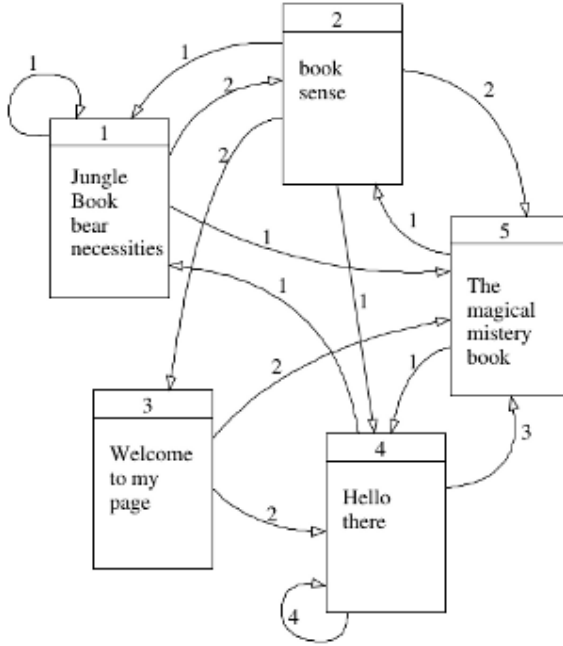


Google Nasıl Isler?

Google arama motoruna bir kelime yazdigimizda geri gelen sonuclar nasıl kararlaştirilir? İlk akla gelen yontem tabii ki Web'deki tum sayfaların (milyarlarca sayfa) sayfalar üzerindeki kelimelerin o sayfa ile iliskilendirilmesi ve arama yapilınca kelimeye göre sayfa geri getirilmesi. Mesela alttaki ornekte “book (kitap)” yazınca geriye 1., 2. ve 5. sayfalar geri gelecek. Fakat hangi sirada? Bu sayfalardan hangisi digerlerinden daha onemli?



Google'in arama motorlarına getirdigi en büyük yeniliklerden biri PageRank algoritmasıdır. Bu algoritmanın temelinde daha fazla referans edilen sayfalar daha üstte cikması yatar. Hatta o referans eden sayfaların kendilerine daha fazla referans var ise bu etki ta en sondaki sayfaya kadar yansitilir, hatta bu zincir bastan sona her seviyede hesaplanabilir. Peki bu nasıl gerçekleştirilir?

PageRank Web sayfalarını bir Markov Zinciri olarak görür. Markov Zincirleri seri halindeki $X_n, n = 0, 1, 2, \dots$ rasgele değişkenini modeller ve bu değişkenler belli sayıdaki konumların birinde olabilirler. Mesela konumları bir doğal sayı ile ilintilendirirsek $X_n = i$ olabilir ki $i = \{0, 1, \dots\}$ diye kabul edelim.

Markov Zincirlerinde (MZ) i konumundan j konumuna geçiş olasılığını, P_{ij} , biliriz ve bu $P(X_{n+1} = j | X_n = i)$ olarak acilabilir. Acilimdan görüleceği üzere bir MZ sonraki adıma geçiş olasılığı için sadece bir önceki adıma bakar. Bu tür önce/sonra yapısındaki iki boyutlu hal, çok rahat bir şekilde matrisine çevirilebilir / gösterilebilir. Önceki konum satırlar, sonraki konum kolonlar olarak betimlenir mesela.

Örnek

Bir sonraki günde yağmur yağmayacağını bir MZ olarak tasarlayalım. Bir sonraki günde yağmur yağmayacağını sadece bugün etkiliyor olsun. Eğer bugün yağmur

yagiyorsa yarın yağmur yağması 0.7, eğer bugün yağmıyor ise yarın yağması 0.4. MZ şöyle

$$P = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix}$$

Gecis olasiliklarindan bahsettigimize gore ve elimizde sinirli / belli sayida konum var ise, bir MZ'nin her satirindaki olasiliklari toplami tabii ki 1'e esit olmalidir.

MZ'lerin ilginç bir özelliği n adım sonra i, j gecisinin P^n hesabıyla yapılabilmesidir. Yani P 'yi n defa kendisiyle carpip i, j kordinatina bakarsak n adım sonrasını rahatca görebiliriz. Bunun ispatını burada vermeyeceğiz.

Mesela üstteki örnekte, eğer bugün yağmur yagiyorsa 4 gün sonra yağmur yağma olasılığı nedir?

```
import numpy.linalg as lin
P = np.array([[0.7,0.3],[0.4,0.6]])
P4 = lin.matrix_power(P,4)
print P4

[[ 0.5749  0.4251]
 [ 0.5668  0.4332]]
```

Aradığımız gecis için kordinat 0,0'a bakıyoruz ve sonuç 0.5749. Numpy `matrix_power` bir matrisi istediğimiz kadar kendisiyle carpmamızı sağlıyor.

