Lecture 5: Probability II

谢丹
清华大学数学系

October 29, 2025

CHAPTER 6: Graphic probability model

Section 1: Graphic probability model

1. The probability models we have used are predominantly **IID** — observations are assumed to be independent of one another.
2. They are known as **supervised models** as the data is separated as (X, y) with X the feature, and y the target. The goal is to predict the target from new feature.

However, many real-world applications **violate** this independence assumption — for instance, in domains such as:

- Natural language
- Audio signals
- Other sequential or structured data

In such cases, observations exhibit **dependencies**, making the IID probability models less suitable.

Beyond Supervised Learning

Unsupervised Learning Scenarios

We often encounter situations with **unlabeled data**—that is, we only have feature vectors $\mathbf{X}$. The goal in these cases is typically to:

- Discover inherent structures (**Clustering**)
- Learn the underlying data distribution (**Generative Modeling**)

The Modeling Challenge

Modeling such complex, dependent data with general probability models is often intractable.

A Powerful Alternative: Graphical Models

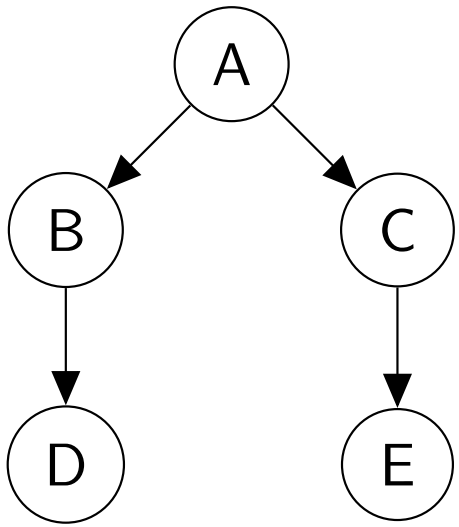
A promising approach uses **graphical models**, which encode conditional dependencies via graph structures. This framework is especially powerful for **Bayesian model learning**.

What are Oriented Graphs in Probability?

- Oriented graphs (directed graphs) represent relationships between random variables
- Nodes represent random variables
- Edges represent conditional dependencies
- Provide compact representation of complex probability distributions
- Enable efficient inference and learning algorithms

