Lecture 23

Monte Carlo

David Semeraro

University of Illinois at Urbana-Champaign

November 19, 2013

Markov Application

We consider a light weight version of computing a realistic BCS ranking.One difficult aspect of the BCS rankings for college football is that not every team plays each other.

- Consider a simpler version of ranking Big Ten teams after the first four weeks of play.
- Not every team has played each other (or even played another Big Ten Team).

Markov Application

Consider the following games:

Michigan	16	Purdue	13
Iowa	38	Wisconsin	17
Iowa	28	Illinois	23
Minnesota	34	Michigan	21
Minnesota	23	Purdue	10
Purdue	31	Michigan	6
Wisconsin	33	Illinois	25
Wisconsin	38	Purdue	23
Illinois	27	Iowa	6
Illinois	20	Wisconsin	12

Markov Application

The adjacency matrix for this problem is:

$$A_{i,j} = \begin{cases} w_{i,j} & \text{team } i \text{ beats team } j \\ 0 & \text{otherwise} \end{cases}$$

where $w_{i,j}$ is the absolute value of the difference between scores. Order the teams 1-Michigan, 2-Iowa, 3-Minnesota, 4-Purdue, 5-Wisconsin, 6-Illinois. Now, $w_{1,3}$ represents a victory by Michigan over Minnesota by the amount assigned to $w_{1,3}$.

Markov Application

As with the Google matrix we need the column sums to be one (to guarantee values of the eigenvector to be in $[0, 1]$), so let

$$H_{i,j} = \frac{1}{\sum_{k=1}^{n} A_{k,j}} A_{i,j}$$

where we ignore any zero columns.

Markov Application

With this we have

$$A = \begin{bmatrix} 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 21 & 5 \\ 13 & 0 & 0 & 13 & 0 & 0 \\ 25 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 15 & 0 & 8 \\ 0 & 21 & 0 & 0 & 8 & 0 \end{bmatrix}$$

and

$$H = \begin{bmatrix} 0 & 0 & 0 & 3/31 & 0 & 0 \\ 0 & 0 & 0 & 0 & 21/29 & 5/13 \\ 13/38 & 0 & 0 & 13/31 & 0 & 0 \\ 25/38 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 15/31 & 0 & 8/13 \\ 0 & 1 & 0 & 0 & 8/29 & 0 \end{bmatrix}.$$

Markov Application

Teams that are undefeated have zero columns in H. We transform H into a *stochastic matrix* (columns add to 1) by performing a rank-1 update:

$$H \leftarrow H + ua^T.$$

Letting a be 1 for undefeated teams and 0 otherwise and u be $1/6$ we have

$$a = [0\,0\,1\,0\,0\,0]^T \qquad u = (1/6)[1\,1\,1\,1\,1\,1]^T$$

Thus

$$H + ua^T = \begin{bmatrix} 0 & 0 & 1/6 & 3/31 & 0 & 0 \\ 0 & 0 & 1/6 & 0 & 21/29 & 5/13 \\ 13/38 & 0 & 1/6 & 13/31 & 0 & 0 \\ 25/38 & 0 & 1/6 & 0 & 0 & 0 \\ 0 & 0 & 1/6 & 15/31 & 0 & 8/13 \\ 0 & 1 & 1/6 & 0 & 8/29 & 0 \end{bmatrix}.$$

Now entry i of column j for $H + ua^T$ is the probability that team j will lose to team i.

Markov Application

As with PageRank, we have a probability parameter $\alpha = 0.85$. In the BCS rankings, this would correspond to the likelihood that a voter would change their vote based on a loss to a higher ranked team. The final Google-like matrix is then

$$G = \alpha(H + ua^T) + (1 - \alpha)(1/6)ee^T$$

or

$$G = 0.85 \begin{bmatrix} 0 & 0 & 1/6 & 3/31 & 0 & 0 \\ 0 & 0 & 1/6 & 0 & 21/29 & 5/13 \\ 13/38 & 0 & 1/6 & 13/31 & 0 & 0 \\ 25/38 & 0 & 1/6 & 0 & 0 & 0 \\ 0 & 0 & 1/6 & 15/31 & 0 & 8/13 \\ 0 & 1 & 1/6 & 0 & 8/29 & 0 \end{bmatrix}$$

$$+0.15(1/6) \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Markov Application

Applying the power method to the matrix G gives the team rankings. The number one team corresponds to the largest element of the eigenvector. After 20 iterations beginning with a random vector the normalized eigenvector is:

$$\begin{bmatrix} 0.0780 \\ 0.5663 \\ 0.1315 \\ 0.1124 \\ 0.4590 \\ 0.6577 \end{bmatrix} \rightarrow \begin{bmatrix} Illinois \\ Iowa \\ Wisconsin \\ Minnesota \\ Purdue \\ Michigan \end{bmatrix}$$

This shows that the team listed in position six has the largest component and it therefore first in the ranking.

