nature neuroscience

Article

https://doi.org/10.1038/s41593-024-01671-x

Reward prediction error neurons implement an efficient code for reward

Authors: Heiko H. Schütt, Dongjae Kim, Wei Ji Ma

Presented by Snow

Efficient Coding

sensory neuroscience

neuronal codes are optimized for efficiency, that is, to convey as much information as possible with given budgets for the number of neurons and the number of action potentials

Efficient Coding

sensory neuroscience

neuronal codes are optimized fo efficiency, that is, to convey as much information as possible w th given budgets for the number of neurons and the number of ac ion potentials

reward

Reward Prediction Error Neurons?
(RPEN)

Overview

1. Derive the efficient code

2. Efficient code satisfies properties of real neurons

3. How do neurons learn the tuning? Distributional RL

Overview

1. Derive the efficient code

Efficient Sensory Encoding and Bayesian Inference with Heterogeneous Neural Populations

Deep Ganguli and **Eero P. Simoncelli**
Howard Hughes Medical Institute, Center for Neural Science, and Courant Institute of Mathematical Sciences, New York University, New York, NY 10003

Deep Ganguli: dganguli@cns.nyu.edu; Eero P. Simoncelli: eero.simoncelli@nyu.edu

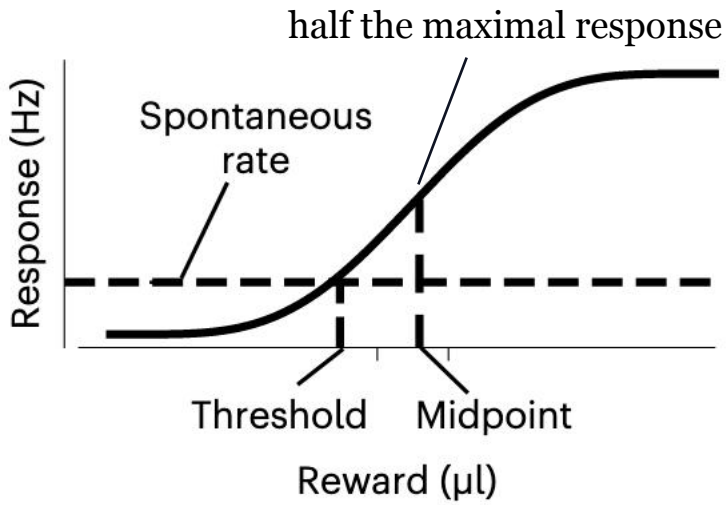
with some modifications

Overview

1. Derive the efficient code

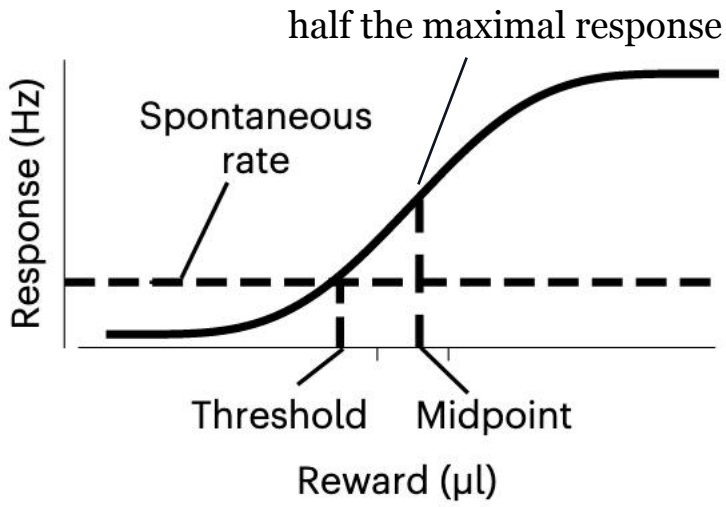
Main idea: Optimize Fisher Information(with certain constraints)

Efficient Code computation



RPEN's sigmoid tuning curve

What are the elments of a population coding?



RPEN's sigmoid tuning curve

(given N neurons)

- ❑ Tuning curves(response functions)

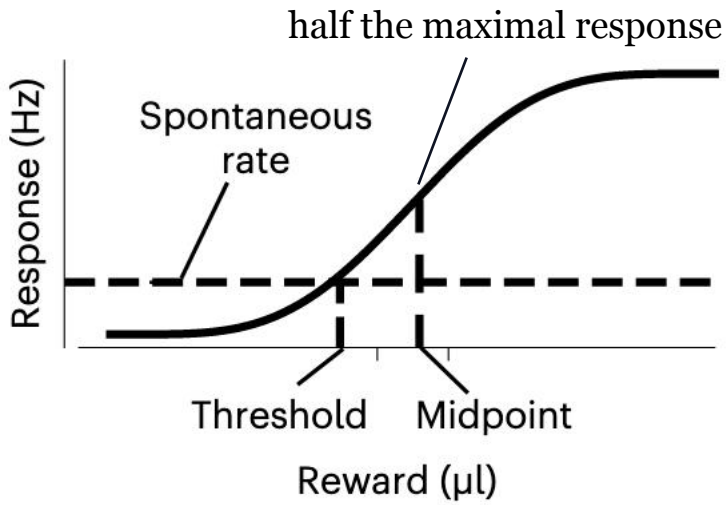
(assume they are the same w.r.t. given midpoint=μ)

- ❑ Density of neurons with midpoint=μ
- ❑ Regularization:Mapping from reward to unit interval

$$h_\mu(R) = h_0[F(R) - F(\mu)]g(\mu)$$

Efficient Code computation

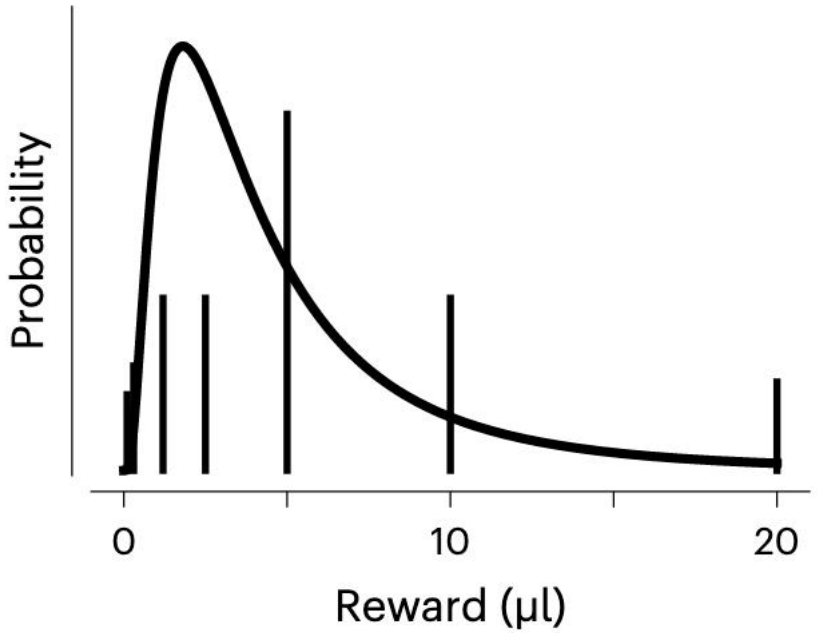
$$h_\mu(R) = h_0[F(R) - F(\mu)]g(\mu)$$



RPEN's sigmoid tuning curve

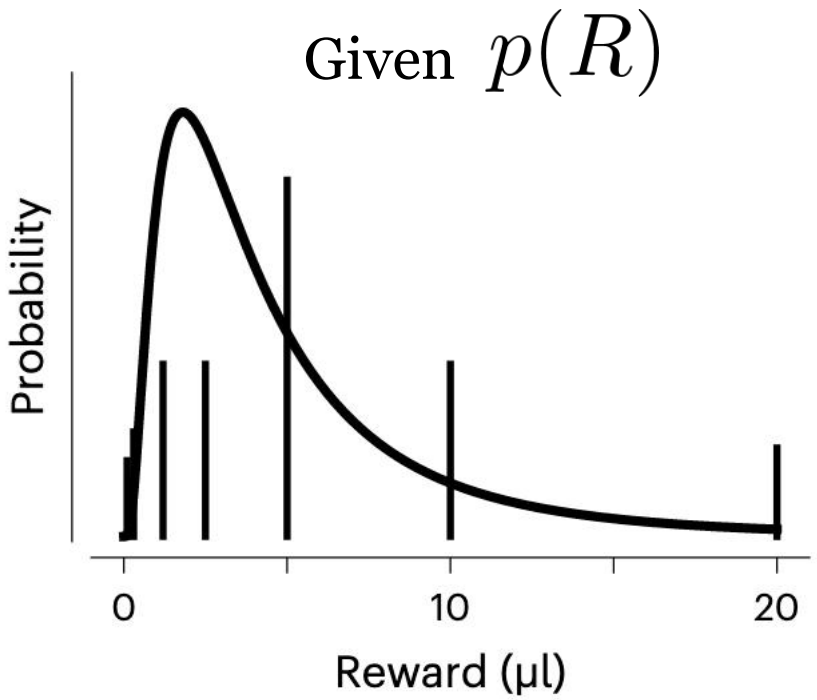
- Response function, midpoint = μ
- Reward mapping $F : R \rightarrow [0, 1]$
- A prototypical sigmoidal function h_0
- Neural Gain g as a function of R

Problem of efficient code:



What is the most efficient population to encode the reward stimuli?

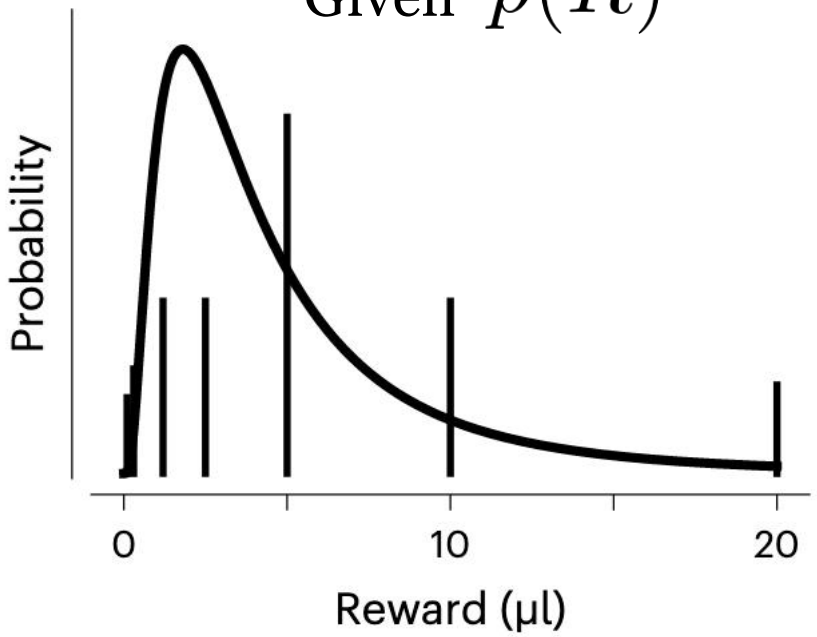
Problem of efficient code:



What is the most efficient population to encode the reward stimuli?

Problem of efficient code:

Given $p(R)$



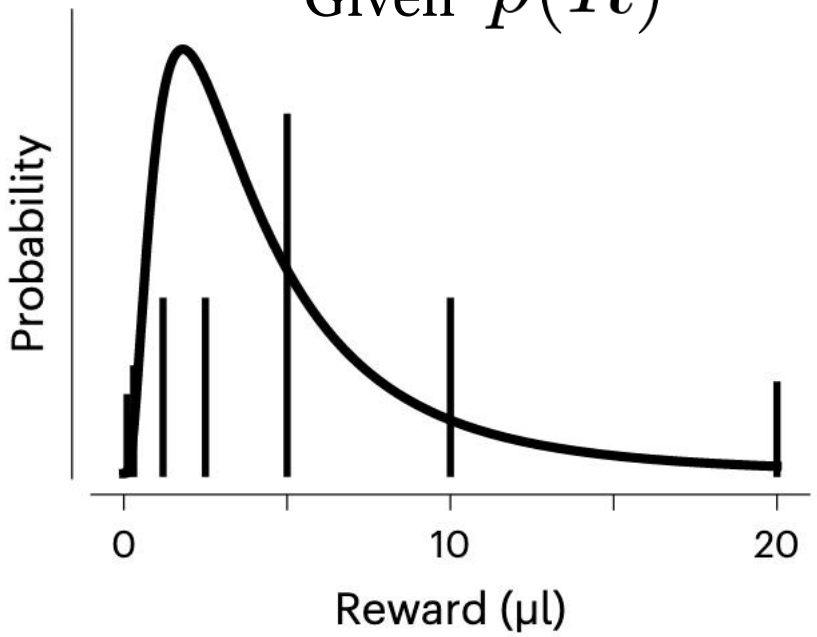
Fisher Information

maximize

What is the most efficient population to encode the reward stimuli?

