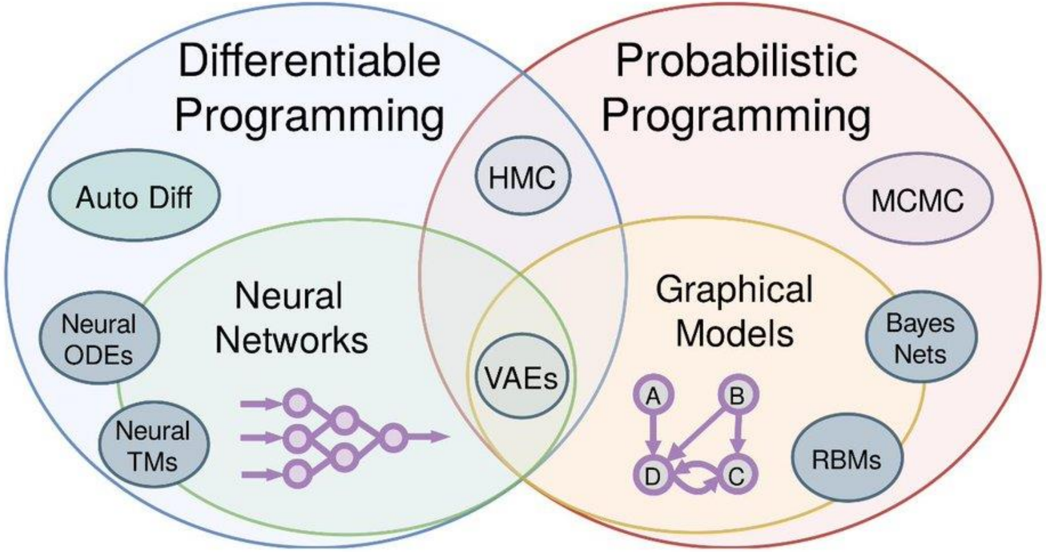
Optimization Theory

(A crash course)

Danda Pani Paudel

Computer Vision Lab @ ETH Zurich

Differential Programming

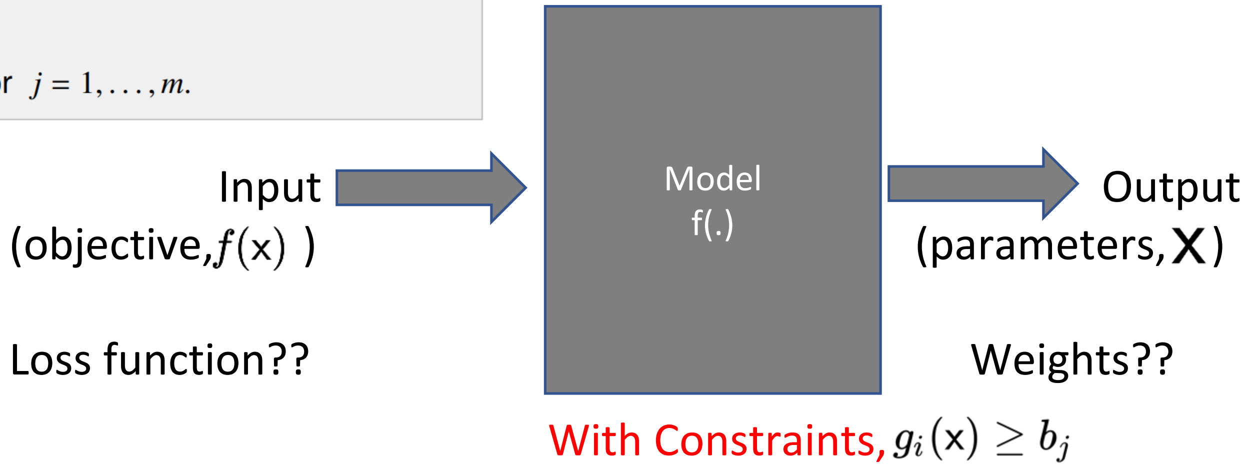


Optimization Problem

Definition : The Optimization Problem

A mathematical optimization problem (or optimization problem) on a vector of optimization variables $\mathrm{x} = (x_1, x_2, \ldots, x_n)^T$ is generally defined, for an objective $f : \mathbb{R}^n \to \mathbb{R}$ and constraints $g_j : \mathbb{R}^n \to \mathbb{R}$, in the following form :

$$\begin{array}{ll} \min\limits_{\mathrm{x}} & f(\mathrm{x}) \\ \text{subject to} & g_j(\mathrm{x}) \geq b_j, \quad \text{for } j = 1, \ldots, m. \end{array}$$



