Week 8

Lecture 1: Introduction to Optimization

Three Pillars of Machine Learning

Machine learning theory rests on three foundational mathematical pillars:

Pillar 1: Linear Algebra

Role: Structure and relationships in data.

Linear algebra helps model structural relationships between variables. For example, suppose we collect data on height and weight of individuals:

- Each data point is represented as a vector in $\mathbb{R}^2$.
- A linear relationship between height and weight can be modeled as:

$$y = wx + b$$

where:

- x represents height,
- y represents weight,
- w, b are parameters.

This structural modeling relies fundamentally on linear algebra.

Pillar 2: Probability

Role: Modeling noise and uncertainty in data.
In real data:

- Observations do not lie exactly on a single line.
- Deviations occur due to measurement errors, variability, or missing information.
  We model these deviations as noise:

$$y = wx + b + \epsilon$$

  where:
- ϵ represents uncertainty or noise.

Probability theory provides the mathematical framework to model and reason about such uncertainty.

Pillar 3: Optimization

Role: Converting data into decisions.
There are infinitely many possible lines relating height and weight. The question is:

> Which line best represents the relationship in the observed data?

This requires defining a notion of "best."
Optimization provides the mathematical framework to:

- Compare different models,
- Quantify their performance,
- Select the best one according to a defined criterion.

Example: Height and Weight Prediction

Data Representation

Let data be:

$$\{(x_i, y_i)\}_{i=1}^{n}$$

where:

- x_i is height,
- y_i is weight.

We assume a linear model:

$$\hat{y}_i = wx_i + b$$

Multiple Possible Models

There are infinitely many candidate lines:

$$y = wx + b$$

Each choice of w, b defines a different line.
Thus, we must choose (w, b) optimally.

Optimization as Decision-Making

Objective Function

To determine the best parameters, define a loss function:

$$L(w, b) = \frac{1}{n} \sum_{i=1}^{n} (y_i - (wx_i + b))^2$$

The goal is:

$$\min_{w,b} L(w, b)$$

This is an optimization problem.

Summary of the Three Pillars

Linear Algebra

- Models structure in data.
- Represents data as vectors and matrices.
- Captures linear relationships.

Probability

- Models uncertainty and noise.
- Quantifies randomness.
- Allows principled statistical reasoning.

Optimization

- Selects the best model among many.
- Converts data and models into decisions.
- Solves problems of the form:

$$\min_{\theta} f(\theta) \quad \text{or} \quad \max_{\theta} f(\theta)$$

Optimization: Conceptual Definition

Optimization is the mathematical discipline concerned with:

> Finding the best element from a set of feasible solutions according to a given objective function.

In machine learning:

- Feasible solutions: model parameters.
- Objective function: loss or risk.
- Best: minimum loss or maximum likelihood.

Core Insight

Machine learning requires:

1. Structure modeling via linear algebra.
2. Uncertainty modeling via probability.
3. Decision selection via optimization.
   Together, these three pillars form the foundation upon which machine learning theory is built.

```
******************************************************************************
```

Lecture 2

Introduction to Optimization

The Three Pillars of Machine Learning

Machine learning rests on three foundational mathematical pillars:

1. Linear Algebra
2. Probability
3. Optimization

Role of Each Pillar

- Linear Algebra: Models structure and relationships in data.
- Probability: Models uncertainty and noise in data.
- Optimization: Converts data and models into decisions.

Optimization is the mathematical tool that enables selection of the best model, classifier, or representation according to a well-defined criterion.

Why Optimization in Machine Learning

In supervised learning, we seek the best classifier.
Let:

- Input data: labeled examples
- Goal: classify new data points
  There are infinitely many possible classifiers. The notion of best must be formalized.

Best as Minimization or Maximization

- Best classifier = one with least loss
- Best policy = one with maximum reward

Thus, machine learning problems often reduce to:

$$\text{minimize loss}$$

or

$$\text{maximize reward}$$

Optimization formalizes these objectives.

Motivating Example: Cow, Rope, Fence, Grass

Setup

Field modeled as a 2D plane.

