Numerical Methods for Data Science: Latent Factor Models, Part I

David Bindel

17 June 2019

Department of Computer Science
Cornell University

Cornell CS 6241
Spring 2018

SJTU CS 258
June 2018 + May 2019

How do we teach numerical methods in a "data science" age?

- CS 4220 (S12): Data science projects in NA course
- CS 6241 (S18): "Numerical methods for data science"
- SJTU CS 258 (Jun 18+19): Undergrad version
- CS 3220 (F19): "Computational math for CS"

Lecture plan

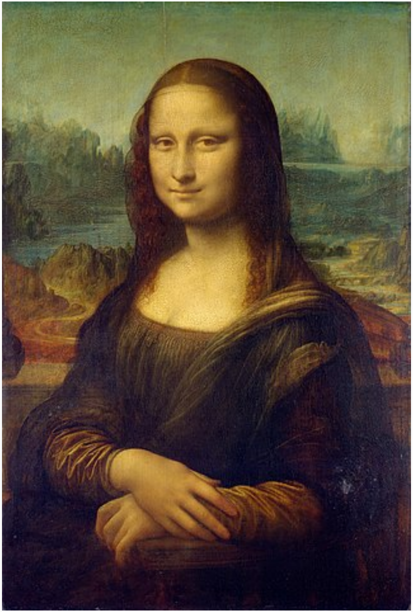
Three threads from “lay of the land” to current research:

- Monday: Latent Factor Models
    - **Matrix data and decompositions**
    - NMF and topic models
- Wednesday: Scalable Kernel Methods
    - Structure and interpretation of kernels
    - Making kernel methods scale
- Friday: Spectral Network Analysis
    - Network spectra, optimization, and dynamics
    - Network densities of states

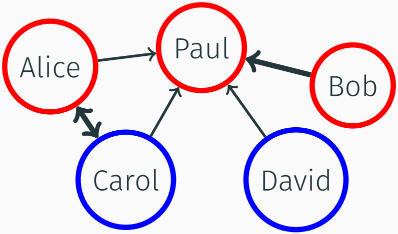
Matrix Data: Relations and Ranking

	Casa Blanca	Forest Gump	Rocky	The Matrix	…
Alice	5	5	1		
Bob	1		5	5	
Carol		2			
Dan		5		5	
…					

Matrix Data: Images and Functions



Matrix Data: Networks and Relations



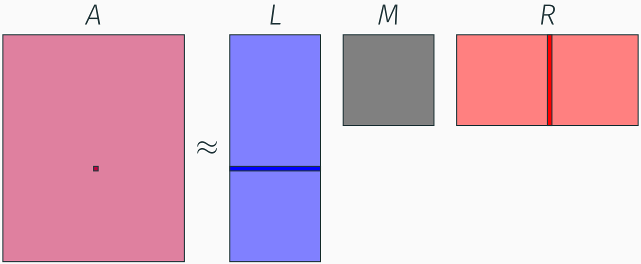
$$W = \begin{bmatrix} 0 & 0 & w_{AC} & 0 & 0 & w_{AP} \\ 0 & 0 & 0 & 0 & 0 & w_{BP} \\ w_{CA} & 0 & 0 & 0 & 0 & w_{CP} \\ 0 & 0 & 0 & 0 & 0 & w_{DP} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Matrix Data

- Relations between objects and features, e.g.
  - Documents and word frequencies ("bag of words" models)
  - User rankings of movies, songs, etc
  - Images and pixel values
  - Indicators for DNA markers in organisms
  - Treatments and outcome histograms
  - Snapshots of a state vector at different times
- Grid samples of bivariate functions
  - Image pixels indexed by row and column
  - PMF values for a discrete 2D random variable
- Relations between pairs of objects
  - Network relations (e.g. friendships)
  - Co-occurrence statistics, interaction frequencies, etc

Can generalize beyond pairwise data with *tensor* methods.

