Data Mining & Machine Learning

CS37300
Purdue University

Nov 3, 2023

Clustering

Clustering score functions

- score(C,D) = f(wc(C,D), bc(C))

cluster j centroid:

$$\underline{r}_j = \frac{1}{|C_j|} \sum_{\underline{x}_i \in C_j} \underline{x}_i$$

between-cluster distance:

$$\mathrm{bc}(C) = \sum_{1 \le j \le m \le k} \left\| \underline{r}_j - \underline{r}_m \right\|^2$$

within-cluster distance:

$$\mathrm{wc}(C,D) = \sum_{j=1}^{k} \sum_{\underline{x}_i \in C_j} \left\| \underline{x}_i - \underline{r}_j \right\|^2$$

k-means

- Algorithm:
  - Start with k randomly chosen centroids
  - Repeat until no changes in assignments
    - Assign each sample to closest centroid
    - Recompute cluster centroids (average of points in the cluster)

Score function: $$\text{score}(C,D) = \text{wc}(C,D) = \sum_{j=1}^{k} \sum_{\underline{x}_i \in C_j} \left\| \underline{x}_i - \underline{r}_j \right\|^2$$

Variations

- Selection of initial centroids
  - Run with multiple random selections, pick result with best score
  - Use hierarchical clustering to identify likely clusters and pick seeds from distinct groups

- Algorithm modifications:
  - Recompute centroid after each point is assigned
  - Allow for merge and split of clusters (for instance, if cluster becomes empty, start a new one from randomly selected point)

K-means++

- Selection of initial centroids
  - Choose a first center uniformly at random from the data points
  - For each data point x, computer D(x)= distance from x to the nearest center that has already been chosen
  - Choose the next center randomly according to a probability distribution P with P(x) proportional to $D(x)^2$ for each x
  - Repeat until we have k centers chosen, then run the k-means algorithm

- With this initialization, k-means typically converges faster
- Also, this initialization guarantees (in expectation) that the k-means score function upon convergence is within a factor log(k) of optimal

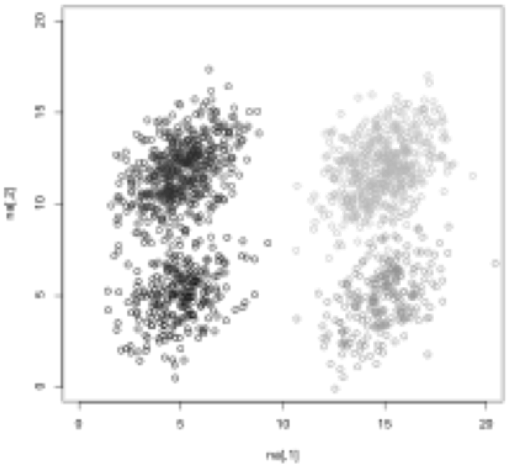
k-means summary

- Knowledge representation
  - k clusters are defined by canonical members (centroids)
- Model space the algorithm searches over?
  - *All possible partitions of the examples into k groups*
- Score function?
  - *Minimize within-cluster Euclidean distance*
- Search procedure?
  - *Iterative refinement correspond to greedy hill-climbing*