- Cow at position:

$$(20, 30)$$

- Rope length:

$$10$$

- Fence: vertical line passing through:

$$(25, 0)$$

- Grass at:

$$(40, 40)$$

Objective: Distance to Grass

Let cow position be:

$$x = (x_1, x_2)$$

Distance to grass:

$$d = \sqrt{(x_1 - 40)^2 + (x_2 - 40)^2}$$

We minimize squared distance:

$$(x_1 - 40)^2 + (x_2 - 40)^2$$

Constraints

Rope Constraint

Cow must lie within radius 10 of (20, 30):

$$(x_1 - 20)^2 + (x_2 - 30)^2 \leq 100$$

Fence Constraint

Cow must lie on left side of fence:

$$x_1 \leq 25$$

Complete Optimization Problem

$$\min_{x_1,x_2}(x_1-40)^2+(x_2-40)^2$$

subject to

$$(x_1-20)^2+(x_2-30)^2 \leq 100$$

$$x_1 \leq 25$$

This is a constrained optimization problem.

General Form of Optimization Problem

Let:

$$x \in \mathbb{R}^d$$

Objective

$$\min_{x\in\mathbb{R}^d} f(x)$$

Inequality Constraints

$$g_i(x) \leq 0, \quad i=1,\dots,k$$

Equality Constraints

$$h_j(x) = 0, \quad j=1,\dots,\ell$$

Terminology

Objective Function

Function being minimized or maximized:

$$f(x)$$

Variable or Parameter

Optimization variable:

$$x$$

Inequality Constraints

$$g_i(x) \leq 0$$

Equality Constraints

$$h_j(x) = 0$$

Standardization of Constraints

Any inequality constraint of form:

$$a(x) \leq b$$

can be written as:

$$a(x) - b \leq 0$$

Any equality constraint:

$$a(x) = b$$

can be written as:

$$a(x) - b = 0$$

Thus the general form is without loss of generality.

Maximization vs Minimization

Maximization:

$$\max_x f(x)$$

Equivalent to:

$$\min_x -f(x)$$

Thus all optimization problems can be written in minimization form.

Key Observations

1. Optimization converts structure and uncertainty into decisions.
2. Every machine learning algorithm ultimately solves an optimization problem.
3. Optimization problems consist of:
    - Objective

  - Variables
  - Constraints
4. Both minimization and maximization reduce to minimization.
   Next step: developing algorithms to solve optimization problems.

```
********************************************************************************
```

Lecture 3

Solving an Unconstrained Optimization Problem Part 1

Unconstrained Optimization

General unconstrained problem:

$$\min_{x \in \mathbb{R}} f(x)$$

No inequality or equality constraints.

Simple Example

$$\min_{x \in \mathbb{R}} (x-5)^2$$

Direct Reasoning

- Function is nonnegative.
- Value is zero at $x = 5$.

Thus:

$$x^\star = 5$$

$$f(x^\star) = 0$$

Derivative Based Approach

Let:

$$f(x) = (x-5)^2$$

Compute derivative:

$$f'(x) = 2(x-5)$$

Set derivative to zero:

$$f'(x) = 0$$

$$2(x-5) = 0$$

$$x = 5$$

This works for simple problems, but does not scale well.

Harder Example

$$\min_{x \in \mathbb{R}} 3x^6 + 2x^5 + 3x^3 + 5x^2 + 2$$

Derivative:

$$f'(x) = 18x^5 + 10x^4 + 9x^2 + 10x$$

Setting:

$$f'(x) = 0$$

Leads to solving a degree 5 polynomial, which is not straightforward.
Thus, need a systematic, iterative algorithm.

Iterative Optimization Framework

Start with:

$$x_0 \in \mathbb{R}$$

Iterative update:

$$x_{t+1} = x_t + d$$

where d is a direction to move.
Goal: choose d so that the objective decreases.

