Computational Mathematics for Learning and Data Analysis

Federico Matteoni

A.A. 2021/22

Index

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.

Capitolo 1

Numerical Analysis

1.1 Quick recap of linear algebra

Matrix - Vector multiplication, with $A \in R^{4\times3}, c \in R^3, b \in R^4$

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

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.

Norms

$$||x||_2 = \sqrt{x_1^2 + x_2^2 + \ldots + x_n^2} = \sqrt{x^T x} = [x_1 \ldots x_n] \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_b \end{bmatrix}$$

Many matrices preserve norm 2

Orthogonal A square matrix $U \in R^{n \times n}$ is orthogonal if $U^T U = I \vee UU^T = I \vee U^{-1} = U^T$ equivalently
Theorem: for an orthogonal matrix $U \in R^{n \times n}$ and every vector x, $||Ux|| = ||x||$
More generally, $x^T y = (Ux)^T (Xy)$ for all vectors $x, y \in R^n$
The columns of a orthogonal matrix are **orthonormal**: $u_i^T u_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$
If $U, V \in R^{n \times n}$ are orthogonal $\Rightarrow UV$ is orthogonal

Eigenvalues and eigenvectors Given a square matrix $A \in R^{m \times m}$, if $Av = \lambda v$ for $v \in R^m, \lambda \in R$ then we call λ and **eigenvalue** and v an **eigenvector** of A.
For many matrices we can find $v_1, \ldots, v_m$ eigenvectors that are a basis of R^m, i.e. $[v_1 | \ldots | v_m] = V$ is invertible.
$A = V \cdot \Lambda \cdot V^{-1} \Leftrightarrow AV = V\Lambda$, with $\Lambda = \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_m \end{bmatrix}$
Given $x \in R^m$ I can write $x = v_1\alpha_1 + \ldots + v_m\alpha_m$ then it's easy to compute the coordinates $Ax = A(v_1\alpha_1 + \ldots + v_m\alpha_m) = v_1(\lambda_1\alpha_1) + \ldots + v_m(\lambda_m\alpha_m)$
Also $A^k x = v_1(\lambda_1^k \alpha_1) + \ldots + v_m(\lambda_m^k \alpha_m)$
Eigenvalue decompositions do not exists unique for all matrices. If v is an eigenvector, then $v \cdot \alpha$ is still one, and if v, w are eigenvectors of the same eigenvalue then $v + w$ is still an eigenvector of the same eigenvalue, and the same for $\alpha v + \beta w$ (linear combination). Some matrices have complex eigenvalues/eigenvectors only, and some do not have enough eigenvectors to make a masis.

Symmetry A is symmetric if $A = A^T$

Spectral Theorem For a symmetric matrix $A \in R^{m \times m}$ we can always write $A = U\Lambda U^{-1}$

Quadratic form Given a symmetric matrix $Q = Q^T \in R^{m \times m}$ we can consider the function $x \in R^m \mapsto f(x) = x^T Q x \in R$
Theorem For all symmetric matrix $Q \in R^{m \times m}$, $\lambda_{min}||x||^2 \leq x^T Q x \leq \lambda_{max}||x||^2$, where $\lambda_{min}, \lambda_{max}$ are the smallest and largest eigenvalue of Q.

Positive Semidefinite $Q = Q^T$ is positive semidefinite if all eigenvalues $\lambda_i \geq 0 \Leftrightarrow \lambda_{min} \geq 0$ hence $x^T Q x \geq 0$ for all $x \in R^m$
Positive definite if all eigenvalues $\lambda_i > 0 \Leftrightarrow \lambda_{min} > 0$ hence $x^T Q x > 0$ for all $x \in R^m, x \neq 0$

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

$$\frac{x^TQx}{x^Tx} = \frac{(\not{\alpha}z)^TQ(\not{\alpha}z)}{\not{\alpha}^{\not{2}}}$$

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 $[u_1|u_2|\ldots|u_m]\begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_m \end{bmatrix}\begin{bmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_m^t \end{bmatrix} = u_1\sigma_1v_1^T + \ldots + u_m\sigma_mv_m^T$ 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.
$= u_i\sigma_1v_1^T + \ldots + u_n\sigma_nv_n^T$ with $u_{n+1}, \ldots, u_m$ not used. So we can "chop off" the unused parts and get the same result.
$= u_i\sigma_1v_1^T + \ldots + u_{min(m,n)}\sigma_{min(m,n)}v_{min(m,n)}^T$
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 $A^TA = (U\Sigma V^T)^T(U\Sigma V^T) = V\Sigma^TU^TU\Sigma VV^T = V\Sigma^T\Sigma V^T$ 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.

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

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

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

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 $X = u_1\sigma_1v_1^T + \ldots + u_k\sigma_kv_k^T$
where $A = u_1\sigma_1v_1^T + \ldots + u_{min(m,n)}\sigma_{min(m,n)}v_{min(m,n)}^T$ is an SVD, $A = U\Sigma V^T$.

Ranks If A has rank 1, then $A = u \cdot v^T$ and $A_{ij} = u_i \cdot v_j$
If A has rank 2, then $A = u_1 \cdot v_1^T + u_2 \cdot v_2^T$ and $A_{ij} = (u_1)_i \cdot (v_1)_j + (u_2)_i \cdot (v_2)_j$

1.2 SVD Approximation

$X_1 = u_i\sigma_1v_1^T$ = best approximation score of student $i \cdot n\cdot$ best approximation difficulty of exercise j
As a statistical estimator: suppose my scores are of the form $A_{ij} = u_iv_j + \epsilon_{ij}$ with ϵ_{ij} being the error in the score for instance gaussian with variance λ.
The rank 1 approx of $(u_1\sigma_1)$, v_1 given by SVD is the one that minimizes $\sum(A_{ij} - u_iv_j)^2 = ||A - X_1||_F^2 = \sum\epsilon_{ji}^2$. This

is the maximum-likelihood estimation of abilities $(u_1)_i, (v_1)_j$.

The best rank 2 approximation is $X_2 = u_1\sigma_1 v_1^T + u_2\sigma_2 v_2^T$ which can be viewed as first approximation plus corrections.

$\sigma_1 >> \sigma_2, \ldots$ then A very close to rank 1.
$\sigma_1, \sigma_2 >> \sigma_3, \ldots$ then A very close to rank 2, and so on.

Best approximations $X =$ best rank-1 approx of I, $X = u_1\sigma_1 v_1^T$, $x_{ij} = (u_1)_j \sigma_1 (v_1^T)_j$
Best rank-2 $X_2 = u_1\sigma_1 v_1^T + u_2\sigma_2 v_2^T$
Best rank-3 $X_3 = u_1\sigma_1 v_1^T + u_2\sigma_2 v_2^T + u_3\sigma_3 v_3^T$ And so on...
The original image was $256\times256 = 2^{16}$ reals. The compressed version, with $k = 25$ we have $256 \cdot 5 \cdot 2 + 25$ which is about a factor of 5 less.

What is $||A - X_k||_F = \sqrt{\sum (a_{ij} - x_{ij})^2} = ||U\Sigma V^T - U[\text{main diagonal of } \sigma_i \text{ untile } \sigma_k]V^T|| = ||U([\ldots = \sqrt{\sum_{i=k+1}^{min(m,n)} \sigma_i^2}$

Linear Least Squares problems Given vectors $a_1, \ldots, a_n \in R^m$, so that $A = [a_1|\ldots|a_n] \in R^{m\times n}$, and targets $b \in R^m$, find $x_1, \ldots, x_n \in R \mid a_1x_1 + \ldots + a_nx_n = b$
Not always solvable, for example $\begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix} x_1 + \begin{bmatrix} 1 \\ 3 \\ 0 \end{bmatrix} x_2 = \begin{bmatrix} 5 \\ 5 \\ 1 \end{bmatrix}$ because the third component is always $0 \neq 1$. As a backup question, how close can I get to b? I can get $\begin{bmatrix} 5 \\ 5 \\ 0 \end{bmatrix}$
In general $min_{x\in R^m}||Ax - b||_2 = min_{x\in R^m}\sqrt{\sum((Ax)_i - b_i)^2}$
The special case is $m = n$, i.e. # vectors = length, then the problem is solvable $\Leftrightarrow$ the vectors are a basis $\Leftrightarrow$ or A invertible. Typical case il A long thin, we cannot get all vectors b but still $min_{x\in R^m}||Ax - b||_2$ is a question that makes sense.

