Reward prediction error neurons implement an efficient code for reward

Dongjae Kim[1,2,3†], Heiko H. Schütt[1,2†] and Wei Ji Ma[1,2*]

[1*]Center for Neural Science, New York University, 4 Washington place, New York, 10003, NY, USA.
[2]Department of Psychology, New York University, 4 Washington place, New York, 10003, NY, USA.
[3]Department of AI-Based Convergence, Dankook University, 152 Jukjeon-ro, Yongin-si, 16890, Gyeonggi-do, Republic of Korea.

*Corresponding author(s). E-mail(s): weijima@nyu.edu;
Contributing authors: dongjaekim@dankook.ac.kr;
heiko.schuett@nyu.edu;
†These authors contributed equally to this work.

Abstract

We apply efficient coding principles to derive the optimal population of neurons to encode rewards from a distribution. Similar to this optimal population, dopaminergic reward prediction error neurons have a broad distribution of optimistically placed thresholds, neurons with higher thresholds have higher gain and the curvature of their responses depends on the threshold. Thus, these neurons may broadcast an efficient reward signal, not necessarily a reward prediction error.

Keywords: distributional reinforcement learning, efficient coding, value, reward, reward prediction error

Main

Reward processing is central for a wide range of cognitive tasks including decision making, planning and learning. One important reward representation in the brain is maintained by reward prediction error neurons (RPENs) [1].

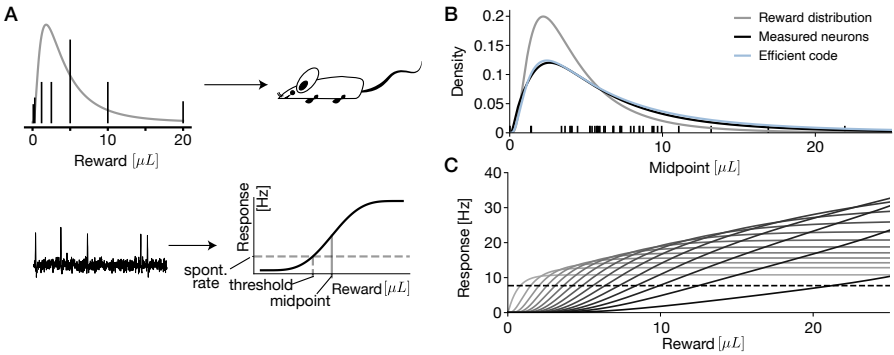

Fig. 1 Experimental design and basic results. **A**: Experimental design: a mouse is presented rewards from a distribution, while the responses of RPENs are recorded to create a tuning curve for reward. In the experiment by Eshel et al.[12], the rewards were chosen from a discrete distribution, which we approximate with the displayed moment-matched log-normal distribution for our analyses. **B**: Log-normal kernel density estimates for the density of neuron midpoints for the optimal population (cyan) and the measured population (black), replotting the continuous approximation to the reward distribution (gray). **C**: Efficient code with 15 sigmoidal neurons for the reward distribution. The dashed line represents the fitted spontaneous fitting rate.

These dopaminergic neurons in the midbrain react to received rewards relative to an expectation based on past experience. The existence of these neurons is the most prominent evidence in favor of reinforcement learning (RL) in the brain [2]. Also, RPENs have been implied in a broad range of tasks that require value-based cognition[3]. Thus, RPENs' encoding of reward is a cornerstone of our understanding of reward signals in the brain.

One guiding principle to explain neural codes is the hypothesis that neuronal codes might be optimized for *efficient coding*, i.e. to convey as much information as possible with a given budget for the number of neurons and the number of spikes [4, 5]. This efficient coding hypothesis is quite successful at predicting the codes employed by sensory neurons [6–8], but this framework is not limited to sensory processing and can be applied to reward encoding, too. Notably, efficient coding has been applied successfully to rewards to explain preference-based decisions [9–11], which corroborates the idea that rewards are represented using an efficient code somewhere in the brain.

Here, we propose that RPENs implement an efficient code for reward value. To test our hypothesis, we first derive the most efficient population of sigmoidally tuned neurons to encode rewards sampled from an arbitrary given distribution. We then compare this optimized population to a population of RPENs measured by Eshel et al. [13, 14]. We find three aspects of the optimized population confirmed: First, RPEN thresholds cover the range of the distribution with an emphasis on higher thresholds. Second, RPENs with higher thresholds have higher gains. And third, the asymmetry of RPENs' responses around their thresholds depends on their threshold switching from concave to convex with growing threshold.

We derive the optimal population of neurons to encode rewards analytically by extending an existing framework by Ganguli and Simoncelli [15]. Assuming that RPENs have a sigmoid tuning curve for reward [16, 17], we start with a broad class of populations, among which we search for the most efficient one. Here, we construct the class based on a base population of neurons on the unit interval. We then allow any smooth, monotone function to map from rewards to the unit interval and define the responses of the neurons in terms of the responses of the base population at the mapped location. Additionally, we allow an arbitrary scaling of the neurons' response gain and of the density of neurons depending on their placement. In mathematical terms, we use a prototypical sigmoidal response function h_0 on the unit interval to define the response of a neuron with midpoint at μ as:

$$h_\mu(R) = h_0(F(R) - F(\mu))g(\mu), \tag{1}$$

where $F(R) = \int_{-\infty}^{R} f(t)dt$ is the mapping from reward space to the unit interval with derivative $f \geq 0$, which gives the local conversion rate between reward space and the unit interval. g is the gain of neurons placed at μ. Finally, d is the density of neurons, i.e. how likely it is that a neuron is placed at a given reward.

