Least Squares Problem Sensitivity Analysis

Computational Linear Algebra for Large Scale Problems Homework

Giulio Zabotto
s279487@studenti.polito.it

February 16, 2022

Abstract

We present the least squares problem and its solution via QR factorization and the conditioning of the problem. We explore the accuracy of the modified Gram-Schmidt and the Householder algorithms in a simple simulation.

1 Solving the least squares problem

1.1 Problem statement

A linear system with m equations and n unknowns is described by $Ax = b$ where $A \in \mathbb{C}^{m \times n}$ is the coefficient matrix with m rows and n columns, x is the vector of unknowns and $b \in \mathbb{C}^m$ is the vector of constant terms.

In most cases, this linear system has no solution. By the Rouché-Capelli theorem, a linear system has a unique solution if and only if the rank of the coefficient matrix is equal to the rank of the augmented matrix $(A|b)$. From a geometric perspective, the theorem states that unless b is spanned onto the range of A, then no solution exists. Since A may have at most rank n, the cases in which $n = rank(A|b)$ are few, given that $m > n$. This kind of linear system is called *overdetermined*.

One may solve the problem asking a different question. Let us define the residual r as the difference between the point b we are aiming at and our solution Ax :

$$r = b - Ax.$$

A reasonable solution to the problem would require to have the residual as small as possible. In symbols, we want to find

$$\min_x \|b - Ax\|_2 .$$

The 2-norm is the natural choice thinking at the problem in a euclidean space. Moreover, this norm naturally arises when we give a statistical interpretation of the problem. Indeed, it can be shown that the minimization of the residual

maximizes the variance explained by a linear regression performed on the data collected in our coefficient matrix A.

1.2 QR factorization

The least squares problem is solved either by Cholesky decomposition, or singular value decomposition, or QR factorization. We are going to emply the latter in this project.

The objective of a factorization is to transform a matrix A into a product of matrices with a set of properties that are useful for a given purpose. The aim of a QR factorization is to find a set of orthonormal vectors that span the successive ranges of a given matrix. With *successive ranges* of a matrix A, we refer to the following vector spaces

$$span(a_1) \supseteq span(a_1, a_2) \supseteq span(a_1, a_2, a_3) \supseteq \cdots$$

where a_i is the i-th column vector of the matrix A.

Thus, the factorization $A = QR$ requires Q to be the matrix containing the orthonormal vectors spanning the successive ranges of A, and R an upper triangular matrix hosting the non-zero coefficients of those linear combinations.

$$\begin{aligned} a_1 &= r_{11} q_1 \\ a_2 &= r_{12} q_1 + r_{22} q_2 \\ &\vdots \\ a_k &= r_{1k} q_1 + \cdots + r_{kk} q_k \end{aligned} \tag{1}$$

1.3 Solving the problem

We are looking for a vector x such that the residual 2-norm $\|b - Ax\|_2$ is minimized. Geometrically, this amounts to finding the vector r that minimizes the distance between the point b and the range(A). It is easy to see that r should be orthogonal to the range of A.

The residual r can also be interpreted as the difference between the point b and its projection onto range(A), that is, $r = b - Pb$, where P is the projection matrix. With the same reasoning as above, the vector r should be perpendicular to range(A), that entails that P must be an orthogonal projector.

Next, we need to compute P. It can be shown that the product of unitary matrix Q times its conjugate Q^* yields an orthogonal projector onto range(Q).

Now we have all the tools to solve the least squares problem. Finding x such that the 2-norm of the residual $\|b - Ax\|_2^2$ is minimized is equal to asking for a projector P orthogonal to range(A):

$$Ax = Pb. \tag{2}$$

Given a QR factorization of A, $A = QR$, we know that an orthogonal projector onto range(A) is given by

$$P = QQ^* \tag{3}$$

Plugging 3 in 2 with A factorized yields

$$Rx = Q^*b \tag{4}$$

The solution of this linear system by x is trivial as R is a triangular matrix, and it completes our objective.

We will not discuss the existence or the uniqueness of a QR factorization of a matrix. We leave that discussion to the main reference of this report *Numerical Linear Algebra* by Trefethen and Bau [TB97].

2 Conditioning numbers for the least squares problem

Sensitivity analysis is the study of how much a result changes given a perturbation on the input of the problem.

In linear algebra, this is the study of the *conditioning* of a problem. Given a problem $f : X \to Y$, where X and Y are the input and output vector space, respectively. Hence, a problem is either *ill-conditioned* if, given a perturbation δx, the yielded $\delta f = f(x + \delta x) - f(x)$ is large, *well-conditioned* otherwise.

