CS/NEUR125 Brains, Minds, and Machines

Lab 9: Attention and perception: visual search and detection

Due: Wednesday, April 12

In this lab, we will program and run simple versions of two classic "psychophysics" experiments in MATLAB, to demonstrate some basic properties of human attention and perception. This exercise is adapted from Chapters 3 and 5 of *Matlab for Neuroscientists* by Wallisch et al.

To begin, create a copy of this Google document, modify the title of the copy to include your partner names, and share the copy between partners, as you did in previous labs. The questions for you to answer as you progress through the lab are shown in blue.

Start MATLAB on your lab Mac and use **Fetch** to download the folder named **psychophys_Lab** from the CS file server to the Desktop on your Mac. You will find this folder inside the cs125/download folder in your individual account on the CS server. For this lab, set the **Current Folder** in MATLAB to the **psychophys_Lab** folder.

I. Visual search and pop-out: feature primitives and the attentional bottleneck

In this part of the lab we'll collect and analyze **reaction time** data using MATLAB. This refers to measures of how long it takes a person to respond to a stimulus, in specific circumstances. The following introductory paragraphs are taken from Chapter 3 of *Matlab for Neuroscientists*.

"Reaction time measures to probe the mind have been the backbone of experimental psychology at least since the time of Donders (1868). The basic premise underlying the use of reaction times in cognitive psychology is the assumption that cognitive operations take a certain and measurable amount of time. In addition, it is assumed that additional mental processes add (more or less) linearly. If this is the case, increased reaction times reflect additional mental processes. We'll assume for the time being that this is a reasonable framework and write a program to collect reaction time data.

Understanding how the mind/brain decomposes a sensory scene into **features** is one of the fundamental problems in experimental psychology and systems neuroscience. We take it for granted that the visual system, for example, appears to decompose objects into different *edges, colors, textures, shapes, and motion features*. However, it is not obvious a priori which features actually represent **primitives** that are encoded in the visual system. Many neurophysiological experiments have searched for neurons that are tuned to features that were chosen somewhat arbitrarily based on the intuitions of the experimenters.

Psychologists, however, have developed behavioral experiments by which **feature primitives** can be revealed. In particular, a study by Treisman and Gelade (1980) has been particularly influential. This is probably due to the fact that it is extremely simple to grasp, yet the pattern of results suggests provocative hypotheses about the nature of perception (e.g., feature primitives, serial search, etc.).

So what is the visual search and pop-out paradigm that was used in the Treisman study? Subjects are asked to report the presence or absence of a target stimulus (in this case, a colored lowercase letter “o”) among various numbers of distractor stimuli. If the distractor stimuli are just of a different color--that is, if they differ by a single feature--you usually find the “pop-out” effect: the reaction time to detect the target is independent of the number of distractors. Conversely, if more than one stimulus dimension has to be considered to distinguish targets and distractors (conjunction search), you typically find a linear relationship between reaction time and the number of distractors. [Refer to Figure 1 below.]

