PS1: Introduction to Probablity and Statistics

Deepthi Gorthi
AY250: Stellar Populations

September 10, 2016

Problem 1.

(a) In 1995, they introduced blue M&M's. Before then, the color mix in a bag of plain M&M's was 30% brown, 20% yellow, 20% red, 10% green, 10% orange, and 10% tan. Afterward, it was 24% blue, 20% green, 16% orange, 14% yellow, 13% red, 13% brown.

Suppose there are two bags of M&M's, one from 1994 and one from 1996 and you are randomly given one M&M from each bag. One is yellow, one is green. Using Bayes's theorem and a probability table to determine the relative probability that the yellow M&M came from the 1994 bag.

(b) Evaluate the "Evidence" and determine the normalized probability that the yellow M&M came from the 1994 bag.

Solution:

(a) The given distribution of M&M's can be summarized as follows:

Colour	1994	1996
Brown	0.3	0.13
Yellow	0.2	0.14
Red	0.2	0.13
Green	0.1	0.2
Orange	0.1	0.16
Other	0.1	0.24

Given data: One M&M was drawn from each sample and one of them is yellow and another green. To evaluate the relative probablity that the yellow M&M was drawn from the 1994 bag (and hence the green from the 1996 bag), define the two hypothesis as:

H_1: Yellow M&M- 1994 bag, Green M&M- 1996 bag.

H_2: Yellow M&M- 1996 bag, Green M&M- 1994 bag.

Since we have no prior knowledge about either hypothesis, we should begin with flat priors- either hypothesis is equally likely. For probability of the data given the hypothesis we can multiply the probabilities of drawing the green and yellow M&Ms from the respective bag, since they are independent events.

Hypothesis	P(H)	P(D|H)	P(D|H)*P(H)
H_1	1/2	(0.2)*(0.2)	0.02
H_2	1/2	(0.14)*(0.1)	0.007

The relative probability that the yellow M&M was drawn from the 1994 bag is $\boxed{0.02}$.

(b) The 'evidence' is also the probability of the data given all the possible hypotheses. This can be obtained by summing over the likelihoods of all the hypotheses.

$$P(D) = 0.02 + 0.007 = 0.027 \tag{1}$$

Hence the normalized probability that the yellow M&M came from the 1994 bag is:

$$P(H_1|D) = \frac{P(D|H_1) * P(H_1)}{P(D)} = \frac{0.02}{0.027} \sim 74\% \tag{2}$$

The normalized probability of our hypothesis that the yellow M&M was drawn from the 1994 bag is $\boxed{74\%}$.

Problem 2.

Using notes from class, code up your own Metropolis-Hastings (M-H) MCMC sampler. Although it is technically impossible to prove that an MCMC sampler definitively converges (this would require infinite runtime), a simple sanity test it is to sample from a one dimensional Gaussian distribution:

$$P(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(\mu-x)^2}{2\sigma^2}} \tag{3}$$

where μ and σ are values you choose, and x values are generated by your M-H sampler. The density of samples should trace the input distribution. For example, if you select $\mu = 5$ and $\sigma = 1$, the density of samples for x should be a 1d Gaussian with these values (perhaps modulo a normalization constant). Make plots that qualitatively demonstrate convergence of your sampler to a steady state (e.g., lnP vs. x, lnP vs. step number) and a plot that shows your samples relative to the true distribution. Your choice in step size should yield an acceptance fraction between ∼0.25 and 0.5.

Solution

The Metropolis-Hastings Algorithm that I wrote up to sample the given gaussian function is included in this repository as `mcmc_sampler.py`

The proposal density was chosen to be a Gaussian function centered around the current point. The width of the gaussian was found to affect the step size and hence the acceptance rate of the sampler. It was fixed by hit and trial till an acceptance rate ~ 0.32 was obtained.

The Markov chain's convergence can be see from the plot of figure 1. The left panel shows that the walker is moving between high and low probability regions randomly, and the right panel shows that over the simulation time the walker spends more time in the high probability area around the peak of the gaussian as compared to the wings like we expect. The convergence can also be inferred from the acceptance rate which is ~ 0.32, and the auto correlation time which is ~ 1 for this chain. The autocorrelation time is less than a tenth of the chain length (10000) indicating that the samples in the Markov chain are independently drawn and hence representative of the underlying distribution.

Finally, figure 2 shows that the PDF can be reconstructed very accurately, modulo a normalization factor, using the MCMC sampler.

