Chapter 9

Analysis of population data

Given the vast numbers of neurons involved in cognition and information processing, it is perhaps surprising that observation of a single neuron's activity has ever yielded much insight into brain function. The focus through much of the history of systems neuroscience has been on the analysis of those neurons whose firing rates change strongly and reliably in a particular direction, either in response to a stimulus, or as an indication of a forthcoming motor response. This focus arose in part because of limitations of the experimental apparatus available—when recordings of individual neurons are hard to come by, it is important to find and select those neurons with the strongest task-dependent response. Moreover, the observation of a distinct change in a neuron's firing rate is both easier to explain and more dramatic—therefore, more newsworthy and publishable—than any correlation with behavior extracted more painstakingly from changes that are subtle if observed at the level of the individual neuron. Therefore, the desire to find individual neurons with strong, task-specific responses continues today.

Models of neural circuits—such as the majority of those presented in this book—are similarly focused on explaining the strong, task-dependent responses of certain neurons. In part, such models are easier to formulate and understand than those in which the encoding of information is almost impossible to pinpoint. In this book, the one network we have considered with near impenetrable encoding is the network of granule cells in the cerebellum, which in our model (Tutorial 8.4) encoded time. While the network activity can produce a temporally precise change in Purkinje cell activity, we would be hard-pressed to find a "time-tuned" neuron in the network. It is quite likely that much neural activity beyond the primary sensory and primary motor areas of any animal's brain is of this ilk[1-9]. We spend less time on such circuits in this introductory book, simply because they are far from being understood and they are more difficult to simulate (see [5,10-16] for examples and further reading)—not because they are less important for brain function.

The first step toward understanding the mechanisms, or even the general principles and algorithms involved, when they are not revealed by straightforward changes in the firing rates of individual neurons, is to employ appropriate methods to combine activity from many cells[17-21]. In this chapter, we consider three such methods of analysis, each of which can be considered the simplest prototype of a large set of related methods.

9.1. Principal component analysis (PCA)

Box 9.1. Principal component analysis (PCA) is a method for reducing high-dimensional data to fewer dimensions in a manner that aims to minimize loss of information.

Box 9.2. Dimensionality of data is the number of distinct values, or coordinates, used to define one data-point. The dimensionality can be the number of variables in the system, or the number of variables multiplied by the number of time-points measured.

Before describing PCA, it is important to understand the general concept of "high-dimensional data" and why "dimensionality reduction" can be useful. The ideas are of general applicability and value to multiple fields of data analysis, ranging from the sorting of spikes (described in the next subsection), to the unbiased scoring of elected officials by analysis of their voting records.

The high dimensionality refers to the number of values used to characterize a data-point. For example, a particular curve, such as a voltage trace, $V(t)$, can be characterized by a large number of sampled points along the curve. In this manner, a curve characterized by 100 points along the trajectory can be represented by a single point in 100-dimensional space. A dataset with many such curves would then be represented by one or more clusters of such points in the 100-dimensional space.

Similarly, the complete set of votes on a resolution can be characterized by a set of values, one value for each legislator voting. For example, votes in favor of, or against the resolution, or abstentions, can be ascribed values of "+1" or "-1" or "0" respectively. In this manner, the votes of 100 senators would be represented by a single point in 100-dimensional space. A dataset containing many such voting outcomes, would then be represented by one or more clusters of such points in the 100-dimensional space, just as in the previous example.

PCA is one of several methods for extracting useful information from such clusters of data points in high-dimensional space. At the heart of PCA is the ideas of choosing a basis in geometry—which means choosing a new set of coordinates. The new basis is chosen based on the variance of the data—and the new coordinates are ordered by the amount of variance they account for. This ends up being useful for several reasons, one of which being that in high-dimensional data, random, uncorrelated fluctuations do not produce such large variance in the data as does correlated variation. In many situations, the correlations among the data points are of prime interest, either because they reveal an underlying mechanism or coordination, or because different underlying processes, which produce distinct sets of correlations, can be separated and revealed.

9.1.1. PCA for sorting of spikes

In order to isolate the spikes of individual neurons from multi-electrode data, it is common to run PCA on the voltage traces of the electrodes (Figure 9.1). In this context, the separate dimensions correspond to the values of voltage in different small time bins around the peak of a spike, as measured on separate electrodes. If, for example, 40 time-bins are used to quantify the voltage trace at each spike, then the number of dimensions would be 40 multiplied by the number of voltage traces. With such characterization, it is possible to extract multiple distinct "spike signatures" and identify more neurons than electrodes used. For example, the data shown in Figure 9.1B appear separable as two distinct clusters of traces on the single electrode shown.

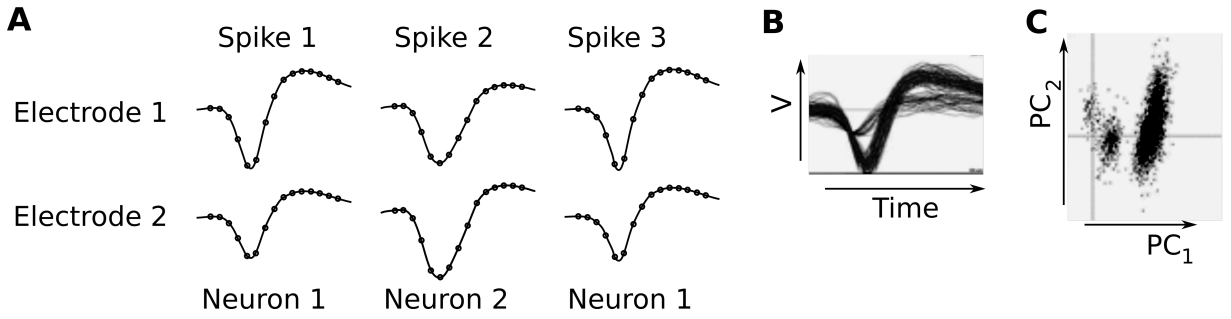

Figure 9.1. Separation of neural spikes by PCA. A. (Cartoon). When a spike is detected, the waveform is characterized by a number of values of the voltage (small circles), which together describe the characteristic shape produced on each electrode. Three spikes are depicted, each spike characterized by 15 values on each of two electrodes, which would produce a single point in 30-dimensions for each spike. Spikes 1 and 3 are similar because they arise from the same neuron, whereas spike 2 is significantly different, because it arises from a different neuron. **B.** Example traces of the multiple waveforms produced by several hundred spikes on a single electrode implanted in rat gustatory cortex. These traces group around two distinct forms, so most likely arise from two distinct neurons. Notice that the voltage trace, dependent on electric fields outside the neurons, can have a very different shape from the membrane potential of the neuron producing a spike. **C.** Each spike appears as a dot in high-dimensional space. The spikes separate into two clusters, representing the two putative neurons. Following PCA, the separation of the clusters is visible here when just the first two principal components of the data are plotted against each other. 40 data points per spike from a single electrode were used for this analysis. (Data provided by the laboratory of Don Katz, Brandeis University).

As noted above, PCA is helpful when there are correlations between the dimensions. For example, in multi-electrode recordings of electrical activity, any source of electric field, such as a single neuron, whose spike affects the voltage in more than one electrode, produces a correlation between the values recorded in the different electrodes. That is, whenever the neuron spikes, a certain shape of deflection of the voltage in one electrode will occur at the same time as a different shape of deflection of the voltage in another electrode. A different neuron would produce a different pair of simultaneous deflections in the two electrodes. Therefore, observation of a shape of deflection in one electrode correlates with the shape of deflection caused by the same cell in another electrode. Moreover, the values of a voltage-trace at successive time-points are highly correlated—if an electrode is at a point of high potential then a fraction of a millisecond later, the potential is likely to remain higher than average.

While one could imagine designing an algorithm to pick out the important features that differentiate the voltage traces of Figure 9.1A or 9.1B, these distinguishing features could vary from electrode to electrode and from cell to cell. The value of PCA in this context is that, once all traces are represented as individual points, it can allow us to see visually whether multiple clusters exist and use simple methods to separate the clusters.

9.1.2. PCA for analysis of firing rates

Once the activities of individual neurons are separated, correlations abound in the dynamics of extracted firing-rates. In this case, correlations between the activities of different cells can be caused by connections between the cells within a network, or by common "upstream" input. The interesting signal causing the correlations across dimensions can be more easily observed or extracted by choosing an appropriate linear combination of the measured data. For example, in a binary decision-making task (Section 6.5) one may want to add together the firing rates of cells responsive to one choice then subtract all the firing rates of cells responsive to the opposite choice, while ignoring the firing rates of cells with no choice preference. PCA can provide a systematic method for determining how much (and of what sign) each cell's firing rate should contribute to a particular "readout".

When analyzing neural activity, the initial, high number of dimensions is typically the number of neurons, with each dimension indicating the firing rate of a particular neuron. The variation of activity as a function of time would be a curve, or trajectory in the high-dimensional space as neural firing rates covary. If the covariation of activity lies predominantly along a 1D line, or a 2D plane, PCA would reveal this. For example, if the rates of all neurons deviated from their mean by different scaled amounts of the same function of time—*i. e.*, their rate changes were all in proportion to each other—the trajectory would be a straight line in firing-rate space. Even with noise added to each neuron's firing rate, PCA could reveal the straight line and produce a much less noisy estimate of the function of time coordinating the dynamics than that available from analysis of any neuron alone. In so doing, PCA may reveal the most important correlate of a stimulus or behavior to be found in the noisy neural activity. In Tutorial 9.1 we will practice such techniques.

9.1.3. PCA in practice

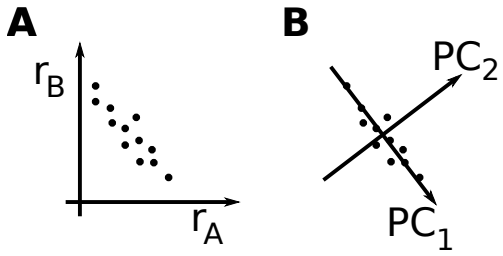

Figure 9.2. Selecting new axes with principal component analysis. A) Each data point corresponds to a specific measurement and comprises a pair of firing rates, r_A and r_B. **B)** The axis corresponding to the first principal component, PC_1, contains the most variance in the dataset. The second axis, PC_2, is perpendicular to the first.

The first step of PCA is the extraction of a new set of perpendicular axes sorted by the amount of variance of the data in the direction of each new axis. Figure 9.2 indicates how PCA would extract new axes from a pair of neurons whose firing rates are anti-correlated, meaning a higher rate of one neuron coincides with a lower rate of the other neuron and vice versa. In this example, the axis containing the most variance in the data, now labeled PC_1, is at 45° to the original axes and corresponds to the direction $r_A + r_B = constant$ (it could as easily have been the direction $-r_B - r_A = constant$). The second axis extracted,

labeled PC_2, must be perpendicular to the first, so in this case is constrained to be the direction $r_A = r_B$.

