Szegedi Tudományegyetem
Természettudományi és Informatikai Kar

Bolyai Intézet
Sztochasztika tanszék

A PageRank és kiszámolása

Szakdolgozat

Készítette: Gáspár Tamás
Matematika BSc hallgató

Témavezető: Dr. Kevei Péter
Egyetemi docens
Sztochasztika tanszék

Szeged, 2019

Tartalomjegyzék

1. Bevezető

Dolgozatom témája a PageRank algoritmus, melynek legfőbb alkalmazási területe az internetes weboldalak bizonyos szempontok szerinti rangsorolása.

Ebben a fejezetben röviden ismertetem a PageRank algoritmus történetét és megemlítek egy PageRank számolásához használható, általam készített programot. A második fejezetben a PageRank algoritmus tárgyalásához szükséges alapvető fogalmakat és definíciókat vezetek be.

1.1. A PageRank algoritmus története

A PageRank algoritmust Larry Page és Sergei Brin alkották meg 1998-ban, a Standford Egyetem hallgatóiként. Arra kerestek megoldást, hogy miként lehet az akkoriban robbanásszerű növekedésnek induló internetet weboldalait továbbra is rangsorolni, mivel látszott, hogy az akkor alkalmazott keresőmotorok erre már hamarosan nem lesznek képesek.

Úgy gondolták, hogy a rangsorolás alapja az internet hiperlink struktúrája kell hogy legyen. Feltették, hogy ha egy oldal linkel egy másikra, az kifejezi azt, hogy az oldal készítői megbíznak a linkelt oldalban, ezért az algoritmusukat úgy építették fel, hogy egy oldal fontossága (ezt szintén PageRanknak nevezik) attól függ, hogy mennyi és milyen fontos oldalak linkelnek rá.

Larry Page és Sergei Brin találmánya olyan jól alkalmazhatónak bizonyult, hogy Google néven saját vállalkozást alapítottak. A cég keresőmotorjának alapja máig a PageRank algoritmus, skálázhatóságát mutatja, hogy a Google ma már több mint 130 billió weboldalt indexel.

1.2. PageRank program

Dolgozatomhoz grafikus felhasználói felülettel ellátott asztali alkalmazást is készítettem, Java nyelven. Ez a program képes előre megadott oldalszámú webet különböző paraméterek alapján véletlenszerűen generálni, majd ehhez PageRankot számolni. A 150 oldalnál kisebb méretű webekhez tartozó mátrixokat meg is tudja jeleníteni.

A program és annak forráskódja is letölthető a következő oldalról:

https://github.com/Gtomika/PageRank/releases/tag/v4.1

A futtatáshoz a számítógépen telepítve kell hogy legyen a Java.

2. Alapfogalmak és definíciók

2.1. Linkek, web és a lógó oldalak

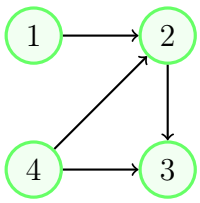
Ahhoz hogy PageRankról beszélhessünk, először a web fogalmát kell bevezetni.

1. Definíció: Linkhalmaz

Legyen V a weboldalak halmaza. Ekkor $L \subset V \times V$ **linkhalmaz**ban (v_1, v_2) ($v_1, v_2 \in V$ pontosan akkor van benne ha v_1 linkel v_2-re.

2. Definíció: Web

Legyen V a weboldalak halmaza, L pedig az ehhez tartozó linkhalmaz. Ekkor a **web** egy irányított gráf, melynek csúcsai V elemei, élei pedig L elemei, ahol ha $v_1, v_2 \in V$ és $(v_1, v_2) \in L$, akkor az él v_1-ből v_2-be mutat.



1. ábra. Egy 4 oldalból álló web

A linkhalmaz definíciója ugyan megengedi azt, hogy egy oldal önmagára linkeljen, mert bár egy valós web esetén ez lehetséges, de a PageRank algoritmus esetén nem akarjuk, hogy az ilyen linkek is beleszámítsanak a rangsorolásba. Ezeket a linkeket ezért egyszerűen elhagyjuk.

Az is megfigyelhető, hogy a definíciókban a linkeknek nincsen multiplicitása, egy oldal vagy linkel egy másikra, vagy nem. Ez azért van, hogy a rangsorolást ne lehessen olyan egyszerűen manipulálni, hogy bizonyos oldalak sok linket tartalmaznak egy másikra.

