Scaling Methods to obtain Doubly stochastic matrices

Krishna Acharya, Nacim Oijid

December 10, 2019

Overview

1. Definition of the problem
2. Applications
3. A Natural Algorithm - Sinkhorn and Knopp
4. Conditions on A
5. Describe types of convergence analysis
6. Knight 2008
7. Livne and Golub
8. Brief Description of Knight and Ruiz
9. Newton's method
10. Splitting method
11. Some Key results from KR 2013
12. Numerical Comparison

Balancing a matrix

- Given a nonnegative square matrix A.
- Find diagonal matrices X and Y, such that XAY is doubly stochastic.

Preconditioning Linear Systems

- We are interested in z, the solution of $Az = b$
- Let $A_1 = XAY$ be doubly stochastic, and we can obtain solution of $A_1 z_1 = Xb$. Then z $= Yz_1$
- Perhaps $A_1 z_1 = Xb$ is more numerically stable, so it helps to do this scaling
- Also note that A and A_1 have the same sparsity.

Back to Balancing

- find positive vectors r and c such that $D(r)AD(c)$ is doubly stochastic
- find positive vectors r and c such that $D(r)AD(c)e = e$ and $D(c)A^T D(r)e = e$
- $r = D(Ac)^{-1}e$ and $c = D(A^T r)^{-1}e$

Iterative method

- The fixed point earlier leads to an iterative method, where $c_{k+1} = D(A^T r_k)^{-1} e$ and $r_{k+1} = D(Ac_{k+1})^{-1} e$
- In MATLAB $c_{k+1} = 1./(A' * r_k), r_{k+1} = 1./(A * c_{k+1})$
- $r_0 = e$ corresponds to the Sinkhorn and Knopp algorithm('52).
- Lets see a run on

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

Important Qs

- Do the c_k and r_k converge to r and c?
- Which non negative A's is this possible for?
- Linear convergence, quadratic ...?

Some relevant definitions

- For $A \in \mathbb{R}^{n \times n}$, a collection of n elements of A is called a **diagonal** of A, provided no two of the elements belong to the same row or column of A.
- A **nonzero diagonal** of A is a diagonal not containing any 0's.
- If G is the bipartite graph whose adjacency matrix is A, then non zero diagonals correspond to the perfect matchings of G.

Matrices for which balancing is possible

- Matrix has **support** if it has at least one non zero diagonal
- a (0-1) matrix A has **total support** provided each of its 1's belongs to a non zero diagonal. That is, if $Aij \neq 0$ we can find a permutation, B, of the rows of A which puts Aij on the leading diagonal and for which $|b_{kk}| > 0$ for all k

SK theorem

Theorem

If $A \in R^{n \times n}$ is non negative, then a necessary and sufficient condition that there exists a doubly stochastic matrix P of the form DAE where D and E are diagonal matrices with positive main diagonals, is that A has total support. If P exists, then it is unique. D and E are also unique up to a scalar multiple iff A is fully indecomposable.

Theorem

A necessary and sufficient condition that the SK algorithm converges is that A has total support

Convergence analysis

There are many papers on the convergence of r_k and c_k, The main idea in these papers is to show that on each iteration a special function(e.g entropy, log barrier, permanent) is decreasing.
For a matrix that can be balanced, a lower bound is known $\rightarrow$ it corresponds to the doubly stochastic case.

- Convex Optimization - Much faster scaling, Cohen
- Log barrier methods(Kalantari and Khachiyan 1996)
- Entropy minimization
- Connections to permanent of a matrix
- Topological methods

Linear convergence for symmetric A, Knight 2008

- Considers the case where A is symmetric. SK algorithm can be applied,
- Looks for diagonal matrix D, such that DAD is doubly stochastic.
- The symmetric analogues of SK algorithm are $x = D(Ax)^{-1}e$ and the iterative step $x_k = D(Ax_{k-1})^{-1}e$.

