Analysis of optimization and numerical approaches to solve the linear least square problem

***Abstract*—The linear least square problem can be tackled using a wide range of optimization or numerical methods. The L-BFGS method of the class of limited-memory quasi-Newton algorithms has been chosen for the former, whilst the thin QR factorization with Householder reflectors for the latter. Both these algorithms have been implemented from scratch using Python language, to finally experiment over their performances in terms of precision, stability and speed. The accordance of the implementations with the underlying theoretical models is also studied and discussed.**

Introduction

Given a dataset composed by a matrix $\hat{X} \in \mathbb{R}^{m \times n}$ with $m \geq n$ and a vector $y \in \mathbb{R}^m$, the solution of the linear least square (LLS) problem is the vector $w \in \mathbb{R}^n$ that fits best the data assuming a linear function between $\hat{X}$ and y. (Nocedal and Wright 2006, 50) This can be formalized as the following minimization problem:

$$w_* = \min_w \|\hat{X}w - y\|_2^2$$

The matrix $\hat{X}$ is actually composed in the following way:

$$\hat{X} = \begin{bmatrix} X^T \\ I \end{bmatrix}$$

Where $X \in \mathbb{R}^{n \times k}$ is a tall thin matrix, thus $m = k + n$. The LLS problem can be dealt both with iterative methods or with direct numerical methods. One algorithm has been chosen for each of these fields to finally discuss their experimental results.

L-BFGS

The L-BFGS is an iterative method of the quasi-Newton limited-memory class. This method is actually a variation of the BFGS method, with which it shares the update rule; at the $i+1$-th iteration the point is updated as follows:

$$w_{i+1} = w_i - \alpha_i H_i \nabla f_i$$

L-BFGS has an inferior space complexity due to the fact that the Hessian approximation H_i is stored implicitly, and built over a fixed number of vector pairs $\{s_j, y_j\}$ of the previous t iterations and an initial matrix H_i^0. Where

$$s_i = w_{i+1} - w_i, \quad y_i = \nabla f_{i+1} - \nabla f_i$$

$$V_i = I - \rho_i y_i s_i^T, \quad \rho_i = \frac{1}{y_k^T s_k}$$

so H_i satisfies the following:

$$\begin{aligned} H_i = {} & (V_{i-1}^T \dots V_{i-t}^T) H_i^0 (V_{i-t} \dots V_{i-1}) \\ & + \rho_{i-t}(V_{i-1}^T \dots V_{i-t}^T + 1) s_{i-t} s_{i-m}^T (V_{i-t+1} \dots V_{i-1}) \\ & + \rho_{i-t+1}(V_{i-1}^T \dots V_{i-t}^T + 2) s_{i-t+1} s_{i-t+1}^T (V_{i-t+2} \dots V_{i-1}) \\ & + \dots \\ & + \rho_{i-1} s_{i-1} s_{i-1}^T \end{aligned}$$

Different strategies to initialize the H_i^0 matrix are proposed in the literature, and so they will be tested experimentally. Finally, the step size α_i is found by performing an inexact line search based on the Armijo-Wolfe conditions.

Thin QR factorization

For the numerical counterpart, the thin QR factorization with Householder reflectors has been implemented as described in (Trefethen and Bau 1997).

By using the Householder QR factorization, the matrix R is constructed in place of $\hat{X}$ and the n reflection vectors $v_1, \dots, v_n$ are stored. The reduced matrix $\hat{R}$ is trivially obtainable by slicing as in $\hat{R} = R_{1:n,1:n}$. In fact, given that $\hat{X}$ is already stored in memory and fully needed, there would be no advantage in directly constructing the reduced matrix.

By using the Householder vectors it is also possible to implicitly compute $\hat{Q}^T y$ to finally obtain w_* by back substitution over the upper-triangular system $\hat{R}w = \hat{Q}^T y$.

Algorithmic analysis

Convergence of L-BFGS

Liu and Nocedal (1989) define three necessary assumptions to prove a theorem stating that the L-BFGS algorithm globally converges and moreover that there exists a constant $0 \leq r < 1$ such that

$$f(w_i) - f(w_*) \leq r^i (f(w_0) - f(w_*)) = \sigma_i$$

so that the sequence w_i converges R-linearly.

