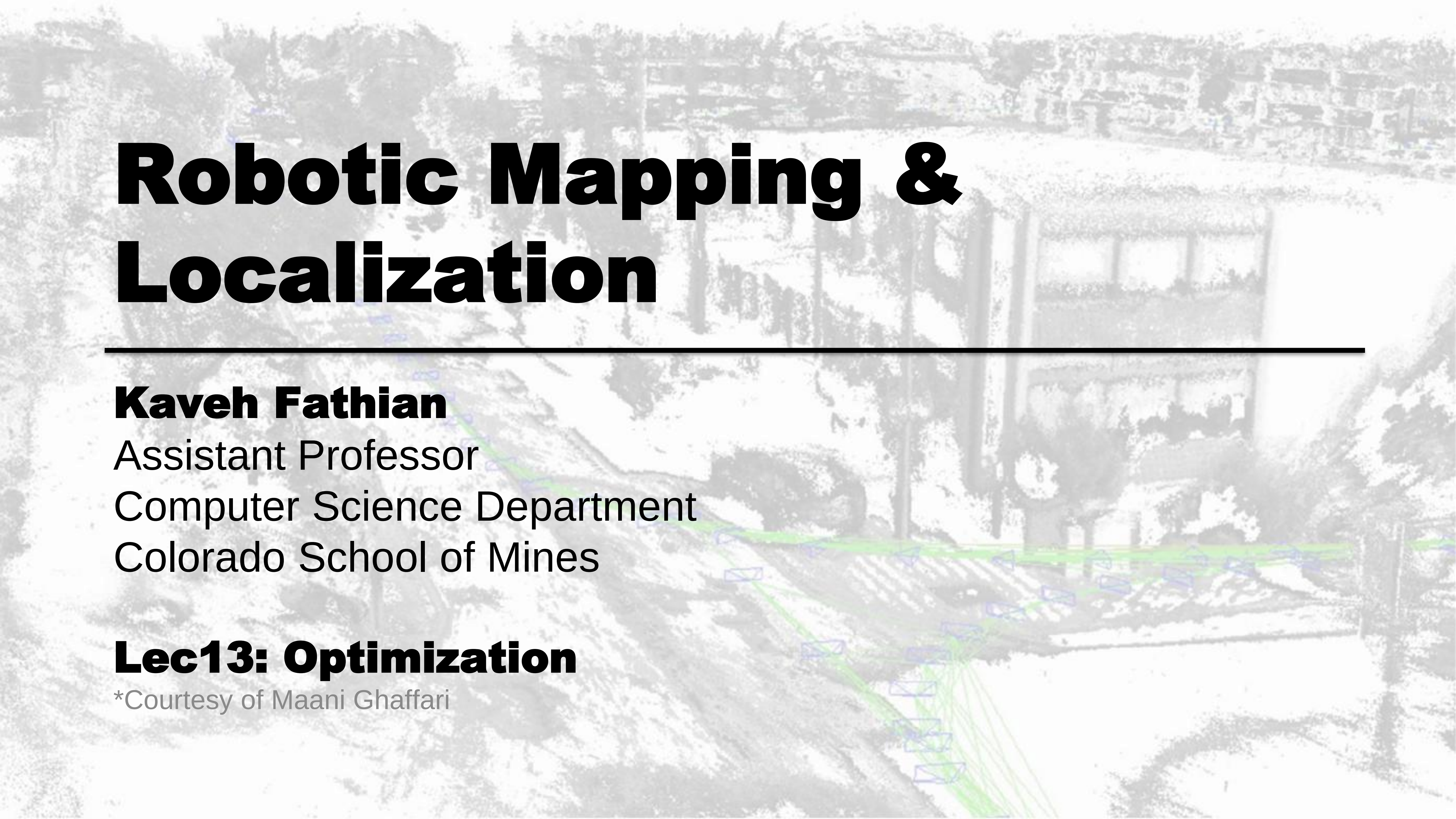


Robotic Mapping & Localization

Kaveh Fathian
Assistant Professor
Computer Science Department
Colorado School of Mines

Lec13: Optimization

*Courtesy of Maani Ghaffari

IMPORTANT

Linear Systems, Solutions

A system of linear equations can have

- No solution;
- Unique solution (one and only one solution);
- Infinite number of solutions.

Linear Systems of Equations: Case I

$$\begin{aligned} x + y &= 4 \\ 2x - y &= -1 \end{aligned}$$

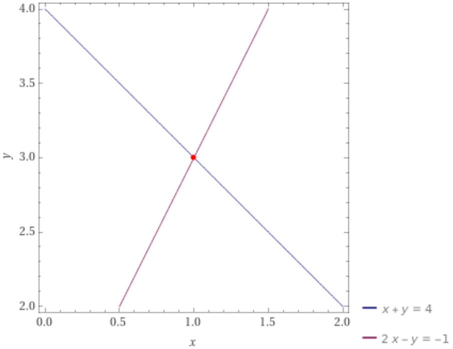


Figure: Graphical solution. Unique solution ⇔ intersecting lines!

Linear Systems of Equations: Case II

$$\begin{aligned} x - y &= 1 \\ 2x - 2y &= -1 \end{aligned}$$

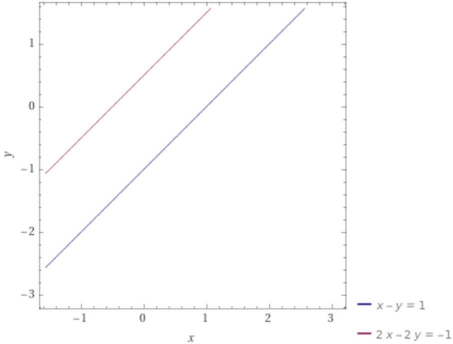


Figure: No solution ⇔ parallel lines!

Linear Systems of Equations: Case III

$$\begin{aligned} x - y &= 1 \\ 2x - 2y &= 2 \end{aligned}$$

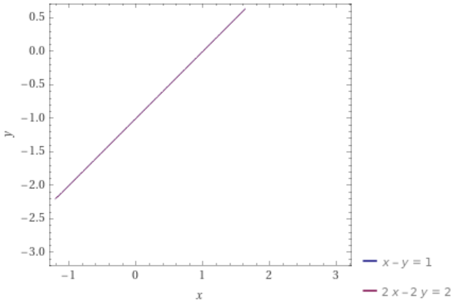


Figure: Infinite number of solutions ⇔ coincident lines!

IMPORTANT

Generalization to n Unknowns

We now write a general system of linear equations as

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ \vdots \qquad &= \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n \end{aligned}$$

IMPORTANT

Generalization to n Unknowns

We can write this system as:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix},$$

where:

$$A := \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, x := \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, b := \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}.$$

IMPORTANT

Lower Triangular Matrices

$$A = \begin{bmatrix} 3 & \mathbf{0} & \mathbf{0} \\ 2 & -1 & \mathbf{0} \\ 1 & -2 & 3 \end{bmatrix}$$

- All terms above the diagonal of the matrix A are zero.
- More precisely, the condition is $a_{ij} = 0$ for all $j > i$.
- Such matrices are called *lower-triangular*.

Lower Triangular Systems and Forward Substitution

We will solve this example using a method called *forward substitution*.

$$\begin{aligned} 3x_1 &= 6 \\ 2x_1 - x_2 &= -2 \\ x_1 - 2x_2 + 3x_3 &= 2 \end{aligned} \iff \underbrace{\begin{bmatrix} 3 & 0 & 0 \\ 2 & -1 & 0 \\ 1 & -2 & 3 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} 6 \\ -2 \\ 2 \end{bmatrix}}_{b}.$$

Upper Triangular Matrices

$$A = \begin{bmatrix} 1 & 3 & 2 \\ \mathbf{0} & 2 & 1 \\ \mathbf{0} & \mathbf{0} & 3 \end{bmatrix}$$

- All terms below the diagonal of the matrix A are zero.

- More precisely, the condition is $a_{ij} = 0$ for $i > j$.

- Such matrices are called *upper-triangular*.

Back Substitution

We solve the upper triangular systems using a method called *back substitution*.

$$\begin{aligned} x_1 + 3x_2 + 2x_3 &= 6 \\ 2x_2 + x_3 &= -2 \\ 3x_3 &= 4, \end{aligned} \quad \Longleftrightarrow \underbrace{\begin{bmatrix} 1 & 3 & 2 \\ 0 & 2 & 1 \\ 0 & 0 & 3 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} 6 \\ -2 \\ 4 \end{bmatrix}}_{b}.$$

IMPORTANT

LU Factorization for Solving Linear Equations

- We wish to solve the system of linear equations $Ax = b$.
- If we can factor $A = L \cdot U$, where U is upper triangular and L is lower triangular. Then

$$L \cdot Ux = b.$$

- Define $U \cdot x =: y$, then

$$\begin{aligned} Ly &= b \\ Ux &= y. \end{aligned}$$

- We first solve for y via forward substitution. Given y, we solve for x via back substitution.

