Low-Rank Approximation

Lecture 2 – Practical methods for low-rank approximation

Leonardo Robol, University of Pisa, Italy

Cagliari, 23–27 Sep 2019

Our goal today

Assume we are given some matrix A, and we are told that it can be efficiently approximated as $A \approx UV^T$. How do we compute U, V, cheaply?

- What does cheap mean? We aim at complexities $\mathcal{O}(n)$ or $\mathcal{O}(n \log n)$, where n is the dominant size of A.
- What about the accuracy? Ideally, working in the 2-norm we would aim at getting the best approximation given by the SVD: in practice, we will only approximate it — trying to stay as close as possible.

We will present these strategies:

1. QR factorization with column pivoting.
2. Golub-Kahan-Lanczos bidiagonalization (close relation with Krylov methods).
3. Randomized methods.
4. Adaptive cross approximation.

A few numerical examples with the SVD

See: `example_svd.m`.

A few numerical examples with the SVD

See: `example_svd.m`.

Take-home message: SVD would be wonderful tool, if it were not that expensive; we shall find a way to approximate it.

QR factorization with column pivoting

- The standard QR factorization looks for an orthogonal matrix Q and an upper triangular R such that $A = QR$.
- Here Q is $m \times m$ and R is $m \times n$.
- If A is low-rank, this can take a particular form:

$$A = QR = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix}, \qquad R_{22} = 0 \text{ (or small)}.$$

QR factorization

Quick reminders:

- A QR factorization is a decomposition $A = QR$, with Q unitary, and R upper triangular.
- Given a vector v, we can always find an Householder reflector P such that $Pv = \pm\alpha e_1$, with $\alpha := \|v\|_2$.
- A can be made upper triangular with a sequence of $n-1$ Householder reflectors.

$$\underbrace{P_1 P_2 \dots P_{n-1}}_{Q^T} A = R \iff A = QR = P_{n-1} \dots P_1 R.$$

- The cost is $\mathcal{O}(m^2 n)$ for an $m \times n$ matrix. No advantage having a low-rank matrix.

Rank-revealing QR factorization

- If we order the columns in a smart way, we "detect" low-rank matrices.
- Such variants are known as rank-revealing QR factorizations.
- Several applications, among which solving (low-rank) least square problems.

Observation: if $A = QR$, and

$$Q = \begin{bmatrix} & & & \\ q_1 & q_2 & \dots & q_n \\ & & & \end{bmatrix},$$

then $\{q_1, \dots, q_k\}$ span the column span of the first k columns of A.

Why we need column permutations

Consider $A = e_n e_n^T$. Then,

- $A = I \cdot A$ is the QR factorization, since A is already upper triangular!
- If we compute the factorization by Householder reflectors, we need $n - 1$ steps before we get a good the vectors we need for the basis of the column span.
- Clearly, if we permute the n-th column into the first 1, at the first step of reduction we already have all the necessary information!

QR with column pivoting

$$A = \begin{bmatrix} & & & \\ a_1 & a_2 & \dots & a_n \\ & & & \end{bmatrix}.$$

- Compute the norms of the columns $\|a_j\|_2$, for $j = 1, \dots, n$.
- Put the column a_k with the largest norm in front.
- Compute an Householder reflector such that $Pa_k = \alpha e_1$.
- Apply it to the matrix, and continue the reduction on the trailing $(n-1) \times (n-1)$ minor.

QR with column pivoting

Final result: $A = QR\Pi$, with Π permutation.

- R has decreasing diagonal entries.
- If the rank of A is k, the method terminates in k steps.
- Cost: $\mathcal{O}(kmn)$.
- The norms of the columns can be downdated at a lower cost than recomputing them from scratch.

Norm updates

Assume we have compute the norms of the columns $\|a_j\|_2$, for $j = 1, \ldots, n$. If we apply a reflector P we end up with:

$$PA = \begin{bmatrix} Pa_1 & Pa_2 & \ldots & Pa_n \end{bmatrix} = \left[\begin{array}{c|ccc} \tilde{a}_{11} & \tilde{a}_{12} & \ldots & \tilde{a}_{1n} \\ \hline 0 & \tilde{a}_2 & \ldots & \tilde{a}_n \end{array}\right].$$

Clearly, we have

$$\|\tilde{a}_j\|_2^2 = \|Pa_j\|_2^2 - |\tilde{a}_{1j}|^2 = \|a_j\|_2^2 - |\tilde{a}_{1j}|^2.$$

If we store the square of the column norms we can get away with $\mathcal{O}(1)$ updates; we just need to be careful with cancellation!

MATLAB[1] code: `[Q,R,P] = qr(A)`.

[1] Unfortunately, MATLAB will always compute the full Q, without stopping early if it detects a low-rank of numerically low-rank matrix.

A quick look at the source code

See: `rrqr.m`

Golub-Kahan-Lanczos: computing eigenvalues of large scale matrices

To understand the Golub-Kahan approach, we first notice that

$$A = U\Sigma V^T \iff AA^T = U\Sigma^2 U^T \text{ and } A^T A = V\Sigma^2 V^T$$

and also

$$A = U\Sigma V^T \iff \begin{bmatrix} V & \\ & U \end{bmatrix} \begin{bmatrix} & A^T \\ A & \end{bmatrix} \begin{bmatrix} V & \\ & U \end{bmatrix}^T = \begin{bmatrix} & \Sigma \\ \Sigma & \end{bmatrix}$$

- There is a close connection between computing singular values of and eigenvalues of symmetric matrices. Indeed, the second formula is a reduction to a 2×2 block diagonal matrix with blocks of the form

$$\begin{bmatrix} 0 & \sigma_j \\ \sigma_j & 0 \end{bmatrix}, \quad \text{which has eigenvalues } \pm\sigma_j, \quad \text{and eigenvectors } \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ \pm 1 \end{bmatrix}.$$

- We are only interested in the largest singular values and their singular vectors. Can this be exploited?

Large-scale eigenvalue problems

- Let us focus on approximating the eigenvalues of A, i.e., solving the eigenvalue problem $Av = \lambda v$.
- we need to some structure, so we assume that we can compute $v \mapsto Av$ "fast".
- We only care about large eigenvalues.

Large-scale eigenvalue problems

- Let us focus on approximating the eigenvalues of A, i.e., solving the eigenvalue problem $Av = \lambda v$.
- we need to some structure, so we assume that we can compute $v \mapsto Av$ "fast".
- We only care about large eigenvalues.

The simplest idea is the power method. Consider the iteration

$$v_0 := \text{random vector}, \qquad v_{\ell+1} = Av_\ell.$$

If A has a single dominant eigenvalue λ_1 such that

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \dots,$$

then $v_\ell / \|v_\ell\|_2$ converges to the eigenvector relative to λ_1.

