Optimization Methods

1. Introduction.
2. Greedy algorithms for combinatorial optimization.
3. LS and neighborhood structures for combinatorial optimization.
4. Variable neighborhood search, neighborhood descent, SA, TS, EC.
5. Branch and bound algorithms, and subset selection algorithms.
6. Linear programming problem formulations and applications.
7. Linear programming algorithms.
8. Integer linear programming algorithms.
9. Unconstrained nonlinear optimization and gradient descent.
10. Newton's methods and Levenberg-Marquardt modification.
11. Quasi-Newton methods and conjugate direction methods.
12. Nonlinear optimization with equality constraints.
13. Nonlinear optimization with inequality constraints.
14. Problem formulation and concepts in multi-objective optimization.
15. Search for single final solution in multi-objective optimization.
16: Search for multiple solutions in multi-objective optimization.

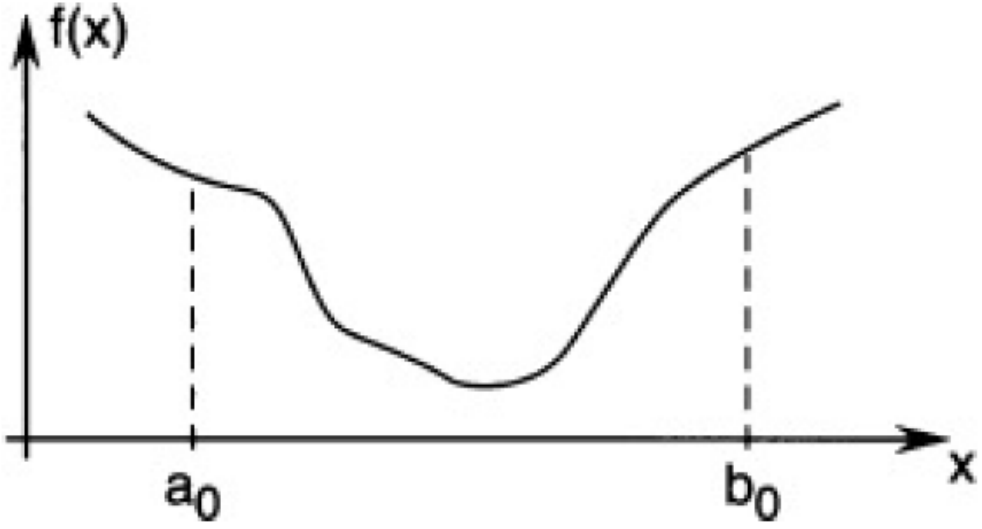
Golden Section Search

Problem: Minimization of $f(x)$ subject to $a_0 \le x \le b_0$

Assumption: $f(x)$ is unimodal ($f(x)$ has a single local minimum)
We do not know the mathematical form of $f(x)$.

Algorithm (Your Task): To determine $x^{(1)}$, $x^{(2)}$, $x^{(3)}$, ...

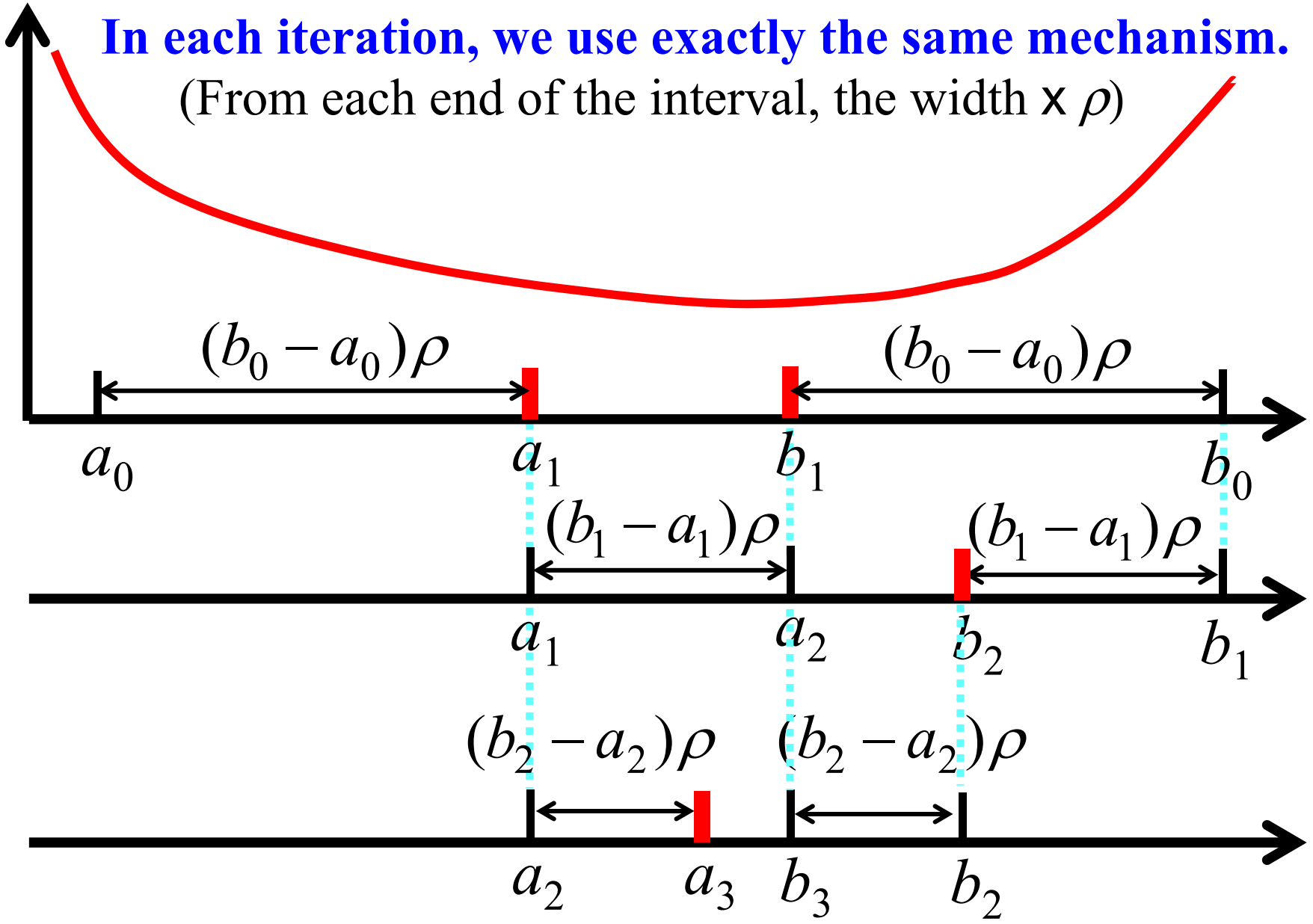
We choose these points in such a way that an approximation to the minimizer of *f* may be achieved in as few evaluations as possible. Our goal is to narrow the range progressively until the minimizer is "boxed in" with sufficient accuracy.



Golden Section Search $a_0 \le x \le b_0$ $\rho = 0.382$

In each iteration, we use exactly the same mechanism.

(From each end of the interval, the width x ρ)



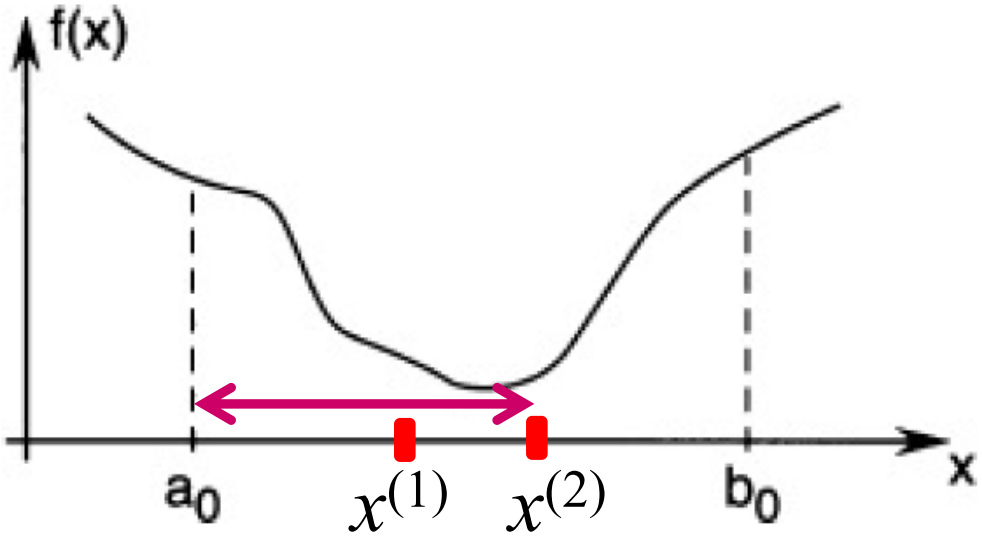
If you can examine only two inside points:

Problem: Minimization of $f(x)$ subject to $a_0 \le x \le b_0$

Assumption: $f(x)$ is unimodal ($f(x)$ has a single local minimum)
We do not know the mathematical form of $f(x)$.

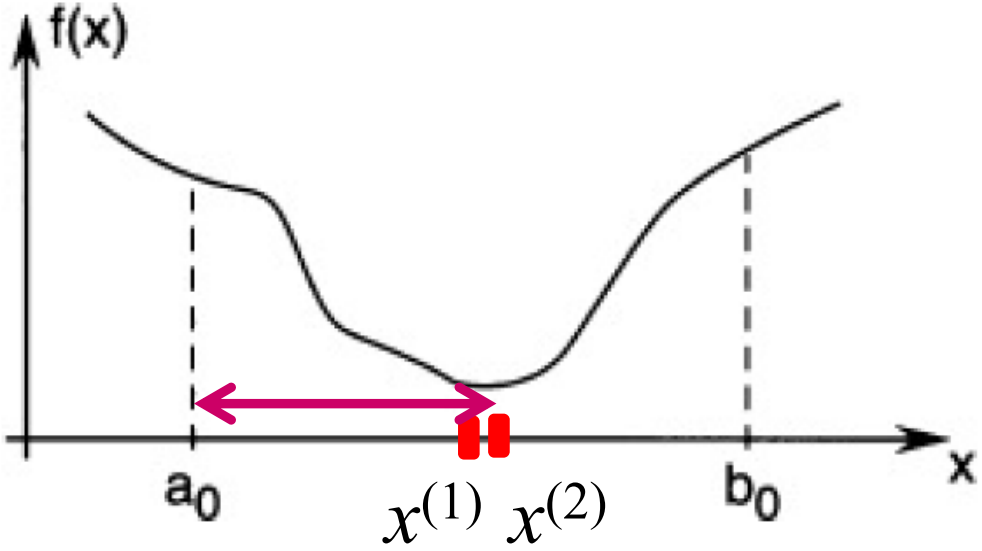
Algorithm (Your Task): To determine $x^{(1)}$, $x^{(2)}$

To minimize the range where the optimal solution is included.



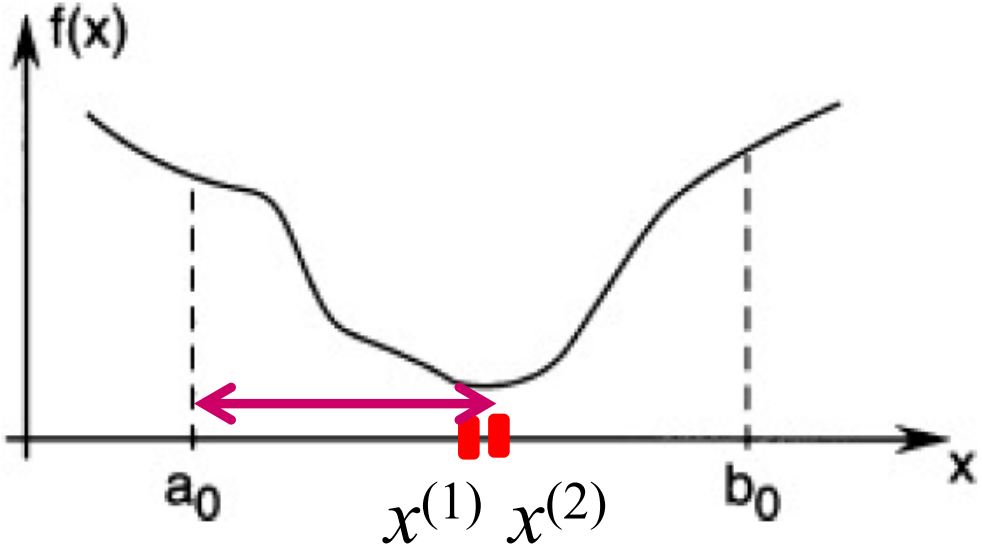
Q. What is the best choice of the two points ? **A.** _____.

If you can examine only two inside points:



$$x^{(1)} = (a_0 + b_0)/2 - \varepsilon \qquad x^{(2)} = (a_0 + b_0)/2 + \varepsilon$$

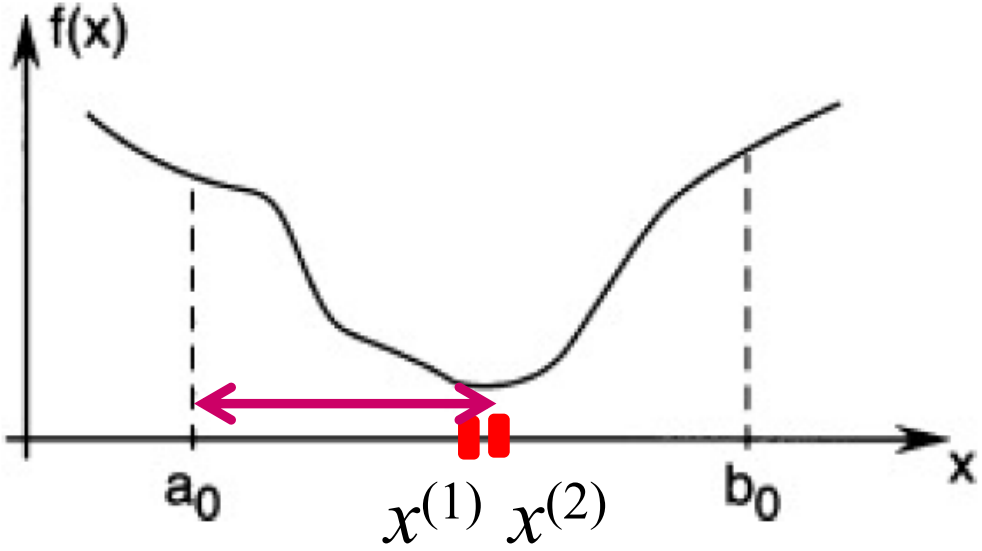
If you can examine only two inside points:



$$x^{(1)} = (a_0 + b_0)/2 - \varepsilon \qquad x^{(2)} = (a_0 + b_0)/2 + \varepsilon$$

This mechanism can be viewed as estimating $f'(x)$ at the mid point.

If you can examine only two inside points:



$$x^{(1)} = (a_0 + b_0)/2 - \varepsilon \qquad x^{(2)} = (a_0 + b_0)/2 + \varepsilon$$

The range becomes about 1/2 of the original size.