Solving $Ax = b$ via LU with Row Permutation

Solving $Ax = b$ when A is square and $P \cdot A = L \cdot U$.

- $Ax = b \iff P \cdot Ax = P \cdot b \iff L \cdot Ux = P \cdot b$.
- Define $Ux =: y$, then

$$\begin{aligned} Ly &= P \cdot b \\ Ux &= y. \end{aligned}$$

- We first solve for y via forward substitution. Given y, we solve for x via back substitution.

Gram-Schmidt Process

Suppose that that the set of vectors $\{u_1, u_2, \ldots, u_m\}$ is linearly independent and you generate a new set of vectors by

$$
\begin{aligned}
v_1 &= u_1, \\
v_2 &= u_2 - \left(\frac{u_2 \bullet v_1}{v_1 \bullet v_1}\right) v_1, \\
v_3 &= u_3 - \left(\frac{u_3 \bullet v_1}{v_1 \bullet v_1}\right) v_1 - \left(\frac{u_3 \bullet v_2}{v_2 \bullet v_2}\right) v_2, \\
&\vdots \\
v_k &= u_k - \sum_{i=1}^{k-1} \left(\frac{u_k \bullet v_i}{v_i \bullet v_i}\right) v_i, \quad \text{(General Step)}
\end{aligned}
$$

Gram-Schmidt Process

Then the set of vectors $\{v_1, v_2, \ldots, v_m\}$ is

- orthogonal, meaning, $i \neq j \implies v_i \bullet v_j = 0$
- span preserving, meaning that, for all $1 \leq k \leq m$,
$$\operatorname{span}\{v_1, v_2, \ldots, v_k\} = \operatorname{span}\{u_1, u_2, \ldots, u_k\},$$
- and linearly independent.

Gram-Schmidt Process

Remark

The unit vectors $\{e_1 = \frac{v_1}{\|v_1\|}, e_2 = \frac{v_2}{\|v_2\|}, \ldots, e_m = \frac{v_m}{\|v_m\|}\}$ *form an orthonormal set.*

QR Factorization

Suppose that A is an $n \times m$ matrix with linearly independent columns.

Fact

Then there exists an $n \times m$ matrix Q with orthonormal columns and an upper triangular, $m \times m$, invertible matrix R such that $A = Q \cdot R$.

Least Squares Solutions to $A_{n\times m}\cdot x_{m\times 1}=b_{n\times 1}$

- Assume $A^{\mathsf{T}}A$ is invertible, i.e., the columns of A are linearly independent.

- Then there is a unique vector $x^*\in\mathbb{R}^m$ achieving $\min_{x\in\mathbb{R}^m}||Ax-b||^2$ and it satisfies the equation (called *the normal equations*)
$$\left(A^{\mathsf{T}}A\right)x^*=A^{\mathsf{T}}b.$$

-
$$x^*=(A^{\mathsf{T}}A)^{-1}A^{\mathsf{T}}b\iff x^*=\operatorname*{arg\,min}_{x\in\mathbb{R}^m}\|Ax-b\|^2\iff\left(A^{\mathsf{T}}A\right)x^*=A^{\mathsf{T}}b.$$

Least Squares via the QR Factorization

Whenever the columns of A are linearly independent, a least squared error solution to $Ax = b$ is computed as

- factor $A =: Q \cdot R$,
- compute $\overline{b} := Q^{\mathsf{T}} b$, and then
- solve $Rx = \overline{b}$ via back substitution.

Relation of Null Space and Range to Solutions of Linear Equations

Suppose that A is $n \times m$. Here are the key relations between solutions of $Ax = b$ and the null space and range of A.

1. $Ax = b$ has a solution if, and only if, $b \in \mathrm{range}(A)$.

Relation of Null Space and Range to Solutions of Linear Equations

Suppose that A is $n \times m$. Here are the key relations between solutions of $Ax = b$ and the null space and range of A.

2 If $Ax = b$ has a solution, then it is unique if, and only if, $\mathrm{null}(A) = \{0_{m \times 1}\}$.

Relation of Null Space and Range to Solutions of Linear Equations

Suppose that A is $n \times m$. Here are the key relations between solutions of $Ax = b$ and the null space and range of A.

3 Suppose that $\tilde{x}$ is a solution of $Ax = b$, so that $A\tilde{x} = b$. Then the set of all solutions is

$$\begin{aligned}\{x \in \mathbb{R}^m \mid Ax = b\} &= \tilde{x} + \mathrm{null}(A) \\ &:= \{\hat{x} \in \mathbb{R}^m \mid \hat{x} = \tilde{x} + \eta, \eta \in \mathrm{null}(A)\}.\end{aligned}$$

Relation of Null Space and Range to Solutions of Linear Equations

Suppose that A is $n \times m$. Here are the key relations between solutions of $Ax = b$ and the null space and range of A.

4 $Ax = b$ has a unique solution if, and only if $b \in \mathrm{range}(A)$ and $\mathrm{null}(A) = \{0_{m\times 1}\}$.

Relation of Null Space and Range to Solutions of Linear Equations

Suppose that A is $n \times m$. Here are the key relations between solutions of $Ax = b$ and the null space and range of A.

5 When $b = 0_{n\times 1}$, then it is always true that $b \in \mathrm{range}(A)$. Hence we deduce that $Ax = 0_{n\times 1}$ has a unique solution if, and only if, $\mathrm{null}(A) = \{0_{m\times 1}\}$.

Root of an Equation

- We will limit our notion of a solution to the set of real numbers or real vectors.
- For example, $x^2 + 1 = 0$, has no real solutions because its discriminant is $\Delta = b^2 - 4ac = -4 < 0$.
- Nevertheless, many interesting problems in Engineering and Science can be formulated and solved in terms of "real solutions" to systems of equations.

Root of an Equation

- Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function. Then $f(x) = 0$ defines an equation.

- A solution to the equation is also called a *root* that is $x^* \in \mathbb{R}^n$ is a root of $f(x) = 0$ if

$$f(x^*) = 0.$$

- Just as with quadratic equations, it is possible to have multiple real solutions or no real solutions.

Root of an Equation

- What about $f(x) = \pi$?

- Define a new function, $\bar{f}(x) := f(x) - \pi$, then

$$\bar{f}(x^*) = 0 \iff f(x^*) - \pi = 0 \iff f(x^*) = \pi.$$

- x^* is a root of our new function $\bar{f}(x)$.

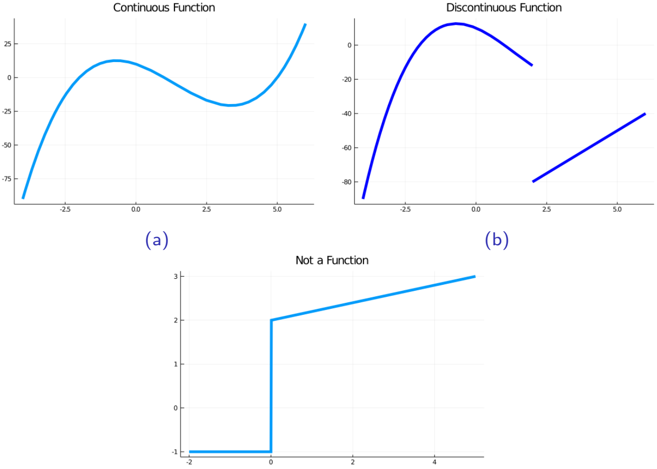
IMPORTANT

Continuous Functions

- Informally, a function $f : \mathbb{R} \to \mathbb{R}$ is *continuous* if you can draw the graph of $y = f(x)$ on a sheet of paper without lifting your pencil (from the paper).

- Also, a function is valid, if for a given $x \in \mathbb{R}$, there can be only one value of $y \in \mathbb{R}$ such that $y = f(x)$.