Box 9.3. Projection of data onto an axis (or plane) reduces the dimensionality of data in the same way that a 3D image can be projected onto a 2D plane as a photograph—all depth information is lost. For example, if a set of (x, y) points are projected onto the x-axis then only the x-values remain and the y-values are lost.

The variance in the data accounted for by PC_1 is so much greater than that accounted for by PC_2 in this example, that a description of each data point by its first principal component alone—its value of PC_1—would not cause much loss of information. That is, if the data points in Figure 9.2B were moved to the nearest point on the PC_1 axis (such a movement is perpendicular to the axis and is called a projection onto the axis—see Figure 9.3) then little change in their position would ensue.

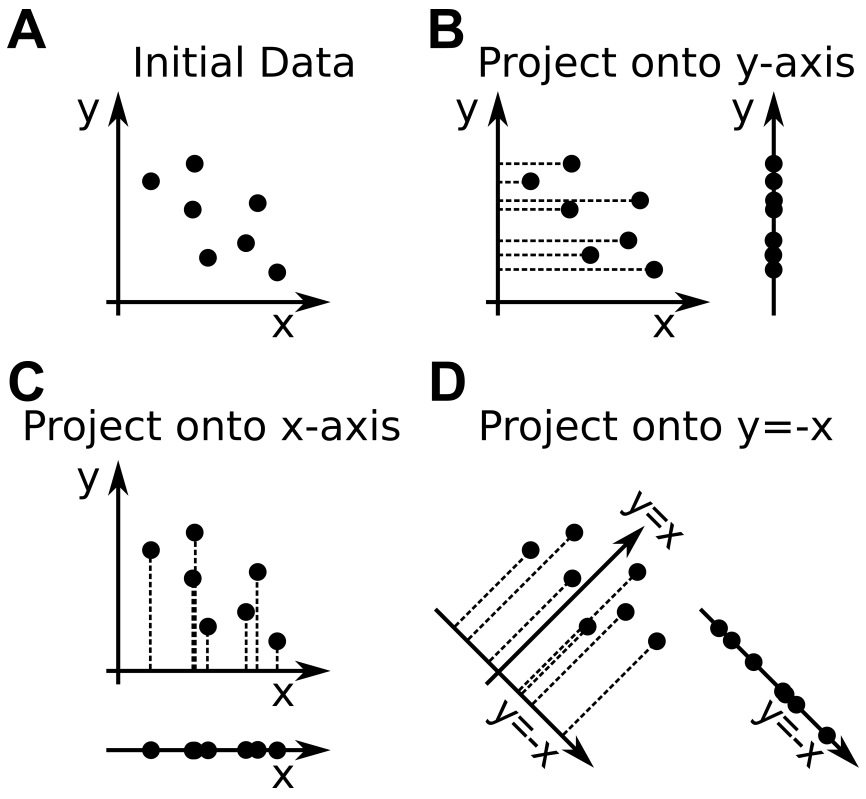

Figure 9.3. Projection of data. A. A set of data points in 2-dimensions can be projected to a single dimension in multiple ways. **B-D.** Projection of each data point (x_i, y_i) onto an axis is shown as a dashed line. The resulting set of projected data falls on a single line, so each data point is represented by a single number, its y-value (y_i) in B, its x-value (x_i) in C, or the scaled difference between the two $\left(\frac{x_i - y_i}{\sqrt{2}}\right)$ in D.

If the original data were three-dimensional (so had labels x, y and z, or equivalently, r_1, r_2 and r_3), it is possible that all the data points fall within a shape that looks like a tilted dinner plate. Such data would be closer to 2D as a plate is approximately a flat surface and a flat surface is a 2D plane. PCA would pick out two directions at right angles to each other within this plane, where the greatest spread in the data is found. If one plotted the data points on coordinates produced by these first two principal components one could see the data sitting essentially in an ellipse. The third principal component (in the direction of the thickness of the dinner plate, perpendicular to its plane) could most likely be ignored with little loss of information about the coordinates of any data point. Reducing the number of dimensions (or coordinates) used to describe data with minimal information loss is a form of data compression that can be used in some forms of data transfer.

Perhaps a key step in the use of PCA, or any other method for analysis of high-dimensional data, is the choice of what components of the data are used to produce new axes and extra dimensions for each data point, versus what components are used to produce new data points. For example, when firing rates of many neurons are measured as a function of time, the dimensionality could be the number of neurons. In this case, each time-point would provide a single data point. The set of data points for a single trial can then be joined together as a single curve, indicating the trajectory of neural activity as a function of time. PCA allows us to visualize such trajectories[5,22].

Alternatively, each time-point could add a new set of dimensions, such that the total dimensionality of the system becomes the number of neurons multiplied by the number of time bins. In this case, each trial would correspond to a single high-dimensional point (like each time window around a spike in Figure 9.1). Multiple trials are likely to produce a cluster of points, with trials from a different stimulus producing an alternative cluster (*cf*. Figure 9.1C). PCA is likely to extract the most significant distinctions between stimuli. These distinguishing features could then be mapped back to distinct signatures in the firing rate trajectories of the neurons.

Box 9.4. Covariance: The degree to which two series of data vary in the same direction as each other (positive covariance) versus in the opposite direction as each other (negative covariance). Zero covariance can arise if the two series vary independently of each other, or if either one or both of the series has no variation (zero variance).

Box 9.5. Eigenvectors: Eigenvectors are a set of orthonormal vectors—meaning perpendicular vectors of unit length—associated with a particular matrix. The eigenvectors are special, in that when multiplied by the matrix they are simply scaled rather than altered in direction.

Box 9.6. Eigenvalues: An eigenvalue corresponds to a particular eigenvector and is the amount by which that eigenvector is scaled when multiplied by the matrix.

9.1.4. The procedure of PCA

In this section, we describe the algorithmic steps that are required for PCA. In this description, each data point will correspond to the values of firing rates of each cell in a single time-bin, so the dimensionality is the number of neurons.

1) Center the original data by subtracting the mean in each dimension—*i. e.*, for every data point, subtract the mean (time-averaged) firing rate of the corresponding neuron, so we are analyzing deviations from the mean. The mean values should be stored if firing rates are to be recalculated. Subtraction of the mean facilitates rapid calculation of variances and covariances (recall that variance is calculated using deviations from the mean).

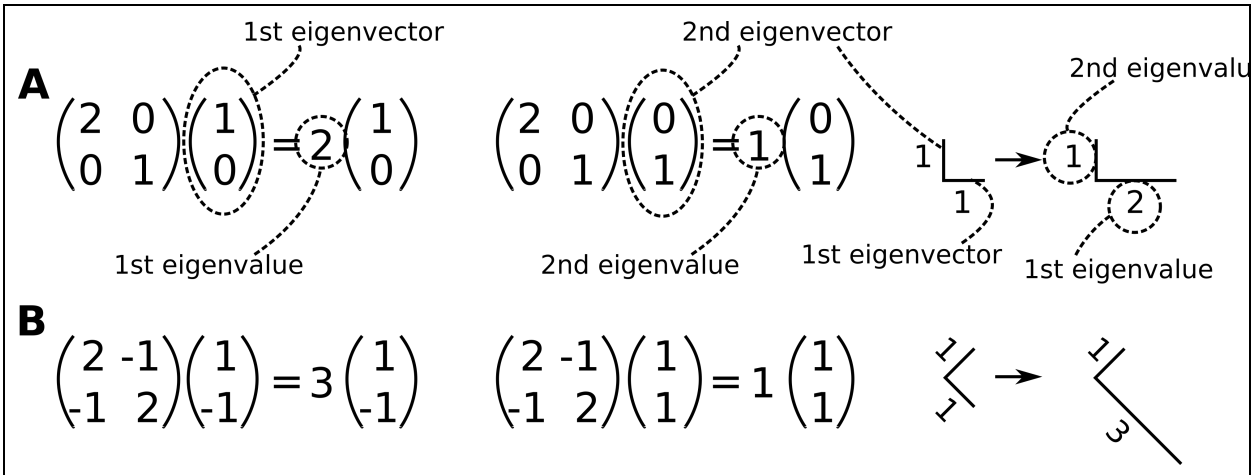

Figure 9.4. Eigenvectors and eigenvalues. A. The matrix $\begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$ has an eigenvector $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ directed along the x-axis with an eigenvalue of 2, because multiplication of that eigenvector by the matrix simply scales the vector by a factor of 2. The second eigenvector, $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$, is perpendicular to the first, directed along the y-axis, and has an eigenvalue of 1. Notice that the eigenvectors could equally have been chosen to be in the opposite direction, *e. g.*, $\begin{pmatrix} -1 \\ 0 \end{pmatrix}$ is an eigenvector, and they would still be perpendicular with the same eigenvalue. **B.** The matrix $\begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$ has eigenvectors rotated by 45°. Since eigenvectors should be of unit length, the actual eigenvectors are $\frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ -1 \end{pmatrix}$ and $\frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix}$, with eigenvalues of 3 and 1 respectively. Notice that when the matrix multiplies a vector that is not an eigenvector, the result is not a scalar multiple. *e. g.*, $\begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 2 \\ -1 \end{pmatrix}$ means a vector along the x-axis is rotated clockwise as well as scaled by the matrix in B.

2) Calculate the covariance matrix of the centered data (this can also be done without step-1 since the covariance is mean subtracted). The covariance matrix will be a symmetric $P \times P$ matrix if there are P neurons. Each entry in the covariance matrix is calculated for two neurons, by taking the product of their firing rates together in each time-bin then calculating the mean of these values across time bins. The diagonal of the matrix is simply the variance in firing rate of the corresponding neuron.

3) Find the eigenvectors of the covariance matrix with their eigenvalues. (An eigenvector is a particular direction or axis, such that a vector in this direction, when multiplied by the matrix, is simply scaled in the same direction. The amount of scaling is the eigenvalue—see Figure 9.4. Many coding languages can produce eigenvectors and eigenvalues of matrices using a single command—it is "`eig`" in Python if you `import` the `numpy.linalg` module.)

4) Sort, then order, the eigenvectors from the one with the largest eigenvalue to the one with the smallest eigenvalue (all eigenvalues are non-negative real numbers, since the covariance matrix is real and symmetric). The sum of the eigenvalues is the sum of the variances of firing rates of all neurons. When each eigenvalue is divided by this sum, the result yields the fraction of the total variance of neural activity due to coherent variation of the system in the direction of the corresponding eigenvector.

The eigenvectors, each of which defines a direction providing a new axis, are perpendicular when created, so we have generated a new set of perpendicular axes for the data—*i. e.*, a new basis. The axes are ordered according to the directions with highest to lowest variance in the original data, which is the order of the principal components.