We then optimize the three functions to find the most efficient populations that can be generated this way to encode rewards from a distribution with density $p(R)$ and cumulative density $P(R)$ (see Online Methods). This yields a family of equally good populations, which we parameterized by $\alpha \in [0, 1]$:

$$f(R) = p(R) \qquad d(R) \propto \frac{p(R)^\alpha}{(1 - P(R))^{1-\alpha}} \qquad g(R) \propto \frac{p(R)^{1-\alpha}}{(1 - P(R))^\alpha} \tag{2}$$

This solution extends the earlier result [15], which assumed a uniform density of neurons on the unit interval, effectively setting $d(R) = f(R)$ in our formulation. Their solution is a special case of our family with $\alpha = 1$ and thus $d(R) = p(R)$ and $g(R) = \frac{1}{1-P(R)}$.

We compare the predictions of the efficient coding model to data from the 'variable-reward task' of Eshel et al. [13] (Fig. 1A). In this task, mice were given one of seven reward magnitudes in each trial with certain probabilities, while RPENs responses were recorded. We optimize the efficient code for a continuous approximation of the discrete probability distribution, in the form of a log-normal distribution matching the mean and variance of the discrete distribution (μ=1.29 and σ^2=0.71). To compare our predictions to the data, we set free parameters of the derived population to match the population observed in the neural recordings. We used $n = 39$ neurons with a summed expected firing rate of $r_{\max} = 245.41$, and set the spontaneous firing rate r^*, the parameter of our family of solutions α, and k a parameter controlling the overall slope to optimize the match to the empirically observed midpoints and

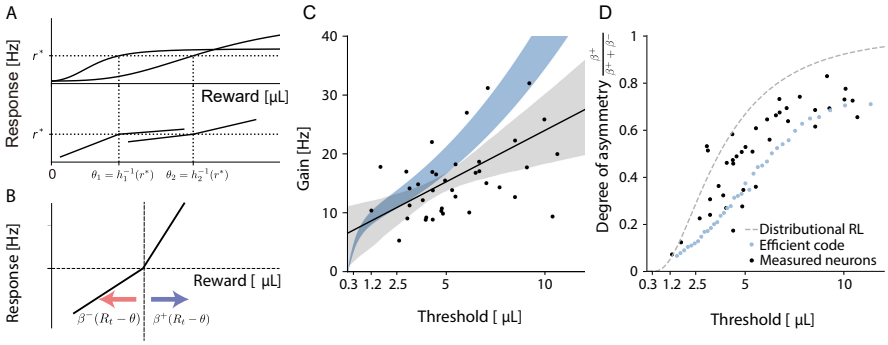

Fig. 2 Gain and asymmetry of tuning curves co-vary with the threshold in both the efficient code and measured RPENs. **A**: Efficient coding explanation for the relationship between asymmetry and threshold: The spontaneous activity r^* is reached at θ_1 within the concave part of the sigmoidal response function for a neuron with low threshold and gain, and at θ_2 within the convex part of a sigmoidal response function for a neuron with high threshold and gain. **B**: Asymmetric TD learning or distributional RL explanation for the relationship between asymmetry and threshold: A reward value R_t shift the response function down proportional to β^- if below the threshold ($R_t < \theta$) and up proportional to β^+ if above the threshold ($R_t \geq \theta$). An equilibrium is achieved when the two expected shifts cancel each other. **C**: RPEN gain, i.e. maximal response of a fitted sigmoid function, plotted against threshold. The black line gives a linear regression fit ($r(37) = 0.527$, $p < 0.001$). The colored lines give the relationship in the efficient code with a 90% confidence interval due to our uncertainty about the overall gain. **D**: Degree of asymmetry of neural responses ($\frac{\beta^+}{\beta^- + \beta^+}$) plotted against threshold. The neurons of an efficient population (light blue) closely approximate the the measured population (black). The dashed line is expected from the ideal distributional TD learning.

thresholds, resulting in $r^* = 7.67$, $\alpha = 0.77$, and $k = 5.07$ (see Online Methods for details).

The midpoints of neurons in the optimized population cover the range of the reward distribution, with some emphasis on higher quantiles (Fig. 1C). This pattern represents a trade-off between two pulls on the efficient code: Neurons generally need to be distributed relatively uniformly across the encountered reward distribution, because investing more spikes to encode a range of rewards more accurately yields less additional information transfer the more accurate the reward value is already encoded. However, neurons with higher thresholds are activated by fewer reward values and are thus cheaper. This pattern matches the measured population (Fig. 1C).

In the efficient population of sigmoidal neurons, the gain is higher for neurons with higher reward thresholds. This effect does not happen for unimodal tuning functions [15, 18]. The increase in gain is optimal, because neurons with high thresholds respond to fewer rewards and can thus afford a higher gain with the same expected number of spikes. In the data, we observe the expected increase of gain, albeit slightly weaker than expected (Fig. 2C).

The higher gain for neurons with higher thresholds leads to concave tuning curves for neurons with low thresholds and convex tuning curves for neurons with high thresholds (Fig. 2A). This effect is due to the definition of the

threshold as the point where the spontaneous firing rate is surpassed. As long as the gain increases faster than the spontaneous firing rate, the threshold will move down relative to the sigmoid shape moving from the upper concave part of the sigmoid down into the convex part.

To quantify the degree of asymmetry of neural responses, we followed previous work and approximated the neural tuning curves with two linear response functions, one above and one below the spontaneous firing rate r^*. This allows for an asymmetric response of the RPENs, with the slope above threshold β^+ being different from the slope below threshold β^- [19, 20]. The ratio of slopes $\frac{\beta^+}{\beta^- + \beta^+} \in [0, 1]$ can then be used as an index of asymmetry, which is close to one for convex tuning curves and close to 0 for convex tuning curves.

The relationship between asymmetry on threshold was similar in the optimal population and in the measured population (Fig. 2D). This relationship was captured even better than by distributional reinforcement learning [14] (dashed line in Fig. 2D), a recent mechanistic explanation that linked the asymmetry around threshold to the asymmetry of the update equations necessary to converge to different expectiles of the reward distribution.

