

# CS5489

## Lecture 3.1: Gradient Descent and Its Variants

Kede Ma

City University of Hong Kong (Dongguan)



Slide template by courtesy of Benjamin M. Marlin  
Part of slides are borrowed from Stephen P. Boyd

## Generative vs Discriminative Classifiers

- The Bayes optimal classifier, NB and LDA are said to be *generative* classifiers because they explicitly model the joint distribution  $p(\mathbf{x}, y)$  of the feature vectors  $\mathbf{x}$  and the labels  $y$
  - To build a probabilistic classifier, all we really need to model is  $p(y|\mathbf{x})$
  - Classifiers based on directly estimating  $p(y|\mathbf{x})$  are called *discriminative* classifiers because they don't care about the class conditional distribution  $p(\mathbf{x}|y)$

# Logistic Regression

- Logistic regression is a probabilistic discriminative classifier
- It directly models the decision boundary using a linear function:

$$p(y = +1|\mathbf{x}) = \frac{1}{1 + \exp(-(\mathbf{w}^T \mathbf{x} + b))}$$

$$\log \frac{p(y = +1|\mathbf{x})}{p(y = -1|\mathbf{x})} = \mathbf{w}^T \mathbf{x} + b$$

- The classification function is:

$$f_{\text{LR}}(\mathbf{x}) = \arg \max_{c \in \{1, \dots, C\}} p(y = c|\mathbf{x})$$

- Learning the model parameters cannot be performed analytically and has to resort to numerical optimization methods

# Optimization

- Much of machine learning can be written as an optimization problem:

$$\min_{\boldsymbol{\theta}} \ell(\mathcal{D}; \boldsymbol{\theta}) = \min_{\boldsymbol{\theta}} \frac{1}{M} \sum_{i=1}^M \ell(\mathbf{x}^{(i)}, y^{(i)}; \boldsymbol{\theta}) + r(\boldsymbol{\theta})$$

- $\boldsymbol{\theta}$  is a parameter vector to be optimized
- $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)}), i = 1, \dots, M\}$  is a finite training set
- $r(\cdot)$  is a regularizer to prevent overfitting to  $\mathcal{D}$
- In logistic regression ( $\boldsymbol{\theta} = \{\mathbf{w}, b\}$ )
  - $\ell(\mathbf{x}^{(i)}, y^{(i)}; \boldsymbol{\theta}) = \log(1 + \exp(-y^{(i)}(\mathbf{w}^T \mathbf{x}^{(i)} + b)))$
  - $r(\boldsymbol{\theta}) = \frac{1}{C} \mathbf{w}^T \mathbf{w}$

# Types of Optimization

- Convex optimization
  - The easy case
    - Includes logistic regression, linear regression, SVM
- Nonconvex optimization
  - Hard in general
    - Includes deep learning

# Convex Set

- **Line segment** between  $\mathbf{x}^{(1)}$  and  $\mathbf{x}^{(2)}$ : all points

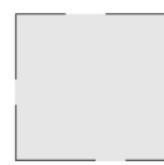
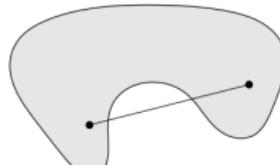
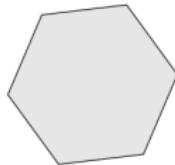
$$\mathbf{x} = \alpha\mathbf{x}^{(1)} + (1 - \alpha)\mathbf{x}^{(2)}$$

with  $0 \leq \alpha \leq 1$

- **Convex set**: contains line segment between any two points in the set

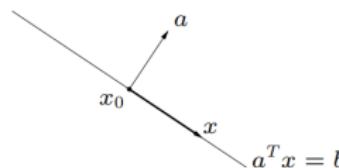
$$\mathbf{x}^{(1)}, \mathbf{x}^{(2)} \in \mathcal{X}, 0 \leq \alpha \leq 1 \implies \alpha\mathbf{x}^{(1)} + (1 - \alpha)\mathbf{x}^{(2)} \in \mathcal{X}$$

- Examples (one convex, two nonconvex sets)

