1 23rd of November 2018 — F. Poloni

1.1 Gaussian elimination on symmetric matrices

Do you recall?

In Gaussian elimination we had A and we multiplyed it by L_1 in order to get a big chunk of 0s in the first column

$$\begin{pmatrix} 1 & & & & \\ * & 1 & & & \\ * & & 1 & & \\ * & & & 1 & \\ * & & & & 1 \end{pmatrix} \begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{pmatrix} = \begin{pmatrix} * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \end{pmatrix},$$

Let us consider an upgrade of Gaussian elimination in the case of $A \in S(m, \mathbb{R})$.
Let us see what happens if we multiply $L_1 A$ on the right by the transpose of L_1:

Step 1:

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

Step 2:

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

Step m:

$$L_{m-1} L_{m-2} \dots L_1 A L1^T \dots L_{m-2}^T L_{m-1}^T = D,$$

where D is diagonal, or

$$A = L_1 L_2 \dots L_{m-1} D L_{m-1}^T \dots L_2^T L_1^T = LDL^T.$$

Observation 1.1 (Stroke of luck)**.** *Notice that the stroke of luck of* **??** *holds in this case too, hence we pay nothing to compute matrix L.*

$$\begin{bmatrix} 1 & & & & \\ -a_2 & 1 & & & \\ -a_3 & & 1 & & \\ -a_4 & & & 1 & \\ -a_5 & & & & 1 \end{bmatrix}^{-1} \cdot \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & -b_3 & 1 & & \\ & -b_4 & & 1 & \\ & -b_5 & & & 1 \end{bmatrix}^{-1} \cdot \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & -c_4 & 1 & \\ & & -c_5 & & 1 \end{bmatrix}^{-1} \cdot \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & -d_5 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & & & & \\ a_2 & 1 & & & \\ a_3 & b_3 & 1 & & \\ a_4 & b_4 & c_4 & 1 & \\ a_5 & b_5 & c_5 & d_5 & 1 \end{bmatrix}$$

Theorem 1.1 (Symmetric Gaussian elimination)**.** *Let $A \in S(m, \mathbb{R})$ such that during Gaussian elimination we don't encounter any 0 pivot. A admits a factorization $A = LDL^T$, where L is lower triangular with ones on its diagonal, and D is diagonal.*

A Matlab implementation of symmetric Gaussian elimination is shown in Algorithm 1.1.

Algorithm 1.1 Symmetric Gaussian factorization, Matlab implementation.

```
function [L, D] = ldl_factorization(A)
  m = size(A, 1);
  L = eye(m); D = zeros(m);
  for k = 1:m-1
    D(k, k) = A(k, k);
            L(k+1:end, k) = A(k+1:end, k) / A(k, k);
            A(k+1:end, k+1:end) = A(k+1:end, k+1:end) ...
        - L(k+1:end, k) * A(k, k+1:end);
  end
  D(m, m) = A(m, m);
```

It is possible to make an optimization of this algorithm: since A is supposed to be symmetric, we only need to update the lower triangular part of A, since the rest is mirrored by symmetry, hence the computational complexity is half the one of Gaussian elimination.

This algorithm is not stable, exaclty like the one on non symmetric matrices. Pivoting may be performed in order to augment stability. It comes without saying that the row swap should be done consistently on the columns to preserve symmetry.

Of course there are some matrices (like the ones with all 0s on the diagonal) that cannot be "pivoted". There are some libraries that permute the rows and the columns in order to get better pivots. As an example, Matlab's `[L, D, P] = ldl(A)` produces matrices such that $P^T AP = LDL^T$, where D may have 2×2 diagonal blocks.

Do you recall?

We recall the definition of **positive definite matrix** $A \in M(m, \mathbb{R})$: all its eigenvalues are strictly positive. In other words, A is positive definite if $\forall z \neq 0 \in \mathbb{R}^m \; z^T Az > 0$.

Lemma 1.2. *In the context of positive definite matrices the following holds:*

1. *Let A be a symmetric matrix. A is positive definite if and only if MAM^T is so, for some invertible $M \in M(m, \mathbb{R})$. Formally, $\forall A \in S(m, \mathbb{R})$ s.t. $A \succ 0 \Leftrightarrow \exists M \in GL(m, \mathbb{R})$ s.t. $MAM^T \succ 0$;*

2. *Let A a symmetric positive definite matrix such that $A = \left[\begin{smallmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{smallmatrix}\right]$, then A_{11} and A_{22} are, too. Formally, $\forall A \in S(m, \mathbb{R})$ s.t. $A \succ 0$ and $A = \left[\begin{smallmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{smallmatrix}\right] \Rightarrow A_{11} \succ 0$ and $A_{22} \succ 0$.*

Proof.