Problem of efficient code:

Given $p(R)$



Fisher Information

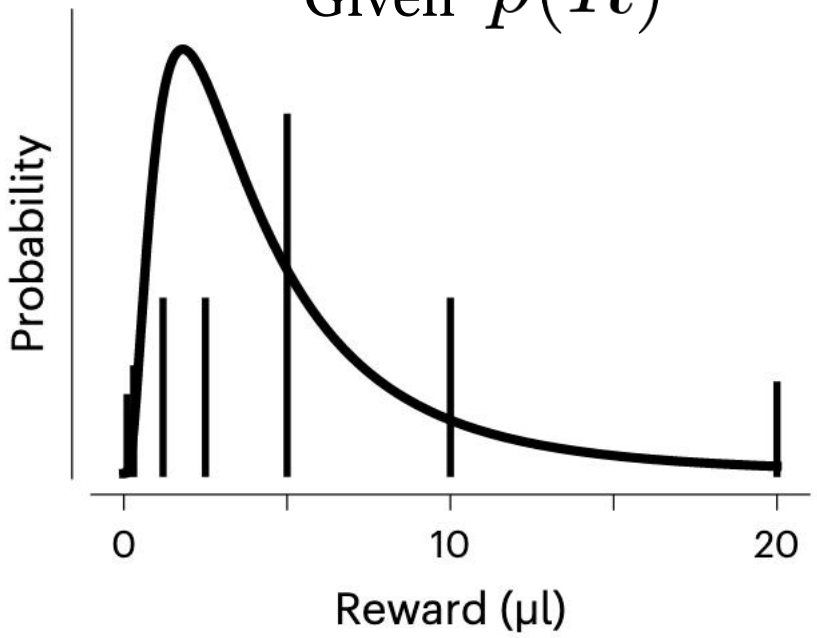
maximize
under
constraints:

upper bound to the expected number of spikes

What is the most efficient population to encode the reward stimuli?

Problem of efficient code:

Given $p(R)$



What is the most efficient population to encode the reward stimuli?

Fisher Information

maximize under constraints:

upper bound to the expected number of spikes

$$f(R) = p(R)$$

$$d(R) \propto \frac{p(R)}{[1 - P(R)]^{1-\alpha}}$$

$$g(R) \propto \frac{1}{[1 - P(R)]^{\alpha}}$$

α is a free parameter, $\alpha \in [0, 1]$

Zoom back into a single neuron: $h_\mu(R) = h_0[F(R) - F(\mu)]g(\mu)$

$$f(R) = p(R)$$

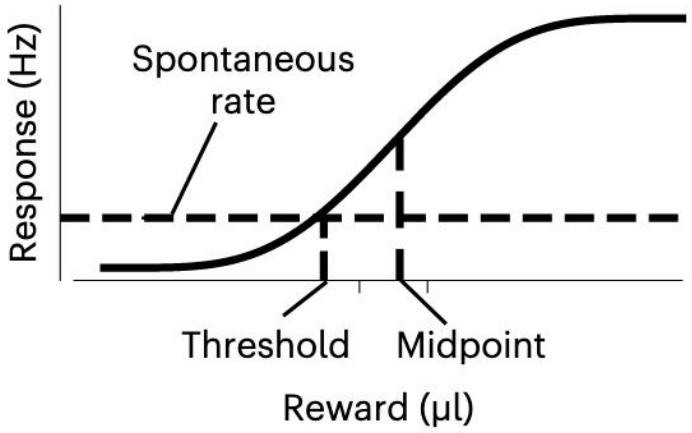
$$d(R) \propto \frac{p(R)}{[1 - P(R)]^{1-\alpha}}$$

$$g(R) \propto \frac{1}{[1 - P(R)]^{\alpha}}$$

α is a free parameter, $\alpha \in [0, 1]$

these results are independent of the exact shape of the sigmoid tuning curves(h_0)

Zoom back into a single neuron: $h_\mu(R) = h_0[F(R) - F(\mu)]g(\mu)$

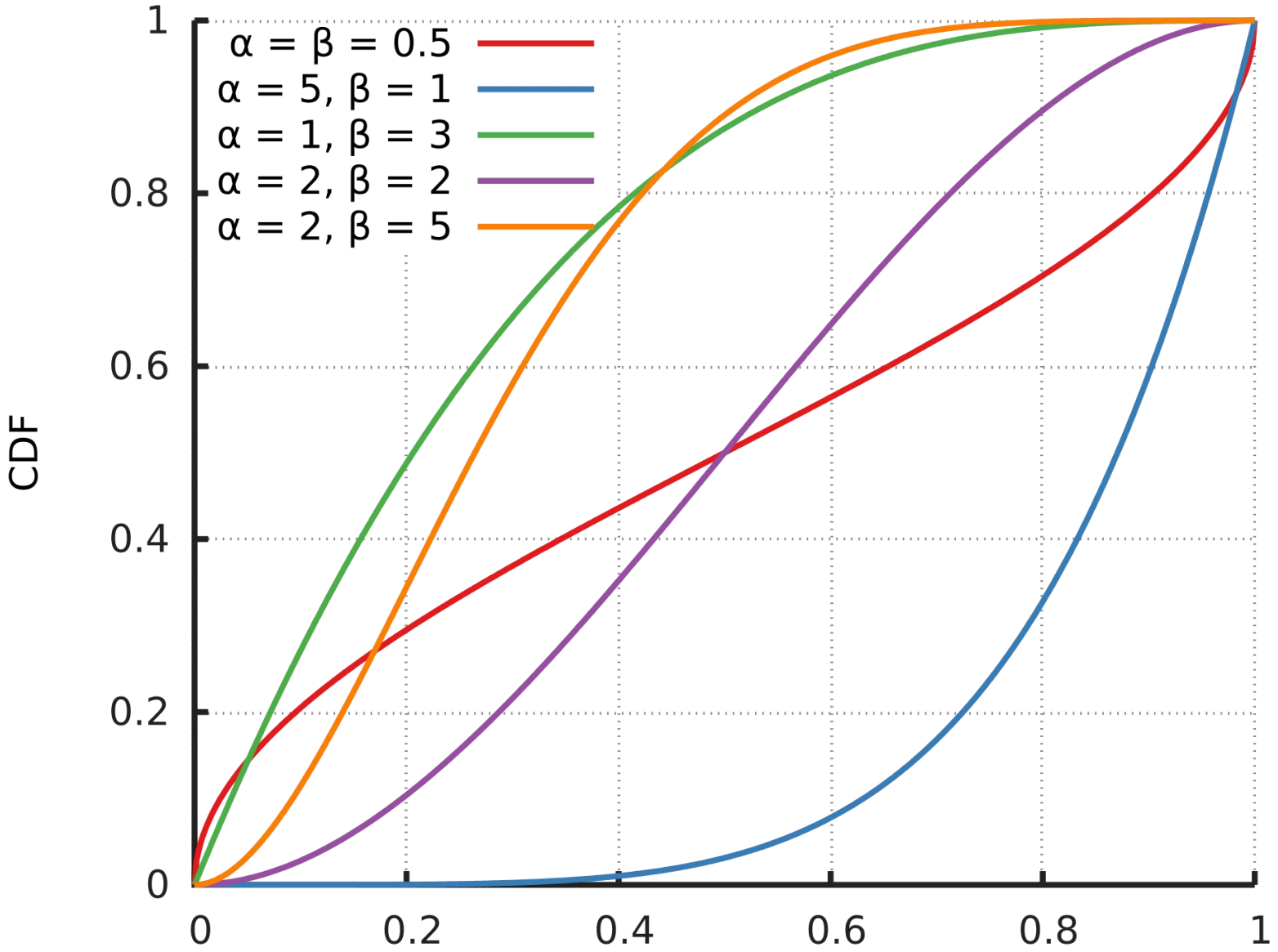


slope k; spontaneous firing rate r*

we fit these two parameters:
minimize MSE of thesholds
(between efficient code and measured neurons)

c.d.f. of beta distributions as the sigmoid response functions

Cumulative Distribution Function of Beta Distribution



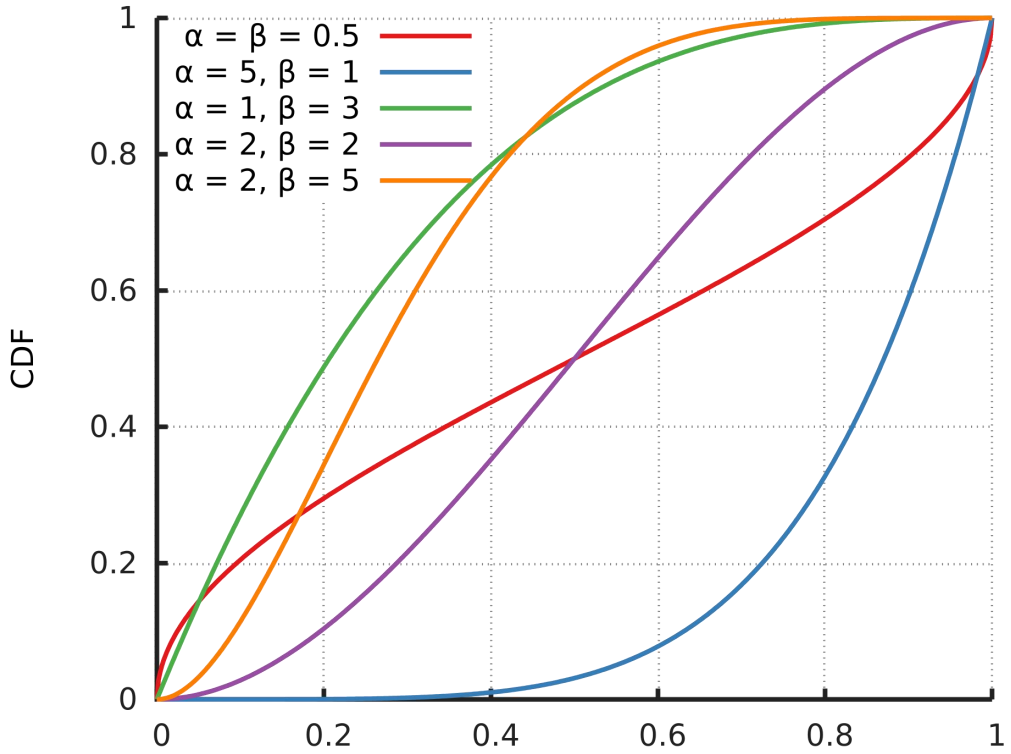
Zoom back into a single neuron:

c.d.f. of beta distributions as the sigmoid response functions

beta distribution is parameterized by a, b

for a neuron with slope k, midpoint p

set a=kp, b=k(1-p)



good boundary behavior

What's new?

