Basic Optimization

Mohsen Moghaddam, Ph.D.

Gary C. Butler Family Associate Professor
H. Milton Stewart School of Industrial and Systems Engineering
George W. Woodruff School of Mechanical Engineering
Georgia Institute of Technology

SAIL
Symbiotic & Augmented Intelligence Laboratory

Spring 2026

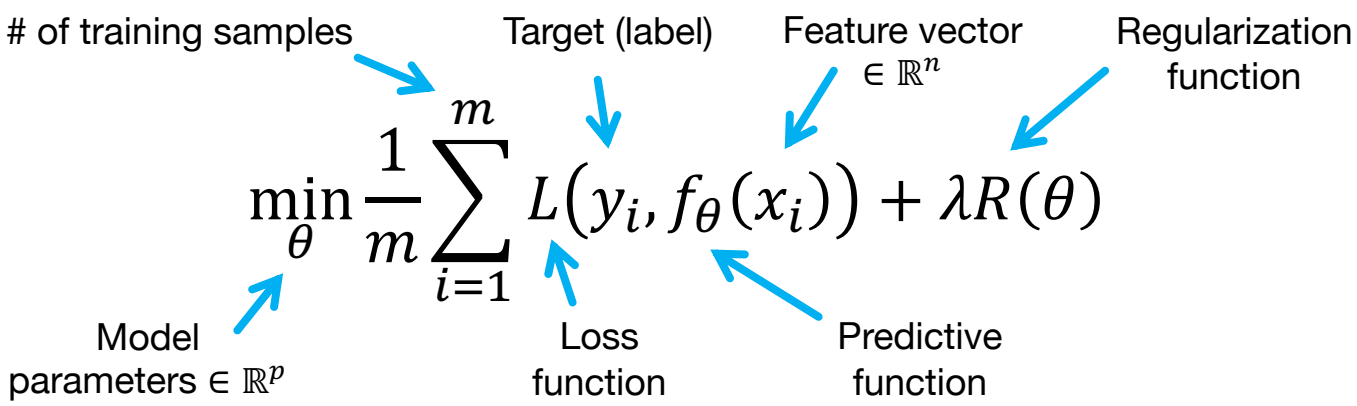
Learning Outcomes

- Formulate machine learning problems as optimization problems and explain the role of objectives, variables, and constraints
- Distinguish convex and non-convex problems and explain why convexity enables global optimality
- Recognize common convex functions and sets arising in machine learning models
- Apply gradient-based methods, including batch and stochastic gradient descent, to empirical risk minimization
- Select step-size strategies and stopping criteria for practical optimization
- Express optimization problems in matrix–vector form for efficient computation
- Formulate constrained optimization problems using the Lagrangian and interpret duality and KKT conditions in machine learning contexts

Motivation & Fundamentals

Why Optimization in ML?

Most ML problems can be written as:



Three components:

- Decision variables
- Constraints
- Objective function

Convex/nonconvex optimization

Modeling

- Statistics and geometric models
- Clustering
- PCA
- MLE
- Regression
- Support vector machine
- Deep learning

Solution Methods

- Heuristics (e.g., K-means)
- Special methods (e.g., PCA, EM)
- General methods (e.g., gradient descent, SGD)

Convex Optimization

Definition: An optimization problem specified by:

$$\begin{aligned} \text{minimize} \quad & f_0(x) && && \leftarrow \text{Convex objective function} \\ \text{subject to} \quad & f_i(x) \leq 0, && i = 1, \ldots, m && \leftarrow \text{Convex inequality constraints} \\ & h_i(x) = 0, && i = 1, \ldots, p && \leftarrow \text{Affine equality constraints} \end{aligned}$$

ML mapping:

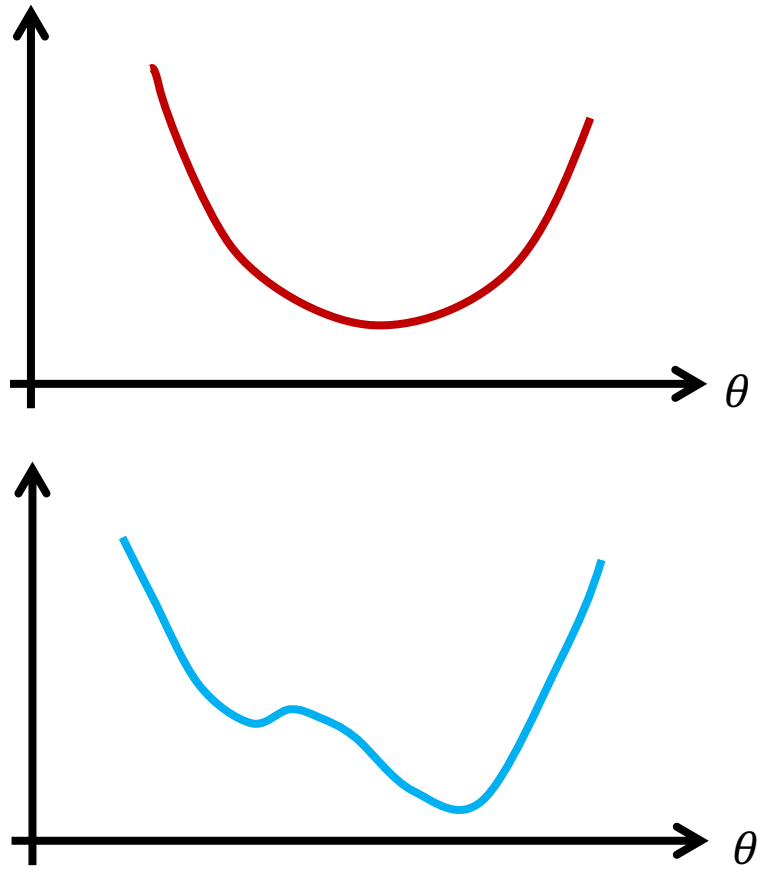
- Objective function → loss function
- Constraints → regularization or feasibility
- Decision variables → model parameters

