Acta Epsilonica
Online First, Pages 1–15.
Received: December 3rd, 2015, Accepted: January 25th, 2016.

Google PageRank の数理

田中未来 *

概要

本稿では Google の検索エンジンのランキング手法—Google PageRank の原理を解説する. 検索エンジンのランキングシステムは検索ワードにマッチするウェブページを重要なものから順に並べるための方法である. なかでも Google PageRank はウェブを Markov 連鎖をモデル化し, その定常状態を計算することでランキングを計算する. この計算は本質的には大規模行列の固有値計算であり, 線形代数の重要な応用例の 1 つともいえる. 本稿ではこれらを理解するために必要な数学的な道具を紹介しつつ, その原理を解説する.

キーワード: 検索エンジン, 数理モデル, 線形代数, 数値計算, Markov 連鎖.

1 はじめに

現在の情報化社会においてウェブ検索エンジンは必要不可欠なものとなっている. しかしながら, この技術がどのように実現されているかを知る人は多くない. 実は, その内部では線形代数などの数学的な道具が使われている. 本稿では Google の検索エンジンの内部で使われている Google PageRank の原理をできるだけわかりやすく解説する. また, そのために必要な数学的知識についても必要に応じて紹介する.

なお, 本稿の多くの部分は以下に挙げる文献を参考にしている: MacCormick の本 [1] はアルゴリズムに関する一般向けの読み物で, 第 2 章と第 3 章がウェブ検索エンジンの内部で用いられるアルゴリズムの解説になっている. Langville と Meyer の本 [2] は Google PageRank に関する専門書である. 背景知識として知っておくべき数学の理論についても書かれており, この 1 冊だけで Google PageRank の理論から実装までを知ることができる[1].

* 東京理科大学 理工学部 経営工学科, mirai at rs.tus.ac.jp.

[1] ただし, [2] の翻訳の質は残念ながら今ひとつである. 一般に, 原著の翻訳はわかりにくいことが多く, 英語に慣れてさえいれば原著を読んだほうがわかりやすいことが多い.

図 1　検索ワードとして “線形代数” を入力したときの Google 検索エンジンの出力. 線形代数に関する複数のウェブページが出力されている. これらは, なんらかの意味でよいものから順に並べられている.

本稿の残りの部分は以下のように構成される: 第 2 節では, ウェブ検索エンジンの仕組みについて概説する. 第 3 節では, Google PageRank の計算を行なうために必要な大規模疎行列について説明する. 具体的には大規模疎行列を格納するためのデータ構造や, 身近に現れる大規模疎行列の例, 大規模疎行列に関する数値計算法などに触れる. 第 4 節では, Google PageRank の原理を理解するために必要なウェブの数理モデルについて述べ, そのモデルの解として Google PageRank ベクトルを定義する. さらにその数値計算法についても解説する. 第 5 節には読者の理解を深めるための演習問題を載せる.

2　ウェブ検索エンジンの仕組み

ウェブ検索エンジンとは, 利用者が検索ワードを入力すると, ウェブ上に数多あるウェブページからそれに関連するものを抽出し, なんらかの意味でよいものから順に出力するシステムのことである. よく用いられる検索エンジンに Yahoo! や Google (図 1) などがある. 他にも, 歴史的に重要な AltaVista, Google とは異なるランキングアルゴリズムを用いている Ask.com, 近年では珍しくユーザの情報を記録しない DuckDuckGo などさまざまな検索エンジンがある.

ウェブ検索エンジンはマッチングとランキングの 2 段階からなる (図 2). マッチングとは, 検索

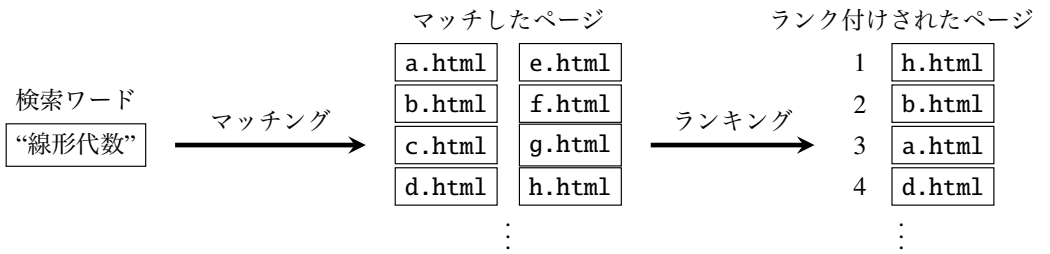

図 2　検索エンジンの仕組みを表す概念図.

ワードと関連するウェブページを抽出することを意味し, ランキングとは, マッチするウェブページをなんらかの意味でよいものから順に並べることを意味する. 今回はこれらのうちランキングに焦点を当てる[2]. ランキングにはさまざまな手法が知られているが, 多くのものの内部では, 線形代数を筆頭にグラフ理論や確率論などの数学的な道具が用いられている. Google PageRank はそのようなランキング手法の 1 つである. この技術が当時 Stanford 大学の大学院生に過ぎなかった Page[3] と Brin によって開発されたことからもわかるように, その基本原理は決して難しいものではない. しかしながら, 巨大なウェブを線形代数の道具—行列を用いて表現するためには, 非常に大規模な行列を扱う必要がある. 以下では大規模行列の効率のよい扱い方について考える.