Polynomial Fitting Statistical version: given (x_i, y_i), what is the choice of coefficients that "most likely" generated them? I can get (x_i, y_i) starting from every polynomial, with the right set of random numbers. The **maximum likelihood estimator** on this problem is $min_{coeff}||Ax - y||_2^2$

Theory of least-squares problems When does $min||Ax - b||_2$ have a unique solution? With $A \in R^{m\times n}$
We know that if $m = n$ then $Ax = b$ has a unique solution $\Leftrightarrow$ A is an invertible matrix. If this happens, then $O = min||Ax - b||$ with unique x.
We say that $A \in R^{m\times n}$ has **full column rank** if $Ker(A) = \{0\} \Leftrightarrow$ there is no $z \in R^n z \neq 0 \mid Az = 0 \Leftrightarrow rk(A) = n$ and this can only happen is $m \geq n$

Theorem The least-squares problem $min||Ax - b||$ has unique solution $x \Leftrightarrow A$ has full column rank.
Lemma: A has full column rank $\Leftrightarrow A^TA$ is positive definite.
Proof $Ax \neq 0 \ \forall\, z \in R^n, z \neq 0$

$\Leftrightarrow\ ||Az||_2 \neq 0 \ \forall\, z \in R^n, z \neq 0$

$\Leftrightarrow\ ||Az||_2^2 \neq 0 \ \forall\, z \in R^n, z \neq 0$

$\Leftrightarrow\ (Az)^T(Az) \neq 0 \ \forall\, z \in R^n, z \neq 0$

$\Leftrightarrow\ z^TA^TAz \neq 0 \ \forall\, z \in R^n, z \neq 0 \longleftarrow$ definition of $A^TA > 0$

By manipulating the original problem $min_{x\in R^n}||Ax - b||_2$ we obtain

$$min||Ax - b||_2 = minx^TA^TAx - 2b^TAx + b^Tb \Leftrightarrow f(x) = x^TQx + q^Tx + c$$

which is a quadratic problem and find that it has a unique minimum $x \Leftrightarrow$ it is strongly convex $\Leftrightarrow Q > 0$
$f(x)$ convex $\Leftrightarrow Q \geq 0$, strongly/strictly convex $\Leftrightarrow Q > 0$

So the least-squares problem $min_x||Ax - b||$ has unique solution

$\Leftrightarrow$ $f(x)$ has a unique minimum point

$\Leftrightarrow$ $2A^TA = Q > 0$ (positive definite)

$\Leftrightarrow$ $A^TA > 0 \Leftrightarrow A$ has full column rank (for the lemma)

The minimum is when $grad\, f(x) = 0 \Leftrightarrow 2Qx + q = 0 \Leftrightarrow 2A^TAx - 2A^Tb = 0$ so when $A^TAx = A^Tb$ square linear system, with A^TA invertible (because positive definite).
x is obtained (intuitively) from multiplying $Ax = b$ on the left with A^T.

Algorithm

1. Form A^TA, $n \times m \cdot m \times n$ product so it costs $2mn^2$ floating point operations (flops) plus lower order terms
2. Form A^Tb, costs $2mn$ flops lower order terms
3. Solve $A^TAx = A^Tb$ (for example with gaussian elimination or LU factorization) costs $\frac{2}{3}n^3$ flops plus lower order terms

If $m \geq n$ then the overall complexity is $O(mn^2)$ same as SVD.
Possible optimizations:

1. A^TA symmetric so can compute only upper triangle then mirror the rest so from $2mn^2$ becomes mn^2 flops
2. Already a cheap step
3. Other algorithms to solve this linear system because the matrix A^TA is positive definite (example: Cholesky factorization, complexity is $\frac{1}{3}n^3$ flops, half the cost)

Pseudoinverse $x = A^TA^{-1}b$ can be denoted as the product of $A^+ = A^TA^{-1}A^T$ and b. A^+ is the pseudoinverse, or **Moore-Penrose pseudoinverse**. The definition is valid only when A has full column rank. If $A \in R^{m\times n}$ then $A^+ \in R^{n\times m}$. Note that $A^+A = (A^TA)^{-1}(A^TA) = I \in R^{n\times n}$, while $AA^+ = A(A^TA)^{-1}A^T \neq I \in R^{m\times m}$. The latter is impossible, because the columns of AA^+ are linear combinations of the columns of A, so AA^+ has rank of at most n. As consequences, if x_1 is solution of $min||Ax - b_1||$ and x_2 is solution of $min||Ax - b_2||$ then $x_1 + x_2$ is solution of $min||Ax - (b_1 + b_2)||$

Sometimes ML problems are formulated "from the left side". With $w \in R^{1\times n}$ row vector of weights, then $X \in R^{n\times m}$ short-fat ($n \leq m$) that has a row for each "feature" in the input pattern.
$y \in R^{1\times m}$ row vector "target"
The problem is $min||wX - y||$, same problem just transposed. Solution $w = yX^+$ with $X^+ = X^T(XX^T)^{-1}$ if X has full row rank.

1.3 QR factorization

Given $x \in R^n$, find an orthogonal matrix H such that Hx is a vector of the form $s \cdot e_1 = \begin{bmatrix} s \\ 0 \\ \vdots \\ 0 \end{bmatrix}$ with $e_1 = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$

Since H is orthogonal, $||x|| = ||\begin{bmatrix} s \\ 0 \\ \vdots \\ 0 \end{bmatrix}|| = |s| \Rightarrow s = \pm||x||$. We will find H in a very specific form

$$H = I - \frac{2}{v^Tv}vv^T =$$

$$= I - \frac{2}{||v||^2}vv^T =$$

$$= I - 2uu^T$$

for a certain $v \in R^n$ with $u = \frac{v}{||v||}$, so it has norm = 1 and is parallel to v. This form is called **Householder reflector**.
Lemma: for every $v \in R^n$, H is orthogonal *and* symmetric. Proof:

Symmetric: $H^T = (I - 2uu^T)^T = I^T - (2uu^T)^T = I - 2uu^T = H$

Orthogonal: $H^T H = H^2 = (I - 2uu^T)(I - 2uu^T) = I - 2uu^T - 2uu^T + 4uu^T uu^T$ but $u^T u = ||u||^2 = 1$ so $= I - \not{4}uu^T + \not{4}uu^T = I$

By computing Hx as $x - 2u(u^T x)$, meaning you compute $\alpha = u^T x$ in $O(n)$ and then compute $x - 2u\alpha$ in $O(n)$, you can reduce the complexity of Hx from $O(n^2)$ to $O(n)$. By doing the same on every column you reduce HA for an arbitrary $A \in R^{n \times n}$ from $O(n^3)$ to $O(n^2)$.
Lemma: given any two vectors $x, y \in R^n$ with $||x|| = ||y||$, the Householder reflector built with $v = x - y$ is such that $Hx = y$

Numerical problems If x_1 is close to $s = ||x||$ there are problems, because we do a subtraction between close numbers. Quick fix: switch to $s = -||x||$, then $x_1 - s = x_1 + \sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}$, an addition between positive numbers whenever $x_1 > 0$. In general, we choose:

if $x_1 \geq 0$, we take $s = -||x||$ and $y = \begin{bmatrix} -||x|| \\ 0 \\ \vdots \\ 0 \end{bmatrix}$

if $x_1 z 0$, we take $s = ||x||$ and $y = \begin{bmatrix} ||x|| \\ 0 \\ \vdots \\ 0 \end{bmatrix}$

This way $x_1 - s$ is never a "true subtraction", because x_1 and s have different signs.

