Numerical Linear Algebra Primer

Ryan Tibshirani
Convex Optimization 10-725/36-725

Last time: proximal gradient descent

Consider the problem

$$\min_x \; g(x) + h(x)$$

with g, h convex, g differentiable, and h "simple" in so much as

$$\text{prox}_t(x) = \underset{z}{\text{argmin}} \; \frac{1}{2t}\|x - z\|_2^2 + h(x)$$

is computable. Proximal gradient descent: let $x^{(0)} \in \mathbb{R}^n$, repeat

$$x^{(k)} = \text{prox}_{t_k}\big(x^{(k-1)} - t_k \nabla g(x^{(k-1)})\big), \quad k = 1, 2, 3, \ldots$$

Step sizes t_k chosen to be fixed and small, or via backtracking

If ∇g is Lipschitz with constant L, then this has convergence rate $O(1/\epsilon)$. Lastly we can accelerate this, to optimal rate $O(1/\sqrt{\epsilon})$

Outline

Today:

- Complexity of basic operations
- Solving linear systems
- Matrix factorizations
- Sensitivity analysis
- Alternative indirect methods

Complexity (flop counts) of basic operations

Flop (floating point operation):

- One addition, subtraction, multiplication, or division of two floating point numbers
- Serves as a basic unit of computation
- We are interested in rough, not exact flop counts

Vector-vector operations: given $a, b \in \mathbb{R}^n$:

- Addition, $a + b$, costs n flops
- Scalar multiplication, $c \cdot a$, costs n flops
- Inner product, $a^T b$, costs $2n$ flops

Flops do not tell the whole story: setting every element of a to 1 costs 0 flops

Matrix-vector product: given $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^n$, consider Ab:

- In general, costs $2mn$ flops
- For s-sparse A, costs $2s$ flops
- For k-banded $A \in \mathbb{R}^{n \times n}$, costs $2nk$ flops
- For $A = \sum_{i=1}^{r} u_i v_i^T \in \mathbb{R}^{m \times n}$, costs $2r(m+n)$ flops
- For $A \in \mathbb{R}^{n \times n}$ a permutation matrix, costs 0 flops

Matrix-matrix product: for $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{n \times p}$, consider AB:

- In general, costs $2mnp$ flops
- For s-sparse A, costs $2sp$ flops (less if B is also sparse)

Matrix-matrix-vector product: for $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{n \times p}$, $c \in \mathbb{R}^p$, consider ABc:

- Costs $2np + 2mn$ flops if done properly
- Costs $2mnp + 2mp$ flops if done improperly!

Solving linear systems

For nonsingular $A \in \mathbb{R}^{n \times n}$, consider solving linear system $Ax = b$, i.e., computing $x = A^{-1}b$:

- In general, costs about n^3 flops—we'll see more on this later
- For diagonal A, costs n flops:

$$x = (b_1/a_1, \ldots b_n/a_n)$$

- For lower triangular A (i.e., $A_{ij} = 0$ for $j > i$), costs n^2 flops:

$$\begin{aligned} x_1 &= b_1/A_{11} \\ x_2 &= (b_2 - A_{21}x_1)/A_{22} \\ &\vdots \\ x_n &= (b_n - A_{n,n-1}x_{n-1} \ldots - A_{n1}x_1)/A_{nn} \end{aligned}$$

  This is called forward substitution

- For upper triangular A, costs n^2, by backward substitution

- For s-sparse A, often costs $\ll n^3$ flops, but exact (worse-case) flop counts are not known for abitrary sparsity structures
- For k-banded A, costs nk^2 flops—more later
- For orthogonal A, we have $A^{-1} = A^T$, and so $x = A^T b$ costs $2n^2$ flops
- For permutation A, again $A^{-1} = A^T$, and so $x = A^T b$ costs 0 flops. Example:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad A^{-1} = A^T = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matrix factorizations

As you've probably learned, we can solve $Ax = b$ by, e.g., Gaussian elimination. But instead it can be useful to factorize A:

$$A = A_1 A_2 \dots A_k$$

and then compute $x = A_k^{-1} \dots A_2^{-1} A_1^{-1} b$. Usually $k = 2$ or 3, and

- Computing the factorization is expensive, about n^3 flops
- Applying $A_1^{-1}, \dots A_k^{-1}$ is cheaper, about n^2 flops
- This is because $A_1, \dots A_k$ are structured: either orthogonal, triangular, diagonal, or permutation matrices

This is useful when we want to solve $Ax = b$, $Ax = b'$, ... many linear systems in A. Also, if A undergoes a simple change, then an old factorization for A can often be efficiently updated

Cholesky decomposition

If $A \in \mathbb{S}^n_{++}$ (symmetric, positive definite), then it has a Cholesky decomposition:

$$A = LL^T$$

with $L \in \mathbb{R}^{n \times n}$ lower triangular. Can compute this in $n^3/3$ flops

Once we have Cholesky factors, we solve $Ax = b$:

$$y = L^{-1}b \quad \text{by forward substitution, } n^2 \text{ flops}$$
$$x = (L^T)^{-1}y \quad \text{by back substitution, } n^2 \text{ flops}$$

So solving costs $2n^2$ flops

Factorization and solve steps together cost $n^3/3 + 2n^2$ flops

Least squares problems and Cholesky

Now given $y \in \mathbb{R}^n$, $X \in \mathbb{R}^{n \times p}$, consider the least squares problem:

$$\min_{\beta \in \mathbb{R}^p} \|y - X\beta\|_2^2$$

Assuming X has full column rank, solution is $\hat{\beta} = (X^T X)^{-1} X^T y$. How expensive?