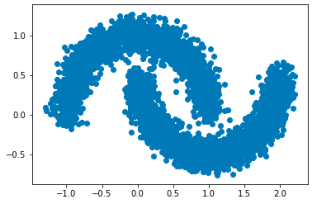
Probabilistic Models

Descriptive Modeling through Modeling Distribution

- Model the probability distribution $p(x)$

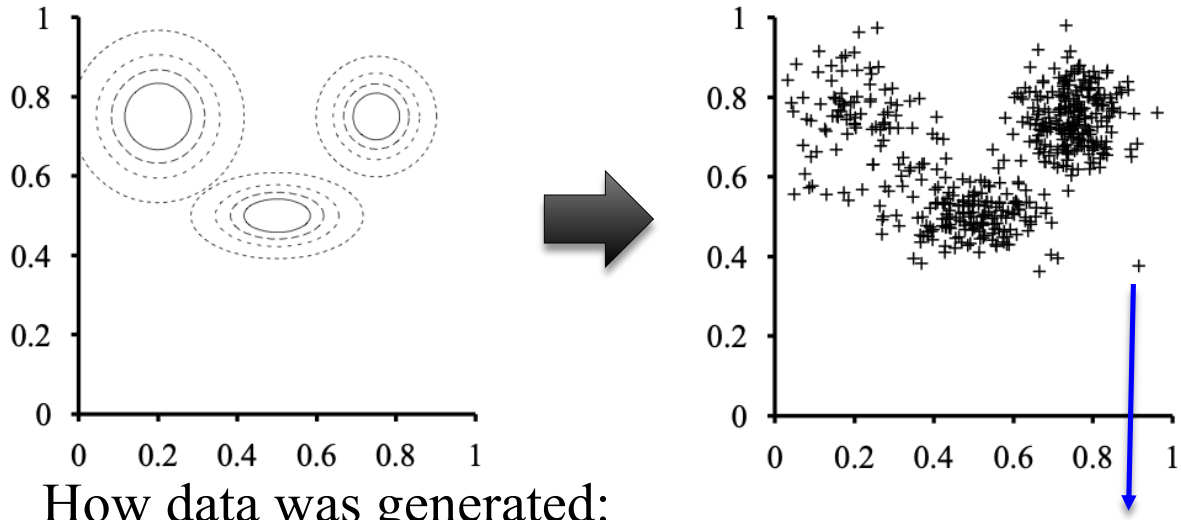


- Hard to describe using a single probability distribution
  - We will use a mixture of simple distributions
  - *Drawback*: data is may **not** be mixture of simple distributions:



- K-means is just an approximate solution (heuristic) of a specific type of mixture model (Gaussian Mixture Model)

Soft Clustering with Gaussian Mixture Models



How data was generated:

$$P[x_i, \text{Cluster}_i]$$

Given $\{x_i\}_I$ but not Cluster_i

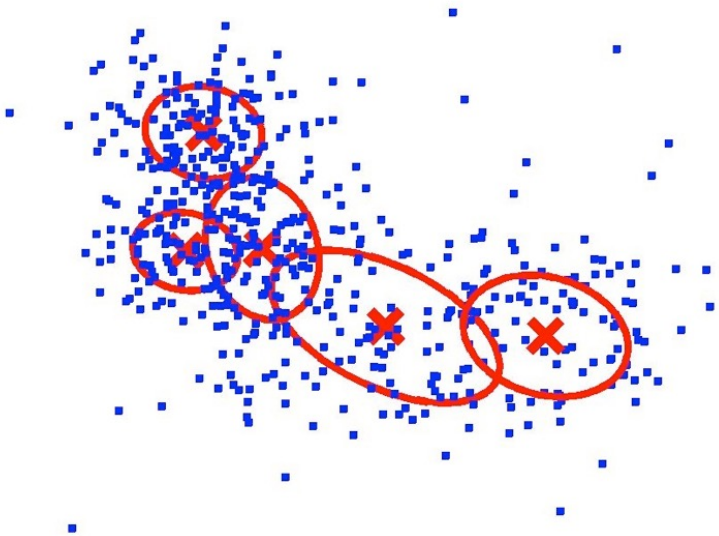
Goal is to find: $P[\text{Cluster}_i | x_i]$

Soft clustering assigns probability point belongs to cluster

Mixture models: Generative Story

1. Repeat:
   1. Choose a component according to $\mathbf{P(C)}$
   2. Generate the $\mathbf{X}$ as a sample from $\mathbf{P[X|C]}$