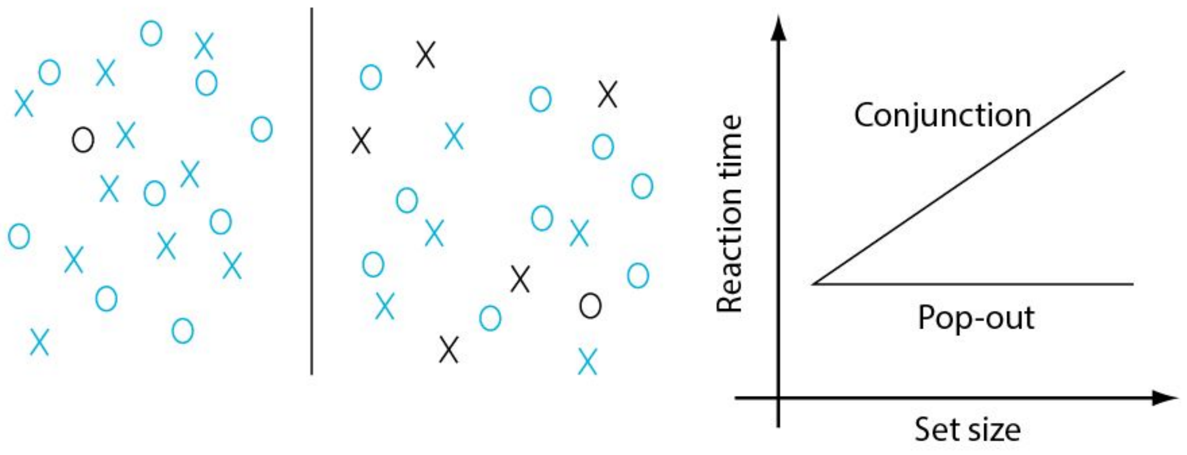

Figure 1: Popout (left) and conjunction (middle) conditions in the visual search reaction time task. The target is the black O.

As pointed out previously, this pattern of results immediately suggests the existence of “feature primitives,” fundamental **dimensions** that organize and govern human perception as well as a **serial scanner** in the case of conjunction search, where one element of the stimulus set after the other is considered as a target (and confirmed or discarded).”

> **Q1.** Before we do experiments like these, we don’t know what features are the “primitives” that our perceptions are built out of. For example, the *colors* or the *shapes* might correspond to fundamental primitive feature dimensions, or neither of them might. Which feature dimension is suggested to be “primitive” based on the hypothetical data shown in Figure 1?

We’ll discuss these implications further, but right now we’d like to collect reaction time data for this experiment where **the task is to search a visual display for a specific target, and report whether the target is present or absent on each trial.** Our specific goal for this part of the lab is to reproduce a visual search experiment to generate a result like the one shown in Figure 1 above. Specifically, our <u>hypothesis</u> is that when the target is distinguished by a single property, like color, then reaction time will not change with the number of items in the display, but when the target is defined by the conjunction of two properties, such as color and shape, then more mental processing is required and reaction time will increase with the number of items in the display.

Given that hypothesis, we want to develop a program to run the experiment and plot the results figure. *Thinking about what information we would need to plot that figure can help us to outline and organize our code for collecting the data.*

Looking at the figure we can see that we want to compare how long it takes to find a visual target in two conditions corresponding to the two lines plotted: (1) "**pop-out**," where the target differs from all the distractors by a **single feature** (e.g. color red), vs. (2) when the target is distinguished from the distractors by a "**conjunction"** (i.e. combination) of two features (e.g. shape and color: blue "O"). So our computer program will need to display trials for these two kinds of conditions.

What we're comparing between these two conditions is how reaction time changes when the number of distractors on the screen increases. So our computer program will have to make trials with different numbers of distractors, aka "set size." The plots themselves would be generated by plotting reaction times vs. set size with code something like:

```
plot(setsize_vec, meanRT)
```

where setsize_vec would be a vector of set sizes we tested, to be used as the x-axis, and meanRT would be a vector with an average reaction time for each of those set sizes, to plot. Except--we want to plot two different curves, one of meanRT's for the single-feature condition, and one for the conjunction condition. We can keep the reaction time data for both conditions in the same matrix if we set aside an empty matrix with dimensions of appropriate size like this:

```
meanRT_targs = zeros(2,nsets);   % to hold avg reaction time on CORRECT target trials for
                                 % each set size (i.e. number of distractors)
                                 % row 1 = popout, row 2 = conjunction

meanRT_notargs = zeros(2,nsets);  % to hold avg reaction time on CORRECT no-target trials
                                  % for each set size
                                  % row 1 = popout, row 2 = conjunction

correct_targs = zeros(2,nsets);       % to hold percent correct on target trials
                                      % for each setsize and conjunction condition
correct_notargs = zeros(2,nsets);     % to hold percent correct on no-target trials
```

We have two similar matrices with suffixes _targs and _notargs because we'll have trials with targets present and others with no target--otherwise there would be no need to search to report whether the target was present! But in the end we're mainly interested in the _targ reaction time data to make our figure. We're mostly interested in these reaction time data, but we should also keep track of the subject's performance in terms of percent correct for each condition: that's the reason for the correct_targs and correct_notargs matrices we're initializing in the lines above.

