MCN 2024, Problem Set 1

1 PCA and SVD

Experimental technologies enable the simultaneous recording of large numbers of neurons, but visualizing and analyzing such high-dimensional neural trajectories can be challenging. In this problem, you will visualize the activity trajectory of a population of neurons using the classic PCA and SVD techniques for reducing dimensionality.

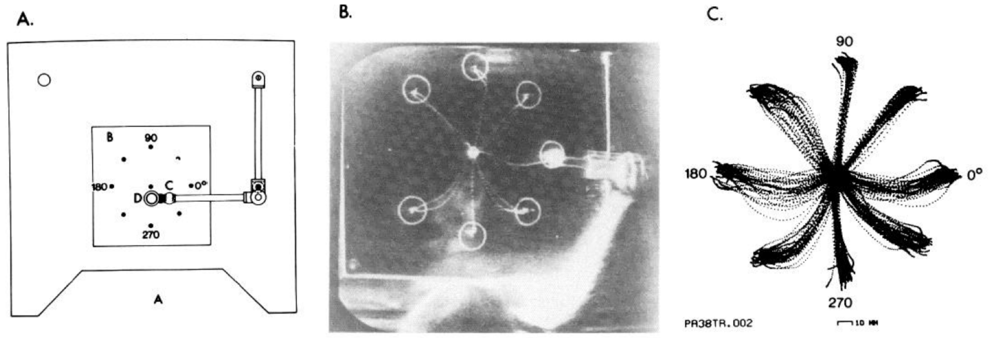

Figure 1: **(a)** A diagram of the manipulandum. **(b)** A cave painting of a monkey operating a manipulandum. **(c)** Hand trajectories for a center-out reach task.

The Experiment: A monkey was instructed to move a manipulandum, which is an exoskeleton that fits over the arm and constrains movement to a 2-D plane. Think of the manipulandum as a joystick controlled by the whole arm. The behavioral task was the center-out paradigm pioneered by Georgopoulos and colleagues (1982). The monkey first holds the cursor over the center target for 500 ms. Then a peripheral target appears at one of eight locations arranged in a circle around the center target. In our task there is an instructed delay, which means that after the peripheral target appears the monkey must wait approximately 1-2s for a go cue. After the go cue, the monkey moves its hand to the peripheral target and holds for 500 ms, and the trial is completed. The population of neurons that you will analyze was recorded from motor and pre-motor areas of a monkey performing this center-out reaching task. These data are adapted from an assignment for Nicho Hatsopoulos' Computational Neuroscience course at University of Chicago.

The Data: `HatsopoulosReachTask.mat` is a file that contains the neural activity. Load this file in MATLAB or python. It contains the following variables:

- `numNeurons, numTimebins, numTrials`
- `firingRate` (`numNeurons x numTimebins x numTrials`) firing rate of each neuron, in each time bin, on each trial
- `dt` length of each time bin in seconds
- `cueTime, goTime`, time of the instruction cue and the go signal in seconds
- `direction` (`1 x numTrials`) the direction of the reach (takes a value from 1-8, starting from 0°, then 45°, on to 315°)
- `brainRegion` (`1 x numNeurons`) either "M1" or "PMd."

Before doing any computation, explore the raw data through simple visualizations to understand its structure.

1.1 Construct the data matrix

Make a new 3D array that contains the trial-averaged activity trajectory of each neuron for each of the 8 reach directions (this array should be `numNeurons x numTimebins x 8`). Both PCA and SVD are a 2D matrix decompositions, and there are several ways in which we can construct a 2D data matrix from the 3D raw data, each of which can be useful for different types of analysis. Construct a 2D data matrix `X` by concatenating the elements of the last dimension (i.e. the reach directions) to yield a 2D matrix that is `numNeurons x (numTimebins * 8)`.

1.1.1 What might be the goal of averaging and concatenating the data in this way?

1.1.2 **Extra credit:** We encourage you to try other ways of constructing a 2D data matrix and to think about what questions can be answered by performing dimensionality reduction on these alternative data matrices.

