統計計算與模擬

政治大學統計系余清祥

2026年4月8日～15日

第五單元：求根、極值(優化)

http://csyue.nccu.edu.tw

關於期末報告

如何整理量化分析的結果

- 分析大數據的策略與傳統資料分析並無不同，僅在大量、時效(速度)的需求及差異，特別需要資料所屬領域的知識支持，尤其定義研究目的、量化目標變數（標的）。
- 藉由大數據探討某個議題，如同撰寫論文、報告，具備幾個必要的元素。

→即使研究主題相同，因為切入角度、研究素材（資料）、方法理論（研究者專業）等，使得研究方向、甚至研究結論會大異其趣。

報告撰寫的幾個關鍵要素

- 一篇研究報告包含至少三個要素：

動機與目標：問題背景及動機、問題的重要性及其影響、具體(或量)研究目標；

文獻探討：相關研究方法及參考文獻、現有方法優勢及限制、本篇研究貢獻(區隔)；

本研究特色：研究方法及素材、主要研究發現（及其意涵）、本文適用時機及限制。

- 註：研究發現的價值除了創新之外，也希望能夠具有實質意涵及影響。

報告的格式要求

- 一篇完整報告至少包含以下幾項：

→標題 (Title)

→摘要(Summary或Abstract)

→報告主體。其中包括動機及背景、研究目的、資料來源與研究方法、分析結果說明、詮釋與結論

→參考文獻

→附錄（圖表、附件、...）

標題與摘要

- 建議標題另起一頁，與作者、日期等訊息獨立放在首頁。
- 摘要為一篇文章的濃縮，建議篇幅不多於一頁，以簡明扼要的原則闡述文章的撰寫動機（即背景）、目的、使用的資料及方法、大致的結論。

優化(Optimization)

- Optimization是許多資料分析的核心，無論對統計或機器學習領域都適用。

→統計分析與數值分析中的線性規劃很有關聯，傳統的資料分析多屬於線性，透過微分、線性代數等方法可以解決。

→如果屬於非線性分析，就需要更複雜的設計，而且不能保證找出正確的解答。

註：**Global** vs. **Local** Minimums。

→統計經常需要**求根**、**極值**。

什麼是優化？

- Optimization的目標是在給定限制條件下，透過系統化建模與演算法分析，使目標函數達到最大化(或最小化)的最佳解。
- 優化通常包含以下幾個要素：

→目標函數(objective function)；

→決策變數(decision variables)；

→限制條件(constraints)；

→最適化演算法(Algorithms, 如梯度下降)。

https://www.britannica.com/science/optimization?utm_source=chatgpt.com

優化的統計分析範例

- 統計分析中經常出現優化，常見的迴歸分析多半藉由(加權)最小平方法，尋找方程式的參數估計值。

→定義目標函數 $F = \sum_i (y_i - x_i'\beta)^2$ 。

→Constrained MLE也屬於優化問題，像是LASSO和Ridge Regression:

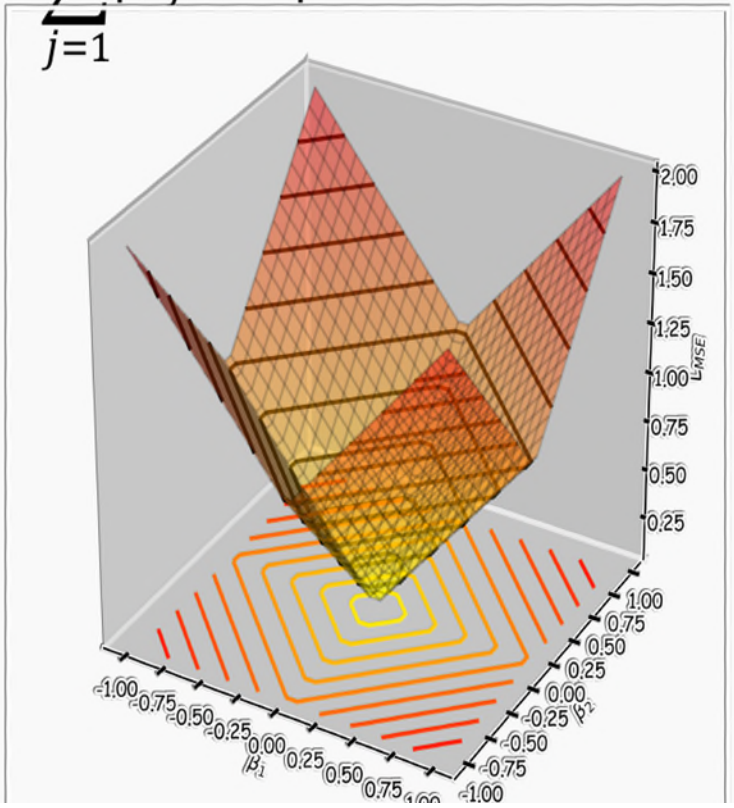
$$\hat{\beta} = (X'X + \lambda I)^{-1} X'Y$$

註：Lagrange multiplier(拉格朗日乘值法)？

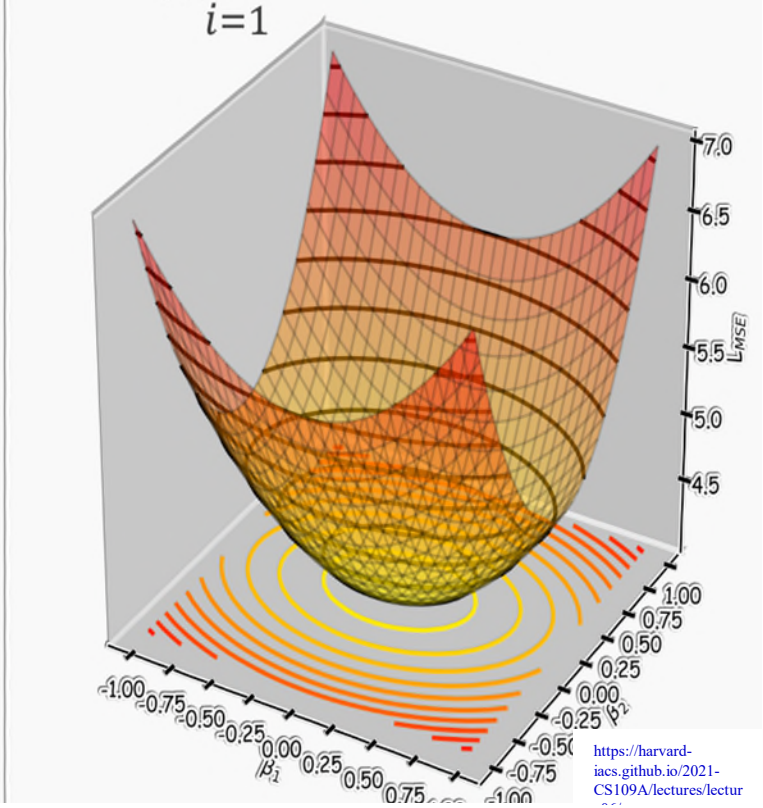
LASSO (least absolute shrinkage and selection operator)

$$L_{LASSO}(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n} \left| y_i - \boldsymbol{\beta}^T \boldsymbol{x} \right|^2 + \lambda \sum_{j=1}^{J} |\beta_j|$$

$$L_1 = \lambda \sum_{j=1}^{J} \left| \hat{\beta}_j^{LASSO} \right|$$



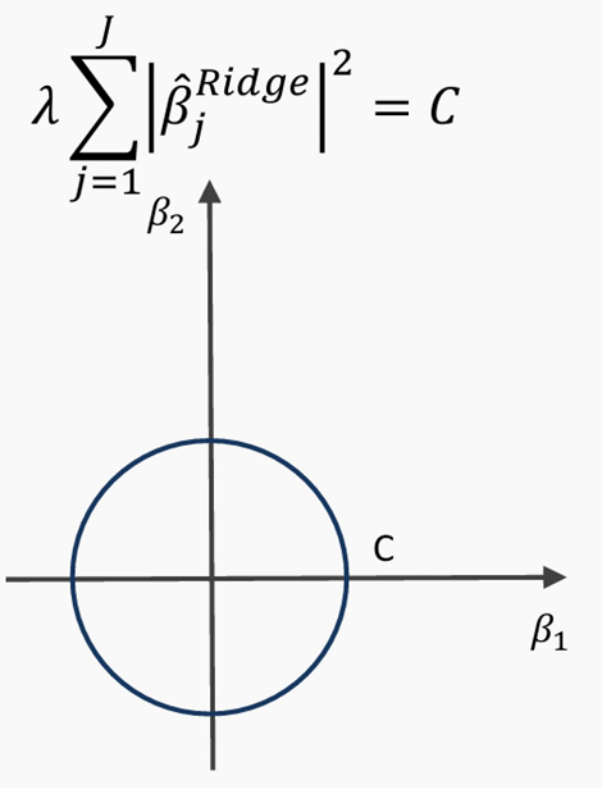
$$L_{MSE}(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n} \left| y_i - \boldsymbol{\beta}^T \boldsymbol{x} \right|^2$$

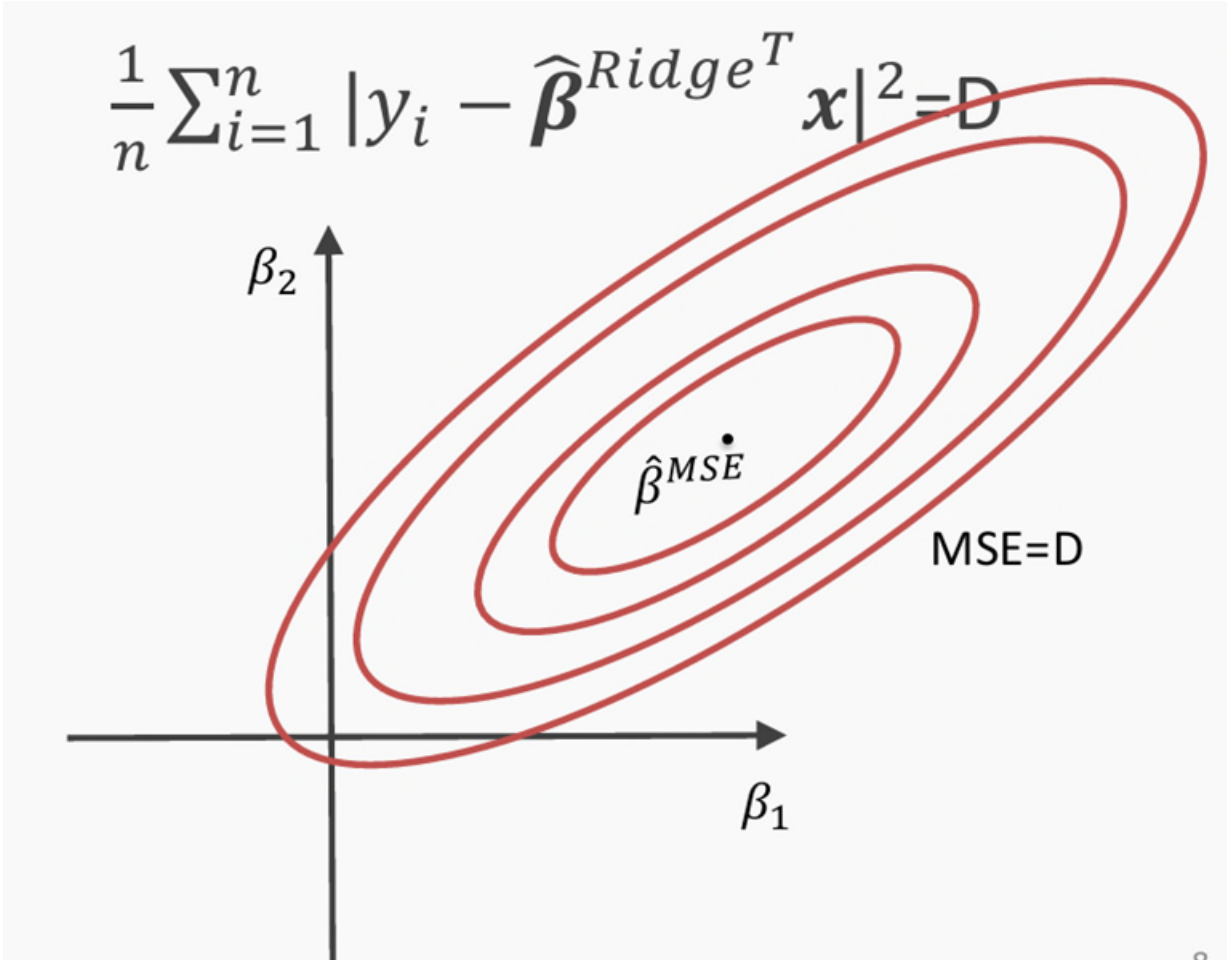


https://harvard-iacs.github.io/2021-CS109A/lectures/lectur

Ridge Regression

$$L_{Ridge}(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n} \left| y_i - \boldsymbol{\beta}^T \boldsymbol{x} \right|^2 + \lambda \sum_{j=1}^{J} (\beta_j)^2$$





