random stuff that caught my fancy that I would otherwise forget

Saturday, 1 February 2014

quantum walking

Labels: algorithm, probability, python, science

It's possible to build a quantum random walk simulator in Python/NumPy with code that is very close to the mathematical definitions. Here's how.

First, we need to import NumPy (to do the array operations) and matplotlib (to visualise the results).

```
from numpy import *
from matplotlib.pyplot import *
```

We define the number of steps, N, that we are going to walk. We also define the total number of different positions the walker can be in after N steps.

```
N = 100      # number of random steps
P = 2*N+1    # number of positions
```

a quantum coin

We toss a quantum coin to decide whether to go left or right (or a superposition). In ket notation, we can write a general quantum coin as an arbitrary superposition of two states:

$$|coin\rangle = a|0\rangle_c + b|1\rangle_c;\ \text{where}|a|^2 + |b|^2 = 1$$

The ket notation is a convenient shorthand for the actual vectors representing the state:

$$|0\rangle_c = \begin{pmatrix} 1 \\ 0 \end{pmatrix};\ |1\rangle_c = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

So we can use a NumPy array to define a coin state:

```
coin0 = array([1, 0])  # |0>
coin1 = array([0, 1])  # |1>
```

the Hademard coin operator

We will see terms like $|0\rangle_c\langle 0|$ in what follows. This is the *outer product* of the relevant vectors, resulting in a matrix. We can use NumPy to calculate these:

```
C00 = outer(coin0, coin0)  # |0><0|
C01 = outer(coin0, coin1)  # |0><1|
C10 = outer(coin1, coin0)  # |1><0|
C11 = outer(coin1, coin1)  # |1><1|
```

Quantum operators are unitary matrices. The coin operator, that can be used to flip a quantum coin into a superposition, is:

$$\hat{C} = \frac{1}{\sqrt{2}}(|0\rangle_c\langle 0| + |0\rangle_c\langle 1| + |1\rangle_c\langle 0| - |1\rangle_c\langle 1|)$$

```
C_hat = (C00 + C01 + C10 - C11)/sqrt(2.)
```

position on a line

In ket notation, we can write the fully general position of the walker on the line as an arbitrary superposition of the P possible states:

$$|posn\rangle = \sum_k \alpha_k |k\rangle_p;\ \text{where} \sum_k |\alpha_k|^2 = 1$$

about me

Susan Stepney

View my complete profile

an infrequent blogger

This is an eclectic collection of stuff that I might want to remember, but doesn't fit well into **my more formal website**. That is where I collect my sf and technical reviews, publications, and other less ephemeral things. Also see **my Google+ page** for bits and bobs.

search this blog, and more

Search

labels

3D printer (4) algorithm (79) Amsterdam (4) architecture (13) astronomy (86) Auckland (4) Australia (1) Belgium (3) birds (37) Birmingham (6) Bonnie Tyler (3) books (265) Bradford (1) Brazil (1) Brighton (1) Bristol (5) Brussels (3) butterfly (8) cake (6) Cambridge (3) Canada (3) Cancun (7) Castle Howard (2) cat (13) chair (3) Chatsworth House (1) chemistry (7) Chicheley Hall (2) China (3) Christmas (19) cognition (40) coin (2) comet (4) complexity (84) computer (185) conference (57) dance (10) design (10) dinosaur (1) Doctor Who (40) dog (9) driving (24) Dublin (5) ducks (20) Dundee (3) duodecimal (1) Durham (4) economics (6) Edinburgh (4) education (41) electricity (18) Ely (2) engineering (4) equality (1) estimation (12) Evernote (12) evolution (26) Feynman (1) film (22) Finland (12) flight (13) font (10) Fontainebleau (2) food (43) fractals (19) France (23) frog (5) fruit (3) game (28) garden (83) Gatwick (4) geometry (4) Germany (2) Glasgow (3) glass (2) gotcha (3) GPS (1) Granada (1) graphics (76) Greece (4) grimoire (4) Groningen (1) growth (6) Hanover (1) Haskell (2) Heathrow (5) Helsinki (9) history (112) holiday (41) humour (55) illusion (10) Italy (5) Jodrell Bank (1) Jupiter (7) language (83) LaTeX (13) Leeds (4) lego (8) Leiden (1) lol (90) London (18) London Ontario (2) Los Angeles (2) Lyon (12) Malaga (1) Manchester (10) maps (23) Mars (7) mathematics (78) medicine (18) Mercury (4) Mexico (6) Milan (2) money (8) moon (31) music (40) Narnia (3) nature (16) Neptune (2) Netherlands (7) New York (1) New Zealand (17) Newcastle upon Tyne (1) Norway (3) Nottingham (1) obituary (9) Orleans (1) Paris (6) Peak District (1) pedantry (29) philosophy (35) Phoenix Arizona (1) Pluto (6) poetry (3) politics (138)

