

Computational Vision

U. Minn. Psy 5036
Daniel Kersten
Lecture 3: The Ideal Observer

■ Initialize standard library files:

```
In[3]:= <<Graphics`Graphics`  
        <<Statistics`DiscreteDistributions`
```

We use `BarChart[]` in `Graphics.m` and `PoissonDistribution[]` in `DiscreteDistributions.m`.

Goals

Last time

■ Introduced basic strategy of our approach to vision:

A: Understand the information processing problem

- 1) Generative modeling--Understand the image (data) formation process: $S \rightarrow I$
- 2) Inference/estimation modeling--Understand how to best guess causes of data: $I \rightarrow S'$

B: Understand how well human vision handles the information ($I \rightarrow S''$), and from there how it works.

Generative model terminology: Some similar terms:

(Causes, scene properties, signals, hypotheses) \rightarrow (data, images, image measurements, features)

■ Limits to light discrimination

In the famous experiment of Hecht, Schlaer, and Pirenne, measurements were made of the proportion of times observers saw a dim small briefly flashed spot of light. Why was the subject response curve sigmoidal? If there was a specific light intensity (or number of photons) at transition from invisible to visible, we should see a "step" function--but we don't. The answer is *variability or "noise"*. But the variability may be due to several causes: the statistics of photon emission and absorption we've mentioned, "noisiness" in human responses, or in neural transmission. The physical limits to visual detection and discrimination can be traced to the stochastic nature of photon absorption and emission. This stochastic nature or variability is a consequence of the particle nature of light. A careful understanding of these limits in conjunction with psychophysical measurements is an essential component to understanding human sensitivity. Let's try to understand how to model variability in the physical stimulus and thereby understand the ideal observer's limitations. Eventually we will return to the question of what are the primary sources of variability in the human discrimination of light intensity.

The computational theory of signal detection and estimation was developed in the 1940's and 1950's. More information about **Signal Detection Theory (SDT)** can be found in: Green, D. M., & Swets, J. A. (1974) and in Van Trees, H. L. (1968).

Although, Hecht et al. measured thresholds for absolute detection; now we will study discrimination because it makes the key points clearer and generalizes better to the study of signal discrimination in many domains including those outside of vision.

Today

I am going to introduce some of the key concepts in Signal Detection Theory (SDT) and ideal observer analysis in the context of light intensity discrimination. The essence of the ideal observer approach is to ask: What are the theoretical limits to performance on a task? What is the optimal or ideal strategy for this task? In particular for our example, what are the limitations on the discrimination of light levels that any observer must face?

Our goals are:

- o Practice some *Mathematica* tools useful for modeling variability and ideal decisions
- o Intuitive introduction to the basics of probability distributions, cumulative distributions.
- o Introduce the idea of an **ideal observer** that models "external variability" and makes the best guesses, in the sense of minimizing error rate.
- o Introduce the yes/no task, and false alarm rates. Hit and false alarm rates completely characterizes the ideal's performance (Recall Hecht et al.'s experiment only reported hit rate, the "proportions of yes's" when the light was flashed).

Preview of needed concepts

Basic *Mathematica* concepts

- **Built-in functions.** *Mathematica* has a very large library of built-in functions. They all begin with an uppercase letter. You can get information about a function through the **Help Browser** under the **Help** menu item. You can also get, help on-line, e.g. for the exponential of a function, or for plotting graphs:

```
In[5]:= ?Exp
```

```
Exp[z] is the exponential function. More...
```

Exp[z] is the exponential function.

```
In[6]:= ?Plot
```

```
Plot[f, {x, xmin, xmax}] generates a plot
of f as a function of x from xmin to xmax. Plot[{f1,
f2, ... }, {x, xmin, xmax}] plots several functions fi. More...
```

Plot[f, {x, xmin, xmax}] generates a plot of f as a function of x from xmin to xmax. Plot[{f1, f2, ...}, {x, xmin, xmax}] plots several functions fi.

If you type two question marks before a function, ??Plot, you'll get more information. Try it. What does the **Random** function do?. **Add-in** function, such as BarChart[] need to be loaded in from standard packages (e.g. as above, <<Graphics`-Graphics` and <<Statistics`DiscreteDistributions`)

- **Defining functions.** You can define your own functions. Here is a gaussian function that we will use later:

```
In[7]:= gauss[x_, mean_, std_] := Exp[-((x - mean)^2) / (2 std^2)] / (std Sqrt[2 Pi]);
```

And you can view the definition in standard form by evaluating:

```
In[8]:= gauss [x, μ, σ]
```

```
Out[8]=
```

$$\frac{e^{-\frac{(x-\mu)^2}{2\sigma^2}}}{\sqrt{2\pi}\sigma}$$

The underscore, **x_ is important** because it tells *Mathematica* that x represents a slot, not an expression. Note that we've used a colon followed by equals (:=) instead of just an equals sign (=). When you use an equals sign, the value is calculated immediately. When there is a colon in front of the equals, the value is calculated only when called on later. So here we use := because we need to define the function for later use.

