Thresholds Part 1

Introduction

The goal of this lab is to introduce you to the measurement of psychophysical thresholds. A threshold is defined generally as the smallest change in some aspect of a stimulus required for an observer to be able to detect the change or to discriminate some aspect of the stimulus.

There are lots of ways to measure threshold. One could, for example, ask the observer to adjust the stimulus until he/she can just barely perform the requisite visual task (e.g., see that it is there, tell which way it is moving, etc.) In modern work, generally some sort of forced-choice task is preferred.

In a one-interval presentation method, the stimulus in question is presented once on each trial and the observer answers a question about it. The question usually has just two possible answers, although generalizations of this are possible. The question is sometimes phrased in yes-no form (e.g., "was the stimulus presented on this trial?") or in terms of some aspect of the stimulus (e.g., "which direction, left or right, was the stimulus moving?"). In the first part of this lab, you will use this method to measure threshold for discriminating direction of motion.

There is a second method, the two-alternative forced choice method, that can also be used to measure thresholds. Here there are two presentations, one where the stimulus of interest is present and one where it is not, and the subject is asked to indicate which presentation contained the stimulus of interest. The presentations are generally separated either in space or in time. In the second part of this lab (to come), you will use this method to measure the same type of threshold.

The stimuli

The stimuli in this experiment consist of arrays of correlated moving random dots. Such stimuli are useful for studying motion because a) the provide a stimulus without much structure, so that motion detection mechanisms are isolated and b) they provide a way to titrate the strength of motion. The Newsome et al. reading describes these stimuli and how they were used to probe neural mechanisms of motion processing.

The moving dot arrays are specified by a number of parameters. These determine the number of dots, their size, how fast they change, etc. But within a single experiment, the primary parameter of interest is what we call the dot correlation. This parameter controls the strength of dot motion while leaving other properties of the stimulus unchanged. Dot correlation can range from -1 to 1. Here is what the correlations mean.

a) A correlation of 1 means that all of the dots move to the right together. This is the strongest rightward motion.

b) A correlation of -1 means that all of the dots move to the left together. This is the strongest leftward motion.

c) A correlation of 0 means that the dots all move in random directions, there is no systematic motion at all.

d) Correlations p, where $0 < p < 1$, mean that the fraction p of the dots move together to the right while the rest move randomly. So if $p = 0.5$, one-half of the dots are moving together to the right and the rest are moving randomly. Using values of p between 0 and 1 produces sensations of rightward motion, but not as strong as with $p = 1$. Indeed, by varying p from 0 to 1, you can systematically vary the strength of the rightward motion signal.

e) Correlations p, where $-1 < p < 0$, have the same meaning as their corresponding positive counterparts, except that the dots move to the left.

In the experiment, you judge trials where p varies between -1 and 1.

Most of the other parameters will have meanings that are obvious from the comments in the program. The one that may be mysterious is the dot lifetime. Each dot moves continuously for a time specfied by this variable, then disappears and is replaced at a random location by a new dot. Making the lifetime short generally makes the task harder, all else being equal. Setting the lifetime to Inf means that each dot moves continously forever. Short lifetimes (and similarly short stimulus presentations) also mean that the task must be done on the basis of an overall sense of motion rather than by tracking individual dots.

Running the experiment

The experimental program is called `DotThreshold.m.` This file sets the parameters and runs the experiment. The data are saved in the same general fashion as for our previous program.

On each trial, an array of dots is presented at some correlation, with the correlations as specified in the parameter vector `test.dotCoherence`. The subject responds as to whether they are moving left or right.

Since there is an objectively correct answer to this question (except when dot correlation is 0), the program can provide feedback after each trial as to whether the answer was correct or not. This is controlled by a variable `enableFeedback.` Set this to 1 to provide feedback. Providing feedback is common when using objective psychophysical methods. For trials where the dot coherence is 0, the program indicates correct or incorrect randomly, with half of the time the answer being correct.

The other parameters for the test dots are fixed within a session but can be varied between sessions.

The program runs through the list of dot coherences `nBlocks` times. The data saved are the responses on each trial for each coherence. A value of -1 indicates that the subject responded left, while a value of 1 indicated that the subject responded right.

The program has a number of other features which you can ignore for this lab.

