AM 207: Homework 2

Verena Kaynig-Fittkau and Pavlos Protopapas
Due: 11.59 P.M. Thursday March 3rd, 2015

Instructions:

- Upload your answers in an ipython notebook to Canvas.
- We will provide you imports for your ipython notebook. Please do not import additional libraries.
- Your individual submissions should use the following filenames: AM207_YOURNAME_HW2.ipynb
- Your code should be in code cells as part of your ipython notebook. Do not use a different language (or format).
- **Do not just send your code. The homework solutions should be in a report style. Be sure to add comments to your code as well as markdown cells where you describe your approach and discuss your results.**
- Please submit your notebook in an executed status, so that we can see all the results you computed. However, we will still run your code and all cells should reproduce the output when executed.
- If you have multiple files (e.g. you've added code files or images) create a tarball for all files in a single file and name it: AM207_YOURNAME_HW2.tar.gz or AM207_YOURNAME_HW2.zip

Have Fun!

In [8]:
```
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline

import seaborn as sns
sns.set_style("white")

import time
import timeit

import scipy.stats
import pandas as pd
import pymc as pm

import re
import numpy as np
```

Problem 1: Geweke Convergence Test

In the lecture we have seen the Geweke test as one option to test for convergence of our metropolis hastings chain. Describe in your own words how the Geweke test works and its limitations.

We provide you with the following code for the Geweke test. Write comments for the marked lines and any lines you think would be good to explain in addition.

Write a short explanation of what the `rhot` function does and why it is necessary.

In [4]:
```
#
def rhot(x, t):
    n = len(x)
    return np.corrcoef(x[0:(n-t)], x[t:n])[0,1]

#
def Geweke(trace, intervals, length):
    nsl=length
    #
    jump = int(0.9*len(trace)/(2*intervals))
    #
    first = 0.1*len(trace)

    z =np.empty(intervals)
    #
    for k in np.arange(0, intervals):
        #
        baga = np.int(first+k*jump)
        #
        bagb = len(trace)/2 + k*jump

        #
        sub_trace_a = trace[baga:baga+nsl]
        sub_trace_b = trace[bagb:bagb+nsl]

        #
        theta_a = np.mean(sub_trace_a)
        theta_b = np.mean(sub_trace_b)
        rho_a, rho_b = 1.0, 1.0
        #
        for i in xrange(int(0.1*nsl)):
            #
            rho_a += 2*rhot(sub_trace_a, i+1)
            rho_b += 2*rhot(sub_trace_b, i+1)

        #
        var_a  = np.var(sub_trace_a)*rho_a/length
        var_b  = np.var(sub_trace_b)*rho_b/length

        #
        z[k] = (theta_a-theta_b)/np.sqrt( var_a + var_b)

    return z
```

Problem 2: Message Response Times

The file `hangout_chat_data.csv` contains the response times of your friend Mark to google hangout chat messages in seconds. Use a method of your choice to read the file into a data frame or a numpy array. Your goal for this problem is to model Mark's chat response

time distribution in a Bayesian framework.

The description of the data sounds like a Poisson distribution is a good choice for our likelihood. We have messages arriving independently of each other, and instead of the arrival time we consider the time it took Mark to respond to the messages.

- Load and describe the data by plotting a histogram of the response times.
- Derive and compute the maximum likelihood solution for a Poisson distribution.
- Compare this to the Bayesian solution with a prior of your choice and using your own implementation of Metropolis Hastings to sample from the posterior. Make sure to describe why you chose this prior, as well as the specifics of your Metropolis Hastings implementation.
- Analyze your sampling using traceplots and convergence tests. You can use the Geweke implementation given above.
- Compare your solution to a solution using the MCMC class in pymc and write a brief discussion. Which parameters does your implementation need that the pymc implementation can do without? How do the traceplots compare?
- Check your ML solution and the Bayesian solution against the data. If you know how you can use the posterior predictive for the Bayesian solution, otherwise you can use the MAP estimate or the expectation value of the posterior and compare that distribution to the data histogram.
- Was our model a good choice for this problem? If yes great, if not, come up with a different Bayesian model that is better capable of capturing the data and show that it works better.

Problem 3: Markov Chains

- Given the following transition matrix, examine if the corresponding Markov Chain is irreducible and aperiodic. Note: No formal proof necessary, but you should give a solid argumentation.

$$P = \begin{pmatrix} 0.0 & 0.4 & 0.6 & 0.0 & 0.0 \\ 0.65 & 0.0 & 0.35 & 0.0 & 0.0 \\ 0.32 & 0.68 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.12 & 0.88 \\ 0.0 & 0.0 & 0.0 & 0.56 & 0.44 \end{pmatrix}$$

Problem 4: The Evidence

In lectures we mostly concentrate on the likelihood and the prior and regard the evidence as a mere normalization factor. However, the evidence can be quite useful. In this problem you will compare different models by computing the evidence for each model, aka the probability that randomly selected parameters from a given model class would generate the data X|.

As our models we compare polynomials of degree 0 to 4. For example for degree 2 we have $y = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \epsilon$ where $\epsilon \sim N(0, \sigma)$ and $\theta = [a_0, a_1, a_2, \sigma]$. Assume that for all polynomials $a_i \sim \text{Expo}(1)$ and $\sigma \sim \text{Inverse Gamma}(1, 1)$.

Bayes' theorem states that:

$$P(\theta|x) = \frac{P(x|\theta)P(\theta)}{P(x)}$$

$P(\theta)$ is the prior, $P(x|\theta)$ is the likelihood, $P(x)$ is called the evidence, and $P(\theta|x)$ is the posterior.

Your tasks are:

- Write down the mathematical equation of the evidence in terms of the likelihood and the prior. You don't need to solve the integral analytically, just write down its formula.
- Now you have the evidence in the form of an integral. Solve it by using importance sampling. What is a good choice for your importance sampling distribution?
- Compare the evidence for polynomials of degree 0 to 4. Which polynomial wins?

In [7]:
```
# Here is your data for the problem:
data = np.array([[ -1.85519254,  -2.7009541 ],
       [  4.38291824,  19.61735369],
       [  2.29495208,   3.96481822],
       [  0.02075668,   8.00646088],
       [  0.54097177,   2.8872262 ]])

x=data[:,0]
y=data[:,1]
```

Problem 5: Which YouTube Videos to Watch

Youtube videos have a like and an unlike flag. We can use these up and down votes on the videos to determine if a video is worth watching. However, it is not immediately obvious how to rank a video with just 3 up and 0 down rankings against a video with 300 up and 100 down votes. We will address this problem using a Bayesian approach.

Build two Bayesian models for the average upvote rate of a video. Both models should use the same likelihood, but different priors. Use one prior where people in general are rather undecided about videos, and one where people tend to be very opinionated. Compare the resulting posteriors for each video. How does the different choice of prior change your results?

Given that there are so many videos on YouTube we want a really conservative way to decide if it is worth watching. Compute the 5th percentile for the posterior of each video and rank the videos according to this value. Is the 5th percentile a good indicator for the ranking? What are the benefits, what are potential drawbacks?

In [13]:
```
# Here is the [upvote, downvote] data for 4 different videos:
video_votes = np.array([[3,0],[300,100],[2,2],[200,100]])
```