Monte Carlo Methods

July 3, 2018

In [1]:
```
%matplotlib inline
import time
import pylab as plt
import numpy as np
from IPython import display
plt.rcParams['figure.figsize'] = (10, 4)

def start_plot(N):
    fig, ax = plt.subplots(1, 2, figsize=(10, 4))
    ax[0].set_xlim(-1.2, 1.2)
    ax[0].set_ylim(-1.2, 1.2)
    ax[1].set_xlim(0, N)
    ax[1].set_ylim(3, 3.4)
    ax[1].axhline(np.pi, linestyle='-', color='red')
    ax[0].add_artist(plt.Circle((0, 0), 1.0, fill=False, linestyle='-'))
    return ax
def plot_step(ax, points, i, pi):
    ax[0].plot([points[i:i+10, 0]], [points[i:i+10, 1]], marker='o',
               markersize=1, color='red')
    ax[1].plot([i+1], [pi], marker='x', markersize=3, color='blue')

    display.clear_output(wait=True)
    display.display(plt.gcf())

    time.sleep(.1)
def random_points(N):
    return np.random.rand(N, 2) * 2 - 1
def count_in_circle(points):
    return np.count_nonzero(points[:, 0] ** 2 + points[:, 1] ** 2 < 1)

def f(x):
    return 1/np.sqrt(2*np.pi*0.2) * np.exp(-(x)**2/(2*.2))

def start_plot_2(N):
    fig, ax = plt.subplots(1, 2, figsize=(10, 4))
    ax[0].set_xlim(-1, 1)
    ax[0].set_ylim(0, 1)
    ax[1].set_xlim(0, N)
```

```
    ax[1].set_ylim(.5, 1.5)
    #ax[1].axhline(np.pi, linestyle='-', color='red')
    x = np.linspace(-2, 2, 500)
    ax[0].plot(x, f(x), linestyle='-', color='blue')
    return ax
def random_points_2(N):
    p = np.random.rand(N, 1)
    p[:] = p[:] * 2 - 1
    return p
def plot_step_2(ax, points, i, pi):
    ax[0].plot(points[i:i+10], f(points[i:i+10]), marker='x',
               markersize=5, color='red', linestyle='')
    ax[1].plot([i+1], [pi], marker='x', markersize=3, color='blue')

    display.clear_output(wait=True)
    display.display(plt.gcf())

    time.sleep(.1)
```

1 Monte Carlo Methods

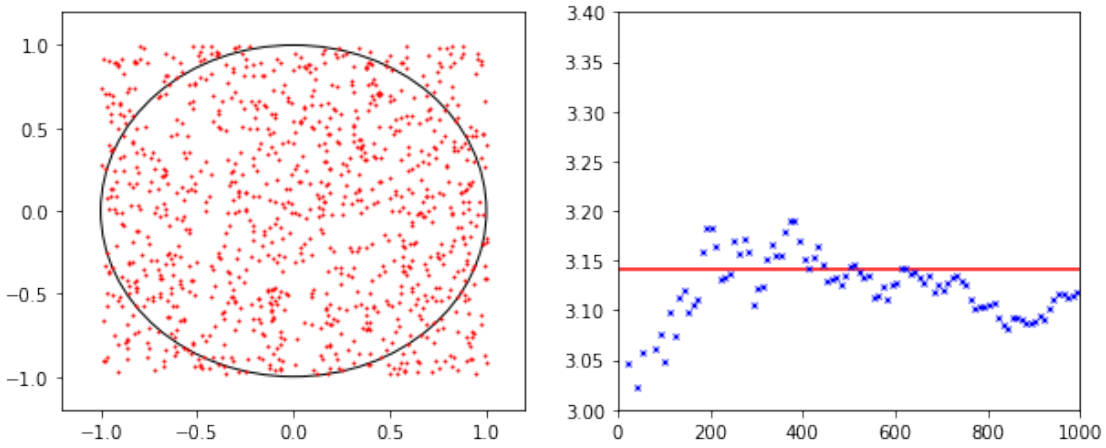
Raphael Michel, 2018-07-03

1.1 Contents

- Introduction to Monte Carlo Integration
- Importance Sampling and Metropolis-Hastings
- Monte Carlo simulation for lattice models
- Monte Carlo in parameter estimation

