Bayesian Inference

DS4S

Announcement

May 26	Data Scientist Panel (Vladimir Iglovikov, Vikram Vijayaraghavan, Galina Malovichko, and Marius Millea)

Image from the Canvas home page, where these are links to LinkedIn or similar

Story about the paper stolen from the printer

Models of Data

Simple example: modeling data from a measurement of length

- Model assumptions:
  - the object being measured has a time-invariant, true length.
  - each measurement has an additive contribution from a stochastic process, the error
  - the errors from one measurement to the next are independent. Each measurement error is the result of a Gaussian random process with zero mean and a fixed known variance

$$d_i = \ell_i + n_i$$

$$P(n_i) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{n_i^2}{2\sigma^2}\right)$$

How might we make this model more realistic?

How might we test some of this model's assumptions?

Bayesian Inference

Model Testing
(Model Comparison)

Parameter Estimation

- Bayesian inference is inference done with a secure, logical foundation. It is the gold standard.
- It is very widely used (science, engineering, medicine, sport, and law — see Wikipedia).
- It is often computationally demanding. Expanding computational power has led to its more widespread use.
- The concepts we present here are foundational important for machine learning.

Bayesian Inference

Model Testing
(Model Comparison)

Parameter Estimation

- Bayesian inference is inference done with a secure, logical foundation. It is the gold standard.
- It is very widely used (science, engineering, medicine, sport, and law — see Wikipedia).
- It is often computationally demanding. Expanding computational power has led to its more widespread use.
- The concepts we present here are foundational important for machine learning.

Parameter estimation

- We assume the model of the data is correct, and use the data to infer a *posterior probability distribution* for the parameters of the model.

For our example, we'll keep it real simple, assuming just one measurement, so

$$d = \ell + n$$

$$P(n) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{n^2}{2\sigma^2}\right)$$

We want to know $P(\ell|d, I)$

where I = "information" — in this case the assumptions of the model and the value of sigma

How to get $P(\ell|d, I)$?

Let's first work out $P(d|\ell, I)$

How do we get that from:

$$d = \ell + n \qquad P(n) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{n^2}{2\sigma^2}\right)$$

How to get $P(\ell|d, I)$?

It's easy to get $P(d|\ell, I)$

How do we get that from: $d = \ell + n$ $\quad P(n) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{n^2}{2\sigma^2}\right)$

If we are given \ell, then for a particular value of d we can determine the error, n so

$$P(d|\ell, I) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(d-\ell)^2}{2\sigma^2}\right)$$

But what we want is the reverse of this…

Bayes' Theorem

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Reverend Bayes
c. 1701 to 1761
England

Bayes' Theorem

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Reverend Bayes
c. 1701 to 1761
England

Derivation:

$$P(A, B) = P(A|B)P(B)$$
$$P(A, B) = B(B|A)P(A)$$

==> $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$

Applying Bayes' Theorem

"posterior" "likelihood" "prior"

$$P(\ell|d, I) = \frac{P(d|\ell, I)P(\ell|I)}{P(d|I)}$$

we can think of as a normalization constant (does not depend on \ell)

this is a model of learning from data
"prior" = what we knew before the data
"posterior" = what we know after the data

Another example dataset and model

Scolnic et al. (2018)