優化與機器、深度學習

- **機器學習**的核心目標為尋找參數，使訓練資料代入模型的預測誤差最小化。

→羅吉斯迴歸透過MLE或最小化交叉熵損失(minimize cross-entropy loss)。

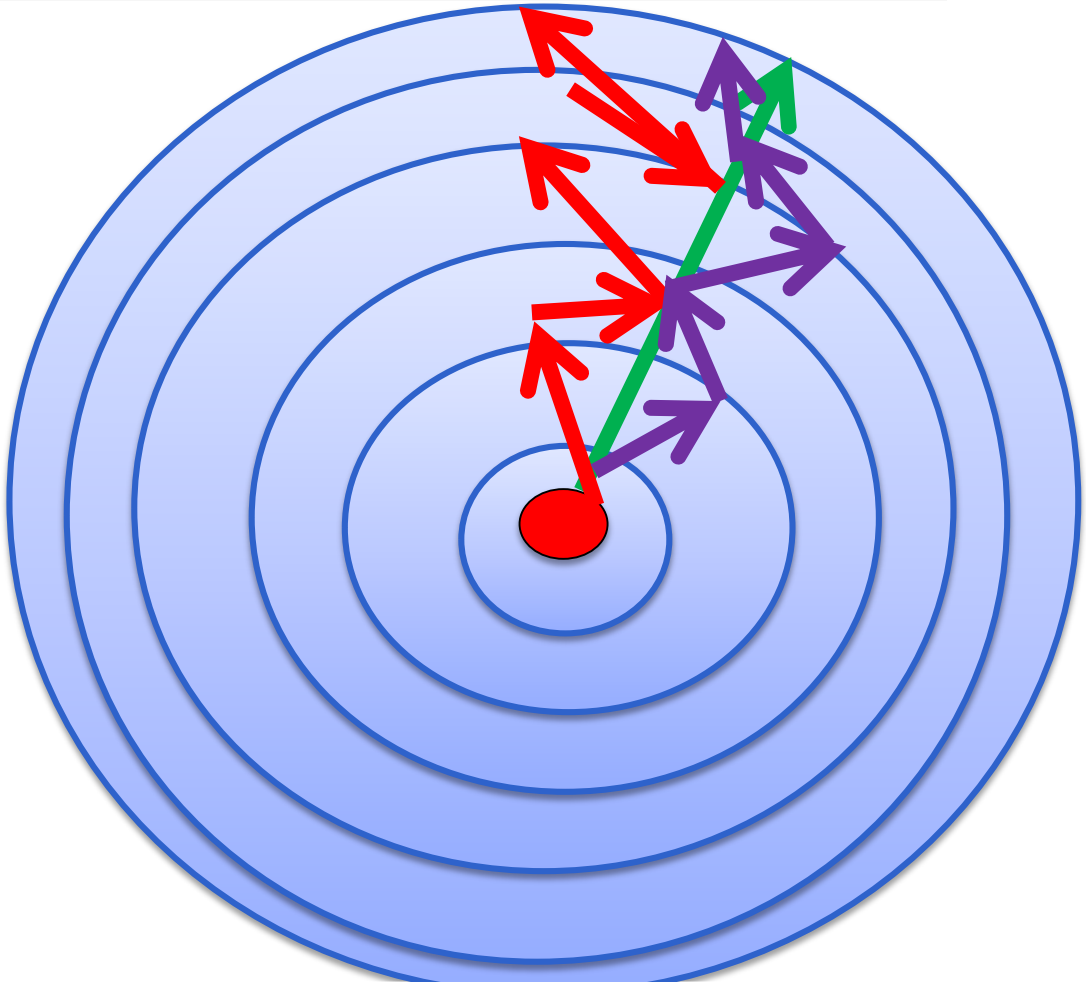
- **深度學習**同樣尋找最佳參數，但參數空間高維且非線性，損失函數可能是交叉熵、MSE 或自訂損失。

註：機器學習、深度學習模型的關鍵，往往在於適合的目標函數及語法。

6 steps of Process Optimization

1. Identify goals
2. Analyze current processes
3. Design process optimization
4. Test optimized processes
5. Implement process optimization
6. Evaluate and continuously improve

https://www.processmaker.com/blog/process-optimization-explained/



求根(Root Finding)

- We often need to find solution(s) of equation $f(x) = 0$.

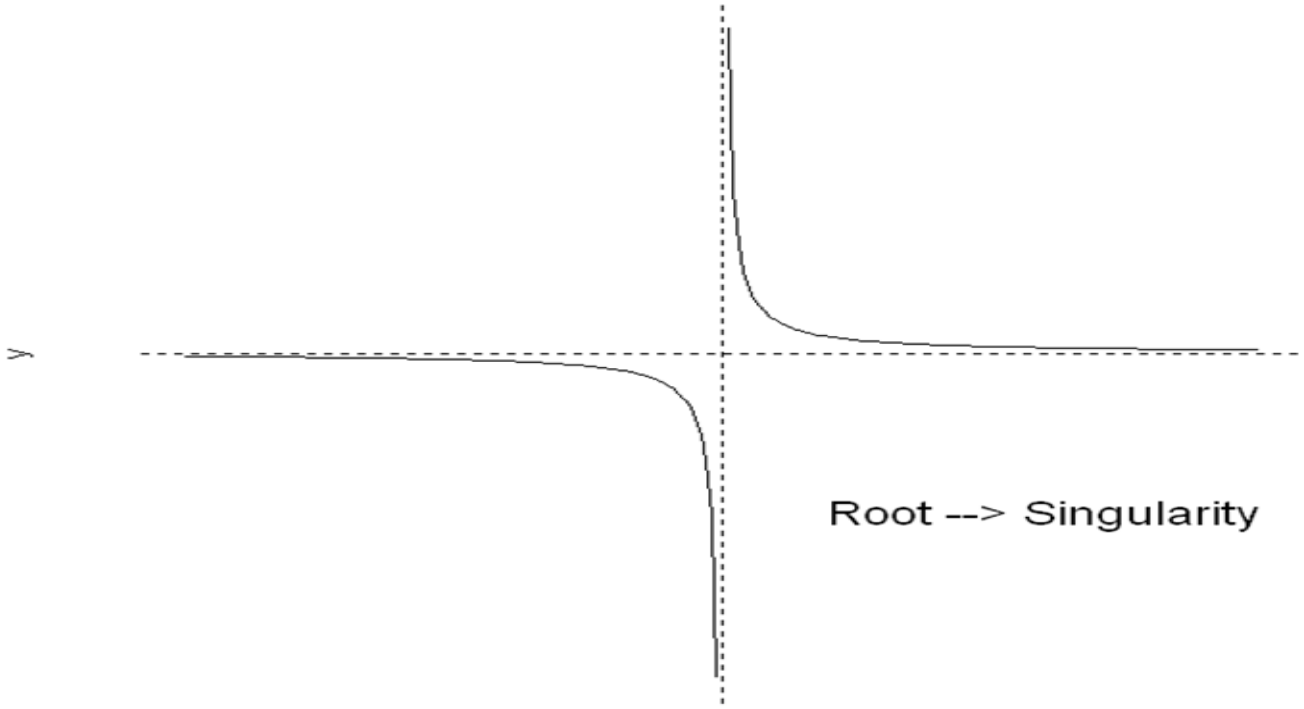
→ In one variable case, it is equivalent to finding the root(s) of a function.

→ Except in linear problems, root finding usually proceeds by iteration. A good algorithm will always converge. (We can determine in advance the rate of convergence of most algorithms.)

→ We will start with one-dim cases.

- Bracketing (括弧)

→ If *f(a)* and *f(b)* have opposite signs, and *f(x)* is continuous, then at least one root lying in *(a,b)*. In this case, we say that a root is *bracketed* in the interval *(a,b)*.



- Bisection Method (二分逼近法)

→ Once we know a root lying in an interval, several classical procedures are available (although most of them converge slowly). Bisection method is one of them and after each iteration the bounds containing the root decrease by a factor of two. That is, if ε_n is length of interval after n iterations,

$$\varepsilon_{n+1} = \varepsilon_n / 2.$$

In other words, if ε is the desire ending tolerance, we need $n = \log_2 \frac{\varepsilon_0}{\varepsilon}$ iterations.

- Notes:

1. Bisection will find at least a root if they exist, and it will converge on the singularity if there are no roots (e.g. *f(x) = 1/(x-c)*).

2. If $\varepsilon_{n+1} = \varepsilon_n / p_n \, (with \; p_n > 1)$ then it is said to converge linearly. On the other hand, if $\varepsilon_{n+1} = c \times (\varepsilon_n)^m, m > 1,$ it converges superlinearly.

3. Because of floating-point numbers, we shall expect the computed number is never zero. Need to set a reasonable tolerance.

- False Positions & Secant Methods (分割法)

→In general, false position and secant methods converges faster than bisection. Secant, in particularly, converges more rapidly near a root of a sufficiently continuous function with

$$\lim_{k\to\infty} |\varepsilon_{k+1}| \cong c \times |\varepsilon_k|^{1.618}$$ (golden ratio)

→Secant method: The root does not necessarily remain bracketed, since it retains the most recent of the prior estimate. (False position retains those with opposite sign.)

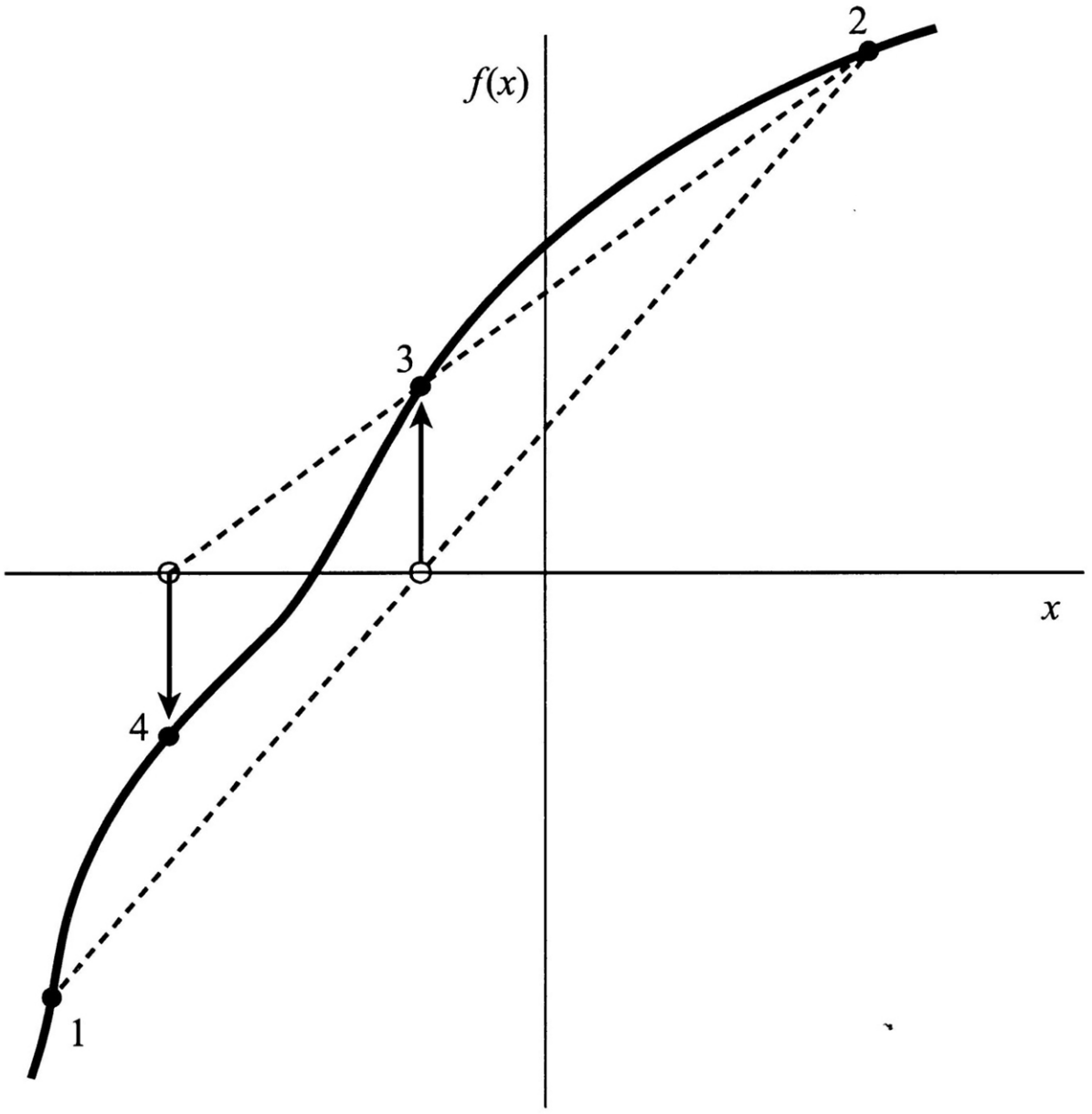


Figure 9.2.1. Secant method. Extrapolation or interpolation lines (dashed) are drawn through the two most recently evaluated points, whether or not they bracket the function. The points are numbered in the order that they are used.

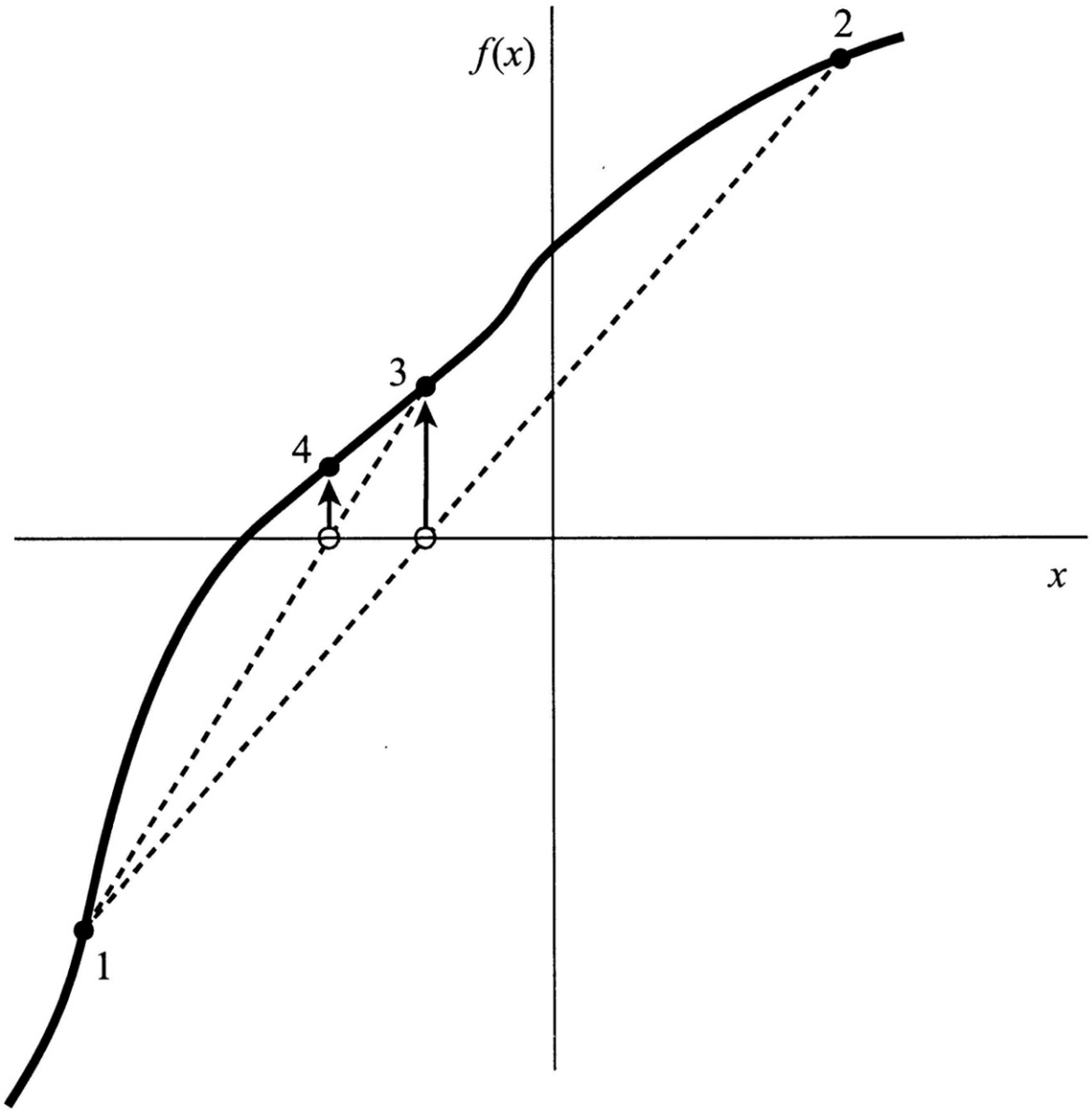


Figure 9.2.2. False position method. Interpolation lines (dashed) are drawn through the most recent points *that bracket the root*. In this example, point 1 thus remains "active" for many steps. False position converges less rapidly than the secant method, but it is more certain.

