Homework 7

MLE, MCMC, Interpolation, Expectation Maximization (EM), and Resampling Methods

This notebook is arranged in cells. Texts are usually written in the markdown cells, and here you can use html tags (make it bold, italic, colored, etc). You can double click on this cell to see the formatting.

The ellipsis (...) are provided where you are expected to write your solution but feel free to change the template (not over much) in case this style is not to your taste.

Hit "Shift-Enter" on a code cell to evaluate it. Double click a Markdown cell to edit.

Link Okpy

In [ ]:

```
from client.api.notebook import Notebook
ok = Notebook('hw7.ok')
_ = ok.auth(inline = True)
```

Imports

In [2]:

```
import numpy as np
from scipy.integrate import quad
#For plotting
import matplotlib.pyplot as plt
%matplotlib inline
```

Supernova Cosmology Project

In this homework, we use a compilation of supernovae data to show that the expansion of the universe is accelerating, and hence it contains dark energy. This is the Nobel prize winning research in 2011 (https://www.nobelprize.org/nobel_prizes/physics/laureates/2011/ (https://www.nobelprize.org/nobel_prizes/physics/laureates/2011/)), and Saul Perlmutter, a professor of physics at Berkeley, shared a prize in 2011 for this discovery.

"The expansion history of the universe can be determined quite easily, using as a “standard candle” any distinguishable class of astronomical objects of known intrinsic brightness that can be identified over a wide distance range. As the light from such beacons travels to Earth through an expanding universe, the cosmic expansion stretches not only the distances between galaxy clusters, but also the very wavelengths of the photons en route. By the time the light reaches us, the spectral wavelength λ has thus been redshifted by precisely the same incremental factor $z = \Delta\lambda/\lambda$ by which the cosmos has been stretched in the time interval since the light left its source. The recorded redshift and brightness of each such object thus provide a measurement of the total integrated expansion of the universe since the time the light was emitted. A collection of such measurements, over a sufficient range of distances, would yield an entire historical record of the universe’s expansion." (Saul Perlmutter, http://supernova.lbl.gov/PhysicsTodayArticle.pdf (http://supernova.lbl.gov/PhysicsTodayArticle.pdf)).

Supernovae emerge as extremely promising candidates for measuring the cosmic expansion. Type I Supernovae arises from the collapse of white dwarf stars when the Chandrasekhar limit is reached. Such nuclear chain reaction occurs in the same way and at the same mass, the brightness of these supernovae are always the same. The relationship between the apparent brightness and distance of supernovae depend on the contents and curvature of the universe.

We can infer the "luminosity distance" D_L from measuring the inferred brightness of a supernova of luminosity L. Assuming a naive Euclidean approach, if the supernova is observed to have flux F, then the area over which the flux is distributed is a sphere radius D_L, and hence

$$F = \frac{L}{4\pi D_L^2}.$$

In Big Bang cosmology, D_L is given by:

$$D_L = \frac{\chi(a)}{a}$$

where a is the scale factor ($\frac{\lambda_0}{\lambda} = 1 + z = \frac{a_0}{a}$, and the quantity with the subscript 0 means the value at present. Note that $a_0 = 1, z_0 = 0$.), and χ is the comoving distance, the distance between two objects as would be measured instantaneously today. For a photon, $cdt = a(t)d\chi$, so $\chi(t) = c\int_t^{t_0} \frac{dt'}{a(t')}$. We can write this in terms of a Hubble factor ($H(t) = \frac{1}{a}\frac{da}{dt}$), which tells you the expansion rate:
$\chi(a) = c\int_a^1 \frac{da'}{a'^2 H(a')} = c\int_0^z \frac{dz'}{H(z')}$. (change of variable using $a = \frac{1}{1+z}$.)

Using the Friedmann equation (which basically solves Einstein's equations for a homogenous and isotropic universe), we can write H^2 in terms of the mass density ρ of the components in the universe:
$H^2(z) = H_0^2[\Omega_m(1+z)^3 + (1-\Omega_m)(1+z)^2]$.

Ω is the density parameter; it is the ratio of the observed density of matter and energy in the universe (ρ) to the critical density ρ_c at which the universe would halt is expansion. So Ω_0 (again, the subscript 0 means

the value at the present) is the total mass and energy density of the universe today, and consequently $\Omega_0 = \Omega_m$ (matter density parameter today; remember we obtained the best-fit value of this parameter in Project 1?) = $\Omega_{\text{baryonoic matter}} + \Omega_{\text{dark matter}}$. If $\Omega_0 < 1$, the universe will continue to expand forever. If $\Omega_0 > 1$, the expansion will stop eventually and the universe will start to recollapse. If $\Omega_0 = 1$, then the universe is flat and contains enough matter to halt the expansion but not enough to recollapse it. So it will continue expanding, but gradually slowing down all the time, finally running out of steam only in the infinite future. Even including dark matter in this calculation, cosmologists found that all the matters in the universe only amounts to about a quarter of the required critical mass, suggesting a continuously expanding universe with deceleration. Then, using all this, we can write the luminosity distance in terms of the density parameters:

$$D_L = \frac{\chi(a)}{a} = c(1+z)\int_0^z \frac{dz'}{H(z')} = c(1+z)\int_0^z \frac{dz'}{H_0[\Omega_m(1+z')^3 + (1-\Omega_m)(1+z')^2]^{1/2}}$$

$$= \frac{2997.92458}{h}(1+z)\int_0^z \frac{dz'}{[\Omega_m(1+z')^3 + (1-\Omega_m)(1+z')^2]^{1/2}} \; [unit\ of\ Mpc]$$

where $H_0 = 100 \cdot h\ [km \cdot s^{-1} Mpc^{-1}]$.

