Lec 5: Clustering, K-means and Gaussian Mixture Models (CS3780/5780, Sp26)

Setup: Unsupervised learning, Clustering

Past couple of lectures we looked at the supervised learning problem. In this lecture we look at the so called *unsupervised learning* problem where we are given an unlabeled dataset

$$D = \{x_1, \ldots, x_n\},$$

with *no* labels y^j. Our goal is to uncover latent structure in the data. In the next lecture the latent structure will be a low dimensional representation of the dataset, but for this lecture we are going to look at the problem of "clustering". That is given the dataset D, we want to find a natural way to partition the points in D into K clusters or groups. That is, for each x_i we associate with it a cluster identity $C_i \in \{1, \ldots, k\}$ such that this grouping makes some intuitive or natural sense. So a clustering of the dataset $D = \{x_1, \ldots, x_n\}$ yields the cluster assignment $\{C_1, \ldots, C_n\}$.

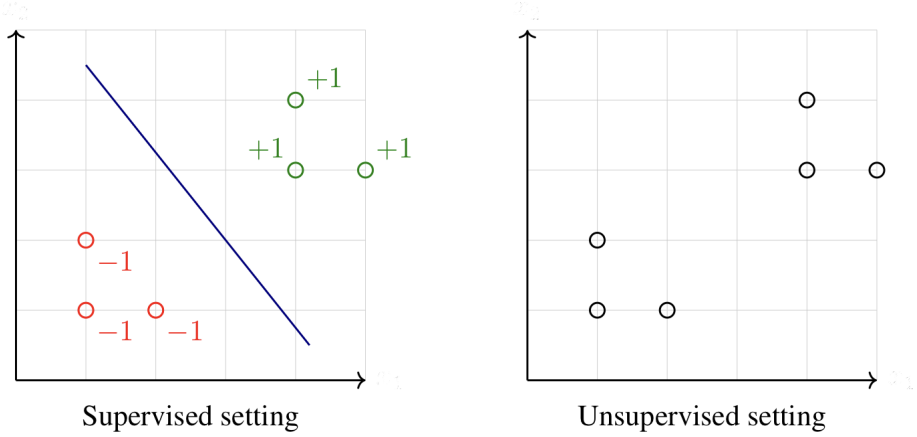

Figure 1: Supervised Vs Clustering

1 K-means clustering

Given the number of clusters k, our goal is to find a good clustering of the data points. What does it mean for clusters to be "good"? In k-means, we define goodness by the distance of each point to its cluster mean (centroid). This connects to the intuition "similar points are labeled similarly" (as in k-nearest neighbors), but now without labels.

1.1 The clustering algorithm

Given data points $D = x_1, \ldots, x_n$ with $x_j \in \mathbb{R}^d$ and an integer k, we want an assignment of points to k clusters that minimizes distance to cluster centers. Let the assignment be $C_j = i$ meaning point x_j belongs to cluster i, for $i \in \{1, \ldots, k\}$. This is a *hard assignment*: each point belongs to exactly one cluster.

The Lloyds algorithm also nick named k-means algorithm performs iteratively the following steps to convergence (or long enough) tp produce the final clustering assignments of the points:

(a) Randomly initialize k cluster centers $\mu_1, \ldots, \mu_k \in \mathbb{R}^d$.

(b) Reassign each point to the nearest center (for ℓ_2 / Euclidean distance):

$$C_j \;=\; \arg\min_{i \in \{1,\ldots,k\}} \|x_j - \mu_i\|_2 \,.$$

(c) Recompute each center as the mean of its assigned points:

$$\mu_i \;=\; \frac{\sum_{j=1}^{n} \mathbf{1}\{C_j = i\}\, x_j}{\sum_{j=1}^{n} \mathbf{1}\{C_j = i\}},$$

where $\mathbf{1}\{\cdot\}$ is the indicator function.

Repeat steps (b) and (c) until assignments do not change; then k-means is said to have converged.

1.2 Properties of k-means

Define the objective function function:

$$J(C, \mu) \;=\; \sum_{j=1}^{n} \left\| x_j - \mu_{C_j} \right\|_2^2 .$$

This above objective is often referred to as the K-means objective and the Lloyds algorithm can be seen as a greedy iterative procedure that tries to find clustering that minimizes this objective.

