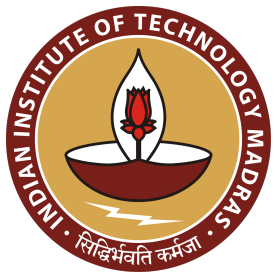
Unconstrained Optimization

Dr Ramkrishna Pasumarthy

Department of Electrical Engineering, IIT Madras
Chair UG Programs, IITM Zanzibar

DSA July 2026



Overview

Unconstrained Optimization [1]

- The objective function f to be minimized depends on n real variables $(x_1, x_2, \ldots, x_n)$ that can be chosen freely without constraints

Formulation

$$min_x \quad f(x)$$
$$x = (x_1, x_2, \ldots, x_n)$$

[1] This content is largely based on the book "Nocedal, J., Wright, S. J. (2006). Numerical optimization (2nd ed.). Springer Science+Business Media.

Application - Curve Fitting

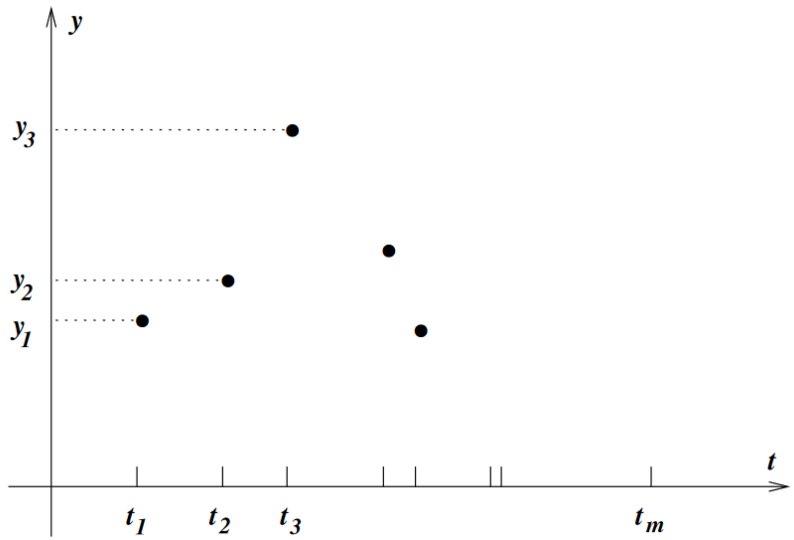


Figure: Curve fitting

- **Known:** $y(t_1), y(t_2), \ldots, y(t_m)$
- **Curve form:** $\phi(t, a)$ where a is parameter vector
- **Error:** $r_i(a) = y(t_i) - \phi(t_i, a)$
- **Objective:**

$$min_a \quad \sum_{i=1}^{m} r_i(a)^2$$

Notions in Optimization

Global Minimizer

x^* is called a **global minimizer** of f if $f(x^*) \leq f(x)$ for all $x \in \mathbb{R}^n$

Local Minimizer

x^* is called a **local minimize**r of f if there exists a neighborhood N containing x^* such that $f(x^*) \leq f(x)$ for all $x \in N$

Strict Global Minimizer

x^* is called a **strict global minimizer** of f if $f(x^*) < f(x)$ for all $x \in \mathbb{R}^n$

Strict Local Minimizer

x^* is called a **strict local minimizer** of f if there exists a neighborhood N containing x^* such that $f(x^*) < f(x)$ for all $x \in N$