Continuous Functions



(a)

(b)

(c)

Intermediate Value Theorem

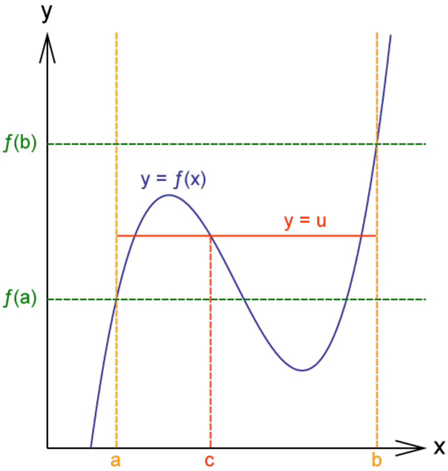
Theorem

Assume that f is a continuous real valued function and you know two real numbers $a < b$ such that $f(a) \cdot f(b) < 0$. Then there exists a real number c such that

- $a < c < b$ *(c is between a and b), and*
- $f(c) = 0$ *(c is a root).*

The values a and b are said to bracket the root, c.

Intermediate Value Theorem



Essence Newton’s Method

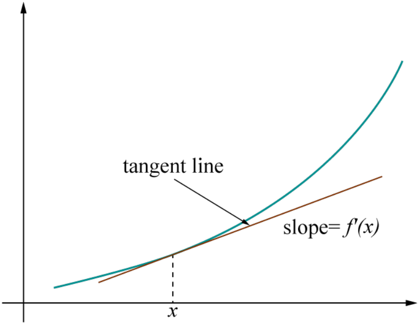
- Recall Newton’s method is a root-finding algorithm which produces successively better approximations to the roots (or zeroes) of a real-valued function.

- The idea is to start with an initial guess (reasonably close to the true root), then to *approximate the function by its tangent line*, and finally to compute the x-intercept of this tangent line by elementary algebra.

Essence Newton's Method

Core concept: approximate the nonlinear function by its tangent line at the current operating point.

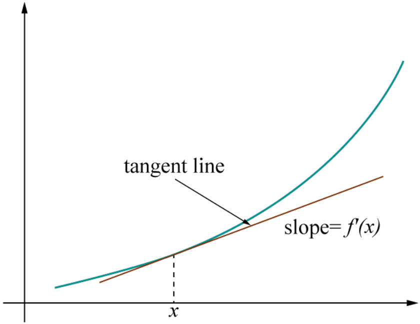
Q. What is the best linear approximation of a nonlinear function?



Linear Approximation at a Point

- The linear function $y(x)$ that passes through the point (x_0,y_0) with slope a can be written as

$$y(x) = y_0 + a\,(x - x_0)\,.$$



IMPORTANT

Linear Approximation at a Point

- We define the *linear approximation of a function at a point* x_0 by taking $y_0 := f(x_0)$ and $a := \frac{df(x_0)}{dx} = f'(x_0)$.

- This gives us

$$\boxed{f(x) \approx f(x_0) + \frac{df(x_0)}{dx}(x - x_0)\,.}$$

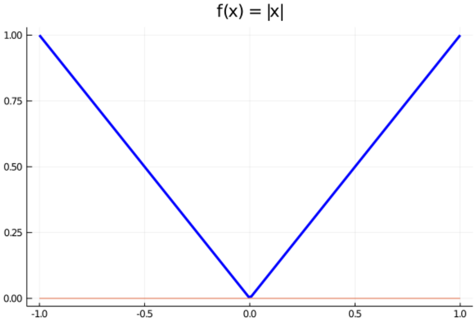
Numerical Approximations of a Derivative

For a smooth (continuous and differentiable) function and sufficiently small h, the following *finite difference* approximations of the derivatives are possible.

- Forward difference $\frac{df(x_0)}{dx} \approx \frac{f(x_0+h)-f(x_0)}{h}$.
- Backward difference $\frac{df(x_0)}{dx} \approx \frac{f(x_0)-f(x_0-h)}{h}$.
- Symmetric or central difference $\frac{df(x_0)}{dx} \approx \frac{f(x_0+h)-f(x_0-h)}{2h}$.

Example

Explain why the function $f(x) = |x|$ is not differentiable at $x_0 = 0$.



Example

We compute

- forward difference:

$$\frac{df(0)}{dx} \approx \frac{f(0+h)-f(0)}{h} = \frac{|h|-0}{h} = \frac{h}{h} = \boxed{+1},$$

- backward difference:

$$\frac{df(0)}{dx} \approx \frac{f(0)-f(0-h)}{h} = \frac{0-|-h|}{h} = \frac{-h}{h} = \boxed{-1},$$

- and central difference:

$$\frac{df(0)}{dx} \approx \frac{f(0+h)-f(0-h)}{2h} = \frac{|h|-|-h|}{2h} = \frac{h-h}{2h} = \boxed{0}.$$

Example

Remark

These three methods giving very different approximations to the "slope" at the origin is a strong hint that the function is not differentiable at the origin.
By following different paths as we approach x_0, approaching x_0 from the left versus the right for example, gives different answers for the "slope" of the function at x_0. In Calculus, you'll learn that this means the function is not differentiable at x_0.

IMPORTANT

Linear Approximation of $f : \mathbb{R}^m \to \mathbb{R}^n$

Consider a function $f : \mathbb{R}^m \to \mathbb{R}^n$. We seek a means to build a linear approximation of the function near a given point $x_0 \in \mathbb{R}^m$.

- When $m = n = 1$, we can approximate a function by

$$f(x) \approx f(x_0) + \frac{df(x_0)}{dx}(x - x_0) =: f(x_0) + a(x - x_0).$$

- For the general case of $f : \mathbb{R}^m \to \mathbb{R}^n$, can we find an $n \times m$ matrix A such that

$$f(x) \approx f(x_0) + A(x - x_0).$$

IMPORTANT

Linear Approximation of $f : \mathbb{R}^m \to \mathbb{R}^n$

- A function $f : \mathbb{R}^m \to \mathbb{R}^n$ is called a vector-valued function.
- We compute its best linear approximation via the generalization of the derivative $A := \frac{\partial f(x)}{\partial x}\Big|_{x=x_0}$ such that

$$f(x) \approx f(x_0) + A(x - x_0).$$

IMPORTANT

The Gradient

- For the special case of $n = 1$, $f : \mathbb{R}^m \rightarrow \mathbb{R}$, the matrix A is a *column* vector and called the *gradient* of f.

- The following notations are all common

$$A^T = \nabla f = \mathrm{grad}\ f.$$

- The symbol ∇ (nabla or del) is a common notation to refer to the gradient of f.

IMPORTANT

The Jacobian

- For the general case of $f : \mathbb{R}^m \to \mathbb{R}^n$, A is an $n \times m$ matrix and called the *Jacobian* of f, i.e.,

$$A_{n\times m} = \begin{bmatrix} a_1^{\text{col}} & \dots & a_m^{\text{col}} \end{bmatrix} = \frac{\partial f(x)}{\partial x}.$$

- Each column of the Jacobian $a_i^{\text{col}} = \frac{\partial f(x)}{\partial x_i} \in \mathbb{R}^n$ shows the rate of change of f along e_i.

Example

For the function

$$f(x_1,x_2,x_3) := \left[\begin{array}{c} x_1x_2x_3 \\ \log(2+\cos(x_1)) + x_2^{x_1} \\ \frac{x_1x_3}{1+x_2^2} \end{array}\right],$$

compute its Jacobian at the point

$$x_0 = \left[\begin{array}{c} \pi \\ 1.0 \\ 2.0 \end{array}\right]$$

and evaluate the “accuracy” of its linear approximation.

Example

Using $h = 0.001$ and central differences we get

$$a_1^{\text{col}} = \frac{\partial f(x_0)}{\partial x_1} = \begin{bmatrix} 2.0 \\ 0.0 \\ 1.0 \end{bmatrix},$$

$$a_2^{\text{col}} = \frac{\partial f(x_0)}{\partial x_2} = \begin{bmatrix} 6.2832 \\ 3.1416 \\ -3.1416 \end{bmatrix},$$

$$a_3^{\text{col}} = \frac{\partial f(x_0)}{\partial x_3} = \begin{bmatrix} 3.1416 \\ 0.0000 \\ 1.5708 \end{bmatrix}.$$

Example

The Jacobian at x_0 is

$$A := \frac{\partial f(x_0)}{\partial x} = \begin{bmatrix} 2.0000 & 6.2832 & 3.1416 \\ 0.0000 & 3.1416 & 0.0000 \\ 1.0000 & -3.1416 & 1.5708 \end{bmatrix},$$

and the linear approximation is

$$f(x) \approx f(x_0) + A(x - x_0) =$$

$$\begin{bmatrix} 6.2832 \\ 1.0000 \\ 3.1416 \end{bmatrix} + \begin{bmatrix} 2.0000 & 6.2832 & 3.1416 \\ 0.0000 & 3.1416 & 0.0000 \\ 1.0000 & -3.1416 & 1.5708 \end{bmatrix} \begin{bmatrix} x_1 - \pi \\ x_2 - 1.0 \\ x_3 - 2.0 \end{bmatrix}.$$

Newton-Raphson for Vector Functions

- We consider functions $f : \mathbb{R}^n \to \mathbb{R}^n$ and seek a root $f(x_0) = 0$.

