# 5.1 거듭제곱법

YouTube 동영상 강의: http://youtu.be/CLxjkZuNJXw

이 절의 목표는 행렬의 가장 큰 고유값을 수치적으로 구하는 방법을 소개하는 것이다. 사회 현상에서 유도되는 많은 행렬 모델에 대응하는 행렬의 가장 큰 고유값은 그 모델의 변화에 대한 예측을 가능하게 하는 많은 정보를 준다. 따라서 가장 큰 고유값만 찾아도 원하는 답을 얻기에 충분한 경우가 많다. 그러나 손으로는 물론 소프트웨어를 이용해도 행렬이 조금만 커지면 정확한 고유값들을 모두 구하는 것은 힘들어진다. 따라서 크기가 큰 행렬의 경우에 모든 고유값을 찾기보다는 가장 큰 고유값만을 손쉽게 찾는 새로운 방법을 연구해 낸 것이 바로 행렬의 거듭제곱을 이용하는 '거듭제곱법(Power Method)'이다.

만약 $A$가 $\lambda_1, \lambda_2, \cdots, \lambda_n$을 서로 다른 고유값으로 갖는 $n \times n$ 행렬이라고 가정하면, 일반성을 잃지 않고 고유값을 다음처럼 크기순으로 배열할 수 있다. 주로 감소하는 방향으로 순서를 정한다.

$$|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n|$$

이때 크기(절댓값)가 가장 큰 (실수의) 고유값인 $\lambda_1$을 **우세한**(dominant) 고유값이라고 한다. 또 각각의 고유값에 대응하는 고유벡터를 각각

$$\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_n$$

이라고 표시하자. 그러면 "고유값들이 서로 다르므로 그에 대응하는 고유벡터들의 집합은 일차독립인 집합이다." 고유값/고유벡터의 정의에 의하여,

$$A\mathbf{v}_i = \lambda_i \mathbf{v}_i, \quad i = 1, 2, \cdots, n$$

이라 하면, 우리의 문제는 주어진 행렬 $A$에 대해 크기가 가장 큰 고유값 $\lambda_1$과 그에 대응하는 고유벡터 $\mathbf{v}_1$을 찾는 것이다. 이 일을 하기 전에 먼저 다음과 같은 사실을 기억하자.

i) 고유벡터의 0이 아닌 스칼라배도 같은 고유값에 대응하는 고유벡터이다.
ii) 고유값과 고유벡터 관계를 확인하는 가장 정확한 방법은 $A\mathbf{x} = \lambda\mathbf{x}$가 되는 것을 확인하는 방법이다.
iii) 우리는 $\mathbf{e}_i$를 열벡터 $[0\,0\cdots 1\cdots 0\,0]^T$라고 표시하자. 즉 $\mathbf{e}_i$는 $i$번째 성분만 1이고, 나머지 성분은 모두 0인 열벡터이다. 그러므로 $A\mathbf{e}_i$는 $A$의 $i$번째 열이 될 것이다.

거듭제곱법은 간단하다. 일차독립인 벡터들의 집합 $\{\mathbf{v}_i\}_{i=1}^n$은 **정리 7.4**에 의하여 $R^n$의 기저를

이루기 때문에 임의의 벡터 $\mathbf{x}$는

$$\mathbf{x}= c_1\mathbf{v}_1 + c_2\mathbf{v}_2 +\cdots+ c_n\mathbf{v}_n$$

으로 표현할 수 있다. 양변에 $A$를 곱하고, 이를 전개하면,

$$A\mathbf{x}= c_1A\mathbf{v}_1 + c_2A\mathbf{v}_2 +\cdots+ c_nA\mathbf{v}_n$$

이 되며 또한 $A\mathbf{v}_i = \lambda_i\mathbf{v}_i,\ i = 1, \cdots, n$이므로

$$A\mathbf{x}= c_1\lambda_1\mathbf{v}_1 + c_2\lambda_2\mathbf{v}_2 +\cdots+ c_n\lambda_n\mathbf{v}_n$$

이 된다. 여기서 양변에 $A$를 $k$번 반복하여 곱하면,

$$\begin{aligned} A^k\mathbf{x} &= c_1\lambda_1^k\mathbf{v}_1 + c_2\lambda_2^k\mathbf{v}_2 +\cdots+ c_n\lambda_n^k\mathbf{v}_n \\ &= \lambda_1^k\left\{c_1\mathbf{v}_1 + c_2\left(\frac{\lambda_2}{\lambda_1}\right)^k\mathbf{v}_2 +\cdots+ c_n\left(\frac{\lambda_n}{\lambda_1}\right)^k\mathbf{v}_n\right\} \end{aligned}$$

을 얻을 수 있다. 충분히 큰 $k$값에 대해서 $\lambda_1$의 크기가 다른 $\lambda_i$들의 크기보다 크기 때문에 $|\lambda_1|^k$은 모든 $|\lambda_i|^k\ (i = 2, \cdots, n)$보다 매우 커질 것이다. 즉 $\left(\frac{\lambda_i}{\lambda_1}\right)^k \to 0$.

그러므로 우변의 첫 번째 항의 값은 나머지 항의 값을 압도(dominate)할 것이고, 따라서 충분히 큰 $k$에 대해서

$$A^k\mathbf{x} \approx c_1\lambda_1^k\mathbf{v}_1$$

을 얻을 수 있다.

게다가 고유벡터의 (0이 아닌) 스칼라배도 역시 고유벡터이므로 우리는 고유값 $\lambda_1$에 대응하는 고유벡터를 두 벡터 $A^{k+1}\mathbf{x}$와 $A^k\mathbf{x}$의 성분들을 비교하여 찾을 수 있다. 이때 $A^{k+1}\mathbf{x}$의 각 성분은 $A^k\mathbf{x}$의 각 성분의 $\lambda_1$배이다.

그러면 $\mathbf{x}$는 무엇일까? 그것은 임의로 선택될 수 있다. 그러나 다른 고유벡터 중에 하나이거나 그들의 1차 결합 중 하나이기는 원하지 않으므로 '거듭제곱법'에서는 보통 $\mathbf{x}=\mathbf{x}_0$를 안전하게 열벡터

$$[\,1\ 1\ \cdots\ 1\ \cdots\ 1\ 1\,]^T$$

로 선택한다. 그리고 적당한 $k$에 대하여

$$\begin{aligned} \mathbf{x}_1 &= A\mathbf{x}_0 \\ \mathbf{x}_2 &= A^2\mathbf{x}_0 = A\mathbf{x}_1 \\ &\vdots \\ \mathbf{x}_{k+1} &= A^{k+1}\mathbf{x}_0 \approx \lambda A^k\mathbf{x}_0 = \lambda\mathbf{x}_k \end{aligned}$$

되는 $\lambda$를 찾으면, 그것이 바로 우리가 찾는 크기가 가장 큰 '우세한(dominant) 고유값 $\lambda_1$'이 된다.

그리고 $\mathbf{x}_k$가 대응하는 고유벡터이다. 아래의 예를 통해 위의 설명을 확인하자.

**예제 1** 행렬 $A$가 다음과 같다고 하자.

$$A = \begin{bmatrix} 1 & 2 & 0 \\ 2 & 1 & 2 \\ 1 & 2 & 3 \end{bmatrix}$$

$\mathbf{x}=(1, 1, 1)$로 하여 Sage나 MATLAB, Maple 또는 MATHEMATICA를 이용하여 행렬의 충분히 큰 (약 10 또는 20 정도) 거듭제곱을 구하면 다음과 같다.

Sage http://www.sagenb.org/home/pub/244/ 또는 http://math3.skku.ac.kr/pub/2758/

$$A^{10}\mathbf{x} = \begin{bmatrix} 3498205 \\ 6681695 \\ 9264127 \end{bmatrix}, \quad A^{11}\mathbf{x} = \begin{bmatrix} 16861595 \\ 32206359 \\ 44653976 \end{bmatrix}$$

이 값들만으로 보면 두 벡터 사이의 연관성이 적어 보인다. 그러나 각각을 자신의 첫 번째 성분으로 나누면 다음을 얻게 된다.

$$A^{10}\mathbf{x} = 3498205 \begin{bmatrix} 1 \\ 1.910 \\ 2.648 \end{bmatrix}$$

$$A^{11}\mathbf{x} = 16861595 \begin{bmatrix} 1 \\ 1.910 \\ 2.648 \end{bmatrix}$$

이제 연관성이 쉽게 확인된다. 만약 이 단계에서 이런 연관성(수렴하는 고유벡터)이 없으면 극한에 가까이 갈 정도로 충분히 큰 $k$값을 이용하면 궁극적으로 위와 같은 연관성을 갖는다.

