Frequency of Seeing, Part 2

Introduction

In this HW, you will use parameter search to recover the detection criterion and fraction of photons absorbed from a simulated frequency of seeing curve.

Step 1

Download and review the file poissonFOSFitExample.m from the class website. This starts with a slightly generalized solution to part 1 of the homework. The sense in which it is generalized is that it expresses the stimulus in terms of the mean number of quanta in the test flash, and has a parameter called fractionAbsorbed that converts this to the number of quanta absorbed. [It also has a parameter called darkNoise. You can ignore this parameter for the required part of the HW, by keeping it set to zero. This parameter describes the fact that in real photoreceptors, there will be a certain number of times when a photopigment molecule isomerizes spontaneously, even when no light is entering the eye. This is an additional source of stimulus independent noise, which a more advanced model of absolute threshold than we have considered would take into account.]

The example program then uses Matlab's fmincon to find the detection criterion from the simulated data. In doing so, it assumes that the values for the fractionAbsorbed and darkNoise parameters are known – that is, these parameters are passed to the error function and used in the computation of the likelihood. The example program is basically the one used in class last time, but with a few tweaks and comments added.

Before you proceed further, you should study this program and make sure you know understand how it works. Useful things to do as part of this include a) varying nTrials (small number means noisy simulated data, large number means low noise simulated data) and seeing how this affects the precision of recovery and b) varying the starting point for the search to get a sense of how far from the underlying simulated value it can get before the search fails.

Step 2

Your mission is to modify the program so that it finds both the detection criterion and the fraction absorbed parameter.

First, delete the section that makes a plot of the error function, since you won't need that and generalizing that plot to two search variables is not the focus of the homework.

Second, and the part that is the focus, modify the section that does the searching so that it works on a vector rather than a scalar. So instead of passing the parameter fractionAbsorbed to the error function, this parameter should be represented as the second entry of your parameter vector. You'll need to initialize the parameter vector with guesses for the criterion and fraction absorbed, use these parameters in the computation of the error function, and extract them from the parameter vector returned by fmincon.

Once you get it working, explore the effect of varying nTrials as well as the simulated values for the criterion and fraction absorbed. Is there a regime where the search works well and where it doesn't? Email me your program and a few thoughts on what you found.

Note: The example program has a function poissonFractionSeen which computes the theoretical fraction seen for any set of parameter values. This is written so that it produces a continuously varying prediction as a function of the criterion. You don't need to modify this routine, it is set up so that it will work for the assignment. It does this by interpolating between the values between the two nearest integer values of the passed criterion. Even though it really only makes sense in the context of a Poisson process to have integer values, parameter search programs vary their parameters as real numbers, and they will get stuck if something sensible isn't returned when they pass non-integer numbers. "Something sensible" includes the fact that a small change in a parameter should produce a change in the error function – if the error function is constant across a range of parameter values, the fmincon will think it has reached a minimum. There are search routines that search over integers, and using one of those would be an alternative here. But most models we consider in neuroscience don't need integer search so I generally find it easier just to deal with integers in the error function as is done here.]

Extra Credit (or just for fun)

Further modify your program so that it searches on the level of dark noise as well as the two parameters above. You will discover, once you get your program working correctly, that you can get very good fits to the data with parameter values that are wildly off. This isn't a bug (at least, not if you wrote your program correctly). Rather, sometimes a set of model parameters is underdetermined by data. In such cases, small measurement error can result in large deviations of the estimated model parameters from their true values. This is called a problem of identifiability in control theory. Basically, the type of frequency of seeing curve we're simulating can do a pretty good job of identifying two of the three parameters we've considered, but doesn't have enough power to identify all three. This type of ambiguity places limits on what can be inferred about the underlying process from model fits to behavioral data, and it is good to be aware of this possibility.