IFT Summer school: Statistics exercises

Will Handley

Monday, March 13, 2017

Background Theory

The normalised Friedmann equation of the universe can be written as:

$$\left(\frac{H}{H_0}\right)^2 = \Omega_{r,0}\, a^{-4} + \Omega_{m,0}\, a^{-3} + \Omega_{k,0}\, a^{-2} + \Omega_{\Lambda,0}, \qquad a = \frac{1}{1+z} \tag{1}$$

where:

- a is the time-dependent scale-factor of the universe, normalised to 1 today.
- z is the observed redshift of an object located at a previous epoch.
- $H = \frac{d}{dt}\log a$ is the Hubble parameter, with present value H_0.
- $\Omega_i = 8\pi G\rho_i/3H^2$ is the density parameter for component i, with present value $\Omega_{i,0}$.

From these definitions, you should note that $\Omega_{i,0} \sim 1$, and that by setting $a = 1$ (today), $1 = \Omega_{r,0} + \Omega_{m,0} + \Omega_{k,0} + \Omega_{\Lambda,0}$.

With this, along with a little geometry, we may compute several cosmological distance measures:

$$d_L = \frac{d_M}{1+z} \qquad \text{(Luminosity dist.)} \tag{2}$$

$$d_M = \begin{cases} \frac{d_H}{\sqrt{\Omega_{k,0}}} \sinh \frac{d_C\sqrt{\Omega_{k,0}}}{d_H} & \Omega_{k,0} > 0 \\ d_C & \Omega_{k,0} = 0 \\ \frac{d_H}{\sqrt{|\Omega_{k,0}|}} \sin \frac{d_C\sqrt{|\Omega_{k,0}|}}{d_H} & \Omega_{k,0} < 0 \end{cases} \qquad \text{(Transverse comoving dist.)} \tag{3}$$

$$d_C = d_H \int_0^z \frac{dz}{\sqrt{\Omega_{r,0}a^{-4} + \Omega_{m,0}a^{-3} + \Omega_{\Lambda,0}}} \qquad \text{(Comoving dist.)} \tag{4}$$

$$d_H = \frac{c}{H_0} \qquad \text{(Hubble dist.)} \tag{5}$$

For specific kinds of supernovae/astronomical objects, we can directly measure the magnitude $\mu = m - M$, which is related to their luminosity distance d_L:

$$\mu = 5(\log_{10} d_L - 1) \qquad \text{(obviously...)} \tag{6}$$

We measure the magnitudes μ and redshifts z of N supernovae, and combine them into two data vectors $y = (\mu_1, \ldots, \mu_N)$, $x = (z_1, \ldots, z_N)$. These magnitudes also come with an associated error, or more precisely a covariance matrix Σ.[1]. Given a specific cosmology defined by the parameters $\theta = (H_0, \Omega_{r,0}, \Omega_{m,0}, \Omega_{r,0}, \Omega_{\Lambda,0}, \Omega_{k,0})$, we can compute the luminosity distance (2) as a function of redshift, and thus the theoretical magnitude of the object $\hat{\mu}(z;\theta)$ for any given z, and thus the theoretical data vector $\hat{y}(\theta) = (\hat{\mu}(z_1;\theta), \ldots, \hat{\mu}(z_N;\theta))$. The likelihood of observing this data is then:

$$\mathcal{L}(\theta) = P(y, x|\theta) = \frac{1}{\sqrt{\det 2\pi\Sigma}} \exp\left(-\frac{1}{2}[y - \hat{y}(\theta)]^T \Sigma^{-1} [y - \hat{y}(\theta)]\right) \tag{7}$$

This object forms the centre of our inference. The data are taken from:
`http://supernova.lbl.gov/Union/`
which you should visit to see some example figures.

1 Set up

Copy the work directory into your local user area, and source the relevant files:

```
ssh -X <username>@hydra.ift.uam-csic.es
cp -r /home/prof4/PolyChord ~/PolyChord
cd ~/PolyChord
source modules
```

Alternative: Local install for unix-like systems (MAC, linux)

```
wget https://www.mrao.cam.ac.uk/~wh260/PolyChord.tar.gz
tar -xvf PolyChord.tar.gz
cd ~/PolyChord
make veryclean
make PyPolyChord
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$PWD/lib
export LD_PRELOAD=/usr/lib/openmpi/libmpi.so
```

(NB: You will need to substitute the location of your openmpi library into LD_PRELOAD)

You will also need the python modules (both pip-installable):

[1] These statements are grossly misleading. Much of the research/controversy in this field goes into correctly modelling the errors in the Hubble distance ladder

- getdist `https://pypi.python.org/pypi/GetDist/`
- MPI4py `https://pypi.python.org/pypi/GetDist/`

as well as a gfortran compiler + mpi. Alternatively, you can run make veryclean; make PyPolyChord MPI=, if you don't want the faff of setting up MPI.

