CHAPTER 9

Unsupervised Learning Techniques

Most real-world data is **unlabeled**, even though supervised learning dominates practical Machine Learning applications. Unsupervised learning focuses on extracting structure, patterns, and information from such unlabeled data. As Yann LeCun famously noted, unsupervised learning forms the core of intelligence, while supervised learning and reinforcement learning build on top of it. This chapter explores several key unsupervised learning tasks—**clustering, anomaly detection, and density estimation**—and introduces widely used algorithms that address these problems.

Clustering

Clustering is the task of grouping similar instances into clusters without knowing any labels in advance. Unlike classification, where the target classes are predefined, clustering discovers structure inherent in the data. Humans naturally perform clustering when recognizing similar objects, such as grouping plants by appearance, even without knowing their species.

Clustering algorithms are widely used in customer segmentation, recommender systems, data analysis, dimensionality reduction, anomaly detection, semi-supervised learning, search engines, and image segmentation. Importantly, there is no single definition of a “cluster”: some algorithms search for spherical groups around centroids, others detect dense regions of arbitrary shapes, and some build hierarchical relationships between clusters.

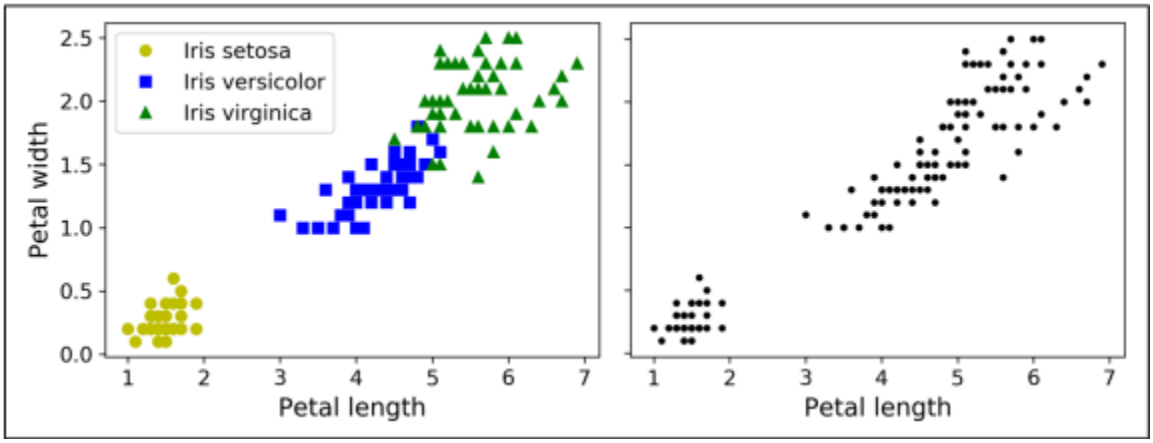

Figure 9-1. Classification (left) versus clustering (right)

K-Means

K-Means is one of the simplest and fastest clustering algorithms. It works best when clusters are compact, roughly spherical, and of similar size. The algorithm requires the number of clusters $k$to be specified in advance and alternates between assigning each instance to the nearest centroid and updating centroids as the mean of their assigned instances. This process is guaranteed to converge, usually in a small number of iterations.

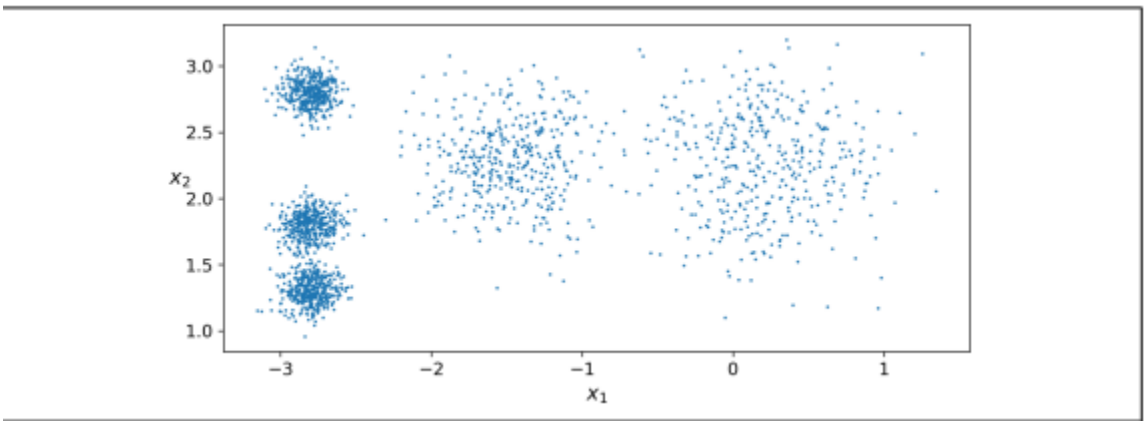

Figure 9-2. An unlabeled dataset composed of five blobs of instances

Once trained, K-Means assigns each instance a cluster label (not to be confused with class labels in supervised learning) and stores the cluster centroids. The resulting decision boundaries form a **Voronoi tessellation**, where each region corresponds to the set of points closest to a centroid.