9.2. Tutorial 9.1. Principal component analysis of firing rate trajectories

In this tutorial, you will first generate firing rate trajectories of 50 neurons. Each neuron's firing rate is produced by a weighted combination of two input signals, mixed in with a lot of noise. You will then use PCA to produce less noisy firing rates for each cell (de-noising) and to extract the input signals from the noisy set of firing rates.

1. a) Define two vectors to represent time-dependent oscillating inputs, I_A and I_B, such that $I_A = A\sin(2\pi f t)$ and $I_B = B\cos(2\pi f t)$, with frequency $f = 0.5$Hz, and amplitudes $A = 20$ and $B = 10$. You can use time-steps of 1ms and a duration of 10s.

   b) Set up a matrix to contain the firing rates of 50 neurons across this time-span. Generate the firing rate of each neuron as a column in the matrix, with each row representing a separate time-point. The time-dependent firing rate of a neuron, labeled i, is given by:

   $$r_i(t) = 100 + W_i^{(0)} I_0 + W_i^{(A)} I_A + W_i^{(B)} I_B + \sigma \cdot \eta_i(t),$$

   where $I_0 = 50$, the static input weights, $W_i^{(0)}$, $W_i^{(A)}$, and $W_i^{(B)}$, are each independent numbers selected separately, once for each neuron, from the Normal distribution

with unit standard deviation and zero mean (defined as $N(0,1)$ via numpy's `random.randn()` in Python). σ=10 scales the noise, and $\eta_i(t)$ is a series of Normally distributed random variables, $N(0,1)$, selected independently at each time-point for each neuron.

c) Carry out principal component analysis on the rate matrix, using (in Python) the `PCA` function imported from `sklearn.decomposition`:

```
pca = PCA()
SCORE = pca.fit_transform(rate)
COEFF pca.components_
EXPLAINED = pca.explained_variance_ratio_
MU = pca.mean_
```

Note that the `pca` function requires the data in each dimension to be stored as a set of column vectors, with each column being a different dimension (*i. e.,* each row is a time point and each column a neuron in this tutorial). The important outputs for this tutorial are calculated above to produce:
`COEFF`, which defines the new basis. Each row vector in `COEFF` is a principal component, which is an eigenvector of the covariance matrix. The vector has unit length with column elements given by the relative contribution of each neuron (in this example) to the principal component.
`SCORE`, contains the representation of the centered data in the new basis. Each row of score which corresponds to a single data point, with each column entry its value in the new coordinate system, that is, its projection on each successive principal component.
`EXPLAINED` is a vector containing the percentage of variance explained by each principal component.
`MU` is a vector containing the mean firing rates of each neuron, as needed to recreate the original rates.

d) Plot the first column of coefficients, `COEFF`, (corresponding to the contribution of each neuron to the first principle component) against the vector of input weights, $W_i^{(A)}$, and, on a separate subplot, the second column of coefficients (corresponding to contribution of each neuron to the second principal component) against the vector of input weights $W_i^{(B)}$. Does the sign of the slope of any observed trends matter?

e) Plot the variable, `EXPLAINED`, to ensure the bulk of the variability in the data is contained in the first two principal components and calculate that fraction explained by summing its first two values.

f) To de-noise the data, you will produce a new matrix of firing rates by multiplying the first two columns of `SCORE` by the first two rows of `COEFF` and adding to all

entries of each column produced in this manner, the value of the corresponding entry of `MU` (the mean rate of that neuron).
In this step, we are assuming that once in the coordinates of the principle components, any changes in rates on principal component axes 3 or higher correspond to noise fluctuations that can be ignored. Therefore, we are recreating the firing rates of the original neurons just using the projection of each neuron's firing rate on the axes of the first two principal components.

g) Compare the behavior of the de-noised data with the original data by plotting (on separate subplots) the original rates of two neurons as a function of time, then the corresponding columns of the new matrix to reveal the de-noised rates as a function of time.

h) Select two neurons and plot the rate of one against the other, both using the original noisy rates, and, in a separate subplot, using the de-noised rates.

i) Plot separately the first and second columns of the matrix, `SCORE`, to show the time-dependence of the system's first and second principal components.

Be sure to comment on all of your observations and relate them to the initial set of inputs.

2. Repeat question 1, but in part b) define $I_A = A\sin(2\pi f_A t)$ and $I_B = B\cos(2\pi f_B t)$, where $f_A = 1$Hz and f_B=0.5Hz, keeping all other parameters the same.

3. Repeat question 1), but define one input as a slowly ramping current, $I_A = At$ (from $t = 0$ to 10s), and the other as a transient signal during the ramp, $I_B = B\sin(2\pi f t)$ for 4s $< t <$ 5s, otherwise $I_B = 0$.

9.3. Single-trial versus trial-averaged analyses

Most analyses of population data based on the firing rates of neurons initially average over many trials using the methods of Chapter 3. Such across-trial averaging can be necessary to produce a smoothly varying firing rate from the series of instantaneous spikes. However, the averaging procedure implicitly assumes that the multiple trials contain fundamentally identical data, and when this is not the case, the point of alignment matters. For example, in the model of a decision-making network (Tutorial 6.2) the average activity depended on whether trials were aligned to stimulus onset or to response time.

In the absence of reliable, reproducible circuit dynamics following alignment of activity to an externally identifiable time-point, it is preferable to analyze the data with single-trial methods. These are methods that either take into account across-trial variability, or do not assume multiple trials. In this chapter, we will consider two such methods, Hidden Markov modeling (HMM, Section 9.5) and Bayesian filtering (Section 9.6).

Single-trial methods are particularly beneficial when multiple neurons are recorded simultaneously, or more generally, when many simultaneously acquired data streams can be combined. Modern methods of electrophysiology, such as use of high-density electrode

arrays[23], allow for the extracellular recording of spikes from many neurons, even hundreds, simultaneously. Optical imaging, most commonly using calcium-responsive (and increasingly voltage-sensitive) dyes[24], allow for the activity of an even greater number of cells, perhaps thousands[25,26], to be observed almost simultaneously (only "almost", because there is typically a need for repeated scanning of the image line by line). Furthermore, non-invasive recordings of neural activity in humans typically acquire multiple data streams simultaneously. For example, using electroencephalography (EEG) and functional magnetic resonance imaging (fMRI), neuroscientists record from tens of electrodes and millions of voxels (volume elements) respectively. All of these methods produce data that have the potential to contain rich information within temporal correlations that may be obscured if the data are averaged across multiple trials.

Box 9.7. Electroencephalography (EEG) is the analysis of data acquired from multiple electrodes placed onto the scalp. Each electrode, typically a small flat circle, detects the electric fields produced by neural activity of groups of aligned cells within the brain. The electrodes are usually connected together in a "hood" so as to surround the head.

Box 9.8. Functional magnetic resonance imaging (fMRI) is a measurement of blood flow via its oxygenation level in many thousands of tiny cuboids called voxels throughout the brain, using the behavior of hydrogen nuclei in very strong magnetic fields. Since increased neural activity causes increased blood flow to the active neurons, fMRI indicates which voxels contain responsive neurons and how those voxels are correlated as a subject carries out different cognitive tasks, while strapped within a huge magnet.

9.4. Change-point detection

As a prelude to HMM, we consider the problem of a single, noisy spike-train, whose underlying firing rate changes discretely at a single point in time. The goal is, from observations of the spikes alone, to detect if and when the firing rate changes. If the change is drastic, then the change-point may be visible by eye alone. If multiple trials are carried out and the change-point is at an identical time in each trial, then the PSTH (Section 3.1.2) could reveal the change-point, even if the rates differ by a small amount. However, with just a single trial and a small change in firing rate, the change-point can only be estimated probabilistically. Here we will consider how to find the probability of a change-point at any time-point in the trial and use the maximum of this probability to estimate the change-point in trials when we predict there is such a change. To proceed, we assume spikes are produced with Poisson statistics at unknown underlying rates (see Section 3.3.3).

If we split a spike-train with a number of spikes, N, and with a duration, T, into two intervals, the first interval containing N_1 spikes and with a duration of T_1, then the second interval contains $N_2 = N - N_1$ spikes and has a duration of $T_2 = T - T_1$. We can first ask the probability of such an apportioning of spikes by chance.

Although the rates of the two processes are unknown, we can show (Appendix B) that the optimal rates are $r_1 = N_1/T_1$ and $r_2 = N_2/T_2$. These rates are what one might

expect—if we count a given number of spikes in a given time interval our best guess at the rate of any underlying process is the mean rate observed, calculated as number of spikes divided by the time interval.

The probability of the observed sequence of spikes arising from two processes with a transition at a certain time-point is equal to the product of the two individual sequences on either side of that time-point arising, each with its own rate. Some analysis (Appendix B) shows that this product is proportional to:

$$P(r_1, N_1)P(r_2, N_2) \propto (r_1)^{N_1}\exp(-N_1)(r_2)^{N_2}\exp(-N_2). \quad \text{Eq. 9.1}$$

Computationally, since the product of terms $\exp(-N_1)\exp(-N_2) = \exp[-(N_1 + N_2)]$ depends only on the total number of spikes, not on the change-point, we just need to look for the maximum of:

$$(r_1)^{N_1}(r_2)^{N_2} = (N_1/T_1)^{N_1}(N_2/T_2)^{N_2} \quad \text{Eq. 9.2}$$

as we vary T_1 (with $T_2 = T - T_1$), to find the most likely change-point.

It should be noted that we have used Bayes' Theorem (section 1.4.3, Eq. 1.32) to obtain this result, assuming a uniform prior on firing rates (*i. e.,* all rates are equally likely prior to the observation) and a uniform prior on the time of the change-point (the change in rates is equally likely to occur at any point in the time-window prior to observation). Given those assumptions of equal priors (which defines a Maximum Likelihood Estimate), we have been able to assume the probability of a particular firing rate and time-interval given the spike train is proportional to the probability of that particular spike train given the firing rate and time-interval. That is, Bayes' Theorem simplifies:

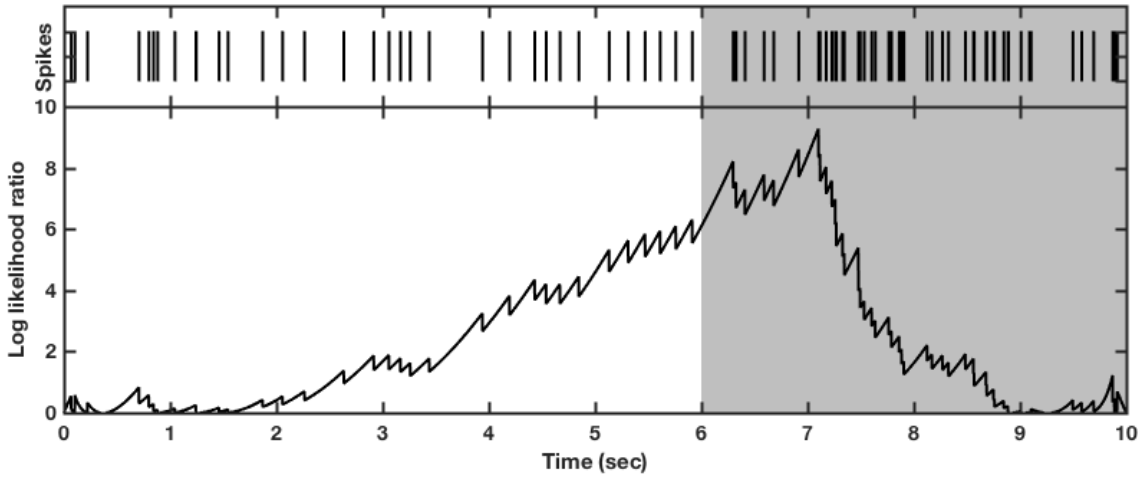

Figure 9.5. Detecting a change in firing rate. Top. Each vertical line indicates the time of a spike generated by a Poisson process whose rate changes from 5Hz (white region) to 10Hz (gray region) at a time of 6 seconds (the change-point). **Bottom.** The log likelihood ratio is the log of the probability of the spikes being produced by a Poisson process whose rate changes at that given point of time divided by the probability of those spikes being produced by a process whose rate does not change. In practice, it is rare that the log likelihood ratio exceeds 4 in the absence of a rate change. This figure was produced by the online code `single_jumprate_detect.m`.

$$P(r_i, T_i|\{s\}) = \frac{P(\{s\}|r_i, T_i)P(r_i, T_i)}{P(\{s\})} \propto P(\{s\}|r_i, T_i), \quad \text{Eq. 9.3}$$

where the final proportionality is valid if the prior, $P(r_i, T_i)$, is independent of r_i and T_i. In Eq. 9.3, r_i is the rate and T_i is the duration of any interval of fixed rate and $\{s\}$ indicates the set of spike times within the interval.

We can take the calculation of Eq. 9.2 one step further and compare the likelihood of the change-point at a given time, T_1, to the likelihood of the spikes arising from a process with the same fixed rate at all times. This requires a division of Eq. 9.2 by $(N/T)^N$ and produces a likelihood ratio (Figure 9.5). Simulations suggest that a likelihood ratio that exceeds 50 for some values of T_1 is an indication of a change in rates within the entire time interval (see Tutorial 9.2). Again, the value of T_1 that produces the greatest likelihood ratio is the value best chosen as the change-point. Using this method, the sub-intervals so obtained can be further investigated to assess whether there is evidence for an additional transition in firing rates within each sub-interval.

9.4.1. Computational note

When calculating the probabilities of spike trains, the product of many quantities yields terms that grow exponentially with the number of spikes, N. These terms can produce "overflow"—a condition where the corresponding numbers are greater than those able to be stored given the computer's standard memory allocation per number—or, when the numbers are combined to calculate the probability, "underflow"—a condition where the result can be so small that the computer equates the very small probability to zero. Such problems can be avoided by calculating the logarithm of the probabilities (or the log likelihood ratio in this case). The product of individual probabilities then converts to the sum of their logarithms, producing numbers on the order of N, which can be dealt with easily. The point of maximum probability is identical to the point at which its logarithm is a maximum, so the latter can be used to estimate a change-point.

9.5. Hidden Markov modeling (HMM)

Box 9.9. A Markov process is a dynamical process based on states, with the probability of a state transition per unit time being based only on the current state, not on the prior history of states, except in so far as such history led to the current state. In this sense, Markov processes are considered memoryless.

A Hidden Markov model (HMM) describes a system in terms of discrete states with transitions between them (Figure 9.6). The parameters of the states are not directly observable, but can be estimated from the times of emitted events, which in our formulation will be the spike trains. In this sense, Hidden Markov modeling resembles the change-point detection method of the previous section, where the spike trains are observed, but the underlying firing rate (the hidden parameter) is unknown. Also like the change-point detection method, a Hidden Markov model can produce the probability of a change in state as a function of time, when initially the number of state changes and their timings (if there are any) are unknown.

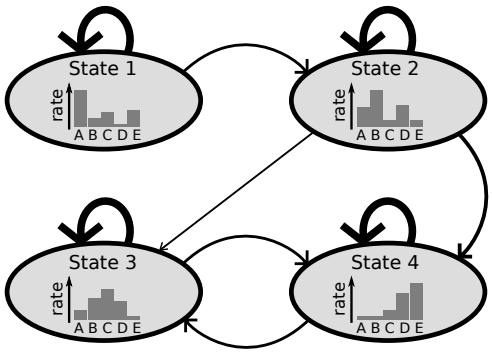

Figure 9.6. Example of a Hidden Markov model. A model with four states (ovals), as shown, would be defined by a 4x4 transition matrix, indicating the probability per time bin of a transition from one state to another (represented as arrows between ovals). In models of neural activity, the strongest transition is found to be from a state to itself, indicating the persistence of a state across many time-bins. Each state is defined by the probability of spike emissions, defined by the firing rate of each cell. In this example, rates of 5 neurons (labeled A-E) define the states and are depicted as histograms. The set of rates is unique to each state.

Hidden Markov modeling works well in the analysis of neural activity patterns, when multiple neurons change their rates coherently. For example, if one neuron were to change its firing rate from 5Hz to 10Hz, it would take a few hundred milliseconds around the transition point to accumulate enough spikes to signify a rate change. However, if ten such neurons all changed their rate at the same transition point, both the duration of trial needed to detect the change and the uncertainty in the time of the transition would reduce by ten-fold.

Box 9.10. Transition matrix: A matrix denoting the probability per time-step of the system making a transition from the current state (the row index) to another state (the column index). Since the system will be in one state or another on the next time-step, the elements on each row of a transition matrix sum to one. The diagonal elements correspond to the probability of remaining in that state on successive time-points.

Box 9.11. Emission matrix: A matrix denoting the state-dependent probability per time-step of emitting a particular event, which in neural recordings could be a spike from a particular neuron, a set of multiple spikes, or an absence of spike. Since the set of events must cover all possibilities, the sum of entries in each row of an emission matrix, which corresponds to the probabilities of each event in a given state, must sum to one.

An HMM possesses the Markov property in two manners. The Markov property means that the probability of a transition or an emission depends only on the current state of the system, not on the past sequence of states. The first manner in which the Markov property arises in HMMs is in the transition probability, which can be represented as a matrix denoting the probability per unit time of a transition from one state (which must be the current state) to another. Given the current state, the probability per unit time of transitioning to any new state is constant, independent of how long the system has spent in the current state. The second manner in which the Markov property arises in HMMs is in the emission probability, which again can be represented as a matrix denoting the constant

probability per unit time of any event given the current state. In practice, for neural spike trains, this means that spikes are emitted as a Poisson process (Section 3.3.3).

Hidden Markov models are more complicated than the example of change-point detection we saw in the prior subsection, for several reasons:

1. Spike trains from multiple neurons contribute to the analysis, so an individual "event" in a small time-bin is not just the emission or absence of a spike, but can be a spike from any one of the neurons, or a combination of spikes from different neurons. Related to this, a state of the system would be described by the probabilities of all of the possible "events", requiring as a minimum the firing rates of all neurons.
2. The number of possible states is usually greater than two, so several sets of firing rates coordinated across the measured neurons is common.
3. Transitions can be possible in a non-sequential order, with reverse, or back-and-forth transitions also possible (for example, commencing from State 1 in Figure 9.6, one can follow arrows to produce a sequence of states such as 1-2-4-3-4-3).
4. The output of HMM produces the probability per unit time of being in each state, so provides an indication of the abruptness of any state changes.

The model is then defined by two matrices, the matrix of emission probabilities and the matrix of transition probabilities, plus an initial state, all of which are optimized by an iterative algorithm[27,28].

Box 9.12. Iterative algorithm: A computational procedure that repeatedly loops through a set of calculations, using the result of the previous loop as an input to the next loop. An initial input must be provided, which often constitutes an initial "guess" at the solution, which gets improved via an optimization process on successive loops through the code.

In the example shown in Figure 9.6, the emission probabilities for the 5 neurons should be contained in a matrix with a size of at least 4x6, like:

$$\begin{pmatrix} 0.20 & 0.03 & 0.05 & 0.01 & 0.06 & 0.65 \\ 0.10 & 0.20 & 0.03 & 0.10 & 0.03 & 0.56 \\ 0.05 & 0.12 & 0.15 & 0.10 & 0.02 & 0.56 \\ 0.01 & 0.01 & 0.04 & 0.14 & 0.20 & 0.60 \end{pmatrix}.$$

The matrix must have 4 rows, one for each state (1-4), and at least 6 columns, given the possibility in a time-bin of a spike from any of the 5 neurons (A-E) plus the possibility of an absence of spikes in the time-bin. Each entry in such an emission matrix is the probability per time bin of a spike being produced by a given neuron (or no neuron) in the corresponding state. In this formulation, the possibility of multiple spikes per time-bin is excluded and the first five values in each row are proportional to firing rates of the neurons (the histograms of Figure 9.6) while the last column ensures that the rows sum to one. When more than one spike occurs in a time-bin in the data, one would have to shift the randomly chosen lost spikes to neighboring time-bins, or lose them (in which case rates would be slightly underestimated in the model). More accurately, combinations of multiple

spikes can be included in a more extensive emission matrix, with each possible spike-combination producing an extra column in the emission matrix.

In the same example, the transition probabilities would be contained in a 4x4 matrix like:

$$\begin{pmatrix} 0.98 & 0.02 & 0.00 & 0.00 \\ 0.00 & 0.97 & 0.01 & 0.02 \\ 0.00 & 0.00 & 0.98 & 0.02 \\ 0.00 & 0.00 & 0.02 & 0.98 \end{pmatrix}$$

where the large entries on the diagonal indicate that across consecutive time-bins the system is most likely to remain in the same state—a common result of such analysis of neural data[29-31]—while the off-diagonal entries indicate the probability of transitioning from one state (given by a row) to another state (given by the column).

Alongside the model, which is always optimized according to a particular dataset, is the analysis of the data in terms of the model. Such analysis yields the probability, in each time-bin, of the system being in one state or another. Rapid changes in these probabilities are indicative of sudden changes in the firing rates underlying the emission of the observed spikes. A single model can be trained from multiple trials of a task, with the final analysis yielding different timings of transitions and even different sequences that best match the data on the different trials[30-32].