Theorem $\forall\, A \in R^{m \times n}\ \exists\, Q \in R^{m \times m}$ orthogonal, $R \in R^{m \times n}$ upper triangular $\mid A = QR$

When $n = 1$ we already solved: $\forall\, x \in R^{m \times 1}\ \exists$ a Householder reflector $H \mid Hx = \begin{bmatrix} s \\ 0 \\ \vdots \\ 0 \end{bmatrix} \Leftrightarrow x = H \begin{bmatrix} s \\ 0 \\ \vdots \\ 0 \end{bmatrix}$ with x being the $m \times 1$ matrix, H orthogonal and the array is upper triangular.
The general case is $A \in R^{m \times n}$:

1. $[u_1, s_1]$ = `householder_vector(A(:,1))` $H_1 = I - 2u_1 u_1^T$
   $H_1 A = \begin{bmatrix} s_1 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \end{bmatrix} = A_1$
2. $A_2 = H_2 A_1 = \begin{bmatrix} * & s_2 & * & * \\ * & 0 & * & * \\ * & 0 & * & * \\ * & 0 & * & * \\ * & 0 & * & * \end{bmatrix}$ won't work because it spoils the first column.
   So $[u_2, s_2]$ = `householder_vector(`A_1`(2:`m`,2))`, which the 2nd to mth row of the second column of A_1. Because if we multiply $\left[\begin{array}{c|c} 1 & 0 \\ \hline 0 & H_2 \end{array}\right] \cdot \left[\begin{array}{c} B \\ \hline C \end{array}\right]$ we get $\left[\begin{array}{c} B \\ \hline H_2 \cdot C \end{array}\right]$, so we say that $Q_2 = \left[\begin{array}{c|c} 1 & 0 \\ \hline 0 & H_2 \end{array}\right]$
   and $Q_2 A_1 = \begin{bmatrix} s_1 & * & * & * \\ 0 & s_2 & * & * \\ 0 & 0 & * & * \\ 0 & 0 & * & * \\ 0 & 0 & * & * \end{bmatrix} = A_2$

3. With H_3 from $[u_3, s_3]$ = `householder_vector(`A_2`(3:`m`,3))` we do

$$Q_3A_2 = \left[\begin{array}{cc|ccc} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & & & \\ 0 & 0 & & H_3 & \\ 0 & 0 & & & \end{array}\right] A_2 = \left[\begin{array}{cccc} s_1 & * & * & * \\ 0 & s_2 & * & * \\ 0 & 0 & s_3 & * \\ 0 & 0 & 0 & * \\ 0 & 0 & 0 & * \end{array}\right] = A_3$$

4. $H_4 = I - 2u_4u_4^T$ from $[u_4, s_4]$ = `householder_vector(`A_3`(4:`m`,4))`

$$Q_4A_3 = \left[\begin{array}{c|c} I_3 & 0 \\ \hline 0 & H_4 \end{array}\right] A_3 = \left[\begin{array}{cccc} s_1 & * & * & * \\ 0 & s_2 & * & * \\ 0 & 0 & s_3 & * \\ 0 & 0 & 0 & s_4 \\ 0 & 0 & 0 & 0 \end{array}\right] = A_4$$

So $Q_4Q_3Q_2Q_1A = R$ is an upper triangular matrix.

$$\begin{aligned} \not{Q}_4^T \not{Q}_4Q_3Q_2Q_1A &= Q_4^TR \\ \not{Q}_3^T \not{Q}_3Q_2Q_1A &= Q_3^TQ_4^TR \\ \not{Q}_2^T \not{Q}_2Q_1A &= Q_2^TQ_3^TQ_4^TR \\ \not{Q}_1^T \not{Q}_1A &= Q_1^TQ_2^TQ_3^TQ_4^TR \end{aligned}$$

So $A = Q_1^TQ_2^TQ_3^TQ_4^TR$, and $Q_i = Q_i^T$ so we can omit transpose, giving $A = QR$ with Q orthogonal and R triangular.

The QR factorization can be used to solve least squares problems $min||Ax - b||_2$ with $A \in R^{m\times n}$ and $m \geq n$, $b \in R^m$.

$A = QR = [Q_1 \mid Q_2] \cdot \left[\begin{array}{c} R_1 \\ \hline \\ 0 \end{array}\right]$ with $Q_1 \in R^{m\times n}, Q_2 \in R^{m\times m-n}, R_1 \in R^{n\times n}$ and the last part of $n \times m - n$ zeroes. We rework the objective function

$$||Ax - b|| = ||QRx - b|| = ||Q^T(QRx - b)|| = ||Rx - Q^Tb|| = \left|\left|\left[\begin{array}{c} R_1 \\ 0 \end{array}\right] x - \left[\begin{array}{c} Q_1^T \\ Q_2^T \end{array}\right] b\right|\right| = \left|\left|\left[\begin{array}{c} R_1x - Q_1^Tb \\ -Q_2^Tb \end{array}\right]\right|\right|$$

How to make this norm as small as possible? The norm of the second block, $-Q_2^Tb$, doesn't depend on x. The norm of the first block does. Can I obtain $R_1x - Q_1^Tb = 0$? If R_1 is invertible, $R_1x = Q_1^Tb$ is a square linear system of solution $x = R_1^{-1}Q_1^Tb$. Algorithm:

1. Compute the QR factorization $A = QR = Q_1R_1$ (thin is enough) in $\frac{4}{3}n^3, 2mn^2$ operations.
2. Compute $c = Q_1^Tb$, with $2mn$ operations
3. Solve the linear system $R_1x = c$ with back-substitution, n^2 operations

Step 1 is the most expensive, cubic. When is R_1 invertible? Recall that the least-squares problem has unique solution $\Leftrightarrow$ A has full column rank $\Leftrightarrow$ A^TA is positive definite $\Leftrightarrow$ A^TA invertible $\Leftrightarrow$ R_1 invertible

$$A^TA = (QR)^TQR = R^T \not{Q}^T \not{Q}R = [R_1^T 0] \left[\begin{array}{c} R_1 \\ 0 \end{array}\right] = R_1^TR_1$$

The least-squares problem can also be solved with SVD. Given $A = USV^T$ we have

$$||Ax - b|| = ||USV^Tx - b|| = ||U^T(USV^Tx - b)|| = ||SV^Tx - U^Tb|| =$$

With $S = \left[\begin{array}{ccc} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \\ & 0 & \end{array}\right]$ and $V^Tx = y$ we have

$$= \left|\left|\left[\begin{array}{c} \sigma_1y_1 \\ \vdots \\ \sigma_ny_n \\ 0 \\ \vdots \\ 0 \end{array}\right] - \left[\begin{array}{c} u_1^Tb \\ \vdots \\ u_m^tb \end{array}\right]\right|\right| = \left|\left|\left[\begin{array}{c} \sigma_1y_1 - u_1^Tb \\ \vdots \\ \sigma_ny_n - u_n^Tb \\ -u_{n+1}^Tb \\ \vdots \\ -u_m^Tb \end{array}\right]\right|\right|$$

The first n entries become 0 if $y_i = \frac{u_i^T b}{\sigma_i}$

$$x = Vy = v_1 y_1 + \ldots + v_n y_n = v_1 \frac{u_1^T b}{\sigma_1} + \ldots + v_n \frac{u_n^T b}{\sigma_n}$$

$$x = V S_1^{-1} U_1^T b = A^+ b$$

The solution of $min||Ax - b||$ is unique $\Leftrightarrow$ $A^T A$ is invertible $\Leftrightarrow$ $A^T A = (USV^T)^T USV^T = VS^T \not{U}^T \not{U} SV^T = V \begin{bmatrix} \sigma_1^2 & & \\ & \ddots & \\ & & \sigma_n^2 \end{bmatrix} V^T$ which is the eigenvalue decomposition for $A^T A$

So $A^T A$ is invertible $\Leftrightarrow$ non of the σ_i are zeroes $\Leftrightarrow$ S_1 is invertible.