Advantages and drawbacks of the power method

- Very simple to implement, only need matrix-vector products.
- The convergence rate is given by $|\lambda_1|/|\lambda_2|$. Could be slow if the eigenvalues are not well separated!
- What if $|\lambda_1| = |\lambda_2|$? Things can go very badly ...
- What if we want more eigenvalues? We may compute λ_1 and its eigenvectors v_1, and then reapply the method to the deflated matrix

$$A_1 := A - \lambda_1 v_1 v_1^T, \qquad A_2 := A_1 - \lambda_2 v_2 v_2^T, \qquad \dots$$

Advantages and drawbacks of the power method

- Very simple to implement, only need matrix-vector products.
- The convergence rate is given by $|\lambda_1|/|\lambda_2|$. Could be slow if the eigenvalues are not well separated!
- What if $|\lambda_1| = |\lambda_2|$? Things can go very badly ...
- What if we want more eigenvalues? We may compute λ_1 and its eigenvectors v_1, and then reapply the method to the deflated matrix

$$A_1 := A - \lambda_1 v_1 v_1^T, \qquad A_2 := A_1 - \lambda_2 v_2 v_2^T, \qquad \dots$$

Note: After ℓ steps of the power method, the vector $v_\ell := A^\ell v$, so it belongs to the Krylov subspace $\mathcal{K}_\ell(A, v_0)$. Idea: instead of v_ℓ, we may take the largest eigenvalue of A projected onto $\mathcal{K}_\ell(A, v_0)$ as approximation to λ_1.

Krylov methods

High-level description of the idea:

- Compute the projection $A_\ell = Q_\ell^T A Q_\ell$.
- Compute the eigenvalues of A_ℓ and use them as approximation to the largest eigenvalues of A. If $U_\ell^T A_\ell U_\ell = D_\ell$, then the eigenvectors are approximated by $V_\ell = Q_\ell U_\ell$.

The method is harder to analyze than the power method, but is very powerful.

- The projection A_ℓ and the orthogonal basis Q_ℓ are easy to compute (recall the Arnoldi projection).
- Eigenvalues (and eigenvectors) are approximated all together, we do not need to restart from the beginning with deflation.
- How fast do the eigenvalues converge? Not so easy to say, in general.
- However, note that we can always compute the residual $\|Av_j - \lambda_j v_j\|_2$, which is an indication of the backward error.

Arnoldi method for symmetric matrices: the Lanczos iteration

When applied to symmetric matrices, the Arnoldi projection scheme is called Lanczos[2].

- The Lanczos iteration could be run "without" the reorthogonalization step.
- For improved stability, we will always run Lanczos with reorthogonalization.
- A key difference is that A_ℓ is upper Hessenberg and symmetric (being the projection of a symmetric matrix), so is tridiagonal. This is what makes it possible to derive a cheaper orthogonalization strategy.

[2] The reason is that it was originally formulated for tridiagonal reduction of symmetric matrices, and then extended to general ones for reduction to upper Hessenberg form.

Adaptation to computing the SVD

- When dealing with symmetric eigenvalue problems, we approximate

$$A \approx Q_\ell A_\ell Q_\ell^T.$$

  Such approximation is accurate if the eigenvalues go to zero quickly – and the ones in A_ℓ converge to the large eigenvalues of A.

- Working with the SVD, we need to consider two different bases:

$$A \approx Q_\ell A_\ell U_\ell^T,$$

  where Q_ℓ spans $\mathcal{K}_\ell(A, b)$ and U_ℓ spans $\mathcal{K}_\ell(A^T, Ab)$, for an appropriate vector b.

The method in practice

We iteratively construct the two bases Q, U as follows:

$$Q = \begin{bmatrix} v_1 & v_2 & \dots \end{bmatrix} \qquad U = \begin{bmatrix} w_1 & w_2 & \dots \end{bmatrix}$$

where we impose: $Av_j \in \operatorname{span}\{w_1 \dots w_j\}$ and $A^T w_j \in \operatorname{span}\{v_1 \dots v_{j+1}\}$.

- $v_1 := b/\|b\|_2$.
- w_1 needs to span Av_1, so we choose it as $w_1 := \frac{Av_1}{\|Av_1\|_2}$.
- $A^T w_1$ needs to be in $\operatorname{span}\{v_1, v_2\}$, so we choose v_2 as

$$v_2 := \frac{A^T w_1 - \langle A^T w_1, v_1 \rangle v_1}{\|A^T w_1 - \langle A^T w_1, v_1 \rangle v_1\|_2}.$$

- Av_2 must belong to $\operatorname{span}\{w_1, w_2\}$, so we choose it similarly.
- ...

Breakdown

The method might break at any step for a division by zero. When does this happen?

- A quick check shows that either we have found an invariant subspace of $A^T A$, or an invariant subspace of AA^T.
- We need to handle this condition and restart from a new vector orthogonal to the invariant subspace.
- Slightly tricky to do in practice because, instead of a division by zero, we might have a division by a small number.

Example application

Given x, y positive vectors, consider the symmetric Cauchy matrix

$$C_{ij} := \frac{1}{x_i + x_j}.$$

- This matrix has numerically low-rank (why? We will see this in a couple of days).
- Find a low-rank approximation $C \approx UV^T$ to a certain accuracy ϵ.
- For being able to apply the Golub-Kahan-Lanczos method, we need a fast matrix-vector multiplication $v \mapsto Cv$.

Hankel matrices

Let us choose $x := [1\ 2\ 3\ 4\ \ldots]$. Then, C is the Hilbert matrix:

$$C = \begin{bmatrix} \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \\ \frac{1}{3} & \frac{1}{4} & \iddots & & \\ \frac{1}{4} & \iddots & & & \\ \frac{1}{5} & & & & \end{bmatrix}$$

- A matrix constant on the antidiagonal is called Hankel.
- The product by a vector can be computed in $\mathcal{O}(n \log n)$ time.
- Clearly, it is symmetric – so the product by C^T is easy as well.

Fast matrix vector multiplication

Note that, if we assume that C is left-antitriangular:

$$Cv = \begin{bmatrix} c_n & \dots & c_0 \\ \vdots & \iddots & \\ c_0 & & \end{bmatrix} \begin{bmatrix} v_0 \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} c_0 v_n + \dots + c_n v_0 \\ \vdots \\ v_0 c_0 \end{bmatrix}$$

- The right hand side contains (part of) the coefficients of $c(z)v(z)$, where

$$c(z) = \sum_{j=0}^{n} c_j z^j, \qquad v(z) = \sum_{j=0}^{n} v_j z^j.$$

- The product of two polynomials can be evaluate efficiently by combining evaluation and interpolation using the FFT:
  1. Evaluate the polynomials $v(\xi^j)$ and $c(\xi^j)$ where ξ are roots of the unity (FFT).
  2. Compute the evaluation of the product $v(\xi_j)c(\xi^j)$.
  3. Interpolate the product by invrese FFT.

Trying it out in practice

See: `example_hankel.m`

Randomized sampling

We will now see a different technique which is also a valid choice for problems where a fast mat-vec is available:

- It is based on randomized methods – less prone to be stuck into invariant subspaces, as it might happen to the Lanczos iteration.
- Well understood convergence theory.
- Very easy to implement and adapt to particular needs.