Linear Case w/o Constraints

- ❑ Linear system of equations

$$\mathsf{Ax} = \mathsf{b}$$

- ❑ Linear Least Square method
  - Given **A** and **b**, objective:

$$f(\mathsf{x}) = ||\mathsf{Ax} - \mathbf{b}||^2$$

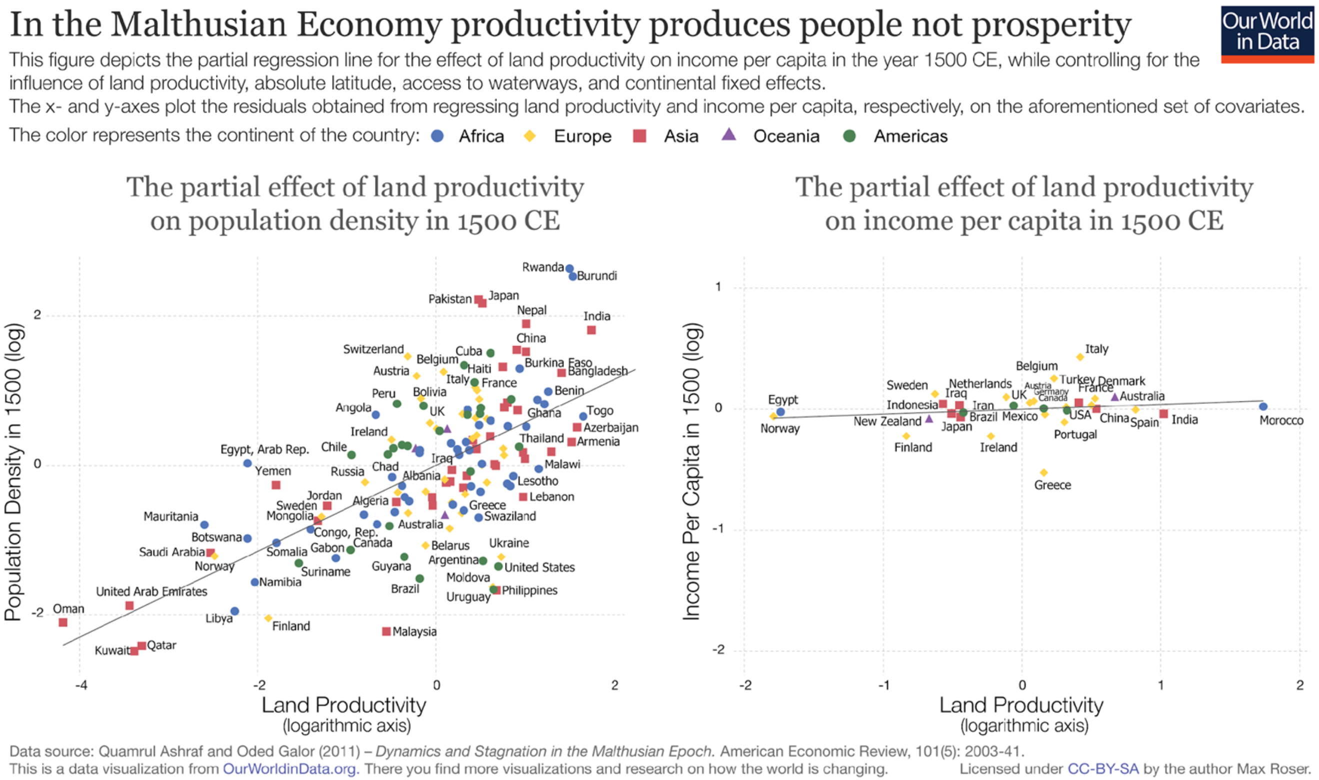
- ❑ Optimization problem

$$\min_{\mathsf{x}} \ ||\mathsf{Ax} - \mathbf{b}||^2$$

- ❑ Optimal Solution

$$\mathsf{x}^* = (\mathsf{A}^\mathsf{T}\mathsf{A})^{-1}\mathsf{A}^\mathsf{T}\mathbf{b}$$

Example: Line fitting



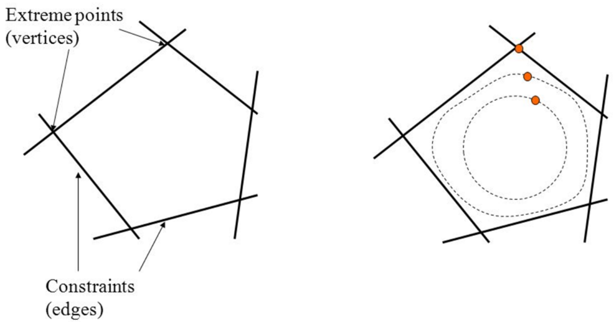
Source: Our World in data

Linear Case with Constraints

- ❑ Linear Programming

$$\min_{x} \; c^T x, \qquad s.t. \; Ax = 0, \quad Bx \geq 0.$$

- ❑ Solutions:
  - Simplex method
  - Interior-point method
  - Globally optimal and fast!