We thus conclude that RPENs indeed implement an efficient code for reward. This proposal provides a normative reason for the tuning of RPEN responses to rewards, in particular for the following observations: First, neurons encode reward relative to a broad range of quantiles of the reward distribution with some bias towards higher thresholds. Second, neurons with higher reward thresholds have higher gain than neurons with lower thresholds. Third, neurons with high (low) thresholds have convex (concave) tuning curves.

Our qualitative observations are all robust to changing the measure of information used for optimizing the population, the assumed distribution of neural responses or the sigmoid shape. Different objective functions or different response distributions lead to different relationships between the density of the reward distribution and the optimal density and gain of neurons[15]. However, the relationship to neural density is always monotonically increasing, as it is more efficient to focus neural resources on probable rewards. Analogously, all objectives lead to higher gains at higher thresholds, as gain increases are "cheaper" for neurons with higher thresholds. Furthermore, different shapes for the response functions do not change the density and gain predictions, which were derived independent of the shape, and only lead to slightly different predictions for the asymmetry of the responses around the threshold. For any shape that is convex in its lower response range and concave in its upper response range, the qualitative argument for the dependence between threshold and asymmetry will hold.

Typically, RPEN responses are described in terms of reward prediction error instead of rewards as we do here. However, efficient coding always assumes that the reward distribution is known by both sender and receiver, such that encoding the expected reward does not communicate additional information. As a consequence, an efficient code does not encode the expected

value and the optimal populations for reward and for reward prediction error are exactly equal. From this one code, a receiver can then decode both value and reward prediction error.

To simplify our analysis, we restricted ourselves to a task with a single reward supplied in each independent episode without any action required by the animal. In this simple case, the reward received and the change in value of the current state (potentially with temporal discounting) are confounded. Thus, all our analyses are agnostic to this distinction. Earlier literature suggests that RPENs take future rewards into account with some temporal discounting in multi-step settings[21] and delayed reward tasks[22]. These observations speak in favor of a temporally discounted value interpretation rather than a pure reward interpretation. Distinguishing these concepts and explaining the ramping and phasic activities of RPENs[e.g. 23], will require future research.

There are three alternative explanations of the data by Eshel et al. [13] we are aware of: The distributional RL expectile explanation [14], the Laplace code [24], and normalized reinforcement learning [25]. These reinforcement learning-based explanations primarily give a mechanistic explanation how RPENs learn their code. Distributional RL[14] proposes that RPENs update their thresholds with the same asymmetry shown by their response profile, which leads to convergence to different expectiles of the distribution. This induces the relationship between threshold and asymmetry. The Laplace code[24] proposes that TD-learning neurons with different parameters can be used to encode temporal aspects and the whole distribution of rewards by applying an analogue of the Laplace transform to the neural responses. In the paper on the Laplace code, the authors also already note that the dependence between asymmetry and threshold can be created by cutting a sigmoid response curve at different response levels, although they moved the sigmoid up and down instead of the gain change we predict and observe. Finally, normalized reinforcement learning [25] proposes that reward encoding neurons perform divisive normalization with different half saturation constants. This yields sigmoid (Naka-Rushton non-linearity [26]) neurons with different thresholds and the asymmetry around those thresholds is explained again by cutting the sigmoids at different heights, similar to the explanation for the Laplace code. All these earlier explanations are not normative but mechanistic, do not note or explain the increase in gain and one would not expect the bias towards higher thresholds from these theories.

Efficient coding as a normative account is compatible with any mechanistic theory how the code is implemented and learned. Classically, the efficient coding literature has ignored learning and studied the converged steady state for distributions that are learned over a lifetime. Some recent research proposed mechanisms how neurons may learn efficient codes[e.g. 27–30]. However, the mechanistic explanations for the asymmetry threshold relationship we discussed above[24, 25] could most likely be modified to converge to an efficient code for rewards if neural gain was also adjusted and the right distribution of

neuron parameters were chosen. Making these learning rules concrete and comparing them to the efficient code learning proposals is an interesting direction for future research.

Acknowledgments. We thank Hsin-Hung Li for valuable discussions.

References

[1] Schultz, W., Dayan, P. & Montague, P. R. A neural substrate of prediction and reward. *Science* **275** (5306), 1593–1599 (1997) .

[2] Sutton, R. S. & Barto, A. G. *Reinforcement learning: An introduction* (2018).

[3] Balleine, B. W., Daw, N. D. & O'Doherty, J. P. in *Chapter 24 - multiple forms of value learning and the function of dopamine* (eds Glimcher, P. W., Camerer, C. F., Fehr, E. & Poldrack, R. A.) *Neuroeconomics* 367–387 (Academic Press, London, 2009).

[4] Attneave, F. Some informational aspects of visual perception. *Psychological Review* **61** (3), 183–193 (1954). https://doi.org/10.1037/h0054663 .

[5] Barlow, H. B. in *Possible Principles Underlying the Transformations of Sensory Messages* (ed.Rosenblith, W. A.) *Sensory Communication* 216–234 (The MIT Press, 1961).

[6] Laughlin, S. A Simple Coding Procedure Enhances a Neuron's Information Capacity. *Zeitschrift für Naturforschung C* **36** (9-10), 910–912 (1981). https://doi.org/10.1515/znc-1981-9-1040 .

[7] Schwartz, O. & Simoncelli, E. P. Natural signal statistics and sensory gain control. *Nature Neuroscience* **4** (8), 819–825 (2001). https://doi.org/10.1038/90526 .

[8] Wei, X.-X. & Stocker, A. A. Lawful relation between perceptual bias and discriminability. *Proceedings of the National Academy of Sciences* **114** (38), 10244–10249 (2017). https://doi.org/10.1073/pnas.1619153114 .

