Unsupervised Learning

Given: Unlabeled data $\mathcal{D} = \{x_1, \dots, x_n\} \subseteq \mathbb{R}^d$ ← No labels y_i

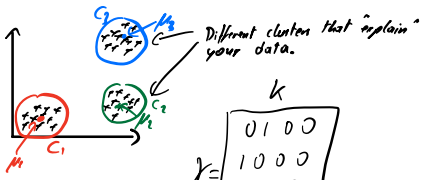
Unsupervised learning attempts to find (hidden) structure in your data.

Clustering: K-Means

(MacQueen 1967)

Assumptions: - We know there are k clusters
- Euclidean distance measure similarity
- clusters are spherical
- clusters have similar variance

Let data indices be $\{1, \dots, n\}$.



Cluster assignment: $\gamma_{il} = \begin{cases} 1 & \text{if } \vec{x}_i \text{ falls in cluster } l \\ 0 & \text{otherwise.} \end{cases}$

$\gamma \in \mathbb{R}^{n \times k}$

$\gamma = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$ (n rows, k columns)

Binary assignment. Each $\vec{x}_i$ is in exactly one cluster.

$\gamma_{il} \in \{0, 1\}$

Cluster center: $\vec{\mu}_l = \frac{1}{m_l} \sum_{i=1}^{n} \gamma_{il} \vec{x}_i$

$m_l = \sum_{i=1}^{n} \gamma_{il}$

m_l denotes the number of points assigned to cluster l.

Loss function: $\mathcal{L}(\gamma_{11}, \dots, \vec{\mu}_1, \dots, \vec{\mu}_k) = \frac{1}{n} \sum_{l=1}^{k} \sum_{i=1}^{n} \gamma_{il} \|\vec{x}_i - \vec{\mu}_l\|_2^2$ ← Average distance from each point to its cluster center.

↳ If you were to represent each $\vec{x}_i$ with its corresponding $\vec{\mu}_l$ instead, this would be the avg. squared error.

① Minimize $\mathcal{L}$ if γ is fixed (cluster assignment): (Assign $\vec{\mu}_1, \dots, \vec{\mu}_k$)

$\vec{\mu}_l = \underset{\vec{\mu}_l}{\operatorname{argmin}} \mathcal{L}(\gamma) \Rightarrow \vec{\mu}_l = \frac{1}{m_l} \sum_{i=1}^{n} \gamma_{il} \vec{x}_i$

mean of all $\vec{x}_i$ in l

For each $\vec{\mu}_l$, the contribution to the loss is $\sum_i \gamma_{il} (\vec{x}_i - \vec{\mu}_l)^2$ ← Parabola

② Minimize $\mathcal{L}$ if $\vec{\mu}_1, \dots, \vec{\mu}_k$ are fixed (cluster centers): (Assign γ)

$\gamma_{il} \leftarrow \begin{cases} 1 & \text{if } l = \underset{l'}{\operatorname{argmin}} (\vec{x}_i - \vec{\mu}_{l'})^2 \\ 0 & \text{otherwise} \end{cases}$ ← if $\vec{\mu}_l$ is the closest cluster center to $\vec{x}_i$

All points that are closest to μ_l.

For each x_i you can contribute $(\vec{x}_i - \vec{\mu}_1)^2$ or $(\vec{x}_i - \vec{\mu}_2)^2$ or ... or $(\vec{x}_i - \vec{\mu}_k)^2$.
For each x_i pick the smallest one to minimize $\mathcal{L}$.

K-means algorithm (aka Lloyd's algorithm):

1) Initialize $\vec{\mu}_1, \dots, \vec{\mu}_k$ somehow (e.g. random points, or heuristic)
2) Assign γ with ②
3) Assign $\vec{\mu}_1, \dots, \vec{\mu}_k$ using ①

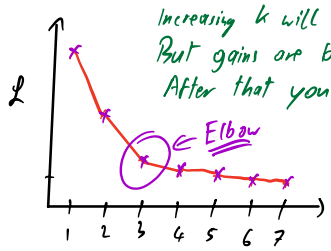
Repeat until convergence

Easy to show: $\mathcal{L}$ can never increase.