Understanding Direction for the Example

Recall:

$$f(x) = (x-5)^2$$

Derivative:

$$f'(x) = 2(x-5)$$

Observations

- If $x > 5$, then $f'(x) > 0$
- If $x < 5$, then $f'(x) < 0$

Desired movement:

- If $x > 5$, move left
- If $x < 5$, move right

Thus:

- If $x > 5$, want $d < 0$
- If $x < 5$, want $d > 0$

Choosing Direction

Since:

- $f'(x) > 0$ when $x > 5$
- $f'(x) < 0$ when $x < 5$

Opposite sign gives desired direction.
Choose:

$$d = -f'(x)$$

Thus update rule becomes:

$$x_{t+1} = x_t - f'(x_t)$$

For this example:

$$x_{t+1} = x_t - 2(x_t - 5)$$

Example Iterations

Let:

$$x_0 = 10$$

Step 1

Compute direction:

$$d_0 = -f'(x_0) = -2(10 - 5) = -10$$

Update:

$$x_1 = 10 - 10 = 0$$

Step 2

Compute direction at $x_1 = 0$:

$$d_1 = -f'(0) = -2(0 - 5) = 10$$

Update:

$$x_2 = 0 + 10 = 10$$

Step 3

$$x_3 = 0$$

Thus sequence oscillates:

$$10 \to 0 \to 10 \to 0 \to \dots$$

Key Observation

The direction is correct, but the step size is too large.
Problem is not:

- Direction of movement

Problem is:

- Magnitude of movement

We overshoot the minimum at $x = 5$.
Findings from this example:

1. Direction should depend on x
2. Negative derivative gives correct direction
3. Step size must be controlled

This motivates introducing a scaling factor in the update rule.

```
************************************************************************************
```

Lecture 4

Solving an Unconstrained Optimization Problem Part 2

Recap: Gradient Based Update

We consider an unconstrained optimization problem:

$$\min_{x \in \mathbb{R}} f(x)$$

The iterative update rule is:

$$x_{t+1} = x_t - \eta_t f'(x_t)$$

Where:

- $f'(x_t)$ determines the direction
- $\eta_t > 0$ is a scalar step size
- $-f'(x_t)$ is the descent direction

The issue observed earlier:

- Direction is correct.
- Step magnitude may be too large.
- This can cause oscillations.

Hence we introduce a step size.

Step Size

Updated rule:

$$x_{t+1} = x_t - \eta_t f'(x_t)$$

Where:

- η_t is positive
- It may depend on iteration t
- It controls how far we move in the descent direction

First Attempt at Step Size

Consider:

$$\eta_0 = 1, \quad \eta_1 = \frac{1}{2}, \quad \eta_2 = \frac{1}{4}, \quad \eta_3 = \frac{1}{8}, \dots$$

General form:

$$\eta_t = \frac{1}{2^t}$$

Properties:

- Step size decreases geometrically.
- Appears reasonable to avoid oscillation.

Cumulative Step Size

Consider total movement if direction is constant:

$$\sum_{t=0}^{\infty} \eta_t = \sum_{t=0}^{\infty} \frac{1}{2^t}$$

This is a geometric series:

$$1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots = 2$$

Thus:

$$\sum_{t=0}^{\infty} \eta_t = 2$$

Observation

- Total movement is bounded.
- Even after infinitely many steps, we can move at most 2 units.
- If optimum is farther away, we will never reach it.
- Hence this step size sequence is not suitable.

Desired Properties of Step Sizes

We need:

1. Step size decreases to avoid oscillations.
2. Cumulative step sizes should not be bounded.

That is:

$$\eta_t \to 0$$

and

$$\sum_{t=0}^{\infty} \eta_t = \infty$$

Improved Step Size Sequence

Consider:

$$\eta_0 = 1, \quad \eta_1 = \frac{1}{2}, \quad \eta_2 = \frac{1}{3}, \quad \eta_3 = \frac{1}{4}, \dots$$

General form:

$$\eta_t = \frac{1}{t+1}$$

Cumulative Sum

$$\sum_{t=0}^{\infty} \eta_t = \sum_{t=0}^{\infty} \frac{1}{t+1} = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots$$

This is the harmonic series.
Key property:

$$\sum_{t=0}^{\infty} \frac{1}{t+1} = \infty$$

Important Observations

- Step size decreases to zero.
- Total cumulative movement is unbounded.
- No matter how far the starting point is from optimum, it is theoretically possible to reach the optimum.
- Avoids oscillation from large constant step sizes.
- Avoids stagnation from overly fast geometric decay.