Convex optimization problems are attractive because they have no "bad" local minima

Unconstrained Optimization

In ML, unconstrained problems are the most common case:

$$\text{minimize}_x \ f_0(x)$$



- **Examples:** Linear and logistic regression
- Constraints are often absorbed into the objective via **regularization**
- First order optimality condition:

$$\nabla f_0(x) = 0$$

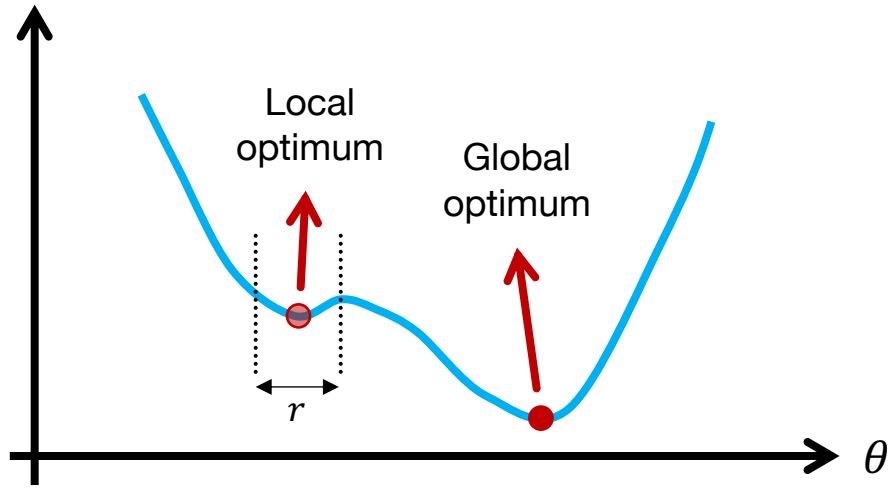
If $\nabla f_0(x) = 0$, does that guarantee we found the best solution?

Global vs Local Optimum

Global optimum: A point x^* in the feasible set is global optimum iff

$$f_0(x^*) \leq f_0(x)$$

for all x in the feasible set



Local optimum: A point x^* in the feasible set is local optimum iff there exists $r > 0$ such that for all $x \in \{x \mid \|x - x^*\| \leq r\}$ and also in the feasible set, we have

$$f_0(x^*) \leq f_0(x)$$

For convex optimization problem, any local optimum is also a global optimum

Constrained vs Unconstrained

Clustering: Unconstrained

- Optimization variables = cluster centers c
- Each cluster center is free to move anywhere in $\mathbb{R}^n$

$$\min_{c,\pi} \frac{1}{m}\sum_{i=1}^{m}\left\|x^i - c^{\pi(i)}\right\|^2$$

Regression: Unconstrained

- Parameter θ lives in $\mathbb{R}^n$ with no explicit constraint
- Regularization (Ridge/LASSO) is usually added

$$\min_{\theta} \frac{1}{m}\sum_{i=1}^{m}\left(y^i - \theta^\top x^i\right)^2$$

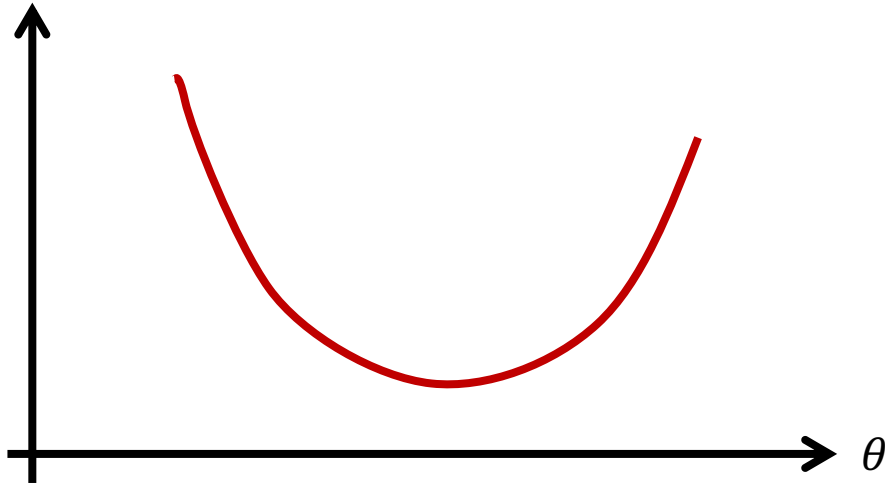
PCA: Constrained; $\|w\| = 1$ serves as an essential, **hard constraint**

$$\max_{w:\|w\|\leq 1} \frac{1}{m}\sum_{i=1}^{m}\left(w^\top x^i - w^\top \mu\right)^2$$

Convex vs Non-Convex

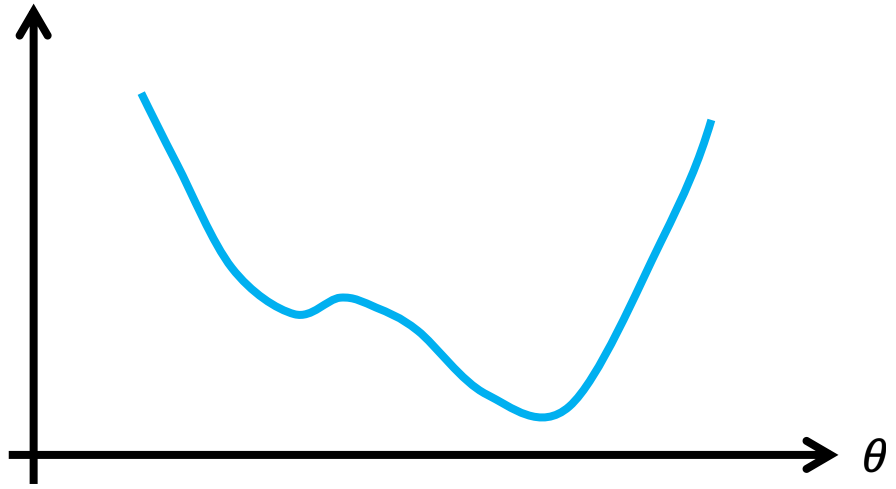
Convex Problem

- e.g., linear regression
- Can be solved efficiently (find global optimal solution) in polynomial time
- Gradient descent (or many acceleration methods; e.g., Newton method) can converge to global solution