3　大規模疎行列

現実の世界に現れる行列は非常に大規模である. そのため, うまく扱わなければ非常に効率の悪い計算を行なうことになったり, そもそも計算機で扱うことができなかったりする. このことを理解するために, 大規模行列の情報量について考えてみよう. 単純化のため, 正方行列について考えることとし, そのサイズを n としよう. この行列の各成分を 8 [byte] の倍精度浮動小数点型 (`double` 型) でメモリに格納するためには $8n^2$ [byte] が必要となる. これは $n = 10^4$ のとき 0.8 [GB] であり, 普通の計算機であればメモリの大部分を消費することになる. さらに $n = 10^5$ のとき 80 [GB] となり, 普通の計算機ではメモリに格納することすらできないことがわかる.

一方で, 現実の世界に現れる行列は疎であるという特徴をもつ. ここである行列が疎であるとは, その行列のほとんどの要素が 0 であることを意味する. 図 3 はある微分方程式を解くときに出てくる行列を図示したものである. 行列の要素が 0 である部分は白く, 非ゼロ要素は黒く塗られている. 非対角成分はほとんどが 0 であり, この行列が疎であることが見てわかる. このような行列を計算機で扱う際に, 非ゼロ要素だけをメモリに格納すればメモリ消費量を削減できることは明らかだろう. そのような方法は複数あるが, ここでは最も簡単な座標リスト方式を扱う.

座標リスト方式は疎行列を格納するための最も簡単な方法で, 非ゼロ要素 A_{ij} の座標 (i, j) と値 A_{ij} を (特に工夫せず) 格納する方法である. 例として, 次のような行列 $\boldsymbol{A}$ を座標リスト方式を用

[2] マッチングについては MacCormick の本 [1] の第 2 章などを参照されたい.

[3] PageRank という名前はウェブページの “page” と 開発者の名前の “Page” をかけたものになっている.

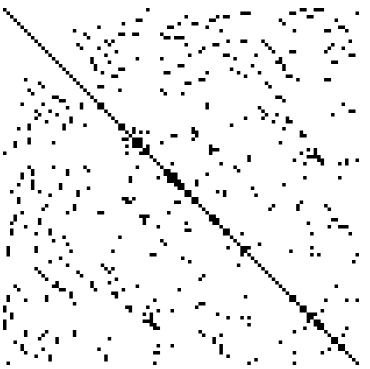

図 3　ある微分方程式を解くときに出てくる疎行列.

いて格納する方法を考えよう:

$$\boldsymbol{A} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 1.0 & 0 & -1.0 & 0 \\ 0 & 0 & 0 & 1.5 \\ 2.0 & -0.5 & 0 & 0 \end{pmatrix} \end{array}.$$

この行列の非ゼロ要素は $A_{11} = 1.0, A_{13} = -1.0, A_{24} = 1.5, A_{31} = 2.0, A_{32} = -0.5$ の 5 個である. これらの座標および値を (i, j) についての昇順でまとめると次のようになる:

$$\begin{array}{c} i \\ j \\ A_{ij} \end{array} \begin{bmatrix} 1 & 1 & 2 & 3 & 3 \\ 1 & 3 & 4 & 1 & 2 \\ 1.0 & -1.0 & 1.5 & 2.0 & -0.5 \end{bmatrix}.$$

これが行列 $\boldsymbol{A}$ の座標リスト方式による表現である.

3.1　有向グラフと隣接行列

現実の世界に現れる大規模疎行列の例に (有向) グラフの隣接行列がある. ここでいうグラフとは頂点とそれらを結ぶ枝からなる離散構造のことである. 特に, 枝の向きを考えるものを有向グラフ, そうでないグラフを無向グラフという. ここでは有向グラフを扱うが, 正確に定義することはせず, 例を用いて説明を行なう.

図 4 に有向グラフの例を示す. 有向グラフは図 4 のように “丸” が “矢印” でつながったものと考えてよい. ここで, “丸” は頂点と呼ばれ, “矢印” は枝と呼ばれる. なお, 頂点は節点やノードなどと呼ばれることもあり, 枝は辺や弧などと呼ばれることもある. 特に, 有向グラフでは枝の向きを考えることを明確にするために有向枝と呼ぶこともある. 図 4 の有向グラフにおいて, 頂点は $1,2,3,4,5$ の 5 個であり, 枝は $(1,2),(1,4),(2,3),(3,2),(3,5),(4,1),(4,5)$ の 7 本である.

グラフを用いると現実のさまざまなものごとを表現できる. 友人関係や Twitter のフォロー関係, そしてウェブページのハイパーリンク関係などは有向グラフを用いて表現できる. 例えば, 図 4 の有向グラフは人 $1,2,3,4,5$ の友人関係を表すものとみなすこともできる. 頂点はそれぞれの人に対応し, 枝 (i, j) は人 i が人 j を友人だと思っていることに対応する.

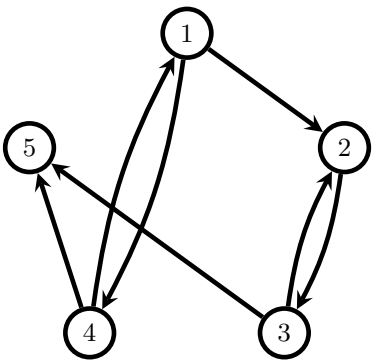

図 4　有向グラフの例.