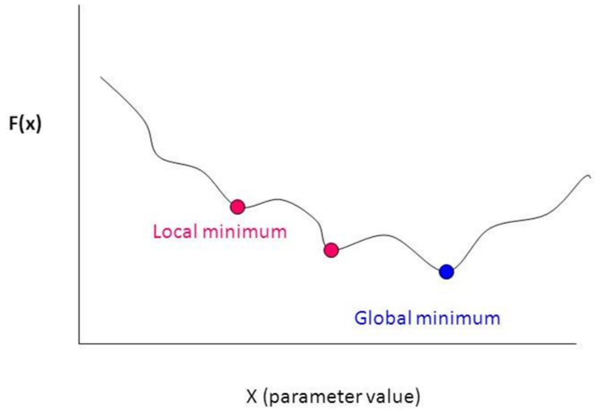


Simplex: search from vertex to vertex along the edges

Interior-point methods: go through the inside of the feasible space

Non-linear Optimization

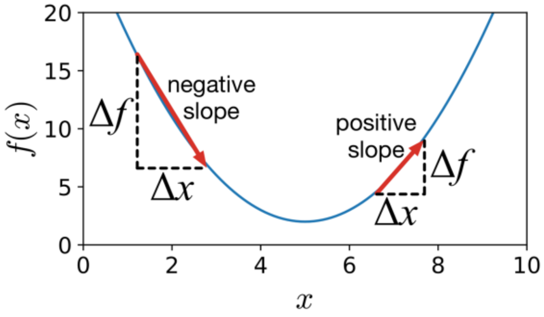
- ❑ Usually non-convex, therefore difficult
- ❑ Approaches: local and global methods

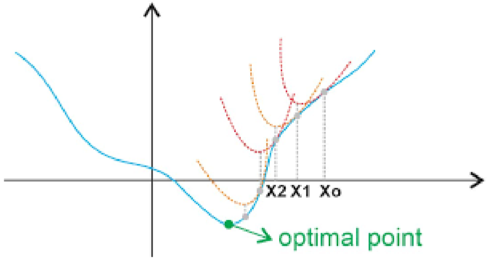


Local	Global
• Seek a solution that minimizes the objective locally	• Seek a solution that best minimizes the objective function (throughout the search space)
• No optimality certificate	• Optimal
• Require initilization	• No initialization required
• Suitable: when global solution not necessary or, a good initialization is provided	• Suitable: when problems are small in size (in terms of both variables and constraints), and the value of finding the best solution is very high, and the computation time is not critical.

Non-linear Case w/o Constraints

- Local methods
  - Gradient descent
  - Gauss-newton method
  - Levenberg Marquardt

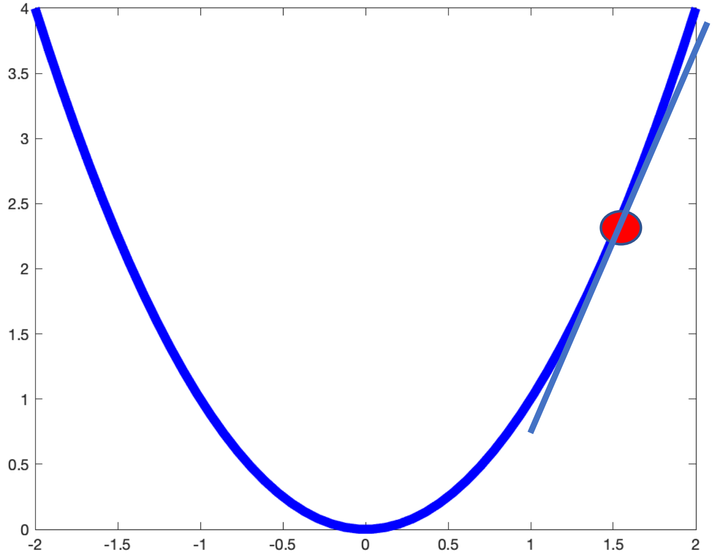




- Assumption: objective is differentiable w.r.t parametres
- Neural networks ⇒ require differentiable loss functions

Going against the Gradient

- Minimization



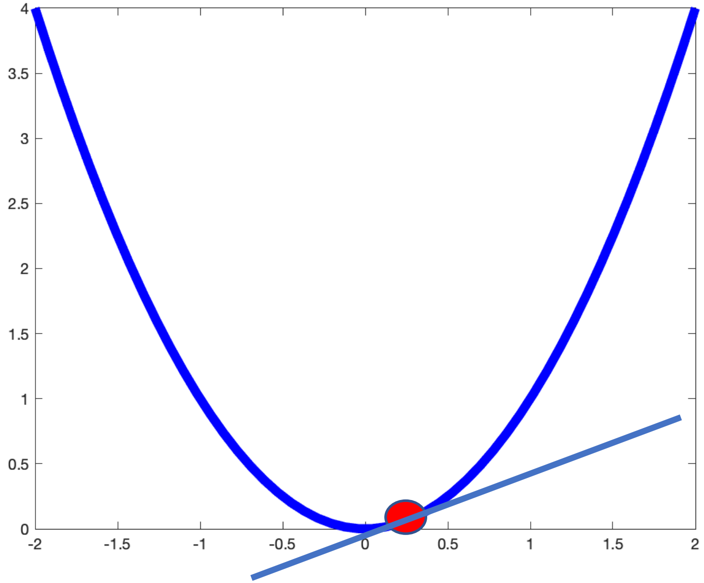
$$x^* = x^0 - \lambda \ \mathrm{dir}(\frac{\partial y}{\partial x})$$

$$x^* = x^0 - \lambda \ \mathrm{dir}(\frac{\partial y}{\partial x}) \ \mathrm{magnitude}(\frac{\partial y}{\partial x})$$

Update rate
(Learning rate??)