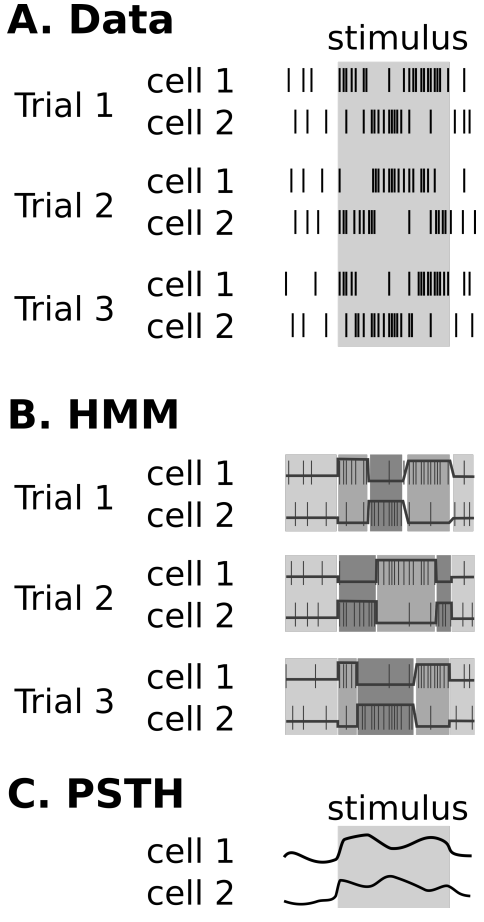

Figure 9.7. HMM extracts trial-specific state sequences. A. Sketch of spike trains from two cells, which respond preferentially to one of the two percepts that arise when a stimulus induces the bistable state. On separate trials the duration of percepts and the initial percept can vary. **B.** A Hidden Markov model of the system would, most likely, extract three states (indicated by the shade of gray), one pre- and post-stimulus, the other two corresponding to the distinct activity states present during the distinct percepts when the stimulus is present. The firing rates, shown as continuous bold lines, transition between distinct values, one value for each cell within each state (*cf*. Figure 9.6). State transitions are evaluated probabilistically, with short periods without shading between states representing periods when the model does not predict a single state's presence with high probability. **C.** A peri-stimulus time histogram (PSTH), like other standard analyses, which begin by averaging data across trials aligned to stimulus onset, would obscure evidence of these state transitions.

In summary, HMM assumes all trials follow the same probabilistic framework for the progression of neural activity through distinct states, but allows for trial-to-trial differences in the activity patterns that arise from the framework. For example, during perceptual bistability (Section 7.5) the same perceptual states are present on different trials, but both the transition times between percepts, and the identity of the initial percept, can vary. Therefore, the trial-averaged data would merge the activity arising from each of the two percepts and obscure the most important features of the neural dynamics that correspond to the perceptual switches[33,34]. In this and similar examples, the underlying neural activity is best analyzed via HMM (Figure 9.7).

9.6. Tutorial 9.2. Change-point detection for a Poisson process

Neuroscience goals: learn how to extract state changes from noisy spike-trains and learn some limitations of the process.
Computational goals: calculation of log-likelihood ratios to combine probabilities; produce scatter plots and a correlation coefficient.

In this tutorial, you will produce a Poisson process for a single neuron, whose rate changes abruptly at a randomly determined time-point within a trial. You will produce multiple such trials and assess how well a change-point detection algorithm extracts the correct change-point. Explain what you observe.

1) Define a set of 50 different change-points, each randomly chosen in the interval from 0 to 10s.

2) Produce a set of 50 spike trains, one spike train for each value of the change-point, with Poisson emission of spikes at a rate $r_1 = 5$Hz before the change-point and at a rate $r_2 = 10$Hz after the change-point.

3) For the first spike-train, plot as a function of T_1, which can vary in 1ms increments from 0.001sec to 9.999sec, the log likelihood ratio indicating the probability of the spike-train being produced by an inhomogeneous Poisson process with a rate-jump at T_1 divided by the probability of it being produced by a single homogeneous Poisson process. That is, you can use the formula:

$$P(T_1) = \left(\frac{N_1}{T_1}\right)^{N_1}\left(\frac{N_2}{T_2}\right)^{N_2} \Big/ \left(\frac{N}{T}\right)^{N}$$

which yields

$$\ln\left[P(T_1)\right] = N_1 \ln\left[\frac{N_1}{T_1}\right] + N_2 \ln\left[\frac{N_2}{T_2}\right] - N \ln\left[\frac{N}{T}\right],$$

where $T = 10$sec, $T_2 = T - T_1$, N is the total number of spikes in time T, N_1 is the number of spikes before T_1, and $N_2 = N - N_1$ is the remaining number of spikes after T_1 (in the sub-interval of length T_2).

4) Estimate the change-point as the value of T_1 with maximum log-likelihood ratio.

5) Plot the estimated firing rate as a function of time, assuming estimated rates of N_1/T_1 and N_2/T_2 respectively before and after the estimated change-point. On the same graph plot the firing rate used to generate the spike train.

6) Calculate the square-root of the mean-squared error in your estimate of the firing rate, and the square-root of the mean-squared error if the firing rate were assumed to be fixed (at N/T) for the entire trial. Plot these two values as a point on a scatter graph.

7) Repeat the calculation in 3) (but not the plot) and repeat 4)-6), for all 50 trials.

8) Calculate the correlation between the estimated change-point and the change-point used to generate each spike-train. Plot these values on a scatter plot and comment on the results.

Box 9.13. The hippocampus, named after its shape as "seahorse", is a region of the brain deep within the cerebral cortex, known to be essential for formation of memories in context and of sequences of events (episodic memory). The hippocampus is also highly involved in navigation, as its neural activity is location-dependent.

Box 9.14. A place field is the receptive field of a neuron, called a place cell, whose firing rate is greatest at specific locations in the environment. The place field is the set of locations producing high firing rate in such a cell.

9.7. Decoding position from multiple place fields

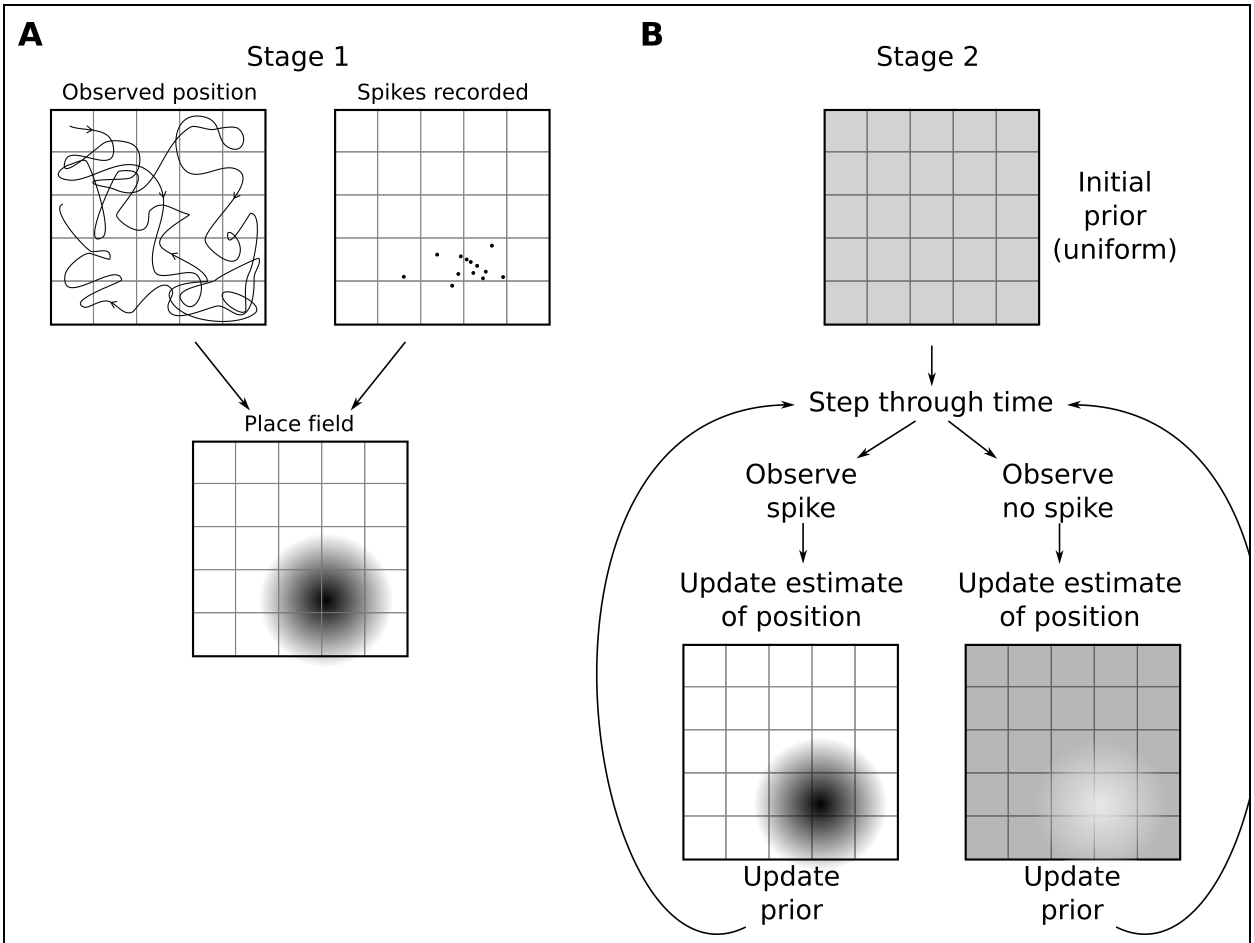

Figure 9.8. Algorithm for decoding position from place fields. A. In Stage 1, place fields are calculated for each neuron (bottom), by combining the observed position of the animal as a function of time (trajectory, top left) with the spikes emitted by a neuron (dots, top right). **B.** In Stage 2, the position at each time-point is estimated as a probability distribution, which is updated every time-step. An initial prior distribution is assumed to be flat, with no evidence for a particular location of the animal (top). If a neuron emits a spike (left), the neuron's place field is multiplied by the prior, to obtain an updated estimate of the animal's position (bottom left). The updated estimate becomes the new prior in the next time-step. When a neuron does not emit spikes (right), the estimate of position is also impacted, and a new prior is produced (bottom right). In this figure, the darker shading of a region indicates the greater the firing rate of the neuron in A, and the greater the probability of it being the animal's location in B.

Neurons in the hippocampus have firing rates that depend on the spatial location of an animal. Within a particular environment, the place fields—meaning the set of locations at which a neuron fires—are reliable for periods ranging from a few hours or days, possibly up to several weeks[26], and are compact, peaking at a particular location in space and decreasing monotonically with distance away from the peak. Cells with such responses are called place cells.

In this section, we will consider how information from the spike trains of multiple place cells can be combined to provide an estimate of an animal's position that is more precise than one might expect given the number of neurons and size of their place fields. The method requires the multiplication together of probability distributions, where each information source can provide one such distribution. For a review of combining probabilities, see Section 1.4.3. Two main principles are required in these methods:

1) When combining information from multiple sources, the probability of an occurrence is calculated by multiplying together the contributions from the separate information sources.
2) The sum of probabilities over all possible occurrences yields one. When decoding an animal's position, an occurrence means the animal being at a particular location, so the probability distribution means the probability of the animal being at any point in space. This second principle means that the sum over all locations of the probability distribution is one—that is, the animal cannot be in two places at once and it must be somewhere.

