Parallelized Multiresolution Matrix Factorization (pMMF)

Risi Kondor, Nedelina Teneva, Pramod Kaushik Mudrakarta, Jonathan Eskreis-Winkler
Departments of Statistics and Computer Science, University of Chicago

THE UNIVERSITY OF CHICAGO

Abstract

Multiresolution Matrix Factorization (MMF) is a recently introduced method for finding multiscale structure and defining wavelets in large and complex datasets, networks, and matrices. It finds diverse applications - e.g. in matrix compression, Gaussian processes and preconditioning linear systems. The original MMF algorithm of [Kondor, Teneva & Garg, 2014] scales with n^3 or worse (where n is the number of rows/columns in the matrix to be factorized) and makes it infeasible for large scale problems. We propose a new fast parallelized MMF algorithm, which can scale to n in the range of millions. By using the resources of UChicago's Research Computing Center, our experimental results confirm the theoretical analysis on algorithm speed and show that pMMF outperforms competing algorithms. We also used RCC to identify best parameters for which pMMF achieves best scalability.

Multiresolution Matrix Factorization

- Traditional multiresolution analysis (MRA)
  - A given set of (square-integrable) functions is recursively split into sets of smooth V_k and coarse W_k functions, where $k = 1, \ldots, L$
  - The basis transforms $V_k \to V_{k+1}$ and $W_k \to W_{k+1}$ are sparse and the basis for W_k is localized

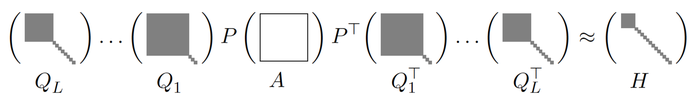
$$L_2(X) \to \quad \ldots \quad \to V_0 \to V_1 \to V_2 \to \quad \ldots$$
$$\searrow \quad \searrow \quad \searrow$$
$$W_1 \quad W_2 \quad W_3$$

- Extension of MRA to matrix analysis
  For a symmetric matrix $A \in \mathbb{R}^{n \times n}$, the MMF is a L-level factorization where

$$A \approx Q_1^T \cdots Q_{L-1}^T Q_L^T H Q_L Q_{L-1} \cdots Q_1$$

and each Q_l is an orthogonal matrix with the properties:

1. Q_l is a sparse k-point rotation matrix. $k = 2$ corresponds to a Givens rotation
2. The set of columns on which the rotations effectively act are nested across levels
3. H is an S_L–core-diagonal matrix which is a diagonal matrix apart from the submatrix $[H]_{S_L,S_L}$.



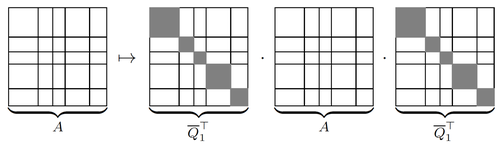
MMF Algorithmic Strategy

At each level:

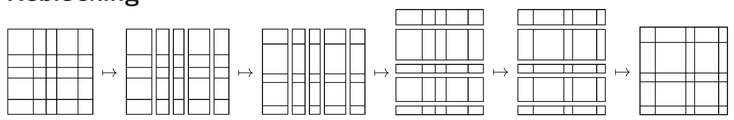
1. Cluster the columns of the active part of matrix at this level
2. Randomly select one row/column and pair it with other columns in same cluster with highest inner product
3. Perform rotations among pairs
4. Deactivate a subset of the rotated columns and pass the resulting smaller matrix to the next level

Parallel MMF (pMMF)

- **Matrix Blocking and Stages**
  To make the MMF computation parallel, the matrix is blocked. The shaded regions correspond to the sets of columns on which the different k-point or Givens rotations in a level act.