[9] Louie, K., Glimcher, P. W. & Webb, R. Adaptive neural coding: From biological to behavioral decision-making. *Current Opinion in Behavioral Sciences* **5**, 91–99 (2015). https://doi.org/10.1016/j.cobeha.2015.08.008 .

[10] Polanía, R., Woodford, M. & Ruff, C. C. Efficient coding of subjective value. *Nature Neuroscience* **22** (1), 134–142 (2019). https://doi.org/10.1038/s41593-018-0292-0 .

[11] Bhui, R., Lai, L. & Gershman, S. J. Resource-rational decision making. *Current Opinion in Behavioral Sciences* **41**, 15–21 (2021). https://doi.org/10.1016/j.cobeha.2021.02.015 .

[12] Eshel, N., Tian, J., Bukwich, M. & Uchida, N. Dopamine neurons share common response function for reward prediction error. *Nature Neuroscience* **19** (3), 479–486 (2016). https://doi.org/10.1038/nn.4239 .

[13] Eshel, N. *et al.* Arithmetic and local circuitry underlying dopamine prediction errors. *Nature* **525** (7568), 243–246 (2015) .

[14] Dabney, W. *et al.* A distributional code for value in dopamine-based reinforcement learning. *Nature* **577** (7792), 671–675 (2020). https://doi.org/10.1038/s41586-019-1924-6 .

[15] Ganguli, D. & Simoncelli, E. P. Efficient Sensory Encoding and Bayesian Inference with Heterogeneous Neural Populations. *Neural Computation* **26** (10), 2103–2134 (2014). https://doi.org/10.1162/NECO_a_00638 .

[16] Fiorillo, C. D., Tobler, P. N. & Schultz, W. Discrete coding of reward probability and uncertainty by dopamine neurons. *Science* **299** (5614), 1898–1902 (2003) .

[17] Cohen, J. D. & Servan-Schreiber, D. A theory of dopamine function and its role in cognitive deficits in schizophrenia. *Schizophrenia bulletin* **19** (1), 85–104 (1993) .

[18] Wei, X.-X. & Stocker, A. A. in *Bayesian Inference with Efficient Neural Population Codes* (eds Hutchison, D. *et al.*) *Artificial Neural Networks and Machine Learning – ICANN 2012*, Vol. 7552 523–530 (Springer Berlin Heidelberg, Berlin, Heidelberg, 2012).

[19] Frank, M. J., Seeberger, L. C. & O'reilly, R. C. By carrot or by stick: cognitive reinforcement learning in parkinsonism. *Science* **306** (5703), 1940–1943 (2004) .

[20] Mikhael, J. G. & Bogacz, R. Learning reward uncertainty in the basal ganglia. *PLoS computational biology* **12** (9), e1005062 (2016) .

[21] Kobayashi, S. & Schultz, W. Influence of reward delays on responses of dopamine neurons. *Journal of neuroscience* **28** (31), 7837–7846 (2008) .

[22] Roesch, M. R., Calu, D. J. & Schoenbaum, G. Dopamine neurons encode the better option in rats deciding between differently delayed or sized rewards. *Nature neuroscience* **10** (12), 1615–1624 (2007) .

[23] Kim, H. R. *et al.* A Unified Framework for Dopamine Signals across Timescales. *Cell* **183** (6), 1600–1616.e25 (2020). https://doi.org/10.1016/j.cell.2020.11.013 .

[24] Tano, P., Dayan, P. & Pouget, A. A Local Temporal Difference Code for Distributional Reinforcement Learning. *Advances in Neural Information Processing Systems* **34**, 12 (2020) .

[25] Louie, K. Asymmetric and adaptive reward coding via normalized reinforcement learning. Preprint, Neuroscience (2021).

[26] Naka, K. I. & Rushton, W. A. H. An attempt to analyse colour reception by electrophysiology. *The Journal of Physiology* **185** (3), 556–586 (1966). https://doi.org/10.1113/jphysiol.1966.sp008002 .

[27] Bredenberg, C., Simoncelli, E. P. & Savin, C. Learning efficient task-dependent representations with synaptic plasticity. Preprint, Neuroscience (2020).

[28] Savin, C. & Triesch, J. Emergence of task-dependent representations in working memory circuits. *Frontiers in computational neuroscience* **8**, 57 (2014) .

[29] Gerstner, W., Lehmann, M., Liakoni, V., Corneil, D. & Brea, J. Eligibility traces and plasticity on behavioral time scales: experimental support of neohebbian three-factor learning rules. *Frontiers in neural circuits* **12**, 53 (2018) .

[30] Frémaux, N. & Gerstner, W. Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules. *Frontiers in neural circuits* **9**, 85 (2016) .

[31] Wei, X.-X. & Stocker, A. A. A Bayesian observer model constrained by efficient coding can explain 'anti-Bayesian' percepts. *Nature Neuroscience* **18** (10), 1509–1517 (2015). https://doi.org/10.1038/nn.4105 .

[32] Brunel, N. & Nadal, J.-P. Mutual Information, Fisher Information, and Population Coding. *Neural Computation* **10** (7), 1731–1757 (1998). https://doi.org/10.1162/089976698300017115 .

[33] Cover, T. M. & Thomas, J. A. *Elements of Information Theory* Wiley Series in Telecommunications (Wiley, New York, 1991).

[34] Wei, X.-X. & Stocker, A. A. Mutual Information, Fisher Information, and Efficient Coding. *Neural Computation* **28** (2), 305–326 (2016). https://doi.org/10.1162/NECO_a_00804 .

[35] Bethge, M., Rotermund, D. & Pawelzik, K. Optimal Short-Term Population Coding: When Fisher Information Fails. *Neural Computation* **14** (10), 2317–2351 (2002). https://doi.org/10.1162/08997660260293247 .

[36] Nash, S. G. Preconditioning of truncated-newton methods. *SIAM Journal on Scientific and Statistical Computing* **6** (3), 599–616 (1985) .

