Applied Linear Algebra in Data Analysis

Introduction to Optimization

Sivakumar Balasubramanian

Department of Bioengineering
Christian Medical College, Bagayam
Vellore 632002

Optimization

- Optimization is the process of finding the best solution to a problem from a set of possible solutions.

- Optimization problems come up in many applications in engineering, science, economics, biology, medicine, operations research, etc.

- Optimization problems can be classified in different ways, but one major classification gives us: **unconstrained** and **constrained** optimization problems.

A general optimization problem

- A general optimization problem can be fomulated as the following,

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{g}(\mathbf{x}) \leq \mathbf{0},\ \mathbf{g}(\mathbf{x}) = \begin{bmatrix} g_1(\mathbf{x}) & g_2(\mathbf{x}) & \cdots & g_p(\mathbf{x}) \end{bmatrix}^\top \\ & \mathbf{h}(\mathbf{x}) = \mathbf{0},\ \mathbf{h}(\mathbf{x}) = \begin{bmatrix} h_1(\mathbf{x}) & h_2(\mathbf{x}) & \cdots & h_q(\mathbf{x}) \end{bmatrix}^\top \end{aligned}$$

  where, $f(\mathbf{x})$ is the **objective function** and $\mathbf{g}(\mathbf{x})$ represents the set of **inquality constaints** and $\mathbf{h}(\mathbf{x})$ represents the set of **equality constraints**.

- In this course, we will only focus on optimization problems over $\mathbb{R}^n$, and mostly problems where the objective function and the constraints are differentiable.

A general optimization problem

- Most optimization problems of practical significance cannot be solved analytically, and we must resort to numerical iterative methods to find a solution.

- We can never solve these problems exactly through numerical means, and must content outselves with finding an approximate “good enough” solution.

Mathematical preliminaries: Sequences and Limits

We first review the notions of continuity and differentiability of functions of single and multiple variables, since we will be dealing with differentiable functions in optimization problems.

Sequences and Limits:

- A sequence of real numbers is a function whose domain is a set of natural numbers $1, 2, \ldots, k, \ldots$ and whose range is a set of real numbers. The sequence is denoted by $\{x_k\}_{k=1}^{\infty}$ or $\{x_k\}$.

- A number x^* is said to be the **limit** of the sequence $\{x_k\}$ if for every $\epsilon > 0$, there exists an integer K such that for all $k > K$, we have $|x_k - x^*| < \epsilon$.

$$\lim_{k \to \infty} x_k = x^\star \quad \text{or} \quad x_k \to x^\star$$

A sequence that has a limit is called a **convergent sequence**.

Sequences and Limits

We can extend these ideas to $\mathbb{R}^n$.

- A sequence in $\mathbb{R}^n$ is a function whose domain is a set of natural numbers $1, 2, \ldots, k, \ldots$ and whose range is $\mathbb{R}^n$. The sequence is denoted by $\{\mathbf{x}_k\}_{k=1}^{\infty}$ or $\{\mathbf{x}_k\}$.

- $\mathbf{x}^*$ is said to be the **limit** of the sequence $\{\mathbf{x}_k\}$ if for every $\epsilon > 0$, there exists an integer K such that for all $k > K$, we have $\|\mathbf{x}_k - \mathbf{x}^*\| < \epsilon$.

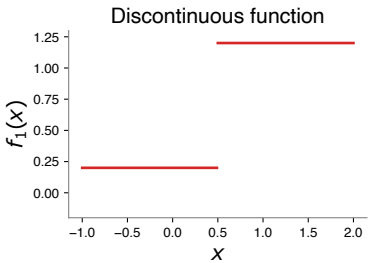
$$\lim_{k \to \infty} \mathbf{x}_k = \mathbf{x}^\star \quad \text{or} \quad \mathbf{x}_k \to \mathbf{x}^\star$$

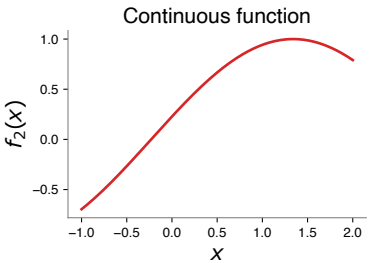
- The limit of a convergent sequence is unique.

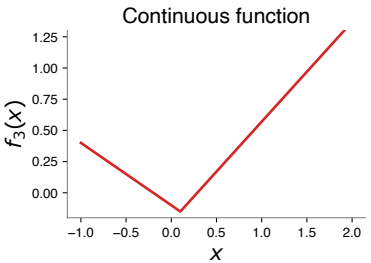
Continuity

Consider the function $f : \Omega \to \mathbb{R}$, where $\Omega \subseteq \mathbb{R}^n$. This function is continuous at the point $\mathbf{x}_0 \in \Omega$, if and only if,

$$\lim_{\mathbf{x} \to \mathbf{x}_0} f(\mathbf{x}) = f(\mathbf{x}_0)$$







Differentiability

Differentiability is a local property of a function, like continuity.
Consider a function $f : \Omega \to \mathbb{R}$, where $\Omega \subseteq \mathbb{R}$. Let $x_0 \in \Omega$,

$$\frac{\delta f\left(x_0\right)}{\delta x} = \frac{f\left(x_0 + \delta x\right) - f\left(x_0\right)}{\delta x}$$

The function f is said to be differentiable at the point $x_0 \in \Omega$, if and only if,