Előfordulhat olyan web, ahol bizonyos oldalakról nincsenek kimenő linkek. Az ilyen oldalakat lógó oldalnak nevezzük (azokat a linkeket amelyek rájuk mutatnak pedig lógó linkeknek). A későbbiekben ezek az oldalak problémákat okoznak, ezért megadunk egy módszert, mellyel a ezeket az oldalakat el lehet tüntetni: minden lógó oldalt helyettesítünk egy, az összes másik oldalra linkelővel (ebben a kivételes esetben megengedjük az önmagára linkelést is).

Egy lógó oldallal (4) rendelkező web, és ugyanez a web, helyettesített oldallal.

A helyettesítés mögötti heurisztika az, hogy ha egy böngésző egy olyan oldalra érkezik, ahonnan a linkeken keresztül nem tud továbbmenni, akkor véletlenszerűen, egyenletes eloszlás szerint választ az összes oldal közül.

2.2. A PageRank definíciója

Egy oldal fontossága azon múlik, hogy mennyi oldal linkel rá, és hogy ezek milyen fontosak. A linkelő oldalak fontosságára azért van szükség, mert enélkül egy oldaltulajdonos tudná úgy növelni a weboldalának fontosságát, hogy rengeteg oldalt hoz létre, melyek mind linkelnek egymásra és a saját oldalára (ezt link farmnak nevezik [3]).

Egy webet el lehet úgy is képzelni, mint az oldalak demokráciáját ahol minden oldalnak szavazata van és ezt a szavazatot (és még a kapott szavazatokat is) továbbosztja úgy, hogy linkel a többi oldalra.

3. Definíció: Weboldal PageRankja

Legyen adott egy web, V az oldalak, L a linkek halmaza. Legyen $v_i \in V$ oldal PageRankja $r(v_i)$, az oldalról kimenő linkek száma pedig $\mid v_i \mid$.

Jelölje $B_i \subset V$ azon oldalak halmazát amelyen linkelnek v_i-re, azaz

$$B_i = \{v_j \in V : \exists l \in L, \quad l = (v_j, v_i)\}$$

Ekkor bármely v_i oldal PageRankját megkapjuk a következő módon:

$$r(v_i) = \sum_{v_j \in B_i} \frac{r(v_j)}{\mid v_j \mid}$$

A definícióban megjelenik az is, hogy egy oldalnak mennyi kimenő linkje van. Minél több oldalra linkel, annál kevesebbet fog számítani az ő linkjének értéke. Ez ellensúlyozza a már említett link farmokat.

A szummában szereplő oldalak egyikénél sem lehet a kimenő linkek száma nulla, mert mindegyik oldal eleme a B_i halmaznak, azaz legalább v_i-re linkelnek. Előfordulhat, hogy egy oldalra nincs egyetlen link sem. A definícióban ilyenkor egy üres összeg szerepel, azaz az oldal rangja 0.

A PageRank definíciója tehát rekurzív. Egy oldal rangjának meghatározásához minden rá linkelő oldal rangját ismernünk kell. Ha a webünk n db oldalt tartalmaz, akkor a definíció meghatároz n db lineáris egyenletet.

A cél az, hogy olyan algoritmust adjunk meg, amely egyrészt pontosan meghatározza minden oldal PageRankját, másrészt nagyon nagy n-re is hatékonyan működik, mind idő-, mind tárigény szempontjából.

3. A sajátértékprobléma

3.1. Linkmátrix és kapcsolata az egyenletrendszerrel

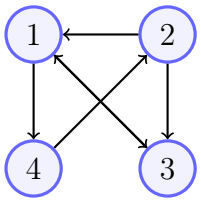
Egy web linkmátrixa az oldalak közötti kapcsolatokat reprezentálja mátrixos formában.

4. Definíció: Linkmátrix

Legyen adott egy web ahol az oldalak halmaza V. Ennek linkmátrixa legyen $A = (a_{i,j}) \quad i, j = 0, 1, ..., |\ V|$, ahol

$$a_{i,j} = \begin{cases} \frac{1}{|\ v_i|}, & \text{ha } (v_i, v_j) \in L. \\ 0, & \text{egyébként.} \end{cases} \tag{1}$$

Tehát a linkmátrix sorai kifejezik azt, hogy egy oldal mennyi és melyik másik oldalakra linkel. Megfigyelhető, hogy amennyiben a web nem tartalmaz lógó oldalt, úgy a linkmátrix sztochasztikus lesz, azaz sorainak összege mindig 1, amennyiben pedig tartalmaz ilyen oldalt, a 2.1 alfejezetben megadott helyettesítéssel kapott mátrix már sztochasztikus lesz.