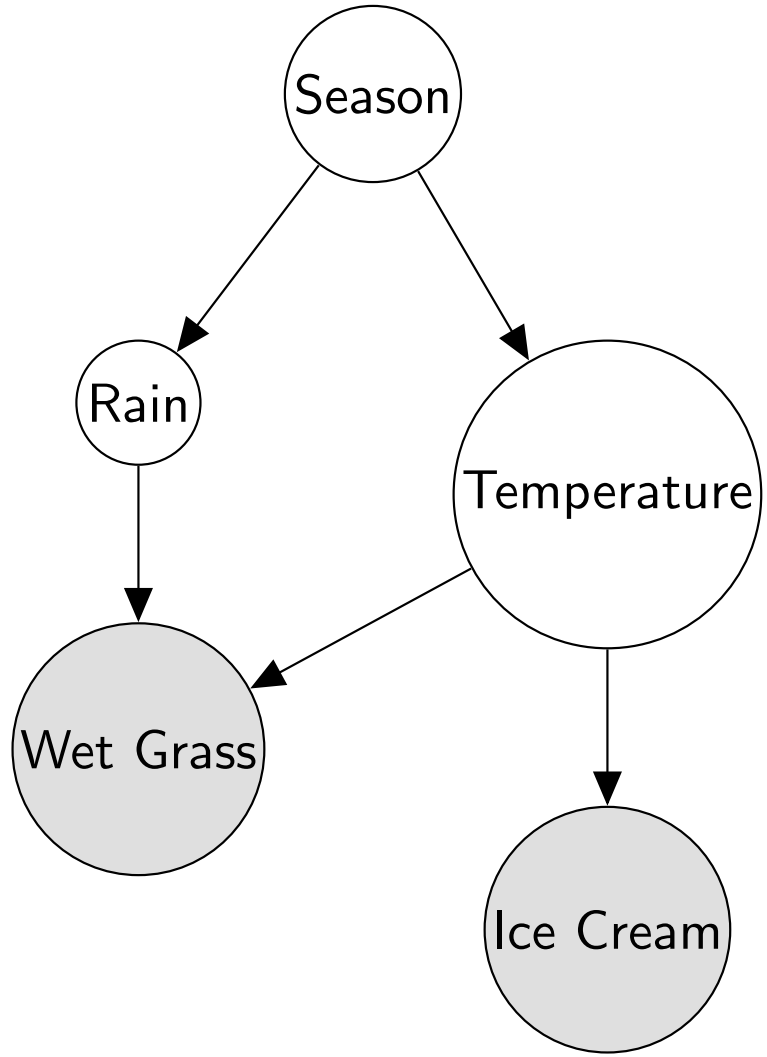
Graph Terminology

- **Nodes/Vertices**: Random variables
- **Edges/Arcs**: Conditional dependencies
- **Parents**: Direct predecessors
- **Children**: Direct successors
- **Ancestors**: All predecessors
- **Descendants**: All successors



Bayesian Networks (Directed Graphical Models)

- Directed acyclic graphs (DAGs)
- Represent joint probability distributions
- Factorization: $P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | \text{parents}(X_i))$
- Encode conditional independence relationships
- Used for reasoning under uncertainty



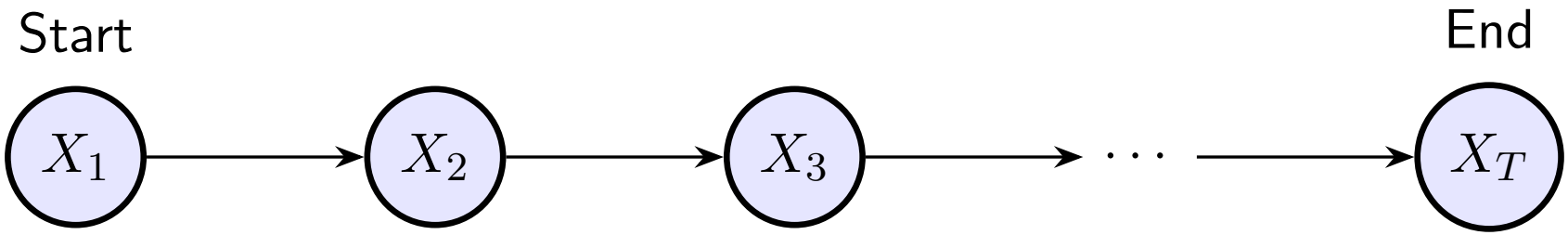
Example 1: Markov Chain

Representing the Markov Property

The Markov Property

$$P(X_t \mid X_{t-1}, X_{t-2}, \ldots, X_1) = P(X_t \mid X_{t-1})$$

"The future is independent of the past, given the present."



- The **nodes** represent the state of the system at different times.
- The **edges** represent transition probabilities between states.
- The graph structure visually encodes the conditional independence relationships of the Markov Property.