1.2 Compute the principal components from the neuron point of view

1.2.1 First, mean-center your 2D data matrix by subtracting the means off each row. In this case, we are removing the mean firing rate from each neuron. Display your centered 2D data matrix and describe any structure you see in this data.

1.2.2 Calculate the covariance matrix between neurons and plot it. ***Note: do not use the built-in "cov" function, we want you to compute the covariance matrix manually.***

1.2.3 Perform an eigendecomposition on the covariance matrix. Plot the eigenvalues and comment on the underlying dimensionality of the data. How many eigenvectors are needed to capture 90 percent of the variance?

1.2.4 We started out with eight high-dimensional neural trajectories, one for each reach direction. In order to visualize your data, project it onto the first 3 principal components to yield a 3D representation of the neural trajectories. Your new matrix should be 3 x (numTimebins * 8), which you should reshape into a 3 x numTimebins x 8 array. Plot all 8 trajectories on the same 3D plot, each in a different color. Comment on the structure of the trajectories. How well separated are the different reach directions?

1.2.5 Why is it useful to center the matrix? (What happens if you don't center the matrix?)

1.2.6 Suppose you instead centered the data by subtracting the means off the columns. What happens when you apply PCA to this matrix? In particular, what happens to the set of eigenvalues?

1.2.7 What happens if you z-score the data (i.e. subtract the mean and divide by the standard deviation across the rows)? Why might this be good or bad?

1.2.8 More generally, what properties of a dataset would make mean subtraction column-wise and/or row-wise a good or bad idea? What about z-scoring?

1.3 Compute the singular value decomposition

Compute the singular value decomposition $X = USV^T$ on the mean-centered data matrix X (means subtracted off of the rows).

1.3.1 How many rows and columns do U, S and V have? Provide an interpretation for the columns of U and V in terms of the original experiment and the neural recording. Plot the singular values of the data matrix and comment on the underlying dimensionality of these data. What is the relationship between the singular values and the eigenvalues from PCA?

1.3.2 To visualize the data, project it onto the first 3 left-singular vectors to yield a 3-dimensional representation of the neural trajectories. Your new matrix should be `3 x (numTimebins * 8)`, which you should reshape into a `3 x numTimebins x 8` array. Plot the neural trajectories using the same colors as for PCA. How does the low-dimensional projection compare to that obtained with PCA? What assumption is needed on the data matrix X for PCA and SVD to yield equivalencies between the singular values/eigenvalues, and the eigenvectors/left singular vectors?

1.4 Learning linear decoders for neural activity

The analysis we have done so far help us visualize the activity of these neurons in a low-dimensional space. However, these visualizations can sometimes be misleading, because there are many more neurons (or dimensions) than we can visualize. Analyses where we learn decoders for the neural activity in the full representational space can yield insight into the full geometry of the representations. For the following questions, use the data matrix collapsed across time points, so that in each case you have a matrix with (numTimeBins * numTrials) x numNeurons. (Note: You can use fewer of the time bins to both test different hypotheses and speed up your decoder fitting.)

1.4.1 Train a linear decoder to determine from the neural activity whether the animal is reaching to the "left" or to the "right" (don't worry about getting the true direction, splitting the directions into two halves is all right, see fig. 2).

1.4.2 Try a few more ways of splitting the directions into two, contiguous groups. Can you successfully decode all of them? Or are certain splits "special"?

1.4.3 This binary classification of two groups of directions is consistent with a flat picture of direction space. However, is there more structure here? Can you use a linear decoder to decode groups of directions that are non-contiguous (e.g., an XOR-like grouping of the directions, see fig. 2)? What does this mean about higher dimensional structure in the representation? What would being able to decode these groupings mean about the structure of the representation if you used a nonlinear decoder? Discuss how this form of analysis complements (or doesn't) the analyses you did earlier in this problem.

