GeneratingRandomWalksHintsPython

January 9, 2024

1 Generating random walks

(Sethna, “Entropy, Order Parameters, and Complexity”, ex. 2.5)

Import packages

```
[ ]: %matplotlib inline
from matplotlib.pyplot import plot, figure, axes, hist
from numpy import *
```

One can efficiently generate and analyze random walks on the computer.

Write a routine RandomWalk(N,d) to generate an N-step random walk in d dimensions, with each step uniformly distributed in [-1/2,1/2] in each dimension. (Generate the steps first as an $N \times d$ array, and then do a cumulative sum.)

```
[ ]: def RandomWalk(N, d):
    """
    Use random.uniform(min, max, shape) to generate an array of steps of shape␣
 ↪(N,d),
    and then use cumsum(..., axis=0) (which adds them up along the 'N' axis).
    """
    steps = ...
    walks = ...
    return walks
```

Plot some one dimensional random walks versus step number, for N=10, 100, and 10000 steps. Does multiplying the number of steps by 100 roughly increase the distance by 10?

```
[ ]: for i in range(10):
    plot(RandomWalk(...,1));
figure()
for i in range(10):
    plot(RandomWalk(...));
figure()
for i in range(10):
    plot(RandomWalk(...));
```

Plot some two-dimensional random walks with N=10000 steps, setting axes(aspect='equal') beforehand to make the x and y scales the same. (Your routine gives x, y pairs, and you want x and y as arrays to plot, so you need to transpose.)

```
[ ]: axes(aspect='equal')
     for i in range(10):
         x, y = RandomWalk(...).transpose()
         plot(x,y);
```

Each random walk is different and unpredictable, but the ensemble of random walks has elegent, predictable properties.

Write a routine Endpoints(W, N, d) that just returns the endpoints of W random walks of N steps each in d dimensions. (No need to use cumsum; just sum. If you generate a 3D array of size (W, N, d), sum over axis=1 to sum over the N steps of each walk.

```
[ ]: def Endpoints(W, N, d):
         steps = ...
         return sum(..., axis=...)
```

Plot the endpoints of 10000 random walks of length 10. Then plot the endpoints of 10000 random walks of length 1. Discuss how this illustrates an emergent symmetry

```
[ ]: axes(aspect='equal')
     x, y = Endpoints(...).transpose()
     plot(x,y,'.')
     x, y = Endpoints(...).transpose()
     plot(...)
```

The most useful property of random walks is described by the central limit theorem. The endpoints of an ensemble of N-step random walks with RMS step-size a has a Gaussian or normal distribution as $N \to \infty$,

$$\rho(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-x^2/(2\sigma^2)),$$

with $\sigma = \sqrt{N}a$.

Calculate the RMS step-size a for one-dimensional steps uniformly distributed in (-1/2, 1/2). Compare the normalized histogram of 10000 endpoints with a normalized Gaussian of width σ predicted above, for $N = 1$, 2, and 5. How quickly does the Gaussian distribution become a good approximation for random walks?

```
[ ]: N = 1
     hist(Endpoints(...), bins = 50, density=True);
     sigma = sqrt(...)
     x = arange(-3.*sigma, 3.*sigma, 0.1*sigma)
     gauss = (1./...)*exp(...)
     plot(x,gauss,'r');
```

[ ]:
```
N = 2
...
```

[ ]:
```
N = 5
...
```