Why

- Central to the PageRank (and many many other applications in finance, science, informatics, etc) is that we randomly process something
- what we want to know is “on average” what is likely to happen
- what would happen if we have an infinite number of samples?
- let’s take a look at integral (a discrete limit in a sense)

Integration

- integral of a function over a domain

$$\int_{x \in D} f(x)\, dA_x$$

- the size of a domain

$$A_D = \int_{x \in D} dA_x$$

- average of a function over some domain

$$\frac{\int_{x \in D} f(x) dA_x}{A_D}$$

integral example

The average “daily” snowfall in Champaign last year

- domain: year (1d time interval)
- integration variable: day
- function: snowfall depending on day

$$average = \frac{\int_{day \in year} s(day) d_{day}}{lengthofyear}$$

integral example

The average snowfall in Illinois

- domain: Illinois (2d surface)
- integration variable: (x, y) location
- function: snowfall depending on location

$$average = \frac{\int_{location \in Illinois} s(location) d_{location}}{areaofIllinois}$$

integral example

The average snowfall in Illinois today

- domain: Illinois $\times$ year (3d space-time)
- integration variable: location and day
- function: snowfall depending on location and day

$$average = \frac{\int_{day \in year} \int_{location \in Illinois} s(location, day) d_{location,day}}{areaofIllinois \cdot lengthofyear}$$

discrete random variables

- random variable x
- values: $x_0, x_1, \ldots, x_n$
- probabilities $p_0, p_1, \ldots, p_n$ with $\sum_{i=0}^{n} p_i = 1$

throwing a die (1-based index)

- values: $x_1 = 1, x_2 = 2, \ldots, x_6 = 6$
- probabilities $p_i = 1/6$

expected value and variance

- expected value: average value of the variable

$$E[x] = \sum_{j=1}^{n} x_j p_j$$

- variance: variation from the average

$$\sigma^2[x] = E[(x - E[x])^2] = E[x^2] - E[x]^2$$

throwing a die

- expected value: $E[x] = (1 + 2 + \cdots + 6)/6 = 3.5$
- variance: $\sigma^2[x] = 2.916$

estimated $E[x]$

- to estimate the expected value, choose a set of random values based on the probability and average the results

$$E[x] = \frac{1}{N} \sum_{j=1}^{N} x_i$$

- bigger N gives better estimates

throwing a die

- 3 rolls: $3, 1, 6 \rightarrow E[x] \approx (3 + 1 + 6)/3 = 3.33$
- 9 rolls:
  $3, 1, 6, 2, 5, 3, 4, 6, 2 \rightarrow E[x] \approx (3 + 1 + 6 + 2 + 5 + 3 + 4 + 6 + 2)/9 = 3.51$