Also note that we could define our function with N[. *Mathematica* tries to keep everything exact as long as possible and thus will try to do symbol manipulation if we don't explicitly tell it that we want numbers. Try it:

```
In[9]:= ngauss [x_, mean_, std_] :=
  N[Exp[-((x - mean) ^ 2) / (2 std ^ 2)] / (std Sqrt[2 Pi])];
ngauss [x, u, s]
```

General::spell1 :

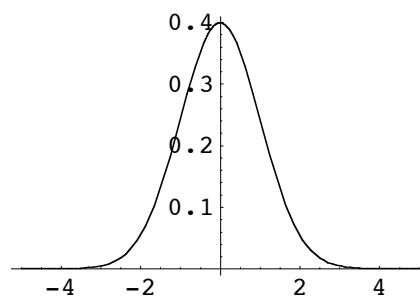
Possible spelling error: new symbol name "ngauss" is similar to existing symbol "gauss". More...

```
Out[10]=
```

$$\frac{0.398942 2.71828^{-\frac{0.5(x-u)^2}{s^2}}}{s}$$

■ **Graphics.** Let's plot the graph of the gaussian function:

```
In[11]:= Plot[gauss[x,0,1], {x,-5, 5}];
```



- **Lists.** In this course, we are going to do a lot of work with lists, in particular with vectors and matrices to represent images. (Sometimes we'll use vectors to represent images and matrices transformations on those images. Other times, we'll use matrices to represent images). Here is a four-dimensional vector which we'll call \mathbf{x} . \mathbf{x} could represent the pixel intensities in an image, or the input signal strengths at 4 receptors:

```
In[12]:= x = {2,3,0,1};
```

By ending a line with a semi-colon, you suppress the output after hitting the return key. Now let's make another vector, this one will be a list of "weights", say, representing the efficiency with which the inputs are weighted before being summed in the visual receptive field:

```
In[13]:= w = {2,1,-2,3};
```

Now the output of a model neuron that simply takes a weighted sum of the inputs is just the dot product of the input with the weights:

```
In[14]:= y = w.x
```

```
Out[14]= 10
```

This kind of operation is sometimes referred to as a "cross-correlator". It takes a signal x , and cross-correlates it with a template, w . In your homework you calculate the performance of a cross-correlator. A cross-correlator is the simplest model of what is called the receptive field of a neuron. But more on this later.

- **Using Tables to make Lists.** You can make a list whose elements are the squares of the element location.

```
In[15]:= s = Table[x^2,{x, 1, 8}]
```

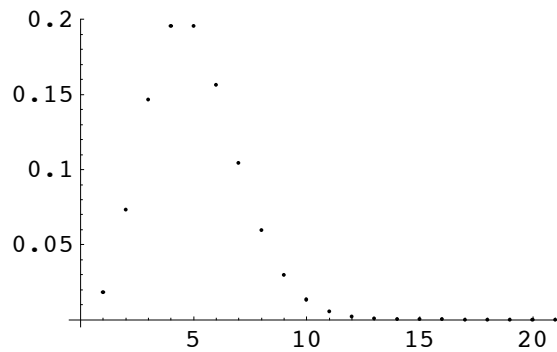
```
Out[15]= {1, 4, 9, 16, 25, 36, 49, 64}
```

```
In[16]:= poisson[x_,a_] := Exp[-a] a^x / Factorial[x]  
poisson[x,a]
```

```
Out[17]=  $\left\{ \frac{1}{2} a^2 e^{-a}, \frac{1}{6} a^3 e^{-a}, e^{-a}, a e^{-a} \right\}$ 
```

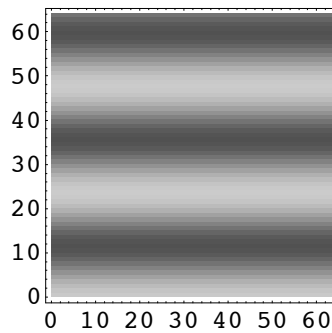
To make a one-dimensional list, you can use **Table[]**. We will use **Table[]** quite a bit, because it provides a compact way to iterate function values into an array. To see the result, we use **ListPlot**:

```
In[18]:= p1 = Table[N[poisson[x,4]], {x,0,20}];  
ListPlot[p1];
```



Later, we will use the add-in function, `BarChart[]`. You can also use `Table[]` to make a list of lists, e.g. to define a 2D image. Here is a 64x64 matrix that we can plot up as a Density Plot:

```
In[20]:= A = Table[Cos[2 Pi (1/24) x], {x,1,64}, {y,1,64}];  
ListDensityPlot[A,Mesh->False, PlotRange->{-2,2}];
```



- Below we are going to use the following functions:

Built-in Functions: Table, Range, Map, Count, ListPlot, Show.

We will also use Function[], but in short-hand form.

