PHYS689
Homework 7
Will Wainwright
Repository: https://github.com/wjwainwright/PHYS689

Discussion

This homework took me a bit to grasp conceptually and longer to implement it in Python. I used Scipy to create a multivariate Gaussian and draw 5000 (x,y) points from that distribution. I then gave these points to the MCMC to fit. The original distribution had a mean of $\mu_x = 1$ and $\mu_y = 9$. After 10000 steps through the MCMC with a proposal width of 0.1, I plotted the results. The posterior distribution has median values of $\mu_x = 1.038045127251182$ and $\mu_y = 8.879400073855749$. The standard deviation on those values is 1.5362625820473927 and 1.9741523862764494 respectively. These standard deviations are quite high, partly because of the starting point of (0,0), but also because the MCMC tended to drift around the actual value quite a lot compared to example MCMCs that I have seen. The drifting can be seen in Figure 4. The outcome of the MCMC seems quite variable, and some iterations the median value is farther from the true value than in this case. Regardless, the true value tends to be within a standard deviation of the median, and I would consider that to be a success.

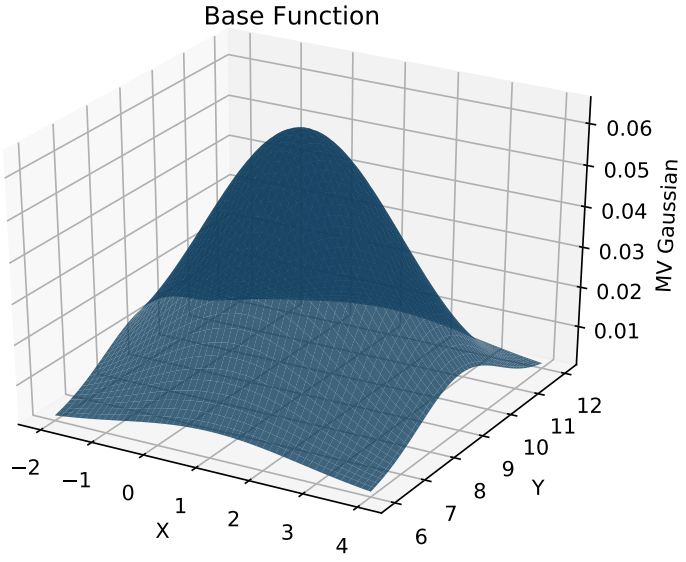

Figure 1: Base PDF used to generate the random samples in Figure 3. Multivariate Gaussian with a mean of [1,9] and a variance of [3,2].

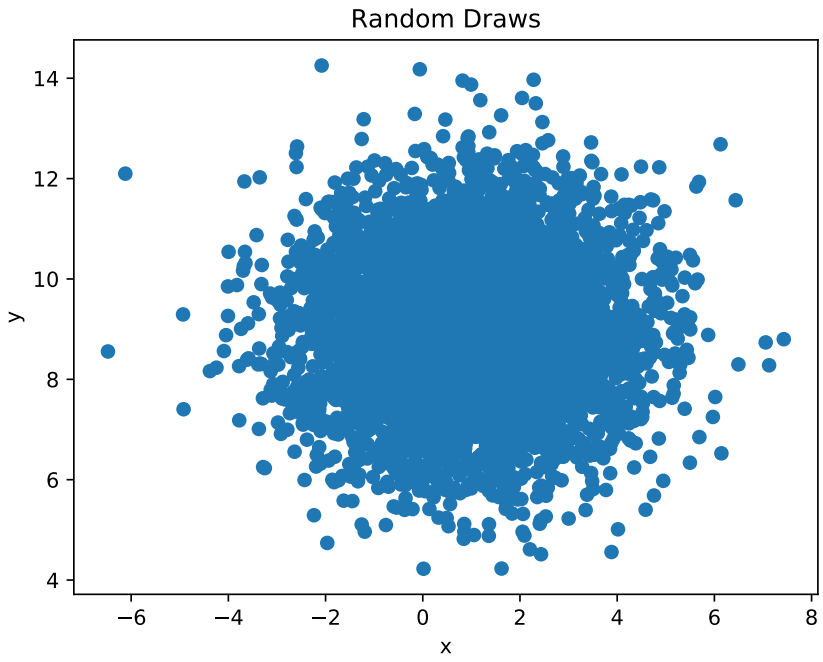

Figure 2: Draws of x & y from the PDF in Figure 1. These points were used to calculate the likelihood in the MCMC.

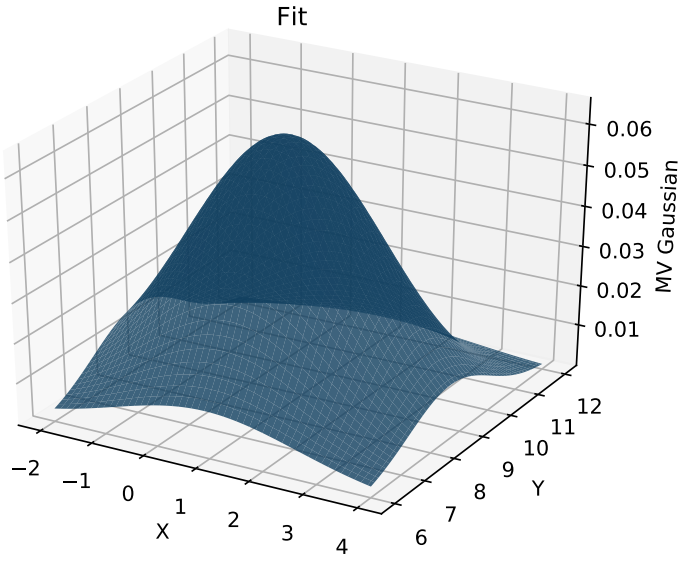

Figure 3: Plot of the fit parameters using the same method used to create Figure 1. The central location of the Gaussian is slightly off compared to in Figure 1.

