Matematikk og informasjonssøking på nettet

D. Laksov

Matematiska Institutionen, KTH
SE–100 44 Stockholm
laksov@math.kth.se

Innledning[1]

Matematikken spiller en fundamental rolle for nesten alle funksjoner i datamaskiner, og for all elektronisk kommunikasjon. Vi skal illustrere dette ved å forklare hvordan matematikken brukes i informasjonssøking på *nettet*. En av de forbausende egenskapene med denne anvendelsen er at matematikken har størst betydning når det gjelder å avgjøre hvilke nettsider som er «viktigst» blandt de nettsidene som inneholder samme informasjon. Det vil si, matematikken er sentral nettopp i den delen av søkeprosessen der det virker som om smak og subjektiv bedømning spiller en vesentlig rolle. Dette viser hvordan matematikken trenger gjennom problemer der det er mange uoversiktlige faktorer som gjensidig påvirker hverandre, og hvordan den gir en klar og objektiv analyse av situasjonen.

Vi skal vise hvordan problemet med å rangordne nettsider blir løst ved enkle, velkjente og *klassiske* matematiske ligninger, takket være en genial idé og god kjennskap til hvordan nettet fungerer. Ettersom ligningssystemene er så store, med veldige mengder variable behøves det spesielle metoder for å løse dem effektivt. Metoden for å løse disse ligningene bygger på resultater fra begynnelsen av nittenhundretallet. Det er interessant at disse metodene bare gjelder under betingelser som vi ikke kan vise er oppfylt for de ligninger vi vil behandle. I praksis fungerer imidlertid metodene utmerket. Dette viser at matematikken er til nytte selv i tilfellene der alle de tekniske matematiske detaljene ikke har noen betydning. Matematikken forteller i hvilken retning vi skal lete for å løse problemene som oppstår.

[1]En varm takk til Tommy Ekola (KTH) for all hjelp med denne artikkelen.

De viktigste delene av denne artikkelen bør være forståelige for lesere med vanlige matematikkunnskaper fra gymnaset. I avsnitt 3 har vi tatt med noe mer krevende matematiske resultater.

1. Informasjonssøking på nettet

1.1 Nettlesere. En av de viktigste anvendelsene av datamaskiner er å søke informasjon på nettet. I dag har nesten alle tilgang til en *nettleser*, som Explorer, Mozilla, Netscape, eller Opera, og med disse programmene får man tilgang til en rekke ulike *søkemotorer*, som Alta Vista, Google, Lycos, eller Yahoo. Tatt i betraktning at det i dag finnes mer enn 3 milliarder[2] *nettsider* spredt over hele verden, og mange av dem inneholder store mengder med dokumenter er det vanskelig å fatte at det er mulig å lete gjennom disse enorme datamengdene, og på bare noen sekunder komme opp med de viktigste nettsidene som inneholder informasjonen vi søker. En oppmerksom bruker vil oppdage at det er betydelige forskjeller mellom de ulike søkemotorene, både når det gjelder hastighet og den informasjonen de finner. For å forstå forskjellen mellom programmene og hvordan det er mulig å oppnå slike søkehastigheter er det nødvendig å kjenne til prinsippene og teoriene bak programmene.

Som så ofte når det gjelder viktige tekniske anvendelser, og spesielt når det gjelder *verktøyene* som hjelper oss å bruke nettet, så er det matematiske resultater og formler som ligger til grunn for anvendelsene, og som forklarer hvordan en slik effektivitet og presisjon er mulig. Det er også karakteristisk for slike anvendelser at den matematikken som brukes er *klassisk* og ble funnet uten tanke på disse anvendelsene.

Vi skal her forklare i hvilken del av søkingene etter informasjon som matematikken spiller en avgjørende rolle, og vi skal gi et lite innblikk i den matematikken som behøves.

1.2 Søkemotorer. En enkel modell for en søkemotor på nettet består av tre deler:

1. En robot. Den første ingrediensen er en *robot* som døgnet rundt søker opp nettsider og laster ned dokumentene på nettsidden i en *database*. Den mest kjente søkemotoren er *google* som kan søke opp og laste ned mer enn 2 milliarder nettsider i løpet av en uke. Det vil si, den klarer 3300 nettsider per sekund.

Grunnen til at dette arbeidet er mulig er at hver maskin med nettsider har et internettnummer som er forholdsvis lett å finne på nettet. Har man først funnet en maskin er det lett å finne nettsidene på maskinen, og dermed også alle *lenkene* på hver nettside. Ved hjelp av lenkene kan roboten også komme videre til nye nettsider.