Initialization: - Can be unlucky. Perform multiple runs, pick solution with lowest $\mathcal{L}$.
- k-means++ (Arthur et.al. 2007): - pick $\vec{\mu}_1$ randomly from $\mathcal{D}$
  - For $l = 2$ to k:
    - $\forall x_i \in \mathcal{D}$ define $d_i^2 = \min_{l' < l} (x_i - \mu_{l'})^2$ ← squared Euclidean distance to closest existing center
    - pick μ_l randomly from $\mathcal{D}$ proportional to d_i^2. ← Avoids cluster centers that are very close

How to find k if it is unknown?

Run k-means for $k = 1, 2, \ldots$ monitor the loss $\mathcal{L}$
(For each k you have to average over multiple runs, due to the randomness of the initialization.)



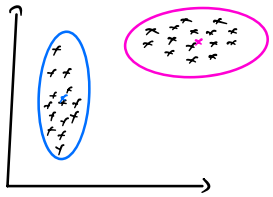
Increasing k will always decrease the best achievable $\mathcal{L}$.
But gains are bigger until you discovered the "true" k.
After that you will only split clusters into two.

Strength:
- Easy to implement
- fast for small k
- easily parallelizable

Limitations:
- sensitive to outliers
- slow for large k
- assumes all clusters are spherical
- makes only hard cluster assignments

Generalization: Gaussian Mixture Model (GMM)

Each cluster becomes a Gaussian $\mathcal{N}(\vec{\mu}_\ell, \Sigma_\ell)$



Almost the same as k-means, but each point has a distribution over clusters $\gamma_{\ell i} = p(C_\ell | x_i)$.

$$p(C_\ell | \vec{x}_i) = \frac{p(\vec{x}_i | C_\ell) p(C_\ell)}{\sum_{\ell'} p(\vec{x}_i | C_{\ell'}) p(C_{\ell'})} \quad \text{(Bayes Rule)}$$

$$\vec{\mu}_\ell \leftarrow \frac{1}{\sum_j \gamma_{\ell j}} \sum_{i=1}^{n} \gamma_{\ell i} \vec{x}_i \leftarrow \text{weighted average / expected center of cluster } \ell$$

$$\vec{\Sigma}_\ell \leftarrow \frac{1}{\sum_j \gamma_{\ell j}} \sum_{i=1}^{n} \gamma_{\ell i} (\vec{x}_i - \vec{\mu}_\ell)(\vec{x}_i - \vec{\mu}_\ell)^T$$

$$\pi_\ell \leftarrow \frac{\sum \gamma_{\ell i}}{n} \leftarrow \pi_\ell = p(C_\ell)$$

$$\gamma_{\ell i} \leftarrow \frac{p(x_i; \mu_\ell, \Sigma_\ell)\, \pi_\ell}{\sum_{\ell'} p(x_i; \mu_{\ell'}, \Sigma_{\ell'})\, \pi_{\ell'}}$$

M-Step — Maximization

E-Step — Estimation

$\gamma =$

0.5	0.2	0.1	0.1
0.1	0.3	0.4	0.2
0.9	0.0	0.0	0.1
0.8	0.1	0.05	0.05
0.7	0.1	0.1	0.1
0.0	0.0	0.9	0.1
1.0	0.0	0.0	0.0
0.3	0.2	0.5	0.0

soft assignment: $\gamma_{i\ell} \in (0, 1)$ $\quad \sum_\ell \gamma_{i\ell} = 1$

Strengths:
- more flexible than k-means
- probabilities tell you how well a sample fits into cluster

Weaknesses:
- slower than k-means
- can suffer from singularities (single point clusters)

(If $\Sigma_\ell = I\sigma^2$ GMM becomes k-means as $\sigma^2 \to 0$.)

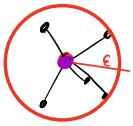
$\Sigma_\ell = I\sigma^2$: spherical

$\sigma^2 \to 0$: hard assignments

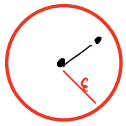
DBSCAN:

1. connect each point with all neighbors within ϵ-radius.

(Density based spatial clustering with noise)
Ester et. al 1996



core point
$\geq M$ neighbors



non-core point
$< M$ neighbors

Assumption:

- Clusters have similar density
- Similar points share clusters

Repeat:

- Start with random unassigned core-point.
  assign it to a new cluster ID l.
- Until no longer possible:
  - Assign all core points with a core-point neighbor in cluster l also to l.
- Assign all non-core points with a core point neighbor in cluster l to l.
- Assign all non-core points without core-point neighbor as noise

Apply DBSCAN on this ϵ-nearest neighbor graph: