

# Mathematics for Machine Learning

## — Continuous Optimization: Gradient Descent and Constrained Optimization

Joseph Chuang-Chieh Lin

Department of Computer Science & Information Engineering,  
Tamkang University

Fall 2023

## Credits for the resource

- The slides are based on the textbooks:
  - *Marc Peter Deisenroth, A. Aldo Faisal, and Cheng Soon Ong: Mathematics for Machine Learning. Cambridge University Press. 2020.*
  - *Howard Anton, Chris Rorres, Anton Kaul: Elementary Linear Algebra. Wiley. 2019.*
- We could partially refer to the monograph:  
*Francesco Orabona: A Modern Introduction to Online Learning.*  
<https://arxiv.org/abs/1912.13213>

# Outline

- 1 Preface / Introduction
- 2 Optimization Using Gradient Descent
  - Gradient Descent with Momentum
  - Stochastic Gradient Descent
- 3 Constrained Optimization

# Motivation

- **Machine** learning algorithms are solving mathematical formulations which are expressed as **numerical optimization** methods.
- We focus on basic numerical methods for training machine learning models.
  - This boils down to *finding a “good” set of parameters*.
  - Goodness: determined by the objective function or the probabilistic model.
- Given an objective function, finding the best value of parameters is done using **optimization algorithms**.

# Remark

- We will discuss two branches of continuous optimization:
  - Unconstrained optimization.
  - Constrained optimization.
- Assume that the objective functions are **differentiable**.
- We focus on “minimization” objectives.
- We will make use of the “gradients”.

## Example

Consider the loss function  $\ell(x) = x^4 + 7x^3 + 5x^2 - 17x + 3$ .

The gradient:

$$\frac{d\ell(x)}{dx} = 4x^3 + 21x^2 + 10x - 17.$$

The second derivative:

$$\frac{d^2\ell(x)}{dx^2} = 12x^2 + 42x + 10.$$

Solving  $\frac{d\ell(x)}{dx} = 0$  we get  $x = -4.5, -1.4$ , or  $0.7$ .

## Example

Consider the loss function  $\ell(x) = x^4 + 7x^3 + 5x^2 - 17x + 3$ .

The gradient:

$$\frac{d\ell(x)}{dx} = 4x^3 + 21x^2 + 10x - 17.$$

The second derivative:

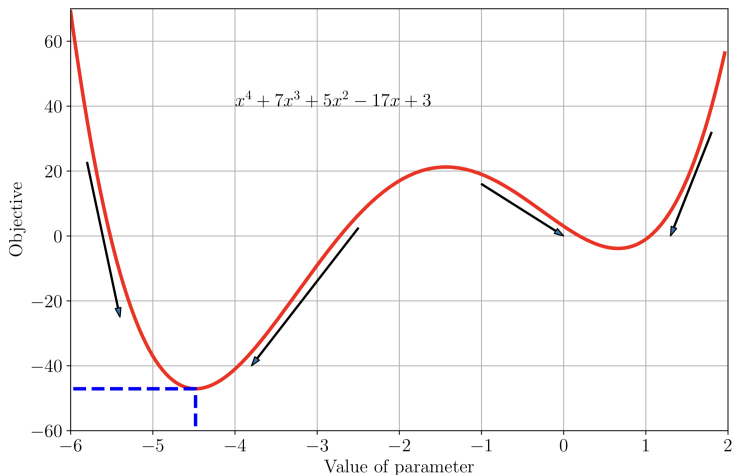
$$\frac{d^2\ell(x)}{dx^2} = 12x^2 + 42x + 10.$$

Solving  $\frac{d\ell(x)}{dx} = 0$  we get  $x = -4.5, -1.4$ , or  $0.7$ .

By checking whether  $\frac{d^2\ell(x)}{dx^2}$  is positive or negative at the stationary point(s), we know  $x = -1.4$  is a (local) maximum.

# Function Plot & Negative Gradients of Univariate $\ell(x)$

Start at some  $x_0$ , and then the negative gradient leads us to some (local) minimum.





# Heads Up

- For **convex functions**, there is no such a tricky dependency on the starting point.
  - All local minimums are *global* minimum.

# Heads Up

- For **convex functions**, there is no such a tricky dependency on the starting point.
  - All local minimums are *global* minimum.
- For “maximization” objectives, we shall follow the (positive) gradients.
  - Minimization objective  $\implies$  follows the negative gradient  $\implies$  “gradient descent”.
  - Maximization objective  $\implies$  follows the (positive) gradient  $\implies$  “gradient ascent”.

# Heads Up

- For **convex functions**, there is no such a tricky dependency on the starting point.
  - All local minimums are *global* minimum.
- For “maximization” objectives, we shall follow the (positive) gradients.
  - Minimization objective  $\implies$  follows the negative gradient  $\implies$  “gradient descent”.
  - Maximization objective  $\implies$  follows the (positive) gradient  $\implies$  “gradient ascent”.
- For optimization in higher dimensions, it is almost impossible to visualize the idea of gradients, descent directions and optimal values.

# Outline

- 1 Preface / Introduction
- 2 Optimization Using Gradient Descent
  - Gradient Descent with Momentum
  - Stochastic Gradient Descent
- 3 Constrained Optimization

# The Problem

Solving for the minimum of a real-valued function

$$\min_{\mathbf{x}} f(\mathbf{x}),$$

where  $f : \mathbb{R}^d \mapsto \mathbb{R}$  is the objective function which is assumed to be differentiable.

# Gradient Descent

- Gradient descent is a first-order optimization algorithm.

## Gradient Descent

- Starting at a particular location  $\mathbf{x}_0$ .

The algorithm runs iteratively by giving

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \gamma_i((\nabla f)(\mathbf{x}_i)).$$

where  $\gamma \geq 0$  is called the **step-size** (or **learning rate**).

**Goal:**  $f(\mathbf{x}_0) \geq f(\mathbf{x}_1) \geq \dots$  converges to a **local minimum**.

## Example (1/2)

### Example

Consider

$$f\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \frac{1}{2} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^\top \begin{bmatrix} 2 & 1 \\ 1 & 20 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} 5 \\ 3 \end{bmatrix}^\top \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.$$

Compute  $\nabla f\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) =$

## Example (1/2)

### Example

Consider

$$f\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \frac{1}{2} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^\top \begin{bmatrix} 2 & 1 \\ 1 & 20 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} 5 \\ 3 \end{bmatrix}^\top \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.$$

Compute  $\nabla f\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) =$



## Example (1/2)

### Example

Consider

$$f\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \frac{1}{2} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^\top \begin{bmatrix} 2 & 1 \\ 1 & 20 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} 5 \\ 3 \end{bmatrix}^\top \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.$$

Compute  $\nabla f\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^\top \begin{bmatrix} 2 & 1 \\ 1 & 20 \end{bmatrix} - \begin{bmatrix} 5 \\ 3 \end{bmatrix}^\top.$

## Example (1/2)

### Example

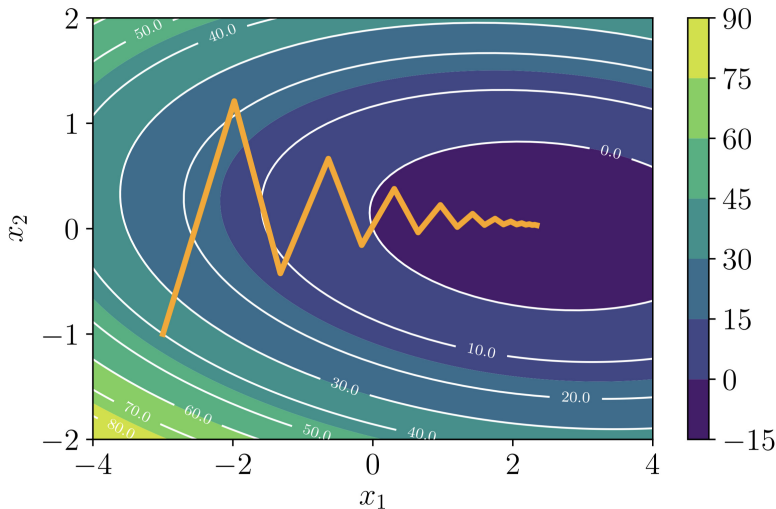
Consider

$$f\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \frac{1}{2} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^\top \begin{bmatrix} 2 & 1 \\ 1 & 20 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} 5 \\ 3 \end{bmatrix}^\top \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.$$

Compute  $\nabla f\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^\top \begin{bmatrix} 2 & 1 \\ 1 & 20 \end{bmatrix} - \begin{bmatrix} 5 \\ 3 \end{bmatrix}^\top.$

Running gradient descent and starting at  $\mathbf{x}_0 = [-3, -1]^\top$ , what's  $\mathbf{x}_1$ ?  
And what's  $\mathbf{x}_2$ ?

## Example (2/2)



## Remark

The gradient points in a direction orthogonal to the contour lines of the function.

## Remark

The gradient points in a direction orthogonal to the contour lines of the function.