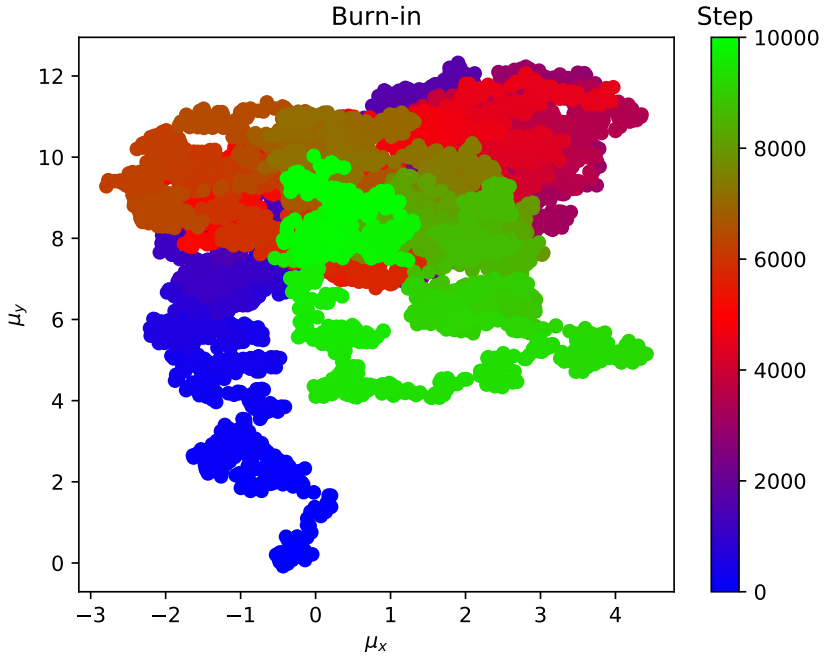

Figure 4: Plot showing the value of μ_x and μ_y as a function of the step in the MCMC. The value tends to drift around the actual value of [1,9]. I would say that the burn-in time is at least 2500 samples, but given the variability seen around the true value, I would say 5000 samples are needed to draw a more reliable median or mean that is not as skewed by the first 2000 points.

```
# -*- coding: utf-8 -*-

import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import multivariate_normal as mv
from scipy.stats import norm

func = mv(mean=[1,9],cov=[[3,0],[0,2]])

x, y = np.mgrid[-2.0:4.0:100j, 6.0:12.0:100j]
xy = np.column_stack([x.flat,y.flat])
z = func.pdf(xy)
z = z.reshape(x.shape)

from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = fig.add_subplot(111,projection='3d')
ax.plot_surface(x,y,z,alpha=0.8)
ax.set_title('Base Function')
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('MV Gaussian')
plt.savefig('pdf.pdf')


def mcmc(samp,z,init,cov,prior_mu,samples=1000,proposal_width=0.1):

    posterior = np.empty((0,len(init)))

    prior_std = cov

    var_current = []
    for var in init:
        var_current.append(var)

    for sample in range(samples):

        post = []

        for i in range(len(init)):


            var_proposed = norm(var_current[i],proposal_width).rvs()

            params_prop = [a for a in var_current]
            params_prop[i] = var_proposed


            #Assuming sigma=1 for the simplest case


            likelihood_current = mv(mean=var_current,cov=cov).pdf(samp)
            likelihood_proposed = mv(mean=params_prop,cov=cov).pdf(samp)

            prior_current = mv(mean=prior_mu,cov=prior_std).pdf(var_current)
            prior_proposed = mv(mean=prior_mu,cov=prior_std).pdf(params_prop)
```

```
            p_current = np.sum([np.log(a) for a in likelihood_current])*prior_current
            p_proposed = np.sum([np.log(a) for a in likelihood_proposed])*prior_proposed
            #print(p_proposed/p_current)
            accept = p_proposed / p_current > np.random.rand()

            if accept:
                #print(f"{i}  |  {var_current}  ->  {var_proposed}")
                var_current[i] = var_proposed
            post.append(var_current[i])
        posterior = np.r_[posterior,[post]]
    return posterior

samp = func.rvs(5000)

plt.figure()
plt.scatter(samp[:,0],samp[:,1])
plt.title('Random Draws')
plt.xlabel('x')
plt.ylabel('y')
plt.savefig('draws.pdf')



post = mcmc(samp,z,[0,0],[[3,0],[0,2]],[1,9],samples=10000)

mu = [np.mean(post[:,0]),np.mean(post[:,1])]

fit = mv(mean=mu,cov=[[3,0],[0,2]])

fitz = fit.pdf(xy)
fitz = fitz.reshape(x.shape)

fig = plt.figure()
ax = fig.add_subplot(111,projection='3d')
ax.plot_surface(x,y,fitz,alpha=0.8)
ax.set_title('Fit')
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('MV Gaussian')
plt.savefig('fit.pdf')

color = np.linspace(0,len(post),len(post))
plt.figure()
plt.title('Burn-in')
plt.xlabel('$\mu_x$')
plt.ylabel('$\mu_y$')
plt.scatter(post[:,0],post[:,1],c=color)
plt.set_cmap('brg')
clb = plt.colorbar()
clb.ax.set_title("Step")
plt.savefig('burn.pdf')
```