 & 0 & 0 & 1 \end{array}
\qquad\qquad
M \quad \begin{array}{c|ccc} & s_1 & s_2 & s_3 \\ \hline s_1 & 1 & \gamma & \sum_{k=2}^{\infty} \gamma^k \\ s_2 & 0 & 1 & \sum_{k=1}^{\infty} \gamma^k \\ s_3 & 0 & 0 & \sum_{k=0}^{\infty} \gamma^k \end{array}$$

$$\begin{tikzcd} s_1 \arrow[r, "1"'] & s_2 \arrow[r, "1"'] & s_3 \arrow[loop right, "1"] \end{tikzcd}
\qquad\qquad
\begin{tikzcd} s_1 \arrow[r, "\gamma"'] \arrow[rr, bend left, "\sum_{k=2}^{\infty}\gamma^k"] & s_2 \arrow[r, "\sum_{k=1}^{\infty}\gamma^k"'] & s_3 \arrow[loop right, "\sum_{k=0}^{\infty}\gamma^k"] \end{tikzcd}$$

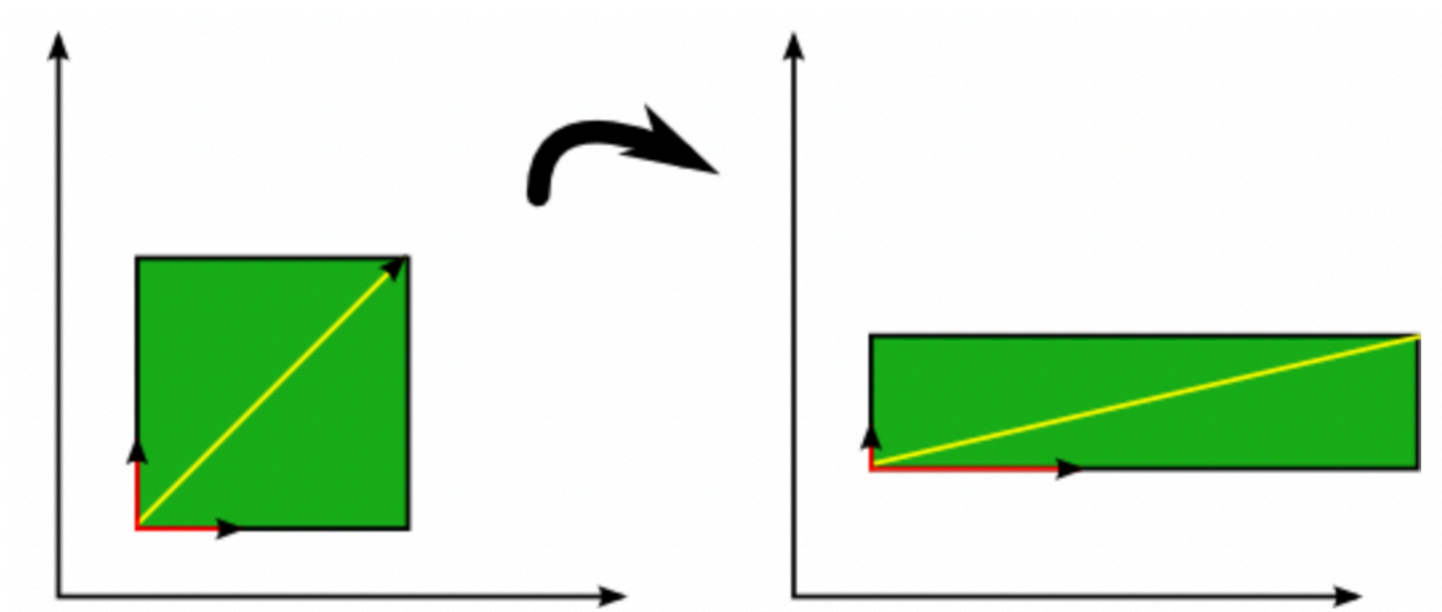
- Link one state to another according to "how many discounted times the agent can expect to visit state 2 from its current state"
- More efficient, as in this example: the agent already can access informations from s_3 while being in s_1.

INTERMEDIARY : EIGENVECTORS

- Eigenvectors: vectors that do not change direction when applying a linear transformation

$$\boldsymbol{A} \cdot \boldsymbol{v} = \lambda \cdot \boldsymbol{v}$$

$$(\boldsymbol{A} - \lambda \cdot I) \cdot \boldsymbol{v} = \vec{0}$$



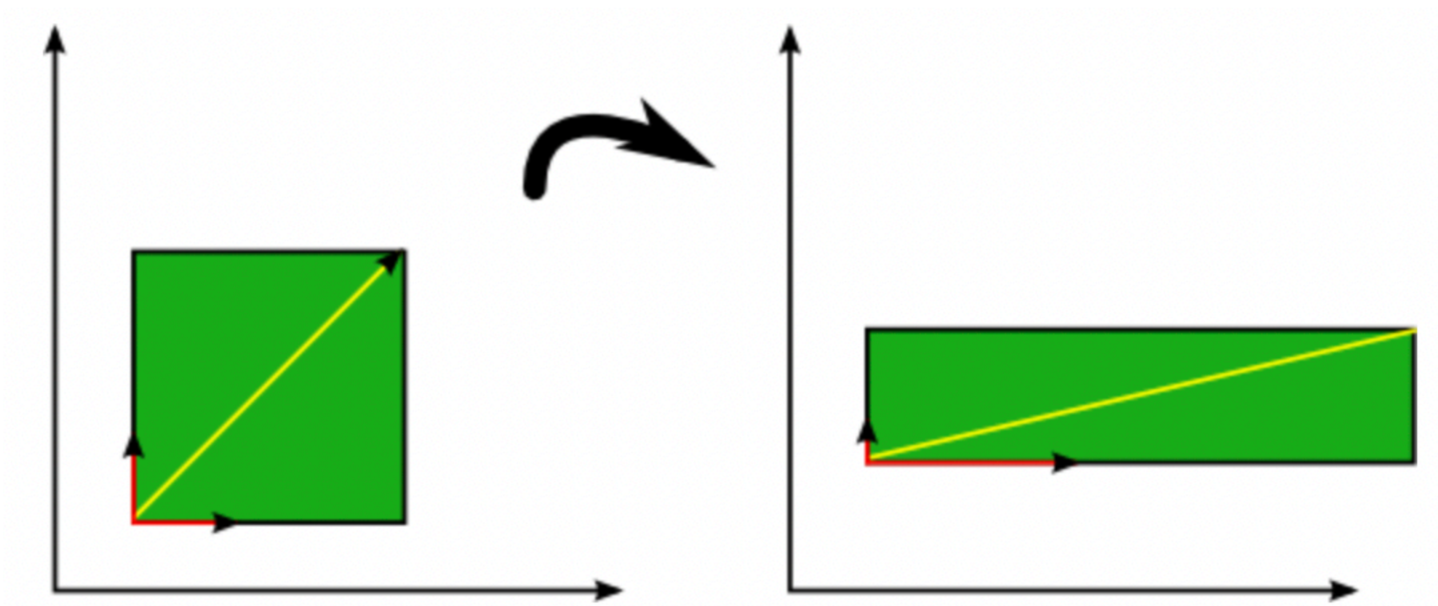
$$\boldsymbol{I} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ & & \dots & & \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}$$

INTERMEDIARY : EIGENVECTORS

- v is an “eigenvector”
- Lambda is an “eigenvalue”
- Eigenvectors capture direction of main actions of the multiplication by A
- Eigenvalues capture how much this direction
    - gets extended (if lambda >1)
    - or squeezed (if lambda <1)

$$\boldsymbol{A} \cdot \boldsymbol{v} = \lambda \cdot \boldsymbol{v}$$

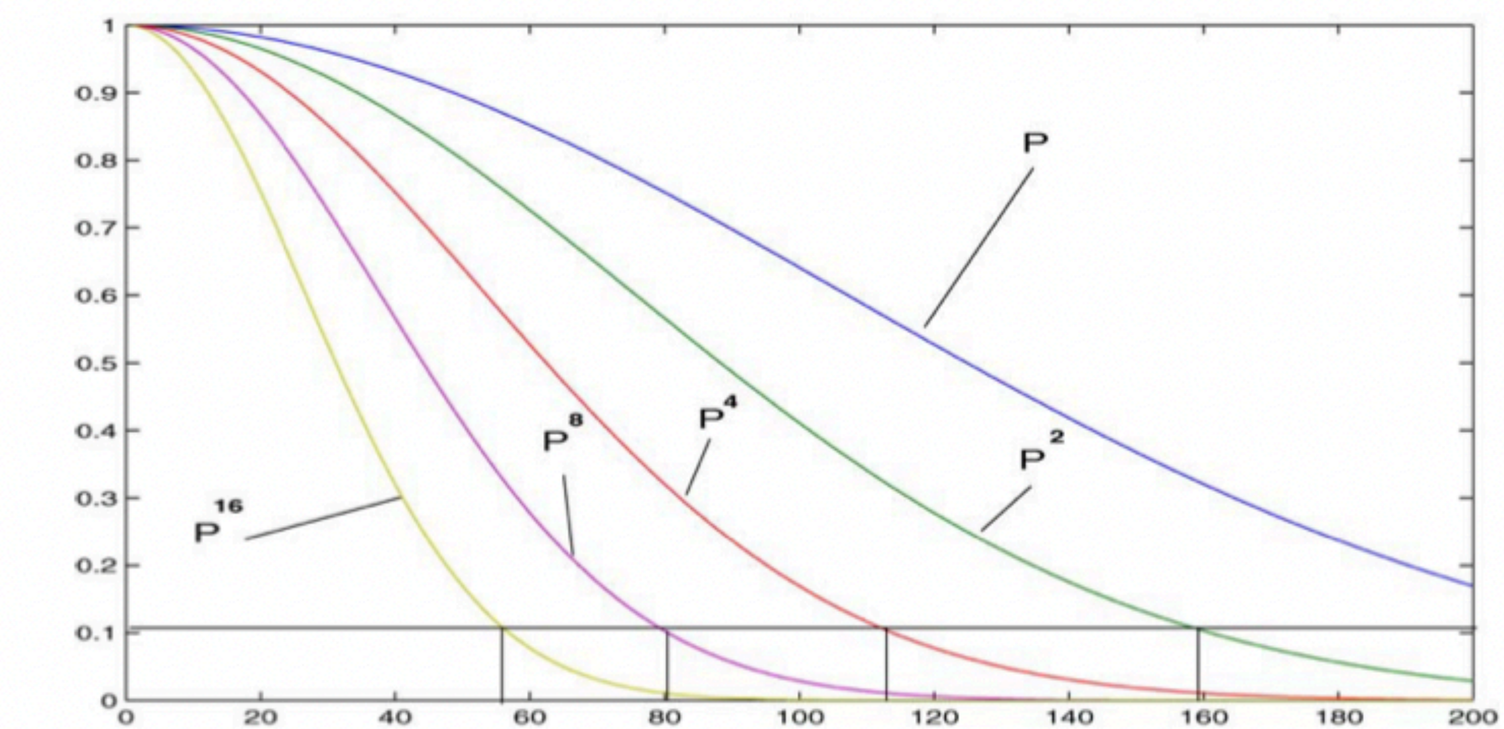
$$(\boldsymbol{A} - \lambda \cdot I) \cdot \boldsymbol{v} = \vec{0}$$



$$\boldsymbol{I} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ & & \dots & & \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}$$

