Doing Bayesians in MATLAB

In the previous lecture, we talked about how to use Bayes' Rule and the Bayesian statistics to model some aspects of human cognition. This tutorial aims to reinforce that by illustrating how this model can be implemented in MATLAB.

The core of almost all Bayesian models of cognition is the Bayes' Rule:

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

D refers to the observed data and h refers to a particular hypothesis.

I. Considering all possible hypotheses

In Bayesian models of cognition, the posterior probability $P(h|D)$ is often taken as a quantity to describe how humans perceive a certain hypothesis h to be true after some observations D. Therefore, the goal is to compute $P(h|D)$ for all possible hypotheses h. Recall the example used in the lecture, in which we only have to consider two possible hypotheses (i.e., "a fair coin" and "a trick coin that always gives heads"). In many other problems, however, there could be many more possible hypotheses, which complicates the issue.

Consider the problem of judging the fairness of a coin. Let's denote the quantity $P(Head)$ by θ, which tells you how "fair" a coin is. An absolutely fair coin is a coin with $\theta = 0.5$. Now, it seems that "fairness" is a continuum, because a coin with $\theta = 0.4$ seems to be less fair, but it's still "fairer" than coin with $\theta = 0$, which always gives you tail. Therefore, the number of possible hypotheses depends on θ, which could be any value from 0 to 1.

Theoretically, there are infinitely many possible θ values. It is usually very hard because, if θ is treated as continuous, you'll need to analytically compute the posterior distribution (trust me, you won't want to do it). **Fortunately, using MATLAB, we can slice up the continuous range into a finite number of possibilities.**
Within the [0, 1] range of θ, let's only focus on the following possibilities:

$$\theta = [0,\ 0.05,\ 0.10,\ 0.15,\ 0.20,\ 0.25,\ \ldots, 1];$$

We've learned how to do this in MATLAB. In fact, you may just type the above statement directly (with all numbers in) on MATLAB to define θ (we can just call the variable `theta`).

Alternatively, you can use the colon : operator to define this equally spaced vector from 0 to 1.

```
theta = 0 : 0.05 : 1;
```

We now have a set of possible hypotheses to start working with. Let's look at how to compute the two terms on the numerator of the right hand side of the Bayes' Rule, which are the likelihood term $P(D|h)$ and the prior term $P(h)$. **The ultimate goal is to find the hypothesis which gives you the maximum value of posterior $P(h|D)$.** This is left as an exercise at the end of this handout.

II. Computing the likelihood term $P(D|h)$ for each possible hypothesis

Given a fixed value of θ, we can compute the likelihood term $P(D|\theta)$ (I've changed the notation from h to θ here, because each value of θ essentially corresponds to one single hypothesis). Recall from the lecture that the probability of obtaining a *particular* sequence of N_H heads and N_T tails from a coin with $P(Head) = \theta$ is given by the formula

$$P(D|\theta) = \theta^{N_H}(1-\theta)^{N_T}$$

For example, suppose you've observed HHTHT after flipping the coin 5 times (i.e., D = {HHTHT}, $N_H = 3, N_T = 2$). Let's compute the likelihood of obtaining this flip sequence under the hypothesis of $\theta = 0.3$:

$$P(D|\theta) = P(\{HHTHT\}|\theta = 0.3) = (0.3)^3(1-0.3)^2 = 0.5170$$

It's easy to compute this in MATLAB: `(0.3^3)*((1-0.3)^2)`
But how about the hypothesis $\theta = 0.1$? And $\theta = 0.15$, $\theta = 0.2$, and others? You probably don't want to type the above line many times to compute the likelihood for each θ value. **Now, using MATLAB's element-by-element operator `.` (period), we can easily compute the likelihood for EACH value stored in a vector.** In fact, you can plug the vector `theta` in almost any arithmetic functions as if it was a single number. For example, to evaluate the likelihood $P(\{HHTHT\}|\theta)$ for the range of θ values we defined earlier:

```
Nh = 3;
Nt = 2;
likelihood =(theta.^Nh).*((1-theta).^Nt);
```