Going against the Gradient

- Minimization



$$x^* = x^0 - \lambda \ \mathrm{dir}(\frac{\partial y}{\partial x})$$

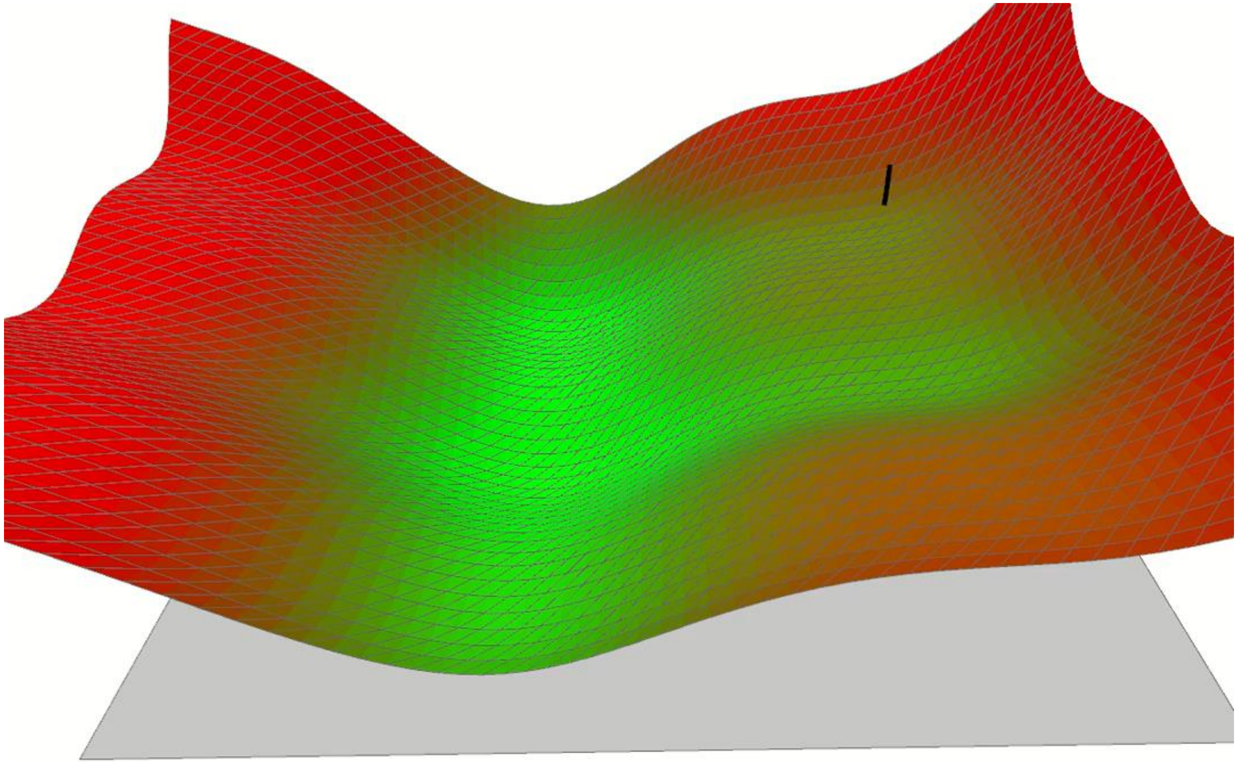
$$x^* = x^0 - \lambda \ \mathrm{dir}(\frac{\partial y}{\partial x}) \ \mathrm{magnitude}(\frac{\partial y}{\partial x})$$

$$y = x^2$$

← An example.

$$\frac{\partial y}{\partial x} = \frac{\partial x^2}{\partial x} = 2x$$

Going against the Gradient



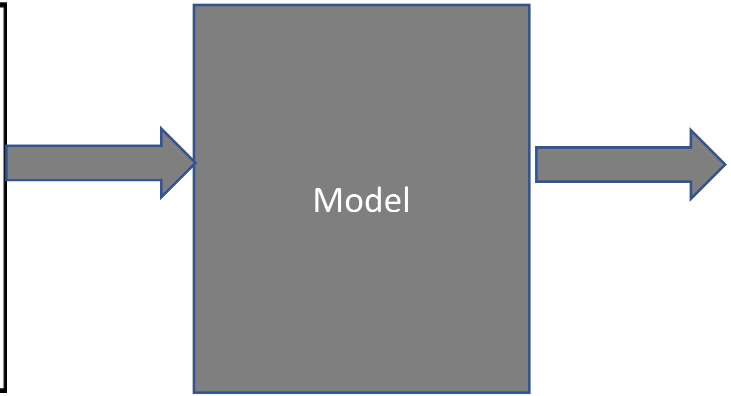
Visual Break!