After T iterations of this mechanism, the range is $(1/2)^T$.
After S solution examinations, the range is $(1/2)^{S/2} = 0.7071^S$

Golden selection search: The range is 0.618^{S-1}

(Two solutions are examined only in the first step)

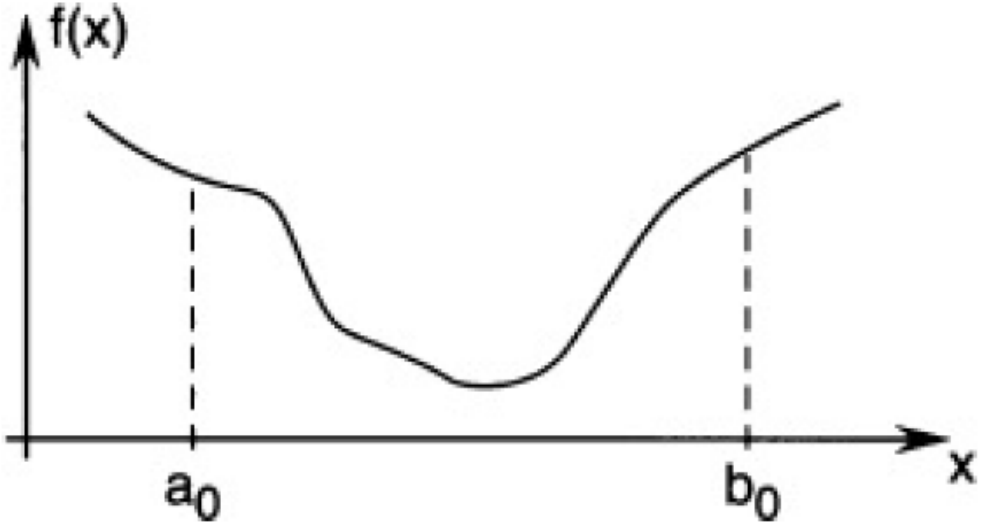
Fibonacci Method

Problem: Minimization of $f(x)$ subject to $a_0 \le x \le b_0$

Assumption: $f(x)$ is unimodal ($f(x)$ has a single local minimum)
We do not know the mathematical form of $f(x)$.

Algorithm (Your Task): To determine $x^{(1)}$, $x^{(2)}$, $x^{(3)}$, ...

We choose these points in such a way that an approximation to the minimizer of *f* may be achieved in as few evaluations as possible. Our goal is to narrow the range progressively until the minimizer is "boxed in" with sufficient accuracy.

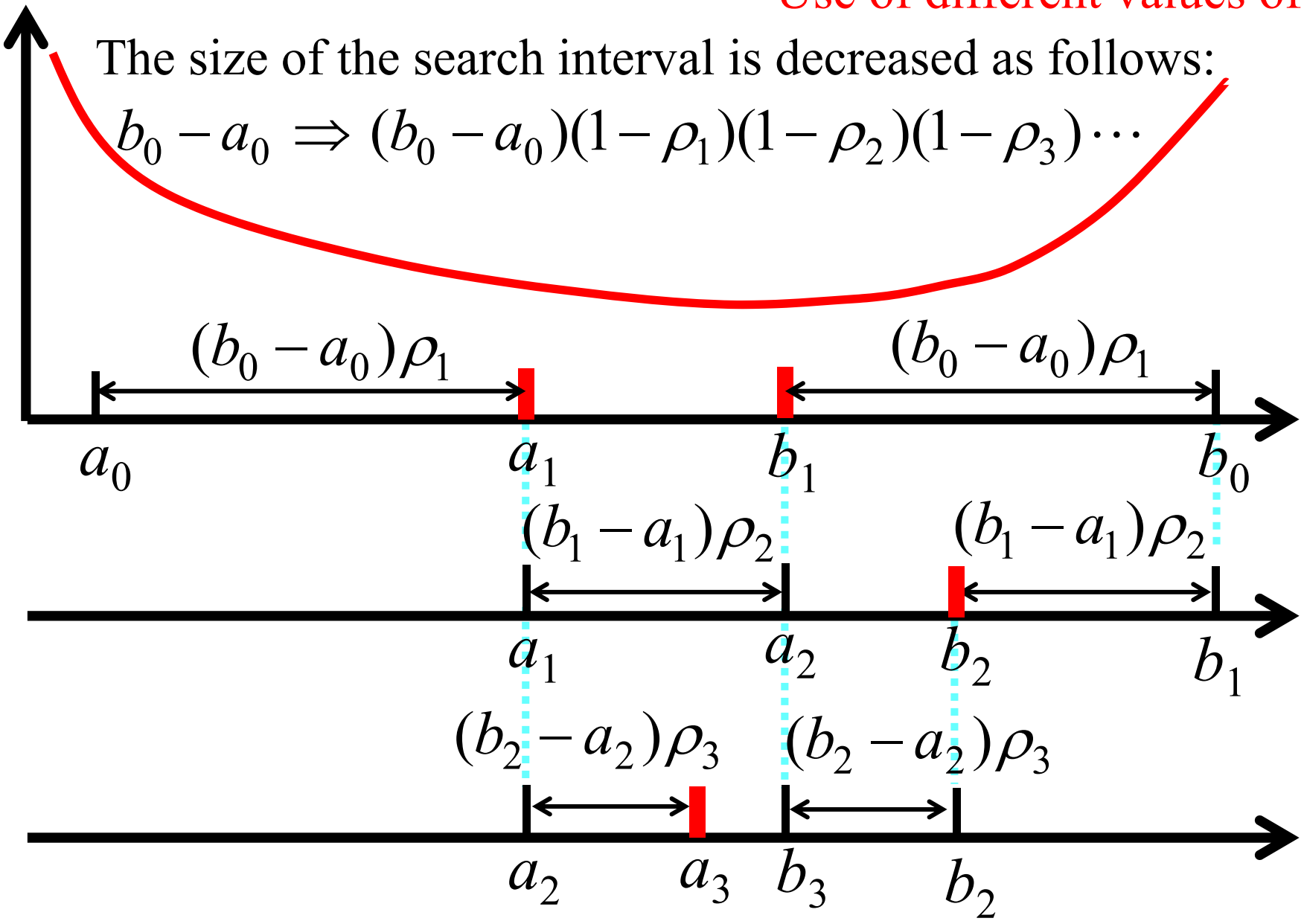


Fibonacci Method $a_0 \le x \le b_0$ $\rho_1, \rho_2, \rho_3, \rho_4, \cdots$

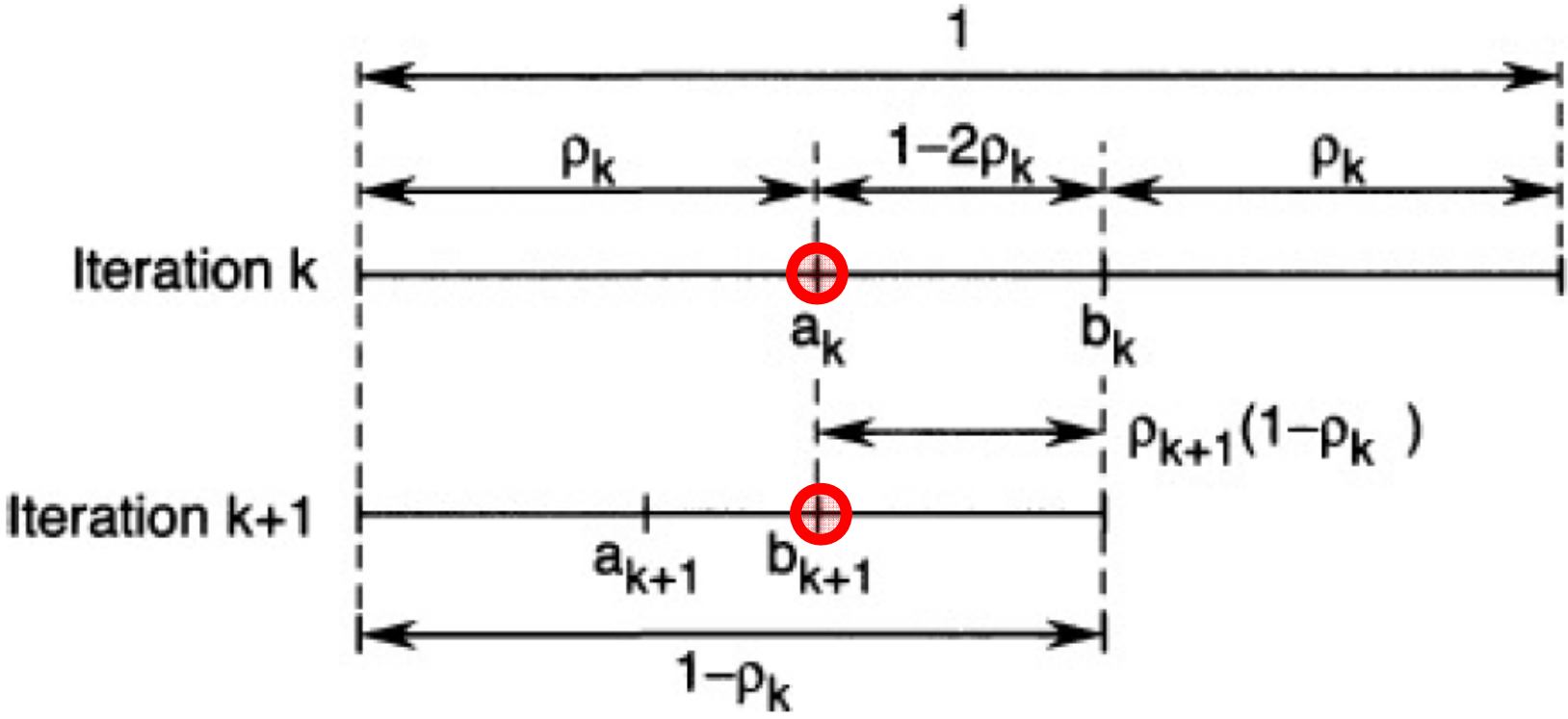
Use of different values of ρ

The size of the search interval is decreased as follows:

$$b_0 - a_0 \Rightarrow (b_0 - a_0)(1-\rho_1)(1-\rho_2)(1-\rho_3)\cdots$$



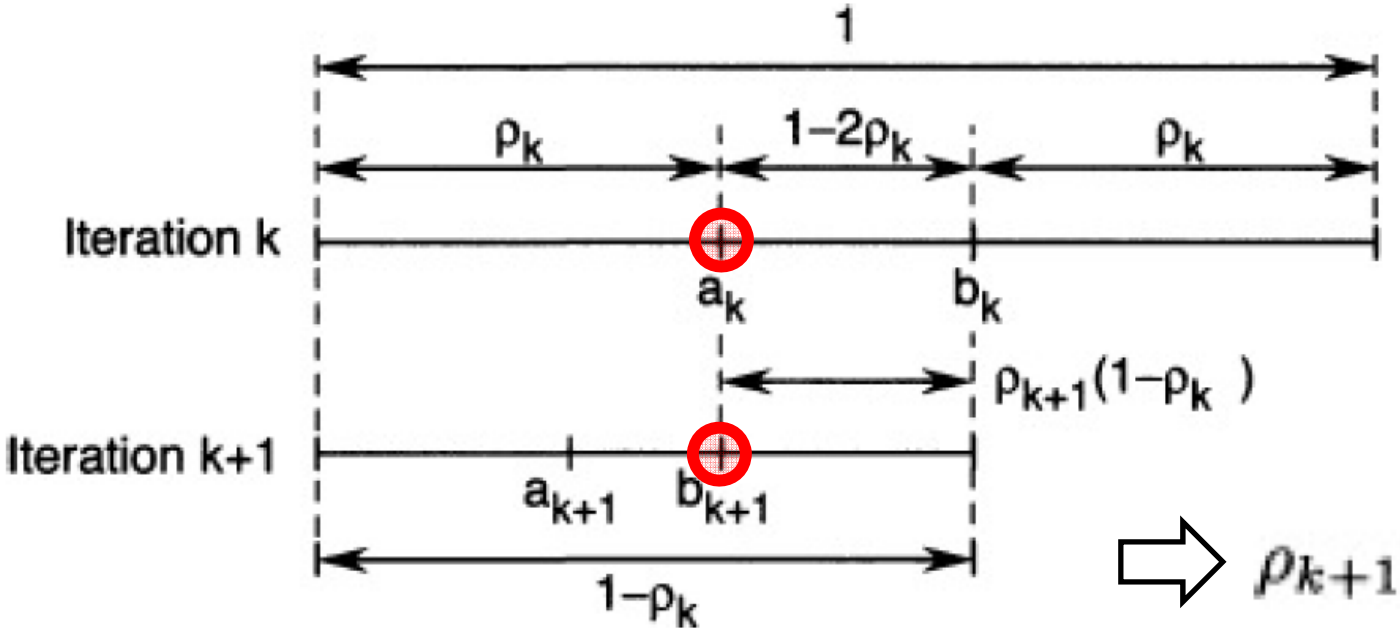
Fibonacci Method $a_0 \le x \le b_0$ $\rho_1, \rho_2, \rho_3, \rho_4, \ldots$



A single point is examined in each iteration (e.g., $a_k = b_{k+1}$)

$$\rho_{k+1}(1-\rho_k) = 1 - 2\rho_k \quad \Rightarrow \quad \rho_{k+1} = 1 - \frac{\rho_k}{1-\rho_k}$$

Fibonacci Method $a_0 \le x \le b_0$



$$\Rightarrow \rho_{k+1} = 1 - \frac{\rho_k}{1-\rho_k}$$

How to determine $\rho_1, \rho_2, \rho_3, \rho_4, \ldots$

$$\text{minimize} \quad (1-\rho_1)(1-\rho_2)\cdots(1-\rho_N)$$ (Minimize the range)

$$\text{subject to} \quad \rho_{k+1} = 1 - \frac{\rho_k}{1-\rho_k}, \; k = 1, \ldots, N-1$$

$$0 \le \rho_k \le \frac{1}{2}, \; k = 1, \ldots, N.$$

$$\begin{aligned} \text{minimize} \quad & (1-\rho_1)(1-\rho_2)\cdots(1-\rho_N) \\ \text{subject to} \quad & \rho_{k+1} = 1 - \frac{\rho_k}{1-\rho_k}, \; k = 1, \ldots, N-1 \\ & 0 \le \rho_k \le \frac{1}{2}, \; k = 1, \ldots, N. \end{aligned}$$

(Minimize the range)

Q. How to solve this problem ? Your Idea: _____.

Note 1: If we use an additional condition $\rho_1 = \rho_2 = \rho_3 = \rho_4 = \ldots$,
we have the golden selection search.

$$\begin{aligned} \text{minimize} \quad & (1-\rho_1)(1-\rho_2)\cdots(1-\rho_N) \\ \text{subject to} \quad & \rho_{k+1} = 1 - \frac{\rho_k}{1-\rho_k}, \; k = 1, \ldots, N-1 \\ & 0 \le \rho_k \le \frac{1}{2}, \; k = 1, \ldots, N. \end{aligned}$$

(Minimize the range)

Q. How to solve this problem ? Your Idea: _____.

Note 1: If we use an additional condition $\rho_1 = \rho_2 = \rho_3 = \rho_4 = \ldots$, we have the golden selection search.