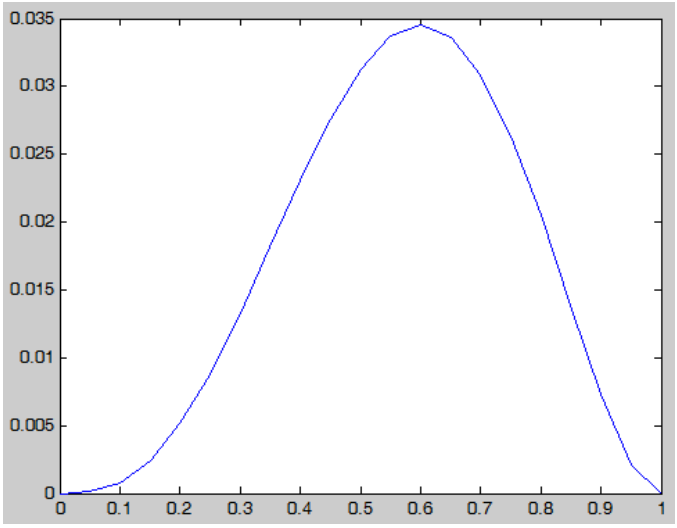
What I did here is simply replacing `0.3` in the earlier statement with `theta`, and make all multiplication and power operators element-by-element by adding `.` before them.
After doing this, you should obtain a new vector `likelihood`, which represents the likelihood for obtaining the observed data for each corresponding value in `theta`.
You may display the values of theta and likelihood side-by-side to take a look at them together:

```
[theta' likelihood']

ans =
         0         0
    0.0500    0.0001
    0.1000    0.0008
    0.1500    0.0024
    0.2000    0.0051
    0.2500    0.0088
     ...       ...
    0.9500    0.0021
    1.0000         0
```

(Note: Since both `theta` and `likelihood` are row vectors, the transpose `'` changes each of them into a column vector. Putting two column vectors with equal number of elements side by side within brackets [ ] concatenates them to form a two-column matrix).

It is always helpful to visualize the numbers in a graph than on a list of numbers. Here, an interesting plot will be to look at how `likelihood` (Y) varies as `theta` (X) varies:




```
plot(theta, likelihood);
```

From the plot, we can see that the θ value (on X) that gives you the maximum likelihood (on Y) is 0.6. **We say that the maximum likelihood estimate (MLE) of θ is 0.6, because, under the hypothesis of $\theta = 0.6$, the probability for us to observe HHTHT is the highest.**

However, if you have a complicated form of likelihood function and/or thousands of values of θ, this manual approach of finding maximum is not very reliable (actually not possible). Therefore, we may want to obtain the MLE more systematically instead of visually locating it from a graph because. Now, below is a very cool trick that you'll find useful in your future MATLAB endeavor.

The built-in function `max()` tells you *what* and *where* the largest number in a vector is:

```
>> [MaxL MaxID] = max(likelihood)

MaxL =

    0.0346   % the largest number in likelihood

MaxID =

     13      % the position of the largest number in likelihood
```

(Note: `MaxL` and `MaxID` are not keywords. Naming is arbitrary here)

This tells us that the maximum number in `likelihood` is 0.0346, which is the 13th number. Now, because each value in `theta` corresponds to one value in `likelihood` based on position, the θ value that gives the maximum likelihood is also at the 13th position in `theta`. Therefore, to obtain the θ value that gives the maximum likelihood, we simply type:

```
>> theta(MaxID)
```

```
ans =

    0.6000     % consistent with our eyeballing from the graph
```

III. Computing the prior distribution $P(h)$ for possible hypothesis

The prior term $P(h)$ is the probability for a certain hypothesis (e.g., h: $\theta = 0.3$) to be true. It's often referred to as the "experience" or "knowledge" in the human mind. **It has nothing to do with the data observed, and should be determined *prior to* having any observations**.

We use a probability density function (pdf) to characterize the prior (since it is a probability distribution over the range of all possible hypotheses). The common practice in Bayesian modeling is that the modeler proposes a reasonable pdf (together with the necessary parameter values) for the prior based on previous statistics or based on other theories.

In our case of coin flipping, we assume that the observer has already seen a certain number of heads and tails before observing the data. The beta probability density function $Beta(\alpha, \beta)$ precisely describes this situation:

$$Beta(\theta;\ \alpha, \beta) = \frac{\theta^{\alpha-1}(1-\theta)^{\beta-1}}{B(\alpha, \beta)}$$

Here, α and β are the two parameters for the beta pdf (as in μ and σ^2 in a normal distribution, despite the difference in meanings). They specify, respectively, how many times the observer has seen heads and tails from coin flips prior to observing the data. $B(\alpha, \beta)$ is the beta function, which is quite painful to write out in full even in MATLAB (we'll use a built-in function for it).