2 Plotting the data (5-10 mins)

1. Run the script which produces a plot of the supernovae:

```
python plot_sn.py
gedit_bg plot_sn.py &
```

   (gedit_bg is an alias that was loaded when you sourced "modules", which just pipes the stderr to /dev/null. gedit is a simple gui text editor. Feel free to substitute your own preferred text editor.[2])

2. Modify the script so that instead of plotting magnitudes, you plot luminosity distances d_L against redshift z (Hint: consider equation (6), and make sure you calculate the errors correctly).
3. Question: Why can't we turn this plot into something more theoretically intuitive, such as scale-factor against cosmic time?

3 Metropolis Hastings (20-30 mins)

I have written up the Gaussian likelihood for you (if you are interested, it's all in SNE/supernova_data.py). For now, we will assume that the universe is flat ($\Omega_{k,0} = 0$), with only matter and dark energy

4. Write a two-dimensional Metropolis-Hastings algorithm to explore the likelihood for flat universe with only dark energy and matter in it. You should choose your parameters to be H_0 and $\Omega_{m,0}$, with the value of dark energy therefore being $\Omega_{\Lambda,0} = 1 - \Omega_{m,0}$. You should start by modifying the file MH.py (If you haven't managed to debug your code after 20 minutes or so, you can copy the solution in answers/MH_1.py)
5. Question: Comment on the properties of the chain produced
6. How do the properties of the chain change if you choose a more sensible starting point? or a different step size? Even if you tune these correctly, do you notice anything funny about the properties of your chains?
7. Plot your chains using getdist:

[2] In my opinion, any text editor is fine, so long as its vim or emacs.

```
O_L_array = 1. - np.array(O_m_array)
samples = np.array([H0_array, O_m_array, O_L_array]).T
weights = np.array(count_array)
names = ['H0','O_m',r'O_L*']
labels = ['H_0',r'\Omega_m',r'\Omega_\Lambda']
samples = getdist.MCSamples(samples=samples,
                            weights=weights,
                            names=names,
                            labels=labels)
g = getdist.plots.getSubplotPlotter()
g.triangle_plot(samples, filled=True)
plt.show()
```

8. Is there anything different if you use the likelihood that doesn't take into account systematic errors? (change loglikelihood_sys for loglikelihood_nosys)

4 PolyChord (20 mins)

9. Start by using polychord to plot what you just worked on:

```
mpirun -np 1 python run_PyPolyChord.py
```

You can change -np 1 to a higher number to increase the number of MPI cores which PolyChord runs on. What is the fundamental difference between the likelihood which includes systematic errors, and the one that does not? Have a look at the script to see how PolyChord is set up. Most of the code is purely interfacing our code with PolyChord's input, but some of the settings can be important.

10. Now for the punch-line. Run:

```
mpirun -np 1 python run_all.py
```

This will run three models:

(a) matter + dark energy (flat)
(b) matter + dark energy + curvature
(c) matter + curvature (no dark energy).

After it's done, what can you say about both the evidences of each model, and the comparison of the posteriors with & without curvature?

11. Modify the above script by adding in/removing components of the universe (radiation etc).

12. Plot the predictive posterior distribution using:

```
python compute_contours.py
python plot.py
```

This plots the predictive posterior distribution for the flat matter dark energy universe. Modify compute_contours.py to do it for the curved matter dark energy universe.

5 Bonus Questions/extended investigations

To be attempted in any order

13. Modify code to allow for a dark energy with a variable equation of state parameter w (i.e. change $\Omega_{\Lambda,0} \to \Omega_{\Lambda,0}\, a^{-3(1+w)}$).

14. Modify your MH algorithm to include convergence diagnostics, such as split-Rhat. A good reference can be found in the stan manual: `https://github.com/stan-dev/stan/releases/download/v2.14.0/stan-reference-2.14.0.pdf` (28.3. Initialization and Convergence Monitoring)

15. How is run-time and evidence accuracy affected by the PolyChord setting nlive?

16. How sensitive are your conclusions to the prior widths (qualitatively and quantitatively)?