- Let  $\gamma(t) = (x(t), y(t)) \in \mathbb{R}^2$  be a curve.
  - A **contour** of  $f$ :  $f(\gamma(t)) = C$  for some constant  $C \in \mathbb{R}$ .

## Remark

The gradient points in a direction orthogonal to the contour lines of the function.

- Let  $\gamma(t) = (x(t), y(t)) \in \mathbb{R}^2$  be a curve.
  - A **contour** of  $f$ :  $f(\gamma(t)) = C$  for some constant  $C \in \mathbb{R}$ .

$$\frac{df}{dt} = \mathbf{0}.$$

But

$$\frac{df}{dt} = \frac{df}{d\gamma} \frac{d\gamma}{dt} = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \end{bmatrix} \begin{bmatrix} \frac{\partial x}{\partial t} \\ \frac{\partial y}{\partial t} \end{bmatrix} = \langle \nabla_{\gamma} f, \nabla_t \gamma(t) \rangle$$

## On the Step-size

- Adaptive gradient descent: rescale the step-size  $\gamma$  at each iteration.

# On the Step-size

- Adaptive gradient descent: rescale the step-size  $\gamma$  at each iteration.
- Two simple heuristics:
  - When the function value  $\uparrow$  after a gradient step  $\implies$  undo the step and decrease the step-size.
  - When the function value  $\downarrow$  after a gradient step  $\implies$  try to increase the step-size.



## Another Example: Solving a Linear Equation System

### Example

Consider  $\mathbf{Ax} = \mathbf{b}$ .

## Another Example: Solving a Linear Equation System

### Example

Consider  $\mathbf{Ax} = \mathbf{b}$ .  $\implies \mathbf{Ax} - \mathbf{b} = \mathbf{0}$ .

- We want to find the solution *approximately* by minimizing the squared error

$$\ell(\mathbf{x}) = \|\mathbf{Ax} - \mathbf{b}\|^2 = (\mathbf{Ax} - \mathbf{b})^\top (\mathbf{Ax} - \mathbf{b})$$

where  $\|\cdot\|$  is the  $\ell_2$ -norm.

- $\nabla_{\mathbf{x}}\ell(\mathbf{x}) =$

## Another Example: Solving a Linear Equation System

### Example

Consider  $\mathbf{Ax} = \mathbf{b}$ .  $\implies \mathbf{Ax} - \mathbf{b} = \mathbf{0}$ .

- We want to find the solution *approximately* by minimizing the squared error

$$\ell(\mathbf{x}) = \|\mathbf{Ax} - \mathbf{b}\|^2 = (\mathbf{Ax} - \mathbf{b})^\top (\mathbf{Ax} - \mathbf{b})$$

where  $\|\cdot\|$  is the  $\ell_2$ -norm.

- $\nabla_{\mathbf{x}}\ell(\mathbf{x}) = 2(\mathbf{Ax} - \mathbf{b})^\top \mathbf{A}$ , run the gradient descent algorithm.

## Another Example: Solving a Linear Equation System

### Example

Consider  $\mathbf{Ax} = \mathbf{b}$ .  $\implies \mathbf{Ax} - \mathbf{b} = \mathbf{0}$ .

- We want to find the solution *approximately* by minimizing the squared error

$$\ell(\mathbf{x}) = \|\mathbf{Ax} - \mathbf{b}\|^2 = (\mathbf{Ax} - \mathbf{b})^\top (\mathbf{Ax} - \mathbf{b})$$

where  $\|\cdot\|$  is the  $\ell_2$ -norm.

- $\nabla_{\mathbf{x}}\ell(\mathbf{x}) = 2(\mathbf{Ax} - \mathbf{b})^\top \mathbf{A}$ , run the gradient descent algorithm.

Okay, this is just an example and somehow not a good idea because there is an analytic solution.

## Another Example: Solving a Linear Equation System

### Example

Consider  $\mathbf{Ax} = \mathbf{b}$ .  $\implies \mathbf{Ax} - \mathbf{b} = \mathbf{0}$ .

- We want to find the solution *approximately* by minimizing the squared error

$$\ell(\mathbf{x}) = \|\mathbf{Ax} - \mathbf{b}\|^2 = (\mathbf{Ax} - \mathbf{b})^\top (\mathbf{Ax} - \mathbf{b})$$

where  $\|\cdot\|$  is the  $\ell_2$ -norm.

- $\nabla_{\mathbf{x}}\ell(\mathbf{x}) = 2(\mathbf{Ax} - \mathbf{b})^\top \mathbf{A}$ , run the gradient descent algorithm.

