LETTER **Communicated by Yuri Dabaghian**

Decoding of Neural Data Using Cohomological Feature Extraction

Erik Rybakken
erik.rybakken@ntnu.no
Nils Baas
nils.baas@ntnu.no
Department of Mathematical Sciences, Norwegian University of Science and Technology, 7491 Trondheim, Norway

Benjamin Dunn
benjamin.dunn@ntnu.no
Kavli Institute for Systems Neuroscience, Norwegian University of Science and Technology, 7491 Trondheim, Norway

We introduce a novel data-driven approach to discover and decode features in the neural code coming from large population neural recordings with minimal assumptions, using cohomological feature extraction. We apply our approach to neural recordings of mice moving freely in a box, where we find a circular feature. We then observe that the decoded value corresponds well to the head direction of the mouse. Thus, we capture head direction cells and decode the head direction from the neural population activity without having to process the mouse's behavior. Interestingly, the decoded values convey more information about the neural activity than the tracked head direction does, with differences that have some spatial organization. Finally, we note that the residual population activity, after the head direction has been accounted for, retains some low-dimensional structure that is correlated with the speed of the mouse.

1 Introduction

The neural decoding problem is that of characterizing the relationship between stimuli and the neural response. For example, head direction cells (Ranck, 1985) respond with an elevated activity whenever the animal is facing a specific direction. To be able to determine this relationship, however, an experiment must be designed such that the relevant behavior of the animal can be properly sampled and tracked. Accurate tracking of the animal's complete behavior, including movements of the eyes, whiskers, head, body, and limbs, is extremely difficult in any freely behaving situation; external cues such as sounds in the frequency range of the animal, odors, or visual cues are also similarly difficult to account for appropriately, and yet all of

Neural Computation **31, 68–93** (2019)
doi:10.1162/neco_a_01150

these aspects of the experiment are just a subset of the potential features driving the activity of the neurons. This process of identifying the relevant features is known as model selection (for recent examples, see Hardcastle, Maheswaranathan, Ganguli, & Giocomo, 2017; Mimica, Dunn, Tombaz, Bojja, & Whitlock, 2018). As it is typically performed, model selection indicates the most likely feature or combination of features behind the activity of the neurons. It does not, however, indicate a feature that is not explicitly included. We would therefore prefer to skip the problem of devising and testing all of the infinite ways of tracking and processing the behavioral data and instead allow the neural data to speak for themselves in data-driven model selection.

Data-driven approaches such as ours have been made possible by recent advances in recording technology that permit simultaneous recordings of a large number of neurons. While the possible states of these neurons form a very high-dimensional space, one expects that the neural activity can be described by a smaller set of parameters (Tsodyks, Kenet, Grinvald, & Arieli, 1999). Standard dimensionality reduction techniques such as principal component analysis (PCA) and factor analysis (see Cunningham & Yu, 2014, for a comprehensive review) can be used to obtain a low-dimensional version of the data. However, most of these methods give us data lying in some Euclidean space, while often a different space would be more appropriate. For instance, if the neurons encode head direction, the population activity should be confined to points on a circle, corresponding to specific directions in the room. If the neurons are tuned to more complex features, or perhaps a combination of features, then spaces with richer topology would be appropriate. Since these spaces often have nontrivial topology, we argue that topological methods should be used in addition to traditional methods. Here we use persistent cohomology and circular parameterization (de Silva, Morozov, & Vejdemo-Johansson, 2011; Vejdemo-Johansson, Pokorny, Skraba, & Kragic, 2015) to identify the shape of the underlying space and then decode the time-varying position of the neural activity on it.

In summary, our method consists of combining four key steps:

1. Dimensional reduction using PCA
2. Feature identification using persistent cohomology
3. Decoding using circular parameterization
4. Removing the contribution of the decoded features on the data using a generalized linear model (GLM)

This procedure is quite general and may qualify to be called *cohomological feature extraction*. Steps 1 to 3 are illustrated in Figure 1. In step 4, the process of removing the contribution of a given feature results in a new data set to which the method can then be reapplied, in a way similar to Spreemann, Dunn, Botnan, and Baas (2015), in order to characterize additional features in the neural activity.

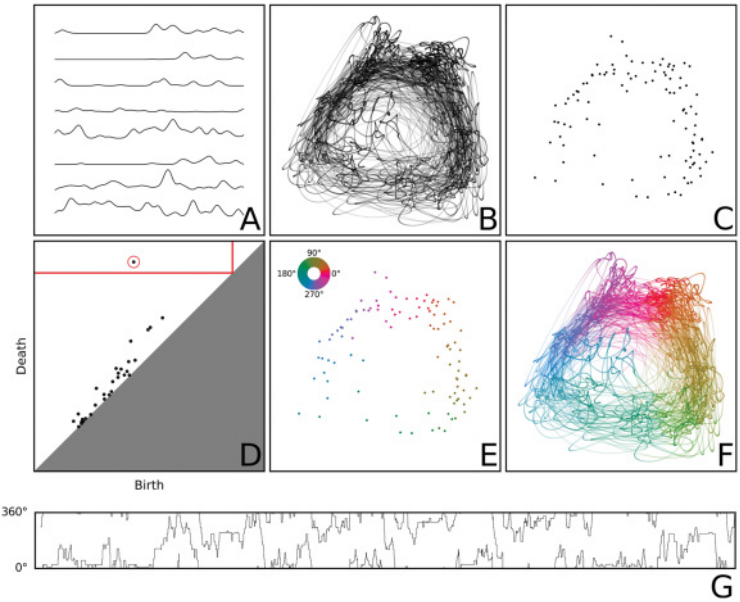


Figure 1: The decoding procedure. We start with an estimated firing rate for each neuron (A) obtained by smoothing the spike trains of the neurons. The firing rates are sampled at fixed time intervals, giving a point cloud (B), which is then simplified (see appendix C), obtaining a reduced point cloud (C). The first persistence diagram (D) of the reduced point cloud is computed. We select the longest living feature (circled in red) and a scale where this feature exists (red lines). Circular parameterization is used to obtain a circular map (E) on the reduced point cloud, where the color of a point represents its circular value according to the color wheel in the upper left corner (the same coloring scheme will be used throughout the rest of this letter). The map is extended to the full point cloud (F) by giving each point the value of its closest point in the reduced point cloud. (G) The decoded circular value as a function of time for the first 20 minutes of the recording, where the circular value at a given time point is the value of the corresponding point in panel F. The point clouds in panels B, C, E, and F are displayed as 2D projections.

As an example, we apply this method to neural data recorded from freely behaving mice (Peyrache, Lacroix, Petersen, & Buzsáki, 2015; Peyrache & Buzsáki, 2015), discover a prominent circular feature, and decode the time-varying position on this circle. We demonstrate that this corresponds well with the head direction, but with subtle, interesting differences. Finally, we consider the remaining features, following the removal of the head direction component, and find that a structure still there is correlated to the speed of the mouse.

2 Method

2.1 Decoding Procedure. Here we provide details on the decoding procedure summarized in Figure 1. We are given a set $\mathcal{N} = \{1, 2, \ldots, N\}$ of neurons and a spike train $(s_1^i, s_2^i, \ldots, s_{n^i}^i)$ for each neuron $i \in \mathcal{N}$, consisting of the time points when the neuron fires.[1] Neurons with a mean firing rate lower than 0.05 spikes per second are discarded. The remaining spike trains are smoothed with a gaussian kernel of standard deviation $\sigma = 1000$ ms and then normalized to take values between 0 and 1. This gives us estimated normalized firing rates $f_i : \mathbb{R} \to [0, 1]$ for each neuron $i \in \mathcal{N}$, as shown in Figure 1A. The firing rates are then sampled at fixed time intervals of length $\delta = 25.6$ ms,[2] resulting in a sequence $(\bar{\mathbf{x}}_0, \bar{\mathbf{x}}_1, \ldots)$ of points in $\mathbb{R}^N$, where $\bar{\mathbf{x}}_t = (f_1(\delta t), f_2(\delta t), \ldots, f_N(\delta t))$. In order to reduce noise, we project this point cloud onto its first $d = 6$ principal components,[3] resulting in a sequence $\mathcal{X} = (\mathbf{x}_0, \mathbf{x}_1, \ldots)$ of points in $\mathbb{R}^d$ as shown in Figure 1B (in 2D). We then simplify the point cloud $\mathcal{X}$, obtaining the point cloud $\widetilde{\mathcal{X}}$, shown in Figure 1C (see appendix C).