Visualization of local versus global minima

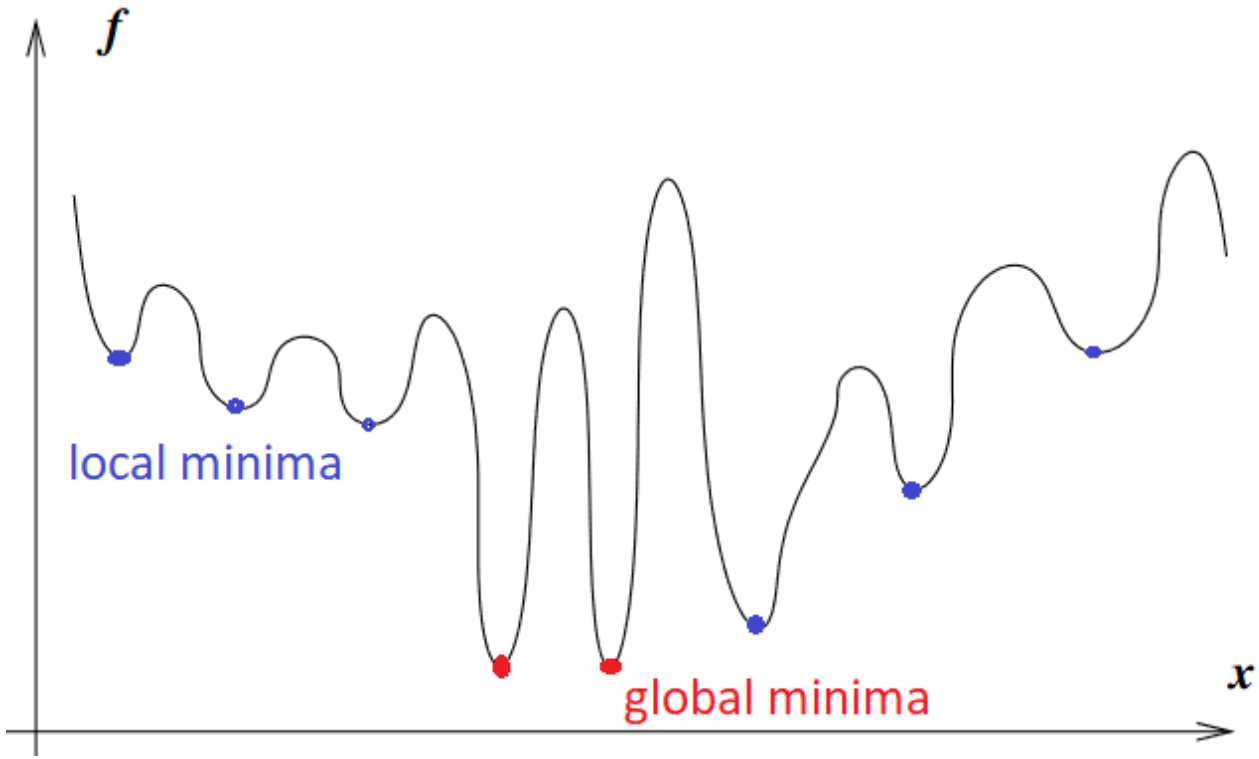


Figure: Difference between local and global minima

Taylor's Theorem

- Define: gradient of a function f as the vector
$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n} \right)$$

Taylor's Theorem: First Order

If f is continuously differentiable, then there exists $t \in (0, 1)$ such that

$$f(x + p) = f(x) + \nabla f(x + tp)^T p$$

- Define: Hessian of a function f as the matrix $(\nabla^2 f)_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}$

Taylor's Theorem: Second Order

If f is continuously differentiable, then there exists $t \in (0, 1)$ such that

$$f(x + p) = f(x) + \nabla f(x)^T p + p^T \nabla^f (x + tp) p$$

Recognizing local minimum - First Order Conditions

Necessary First Order Condition for local minimum

If x^* is a local minimizer of a continuously differentiable function f, then

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

Warning: This is necessary but not sufficient

- x^* is local minimizer $\implies \nabla f(x^*) = 0$
- But $\nabla f(x^*) = 0 \not\implies x^*$ is local minimizer

Counterexample:

- Even local maximizers satify this condition: $f(x) = -x^2$, $x^* = 0$

Proof of First Order Condition

- Assume x^* locally minimizes but $\nabla f(x^*) \neq 0$
- $\nabla f(x^*)^T \nabla f(x^* + tp) < 0$ when $t = 0$. So, there exists a $T > 0$ such that the inequality holds for all $t \in [0, T]$
- Hence for any $\bar{t} \in (0, t]$, we have

$$f(x^* + \bar{t}p) = f(x^*) + \bar{t} \underbrace{p^T \nabla f(x^* + tp)}_{<0 \text{ when } p = -\nabla f(x^*)}$$

  for some $t \in (0, \bar{t}]$.
- This would imply $f(x^* + tp) - f(x^*) < 0$ fot all $t \in (0, \bar{t}]$ which is a contradiction

Recognizing local minimum - Second Order Condition

Sufficient Second Order Condition for local minimum

If x^* satisfies the first order condition ($\nabla f(x^*) = 0$) and the Hessian $\nabla^2 f(x^*)$ is positive definite - i.e.

$$\nabla^2 f(x^*) \succ 0$$

(has all positive eigen values), then x^* is a local minimizer

Warning: This is sufficient but not necessary

- $\nabla f(x^*) = 0, \nabla^2 f(x^*) \succ 0 \implies x^*$ is a local minimizer
- But x^* is a local minimizer $\not\implies (x^*) = 0, \nabla^2 f(x^*) \succ 0$

Counterexample:

- $f(x) = x^4$, $x^* = 0$

Proof of Second Order Condition

- Since $\nabla^2 f(x^*) \succ 0$, for small enough $0 \leq t < T$, we have $\nabla^2 f(x^* + tp) \succ 0$.
- This means that for any $p \neq 0$, $p^T \nabla^2 f(x^* + tp)p > 0$ for all $t \in [0, T]$
- Putting $\nabla f(x^*) = 0$ in Taylor's theorem we get that for some $t \in (0, \bar{t}]$, we have

$$f(x + \bar{t}p) = f(x) + \frac{1}{2}\bar{t}^2 \underbrace{p^T \nabla^2 f(x^* + tp)p}_{>0 \text{ for all } p}$$

- This gives $f(x + \bar{t}p) - f(x) > 0$ for any $t \in (0, \bar{t})$ and hence it is a local minimum

Convex Functions - where local is global

Convexity

f is called convex it it satisfies for all $x, y \in \mathbb{R}^n$, for all $0 \leq t \leq 1$,

$$f(tx + [1-t]y) \leq tf(x) + [1-t]f(y)$$

Theorem

- If f is convex, any local minimum is a global minimum
- If f is convex and differentiable, $\nabla f(x^*) = 0 \iff x^*$ is a global minimizer

Example: A quadratic function $f(x) = a + bx + x^T Px$ with $P \succ 0$, is convex

Proof for convex functions

- Assume x^* is a local minimizer but not a global minimizer for a convex function. Then, we can find z such that $f(z) < f(x^*)$.
- Consider the line segment joining x^* and z where an arbitrary point x is described by $x = (1-\lambda)x^* + \lambda z$ with $0 \leq \lambda \leq 1$.
- By convexity,

$$f(x) = f((1-\lambda)x^* + \lambda z) \leq (1-\lambda)f(x^*) + \underbrace{\lambda f(z)}_{<f(x)} < f(x^*)$$

- This holds even when λ is small which means when x is close to x^* and contradicts the fact that x^* is a local minimum of f

Minimizing Quadratic Functions

- Let $f(x) = a + bx + \frac{1}{2}x^T Px$ where $P \succ 0$
- Since f is convex, first order condition is necessary and sufficient for a global minimum
- $\nabla f(x) = b + Px$ and hence the minimum is attained when

$$x^* = -P^{-1}b$$

- This is useful when f can be approximated by a second order Taylor expansion in the neighborhood of a point

Algorithms for finding minimizers - Broad Classification

- In any algorithm, a current iterate x_k is updated to $x_{k+1} = x_k + \alpha p_k$ where $\alpha > 0$ is a distance and $p_k \in \mathbb{R}^n$ is a direction

Line Search Methods

- $x_k \rightarrow x_{k+1}$: fix a direction p_k first
- Then find suitable $\alpha > 0$ such that

$$x_{k+1} = x_k + \alpha p_k$$
$$\text{such that } f(x_{k+1}) < f(x_k)$$

Trust Region Methods

- Approximate f around x_k by a model function m_k that holds well in a trust region T_k (points that are close enough to x_k)
- Choose p_k such that x_{k+1} minimizes m_k in the trust region T_k

Search Directions: Line Search

Steepest Descent

$$p_k = -\nabla f(x_k)$$

- Computation: Only one derivative ∇_k (lesser)
- Convergence: Linear (slower) in iterates

Newton Direction

$$p_k = -(\nabla^2 f)_k^{-1} \nabla f_k$$

if $\nabla^2 f_k \succ 0$

- Computation: Two derivatives (higher) ∇_k, ∇_k^2

Motivation: Steepest Descent

$$\frac{f(x_k + tp) - f(x_k)}{t} \approx \left.\frac{df}{dt}\right|_{x_k+tp} = \nabla f(x_k)^T p$$

- $\frac{df}{dt}$ is the most negative (maximum decrease of f) only if $p = -\nabla f(x_k)$
- So, the negative gradient direction provides the maximum decrease of the function for a give magnitude of p
- However any direction with $p^T \nabla f(x_k) < 0$ will work as well

