22 April 2019

Random Walk Algorithms: Lecture 10

David A. Meyer

In which the random walk evolution of a probability distribution inspires the solution to the heat equation with specified initial condition.

Initial conditions

The heat equation describes the time evolution of a function of position, so a well-defined problem is to compute the evolution of some *initial condition*, $u(0,s) = f(s)$. To figure out how to do this using the heat kernel we found in Lecture 8, we draw inspiration again from the $B = (X + 2I + X^{-1})/4$ random walk on $\mathbb{Z}$.

Suppose the initial probability distribution vector is not $\hat{e}_0$, but is rather

$$\vec{u}_0 = \sum_r u_{0,r}\hat{e}_r.$$

Then

$$u_{0,s} = \langle \vec{u}_0, \hat{e}_s \rangle = \langle \sum_r u_{0,r}\hat{e}_r, \hat{e}_s \rangle = \sum_r u_{0,r}\delta_{s,r}, \tag{1}$$

where $\langle \cdot, \cdot \rangle$ denotes the usual inner (dot) product and

$$\delta_{s,r} = \begin{cases} 1 & \text{if } s - r = 0; \\ 0 & \text{otherwise} \end{cases}$$

is the Kronecker delta. After one timestep,

$$u_{1,s} = \langle \vec{u}_1, \hat{e}_s \rangle = \langle B \sum_r u_{0,r}\hat{e}_r, \hat{e}_s \rangle = \sum_r u_{0,r} \langle B\hat{e}_r, \hat{e}_s \rangle = \sum_r u_{0,r} \frac{1}{4}(\delta_{s,r+1} + 2\delta_{s,r} + \delta_{s,r-1}),$$

so, since B is symmetric, we can write

$$B\delta_{s,r} = \frac{1}{4}(\delta_{s,r+1} + 2\delta_{s,r} + \delta_{s,r-1}).$$

Similarly, after t timesteps,

$$u_{t,s} = \langle \vec{u}_t, \hat{e}_s \rangle = \langle B^t \sum_r u_{0,r}\hat{e}_r, \hat{e}_s \rangle = \sum_r u_{0,r} \langle B^t\hat{e}_r, \hat{e}_s \rangle = \sum_r u_{0,r} \frac{1}{2^{2t}} \sum_{k=0}^{2t} \binom{2t}{k} \delta_{s,r+t-k}, \tag{2}$$

and we write

$$B^t\delta_{s,r} = \sum_{k=0}^{2t} \binom{2t}{k} \delta_{s,r+t-k}.$$

These equations have continuous analogues. Suppose $f \in C_0^\infty(\mathbb{R})$. Then the analogue of (1) is:

$$u(0,s) = f(s) = \langle f, \delta_s \rangle = \int f(r)\, \delta(s-r) \mathrm{d}r, \tag{3}$$

and the analogue of (2) is:

$$u(t,s) = \int f(r)\, \frac{1}{\sqrt{4\pi\alpha t}} e^{-(s-r)^2/(4\alpha t)} \mathrm{d}r. \tag{4}$$

Combining functions in this way is often useful, and has a name:

DEFINITION. Let f and g be integrable functions on $\mathbb{R}$. Then the *convolution* of f with g is

$$(f * g)(s) = \int f(r)\, g(s-r) \mathrm{d}r.$$

THEOREM. *The convolution of f with the Gaussian kernel given in* (4) *solves the initial value problem $u(0,x) = f(x)$ with $u(t,s)$ satisfying the heat equation.*

Proof. At $t = 0$, (4) becomes (3), so it satisfies the initial condition. Fot $t > 0$, the integrand in (4) is a continuous and differentiable function of t so

$$\frac{\partial}{\partial t} \int f(r)\, \frac{1}{\sqrt{4\pi\alpha t}} e^{-(s-r)^2/(4\alpha t)} \mathrm{d}r = \int f(r)\, \frac{\partial}{\partial t} \Big(\frac{1}{\sqrt{4\pi\alpha t}} e^{-(s-r)^2/(4\alpha t)} \Big) \mathrm{d}r.$$

Similarly,

$$\alpha \frac{\partial^2}{\partial s^2} \int f(r)\, \frac{1}{\sqrt{4\pi\alpha t}} e^{-(s-r)^2/(4\alpha t)} \mathrm{d}r = \int f(r)\, \alpha \frac{\partial^2}{\partial s^2} \Big(\frac{1}{\sqrt{4\pi\alpha t}} e^{-(s-r)^2/(4\alpha t)} \Big) \mathrm{d}r.$$

But we already learned that the heat kernel satisfies the heat equation, so these two expressions are equal. ∎

Simulation

In fact, the approximation to the binomial distribution by the heat kernel we learned in Lecture 9 also justifies the following simulation algorithm for the heat equation, in which $N \in \mathbb{N}$ is chosen large enough to achieve a desired accuracy:

input: $a < b \in \mathbb{R}$; $f : \mathbb{R} \to \mathbb{R}$, $f(x) = 0$ for $x \notin [a,b]$, $\int_a^b f(x)\, \mathrm{d}x < \infty$; $t_\mathrm{f} > 0$.
output: $u(t_\mathrm{f}, x)$ for $u(t,x)$ solving the heat equation with $u(0,x) = f(x)$.

$\Delta x \leftarrow (b-a)/N$
for $s = 0$ to $N-1$ do

$$u_{0,s} \leftarrow \int_{a+(s-1)\Delta x}^{a+s\Delta x} f(x)\,\mathrm{d}x$$
$\Delta t \leftarrow (\Delta x)^2/(4\alpha)$
$n \leftarrow \lfloor t_\mathrm{f}/\Delta t \rceil$
$\vec{u}_n \leftarrow B^n \vec{u}_0$
return $\vec{u}_n$

where the output is interpreted as $u(t_\mathrm{f}, x) = u_{n,\lfloor (x-a)/\Delta x \rceil}$. This algorithm is not explicit about how to calculate $B^n\vec{u}_0$; we will learn more about that in the following lectures.