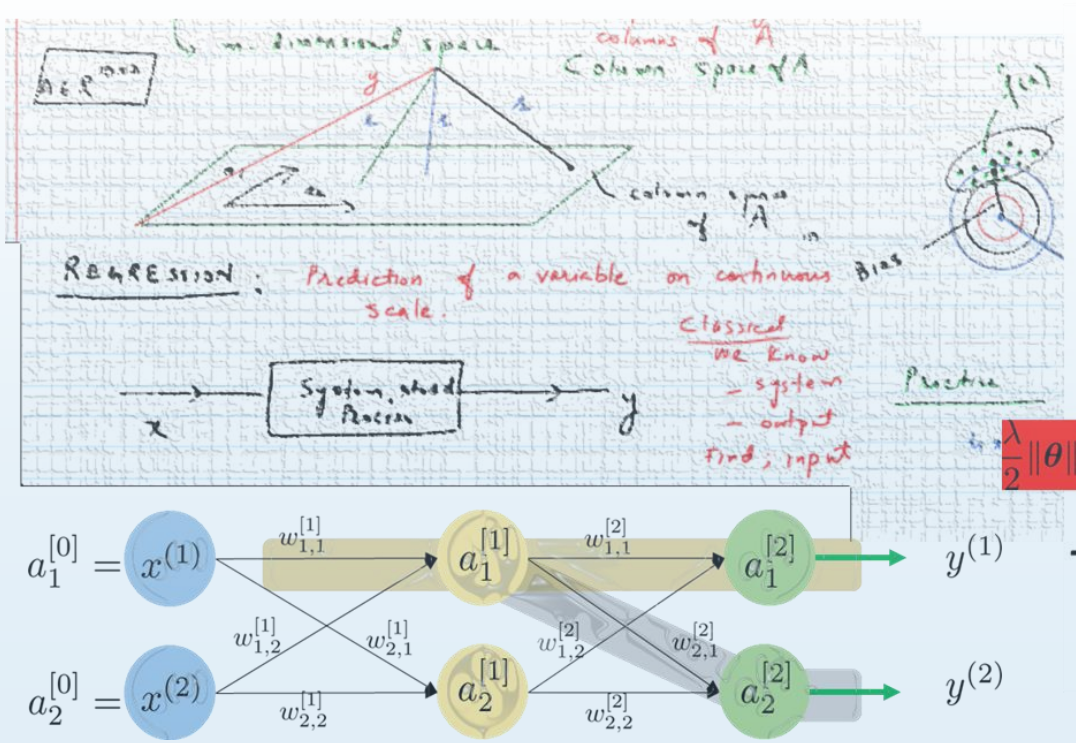
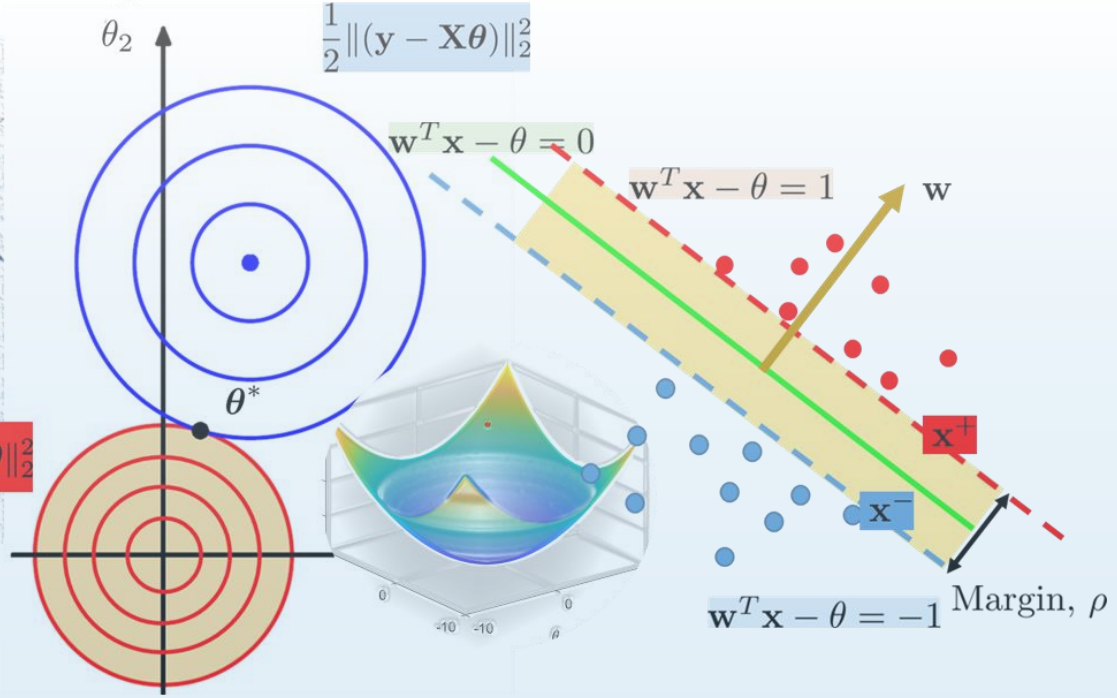




AI-501 Mathematics for AI

Vector Calculus

Zubair Khalid
School of Science and Engineering

https://www.zubairkhalid.org/ai501_2024.html

LUMS
A Not-for-Profit University

Outline

- Functions
- Differentiation
- Convex Functions
- Gradient Descent Algorithm

Calculus

Functions:

A **function** represents a relationship between a set of inputs and a set of possible outputs, where each input is related to exactly one output. Functions are commonly denoted as

$$f : \mathbb{R}^n \to \mathbb{R}^m$$

where:

n represents the dimension of the input space, and

m represents the dimension of the output space.

Calculus

Functions:

A **function** represents a relationship between a set of inputs and a set of possible outputs, where each input is related to exactly one output. Functions are commonly denoted as

$$f : \mathbb{R}^n \to \mathbb{R}^m$$

where:

n represents the dimension of the input space, and

m represents the dimension of the output space.

Domain: This is the set of all possible inputs a function can take. For a function $f : \mathbb{R}^n \to \mathbb{R}^m$, the domain is a subset of $\mathbb{R}^n$.

Co-Domain: This is the set in which the output values of the function are expected to lie. For example, in $f : \mathbb{R}^n \to \mathbb{R}^m$, the co-domain is $\mathbb{R}^m$.

Image: The image of a function (sometimes also called the "range") is the set of all actual values that f maps to within the co-domain.

Calculus

Functions:

Scalar-Valued Functions:

- When $m = 1$, the function's output is a single scalar value.

Vector-Valued Functions:

- When $m > 1$, $f : \mathbb{R}^n \to \mathbb{R}^m$ means the output is a vector in $\mathbb{R}^m$.

Univariate Functions ($n = 1$):

Bivariate Functions ($n = 2$):

- If $n = 2$, the function has two input variables, often written as $f(x, y)$.

Multivariate Functions ($n > 2$):

- When $n > 2$, the function takes multiple inputs, often denoted as vectors, such as $f(\mathbf{x})$ where $\mathbf{x} \in \mathbb{R}^n$.

Calculus

Functions – Examples:

Linear function for $n = 1$, $f(x) = ax$

Linear function for $n = 1$, $f(x) = ax + b$

Linear function for n, $f(x) = a^T x$

Linear function for n, $f(x) = a^T x + b$

Quadratic n, $f(x) = x^T x$

Generalized Quadratic, $f(x) = x^T Ax$

Calculus

Functions – Examples:

In polynomial regression, we used **polynomial function** to capture non-linear relationships. A polynomial function of degree p is:

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_p x^p$$

where $a_0, a_1, \ldots, a_p$ are coefficients.

Outline

- Functions
- **Differentiation**
- Convex Functions
- Gradient Descent Algorithm

Calculus

Functions – Differentiation:

Differentiation allows us to compute derivatives of the function.

