Cluster Analysis

Nishanth Nair

April 11, 2020

Overview

- Form of Unsupervised learning.
- Clustering is a data mining (machine learning) technique used to place data elements into related groups without advance knowledge of the group definitions.

Types of clustering

- **Hard Clustering**: In hard clustering, each data point either belongs to a cluster completely or not.
- **Soft Clustering**: In soft clustering, instead of putting each data point into a separate cluster, a probability or likelihood of that data point to be in those clusters is assigned.
  - Makes more sense for applications like creating browsable hierarchies.
  - You may want to put a pair of sneakers in two clusters: (1) sports apparel and (2) shoes.

Assessing clustering tendency

- Before we begin to perform clustering, we'd like to know how to figure out how susceptible our data is to clustering methods.
- Is our data relatively homogeneous already or will we be able to find meaningful seperations?
- **Domain Knowledge**
  - nature of the data and collection methods can inform us about the need for clustering
- **Summary Statistics**
  - When our data is relatively clean and low-dimensional, looking at a table of summary statistics or some scatter plots can usually reveal how good clustering would be on the data
- **Hopkins Statistic**

  - A data set without any clustering tendency would look like data from a uniform distribution.
  - This can be tested using hypothesis testing with null hypothesis that the data is drawn from a uniform distribution.
  - Hopkins statistic measures deviation from a uniform distribution.

Preparing data for cluster analysis

- Standardization can help resultant clusters be equally influenced by each feature or dimension
  - Removes the effects of different measurement metrics.
  - Features with large ranges will dominate distance metrics between points. Standardization prevents this.
  - Z-Score standardization: Subtract mean and divide by standard deviation.
- Remove or impute all the missing values in our data
- High degrees of correlation between features and highly noisy features can make it more difficult to achieve a meaningful clustering,
  - Methods like Principal Components are often performed on the data prior to clustering.

Key Components of cluster analysis

- Distance or similarity or dissimilarity function
- Loss function to evaluate clusters
- Algorithm to optimize loss function

Distance Metrics

- Since this is an unsupervised setting, you would want to assign the same label to data points that are "close" to each other.
- clustering algorithms rely on a distance metric between data points to do this.
- This should be invariant to rotation and translation.
- Properties of a distance metric:
  - Non Negative
  - Symmetric
  - Satisfies triangle inequality

For numeric features

- **Minkowski distance**, L^n Norm: $L^n(x_1, x_2) = \sqrt[n]{\sum_i^N |x_{1,i} - x_{2,i}|^n}$
  - Manhattan distance, n=1
  - Euclidean distance, n=2

  - Chebyshev distance, $\lim_{x\to\infty}$, L^n is the max distance among all components of x.
- **Cosine Similarity**
  - Only angle between vectors matter
  - Similarity is the size of the angle

$$\text{Cosine similarity} = \frac{A.B}{||A||.||B||} = \frac{\sum_{i=1}^{n} A_i X B}{\sqrt{\sum_{i=1}^{n}(A_i)^2}.\sqrt{\sum_{i=1}^{n}(B_i)^2}}$$

For symbolic features

These could be features that have binay values or categorical values.

- **Levenshtein (edit) distance**
- **Hamming distance** - Measure of overlap between 2 vectors.
  - Example: A = 1011001001, B - 1001000011, Overlap = Hamming distance = 3

Types of clustering algorithms

- **Connectivity Models**(Hierarchical)
  - based on the notion that the data points closer in data space exhibit more similarity to each other than the data points lying farther away
  - Approach 1: Bottom up
    * First, classify all data points into separate clusters
    * Next, aggreegate clusters as the distance decreases
  - Approach 2: Top down
    * all data points are classified as a single cluster
    * Next partition them as the distance increases
  - Examples:
    * Hierarchical clustering
- **Centroid models**(Exclusive Clustering)
  - These are iterative clustering algorithms
  - notion of similarity is derived by the closeness of a data point to the centroid of the clusters
  - Number of clusters required at the end have to be mentioned beforehand, which makes it important to have prior knowledge of the dataset
  - Example:
    * K-Means clustering
- **Distribution Models** (Probablistic clustering)
  - Similiraty is based on probability that all data points in the cluster belong to the same distribution.
  - Example:
    * Expectation maximization
    * Gaussian mixture models