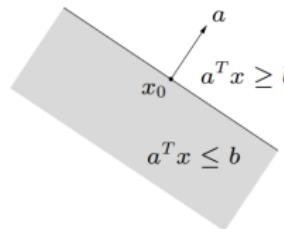


# Hyperplanes and Halfspaces

- **Hyperplane:** set of the form  $\{\mathbf{x} | \mathbf{a}^T \mathbf{x} = b\}$  ( $\mathbf{a} \neq 0$ )
  - $\mathbf{a}$  is the normal vector



- **Halfspace:** set of the form  $\{\mathbf{x} | \mathbf{a}^T \mathbf{x} \leq b\}$  ( $\mathbf{a} \neq 0$ )



- Hyperplanes and halfspaces are convex

# Operations that Preserve Convexity

- Practical methods for establishing convexity of a set  $\mathcal{X}$

- 1 Apply definition

$$\mathbf{x}^{(1)}, \mathbf{x}^{(2)} \in \mathcal{X}, 0 \leq \alpha \leq 1 \implies \alpha \mathbf{x}^{(1)} + (1 - \alpha) \mathbf{x}^{(2)} \in \mathcal{X}$$

- 2 Show that  $\mathcal{X}$  is obtained from simple convex set (hyperplanes, halfspaces, etc.) by operations that preserve convexity

- Intersection
- Affine functions
- ...

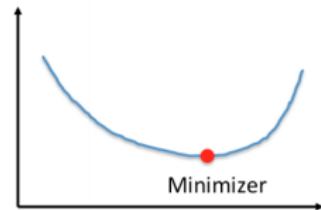
# Convex Function

- $f : \mathbb{R}^N \mapsto \mathbb{R}$  is convex if  $\text{dom}(f)$  is a convex set and

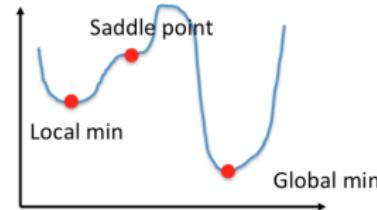
$$f(\alpha \mathbf{x}^{(1)} + (1 - \alpha) \mathbf{x}^{(2)}) \leq \alpha f(\mathbf{x}^{(1)}) + (1 - \alpha) f(\mathbf{x}^{(2)})$$

for all  $\mathbf{x}^{(1)}, \mathbf{x}^{(2)} \in \text{dom}(f)$ ,  $0 \leq \alpha \leq 1$

Convex



Non-Convex



- $f$  is concave if  $-f$  is convex

# Examples on $\mathbb{R}$

## ■ Convex:

- Affine:  $ax + b$  on  $\mathbb{R}$ , for any  $a, b \in \mathbb{R}$
- Exponential:  $e^{ax}$ , for any  $a \in \mathbb{R}$
- Powers:  $x^a$  on  $\mathbb{R}_{++}$  for  $a \geq 1$  or  $a \leq 0$
- Powers of absolute value:  $|x|^p$  on  $\mathbb{R}$ , for  $p \geq 1$
- Negative entropy:  $x \log x$  on  $\mathbb{R}_{++}$

## ■ Concave:

- Affine:  $ax + b$  on  $\mathbb{R}$ , for any  $a, b \in \mathbb{R}$
- Powers:  $x^a$  on  $\mathbb{R}_{++}$  for  $0 \leq a \leq 1$
- Logarithm:  $\log x$  on  $\mathbb{R}_{++}$

# Examples on $\mathbb{R}^N$ and $\mathbb{R}^{M \times N}$

- Examples on  $\mathbb{R}^N$ 
  - Affine function:  $\mathbf{a}^T \mathbf{x} + b$
  - Norms:  $\|\mathbf{x}\|_p = (\sum_{j=1}^N |x_j|^p)^{1/p}$  for  $p \geq 1$ ;  $\|\mathbf{x}\|_\infty = \max_j |x_j|$
- Examples on  $\mathbb{R}^{M \times N}$  ( $M \times N$  matrices)
  - Affine function:

$$f(\mathbf{X}) = \sum_{i=1}^M \sum_{j=1}^N A_{ij} X_{ij} + b = \text{tr}(\mathbf{A}^T \mathbf{X}) + b$$

- Spectral norm (maximum singular value):

$$f(\mathbf{X}) = \|\mathbf{X}\|_2 = \sigma_{\max}(\mathbf{X}) = (\lambda_{\max}(\mathbf{X}^T \mathbf{X}))^{1/2}$$

