Supervised and unsupervised learning

Supervised learning problems:
- Data $(X, Y)$ is “labelled” (input/output) with joint density $P(X, Y)$.
- We are mainly interested by the conditional density $P(Y|X)$.
- Example: regression problems, classification problems, etc..

Unsupervised learning problems:
- Data $X$ is not labelled and has density $P(X)$.
- We want to infer properties of $P(X)$ without the help of a “supervisor” or “teacher”.
- Examples: Density estimation, PCA, ICA, sparse autoencoder, clustering, etc..

Clustering

We try to partition observations into “clusters” such that:
- Intra-cluster distance is minimized.
- Inter-cluster distance is maximized.

For graphs, we want vertices in the same cluster to be highly connected, and vertices in different clusters to be mostly disconnected.
The K-means algorithm

- Goes back to Hugo Steinhaus (of the Banach–Steinhaus theorem) in 1957.

Steinhaus authored over 170 works. Unlike his student, Stefan Banach, who tended to specialize narrowly in the field of functional analysis, Steinhaus made contributions to a wide range of mathematical sub-disciplines, including geometry, probability theory, functional analysis, theory of trigonometric and Fourier series as well as mathematical logic. He also wrote in the area of applied mathematics and enthusiastically collaborated with engineers, geologists, economists, physicians, biologists and, in Kac’s words, "even lawyers".


The K-means algorithm (cont.)

The K-means algorithm is a popular algorithm to cluster a set of points in $\mathbb{R}^p$.

- We are given $n$ observations $x_1, x_2, \ldots, x_n \in \mathbb{R}^p$.
- We are given a number of clusters $K$.
- We want a partition $\hat{S} = \{S_1, \ldots, S_K\}$ of $\{x_1, \ldots, x_n\}$ such that
  $$\hat{S} = \arg\min_S \sum_{i=1}^K \sum_{x_j \in S_i} \|x_j - \mu_i\|^2,$$
  where $\mu_i = \frac{1}{|S_i|} \sum_{x_j \in S_i} x_j$ is the mean of the points in $S_i$ (the "center" of $S_i$).
- The above problem is NP hard.
- Efficient approximation algorithms exist (converge to a local minimum though).

Some equivalent formulations

- Note that
  $$\frac{1}{2} \sum_{i=1}^K \sum_{x_j \in S_i} \sum_{x_k \in S_i} \|x_j - x_k\|^2 = \sum_{i=1}^K |S_i| \sum_{x_j \in S_i} \|x_j - \mu_i\|^2,$$
  which leads to an equivalent formulation of the above problem.
- For any $S \subset \{x_1, \ldots, x_n\}$,
  $$\mu_S := \frac{1}{|S|} \sum_{x_i \in S} x_i = \arg\min_m \sum_{x_i \in S} \|x_i - m\|^2.$$
  Thus, the K-means problem is equivalent to
  $$\arg\min_{S,(m_1, \ldots, m_K)} \sum_{i=1}^K \sum_{x_j \in S_i} \|x_j - m_i\|^2.$$
- Other equivalent problem: solve
  $$\arg\min \sum_{(m_1, \ldots, m_K)} \min_{1 \leq i \leq K} \|x_j - m_i\|^2,$$
  and let $S_i := \{x_j : \|x_j - m_i\|^2 \leq \|x_j - m_k\|^2 \forall k = 1, \ldots, K\}$.

Lloyd’s algorithm

Lloyd’s algorithm for K-means clustering

- Denote by $C(i)$ the cluster assigned to $x_i$.
- Lloyd’s algorithm provides a heuristic method for optimizing the K-means objective function.

Start with a “cluster centers” assignment $m_1^{(0)}, \ldots, m_K^{(0)}$. Set $t := 0$. Repeat:

- Assign each point $x_j$ to the cluster whose mean is closest to $x_j$:
  $$S_i^{(t)} := \{x_j : \|x_j - m_i^{(t)}\|^2 \leq \|x_j - m_k^{(t)}\|^2 \forall k = 1, \ldots, K\}.$$
- Compute the average $m_i^{(t+1)}$ of the observations in cluster $i$:
  $$m_i^{(t+1)} := \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j.$$
- $t \leftarrow t + 1$.