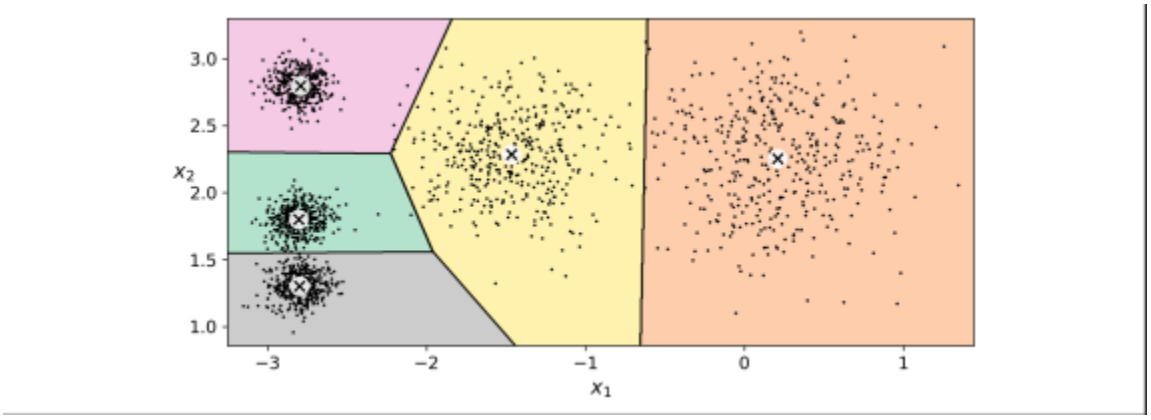

Figure 9-3. K-Means decision boundaries (Voronoi tessellation)

K-Means can also be used for **soft clustering** by measuring distances from instances to all centroids. These distances can serve as new features, effectively transforming an n-dimensional dataset into a k-dimensional representation, which can act as a nonlinear dimensionality reduction technique.

How K-Means Works

The algorithm starts by initializing centroids (often randomly), then repeatedly assigns instances to the closest centroid and recomputes centroids until they stabilize. Although convergence is guaranteed, K-Means may converge to **suboptimal local minima** depending on the initialization.

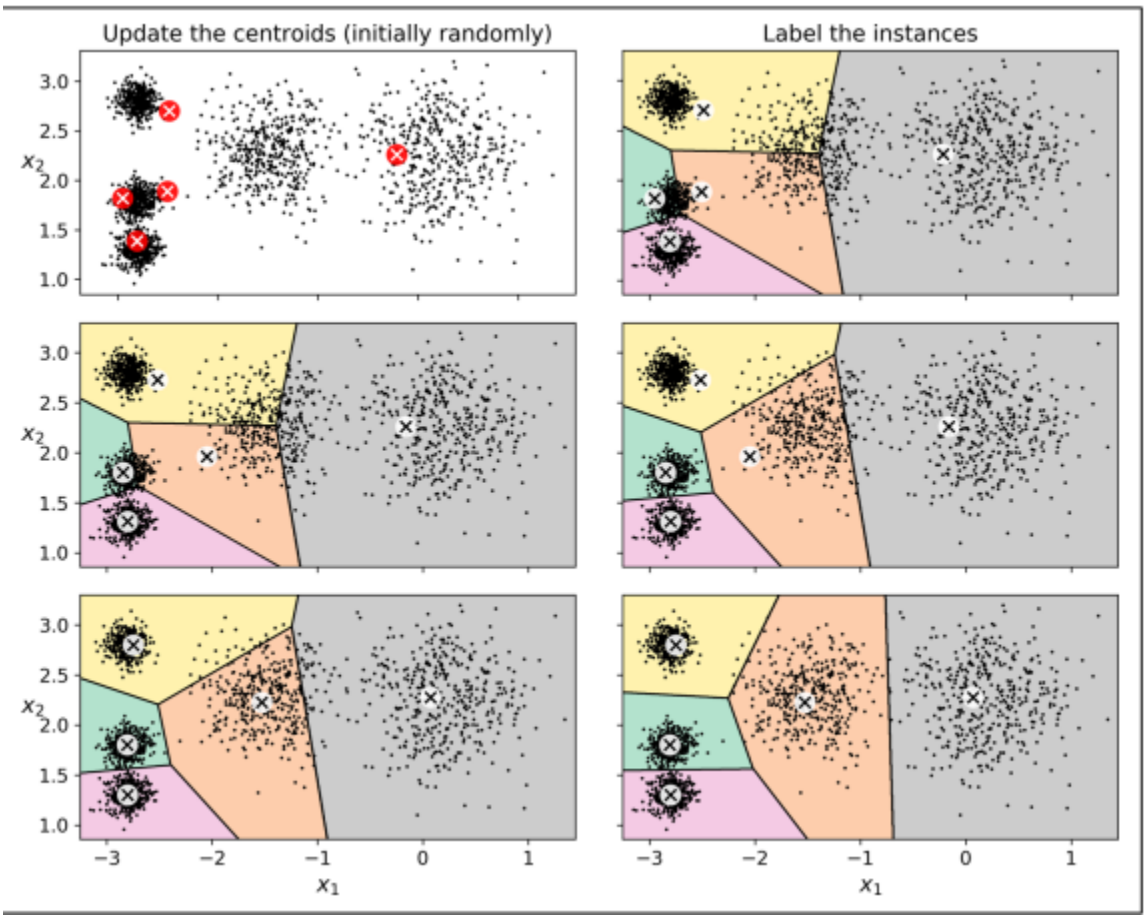

Figure 9-4. The K-Means algorithm

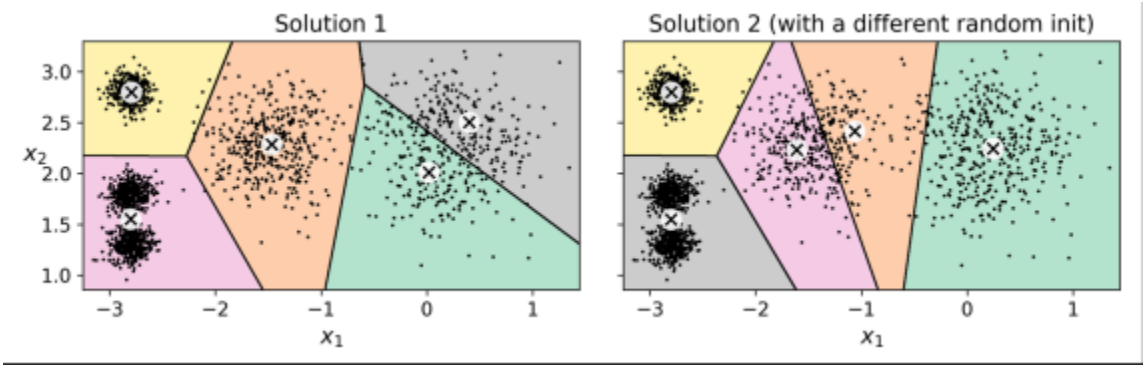

Figure 9-5. Suboptimal solutions due to unlucky centroid initializations

To mitigate this, K-Means is typically run multiple times with different initializations, selecting the solution with the lowest **inertia**, defined as the mean squared distance between instances and their closest centroid. Scikit-Learn automates this using the n_init parameter.