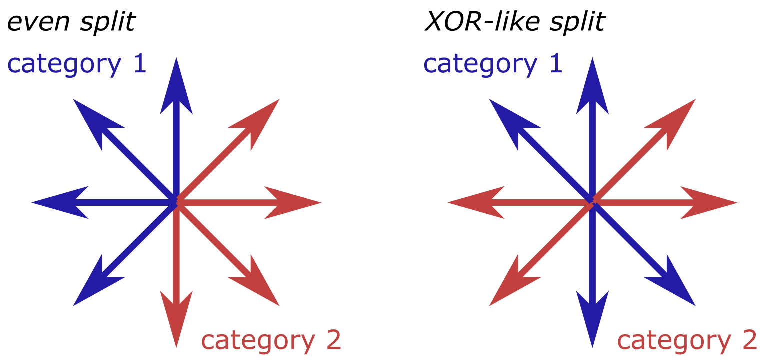

Figure 2: (left) A schematic of a possible "left-right" split for decoding. (right) A schematic of a possible "XOR-like" split for decoding.

2 Integrate and Fire Neuron

In this question we explore a series of very simple neuron models that are tractable to thorough analysis.

2.1 Compute the F-I curve for the leaky integrate-and-fire model.

The membrane potential, V, obeys the following dynamics:

$$C\frac{dV}{dt} = -g_L(V - E_L) + I\,, \tag{1}$$

where C is the membrane capacitance, g_L is the leak conductance, E_L is the equilibrium/reversal potential of the leak current, and I is the input current. The model has a threshold voltage V_T, and a reset voltage V_R (with $V_R < V_T$). Compute the F-I curve (plot of spike frequency vs input current) of this model. How does this curve depend on the threshold and reset voltages.

2.2 Do the same for a version with excitatory synaptic input

Instead of an applied current through an electrode, assume the input to the neuron is through a synaptic conductance, g_S, with reversal potential E_S:

$$C\frac{dV}{dt} = -g_L(V - E_L) - g_S(V - E_S)\,, \tag{2}$$

Assume $E_S > V_T$ and $g_S > 0$. Plot the curve as a function of g_S when $C = 1\mu\text{F/cm}^2$, $E_L = V_R = -65\text{mV}$, $g_L = 0.1\text{mS/cm}^2$. For fun, simulate the output using MATLAB, python, XPP, or your cell phone.

2.3 Subthreshold dynamics of the quadratic integrate and fire model with adaptation

Consider a quadratic integrate-and-fire model with an adaptation variable W:

$$\frac{dV}{dt} = V^2 + I - W \tag{3}$$

$$\frac{dW}{dt} = a(bV - W)\,. \tag{4}$$

First find conditions where there are 2 equilibria and no equilibria and determine the stability of the equilibria when they exist. Assume $a > 0$. Find and plot a curve of Hopf bifurcations, $b(I)$. Similarly plot a curve of saddle-node bifurcations. Where do the two curves meet (Bogdanov-Takens bifurcation) as a function of a? For fixed positive a, sketch out regions in the two-dimensional parameter space of I and b corresponding to different dynamical behaviors of the model neuron (eg, silent, subthreshold oscillations, tonic spiking (by repeatedly blowing up to threshold)).

2.4 Fun with planar dynamics

In all cases below, we study the following system:

$$\frac{dV}{dt} = f(V, n) \tag{5}$$

$$\frac{dn}{dt} = g(V, n)\,, \tag{6}$$

where f and g are continuously differentiable nonlinear functions.

For all but the extra credit question, we assume:

$$f_n < 0,\ g_n < 0,\ g_V > 0$$

where f_n means the partial derivative with respect to n, etc. This is related to the Morris-Lecar model discussed in the Ermentrout lecture.

2.4.1 Suppose that $f_V < 0$ at an equilibrium point. Prove that this equilibrium is stable.

2.4.2 Further suppose that $f = 0$ has the usual cubic shape (decreasing left/right branches and increasing middle branch). Also assume that $g = 0$ has the usual monotonically increasing shape. Show that if there is an equilibrium on the middle branch of the cubic, then $f_V > 0$ for this equilibrium.