The first assumption required is on the objective function f, that should be twice continuously differentiable. This is in fact true and we can define the gradient and the Hessian of the objective function as in:

$$\nabla f(w) = \hat{X}^T(\hat{X}w - y)$$

$$\nabla^2 f(w) = \hat{X}^T\hat{X}$$

Moreover the Hessian is positive definite since it can be rearranged in the following way:

$$\begin{aligned}\nabla^2 f(w) &= \hat{X}^T\hat{X} \\ &= \left|XI\right| \begin{vmatrix} X^T \\ I \end{vmatrix} \\ &= XX^T + I\end{aligned}$$

Being the Hessian positive definite, the objective function f is a convex function. This comes in handy for the second assumption requiring the sublevel set $D = \{w \in \mathbb{R}^n | f(w) \leq f(w_0)\}$ must be convex, it can be easily proved that if a function is convex all of its sublevel sets are convex sets.

$$\begin{aligned}&\forall x, y \in D,\ \lambda \in [0,1] \\ &\qquad \text{f convex} \\ &\qquad \implies f(\lambda x + (1-\lambda)y) \\ &\qquad \leq \lambda f(x) + (1-\lambda)f(y) \\ &\qquad \leq \lambda f(w_0) + (1-\lambda)f(w_0) \\ &\qquad = f(w_0) \\ &\qquad \implies \lambda x + (1-\lambda y) \in D\end{aligned}$$

The third and last assumption requires the existence of two positive constants M_1 and M_2 such that $\forall z \in \mathbb{R}^n, w \in D$:

$$M_1\|z\|^2 \leq z^T\nabla^2 f(w) \leq M_2\|z\|^2$$

or equivalently

$$M_1 I \preceq \nabla^2 f(w) \preceq M_2 I$$

The first part of the equation is surely satisfied by $M_1 = 1$, keeping in mind the previous decomposition $\nabla^2 f(w) = XX^T + I$. Considering also that all the eigenvalues in a positive definite matrix are real and positive, it is possible to use the largest eigenvalue as in $M_2 = \lambda_{max}$.

Other then these assumptions, the theorem requires for the sequence of Hessian substitutes H_i to be bounded. This obviously depends on the initialization technique used to generate H_i^0, various techniques are suggested in the literature and so they will be empirically tested.

Finally the convergence requires to perform a line search respectful of the Armijo-Wolfe conditions. The algorithm described in Al-Baali and Fletcher (1986) to perform an inexact line search is ensured to converge under the assumption that $\sigma > \rho$ where $\rho \in (0, \frac{1}{2}), \sigma \in (0,1)$, respectively the constant for the Armijo condition and for the Wolfe one.

Analysis of standard and modified thin QR

Ignoring constants, we know from theory that the standard QR factorization algorithm applied on the matrix $\hat{X}$ yields a time complexity of $O(mn^2)$. Actually, given that we are generally dealing with a very tall and thin matrix X, the resulting $\hat{X}$ is going to be squarish ($m \approx n$). This means that we can consider the complexity to be cubic in n.

From now on, we show a way to bring down the time complexity of the algorithm from $O(mn^2)$ to $O(kn^2)$, with $k = m - n$. The resulting modified QR factorization algorithm will become useful when k is much smaller than m, as in our case.
In the standard algorithm, at each step we focus on a single column of the input matrix, constructing a householder vector to zero out all the entries below the diagonal. Following the geometric reasoning in (Trefethen and Bau 1997), this brings the algorithm to depend on m. While this cannot be avoided in general, in our particular case we can be a little bit smarter.