Okay, this is just an example and somehow not a good idea because there is an analytic solution. (Solving  $\nabla_{\mathbf{x}}\ell(\mathbf{x}) = \mathbf{0}$ )

# Outline

- 1 Preface / Introduction
- 2 Optimization Using Gradient Descent
  - Gradient Descent with Momentum
  - Stochastic Gradient Descent
- 3 Constrained Optimization

# Gradient Descent with Momentum

- The convergence of gradient descent could be slow due to the curvature of the optimization surface.
- **Idea:** Give gradient descent some *memory*.
  - Introducing an additional term to remember what happened in the previous iteration.
- The steps (for  $\alpha \in [0, 1]$ ):

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \gamma_t((\nabla f)(\mathbf{x}_t))^{\top} + \alpha \Delta \mathbf{x}_t$$

$$\Delta \mathbf{x}_t = \mathbf{x}_t - \mathbf{x}_{t-1}$$

# Gradient Descent with Momentum

- The convergence of gradient descent could be slow due to the curvature of the optimization surface.
- **Idea:** Give gradient descent some *memory*.
  - Introducing an additional term to remember what happened in the previous iteration.
- The steps (for  $\alpha \in [0, 1]$ ):

$$\begin{aligned}\mathbf{x}_{t+1} &= \mathbf{x}_t - \gamma_t((\nabla f)(\mathbf{x}_t))^{\top} + \alpha \Delta \mathbf{x}_t \\ \Delta \mathbf{x}_t &= \mathbf{x}_t - \mathbf{x}_{t-1} = \alpha \Delta \mathbf{x}_{t-1} - \gamma_{t-1}((\nabla f)(\mathbf{x}_{t-1}))^{\top}\end{aligned}$$



# Outline

- 1 Preface / Introduction
- 2 Optimization Using Gradient Descent
  - Gradient Descent with Momentum
  - Stochastic Gradient Descent
- 3 Constrained Optimization

# Stochastic Gradient Descent (1/5)

Motivation:

- Computing the gradient can be very time consuming.
- Approximating the gradient is useful.
  - We aim at only knowing a noisy approximation to the gradient.

## Stochastic Gradient Descent (2/5)

The objective function:

$$L(\boldsymbol{\theta}) = \sum_{i=1}^N L_i(\boldsymbol{\theta}),$$

which is sum of losses  $L_i$  incurred by each sample  $i$ .  $\boldsymbol{\theta}$  is the vector of parameters of interest.

- **Goal:** Find  $\boldsymbol{\theta}$  that minimizes  $L$ .

Example: log-likelihoods

$$L(\boldsymbol{\theta}) = - \sum_{i=1}^N \log p(y_i \mid \mathbf{x}_i, \boldsymbol{\theta}),$$

for the training inputs  $\mathbf{x} \in \mathbb{R}^D$ , training targets  $y_i$ , and the parameters  $\boldsymbol{\theta}$  of the model.

# Stochastic Gradient Descent (3/5)

Updating  $\theta$ :

$$\theta_{t+1} \leftarrow \theta_t - \gamma_i (\nabla L(\theta_t))^\top$$

# Stochastic Gradient Descent (3/5)

Updating  $\theta$ :

$$\theta_{t+1} \leftarrow \theta_t - \gamma_i (\nabla L(\theta_t))^{\top} = \theta_t - \gamma_i \sum_{i=1}^N (\nabla L_i(\theta_t))^{\top}$$

according to suitable  $\gamma_i$ 's.

## Stochastic Gradient Descent (3/5)

Updating  $\theta$ : (Note that the differential operator is linear)

$$\theta_{t+1} \leftarrow \theta_t - \gamma_i (\nabla L(\theta_t))^\top = \theta_t - \gamma_i \sum_{i=1}^N (\nabla L_i(\theta_t))^\top$$

according to suitable  $\gamma_i$ 's.

### Issues

When training set is enormous or no simple formulas exist for evaluating the (sum of) gradients.

## Stochastic Gradient Descent (4/5)

**Idea:** Consider taking a sum of a **smaller** set of  $L_n$ .

- For  $i = 1, 2, \dots, N$ , randomly choose a subset of  $L_i$  for **mini-batch** gradient descent.

# Stochastic Gradient Descent (4/5)

**Idea:** Consider taking a sum of a **smaller** set of  $L_n$ .

- For  $i = 1, 2, \dots, N$ , randomly choose a subset of  $L_i$  for **mini-batch** gradient descent.
- Reasons:
  - We only require the gradient to be an unbiased estimate of the true gradient.