Note 2: You need to solve this problem just once. Then, you can use the result as the algorithm (i.e., you do not have to solve this problem many times for each optimization task).

Before we give the solution to the optimization problem above, we need to introduce the *Fibonacci sequence* $F_1, F_2, F_3, \ldots$. This sequence is defined as follows. First, let $F_{-1} = 0$ and $F_0 = 1$ by convention. Then, for $k \geq 0$, $F_{k+1} = F_k + F_{k-1}$.

Some values of elements in the Fibonacci sequence are:

F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8
1	2	3	5	8	13	21	34

It turns out that the solution to the optimization problem above is

$$\rho_1 = 1 - \frac{F_N}{F_{N+1}},$$
$$\rho_2 = 1 - \frac{F_{N-1}}{F_N},$$
$$\vdots$$
$$\rho_k = 1 - \frac{F_{N-k+1}}{F_{N-k+2}},$$
$$\vdots$$
$$\rho_N = 1 - \frac{F_1}{F_2},$$

where the F_k are the elements of the Fibonacci sequence. The resulting algorithm is called the *Fibonacci search method*. We present a proof for the optimality of the Fibonacci search method later in this section.

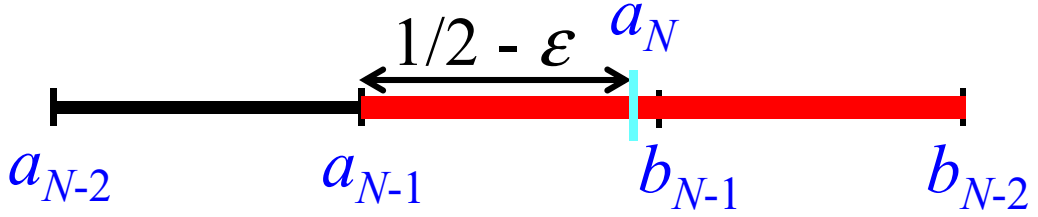
In the Fibonacci search method, the uncertainty range is reduced by the factor

$$(1-\rho_1)(1-\rho_2)\cdots(1-\rho_N) = \frac{F_N}{F_{N+1}}\frac{F_{N-1}}{F_N}\cdots\frac{F_1}{F_2} = \frac{F_1}{F_{N+1}} = \frac{1}{F_{N+1}}.$$

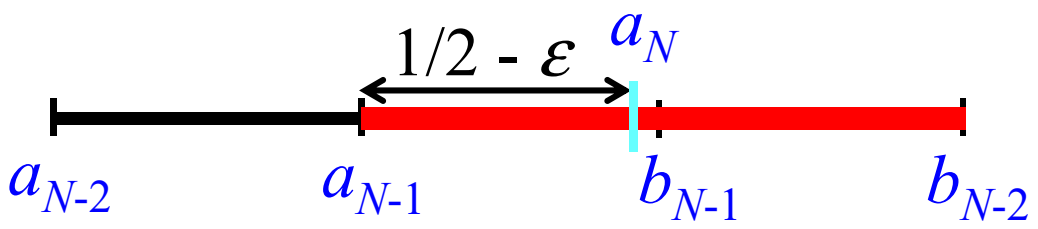
Because the Fibonacci method uses the optimal values of ρ_1, p_2, …, the reduction factor above is less than that of the golden section method. In other words, the Fibonacci method is better than the golden section method in that it gives a smaller final uncertainty range.

We point out that there is an anomaly in the final iteration of the Fibonacci search method, because $\rho_N = 1 - \frac{F_1}{F_2} = \frac{1}{2}$.

Recall that we need two intermediate points at each stage, one that comes from a previous iteration and another that is a new evaluation point. However, with ρ_N = 1/2, the two intermediate points coincide in the middle of the uncertainty interval, and therefore we cannot further reduce the uncertainty range. To get around this problem, we perform the new evaluation for the last iteration using ρ_N = 1/2 − ε, where ε is a small number. In other words, the new evaluation point is just to the left or right of the midpoint of the uncertainty interval. This modification to the Fibonacci method is, of course, of no significant practical consequence.



$\rho_N = 1/2$



As a result of the modification above, the reduction in the uncertainty range at the last iteration may be either $1 - \rho_N = \frac{1}{2}$ or

$$1 - (\rho_N - \varepsilon) = \frac{1}{2} + \varepsilon = \frac{1 + 2\varepsilon}{2},$$

depending on which of the two points has the smaller objective function value. Therefore, in the worst case, the reduction factor in the uncertainty range for the Fibonacci method is $\frac{1 + 2\varepsilon}{F_{N+1}}$.

For proof, see the book.

Bisection Method

Problem: Minimization of $f(x)$ subject to $a_0 \le x \le b_0$

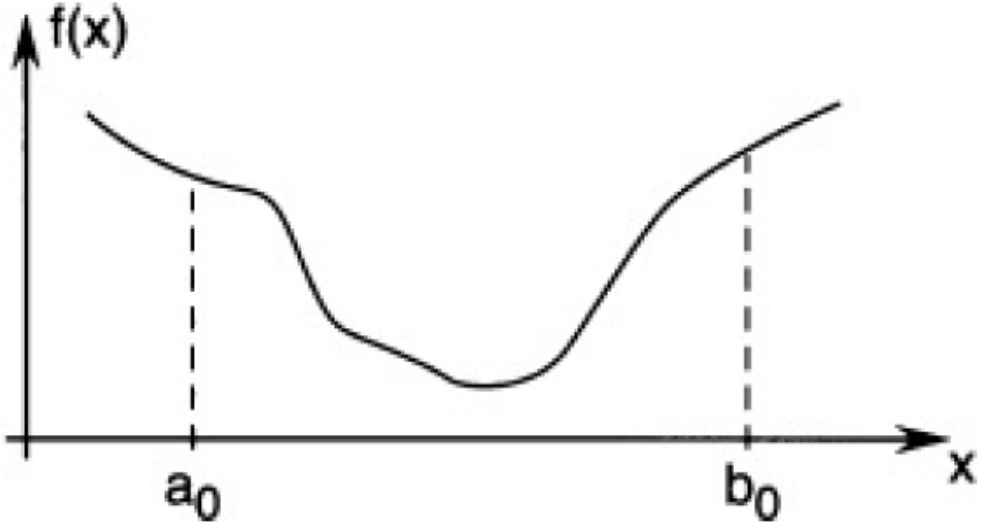
Assumption: $f(x)$ is unimodal ($f(x)$ has a single local minimum).

The value of $f'(x)$ at each point can be used.

We do not know the mathematical form of $f(x)$.

We cannot use the mathematical form of $f'(x)$.

Algorithm (Your Task): To determine $x^{(1)}$, $x^{(2)}$, $x^{(3)}$, ...



Bisection Method

Problem: Minimization of $f(x)$ subject to $a_0 \le x \le b_0$

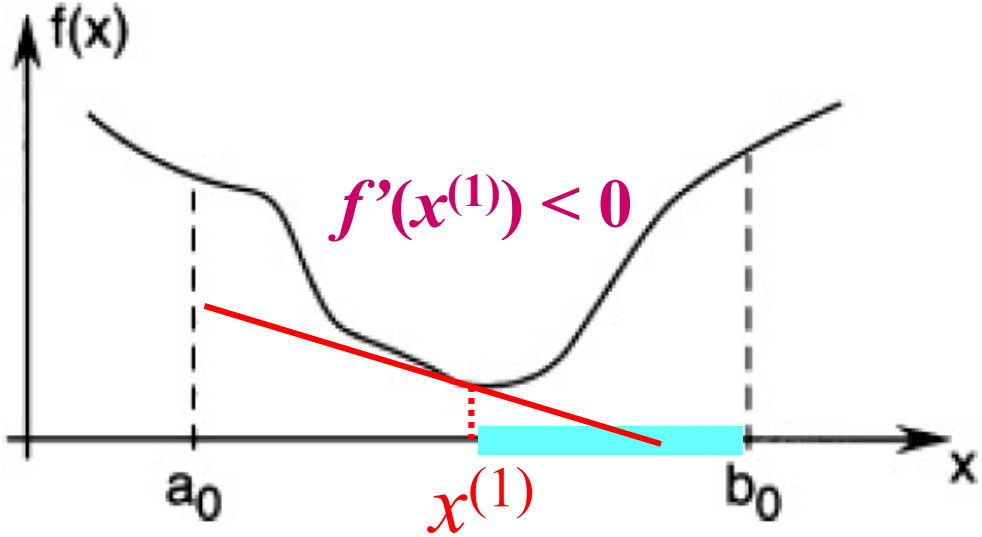
Assumption: $f(x)$ is unimodal ($f(x)$ has a single local minimum).

The value of $f'(x)$ at each point can be used.

We do not know the mathematical form of $f(x)$.

We cannot use the mathematical form of $f'(x)$.

Algorithm (Your Task): To determine $x^{(1)}$, $x^{(2)}$, $x^{(3)}$, ...



$$x^{(1)} = (a_0 + b_0)/2$$

$$\cdots$$

$$x^{(k+1)} = (a_k + b_k)/2$$

$$\cdots$$

After N iterations, the range is $(1/2)^N$

Newton's Method

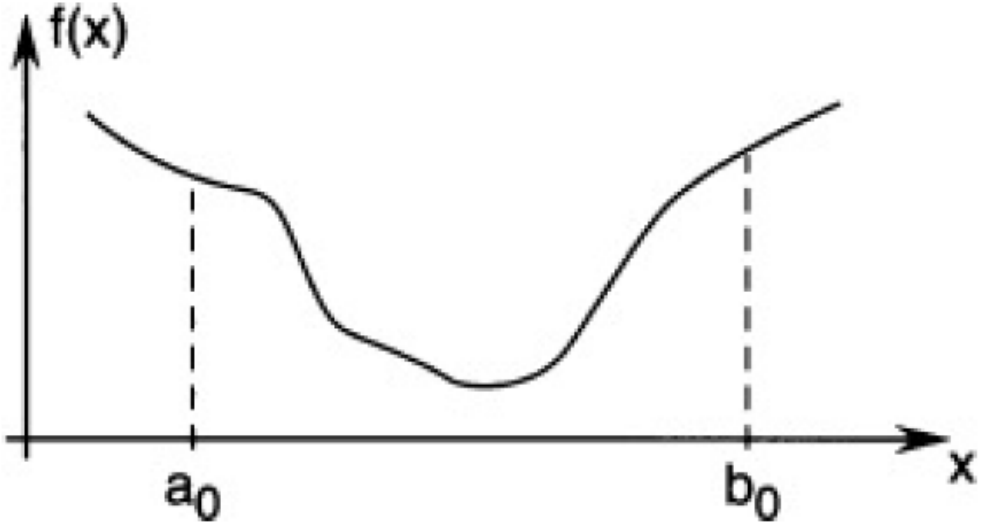
Problem: Minimization of $f(x)$ subject to $a_0 \le x \le b_0$

Assumption: $f(x)$ is unimodal ($f(x)$ has a single local minimum).

The values of $f(x)$, $f'(x)$ and $f''(x)$ are available at each measurement point $x^{(1)}$, $x^{(2)}$, $x^{(3)}$, ...

No mathematical form is available.

Algorithm (Your Task): To determine $x^{(1)}$, $x^{(2)}$, $x^{(3)}$, ...

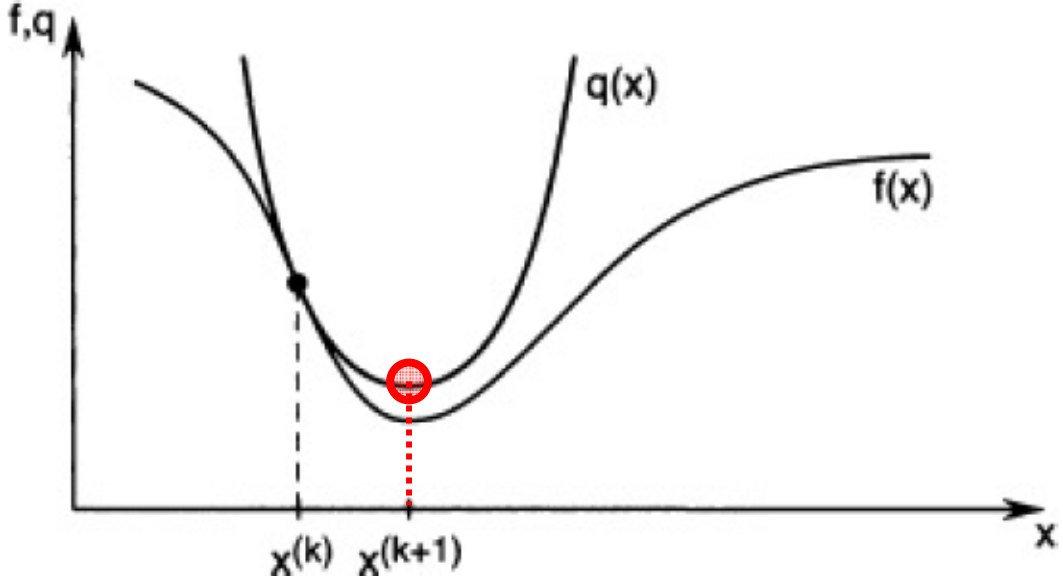


Newton's Method

Basic Idea: Use of a quadratic function $q(x)$ which approximates $f(x)$ around the measurement point $x(k)$ under the following conditions: $q(x^{(k)}) = f(x^{(k)})$, $q'(x^{(k)}) = f'(x^{(k)})$ and $q''(x^{(k)}) = f''(x^{(k)})$. This quadratic function is

$$q(x) = f(x^{(k)}) + f'(x^{(k)})(x - x^{(k)}) + \frac{1}{2} f''(x^{(k)})(x - x^{(k)})^2.$$

Then $x(k+1)$ is determined by the point which minimizes $q(x)$.