A major improvement, **K-Means++**, initializes centroids more carefully by favoring points far from existing centroids. This significantly reduces the risk of poor solutions and is the default initialization strategy in Scikit-Learn.

Accelerated and Mini-Batch K-Means

Several optimizations make K-Means more scalable. Elkan's algorithm accelerates K-Means by reducing distance computations using the triangle inequality. **Mini-Batch K-Means** further improves scalability by updating centroids using small random subsets of the data, making it suitable for very large datasets. While Mini-Batch K-Means is much faster, it typically yields slightly higher inertia than full K-Means.

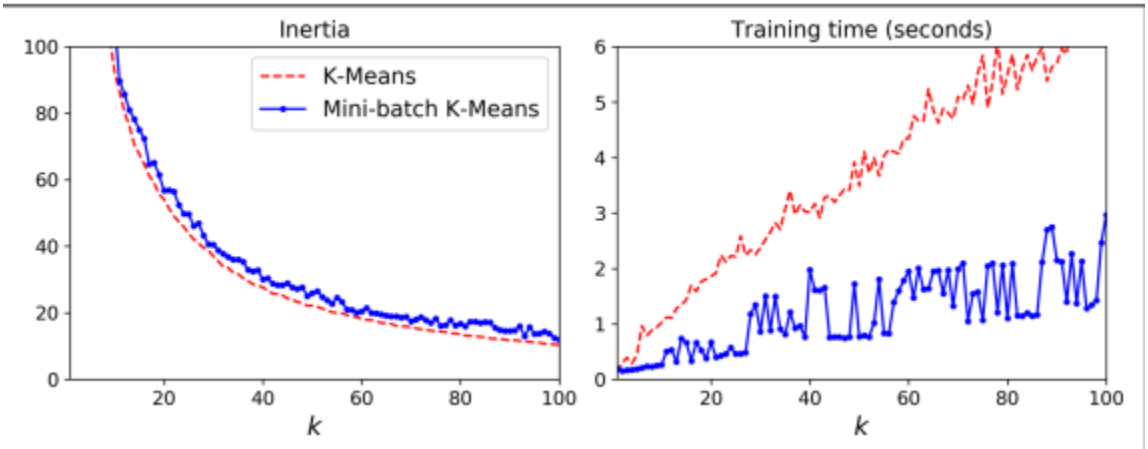

Figure 9-6. Mini-batch K-Means has higher inertia but is much faster

---

Choosing the Number of Clusters

Selecting the right value of *k* is critical. Inertia alone is not sufficient, as it always decreases when *k* increases. The **elbow method** examines the inertia curve to identify a point where improvements slow down significantly.

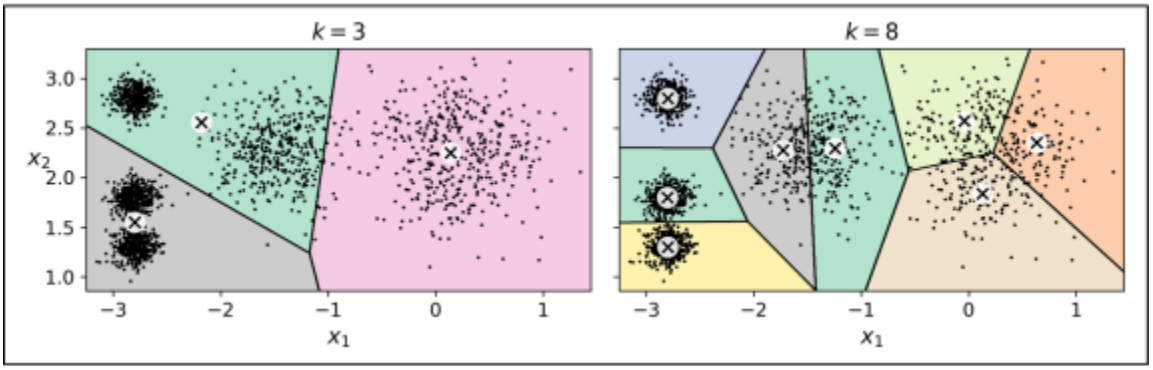

Figure 9-7. Bad choices for the number of clusters

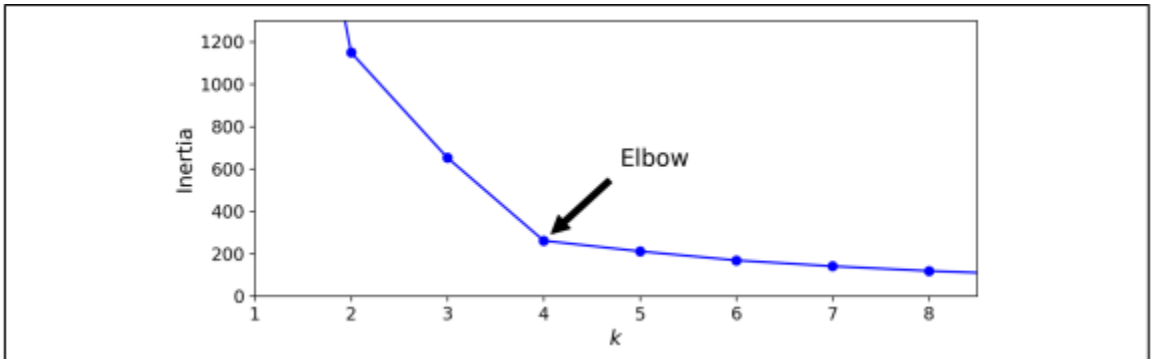

Figure 9-8. Elbow method for selecting k

A more informative metric is the **silhouette score**, which measures how well each instance fits within its cluster compared to neighboring clusters. Silhouette diagrams provide a detailed view of cluster quality and balance.

