1 Introduction

Main question: Can we learn the hidden structure in (unlabeled) data?

Problems in USL:

- Clustering: partitioning of data into a finite number of typically disjoint (or overlapping) groups
- Projection/Dimensionality Reduction: project high-dimensional data to lower-dimensional space

Use cases:

- data analysis, data summarization/exploration/visualization
- feature selection for supervised learning (make $d$ smaller)
- data compression
- data/image denoising
- prototype learning (cf. RBF networks, or to make kernel methods more tractable for big data)
- recommendation systems
- topic modeling

Techniques:

1. Clustering: $k$-means, Gaussian Mixture Models (GMMs)
2. Dimensionality Reduction: PCA/ SVD, spectral methods

2 Clustering

Goal: partition data points $\{x_i\}_{i=1}^n$ into $k$ groups (clusters) such that points within a cluster are close and points in different clusters are far away.

Given: $D = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$

Task: identify clusters and cluster assignment for each point.
2.1  

$k$-means Clustering

$k$-means is one of the simplest and arguably the most popular clustering algorithm.

Assumptions:

- number of clusters $k$ is known
- each input can only belong to a single cluster

Notation:

- cluster assignment $z_i = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \end{pmatrix} \in \mathbb{R}^k$ is one-hot encoding and the index $\alpha$ of the “1” means the $i$-th point belongs to $\alpha$-th cluster, with $i = 1, \ldots, n$ and $\alpha = 1, \ldots, k$
- cluster representative/prototype is the cluster center (also cluster mean):

\[
\mu_\alpha = \frac{\sum_{i=1}^{n} [z_i]_\alpha x_i}{\sum_{i=1}^{n} [z_i]_\alpha}
\]  

2.1.1 Objective and Algorithm

We aim to minimize the distance of each input point to its closest cluster center leading to the following objective function:

\[
D(\mu_1, \ldots, \mu_k, z_1, \ldots, z_n) = \sum_{i=1}^{n} \sum_{\alpha=1}^{k} [z_i]_\alpha d_{i\alpha},
\]

where $d_{i\alpha}$ is the distance of the $i$-th input point to cluster $\alpha$, for instance $d_{i\alpha} = ||x_i - \mu_\alpha||^2$. Unfortunately, this objective is non-convex and non-continuous and therefore hard to optimize.

So, we have to resort to an approximation solution for instance by performing alternate minimization. This algorithm known as $k$-means clustering is stated in Algorithm 1. The first for loop updates cluster assignment and the second for loop updates cluster centers.

Algorithm 1  $k$-means clustering

repeat
  for all $i = 1, \ldots, n$ do
    \[
    [z_i]_\alpha = \begin{cases} 
    1 & \text{if } \alpha = \arg \min_{\alpha} ||x_i - \mu_\alpha||^2 \\
    0 & \text{otherwise}
    \end{cases}
    \]
  end for
  for all $\alpha = 1, \ldots, k$ do
    \[
    \mu_\alpha = \frac{\sum_{i=1}^{n} [z_i]_\alpha x_i}{\sum_{i=1}^{n} [z_i]_\alpha}
    \]
  end for
until convergence
2.1.2 Visualization of the $k$-means Algorithm

Figure 2 illustrates the $k$-means algorithm using a data set of eruptions of the Old Faithful geyser in Yellowstone NP. The features are *eruption time* (in min on x-axis) and *waiting time to next eruption* (in min on y-axis).

![Image of $k$-means algorithm](image)

Interpreting the results

Since clustering is inherently unsupervised, we typically have to interpret the results and verify if it makes sense. In our example application, the two clusters reveal that there are two series of eruptions of the Old Faithful geyser: eruptions with short intervals and eruptions with long intervals. The eruptions with long intervals last longer than short interval eruptions, most likely because longer eruptions require more effort than short interval discharges. Furthermore, the geyser has a higher number of long eruptions than shorter eruptions and the variance in that cluster is higher. Now, we leave it up to the geologists to interpret and use this result in more depth.
2.1.3 Choosing the Number of Clusters $k$

Typically, the number of clusters $k$ is unknown. So, we could pick the $k$ that leads to the minimal value of our objective function $D$, cf. Eq. (2). However, $D$ always decreases as $k$ increases.\footnote{Note that $D$ is of course minimized at the trivial and uninteresting case of $k = n$, where each input has its own cluster.}

So, in order to find an appropriate $k$ we can use the following strategy: apply $k$-means repeatedly with an increasing value of $k$ and observe the value of the objective function $D$ (ideally average over several different initialization). Now, plot the values for $D$ and find the “kink” in the curve.