Until convergence.
Convergence of Lloyds’s algorithm

Note that Lloyds’s algorithm uses a greedy approach to sequentially minimize:

\[ \sum_{i=1}^{K} \sum_{x_j \in S_i} \|x_j - m_i\|^2. \]

- Both steps of the algorithm decrease the objective.
- Thus, Lloyds’s algorithm converges a local minimum of the objective function.

There is no guarantee that Lloyds’ algorithm will find the global optimum.

As a result, we use different starting points (i.e., different choices for the initial means \(m_i^{(0)}\)).

Common initialization methods:
- **The Forgy method**: Pick \(K\) observations at random from \(\{x_1, \ldots, x_n\}\) and use these as the initial means.
- **Random partition**: Randomly assign a cluster to each observation and compute the mean of each cluster.

Illustration of the K-means algorithm

- 100 random points in \(\mathbb{R}^2\).
- The algorithm converges in 7 iterations (with a random centers initialization).

Source: https://datasciencelab.wordpress.com
Illustration of the K-means algorithm

- 100 random points in $R^2$.
- The algorithm converges in 7 iterations (with a random centers initialization).

Source: https://datasciencelab.wordpress.com

- Assume \( \{x_1, \ldots, x_n\} \subset \mathbb{R}^p \) are iid from a distribution \( P \) on \( \mathbb{R}^p \).
- Let \( P_n \) denote the empirical measure for a sample of size \( n \).
- For a given probability measure \( Q \) on \( \mathbb{R}^p \), and any set \( A \subset \mathbb{R}^p \), let
  \[
  \Phi(A, Q) := \int \min_{a \in A} \|x - a\|^2 \, dQ(x),
  \]
  and let
  \[
  m_k(Q) := \inf\{\Phi(A, Q) : A \text{ contains } k \text{ or fewer points}\}.
  \]
- For a given \( k \), the set \( A_n = A_n(k) \) of optimal cluster centers is chosen to satisfy
  \[
  \Phi(A_n, P_n) = m_k(P_n).
  \]
- Let \( A = A(k) \) satisfy
  \[
  \Phi(A, P) = m_k(P).
  \]

Consistency of K-means (cont.)

Theorem (Pollard, 1981)

Suppose:
- \( \int \|x\|^2 \, dP(x) < \infty \) and
- for \( j = 1, 2, \ldots, k \) there is a unique set \( A(j) \) for which
  \[
  \Phi(A(j), P) = m_j(P).
  \]

Then \( A_n \to A(k) \) a.s. (in the Hausdorff distance), and
\[
\Phi(A_n, P_n) \to m_k(P) \text{ a.s.}
\]

Pollard’s theorem guarantees consistency under mild assumptions.

Note however, that the theorem assumes that the clustering was obtained by globally minimizing the K-means objective function (not true in applications).

Example: clustering the zip data

Is there a nice cluster structure in the zip dataset?

```python
# Load zip data
est = KMeans(n_clusters=10, verbose=1)  # Note: verbose=1 is just to
# see what sklearn is doing...
est.fit(X_train)

Prop_mat = np.zeros((10,10))  # Percentage of label i that is digit j
for i in range(10):
    N_i = np.sum(est.labels_ == i)  # Number of samples with label i
    for j in range(10):
        Prop_mat[i,j] = np.sum(y_train[est.labels_ == i] == j)/np.double(N_i)*100

Prop_mat =

0.00 0.80 2.45 0.38 0.94 0.57 0.00 83.96 0.19 11.51
14.78 0.00 0.77 0.26 0.77 14.66 00.00 0.00 0.00 0.00
50.37 0.00 2.28 0.18 0.18 12.19 06.10 0.00 0.70 0.00
88.96 0.00 0.51 0.34 0.00 2.72 7.13 0.00 0.34 0.00
1.98 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
63.61 0.00 5.66 1.13 62.23 5.66 1.41 3.25 1.41 17.82
1.63 0.00 3.85 53.32 0.00 12.00 0.00 0.00 3.25 0.22
0.00 93.03 0.37 0.09 3.90 0.00 0.84 0.28 1.02 0.46
0.00 0.12 1.10 1.46 16.93 0.61 0.24 20.46 4.90 54.08
```

Example: clustering the zip data (cont.)