# Example: Quadratic

- $f(x) = x^2$

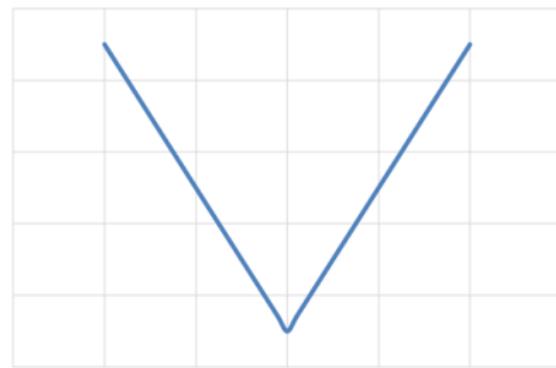


- Proof of convexity:

$$\begin{aligned}(\alpha x + (1 - \alpha)y)^2 &= \alpha^2 x^2 + 2\alpha(1 - \alpha)xy + (1 - \alpha)^2 y^2 \\&= \alpha x^2 + (1 - \alpha)y^2 - \alpha(1 - \alpha)(x^2 - 2xy + y^2) \\&\leq \alpha x^2 + (1 - \alpha)y^2\end{aligned}$$

# Example: Abs

- $f(x) = |x|$

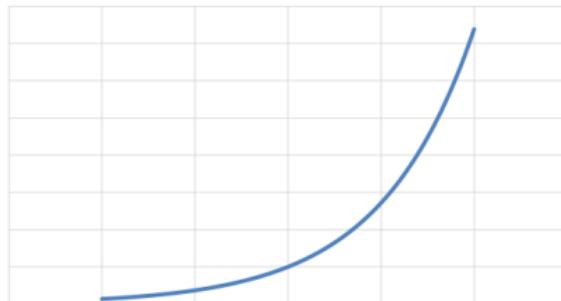


- Proof of convexity:

$$\begin{aligned} |\alpha x + (1 - \alpha)y| &\leq |\alpha x| + |(1 - \alpha)y| \\ &= \alpha|x| + (1 - \alpha)|y| \end{aligned}$$

# Example: Exponential

- $f(x) = e^x$



- Proof of convexity (assuming  $x > y$ ):

$$\begin{aligned} e^{\alpha x + (1-\alpha)y} &= e^y e^{\alpha(x-y)} = e^y \sum_{n=0}^{\infty} \frac{1}{n!} \alpha^n (x-y)^n \\ &\leq e^y \left( 1 + \alpha \sum_{n=1}^{\infty} \frac{1}{n!} (x-y)^n \right) \\ &= e^y ((1-\alpha) + \alpha e^{x-y}) = (1-\alpha)e^y + \alpha e^x \end{aligned}$$

# Properties of Convex Functions

- Any line segment we draw between two points lies above the curve
- Every local minimum is a **global** minimum
  - Why?
- This is what makes convex optimization easy
  - It suffices to find a local minimum because we know it will be global

# Properties of Convex Functions

- Non-negative combinations of convex functions are convex

$$h(\mathbf{x}) = af(\mathbf{x}) + bg(\mathbf{x})$$

- Affine scalings of convex functions are convex

$$h(\mathbf{x}) = f(\mathbf{Ax} + \mathbf{b})$$

- Pointwise maximum of convex functions are convex

$$h(\mathbf{x}) = \max\{f(\mathbf{x}), g(\mathbf{x})\}$$

- Compositions of convex functions are **NOT** generally convex
  - Neural nets are like this  $h(\mathbf{x}) = f(g(\mathbf{x}))$

# First-Order Condition

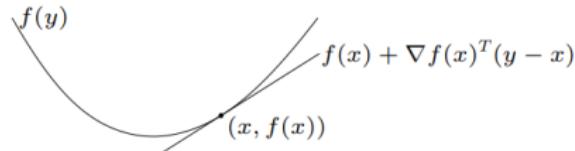
- $f$  is **differentiable** if the gradient

$$\nabla f(\mathbf{x}) = \left[ \frac{\partial f(\mathbf{x})}{\partial x_1}, \frac{\partial f(\mathbf{x})}{\partial x_2}, \dots, \frac{\partial f(\mathbf{x})}{\partial x_N} \right]^T$$

exists at each  $\mathbf{x} \in \text{dom}(f)$

- **1st-order condition:** differentiable  $f$  with convex domain is convex iff

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



- First-order approximation of  $f$  is global underestimator

# Second-Order Condition

- $f$  is **twice differentiable** if the Hessian  $\nabla^2 f(\mathbf{x}) \in \mathbb{S}^N$

$$\nabla^2 f(\mathbf{x})_{ij} = \frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j}$$