Final Iterative Algorithm

Given objective:

$$\min_{x \in \mathbb{R}} f(x)$$

Algorithm:

1. Initialize x_0.
2. For $t = 0, 1, 2, \dots, T$:

   $$x_{t+1} = x_t - \eta_t f'(x_t)$$

   with

   $$\eta_t = \frac{1}{t+1}$$

3. Output x_T.

Conceptual Insight

Two conflicting objectives:

- Reduce step size to avoid oscillations.
- Maintain sufficient total movement to reach optimum.

The harmonic step size balances both:

- $\eta_t \to 0$
- $\sum \eta_t = \infty$

This completes the construction of a principled step size schedule for gradient based optimization.

```
***************************************************************************************
```

Lecture 5

Gradient Descent and Local Minima

Unconstrained Optimization Setup

We consider the unconstrained minimization problem

$$\min_{x \in \mathbb{R}} f(x)$$

where

- $f : \mathbb{R} \to \mathbb{R}$
- f is differentiable

Gradient Descent Algorithm

Initialization

Choose an arbitrary starting point

$$x_0 \in \mathbb{R}$$

Iterative Update Rule

For $t = 0, 1, 2, \ldots$

$$x_{t+1} = x_t - \eta_t f'(x_t)$$

where

$$\eta_t = \frac{1}{t+1}$$

Interpretation

- $-f'(x_t)$ is the **descent direction**
- η_t is the **step size**
- The update moves in the direction of steepest decrease

This algorithm is called the **Gradient Descent Algorithm**.
It is a **first-order method** because it uses only first derivative information.

Generalization to Higher Dimensions

For

$$\min_{x \in \mathbb{R}^d} f(x)$$

where

$$f : \mathbb{R}^d \to \mathbb{R}$$

the derivative is replaced by the **gradient**

$$\nabla f(x)$$

Update rule becomes

$$x_{t+1} = x_t - \eta_t \nabla f(x_t)$$

Convergence Property

If

$$\eta_t = \frac{1}{t+1}$$

then the algorithm converges.
More precisely,
Gradient descent converges to a local minimum.

Local vs Global Minimum

Global Minimum

A point $x^\star$ is a global minimum if

$$f(x^\star) \leq f(x) \quad \text{for all } x$$

Local Minimum

A point $\hat{x}$ is a local minimum if
there exists $\epsilon > 0$ such that

$$f(\hat{x}) \leq f(x) \quad \text{for all } x \in (\hat{x} - \epsilon, \hat{x} + \epsilon)$$

Thus minimality holds only in a neighborhood.

Behavior of Gradient Descent

- Converges to a local minimum
- The final solution depends on initialization x_0
- Different starting points may lead to different local minima

---

Why Gradient Descent Stops at Local Minima

At a local minimum $\hat{x}$:

$$f'(\hat{x}) = 0$$

Thus

$$x_{t+1} = x_t - \eta_t f'(x_t)$$

becomes

$$x_{t+1} = x_t$$

The algorithm cannot move further because the descent direction vanishes.

Limitation

Gradient descent does not guarantee convergence to a global minimum for arbitrary functions.
It only guarantees convergence to a local minimum.

Special Case: Convex Functions

For certain functions, every local minimum is also a global minimum.
Formally, for such functions

$$\text{Local minimum} \Rightarrow \text{Global minimum}$$

Example:

$$f(x) = (x-5)^2$$

- Unique minimum at $x = 5$
- Local minimum equals global minimum

For such functions, gradient descent finds the optimal solution.
These functions are called **convex functions**.