Fluxes can be expressed in magnitudes m, where $m = -2.5 \cdot \log_{10} F$ + const. The distance modulus is $\mu = m - M$ (M is the absolute magnitude, the value of m if the supernova is at a distance 10pc. Then, we have:

$$\mu = 25 + 5 \cdot \log_{10}\Big(D_L\ [in\ the\ unit\ of\ Mpc]\Big)$$

In this assignment, we use the SCP Union2.1 Supernova (SN) Ia compilation. (http://supernova.lbl.gov/union/ (http://supernova.lbl.gov/union/))

First, load the measured data: z (redshift), μ (distance modulus), $\sigma(\mu)$ (error on distance modulus)

In [ ]:

```
data = np.loadtxt("sn_z_mu_dmu_plow_union2.1.txt", usecols=range(1,5))
# z
z_data = data[:,0]
# mu
mu_data = data[:,1]
# error on mu (sigma(mu))
mu_err_data = data[:,2]
```

1. Plot the measured distance modulus as a function of redshift with errorbars. Then, assume three different scenarios: $\Omega_m = 0, 0.3, 1.$

Remember:

$$D_L = \frac{2997.92458}{h}(1+z)\int_0^z \frac{dz'}{[\Omega_m(1+z')^3 + (1-\Omega_m)(1+z')^2]^{1/2}}$$

$$\mu = 25 + 5 \cdot \log_{10}(D_L)$$

Now, plot three curves of μ *as a function of* z *for* $\Omega_m = 0, 0.3, 1$ *on top of the measured data (Calculate* D_L *using quad. For now, assume* $h = 0.7$*.) How do they fit?*

In [ ]:

```
import math

...
```

In [ ]:

```
plt.figure(figsize = (20,14))

...

plt.legend()
plt.xlim(0.01, 1.5)
plt.xlabel('$z$')
plt.ylabel('$\mu$')
plt.show()
```

You should find that the measured data do not fit well to all three scenarios. "The high-redshift supernovae are fainter than would be expected even for an empty cosmos (corresponding to $\Omega_m = 0$)." So what's wrong?

"If these data are correct, the obvious implication is that the simplest cosmological model must be too simple. The next simplest model might be one that Einstein entertained for a time. Believing the universe to be static, he tentatively introduced into the equations of general relativity an expansionary term he called the "cosmological constant" (Λ) that would compete against gravitational collapse. After Hubble's discovery of the cosmic expansion, Einstein famously rejected Λ as his "greatest blunder." In later years, Λ came to be identified with the zero-point vacuum energy of all quantum fields. It turns out that invoking a cosmological constant allows us to fit the supernova data quite well." (Saul Perlmutter, https://www.nobelprize.org/nobel_prizes/physics/laureates/2011/ (https://www.nobelprize.org/nobel_prizes/physics/laureates/2011/))

So in short, the data indicates that faint supernovae are further away from the earth than had been theoretically expected. The expansion rate of the universe is increasing indeed. It seems that some mysterious material (which we call "dark energy") is causing such antigravity effects. The cosmological constant, Λ, the value of the energy density of the vacuum of space is widely accepted as a leading candidate of dark energy.

Now let us add a general form of dark energy to our model.

$$H^2(z) = H_0^2[\Omega_m(1+z)^3 + \Omega_{DE}(1+z)^{3(1+w)} + (1-\Omega_m-\Omega_{DE})(1+z)^2].$$

w is the dark energy equation of state, which is the ratio of its pressure to its energy density. $w = -1$ for the cosmological constant Λ.

$\Omega_0 = \Omega_m$ (matter density parameter today) + Ω_{DE} (dark energy density parameter today), and

$$D_L = \frac{\chi(a)}{a} = c(1+z)\int_0^z \frac{dz'}{H(z')} = c(1+z)\int_0^z \frac{dz'}{H_0[\Omega_m(1+z')^3 + \Omega_{DE}(1+z')^{3(1+w)} + (1-\Omega_m-\Omega$$

$$= \frac{2997.92458}{h}(1+z)\int_0^z \frac{dz'}{[\Omega_m(1+z')^3 + \Omega_{DE}(1+z')^{3(1+w)} + (1-\Omega_m-\Omega_{DE})(1+z')^2]^{1/2}} \ [unit\ of\ M$$

where $H_0 = 100 \cdot h\ [km \cdot s^{-1}Mpc^{-1}]$.

2. Now assume three different scenarios: ($\Omega_m = 0.3, \Omega_{DE} = 0$), ($\Omega_m = 0, \Omega_{DE} = 1, w = -1$), and ($\Omega_m = 0.3, \Omega_{DE} = 0.7, w = -1$). Again, plot three curves of μ as a function of z on top of data (assume $h = 0.7$)