- Compute $X^T y$, in $2pn$ flops
- Compute $X^T X$, in $p^2 n$ flops
- Compute Cholesky of $X^T X$, in $p^3/3$ flops
- Solve $(X^T X)\beta = X^T y$, in $2p^2$ flops

Thus in total, about $(n + p/3)p^2$ flops (or np^2 flops if $n \gg p$)

QR decomposition

If $A \in \mathbb{R}^{m \times n}$ with $m \geq n$, then it has a QR decomposition:

$$A = QR$$

with $Q \in \mathbb{R}^{m \times n}$ orthogonal (i.e., $Q^T Q = I$), and $R \in \mathbb{R}^{n \times n}$ upper triangular. Can compute this in $2(m - n/3)n^2$ flops

- If A has rank n, then all diagonal elements of R are nonzero, and the columns of Q form an orthonormal basis for $\mathrm{col}(A)$
- If A has rank r, then the first r diagonal elements of R are nonzero, and the first r columns of Q form a basis for $\mathrm{col}(A)$

Key identity: $\|x\|_2^2 = \|Q^T x\|_2^2 + \|\tilde{Q}^T x\|_2^2$, where $\tilde{Q} \in \mathbb{R}^{m \times (m-n)}$ completes the basis from Q

Least squares problems and QR

Now back to $y \in \mathbb{R}^n$, $X \in \mathbb{R}^{n \times p}$ full column rank, and the least squares problem:

$$\min_{\beta \in \mathbb{R}^p} \|y - X\beta\|_2^2$$

Let $X = QR$ be a QR decomposition. Then

$$\|y - QR\beta\|_2^2 = \|Q^T y - R\beta\|_2^2 + \|\tilde{Q}^T y\|_2^2$$

Second term does not depend on β. So for least squares solution:

- Compute $X = QR$, in $2(n - p/3)p^2$ flops
- Compute $Q^T y$, in $2pn$ flops
- Solve $R\beta = Q^T y$, in p^2 flops

Hence in total, about $2(n - p/3)p^2$ flops (or $2np^2$ flops if $n \gg p$)

Linear systems and sensitivity

Consider first the linear system $Ax = b$, for nonsingular $A \in \mathbb{R}^{n\times n}$. The singular value decomposition of A:

$$A = U\Sigma V^T,$$

where $U, V \in \mathbb{R}^{n\times n}$ are orthogonal, and $\Sigma \in \mathbb{R}^{n\times n}$ is diagonal with elements $\sigma_1 \geq \ldots \geq \sigma_n > 0$

Even if A is full rank, it could be near a singular matrix B, i.e.,

$$\text{dist}(A, \mathcal{R}_k) \;=\; \min_{\text{rank}(B)=k} \|A - B\|_{\text{op}}$$

could be small, for some $k < n$. An easy SVD analysis shows that $\text{dist}(A, \mathcal{R}_k) = \sigma_{k+1}$. If this is small, then solving $x = A^{-1}b$ could pose problems

From the lens of the SVD:

$$x = A^{-1}b = V\Sigma^{-1}U^T b = \sum_{i=1}^{n} \frac{v_i u_i^T b}{\sigma_i}$$

We can see that if some $\sigma_i > 0$ is small (close to set of rank $i-1$ matrices) then we could be in trouble

Precise sensitivity analysis: fix some $F \in \mathbb{R}^{n\times n}$, $f \in \mathbb{R}^n$. Solve

$$(A + \epsilon F)x(\epsilon) = (b + \epsilon f)$$

Theorem: The solution to the perturbed system satisfies

$$\frac{\|x(\epsilon) - x\|_2^2}{\|x\|_2} \leq \kappa(A)(\rho_A + \rho_b) + O(\epsilon^2)$$

where $\kappa(A) = \sigma_1/\sigma_n$ is the condition number of A, and ρ_A, ρ_b are the relative errors $\rho_A = |\epsilon|\|F\|_{\text{op}}/\|A\|_{\text{op}}$, $\rho_b = |\epsilon|\|f\|_2/\|b\|_2$

Proof:

- By implicit differentiation,

$$\frac{dx}{d\epsilon}(0) = A^{-1}(f - Fx)$$

  where we abbreviate $x = x(0)$

- Using a Taylor expansion around 0,

$$x(\epsilon) = x + \epsilon A^{-1}(f - Fx) + O(\epsilon)^2$$

- Rearranging gives

$$\frac{\|x(\epsilon) - x\|_2}{\|x\|_2} \leq |\epsilon| \|A^{-1}\|_{\text{op}} \left(\frac{\|f\|_2}{\|x\|_2} + \|F\|_{\text{op}} \right) + O(\epsilon)^2$$

  Multiplying and dividing by $\|A\|_{\text{op}}$ proves the result, since $\kappa(A) = \|A\|_{\text{op}} \|A^{-1}\|_{\text{op}}$

Cholesky versus QR for least squares

Linear systems: worse conditioning means great sensitivity. What about for least squares problems?

$$\min_{\beta \in \mathbb{R}^p} \|y - X\beta\|_2^2$$

- Recall Cholesky solves $X^T X\beta = X^T y$. Hence we know that sensitivity scales with $\kappa(X^T X) = \kappa(X)^2$
- Meanwhile, QR operates on X, never forms $X^T X$, and can show that sensitivity scales with $\kappa(X) + \rho_{\mathrm{LS}} \cdot \kappa(X)^2$, where $\rho_{\mathrm{LS}} = \|y - X\hat{\beta}\|_2^2$

Summary: Cholesky is cheaper (and uses less memory), but QR is more stable when ρ_{LS} is small and $\kappa(X)$ is large

Some advanced topics

- Updating matrix factorizations: can often be done efficiently after a simple change. E.g., QR of $A \in \mathbb{R}^{m \times n}$ can be updated in $O(m^2)$ flops after adding or deleting a row, and $O(mn)$ flops after adding or deleting a column
- Underdetermined least squares: if $X \in \mathbb{R}^{n \times p}$ and $\mathrm{rank}(X) < p$, the criterion $\|y - X\beta\|_2^2$ has infinitely many minimizers. One with smallest ℓ_2 norm can be computed using QR
- Banded matrix factorizations: if $A \in \mathbb{S}^n_{++}$ is k-banded, then we can compute its Choleksy decomposition in $nk^2/4$ flops, and apply it in $2nk$ flops
- Sparse matrix factorizations: this is in general a lot trickier, and can require very complex pivoting schemes. Theoretical analysis is loose, but practical performance is extremely good. See Davis (2006), "Direct methods for sparse linear systems" and SuiteSparse

Alternative indirect methods

So far we've been talking about direct methods for linear systems. These return the exact solution (in perfect computing environment)

Indirect methods (iterative methods) produce $x^{(k)}$, $k = 1, 2, 3, \ldots$ converging to a solution x. Most often used for very large, sparse systems

- Jacobi iterations are the most basic approach. Suppose that $A \in \mathbb{S}^n_{++}$, initialize $x^{(0)} \in \mathbb{R}^n$, and repeat for $k = 1, 2, 3, \ldots$

$$x_i^{(k+1)} = \left(b_i - \sum_{j \neq i} A_{ij} x_j^{(k)} \right) / A_{ii}, \quad i = 1, , \ldots n$$

- Gauss-Seidl iterations are similar but always use most recent iterates, i.e., use $\sum_{j<i} A_{ij} x_j^{(k+1)} + \sum_{j>i} A_{ij} x_j^{(k)}$ instead of above sum. Gauss-Seidl iterations always converge, but Jacobi iterations do not

- Gradient descent on $f(x) = \frac{1}{2}x^T Ax - b^T x$: this repeats

$$r^{(k)} = b - Ax^{(k)}$$
$$x^{(k+1)} = x^{(k)} + t_{\text{exact}} \cdot r^{(k)}$$

  Since $A \in \mathbb{S}^n_{++}$, the criterion f is strongly convex, implying linear convergence. But the contraction depends adversely on $\kappa(A)$. That is, gradient directions $r^{(k)}$ are not diverse enough across iterations

- Conjugate gradient method replaces gradient directions above with clever directions $p^{(k)}$ satisfying

$$p^{(k)} \in \text{span}\{Ap^{(1)}, \ldots Ap^{(k-1)}\}^{\perp}$$

  Note these directions are constructed to be diverse. Conjugate gradient method still uses one A multiplication per iteration, and in principle, it takes n iterations or much less. In practice, this is not true (numerical errors), and preconditioning is used

References and further reading

General:

- S. Boyd, Lecture notes for EE 264A, Stanford University, Winter 2014-2015
- S. Boyd and L. Vandenberghe (2004), "Convex optimization", Appendix C
- G. Golub and C. van Loan (1996), "Matrix computations", Chapters 1–5, 10

Sparse numerical linear algebra/special systems:

- T. Davis (2006), "Direct methods for sparse linear systems". Find his state-of-the-art (and free) C++ package SuiteSparse at `http://faculty.cse.tamu.edu/davis/suitesparse.html`
- N. Vishnoi (2013), "$Lx = b$; Laplacian solvers and their applications"