- **Density Moddels**(Overlapping clustering)
  - These models search the data space for areas of varied density of data points in the data space.
  - It isolates various different density regions and assign the data points within these regions in the same cluster.
  - Example:
    * DBSCAN
    * OPTICS

K-Means Clustering

- Goal: assign each of N points or observations to one of K clusters, where K is determined a priori
- Each cluster has a centroid μ_k
- Loss Function (Distance metric):

$$L = \sum_{i-1}^{N} \sum_{k=1}^{k} ||x_i - \mu_k||_2^2 \text{ ,(Euclidean distance)}$$

  - Where, N = Number of data points
  - k = Number of clusters
- Minimize Loss,

$$\text{arg } \min_c(L)$$

Algorithm

- STEP 0: Initialize μ_k to either a random value or some random point in the dataset for k= 1,..k
- STEP 1: Assign each datapoint to a cluster:

$$c_i = \{j : d(x_j, \mu_i) \leq d(x_j, \mu_l), l \neq j, j = 1, ..N\}, i = 1..k$$

- STEP 2: Update mean for each cluster to now be mean of all points assigned to cluster.

$$\mu_i = \frac{1}{|c_i|} \sum_{j \in c_i} x_j, \forall i$$

  - $|c|$ indicates number of elements in c.
- STEP 3: Repeat steps 1 and 2 until convergence. Two options to determine convergence:
  - Set max number of iterations. Stop after limit is reached.
  - Compare max difference between centroids from previous iteration. Set threshold or tolrance and stop when within this value.

Evaluating k-means results

- Use a confusion matrix or compute purity if labels are available to evaluate clustering results.
- Purity = correct assignments/total cluster assignments

Determine optimal K

- Structural knowledge important
- Loss will decrease as k increases
- Automatic mmethods
  - Gap statistic
  - Intracluster correlation
  - Elbow method
  - Silhouette algorithm

Elbow method

- **Distortion**:
  - It is calculated as the average of the squared distances from the cluster centers of the respective clusters.
  - Typically, the Euclidean distance metric is used.
- **Inertia**:
  - It is the sum of squared distances of samples to their closest cluster center.
- Approach:
  - We iterate the values of k from 1 to n and calculate the distortion and inertia for each value of k in the given range.
  - Plot the results
  - Select the value of k at the "elbow" i.e, the point after which the distortion or inertia start decreasing in a linear fashion

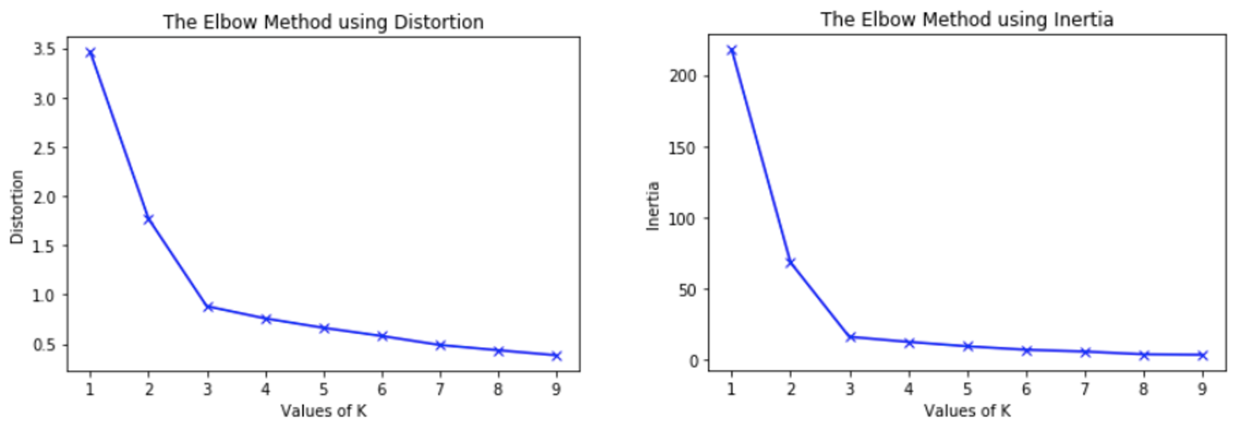


Figure 1: elbow method

Silhouette algorithm

- Assume that the data has already been clustered into k clusters by a clustering technique.
- For each data point, we define the following:
  - c_i - cluster assigned to a data point
  - $|c_i|$ - Number of points in a cluster
  - a(i) - how well assigned i is within its cluster