We assume the line is actually on a circle, so the positions at the ends wrap around. However, we will always make the circle big enough so that this doesn't happen during a walk. The tensor product $\otimes$ is implemented with the NumPy kron operation:

```
ShiftPlus = roll(eye(P), 1, axis=0)
ShiftMinus = roll(eye(P), -1, axis=0)
S_hat = kron(ShiftPlus, C00) + kron(ShiftMinus, C11)
```

walk operator

The walk operator combines the coin operator on the coin state, and a step operator on the combined coin and position state:

$$\hat{U} = \hat{S}\left(\hat{C} \otimes \hat{\mathbf{I}}_p\right)$$

```
U = S_hat.dot(kron(eye(P), C_hat))
```

initial state

Let's take the initial state of the system to be a coin in a superposition of left and right, and the walker at position 0:

$$|\psi\rangle_0 = |coin\rangle_0 \otimes |posn\rangle_0 = \frac{1}{\sqrt{2}}(|0\rangle_c + i|1\rangle_c) \otimes |0\rangle_p$$

```
posn0 = zeros(P)
posn0[N] = 1     # array indexing starts from 0, so index N is the central p
psi0 = kron(posn0,(coin0+coin1*1j)/sqrt(2.))
```

state after N steps

Then walking N steps is just applying the walk operator N times:

$$|\psi\rangle_N = \hat{U}^N |\psi\rangle_0$$

```
psiN = linalg.matrix_power(U, N).dot(psi0)
```

And we're done! $|\psi\rangle_N$ is the state of the system after N random quantum steps.

measurement operator

We can measure the state at position k using the measurement operator

$$\hat{M}_k = \hat{\mathbf{I}}_c \otimes |k\rangle_p\langle k|$$

We can use this to build up an array of probabilities, by taking the modulus squared of the state value at each position. (We can calculate the whole distribution in one go in simulation, but we would only get one measurement per experiment on the real quantum system.)

```
prob = empty(P)
for k in range(P):
    posn = zeros(P)
    posn[k] = 1
    M_hat_k = kron( outer(posn,posn), eye(2))
    proj = M_hat_k.dot(psiN)
    prob[k] = proj.dot(proj.conjugate()).real
```

plot the distribution

We can then plot the probabilities.

```
fig = figure()
ax = fig.add_subplot(111)

plot(arange(P), prob)
plot(arange(P), prob, 'o')
loc = range (0, P, P / 10) #Location of ticks
```

Russia (1) San Francisco (1) Saturn (3) science (150) science fiction (246) Scotland (3) security (11) Sheffield (2) shoes (4) Sicily (3) simulation (10) solar power (35) space flight (53) Spain (3) spider (4) stately home (7) statistics (66) storage (1) Sweden (1) Swindon (4) Switzerland (1) telephone (5) Thessalonika (1) Tokyo (4) Toronto (1) toy (4) TPS (11) trains (19) tree (35) Trello (4) Trondheim (3) Turku (1) TV (38) university (5) Uranus (1) USA (7) Utrecht (1) Venus (7) video (56) Volgograd (1) weather (124) web (65) York (2) zombies (1)

this month's popular posts

(no title)

John Baez lays out his approach to solving problems – interesting if you are spending your life at the task (ie, are an academic!) ...

You know you are reading a Peter Hamilton novel when ...

You know you are reading a Peter Hamilton novel when you think "oh, I'm almost finished now!", and there's still more than 200 pages to go...

quantum walking

It's possible to build a quantum random walk simulator in Python/NumPy with code that is very close to the mathematical definitions. H...

Angel's Feather

So, my upgraded phone not only no longer has the wonderful "smart alarm", that gradually increases volume, waking me up gently rather than ...

PythonTeX

I write technical documents in LaTeX, and my programming language of choice is (currently) Python. I'm about to start writing a large LaT...

Château de Fontainebleau

This afternoon was the conference trip, to Fontainebleau palace . This is right in Fontainebleau itself, so no long coach trip required. I...

