# 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 more dramatic—and 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 response 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 9.4) encoded time. Whereas time is clearly encoded in the network—the network activity at any point in time can produce a desired change in Purkinje cell activity if so-trained—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. 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—not because they are less important for brain function.

The first step to 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. In this chapter, we consider three such methods of analysis.

## Principal Components Analysis (PCA)

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

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, but here we will focus primarily on neural firing rates. In this context, the individual dimensions of the initial data

correspond to the firing rates of individual neurons, so the number of dimensions is equal to the number of neurons.

First though, it is worth noting that to isolate the spikes of individual neurons from multi-electrode data, it is typical to run PCA on the voltage traces of the electrodes. In this latter context, the separate dimensions correspond to the values of voltage measured on separate electrodes, so the number of dimensions is the number of voltage traces. It can be possible to extract separate "spike signatures" and identify more neurons than electrodes used—for example, with two electrodes the peak voltage deflection during a spike from four different neurons may be separable as "high on electrode 1, low on electrode 2", or "low on electrode 1, high on electrode 2", or "high on both", or "low on both".

PCA is helpful if there are correlations between the dimensions—in multi-electrode recordings of electrical activity, any source, such as a neuron, that affects the voltage in more than one electrode produces a correlation between the values recorded in the different electrodes. In firing-rate data, correlations between the activities of different cells can be caused by connections between the cells within a network, or by common "upstream" input. In each case, 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 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".

The idea of choosing a basis in geometry—which means choosing a new set of coordinates—underlies PCA. The new basis is chosen based on the variance of the data.

When analyzing neural activity, the initial, high number of dimensions is typically the number of neurons, with each dimension being 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 and allow us to define new axes, each new axis corresponding to the coordinated activity of a particularly weighted set of neurons that defines a dimension of high variance. Once the activity is reduced to the principal components—that is, its projection onto these new axes with high variance—it can be analyzed with a particular model in mind, or simply be made clearly visible in a graph.

The first step of PCA is to extract a new set of orthogonal axes sorted by the amount of variance of the data in their directions. Figure 9.1 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 approximately 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 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 9.1B 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) then little change in their position would ensue.

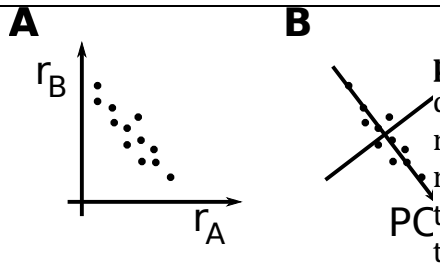


**Figure 9.1. Selecting new axes with principal components analysis. A)** Each data point corresponds to a specific measurement and comprises a pair of firing rates, $r_A$ and $r_B$. **B)** The axis corresponding to the first principal component, $PC_1$, contains the most variance in the dataset. The second axis, $PC_2$, is orthogonal to the first.

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 Frisbee. Such data would be closer to 2D and PCA would pick out two directions at right angles on the face of the Frisbee, which correspond to the greatest spread in the data. 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 Frisbee) could most likely be ignored with little loss of information about 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.

One common method for producing a set of data points like that shown in Figure 9.1 is to measure the neurons' combined response to each of many distinct stimuli. In this case projection of the data onto $PC_1$ would be the best way to distinguish the stimuli using a single scalar value. Alternatively, the data points could each correspond to the firing rates in distinct time-bins when the neurons respond to a single stimulus. In this case, it is more common to join the points together to produce a trajectory. We will use the latter method in Tutorial 9.1 in an attempt to uncover the underlying "true" trajectory as a combination of neural firing rates when each neuron's rate is affected by uncorrelated noise.

*How PCA works: Descriptive version*

1) Center the original data by subtracting the mean in each dimension (*i. e.* for every data point, subtract the mean firing rate of the corresponding neuron, so we are recording deviations from the mean).
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.
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. Many coding languages can produce eigenvectors and eigenvalues of matrices using a single command—it is "`eig`" in Matlab.)

4) Sort, then order, the eigenvectors from the one with the largest eigenvalue to the smallest (all eigenvalues should be positive since the covariance matrix should be real and symmetric).

The eigenvectors are now ordered in terms of the principal components. The eigenvectors are orthogonal when created so we 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.