Probabilistic mixture model



$$p(x) = \sum_{k=1}^{K} w_k \, p(x; \theta_k)$$

parameters describe cluster k model

probability of point belonging to cluster k

probability of observing x

probability of x being generated from cluster k

Mixture Models

$$P(x) = \sum_{i=1}^{N} P(x|c_i)P(c_i)$$

- Objective function: log likelihood of data
- Naïve Bayes: $P(x|c_i) = \prod_{j=1}^{V} P(x_j|c_i)$
- Gaussian Mixture Model (GMM)
  - $P(x|c_i)$ is multivariate Gaussian
- Generally distributions, $P(x|c_i)$,can be anything

Note that cluster assignments are latent (unobservable)

Gaussian Mixture Models (GMMs) (mixture of Gaussians)

- A natural choice for continuous data

- Parameters:
  - Component prior probabilities $P[c_i = k]$
  - Mean of each component μ_i
  - Covariance of each component Σ_i

Multidimensional Gaussian

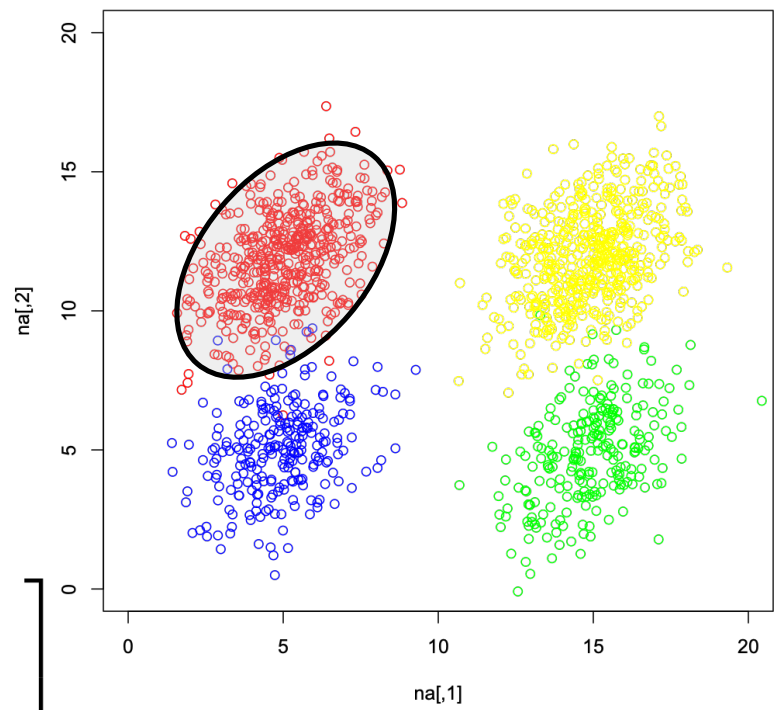
- A multi-dimensional Gaussians, for data with p dimensions is specified as follows

$$x \sim \mathcal{N}(\mu, \Sigma)$$

where:

$$\mu = \Big(E[X_1], ..., E[X_p]\Big)$$

$$\Sigma = \begin{bmatrix} Var(X_1) & ... & Cov(X_1, X_p) \\ ... & ... & ... \\ Cov(X_1, X_p) & ... & Var(X_p) \end{bmatrix}$$



Recall: $Cov(X_i, X_j) = \mathbb{E}[(X_i - \mathbb{E}[X_i])(X_j - \mathbb{E}[X_j])]$

$$p(\mathbf{x}) = p(x_1, ..., x_p) = \frac{1}{\sqrt{(2\pi)^p |\Sigma|}} exp\left(-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu)\right)$$

$$|\Sigma| = \text{ determinant}$$

Generative process (revisited)

- Assume that the data are generated from a mixture of k multi-dimensional Gaussians, where each component is has parameters: $N_k(\mu_k, \Sigma_k)$

covariance matrix (next slide)