# Stochastic Gradient Descent (4/5)

**Idea:** Consider taking a sum of a **smaller** set of  $L_n$ .

- For  $i = 1, 2, \dots, N$ , randomly choose a subset of  $L_i$  for **mini-batch** gradient descent.
- Reasons:
  - We only require the gradient to be an unbiased estimate of the true gradient.
  - Even the term  $\sum_{i=1}^N (\nabla L_i(\theta_i))$  is just an empirical estimate of the expected value of the gradient.

## Stochastic Gradient Descent (4/5)

**Idea:** Consider taking a sum of a **smaller** set of  $L_n$ .

- For  $i = 1, 2, \dots, N$ , randomly choose a subset of  $L_i$  for **mini-batch** gradient descent.
- Reasons:
  - We only require the gradient to be an unbiased estimate of the true gradient.
  - Even the term  $\sum_{i=1}^N (\nabla L_i(\theta_i))$  is just an empirical estimate of the expected value of the gradient.
- Benefits for mini-batch:

## Stochastic Gradient Descent (4/5)

**Idea:** Consider taking a sum of a **smaller** set of  $L_n$ .

- For  $i = 1, 2, \dots, N$ , randomly choose a subset of  $L_i$  for **mini-batch** gradient descent.
- Reasons:
  - We only require the gradient to be an unbiased estimate of the true gradient.
  - Even the term  $\sum_{i=1}^N (\nabla L_i(\theta_i))$  is just an empirical estimate of the expected value of the gradient.
- Benefits for mini-batch:
  - Quick to estimate.
  - Noisy estimate allows us to get out of some bad local optima.

# Stochastic Gradient Descent (4/5)

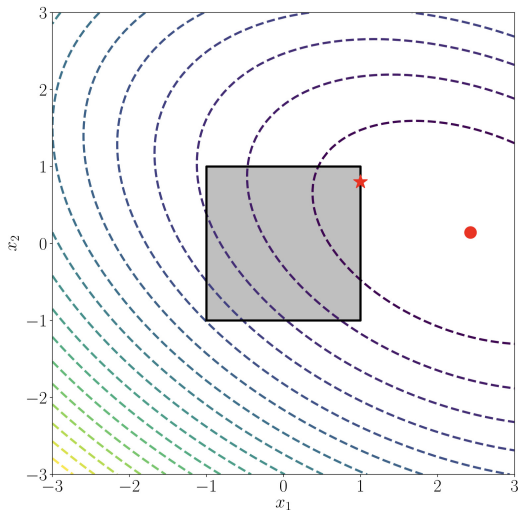
**Idea:** Consider taking a sum of a **smaller** set of  $L_n$ .

- For  $i = 1, 2, \dots, N$ , randomly choose a subset of  $L_i$  for **mini-batch** gradient descent.
- Reasons:
  - We only require the gradient to be an unbiased estimate of the true gradient.
  - Even the term  $\sum_{i=1}^N (\nabla L_i(\theta_i))$  is just an empirical estimate of the expected value of the gradient.
- Benefits for mini-batch:
  - Quick to estimate.
  - Noisy estimate allows us to get out of some bad local optima.
  - Good for generalization.

# Outline

- 1 Preface / Introduction
- 2 Optimization Using Gradient Descent
  - Gradient Descent with Momentum
  - Stochastic Gradient Descent
- 3 Constrained Optimization

# Constrained Optimization



# Constrained Optimization

The objective function:

$$\begin{array}{ll} \min_{\mathbf{x}} & f(\mathbf{x}) \\ \text{subject to} & g_i(\mathbf{x}) \leq 0, \text{ for all } i = 1, \dots, m. \end{array}$$

**Note:**  $f$  and  $g_i$  could be non-convex in general.

# Constrained Optimization

The objective function:

$$\begin{array}{ll} \min_{\mathbf{x}} & f(\mathbf{x}) \\ \text{subject to} & g_i(\mathbf{x}) \leq 0, \text{ for all } i = 1, \dots, m. \end{array}$$

**Note:**  $f$  and  $g_i$  could be non-convex in general.

## An Easy Unconstrained Objective

$$J(\mathbf{x}) = f(\mathbf{x}) + \sum_{i=1}^m \mathbf{1}(g_i(\mathbf{x})),$$

where  $\mathbf{1}(z)$  is an infinite step function  $\mathbf{1}(z) = \begin{cases} 0 & \text{if } z \leq 0 \\ \infty & \text{otherwise} \end{cases}$ .