What happens if $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_r > \sigma_{r+1} = \ldots \sigma_n = 0$? We would have $\left|\left|\begin{bmatrix} \sigma_1 y_1 - u_1^T b \\ \vdots \\ \sigma_r y_r - u_r^T b \\ -u_{r+1}^T b \\ \vdots \\ -u_n^T b \\ -u_{n+1}^T b \\ \vdots \\ -u_m^T b \end{bmatrix}\right|\right|$

How to choose a special solution here? Let's define another problem: $min||x||$ of all $x \in$ solutions of $min||Ax - b||$. It's minimized when $y_1 = \frac{u_1^T b}{\sigma_1}, \ldots, y_r = \frac{u_r^T b}{\sigma_r}, y_{r+1} = 0, \ldots, y_n = 0$

The problem is that on computers the zeroes are very often not zeroes. Even including a term $v_{r+1} \frac{u_{r+1}^T b}{\sigma_{r+1}}$ with a small σ_{r+1} gives a huge contribution to the sum. In many case, stopping the sum early is beneficial: **truncated SVD**, with $i = 1, \ldots, k$ and $k < m$. The Eckart-Young aproximation theorem can help too.

Effect of noise in data Suppose to know $A + E = \overline{A}$, with A exact data, E error/noise/etc. and $\overline{A}$ the observed data. We have

$\sigma_1, \ldots, \sigma_n = \text{SVD}(A)$

$\overline{\sigma_1}, \ldots, \overline{\sigma_n} = \text{SVD}(\overline{A})$

Then $|\sigma_i - \overline{\sigma_i}| \leq ||E||_2$

The effect of noise is that there are longer relative changes to smaller singular values, another reason why it's beneficial to drop them.

Tikhonov Regularization/Ridge Regression Another solution, alternative to truncated SVD. Change the problem to $min_{x \in R^n} ||Ax - b||^2 + \lambda^2 ||x||^2 = min|| \begin{bmatrix} A \\ \lambda I \end{bmatrix} x - \begin{bmatrix} b \\ 0 \end{bmatrix} ||^2$ with solution $x = (A^T A + \lambda^2 I)^{-1} A^T b$

Sensitivity or conditioning of a problem A problem maps input to output: $A, b \mapsto X = A^{-1} b$ or $(x_i, y_i) \mapsto$ weights...

If $f(x, y) = x + y$ then $f(x + \delta, y) = x + y + \delta$, the input change is δ and the output change is $|x + y + \delta - (x + y)| = \delta$

$$f(x + \delta) = f(x) + \delta \cdot f'(x) + O(\delta^2)$$

The (absolute) condition number of a function f (of an input x) is the best possible bound K of the form

$$|f(x + \delta) - f(x)| \leq K|\delta| + o(\delta)$$

with K being the **condition number**

The absolute condition number of (differentiable) f in x is $|f'(x)| = K_{abs}(f, x)$ For **multivariate functions** there are possibly different notes of change across different directions. So, the **condition number** of a function $f : R^m \to R^n$ is the best constant K that I can use to bound

$$||f(x + \delta) - f(x)|| \leq K \cdot ||\delta|| + o(||\delta||)$$

The absolute condition number K_{abs} is $K_{abs}(f, x) = \lim_{\delta \to 0} sup_{||d|| \leq \delta} \frac{||f(x+\delta) - f(x)||}{||d||} K_{abs}(f, x) = || \bigtriangledown_x f||$ 2-norm of the gradient, if f differentiable.

The **condition number measures the sensitivity of a problem to changes in input**.

Example $x \in R^2, x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, f(x) = x_1 + x_2, d = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$
$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \delta \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} x_1 + \delta \\ x_2 + \delta \end{bmatrix}$

$$\frac{||f(x+\delta d) - f(x)||}{||\delta d||} = \frac{||x_1 + \delta + x_2 + \delta - x_1 - x_2||}{||\begin{bmatrix} \delta \\ \delta \end{bmatrix}||} = \frac{||2\delta||}{\sqrt{\delta^2 + \delta^2}} = \frac{2\delta}{\sqrt{2}\delta} = \sqrt{2}$$

which is the rate of change of perturbations parallel to d.
With $d = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ we would have $= 0$, and with $d = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ we would have $= 1$
Also checking other directions, it turns out that $\sqrt{2}$ is the largest possible rate of change (no coincidence that it is $|| \bigtriangledown_x f||$)

Linear Systems The condition number of solving square systems of linear equations.
$Ax = b, A \in R^{n \times n}, b \in R^n$, we perturb input b to $\overline{b} \in R^n$ and call $\overline{x}$ the solution of $A\overline{x} = \overline{b}$, while x solution of $Ax = b$.
The absolute bound is

$$||\overline{x} - x|| = ||A^{-1}\overline{b} - A^{-1}b|| = ||A^{-1}(\overline{b} - b)|| \leq ||A^{-1}|| \cdot ||\overline{b} - b||$$

and

$$||b|| = ||Ax|| \leq ||A|| \cdot ||x||$$

Putting all together gives

$$\frac{||\overline{x} - x||}{||x||} \leq ||A^{-1}|| \cdot ||A|| \cdot \frac{||\overline{b} - b||}{||b||}$$

because $K(A) = ||A|| \cdot ||A^{-1}||$ the condition number of A matrix.

What if we perturb $\overline{A} = A + \Delta$? One can prove, if $\overline{x}$ is solution of $\overline{A}\overline{x} = b$, that

$$\frac{||\overline{x} - x||}{||x||} \leq ||A|| \cdot ||A^{-1}|| \cdot \frac{||\overline{A} - A||}{||A||} + o(||\Delta||)$$

In terms of SVD we know that $||A|| = \sigma_1$. For a square invertible matrix $A = USV^T$ we have that $||A^{-1}|| = \frac{1}{\sigma_n}$, so $K(A) = ||A|| \cdot ||A^{-1}|| = \frac{\sigma_1}{\sigma_n}$.
A related quantity is the distance between A and the closest singular matrix. A matrix $B \in R^{n \times n}$ is singular if $rk(B) < n$. By the Eckhart-Young theorem, $min_{B\ sing}||A - B|| = min_{rk(B) \leq n-1}||A - B|| = \sigma_n$ because $B = \sum_{i=1}^{n-1} u_i \sigma_i v_i^T$.
Hence $\frac{1}{K(A)} = \frac{\sigma_n}{\sigma_1} = \frac{min_{B\ sing}||A-B||}{||A||}$

Least Squares Problem **Theorem**: if $A \in R^{m \times n}, m \geq n$ we define $K(A) = \frac{\sigma_1}{\sigma_n}$

Theorem The condition number of the least-squares problem $min||Ax - b||$ for a full column rank matrix $A \in R^{m \times n}, b \in R^m$

$$K_{rel, b \to x} \leq \frac{K(A)}{\cos\theta}$$

$$K_{rel, A \to x} \leq K(A) + K(A)^2 \cdot \tan\theta$$

where

$$\theta = \arccos\frac{||Ax||}{||b||}$$

Capitolo 2

Optimization

Making sense of the huge amounts of data generated and collected means taking something big and unwieldy and producing something small and nimble that can be used: a **mathematical model**. It should be: accurate (describes well the process), computationally inexpensive (fast), general (can be applied to many different processes). Typically impossible to have all three. Developing general models is useful, work once apply many. But the shape of the model controls the computational cost. How to get accuracy for any given application? A **model is parametric** and must **learn the right values of the parameters**. In other words, **fitting**: within the family of (usually) infinitely many models with the given shape, find the one that better represent your phenomenon. This is an optimization problem, and solving fitting is typically the bottleneck.

Fitting means minimizing training error, while machine learning means minimizing the generalization error.

Example: Linear estimation Phenomenon measured by one number y and believed to depend on a vector $x = [x_1, \dots, x_n]$. Available set of observations $(x_1, y_i), \dots, (x_m, y_m)$.
And optimistic assumption: the dependence is linear, hence $y = \sum_{i=1}^{n} w_i x_i + w_0 = w_x + w_0$ for fixed $n+1$ real parameters $w = [w_0, w_+ = [w_1, \dots, w_n]]$ and find the w for which is less untrue $min_w\ L(w) = ||y - xw||$

Example: Low-rank approximation A large, sparse matrix $M \in R^{n \times m}$ describes a phenomenon depending on pairs. Find a tall and thin $A \in R^{n \times k}$ and a fat and large $B \in R^{k \times m}$, meaning $k << n, m$, such that $A \cdot B = M$ with $min_{A,B} L(A, B) = ||M - AB||$
A, B can be obtained from eigenvectors of M^T and MM^T, but possibly huge and dense matrix. Efficiently solving this problem requires

Low complexity computation

Avoiding ever explicitly forming $M^T M$ and MM^T (too much memory)

Exploiting the structure of M (sparsity, similar columns...)

Ensuring that the solution is numerically stable.

Example: Support Vector Machines Same setting as the first example, but $y_h \in \{1, -1\}$: want to linearly separate the two sets. Which separating hyperplane to choose? Intuitively, base on the margin: more margin, more robust classification.
The distance of parallel hyperplanes (w_+, w_0) and (w_+, w_0') is $\frac{|w_0 - w_0'|}{||w_+||}$. We can always take the hyperplane in the middle and scale w: $w_+ x_h + w_0 \geq 1$ if $y_h = 1$, $w_+ x_h + w_0 \leq 1$ if $y_h = -$
The **maximum margin separating hyperplane** is the solution of $min_w \{||w_+||^2 \,|\, y_h(w_+ x_h + w_0) \geq 1)\, h = 1, \dots, m\}$, and the margin is $\frac{2}{||w_+||}$ assuming any exists.
If it doesn't exists, soft-margin SVM: $min_w ||w_+||^2 + C \cdot L(w) = \sum_{h=1}^{m}\ max(1 - y_h(w_+ x_h + w_0), 0)$, C **weight loss**
Non differentiable, reformulation $min_{w,\xi} ||w_+||^2 + C \sum_{h=1}^{m} \xi_h$

2.1 Optimization problems

Given X *any* set and $f : X \to R$ *any* function, we have an optimization problem in the form

$$(P) \quad f_* = min\{f(x) : x \in X\}$$

with X **feasible region** and f **objective function**.

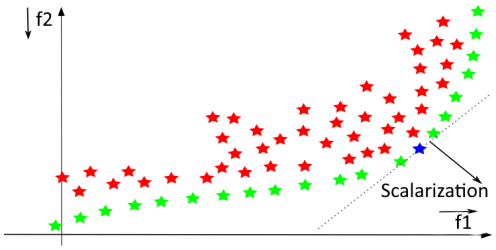
$$min\{f(x) : x \in X\} = -max\{-f(x) : x \in X\}$$

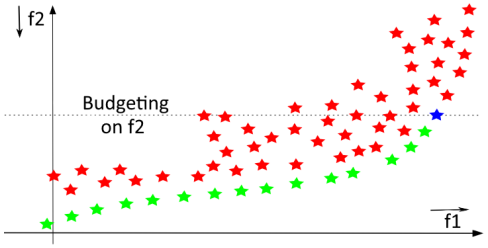
$x \in X$ is the **feasible solution**, often $X \subset F, x \in F - X$ **unfeasible solution**. We want (any) optimal solution $x_* \in X \mid \forall\, x \in X \;\; f(x_*) \leq f(x)$ and we don't care if $\exists\, x' \neq x_* \mid f(x') = f(x_*)$: **all optimal solutions are** $\equiv$. $f_* = f(x_*) = v(P)$ is the **optimal value** (which is **unique**) if x_* exists, which it may not.

Multi-objective optimization Often we have multiple objective function, and often they have incomparable values (for example, loss and regularization in machine learning). There are two practical solutions:

Scalarization, maximize one of the losses compared to the others: $min\{f_1(x) + \alpha f_2(x) \mid x \in X\}$, but which α?

Budgeting: bound the value of one of the losses $min\{f_1(x) \mid f_2(x) \leq \beta_2, x \in X\}$, but which β_2?





We will assume that this is done at modelling stage.

2.1.1 Optimization is hard

Even with single-objective, optimization is hard. It's impossible if f has no minimum in X, for example $f(x) = x$: if (P) is unbounded below, then $v(P) = -\infty$. Solving (P) is actually at least two different things:

finding x_* and **proving** x_* **it's optimal** (how?)

constructively **proving** f **unbounded below** on X (how?)

It's also impossible if $f_* > -\infty$ but $\nexists\, x_*$, for example $f(x) = e^x$ or $f(x) = \begin{cases} 1 & \text{if } x = 0 \\ |x| & \text{if } x \neq 0 \end{cases}$, but there are plenty of ϵ-approximate solutions (ϵ-optima)

$$f(x_\epsilon) \leq f_* + \epsilon \quad \forall\, \epsilon > 0$$

and on computers $x \in R$ is actually $x \in Q$ with up to 16 digits precision, so approximation errors are unavoidable anyway. Exact algebraic computation is possible but too slow, so ML is actually going the opposite way (float, half, small integers...). And anyway, finding the exact x_* is impossible in general.

Optimization need to be approximate

Absolute gap: $\{a_i = A(x_i) = f(x_i) - f_*\}$ so $A(x) = f(x) - f_* \;(\geq 0)$

Relative gap: $\{r_i = R(x_i) = \frac{f(x_i) - f_*}{|f_*|} = \frac{A(x_i)}{|f_*|}\}$ so $R(x) = \frac{f(x) - f_*}{|f_*|} = \frac{A(x)}{|f_*|} \;(\geq 0)$

The relative gap is useful because $\forall\, \alpha > 0$ we have that $(P) \equiv (P_\alpha) = min\{\alpha f(x) \mid x \in X\}$, and for the same x_* we have $v(P_\alpha) = \alpha v(P) \Rightarrow$ same $R(x)$, different $A(x)$
But in general computing the absolute/relative 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 is really hard Impossible, even, because isolated minima can be anywhere, and restricting to $x \in X = [x_-, x_+]$ with $-\infty < x_- < x_+ < +\infty$ doesn't help: still uncountable many points to try. Also f can have isolated downward spike anywhere. Even on $X = [x_-, x_+]$ the spikes can be arbitrarily narrow.

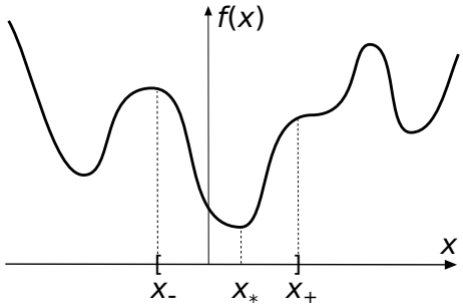
Optimization at least possible We can impose $X = [x_-, x_+]$ with $D = x_+ - x_- < \infty$, meaning with a fixed finite diameter. We can also impose that the f's spikes can't be arbitrarily narrow, so f cannot change too fast $\equiv$ f Lipschitz continuous (L-c) on X:

$$\exists\, L > 0 \mid \ |f(x) - f(y)| \leq L|x - y| \ \ \forall x, y \in X$$

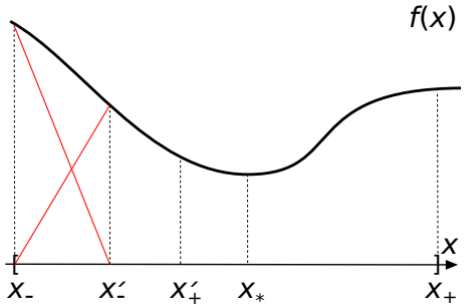
f L-c $\Rightarrow$ doesn't "jump" and one ϵ-optimum can be found with $O(\frac{LD}{\epsilon})$ evaluations by uniformly sampling X with step $\frac{2\epsilon}{L}$. There's a bad news: no algorithm can work in less that $\Omega(\frac{LD}{\epsilon})$, but it's the worst case of f (constant with one spike).
The 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.

