Algorithmique numérique

Eigen and singular decompositions

Gilles Marait & The IS104 team

12/04/2024

Bordeaux INP AQUITAINE

Plan du cours

Reminder about complex arithmetic

Dot product

In a hermitian space, the dot product of two vectors x and $y \in \mathbb{C}^n$ is:

$$x \cdot y = \overline{x^T} y$$

Let's call conjugate transpose (trans-conjugué) or adjoint of x, written x^* :

$$\boxed{x^* = \overline{x^T}}$$

Remark: this definition allows to keep the dot product's good properties, in particular: $x^* x = ||x||_2^2$, whereas $x^T x$ is complex in general.

Unitary matrices

The equivalent of orthogonales matrices in complex arithmetic are unitary matrices, which have similar properties:

$$Q^* Q = Id$$

$$Q^* = Q^{-1}$$

...

Plan

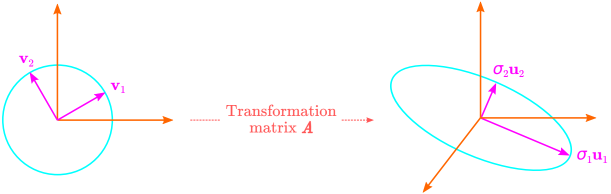
Plan

1. Singular value decomposition
   - Introduction to the SVD
   - SVD definition
   - Comparison between SVD and Eigenvalue Decomposition
   - Properties of the SVD

A geometric observation

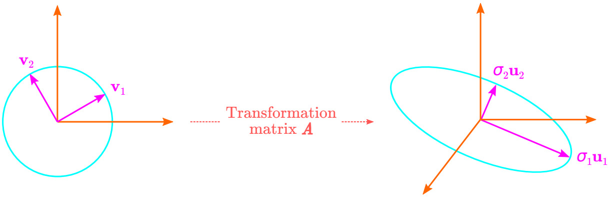
The image of the unit sphere in $\mathbb{R}^n$ under any $m \times n$ matrix is a hyperellipse in $\mathbb{R}^m$.

With a matrix A of dimensions 3×2, let's take the image of two orthogonal vectors with norm 1: v_1 and v_2.



Source : https://pabloinsente.github.io/intro-linear-algebra, Pablo Caceres (modified)

Definitions



Définitions

Assuming (for now) $m \geq n$, let's call:

- singular values : $\{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ the lengths of the semiaxes.
- left singular vectors: $\{u_1, u_2, \ldots, u_n\}$ the directions of the semiaxes (unit vectors).
- right singular vectors: $\{v_1, v_2, \ldots, v_n\}$ the vectors of the orthonormal basis of $\mathbb{R}^n$.

Yes, it's normal that right singular vectors are on the left hand side of the picture and vice versa

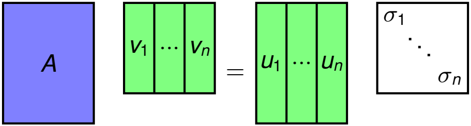
Reduced SVD

We have $\forall j \in [1, \ldots, n]$:

$$Av_j = \sigma_j u_j$$

In matrix format:

$$AV = \hat{U}\hat{\Sigma}$$



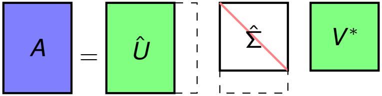
- A an $m \times n$ matrix
- $\hat{\Sigma}$ is $n \times n$ diagonal, with values > 0
- $\hat{U}$ is $m \times n$ with orthonormal columns
- V is $n \times n$, unitary

  Multiplying of the right by V^*, we can write the reduced singular value decomposition:

$$A = \hat{U}\hat{\Sigma}V^*$$

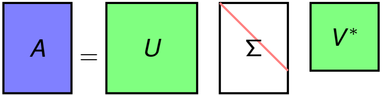
Full SVD

In the same way we had a reduced and a full QR factorization, we can complete the basis of $\hat{U}$ to obtain a unitary matrix U:



Which can be written:

$$\boxed{A = U \Sigma V^*}$$



- We now have a unitary U !
- Storage is more expensive

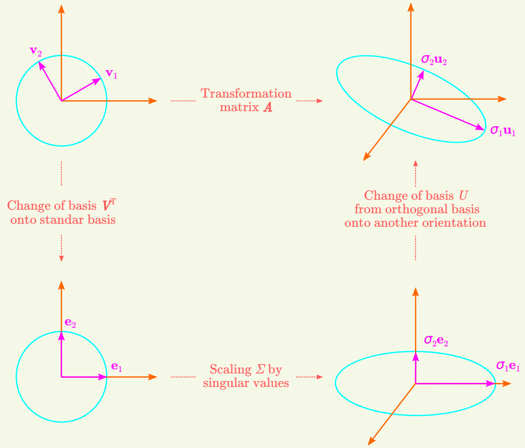
History

Independant discovery by Eugenio Beltrami (Italian) in 1873 and Camille Jordan (French) in 1874.

Description géométrique

Example

With A of dimensions 3×2 :



Source : https://pabloinsente.github.io/intro-linear-algebra, Pablo Caceres (modified)

Bordeaux INP AQUITAINE

Plan

Formal Definition

Let $A \in \mathbb{C}^{m\times n}$, not necessarily with $m \geq n$, without any particular properties.
We define the singular value decomposition (SVD) of A the following decomposition:

$$\boxed{A = U\Sigma V^*}$$

Where:

- U is unitary $\mathbb{C}^{m\times m}$
- V is unitary $\mathbb{C}^{n\times n}$
- Σ is diagonal $\mathbb{R}^{m\times n}$
  Moreover, the diagonal values of Σ are positive or null, and sorted in decreasing order:

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p \geq 0, \text{ where } p = min(m, n)$$

French

The French translation of SVD is *décomposition en valeurs singulières*.

Existence and uniqueness of the SVD

Theorems

- For every matrix $A \in \mathbb{C}^{m\times n}$ there exists a singular value decomposition.
- Furthermore, the singular values $\{\sigma_j\}$ are uniquely determined.
- If A is square and the σ_j are distinct, the left and right singular vectors $\{u_j\}$ and $\{v_j\}$ are uniquely determined up to the complex signs (*i.e.* a factor $z \in \mathbb{C}$ such that $|z| = 1$).

A change of basis

The SVD makes it possible for us to say that every matrix is diagonal, if only one uses the proper bases. Let's consider:

$$b = A.x \qquad \text{and the SVD of } A : \qquad A = U\Sigma V^*$$

Now we write:

$$b' = U^* b \qquad \text{et} \qquad x' = V^* x$$

The relation becomes:

$$\begin{aligned} b &= A.x \\ U^*.b &= U^*.A.x \\ \underbrace{U^*.b}_{b'} &= \underbrace{U^*.U}_{Id}.\Sigma.\underbrace{V^*.x}_{x'} \\ b' &= \Sigma x' \end{aligned}$$

Plan

Vocabulary

We want to compare the two decompositions. Let's start with some vocabulary.

Français	Anglais
Valeur propre	*Eigenvalue*
Vecteur propre	*Eigenvector*
Décomposition en valeurs propres	*Eigendecomposition* or *Eigenvalue decomposition* or *EVD*

Valeur singulière	*Singular value*
Vecteur singulier à droite	*Right singular vector*
Décomposition en valeurs singulières	*Singular value decomposition* or *SVD*

EVD definition

EVD definition

Let $A \in \mathbb{C}^{m \times m}$ be a diagonalisable matrix and $X \in \mathbb{C}^{m \times m}$:

$$A = X \Lambda X^{-1}$$

where Λ is a diagonal matrix $m \times m$ whose values are the eigenvalues of A.

A change of basis

It is also possible to choose an appropriate basis to manipulate diagonal matrices.

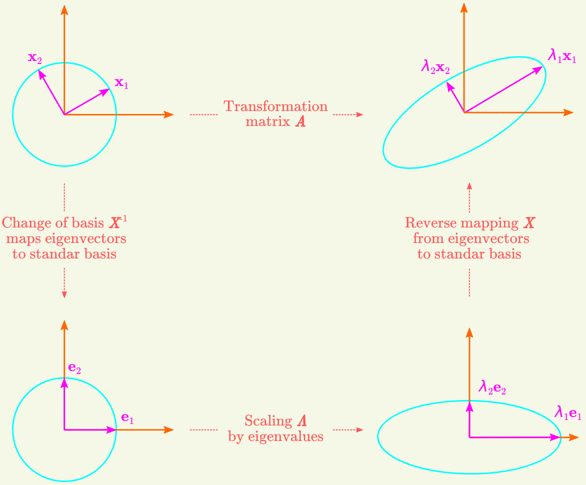
With A diagonalisable: $A = X \Lambda X^{-1}$, we consider $b = A.x$ as we did previously.

