AM 207: Homework 4

Verena Kaynig-Fittkau and Pavlos Protopapas
Due: 11.59 P.M. Thursday April 7th, 2016

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_HW4.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_HW4.tar.gz or AM207_YOURNAME_HW4.zip

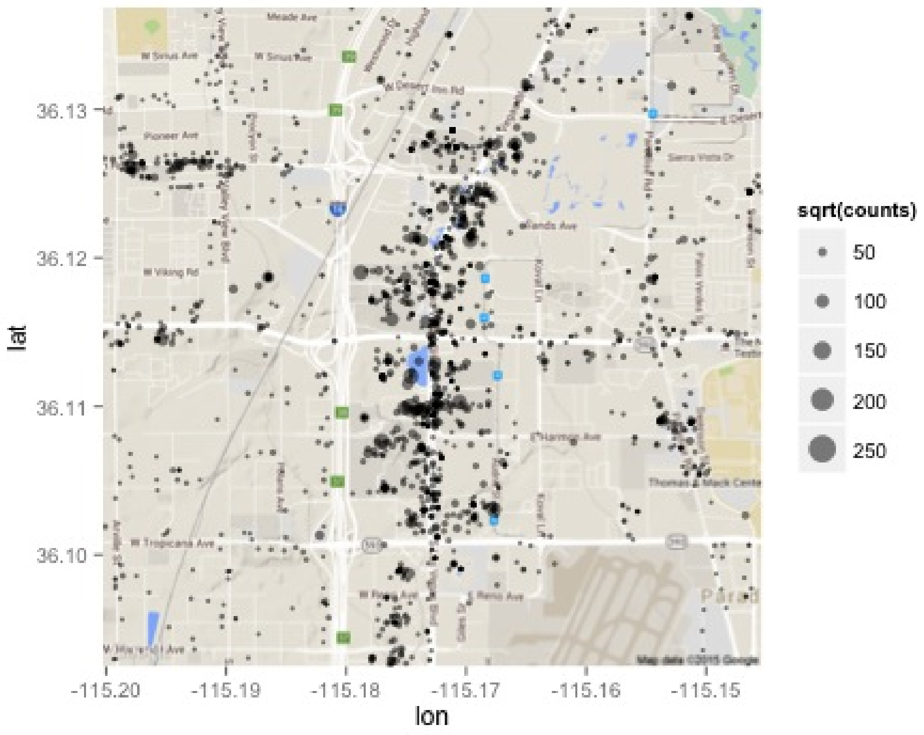
Have Fun!

In [1]:
```
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
```

```
C:\Users\vkaynig\Anaconda2\lib\site-packages\matplotlib\__init__.py:
872: UserWarning: axes.color_cycle is deprecated and replaced with a
xes.prop_cycle; please use the latter.
  warnings.warn(self.msg_depr % (key, alt_key))
```

Problem 1: Bayesian Models for Yelp Check-in Data

Below is a visualization of Yelp checkins on a portion of the Las-Vegas strip. The goal of this problem is to model an intensity surface on this map which is indicative of the propensity of an individual to check in at a particular location.



Such an intensity surface is very valuable for example to decide what would be good places to open another casino or restaurant.

The first hierarchical model will use a conjugate prior (and hence we can compute an analytical solution) and the second will use a non-conjugate set up with correlated priors.

Data description

We have M grid squares of width $w = 0.005$. The variable `checkins` contains M checkin values, one for each square. The variable `X` contains the coordinates of the center of each square. This data set is a subset of the map shown above.

In [5]:
```
#load checkins
checkins = np.genfromtxt('checkin_counts.csv', delimiter=',')
#load latitude and longitude values associated with each count value
X = np.genfromtxt('X.csv', delimiter=',')
#set M to the length of the checkins vector
M = checkins.size
#set grid width w to be .0005 latitude/longitude
w = .0005
```

Conjugate Model

In the first model, we assume that a map (i.e., the Las Vegas strip) is discretized into M grid squares of width w, and the checkins in cell i, denoted by Y_i, are distributed according to a $Pois(w^2\lambda_i)$ distribution. (That is, the check-ins follow a Poisson process conditioning on the λ, and $\lambda \in \mathbb{R}^m$ is the intensity surface we would like to infer.) Furthermore λ_i is drawn from a conjugate $Gamma(\alpha, \beta)$ distribution. The Y_i are conditionally independent on λ_i, and λ_i is indepndent of λ_j for $i \neq j$.