$$a(i) = \frac{1}{|c_i| - 1} \sum_{c_i} d(i,j), i \neq j$$

  - b(i) - dissimilarity from closest cluster to i

$$b(i) = min(\frac{1}{|c_j|} \sum_{j \in c_j} d(i,j)), i \neq j, j = 1, ..k$$

- Compute the silhouette coefficient as:

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

- Determine the average silhouette for each value of k.
- The value of k which has the maximum value of s(i) is considered the optimal number of clusters for the unsupervised learning algorithm.

Gaussian Mixture Models (GMM)

Overview

- We build a generative model that tries to explain the data.
- Use a gaussian mixture model to approximate the probability distribution of the observed data.
  - i.e. Compute probability that each data point was generated by each Gaussian distribution.
- Fit with expectation-maximization algorithm.
- Uses of GMM:
  - Speaker ID
  - Anomaly detection
    - Enumerate various features of the input x.
    - Model normal users using p(x).
    - Flag users with p(x) < e.

Expectation Maximization

- Expectation maximization (EM) algorithm is an iterative method to find maximum likelihood estimates of parameters in statistical models, where the model depends on unobserved latent variables.
- The EM iteration alternates between performing an expectation (E) step and a maximization (M) step
- **E-Step**:
  - Creates a function for the expectation of the log-likelihood evaluated using the current estimate for the parameters
- **M-Step**:
  - Computes parameters maximizing the expected log likelihood found on the E step.
  - These parameter estimates are then used to determine the distribution of the latent variables in the next E step.

properties of Gaussian (or normal) Distribution

- Given by:

$$N_{\mu,\sigma^2}(x) = \frac{1}{\sqrt{2\pi\sigma}}.e^{-\frac{(x-\mu^2)}{2\sigma^2}}$$

- **Central limit theorem**: Sum of samples from any random variable tends to be normally distributed.

- Sums and differences of Gaussians are also Gaussian.

- Negative log-liklehood looks like Euclidean distance.

$$ln\sqrt{2\pi\sigma} + \frac{(x-\mu^2)}{2\sigma^2}, \text{ (Negative log-likelihood)}$$

- **Bivariate Normal distribution** is given by:

$$N_{\mu,\Sigma} = \frac{1}{\sqrt{(2\pi)^2|\Sigma|}} e^{\frac{-1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}$$

$$\Sigma = \begin{vmatrix} \sigma_1^2 & r.\sigma_1.\sigma_2 \\ r.\sigma_1.\sigma_2 & \sigma_1^2 \end{vmatrix}$$

where,
x : 2D column vector
$|\Sigma|$: determinant of Σ
r : Correlation between the 2 variables
Σ : Covariance matrix

- **Multivariate Normal distribution** is given by:

$$f(x_1, x_2...x_k) = N_{\mu,\Sigma} = \frac{1}{\sqrt{(2\pi)^k|\Sigma|}} e^{\frac{-1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}$$

where,
k : number of dimensions

- If we have 2 random variables with probability functions N_{μ_1,σ_1^2} and N_{μ_2,σ_2^2}.
  - If the correlation between them is 0 (i.e they are independent), their joint distribution would be:

$$N_{\mu,\sigma^2} = N_{\mu_1,\sigma_1^2}.N_{\mu_2,\sigma_2^2}$$

- Larger variances give more spread distributions.
- Smaller variances give more peaked distributions.
- Non zero covariance values influence angle of distributions.