- exists at each  $\mathbf{x} \in \text{dom}(f)$
- **2nd-order condition:** twice differentiable  $f$  with convex domain is convex iff

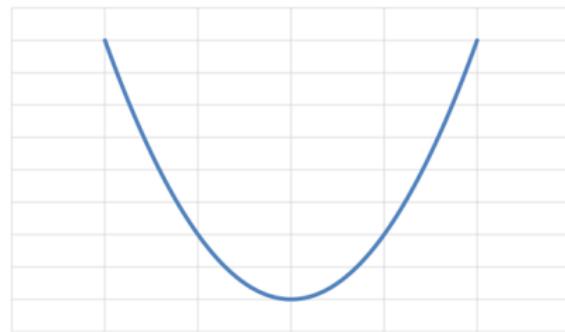
$$\nabla^2 f(\mathbf{x}) \succeq 0$$

- This means that the Hessian matrix is positive semidefinite

$$\mathbf{A} \succeq 0 \iff \forall \mathbf{x}, \mathbf{x}^T \mathbf{A} \mathbf{x} \geq 0$$

# Example: Quadratic

- $f(x) = x^2$

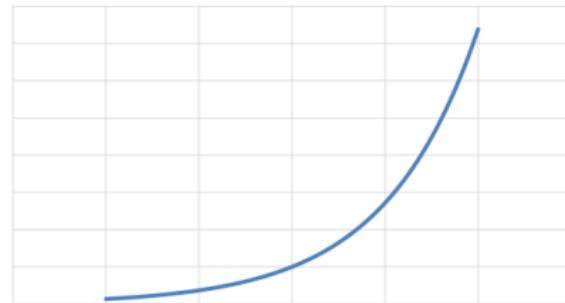


- Proof of convexity:

$$f''(x) = 2 \geq 0$$

# Example: Exponential

- $f(x) = e^x$

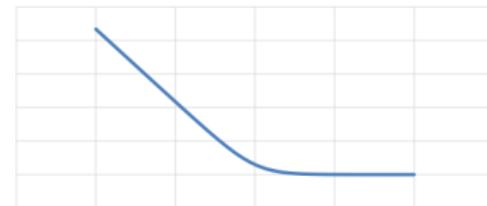


- Proof of convexity:

$$f''(x) = e^x \geq 0$$

# Example: Logistic Loss

- $f(x) = \log(1 + e^{-x})$



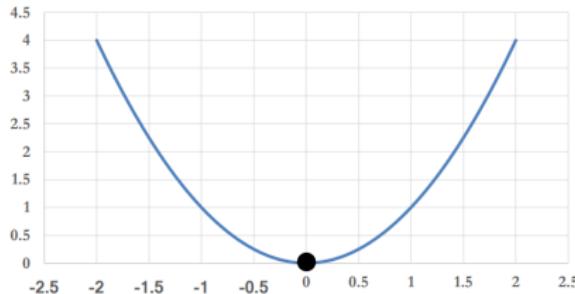
- Proof of convexity:

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

$$f''(x) = \frac{1}{(1 + e^x)(1 + e^{-x})} \geq 0$$

# Parameter Estimation

- Optimization in machine learning
  - Goal is to optimize a loss function defined in a machine learning problem for parameter estimation
- Convex optimization in machine learning
  - Goal is to minimize a convex loss function with respect to parameters
  - Example:  $\ell(\theta) = \theta^2$

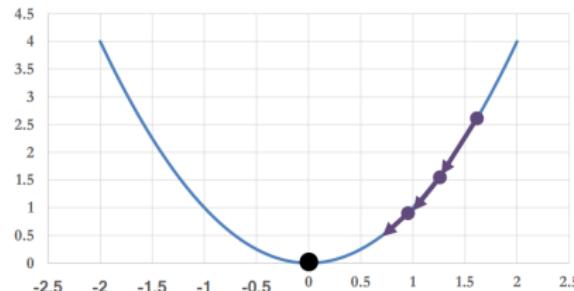


# Gradient Descent

- Gradient descent (GD) uses the following update formula

$$\theta \leftarrow \theta - \alpha \nabla \ell(\theta)$$

- $\alpha$  is the learning rate
- $\nabla \ell(\theta)$  is the gradient of  $\ell(\theta)$  evaluated at  $\theta$
- Example:  $\ell(\theta) = \theta^2$ ,  $\nabla \ell(\theta) = 2\theta$ , and  $\alpha = 0.125$