SR: A DESCRIPTION OF THE STATE OCCUPANCY AT DIFFERENT TIMESCALES

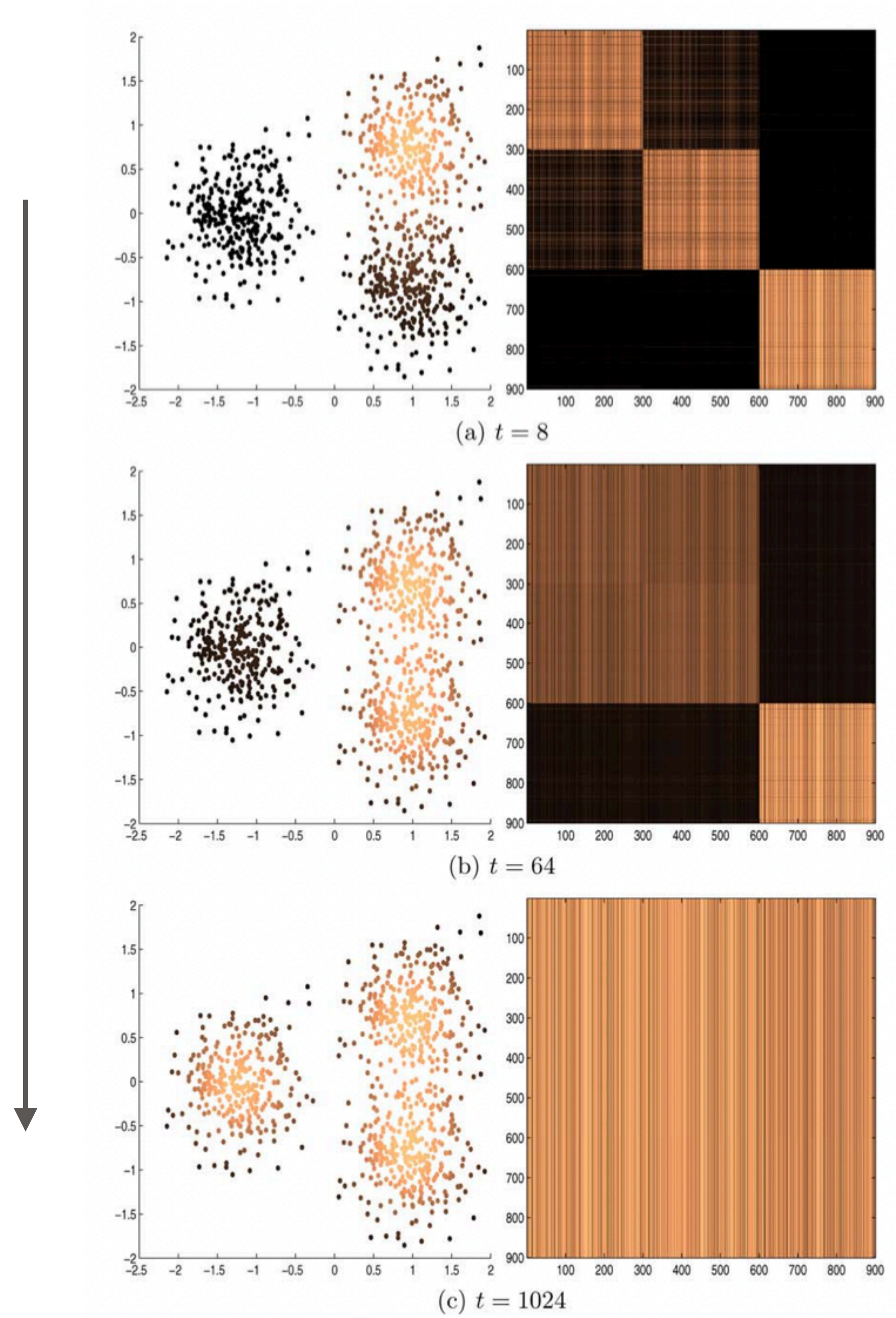
The numerical rank of P^k (=the "dimension of the number of directions" that it represents) decreases with time:



Spectral decomposition of the SR= eigenvalues

P^t represents the states occupancy description at time t = after t timesteps in the Markov chain

SR: A DESCRIPTION OF THE STATE OCCUPANCY AT DIFFERENT TIMESCALES



(a) $t = 8$

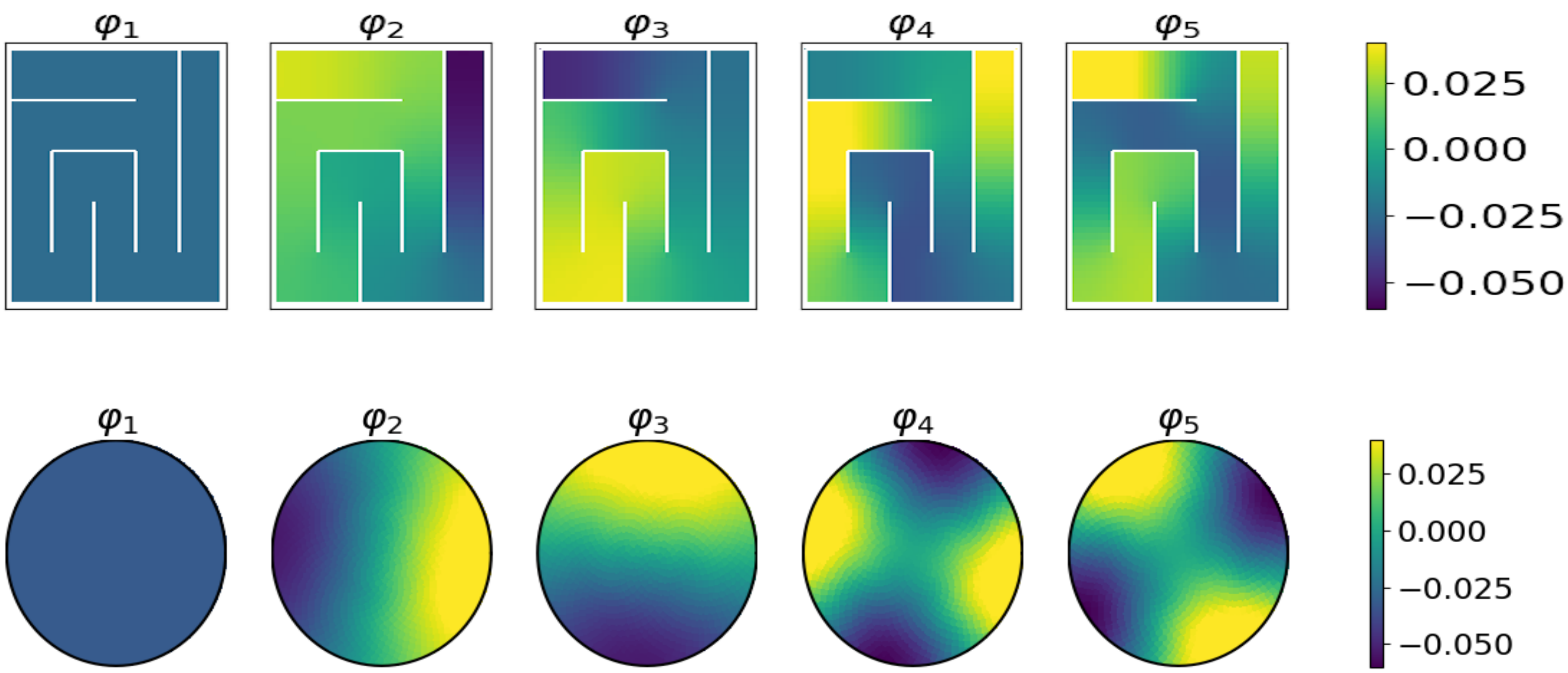
(b) $t = 64$

(c) $t = 1024$

As time evolve, the graph of points progressively converges to the limit of state-occupancy

SR: THE EIGENVECTORS PROVIDE A FORM OF 'FREQUENCY' DISTRIBUTION OF THE STATE OCCUPANCY

$$\lambda_1 = 1 > \lambda_2 \ldots . \lambda_N \geq 0$$



One can cut the dimension with minimal predictive harms.

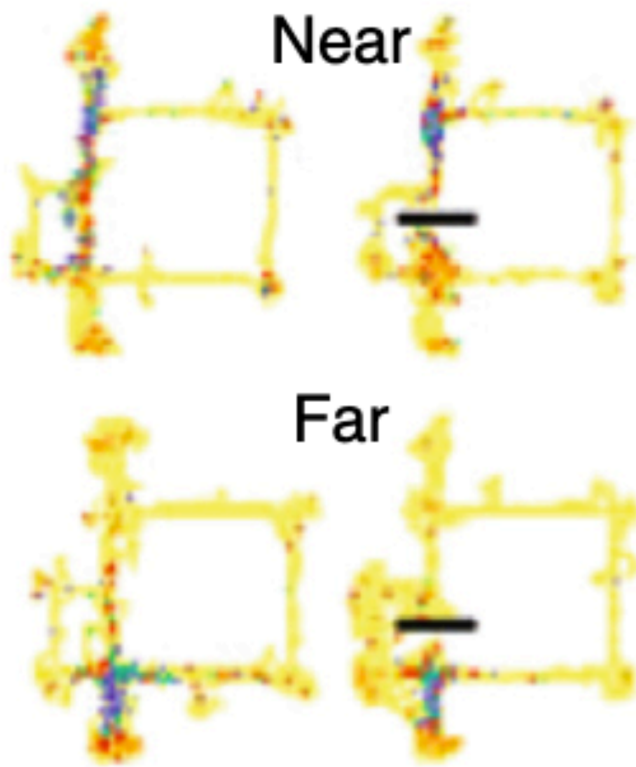
SR AND GAUSSIAN KERNEL: GENERALISATION MECHANISMS

- Gaussian kernels link two states according to how far away they are.
  - Usually, this distance is defined from the Euclidean distance
- In the SR representation, it is similar but it is about ‘accessible distance’:
  - Depends on the transition in the environment and the policy of the agent

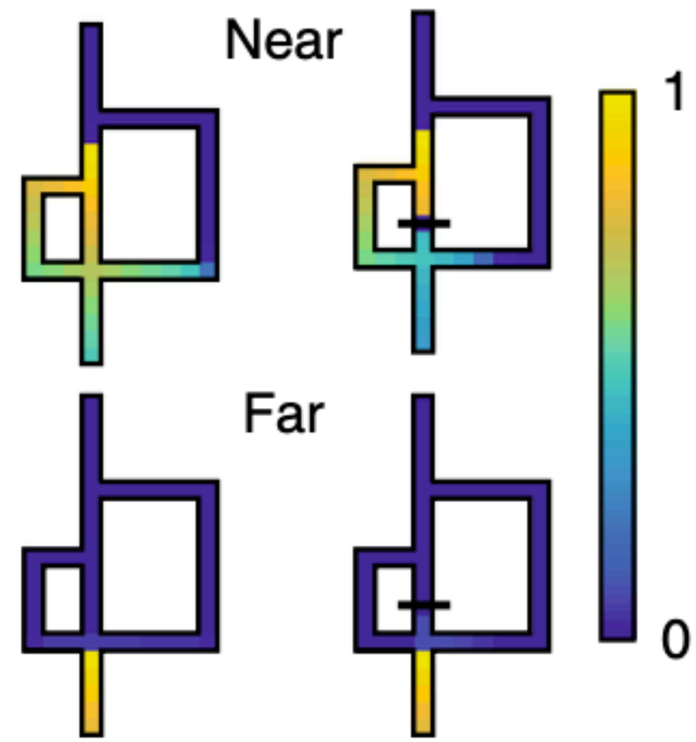
SR: CAN REPRODUCE PLACE CELLS 'REMAPPING'

Data- place field

(Alvernhe et al., *Eur. J. Neurosci.*,2011)



SR- place field



- When a new barrier is introduced, the transition probability changes
- Both hippocampal place cells and SR column vectors adjust to that change in a similar way
- This lead to a new model of the hippocampus as a 'predictive map'

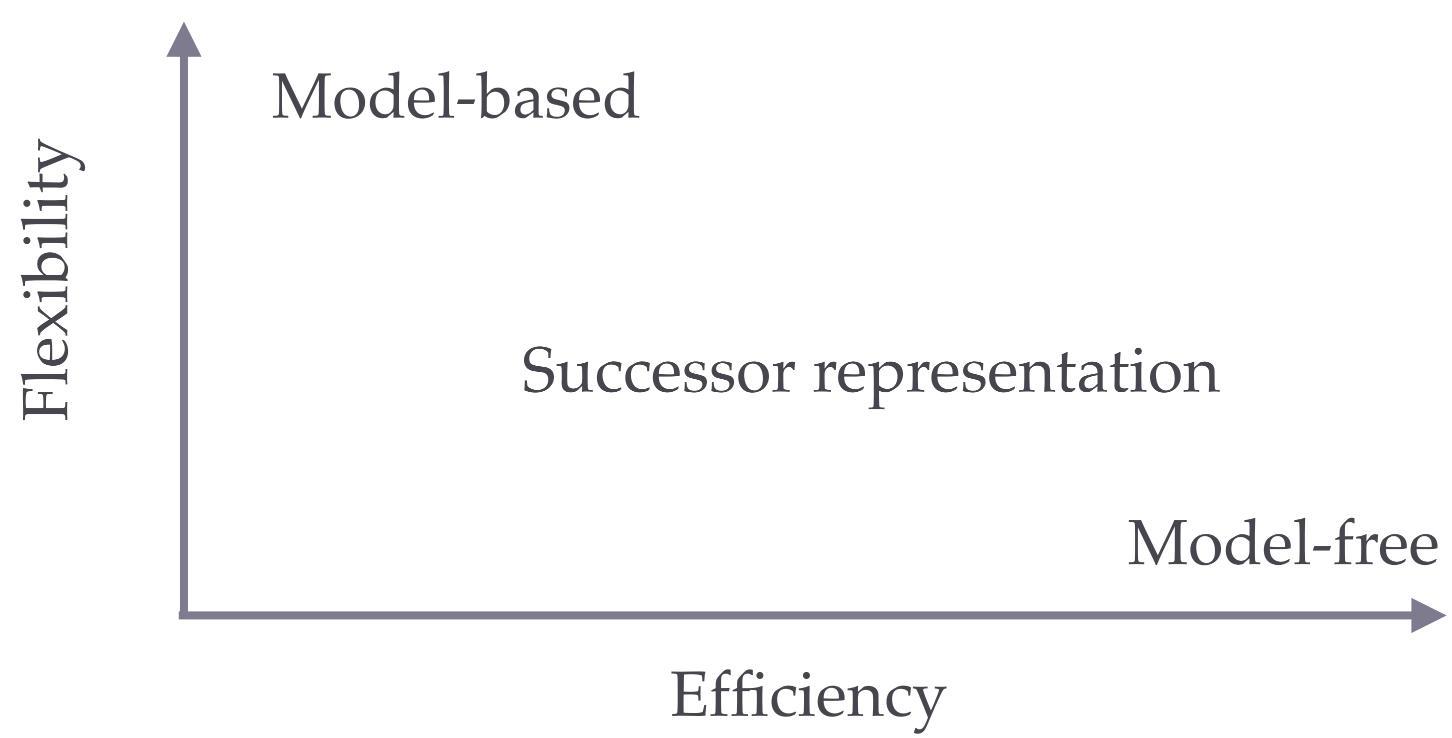
Stachenfeld, Botvinick and Gershman, *Nat. Neurosci.*, 2017

SUMMARY SR

- Links state, space and time
- Can be learned using TD error:

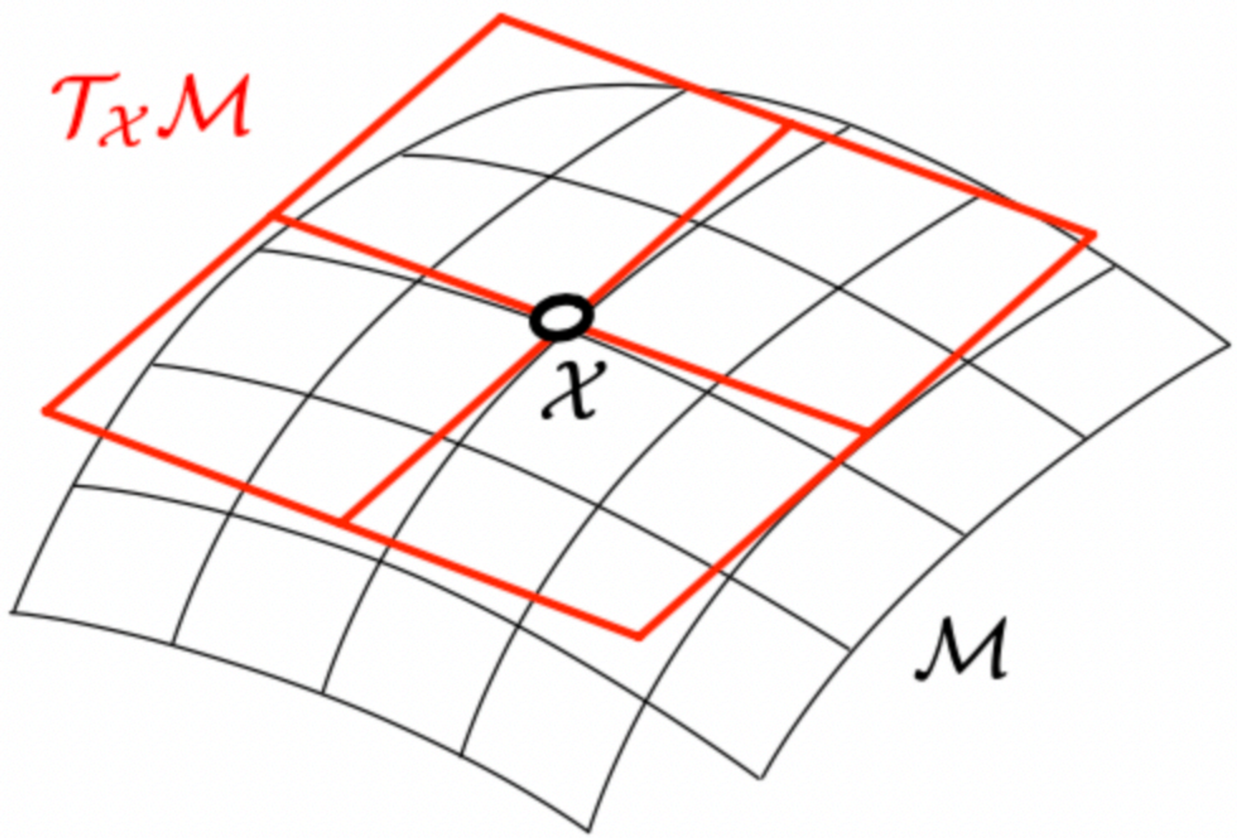
$$\delta_t(s) = \mathcal{I}(s_t, s) + \gamma \hat{M}(s_{t+1}, s) - \hat{M}(s_t, s)$$

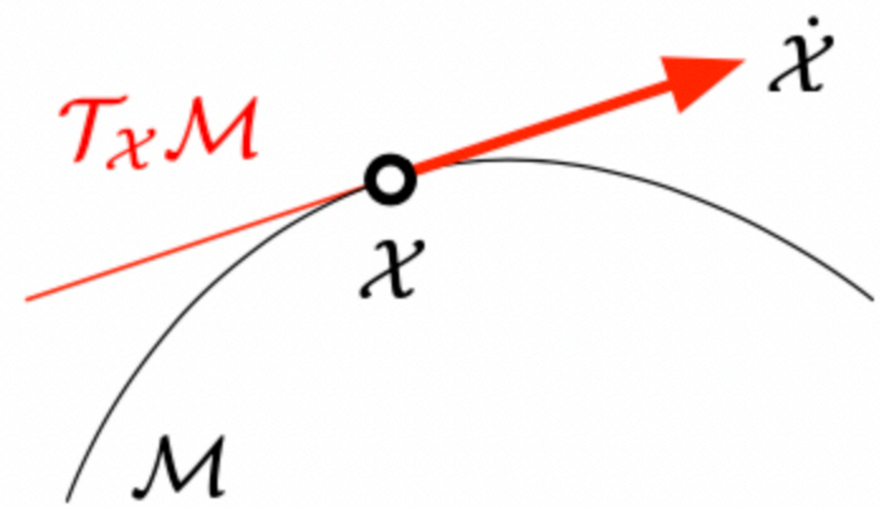
- Can explain the evolution of neural coding based on environmental changes
- Is more flexible than model-free but more efficient than model-based approaches.



NON-RL APPROACHES TO DECIPHERING THE NEURAL CODE

MANIFOLD:
TOPOLOGICAL SPACE THAT LOCALLY RESEMBLES AN EUCLIDEAN SPACE



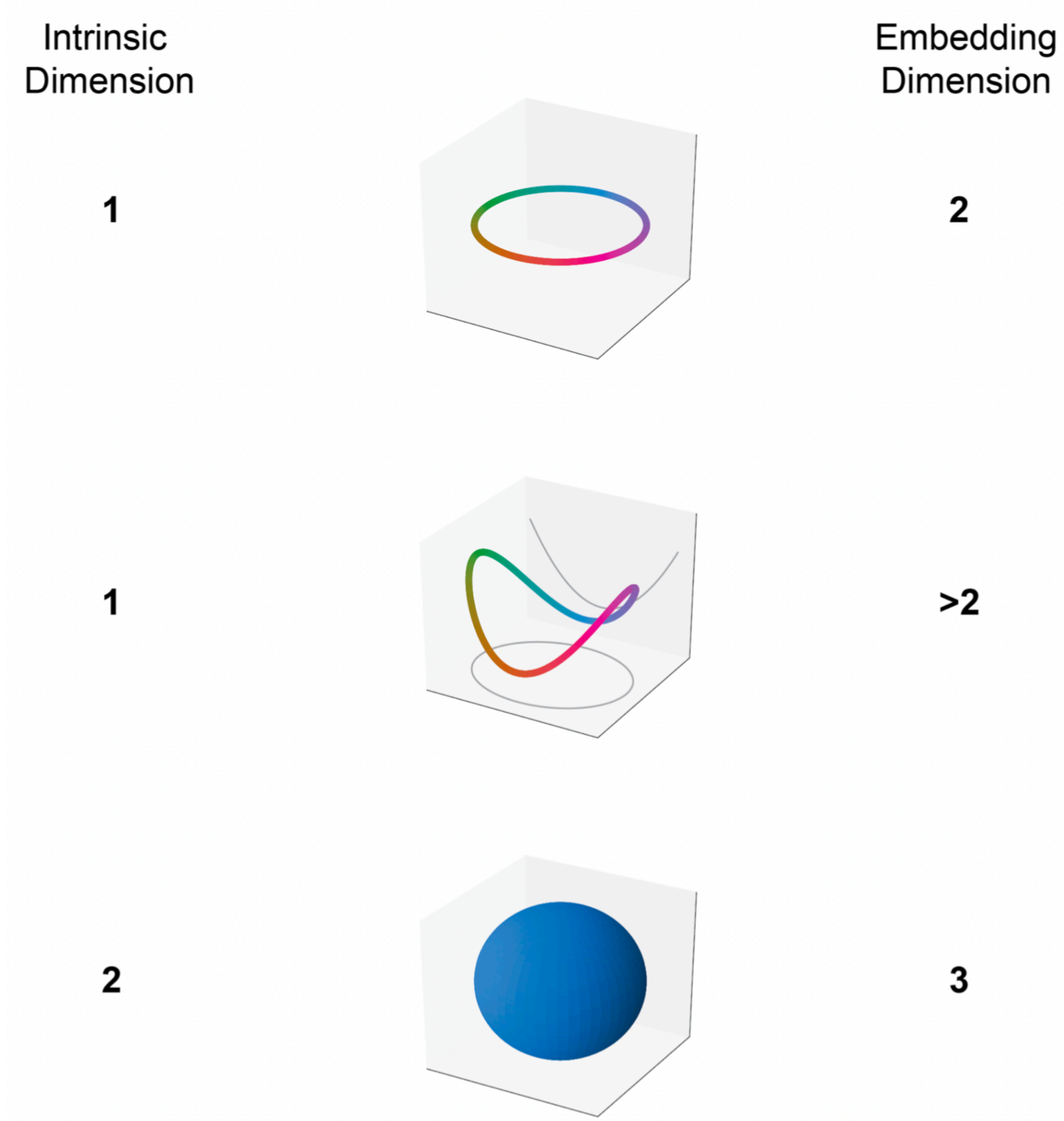


- Example: Sphere
- Allow more complicated structures to be expressed and understood in terms of simpler spaces

MANIFOLD ANALYSIS: TOOL FOR DIMENSIONALITY REDUCTION

Characterizing manifolds and their local euclidean approximation:

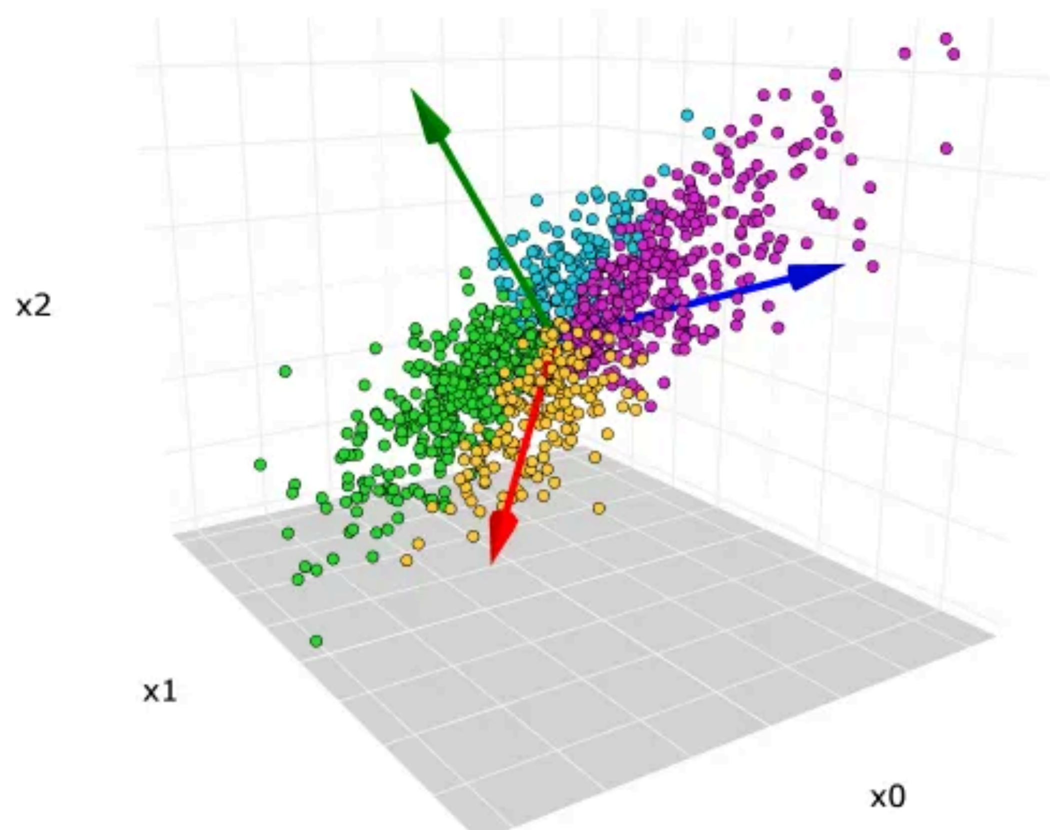
- gives information on behavior of objects in that space
- Enables dimensionality reduction

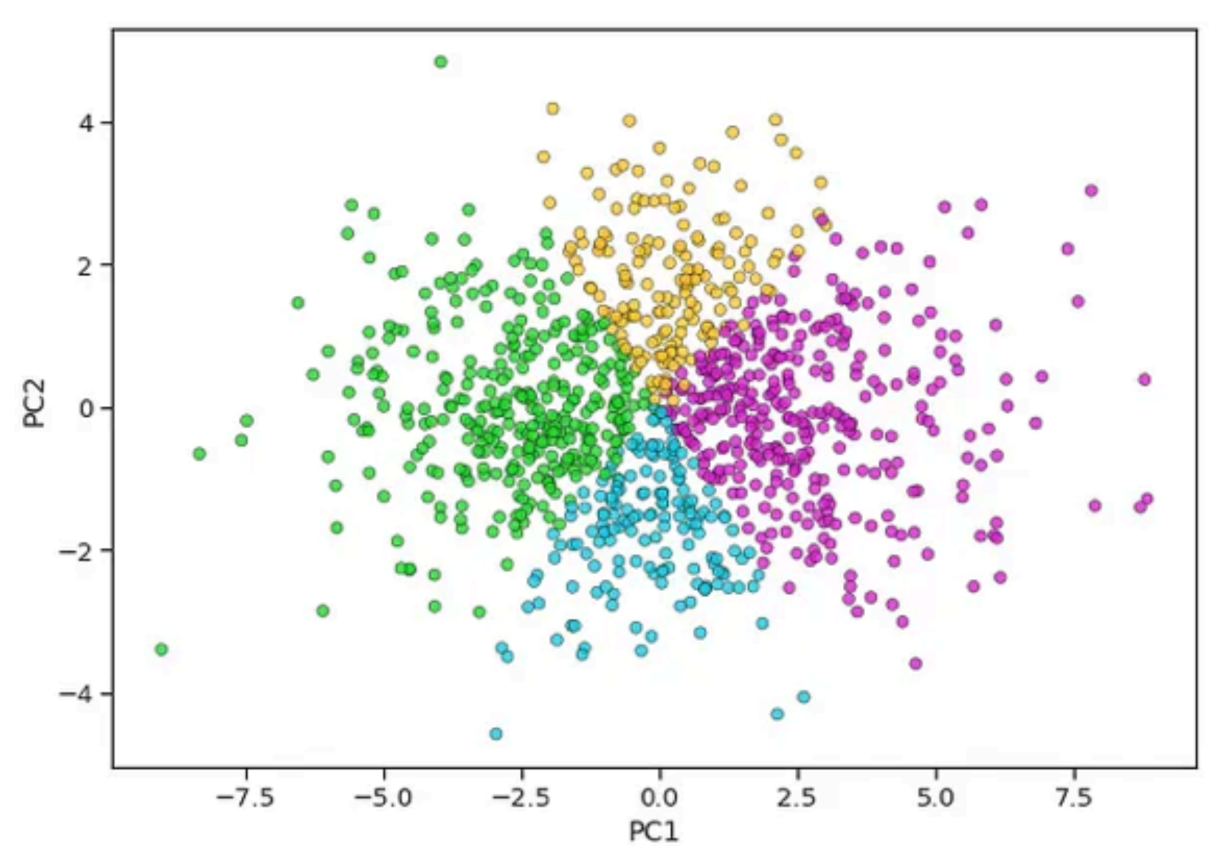


Jazayeri, and Ostojic. *Current Opinion in Neurobiology*. 2021.