Multivariate functions

- ❑ Multiple model variables

Model Parameters

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$



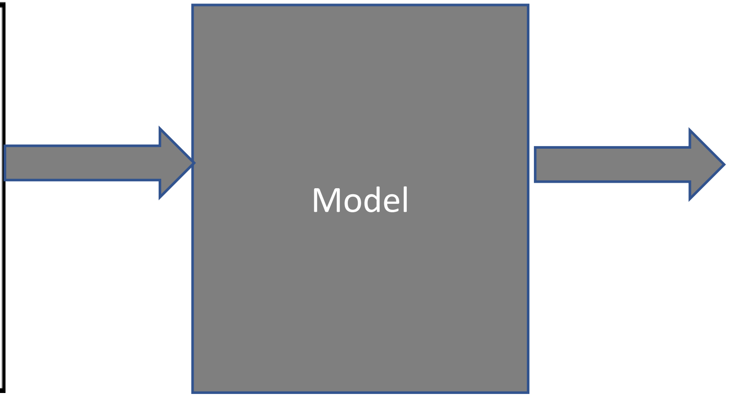
$$y = f(\mathbf{x})$$

Derivatives for multiple variables

❑ The Jacobian

Model Parameters

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$



$$y = f(\mathbf{x})$$

$$J(\mathbf{x}) = \begin{bmatrix} \dfrac{\partial y}{\partial x_1} \\ \dfrac{\partial y}{\partial x_2} \\ \vdots \\ \dfrac{\partial y}{\partial x_n} \end{bmatrix}^\top$$

Update equations

- Gradient descent

$$x_1^* = x_1^0 - \lambda \ \mathrm{dir}(\frac{\partial y}{\partial x_1}) \ \mathrm{magnitude}(\frac{\partial y}{\partial x_1})$$

$$x_2^* = x_2^0 - \lambda \ \mathrm{dir}(\frac{\partial y}{\partial x_2}) \ \mathrm{magnitude}(\frac{\partial y}{\partial x_2})$$

$$\vdots$$

$$x_n^* = x_n^0 - \lambda \ \mathrm{dir}(\frac{\partial y}{\partial x_n}) \ \mathrm{magnitude}(\frac{\partial y}{\partial x_n})$$

Update equations

- Gradient descent

$$x_1^* = x_1^0 - \lambda \frac{\partial y}{\partial x_1}$$

$$x_2^* = x_2^0 - \lambda \frac{\partial y}{\partial x_2}$$

$$\vdots$$

$$x_n^* = x_n^0 - \lambda \frac{\partial y}{\partial x_n}$$

$$\mathbf{x}^* = \mathbf{x}^0 - \lambda \begin{bmatrix} \frac{\partial y}{\partial x_1} \\ \frac{\partial y}{\partial x_2} \\ \vdots \\ \frac{\partial y}{\partial x_n} \end{bmatrix}$$

Update equations

- Gradient descent

$$x_1^* = x_1^0 - \lambda \frac{\partial y}{\partial x_1}$$

$$x_2^* = x_2^0 - \lambda \frac{\partial y}{\partial x_2}$$

$$\vdots$$

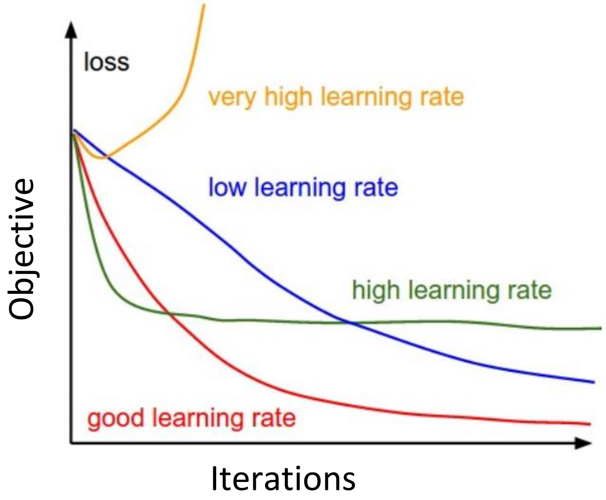
$$x_n^* = x_n^0 - \lambda \frac{\partial y}{\partial x_n}$$

$$\boxed{\mathbf{x}^* = \mathbf{x}^0 - \lambda J(\mathbf{x}^0)^\top}$$

Learning rate!