Notes:

→In general, the secant and false position methods converge much faster than bisection (although the false position usually converges linearly). But their convergence could be very slow if the function has some special structures.

→Unlike bisection, the starting points of the secant and false position methods need not bracket the root.

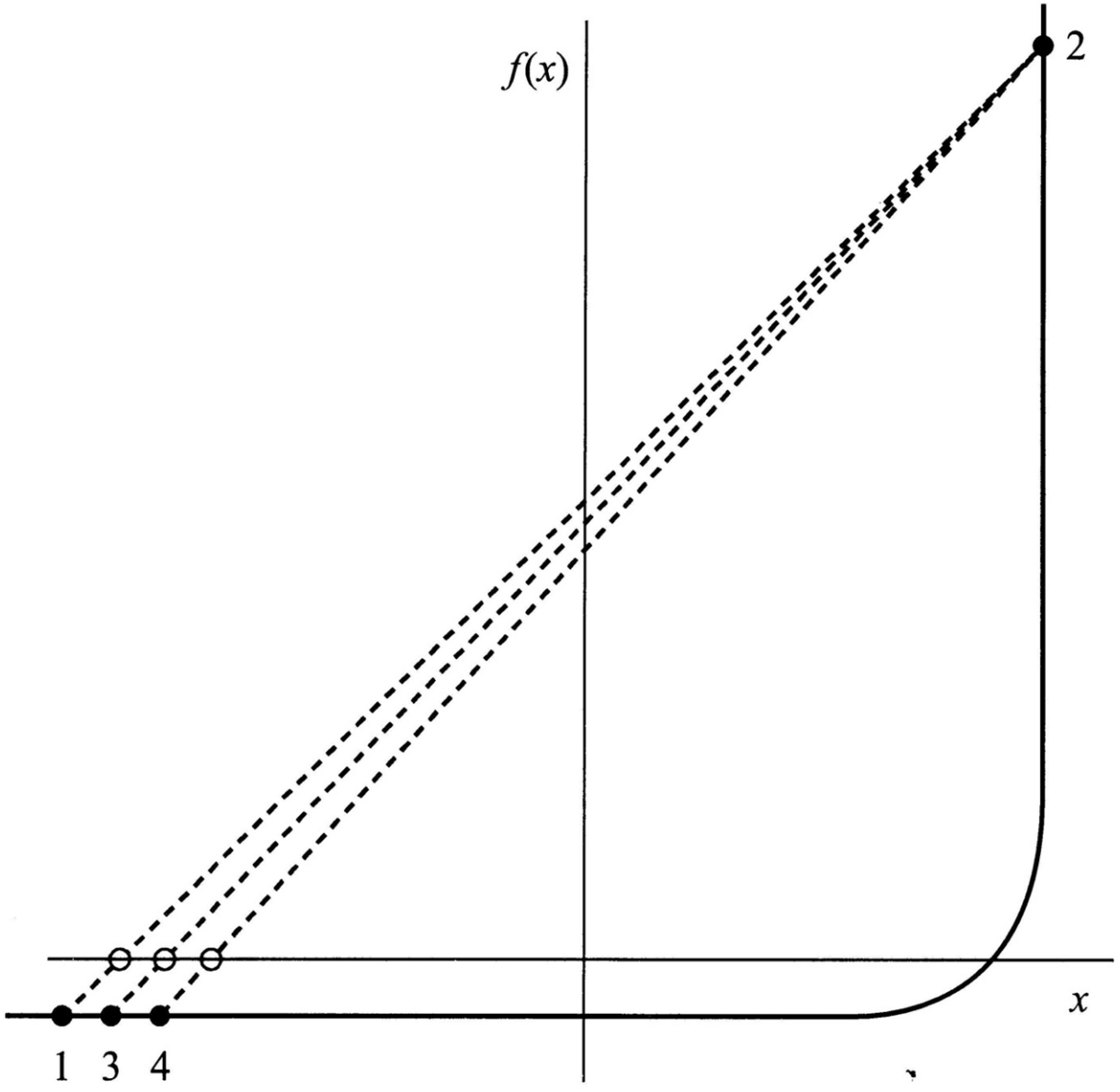


Figure 9.2.3. Example where both the secant and false position methods will take many iterations to arrive at the true root. This function would be difficult for many other root-finding methods.

- Ridder's Method:

→ A powerful variant of false position. Let x_1 and x_2 be two bracket points, and $x_3=(x_1+x_2)/2$ be the mid-point. The next point is solved by

$$f(x_1)-2f(x_3)e^{Q}+f(x_2)e^{2Q}=0$$

$$or\ \ e^{Q}=\frac{f(x_3)+sign[f(x_2)]\sqrt{f(x_1)-2f(x_3)+f(x_2)}}{f(x_2)}$$

In other words, x_4 is given by

$$x_4=x_3+(x_3-x_1)\frac{sign[f(x_1)-f(x_2)]f(x_3)}{\sqrt{f(x_3)^2-f(x_1)f(x_2)}}$$

Notes:

(1) x_4 is guaranteed to lie in the interval (x_1, x_2).

(2) The convergence is quadratic, or $m = 2$.

(3) In both reliability and speed, Ridder's method is generally competitive with more highly developed and better established methods (such as Brent method).

- Brent's Method:

→ Brent's method combines the sureness of bisection with the speed of a higher-order method when appropriate. (It combines root bracketing, bisection, and inverse quadratic interpolation.

→Brent's method is recommended for general one-dimensional root finding where a function's values only (and not its derivative or functional form) are available.

Note: Due to its complex form, we will not show the algorithm of Brent's method.

- Newton-Raphson Method: (using derivative)

→ It is also called Newton's method, perhaps the most celebrated of all one-dimensional root-finding routines.

→ The idea is via Taylor's series expansion:

$$f(x+\delta) \approx f(x) + f'(x)\delta$$

If $f(x+\delta) = 0$ ➔ $\delta = -\dfrac{f(x)}{f'(x)}$.

In other words, $x_{n+1} = x_n - \dfrac{f(x_n)}{f'(x_n)}$.

Notes:

1. Far from a root, the Newton-Raphson formula can give grossly inaccurate, meaningless corrections.
2. By the Newton-Raphson formula

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \Rightarrow \varepsilon_{n+1} = \varepsilon_n - \frac{f(x_n)}{f'(x_n)},$$

$$Or,\ \ \varepsilon_{n+1} = -\varepsilon_n^2 \cdot \frac{f''(x_n)}{2f'(x_n)}.$$

i.e., quadratic convergence.

Notes: (continued)

3. It is very common to “polish up” a root with one or two steps of Newton-Raphson, which can multiply by two or four its number of significant figures.
4. The Newton-Raphson formula can be used to multiple dimensions as well.
5. The Newton-Raphson formula also uses the derivative of a function.

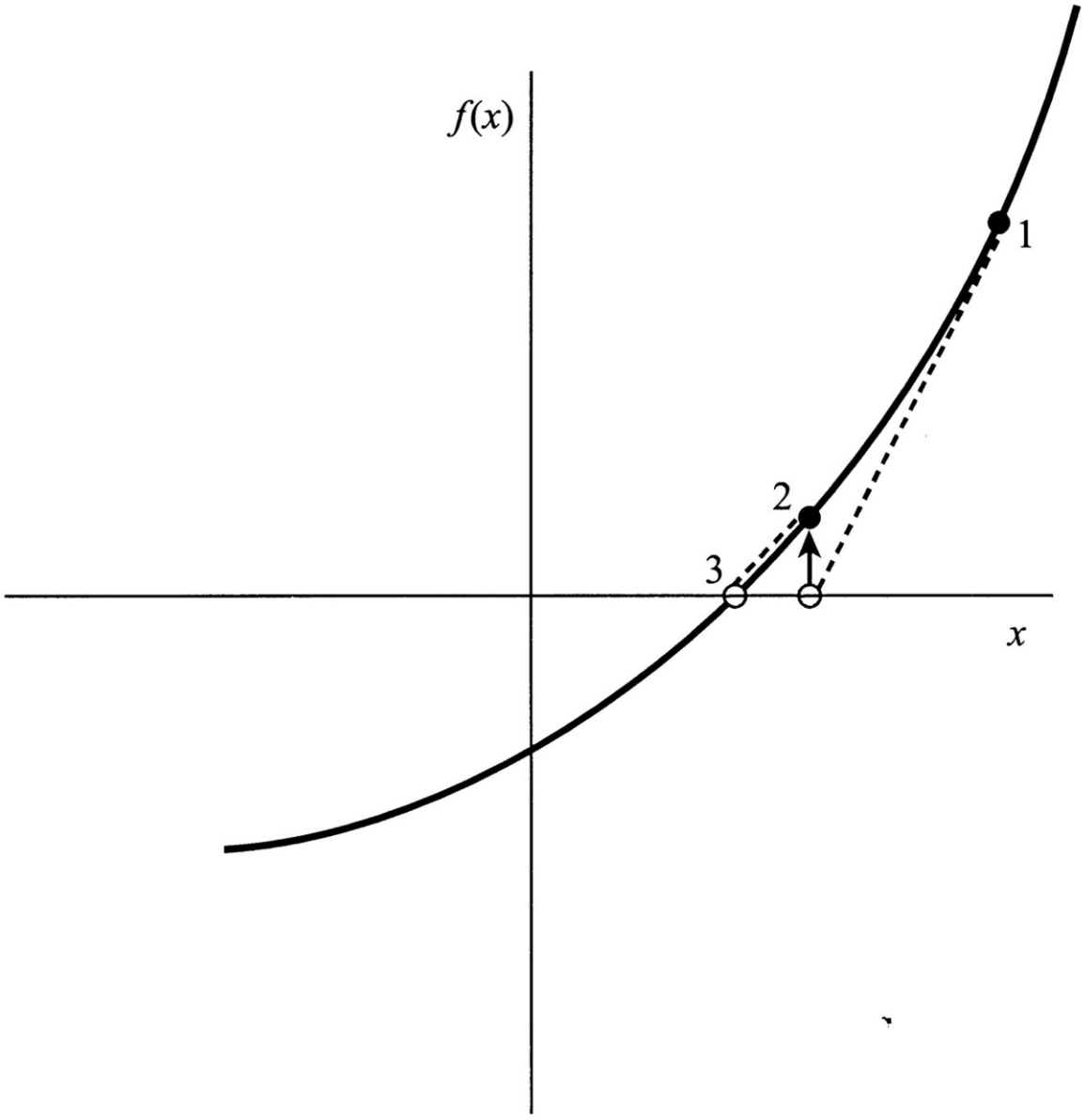


Figure 9.4.1. Newton's method extrapolates the local derivative to find the next estimate of the root. In this example it works well and converges quadratically.