우리는 고유값/고유벡터의 정의를 이용하여 다음의 결과를 확인할 수 있다. 즉 우세한(dominant) 고유값은

$$\lambda_1 = \frac{16861595}{3498205} = 4.820$$

이며 그것에 대응하는 고유벡터는 $[1 \ 1.910 \ 2.648]^T$이다. 실제로

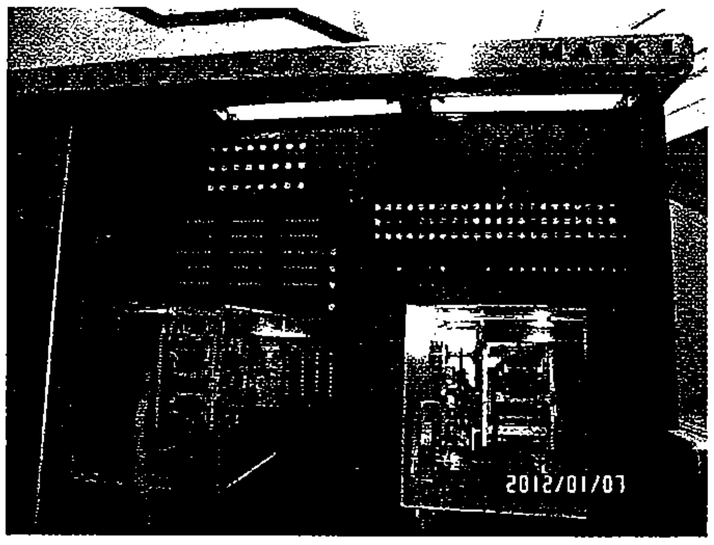
$$\begin{bmatrix} 1 & 2 & 0 \\ 2 & 1 & 2 \\ 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 1.910 \\ 2.648 \end{bmatrix} = 4.820 \begin{bmatrix} 1 \\ 1.910 \\ 2.648 \end{bmatrix}$$

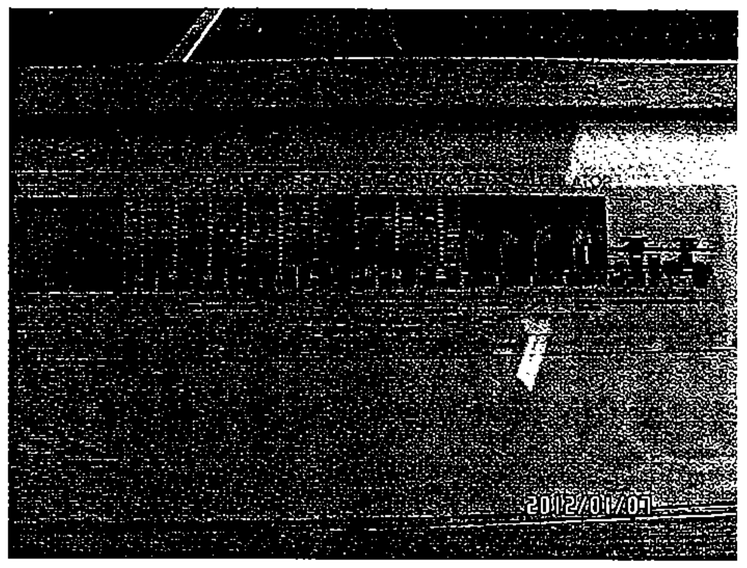
을 만족한다. ■

예제 1에서 보듯이 우리는 가장 큰 고유값과 그에 대응하는 고유벡터를 구하기 위해 행렬 $A$의 거듭제곱을 이용하였다. 그러면 도대체 얼마나 큰 거듭제곱이 필요할까? 위의 예에서는 임의로 $A^{10}\mathbf{x}$, $A^{11}\mathbf{x}$를 소수 3자리까지 계산할 수 있었다. 그러나 더욱 정확한 고유값를 원한다면 연속적

인 더 높은 거듭제곱의 행렬을 이용하면 된다. 아래 웹사이트에서 제공하는 '**페론-프로베니우스 정리**(Perron-Frobenius theorem)'는 "음이 아닌 어떤 행렬에 대해서도 실수값으로 유일하게 존재하는 크기가 가장 큰 고유값은 위의 방법으로 언제나 구할 수 있다"는 것을 이론적으로 보여준다.

[참고] http://matrix.skku.ac.kr/sglee/perron_frobenius/perron_frobenius.html




여기서 잠깐

애플의 최신 스마트폰인 아이폰4S가 최근 주목받게 된 건 새롭게 탑재된 '시리(Siri)' 때문이다. 음성인식 서비스인 시리는 똑똑한 개인비서의 역할을 척척 해내면서 호평을 받고 있다. 음성인식기술은 이미 잘 알려졌지만, 시리의 경우는 활용범위나 응답의 적절성 면에서 차원이 다른 서비스를 제공한다. 시리는 인공지능과 음성인식, 검색 기술을 두루 사용해 사용자의 음성명령을 수행한다. 특히 '주인'의 명령에 즉각 반응을 보일 수 있게 된 데는 첨단검색기술로 알려진 울프럼알파(Wolframalpha) 검색엔진의 힘이 절대적이다.

애플은 울프럼알파 엔진과 음성인식기술을 결합한 후 여기에 인공지능 학습기능을 추가했다. 이 기능은 사용자가 질문했던 내용이나 개인정보를 연계해 사용자의 행동유형을 파악하는 것이다. 시리는 사용자가 질문을 하면 문맥을 곧바로 이해한다. 그러다보니 질문에 대해 한층 더 자연스러운 답변을 내놓을 수 있다. 게다가 이용자들이 접근을 허용한 개인정보까지 능숙하게 활용한다. 울프럼알파는 수학 연산프로그램인 매스매티카(Mathematica) 개발자인 스티븐 울프럼이 만든 지식검색엔진이다.

## 5.1 연습문제

YouTube

**1.** 다음은 각각 어떤 행렬 $A$의 서로 다른 고유값들을 구한 것이다. 이 중에서 크기가 가장 큰(우세한, dominant) 고유값은 무엇인가?

(1) $\lambda_1 = 7,\ \lambda_2 = 3,\ \lambda_3 = -8,\ \lambda_4 = 1$

(2) $\lambda_1 = -5,\ \lambda_2 = 3,\ \lambda_3 = 2,\ \lambda_4 = 5$

**[2-3]** 다음 행렬 $A$를 Sage나 Matlab, MATHEMATICA, Maple을 이용하여 예제 1과 같은 방법으로 "우세한(dominant) 고유값"과 그에 대응하는 고유벡터를 구하여라.

http://matrix.skku.ac.kr/CLA12/Sage-5-1-2.html

**2.** $A = \begin{bmatrix} -7 & -12 \\ 8 & 13 \end{bmatrix},\ \mathbf{x}_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

**Ans** 우세한 고유값: 5

대응하는 고유벡터 $\begin{bmatrix} 1 \\ -1 \end{bmatrix}$ ■

**3.** $A = \begin{bmatrix} 2 & 3 \\ 4 & 6 \end{bmatrix},\ \mathbf{x}_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

**Ans** 우세한 고유값: 8

대응하는 고유벡터 $\begin{bmatrix} 1 \\ 2 \end{bmatrix}$ ■

**4.** 다음 행렬 $A$를 Sage나 Matlab, MATHEMATICA, Maple을 이용하여 거듭제곱이 커질수록 더 정확하게 우세한(dominant) 고유값과 그에 대응하는 고유벡터를 구할 수 있음을 확인하여라.

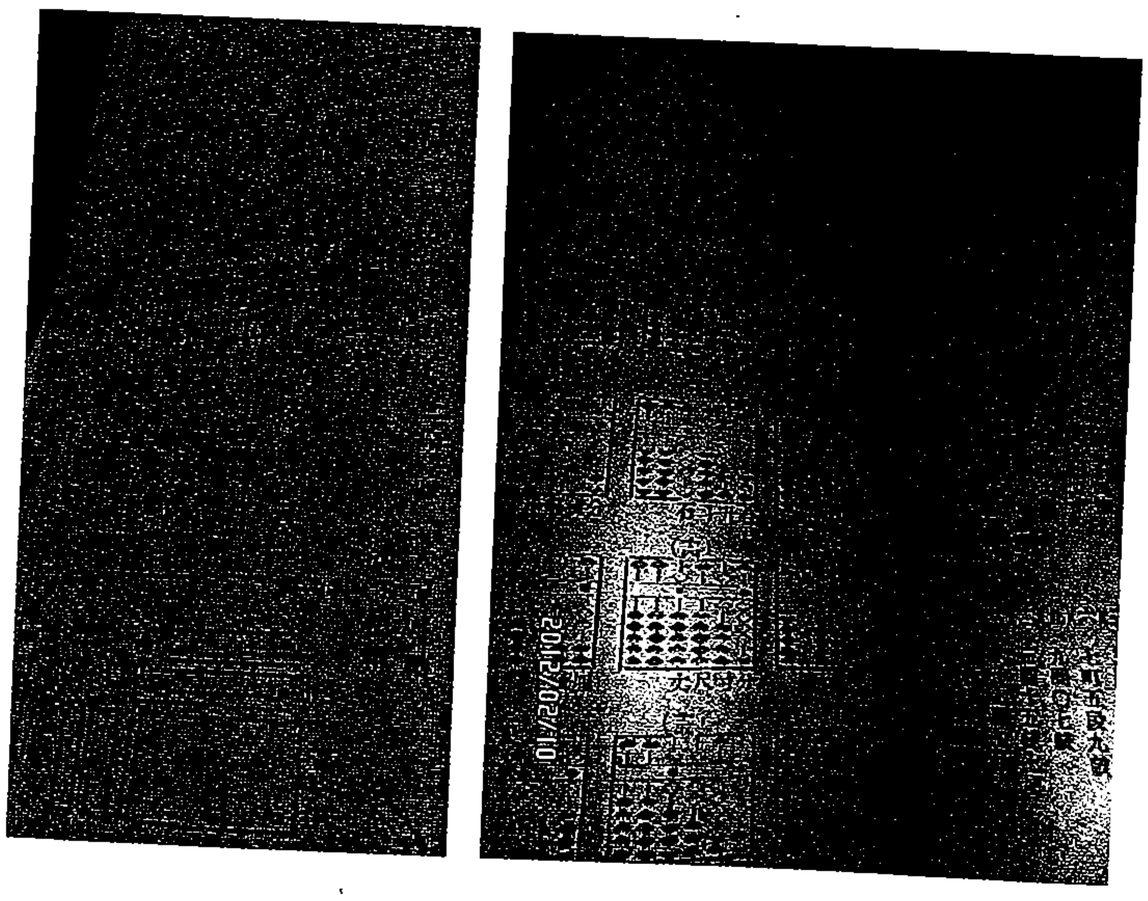
http://matrix.skku.ac.kr/CLA12/Sage-5-1-4.html

$$A = \begin{bmatrix} 3 & 2 & 3 \\ 2 & 6 & 6 \\ 3 & 6 & 11 \end{bmatrix},\ \mathbf{x}_0 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$A^3\mathbf{x}_0 = \begin{bmatrix} 13 \\ 14 \end{bmatrix}$$

$$A^4\mathbf{x}_0 = \begin{bmatrix} 41 \\ 40 \end{bmatrix}$$

$$A^5\mathbf{x}_0 = \begin{bmatrix} 121 \\ 122 \end{bmatrix}$$ ■



# 5.4 Markov 체인과 선형 모델

YouTube 동영상 강의 : http://youtu.be/156ezier6HQ

사회과학분야, 특히 경영이나 경제에서 어떤 현상에 대해 분석하거나 이론을 정확히 표현하기 위해서 우리가 배우고 공부하는 수학적 기법을 이용한다. 경영분석을 위한 다양한 수학적 도구 중 하나로 Markov 체인이 쓰이곤 한다.