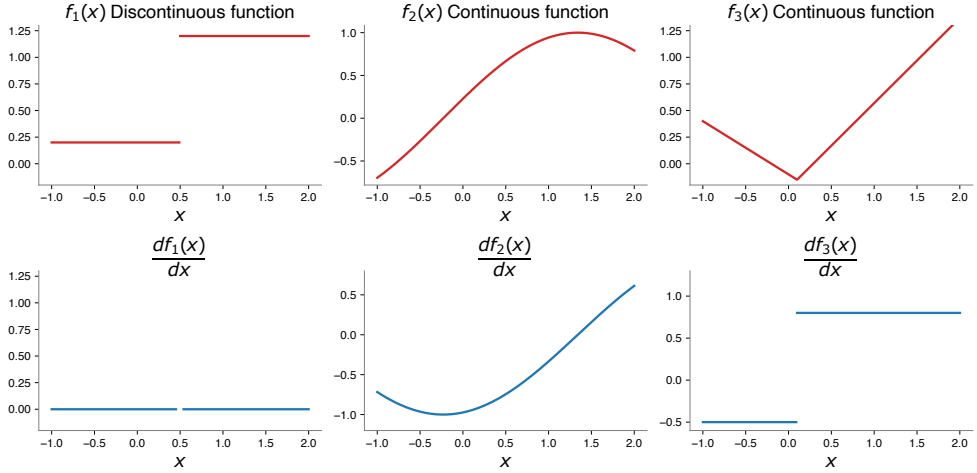
- $f\left(x\right)$ is continuous at x_0.
- $\lim_{\delta x \to 0} \frac{\delta f(x_0)}{\delta x} = \lim_{\delta x \to 0^-} \frac{\delta f(x_0)}{\delta x} = \lim_{\delta x \to 0^+} \frac{\delta f(x_0)}{\delta x}$
- $\lim_{\delta x \to 0} \frac{f(x_0)}{\delta x}$ is finite.

Then the derivative of the function f at the point x_0 is defined as,

$$\frac{df\left(x_0\right)}{dx} = \lim_{\delta x \to 0} \frac{f\left(x_0 + \delta x\right) - f\left(x_0\right)}{\delta x}$$

Differentiability

Three functions f_1, f_2, f_3 defined over the set $\Omega = [-1, 2] \subseteq \mathbb{R}$.



Differentiability in $\mathbb{R}^n$

Consider the function $f : \Omega \to \mathbb{R}$, where $\Omega \subseteq \mathbb{R}^n$.

$$f(\mathbf{x}) = f(x_1, x_2, \ldots, x_n)$$

f maps a column vector $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}^\top \in \mathbb{R}^n$ to a real number.

The partial derivative of the function $f(\mathbf{x})$ at $\mathbf{x}_0$ is defined as,

$$\frac{\partial f(\mathbf{x}_0)}{\partial x_i} = \lim_{\delta x \to 0} \frac{f(\mathbf{x}_0 + \delta x\, \mathbf{e}_i) - f(\mathbf{x}_0)}{\delta x}$$

$\frac{\partial f(\mathbf{x})}{\partial x_i}$ is the rate of change of the function f when move along the i-th coordinate direction at the point $\mathbf{x}_0$.

The function f is said to be differentiable at the point $\mathbf{x}_0 \in \Omega$, if and only if, the partial derivatives of the function f w.r.t. all x_i exist.

Differentiability in $\mathbb{R}^n$

The derivative of the function $f : \Omega \to \mathbb{R}$, where $\Omega \subseteq \mathbb{R}^n$ with respect to the column vector $\mathbf{x}$ at the point $\mathbf{x}_0 \in \Omega$ is defined as the following,

$$\nabla f(\mathbf{x}_0) = \left[\frac{\partial f}{\partial x_1}(\mathbf{x}_0) \quad \frac{\partial f}{\partial x_2}(\mathbf{x}_0) \quad \cdots \quad \frac{\partial f}{\partial x_n}(\mathbf{x}_0) \right] \in \mathbb{R}^n$$

Notice that $\nabla f(\mathbf{x}_0)$ is a row vector, and it is called the *gradient* of the function f at the point $\mathbf{x}_0$.

We follow the following convention when dealing with derivative of functions of multiple variables $f : \Omega \to \mathbb{R}$:

- The gradient with respect to a column vector $\mathbf{x}$ is a row vector $\nabla_{\mathbf{x}} f(\mathbf{x})$.

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = \left[\frac{\partial f}{\partial x_1} \quad \frac{\partial f}{\partial x_2} \quad \cdots \quad \frac{\partial f}{\partial x_n} \right]$$

- The gradient with respect to a row vector $\mathbf{x}^\top$ is a column vector $\nabla_{\mathbf{x}^\top} f(\mathbf{x})$.

$$\nabla_{\mathbf{x}^\top} f(\mathbf{x}) = \left[\frac{\partial f}{\partial x_1} \quad \frac{\partial f}{\partial x_2} \quad \cdots \quad \frac{\partial f}{\partial x_n} \right]^\top$$

Differentiability in $\mathbb{R}^n$: Jacobian of a Vector-valued function

Consider the function $\mathbf{h} : \mathbb{R}^q \to \mathbb{R}^p$, where

$$\mathbf{h}(\mathbf{x}) = \begin{bmatrix} h_1(\mathbf{x}) & h_2(\mathbf{x}) & \cdots & h_p(\mathbf{x}) \end{bmatrix}^\top \quad \mathbf{x} \in \mathbb{R}^q$$