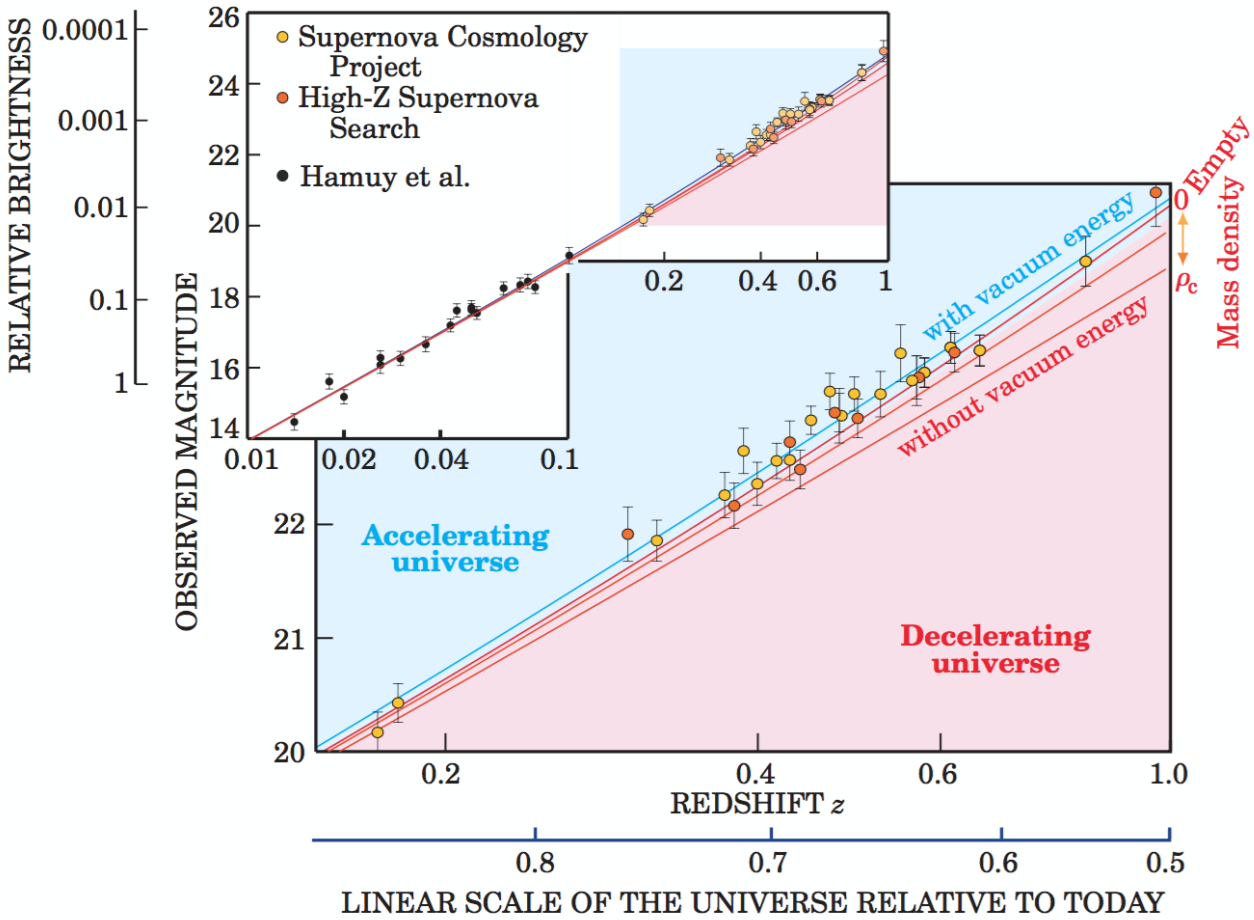
In [ ]:

```
...
```

In [ ]:

```
plt.figure(figsize = (20,14))

...

plt.legend()
plt.xlim(0.01, 1.5)
plt.xlabel('$z$')
plt.ylabel('$\mu$')
plt.show()
```

You basically reproduced the below figure!



You should see that $\Omega_m = 0.3$ and $\Omega_m = 0.7$ fits the data best. In combination with the CMB data, this shows that about 70% of the total energy density is vacuum energy and 30% is mass.

Now, with measurements of the distance modulus μ, use Bayesian analysis to estimate the cosmological parameters.

let us assume that the universe is flat (which is a fair assumption since the CMB measurements indicate that the universe has no large-scale curvature). $\Omega_0 = \Omega_m + \Omega_{DE} = 1$. Then, we do not need to worry about the curvature term:

$$D_L = \frac{\chi(a)}{a} = c(1+z)\int_0^z \frac{dz'}{H(z')} = c(1+z)\int_0^z \frac{dz'}{H_0[\Omega_m(1+z')^3 + (1-\Omega_m)(1+z')^{3(1+w)}]^{1/2}}$$

$$= \frac{2997.92458}{h}(1+z)\int_0^z \frac{dz'}{[\Omega_m(1+z')^3 + (1-\Omega_m)(1+z')^{3(1+w)}]^{1/2}}\ [unit\ of\ Mpc]$$

where $H_0 = 100 \cdot h\ [km \cdot s^{-1} Mpc^{-1}]$.

Assuming that errors are Gaussian (can be justified by averaging over large numbers of SN; central limit theorem), we calculate the likelihood L as:

$$L \propto \exp\Big(-\frac{1}{2}\sum_{i=1}^{N_{\rm SN}} \frac{[\mu_{i,\,data}(z_i) - \mu_{i,\,model}(z_i, \Omega_m, w)]^2}{\sigma(\mu_i)^2} \Big)$$

where $z_i, \mu_i, \sigma(\mu_i)$ are from the measurements, and we compute μ_{model} as a function of z, Ω_m, w.

First, try the **maximum likelihood estimation (MLE)**.

3. Assuming that h = 0.7, find the maximum likelihood estimation of Ω_m and w (i.e. find Ω_m and w which maximizes the likelihood.

(Hint: This is very similar to Problem 2-1, HW6. Take the log of the likelihood and maximize it using scipy.optimize.fmin (https://docs.scipy.org/doc/scipy-0.19.1/reference/generated/scipy.optimize.fmin.html (https://docs.scipy.org/doc/scipy-0.19.1/reference/generated/scipy.optimize.fmin.html)). Note that you need to make initial guesses on the parameters in order to use fmin. You can set them to be 0. Caveat: "fmin" minimizes a given function, so you should multiply the log-likelihood by -1 in order to maximize it using fmin.)

In [ ]:

```
...
```

In [ ]:

```
from scipy import optimize

def minus_log_likelihood(param):

    Omegam, w = param

    if(Omegam<=0 or w>=0):
        lnL = -1.e100
    else:

        ...
        lnL = ...

    return -lnL
```

In [ ]:

```
Omegam, w = optimize.fmin(...)
print('MAP solution')
print('Omega_m = ', Omegam, ', w = ', w)
```

Next, write an MCMC code using the **Metropolis algorithm**. In this problem, assume that the universe is flat, and $w = -1$ (dark energy is Λ.) We call this flat ΛCDM cosmology. By fixing h and w, $D_L = D_L(z, \Omega_m)$.