Markov 체인은 '과거와 현재가 주어진 상황에서 미래는 과거와 관계없이 현재에만 의존한다.'라는 Markov 성질을 가진 확률과정인 Markov 과정(process) 중에서 시간공간과 상태공간이 이산적인 경우를 말한다. 이런 Markov 체인은 다양한 경영 시스템을 모형화하는 수학적 기법으로써 과거에 있었던 변화를 토대로 시스템의 여러 변수들이 갖고 있는 동적 성격을 파악하여 미래에 있을 변화를 연속적으로 예측하는 데 사용된다. 이를 이용하는 분야는 생산관리나 재고관리, 판매계획, 상표의 대체 등 경영 전반에 이용하게 된다.

이제 이런 Markov 체인이 무엇인지에 대해 알아보자. 다음에 소개하는 Markov 체인의 모델은 여러 교재에 소개되는 단순하면서도 정확한 개념을 전달하는 예이다.

경쟁관계에 있는 TV뉴스채널인 채널 1과 채널 2는 현재 각각 60%와 40%의 시청자를 확보하고 있다. 1년마다 채널 2의 시청자 중 20%가 채널 1로 이동하고, 채널 1의 시청자 중 30%가 채널 2로 이동한다. 이때 1년 후의 각 채널의 시청자 비율은 어떻게 되었는지를 알아보자. 일단 시간에 종속된 변수들을 소개하면

$$x_1(t) = t \text{ 시점에서 채널 1에 속한 시청자 비율}$$
$$x_2(t) = t \text{ 시점에서 채널 2에 속한 시청자 비율}$$

그리고 열벡터

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} \begin{matrix} \leftarrow t \text{ 시점에서 채널 1에 속한 시청자 비율} \\ \leftarrow t \text{ 시점에서 채널 2에 속한 시청자 비율} \end{matrix}$$

의 변수 $x_1(t)$와 $x_2(t)$는 시간이 $t$일 때 상태가 벡터 $\mathbf{x}(t)$인 Dynamical System을 구성한다. 시간 $t$가 0일 때 두 채널이 가지고 있는 시청자 비율은 각각 60%, 40%라 하면 그 때의 상태는

$$\mathbf{x}(0) = \begin{bmatrix} x_1(0) \\ x_2(0) \end{bmatrix} = \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix} \begin{matrix} \leftarrow 0 \text{ 지점에서 채널 1의 시청자 비율} \\ \leftarrow 0 \text{ 지점에서 채널 2의 시청자 비율} \end{matrix}$$

위의 시간은 연(year) 단위이다. 이제 1년이 지난 후, 즉 $t=1$일 때 상태를 알아보자. 1년이 지난 후 채널 1에는 처음 60% 시청자 중 70%(채널 2에 30%를 잃음)를 유지하고, 채널 2의 처음 시청자

바율 40% 중 20%를 새로 확보한다. 그러면

$$x_1(1) = 0.7(0.6) + 0.2(0.4) = 0.5$$

이다.

이와 마찬가지로, 채널 2는 채널 1의 처음 시청자인 60% 중 30%를 새로 확보하고, 자신의 처음 시청자인 40% 중에서 80%를 유지한다. 그러면

$$x_2(1) = 0.3(0.6) + 0.8(0.4) = 0.5$$

이다.

그러므로 $t = 1$일 때 시청자 확보율의 상태는

$$\mathbf{x}(1) = \begin{bmatrix} x_1(1) \\ x_2(1) \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} \begin{array}{l} \leftarrow 1 \text{ 지점에서 채널 1의 시청자 비율} \\ \leftarrow 1 \text{ 지점에서 채널 2의 시청자 비율} \end{array}$$

이다. 즉 채널 1과 2는 1년 후 각각 50%와 50%의 시청자를 갖게 된다는 것을 알게 된다. 같은 식에서 5년 후의 각 채널의 시청자 비율을 예측하기 위하여 위 식을 행렬의 형태로 표현하면,

$$\mathbf{x}(k+1) = \begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 0.7 & 0.2 \\ 0.3 & 0.8 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} = \begin{bmatrix} 0.7 & 0.2 \\ 0.3 & 0.8 \end{bmatrix} \mathbf{x}(k)$$

이다.

이를 통해 $k = 0$일 때 이 방법을 이용하면

$$\mathbf{x}(1) = \begin{bmatrix} x_1(1) \\ x_2(1) \end{bmatrix} = \begin{bmatrix} 0.7 & 0.2 \\ 0.3 & 0.8 \end{bmatrix} \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

가 된다.

이런 방법으로 $\mathbf{x}(5)$의 값을 구하면

$$\mathbf{x}(2) = \begin{bmatrix} 0.7 & 0.2 \\ 0.3 & 0.8 \end{bmatrix} \mathbf{x}(1) = \begin{bmatrix} 0.7 & 0.2 \\ 0.3 & 0.8 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 0.45 \\ 0.55 \end{bmatrix},$$

$$\mathbf{x}(3) = \begin{bmatrix} 0.7 & 0.2 \\ 0.3 & 0.8 \end{bmatrix} \mathbf{x}(2) = \begin{bmatrix} 0.7 & 0.2 \\ 0.3 & 0.8 \end{bmatrix} \begin{bmatrix} 0.45 \\ 0.55 \end{bmatrix} = \begin{bmatrix} 0.425 \\ 0.575 \end{bmatrix},$$

$$\mathbf{x}(4) = \begin{bmatrix} 0.4125 \\ 0.5875 \end{bmatrix}, \quad \mathbf{x}(5) = \begin{bmatrix} 0.40625 \\ 0.59375 \end{bmatrix}.$$

5년 뒤에 채널 1은 약 40%의 시청자를 보유하고, 채널 2는 약 60%의 시청자를 보유한다는 것을 알 수 있다. 더 나아가 이 과정을 계속하여 장기적으로 어런 시장 점유율의 변화를 지켜보면,

$$\mathbf{x}(40) = \begin{bmatrix} 0.4 \\ 0.6 \end{bmatrix} = \begin{bmatrix} 0.7 & 0.2 \\ 0.3 & 0.8 \end{bmatrix}^{40} \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

가 된다. 즉 40년 뒤에는 결국 채널 1은 전체 시장 점유율 40%을 가지고 채널 2는 전체 시장 점유율 60%를 가지게 된다는 예측이 가능하다. 이에 따라 각 회사는 경영 전략을 세우고 변화를 모색하

게 되는 것이다.

이와 같이 주위의 변화하는 상황을 선형연립방정식으로 만들어 예측을 가능하게 하는 모델을 **선형 모델**이라고 하고, 위와 같은 예 중에서 특히 그 시스템의 변수들의 상태는 확실하게 알 수 없지만, 확률적으로 표현할 수 있다는 의미에서 **확률과정**(stochastic process)이라고 부른다.

즉 Markov 체인이란 연속적인 시간공간상에서 성분들의 합이 1이 되는 확률벡터(stochastic vector) $\mathbf{x}(k)$ 사이에 식

$$\mathbf{x}(k+1) = P\mathbf{x}(k)$$

로 관계되어 있는 시스템을 말하며, 사이에 있는 행렬 $P$를 이 시스템의 **전이행렬**(transition matrix)이라 한다.

여기서 잠깐

**Andrey Markov(1871~897), 러시아 수학자**

Markov 체인은 러시아의 수학자 Markov의 이름을 기리기 위해 붙여졌다. Markov는 시를 사랑했으며 1913년 푸시킨의 시 Eugene Onegin에서 Markov 체인을 이용하여 자음과 모음의 교대 형태를 분석했다. Markov는 자신의 체인이 문학적인 작품의 분석 정도에나 쓰이는 것이라고 생각했었는데, 그가 만일 살아있었다면 Markov 체인이 지금 문학, 사회과학은 물론 물리학, 생물학 등에 널리 사용된다는 것을 보고 매우 놀랐을 것이다.

**예제 1** 처음 시내에 사는 사람이 100,000명이고 교외에는 25,000명이 산다. 지역계획위원회에서는 매년 5%의 시내주민을 교외로 이사시키고 3%의 교외주민을 시내로 이사시키는 정책을 수립하기로 결정을 했다.

(1) 전체 인구수는 일정하다고 할 때, 5년 동안 시내와 교외의 인구수를 표를 만들어서 보아라.

(2) 오랜 시간 후에 시내와 교외의 인구가 어떻게 분포되겠는가?