Derivative of a function simply means:

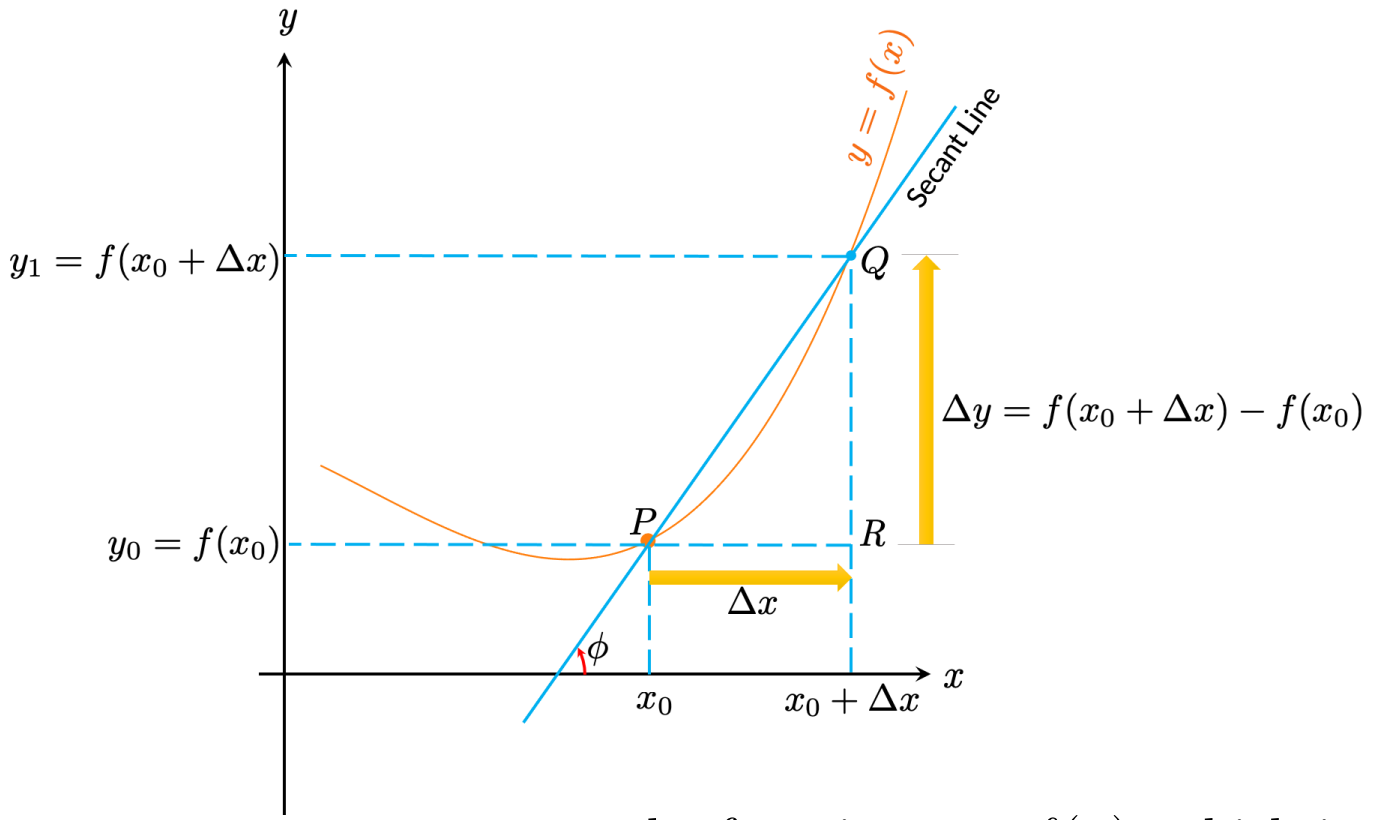
Rate of a change

Zoom in

Localized Information

Calculus

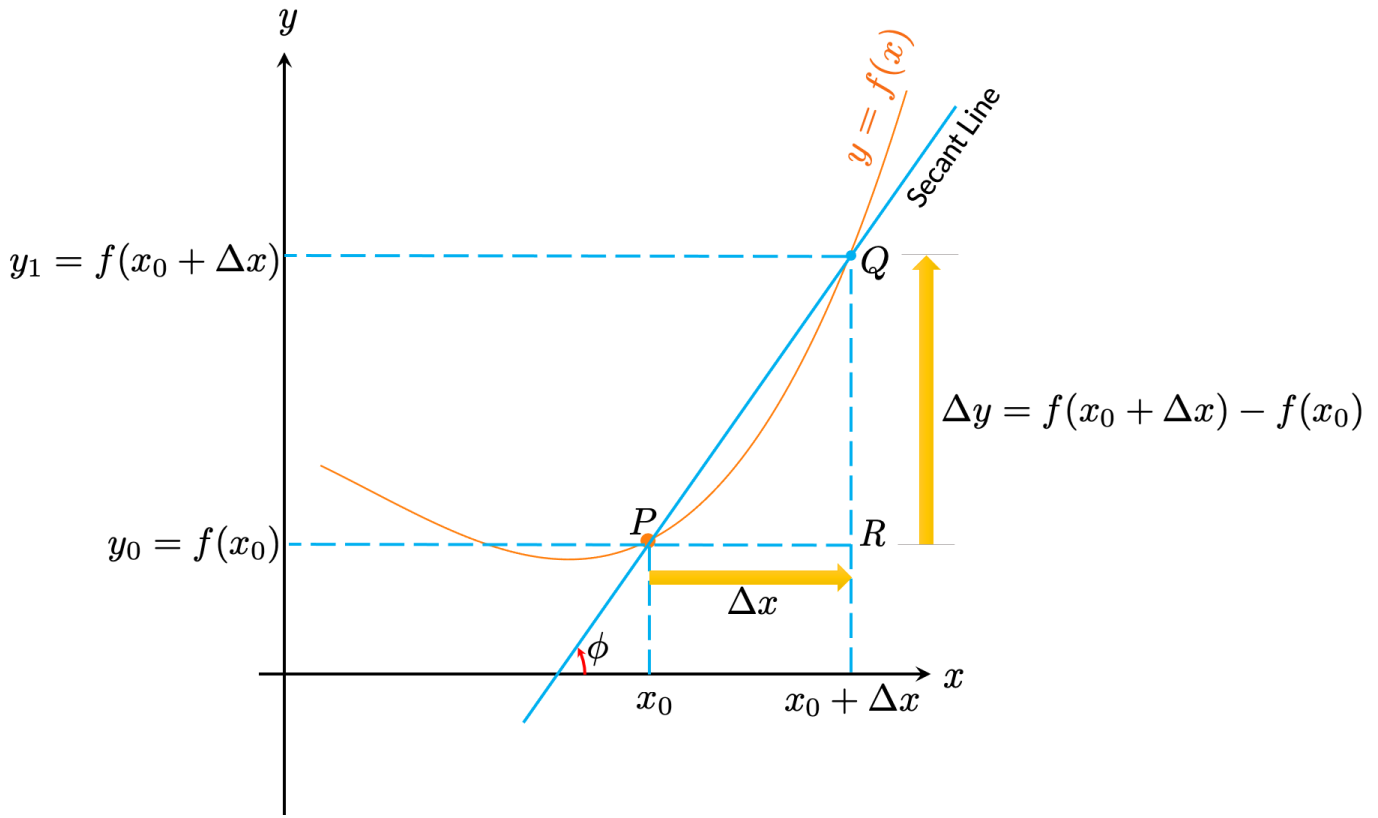
Functions – Differentiation – Geometric Interpretation:



- The orange curve represents the function $y = f(x)$, which is continuous and smooth in the interval around x_0.
- The derivative of the function at x_0 is related to the slope of the tangent line at that point, which can be approximated using the secant line.