$$\begin{bmatrix} 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Példa: egy 4 oldalból álló web és linkmátrixa

Kapcsolat van az oldalak PageRankjának definíciójából kapott egyenletrendszer és a linkmátrix között. Ennek segítségével az egyenletrendszer felírható a következő mátrixegyenletként:

$$x = xA$$

ahol x az oldalak rangjait tartalmazó vektor. Ezt átalakítva a következőt kapjuk:

$$x = A^T x$$

Ebből mátrixegyenletből látszik, hogy a keresett vektor az A^T mátrix 1 sajátértékéhez tartozó sajátvektor. Természetesen egy sajátvektor bármilyen nem nulla skalár szorosa is sajátvektor ezért az egyértelműség kedvéért tegyük fel, hogy az általunk keresett vektor az, ahol a komponensek összege 1, ez a későbbiekben hasznos lesz.

Megjegyzés: Ha webnek az internet egy sok oldalból álló részhalmazát választjuk (márpedig a PageRankokat ilyen esetre akarjuk kiszámolni), akkor a linkmátrix nagyon ritka lesz, azaz szinte az összes eleme 0. Ennek oka, hogy az egy oldal által linkelt további oldalak száma elenyésző az összes oldal számához képest. A linkmátrix ritkasága egy számolási szempontból nagyon kedvező tulajdonság.

3.2. Az 1 sajátérték

Felmerül a kérdés, hogy minden A^T mátrixnak sajátértéke-e az 1. Ehhez elég belátni, hogy az A linkmátrixnak az 1 mindig sajátértéke, mivel egy mátrixnak és transzponáltjának a sajátértékei megegyeznek.

1. Tétel: Az 1 sajátértéke minden lógó oldalt nem tartalmazó web linkmátrixának

Bizonyítás. Legyen A egy tetszőleges, lógó oldalt nem tartalmazó, n oldalból álló web linkmátrixa. Ekkor az i. sorhoz $(i = 1, 2, ..., n)$ tartozó sorösszeg $\mid v_i \mid \times \frac{1}{\mid v_i \mid} = 1$, azaz minden ilyen mátrix sztochasztikus.

Legyen e az az n komponensű oszlopvektor, melynek minden komponense 1. Ekkor az

$$Ae = e$$

egyenlet teljesül, mert az eredményvektor i. komponense ($i = 1, 2, ..., n$)

$$\sum_{j=1}^{n} 1a_{i,j} = 1$$

a mátrix sztochasztikus tulajdonsága miatt. Tehát bármely linkmátrixnak ez az e vektor sajátvektora, 1 sajátértékkel.

□

Ugyan bármely mátrixnak és transzponáltjának sajátértékei megegyeznek, de az nem igaz, hogy ezen sajátértékekhez tartozó sajátvektorok is azonosak, ezért a fenti tételben bevezetett e vektor nem biztos, hogy megoldása lesz a PageRank definíciója által meghatározott egyenletrendszernek. Ezzel a megoldás egyébként is használhatatlan az oldalak rangsorolásában, mivel minden oldalnak azonos rangot ad.

A lógó oldalakat tartalmazó webek esetén az 1 nem lesz mindig sajátértéke az A^T mátrixnak (lehetnek 0 összegű sorok, ami a fenti bizonyítást elrontja), azonban figyeljük meg, hogy ha elvégezzük a 2.1 alfejezetben megadott helyettesítést akkor a kapott mátrix már sztochasztikus lesz, és erre már érvényes az előbbi tétel.

Tehát beláttuk, hogy minden ilyen mátrixnak vagy sajátértéke az 1, vagy egy egyszerű helyettesítéssel el lehet érni, hogy az legyen.

3.3. A legnagyobb sajátérték

2. Tétel: Minden lógó oldalt nem tartalmazó linkmátrix bármely λ sajátértékre $|\lambda| \leq 1$.

Bizonyítás. A sajátértékek abszolút értékének felülről becsléséhez alkalmazzuk a Gershgorin körtételt. Eszerint a mátrix összes sajátértékének benne kell lennie legalább egy Gershgorin körben. A Gershgorin körtétel ugyan komplex mátrixokra is érvényes, de a linkmátrix csak valós elemeket tartalmazhat. A köröket meghatározó halmazok:

$$K_i = \{r \in \mathbb{R} : |\ r - a_{i,i}|\ \leq R_i\} \quad i = 1, 2, ..., n$$

ahol R_i az i. sorösszeg és $a_{i,i}$ az i sorban lévő főátlóbeli elem.