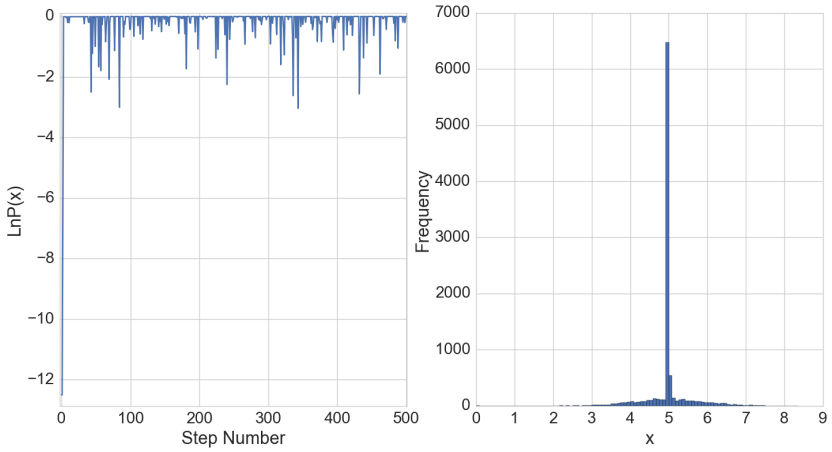

Figure 1: Diagnostic plots of the MCMC sampler to demostrate convergence of the Markov chain. The log likelihood probability distribution is sampled both at high and low probability areas, as shown in panel(1). Only the first 500 samples are shown for illustration. Panel(2) shows that the sampler is predominantly in the high probability region around the peak of the gaussian as compared to the wings.

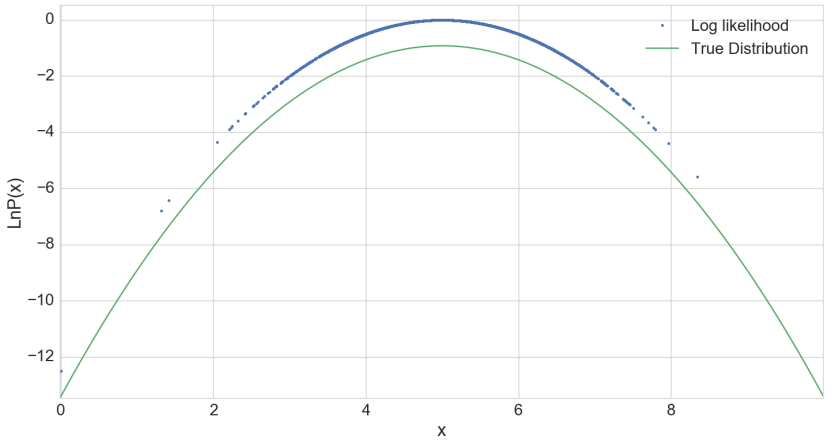

Figure 2: The figure shows the PDF reconstructed from the Markov chain against the actual log probability distribution of the given Gaussian. The sampler suceeds in reconstructing the PDF modulo a normalization constant.

Problem 3.

Using a probabilistic framework, write code that fits a straight line (i.e., $y = mx + b$) to fake data (i.e., data points and error bars). You will need to write a function that simulates fake data that includes Gaussian noise and an arbitrary number of points.

(a) Assume true values of $m = 5$, $b = -2$. Use the M-H MCMC sampler you wrote in problem 2 to infer the true values of m and b for 10, 100, and 1000 data points. Choose a modest amplitude for your uncertainties and clearly indicate your choice. For simplicity, you may assume top hat ('flat') priors for m and b. Make relevant diagnostic plots to indicate convergence, and plot your final results using `corner.py`.

(b) Repeat part (a), but replace your M-H sampler with `emcee`.

Solution

A program to simulate fake data with $m = 5$, $b = -2$ and gaussian uncertainities, and then to fit this straight line using the MCMC sampler from the previous problem is included in this repository under `fit_line.py`

The simulated fake data is shown in figure 3. The Gaussian uncertainities were drawn from a normal distribution centered around zero and of width 3.

The PDF of the slope and the intecept of this data can be sampled using the MCMC sampler we built in the previous section. Before the function can be adapted, we need to define the log-likelihood distribution of both the parameters. This can be simplified as follows:

$$\begin{aligned} P(m, b|D) &= P(m|D, b)P(b|D) \\ &= \frac{P(D|m, b)P(m|b)}{P(D|b)} \frac{P(D|b)P(b)}{P(D)} \\ &= \frac{P(D|m, b)P(m|b)P(b)}{P(D)} \end{aligned} \tag{4}$$

Note that the above equation can be written by interchanging m and b as:

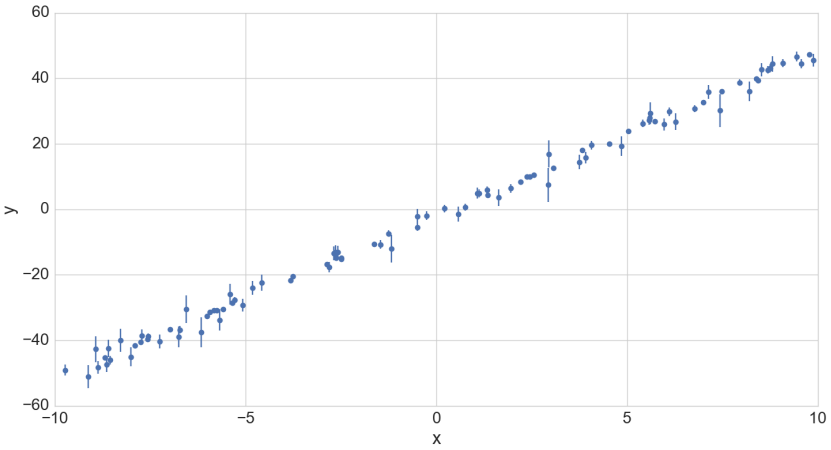

Figure 3: The figure shows data simulated to fit a straight line with slope $m = 5$ and intercept $b = -2$, with gaussian uncertainities drawn from a normal distribution of $\mu = 0$ and $\sigma = 2$.

$$\begin{aligned} P(m, b|D) &= \frac{P(D|m, b)P(m|b)P(b)}{P(D)} \\ &= \frac{P(D|m, b)P(b|m)P(m)}{P(D)} \end{aligned} \tag{5}$$

The log-likelihood distribution that our MCMC sampler needs to explore is the LHS $P(m, b|D)$. Next we find the PDFs for each of the components in the RHS of the above equation. The easiest to start would be $P(m|b)$ or $P(b|m)$. Assuming m and b can be drawn independently, these probabilities are not affected by the conditionality (see equation 6). If we also assume flat priors, $P(m) = P(b) =$ constant and need not be modelled.

$$\begin{aligned} P(m|b) &= P(m) \\ P(b|m) &= P(b) \end{aligned} \tag{6}$$

The only PDF in equation 5 that must be computed is $P(D|m, b)$. It took a little effort to find the right estimator for this distribution. Initially, I tried to model this as the inverse of the χ^2 of the fit. This was motivated by the fact that for any fit, we need to minimize the χ^2, which means

maximizing χ^{-2} (before Dr. David Hogg critisizes this, I had good reason to believe that the data trends were actually linear). Though my home grown sampler (seemingly) converged on the right parameter set, `emcee` failed to converge. This made me question the nature of this estimator, more so because I realised that I have not taken the fact that the errors are gaussian into account.

Using that argument, I thought that since the uncertainities on the data points are known to be Gaussian apriori, we can model the likelihood as the probability of drawing the original data from a normal distribution centered around the data points generated from the current parameters.

A little more clearly, given a parameter set (m, b), the predicted data can be written as y_pred $= mx + b$. If this was the *right* parameter set, the predicted data would deviate from the real data with only gaussian errors. Hence to estimate the likelihood of a parameter set, we can compute the probability of drawing each data point of the original data if it were drawn from a gaussian centered around the corresponding data point in the current data set. If that is still confusing to follow, equation 7 gives the line that generates this likelihood in the program.

$$P(D|m, b) = \texttt{norm.logpdf(y,mean=y\_pred,scale=sigma)} \tag{7}$$

It was more difficult to obtain a converging Markov chain with this estimator since it was very sensitive to deviations. After adding a burn in phase to the function and some more effort, I finally obtained a converging Markov chain. The acceptance ratio of the fit is ~ 0.38 and the autocorrelation time is ~ 1 showing that the Markov chain was well converged. `emcee` also converged within a few hundred steps with a hundred walkers. The results of this fitting process are shown in figures 5, 4, and 6.

The results of running `emcee` for the fit is shown in figure 7. The error bars on the final parameters obtained by running `emcee` are larger than the error bars reported by running the home grown M-H MCMC sampler. I think this has something to do with how `corner.py` computes the errors, and is not an indication that the home grown sampler is more efficient than `emcee`. Give that the data inherently has gaussian errors of $\sigma = 2$, it would not be possible to determine the slope and intercept to the accuracy shown by `corner` in figure 6.