The *Jacobian* of the function $\mathbf{h}(\mathbf{x})$ with respect to $\mathbf{x} \in \mathbb{R}^q$ is defined as the following matrix,

$$\nabla_{\mathbf{x}} \mathbf{h}(\mathbf{x}) \triangleq \begin{bmatrix} \nabla_{\mathbf{x}} h_1(\mathbf{x}) \\ \nabla_{\mathbf{x}} h_2(\mathbf{x}) \\ \vdots \\ \nabla_{\mathbf{x}} h_p(\mathbf{x}) \end{bmatrix} \in \mathbb{R}^{p \times q}$$

Differentiability in $\mathbb{R}^n$: Hessian Matrices

Consider the function $f : \mathbb{R}^n \to \mathbb{R}$ and $\mathbf{x} \in \mathbb{R}^n$.

The Hessian matrix $\mathbf{H}_f(\mathbf{x})$ of the function $f(\mathbf{x})$ is defined as the symmetric matrix $n \times n$ matrix of the second order partial derivatives of f with respect to the components of $\mathbf{x}$, assuming all the second order partial derivatives exists.

The ij^{th} element of the Hessian matrix of $f(\mathbf{x})$ is given by.

$$[\mathbf{H}_f(\mathbf{x})]_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}(\mathbf{x}) = \frac{\partial}{\partial x_i}\left(\frac{\partial f}{\partial x_j}(\mathbf{x})\right) = \frac{\partial}{\partial x_j}\left(\frac{\partial f}{\partial x_i}(\mathbf{x})\right)$$

$$\mathbf{H}_f(\mathbf{x}) \triangleq \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \qquad \mathbf{H}_f(\mathbf{x}) = \nabla_{\mathbf{x}^\top}(\nabla_{\mathbf{x}} f(\mathbf{x})) = \nabla_{\mathbf{x}}(\nabla_{\mathbf{x}^\top} f(\mathbf{x}))$$

Gradient of a function $f : \mathbb{R}^n \to \mathbb{R}$

The levels set of the function $f : \mathbb{R}^n \to \mathbb{R}$ at the level $c \in \mathbb{R}$ is defined as,

$$S = \{\mathbf{x} \mid \mathbf{x} \in \mathbb{R}^n . f(\mathbf{x}) = c\}$$

A level set is a curve for functions $f : \mathbb{R}^2 \to \mathbb{R}$, and is a surface when $f : \mathbb{R}^3 \to \mathbb{R}$.

The different level sets are also called the contours of the function f.

The gradient of the function f at a point $\mathbf{x}_0$ is orthogonal to the level set of the function f at the value $f(\mathbf{x}_0)$.

The gradient is also the direction in $\mathbb{R}^n$ of maximal increase of the value of the function f. This is also called the direction of *steepest ascent*.

Taylor's Theorem

Many results from analysis are used in optimization problems – one of them is the "Taylor's" theorem.
The Taylor's theorem gives an polynomial approximation of a k time differentiaable function $f : \mathbb{R} \to \mathbb{R}$ around at a given point x_0 by a k^{th} order **Taylor polynomial**.
For a smooth function (infinitely differentiable), the k^{th} order Taylor polynomial is a truncation at the order k of the Taylor series expansion of the function f around the point x_0.
Taylor's Theoerem: Suppose a function $f : \mathbb{R} \to \mathbb{R}$ is k times differentiable at a point x_0, then the function f can be approximated by the following polynomial with $\epsilon = x - x_0$,

$$f(x) = f(x_0) + Df(x_0)\frac{\epsilon}{1!} + D^2 f(x_0)\frac{\epsilon^2}{2!} + \cdots + D^{k-1} f(x_0)\frac{\epsilon^{k-1}}{(k-1)!} + D^k f(x_0 + \theta\epsilon)\frac{\epsilon^k}{(k)!}$$

where, $D^l f(x_0)$ is the l^{th} order derivative of the function f at the point x_0, and $0 \leq \theta \leq 1$.

Taylor's Theorem

Now consider a function $f : \mathbb{R}^n \to \mathbb{R}$ and $\mathbf{x}_0 \in \mathbb{R}^n$, and let's assume that f is differentiable twice with respect to $\mathbf{x}$, and let $\boldsymbol{\epsilon} = \mathbf{x} - \mathbf{x}_0$. The polynomial approximation of f is given by,

$$f(\mathbf{x}) = f(\mathbf{x}_0) + \frac{1}{1!}\mathbf{g}(\mathbf{x}_0)^\top \boldsymbol{\epsilon} + \frac{1}{2!}\boldsymbol{\epsilon}^\top \mathbf{H}(\mathbf{x}_0 + \theta\boldsymbol{\epsilon})\,\boldsymbol{\epsilon}$$

where, $\mathbf{g}(\mathbf{x}_0) = \nabla_{\mathbf{x}^\top} f(\mathbf{x}_0)$ is the gradient of the function f with respect to the $\mathbf{x}^\top$ computed at $\mathbf{x}_0$, and $\mathbf{H}(\mathbf{x}_0 + \theta\boldsymbol{\epsilon})$ is the Hessian of the functon f computed at $\mathbf{x}_0 + \theta\boldsymbol{\epsilon}$, and $0 \leq \theta \leq 1$.