Calculus

Functions – Differentiation – Geometric Interpretation:



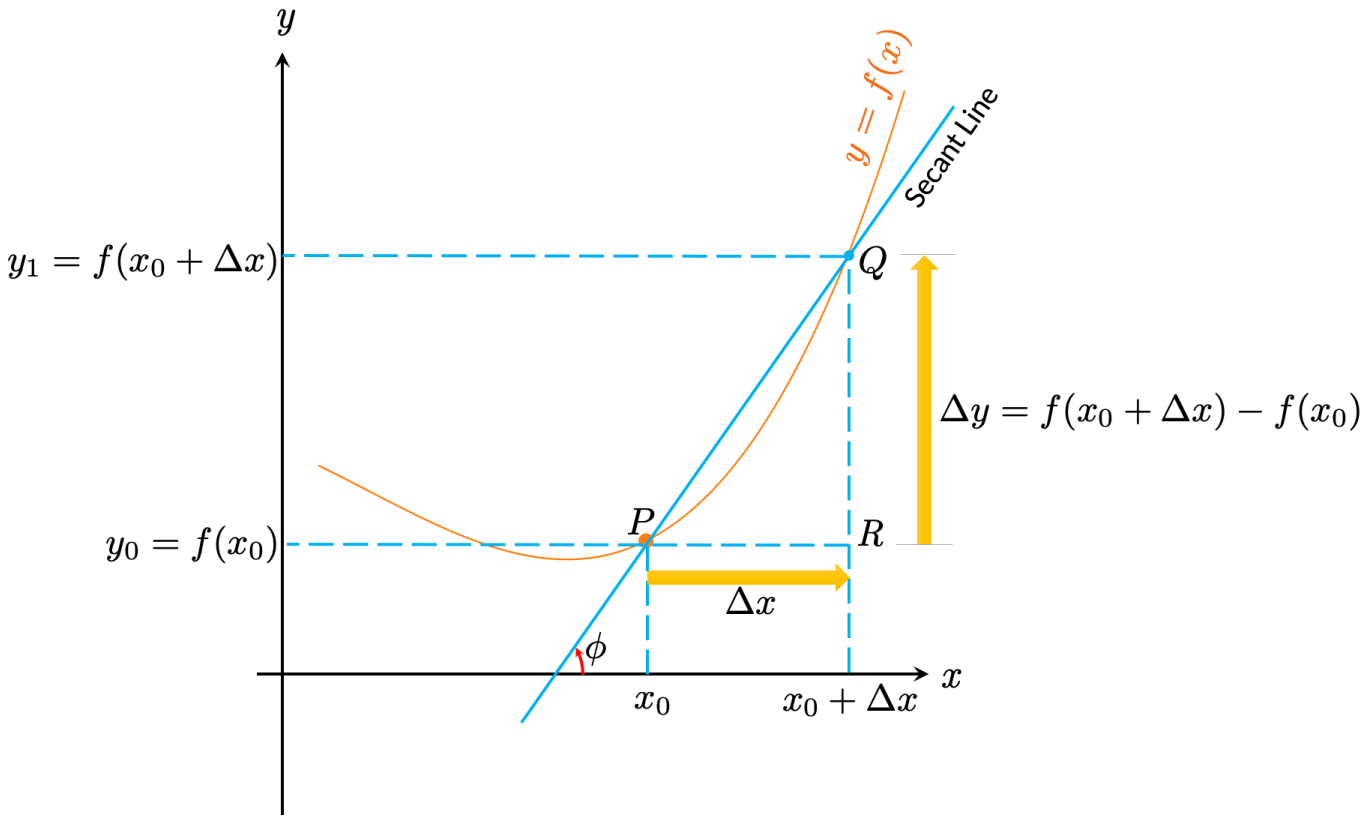
Point P: (x_0, y_0)

Point Q: $(x_0 + \Delta x, y_1)$

- The line passing through points P and Q is known as the **secant line**. This line represents an average rate of change of $f(x)$ over the interval $[x_0, x_0 + \Delta x]$.
- The slope of this secant line provides an approximation of the derivative at x_0.

Calculus

Functions – Differentiation – Geometric Interpretation:



The slope of the secant line:

$$\frac{\Delta y}{\Delta x} = \frac{f(x_0+\Delta x)-f(x_0)}{\Delta x}.$$

This ratio represents

the average rate of change

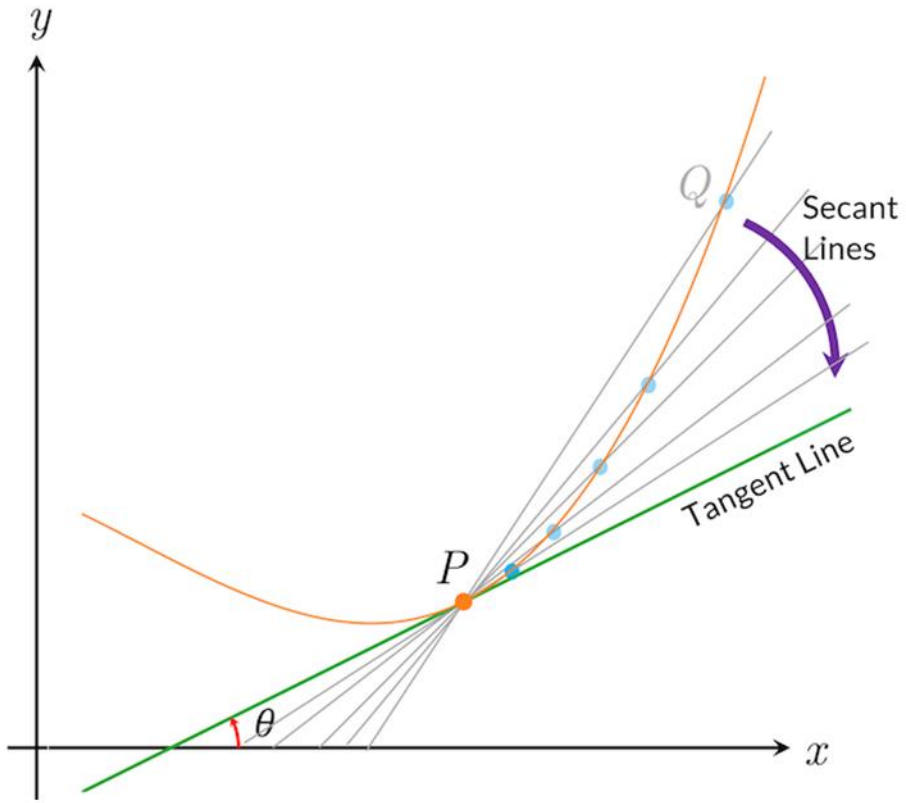
of the function $f(x)$
over the interval $[x_0, x_0 + \Delta x]$.

- **Horizontal Change (Δx)**: The horizontal distance between x_0 and $x_0 + \Delta x$.
- **Vertical Change (Δy)**: The vertical distance between y_0 and y_1, calculated as:

$$\Delta y = f(x_0 + \Delta x) - f(x_0).$$

Calculus

Functions – Differentiation – Geometric Interpretation:



The derivative at x_0, denoted $f'(x_0)$, is defined as the limit of the secant slope as Δx approaches zero:

$$f'(x_0) = \lim_{\Delta x \to 0} \frac{f(x_0+\Delta x)-f(x_0)}{\Delta x}.$$

– Geometrically, as Δx becomes very small, the secant line between P and Q becomes the **tangent line** at P. The slope of this tangent line is the derivative, which represents the **instantaneous rate of change** of $f(x)$ at x_0.

Calculus

Functions – Differentiation – Summary:

Definition of the Derivative: Derivative: The derivative of a function $f : \mathbb{R} \to \mathbb{R}$ at a point x is defined as:

$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}.$$

The derivative measures the slope of the tangent line to the graph of f at x.

Higher-Order Derivatives: Second Derivative: The second derivative of f is the derivative of the derivative, representing the rate of change of $f'(x)$:

$$f''(x) = \frac{d}{dx}\left(f'(x)\right).$$

Higher-order derivatives (third, fourth, etc.) can provide information on the function's concavity and curvature.