グラフを計算機上で表現するためのデータ構造の 1 つに隣接行列がある. あるグラフの頂点数が n のとき, その隣接行列は n 次正方行列で, それぞれの成分は

$$A_{ij} = \begin{cases} 1 & (\text{枝 } (i,j) \text{ があるとき}), \\ 0 & (\text{枝 } (i,j) \text{ がないとき}) \end{cases}$$

のように定義される. 図 4 の有向グラフの隣接行列は次のようになる:

$$\boldsymbol{A} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}.$$

このように, 隣接行列は対応するグラフの頂点同士のつながりを表現するものである.

現実に現れるグラフは, 頂点数 n が大きく, すべての頂点対に枝がある場合[4] の枝の数 $n(n-1)/2$ に比べて枝の数が小さいことが多い. このようなグラフのことを疎なグラフという. 疎なグラフに対応する隣接行列は疎行列になる.

3.2　行列の疎性を利用した線形計算

ベクトル $\boldsymbol{x} \in \mathbb{R}^n$ と行列 $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ が与えられたときに, それらの積 $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ を計算することを考える. もちろん, 定義よりその第 i 要素は

$$y_i = \sum_{j=1}^{n} A_{ij} x_j$$

と書ける. さて, $\boldsymbol{A}$ が大規模疎行列のとき, どのように $\boldsymbol{y}$ を計算すると効率がよいだろうか.

[4] このようなグラフのことを完全グラフと呼ぶ.

最も単純な方法は行列 $\boldsymbol{A}$ を 2 次元配列に変換してから計算する方法である. この方法の擬似コード[5] をアルゴリズム 1 に示す. ここで擬似コード中の記号 := は代入を意味する. このアルゴリズムは非効率的である. その理由は, ほとんどの i, j に対して $A_{ij} = 0$ であるにもかかわらず, $A_{ij} \cdot x_j$ という計算を大量に実行しているためである. なお, n が非常に大きな場合は 2 次元配列 $\boldsymbol{A}$ のメモリを確保することすらできないだろう.

アルゴリズム 1 疎行列 $\boldsymbol{A}$ とベクトル $\boldsymbol{x}$ の積を計算する効率の悪いアルゴリズムの擬似コード

入力: 疎行列 $\boldsymbol{A}$ とベクトル $\boldsymbol{x}$
出力: 疎行列 $\boldsymbol{A}$ とベクトル $\boldsymbol{x}$ の積 $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$
疎行列 $\boldsymbol{A}$ を 2 次元配列で保持するためのメモリを確保
疎行列 $\boldsymbol{A}$ を 座標リスト方式から 2 次元配列に変換
$\boldsymbol{y} := \boldsymbol{0}$
for $i = 1, \ldots, m$ **do**
    for $j = 1, \ldots, n$ **do**
        $y_i := y_j + A_{ij} x_j$
    end for
end for

そこで, 非ゼロ要素だけを取り出して計算する方法を考える. いま, 行列 $\boldsymbol{A}$ の座標リスト方式が以下のように書かれているものとする:

$$\begin{array}{c} i \\ j \\ A_{ij} \end{array} \begin{bmatrix} \cdots & i_0 & i_0 & \cdots & i_0 & \cdots \\ \cdots & j_1 & j_2 & \cdots & j_K & \cdots \\ \cdots & A_{i_0 j_1} & A_{i_0 j_2} & \cdots & A_{i_0 j_K} & \cdots \end{bmatrix}.$$

このとき, $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ の第 i_0 要素の計算は

$$y_{i_0} = \sum_{j=1}^{n} A_{i_0 j} x_j = \sum_{k=1}^{K} A_{i_0 j_k} x_{j_k}$$

のように書き換えることができる. このように $A_{i_0 j} = 0$ に対応する要素の計算を省略することで, 無駄な $0 \cdot x_j$ の計算を回避することができる.

このように疎行列とベクトルの積を効率よく計算することにより疎行列の固有値の計算を効率よく行なうことができる. 以下では固有値の数値計算法について述べる.

[5] アルゴリズムの枠組みを表現するためによく用いられる非常に高級なプログラミング言語を模した表現方法のこと. 大昔はそのような用途でフローチャートが用いられていたが, フローチャートを用いてプログラムを実装すると `goto` 文を多用したスパゲティコードになりやすいという欠点がある. そのため, 近年では擬似コードのほうがよく用いられている.

3.3 優越固有値の数値計算法: べき乗法

行列の固有値が線形代数の理論において重要であることは言うまでもないが, 線形常微分方程式の解法や主成分分析などでも目にするように工学的にも重要な値である. これらのことを実際に行おうとするとき, 与えられた行列の固有値を計算する必要がある.

私たちが手計算で正方行列 $\boldsymbol{A}$ の固有値 λ を求めるとき, 普通は固有方程式 $\det(\boldsymbol{A} - \lambda \boldsymbol{I}) = 0$ を解く. しかしながら, 計算機を用いて固有値を計算するときは異なる方法が用いられる. 固有値の計算に用いられる数値計算法にはべき乗法や QR 法などがあるが, ここではべき乗法を扱う. べき乗法とは, 正方行列 $\boldsymbol{A}$ が与えられたとき, ある初期ベクトル $\boldsymbol{x}_0$ に $\boldsymbol{A}$ のべき乗を掛けて生成されるベクトル列 $\boldsymbol{x}_0, \boldsymbol{A}\boldsymbol{x}_0, \boldsymbol{A}^2\boldsymbol{x}_0, \ldots$ を利用して, 優越固有値[6] とそれに対応する固有ベクトルを求めるアルゴリズムである.