The first persistent cohomology of $\widetilde{\mathcal{X}}$ is then computed (see appendix A), giving us a summary of the one-dimensional holes in the data. We are interested in the longest living holes because they represent stable features. In our analysis, we focus on the longest living hole with life span $[a, b)$, and we want to understand how this hole is being reflected in the data. To do this, we first pick a scale $\epsilon \in [a, b)$ where this feature exists. We used $\epsilon = a + 0.9(b - a)$ in our analysis.[4] Figure 1D shows the persistence diagram. The chosen feature $[a, b)$ is circled in red, and the scale ϵ is marked with red lines. We then construct a continuous map from the Rips complex (see appendix A) of $\widetilde{\mathcal{X}}$ at scale ϵ to the circle S^1 in such a way that the selected hole gets sent to the hole inside the circle. By restricting this map to the points of $\widetilde{\mathcal{X}}$, we obtain a circular value—an angle $\theta(x)$ for each point $x \in \widetilde{\mathcal{X}}$, as shown in Figure 1E. Intuitively, the angle of a point indicates where the point lies relative to the selected hole. The process of constructing a circular map on $\widetilde{\mathcal{X}}$ given a 1-dimensional hole is called *circular parameterization* (see appendix D) and uses the representation of the 1-dimensional cohomology group of a space as maps from the space to S^1. We use an improvement on the original procedure by de Silva et al. (2011), which gives better results

[1] The spike trains that we analyzed had a duration of about 30 to 40 minutes.

[2] Different interval lengths are possible, but we chose the same interval lengths as were used for the camera tracking for practical purposes.

[3] Care should be taken here, as is also noted in section 5.1 of Vejdemo-Johansson et al. (2015), since if the projecting dimension d is too small, the projection could give rise to intersections of the underlying space, giving topological artifacts. It lies in our assumptions that the underlying shape is not too complex and is still an embedding when projected to six principal components.

[4] Our experience, which is shared with de Silva et al. (2011), is that picking a scale near the end of the life span of the selected feature resulted in a better circular parameterization.

in cases when the data are nonuniformly distributed around the hole. (This improvement is described in appendix D in section D.4). We then extend the function θ to $\mathcal{X}$ as shown in Figure 1F by giving a point $\mathbf{x}_t \in \mathcal{X}$ the circular value of its closest point in $\widetilde{\mathcal{X}}$. In the end, we get a circular value $\theta(\mathbf{x}_t)$, for each time point δt—a time-dependent circular value as shown in Figure 1G.

To improve the decoding, we first run the procedure described above using a large smoothing width of $\sigma = 1000$ ms such that only features with slow dynamics remain, while arguably less interesting features such as local theta phase preferences are ignored. Having decoded the feature, we use an information-theoretic measure to determine which cells in the population are selective for this feature (Skaggs, McNaughton, & Gothard, 1993). We then run the procedure again with only the spike trains of the selective neurons, this time using a smaller smoothing width of $\sigma = 250$ ms to allow for a finer decoding.

2.2 Residual Analysis. Identification of a single feature from a neural recording is a first step. A single recording, however, could contain a mixture of cells responding to different features or even multiple features (Rigotti et al., 2013). We therefore take an iterative approach, identifying features using topological methods and then explaining away by a statistical model to reveal any additional features, similar to Spreemann et al. (2015). We use a generalized linear model (GLM) (McCullagh & Nelder, 1989) to predict the neural activity given the decoded angle. We next subtract the predicted neural activity from the original spike trains, obtaining residual spike trains. We can then apply our decoding procedure to the residual spike trains to uncover remaining features in the data.

3 Results

3.1 Rediscovering Head Direction Cells. We applied our decoding procedure, as summarized in Figure 1, on spike train data from multielectrode array recordings of neurons in the anterodorsal thalamic nucleus (ADn) and the postsubiculum (PoS) of seven freely moving mice (Peyrache & Buzsáki, 2015; Peyrache et al., 2015). This revealed a prominent circular feature that was then decoded as shown in Figures 2 and 3, resulting in a one-dimensional circular time-dependent value. This decoded trajectory corresponded very well to the tracked head direction, as shown in Figure 3 and video 1 (see the online supplement).

In Figures 4B to 4D, we see that the decoded trajectories convey more information about the neural activity than the tracked head direction does. We were able to resolve moments of drift during the experiment—moments when the neural data are better explained by the decoded angle than by the tracked head direction. This was done by evaluating the time-varying log-likelihood ratio of two GLMs—one with the decoded angle and the other with the tracked head direction. The difference of these two log likelihoods

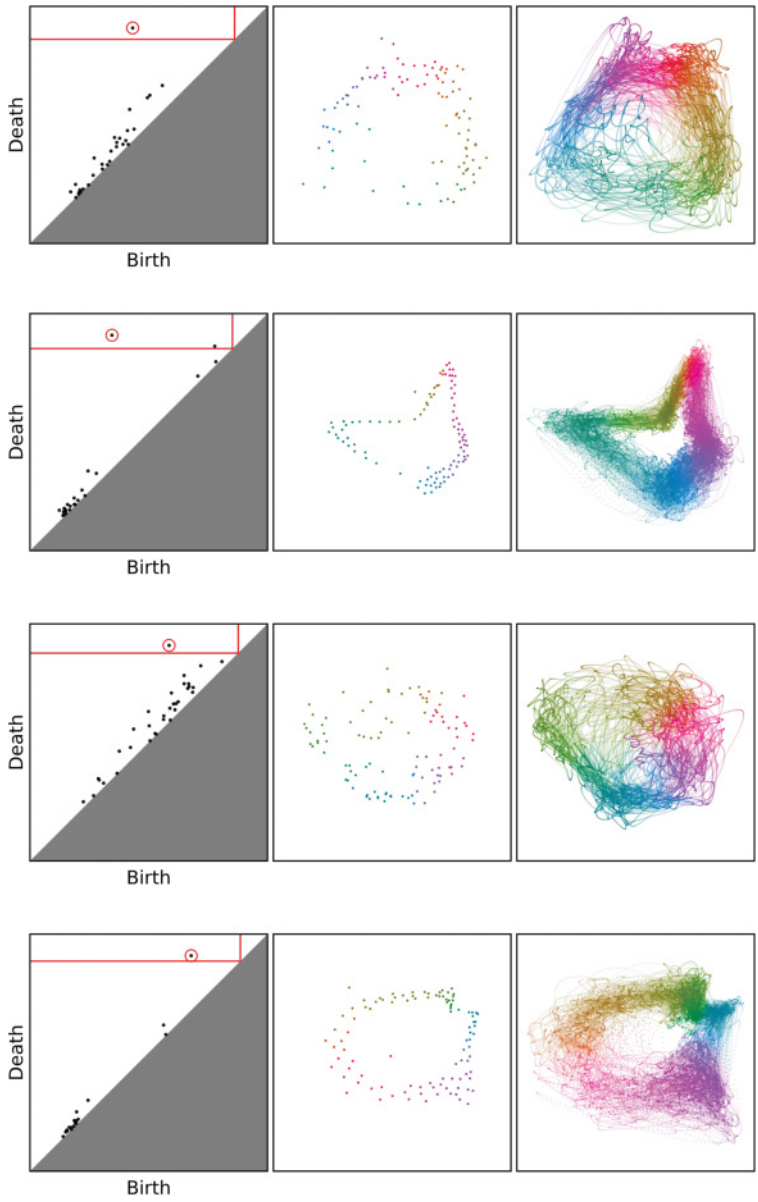


Figure 2: The decoding procedure applied to two recordings. From top to bottom: mouse28-140313 (using all neurons and only identified selective neurons), mouse25-140130 (using all neurons and only identified selective neurons). For each round, we show the persistence diagram, the decoded circular value on the reduced point cloud, and the circular value on the full point cloud. The point clouds in the second and third column are displayed as 2D projections.

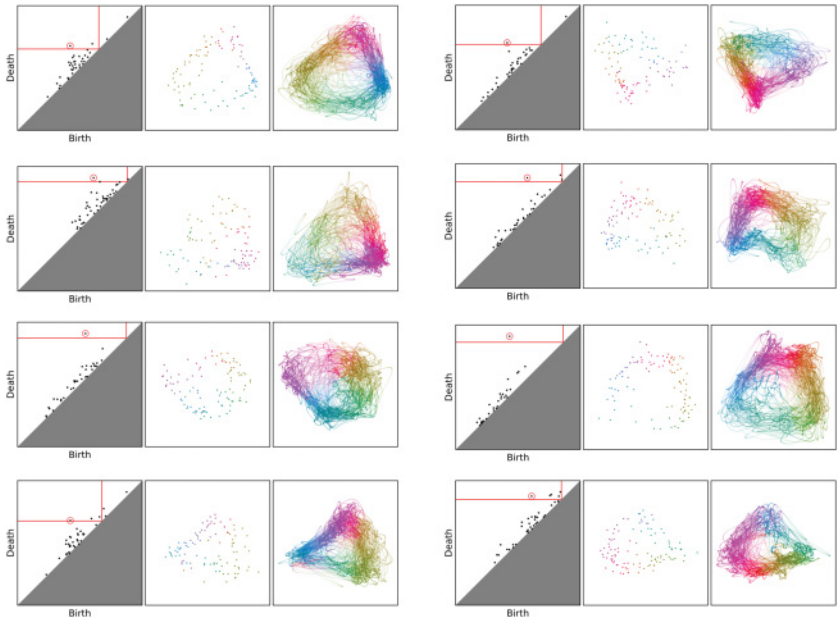


Figure 3: The decoding procedure applied to different recordings, using all neurons.

is shown in Figure 4B for each neuron. The GLM test found 90 moments of drift, lasting 2.01 seconds on average, that were dispersed throughout the experiment. Consistent with our findings, drift of the head direction representation has been previously reported in rodents (Taube, Muller, & Ranck, 1990; Yoder, Peck, & Taube, 2015; Mizumori & Williams, 1993; Goodridge, Dudchenko, Worboys, Golob, & Taube, 1998).