$$b' = X^{-1}.b \qquad \text{and} \qquad x' = X^{-1}.x$$

to obtain: $b = A.x \iff b' = \Lambda x'$

A geometric illustration

2D example



Source : https://pabloinsente.github.io/intro-linear-algebra, Pablo Caceres (modified)

SVD versus EVD

	SVD	EVD
Formula	$A = U\Sigma V^*$	$A = X\Lambda X^{-1}$
Number of bases	2	1 (endomorphisme)
Are the bases orthonormal ?	Yes	No
A decomposition always exists ?	Yes	No
Applications	Using A or A^{-1} Data analysis on A $\dots$	Computing iterations A^k e^{tA} Polynomials in A $\dots$

A relation SVD - EVD

A link between eigenvalues and singular values

The singular values of A which are not null are the square roots of the eigenvalues of A^*A or AA^*.

Indeed we can write:

$$\begin{aligned} A^*A &= (U\Sigma V^*)^*(U\Sigma V^*) \\ &= V\Sigma^* U^* U\Sigma V^* \\ &= V(\Sigma^*\Sigma)V^* \end{aligned}$$

- V is an eigenvectors basis of A^*A
- U is an eigenvectors basis of AA^*
- $\sigma_i^2 = \lambda_i$

Remark

Warning: it's generally a bad method to compute singular values: $\kappa(A^*A) \approx \kappa(A)^2$.

Plan

SVD and rank

Theorem

The rank of A is r, the number of nonzero singular values.

$$A = U.\begin{pmatrix} \sigma_1 & & & & & \\ & \ddots & & & & \\ & & \sigma_r & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \\ & & & & & \end{pmatrix}.V^*$$

Proof

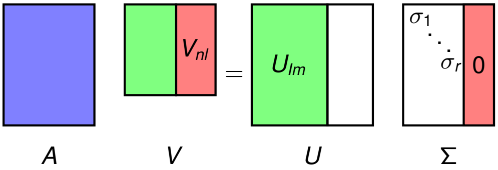
$$A = \underbrace{U}_{\text{full rank}} \Sigma \underbrace{V^*}_{\text{full rank}}$$

$\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r > 0$ so $rank(A) = rank(\Sigma) = r$.

Image and null space

Theorem

$$range(A) = span(u_1, \ldots, u_r)$$
$$null(A) = span(v_{r+1}, \ldots, v_n)$$



Preuve

Consequence of :

$$range(\Sigma) = span(e_1, \ldots, e_r) \subseteq \mathbb{C}^m$$
$$null(\Sigma) = span(e_{r+1}, \ldots, e_n) \subseteq \mathbb{C}^n$$

Norms

Induced matrix norm

Using the matrix norm induced by norm-2:

$$|||A||| = \sigma_1$$

Frobenius norm

The Frobenius norm of A is defined:

$$||A||_F = \sqrt{\sum_{i=1}^{m} \left(\sum_{j=1}^{n} |a_{i,j}|^2 \right)}$$

Which is also equal to:

$$||A||_F = \sqrt{\sigma_1^2 + \sigma_2^2 + \ldots + \sigma_r^2}$$

Determinant, Low rank approximation

Determinant

For $A \in \mathbb{C}^{m \times m}$ (like for eigenvalues):

$$|det(A)| = \prod_{i=1}^{m} \sigma_i$$

Low rank approximation

(In French: *approximation de rang faible*).
A is the sum of r matrices of rank 1:

$$A = \sum_{j=1}^{r} \sigma_j u_j v_j^*$$

Application: image compression (project) !

Compression

Orginal picture

Image compressed by SVD (rank 50)

Computing the SVD

How to obtain the singular value decomposition ?

Do the project !

From now on, we will treat the computation of the eigendecomposition, which is similar the SVD computation.