Convergence of Gradient Descent

- ❑ What happens if the gradient magnitude is very high?
- ❑ What happens if the gradient magnitude is very low?



Gauss Newton's method
(Second order approximation)

❑ Motivation

For faster convergence assume second-order approximation of cost

Taylor's series:

$$f(x) = f(x_0) + (x - x_0)f'(x_0) + \frac{(x-x_0)^2 f''(x_0)}{2!} + \ldots$$

Gauss Newton's method

- Multi dimensional case

- Taylor's series:

$$f(\mathsf{x}) = f(\mathsf{x}_0) + \mathsf{J}(\mathsf{x} - \mathsf{x}_0) + \frac{(\mathsf{x}-\mathsf{x}_0)^\mathsf{T}\mathsf{H}(\mathsf{x}-\mathsf{x}_0)}{2!} + \dots$$

$$\mathsf{J} = \mathsf{J}_f(\mathsf{x}_0) = \nabla f(\mathsf{x}_0)^\mathsf{T},$$

$$\mathsf{H} = \mathsf{H}_f(\mathsf{x}_0) = \nabla \mathsf{J}_f(\mathsf{x}_0).$$

- Newton's method?

Gauss Newton's method

$$f(\mathsf{x}) \approx f(\mathsf{x}_0) + \mathsf{J}(\mathsf{x} - \mathsf{x}_0)$$

- With $\Delta = (\mathsf{x} - \mathsf{x}_0)$, the task of finding Δ minimizing the sum of squares of the right-hand side; i.e.,

$$\min_{\Delta} \quad \| f(\mathsf{x}_0) + \mathsf{J}(\mathsf{x} - \mathsf{x}_0) \|_2^2$$

- Recall LLS solution, the optimal solution and update:

$$\Delta^* = \mathsf{x}^* - \mathsf{x}_0 = -(\mathsf{J}^\mathsf{T}\mathsf{J})^{-1}\mathsf{J}^\mathsf{T} f(\mathsf{x}_0), \rightarrow \mathsf{x}^* = \mathsf{x}_0 - (\mathsf{J}^\mathsf{T}\mathsf{J})^{-1}\mathsf{J}^\mathsf{T} f(\mathsf{x}_0)$$

Levenberg Marquardt Method

- Motivation

Combine both Gradient Descent and Gauss - Newton

$$\mathbf{x}^* = \mathbf{x}_0 - (\mathbf{J}^\top \mathbf{J} + \alpha \mathbf{I})^{-1} \mathbf{J}^\top f(\mathbf{x}_0)$$

Damping or regularizer

Levenberg Marquardt Method

- LM is often the first method of choice,

  BUT

  not for deep learning........

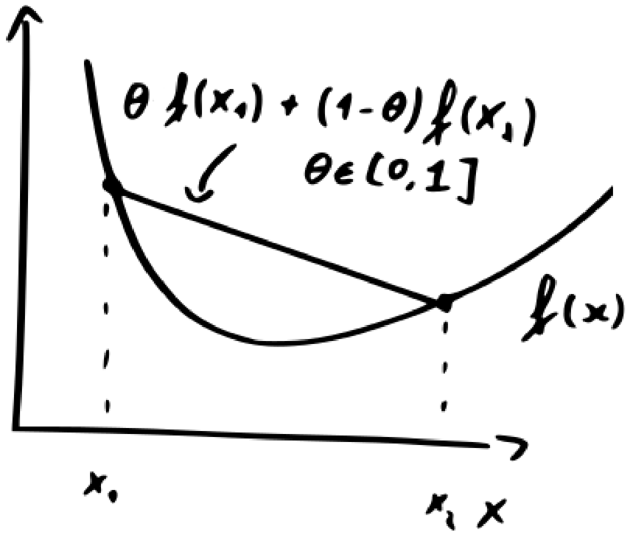
Why?

Fun Fact! The real name of Australian Camp is actually “Thulo Kharka” meaning big pasture. It used to be seasonal herding place of buffalo and cow herders from the villages bellow Dhampus and other. It is said that during late 80s, people from Austria found it so beautiful and they used to come and camp there for several days. Since then people started to call it Austrian camp and as it is difficult to pronounce for local Nepalese people and then pronounced it Australian camp this is how the place got new name as Australian Camp.