Let's go ahead and open a new m-file in the MATLAB editor and paste in the lines above (do not include the earlier plot command). Notice that the first line creates a matrix with two rows and nsets columns, to hold average reaction times. The two rows are for reaction times from the popout or conjunction conditions, and the columns are for the different numbers of distractors---but we haven't yet specified what those are. So add the lines below BEFORE the lines above in your m-file, to create setsize_vec, and from it determine the number of setsizes we'll be testing, nsets. We're also including a line to specify how many trials to run for each condition and setsize.

```
% loop through 2 conditions and multiple setsizes, keep reaction time results for each
setsize_vec = [4 8 12 16];      % each setsize is how many distractors of each type to display
ntrials = 10;                   % number of trials to run for each condition*setsize
nsets = length(setsize_vec);
```

And let's give a brief description of our m-file at the very top of the file; this description also reminds us what we're trying to code as we write the program! We're also including lines to clear the MATLAB workspace and close figures to start fresh.

```
% visualSearch
% Wallisch p60, Ch3.4 p67
% popout vs conjunction with varying set size
% present stimulus screen, respond target present (".") or absent (",")
% plot reaction times vs. # of distractors for popout and conjunction search

clear all
close all
```

Now save this draft of our script as visualSearch.m.

We've created the variables that will hold the data to make our figure, but we still need to generate trials that show targets and distractors, and collect the reaction time data. We have a situation where we want to present trials for each of the two conditions, for each of the four set-sizes. And for each combination of condition and set-size, we want to run multiple trials (How many? Whatever you set ntrials to). The underlined "for"s in the previous sentences are meant to suggest that we can cover all the combinations of condition (=popout or conjunction) and setsize (=[4 8 12 16]) by using **nested for-loops**.

Paste in these lines at the bottom of your m-file and run to illustrate this for yourself:

```
for icondition = 1:2 % 1=popout, 2=conjunction
   conjunction = icondition-1; % 0 = popout, 1 = conjunction
   iset = 0; % to count (ie. index) which setsize we're on in the loop below

   for setsize = setsize_vec     % how many of each distractor type
      iset = iset + 1;  % to count (ie. index) which setsize we're on

      icondition
      setsize

   end % loop over different set sizes (ie. number of distractors of each type)
end % loop over two iconditions: 1=popout, 2=conjunction search
```

icondition is an index to say which condition we're on, and setsize is which setsize we're on (from the list of setsizes stored in setsize_vec). Running this just prints out each combination of condition and set-size to the screen so you can see that every combination is covered. Once you see that, you can delete (or comment out) the two lines that display those values. The "iset" and "conjunction" variables defined in the lines above will come in handy as we'll see below.

For each of the condition*set-size combinations we're looping through in this nested loop, we need to run ntrials trials. So we'll just add another for-loop, this time over trials, *inside* the first two! Then we'll have ntrials trials for each condition and set-size. So add the lines below inside your inner for-loop, and use "smart indent" to format the whole m-file.

```
% background xs and os, same color for popout, different color for
% conjunction

% loop over trials for this condition and setsize, for each trial
% keep track of actual targ/notarg and subject's response

targs = zeros(1,ntrials);  % change to 1 if target present on trial i, zero if not
responses = zeros(1,ntrials); % change to 1 if subject says present on trial i, zero if
                              % subject says target absent
reactiontimes = zeros(1,ntrials); % to hold reaction time on each trial

for itrial=1:ntrials

    disp([ 'Trial number ' num2str(itrial + nsets*ntrials*conjunction + ntrials*(iset-1)) ])

end % loop over trials
```

We're including some comment lines to describe what we want to do, and some new variables to keep track of results for each set of trials we run. targs is a vector that will hold a 1 for those trials on which a target was present, so we know when the subject's response is correct. responses is a vector to hold a 1 for those trials when the subject said the target was present. reactiontimes is obviously to hold the reaction times for each trial, so we can average reaction times on the correct target-present trials for plotting.