Local and Global Minimizers

We distinguih between two types of minimizers of a function $f : \mathbb{R}^n \to \mathbb{R}$: Global and local minimizers.

Global minimizer: A point $\mathbf{x}^\star \in \mathbb{R}^n$ is said to be a *global minimizer* of the function $f(\mathbf{x})$ if and only if, $f(\mathbf{x}^\star) \leq f(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{R}^n - \{\mathbf{x}^\star\}$.

A global minimizer is a *strict global minimizer* if $f(\mathbf{x}^\star) < f(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{R}^n - \{\mathbf{x}^\star\}$.

Local minimizer: A point $\mathbf{x}^\star \in \mathbb{R}^n$ is said to be a *local minimizer* of the function $f(\mathbf{x})$ over the set $\Omega \subset \mathbb{R}^n$, if there exists $\epsilon > 0$ such that $f(\mathbf{x}^\star) \leq f(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{R}^n - \{\mathbf{x}^\star\}$ and $\|\mathbf{x} - \mathbf{x}^\star\| < \epsilon$.

This is a *strict local minimizer* if $f(\mathbf{x}^\star) < f(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{R}^n - \{\mathbf{x}^\star\}$ and $\|\mathbf{x} - \mathbf{x}^\star\| < \epsilon$.

Conditions for Local Minimizers

Consider the twice differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, and with gradient vector $\nabla_{\mathbf{x}^\top} f(\mathbf{x})$ and Hessian matrix $\mathbf{H}(\mathbf{x})$.

First order necessary condition (FONC) for local minimizers: If $\mathbf{x}^\star$ is a local minimizer of f, then

$$\nabla_{\mathbf{x}^\top} f(\mathbf{x}^\star) = \mathbf{0}$$

Second order necessary condition (SONC) for local minimizers: If $\mathbf{x}^\star$ is a local minimizer of f, then

$$\nabla_{\mathbf{x}^\top} f(\mathbf{x}^\star) = \mathbf{0} \quad \text{and} \quad \mathbf{d}^\top \mathbf{H}(\mathbf{x}^\star)\, \mathbf{d} \geq 0,\ \mathbf{d} \in \mathbb{R}^n$$

Second order sufficient condition (SONC) for local minimizers: If $\mathbf{x}^\star$ is a local minimizer of f, then

$$\nabla_{\mathbf{x}^\top} f(\mathbf{x}^\star) = \mathbf{0} \quad \text{and} \quad \mathbf{d}^\top \mathbf{H}(\mathbf{x}^\star)\, \mathbf{d} > 0,\ \mathbf{d} \in \mathbb{R}^n$$

Unconstrained Optimization: Single variable case

Consider a function $f : \mathbb{R} \to \mathbb{R}$, and we are interested in finding the minimizer $x^\star$.

The SOSC for this case is: $\frac{df(x)}{dx} = 0$ and $\frac{d^2 f(x)}{dx^2} > 0$.

We might not be able to solve things analytically even for the single variable case, and will need to resort to iterative approaches. Such methods are called *line search* methods.

Iterative search methods: We start with an initial guess x_0, and then update the guess using a rule,

$$x_{k+1} = x_k + \alpha_k h\left(f\left(x_k\right)\right), \ k = 0, 1, 2, \ldots$$

where, α_k is the step size, and $h\left(f\left(x_k\right)\right)$ is the search direction.

The iteration is continued until some stopping criteria are satisfied.

Unconstrained Optimization: Single variable case

Assume that our current value of x in our search process is x_k where the where the stopping criteria are not satisfied, and we need to continue our search.

Gradient decent: One possible approach toward obtaining the next search value is to use the derivative of the function f at x_k to guide our search.

$$|f'(x_k)| \rightarrow \text{How fast is the function change?}$$
$$\text{sign}\left(f'(x_k)\right) \rightarrow \text{In which direction does the function increase?}$$

Moving in the direction of the negative of the derivative of the function f at x_k is a reasonable choice for the search direction,

$$x_{k+1} = x_k - \alpha_k f'(x_k)$$

$\alpha_k > 0$ is the step size that determines how far we move in the search direction. From the Taylor's theorem, it can be shown that for sufficiently small α_k,

$$f(x_{k+1}) \leq f(x_k)$$

Unconstrained Optimization: Single variable case

Gradient decent: The choice of α_k is crucial.

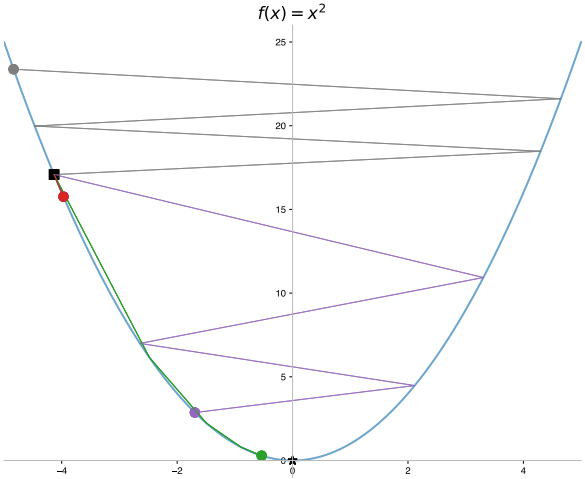
$$\text{small } \alpha_k \rightarrow \text{Slow convergence}$$
$$\text{large } \alpha_k \rightarrow \text{Divergence}$$

An appropriate choice for α_k deepnds on the nature of $f(x)$, i.e. its curvature $\rightarrow f''(x)$.