Efficient Sensory Encoding and Bayesian Inference with Heterogeneous Neural Populations

Deep Ganguli and **Eero P. Simoncelli**
Howard Hughes Medical Institute, Center for Neural Science, and Courant Institute of Mathematical Sciences, New York University, New York, NY 10003

Deep Ganguli: dganguli@cns.nyu.edu; Eero P. Simoncelli: eero.simoncelli@nyu.edu

$$f(R) = p(R)$$

$$d(R) \propto \frac{p(R)}{[1 - P(R)]^{1-\alpha}}$$ a special case: $\alpha = 1$

$$g(R) \propto \frac{1}{[1 - P(R)]^{\alpha}}$$

Overview

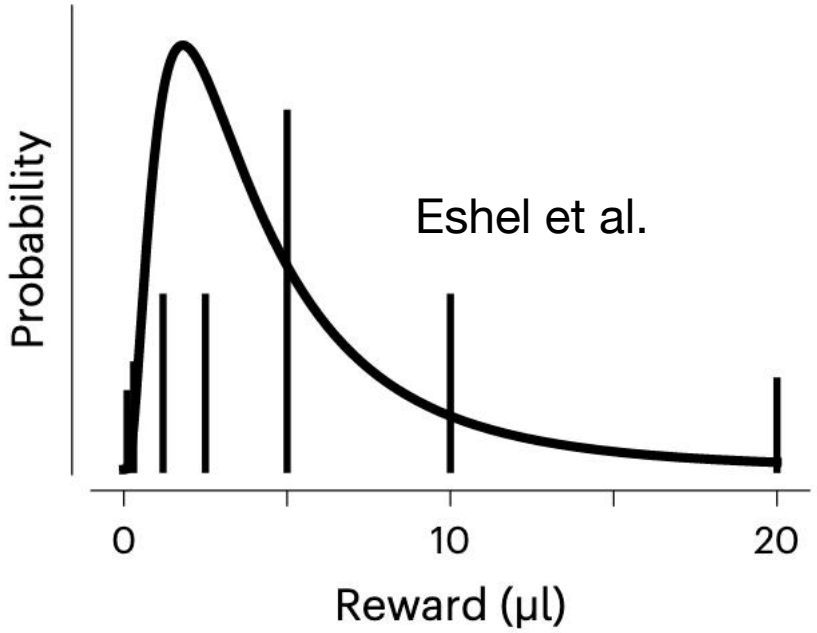
1. Derive the efficient code

2. Efficient code satisfies properties of real neurons

3. How do neurons learn the tuning? Distributional RL

Application 1: variable-reward task

reward condition:



vertical bars: seven values of reward with probability

smooth curve: fitted moment-matched log-normal distribution

Application 1: variable-reward task

Data & Model fitting

Data & Model fitting

Step 1: Fitting Logistic sigmoid functions to measured neurons

$$\sigma(R) = \frac{a_1}{1 - e^{-a_2(R-a_3)}}$$

maximum-likelihood estimation based on Poisson noise

$$\hat{\mathbf{a}} = \text{argmin}_{\mathbf{a}} \sum_{j=1}^{T} (r_j \log(\sigma(R_j; \mathbf{a})) - \sigma(R_j; \mathbf{a}))$$

where $\mathbf{a} = (a_1, a_2, a_3)$

Data & Model fitting

Step 2: α

$$f(R) = p(R)$$

$$d(R) \propto \frac{p(R)}{[1 - P(R)]^{1-\alpha}}$$

$$g(R) \propto \frac{1}{[1 - P(R)]^{\alpha}}$$

α is a free parameter, $\alpha \in [0, 1]$

Application 1: variable-reward task

Data & Model fitting

Step 2: α

maximize the probability of the observed midpoints

Application 1: variable-reward task

Data & Model fitting

Step 2: α

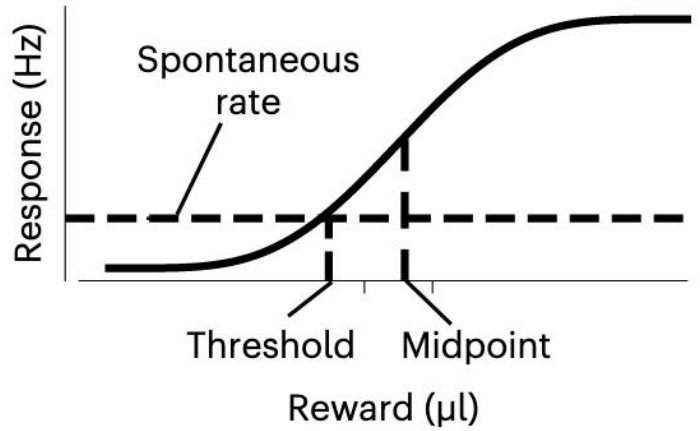
maximize the probability of the observed midpoints

$$\alpha = 0.673$$

Data & Model fitting

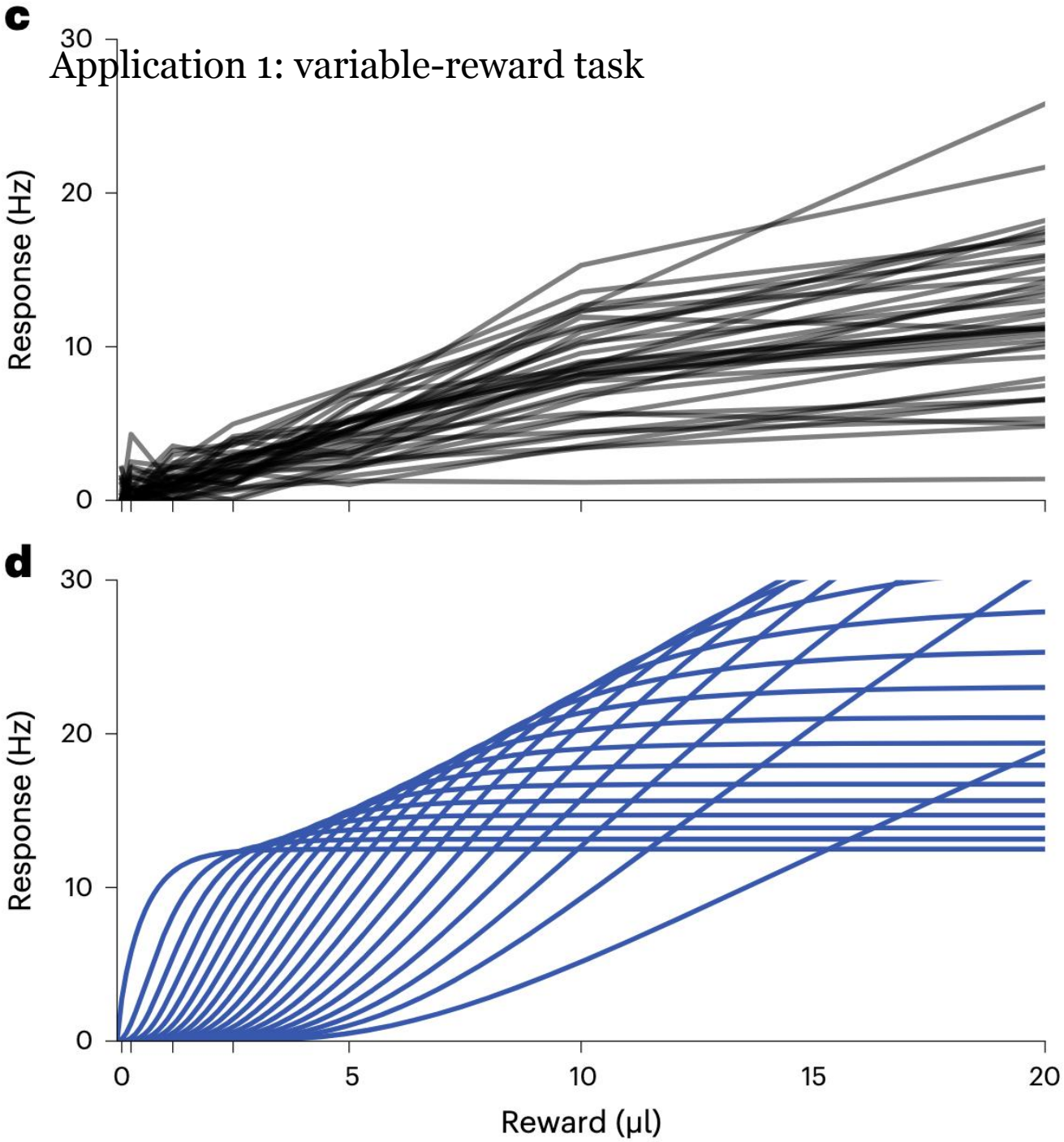
Step 3: slope k; spontaneous firing rate r*

minimize MSE of thesholds
(between efficient code and
measured neurons)



$$\text{argmin}_{r^*,k}\ E\left[\theta_i - h_i^{-1}(r^*, k)\right]^2$$

Application 1: variable-reward task



39 dopaminergic RPENs measured by Eshel et al.

(preprocessed)

Efficient code

Application 1: variable-reward task

Predictions:

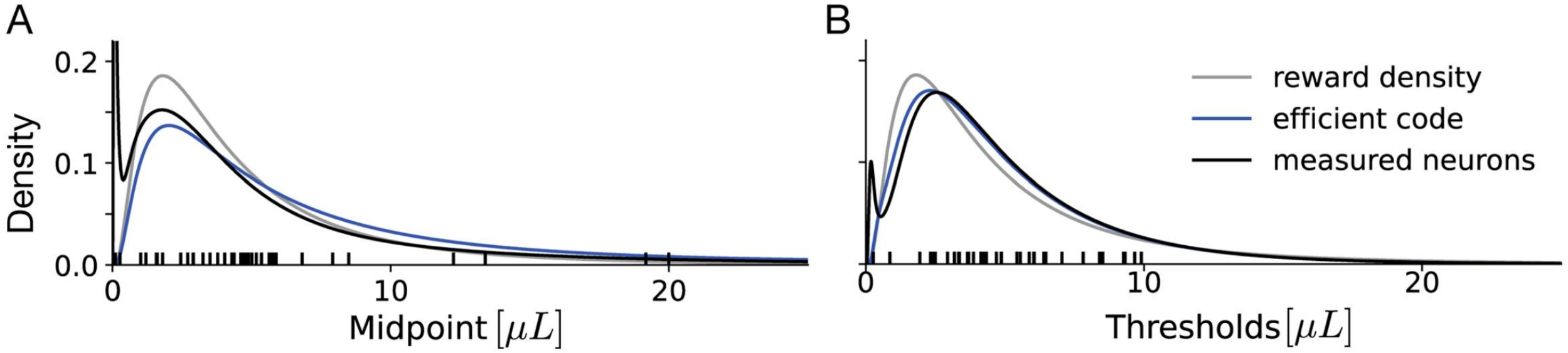
1. RPEN midpoints cover the reward distribution
2. RPEN gain increases with threshold
3. RPEN tuning curve asymmetry flips with increasing threshold
4. RPEN slope decreases with threshold

My question: Why *these* predictions?