2. Ordliste. Den andre ingrediensen er en *ordliste*, som inneholder de fleste viktige ordene som forekommer på dokumentene på de nettsidene vi har lastet ned ved hjelp av roboten. En slik liste kan inneholde mer enn 100 millioner ord, hvilket kan sammenliknes med de par tusen ordene vi bruker i daglig skrift og tale. Til hvert ord i ordlisten finnes en *peker* til de nettsidene der ordet forekommer. Hver gang vi

[2] Alle oppgaver om tider og antall må taes med en klype salt. Nettet og datamaskinene endrer seg med en skremmende hastighet. Nye brukere kommer til, maskinene blir hurtigere, og hukommelsene større.

skriver inn ett, eller flere, ord vi vil søke på, leter søkemaskinen i ordlisten til den finner alle ordene, og den vet da også hvilke nettsider ordene forekommer på. Med de hurtigste søkemaskinene tar det mindre enn et sekund å finne ordkombinasjoner med 5–6 ord.

Det tar selvsagt lang tid å sette opp en ordliste over ordene som forekommer i en database med milliarder av dokumenter, men det er mindre krevende enn å finne og laste ned dokumentene i databasen.

3. Rangordning. Den tredje ingrediensen består i å rangordne nettsidene etter hvor *viktige* de er. Hver gang vi søker på et, eller flere, ord, finnes ordet, eller ordene, oftest på tusentalls nettsider. For at et søk skal være meningsfull er det derfor helt avgjørende at de nettsidene som kommer først opp på skjermen er de som har størst betydning for den som søker. Prøver vi for eksempel å finne opplysninger om Edvard Grieg vil vi ikke gjerne komme til tusenvis av nettsider som handler om noe helt annet, og der Grieg bare er nevnt i forbifarten. Vi vil komme direkte til de store sentrale arkivene som inneholder viktig informasjon om Grieg og hans verk.

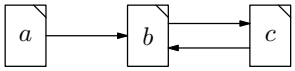
Det er lett å forstå at det er en vanskelig oppgave å rangordne alle nettsidene på nettet. At en nettside er «viktigere» enn en annen virker å bero på en *subjektiv* vurdering. Men subjektive bedømmelser av milliarder av nettsider, mange av dem med store mengder dokumenter, er selvsagt umulig. Dessuten er nettsidene veldig ulike. De spenner fra personlige nettsider som innholder noen få linjer tekst til store databaser med millioner av dokumenter. For å rangordne sidene må vi derfor prøve å finne *objektive* kriterier for hva som er «viktig», og som bare avhenger av den *formelle strukturen* av dokumentene, og ikke beror på deres innhold. Det finnes en rekke forslag på hvordan man skal rangordne nettsider etter «viktighet», og leseren kan selv tenke gjennom hvilke løsninger som kan være interessante. Skal man bruke antallet ganger et ord forekommer på en nettside, eller kanskje antallet henvisninger til andre sider, eller kanskje antallet dokumenter på nettsiden? Den store forskjellen mellom de ulike søkemaskinene ligger nettopp i hvordan de løser denne oppgaven. De som klarer det best blir mest benyttet kommersielt og får derfor best råd til å kjøpe inn flere datamaskiner og mer minne, og dermed bli markedsledende. I dag har de største søkemaskinene titusentalls datamaskiner som arbeider parallellt med å søke og lagre data.

De fleste søkemaskinene bruker mange ulike kriterier for å rangordne nettsider, som ordfrekvens, plassering av ordene i dokumentene, og avstanden mellom ord. Vi skal i resten av denne artikkelen forklare en metode for å rangordne nettsider, eller i det hele tatt store datamengder, som bare avhenger av *lenkene* på nettsidene. Metoden kalles *PageRank* og ble foreslått av Sergey Brin og Larry Page [1] for omkring 5 år siden, da de var studenter ved Stanford University. PageRank er hjertet i den fenomenale søkemotoren *google* og bygger på enkle og fundamentale matematiske idéer og resultater.

1.3 «Viktighet»/relevans. Vi har brukt adjektivet *viktig* om nettsider for en egenskap som brukes til å rangordne nettsidene. Ordet *viktig* er sterkt følelseladet. Derfor vil vi påpeke at ordet i denne forbindelsen betyr at informasjonen på nettsiden har stor betydning for det spesielle søket vi gjør, og ikke er noe forsøk til en subjektiv eller objektiv bedømning av nettsiden. Når vi sier at «en nettside er viktig» betyr det at «nettsiden har stor relevans, eller betydning, for den aktuelle søkningen».

2. PageRank

2.1 Lenker. Idéen bak PageRank er å bruke *lenkene* mellom nettsidene til å avgjøre hvor viktige de er. En lenke fra en nettside a til en nettside b er en henvisning i et dokument på nettsiden a som er slik at om man *klikker* på henvisningen så kommer man til nettsiden b. Alle som har sett en nettside vet hvordan slike lenker ser ut og har brukt dem til å komme fra en nettside til en annen. Vi skal bruke litt *matematisk terminologi* og si at a peker på b og skriver



om a, b og c er nettsider og a og c peker på b, mens nettsiden b peker på nettsiden c.