Non-Convex Problem

- e.g., clustering, PCA
- Cannot find global solution in polynomial time (NP-hard)
- Use heuristics to find local optimal solution
- “Non-convex” does not mean “unsolvable,” but it does mean we rely on heuristics and initialization



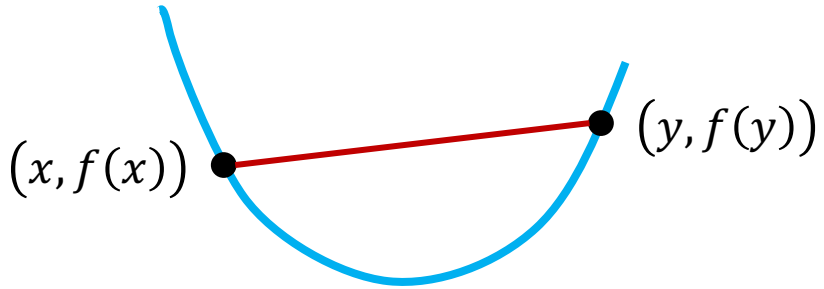
Convex Functions & Sets

Convex Functions

Definition: A function $f: \mathbb{R}^n \to \mathbb{R}$ is **convex** if the domain $\mathbf{dom} f$ is a convex set and if for all $x, y \in \mathbf{dom} f$ and $0 \leq \alpha \leq 1$, we have

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y)$$

Geometrically, the line segment between $\big(x, f(x)\big)$ and $\big(y, f(y)\big)$ lies **above** the graph of f



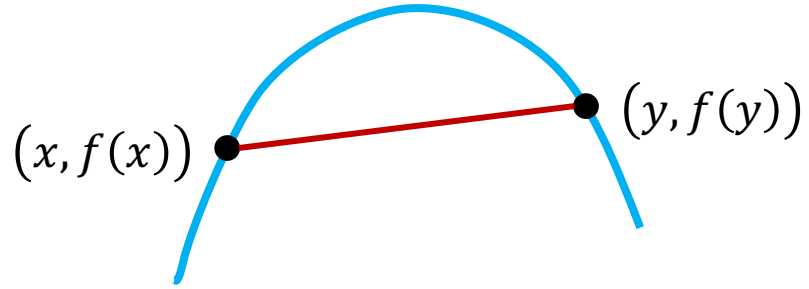
A convex function has a single "bowl-shaped" landscape

Concave Functions

Definition: A function $f: \mathbb{R}^n \to \mathbb{R}$ is **concave** if the domain $\mathbf{dom} f$ is a convex set and if for all $x, y \in \mathbf{dom} f$ and $0 \leq \alpha \leq 1$, we have

$$f(\alpha x + (1-\alpha)y) \geq \alpha f(x) + (1-\alpha) f(y)$$

Geometrically, the line segment between $\big(x, f(x)\big)$ and $\big(y, f(y)\big)$ lies **below** the graph of f



Maximizing a concave function is equivalent to minimizing a convex function

Examples

Function	Convex?	Concave?
Exponential	$f(x) = e^{ax}$ for any $a \in \mathbb{R}$	–
Powers	$f(x) = x^a$ on $\mathbb{R}_{++}$ when $a \geq 1$ or $a \leq 0$	$f(x) = x^a$ on $\mathbb{R}_{++}$ when $0 \leq a \leq 1$
Powers of absolute value	$f(x) = \lvert x \rvert^p$ for $p \geq 1$	–
Logarithm	–	$f(x) = \log x$ on $\mathbb{R}_{++}$
Negative entropy	$f(x) = x \log x$ on $\mathbb{R}_{++}$	–
Norms	All norms	–
Max function	$f(x) = \max\{x_1, \ldots, x_n\}$	–
Log-determinant	$f(x) = -\log \det X$	$f(x) = \log \det X$

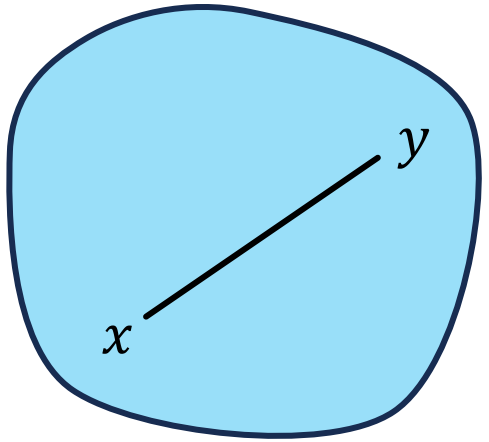
Convex Sets

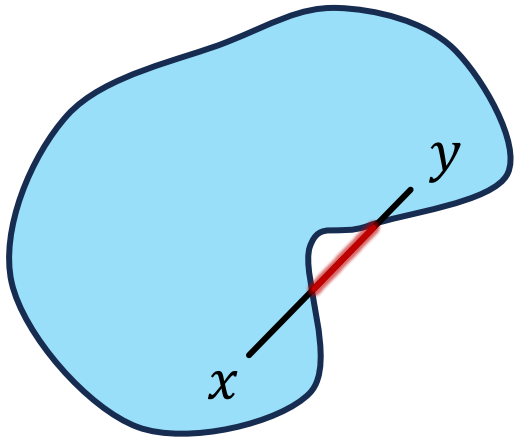
Definition: A set A is convex, if for every $0 \leq \alpha \leq 1$ it satisfies