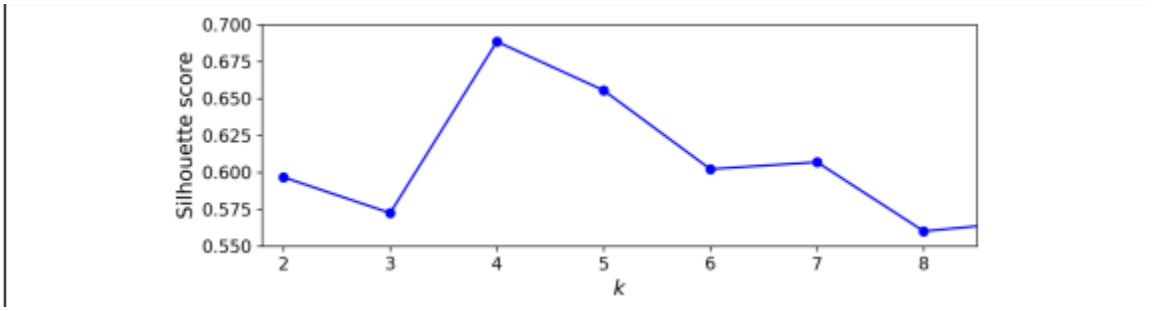

Figure 9-9. Selecting k using silhouette score

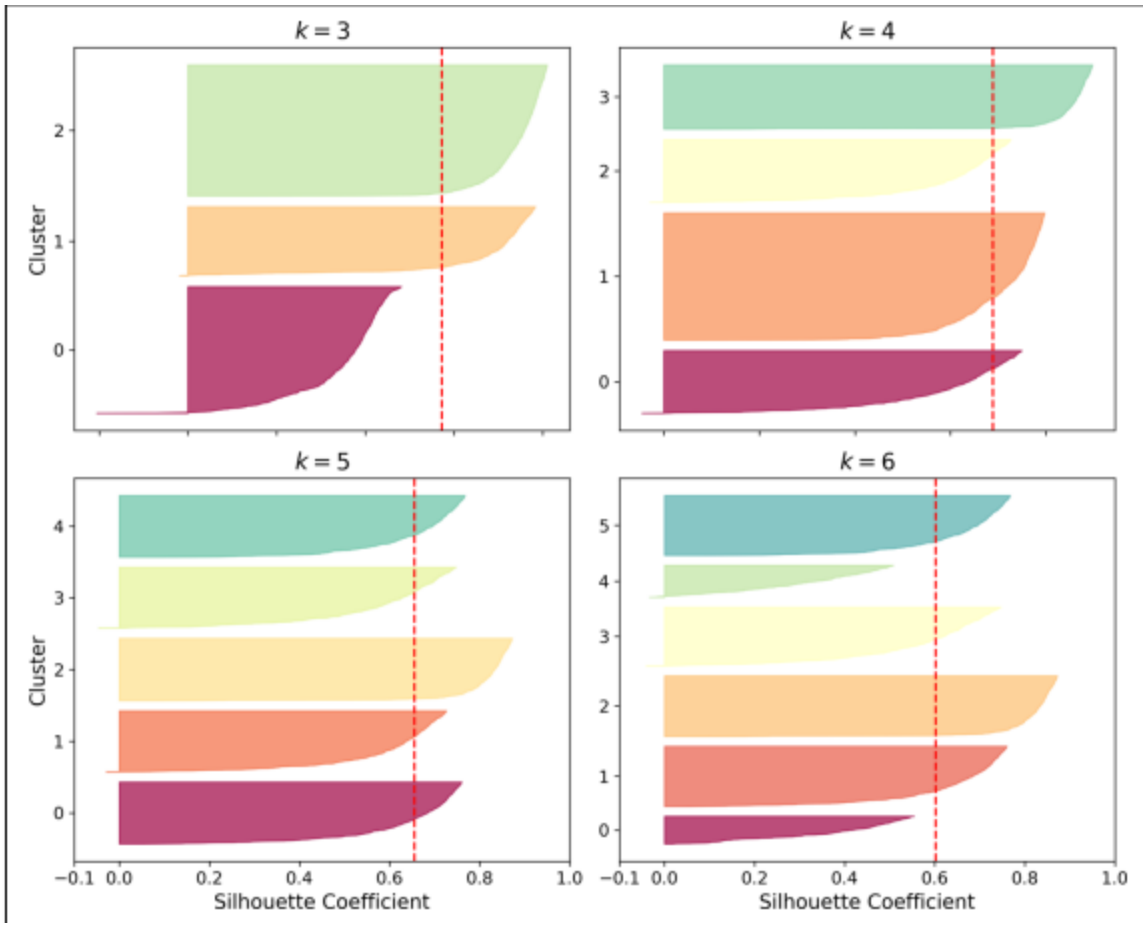

Figure 9-10. Silhouette diagrams for various k

Limits of K-Means

Despite its efficiency, K-Means struggles when clusters differ in size, density, or shape, especially for non-spherical or elongated clusters. Feature scaling is essential before applying K-Means, but even with scaling, some datasets are inherently unsuitable.

