Clustering

Shishir K. Shah


Clustering is a technique for finding similarity groups in data, called clusters. i.e.,
- it groups data instances that are similar to (near) each other in one cluster and data instances that are very different (far away) from each other into different clusters.
- Clustering is often called an unsupervised learning task as no class values denoting an a priori grouping of the data instances are given.
An illustration

- The data set has three natural groups of data points, i.e., 3 natural clusters.

What is clustering for?

- Let us see some real-life examples
- **Example 1**: groups people of similar sizes together to make “small”, “medium” and “large” T-Shirts.
  - Tailor-made for each person: too expensive
  - One-size-fits-all: does not fit all.
- **Example 2**: In marketing, segment customers according to their similarities
  - To do targeted marketing.
What is clustering for? (cont…)

- **Example 3**: Given a collection of text documents, we want to organize them according to their content similarities,
  - To produce a topic hierarchy
- **In fact, clustering is one of the most utilized data mining techniques.**
  - It has a long history, and used in almost every field, e.g., medicine, psychology, botany, sociology, biology, archeology, marketing, insurance, libraries, etc.

Outline of discussion

- **Fundamental clustering algorithm**
  - k-means
  - mean shift
  - EM
- **Different ways of measuring similarity**
- **Measure validity of clusters**
  - How can we tell the generated clusters are good?
  - How can we judge if the clusters are biologically meaningful?
K-means clustering

- K-means is a **partitional clustering** algorithm.
- Let the set of data points (or instances) $D$ be
  \[ \{x_1, x_2, \ldots, x_n\}, \]
  where $x_i = (x_{i1}, x_{i2}, \ldots, x_{ir})$ is a **vector** in a real-valued space $X \subseteq \mathbb{R}^r$, and $r$ is the number of attributes (dimensions) in the data.
- The $k$-means algorithm partitions the given data into $k$ clusters.
  - Each cluster has a cluster **center**, called **centroid**.
  - $k$ is specified by the user.

K-means algorithm

- Given $k$, the **k-means** algorithm works as follows:
  1) Randomly choose $k$ data points (seeds) to be the initial **centroids**, cluster centers
  2) Assign each data point to the closest **centroid**
  3) Re-compute the **centroids** using the current cluster memberships.
  4) If a convergence criterion is not met, go to 2).
**K-means algorithm – (cont …)**

Algorithm $k$-means($k, D$)

1. Choose $k$ data points as the initial centroids (cluster centers)
2. repeat
   3. for each data point $x \in D$ do
      4. compute the distance from $x$ to each centroid;
      5. assign $x$ to the closest centroid // a centroid represents a cluster
   6. endfor
3. re-compute the centroids using the current cluster memberships
4. until the stopping criterion is met

**Stopping/convergence criterion**

1. no (or minimum) re-assignments of data points to different clusters,
2. no (or minimum) change of centroids, or
3. minimum decrease in the sum of squared error (SSE),

$$SSE = \sum_{j=1}^{k} \sum_{x \in C_j} \text{dist}(x, m_j)^2$$  \hspace{1cm} (1)$$

- $C_j$ is the $j$th cluster, $m_j$ is the centroid of cluster $C_j$ (the mean vector of all the data points in $C_j$), and $\text{dist}(x, m_j)$ is the distance between data point $x$ and centroid $m_j$. 

An example

(A). Random selection of $k$ centers

Iteration 1: (B). Cluster assignment
              (C). Re-compute centroids

An example (cont ...)