Online Methods

Analysis of neural data

We used the data from the variable-reward task by [13] as preprocessed by [14]. For details of data acquisition such as data inclusion and identifying neurons, please refer to [13]. The recording techniques are described in detail in [17]. We only give a brief summary of important information we took from their articles:

In this task, five mice were given one of seven reward magnitudes (0.1, 0.3, 1.2, 2.5, 5, 10 and 20 µl) randomly chosen in each trial with probabilities 0.066, 0.091, 0.15, 0.15, 0.31, 0.15, and 0.077, respectively. Electrophysiological data were recorded from $n = 40$ ventral tegmental area dopaminergic RPENs. From each of the mice 3, 6, 9, 16, and 6 neurons were selected respectively. One neuron was excluded because it had never shown a single response larger than its baseline for any reward magnitude.

Following previous analyses of this dataset [14], we used average spike rates from 200 ms to 600 ms as a response. Data were originally binned into 1 ms bins. The resulting peri-stimulus time histograms were then smoothed by convolving with the function $\left(1 - e^{-t/\mathrm{ms}}\right) e^{-t/(20\mathrm{ms})}$.

Logistic fits

To extract the parameters predicted by the efficient coding framework from the data, we fitted logistic sigmoid functions $\sigma(x) = \frac{a_1}{1-e^{-a_2\,(x-a_3)}}$ to the data of each neuron. This allows us to extract a gain a_1, a slope a_2, and a midpoint a_3 for each neuron. We used least-squares minimization to estimate parameters of the sigmoid functions: $\hat{\boldsymbol{a}} = \underset{\boldsymbol{a}}{\operatorname{argmin}} \sum_{j=1}^{T} (R_j - \sigma(x_j;\ \boldsymbol{a}))^2$ where R_j is the measured firing rate of a neuron responding to the reward magnitude x_j at j-th trial of T trials.

The one parameter that influences the efficient code is the overall expected firing rate r_{max}. For this parameter, we computed a 90% confidence interval of $[209.60, 264.21]$ by bootstrap resampling neurons and measured trials for them $5,000$ times. As all firing rates in the efficient code scale proportionally to the expected firing rate, this confidence interval directly translates into a confidence interval for the gain at any threshold (Fig. 2C).

Fitting of asymmetric slopes

To enable comparisons to the data by [14], we repeated their analysis on their data and apply the same analysis technique to the simulated neurons in our population optimized for efficient coding.

First, we subtracted a baseline from the smoothed PSTHs for each neuron and each trial. The baseline was the mean firing rate over the 1000 ms before stimulus onset. We then estimated asymmetric slopes, η^+ and η^- by separately fitting linear function to the responses above and below threshold for each neuron. Following Dabney et al. [14], we used estimated utility space as x-axis for this step, but the results are not sensitive to the choice of utility function.

In math terms, the slope above threshold η^+ and the slope below threshold η^- for responses y_j to rewards R_j are thus defined as:

$$\eta_i^+ = \underset{\eta_i}{\operatorname{argmin}} \sum_{j=1}^{T} \left[r^* + \eta_i \left(R_j - \theta\right) - y_j\right]^2, \text{ for } R_j \geq \theta$$

$$\eta_i^- = \underset{\eta_i}{\operatorname{argmin}} \sum_{j=1}^{T} \left[r^* + \eta_i \left(R_j - \theta\right) - y_j\right]^2, \text{ for } R_j \leq \theta,$$

where r^* is the spontaneous firing rate and the sum over j runs over the T trials.

To achieve more efficient fitting than Dabney et al. [14], we derive analytic solutions for η^+ and η^- for a given θ, which are:

$$\eta_i^+ = \frac{\sum_{j=1}^{T}(R_j - \theta)(y_j - r^*)}{\sum_{j=1}^{T}(R_j - \theta)^2} \text{ for } R_j \geq \theta \tag{3}$$

$$\eta_i^- = \frac{\sum_{j=1}^{T}(R_j - \theta)(y_j - r^*)}{\sum_{j=1}^{T}(R_j - \theta)^2} \text{ for } R_j \leq \theta. \tag{4}$$

Thresholds – called reversal points in [14] – were then chosen to minimize the overall sum of squares, by sampling 20,000 thresholds from an uniform distribution between the range of rewards and choosing the best fitting one. Dabney et al. used the same random sampling procedure for η^+ and η^-.

For fitting our simulated neurons, we set the threshold $\theta = h_i^{-1}(r^*)$ and $y_j = h_i(v_j)$ and fit the ηs based on the results for 10^6 trials sampled from the approximated log-normal reward distribution.

Derivation of the efficient code

To derive the efficient code for reward, we extend the framework for analytic solutions proposed by Ganguli and Simoncelli [15]. In this framework, a uniform standard population is defined on the unit interval $[0, 1]$ [1]. By using an adaptable mapping between the unit interval and the stimulus space and functions that transform the population, we can define a broad class of populations among which we can find the most efficient ones analytically. We extend this framework by allowing the distribution of neurons to be non-uniform , which decouples the local slope and density of neurons that are coupled in the original framework.

Definitions

We start by defining a "standard" population of RPENs. This standard population is a collection of sigmoidal tuning curves $s_\omega : [0, 1] \to [0, 1]$ indexed by their 'center' $\omega \in [0, 1]$. Individual neurons are assumed to be monotonically increasing functions whose derivatives are unimodal functions with their peak near ω. Additionally, we assume that the overall Fisher information $I_f(x)$ of this population about the position $x \in [0, 1]$ and the overall increase in firing rate are independent of x i.e.:

$$\int s'_\omega(x) d\omega \approx \text{const} \tag{5}$$

$$I_f(x) = \int I_f(x, \omega) d\omega \approx \text{const}, \tag{6}$$

where $I_f(x, \omega)$ is the Fisher information provided by the tuning function with center ω and $s'_\omega(x)$ is the derivative of the tuning function.