Calculus

Functions – Differentiation – Rules:

Sum Rule: The derivative of the sum of two functions is the sum of their derivatives. If $f(x)$ and $g(x)$ are differentiable, then:

$$\frac{d}{dx}\left(f(x)+g(x)\right)=f'(x)+g'(x).$$

Product Rule: The derivative of the product of two functions is given by:

$$\frac{d}{dx}\left(f(x)\cdot g(x)\right)=f'(x)\cdot g(x)+f(x)\cdot g'(x).$$

Quotient Rule: The derivative of the quotient of two functions is given by:

$$\frac{d}{dx}\left(\frac{f(x)}{g(x)}\right)=\frac{f'(x)\cdot g(x)-f(x)\cdot g'(x)}{[g(x)]^2}.$$

Calculus

Functions – Differentiation – Rules:

Chain Rule: The derivative of a composition of two functions is the product of the derivatives of the outer and inner functions. If $y = f(g(x))$, then:

$$\frac{dy}{dx} = f'(g(x)) \cdot g'(x).$$

Example Let $f(u) = u^2$ and $g(x) = \sin(x)$. Then $y = f(g(x)) = \sin^2(x)$, and:

$$\frac{dy}{dx} = 2\sin(x) \cdot \cos(x) = \sin(2x).$$

If $f(x_1, x_2)$ is a function of x_1 and x_2, where $x_1(s,t)$ and $x_2(s,t)$ are themselves functions of two variables s and t, the chain rule yields the partial derivatives:

$$\frac{\partial f}{\partial s} = \frac{\partial f}{\partial x_1}\frac{\partial x_1}{\partial s} + \frac{\partial f}{\partial x_2}\frac{\partial x_2}{\partial s}$$
$$\frac{\partial f}{\partial t} = \frac{\partial f}{\partial x_1}\frac{\partial x_1}{\partial t} + \frac{\partial f}{\partial x_2}\frac{\partial x_2}{\partial t}$$

Calculus

Functions – Differentiation – Gradient:

- For a function $f(\mathbf{x})$ that maps $\mathbf{x} \in \mathbf{R}^d$ to $\mathbf{R}$, we define a gradient (direec-tional derivative) with respect to $\mathbf{x}$ as

$$\nabla f(\mathbf{x}) = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_d}\right]^T \in \mathbf{R}^d$$

- Interpretation: Quantifies the rate of change along different directions.

Examples:

- $f(\mathbf{x}) = \mathbf{x}^T\mathbf{P}\mathbf{x}$

  $\nabla f(\mathbf{x}) = 2\mathbf{P}\mathbf{x}$

- $f(\mathbf{x}) = \mathbf{a}^T\mathbf{x} = \mathbf{x}^T\mathbf{a}$

  $\nabla f(\mathbf{x}) = \mathbf{a}$

- $f(\mathbf{x}) = \mathbf{x}^T\mathbf{x}$

  $\nabla f(\mathbf{x}) = 2\mathbf{x}$

Simple Example: Let $f(x, y) = x^2 + y^2$. Then:

$$\frac{\partial f}{\partial x} = 2x, \quad \frac{\partial f}{\partial y} = 2y.$$

Thus, the gradient of f is:

$$\nabla f(x, y) = \begin{bmatrix} 2x \\ 2y \end{bmatrix}.$$

Calculus

Functions – Differentiation – Jacobian

For a vector-valued function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$, the Jacobian matrix $J_{\mathbf{f}}(x)$ is defined as:

$$J_{\mathbf{f}}(x) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}.$$

Each row corresponds to the gradient of one component of $\mathbf{f}$.

Interpretation: The Jacobian describes the rate of change of a vector function with respect to each input variable.

Example: Let $\mathbf{f}(x, y) = \begin{bmatrix} x^2 \\ xy \end{bmatrix}$. The Jacobian matrix is:

$$J_{\mathbf{f}}(x, y) = \begin{bmatrix} \frac{\partial}{\partial x}(x^2) & \frac{\partial}{\partial y}(x^2) \\ \frac{\partial}{\partial x}(xy) & \frac{\partial}{\partial y}(xy) \end{bmatrix} = \begin{bmatrix} 2x & 0 \\ y & x \end{bmatrix}.$$

Example:

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

Jacobian?

Calculus

Functions – Differentiation – Hessian:

For a twice-differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, the Hessian matrix $H_f(x)$ is the square matrix of second-order partial derivatives:

$$H_f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}.$$

Interpretation:

Extension of a second-order derivative.

The Hessian provides information on the local curvature of f.

The Hessian is used to assess the convexity of a function and is fundamental in second-order optimization methods.

Calculus

Functions – Differentiation – Hessian:

Example: Let $f(x, y) = x^2 + y^2$. Then:

$$\frac{\partial^2 f}{\partial x^2} = 2, \quad \frac{\partial^2 f}{\partial y^2} = 2, \quad \frac{\partial^2 f}{\partial x \partial y} = 0.$$

The Hessian of f is:

$$H_f(x, y) = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}.$$

Calculus

Functions – Differentiation – Hessian:

Example: Hessian of Least-Squares function. Recall

$$f(\theta) = \frac{1}{2}\|X\theta - y\|^2 = \frac{1}{2}(X\theta - y)^T(X\theta - y)$$

$$f(\theta) = \frac{1}{2}(\theta^T X^T X\theta - 2y^T X\theta + y^T y).$$

$X \in \mathbb{R}^{n\times d}$ is a matrix,

$\theta \in \mathbb{R}^d$ is the vector of variables,

$y \in \mathbb{R}^n$ is the observed data vector.

The gradient of $f(\theta)$ with respect to θ is:

$$\nabla f(\theta) = X^T(X\theta - y).$$

The Hessian of $f(\theta)$, denoted $H_f(\theta)$, is the matrix of second partial derivatives of $f(\theta)$ with respect to θ. Since $\nabla f(\theta) = X^T X\theta - X^T y$, we see that the gradient is a linear function of θ, with the term $X^T X$ being constant with respect to θ. Thus, the Hessian is simply:

$$H_f(\theta) = X^T X.$$

Calculus

Functions – Differentiation – Taylor Series:

The **Taylor series** of a function $f(x)$ around a point $x = a$ provides an approximation of $f(x)$ using its derivatives at a. For a sufficiently smooth function f, the Taylor series expansion at $x = a$ is given by:

$$f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \cdots$$

or more compactly,

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n.$$

This expansion approximates $f(x)$ near $x = a$ by using polynomial terms.

Interpretation: The Taylor series represents $f(x)$ as an infinite sum of terms based on the function's derivatives. It is a powerful tool for approximating functions locally.

Calculus

Functions – Differentiation – Taylor Series Approximation:

Taylor Series Approximation $T_n(x)$: The Taylor series approximation of a function $f(x)$ at $x = a$ up to the n-th derivative term is given by:

$$T_n(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \cdots + \frac{f^{(n)}(a)}{n!}(x-a)^n.$$

This approximation $T_n(x)$ provides a polynomial of degree n that approximates $f(x)$ near $x = a$.

Interpretation: $T_n(x)$ is the Taylor polynomial of degree n, offering an n-term approximation of $f(x)$ around $x = a$.