- Note that the domain and range are both $\mathbb{R}^n$ and thus this is the nonlinear equivalent of solving a square linear equation $Ax - b = 0$.

- We recall that $\det(A) \neq 0$ was our magic condition for the existence and uniqueness of solutions to $Ax - b = 0$.

Newton-Raphson for Vector Functions

- Let x_k be our current approximation of a root of the function f.

- We write the linear approximation of f about x_k as

$$f(x) \approx f(x_k) + A(x - x_k), \quad A = \frac{\partial f(x_k)}{\partial x}.$$

Newton-Raphson for Vector Functions

- We want to chose x_{k+1} so that $f(x_{k+1}) = 0$.

- $f(x_{k+1}) = f(x_k) + A(x_{k+1} - x_k) = 0$.

- If $\det(A) \neq 0$, we could naively solve for x_{k+1}, giving us

$$\boxed{x_{k+1} = x_k - A^{-1} f(x_k).}$$

Newton-Raphson Algorithm

Remark

Explicitly inverting the Jacobian matrix A is not desired for algorithmic implementation.
We wish to find x_{k+1} rather than A^{-1}.

Newton-Raphson Algorithm

Let's break $x_{k+1} = x_k - A^{-1} f(x_k)$ for finding x_{k+1} into two steps.

- Define $\Delta x_k := x_{k+1} - x_k$. Then $x_{k+1} = x_k + \Delta x_k$.
- $f(x_{k+1}) = 0 \implies A\Delta x_k = -f(x_k)$.

IMPORTANT

Newton-Raphson Algorithm

To summarize:

1. Start with an initial guess x_0 $(k = 0)$.
2. Solve the linear system $A\Delta x_k = -f(x_k)$.
3. Update the estimated root $x_{k+1} = x_k + \Delta x_k$.
4. Repeat (go back to 2) until convergence.

Damped Newton-Raphson Algorithm

Remark

In practice, if we take a full step using Δx_k, the algorithm might not converged. A solution is to use a step size to control how large each step should be

$$x_{k+1} = x_k + \epsilon \Delta x_k.$$

The step size $\epsilon > 0$ (α is a common notation too) can be fixed or found at each iteration using a method (often called "line search").

Example

Find a root of $F : \mathbb{R}^4 \to \mathbb{R}^4$ near $x_0 = \left[\begin{array}{cccc} -2.0 & 3.0 & \pi & -1.0 \end{array}\right]^\mathsf{T}$ for

$$F(x) = \left[\begin{array}{c} x_1 + 2x_2 - x_1(x_1 + 4x_2) - x_2(4x_1 + 10x_2) + 3 \\ 3x_1 + 4x_2 - x_1(x_1 + 4x_2) - x_2(4x_1 + 10x_2) + 4 \\ 0.5\cos(x_1) + x_3 - (\sin(x_3))^7 \\ -2(x_2)^2\sin(x_1) + (x_4)^3 \end{array}\right].$$

Example

We used a symmetric difference approximation for the derivatives, with $h = 0.1$. Below are the first five results from the algorithm:

$$x_k = \left[\begin{array}{rrrrrr} k=0 & k=1 & k=2 & k=3 & k=4 & k=5 \\ -2.0000 & -3.0435 & -2.4233 & -2.2702 & -2.2596 & -2.2596 \\ 3.0000 & 2.5435 & 1.9233 & 1.7702 & 1.7596 & 1.7596 \\ 3.1416 & 0.6817 & 0.4104 & 0.3251 & 0.3181 & 0.3181 \\ -1.0000 & -1.8580 & -2.0710 & -1.7652 & -1.6884 & -1.6846 \end{array}\right]$$

and

$$f(x_k) = \left[\begin{array}{rrrrrr} k=0 & k=1 & k=2 & k=3 & k=4 & k=5 \\ -39.0000 & -6.9839 & -1.1539 & -0.0703 & -0.0003 & -0.0000 \\ -36.0000 & -6.9839 & -1.1539 & -0.0703 & -0.0003 & -0.0000 \\ 2.9335 & 0.1447 & 0.0323 & 0.0028 & 0.0000 & -0.0000 \\ 15.3674 & -5.1471 & -4.0134 & -0.7044 & -0.0321 & -0.0001 \end{array}\right].$$

IMPORTANT

Norm Ball

We define a norm ball in $\mathbb{R}^m$ as

$$\mathcal{B}(x_c, r) := \{x \in \mathbb{R}^m : \|x - x_c\| \leq r\}.$$

Basic Terminology

- Objective function $f : \mathbb{R}^m \to \mathbb{R}$ and decision variable $x \in \mathbb{R}^m$

$$\underset{x \in \mathbb{R}^m}{\text{minimize}} \ f(x), \qquad x^* = \underset{x \in \mathbb{R}^m}{\arg\min} \, f(x)$$

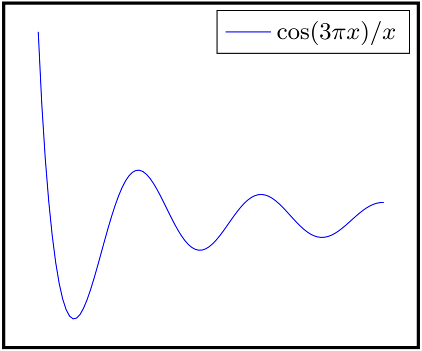
- Global minimum

$$f(x^\star) \leq f(x) \qquad \underbrace{\text{for all } x \in \mathbb{R}^m}_{\text{global}}$$

- Local minimum

$$f(x^*) \leq f(x) \qquad \underbrace{\text{for all } x \in \mathcal{B}_{r>0}(x^*)}_{\text{local}}$$

Example

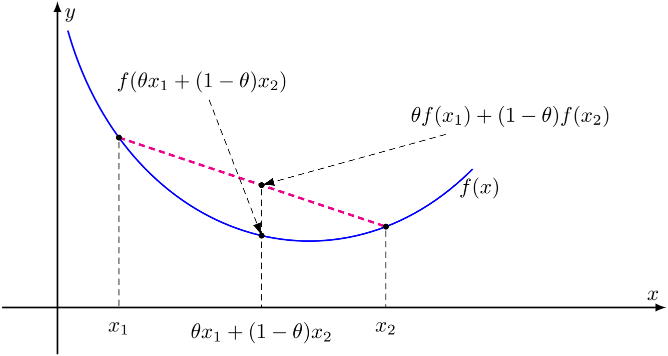


Structure: Convexity

$f : \mathbb{R}^m \to \mathbb{R}$ $(\mathrm{dom} f = \mathbb{R}^m)$ is convex iff:

1 For all $x_1, x_2 \in \mathbb{R}^m$ and all $\theta \in [0,1]$:

$$f(\theta x_1 + (1-\theta)x_2) \leq \theta f(x_1) + (1-\theta) f(x_2)$$



y
$f(\theta x_1 + (1-\theta)x_2)$
$\theta f(x_1) + (1-\theta) f(x_2)$
$f(x)$
x
x_1
$\theta x_1 + (1-\theta)x_2$
x_2

Structure: Convexity

$f : \mathbb{R}^m \to \mathbb{R}$ $(\mathrm{dom} f = \mathbb{R}^m)$ is convex iff:

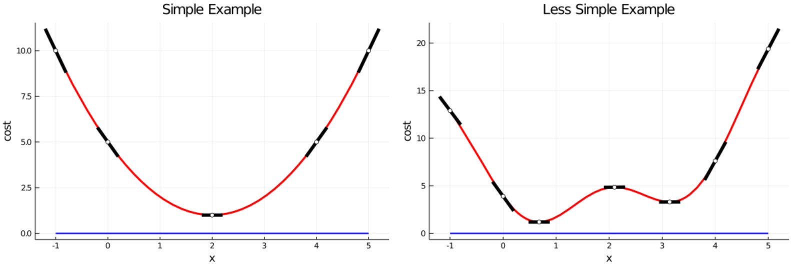
1. For all $x_1, x_2 \in \mathbb{R}^m$ and all $\theta \in [0,1]$:

$$f(\theta x_1 + (1-\theta)x_2) \leq \theta f(x_1) + (1-\theta) f(x_2)$$

2. First-order condition: For all $x, x_0 \in \mathbb{R}^m$:

$$f(x) \geq f(x_0) + \nabla f(x_0)(x - x_0)$$

The Derivatives of the Objective Functions



(a)

(b)

Recognizing Local Minima

Fact

First-order necessary condition for x to be a local extremum of f is

$$\nabla f(x) = 0.$$

IMPORTANT

Example: Linear Least Squares

Objective function: $f(x) = \frac{1}{2}\|Ax - b\|^2$

- Gradient: $\nabla f(x) = A^\mathsf{T} Ax - A^\mathsf{T} b$,

- $\nabla f(x^\star) = 0 \Rightarrow A^\mathsf{T} Ax^\star = A^\mathsf{T} b$ (Normal Equations).

Assumption

- $A \in \mathbb{R}^{n\times m}$ *and* $b \in \mathbb{R}^n$
- $n \geq m \Leftrightarrow A$ *is a tall matrix*
- $\mathrm{rank}(A) = m$ *(i.e., columns of* A *are linearly independent)*

IMPORTANT

Descent Direction

Claim

The vector $\Delta x_k \in \mathbb{R}^m$ is a descent direction, then

$$\langle \nabla f(x_k), \Delta x_k \rangle < 0 \quad \Longleftrightarrow \quad \Delta x_k \text{ is a descent direction}$$

Proof.

Using the linear approximation of f, we have

$$\begin{aligned} f(x_{k+1}) &\approx f(x_k) + \frac{df(x_k)}{dx}(x_{k+1} - x_k) \\ f(x_{k+1}) - f(x_k) &\approx \nabla f(x_k) \Delta x_k \\ \Delta f(x_k) := f(x_{k+1}) - f(x_k) &\approx \langle \nabla f(x_k), \Delta x_k \rangle \\ \Delta f(x_k) < 0 &\iff \langle \nabla f(x_k), \Delta x_k \rangle < 0. \end{aligned}$$

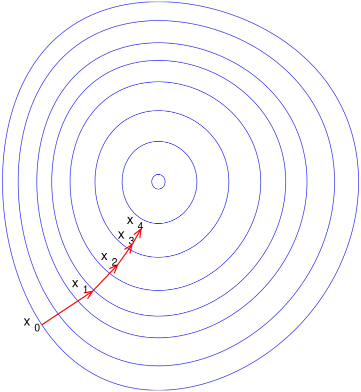
□

Descent Direction

Q. What is a “good” or natural direction (Δx_k) to follow at any point?

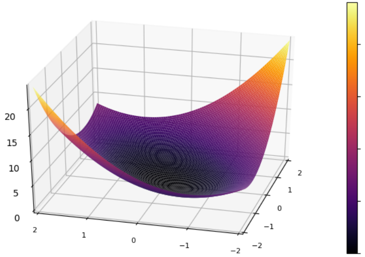
Descent Direction

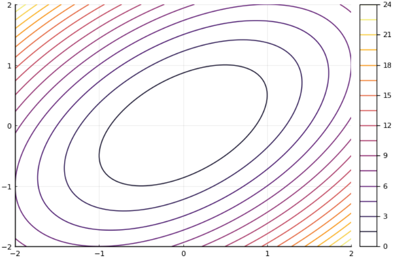
It turns out the gradient shows the fastest ascent direction; hence, the negative of the gradient is the fastest descent direction.



Descent Direction

To see why we need to take a look at the contour plot of the objective function. The curves in the right figure show the function's level sets (the function is constant along each curve).

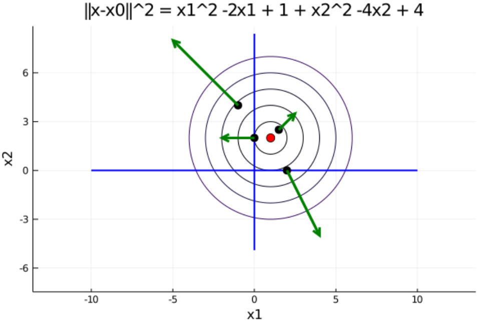




But if the function is constant along a curve, then $\Delta f(x_k) = 0$. Hence, $\boxed{\Delta f(x_k) = \langle \nabla f(x_k), \Delta x_k \rangle = 0.}$

Remark

The only explanation, if $\nabla f(x_k) \neq 0$, is that the gradient is orthogonal to any Δx_k along the curves.



Fastest Ascent and Descent Directions

Corollary

We can verify the followings, by setting $\Delta x_k = \pm \nabla f(x_k)$.

1. $\langle \nabla f(x_k), \Delta x_k \rangle = \langle \nabla f(x_k), \nabla f(x_k) \rangle = \|\nabla f(x_k)\|^2 > 0$.
   Hence, $\Delta x_k = \nabla f(x_k)$ is the fastest ascent direction.

2. $\langle \nabla f(x_k), \Delta x_k \rangle = \langle \nabla f(x_k), -\nabla f(x_k) \rangle = -\|\nabla f(x_k)\|^2 < 0$.
   Hence, $\Delta x_k = -\nabla f(x_k)$ is the fastest descent direction.

Gradient Descent

We use a step size $\alpha > 0$ to control each update size.

1. Start with an initial guess x_0 $(k = 0)$.
2. Evaluate $\nabla f(x_k)$. If $\|\nabla f(x_k)\| = 0$, then the algorithm is converged.
3. Update the decision variable via $x_{k+1} = x_k - \alpha \nabla f(x_k)$.
4. Repeat (go back to 2) until convergence.

Optimization as a Root Finding Problem: the Hessian

- Objective function $f : \mathbb{R}^m \to \mathbb{R}$.
- First-order *necessary* condition for x to be a local extremum of f is $\nabla f(x) = 0$.
- Therefore, our locally minimizing solutions are roots of the derivative (gradient) of the objective function.
- We can think of the gradient as a map $\nabla f : \mathbb{R}^m \to \mathbb{R}^m$ (column vector).

Newton-Raphson for Root Finding

- We consider $\nabla f : \mathbb{R}^m \to \mathbb{R}^m$ and seek a root $\nabla f(x_0) = 0$.

- Note that the domain and range are both $\mathbb{R}^m$ and thus this is the nonlinear equivalent of solving a square linear equation $Ax - b = 0$.

- We recall that $\det(A) \neq 0$ was our magic condition for the existence and uniqueness of solutions to $Ax - b = 0$.

Newton-Raphson for Root Finding

- Let x_k be our current approximation of a root of the function ∇f.

- We write the linear approximation of ∇f about x_k as

$$\nabla f(x) \approx \nabla f(x_k) + H_k(x - x_k),\ H_k := \frac{\partial}{\partial x}\nabla f(x_k) = \nabla^2 f(x_k).$$

IMPORTANT

Hessian

- $H_k := \frac{\partial}{\partial x}\nabla f(x_k) = \nabla^2 f(x_k)$, called *Hessian*, is the Jacobian of ∇f.

- Or when we use the row vector convention, the Hessian of a function $f : \mathbb{R}^m \to \mathbb{R}$ is the Jacobian of the transpose of the gradient of the function

$$\nabla^2 f(x) := \frac{\partial}{\partial x}\nabla f(x)^\mathsf{T}.$$

IMPORTANT

Hessian

- The Hessian is a symmetric square matrix of second-order partial derivatives of a scalar-valued function $f : \mathbb{R}^m \to \mathbb{R}$.

$$\nabla f(x)^{\mathsf{T}} := \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} \in \mathbb{R}^m$$

$$H(x) := \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1\,\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1\,\partial x_n} \\ \frac{\partial^2 f}{\partial x_2\,\partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2\,\partial x_n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial^2 f}{\partial x_n\,\partial x_1} & \frac{\partial^2 f}{\partial x_n\,\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \in \mathbb{R}^{m\times m}$$

Newton-Raphson for Root Finding

- We want to chose x_{k+1} so that $\nabla f(x_{k+1}) = 0$.
- $\nabla f(x_{k+1}) = \nabla f(x_k) + H_k(x_{k+1} - x_k) = 0.$
- Define $\Delta x_k := x_{k+1} - x_k$. Then $x_{k+1} = x_k + \Delta x_k$.
- $\nabla f(x_{k+1}) = 0 \implies H_k \Delta x_k = -\nabla f(x_k)$.