The formula in the line that displays the current trial number is there so that we don't repeatedly count from 1 to 10 (if ntrials = 10), but instead count the total number of trials across the different conditions.

Now, to plot the distractors paste the following lines inside the inner loop, below the line that displays the Trial number. Use smart indent again to format your code.

```
% create a trial
   h = figure(1);  % default axes are 0 to 1
   title('target is blue O')
   xpos_o = rand(1,setsize);  % x coordinates of o's
   ypos_o = rand(1,setsize);  % y coordinates of o's
   xpos_x = rand(1,setsize);  % x coordinates of x's
```

```
ypos_x = rand(1,setsize);  % y coordinates of x's
for i=1:setsize                 % setsize is how many distractors of each type to display
   g = text(xpos_o(i),ypos_o(i), 'O');  % plot the Os in red
   set(g,'color','r','fontsize',20);
   g = text(xpos_x(i),ypos_x(i), 'X');
   set(g,'color','r','fontsize',20);
   if conjunction
      set(g,'color','b');            % if conjunction condition then plot the Xs in blue
   end % change color of one distractor type if in conjunction condition
end % loop over distractors
```

Notice that we used the rand function to generate random positions for all the distractors, and an if statement sets the display appropriately for the pop-out or conjunction conditions.

Now we want to add a target, but only on about half of our trials. We'll use the rand function again. Each time we call rand, it returns a random number between zero and one. The numbers from rand are "uniformly distributed," which means that about half the time they will be greater than 0.5. So if we only display a target when rand returns a number bigger than 0.5, it will happen about half the time. Add the following lines below the previous lines to take care of the target.

```
rnd = rand;              % random number between 0 and 1, uniformly distributed
if (rnd>0.5)             % only add target on half of trials
   % disp('target!')
   targs(itrial) = 1;  % to record that this trial had a target present
   xtarg = rand;  % x coord
   ytarg = rand;  % y coord
   gt = text(xtarg,ytarg,'O');  % it's an O so for conjunction search we want to make it
                                % the same color as the Xs
   set(gt,'color','b','fontsize',20);
   xlim([ -.1 1.1 ])
   ylim([ -.1 1.1 ])
end                      % target or not if
```

Now we have code to generate targets and distractors on a figure for multiple trials. We still need to collect the subject's response. But before we do that, let's add a few lines to create an initial display for getting ready before the first trial. Add these lines right before the first for loop in your m-file:

```
h = figure(1);
g = text(0.2,0.5, 'click here to begin');
set(g,'color','r','fontsize',20);
waitforbuttonpress;
close(1)
```

We still need a way to get the subject's response: whether the target was present or absent on each trial. We'll use the **tic** and **toc** functions. To see what these functions do, go to the command line and type "tic" then return, then type "toc" and return. The toc function returns the time that has passed since you ran the tic function! So basically we will run tic right after we display the figure of distractors (with or without a target), and call toc as soon as the subject responds, to see how long the response took: the **reaction time**. We'll use the pause command together with the get function as shown below to check what key the subject pressed. Note from the comment which key signals the presence of a target on the current trial.

```
% get subject's response: '.' for target present, ',' for absent
tic
pause % get keypress
hr = get(h,'CurrentCharacter'); % get character from keypress--only works if figure is
                                 % selected!
reactiontimes(itrial) = toc;
responses(itrial) = strcmp('.',hr);  % 1 if subject said target present
close(1)
```