Add-on functions: BarChart, PoissonDistribution

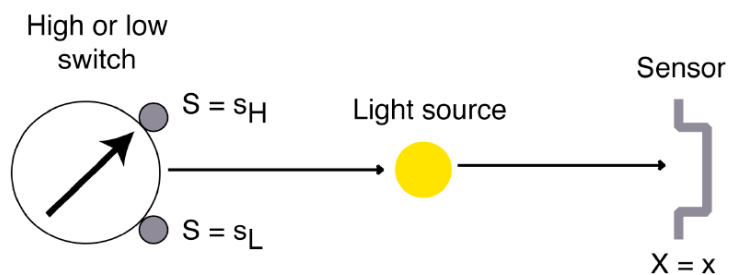
Probability concepts

In the next lecture, we will systematically go over the elements of probability and inference in detail. But today, we will take an intuitive and inductive approach, letting the light discrimination problem motivate the need for the concepts we need. We will talk about: **Random variable**, **Probability distribution**, **Conditional probability**, **Prior probability**, **Posterior probability**, and let the problem context help to develop an understanding.

Modeling external variability: The Ideal Observer

SDT Poisson signal discrimination example

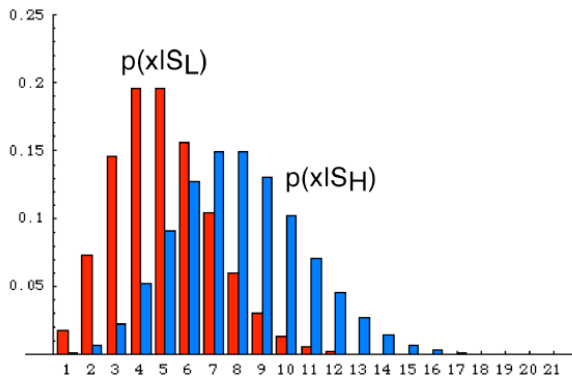
- Schematic of set-up for ideal photon counter



■ Histograms give an estimate of the conditional probabilities

We are going to use *Mathematica* to study the properties of photon counts. Imagine first that we do an experiment described at the end of the last lecture.

First set the light level to S_L . Now we begin to send flashes by opening and closing the shutter. No matter how hard we try to make the apparatus perfect, we would discover that the photon counter doesn't always record the same number of photons for each flash. Then we do the same for the switch setting, S_H . So, we compile a histogram of photon counts and find something like this:



The main observation is: photon detection is inherently statistical, and in fact because of the independence of photon absorption, the histogram can be modeled as a *Poisson distribution*. As the number of samples (i.e. test flashes in our case) grows, the better the model fits the histogram data. The Poisson distribution is characterized by the mean (or expected) number of photons flashed when the switch is set to S_L . The spread in the distribution (measured by the standard deviation) is equal to the square root of the mean. Specifically, the probability of k photons being detected for a Poisson distribution is given by:

```
In[22]:= poisson[x_,a_] := Exp[-a] a^x / Factorial[x]
poisson[k,a]

Out[23]=  $\frac{a^k e^{-a}}{k!}$ 
```

with a mean level and variance of a . The standard deviation is the square root of the variance. (There are many standard statistical distributions that describe various kinds of uncertainty. Below we read in one of the Standard Packages that has a set of routines for defining and sampling from Poisson distributions.

In the next lecture, we will approximate the Poisson distribution using the normal or Gaussian distribution developed by de Moivre, Laplace, and Gauss. For various reasons, the Gaussian distribution is a fundamental probability density...not to mention that it is the function that used to be on the German 10 mark note, before the advent of the euro.)

Let's simulate the process of photon generation and do some histogram counting using *Mathematica*.

The generative model for light intensity differences: What is the photon count?

Given the switch setting, what measurements result?

■ Experimental -- Simulated photon counts, and histograms corresponding to high and low switch settings:

We compile a histogram of photon counts conditional on the switch setting. Define high and low models:

```
In[24]:= highpdist = PoissonDistribution[7];
lowpdist = PoissonDistribution[4];
nsamples = 1000;
```

Draw nsamples (e.g. = 1000 flashes) for each switch setting.

```
In[27]:= sample[ntimes_] :=
  Table[Random[lowpdist], {ntimes}];

z = sample[nsamples];
domain = Range[0,15];
lowFreq = N[Map[Count[z, #] &, domain] / Length[z]];
```

General::spell1 :

Possible spelling error: new symbol name "domain" is similar to existing symbol "Domain". More...