2.2 Rangordning. Problemet er å gi hver nettside a en *rang* R_a. Det vil si, vi vil tilordne et tall R_a til nettsiden a som forteller hvor viktig nettsiden er i forhold til de andre nettsidene. Med andre ord, om nettsiden a er «viktigere» enn nettsiden b så skal R_a være større enn R_b, og a skal derfor komme opp på skjermen før b om begge nettsidene a og b inneholder ordene vi søker. Det er for å løse dette problemet at vi behøver matematikk. For å forklare metoden begynner vi derfor med litt matematisk *notasjon*:

> For hver nettside a betegner vi dens rang med R_a. Det er disse tallene vi skal bestemme ved å finne ligninger som de tilfredsstiller. Vi betegner med $|F_a|$ antallet lenker som finnes på nettsiden a. Videre betegner vi med B_a alle nettsidene som peker på a, det vil si alle nettsidene som har et dokument som inneholder en lenke til a. En litt foreklet form for ligningene som bestemmer tallene R_a i PageRank er
>
> $$\lambda R_a = \sum_{b \in B_a} \frac{R_b}{|F_b|}, \tag{PR}$$
>
> der λ er et tall som vi også må bestemme.

Det er et par ting vi bør merke oss med dette ligningssystemet før vi kan forstå hvordan denne modellen for rangordning fungerer.

2.3 Bemerkning. Rangordningen R_a er bestemt av de nettsidene som peker på a og hvor mange lenker som finnes på disse nettsidene. Bidraget til R_a blir stort om mange sider med høy rangordning, og med få lenker, peker på den. Dette er en rimelig *modell* ettersom en nettside bør være viktig om den har mange viktige lenker til seg. Dessuten er disse lenkene mer verdt om de kommer fra nettsider med få lenker, hvilket også er tiltalende.

En annen fordel ved denne modellen er at antallet nettsider som peker på en gitt side er vanskelig å manipulere for de som har kommersielle interesser og vil at deres side skal ha høy rangordning for å synes først ved et søk. Vi kan selv velge hvilke sider vi vil lenke til, men ikke hvilke sider som skal lenke til oss.

2.4 Bemerkning. Rangordningen avhenger ikke av hvilket ord vi søker. Om to ord står på de samme nettsidene vil de samme nettsidene komme opp på skjermen, og

i samme rekkefølge når vi søker på de to ordene. For eksempel, om navnene *Scylla* og *Carybdis* forekommer på de samme nettsidene vil samme nettsider komme opp i samme rekkefølge, om vi søker på ordet Scylla, som når vi søker på ordet Carybdis. Meningen med et ord spiller heller ingen rolle. Søker vi på ordet *blad*, så kommer de viktigste nettsidene som inneholder dette ordet opp, uansett om *blad* på disse nettsidene henviser til et blad på et tre, et ukeblad, eller arkene i en bok.

2.5 Bemerkning. Rangordningen til en side R_b blir like fordelt mellom alle nettsidene den peker på, og bidrar til hver nettside med $R_b/|F_b|$, det vil si, termen $R_b/|F_b|$ forekommer i alle ligningen for R_a der b peker på a, og den forekommer bare i disse ligningene. Summerer vi over alle nettsidene H får vi derfor

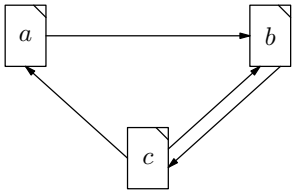
$$\lambda \sum_{a\in H} R_a = \sum_{a\in H} \sum_{b\in B_a} \frac{R_b}{|F_b|} = \sum_{a\in H'} R_a,$$

der H' er nettsidene som inneholder lenker. Vi ser at tallet λ måler kvoten mellom summen av rangordningen til de sidene som inneholder lenker, og summen av rangordningen til alle nettsidene. Ettersom det alltid finnes nettsider uten lenker vil $0 < \lambda < 1$.

2.6 Bemerkning. Vi vet at nettsidene, i gjennomsnitt, har 11 lenker. Det vil derfor også være i gjennomsnitt 11 lenker som peker på en gitt nettside. Derfor vil hver ligning PR stort sett inneholde 12 ukjente med ikke null koeffisienter.

Vi gir nu noen eksempler som viser hvordan ligningene PR ser ut for enkle nett bestående av 3 nettsider. Sammenliknet med det virkelig nettet med milliarder av nettsider er dette ganske lite, men det gir en god ide om hvordan ligningene PR ser ut.

2.7 Eksempel. (Redusible tilfellet)



gir ligningene

$$\begin{aligned} \lambda R_a &= & \tfrac{1}{2}R_c \\ \lambda R_b &= R_a & + \tfrac{1}{2}R_c \\ \lambda R_c &= & R_b \end{aligned}$$

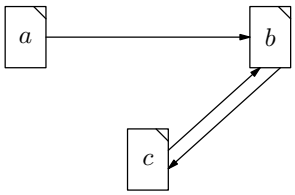
Legger vi sammen de venstre og høyre termene i disse tre ligningene får vi at

$$\lambda(R_a + R_b + R_c) = R_a + R_b + R_c,$$

som viser at enten er $R_a = R_b = R_c = 0$, som vi ikke vil ha, eller vi har at $\lambda = 1$. Med $\lambda = 1$ kan vi løse ligningssystemet og får at $R_a = \frac{1}{2}R_b = \frac{1}{2}R_c$. Vi kan velge R_a vilkårlig til $R_a = 1$, og får rangordningen $R_a = 1, R_b = \frac{1}{2}, R_c = \frac{1}{2}$.