Visualization: Steepest Descent

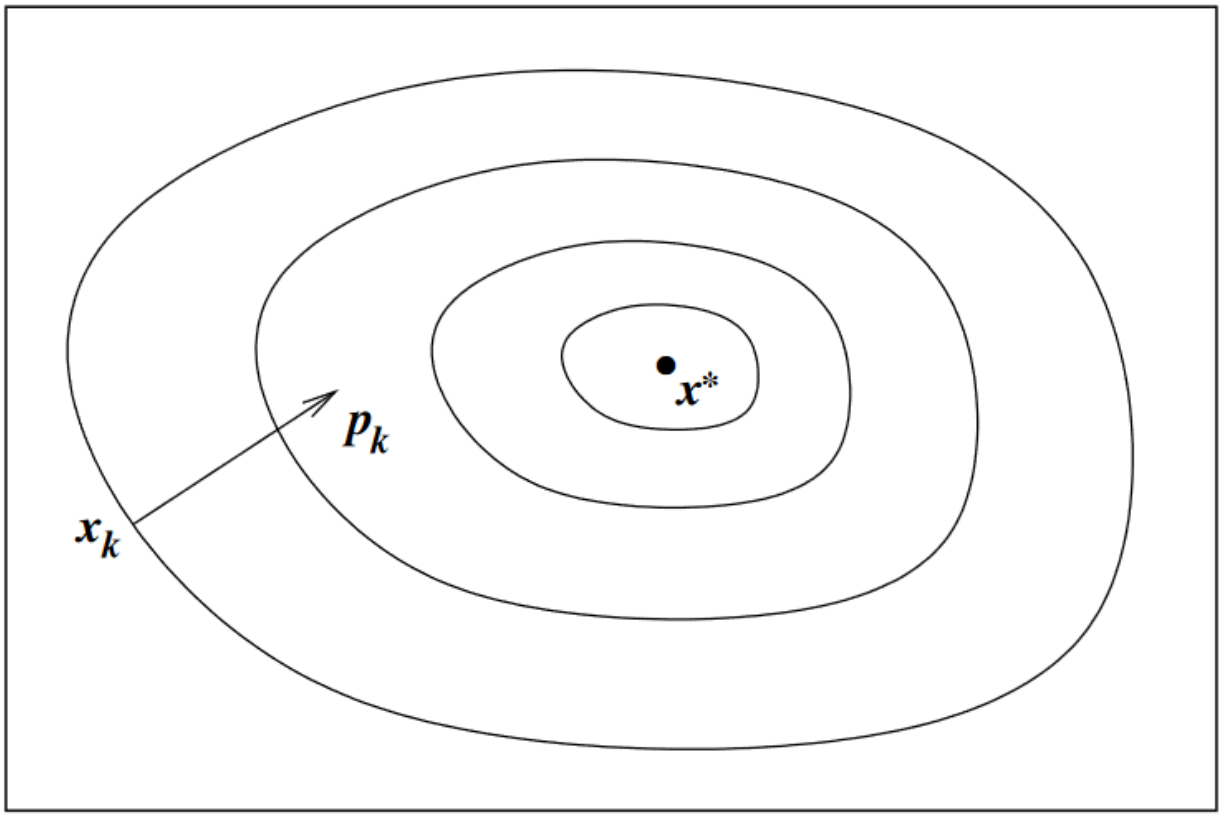


Figure: Visualizing steepest descent for a function of two variables

Motivation: Newton Direction

- Newton direction arises when one tries to minimize f upto second order approximation

$$f(x_k + p) \approx f(x_k) + p^T \nabla f(x_k) + \frac{1}{2} p^T \nabla^2 f_k p$$

- Quadratic function minimized when $p = -\nabla^2 f_k^{-1} \nabla f_k$ and hence the choice for p_k

Quasi-Newton Search Directions

- approximate $\nabla^2 f_k$ by B_k by observing the changes in gradients

Quasi-Newton Method

$$p_k = -B_k^{-1} \nabla f_k$$
$$s_k := x_{k+1} - x_k, \quad y_k := \nabla f_{k+1} - \nabla f_k$$

- **SR1-symmetric rank1 formula:**

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)(y_k - B_k s_k)^T}{(y_k - B_k s_k)^T s_k}$$