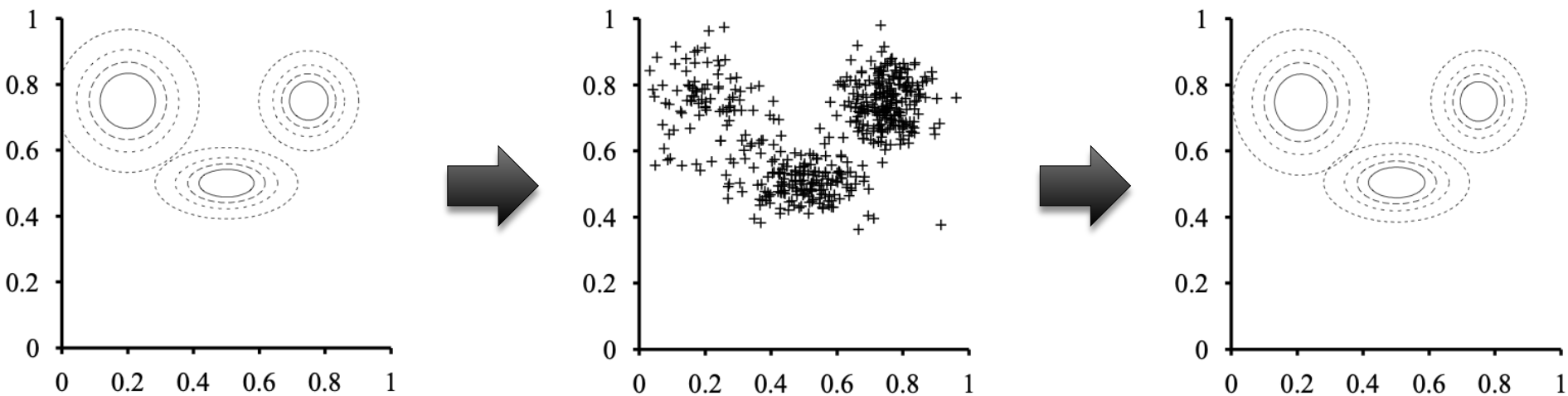
average (vector)

- For each data point:
  - Pick component Gaussian randomly with probability p(k)
  - Draw point from that Gaussian randomly by sampling from: $N_k(\mu_k, \Sigma_k)$

$$p(x) = \sum_{k=1}^{K} p(k)p(x|k)$$

$$= \sum_{k=1}^{K} p(k)p\Big(x|x \sim N(\mu_k, \Sigma_k)\Big)$$

GMM Parameter Estimation



Q: how can we learn parameters?

- *Chicken and egg problem:*
  - If we knew which component generated each data point it would be easy to recover the component Gaussians
  - If we knew the parameters of each component, we could infer a distribution over components to each data point.
- Problem: we know neither the assignments nor the parameters

Learning the model from data

- We want to invert this process
- Given the dataset, find the parameters
  - Mixing coefficients $p(k)$
  - Component means and covariance matrix $N_k(\mu_k, \Sigma_k)$
- If we knew which component generated each point then the MLE solution would involve fitting each component distribution to the appropriate cluster points
- Problem: the cluster memberships are **hidden**

Mixture models

- How to learn the model from data?
- We don't know the mixing coefficients ($w_{1...k}$) or the component parameters (θ)
- Solution:
  - Interpret mixing coefficients as prior probabilities of cluster membership
  - Use **Expectation-Maximization** algorithm to estimate model *(Dempster, Laird, Rubin, 1977)*

$$p(x) = \sum_{k=1}^{K} \underbrace{w_k}_{p(k)} \, p(x; \theta_k)$$

Expectation-maximization (EM) algorithm

- Popular algorithm for parameter estimation in data with hidden/unobserved values
  - Hidden variables=cluster memberships
- Basic idea
  - Initialize hidden variables and parameters
  - “Expectation” step: Estimate distributions of hidden variables given current estimates of the parameters
  - “Maximization” step: Update parameters by maximizing the expected log-likelihood (expectation under the estimated distributions of the hidden variables)
  - Repeat