Már korábban beláttuk, hogy a feltételeknek eleget tévő linkmátrixok sztochasztikusak, azaz a sorösszeg minden sornál egy. A linkhalmaz 2.1 alfejezetben szereplő definíciójában nem engedjük meg azt, hogy egy oldal önmagára linkeljen, ezért $a_{i,i}$ minden sorra 0.

Ezen állítások miatt a köröket meghatározó halmazok a következőre egyszerűsödnek minden sor esetén:

$$K_i = \{r \in \mathbb{R} : |\ r|\ \leq 1\} \quad i = 1, 2, ..., n$$

Ebből következik, hogy bármely λ sajátértékre $|\ \lambda|\ \leq$ 1-nek teljesülnie kell. □

Beláttuk, hogy az egy minden esetben sajátérték, és azt is hogy ennél (abszolút értékben) nem is lehet nagyobb sajátérték. Ebből azonban nem következik az, hogy domináns is lenne. Előfordulhat olyan eset is, hogy a -1 is sajátérték, vagy az 1 többszörös multiplicitással szerepel.

3.4. A hatványiteráció

Ha sok oldalból álló webek esetén akarjuk meghatározni a rangokat, akkor a pontos megoldást adó módszerek nagyon lassúak, ezért közelítő, iteratív módszerre van szükség. Ideális választás a hatványiteráció, mert ebben az esetben elég tárolni mindössze a linkmátrix transzponáltját, és a PageRank vektor éppen aktuális iterációját a számoláshoz.

A hatványiteráció általános alakja:

$$x^{i+1} = Mx^i$$

ahol $M \in \mathbb{R}^{n \times n}$ és $x^0 \in \mathbb{R}^n$.

A hatványiteráció azonban a domináns sajátértékhez tartozó sajátvektorhoz konvergál. Ha ilyen nincsen, akkor nem tudjuk garantálni a konvergenciát. Ahhoz, hogy alkalmazhassuk az 1-hez tartozó sajátvektor megtalálására, vagy be kell látni, hogy az 1 mindig domináns sajátérték, vagy úgy kell módosítani a linkmátrixon, hogy az legyen.

4. Konvergencia a PageRank vektorhoz

4.1. Gondot okozó tényezők

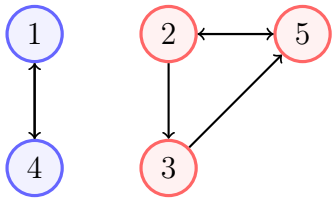
Amikor egy nagy webhez tartozó PageRank vektort szeretnénk közelíteni, fontos tudni, hogy az iteráció úgy fog viselkedni ahogyan várjuk.

A következő lehetőségeket kell ellenőrizni:

1. Biztosan konvergálni fog-e a hatványiteráció?
2. Ha igen, akkor az indítási helytől függetlenül mindig ugyanabba a vektorba?

Egyértelmű, hogy az indulási helytől független, biztos konvergenciát a 3.3 alfejezet végén említett szélsőséges esetek ronthatják el [1].

Ezek az eseten olyan webek esetén fordulnak elő, amelyek valójában több egymástól független (azaz nincsenek közöttük linkek) webből állnak össze [5].



$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Két egymással nem érintkező "szubwebből" álló web és linkmátrixa.

A fenti ábrán látható web linkmátrixának sajátértékei között szerepel a -1 és az 1 is, kétszeres multiplicitással.

4.2. Webekhez rendelt Markov-láncok

Vegyünk egy lógó oldalt nem tartalmazó webet. Legyen adott egy úgynevezett véletlen szörföző, aki minden lépésben egyenletes eloszlás szerint választ a jelenlegi oldalon lévő linkek közül és továbbhalad a választott link által mutatott oldalra. A kezdeti eloszlás szintén legyen egyenletes.

Az ehhez a Markov-lánchoz tartozó átmenetmátrix éppen a linkmátrix lesz. A 3.1 alfejezetben meghatároztuk, hogy a keresett PageRank vektor

megoldása az $x = xA$ mátrixegyenletnek. Látszik, hogy ez éppen a Markov-lánc invariáns eloszlása. Ezért mondtuk azt, hogy a sajátvektorok közül azt választjuk, ahol a komponensek összege 1.