Plan

Plan

2. Eigendecomposition
   - Problem presentation
   - Difficulty of the problem

Problem presentation

Let's consider $A \in \mathbb{C}^{n\times n}$, diagonalisable. We want to get the eigenvalues λ and their associated eigenvectors x_λ such that:

$$\begin{cases} Ax_\lambda = \lambda x_\lambda \\ x_\lambda \neq 0 \end{cases}$$

Tony Stark and EVD

In *Avenger : endgame*, Tony Stark asks his assistant, the artificial intelligence F.R.I.D.A.Y. :

"F.R.I.D.A.Y., compute this eigenvalue"

It corresponds to a calculus of quantum physics on a Möbius strip topology, in order to travel in time !

Indeed, in quantum physics, when one wants to search for eigenstates using the time-independent Schrödinger equation:

$$\hat{H}\psi = E\psi$$

where E is the energy, eigenvalue of the hamiltonian operator $\hat{H}$, and ψ the wave function of the particule in an eigenstate !

Plan

2 Eigendecomposition
- Problem presentation
- Difficulty of the problem

Difficulty of the problem

Poll: do you think there exists a direct method for the computation of eigenvalues ?

A yes, like for solving linear systems

B yes, but they are too costly and therefore not used

C no, there are not

Difficulty of the problem

The problem can be solved by finding the roots of a polynomial of degree n: the characteristic polynomial.

$$P_A(z) = det(zId - A)$$
$$P_A(z) = z^n + \alpha_{n-1} z^{n-1} + ... + \alpha_0$$

Reciprocally if we have such a polynomial $P(z)$, there exists a companion matrix whose characteristic polynomial is:

$$A_P = \begin{pmatrix} 0 & 0 & \cdots & 0 & -\alpha_0 \\ 1 & 0 & \cdots & 0 & -\alpha_1 \\ 0 & 1 & \cdots & 0 & -\alpha_2 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \cdots & 1 & -\alpha_{n-1} \end{pmatrix}$$

So the two problems are **equivalent** !

EVD computation $\Longleftrightarrow$ Computing the roots of a polynomial

Difficulty of the problem

Is there a direct method to compute the roots of a polynomial ?

Abel (1802 - 1829),
norvegian mathematician,
demonstrated that there is no generic formula to calculate the roots of a polynomial of degree ≥ 5 using the usual operations $(+, -, *, /, \sqrt{})$
(**Abel-Ruffini theorem**)

Galois (1811 - 1832),
french mathematician, confirms and generalises the result in his **Galois theory**.

See chapter 4 for the computation of the roots of a polynomial in practice.

Difficulty of the problem

Poll: do you think there exists a direct method for the computation of eigenvalues ?

- A ~~yes, like for solving linear systems~~
- B ~~yes, but they are too costly and therefore not used~~
- C no, there are not

Hopefully, there exists efficient iterative methods.

Plan

Plan

The Jacobi method

We shall start with a first iterative method: the Jacobi method.

Principal of the method

In the real symmetric case, the Jacobi method aims at converging the a diagonal matrix using rotations.

$$O_k^T.A.O_k \underset{k\to\infty}{\longrightarrow} D$$

where:

- D is a diagonal matrix.
- the O_k are rotation matrices (therefore orthogonal).

The Jacobi method

Let p be a row index, q a column index and $\theta \in]-\pi;\pi]$ a real.
Then we have:
where:

- $O_{p,p} = O_{q,q} = \cos(\theta)$
- $O_{p,q} = \sin\theta$
- $O_{q,p} = -\sin\theta$

The Jacobi method

Example of a matrix Q_k

Let's consider the transformation matrix A, restricted to the rows p and q. We have

$$B = O^T.A.O$$

$$B = \begin{pmatrix} b_{p,p} & b_{p,q} \\ b_{q,p} & b_{q,q} \end{pmatrix}$$

$$= \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} . \begin{pmatrix} a_{p,p} & a_{p,q} \\ a_{q,p} & a_{q,q} \end{pmatrix} . \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$$

Choose $\theta \in]-\frac{\pi}{4}; \frac{\pi}{4}[$ so that $b_{p,q}$ and $b_{q,p}$ are null. If $b_{p,q}$ and $b_{q,p}$ are null, then $\theta = 0$, otherwise,

$$b_{p,q} = b_{q,p} = a_{p,q} cos(2\theta) + \frac{a_{p,p} - a_{q,q}}{2} sin(2\theta)$$

$$\Rightarrow cotan(2\theta) = \frac{a_{q,q} - a_{p,p}}{2a_{p,q}}$$

The Jacobi method

Iterations

Let $A_0 = A$ and $O_0 = Id$, we construct a sequence such that:

$$\begin{cases} A_{k+1} = O^T.A_k.O \\ O_{k+1} = O_k.O \end{cases}$$

Steps of the algorithm:

- choose a nonzero extradiagonal element $a_{p,q}$ of A;
- construct the rotation matrix O adapted to this element;
- construct the matrix $O^T.A.O$

The Jacobi method

Optimizations

Only the rows and columns with index p and q of the matrix A are modified at each iteration:

$$\begin{cases} b_{i,j} = a_{i,j} & \text{si } i,j \neq p,q \\ b_{p,i} = a_{p,i}\cos\theta - a_{q,i}\sin\theta & \text{si } i \neq p,q \\ b_{q,i} = a_{p,i}\sin\theta - a_{q,i}\cos\theta & \text{si } i \neq p,q \\ b_{p,p} = a_{p,p} - a_{p,q}\tan\theta \\ b_{q,q} = a_{p,p} + a_{p,q}\tan\theta \\ b_{p,q} = b_{q,p} = 0 \end{cases}$$

Summary

- Choosing the pivot: extradiagonal element of maximal norm $\Rightarrow$ ensures convergence
- Complexity: $\mathcal{O}(n)$ at each step

Plan

3 Computing eigenvalues

- The Jacobi method
- Transformations to simpler matrices
- Dichotomous method
- The QR algorithm

Transformations to simpler matrices

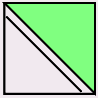
With a direct algorithm, we can however simplify the shape of the matrix:

- EVD of a symmetric matrix: reduction to a tridiagonal matrix.
- EVD of a non symmetric matrix: reduction to a Hessenberg matrix.
- SVD : reduction to a bidiagonal matrix (see projet)

Hessenberg matrices

It is an *almost triangular* matrix.

An upper Hessenberg matrix is a upper triangular matrix with one sub-diagonal.



Transformations to simpler matrices

We transform A using a direct method and then an iterative method.

Decomposition Matrix A	EVD Non symmetric	EVD Symmetric	SVD (any matrix)
Direct method	$\downarrow$	$\downarrow$	$\downarrow$
Intermediate shape	Hessenberg	Tridiagonal	Bidiagonal
Iterative method	$\downarrow$	$\downarrow$	$\downarrow$
Final shape	Triangular	Diagonal	Diagonal

From the final shape, we can read the eigenvalues / singular values on the diagonal.

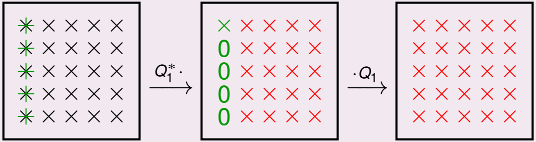
Reduction to Hessenberg or tridiagonal

In order to keep the eigenvalues throughout the process, we apply unitary transformations (rotations or reflections): changes of basis using unitary matrices Q:

$$A \to Q^* A Q$$

A bad idea (a priori)

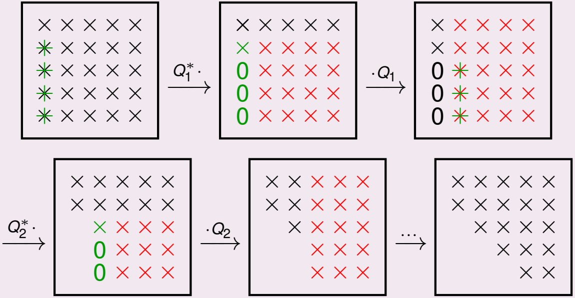
Using a Housholder matrix to get a triangular matrix:



All rows are modified when multiplying on the left by Q_1^*, so all columns are modified when multiplying by Q_1 on the right.

Reduction to Hessenberg or tridiagonal

A good idea



Using a Housholder matrix to get a Hessenberg matrix:

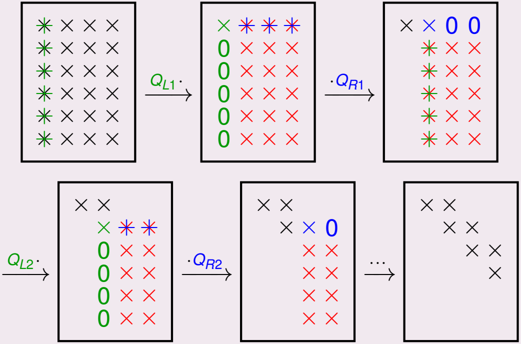
In the symmetric case, the multiplication of Q_k also nullifies coefficients of row k $\rightarrow$ the final shape is tridiagonal.