Små eksempler er selvsagt ikke *realistiske*. For eksempel fant vi at $\lambda = 1$, som ikke forekommer i praksis, som vi så i Bemerkning 2.5. Vi må tenke oss at eksempelet er en del av et større nettverk.

2.8 Eksempel. (Loop)



gir ligningene

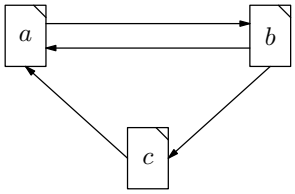
$$\begin{aligned}\lambda R_a &= 0\\ \lambda R_b &= R_a \quad + R_c\\ \lambda R_c &= \quad R_b\end{aligned}$$

Legger vi sammen termene til venstre og høyre i disse tre ligningen får vi at

$$\lambda(R_a + R_b + R_c) = R_a + R_b + R_c,$$

som viser at enten vil $R_a = R_b = R_c = 0$, som vi ikke vil ha, eller så er $\lambda = 1$. Med $\lambda = 1$ kan vi løse ligningene, og vi får $R_a = 0$ og $R_b = R_c$. Vi kan velge R_b vilkårlig til $R_b = 1$, og får rangordningen $R_a = 0, R_b = 1, R_c = 1$. Som i forrige eksempel er ikke dette eksempelet heller spesielt realistisk, og vi må tenke oss at dette også er en del av et større nettverk.

2.9 Eksempel. (Hengende nettsider)



gir ligningene

$$\begin{aligned}\lambda R_a &= \quad \tfrac{1}{2}R_b\\ \lambda R_b &= \tfrac{1}{2}R_a\\ \lambda R_c &= \tfrac{1}{2}R_a + \tfrac{1}{2}R_b\end{aligned}$$

Legger vi sammen termene til venstre og høyre i disse tre ligningene får vi at

$$\lambda(R_a + R_b + R_c) = R_a + R_b.$$

For $\lambda = 0$ får vi $R_a = R_b = 0$, og velger vi R_c vilkårlig til 1 får vi rangordningen $R_a = 0, R_b = 0, R_c = 1$. Når $\lambda \neq 0$ kan vi løse ligningene for R_a og får $R_a = 2\lambda R_b =$

$4\lambda^2 R_a$. Om vi vil ha $R_a \neq 0$ får vi at $\lambda = \frac{1}{2}$. Dette gir $R_a = R_b$, og $R_c = R_a + R_b$. Vi kan velge R_a vilkårlig til $R_a = \frac{1}{2}$ og får rangordningen $R_a = \frac{1}{2}, R_b = \frac{1}{2}, R_c = 1$.

2.10 Bemerkning. I praksis vil man unngå *hengende nettsider*, det vil si nettsider som ikke inneholder noen lenker. Disse taes derfor bort under beregningene og settes tilbake til slutt.

Vi vil også unngå *looper*, det vil si kjeder av nettsider der hver nettside peker til den neste i kjeden, og der det finnes en nettside som ikke inngår i kjeden, men som peker til en av medlemmene av kjeden. Det er for å håndtere slike looper at man i praksis bruker en variant på ligningene PR.

2.11 Løsninger. Vi har ikke løst problemet med å rangordne sider bare fordi vi har satt opp ligningssystemet PR. Vi må også finne en metode for å løse disse ligningene i løpet av en rimelig tid. De tradisjonelle metodene som vi lærer på universitetene, for eksempel *Gauss–Jordan eliminasjon*, er altfor langsomme og for vanskelige å håndtere for ligningssystemer som inneholder så mange som 3 milliarder ligninger i like mange ukjente, selv når de fleste koeffisientene er 0. Når koeffisientene er positive eller null finnes det imidlertid andre metoder, som bruker *iterasjon*. Dette viser seg å være veldig effektivt for ligningene PR. I praksis rekker det med omkring 50 iterasjoner for å få en meget god rangordning for nettsidene. Vi skal i de neste seksjonene gi en matematisk motivasjon til hvorfor det fungerer så bra å *iterere* disse ligningene.

3. Litt matematikk

I dette avsnittet skal vi uttrykke ligningene fra forrige avsnitt ved hjelp av matriser og vektorer. Vi skal også indikere hvilke metoder man bruker for å løse disse ligningene. Matematikken burde være forståelig for lesere med gode gymnaskunnskaper, og er lett forståelig for lesere med et første års kurs i lineær algebra på et universitet, eller en høyskole.