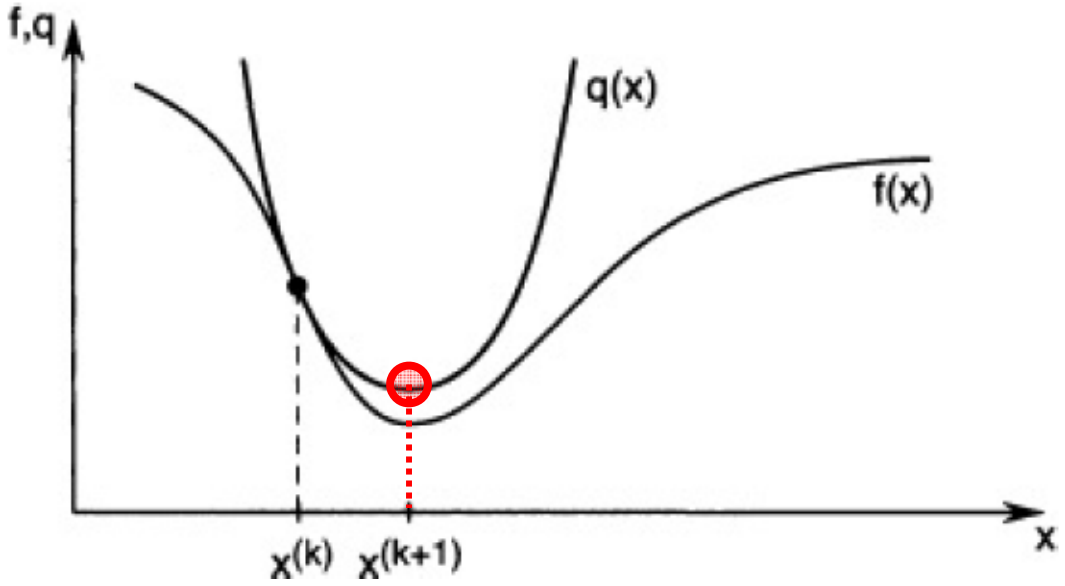
Newton's Method

$$q(x) = f(x^{(k)}) + f'(x^{(k)})(x - x^{(k)}) + \frac{1}{2} f''(x^{(k)})(x - x^{(k)})^2.$$

Then $x^{(k+1)}$ is determined by the point which minimizes $q(x)$.

$$0 = q'(x) = f'(x^{(k)}) + f''(x^{(k)})(x - x^{(k)}).$$

$$\Rightarrow \quad x^{(k+1)} = x^{(k)} - \frac{f'(x^{(k)})}{f''(x^{(k)})}.$$



Newton's Method

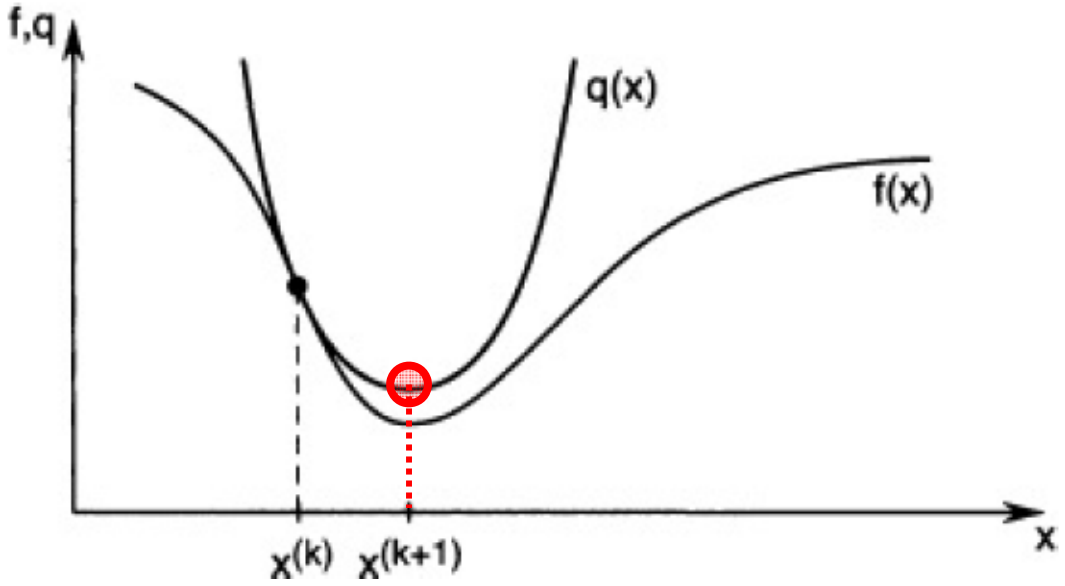
$$q(x) = f(x^{(k)}) + f'(x^{(k)})(x - x^{(k)}) + \frac{1}{2} f''(x^{(k)})(x - x^{(k)})^2.$$

Then $x^{(k+1)}$ is determined by the point which minimizes $q(x)$.

$$0 = q'(x) = f'(x^{(k)}) + f''(x^{(k)})(x - x^{(k)}).$$

$$\Rightarrow \quad x^{(k+1)} = x^{(k)} - \frac{f'(x^{(k)})}{f''(x^{(k)})}.$$

Looks nice.



Newton's Method

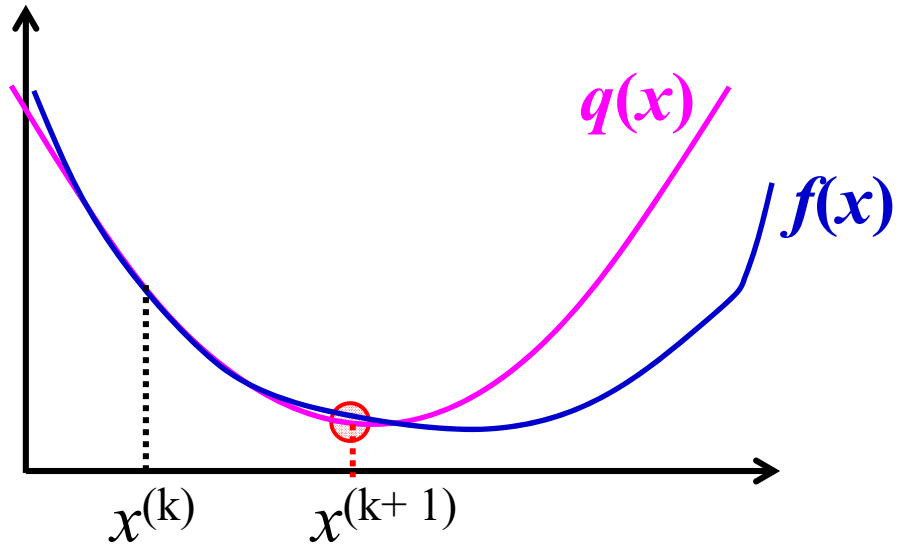
$$q(x) = f(x^{(k)}) + f'(x^{(k)})(x - x^{(k)}) + \frac{1}{2} f''(x^{(k)})(x - x^{(k)})^2.$$

Then $x^{(k+1)}$ is determined by the point which minimizes $q(x)$.

$$0 = q'(x) = f'(x^{(k)}) + f''(x^{(k)})(x - x^{(k)}).$$

$$\Rightarrow \quad x^{(k+1)} = x^{(k)} - \frac{f'(x^{(k)})}{f''(x^{(k)})}.$$

Looks nice.



Newton's Method

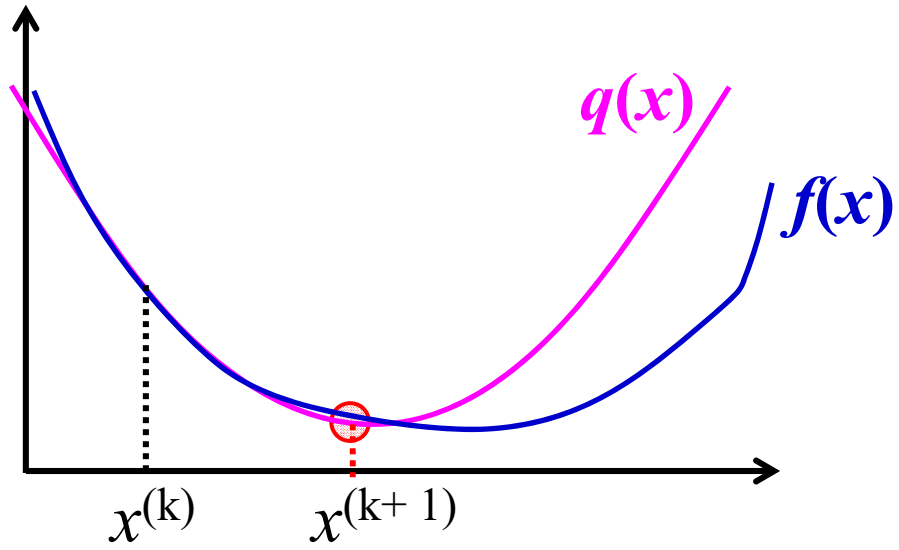
$$q(x) = f(x^{(k)}) + f'(x^{(k)})(x - x^{(k)}) + \frac{1}{2} f''(x^{(k)})(x - x^{(k)})^2.$$

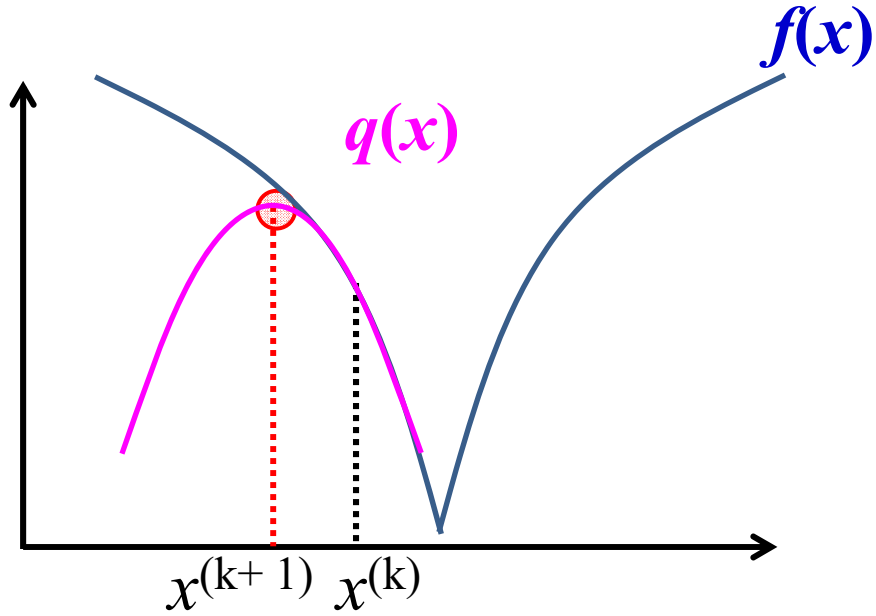
Then $x^{(k+1)}$ is determined by the point which minimizes $q(x)$.

$$0 = q'(x) = f'(x^{(k)}) + f''(x^{(k)})(x - x^{(k)}).$$

$$\Rightarrow \quad x^{(k+1)} = x^{(k)} - \frac{f'(x^{(k)})}{f''(x^{(k)})}.$$

Not always good.





Newton's Method

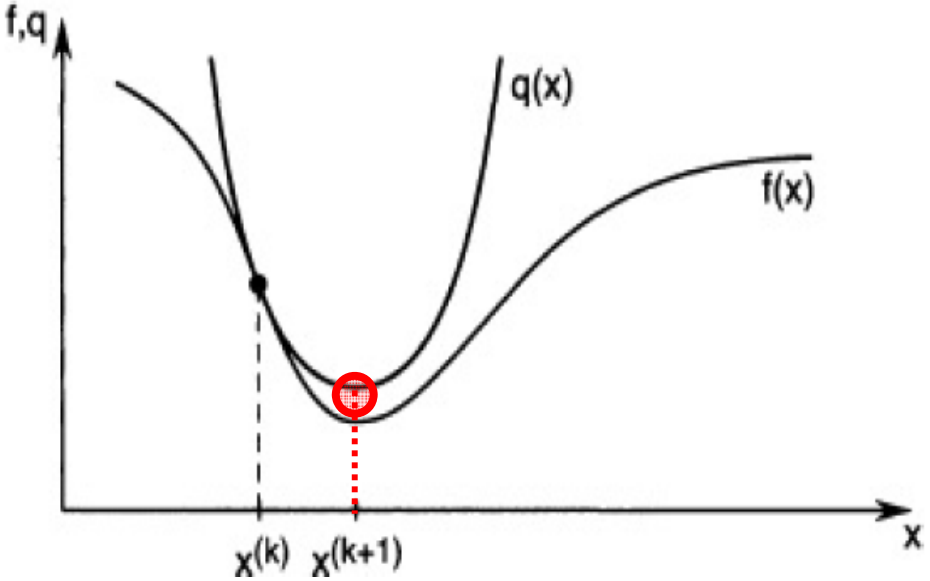
$$q(x) = f(x^{(k)}) + f'(x^{(k)})(x - x^{(k)}) + \frac{1}{2} f''(x^{(k)})(x - x^{(k)})^2.$$

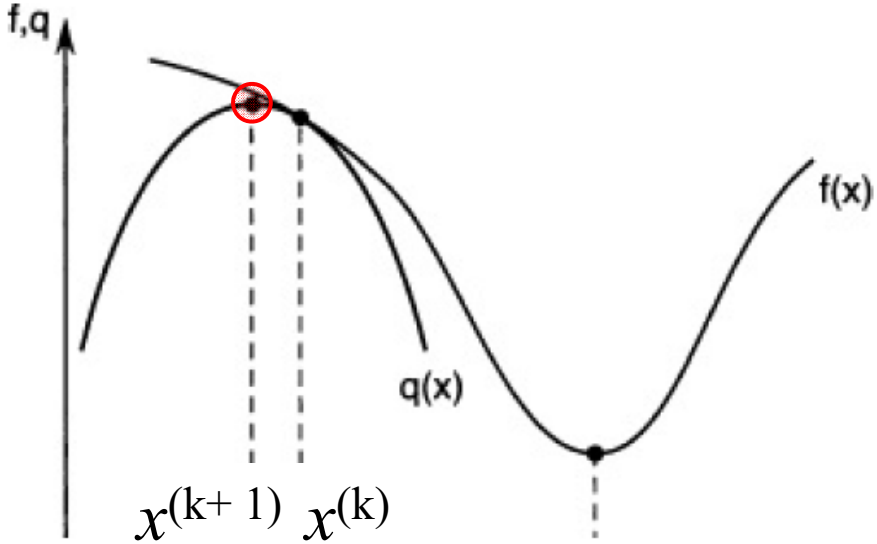
Then $x^{(k+1)}$ is determined by the point which minimizes $q(x)$.

$$0 = q'(x) = f'(x^{(k)}) + f''(x^{(k)})(x - x^{(k)}).$$

$$\Rightarrow \quad x^{(k+1)} = x^{(k)} - \frac{f'(x^{(k)})}{f''(x^{(k)})}.$$

Not always good.





