Likelihood Functions and Markov Chain Monte Carlo

The normal distribution

- In the last few lectures, we've been working with χ^2 goodness-of-fit tests
- The origin of χ^2 is the assumption that data are drawn from a normal distribution. The probability density function (PDF) for the Gaussian distribution is (equations courtesy of wikipedia):

$$f(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\,\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

- How the χ^2 test works should be clear from the exponent
  - we sum the deviations from the expectation value (the theoretical mean) to test whether a set of data is distributed normally

The likelihood function

- The PDF for the distribution gives us the likelihood of an individual data point (x) given a model distribution (i.e. given a Gaussian with some mean and variance)
- To test if a *set* of data is likely for a particular model, we would determine the likelihood of each datum, and multiply them to determine an overall likelihood. The most probable model would maximize this likelihood:

$$\mathcal{L}(\mu, \sigma) = \prod_{i=1}^{n} f(x_i \mid \mu, \sigma^2) = \left(\frac{1}{2\pi\sigma^2}\right)^{n/2} \exp\left(-\frac{\sum_{i=1}^{n}(x_i - \mu)^2}{2\sigma^2}\right)$$

- In the χ^2 test maximizing the product of probabilities becomes *minimizing the sum across the exponent values*
  - i.e. try a model (μ,σ), compare to each datum (x_i), sum the difference, find the minimum χ^2

The log likelihood function

- Given the sum in the exponent term, it is typically easier to rewrite this likelihood as the log of the likelihood (with no loss of generality as maximizing a log-likelihood is the same as maximizing a likelihood)
- So, finding the best model (finding the probability of the data for the best model) is equivalent to maximizing:

$$\ln\left(L(\mu,\sigma)\right) = -\frac{1}{2}\sum_i\left[\frac{(O_i - E_i)^2}{\sigma_i^2} + \ln(2\pi\sigma_i^2)\right]$$

- I've switched back to using the more familiar data ("observed" O) and model ("expected" E) notation from the χ^2 lecture notes

Bayes' theorem

- Bayes' theorem famously relates the conditional probability of *A* given that *B* is true to that of *B* given that *A* is true and the individual probabilities of *A* and *B*:

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

- If we wish to determine the best model given our data, we could rewrite this as:

$$P(model|data) = \frac{P(data|model)P(model)}{P(data)}$$

- The probability of the data given the model is simply the likelihood function from the previous slides
- $P(model)$ is the prior probability of the model in the absence of any new data

Bayes' theorem and the likelihood function

- For model inference, Bayes' theorem is sometimes written as:

$$\text{Posterior Probability} \propto \text{Likelihood} \times \text{Prior}$$

- Where the *Posterior Probability* is the probability of the model (given the data) and the *P*(*data*) term is encoded in the proportionality, due to the fact that the equation has to equal 1 (as we are determining a probability)
- For log-likelihoods, the right-hand-side of this equation could be written as:

$$\ln(\text{Likelihood}) + \ln(\text{Prior})$$

- The likelihood (for a normal distribution) is as given a few slides ago

Log-likelihood function for a straight line model

- Last week when we were considering goodness-of-fit and the χ^2 statistic our model of choice was always a line given by:

$$y = \mathrm{m}x + \mathrm{b}$$

- For such a model, the log of the likelihood function (the probability of the data given the model) would be:

$$\ln\left(L(y|x,\sigma,m,b)\right) = -\frac{1}{2}\sum_i \left[\frac{(y_i-(mx_i+b))^2}{\sigma_i^2} + \ln(2\pi\sigma_i^2)\right]$$

- Check this equation makes sense, given the derivations on the first few slides of this lecture ... and note, again the similarity with χ^2 for part of this equation

Uninformative Priors

$$\text{Posterior Probability} \propto \text{Likelihood} \times \text{Prior}$$

- The prior acts to reweight the likelihood based on our existing knowledge. Any justifiable choice can be made
- Remember, the prior looks like a *probability*, so it must exist in the range 0 to 1
- The most basic acceptable choice is to use a *flat* or *uninformative* prior, which is basically setting the acceptable fitting range (i.e. the parameter space you believe you'd have to search to find the best model)
- For instance, if I only wanted to fit an intercept in the range $0 < b < 10$ then my prior would be:

0 for $b > 10$ or $b < 0$ $\quad\quad$ 1 for $0 < b < 10$

Model Assumptions

- Note the large number of assumptions that appear to be entering into this method of fitting a model
  - We assumed a Gaussian PDF to derive the likelihood function (we could have assumed any PDF)
  - We have the option to choose priors in any form
- These assumptions were *also* being implicitly made for χ^2 fitting. The difference is that the Bayesian approach is making these assumptions *explicit*
- The Bayesian framework allows model inference to be made with a wider range of assumptions but the Gaussian-based likelihood function I have derived, combined with flat priors *will always be as good as* χ^2

Markov Chain Monte Carlo

- For χ^2 we made a grid of values at some resolution, and then found the minimum χ^2 for the values in that grid
- This is inefficient unless we have some a priori knowledge about the correct grid resolution
- MCMC samplers instead walk through probability space with steps appropriate to the probability density, to efficiently map the posterior probability, P(*model*|*data*)
- Another reason walking is more practical than using a grid is that if the probability space is correctly traced, then there is no need to mess around with, e.g., $\Delta\chi^2$
  - A 68% contour will enclose 68% of the probability, thus naturally mapping out a 68% confidence limit

The Metropolis-Hastings Algorithm

- A common MCMC algorithm for mapping out probability space is Metropolis-Hastings:
- Select initial parameter values
  - e.g., for a straight line choose m and b values
- Move away from those values using a *proposal function*
  - a typical choice is to move away using a Gaussian (so for a straight line shift m by Δm and b by Δb according to a Gaussian centered on m and b)
  - Note that for a Gaussian proposal you get to choose the standard deviation. A rule of thumb is that if your step is efficient then ultimately ~30% of proposals will be accepted (see next slide)

The Metropolis-Hastings Algorithm

- Move away from those values using a *proposal function*
  - The *proposal function* does not have to be Gaussian, you can choose any *symmetric* proposal
  - A wrong choice of *proposal function* does not ruin the method, it just makes the sampler inefficient
- For the old and the new model parameters, determine the posterior (the log likelihood + the log prior) and find $R = P_{new}/P_{old}$ (which is $\ln R = \ln P_{new} - \ln P_{old}$ in log space)
  - If $R > 1$ always accept the new parameters
  - If $R < 1$ accept the new parameters with probability R
  - Check how often the new parameters are accepted. If this is far from ~30%, change the *proposal* step size

The Metropolis-Hastings Algorithm

- The values of the (log) posterior probability, the (log) likelihood and of each parameter should be recorded
  - this series of parameter values is called the *chain*
- Chain values corresponding to the maximum posterior probability represent the model that best fits the data
  - In the absence of a prior these would be called the *maximum likelihood* values
- The, e.g., 68% (or, e.g., 95%) of values enclosing the maximum posterior probability are the confidence limits
  - note these drop naturally out of the method ... as we have been sampling *probability space*, so, where the space is more probable we have more values

Python tasks

1. In my week14 directory in SVN is a file of (x,y) data called "line.data". Each of the 10 columns corresponds to an x bin of $0 < x < 1$, $1 < x < 2$, $2 < x < 3$ up to $9 < x < 10$. Each of the 20 rows is a y measurement in that x bin
   - Read the file and find the variance (σ_i; remember to pass *ddof=1)* and mean (y_i) of the y data in each bin (i) of x
2. The data have been drawn from a straight line of the form $y = mx + b$ and scattered according to a Gaussian
   - Write a function that calculates the (ln) posterior probability for a straight line model for the data when passed values of m and b
   - Use "flat" priors that correspond to the extent of the m, b space that needs to be sampled (e.g. $0 < b < 10$)

Python tasks

3. Using the Metropolis-Hastings algorithm walk through the parameter space and create an MCMC chain
   - Start at values of m and b that seem reasonable
   - Use a Gaussian proposal function with a step size (the standard deviation of the Gaussian) of 0.1
4. Assess whether your proposal function produces an acceptance rate of about 30%
   - If not, alter the step size of the proposal function to increase or reduce the acceptance rate
5. Use your chain to determine the most probable values of m and b and the 68% confidence limits on m and b