Computational Mathematics for Learning and Data Analysis

Federico Matteoni

A.A. 2021/22

Index

0.1 Introduction 2

0.1 Introduction

Exam: project (groups of 2) + oral exam.
This course's goals is to make sense of the huge amounts of data, take something big and unwieldy and produce something small that can be used, a **mathematical model**.
The mathematical model should be accurate, computationally inexpensive and general, but generally is not possible to have all three. General models are convenient (work once, apply many), they are parametric so we need to learn the right values of the parameters. Fitting is finding the model that better represents the phenomenon given a family of possible models (usually, infinitely many). Is an optimization model and usually is the computational bottleneck.
ML is better than fitting because fitting reduces the training error, the empirical risk, but ML reduces the test error, so the generalization error.
Solve general problem $min_{x \in S} f(x)$, with Poloni solve $min_{x \in R^n} ||Ax - b||_2$ which is easier and can be solved exactly.

Quick recap of linear algebra

Matrix - Vector multiplication, with $A \in R^{4\times 3}, c \in R^3, b \in R^4$

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ \vdots & \vdots & \vdots \\ A_{41} & A_{42} & A_{43} \end{bmatrix} \cdot \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} \qquad \begin{array}{l} b_i = \sum_{j=1}^{4} A_{ij} c_j \\ A_{11}c_1 + A_{12}c_2 + A_{13}c_3 = b_1 \end{array}$$

or linear combination of the columns

$$\begin{bmatrix} A_{11} \\ A_{21} \\ A_{31} \\ A_{41} \end{bmatrix} c_1 + \begin{bmatrix} A_{12} \\ A_{22} \\ A_{32} \\ A_{42} \end{bmatrix} c_2 + \begin{bmatrix} A_{13} \\ A_{23} \\ A_{33} \\ A_{43} \end{bmatrix} c_3 + \begin{bmatrix} A_{14} \\ A_{24} \\ A_{34} \\ A_{44} \end{bmatrix} c_4 = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}$$

with c_1, c_2, c_3 and c_4 called coordinates.

Basis: tuple of vectors $v_1, v_2, \ldots, v_n \mid$ you can write all vectors b in a certain space as a linear combination $v_1\alpha_1 + v_2\alpha_2 + \ldots + v_n\alpha_n$ with **unique** $a_1, \ldots, a_n$. The canonical basis is

$$c_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad c_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad c_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad c_4 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

and, for example

$$\begin{bmatrix} 3 \\ 5 \\ 7 \\ 9 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \cdot 3 + \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \cdot 5 + \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \cdot 7 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \cdot 9$$

Image ImA = set of vectors b that we can reach with A

Kernel $KerA$ = set of vectors $x \mid Ax = 0$ ($x = 0$ is certainly one, there may be others)

Invertible A if this problem has exactly one solution.
$\forall\, b \in R^m$, A must be square and the columns of A are a basis of $R^m \Rightarrow x = A^{-1}b$ where A^{-1} is another square matrix $\mid A \cdot A^{-1} = A^{-1} \cdot A = I$ identity matrix (1 on the diagonal, 0 otherwise)
Implementation detail: `inv(A) * b` is not the best choice. Better: in Python `scipy.linalg.solv(A, b)` or, in Matlab, `A  b`.

Cost, with $A \in R^{m\times n}, B \in R^{n\times p}, C \in R^{m\times p}$ (vectors $\Leftrightarrow n \times 1$ matrices), then the cost of multiplication is $mp(2n-1)$ floating point ops (*flops*), or $O(mnp)$.
In particular, A, B squared $\Rightarrow AB$ costs $O(m^3)$. With A, v vector $\Rightarrow Av$ costs $O(m^2)$. Faster alternatives are not worth it usually. And remember that $AB \neq BA$ generally, and also that $CA = CB \not\Rightarrow A = B$ with C matrix.
If there's $M \mid MC = I$, then $A = (MC)A = (MC)B = B$ (multiplying *on the left* by M on both sides)