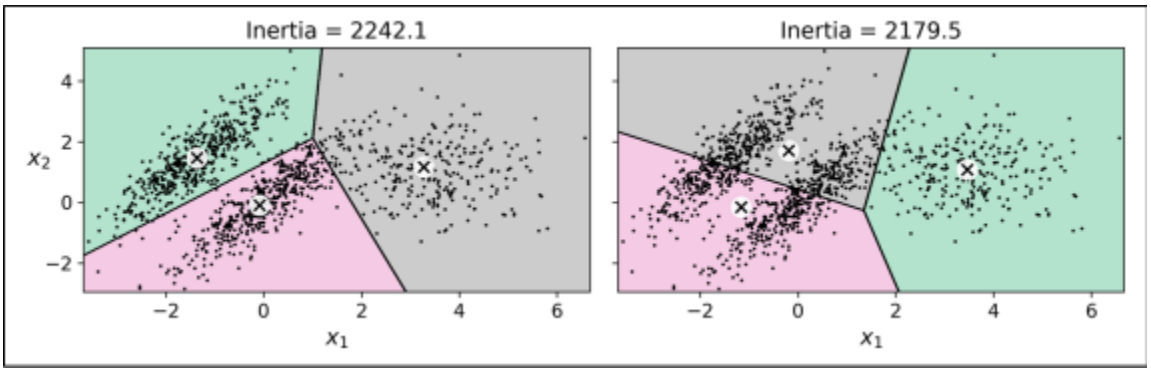

Figure 9-11. K-Means failing on ellipsoidal clusters

Using Clustering for Image Segmentation

Clustering can be applied to image segmentation by grouping pixels based on color similarity. By clustering RGB values and replacing each pixel with its cluster's mean color, images can be simplified significantly. This approach works well for tasks like satellite image analysis, though it may fail to isolate small but distinctive objects due to K-Means' bias toward similarly sized clusters.

Figure 9-12. Image segmentation using K-Means

Clustering as Preprocessing

Clustering can act as a powerful preprocessing step. By replacing raw features with distances to cluster centroids, supervised models can achieve higher accuracy. Applied to the digits dataset, K-Means preprocessing combined with Logistic Regression significantly improves classification performance, especially when the number of clusters is tuned via cross-validation.

Clustering for Semi-Supervised Learning

When labeled data is scarce, clustering enables efficient **label propagation**. By labeling only representative instances (such as those closest to cluster centroids) and propagating labels carefully, models can achieve high accuracy with minimal labeling effort. Restricting label propagation to instances close to centroids further reduces noise and improves results.

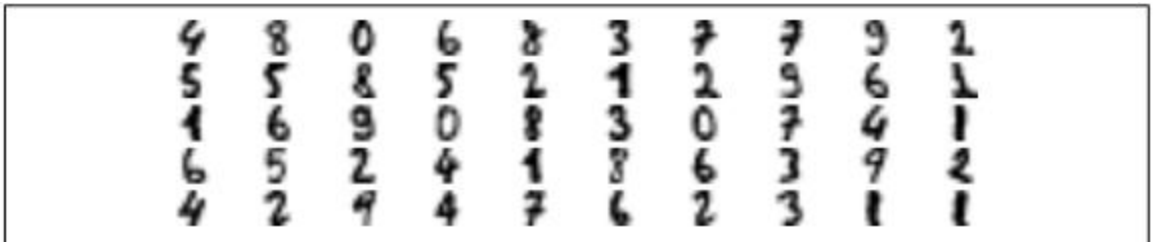

Figure 9-13. Representative digit images

Active Learning

Active learning further reduces labeling costs by allowing the model to request labels for the most informative or uncertain instances. Strategies such as uncertainty sampling or disagreement between models help prioritize which instances should be labeled next.

DBSCAN

DBSCAN defines clusters as dense regions separated by areas of low density. It identifies **core instances** based on neighborhood density and naturally detects outliers. Unlike K-Means, DBSCAN does not require specifying the number of clusters and can find clusters of arbitrary shapes. However, it is sensitive to its density parameters and struggles when cluster densities vary significantly.