Latent Factor Modeling



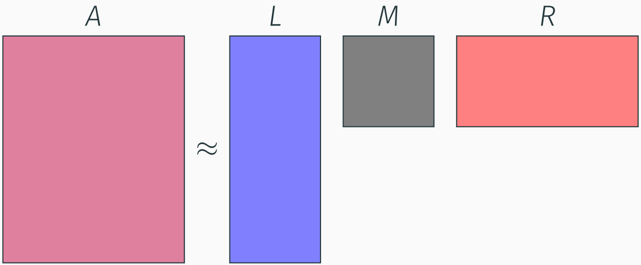
Map objects and features to *latent coordinates*, model data as bilinear function of coordinates:

$$a_{ij} \approx l_{i,:} M r_{:,j}.$$

Unknowns: latent coordinates for points, bilinear form.

This is underdetermined — need constraints on L, M, R for uniqueness.

Latent Factor Modeling



Model data arranged as a matrix $A \in \mathbb{R}^{m \times n}$ by

$$A \approx LMR, \quad L \in \mathbb{R}^{m \times r}, M \in \mathbb{R}^{r \times r}, R \in \mathbb{R}^{r \times n}$$

perhaps with constraints on L, M, and R.

Latent Factor Modeling

Model data arranged as a matrix $A \in \mathbb{R}^{m \times n}$ by

$$A \approx LMR, \quad L \in \mathbb{R}^{m \times r}, M \in \mathbb{R}^{r \times r}, R \in \mathbb{R}^{r \times n}$$

perhaps with constraints on L, M, and R.

From this, we would like to

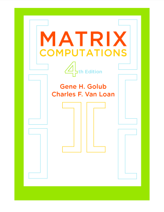
- Compress data
- Remove “noise” (including outliers)
- Fill in missing data
- Cluster and classify objects
- Find meaningful “parts” to data

with the Magic of Matrices.

The Magic of Matrices (or Machine Learning)

https://xkcd.com/1838/

What is The Matrix?



$A \in \mathbb{R}^{m \times n}$ is an m-by-n array of real numbers:

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$

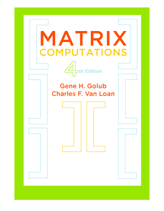
Is this right? Consider some a_{ij} meanings:

- Image pixel at row i and column j
- Departure of train j from station i
- Frequency of word j in document i

None seem all that linear algebraic!

Seek useful puns between *matrices* and *arrays*.

What is The Matrix?



In LA, a matrix represents (w.r.t. basis choice):

Linear map	$L : \mathcal{V} \to \mathcal{W}$	$w = Av$
Linear operator	$L : \mathcal{V} \to \mathcal{V}$	$w = Av$
Sesquilinear form	$b : \mathcal{V} \times \mathcal{W} \to \mathbb{R}$	$b = w^* Av$
Quadratic form	$\phi : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$	$\phi = v^* Av$

Different possible attitudes toward bases

- Pure LA: The LA object is the thing
  - Think basis independent
    (except canonical choices)
- Numerical LA: Sometimes bases matter!
  - Sparsity, shape, non-negativity
- Data: The matrix (array) is the thing

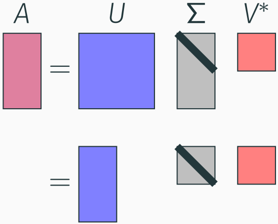
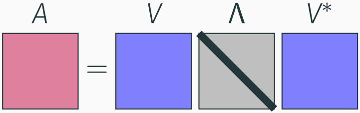
Matrices in Linear Algebra: Canonical Forms

	General bases	Orthonormal bases
Linear map (or bilinear form)	Rank/nullity $A = XI_kY^*$	**SVD** $A = U\Sigma V^*$
Linear operator	Jordan form $A = VJV^{-1}$	Schur form $A = UTU^*$
Quadratic form	Sylvester inertia $A = X_1X_1^* - X_2X_2^*$	**Symm eigendecomp** $A = V\Lambda V^*$

What if basis choice (identity of rows and columns) matters?

- Natural transformations are basis permutations.
- Permutation matrices $\subset$ orthogonal matrices.
- Orthogonal decompositions make good building blocks!