Reduction to bidiagonal form for SVD

For the computation of SVD, the problem is different because we construct 2 distinct bases. Thus, it is not necessary to apply two inverse unitary transformations on the right and on the left

$$A \to Q_L \cdot A \cdot Q_R$$

Bidiagonalisation



Plan

Dichotomous method

In the symmetric case, let's consider the tridiagonal matrix we obtained:

$$\begin{pmatrix} b_1 & c_1 & & & \\ c_1 & b_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & c_{n-2} & b_{n-1} & c_{n-1} \\ & & & c_{n-1} & b_n \end{pmatrix}$$

Characteristic polynomial

If we consider A_k the matrix restricted to the k first rows and columns, we have the following characteristic polynomial:

$$\begin{cases} P_0(X) = 1 \\ P_1(X) = b_1 - X \\ P_k(X) = (b_k - X)P_{k-1}(X) - c_{k-1}^2 P_{k-2}(X) \end{cases}$$

Dichotomous method

Let's assume all eigenvalues are distinct:

Properties of the characteristic polynomial P_k

- P_k of degree k
- $\lim_{x \to -\infty} P_k(x) = +\infty$ and $\lim_{x \to +\infty} P_k(x) = (-1)^k \infty$
- if x is a root of P_k, then $P_{k-1}(x)$ and $P_{k+1}(x)$ are not null and of opposite signs

Getting the eigenvalues

Searching for the roots of $P_k(X)$

Searching for the roots

Variation table of those polynomials:

	$-\infty$		e_1	d_1	e_2	b_1	d_2	e_3		$+\infty$
Variations de P_0 :	1					$+$				1
Variations de P_1 :	$+\infty$			$+$		0	$-$			$-\infty$
Variations de P_2 :	$+\infty$		$+$	0	$-$		0	$+$		$+\infty$
Variations de P_3 :	$+\infty$	$+$	0	$-$	0		$+$	0	$-$	$-\infty$

See chapter 4 for the method to locate the roots of this sequence.

Plan

The QR algorithm

An iterative algorithm, simple to implement, using the QR decomposition on the Hessenberg or tridiagonal matrix previously computed:

Algorithme

- $A^{(0)} = A$
- For $k = 1, 2, \ldots$
- $\quad Q^{(k)}, R^{(k)} \leftarrow factoQR(A^{(k-1)})$
- $\quad A^{(k)} \leftarrow R^{(k)}.Q^{(k)}$

Remark

The stopping criterion is not an obvious choice.

The QR algorithm

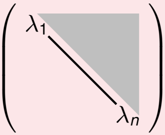
Theorem: convergence of the QR algorithm

If A is diagonalisable and its eigenvalues $\{\lambda_1, \ldots, \lambda_n\}$ are distinct, such that: $|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n|$, then:

$$\begin{cases} \lim_{k\to\infty} a_{i,i}^{(k)} = \lambda_i \\ \lim_{k\to\infty} a_{i,j}^{(k)} = 0 \quad \forall j < i \end{cases}$$

We converge towards a matrix:

$A^{(k)} =$



The QR algorithm

Remark

The algorithm preserves the shape of the matrix (tridiagonal or Hessenberg) at each step.

Remark

The more the eigenvalues are close is modulus, the slower the convergence. Several optimized algorithms use shifting:

$$Q^{(k)}, R^{(k)} \leftarrow factoQR(A^{(k-1)} - \sigma_{(k-1)} Id)$$

where $\sigma_{(k-1)}$ is an estimate of an eigenvalue.

Convergence

Iterations are expensive (reminder: QR decomposition in $\mathcal{O}(n^3)$ in general, although it can be optimized in the tridiagonal case), but the converge est generally fast.

Plan

Rayleigh quotient

Definition

The **Rayleigh quotient** of a vector $x \in \mathbb{R}^n$ not null is:

$$r(x) = \frac{x^T A x}{x^T x}$$

- If x is an eigenvector: $r(x) = \lambda$ is the associated eigenvalue.
- Otherwise, $r(x)$ is the **best eigenvalue** to consider if x is close but not necessarily equal to an eigenvector.