![Figure 3: A 2d data set with two inherent clusters](image)

Figure 3: A 2d data set with two inherent clusters

Figure 3 shows a dataset with two inherent clusters and how the objective $D$, cf. Eq. (2), varies as $k$ increases. We can see that the objective value $D$ will always decrease (on average) as $k$ increases, however, the amount of additional improvement diminishes once the “true” underlying number of clusters has been reached. This effect leads to a kink in the decreasing curve of $D$ and points us to a good value for $k$.

Figure 4 is another 2d toy data set with four inherent clusters.

![Figure 4: A 2d data set with four inherent clusters.](image)

2.1.4 Applications of $k$-means

- use cluster centers as data representatives/prototypes as illustrated in Fig. 5 (e.g. in RBF networks)
- use clusters as compressed information as illustrated in Fig. 6 (cf. vector quantization)
Figure 5: The $k = 20$ cluster centers of k-means applied to a data set of handwritten digits. The cluster centers consist of the average image in each cluster and are not actual data points in the original data set. However, they clearly show that the data set has strong cluster structure defined by the different digit types.

Figure 6: An example application of $k$-means used for color reduction. The image pixels are represented in a 3d space (red,green,blue) and clustered with $k$-means. After clustering all pixels are substituted by their corresponding cluster centers, effectively reducing the number of different colors to only $k$ colors.
2.1.5 Summary

+ easy to implement
+ fast for small \(k\)
+ easily parallelizable
- can be dominated by outliers and initialization
- only allows hard cluster assignments
- works well when clusters are spherical, cf. Figure 7 for an illustration

Figure 7: \(k\)-means works well for spherical clusters (left), but fails to give good results for oblong or elliptical clusters (right) [PDSh Ch5].

To overcome this last drawback, we will now discuss two generalizations of the \(k\)-means model: kernel \(k\)-means and Gaussian mixture model clustering.

2.2 Kernel \(k\)-means

If our input data has non-convex clusters as for instance the illustrated in Figure 8, we can apply the same trick as for supervised learning, kernelize the learning method. To kernelize \(k\)-means, we need to rephrase the equations of Algorithm 1 such that \(x_i\) only appears in inner products. Then, all inner products can be replaced by a kernel function.

This works out nicely for the cluster assignment in Eq. (3):

\[
d_{i\alpha} = \left\| x_i - \mu_\alpha \right\|^2 = (x_i - \mu_\alpha)^T (x_i - \mu_\alpha) = (x_i - \frac{\sum_{j=1}^n [z_j]_\alpha x_j}{\sum_{j=1}^n [z_j]_\alpha})^T (x_i - \frac{\sum_{j=1}^n [z_j]_\alpha x_j}{\sum_{j=1}^n [z_j]_\alpha})
\]

\[
= \frac{1}{\left\langle \phi(x_i), \phi(x_i) \right\rangle} - \frac{2}{\sum_{j=1}^n [z_j]_\alpha} \sum_{j=1}^n [z_j]_\alpha \left\langle \phi(x_i), \phi(x_j) \right\rangle + \frac{1}{\left\langle \sum_{j=1}^n [z_j]_\alpha, \sum_{j=1}^n [z_j]_\alpha \right\rangle^2} \sum_{j=1}^n \sum_{l=1}^n [z_j]_\alpha [z_l]_\alpha \left\langle \phi(x_j), \phi(x_l) \right\rangle
\] (5)
However, in Eq. (4) the **cluster centers** $\mu_\alpha$ cannot be computed for infinite dimensional $\phi(x)$:

$$
\mu_\alpha = \frac{\sum_{i=1}^{n} [z_i]_\alpha \phi(x_i)}{\sum_{i=1}^{n} [z_i]_\alpha}
$$

(6)

So, we have to come up with an alternative $k$-means algorithm that does not compute the cluster centers. Instead, it computes the closest cluster $\alpha$ for each data point $x_i$ based on the minimal $d_{i\alpha}$, cf. Algorithm 2.