Figure 4E shows that the discrepancy is mostly independent of the speed of the mouse, but with a slight clockwise deviation at slow speeds and a slight counterclockwise deviation at higher speeds. In Figure 4F, we observe what appears to be a spatial dependence of the deviation, where the decoded angle is skewed counterclockwise in some parts of the box and skewed clockwise in other parts. This suggests that the difference is not simply due to a random drift in the network representation (Zhang, 1996), but rather that the internal representation is occasionally distorted by the environment (Dabaghian, Brandt, & Frank, 2014; Knierim, Kudrimoti, & McNaughton, 1998; Peyrache, Schieferstein, & Buzsáki, 2017). Another possible reason might be that the head direction is not precisely what is driving the neural activity but rather something similar, such as gaze direction, the direction the body is facing, or the direction that the animal is attending to. It could also be partially due to tracking error since the animal was tracked

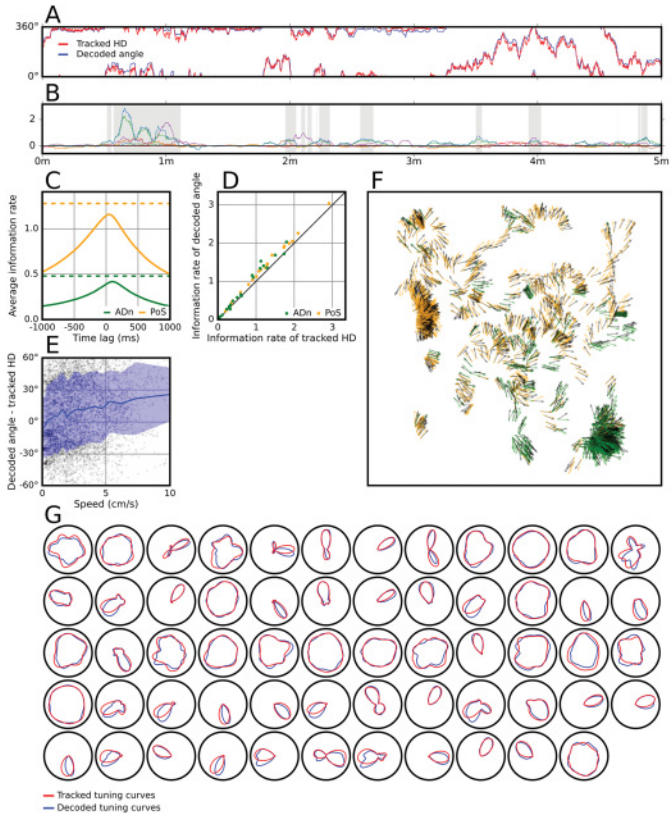


Figure 4: Result of the decoding procedure on mouse28-140313 (using only the identified selective neurons). (A) Tracked head direction (HD) and decoded angle for the first 5 minutes. (B) Difference between the log likelihood of observed spike trains given decoded angle and tracked HD over individual neurons. The shaded time intervals represent the moments of drift. (C) Average information rate over all ADn neurons (green lines) and PoS neurons (orange lines) of the tracked HD at different time lags (solid lines) and the decoded angle (dashed lines) at 0 time lag. The average information rate of the tracked HD peaked at 95 ms over ADn neurons and at 40 ms over PoS neurons. (D) Information rates of the tracked HD over all PoS neurons at 95 ms time lag and over all ADn neurons at 40 ms time lag, which is when their average information rates peaked, against information rates of the decoded angle at 0 time lag. (E) Angular difference (decoded angle − tracked HD) for each time step during drift, plotted against the speed of the mouse. We also display the average angular difference (blue line) ± 1 SD (shaded region). (F) For each time step during drift, we plot the tracked HD (black arrows) and the decoded angles, shown as a green (resp. orange) arrow when deviated clockwise (resp. counterclockwise) of the tracked HD. The arrows are rooted at the position of the mouse at that time step. (G) Tracked and decoded tuning curves of all neurons.

using a single camera at 30 frames per second and two LEDs on the animal's head.

3.2 Capturing Speed Cells. We then considered the residual activity remaining after accounting for the decoded angle in a GLM and applied the decoding procedure as before (shown in Figure 6). This time we did not find any additional cohomological features, but the eigenvalues of the covariance matrix of the residual point cloud suggest that a one-dimensional feature remains in the data, as shown in Figure 5A. Two possible candidates are mouse speed and angular velocity, and Figures 5B and 5C show that both candidates are correlated with the neural activity. By fitting a GLM including speed and a GLM including angular velocity to the data, we see in Figure 5D that most of the neurons are more selective for speed than for angular velocity. Finally, we included decoded HD, speed, and angular velocity in the GLM and created a new residual point cloud as before (shown in Figure 6C). This time the eigenvalues (see Figure 5E) are closer to random. We also tried to include only the residual spike trains of the neurons that had an initial mean firing rate higher than a certain threshold, varying the threshold between 0.05 and 20 spikes per second, but we were not able to uncover any structure in the remaining data (see Figures 7 and 8).

4 Discussion

As illustrated in our example, the main benefit of using persistent cohomology is that it allows us to understand the shape of the neural data. Statistical methods such as latent variable methods (Zhao & Park, 2017; Archer, Koster, Pillow, & Macke, 2014; Frigola, Chen, & Rasmussen, 2014; Koyama, Pérez-Bolde, Shalizi, & Kass, 2010; Macke et al., 2011; Pfau, Pnevmatikakis, & Paninski, 2013; Yu et al., 2008) can give a good low-dimensional representation of neural activity, but they have not yet been developed to characterize the shapes underlying the data. It is often difficult, or even impossible, to identify the shape by looking at a 2D or 3D projection, as seen, for instance, in the first round on mouse25-140130 (see Figure 2), and it would likely be even more difficult for more elaborate features. As a preprocessing step, however, latent variable methods are far more flexible and powerful than PCA, allowing, for example, for known covariates and priors to be included in a simple and straightforward manner. Here we chose PCA for simplicity but see a bright future combining the benefits of cohomological feature extraction and the framework of latent variable models.

We emphasize that our method does not rely on knowing a priori what we are looking for, such as head direction. In our analysis, the tracked head direction is only used in the end to demonstrate that the method gives us the expected feature for these data. In practice, the discovery of a circular feature, together with a circular parameterization, would give the researcher useful knowledge of what kinds of data are being encoded. For instance, if

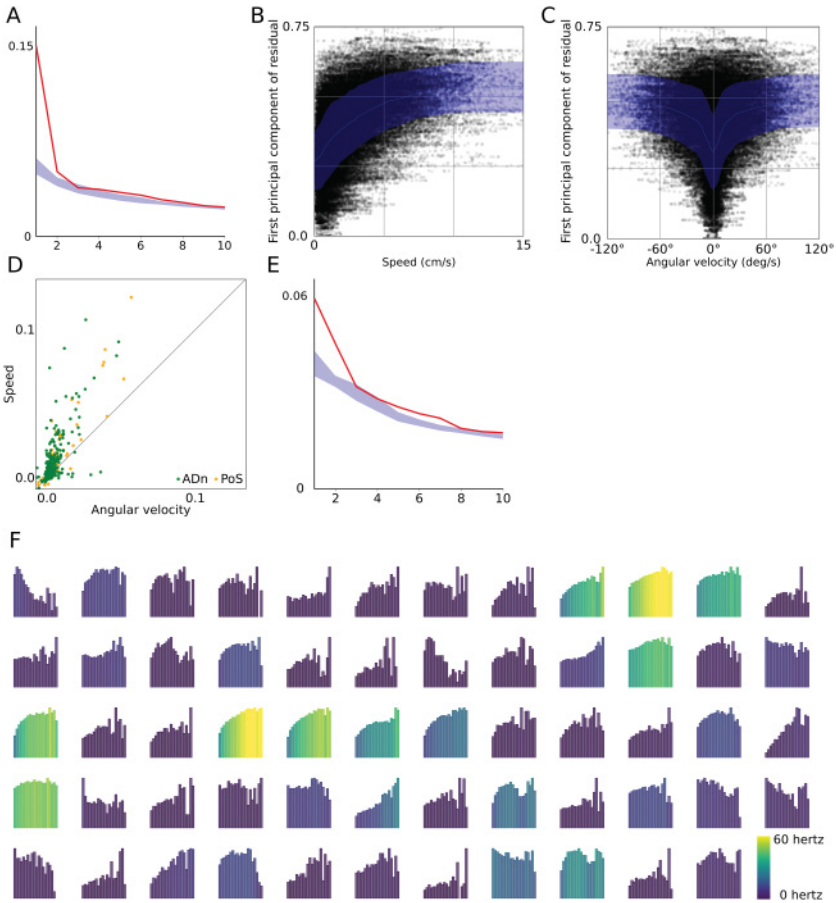


Figure 5: Residual analysis. (A) The 10 largest eigenvalues (red line) of the residual point cloud after accounting for decoded HD, compared with a 96% confidence interval of the 10 largest eigenvalues of each of 100 shuffled point clouds (see appendix B). (B,C) The first principal component of the residual point cloud against the speed (resp. angular velocity) of the mouse, with average values (blue line) ± 1 SD (shaded region). (D) The pseudo-R^2 score of a GLM including speed (x-axis) and a GLM including angular velocity (y-axis), averaged over a five-fold cross-validation on all recorded neurons in the data set. (E) The same as panel A but for the residual point cloud after accounting for decoded HD, speed, and angular velocity. (F) The speed tuning curves for all the neurons in mouse28-140313. On the x-axis of each tuning curve is the speed of the mouse, going from 0 cm/s to 15 cm/s. On the y-axis is the average firing rate of the neuron given the speed of the mouse. Note that the y-axes are scaled independently for each tuning curve.