Application 1: variable-reward task

Predictions

Prediction 1: RPEN midpoints cover the reward distribution

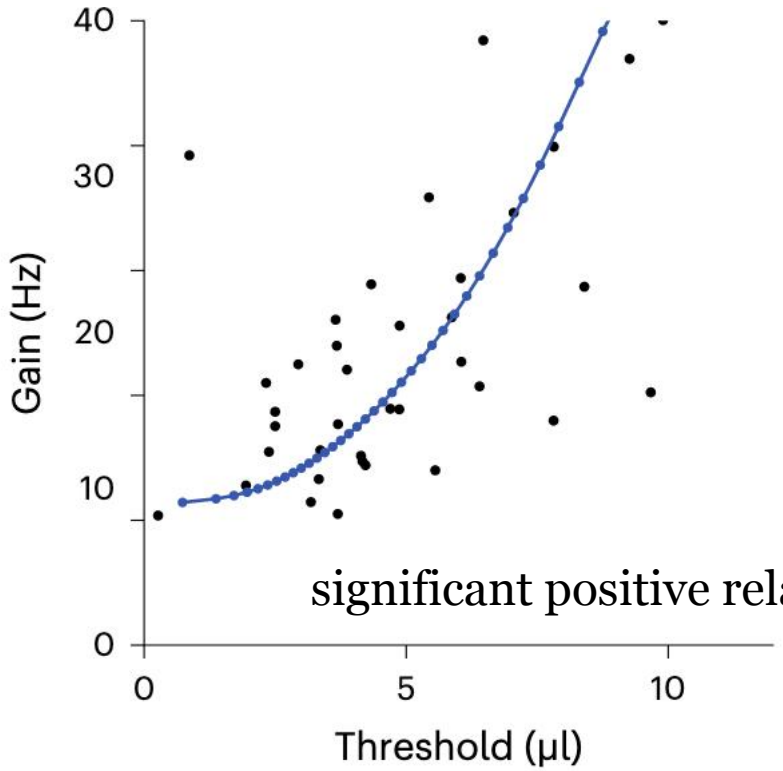


difference not significant

Application 1: variable-reward task

Predictions

Prediction 2: RPEN gain increases with theshold



significant positive relationship

with same expected number of spikes:

higher theshold→respond to fewer rewards

↓

afford a higher gain

Blue: Efficient code

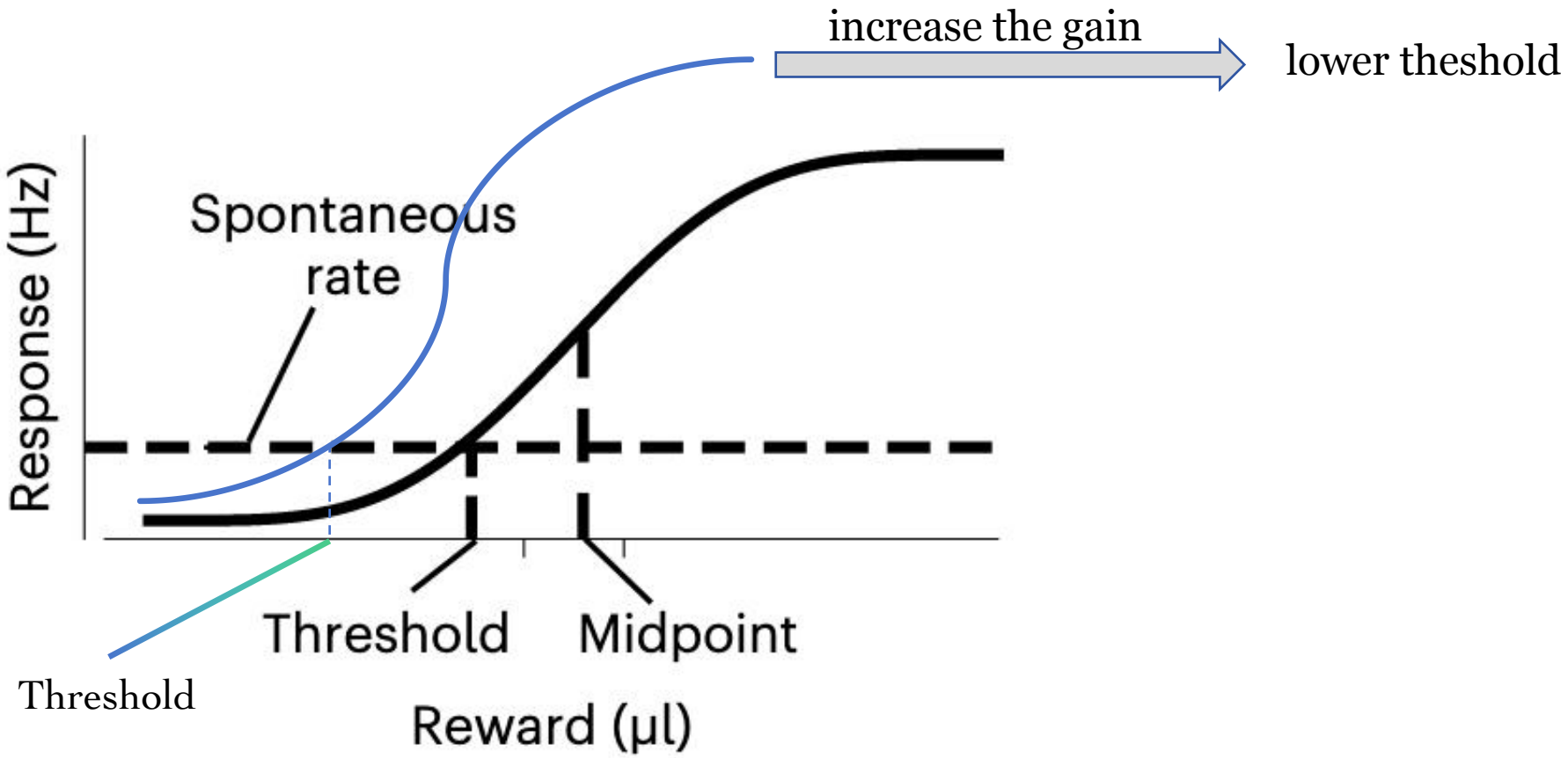
Black: Measured neurons

this empirical finding has not been reported before

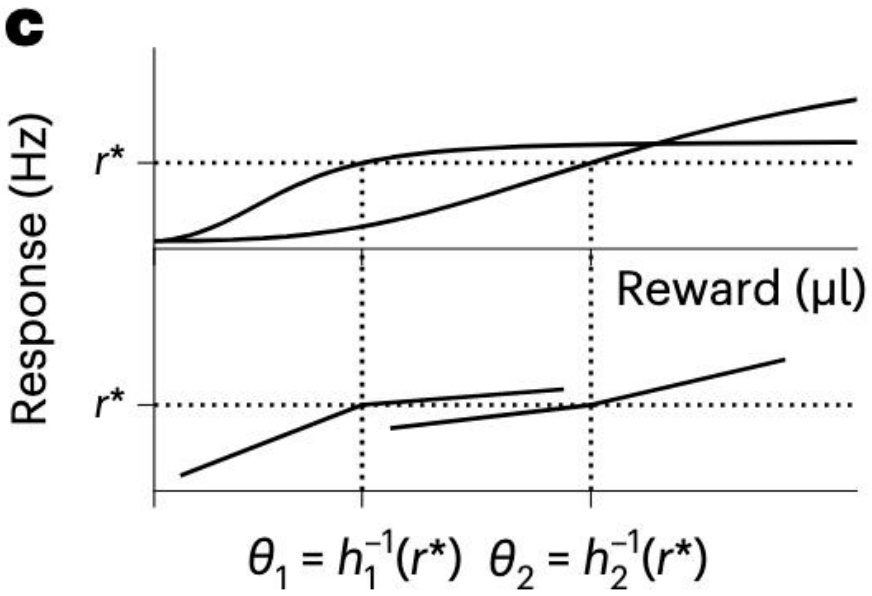
Application 1: variable-reward task

Predictions

Prediction 3: RPEN tuning curve asymmetry flips with increasing theshold



Prediction 3: RPEN tuning curve asymmetry flips with increasing theshold



Degree of asymmetry:

$$\frac{\beta+}{\beta+ + \beta-}$$

$\beta+$ → slope above theshold

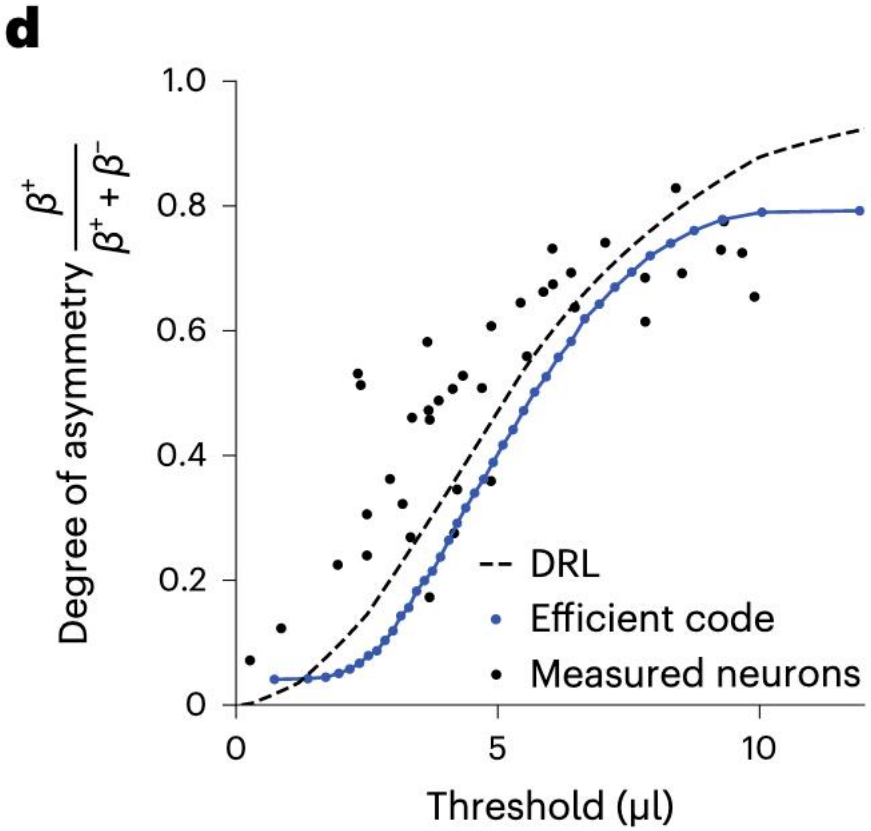
$\beta+ + \beta-$ → slope below theshold

Fit two linear response functions
(following Dabney's work)

Application 1: variable-reward task

Predictions

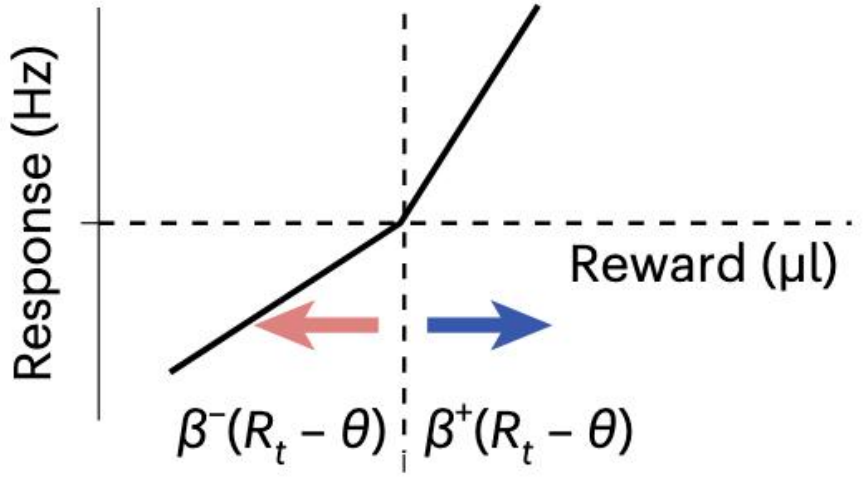
Prediction 3: RPEN tuning curve asymmetry flips with increasing theshold



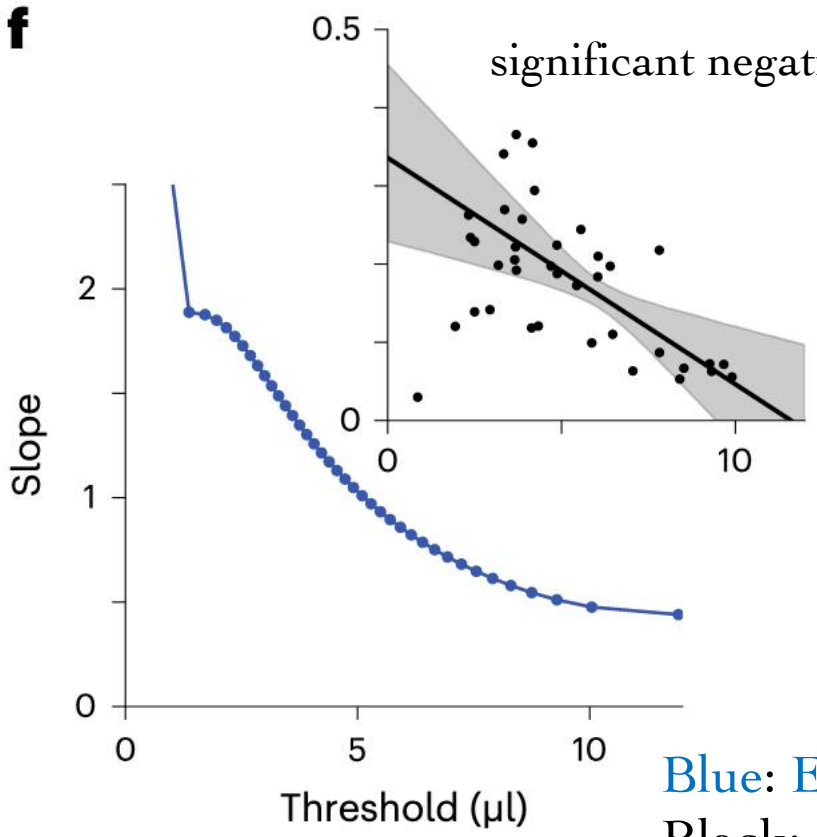
strong positive relationship

Prediction 3: RPEN tuning curve asymmetry flips with increasing theshold

What is the meaning of asymmetry here?