Your mission

You mission is to measure how threshold for detecting motion direction in random dots varies as a function of some other parameter. That other parameter could be dot speed, dot lifetime, dot size, etc. But, although the program will let you show adapting dots, don't do this.

For each value of the "other" parameter, you should measure a psychometric function using the method of constant stimuli. You will then need to write analysis code that extracts threshold from data. To do this, fit a cumulative normal curve to the data (using your now excellent skills with fmincon) and extract threshold from the fit. As part of this, you'll need to write code to read and aggregate the data across sessions. The data analysis code from the magnitude estimation lab should provide a good starting point for the latter.

The most natural form to plot and fit the psychometric functions in for this experiment is of the form probability subject indictes one of the responses (e.g. 'right') versus dot correlation. This function should vary between 0 and 1 as dot correlation goes from -1 to 1. When you fit the psychometric function, you will find the two parameters of a normal (mean and standard deviation) such that the cumulative normal maximizes the likelihood of the data. The function `normcdf` will compute the cumulative normal given the mean and standard deviation. The techniques you are learning the the frequency of seeing HW will help you write the fitting function.

You will not need to modify the experimental program to do this experiment, but you will need to think about the parameters you choose. You want to run some pilot experiments to figure out what good dot correlations to use are, for whatever other test parameters you pick. You want the test parameters to be such that performance ranges from essentially perfect at the extremes to near chance in the middle, with some values along the rising part of the psychometric function. Note that the dot coherences in the program as supplied are probably not a good choice.

For this experiment, it is OK for the experimenters to be their own subjects. So, each of you should observe in your own experiment, and you don't need to go out and find a bunch of subjects. You should make measurements for (at least) two values of the "other" parameter, and should repeat each condition at least twice for each subject. More generally, the lab will be more satisfying if you get enough data for each subject to map out a reliable frequency of seeing curve.

What you should turn in

Each group should produce a wik page that summarizes their data (with graphs), along with captions that explain the graphs. At least one graph should show an example of a measured psychometric function with a cumulative normal fit through it, and the caption for this graph should explain how an estimate of threshold was extracted from the fit. At least one graph should show how threshold changed (or didn't) with whatever "other" parameter you explored. The graph or graphs should allow comparison of the data from individual subjects.

You should also be sure to write down your methods (on the wiki page). We will show how to use the wiki.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% EXPERIMENTAL PARAMETERS
%
% Adaptation parameters.  The adaptor dot correlation determines
% how strong the adapting stimulus is.  The number must between
% -1 and 1.  Negative numbers move to the left, positive numbers move
% to the right (when adaptHOrV = 0).
adaptDotCorr = 0.0;           % Adapter dot correlation
nAdaptDots = 1000;            % Number of adapter dots
adaptRectSize = 479;          % Size of adapter rectangle
adaptDotSize = 2;             % Size of adapter dots
adaptDotMove = 3;             % Controls size of dot moves, adapter
adaptHOrV = 0;                % Angle of dot motion, 0=>left/right, 90=>up/down, etc.
adaptNoRandom = 1;            % Orthogonal motion?, 1=>yes, 0=>no

% Test parameteters.  The list testDotCorrs determines what dot
% correlations you will be tested on.  The numbers must be between
% -1 and 1.  Negative numbers move to left, positive numbers move
% to the right (when testHOrV = 0).  It is wise to use a symmetric
% list and to include 0.0.
testDotCorrs = [-0.15 -0.12 -0.09 -0.06 -0.03 0.0 0.03 0.06 0.09 0.12 0.15];
nTestDots = 300;              % Number of test dots
testRectSize = 250;           % Size of test rectangle
testDotSize = 2;              % Size of test dots
testDotMove = 1;              % Controls size of dot moves, test
testHOrV = 0;                 % Angle of dot motion, 0=>left/right, 90=>up/down, etc.

% Experimental parameters.
nBlocks = 8;                  % Number of blocks
trialDuration = 250;          % Trial duration (milliseconds) ...
initialAdaptTime = 1;         % Time for initial adaptation (seconds)
topUpAdaptTime = 1;           % Top up adapt time (seconds)
waitKey = 0;                  % Wait on trials?

bgColor = [0 0 0];            % Color of background (RGB, each between 0 and 255)
adaptColor = [255 255 255];   % Color of adapter dots (RGB, each between 0 and 255)
testColor = [255 255 255];    % Color of square (RGB, each between 0 and 255)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```