The SVD and the Symmetric Eigenvalue Problem

SVD	SEP
	
• U, V orthonormal	• V orthonormal
• $\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \ldots)$	• $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots)$
• σ_j descending, nonneg	• λ_j descending
• Full: Σ rectangular	• $A = A^*$
• Economy: U rectangular	• Agrees with SVD if SPD

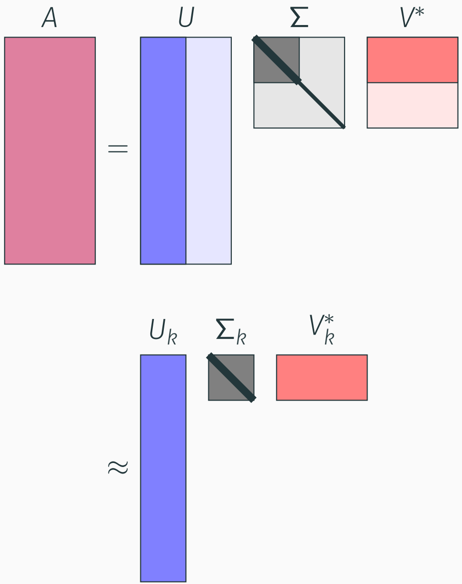
Construction

Variational construction of SVD (similar for SEP):

- Maximize $\|Av\|$ over $\|v\| = 1$
  - Unique maximum value, no local maxima!
  - Result: $Av_1 = \sigma_1 u_1$ where $u_1 = Av_1/\|Av_1\|$
- Maximize $\|Av\|$ over $\|v\| = 1$ and $v \perp v_1$
  - Again a "nice" optimization problem
  - Result: $Av_2 = \sigma_2 u_2$
- Continue in this fashion

Completely greedy is OK in principle! No need to backtrack.

Matrices and Data: Low-Rank Approximation by SVD



Consider the (economy) SVD of $A \in \mathbb{R}^{m \times n}$ for $m \geq n$

$$A = U\Sigma V^*, \quad U \in \mathbb{R}^{m \times n}, \Sigma \in \mathbb{R}^{n \times n}, V \in \mathbb{R}^{n \times n}$$

where $U^*U = I$, $V^*V = I$ and Σ is diagonal with

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n \geq 0$$

Eckart-Young: Best rank k approximation is the truncated SVD

$$\hat{A} = U_k \Sigma_k V_k^*.$$

True in Froebenius norm, spectral norm. Error is

$$\|A - \hat{A}\|_F^2 = \sum_{j=k+1}^{n} \sigma_j^2, \qquad \|A - \hat{A}\|_2 = \sigma_{k+1}$$

Actual Computation

SVD and SEP admit similar computational schemes

- Small and dense (up to a couple 1000, MATLAB eig/svd):
  - Orthogonal reduction to bidiagonal / tridiagonal ($O(n^3)$)
  - Cheaper reduction rest of the way to the decomposition
- Subspace iteration
  - Start with random orthonormal columns
  - Multiply by A (and maybe A^*) a couple times
  - RandNLA: Few steps for OK accuracy (with oversample)
  - Good for cache use, great for modest accuracy
- Krylov methods (Matlab eigs and svds)
  - Repeatedly multiply a few vectors by A (and maybe A^*)
  - Orthogonalize at each step (parallel bottleneck)
  - Various restarting and acceleration tricks
  - High accuracy for a few extreme pairs

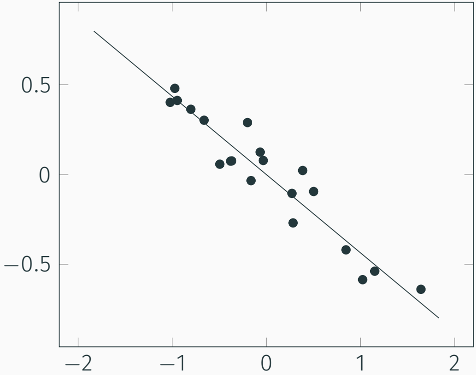
Some Computational Aspects

Big takeaways:

- We have good codes that (mostly) “just work”
- People have thought through how to make them run fast
- Have good backward stability properties
- No misconvergence or sensitivity to starting point

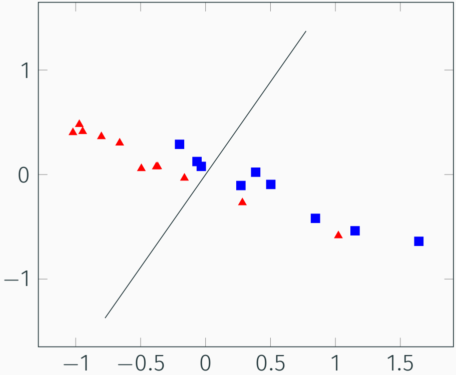
These are *great* building blocks

Example: PCA



- Matrix rows represent different object properties
- Typically center (subtract column means); may scale
- Run SVD on resulting matrix
- Dominant singular values are “principal components”

Example: LDA



- Matrix rows represent different object properties
- Rows are labeled, want to discriminate these labels
- Solution involves a generalized SEP
- May be very different from PCA directions

Example: Latent semantic analysis

Vector space model (Salton and Yang, 1975)

- Columns are documents, row entries are word frequency
- Scale raw data (tf-idf)
- Take SVD of resulting matrix
- Rows of $U \approx$ latent word representations
- Columns of $V \approx$ latent document representations
- Useful for comparing words to words, docs to docs
- Also useful for searching for docs by keywords

Problem: latent coordinates are hard to interpret!

Example: Spectral Clustering

Goal: Cluster objects (rows) according to features

- Compute a truncated SVD: $A \approx U_r \Sigma_r V_r^*$
- Treat rows of $U_r \Sigma_r$ as latent coordinates
- Run k-means to cluster points

Essentially gives

$$A \approx LC^*$$

where

$$l_{ij} = \begin{cases} 1, & \text{item } i \text{ in class } j \\ 0, & \text{otherwise} \end{cases} \qquad C_{:,j} = \text{centroid of class } j$$

Several types of spectral clustering – will discuss again later!

The Story So Far

SVD and SEP provide useful dimension reduction

- Lower-dimensional “latent spaces” for further analysis
- Latent space clarifies object similarity / dissimilarity
- But interpreting the coordinate system is hard!

So what do we want beyond SVD and SEP?

- Linear dimension with interpretable structure (today)
- Nonlinear dimension reduction (a little coming up)

Recall: The Latent Factor Framework

Factor data matrix $A \in \mathbb{R}^{m \times n}$ as

$$A \approx LMR, \quad L \in \mathbb{R}^{m \times r}, M \in \mathbb{R}^{r \times r}, R \in \mathbb{R}^{r \times n}$$

with different structures on L, M, and R.

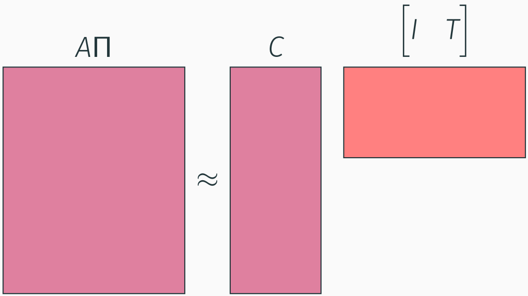
- R (or MR) as maps original features to “latent” features
- Can impose constraints on L, M, and R
- Orthogonality constraints gets us to SVD (easy)
- Other types of constraints are harder
- ...but other constraints improve interpretability

Factor Selection and Skeletonization

Idea: Express latent factors via rows/columns of A

- Improves interpretability (maybe — more tricks help)
- May improve cost of working with matrix
  - Can form part of a row/column without forming all
  - Useful in both experiments and computation
- *But* how do we choose good representative rows/cols?
  - Want things to be as linearly independent as possible
  - Also want good approximation quality
  - SVD provides a “speed of light” bound