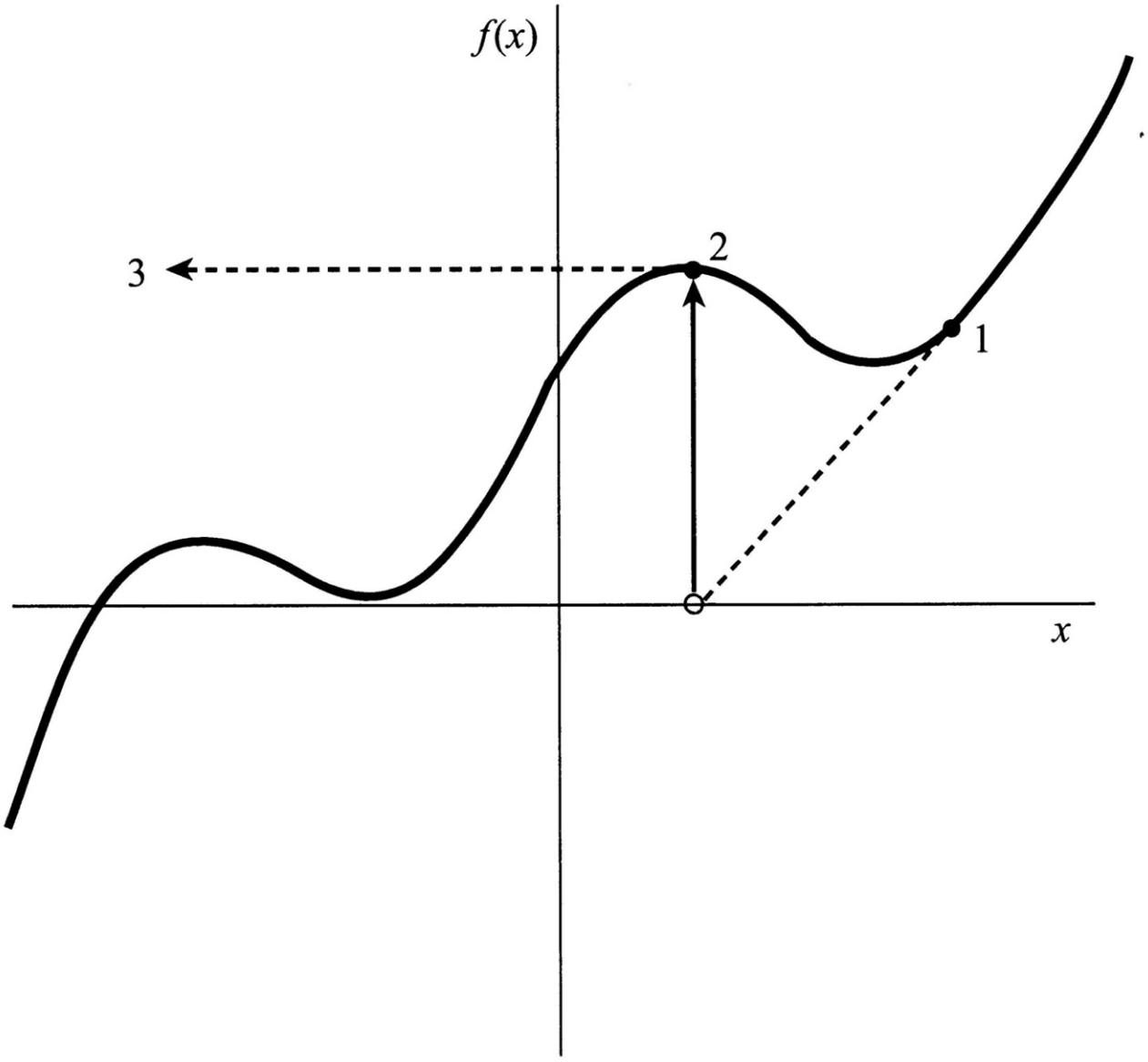


Figure 9.4.2. Unfortunate case where Newton's method encounters a local extremum and shoots off to outer space. Here bracketing bounds, as in `rtsafe`, would save the day.

- Roots of Polynomials

→ The effort of seeking roots of a polynomial can be reduced by the use of *deflation*, i.e., $P(x) = (x-r)Q(x)$, since the roots of $Q(x)$ are exactly the remaining roots of $P(x)$. Two types of deflation: *Forward deflation* and *Backward deflation*.

→ Because each root is known with only finite accuracy, errors creep into the determination of the coefficients of the successively deflated polynomial.

- Roots of Polynomials (Continued)

There are some good methods can be used to solve the roots of a polynomial.

→Muller's method (like Secant's method)

→Laguerre's method (very complex method)

→Eigen-value method (form a matrix)

Note: In general, it is not recommended that we find the roots of a polynomial one by one, due to the rounding errors.

- Example 1. Find the root of $f(x) = x^2 - 2 = 0$.

→ Initial values (1,2), i.e., looking for the root $\sqrt{2}$ and convergence criterion ($|x_{i+1} - x_i| <$)10^{-6} is used. We shall compare the bisection, false position, and secant methods.

Method	Bisection	False	Secant
iterations	20	9	6

- Notes:

(1) There are 5 iterations if Newton's method is used with starting point 1.

(2) If we use the command "nlminb" in R by setting the objective function as $(x^2-2)^2$, then there are 7 iterations with starting point 1. (You could use "uniroot", "polyroot", "nlm" or "optim" in R.)

(3) Don't turn an optimization problem into a root-finding problem, or vice versa. It is likely to be inefficient.

- Example 1(continued).

  Let $f(x) = x^3 - 3x^2 + 3x - 3$. We also compare the bisection, false position, and secant methods. The convergence criterion is 10^{-6} and the starting points are (2,3).

Method	Bisection	False	Secant
iterations	20	15	7

Note: The secant and false position methods don't need to have the root(s) bracketed; 5 and 7 iterations if Newton's method and *nlminb* are used with starting point 2.

- Example 2. Maximum likelihood (Poisson Regression, Thisted and Efron, 1986)

j =	0	1	2	3	4	5	6	7	8	9
M_j	40	10	21	16	11	7	4	4	2	3
X_j	30.11	13.91	10.63	8.68	7.14	5.99	5.23	4.37	4.09	3.54

→The likelihood function is defined as

$$l(\theta) = -n\alpha + \sum M_j \log(\alpha + \theta x_j) - \sum \log(M_j!).$$

→ The MLE can be done via letting $l'(\theta) = 0$,

i.e., $l'(\theta) = \sum \frac{M_j x_j}{\alpha + \theta x_j}$. (Note: $\alpha = \overline{M}$.)

Using the secant and Newton-Raphson (derivative) methods to solve θ, starting from $\hat{\theta}_0 = 1.0$ and setting convergence criterion 10^{-6}, Newton-Rapson converges in four iterations and the secant method in six, with

$$(\hat{\theta}, l(\hat{\theta})) = (1.5062598, -36.300727).$$

- Solving $f(x) = 0$: The vector case

→ When we deal with the vector case, i.e., several parameters must be estimated simultaneously.

→ The method of using derivative, such as Newton's method, is easily to extend. For example, the formula of iteration becomes

$$x_{n+1} = x_n - [\nabla f(x_n)]^{-1} f(x_n),$$

which is again derived via Taylor's expansion.

- Example 3. Binomial/Poisson Mixture

  The following data are believed to be drawn from a mixture of Binomial/Poisson dist.

$$X \sim \begin{cases} 0 & with\ prob.\ \xi \\ Poisson(\lambda) & with\ prob.\ 1-\xi \end{cases}$$

X	0	1	2	3	4	5	6
Count	3062	587	284	103	33	4	2

→ First, write down the log-likelihood:

$$l(\xi,\lambda) = n_0 \log(\xi + (1-\xi)e^{-\lambda})$$

$$+ (N - n_0)[\log(1-\xi) - \lambda] + \sum_{i=1}^{\infty} i\, n_i \log\lambda.$$

Then, we need to find the derivative of $l(\xi,\lambda)$ as the function $f(x)$, and $\nabla f(x)$ as well.

The initial guess for ζ is $n_0/N \approx 0.75$ and for λ is $\overline{X} \approx 0.40$. The convergence criterion is 10^{-6} measured in the L_1 norm. Using the Newton method, the process stops after 6 iterations.

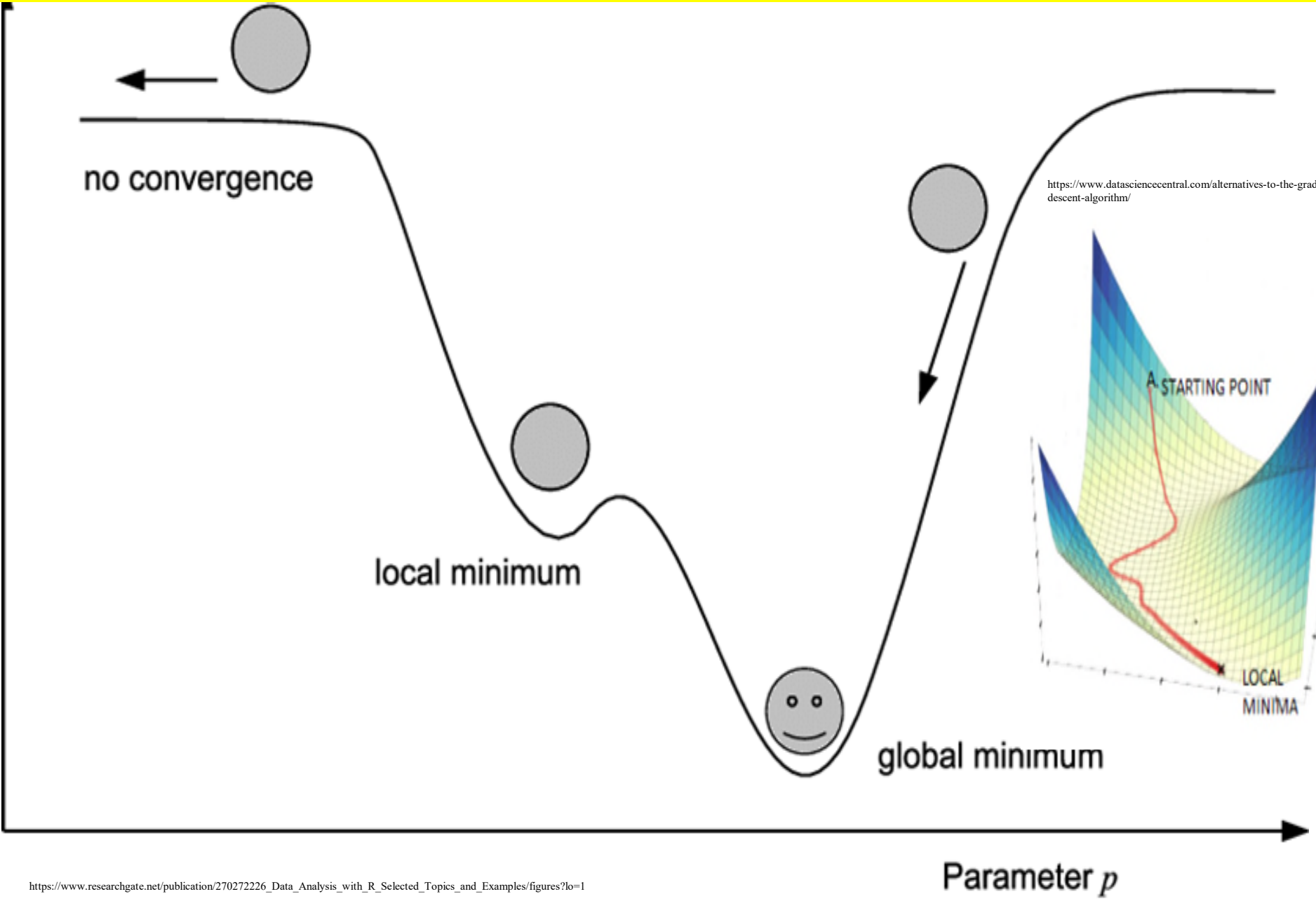
- References:

→ Numerical Recipes in C (Chapters 9 &10)
(http://www.nrbook.com/a/bookcpdf.php)

→ Numerical Methods of Statistics (Monahan)
(Chapters 8 & 9)

→ Elements of Statistical Computing (Thisted)
(Chapter 4)

優化的難度更高！



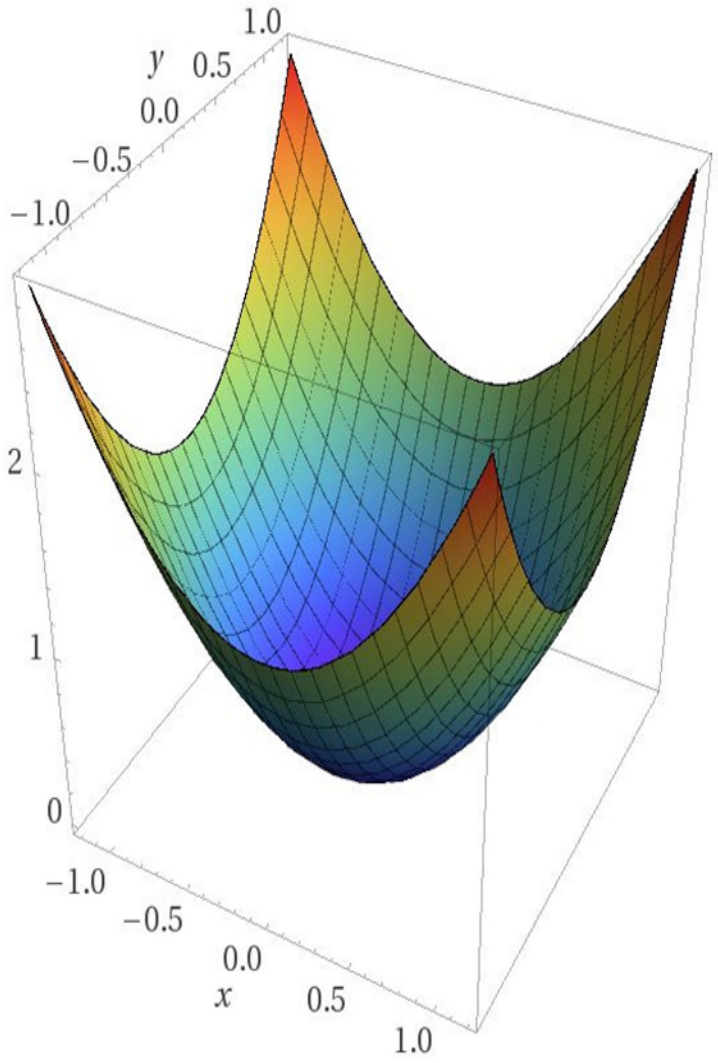
https://www.datasciencecentral.com/alternatives-to-the-gradient-descent-algorithm/