# Gradient Descent

- GD first chooses an initial point  $\theta^{(0)}$  and repeats:

$$\theta^{(t+1)} = \theta^{(t)} - \alpha \nabla \ell(\mathcal{D}; \theta^{(t)}) = \theta^{(t)} - \alpha \frac{1}{M} \sum_{i=1}^M \nabla \ell(\mathbf{x}^{(i)}, y^{(i)}; \theta^{(t)})$$

- How to stop?
  - $\|\theta^{(t+1)} - \theta^{(t)}\|_2 \leq \epsilon$
  - $\|\nabla \ell(\mathcal{D}; \theta^{(t)})\|_2 \leq \epsilon$ 
    - $\|\theta\|_2 = \sqrt{\sum_j |\theta_j|^2}$  is the  $\ell_2$ -norm of  $\theta$
    - $\epsilon$  is a small positive constant, e.g.,  $10^{-6}$

# Gradient Descent

- A negative gradient step can decrease the loss function

**Proof.** Let  $\boldsymbol{\theta}^{(0)}$  be any initial point. Using the first-order Taylor expansion for the next point  $\boldsymbol{\theta}^{(1)} = \boldsymbol{\theta}^{(0)} - \alpha \nabla \ell(\boldsymbol{\theta}^{(0)})$ , we have

$$\begin{aligned}\ell(\boldsymbol{\theta}^{(1)}) &= \ell(\boldsymbol{\theta}^{(0)}) + \nabla \ell(\boldsymbol{\theta}^{(0)})^T (\boldsymbol{\theta}^{(1)} - \boldsymbol{\theta}^{(0)}) \\ &= \ell(\boldsymbol{\theta}^{(0)}) + \nabla \ell(\boldsymbol{\theta}^{(0)})^T (-\alpha \nabla \ell(\boldsymbol{\theta}^{(0)})) \\ &= \ell(\boldsymbol{\theta}^{(0)}) - \alpha \|\nabla \ell(\boldsymbol{\theta}^{(0)})\|_2^2\end{aligned}$$

Hence, if  $\nabla \ell(\boldsymbol{\theta}^{(0)}) \neq 0$  and  $\alpha$  is sufficiently small, we have

$$\ell(\boldsymbol{\theta}^{(1)}) - \ell(\boldsymbol{\theta}^{(0)}) = -\alpha \|\nabla \ell(\boldsymbol{\theta}^{(0)})\|_2^2 < 0$$

For sufficiently small  $\alpha$ ,  $\boldsymbol{\theta}^{(1)}$  is an improvement over  $\boldsymbol{\theta}^{(0)}$

# Gradient Descent

- Initial point matters
  - Consider  $y = f(x) = x \cos(x)$

- The final convergence point depends a lot on the initial point
- To improve optimization, run GD multiple times with different initializations, and keep the best minimum of all times (may take a long time to run)

[www.charlesbordet.com/assets/images/gradient-descent/gradientdescent-f3.gif](http://www.charlesbordet.com/assets/images/gradient-descent/gradientdescent-f3.gif)

# Gradient Descent

## ■ Learning rate matters

- If  $\alpha$  is too big, GD moves more quickly with a high risk that the algorithm will never converge
- If  $\alpha$  is too small, GD moves slowly...so slowly that you might just lose patience and never reach the minimum
- Consider  $y = f(x) = 2x^2 \cos(x) - 5x$

- Unfortunately, no magic bullet exists to find the perfect learning rate, and you have to test several values and pick the best

[www.charlesbordet.com/assets/images/gradient-descent/gradientdescent-alpha0.001.gif](http://www.charlesbordet.com/assets/images/gradient-descent/gradientdescent-alpha0.001.gif)

[www.charlesbordet.com/assets/images/gradient-descent/gradientdescent-alpha0.2.gif](http://www.charlesbordet.com/assets/images/gradient-descent/gradientdescent-alpha0.2.gif)