**풀이** (1) $\mathbf{x}_0 = \begin{bmatrix} 100{,}000 \\ 25{,}000 \end{bmatrix}$, $P = \begin{bmatrix} 0.95 & 0.03 \\ 0.05 & 0.97 \end{bmatrix}$ (column stochastic 행렬)

$$\mathbf{x}_1 = \begin{bmatrix} 0.95 & 0.03 \\ 0.05 & 0.97 \end{bmatrix}\mathbf{x}_0 = \begin{bmatrix} 100{,}000 \times 0.95 + 25{,}000 \times 0.03 \\ 100{,}000 \times 0.05 + 25{,}000 \times 0.97 \end{bmatrix} = \begin{bmatrix} 95750 \\ 29250 \end{bmatrix}$$

$$\mathbf{x}_2 = P\mathbf{x}_1 = \begin{bmatrix} 0.95 & 0.03 \\ 0.05 & 0.97 \end{bmatrix}\begin{bmatrix} 95{,}750 \\ 29{,}250 \end{bmatrix}$$

$$= \begin{bmatrix} 95{,}750 \times 0.95 + 29{,}250 \times 0.03 \\ 95{,}750 \times 0.05 + 29{,}250 \times 0.97 \end{bmatrix} = \begin{bmatrix} 91{,}840 \\ 33{,}160 \end{bmatrix}$$

$$\mathbf{x}_3 = P\mathbf{x}_2 = \begin{bmatrix} 0.95 & 0.03 \\ 0.05 & 0.97 \end{bmatrix}\begin{bmatrix} 91{,}840 \\ 33{,}160 \end{bmatrix}$$

$$= \begin{bmatrix} 91{,}840 \times 0.95 + 33{,}160 \times 0.03 \\ 91{,}840 \times 0.05 + 33{,}160 \times 0.97 \end{bmatrix} \approx \begin{bmatrix} 88{,}243 \\ 36{,}757 \end{bmatrix}$$

$$\mathbf{x}_4 = P\mathbf{x}_3 = \begin{bmatrix} 0.95 & 0.03 \\ 0.05 & 0.97 \end{bmatrix} \begin{bmatrix} 88{,}243 \\ 36{,}757 \end{bmatrix}$$

$$= \begin{bmatrix} 88{,}243 \times 0.95 + 36{,}757 \times 0.03 \\ 88{,}243 \times 0.05 + 36{,}757 \times 0.97 \end{bmatrix} \approx \begin{bmatrix} 84{,}934 \\ 40{,}066 \end{bmatrix}$$

$$\mathbf{x}_5 = P\mathbf{x}_4 = \begin{bmatrix} 0.95 & 0.03 \\ 0.05 & 0.97 \end{bmatrix} \begin{bmatrix} 84{,}934 \\ 40{,}066 \end{bmatrix}$$

$$= \begin{bmatrix} 84{,}934 \times 0.95 + 40{,}066 \times 0.03 \\ 84{,}934 \times 0.05 + 40{,}066 \times 0.97 \end{bmatrix} \approx \begin{bmatrix} 81{,}889 \\ 43{,}111 \end{bmatrix} \quad (*)$$

**표 5.4.1** 5년간의 인구수 변화

| | 시내 | 교외 |
|---|---|---|
| 현재($\mathbf{x}_0$) | 100,000 | 25,000 |
| 1년 후($\mathbf{x}_1$) | 95,750 | 29,250 |
| 2년 후($\mathbf{x}_2$) | 91,840 | 33,160 |
| 3년 후($\mathbf{x}_3$) | 88,243 | 36,757 |
| 4년 후($\mathbf{x}_4$) | 84,934 | 40,066 |
| 5년 후($\mathbf{x}_5$) | 81,889 | 43,111 |

### Sage-math 실습

(1)을 Sage-math로 구현해보자. (# 이후의 내용은 주석이다.")

http://math1.skku.ac.kr/home/pub/431

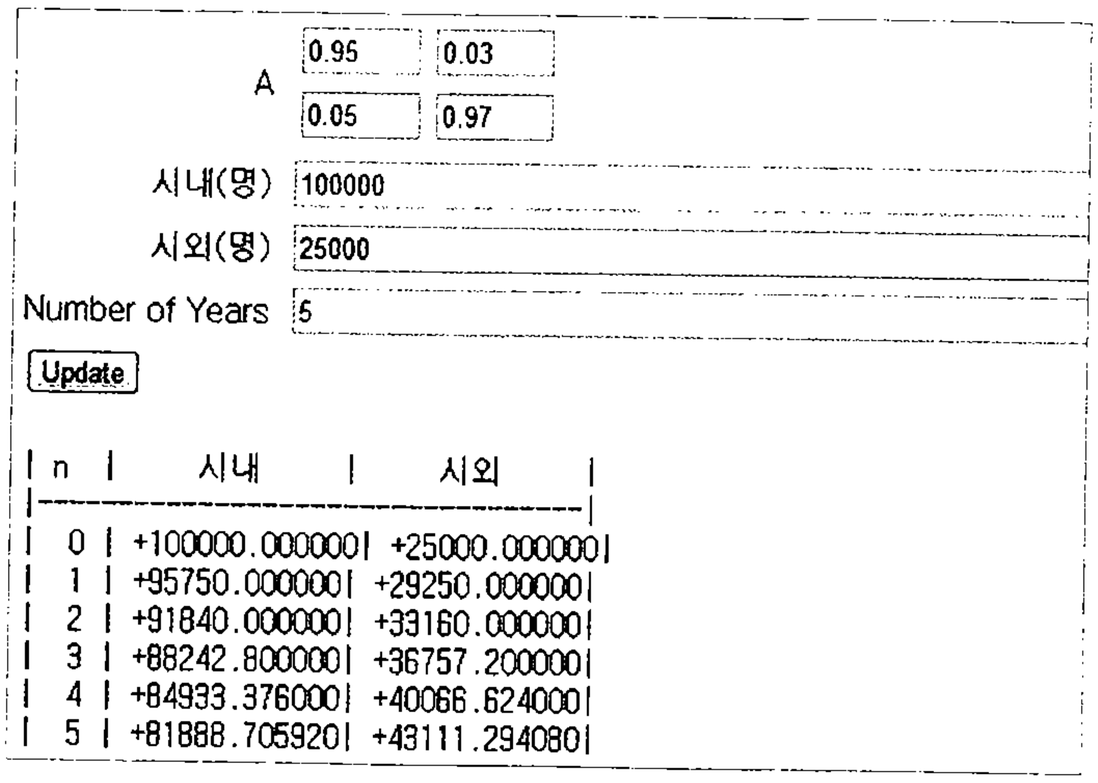



**그림 5.4.1**

5년 후 시내와 교외의 인구수는 다음과 같다.

$$\mathbf{x}_5 = P\mathbf{x}_4 = \begin{bmatrix} 81,889 \\ 43,111 \end{bmatrix} \qquad (*)$$

이렇게 계산한 인구 변화를 살펴보면 다음과 같은 그래프를 구할 수 있다.

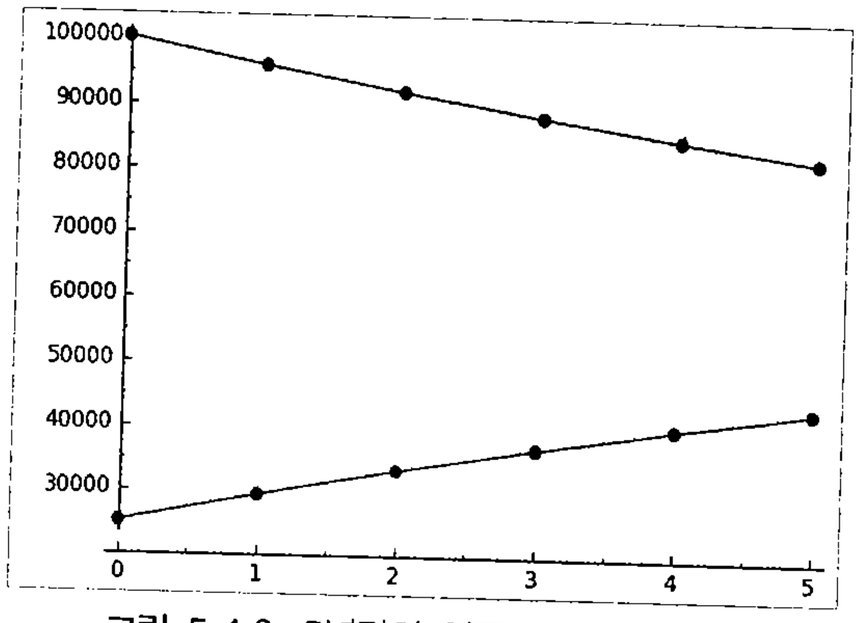



그림 5.4.2 5년간의 인구수 변화 그래프

```
P = matrix(2, 2, [0.95, 0.03, 0.05, 0.97]) # transition matrix
print(P)                                   # print the matrix P
```

행렬 $P$는 전이행렬(transition matrix)로 1열은 시내주민의 이동을, 2열은 시외주민의 이동을 나타냈다.

```
x0 = matrix(2, 1, [100000, 25000]) # stochastic vector x(0)
print(x0)                          # print x(0)
```

```
x1 = P*x0                          # stochastic vector x(1)
print(x1.apply_map(round))         # print x(1) using round function that is round off the
                                     number.
```

```
city1=vector([x0[0,0],x1[0,0],x2[0,0],x3[0,0],x4[0,0],x5[0,0]]); city1
city2=vector([x0[1,0],x1[1,0],x2[1,0],x3[1,0],x4[1,0],x5[1,0]]); city2
lp1=list_plot(city1,plotjoined=True,rgbcolor=(1,0,0)) # city1 : red line
lp2=list_plot(city2,plotjoined=True,rgbcolor=(0,1,0)) # city2 : green line
show(lp1+lp2)
```