# Constrained Optimization

The objective function:

$$\begin{array}{ll} \min_{\mathbf{x}} & f(\mathbf{x}) \\ \text{subject to} & g_i(\mathbf{x}) \leq 0, \text{ for all } i = 1, \dots, m. \end{array}$$

**Note:**  $f$  and  $g_i$  could be non-convex in general.

## An Easy Unconstrained Objective

$$J(\mathbf{x}) = f(\mathbf{x}) + \sum_{i=1}^m \mathbf{1}(g_i(\mathbf{x})),$$

where  $\mathbf{1}(z)$  is an infinite step function  $\mathbf{1}(z) = \begin{cases} 0 & \text{if } z \leq 0 \\ \infty & \text{otherwise} \end{cases}$ .

The infinite step function is difficult to optimize...

## A Workaround Approach: Lagrange Multipliers

For  $\lambda_i \geq 0$ , define

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \sum_{i=1}^m \lambda_i g_i(\mathbf{x})$$

# A Workaround Approach: Lagrange Multipliers

For  $\lambda_i \geq 0$ , define

$$\begin{aligned}\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) &= f(\mathbf{x}) + \sum_{i=1}^m \lambda_i g_i(\mathbf{x}) \\ &= f(\mathbf{x}) + \boldsymbol{\lambda}^\top \mathbf{g}(\mathbf{x}).\end{aligned}$$

## A Workaround Approach: Lagrange Multipliers

For  $\lambda_i \geq 0$ , define

$$\begin{aligned}\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) &= f(\mathbf{x}) + \sum_{i=1}^m \lambda_i g_i(\mathbf{x}) \\ &= f(\mathbf{x}) + \boldsymbol{\lambda}^\top \mathbf{g}(\mathbf{x}).\end{aligned}$$

- We concatenate all constraints  $g_i(\mathbf{x})$  into a vector  $\mathbf{g}(\mathbf{x})$ .

## A Workaround Approach: Lagrange Multipliers

For  $\lambda_i \geq 0$ , define

$$\begin{aligned}\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) &= f(\mathbf{x}) + \sum_{i=1}^m \lambda_i g_i(\mathbf{x}) \\ &= f(\mathbf{x}) + \boldsymbol{\lambda}^\top \mathbf{g}(\mathbf{x}).\end{aligned}$$

- We concatenate all constraints  $g_i(\mathbf{x})$  into a vector  $\mathbf{g}(\mathbf{x})$ .
  - This technique is widely used in building machine learning models!

## A Workaround Approach: Lagrange Multipliers

For  $\lambda_i \geq 0$ , define

$$\begin{aligned}\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) &= f(\mathbf{x}) + \sum_{i=1}^m \lambda_i g_i(\mathbf{x}) \\ &= f(\mathbf{x}) + \boldsymbol{\lambda}^\top \mathbf{g}(\mathbf{x}).\end{aligned}$$

- We concatenate all constraints  $g_i(\mathbf{x})$  into a vector  $\mathbf{g}(\mathbf{x})$ .
  - This technique is widely used in building machine learning models!

- $\mathbf{x}$ : primal variables.
- $\boldsymbol{\lambda}$ : dual variables.

# Primal & Dual Problems

## The primal problem

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & g_i(\mathbf{x}) \leq 0, \text{ for } i \in [m]. \end{aligned}$$

## The dual problem

$$\begin{aligned} \max_{\boldsymbol{\lambda} \in \mathbb{R}^m} \quad & \mathcal{D}(\boldsymbol{\lambda}) \\ \text{subject to} \quad & \boldsymbol{\lambda} \geq \mathbf{0}. \end{aligned}$$

$$\mathcal{D}(\boldsymbol{\lambda}) := \min_{\mathbf{x} \in \mathbb{R}^d} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}).$$

# Primal & Dual Problems

## The primal problem

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & g_i(\mathbf{x}) \leq 0, \text{ for } i \in [m]. \end{aligned}$$

## The dual problem

$$\begin{aligned} \max_{\boldsymbol{\lambda} \in \mathbb{R}^m} \quad & \mathcal{D}(\boldsymbol{\lambda}) \\ \text{subject to} \quad & \boldsymbol{\lambda} \geq \mathbf{0}. \end{aligned}$$

$$\mathcal{D}(\boldsymbol{\lambda}) := \min_{\mathbf{x} \in \mathbb{R}^d} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}).$$

- This is **unconstrained** optimization problem for a **given value of  $\boldsymbol{\lambda}$** .  
( $\boldsymbol{\lambda} \geq \mathbf{0} \Leftrightarrow \lambda_i \geq 0$  for each  $i$ )