Randomized sampling

We will now see a different technique which is also a valid choice for problems where a fast mat-vec is available:

- It is based on randomized methods – less prone to be stuck into invariant subspaces, as it might happen to the Lanczos iteration.
- Well understood convergence theory.
- Very easy to implement and adapt to particular needs.

Let us check in person the (possible) limitations of Lanczos: `example_invariant.m`.

Approximating the range

To understand randomized approximation, we concentrate on a simpler subproblem:

Range approximation problem
Given A, we want to approximate its range up to some accuracy ϵ, i.e., we want to find an orthogonal basis of a k-dimensional subspace

$$Q := \begin{bmatrix} q_1 & \dots & q_k \end{bmatrix}$$

such that, for every $v \in \mathrm{Range}(A)$, we can find x

$$\|Qx - v\|_2 \leq \|v\|_2 \cdot \epsilon.$$

Projectors

The same statement can be given in terms of projectors[3].

If we call $\mathcal{V} := \mathrm{span}(Q)$, then

$$QQ^T \text{ is a projector on } \mathcal{V}, \qquad \text{and } (I - QQ^T) \text{ on } \mathcal{V}^\perp.$$

- If Q approximates the range of A up to ϵ then $\|A - QQ^T A\|_2 \leq \|A\|_2 \cdot \epsilon$.
- The two problems are almost equivalent – we will focus on this formulation.
- The problem can be seen from the opposite perspective: if I fix the dimension k, what is the best ϵ that I can achieve?
- In the latter case, the optimal projector is given by the SVD.

[3] Here we always refer to orthogonal projections on a low-dimensional subspace

Optimal projector by SVD

Consider the SVD of A, given by:

$$A = U\Sigma V^T \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma_1 & \\ & \Sigma_2 \end{bmatrix} \begin{bmatrix} V_1 & V_2 \end{bmatrix}^T,$$

then $Q := U_1$ is the optimal projector for dimension k, and attains the accuracy $\epsilon = \sigma_{k+1}$. Why?

Proof.
By unitary invariance of the two norm, plus orthogonality of the columns in U_1 and U_2,

$$\|A - U_1 U_1^T A\|_2 = \|U^T A V - U^T U_1 U_1^T U U^T A V\|_2 = \left\| \begin{bmatrix} \Sigma_1 - \Sigma_1 & \\ & \Sigma_2 \end{bmatrix} \right\|_2 = \|\Sigma_2\|_2 = \sigma_{k+1}$$

□

- If the optimal projector is known, it is easy to recover the singular values and vector in Σ_1 (more on this later).

A simple idea

How to recover an approximate range?

- Simple idea: multiply A by random vectors.
- Consider Q_j be the orthogonal basis of $A[\omega_1 \dots \omega_j]$, with ω_j being independent Gaussian distributed vectors.
- Can we say something on $\|A - Q_j Q_j^T A\|_2$?

See: `example_randomvec.m`.

Intuition behind these facts

Let us consider the case where $\mathrm{rank}(A) = k$, and consider random vectors $\omega_1, \ldots, \omega_k$.

- Intuitively, two random vectors ω_1 and ω_2 are independent with probability 1 — to be dependent one should have $\omega_1 = \lambda\omega_2$, and this is a set of measure zero.
- This argument generalizes easily to $k \leq n$ vectors.
- Similarly, one can show that (with probability 1), none of their linear combination is in the kernel of A if they are at most k.

Using these arguments, AW with $W = [\omega_1 \ \ldots \ \omega_j]$ is a basis for the range (with probability 1), so we can reorthogonalize it and we have found Q.

- However, we are not in the "exact rank" case;
- For this purpose, to get a rank k approximation, we multiply with $k + p$ vectors — p is called the oversampling parameter.

Theorem
Let A be a real $m \times n$ matrix. Then, let $k + p \leq \min\{m, n\}$. If Q spans AW with $W = [\omega_1 \dots \omega_{k+p}]$ Gaussian random vectors as above, then

$$\mathbb{E}\left[\|A - QQ^T A\|_2\right] \leq \left(1 + \frac{4\sqrt{k+p}}{p-1}\sqrt{\min\{m,n\}}\right)\sigma_{k+1}(A).$$

- Not too far from the optimal!
- This result is only about the average case.
- $\sqrt{k}$ very similar to the results based on rank-revealing QR decompositions.

The error is small with high probability

Theorem
Let A be a real $m \times n$ matrix. Then, let $k + p \leq \min\{m, n\}$. If Q spans AW with $W = [\omega_1 \dots \omega_{k+p}]$ Gaussian random vectors as above, then

$$\mathbb{P}\left\{\|A - QQ^T A\|_2 \leq (1 + 9\sqrt{k+p}\sqrt{\min\{m,n\}})\sigma_{k+1}(A)\right\} \geq 1 - 3p^{-p},$$

under some mild hypotheses on p.

- This tells us that the average case is representative of the typical performance of the algorithm.
- With $p = 5$ we have $3p^{-p} \approx 10^{-3}$, with $p = 10$ we have $3p^{-p} = 3 \cdot 10^{-10}$.

From the range to the SVD

Assume we have a basis Q that approximately span the range of A. The SVD can be recovered as follows:

- Compute $W = Q^T A$, which has size $k \times n$.
- Obtain a reduced QR factorization $VR = W^T$, with V of size $n \times k$ and a $k \times k$ matrix R.
- Now,
$$A \approx QQ^T A = QR^T V^T.$$
- Get an SVD $R^T = U_R \Sigma V_R^T$.
- The reduced SVD can be formed by
$$A \approx QU_R \cdot \Sigma \cdot (VV_R)^T,$$
since both QU_R and VV_R have orthogonal columns.

Total cost: $\mathcal{O}(k^3 + nk^2 + k \cdot T_m)$, where T_m is the cost of a multiplication by A or A^T.

Adaptive approximation

- We have discussed the case where we know the target rank k a priori.
- Often, we don't, so we need to estimate the accuracy of our approximation.

Lemma
Let Q be an approximation for the range of A, and ω_i for $i = 1, \ldots, r$ iid Gaussian vectors. Then,

$$\|A - QQ^T A\|_2 \leq \alpha \sqrt{\frac{2}{\pi}} \max_{i=1,\ldots,r} \|A\omega_i - QQ^T A\omega_i\|_2$$

with probability at least $1 - \alpha^{-r}$.

- Note that this information is available while we add more vectors to the testing set.
- If we are not satisfied, we can use the computed products $A\omega_i$ to enlarge it.
- Alternative approaches are possible: for instance, power iterations on $(A - QQ^T A)^T (A - QQ^T A)$.

An even cheaper criterion

Since we are multiplying the vector ω_i incrementally, we can make a smarter choice. Assume we have already computed j vectors, and we have an ortohgonal basis Q_j for the span of $A[\omega_1, \ldots, \omega_j]$.

