SMM - Section 5 - Linear Systems

Simone Montali - monta.li

22 settembre 2020

1 Linear systems

1.1 Introduction to linear systems

Now, using the matrix decompositions, we are able to solve linear systems. A linear system can be written as:

$$Ax = b$$

where A is a mtrix of size $m \times n$, with $m \geq n$ and x is a column vector of length n and b is a column vecotr of length m. This form can be expanded as a list of equations. We are interested in the existence/uniqueness of a solution, the numerical methods to find it (there can be more methods) and the conditioning of the problem. We are looking for two features: precision of the solution (related to an estimate of error, the numerical solution always has an error, therefore different algorithms produce different errors), and time of solution (an algorithm is efficient if it requires the lesser possible time).
We can now divide the solution in two categories: square and non-square systems.

1.2 Square linear systems

We now have a linear system

$$Ax = b$$

with A of dimension $n \times n$, x and b of dimension n exists and is unique iff one of the following conditions holds:

- A is non-singular
- rank$(A) = n$
- The system $Ax = 0$ only admits the solution $x = 0$

In principle, we can check that the solution is computable by dividing b by A, thus multiplying $A^{-1}b$. The computation of the inverse of A is, sadly, computationally hard. So other solutions are needed. One of these is the *Cramer's rule*, where each component x_i of the solution is the ratio between $\frac{detA_i}{detA}$. This method is very costly too, therefore it is not widely used. The numerical methods follow two different approaches: the **direct methods** (which are more precise but computationally costly) and the **iterative methods** (which require a thoretically infinite number of steps, so they are less precise but faster).

1.2.1 Direct methods

These methods are based on the factorization of the matrix, like the LU factorization (computational cost $O\left(\frac{n^3}{3}\right)$). The solution is therefore computable by solving two triangular systems: $Ly = b$ and $Ux = y$. Why is this kind of complicated method used? Because the solution of a triangular system is much less expensive, and it is solvable by forward or backward substitution (which have computational cost of $O\left(\frac{n^2}{2}\right)$). In the case of the pivoting algorithm, we multiply both terms in $Ax = b$ by P, thus obtaining $PAx = Pb$ and then $Ly = Pb, Ux = y$.

1.2.2 Iterative methods

The basic idea, here, is constructing a sequence of vectors that converge to the exact solution.

$$x^* = lim_{k\to\infty} x_k$$

The convergence means that each component converges to a component of the exact solution. We've got to have a starting guess x_0. The sequence of vector is obtained by applying the same function (or set of operators) iteratively. One might ask: *when can I stop the iteration?* This is the most critical part. **Stopping criteria** are the conditions that allow the machine to stop, and they're based on some quantity related to the residual, defined by $r_k = b - AX_k$ at iteration k. We can even base on the absolute criterion $||x_{k+1} - x_k|| \leq \tau$.

1.2.3 Sparse matrices

A sparse matrix is a matrix with a very low percentage of non-null elements, i.e. the majority of items are equal to 0. These are important for data representation, since only the non-null elements are stored, saving a lot of memory.

1.3 Inherent errors in linear systems

Always remember that inherent errors are due to errors in the data representation and do not depend on the algorithm. Even if we start with a really small representation error, we can have large output errors. We distinguish between well posed (where $|\Delta y| \approx |\Delta x|$) and ill posed (where $|\Delta y| >> |\Delta x|$) problems. In the second case, even with small representation errors, there's a tangible risk of great errors. Considering a linear system $Ax = b$, when b slightly change we get $A(x + \Delta x) = b + \Delta b$. If now we wanted to compare Δx with Δb, specifically their relative values $||\frac{\Delta x}{x}||$ and $||\frac{\Delta b}{b}||$.

1.4 Linear least squares

Considering an overdetermined system, finding the solution is impossible (b does not lie on the subspace spanned by the columns of A), therefore we're looking for approximate solutions, specifically the best one. We then need some tools.

1.4.1 Orthogonality and projections

Since, when working in machine learning, visualizing data in a useful way can be pretty difficult, projections come in help. These compress the data while minimizing the loss of useful informations, therefore making data more *densely informative.*