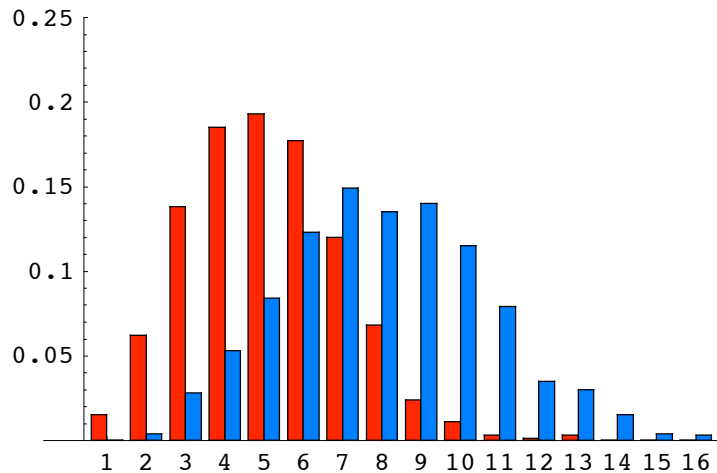
```
In[31]:= sample[ntimes_] :=
  Table[Random[highpdist], {ntimes}];

z = sample[nsamples];
domain = Range[0,15];
highFreq = N[Map[Count[z, #] &, domain] / Length[z]];
```

Use the Help Browser to look up the definitions of: **Count[]**, **Map[]**, and **Function[]**.

Plot up the histograms of the frequency that the various possible photon counts occur:

```
In[35]:= BarChart [lowFreq, highFreq, PlotRange -> {0, .25}];
```



■ Theoretical -- Compare Poisson probability distributions to high and low experimental histograms

We don't have to remember the formula for the Poisson distribution above. Instead, if the add-in library is read in (`<<-Statistics`DiscreteDistributions``), then `PoissonDistribution` is built in for us and can be used to define the probability distribution function (PDF):

```
In[36]:= ?? PoissonDistribution
```

`PoissonDistribution[mu]` represents the Poisson distribution with mean `mu`.

[More...](#)

`Attributes[PoissonDistribution] = {Protected, ReadProtected}`

```
In[37]:= PDF [PoissonDistribution [u], x]
```

```
Out[37]= PDF(PoissonDistribution(u), {2, 3, 0, 1})
```

This is the same formula we wrote down earlier. In standard probability notation, if the mean value is a , then the probability of random variable X taking on the value of x photons is:

$$p(X=x \text{ photons}) = \frac{a^x e^{-a}}{x!}$$

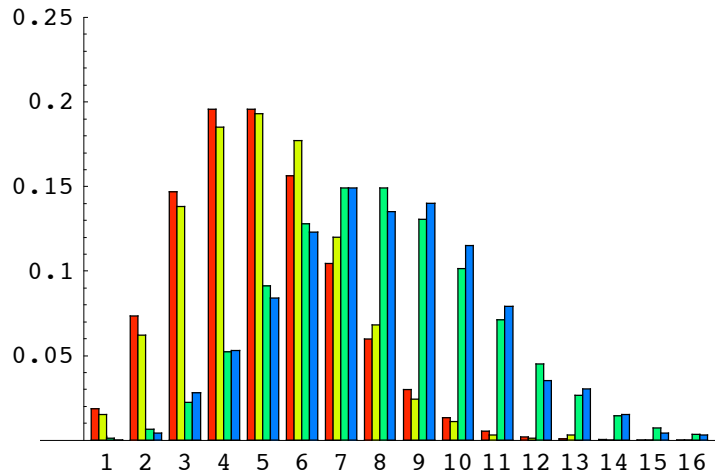
The mean or expectation of X is $E(X) = a$, and for a Poisson distribution, the variance, $\text{var}(X) = E[(X - a)^2]$ is also equal to a . Recall that the mean gives a measure of central tendency, and the variance is measure of spread of the distribution. (Note that earlier we wrote our own poisson function simply as: `poisson[x,a_] := Exp[-a] a^x / Factorial[x]`);

Then we make two lists corresponding to means of 4 and 7:

```
In[38]:= theorylowFreq = Table[N[PDF[PoissonDistribution[4], x]], {x, 0, 15}];
theoryhighFreq = Table[N[PDF[PoissonDistribution[7], x]], {x, 0, 15}];
```

Compare simulated photon count probabilities with theory predictions. What does the plot look like if nsamples is much smaller, say nsamples = 50?

```
In[40]:= BarChart [theorylowFreq , lowFreq , theoryhighFreq , highFreq,
  PlotRange -> {0, .25}];
```



Note that the more samples we draw, the better the experimental histograms match the theoretical.

The main point is: photon production is inherently statistical, and in fact because of the independence of photon absorption, the histogram for ideal photon counting can be modeled as a Poisson distribution.

■ Conditional probabilities

$p(x|S_L)$, $p(x|S_H)$ are conditional probabilities. I.e. they represent the probability of observing a photon count of x , given (or "conditional on") the switch setting $H = \{S_L \text{ or } S_H\}$.

A note on terminology: $p(x|S_L)$ is the probability of x conditional on S_L . But below when we consider optimal inference, we will talk about likelihood functions, where $p(x|S_L)$ is the *likelihood* of S_L given a measurement of x .

The knowledge from a generative model gets embodied in the conditional probability--given a cause H (where $H = S_L$ or S_H), the conditional probability tells us the probability of a measurement x .