3.1 Ligningene på matriseform. La $A = (R_{ab})$ være matrisen med koeffisienter $R_{ab} = 1/|F_a|$ om b peker mot a og der $R_{ab} = 0$ om b *ikke* peker mot a. La videre $v = (R_a)$ være søylevektoren hvis a'te koordinat er lik R_a. Da kan ligningene PR skrives på *matriseform*:

$$Av = \lambda v.$$

Alle som har lest litt *lineær* algebra vil kjenne igjen dette uttrykket. Vi sier at λ er en *egenverdi* for matrisen A, og at v er en *egenvektor* for matrisen A *tilhørende* egenverdien λ.

Vi påminner om at *egenverdiene* til en $n \times n$-matrise A er røttene til n-tegradspolynomet $\det(tI_n - A)$ i den variable t, der I_n er $n \times n$ identitetsmatrisen. Polynomet $\det(tI_n - A)$ kalles det *karakteristiske* polynomet for A. For hver egenverdi λ til A har ligningen $Av = \lambda v$ i vektoren v løsninger, og løsningene kalles *egenvektorer* tilhørende egenverdien λ.

3.2 Eksempel. I Eksempel 2.7 ovenfor vil matrisen være

$$A = \begin{pmatrix} 0 & 0 & \frac{1}{2} \\ 1 & 0 & \frac{1}{2} \\ 0 & 1 & 0 \end{pmatrix}$$

I Eksempel 2.8 er matrisen

$$A = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix},$$

og i Eksempel 2.9 er matrisen

$$A = \begin{pmatrix} 0 & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix}.$$

Matrisen A har ikke-negative koordinater R_{ab}. Når den tilfredsstiller visse tilleggsbetingelser, som vi skal beskrive nedenfor, finnes det en vakker *klassisk* teori for egenverdier og egenvektorer. For å beskrive de resultatene vi skal bruke innfører vi først litt terminologi.

3.3 Definisjon. En matrise $A = (a_{ij})$ kalles *ikke-negativ* om $a_{ij} \geq 0$ for alle i, j, og den kalles *positiv* om $a_{ij} > 0$ for alle i, j.

3.4 Definisjon. En $n \times n$-matrise er *redusibel* om den er på *blokkformen*

$$\begin{pmatrix}
* & \cdots & * & * & * & \cdots & * & * & * & \cdots & * & * & * & \cdots & * \\
\vdots & & & & & & & & & & & & & & \vdots \\
* & \cdots & * & * & * & \cdots & * & * & * & \cdots & * & * & * & \cdots & * \\
* & \cdots & * & 0 & * & \cdots & * & 0 & * & \cdots & * & 0 & * & \cdots & * \\
* & \cdots & * & * & * & \cdots & * & * & * & \cdots & * & * & * & \cdots & * \\
\vdots & & & & & & & & & & & & & & \vdots \\
* & \cdots & * & * & * & \cdots & * & * & * & \cdots & * & * & * & \cdots & * \\
* & \cdots & * & 0 & * & \cdots & * & 0 & * & \cdots & * & 0 & * & \cdots & * \\
* & \cdots & * & * & * & \cdots & * & * & * & \cdots & * & * & * & \cdots & * \\
\vdots & & & & & & & & & & & & & & \vdots \\
* & \cdots & * & * & * & \cdots & * & * & * & \cdots & * & * & * & \cdots & * \\
* & \cdots & * & 0 & * & \cdots & * & 0 & * & \cdots & * & 0 & * & \cdots & * \\
* & \cdots & * & * & * & \cdots & * & * & * & \cdots & * & * & * & \cdots & * \\
\vdots & & & & & & & & & & & & & & \vdots \\
* & \cdots & * & * & * & \cdots & * & * & * & \cdots & * & * & * & \cdots & *
\end{pmatrix},$$

det vil si, den inneholder en *blokk* av nuller, der 0'ene står i rekkene $i_1, \ldots, i_p$ og i søylene $j_1, \ldots, j_q$, og der $p + q = n$ med n lik størrelsen av matrisen og

$\{i_1, \ldots, i_p, j_1, \ldots, j_q\} = \{1, 2, \ldots, n\}$. Ekvivalent er den redusibel om vi ved samtidig å endre rekkefølgen av rekke- og søylenummerene $1, 2, \ldots, n$ kan overføre den til formen

$$\begin{pmatrix} B & 0 \\ C & D \end{pmatrix}$$

der B er en $p \times p$-matrise, D er en $q \times q$-matrise, og 0-matrisen i øverste høyre hjørne er en $p \times q$-matrise. En matrise som ikke er redusibel kaller vi *irredusibel*.