For a quadrating function $f(x) = ax^2 + bx + c$, there is a upper bound for alpha, beyond which the interative method will diverge.

$$0 < \alpha < \frac{2}{f''(x)} \longrightarrow x_k \text{ will coverge.}$$

Unconstrained Optimization: Single variable case



Effect of Step Size

Iteration $k = 5$

$\alpha_1 = 0.005$

$\alpha_2 = 0.200$

$\alpha_3 = 0.900$

$\alpha_4 = 1.0200$

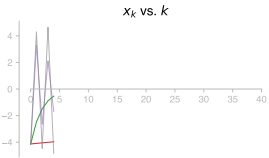
$x^* = 0.0$

x_5 = -3.970

x_5 = -0.536

x_5 = -1.693

x_5 = -4.835



Unconstrained Optimization: Newton's method

One way to address the issue with choosing an appropriate α_k is to make use of the local curvatute of f, i.e. use the second derivative.

One of the most common line search methods is the Newton's methods, which uses the first and second derivatives of the function f to iteratively compute a local minimizer for a function.

At any given iteration k, the Newton's method uses $f(x_k)$, $f'(k)$, and $f''(x_k)$ to fit a quadratic approximation of the function as the following,

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

We mininmize quadratic this approximiation $q(x)$ to find the next guess x_{k+1}. By settinng, $q'(x) = f1(x_k) + f''(x - x_k) = 0$, we get the next guess as,

$$x_{k+1} = x_k - \frac{f`(x_k)'}{f''(x_k)}, \quad \left(\text{Note: } \alpha_k = \frac{1}{f''(x_k)}\right)$$

Unconstrained Optimization: Secant method

What if we did not have access to the f''? We can use an approximation for f'' instead, which gives us the Secant method for line search.

f' **is unknown**: We can use f' to approximate f'' as the following,

$$\hat{f}''(x_k) = \frac{f'(x_k) - f'(x_{k-1})}{x_k - x_k - 1}$$

Using this approximation in the Newton's method and simplifying the expression, we get

$$x_{k+1} = \frac{f'(x_k)\, x_{k-1} - f'(x_{k-1})\, x_k}{f'(x_k) - f'(x_{k-1})}$$

It left as an exercise to shown that x_{k+1} is the minimizer of the quadratic approximation of the function f at the point x_k.

Unconstrained Optimization: Secant method

Both the Newton's and Secant methods are examples of *quadratic fit* methods.

A third possible method foregoes the requirement of the first derivative f' and use only the value of the function at three points to fit a quadratic approximation.

The derivation of the iteration rule for this method is left as an exercise.

Unconstrained Optimization: Newton's method

When do the Newton's or the Secant method fail? When $f''(x_k) \leq 0$.

$$x_{k+1} = x_k - \frac{f`(x_k)'}{f''(x_k)}$$

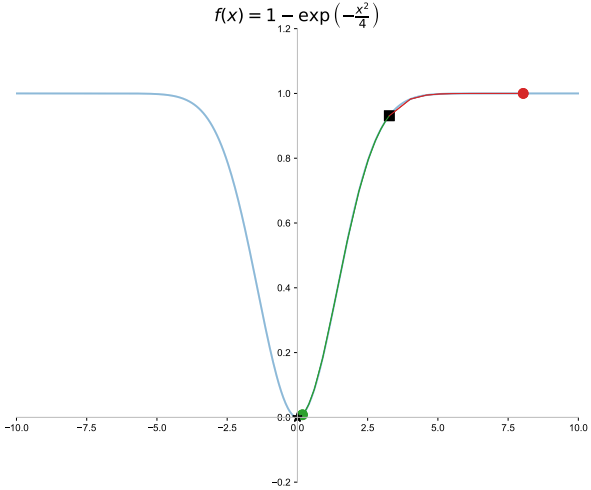
- $f''(x_k) \approx 0$: Step size becomes large.
- $f''(x_k) < 0$: Step size changes sign.

One simple way to address this is to never let the demoninator become too small or negative. This can be achieved by adding a small positive number to the denominator,

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k) + \lambda}, \quad \lambda > 0$$

The parameter λ is called the *damping parameter*.

Unconstrained Optimization: Newton's method



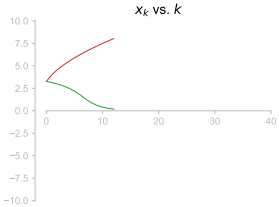
Netwon's Method

$x_{13} = 8.038$
$f(x_{13}) = 1.000$
$f'(x_{13}) = 0.000$
$f''(x_{13}) = -0.000$

Levenberg-Marquardt Method

$x_{13} = 0.179$
$f(x_{13}) = 0.008$
$f'(x_{13}) = 0.089$
$f''(x_{13}) = 0.488$

$x^* = 0.0$



Multivariate Unconstrained Optimization

We need to make two decisions in the multivariate case,

- The search direction $\mathbf{d}_k$.
- The step size α_k.

Once a search direction $\mathbf{d}_k$ is chosen at each iteration step, then the problem is equivalent to the univariate case.

We search along that line in the direction $\mathbf{d}_k$. This is often called the *line search* method.