Interpolative Decomposition / Skeletonization



- C consists of the leading k columns of A
- $T = C^\dagger (A\Pi)_{:,k+1:n}$ chosen to minimize Frobenius norm error
- Exists some Π — not easily computed — such that
  - Entries of T are at most 2
  - Singular values of $\begin{bmatrix} I & T \end{bmatrix}$ in $[1, 1 + \sqrt{k(n-k)}]$
  - Approximation error within $1 + \sqrt{k(n-k)}$ of optimal

CUR Decomposition

Idea: $A \approx CUR$ where

- C consists of columns of A
- R consists of rows of A
- $U = C^\dagger A R^\dagger$ is optimal given C, R
- Selection of good C and R is again the challenge

Can also consider symmetric variants where $R = C^*$.

Pivoted QR

Pivoted QR decomposition is

$$A\Pi = QR$$

where R is upper triangular, r_{ii} are positive and decreasing.

Idea: Be greedy (as in SVD), but choose among columns of A

- Choose column a_i with maximum norm
  - Set $r_{11} = \|a_i\|$ and $q_1 = a_i/r_{11}$ (so $a_1 = q_1 r_1$)
  - Orthogonalize vs q_1: $a_j' = a_j - q_1 r_{j1}$ with $r_{j1} = q_1^T a_j$
- Choose modified column a_i' with maximum norm
  - Set $r_{22} = \|a_i'\|$ and $q_2 = a_i'/r_{22}$
  - Orthogonalize vs q_2 to get new A' matrix
- Keep repeating the process (re-ordered Gram-Schmidt)

Unlike SVD this is *not* optimal — but often pretty good.

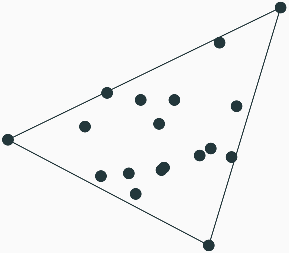
Computing Pivoted QR

As with SVD, constructive definition $\neq$ usual computation.

- Dense case is a standard building block (qr in MATLAB)
- Ongoing work to improve parallelism and cache efficiency
- Clever tournament pivoted (TSQR) for $m \gg n$

Again: A great building block to borrow from someone else!

The Geometry of Pivoted QR



Chosen column is on the convex hull of a 1D projection
$\implies$ it is a point on the convex hull of the original columns.

Pivoted QR to Pivoted Cholesky

Symmetric positive definite matrices $\equiv$ Gram matrices:

$$H = A^T A$$

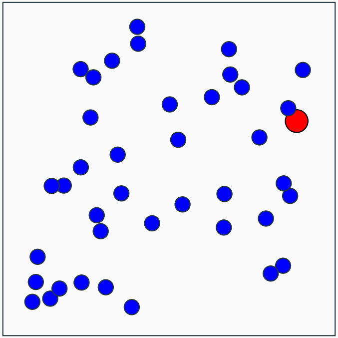
Decompose:

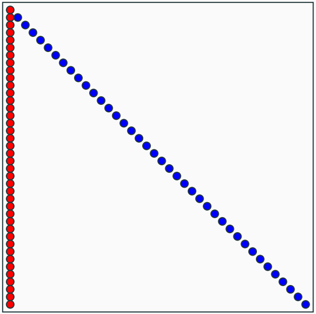
$$A\Pi = QR \implies$$
$$\Pi^T H \Pi = R^T Q^T Q R = R^T R$$

Running pivoted Cholesky is equivalent to pivoted QR.

Esp useful when A is implicit (only access via H).
Let's look briefly at an example we'll see again on Wednesday.