Finally, we assume that the total amount of Fisher information provided by each neuron is approximately constant:

$$\int I_f(x, \omega) dx \approx \text{const} \tag{7}$$

One simple way to construct a population that fullfills these criteria is to start with a prototypical sigmoid function s_0 centered on 0 and defining the other tuning curves as $s_\omega(x) = s_0(x - \omega)$. This construction often leads to strong edge effects at 0 and 1, because a substantial part of the sigmoid response curves near the borders will fall outside the $[0, 1]$ interval. We ameliorate these effects by using a more sophisticated population based on cumulative beta distributions below. Nonetheless, this simple construction illustrates the intuition behind a uniform population of sigmoidal neurons well and was actually used by Ganguli and Simoncelli in their implementation.

To map between the unit interval and stimulus space we use a strictly monotonic function $F(R) = \int_{-\infty}^{R} f(s) ds$ from the stimulus space into the unit

[1] Ganguli and Simoncelli [15] used the interval $[0, N]$, where N is the number of neurons, which is entirely equivalent.

interval, parameterized by its derivative $f(s) > 0$, which we will optimize below. As F is strictly monotonic, it is also invertible, such that this also defines a function F^{-1} mapping the unit interval into stimulus space. Additionally, we allow a gain function $g > 0$ of stimulus space, which scales the tuning curve of neurons placed at different positions in stimulus space.

With these definitions, we can now define the tuning function h_μ of a neuron placed at stimulus level μ:

$$h_\mu(R) = g(\mu) s_{F(\mu)}(F(R)), \tag{8}$$

i.e. we map both the positions of the neurons and the stimuli into the unit interval, evaluate the sigmoids there and scale the result with the gain.

Up to this point our definitions are equivalent to those of Ganguli and Simoncelli [15], who now assume that the neurons are equally spaced in the interval. This additional assumption makes it impossible to place more neurons in some stimulus range without changing their shape. This removes one of the possible adaptations of a neural populations mentioned by other authors like Wei and Stocker [31] for example. To allow different distributions of the neurons, we instead assume that the μ_i are drawn from a distribution over reward space with density d, which we will optimize below. The equal placement assumed by Ganguli and Simoncelli is essentially equivalent to the assumption $d(R) = f(R)$, which yields a uniform distribution over the placements in the unit interval ω.

Optimization Objective

We further assume that the firing rates of the RPENs are subject to independent Poisson noise, which is the simplest noise assumption. We can then calculate the Fischer information $I_f(R;\ \mu)$ provided by a neuron centered on μ:

$$I_f(R;\ \mu) = g(\mu) \frac{s'_{F(\mu)}(F(R))^2}{s_{F(\mu)}(F(R))} f^2(R). \tag{9}$$

We then optimize the expected value (over both R and μ) of the logarithm of the Fisher information I_0, based on N neurons:

$$I_N = \mathbb{E}_R \mathbb{E}_\mu \left[\log \left(\frac{1}{N} \sum_i I_f(R;\ \mu_i) \right) \right] \tag{10}$$

This objective was originally proposed as a lower bound on the mutual information between the neural responses and the stimulus [15, 32, 33], which becomes tight for small Gaussian shaped errors on the encoded value. However, some later publications cast doubt on this interpretation: In particular [34] interpret this equation as an upper bound on mutual information instead. Additionally, small firing rates and/or steep changes in firing rate can lead to this approximation being highly inaccurate and this realm is actually reached for cases in which neurons are measured only for short times [35]. Nonetheless,

I remains the best Fisher information based approximation to mutual information and we follow Ganguli and Simoncelli [15] in optimizing it. They also tried a few other functionals of Fisher information and got qualitatively similar results. Mutual information itself is unfortunately untractable for these derivations.

Assuming further that there is a large population of RPENs, we can now look at this distribution in the limit of infinitely many neurons. According to the strong law of large numbers, the logarithm of the mean in (10) almost surely converges to the logarithm of the expected Fisher information, because the Fisher information provided by each single neuron has finite variance:

$$\lim_{N\to\infty} \log\left(\frac{1}{N}\sum_i I_f(R;\ \mu_i)\right) = \log \mathbb{E}_\mu\left[I_f(R;\ \mu)\right] \tag{11}$$

Because this is a constant with respect to μ, we can drop the outer expected value, yielding:

$$I \equiv \lim_{N\to\infty} I_N = \mathbb{E}_R\left[\log\left(\mathbb{E}_\mu\left[I_f(R;\ \mu_i)\right]\right)\right] \tag{12}$$

To make this expression amenable to optimization, we simplify it as follows:

$$\mathbb{E}_R\left[\log\left(\mathbb{E}_\mu\left[I_f(R;\ \mu)\right]\right)\right] \tag{13}$$

$$= \int_{-\infty}^{\infty} p(R)\log\left(\int_{-\infty}^{\infty} d(\mu)g(\mu)\frac{s'_{F(\mu)}(F(R))^2}{s_{F(\mu)}(F(R))}f^2(R)d\mu\right)dR \tag{14}$$

$$\approx \int_{-\infty}^{\infty} p(R)\log\left(d(R)g(R)f(R)\int_{-\infty}^{\infty} f(\mu)\frac{s'_{F(\mu)}(F(R))^2}{s_{F(\mu)}(F(R))}d\mu\right)dR \tag{15}$$

$$\approx \text{constant} + \int_{-\infty}^{\infty} p(R)\log\left(d(R)g(R)f(R)\right)dR. \tag{16}$$

Here, we eliminated the inner integral through two steps, which are only approximately correct: First, we observe for sigmoidal functions, that $\frac{h'_0(F(R)-F(\mu))^2}{h_0(F(R)-F(\mu))}$ is a unimodal function that is centered around R, which effectively acts as a smoothing kernel that is convolved with the rest of the integral. If $d(\mu), f(\mu)$ and $g(\mu)$ change little over the range of this smoothing kernel, we can replace them by their values at R and pull them out of the integral. Second, the integral over μ we are left with is (approximately) shift-invariant, i.e. the same for all R and invariant to the monotonic transformation F, if the original population is (approximately) uniform as defined in equations (6) & (7).