- We compute $q_{j+1} = A\omega_{j+1}$.
- We orthogonalize it against Q_j, and we renormalize:

$$\tilde{q}_j := (I - Q_j Q_j^T) q_j, \qquad \hat{q}_j := \frac{\tilde{q}_j}{\|\tilde{q}_j\|_2}.$$

- We construct the new basis

$$Q_{j+1} = [Q_j \ \hat{q}_j].$$

- If we encounter r consecutive vectors for which $\|\tilde{q}_j\|_2 \leq \tau$ – with τ chosen according to the previous Lemma! – we stop.

A closer look at the convergence bounds

Theorem
Let A be a real $m \times n$ matrix, and fix $k \geq 2$ and an oversampling parameter $p \geq 2$, with $k + p \leq \min\{m, n\}$. If Q is an orthogonal basis of $A\Omega$, with Ω an $n \times (k + p)$ Gaussian matrix, then

$$\mathbb{E}\left[\|A - QQ^T A\|_F\right] \leq \sqrt{1 + \frac{k}{p-1}} \cdot \sqrt{\sum_{j>k} \sigma_j(A)^2}$$

and

$$\begin{aligned}\mathbb{E}\left[\|A - QQ^T A\|_2\right] &\leq \left(1 + \sqrt{\frac{k}{p-1}}\right)\sigma_{k+1}(A) + e\frac{\sqrt{k+p}}{p-1} \cdot \sqrt{\sum_{j>k} \sigma_j(A)^2} \\ &\leq \left(1 + \sqrt{\frac{k}{p-1}} + e\frac{\sqrt{k+p}}{p-1}\sqrt{\min\{k+p\}}\right)\sigma_{k+1}(A)\end{aligned}$$

Slowly decaying singular values

- If the singular values decay slowly, we might have to choose a larger oversampling parameter – which can be unconvenient.
- An alternative strategy is to approximate the range of $(AA^T)^q A$, for some $q \geq 1$. These two matrices have the same range, but:

$$\sigma_j(B) = \sigma_j(A)^{2q+1}, \qquad B := (AA^T)^q A$$

- For instance, with $q = 1$,

$$AA^T A = (U\Sigma V^T)(V\Sigma U^T)(U\Sigma V^T) = U\Sigma^3 V^T.$$

- For those of you familiar with generalized matrix functions, this is nothing else than $z \mapsto z^{2q+1}$.

Accelerating the approach for dense matrices

- Assume we are given an $n \times n$ matrix A without any apparent structure, but we are told that it is rank k (numerically).
- Applying the randomized sampling would give us a rank k parametrization by $\mathcal{O}(k)$ matrix-vector products, which amounts to $\mathcal{O}(n^2 k)$ flops.

Accelerating the approach for dense matrices

- Assume we are given an $n \times n$ matrix A without any apparent structure, but we are told that it is rank k (numerically).
- Applying the randomized sampling would give us a rank k parametrization by $\mathcal{O}(k)$ matrix-vector products, which amounts to $\mathcal{O}(n^2 k)$ flops.

We are using Gaussian vectors because they make the theory "easy", but we are not forced to do so — using a structured sampling space allows faster approximation (sometimes!).

Accelerating the approach for dense matrices

- Assume we are given an $n \times n$ matrix A without any apparent structure, but we are told that it is rank k (numerically).
- Applying the randomized sampling would give us a rank k parametrization by $\mathcal{O}(k)$ matrix-vector products, which amounts to $\mathcal{O}(n^2 k)$ flops.

We are using Gaussian vectors because they make the theory "easy", but we are not forced to do so — using a structured sampling space allows faster approximation (sometimes!).

Idea: choose Ω such that $A\Omega$ is cheap to compute independently of A.

Some structured vectors

We need some vectors ω_i such that $A\omega_i$ has linear (or almost linear) cost. Some examples: itemize. A few examples!

- $\omega_i = G_1 \dots G_j e_1$, where G_i are Givens rotations.

Some structured vectors

We need some vectors ω_i such that $A\omega_i$ has linear (or almost linear) cost. Some examples: itemize. A few examples!

- $\omega_i = G_1 \dots G_j e_1$, where G_i are Givens rotations.
- A subsampled random Fourier transform — what we will consider today. Use Ω defined by:

$$\Omega = \sqrt{\frac{n}{k}} DFR, \qquad \text{where}$$

  - D is diagonal with random unimodular entries.
  - F is the FFT matrix of size n.
  - R is an $n \times k$ matrix that samples the columns at random.

The cost of computing $A\Omega$ with this second version of Ω becomes $\mathcal{O}(n^2 \log k)$, instead of $\mathcal{O}(mnk)$.

Computing a subsampled FFT

- The FFT of a vector v can be written as

$$F^{(n)}v = \left[\sum_{j=1}^{n} \xi_n^{(i-1)(j-1)n} v_j\right]_{i=1,\dots,n}$$

- If we assume $n = km$, we can rearrange the sum in a clever way so that:

$$F^{(n)}v = \left[\sum_{j_2=1}^{m} \xi_m^{(i_1-1)(j_2-1)} \cdot \xi_n^{(i_2-1)(j_2-1)} \cdot \sum_{j_1=1}^{k} \xi_k^{(i_2-1)(j_1-1)} v_{(j_1-1)m+j_2}\right]_{\substack{i_1=1,\dots,m \\ i_2=1,\dots,k}}$$

- The right most summation requires $\mathcal{O}(km \log k)$, since it has to be applied to m vectors. Then, there is a scalar multiplication, and if we want $\mathcal{O}(k)$ entries we just need a total cost of $\mathcal{O}(km \log k + km)$.

A few more comments

- This strategy works quite well in practice, even though it is less straightforward to implement.
- In general, a higher oversampling is required, and it is only convenient for medium ranks – for small ranks the classical approach turns out to be more useful!
- All the strategy that we have described are easily parallelizable! This is in contrast with Lanczos or the rank-revealing QR – which are inherently sequential methods.

Recompression

Quite often, one ends up with a low-rank parametrization $A \approx UV^T$ which is redundant, i.e., not minimal. How do we recompress it, up to some truncation tolerance ϵ?

- Compute economic-size QR factorizations of U, V:

$$Q_U R_U = U, \qquad Q_V R_V = V.$$

- Find the SVD of $R_U R_V^T = \hat{U} S_K \hat{V}^T$. Then,

$$(Q_U \hat{U}) S (Q_V \hat{V})^T = UV^T$$

  is a reduced SVD of UV^T, so we can perform a truncation dropping the small singular values in S, and the related singular vectors.

- Note that this allows for truncation in both the 2 and the Frobenius norm!

Cross approximation

It's not always possible to design a fast matrix-vector product "easily". Therefore, it is of interest to obtain good low-rank approximations by just "sampling" the matrix.

Cross approximation

It's not always possible to design a fast matrix-vector product "easily". Therefore, it is of interest to obtain good low-rank approximations by just "sampling" the matrix.

- The usual approach in these cases is cross approximation.
- Based on selecting a few rows and columns – and ignoring the rest.
- Can be very close to optimality.

A simple example

Assume that the $m \times n$ matrix A has rank k, then:

- We select two sets of indices with cardinality k

$$I \subseteq \{1, \ldots, m\}, \qquad J \subseteq \{1, \ldots, n\}.$$

  such that $A(I, J)$ is invertible.
- Then, $A = A(:, J)A(I, J)^{-1}A(I, :)$.

