CHAPTER

22

Neural Decoding II: Continuous Variables

22.1 GOALS OF THIS CHAPTER

The preceding chapter explored methods of decoding movement direction from neuronal signals. The movement direction was a discrete stimulus, taking on one of eight possible values. In this chapter, you will look at how to decode a continuous, time-varying stimulus from neuronal signals. Specifically, this chapter will address how to decode the instantaneous hand position from a population of motor cortical neurons recorded from a macaque monkey. You will also see how information about how the hand position changes over time can be used to improve your decoding.

Note that this chapter assumes completion of Chapter 17, "Neural Data Analysis I: Encoding," Chapter 18, "Neural Data Analysis II: Binned Spike Data," and Chapter 21, "Neural Decoding I: Discrete Variables."

22.2 BACKGROUND

We previously discussed how neurons in the motor cortex carry information about upcoming movements. In the last chapter, you were able to use this information to decode the direction of a movement made to one of eight targets. But what do you do if movement isn't so constrained as it is in the center-out task? Another common experimental paradigm in motor control literature can be described as the "pinball" task: a target appears somewhere in a 2D playing field, and as soon as the participant moves the cursor within this target, a new target appears at a different, randomly selected position. There are no hold times in this task, unlike the center-out task, so the hand is constantly moving.

You are interested in decoding the trajectory of the hand, meaning you want to know *X*- and *Y*-positions of the hand at each time point. *X*- and *Y*-positions are examples of kinematic variables, meaning they describe the motion of the object (the hand), but not the forces that generated the motion. Considering just the limb kinematics is easier than considering the full limb dynamics, which requires a model of how muscle forces and the physical properties of the limb interact to produce motion.

What method can you use to decode these kinematic variables? You could just modify the algorithms you have already seen. Recall that the population vector has a magnitude as well as direction. If you assume this magnitude is proportional to the instantaneous speed, you could simply compute a population vector for each time bin and then add them tail to tip to create an estimate of the trajectory. This was, in fact, an early approach to the problem (Georgopoulos, 1988).

In this chapter we will take a couple of different approaches. We mentioned previously that a simple neuronal encoding model assumes the firing rate varies linearly with stimulus intensity. You can apply this to motor cortical neurons and assume they fire linearly with the instantaneous hand position. For decoding, you simply invert this equation and derive an estimate of hand position using a linear function of firing rates. Another approach is to divide the *X*- and *Y*-axes into a grid and then use the same maximum likelihood method we used for center-out data to determine which grid square the hand is located in. As you will see, the fact that the hand position varies smoothly as a function of time introduces some new wrinkles.

Load the dataset for this chapter, available on the companion web site (data courtesy of the Hatsopoulos laboratory). There are two main variables: *kin* stores the *X*- and *Y*-positions (sampled every 70 ms), and *rate* stores the number of spikes in each 70 ms bin. Now look at the relationship between just one kinematic variable (*Y*-position) and one neuron's spike count. Notice how the indices are offset to create a vector of spike counts that lead the kinematics by two time bins (or 140 ms):

```
lag = 2;                 %Lag between neural firing and hand position
y = kin(lag + 1:end,2);  %Y-position of the hand
s = rate(1:end-lag,36);  %Corresponding spike count of one neuron
```

We are interested in how the spike count varies as a function of hand position (for simplicity, right now we only focus on position in the *Y*-direction). However, if you were to plot the position versus the raw spike count as a scatter plot, the result would be confusing, because the spike counts are highly variable. It is better to look at how the mean spike count varies as a function of position. To do this, we will divide the position variable into 15 1-cm wide bins, and average the corresponding spike counts. We will also keep track of the standard deviation, so we can compute a standard error: this is equal to the standard deviation divided by the square root of the number of data points. We can then plot a confidence interval for the empirical mean, as the "true" mean should fall within two standard errors of empirical mean 95% of the time. The error bars corresponding to a confidence interval can be plotted using the MATLAB® function **errorbar.**