Iteration 2: (D). Cluster assignment
              (E). Re-compute centroids

Iteration 3: (F). Cluster assignment
              (G). Re-compute centroids
An example distance function

The $k$-means algorithm can be used for any application data set where the mean can be defined and computed. In the Euclidean space, the mean of a cluster is computed with:

$$m_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$$

(2)

where $|C_j|$ is the number of data points in cluster $C_j$. The distance from one data point $x_i$ to a mean (centroid) $m_j$ is computed with

$$\text{dist}(x_i, m_j) = \| x_i - m_j \|$$

$$= \sqrt{(x_{i1} - m_{j1})^2 + (x_{i2} - m_{j2})^2 + \ldots + (x_{ir} - m_{jr})^2}$$

(3)

K-means for Clustering

- K-means
  - Start with a random guess of cluster centers
  - Determine the membership of each data points
  - Adjust the cluster centers
K-means for Clustering

- K-means
  - Start with a random guess of cluster centers
  - Determine the membership of each data point
  - Adjust the cluster centers
K-means

1. Ask user how many clusters they’d like. (e.g. k=5)

2. Randomly guess k cluster Center locations
K-means

1. Ask user how many clusters they’d like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it’s closest to. *(Thus each Center “owns” a set of datapoints)*
K-means

1. Ask user how many clusters they’d like. (e.g. k=5)
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it’s closest to.
4. Each Center finds the centroid of the points it owns

Any Computational Problem?

Strengths of k-means

- Strengths:
  - Simple: easy to understand and to implement
  - Efficient: Time complexity: $O(tkn)$, where $n$ is the number of data points, $k$ is the number of clusters, and $t$ is the number of iterations.
  - Since both $k$ and $t$ are small. k-means is considered a linear algorithm.
- K-means is the most popular clustering algorithm.
- Note that: it terminates at a local optimum if SSE is used. The global optimum is hard to find due to complexity.
Weaknesses of k-means:

- The algorithm is only applicable if the mean is defined.
  - For categorical data, k-mode - the centroid is represented by most frequent values.
- The user needs to specify $k$.
- The algorithm is sensitive to outliers.
  - Outliers are data points that are very far away from other data points.
  - Outliers could be errors in the data recording or some special data points with very different values.

Weaknesses of k-means: Problems with outliers

(A): Undesirable clusters

(B): Ideal clusters
Weaknesses of k-means: To deal with outliers

- One method is to remove some data points in the clustering process that are much further away from the centroids than other data points.
  - To be safe, we may want to monitor these possible outliers over a few iterations and then decide to remove them.
- Another method is to perform random sampling. Since in sampling we only choose a small subset of the data points, the chance of selecting an outlier is very small.
  - Assign the rest of the data points to the clusters by distance or similarity comparison, or classification

Weaknesses of k-means (cont …)

- The algorithm is sensitive to initial seeds.
Weaknesses of k-means (cont …)

- If we use different seeds: good results

There are some methods to help choose good seeds

(A). Random selection of $k$ seeds (centroids)

(B). Iteration 1

(C). Iteration 2

---

Weaknesses of k-means (cont …)

- The $k$-means algorithm is not suitable for discovering clusters that are not hyper-ellipsoids (or hyper-spheres).

(A). Two natural clusters

(B). $k$-means clusters
Improve K-means

- Group points by region
  - KD tree
  - SR tree
- Key difference
  - Find the closest center for each rectangle
  - Assign all the points within a rectangle to one cluster

Improved K-means

- Find the closest center for each rectangle
- Assign all the points within a rectangle to one cluster
Improved K-means
Improved K-means
Improved K-means
Improved K-means
Improved K-means

Mean Shift

- Mean is a good estimate for data that is Gaussian distributed: what about non-Gaussian distributed data?
- Given a distribution of points, mean shift is a procedure for finding the densest region.
- Example for simple 2D case:
  - Start from arbitrary point in the distribution
  - Region of interest is a circle centered in this point
  - On each iteration find the center of the mass for the region of interest
  - Move the circle to the center of the mass
  - Continue the iterations until convergence
Intuitive Description

Objective: Find the densest region
Distribution of identical billiard balls
Objective: Find the densest region
Distribution of identical billiard balls
Intuitive Description

Objective: Find the densest region
Distribution of identical billiard balls
**Intuitive Description**

Objective: Find the densest region
Distribution of identical billiard balls

**What is Mean Shift?**

A tool for:
Finding modes in a set of data samples, manifesting an underlying probability density function (PDF) in \( \mathbb{R}^n \)

PDF in feature space
- Color space
- Word space
- Actually any feature space you can conceive
- ...