Notation

- $A \succeq B \quad \Leftrightarrow \quad A - B$ is positive semidefinite

- $A \succ B \quad \Leftrightarrow \quad A - B$ is positive definite

Recall

An $n \times n$ symmetric matrix is positive definite if and only if all of its eigenvalues are positive.

Recall

An $n \times n$ symmetric matrix is positive semi-definite if and only if all of its eigenvalues are non-negative.

IMPORTANT

Recognizing Local Minima

- First-order *necessary* condition

$$\nabla f(x) = 0$$

- Second-order *necessary* condition

$$\nabla f(x) = 0 \quad \text{and} \quad H(x) \succeq 0$$

- Second-order *sufficient* condition

$$\nabla f(x) = 0 \quad \text{and} \quad H(x) \succ 0$$

IMPORTANT

Example: Linear Least Squares

Objective function: $f(x) = \frac{1}{2}\|Ax - b\|^2$

- Gradient: $\nabla f(x) = A^\mathsf{T} Ax - A^\mathsf{T} b$,
- $\nabla f(x^\star) = 0 \Rightarrow A^\mathsf{T} Ax^\star = A^\mathsf{T} b$ (Normal Equations),
- Hessian: $H(x) = A^\mathsf{T} A \succ 0$.

Assumption

- $A \in \mathbb{R}^{n \times m}$ *and* $b \in \mathbb{R}^n$
- $n \geq m \Leftrightarrow A$ *is a tall matrix*
- $\mathrm{rank}(A) = m$ *(i.e., columns of* A *are linearly independent)*

Damped Newton-Raphson Algorithm

We use a step size $0 < \alpha < 1$ to control each update size (damped Newton).

1. Start with an initial guess x_0 $(k = 0)$.
2. If $\|\nabla f(x_k)\| = 0$, then the algorithm is converged.
3. Solve the linear system $H_k \Delta x_k = -\nabla f(x_k)$.
4. Update the decision variable via $x_{k+1} = x_k + \alpha \Delta x_k$.
5. Repeat (go back to 2) until convergence.

Notation

- $A \succeq B \quad \Leftrightarrow \quad A - B$ is positive semidefinite
- $A \succ B \quad \Leftrightarrow \quad A - B$ is positive definite
- $\|x\| := \|x\|_2 := \sqrt{x^\mathsf{T} x}$
- $\|x\|_1 := |x_1| + \cdots + |x_n| = \sum_{i=1}^n |x_i|$
- $\|x\|_p := \big(|x_1|^p + \cdots + |x_n|^p\big)^{1/p} = \big(\sum_{i=1}^n |x_i|^p\big)^{1/p}$
- $x \cdot y := \langle x,y \rangle := x^\mathsf{T} y$
- Norm ball $\mathcal{B}(x_c,r) := \{x \in \mathbb{R}^n : \|x - x_c\| \leq r\}$

Structures: Smoothness

- $f : \mathbb{R}^n \to \mathbb{R}$

$$\overset{\text{gradient} \in \mathbb{R}^n}{\overset{\uparrow}{\nabla f(x)}} \;:=\; \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} \qquad \overset{\text{Hessian} \in \mathbb{S}^n}{\overset{\uparrow}{H(x)}} \;:=\; \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1\,\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1\,\partial x_n} \\ \frac{\partial^2 f}{\partial x_2\,\partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2\,\partial x_n} \\ \vdots & \vdots & \ldots & \vdots \\ \frac{\partial^2 f}{\partial x_n\,\partial x_1} & \frac{\partial^2 f}{\partial x_n\,\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

- f is continuously differentiable (and analytic) → Taylor's Theorem

$$f(x+d) = f(x) + \nabla f(x)^{\mathsf{T}} d + \frac{1}{2}\, d^{\mathsf{T}} H(x)\, d + o(\|d\|^2)$$

Second-order Taylor Approximation

- Local quadratic approximation

$$f(x_0 + d) \approx f(x_0) + \nabla f(x_0)^{\mathsf{T}} d + \frac{1}{2}\, d^{\mathsf{T}} H(x_0)\, d$$

- Change of variables $x := x_0 + d$

$$f(x) \approx f(x_0) + \nabla f(x_0)^{\mathsf{T}} (x - x_0) + \frac{1}{2}\, (x - x_0)^{\mathsf{T}} H(x_0)\, (x - x_0)$$

Recognizing Local Minima

- First-order *necessary* condition

$$\nabla f(x) = 0$$

- Second-order *necessary* condition

$$\nabla f(x) = 0 \quad \text{and} \quad H(x) \succeq 0$$

- Second-order *sufficient* condition

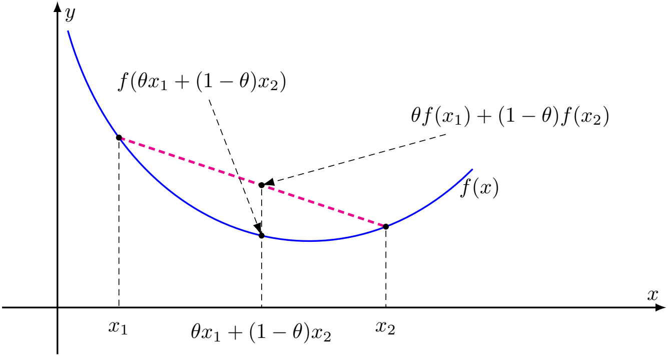
$$\nabla f(x) = 0 \quad \text{and} \quad H(x) \succ 0$$

Structure: Convexity

$f : \mathbb{R}^n \to \mathbb{R}$ $(\mathrm{dom} f = \mathbb{R}^n)$ is convex iff:

1. For all $x_1, x_2 \in \mathbb{R}^n$ and all $\theta \in [0,1]$:

$$f(\theta x_1 + (1-\theta)x_2) \le \theta f(x_1) + (1-\theta) f(x_2)$$



y
$f(\theta x_1 + (1-\theta)x_2)$
$\theta f(x_1) + (1-\theta) f(x_2)$
$f(x)$
x
x_1
$\theta x_1 + (1-\theta)x_2$
x_2

Structure: Convexity

$f : \mathbb{R}^n \to \mathbb{R}$ $(\mathrm{dom} f = \mathbb{R}^n)$ is convex iff:

1. For all $x_1, x_2 \in \mathbb{R}^n$ and all $\theta \in [0,1]$:

$$f(\theta x_1 + (1-\theta)x_2) \leq \theta f(x_1) + (1-\theta) f(x_2)$$

2. First-order condition: For all $x, y \in \mathbb{R}^n$:

$$f(y) \geq f(x) + \nabla f(x)^{\mathsf{T}}(y - x)$$

3. Second-order condition: For all $x \in \mathbb{R}^n$:

$$H(x) \succeq 0$$

Linear Least Squares (LS)

$f(x) = \frac{1}{2}\|Ax - b\|^2$

- Gradient: $\nabla f(x) = A^\mathsf{T} Ax - A^\mathsf{T} b$
- Hessian: $H(x) = A^\mathsf{T} A$

Assumption

- $A \in \mathbb{R}^{m \times n}$ *and* $b \in \mathbb{R}^m$
- $m \geq n \Leftrightarrow A$ *is a tall matrix*
- $\operatorname{rank}(A) = n$ *(i.e., columns of* A *are linearly independent)*

Claim

$\nabla f(x) = 0$ *is necessary and sufficient for global optimality.*

Claim

Unique minimizer iff $\operatorname{rank}(A) = n$.