1.1.1 Let's calculate Pi

In [2]:
```
N = 1000
ax = start_plot(N)
points = random_points(N)
for i in range(0, N, 10):
    M = count_in_circle(points[:i + 1])
    pi = 4 * M / (i + 1)
    plot_step(ax, points, i, pi)
plt.close()
pi
```




```
Out[2]: 3.1200807265388497
```

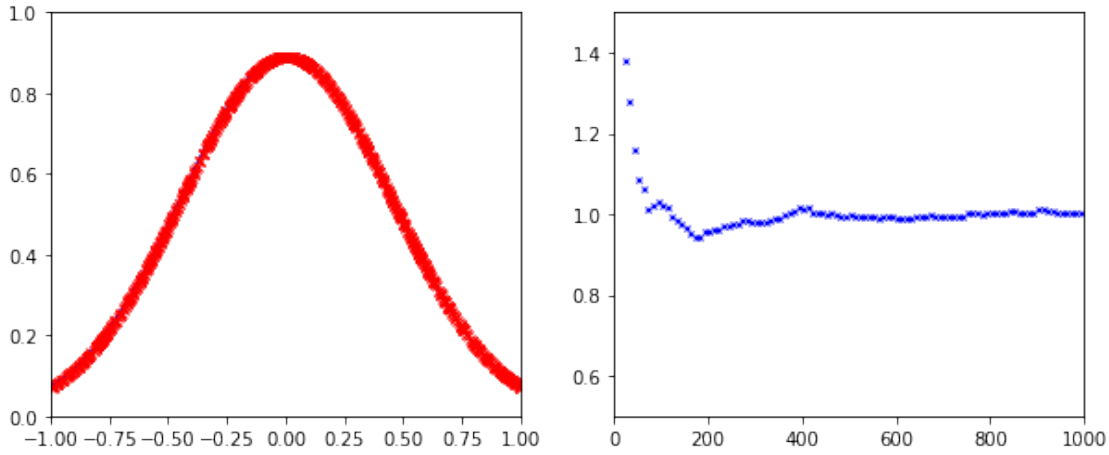
```
In [3]: N = 10_000_000
        points = random_points(N)
        M = count_in_circle(points)
        pi = 4*M/N
        "{}  error: {:.5f}%".format(pi, (pi-np.pi)/pi * 100)
```

```
Out[3]: '3.14183  error: 0.00755%'
```

1.2 Are there useful things to use this for?

1.3 Absolutely!

```
In [4]: N = 1000
        ax = start_plot_2(N)
        points = random_points_2(N)
        for i in range(0, N, 10):
            integral = 2 * np.sum(f(points[:i + 10])) / (i + 1)
            plot_step_2(ax, points, i, integral)
        plt.close()
        integral
```



Out[4]: 1.0043570481894024

1.4 Slightly more formal

Consider

$$I = \int_V f(\mathbf{x}) \mathrm{d}^d \mathbf{x}. \tag{1}$$

With N uniformly drawn random vectors $\mathbf{x}_i \in V$, we can calculate

$$I_N = \frac{V}{N} \sum_i^N f(\mathbf{x}_i) \tag{2}$$

and we get

$$\lim_{N \to \infty} I_N = I \tag{3}$$

1.5 Can we quantify the error here?

Let's define

$$y_i = V f(\mathbf{x}_i) \tag{4}$$

and write

$$I_N = \frac{V}{N} \sum_i^N f(\mathbf{x}_i) \equiv \frac{y_1 + y_2 + \ + y_N}{N} \tag{5}$$

This is a sum of *independent* and *identically distributed* random variables.

1.6 Refresher: Central Limit Theorem

Lindeberg–Lévy: Suppose $\{X_1, X_2, , X_n\}$ is a sequence of independent and identically distributed random variables with $\mathrm{E}[X_i] =$ and $\mathrm{Var}[X_i] = \sigma^2 < .$ Then as n approaches infinity, the random variables $\sqrt{n}\left(\left(\frac{1}{n}\sum_{i=1}^n X_i\right) - \mu\right)$ converge in distribution to a normal $\mathrm{N}(0, \sigma^2)$.