Prediction 4: RPEN slope decreases with theshold



significant negative linear relationship

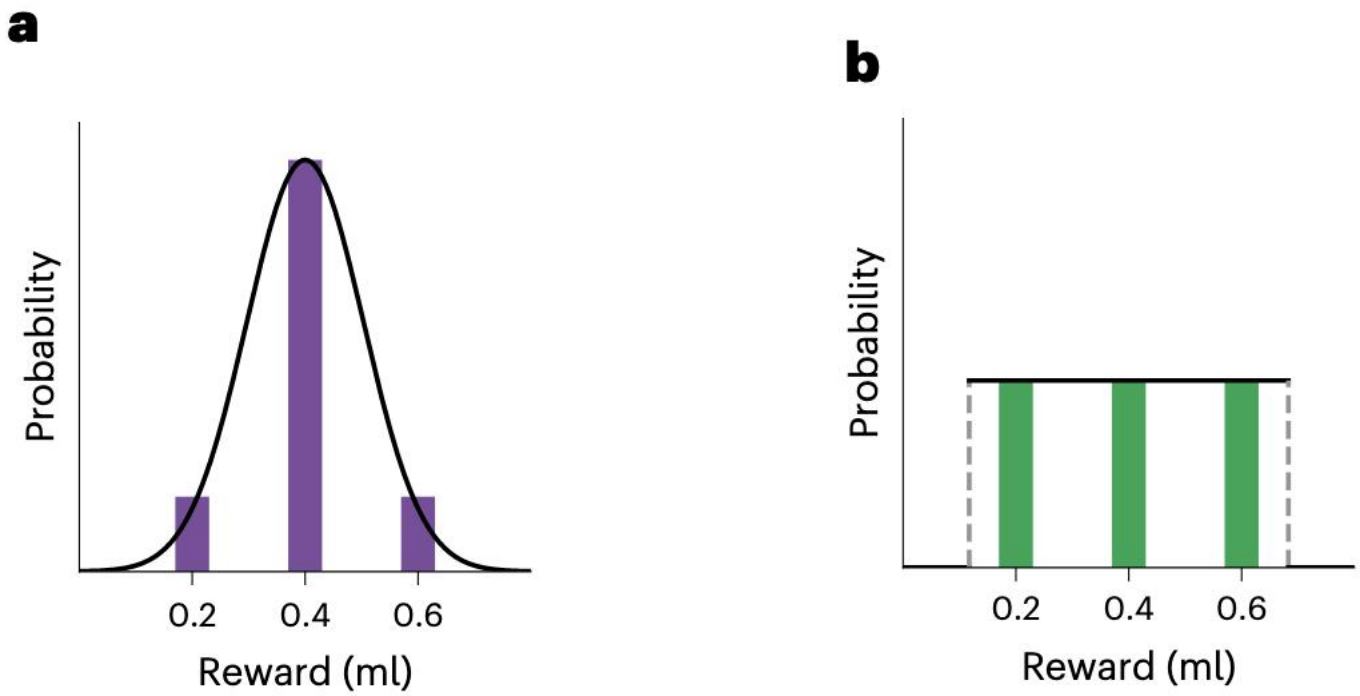
response functions are shallower in regions with lower reward density
(⟹cover broader reward ranges with fewer neurons)

Blue: Efficient code
Black: Measured neurons

this empirical finding has not been reported before

Application 2: variable-distribution task

reward condition: Rothenhoefer et al.

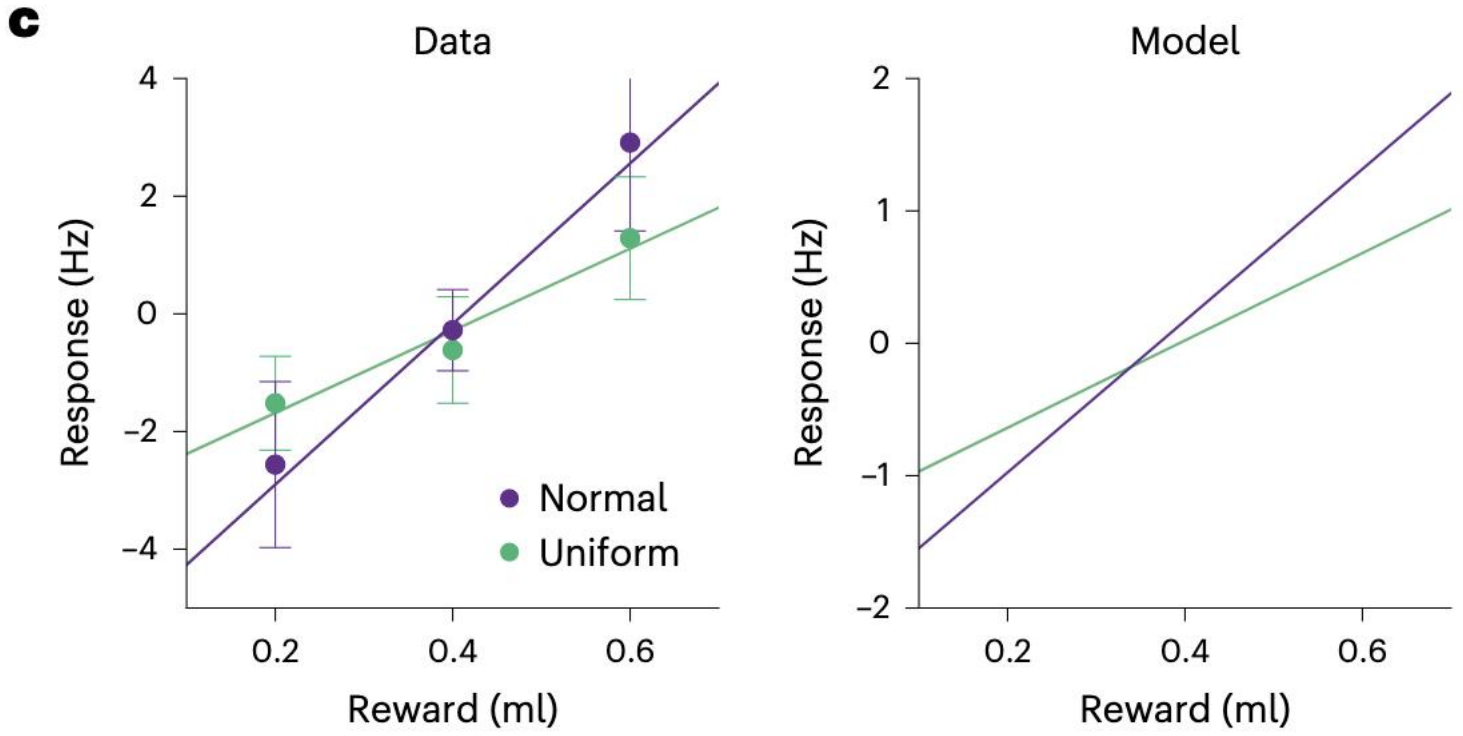


‘normal’ distribution ‘uniform’ distribution

(with the same mean)

Application 2: variable-distribution task

example neurons



main finding of Rothenhoefer et al. :

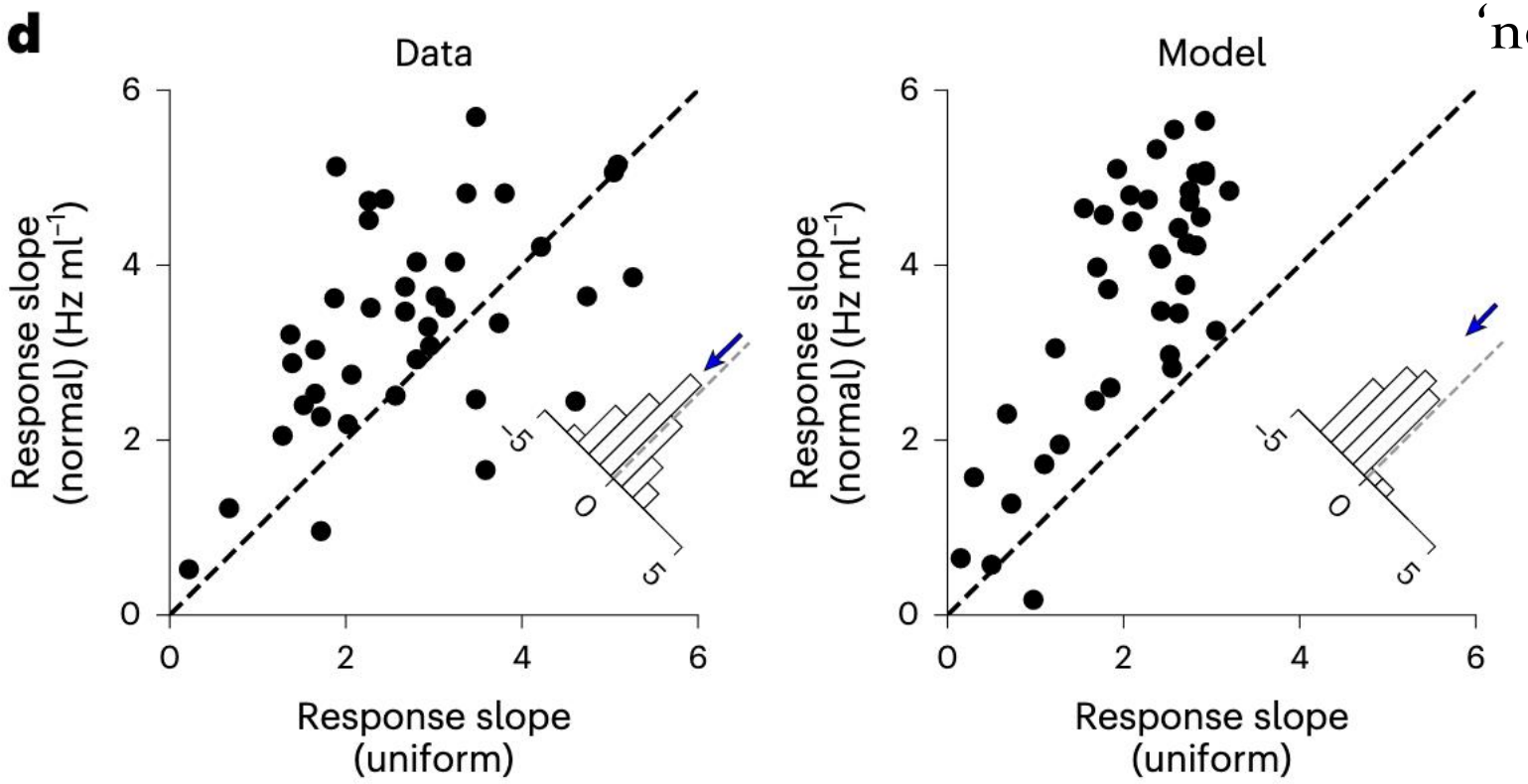
steeper response functions for 'normal' than 'uniform'

(dopamine responses are amplified for rare rewards)

Application 2: variable-distribution task

all neurons

steeper response functions for ‘normal’ than ‘uniform’