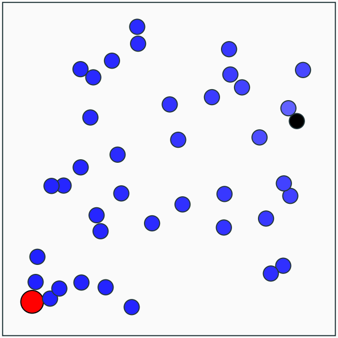
Example: Pivoted Cholesky on a Kernel Matrix

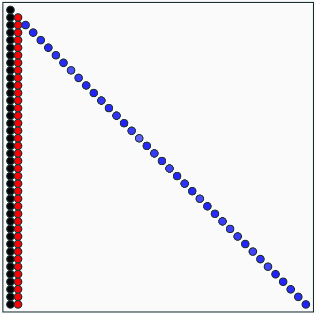




Diagonal element: 1.00e+00

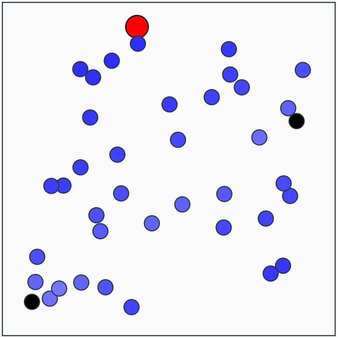
Example: Pivoted Cholesky on a Kernel Matrix

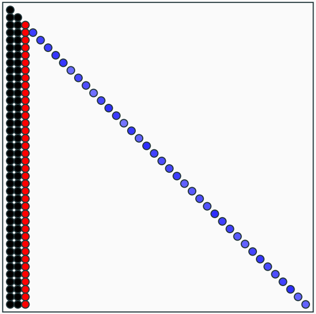




Diagonal element: 6.77e-02

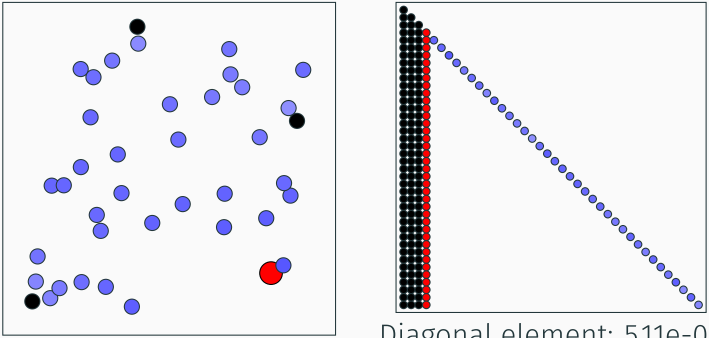
Example: Pivoted Cholesky on a Kernel Matrix





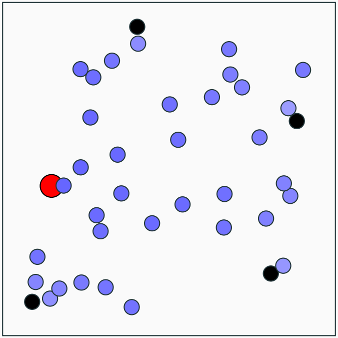
Diagonal element: 1.91e-02

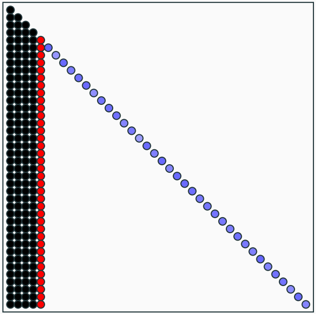
Example: Pivoted Cholesky on a Kernel Matrix



Diagonal element: 5.11e-04

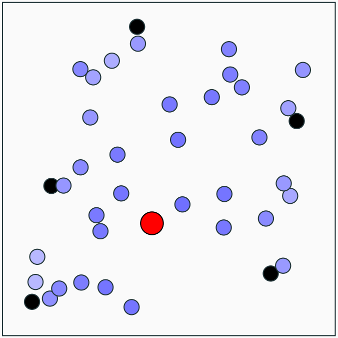
Example: Pivoted Cholesky on a Kernel Matrix