Rate of convergence analysis, Knight 2008

Theorem

If A is fully indecomposable then the SK algorithm will converge linearly to vectors r_* and c_*, such that $D(r_*)AD(c_*) = P$ where P is doubly stochastic. Furthermore, there exists $K \in \mathbb{Z}$ such that for $k \geq K$.

$$\left\| \begin{bmatrix} r_{k+1} \\ c_{k+1} \end{bmatrix} - \begin{bmatrix} r_* \\ c_* \end{bmatrix} \right\| \leq \sigma_2^2 \left\| \begin{bmatrix} r_k \\ c_k \end{bmatrix} - \begin{bmatrix} r_* \\ c_* \end{bmatrix} \right\|$$

Where σ_2 is the second singular value of P.

What LG 2004 achieve

- Tackle the problem of Bi-normalization
- Provide an Iterative algorithm **BIN** for scaling all rows and columns of a real symmetric matrix to unit-2 norm.

Important notions LG 2004

- $\tilde{A} = DAD$
- $B_{ij} = A_{ij}^2$
- $\sum_j \tilde{A}_{ij}^2 = \sum_j d_i^2 A_{ij}^2 d_j^2 = c \quad \forall i \in [n]$
- We are looking for positive solution $\tilde{x}$ to $B(x)x = be$, where b is a positive real and $B(x) = D(Bx)$
- Equivalently(used in GS method) we are looking for a positive solution to $(I_n - \frac{1}{n}ee^T)B(x)x = be - \frac{1}{n}ee^T be$

Existence and uniqueness

- Matrix is defined to be scalable if $B(x)x = be$ has a positive solution.
- What we are trying to solve is an $n \times n$ system of quadratic equations in $x_1, \dots x_n$. (positive x). Solution does not always exist.

Theorem

If a matrix is scalable, it is either (a) not diluted ,or (b) $n = 2$ and $A = \begin{pmatrix} 0 & a \\ a & 0 \end{pmatrix}$

- Diluted, if $\exists i \in [n], A_{ii} \neq 0$ and $A_{kj} = 0 \;\; \forall k \neq i$ and $j \neq i$

BIN Algorithm

Algorithm 1: BIN

Input: $A, n \times n$ real symm matrix, $d^0 \in \mathbb{R}^n$,TOL-tolerance

Output: $\tilde{A}, d \in \mathbb{R}^n$

$B = A \circ A; \quad x = ((d_1^0)^2, \dots, (d_n^0)^2)^T$;

Do Update rule;

while $s(x) > TOL \cdot \bar{\beta}(x)$ *and sweeps* $<$ *MaxSweeps* **do**

  for $i \leftarrow 1$ **to** n **do**

    Solve equation i for x_i keeping all other x_j fixed at current values;

  end

  Do update rule ;

end

Update rule:

- $d_i = \bar{\beta}^{-1/2} x_i \forall i \in [n]$
- $\tilde{A} = DAD$
- $s(x) = (\frac{1}{n} \sum_k (x_k \beta_k - \bar{\beta})^2)^{1/2}$

Notations

- Let D be the operator that transform a vector into a diagonal matrix of his elements :

$$D : x = \begin{pmatrix} x_1 \\ \vdots \\ \vdots \\ x_n \end{pmatrix} \mapsto D(x) = \begin{pmatrix} x_1 & 0 & \dots & \dots & 0 \\ 0 & x_2 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & \dots & 0 & x_n \end{pmatrix}$$

Notations

- Let D be the operator that transform a vector into a diagonal matrix of his elements :

$$D : x = \begin{pmatrix} x_1 \\ \vdots \\ \vdots \\ x_n \end{pmatrix} \mapsto D(x) = \begin{pmatrix} x_1 & 0 & \dots & \dots & 0 \\ 0 & x_2 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & \dots & 0 & x_n \end{pmatrix}$$