**Tutorial 9.1. Principal components 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 firing rates for each cell that are less noisy (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 $I_A = A \cdot \sin(2\pi f t)$ and $I_B = B \cdot \cos(2\pi f t)$, 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 row in the matrix, with columns representing the time-points. 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 for each neuron from the Normal distribution with unit standard deviation and zero mean (defined as $N(0,1)$ via `randn()` in Matlab). $\sigma$=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 transpose of the rate matrix, using (in Matlab) the command:

```
[COEFF,SCORE,LATENT,TSQUARED,EXPLAINED,MU] = pca(rate');
```

(Use "`help pca`" to find the meanings of the outputs).

d) Plot the first column of coefficients, `COEFF`, (corresponding 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 the second principal component) against the vector of input weights $W_i^{(B)}$. Does the sign of the gradient 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 `COEFF` by the first two rows of `SCORE'` (the transpose of `SCORE`) and adding to all entries of each row produced in this manner, the value of the corresponding entry of `MU` (the mean for that neuron).

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 rows 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 $I_A = A \cdot \sin(2\pi f_A t)$ and $I_B = B \cdot \cos(2\pi f_B t)$, 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, $I_A = A \cdot t$ (from $t = 0$ to 10s), and the other as a transient signal during the ramp, $I_B = B \cdot \sin(2\pi f t)$ for $4\text{s} < t < 5\text{s}$, otherwise $I_B = 0$.

**Single-trial versus trial-averaged analyses**

Analyses of population data based on the firing rates of neurons, implicitly assume that many trials of fundamentally identical data have been averaged over (using the methods of Chapter 2) in order to produce a firing-rate from the series of instantaneous spikes. When averaging across trials, the point of alignment matters if the underlying network activity is not fundamentally identical on different trials. For example, in the model of a decision-making network (Tutorial 5.2) 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) and General Linearized Methods (GLM).

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 tetrode arrays, allow for the extracellular recording of spikes from many neurons, even hundreds, simultaneously. Optical imaging, most commonly using calcium-responsive dyes, but now also using voltage-responsive dyes, allow for the activity of an even greater number of cells, perhaps thousands, to be observed almost simultaneously (almost because there is typically a need for scanning the image line by line). Furthermore, the non-invasive methods most commonly used with humans, acquire multiple data streams simultaneously. Tens of electrodes are commonplace when producing an electroencephalogram (EEG) and millions of voxels (volume elements) can be measured via functional magnetic resonance imaging (fMRI). 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.

**Change-point detection**

As a prelude to HMM, we consider the problem of a single, noisy spike-train, whose underlying firing rate changes at some 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 (Chapter 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 know there is such a change. To proceed, we assume spikes are produced with Poisson statistics at unknown underlying rates.

If we split a spike-train of $N$ spikes and duration $T$ into two intervals, the first interval with $N_1$ spikes and duration $T_1$, the second interval has $N_2 = N - N_1$ spikes and duration $T_2 = T - T_1$. 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 $r_2 = N_2/T_2$, which is what one might expect—if we count a given number of spikes in a given time interval our best guess at the rate of the underlying process is the mean rate observed.

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 arising. Some analysis (Appendix B) shows that this product is proportional to:

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

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:

$$(r_1)^{N_1}(r_2)^{N_2} = (N_1/T_1)^{N_1}(N_2/T_2)^{N_2}$$

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 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

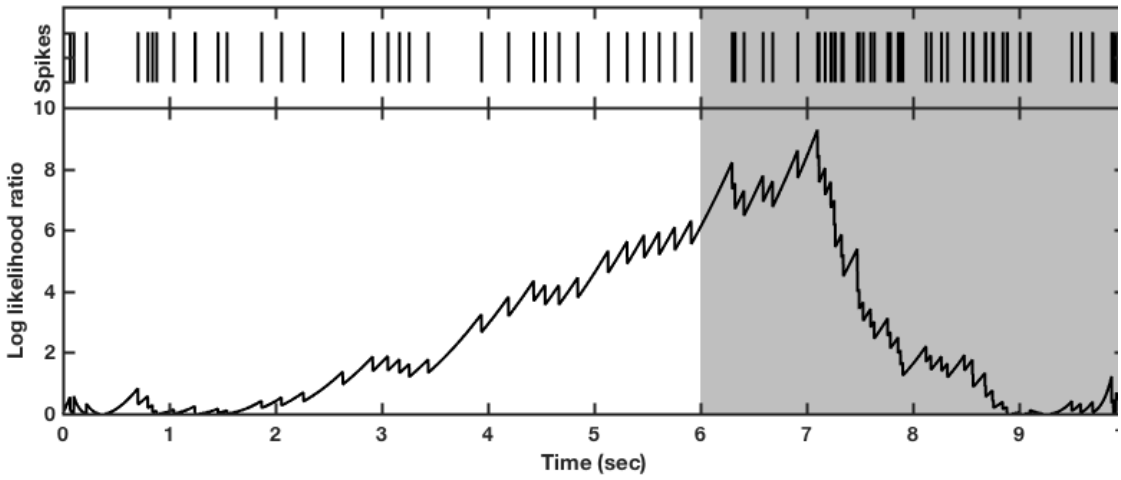


**Figure 9.2. Detecting a change in firing rate. Top** Each vertical line indicates the time of a spike generated by a Poisson process whose rate changes from 5Hz (white region) to 10Hz (gray region) at a time of 6 seconds (the change-point). **Bottom** The log likelihood ratio is the log of the probability of the spikes being produced by a Poisson process whose rate changes at that given point of time divided by the probability 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.