Newton's Method

$$q(x) = f(x^{(k)}) + f'(x^{(k)})(x - x^{(k)}) + \frac{1}{2} f''(x^{(k)})(x - x^{(k)})^2.$$

Then $x^{(k+1)}$ is determined by the point which minimizes $q(x)$.

$$0 = q'(x) = f'(x^{(k)}) + f''(x^{(k)})(x - x^{(k)}).$$

$$\Rightarrow \quad x^{(k+1)} = x^{(k)} - \frac{f'(x^{(k)})}{f''(x^{(k)})}.$$

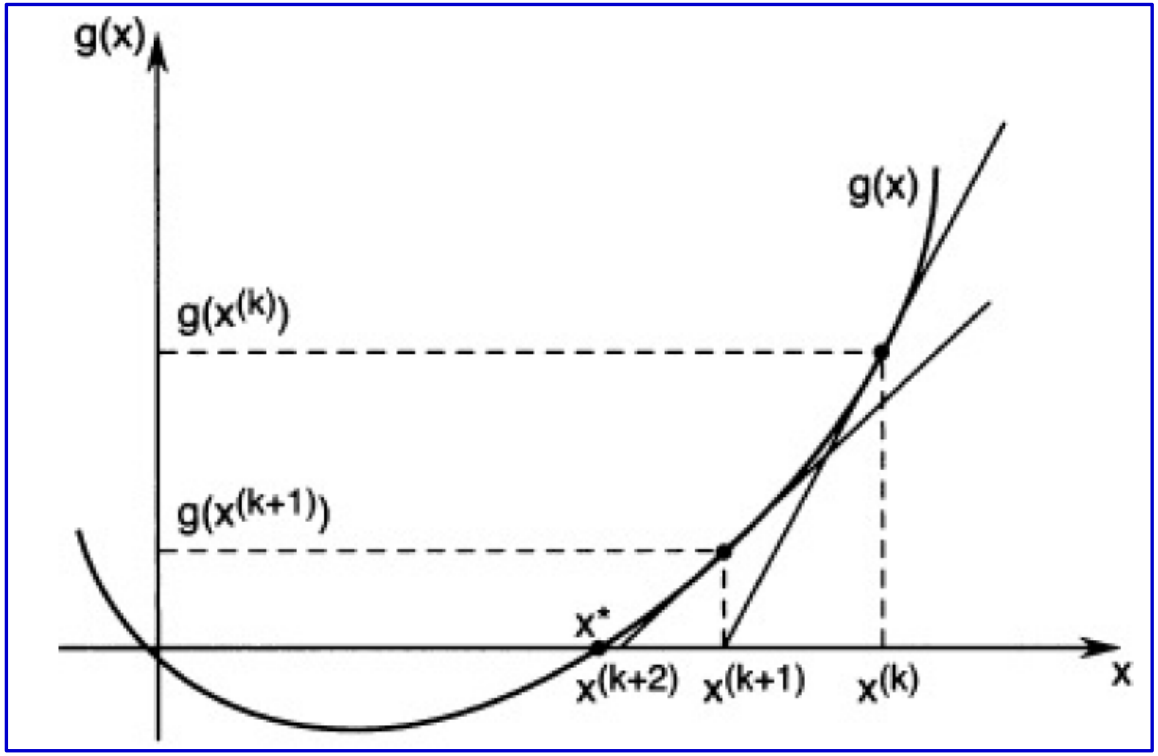
Newton's method can also be viewed as a way to drive the first derivative of f to zero. Indeed, if we set $g(x) = f'(x)$, then we obtain a formula for iterative solution of the equation $g(x) = 0$: $x^{(k+1)} = x^{(k)} - \frac{g(x^{(k)})}{g'(x^{(k)})}$.

In other words, we can use Newton's method for zero finding.

Newton's method can also be viewed as a way to drive the first derivative of f to zero. Indeed, if we set $g(x) = f'(x)$, then we obtain a formula for iterative solution of the equation $g(x) = 0$: $x^{(k+1)} = x^{(k)} - \dfrac{g(x^{(k)})}{g'(x^{(k)})}$.

In other words, we can use Newton's method for zero finding.

Newton's method for solving equations of the form $g(x) = 0$ is also referred to as *Newton's method of tangents*. This name is easily justified if we look at a geometric interpretation of the method when applied to the solution of the equation $g(x) = 0$



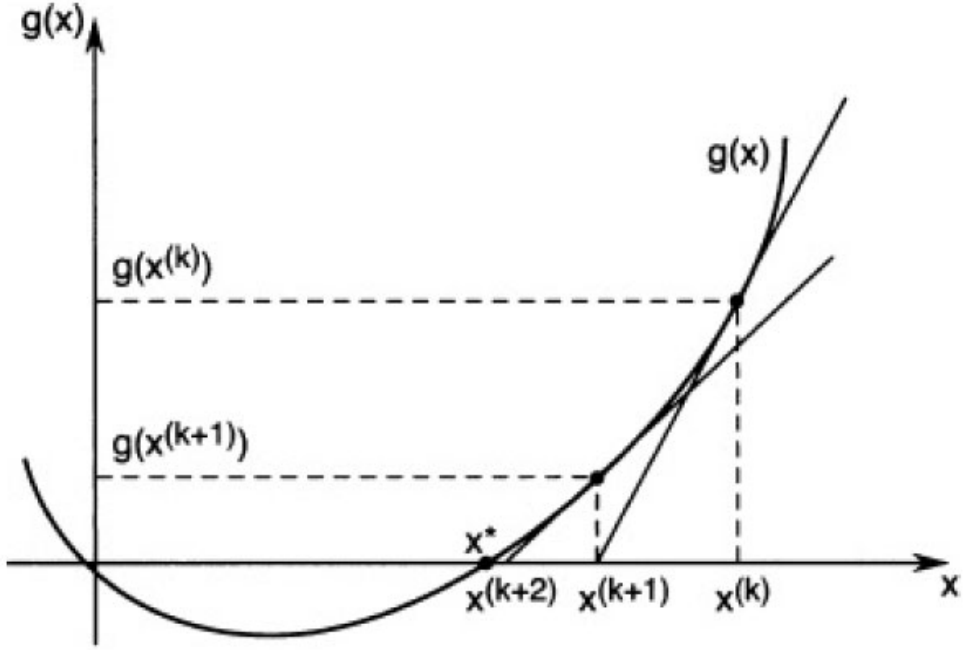


Figure 7.8 Newton's method of tangents.

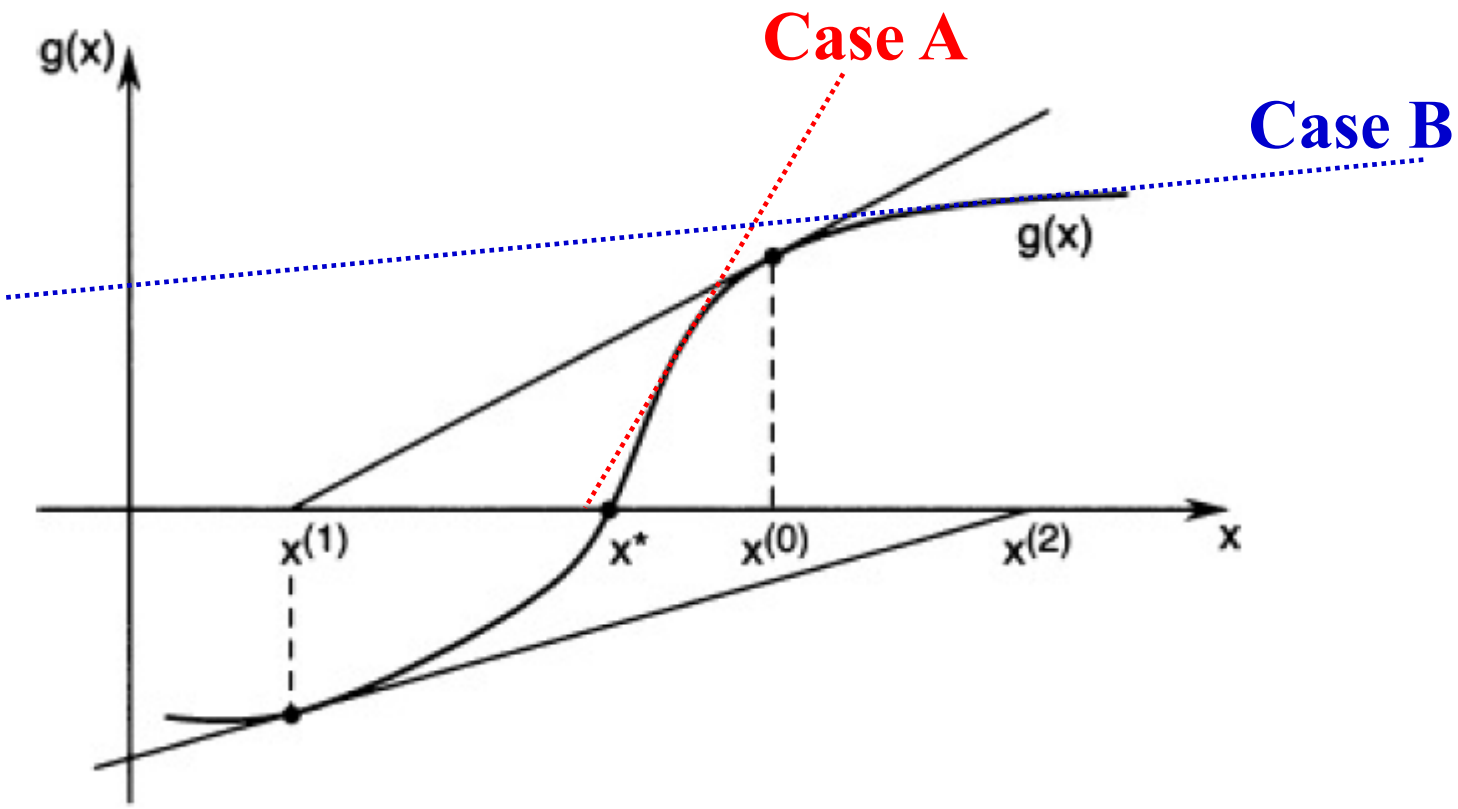
If we draw a tangent to $g(x)$ at the given point $x^{(k)}$, then the tangent line intersects the x-axis at the point $x^{(k+1)}$, which we expect to be closer to the root x^* of $g(x) = 0$. Note that the slope of $g(x)$ at $x^{(k)}$ is $g'(x^{(k)}) = \dfrac{g(x^{(k)})}{x^{(k)} - x^{(k+1)}}$.

Thus,

$$x^{(k+1)} = x^{(k)} - \frac{g(x^{(k)})}{g'(x^{(k)})}.$$

Newton's method of tangents may fail if the first approximation to the root is such that the ratio $g(x^{(0)})/g'(x^{(0)})$ is not small enough (see Figure 7.9). Thus, an initial approximation to the root is very important.

Figure 7.9 Example where Newton's method of tangents fails to converge to the root x^* of $g(x) = 0$.



Secant Method

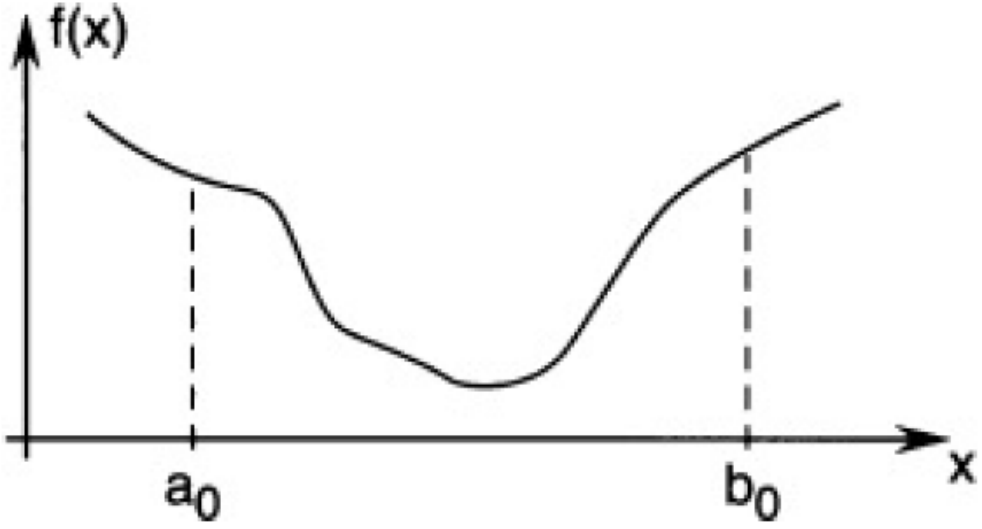
$f(x)$ is unimodal ($f(x)$ has a single local minimum).

The values of $f(x)$ and $f'(x)$ are available at each point.

The value of $f''(x)$ is not available, which is approximated.

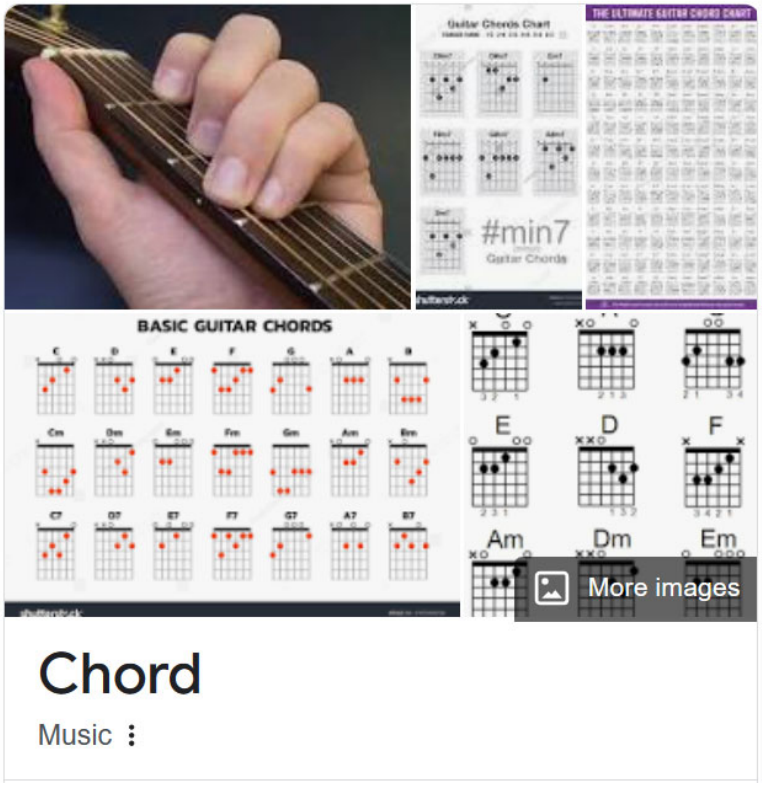
In Newton's Methods:

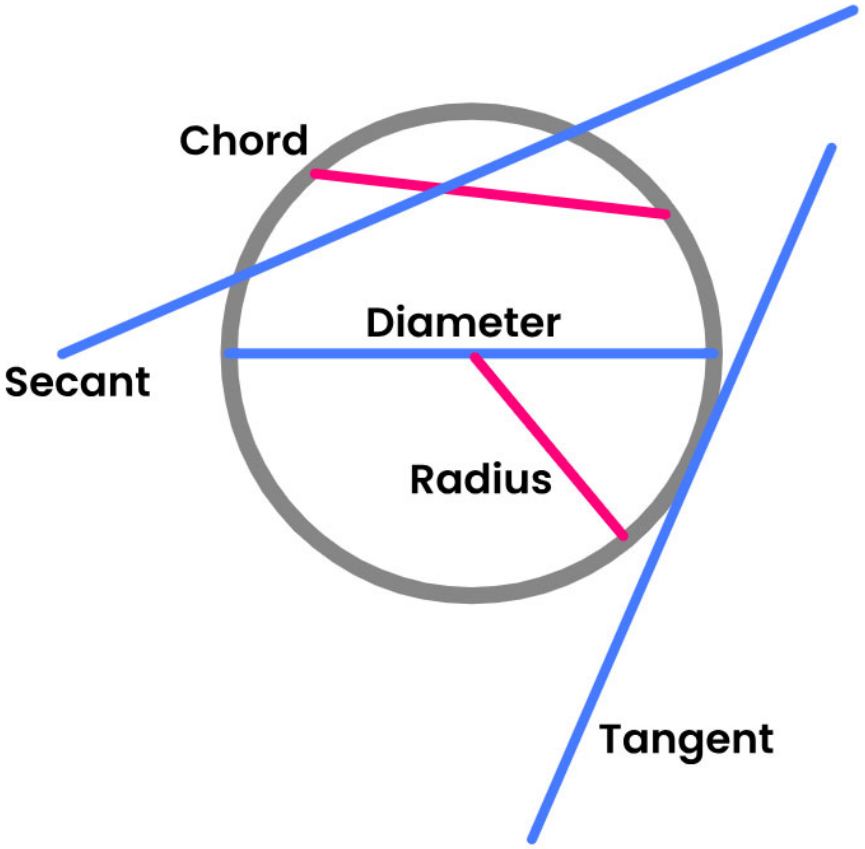
The values of $f(x)$, $f'(x)$ and $f''(x)$ are available.