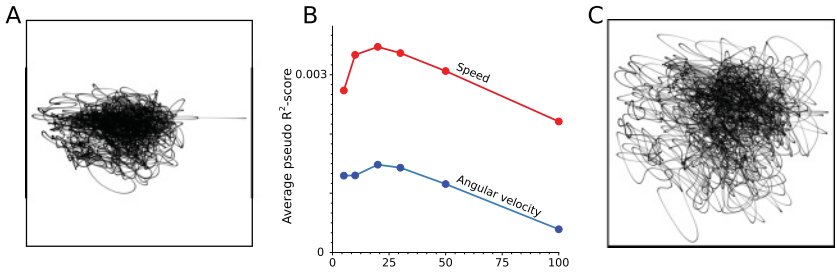


Figure 6: Supplement to residual analysis in Figure 5. (A) Residual point cloud after accounting for the decoded angle. (B) The pseudo-R^2 score of a GLM including speed (red line) and a GLM including angular velocity (blue line) for different number of bins, averaged over a five-fold cross-validation on all recorded neurons in the data set. (C) Residual point cloud after accounting for decoded angle, speed and angular velocity.

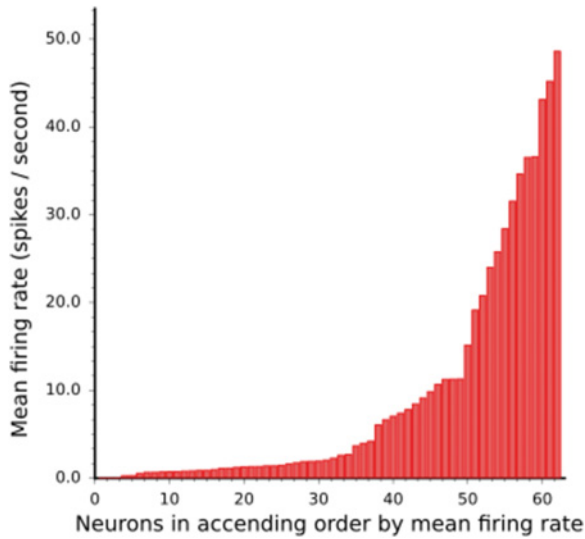


Figure 7: The mean firing rates of all 62 neurons of mouse28-140313.

the circular value were instead moving in one direction on the circle with a certain frequency, this would suggest that something periodic is being encoded.

Topological data analysis has been previously applied to identify the shape of the underlying space in neural data (Singh et al., 2008; Dabaghian, Mémoli, Frank, & Carlsson, 2012; Curto & Itskov, 2008). This work, however, is the first to develop and apply topological methods for decoding

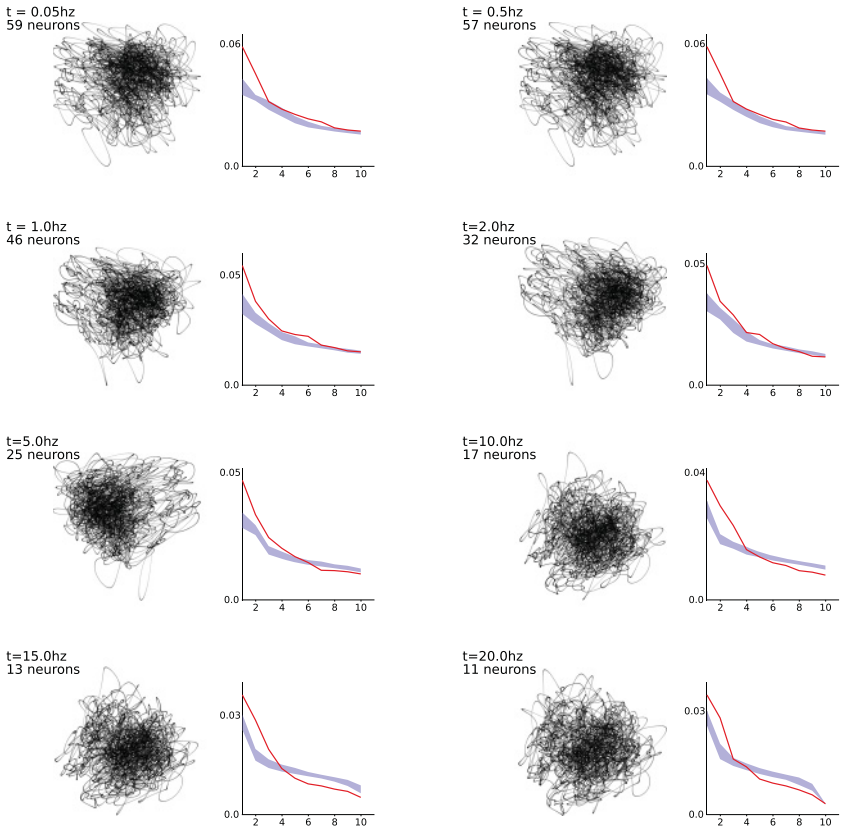


Figure 8: We show two figures for each threshold. The first, showing the resulting point cloud (in 2D) after removing the contribution of the decoded angle, speed, and angular velocity, and the second showing the 10 largest eigenvalues (red line) of the residual point cloud compared with a 96% confidence interval of the 10 largest eigenvalues of each of 100 shuffled point clouds. We also show the threshold t and the number of neurons included.

the time-dependent variable. It is also a first example of using topological methods for systematic identification of relevant parameters—model selection (see Spreemann et al., 2015, for a demonstration on simulated data).

Some applications of topological methods to neural data (Spreemann et al., 2015; Curto & Itskov, 2008; Giusti, Pastalkova, Curto, & Itskov, 2015; Dabaghian et al., 2012) have constructed spaces based on correlations between neuron pairs. This may work well when the tuning curves of the neurons are convex, but this assumption might be too strong. For instance, some of the neurons in Figure 4G are selective for two different directions.

This could be due to errors in preprocessing where the activity of two neurons is combined or might even be a property of the cells themselves. Our method does not require this assumption.

In our analysis, we focused on only the longest-living 1-dimensional hole, but in general, this should depend on the situation. If more than one hole is considered significant, it would make sense to apply circular parameterization with respect to each feature. For instance, if the underlying space was a torus, persistent cohomology would detect two 1-dimensional features corresponding to the two orthogonal ways to traverse a torus, as well as one 2-dimensional feature. The 1-dimensional features would then give circular parameterizations that together determine the time-dependent position on the torus. Additional theoretical work, such as that by Perea (2016), should reveal the extent to which other features can be identified and decoded.

Ideally, when performing persistent cohomology, the real features are well separated from noise in the persistence diagram. For instance, in the diagrams in Figure 2, there is a prominent 1-dimensional hole. In other cases, however—for instance, some of the diagrams in Figure 3—it is not easy to separate the features from noise. However, even in these cases, the circular parameterizations could still be meaningful, and our method of performing the decoding procedure, again with only the neurons selective for the chosen feature, might be able to remove some of the noise, making the interesting features more prominent.

Topological data analysis methods have developed into a powerful, intuitive set of tools, built with mathematical rigor, that can now be applied to large population neural recordings. Here we show one clear example of how these methods can recover the relevant features of the animal's behavior, essentially performing unsupervised decoding and model selection. With the continued emergence of large population neural recordings, we expect to see topological methods playing an important role in exploring what the neural data are trying to tell us.

Appendix A: Topological Background

The state of a neural population at a given time point can be represented as a high-dimensional vector $x \in \mathbb{R}^n$. When collecting all the vectors corresponding to each time point, we get what is called a point cloud $\mathcal{X} \subset \mathbb{R}^n$. Describing the shape of point clouds such as these is nontrivial but can be done using tools from the field of algebraic topology.