Overview

1. Derive the efficient code

2. Efficient code satisfies properties of real neurons

3. How do neurons learn the tuning? Distributional RL

Reinforcement Learning

$$Q(x, a) = \mathbb{E}R(x, a) + \gamma \mathbb{E}Q(X', A')$$

Distributional Reinforcement Learning

$$Z(x, a) \stackrel{D}{=} R(x, a) + \gamma Z(X', A')$$

↓

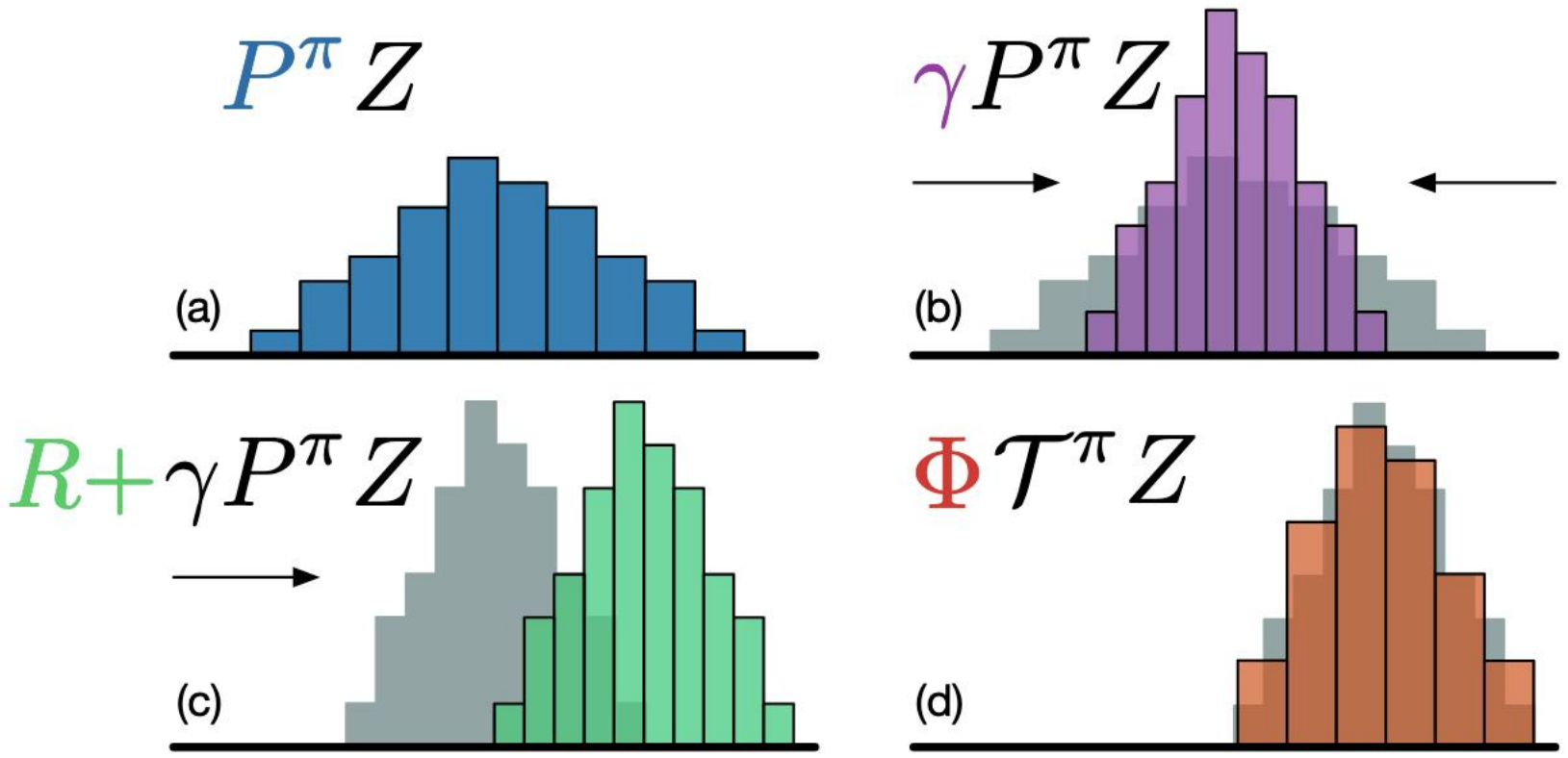
random variable instead of an expected value

A Distributional Perspective on Reinforcement Learning

Marc G. Bellemare [*1] **Will Dabney** [*1] **Rémi Munos** [1]

Distributional Reinforcement Learning

$$Z(x,a) \overset{D}{=} R(x,a) + \gamma Z(X',A')$$



Distributional Reinforcement Learning

The Wasserstein Metric

calculate the distance between two distributions

$$d_p(F, G) := \inf_{U,V} \|U - V\|_p$$

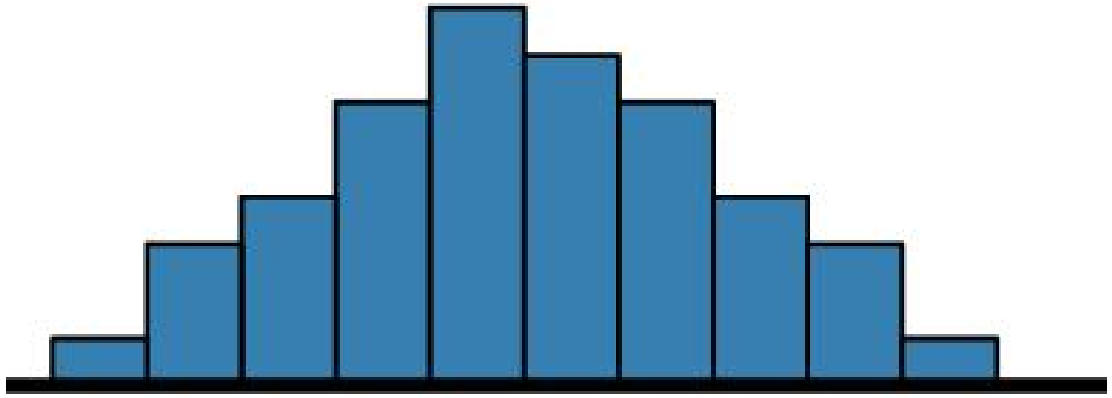
Distributional Reinforcement Learning

How to implement in tasks?

Distributional Reinforcement Learning

How to implement in tasks?

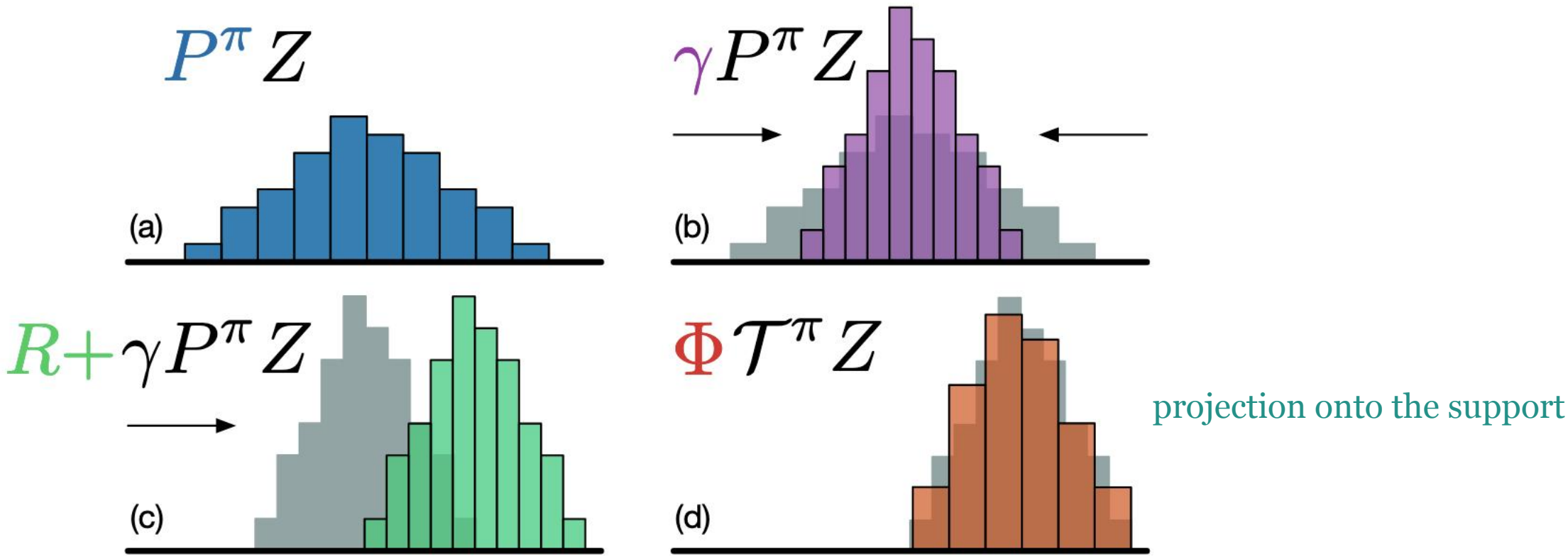
Approximate the distribution(continuous) with a discrete set of parametrized value functions



"highly expressive and computationally friendly"

Distributional Reinforcement Learning

$$Z(x,a) \overset{D}{=} R(x,a) + \gamma Z(X',A')$$



projection onto the support

DRL in dopamine neurons

Article

A distributional code for value in dopamine-based reinforcement learning

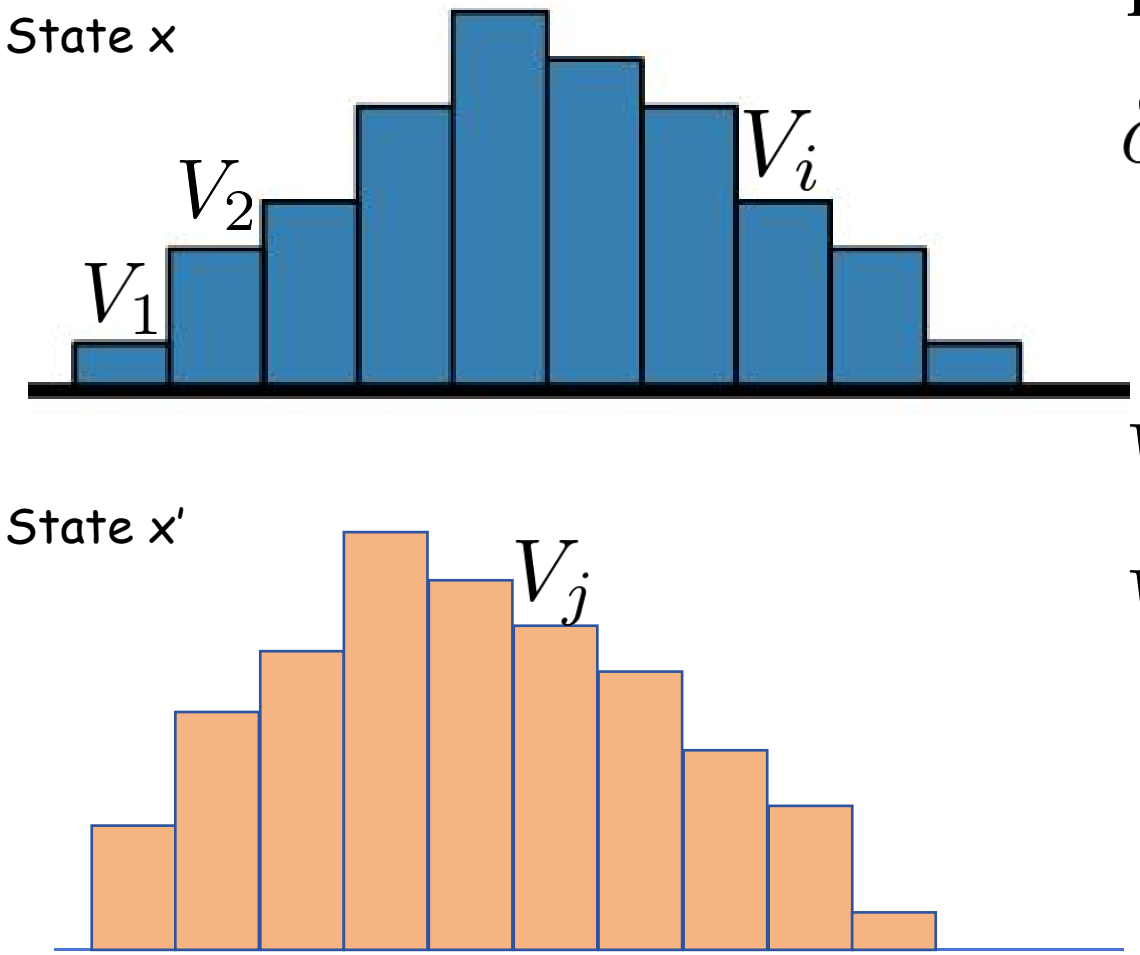
https://doi.org/10.1038/s41586-019-1924-6