1.

   ⇒) $A \in S(m, \mathbb{R})$ and $A \succ 0 \Longrightarrow MAM^T \in S(m, \mathbb{R})$ and $MAM^T \succ 0$.
   Take $z \in \mathbb{R}^m$, $z \neq 0$ $z^T MAM^T z = y^T Ay > 0$, where we performed a variable change $y = M^T z$. Notice that $y \neq 0$ because M is invertible (and $ker(M) = \{0\}$). The symmetry of the matrix MAM^T follows from $(MAM^T)^T = M^{T^T} A^T M^T = MAM^T$;

   ⇐) $MAM^T \in S(m, \mathbb{R})$ and $MAM^T \succ 0 \Longrightarrow A \in S(m, \mathbb{R})$ and $A \succ 0$.
   This proof follows from the previous arrow, where the substitution is $z = M^{-1}y$.

2. $A = \left[\begin{smallmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{smallmatrix}\right]$ positive definite $\Longrightarrow A_{11}$ and A_{22} are positive definite too.

   Since A is positive definite, its scalar product is greater than zero with all the vectors in $\mathbb{R}^m$.

   A_{11}) Let us take $\mathbf{z} = \left[\begin{smallmatrix} \mathbf{z}_1 \\ 0 \end{smallmatrix}\right]$.
   $\left[\begin{smallmatrix} \mathbf{z}_1{}^T & 0 \end{smallmatrix}\right] \cdot \left[\begin{smallmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{smallmatrix}\right] \cdot \left[\begin{smallmatrix} \mathbf{z}_1 \\ 0 \end{smallmatrix}\right] = \mathbf{z}_1{}^T A_{11} \mathbf{z_1} > 0$, $\forall \mathbf{z}_1 \in \mathbb{R}^{sizeof A_{11}}$

   A_{22}) Let us take $\mathbf{z} = \left[\begin{smallmatrix} 0 \\ \mathbf{z}_2 \end{smallmatrix}\right]$.
   $\left[\begin{smallmatrix} 0 & \mathbf{z}_2{}^T \end{smallmatrix}\right] \cdot \left[\begin{smallmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{smallmatrix}\right] \cdot \left[\begin{smallmatrix} 0 \\ \mathbf{z}_2 \end{smallmatrix}\right] = \mathbf{z}_2{}^T A_{22} \mathbf{z_2} > 0$, $\forall \mathbf{z}_2 \in \mathbb{R}^{sizeof A_{22}}$

□

Corollary 1.3. *Let $A \in M(n, \mathbb{R})$ such that A is poitive definite. When computing the LDL^T factorization of A, at each step we have $D_{kk} > 0$, hence we need no pivoting technique.*

Proof. From the first point of Lemma 1.2 we have that, since $A \succ 0$, $L_1 A L_1{}^T$ is positive definite. Thanks to the second point of the same lemma we have $L_1 A L_1{}^T = \begin{pmatrix} * & 0 & 0 & 0 & 0 \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \end{pmatrix}$ and so the first and the second diagonal blocks are positive definite ($D_{11} > 0$ and $D_{22} \succ 0$).

Notice that, since A is positive definite $A_{11} \succ 0$, but it's a scalar, hence $A_{11} > 0$ and this implies no breakdown case. □

We may introduce another kind of factorization.

1.2 Cholesky factorization

The key idea is to write the diagonal matrix of the Gaussian elimination D as product of $D^{1/2}$ times itself:

$$D = \begin{pmatrix} d_{11} & & & \\ & d_{22} & & \\ & & \ddots & \\ & & & d_{mm} \end{pmatrix} = \begin{pmatrix} \sqrt{d_{11}} & & & \\ & \sqrt{d_{22}} & & \\ & & \ddots & \\ & & & \sqrt{d_{mm}} \end{pmatrix} \cdot \begin{pmatrix} \sqrt{d_{11}} & & & \\ & \sqrt{d_{22}} & & \\ & & \ddots & \\ & & & \sqrt{d_{mm}} \end{pmatrix}$$

The Gaussian elimination may be rewritten as follows

$$A = LDL^T = LD^{1/2}(D^{1/2^T} L^T) = CC^T,$$

where $D^{1/2} = \operatorname{diag}(D_{11}^{1/2}, D_{22}^{1/2}, \dots, D_{mm}^{1/2})$, and C is lower triangular (but not anymore with ones on the diagonal).

In Matlab the Cholesky factorization of a positive definite matrix is performed by the function `chol(A);` and returns C^T.