MANIFOLD ANALYSIS EXAMPLE: PRINCIPAL COMPONENT ANALYSIS (PCA)

- Project data point into transformed dimensions
- Those dimensions are such that they maximise the variance of the dataset





MANIFOLD ANALYSIS EXAMPLE: PRINCIPAL COMPONENT ANALYSIS

- The axis correspond to eigenvectors of the covariance matrix of the data points:

$$\boldsymbol{C} = \frac{1}{n-1}\sum_{i=1}^{n}(\boldsymbol{X}_i - \bar{\boldsymbol{X}})^T(\boldsymbol{X}_i - \bar{\boldsymbol{X}})$$

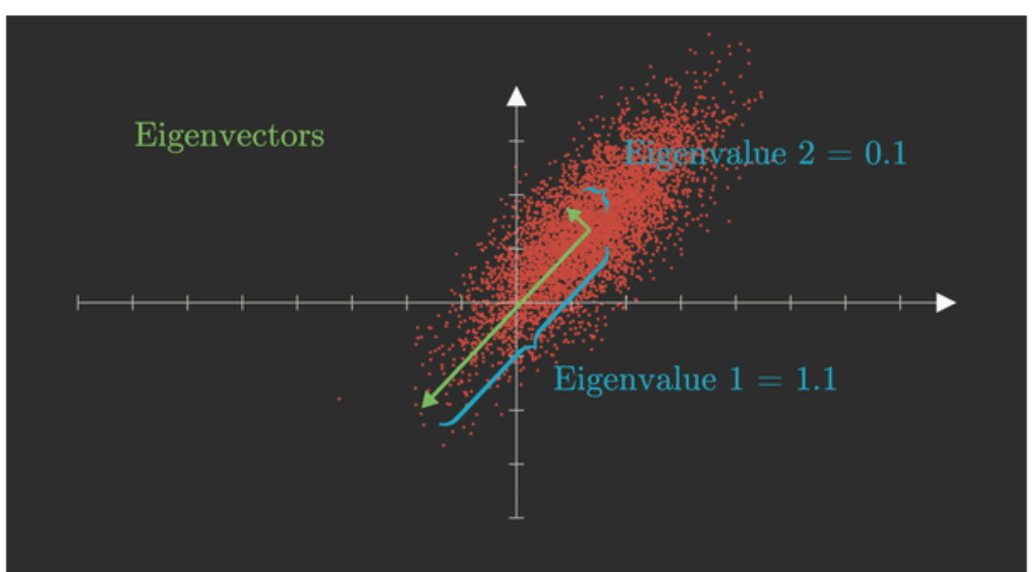
Variance of variable 1

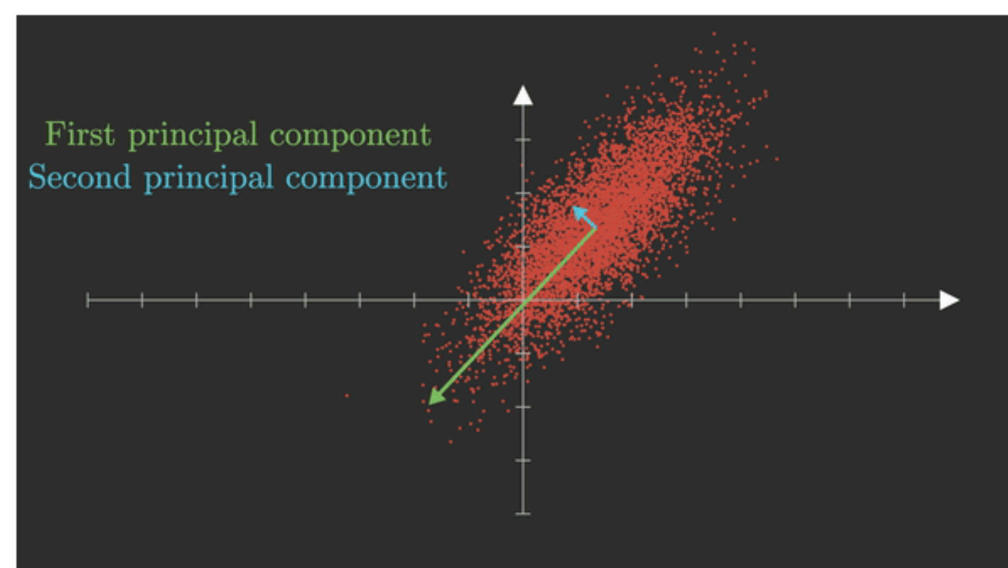
Covariance between var 1 and var 2

$$C = \begin{bmatrix} 0.4713125 & 0.375 \\ 0.375 & 0.5114 \end{bmatrix}$$

Covariance between var 1 and var 2

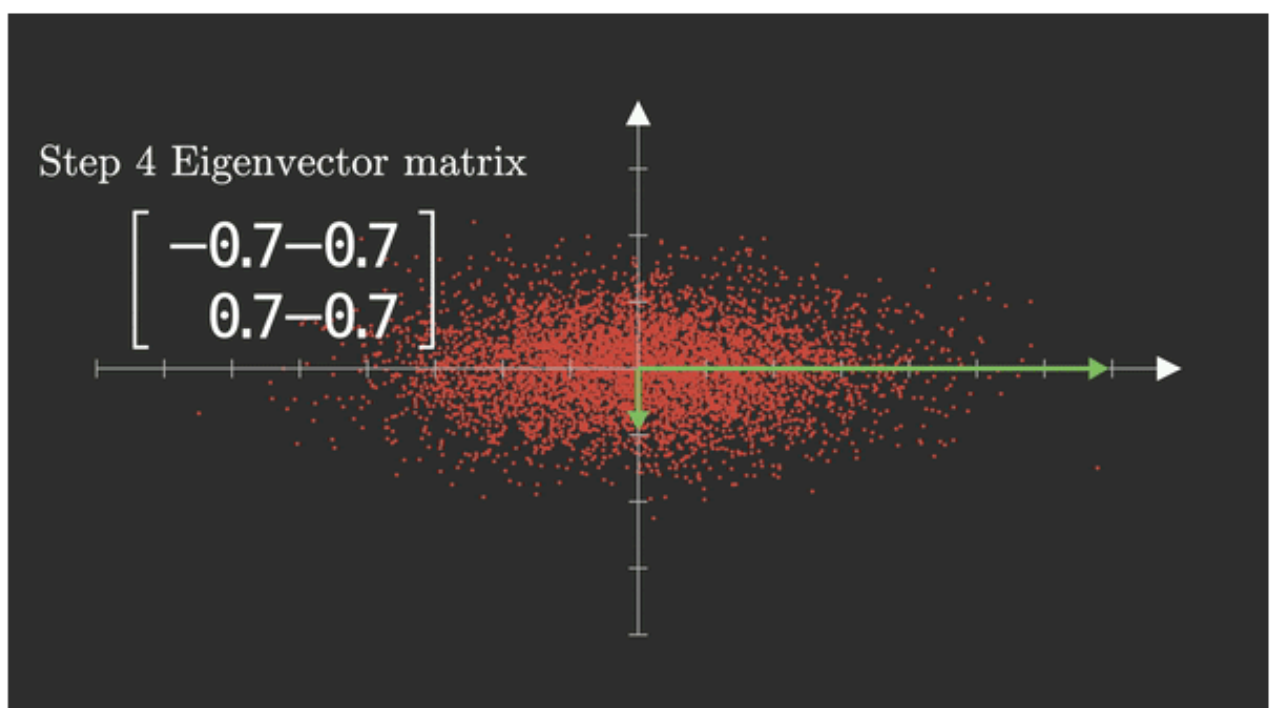
Variance of variable 2



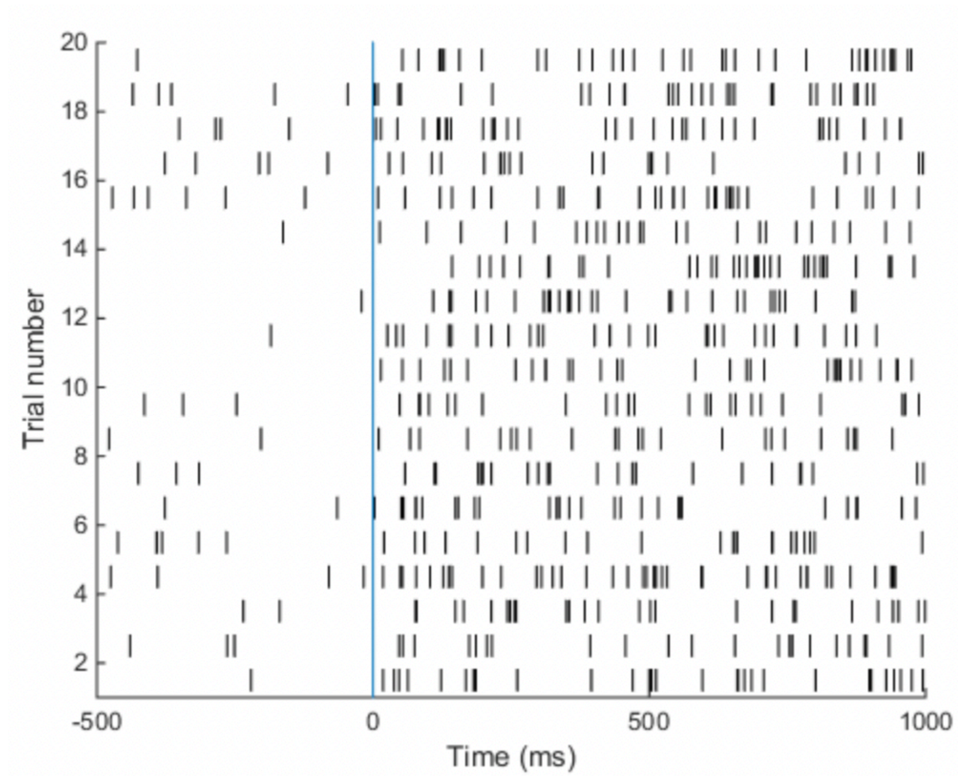


MANIFOLD ANALYSIS EXAMPLE: PRINCIPAL COMPONENT ANALYSIS (PCA)