# Stochastic Gradient Descent

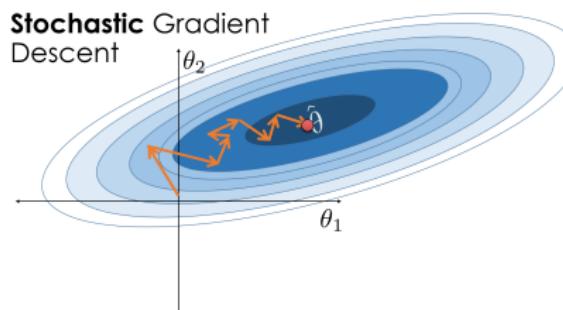
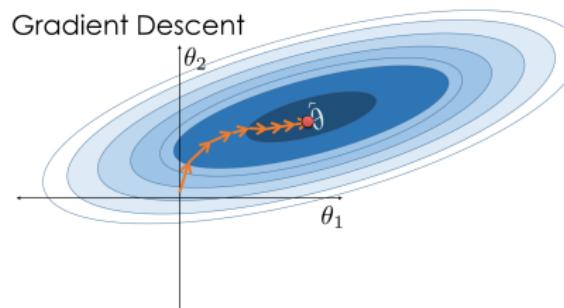
- The computational cost for GD at each iteration is  $\mathcal{O}(M)$ , which grows linearly with  $M$ 
  - It is prohibitive to scale up to huge datasets
- Stochastic gradient descent (SGD)
  - Reduces the computational cost at each iteration to  $\mathcal{O}(1)$ 
    - At  $t$ -th iteration, randomly sample a single training example and compute the gradient to update  $\theta^{(t)}$

$$\theta^{(t+1)} = \theta^{(t)} - \alpha \nabla \ell(\mathbf{x}^{(i)}, y^{(i)}; \theta^{(t)})$$

- In expectation, SGD is just GD

$$\mathbb{E}[\nabla \ell(\mathbf{x}, y; \theta)] = \frac{1}{M} \sum_{i=1}^M \nabla \ell(\mathbf{x}^{(i)}, y^{(i)}; \theta) = \nabla \ell(\mathcal{D}; \theta)$$

# Stochastic Gradient Descent



# Stochastic Gradient Descent

## ■ Advantages:

- It is easy to fit into memory and computationally fast due to a single training sample being processed at each iteration
- The steps have oscillations, which may help the algorithm get out of bad local minima

## ■ Disadvantages:

- The steps are noisy. The objective does not always decrease for each step and it may take longer to achieve convergence
- It loses the advantage of vectorized operations as it deals with only a single example at a time

# Mini-Batch Gradient Descent: A Compromise

- Unlike SGD, mini-batch gradient descent randomly samples a small subset of training data  $\mathcal{B} \subset \mathcal{D}$  at  $t$ -th iteration, and compute the gradient to update  $\theta^{(t)}$

$$\theta^{(t+1)} = \theta^{(t)} - \alpha \nabla \ell(\mathcal{B}; \theta^{(t)}) = \theta^{(t)} - \alpha \frac{1}{|\mathcal{B}|} \sum_{(\mathbf{x}, y) \in \mathcal{B}} \nabla \ell(\mathbf{x}, y; \theta^{(t)})$$

- Mini-batch gradient descent and its variants are extremely popular and extensively used in deep learning

# Summary

- Convex optimization
  - Convex set:
    - $\mathbf{x}^{(1)}, \mathbf{x}^{(2)} \in \mathcal{X}, 0 \leq \alpha \leq 1 \implies \alpha\mathbf{x}^{(1)} + (1 - \alpha)\mathbf{x}^{(2)} \in \mathcal{X}$
  - Convex function:
    - $\text{dom}(f)$  is convex
    - $f(\alpha\mathbf{x} + (1 - \alpha)\mathbf{y}) \leq \alpha f(\mathbf{x}) + (1 - \alpha)f(\mathbf{y})$
  - Establish convexity
    - By definition
    - First-order condition:  $f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^T (\mathbf{y} - \mathbf{x})$
    - Second-order condition:  $\nabla^2 f(\mathbf{x}) \succeq 0$
- Numerical optimization methods
  - GD - guarantee to decrease loss; not scalable to large datasets
  - SGD - super fast to compute with noisy gradients
  - Mini-batch GD - a compromise between GD and SGD; the de facto method in deep learning