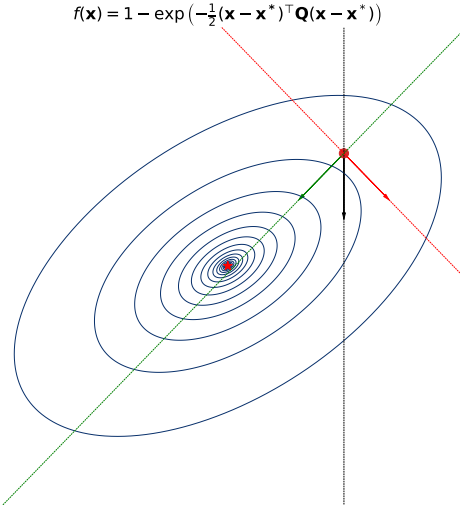
Iterative methods for multivariate optimization are of the following form,

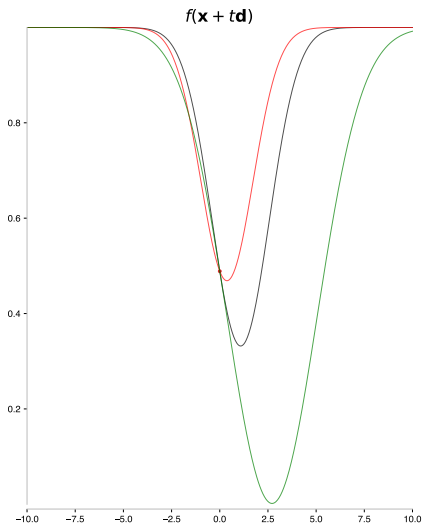
$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$$

where, $\mathbf{d}_k \in \mathbb{R}^n$ is the search direction, and α_k is the step size.

$\mathbf{d}_k$ is chosen based on the local information of the function f at the point $\mathbf{x}_k$. We choose a search direction that will lead to a decrease in the value of the function f.

Multivariate Unconstrained Optimization





Multivariate Unconstrained Optimization

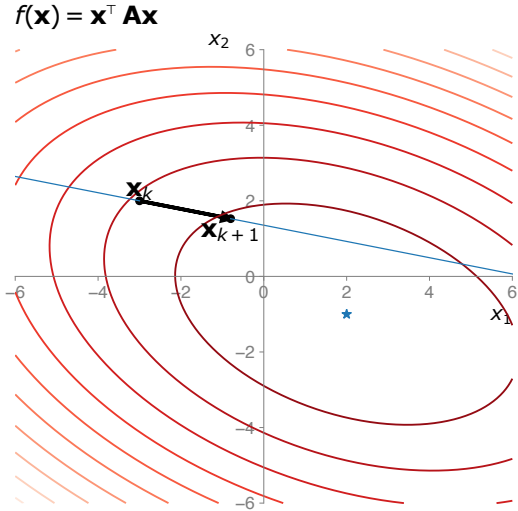
The line search algorithm is shown in the figure.

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$$

The arrow indicates that change $\alpha_k \mathbf{d}_k$. Choosing α_k is a critical step and is often posed as a one-dimensional optimization problem at each iteration k,

$$\alpha_k^\star = \arg \min_{\alpha_k \in \mathbb{R}_{\geq 0}} f\left(\mathbf{x}_k + \alpha_k \mathbf{d}_k\right)$$

An "exact line search" will attempt to solve for $\alpha_k^\star$. But this is not necessary in practice.



Multivariate Unconstrained Optimization

Inexact line searches are used in practice and with more focus on the original objective of minimizing $f(\mathbf{x})$.

Let $\phi_k(\alpha_k) = f(\mathbf{x}_k + \alpha_k \mathbf{d}_k)$. **Backtracking algorithm**

(1) Given $\mathbf{x}_k, \mathbf{d}_k, \alpha_{init}$.

(2) Initialize $\alpha^{(0)} = \alpha_{init}$ and $l = 0$.

(3) Until $\phi\left(\alpha^l\right) \leq \phi(0)$

(i) Set $\alpha^{(l+1)} = \tau\alpha^{(l)}$, where $\tau \in (0,1)$ is fixed $\left(\tau = \frac{1}{2}\right)$.

(ii) $l = l + 1$

(4) Set $\alpha_k = \alpha^{(l)}$.

This algorithm prevents the step size from becoming too small, but does not prevent it from becoming too large.

Multivariate Unconstrained Optimization

A commonly used condition is the *Armijo-Goldstein* condition, which ensures that the step size α_k is neihter too large nor too small.

Ensure α_k is not too large: Let $\epsilon \in (0, 1)$ and $\eta \in (\epsilon, 1)$,

$$\text{Condition 1} : \phi_k(\alpha_k) \leq \phi_k(0) + \epsilon \alpha_k \phi'(0)$$

Ensure α_k is not too small

$$\text{Condition 2} : \phi_k(\alpha_k) \geq \phi_k(0) + \eta \alpha_k \phi_k'(0)$$

Multivariate Unconstrained Optimization

Armijo backtracking algorithm

(1) Given $\mathbf{x}_k, \mathbf{d}_k, \alpha_{init}$.

(2) Initialize $\alpha^{(0)} = \alpha_{init}$, $l = 0$, and $\epsilon \in (0, 1)$.

(3) Until $\phi\left(\alpha^l\right) \leq \phi(0) + \epsilon\alpha^l\phi'(0)$

(i) Set $\alpha^{(l+1)} = \tau\alpha^{(l)}$, where $\tau \in (0, 1)$ is fixed $\left(\tau = \frac{1}{2}\right)$.

