Introduction to Unsupervised Learning

Motivation — The goal of unsupervised learning is to find hidden patterns in unlabeled data \( \{x(1),...,x(m)\}\).

Jensen’s inequality — Let \( f \) be a convex function and \( X \) a random variable. We have the following inequality:
\[
E[f(X)] \geq f(E[X]) \geq f(E[f(X)])
\]

Clustering

Expectation-Maximization

Latent variables — Latent variables are hidden/unobserved variables that make estimation problems difficult, and are often denoted \( z \). Here are the most common settings where there are latent variables:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Latent variable ( z )</th>
<th>( x \mid z )</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mixture of ( k ) Gaussians</td>
<td>Multinomial(( \phi ))</td>
<td>( N(\mu_j,\Sigma_j) )</td>
<td>( \mu_j \in \mathbb{R}^n, \phi \in \mathbb{R}^k )</td>
</tr>
<tr>
<td>Factor analysis</td>
<td>( N(0,I) )</td>
<td>( N(\mu+\Lambda z,\psi) )</td>
<td>( \mu_j \in \mathbb{R}^n, \psi \in \mathbb{R}^k )</td>
</tr>
</tbody>
</table>

Algorithm — The Expectation-Maximization (EM) algorithm gives an efficient method at estimating the parameter \( \theta \) through maximum likelihood estimation by repeatedly constructing a lower-bound on the likelihood (E-step) and optimizing that lower bound (M-step) as follows:

- **E-step**: Evaluate the posterior probability \( Q(zi)Q(z(i)) \) that each data point \( x(i) \mid x(i) \) came from a particular cluster \( z(i) \mid z(i) \) as follows:
  \[
  Q(zi) = P(zi) \mid x(i) ; \theta \) \cdot Q(z(i)) = P(z(i) \mid x(i) ; \theta)
  \]

- **M-step**: Use the posterior probabilities \( Q(zi)Q(z(i)) \) as cluster specific weights on data points \( x(i) \mid x(i) \) to separately re-estimate each cluster model as follows:

\[
\theta = \arg \max \theta \sum Q(z(i))Q(z(i)) \log(P(x(i), z(i) ; \theta)) dz(i)
\]
**kk-means clustering**

We note $c(i)$ the cluster of data point $i$ and $\mu_j$ the center of cluster $j$.

**Algorithm** — After randomly initializing the cluster centroids $\mu_1, \mu_2, \ldots, \mu_k \in \mathbb{R}^n$, the kk-means algorithm repeats the following step until convergence:

$$c(i) = \arg\min_j ||x(i) - \mu_j||^2$$

and

$$\mu_j = \frac{1}{\sum_{i=1}^m 1_{c(i)=j}} \sum_{i=1}^m x(i) 1_{c(i)=j}$$

**Distortion function** — In order to see if the algorithm converges, we look at the distortion function defined as follows:

$$J(c, \mu) = \sum_{i=1}^m ||x(i) - \mu_{c(i)}||^2$$

**Hierarchical clustering**

**Algorithm** — It is a clustering algorithm with an agglomerative hierarchical approach that build nested clusters in a successive manner.

**Types** — There are different sorts of hierarchical clustering algorithms that aims at optimizing different objective functions, which is summed up in the table below:

<table>
<thead>
<tr>
<th>Ward linkage</th>
<th>Average linkage</th>
<th>Complete linkage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimize within cluster distance</td>
<td>Minimize average distance between cluster pairs</td>
<td>Minimize maximum distance of between cluster pairs</td>
</tr>
</tbody>
</table>

**Clustering assessment metrics**

In an unsupervised learning setting, it is often hard to assess the performance of a model since we don't have the ground truth labels as was the case in the supervised learning setting.