Data

Non-parametric Density GRADIENT Estimation (Mean Shift)
Non-Parametric Density Estimation

Assumption: The data points are sampled from an underlying PDF

Data point density implies PDF value!

Assumed Underlying PDF  Real Data Samples

Non-Parametric Density Estimation

Assumed Underlying PDF  Real Data Samples
**Non-Parametric Density Estimation**

Assumed Underlying PDF  
Real Data Samples

---

**Parametric Density Estimation**

Assumption: The data points are sampled from an underlying PDF

$$PDF(x) = \sum_{i} c_i \cdot e^{-\frac{(x-\mu_i)^2}{2\sigma_i^2}}$$

Assumed Underlying PDF  
Real Data Samples
Kernel Density Estimation
Parzen Windows - Function Forms

\[ P(x) = \frac{1}{n} \sum_{i=1}^{n} K(x - x_i) \]

A function of some finite number of data points \( x_1, \ldots, x_n \)

In practice one uses the forms:

\[ K(x) = c \prod_{i=1}^{d} k(x_i) \quad \text{or} \quad K(x) = c k(\|x\|) \]

Same function on each dimension \hspace{1cm} Function of vector length only

Kernel Density Estimation
Various Kernels

\[ P(x) = \frac{1}{n} \sum_{i=1}^{n} K(x - x_i) \]

A function of some finite number of data points \( x_1, \ldots, x_n \)

Examples:

• Epanechnikov Kernel

\[ K_\xi(x) = \begin{cases} c \left(1 - \|x\|^2\right) & \text{if } \|x\| \leq 1 \\ 0 & \text{otherwise} \end{cases} \]

• Uniform Kernel

\[ K_U(x) = \begin{cases} c & \|x\| \leq 1 \\ 0 & \text{otherwise} \end{cases} \]

• Normal Kernel

\[ K_N(x) = c \cdot \exp \left(-\frac{1}{2} \|x\|^2\right) \]
Kernel Density Estimation

**Gradient**

\[ \nabla P(x) = \frac{1}{n} \sum_{i=1}^{n} \nabla K(x - x_i) \]

*Give up estimating the PDF! Estimate **ONLY** the gradient*

Using the Kernel form:

\[ K(x - x_i) = c_k \left( \frac{|x - x_i|}{\delta} \right) \]

Size of window

We get:

\[ \nabla P(x) = \frac{C}{n} \sum_{i=1}^{n} \nabla k_i = \frac{C}{n} \left[ \sum_{i=1}^{n} g_i \right] g \left[ \frac{\sum_{i=1}^{n} x_i g_i}{\sum_{i=1}^{n} g_i} - x \right] \]

**Computing The Mean Shift**

\[ \nabla P(x) = \frac{C}{n} \sum_{i=1}^{n} \nabla k_i = \frac{C}{n} \left[ \sum_{i=1}^{n} g_i \right] g \left[ \frac{\sum_{i=1}^{n} x_i g_i}{\sum_{i=1}^{n} g_i} - x \right] \]

*\( g(x) = -k'(x) \)
Computing The Mean Shift

Simple Mean Shift procedure:
- Compute mean shift vector

\[ m(x) = \frac{\sum_{i=1}^{n} x_i g_i \left( \frac{x-x_i}{h} \right)}{\sum_{i=1}^{n} g_i \left( \frac{x-x_i}{h} \right)} - x \]

- Translate the Kernel window by \( m(x) \)

Mean Shift Mode Detection

Updated Mean Shift Procedure:
- Find all modes using the Simple Mean Shift Procedure
- Prune modes by perturbing them (find saddle points and plateaus)
- Prune nearby – take highest mode in the window

What happens if we reach a saddle point?
Perturb the mode position and check if we return back
**Mean Shift Properties**

- Automatic convergence speed – the mean shift vector size depends on the gradient itself.
- Near maxima, the steps are small and refined
- Convergence is guaranteed for infinitesimal steps only \( \Rightarrow \) infinitely convergent, (therefore set a lower bound)
- For Uniform Kernel ( ), convergence is achieved in a finite number of steps
- Normal Kernel ( ) exhibits a smooth trajectory, but is slower than Uniform Kernel ( ).