First, I precalculated D_L for $h = 0.7$ from 2 one-dimensional vectors giving the tabulated values of the parameters z and Ω_m in the range $0.01 < z < 1.5$ and $0.01 < \Omega_m < 1$. We call the tabulated values of z and Ω_m as "z_fit" (length 200) and "Om0_fit" (length 100). Then, "DL_fit" is a 2-dimensional grid of tabulated values of D_L (its dimension 200 × 100. i.e. D_L[i,j] is $D_L\left(z = \text{z\_fit[i]}, \Omega_m = \text{Om0\_fit[j]}\right)$

Now using a 2-D spline interpolation, estimate $D_L(z, \Omega_m)$ for any z and Ω_m.

4. Using scipy.interpolate.RectBivariateSpline, estimate $D_L(z, \Omega_m)$ for any z and Ω_m. Plot $\mu = 25 + 5 \cdot \log_{10}(D_L(z, \Omega_m = 0.3))$ as a function of z on top of the measured data. How does it fit to the data?

(Hint: Let z_spline = RectBivariateSpline(x_fit,y_fit,z_fit). Then, z_spline.ev(x,y) will evaulate the spline at given positions x and y.)

In [ ]:

```
DL_fit = np.loadtxt("DL_fit.txt").T
Om0_fit = np.loadtxt("Om0_fit.txt")
z_fit = np.loadtxt("z_fit.txt")
```

In [ ]:

```
from scipy.interpolate import RectBivariateSpline
```

In [ ]:

```
...
```

In [ ]:

```
plt.figure(figsize = (20,14))

...

plt.legend()
plt.xlim(0.01, 1.5)
plt.xlabel('$z$')
plt.ylabel('$\mu$')
plt.show()
```

Now, run the MCMC code to estimate Ω_m.

In [ ]:

```
# The following is a modified version of the code written by Alan Heavens (htt
p://www.imperial.ac.uk/media/imperial-college/research-centres-and-groups/astrop
hysics/public/icic/data-analysis-workshop/2016/SNcodePython.txt)

# Import data
data = np.loadtxt("sn_z_mu_dmu_plow_union2.1.txt", usecols=range(1,5))
# z
z_data = data[:,0]
# mu
mu_data = data[:,1]
# error on mu (sigma(mu))
mu_err_data = data[:,2]

# length of MCMC chains
nsamples = 10000
# number of parameters
npars    = 1

# Define (gaussian) width of the proposal distribution, one for each parameter.
 This determines how far you propose jumps
Sigma = [0.01]

# Number of supernova:
nSN = len(z_data)

# Declare an empty array of the parameter values of each point.
# Theta[:,0] stores a trace of the parameter \Omega_m
# Theta[:,1] stores log-likelihood values at each point
Theta          = np.empty([nsamples,npars+1])

# Dmu stores mu(data)-mu(theory), temporarily:
Dmu = np.empty(nSN)

# Random starting point in parameter space
# Set initial likelihood to low value so next point is accepted (could compute i
t instead):
Theta[0,:] = [np.random.uniform(), -1.e100]
```

In Part 4, you calculated μ from the ΛCDM model given z and Ω_m, using D_L from the 2D spline interpolation. Using this result, define a function μ_{model} which outputs μ from the ΛCDM theory model given z and Ω_m. Then, Dmu[j] = mu_data[j]-mu_model(z*data[j],Omegam), and the log-likelihood is*

*$$ \mathrm{ln}(L) \approx -\frac{1}{2} \sum{i = 1}^{N{\mathrm{SN}}} \frac{[\mu{i,\ data}(zi) - \mu{i,\ model}(z_i, \Omega_m)]^2}{\sigma(\mui)^2} = -\frac{1}{2} \sum{i = 1}^{N_{\mathrm{SN}}} \frac{Dmu_i^2}{\sigma(\mu_i)^2} $$

5. Define a function for mu_model (mu predicted from theory) and lnL (log-likelihood). Then, you can run the MCMC code and plot the posterior (the routine already given).

In [ ]:

```
def mu_model(z,Omegam):
    ...
    return ...

# Define the likelihood function:

def lnL(Omegam):

    # Treat unphysical regions by setting likelihood to (almost) zero:
    if(Omegam<=0):
        lnL = -1.e100
    else:

    # Compute difference with theory mu at redshifts of the SN, for trial Omegam
        ...

    # Compute ln(likelihood) assuming gaussian errors
        lnL = ...

    return lnL


# Now run the MCMC code (all routine already given below)

# Draw new proposed samples from a proposal distribution, centred on old values
 Omegam[i-1]
# Accept or reject, and colour points according to ln(likelihood):

# Compute initial likelihood value:
Theta[0,npars] = lnL(Theta[0,0])

progress = nsamples/10; val = 0
for i in range(1,nsamples):

    if i%progress == 0:
        val = val + 10
        print("%d percent done" %val)

    lnLPrevious = Theta[i-1,npars]
    OmegamProp = np.random.normal(Theta[i-1,0],Sigma[0])

    lnLProp    = lnL(OmegamProp)

    # Metroplis-Hastings algorithm:

    if(lnLProp > lnLPrevious):
    # Accept point if likelihood has gone up:
        Theta[i,0]     = OmegamProp
        Theta[i,npars] = lnLProp
    else:
    # Otherwise accept it with probability given by ratio of likelihoods:
        alpha = np.random.uniform()

        if(lnLProp - lnLPrevious > np.log(alpha)):
            Theta[i,0]     = OmegamProp
            Theta[i,npars] = lnLProp
        else:
        # Reject; Repeat the previous point in the chain:
            Theta[i,0]     = Theta[i-1,0]
```

```
            Theta[i,npars] = lnLPrevious

# Remove a burn in period, arbitrarily chosen to be the first 10% of the chain:
nburn = math.floor(nsamples/10)


# Plot the histogram of Omegam after the burn-in phase:
plt.hist(Theta[nburn:,0],bins=30)
plt.xlabel(r'$\Omega_m$')
plt.show()

# Determine the best-fit value and constraint
print ('Omega_m = ',np.mean(Theta[nburn:nsamples,0]), '+/-' ,np.std(Theta[nburn:
nsamples,0]))
```

Now, assume a more general form of dark energy, as in Part 3. (Do not fix w to -1; add w as a parameter.)