- **BFGS formula:**

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}$$

Model Function: Trust Region Methods

First Order Approximation

$$m_k(x_k + p) = f_k + p^T \nabla f_k$$

Second Order Approximation

$$m_k(x_k + p) = f_k + p^T \nabla f_k + \frac{1}{2} p^T B_k p$$

And finally, whatever be the model function,

$$x_{k+1} - x_k = p_k = min_p \;\; m_k(x_k + p)$$

Visualization: Trust Region Method

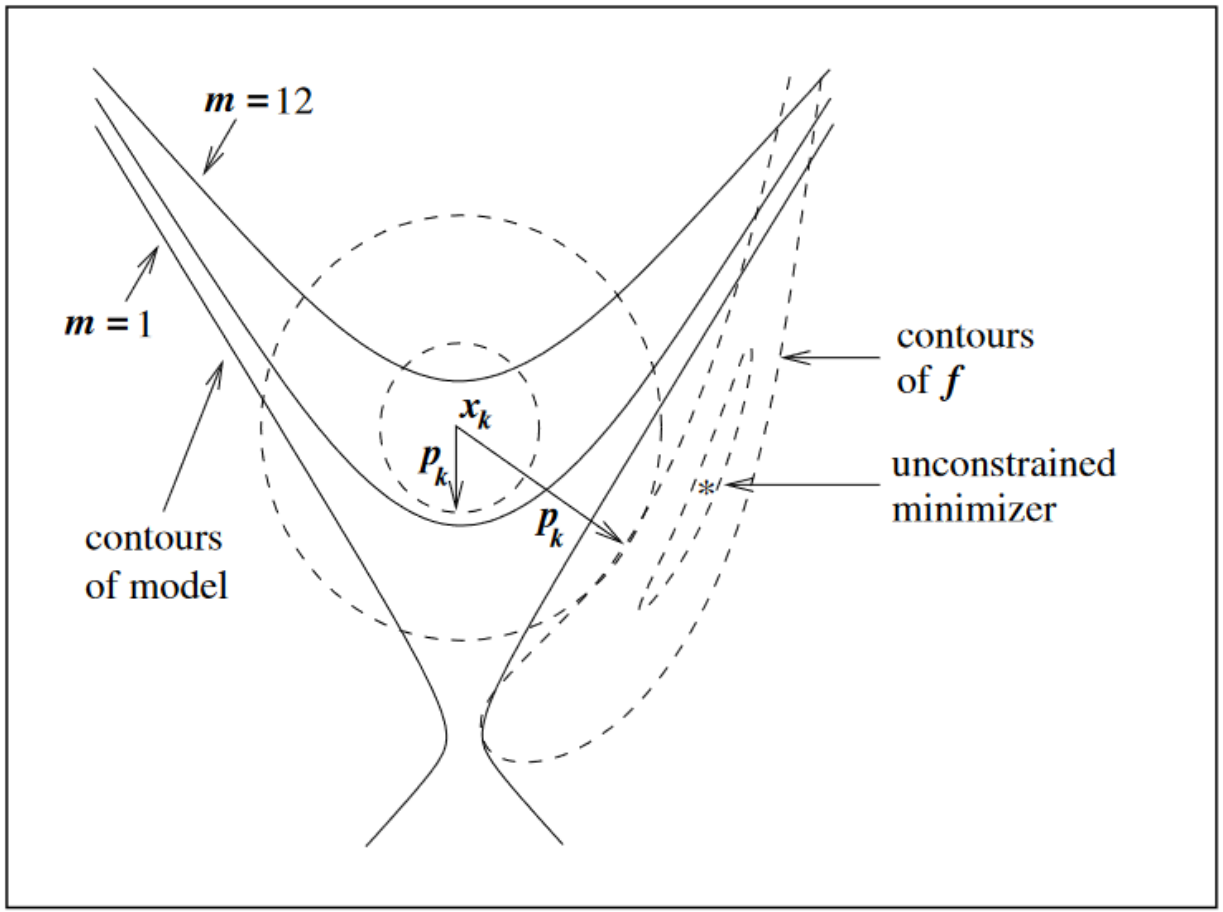


Figure: Visualizing two possible trust regions (circles) and their corresponding steps p_k where the solid lines are contours of the model function , dashed lines, the contours of $f(x) = 10(x_2 x_1^2)^2 + (1 - x_1^2)$

First Order Model Function

$$\begin{aligned} min_p \quad & (f_k + p^T \nabla f_k) \\ \text{sub to} \quad & ||p|| \leq \Delta_k \end{aligned}$$

- **Closed form solution:** $p_k = -\Delta_k \frac{\nabla f_k}{||\nabla f_k||}$
- this simply is steepest descent with distance = trust region radius

Second Order Model Function

$$\begin{aligned} min_p \quad & (f_k + p^T \nabla f_k) \\ \text{sub to} \quad & ||p|| \leq \Delta_k \end{aligned}$$

Solution: Find $p^* \in \mathbb{R}^n$ with $||p^*|| \leq \Delta_k$ and $\lambda \geq 0$

$$\begin{aligned} (B + \lambda I)p^* &= -\nabla f_k \\ \lambda(\Delta_k - ||p^*||) &= 0 \\ B + \lambda I &\succ 0 \end{aligned}$$

Scaling

- $f(x_1, x_2) = 10^8 x_1^2 + x_2^2$ is extremely sensitive to x_1 compared to x_2
- Defining $z_1 = 10^4 x_1, z_2 = x_2$ gives $f(z_1, z_2) = z_1^2 + z_2^2$ that is equally sensitive to both the variables
- Scaling normally involves the change of units of some variables, say from meters to millimeters. If we do, the range of those variables and their size relative to the other variables will both change.
- Some algorithms, such as steepest descent, are sensitive to poor scaling, while others, such as Newton's method, are unaffected by it.