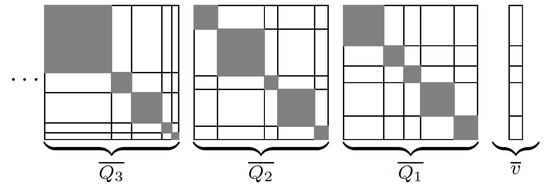
  Different columns of blocks can be sent to different processors.
- **Reblocking**



  After a stage is complete, rows/columns are reclustered

Matrix free arithmetic

The compressed matrix is *never* computed explicitly. Instead, when applying it to a vector, apply the rotations individually:



When applying an pMMF factorization to a vector, the vector must go through the same reblocking.

Theoretical complexity of the pMMF components

	serial MMF		pMMF operations		pMMF time		N_{proc}
	dense	sparse	dense	sparse	dense	sparse	
Computing Gram matrices	n^3	γn^3	Pcn^2	γPcn^2	Pc^3	γPc^3	m^2
Finding Rotations	n^3	n^3	cn	cn	c^2	c	m
Updating Gram matrices	n^3	$\gamma^2 n^3$	$c^2 n$	$\gamma^2 c^2 n$	c^3	$\gamma^2 c^3$	m
Applying rotations	kn^2	γkn^2	kn^2	γkn^2	kc^2	γkc^2	m^2
Clustering			pmn^2	γpmn^2	pcn	γpcn	m^2
Reblocking			pn^2	γpn^2	pcn	γpcn	m
Factorization total	n^3	n^3	Pcn^2	γPcn^2	Pc^3	γPc^3	m^2

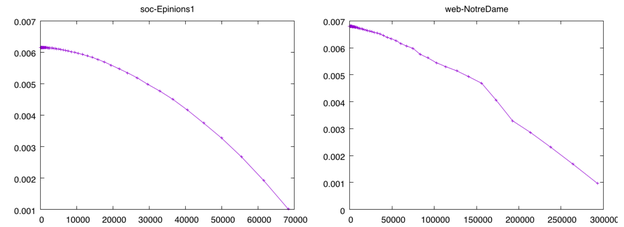
Assuming N_{proc}–fold parallelism. n: num. rows of symmetric matrix A. γ: fraction of non-zero entries A, k: order of the rotations. In pMMF, P: number of stages/levels, m: num. clusters, c: cluster size (thus, $c = \theta(n/m)$). $k \le P \le c \le n$, but $n = o(c^2)$.

pMMF software library

- Multi-core implementation of MMF in C++11
- Fully object oriented design with a transparent API
- Optional usage of LAPACK and BLAS for linear algebra tasks
- Visualization using OpenGL and GLUT
- MATLAB interface

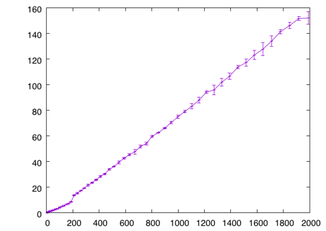
Accuracy of MMF

We ran experiments on the Broadwell nodes of Midway2 using 28 cores. Following plots show MMF approximation error as a function of core size.



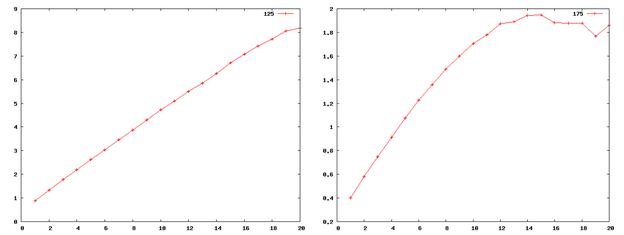
Scale - Effect of matrix size

We confirm empirically that the runtime (y-axis) of pMMF algorithm scales linearly with matrix size (x-axis, $x10^3$). This is true even when we use sizes running into hundreds of thousands



Exploiting Parallelism - best MMF parameters

We used RCC resources to find best parameters for parallelism. For example, we concluded that using a small cluster size such that each cluster fits on the L2 cache of a core works best. The left plot corresponds to a smaller cluster size and the right one to larger.



MMF matrix compression vs. low-rank matrix compression

In matrix compression/sketching, MMF (red curve in the plots) outperforms its competitors on most datasets, w.r.t. Frobenius norm error