An important aspect of the approach used here is the incorporation of the most recent estimate of an animal's position as a probability distribution in the calculation of the estimate of its position at the next time-point. Specifically, the latest estimate of position produces the prior probability at each location for the next estimate, as shown in Figure 9.8B. The prior probabilities need not be identical to the previous position estimate, because the rat is known to move and its typical movements—how much it shifts position between time-points—can be taken into account in the update (see below). Therefore, three distinct sources of information are combined together to update the estimate of the animal's position:

1) The probability distribution produced for the previous estimate of position.
2) A model of the animal's typical movements to update the previous estimate of position into a prior for the next estimate of position.
3) A probability distribution for each cell, proportional to the place field of each cell that spikes, or with a dip at the place field of each cell that does not spike. These probability distributions are multiplied together with the prior to produce a new position estimate.

These methods, called Bayesian filtering[35], were developed by Emery Brown, Uri Eden, and colleagues[36,37] and are similar to a method called Kalman filtering (see, *e. g.*,[38] for an example and see[39] for an introduction).

When decoding position from neural activity in this manner, two stages are required (Figure 9.8):

Stage 1: The position must be observed while spikes from each cell are counted. The firing rate of a cell in each position can then be calculated as the number of spikes produced while in a position divided by the total time spent in that position. For such a calculation, one can split the continuous environment into a square grid. The resulting set of spike counts will be quite noisy, so a smooth function should then be fit to the data. The most common function is a two-dimensional Gaussian, which has a maximum at a particular location and a spread whose standard deviation can vary as a function of direction to define an ellipse, whose long axis can have any direction. For the code used to produce Figure 9.9, we just assume circular two-

dimensional Gaussians, such that the standard deviation is the same in all directions. These two-dimensional Gaussians define the place-fields of each neuron.

Stage 2: The estimated place-fields are used to decode the position of the animal using the subsequent spike trains. The firing rates can be converted to a probability of a spike in a small time-bin. These probabilities can be subtracted from one to obtain the probability of no spike in a small time-bin. Bayes' theorem must then be used to obtain the probability of the animal being in a particular location given a spike (or no spike) in a time-bin using the probability of a spike (or no spike) when in that location calculated in Stage 1. For example,

$$P(position|spike) = \frac{P(position)}{P(spike)} P(spike|position) \quad \text{Eq. 9.4}$$

where

$$P(spike) = \sum_{all\ positions} P(spike|position). \quad \text{Eq. 9.5}$$

Use of Bayes' theorem (Eq. 9.4) for each neuron's place-field is then akin to normalization, so that each probability distribution for location given a spike (or no spike) from a cell sums to one.

The procedure then is to step through time. First, take the previous probability distribution for the animal's location. Alter that distribution as necessary to indicate how the animal could change its location from one time-bin to the next, to produce a prior. Then, for each neuron, successively multiply the prior either by its probability distribution for location given a spike (when it spikes, Figure 9.8B bottom left) or by its probability distribution for location given no spike (when it does not spike, Figure 9.8B bottom right). Finally normalize the resultant distribution so that it sums to one.

The whole procedure can be called Bayesian filtering, because the method of combining the prior probability distribution with the incoming information is an implementation of Bayes' rule. That is, we can expand Eq. 9.4 for the whole process and write it as Bayes' theorem:

$$P(position|set\ of\ spikes) = \frac{P(position)P(set\ of\ spikes|position)}{P(set\ of\ spikes)}. \quad \text{Eq. 9.6}$$

In Eq. 9.6, the term $P(position)$ is the prior, based on the probability of a position at the previous time-point and on the model of the animal's movement. The terms $P(set\ of\ spikes|position)$ and $P(set\ of\ spikes)$ are both produced by multiplying together the corresponding probabilities for each neuron.

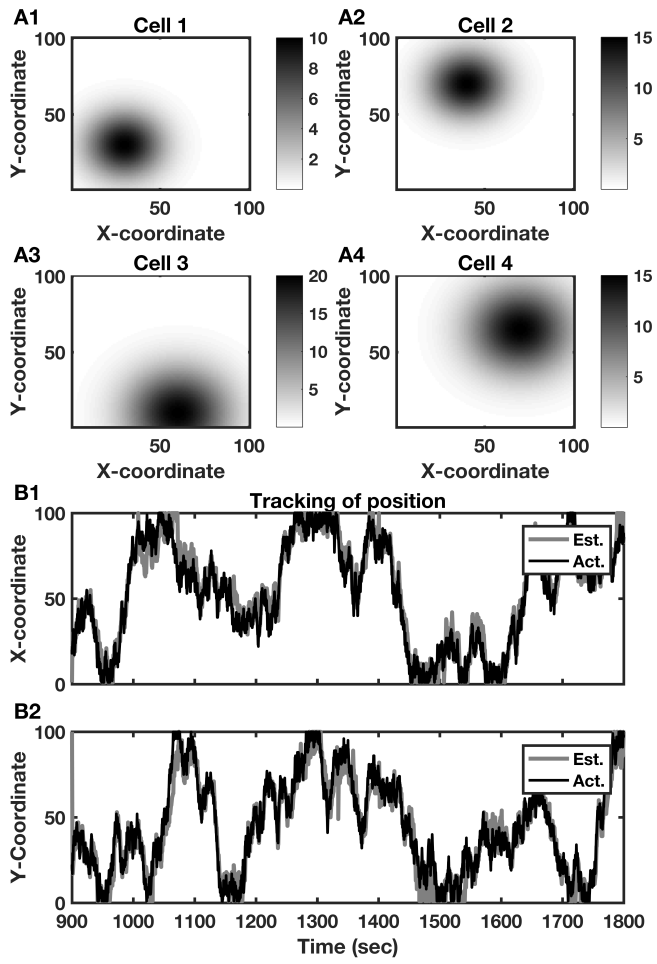

Figure 9.9. Decoding of position using the spikes produces by place-cells. A1-A4. Circular place fields of simulated cells with the neurons' firing rates peaked at a particular location (higher firing rate is indicated by the darker shading—see color bars). **B1-B2.** An adaptive decoder allows spikes from the four cells represented in A) to update *estimates* of the position (gray curves, Est.), tracking the *actual* position (black curves, Act.) with a high degree of accuracy. Indeed, the error between estimates and actual position is generally a lot lower than the standard deviations of the mostly non-overlapping place fields shown in A. **B1.** Estimate and actual value of the x-coordinate as a function of time. **B2.** Estimate and actual value of the y-coordinate as a function of time. The second half of a trial was used for decoding (Figure 9.8B), while the first half was used for estimation of the place fields (Figure 9.8A). This figure was produced by the online code `place_decoding.m`. If you run the code, you will see the two-dimensional probability distribution for the estimate of position evolve over time.

In the online code (`place_decoding.m`) used to produce Figure 9.9, we simulated a random walk for the animal's movement in Stage 1. We can then use the random-walk assumption in the model needed to update the prior on the animal's position across time-steps. For example, if we knew the animal's location precisely at a previous time-point, then, without any information from neural activity, we would expect the animal to be within a small distance of that prior position, with equal likelihood in any direction. Therefore, without information from spikes, the estimate of position gets less certain—the peak of high probability spreads out—as time goes on. In Stage 2 of the code, therefore, with possible positions given as grid-squares, we take the probability distribution of the previous time-point and allocate a fraction of the probability on each grid-square equally to neighboring grid-squares. In this manner, we update the position estimate in terms of the animal's movement.

Alternatively, in a more sophisticated method, one could select specific directions based on the prior movement of the animal and the observed tendency to keep going in the

same direction. In general, any observed regularity in motion can be incorporated by modifying the update from the previous probability distribution when producing the prior. Similarly, any preference for certain positions in the environment can be incorporated by multiplying by an additional prior proportional to the relative preferences for each position. In general, any prior information can be translated into a probability distribution, which multiplies any current estimate of position. By incorporating more information, such priors tighten the estimate of position, but they could lead to errors, in particular if a probability of zero is given to any location. For example, one might be tempted to provide a probability of zero to any location where the animal has not been observed, but this would be an error—just because the animal has not done something in the past does not make it impossible in the future, yet a probability of zero implies impossibility.

Questions for Chapter 9.

1) In a group of 10 neurons recorded during a task, the firing rates of 6 of them vary in proportion with each other, while the firing rates of the other 4 decrease from 50Hz in proportion with each other. The mean rate of the group of 4 is 50 minus 1.5 times the mean rate of the group of 6. How many principal components are needed to explain the full variance of the set of 10 neurons?

2) When two neurons are recorded during a binary decision-making task, the first principal component is their mean firing rate.

a) What is the second principal component?

b) What would it mean about their tuning if the first principal component of activity is most correlated with the decision?

c) What would it mean about their tuning if the second principal component of activity is most correlated with the decision?

3) a) You observe a 1-second spike train produced by an inhomogeneous Poisson process with rate of 4Hz in the first half-second then 8Hz in the second half-second. What is the probability you see more spikes in the first half-second than the second half-second?

b) What is the probability that a homogeneous 4Hz Poisson process, observed for a one-second interval, produces at least twice as many spikes in one half of the interval compared to the other half?

4) Two neurons with place fields of identical shape but with different centers, one with a peak rate of 20Hz, the other with a peak rate of 10Hz, each emit two spikes in a small, 50ms, time-window. Explain as quantitatively as you can, where the decoder of Section 9.7 would place the most likely location of the animal.

9.8. Appendix A. How PCA works: Choosing a direction to maximize the variance of the projected data

The first "principal component" is the direction maximizing the variance of the data when the data are projected onto that direction. To understand this section, a bit of linear algebra is needed, along with the geometric idea of choosing a basis—choosing a new set of axes in terms of the original axes.

Here we consider the original "basis" or set of directions to be given by $\boldsymbol{x}$, a vector in P dimensions. The original set contains N data points $\{\boldsymbol{x}(1), \boldsymbol{x}(2), \boldsymbol{x}(3), \cdots \boldsymbol{x}(N)\}$. In our examples, the N data points correspond to the N values of time at which firing rates are calculated. Then at each time point the vector, $\boldsymbol{x}(t)$ has P components corresponding to P cells. For example, $\boldsymbol{x}(1) = [x_1(1), x_2(1), x_3(1), \cdots x_P(1)]$, where the (1) indicates the first time-point and the subscripts indicate the cell labels.

We will generate a new basis, $\boldsymbol{u}$, which will also have P components, where each component in the new basis is a linear combination of components of the old basis,

$$u_i = \sum_{j=1}^{P} M_{ij} x_j. \qquad \text{Eq. 9.7}$$

In Eq. 9.7, the matrix, M, simply tells us how each of the new directions in $\boldsymbol{u}$ depends on the old directions in $\boldsymbol{x}$. In particular, the principal component (which we desire to be the direction with most variance) is given by:

$$u_1 = \sum_{j=1}^{P} M_{1j} x_j .$$

The matrix, M, is an orthonormal matrix, meaning each row is a vector of unit length at right-angles to all others. The sum over the index j means each entry in a row of the matrix, M, is multiplied by a corresponding entry in the column vector $\boldsymbol{x}$, then all products are summed together—*i. e.*, the notation describes matrix multiplication (see Section 1.4.2).