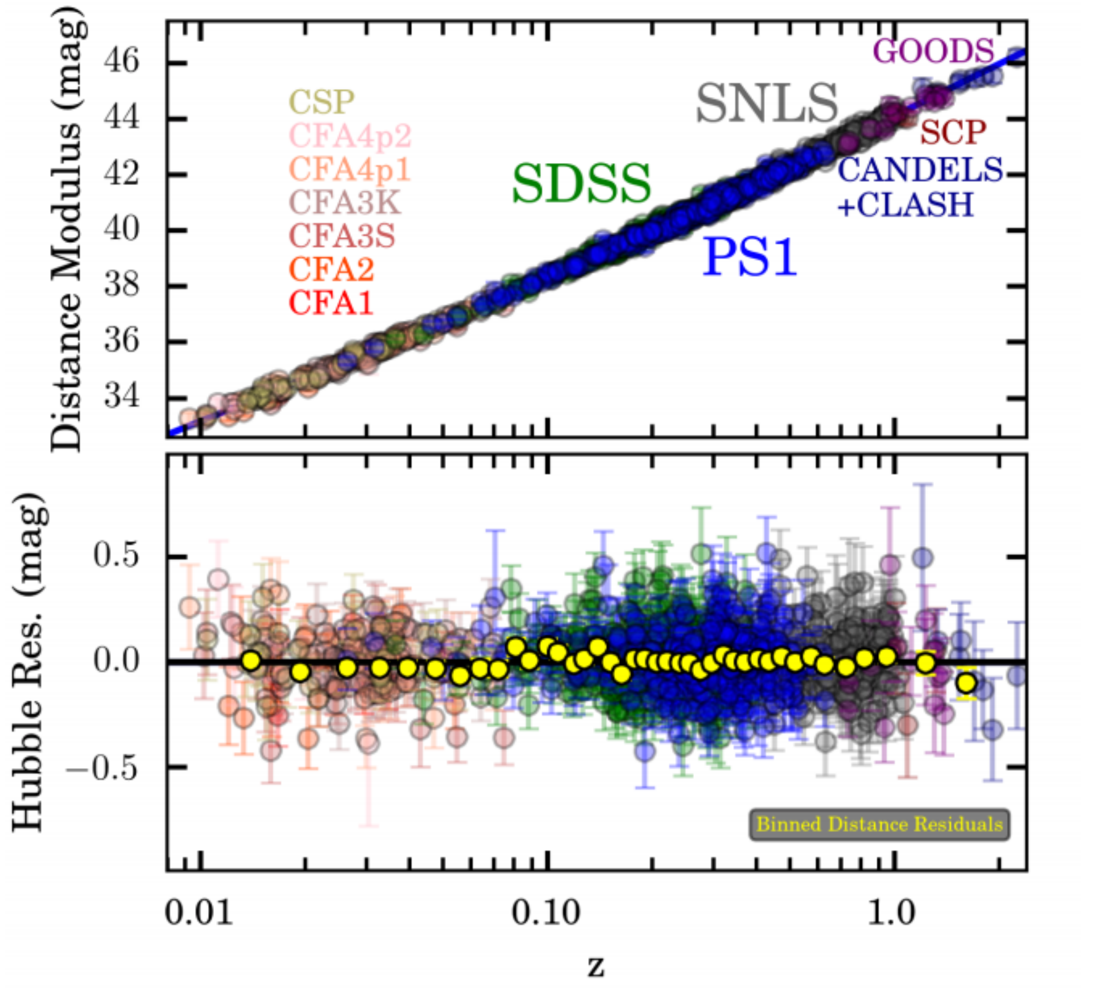


Figure 11. The Hubble diagram for the Pantheon sample. The top panel shows the distance modulus for each SN; the bottom panel shows residuals to the best fit cosmology. Distance modulus values are shown using G10 scatter model.

arXiv: 1710.00845

- Data:
  - Distance modulus (a measure of distance based on brightness of a "standard candle")
  - Redshift, z (1+z = observed wavelength/ emitted wavelength
- Model:
  - General Relativity
  - Homogeneity and Isotropy
  - Ingredients: non-relativistic matter, a cosmological constant, and mean curvature
  - Model of errors