Vi skal nu skrive ned resultatene om positive og ikke-negative matriser som danner bakgrunnen for å løse ligninger ved iterasjon, og som vi har nevnt flere ganger ovenfor:

3.5 Theorem. (Perron 1907) *Om A er en positiv matrise har den en positiv egenverdi $\lambda(A)$ som er en enkel rot i det karakteristiske polynomet, og som er ekte større enn absoluttverdien for de andre egenverdiene. Til $\lambda(A)$ svarer en egenvektor som er positiv.*

3.6 Theorem. (Frobenius 1908-1912) *En irredusibel ikkenegativ matrise A har en positiv egenverdi $\lambda(A)$ som er en enkel rot i det karakteristiske polynomet, og som er ekte større enn absoluttverdien for de andre egenverdiene. Til $\lambda(A)$ svarer en egenvektor som er positiv.*

Disse viktige resultatene inngår oftest ikke i et første års kurs i lineær algebra. Bevisene er vakre, men ganske flokete, og vi gir dem ikke her. (For bevis, se for eksempel [2].) Vi skal heller ikke gi bevis for følgende to resultater som også ofte er nyttige, men som har mye enklere bevis (se for eksempel [3]):

3.7 Lemma. *Om A er en ikke-negativ irredusibel $n \times n$-matrise så vil $(I_n + A)^{n-1}$ være positiv, der I_n er $n \times n$-enhetsmatrisen.*

3.8 Proposisjon. *Om A er en ikke-negativ irredusibel matrise slik at $a_{ii} > 0$ for minst en i så har A en positiv egenverdi som er ekte større enn absoluttverdien til de andre egenverdiene.*

Vi vil nu vise hvorfor Perrons og Frobenius setninger er så viktige for å finne egenverdier for matriser som er positive, eller ikke-negative og irredusible. Ettersom bevisene er enkle og resultatene så nyttige tar vi med bevisene. Først påminner vi om noen kjente begreper fra teorien for matriser.

3.9 Definisjon. La $A = (a_{ij})$ være en $m \times n$-matrise. Vi kaller $n \times m$-matrisen ${}^{\mathrm{t}}A = (a_{ji})$ den *transponerte* matrisen til A, og matrisen $\overline{A} = (\overline{a}_{ij})$ den *konjugerte* matrisen til A, der $\overline{a}_{ij}$ er den kompleks konjugerte til a_{ij}. Vi sier at en $n \times n$-matrise U er *unitær* om

$${}^{\mathrm{t}}\overline{U}U = I_n.$$

Vi uttrykker også at U er unitær ved å si at rekkene, eller ekvivalent søylene, i U er *ortonormale*.

3.10 Schurs metode. For hver $n \times n$-matrise $A = (a_{ij})$ kan vi finne en unitær matrise U slik at

$${}^{\mathrm{t}}\overline{U}AU = B$$

er *øvre triangulær*, det vil si $B = (b_{ij})$ og $b_{ij} = 0$ når $i > j$. Dette viser vi lett ved induksjon etter n som følger:

La λ_1 være en egenverdi for A og la u_1 være en egenverdi for λ_1 av lengde 1. Ved den kjente Gram–Schmidts ortogonaliseringsprosess, som inngår i alle kurser i lineær algebra, kan vi finne ortonormale vektorer $u_1, u_2, \dots, u_n$. La U' være den unitære matrisen med søyler $u_1, u_2, \dots, u_n$. Da vil

$$AU' = \begin{pmatrix} \lambda_1 u_{11} & c_{12} & \cdots & c_{1n} \\ \vdots & \vdots & \vdots & \vdots \\ \lambda_1 u_{n1} & c_{n2} & \cdots & c_{nn} \end{pmatrix},$$

for noen tall c_{ij}, og der $u_{11}, u_{21}, \dots, u_{n1}$ er koeffisientene for u. Ettersom vektorene $u_1, u_2, \dots, u_n$ er ortonormale får vi derfor at

$${}^{\mathrm{t}}\overline{U'}AU' = \begin{pmatrix} \lambda_1 & b_2 & \cdots & b_n \\ 0 & & & \\ \vdots & & A_1 & \\ 0 & & & \end{pmatrix},$$

for noen tall $b_2, \dots, b_n$, der A_1 er en $(n-1) \times (n-1)$-matrise. Antar vi at Schurs metode holder for $(n-1) \times (n-1)$ matriser kan vi finne en unitær $(n-1) \times (n-1)$-matrise U_1 slik at ${}^{\mathrm{t}}\overline{U_1}A_1U_1 = B_1$ er en øvre triangulær $(n-1) \times (n-1)$ matrise. Sett

$$U_1' = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & & & \\ \vdots & & U_1 & \\ 0 & & & \end{pmatrix}.$$

Da er U_1' en unitær $n \times n$-matrise og

$$\begin{aligned} {}^{\mathrm{t}}\overline{U_1'} \begin{pmatrix} \lambda_1 & b_2 & \cdots & b_n \\ 0 & & & \\ \vdots & & A_1 & \\ 0 & & & \end{pmatrix} U_1' &= \begin{pmatrix} \lambda_1 & d_2 & \cdots & d_n \\ 0 & & & \\ \vdots & & {}^{\mathrm{t}}\overline{U_1}A_1U_1 & \\ 0 & & & \end{pmatrix} \\ &= \begin{pmatrix} \lambda_1 & d_2 & \cdots & d_n \\ 0 & & & \\ \vdots & & B_1 & \\ 0 & & & \end{pmatrix} = B \end{aligned}$$

er øvre diagonal. Om vi setter $U = U'U_1'$ får vi derfor

$${}^{\mathrm{t}}\overline{U}AU = {}^{\mathrm{t}}\overline{U_1}{}^{\mathrm{t}}\overline{U'}AU'U_1' = B$$

som vi ville vise.