2.1.2 Local Optimization

Even if I stumble in x_* how do I recognize it? This is the difficult thing. It's 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 $f(x_*) < f(y) \ \ \forall\, y \in X(x_*, \epsilon) - \{x_*\}$
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 $[x_*, x_+]$. If x_*, then typically $\exists\, \epsilon >= \mid f$ is (strictly) unimodal on $X(x_*, \epsilon)$ 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 A generally powerful concept is to optimize the worst-case behavior $\Rightarrow$ shrink the intervals as quickly as possible.
Each iteration dumps either $[x_-, x'_-]$ or $[x'_+, x_+]$, we don't know which so they should be of equal size $\Rightarrow$ select $r \in (\frac{1}{2}, 1)$ so that $x'_- = x_- + (1-r)D$ and $x'_+ = x_- + rD$
r larger $\Rightarrow$ faster convergence, so $r = \frac{D}{2} + \epsilon \Leftrightarrow x'_\pm = x_- + \frac{D}{2} \pm \epsilon$ but next iteration will have two entirely different x'_-, x'_+ to evaluate f on.
So we actually want to minimize function evaluations by reusing the surviving point.

$$r : 1 = (1-r) : r \Leftrightarrow r \cdot r = 1 - r \Leftrightarrow r = \frac{\sqrt{5}-1}{2} = 0.618 = \frac{1}{g}$$

with g being the golden ratio, $g = \frac{\sqrt{5}+1}{2} = 1.618 \Rightarrow g = 1 + r = 1 + \frac{1}{g}$
Theorems breed algorithms: **golden ratio search**

```
procedure x = GRS(f, xl, xr, delta)
  xl2 = xl + (1-r)(xr-xl)
  xr2 = xl + r(xr - xl)
  compute f(xl2), f(xr2)
  while (xr - xl > delta):
    if(f(xl2) > f (xr2))
      xl = xl2
      xl2 = x
      x = xr2
      xr2 = xl + r(xr - xl)
      compute f(xr2)
    else:
      xr = xr2
      xr2 = x
      x = xl2
      xl2 = xl + (1-r)(xr-xl)
      compute f(xl2)
```

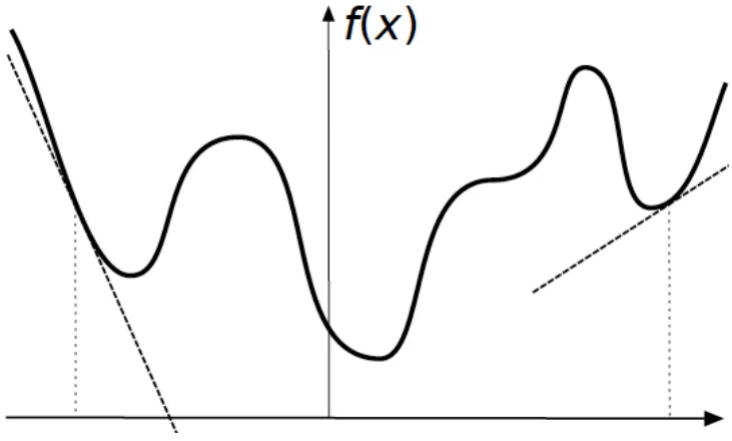
After k iterations, $x_+^k - x_-^k = Dr^k$ stops when $Dr^k \leq \delta$, so when $k = 2\log\frac{D}{\delta}$: exponentially faster, can work with small δ.
Asymptotically optimal if no other information is available. $\delta \neq \epsilon$ but f L-c $\Rightarrow A(x^k) \leq \epsilon$ when $k = 2\log\frac{LD}{\epsilon}$
First example of linear convergence $A(x^k) \leq Sr^k \leq \epsilon$ with $r < 1$, as fast as a negative exponential $\Rightarrow k \geq \frac{\log\frac{S}{\epsilon}}{\log\frac{1}{r}}$

$O(\log(\frac{1}{\epsilon}))$ good, but the constant $\to \infty$ as $r \to 1$

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:

$$L_x(y) = f'(x)(y - x) + f(x)$$

$L_x(y) \simeq f(y) \quad \forall\, y \in [x - \epsilon, x + \epsilon]$ for some small $\epsilon > 0$.
x_* local minimum $\Rightarrow f'(x_*) = 0 \Leftrightarrow$ root of $f' \Leftrightarrow$ stationary point. If $f'(x) < 0$ or $f'(x) > 0$, then x is clearly not a local minimum. Hence, $f'(x) = 0$ for the all local minima (hence in the global minimum as well) but this is true for the local maxima (hence global maximum as well), as well in the plateau and saddle points. To tell them apart, look at the second derivative f''.



In simple cases we get the answer by a closed formula. In $f = bx + c$ linear, if $b > 0$ 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 $a < 0$.
Only polynomial whose root have a closed formula (degree 3 and some degree 4), with basically no hope for most trascendental, trigonometric and mixed equations. We need an algorithm for solving non-linear equations.

Dichotomic Search f' continuous and the intermediate value theorem gives that

$$f'(x_-) < 0 \wedge f'(x_+) > 0 \Rightarrow \exists\, x \in [x_-, x_+] \mid f'(x) = 0$$

Theorems breed algorithms $\rightarrow$ **Dichotomic Search**.

```
procedure x = DS(f, xl, xr, eps)
  while (true) do:  # invariant: df(xl)<-eps, df(xr)>eps
    x = in_middle_of(xl, xr)
    compute df(x)
    if (abs(df(x)) <= eps): break
    if (df(x) < 0):
      xl = x
    else:
      xr = x
```

With `df` meaning f'
For `in_middle_of(xl, xr)` the obvious choice is `return (xl + xr)/2;`. We have linear convergence with $\gamma = 0.5 < 0.618 \Rightarrow k = 1.45 \log(\frac{LD}{\epsilon}) < 2 \log(\frac{LD}{\epsilon})$
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 until 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 on $X \Rightarrow |f'(x)| \leq L \forall\, x \in X$

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

Fastest local optimization Interpolation, for improving the dichotomic search. Choosing x "right in the middle" is the dumbest possible approach, because we know a lot about f: $f(x_-), f(x_+), f'(x_-), f'(x_+) \ldots$. So let's use that, by constructing a model of f based on known information. Much better choosing x close to x_*.
But remember that **the model is an estimate**, so never completely trust the model, but regularize, stabilize... in this case, the minimum guaranteed decrease is with $\sigma < 0.5$, and the worst case is linear convergence with $r = 1 - \sigma$, but hopefully is much faster than that when the model is "right".

2.1.3 Measuring algorithms speed

Given the sequences

$\{x_i\}$

$\{d_i = |x_i - x_*|\}$

$\{f_i = f(x_i)\}$

$\{a_i = A(x_i) = f(x_i) - f_*\}$ absolute gap

$\{r_i = R(x_i) = \frac{f(x_i)-f(x)}{|f_*|})\frac{A(x_i)}{|f_*|}\}$ relative gap

We have convergence when $\{a_i\} \to 0, \{r_i\} \to 0 \Leftarrow \{d_i\} \to 0$ (but $\not\Rightarrow$), but how rapidly? **Rate of convergence**

$$\lim_{i\to\infty}\left(\frac{f_{i+1}-f_*}{f_i-f_*}\right)^p = \lim_{i\to\infty}\left(\frac{a_{i+1}}{a_i}\right)^p = \lim_{i\to\infty}\left(\frac{r_{i+1}}{r_i}\right)^p = r$$

$p = 1$ $r = 1 \Rightarrow$ sublinear

$\frac{1}{i} \Rightarrow k \in O(\frac{1}{\epsilon})$ (bad)

$\frac{1}{i^2} \Rightarrow k \in O(\frac{1}{\sqrt{\epsilon}})$ (a bit better)

$\frac{1}{\sqrt{i}} \Rightarrow k \in O(\frac{1}{\epsilon^2})$ (horrible)

$r < 1 \Rightarrow$ linear, $r_i \to i \in O(\log(\frac{1}{\epsilon})$, good unless $r = 1$

$p \in (1, 2)$ $p = 1, r = 0 \Rightarrow$ superlinear

$p = 2$ $r > 0 \Rightarrow$ quadratic, best we can reasonably hope for
$\frac{1}{2^{2^i}} \Rightarrow i \in O(\log(\log(\frac{1}{\epsilon})))$, which is basically $O(1)$: the number of correct digits double at each iteration

Improving dichotomic search Quadratic interpolation has superlinear convergence if started "close enough".
$f \in C^3, f'(x_*) = 0 \wedge f''(x_*) \neq 0 \Rightarrow \exists\, \delta > 0 \mid x_0 \in [x_* - \delta, x_* + \delta] \Rightarrow \{x_i\} \to x_*$ with $p = \frac{1+\sqrt{5}}{2}$ exponent of superlinear convergence.
x_0 is the starting point of the algorithm.