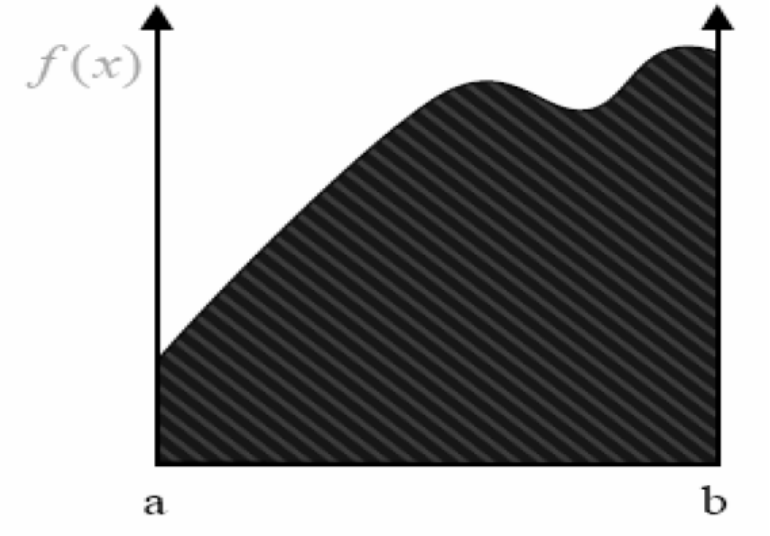
law of large numbers

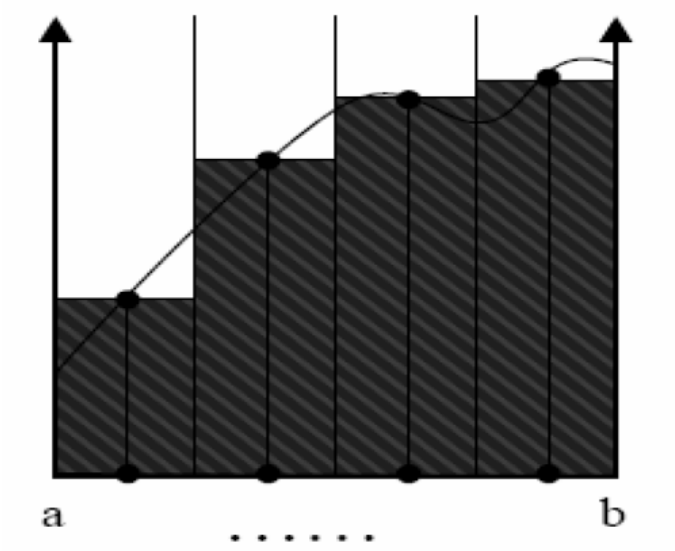
- by taking N to ∞, the error between the estimate an the expected value is statistically zero. That is, the estimate will converge to the correct value

$$P\left(E[x] = lim_{N\to\infty}\frac{1}{N}\sum_{i=1}^{N} x_i\right) = 1$$

(deterministic) numerical integration

- split domain into set of fixed segments
- sum function values with size of segments (Riemann!)





Algorithm

Consider the MVT for integrals:

$$\int_a^b f(x)\,dx = f(c)(b-a)$$

Where $f(c)$ is the average value of f in $[a,b]$.
We have for a random sequence $x_1, \ldots, x_n$

$$\int_0^1 f(x)\,dx \approx \frac{1}{n}\sum_{i=1}^{n} f(x_i)$$

```
1  n=100;
2  x=rand(n,1);
3  a=f(x);
4  s=sum(a)/n;
```

1D Integration

Given the previous expression for the average of a function on an interval we can evaluate through the MVT the value of an integral using random values. For example, the following python program evaluates $\int_0^1 x^2\,dx$.

```
1  import numpy as np
2 n = 10000
3 x = np.random.rand(n)
4 fx = sum(x*x)
5 print fx/n
```

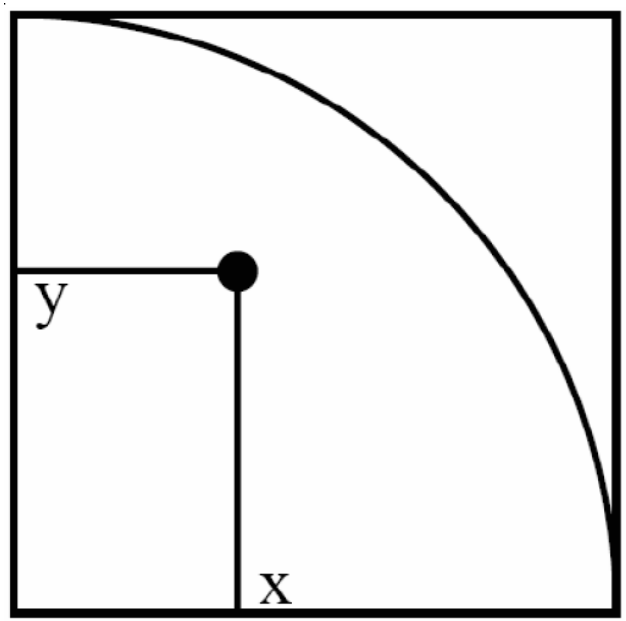
The value arrived at for n = 10000 is 0.33546. Not terribly accurate but not bad.