Maximum Likelihood Estimate

- Let $X_1, ...X_n$ be a random sample from a distribution with pdf

$$f(x_1, x_2..., x_n; \theta_1, \theta_2...\theta_n)$$

- Where $\theta_1, \theta_2...\theta_n$ have unknown values.
- When x_i's have known sample values, the pdf becomes a function of θ and it is called the likelihood function.
- When the sample size is large, the maximum likelihood estimator is atleast approximately unbiased i.e. $\mathrm{E}(\hat{\theta}) \approx \theta$ and has variance that is exactly or atleast approximately the Minimum Variance Unbiased Estimator (MVUE) of θ
- Now, if we assume the pdf is a Gaussian,
  - For the univariate case,

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2$$

```
- For the multivariate case,
```

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$\Sigma = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^T.(x_i - \mu)$$

$$\Sigma_{12} = \Sigma_{21}$$

Gaussian Mixtures

- Not all distributions are Gaussian
- Any distribution can be approximated if enough Gaussians used.
- We can use a mixture of Gaussians to model complex data.
- Each Gaussian has probability reflecting importance to overall model.
- Probability of each observation is weighted combination of Gaussians.
- If we have k Gaussians,

$$P(x_i) = \sum_{j=1}^{k} p_j . N^i_{\mu_j, \Sigma_j}$$

$$N^i_{\mu_j, \Sigma_j} = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} e^{\frac{-1}{2}(x_i - \mu)^T \Sigma^{-1} (x_i - \mu)}$$

where,
k : Number of gaussians
p_j : Weight of gaussian j
d : Number of dimensions of x or features

GMM Steps

- STEP 1: Initialize parameter values
  - Number of gaussians = k
  - Weight of each Gaussian, p_j = 1/k
  - Set μ to a random value or compute using k-means
  - Set σ for each Gaussian to be 1 and correlation to 0 or compute using k-means
- STEP 2: E-Step
  - We have $x_i, p_j and N_j$ from step 1.i is the data point and j is the Gaussian index.
  - Compute P(x_i), the probability distribution of x_i

$$P(x_i) = \sum_k p_j.N_j^i$$

  - Compute probability of Gaussian j given point x i.e the probability that x_i comes from distribution j

$$\hat{P}(j|x_i) = \frac{p_j.N_j^i}{P(x_i)}$$

- STEP 3: M-Step
  - Now we have fractional counts (probability that x_i comes from distribution j) for each data point, reflecting our belief about the hidden variables.
  - We now compute the weighted means and variances. We also update the weight of each Gaussian.
  - We use MLE, but with a slight modification. We weight each point by their fractional counts. This gives us:

$$\mu_j = \frac{\sum_{i=1}^{N} \hat{P}(j|x_i).x_i}{\sum_{i=1}^{N} \hat{P}(j|x_i)}$$

$$\Sigma_j = \frac{\sum_{i=1}^{N} \hat{P}(j|x_i).(x_i - \mu_j)^T.(x_i - \mu_j)}{\sum_{i=1}^{N} \hat{P}(j|x_i)}$$

$$p_j = \frac{\sum_{i=1}^{N} \hat{P}(j|x_i)}{N}$$

where,

N : Number of data points

$\hat{P}(j|x_i)$: Fractional count of point i in Gaussian j or prob that i is in j