Four conditions $\Rightarrow$ can fit a cubic polynomial and use its minima. Theoretically pays: quadratic convergence ($p = 2$) and seems to work well in practice.

Newton's method More derivatives, so same information with less points. First order model of f' at x_i

$$L_i'(x) = L_{x_i}'(x) = f'(x_i) + f''(x_i)(x - x_i) \simeq f'(x)$$

and solve $L_i'(x) = 0 \simeq f'(x) = 0 \Rightarrow x = x_i - \frac{f'(x_i)}{f''(x_i)}$

```
procedure x = NM(f, x, eps)
  while (abs(df(x)) > eps):
    x = x - (df(x)/ddf(x))
```

With `df` meaning f' and `ddf` meaning f''.
Alternatively construct a second order model

$$Q_i(x) = Q_{x^i}(x) = f(x^i) + f'(x^i)(x - x^i) + f''(x^i)\frac{(x - x^i)^2}{2}$$

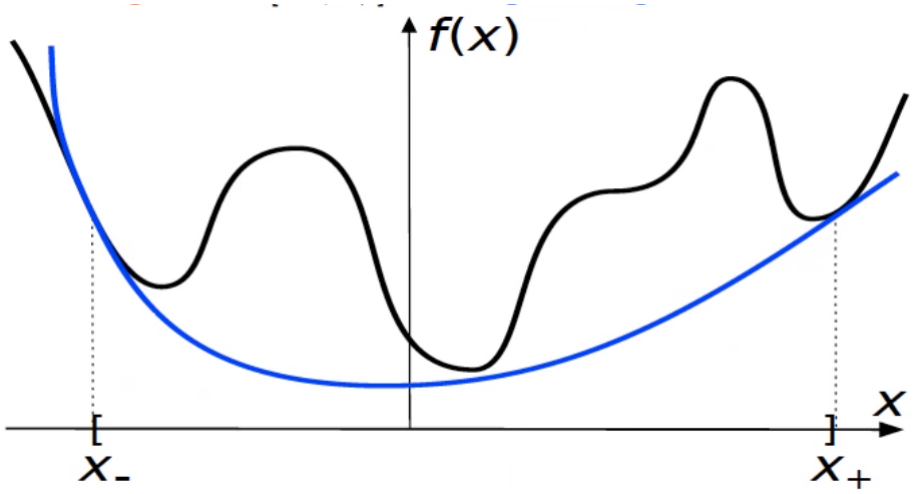
and then minimize it.

Numerically delicate: what if $f''(x) \simeq 0$? Converges (at all) only if started close enough to x_*.
If we get within a short enough distance δ, then it will converges extremely fast with $p = 2$. Mathematically, $f \in C^3, f'(x_*) = 0 \wedge f''(x_*) \neq 0 \Rightarrow \exists\, \delta > 0 \mid x_0 \in [x_* - \delta, x_* + \delta] \Rightarrow \{x_i\} \to x_*$ with $p = 2$.

2.1.4 Global optimization

Unless strong assumptions are made, we can't say much about global optimization.
The obvious one would be unimodal, but not easy to verify/construct. Workable alternative: f convex ($\Rightarrow$ unimodal).

Convexity Convex means that f' monotone non decreasing and $f'' \geq 0$. But convexity $\not\Rightarrow C^1$. Some functions are convex and a few operators preserve convexity. Many models are purposely constructed convex $\Rightarrow$ **Spatial Branch-and-Bound approach**: sift through all $X = [x_-, x_+]$ using clever guide, convex lower approximation $\underline{f}$ of nonconvex f on X.



"Easily" find local $\equiv$ global minimum $\overline{x}$ giving $\underline{f}(\overline{x}) \leq f_* \leq f(\overline{x})$. If the gap $f(\overline{x}) - \underline{f}(\overline{x})$ is too large, we partition X and iterate. If on some partition $\underline{f}(\overline{x}) \geq$ best f-value so far, then that partition is killed.
In the worst case, exponential complexity because you keep dicing and slicing X. But it is exponential in practice too. It depends on how much non-convex f is and how good of a lower approximation $\underline{f}$ is. A cleverer approach is carefully choosing the non-convexities.

2.2 Unconstrained optimization

From now on we'll use $f : R^n \to R$, i.e. $f(x_1, x_2, \ldots, x_n) = f(x)$ with $x = [x_i]_{i=1}^n = [x_1, \ldots, x_n] \in R^n$. Note that $R^n = R \times R \times \ldots \times R$, which is **exponentially larger** than R.
$I = [x_-, x_+], X = I \times I \times \ldots \times I$ hypercube (or hyperrectangle if intervals are disequal).

We need f to be L-c, and sadly no algorithm can work in less than $\Omega((\frac{LD}{\epsilon})^n)$. **Curse of dimensionality**: not really doable unless $n = 3, 5, 10$ tops. Can make to $O((\frac{LD}{\epsilon})^n)$, with multidimensional grid and small enough step (standard approach to hyperparameter optimization). If f analytic, clever B&B can give global optimum. If f black-box (typically, no derivatives), many heuristics can give good solutions, probably not optimal.

Unconstraint global optimization If f is convex, then global $\equiv$ local which is much better: most (but not all) convergence results are dimension independent and if there's dependence it is not exponential. Doesn't mean that all local algorithms are fast: speed may be low (badly linear), cost of f or derivatives computation increases with n dimension (for large n even $O(n^2)$ may be too much) and some dependency on n may be hidden in $O(\,)$ constraints. Yet, large scale optimization can be done.

Notation

Scalar product $\langle x, y \rangle = x^T y = \sum_{i=1}^n x_i y_i = x_1 y_1 + \ldots + x_n y_n$

Norm $||x|| = \sqrt{x_1^2 + \ldots + x_n^2} = \sqrt{\langle x, x \rangle}$

Distance $d(x, y) = ||x - y|| = \sqrt{(x_1 - y_1)^2 + \ldots + (x_n - y_n)^2}$

Ball with center $r \in R^n$ and radius $r > 0$ is $B(x, r) = \{y \in R^n \mid ||y - x|| \leq r\}$

Usually $f : D \to R$ with $D = dom(f)$ domain of f which may not be all R^n, but usually ok to ignore $dom(f)$ and assume $f(x) = \infty$ for $x \notin D$

Graph of f lives in R^{n+1}: $gr(f) = \{(f(x), x) \mid x \in R^n\}$

Epigraph of f lives in R^{n+1}: $epi(f) = \{(v, x) \in R^{n+1} \mid v \geq f(x)\}$

Level set at value v: $L(f, v) = \{x \in R^n \mid f(x) = v\}$

Sublevel set at value v: $S(f, v) = \{x \in R^n \mid f(x) \leq v\}$

We have that $x_* \in S(f, v) \ \ \forall\, v \geq f_*$ and $S(f, v) = \emptyset \ \ \forall\, v < f_*$

Tomography $f : R^n \to R$ with $x \in R^n$ origin and $d \in R^n$ direction. You can define $\phi_{x,d}(\alpha) = f(x + \alpha d) : R \to R$ **tomography** of f from x along d.
$\phi_{x,d}$ can always be pictured, but there are infinitely many of them: which x, d?