Cohomology (see chapter 3 of Hatcher, 2002), one such tool, is able to distinguish between shapes, such as a circle, sphere, or torus.[5] Intuitively, the 0-dimensional cohomology counts the number of connected components of

[5] Technically speaking, the homotopy types.

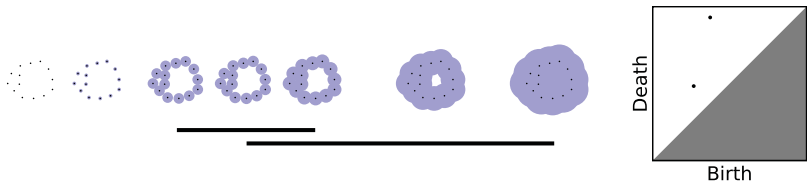


Figure 9: (Left) A filtration together with its first persistent cohomology, drawn as a bar code. Two 1-dimensional holes appear and disappear during the filtration, as reflected in the bars. (Right) First persistence diagram of the filtration.

the space, and for $n \geq 1$, the n-dimensional cohomology counts the number of n-dimensional holes in the space. For instance, a circle has one connected component and one 1-dimensional hole, a sphere has one connected component and one 2-dimensional hole, and a torus has one connected component, two 1-dimensional holes, and one 2-dimensional hole.

If we try to naively compute the cohomology of a finite point cloud $\mathcal{X} \subset \mathbb{R}^n$, we run into a problem, however, since this is just a finite set of points with no interesting cohomology. Instead we consider the cohomology of the ϵ-thickening of $\mathcal{X}$, denoted by $\mathcal{X}_\epsilon$, consisting of the points in $\mathbb{R}^n$ that are closer than ϵ to a point in $\mathcal{X}$.[6]

If we are able to find a suitable scale ϵ, we might recover the cohomology of the underlying shape. Often, however, there is noise in the data giving rise to holes, and there is no way to separate the noise from the real features. The solution is to consider all scales rather than fixing just one scale. This is where *persistent cohomology* (de Silva et al., 2011) enters the picture.[7] Persistent cohomology tracks the cohomology of the space $\mathcal{X}_\epsilon$ as ϵ grows from 0 to infinity. Such a sequence of growing spaces is called a *filtration*. When $\epsilon = 0$, the set $\mathcal{X}_\epsilon$ is equal to $\mathcal{X}$ and has no interesting cohomology. When ϵ is large enough, the space $\mathcal{X}_\epsilon$ becomes one big component with no holes. In between these two end points, however, holes may appear and disappear.

The nth persistent cohomology gives us the birth scale and death scale of all n-dimensional features. A feature with birth scale a and death scale b is denoted by $[a, b)$. The features can then be drawn as a bar code, or alternatively, each feature $[a, b)$ can be drawn as the point (a, b) in the plane, obtaining the nth persistence diagram, as shown in Figure 9. The features far away from the diagonal persist longer in the filtration and are considered more robust, while features near the diagonal are considered as noise.

[6] In practice, we replace the ϵ-thickenings by approximations called Rips complexes. See appendix D.

[7] In topological data analysis, persistent homology is often used instead, but we will need cohomology for the decoding step. We refer to Ghrist (2008); Edelsbrunner and Harer (2010) for introductions to persistence and Ghrist (2014); and Carlsson (2009) for introductions to applied topology in general.

Once the persistence diagram has been computed, we want to relate the discovered features to the point cloud. This can be done using circular parameterization (de Silva et al., 2011; Vejdemo-Johansson et al., 2015); see also appendix D), which turns any chosen 1-dimensional feature existing at a scale ϵ into a circle-valued function $f : \mathcal{X} \to S^1$, revealing the feature. (See Figure 1.)

Appendix B: Details about the Methods

B.1 Identifying Selective Neurons. We compute the information rate, as described in Skaggs et al. (1993), for each neuron relative to the circular value using the formula

$$I = \int_{S^1} \lambda(\theta) \log_2 \frac{\lambda(\theta)}{\lambda} p(\theta)\, d\theta,$$

where $\lambda(\theta)$ is the mean firing rate of the neuron when the circular value is θ, λ is the overall mean firing rate of the neuron, and $p(\theta)$ is the fraction of time points where the circular value is θ. The integral is estimated by partitioning the circle in 20 bins of width 18 degrees and assuming that the mean firing rate $\lambda(\theta)$ is constant in each bin. We identify a neuron as being selective if its information rate is larger than 0.2. The method can then be run again without the spike trains of the nonselective neurons. The information rates of each neuron relative to the tracked HD, shown in Figures 4C and 4D, were calculated in the same way.

B.2 Comparison with Tracked HD. For the comparisons with the tracked head direction in Figure 4, we rotated the decoded circular value to minimize the mean squared deviation from the tracked HD. The tuning curves in Figure 4G were made by taking the histogram of the angles at the spikes of each neuron, dividing by the time spent in each angle, and smoothing with a gaussian kernel of 10 degree standard deviation.

B.3 Calculating Mouse Speed and Angular Velocity. The speed of the mouse was calculated by first smoothing the tracked position of the mouse separately in the x-axis and y-axis with a gaussian kernel of 0.1 s, and then taking the central difference derivative of the smoothed position at each time step. Similarly, the angular velocity was calculated by first smoothing the tracked HD of the mouse with a gaussian kernel of 0.1 s and then taking the central difference derivative at each time step.

B.4 GLM. We fit a GLM to the activity of each neuron $i \in \mathcal{N}$ as follows. First, we binned the spike train of the neuron into bins of size 25.6 ms and the circle into angular bins of width 36 degrees. Let $X_j(t)$ be the indicator function, which is 1 if the decoded value is in the jth angular bin at time

step t and 0 otherwise, and let y_t be the observed spike count of the neuron in time bin t. Assuming that the firing of the neuron is an inhomogeneous Poisson process with instantaneous intensity at time step t given by

$$H(t) = \sum_{j=0}^{9} \beta_j X_j(t) + h,$$

where h and the β_j are parameters, the log likelihood of observing the spike train of neuron i is given by

$$L_i^{decoded} = \sum_t -H(t) + y_t \log(H(t)) - \log(y_t!).$$

We used Ramkumar (2016) to obtain the parameters $(h, \{\beta_j\})$ that maximize the log likelihood. The log likelihood of the neuron having y_t spikes in bin t is now given by

$$L_i^{decoded}(t) = -H(t) + y_t \log(H(t)) - \log(y_t!).$$

We then repeated this process, replacing the function $X_j(t)$ by the indicator function corresponding to the tracked HD and obtained the corresponding values $L_i^{tracked}(t)$. The difference $L_i^{decoded}(t) - L_i^{tracked}(t)$ represents how much better the neuronal activity at time step t is explained by the decoded value compared to the tracked HD. This difference is shown for each neuron in Figure 4B. The time intervals when the sum of these differences was above a certain threshold (we used $thresh = 1$) is referred to as *moments of drift*.

B.5 Residual Analysis. Given the above GLM, we made a residual spike train for each neuron $i \in \mathcal{N}$ by taking the original spike train of the neuron and adding for each time step t a negative spike of magnitude $H(t)$ at the center of bin t. This is similar to the residual spike trains described in Spreemann et al. (2015).

Figure 5A was made as follows. The red line shows the 10 largest eigenvalues of the covariance matrix of the residual point cloud after accounting for the decoded angle in the GLM. Each of the residual spike trains that gave rise to this point cloud was then shifted in time by n time steps, where n is a number sampled uniformly in the range from 0 to 100. This was done 100 times, resulting in 100 point clouds. For each of these point clouds, the 10 largest eigenvalues of their covariance matrix were chosen. The blue band in Figure 5B shows for each index i a 96% confidence interval of the ith largest eigenvalue, sampled from this empirical distribution.

Figures 5D and 6B were made as follows. We binned the mouse speed in $n-1$ equally sized bins in the range $[0, 15)$ and one bin $[15, \infty)$, and the angular velocity in $n-2$ equally sized bins in the range $[-\frac{\pi}{2}, \frac{\pi}{2})$ and the

two bins $(-\infty, \frac{\pi}{2})$ and $[\frac{\pi}{2}, \infty)$. We then defined two GLMs as before, letting the functions $X_j(t)$ be the indicator functions on the speed bins (resp. the angular velocity bins). These two models were cross-validated using five folds on each recorded neuron of every mouse in the data set for different choices of the number of bins n. The average pseudo-R^2 score over all folds and over all neurons is shown in Figure 6B for the two models and for different values of n. The average pseudo-R^2 score over all folds for each indivual neuron is shown in Figure 5D for the two models, using 20 bins in each model, which is where the average pseudo-R^2 scores peaked.

Figure 5E was made the same way as Figure 5A, but for the residual point cloud after accounting for decoded angle, speed, and angular velocity in the GLM.

Appendix C: Point Cloud Simplification

The procedure of simplifying the point cloud $\mathcal{X}$ to obtain the point cloud $\widetilde{\mathcal{X}}$ consists of two steps: radial distance and topological denoising.