In the flat universe,

$$D_L = \frac{\chi(a)}{a} = c(1+z)\int_0^z \frac{dz'}{H(z')} = c(1+z)\int_0^z \frac{dz'}{H_0[\Omega_m(1+z')^3 + (1-\Omega_m)(1+z')^{3(1+w)}]^{1/2}}$$

$$= \frac{2997.92458}{h}(1+z)\int_0^z \frac{dz'}{[\Omega_m(1+z')^3 + (1-\Omega_m)(1+z')^{3(1+w)}]^{1/2}}\ [unit\ of\ Mpc]$$

where $H_0 = 100 \cdot h\ [km \cdot s^{-1} Mpc^{-1}]$. Here, we fix $h = 0.7$.

We calculate the likelihood L as:

$$\ln(L) \approx -\frac{1}{2}\sum_{i=1}^{N_{SN}} \frac{[\mu_{i,\,data}(z_i) - \mu_{i,\,model}(z_i, \Omega_m, w)]^2}{\sigma(\mu_i)^2} = -\frac{1}{2}\sum_{i=1}^{N_{SN}} \frac{Dmu_i^2}{\sigma(\mu_i)^2}$$

where

$$\mu_{i,\,model}(z_i, \Omega_m, w) = 25 + 5 \cdot \log_{10}(D_{L,\,i})$$

$$D_{L,\,i} = \frac{2997.92458}{0.7}(1+z_i)\int_0^{z_i} \frac{dz'}{[\Omega_m(1+z')^3 + (1-\Omega_m)(1+z')^{3(1+w)}]^{1/2}}$$

6. Modify the code in Part 5 (most routine already given) and run the MCMC code to estimate w and Ω_m. (npars = 2 in this case). Plot 1-d posterior of w and Ω_m as well as 2-d posterior (i.e. plot the chain in two-dimensional parameter space. Make sure that the chain has converged (you can change nsamples, nburn).

In [ ]:

```
# Import data
data = np.loadtxt("sn_z_mu_dmu_plow_union2.1.txt", usecols=range(1,5))
# z
z_data = data[:,0]
# mu
mu_data = data[:,1]
# error on mu (sigma(mu))
mu_err_data = data[:,2]

# length of MCMC chains
nsamples = 15000
# number of parameters
npars    = 2

# Define (gaussian) width of the proposal distribution, one for each parameter.
 This determines how far you propose jumps
Sigma = [0.01, 0.01]

# Number of supernova:
nSN = len(z_data)

# Declare an empty array of the parameter values of each point.
# Theta[:,0] stores a trace of the parameter \Omega_m
# Theta[:,1] stores a trace of the parameter w
# Theta[:,2] stores log-likelihood values at each point
Theta          = np.empty([nsamples,npars+1])

# Dmu stores mu(data)-mu(theory), temporarily:
Dmu = np.empty(nSN)

# Random starting point in parameter space
# Set initial likelihood to low value so next point is accepted (could compute i
t instead):
Theta[0,:] = [np.random.uniform(), -np.random.uniform(), -1.e100]
```

In [ ]:

```
def mu_model(z,Omegam,w):
    ...
    return ...
```

In [ ]:

```
# Define the likelihood function:

def lnL(Omegam, w):

    # Treat unphysical regions by setting likelihood to (almost) zero:
    if(Omegam<=0 or w>=0):
        lnL = -1.e100
    else:

    # Compute difference with theory mu at redshifts of the SN, for trial Omegam
        ...

    # Compute ln(likelihood) assuming gaussian errors
        lnL = ...

    return lnL


# Draw new proposed samples from a proposal distribution, centred on old values
 Omegam[i-1]
# Accept or reject, and colour points according to ln(likelihood):

# Compute initial likelihood value:
Theta[0,npars] = lnL(Theta[0,0], Theta[0,1])

progress = nsamples/10; val = 0
for i in range(1,nsamples):

    if i%progress == 0:
        val = val + 10
        print("%d percent done" %val)

    lnLPrevious = Theta[i-1,npars]
    OmegamProp = np.random.normal(Theta[i-1,0],Sigma[0])
    wProp = np.random.normal(Theta[i-1,1],Sigma[1])

    lnLProp    = lnL(OmegamProp, wProp)

    # Metroplis-Hastings algorithm:

    if(lnLProp > lnLPrevious):
    # Accept point if likelihood has gone up:
        Theta[i,0]     = OmegamProp
        Theta[i,1]     = wProp
        Theta[i,npars] = lnLProp
    else:
    # Otherwise accept it with probability given by ratio of likelihoods:
        alpha = np.random.uniform()

        if(lnLProp - lnLPrevious > np.log(alpha)):
            Theta[i,0]     = OmegamProp
            Theta[i,1]     = wProp
            Theta[i,npars] = lnLProp
        else:
        # Reject; Repeat the previous point in the chain:
            Theta[i,0:2]     = Theta[i-1,0:2]
            Theta[i,npars] = lnLPrevious

# Remove a burn in period, arbitrarily chosen to be the first 20% of the chain:
```

```
nburn = 2*math.floor(nsamples/10)


# Plot the histogram of Omegam after the burn-in phase:
plt.hist(Theta[nburn:,0],bins=30)
plt.xlabel(r'$\Omega_m$')
plt.show()

# Plot the histogram of w after the burn-in phase:
plt.hist(Theta[nburn:,1],bins=30)
plt.xlabel('w')
plt.show()

# Scatter plot of the samples (2-d posterior):
plt.scatter(Theta[nburn:,0], Theta[nburn:,1], c = -Theta[nburn:,npars])
plt.xlabel(r'$\Omega_m$')
plt.ylabel('w')
plt.show()

# Print best-fit values and constraints
print ('Omega_m = ',np.mean(Theta[nburn:nsamples,0]), '+/-' ,np.std(Theta[nburn:
nsamples,0]))
print ('w = ',np.mean(Theta[nburn:nsamples,1]), '+/-' ,np.std(Theta[nburn:nsampl
es,1]))
```

Now, include the distance modulus of 12 additional supernovae, which are not-so-good standard candles. They are 3σ away from the best-fit mode.

