Examples of Maximum Likelihood Estimation (MLE)

Part A: Let's play a game. In this bag I have two coins: one is painted green, the other purple, and both are weighted funny. The green coin is biased heavily to land heads up, and will do so about 90% of the time. The purple coin is slightly weighted to land tails up, about 60% of flips. Both coins are otherwise identical. In this game, I'll pull a coin out of the bag without looking, flip it in secret, and tell you what landed up, either heads or tails. To win this game, you have to guess which color of coin I picked out of the bag.

At first glance this game may seem to be "a coin toss" (pun!), with a chance of guessing the color right 50% of the time. But it turns out to be easier than that, since we know that each coin behaves differently. Suppose I tell you that I flipped a tails. Well, we know the green coin hits tails only 10% of flips, while the purple coin could hit tails 60% of flips. i.e.,

> Probability of tails given coin is green is 10%,
> Probability of tails given coin is purple is 60%.

Now since we know the probabilities of tails conditioned on which coin is drawn, we know the likelihoods:

> Likelihood of having a green coin given tails was flipped is 10%,
> Likelihood of having a purple coin given tails was flipped is 60%.

When I tell you that a tails was flipped, you can infer that it is 6 times more likely that the coin I drew was a purple coin. So if you wanted to maximize your chances of winning the game, you should choose the coin color which maximizes your likelihood of wining, given the information you have about what was flipped.

That is, if you see tails, choose purple; if you see heads, choose green.

THIS IS EXACTLY MAXIMUM LIKELIHOOD DECODING! It is nothing more than: you choose the option that is most likely to be true given your observation. Here, the observation is a heads or a tails (you could say, a 1 or a 0) and you are choosing a color. In neuroscience, it is a spike count (a number) and you are choosing a stimulus value.

Now, what's the fraction of correct responses you will deliver when being given an observed responses, using maximum likelihood decoding?

To figure this out, consider the two possibilities.

Say the truth is that a green coin was flipped. How often will you get it right? 90% of the time a heads will occur, and according to your maximum likelihood decoding rule you will say "green," so be correct. So 90% of the time when green is flipped, you're correct.

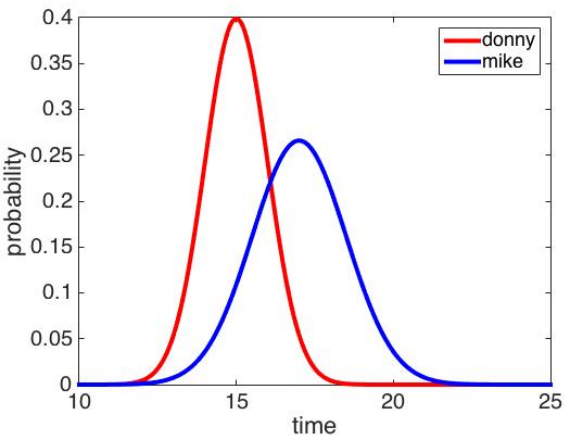
Say the truth is that a purple coin was flipped. How often will you get it right? 60% of the time you'll get a tails, and say "purple," so be correct. So 60% of the time when green is flipped, you're correct.

What's your error rate on balance? We are assuming each coin in truth is flipped an equal number of times. So your fraction of correct responses is the average of these numbers, 75%.

Cool! Maximum likelihood decoding sure beats guessing.

THIS IS EXACTLY THE PROCEDURE YOU WILL FOLLOW IN GENERAL to figure out your fraction of correct responses. The only difference in the neural case is that there are more than two possible observations (heads and tails), instead integer-valued spike counts. But you do the same thing. You have your maximum likelihood decoding rule in hand. You assume one of the options (i.e., one stimulus value) is true, and see what percentage of the time your rule gives you the right answer. Then you assume the other option is true, and see what percentage of the time your rule gives you the right answer. You average these percentages, and that is your fraction of correct responses.

Part B: Let's play another game. In this game we have two world class sprinters running the 150m dash: Donovan Bailey, and Michael Johnson. Each runner has a normal (Gaussian) distribution for their finishing times: Donovan has a mean of 15 seconds with a standard deviation of 1second, Michael has a mean of 17 seconds with a standard deviation of 1.5 seconds. In this game, I'll tell you the finishing time of one of the runners, and you win if you guess who ran that time correctly. To get started, you could use MATLAB to plot the finishing time distributions of each runner, and get the plot on the right, via this code: (using the gaussian formula from class together with linspace, xlabel, ylabel, legend commands:)




```
xlist=linspace(10,25,1000);

m1=15
s1=1
m2=17
s2=1.5

gsn1=1/(sqrt(2*pi*s1^2))*exp(-(xlist-m1).^2/(2*s1^2));
gsn2=1/(sqrt(2*pi*s2^2))*exp(-(xlist-m2).^2/(2*s2^2));


%makes text bigger in upcoming plot labels
set(0,'defaultaxesfontsize',20);
set(0,'defaulttextfontsize',20);

figure
plot(xlist,gsn1,'r','LineWidth',4)
hold on
plot(xlist,gsn2,'b','LineWidth',4)
legend('donny','mike')
xlabel('time'); ylabel('probability')
```

Suppose I give you a running time of 21.5 seconds. While it's a slow time for the American Johnson, the probability of Canadian Bailey running over 20 seconds is extremely small. Thus if you'd want to win, you're better off to say that Michael Johnson ran the 21.5.