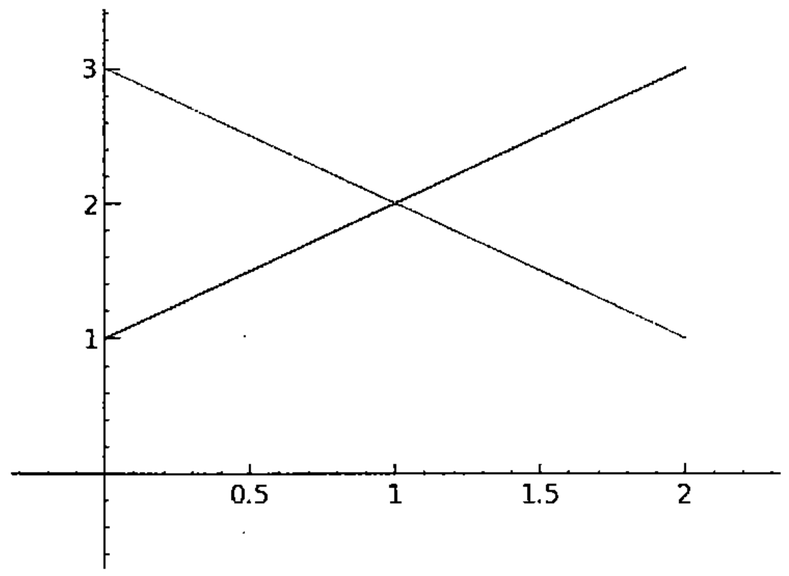



그림 5.4.3

(2) 여기에서 regular[1]인 전이행렬 $P=\begin{bmatrix}0.95 & 0.03\\ 0.05 & 0.97\end{bmatrix}$과 확률변수인 q에 대하여 $(I-P)\,\mathrm{q}=0$을 이용하면

$$\begin{bmatrix}0.05 & -0.03\\ -0.05 & 0.03\end{bmatrix}\begin{bmatrix}q_1\\ q_2\end{bmatrix}=\begin{bmatrix}0.05q_1-0.03q_2\\ -0.05q_1+0.03q_2\end{bmatrix}=0$$

이다. 그러면 $q_1=\frac{3}{5}s$, $q_2=s$라고 두면 $q$는 확률벡터이기 때문에

$$q_1+q_2=\frac{8}{5}s=1$$

이다. 즉 $s=q_2=\frac{5}{8}$, $q_1=\frac{3}{8}$이다. 이는 $P^k\mathbf{x}_0\to q,\ k\to\infty$이다.

그러므로 오랜 시간 후에 인구 분포는 $125000\begin{bmatrix}\frac{3}{8}\\ \frac{5}{8}\end{bmatrix}=\begin{bmatrix}46{,}875\\ 78{,}125\end{bmatrix}$(**)이 될 것이다. ■

**예제 2** $A$는 미팅할 때마다 만나는 파트너의 성격, 지식, 용모 등을 고려하여 5등급으로 분류한다. 1등급은 더할 나위 없이 마음에 드는 파트너(eye-opener)이고 5등급은 피하고 싶은 파트너(disheartener)이다. 그는 내심 다음과 같은 마음의 결정을 보았다.

(1) 1등급 파트너를 만나면 "애프터"를 신청하고, 신청이 받아들여지면 사귄다. (받아들여지지 않으면 계속 미팅을 한다. 받아들여질 확률은 0.3)

(2) 5등급 파트너를 만나면 미팅과는 인연이 없는 것으로 알고 영원히 미팅계를 떠난다.

그는 과거 수많은 자신의 미팅을 분석한 결과 등급에 관한 다음과 같은 전이행렬을 얻었다.

1) regular : column stochastic 행렬이면서 positive definite의 의미이다.

$$P=\begin{bmatrix}0.05 & 0.07 & 0.03 & 0.08 & 0.1\\ 0.2 & 0.25 & 0.2 & 0.3 & 0.3\\ 0.4 & 0.4 & 0.4 & 0.3 & 0.3\\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2\\ 0.15 & 0.08 & 0.17 & 0.12 & 0.1\end{bmatrix}$$

가장 최근의 미팅파트너는 3등급이었다.

$A$의 미팅 일생이 결국 해피엔딩으로 끝날 확률을 구하여라.

풀이

$$P\begin{bmatrix}0\\0\\1\\0\\0\end{bmatrix}=\begin{bmatrix}0.05 & 0.07 & 0.03 & 0.08 & 0.1\\ 0.2 & 0.25 & 0.2 & 0.3 & 0.3\\ 0.4 & 0.4 & 0.4 & 0.3 & 0.3\\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2\\ 0.15 & 0.08 & 0.17 & 0.12 & 0.1\end{bmatrix}\begin{bmatrix}0\\0\\1\\0\\0\end{bmatrix}=\begin{bmatrix}0.03\\0.2\\0.4\\0.2\\0.3\end{bmatrix}$$

이다. 이때 1등급의 파트너일 확률은 0.03이다. 그럼 이 파트너가 애프터를 받아들일 확률은 0.3이므로 그녀와 함께 행복하게 미팅계를 떠날 확률은 $0.03\times 0.3=0.009$이다. ■

이 절에서는 시간의 흐름에 따라 현상을 파악해나가면서 행렬의 거듭제곱의 성질들을 이용하여 초기값을 대입하여 시간이 지남에 따라 궁극적으로 그대로 둔다면 어떤 결과에 이르게 되는지를 살펴볼 수 있었다.

여기서 잠깐

Markov 체인에 대하여 더 자세한 사항을 알고 싶은 독자는 다음과 같은 웹사이트나 책을 참고할 수 있다.

http://nlp.korea.ac.kr/new/seminar/2002winter/master_fsnlp/ch9.ppt

Kemeny and J.Snell, Finite Markov Chains, Springer-Verlag, New York, 1976

http://journals.cambridge.org/action/displayFulltext?type=1&fid=637500&jid=SIC&volumeId=19&issueId=04&aid=637496

1973년 경제학자 레온티예프(Wassily Leontief)는 경제현상에서 다양한 요인들 사이의 관계를 연구하며 행렬을 이용한 방법을 소개한 공로로 노벨 경제학상을 수상하였다. 이 절에서는 마지막으로 레온티예프가 소개한 아이디어인 투입산출모델의 일부를 소개한다.

아래의 경제시스템을 생각해보자. 입력과 출력은 1만원 단위로 생각하자. 다음 표에서는 만원가치의 출력(생산품)을 생산하기 위해 요구되는 입력(투자)을 만원단위로 표시했다고 하자.

표 5.4.2

| | 출력 | | | |
|---|---|---|---|---|
| 입력 | | 생산업 | 농업 | 기타 |
| | 생산업 | 0.6 | 0.2 | 0.2 |
| | 농업 | 0.1 | 0.6 | 0.1 |
| | 기타 | 0.2 | 0.1 | 0.4 |

예

# 5.5 선형대수학과 구글 검색엔진– 페이지랭크 알고리즘

YouTube 동영상 강의 : http://youtu.be/WNUoXLh8i_E

영국 파이낸셜타임스(FT)와 컨설팅그룹 밀워드브라운이 공동 조사한 '2010 글로벌 100대 브랜드 기업'을 보면 랭킹 1위는 미국의 인터넷 기업 구글로, 브랜드 가치가 1,143억 달러를 기록해 2007년부터 4년 연속 최고의 자리를 지켰다.[3] 최근 구글의 모토로라 인수가 삼성과 한국의 경제에 큰 영향을 줄 것이라는 언론 보도를 접하였듯이 격변하는 환경에서 '모든 학생은 왜 수학을 배워야 하는가?'에 대한 답의 하나로 이 절에서는 경제와 생활에 막대한 영향을 미치는 "구글 검색엔진 속의 수학이론"에 대하여 소개한다.

구글(Google)의 역사는 《Matrix Computation》의 저자 Gene Golub이 강의하던 스탠포드대에 다니던 20대 초반의 대학원생 페이지(Larry Page)와 브린(Sergey Brin)이 1995년 기존의 검색엔진들이 나열해주는 정리되지 않는 검색 결과의 목록에 불만을 가지고 새로운 검색기법 연구에 몰두하여 마침내 페이지랭크(PageRank) 기술과 링크(link)의 매칭기술을 개발하는 데 성공하면서 시작된다. 이들은 '이 사이트가 링크를 만들어 연결시켜 줄 얼마만큼의 가치를 가지고 있을까?'라는 기준을 가지고 검색된 사이트들의 순위를 매기는 구글 특유의 차별화된 검색방법을 확립한다.

미국인 수학자 케스너(Edward Kasner)의 조카인 시로타(Milton Sirota)가 10의 100제곱을 뜻하는 말로 만든 'googol'이라는 신조어를 변형시켜 페이지와 브린이 처음 구글이란 말을 쓴 것은 1997년이었다. 그들은 인터넷의 광대한 정보를 구글이 모두 담겠다는 의지를 이 용어에 담았다.

1998년 4월 개최된 월드와이드웹 컨소시엄(W3C)에서 페이지와 브린은 자신들의 연구 결과를 발표하게 되고, 이때부터 검색엔진 업계와 학계에서 관심을 끌기 시작했다. 구글은 사이트가 가지고 있는 메타태그나 키워드에만 의지하지 않고 페이지랭크라는 독특한 기법을 사용하여 웹페이지의 합리적인 순위를 매긴다.

순수하게 수학적으로 접근한 구글 검색엔진의 새로운 아이디어와 수학적 알고리즘인 구글행렬과 페이지랭크를 계산하는 방법은 인터넷 검색엔진 시장에 획기적인 새 패러다임을 제공하였다. 이를 통하여 학생들이 대학에서 배우는 선형대수학의 기본적인 내용이 생활 속에 널리 이용되고 있음을 확인할 수 있다.

3) 2011년에는 애플이 구글을 제치고 브랜드 가치 세계 1위를 차지했다고 보도했다. 구글은 2위(1,724억 달러,약 180조원)로 밀려났다. 유일하게 포함된 국내 기업 삼성은 67위를 차지했다.

## 1. 구글 검색엔진의 기본 아이디어

우리는 구글 검색엔진에 사용된 페이지랭크를 통한 웹페이지의 외부인기도가 검색 순위에 반영될 때 페이지랭크 점수가 어떻게 구해지는지 알아보고 그 뒤에 숨어 있는 수학적 아이디어의 핵심을 찾아 설명하도록 한다.