2.4.3 Suppose that there is a middle branch equilibrium. Prove that this equilibrium is a saddle (node) if the slope of the n-nullcline is less (greater) than the slope of the V-nullcline.

2.4.4 **Extra Credit:** Consider a general system. Prove that there are no limit cycles in a planar system when $f_n g_V > 0$ for every V and n.

2.4.5 **Extra Credit:** Numerical fun - find a value of the current such that the standard HH equations have *four* limit cycles.

3 Spike-Triggered Average and Covariance

Here we will explore some methods for spike train analysis. To study the effectiveness of the methods, we will generate the stimulus and spikes ourselves and compare the results of the analysis to the ground truth.

Reference: Sharpee TO (2013) Computational identification of receptive fields. *Ann Rev. Neurosci.*

3.1 Design your stimulus

Generate a 1D Gaussian white noise stimulus $s(t)$, with time bin width of 2 ms, and t ranging between 0 and a T of your choosing.

3.2 Simulate data

Generate synthetic data from a model neuron with a linear temporal filter and a spiking nonlinearity. Define the neuron's linear temporal filter by the kernel

$$f(t) = e^{-t/\tau} \sin(\omega t) \qquad 0 \le t \le 50 \text{ ms} \tag{7}$$

where $\tau = 10$ ms and $\omega = 0.3$ rad/ms. Sketch this function.

The excitatory drive to the neuron at time point t is given by the discrete time convolution:

$$u_f(t) = \sum_{0 \le \tau \le 50} s(t - \tau) \cdot f(\tau)\,. \tag{8}$$

The firing rate has a sigmoidal dependence on the excitatory drive:

$$r(u_f(t)) = \frac{r_{\max}}{1 + \exp\left[(\theta - u_f(t))/(\Delta)\right]}\ . \tag{9}$$

For the purposes of this question, let $\theta = 5$ and $\Delta = 1$. Start with $r_{\max} = 1$, and simulate spikes at rate r. Then tune $r_{\max}$ so that the neuron spikes at around 20 Hz.

3.3 STA

Compute the spike-triggered average on your simulated spike train. How close is it to $f(t)$? Roughly how many spikes do you need in order to accurately estimate $f(t)$?

3.4 STC: one-dimensional case

Now consider a model with a quadratic non-linearity applied before the sigmoid:

$$r(u_f(t)) = \frac{r_{\max}}{1 + \exp\left[(\theta - u_f(t)^2)/\Delta\right]}\ . \tag{10}$$

In this case, let $\theta = 15$ and $\Delta = 2$.

3.4.1 Simulate a spike train and compute the STA. Are you able to recover $f(t)$? Explain why.

3.4.2 Compute the STC and discuss its success in recovering the model parameters.

3.5 STC: two-dimensional case

Now consider a model with two linear filters, f and g:

$$r(u_f(t), u_g(t)) = \frac{r_{\max}}{(1 + \exp[(\theta_f - u_f(t)^2)/\Delta_f])\,(1 + \exp[(\theta_g - u_g(t)^2)/\Delta_g])}\ . \tag{11}$$

For simplicity, let $\theta_f = \theta_g = 15$ and $\Delta_f = \Delta_g = 2$. Let $g(t) = -e^{-t/\tau_g}\cos(\omega_g t)$ with $\tau_g = 30$ ms and $\omega_g = 0.2$ rad/ms. Use eq. (8) to compute $u_g(t)$ in analogy to $u_f(t)$.

3.5.1 Discuss the meaning of the parameters Δ_f, Δ_g, θ_f, θ_g, and sketch r in the space $(u_f\ , u_g)$.

3.5.2 Compute the STC. Do you get the same filters as you put in? Why or why not?

3.5.3 Plot the nonlinearity from the two dimensional distributions.