Note that:

- Such matrix always exists — if A has rank k at least one $k \times k$ minor needs to have nonzero determinant.
- Equality holds for the exact rank. What can we say about stability?
- We use MATLAB notation for the indices.

Approximate rank

Assume $A = A_0 + E$, with:

- A_0 of rank k;
- $\|E\|_2 \leq \epsilon$.

Suppose we select an invertible $k \times k$ matrix; then,

$$\|A - A(:, J)A(I, J)^{-1}A(I, :)\|_2 \sim \mathcal{O}(\epsilon \cdot \|A(I, J)^{-1}\|_2^2 \cdot \|A\|_2^2).$$

Proof.
Clearly, for small enough perturbations $A_0(I, J)$ will be invertible as well, so we have that $A_0 = A_0(:, J)A_0(I, J)^{-1}A_0(I, :)$. Then, we perform first order expansions:

$$\begin{aligned} A(:, J)A(I, J)^{-1}A(I, :) &= (A_0(:, J) + E(:, J)) \cdot (A_0(I, J) + E(I, J))^{-1} \cdot (A_0(I, :) + E(I, :)) \\ &\doteq \dots \end{aligned}$$

Putting the pieces together yields the desired bound. □

Choosing a good submatrix

Having a look at the bound hints at the features for the "optimal submatrix $A(I, J)$":

$$\|A - A(:, J)A(I, J)^{-1}A(I, :)\|_2 \sim \mathcal{O}(\epsilon \cdot \|A(I, J)^{-1}\|_2^2 \cdot \|A\|_2^2).$$

- We aim at finding a well-conditioned submatrix.
- A similar concept is maximizing the volume of the submatrix $A(I, J)$, i.e., finding the index sets I, J such that $|\det A(I, J)|$ is maximum.
- The latter problem has a very bad complexity in general (NP-hard) – but has many studied by many people (most notably, D. Knuth).

Maximum volume submatrices

The following result relates the property of being a submatrix of maximum subvolume and being an accurate low-rank approximation.

Theorem
Let A be an $m \times n$ matrix, and $A(I, J)$ an $r \times r$ submatrix of maximum subvolume. Then:

$$\|A - A(:, J)A(I, J)^{-1}A(I, :)\|_C \leq (r + 1)\sigma_{r+1}(A).$$

- $\|\cdot\|_C$ is the Chebyshev norm – defined as the maximum of the absolute value of the entries.
- Note that the rows in I and the columns in J are approximated exactly.
- Deriving bounds in unitarily invariant norms requires a few more steps.

A model problem

Let U be a $m \times r$ matrix with orthogonal columns, i.e.,

$$U \in \mathbb{R}^{m \times r}, \qquad U^T U = I.$$

- The optimal cross approximation problem for U is as follows: choose a submatrix $\hat{U}$ such that its inverse is as small as possible.
- Denote by $M(U)$ the set of $r \times r$ submatrices of U.

Lemma
Let $\tau_U := \min_{\hat{U} \in M(U)} \|\hat{U}^{-1}\|_2$, *where* U *is an* $m \times r$ *unitary matrix as above. Then,* $\tau_U \leq \sqrt{1 + r(m-r)}$.

- The property of being unitary imposes that the bound only depends on m and r!

Proof for the model problem

Let $\hat{U}$ be the matrix of maximum volume. Without loss of generality, we can assume that $\hat{U}$ is in the first r rows of U, i.e.,

$$U = \begin{bmatrix} \hat{U} \\ W \end{bmatrix} \implies U\hat{U}^{-1} = \begin{bmatrix} I \\ V \end{bmatrix}, \qquad V := W\hat{U}^{-1}.$$

Claim: The entries of V satisfies $|V_{ij}| \leq 1$. Indeed, if $|V_{ij}| > 1$ for some i, swapping rows i and $i + r$ in U gives us a matrix of volume larger than $\hat{U}$.

Hence, we conclude noting that

$$\|\hat{U}^{-1}\|_2 = \|U\hat{U}^{-1}\| = \left\| \begin{bmatrix} I \\ V \end{bmatrix} \right\|_2 \leq \sqrt{\|I\|_2^2 + \|V\|_2^2},$$

and bounding $\|V\|_2^2 \leq \|V\|_F^2 \leq r(m + r)$.

Extension to the general case

- The previous result covers a very particular case of matrices: $m \times r$ orthogonal matrices of rank r.
- We would like to handle the general case of $m \times n$ matrices.
- In most situation, we will have only approximate rank r.

Theorem
Let A be any matrix such that $A = A_0 + E$, where A has rank r, and $\|E\|_2 \leq \epsilon$. Then, there exist choise of index sets I, J, of cardinality r, such that[4]

$$\|A - A(:, J)A(I, J)^{-1}A(I, :)\|_2 \leq (1 + 2\sqrt{r}\sqrt{\max\{m, n\}})\epsilon.$$

[4]The best result is actually sharper than this — this is only what we will prove today, for simplicity

Proof

We build the economy size SVD of A_0, which has rank r:

$$A_0 = U\Sigma V^T, \qquad U \in \mathbb{R}^{m\times r}, V \in \mathbb{R}^{n\times r}, \qquad \Sigma \in \mathbb{R}^{r\times r}.$$

Both U and V contain submatrices $\hat{U}, \hat{V}$ of maximum volume, which satisfy:

$$\|\hat{U}^{-1}\|_2 \leq \sqrt{1 + r(m-r)}, \qquad \|\hat{V}^{-1}\|_2 \leq \sqrt{1 + r(n-r)}.$$

Consider I, J indices such that $\hat{U} = U(I,:)$ and $\hat{V} = V(J,:)$. If we use these for the cross approximation we have

$$\begin{aligned} A - A(:,J)A(I,J)^{-1}A(I,:) &= A_0 + E - (A_0(:,J) + E_R)(A_0(I,J) + E_{RC})^{-1}(A_0(I,:) + E_C) \\ &\doteq A_0 - A_0(:,J)A_0(I,J)^{-1}A_0(I,:) + E - E_R A_0(I,J)^{-1}A_0(I,:) \\ &\quad - A(:,J)A_0(I,J)^{-1}E_C + A_0(:,J)A_0(I,J)^{-1}E_{CR}A_0(I,J)^{-1}A_0(I,:). \end{aligned}$$

Remaining steps: we have $A_0 - A_0(:,J)A_0(I,J)^{-1}A_0(I,:) = 0$; then, write $A_0 = U\Sigma V^T$, $A_0(I,:) = \hat{U}\Sigma V^T$, $A(:,J) = U\Sigma\hat{V}^T$, and $A_0(I,J) = \hat{U}\Sigma\hat{V}^T$, and take norms.