Summary

Gradient Descent Algorithm:

1. Initialize x_0
2. Update

$$x_{t+1} = x_t - \eta_t f'(x_t)$$

3. Choose step size

$$\eta_t = \frac{1}{t+1}$$

Properties:

- Uses first-order information
- Converges under suitable step sizes
- Converges to a local minimum
- For convex functions, local minimum equals global minimum

This forms the foundational algorithm for optimization in machine learning.

```
**********************************************************************************
```

Lecture 6

Why Move in the Negative Derivative Direction

Unconstrained Minimization Setup

We consider the unconstrained optimization problem

$$\min_{x \in \mathbb{R}} f(x)$$

The gradient descent update rule is

$$x_{t+1} = x_t + \eta_t \left(-f'(x_t)\right)$$

where

- $\eta_t > 0$ is the step size
- $-f'(x_t)$ is the descent direction
  The central question:
  Why choose the direction $d = -f'(x)$?

Taylor Series Expansion

For a differentiable function f, the Taylor expansion around x gives

$$f(x + \eta d) = f(x) + \eta d f'(x) + \frac{\eta^2 d^2}{2} f''(x) + \cdots$$

Key Observation

All derivatives are evaluated at the point x.
Thus:
Local information at x determines the behavior of the function near x.

First-Order Approximation

If η is small and positive, higher order terms become negligible.
Hence, for sufficiently small η:

$$f(x + \eta d) \approx f(x) + \eta d f'(x)$$

Subtracting $f(x)$:

$$f(x+\eta d) - f(x) \approx \eta d f'(x)$$

Condition for Descent

To decrease the function value, we require

$$f(x+\eta d) - f(x) < 0$$

Using the approximation:

$$\eta d f'(x) < 0$$

Since $\eta > 0$, this reduces to

$$d f'(x) < 0$$

Choosing the Descent Direction

We must choose d such that

$$d f'(x) < 0$$

A natural choice is

$$d = -f'(x)$$

Then:

$$d f'(x) = -f'(x)^2$$

Since

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

we get

$$-f'(x)^2 \leq 0$$

Strictly negative whenever $f'(x) \neq 0$.
Thus, for small enough η:

$$f(x+\eta d) < f(x)$$

This guarantees descent.

Interpretation

- $f'(x)$ gives the slope at x
- Moving in direction $-f'(x)$ moves opposite to the slope
- For small step size, function value decreases
  Hence:
  Negative derivative is a descent direction.

Extension to Higher Dimensions

For $x \in \mathbb{R}^d$:

- Derivative generalizes to gradient $\nabla f(x)$
- Update rule becomes

$$x_{t+1} = x_t$$

- \eta_t
  \nabla f(x_t)

  $Thegradientisthevectrfpartiaderivatives. Derivativeinnedimensincrrespndstgradientinhigherdi$

Core Insight

Using Taylor expansion:

$$f(x + \eta d) \approx f(x) + \eta d f'(x)$$

To decrease the function locally:

$$d = -f'(x)$$

Thus gradient descent moves in the direction of steepest local decrease.
This provides the theoretical justification for the gradient descent update rule.

```
*********************************************************************************
```

Lecture 7

Gradient Descent in Higher Dimensions

Unconstrained Optimization in Higher Dimensions

Problem Setup

We now consider the unconstrained optimization problem

$$\min_{x \in \mathbb{R}^d} f(x)$$

where

$$x = (x_1, x_2, \ldots, x_d)$$

and

$$f : \mathbb{R}^d \to \mathbb{R}$$

is differentiable.
In the one dimensional case, the update rule was

$$x_{t+1} = x_t - \eta_t f'(x_t)$$

We now generalize this to multiple dimensions.

Gradient as Generalization of Derivative

Definition: Partial Derivatives

For a function

$$f(x_1, x_2, \ldots, x_d)$$

the partial derivative with respect to coordinate x_i is

$$\frac{\partial f}{\partial x_i}$$

obtained by treating all other variables as constants.