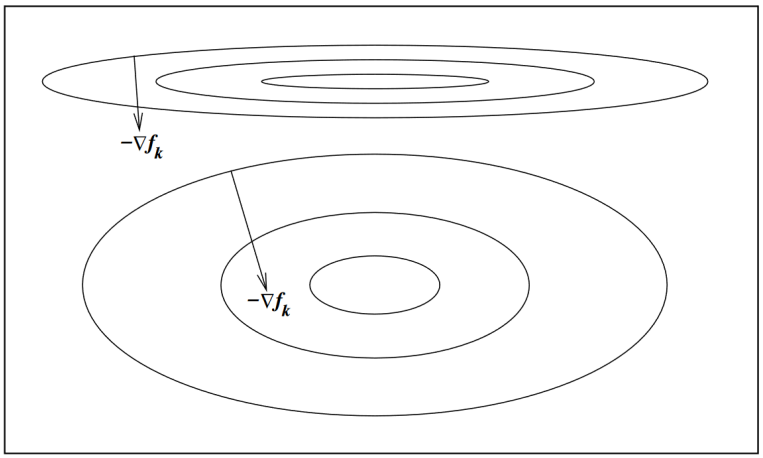


Figure: A poorly scaled and a well scaled problem

Application: Back to Least Squares Problem

- Consider the curve fitting problem again with linear fit $\phi(t_i, a) = Ja$ and the error $r_i = (Ja)_i - y_i$ but with the objective function as sum of squares of errors. Replacing a by x, we have

$$min_x \frac{1}{2}||Jx - y||^2$$

$$\nabla f(x) = J^T(Jx - y)$$
$$\nabla^2 f(x) = J^T J$$

- Convexity of the quadratic function determines that the global minimizer satisfies

$$J^T J x^* = J^T y$$

- In most cases, J^J is full rank and hence $x^* = (J^T J)^{-1} J^T y$. But matrix inversion is difficult to compute directly

Algorithms for least squares: Cholesky factorization

- Compute $J^T J$ and $J^T y$ separately
- Since $J^T J$ is symmetric and positive definite, compute the Cholesky factorization $J^T J = R^T R$ where R is an upper triangular matrix
- The equation to be solved now becomes

$$R^T R x^* = J^T y$$

- By performing two triangular substitutions, solve for x^*: i.e.
  - Call $Rx^* = x'$. The equation is now $R^T x' = J^T y$. Find x' by substitution as R^T is triangular
  - Having found x', now solve $Rx = x'$ by substitution since R is triangular

Algorithms for least squares: QR factorization

- Compute $J^T J$ and $J^T y$ separately
- Since $J^T J$ is symmetric and positive definite, compute the QR factorization $J^T J = QR$ where R is a triangular matrix and Q is an orthogonal matrix with $Q^{-1} = Q^T$
- The equation to be solved now becomes

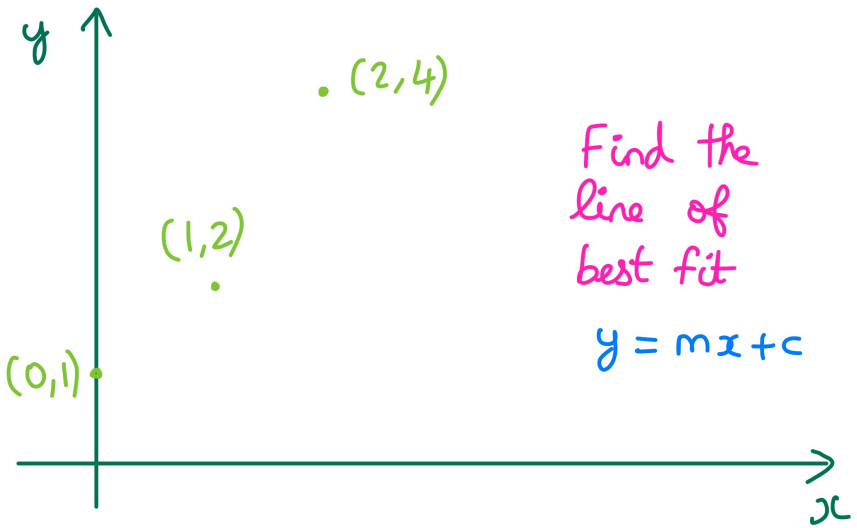
$$QRx^* = J^T y$$

- Multiplying Q^T both sides and observing $Q^T Q = I$, we have

$$Q^T(QRx^*) = Rx^* = Q^T J^T y$$

- Solve for x^*, the equation $Rx^* = Q^T J^T y$ by substitution as R is a triangular matrix