To proceed (and to make things easier) we will assume that we have already subtracted the mean of each component to produce our data, so each data point has the mean rate of each neuron subtracted. Thus, the mean of all the data points is now zero. Our new basis will be a new set of orthogonal axes going through the same origin, so the mean of the data points in the new basis will also be zero. The variance in one direction such as the principal component is given simply by its mean-square value now. Therefore, we wish to choose the first principal component, $u_1 = \sum_j M_{1j} x_j$ to maximize the sum of the squares of the data points projected onto its axis:

$$\begin{aligned}
\left[\sum_{n=1}^{N} u_1(n)\right]^2 &= \left[\sum_{n=1}^{N}\sum_{j=1}^{P} M_{1j} x_j(n)\right]^2 \\
&= \left[\sum_{m=1}^{N}\sum_{k=1}^{P} M_{1k} x_k(m)\right]\left[\sum_{n=1}^{N}\sum_{j=1}^{P} M_{1j} x_j(n)\right] \\
&= \sum_{j=1}^{P}\sum_{k=1}^{P} M_{1k} M_{1j} \sum_{n=1}^{N}\sum_{m=1}^{N} x_k(m) x_j(n) \\
&= \sum_{j=1}^{P}\sum_{k=1}^{P} M_{1k} M_{1j} C_{kj}
\end{aligned} \qquad \text{Eq. 9.8}$$

where C is the covariance matrix of the original data—a symmetric matrix ($C_{kj} = C_{jk}$) which would be a diagonal matrix (with individual variances on the diagonal) if all neurons had zero correlation with each other.

In order to show that the direction of maximum variance is an eigenvector, we can use the method of Lagrange multipliers outlined below. Since eigenvectors must be normalized, there is a constraint:

$$\sum_{k=1}^{P} (M_{1k})^2 = 1. \qquad \text{Eq. 9.9}$$

To find a maximum of the variance (which we achieve by altering any/all of the coefficients, M_{1k}) we want the derivative of the following to be zero:

$$\sum_{j=1}^{P}\sum_{k=1}^{P} M_{1k} M_{1j} C_{kj} + \lambda\left[1 - \sum_{k=1}^{P} (M_{1k})^2\right] \qquad \text{Eq. 9.10}$$

with respect to variation in a coefficient. If you have not seen Lagrange multipliers, you can think of the last term, with the added Lagrange multiplied, λ, as a trick, which adds "zero" to the first term. Rather, the Lagrange term incorporates the constraint of Eq. 9.9, in a manner

that is satisfied when Eq. 9.10 is maximized. Taking the derivative to be zero with respect to M_{1l} (to find the value of the l-th coefficient) we have:

$$\delta_{jl}\sum_{k=1}^{P} M_{1k}C_{kj} + \delta_{kl}\sum_{j=1}^{P} M_{1j}C_{kj} - 2\lambda M_{1k}\delta_{kl} = 0 \qquad \text{Eq. 9.11}$$

or

$$2\lambda M_{1l} = \sum_{k=1}^{P} M_{1k}C_{kl} + \sum_{j=1}^{P} M_{1j}C_{lj} = 2\sum_{j=1}^{P} C_{lj}M_{1j}, \qquad \text{Eq. 9.12}$$

where we have used the symmetry of the covariance matrix ($C_{lj} = C_{jl}$). Thus, given the definitions of eigenvectors and eigenvalues (Figure 9.4), equation 9.12 shows that the vector of components, M_{1l}, is an eigenvector of the covariance matrix, C, with eigenvalue λ. A similar rearrangement shows that the variance of the data (Eq. 9.8) is simply proportional to the eigenvalue, λ. Therefore, the direction with highest variance is the eigenvector of the covariance matrix with highest eigenvalue.

9.8.1. Carrying out PCA without a built-in function

The above description of principal component analysis indicates how principal components can be extracted using standard mathematical function, if your software lacks a built-in function for PCA.

Descriptively, the process is as follows:

1) Center the original data by subtracting the mean in each dimension (*i. e.* for each cell subtract its mean firing rate from every data point so we are recording deviations from the mean). Record those mean values to later recreate neural firing rates. These values comprise the vector "`MU`" in Tutorial 9.1.

2) Calculate the covariance matrix (this can be done without step-1 since the covariance is mean subtracted). This will be a $P \times P$ matrix if there are P neurons, and is matrix C in the above section.

3) Find the eigenvectors of the covariance matrix with their eigenvalues (Figure 9.4)—nearly all software packages have standard functions for this.

4) Sort, then order the eigenvectors from the one with the largest eigenvalue to the smallest (all eigenvalues are non-negative real numbers since the covariance matrix is real and symmetric). The vector of sorted eigenvalues is the same as "`LATENT`" in Tutorial 9.1, and if divided by their sum and multiplied by 100 it is the same as "`EXPLAINED`" in Tutorial 9.1. The matrix of eigenvectors, after ordering, is equivalent to the matrix "`COEFF`" in Tutorial 9.1.

5) To obtain the projection of the mean-subtracted original data on the principal components, multiply the matrix of centered original data from 1) by the eigenvectors obtained in 4). The resulting matrix of projections is equivalent to the matrix "`SCORE`" in Tutorial 9.1.

The eigenvectors are now ordered in terms of the principal components. The eigenvectors are orthonormal when created (perpendicular and of unit length) so you will have generated a new set of axes for the data. The axes are ordered according to the directions with highest to lowest variance in the original data.

Each principal component corresponds to a new axis in the high-dimensional space, but whether one direction along that axis is positive or negative is arbitrary. Therefore, the results of different methods for evaluating principal components can give rise to eigenvectors (columns of "COEFF") and the corresponding projections on those eigenvectors (rows of "SCORE"), which differ together by a sign-flip.

A code that can produce these results follows:

```
% First store the mean of the firing rates
mean_rates = mean(rate);      % same as "MU"
% Use dev_rate as the deviation from mean rate for each cell
dev_rate = rate - ones(Nt,1)*mean_rates;

% Next calculate the covariance matrix of firing rates
C_matrix = cov(rate);

% Finds eigenvectors as columns then eigenvalues in a diagonal matrix
[eig_vectors Diag_evals] = eig(C_matrix);

% Form a column of eigenvalues
eig_vals = diag(Diag_evals);

% Sort the eigenvalues into descending order
[ordered_eig_vals, new_indices] = sort(eig_vals,'descend');

% Divide by sum of eigenvalues for fraction of variance explained
var_explained = ordered_eig_vals/sum(ordered_eig_vals); % "EXPLAINED"

% Sort the eigenvectors according to variance explained
new_basis = eig_vectors(:,new_indices); % new_basis is like "COEFF"

% Find the projection of the mean-subtracted data on PCs.
PC_rates = dev_rate*new_basis;          % PC_rates is like "SCORE"
```

9.9. Appendix B. Change-point detection for a Poisson process.

9.9.1. Optimal rate

Our first goal is to show that the best estimate for the unknown firing rate, $r^{(opt)}$, of a Poisson process producing N spikes in time T is $r^{(opt)} = N/T$, if we assume *a priori* that all firing rates are equally likely (*i. e.*, a uniform prior). The assumption of uniform prior means our estimate is the Maximum Likelihood Estimate (MLE).

With a uniform prior, Bayes' Theorem (Eq. 1.32) tells us that:

$$P(r|N,T) \propto P(N|r,T), \qquad \text{Eq. 9.13}$$

so we just need to find which firing rate is most likely to yield the observed number of spikes. For this calculation, we use the Poisson formula (Section 3.3.3),

$$P(N|r,T) = \frac{(rT)^N e^{-rT}}{N!}. \qquad \text{Eq. 9.14}$$

The probability approaches zero as r approaches either zero or infinity and is always greater than zero between these values, so has a maximum. Calculus tells us the maximum is the point at which the rate of change of this probability with respect to r is zero, that is $r^{(opt)}$ is the value of r at which

$$\frac{dP(N)}{dr} = 0. \qquad \text{Eq. 9.15}$$

If we cancel out the denominator, $N!$ (which is independent of r), the maximum requires:

$$N\left(r^{(opt)}T\right)^{N-1} e^{-rT} - \left(r^{(opt)}T\right)^N e^{-rT}/T = 0 \qquad \text{Eq. 9.16}$$

which simplifies to

$$r^{(opt)} = N/T. \qquad \text{Eq. 9.17}$$

9.9.2. Evaluating the change-point, Method 1

We divide an interval of length T into two sub-intervals, one of length T_1, the other of length $T_2 = T - T_1$. To test whether the time-point, T_1, is the most likely time-point for a change in rate, we treat the rates before and after T_1 as the optimal values, $r_1 = N_1/T_1$ and $r_2 = N_2/T_2$ respectively, where N_1 is the number of spikes in the interval $0 < t \leq T_1$ and N_2 is the number of spikes in the interval $T_1 < t \leq T$.

We can then divide each subinterval into time-bins of width δt. For each time-bin, find the probability of the observed spike or lack of spike given the rate of the process, then multiply together these individual probabilities to obtain the probability of the entire sequence of spikes. There are $N_{T_1} = T_1/\delta t$ such bins in the first sub-interval and $N_{T_2} = T_2/\delta t$ such bins in the second interval. The probability of the entire sequence, assuming optimal rates, becomes then:

$$P(T_1, T_2|N_1, N_2) = (r_1\delta t)^{N_1}(1 - r_1\delta t)^{N_{T_1} - N_1}(r_2\delta t)^{N_2}(1 - r_2\delta t)^{N_{T_2} - N_2}. \qquad \text{Eq. 9.18}$$

To evaluate Eq. 9.18 in the limit $\delta t \to 0$, it is easier to take the logarithm before expanding in small δt, so:

$$\begin{aligned} \ln[P(T_1, T_2|N_1, N_2)] &= N_1 \ln(r_1\delta t) + \left(N_{T_1} - N_1\right)\ln(1 - r_1\delta t) \\ &\quad + N_2 \ln(r_2\delta t) + \left(N_{T_2} - N_2\right)\ln(1 - r_2\delta t) \\ &\cong N_1 \ln(r_1\delta t) - \left(N_{T_1} - N_1\right) r_1\delta t + N_2 \ln(r_2\delta t) \\ &\quad - \left(N_{T_2} - N_2\right) r_2\delta t \\ &\cong N_1 \ln(r_1) + N_2 \ln(r_2) + N \ln(\delta t) - T_1 r_1 - T_2 r_2 \\ &= N_1 \ln(r_1) + N_2 \ln(r_2) + N \ln(\delta t) - N. \end{aligned} \qquad \text{Eq. 9.19}$$

Eq. 9.19 leads to

$$P(T_1, T_2|N_1, N_2) \cong {r_1}^{N_1} {r_2}^{N_2}\, \delta t^N e^{-N}. \qquad \text{Eq. 9.20}$$

Removing the last two terms that do not depend on the separation into subintervals, we find

$$P(T_1, T_2|N_1, N_2) \propto {r_1}^{N_1} {r_2}^{N_2} = \left(\frac{N_1}{T_1}\right)^{N_1} \left(\frac{N_2}{T_2}\right)^{N_2}. \qquad \text{Eq. 9.21}$$

Computationally we can vary T_1, count the spikes in the two sub-intervals and evaluate the probability of this being the change-point according to the above formula. The value of T_1 that maximizes $P(T_1, T_2|N_1, N_2)$ is selected as the change-point.