べき乗法が拠り所とする理論的背景は以下の通りである: n 次正方行列 $\boldsymbol{A}$ が n 個の線形独立な固有ベクトル $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n$ をもつことを仮定する. また, それぞれの固有ベクトルに対応する固有値を $\lambda_1, \lambda_2, \ldots, \lambda_n$ とし, $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|$ であることを仮定する. この仮定の下で λ_1 はただ 1 つの優越固有値である. このとき, 初期ベクトル $\boldsymbol{x}_0$ は $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n$ の線形独立性より $\boldsymbol{x}_0 = c_1\boldsymbol{v}_1 + c_2\boldsymbol{v}_2 + \cdots + c_n\boldsymbol{v}_n$ と展開できる. ただし, 以下では $c_1 \neq 0$ を仮定する[7]. 固有値の定義より, $\boldsymbol{A}\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i$ $(i = 1, 2, \ldots, n)$ となるので,

$$\begin{aligned} \boldsymbol{A}^k\boldsymbol{x}_0 &= c_1\boldsymbol{A}^k\boldsymbol{v}_1 + c_2\boldsymbol{A}^k\boldsymbol{v}_2 + \cdots + c_n\boldsymbol{A}^k\boldsymbol{v}_n \\ &= c_1\lambda_1^k\boldsymbol{v}_1 + c_2\lambda_2^k\boldsymbol{v}_2 + \cdots + c_n\lambda_n^k\boldsymbol{v}_n \end{aligned}$$

となる. ここで, 優越固有値 λ_1 をくくりだすと,

$$\boldsymbol{A}^k\boldsymbol{x}_0 = \lambda_1^k \left(c_1\boldsymbol{v}_1 + c_2 \left(\frac{\lambda_2}{\lambda_1}\right)^k \boldsymbol{v}_2 + \cdots + c_n \left(\frac{\lambda_n}{\lambda_1}\right)^k \boldsymbol{v}_n \right)$$

となる. いま, $k \longrightarrow \infty$ のときに $(\lambda_i/\lambda_1)^k \longrightarrow 0$ $(i \neq 1)$ となることに着目すると, 十分大きな k について,

$$\boldsymbol{A}^k\boldsymbol{x}_0 \simeq c_1\lambda_1^k\boldsymbol{v}_1$$

と近似できることがわかる. すなわち, $\boldsymbol{A}^k\boldsymbol{x}_0$ は $k \longrightarrow \infty$ のとき, 優越固有値に対応する固有ベクトルの方向に近づく.

以上の議論から, 適当な初期ベクトル $\boldsymbol{x}_0$ に正方行列 $\boldsymbol{A}$ を繰り返し掛けていくことで, $\boldsymbol{A}$ の優越固有値に対応する固有ベクトルを近似的に求めることができることがわかる. ただし, 単純に $\boldsymbol{A}^k\boldsymbol{x}_0$ を計算するだけでは $|\lambda_1| > 1$ のときに発散するなど都合が悪い. そのため, 実装する際には各反復でベクトルをなんらかの意味で正規化するとよい. 例えば, $\|\boldsymbol{x}\| = 1$ を満たす固有ベクトルを計算したい場合はアルゴリズム 2 のような計算をすればよい. 十分大きな k について $\boldsymbol{x}_k$ は $\boldsymbol{v}_1$ のよい近似となる. このアルゴリズムに必要な演算は,

---

[6] 絶対値が最大の固有値のこと.

[7] この仮定は $\boldsymbol{x}_0$ の各成分を標準正規乱数を用いて生成することにより高い確率で達成できる.

- 行列とベクトルの積,
- ベクトルのノルム,
- ベクトルのスカラー倍,
- ベクトルの内積

であることに着目しよう. $\boldsymbol{A}$ が疎行列の場合は行列とベクトルの積が高速に計算できるため, このアルゴリズムは高速に実行できる.

アルゴリズム 2 行列 $\boldsymbol{A}$ の優越固有値 λ と対応する固有ベクトル $\boldsymbol{x}$ を求めるべき乗法の擬似コード

入力: 行列 $\boldsymbol{A}$
出力: 行列 $\boldsymbol{A}$ の優越固有値 λ と対応する固有ベクトル $\boldsymbol{x}$
$\boldsymbol{x}$ を標準正規乱数を用いて生成する
while 適当な終了条件 **do**
    $\boldsymbol{y} := \boldsymbol{A}\boldsymbol{x}$
    $\boldsymbol{x} := \boldsymbol{y}/\|\boldsymbol{y}\|$
    $\lambda := \boldsymbol{x}^\top \boldsymbol{y}$
end while

4 ウェブの数理モデルと Google PageRank

現実のさまざまな問題に対する解決策を導くための方法の 1 つに数理モデルを用いた解析がある. モデル[8] とは, 現実のものごとを単純化したもののことをいう. 現実の問題に対応する数理モデルは 1 つとは限らない. 単純なモデルは解析がしやすい一方で, 現実をよく反映していないことがある. 一方で, 複雑なモデルは現実をよく反映しているが, 解析は困難である. このように, 適切なモデルの構築[9] には現実の問題に対する洞察や経験に加えて, さまざまなモデルの解析のしやすさに関する数学的知識などが必要となる.