Derive the posterior distribution $p(\lambda_i|Y_i)$ for this model analytically as a function of hyperparameters α and β. Use the posterior mean as an estimate for λ_i, and then plot the posterior mean of λ_i on a grid for varying values of α and β. Discuss if this model leads to a useful intensity surface, and how the usefulness of the result is influenced by the choice of α and β.

Non-Conjugate Model

Now let's move on to a slightly more complicated model. Again assume that the grid is discretized into M cells of the same area, and the checkins in cell i, denoted by Y_i, are distributed according to a $Pois(w^2\lambda_i)$ distribution. However there is a twist: we assume λ_i is equivalent to $exp[\alpha + Z_i]$ where $Z \in \mathbb{R}^M$ is drawn from a multivariate normal $MVN(0, \Sigma)$ distribution, where Σ comes from a squared exponential kernel, and α is drawn according to a N(0,1) distribution. In this way, the λ_i for distinct cells are no longer independent but are correlated.(If you'd like to delve deeper, this process is called a "Log-Guassian Cox process".)

Describe what the meaning of α and Σ is for this model.

We model Σ using a square exponential kernel such that the covariances decay exponentially quickly as points get further apart from each other. That is

$$\Sigma_{ij} = Exp[-||X_i - X_j||_2^2/\phi]$$

Where X_i and X_j are the ith and jth rows of X respectively. Describe in which way the choice of ϕ influences the kernel. By default, use $\phi = .00001$.

To simplify your code, consider the prior parameters/latent variables (α, Z) jointly as a vector in $\mathbb{R}^{M+1}$. Determine the covariance matrix for the multivariate normal prior on this vector and store it all together in a variable called Sigma. Give a short explanation why we can do this. When you implement the computation of $Sigma$, small numerical instabilities can lead to a covariance matrix that is not positive-semidefinite. You can check this by attempting a Cholesky decomposition of Σ with numpy. If it throws an error, you can add a small offset to the diagonal of the matrix to help with numerical stability.

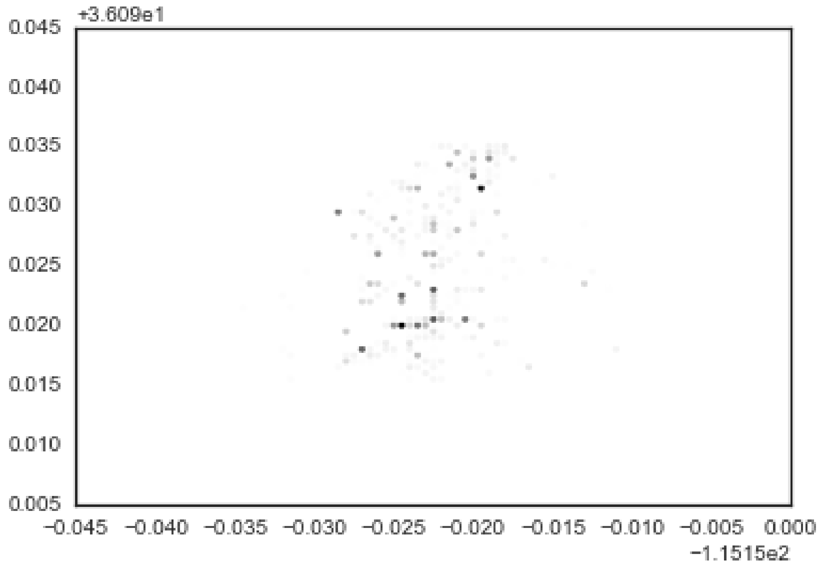
We will provide you with code for an elliptical slice sampler for this problem. This sampling method was directly devloped for problems like this and we will discuss it later in class. For now it is sufficient for you to use the code, without neccessarily understanding it. Use the provided sampler to estimate the posterior mean for this non-conjugate model. In addition implement the model also in PYMC or vanilla Metropolis Hastings, and compare the result.