In [ ]:

```
data = np.loadtxt("sn_z_mu_dmu_plow_union2.1_outlier.txt", usecols=range(1,5))
# z
z_data = data[:,0]
# mu
mu_data = data[:,1]
# error on mu (sigma(mu))
mu_err_data = data[:,2]
```

So we have a total of 592 supernovae, and we can see that the last 12 supernovae seem to be outliers. (i.e. mu_data[580:] contains the distance modulus measurements of these 12 supernovae.)

7. First run the MCMC code in Part 6 with the new data (total of 592 supernovae). Then, using the estimates of Ω_m and w from the MCMC chain, calculate the distance modulus from theory and plot the curve on top of the measured data. Plot the measurements of the last 12 supernovae with different color.

In [ ]:

```
...
```

In [ ]:

```
plt.figure(figsize = (20,14))

...

plt.legend()
plt.xlim(0.01, 1.5)
plt.xlabel('$z$')
plt.ylabel('$\mu$')
plt.show()
```

Remember that in HW6, we used the Gaussian mixture to better model the measurements with outliers. Let us apply the same technique in this case.

$$L = \prod_{i=1}^{N_{\rm SN}} \Big[\frac{g}{\sqrt{2\pi\sigma(\mu_i)^2}} \exp\Big(-\frac{1}{2} \frac{[\mu_{i,\,data}(z_i) - \mu_{i,\,model}(z_i, \Omega_m, w)]^2}{\sigma(\mu_i)^2} \Big) + \frac{1-g}{\sqrt{2\pi\sigma_B^2}} \exp\Big(-\frac{1}{2} \frac{[\mu_{i,\,data}(z_i) -$$

Here, we have 5 free parameters: $\Omega_m, w, g, \sigma_B, \Delta\mu$.

With outliers, we think there is something in the noise we do not really understand, which makes error distribution non-Gaussian. So we hope adding a second Gaussian to the model would better describe the pdf. g determines weights on the two Gaussians. σ_B^2 is the variance of the second Gaussian, which we assume to be larger than the variance of the first Gaussian. $\Delta\mu$ is the distance modulus offset in the second Gaussian.

8. Re-run the MCMC code with this new model. Plot 1-d and 2-d constrains of Ω_m and w as in Part 6 and 7.

In [ ]:

```
# Setup (all routine already given)

data = np.loadtxt("sn_z_mu_dmu_plow_union2.1_outlier.txt", usecols=range(1,5))
# z
z_data = data[:,0]
# mu
mu_data = data[:,1]
# error on mu (sigma(mu))
mu_err_data = data[:,2]

# length of MCMC chains
nsamples = 15000
# number of parameters
npars    = 5

# Define (gaussian) width of the proposal distribution, one for each parameter.
 This determines how far you propose jumps
Sigma = 0.01*np.ones(npars)
Sigma[2] = 0.03*np.ones(1)
Sigma[3] = 0.1*np.ones(1)
Sigma[4] = 0.01*np.ones(1)

# Number of supernova:
nSN = len(z_data)

# Declare an empty array of the parameter values of each point.
# Theta[:,0] stores a trace of the parameter \Omega_m
# Theta[:,1] stores a trace of the parameter w
# Theta[:,2] stores a trace of the parameter g
# Theta[:,3] stores a trace of the parameter sigma_B
# Theta[:,4] stores a trace of the parameter delta mu
# Theta[:,5] stores log-likelihood values at each point
Theta          = np.empty([nsamples,npars+1])

# Dmu stores mu(data)-mu(theory), temporarily:
Dmu = np.empty(nSN)

# Random starting point in parameter space
# Set initial likelihood to low value so next point is accepted (could compute i
t instead):
Theta[0,:2] = [np.random.uniform(), -np.random.uniform()]
Theta[0,2] = np.random.normal(0.5, 0.1)
Theta[0,3] = np.random.normal(10, 0.1)
Theta[0,4] = np.random.uniform()
Theta[0,npars] = -1.e100
```

In [ ]:

```
# Define mu from theory
def mu_model(z,Omegam,w):
    ...
    return ...

# Define the likelihood function:

def lnL(Omegam, w, sigmaB, gs, errs, muoffset):

    # Treat unphysical regions by setting likelihood to (almost) zero:
    if(Omegam<=0 or w>=0 or gs < 0 or gs > 1.):
        lnL = -1.e100
    else:

    # Compute difference with theory mu at redshifts of the SN, for trial Omegam
        ...
        lnL = ...

    return lnL


# Run the MCMC code

# Draw new proposed samples from a proposal distribution, centred on old values
 Omegam[i-1]
# Accept or reject, and colour points according to ln(likelihood):

# Compute initial likelihood value:
Theta[0,npars] = lnL(Theta[0,0], Theta[0,1], Theta[0,3], Theta[0,2], mu_err_data
, Theta[0,4])

progress = nsamples/10; val = 0
for i in range(1,nsamples):

    if i%progress == 0:
        val = val + 10
        print("%d percent done" %val)

    lnLPrevious = Theta[i-1,npars]
    OmegamProp = np.random.normal(Theta[i-1,0],Sigma[0])
    wProp = np.random.normal(Theta[i-1,1],Sigma[1])
    gvalProp   = np.random.normal(Theta[i-1,2],Sigma[2])
    sigmaProp   = np.random.normal(Theta[i-1,3],Sigma[3])
    offsetProp   = np.random.normal(Theta[i-1,4],Sigma[4])

    lnLProp    = lnL(OmegamProp,wProp,sigmaProp,gvalProp,mu_err_data,offsetProp)

    # Metroplis-Hastings algorithm:

    if(lnLProp > lnLPrevious):
    # Accept point if likelihood has gone up:
        Theta[i,0]     = OmegamProp
        Theta[i,1]     = wProp
        Theta[i,2] = gvalProp
        Theta[i,3] = sigmaProp
        Theta[i,4] = offsetProp
        Theta[i,npars] = lnLProp
    else:
    # Otherwise accept it with probability given by ratio of likelihoods:
```

```
        alpha = np.random.uniform()

        if(lnLProp - lnLPrevious > np.log(alpha)):
            Theta[i,0]     = OmegamProp
            Theta[i,1]     = wProp
            Theta[i,2] = gvalProp
            Theta[i,3] = sigmaProp
            Theta[i,4] = offsetProp
            Theta[i,npars] = lnLProp
        else:
        # Reject; Repeat the previous point in the chain:
            Theta[i,0:5]     = Theta[i-1,0:5]
            Theta[i,npars] = lnLPrevious

# Remove a burn in period, arbitrarily chosen to be the first 40% of the chain:
nburn = 4*math.floor(nsamples/10)


# Plot the histogram of Omegam after the burn-in phase:
plt.hist(Theta[nburn:,0],bins=30)
plt.xlabel(r'$\Omega_m$')
plt.show()

# Plot the histogram of w after the burn-in phase:
plt.hist(Theta[nburn:,1],bins=30)
plt.xlabel('w')
plt.show()

# Scatter plot of the samples (2-d posterior):
plt.scatter(Theta[nburn:,0], Theta[nburn:,1], c = -Theta[nburn:,npars])
plt.xlabel(r'$\Omega_m$')
plt.ylabel('w')
plt.show()

# Print best-fit values and constraints
print ('Omega_m = ',np.mean(Theta[nburn:nsamples,0]), '+/-' ,np.std(Theta[nburn:
nsamples,0]))
print ('w = ',np.mean(Theta[nburn:nsamples,1]), '+/-' ,np.std(Theta[nburn:nsampl
es,1]))
```