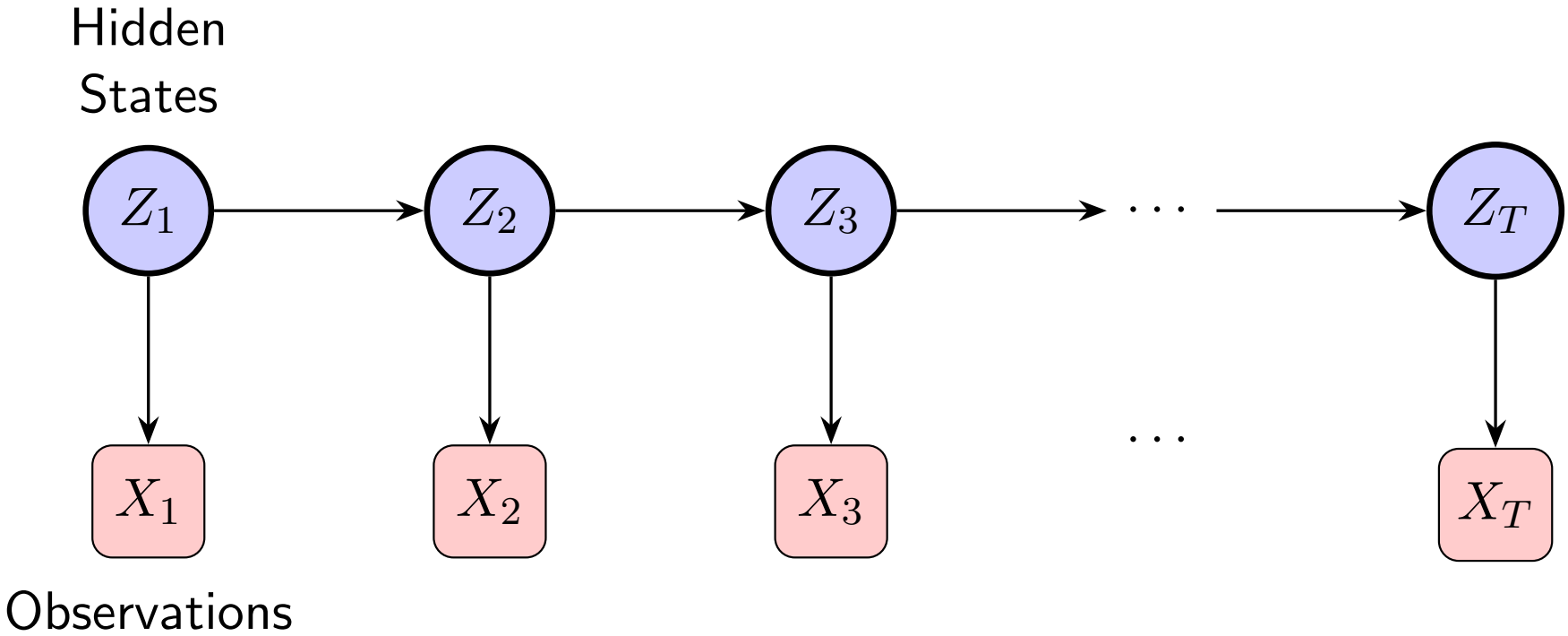
The joint probability from the graph for the Markov chain is

$$P(x_1)P(x_1|x_2)P(x_2|x_3)\ldots P(x_T|x_{T-1})$$

The Markov chain plays an important role later.

Example 2: Hidden Markov Model

A Markov Chain with Observable Outputs



- The hidden states $\{Z_1, Z_2, ..., Z_T\}$ form a **Markov chain**.
- The observations $\{X_1, X_2, ..., X_T\}$ are **independent** given the hidden states.
- $P(X_t|Z_t)$ is the **emission probability**.

The joint probability is

$$P(z_1)P(x_1|z_1)P(z_2|z_1)\dots$$

The HMM is useful for modeling languages.

An application: Graphs Encode Independence

Probabilistic Graphical Models (PGMs) represent complex joint distributions visually.

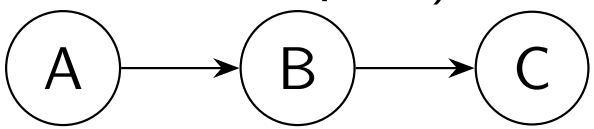
Key Benefit

The graph structure allows us to **read off** conditional independence relationships **without doing any probability calculations**.

The tool for doing this in directed graphs (Bayesian networks) is called **d-separation**.

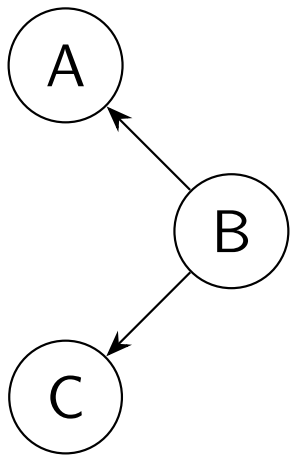
The Three Fundamental Structures

Chain
(Head-to-Tail: oriented path)



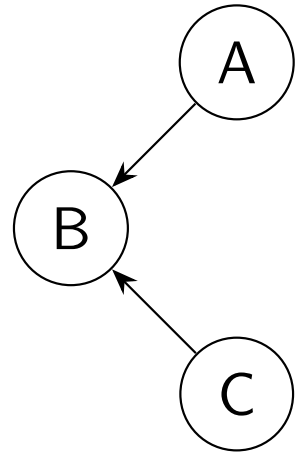
$A \to B \to C$

Fork
(Tail-to-Tail: source)



$A \leftarrow B \to C$

Collider
(Head-to-Head: sink)



$A \to B \leftarrow C$

When is a Path Blocked?

