hw10

December 8, 2018

```
In [60]: import numpy as np
         import matplotlib.pyplot as plt
         import matplotlib
         import seaborn as sns
         matplotlib.rcParams['figure.figsize'] = [15, 8]
```

```
/home/cfizette/anaconda3/lib/python3.6/importlib/_bootstrap.py:219: RuntimeWarning: numpy.dtype
  return f(*args, **kwds)
```

1 6b

1.0.1 Posterior distribution

$$\pi(\theta|x) \propto \exp(-\frac{n}{2}(\theta - \bar{X}_n)^2) \cdot \frac{1}{\theta^2 + 1}$$

```
In [10]: samp = [-1.249, 3.134, 0.825, 0.036, 0.814, -0.105]
         n = len(samp)
```

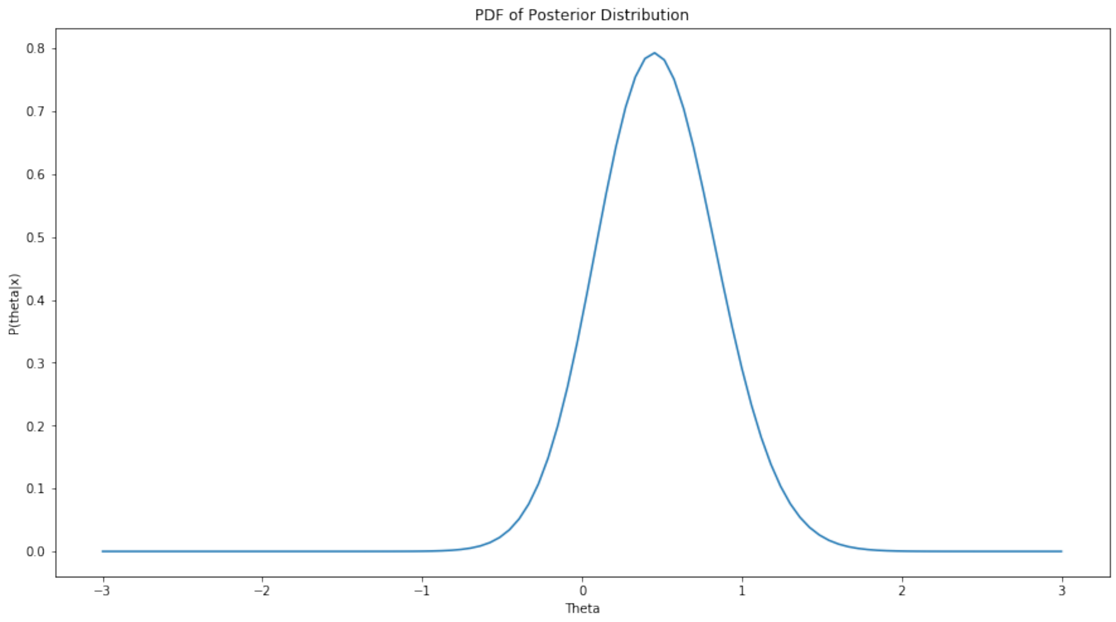
```
In [6]: x_bar = np.mean(samp)
```

```
In [25]: thetas = np.linspace(-3,3,100)
```

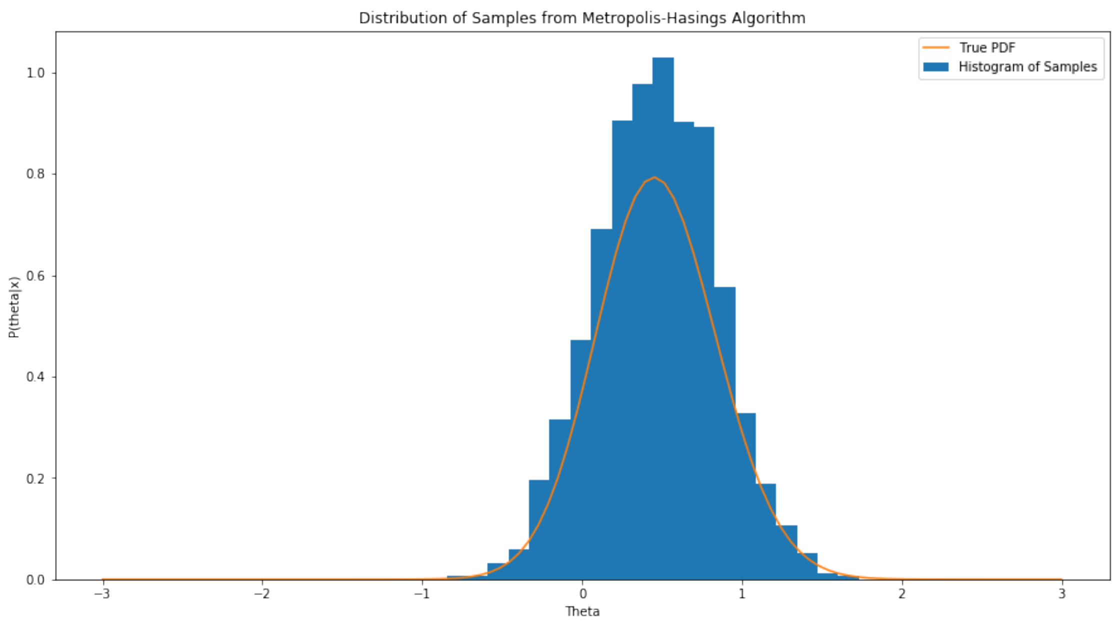
```
In [26]: pdf = np.exp(-(n/2)*(thetas - x_bar)**2)*(1/(np.power(thetas,2) + 1))
```

```
In [27]: plt.plot(thetas, pdf)
         plt.title('PDF of Posterior Distribution')
         plt.xlabel('Theta')
         plt.ylabel('P(theta|x)')
```

```
Out[27]: Text(0,0.5,'P(theta|x)')
```



```
In [32]: def f(x):
             return np.exp(-(n/2)*(x - x_bar)**2)*(1/(np.power(x,2) + 1))

In [33]: def q(x, mu):
             return np.exp(-3*(x-mu)**2)

In [36]: def rho(x,y):
             ratio = (f(y)*q(x,y))/(f(x)*q(y,x))
             return min(ratio, 1)

In [62]: sigma=1/6

In [63]: X = [0]
         for i in range(6000):
             y = np.random.normal(X[i], sigma)
             rho_ = rho(X[i],y)
             X_new = float(np.random.choice([y,X[i]],1, p=[rho_, 1-rho_]))
             X.append(X_new)
         X = np.array(X[1000:])

In [76]: plt.hist(X, normed=True, bins=20, label='Histogram of Samples')
         plt.plot(thetas, pdf, label='True PDF')
         plt.title('Distribution of Samples from Metropolis-Hasings Algorithm')
         plt.xlabel('Theta')
         plt.ylabel('P(theta|x)')
         plt.legend(loc='best')

Out[76]: <matplotlib.legend.Legend at 0x7f86ad890be0>
```



2 6c

```
In [73]: e_x = np.mean(X)
```

```
In [85]: p_0_1 = len(X[(0 < X) & (X < 1)])/len(X)
```

```
In [87]: print("E(theta|x) = {}".format(e_x))

E(theta|x) = 0.46470057589611125
```

```
In [89]: print("P(0 < theta < 1 | x) = {}".format(p_0_1))

P(0 < theta < 1 | x) = 0.8112377524495101
```

```
In [91]: np.mean(samp)

Out[91]: 0.5758333333333333
```

We see that the bayesian expected value of theta is less than the frequentist expectation. This is due to us using a non-uniform prior distribution for theta.