We want to know the posterior probability density

$$P(\theta|d, I) \propto P(d|\theta, I)P(\theta)$$

where $\theta = \{\theta_1, \theta_2, ...\theta_n\}$

is the set of model parameters

in our case n = 3 and they could be

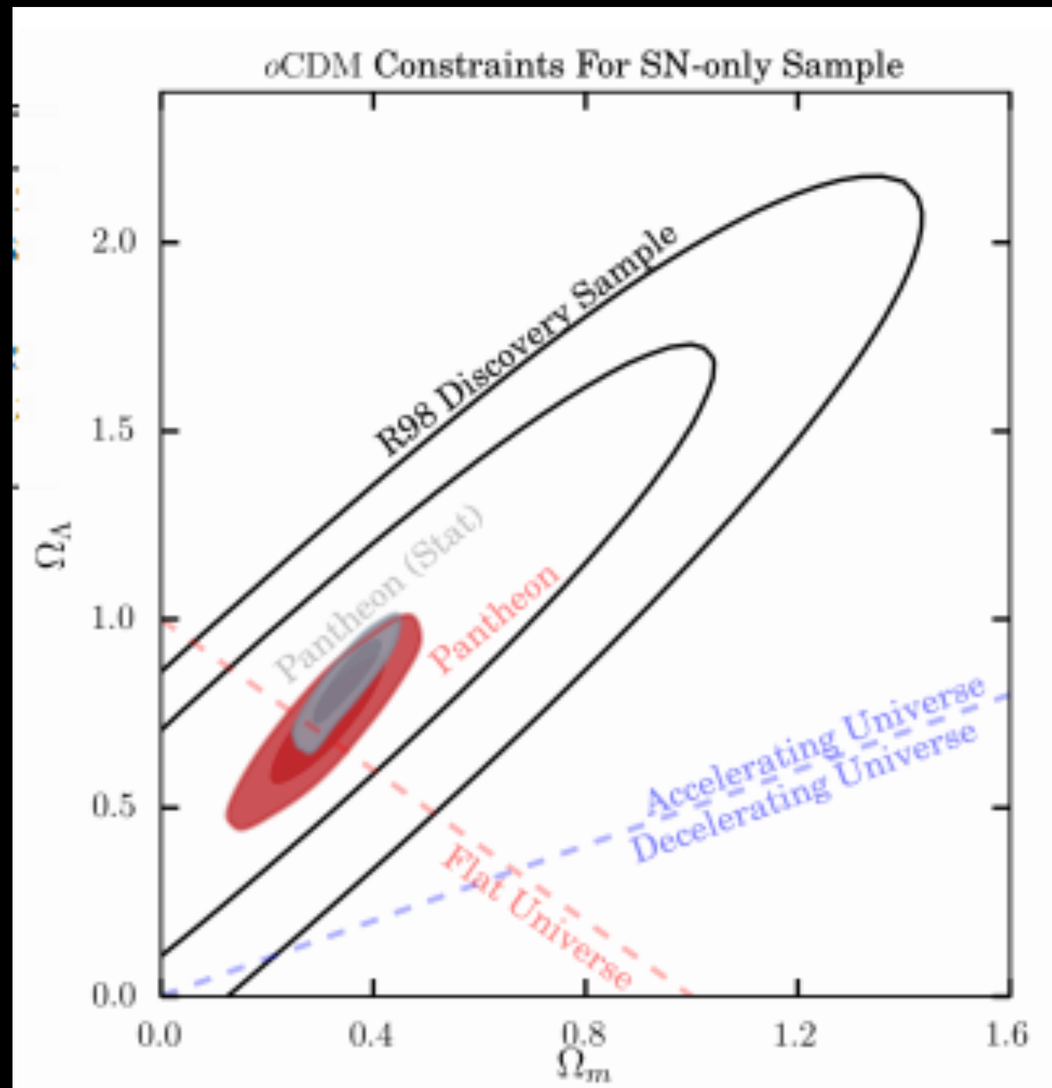
$\rho_{\mathrm{m},0}$ mass density of non-relativistic matter today

ρ_Λ mass density of the cosmological constant

$\mathrm{and}\ H_0$ the rate of expansion of space today

With the posterior probability density in hand, we can make plots like this, showing the 68% and 95% confidence regions — regions with posterior probability density above a threshold, with threshold chosen so that 68% (or 95%) of the probability is contained in that region

parameter related to density of cosmological constant and expansion rate today



parameter related to density of non-relativistic matter and expansion rate today

How to calculate the posterior?

$$P(\theta|d, I) \propto P(d|\theta, I)P(\theta)$$

Let's say we can quickly evaluate the likelihood.
and we make some simple choice for the prior, such as uniform in our chosen parameters.