A path is **blocked** by a conditioning set $\mathbf{Z}$ if it contains a node where:

Rule 1: Chain or Fork

The node is **in $\mathbf{Z}$** and this node gives either an oriented path or is a source .

Rule 2: Collider

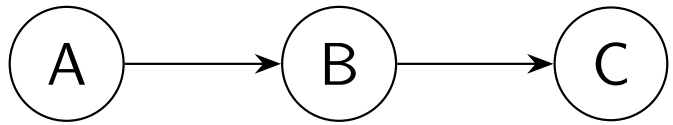
The node is a **sink** and is **NOT in $\mathbf{Z}$**, and **none of its descendants** are in $\mathbf{Z}$.

The d-Separation Algorithm

To check if $(X \perp\!\!\!\perp Y|Z)$ (the conditional independence):

1. Find **all undirected paths** between node X and node Y.
2. For **each path**, check if it is **blocked** by Z (using the rules).
3. If **all paths** are blocked $\Rightarrow$ **d-separated** $\Rightarrow$ **Conditionally Independent** $(X \perp\!\!\!\perp Y|Z)$.
4. If **even one path** is unblocked $\Rightarrow$ **not d-separated** $\Rightarrow$ **Not independent**.

Example 1: Simple Chain

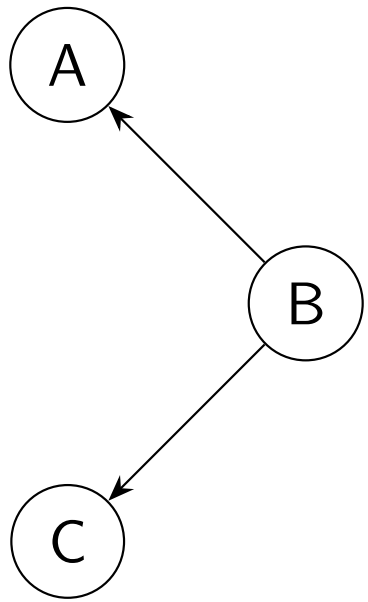


- Is $(A \perp\!\!\!\perp C)$? No. Path is unblocked (no colliders, not conditioning on anything).
- Is $(A \perp\!\!\!\perp C|B)$? Yes! Path $A \to B \to C$ is blocked by B (Rule 1).

$$p(a, b, c) = p(a)p(b|a)p(c|b)$$

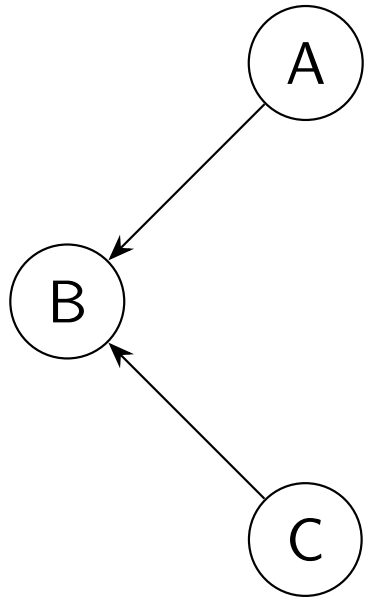
$$p(a, c|b) = \frac{p(a, b, c)}{p(b)} = \frac{p(a)p(b|a)p(c|b)}{p(b)} = p(a|b)p(c|b)$$

Example 2: Common Cause (Fork)



- Is $(A \perp\!\!\!\perp C)$? No. Common cause B creates dependence.
- Is $(A \perp\!\!\!\perp C|B)$? Yes! Path $A \leftarrow B \rightarrow C$ is blocked by conditioning on B (Rule 1).

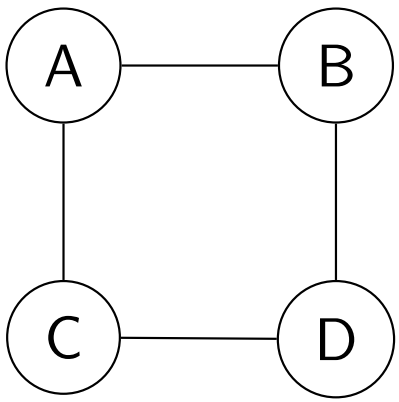
Example 3: Collider (V-structure)



- Is $(A \perp\!\!\!\perp C)$? Yes! The collider B is not conditioned on, so it blocks the path (Rule 2).
- Is $(A \perp\!\!\!\perp C|B)$? No! Conditioning on the collider B *unblocks* the path ("explaining away").