Remark

If x is normalized in norm-2, we have $r(x) = x^T A x$

Plan

4 Obtaining the eigenvectors
- Power iteration
- Inverse iteration with shifting
- Summary
- The 25 billion dollars eigenvector

Power iteration

In French: *Méthode de la puissance itérée*

Idea

There is one particular eigenvector that is easy to identify: the one associated to the biggest eigenvalue (in modulus).

Let $(e_1, \ldots, e_n)$ be a basis of eigenvectors associated to the eigenvalues $\{\lambda_1, \ldots, \lambda_n\}$, sorted by modulus: $|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|$

Let x_0 be a non null vector. Let's take a look at the sequence of $x_k = A^k x_0$. In the basis (e_i):

$$\begin{cases} x_0 = \alpha_1 e_1 + \alpha_2 e_2 + \ldots + \alpha_n e_n \\ x_k = \lambda_1^k \alpha_1 e_1 + \lambda_2^k \alpha_2 e_2 + \ldots + \lambda_n^k \alpha_n e_n \end{cases}$$

If $|\lambda_1| > |\lambda_2|$, asymptotically $||x_k|| \xrightarrow[k \to \infty]{} |\lambda_1|^k$

Power iteration

Algorithme

- Choose $x_0 \neq 0$
- For $k = 1, 2, \ldots$
- $x_{k+1} \leftarrow \frac{Ax_k}{||Ax_k||}$: multiplication by A and normalization
- $\lambda_{k+1} \leftarrow x_{k+1}^T A x_{k+1}$: Rayleigh quotient

Convergence

If we have $|\lambda_1| > |\lambda_2|$, we converge towards the eigenvector associated to λ_1.

Remark

If we perform the algorithm on A^{-1}, we converge towards the eigenvector associated to the smallest eigenvalue of A (in modulus). This is the algorithm called inverse iteration (*méthode de la puissance inverse*). The multiplication $x_{k+1} = A^{-1}x_k$ is replaced by the resolution of the system $Ax_{k+1} = x_k$.

Plan

Inverse iteration with shifting

The methods consists in searching for a particular eigenvector by targetting the associated eigenvalue.

Let $\tilde{\lambda}$ by a value close but distinct of λ (closer than any other eigenvalue).
Let x be a vector, we want to find y solution of the system:

$$(A - \tilde{\lambda}.Id)y = x$$

In the eigenvector basis:

$$\begin{cases} x & = & \alpha_1 e_1 & + & \alpha_2 e_2 & + & \dots & + & \alpha_n e_n \\ y & = & \gamma_1 e_1 & + & \gamma_2 e_2 & + & \dots & + & \gamma_n e_n \\ (A - \tilde{\lambda}.Id)y & = & (\lambda_1 - \tilde{\lambda})\gamma_1 e_1 & + & (\lambda_2 - \tilde{\lambda})\gamma_2 e_2 & + & \dots & + & (\lambda_n - \tilde{\lambda})\gamma_n e_n \end{cases}$$

so we have: $\forall k$

$$\boxed{\gamma_k = \frac{\alpha_k}{\lambda_k - \tilde{\lambda}}}$$

Construction: $x_0 \neq 0$ and a sequence (x_k) :

$$\boxed{(A - \tilde{\lambda}.Id)x_{k+1} = x_k}$$

Inverse iteration with shifting

Component-wise, (x_k) is a geometric sequence with a common ratio: $\frac{1}{\lambda_k - \tilde{\lambda}}$.

Algorithm

- Choose $x_0 \neq 0$
- For $k = 1, \ldots$
- $\quad x_{k+1} \leftarrow$ Solve $(A - \tilde{\lambda}.Id)x_{k+1} = x_k$
- $\quad x_{k+1} \leftarrow \frac{x_{k+1}}{||x_{k+1}||}$
- $\quad \tilde{\lambda} \leftarrow \ x_{k+1}^T A x_{k+1}$: Rayleigh quotient (optional)

Remarks

- It's simply the iterative power method on the matrix $(A - \tilde{\lambda}.Id)^{-1}$
- We can perform once the LU decomposition of $A - \tilde{\lambda}.Id$, and then perform only the triangular solves $L.U.x_{k+1} = x_k$ at each iteration
- We can modify $\tilde{\lambda}$ at the last step of the algorithme, but then we have to compute the LU factorization again with the new matrix for the next solve.