Linear Least Squares

Lemma

$A \in \mathbb{R}^{m\times n}$ *has linearly independent columns* $\Leftrightarrow A^\mathsf{T} A \succ 0$.

$$\nabla f(x^\star) = 0 \Rightarrow x^\star = (A^\mathsf{T} A)^{-1} A^\mathsf{T} b$$

- A is full (column) rank $\Rightarrow A^\mathsf{T} A \succ 0$ is invertible
- Solve a linear system — "Normal Equations"

$$(A^\mathsf{T} A) x^\star = A^\mathsf{T} b$$

- Cholesky ($A^\mathsf{T} A = LL^\mathsf{T}$) or QR ($A = QR$) factorization

IMPORTANT

Nonlinear Least Squares (NLS)

$f(x) = \frac{1}{2}\|r(x)\|^2$

- $r : \mathbb{R}^n \to \mathbb{R}^m \; (m \geq n)$
- r is smooth, but not necessarily affine (i.e., $Ax + b$)
- $\|r(x)\|^2 = \sum_{i=1}^{m} r_i^2(x)$ where $r_i : \mathbb{R}^n \to \mathbb{R}$
- First-order Taylor expansion:

$$r_i(x) \approx r_i(x_0) + \nabla r_i(x_0)^{\mathsf{T}}(x - x_0)$$

- Stack r_i's:

$$r(x) \approx r(x_0) + \underset{\text{Jacobian}}{J(x_0)}\,(x - x_0)$$

- Change of variable:

$$r(x_0 + d) \approx r(x_0) + J(x_0)d$$

IMPORTANT

Jacobian

$$J(x) := \frac{\partial r(x)}{\partial x} = \begin{bmatrix} \frac{\partial r_1}{\partial x_1} & \frac{\partial r_1}{\partial x_2} & \cdots & \frac{\partial r_1}{\partial x_n} \\ \frac{\partial r_2}{\partial x_1} & \frac{\partial r_2}{\partial x_2} & \cdots & \frac{\partial r_2}{\partial x_n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial r_m}{\partial x_1} & \frac{\partial r_m}{\partial x_2} & \cdots & \frac{\partial r_m}{\partial x_n} \end{bmatrix} \in \mathbb{R}^{m \times n}$$

IMPORTANT

Gauss-Newton (GN)

1. Start from an initial guess x^0

for $k = 0{,}1, \cdots$ and until "convergence":

2. Linearize the residual at the current guess x^k

$$r(x^k + d) \approx r(x^k) + J(x^k)d$$

3. Solve the resulting linear least squares to find the step d

$$\underset{d}{\text{minimize}} \quad \|r(x^k) + J(x^k)d\|^2$$

$$(J_k^{\mathsf{T}} J_k)d = -J_k^{\mathsf{T}} r(x^k)$$

4. $x^{k+1} = x^k + d$

Newton

- Gradient $g_k := \nabla f(x^k)$
- Hessian $H_k := \nabla^2 f(x^k)$
- Second-order Taylor:

$$f(x^k + d) \approx m_k(d) := \frac{1}{2} d^\mathsf{T} H_k d + g_k^\mathsf{T} d + \underbrace{f(x^k)}_{\text{constant}}$$

- $m_k(d)$ gives the local quadratic approximation
- Find d by minimizing $m_k(d)$:

$$\nabla m_k(d) = 0 \Rightarrow H_k d + g_k = 0$$

- Well-defined if $H_k \succ 0 \Rightarrow \boxed{d = -H_k^{-1} g_k}$ and $x^{k+1} = x^k + d$
- In general has no preference for local minima over local maxima (i.e., stationary points)
- Very fast (quadratic) convergence near solutions

Newton vs. Gauss-Newton

- Nonlinear least squares

$$f_{\mathsf{NLS}}(x) = \frac{1}{2}\|r(x)\|^2$$

- Gradient and Hessian of f_{NLS}

$$\nabla f_{\mathsf{NLS}}(x^k) =: g_k = J_k^{\mathsf{T}} r(x^k)$$

$$\nabla^2 f_{\mathsf{NLS}}(x^k) =: H_k = J_k^{\mathsf{T}} J_k + \underbrace{\sum_{i=1}^{m} r_i(x^k)\nabla^2 r_i(x^k)}_{S}$$

Newton vs. Gauss-Newton

- Newton iteration

$$(J_k^\mathsf{T} J_k + S)d = -J_k^\mathsf{T} r(x^k)$$

- Gauss-Newton iteration

$$(J_k^\mathsf{T} J_k)d = -J_k^\mathsf{T} r(x^k)$$

- Gauss-Newton is expected to behave like Newton (fast convergence close to a solution) if S is “small” (e.g., small-residual regime $r_i(x^\star) \approx 0$)

- $J_k^\mathsf{T} J_k$ is a PSD approximation of Hessian — S can make Hessian non-PSD!

IMPORTANT

Globalization Strategies: Line Search

- $x^{k+1} = x^k + \alpha d$ where α is the step size
- d is a descent direction if $f(x^{k+1}) < f(x^k)$ for a sufficiently small step size

directional derivative $\displaystyle \lim_{\alpha \to 0} \frac{f(x^k + \alpha d) - f(x^k)}{\alpha} = g_k^\mathsf{T} d$

$g_k^\mathsf{T} d < 0 \Rightarrow$ d is a descent direction

1. Pick a *descent* direction d
   - Newton direction is a descent direction if $H_k \succ 0$
   - Gauss-Newton direction is a descent direction if J_k is full column rank
2. Find the best step size α (exact line search)

$$\underset{\alpha \in \mathbb{R}_{\geq 0}}{\text{minimize}} \; f(x^k + \alpha d)$$

- In practice $\rightarrow$ *inexact* line search (backtracking) + armijo rule
- Leads to *damped* Newton/Gauss-Newton

Globalization Strategies: Trust-Region Methods

- **Q.** How much do we trust our local approximate quadratic model $m_k(d)$ away from $d = 0$?

1. Pick a maximum step size Δ
2. Pick d by solving the trust-region subproblem

$$\underset{d}{\text{minimize}} \; m_k(d) \;\text{ s.t. } \|d\| \leq \Delta$$

3. Quantify and re-evaluate our trust on the model (i.e., Δ) based on

$$\frac{\text{actual reduction}}{\text{expected reduction}} = \frac{f(x^k) - f(x^k + d)}{m_k(0) - m_k(d)}$$

4. If ratio is below a threshold, reject d and shrink Δ by a factor; otherwise accept d and increase Δ by a factor

Levenberg-Marquardt

- has a trust-region interpretation
- instead of solving the trust-region subproblem, adds a penalty term $\lambda\|d\|^2$ to $m_k(d)$ to penalize a large d

$$\frac{1}{2}d^\mathsf{T}(H_k + \lambda I)d^\mathsf{T} + g_k^\mathsf{T} d + f(x^k)$$

- larger $\Delta \Leftrightarrow$ larger trust region $\Leftrightarrow$ smaller penalty factor λ
- λ is updated similar to Δ
- nonlinear least squares:
  - Levenberg $(J_k^\mathsf{T} J_k + \lambda I)d = -J_k^\mathsf{T} r(x^k)$
  - Marquardt $(J_k^\mathsf{T} J_k + \lambda \operatorname{diag}(J_k^\mathsf{T} J_k))d = -J_k^\mathsf{T} r(x^k)$
- interpolation between gradient descent (large λ) and Gauss-Newton (small λ)

Direct methods for solving linear systems

- Ultimately need to solve $Ad = b$ where $A \in \mathsf{Sym}(n)$ and $b \in \mathbb{R}^n$
  - e.g., in Gauss-Newton

    $$A = (J_k^\mathsf{T} J_k) \text{ and } b = -J_k^\mathsf{T} r(x^k)$$

  - e.g., in Levenberg-Marquardt

    $$A = (J_k^\mathsf{T} J_k + \lambda I) \text{ and } b = -J_k^\mathsf{T} r(x^k)$$

- Do not invert A!
  - will lose structure (e.g., A may be sparse but A^{-1} will be generally dense)
  - numerical stability

- We consider two direct methods based on Cholesky and QR factorizations.

Cholesky solver

- Solving triangular systems is fast/easy (forward/backward substitution):

$$\begin{bmatrix} \ell_{11} & 0 & 0 \\ \ell_{12} & \ell_{22} & 0 \\ \ell_{13} & \ell_{23} & \ell_{33} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

- Cholesky decomposition (assuming $A \succ 0$)

  i. $A = LL^\mathsf{T}$ where L is lower triangular and thus L^T is upper triangular

  $$L \underbrace{L^\mathsf{T} d}_{y} = b$$

  ii. solve $Ly = b$ via forward substitution

  iii. solve $L^\mathsf{T} d = y$ via backward substitution

QR Solver

- “Economic” QR factorization of $A = QR$
  - $Q \in \mathbb{R}^{m \times n}$ and $Q^\mathsf{T} Q = I_n$
  - $R \in \mathbb{R}^{n \times n}$ is upper triangular

- Solve $Rd = Q^\mathsf{T} c$ instead of $Ad = b$

$$Ad = b \Rightarrow Q^\mathsf{T} Q R d = Q^\mathsf{T} c \qquad Q^\mathsf{T} Q = I_n$$

$$\Rightarrow \boxed{Rd = Q^\mathsf{T} c}$$

solve via backward substitution

QR vs. Cholesky

✓ QR does not need to form A - works with J_k or $\begin{bmatrix} J_k \\ \sqrt{\lambda} I_n \end{bmatrix}$

✓ Better numerical stability than Cholesky

× Slower than Cholesky

Nonlinear Least Squares (NLS)