力学の例を用いてこれを説明しよう. 物体をある高さまで持ち上げて手を離すと運動方程式 $m\mathrm{d}^2x/\mathrm{d}t^2 = -mg$ に従って自由落下する—これは数理モデルの例になっている. 実際, この運動方程式は物体が本来うける空気抵抗などの重力以外の力を無視する単純化に基づく. この単純化を行なうことで, われわれは運動方程式を簡単に解くことができる. 鉄球のように物体の質量が大きく, 空気抵抗が重力に比べて小さければ, それを無視して解析を行なっても現実との間に大きな違いは出ない. むしろ, 解析が簡単であるメリットのほうが大きい. 一方で, 羽根のように空気抵抗が大きい物体の場合は空気抵抗を無視したモデルは適さない. しかしながら, 空気抵抗を考慮した自由落下の運動方程式のような複雑なモデルは解くことが難しい.

[8] 対応する漢語は模型である.
[9] これをモデル化という.

4.1 ウェブのモデル化

検索エンジンにおけるランキングの目的は, 検索ワードにマッチしたウェブページをなんらかの意味でよいものから順に並べることだった (図 2). さて, どのようなウェブページが “よい” のだろうか. 以下ではウェブページの “よさ” を定量化するためにウェブをモデル化する.

以下では, ウェブページ間のハイパーリンクの意味がネガティブなものに比べてポジティブなもののほうが多いということを仮定する. これはそれなりに現実的な仮定であることが知られている. この仮定の下, 以下のようなウェブページを “よい” ウェブページとする:

- 多くのページからリンクされているウェブページ,
- “よい” ウェブページからリンクされているウェブページ.

このようなウェブページを発見するために以下ではネットサーフィンのモデル化を行なう.

4.2 ネットサーフィンのモデル化

ネットサーフィンをする際, ユーザはあるウェブページを訪問し, そのウェブページを適当な時間だけ閲覧し, 別のウェブページを訪問するという行動を続ける. ここでは解析を簡単にするため時間を離散化し, 各ユーザは各時点で以下の行動のいずれかをとるものとする:

ハイパーリンク移動 別のページへのハイパーリンクが存在するページではハイパーリンクをたどって別のページに移動する. そうでないページではテレポーテーション移動を行なう.

テレポーテーション移動 ランダムに別のページに移動する. これはブックマークの利用や URL の直接入力などに対応している.

すべてのユーザがこのように行動すると, 十分な時間が経過した後で “よい” ウェブページに多くのユーザが集まることが予想される. つまり, 多くのページからリンクされているウェブページは, その分だけ流入してくるユーザが多いことが期待できる. いわば “塵も積もれば山となる” の理屈である. また, “よい” ウェブページからリンクされているウェブページは, その “よい” ウェブページから多くのユーザが遷移してくることが期待できる. こちらは “親の光は七光り” である. 以下ではこれらの詳細を述べる.

なお, 別のページへのハイパーリンクが存在しないページは, 対応する有向グラフにおける頂点のうちその頂点から出て行く枝が存在しないものに対応する. このような頂点のことを終点という. 例えば, 図 4 の有向グラフでは頂点 5 は終点であり, それ以外の頂点は終点ではない.

4.2.1 ハイパーリンク移動

終点以外の頂点におけるハイパーリンク移動では, ユーザは現在の頂点から出て行く枝を等確率で選択し, 次の時刻に隣接する頂点に移動するものとする. これは, ユーザが現在閲覧しているページに張られているハイパーリンクを等確率でクリックし, 次の時刻にリンク先のページに移動するということに対応する.

例えば, 図 4 の有向グラフにおいて, 頂点 1 から出て行く枝は 2 本あり, それらは頂点 2,4 に接続

している. ある時刻に頂点 1 に対応するウェブページを閲覧しているユーザがハイパーリンク移動を行なうとき, 次の時刻には確率 1/2 で頂点 2 に対応するウェブページを, 確率 1/2 で頂点 4 に対応するウェブページを閲覧する. 一方, 頂点 2 から出て行く枝は 1 本しかなく, その先は頂点 3 である. ある時刻に頂点 2 にいるユーザがハイパーリンク移動を行なう場合, 次の時刻には確率 1 で頂点 3 に移動する.

このような確率をまとめた行列を Google PageRank の文脈ではハイパーリンク行列とよぶ. 具体的には, ハイパーリンク行列 $\boldsymbol{H}$ の各成分は

$$H_{ij} = \begin{cases} \text{上述のモデルで頂点 } i \text{ から次の時刻に頂点 } j \text{ に移動する確率} & \text{(頂点 } i \text{ が終点でないとき)}, \\ 0 & \text{(頂点 } i \text{ が終点であるとき)} \end{cases}$$

と定義される. 図 4 の有向グラフに対応するハイパーリンク行列 $\boldsymbol{H}$ は次のようになる:

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 0 & 1/2 & 0 & 1/2 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1/2 & 0 & 0 & 1/2 \\ 1/2 & 0 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}.$$

一般に, ある有向グラフに対応するハイパーリンク行列 $\boldsymbol{H}$ の各成分は隣接行列 $\boldsymbol{A}$ を用いて次のように書くことができる:

$$H_{ij} = \begin{cases} A_{ij} / \sum_j A_{ij} & \text{(頂点 } i \text{ が終点でないとき)}, \\ 0 & \text{(頂点 } i \text{ が終点であるとき)}. \end{cases}$$

一方, ユーザが終点に対応するウェブページを閲覧しているとき, そのページには別のページへのハイパーリンクが存在しないので, ハイパーリンクをクリックして別のページに移動することはできない. このとき, ユーザは次の時刻ではブックマークを利用したり URL を直接入力したりすることで, ウェブ上の任意のページに等確率で移動するものとする[10].