Observe that the algorithm:

- minimizes J w.r.t. C holding μ fixed (step (b)),
- then minimizes J w.r.t. μ holding C fixed (step (c)).

Hence J decreases monotonically (decreases or stays the same), and the objective is non-negative. Hence k-means is guaranteed to converge eventually.

While unlikely in practice, k-means can oscillate between different clusterings with the same objective value J.

Global optimality? In general, k-means does *not* guarantee finding a global minimum of J. In fact, finding a clustering assignment that minimizes J globally is known to be an np hard problem. Different initial centroids can yield different local minima using the Lloyd's algorithm. A common practical strategy is to run k-means multiple times with different random initializations and choose the run with the lowest J.

1.3 On choosing the number of clusters

There is no universally correct way to choose k; domain knowledge and application requirements help. A common heuristic is the *elbow method*:

(a) Run k-means for different values of k.

(b) For each k, compute $J(C, \mu)$ after convergence (often averaged over multiple random initializations).

(c) Choose the "elbow" point beyond which decreases in J are minimal.

Note: Increasing k will always decrease J (e.g., if each point is its own cluster), but improvements diminish after the "true" number of clusters (in an informal sense).

2 Gaussian Mixture Model

Consider the example where from telescope we have measurements of objects in night sky from telescope (Eg. photometry and spectroscopy). Without supervision we can model the data as gaussian mixture model to cluster objects into quasars, galaxies and stars.

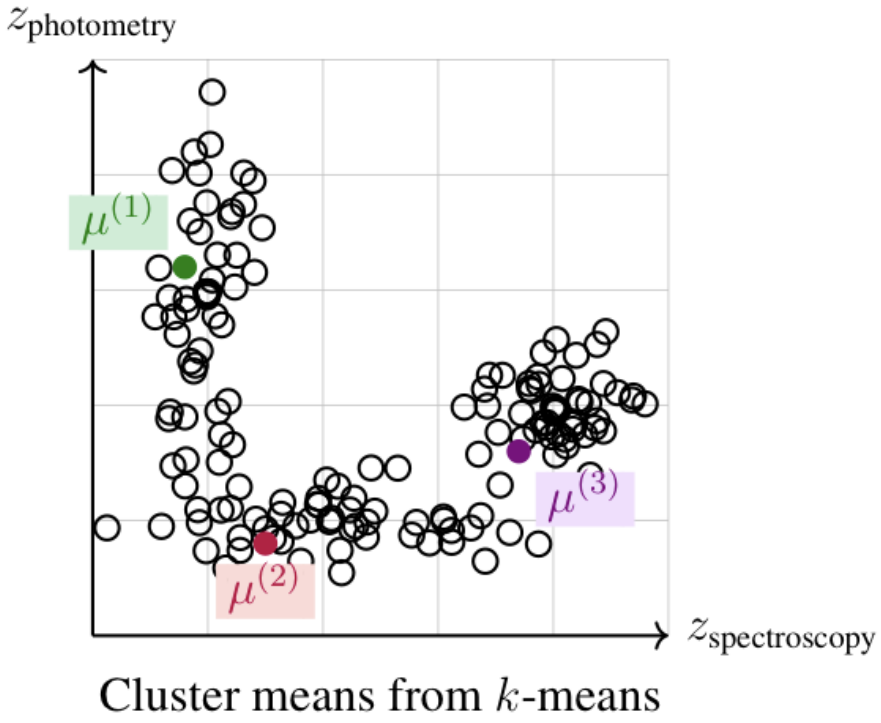

Figure 2: Quasars Vs Galaxies Vs Stars

2.1 Assumptions

We assume:

- There are k clusters (components), and k is known.

- Data from each cluster is (approximately) modeled by a Gaussian.
- Mixing proportions are unknown ("unknown mixture"): components can be sampled with different probabilities.

A mixture of Gaussians can approximate many continuous distributions arbitrarily well given enough components.

We only get data set D and dont know which points came from which cluster/gaussian. Since we are talking about gaussian distribution, one can never completely rule out that a particular point x_j does not belong to cluster/gaussian i. Hence we take a probabilistic view that yields *soft assignments*. Instead of stating $C_j = i$, we model the posterior probability

$$P(Z_j = i \mid x_j),$$

where Z_j is a latent variable indicating which component generated x_j. You can think of Z as a multinomial random variable indicating the outcome of rolling a k-faced die.