Secant and Chord

A *secant* is a line that intersects a circle or arc at two points. The part of the secant that is inside the circle or arc is a *chord*.





Secant Method

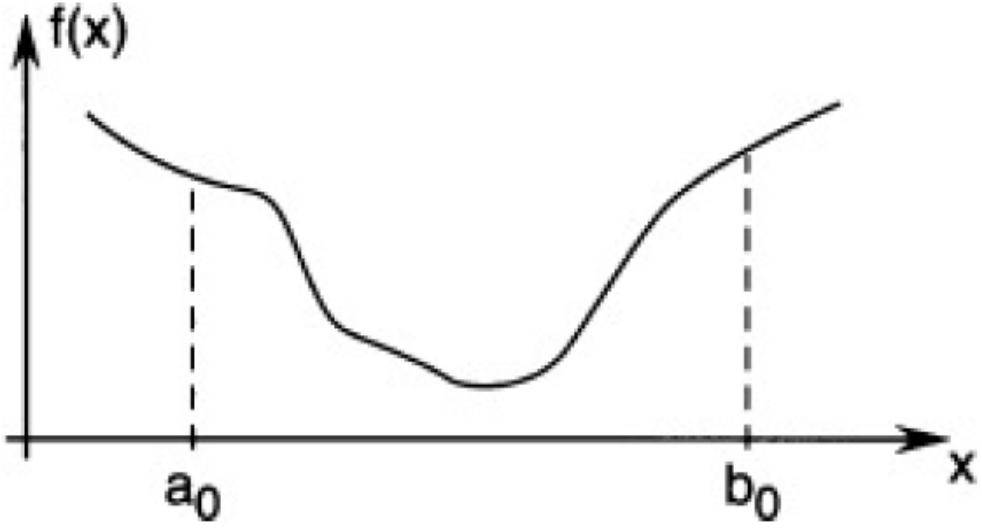
$f(x)$ is unimodal ($f(x)$ has a single local minimum).

The values of $f(x)$ and $f'(x)$ are available at each point.

The value of $f''(x)$ is not available, which is approximated.

In Newton's Methods:

The values of $f(x)$, $f'(x)$ and $f''(x)$ are available.



Secant Method

Newton's method for minimizing f uses second derivatives of f:

$$x^{(k+1)} = x^{(k)} - \frac{f'(x^{(k)})}{f''(x^{(k)})}.$$

If the second derivative is not available, we may attempt to approximate it using first derivative information. In particular, we may approximate $f''(x^{(k)})$ above with $\frac{f'(x^{(k)}) - f'(x^{(k-1)})}{x^{(k)} - x^{(k-1)}}$.

Using the foregoing approximation of the second derivative, we obtain the algorithm

$$x^{(k+1)} = x^{(k)} - \frac{x^{(k)} - x^{(k-1)}}{f'(x^{(k)}) - f'(x^{(k-1)})} f'(x^{(k)}),$$

called the *secant method*. Note that the algorithm requires two initial points to start it, which we denote $x^{(-1)}$ and $x^{(0)}$. The secant algorithm can be represented in the following equivalent form: $x^{(k+1)} = \frac{f'(x^{(k)})x^{(k-1)} - f'(x^{(k-1)})x^{(k)}}{f'(x^{(k)}) - f'(x^{(k-1)})}$.

Observe that, like Newton's method, the secant method does not directly involve values of $f(x^{(k)})$. Instead, it tries to drive the derivative f' to zero. In fact, as we did for Newton's method, we can interpret the secant method as an algorithm for solving equations of the form $g(x) = 0$. Specifically, the secant algorithm for finding a root of the equation $g(x) = 0$ takes the form

$$x^{(k+1)} = x^{(k)} - \frac{x^{(k)} - x^{(k-1)}}{g(x^{(k)}) - g(x^{(k-1)})} g(x^{(k)}), \text{ or, } x^{(k+1)} = \frac{g(x^{(k)})x^{(k-1)} - g(x^{(k-1)})x^{(k)}}{g(x^{(k)}) - g(x^{(k-1)})}.$$

The secant method for root finding is illustrated in Figure 7.10 (compare this with Figure 7.8). Unlike Newton's method, which uses the slope of g to determine the next point, the secant method uses the "secant" between the $(k-1)$th and kth points to determine the $(k+1)$th point.

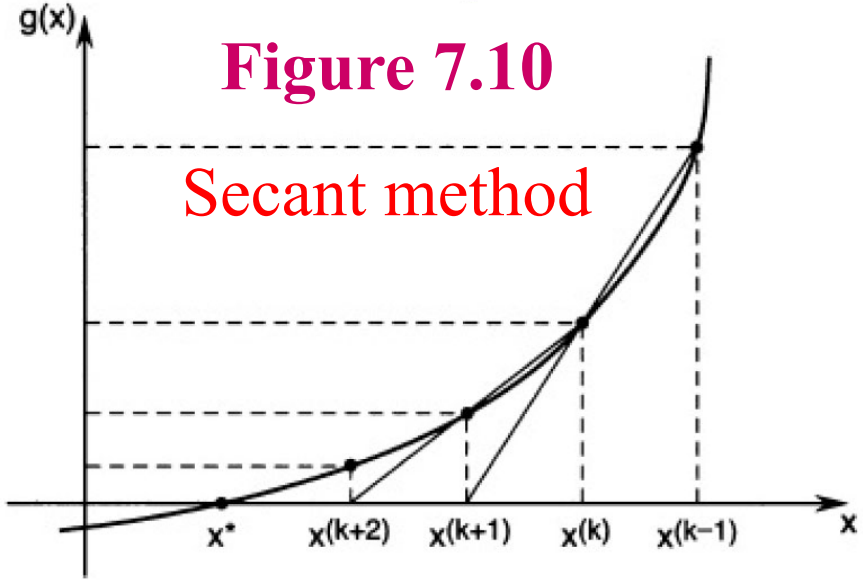


Figure 7.10

Secant method

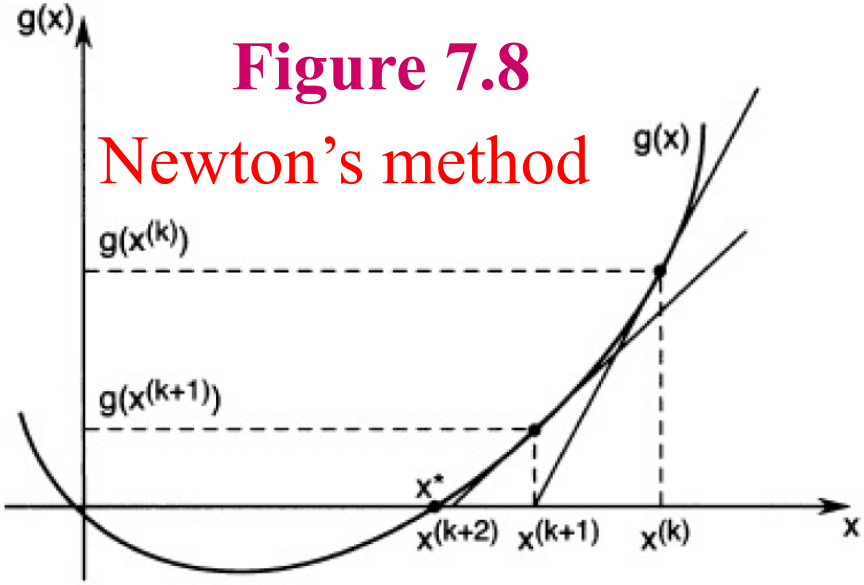


Figure 7.8

Newton's method

Bracketing: How to find a_0 and b_0.

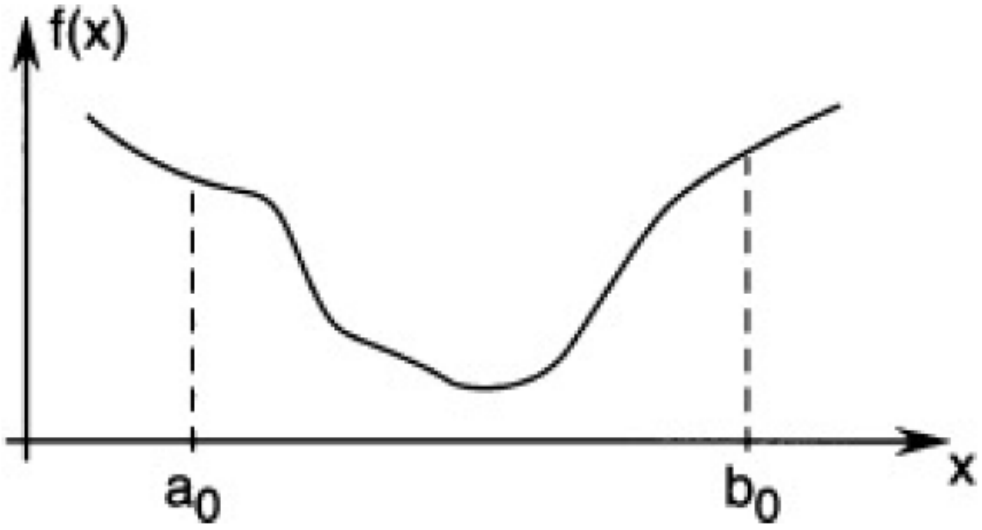
After finding a_0 and b_0, we can use the above mentioned methods such as Golden Section Search.

Golden Section Search

Problem: Minimization of $f(x)$ subject to $a_0 \le x \le b_0$

Assumption: $f(x)$ is unimodal ($f(x)$ has a single local minimum)
We do not know the mathematical form of $f(x)$.

Algorithm (Your Task): To determine $x^{(1)}$, $x^{(2)}$, $x^{(3)}$, ...



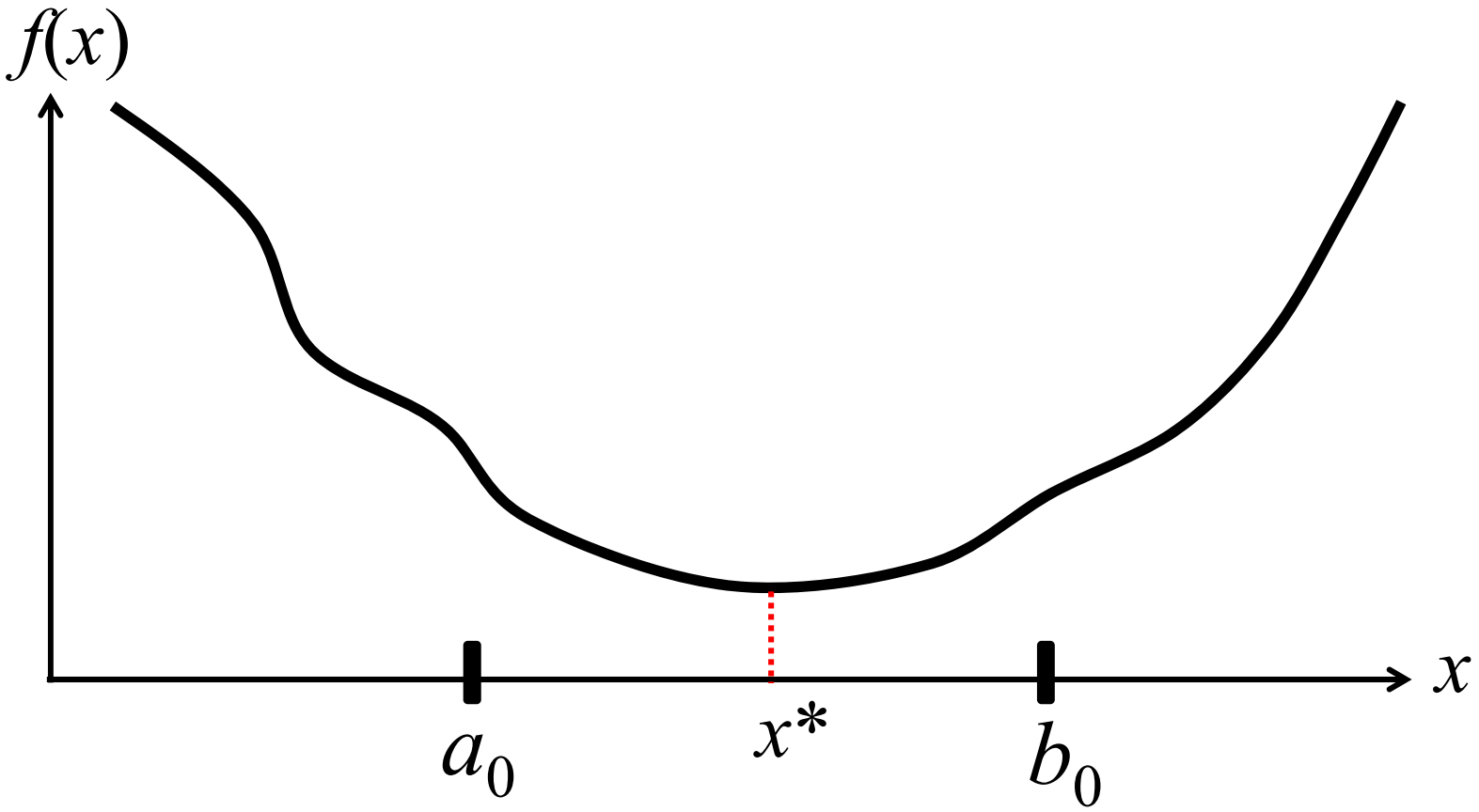
Bracketing

Q. How to find the initial interval $[a_0, b_0]$ such that $a_0 \le x^* \le b_0$

Assumption: Only the value of $f(x)$ is available at each point.

$f(x)$ is unimodal ($f(x)$ has a single local minimum)

Your answer: _____.



Bracketing

Step 1: Generate three points x_0, x_1, x_2 ($x_0 < x_1 < x_2$).