例えば, 図 4 の有向グラフにおいて頂点 5 は終点である. ある時刻に頂点 5 にいるユーザは, 次の時刻で各頂点に確率 1/5 で移動する.

以上のことをベクトルと行列を用いて表現してみよう. 各頂点が終点であるかどうかを表すベクトル $\boldsymbol{a}$ を次のように定義する:

$$a_i = \begin{cases} 1 & \text{(頂点 } i \text{ が終点)}, \\ 0 & \text{(頂点 } i \text{ が終点でない)}. \end{cases}$$

さらに, このベクトルと要素がすべて 1 のベクトル $\boldsymbol{e} = (1,1,\ldots,1)^\top$ を用いて行列 $\boldsymbol{P} = \boldsymbol{H} + (1/n)\boldsymbol{a}\boldsymbol{e}^\top$ を定義する. すると, その各成分 P_{ij} は各ユーザがある時刻に頂点 i から頂点 j に次の時刻に移動する確率となる. このような行列 $\boldsymbol{P}$ を推移確率行列という. 推移確率行列の必要十分条件は $\boldsymbol{P} \geq \boldsymbol{O}$ (各成分が非負), $\boldsymbol{P}\boldsymbol{e} = \boldsymbol{e}$ (行和が 1) である.

[10] ブラウザの戻るボタンを押して前の時刻で閲覧していたページに戻るという行動はもっともらしいが, ここではモデルの単純化のためそれは考えない.

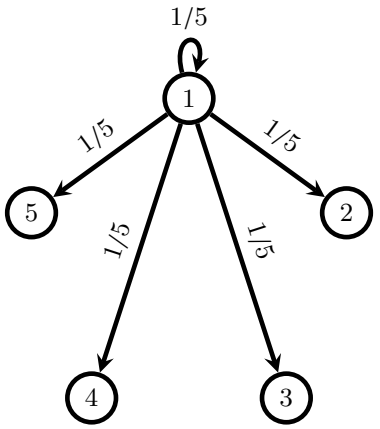

図 5　図 4 の有向グラフ上の頂点 1 からのテレポーテーション移動の例.

例えば, 図 4 の有向グラフのハイパーリンク移動に関する推移確率行列は次のようになる:

$$\boldsymbol{P} = \boldsymbol{H} + \frac{1}{n}\boldsymbol{a}\boldsymbol{e}^{\top} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 0 & 1/2 & 0 & 1/2 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1/2 & 0 & 0 & 1/2 \\ 1/2 & 0 & 0 & 0 & 1/2 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \end{pmatrix} \end{array}.$$

4.2.2　テレポーテーション移動

ユーザは終点以外でもブックマークを利用したり URL を直接入力したりする. これを表現するのがテレポーテーション移動である. 具体的には, ある頂点にいるユーザがテレポーテーション移動を行なうとき, 次の時刻に各頂点に等確率で移動するものとする.

例えば, 図 4 の有向グラフにおいて頂点 1 からテレポーテーション移動を行なうとき, 次の時刻には各頂点に確率 1/5 で移動する (図 5). 他の頂点についても同様である. なお, 終点についてはハイパーリンク移動とテレポーテーション移動が一致する.

テレポーテーション移動を表現する推移確率行列は $(1/n)\boldsymbol{e}\boldsymbol{e}^{\top}$ と書ける. ここで n はグラフの頂点数である.

例えば, 図 4 の有向グラフのテレポーテーション移動に関する推移確率行列は次のようになる:

$$\frac{1}{n}\boldsymbol{e}\boldsymbol{e}^{\top} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \end{pmatrix} \end{array}.$$

4.3 Google PageRank の定義

Google PageRank を数学的に定義しよう. Google PageRank とは, ウェブページのハイパーリンク構造を受け取り, これまでに定義したモデルを用いて各ウェブページに対する PageRank 値を計算し, その値が大きいものから順にウェブページを並び替えたものを返すアルゴリズムである. ここで, あるページの PageRank 値とは, ユーザがハイパーリンク移動とテレポーテーション移動を繰り返し, 十分な時間が経過した後でそのページを閲覧している確率のことである. 以前の洞察から, “よい” ページほどこの値が高くなることが期待できる. 以下ではこれを厳密に定義する.

ユーザは各時刻において確率 α $(0 < \alpha < 1)$ でハイパーリンク移動, 確率 $1 - \alpha$ でテレポーテーション移動を行なうものとする. この行動を表す推移確率行列は次のように書ける:

$$\boldsymbol{G} = \alpha \left(\boldsymbol{H} + \frac{1}{n} \boldsymbol{a} \boldsymbol{e}^\top \right) + \frac{1 - \alpha}{n} \boldsymbol{e} \boldsymbol{e}^\top.$$

この行列 $\boldsymbol{G}$[11] は推移確率行列になっている. すなわち, その各成分 G_{ij} はある時刻にページ i にいるユーザが次の時刻にページ j に移動する確率を表す.