- Grid-based evaluation: quickly becomes a very large number of evaluations as the number of parameters grows beyond 3
- We will use a Monte Carlo approach instead

Upcoming Assignments

- Distance-Redshift data analysis group project will be assigned on Tuesday next week
- The “Bayesian Inference” project is an individual assignment we have as a warm-up exercise with Bayesian inference (plus version control, testing, use of an IDE, …).
- Next I’ll introduce the Bayesian Inference assignment and then describe our Monte Carlo approach to getting the posterior probability distribution for e.
- You’ll use the same Monte Carlo approach in the group project.

The Bayesian Inference assignment

- Numpy has a built-in Gaussian random number generator that you can use to draw samples from a Gaussian distribution with zero mean and unit variance.

- The assignment is to use a set of such samples to estimate e, the base of the natural logarithm. Actually, rather than an estimate of e, you are going to deliver the posterior probability density of e given the data (your random samples).

- Of course this is a horrible way to calculate e. But it is a perfectly well defined problem. We assume that the random number generator is creating samples, x, from a distribution

$$P(x|a) \propto a^{-x^2/2}$$

and we use our set of samples to determine a (which we know is e)

Calculating P(a|data)

- We will use Bayes' theorem that tells us P(a|d) \propto P(d|a)P(a)
- We will take a uniform prior so that P(a) is independent of a. This choice is ad hoc (why not uniform in ln a, e.g.?) and we should keep in mind the possibility our result might depend on this ad hoc choice.
- We've been told: $$P(x|a) \propto a^{-x^2/2}$$

but that proportionality could be hiding some dependence on a.

How can we determine the proportionality constant?

determining the proportionality constant

$$\int dx p(x|a) = 1$$

$$P(x|a) = N(a) a^{-x^2/2}$$

$$\rightarrow \int dx N(a) a^{-x^2/2} = 1$$

Multiple samples

$$P(x_1, ...x_n|a) = \prod_{i=1}^{n} P(x_i|a) = N^n(a)a^{-\Sigma_i x_i^2/2}$$

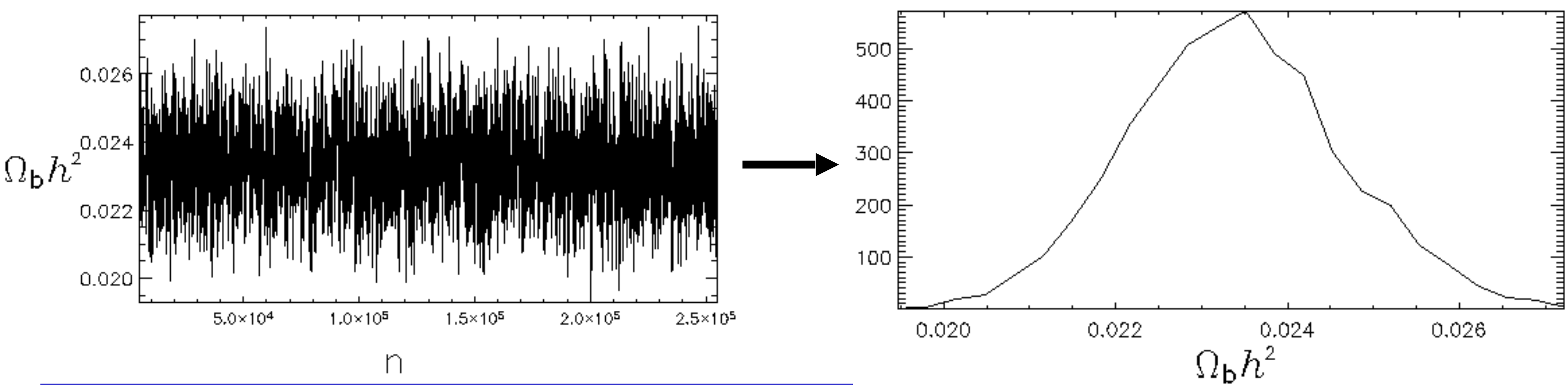
Markov Chain Monte Carlo

- We will employ an algorithm that produces an object called a “chain.”
- The chain is a list of locations in the parameter space.
- The chain has the desired property that sampling from the chain is equivalent to sampling from the posterior probability distribution.
- It is an extremely useful object.
- Yes, this method is overkill for the one-dimensional parameter space we’re dealing with here.