**Silhouette coefficient** — By noting $a$ and $b$ the mean distance between a sample and all other points in the same class, and between a sample and all other points in the next nearest cluster, the silhouette coefficient $s$ for a single sample is defined as follows:

$$s = b - \max(a, b)$$

**Calinski-Harabaz index** — By noting $k$ the number of clusters, $B_k$ and $W_k$ the between and within-clustering dispersion matrices respectively defined as
Independent component analysis

**Algorithm** — The Principal Component Analysis (PCA) procedure is a dimension reduction technique that projects the data on \( kk \) dimensions by maximizing the variance of the data as follows:

1. **Step 1:** Normalize the data to have a mean of 0 and standard deviation of 1.
   \[
   x(i) = x(i) - \mu_j \text{ and } \sigma_j^2 = \sum_{i=1}^{m} x(i)^2 \text{ and } \sigma_j^2 = \sum_{i=1}^{m} (x(i) - \mu_j)^2
   \]
2. **Step 2:** Compute \( \Sigma = \sum_{i=1}^{m} x(i)x(i)^T \in \mathbb{R}^{n \times n} \), which is symmetric with real eigenvalues.
3. **Step 3:** Compute \( u_1, ..., u_k \in \mathbb{R}^n \) the \( k \) orthogonal principal eigenvectors of \( \Sigma \), i.e. the orthogonal eigenvectors of the \( k \) largest eigenvalues.
4. **Step 4:** Project the data on \( \text{span}(u_1, ..., u_k) \text{span}(u_1, ..., u_k) \).

This procedure maximizes the variance among all \( k \)-dimensional spaces.

**Remark:** the eigenvector associated with the largest eigenvalue is called principal eigenvector of matrix \( AA \).

**Principal component analysis**

It is a dimension reduction technique that finds the variance maximizing directions onto which to project the data.

**Eigenvalue, eigenvector** — Given a matrix \( A \in \mathbb{R}^{n \times n} \), \( \lambda \) is said to be an eigenvalue of \( AA \) if there exists a vector \( z \in \mathbb{R}^n \backslash \{0\} \), called eigenvector, such that we have:

\[
Az = \lambda z
\]

**Spectral theorem** — Let \( A \in \mathbb{R}^{n \times n} \). If \( AA \) is symmetric, then \( AA \) is diagonalizable by a real orthogonal matrix \( U \in \mathbb{R}^{n \times n} \). By noting \( \Lambda = \text{diag}(\lambda_1, ..., \lambda_n) \Lambda = \text{diag}(\lambda_1, ..., \lambda_n) \), we have:

\[
\exists \Lambda \text{ diagonal, } A = U \Lambda U^T \Lambda \text{ diagonal, } A = U \Lambda U^T
\]

**Dimension reduction**

It is a technique meant to find the underlying generating sources.
**Assumptions** — We assume that our data $x$ has been generated by the $n$-dimensional source vector $s=(s_1, ..., s_n)s=(s_1, ..., s_n)$, where $s_i$ are independent random variables, via a mixing and non-singular matrix $A$ as follows:

$$x=As$$

The goal is to find the unmixing matrix $W=A^{-1}$. 

**Bell and Sejnowski ICA algorithm** — This algorithm finds the unmixing matrix $W$ by following the steps below:

- Write the probability of $x=Ws=As=As$ as:
  $$p(x)=n!ps(wTix)^{-1}W|p(x)=n!ps(wTix)^{-1}W$$

- Write the log likelihood given our training data $\{x(i), i \in [1,m]\}$ and by noting $g$ the sigmoid function as:
  $$l(W)=m\sum_{i=1}^{m}(n\sum_{j=1}^{n}\log (g'(w_jTix(i))))+\log |W|$$

Therefore, the stochastic gradient ascent learning rule is such that for each training example $x(i)x(i)$, we update $W$ as follows:

$$W\leftarrow W+\alpha \left(1-2g(w_1Tix(i))1-2g(w_2Tix(i)) \cdots 1-2g(w_nTix(i)) \sum_{i=1}^{m}(x(i)T+(WT)-1)\right)$$

$$W\leftarrow W+\alpha (1-2g(w_1Tix(i))1-2g(w_2Tix(i)) \cdots 1-2g(w_nTix(i)))x(i)T+(WT)-1)$$