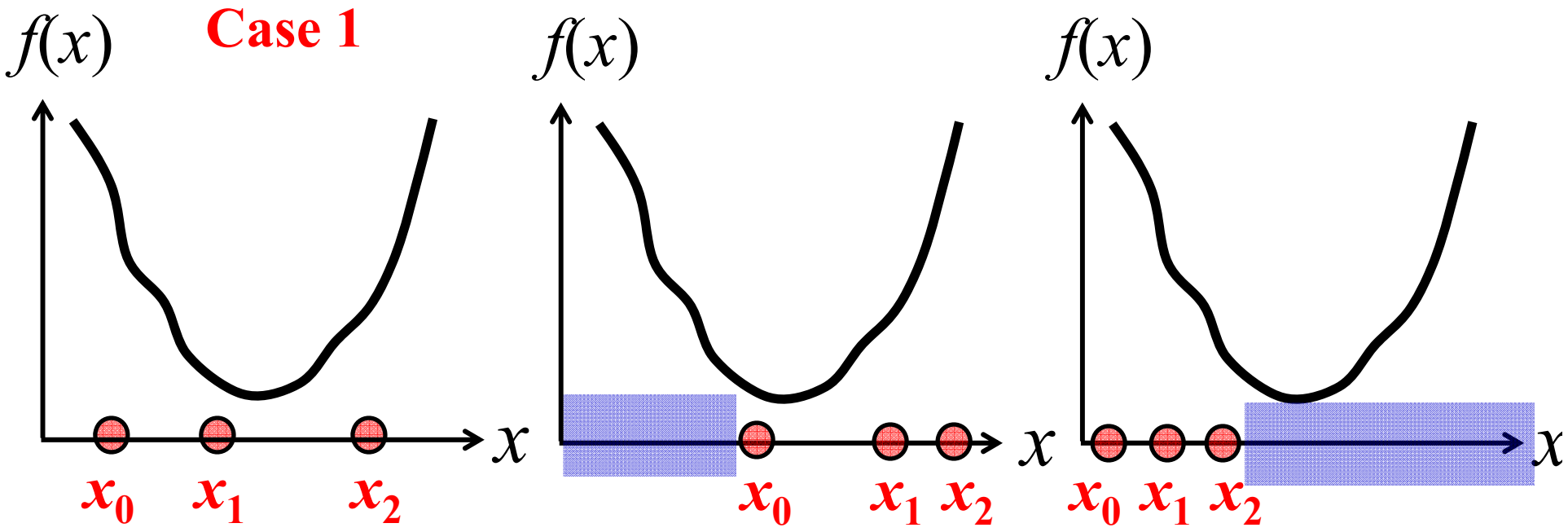
Step 2: Calculate $f(x_0)$, $f(x_1)$, and $f(x_2)$.

Case 1: If $f(x_1)$ is smallest, [x_0, x_2] is the initial interval.

Case 2: If $f(x_0)$ is the smallest, generate a new x_3 ($x_3 < x_0$).

Case 3: If $f(x_2)$ is the smallest, generate a new x_3 ($x_2 < x_3$).

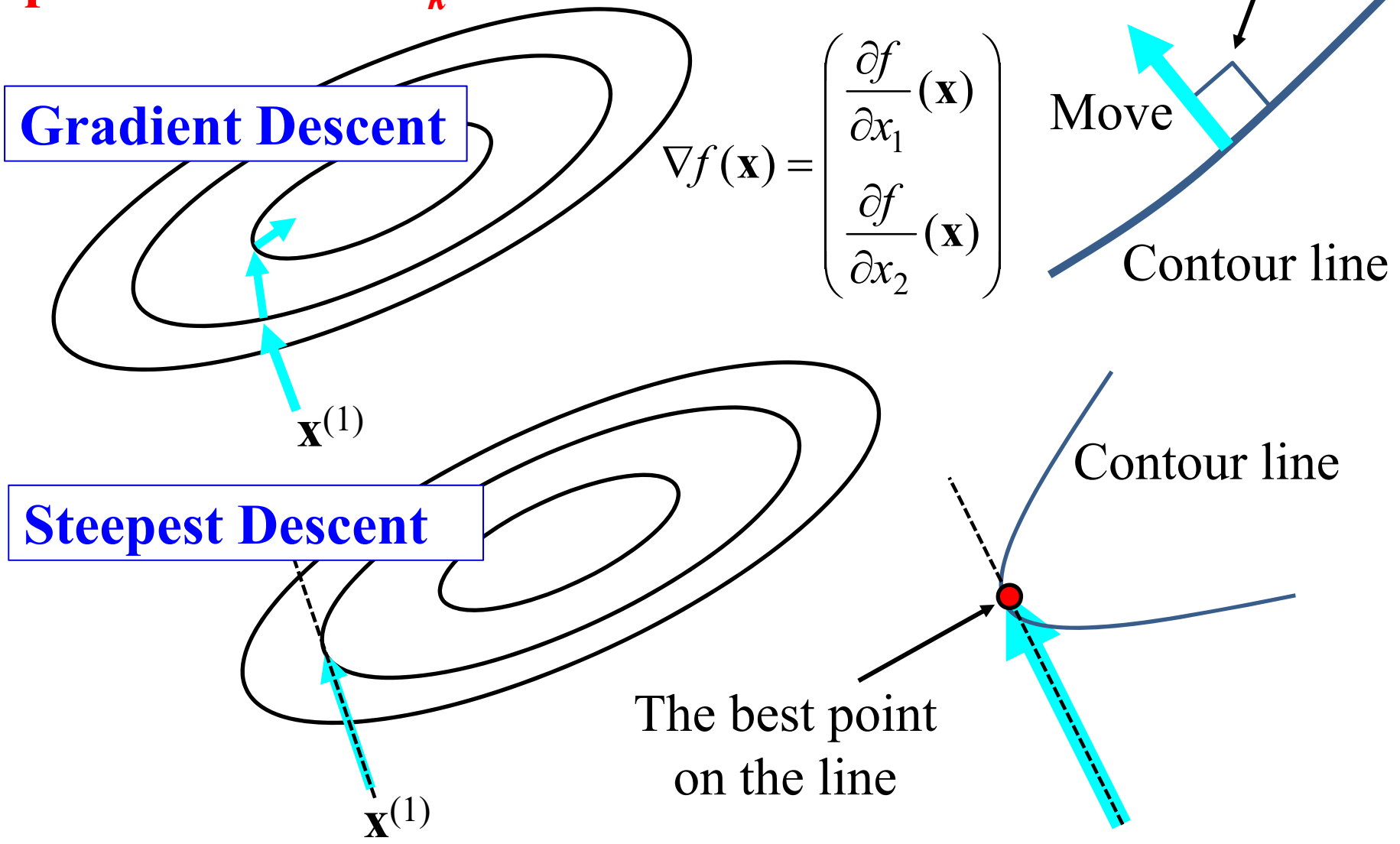
Step 3: Continue to do the same procedure as Step 2.



Line Search in Multidimensional Optimization

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha_k \nabla f(\mathbf{x}^{(k)})$$

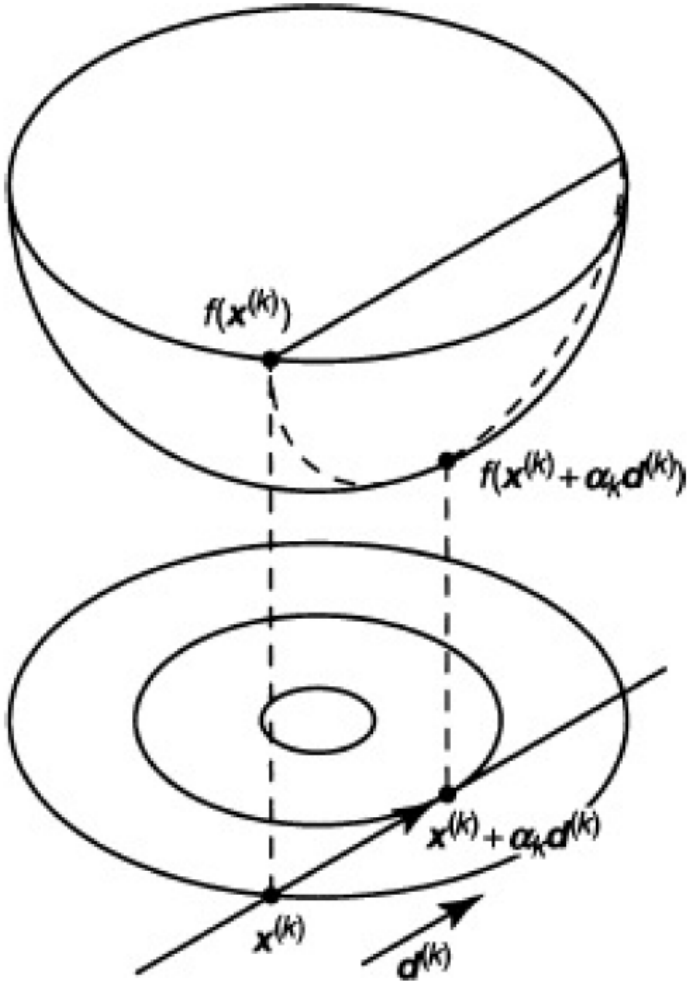
Specification of α_k



Line Search in Multidimensional Optimization

Solution update mechanism: $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}$

α_k: Step size, $\mathbf{d}^{(k)}$: Search direction



Specification of α_k:

Minimization of

$$\phi(\alpha_k) = f(\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)})$$

We can use a one-dimensional line search method to specify α_k.

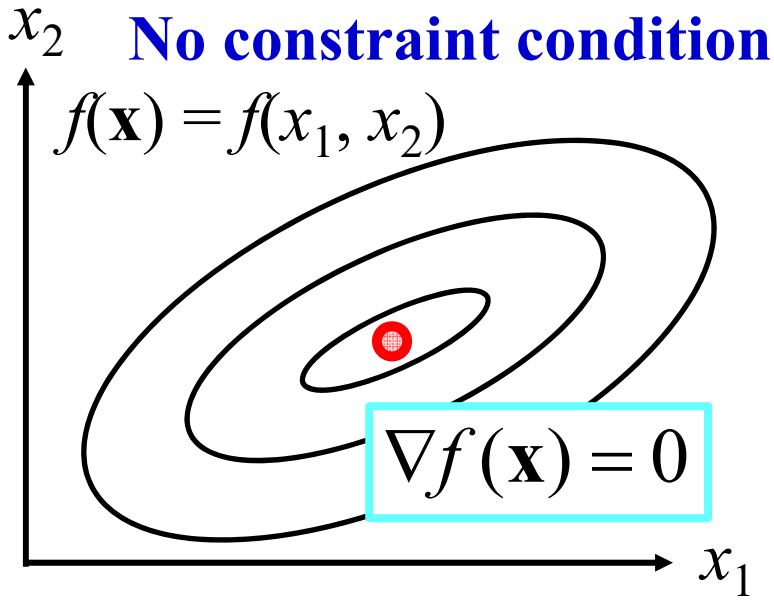
Solution update mechanism: $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}$

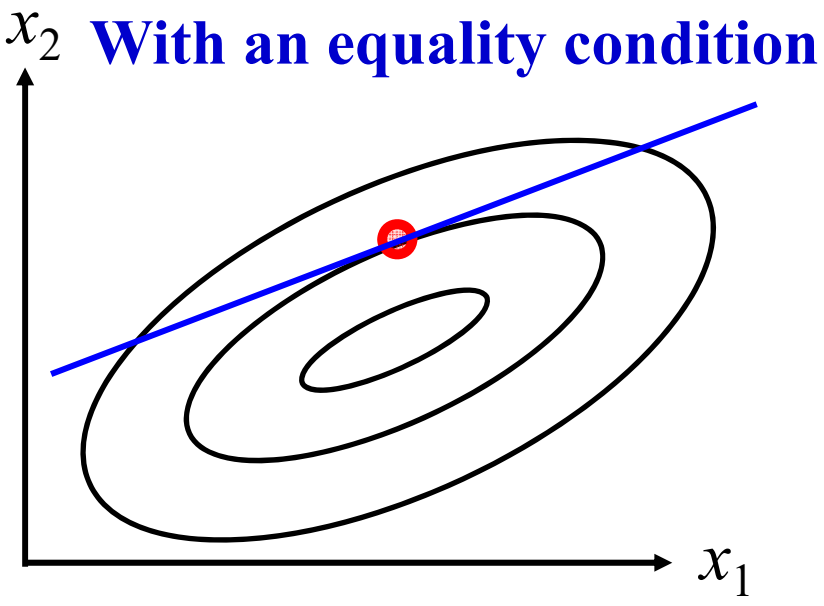
Specification of α_k : Minimization of

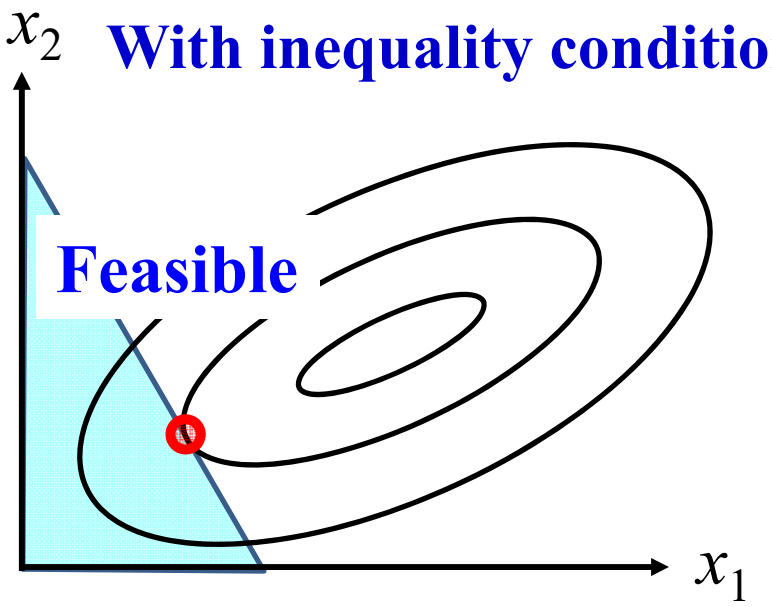
$$\phi(\alpha_k) = f(\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)})$$

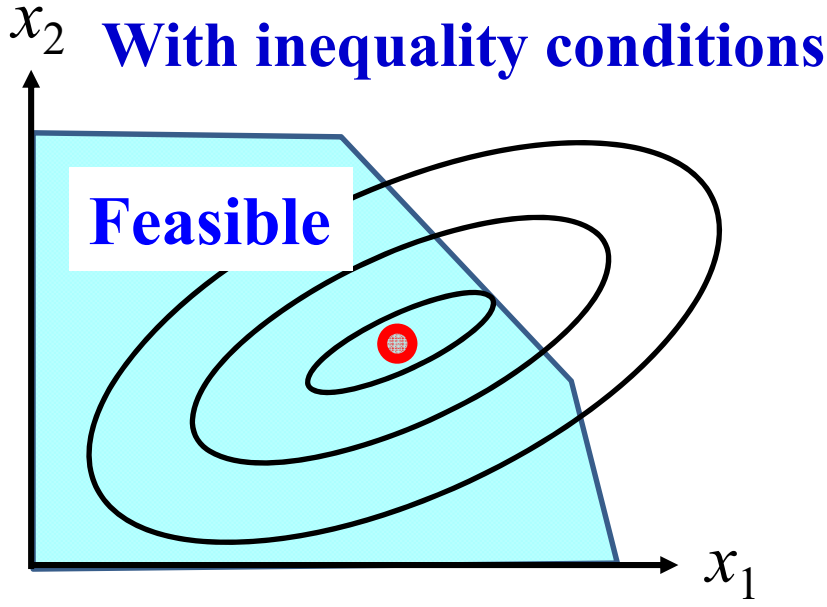
Linesearch algorithms used in practice involve considerations that we have not yet discussed thus far. First, determining the value of α_k that exactly minimizes ϕ_k may be computationally demanding; even worse, the minimizer of ϕ_k may not even exist. Second, practical experience suggests that it is better to allocate more computational time on iterating the multidimensional optimization algorithm rather than performing exact line searches. These considerations led to the development of conditions for terminating linesearch algorithms that would result in low-accuracy line searches while still securing a sufficient decrease in the value of the f from one iteration to the next. The basic idea is that we have to ensure that the step size α_k is not too small or too large.