**Algorithm 2 Kernel $k$-means**

Randomly initialize $z_1, \ldots, z_n$ (or use result of standard $k$-means)

repeat

for all $i$ & $\alpha$

$d_{i\alpha} \leftarrow$ Eq. (5) with $\langle \phi(x_i), \phi(x_j) \rangle = k(x_i, x_j)$

end for

update $z_i$ using $\alpha = \arg \min_{\alpha} d_{i\alpha}$

if $z_i$ do not change then

STOP

end if

until convergence

Note, that now we can cluster non-vectorial data (text, graphs, molecules, ...). We only need to be able to compute kernel values among the data objects.
2.3 Gaussian Mixture Model Clustering

The Gaussian Mixture Model (GMM) clustering can be thought of as a generalization of the \( k \)-means algorithm for non-spherical clusters using soft cluster assignments, cf. Figure 9.

![Figure 9: Illustration of gmm for spherical clusters (left) and non-spherical clusters (right) [PDSh Ch5].](image)

2.3.1 Setup

Define \([z_i]_\alpha \in [0,1]\) as the probability that \(x_i\) belongs to cluster \(\alpha\).

\[
\sum_{\alpha=1}^{k} [z_i]_\alpha = 1
\]  

⇒ soft cluster assignments

Assumption: dataset \(D\) comes from a mixture of Gaussians

⇒ points in each cluster \(c_\alpha\) come from a different multivariate Gaussian:

\[
x_i \sim \mathcal{N}(\mu_\alpha, \Sigma_\alpha) = p(x_i \mid c_\alpha)
\]  

⇒ cluster center = mean of the Gaussian \(\mu_\alpha\)
⇒ “radius” = covariance of the Gaussian \(\Sigma_\alpha\)

The probability density function (PDF) of the Gaussian Mixture Model (GMM) is given by:

\[
p(x \mid \pi, \{\mu, \Sigma\}_{\alpha=1}^{k}) = \sum_{\alpha=1}^{k} \pi_\alpha \mathcal{N}(x \mid \mu_\alpha, \Sigma_\alpha)
\]  

where \(\pi_\alpha\) are the mixing coefficients acting as the prior probability of each cluster, \(\pi_\alpha \in [0,1]\) and \(\sum_\alpha \pi_\alpha = 1\). Figs. 10 and 11 illustrate a GMM on 1D and 2D data respectively.
2.3.2 Generative Model and GMM Algorithm

GMMs are generative models, meaning that we can model how the points in $D$ are generated:

- pick a cluster $\alpha$ according to some probability $\pi$ over $k$ categories.
- pick a data point from $N(\mu_{\alpha}, \Sigma_{\alpha})$

To derive GMM clustering, we can now model the cluster assignments using this generative model:

$$[z_i]_\alpha = p(c_\alpha \mid x_i) = \frac{p(x_i \mid c_\alpha)p(c_\alpha)}{\sum_l p(x_i \mid c_l)p(c_l)} = \frac{p(x_i \mid \mu_{\alpha}, \Sigma_{\alpha})\pi_{\alpha}}{\sum_l p(x_i \mid \mu_l, \Sigma_l)\pi_l}$$

(10)

where $[z_i]$ is a random variable that is never observed, which thus is also called a hidden/latent variable.

→ to get the cluster assignments $\{z_i\}_i$, we need to estimate the following parameters: $\pi_{\alpha}, \mu_{\alpha}, \Sigma_{\alpha} \forall \alpha$

→ to estimate the parameters $\pi_{\alpha}, \mu_{\alpha}, \Sigma_{\alpha} \forall \alpha$ using the sample mean, covariance and sample cluster probability estimates, we need the $z_i$’s.