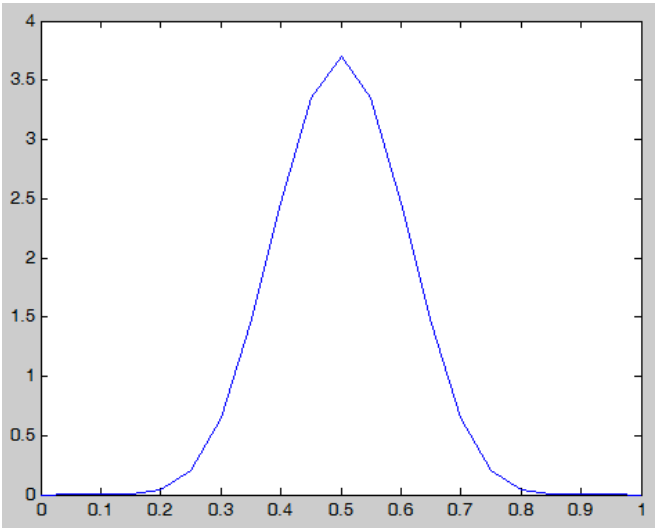
Formally, let's define V_H and V_T as the numbers of heads and tails experienced before observing the data, respectively. Then, $\alpha = V_H + 1,\ \beta = V_T + 1$.
For demonstration purpose, let's assume our Bayesian observer has experienced 10 heads ($V_H = 10$) and 10 tails ($V_T = 10$) before observing the HHTHT sequence. Then, we can choose the $Beta(11,11)$ pdf as our prior. Below is how you compute it in MATLAB:

```
Vh = 10;
Vt = 10;
aval = Vh+1;
bval = Vt+1;
prior = ((theta.^(aval-1)).*((1-theta).^(bval-1)))/beta(aval, bval);
```

This statement computes the probability density based on beta distribution with two parameters for each value in `theta`, and stores the answer in `prior`.

You may want to visualize the result in the same way as we did for the likelihoods:

```
plot(theta, prior);
```



From the graph, we can see that $\theta = 0.5$ gives the highest prior value. **This is reasonable because our Bayesian observer has experienced an equal number of heads (10) and tails (10) prior to observing the data.** Based on this knowledge, it believes most strongly that θ should be 0.5, although other possibilities (e.g., $\theta = 0.4$) are also probable.

One technical but important note is that prior is a probability distribution over all possible hypotheses (or the entire hypothesis space). **Based on one of the probability axioms, all prior probabilities should sum to 1.** Algebraically, the axiom requires

$$\int_0^1 P(\theta) d\theta = 1$$

We'd like to do a sanity check on whether the prior probabilities we computed earlier satisfy the above requirement. However, since we've sliced up the continuous range of θ values, we can't (or don't have to ☺) do the analytical integration here (which is to integrate the beta pdf!!).

With our discrete θ values, the above integral can be *approximated* by the Riemann sum:

$$\sum_{i=1}^{N} P(\theta_i)\, \delta \approx 1$$

Here, δ is our stepsize in `theta`, which is 0.05. Each value of $P(\theta_i)$ is simply stored in the vector `prior`. So, what we want to check is whether the above value is (approximately) equal to 1. For a quick and easy way to do summation across numbers in a vector, we can use the built-in `sum()` function (`sum(A)` returns the sum of all elements in vector `A`):

```
>> sum(prior*0.05)

ans =

    1.0000
```

Checked. We know that we're doing things right.

[In-class Exercise Q1, generate q1.m file]

IV. Finding the maximum a posteriori (MAP) estimate

Now, we have obtained the likelihood $P(D|\theta)$ and prior $P(\theta)$ for each value of θ. We should be able to obtain the maximum a posterior (MAP) estimate, which is the value of θ that gives you the maximum posterior probability.

If you look back to the Bayes' Rule, you may wonder why we skip the term $P(D)$. It is because $P(D)$ doesn't change for different values of θ, and is usually treated as a constant if the goal is simply to find the MAP estimate. In fact, in the context of finding the MAP estimate, the Bayes' Rule is often simplified as

$$P(\theta|D) \propto P(D|\theta)P(\theta)$$

Therefore, the θ value that gives you the maximum value for the product on the right-hand side should also give you the maximum value for the left-hand side. **Now, our goal becomes finding the θ value which gives the maximum product between likelihood and prior. This θ value is called the MAP estimate.**

Based on the steps and techniques we've covered so far, you should be able to do it yourself. The steps below may serve as some hints:

1) Multiply `likelihood` by `prior`, element-by-element, to obtain `post`
2) Find what and where the maximum value is in `post`
3) Retrieve the value in `theta` which has the same index as the value found in step 2)
4) Plot `post` against `theta`. Visually locate the MAP estimate. Use this answer to verify your answer in 3) (they should give you the same value).

[In-class Exercise Q2, generate q2.m file]

Write a script to include all the steps discussed in this handout to find the MAP estimate in the coin flipping problem. Define the following quantities as variables instead of hardcoding them as constants in your program:

- stepsize for `theta`

Now, try to play with this program by changing the values in the parameters. For example,

- Try out some extreme data (e.g., HHHHHH), and see how your Bayesian model infers the fairness of the coin.
- Try to give a very strong fair prior (e.g., $V_H = 1000, V_T = 1000$) or a weak one (e.g., $V_H = 2, V_T = 2$) and see how this is reflected in the "reasoning" of your model
- Try to rerun the model with the quantities defined in this handout, but with a smaller stepsize (e.g., 0.01). Do you get the same result? Think about the pros and cons of using different stepsizes, from the implementation perspective. Write down a short paragraph at the end of q2.m file to summarize your finding on the impact of stepsize on the simulation result.