Reference: See pg. 8-16 (https://lear.inrialpes.fr/~jegou/bishopreadinggroup/chap9.pdf (https://lear.inrialpes.fr/~jegou/bishopreadinggroup/chap9.pdf))

For this Gaussian mixture model, we wish to maximize the likelihood function with respect to the parameters $g, \sigma_B, \Delta\mu$ for $\Omega_m = 0.3, w = -1$. In order to do this, we will apply the **expectation-maximization (EM)** algorithm. This is an iterative method to find maximum likelihood in the case where the model depends on the hidden/latent variable. Here, we call binary variable **a** as our latent variable such that $p(a_k = 1) = \pi_k$

Re-write the likelihood as:

$$L = \prod_{i=1}^{N_{\text{SN}}} \Big[\frac{\pi_1}{\sqrt{2\pi\sigma(\mu_i)^2}} \exp\Big(-\frac{1}{2} \frac{[\mu_{i,\,data}(z_i) - \mu_{i,\,model}(z_i, \Omega_m = 0.3, w = -1)]^2}{\sigma(\mu_i)^2} \Big)$$

$$+ \frac{\pi_2}{\sqrt{2\pi\sigma_B^2}} \exp\Big(-\frac{1}{2} \frac{[\mu_{i,\,data}(z_i) - \mu_{i,\,model}(z_i, \Omega_m = 0.3, w = -1) - \mu_{\text{offset}}]^2}{\sigma_B^2} \Big) \Big]$$

$$= \prod_{i=1}^{N_{\text{SN}}} \Big[\pi_1 \cdot \text{Normal}\Big(\Delta\mu_i = \mu_{i,\,data} - \mu_{i,\,model} \Big| \, \overline{\Delta\mu}_{\text{class 1}} = 0, \sigma(\mu_i)^2\Big) + \pi_2 \cdot \text{Normal}\Big(\Delta\mu_i = \mu_{i,\,data} - \mu_{i,\,m}$$

where $\mu_{i,\,model}$ assumes $\Omega_m = 0.3, w = -1$. Suppose that we measure $\Delta\mu = \mu_{i,\,data} - \mu_{i,\,model}$. For the first Gaussian (expected to describe the distribution of 580 non-outlier, standard-candle supernovae), the mean value of $\Delta\mu$ is 0, and its variance is the measurement noise $\sigma(\mu)^2$. For the second Gaussian which expects to describe the distribution of 12 outliers, we assume that there will be some offset in μ (μ_{offset}), so the mean value of $\Delta\mu$ is μ_{offset}, and it has some unknown variance σ_B^2.

Now apply the EM algorithm.

1. First, initialize: choose $\pi_1 = 0.95$ and $\pi_2 = 0.05$. Let $\mu_{\text{offset}} = 0, \sigma_B = 0.5$ initially.

2. **Expectation (E) step**: Evaluate the responsibilities using the current parameter values.

$$\gamma_{1,\,i} = \frac{\pi_1 \cdot \text{Normal}\Big(\Delta\mu_i \Big| \, \overline{\Delta\mu}_{\text{class 1}}, \sigma(\mu_i)^2\Big)}{\pi_1 \cdot \text{Normal}\Big(\Delta\mu_i \Big| \, \overline{\Delta\mu}_{\text{class 1}}, \sigma(\mu_i)^2\Big) + \pi_2 \cdot \text{Normal}\Big(\Delta\mu_i \Big| \, \overline{\Delta\mu}_{\text{class 2}}, \sigma_B^2\Big)}$$

$$\gamma_{2,\,i} = \frac{\pi_2 \cdot \text{Normal}\Big(\Delta\mu_i \Big| \, \overline{\Delta\mu}_{\text{class 2}}, \sigma_B^2\Big)}{\pi_1 \cdot \text{Normal}\Big(\Delta\mu_i \Big| \, \overline{\Delta\mu}_{\text{class 1}}, \sigma(\mu_i)^2\Big) + \pi_2 \cdot \text{Normal}\Big(\Delta\mu_i \Big| \, \overline{\Delta\mu}_{\text{class 2}}, \sigma_B^2\Big)}$$

   where $i = 1, \ldots, N_{SN}$ (number of measurements). Note that γ_1 and γ_2 are vectors of length N_{SN}. For a supernova i, $\gamma_{1,\,i}$ describes its probability of belonging to the first class (described by the first Gaussian). (Note: Therefore, in the end, we expect 12 outliers have much higher values of γ_2 than normal 580 supernovae - i.e. they have much greater probability of belonging to the second class. This is a systematic way to identify an outlier.)