**Real Modality Analysis**

Tessellate the space with windows
Run the procedure in parallel
Real Modality Analysis
An example

The blue data points were traversed by the windows towards the mode.

Window tracks signify the steepest ascent directions.
Mean Shift Strengths & Weaknesses

**Strengths:**
- Application independent tool
- Suitable for real data analysis
- Does not assume any prior shape (e.g. elliptical) on data clusters
- Can handle arbitrary feature spaces
- Only ONE parameter to choose
- $h$ (window size) has a physical meaning, unlike K-Means

**Weaknesses:**
- The window size (bandwidth selection) is not trivial
- Inappropriate window size can cause modes to be merged, or generate additional “shallow” modes ➔ Use adaptive window size

A Gaussian Mixture Model for Clustering

- Assume that data are generated from a mixture of Gaussian distributions
- For each Gaussian distribution
  - Center: $\mu_i$
  - Variance: $\Sigma_i$ (ignore)
- For each data point
  - Determine membership
  $z_{ij}$: if $x_j$ belongs to j-th cluster
Learning a Gaussian Mixture
(with known covariance)

- Probability
  \[ p(x = x_i) = \sum_{\mu_j} p(x = x_i, \mu = \mu_j) = \sum_{\mu_j} p(\mu = \mu_j) p(x = x_i | \mu = \mu_j) \]

- Log-likelihood of data
  \[ \sum_{i} \log p(x = x_i) = \sum_{i} \log \left( \sum_{\mu_j} p(\mu = \mu_j) \frac{1}{(2\pi\sigma^2)^{d/2}} \exp \left( -\frac{\|x_i - \mu_j\|_2^2}{2\sigma^2} \right) \right) \]

- Apply MLE to find optimal parameters \( \{p(\mu = \mu_j), \mu_j\}_j \)
Expectation Maximization (EM)

- A more general probabilistic method for maximizing likelihood in the presence of missing data (i.e., assignments of points to lines)

\[
- \log L(\theta) = \sum_k \sum_{i \in k} (a_k x_i + b_k y_i - d_k)^2
\]
**Expectation Maximization (EM)**

- We want to maximize the likelihood of the data points given parameters $\theta = \{a_k, b_k, d_k\}$ of our lines:
  \[
  -\log L(\theta) = \sum_k \sum_{i \in l_k} (a_k x_i + b_k y_i - d_k)^2
  \]
- Missing data: assignment of points $i$ to lines $l_k$
- Iterate:
  - Guess missing data using current model parameters
  - Form maximum likelihood estimate of parameters using filled-in values for missing data
- Technical detail: need to take the expectation with respect to the distribution of the missing data
  - The resulting procedure is guaranteed to increase likelihood at every iteration

\[
\sum_{i} \sum_{l \in l_k} (a_k x_i + b_k y_i - d_k)^2
\]

**The Gaussian Distribution**

- Multivariate Gaussian
  \[
  \mathcal{N}(x|\mu, \Sigma) = \frac{1}{(2\pi)^{D/2} |\Sigma|^{1/2}} \exp \left\{ -\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu) \right\}
  \]

- Maximum likelihood
  \[
  \mu_{ML} = \frac{1}{N} \sum_{n=1}^{N} x_n
  \]
  \[
  \Sigma_{ML} = \frac{1}{N} \sum_{n=1}^{N} (x_n - \mu_{ML})(x_n - \mu_{ML})^T
  \]

Slide credit: C. Bishop
Gaussian Mixtures

- Linear super-position of Gaussians
  \[ p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k) \]

- Normalization and positivity require
  \[ \sum_{k=1}^{K} \pi_k = 1 \quad 0 \leq \pi_k \leq 1 \]

Example: Mixture of 3 Gaussians

Slide credit: C. Bishop
Maximum Likelihood for the GMM

- \log \log p(D|\pi, \mu, \Sigma) = \sum_{n=1}^{N} \log \left( \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k) \right)

- Sum over components appears \textit{inside} the log
  - no closed form ML solution