$$\forall x, y \in A \quad \Leftrightarrow \quad \alpha x + (1-\alpha) y \in A$$

The line segment between any two points is also in the set

Convex set





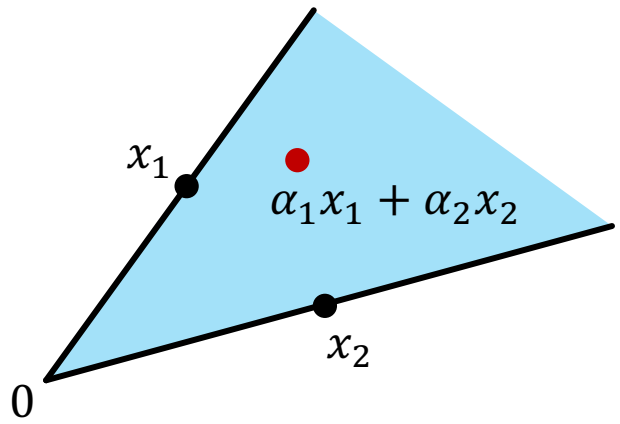
Non-convex set

Convex sets ensure feasibility regions without "holes"

Common Convex Sets

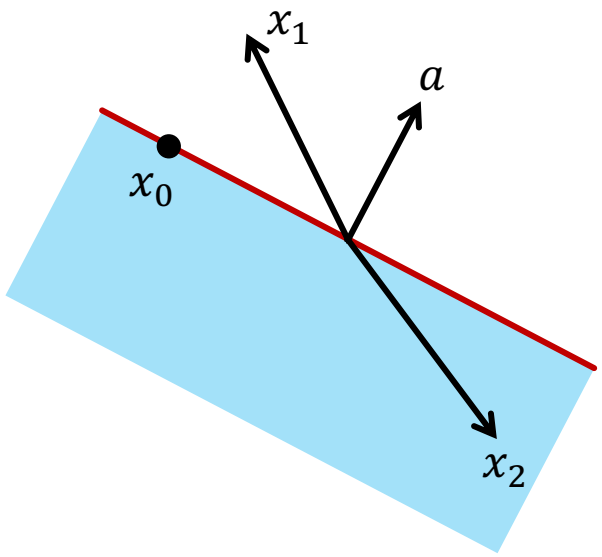
Cones: A set C is a convex cone, if for any $x_1, x_2 \in C$ and $\alpha_1, \alpha_2 \geq 0$, we have

$$\alpha_1 x_1 + \alpha_2 x_2 \in C$$



Hyperplanes: A set is hyperplane if

$$\{x \mid a^\top (x - x_0) = 0, a \neq 0\}$$



Halfspaces: A set is halfspace if

$$\{x \mid a^\top (x - x_0) \leq 0, a \neq 0\}$$

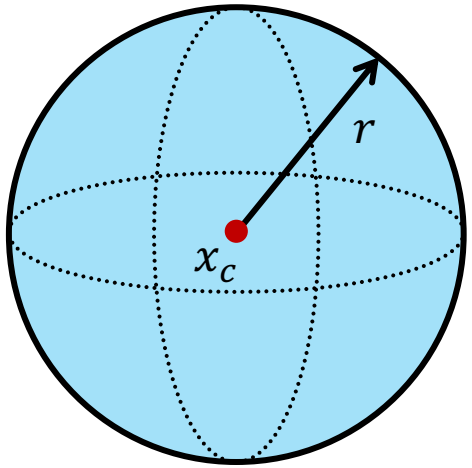
Is a hyperplane itself a feasible region or a boundary?

Common Convex Sets (cont.)

Euclidean balls: A Euclidean ball has the form

$$B(x_c, r) = \{x \mid \|x - x_c\|_2 \leq r\}$$

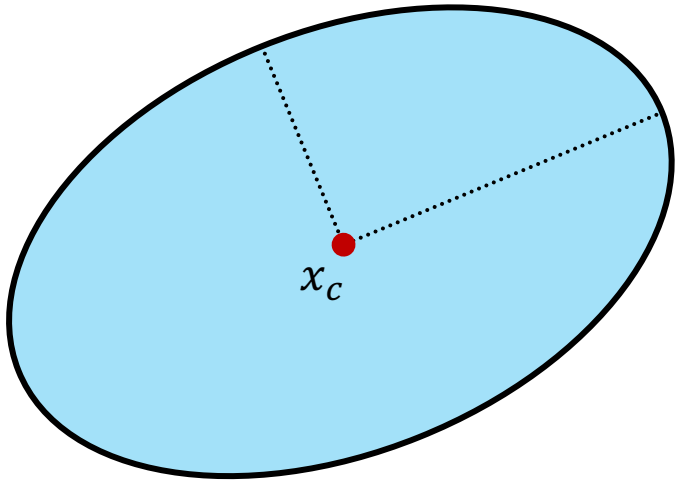
ℓ_2 regularization corresponds to Euclidean balls



Ellipsoids: Used in support vector novelty detection

$$E = \{x \mid (x - x_c)^\top P^{-1}(x - x_c) \leq r\}$$

The eigenvectors and eigenvalues determine the direction and shape of the semi-axes