3. **Maximization (M) step**: Re-estimate the parameters using the current responsibilities

   The mean ($\overline{\Delta\mu}_{\text{class 1}} = 0$) and variance, $\sigma(\mu_i)^2$, of the first Gaussian are fixed at initial values

$$N_1 = \sum_{i=1}^{N_{SN}} \gamma_{1,\,i}, \quad N_2 = \sum_{i=1}^{N_{SN}} \gamma_{2,\,i}$$

$$\overline{\Delta\mu}_{\text{class 2}} = \frac{1}{N_2}\sum_{i=1}^{N_{SN}} \gamma_{2,\,i}\cdot\Delta\mu_i$$

$$\sigma_B^2 = \frac{1}{N_2}\sum_{i=1}^{N_{SN}} \gamma_{2,\,i}\cdot(\Delta\mu_i - \overline{\Delta\mu}_{\text{class 2}})^2$$

$$\pi_1 = \frac{N_1}{N_{SN}},\quad \pi_2 = \frac{N_2}{N_{SN}}$$

4. Evaluate the log-likelihood and check for convergence of either the parameters or the log likelihood. If the convergence criterion is not satisfied return to step 2.

9. Using EM, calculate the converged values of π_1, π_2, and N_2. N_2 is the total number of SN in the second class (can be identified as outliers). Iterate until you reach the convergence (parameters not changing). Then, print out the values of γ_2 and show that 12 outliers have higher values of γ_2 than other supernovae.

In [ ]:

```
data = np.loadtxt("sn_z_mu_dmu_plow_union2.1_outlier.txt", usecols=range(1,5))
# z
z_data = data[:,0]
# mu
mu_data = data[:,1]
# error on mu (sigma(mu))
mu_err_data = data[:,2]

...
...
```

Finally, we use a **Boostrap** resampling method to estimate the posterior of Ω_m and w.

Suppose that we have 10 measurements of x: [3.7, 3.2, 3.3, 3.1, 3.2, 3.5, 2.9, 3.4, 3.0, 3.1]. Now, randomly take 5 samples of 10 data measurements "with replacement."

In [3]:

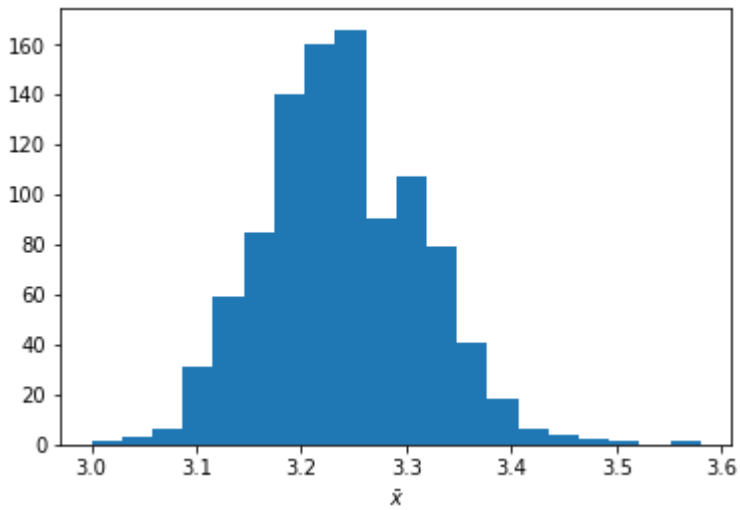
```
x = np.array([3.7, 3.2, 3.3, 3.1, 3.2, 3.5, 2.9, 3.4, 3.0, 3.1])

num_samples = 5
len_x = len(x)
idx = np.random.randint(0, len_x, (num_samples, len_x))
print("After bootstrap re-sampling")
print(x[idx])
```

```
After bootstrap re-sampling
[[ 2.9  3.1  3.   3.   3.2  3.1  3.3  3.5  3.4  2.9]
 [ 3.4  3.3  3.   3.2  3.1  3.1  3.5  3.1  3.1  3.4]
 [ 3.2  3.2  3.2  3.2  3.7  3.1  3.2  3.2  3.3  3. ]
 [ 2.9  2.9  2.9  3.2  3.   3.2  3.5  3.1  3.7  3.5]
 [ 3.2  2.9  3.2  3.2  3.3  3.3  3.2  3.2  3.4  3.7]]
```

Say you wish to see the probability distribution of $\bar{x}$. Then, take 100 samples using bootstrap and plot the histogram of $\bar{x}$.

In [4]:

```
num_samples = 1000
len_x = len(x)
idx = np.random.randint(0, len_x, (num_samples, len_x))
x_bar = np.mean(x[idx], axis = 1)
plt.hist(x_bar,bins=20)
plt.xlabel(r'$\bar{x}$')
plt.show()
```



Now use bootstrap resampling technique to estimate the posterior of Ω_m and w.

10. Take 200 (or more) samples of 580 supernova distance modulus measurements and estimate Ω_m and w using maximum likelihood estimation, as in Part 3. Plot the 1-d posteriors (histogram).

In [ ]:

```
data = np.loadtxt("sn_z_mu_dmu_plow_union2.1.txt", usecols=range(1,5))
# z
z_data = data[:,0]
# mu
mu_data = data[:,1]
# error on mu (sigma(mu))
mu_err_data = data[:,2]
```

In [ ]:

```
...
```

To Submit

Execute the following cell to submit. If you make changes, execute the cell again to resubmit the final copy of the notebook, they do not get updated automatically.
We recommend that all the above cells should be executed (their output visible) in the notebook at the time of submission.
Only the final submission before the deadline will be graded.

In [ ]:

```
_ = ok.submit()
```