Finally discuss the differences in results from the conjugate and the non-conjugate model and which one you would prefer.

Helper functions

Here is some code that should be useful:

In [10]:
```
# This function takes a vector of M intensities and plots it according
# The values of the intensities are normalized to [0,1] and darker mean
def plot_checkins(checkins):
    colors = checkins / np.double(np.max(checkins))
    plt.scatter(X[:,0], X[:,1], s=10, c=colors, cmap='gray_r', edgecolo
    plt.show()

plot_checkins(checkins)
```



In [11]:
```
#elliptical slice sampler (Murray, Adams, and MacKay JMLR 2010)
# Inputs:
# Y: observed data
# Sig: covariance matrix of the prior
# log_lik: the log likelihood function
# N_mcmc: number of desired samples
# burn_in: number of burn in iterations
# N: dimensionality of the input, this should be M+1 because alpha is p

# returns an array of samples of size N_mcmc x N


def ess(Y, Sig,log_lik, N_mcmc, burn_in, N):
    print 'running elliptical slice sampling...'
    mcmc_samples = np.zeros((N_mcmc+burn_in, N))
    norm_samples = np.random.multivariate_normal(np.zeros(N), Sig, N_mc
    unif_samples = np.random.uniform(0,1,N_mcmc+burn_in)
    theta = np.random.uniform(0,2*np.pi,N_mcmc+burn_in)
    theta_min = theta-2*np.pi
    theta_max = theta+2*np.pi
    for i in range(1,N_mcmc+burn_in):
        f = mcmc_samples[i-1,:]
        llh_thresh = log_lik(f,Y) + np.log(unif_samples[i])
        f_star = f*np.cos(theta[i])+norm_samples[i,:]*np.sin(theta[i])
        while log_lik(f_star,Y) < llh_thresh:
            if theta[i] < 0:
                theta_min[i] = theta[i]
            else:
                theta_max[i] = theta[i]
            theta[i] = np.random.uniform(theta_min[i],theta_max[i],1)
            f_star = f*np.cos(theta[i])+norm_samples[i,:]*np.sin(theta[
        mcmc_samples[i,:] = f_star
    return mcmc_samples[(burn_in+1):(burn_in+N_mcmc),]
```

Problem 2: Point Matching

In this problem you are given two point clouds `points1` and `points2`. Your task is to estimate an affine transformation which can map the points of `points1` onto the points of `points2`. Unfortunately you don't know which points from the two point sets correspond to each other.

We are modeling the correspondences in a correspondence matrix M, which is of size $n_1 \times n_2$ where n_i is the number of points in point set i. $M_{ij} = 1$ indicates that point x_i from `points1` and y_j from `points2` correspond to each other. Each point is only allowed to have one correspondence. So for all i we have $\sum_l^{n_2} M_{il} = 1$

The likelihood for this problem is:

$$\prod_i^{n_1} \prod_j^{n_2} \varphi_{0,\sigma}\left(||Ax_i - y_j||^2\right)^M_{ij}$$

where $\varphi_{0,\sigma}$ is a normal distribution with mean 0 and variance σ^2. We use $\sigma^2 = 0.1$ as a default for this problem.

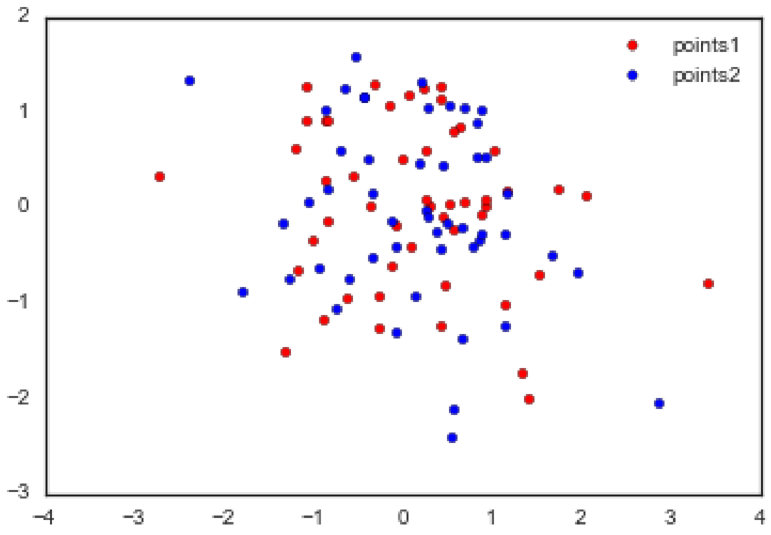
Note: The M_{ij} is not rendering correctly in my notebook. The indices ij are supposed to belong to M. Please look at the latex code for clarification.