EM Algorithm – Informal Derivation

$$\ln L = \sum_n \ln \left( \sum_k \pi_k N_{kn} \right)$$

where:

$$N_{kn} = \mathcal{N}(x_n|\mu_k, \Sigma_k)$$

$$\frac{\partial \ln L}{\partial \mu_j} = \sum_n \nabla_j N_j(k) \frac{1}{N_{kn}} \frac{\partial N_{kn}}{\partial \mu_j} = 0$$

Thus:

$$0 = \sum_n \nabla_j N_j(k) (x_n - \mu_j)$$

$$\Rightarrow \text{solve for } \mu_j$$

Slide credit: C. Bishop
**EM Algorithm – Informal Derivation**

- **M step equations**
  
  \[ \mu_j = \frac{\sum_{n=1}^{N} \gamma_j(x_n)x_n}{\sum_{n=1}^{N} \gamma_j(x_n)} \]
  
  \[ \pi_j = \frac{1}{N} \sum_{n=1}^{N} \gamma_j(x_n) \]

  \[ \Sigma_j = \frac{\sum_{n=1}^{N} \gamma_j(x_n)(x_n - \mu_j)(x_n - \mu_j)^T}{\sum_{n=1}^{N} \gamma_j(x_n)} \]

**EM Algorithm – Informal Derivation**

- **E step equation**
  
  \[ \gamma_j(x) = \frac{\pi_j \mathcal{N}(x | \mu_j, \Sigma_j)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(x | \mu_k, \Sigma_k)} \]

(Slide credit: C. Bishop)
Learning a Gaussian Mixture
(with known covariance)

**E-Step**

\[ E[z_{ij}] = p(\mu = \mu_j \mid x = x_i) \]

\[ = \frac{p(x = x_i \mid \mu = \mu_j) p(\mu = \mu_j)}{\sum_{n=1}^{k} p(x = x_i \mid \mu = \mu_n) p(\mu = \mu_j)} \]

\[ = \frac{\sum_{n=1}^{k} e^{-\frac{1}{2\sigma^2}(x_i - \mu_j)^2} p(\mu = \mu_j)}{\sum_{n=1}^{k} e^{-\frac{1}{2\sigma^2}(x_i - \mu_n)^2} p(\mu = \mu_n)} \]

**M-Step**

\[ \mu_j \leftarrow \frac{1}{m} \sum_{i=1}^{m} E[z_{ij}] x_i \]

\[ p(\mu = \mu_j) \leftarrow \frac{1}{m} \sum_{i=1}^{m} E[z_{ij}] \]
Gaussian Mixture Example: Start

After First Iteration
After 2nd Iteration

After 3rd Iteration
After 6th Iteration

After 20th Iteration
Summary

- Despite weaknesses, $k$-means is still the most popular algorithm due to its simplicity, efficiency and
  - other clustering algorithms have their own lists of weaknesses.
- No clear evidence that any other clustering algorithm performs better in general
  - although they may be more suitable for some specific types of data or applications.
- Comparing different clustering algorithms is a difficult task. No one knows the correct clusters!

Distance functions

- Key to clustering: “similarity” and “dissimilarity” are commonly used terms.
- There are numerous distance functions for
  - Different types of data
    - Numeric data
    - Nominal data
  - Different specific applications
Distance functions for numeric attributes

- Most commonly used functions are
  - Euclidean distance and
  - Manhattan (city block) distance
- We denote distance with: \( dist(x_i, x_j) \), where \( x_i \) and \( x_j \) are data points (vectors)
- They are special cases of Minkowski distance. \( h \) is a positive integer.
  \[
dist(x_i, x_j) = \left( (x_{i1} - x_{j1})^h + (x_{i2} - x_{j2})^h + \cdots + (x_{ir} - x_{jr})^h \right)^{\frac{1}{h}}
\]

