K-Means Clustering

- **Clustering**: Task of partitioning data into groups called clusters based on similarity

Comparisons with kNN:

k-Means	k-Nearest Neighbors
Unsupervised	Supervised
Clustering	Classification
Parametric	Non-parametric
k = number of clusters (high = overfitting)	k = number of neighbors (low = overfitting)

Common Applications

1. Data Exploration: Summarize/ compress data, Partition data into groups
2. Customer Segmentation: Group customers based on purchasing behavior, Target marketing
3. Document Clustering: Group similar documents together, Topic modeling

K-Means

- One of the most popular clustering algorithms
- Simple and easy to implement
- **Basic Algorithm**:
  1. Randomly initialize K centroids
  2. Assign each data point to the nearest centroid
  3. Update the centroids to the mean of the data points assigned to it
  4. Repeat steps 2 and 3 until convergence
- **Properties**:
  - Will always converge (not necessarily to the right answer)
  - Sensitive to intialization
  - Terminates when centroids do not change
  - Makes *linear* decision boundaries
  - **MUST SCALE DATA** before applying K-means
    - Because K-means uses distance
    - If features are on different scales, the clustering will be biased towards the features with larger scales

---

- **Input**:
  - X: Set of n data points
  - K: Number of clusters
- **Output**:
  - K clusters with centroids $\mu_1, \mu_2, \ldots, \mu_K$

Choosing K

- K is a hyperparameter
  - As K increases -> smaller clusters
- No perfect way to choose K

1. **Elbow Method**:
   - Specific to k-means
   - $\text{Inertia} = \text{sum of intra-cluster distances}$
     - Sum of centroid to point distances of all points in the cluster of all clusters $\sum_{i=1}^{K} \sum_{x \in C_i} ||x - \mu_i||^2$
   - Inertia decreases as K increases, until it reaches 0 when K = n
   - Plot inertia vs K
     - Elbow point: point where inertia starts to decrease more slowly
2. **Silhouette Score**:
   - **Not dependent on cluster centers ->** can be used to compare different clustering algorithms
   - Gets worst as K increases, since being closer to neigouring clusters

$$\text{Silhouette Score} = \frac{b - a}{\max(a, b)}$$

   - a: Mean distance between a sample and all other points in the same cluster
   - b: Mean distance between a sample and all other points in the next nearest cluster
   - Range: $[-1, 1]$ (higher is better = high correlation within cluster)
   - **y-axis**: Sample number (similar thickness = balanced cluster sizes)
   - **x-axis**: Silhouette score

Gaussian Mixture Models (High-Level Overview)

- `GaussianMixture(n_components=3, covariance_type='full')`, `covariance_type`:
  - `full`: Each component has its own general covariance matrix
    - size: $K \times n\_features \times n\_features$
  - `tied`: All components share the same general covariance matrix
    - size: $n\_features \times n\_features$
  - `diag`: Each component has its own diagonal covariance matrix
    - size: $K \times n\_features$
  - `spherical`: Each component has its own single variance
    - size: K

How GMMs Work

$$P(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)$$

- $P(x)$: Probability of observing x
- π_k: Weight of the kth Gaussian (between 0 and 1)
- k: Number of clusters
- $\mathcal{N}(x|\mu_k, \Sigma_k)$: Gaussian distribution with mean μ_k and covariance Σ_k
- **Generative Model**: models the probability of a data point being generated from the mixture of Gaussians
- **High Level Algorithm** (non-convex optimization):
  1. Initialize π_k, μ_k, Σ_k (sensitive to init, can init with k-means)
  2. E-step: Each point, Compute the probability of each data point belonging to each cluster
  3. M-step: Each cluster, Update π_k, μ_k, Σ_k to maximize the likelihood of the data
  4. Repeat steps 2 and 3 until convergence
- Under the hood, GMMs use the **Expectation-Maximization (EM)** algorithm.
  - Basic idea: treat cluster assignments as hidden variables and iteratively update them
- **Other Properties**:
  - Can constrain the covariance matrix
  - Number of clusters is a hyperparameter and has a significant impact on the model