Why a real valued function? Strong assumption, given x' and x'', I can always tell which one I like best (**total order** of R). Often more than one objective function, with contrasting and/or incomparable units (ex: loss function vs regularity in ML).
But R^k with $k > 1$ has no total order $\Rightarrow$ no *best* solution, only non-dominated ones.
Two practical solutions: maximize return with budget on maximum risk or maximize...
Even with a single objective function optimization is hard, impossible if f has no minimum in X (so, the problem P is unbounded below. Hardly ever happens in ML, because loss and regularization are ≥ 0
Also impossible if $f > -\infty$ but $\nexists x$, for example in $f(x) = e^x$. However plenty of ϵ-approximate solutions (ϵ-optima).
On PC $x \in R$ is in fact $x \in Q$ with up to 16 digits precision, so approximation errors are unavoidable anyway. Exact algebraic computation is possible but usually slow, and ML is going the opposite way (less precision: floats, half, small integer weights...).
Anyway finding the exact x_* is impossible in general.

Optimization need to be approximate Absolute gap, relative gap... but in general computing the gap is hard because we don't know f_*, which is what we want to estimate. So it's hard to estimate how good a solution is. Could argue that this is the "issue" in optimization: compute an estimate of f_*.

Optimization at least possible The f's spikes can't be arbitrarily narrow, so f cannot change too fast
f Lipsichitz continuous (L-c) on X: $\exists\, L > 0 \mid |f(x) - f(y)| \leq L|x - y| \quad \forall x, y \in X$
f L-c $\Rightarrow$ doesn't "jump" and one ϵ-optimum can be found with $O(\frac{LD}{\epsilon})$ evaluations by uniformly sampling X with step $\frac{2\epsilon}{L}$
Bad news: no algorithm can work in less that $\Omega(\frac{LD}{\epsilon})$, but it's the worst case of f (constant with one spike).
Number of steps is inversely proportional to accuracy: just not doable for small ϵ. Dramatically worse with $X \subset R^n$.
Also generally L is unknown and not easy to estimate, but algorithms actually require/use it.

Local Optimization Even if I stumble in x_* how do I recognize it? This is the difficult thing. Simpler to start with a weaker condition: x_* is the local minimum if it solves $min\{f(x) \mid f \in X(x_*, \epsilon) = [x_* - \epsilon, x_* + \epsilon]\}$ for some $\epsilon > 0$.
Stronger notion: **strict** local minimum if $f(x_*) < f(y)$
f (strictly) unimodal on X if has minimum $x_* \in X$ and it is (strictly) decreasing on the left $[x_-, x_*]$ and (strictly) increasing on the right $[x_*, x_+]$
Most functions are not unimodal, but they are if you focus on the attraction basin of x_* and restrict there. Unfortunately it's true for every local optimum, they all look the same.
Once in the attraction basin, we can restrict it by evaluating f in two points and excluding a part. How to choose the part so that the algorithm go as fast as possible? Each iteration dumps the left or the right part, don't know which $\Rightarrow$ should be equal $\Rightarrow$ select $r \in (\frac{1}{2}, 1), x'_- = x'_- + (1 - r)D, x'_+ = x_- + rD$
Faster if r larger $\Rightarrow r = \frac{D}{2} + \epsilon = x'_\pm = x_- + \frac{D}{2} \pm \epsilon$ but next iteration will have two entirely different x'_-, x'_+ to evaluate f on.

Optimally choosing the iterates

Norms

Eigenvalues and eigenvectors

Semidefinite

Recall theorem $\lambda_{min}(x^T x) \leq x^T Q x \leq \lambda_{max}(x^T x)$ for all symmetric Q so $\lambda_{min} \leq \frac{x^T Q x}{x^T x} \leq \lambda_{max}$
Slightly different form $\lambda_{min} \leq z^T Q z \leq \lambda_{max}$ for all vectors z with $||z|| = 1$. Equivalent to the other form with $x = \alpha z$, for $\alpha = ||x||$ and a vector z with $||z|| = 1$

$$\frac{x^T Q x}{x^T x} = \frac{(\not{\alpha} z)^T Q (\not{\alpha} z)}{\alpha^2}$$

Generalization for complex matrices $||x||^2 = |x_1|^2 + \ldots + |x_n|^2$, $x^T \longrightarrow \overline{x^T} = x^*$. For orthogonal matrices $U^* U = I \Rightarrow U$ is unitary. For symmetry $Q^* = Q \Rightarrow Q$ is Hermitian.

Singular Value Decomposition Each $A \in R^{n\times n}$ can be decomposed as $A = U\Sigma V^T$ with U, V orthogonal and Σ diagonal with σ_i on the diagonal with $\sigma_1 \geq \ldots \geq \sigma_n \geq 0$.
The first notable difference is it exists for every square matrix. The second difference is V^T which is not the inverse of U.
Another notation is $[u_1|u_2|\ldots|u_m]\begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_m \end{bmatrix}\begin{bmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_m^t \end{bmatrix} = u_1\sigma_1 v_1^T + \ldots + u_m\sigma_m v_m^T$ sum of m rank-1 matrices
Geometric idea: there is an orthogonal basis $v_1, \ldots, v_m$ so that A maps $\{v_i\}$ into multiples of another orthogonal basis $Av_i = u_i\sigma_i$
$\{\sigma_i\}$: singular values of A, defined uniquely for each A
Rectangular SVD: each $A \in R^{m\times n}$ can be decomposed as $A = U\Sigma V^T$ where $U \in R^{m\times m}$, $V \in R^{n\times n}$ are orthogonal and $\Sigma \in R^{m\times n}$ is diagonal, so $\sigma_{i,j} = 0$ whenever $i \neq j$ again with $\sigma_1 \geq \ldots \geq \sigma_{min(m,n)} \geq 0$
So only the first n vectors of U matches with non-zero values in Σ, all the last $n - m$ columns combine with zeroes.
$= u_i\sigma_1 v_1^T + \ldots + u_n\sigma_n v_n^T$ with $u_{n+1}, \ldots, u_m$ not used. So we can "chop off" the unused parts and get the same result.
$= u_i\sigma_1 v_1^T + \ldots + u_{min(m,n)}\sigma_{min(m,n)} v_{min(m,n)}^T$
In Matlab, `svd(A, 'econ')` costs $max(m,n)\cdot min(m,n)^2$, still cubic but linear in the largest dimension. the full `[U, S, V] = svd(A)` cannot be linear because one of the outputs will be a huge orthogonal matrix of $max(m,n) \times max(m,n)$, so it will cost more in time and memory.

The rank of a matrix A is equal to the number of non-zero σ_i. $\sigma_1 \geq \ldots$ so at one point a $\sigma_r > \sigma_{r+1} = \ldots = \sigma_{min(m,n)} = 0$
Given $A = U\Sigma V^T$ we can compute $A^TA = (U\Sigma V^T)^T(U\Sigma V^T) = V\Sigma^T U^T U\Sigma V V^T = V\Sigma^T\Sigma V^T$ with $\Sigma^T\Sigma$ diagonal and $V\Sigma^T\Sigma V^T$ is both an eigenvalue decomposition and an SVD. This proved that the eigenvalues of A^TA Are the squares of the singular values of A plus additional zeroes for dimension reasons.

$$||A||_2 = ||U\Sigma V^T||_2 = ||\Sigma V^T||_2 = ||\Sigma||_2 = \Sigma_1$$

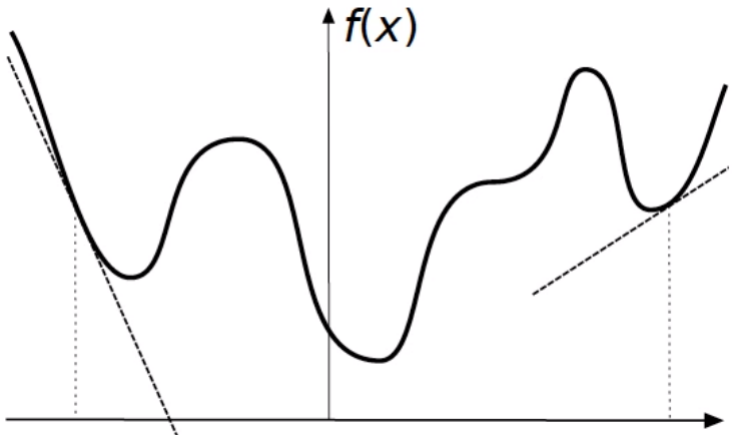
$$||A|| = max_{||z||=1}||\Sigma V^T|| = \sqrt{\sigma_1^2 z_1^2 + \ldots + \sigma_n^2 z_n^2} \leq \sigma_1\sqrt{z_1^2 + \ldots + z_n^2} = \sigma_1||z|| = \sigma_1$$