LS: Example



$(0,1)$ $\quad 1 = m(0) + c = c$

$r_1 = 1 - c$

$(1,2)$ $\quad 2 = m(1) + c = m + c$

$r_2 = 2 - m - c$

$(2,4)$ $\quad 4 = m(2) + c = 2m + c$

$r_3 = 4 - 2m - c$

LS: Example

$$\min_{m,c} \; r_1^2 + r_2^2 + r_3^2$$

$$= \left\| \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} m \\ c \end{bmatrix} - \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix} \right\|^2$$

$$J \qquad x \qquad y$$

$$J^T J = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 5 & 3 \\ 3 & 3 \end{bmatrix}$$

$$\begin{bmatrix} m \\ c \end{bmatrix} = (J^T J)^{-1} J^T \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix}$$

$$= \frac{1}{6} \begin{bmatrix} 3 & -3 \\ -3 & 5 \end{bmatrix} \begin{bmatrix} 0 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix}$$

$$= \begin{pmatrix} 1.5 \\ 0.833 \end{pmatrix}$$

LS: Gradient Descent

This same example is now iteratively solved through gradient descent

Graddsc.jog https://drive.google.com/drive/u/0/my-drive

$$\nabla f(x) = J^T(Jx - y) = J^TJx - J^Ty$$

Choosing constant step size α,

$$x_{k+1} = x_k - \alpha \nabla f(x_k)$$

choose $\alpha = 0.1$,

$$x_{k+1} = x_k - 0.1\, J^T(Jx_k - y)$$

$$\Rightarrow \boxed{x_{k+1} = x_k - 0.1\, J^TJ\, x_k + 0.1\, J^Ty}$$

Put $\begin{pmatrix} x_k^1 \\ x_k^2 \end{pmatrix} := \begin{pmatrix} m_k \\ c_k \end{pmatrix}$, we get

algorithm to be coded

$$\begin{pmatrix} m_{k+1} \\ c_{k+1} \end{pmatrix} = \begin{pmatrix} m_k \\ c_k \end{pmatrix} - 0.1\, J^TJ \begin{pmatrix} m_k \\ c_k \end{pmatrix} + 0.1\, J^Ty$$

1 of 1 11/14/2022, 1:48 PM

Working of first few iterates

- $m_0 = c_0 = 0$ -initial guess for the straight line
-

$$\begin{bmatrix} m_1 \\ c_1 \end{bmatrix} = \underbrace{\begin{bmatrix} m_0 \\ c_0 \end{bmatrix}}_{0,0} - 0.1 \begin{bmatrix} 5 & 3 \\ 2 & 3 \end{bmatrix} \underbrace{\begin{bmatrix} m_0 \\ c_0 \end{bmatrix}}_{0,0} + 0.1 \begin{bmatrix} 0 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix} \quad (1)$$

$$= \begin{bmatrix} 1 \\ 0.7 \end{bmatrix} \quad (2)$$

Working of first few iterates

- $m_1 = 1, c_1 = 0.7$ - guess for the second iteration
-

$$\begin{bmatrix} m_2 \\ c_2 \end{bmatrix} = \begin{bmatrix} m_1 \\ c_1 \end{bmatrix} - 0.1 \begin{bmatrix} 5 & 3 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} m_1 \\ c_1 \end{bmatrix} + 0.1 \begin{bmatrix} 0 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix} \tag{3}$$

$$= \begin{bmatrix} 1.29 \\ 0.89 \end{bmatrix} \tag{4}$$

Working of first few iterates

- $m_2 = 1.29, c_1 = 0.89$ - guess for the third iteration
-

$$\begin{bmatrix} m_3 \\ c_3 \end{bmatrix} = \begin{bmatrix} m_2 \\ c_2 \end{bmatrix} - 0.1 \begin{bmatrix} 5 & 3 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} m_2 \\ c_2 \end{bmatrix} + 0.1 \begin{bmatrix} 0 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix} \tag{5}$$

$$= \begin{bmatrix} 1.378 \\ 0.936 \end{bmatrix} \tag{6}$$

Working of first few iterates

- $m_3 = 1.378, c_3 = 0.936$ - guess for the fourth iteration
-

$$\begin{bmatrix} m_4 \\ c_4 \end{bmatrix} = \begin{bmatrix} m_3 \\ c_3 \end{bmatrix} - 0.1 \begin{bmatrix} 5 & 3 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} m_3 \\ c_3 \end{bmatrix} + 0.1 \begin{bmatrix} 0 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix} \tag{7}$$