C.1 Radial Distance. To make the computations faster, we simplify $\mathcal{X}$ using the radial distance method (de Koning, 2011). Start with the first point in $\mathcal{X}$ and mark it as a key point. All consecutive points that have a distance less than a predetermined distance ϵ to the key point are removed. The first point that has a distance greater than ϵ to the key point is marked as the new key point. The process repeates itself from this new key point and continues until it reaches the end of the point cloud. This procedure results in a smaller point cloud $\mathcal{X}'$ that is close to the original point cloud.[8] The parameter ϵ is typically determined proportionally to the spread of the point cloud. In our analysis, we used $\epsilon = 0.02$.

C.2 Topological Denoising. Since the point cloud is noisy, we need to reduce the amount of noise before we can look for topological features. We use a method for topological denoising that was introduced in Kloke and Carlsson (2010). Given a subset S of $\mathcal{X}'$, we define the function

$$F(S, x) = \frac{1}{|\mathcal{X}'|} \sum_{p \in \mathcal{X}'} e^{\frac{-||x-p||^2}{2\sigma^2}} - \frac{\omega}{|S|} \sum_{p \in S} e^{\frac{-||x-p||^2}{2\sigma^2}}.$$

The parameter σ is an estimate on the standard deviation of the noise, and the parameter w determines how much the points in S should repel each other. We maximize the function F by iteratively moving each point in S in the direction of the gradient of F. Starting with a sample $S_0 \subset \mathcal{X}'$, we let

[8] Since each point in $\mathcal{X}'$ has a distance less than ϵ to a point in X, the Hausdorff distance between the two point clouds is less than ϵ.

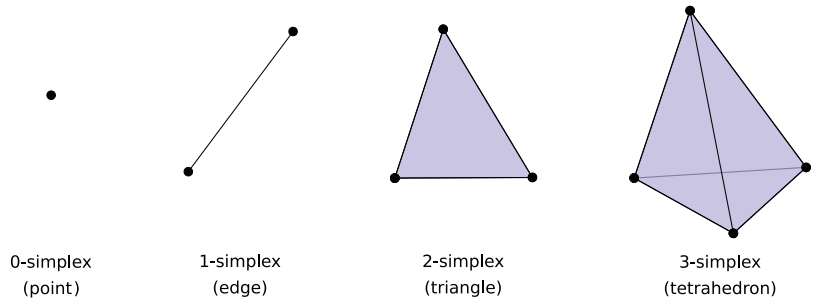


Figure 10: Simplices.

$$S_{n+1} = \left\{ p + c\frac{\nabla F(S_n, p)}{M} \right\}$$

for all n, where

$$M = \max_{p \in S_0} ||\nabla F(S_0, p)||$$

and c is a parameter determining the maximum distance the points in S_n can move. In our case, we constructed S_0 by taking every kth point of $\mathcal{X}'$, where k is the largest number such that S_0 had 100 points. We used the parameters $\sigma = 0.1$ s, $\omega = 0.1$, and $c = 0.05$, where s is the standard deviation of $\mathcal{X}'$.[9] We did 60 iterations resulting in the point cloud $\widetilde{\mathcal{X}} = S_{60}$.

Appendix D: Circular Parameterization

The material in this appendix is mostly a reformulation of parts of de Silva et al. (2011) with the exception of section D.4, our own contribution.

D.1 Simplicial Complexes. In topological data analysis, shapes are modeled by simplicial complexes. A simplicial complex can be thought of as a space that is constructed by gluing together basic building blocks called simplices (see Figure 10).

Definition 1. *A simplicial complex is a pair* (X, S)*, where* X *is a finite set called the underlying set and* S *is a family of subsets of* X*, called simplices, such that*

1. *For every simplex* $\sigma \in S$*, all of its subsets* $\sigma' \subseteq \sigma$ *are also simplices.*
2. *For every element* $x \in X$*, the one element set* $\{x\}$ *is a simplex.*

[9] The parameters used in our analysis were chosen experimentally, but automatical methods could be used instead.

An *n-simplex* is a simplex consisting of $n+1$ elements. The 0-simplices, 1-simplices, and 2-simplices are called, respectively, points, edges, and triangles. The simplicial complexes used in our method are called *Rips complexes*, defined as follows:

Definition 2. *Let $\mathcal{X} \subset \mathbb{R}^n$ be a point cloud. The Rips complex of $\mathcal{X}$ at scale ϵ, denoted $R_\epsilon(\mathcal{X})$, is the simplicial complex defined as follows:*

- *The underlying set is $\mathcal{X}$.*
- *A subset $\sigma \subset \mathcal{X}$ is a simplex iff $||x - y|| \leq \epsilon$ for all $x, y \in \sigma$.*

D.2 Cohomology. Let X be a simplicial complex, and let X_0, X_1, and X_2, be respectively, the points, edges, and triangles of X.

We will assume a total ordering on the points in X. If $\{a, b\}$ is an edge with $a < b$, we will write it as ab. Similarly, if $\{a, b, c\}$ is a triangle with $a < b < c$, we will write it as abc.

Given a commutative ring $\mathbb{A}$, for instance, $\mathbb{Z}$, $\mathbb{R}$, or $\mathbb{F}_p$, we define 0-chains, 1-chains, and 2-chains with coefficients in $\mathbb{A}$ as follows:

$$C^0 = C^0(X; \mathbb{A}) := \{f : X_0 \to \mathbb{A}\},$$
$$C^1 = C^1(X; \mathbb{A}) := \{\alpha : X_1 \to \mathbb{A}\},$$
$$C^2 = C^2(X; \mathbb{A}) := \{A : X_2 \to \mathbb{A}\}.$$

These are modules over $\mathbb{A}$. We define the coboundary maps $d_0 : C^0 \to C^1$ and $d_1 : C^1 \to C^2$ as follows:

$$d_0(f)(ab) = f(b) - f(a),$$
$$d_1(\alpha)(abc) = \alpha(ab) + \alpha(bc) - \alpha(ac).$$

A 1-chain α is called a *cocycle* if $d_1(\alpha) = 0$, and it is called a *coboundary* if there exists a 0-chain $f \in C^0$ such that $\alpha = d_0(f)$. It is easy to show that all coboundaries are cocycles. We define the first cohomology of X with coefficients in $\mathbb{A}$ to be the module

$$H^1(X; \mathbb{A}) = \frac{\mathrm{Ker}(d_1)}{\mathrm{Im}(d_1)}.$$

Two 1-chains α and β are said to be cohomologous, or belonging to the same cohomology class if $\alpha - \beta$ is a coboundary.

D.3 Circular Parameterization. The idea behind circular parameterization comes from the following theorem:

Theorem 1. *There is an isomorphism*

$$H^1(X, \mathbb{Z}) \simeq [X, S^1]$$

between the 1-dimensional cohomology classes with integer coefficients of a space X and the set of homotopy classes of maps from X to the circle $S^1 = \mathbb{R}/\mathbb{Z}$. (This is a special case of theorem 4.57 in Hatcher, 2002.)

Given a representative cocycle $\alpha \in C^1(X; \mathbb{Z})$, the associated map $\theta : X \to \mathbb{R}/\mathbb{Z}$ is given by sending all the points in X to 0, sending each edge ab around the circle with winding number $\alpha(ab)$ and extending linearly to the rest of X. This extension is well defined since $d_1(\alpha) = 0$.

The circular maps obtained this way are not very smooth, since all points in X are sent to the same point on the circle. In order to allow for smoother maps, we consider cohomology with real coefficients. Consider α as a real cocycle, and suppose we have another cocycle $\beta \in C^1(X; \mathbb{R})$ cohomologous to α. Since it is cohomologous, it can be written as $\beta = \alpha + d_0(f)$ for some $f \in C^0(X; \mathbb{R})$. We define the map $\theta : X \to \mathbb{R}/\mathbb{Z}$ by sending a point a to $\theta(a) = f(a) (\mathrm{mod}\, \mathbb{Z})$ and an edge ab to the interval of length $\beta(ab)$ starting at $\theta(a)$ and ending at $\theta(b)$. This map is extended linearly as before to the higher simplices of X.

Given a cocycle $\alpha \in C^0(X; \mathbb{Z})$, we now want to find a real 0-chain $f \in C^0(X; \mathbb{R})$ such that the cocycle $\beta = d_0(f) + \alpha$ is smooth, meaning that the edge lengths $||\beta(ab)||$ are small. Define

$$||\beta||^2 = \sum_{ab \in X_1} ||\beta(ab)||^2.$$

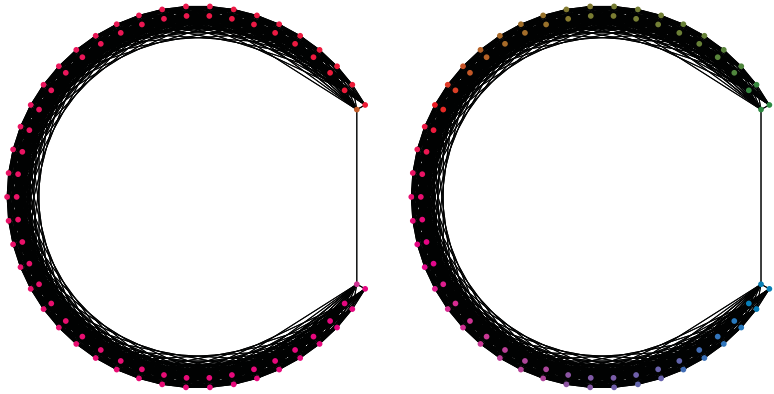
We want to minimize this value. The desired 0-chain can be expressed as

$$f = \underset{\bar{f}}{\operatorname{argmin}} ||d_0(\bar{f}) + \alpha||. \tag{D.1}$$

This is a least squares problem and can be solved using, for instance, Jones, Oliphant, Peterson et al. (2000).