3.11 Proposisjon. *La A være en matrise og ρ et positivt tall som er større enn absoluttverdien til egenverdiene til A. For hver vektor v vil da*

$$\lim_{n\to\infty} \frac{A^n v}{\rho^n} = 0.$$

Bevis. Ved Schurs metode finner vi en unitær matrise U slik at ${}^{\mathrm{t}}\overline{U}AU$ er øvre triangulær. Da består diagonalen til ${}^{\mathrm{t}}\overline{U}AU$ av egenverdiene til A. La D være diagonalmatrisen med koordinater $d_{ii} = \varepsilon^{i-1}$. Da vil matrisen $D^{-1\mathrm{t}}\overline{U}AUD$ være øvre diagonal, og alle koordinatene ovenfor diagonalen vil være et produkt med en positiv potens av ε. Det følger at om vi velger ε liten så vil $0 = \lim_{n\to\infty}(D^{-1\mathrm{t}}\overline{U}AUD)^n w/\rho^n$ for hver vektor w. Men vi har at

$$\lim_{n\to\infty}\frac{(D^{-1\mathrm{t}}\overline{U}AUD)^n w}{\rho^n} = \lim_{n\to\infty}\frac{D^{-1\mathrm{t}}\overline{U}A^nUDw}{\rho^n} = D^{-1\mathrm{t}}\overline{U}\left(\lim_{n\to\infty}\frac{A^nUDw}{\rho^n}\right).$$

Derfor vil $\lim_{n\to\infty} A^nUDw/\rho^n = 0$, og velger vi $w = D^{-1\mathrm{t}}\overline{U}v$ får vi at

$$\lim_{n\to\infty}\frac{A^n v}{\rho_n} = 0.$$

3.12 Iterasjon. La A være en matrise og u en ikke null vektor. Vi definerer en følge av vektorer $v_0, v_1, v_2, \ldots$ rekursivt ved

$$v_0 = \frac{u}{|u|}, \quad v_1 = \frac{Av_0}{|Av_0|}, \quad v_2 = \frac{Av_1}{|Av_1|}, \ldots,$$

det vil si

$$v_{n+1} = \frac{Av_n}{|Av_n|} \qquad \text{for} \quad n = 0, 1, 2, \ldots.$$

Da vil

$$v_n = \frac{A^n u}{|A^n u|} \qquad \text{for} \quad n = 0, 1, 2, \ldots.$$

Spesielt har alle vektorene v_n lengde 1. Vi skal gi noen enkle betingelser for at følgen $v_0, v_1, v_2, \ldots$ *konvergerer* mot en vektor v som er en egenvektor for A. Det der dette vi mener med å *løse ligningen* $Av = \lambda v$ ved *iterasjon.*

3.13 Setning. *La A være en matrise med en positiv egenverdi $\lambda(A)$ som er en enkel rot i det karakteristiske polynomet til A, og som er større en absoluttverdien av de andre egenvektorene til A. For «nesten alle» vektorer u vil da følgen av vektorer $v_0 = u/|u|$, $v_1 = Au/|Au|$, $v_2 = A^2u/|A^2u|, \ldots$ konvergere og*

$$\lim_{n\to\infty} v_n = \lim_{n\to\infty}\frac{A^n u}{|A^n u|} = v,$$

der v er en egenvektor for A av lengde 1 som tilhører egenverdien $\lambda(A)$.

Bevis. Ettersom $\lambda(A)$ er en enkel rot i minimalpolynomet kan vi finne en basis $u_1, u_2, \ldots, u_n$ for vektorrommet slik at $v_1 = u_1/|u_1|$ er en egenvektor for matrisen A av lengde 1 tilsvarende egenverdien $\lambda(A)$, og slik at A med hensyn til denne basisen kan skrives på formen

$$A = \begin{pmatrix} \lambda(A) & 0 & \cdots & 0 \\ 0 & & & \\ \vdots & & A' & \\ 0 & & & \end{pmatrix},$$

der A' er en $(n-1)\times(n-1)$-matrise. Skriver vi $u = a_1u_1 + a_2u_2 + \cdots + a_nu_n$ og setter $u' = a_2u_2 + a_3u_3 + \cdots + a_nu_n$ så får vi at

$$A^n u = \begin{pmatrix} \lambda(A)^n a_1 \\ (A')^n u' \end{pmatrix}.$$

Bortsett fra egenverdien $\lambda(A)$ har A og A' samme egenverdier. Det følger derfor av Proposisjon 3.11 at

$$\lim_{n\to\infty} \frac{(A')^n u'}{\lambda(A)^n} = 0$$

og derfor at

$$\lim_{n\to\infty} \frac{A^n u}{\lambda(A)^n} = \begin{pmatrix} a_1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = a_1u_1.$$