- let e represent the vector $\begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$

Quick recall on iterativ method

- We are looking for 2 vectors r and c such that $D(r)AD(c)$ is doubly stochastic.
  That means $D(r)Ac = e$ and $D(c)A^T r = e$

Quick recall on iterativ method

- We are looking for 2 vectors r and c such that $D(r)AD(c)$ is doubly stochastic.
  That means $D(r)Ac = e$ and $D(c)A^T r = e$
- we have the following :

$$c = D(A^T r)^{-1} e \tag{1}$$

$$r = D(Ac)^{-1} e. \tag{2}$$

Quick recall on iterativ method

- We are looking for 2 vectors r and c such that $D(r)AD(c)$ is doubly stochastic.
  That means $D(r)Ac = e$ and $D(c)A^T r = e$
- we have the following :

$$c = D(A^T r)^{-1} e \tag{1}$$

$$r = D(Ac)^{-1} e. \tag{2}$$

- iterative method : $c_{k+1} = D(A^T r_k)^{-1} e$, $r_{k+1} = D(Ac)^{-1} e$
- So, if A is symmetric, by uniqueness of r and c, we have $r = c$. So, the unique solution x^* is solution of the equation

$$f(x^*) = D(x^*)Ax^* - e = 0 \tag{3}$$

Quick recall on iterativ method

- We are looking for 2 vectors r and c such that $D(r)AD(c)$ is doubly stochastic.
  That means $D(r)Ac = e$ and $D(c)A^T r = e$
- we have the following :

$$c = D(A^T r)^{-1} e \tag{1}$$

$$r = D(Ac)^{-1} e. \tag{2}$$

- iterative method : $c_{k+1} = D(A^T r_k)^{-1} e$, $r_{k+1} = D(Ac)^{-1} e$
- So, if A is symmetric, by uniqueness of r and c, we have $r = c$. So, the unique solution x^* is solution of the equation

$$f(x^*) = D(x^*)Ax^* - e = 0 \tag{3}$$

- The iterative step therefore is $x_{k+1} = D(Ax_k)^{-1} e$. But we don't do the iterations as that would be SK. We will go back to the equation 3 and find a solution of it with Newton's method.

Algorithm 2: Rows Columns scaling

Input: a $m \times n$ matrix A

$A^{(0)} \leftarrow A$;

$D^{(0)} \leftarrow I_m$;

$E^{(0)} \leftarrow I_n$;

for $k \leftarrow 0$ **to** *convergence* **do**

$\quad R \leftarrow D(\sqrt{||r_i^k||})$;

$\quad C \leftarrow D(\sqrt{||c_j^k||})$;

$\quad A^{(k+1)} \leftarrow R^{-1} A^{(k)} C^{-1}$;

$\quad D^{(k+1)} \leftarrow D^{(k)} R^{-1}$;

$\quad C^{(k+1)} \leftarrow E^{(k)} C^{-1}$;

end

Newton's method

- remind : Newton's method (on the blackboard)

Newton's method

- Let $J(X)$ be the jacobian of f. We have $J(x) = \frac{\partial}{\partial x}(D(x)Ax - e) = D(x)A + D(Ax)$

Newton's method

- Let $J(X)$ be the jacobian of f. We have $J(x) = \frac{\partial}{\partial x}(D(x)Ax - e) = D(x)A + D(Ax)$
- result :

$$x_{k+1} - x_k = -J(x)^{-1} f(x) \tag{4}$$

$$x_{k+1} = x_k - \big(D(x)A + D(Ax)\big)^{-1}\big(D(x^*)Ax^* - e\big) \tag{5}$$

Newton's method

- Let $J(X)$ be the jacobian of f. We have $J(x) = \frac{\partial}{\partial x}(D(x)Ax - e) = D(x)A + D(Ax)$
- result :