Calculus

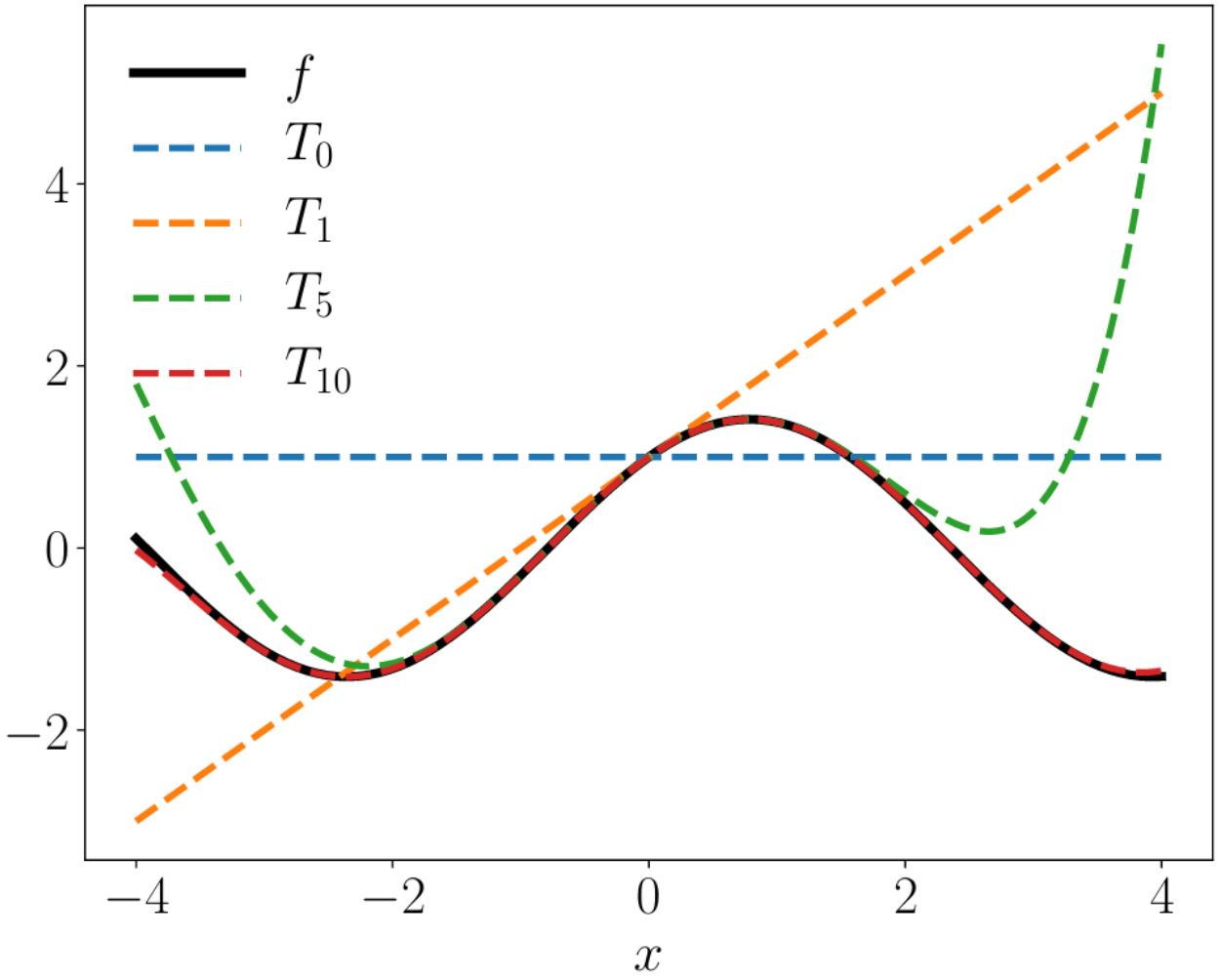
Functions – Differentiation – Taylor Series Approximation:

Figure 5.4 (From the MML Book)

The original function
$f(x) = \sin(x) + \cos(x)$ (black, solid)
is approximated by Taylor polynomials (dashed)
around $x_0 = 0$.

Higher-order Taylor polynomials approximate
the function f better and more globally.

T_{10} is already similar to f
in the interval $[-4, 4]$.



Calculus

Functions – Differentiation – Taylor Series Approximation:

For a multivariable function $f : \mathbb{R}^n \to \mathbb{R}$, the Taylor series expansion around a point $\mathbf{a}$ (up to the second order) is:

$$f(\mathbf{x}) \approx T_2(\mathbf{x}) = f(\mathbf{a}) + \nabla f(\mathbf{a})^T(\mathbf{x} - \mathbf{a}) + \frac{1}{2}(\mathbf{x} - \mathbf{a})^T H_f(\mathbf{a})(\mathbf{x} - \mathbf{a}).$$

where:

$\nabla f(\mathbf{a})$ is the gradient of f at $\mathbf{a}$,

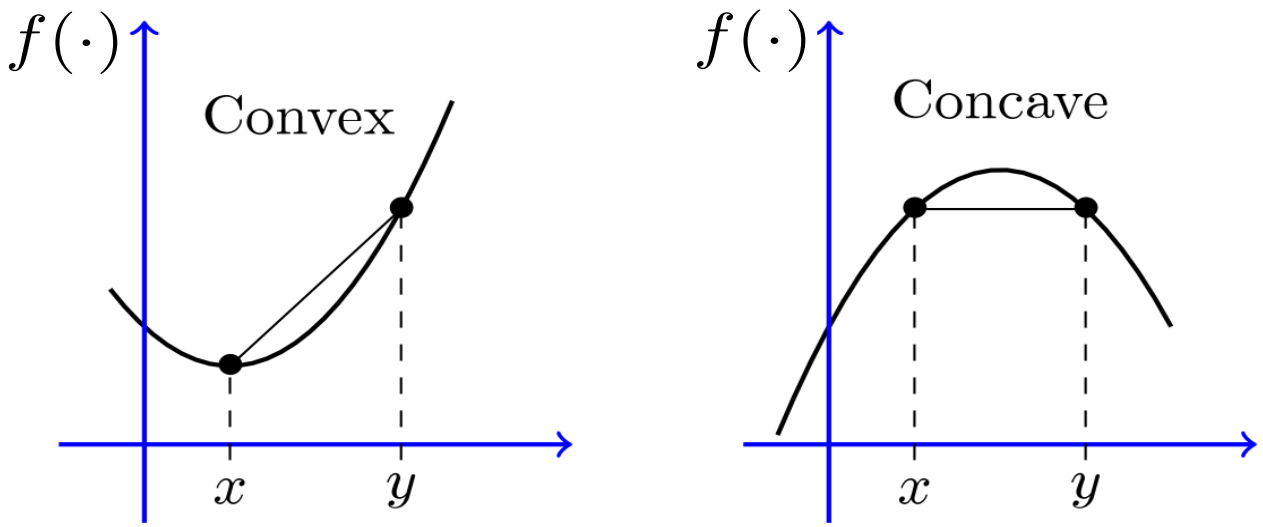
$H_f(\mathbf{a})$ is the Hessian matrix of f at $\mathbf{a}$.

Calculus

Convex Functions:

Convex Function: A function $f : \mathbb{R}^n \to \mathbb{R}$ is called convex if for all $x, y \in \text{dom}(f)$ and $\theta \in [0, 1]$, if the function satisfies **Jensen's Inequality** given by

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta) f(y).$$



This means that the line segment between any two points on the function lies above the graph of the function.

Calculus

Convex Functions – Another Interpretation:

Jensen's Inequality: For a convex function f and a random variable X,

$$f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)].$$

This inequality is fundamental in probability and statistics and shows that the expectation of a convex function is at least the function evaluated at the expectation.

Calculus

Convex Functions – Examples:

Linear Functions: Any linear function $f(x) = a^T x + b$ is convex.

Quadratic Functions: Functions of the form $f(x) = x^T Q x + b^T x + c$ are convex if $Q \succeq 0$ (i.e., Q is positive semi-definite).

Exponential Functions: $f(x) = e^x$ is convex for $x \in \mathbb{R}$.

Logarithmic Functions: $f(x) = -\log(x)$ is convex for $x > 0$.

Norms: The ℓ_p-norm $f(x) = \|x\|_p$ is convex for $p \geq 1$.

$\succeq$ is a generalized inequality, usually used to define a matrix that is positive semi-definite.

Calculus

Convex Functions – Operations That Preserve Convexity:

Convex functions retain their convexity under certain operations:

Nonnegative Weighted Sum: If $f_1, f_2, \ldots, f_k$ are convex, then $g(x) = \sum_{i=1}^{k} \alpha_i f_i(x)$ is convex for $\alpha_i \geq 0$.

Affine Composition: If f is convex and $Ax + b$ is an affine function, then $g(x) = f(Ax + b)$ is convex.

Pointwise Maximum: The pointwise maximum of convex functions, $g(x) = \max\{f_1(x), f_2(x), \ldots, f_k(x)\}$, is convex.