Videre følger det at

$$\lim_{n\to\infty} \frac{|A^n u|}{\lambda(A)^n} = |a_1|\,|u_1| = |a_1|.$$

Om $|a_1| \neq 0$, det vil si, for alle vektorer som ikke ligger i vektorrommet *utspent* av $u_2, u_3, \ldots, u_n$ får vi

$$\begin{aligned} \lim_{n\to\infty} \frac{A^n u}{|A^n u|} &= \lim_{n\to\infty} \left(\frac{A^n u}{\lambda(A)^n} \frac{\lambda(A)^n}{|A^n u|} \right) \\ &= \lim_{n\to\infty} \left(\frac{A^n u}{\lambda(A)^n} \right) \lim_{n\to\infty} \left(\frac{\lambda(A)^n}{|A^n u|} \right) = \frac{a_1}{|a_1|} u_1 = \pm u_1. \end{aligned}$$

3.14 Bemerkning. Vi har med vilje brukt det litt upresise begrepet «nesten alle vektorer» i Setningen. Som vi ser av beviset for Setningen kan vi mer presist si «for alle vektorer som ikke ligger i et underrom av dimensjon $n-1$».

4. Tilbake til PageRank

4.1 Matematikk og praksis. Vi har tidligere bemerket at matrisen $A = (R_{ab})$ er ganske spesiell. I søylen b er det nøyaktig $|F_b|$ koordinater som ikke er null, og alle ikke null koordinater er lik $1/|F_b|$. Videre har hver rekke omkring $|F_b|$ koordinater som er forskjellig fra null. I praksis er $|F_b|$ omtrent lik 11, så «nesten alle» koordinatene i $A = (R_{ab})$ er like null. Derfor er det ganske lett å regne ut vektorene $v_n = A^n u/|A^n u|$ for en vilkårlig vektor u. Bestemmer man kvotientene $A^n u/|A^n u|$ for $n = 1, 2, \ldots$ så viser det seg at når n nærmer seg 50 så skiller vektorene $A^n u/|A^n u|$ og $A^{n+1}u/|A^{n+1}u|$ seg veldig lite. Det er derfor rimelig å bruke vektoren $A^n u/|A^n u|$ for en $n \geq 50$ som en *rangvektor*. Som vi merker av den fabelaktige prestasjonsevnen til søkemaskinene som bruker PageRank fungerer

dette aldeles utmerket i praksis. Tidsmessig tar det bare noen timer å utføre de 50 iterasjonene på en større datamaskin.

Matematikken vi skisset i forrige seksjon er relevant for å sannsynliggjøre at følgen $u/|u|, Au/|Au|, A^2u/|A^2u|, \dots$ konvergerer mot en egenvektor. For å bruke Setning 3.13 er det imidlertid nødvendig at forutsetningen om at $A = (R_{ab})$ har en positiv egenverdi som er en enkel rot i det karakteristiske polynomet, og som er større enn absoluttverdien av de andre egenverdiene. Om A var positiv ville dette følge av Perrons Setning 3.5, men som vi har sett er A langt fra positiv. For å bruke Frobenius Setning 3.6 må vi, blandt annet, vite at matrisen er irredusibel. Dette er langt fra klart. At den er irredusibel betyr, litt upresist, at det ikke finnes grupper av nettsider som bare henviser til hverandre. Det er blandt annet dette vi prøver å unngå ved å modifisere ligningene PR, og ved å ta bort hengende nettsider. Om A er irredusibel må vi, for å bruke Frobenius Setning, dessuten vite at den positive egenverdien $\lambda(A)$ er ekte større enn absoluttverdien for de andre egenverdiene. Dette holder, ved Proposisjon 3.8, om R_{aa} er forskjellig fra 0 for noe a. Dette vet vi ikke holder. Derimot vil dette bli tilfredsstilt om vi modifiserer ligningssystemet ved å betrakte hver nett som lenket til seg selv. En slik modifikasjon gjør at vi også kan bruke Lemma 3.7 som sier at $(I_n + A)^{n-1}$ er positiv, og dermed Perrons Setning. Dette er imidlertid upraktisk ettersom n er så stor.

Man kan spørre seg om disse resonnementene er interssante eller nødvendige, ettersom iterasjon fungerer i praksis. Svaret er at vi klarer oss uten resonnementene, men at det er takket være matematikken, og resultatene vi har nevnt, at vi i det hele tatt skulle komme på tanken å løse ligningene ved iterasjon. Dette er en annen grunn til at matematikken er så fundamental. Den gjør det mulig å sette opp de rette modellene, og den antyder hvordan modellene skal analyseres.

Bibliografi

[1] Sergey Brin and Larry Page, *The PageRank Citation Ranking*: *Bringing order to the web*, google search engine, `http://google.stanford.edu`

[2] F. R. Gantmacher *Applications of the theory of matrices*, Interscience publishers, inc., London–New York 1959.

[3] Roger A. Horn & Charles R. Johnson, *Matrix Analysis*, Cambridge Univ. Press 1985.