Ez logikus, hiszen az invariáns eloszlás mutatja, hogy hosszú távon az idő mekkora részét tölti a véletlen szörföző az állapotokban, tehát azok az oldalak lesznek a legfontosabbak, ahol várhatóan a legtöbb időt töltjük.

Megjegyzés: Nem minden webhez rendeletünk egy ilyen Markov-láncot. A lógó oldalakat tartalmazó webek linkmátrixában vannak csupa nulla sorok, melyek elrontják a mátrix sztochasztikusságát, de a 2.1 alfejezetben megadott helyettesítéssel kapott webek már megfelelőek.

4.3. A konvergencia garantálása

A domináns sajátérték (és ezzel együtt a konvergencia) létezésének garantálásához használjuk fel a Perron–Frobenius tételt.

Perron–Frobenius tétel: Ha egy $A = (a_{i,j})$ $n \times n$ mátrix minden eleme pozitív, akkor biztosan létezik domináns sajátérték, azaz ha a sajátértékek $\lambda_1, \lambda_2, ..., \lambda_n$, akkor

$$\exists r \in \{1, 2, ..., n\} : \quad \forall i \neq r \quad |\lambda_i| < \lambda_r$$

A tétel értelmében tehát az egyértelmű PageRank vektor létezéséhez annyit kell tenni, hogy a linkmátrixot úgy módosítjuk, hogy csak pozitív elemei legyenek.

5. Definíció: Google mátrix

Legyen $A \in \mathbb{R}^{n \times n}$ egy lógó oldalt nem tartalmazó web linkmátrixa, S pedig egy egy olyan $n \times n$-es mátrix, melynek minden eleme $\frac{1}{n}$. Ekkor a Google mátrix a következőképpen áll elő:

$$G = \alpha A + (1 - \alpha)S$$

ahol $\alpha \in [0, 1]$.

Látható, hogy a Google mátrix értéke függ az α paramétertől, de majdnem minden esetben (az $\alpha = 1$-et kivéve) a kapott mátrix minden eleme biztosan pozitív, azaz létezik domináns sajátérték és a hatványiteráció indulási helytől függetlenül az ehhez tartozó sajátvektorhoz fog konvergálni.

$$\begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad \begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{bmatrix}, \quad \begin{bmatrix} 0,038 & 0,463 & 0,463 & 0,038 \\ 0,321 & 0,038 & 0,321 & 0,321 \\ 0,321 & 0,321 & 0,38 & 0,321 \\ 0,038 & 0,038 & 0,888 & 0,038 \end{bmatrix}$$

Egy web linkmátrixa, az S mátrix és az $\alpha = 0,7$ paraméterrel kapott Google mátrix.

Megjegyzés: A helyettesítés mögötti heurisztika.
Tekintsük a webhez rendelt Markov-láncot. Ezt úgy módosítjuk, hogy a véletlen szörföző most már nem csak a linkeken keresztül juthat el a következő oldalra, hanem bizonyos valószínűséggel egy egyenletes eloszlás szerint választott véletlen oldalra ugrik. Ebben a modellben már nem lehetnek teljesen különálló oldalhalmazok, a véletlen szörföző bárhonnan eljuthat bárhova.

Az így kapott Markov-lánc átmenetmátrixa éppen a Google mátrix lesz, ahol az α paraméter dönt arról, hogy mekkora valószínűséggel követjük a linkeket és mekkorával ugrunk véletlen oldalra.

5. A Google mátrix és az α paraméter

5.1. A Google mátrix tulajdonságai

A Google mátrix szintén sztochasztikus lesz, bármely α paraméter esetén. A linkmátrixról (A) és a csupa $\frac{1}{n}$ komponensből álló (S) mátrixokról tudjuk, hogy sztochasztikusak, ezért αA és $(1-\alpha)S$ mátrixok sorösszegei rendre α és $(1-\alpha)$. A Google mátrix esetén így a sorok összege $\alpha + (1-\alpha) = 1$ lesz.

A 3.1 alfejezetben említettük, hogy a linkmátrix általában egy ritka mátrix és hogy ez számítási szempontból nagyon előnyös. A Google mátrix ennek éppen az ellentéte, nemhogy sűrű, de biztosan nem tartalmaz egyetlen 0 elemet sem, ugyanis ez szükséges ha a hatványiteráció konvergenciáját a Perron–Frobenius tétellel akarjuk garantálni. Sűrű mátrixok esetében azonban a mátrixszorzás sokkal lassabb mint ritkáknál.