Inference model for light intensity discrimination: How does the ideal observer decide the switch setting?

■ Yes/no task

In a yes/no task, we designate one switch position as the "signal" and the other as "not the signal", or "noise". We'll have the observer say "yes" when the switch is set to high.

■ Posterior probability distribution

Given a measurement, what was the switch setting? We've seen that because of variability in the measurement, it is impossible to answer this question correctly every time just from the data. However, we can ask how the decision should be made so that we are correct as often as possible. More precisely, we can ask:

What decision strategy will minimize the average probability of error?

The observer that achieves a pre-defined optimality criterion like this is called an "*ideal observer*". We'll see other ideal criteria later. Intuitively one might guess (correctly) that, given a photon measurement count, the ideal observer would pick the switch setting that has the biggest probability of the two. In other words:

If $P(S_L | x) > P(S_H | x)$, choose S_L

otherwise, choose S_H .

$P(S_L | x)$ is "the probability of the switch taking on the value of S_L , conditional on a measurement x ", and similarly for $P(S_H | x)$. $P(S_L | x)$ and $P(S_H | x)$ are called *posterior* probabilities. Why "posterior"? Even without any measurements, $P(S_L)$ and $P(S_H)$ provides important information for optimal decisions, and are called prior probabilities, i.e. the information *prior* to the arrival of the data measurements.

For the problems of visual inference, it is hard to directly specify the posterior probability, and it is easier to model the prior probabilities and the likelihoods, and to use Bayes rule to relate them to each other:

$$p(H | x) = \frac{p(x | H) p(H)}{p(x)}$$

Optimal performance is determined by the posterior probability, and that is why Bayes' rule is of central importance in determining optimal theories of inference. Let's consider the roles the priors and likelihoods play separately.

■ Prior only

Suppose the light transmission switch is set to S_L 4/5 of the time, and to S_H 1/5 of the time.

```
In[41]:= pH = { 4 / 5, 1 / 5 };
```

But the photon counter isn't working at all--so there are no data. What is the observer's optimal strategy to guess whether it was the high or bright flash? First we have to specify the receiver's goal, and then it makes sense to answer this question. If the receiver wants to minimize the average probability of error, then it should always pick S_L .

In maximum a posteriori estimation, the rule is the minimum error rule:

choose S_H if $p(S_H) > p(S_L)$

choose S_L if $p(S_L) > p(S_H)$

So if $p(S_H) > p(S_L)$, the rule says to always choose S_H . Interesting enough, people usually don't do this. With feedback, they tend to match probabilities.

Following the above rule, S_H is presented $p(S_H)$ fraction of the time--so this is exactly the error rate, i.e. the fraction of the time that the observer gets the wrong answer using the minimum error rule.

Exercise: What is the probability of error if the probabilities are matched?

■ Likelihood

We use *Mathematica* to define $p(x|H) = \text{poisson}[x,a]$, where the hypothesis or switch setting H , determines the average number of samples, x :

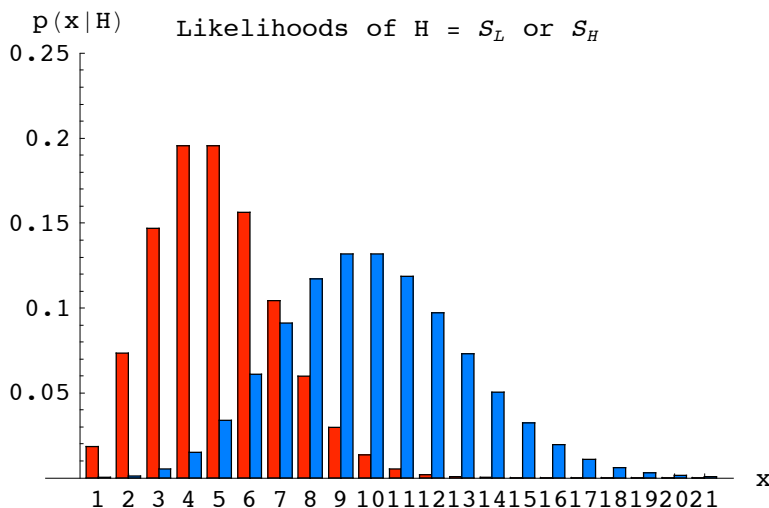
```
In[42]:= poisson[x_,a_] := Exp[-a] a^x / Factorial[x]
```

Then we make two lists of probabilities corresponding to means of 4 and 9:

```
In[43]:= p1 = Table[N[poisson[x, 4]], {x, 0, 20}];
p2 = Table[N[poisson[x, 9]], {x, 0, 20}];
```

Use BarChart again in the Graphics package to check the distributions.

```
In[45]:= BarChart[p1, p2, PlotRange -> {0, .25}, AxesLabel -> {"x", "p(x|H)"},
PlotLabel -> "Likelihoods of H = S_L or S_H"];
```



Note that the plot shows the frequencies of observing x photons under the two possible switch settings, **low** and **high**.