Optimality Conditions (Very Rough Visual Explanations)

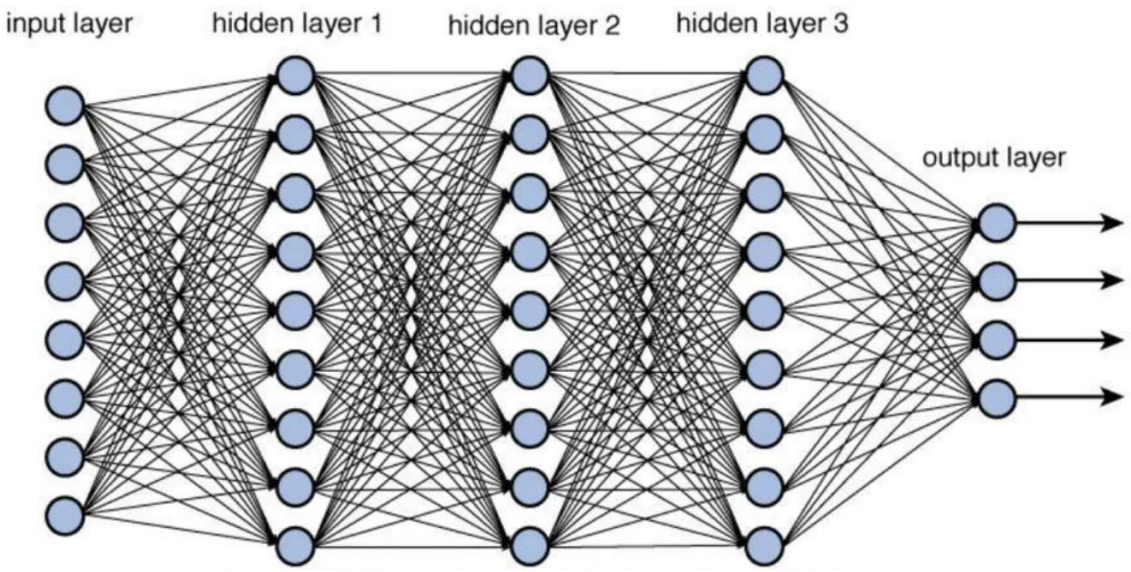








Learning of Neural Networks



Output	Target
0.0245	0
0.1325	0
0.3291	0
0.8234	1

Basic Idea: The error function $E(\boldsymbol{w})$ is defined based on the difference between the desired outputs (i.e., targets) and the actual outputs from the neural network with the weights **w**. The learning of the neural network is to minimize $E(\boldsymbol{w})$ by adjusting **w**.

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \frac{\partial E(\mathbf{w}^{(t)})}{\partial \mathbf{w}^{(t)}}$$

Gradient Descent

with Constant Step Size

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha \nabla f(\mathbf{x}^{(k)})$$

Basic Idea: The error function $E(\boldsymbol{w})$ is defined based on the difference between the desired outputs (i.e., targets) and the actual outputs from the neural network with the weights **w**. The learning of the neural network is to minimize $E(\boldsymbol{w})$ by adjusting **w**.

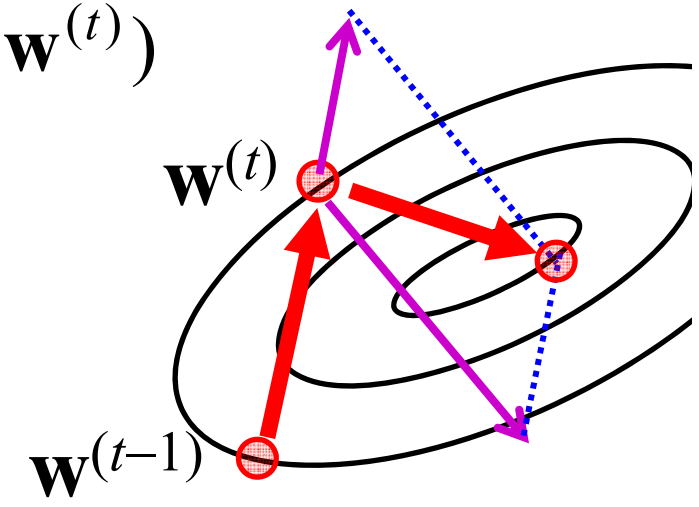
$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \frac{\partial E(\mathbf{w}^{(t)})}{\partial \mathbf{w}^{(t)}}$$

Gradient Descent
with Constant Step Size

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha \nabla f(\mathbf{x}^{(k)})$$

Simple Modification: Momentum Term

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \frac{\partial E(\mathbf{w}^{(t)})}{\partial \mathbf{w}^{(t)}} + \alpha(\mathbf{w}^{(t-1)} - \mathbf{w}^{(t)})$$



Basic Idea: The error function $E(\boldsymbol{w})$ is defined based on the difference between the desired outputs (i.e., targets) and the actual outputs from the neural network with the weights **w**. The learning of the neural network is to minimize $E(\boldsymbol{w})$ by adjusting **w**.

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \frac{\partial E(\mathbf{w}^{(t)})}{\partial \mathbf{w}^{(t)}}$$

Gradient Descent
with Constant Step Size

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha \nabla f(\mathbf{x}^{(k)})$$

Simple Modification: Momentum Term

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \frac{\partial E(\mathbf{w}^{(t)})}{\partial \mathbf{w}^{(t)}} + \alpha(\mathbf{w}^{(t-1)} - \mathbf{w}^{(t)})$$

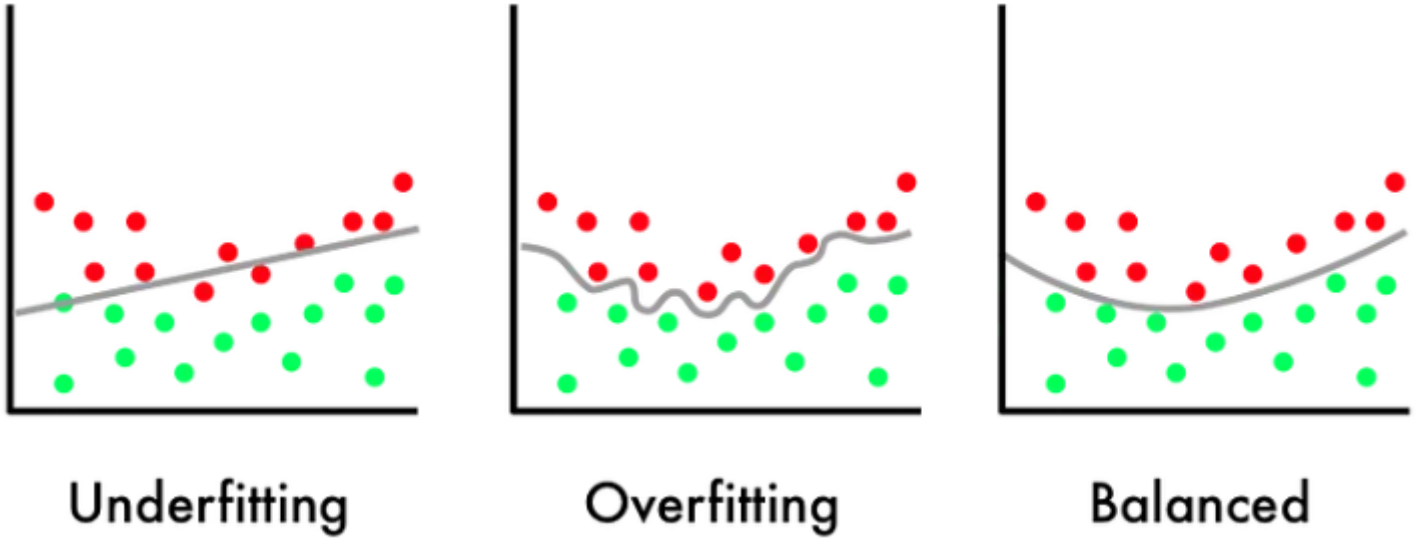
Other Modification

Adaptation of η and α

(Constant step size does not always work well).

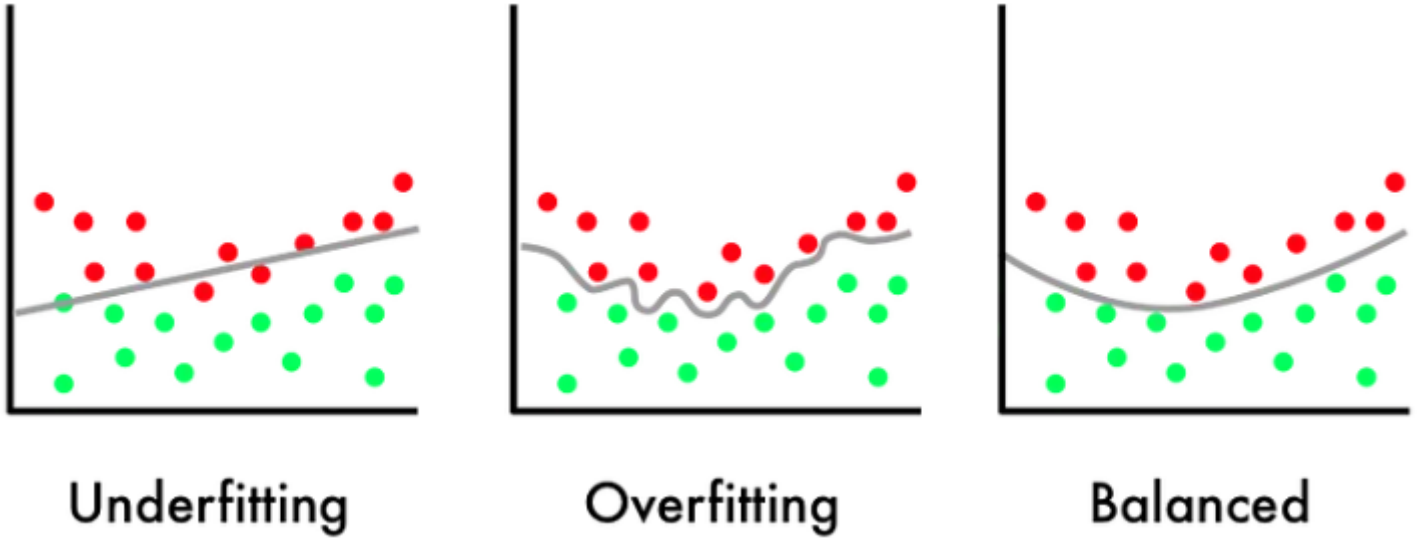
Other Modification: Regularization

$$E(\boldsymbol{w}) \quad \Rightarrow \quad E(\boldsymbol{w}) + \lambda \|\mathbf{w}\|^2$$



Other Modification: Regularization

$$E(\boldsymbol{w}) \quad \Rightarrow \quad E(\boldsymbol{w}) + \lambda \|\mathbf{w}\|^2$$



Single-Objective Problem

Minimization of $E(\boldsymbol{w}) + \lambda \|\mathbf{w}\|^2$

Multi-Objective Problem

Minimization of $E(\boldsymbol{w})$ and $\|\mathbf{w}\|^2$

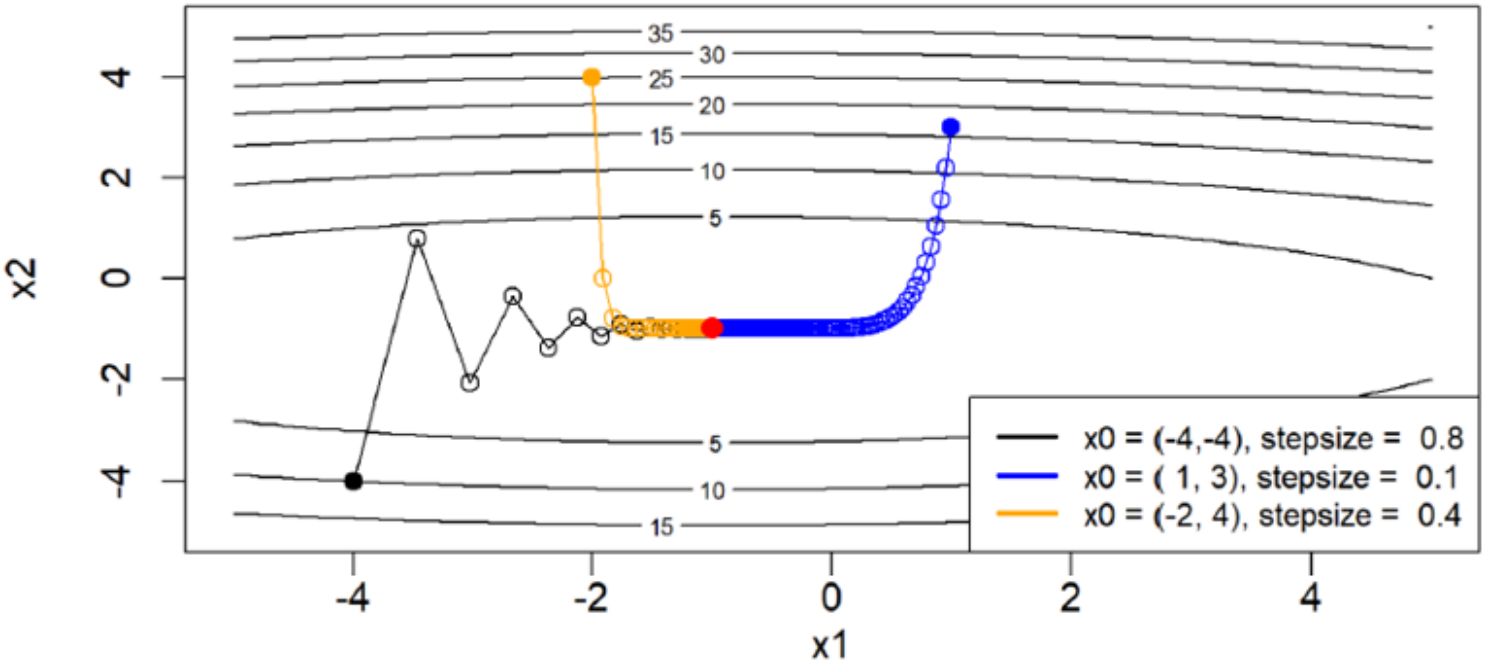
Lab Session Task:

Show positive and negative effects of including the momentum term in the gradient decent algorithm with a constant step size using the following very simple problem and some other test problems.

$$\text{Minimize } f(\mathbf{x}) = (x_1+1)^2/9 + (x_2+1)^2$$

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha \nabla f(\mathbf{x}^{(k)})$$

$$\Rightarrow \quad \mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha \nabla f(\mathbf{x}^{(k)}) + \beta(\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)})$$



Lab Session Task:

Show positive and negative effects of including the momentum term in the gradient decent algorithm with a constant step size using the following very simple problem and some other test problems.

$$\text{Minimize } f(\mathbf{x}) = (x_1+1)^2/9 + (x_2+1)^2$$

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha \nabla f(\mathbf{x}^{(k)})$$

$$\Rightarrow \quad \mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha \nabla f(\mathbf{x}^{(k)}) + \beta(\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)})$$