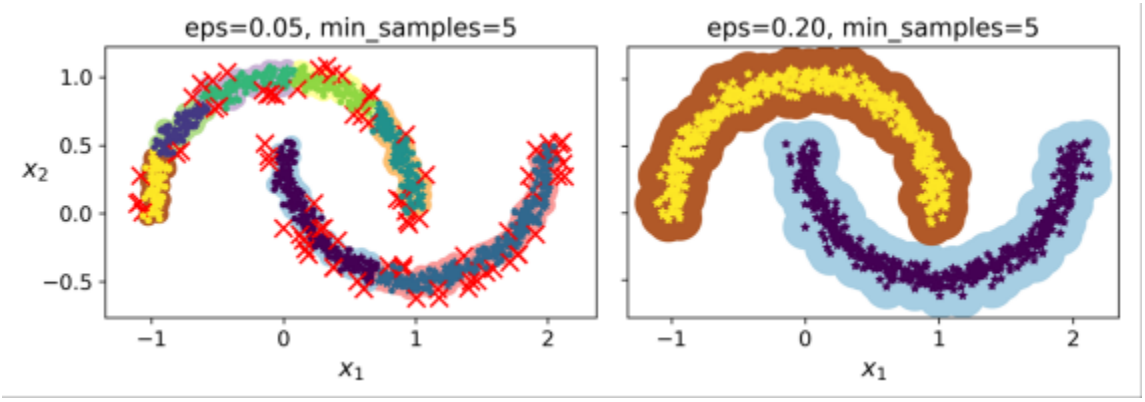

Figure 9-14. DBSCAN with different neighborhood radii

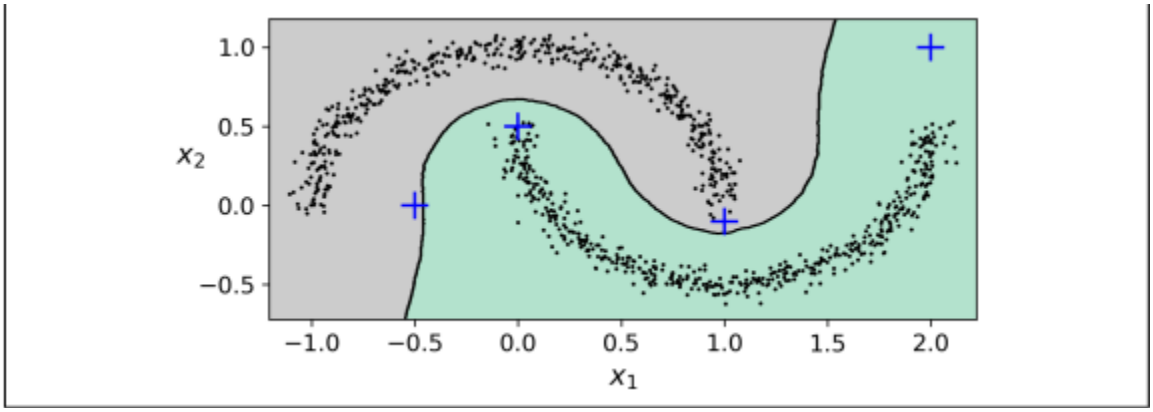

Figure 9-15. Decision boundary after DBSCAN clustering

Other Clustering Algorithms

Several alternative clustering methods address different data characteristics. Agglomerative clustering builds hierarchical structures, BIRCH targets very large datasets, Mean-Shift and Affinity Propagation rely on density and similarity, and Spectral Clustering leverages graph-based embeddings. Each method trades off scalability, flexibility, and interpretability.

Gaussian Mixture Models

Gaussian Mixture Models (GMMs) are probabilistic models that assume data is generated from a mixture of Gaussian distributions. Unlike K-Means, GMMs support **soft assignments** and can model clusters with different shapes, sizes, and orientations. They are trained using the **Expectation-Maximization (EM)** algorithm, which alternates between estimating cluster responsibilities and updating distribution parameters.

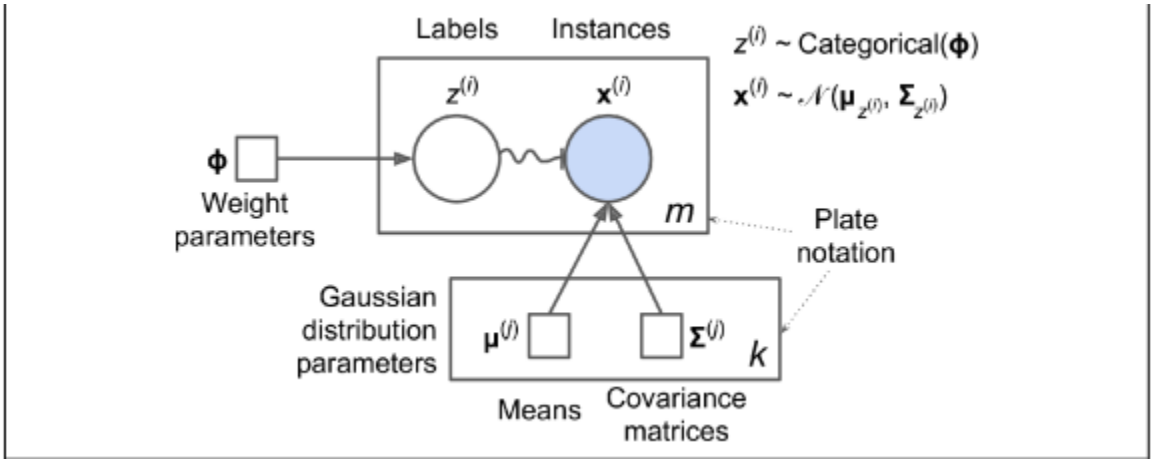

Figure 9-16. Graphical model of a Gaussian mixture

GMMs enable clustering, density estimation, and data generation. They also support anomaly detection by identifying instances in low-density regions.

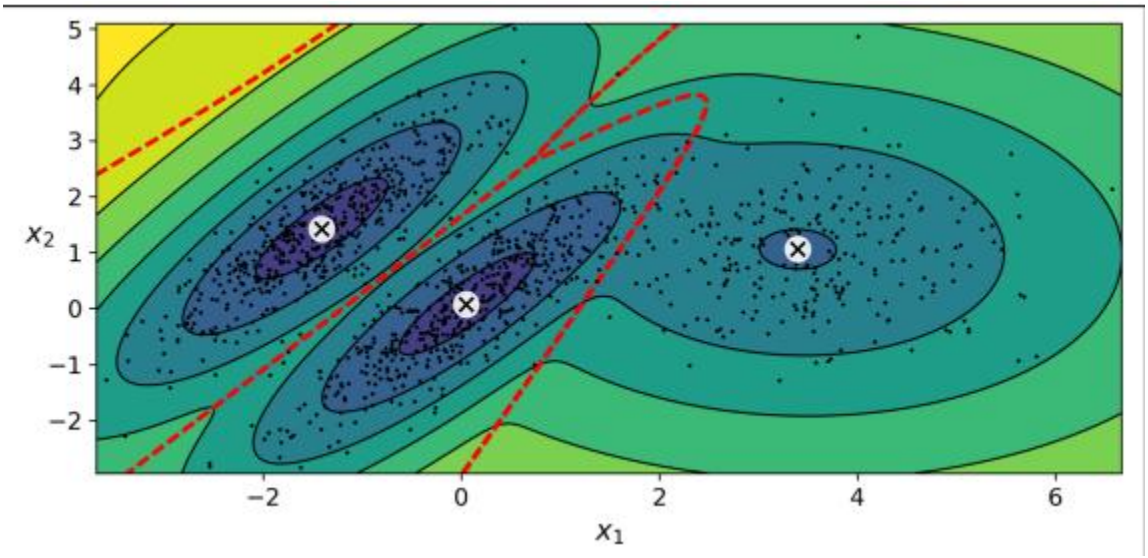

Figure 9-17. Gaussian mixture decision boundaries and density contours

---

Model Selection and Bayesian GMMs

Choosing the number of components in a GMM is typically done using information criteria such as **AIC** or **BIC**, which balance model fit and complexity.