- Eigenvectors take home message: “dimension of highest spread” of your data
- PCA take home message: “dimension of highest spread of the variance”



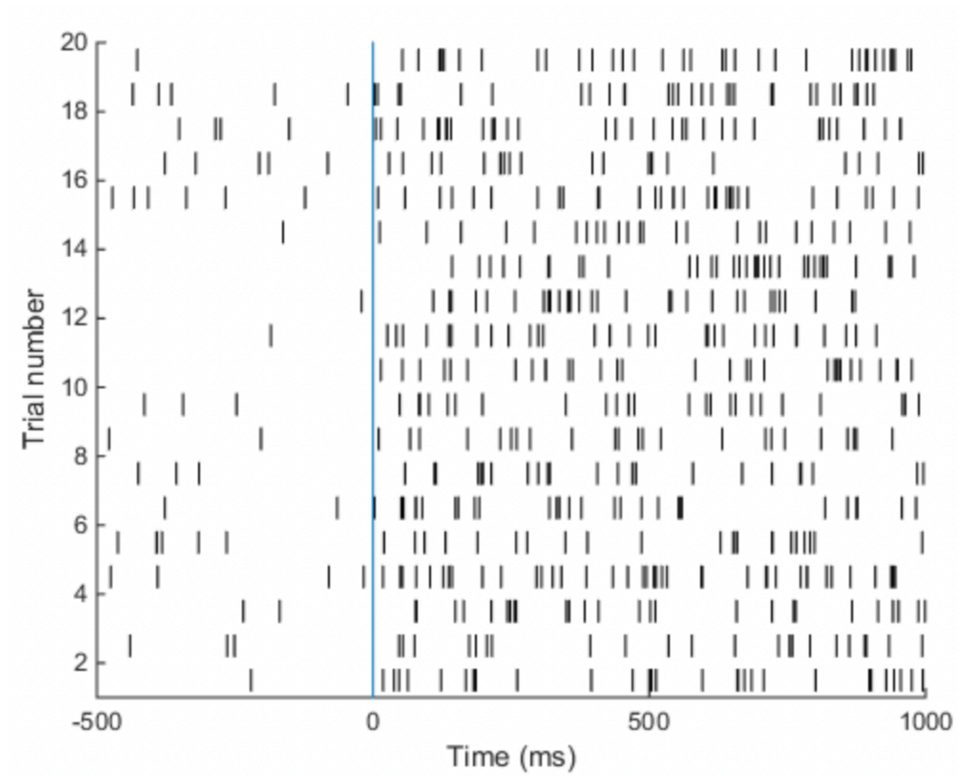
PCA TO INVESTIGATE THE MANIFOLDS OF NEURAL ACTIVITY



- Neural computations can be hidden at the level of single-neuron firing rates.

Urai, *et al. Nat. Neuro.*. 2022.

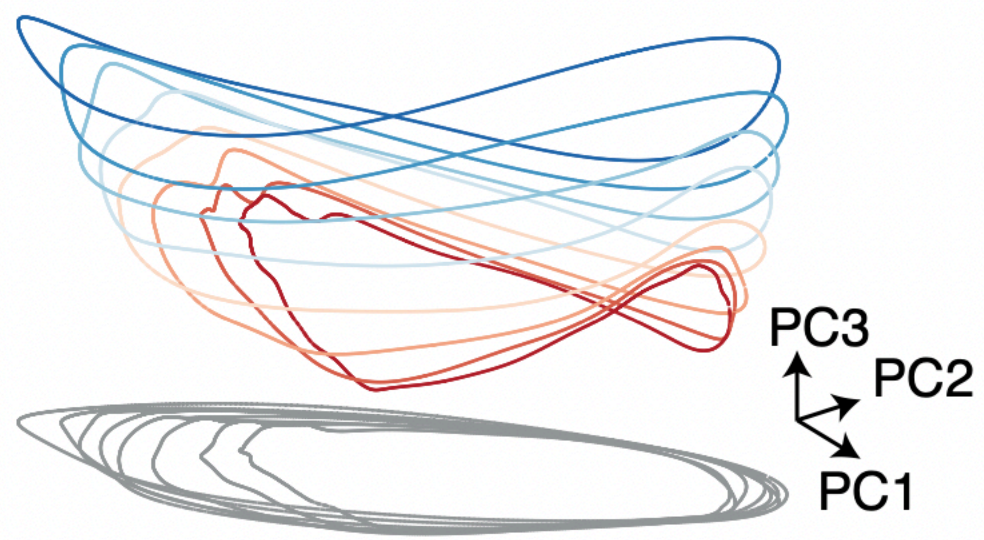
PCA TO INVESTIGATE THE MANIFOLDS OF NEURAL ACTIVITY



- Neural computations can be hidden at the level of single-neuron firing rates.
- Task-specific dynamics can be highlighted using PCA reduction.

Urai, *et al. Nat. Neuro.*. 2022.

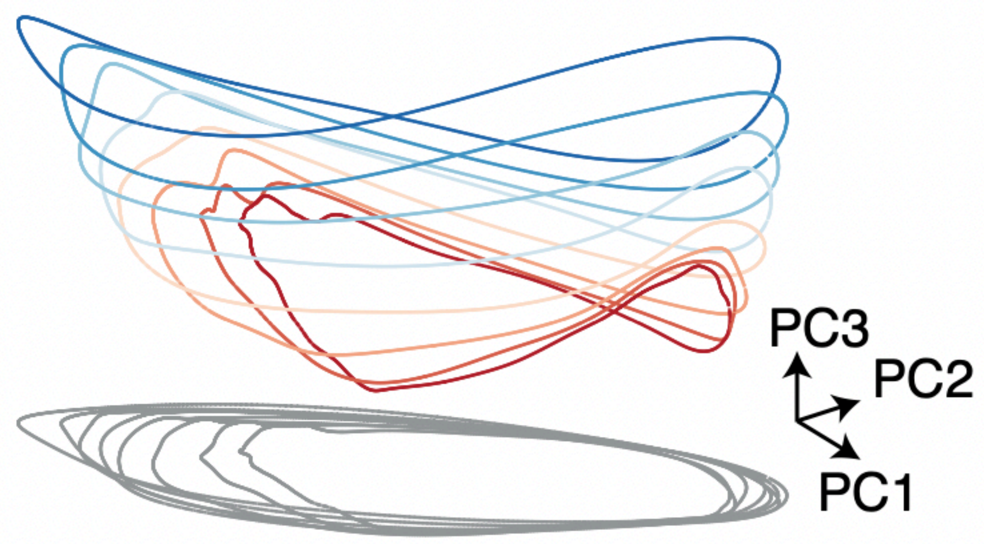
PCA TO INVESTIGATE THE MANIFOLDS OF NEURAL ACTIVITY



- Motor cortex recordings during cycling at different speed
- Picture shows neural responses:
  - each loop is once around a repeating cycle
  - blue is slowest.

Urai, *et al*. *Nat*. *Neuro*.. 2022.

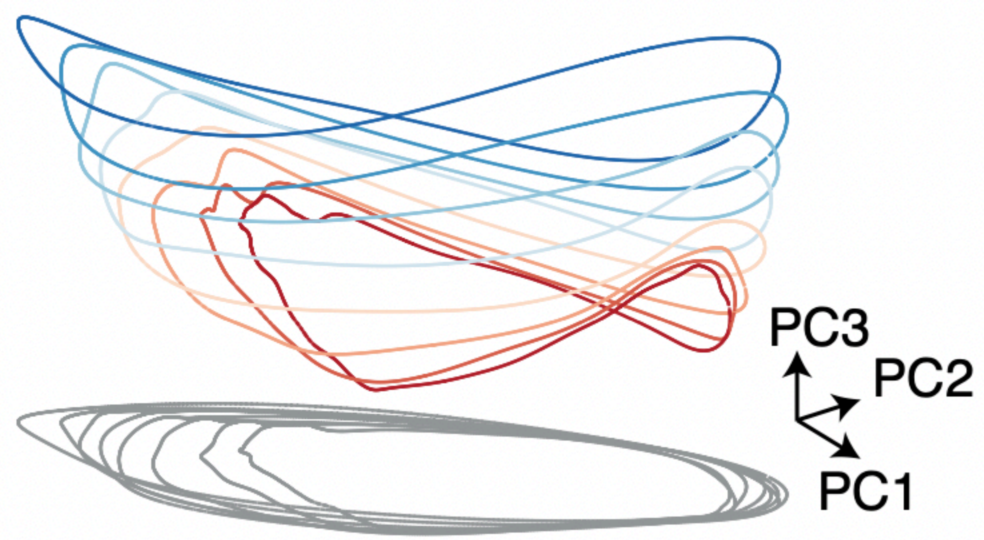
PCA TO INVESTIGATE THE MANIFOLDS OF NEURAL ACTIVITY



- Motor cortex recordings during cycling at different speed
- Picture shows neural responses:
  - each loop is once around a repeating cycle
  - blue is slowest.
- The 2D manifold PC1,PC2 clearly represents the cyclic motion

Urai, *et al. Nat. Neuro.*. 2022.

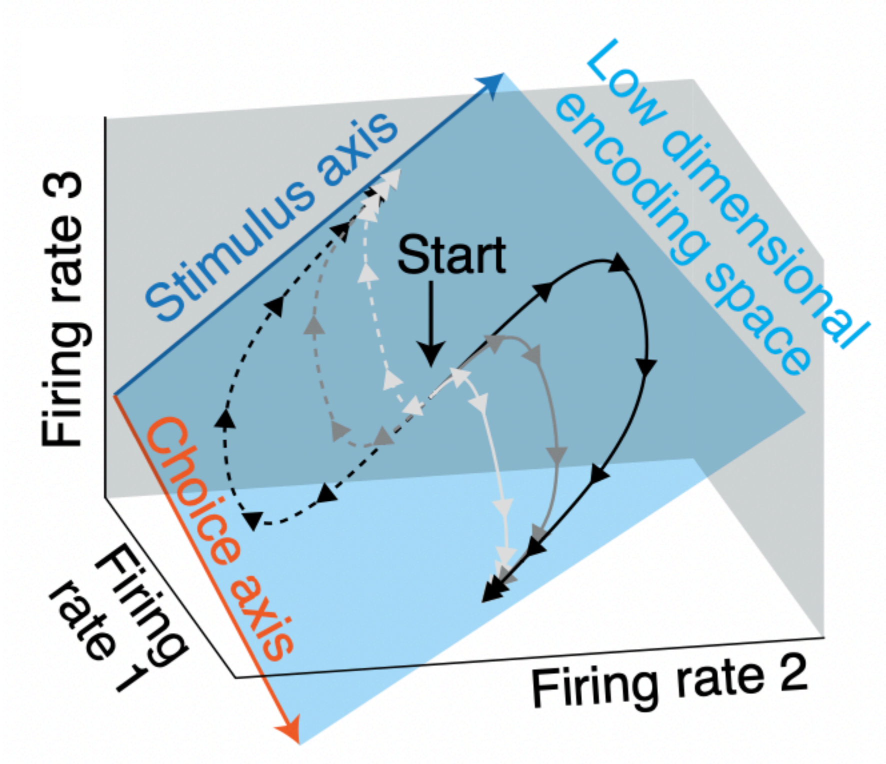
PCA TO INVESTIGATE THE MANIFOLDS OF NEURAL ACTIVITY



- Motor cortex recordings during cycling at different speed
- Picture shows neural responses:
  - each loop is once around a repeating cycle
  - blue is slowest.
- The 2D manifold PC1,PC2 clearly represents the cyclic motion
- The third dimension adds separability in the speed domain

Urai, *et al. Nat. Neuro.*. 2022.