D.4 Improved Smoothing. When we tried to apply circular parameterization to the data sets, we quickly discovered that it often produces unsatisfactory results. We demonstrate this using a constructed example, the Rips complex shown in Figure 11a. The first cohomology of this complex with integer coefficient is $\mathbb{Z}$, generated by the cocycle, which has the value one on the right-most edge and zero on all the other edges. The color of a point indicates its circular value after applying circular parameterization.

We see that the original smoothing sends the points to circular values that are close to each other. The reason for this behavior is that the edges in the simplicial complex are not evenly distributed around the circle; there is only one edge covering the right-most part of the circle. Since this edge gives a relatively small contribution to the sum of squares of edge lengths, it will be stretched around the circle so that the other edges can get smaller.



(a) Original smoothing (b) Improved smoothing

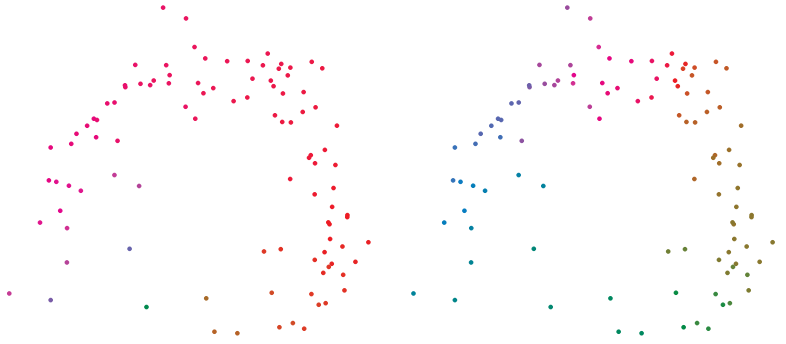
Figure 11: Comparison between original and improved smoothing for the constructed example.

This is problematic because we want the geometry of the complex to be preserved.

We describe a heuristical approach to improve the situation. First, we assume that the underlying set of the simplicial complex comes with a metric $d : X \times X \to \mathbb{R}$. When the simplicial complex is a Rips complex of a point cloud $\mathcal{X} \in \mathbb{R}^n$, we take the metric to be the Euclidean metric. Let $\alpha \in C^1(X, \mathbb{Z})$ be the cocycle that we want to use for circular parameterization. Now, instead of treating every edge equally, we assign a positive weight $w : X_1 \to (0, \infty)$ to each edge in the simplicial complex. We define the weighting by the following procedure. First, we solve the original optimization problem 1.1 to obtain a cocycle $\beta = d_0(f) + \alpha$. We then construct a weighted directed graph G as follows. The vertices of G are the points $a \in X^0$. For each edge $ab \in X^1$ with $\beta(ab) \geq 0$, there is an edge in G from a to b denoted ab, and for each edge $ab \in X^1$ with $\beta(ab) < 0$, there is an edge in G from b to a denoted ba. This gives us a bijection between edges in X and edges in G. The weight of an edge ab is given by $d(a, b)$. It can be shown that every edge ab in G has at least one directed cycle going through it. Now, for every edge in G, take the shortest directed cycle in the graph going through it. This gives us a collection of cycles in G. We now define

$$w(ab) = \frac{l(ab)}{d(a, b)^2},$$

where $l(ab)$ is the number of cycles going through the corresponding edge in G.



(a) Original smoothing (b) Improved smoothing

Figure 12: Comparison between original and improved smoothing for real data.

Given this weighting, we then define

$$||\beta||_w^2 = \sum_{ab \in X_1} w(ab)\, ||\beta(ab)||^2.$$

Our new optimization problem then becomes

$$f = \text{argmin}_{\bar{f}} ||d_0(\bar{f}) + \alpha||_w,$$

which is again a least squares problem. After solving this system, we obtain a new circular parameterization. The result of applying this procedure to the Rips complex in our example is shown in Figure 11b. Here the rightmost edge is given a much larger weighting than the other edges, since every directed cycle has to go through this edge. The result is that the circular values of the points are more evenly distributed around the circle. In Figure 12, we compare the original and improved smoothing on the circular parameterization done in the first round of mouse28-140313.

Appendix E: Persistent Cohomology

By varying the parameter ϵ of the Rips complex of a point cloud X, we get a parameterized family of simplicial complexes $R_\epsilon(\mathcal{X})$, where ϵ goes from 0 to infinity. For every pair of parameters $0 \leq a \leq b$, we have a natural inclusion map $i_a^b : R_a(\mathcal{X}) \to R_b(\mathcal{X})$. We have the following two properties:

1. The map i_a^a is the identity map on $R_a(\mathcal{X})$ for all $a \in \mathbb{R}$.
2. For all parameters $a \leq b \leq c$, we have $i_b^c \circ i_a^b = i_a^c$.

Such a sequence of simplicial complexes, together with maps satisfying the above two conditions, is called a *filtration*. Let p be a prime, and let $\mathbb{F}_p$ be the field of order p. When taking the cohomology of each space $R_\epsilon(\mathcal{X})$ with coefficients in $\mathbb{F}_p$, we get a parameterized family of vector spaces $H^n(R_\epsilon(\mathcal{X}))$ together with the induced linear maps $\hat{i}_a^b = H^n(i_a^b) : H^n(R_b(\mathcal{X})) \to H^n(R_a(\mathcal{X}))$ for every pair of parameters $0 \leq a \leq b$. We have the following two properties:

1. The map $\hat{i}_a^a$ is the identity map on $H^n(R_a(\mathcal{X}))$ for all $a \in \mathbb{R}$.
2. For all parameters $a \leq b \leq c$ we have $\hat{i}_a^b \circ \hat{i}_b^c = \hat{i}_a^c$.

Such a sequence of vector spaces, together with linear maps satisfying the above two conditions, is called a *persistence module*. The persistence module obtained by taking the cohomology of a filtration is called the *persistent cohomology* of the filtration. Another kind of persistence module is an *interval module*.

Definition 3. *Let J be an interval on the real line. A J-interval module, written $\mathbb{I}^J$, is the persistence module with vector spaces*

$$\mathbb{I}_a = \begin{cases} \mathbb{F} & \text{if } a \in J \\ 0 & \text{otherwise} \end{cases}$$

and linear maps

$$f_a^b = \begin{cases} I & \text{if } a, b \in J \\ 0 & \text{otherwise} \end{cases},$$

where I denotes the identity map on $\mathbb{F}$.

Theorem 2. *Given a finite point cloud $\mathcal{X}$, the nth persistent cohomology of the Rips complex of $\mathcal{X}$ can be decomposed into a sum of interval modules*

$$H^n(R(\mathcal{X})) \simeq \mathbb{I}^{[a_1,b_1)} \oplus \mathbb{I}^{[a_2,b_2)} \oplus \ldots \oplus \mathbb{I}^{[a_k,b_k)} = \mathbb{I}.$$

(This is a special case of theorem 2.7 in Chazal, Outdot, Glisse, and de Silva, 2016.)

The persistent cohomology algorithm, described in de Silva et al. (2011), finds the intervals in the decomposition, along with a representative cocycle $\alpha_{i,\epsilon} \in C^1(R_\epsilon(\mathcal{X}))$ for every interval $[a_i, b_i)$ and $a_i \leq \epsilon < b_i$, such that

$$\varphi_\epsilon([\alpha_{i,\epsilon}]) = (0, \ldots, 0, 1, 0, \ldots, 0),$$

where the 1 appears in the ith position.

By drawing each interval $[a, b)$ as a point (a, b) in the plane, we obtain the nth persistence diagram of the filtration. We may now apply

circular parameterization as follows. Pick an interval $[a_i, b_i)$ in the decomposition. Then choose a scale $a_i \leq \epsilon < b_i$, and take the corresponding cocycle $\alpha_{i,\epsilon}$. This cocycle has coefficients in $\mathbb{F}_p$, but circular parameterization requires a cocycle with integer coefficients. We need to lift $\alpha_{i,\epsilon}$ to an integer cocycle. We do this by first constructing an integer 1-chain by replacing each coefficient in $\alpha_{i,\epsilon}$ with the integer in the same congruence class lying in the range $\{-\frac{p-1}{2}, \ldots \frac{p-1}{2}\}$. This almost always gives an integer cocycle $\hat{\alpha}_{i,\epsilon} \in C^1(R_\epsilon(\mathcal{X}), \mathbb{Z})$.[10] We can then apply circular parameterization to this cocycle.