$$= \begin{bmatrix} 1.4082 \\ 0.9418 \end{bmatrix} \tag{8}$$

Working of first few iterates

- $m_4 = 1.4082, c_4 = 0.9418$ - guess for the fifth iteration
-

$$\begin{bmatrix} m_5 \\ c_5 \end{bmatrix} = \begin{bmatrix} m_4 \\ c_4 \end{bmatrix} - 0.1 \begin{bmatrix} 5 & 3 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} m_4 \\ c_4 \end{bmatrix} + 0.1 \begin{bmatrix} 0 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix} \tag{9}$$

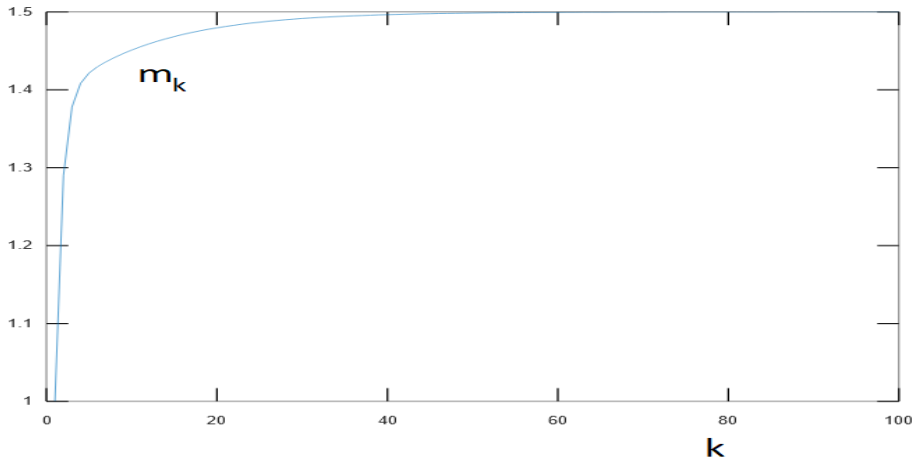
$$= \begin{bmatrix} 1.4216 \\ 0.9368 \end{bmatrix} \tag{10}$$

Other iterates

Some of the iterates that follow are given below

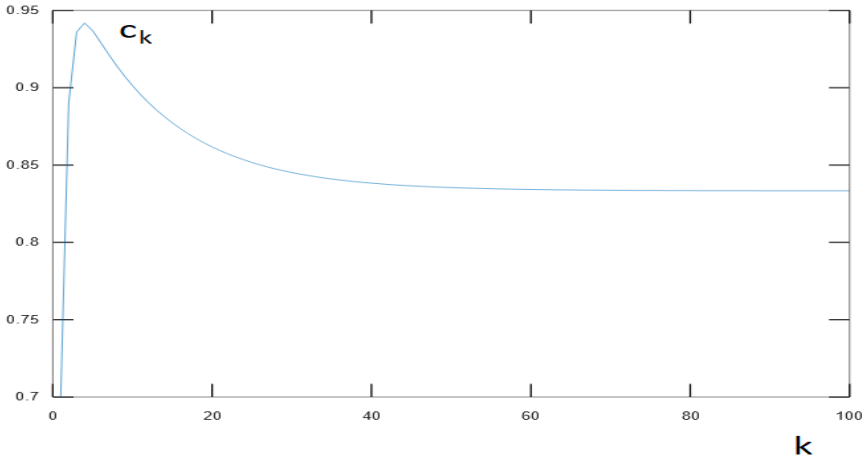
- $$\begin{bmatrix} m_{10} \\ c_{10} \end{bmatrix} = \begin{bmatrix} 1.4510 \\ 0.9013 \end{bmatrix}$$
- $$\begin{bmatrix} m_{20} \\ c_{20} \end{bmatrix} = \begin{bmatrix} 1.4796 \\ 0.8617 \end{bmatrix}$$
- $$\begin{bmatrix} m_{50} \\ c_{50} \end{bmatrix} = \begin{bmatrix} 1.4985 \\ 0.8354 \end{bmatrix}$$
- $$\begin{bmatrix} m_{100} \\ c_{100} \end{bmatrix} = \begin{bmatrix} 1.5000 \\ 0.8334 \end{bmatrix}$$

Plots



0.5

Figure: Plot of $m_k \to 1.5$



0.5

Figure: Plot of $c_k \to 0.8333$