Now we have code to run through ten trials for each condition and setsize--but we still need some lines to record the average reaction times for each of those conditions so we can plot them. The following lines figure out which are the correct target and no-target trials, and keep track of the results of each set of ten trials. The final two lines also compute the subject's percent correct for each condition. Insert these lines into your code *after* the trials loop, but *inside* the other two loops.

```
errortrials = find(targs-responses);  % should be indices of incorrect trials
correcttrials = setdiff(1:ntrials,errortrials);        % the correct ones are the ones that aren't
                                                        % wrong!
targinds = find(targs);        % indices of nonzeros in vector targs--i.e. indices of target trials
correctTargtrials = intersect(correcttrials,targinds); % intersection of correct trials and target
                                            % trials means those target trials that were correct!
correctNotargtrials = intersect(correcttrials,setdiff(1:ntrials,targinds)); % notarg trials are the
                                                                  %ones that aren't targs
% record results for this set size
meanRT_targs(icondition,iset) = mean(reactiontimes(correctTargtrials));
meanRT_notargs(icondition,iset) = mean(reactiontimes(correctNotargtrials));
correct_targs(icondition,iset) = 100*(length(correctTargtrials)/length(targinds)); % percent
                                                                            % correct
correct_notargs(icondition,iset) = 100*(length(correctNotargtrials)/(ntrials-length(targinds)));
```

Now place the following lines at the end of your file outside all loops to plot results of the experiment.

```
figure(2)
plot(setsize_vec,meanRT_targs(1,:),'b--')
hold on
plot(setsize_vec,meanRT_notargs(1,:),'r--')
plot(setsize_vec,meanRT_targs(2,:),'b')
plot(setsize_vec,meanRT_notargs(2,:),'r')
hold off
xlabel('Number of distractors of each type')
```

```
ylabel('Mean reaction time (s)')
legend('Popout target','Popout No-target','Conjunc target','Conjunc No-target')

figure(3)
plot(setsize_vec,correct_targs(1,:),'b--')
hold on
plot(setsize_vec,correct_notargs(1,:),'r--')
plot(setsize_vec,correct_targs(2,:),'b')
plot(setsize_vec,correct_notargs(2,:),'r')
hold off
xlabel('Number of distractors of each type')
ylabel('percent correct')
legend('Popout target','Popout No-target','Conjunc target','Conjunc No-target')
```

Smart indent, save, and run your experiment! If your first few trials are not real because you were still figuring out how the task works, break out of the program using **ctrl-c** and start over. (If for some reason you are not able to run the script from the editor after using ctrl-c, you can type the name of the script in the Command window to run it.)

> **Q2.** Drag your results figure for each partner into this document. Briefly summarize the results in words. Do they support the hypothesis that we proposed at the outset? If not, what might explain why not?

Implications from the visual search experiment:

- **Parallel search for a single feature:** Figure 1 would suggest that when a target can be identified by a single primitive feature (color), our brain can "check" all parts of the visual field simultaneously in parallel, so that adding more distractors makes little difference to reaction time. This makes sense if we imagine a network of "blue detector" neurons whose receptive fields cover the visual field--they all operate at the same time and any one of them can signal--I found it here!
- **Serial search for feature conjunctions:** When the target can only be identified by a combination of features (color and shape), adding more distractors increases reaction time on average. This suggests that in this case the brain must check each item serially, one after another, to find the target.
- In the conjunction case, psychologists sometimes speak of a "**processing bottleneck**" due to a **"limited attentional resource**." The "bottleneck" metaphor refers to the idea that only a limited amount of sensory information can "get through" to perception at a time. One way of conceptualizing this experimental result is to posit that to perceive feature conjunctions requires directing attention to one candidate item at a time.
- *But why is the brain limited in this way? What is different about perceiving combinations of features as compared to noticing a single feature popping out from the visual field?* We hope to address these questions in future classes.

II. Visual detection: the concept of a probabilistic threshold for perception

Stimuli that are too weak to be perceived can still stimulate a neural response. So what is different about the neural response when a stimulus is consciously perceived as compared to when it remains unconscious? Is it just "bigger"?

In this part of the lab, we'll run a simple visual detection task to illustrate the idea of a psychometric curve, and the threshold for perception. We'll be led to the idea that even a mental process as simple as visual detection can be unpredictable, or "stochastic" in nature. Specifically, the same dim light can be seen some of the times it is presented, and not seen other times. This fact turns out to be important in our quest to discover **neural correlates of consciousness**, because it means we can compare brain responses to *the same stimulus* when it is consciously perceived and not perceived.