웹페이지에 대한 객관적인 순위를 만들어내기 위하여 구글은 인터넷의 광범위한 구조를 직접 이용한다. 대부분의 다른 검색엔진들은 관계있는 사이트를 결정하기 위해 웹페이지의 제목과 내용을 체크한다. 그러나 구글은 한 웹페이지에서 다른 웹페이지로 연결하는 링크가 있으면, 그 링크를 일종의 투표로 보고, 많이 투표된 웹페이지를 중심으로 평가점수를 부여한다. 구글의 페이지랭크 방식은 어떤 웹페이지가 다른 웹페이지와 밀접한 관계가 있는지를 결정하고 관련된 구조를 표현하기 위해 하이퍼링크[4] 행렬을 만들고, 관련 행렬의 가장 큰 고유값을 이용해서 검색에서 가장 근접한 사이트들을 찾아내는 것이다. 이 과정에서 구글은 거듭제곱법와 페이지랭크 알고리즘을 이용한다.

구글과 같은 검색엔진이 수행하는 기본적인 업무 중 첫 번째는 인터넷에 상주하며 사용자들이 접근하는 웹페이지를 파악하는 것이다. 두 번째 작업은 파악된 웹페이지에 대한 데이터를 수집하는 것이다. 수집된 데이터들은 검색어와 관련된 단어나 문구를 검색하는 데 효과적으로 사용하게 된다. 세 번째 단계는 데이터로 수집한 웹페이지에 중요도(웹페이지가 가지고 있는 중요성을 다른 웹페이지와 비교하여 점수로 표현한 것)를 기록하고 사용자가 검색을 할 때 수집하여 가지고 있던 웹페이지들의 데이터 중에 좀더 중요한 자료를 검색결과의 상단에 보여주는 것이다.

우리는 이 세 번째 과정에서 수집된 웹페이지 데이터들의 집합에서 각각의 웹페이지에 대한 중요도를 어떠한 방법으로 결정하며 그 비율을 어떻게 정했는지에 대해 알아본다.

## 2. 페이지랭크 공식의 건설

구글의 페이지랭크(PageRank)라는 개념은 쉽게 이야기해서 '사람들이 링크를 많이 거는 웹페이지는 사람들이 많이 찾아가는 곳일 테고, 그 만큼 정확한 정보가 있는 웹페이지일 것이므로 웹페이지의 페이지랭크 점수를 높게 주자.'라는 이론이다.

페이지랭크는 임의의 방문자가 어떤 웹페이지를 방문하는 빈도로 이해될 수도 있다. 새로운 웹페이지는 자신이 담고 있는 링크의 적절성에 따라 페이지랭크를 증가시키기도 하고, 감소시키기도 한다. (Avrachenkov & Litvak, 2005)

이제 구글 검색엔진 안의 수학적 모델에 대하여 살펴보자.

---

4) 하이퍼링크(Hyperlink): 〈컴퓨터 전문용어〉 하이퍼텍스트 문서 내의 단어나 구, 기호, 화상과 같은 요소를 그 하이퍼텍스트 내의 다른 요소 또는 다른 하이퍼텍스트 내의 다른 요소와 연결한 것. 하이퍼링크 행렬을 링크행렬이라 부르기로 한다.

1단계 : 인접(adjacency)행렬의 건설

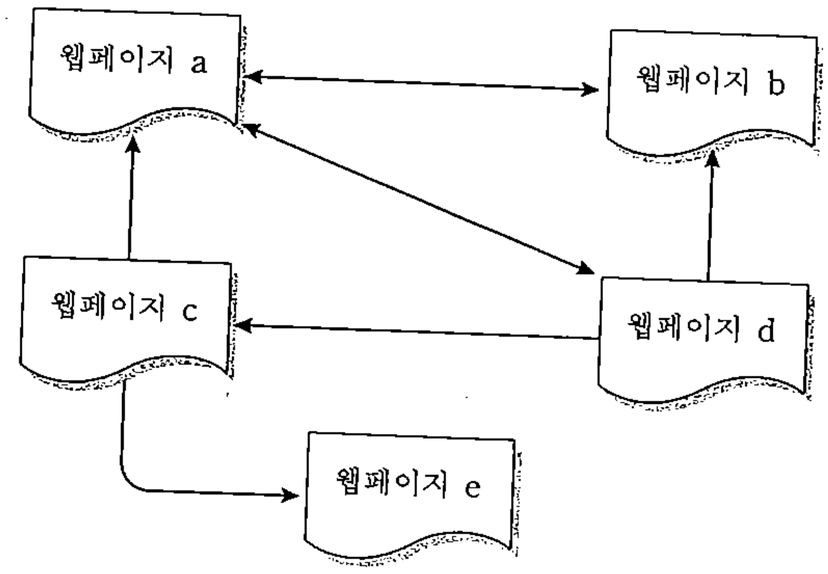



**그림** 5.5.1 5개 웹페이지로 이루어진 간단한 웹

위의 그림은 간단한 웹을 표현한 것이다. 주어진 웹페이지들로부터 얻어진 위의 연결관계를 다음과 같이 $n \times n$ 인접행렬 $A$를 사용하여 나타낼 수 있다. (Page; Brin; Motwani & Winograd, 1998)

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{1}$$

이 경우 그림 5.5.1의 웹페이지 e는 다른 웹페이지로의 링크를 가지고 있지 않으므로 행렬 $A$의 마지막 열은 0의 값을 갖는다. Dangling node[5]라 불리는 외부로의 링크를 가지지 않는 웹페이지는 특별한 케이스이다. 게다가 인접행렬을 만들 때 각 웹페이지의 자기 자신으로의 링크는 무시하므로 행렬 $A$의 대각선 성분은 모두 0의 값을 갖는 것을 알 수 있다.

2단계 : 열정규화된 인접행렬 $H$의 건설

행렬 $A$의 각각의 열 $A_j$를 $A_j$의 성분들의 합으로 나누어 새로운 행렬 $H$를 얻는다. 이 행렬을 열정규화된 인접행렬이라고 한다. 행렬 $H$의 열을 $H_j$라 하면 다음과 같은 공식에 의해 얻어진다.

$$H_j = \frac{A_j}{\sum_{k=1}^{n} A_{kj}}, \quad j = 1, \cdots, n$$

이 과정을 식 (1)의 행렬 $A$에 적용하면

5) dangling node: 다른 웹페이지들과 연결이 끊겨 버린 상태

$$H=\begin{bmatrix} 0 & 1 & \frac{1}{2} & \frac{1}{3} & 0 \\ \frac{1}{2} & 0 & 0 & \frac{1}{3} & 0 \\ 0 & 0 & 0 & \frac{1}{3} & 0 \\ \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 0 \end{bmatrix} \tag{2}$$

을 얻는다.

3단계: 행렬 $H$의 열 stochastic화 (각 열성분의 합이 모두 1인 행렬로 만드는 과정)

전 단계에서 얻어진 행렬 $H$가 바로 페이지랭크 알고리즘에 이용되는 것은 아니다. 그 이유는 페이지랭크 알고리즘의 수렴성을 확보하기 위해서는 stochastic 행렬이 필요한데, 행렬 $H$는 dangling node에 의해 열 stochastic 행렬이 아닐 수 있기 때문이다. 열 stochastic 행렬은 '행렬의 모든 성분이 음수가 아니어야 하고 열의 합이 모두 1이어야 한다'를 만족해야 한다. 일단 위의 행렬 $H$는 dangling node인 웹페이지에 의해 마지막 열의 합이 0이므로 열 stochastic 행렬이 아니다. 그래서 행렬 $H$로부터 모든 열의 합을 1이 되는 열 stochastic 행렬 $S$를 다음과 같이 정의한다.

$$S=H+\frac{\mathrm{ea}^T}{n}$$

여기서 벡터 e는 모든 성분이 1인 열벡터이고, a는 $\sum_{i=1}^{n} H_{ij}=0$이면 $a_j=1$이고 $\sum_{i=1}^{n} H_{ij}\neq 0$이면 $a_j=0$인 열벡터이다. 그러면 행렬 $H$로부터 열 stochastic 행렬 $S$를 건설할 수 있다. Gerschgorin 정리에 의해 열 stochastic 행렬(혹은 Markov 행렬) $S$의 가장 큰 고유값의 크기 $\rho(S)$는 1보다 작거나 같다. 또한 $S-I$의 행벡터(혹은 열벡터)의 합은 0이 되는데, 이것은 trivial sum이 아니다. 따라서 $\det(S-I)=0$으로부터 어떤 $i\in 1,\cdots,n$에 대하여 $\lambda_i=1$인 고유값이 존재함을 알 수 있다. 그런데 문제점은 링크행렬이 크기가 매우 큰 sparse 행렬이라는 것이다. 따라서 일반적인 계산을 통해 고유벡터를 구하기보다는 거듭제곱법을 사용하는 것이 효율적이다. 우선 식 (2)의 행렬 $H$를 사용하여 행렬 $S$를 만들면 다음과 같이 된다.

$$S=\begin{bmatrix} 0 & 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{5} \\ \frac{1}{2} & 0 & 0 & \frac{1}{3} & \frac{1}{5} \\ 0 & 0 & 0 & \frac{1}{3} & \frac{1}{5} \\ \frac{1}{2} & 0 & 0 & 0 & \frac{1}{5} \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{5} \end{bmatrix} \tag{3}$$

4단계 : 구글행렬 $G$의 건설

행렬 $S$가 유일한 stationary distribution 벡터를 갖는다는 것을 보증할 수 없기 때문에 아직 끝난 것이 아니다. (즉 정확한 하나의 페이지랭크 벡터가 없을 수도 있다는 의미) 따라서 수렴하는 하나의 stationary distribution 벡터 x가 있다는 것을 보장하기 위해 행렬 $S$가 irreducible인 동시에 stochastic임을 확실히 해야 한다. 정의에 의하여 "주어진 정사각행렬 $A$에 대하여, $P^TAP=\begin{pmatrix} X & Y \\ 0 & Z \end{pmatrix}$인 치환행렬 $P$와 정사각행렬 $X$, $Z$가 존재하지 않을 때 행렬 $A$는 irreducible이다." 행렬이 irreducible인 것은 대응하는 그래프가 strongly connected하다는 것과 동치이다(Horn & Johnson, 1985). 또 $A$가 primitive 행렬이면, 언제나 $A$는 irreducible 행렬이다. 이에 보태서 주어진 정사각행렬 $A$가 primitive일 필요충분조건은 어떤 자연수 $k$에 대하여 $A^k$가 양의 행렬이 되는 것이다.[6] 그래서 우리는 식 (3)의 행렬 $S$를 이용하여 irreducible한 열 stochastic 행렬 $G$를 다음과 같이 정의한다.

$$G = mS + (1-m)E \tag{4}$$

여기서 $m$은 $0 \le m \le 1$이고 $E = \frac{\mathbf{e}\mathbf{e}^T}{n}$이다. 구글 검색에서는 보통 $m$을 0.85로 이용한다. 이 행렬 $G$가 바로 우리가 필요로 했던 구글행렬이 된다.