DBSCAN

- Density-Based Spatial Clustering of Applications with Noise
- **Idea**: Clusters are dense regions in the data space, separated by low-density regions
- Addresses K-Means' weaknesses:
  - No need to specify number of clusters
  - Can find clusters of arbitrary shapes
  - Can identify points that don't belong to any cluster (points don't have to belong to any cluster, label = `-1`)
  - Initialization is not a problem
- Comparison to K-means:
  - does not have to assign all points to clusters
  - no `predict` method unlike K-means
  - non-parametric
- Other Con: Needs tuning of 2 non-trivial hyperparameters:
  - `eps` (default=0.5): maximum distance between two samples for one to be considered as in the neighborhood of the other.
  - `min_samples` (default = 5): number of samples in a neighborhood for a point to be considered as a core point
- **DBSCAN Failure Cases**: Different densities of clusters

How DBSCAN works

- **Kinds of points**:
  - **Core point**: A point that has at least `min_samples` points within `eps` of it
  - **Border point**: A point that is within `eps` of a core point, but has less than `min_samples` points within `eps` of it
  - **Noise point**: A point that is neither a core point nor a border point
- **Algorithm**:
  - randomly pick a point that has not been visited
  - Check if it's a core point
    - See `eps` distance around the point if there are `min_samples` points
  - If yes, start a cluster around this point
  - Check if neighbors are core points and repeat
  - Once no more core points, pick another random point and repeat

Hierarchical Clustering

- Hard to decide how many clusters
  - So get complete picture of similarity between points then decide
- **Main idea**:
  - Start with each point as a cluster
  - Merge the closest clusters
  - Repeat until only a single cluster remains (n-1 steps)
- Visualized as a `dendrogram`

Dendrogram

- **x-axis**: data points
- **y-axis**: distance between clusters

- Is a tree like plot
  - New parent node for each 2 clusters that are merged
- Length of the vertical line at the point of merging is the distance between the clusters

Linkage Criteria

- Linkage Criteria determines how to find similarity between clusters
- **Single Linkage**:
  - Merge smallest **min** distance between points in two clusters
  - Can lead to chaining
- **Complete Linkage**:
  - Merge smallest **max** distance between points in two clusters
  - Can lead to crowding (tight clusters)
- **Average Linkage**:
  - Merge smallest **mean** distance between points in two clusters
- **Ward's Method**:
  - Minimizes the variance of the clusters being merged
  - Leads to equally sized clusters

Simplifying the Dendrogram

1. **Truncation**:
   - `scipy.cluster.hierarchy.dendrogram` has a `truncate_mode` parameter
   - Two levels:
     - `lastp`: show last p merged clusters (only p nodes are shown)
     - `level`: level is all nodes with `p` merges from the root
2. **Flatten**
   - Directly cut the dendogram at certain condition (e.g. distance or max number of clusters)

Principal Component Analysis (PCA)

Dimensionality Reduction Motivation

- **Curse of Dimensionality**
  - As the number of dimensions increases, the volume of the space increases so fast that the available data become sparse + computational complexity increases
- **Data Visualization**
  - It is difficult to visualize data in high dimensions

PCA Overview

- **Goal**: Find a low-dimensional representation of the data that captures as much of the variance as possible
- **Approach**
  - Find the lower dimension hyperplane that minimizes the reconstruction error
  - Model is the best-fit hyperplane

PCA Terminology

$$X = ZW$$

$$(n \times d) = (n \times k) \cdot (k \times d)$$

- X: original data, $(n \times d)$
- Z: coordinates in the lower dimension, $(n \times k)$
- W: lower dimension hyperplane, $(k \times d)$
  - W is the principal components
  - Rows of W are orthogonal to each other

Can **reconstruct** the original data with some error:

$$\hat{X} = ZW$$

Note: if $k = d$, then Z is not necessarily X but $\hat{X} = X$ (Frobenius norm)