Estimate the affine transform A that matches the two point sets using expectation maximization and simulated annealing and compare the results. What are the strength of each algorithm? Which one would you prefer in which scenario?

Here are some hints that should be useful:

- The M-step of the problem will be a weighted least squares solution.
- To use linear algebra to solve the M step it is useful to extend the point matrices to contain all possible pairwise correspondence pairs.
- The weights for the weighted least squares should be updated in your E-step

```
In [24]: points1 = np.array([[-0.855,  0.905,  1.   ],[ 1.417, -1.999,  1.   ],[
                             [ 2.055,  0.131,  1.   ],[ 0.694,  0.056,  1.   ],[
                             [-1.081,  0.91 ,  1.   ],[-0.273, -1.258,  1.   ],[
                             [-0.007,  0.507,  1.   ],[-0.262, -0.922,  1.   ],[
                             [ 0.528,  0.036,  1.   ],[ 0.31 ,  0.025,  1.   ],[
                             [-0.839, -0.137,  1.   ],[ 0.924,  0.012,  1.   ],[
                             [ 1.345, -1.731,  1.   ],[-0.362,  0.012,  1.   ],[
                             [ 0.446, -0.088,  1.   ],[-1.006, -0.335,  1.   ],[
                             [ 1.525, -0.699,  1.   ],[ 0.574,  0.805,  1.   ],[
                             [-0.306,  1.291,  1.   ],[-0.07 , -0.174,  1.   ],[
                             [-0.845,  0.914,  1.   ],[ 1.136, -1.001,  1.   ],[
                             [-0.617, -0.941,  1.   ],[ 0.438,  1.147,  1.   ],[
                             [ 0.923,  0.096,  1.   ],[ 0.563, -0.235,  1.   ]])

         points2 = np.array([[-0.131, -0.134,  1.   ],[-2.402,  1.347,  1.   ],[
                             [ 0.283,  1.049,  1.   ],[-0.734, -1.058,  1.   ],[
                             [ 0.211,  1.31 ,  1.   ],[ 0.874,  1.023,  1.   ],[
                             [-0.073, -1.308,  1.   ],[-1.058,  0.075,  1.   ],[
                             [ 0.666, -1.36 ,  1.   ],[ 0.698,  1.059,  1.   ],[
                             [ 1.141, -1.229,  1.   ],[ 0.89 , -0.265,  1.   ],[
                             [-0.686,  0.603,  1.   ],[ 0.502, -0.169,  1.   ],[
                             [-0.33 ,  0.149,  1.   ],[-0.431,  1.168,  1.   ],[
                             [-0.872,  1.033,  1.   ],[ 0.378, -0.252,  1.   ],[
                             [ 1.147, -0.275,  1.   ],[-0.073, -0.41 ,  1.   ],[
                             [-0.444,  1.163,  1.   ],[ 0.459,  0.452,  1.   ],[
                             [ 0.838,  0.524,  1.   ],[ 0.188,  0.471,  1.   ],[
                             [-0.382,  0.514,  1.   ],[ 2.857, -2.04 ,  1.   ]])

         plt.scatter(points1[:,0], points1[:,1], c='r', label='points1')
         plt.scatter(points2[:,0], points2[:,1], c='b', label='points2')
         plt.legend()
         plt.show()
```



Problem 3 Time Series

Below you find 200 data points from a time series. Your task is to model this data using a simple autoregressive model:

$$X_t = \alpha X_{t-1} + r_t$$

where r_t is normally distributed: $r_t \sim \mathcal{N}(0, \sigma^2)$

Here are your tasks:

- Implement a maximum likelihood estimation(MLE) method to estimate α
- Use any MCMC algorithm of your choice to infere σ^2 and α
- Forecast 3 time points ahead and provide posteriors for these three points

In [28]:

```
# Here is the data
y=np.array([0.00000000e+00,  -2.41884381e-01,  -2.19815584e-01, -5.9265
         6.09244734e-02,  -1.36663360e+00,  -1.63887006e+00, -1.3172135
        -6.63029664e-01,  -5.97693396e-01,   1.26359198e-01, 6.55116724
         4.04113046e-01,   7.29893878e-01,  -1.96731327e-03, 4.64230686
        -4.03256074e-01,  -8.70131096e-01,  -8.25921060e-01, 4.69417714
        -1.05978178e-01,  -1.48916430e-01,  -5.43573786e-02, 4.61970441
         1.26715777e+00,   6.41828280e-01,   8.77364512e-01, 3.67214345
         4.28449069e-01,   7.31450267e-01,   6.19131504e-01, 6.33373751
        -2.48995087e-02,  -2.53030153e-01,  -2.15503832e-01,-6.20238803
         3.32834574e-01,   4.01426008e-01,  -6.37956657e-01,-4.52884655
        -6.19309194e-01,  -4.78009274e-01,  -2.32960888e-01, 3.80860392
        -1.71939822e-01,  -2.22714248e-02,   1.69698173e-01, 2.26875861
        -2.00412521e-01,   1.62853318e-02,  -1.64492388e-02,-5.92514844
        -1.07109089e+00,  -5.47941202e-01,  -6.80501959e-01,-1.22073393
        -6.30383282e-01,  -8.67182207e-01,  -9.67761290e-01,-1.79674059
         2.28199545e-01,   4.21792446e-01,   1.69467875e-01,-2.59645693
        -6.05873935e-01,  -6.11037064e-01,   1.01647103e-01,-1.44991643
        -2.85514261e-01,  -3.95586179e-01,  -3.57194799e-01,-2.20170569
         7.41558107e-01,   8.33120964e-01,   3.73995334e-01, 4.83141624
         5.78378631e-01,   1.43550174e-01,  -1.31226657e-01,-5.30885377
        -5.02379201e-01,  -5.14585915e-01,  -4.64397849e-01, 1.65475660
        -8.60566293e-01,  -2.36405434e-01,  -4.12606502e-01,-5.58566210
         4.41653164e-01,   1.59249052e-01,  -1.74168536e-01, 5.63049814
         2.79509214e-01,  -6.26883609e-01,  -4.44982987e-01,-8.69359696
        -1.35445853e+00,  -1.12329955e+00,  -4.94572306e-01,-3.40507834
        -7.69088724e-01,  -4.77378429e-01,   4.52315984e-02, 8.76114754
         6.32190755e-01,   8.18383124e-01,   2.95975304e-01,-2.38797237
        -3.15966789e-01,  -3.16808474e-01,  -1.16651970e+00,-1.01729199
        -7.65156825e-01,  -5.10469718e-01,  -3.25334859e-01, 1.06358791
         5.87793605e-01,   4.38191557e-01,  -4.17879322e-01,-3.19715350
         3.69746949e-01,   9.62398763e-02,  -2.81485049e-01,-3.37776564
         6.94986165e-01,   2.41137851e-01,   2.79478944e-01, 5.13500357
         1.25038407e+00,   8.17699739e-01,   8.78811468e-02,-1.20413802
         4.16043086e-01,   6.45585187e-02])
```

Extra Problem 4: Final Project Proposal

Think about your final project proposal. Your final project is your chance to show off what you learned in AM207! You have seen a lot of different models and examples in lecture and in the homework assignments. In your final project you are free to apply all the techniques you learned to an area of your choice. Be careful not to choose data that you still need to label yourself or that will take you ages to clean. The main focus should be on finding a good model and comparing different techniques for sampling, optimization, etc.

Please note that there will be a google spreadsheet where you can submit your final project proposal abstract. No need to write it into this notebook.

The abstract should be short, but contain a short description of your final project idea, which methods you are planning on using, and where your data will come from.

```
In [ ]:
```