Definition: Gradient

The gradient of f at point $x \in \mathbb{R}^d$ is defined as

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(x) \\ \frac{\partial f}{\partial x_2}(x) \\ \vdots \\ \frac{\partial f}{\partial x_d}(x) \end{bmatrix}$$

Thus

- $\nabla f(x)$ is a vector
- Each coordinate captures rate of change along one axis

The derivative in one dimension becomes the gradient vector in higher dimensions.

Example 1

Consider

$$f(x_1, x_2) = x_1^2 + 4x_2 + 8x_2^2$$

Step 1: Compute Gradient

$$\nabla f(x_1, x_2) = \begin{bmatrix} 2x_1 \\ 4 + 16x_2 \end{bmatrix}$$

Step 2: Evaluate at a Point

At $(1, 3)$:

$$\nabla f(1,3) = \begin{bmatrix} 2 \\ 4 + 16 \cdot 3 \end{bmatrix} = \begin{bmatrix} 2 \\ 52 \end{bmatrix}$$

This is a direction vector in $\mathbb{R}^2$.

Example 2: Distance Function

Let grass be located at $(40, 40)$ and define squared distance

$$d(x_1, x_2) = (x_1 - 40)^2 + (x_2 - 40)^2$$

Gradient

$$\nabla d(x_1, x_2) = \begin{bmatrix} 2(x_1 - 40) \\ 2(x_2 - 40) \end{bmatrix}$$

Evaluate at $(5, 2)$

$$\nabla d(5,2) = \begin{bmatrix} 2(5-40) \\ 2(2-40) \end{bmatrix} = \begin{bmatrix} -70 \\ -76 \end{bmatrix}$$

Thus negative gradient is

$$-\nabla d(5,2) = \begin{bmatrix} 70 \\ 76 \end{bmatrix}$$

This direction points toward $(40, 40)$.

Evaluate at $(30, 50)$

$$\nabla d(30,50) = \begin{bmatrix} 2(30-40) \\ 2(50-40) \end{bmatrix} = \begin{bmatrix} -20 \\ 20 \end{bmatrix}$$

Negative gradient:

$$-\nabla d(30,50) = \begin{bmatrix} 20 \\ -20 \end{bmatrix}$$

Again, this direction moves toward $(40, 40)$.

Multivariate Taylor Expansion

To justify the negative gradient direction, consider first order Taylor expansion.
For small η and direction d:

$$f(x + \eta d) = f(x) + \eta \nabla f(x)^T d + O(\eta^2)$$

For sufficiently small η, higher order terms are negligible:

$$f(x + \eta d) \approx f(x) + \eta \nabla f(x)^T d$$

Descent Condition

We want function value to decrease:

$$f(x + \eta d) - f(x) < 0$$

Using approximation:

$$\eta \nabla f(x)^T d < 0$$

Since $\eta > 0$, this reduces to

$$\nabla f(x)^T d < 0$$

Thus any direction d satisfying this gives descent.

Choosing the Steepest Descent Direction

Let

$$d = -\nabla f(x)$$

Then

$$\nabla f(x)^T d = \nabla f(x)^T (-\nabla f(x)) = -\|\nabla f(x)\|^2$$

Since norm squared is nonnegative,

$$-\|\nabla f(x)\|^2 \leq 0$$

Strictly negative unless gradient is zero.
Thus negative gradient guarantees descent for sufficiently small step size.

Gradient Descent Algorithm in $\mathbb{R}^d$

Initialization

Choose arbitrary

$$x_0 \in \mathbb{R}^d$$

Iterative Update

$$x_{t+1} = x_t - \eta_t \nabla f(x_t)$$

where

- x_t is a vector
- $\nabla f(x_t)$ is a vector
- η_t is a positive scalar step size

Important Observations

1. Gradient generalizes derivative to multiple dimensions.
2. Negative gradient gives direction of maximum local decrease.
3. Taylor expansion justifies the descent property.
4. With appropriate step sizes, gradient descent converges to a local minimum.

Convergence Property

For differentiable f:
Gradient descent converges to a local minimum under suitable step size conditions.
If f is convex, then:

$$\text{local minimum} \implies \text{global minimum}$$

Thus for convex functions, gradient descent finds the global optimum.