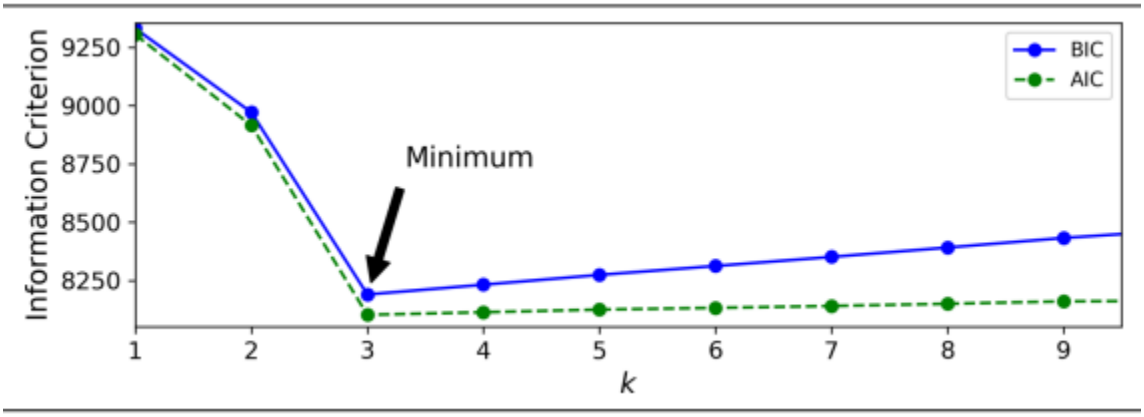

Figure 9-21. AIC and BIC for different numbers of clusters

Bayesian Gaussian Mixture Models extend GMMs by placing priors over parameters, allowing the model to automatically infer the effective number of clusters by assigning negligible weights to unnecessary components.

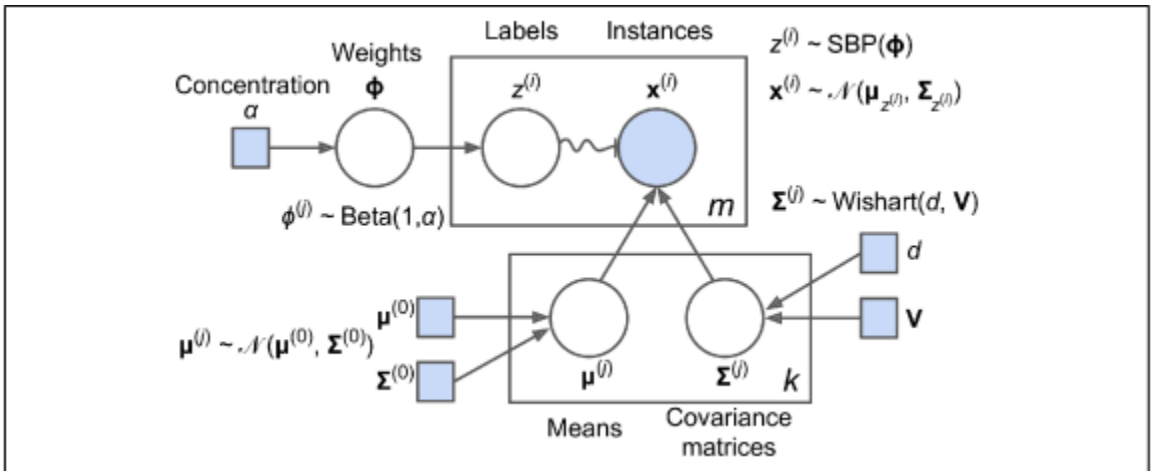

Figure 9-22. Bayesian Gaussian mixture model

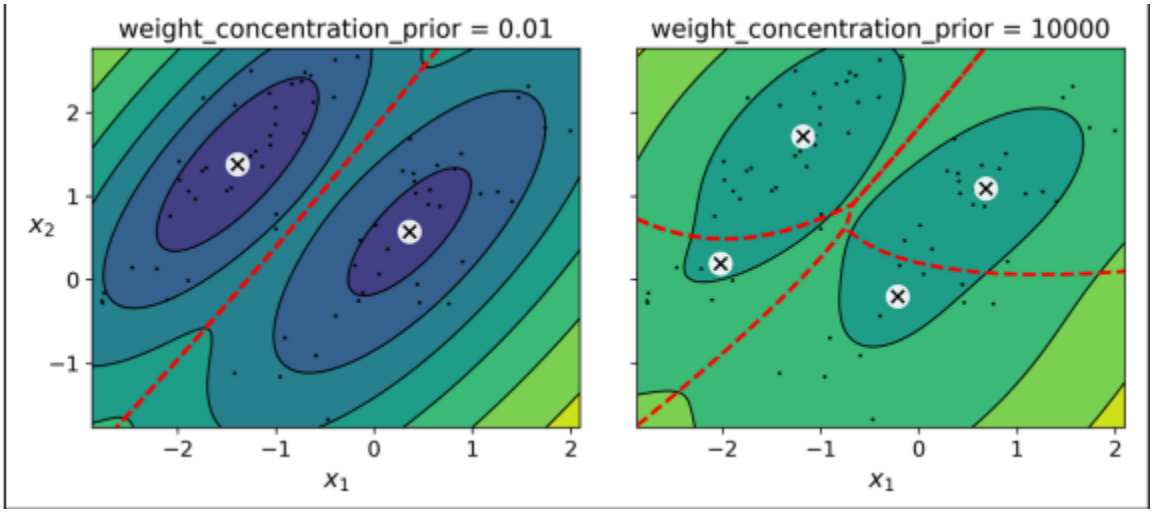

Figure 9-23. Effect of different concentration priors

Likelihood and Bayesian Inference

This chapter distinguishes probability from likelihood and explains how likelihood maximization underpins parameter estimation. Bayesian inference incorporates prior beliefs and updates them using observed data, often approximated through variational inference when exact computation is intractable.