Let's put the two likelihoods together into one 2x21 array:

```
In[46]:= poispxH = {p1, p2};
```

Choosing the hypothesis with the largest likelihood will minimize the probability of error when the prior probabilities are uniform—i.e. maximum likelihood decisions are the same as maximum a posteriori decisions in this case. But what if the priors are not uniform (i.e. the same constant probability for all outcomes)?

■ Combining the prior and likelihood: Maximum a posteriori (MAP) rule

Suppose (as we assumed for the light discrimination example) that we know the likelihoods: $p(x|S_L)$ and $p(x|S_H)$. Bayes' rule tells us how to combine them to obtain the a posteriori probability of the hypotheses conditional on the data:

$$p(H|x) = \frac{p(x|H)p(H)}{p(x)} = \frac{p(x|H)p(H)}{\sum_H p(x|H)p(H)}$$

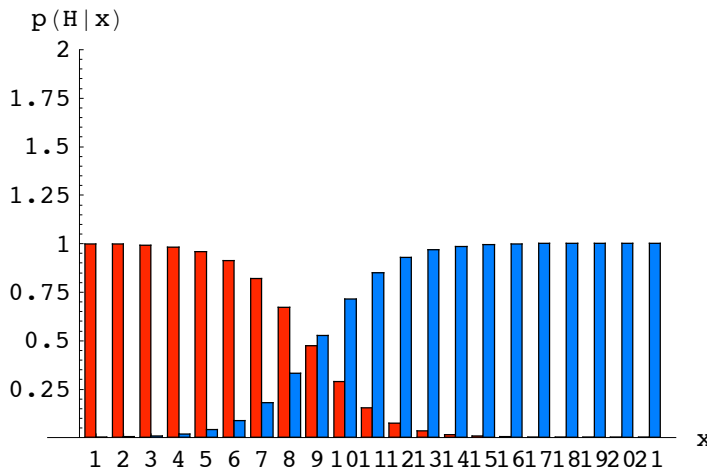
(We'll go over this and the basic rules of probability in the next lecture.)

We can calculate the posterior using the following, somewhat opaque, line of *Mathematica*:

```
In[47]:= posterior = Transpose [Transpose [ (poispXH pH) ] / Plus @@ (poispXH pH) ] ;
```

Below is a plot of the posterior probabilities, $p(S_H|x)$ and $p(S_L|x)$, as function of the data x :

```
In[48]:= BarChart [posterior [[1]], posterior [[2]], PlotRange -> {0, 2},
  AxesLabel -> {"x", "p(H|x)"} ;
```



Note that the plot shows the probability of the switch being **low** or **high**, given that x photons were counted. (Why does the posterior plot look so different from the likelihood plot?)

■ Maximum a posteriori rule (MAP)

For maximum a posteriori estimation, the rule is:

choose S_H if $p(S_H|x) > p(S_L|x)$

choose S_L if $p(S_L|x) > p(S_H|x)$

More generally, a function which chooses the value of argument ($H = S_L$ or S_H) that maximizes a function is sometimes written as:

$$\operatorname{argmax}_H p(H|x)$$

Suppose 8 photons were counted, the MAP rules says to pick "low", because the red bar is higher than the blue bar. Compare this plot with: $p_H = \{1/2, 1/2\}$.

■ Maximum likelihood rule

In maximum likelihood estimation, the rule is:

choose S_H if $p(x | S_H) > p(x | S_L)$

choose S_L if $p(x | S_L) > p(x | S_H)$

Because of Bayes rule, if the prior probabilities are equal, then MAP is equivalent to maximum likelihood.

$p(x | S_H)/p(x | S_L)$ is called the likelihood ratio.

Suppose 8 photons were counted, what is the maximum likelihood decision? (See $p(x|H)$ plots above, or recompute $p(H|x)$ using $p_H = \{1/2, 1/2\}$.)

■ Simple OPTIMAL Rule for light discrimination: Compare the photon count to a criterion.

Let's assume the priors are equal, so we base our decisions using the maximum likelihood rule. We will use *Mathematica* to do some symbolic manipulations to find the transition point where the likelihood goes from favoring a low to a high switch setting. This is where the likelihood ratio is 1, or the log of the ratio is zero.

```
In[49]:= loglikelihoodratio = Log[poisson[x, b] / poisson[x, d]]
```

```
Out[49]:= {log(b^2 e^{d-b} / d^2), log(b^3 e^{d-b} / d^3), log(e^{d-b}), log(b e^{d-b} / d)}
```

```
In[50]:= eologlikelihoodratio = PowerExpand[loglikelihoodratio]
```

```
General::spell1 : Possible spelling error: new symbol  
name "eologlikelihoodratio" is similar to existing symbol "loglikelihoodratio". More...
```

```
Out[50]:= {-b + d + 2 log(b) - 2 log(d), -b + d + 3 log(b) - 3 log(d), d - b, -b + d + log(b) - log(d)}
```

```
In[51]:= Solve[eologlikelihoodratio == 0, x]
```

```
General::ivar : 1 is not a valid variable. More...
```

```
General::ivar : 1 is not a valid variable. More...
```

```
Out[51]:= Solve[{-b + d + 2 log(b) - 2 log(d), -b + d + 3 log(b) - 3 log(d), d - b, -b + d + log(b) - log(d)} == 0, {2, 3, 0, 1}]
```