Euclidean distance and Manhattan distance

- If \( h = 2 \), it is the Euclidean distance
  \[
dist(x_i, x_j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \cdots + (x_{ir} - x_{jr})^2}
\]
- If \( h = 1 \), it is the Manhattan distance
  \[
dist(x_i, x_j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \cdots + |x_{ir} - x_{jr}|
\]
- Weighted Euclidean distance
  \[
dist(x_i, x_j) = \sqrt{w_1(x_{i1} - x_{j1})^2 + w_2(x_{i2} - x_{j2})^2 + \cdots + w_r(x_{ir} - x_{jr})^2}
\]
Squared distance and Chebychev distance

- **Squared Euclidean distance**: to place progressively greater weight on data points that are further apart.
  \[ \text{dist}(x_i, x_j) = (x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \ldots + (x_{ir} - x_{jr})^2 \]

- **Chebychev distance**: one wants to define two data points as "different" if they are different on any one of the attributes.
  \[ \text{dist}(x_i, x_j) = \max(|x_{i1} - x_{j1}|, |x_{i2} - x_{j2}|, \ldots, |x_{ir} - x_{jr}|) \]

Distance functions for binary and nominal attributes

- **Binary attribute**: has two values or states but no ordering relationships, e.g.,
  - Gender: male and female.
- We use a confusion matrix to introduce the distance functions/measures.
- Let the \(i\)th and \(j\)th data points be \(x_i\) and \(x_j\) (vectors)
Confusion matrix

\[
\begin{array}{c|cc}
\text{Data point } j & 1 & 0 \\
\hline
1 & a & b \\
0 & c & d \\
\hline
a+b & c+d & a+c & b+d & a+b+c+d
\end{array}
\]  

\(a\): the number of attributes with the value of 1 for both data points.
\(b\): the number of attributes for which \(x_{if} = 1\) and \(x_{jf} = 0\), where \(x_{if}(x_{jf})\) is the value of the \(f\)th attribute of the data point \(x_i\) (\(x_j\)).
\(c\): the number of attributes for which \(x_{if} = 0\) and \(x_{jf} = 1\).
\(d\): the number of attributes with the value of 0 for both data points.

Symmetric binary attributes

- A binary attribute is **symmetric** if both of its states (0 and 1) have equal importance, and carry the same weights, e.g., male and female of the attribute Gender.
- Distance function: Simple Matching Coefficient, proportion of mismatches of their values

\[
dist(x_i, x_j) = \frac{b + c}{a + b + c + d}
\]
Symmetric binary attributes: example

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>x₁</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x₂</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

\[ \text{dist}(x_i, x_j) = \frac{2 + 1}{2 + 2 + 1 + 2} = \frac{3}{7} = 0.429 \]

Asymmetric binary attributes

- **Asymmetric**: if one of the states is more important or more valuable than the other.
  - By convention, state 1 represents the more important state, which is typically the rare or infrequent state.
- **Jaccard coefficient** is a popular measure
  \[ \text{dist}(x_i, x_j) = \frac{b + c}{a + b + c} \]
  - We can have some variations, adding weights
Data standardization

- In the Euclidean space, standardization of attributes is recommended so that all attributes can have equal impact on the computation of distances.
- Consider the following pair of data points
  - \( x_i: (0.1, 20) \) and \( x_j: (0.9, 720) \).

\[
\text{dist}(x_i, x_j) = \sqrt{(0.9 - 0.1)^2 + (720 - 20)^2} = 700.000457,
\]

- The distance is almost completely dominated by \((720-20) = 700\).
- **Standardize attributes**: to force the attributes to have a common value range

Interval-scaled attributes

- Their values are real numbers following a linear scale.
  - The difference in Age between 10 and 20 is the same as that between 40 and 50.
  - The key idea is that intervals keep the same importance through out the scale
- Two main approaches to standardize interval scaled attributes, **range** and **z-score**. \( f \) is an attribute

\[
\text{range}(x_{ij}) = \frac{x_{ij} - \min(f)}{\max(f) - \min(f)},
\]
Interval-scaled attributes (cont)

- **Z-score:** transforms the attribute values so that they have a mean of zero and a mean absolute deviation of 1. The mean absolute deviation of attribute \( f \), denoted by \( s_f \), is computed as follows

\[
s_f = \frac{1}{n} \left( | x_{1f} - m_f | + | x_{2f} - m_f | + \ldots + | x_{nf} - m_f | \right),
\]

\[
m_f = \frac{1}{n} \left( x_{1f} + x_{2f} + \ldots + x_{nf} \right),
\]

Z-score: \( z(x_{if}) = \frac{x_{if} - m_f}{s_f} \).