$$||A||_F = ||U\Sigma U^T|| = \ldots = \Sigma_1$$

Eckart-Young Theorem Most important property of the SVD decomposition.
we are interested in approximating A with matrices of rank $\leq K$, if $K = 1$ this means find two vectors u, v so that $A = u^Tv$, with $K = 2$ then $A = u_1^Tv_1 + u_2^Tv_2$. What is "how close": $min_{rank(X)\leq K}||A - X||$. The theorem states that the solution is related to SVD.
The optimal solution of $min_{rank(X)\leq X}||A-X||$ is $X = u_1\sigma_1 v_1^T + \ldots + u_k\sigma_k v_k^T$ where $A = u_1\sigma_1 v_1^T + \ldots + u_{min(m,n)}\sigma_{min(m,n)} v_{min(m,n)}^T$ is an SVD, $A = U\Sigma V^T$.

Matrix Norms

To make it go faster, give it more information Two points are needed to see in which direction f is decreasing. If we could see this directly we could make it with one point, faster. Look at the linear function that best locally approximates f, trusty old first derivative $f'(x)$: slope of the tangent line to the graph of f in x
First order model of f at x: $L_x(y) = f'(x)(y - x) + f(x)$. $L_x(y) = f(y) \forall\ y \in [x - \epsilon, x + \epsilon]$ for some small $\epsilon > 0$. x_* local minimum then $f'(x_*) = 0$ = root of f' = stationary point. If $f'(x) <$ or $f'(x) > 0$, then x is clearly not a local minimum. Hence, $f'(x) = 0$ for the all local minima (hence global) but this is true for the local maxima (hence global).



In simple cases we get the answer by a closed formula. In $f = bx + c$ linear and $b > 0$ then the minimum is x_- and maximum is x_+, viceversa if $b < 0$. For $f = ax^2+bx+c$ quadratic then if $a > 0$ the minimum is $\min\{x_+, max\{x_*, x_-\}\}$ and the maximum is $\text{argmax}\{f(x_-), f(x_+)\}$, and viceversa if $a < 0$.

Golden Ratio Search

Dichotomic Search f' continuous + intermediate value theorem gives that $f'(x_-) < 0 \wedge f'(x_+) > 0 \Rightarrow \exists\, x \in [x_-, x_+] \mid f'(x) = 0$
$\rightarrow$ **dichotomic search**. The condition $f'(x_-) < -\epsilon, f'(x_+) > \epsilon$ is important. What if is not satisfied? Obvious solution, moving the interval more and more to the right where the derivative is possible the same in reverse of x_- with $\Delta x = -1$. This works in practice for all "reasonable" functions. Works if f coercive ($\lim_{|x|\to\infty} f(x) = \infty$

$f' \in C^0 \Leftrightarrow f \in C^1 \Leftrightarrow$ continuously differentiable $\Rightarrow f \in C^0$

$f'' \in C^0 \Leftrightarrow f \in C^2 \Leftrightarrow f' \in C^1 \Rightarrow f' \in C^0 \Rightarrow f \in C^1 \Rightarrow f \in C^0$

$f \in C^1$ globally L-c ib $X \Rightarrow |f'(x)| \leq L \forall\, x \in X$

Extreme value theorem $f \in C^0$ on $X = [x_-, x_+]$ finite $\Rightarrow \max\{f(x) \mid x \in X\} < \infty$, $\min\{f(x) \mid x \in X\} > -\infty$

$f \in C^1$ on X finite $\Rightarrow f$ globally L-c on X
Best possible case is $f \in C^2$ on finite $X \Rightarrow$ both f and f' globally L-c on X