2.2 The mixture of Gaussians model (1D example)

For simplicity, consider 1D and $k = 2$. Let two Gaussians have means μ_1 and $\mu_{(2)}$ (and variances σ_1^2, σ_2^2). Suppose we sample from component 1 with probability p and from component 2 with probability $1 - p$. Then the prior probabilities are

$$P(Z_j = 1) = p, \qquad P(Z_j = 2) = 1 - p.$$

If we somehow knew which points came from which component, we could estimate parameters by computing within-cluster means/variances, and estimate p by frequency. Collect parameters as

$$\Theta = \{\mu_1, \mu_{(2)}, \sigma_1, \sigma_2, p\}.$$

Then we can compute the posterior via Bayes' rule:

$$P(Z_j = i \mid x_j; \Theta) = \frac{P(x_j \mid Z_j = i; \Theta)\, P(Z_j = i; \Theta)}{P(x_j)}.$$

By assumption,

$$x_j \mid \{Z_j = i\} \sim \mathcal{N}\left(\mu_i, \sigma_i^2\right).$$

The marginal density is

$$P(x_j) = \sum_{\ell=1}^{k} P(x_j \mid Z_j = \ell; \Theta)\, P(Z_j = \ell; \Theta).$$

For $k = 2$, this is

$$P(x_j) = p\, P(x_j \mid Z_j = 1; \Theta) + (1 - p)\, P(x_j \mid Z_j = 2; \Theta).$$

2.3 Formalization (general d and k)

Given $x_1, \ldots, x_n \in \mathbb{R}^d$ and k, our goal is to estimate

$$P(Z_j = i \mid x_j),$$

the probability that x_j belongs to cluster i given we observed x_j.

Model:

$$Z_j \sim \pi \quad \text{(for } k = 2 \text{ this reduces to Bernoulli}(p)\text{)},$$

and

$$x_j \mid \{Z_j = i\} \sim \mathcal{N}(\mu_i, \Sigma_i) \quad \text{(in 1D, } \Sigma_i = \sigma_i^2\text{)}.$$

We do not observe Z_j; it is latent.

2.4 The Expectation Maximization (EM) algorithm

Keep k-means in mind: guess centroids, assign points, recompute centroids. For mixture of Gaussians, we iteratively estimate parameters and compute *soft assignments*.

(a) Randomly initialize $\Theta = (\ \mu_1, \ldots, \mu_k, \Sigma_1, \ldots, \Sigma_k, \pi)$.

(b) Compute posteriors (soft assignments) using Bayes' rule. For $i \in \{1, \ldots, K\}$,

$$P(Z_j = i \mid x_j; \Theta) = \frac{P(x_j \mid Z_j = i; \Theta)\, \pi_i}{\sum_{l=1}^{k} P(x_j \mid Z_j = l; \Theta)\, \pi_l}.$$

Using the Gaussian density:

$$P(x_j \mid Z_j = i; \Theta) = \frac{1}{\sqrt{2\pi|\Sigma_i|}} \exp\left(-\frac{1}{2}(x_j - \mu_i)^\top \Sigma_i^{-1}(x_j - \mu_i)\right).$$

(c) Update parameters using the soft assignments:

$$\pi_i = \frac{1}{n}\sum_{j=1}^{n} P(Z_j = i \mid x_j)$$

$$\mu_i = \frac{\sum_{j=1}^{n} P(Z_j = i \mid x_j)\, x_j}{\sum_{j=1}^{n} P(Z_j = i \mid x_j)},$$

$$\Sigma_i = \frac{\sum_{j=1}^{n} P(Z_j = 1 \mid x_j)\, (x_j - \mu_i)(x_j - \mu_i)^\top}{\sum_{j=1}^{n} P(Z_j = 1 \mid x_j)},$$

Repeat steps (b) and (c) until the cluster assignments (posteriors) no longer change appreciably. As with k-means, this resembles coordinate descent and can be susceptible to local optima.

Connection to k-means. If variances $\sigma_i \to 0$ for all components, then the Gaussian likelihood

$$P(x_j \mid Z_j = i) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{(x_j - \mu_i)^2}{2\sigma_i^2}\right)$$

becomes extremely peaked around μ_i, yielding effectively hard assignments to the nearest mean. Thus, mixture-of-Gaussians clustering can be viewed as a generalization of k-means.