**Unsupervised Learning**

Given: Unlabeled data \( D = \{ x_1, \ldots, x_n \} \subset \mathbb{R}^d \) — No labels \( y 

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

**Clustering: K-Means**

*(Ref: Cagan 1969)*

**Assumptions:**
- We know there are \( k \) clusters
- Euclidean distance measures similarity
- Clusters are spherical
- Clusters have equal variance

Let data indices be \( \{1, \ldots, n\} \)

Cluster assignment:
\[
x = \begin{cases} 1 & \text{if } x \text{ falls in cluster } k \\ 0 & \text{otherwise} \end{cases}
\]

Cluster center:
\[
\mu_k = \frac{1}{n_k} \sum_{i=1}^{n_k} x_i
\]

Loss function:
\[
L(\mu_1, \ldots, \mu_k) = \frac{1}{2} \sum_{k=1}^{k} \sum_{i=1}^{n_k} (x_i - \mu_k)^2
\]

1. **Optimize \( \mu \) if \( x \) is fixed.**
   \[
   \mu_k = \arg\min_x L(x) \Rightarrow \mu_k = \frac{1}{n_k} \sum_{i=1}^{n_k} x_i
   \]
   For each \( \mu_k \), the contribution to the loss is \( \sum_i (x_i - \mu_k)^2 \) — Parabola

2. **Minimize \( L \) if \( \mu_1, \ldots, \mu_k \) are fixed.**
   \[
   x_i = \begin{cases} 1 & \text{if } x_i = \arg\min_k (x_i - \mu_k)^2 \\ 0 & \text{otherwise} \end{cases}
   \]
   All points that are closest to \( \mu_k \).
   For each \( k \), you can contribute \( (x_i - \mu_k)^2 \) or \( (x_i - \mu_k)^2 \).
   For each \( k \), pick the smallest one to minimize \( L \).

**K-means algorithm:**

1. Initialize \( \mu_1, \ldots, \mu_k \) somehow
   (e.g., random points or heuristics)
2. Assign \( x \) with \( \mu_k \)
3. Assign \( x \) to \( \mu_k \) using 2
   Repeat until convergence

**Initialization:**
- Can be arbitrary. Perform multiple runs, pick solution with lowest \( L \).
- Heuristics (Arthur 2007):
  - Pick \( \mu_k \) randomly from \( D \)
  - For \( k \) to \( k \):
    - Use \( D \) to define \( d = \max_{i \neq j} (x_i - x_j)^2 \)
    - Pick \( \mu_k \) randomly from \( D \) proportionate to \( d \)

**Easy to show:** \( L \) can never increase.
How to find k if it is unknown?

Run k-means for \( k=1,2,... \) and monitor the cost \( \Lambda \)
(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 \( \Lambda \). But gains are bigger until you discover 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 \( N(\mu_i, \Sigma_i) \)

Almost the same as k-means, but each point has a distribution over clusters \( \pi_i \): \( \forall x \in \mathbb{R}^d \):

\[
\pi_i \sim \frac{p(x_i | \mu_i, \Sigma_i)}{\sum_{j} p(x_i | \mu_j, \Sigma_j)}
\]

\[
\mu_i \sim \frac{1}{\pi_i} \sum_{x_i} x_i
\]

\[
\Sigma_i \sim \frac{1}{\pi_i} \sum_{x_i} (x_i - \mu_i)(x_i - \mu_i)^T
\]

E-Step

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

**Weaknesses:**
- slower than k-means
- can suffer from singularity (single point clustering)
**DBSCAN**: 1. Connect each point with all neighbors within ε-radius.

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

**Assumption:**
- Clusters have similar density
- Similar points share clusters

- **Core point**
- **Non-core point**
- **ε-neighbor**
- **ε-neighbor**

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

Apply DBSCAN on the current network graph.