# Primal & Dual Problems

## The primal problem

$$\begin{array}{ll} \min_{\mathbf{x}} & f(\mathbf{x}) \\ \text{subject to} & g_i(\mathbf{x}) \leq 0, \text{ for } i \in [m]. \end{array}$$

## The dual problem

$$\begin{array}{ll} \max_{\boldsymbol{\lambda} \in \mathbb{R}^m} & \mathcal{D}(\boldsymbol{\lambda}) \\ \text{subject to} & \boldsymbol{\lambda} \succeq \mathbf{0}. \end{array}$$

$$\mathcal{D}(\boldsymbol{\lambda}) := \min_{\mathbf{x} \in \mathbb{R}^d} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}).$$

- This is **unconstrained** optimization problem for a **given value of  $\boldsymbol{\lambda}$** .  
( $\boldsymbol{\lambda} \succeq \mathbf{0} \Leftrightarrow \lambda_i \geq 0$  for each  $i \Leftrightarrow \boldsymbol{\lambda} \succeq \mathbf{0}$ )

# Minimax Inequality

## Minimax Inequality

For any function  $\varphi$  with two arguments  $\mathbf{x}, \mathbf{y}$ ,

$$\max_{\mathbf{y}} \min_{\mathbf{x}} \varphi(\mathbf{x}, \mathbf{y}) \leq \min_{\mathbf{x}} \max_{\mathbf{y}} \varphi(\mathbf{x}, \mathbf{y}).$$

# Minimax Inequality

## Minimax Inequality

For any function  $\varphi$  with two arguments  $\mathbf{x}, \mathbf{y}$ ,

$$\max_{\mathbf{y}} \min_{\mathbf{x}} \varphi(\mathbf{x}, \mathbf{y}) \leq \min_{\mathbf{x}} \max_{\mathbf{y}} \varphi(\mathbf{x}, \mathbf{y}).$$

- Consider the inequality

$$\text{For all } \mathbf{x}_0, \mathbf{y}_0, \quad \min_{\mathbf{x}} \varphi(\mathbf{x}, \mathbf{y}_0) \leq \max_{\mathbf{y}} \varphi(\mathbf{x}_0, \mathbf{y}).$$

- This implies that

$$\min_{\mathbf{x} \in \mathbb{R}^d} \max_{\lambda \geq 0} \mathcal{L}(\mathbf{x}, \lambda) \geq \max_{\lambda \geq 0} \min_{\mathbf{x} \in \mathbb{R}^d} \mathcal{L}(\mathbf{x}, \lambda).$$

Compare  $J(\mathbf{x})$  with  $\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda})$ 

$$J(\mathbf{x}) = f(\mathbf{x}) + \sum_{i=1}^m \mathbf{1}(g_i(\mathbf{x})), \text{ where } \mathbf{1}(z) = \begin{cases} 0 & \text{if } z \leq 0 \\ \infty & \text{otherwise} \end{cases}$$

v.s.

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \boldsymbol{\lambda}^\top \mathbf{g}(\mathbf{x}).$$

## Compare $J(\mathbf{x})$ with $\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda})$

$$J(\mathbf{x}) = f(\mathbf{x}) + \sum_{i=1}^m \mathbf{1}(g_i(\mathbf{x})), \text{ where } \mathbf{1}(z) = \begin{cases} 0 & \text{if } z \leq 0 \\ \infty & \text{otherwise} \end{cases}$$

v.s.

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \boldsymbol{\lambda}^\top \mathbf{g}(\mathbf{x}).$$

- $\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda})$  is a lower bound of  $J(\mathbf{x})$ .  
 $\therefore J(\mathbf{x}) = \max_{\boldsymbol{\lambda} \geq \mathbf{0}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda})$ .

## Compare $J(\mathbf{x})$ with $\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda})$

$$J(\mathbf{x}) = f(\mathbf{x}) + \sum_{i=1}^m \mathbf{1}(g_i(\mathbf{x})), \text{ where } \mathbf{1}(z) = \begin{cases} 0 & \text{if } z \leq 0 \\ \infty & \text{otherwise} \end{cases}$$

v.s.

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \boldsymbol{\lambda}^\top \mathbf{g}(\mathbf{x}).$$

- $\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda})$  is a lower bound of  $J(\mathbf{x})$ .

$$\therefore J(\mathbf{x}) = \max_{\boldsymbol{\lambda} \geq \mathbf{0}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}).$$

- Recall the original problem:

$$\min_{\mathbf{x} \in \mathbb{R}^d} \max_{\boldsymbol{\lambda} \geq \mathbf{0}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}).$$

# Compare $J(\mathbf{x})$ with $\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda})$

$$J(\mathbf{x}) = f(\mathbf{x}) + \sum_{i=1}^m \mathbf{1}(g_i(\mathbf{x})), \text{ where } \mathbf{1}(z) = \begin{cases} 0 & \text{if } z \leq 0 \\ \infty & \text{otherwise} \end{cases}$$

v.s.