ONCE AGAIN, THIS IS MAXIMUM LIKELIHOOD DECODING! You are given an observation (here, a time), and you choose the option that was most likely to produce that time (here, Johnson). You can then apply this method to determine the best guess to win this game given any running time: it's the guess corresponding to the higher likelihood (i.e., maximum likelihood).

To compute the fraction of correct responses, you'd follow the procedure above. That would involve summing (integrating) the area under the histograms that correspond to correct choices. For histograms, as on the assignment, it's easier: you sum up the probabilities in the corresponding bins.

Part C: A big part of maximum likelihood estimation involves working with data and probability distributions. In practice, you will not have smooth curves like the above blue and redo ones, but will need to build your own histograms. Use the following code to generate and then then visualize some random data:

```
D = 2*randn(1,10)+6;
hist(D);
```

You'll see that there are only a few samples in the data, and your histogram is pretty sparse. You might not be able to estimate "overlap" points between this and another histogram very well, to see above and below what value you should choose an option based on this histogram.

Generate 1000 samples from D using the same formula, instead of 10, and plot the histogram. You should be able to get a better idea now. To visualize the 1000 samples more finely, increase the number of histogram bins to 100 with:

```
hist(D,100);
```

To get a good smooth histogram now, you might need to draw even more samples. YOU COULD RUN INTO THIS TYPE OF ISSUE ON YOUR ASSIGNMENT. MAKE SURE YOU ARE DOING ENOUGH SAMPLES TO GET SMOOTH HISTOGRAMS, JUST INCREASING NUMBER OF TRIALS.

Part D: OPTIONAL. In our HW problem, we can get around the issue of not having enough samples by just increasing the trial count, i.e., asking the computer for more! In real life, often we will be given a small set of data and will need to use another strategy to make informed guesses about where other samples came from. We'll discuss how to do this using mean and variance statistics (or the square root of variance, the standard deviation, often called std dev or std). You do not need to do this for the assignment, but it's worth knowing about for sure.

Suppose I have a machine that spits out two different kinds of white powder. One powder is the stuff that covers 'sourpatch kids' candy, is extremely delicious, and I want to eat it. The other powder is Ricin, a powerful neurotoxin. Based on experiments I have conducted with a population of undergrads, I know that the machine will distribute both powders with equal probability, and that the Ricin is less massive, having a lower weight. I have been able to find the weights in grams of 20 powder samples, below as matrix PD1:

```
PD1 = [2.7800    3.2000    3.0400    2.6900    2.9600    3.1300    2.7600    2.9800    2.4900    2.9200;
       1.8600    2.6900    2.3200    2.9100    2.0500    3.0700    2.2600    2.1000    2.1000    2.1200]
```

The rows represent different classes of powders, determined by how the undergrads responded to the powders. The columns are different sample numbers. First, determine which row corresponds to the Ricin, by finding which class has a lower weight. To do this, take the mean across the second (column) dimension:

```
mean(PD1,2)
```

Assign each row as either being one of candy or poison based on the average weight. Now get an idea for the distribution of mass within each group:

```
std(PD1, 0, 2)
```

Note that the 0 argument is necessary, due to an optional flag used in the definition of std which is not important here (see doc std) for details if you're interested. We see that each row has a different variance, but it is difficult to see how these relate to the means. First we can plot the means:

```
plot([1 2], mean(PD1,2), 'b-');
```

which is sort of hard to view, so plot it again replacing the default 'b-' with 'r.' or 'go' or 'mx:' and view the effects (google matlab linespec for further details). We can then incorporate our knowledge of the variance within each row by overlaying the standard deviation from each mean. First, run a 'doc errorbar' and read the options, then try:

```
errorbar([1 2], mean(PD1,2), std(PD1,0,2), 'ro', 'markerSize', 12)
```

The x-axis corresponds to the column number, and the y-axis to the mass in grams. The vertical lines at each datapoint correspond to the standard deviation about the mean of each data row. You can then imagine plotting two normal distributions, as in part B, with the mean/std you've calculated, and doing MLE as before. Here, we'll just use the errorbar plot to make some educated guesses.

> Ask yourself: say I have four samples, represented as ordered pairs of ID letters and weights: (A, 2.28), (B, 3.25), (C, 3.22) , (D, 2.65). Which sample would you say was likely poison? Which sample would you say is safe for human consumption? Between the two remaining samples, which would you wish to sample, and which would you offer to your friend/enemy, and why?