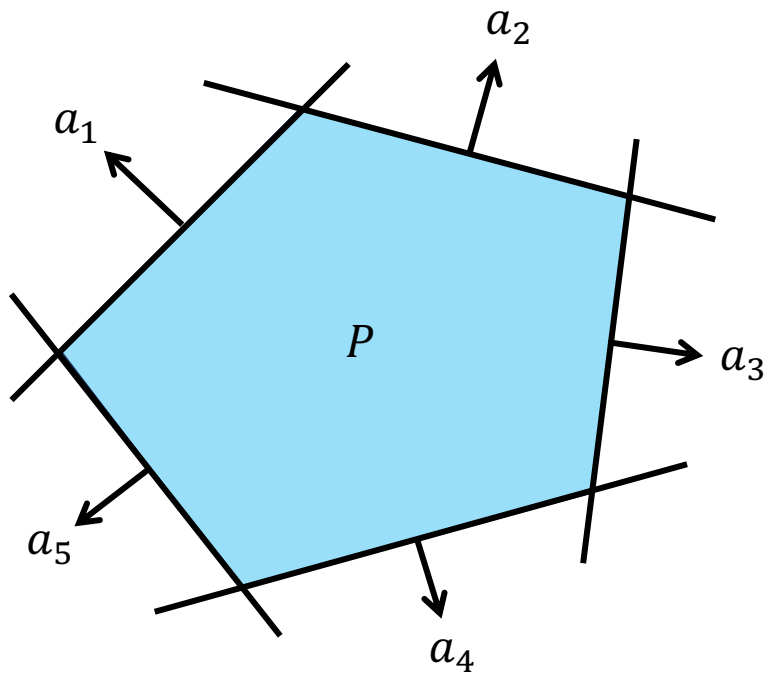


Common Convex Sets (cont.)

Polyhedra: Intersection of a **finite** set of halfspaces or hyperplanes

$$P = \{x \mid a_j^\top x \leq b_j, j = 1, \ldots, m, \qquad c_j^\top x = d_j, j = 1, \ldots, p\}$$

- It is defined as the solution set of a finite number of linear equalities and inequalities
- Used in various machine learning techniques like SVM
- ℓ_1 constraints lead to sparse solutions

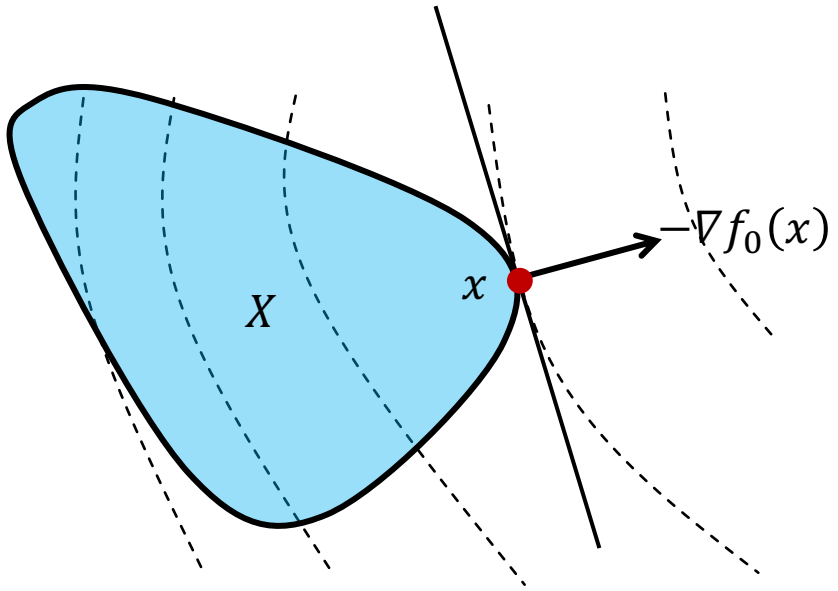


Unconstrained Optimization

First-Order Optimality Condition

Constrained optimization: Given feasible set X

$$x \text{ is optimal} \quad \Leftrightarrow \quad \nabla f_0(x)^\top (y - x) \geq 0, \quad \forall\, y \in X$$



Unconstrained optimization: If $X = \mathbb{R}^n$, then all directions $y = x - \epsilon \nabla f_0(x)$ are feasible

$$x \text{ is optimal} \quad \Leftrightarrow \quad \nabla f_0(x) = 0$$

Constrained boundary optimum: If x is on the boundary of X, not all directions d and so, optimality requires $\nabla f_0(x)^\top d \geq 0$ for all feasible d; therefore,

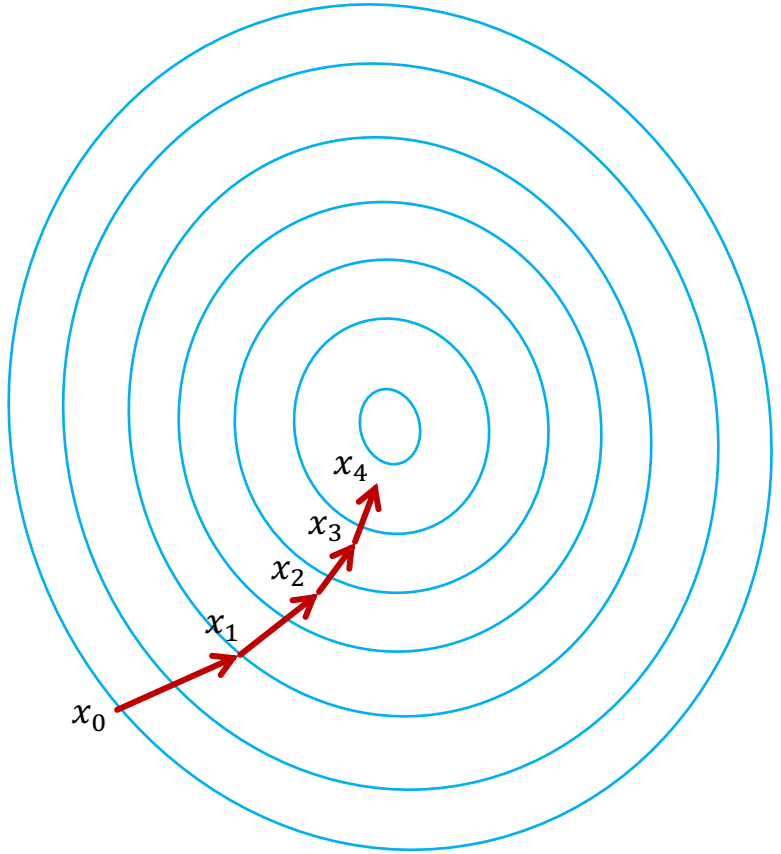
$$-\nabla f_0(x) \;\perp\; \text{tangent space of } X \text{ at } x$$