Global Optimization

- Convex problems:
  - Tractable methods
  - Easy to solve

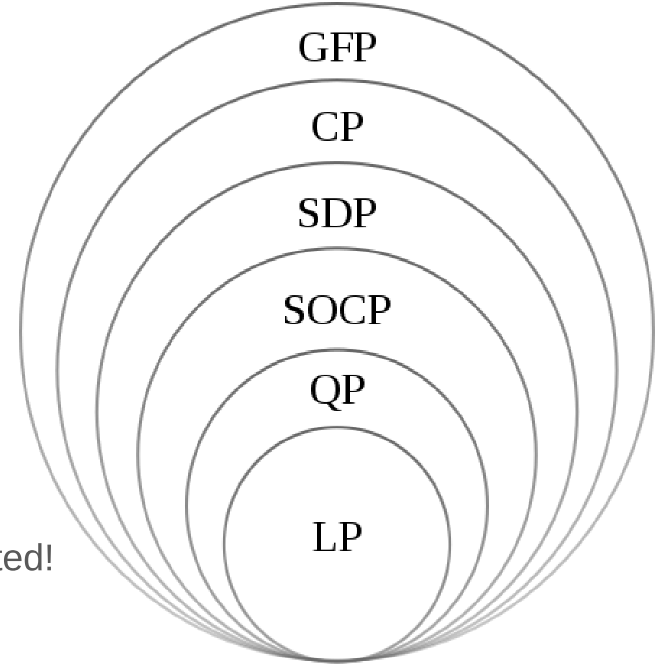
- Otherwise, NP-hard
  - Branch and Bound
  - Genetic algorithm



$$\forall\theta[0,\,1],\, f(\theta \mathrm{x}_1 + (1-\theta)\mathrm{x}_2) \leq f(\mathrm{x}_1) + (1-\theta)f(\mathrm{x}_2)$$

Non-linear convex problems

- A hierarchy of convex problems:
  - LP: linear program
  - QP: Quadratic Programs
  - SOCP: Second-order cone program
  - SDP: Semidefinite program
  - CP: Cone program
  - GFP: graph from program

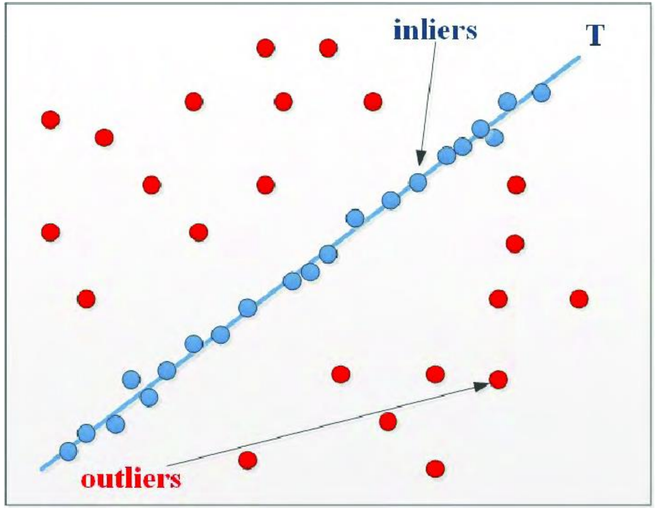


- Take-home message:
  - Know if your problem is one of the above, or can be formulated!

Soruce: https://en.wikipedia.org/wiki/Convex_optimization

Robust Optimization

- The objective function is derived from multiple measurements.
- In practices, measurements are full of noise and outliers
- How can one robustly fit the model in practice?



- Two techniques will be explored
  - M-estimator
  - Consensus maximization

M-estimator

- Residual function

  $$r : \mathbb{R}^n \to \mathbb{R}$$

- Penalty function

  $$\rho : \mathbb{R} \to \mathbb{R}$$

- Influence function

  $$\psi(x) = \frac{\delta\rho(x)}{\delta x}$$

- Weight function

  $$w(x) = \frac{\delta\psi(x)}{x}$$

Name	$\rho(x)$	$\psi(x)$	$\omega(x)$
Least-squares	$x^2/2$	x	1
L_1-norm	$\lvert x\rvert$	$sgn(x)$	$1/\lvert x\rvert$
L_p-norm	$\lvert x\rvert^p/p$	$sgn(x)\lvert x\rvert^{p-1}$	$\lvert x\rvert^{p-2}$
Fair	$\xi^2(\frac{\lvert x\rvert}{\xi} - log(1 + \frac{\lvert x\rvert}{\xi}))$	$\frac{x}{1+\lvert x\rvert/\xi}$	$\frac{1}{1+\lvert x\rvert/\xi}$
Cauchy	$\frac{\xi^2}{2} log(1 + x^2/\xi^2)$	$\frac{x}{(1+x^2/\xi^2)}$	$\frac{1}{(1+x^2/\xi^2)}$
Huber $\begin{cases} \lvert x\rvert \le \xi \\ \lvert x\rvert > \xi \end{cases}$	$\begin{cases} x^2/2 \\ \xi(\lvert x\rvert - \xi/2) \end{cases}$	$\begin{cases} x \\ \xi sgn(x) \end{cases}$	$\begin{cases} 1 \\ \xi/\lvert x\rvert \end{cases}$
Tukey $\begin{cases} \lvert x\rvert \le \xi \\ \lvert x\rvert > \xi \end{cases}$	$\begin{cases} \frac{x^6}{6} - \frac{\xi^2 x^4}{2} + \frac{\xi^4 x^2}{2} \\ \frac{\xi^6}{6} \end{cases}$	$\begin{cases} x\left(\xi^2 - x^2\right)^2 \\ 0 \end{cases}$	$\begin{cases} \left(\xi^2 - x^2\right)^2 \\ 0 \end{cases}$