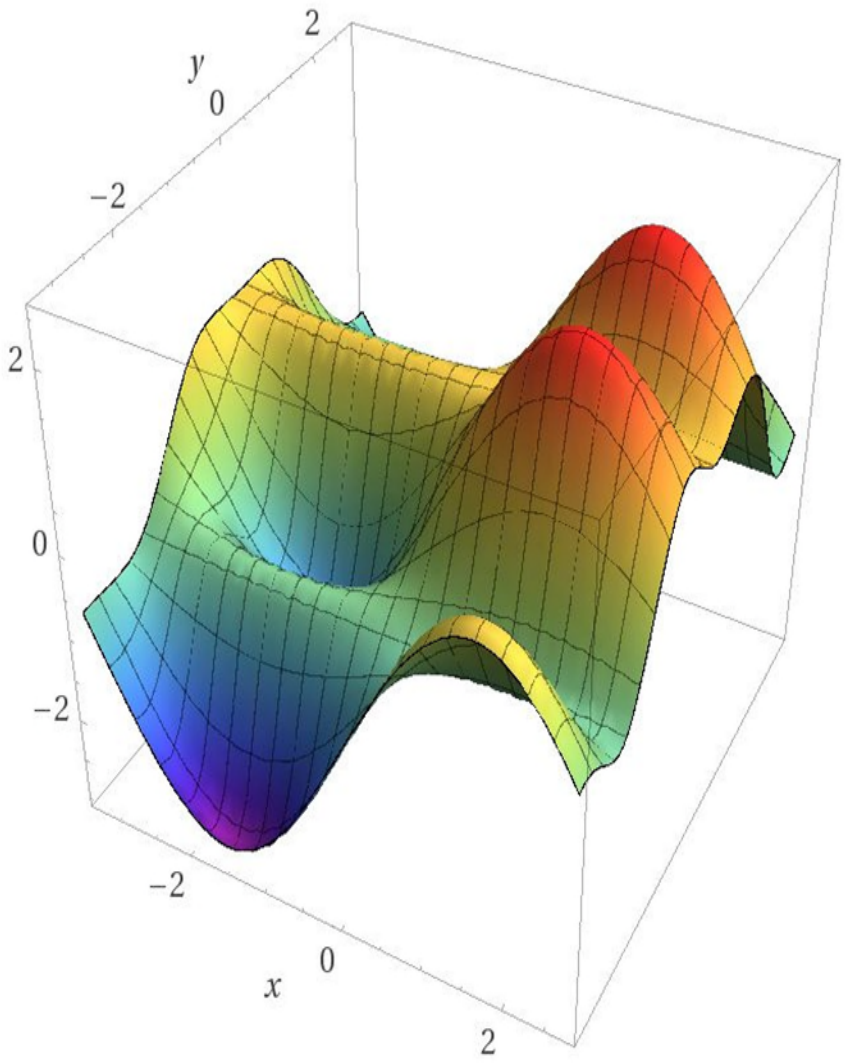
https://www.researchgate.net/publication/270272226_Data_Analysis_with_R_Selected_Topics_and_Examples/figures?lo=1

Minimization (and Maximization)

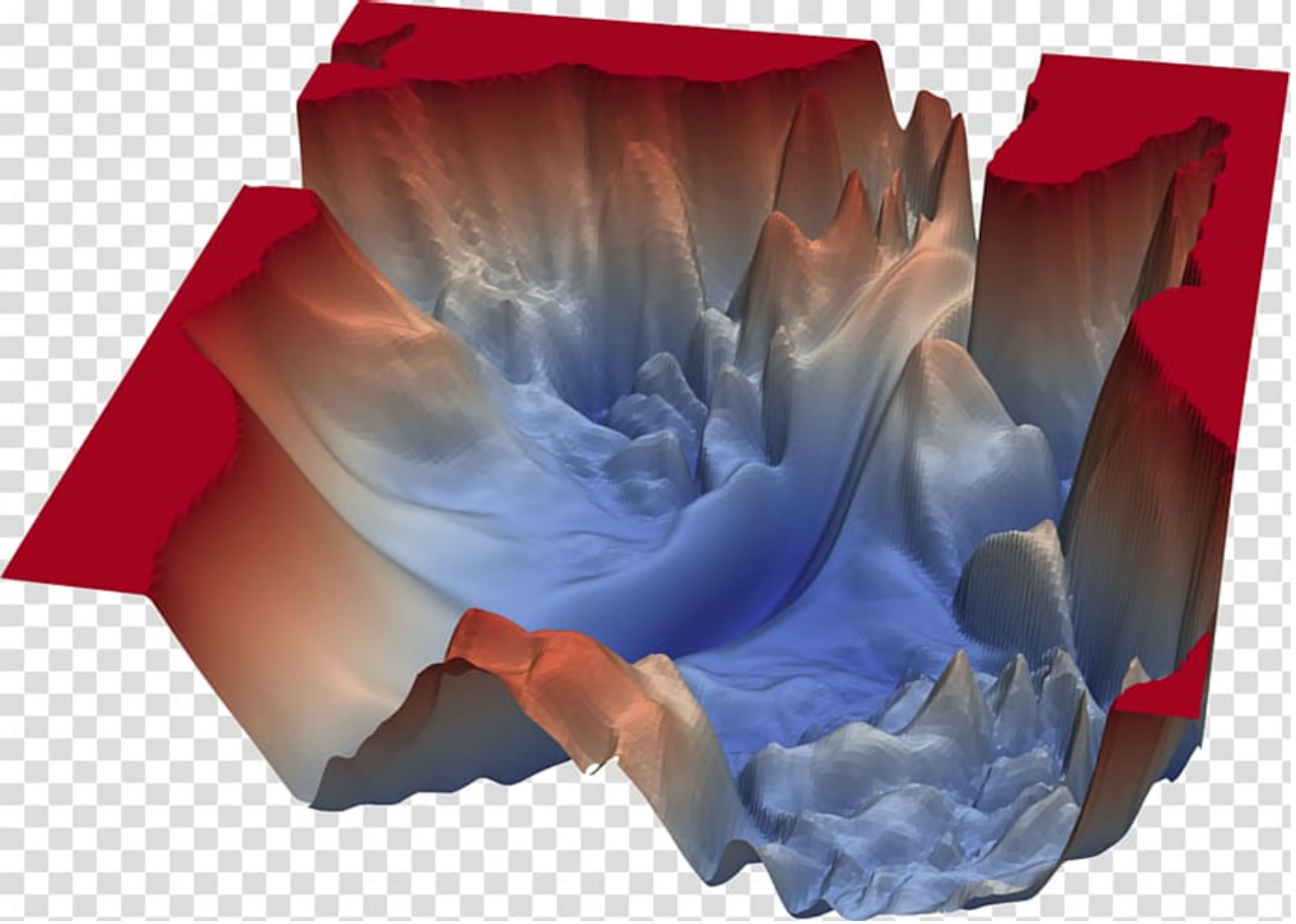
- It is, also known as ***Optimization***, searching for extremums (could be global or local) of a function.
- The applications include in Economics and Engineers (e.g. constrained optimization), in Operational Research (e.g. Linear Programming).

三維以上空間的極值很難目視判斷！





https://blog.paperspace.com/content/images/size/w2000/2018/05/convex_cost_function.jpg



https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.hiclipart.com%2Ffree-transparent-background-png-clipart-fdiju&psig=AOvVaw1yUASEX9e0LTmPCnYxMvZv&ust=1586878669360000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCMD33Yre5egCFQAAAAAdAAAAABAf

Note: If we want to find the extreme values of function f and f is differentiable, could we use the first derivative of f, i.e., $f' = 0$?

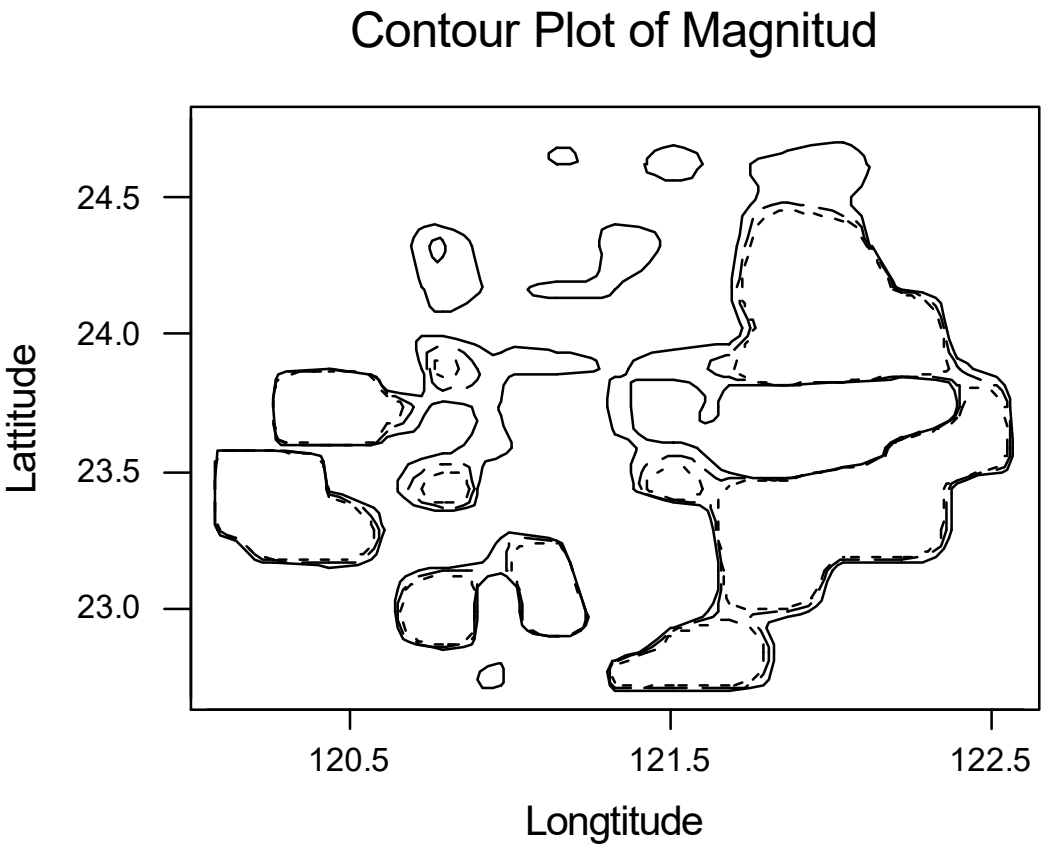
→ For example, given $x, y > 0$, find the minimum of $f(x,y) = xy + 1/xy$. The first derivative $\frac{\partial f}{\partial x} = \frac{\partial f}{\partial y} = 0$ leads to $xy = 1$.

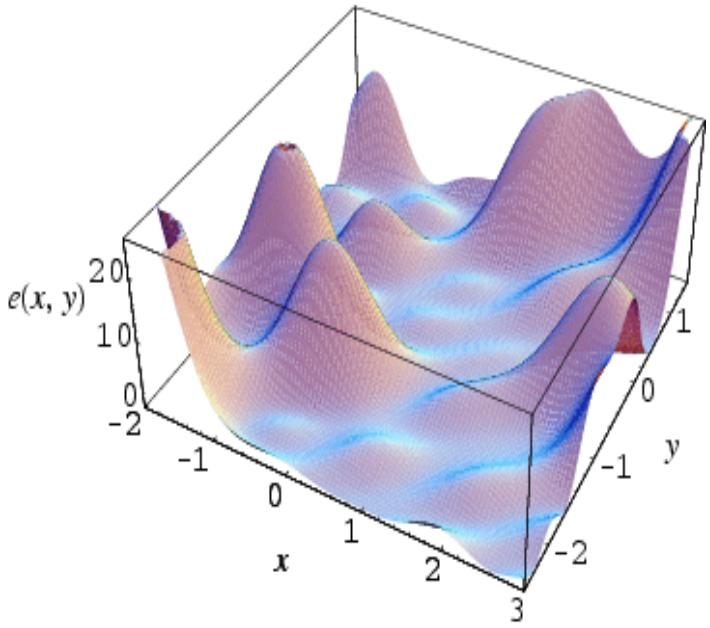
But the Hessian matrix does not show if $xy = 1$ is the minimum, since

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix} \Rightarrow \det(H) = 0.$$

- Grid Search:

→A lattice of points is laid out and F is then evaluated at each point on the lattice.





→ Every lattice can be expressed in the following form: $\{x \mid x = x_0 + Ma, a \in Z^p\}$, where x_0 is a given fixed point and M is a $p \times p$ matrix with linearly independent columns.

→ In higher-dimensional ($p > 2$) problems the full grid search is impractical, but it is impossible to search in one direction at a time by finding the minimum values. Then do it for different initial values (i.e. sequential search) and could use them as starting values of other methods.

- Golden Section Search (one-dim):

→Analogy to bisection, we can also "bracket" a minimum. The idea is that if $a < b < c$ such that $f(b) < f(a)$ and $f(b) < f(c)$, then the next step is to use (a,b,x) if $f(b) < f(x)$, or (b,x,c) if $f(b) > f(x)$. Continue this process until $(1-\varepsilon)b < b < (1+\varepsilon)b$, if b is a minimum.

→ In general, $f(x) \cong f(b) + \frac{1}{2} f''(b)(x-b)^2$,

$$\Rightarrow |x-b| < \sqrt{\varepsilon}\,|b| \sqrt{\frac{2|f(b)|}{b^2 f''(b)}}$$

We only need to ask for $\sqrt{\varepsilon}$ of precision.