$\implies$ $I_N \to I + \frac{\varepsilon}{\sqrt{n}}$ with ε drawn from $\mathrm{N}(0, \sigma^2)$.

1.7 How does this compare to other methods for numerical integration?

Usually, we divide our integration domain into n intervals on every dimension, i.e. $N = n^d$ points.

Error of Simpson's rule: $1/n^4 = 1/(N^{4/d})$

In one dimension, this is way better than Monte Carlo! Starting with $d = 9$, Monte Carlo clearly wins.

1.8 Importance sampling and Metropolis-Hastings

(Blackboard)

1.9 Monte Carlo simulation of lattice models

Assume field $\phi_{\mathbf{x}}$ at each lattice site $\mathbf{x}$. Assuming canonical ensemble:

$$Z = \int \exp\left(-\frac{H(\phi)}{kT}\right) \mathrm{d}\phi_1 \mathrm{d}\phi_2 \ldots \mathrm{d}\phi_N \tag{6}$$

Often, we are interested in thermal averages like

$$\langle A \rangle = \frac{1}{Z} \int A(\phi) \exp\left(-\frac{H(\phi)}{kT}\right) \mathrm{d}\phi_1 \mathrm{d}\phi_2 \ldots \mathrm{d}\phi_N \tag{7}$$

Importance sampling with $p(x) \propto \exp\left(-\frac{H(\phi)}{kT}\right)$

$\implies$ Metropolis-Hastings with

$$r = \min\left(1, \exp\left(-\frac{H(\phi') - H(\phi)}{kT}\right)\right) \tag{8}$$

1.10 Example: Ising Model

$$Z = \sum_{\{s_{\mathbf{x}}\}} \exp\left(-\beta\left(\frac{1}{2}\sum_{\langle \mathbf{x},\mathbf{y}\rangle}(1 - s_{\mathbf{x}}s_{\mathbf{y}}) + B\sum_{\mathbf{x}} s_{\mathbf{x}}\right)\right) \tag{9}$$

with $s_{\mathbf{x}} = \pm 1$, $\beta = 1/(kT)$.

Metropolis-Hastings step: Select a random lattice site $\mathbf{x}$ and propose to switch the spin to $s'_{\mathbf{x}} = -s_{\mathbf{x}}$, then compute $\delta E_{\mathbf{x}} = E_{\mathbf{x}}(s'_{\mathbf{x}}) - E_{\mathbf{x}}(s_{\mathbf{x}})$ locally and accept with

$$r = \min\left(1, \exp\left(-\beta \delta E_{\mathbf{x}}\right)\right) \tag{10}$$

Once we have a very long Markov chain, we can compute thermodynamic quantities by just averaging $A(\phi)$ over the chain.

1.11 Monte Carlo Markov Chains in parameter estimation

We want to use some data

$$\mathbf{z} = (z_1, z_2, , z_n) \tag{11}$$

to infer some parameters

$$\grave{} = (\theta_1, \theta_2, , \theta_n) \tag{12}$$

Our model usually gives us $p(\mathbf{z}|\theta)$, but we are looking for $p(\theta|\mathbf{z})$.

1.11.1 Bayes theorem

1.11.2 Sampling the posterior with Metropolis-Hastings

1. Choose a proposal function $q(\theta'|\theta)$
2. Use acceptance probability:

$$r = \min\left(1, \frac{p(\theta')p(\mathbf{x}|\theta')q(\theta|\theta')}{p(\theta)p(\mathbf{x}|\theta)q(\theta'|\theta)}\right) \tag{13}$$

3. (as before)

1.12 Thank you very much!

1.12.1 Any questions?

1.13 References

- D. P. Landau, K. Binder, A Guide to Monte Carlo Simulations in Statistical Physics 2013 (3rd Edition)
- V. Springel, F. Gräter, Lecture Notes for "Fundamentals of Simulation Methods", 2017
- U. Schwarz, Lecture Notes for "Statistical Physics", 2016
- L. Amendola, Lecture notes for "Statistical Methods", 2017