Observation 1.2. *We will not discuss stability further, but Cholesky is always backward stable even without pivoting ($\|C\| = \|A\|^{1/2}$).*

Observation 1.3. *In a sparse matrix, we can choose the (symmetric) permutation with the only goal of reducing fill-in. The same considerations about LU factorization hold in this case too.*

1.3 Krylov subspace methods

In this part we will discuss how to combine optimization methods to transform a linear system into a minimum problem.

Example 1.1. *Let us assume we are interested in solving a linear system $Ax = b$.*

STEP 1: $x_0 = 0$, $\nabla f(x_0) = -b$;

STEP 2: $x_1 =$*some multiple of* $b \in$ *Span*(b)

$f(x_1) = Ax_1 - b$

$\nabla f(x_1) =$ *some multpiple of* Ab + *some multiple of* $b \in$ *span*(b);

STEP 3: $x_2 =$ *mult. of* x_1 + *mul. of* $\nabla f(x_1)$ + *mult. of* $x_0 + \cdots \in$ *span*(Ab_1, b);

STEP 4: $x_3 = mult.of x_2 + \cdots \nabla f(x_2) +$ *previous iterates* $\cdots$

$Ax_2 - b = A \cdot (sAb + tb) - b = sA^2b + tAb - b \in$ *span*(b, Ab, A^2b), *where* s *and* t *are scalars;*

STEP 5: $x_4 \in$ *span*(b, Ab, A^2b, A^3b).

Notice that we can make some linear combinations of the vectors we have available and $A(A(sb+tAb)+uAb+vb)+e(sb+tAb)+fAb+gb \in span(b, Ab, A^2b, A^3b)$.

Idea: first compute the basis of $span(b, Ab, A^2b, A^3b)$, then look for the best solution inside this subspace.

Observation 1.4. *The cost of multiplying a sparse matrix A with a vector z is $O(nnz(A))$, where $nnz(A)$ is the number of non zero entries of A.*

Proof. Let us assume the matrix A is stored as a vector, whose entries are (i, j, A_{ij}).

The matrix-vector product would then be

1. `x=zeros`
2. `for (i, j,` A_{ij} `) such that` $A_{ij} \neq 0$
3. $x_i = x_i + A_{ij} * z_j$
4. `end`

□

From now on we will consider to have a function `compute_product_with_A` that we use to compute matri-vector products with A. In particular, $x =$`compute_product_with_A(z)` computes $x = Az$, given z.

If we somehow have matrices not really sparse, but allow to compute a clever matrix-vector product this class of algorithms will give good results.

Definition 1.1 (Krylof subspace)**.** *Let $A \in M(m, \mathbb{R})$ and let $b \in \mathbb{R}^m$. The **Krilov subspace** of index n is $\mathbf{K_n}(\mathbf{A}, \mathbf{b}) = span(b, Ab, A^2b, \dots, A^{n-1}b)$.*

Equivalently, $k \in K_n(A, b) \iff \exists \alpha_1, \dots, \alpha_{n-1} \in \mathbb{R}^m$ s.t. $v = \alpha_0 b + \alpha_1 Ab + \alpha_2 A^2 b + \dots + \alpha_{n-1}A^{n-1}b$.

Equivalently, $(\alpha_0 + \alpha_1 A + \alpha_2 A^2 + \dots + \alpha_{n-1}A^{n-1})b = p(A)b$ for a polinomial p of degree such that $deg(p) \leq n-1$.

Observation 1.5 (Properties)**.**

1. *$v, w \in K_n(A, b) \Rightarrow \alpha + \beta W \in K_n(A, b)$;*
2. *$v \in K_n(A, b) \Rightarrow Av \in K_n(A, b)$. Proof. Let us take $v = \alpha_0 b + \dots + \alpha_{n-1} b$, then $Av = A(\alpha_0 b + \dots + \alpha_{n-1}b) = \alpha_0 Ab + \dots + \alpha_{n-1}A^n b$;*
3. $\dim(K_n(A, b)) \leq n$. *It is exactly n if $b, Ab, A^2b, \dots, A^{n-1}b$ are linearly independent.*
4. *Let us assume $dim(K_n(A, b)) \leq n$. In the second point, if A^{n-1} was really necessary $\alpha_{n-1} \neq 0$ or $v \in K_n(A, b)$, $v \notin K_{n-1}(A, b)$, equivalently then $A^n b$ is really necessary to write Av, i.e. $Av \in K_{n+1}(A, b)$ but $Avnot \in K_n(A, b)$.*
5. *We may observe that* $\dim(K_1(A, b)) < \dim(K_2(A, b)) < \dots < \dim(K_{n_{max}}(A, b)) = \dim(K_{n_{max}+1}(A, b)) = \cdots$.