Since the block matrix $\hat{X}$ contains the identity as its lower block, at each step of the algorithm we can just focus on zeroing out the $k = m - n$ entries below the diagonal up to the 1s of the identity. In the modified algorithm then, to obtain R, the matrix $\hat{X}$ is multiplied on the left side by a sequence of matrices $L_i \in \mathbb{R}^{m \times m}$ of the form ($i = 1, \ldots, n$):

$$L_i = \begin{bmatrix} I_{i-1} & 0 & 0 \\ 0 & H_i & 0 \\ 0 & 0 & I_{n-i} \end{bmatrix}$$

where $H_i \in \mathbb{R}^{(k+1)\times(k+1)}$ are all householder reflectors that zero out the k entries in the i-th column of the matrix which is being multiplied by L_i.
To derive the time complexity of this phase we can reason as follows. The right side matrix can be divided in three blocks as in $\left[\begin{smallmatrix} A \\ B \\ C \end{smallmatrix}\right]$. When we multiply this matrix by L_i the only relevant operation is the matrix multiplication H_iB, which costs $O(kn)$. Since the total number of multiplications is n, the total complexity is $O(kn^2)$.
Since each L_i is orthogonal and symmetric it is then possible to reconstruct Q and the reduced $\hat{Q}$ in the following way:

$$\begin{aligned}Q &= L_1 L_2 \ldots L_n \\ \hat{Q} &= L_1 L_2 \ldots L_n \begin{bmatrix} I_n \\ 0 \end{bmatrix}\end{aligned}$$

Applying again the reasoning above, the time complexity of these reconstructions is $O(kn^2)$. It follows that the overall time complexity of the modified QR factorization is $O(kn^2)$.

The least squares problem is then solved through back substitution over the upper-triangular system $\hat{R}w = \hat{Q}^T b$,

which costs $O(n^2)$. Since this is dominated by the factorization cost, the total time complexity for solving the least squares problem through QR factorization is $O(mn^2)$ when using the standard algorithm and $O(kn^2)$ when using the modified one.

Stability and accuracy of the QR algorithm

As stated in (Trefethen and Bau 1997, 140), the algorithm obtained by combining the standard QR algorithm, the $Q^T b$ product and back substitution is backward stable in the context of least squares problems.
We claim that the QR factorization step remains backward stable if we consider the modified version described in this report. Without going into details with an extended proof (?), this can be explained by saying that at each step of the algorithm we apply a transformation L_i doing a smaller number of operations than those of the standard algorithm. Then, since we know that each step of the standard QR factorization is backward stable, this must be true also in the modified version of the algorithm.

Since both versions of the QR algorithm are backward stable, the accuracy of the algorithms will depend mostly on the conditioning of the least squares problem at hand. In fact, following from the definition of backward stability, the algorithms will both find exact solutions to slightly perturbed problems, with perturbations of the order of machine precision. This implies that if the conditioning of the problem is high the real solutions to the perturbed problems are inevitably going to be inaccurate.
If w_* is the exact solution to the least squares problem and $\tilde{w}_*$ is the solution found with one of the QR based algorithms outlined above, the accuracy of the algorithms will therefore follow the general upper bounds of relative errors found in (Trefethen and Bau 1997, 131):

$$\frac{\|\tilde{w}_* - w_*\|}{\|w_*\|} \leq (\kappa(\hat{X}) + \kappa(\hat{X})^2 \tan\theta) \frac{\|\delta\hat{X}\|}{\|\hat{X}\|}$$

$$\frac{\|\tilde{w}_* - w_*\|}{\|w_*\|} \leq \left(\frac{\kappa(\hat{X})}{\cos\theta}\right) \frac{\|\delta y\|}{\|y\|}$$

where θ is the angle such that $\cos\theta = \frac{\|\hat{X}w_*\|}{\|y\|}$.
From these upper bounds we can expect that the algorithm will be more accurate when the angle theta is near 0 and less accurate when it is near $\frac{\pi}{2}$, reminding that in our context the value of θ will depend on the value of the random vector y.

Input data

Bibliography

Al-Baali, M., and R. Fletcher. 1986. "An Efficient Line Search for Nonlinear Least Squares." *Journal of Optimization Theory and Applications* 48 (3): 359–77. https://doi.org/10.1007/BF00940566.

Liu, Dong C., and Jorge Nocedal. 1989. "On the Limited Memory BFGS Method for Large Scale Optimization." *Mathematical Programming* 45 (1-3): 503–28. https://doi.org/10.1007/BF01589116.

Nocedal, Jorge, and Stephen J. Wright. 2006. *Numerical Optimization*. 2nd ed. Springer Series in Operations Research. New York: Springer.

Trefethen, Lloyd N., and David Bau. 1997. *Numerical Linear Algebra*. Philadelphia: Society for Industrial; Applied Mathematics.