(ii) $l = l + 1$

(4) Set $\alpha_k = \alpha^{(l)}$.

This alorithm prevents the step size from becoming too small, but does not prevent it from becoming too large.

Multivariate Unconstrained Optimization

Consider the function $f : \mathbb{R}^n \to \mathbb{R}$ and $\mathbf{x} \in \mathbb{R}^n$.

The best search direction $\mathbf{d}_k$ at at the point $\mathbf{x}_k$ is the direction that maximally minimizes the value of the function f at the point $\mathbf{x}_k$.

The directional derivative of the function $f(\mathbf{x})$ at the point $\mathbf{x}$ along the direction $\mathbf{d}$ is defined as,

$$\lim_{\alpha \to 0} \frac{f(\mathbf{x} + \alpha \mathbf{d}) - f(\mathbf{x})}{\alpha} = \mathbf{d}^\top \nabla_{\mathbf{x}^\top} f(\mathbf{x})$$

We will use $\nabla f(\mathbf{x})$ to denote $\nabla_{\mathbf{x}^\top} f(\mathbf{x})$ from this point foward.

$$\|\mathbf{d}\|_2 = 1 \implies \mathbf{d}^\top \nabla f(\mathbf{x}) \leq \|\nabla f(\mathbf{x})\| \implies \mathbf{d} = \frac{\nabla f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|}$$

Thus, we have,

$$f(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k)) = f(\mathbf{x}_k) - \alpha \|\nabla f(\mathbf{x}_k)\|_2^2 + o(\alpha)$$

Multivariate Unconstrained Optimization

Thus, we have,

$$f\left(\mathbf{x}_k - \alpha \nabla f\left(\mathbf{x}_k\right)\right) = f\left(\mathbf{x}_k\right) - \alpha \| \nabla f\left(\mathbf{x}_k\right) \|_2^2 + o\left(\alpha\right)$$

For a small enough value for $\alpha > 0$, we have,

$$f\left(\mathbf{x}_k - \alpha \nabla f\left(\mathbf{x}_k\right)\right) < f\left(\mathbf{x}_k\right)$$

Thus, the movement along the direction of the negative gradient of the function f at the point $\mathbf{x}_k$ will lead to a decrease in the value of the function f.

Multivariate Unconstrained Optimization: Gradient descent

The gradient descent algorithm with a fixed step size can be used for minimizing the function $f(\mathbf{x})$.

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

The value of α is a simple approach, but runs into similar problem as in the single variable case.

The above equation can be thought of as n separate single variable optimization problems, and a fixed α might is unlikely to be the best choice for all of them. The i^{th} component of the vector $\mathbf{x}_k$ will be updated as,

$$x_{k+1,i} = x_{k,i} - \alpha \frac{\partial f}{\partial x_i}(\mathbf{x}_k)$$

where $\mathbf{x}_k = \begin{bmatrix} x_{k,1} & x_{k,2} & \cdots & x_{k,n} \end{bmatrix}^\top$.

Multivariate Unconstrained Optimization: Gradient descent

Just as in the single variable case, an appropriate choice of α will depend on the curvature of the function $f(\mathbf{x})$, which is given by the Hessian matrix $\mathbf{H}(\mathbf{x})$.

For a quadratic function $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{H}\mathbf{x} + \mathbf{b}^\top \mathbf{x} + c$, the following choice for the step size will ensure convergence of $\mathbf{x}_k$ to the minimizer of the function $f(\mathbf{x})$.

$$0 < \alpha < \frac{2}{\lambda_{max}(\mathbf{H})}, \ \mathbf{H} \geq 0$$

where, $\lambda_{max}(\mathbf{H})$ is the largest eigenvalue of the Hessian matrix $\mathbf{H}$.

Multivariate Unconstrained Optimization: Steepest descent

At the k^{th} iteration, the *steepest descent* method chooses the search direction $\mathbf{d}_k = -\nabla f(\mathbf{x}_k)$, and the optimal step size $\alpha_k^\star$ by performing an exact line search.

$$\alpha_k^\star = \arg \min_{\alpha_k \in \mathbb{R}_{\geq 0}} f(\mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k))$$

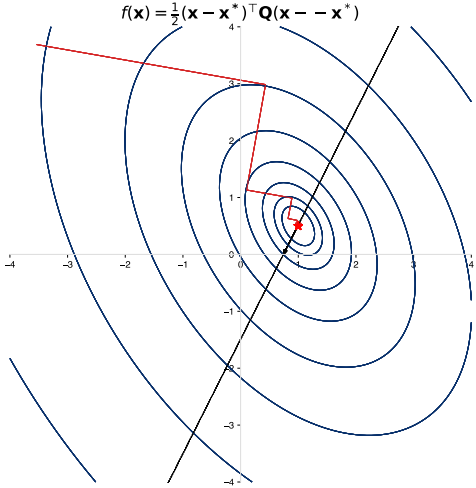
$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k^\star \nabla f(\mathbf{x}_k)$$

Steepest descent has the following interesting property:

$$\mathbf{d}_{k+1}^\top \mathbf{d}_k = \nabla f(\mathbf{x}_{k+1})^\top \nabla f(\mathbf{x}_k) = 0$$

The search direction at two consecutive iterations are orthogonal to each other. Refer to the figure in the next slide.