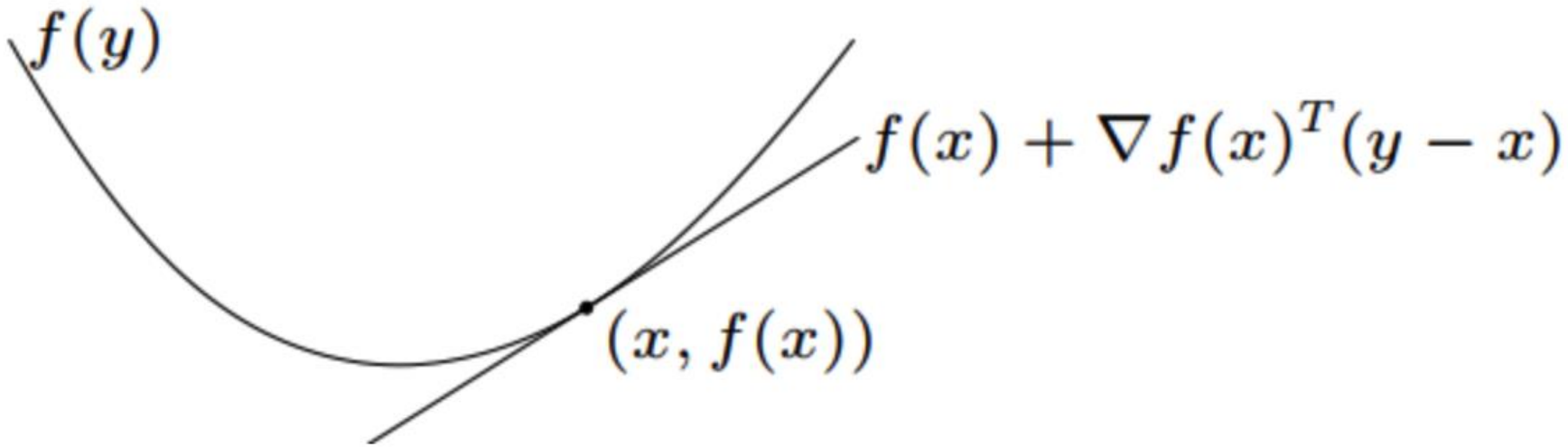
Calculus

Convex Functions:

First-Order Condition for Convexity: A differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if and only if:

$$f(y) \geq f(x) + \nabla f(x)^T (y - x), \quad \forall x, y \in \text{dom}(f).$$

This condition implies that the function lies above its tangent plane at any point.



Calculus

Convex Functions:

Second-Order Condition for Convexity:

A twice-differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if and only if its Hessian matrix $\nabla^2 f(x)$ is positive semi-definite for all $x \in \text{dom}(f)$:

$$\nabla^2 f(x) \succeq 0.$$

If $\nabla^2 f(x) \succ 0$ (positive definite), then f is strictly convex.

Calculus

Convex Functions:

Example: Hessian of Least-Squares function.

$$f(\theta) = \frac{1}{2}\|X\theta - y\|^2 = \frac{1}{2}(X\theta - y)^T(X\theta - y)$$

Recall

$$H_f(\theta) = X^T X.$$

Gradient Descent Algorithm

Optimization and Gradient Descent - Overview

- Optimization refers to finding **optimal** value of your unknown variables under some **constraints** on the variables.
  - Optimal value: usually, maximizing or minimizing the objective function.
  - Constraints: restricting the domain of our variable and are defined by imposing eqality or inequality constraints on the function of the variable.

- An optimization problem of finding a variable $\boldsymbol{\theta}$ is usually formulated as

$$\begin{aligned} &\text{minimize} && f_o(\boldsymbol{\theta}) \\ &\text{subject to} && f_i(\boldsymbol{\theta}) \leq 0, \quad i = 1, 2, \ldots, m \\ &&& h_j(\boldsymbol{\theta}) = 0, \quad , j = 1, 2, \ldots, p \end{aligned}$$

$f_o(\boldsymbol{\theta})$ - Objective function $f_i(\boldsymbol{\theta})$ - Inequality constraint functions $h_j(\boldsymbol{\theta})$ - equality constraint functions

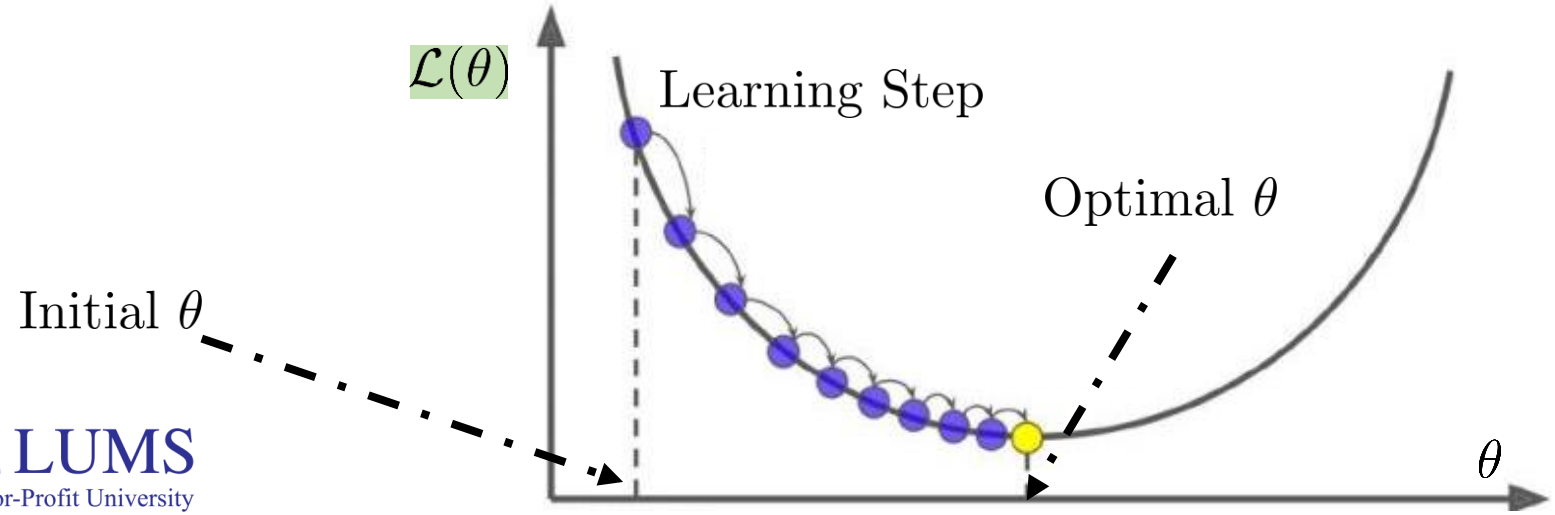
e.g., $\frac{1}{2}\|(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})\|_2^2$

- In ML, various algorithms (e.g., linear regression, neural network etc.) require us to solve an optimization problem.

Gradient Descent Algorithm

Optimization and Gradient Descent - Overview

- To solve the optimization problem, the gradient descent approach or algorithm is the most commonly used method.
- Gradient descent algorithm is best used when tha unknown variables cannot be determined analytically and need to be searched numerically.
- Gradient descent is an iterative algorithm in nature:
  - Initially, chose the coefficients to be something reasonabe (e.g., all zeros).
  - Iteratively update the coefficients in the direction of steepest descent until convergence.
  - Ensures that the new coefficients are better than the previous coefficients.



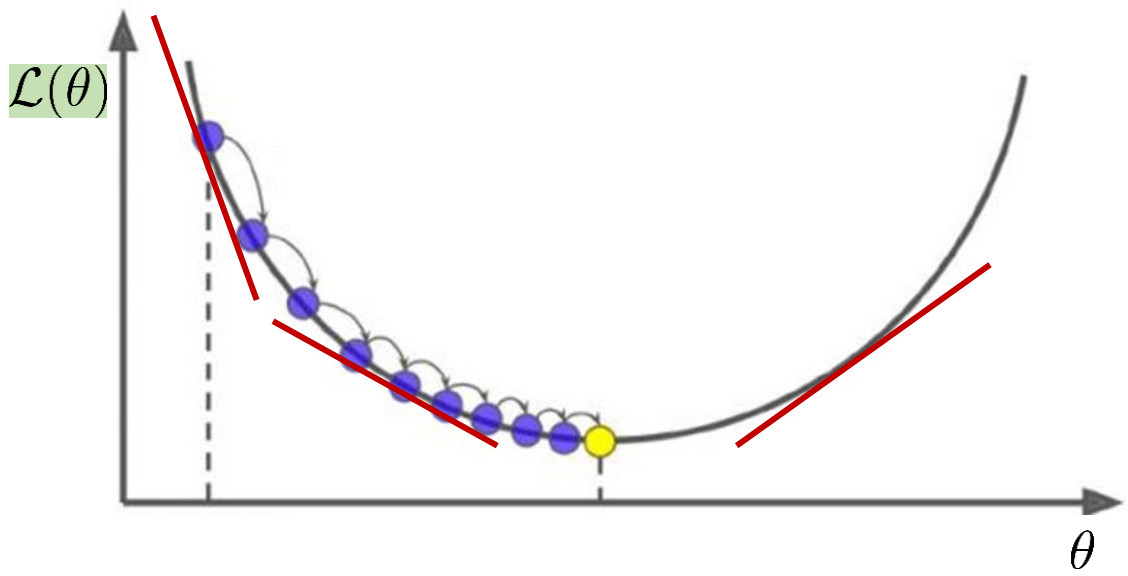
Gradient Descent Algorithm

Formulation:

- Loss function, denoted by $\mathcal{L}(\boldsymbol{\theta})$, required to be minimized. Assume $\boldsymbol{\theta} \in \mathbf{R}^d$.
- Interpretation of $\frac{\partial \mathcal{L}}{\partial \theta_i}$: Rate of change in the loss function with respect to θ_i
  - $\frac{\partial \mathcal{L}}{\partial \theta_i} > 0$: Increasing θ_i increases $\mathcal{L}$
  - $\frac{\partial \mathcal{L}}{\partial \theta_i} < 0$: Increasing θ_i decreases $\mathcal{L}$
- Noting this, the loss function is decreased with the following update:

$$\theta_i \leftarrow \theta_i - \alpha \frac{\partial \mathcal{L}}{\partial \theta_i}, \quad \alpha > 0$$

- This is the essence of gradient descent, that is, the step size in the direction of negative of the derivative.
- α is referred to as step size or learning rate.
  - Too small α: gradient descent can be slow.
  - Too large α: gradient descent can overshoot the minimum and it may fail to converge.



Gradient Descent Algorithm

Algorithm:

Overall:

- Start with some $\boldsymbol{\theta} \in \mathbf{R}^d$ and keep updating to reduce the loss function until we reach the minimum. Repeat until convergence

Pseudo-code:

- Initialize $\boldsymbol{\theta} \in \mathbf{R}^d$.
- Repeat until convergence:

$$\theta_i \leftarrow \theta_i - \alpha \frac{\partial \mathcal{L}}{\partial \theta_i}, \quad \text{for each} \quad i = 1, 2, \ldots, d$$

Equivalently,

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla \mathcal{L}(\boldsymbol{\theta})$$

Note: Simultaneous update.

Convergence and Step size:

- We stop updating $\boldsymbol{\theta}$ if $\nabla \mathcal{L}(\boldsymbol{\theta}) = 0$ or difference between the loss function in successive iterations is less than some threshold.
- We can have a constant step size α (typically 0.01, 0.05, 0.001) for each iteration or adjust it adpatively on each iteration.
- Algorithm converges for constant fixed rate as well due to the automatic smaller step size near the optimal solution.

Gradient Descent Algorithm

Linear Regression Case:

- We minimize mean-squared error (MSE) scaled by 1/2 factor:

$$\mathcal{L}(\theta_0, \boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^{n} \left(\theta_0 + \boldsymbol{\theta}^T \mathbf{x_i} - y_i\right)^2$$

- First we take a single feature regression; $\mathbf{x}$ is a scalar, $\mathbf{x_i} \equiv x_i$.

$$\mathcal{L}(\theta_0, \theta_1) = \frac{1}{2n} \sum_{i=1}^{n} \left(\theta_0 + \theta_1 x_i - y_i\right)^2$$

- We define partial derivatives as

$$\frac{\partial \mathcal{L}}{\partial \theta_0} = \frac{1}{n} \sum_{i=1}^{n} (\theta_0 + \theta_1 x_i - y_i) \qquad \frac{\partial \mathcal{L}}{\partial \theta_1} = \frac{1}{n} \sum_{i=1}^{n} (\theta_0 + \theta_1 x_i - y_i)\, x_i$$

Gradient Descent:

- Repeat until convergence:

$$\theta_0 \leftarrow \theta_0 - \alpha \frac{1}{n} \sum_{i=1}^{n} (\theta_0 + \theta_1 x_i - y_i) \qquad \theta_1 \leftarrow \theta_1 - \alpha \frac{1}{n} \sum_{i=1}^{n} (\theta_0 + \theta_1 x_i - y_i)\, x_i$$

Note:

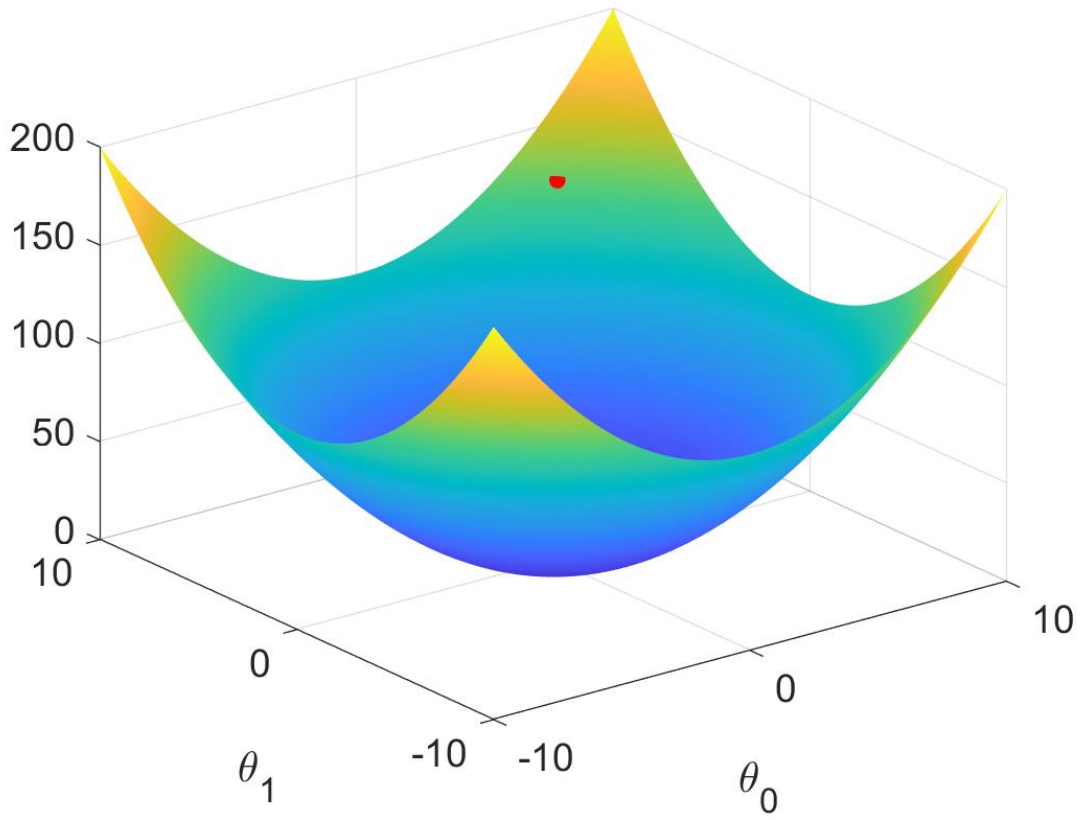
Simultaneous update.

