Computational Mathematics for Learning and Data Analysis

2019 / 2020

Poggiali Alessandro, Berti Stefano

1 Setting the stage

1.1 The problem: Least Square

Our problem is to find an array x such that

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|$$

holds, where

- A is a *tall thin matrix* (so it is a matrix $A \in M(m, n, R)$ where $m \gg n$)
- b is a vector of real number
- $\|.\|$ is the *2-norm* or *Euclidean Norm*: $\|x\| := \sqrt{\sum_{i=1}^{n} x_i}$

and so to find the closest vector to b inside the hyperplane $Im(A)$.

1.2 First algorithm: Conjugate Gradient Method

1.3 Second algorithm: QR factorization with Householder Reflectors

To solve the Least Square Problem, we use the *thin QR factorization* with all the optimization seen (fast Householder-vector product, cancellation problem resolution, manually changing known entries) in order to factorize A as Q_1R_1. We use a variant of the *thin QR factorization* where we do not form the matrix Q, but we keep the Householder vector u_k to perform implicit product with Q and Q^T. With this factorization, we can write $\|Ax - b\|$ as $\left\|\begin{bmatrix} R_1x - Q_1^Tb \\ Q_2^Tb \end{bmatrix}\right\|$ and now we can chose x such that $R_1x - Q_1^Tb = 0$, which is $x = R_1^{-1}Q_1^Tb$ (if R_1 is invertible). Finally we should have $x = \text{argmin}_{x\in\mathbb{R}^n} \|Ax - b\| = R_1^{-1}Q_1^Tb$ and $\|Ax - b\| = \|Q_2^Tb\|$.

2 What to expect from the algorithms

2.1 Conjugate Gradient

2.2 QR with HouseHolder

The QR factorization has a cancellation problem during the Householder reflector construction that is easily fixed by summing first entry and norm instead of subtract it.

Apart from that, since every step is *backward stable*, the factorization algorithm is *backward stable*: the computed Q, R are the exact result of $qr(A+\Delta A)$ where $\|\Delta A\| \leq O(u)\,\|A\|$, so we only have intrinsic representation errors.

The computational cost for thin QR factorization is $2mn^2 - \frac{2}{3}n^3 + O(mn)$ flops, which represents two cases: if $m \approx n$, we have that the cost is $\frac{4}{3}n^3$, if $m \gg n$ the cost scales like $2mn^2$, so it scales linearly with respect to the biggest dimension of A .

Before we said that $x = R_1^{-1} Q_1^T b$, but to be able to calculate it we need R_1 to be invertible.

We know that A has full column rank $\iff$ $Az \neq 0 \forall z \neq 0$ $\iff$ $z^T A^T A z = \|Az\|^2 \forall z \neq 0$ $\iff$ $A^T A$ is positive definite $\iff$ all eigenvalues of $A^T A$ are > 0 $\iff$ 0 is not an eigenvalue of $A^T A$ $\iff$ $A^T A = (Q_1 R_1)^T Q_1 R_1 = R_1^T Q_1^T Q_1 R_1 = R_1^T R_1$ is invertible $\iff$ $det(A^T A) = det(R_1^T) det(R_1) \neq 0$ $\iff$ $det(R_1) \neq 0$ $\iff$ R_1 is invertible.

So if A has full column rank, $\min_{x \in \mathbb{R}^n} \|Ax - b\|$ has solution and it is unique.

R_1 is invertible also if all elements on its diagonal are $\neq 0$.

The cost to solve $x = R_1^{-1}(Q_1^T)b$ is a multiplication $c = (Q_1^T)b$, which costs $O(mn)$, and the resolution of the triangular system $R_1 x = c$ with back-substitution, which costs $O(n^2)$, but the overall cost $O(mn)+O(n^2)$ is negligible with respect to the cost $O(mn^2)$ to compute Q_1, R_1.

3 Input data

Our input data is the matrix A, which is the tall thin matrix used as input in the *ML-cup 2019-2020* competition. Its shape is $(1765, 20)$, we augmented it with few functions of the features of the dataset:

- **24th column**: logarithm of the absolute value of the 1st column
- **25th column**: product of 2nd, 3rd and 4th columns
- **26th column**: 5th column to square

So the final shape of A is $(1765, 23)$.