Final Algorithm Summary

Given differentiable $f : \mathbb{R}^d \to \mathbb{R}$:

$$x_{t+1} = x_t - \eta_t \nabla f(x_t)$$

This is the Gradient Descent algorithm for unconstrained optimization.

```
********************************************************************************
```

Lecture 8

Taylor Series in Higher Dimensions and Geometry of Gradient Descent

Multivariate Taylor Series

Let $f : \mathbb{R}^d \to \mathbb{R}$ and let $x \in \mathbb{R}^d$.
We want to evaluate the function at a nearby point:

$$x + \eta d$$

where

- $\eta > 0$ is a small scalar step size
- $d \in \mathbb{R}^d$ is a direction vector

The multivariate Taylor expansion gives:

$$f(x + \eta d) = f(x) + \eta d^T \nabla f(x) + \text{higher order terms}$$

Neglecting higher order terms for small η:

$$f(x + \eta d) \approx f(x) + \eta d^T \nabla f(x)$$

Thus,

$$f(x + \eta d) - f(x) \approx \eta d^T \nabla f(x)$$

Descent Condition

We want the function value to decrease:

$$f(x + \eta d) - f(x) < 0$$

Using the Taylor approximation:

$$\eta d^T \nabla f(x) < 0$$

Since $\eta > 0$, this is equivalent to:

$$d^T \nabla f(x) < 0$$

Key Condition

A direction d is a descent direction if

$$d^T \nabla f(x) < 0$$

Special Choice: Negative Gradient

Choose

$$d = -\nabla f(x)$$

Then

$$d^T \nabla f(x) = -\nabla f(x)^T \nabla f(x) = -\|\nabla f(x)\|^2$$

Since

$$\|\nabla f(x)\|^2 = \sum_{i=1}^{d} \left(\frac{\partial f}{\partial x_i} \right)^2 \geq 0$$

we get

$$d^T \nabla f(x) \leq 0$$

Strictly negative whenever $\nabla f(x) \neq 0$.
Therefore,

$$d = -\nabla f(x)$$

is always a descent direction.

Geometry of Descent Directions

Let $w = \nabla f(x)$.
Consider the dot product condition:

$$d^T w \begin{cases} > 0 & \text{ascent direction} \\ = 0 & \text{orthogonal direction} \\ < 0 & \text{descent direction} \end{cases}$$

The vector w partitions the space into:

- A half-space where $d^T w > 0$
- A hyperplane where $d^T w = 0$
- A half-space where $d^T w < 0$

All vectors in the half-space satisfying

$$d^T \nabla f(x) < 0$$

are descent directions.
Thus, there are infinitely many descent directions.

Steepest Descent Property

Among all directions with fixed norm,

$$\|d\| = 1$$

the direction that minimizes

$$d^T \nabla f(x)$$

is

$$d = -\frac{\nabla f(x)}{\|\nabla f(x)\|}$$

Thus,

$$-\nabla f(x)$$

gives the maximum rate of decrease.
Hence gradient descent is also called:

Steepest Descent Algorithm

Gradient Descent Update in $\mathbb{R}^d$

For unconstrained optimization:

$$\min_{x \in \mathbb{R}^d} f(x)$$

the iterative update rule is:

$$x_{t+1} = x_t - \eta_t \nabla f(x_t)$$

where

- $x_t \in \mathbb{R}^d$
- $\nabla f(x_t) \in \mathbb{R}^d$
- $\eta_t > 0$ is step size

Important Observations

1. $-\nabla f(x)$ is always a descent direction.
2. It provides the steepest decrease for unit step.
3. Gradient descent converges to a local minimum.
4. In convex functions, local minimum equals global minimum.

Transition to Constrained Optimization

For constrained problems:

$$\min f(x)$$

subject to

$$g(x) \leq 0$$

moving in the negative gradient direction may violate feasibility.
This motivates development of constrained optimization techniques.

```
****************************************************************************************
```