If d and b are the average number of photons for the low and high switch settings, and the photon counts are distributed according to a Poisson distribution, then minimal error will result on average with the following rule:

Say "high" if

$$x > \frac{b-d}{\text{Log}[b] - \text{Log}[d]}$$

Say "low" otherwise. The photon count, x , is said to be the *decision variable*. Because it is monotonically related to the likelihood ratio, it is an optimal decision variable. The transition value is called the *decision criterion*:

$$\text{decision criterion} = \frac{b-d}{\text{Log}[b] - \text{Log}[d]}$$

So now you have all you need to know to write a program that would make optimal decisions about high vs. low--it would simply count photons, and base its decision on whether the count was greater or less than the decision criterion.

■ Summarizing performance

So far so good. But we would like to go a littler farther so that we can calculate theoretically what the average performance of an ideal observer would be. In a yes/no task, an ideal observer can make two kinds of mistakes. It can decide the signal was there when it wasn't (a false alarm or false positive), or it can decide it wasn't there when it was (a miss). These two determine the error rate. An ideal can also be correct in two ways. It can have a hit or a correct rejection. We'll see later that you only need to measure two of these four statistics, because the hit + miss rate have to add up to 100% (if the signal is presented 100 times, you either say yes or no, so the sum of the hit and misses has to be 100).

In the next section you can try competing against the ideal, and have the opportunity to calculate hit and false alarm rates.

Try your luck against the Ideal Observer: Yes/no task

■ Set up the mini-experiment

Let's pretend that photons are like dots. (In 1977, Barlow published a paper which did just that. He pointed out that humans can efficiently, but not perfectly, discriminate between patterns of differing dot density).

As above, let's define a Poisson distribution with a mean of **mean**, with a function to draw a sample from this distribution. Let **highmean** = 220 for the "high" setting, and **darkmean** = 200 for the "low" setting. Then let's initialize an array, **data**, to accumulate the human, ideal, and true answers, and the corresponding hit, false alarm, correct rejection (cr), and miss events.

```
In[177]:= numberofphotons [mean_] := Random[PoissonDistribution [mean] ] ;
highmean = 220; darkmean = 200;
criterion = N [ -  $\frac{\text{highmean} + \text{darkmean}}{\text{Log}[\text{highmean}] - \text{Log}[\text{darkmean}]}$  ] ;
data = { {"myanswer", "idealanswer", "trueanswer", "ideal hit?",
  "my hit?", "ideal false alarm?", "my false alarm?", "ideal cr?",
  "my cr?", "ideal miss?", "my miss?" } } ;
```


■ Execute a trial

Now, randomly turn the switch to "high" or "low", draw a sample, and then input your response (1 for "high" and 0 for "low"). Execute the next cell 100 times.

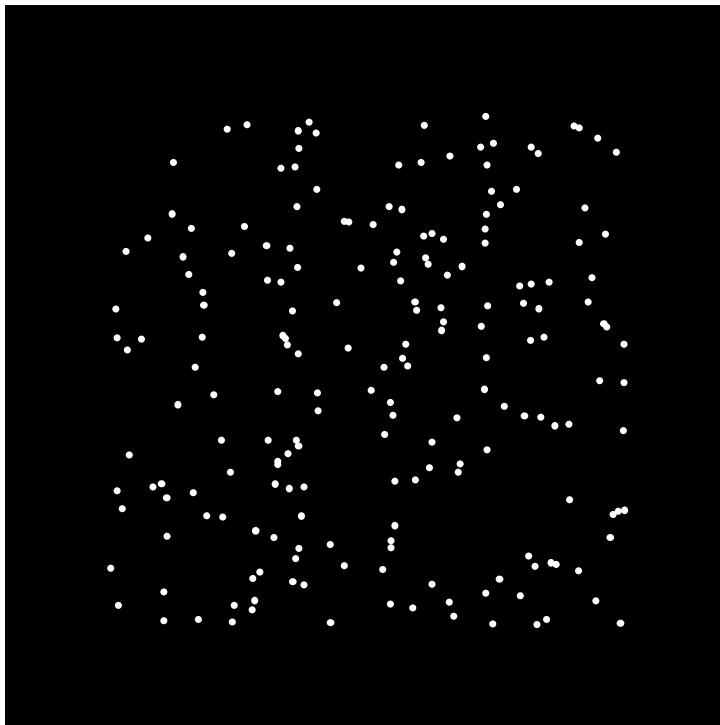
```
In[281]:= whichflash = Random[Integer, {0, 1}];
If[whichflash == 1, numsample = numberofphotons[highmean],
   numsample = numberofphotons[darkmean]];
sample = Table[{Random[], Random[]}, {numsample}];

sampleg = Graphics[{PointSize[0.01], Point/@sample}, AspectRatio → 1,
  Frame → False, FrameTicks → None, Background → GrayLevel[0.0],
  PlotRange → {{-.2, 1.2}, {-.2, 1.2}}];

Show[GraphicsArray[{sampleg}, GraphicsSpacing → .3], Frame → False];

myanswer = Input["High?"];
idealanswer = If[numsample > criterion, 1, 0];
trueanswer = whichflash;

data = Append[data, {myanswer, idealanswer, trueanswer,
  If[(idealanswer == 1) && (trueanswer == 1), 1, ""],
  If[(myanswer == 1) && (trueanswer == 1), 1, ""],
  If[(idealanswer == 1) && (trueanswer == 0), 1, ""],
  If[(myanswer == 1) && (trueanswer == 0), 1, ""],
  If[(idealanswer == 0) && (trueanswer == 0), 1, ""],
  If[(myanswer == 0) && (trueanswer == 0), 1, ""],
  If[(idealanswer == 0) && (trueanswer == 1), 1, ""],
  If[(myanswer == 0) && (trueanswer == 1), 1, ""]}];
```