Optimality in the Frobenius norm

A result in the Frobenius norm can be derived from the element-wise bound that we have stated previously. Recall that:

$$\|A - A(:, J)A(I, J)^{-1}A(I, :)\|_C \leq (r + 1)\sigma_{r+1}(A),$$

where $\|\cdot\|_C$ denotes the Chebyshev norm, i.e., the maximum of the absolute values of the entries. Then,

$$\|A - A(:, J)A(I, J)^{-1}A(I, :)\|_F \leq \sqrt{(m - r)(n - r)}(r + 1)\sigma_{r+1}(A),$$

just by summing up all the $(m - r)(n - r)$ nonzero entries of the residual.

- Note that on the right hand side we have $\sigma_{r+1}(A)$, and instead we would like to have $\sqrt{\sigma^2_{r+1}(A) + \ldots \sigma^2_{\min\{m,n\}}(A)}$.
- Alternatively, one could redo the previous proof for $\|\cdot\|_F$ instead of $\|\cdot\|_2$.

A few comments

- These results justify the applicability of cross approximation methods.
- However, there is some loss of optimality as the rank increases.
- Possible solution: try to approximate rank r with a bigger matrix, and not just r rows and columns.
- Idea very similar to the oversampling for randomized methods.

Projective inverses

Suppose I select a submatrix $A(I, J)$ with $\#I > r$ and $\#J > r$, and possibly also $\#I \neq \#J$. How do I get a rank r cross approximation for A? I need:

$$A \approx A(:, J)A_r^\dagger(I, J)A(I, :), \qquad A_r^\dagger(I, J) \text{ of rank } r.$$

- Recall that we are trying to select the "inverse" in the middle to have norm as small as possible.
- We need that, if $\#I = \#J = r$ we go back to the usual case $A_r^\dagger(I, J) = A(I, J)^{-1}$.
- Most natural definition: consider the projective inverse:

$$A_r^\dagger(I, J) := V \mathrm{diag}(\sigma_1^{-1}, \ldots, \sigma_r^{-1}, 0, \ldots, 0)U^T,$$

  where $A(I, J) = U\Sigma V^T$ is the SVD. Essentially a Moore-Penrose pseudo-inverse with fixed rank r.
- Definition is valid for rectangular matrices as well.

How to choose the submatrix $A(I, J)$?

- Previously, we knew that we had to select the maximum volume submatrix.
- Now, we need a new definition of "volume": the r-projective volume:

Definition
Given $A \in \mathbb{R}^{m \times n}$, we define the *$r$-projective volume* as the product of its first r singular values:

$$\mathcal{V}_r(A) := \sigma_1(A) \dots \sigma_r(A).$$

- If A has any dimension smaller than r, or rank smaller than r, then, $\mathcal{V}_r(A) = 0$.
- If A is $r \times r$, then $\mathcal{V}_r(A) = |\det(A)|$ (the classical volume).
- We would like this volume to have the usual good properties.

The model problem, again

Lemma
Let U be an $m \times r$ matrix with orthogonal columns, i.e., $U^T U = I$. Consider the set of $s \times r$ submatrices of U, with $s \geq r$, defined by $M_s(U)$. Then, if $\hat{U} \in M_s(U)$ has maximum r-projective volume,

$$\|U_r^\dagger\|_2 \leq \sqrt{1 + \frac{(m-s)r}{s-r+1}}.$$

- Proof very similar to the previous result for the usual subvolume.
- Note that if we choose $s = r$, we recover the same result as before:

$$\|U_r^\dagger\|_2 \leq \sqrt{1 + (m-r)r}.$$

- If, instead, we choose $s = 2r - 1$, then

$$\|U_r^\dagger\|_2 \leq \sqrt{1 + (m-2r+1)}.$$

  The bound gets better as r increases!

Theorem
Let A be any $m \times n$ matrix decomposable as $A = A_0 + E$, where A_0 has rank r, and $\|E\|_2 \leq \epsilon$. Choose $s, t \geq r$. Then, there exist I with cardinality s and J with cardinality t such that[5]

$$\|A - A(:, J)A_r^\dagger(I, J)A(I, :)\|_2 \leq \left(1 + \sqrt{1 + \frac{(m-s)r}{s-r+1}} + \sqrt{1 + \frac{(n-t)r}{t-r+1}}\right)\epsilon.$$

- If we choose $s = t = r$, we obtain again the old result.
- If we choose $s = t = 2r - 1$, then the constant can be bounded by

$$\|A - A(:, J)A_r^\dagger(I, J)A(I, :)\|_2 \leq \left(1 + 2\sqrt{\max\{m, n\}}\right)\epsilon.$$

The dependency on r has disappeared!

[5] This result has been made slightly less sharp for readability.

Proof of the result

- The proof is completely analogous to the "old" one.
- Compute the SVD of A_0.
- Select the maximum r-projective volume submatrices $\hat{U}$ and $\hat{V}$ in the SVD bases.
- Use the corresponding rows and cols for cross approximation.

Working in the Chebyshev norm

Recall that for the Chebyshev norm, the one defined as the maximum of the absolute value of the entries, we had the following result:

$$\|A - A(:, J)A(I, J)^{-1}A(I, :)\|_C \leq (r + 1)\sigma_{r+1}(A).$$

- This result can be improved working with projective inverses as well.
- We would like to get rid of the dependency on the rank.

Theorem
Let A be any matrix, and $A(I, J)$ its $s \times t$ submatrix with maximum r-projective volume. Then,

$$\|A - A(:, J)A_r^{\dagger}(I, J)A(I, :)\|_C \leq \sqrt{1 + \frac{r}{s - r + 1}}\sqrt{1 + \frac{r}{t - r + 1}}.$$

Comments on the theorem

Theorem
Let A be any matrix, and $A(I, J)$ its $s \times t$ submatrix with maximum r-projective volume. Then,

$$\|A - A(:, J)A_r^{\dagger}(I, J)A(I, :)\|_C \leq \sqrt{1 + \frac{r}{s - r + 1}}\sqrt{1 + \frac{r}{t - r + 1}}.$$

- As usual, choosing $s = t = r$ gives the bound we had previously.
- If we choose $s = t = 2r - 1$ then the dependency on the rank disappears, and the constant becomes 2!.
- Intermediate way possible: choose $s = 2r - 1$, and $t = r$. Then, the constant grows is $\sqrt{2}\sqrt{1 + r}$.

From theory to practice

- The theory is nice, but finding a maximum volume (or maximum r-projective volume) matrix is NP-hard — how shall we deal with the problem?

We consider two possible strategies:

1. A practical heuristic algorithm, that deals with rectangular $m \times r$ matrices (known as `maxvol`).
2. A heuristic that rephrases the problem in a different way, and allows to characterize the growth of $\|A(I, J)\|_2^{-1}$ with something that we know (and we don't completely understand) since a long time.

Dominant submatrices

Definition
Let A be an $m \times r$ matrix. Then, an $r \times r$ submatrix A_0 is *dominant* if, up to permuting the rows to put A_0 on top,

$$AA_0^{-1} = \begin{bmatrix} I \\ V \end{bmatrix}, \qquad |V_{ij}| \leq 1.$$

Lemma
If A_0 has maximum volume, then it is dominant.