PCA TO INVESTIGATE THE MANIFOLDS OF NEURAL ACTIVITY

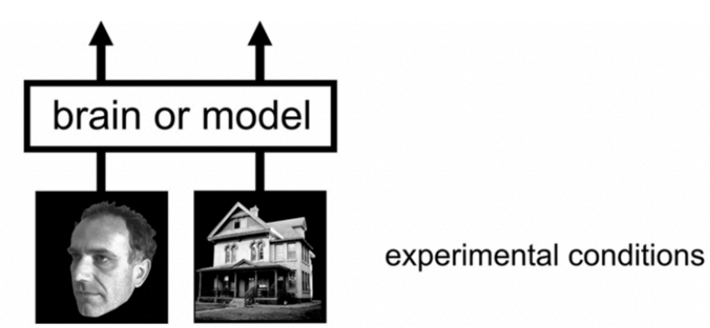


- Behaviorally relevant variables in a sensory decision making task:
  - Neural activities can be embedded in a 2D manifold:
    - A stimulus-axis
    - A choice-axis
- Behaviorally relevant neural variance is often explained by a small number of dimensions (blue, red axes).

Urai, *et al. Nat. Neuro.*. 2022.

REPRESENTATION SIMILARITY ANALYSIS TO INFER NETWORKS SPECIFICITY AND CONNECTIVITY

- Dissimilarity between 2 stimuli/ conditions is:
  - 1-C, C = Correlation between activity elicited by 2 stimuli/conditions
  - Ranges between 0 and 2:
    - 0 for perfect correlation
    - 1 for no correlation
    - 2 for perfect anticorrelation

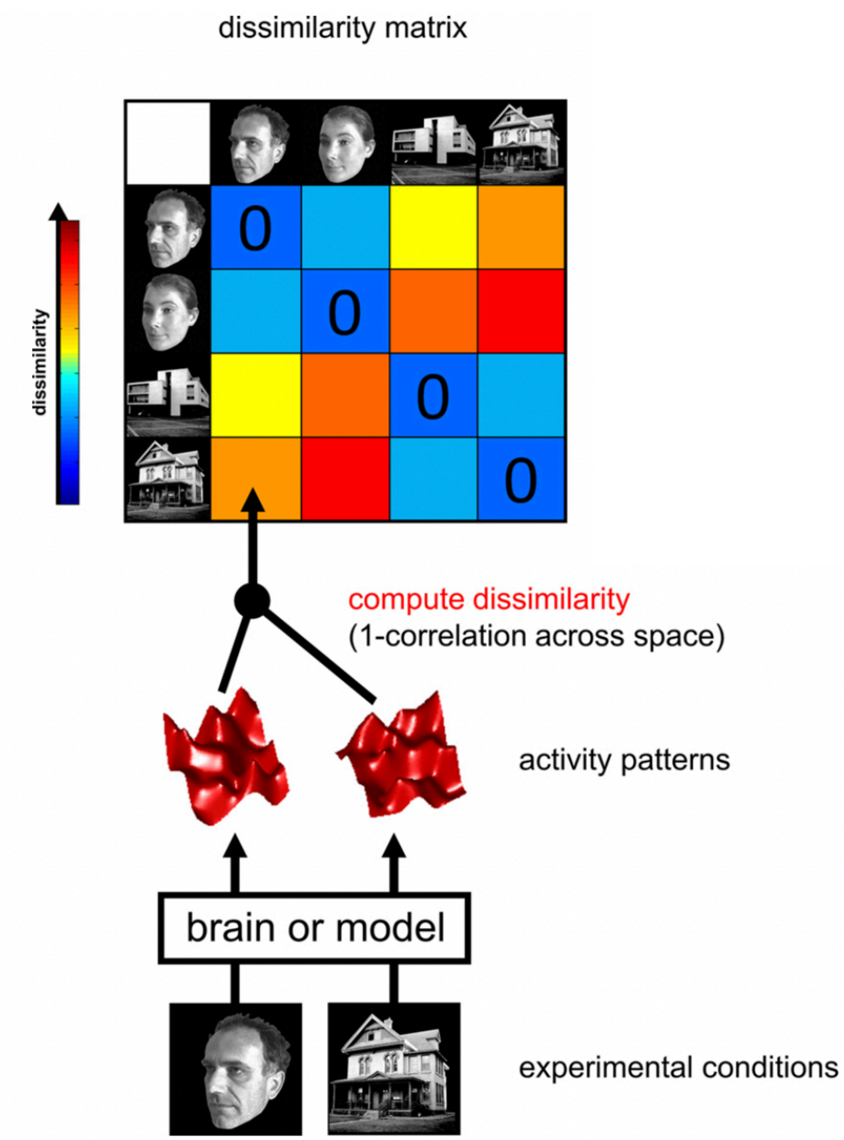


Kriegeskorte, Mur and Bandettini, *Front. Syst. Neurosci.* 2008

REPRESENTATION SIMILARITY ANALYSIS TO INFER NETWORKS SPECIFICITY AND CONNECTIVITY

- Dissimilarity between 2 stimuli / conditions is:
  - 1-C, C = Correlation between activity elicited by 2 stimuli / conditions
  - Ranges between 0 and 2:
    - 0 for perfect correlation
    - 1 for no correlation
    - 2 for perfect anticorrelation

- These dissimilarities for all pairs of conditions are assembled in the Representation Dissimilarity Matrix.
- Each cell of the RDM compares the response patterns elicited by two images.

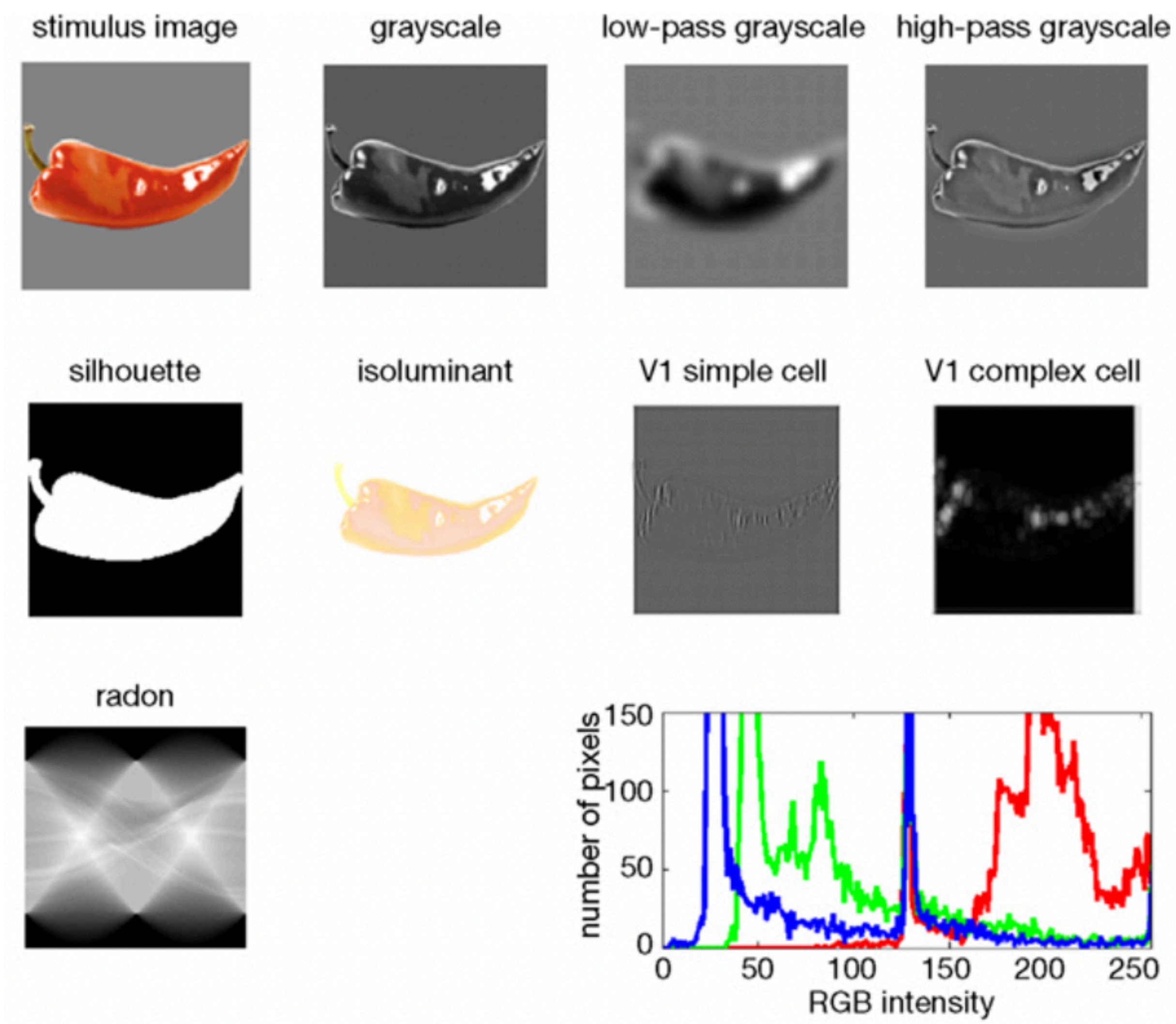


Kriegeskorte, Mur and Bandettini, *Front. Syst. Neurosci.* 2008

REPRESENTATION SIMILARITY ANALYSIS TO INFER ENCODING SPECIFICITY

- RDMs can be compared too
- e.g. to a model's RDM in response to a similar condition