occur at any point in the time-window prior to observation). Given those assumptions of equal priors, 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.

We can take the calculation one step further and compare the likelihood of the change-point at a given time, $T_1$, to the likelihood of the spikes arising at a fixed rate. This requires a division by $(N/T)^N$ and produces a likelihood ratio (Figure 9.2). Simulations suggest (see Tutorial 7.2) 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. 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.

*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"—the corresponding numbers are greater than those able to be stored given the software's standard memory allocation per number. 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.

## Hidden Markov modeling (HMM)

Hidden Markov modeling is, conceptually, an extension of change-point detection to a system in which the firing rates of multiple neurons change at the same times. The change-point then marks a transition between two states, with each state defined by the set of firing rates of an ensemble of neurons. When multiple neurons change their firing rates coherently, the change-point is more easily detected than it would be from a single neuron's spike train.

A Markov process is one in which the probability of an event occurring—such as a particular neuron emitting a spike—depends only on the current state of the system, so is independent of past history except in so far as that history led to the current state. In that sense Markov processes can be called "memory-less", but it is important to realize that memory-dependent processes can be formulated as Markov processes by including those variables that are memory-dependent in the formulation of a state.

A Hidden Markov model is so-called because there are variables that are not directly observed, namely hidden variables. In the change-point detection algorithm of the previous section the underlying firing rate of the Poisson process is a hidden variable, because it is not directly observable and can only be inferred from the emission of spikes. Indeed, we

could use the results of the change-point detection algorithm to produce a Hidden Markov model, because the algorithm assumes that the change-point could occur a priori with equal likelihood at any point in time—that is, independently of how long the initial firing rate had persisted. A Hidden Markov model of the process would simply convert the estimated transition time of the observed spike sequence into a probability per unit time that the neuron would transition from its initial rate to its final rate.

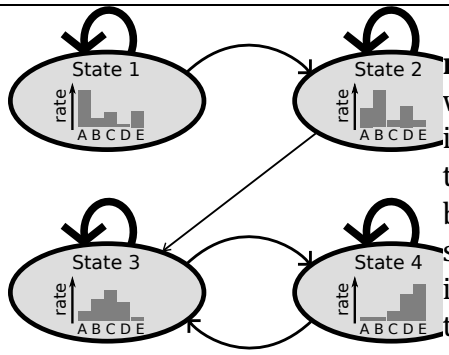


**Figure 9.3. Example of a Hidden Markov model.** A model with four states (ovals), as shown, would be defined by a 4x4 transition matrix, indicating the probability per time bin of a transition from one state to another (arrows between ovals). In neural activity models, the strongest transition is from a state to itself, indicating the persistence of a state across many time-bins. Each state is defined by the probability of spike emissions, proportional to the firing rate of each cell. In this example, rates of 5 neurons (labeled A-E) are depicted. The set of rates is unique to each state.

In general, Hidden Markov models are more complicated for two reasons. First, 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. Second, the number of possible states is usually greater than two, so several sets of firing rates coordinated across the measured neurons is common. Finally, transitions can be possible in a non-sequential order, with reverse transitions also possible (Figure 9.3).

The model is then defined by two matrices, the matrix of emission probabilities and the matrix of transition probabilities.

In the example shown in Figure 9.3, the emission probabilities would be contained in a 4x6 matrix like: $\begin{pmatrix} 0.20 & 0.03 & 0.05 & 0.01 & 0.06 & 0.65 \\ 0.10 & 0.20 & 0.03 & 0.10 & 0.03 & 0.56 \\ 0.05 & 0.12 & 0.15 & 0.10 & 0.02 & 0.56 \\ 0.01 & 0.01 & 0.04 & 0.14 & 0.20 & 0.60 \end{pmatrix}$, where each row corresponds to a particular state (1-4) and each column corresponds to a particular neuron (A-E) with the last column corresponding to the absence of a spike. The entries are the probability per time bin of a spike being produced by a given neuron (or no neuron) in the corresponding state. The first five values in each row are proportional to firing rate (the histograms of Figure 9.3) while the last column ensures that the rows sum to one. In this formulation, the possibility of multiple spikes per time-bin is excluded. More generally, combinations of multiple spikes could be included in a more extensive 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, 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.