It is convenient to have a measure of the sensitivity, for this pursuit, we define the *relative conditioning number* as

$$\kappa = \lim_{\delta \to 0} \sup_{\|\delta x \leq \delta\|} \left(\frac{\|\delta f\|}{\|f(x)\|} \Big/ \frac{\|\delta x\|}{\|x\|} \right).$$

In the least squares problem, the input of the problem are the coefficient matrix A and the vector of constants b. We are going to show only the results regarding perturbation on A, as they are the only relevant to our purposes.

A perturbation on A is a tilting of the range of A, hence, it affects both the projection of b onto the range of A, and we shall call it $y = Pb$, and the solution x.

Given a perturbation in A, the relative conditioning number of y and x are given by [TB97]

$$\kappa_{A \mapsto y} = \frac{\kappa(A)}{\cos\theta} \tag{5}$$

$$\kappa_{A \mapsto x} = \kappa(A) + \frac{\kappa(A)^2 \tan\theta}{\eta} \tag{6}$$

where $\kappa(A)$ is condition number of the rectangular matrix A, given by $\kappa(A) = \|A\| \, \|A^+\|$; θ is the angle formed by the intersection between range(A) and the

vector b; η describes how much $\|y\|$ falls short of its maximum possible value; in symbols,

$$\eta = \frac{\|A\| \, \|x\|}{\|y\|}.$$

So, the first result (5) shows that the projected solution y is inversely proportional to the degree of perpendicularity between the range of A and the vector b; the more perpendicular, the more the condition number grows. This relation is especially useful in the linear regression. In such setting, y is the regression model. Thus, a high condition number $k_{A \mapsto y}$ implies that the accuracy would be compromised. Conversely, equation (6) shows how much the computed solution x may lower its accuracy due to A. Showing the reasons behind the formula is beyond the scope of this report.

3 Algorithms to solve the least squares problem

3.1 Modified Gram-Schmidt orthogonalization

This algorithm computes the orthonormal vectors q_k and their normalization coefficients of the upper triangular matrix R directly from (1). Hence,

$$r_{kk} = \left\| a_k - \sum_{i=1}^{k-1} r_{ik} q_i \right\|_2$$

$$q_k = \frac{a_k - \sum_{i=1}^{k-1} r_{ik} q_i}{r_{kk}}$$

The straightforward implementation of this idea turns out to be numerically unstable: the reflected vectors are not perfectly perpendicular in finite arithmetic. To mitigate the propagation of errors, we reflect the vectors against its computed q_k instead of the original vectors a_k. The *modified* Gram-Schmidt method implements this features.

The following code implements the algorithm as MATLAB function

```
function [Q, R] = mgs(A)

[~, N] = size(A);
Q = A;
R = zeros(N, N);

for i = 1:N
    R(i, i) = norm(Q(:, i));
    Q(:, i) = Q(:, i) / R(i, i);
    for j = i+1:N
        R(i, j) = Q(:,i)' * Q(:, j);
        Q(:, j) = Q(:,j) - R(i, j)*Q(:,i);
    end
end
```

Let us note that the yielded matrix Q is rectangular, thus *reduced*, as it shares the same dimension of the input matrix A.

3.2 Householder triangularization

The main idea leading this method is to find that one orthogonal reflector F mapping the input vector $a_i \in \text{col}(A)$ into a new vector having k non-zero entries, while zeroing the remaining. The successive application of these reflectors F to the given A matrix results in a triangularized matrix R. It can be shown that the projection matrix F is

$$F = I - 2\frac{vv^*}{v^*v}$$

where v is the reflection vector

$$v = sign(x_1) \left\|x\right\|_2 e_1 + x \tag{7}$$

and x_i if the first component of x, the subset of components of the vector $a_i \in A$.

The following code implements the algorithm as MATLAB function.

```
function [W, R] = householder(A)

[M, N] = size(A);
R = zeros(N, N);
W = zeros(M, N);

for k = 1 : N
    x = A(k:end, k);
    x(1) = x(1) + sign(x(1)) * norm(x);
    x = x / norm(x);
    W(k:end, k) = x;
    A(k:end, k:end) = A(k:end, k:end) - ...
        2* x * (x' * A(k:end, k:end));
    R(1:k, k) = A(1:k, k);
end
```

The W matrix collects the reflection vectors defined in 7, whilst R is the full upper triangular matrix.