A Google mátrix sűrűségéből adódó számítási problémát ki lehet küszöbölni. Ha hatványiteráció $i+1$.-ik lépése

$$x^{i+1} = Gx^i$$

ahol x a PageRank vektor, akkor ez ekvivalens a következő alakkal:

$$x^{i+1} = \alpha A x^i + (1-\alpha)S$$

Először tehát a linkmátrix skalárszorosával szorozzuk a PageRank vektor jelenlegi iterációját, majd ezután adjuk hozzá a csupa $\frac{1-\alpha}{n}$ elemekből álló mátrixot. Ekkor az elvégzett mátrixszorzásban már nem szerepel a sűrű Google mátrix, de az iteráció eredménye ugyanaz, mitha azzal szoroztunk volna.

5.2. Az α paraméter kérdése

Az előző fejezetben láttuk, hogy szinte bármilyen $\alpha \in [0, 1]$ választás esetén (az $\alpha = 1$ esetet kivéve) garantáljuk a hatványiteráció konvergenciáját, de felmerül a kérdés, hogy van-e egyéb jelentőség ennek a paraméternek, és ha igen, akkor hogyan válasszuk meg.

A 4.3 alfejezetben megadott heurisztikából kiderül, hogy α nem lesz más, mint annak a valószínűsége, hogy a véletlen szörföző követi a linkeket, ahelyett, hogy tetszőleges oldalra ugrana. Mivel az oldalak fontosságát a linkek alapján szeretnénk meghatározni (és a PageRankot is így definiáltuk), ezért adódik, hogy α-t válasszuk az 1-hez minél közelebbi értéknek, mert így a web struktúrája, a linkek nagy hangsúlyt kapnak a Google mátrixban is. Az 1-hez közeli α azt jelenti, hogy a véletlen szörföző szinte mindig a linkeken keresztül halad tovább.

Tudjuk azonban, hogy a Google, saját PageRank számolásánál $\alpha \approx 0,85$ [2] értéket használ. A fentiek ismeretében miért nem 0,9 vagy akár 0,999 ez az érték?

A válasz az, hogy α-tól nagyban függ a konvergencia sebessége, amit a következő fejezetekben indoklunk.

5.3. A hatványiteráció konvergenciájának sebessége

Tegyük fel, hogy a mátrixnak, melyre a hatványiterációt szeretnénk alkalmazni a következők a sajátértékei:

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq ... \geq |\lambda_n|$$

Látható, hogy λ_1 domináns sajátérték, azaz a hatványiteráció biztosan konvergálni fog. A valódi sajátvektor és az iteráció eredménye közti kü-

lönbség minden lépésben aszimptotikusan csökken $|\ \lambda_1|\ /|\ \lambda_2|$ valamilyen többszörösével [4].

λ_1 domináns sajátérték, ezért a vizsgált hányadosról biztosan tudjuk, hogy:

$$1 < |\ \lambda_1|\ /|\ \lambda_2|$$

Látható, hogy minél kisebb ez a hányados, annál kisebb mértékben fog csökkenni a hiba mértéke az iterációs lépések között, azaz a konvergencia sebessége és a helyzet akkor a legrosszabb amikor

$$|\ \lambda_1|\ /|\ \lambda_2|\ \approx 1$$

$$|\ \lambda_1|\ \approx |\ \lambda_2|$$

A google mátrix estében a sajátértékek

$$1 > |\ \lambda_2|\ \geq |\ \lambda_3|\ \geq ... \geq |\ \lambda_n|$$

ezért a fentebb említett legrosszabb eset a következőképpen módosul:

$$1/|\ \lambda_2|\ \approx 1$$

Minél közelebb van tehát a második legnagyobb sajátérték 1-hez (a domináns sajátértékhez), annál lassabb lesz a konvergencia sebessége.

5.4. α mint sajátérték

Hivatkozások

[1] ??? How google works: Markov chains and eigenvalues, May 2015.

[2] Carl D. Meyer Amy N. Langville. *Deeper inside PageRank, Internet Math.* 2005.

[3] Carl D. Meyer Amy N. Langville. *Google's PageRank and Beyond: The Science of Search Engine Rankings.* Princeton University Press, 2012.

[4] David Bindel. Power iteration, August 2016.

[5] Tanya Leise Kurt Bryan. *The linear algebra behind Google.* 2016.