From the Introduction to Chapter 5 of *Matlab for Neuroscientists:* "Psychophysics deals with the nature of the quantitative relationship between physical and mental qualities. Today, the practice of psychophysics is ubiquitous in all fields of neuroscience that involve the study of behaving organisms, be they man or beast. Curiously enough, the origins of systematic psychophysics can be traced to a single individual: Gustav Theodor Fechner (1801-1887). ...Born as the son of a pastor, he studied medicine…[and] was...professor of physics at the University of Leipzig. In the course of studying afterimages by gazing into the sun for extended periods of time--himself being the primary and sole subject--he almost lost his eyesight and went into deep depression in the early 1840s. This episode lasted for nearly a decade, a time which Fechner spent mostly within a darkened room. Emerging from this secluded state, he was overwhelmed by the sheer brilliant radiance of his surroundings, giving rise to his **panpsychist** worldview: he was now utterly certain that all things have souls, including inanimate objects such as plants and stones. Determined to share his insights with the rest of the humanity, he soon started publishing on the topic, formulating an "**identity theory**" stating that the physical world and the spiritual world are not separate entities, but actually the same--the apparent differences resulting from different perspectives (first versus third person) onto the same object. In his view, this reconciles the incompatible dominant philosophical worldviews of the 19th century: idealism and materialism. ...In order to convince his colleagues of the validity of his philosophical notions, he set out to devise methods that would allow him to empirically link physical and spiritual realms. His rationale being that if it can be shown that mental and physical qualities are in a clear functional relationship, this would lead credence to the notion that they are actually metaphysically identical.

Publishing the results of empirical studies on the topic in his Foundations of Psychophysics in 1860, he showed that this is the case for several mental domains, such as the relationship between physical mass and the perception of heaviness or weight. Fechner formulated several methods to arrive at these results that are still in use today. Importantly, he expressed the results of his investigations in mathematical, functional terms. This allowed the theoretical interpretation of his findings. Doing so, he introduced notions such as **sensory thresholds** quantitatively."

Following Fechner and (Hecht, Shlaer, and Pirenne, 1942) we're going to perform a visual detection task and plot a **psychometric curve** to characterize our performance detecting single pixels of varying brightness against a dark background, like the one in Figure 2 below. Using what is known as the method of constant stimuli, we will see how bright a target dot has to be to be reliably detected.

The script **detectionTask** runs 200 trials with target dots of varying brightness. Your task is to keep your gaze pointed at the fixation dot, and try to notice if there is a dimmer dot to the right of the fixation point. **Press "." for target present**, if you do see the target dot, or **press "," if it is**

absent. As soon as you press either of these buttons the next trial appears, even though there may be no obvious change in the display because the fixation point always looks the same and the figure window does not close between trials.

Run detectionTask.m to perform the experiment on yourself. Running 200 trials could take about ten to fifteen minutes. If it takes you a few trials to figure out how the task works, you might want to start over so your data won't be contaminated. Save a copy of your psychometric curve and your partner's. Since we are not using eye-tracking technology you will have to decide whether and how hard you will try to keep your eyes on the fixation point though you know the target points will appear a little to the right of the fixation point. (It would be interesting to get one curve for looking at the fixation point and another when you look directly where the target points appear….)