$f(x) = \frac{1}{2}\|r(x)\|^2$

- $r : \mathbb{R}^n \to \mathbb{R}^m$ $(m \geq n)$
- r is smooth, but not necessarily affine (i.e., $Ax + b$)
- $\|r(x)\|^2 = \sum_{i=1}^m r_i^2(x)$ where $r_i : \mathbb{R}^n \to \mathbb{R}$
- First-order Taylor expansion:

$$r_i(x) \approx r_i(x_0) + \nabla r_i(x_0)^\mathsf{T}(x - x_0)$$

- Stack r_i's:

$$r(x) \approx r(x_0) + \underset{\text{Jacobian}}{J(x_0)}\,(x - x_0)$$

- Change of variable:

$$r(x_0 + d) \approx r(x_0) + J(x_0)d$$

Gauss-Newton (GN)

1 Start from an initial guess x^0

for $k = 0,1,\cdots$ and until "convergence":

2 Linearize the residual at the current guess x^k

$$r(x^k + d) \approx r(x^k) + J(x^k)d$$

3 Solve the resulting linear least squares to find the step d

$$\underset{d}{\text{minimize}} \;\; \|r(x^k) + J(x^k)d\|^2$$

$$(J_k^\mathsf{T} J_k)d = -J_k^\mathsf{T} r(x^k)$$

4 $x^{k+1} = x^k + d$

Iteratively Reweighted Least Squares (IRLS)

- We wish to minimize $f(x) = \frac{1}{2}\|r(x)\|_p^p$ (p-norm).

- This is no longer the least squares problem. So we can't use the Gauss-Newton algorithm.

- The trick is to convert it to a weighted least squares problem:

  $$\|r(x)\|_p^p = r^{\mathsf{T}}(x) W r(x),$$

  and

  $$W := \mathrm{diag}(|r_1(x)|^{p-2}, \ldots, |r_m(x)|^{p-2}).$$

- In practice, we start with $W = I$ and initialize x using the least squares solution. Then until convergence, we update W at each iteration and solve the least squares problem.

Example: Robust Linear Regression with ℓ_1-Regularizer

Given a dataset $\{(x_i,t_i)\}_{i=1}^N$, where x is the input and t is the target (output), we wish to find a linear model that explains data. The model is linear in weights with nonlinear basis functions.

$$y(x;w) = \sum_{j=0}^{N} w_j \phi_j(x) = w^\mathsf{T} \phi(x),$$

$$w = \mathrm{vec}(w_0,w_1,\ldots,w_N) \quad \text{and} \quad \phi = \mathrm{vec}(\phi_0,\phi_1,\ldots,\phi_N),$$

$\phi_0 = 1$ and w_0 is a bias parameter. A common basis function is the Gaussian (Squared Exponential) basis

$$\phi_j(x) = \exp\left(-\frac{(x-x_j)^2}{2s^2}\right),$$

The hyperparameter s is called the basis bandwidth or length-scale.

Example: Robust Linear Regression with ℓ_1-Regularizer

To find a robust (to outliers in data) and sparse estimate of $w \in \mathbb{R}^{N+1}$, we solve the following regularized problem.

$$\underset{w \in \mathbb{R}^{N+1}}{\text{minimize}} \quad \frac{1}{2} \sum_{i=1}^{N} |t_i - w^{\mathsf{T}} \phi(x_i)| + \frac{\lambda}{2} \|w\|_1,$$

or

$$\underset{w \in \mathbb{R}^{N+1}}{\text{minimize}} \quad f(w) := \frac{1}{2} \|t - \Phi w\|_1 + \frac{\lambda}{2} \|w\|_1,$$

where $t = \text{vec}(t_1, \ldots, t_N)$ and Φ is a $N \times N + 1$ design matrix

$$\Phi = \begin{bmatrix} \phi^{\mathsf{T}}(x_1) \\ \vdots \\ \phi^{\mathsf{T}}(x_N) \end{bmatrix}.$$

Example: Robust Linear Regression with ℓ_1-Regularizer

$$f(w) = \frac{1}{2}\|t - \Phi w\|_1 + \frac{\lambda}{2}\|w\|_1$$

$$f(w) = \frac{1}{2}(t - \Phi w)^{\mathsf{T}} B (t - \Phi w) + \frac{\lambda}{2} w^{\mathsf{T}} G w$$

$$B := \operatorname{diag}(|t_1 - w^{\mathsf{T}}\phi(x_1)|^{-1}, \ldots, |t_N - w^{\mathsf{T}}\phi(x_N)|^{-1})$$

$$G := \operatorname{diag}(|w_0|^{-1}, \ldots, |w_N|^{-1})$$

$$\nabla f(w) = \Phi^{\mathsf{T}} B \Phi w - \Phi^{\mathsf{T}} B t + \lambda G w$$

$$\nabla f(w^\star) = 0 \Rightarrow \boxed{w^\star = (\Phi^{\mathsf{T}} B \Phi + \lambda G)^{-1} \Phi^{\mathsf{T}} B t}$$

Example: Robust Linear Regression with ℓ_1-Regularizer

Remark

To avoid division by zero, we use $\max(\delta, |t_i - w^\mathsf{T}\phi(x_i)|^{-1})$ *and* $\max(\delta, |w_j|^{-1})$. δ *is a small number, e.g.,* $1e-6$.

Remark

$\|\cdot\|_1$ *norm minimization is known as Least Absolute Deviation Regression and is robust to outliers in the data.The* ℓ_1*-regularizer results in a sparse weight vector.*

Remark

There is no guarantee that IRLS converges. If the solver hits the maximum number of iterations, do not trust the solution without an inspection!

Maximum likelihood Type Estimates (M-Estimates)

M-Estimation is a method for making an estimate robust to outliers. An M-Estimate of x is defined by

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad \sum_{i=1}^{m} \rho\left(r_i(x)\right)$$

or by

$$\sum_{i=1}^{m} \frac{\partial \rho\left(r_i(x)\right)}{\partial x} = 0$$

Remark

A particular choice of $\rho(x) = -\log l(x)$ where $l(x)$ is the likelihood function leads to the ordinary maximum likelihood estimate.

Maximum likelihood Type Estimates (M-Estimates)

Using the chain rule we have

$$\sum_{i=1}^{m} \frac{\partial \rho\left(r_i(x)\right)}{\partial x} = \sum_{i=1}^{m} \frac{\partial \rho\left(r_i\right)}{\partial r_i} \cdot \frac{\partial r_i(x)}{\partial x} = 0$$

$$\sum_{i=1}^{m} \frac{\partial \rho\left(r_i\right)}{\partial r_i} \cdot \frac{r_i}{r_i} \cdot \frac{\partial r_i(x)}{\partial x} = \sum_{i=1}^{m} w(r_i)\frac{\partial r_i(x)}{\partial x} r_i = 0$$

where we defined $w(r_i) := \frac{\partial \rho(r_i)}{\partial r_i} \cdot \frac{1}{r_i}$.

Maximum likelihood Type Estimates (M-Estimates)

This allows us to redefine the problem using the following weighted least squares

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad \frac{1}{2} \sum_{i=1}^{m} w(r_i) r_i^2(x).$$

We can solve this problem by using IRLS.

Example: Robust Linear Regression via M-Estimation

To find an M-Estimate of $w \in \mathbb{R}^{N+1}$, we solve the following problem.

$$\underset{w \in \mathbb{R}^{N+1}}{\text{minimize}} \quad \sum_{i=1}^{N} \rho\left(t_i - w^\mathsf{T}\phi(x_i)\right),$$

We use the Cauchy loss function

$$\rho(r) = \frac{\alpha^2}{2}\log(1 + \frac{r^2}{\alpha^2}) \quad \text{and} \quad \frac{\partial \rho}{\partial r} = \frac{r}{1 + \frac{r^2}{\alpha^2}}$$

$$w(r) = \frac{\partial \rho\left(r\right)}{\partial r} \cdot \frac{1}{r} = \frac{1}{1 + \frac{r^2}{\alpha^2}}.$$

α is a parameter that controls where the loss begins to scale sublinearly.

References

- **Optimization books**:
  - Numerical Optimization - Jorge Nocedal, Stephen Wright
    https://www.math.uci.edu/~qnie/Publications/NumericalOptimization.pdf
  - Convex Optimization – Steven Boyd
    https://web.stanford.edu/~boyd/cvxbook/

- **Optimization lectures**:
  - Mobile robotics: methods & algorithms - University of Michigan
    https://github.com/UMich-CURLY-teaching/UMich-ROB-530-public
  - Convex Optimization - Stanford
    https://web.stanford.edu/class/ee364a/lectures.html