In our computations, we used Ripser (Bauer, 2016) to compute the first persistent cohomology with coefficients in $\mathbb{F}_{47}$, giving persistence diagrams and the associated representative cocycles. The procedure described above to lift this cocycle always gave integer cocycles.

Acknowledgments

We thank Adrien Peyrache and György Buzsáki for sharing the data sets (Peyrache and Buzsáki, 2015; Peyrache et al., 2015) used in our analysis for free online.

References

Archer, E. W., Koster, U., Pillow, J. W., & Macke, J. H. (2014). Low-dimensional models of neural population activity in sensory cortical circuits. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems, 27* (pp. 343–351). Red Hook, NY: Curran.

Bauer, U. (2016). Ripser: A Lean C++ code for the computation of Vietoris-Rips persistence barcodes (Version 1.01). [Computer software]. https://ripser.org

Carlsson, G. (2009). Topology and data. *Bulletin of the American Mathematical Society*, *46*(2), 255–308.

Chazal, F., Oudot, S. Y., Glisse, M., & de Silva, V. (2016). *The structure and stability of persistence modules*. Berlin: Springer-Verlag.

Cunningham, J. P., & Yu, B. M. (2014). Dimensionality reduction for large-scale neural recordings. *Nature Neuroscience 17*, 1500–1509.

Curto, C., & Itskov, V. (2008). Cell groups reveal structure of stimulus space. *PLOS Computational Biology*, *4*(10), 1–13.

Dabaghian, Y., Brandt, V. L., & Frank, L. M. (2014). Reconceiving the hippocampal map as a topological template. *eLife*, *3*, e03476.

Dabaghian, Y., Mémoli, F., Frank, L., & Carlsson, G. (2012). A topological paradigm for hippocampal spatial map formation using persistent homology. *PLoS Computational Biology*, *8*(8), 1–14.

de Koning, E. (2011). Radial distance. http://psimpl.sourceforge.net/radial-distance.html

[10] If not, more work has to be done. See de Silva et al. (2011) for more details.

de Silva, V., Morozov, D., & Vejdemo-Johansson, M. (2011). Persistent cohomology and circular coordinates. *Discrete and Computational Geometry*, *45*(4), 737–759.

Edelsbrunner, H., & Harer, J. (2010). *Computational topology: An introduction*. Providence, RI: American Mathematical Society.

Frigola, R., Chen, Y., & Rasmussen, C. E. (2014). Variational gaussian process state-space models. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems*, *27*. Red Hook, NY: Curran.

Ghrist, R. (2008). Barcodes: The persistent topology of data. *Bull. Amer. Math. Soc. (n.s.)*, *45*(1), 61–75.

Ghrist, R. (2014). *Elementary applied topology*. CreateSpace.

Giusti, C., Pastalkova, E., Curto, C., & Itskov, V. (2015). Clique topology reveals intrinsic geometric structure in neural correlations. *Proc. Natl. Acad. Sci. USA*, *112*(44), 13455–13460.

Goodridge, J. P., Dudchenko, P. A., Worboys, K. A., Golob, E. J., & Taube, J. S. (1998). Cue control and head direction cells. *Behavioral Neuroscience*, *112*(4), 749–761.

Hardcastle, K., Maheswaranathan, N., Ganguli, S., & Giocomo, L. M. (2017). A multiplexed, heterogeneous, and adaptive code for navigation in medial entorhinal cortex. *Neuron*, *94*(2), 375–387.

Hatcher, A. (2002). *Algebraic topology*. Cambridge: Cambridge University Press.

Jones, E., Oliphant, E., Peterson, P. et al. (2001). SciPy. Open source scientific tools for Python [Software]. http://www.scipy.org

Kloke, J., & Carlsson, G. (2010). *Topological de-noising: Strengthening the topological signal*. https://arxiv.org/abs/0910.5947v2

Knierim, J. J., Kudrimoti, H. S., & McNaughton, B. L. (1998). Interactions between idiothetic cues and external landmarks in the control of place cells and head direction cells. *Journal of Neurophysiology*, *80*(1), 425–446.

Koyama, S., Pérez-Bolde, L. C., Shalizi, C. R., & Kass, R. E. (2010). Approximate methods for state-space models. *Journal of the American Statistical Association, 105* (489), 170–180.

Macke, J. H., Buesing, L., Cunningham, J. P., Yu, B. M., Shenoy, K. V., & Sahani, M. (2011). Empirical models of spiking in neural populations. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems*, *240*. Red Hook, NY: Curran.

McCullagh, P., & Nelder, J. A. (1989). *Generalized linear models*, 2nd ed. London: Chapman and Hall.

Mimica, B., Dunn, B., Tombaz, T., Bojja, V. S., & Whitlock, J. R. (2018). *Efficient cortical coding of 3D posture in freely behaving rats*. bioRxiv 307785.

Mizumori, S. J., & Williams, J. D. (1993). Directionally selective mnemonic properties of neurons in the lateral dorsal nucleus of the thalamus of rats. *J. Neurosci.*, *13*(9), 4015–4028.

Perea, J. A. (2016). *Multi-scale projective coordinates via persistent cohomology of sparse filtrations*. https://arxiv.org/abs/1612.0286lv3

Peyrache, A., & Buzsáki, G. (2015). *Extracellular recordings from multi-site silicon probes in the anterior thalamus and subicular formation of freely moving mice* [Data set]. http://crcns.org/data-sets/thalamus/th-1

Peyrache, A., Lacroix, M. M., Petersen, P. C., & Buzsáki, G. (2015). Internally organized mechanisms of the head direction sense. *Nat. Neurosci.*, *18*(4), 569–575.

Peyrache, A., Schieferstein, N., & Buzsáki, G. (2017). *Transformation of the head-direction signal into a spatial code*. bioRxiv.

Pfau, D., Pnevmatikakis, E. A., & Paninski, L. (2013). Robust learning of low-dimensional dynamics from large neural ensembles. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems*, *26*. Red Hook, NY: Curran.

Ramkumar, P. (2016). pyglmnet, Python implementation of elastic-net regularized generalized linear models [Software]. https://github.com/glm-tools/pyglmnet

Ranck, J. B. (1985). Head direction cells in the deep cell layer of dorsal presubiculum in freely moving rats. In G. Buzsáki & C. H. Vanderwolf (Eds.), *Electrical activity of archicortex* (pp. 217–220). Budapest: Akadémiai Kiado.

Rigotti, M., Barak, O., Warden, M. R., Wang, X.-J., Daw, N. D., Miller, E. K., & Fusi, S. (2013). The importance of mixed selectivity in complex cognitive tasks. *Nature*, *497*(7451), 585–590.

Singh, G., Memoli, F., Ishkhanov, T., Sapiro, G., Carlsson, G., & Ringach, D. L. (2008). Topological analysis of population activity in visual cortex. *J. Vis.*, *8*(8), 11.1–11.18.

Skaggs, W. E., McNaughton, B. L., & Gothard, K. M. (1993). An information-theoretic approach to deciphering the hippocampal code. In S. J. Hanson, J. D. Cowan, & C. L. Giles (Eds.), *Advances in neural information processing systems, 5* (pp. 1030–1037). San Mateo, CA: Morgan Kaufmann.

Spreemann, G., Dunn, B., Botnan, M. B., & Baas, N. A. (2015). *Using persistent homology to reveal hidden information in neural data*. https://arxiv.org/abs/1510.06629v1

Taube, J. S., Muller, R. U., & Ranck, J. B. (1990). Head-direction cells recorded from the postsubiculum in freely moving rats. II. Effects of environmental manipulations. *Journal of Neuroscience*, *10*(2), 436–447.

Tsodyks, M., Kenet, T., Grinvald, A., & Arieli, A. (1999). Linking spontaneous activity of single cortical neurons and the underlying functional architecture. *Science*, *286*(5446), 1943–1946.

Vejdemo-Johansson, M., Pokorny, F. T., Skraba, P., & Kragic, D. (2015). Cohomological learning of periodic motion. *Applicable Algebra in Engineering, Communication and Computing*, *26*(1), 5–26.

Yoder, R. M., Peck, J. R., & Taube, J. S. (2015). Visual landmark information gains control of the head direction signal at the lateral mammillary nuclei. *Journal of Neuroscience*, *35*(4), 1354–1367.

Yu, B. M., Cunningham, J. P., Santhanam, G., Ryu, S. I., Shenoy, K. V., & Sahani, M. (2008). Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity. *Journal of Neurophysiology*, *102*, 614–635.

Zhang, K. (1996). Representation of spatial orientation by the intrinsic dynamics of the head-direction cell ensemble: A theory. *Journal of Neuroscience*, *16*(6), 2112–2126.

Zhao, Y., & Park, I. M. (2017). Variational latent gaussian process for recovering single-trial dynamics from population spike trains. *Neural Computation*, *29*(5), 1293–1316.

Received July 19, 2018; accepted September 7, 2018.