Received: 3 January 2019

Accepted: 19 November 2019

Will Dabney[1,5]*, Zeb Kurth-Nelson[1,2,5], Naoshige Uchida[3], Clara Kwon Starkweather[3], Demis Hassabis[1], Rémi Munos[1] & Matthew Botvinick[1,4,5]

DRL in dopamine neurons



For each value function V_i, prediction error:

$$\delta_i = r + \gamma V_j(x') - V_i(x)$$

Asymmetry update rule:

$$V_i(x) \leftarrow V_i(x) + \alpha_i^+ \delta_i \quad \text{for } \delta_i > 0$$

$$V_i(x) \leftarrow V_i(x) + \alpha_i^- \delta_i \quad \text{for } \delta_i < 0$$

DRL in dopamine neurons

$$V_i(x) \leftarrow V_i(x) + \alpha_i^+ f(\delta_i) \quad \text{for } \delta_i > 0$$
$$V_i(x) \leftarrow V_i(x) + \alpha_i^- f(\delta_i) \quad \text{for } \delta_i < 0$$

where f is a function that transforms the prediction error

DRL in dopamine neurons

$$V_i(x) \leftarrow V_i(x) + \alpha_i^+ f(\delta_i) \quad \text{for } \delta_i > 0$$
$$V_i(x) \leftarrow V_i(x) + \alpha_i^- f(\delta_i) \quad \text{for } \delta_i < 0$$

take $f(x) = sign(x)$ $\alpha_i^+ = 3, \alpha_i^- = 1$

DRL in dopamine neurons

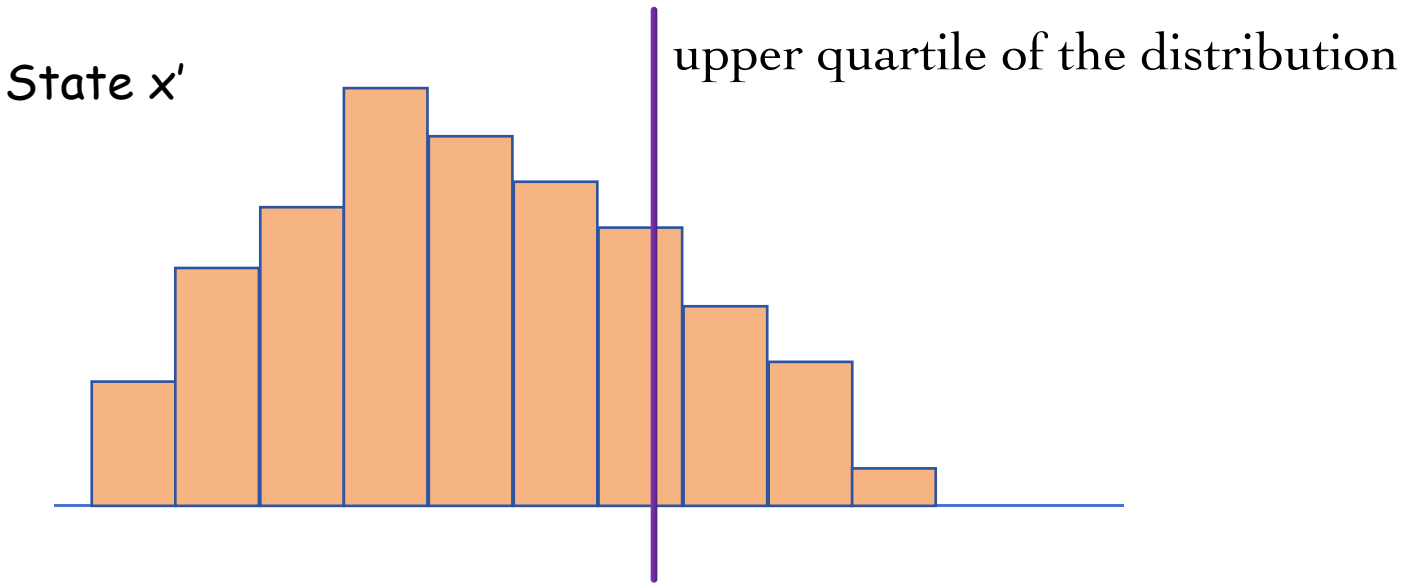
$$V_i(x) \leftarrow V_i(x) + 3 \quad \text{for } \delta_i > 0$$

$$V_i(x) \leftarrow V_i(x) - 1 \quad \text{for } \delta_i < 0$$

DRL in dopamine neurons

$$V_i(x) \leftarrow V_i(x) + 3 \quad \text{for } \delta_i > 0$$

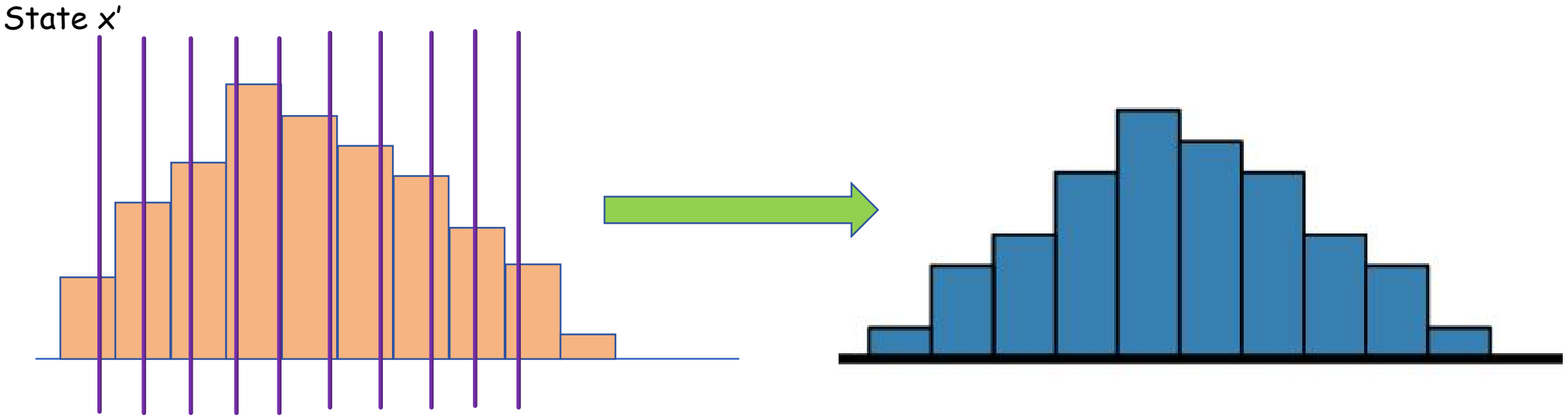
$$V_i(x) \leftarrow V_i(x) - 1 \quad \text{for } \delta_i < 0$$



DRL in dopamine neurons

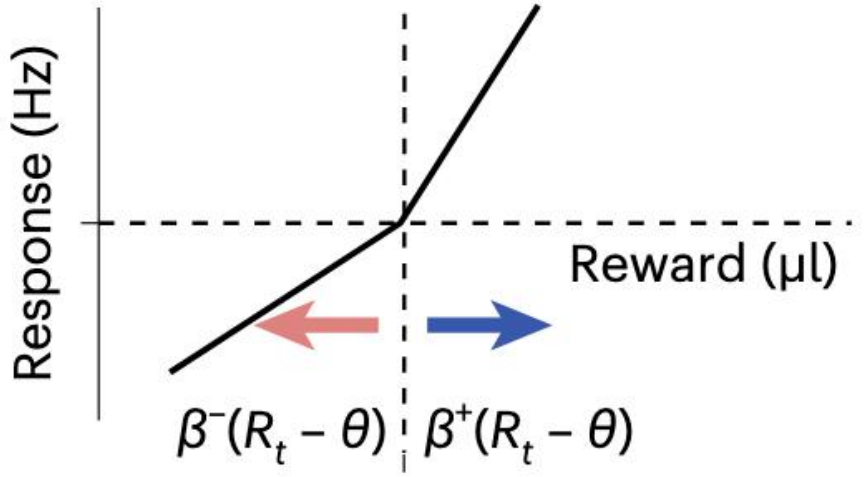
$$V_i(x) \leftarrow V_i(x) + \alpha_i^+ \quad \text{for } \delta_i > 0$$
$$V_i(x) \leftarrow V_i(x) - \alpha_i^- \quad \text{for } \delta_i < 0$$

$$\tau_i = \frac{\alpha_i^+}{\alpha_i^+ + \alpha_i^-}$$



Prediction 3: RPEN tuning curve asymmetry flips with increasing theshold

What is the meaning of asymmetry here?



Learning rules for the efficient code

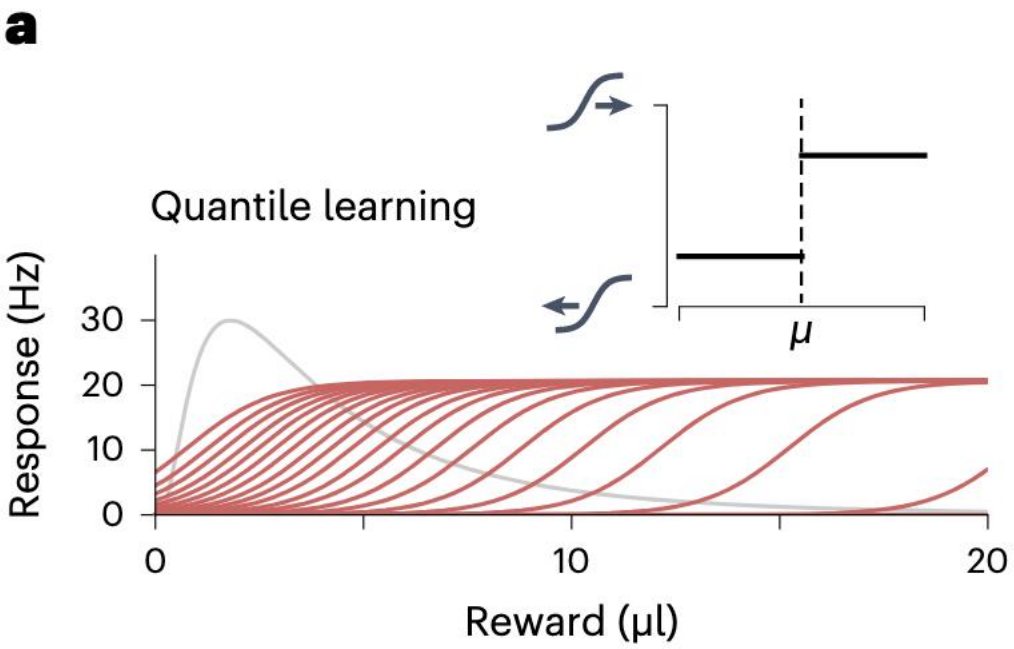
- ❑ Quantile learning
- ❑ Slope learning
- ❑ Gain

Application 1: variable-reward task

Predictions:

1. RPEN midpoints cover the reward distribution
2. RPEN gain increases with threshold
3. RPEN tuning curve asymmetry flips with increasing threshold
4. RPEN slope decreases with threshold