Gradient Descent

- One way to solve an unconstrained optimization problem is gradient descent (GD)
- Given an initial guess, we iteratively refine the guess by taking the direction of the **negative gradient**
- Think about going down a hill by taking the **steepest direction** at each step
- Update rule:

$$x^{t+1} = x^t - \gamma_t \nabla f_0(x^t)$$

$\gamma_t > 0$ is called the step size or **learning rate**



In ML, GD iteratively improves model parameters to reduce training loss

Gradient Descent (cont.)

How GD works: It operationalizes the condition $\nabla f(x) = 0$

- For an unconstrained problem, a necessary optimality condition is

$$\nabla f_0(x^*) = 0$$

- If gradient descent reaches such this point, then

$$x^{t+1} = x^t - \gamma_t \nabla f_0(x^t) = x^t$$

- In practice, GD rarely reaches this point, so it stops when

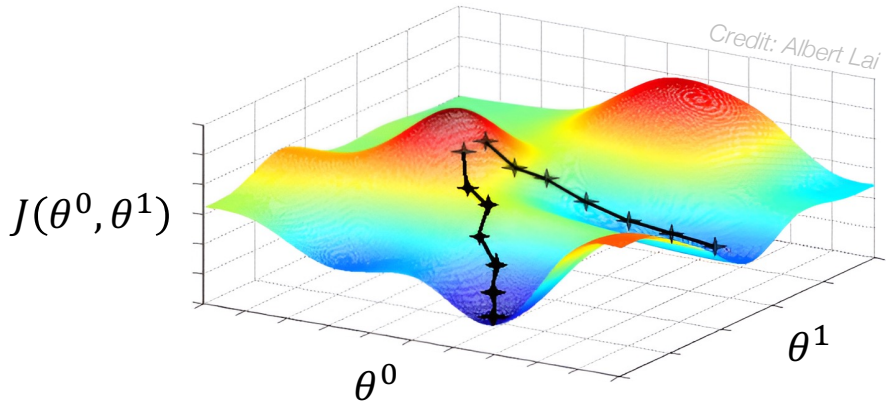
$$\|x^{t+1} - x^t\| < \epsilon$$ ← ϵ: a very mall coefficient

What could go wrong if γ_t is poorly chosen?

Gradient Descent for Linear Regression

Batch gradient descent (or **ascent**) algorithm to minimize MSE:

$$\min_{\theta} L(\theta) \coloneqq \frac{1}{m}\sum_{i=1}^{m}\left(y^i - \theta^\top x^i\right)^2$$



Why average all samples here?

Input: data $\{(x^i, y^i)\}_{i=1}^m$, step size γ_t, tolerance ϵ

Initialize: θ^0

Repeat for $t = 1,2,\ldots$

1. Compute residuals

$$r^i \coloneqq y^i - \theta^\top x^i, \qquad \forall i = 1,\ldots,m$$

2. Update parameters:

$$\theta^{t+1} \coloneqq \theta^t + \boxed{\frac{\gamma_t}{m}\sum_{i=1}^{m} r^i x^i} \quad \nearrow -\gamma_t \nabla L(\theta)$$