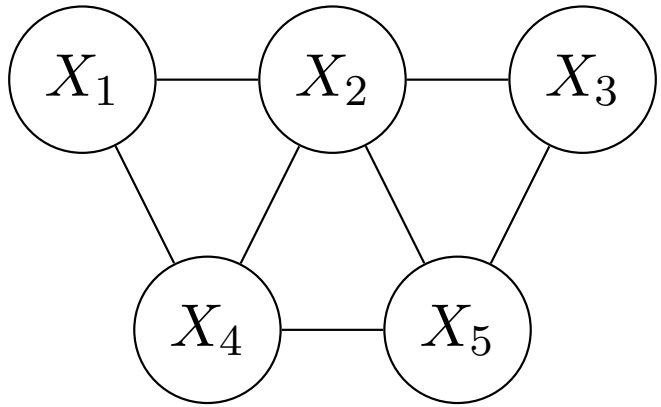
Markov Random Fields (Undirected Graphical Models)

- Undirected graphs
- Represent dependencies without directionality
- Factorization using potential functions
- $P(X) = \frac{1}{Z} \prod_{c \in C} \phi_c(X_c)$
- Z: partition function (normalization)
- Useful for spatial and relational data



Unoriented Graph Probability Models I

Graph Components



Key Features

- **Symmetric dependencies**: No causal direction implied
- **Markov property**: Conditional independence given neighbors
- **Energy-based**: Often use potential functions $\psi_C(X_C)$

This leads us to consider a graphical concept called a clique, which is defined as a subset of the nodes in a graph such that there exists a link between all pairs of nodes in the subset. (Fully connected)

Mathematical Formulation

Joint Probability Distribution

$$P(X) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(X_C)$$

- $\mathcal{C}$: Maximal cliques in the graph
- $\psi_C(X_C) = \exp(-E(X_C))$: Potential function for clique C
- Z: Partition function (normalization constant)

Markov Properties

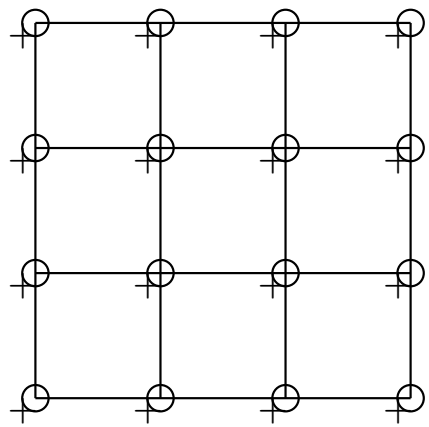
- **Local**: $X_i \perp X_{V \setminus N(i)} \mid X_{N(i)}$
- **Global**: Separation in graph implies conditional independence
- **Pairwise**: $X_i \perp X_j \mid X_{V \setminus \{i,j\}}$ if no edge (i, j)

Ising Model: Graph Representation I

Graph Structure

- **Nodes**: Represent spins/particles $\sigma_i \in \{-1, +1\}$
- **Edges**: Represent interactions between neighboring spins
- **Unoriented**: Interactions are symmetric ($J_{ij} = J_{ji}$)

Example: 2D Lattice



Ising Model: Probability Formulation I

Energy-Based Model

- **Hamiltonian** (Energy function):

$$H(\sigma) = -\sum_{(i,j)\in E} J_{ij}\sigma_i\sigma_j - \mu\sum_i h_i\sigma_i$$

- **Interaction term**: $-J_{ij}\sigma_i\sigma_j$ favors alignment $(J_{ij} > 0)$
- **External field**: $-h_i\sigma_i$ favors specific orientation

Gibbs/Boltzmann Distribution

Ising Model: Probability Formulation II

$$P(\sigma) = \frac{1}{Z} \exp\left(-\beta H(\sigma)\right)$$

- $\beta = 1/kT$: Inverse temperature
- $Z = \sum_{\sigma} \exp(-\beta H(\sigma))$: Partition function
- Low-energy states have higher probability