≅

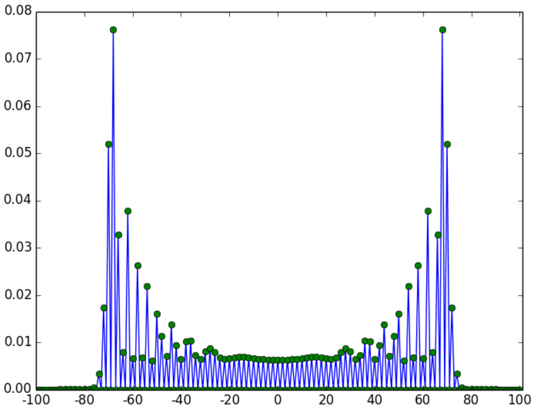
hunt the LaTeX symbol no more

So there you are, writing some maths in LaTeX, and you need that symbol that's a tilde on top of an equals sign: but you can't re...

blog archive

- ► 2019 (48)
- ► 2018 (106)
- ► 2017 (179)

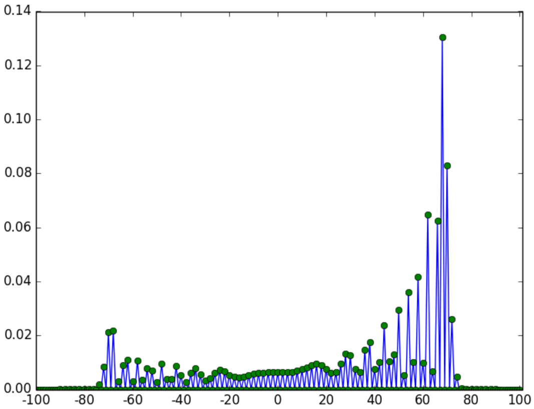
For $N = 100$ we get



probability distribution for a quantum random walk with $N = 100$, symmetric initial coin

The maximum probability occurs at $\approx N/\sqrt{2}$.

If instead of starting with a symmetric initial coin, we start with $|0\rangle_c$, we get



probability distribution for a quantum random walk with $N = 100$, initial coin $|0\rangle$

In neither case do these look *anything* like the bell curve of a classical random walk, with its maximum probability at 0. If you are drunk and trying to stagger home, best be quantum!

What I find most impressive about the Python is how closely we can make the code follow the mathematical formalism thoughout.

at 16:43

8 comments:

Viv Sunday, 2 February 2014 at 12:39:00 GMT

Thanks for this...saves me writing it myself for my Computing 2 students, I'll just direct therm here instead! -- Viv

Reply

Unknown Sunday, 29 January 2017 at 13:42:00 GMT

Thanks a lot for this.

Reply

Unknown Sunday, 17 September 2017 at 08:07:00 BST

Could someone help me understand the ShiftPlus and ShiftMinus operators written in terms of roll function?

► November (18)
► October (23)
► September (21)
► August (24)
► July (15)
► June (26)
► May (12)
► April (8)
► March (23)
▼ February (17)
colours of my tea
computation limits physics
New Testament = Cinderella
sequestering carbon, several books at a time XIX
John Baez lays out his approach to solving problem...
a surfeit of packaging
the chaos game
The Burden of the Bookish
so doge. much funny. wow
windy!
book review: Inside Jokes
sequestering carbon, several books at a time XVIII...
arms, legs, and...?
finding Brazil
writhing spots
solar stats for January
quantum walking
► January (37)
► 2013 (206)
► 2012 (103)
► 2011 (79)

pageviews

follow on Feedly

Follow on feedly

follow by email

Email address... Submit

Both Python as well as the quantum walk problem are new to me. Could someone help me to understand the ShiftPlus and ShiftMinus operators written in terms of some mysterious roll function?

Reply

Replies

Susan Stepney Sunday, 17 September 2017 at 10:25:00 BST

"roll" is a numpy function, with definition and examples defined here: https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.roll.html

Reply

Anonymous Monday, 10 September 2018 at 00:45:00 BST

Hi Susan,
Thank you very much for sharing this great work.
I have question please, If I want to use the same shift but add to it the mod

Reply

Anonymous Tuesday, 23 October 2018 at 17:31:00 BST

quick fyi, the code returns a float P unless we use int, as bellow :-)
loc = range (0, P, P // 10) #Location of ticks
ax.set_xticklabels(range (-N, N+1, P // 10))

Reply

Replies

Susan Stepney Tuesday, 23 October 2018 at 17:33:00 BST

Ah yes -- the code is old enough to be in python2!

Reply

Enter your comment...

Comment as: jaimepereirasar ▾ Sign out

Publish Preview □ Notify me

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

Bad Science

BoingBoing

Charlie's Diary

I Can Haz Cheezburger

Pharyngula

Saturday Morning Breakfast Cereal

Skepchick

Three-Toed Sloth

Tree Lobsters!

Whatever

xkcd

Powered by Blogger.