이제 페이지랭크를 구하는 기본 알고리즘인 식 $\mathbf{x}_{k+1} = H\mathbf{x}_k$에 행렬 $H$ 대신 행렬 $G$를 대입하여 새 페이지랭크 알고리즘을 다음과 같이 정의한다.

$$\mathbf{x}_{k+1} = G\mathbf{x}_k$$

그러나 크기가 1인 고유값이 2개 이상 존재하면 거듭제곱법의 수렴성이 보장되지 않는다. 또한 행렬의 주위에 블록 $O$행렬이 있다면 페이지 랭킹에서 아예 제외되는 웹페이지가 생기는데, 이런 경우 현실 문제를 적절히 반영한다고 볼 수 없다. 따라서 우리가 다루기를 원하는 행렬의 의미를 모두 담고 있으면서 가장 큰 고유값이 1개뿐인(대수적 중복도가 1인) 행렬인 primitive 행렬로 변환시킨 것이 구글행렬이다. 이 부분의 이론에 대하여 소개하고 다음 단계로 넘어간다.

집합 $\sigma(G)$를 행렬 $G$의 고유값들의 집합, 정사각행렬 $G$의 spectral radius를 $\rho(G) = \max_{\lambda \in \sigma(G)} |\lambda|$라 하면 Perron-Frobenius 정리에 의해 구글행렬 $G = mS + (1-m)E$에 대하여 $\rho(G)$=1인 고유값 1은 $G$의 Gerschgorin circle 상의 유일한 고유값이고, 고유값 1의 대수적 중복도는 1임을 알게 된다. 그리고 $\lambda = \rho(G)$는 행렬 $G$의 유일한 가장 큰 고유값이 된다. 더구나 대응하는 그래프는 strongly connected이고 어떤 자연수 $k$에 대하여 $G^k$가 양의 행렬이 되는 irreducible 행렬 $G$는 primitive 행렬이다. 즉 구글행렬 $G = mS + (1-m)E$가 primitive 행렬이라는 의미이다(Horn & Johnson, 1985).

여기서 $\lambda > 0$이고 $\lambda \in \sigma(G)$인 대수적 중복도가 1인 가장 큰 고유값 $\lambda = \rho(G)$를 $G$의 Perron 근(root)이라 하고, $G\mathbf{x} = \lambda\mathbf{x}$인 양의 고유벡터 $\mathbf{x} > 0$가 존재한다. 그리고 $G\mathbf{x} = \lambda\mathbf{x}$이면서 $\mathbf{x} > 0$이

6) http://matrix.skku.ac.kr/sglee/perron_frobenius/perron_frobenius.html

고 $\|\mathrm{x}\|_1 = 1$인 $G$의 고유벡터를 $G$의 Perron 벡터라 한다.

위의 건설 과정에서 우리가 건설한 구글행렬이 수렴하는 유일한 페이지랭크 벡터를 갖는다는 것을 보인 셈이다.

이제 수렴성을 분석해보자. 만일 행렬 $P$가 ($G$와 마찬가지로) primitive 행렬이라 하자. 그러면 $\lim_{k\to\infty} P^k$가 존재한다. 특히, $r = \rho(B)$가 이 행렬 $B$의 Perron 근이고, p와 q를 각각 primitive 행렬 $B$와 $B^T$의 Perron 벡터라 할 때 $\lim_{k\to\infty}(B/r)^k = \frac{\mathrm{p}\mathrm{q}^T}{\mathrm{q}^T\mathrm{p}}$임을 알고 있다(Meyer, 2000). 이를 이용하면 x가 $P$의 (right) Perron 벡터라 하고, $\mathrm{e}/n$를 $P^T$의 Perron 벡터라 하면, $\lim_{k\to\infty} P^k = \frac{(\mathrm{e}/n)\mathrm{x}^T}{\mathrm{x}^T(\mathrm{e}/n)} = \frac{(\mathrm{e})\mathrm{x}^T}{\mathrm{x}^T(\mathrm{e})} = \mathrm{e}\mathrm{x}^T > 0$를 얻는다. 벡터 e는 모든 성분을 1로 갖는 열벡터이다.

이것으로부터 $P\mathrm{x} = \mathrm{x}$, 즉 x가 $P$의 (right) Perron 벡터임이 확인되고, 따라서 $\mathrm{x}^T P^T = \mathrm{x}^T$가 되어 $\mathrm{x}^T$는 $P^T$의 (left) Perron 벡터임이 확인된다. 따라서 $\|\mathrm{x}_0\|_1 = 1$인 임의의 초기벡터 $\mathrm{x}_0$에 대하여 $P^k\mathrm{x}_0 = \mathrm{x}_k$이며, primitive 행렬 $P$에 대하여 $\lim_{k\to\infty} P^k = \mathrm{e}\mathrm{x}^T$이고 $\lim_{k\to\infty} \mathrm{p}_k = \mathrm{x}$이다. $P\mathrm{x} = 1\mathrm{x}$인 유일한 단위 벡터 x가 존재한다는 의미이다.

수렴성을 보여주는 위의 설명은 구글행렬 $G$가 벡터 x를 (right) Perron 벡터로 가지는 것과 그에 대응하는 대수적 중복도가 1인 고유값 $\lambda = \rho(G) = 1$을 가진다는 것을 확인해 준다. 그리고 이 알고리즘을 이용하면 반복연산을 통하여 $\mathrm{x}_{k+1} \approx G\mathrm{x}_k$를 만족하는 근삿값을 페이지랭크 벡터 x로 언제나 구할 수 있다는 것이다. 이 x가 구글행렬 $G$의 Perron root라 하고, $G\mathrm{x} = \lambda\mathrm{x}$인 양의 고유벡터 $\mathrm{x} > 0$가 존재한다. 그리고 $G\mathrm{x} = \lambda\mathrm{x}$이면서 $\mathrm{x} > 0$이고 $\|\mathrm{x}\|_1 = 1$인 행렬 $G$의 고유벡터는 행렬 $G$의 페이지랭크 벡터이고, Perron 벡터이다. 따라서 행렬 $G$의 페이지랭크 벡터를 찾는 것은 행렬 $G$의 (dominant right) 고유벡터 또는 $G^T$의 (dominant left) 고유벡터, 즉 Perron 벡터를 찾는 것과 동치이다.

이제 구글행렬 $G$의 페이지랭크 벡터를 구하는 문제를 생각하자. 구글은 수십억개의 홈페이지를 다루므로, 실제로 그런 크기의 행렬 $G$를 만들어 Perron 벡터를 구하는 것은 현실과 다르다. 인터넷상의 웹페이지는 적어도 80억 이상의 개수를 가지므로 간단하게 계산되어질 수 없다. 따라서 실제로는 페이지랭크 벡터를 아래에 설명하는 거듭제곱법을 이용하여 구한다(Langville & Meyer, 2004).

**5단계** : 거듭제곱법

$$
\begin{aligned}
\mathrm{x}_k &= G\mathrm{x}_{k-1} \\
&= \left(mS + (1-m)\frac{\mathrm{e}}{n}\mathrm{e}^T\right)\mathrm{x}_{k-1} \\
&= mS\mathrm{x}_{k-1} + (1-m)\frac{\mathrm{e}}{n}\mathrm{e}^T\mathrm{x}_{k-1} \\
&= mS\mathrm{x}_{k-1} + (1-m)\frac{\mathrm{e}}{n}
\end{aligned}
$$

$$= m\left(H + \frac{\mathbf{e}\mathbf{a}^T}{n}\right)\mathbf{x}_{k-1} + (1-m)\frac{\mathbf{e}}{n}$$

$$= mH\mathbf{x}_{k-1} + \frac{\mathbf{e}}{n}\left(m\mathbf{a}^T\mathbf{x}_{k-1} + (1-m)\right)$$

거듭제곱법(5장1절, 이상구, 2009)을 이용한 수렴속도는 $\tau$를 $\mathbf{x}_{(k)} - \mathbf{x}_{(k-1)}$라고 할 때 $\frac{\log_{10}\tau}{\log_{10}m}$ 임과 페이지와 브린이 $m = 0.85$로 놓으면 대개 50번에서 100번의 연산을 통하여 원하는 페이지랭크 벡터를 구할 수 있음을 보였다(Langville & Meyer, 2003).

이제 어떻게 페이지랭크 벡터가 계산되는지를 확인해보자. 첫째 웹페이지 $i$의 페이지랭크는 페이지랭크 벡터 $\mathbf{x}$의 $i$번째 성분인 값 $x_i$이다. 여기서 벡터 $\mathbf{x}$는 $\|\mathbf{x}\|_1 = 1$인 Perron 벡터이다. 이 벡터 $\mathbf{x}$를 통해 웹상에서의 페이지 $i$의 순위를 가늠해볼 수 있는 점수를 구한다.