To resolve this issue, we start by fixing the parameters and computing the cluster assignment. Then we update the parameters based on the new clustering and alternate these steps until convergence leading to the GMM clustering algorithm as given in Algorithm 3. This algorithm is also called Expectation-Maximization (EM) with two steps, an (E) step and an (M) step:
(E) step calculates cluster membership probabilities by using the (E)xpectation of the log-likelihood evaluated using the current estimate for the parameters.

(M) step estimates the parameters by (M)aximizing the expected log-likelihood.

Algorithm 3 GMM Algorithm

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialize $\mu_1, \ldots, \mu_k$ to be distinct random data points in $D$</td>
<td></td>
</tr>
<tr>
<td>Initialize $\Sigma_1, \ldots, \Sigma_k$ to $\sigma^2 I$ for any $\sigma &gt; 0$</td>
<td></td>
</tr>
<tr>
<td>Initialize $\pi_1, \ldots, \pi_k$ to $\frac{1}{k}$ (uniform prior)</td>
<td></td>
</tr>
<tr>
<td>repeat</td>
<td></td>
</tr>
<tr>
<td>for all $i$ &amp; $\alpha$ do</td>
<td></td>
</tr>
<tr>
<td>E-step: $[z_i]<em>\alpha = \frac{p(x_i \mid \mu</em>\alpha, \Sigma_\alpha)\pi_\alpha}{\sum_i p(x_i \mid \mu_i, \Sigma_i)\pi_i}$ (11)</td>
<td></td>
</tr>
<tr>
<td>end for</td>
<td></td>
</tr>
<tr>
<td>for $\alpha = 1, \ldots, k$ do</td>
<td></td>
</tr>
<tr>
<td>M-step: $\mu_\alpha = \frac{\sum_{i=1}^n [z_i]<em>\alpha x_i}{\sum</em>{i=1}^n [z_i]_\alpha}$ (12)</td>
<td></td>
</tr>
<tr>
<td>$\Sigma_\alpha = \frac{\sum_{i=1}^n [z_i]<em>\alpha (x_i - \mu</em>\alpha)(x_i - \mu_\alpha)^\top}{\sum_{i=1}^n [z_i]_\alpha}$ (13)</td>
<td></td>
</tr>
<tr>
<td>$\pi_\alpha = \frac{1}{n} \sum_{i=1}^n [z_i]_\alpha$ (14)</td>
<td></td>
</tr>
<tr>
<td>end for</td>
<td></td>
</tr>
<tr>
<td>until convergence</td>
<td></td>
</tr>
</tbody>
</table>

2.3.3 Summary and Remarks

- $k$-means is a special case with $\Sigma_\alpha = I \ \forall \alpha$
- GMM clustering is a latent variable model (LVM) since the $z_i$ are variables that are never observed (latent variables)
- EM actually performs maximum likelihood estimation (MLE) to estimate the parameters for a model that depends on unobserved/latent variables. The likelihood $\ell$ that is maximized is given as:

$$\ell(D) = p(D \mid \pi, \{\mu_\alpha, \Sigma_\alpha\}_{\alpha=1}^k) = \prod_{i=1}^n \sum_{\alpha=1}^k \pi_\alpha p(x_i \mid \mu_\alpha, \Sigma_\alpha)$$

$$\Rightarrow$$

maximize log likelihood w.r.t. $\mu_\alpha$ to get

$$\mu_\alpha = \frac{\sum_{i=1}^n [z_i]_\alpha x_i}{\sum_{i=1}^n [z_i]_\alpha}$$

Note, that intuitively $\mu_\alpha$ is the weighted average of points in cluster $\alpha$. Mathematically it is the MLE solution that can be derived by taking the derivatives of $\ell$ with respect to the $\mu_\alpha$’s and setting them to zero. $\Sigma_\alpha$ and $\pi_\alpha$ can be derived the same way. See FCML 6.3.3 for the full derivation!