Plan

Convergence summary

	Method	Cost	Convergence $\|\|e_{k+1}\|\|/\|\|e_k\|\|$
Power iteration	$x_{k+1} = Ax_k$	kn^2	$\|\lambda_2\|/\|\lambda_1\|$
Inverse iteration	$Ax_{k+1} = x_k$	$n^3 + kn^2$	$\|\lambda_n\|/\|\lambda_{n-1}\|$
Inverse iteration with shifting	$(A - \tilde{\lambda} I)x_{k+1} = x_k$	$n^3 + kn^2$	$\frac{\|\lambda_c - \tilde{\lambda}\|}{\|\lambda_{c2} - \tilde{\lambda}\|}$

With:

- λ_1: the biggest eigenvalue (in modulus)
- λ_2: the second biggest eigenvalue (in modulus)
- λ_n: the smallest eigenvalue (in modulus)
- λ_{n-1}: the second smallest eigenvalue (in modulus)
- λ_c: the eigenvalue the closest to $\tilde{\lambda}$
- λ_{c2}: the second eigenvalue the closest to $\tilde{\lambda}$

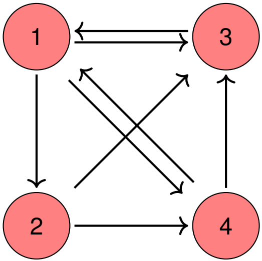
Plan

Problem

A search engine must:

- locate web pages;
- index them;
- sort them according to their importance.

A democratic model



First idea

Sort a page according to the number of pages pointing on it: number of input pointers.

Problem: a link from a "small website" weighs the same as a link from a "big website".

Weighing the pointers

Second idea

Assign a weight to each link and sort the pages by summing the weights of the input pointers.

We can write the linear system:

$$\begin{cases} x_1 = x_3 + x_4 \\ x_2 = x_1 \\ x_3 = x_1 + x_2 + x_4 \\ x_4 = x_1 + x_2 \end{cases}$$

Modeling the problem

What property must be satisfied in order to solve the problem ?

$$x = A.x$$

In other words, we are looking for the eigenvector associated to the eigenvalue 1, where A is the matrix:

$$\begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

Solution existence

Is there a solution to the problem ?

$$\Pi_A(y) = det(A - yI) = \begin{vmatrix} -y & 0 & 1 & 1 \\ 1 & -y & 0 & 0 \\ 1 & 1 & -y & 1 \\ 1 & 1 & 0 & -y \end{vmatrix}$$

$$\Pi_A(y) = y^4 - 2y^2 - 3y - 1$$

So $\Pi_A(1) \neq 0$ and no solution exists.

Remark

Another inconvenient of this method (in addition of not having a solution. . .) is that it gives an "electoral weight" more important to a website which has many output pointers.

Normalization of the weights

Third idea

We normalize the weights: if n_i is the number of output pointers of the website i, the weights are divided by n_i.

$$\begin{cases} x_1 = & & & x_3 + x_4/2 \\ x_2 = x_1/3 & & & \\ x_3 = x_1/3 + x_2/2 & & & + x_4/2 \\ x_4 = x_1/3 + x_2/2 & & & \end{cases}$$

Our new A is:

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

Solution existence

We can check that $\Pi_A(1) = 0$ this time.

More generally, taking the vector $\mathbf{1} = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$, we can see that:

$$A^T \mathbf{1} = \mathbf{1}$$

Therefore 1 is an eigenvalue of A^T. Since A and A^T have the same eigenvalues, 1 is also an eigenvalue of A.

Eigenvalue with biggest modulus

It can be proved that 1 is the eigenvalue with the biggest modulus. Thus we are looking for the eigenvector associated to the biggest eigenvalue $\rightarrow$ power iteration method

This model would have been the reason that the search engine *Google* would perform better than the other ones in the early 2000.

Référence

K. Bryan and T. Leise,The 25,000,000,000 dollars* eigenvector : the linear algebra behind Google, *SIAMReview, 48 (3), 569-581, Sept. 2006*

* estimated value of *Google* in 2004.