until $\|\theta^{t+1} - \theta^t\| \leq \epsilon$

Output: θ^{t+1}

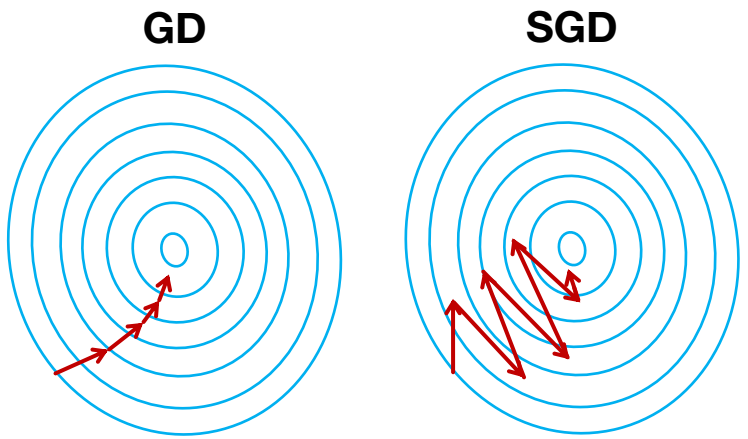
Stochastic Optimization

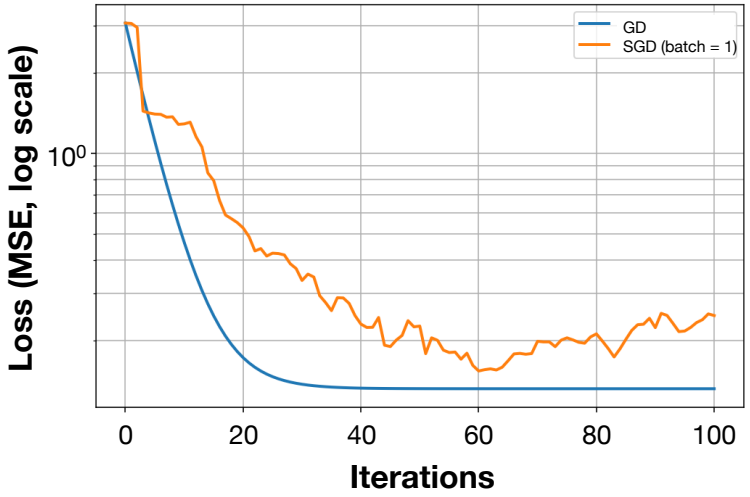
Batch vs Stochastic Gradient

- The algorithm is called **batch gradient descent**
- To compute the gradient at each iteration, we need to sum over all data points in the dataset
- What if we have a huge dataset?
- Note that gradient **decouples**—it consists of sum of evaluations over individual samples:

$$\nabla L(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(y^i - (\theta^t)^\top x^i\right)x^i$$

SGD trades accuracy for speed and scalability





Stochastic Gradient Descent

- At each iteration, randomly sample a small subset S_t data point (x_i, y_i), $i \in S_t$ (extreme case: one sample in S_t)
- Use gradient estimated using a small subset of data

$$\nabla L(\theta) = \frac{1}{|S_t|} \sum_{i \in S_t} \left(y^i - (\theta^t)^\top x^i\right) x^i$$

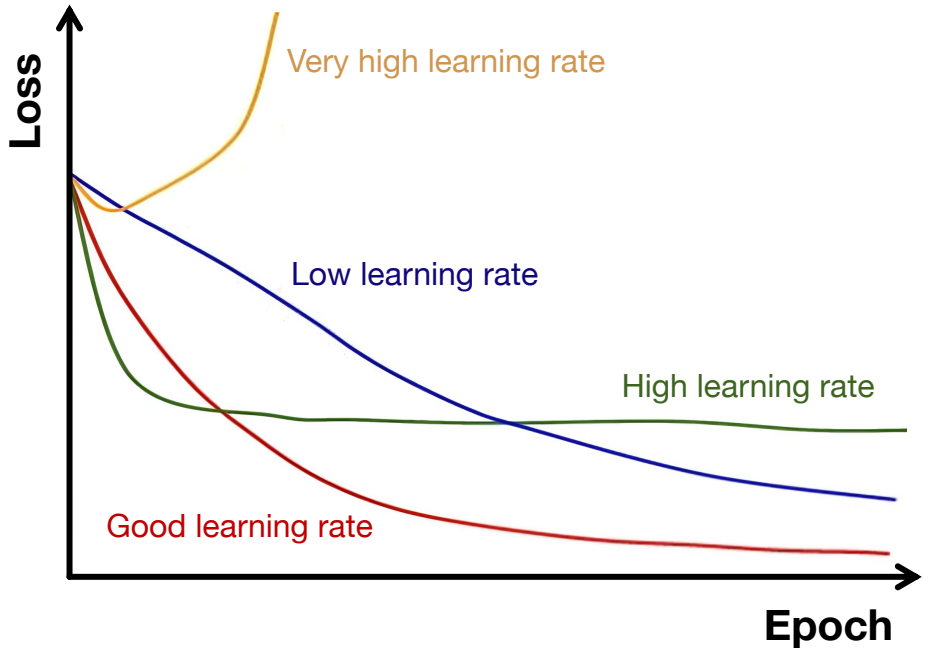
- Each iteration uses a different subset S_t, $t = 1, 2, \ldots$
- Eventually loop through the entire training data, and may loop through the data multiple times

If SGD uses noisy gradients, why doesn't it diverge?

Step Size (Learning Rate)

Step size γ_t (aka **learning rate**) is a positive scalar that controls how far the optimization algorithm moves along the negative gradient direction at each iteration

- If γ_t is too large, the algorithm may diverge; if too small, convergence can be very slow
- Constant step size $\gamma_t = \gamma$
- Diminishing step size $\gamma_t \propto 1/t$
- Constant step length $\gamma_t \propto 1/\|\nabla L(\theta^t)\|$
- Adaptive step-sizes automatically adjust learning rates during training



In practice, adaptive methods (Adam, RMSProp) reduce tuning effort

Stopping Criteria

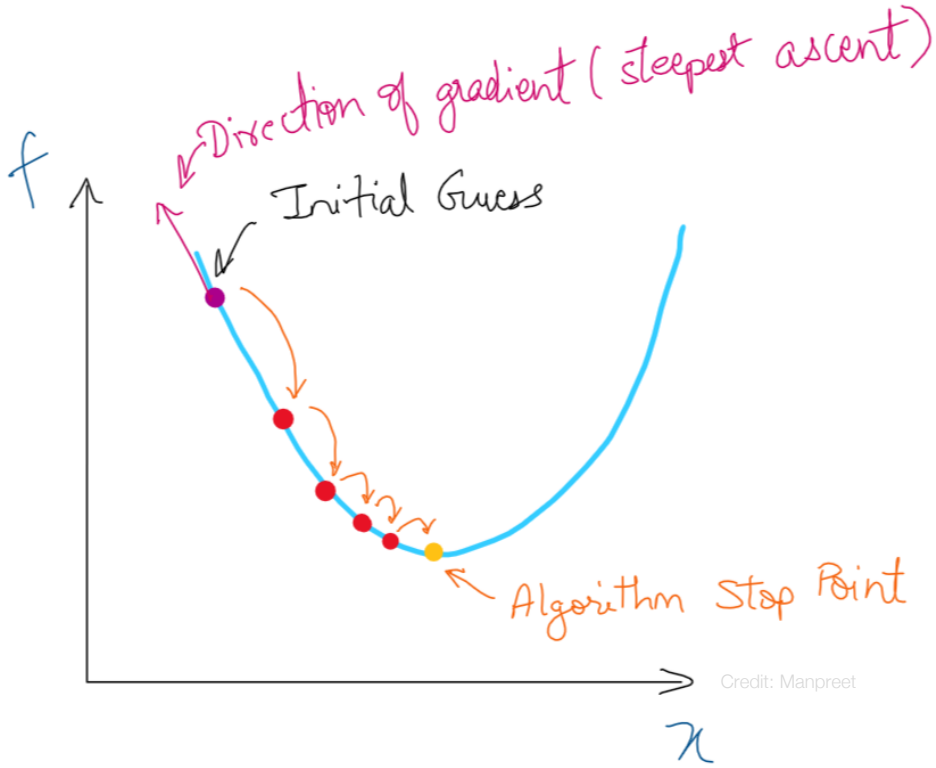
A set of rule that determine when to terminate the optimization algorithm, balancing **computational cost** and **solution accuracy**