Proof.
Construct AA_0^{-1}. Note that the property of being of maximum volume is transferred by the submatrices of A to the ones AA_0^{-1}, since all the sub-determinant are just multiplied by $\det(A_0^{-1})$.

If, by contradiction, there exists $|V_{ij}| > 1$, swap rows i and $i + r$. Then, the top matrix has now determinant V_{ij}, and therefore the previous top submatrix was not of maximum volume, which leads to a contradiction. □

The algorithm `maxvol`

A maximum volume submatrix needs to be dominant: therefore, we can relax the problem into finding a dominant submatrix:

1. Select a starting submatrix A_0.
2. Compute
$$AA_0^{-1} = \begin{bmatrix} I \\ V \end{bmatrix} \quad \text{(up to permutation)}.$$
3. Find the element V_{ij} in V with largest modulus.
4. Update A_0 by swapping its row i with the one in position $i + r$, containing V_{ij}. Go back to step 2.

This algorithm generates a sequence of submatrices of (strictly) increasing volume – so it must converge (since the volumes are bounded). Stopping criterion can be chosen looking at the maximum of $|V_{ij}|$.

Speeding up `maxvol`

- The computational bottleneck in `maxvol` is the computation of $A_0 A_0^{-1}$, which in principle requires $\mathcal{O}(mr^2)$ flops at every step.
- However, notice that at every step A_0 changes just by one row – so by a rank 1 update. We can make use of the Sherman-Morrison formula that updates an $r \times r$ inverse in $\mathcal{O}(r^2)$:

$$(A_0 + uv^T)^{-1} = A_0^{-1} - \frac{A_0^{-1} uv^T A_0^{-1}}{1 + v^T A_0^{-1} u}.$$

- In principle, one may use the rank 1 update of a QR factorization as well, for improved stability – but the matrices A_0 get increasingly well-conditioned, so this is not so important.
- With this change, one step costs $\mathcal{O}(mr)$.

Adaptive cross approximation

Ufortunately, `maxvol` only works for tall and skinny matrices.

- Other heuristics available for finding maximum volume submatrices; often based as `maxvol` as a starting step.
- We consider another heuristic, that tries to build the matrix $A(I, J)$ one step at a time.
- It is called adaptive cross approximation (or ACA), and was proposed by Bebendorf around 2000 for functional approximation (or probably even before).

ACA in a nutshell

At every step, we want to have $A = A_k + R_k$, where A_k is the approximation, and R_k the residual. We start by $A_0 = 0$, and $R_0 = A$.

1. Find a good maximum volume submatrix of size 1×1; in other words, find a large element $|A_{ij}|$ in $|A|$.
2. Use element (i, j) as pivot, and build the rank 1 approximation

$$A_1 := A(:, j)A(i, j)^{-1}A(i, :), \qquad R_1 = A - A_1.$$

3. Replace A with R_1, and repeat until the residual gets sufficiently small.

Does this algorithm sound familiar? It should ...

A “stupid” ACA

Choosing the largest element in A is costly – we need to make a suboptimal choice. Assume we make a fixed selection, and we choose as pivot at the r-th step the element in position (r, r).

Then,

$$A_1 = A(:,1)A(1,1)^{-1}A(1,:), \qquad R_1 = A - A(:,1)A(1,1)^{-1}A(1,:).$$

Then,

$$R_1 = \begin{bmatrix} 0 & 0 & \dots & 0 \\ 0 & \tilde{a}_{22} & \dots & \tilde{a}_{2n} \\ \vdots & \vdots & & \vdots \\ 0 & \tilde{a}_{m2} & \dots & \tilde{a}_{mn} \end{bmatrix}, \qquad \tilde{a}_{ij} = a_{ij} - a_{i1}a_{11}^{-1}a_{1j}.$$

A “stupid” ACA

Choosing the largest element in A is costly – we need to make a suboptimal choice. Assume we make a fixed selection, and we choose as pivot at the r-th step the element in position (r, r).

Then,

$$A_1 = A(:,1)A(1,1)^{-1}A(1,:), \qquad R_1 = A - A(:,1)A(1,1)^{-1}A(1,:).$$

Then,

$$R_1 = \begin{bmatrix} 0 & 0 & \dots & 0 \\ 0 & \tilde{a}_{22} & \dots & \tilde{a}_{2n} \\ \vdots & \vdots & & \vdots \\ 0 & \tilde{a}_{m2} & \dots & \tilde{a}_{mn} \end{bmatrix}, \qquad \tilde{a}_{ij} = a_{ij} - a_{i1}a_{11}^{-1}a_{1j}.$$

R_1 is exactly the matrix obtained after 1 step of LU factorizatino without pivoting – at least in the trailing part.

An important connection

Lemma
If we perform the "stupid" pivoting for the ACA that we had in the previous slide, then if $A = LU$ with no pivoting,

$$\begin{aligned} A_k &= L(:, 1:k)U(1:k, :) \\ &= L(:, 1:k)U(1:k, 1:k)U(1:k, 1:k)^{-1}L(1:k, 1:k)^{-1}U(1:k, :) \\ &= A(:, 1:k)[A(1:k, 1:k)]^{-1}A(1:j, :). \end{aligned}$$

- Therefore, ACA with no pivoting is equivalent to cross approximation selecting the leading matrix as approximant.
- The quality of the approximation depends on the growth of $A(1:k, 1:k)^{-1}$ — very much related to the growth factor of LU!
- For the cases where LU with no pivoting is known to work well, then ACA with no particular pivoting works as well (positive definite matrices, M-matrices, diagonally dominant matrices, ...).

General matrices

For a general A, the LU factorization has a growth factor, defined as

$$\rho_k := \frac{\|L(1:k,1:k)\|_\infty \cdot \|U(1:k,1:k)\|_\infty}{\|A(1:k,1:k)\|_\infty},$$

which grows exponentially, and that is the typical behavior. It allows to control the norm of the selected core by $\|A(I,J)^{-1}\| \leq 4^k \rho_k$ for complete pivoting.

- If we perform a selection of the largest pivot on the row/col, we have LU with column/row pivoting: the worst case growth factor is still exponential, but is very rare in practice!
- If we look for the largest pivot, we have LU with complete pivoting: growth factor very slow, conjecture to be equal to k, but was disproved ≈ 30 years ago. Still very small in practice.

Summary

- Cross approximation very powerful, but related to a difficult problem.
- In practice, ACA is a good enough heuristic, works in most cases.
- It is slightly less reliable than Lanczos or randomized sampling, but only requires a few entries of the matrix!
- The optimal low-rank approximation strategy depends on the features of your problem:
  - Is A small size ($\min\{m, n\} \leq 500$)? Then, do a rank-revealing QR or even SVD if one of the two dimensions is small.
  - Is a fast matrix-vector product by A and A^T available? Then go with randomized sampling or Lanczos.
  - Are single entries of A easily computable? Then, use ACA.
- If all the above assumptions fail — one has to design a custom procedure for the case at hand — but they cover 99.9% of the cases of practical interest.