- **Objective/ Loss Function**:
  - Minimize reconstruction error $\|ZW - X\|_F^2$
    - Frobeinus norm $||A||_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}^2}$
  - NOT the same as least squares
    - LS is vertical distance, PCA is orthogonal distance

PCA Math

- Singular Value Decomposition (SVD)

$$A = USV^T$$

  - A: original data, $(n \times d)$
  - U: **left singular vectors**, $(n \times n)$
    - orthonormal columns $U_i^T U_j = 0$ for all $i \neq j$
  - S: **diagonal matrix** of singular values, $(n \times d)$
    - square root of **eigenvalues** of $A^T A$ or AA^T
    - corresponds to the variance of the data along the principal components (in decreasing order)
  - V: **right singular vectors**, $(d \times d)$
    - orthonormal columns (**eigenvectors**)
    - principal components (get the first k columns)
- For dimensionality reduction, we can use the first k columns of U and the first k rows of V^T (equal to first k columns of V)
- **PCA Algorithm**
  1. Center the data (subtract the mean of each column)
  2. Compute the SVD of the centered data to get the principal components W.
     - W is the first k columns of V
  3. Variance of each PC is the square of the singular value s_i^2
  4. Drop the PCs with the smallest variance
- **Uniquess of PCA**
  - PCA are not unique, similar to eigenvectors
  - Can add constraints to make it closer to unique:
    - Normalization: $||w_i|| = 1$
    - Orthogonality: $w_i^T w_j = 0$ for all $i \neq j$
    - Sequential PCA: $w_1^T w_2 = 0$, $w_2^T w_3 = 0$, etc.
  - The principal components are unique up to sign

Choosing Number of Components

- No definitive rule
- Can look at:
  - Explained variance ratio
  - Reconstructions plot

PCA and Multicollinearity

- PCA can be used to remove multicollinearity
- **Concept**: the principal components are orthogonal to each other so they should not be correlated

PCA Applications

1. **Data Compression**
2. **Feature Extraction**
3. **Visualization of High-Dimensional Data**
4. **Dimensionality Reduction**
5. **Anomaly Detection**
   - Can use the reconstruction error to detect anomalies/ outliers (if the error is too large)
   - Outliers = high reconstruction error

K-means and PCA

- PCA is a generalization of K-means
- K-means is a special case of PCA where the principal components are the cluster centers
- K-means each example is expressed with only one component (one-hot encoding) but in PCA it is a linear combination of all components

NMF (Non-Negative Matrix Factorization)

- Useful for when data is created with several independent sources
  - e.g. music with different instruments
- **Properties**:
  - Coefficients and basis vectors (components) are **non-negative**
    - Unlike in PCA you can subtract, e.g. $X_i = 14W_0 - 2W_2$
    - Since cannot cancel out => **more interpretable**
  - Will get **different results for different number of components**
    - `n_components=2` will point at extreme, `n_components=1` will point at mean
    - Unlike PCA, where first component points at the direction of maximum variance regardless of the number of components
  - Slower than PCA

Comparison of PCA, LSA, and NMF

Differenciating feature	PCA	NMF	LSA
Primary use	General dimensionality reduction, visualization, noise reduction.	Part-based decomposition, collaborative filtering, topic modeling.	Topic modeling, document similarity, information retrieval.
Data types/constraints	Any numerical data.	Non-negative data (e.g., images, text counts), Non-negativity on both features and coefficients.	Text data (term-document matrices), typically applied to non-negative data (e.g., term frequencies).
Output components	Principal components that are orthogonal. (sorted by variance)	Basis vectors with non-negative entries only. (not sorted by variance)	Singular vectors representing latent semantic dimensions. (not orthogonal nor sorted)
Interpretability	Can look at the linear combinations of features. Often difficult to interpret components directly because of the cancellation effects.	Interpretability is better, due to non-negative and additive nature of components.	Moderate