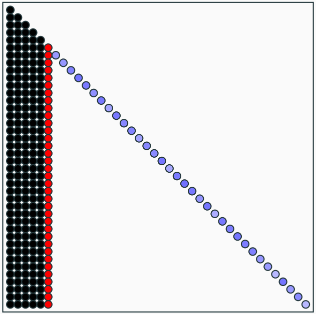




Diagonal element: 1.19e-04

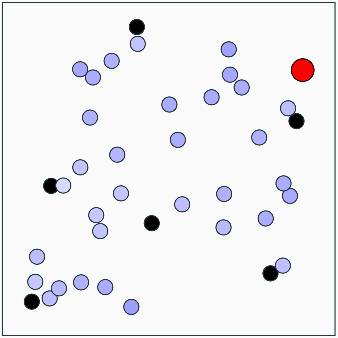
Example: Pivoted Cholesky on a Kernel Matrix

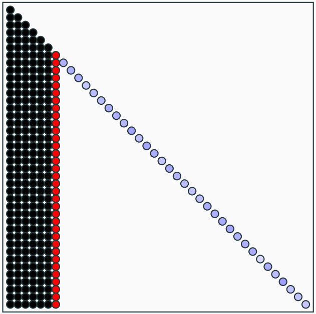




Diagonal element: 4.18e-05

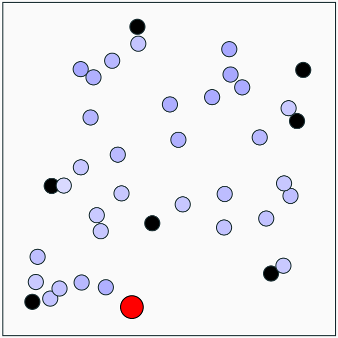
Example: Pivoted Cholesky on a Kernel Matrix

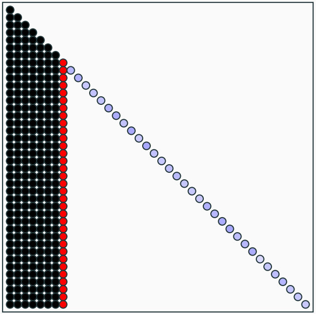




Diagonal element: 8.54e-07

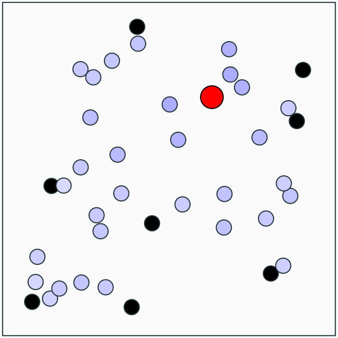
Example: Pivoted Cholesky on a Kernel Matrix

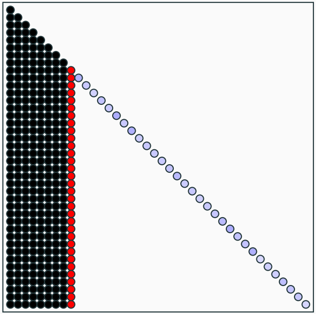




Diagonal element: 3.58e-07

Example: Pivoted Cholesky on a Kernel Matrix





Diagonal element: 1.92e-07

Improving the Decomposition

Recall interpolative decomposition:

$$A\Pi \approx C \begin{bmatrix} I & T \end{bmatrix}$$

and can keep all $|t_{ij}| \leq 2$ with right Π.

Idea: Start from pivoted QR and refine

- Compute truncated pivoted QR

$$A\Pi = QR = \begin{bmatrix} Q_1 & Q_1 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix} \approx Q_1 \begin{bmatrix} R_{11} & R_{12} \end{bmatrix}$$

- Set $C = Q_1 R_{11}$ and $T = R_{11}^{-1} R_{12}$
- For large entries ($|t_{ij}| > 2$), swap columns π_i and π_{r+j}.
- Recompute T; repeat swaps until happy.

Next Time: Topics, Parts, and NMF

What if we want a low-rank factorization with more structure?

- Non-negativity?
- Sparsity of factors?
- Normalization for a probabilistic interpretation?

Hard in general, but there are some effective approaches.

After the break: From non-negative matrix factorization to spectral topic modeling