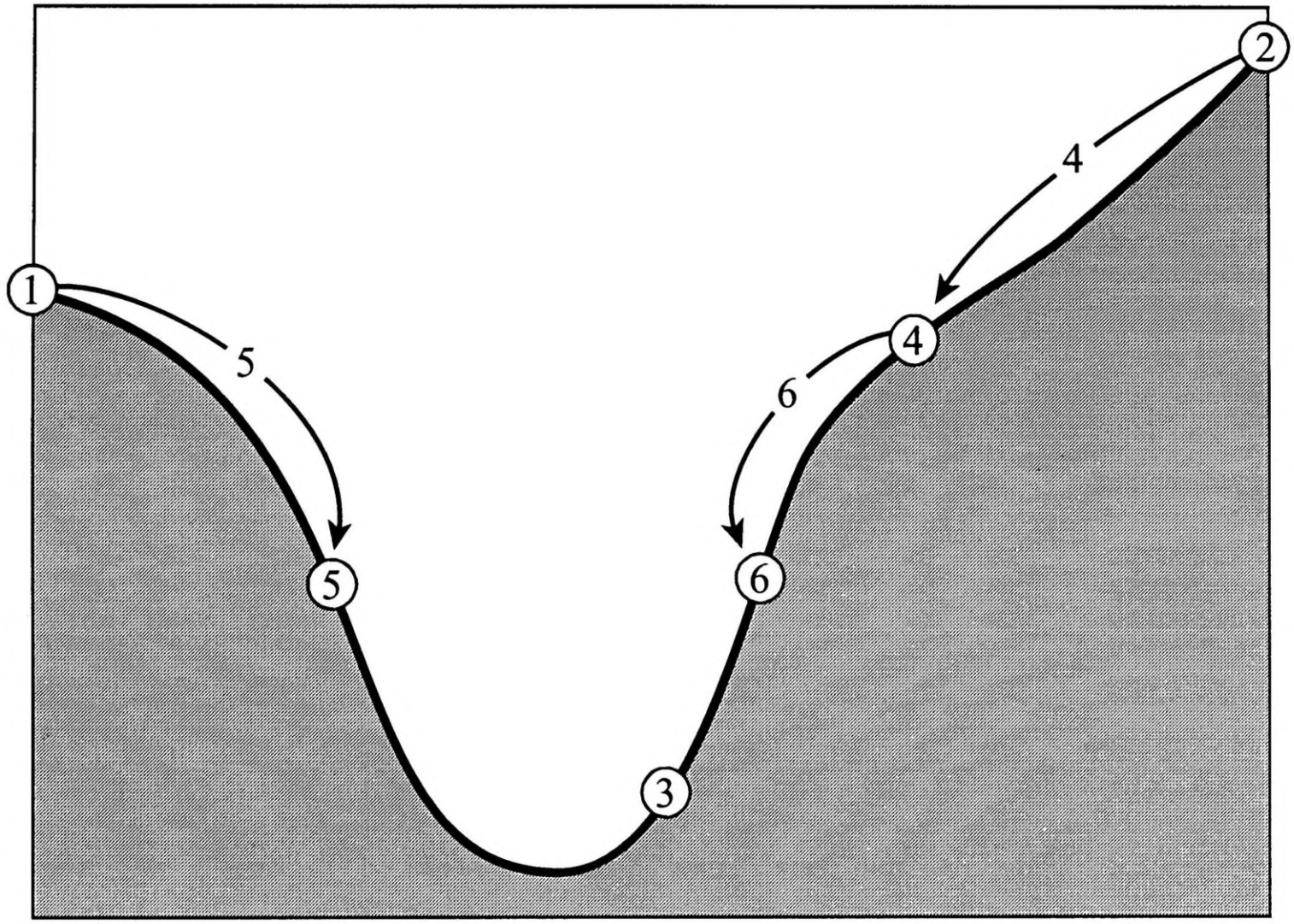


Figure 10.1.1. Successive bracketing of a minimum. The minimum is originally bracketed by points 1,3,2. The function is evaluated at 4, which replaces 2; then at 5, which replaces 1; then at 6, which replaces 4. The rule at each stage is to keep a center point that is lower than the two outside points. After the steps shown, the minimum is bracketed by points 5,3,6.

- Parabolic Interpolation and Brents' Method

→Golden section is similar to bisection, and we shall introduce a method similar to the secant method.

→Given a "bracket," i.e., $a < b < c$, use a local polynomial (such as a parabola) to find the next iteration point, or

$$x = b - \frac{1}{2} \cdot \frac{(b-a)^2[f(b)-f(c)] - (b-c)^2[f(b)-f(a)]}{(b-a)[f(b)-f(c)] - (b-c)[f(b)-f(a)]}.$$

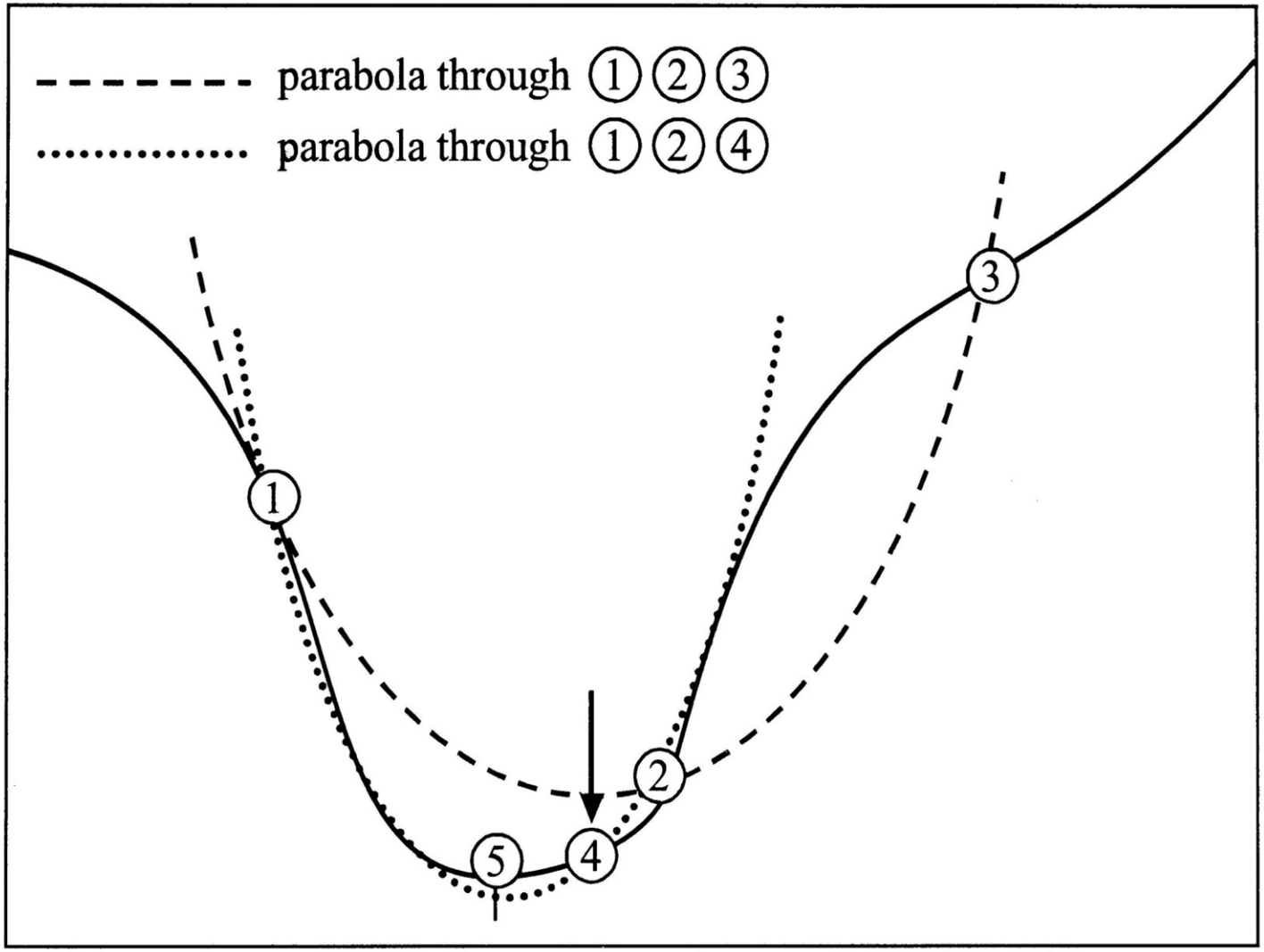


Figure 10.2.1. Convergence to a minimum by inverse parabolic interpolation. A parabola (dashed line) is drawn through the three original points 1,2,3 on the given function (solid line). The function is evaluated at the parabola's minimum, 4, which replaces point 3. A new parabola (dotted line) is drawn through points 1,4,2. The minimum of this parabola is at 5, which is close to the minimum of the function.

→The above formula fails only if the three points are collinear.

→It is also possible to adapt the idea of Brents' method, but six (not necessarily all distinct) functional points (three in the previous case) are required to find the next points. See "*Numerical Recipes in Fortran*" for detailed program code.

- One-dim. Search with first Derivatives

→The function's first derivative can be used to detect extreme values, but we need to distinguish maxima from minima.

→We don't want to give up our strategy of maintaining a rigorous bracket on the minimum at all times. The only way to keep such a bracket is to update it using function (not derivative) information, with the central point in the bracketing triplet always that with the lowest function value.

→ But we don't want to use the root of the first derivative directly. Instead, the first derivative is used to assess the next iteration point. The sign of the derivative at the central point of the bracketing triplet *(a,b,c)* indicates uniquely whether the next point should be taken in *(a,b)* or *(b,c)*.

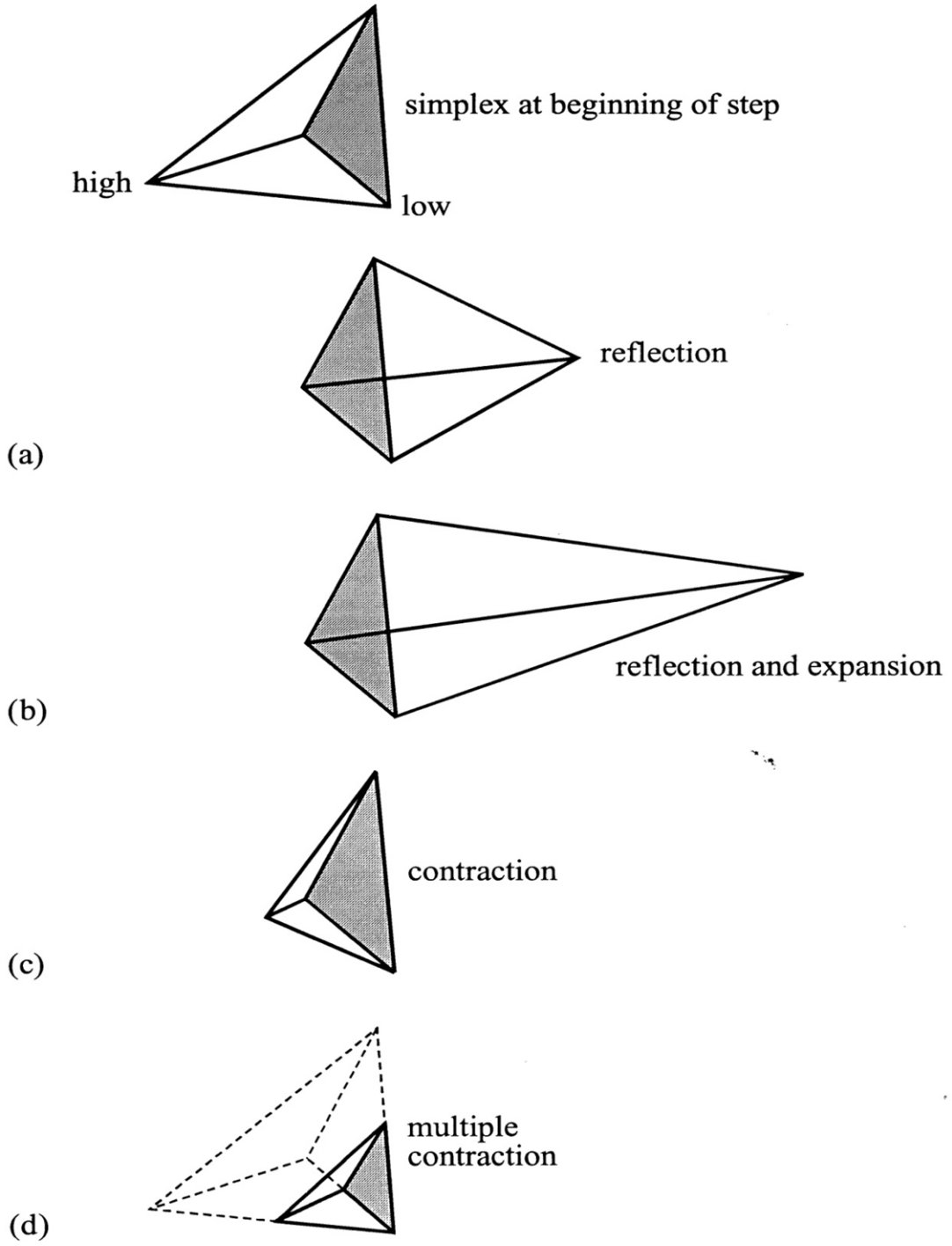
→ The value of this derivative and of the derivative at the second-best-so-far point are extrapolated to zero by the secant method, which is superlinear of order 1.618.

- Downhill Simplex Method (Multi-dim)

→The downhill simplex method requires only function of evaluations, not derivatives.

→It is not very efficient but it may frequently be the best method to use if the figure of merit is "getting something working quickly" for a problem whose computational burden is small.

→A *complex* is the geometrical figure consisting of N+1 points (in N dimensions) and all their inter connecting line segments, polygonal faces, etc.


simplex at beginning of step
high
low
reflection
(a)
reflection and expansion
(b)
contraction
(c)
multiple
contraction
(d)

→ The downhill simplex method must be started not just with a single point, but with all N+1 points on initial simplex. Then takes a series of steps, mostly moving the largest to the opposite face of the simplex to a lower point.

→Termination criteria is different. It can be terminated if the vector distance moved in that step is fractionally smaller in magnitude than some tolerance.

- Methods for using derivatives

→ Newton's steps:

$$d_N = -[f''(x)]^{-1}[f'(x)]$$

→ Steepest descent:

$$d_S = -f'(x)$$

→ Levenberg-Marquardt adjustment

$$d_{LM} = -[f''(x) + \lambda I]^{-1}[f'(x)]$$

→Conjugate-gradient methods

→Other methods

- Example 4. Given $x > 0$, find the minimum of $f(x) = x + 1/x$.

→ In R/S-Plus, we use "nlminb" to find the minimum of $f(x)$.

```
f=function(x)  { x+1/x }
nlminb(start=2,obj=f,lower=0)
```

The minimum 2 is attained at $x = 1$, with 7 iterations. (Larger starting values, more iterations?)

Starting	2	5	8	10	20	50	100
Iterations	7	9	11	10	10	13	16

Note: "nlm" and "optim" can also be used.

- Example 4. (continued) Given $x, y > 0$, find the minimum of $f(xy) = xy + 1/xy$.

→ Similarly, we use the command "nlminb."

The minimum 2 is attained at $x = y = 1$. The following table shows the number of iterations needed for different starting points.

Starting	2,2	5,5	8,8	10, 10	20, 20	50, 50	100, 100
Iterations	7	10	12	10	14	13	14

Note: In other words, we can see that the starting points do have some influences on the number of iterations. If we can use grid search or bisection method to narrow down the range of minimum, then the iterations required in either linear or superlinear search can be thus reduced.

Question: What happen if we use methods other than Newton's method?

- Example 5. We want to calculate the death rate from the central death rate (or vice versa). If the mortality follows the distribution of uniform death, these two terms satisfy

$$q_x = \frac{m_x}{1 + m_x / 2}$$

where $q_x = \frac{d_x}{l_x}$ & $m_x = \frac{d_x}{\int_0^1 l_{x+t} dt} = \frac{d_x}{L_x}$.