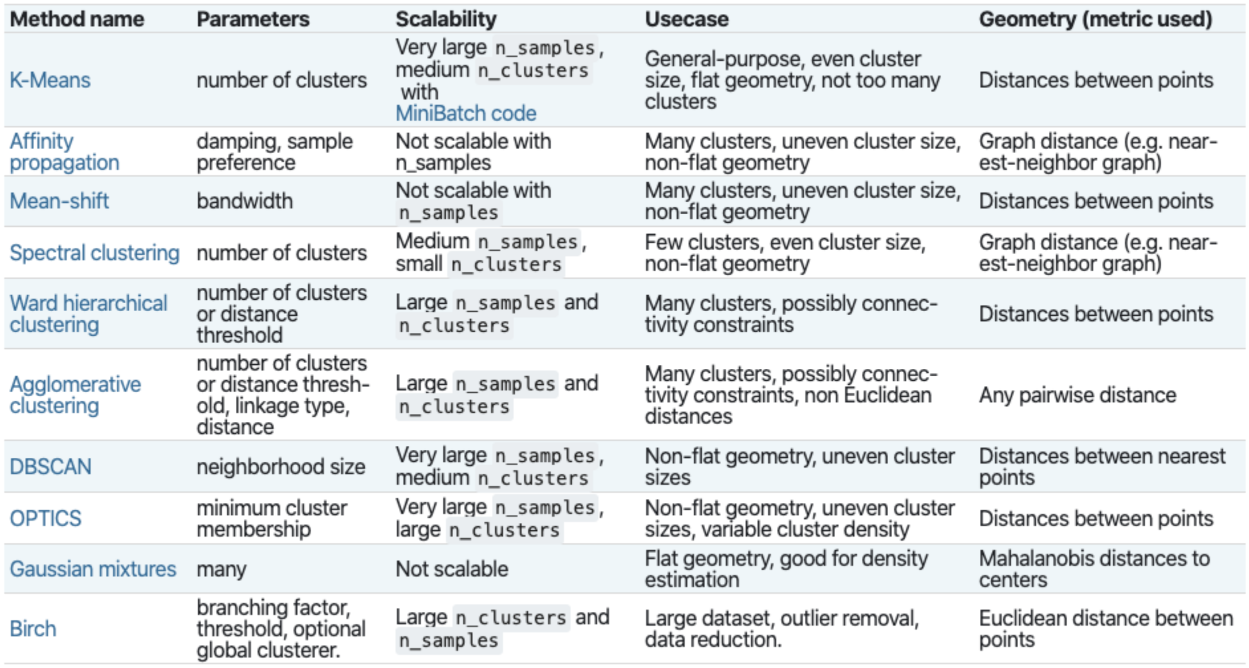
p_j : Weight of gaussian j

- STEP 4: Repeat steps 2 and 3 until convergence.

Considerations

- How do we choose number of Gaussians?
  - **Method 1**: Make an educated guess.
    - ∗ Mixtures of diagonal covariance Gaussians have fewer parameters than mixtures of full covariance Gaussians.
  - **Method 2**: Use a heuristic like the Bayesian Information Criterion (BIC) or Akaike Information Criterion (AIC).
    - ∗ Gives the model scenario a score that balances data likelihood and number of parameters
  - **Method 3**: Identify what works best on your downstream task.
    - ∗ If there is a method of evaluation, choose a number of components that gives the best results.
- How do we initialize parameters in STEP 1?
  - **Method 1**:
    - ∗ Set mixture weights uniformly.
    - ∗ Pick k data points at random for the k Gaussian means.
    - ∗ Set variances to the global variance of the data.
    - ∗ Extension: Run EM from many different starting points.
  - **Method 2**:
    - ∗ Fit a single Gaussian.
    - ∗ Split into two, randomly perturbing the means.
    - ∗ Run EM to fit.
    - ∗ Repeat steps 2 and 3 until you have the desired number of Gaussians.
- Use dev data to choose parameters if possible

Summary of clustering models



Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-Means	number of clusters	Very large `n_samples`, medium `n_clusters` with MiniBatch code	General-purpose, even cluster size, flat geometry, not too many clusters	Distances between points
Affinity propagation	damping, sample preference	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	bandwidth	Not scalable with `n_samples`	Many clusters, uneven cluster size, non-flat geometry	Distances between points
Spectral clustering	number of clusters	Medium `n_samples`, small `n_clusters`	Few clusters, even cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters or distance threshold	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints	Distances between points
Agglomerative clustering	number of clusters or distance threshold, linkage type, distance	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints, non Euclidean distances	Any pairwise distance
DBSCAN	neighborhood size	Very large `n_samples`, medium `n_clusters`	Non-flat geometry, uneven cluster sizes	Distances between nearest points
OPTICS	minimum cluster membership	Very large `n_samples`, large `n_clusters`	Non-flat geometry, uneven cluster sizes, variable cluster density	Distances between points
Gaussian mixtures	many	Not scalable	Flat geometry, good for density estimation	Mahalanobis distances to centers
Birch	branching factor, threshold, optional global clusterer.	Large `n_clusters` and `n_samples`	Large dataset, outlier removal, data reduction.	Euclidean distance between points

Figure 2: summary