$||d||$ only changes the scale: $\phi_{x,\beta d}(\alpha) = \phi_{x,d}(\beta\alpha)$ so often convenient to use normalised direction ($||d|| = 1$)

Simplest case: restriction along i-th coordinate
$f_x^i(\alpha) = f(x_1, \ldots, x_{i-1}, \alpha, x_{i+1}, \ldots, x_n) = \phi_{0,u^i}(\alpha)$ with $||u^i|| = 1$

When x, d clear from context, then $\phi(\alpha)$

Simple Functions

Linear $f(x) = \langle b, x\rangle + c, b \in R^n, c \in R$
Tomography $f(x) = \langle b, x\rangle, x = 0, ||d|| = 1$: $\phi(\alpha) = \alpha\langle b, d\rangle = \alpha||b||cos(\theta)$
Plotting this gives a line, increasing because "b same direction as d", more collinear ⇒ steeper. Collinear means steepest line, less collinear means less steap. 90° angle means a flat line. Decreasing if opposite directions.
min $f(x)$ when $\nexists\, x_*$ if $b \neq 0$ ($\Rightarrow \exists\, d \mid \langle b, d\rangle \neq 0$), $\forall\, x$ if $b = 0$

Quadratic with fixed $Q \in R^{n\times n}, q \in R^n$ we have $f(x) = \frac{1}{2}x^TQx + qx$. If $q = 0$ (no linear term), then **homogeneous quadratic**.
Tomography $\phi(\alpha) = f(\alpha d) = \alpha^2(d^TQd) \Rightarrow$ sign and steepness depend on d^TQd, so we need to know about signs of d^TQd. Steeper when d along one axe, least steep when d along the other axe and intermediate steepness when "in between". Again, steeper along the opposite of one axe and least steep along the opposite of the other axe.
With $q \neq 0$ but Q nonsingular then $\lambda_i \neq 0 \ \ \forall\, i$, then $f(x) = \frac{1}{2}(x - x_*)^TQ(x - x_*)[+c]$ for $x_* = -Q^{-1}q$ and $x_* \neq 0$ center of the level sets (which shapes are determined by the eigenvalues).
$y = x - x_*, f_*(y) = y^TQy[+c]$

Directional/partial derivatives The directional derivative of $f : R^n \to R$ at $x \in R^n$ along the direction $d \in R^n$ is

$$\frac{\partial f}{\partial d}(x) = \lim_{t\to 0} \frac{f(x + td) - f(x)}{t} = \phi'_{x,d}(0)$$

How can $\phi'_{x,d}(0)$, the derivative of the (x, d)-tomography (in 0), be computed? A special case is $\frac{\partial f}{\partial d}(x)$, partial derivative of f with respect to x_i at $x \in R^n$, easy to compute by just treating x_j for $j \neq i$ as constants.
The **gradient** is the column vector of all partial derivatives

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(x) \\ \vdots \\ \frac{\partial f}{\partial x_n}(x) \end{bmatrix}$$

$$f(x) = \langle b, x\rangle \Rightarrow \nabla f(x) = b$$

$$f(x) = \frac{1}{2}x^TQx + qx \Rightarrow \nabla f(x) = Qx + q$$

f differentiable at x if $\exists$ linear function $\psi(h) = \langle b, h\rangle + f(x)$ such that

$$\lim_{||h||\to 0} \frac{|f(x + h) - \psi(h)}{||h||} = 0$$

$$\Rightarrow \psi(0) = f(x) \Rightarrow c = f(x)$$

ψ is equivalent to the first order model of f at x, the error of this equivalence vanishes faster than linearity. So f differentiable at $x \Rightarrow b = \nabla f(x)$ and

$\Rightarrow \frac{\partial f}{\partial x_i}(x)$ exists for every i (but $\Leftrightarrow$ not true)

$\Rightarrow$ first order model of f at x is $L_x(y) = \nabla f(x)(y - x)$

f differentiable $\Rightarrow$ all relevant objects in R^{n+1} and R^n are smooth. If f is non differentiable $\Rightarrow$ kinks appear and things break,

Jacobian Given a vector-valued function $f : R^n \to R^m$, $f(x) = [f_1(x), f_2(x), \ldots, f_m(x)]$, the partial derivatives are the same with extra index

$$\frac{\partial f_j}{\partial x_i}(x) = \lim_{t \to 0} \frac{f_j(x_1, \ldots, x_{i-1}, x_i + t, x_{i+1}, \ldots, x_n) - f_j(x)}{t}$$

The **Jacobian** is the matrix of all mn partial derivatives

$$Jf(X) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(x) & \ldots & \frac{\partial f_1}{\partial x_n}(x) \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1}(x) & \ldots & \frac{\partial f_m}{\partial x_n}(x) \end{bmatrix} = \begin{bmatrix} \nabla f_1(x)^T \\ \vdots \\ \nabla f_m(x)^T \end{bmatrix}$$

A $m \times n$ matrix with gradients as rows.

Hessian The $\frac{\partial f}{\partial x_i} : R^n \to R$ have partial derivatives themselves. **Second order partial derivative**, just do it twice

$$\frac{\partial^2 f}{\partial x_i \partial x_j}$$

$$\frac{\partial^2 f}{\partial x_i \partial x_i} = \frac{\partial^2 f}{\partial x_i^2}$$

So $\nabla f(x) : R^n \to R^n$ have a Jacobian and it's called **Hessian** of f at x

$$\nabla^2 f(x) = J\nabla f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2}(x) & \frac{\partial^2 f}{\partial x_2 \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_1}(x) \\ \vdots & \ddots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_n}(x) & \frac{\partial^2 f}{\partial x_2 \partial x_n}(x) & \ldots & \frac{\partial^2 f}{\partial x_n^2}(x) \end{bmatrix}$$

Requires $O(n^2)$ to store and at least $O(n^2)$ to compute, unless sparse.
$f(x) = \frac{1}{2}x^T Qx + qx \Rightarrow \nabla^2 f(x) = Q$
Second order model = first order model plus second order term (better)

$$Q_x(y) = L_x(y) + \frac{1}{2}(y - x)^T \nabla^2 f(x)(y - x)$$

Theorem $\exists\, \delta > 0 \mid \forall\, y \in B(x, \delta)$ we have that $\frac{\partial^2 f}{\partial x_j \partial x_i}(y)$ and $\frac{\partial^2 f}{\partial x_i \partial x_j}(y)$ exist and are continuous in x

$\Rightarrow \frac{\partial^2 f}{\partial x_j \partial x_i}(y) = \frac{\partial^2 f}{\partial x_i \partial x_j}(y) \Leftrightarrow \nabla^2 f$ symmetric

$\Rightarrow$ all eigenvalues of $\nabla^2 f(x)$ are real

With $f \in C^2$ we have $\nabla^2 f(x)$ continuous everywhere, so symmetric everywhere. C^2 is the best class for optimization.

2.2.1 Optimality conditions

f differentiable at x and x local minimum $\Rightarrow \nabla f(x) = 0 \equiv$ stationary point ($\not\Leftarrow$): to tell them apart we need to look at the curvature of f. If f quadratic I wold know, looking at the eigenvalues of $Q = \nabla^2 f(x)$, so we could approximate f with a quadratic function: the second order model $Q_x(y) = L_x(y) + \frac{1}{2}(y - x)^T \nabla^2 f(x)(y - x)$
$\nabla Q_x(x) = \nabla L_x(x) = \nabla f(x) \Rightarrow \nabla Q_x(x) = 0$, otherwise not minimum. Meaning that in a local minimum there cannot be directions of negative curvature, $\nabla^2 f(x) \geq 0 \Leftrightarrow x$ (global) minimum of Q_x.
Another condition necessary and almost sufficient: $f \in C^2$. So $B = u_1 \sigma_1 v^t$ from SVD of $\overline{A}$ will give me the rank-1 matrix with minimum $||\overline{A} - B||_F^2 = \sum_{i,j} (\overline{A_{i,j}} - B_{i,j})^2$. Suppose now you can only keep a 2D projection of your data. What is the best plane (i.e. minimum sum of squares of errors)? Blue crosses on a plane through the origin = linear combination of tw vectors ($\overline{A}$ = real data, B = plotted data, all cols multiples of the same 2 vectors $\Rightarrow$ rank-2
$\min||\overline{A} - B|| =$