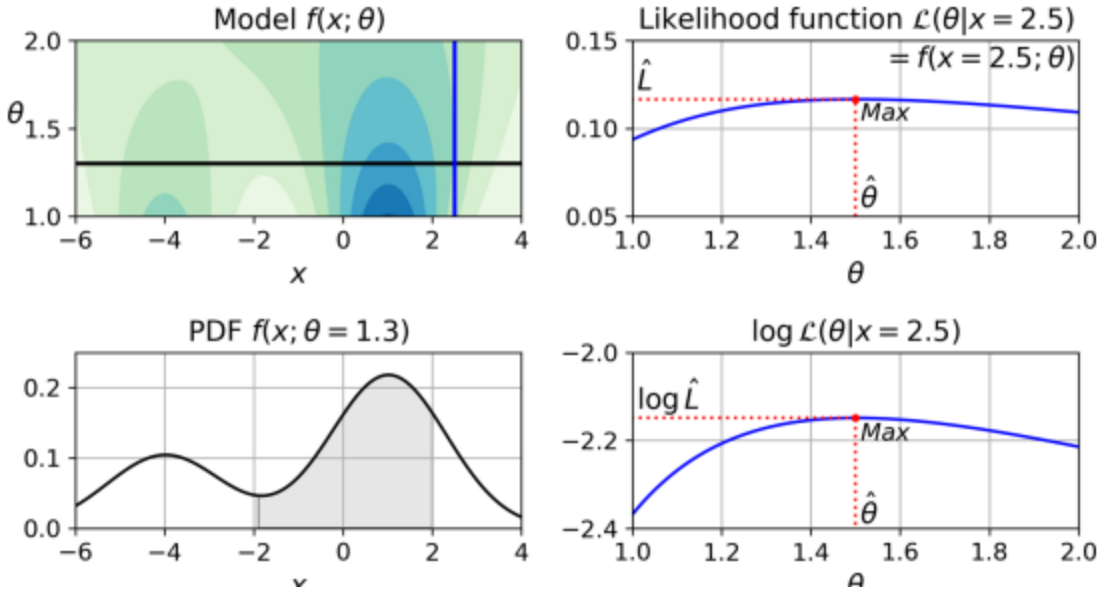

Figure 9-20. PDF, likelihood, and log-likelihood

Anomaly Detection

Unsupervised anomaly detection identifies instances that deviate significantly from normal patterns. Gaussian mixtures, PCA reconstruction error, Isolation Forests, LOF, Fast-MCD, and one-class SVMs each provide different trade-offs in scalability, robustness, and assumptions about data distribution.

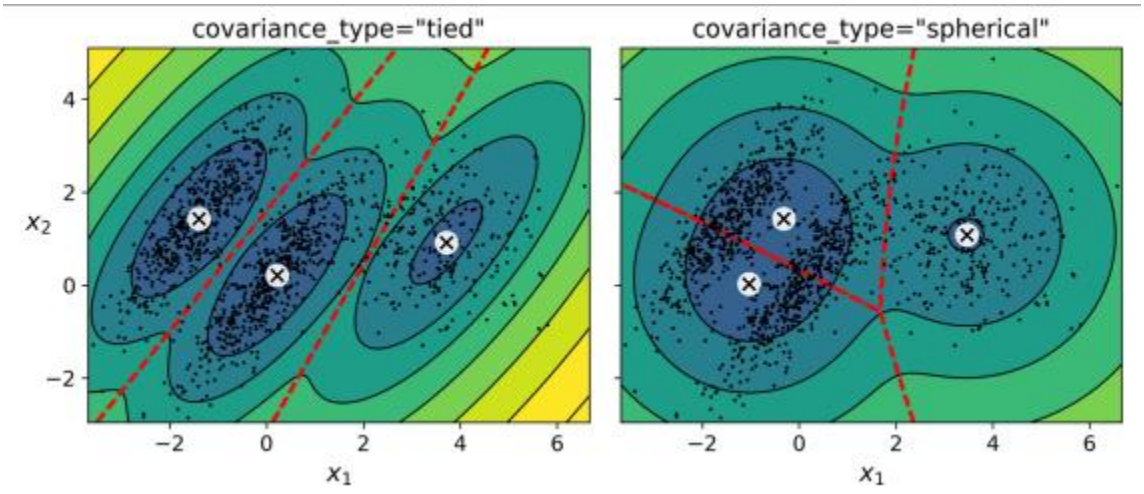

Figure 9-19. Anomaly detection using Gaussian mixtures

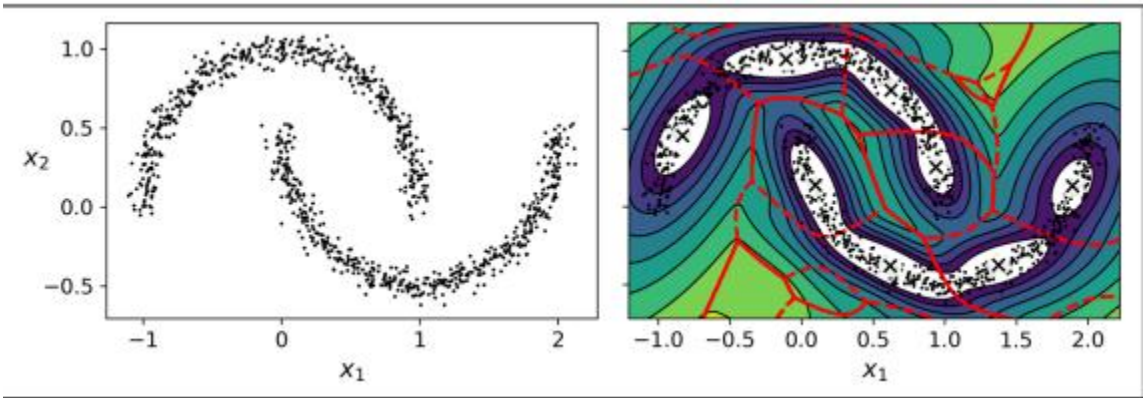

Figure 9-24. GMM failing on non-ellipsoidal clusters

Overall, this chapter demonstrates how unsupervised learning techniques uncover structure, reduce labeling costs, and enable scalable data analysis. Choosing the right algorithm depends heavily on data shape, density, dimensionality, and the downstream task.