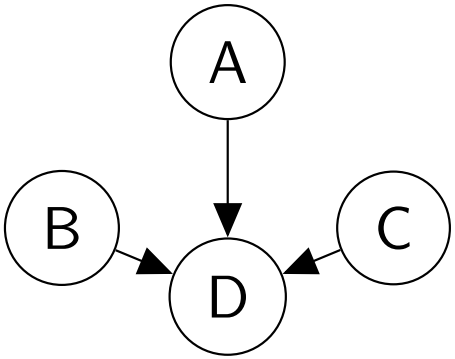
Key Insight

- Models **collective behavior** from local interactions
- Exhibits **phase transitions** at critical temperature
- Foundation for **Markov Random Fields** in machine learning

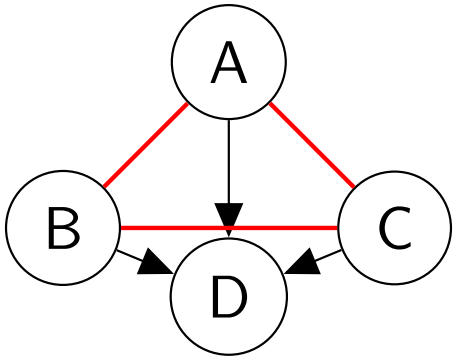
Converting Directed → Undirected Graphical Models

The Moralization Algorithm

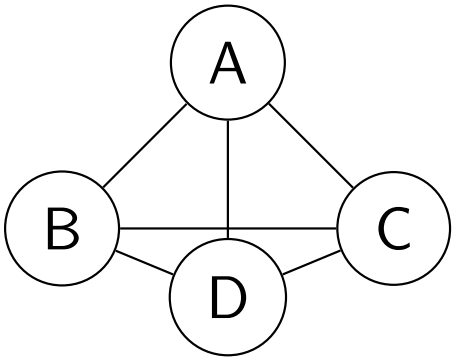
Bayesian Network



Step 1: Marry Parents



Markov Random Field



Algorithm Steps

1. **Marry Parents:** Connect all parents of each node
2. **Drop Directions:** Convert all edges to undirected
3. **Define Potentials:** $\psi(\text{clique}) = P(\text{node}|\text{parents})$

Key Insight

Moralization preserves dependencies but may lose some conditional independencies!

Inference in Graphical Models

We would like to compute various conditional probability.

- **Exact inference**: Variable elimination, belief propagation
- **Approximate inference**: Sampling methods, variational inference

We will discuss them soon.

Section 2: Gaussian mixture model

Clustering: K-means

Before we discuss the general Gaussian mixture models as our first non-iid models, let's discuss an unsupervised learning model called K-means:

- Unsupervised clustering algorithm
- Partitions data into K clusters
- Iterative optimization approach
- Widely used for data exploration and pattern discovery

Problem Formulation

Goal

Given a dataset $X = \{x_1, x_2, ..., x_n\}$ and integer K, partition the data into K clusters where:

- Points in same cluster are similar
- Points in different clusters are dissimilar

Objective Function

Minimize the within-cluster sum of squares:

$$J = \sum_{i=1}^{K} \sum_{x \in C_i} \|x - \mu_i\|^2$$

where μ_i is the centroid of cluster C_i

K-means Algorithm

1: **procedure** $\text{K-MEANS}(X, K)$
2: Initialize K centroids $\mu_1, \mu_2, ..., \mu_K$ randomly
3: **repeat**
4: **Assignment step:** For each point x_j, assign to nearest centroid:

$$c_j = \arg\min_i \|x_j - \mu_i\|^2$$

5: **Update step:** Recompute centroids:

$$\mu_i = \frac{1}{|C_i|} \sum_{x_j \in C_i} x_j$$

6: **until** centroids converge or maximum iterations reached
7: **return** cluster assignments and centroids
8: **end procedure**

Initialization Methods

Random Initialization

- Simple but can lead to poor results
- Multiple runs recommended
- Sensitive to initial choice

K-means++

- Smart initialization technique
- Spreads initial centroids apart
- Leads to better and faster convergence

k-means++ Initialization

1. **First Centroid:**
   - Choose one random data point as first centroid
2. **Subsequent Centroids:**
   - Select each next centroid with probability proportional to:

     Distance to nearest existing centroid
   - Favors points far from current centroids
   - Ensures better spread of initial centers
3. **Greedy k-means++ (Scikit-learn):**
   - Multiple trials at each step
   - Choose candidate that reduces inertia the most
   - Further improves initialization quality

Key Advantage

Better initial centroids $\rightarrow$ Faster convergence + Better clustering results

Choosing the Right K

Elbow Method

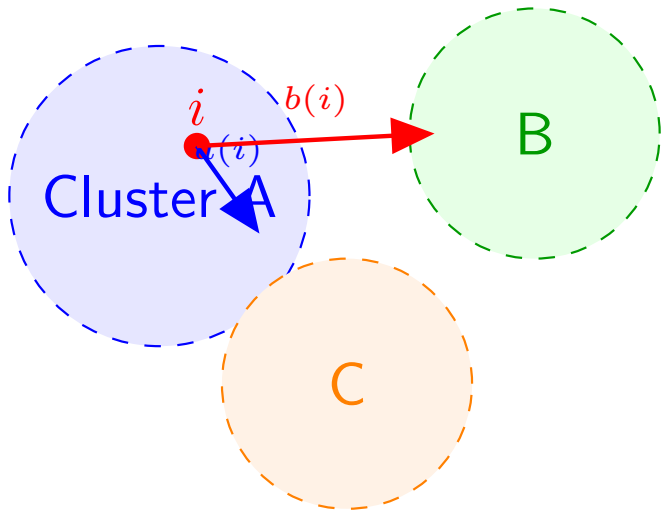
- Plot $J(K)$ (within-cluster sum of squares for trained model) vs K
- Look for "elbow" point where improvement slows
- Subjective but practical

Silhouette Analysis

- Measures how similar points are to their own cluster vs other clusters
- Range: -1 (poor) to $+1$ (excellent)

Silhouette Score: Choosing K in K-means I

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$



Definitions:

- $a(i)$: Avg distance to points in **same** cluster
- $b(i)$: Avg distance to points in **nearest other** cluster

Interpretation:

- $s(i) \approx +1$: Well clustered
- $s(i) \approx 0$: On boundary
- $s(i) \approx -1$: Misclassified

Silhouette Score: Choosing K in K-means II

How to choose K

Compute mean silhouette score for each K: $\frac{1}{n}\sum s(i)$

Choose K with highest average score!

Hard K-Means Limitations

Problems:

- Each point belongs to exactly one cluster
- Sensitive to outliers
- Poor handling of overlapping clusters
- Binary membership

Soft K-Means Concept

Key Idea

Each data point can belong to multiple clusters with varying degrees of membership.

- **Membership weights**: 0 to 1 for each cluster
- **Fuzzy assignments**: Partial cluster membership
- **Probabilistic approach**: More flexible clustering

Mathematical Formulation

Membership Function

$$w_{ij} = \frac{1}{\sum_{k=1}^{K} \left(\frac{d_{ij}}{d_{ik}} \right)^{\frac{2}{m-1}}}$$

Cluster Center Update

$$c_j = \frac{\sum_{i=1}^{N} w_{ij}^m \cdot x_i}{\sum_{i=1}^{N} w_{ij}^m}$$

Where:

- w_{ij}: Membership of point i in cluster j
- d_{ij}: Distance from point i to cluster j
- m: Fuzziness parameter ($m > 1$)
- c_j: Center of cluster j

Algorithm Steps

1. **Initialize** cluster centers randomly
2. **Calculate membership weights** for all points
3. **Update cluster centers** using weighted averages
4. **Repeat** until convergence
   - Maximum iterations reached, or
   - Cluster centers change minimally

Convergence Criterion

$$\|C^{(t)} - C^{(t-1)}\| < \epsilon$$

Fuzziness Parameter (m)

- Controls the degree of fuzziness
- $m = 1$: Hard K-means
- $m \to \infty$: Equal membership
- Typical range: $1.5 \leq m \leq 3.0$

Advantages of Soft K-Means

Benefits

- **Handles overlapping clusters** better
- **More robust** to noise and outliers
- Provides **membership probabilities**
- Better for **uncertain data**
- More **natural** for many applications