What is the Chain in Monte Carlo Markov Chain?

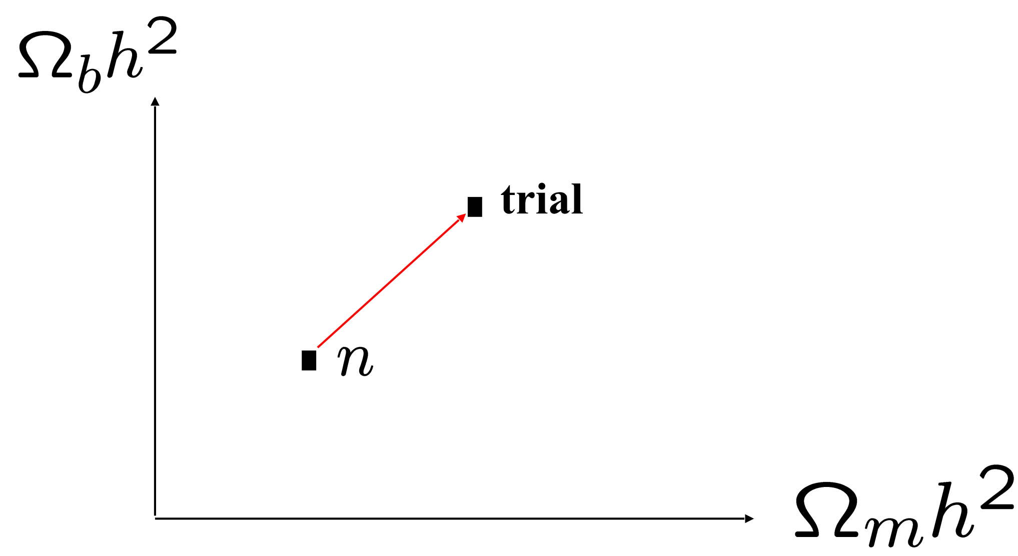
n	$\Omega_b h^2$	$\Omega_m h^2$	...
1.	0.023478	0.1467	...
2.	0.022587	0.1503	....
.	.	.	
.	.	.	
.	.	.	

What good is the chain?



How is the chain built?

- Randomly choose step (according to the *generating function*) to move from current location in the parameter space.
- Evaluate the likelihood.
- Accept or reject movement to new location based on Metropolis-Hastings algorithm that compares current and previous likelihood.
- Write down current location in parameter space as the n+1th chain element. (Note that in the case of rejection, the n+1th element is the same as the nth element).
- Test for convergence.
- Run until post-convergence chain is sufficiently large.