Example

See: `example_aca.m`

Principal Component Analysis

One of applications of low-rank approximation is the so-called principal component analysis, or PCA. Assume we have ℓ independent random variables

$$X_1(\omega), X_2(\omega), \ldots, X_\ell(\omega).$$

- We do not known these variables explicitly.
- Instead, we are given the possibility to take some samples for a small set of events.
- Unfortunately, we cannot measure the variables directly, but only their effects, which are given by certain linear combinations:

$$\begin{aligned} Y_1(\omega) &= M_{1,1}X_1(\omega) + \ldots + M_{1,\ell}X_\ell(\omega) \\ &\vdots \\ Y_m(\omega) &= M_{m,1}X_1(\omega) + \ldots + M_{m,\ell}X_\ell(\omega) \end{aligned}$$

The observed matrix

Given the sample points $\omega_1, \ldots, \omega_n$, we measure a matrix A_{ij} with entries $A_{ij} = Y_i(\omega_j)$:

$$A = \begin{bmatrix} M_{1,1} & \ldots & M_{1,\ell} \\ \vdots & & \vdots \\ M_{m,1} & \ldots & M_{m,\ell} \end{bmatrix} \begin{bmatrix} X_1(\omega_1) & \ldots & X_1(\omega_n) \\ \vdots & & \vdots \\ X_\ell(\omega_1) & \ldots & X_\ell(\omega_n) \end{bmatrix}.$$

- In the typical situation, one has $m, n \gg \ell$.
- The matrix X has rank (at most) ℓ, and therefore A has a low-rank structure.
- Identifying the structure allows to determine the important variables in the solution — restricting the dimension space.

Another viewpoint

- We can think of having m correlated variables.
- PCA recovers another representation of the same samples, where all the variables are uncorrelated.
- The variables are ordered so they contribute less and less to the variance.

See: `example_pca.m`

Matrix completion

There is a low-rank approximation problem that is slightly different from the ones we have seen as of now:

- Assume that a few entries of a large matrix A are given;
- We look for the lowest degree matrix that coincide with A on the given entries.
- Often denoted as low-rank matrix completion.

Several possible solutions: we discuss the one in the paper "Low-rank matrix completion by Riemannian optimization", by B. Vandereycken.

Some notation

Let

$$\Omega = \{(i_1, j_1), \ldots, (i_s, j_s)\}$$

be the set of known indices, and:

$$[P_\Omega(A)]_{ij} := \begin{cases} A_{ij} & \text{if}(i,j) \in \Omega \\ 0 & \text{otherwise} \end{cases}.$$

The problem statement can be given as:

$$\text{minimize } \operatorname{rank}(X), \qquad \text{subject to } P_\Omega(A) = P_\Omega(X) \iff P_\Omega(X - A) = 0,$$

where A is the given data matrix. This problem is NP-hard.

We might relax the formulation to cope with noise:

$$\text{minimize } \operatorname{rank}(X), \qquad \|P_\Omega(X - A)\|_2 \leq \epsilon$$

Another formulation

We consider another formulation that is better suited to be solved numerically:

$$\begin{cases} \min \|P_\Omega(X - A)\|_F^2 \\ \operatorname{rank}(X) = k \end{cases},$$

where k is chosen a priori.

- Rank k matrices form a smooth (Riemannian) manifold.
- If we see the set of rank k matrices as the ambient space, this is a an unconstrained minimization problem over a Riemannian manifold.
- Can be solved by any minimization method (for instance, gradient descent, or Newton-like methods).

Riemannian geometry in a nutshell

Smooth manifold:

- Locally similar to $\mathbb{R}^n$ through a diffeomorphism.
- Can compute tangent space $T_X(\mathcal{M})$ at any point, and construct the tangent bundle $T\mathcal{M}$.

Riemannian structure:

- At any point, one has $g(x, y)$ definite bilinear form over the tangent space $T_X(\mathcal{M})$.
- The above scalar product depends smoothly on the point.

Gradient:

- Given a smooth function $f : \mathcal{M} \to \mathbb{R}$, the gradient is a map from the manifold to the tangent bundle, such that

$$g(\nabla f(X), \xi) = Df_X[\xi],$$

where Df_X is the directional derivative at X, in the direction ξ.

Embedded manifolds

$\mathbb{R}^n$ is a Riemannian manifold with the usual scalar product as metric. Our manifold is a submanifold of the $m \times n$ matrices, which are isomorphic to $\mathbb{R}^{mn}$ with the scalar product $g(X, Y) = \mathrm{tr}(X^T Y)$.

Not a special case!

Theorem (Nash, 1956)
All the Riemannian manifolds admit an isometric embedding into $\mathbb{R}^N$, for sufficiently large N.

We need to move in a direction: possible through a retraction map; $R : T\mathcal{M} \to \mathcal{M}$ is a retraction iff

- $R((X, 0)) = X$;
- $DR((X, 0))[0, \xi] = \xi$.

Intuitively, it goes along geodesics, i.e. it locally approximate the exponential map.

Embedded manifolds (gradient)

Given $f : \mathcal{M} \to \mathbb{R}$, the gradient is given by

- Computing the gradient in the ambient manifold.
- Projecting it onto the manifold.

In our case:

$$\nabla \|P_\Omega(X - A)\|_F^2 = 2P_\Omega(X - A).$$

Very easy if we know how to compute the selected entries, and we do. However, we need to project it back onto the tangent space of rank k matrices.

The tangent space of $\mathcal{M}_k$

If $X = U\Sigma V^T$, then the tangent space can be described as:

$$T_X\mathcal{M}_k := \left\{ \begin{bmatrix} U & U^\perp \end{bmatrix} \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & 0 \end{bmatrix} \begin{bmatrix} V & V^\perp \end{bmatrix} \right\}$$

for arbitrary M_{ij}. Therefore, the projection onto $T_X\mathcal{M}_k$ is defined as:

$$P_{T_X\mathcal{M}_k}(Y) := (I-UU^T)Y(I-VV^T) = UU^TYVV^T + (I-UU^T)YVV^T + UU^TY(I-VV^T).$$

- Very easy to compute for Y already in factorized form.
- Proof: just look at first order perturbations that preserve the rank k structure.

General structure of the gradient descent

- Given some initial point X_0, compute gradient ∇X.
- Perform line search along that direction.
- Choose next point, and iterate.

In practice, the algorithm uses conjugate gradient instead of plain gradient descent: need to compare vectors in different tangent space. This can be done with parallel transport, the unique map that moves the tangent space smoothly along a path compatibly with the metric $g(\cdot, \cdot)$.

In an embedded manifold the parallel transport is given by "translation + projection".

Summary

How do I approximate $A \approx UV^T$?

- Is A small size ($\min\{m, n\} \leq 500$)? Then, do a rank-revealing QR or even SVD if one of the two dimensions is small.
- Is a fast matrix-vector product by A and A^T available? Then go with randomized sampling or Lanczos.
- Are single entries of A easily computable? Then, use ACA.
- Do I know only certain entries of A, and I am trying to complete the rest keeping the rank low? Riemannian optimization or minimization of the nuclear norm ($:=$ sum of singular values, we have not covered this).