$$x_{k+1} - x_k = -J(x)^{-1}f(x) \tag{4}$$

$$x_{k+1} = x_k - \big(D(x)A + D(Ax)\big)^{-1}\big(D(x^*)Ax^* - e\big) \tag{5}$$

- we can arrange this equation :

$$(A + D(x_k)^{-1}D(Ax_k))x_{k+1} = Ax_k + D(x_k)^{-1}e \tag{6}$$

Newton's method

- Let $J(X)$ be the jacobian of f. We have $J(x) = \frac{\partial}{\partial x}(D(x)Ax - e) = D(x)A + D(Ax)$
- result :

$$x_{k+1} - x_k = -J(x)^{-1} f(x) \tag{4}$$

$$x_{k+1} = x_k - \big(D(x)A + D(Ax)\big)^{-1} \big(D(x^*)Ax^* - e\big) \tag{5}$$

- we can arrange this equation :

$$(A + D(x_k)^{-1} D(Ax_k))x_{k+1} = Ax_k + D(x_k)^{-1} e \tag{6}$$

- We introduce the matrix $A_k = (A + D(x_k)^{-1} D(Ax_k)$. So the equation is

$$A_k x_{k+1} = Ax_k + D(x_k)^{-1} e \tag{7}$$

Splitting method

- Suppose $A_k = M_k + N_k$ with M_k non-singular. we will solve this problem with inner-outer iterations. We will denote by *outer* the operations of the Newton step, and by *inner* the one of the splitting method.

Splitting method

- Suppose $A_k = M_k + N_k$ with M_k non-singular. we will solve this problem with inner-outer iterations. We will denote by *outer* the operations of the Newton step, and by *inner* the one of the splitting method.
- The outer operations, as described before, just update x_k using J(x)

Splitting method

- Suppose $A_k = M_k + N_k$ with M_k non-singular. we will solve this problem with inner-outer iterations. We will denote by *outer* the operations of the Newton step, and by *inner* the one of the splitting method.
- The outer operations, as described before, just update x_k using J(x)
- The inner iteration is described by this equation :

$$M_k y_{j+1} = N_k y_j + Ax_k + D(x_k)^{-1} e$$
$$y_0 = x_k$$

Splitting method

- Suppose $A_k = M_k + N_k$ with M_k non-singular. we will solve this problem with inner-outer iterations. We will denote by *outer* the operations of the Newton step, and by *inner* the one of the splitting method.
- The outer operations, as described before, just update x_k using J(x)
- The inner iteration is described by this equation :

$$M_k y_{j+1} = N_k y_j + Ax_k + D(x_k)^{-1} e$$
$$y_0 = x_k$$

- But here, one inner-iteration is enough. It gives us :

$$x_{k+1} = x_k + M_k^{-1}(D(x_k)^{-1} e - Ax_k) \tag{8}$$

Splitting method

- Suppose $A_k = M_k + N_k$ with M_k non-singular. we will solve this problem with inner-outer iterations. We will denote by *outer* the operations of the Newton step, and by *inner* the one of the splitting method.
- The outer operations, as described before, just update x_k using J(x)
- The inner iteration is described by this equation :

$$
\begin{aligned}
M_k y_{j+1} &= N_k y_j + A x_k + D(x_k)^{-1} e \\
y_0 &= x_k
\end{aligned}
$$

- But here, one inner-iteration is enough. It gives us :

$$x_{k+1} = x_k + M_k^{-1}(D(x_k)^{-1} e - A x_k) \tag{8}$$

- By choosing $M_k = D(x_k)^{-1} D(A x_k)$ we found again the equations we had before. So with a good choice of M, things can only go faster.