■ Display the data

```
In[290]:= data // MatrixForm
```

```
Out[290]//MatrixForm=
```

```
(myanswer idealanswer trueanswer ideal hit? my hit? ideal false alarm? my false alarm? ideal cr? my c
  1 1 0 1 1 1 1 1 1
  0 0 0 1 1 1 1 1 1
  1 0 0 1 1 1 1 1 1
  1 0 0 1 1 1 1 1 1
  1 1 1 1 1 1 1 1 1
  0 1 0 1 1 1 1 1 1
  0 0 0 1 1 1 1 1 1
  1 1 0 1 1 1 1 1 1
  1 1 1 1 1 1 1 1 1
  1 1 1 1 1 1 1 1 1
  0 0 0 1 1 1 1 1 1
  0 0 1 1 1 1 1 1 1)
```

■ Analyze the data

Let's drop the table heading stored in row 1, and then transpose the matrix so that the columns become the rows:

```
In[291]:= data2 = Transpose [Drop [data, 1] ]
```

```
Out[291]=
```

```
(1 0 1 1 1 0 0 1 1 1 0 0
 1 0 0 0 1 1 0 1 1 1 0 0
 0 0 0 0 1 0 0 0 1 1 0 1
 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1)
```

Let's use a combination of Map[] and Count[] (used earlier to make histograms) to count up all occurrences of an event type. So the total for myhits is:

```
In[292]:= Map[Count[data2[[5]], #] &, {1}]
```

```
Out[292]= {3}
```

The total times the signal was presented (i.e. switch set to high) is:

```
In[293]:= Map[Count[data2[[3]], #] &, {1}]
```

```
Out[293]= {4}
```

So what is my hit rate?

Next time

In the next lecture, we are going to study hit and false alarm statistics using the Gaussian distribution, rather than the Poisson. The Gaussian approximation has the advantage of theoretical convenience and greater generality in extending to other types of signal discrimination and detection problems.

Modeling internal variability of observer: The Human

We'll see how to calculate the signal-to-noise ratio (d') for a discrimination or detection task in terms of hit and false alarm rates, and other measures.

You've probably noted above that when you are a subject in a yes/no task it is hard to figure out the criterion. Feedback helps a human observer learn, but we'll see that a better way of measuring performance is to use the two-alternative forced-choice (2AFC) task.

Compare ideal and human performance in terms of *statistical efficiency*.

References

- Applebaum, D. (1996). Probability and Information . Cambridge, UK: Cambridge University Press.
- Barlow, H. B. (1962). A method of determining the overall quantum efficiency of visual discriminations. Journal of Physiology (London), 160, 155-168.
- Barlow, H. B. (1977). Retinal and central factors in human vision limited by noise. In B. H. B., & F. P. (Ed.), Photoreception in Vertebrates Academic Press.
- Duda, R. O., & Hart, P. E. (1973). Pattern classification and scene analysis . New York.: John Wiley & Sons.
- Geisler, W. (1989). Sequential Ideal-Observer analysis of visual discriminations. Psychological Review, 96(2), 267-314.
- Green, D. M., & Swets, J. A. (1974). Signal Detection Theory and Psychophysics . Huntington, New York: Robert E. Krieger Publishing Company.
- Van Trees, H. L. (1968). Detection, Estimation and Modulation Theory . New York: John Wiley and Sons.
- Watson, A. B., & Pelli, D. G. (1983). QUEST: A Bayesian adaptive psychometric method. 33, 113-120.