Gradient Descent Algorithm

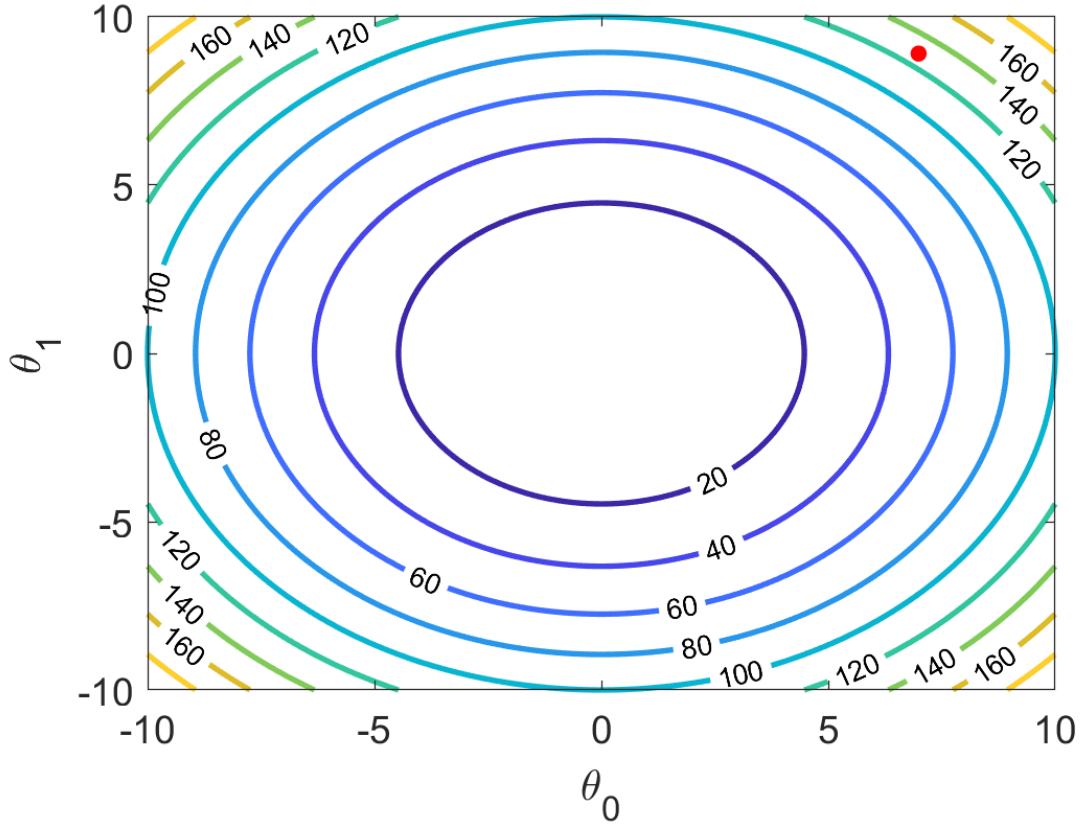
Linear Regression Case:

Visualization (To be shown in class):

$\mathcal{L}(\theta_0, \theta_1)$



Surface plot



Contour plot

$\alpha = 0.05,\ 0.2,\ 0.8,\ 1$

Gradient Descent Algorithm

Linear Regression Case:

- For a multiple feature regression; $\mathbf{x}$ is a vector, $\mathbf{x_i} \in \mathbf{R}^d$.

$$\mathcal{L}(\theta_0, \boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^{n} \left(\theta_0 + \boldsymbol{\theta}^T \mathbf{x_i} - y_i\right)^2$$

- We define partial derivatives as

$$\frac{\partial \mathcal{L}}{\partial \theta_0} = \frac{1}{n} \sum_{i=1}^{n} (\theta_0 + \boldsymbol{\theta}^T \mathbf{x_i} - y_i) \qquad \frac{\partial \mathcal{L}}{\partial \theta_j} = \frac{1}{n} \sum_{i=1}^{n} (\theta_0 + \boldsymbol{\theta}^T \mathbf{x_i} - y_i) \mathbf{x_i}^{(j)}$$

where $\mathbf{x_i}^{(j)}$ denotes the j-th component of $\mathbf{x_i}$.

Gradient Descent:

- Repeat until convergence:

$$\theta_0 \leftarrow \theta_0 - \alpha \frac{1}{n} \sum_{i=1}^{n} (\theta_0 + \boldsymbol{\theta}^T \mathbf{x_i} - y_i) \qquad \boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \frac{1}{n} \sum_{i=1}^{n} (\theta_0 + \boldsymbol{\theta}^T \mathbf{x_i} - y_i) \mathbf{x_i}$$

Note:

Simultaneous update.

Gradient Descent Algorithm

Notes:

- As we have taken all n points for updating at each step, we refer to the algorithm discussed here as **Batch Gradient Descent**.
- We also use the term 'epoch' to refer to one sweep of all the points in the data-set. So far, iteration is same as epoch as we have taken all the points at each step.
- We prefer to use gradient descent also for linear regression despite the fact that we can find the optimal solution analytically. **Why?**
- Gradient descent is easy to implement than the analytical solution.
- Gradient descent is computationally more efficient:
  - Closed-form (direct) solution: $\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$
  - Size of $\mathbf{X}^T\mathbf{X}$ is $d \times d$, matrix inversion computational complexity is $\mathcal{O}(d^3)$.
  - Computational complexity of each update of gradient descent is $\mathcal{O}(n\,d)$.
  - $\mathcal{O}(n\,d)$ is better than $\mathcal{O}(d^3)$ when $d \gg 1$.

Stochastic Gradient Descent:

- For large data-sets such that the computation of gradient for all points in the data-set takes too much time, we use stochastic gradient descent.

Gradient Descent Algorithm

Stochastic Gradient Descent (SGD) - Rationale:

- Generalize the formulation by defining a loss function using model $\hat{f}(\mathbf{x_i}, \mathbf{w})$

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2n}\sum_{i=1}^{n}\left(\hat{f}(\mathbf{x_i}, \mathbf{w}) - y_i\right)^2 = \frac{1}{n}\sum_{i=1}^{n} g(\mathbf{x_i}, \mathbf{w}, y_i)$$

where

$$g(\mathbf{x_i}, \mathbf{w}, y_i) = \frac{1}{2}\left(\hat{f}(\mathbf{x_i}, \mathbf{w}) - y_i\right)^2$$

Quantifies the prediction error for a single input.

- In Batch (or Full) Gradient Descent, we update in each iteration as

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla \mathcal{L}(\mathbf{w})$$

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \frac{1}{2n}\sum_{i=1}^{n} \nabla\left(\hat{f}(\mathbf{x_i}, \mathbf{w}) - y_i\right)^2$$

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \frac{1}{n}\sum_{i=1}^{n} \nabla g(\mathbf{x_i}, \mathbf{w}, y_i)$$

- We are computing gradient for all n points.
  - n can be very large in practice.
  - Computationally expensive.

Gradient Descent Algorithm

Stochastic Gradient Descent (SGD):

- Stochastic gradient descent: update using one data point at each iteration

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla g(\mathbf{x_i}, \mathbf{w}, y_i)$$

- Also referred to as incremental or online gradient descent.
- This update tries to approximate the update of batch gradient descent.
- Q: How do we choose i in each iteration?
  - Stochastic selection: uniformly choose the index in each iteration.
  - Cyclic selection: choose $i = 1, 2, \ldots, n, 1, 2, \ldots, n, 1, 2, \ldots$
- Stochastic (random) selection, mostly used in practice, implies that SGD is using an unbiased estimate of the true gradient at each iteration.

Pros:

- Computationally efficient: iteration cost is independent of n.
- True gradient approximation can help in escaping the local minimum.

Gradient Descent Algorithm

SGD for Linear Regression Case:

- Using cyclic selection, we have the following SGD:

- Initialize $\theta_0 \in \mathbf{R}$ and $\boldsymbol{\theta} \in \mathbf{R}^d$.
- Repeat until convergence:

for $i = 1, 2, \ldots n$

$$\theta_0 \leftarrow \theta_0 - \alpha(\theta_0 + \boldsymbol{\theta}^T \mathbf{x_i} - y_i)$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha\,(\theta_0 + \boldsymbol{\theta}^T \mathbf{x_i} - y_i)\mathbf{x_i}$$

} Iteration

} Epoch

end for

- Even with cyclic seleciton, we shuffle the order in which we are using the data points after each epoch. Otherwise, algorithm can get stuck with the sequence of gradient updates that may cancel each other and conseqeuntly hinder learning.
- For online learning when the data points are arriving in a stream, we need to carry out predictions before we have all the data-points. In such a case, we use SGD for learning.

Gradient Descent Algorithm

Mini-batch Stochastic Gradient Descent (SGD) :

Batch Gradient Descent

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \frac{1}{n} \sum_{i=1}^{n} \nabla g(\mathbf{x_i}, \mathbf{w}, y_i)$$

Stochastic Gradient Descent

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla g(\mathbf{x_i}, \mathbf{w}, y_i)$$

- Mini-batch Stochastic Gradient Descent: update using a subset of k data-points.
- From a set $\mathcal{D}$ of n points, we randomly select a subset, denoted by $\mathcal{S} \subseteq \mathcal{D}$ of $k \ll n$ points and use these k points to update the gradient as

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \frac{1}{n} \sum_{i=1}^{k} \nabla g(\mathbf{x_i}, \mathbf{w}, y_i), \quad (\mathbf{x_i}, y_i) \in \mathcal{S}$$

- In one epoch, we divide the data into mini-batches and run mini-batch SGD on each subset iteratively.