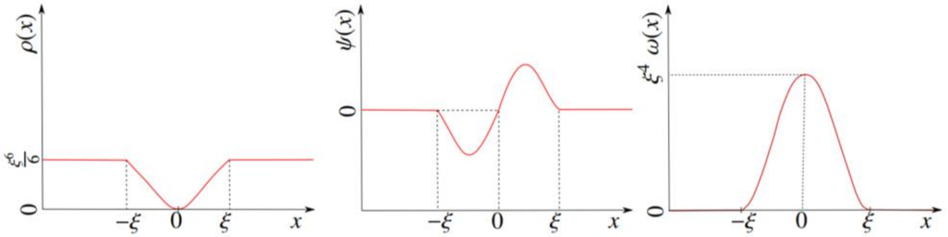


FIGURE – Tukey : penalty function (left), influence function (middle), and weight function (right), for the threshold value of ξ.

M-estimator contd.

Objective:

$$f(\mathsf{x}) = \sum_{k=1}^{p} \rho(r_k(\mathsf{x})).$$

$$\sum_{k=1}^{p} \psi(r_k(\mathsf{x})) \frac{\partial r_k(\mathsf{x})}{\partial \mathsf{x}_i} = 0, \quad \text{for} \quad i = 1, 2, \ldots, n.$$

$$\sum_{k=1}^{p} \omega(r_k(\mathsf{x})) r_k(\mathsf{x}) \frac{\partial r_k(\mathsf{x})}{\partial \mathsf{x}_i} = 0, \quad \text{for} \quad i = 1, 2, \ldots, n.$$

Update:

$$\mathsf{x}^l = \underset{\mathsf{x}}{\operatorname{argmin}} \ \sum_{k=1}^{p} w(r_k(\mathsf{x})^{l-1}) r_k(\mathsf{x})^2.$$

Consensus Maximization

- **Hypothesis:** the set of linear measurements maximize their consensus
- Most common method: RANSAC
  - RANSAC: Random Sample Consensus Search

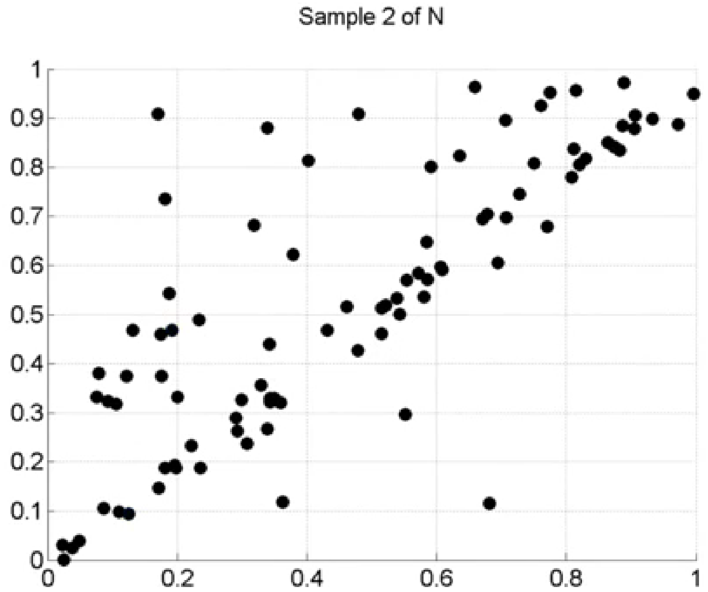
$$
\begin{aligned}
\max_{\mathcal{Z},\, \mathrm{x}} \quad & \sum_{k=1}^{p} z_k \\
\text{subject to} \quad & z_k |r_k(\mathrm{x})| \leq \epsilon, \quad \forall k, \\
& z_k \in \{0, 1\}, \quad \forall k.
\end{aligned}
$$

Algorithm Random Sample Consensus Search

Input : $\mathcal{H}$, numIter

Initialization : $\overline{f} = 0, \mathcal{Z} = \emptyset$
for $i = 1, 2, \ldots,$ numIter **do**
  $\mathcal{A}_i$ = selectRandomSamples($\mathcal{H}$)
  x_i = fitModel($\mathcal{A}_i$)
  $(\mathcal{Z}_i, f_i)$ = countNumInliers($\mathrm{x}_i, \mathcal{H}$)
  if $f_i > \overline{f}$ **then**
    $\overline{f} \leftarrow f_i$ and $\mathcal{Z} \leftarrow \mathcal{Z}_i$
  end if
end for
return $\mathcal{Z}$

The RANSAC Song



Source: https://www.youtube.com/watch?v=1YNjMxxXO-E

Application: Structure from Motion

RANSAC code with an example!

In 3D Vision Course...

Afternoon tea!