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \boldsymbol{\lambda}^\top \mathbf{g}(\mathbf{x}).$$

- $\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda})$  is a lower bound of  $J(\mathbf{x})$ .

$$\therefore J(\mathbf{x}) = \max_{\boldsymbol{\lambda} \geq \mathbf{0}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}).$$

- Recall the original problem:

$$\min_{\mathbf{x} \in \mathbb{R}^d} \max_{\boldsymbol{\lambda} \geq \mathbf{0}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}).$$

Hence,

$$\min_{\mathbf{x} \in \mathbb{R}^d} \max_{\boldsymbol{\lambda} \geq \mathbf{0}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) \geq \max_{\boldsymbol{\lambda} \geq \mathbf{0}} \min_{\mathbf{x} \in \mathbb{R}^d} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}).$$

# Computation Issues

$$\min_{\mathbf{x} \in \mathbb{R}^d} \max_{\boldsymbol{\lambda} \geq \mathbf{0}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) \geq \max_{\boldsymbol{\lambda} \geq \mathbf{0}} \min_{\mathbf{x} \in \mathbb{R}^d} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}).$$

- $\min_{\mathbf{x} \in \mathbb{R}^d} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda})$  is unconstrained, hence it is somehow easy to solve.



# Computation Issues

$$\min_{\mathbf{x} \in \mathbb{R}^d} \max_{\boldsymbol{\lambda} \geq \mathbf{0}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) \geq \max_{\boldsymbol{\lambda} \geq \mathbf{0}} \min_{\mathbf{x} \in \mathbb{R}^d} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}).$$

- $\min_{\mathbf{x} \in \mathbb{R}^d} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda})$  is unconstrained, hence it is somehow easy to solve.
- $\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda})$  is affine w.r.t.  $\boldsymbol{\lambda}$ .

# Computation Issues

$$\min_{\mathbf{x} \in \mathbb{R}^d} \max_{\boldsymbol{\lambda} \geq \mathbf{0}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) \geq \max_{\boldsymbol{\lambda} \geq \mathbf{0}} \min_{\mathbf{x} \in \mathbb{R}^d} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}).$$

- $\min_{\mathbf{x} \in \mathbb{R}^d} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda})$  is unconstrained, hence it is somehow easy to solve.
- $\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda})$  is affine w.r.t.  $\boldsymbol{\lambda}$ .
- $\mathcal{D}(\boldsymbol{\lambda}) = \min_{\mathbf{x} \in \mathbb{R}^d} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda})$  is pointwise minimum of affine functions of  $\boldsymbol{\lambda}$ .

# Computation Issues

$$\min_{\mathbf{x} \in \mathbb{R}^d} \max_{\boldsymbol{\lambda} \geq \mathbf{0}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) \geq \max_{\boldsymbol{\lambda} \geq \mathbf{0}} \min_{\mathbf{x} \in \mathbb{R}^d} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}).$$

- $\min_{\mathbf{x} \in \mathbb{R}^d} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda})$  is unconstrained, hence it is somehow easy to solve.
- $\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda})$  is affine w.r.t.  $\boldsymbol{\lambda}$ .
- $\mathcal{D}(\boldsymbol{\lambda}) = \min_{\mathbf{x} \in \mathbb{R}^d} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda})$  is pointwise minimum of affine functions of  $\boldsymbol{\lambda}$ .
  - The outer problem of maximization over  $\boldsymbol{\lambda}$  can be efficiently computed.  
( $\mathcal{D}(\boldsymbol{\lambda})$  is concave so finding the maximum is easy).

# Remark

- $h_j(\mathbf{x}) \geq 0 \iff -h_j(\mathbf{x}) \leq 0.$

## Remark

- $h_j(\mathbf{x}) \geq 0 \iff -h_j(\mathbf{x}) \leq 0.$
- What's about “equality” constraints?

## Remark

- $h_j(\mathbf{x}) \geq 0 \iff -h_j(\mathbf{x}) \leq 0$ .
- What's about “equality” constraints?

$$h_j(\mathbf{x}_j) = 0 \implies \begin{cases} h_j(\mathbf{x}) \leq 0 & \text{and} \\ -h_j(\mathbf{x}) \leq 0. \end{cases}$$

# Discussions