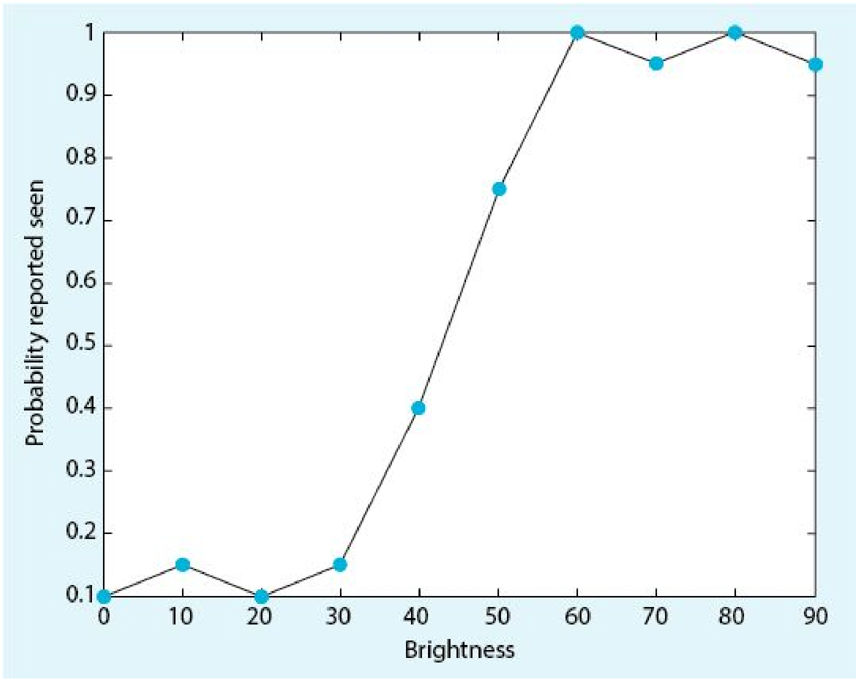

Figure 2: Psychometric curve from a visual detection task. The visual detection *threshold* in this experimental context can be defined as the brightness at which the probability of detection is 0.5.

> **Q3.** Drag the psychometric curve for each partner into this document. In terms of the monitor-specific brightness units on your plot, what is your visual detection threshold? Can you think of any factors that we didn't measure, that might have affected the value of your threshold?

From *Matlab for Neuroscientists*: "Also note that this value...is not inherently meaningful. Without having the monitor calibrated with a photometer, we don't know how much physical light energy this corresponds to. Hence, we can't relate it to the minimum number of light quanta that can be detected or such. However, *this threshold is meaningful in the context of the* *behavioral task*: a shifted threshold under different conditions can give rise to conjectures about the structure and function of the physiological system producing these thresholds….

Moreover, the absolute threshold is a **stochastic** concept. It is not true that lights below it are never seen." Stochastic means random or unpredictable--see below.

Conclusions from the visual detection experiment:

We've defined this idea of the sensory detection threshold as the intensity level where a subject can detect the target half the time. We don't want to misinterpret this as saying, "you can see brightnesses above the threshold and you can't see brightnesses below the threshold." While it may be true that there are some brightnesses that are bright enough that the subject always sees it, and some so dim that the subject never sees it, there are also those intermediate brightnesses--around the threshold--that are only seen some of the times when they are present.

It's not that we "half-see" near-threshold stimuli, it's that we see them on half the trials they are presented in. This means we have a situation where we are presenting the same stimulus multiple times, and some of those times it is "completely" seen, and on other trials it is not seen at all. (And there may be trials where we are not sure if we saw it or not. But generally, you either see it or you don't--whatever it is.) Now, this might not seem so remarkable since we've already noted that uncontrolled factors might determine whether a faint dot was seen or not. For example if your eye wanders towards the target zone you might detect a dot that would have been too faint if your eye had been perfectly directed at the fixation point, and that kind of thing could explain why you see it sometimes and not others.

But it seems that when we manage to control for those kinds of factors we still find that the perception is **probabilistic**--meaning random or not completely predictable.

- We seem to be led to the idea that **perception is "all or none."**[1] Although it is true that we can perceive things as "dim," and we can perceive things dimly, it seems that we perceive them or not. We don't usually half-perceive a dot or anything else.
- Regardless of whether we accept the first point, we have an experimental situation where the stimulus remains constant, so it is the internally generated variability of neurons in the brain that determines whether the target is perceived or not. As we will discuss in class, this sets up an opportunity to study what patterns of neural activity determine the difference between perceiving or not perceiving a stimulus. In other words, **near-threshold stimuli** provide a way to study the **neural correlates of consciousness.** *What has to happen in the brain in order for it to be conscious of something simple like a spot of light--or anything else?*

  Q4. What is the simplest explanation you can think of, in neural terms, for how it could be that the same near-threshold (i.e. weak) stimulus sometimes leads to a conscious perception and sometimes does not?

[1] ...like an action potential. For an action potential it's not completely true though. See Talis Bachmann (2013) essay on whether it's true for perception.