時刻 t においてユーザが頂点 i にいる確率を $p_i^{(t)}$ とし, それらを並べたベクトルを分布ベクトル $\boldsymbol{p}^{(t)}$ と呼ぶ[12]. 分布ベクトルの必要十分条件は $\boldsymbol{p}^{(t)} \geq \boldsymbol{0}$ (各成分が非負), $\boldsymbol{e}^\top \boldsymbol{p}^{(t)} = 1$ (和が 1 に等しい) である. 時刻 $t + 1$ における分布ベクトル $\boldsymbol{p}^{(t+1)}$ は時刻 t における分布ベクトル $\boldsymbol{p}^{(t)}$ と推移確率行列 $\boldsymbol{G}$ を用いて

$$(\boldsymbol{p}^{(t+1)})^\top = (\boldsymbol{p}^{(t)})^\top \boldsymbol{G}$$

と書ける. したがって, ある時刻 t における分布ベクトル $\boldsymbol{p}^{(t)}$ は初期分布ベクトル $\boldsymbol{p}^{(0)}$ を用いて次のように書ける:

$$\boldsymbol{p}^{(t)} = \boldsymbol{G}^\top \boldsymbol{p}^{(t-1)} = (\boldsymbol{G}^\top)^2 \boldsymbol{p}^{(t-2)} = \cdots = (\boldsymbol{G}^\top)^t \boldsymbol{p}^{(0)}.$$

これは $t \longrightarrow \infty$ の極限で初期分布ベクトル $\boldsymbol{p}^{(0)}$ によらない同一の定常分布 $\boldsymbol{p}$ に収束し,

$$\boldsymbol{G}^\top \boldsymbol{p} = \boldsymbol{p} \tag{1}$$

を満たす[13] ことが Perron–Frobenius の定理を用いて証明できる. この $\boldsymbol{p}$ は十分な時間が経過した後にユーザが各ページを閲覧している確率を並べたベクトルとなっている. ページ i を閲覧している確率 p_i をページ i の PageRank 値と呼ぶ.

Perron–Frobenius の定理は, 数値解析, 確率論, 経済学など幅広い分野に応用をもつ. そのため, この定理は非常に重要であり. 応用線形代数における金字塔と言っても過言ではない. ここではその主張のみを紹介しよう:

定理 1（Perron–Frobenius の定理） 既約な非負行列 $\boldsymbol{A}$ は次の性質をもつ正の固有値 ρ をもつ:

[11] なお, この行列は Google 行列と呼ばれることもある.

[12] なお, 通常は転置をした $(\boldsymbol{p}^{(t)})^\top$ を分布ベクトルとすることのほうが多い. しかし, ここでは列ベクトルに統一したほうがわかりやすいと考えて転置したものを分布ベクトルと呼んでいる.

[13] なお, この線形方程式系を PageRank 方程式と呼ぶことがある.

- ρ に対応する固有ベクトルとして要素がすべて正のベクトルがとれる.
- $\boldsymbol{A}$ のその他の固有値の絶対値は ρ を超えない.
- ρ は固有方程式の単根である.

この定理の証明は難しいものではないが, やや面倒である. 応用を志向した線形代数の教科書 (例えば [3, 4]) を参照されたい.

4.4 PageRank 値の数値計算法

以下では PageRank 方程式 (1) の解 $\boldsymbol{p}$ を数値計算により求める方法を考える. 式 (1) は $\boldsymbol{G}^\top$ の固有値 1 に対応する固有方程式とみなすこともできることに着目しよう. 実際, 推移確率行列の性質 $\boldsymbol{G}\boldsymbol{e} = \boldsymbol{e}$ から 1 が $\boldsymbol{G}^\top$ の固有値になっていることがわかる. さらに, Perron–Frobenius の定理を用いると, 1 が $\boldsymbol{G}^\top$ に対する唯一の優越固有値であり, $\boldsymbol{p} > \boldsymbol{0}$ (各成分が正) であることが証明できる. これらの事実から, $\boldsymbol{G}^\top$ に対して任意の初期分布ベクトル $\boldsymbol{p}^{(0)}$ ($\boldsymbol{p}^{(0)} \geq \boldsymbol{0}$ かつ $\boldsymbol{e}^\top\boldsymbol{p}^{(0)} = 1$ を満たす) からべき乗法を適用すると式 (1) の解に収束することを証明できる. また, 任意の t に対して $\boldsymbol{p}^{(t)}$ が分布ベクトルとなることも帰納法から明らかである.

ところで, $\boldsymbol{G}^\top$ に対してべき乗法を適用する際には $\boldsymbol{G}^\top\boldsymbol{p}^{(t)}$ の計算が必要となる. しかしながら, $\boldsymbol{G}^\top$ はその定義から密行列である. さらに, $\boldsymbol{G}^\top$ のサイズは有向グラフの頂点数, すなわちウェブページの数に対応するため, 現実的には $\boldsymbol{G}^\top$ は大規模行列であることが多い. そのため, 計算に多少の工夫が必要となる. いま, $\boldsymbol{e}^\top\boldsymbol{p}^{(t)} = 1$ が成り立つことを利用すると,

$$\boldsymbol{p}^{(t+1)} = \boldsymbol{G}^\top\boldsymbol{p}^{(t)} = \alpha\boldsymbol{H}^\top\boldsymbol{p}^{(t)} + \frac{1}{n}(\alpha\boldsymbol{a}^\top\boldsymbol{p}^{(t)} + 1 - \alpha)\boldsymbol{e}$$

と書ける. この計算に必要な演算は,

- 疎行列とベクトルの積,
- ベクトルの内積,
- ベクトルの和,
- ベクトルのスカラー倍

のみであり, 疎行列とベクトルの積の計算, ベクトル同士の演算, ベクトルとスカラーの積だけで処理できる. なお, 理論的には常に $\boldsymbol{e}^\top\boldsymbol{p}^{(t)} = 1$ が成り立つので正規化を行なう必要はないが, 実際の数値計算では数値誤差の影響で $\boldsymbol{e}^\top\boldsymbol{p}^{(t)} = 1$ が成立しなくなるおそれがあることに注意されたい.