If the survivors follow harmonic assumption,

$$\frac{1}{l_{x+t}} = \frac{1-t}{l_x} + \frac{t}{l_{x+1}} \Rightarrow l_{x+t} = \frac{l_x \, l_{x+1}}{t \, l_x + (1-t) l_{x+1}}$$

If we plug this formula into Lx,

$$L_x = \int_0^1 l_{x+t}\,dt = \int_0^1 \frac{l_x l_{x+1}}{l_{x+1} + t(l_x - l_{x+1})}\,dt$$

$$= l_x \int_0^1 \frac{p_x}{p_x + tq_x}\,dt = l_x \int_0^1 \frac{1}{1 + tq_x / p_x}\,dt$$

$$= l_x \cdot (p_x / q_x) \cdot \log(1 + q_x / p_x)$$

Therefore, $m_x = \dfrac{d_x}{L_x} = \dfrac{q_x}{(p_x / q_x) \cdot \log(1/p_x)}$

$$= \frac{(q_x)^2}{-(1-q_x) \cdot \log(1-q_x)}$$

We can use the optimization to solve for the value of q_x, given m_x. In specific, we use the function "nlminb" in R to find the value.

*ff=function(a) {(-a^2/((1-a)*log(1-a))-0.05)^2}*

nlminb(start=0.05, obj=ff)

It takes 3 iterations to reach the value .04876067. You can also use the bisection method manually to reach similar answer. Note that the Taylor expansion is another possible method, but the Answer is slightly different (0.04872).

- Example 6. We use Japanese and Swedish elderly data (both sets are age-specific data, collected in the time period: 1980-1990) in Kannisto (1994) to check the Gompertz assumption, use the weighted least squares to find the estimates of parameters.

Note:

$$\mu_x = BC^x \;\&\; {}_tp_x = e^{-\int_0^1 \mu_{x+t}dt}$$

$$\Rightarrow {}_5p_x = -BC^x(C^5-1)/\log C$$

$$or\ \log p_{x+1}/\log p_x = C.$$

→ In other words, we want to minimize

$$\underset{B,C}{Min} \sum_{x} {}_5l_x ({}_5p_x + BC^x (C^5 - 1)/\log C)^2.$$

Of course, we can also apply least squares to $log\, p_x$ directly, since $\log p_{x+1} / \log p_x = C.$

Similarly, we can use Maximum likelihood to find the values of B and C as well.

Question: Which approach is the best?

Simulated loss of B for Gompertz Law (unit: 1)

Method: Weights		Japan		Sweden	
		Male	Female	Male	Female
WLS	n_x	.000485	.000309	.002884	.001741
	$\sqrt{n_x}$	.000928	.000455	.003998	.002090
NM	n_x	.000459	.000295	.002570	.001711
	$\sqrt{n_x}$	.005189	.003433	.015640	.012701
MLE	1	.000418	.000248	.002498	.001419

Simulated loss of C for Gompertz Law (unit: .001)

Weights		Japan		Sweden	
		Male	Female	Male	Female
WLS	n_x	.000146	.000093	.000859	.000515
	$\sqrt{n_x}$	.000280	.000138	.001168	.000616
NM	n_x	.000138	.000088	.000759	.000497
	$\sqrt{n_x}$	.001465	.000950	.004239	.003402
MLE	1	.000126	.000074	.000729	.000416

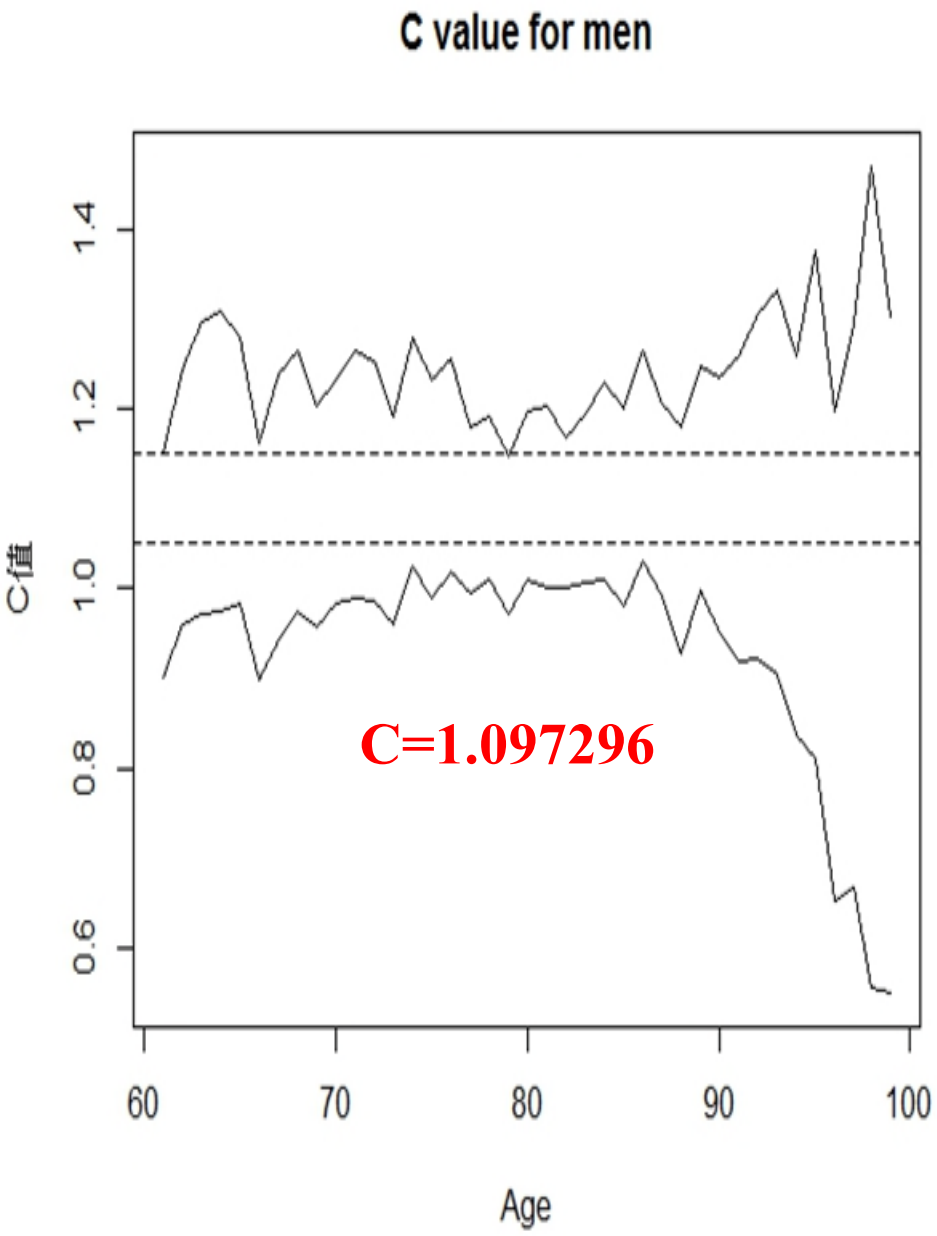
- In practice, if the extreme values are not easy to achieve via optimization methods, it is also possible to use methods such as (weighted) least squares to obtain approximate solutions.

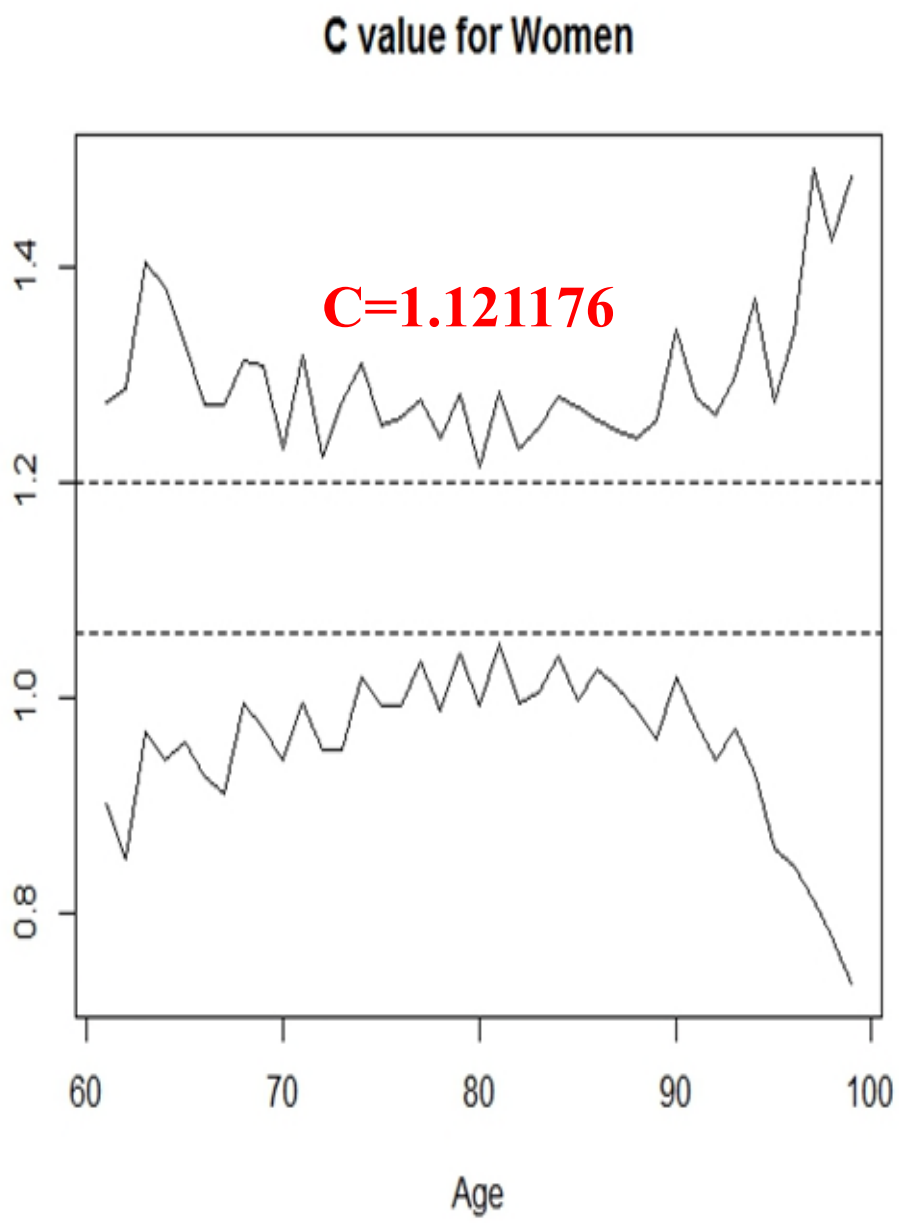
→In this example, since the loss of precision is small, comparing to the instability of nonlinear optimization methods (such as Newton's method), I would recommend using the WLS.

Note: You can download the paper from my homepage.

- Example 6 (Continued). Given the following data and the Gompertz assumption, find the estimates of parameters.

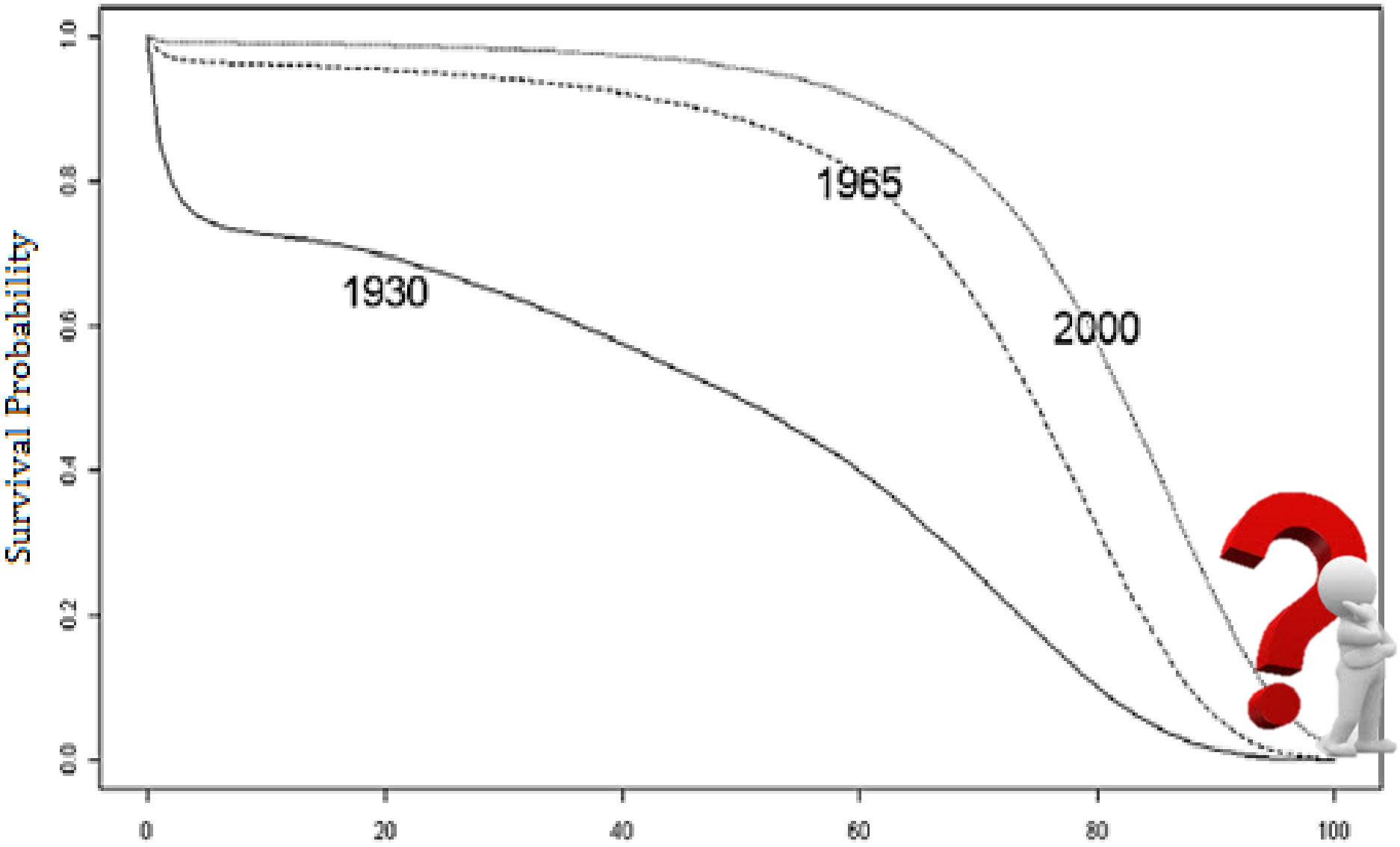
	50~54	55~59	60~64	65~69	70~74	75~79	80~84	85~89	90~94
${}_5l_x$	436981	416303	354898	364844	299732	172914	80739	31606	7165
${}_5d_x$	3369	4766	6000	9071	11103	10431	7903	4503	1510