2d: example computing π

Use the unit square $[0,1]^2$ with a quarter-circle

$$f(x,y) = \begin{cases} 1 & (x,y) \in \ circle \\ 0 & else \end{cases}$$

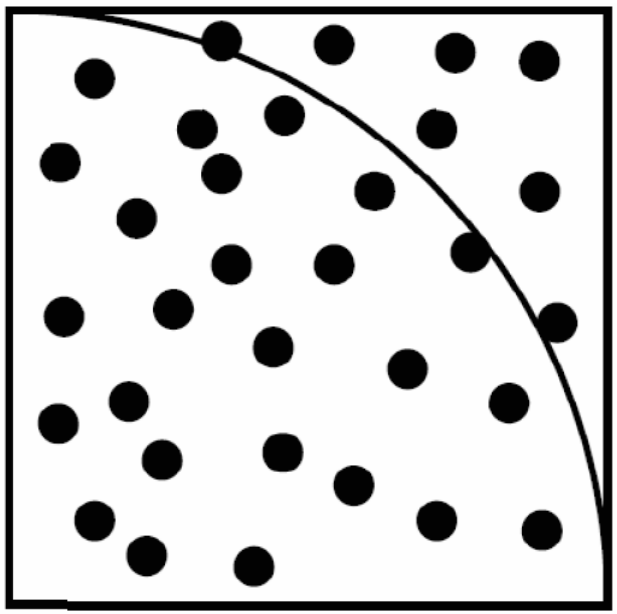
$$A_{quarter-circle} = \int_0^1 \int_0^1 f(x,y)\, dxdy = \frac{pi}{4}$$



2d: example computing π

Estimate the area of the circle by randomly evaluating $f(x, y)$

$$A_{quarter-circle} \approx \frac{1}{N} \sum_{i=1}^{N} f(x_i, y_i)$$



2d: example computing π

By definition

$$A_{quarter-circle} = \pi/4$$

so

$$\pi \approx \frac{4}{N} \sum_{i=1}^{N} f(x_i, y_i)$$

2d: example computing π, algorithm

```
1  input N
2  call rand in 2d
3  for i=1:N
4     sum = sum + f(x_i, y_i)
5  end
6  sum = 4 * sum/N
```

2d: example computing π, algorithm

The expected value of the *error* is $\mathcal{O}\left(\frac{1}{\sqrt{N}}\right)$

- convergence does not depend on dimension
- deterministic integration is very hard in higher dimensions
- deterministic integration is very hard for complicated domains

MVT in higher dimensions

Notice that the average of f over a region is obtained by integrating and dividing by the area, volume, or measure of that region. For example:

$$\frac{1}{8}\int_1^3\int_{-1}^1\int_0^2 f(x,y,z)\,dxdydz$$

is the average of f over the parallelepiped descrived by the following inequalities: $0 \leqslant x \leqslant 2$, $-1 \leqslant y \leqslant 1$, and $1 \leqslant z \leqslant 3$.

Surface integral (example from book)

Solve:

$$\int\int_{\Omega} sin\sqrt{ln(x+y+1)}\,dxdy = \int\int_{\Omega} f(x,y)\,dxdy$$

over the disk in xy-space defined by the inequality:

$$\Omega = \left\{(x,y) : \left(x-\frac{1}{2}\right)^2 + \left(y-\frac{1}{2}\right)^2 \leqslant \frac{1}{4}\right\}$$

Surface Integral

- Generate 5000 random points $p_i = (x_i, y_i)$
- Discard points outside the disk

Estimate the integral by:

$$\int\int_{\Omega} f(x,y)\,dxdy = (\textit{area of disk}\ \Omega) * (\textit{average height of}\ f\ \textit{over}\ n\ \textit{random points})$$

$$= (\pi r^2)\left[\frac{1}{n}\sum_{i=1}^{n} f(p_i)\right]$$

$$= \frac{\pi}{4n}\sum_{i=1}^{n} f(p_i)$$

Surface Integral

```
1  import numpy as np
2  import math
3  def f(x,y):
4      temp = math.sqrt(math.log((x + y + 1.0)))
5      return math.sin(temp)
6  n = 5001
7  iprt = 10
8  j    = 0
9  sum = 0.0
10 x = np.random.rand(n)
11 y = np.random.rand(n)
12 for i in range(n):
13     if (( (x[i] - 0.5)**2 + (y[i] - 0.5)**2 ) <= 0.25 ):
14         j = j + 1
15         sum = sum + f(x[i],y[i])
16         if (j % iprt == 0):
17             vol = math.pi*sum*0.25/j
18             print j, vol
19 vol = math.pi*sum*0.25/j
20 print j,vol
```

Surface Integral

j	average	integral
1000	0.723674995939	0.568373012707
2000	0.722805639908	0.567690222077
3000	0.722887515293	0.567754526854

3953 0.567452566273

Stochastic Simulation

From M. Heath, *Scientific Computing, 2nd ed.*, CS450

- Stochastic simulation mimics physical behavior through random simulations
- ...also known as Monte Carlo method (no single MC method!)
- Usefulness:
  - nondeterministic processes
  - deterministic models that are too complicated to model
  - deterministic models with high dimensionality

`http://www.cse.uiuc.edu/iem/integration/mntcurve/`
`http://www.cse.uiuc.edu/iem/integration/mntcirc/`