Convergence : general case

Definition

To measure convergence rate, if we denote by C the set of all sequences generated by our method, we define :

$$R(x) := \sup\{\limsup ||x_k - x||^{1/k} | (x_k) \in C\}$$

Theorem

If $f : \mathbb{R}^n \to \mathbb{R}^n$ is differentiable in an open neighbourhood of a point x^* where f' is continuous and $f(x^*) = 0$. If $f(x^*) = M(x) - N(x)$, where M is continuous and invertible. If $\rho(G(x^*)) < 1$ where $G = M^{-1}N$. then for $m \geq 1$, the iterative process defined by $x_{k+1} = x_k - \big(\sum_{j=1}^{m} G(x_k)^{j-1}\big)(x_k)^{-1} f(x_k)$ has attraction point x^* and $R(x^*) = \rho(G(x^*)^m)$

Convergence here

Corollary

If A is a symmetric matrix, and $x^* > 0$ is such that $D(x^*)AD(x^*)$ is stochastic, Then, for the iterative process described in 8, we have $R(x) = \rho(M^{-1}N)$ where $M - N$ is the splitting associated to $A + D(x^*)$

Proof.

on the blackboard $\square$

Measure of the convergence rate

- $R(x)$ is invariant by diagonal scaling.
- $D(x*)(A + D(x*)^{-2})D(x*) = P + I$
- So if A is symmetric, it improves a lot the rate of convergence, with a good choice of M.

Quick overview of other methods

- Conjugate gradient

Quick overview of other methods

- Conjugate gradient
- Optimization

Numerical Results for SK, GS, BNEWT

- BNEWT is inexact newton iteration with conjugate gradients, SK is Sinkhorn and Knopp, GS is the Livne and Golub method.
- The number of matrix vector products is the cost.
- Algorithms ran till residual norm $< 10^{-6}$

Hessenberg Matrices

- 10×10 matrix
- $H = (h_{ij}),\ h_{ij} = \begin{cases} 0, & \text{if } j < i - 1 \\ 1, & \text{otherwise} \end{cases}$
- H_2 same as H except $h_{12} = 100$, $H_3 = H + 99I$

	SK	GS	BNEWT
H	110	114	76
H_2	144	150	90
H_3	2008	2012	94

Table: Number of Matrix vector products

Convergence, number of iterations

	n=10	25	50	100
BNEWT	124	300	660	1792
SK	3070	16258	61458	235478

Table: Number of iterations for H_n

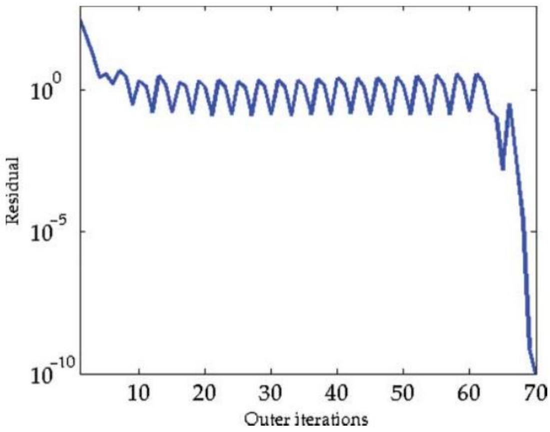

Figure: Convergence graph for BNEWT on H_{50}

References

- Philip A. Knight. "The Sinkhorn-Knopp Algorithm: Convergence and Applications". In: *SIAM J. Matrix Anal. Appl.* 30.1 (2008)
- Oren E. Livne and Gene H. Golub. "Scaling by Binormalization". In: *Numerical Algorithms* 35 (2004), pp. 97–120
- Philip A. Knight and Daniel Ruiz. "A fast algorithm for matrix balancing". In: *Journal of Numerical analysis* 33 (2013)
- Michael Cohen et al. "Matrix Scaling and Balancing via Box Constrained Newton's Method and Interior Point Methods". In: (Apr. 2017)
- Daniel Ruiz and Bora Uçar. "A Symmetry Preserving Algorithm for Matrix Scaling". In: *SIAM Journal on Matrix Analysis and Applications* 35 (July 2014). DOI: 10.1137/110825753