5 演習問題

1. 疎行列を格納するための方法には座標リスト方式の他にも複数の方法が知られている. それらを調べ, 座標リスト方式と比較してどのような長所と短所があるか考えてみよ.
2. 行列を格納する際に 2 次元配列と座標リスト方式のどちらを用いるとメモリ消費量が少なくなるかは, その行列がどのくらい疎であるかによって決まる. n 次正方行列の各非ゼロ成分を `double` 型で, その座標を `long` 型で格納して座標リスト方式とするとき, その行列がどのく

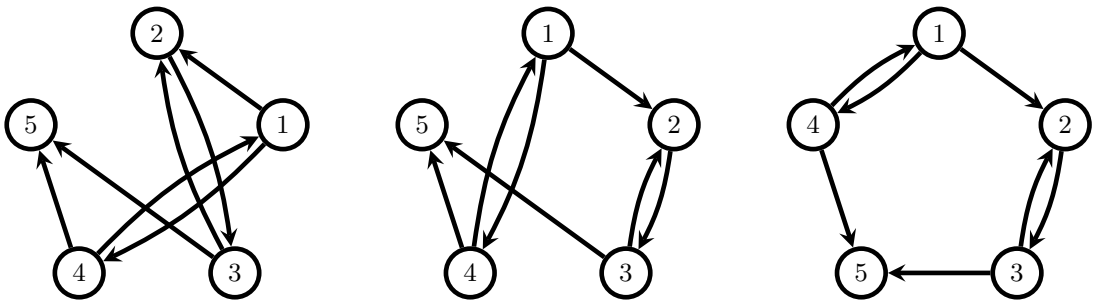

図 6　見やすいグラフと見にくいグラフの例. 中央は図 4 と同じ. 左は頂点 1,2 の位置を入れ替えた見にくいグラフ. 右は頂点 4,5 の位置を入れ替えた見やすいグラフ.

らい疎であれば座標リスト方式のほうがメモリ消費量が少なくなるか考えてみよ.

3. 行列とベクトルの積を計算する際に, 行列を 2 次元配列として扱うときと座標リスト方式として扱うときとでどちらが基本演算[14] の回数が少ないかは $\boldsymbol{A}$ がどのくらい疎であるかによって決まる. 疎行列 $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ の非ゼロ要素数を d として, ベクトル $\boldsymbol{x} \in \mathbb{R}^n$ との積をそれぞれの方法で計算するときの基本演算の回数を比較し, $\boldsymbol{A}$ がどのくらい疎であれば疎性を利用した計算のほうが基本演算の回数が少なくなるか考えてみよ.
4. 疎なグラフは工夫して描くと見やすくなる (図 6 参照). どのような工夫をすればよいか考えてみよ.
5. 各自の好きな言語で Google PageRank システムを実装してみよ. そして, パラメータ α の値を変化させて PageRank 値を計算することで, べき乗法が収束するまでの反復回数や結果に α がどのような影響を及ぼすか調べてみよ. また, Google PageRank の開発者である Page と Brin は $\alpha = 0.85$ を推奨している. なぜこの値が推奨されているか考えてみよ. なお, 計算には実際のウェブのハイパーリンク構造から作られたデータセット[15] を用いるとよい.
6. ウェブ上にはいくつのウェブページがあるか考えてみよ. さらに, それら全体の PageRank 値を計算するためにはどれくらいの計算資源が必要か考えてみよ.
7. 自分がウェブページの管理者であるとして, 自分のウェブページの PageRank 値を上げるためにはどのような工夫をするとよいか考えてみよ.

謝辞

本稿は著者の講義資料を再構成したものです. 原稿をよく読んで間違いを指摘してくれた同僚と学生に感謝します. また, 本稿を書く機会を与えて下さった山下弘一郎氏, 建設的な意見を与えてくれた査読者に感謝します.

[14] 変数への値の代入, 変数へのアクセス, 加減乗除, 比較, 初等関数の呼び出しなど.

[15] 例えば, Data Sets for Link Analysis Ranking Experiments (`http://www.cs.toronto.edu/~tsap/experiments/download/download.html`) など.

参考文献

[1] J. MacCormick: *Nine Algorithms That Changed the Future: The Ingenious Ideas That Drive Today's Computers,* Princeton University Press, 2012. 長尾高弘 (訳): 世界でもっとも強力な 9 のアルゴリズム, 日経 BP 社, 2012.

[2] A. N. Langville and C. D. Meyer: *Google's PageRank and Beyond: The Science of Search Engine Rankings,* Princeton University Press, 2006. 岩野和生, 黒川利明, 黒川洋 (訳): Google PageRank の数理 —最強検索エンジンのランキング手法を求めて—, 共立出版, 2009.

[3] 伊理正夫: 線形代数汎論, 朝倉書店, 2009.

[4] 室田一雄, 杉原正顯: 線形代数 *II,* 丸善出版, 2013.