9.9.3. Evaluating the change-point, Method 2.

In the second method, we will evaluate the ratio of two probabilities: the probability of the total numbers of spikes on either side of a proposed change-point being produced by two distinct Poisson processes of different rates, divided by the probability of those spike counts being produced by a single Poisson process of constant rate.

In this case, we use the Poisson formula (Secton 3.7.1)

$$P(N|rT) = \frac{(rT)^N e^{-rT}}{N!}, \qquad \text{Eq. 9.22}$$

for the probability of N spikes in an interval of length T, with fixed rate, r.

For the two intervals of length T_1 and $T_2 = T - T_1$, this formula yields for the probability of N_1 spikes in a period of T_1 *and* the probability of N_2 spikes in a period of T_2 as:

$$P(N_1, N_2|r_1,\ T_1, r_2, T_2) = \frac{(r_1 T_1)^{N_1} e^{-r_1 T_1}}{N_1!} \cdot \frac{(r_2 T_2)^{N_2} e^{-r_2 T_2}}{N_2!}. \qquad \text{Eq. 9.23}$$

We again use Bayes' Theorem with a uniform prior to obtain the Maximum Likelihood Estimate of rates as $r_1 = N_1/T_1$ and $r_2 = N_2/T_2$ from Eq. 9.17.

We compare with the case where the entire time-interval has no change in firing rate, r, in which case we use Eq. 9.23 with $r_1 = r$ and $r_2 = r$:

$$P(N_1, N_2|r, T_1, r, T_2) = \frac{(rT_1)^{N_1} e^{-rT_1}}{N_1!} \cdot \frac{(rT_2)^{N_2} e^{-rT_2}}{N_2!}. \qquad \text{Eq. 9.24}$$

with $r = N/T$, where $N = N_1 + N_2$.

The ratio of Eq. 9.23 to Eq. 9.24 gives us the probability of a change-point at T_1 as a likelihood ratio in comparison to the probability of no rate-change in the entire period:

$$\frac{P(N_1, N_2|r_1,\ T_1, r_2, T_2)}{P(N_1, N_2|r, T_1, r, T_2)} = \left(\frac{N_1}{T_1}\right)^{N_1} \left(\frac{N_2}{T_2}\right)^{N_2} \Big/ \left(\frac{N}{T}\right)^{N}. \qquad \text{Eq. 9.25}$$

The advantage of Eq. 9.25, is it can allow us to decide whether to introduce a change-point or not based on a criterion for the value of the likelihood ratio. Similarly, the likelihood can be recalculated for the subintervals once a change-point is found, to assess whether they should be further subdivided with additional change-points. In this manner, as with Hidden Markov modeling (Section 9.5), the total number of states and transitions between them, can be produced as a result of the analysis, rather than prescribed beforehand.

References for Chapter 9

1. Rigotti M, Barak O, Warden MR, et al. The importance of mixed selectivity in complex cognitive tasks. *Nature.* 2013;497(7451):585-590.
2. Murray JD, Bernacchia A, Roy NA, Constantinidis C, Romo R, Wang XJ. Stable population coding for working memory coexists with heterogeneous neural dynamics in prefrontal cortex. *Proc Natl Acad Sci U S A.* 2017;114(2):394-399.
3. Jun JK, Miller P, Hernandez A, et al. Heterogenous population coding of a short-term memory and decision task. *The Journal of neuroscience : the official journal of the Society for Neuroscience.* 2010;30(3):916-929.
4. Vargas-Irwin CE, Shakhnarovich G, Yadollahpour P, Mislow JM, Black MJ, Donoghue JP. Decoding complete reach and grasp actions from local primary motor cortex populations. *J Neurosci.* 2010;30(29):9659-9669.
5. Mante V, Sussillo D, Shenoy KV, Newsome WT. Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature.* 2013;503(7474):78-84.
6. Morcos AS, Harvey CD. History-dependent variability in population dynamics during evidence accumulation in cortex. *Nat Neurosci.* 2016;19(12):1672-1681.
7. Afshar A, Santhanam G, Yu BM, Ryu SI, Sahani M, Shenoy KV. Single-trial neural correlates of arm movement preparation. *Neuron.* 2011;71(3):555-564.
8. Fusi S, Miller EK, Rigotti M. Why neurons mix: high dimensionality for higher cognition. *Curr Opin Neurobiol.* 2016;37:66-74.
9. Lehky SR, Tanaka K. Neural representation for object recognition in inferotemporal cortex. *Curr Opin Neurobiol.* 2016;37:23-35.
10. Sussillo D, Abbott LF. Generating coherent patterns of activity from chaotic neural networks. *Neuron.* 2009;63(4):544-557.
11. Laje R, Buonomano DV. Robust timing and motor patterns by taming chaos in recurrent neural networks. *Nat Neurosci.* 2013;16(7):925-933.
12. Miller P. Stimulus number, duration and intensity encoding in randomly connected attractor networks with synaptic depression. *Front Comput Neurosci.* 2013;7:59.
13. Rigotti M, Ben Dayan Rubin D, Morrison SE, Salzman CD, Fusi S. Attractor concretion as a mechanism for the formation of context representations. *Neuroimage.* 2010;52(3):833-847.
14. Rajan K, Harvey CD, Tank DW. Recurrent Network Models of Sequence Generation and Memory. *Neuron.* 2016.
15. Barak O. Recurrent neural networks as versatile tools of neuroscience research. *Curr Opin Neurobiol.* 2017;46:1-6.
16. Chandrasekaran C. Computational principles and models of multisensory integration. *Curr Opin Neurobiol.* 2017;43:25-34.
17. Buzsaki G. Large-scale recording of neuronal ensembles. *Nat Neurosci.* 2004;7:446-451.
18. Churchland MM, Yu BM, Sahani M, Shenoy KV. Techniques for extracting single-trial activity patterns from large-scale neural recordings. *Curr Opin Neurobiol.* 2007;17(5):609-618.
19. Shenoy KV, Sahani M, Churchland MM. Cortical control of arm movements: a dynamical systems perspective. *Annu Rev Neurosci.* 2013;36:337-359.

20. Ganguli S, Sompolinsky H. Compressed sensing, sparsity, and dimensionality in neuronal information processing and data analysis. *Annu Rev Neurosci.* 2012;35:485-508.
21. Maass W, Natschlager T, Markram H. Real-time computing without stable states: a new framework for neural computation based on perturbations. *Neural Comput.* 2002;14(11):2531-2560.
22. Mazor O, Laurent G. Transient Dynamics versus Fixed Points in Odor Representations by Locust Antennal Lobe Projection Neurons. *Neuron.* 2005;48(4):661-673.
23. Franke F, Jackel D, Dragas J, et al. High-density microelectrode array recordings and real-time spike sorting for closed-loop experiments: an emerging technology to study neural plasticity. *Front Neural Circuits.* 2012;6:105.
24. Grienberger C, Konnerth A. Imaging calcium in neurons. *Neuron.* 2012;73(5):862-885.
25. Stirman JN, Smith IT, Kudenov MW, Smith SL. Wide field-of-view, multi-region, two-photon imaging of neuronal activity in the mammalian brain. *Nat Biotechnol.* 2016;34(8):857-862.
26. Ziv Y, Burns LD, Cocker ED, et al. Long-term dynamics of CA1 hippocampal place codes. *Nat Neurosci.* 2013;16(3):264-266.
27. Welch LR. Hidden Markov models and the Baum-Welch algorithm. *IEEE Information Theory Society Newsletter.* 2003;53(4).
28. Baum LE, Petrie T, Soules G, Weiss N. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Ann Math Stat.* 1970; 41:164-171.
29. Abeles M, Bergman H, Gat I, et al. Cortical activity flips among quasi-stationary states. *Proc Natl Acad Sci U S A.* 1995;92(19):8616-8620.
30. Seidemann E, Meilijson I, Abeles M, Bergman H, Vaadia E. Simultaneously recorded single units in the frontal cortex go through sequences of discrete and stable states in monkeys performing a delayed localization task. *J Neurosci.* 1996;16(2):752-768.
31. Jones LM, Fontanini A, Sadacca BF, Miller P, Katz DB. Natural stimuli evoke dynamic sequences of states in sensory cortical ensembles. *Proc Natl Acad Sci U S A.* 2007;104(47):18772-18777.
32. Camproux A-C, Saunier F, Chouvet G, Thalabard J-C, Thomas G. A Hidden Markov Model Approach to Neuron Firing Patterns. *Biophysical Journal.* 1996;71:2404-2412.
33. Miller P, Katz DB. Stochastic transitions between neural states in taste processing and decision-making. *J Neurosci.* 2010;30(7):2559-2570.
34. Miller P, Katz DB. Stochastic Transitions between States of Neural Activity. In: Ding M, Glanzman DL, eds. *The Dynamic Brain: An Exploration of Neuronal Variability and Its Functional Significance*. New York, NY: Oxford University Press; 2011:29-46.
35. Brown EN, Frank LM, Tang D, Quirk MC, Wilson MA. A statistical paradigm for neural spike train decoding applied to position prediction from ensemble firing patterns of rat hippocampal place cells. *J Neurosci.* 1998;18(18):7411-7425.
36. Eden UT, Frank LM, Barbieri R, Solo V, Brown EN. Dynamic analysis of neural encoding by point process adaptive filtering. *Neural Comput.* 2004;16(5):971-998.

37. Truccolo W, Eden UT, Fellows MR, Donoghue JP, Brown EN. A point process framework for relating neural spiking activity to spiking history, neural ensemble, and extrinsic covariate effects. *J Neurophysiol.* 2005;93(2):1074-1089.
38. Deneve S, Duhamel JR, Pouget A. Optimal sensorimotor integration in recurrent cortical networks: a neural implementation of Kalman filters. *J Neurosci.* 2007;27(21):5744-5756.
39. Faragher R. Understanding of the basis of the Kalman filter via a simple and intuitive derivation. *IEEE Signal Processing Magazine.* 2012;29(5):128-132.