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 (Chapter 6) 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. In this and similar examples, the underlying neural activity is best analyzed via HMM (Figure 9.3).


**Figure 9.4. 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 9.2). 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, like other standard analyses, which begin by averaging data across trials aligned to stimulus onset, would obscure evidence of these state transitions.

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

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 $T = 10\text{sec}$, $T_2 = T - T_1$, $N$ 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.

**Decoding position from multiple place fields**

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 of hours to days and are compact, peaking at a particular location in space and decreasing monotonically with distance away from the peak (Figure 9.4). 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. An important aspect of the approach used here is to incorporate the most recent estimate of the animal's position as a probability distribution to produce a prior—the probabilities of particular locations before any information from spikes in the next time-bin is incorporated. A second aspect of the approach is to include an absence of a spike as information in the same manner as the presence of a spike. These methods were pioneered by Emery Brown and Uri Eden and are similar to a method called Kalman filtering.

For a review of combining probabilities, see Chapter 0 Section Aiii. The first main principle involved in these methods is, when combining information from multiple sources,

the probability of an occurrence is calculated by multiplying together the contributions from the separate information sources. The second principle is 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. The 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.

When decoding position in this manner, two stages are required. In the first stage 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, it is necessary to 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 9.4, 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.

In the second stage, 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 from the probability of a spike (or no spike) when in that location. Use of Bayes' theorem 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 sum 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 by its probability distribution for location given a spike (when it spikes) or location given no spike (when it does not spike). Finally normalize the produce to ensure the resultant distribution sums to one.

In the code used to produce Figure 9.4, we simulate a random walk for the animal's movement, so the prior on the animal's position is calculated by taking the probability distribution of the previous time-bin and allocating a fraction of the probability at each grid-square equally to neighboring grid-squares. Alternatively, 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.

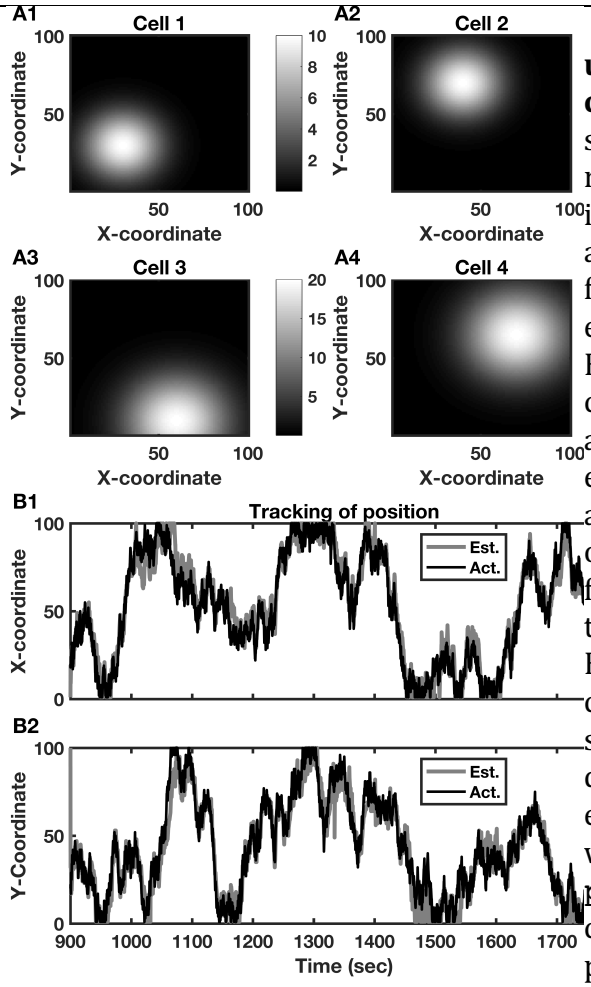


**Figure 9.5. Decoding of position using the spikes produces by place-cellls. A1-A4.** Circular place fields of simulated cells with the neuron's firing rate peaked at a particular location as indicated by the shading. **B1-B2.** An adaptive decoder allows spikes from the four cells represented in A) 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. **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, while the first half was used for estimation of the place fields. 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.

## 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 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 $\boldsymbol{x}$, with the original set of $N$ 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 if there are $P$ cells: $\boldsymbol{x}(1) = [x_1(1), x_2(1), x_3(1), \cdots x_P(1)]$.