Comparison: Hard vs Soft K-Means

Feature	**Hard K-Means**	**Soft K-Means**
Membership	Binary (0 or 1)	Continuous (0 to 1)
Outlier sensitivity	High	Low
Overlapping clusters	Poor	Good
Complexity	Lower	Higher
Interpretation	Clear boundaries	Fuzzy boundaries

Table: Comparison of clustering approaches

Gaussian Mixture Models

Problem Setting

Given unlabeled data, we aim to discover natural groupings through probabilistic modeling. (unlike K-means algorithm)

Gaussian Mixture Model Definition

A GMM represents the data distribution as a weighted sum of K Gaussian components:

$$P(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- **Mixing coefficients**: π_k with $\sum_{k=1}^{K} \pi_k = 1$, $\pi_k \geq 0$
- **Component parameters**: $\boldsymbol{\mu}_k$ (mean), $\boldsymbol{\Sigma}_k$ (covariance)
- Each Gaussian $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ represents one cluster

Parameter Estimation Challenge I

Maximum Likelihood Estimation

We seek parameters that maximize the log-likelihood:

$$\mathcal{L}(\theta) = \sum_{n=1}^{N} \log \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right)$$

Optimization Difficulty

- **No closed-form solution** due to summation inside logarithm
- **Non-convex optimization** with multiple local maxima
- **Singularity issues**: Likelihood unbounded if $\boldsymbol{\Sigma}_k$ collapses to a point

Solution: Expectation-Maximization (EM) Algorithm

Parameter Estimation Challenge II

- Iterative procedure that guarantees likelihood improvement
- Handles missing data (latent variables) naturally
- Provides soft assignments (responsibilities)

It has solid mathematical foundation which will be discussed later.

Latent Variable Representation I

Hidden Variable $\mathbf{z}$

To facilitate inference, we introduce a latent categorical variable $\mathbf{z}$:

- $\mathbf{z} = [z_1, z_2, \ldots, z_K]^T$ with $z_k \in \{0, 1\}$ and $\sum_k z_k = 1$
- $P(z_k = 1) = \pi_k$ (probability that data point belongs to component k)
- $P(\mathbf{x}|z_k = 1) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$

Complete Data Likelihood

The joint distribution factorizes as:

$$P(\mathbf{x}, \mathbf{z}) = P(\mathbf{z})P(\mathbf{x}|\mathbf{z}) = \prod_{k=1}^{K} \pi_k^{z_k} \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_k}$$

Latent Variable Representation II

Learning Objective

Estimate parameters $\{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$ that maximize the likelihood of observed data.

Figure: The graph model by treating z and x as random variables.

There is a general method called EM method to evaluate the maximal likely-hood for the latent model.

EM Algorithm for GMM I

E-step: Compute Responsibilities

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

$\gamma(z_{nk})$ = probability that data point n belongs to component k

M-step: Update Parameters

EM Algorithm for GMM II

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk})$$

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) (\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}})(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}})^T$$

$$\pi_k^{\text{new}} = \frac{N_k}{N}$$

The EM Algorithm in Code

1. Initialize parameters $\boldsymbol{\theta}^{\text{old}} = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$ for $k = 1..K$
2. **While** not converged: **do**
3. **E-Step:** Compute $\gamma(z_{nk})$ for all n, k
4. **M-Step:** Recompute $\boldsymbol{\theta}^{\text{new}}$ using all $\gamma(z_{nk})$
5. $\boldsymbol{\theta}^{\text{old}} \leftarrow \boldsymbol{\theta}^{\text{new}}$
6. **End While**

Convergence: Check if change in log-likelihood $|\ln p(\mathbf{X}|\boldsymbol{\theta}^{\text{new}}) - \ln p(\mathbf{X}|\boldsymbol{\theta}^{\text{old}})| < \text{tol}$.

From GMM to K-means: The Constraints

A full GMM has parameters: $\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$ **To get K-means, apply these constraints:**

1. Covariance matrices are spherical and shared: $\boldsymbol{\Sigma}_k = \epsilon \mathbf{I}$ for all k.
2. Mix coefficients are equal: $\pi_k = 1/K$.

Now, take the limit $\epsilon \to 0$:

- The Gaussian distributions become infinitely narrow.
- The probability for the closest component becomes 1 (hard assignment).
- The GMM's EM algorithm reduces exactly to K-means.