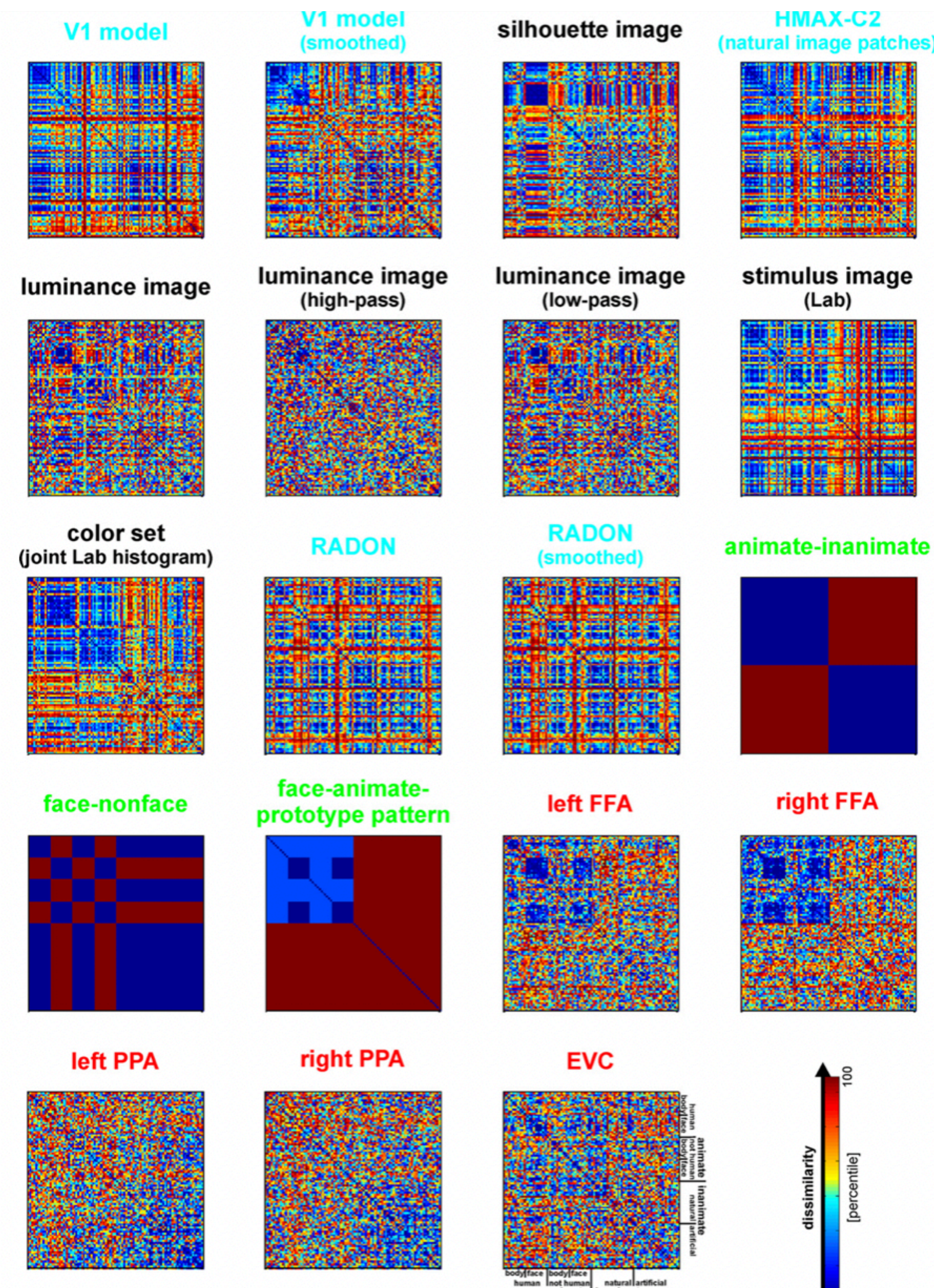
- Example here: the authors transform the image according to different algorithm or processing models



Kriegeskorte, Mur and Bandettini, *Front. Syst. Neurosci.* 2008

REPRESENTATION SIMILARITY ANALYSIS TO INFER ENCODING SPECIFICITY

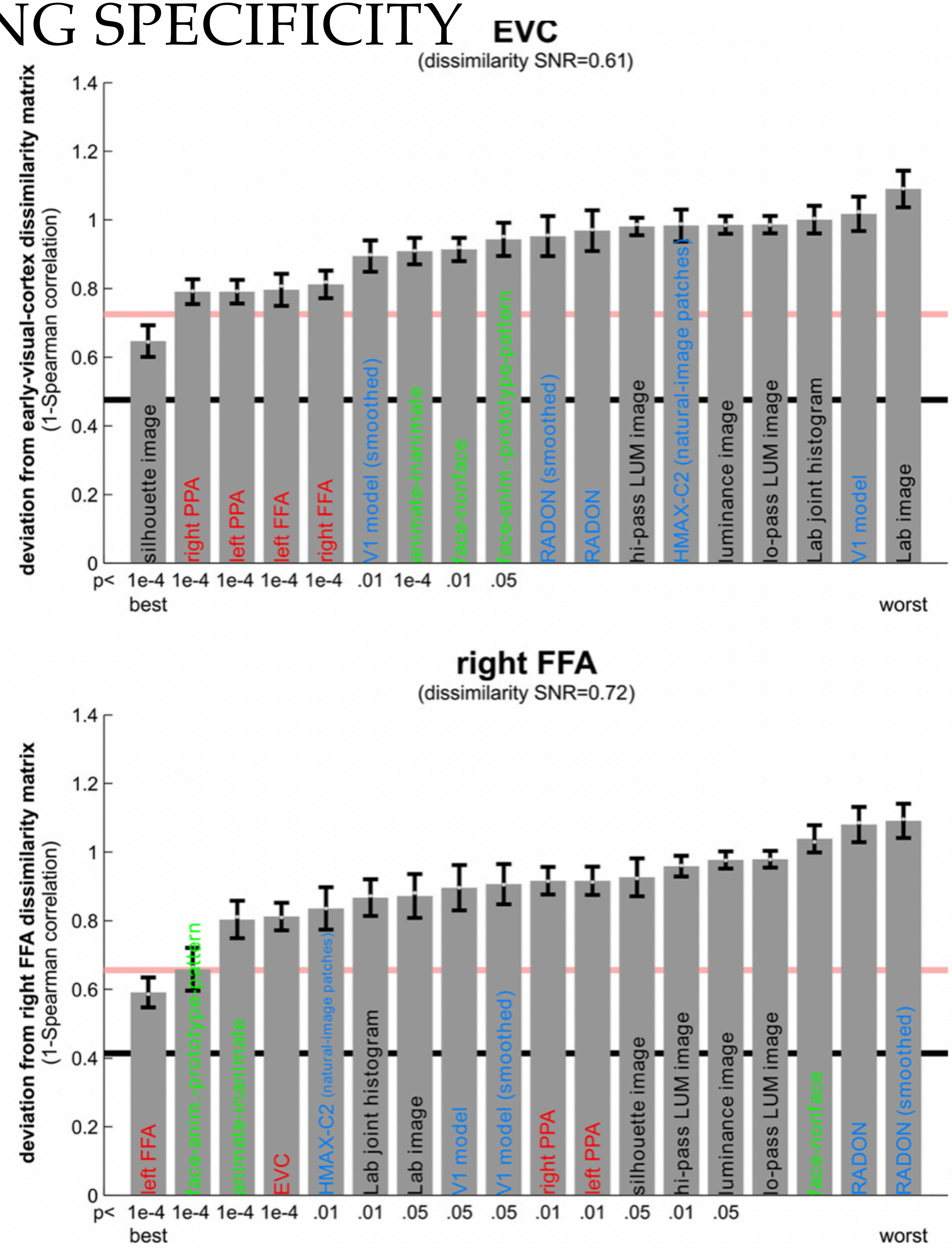
- They produce RDMs for different models and brain regions (obtained from fMRI data)



Kriegeskorte, Mur and Bandettini, *Front. Syst. Neurosci.* 2008

REPRESENTATION SIMILARITY ANALYSIS TO INFER ENCODING SPECIFICITY

- The correlation between the RDMs can help match:
  - region-to-region functional specificity
  - region-to-model functional specificity



Kriegeskorte, Mur and Bandettini, *Front. Syst. Neurosci.* 2008

- SUMMARY - RL AND NON-RL METHODS TO DECIPHERING BRAINS AND NEURAL NETWORKS

- RL:
  - The specificity of the design of the agent chosen enable to shed light on how humans or animals perform in tasks:
    - Its performance related to model design indicates what basic elements are needed to perform a task
    - The different components can help formulate hypothesis on a specific brain region's role in a task

- SUMMARY -
RL AND NON-RL METHODS TO DECIPHERING BRAINS AND NEURAL NETWORKS

- RL: Some direction of comparison of the approaches covered:
  - Dyna: to study replay processes
  - SR: to study prediction
  - Model-free/Model-Based: to investigate their interplay
  - Actor-critic VS policy gradient: to study value representation and action selection

- SUMMARY -
RL AND NON-RL METHODS TO DECIPHERING BRAINS AND NEURAL NETWORKS

- RL: Some direction of comparison of the approaches covered:
    - Dyna: to study replay processes
    - SR: to study prediction
    - Model-free/Model-Based: to investigate their interplay
    - Actor-critic VS policy gradient: to study value representation and action selection

- all of those approaches enable to study behaviors
- they also improve from neurosciences advances
    - in particular, we need more flexible agents!

- SUMMARY -
RL AND NON-RL METHODS TO DECIPHERING BRAINS AND NEURAL NETWORKS

- Manifold:
  - Can capture task-relevant dimensions of neural activities - practical for dimension reduction
  - Comparing those dimension and the stability of the dynamics using modeling enables to shed light on neural computations
  - Embedding useful task-related dynamics within neural network can help perform tasks

- SUMMARY -
RL AND NON-RL METHODS TO DECIPHERING BRAINS AND NEURAL NETWORKS

- RSA:
  - Can capture representational and functional marker of a brain region/ model by looking at its pattern of activity correlations
  - RDMs capture how different do their react to stimuli/experimental conditions
  - Can be used to compare to models and or other brain regions:
    - With models, it gives information on the encoding of the region
    - With brain regions, it can be used to infer connectivity between brain regions