3.3 Algorithm comparison

3.3.1 Experiment I

The Hilbert matrix is known to have a high condition number, hence, it is the perfect setting to estimate the accuracy the algorithms seen so far. Let us define a Hilbert matrix $H \in \mathbb{R}^{100\times6}$, with entries $h_{ij} = \frac{1}{i+j-1}$. The condition number of H is $\kappa(H) = 3.2088e^5$. A high condition number may help us highlight the differences between the algorithms.

In order to compute the accuracy of an algorithm, we need and exact solution to the problem. Hence, we arbitrary set the exact solution to

$$x = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \end{bmatrix}.$$

Next, we compute the vector b as the product of exact solution and the Hilbert matrix, $b = Ax$.

Finally, we compute the new solution $\tilde{x}$ with the QR factorization.

We first experiment with the Householder reflection method. The *MATLAB* code is the following.

```
[W, R] = householder(H);
y = Qb_product(H, b, W);
x = R \ y(1:N);
```

The function `householder(H)` factorizes the Hilbert matrix H according to the Householder method and yields the matrix W, that collects the reflection vectors defined in 7, while R, is the standard upper triangular matrix. The function `Qb_product(H, b, W)` computes the reflection of the vector b onto range(A), that is y, exploiting only the reflection vectors in W. This function avoids the computation of the matrix Q, since it is not needed. The y vector needs to be trimmed to its first N elements as we are working with reduced version of Q, the one that cuts the unnecessary $M - N$ vectors.

```
N = size(A, 2);
for k = 1 : N
    b(k:end) = b(k:end) - 2 * W(k:end, k) ...
        * (W(k:end, k)' * b(k:end));
end
```

The accuracy yielded by the algorithms is

$$\frac{\|\tilde{x} - x\|}{\|x\|} = 9.295251e^{-13}$$

Is it accurate?

To properly answer the question we have to consider stability of our algorithm and the condition number of our problem.

An algorithm is stable if it does not amplify the inevitable errors due to the finite arithmetic.

It can be proven that the Householder algorithm is not only stable, but even *backward* stable, hence, it should not amplify the errors. This does not guarantee that our implementation of the algorithm is ill-conditioned. Before looking at our code searching for ill-conditioned steps, we have to consider the condition number of our problem. For backward stable algorithms, the following results holds [TB97]. Given a problem $f : X \to Y$, with condition number κ, then the relative error satisfy

$$\frac{\left\|\tilde{f}(x) - f(x)\right\|}{\|f(x)\|} = O(\kappa(x)\epsilon_{machine}) \tag{8}$$

In our setting, $\tilde{f}(x)$ corresponds to the computed solution $\tilde{x}$, $f(x)$ to the exact solution x, while $\kappa(x)$ to $\kappa_{A\mapsto x}$ as we showed in (6). Since the computation is performed in IEEE double precision, $\epsilon_{machine} = 2^{-53} \approx 1.1102e^{-16}$.

In our setting, $\kappa_{A\mapsto x} = 3.2191e^{5}$, hence,

$$\frac{\|\tilde{x} - x\|}{\|x\|} = O(3.5739e^{-11})$$

Our solution $\tilde{x}$ is within the upper-bound $3.5739e^{-11}$ by two orders of magnitude.

3.3.2 Experiment II

In the same setting we perform the same task with the modified Gram-Schmidt method.

The code for the calculation is the following.

```
[Q, R] = mgs(H);
x = R \ (Q' * b);
```

The yielded accuracy is

$$\frac{\|\tilde{x} - x\|}{\|x\|} = 6.899554e^{-8}$$

The result is poor. The accuracy exceeds the upperbound defined in (8). We can assert that the algorithm is unstable. We can try to sort out the source of this instability. The first suspect is the lack of orthonormality in the the columns vectors in Q.

We can test the orthornomality of a matrix. If Q were orthonormal, then $Q^*Q = I$. Exploiting this property, we can estimate the orthonormality as

$$\|Q^*Q - I\|.$$

The more it diverges from zero, the far the matrix Q is from orthonormality. The calculation of this quantity amounts to $5.352359e^{-12}$. As we can see, we are not even close to $\epsilon_{machine} \approx 10^{-16}$. Hence, the large error in the accuracy of the solution can be found here.

References

[TB97] Lloyd N. Trefethen and David Bau. *Numerical Linear Algebra*. SIAM, 1997.