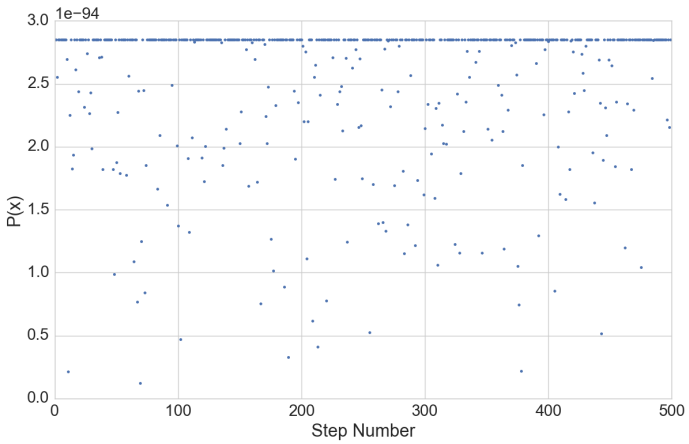

Figure 4: The log likelihood of the first 500 samples after the burn in phase. The probabilities of the samples appear random, indicating that the sampler must be drawing independent samples.

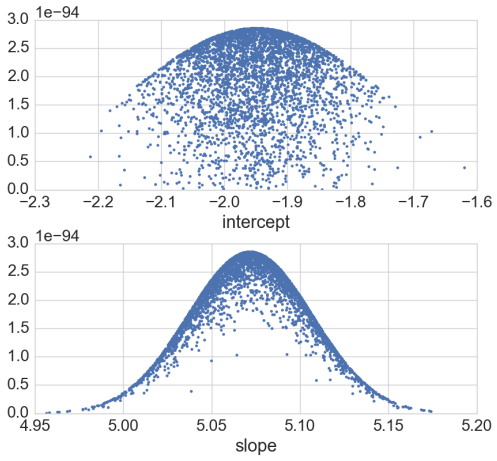

Figure 5: The final PDFs obtained for the slope and intercept. Interestingly, the slope shows a gaussian like nature but the intercept appears to have a wider distribution. Both the PDFs have a single peak, centered close to the actual values of $m = 5$ and $b = -2$. This is another confirmation that the home grown sampler resulted in a converging markov chain.

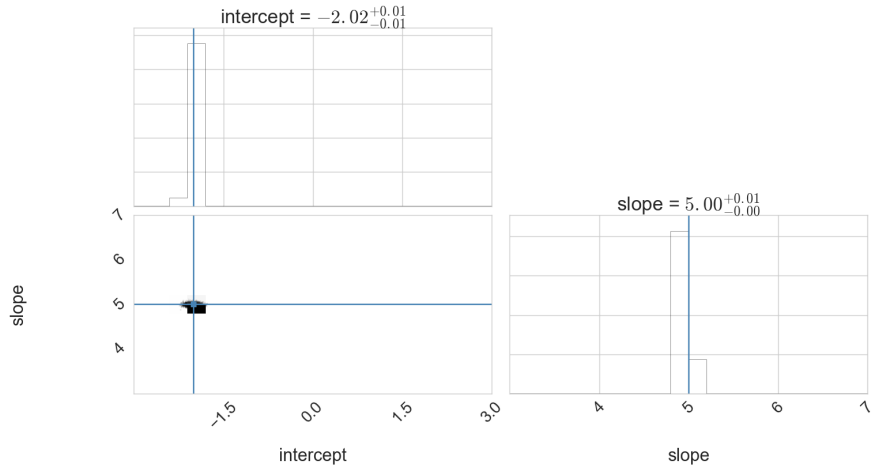

Figure 6: A contour plot generated using `corner.py` to show the marginalized probabilities of (m, b). The peak of the distribution is indicated above the corresponding histogram, but I think the errors reported on those parameters are not indicative. The contours show that the walker has not explored a very wide parameter space.

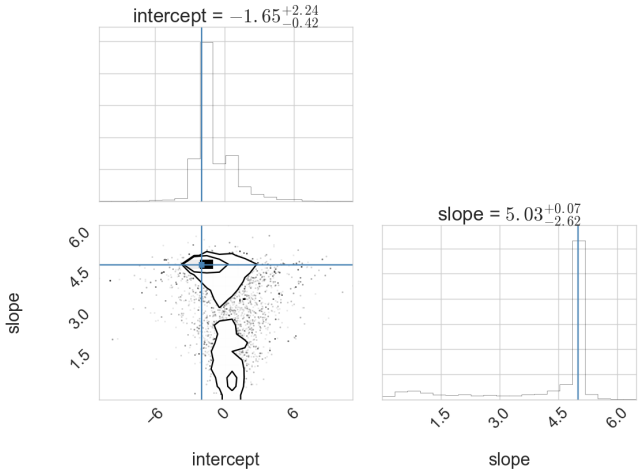

Figure 7: The results obtained by running `emcee` on the log likelihood distribution.