Quantile learning



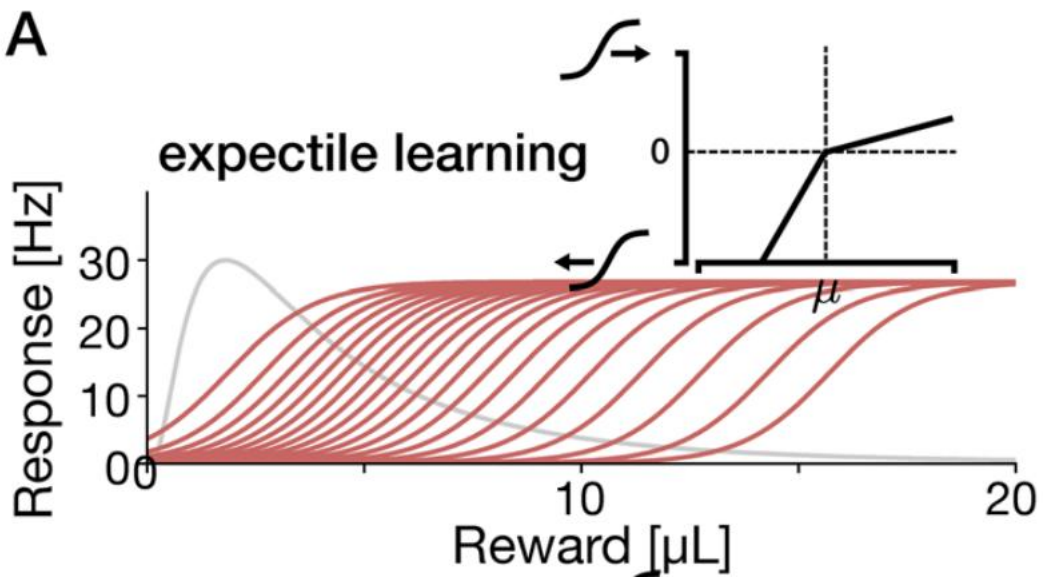
If reward > midpoint, shift the curve to the right, step size = **b**

If reward < midpoint, shift the curve to the left, step size = **a**

converge to the **p** th quantile of the reward distribution

where **ap = b(1 - p)**

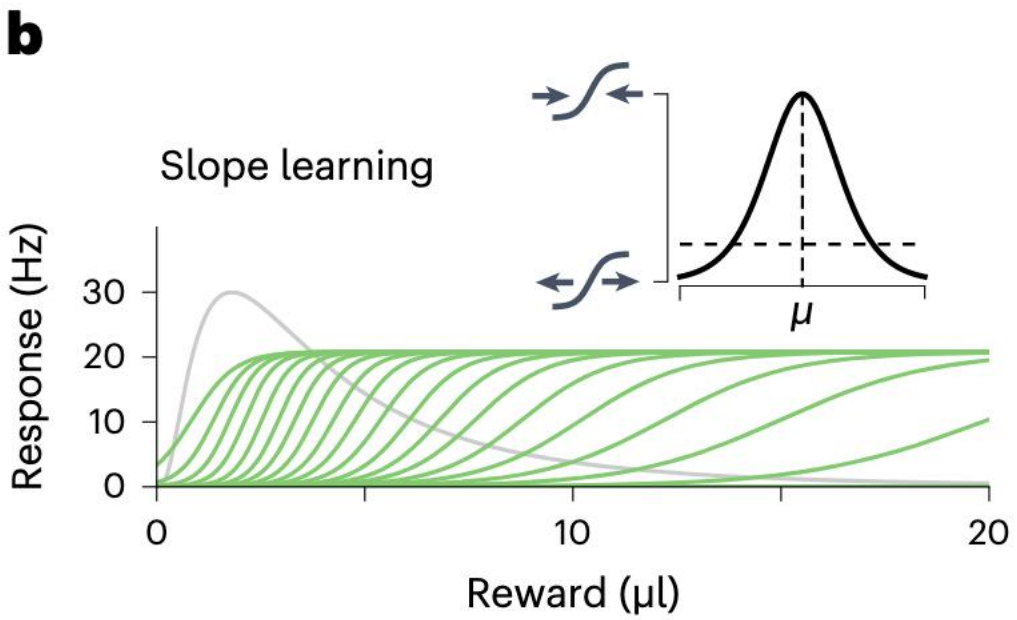
Expectile learning



If reward > midpoint, shift the curve to the right, step size = **b** * prediction error

If reward < midpoint, shift the curve to the left, step size = **a** * prediction error

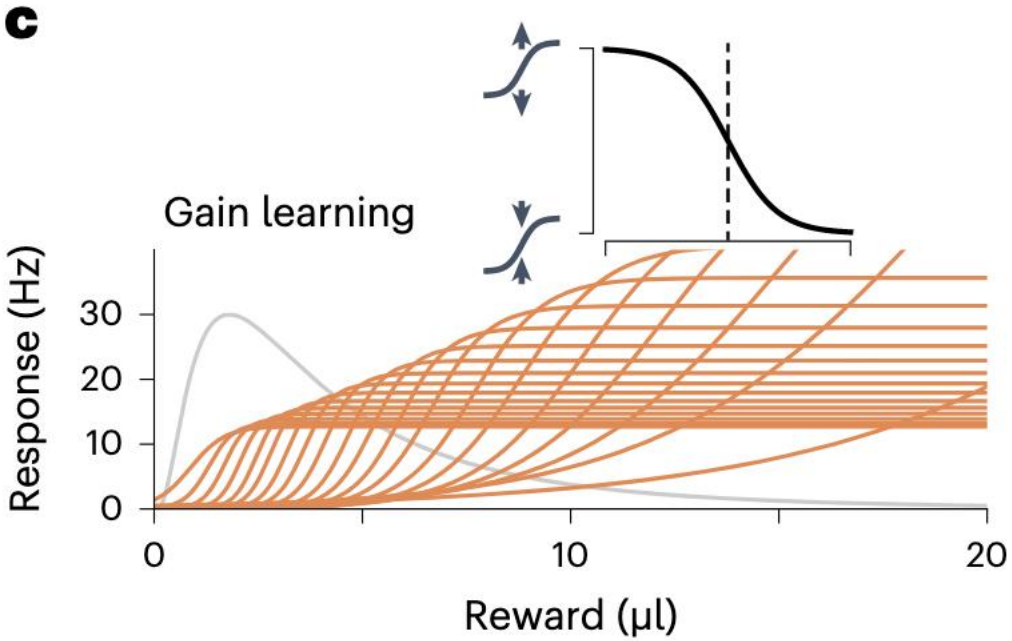
Slope learning



In the efficient code, the RPEN slope should be proportional to the local probability density of rewards

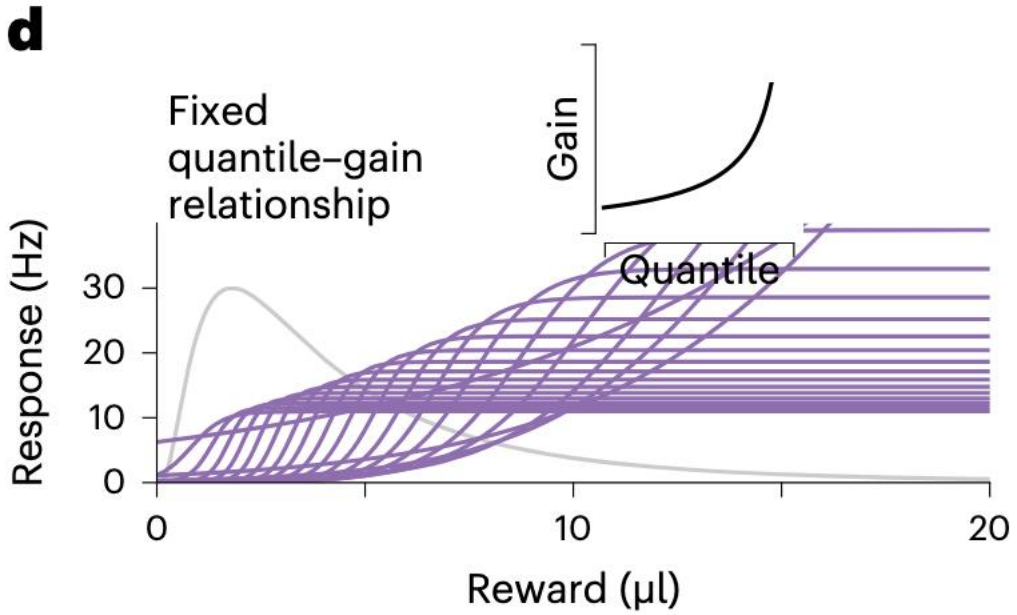
Gain Method 1

Gain learning

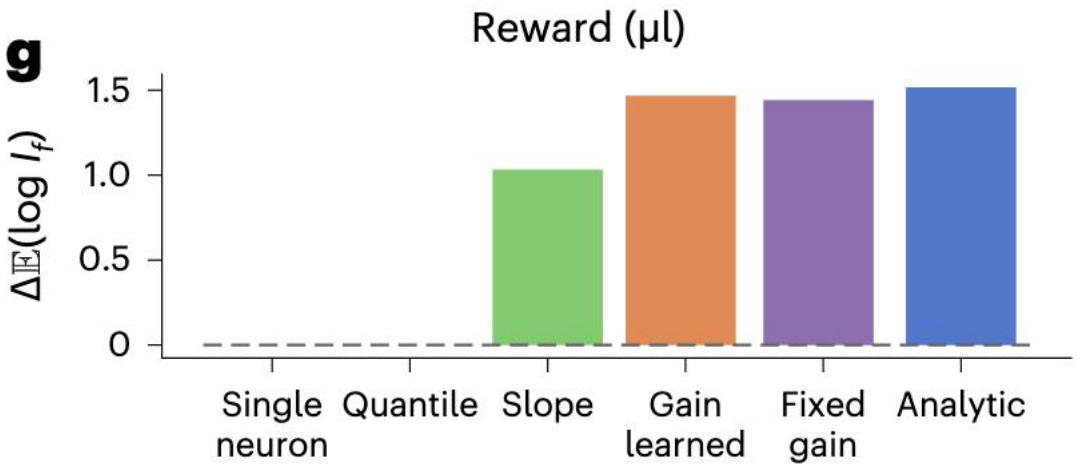


Gain Method 2

Fixed gain per neuron, determined by quantile



Comparison


g
Reward (µl)
1.5
1.0
0.5
0
Single neuron
Quantile
Slope
Gain learned
Fixed gain
Analytic

Efficient Code computation Details

We start by defining a ‘standard’ population of RPENs

Efficient Code computation

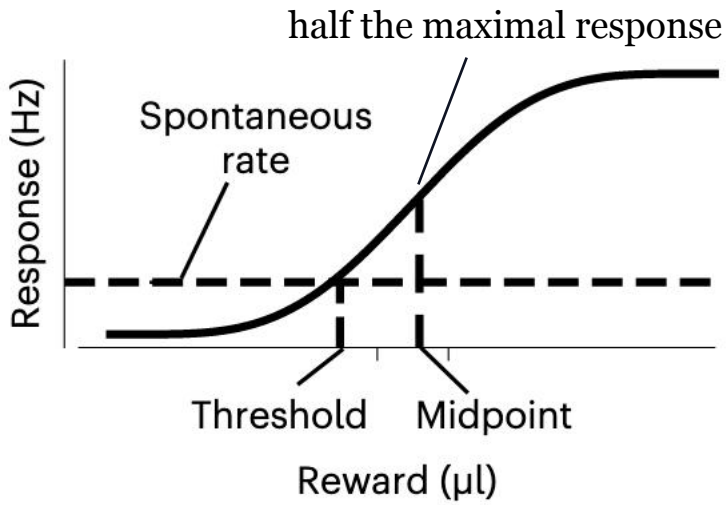
Discussion

Task design: No action required by the animal

the reward received and the change in value of the current state are confounded

Discussion

$$h_\mu(R) = h_0[F(R) - F(\mu)]g(\mu)$$



RPEN's sigmoid tuning curve

- Response function, midpoint = μ
- Reward mapping $F : R \to [0, 1]$
- A prototypical sigmoidal function h_0
- Neural Gain g as a function of R

We assumed that every neuron with a midpoint = μ is the same.

Discussion

In classical reinforcement learning, we set the same learning rate for negative PE and positive PE.

What about setting two different ones respectively? Would that help in better fitting behavioral data, or extract some latent variables for further analysis?