Duragan Dagilim (Stationary Distribution)

Eğer yağmur örneğindeki matrisi carpmaya devam edersek, mesela 8 kere kendisiyle carpsak sonuç ne olurdu?

```
import numpy.linalg as lin
P = np.array([[0.7,0.3],[0.4,0.6]])
P8 = lin.matrix_power(P,8)
print P8

[[ 0.57145669  0.42854331]
 [ 0.57139108  0.42860892]]
```

Dikkat edilirse, her satır bir değer yaklaştırmaya başlıyor. Bu değer MZ'nin duragan dağılımıdır, belli koşullara uyan her MZ'nin böyle bir duragan dağılımı vardır. Bu koşullar MZ'nin periyodik olmayan (aperiodic) ve tekrar eden (recurrent) olmasıdır. Bu şartlar çok “özel” şartlar değildir aslında, daha çok “normal” bir MZ'yi tarif ediyor diyebiliriz. Tüm konumları tekrar eden yapmak kolaydır, MZ tek bağlı (singly connected) hale getirilir, yani her konumdan her diğer konuma bir gecis olur, ve periyodik olmaması için ise MZ'ye olmadığı zamanlarda bir konumdan kendisine gecis sağlanır (az bir gürültü üzerinden).

Nihayetinde duraganlık şu denkleme sebebiyet verir,

$$\pi = \pi P$$

Burada duragan dagilim π 'dir. Bu denklem tanidik geliyor mu? Devrigini alarak soyle gosterelim, belki daha iyi taninir,

$$P^T \pi^T = \pi^T$$

Bir sey daha ekleyelim,

$$P^T \pi^T = 1 \cdot \pi^T$$

Bu ozdeger/vektor formuna benzemiyor mu? Evet! Bu form

$$Ax = \lambda x$$

MZ denklemi sunu soyluyor, 1 degerindeki ozdegere ait ozvektor bir MZ'nin duragan dagilimidir! Bu arada, MZ gecis matrisi P 'nin en buyuk ozdegerinin her zaman 1 oldugunu biliyoruz. Bu durumda en buyuk ozdegere ait ozvektoru hesaplamak yeterli olacaktır. Bunu yapmayi zaten *Lineer Cebir Ders 21*'de ogrenmistik, ust metot (power method) sayesinde bu hesap kolayca yapilabiliyor.

Simdi en basktaki Web sayfalarina ait gecisleri yazalim,

```
P = [[1./4, 2./4, 0, 0, 1./4],
      [1./6, 0, 2./6, 1./6, 2./6],
      [0, 0, 0, 2./4, 2./4],
      [1./8, 0, 0, 4./8, 3./8],
      [0, 1./2, 0, 1./2, 0]]
```

```
P = np.array(P)
```

```
print P
```

```
[[ 0.25      0.5      0.      0.      0.25      ]
 [ 0.16666667 0.      0.33333333 0.16666667 0.33333333]
 [ 0.         0.         0.         0.5       0.5       ]
 [ 0.125      0.         0.         0.5       0.375     ]
 [ 0.         0.5       0.         0.5       0.        ]]
```

Simdi ust metodu kullanarak duragan dagilimi hesaplayalim. Herhangi bir baslangic vektorunu P ile 20 kere carpmak yeterli olur.

```
import numpy.linalg as lin
x=np.array([.5, .3, .1, .1, 0]) # herhangi bir vektor
for i in range(20):
    x = np.dot(x,P)
print 'pi = ', x
```

```
pi = [ 0.10526316  0.18421053  0.06140351  0.38596491  0.2631579 ]
```

Not: Aslında cebirsel olarak P 'yi 20 kere kendisiyle carpmak ve sonucu baslangic vektörü ile bir kere carpmak ta dusunulebilirdi. Fakat 20 kere vektor / matris carpimlari yapmak, 20 kere matris / matris carpimi yapmaktan daha verimli olacaktir. Buyuk Veri ortami icin de bu soylenebilir.

Eger kutuphane cagrisi kullanmak gerekirse,

```
import numpy.linalg as lin
evals, evec = lin.eig(P.T)
pi = evec[:,0] / np.sum(evec[:,0])
print np.abs(pi)
```

```
[ 0.10526316  0.18421053  0.06140351  0.38596491  0.26315789]
```

Sonuc gosteriyor ki “book” yazdigizda Google bize 5. sayfayi en basta olacak sekilde sonuc dondurmeli, cunku onun duragan dagilimi 1,2,5 sayfalarinin arasinda en yuksek.

Duragan Dagilima Bakis

MZ ve duragan dagilimin PageRank'le alakasini bir daha dusunelim. MZ ile n adim sonrasini hesaplayabiliyoruz, duragan dagilim ise sonsuz adim sonrasini ifade ediyor. Ve bu dagilim, bir anlamda, sonsuz yapilan adimlar sirasinda *en fazla hangi konumlarda* zaman gecirilecegini gosteriyor. Konum yerine sayfa dersek duragan dagilimin niye en onemli sayfalari belirlemek icin onemli oldugunu anlariz.

Kullanici herhangi bir sayfada iken hangi diger sayfalara gidecegi o sayfa uzerinde baglantilar uzerinden anlasilir, PageRank bu baglantinin mevcudiyetine bakar sadece, o mevcudiyet uzerinden bir gecis olasiligi hesaplar, ve bu olasiliga gore (raslantisal sekilde) baglantinin takip edilecegi dusunulur. Bu arada cogunlukla sayfalar arasindaki baglantilarin agirligi 1 olacaktir, fakat ornek amaclli 2,3 gibi sayilar da kullaniliyor.

[1] Murphy, K., CS340: Machine Learning Lecture Notes, www.ugrad.cs.ubc.ca/~cs340

[2] Ross, S., Introduction to Probability Models, 8th Edition