- Constant number of iterations
- Stop when loss function change is small*:

$$|L(\theta^t) - L(\theta^{t-1})| < \varepsilon$$

- Stop when parameter change small*:

$$\|\theta^t - \theta^{t-1}\| < \varepsilon$$



In machine learning, stopping is often based on validation loss

* These criteria monitor convergence of the training loss, not generalization performance

Matrix Formulation & Constraints

Matrix-Vector Form: Linear Regression

$$L(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(y^i - \theta^\top x^i\right)^2 \xrightarrow{\text{Gadient}} \nabla L(\theta) = -\frac{2}{m}\sum_{i=1}^{m} x^i\left(y^i - \theta^\top x^i\right)$$

Matrix-vector form: $L(\theta) = \frac{1}{m}\|y - X\theta\|_2^2$

$$y = [y^1 \cdots y^m]^\top \qquad X = \begin{bmatrix} (x^1)^\top \\ \vdots \\ (x^m)^\top \end{bmatrix} \in \mathbb{R}^{m\times n}$$

Gradient: $\nabla L(\theta) = -\frac{2}{m}X^\top(y - X\theta)$

Why do we care about matrix form instead of scalar equations?

Handling Constraints

For an **unconstrained** problem, the condition becomes

$$\nabla f_0(x) = 0$$

For **constrained** problems, we need to use the Lagrange multipliers to convert constraints into penalties

$$L(x, \mu, \lambda) = f_0(x) + \underbrace{\sum_{j=1}^{p} \mu_j h_j(x)}_{\text{Equality Constraints}} + \underbrace{\sum_{i=1}^{m} \lambda_i f_i(x)}_{\text{Inequality Constraints}} \qquad \text{s.t.} \quad \lambda_i \geq 0$$

It is a **lower bound** of $f_0(x)$ for all $x \in X$, since $h_j(x)$=0, $f_i(x) \leq 0$ and $\lambda_i \geq 0$ and $L(x, \mu, \lambda) \leq f_0(x)$ for all $x \in X$

Lagrange Dual Function

The dual problem provides **bounds** and alternative **optimization paths**

$$g(\mu, \lambda) = \inf_x L(x, \mu, \lambda)$$

→ **Definition (Infimum):** Let $S \subseteq R$. The infimum of S, denoted $\inf S$, is the greatest number a such that $a \leq s$, for all $s \in S$

It is the lower bound for the optimal value

$$g(\mu, \lambda) = \inf_x L(x, \mu, \lambda) \leq L(x^*, \mu, \lambda) \leq f_0(x^*)$$

We want to maximize the lower bound to make it tight

$$g(\mu^*, \lambda^*) = \max g(\mu, \lambda)$$

Primal and Dual Problems

Primal problem:

$$\begin{aligned} \text{minimize} \quad & f_0(x) \\ \text{subject to} \quad & f_i(x) \leq 0, \ i = 1, \ldots, m \\ & h_i(x) = 0, \ i = 1, \ldots, p \end{aligned}$$

Dual problem:

$$\begin{aligned} \text{maximize} \quad & g(\mu, \lambda) \\ \text{subject to} \quad & \lambda_i \geq 0, \ i = 1, \ldots, m \end{aligned}$$

Strong duality: For convex problems satisfying regularity conditions:

$$g(\mu^*, \lambda^*) = f_0(x^*)$$

Slater's condition: There exists an x inside the relative interior of the domain X such that $f_i(x) < 0, \ i = 1, \ldots, m$

KKT Conditions

Conditions for an optimal triplet (x^*, μ^*, λ^*):

$$\nabla f_0(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla f_i(x^*) + \sum_{j=1}^{p} \mu_j^* \nabla h_j(x^*) = 0 \quad \leftarrow \text{Stationarity}$$

$$\lambda_i^* f_i(x^*) = 0 \quad \leftarrow \text{Complementary slackness}$$

$$f_i(x^*) \leq 0$$
$$h_j(x^*) = 0 \quad \Big\} \text{Primal feasibility}$$

$$\lambda_i^* \geq 0 \quad \leftarrow \text{Dual feasibility}$$

KKT conditions generalize "gradient = 0" to constrained problems

Key Takeaways

What We Learned This Week

- Machine learning training can be framed as optimization of an objective function over model parameters
- Convex vs. non-convex problems differ fundamentally in guarantees, difficulty, and solution strategies
- Gradient descent optimizes smooth objectives using full-data gradients and has stable convergence
- Stochastic gradient descent uses noisy gradient estimates, enabling scalability to large datasets
- Learning rate, batch size, and stopping criteria critically affect optimization performance
- Matrix-vector formulations enable efficient, scalable implementations
- Constraints and regularization can be handled using Lagrangians, duality, and KKT conditions