The final optimization problem then is to optimize this approximation of I with respect to the three functions d, g, and f:

$$I \approx \int_{-\infty}^{\infty} p(R) \log\left(d(R)g(R)f(R)\right) dR \tag{17}$$

Additionally, we enforce that the expected number of spikes $\mathbb{E}[r]$ under the prior distribution is smaller or equal to a bound $r_{\max}$. The expected number of spikes is:

$$\begin{aligned}
\mathbb{E}[r] &= N\mathbb{E}_R\left[\mathbb{E}_\mu[h_\mu(R)]\right] && (18)\\
&= N\int_{-\infty}^{\infty} p(R) \int_{-\infty}^{\infty} d(\mu)g(\mu)s_{F(\mu)}(F(R))d\mu dR && (19)\\
&\approx N\int_{-\infty}^{\infty} d(\mu)g(\mu) \int_{\mu}^{\infty} p(R)dRd\mu && (20)\\
&= N\int_{-\infty}^{\infty} d(\mu)g(\mu)(1-P(\mu))d\mu, && (21)
\end{aligned}$$

where P is the cumulative density function of the reward distribution. In (20), we approximated the sigmoid with a step function at μ. This approximation is close if $p(R)$ is near constant over the range where the sigmoid h_μ changes from 0 to 1. This is the case if the sigmoids are reasonably steep.

There are two further constraints, one to make d a valid probability density and one to limit the range of F to the unit interval:

$$D = \int_{-\infty}^{\infty} d(R)dR = 1, \qquad d(R) > 0 \tag{22}$$

$$F_{\max} = \int_{-\infty}^{\infty} f(R)dR = 1, \qquad f(R) > 0 \tag{23}$$

Solving the Optimization

Now we are ready to solve this optimization problem using Lagrange multipliers. To do so, we first compute the functional derivatives of our objective function:

$$\frac{\delta I}{\delta f}(R) = \frac{p(R)}{f(R)} \qquad \frac{\delta I}{\delta g}(R) = \frac{p(R)}{g(R)} \qquad \frac{\delta I}{\delta d}(R) = \frac{p(R)}{d(R)} \tag{24}$$

Similarly, we compute the functional derivatives of the constraints aiming for a Lagrange multiplier based solution. First, for the expected spike rate:

$$\frac{\delta \mathbb{E}[r])}{\delta f}(R) = 0 \qquad \frac{\delta \mathbb{E}[r])}{\delta g}(R) = (1-P(R))d(R) \tag{25}$$

$$\frac{\delta \mathbb{E}[r]}{\delta d}(R) = (1 - P(R))g(R) \tag{26}$$

Second, for the integral of the density D:

$$\frac{\delta D}{\delta f}(R) = 0 \qquad \frac{\delta D}{\delta g}(R) = 0 \qquad \frac{\delta D}{\delta d}(R) = 1 \tag{27}$$

And finally, for the upper end of the interval reached by F, $F_{\max}$:

$$\frac{\delta F_{\max}}{\delta f}(R) = 1 \qquad \frac{\delta F_{\max}}{\delta g}(R) = 0 \qquad \frac{\delta F_{\max}}{\delta d}(R) = 0 \tag{28}$$

With these derivations in place, we can finally solve the optimization using Lagrange multipliers $\lambda_R, \lambda_D, \lambda_F$, which result in the following three equations:

$$\frac{p(R)}{f(R)} - \lambda_F = 0 \tag{29}$$

$$\frac{p(R)}{g(R)} - \lambda_R d(R)(1 - P(R)) = 0 \tag{30}$$

$$\frac{p(R)}{d(R)} - \lambda_R g(R)(1 - P(R)) - \lambda_D = 0 \tag{31}$$

From (29), we immediately get:

$$f(R) = \frac{2p(R)}{\lambda_F} \propto p(R) \tag{32}$$

From (30), we obtain:

$$\lambda_R d(R)g(R)(1 - P(R)) = p(R) \tag{33}$$

$$d(R)g(R) = \frac{p(R)}{\lambda_R(1 - P(R))} \propto \frac{p(R)}{1 - P(R)} \tag{34}$$

By inserting this result into (31), we see that it can always be guaranteed with $\lambda_D = 0$.

We conclude that under a constraint on the expected number of spikes, the neural population that optimizes the Fischer information bound on mutual information has the following properties: The inverse slope of the sigmoids is proportional to the probability density. The product of the probability density for neurons with a given midpoint and the gain for neurons with this midpoint is equal to the ratio of the reward probability density and the probability that rewards larger than the midpoint appear.

The formulation by Ganguli & Simoncelli [15] with equally spaced neurons in the abstract space essentially enforces the additional constraint that $d(R) \propto f(R)$ and thus arrives at the special case of $d(R) \propto p(R)$ and $g(R) \propto (1 - P(R))^{-1}$.

Choosing a Solution

In our solution, d and g entirely compensate for each other. Due to the fact that their product needs to be proportional to a product of the two factors $p(R)$ and $\frac{1}{1-P(R)}$, a natural choice to parameterize a family of solutions is to use a parameter $\alpha \in [0, 1]$ that trades off the distribution of the two factors to density and gain:

$$d(R) = \frac{p(R)^{\alpha}}{(1 - P(R))^{1-\alpha}} \qquad g(R) = \frac{p(R)^{1-\alpha}}{(1 - P(R))^{\alpha}} \tag{35}$$

Effectively, α trades off the gain increase at higher thresholds against placing more neurons at higher thresholds. We illustrate this for several different reward distributions and values of α in Fig. 3.