위의 전 과정을 요약하면 다음과 같다. 웹의 연결성으로부터 얻은 행렬 $A$의 각 성분을 $H_j = \frac{A_j}{\sum_{k=1}^{n} A_{kj}}$를 이용하여 정규화한 행렬 $H$를 만들게 된다. 그리고 행렬 $H$로부터 열 stochastic 행렬인 $S$를 $S = H + \frac{\mathbf{e}\mathbf{a}^T}{n}$로 정의하고, 이때 벡터 $\mathbf{e}$는 모든 성분이 1인 열(column)벡터이고, 벡터 $\mathbf{a}$는 $\sum_{i=1}^{n} H_{ij} = 0$이면 $a_j = 1$이고 $\sum_{i=1}^{n} H_{ij} \neq 0$이면 $a_j = 0$인 열벡터이다. 이로부터 여기서 $m$은 $0 \leq m \leq 1$의 값인 0.85로 하고 $E = \frac{\mathbf{e}\mathbf{e}^T}{n}$로 정의하여 구글행렬 $G = mS + (1-m)E$를 얻은 것이다. $m$을 0.85로 선택한 것은 임의로 한 것이 아니다. 이와 관련된 자세한 내용은 (Haveliwala & Kamvar, 2003)에 소개되었다.

이와 같이 문제 표현형식을 바꾸면, 웹페이지의 순위를 매기기 위한 문제가 정사각행렬에서 고유벡터를 찾는 우리가 잘 아는 문제로 바뀌게 된다.(정의에 의해 행렬 $G$의 고유값 $\lambda$와 고유벡터 $\mathbf{x}$는 방정식 $G\mathbf{x} = \lambda\mathbf{x}$를 만족한다.) 위 행렬 $G$를 구글행렬이라 한다. 이제는 열 stochastic 행렬 $G$의 가장 큰 고유값 1에 대응하는 고유벡터 $\mathbf{x}$를 구하는 일이 남는다.

위의 전 과정을 예를 들어 설명하면 다음과 같다. 미리 언급하였듯이 구글에서는 $m = 0.85$을 사용한다. 그래서 구글에서 사용하는 $m = 0.85$의 값을 사용하여 얻은 식 (4)의 행렬 $S$를 이용하여 행렬 $G$를 구하면 아래와 같다.

$$G = \begin{bmatrix} \frac{3}{100} & \frac{22}{25} & \frac{91}{200} & \frac{47}{150} & \frac{1}{5} \\ \frac{91}{200} & \frac{3}{100} & \frac{3}{100} & \frac{47}{150} & \frac{1}{5} \\ \frac{3}{100} & \frac{3}{100} & \frac{3}{100} & \frac{47}{150} & \frac{1}{5} \\ \frac{91}{200} & \frac{3}{100} & \frac{3}{100} & \frac{3}{100} & \frac{1}{5} \\ \frac{3}{100} & \frac{3}{100} & \frac{91}{200} & \frac{3}{100} & \frac{1}{5} \end{bmatrix}$$

페이지랭크가 담고 있는 내용을 정리하면 웹페이지 $i$의 페이지랭크는 $\|\mathbf{x}\|_1 = 1$인 Perron 벡터 $\mathbf{x}$의 $i$번째 $x_i$성분의 값이고 페이지랭크 벡터는 각 웹페이지에 접근하게 될 가능성을 반영하는 Probability 벡터이다(Langville & Meyer, 2003).

우리가 사용하고 있는 인터넷의 모든 사이트(웹페이지)가 $n$개의 웹페이지로 만들어진 웹이라 하자. 그러면 인터넷 사용자들은 어떠한 웹페이지를 시작페이지로 설정하여 인터넷을 이용하게 될 것이다.

인터넷을 이용하던 중에 $r$개의 링크를 가지고 있는 어느 특정 웹페이지를 열어보게 된다면 그 웹페이지에서 $\frac{m}{r}$의 확률(구글에서는 $m$을 0.85를 사용하였다.)을 가지고 웹페이지가 가지고 있는 임의의 링크를 선택하게 된다. 웹페이지에 있는 링크를 선택하지 않을 확률은 $\frac{1-m}{n}$이 되고 인터넷 사용자가 선택할 수 있는 모든 경우의 수는 두 확률의 합으로 계산이 된다. 즉

$$r\frac{m}{r}+n\frac{(1-m)}{n}=1$$

(여기서 $r$은 웹페이지가 가진 링크의 개수, $n$은 인터넷 웹상의 모든 웹페이지의 개수)이 된다.

구글의 페이지랭크 알고리즘에서 기본 전제로 한 '중요한 웹페이지는 다른 웹페이지로부터 더 많이 연결되어 있다.'에 따라 인터넷 사용자는 자주 페이지랭크 점수가 높은 웹페이지로 연결되는 링크를 더 자주 선택하게 되고 페이지랭크 점수가 높은 페이지에 많은 인터넷 사용자가 접근을 하게 된다.

접근을 하는 횟수가 다른 웹페이지에 비하여 많다는 것은 바꿔 생각하면 그만큼 인터넷 사용자들이 해당 웹페이지에 머물게 된다는 것이다. 즉 식 $\mathbf{x}= G\mathbf{x}$에서 정규화된 벡터 $\mathbf{x}$의 성분 $x_i$는 인터넷 사용자들이 웹페이지 $i$에서 머무르는 짧은 시간을 의미하는 것으로 판단 생각할 수 있게 된다. 링크구조를 데이터베이스로 가지는 검색엔진의 설계에서 링크구조는 고정적이다. 고정된 링크구조를 가지는 웹에서는 한번 정의된 $\mathbf{x}$는 계속해서 사용할 수 있다. 하지만 인터넷에서의 웹페이지는 단 하나의 구조로 고정되어 있지 않는다. 수많은 사이트와 웹페이지는 생성되었다가 사라지기도 하고 그들 사이를 잇는 링크들 또한 필요성에 따라 계속해서 생겨나고 사라지는 동적인 공간이기 때문에 웹페이지의 페이지랭크 계산을 통한 고유벡터 $\mathbf{x}$의 개선은 매우 중요한 문제가 된다. 페이지랭크 기술은 키워드별로 인터넷 사용자가 클릭하는 웹페이지의 경로를 알고리즘화해 웹페이지들 간의 상호 관련성을 계산해내고, 웹페이지 간의 관련성과 웹페이지 내의 관련어 배치 등을 고려해 고객이 원하는 결과를 가장 빠르고 정확하게 제공하는 기술이다. 구글 검색엔진의 경우 어떤 내용을 검색창에 넣어도 0.5초면 자신이 찾는 가장 근접한 검색결과를 제시해준다. 이 기능은 아직 어떤 검색엔진도 따라올 수 없는 독보적 기술로 평가받고 있다.

현재 구글은 다양한 분야로 사업을 확장하고 있다. 수학자는 '구글 북스 라이브러리 프로젝트'[7]에 대하여도 관심을 가져야 한다. 구글의 목표 중 하나가 20세기에 발간된 모든 책을 전자화한다는 것

7) http://en.wikipedia.org/wiki/Google_Books_Library_Project

이라고 들었다. 구글은 이미 일본 게이오 대학과 미국 하버드 대학 등 세계 40곳 이상의 도서관 및 3만 곳 이상의 출판사와 연계해 지금까지 1500만 권 이상을 전자화했다. 2011년 3월에는 영국의 대영도서관 소장 장서 25만 권을 전자화하기로 합의했다고 발표했다. 이런 변화가 수학을 하는 우리에게 조만간 또 다른 큰 영향을 미칠 것이라는 것을 느낄 수 있다.

## 3. 결론

인류 역사의 혁명적 발전 시기마다 그 사회가 안고 있던 문제를 해결하는 과정의 중심에 수학이 있었다. 우리나라의 학생들은 보통 수학공부를 하는 이유를 '입학시험을 잘 보기 위해서'라고 생각하고 있다. 따라서 수학은 우리의 삶에 불필요한 과목이라고 생각하기도 한다. 이 절에서는 구글 검색엔진 속의 존재하는 선형대수학 이론의 일부를 간략하게 소개했다. 이는 수학의 발전이 인류 사회의 발전 역사와 함께한다는 것을 보여주는 좋은 예가 될 수 있다.

– 저자의 대한수학회 Newsletter 2011년 9월호 원고 –
http://matrix.skku.ac.kr/2009/2009-MathModeling/lectures/mathmodel-week11.pdf

## 4. 참고문헌


- 이상구(2009). 현대선형대수학 3판, 서울 : 경문사
- Avrachenkov K. & Litvak N. (2005). The Effect of New Links on Google PageRank http://wwwhome.math.utwente.nl/~litvakn/StochModels06.pdf
- Brin S.; Page L.; Motwami R. & Winograd T. (1998). The PageRank Citation Ranking: Bringing Order to the Web, Technical report, Computer Science Department, Stanford University, Stanford, CA
- Haveliwala T. H. & Kamvar S. D. (2003). The Second Eigenvalue of the Google Matrix, Stanford University Technical Report http://www.stanford.edu/~sdkamvar/papers/secondeigenvalue.pdf
- Horn R. & Johnson C.R. (1985). Matrix Analysis, Cambridge, 1985 ISBN 0521305861 Langville A. N. & Meyer C. D. (2003). Fiddling with PageRank http://meyer.math.ncsu.edu/Meyer/PS_Files/FiddlingPageRank.pdf
- Langville A. N. & Meyer C. D. (2004). The Use of the Linear Algebra by Web Search Engines http://meyer.math.ncsu.edu/Meyer/PS_Files/IMAGE.pdf
- Langville A. N. & Meyer C. D. (2005). Deeper inside PageRank, Internet Math.
- Meyer C. D. (2000). Matrix Analysis and Applied Linear Algebra, SIAM, Philadelphia.

## Google Matrix와 PageRank™

$$G = \alpha S + (1-\alpha)1\,v^T$$

Row Stochastic Matrix "Google Matrix"

Hyperlink행렬로부터 구해진 Stochastic Matrix

검색엔진이 보유한 자체산출된 중요도

$$\pi^T G = \pi^T,\ \pi > 0,\ \|\pi\|_1 = 1$$

PageRank Vector (Peron Vector)

Row Stochastic Matrix "Google Matrix"

성균관대학교



구글 페이지랭크