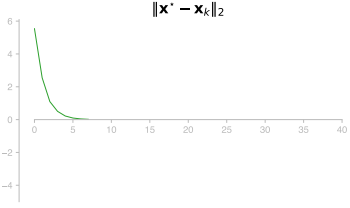
Multivariate Unconstrained Optimization



Use the right arrow key to iterate.
Use 0-2 to select function $f(\mathbf{x})$.
Use the left mouse click to select a location.
Use 'r' to reset search.

Gradient Descent: Backtracking

Iteration $k = 3099$
Step size $\alpha_{3099} = 0.413$
$\mathbf{x}^* = [1.000\ 0.500]^\top$
$\mathbf{x}_{3099} = [1.000\ 0.519]^\top$
$f(\mathbf{x}_{3099}) = 0.000$



Multivariate Unconstrained Optimization: Newton's method

The Newton's method explicitly accounts for the curvature of the function $f(\mathbf{x})$ at the point $\mathbf{x}_k$.

The generalization of the Newton's method to the multivariate case is as follows,

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

This can be derived the same way we did for the univariate case, where at the point $\mathbf{x}_k$ we fit a quadratic approximation of the function $f(\mathbf{x})$ and minimize it to find the next guess $\mathbf{x}_{k+1}$.

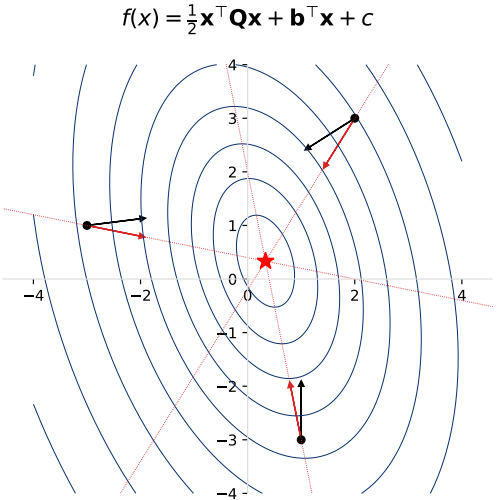
For quardatic $f(\mathbf{x})$, the Newtwon's method will find the minimum in a single iteration by exactly accouting the surface's curvature.

Multivariate Unconstrained Optimization: Newton's method

The figure shows the directions of $-\nabla f(\mathbf{x}_k)$ (in black) and $-\mathbf{H}(\mathbf{x}_k)^{-1}\nabla f(\mathbf{x}_k)$ (in red) at three different points for a quadratic function $f(\mathbf{x})$.

The correction for the curvature ensures that the direction of movement is always pointing towards the location of the function's minimum $\mathbf{x}^\star = -\mathbf{Q}^{-1}\mathbf{b}$.

However, the Newton's method works only when $\mathbf{H}(\mathbf{x}_k)$ is positive definite.

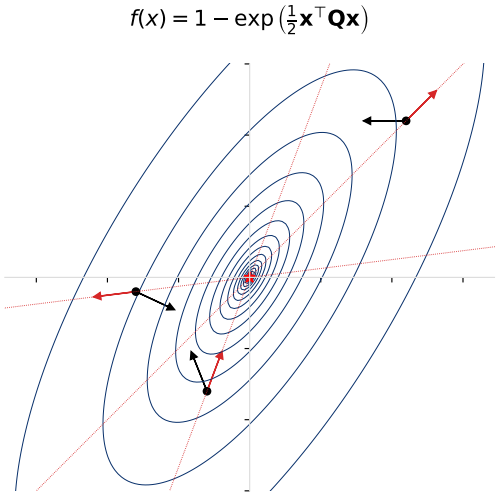


Multivariate Unconstrained Optimization: Newton's method

This is an example where the Newton's method fails when it does not start very close to the minimum.

The red arrows point away from the minimum at two points that are father away from the minimum, because the Hessian matrix $\mathbf{H}(\mathbf{x}_k)$ is not positive definite at these points.

When the Hessian is not positive definite, the gradient vector will get rotated by more then ± 90 deg and thus the search direction points away from the -ve gradient direction.



Multivariate Unconstrained Optimization: Levengerg-Marquardt method

The Levengerg-Marquardt method is a modification of the Newton's method and is often employed for solving nonlinear least squares problems.

The update equation for the Levengerg-Marquardt method is given by,

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \left(\mathbf{H}\left(\mathbf{x}_k\right) + \lambda \mathbf{I}\right)^{-1} \nabla f\left(\mathbf{x}_k\right)$$

where, $\lambda > 0$ is a damping parameter, and $\mathbf{I}$ is the identity matrix.

λ is chosen so that the Hessian matrix $\mathbf{H}\left(\mathbf{x}_k\right) + \lambda \mathbf{I}$ is positive definite, thus ensuring convergence.

The downside of the modification is that the "step size" is lower than the Newton's method (when $\mathbf{H}\left(\mathbf{x}_k\right) \geq 0$), and thus the convergence is slower.

$$\|\mathbf{H}\left(\mathbf{x}_k\right)^{-1}\|_2 < \|\left(\mathbf{H}\left(\mathbf{x}_k\right) + \lambda \mathbf{I}\right)^{-1}\|_2$$

Multivariate Unconstrained Optimization: Levengerg-Marquardt method

The blue arrows indicate the search direction for the Levengerg-Marquardt method, and the red arrows indicate the search direction for the Newton's method.

When $\lambda = 0$, we have the Newton's method, and when $\lambda \to \infty$, the search direction becomes the negative gradient direction.