The Metropolis-Hastings Algorithm

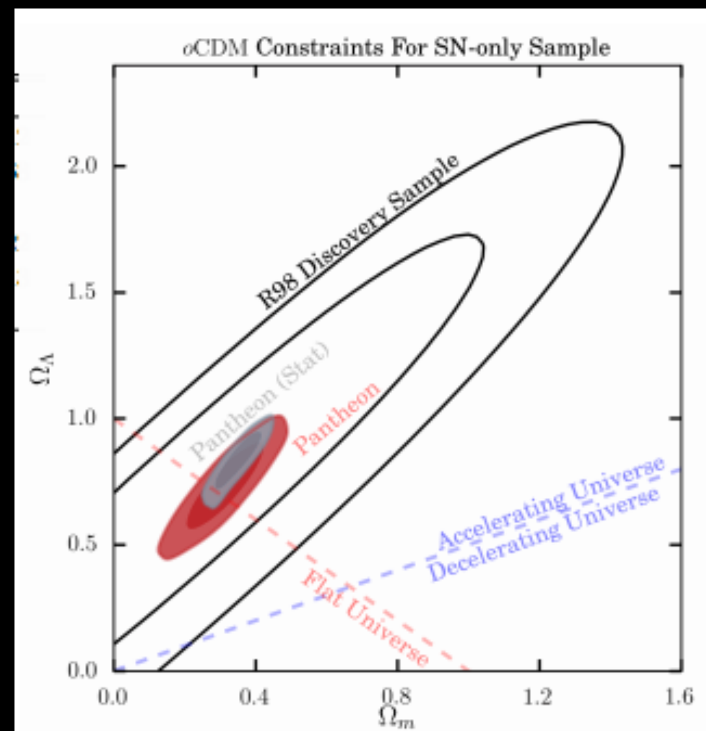
from Wikipedia

1. Initialise
    1. Pick an initial state x_0.
    2. Set $t = 0$.
2. Iterate
    1. *Generate* a random candidate state x' according to $g(x' \mid x_t)$.
    2. *Calculate* the acceptance probability $A(x', x_t) = \min\left(1, \frac{P(x')}{P(x_t)} \frac{g(x_t \mid x')}{g(x' \mid x_t)}\right)$;.
    3. *Accept or reject*:
        1. generate a uniform random number $u \in [0, 1]$;
        2. if $u \leq A(x', x_t)$, then *accept* the new state and set $x_{t+1} = x'$;
        3. if $u > A(x', x_t)$, then *reject* the new state, and copy the old state forward $x_{t+1} = x_t$.
    4. *Increment*: set $t = t + 1$.

g is called the generating function.
It is usually chosen to be a Gaussian with
a covariance matrix that has similar shape to
that of the posterior

The Metropolis-Hastings Algorithm

1. Initialise
    1. Pick an initial state x_0.
    2. Set $t = 0$.
2. Iterate
    1. *Generate* a random candidate state x' according to $g(x' \mid x_t)$.
    2. *Calculate* the acceptance probability $A(x', x_t) = \min\left(1, \frac{P(x')}{P(x_t)} \frac{g(x_t \mid x')}{g(x' \mid x_t)}\right)$;.
    3. *Accept or reject*:
        1. generate a uniform random number $u \in [0, 1]$;
        2. if $u \leq A(x', x_t)$, then *accept* the new state and set $x_{t+1} = x'$;
        3. if $u > A(x', x_t)$, then *reject* the new state, and copy the old state forward $x_{t+1} = x_t$.
    4. *Increment*: set $t = t + 1$.



g is called the generating function.
It is usually chosen to be a Gaussian with a covariance matrix that has similar shape to that of the posterior

ideally, the generating function moves us from one region of high probability to another

Generating Function

- You want a generating function that manages to get you to all regions with significant probability, and that avoids regions with nearly zero probability.

- I recommend you use a Gaussian, so in your 1-dimensional case that just means you have to choose the variance.

$$g(x'|x_t) = \frac{1}{\sqrt{2\pi}\sigma} \exp[-(x' - x_t)^2/(2\sigma^2)]$$

(where here we are using x for parameter and earlier we had used it for an element of our data set)

Why does this work?

- First, by “work” we mean that the resulting chain is such that if you draw an element out from the chain at random, it’s like drawing a sample from the posterior probability distribution.

- For an algorithm to work, it has to “converge.” There is always a transient period, before it settles into a stationary state. To me, getting convergence to happen is a bit of an art.

- Once converged, it has to stay converged. This happens with MH because it ensures *detailed balance*. The probability of transitioning from theta_1 to theta_2 is related to the reverse probability in just the right way so that, in the process of growing the chain, there will be just as many transitions from 1 to 2 as there are from 2 to 1.

Summary

- We build models of data in order to infer truth from data.
- Bayes' theorem provides a framework for updating our probabilities given new data.
- Bayesian inference is inference done on a secure, logical, coherent foundation and is used in many fields (even spam email detection as you may have noticed).
- MCMC is a useful tool for carrying out Bayesian inference.
- You'll use it for an extremely simple data set and model in the assignment due on Tuesday, and then you'll use it in a more interesting setting to complete the group project that we'll assign on Tuesday.

Aside on the normalization constant

$$1 = \int P(\ell|d, I) d\ell$$

$$\rightarrow P(d|I) = \int P(d|\ell, I) P(\ell|I) d\ell$$

sample \ell from the prior

sample d given that value of \ell