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, $u_i = \sum_j M_{ij}x_j$. In this formula, 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: $u_1 = \sum_j M_{1j}x_j$. 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.

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 "zero" 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 so we wish to choose $u_1 = \sum_j M_{1j}x_j$ to maximize:

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

where $C$ is the covariance matrix of the original data—a symmetric matrix ($C_{kj} = C_{jk}$) which would be the identity matrix (ones 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 methods of Lagrangian multipliers outlined below. Since eigenvectors must be normalized, there is a constraint:

$$\sum_{k=1}^{P}(M_{1k})^2 = 1.$$

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:

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

with respect to variation in a coefficient. (If you have not seen Lagrangian multipliers, think of the last term as a trick, which adds "zero" to the first term because it is the constraint that must be satisfied. Taking the derivative to be zero with respect to $M_{1l}$ (to find the value of the $l$-th coefficient) we have:

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

or

$$2\lambda M_{1l} = \sum_{k=1}^{P} M_{1k}C_{kl} + \sum_{j=1}^{P} M_{1j}C_{lj} = 2\sum_{j=1}^{P} C_{lj}M_{1j}$$

where we have used the symmetry of the covariance matrix ($C_{lj} = C_{jl}$). Thus, given the definitions of eigenvectors and eigenvalues, we have proven that the vector of components, $M_{1l}$, is an eigenvector of the covariance matrix, $C$. A similar rearrangement shows that the variance is simply proportional to the eigenvalue (denoted by $\lambda$ above). Therefore, the direction with highest variance is the eigenvector of the covariance matrix with highest eigenvalue.

*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 means to later map back to neural firing rates. This is equivalent to "`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—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 should be positive since the covariance matrix should be real and symmetric). The vector of sorted eigenvalues, if divided by their sum and multiplied by 100 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 orthogonal when created 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.

Principal components correspond 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 of evaluating principal components can give rise to eigenvectors (columns of "COEFF") and the corresponding projections on those eigenvectors (rows of "SCORE") differing by a sign-flip.

A code that can produce these results follows:

```
% First store the mean of the firing rates
mean_rates = mean(rate,2);      % same as "MU"
% Use dev_rate as the deviation from mean rate for each cell
dev_rate = rate - mean_rates*ones(1,Nt);

% 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 = new_basis'*dev_rate;     % similar to transpose of
"SCORE"
```

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

*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 $r^{(opt)} = N/T$, assuming *a priori* that all firing rates are equally likely (*i. e.*, a uniform prior). With a uniform prior, Bayes' Theorem (Chapter 0, Section A.iii) tells us that:

$$P(r|N,T) \propto P(N|r,T),$$

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,

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

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

$$\frac{dP(N)}{dr} = 0.$$

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

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

which simplifies to

$$r^{(opt)} = N/T.$$

*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 $\delta t$, 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:

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

To evaluate this expression in the limit $\delta t \to 0$, it is easier to take the logarithm before expanding in small $\delta t$, so:

$$\begin{aligned}
\ln[P(T_1,T_2|N_1,N_2)] &= N_1\ln(r_1\delta t) + \left(N_{T_1} - N_1\right)\ln(1-r_1\delta t) \\
&\quad + N_2\ln(r_2\delta t) + \left(N_{T_2} - N_2\right)\ln(1-r_2\delta t) \\
&\cong N_1\ln(r_1\delta t) - \left(N_{T_1} - N_1\right)r_1\delta t + N_2\ln(r_2\delta t) - \left(N_{T_2} - N_2\right)r_2\delta t \\
&\cong N_1\ln(r_1) + N_2\ln(r_2) + N\ln(\delta t) - T_1r_1 - T_2r_2 \\
&= N_1\ln(r_1) + N_2\ln(r_2) + N\ln(\delta t) - N.
\end{aligned}$$

This leads to $P(T_1,T_2|N_1,N_2) \cong {r_1}^{N_1}{r_2}^{N_2}\,\delta t^N e^{-N}$. Removing the last two terms that do not depend on the separation into subintervals, we find

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

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.

*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 those spike counts being produced by a single Poisson process of constant rate.

In this case, we use the Poisson formula (Chapter 2, Appendix)

$$P(N|rT) = \frac{(rT)^N e^{-rT}}{N!},$$

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

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

We find the ratio produced by this formula for the case of a change-point with $r_1 = N_1/T_1$ and $r_2 = N_2/T_2$ versus the case of no change-point with $r_1 = r_2 = N/T$, where $N = N_1 + N_2$.

The result gives us the probability of a change-point at $T_1$ as

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

The advantage of this formula, which is a likelihood ratio, 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. Such a procedure has many similarities to Hidden Markov modeling.