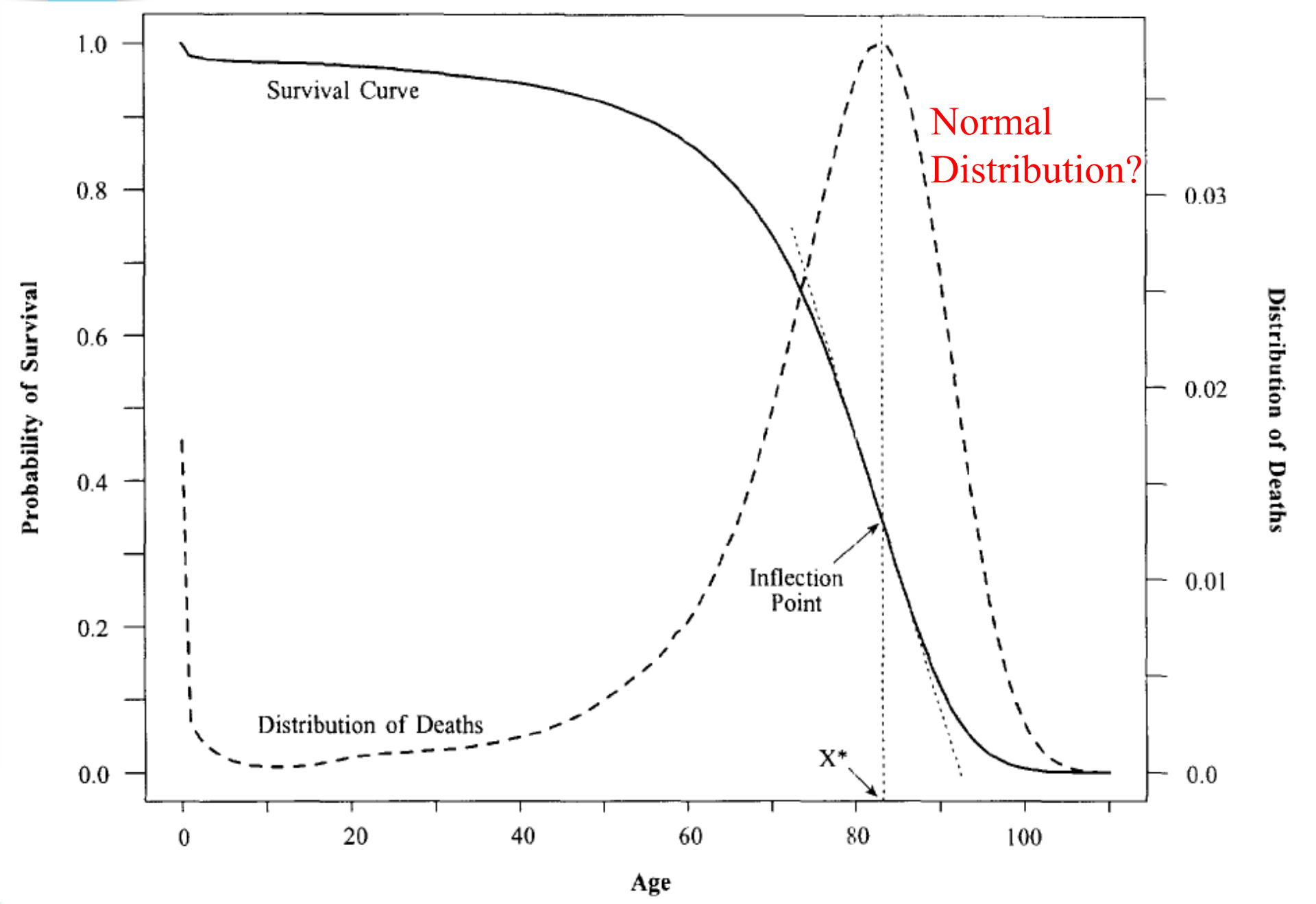


Bootstrap Test for Gompertz Law (2009-11 Taiwan)

Rectangularization of Survival Curve

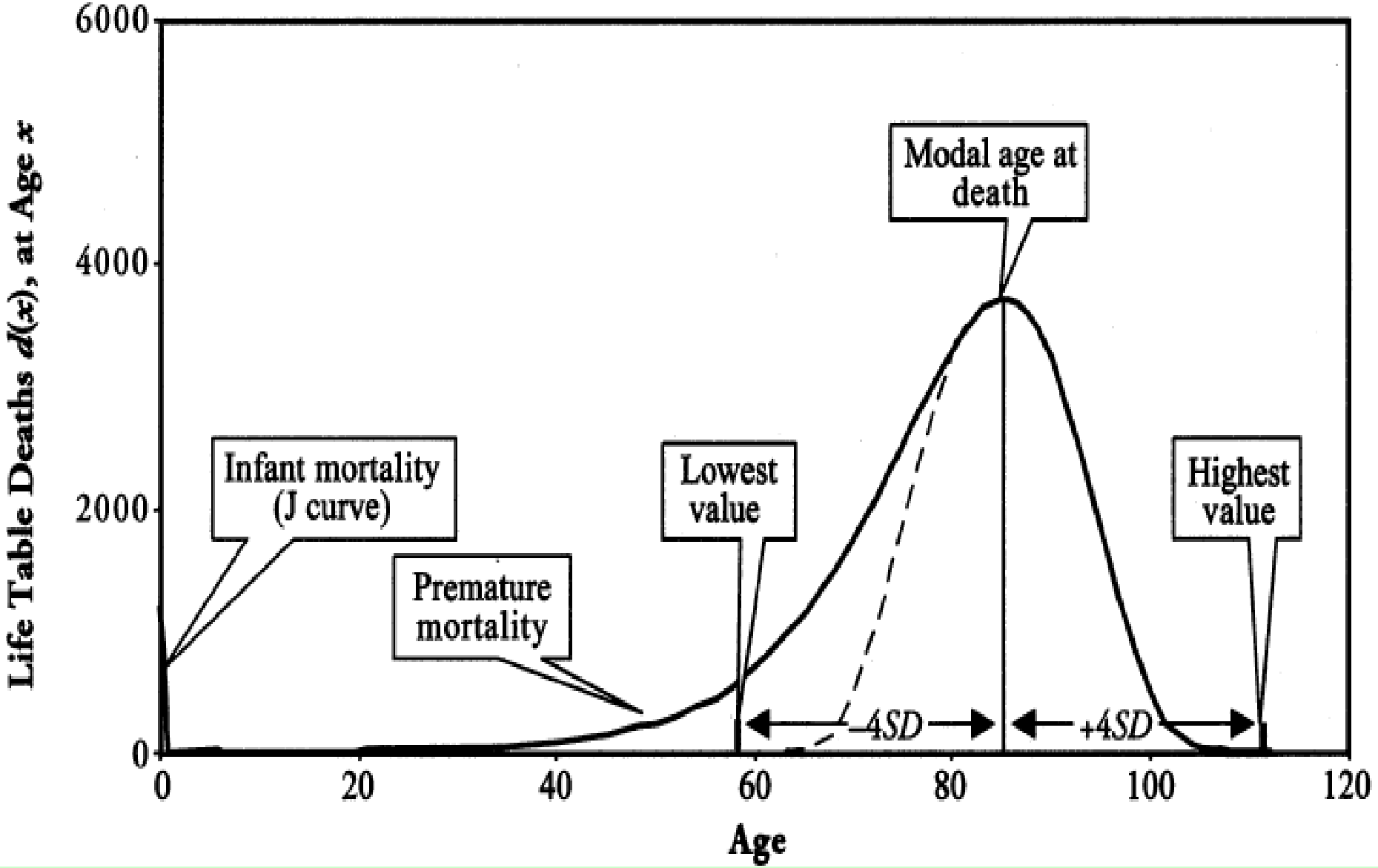


Survival Curves of Taiwan (Female)



Mortality Compression (Wilmoth and Horiuchi, 1999)

Horizontalization, Longevity Extension, Verticalization



Mortality Compression (Cheung et al., 2005)

Proposed Approaches

- The optimization method is model dependent and two distributions are used to verify the NM method.

→Normal distribution (Number of deaths d_x)

$$d_x \propto \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-M)^2}{2\sigma}}$$

→Logistic distribution (Force of Mortality μ_x)

$$\mu_x \propto \frac{a\,e^{bx}}{1+a\,e^{bx}}$$

- Yue, C.J. (2012),"Mortality Compression and Longevity Risk", North American Actuarial Journal, vol. 16(4), 434-448.

Constrained Estimation

- We can handle high-dimensional data, and perform automatic feature selection, by adding penalty terms to the loss function in GLM.

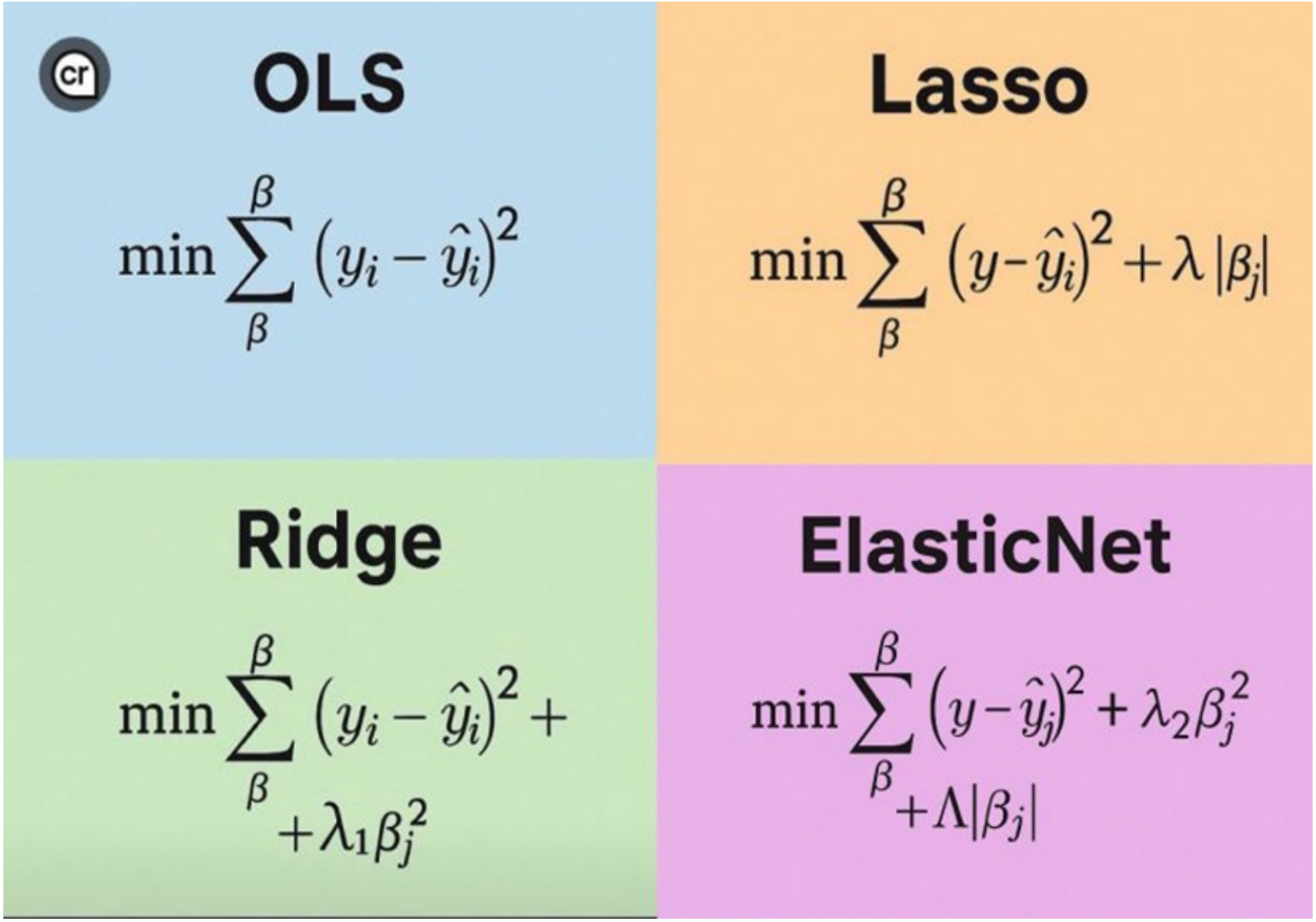
$$Min \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda J_2(f)$$

→e.g., Ridge Regression, LASSO, Elastic Net.

→Use "glmnet" in R and Python; it is based on an efficient coordinate descent algorithm.

- **Ridge**: *Keep all predictors but make them smaller.*
- **LASSO**: *Keep only the important predictors.*
- **Elastic Net**: *Keep important predictors and keep correlated ones together.*

Ridge Regression, LASSO, Elastic Net



https://media.licdn.com/dms/image/v2/D4E22AQE1CmJwIRyNgQ/feedshare-shrink_800/B4EZrQQyM2IIAg-/0/1764430653296?e=2147483647&v=beta&t=uHGpSvTbMEWPy4XNnY0tAd1Qro2eOHi2uv4F-7k5JPQ

L_1, L_2 and L_1+L_2 Norms

https://towardsdatascience.com/wp-content/uploads/2020/11/1nrWncnoJ4V_BkzEf1pd4MA.png