We fitted α using only the midpoints of the measured neurons. To do so, we extracted the midpoints of the measured from logistic function fits as described below. We then estimated α by maximizing the probability of the measured neurons' midpoints under the density d. The best fit was produces by setting $\alpha = 0.77$.

The requirements we put forth in (5)-(7) are approximately fulfilled by many populations of sigmoid shaped tuning curves. Thus, the exact shape and in particular the slope of the sigmoid are not fixed by these efficient coding considerations. For all optimal populations in this paper we use the cumulative density functions of beta distributions, because these sigmoid functions are defined on the unit interval $[0, 1]$ and become exactly 0 and 1 at the ends of the interval, which yields good boundary behavior. These sigmoid functions are not exactly shifted versions of each other as we assumed in the derivations, but functions that are shifted versions of each other lead to relatively strong boundary effects that produce worse deviations from the theory. To place a neuron at a specific position $p \in [0, 1]$, we set the parameters such that p is the mean of the beta distribution ($a = kp, b = k(1 - p)$), which leaves us with one more parameter $n > 0$, which controls the slope of the sigmoid functions. As the efficient coding scheme does not determine what the overall slope should be, we set $k = 5.07$ to fit the threshold distribution of measured neurons as described below. We do not expect other shapes to behave fundamentally differently.

We used a fixed spontaneous firing rate r^* for all RPENs. We fitted this r^* to minimize the discrepancy between the measured thresholds and the ones predicted by the efficient code, using a truncated Newton algorithm [36]. The objective is minimizing the mean squared error between thresholds of the efficient code $\theta_i^* = h_i^{-1}(r^*)$ and thresholds of measured neurons θ_i with respect to r^* and an aforementioned parameter k for the slope of the sigmoid function $h(\cdot)$:

$$\underset{r^*,\, k}{\operatorname{argmin}}\, \mathbb{E}\left[\theta_i - h_i^{-1}(r^*;\ k)\right]^2 .$$

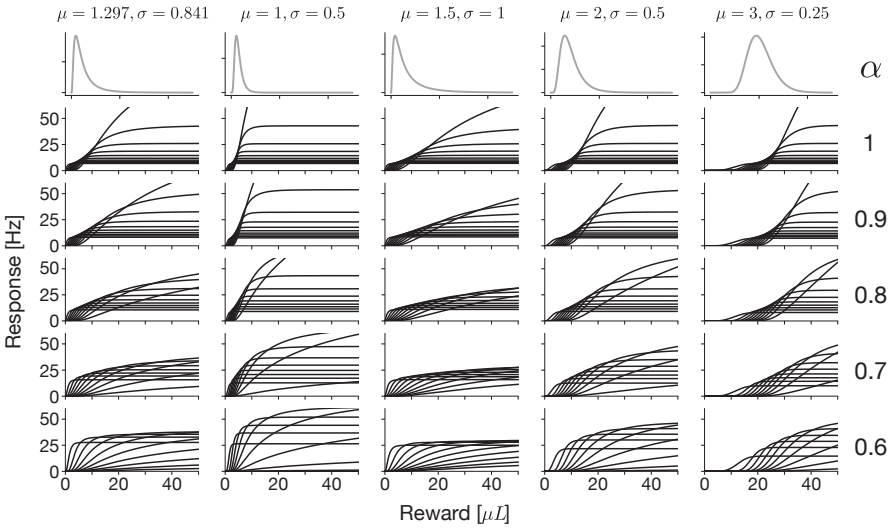

Fig. 3 Illustration of the solution of the efficient coding problem, varying α (rows) and the reward distribution (columns). The reward distributions are all log-normal distributions with their pdfs and parameters plotted at the top.

As a result, we found $r^* = 7.67$ and $k = 5.07$ for the efficient code that we fitted to the neurons measured by [13] used by [14].

Checking the solution

We performed several checks to confirm that the derived populations are indeed efficient given the reward distribution. To do so, we compared the amount of information communicated to some other populations with the same number of neurons and the same expected number of spikes (Fig. 4). The optimized populations retain more information than other populations. The population with the optimized $\alpha = 0.77$ is equally efficient as the solution of Ganguli and Simoncelli [15], which corresponds to $\alpha = 1$.

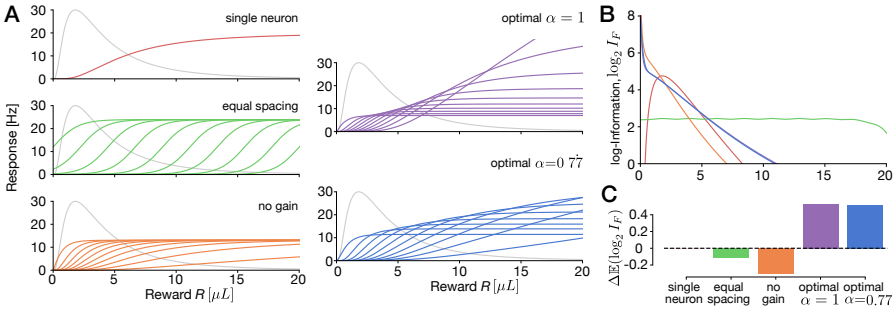

Fig. 4 Comparing encoding populations for reward with 10 neurons and the same expected number of spikes. **A**: Compared neuronal populations: single neuron: All neurons share the same response curve, optimized to maximize transferred information. equal spacing: neurons tile the space, not optimized. no gain: positions and slopes are optimized, but all neurons have equal gain. optimal $\alpha = 1$: fully optimized population as derived previously[15] with density proportional to the distribution. optimal $\alpha = 0.77$: Equally optimal distribution as derived here with α fit to match the midpoint distribution for the optimal code and the experimental data. **B**: Fisher information for each of the populations across reward. **C**: Expected logarithm of Fisher information under the reward distribution relative to the single-neuron case.