Ratio-scaled attributes

- Some numeric attributes, unlike interval-scaled attributes, scale exponentially,
- For example, the total amount of microorganisms that evolve in a time \( t \) is approximately given by

\[ Ae^{Bt}, \]

where \( A \) and \( B \) are some positive constants.
- Do log transform: \( \log(x_{if}) \)

- Then treat it as an interval-scaled attribute
Nominal attributes

- Sometime, we need to transform nominal attributes to numeric attributes.
- Transform nominal attributes to binary attributes.
  - The number of values of a nominal attribute is $v$.
  - Create $v$ binary attributes to represent them.
  - If a data instance for the nominal attribute takes a particular value, the value of its binary attribute is set to 1, otherwise it is set to 0.
- The resulting binary attributes can be used as numeric attributes, with two values, 0 and 1.

Nominal attributes: an example

- Nominal attribute \textit{fruit}: has three values, \textit{Apple}, \textit{Orange}, and \textit{Pear}
- We create three binary attributes called, \textit{Apple}, \textit{Orange}, and \textit{Pear} in the new data.
- If a particular data instance in the original data has Apple as the value for \textit{fruit},
  - then in the transformed data, we set the value of the attribute Apple to 1, and
  - the values of attributes Orange and Pear to 0
Ordinal attributes

- Ordinal attribute: an ordinal attribute is like a nominal attribute, but its values have a numerical ordering. E.g.,
  - Age attribute with values: Young, MiddleAge and Old. They are ordered.
  - Common approach to standardization: treat it as an interval-scaled attribute.

Nominal attributes

- Nominal attributes: with more than two states or values.
  - the commonly used distance measure is also based on the simple matching method.
  - Given two data points $x_i$ and $x_j$, let the number of attributes be $r$, and the number of values that match in $x_i$ and $x_j$ be $q$.

$$\text{dist}(x_i, x_j) = \frac{r - q}{r}$$
Mixed attributes

- Our distance functions given are for data with all numeric attributes, or all nominal attributes, etc.
- Practical data has different types:
  - Any subset of the 6 types of attributes, including:
    - interval-scaled,
    - symmetric binary,
    - asymmetric binary,
    - ratio-scaled,
    - ordinal and
    - nominal

Convert to a single type

- One common way of dealing with mixed attributes is to:
  - Decide the dominant attribute type, and
  - Convert the other types to this type.
- E.g., if most attributes in a data set are interval-scaled,
  - we convert ordinal attributes and ratio-scaled attributes to interval-scaled attributes.
  - It is also appropriate to treat symmetric binary attributes as interval-scaled attributes.
Convert to a single type (cont.)

- It does not make much sense to convert a nominal attribute or an asymmetric binary attribute to an interval-scaled attribute,
- but it is still frequently done in practice by assigning some numbers to them according to some hidden ordering, e.g., prices of the fruits.
- Alternatively, a nominal attribute can be converted to a set of (symmetric) binary attributes, which are then treated as numeric attributes.

Combining individual distances

- This approach computes individual attribute distances and then combine them.

\[
dist(x_i, x_j) = \frac{\sum_{f=1}^{r} \delta_f^{ij} d_f^{ij}}{\sum_{f=1}^{r} \delta_f^{ij}}
\]

This distance value is between 0 and 1. \( r \) is the number of attributes in the data set. The indicator \( \delta_f^{ij} \) is 1 when both values \( x_i^f \) and \( x_j^f \) for attribute \( f \) are non-missing, and it is set to 0 otherwise. It is also set to 0 if attribute \( f \) is asymmetric and the match is 0-0. Equation cannot be computed if all \( \delta_f^{ij} \)'s are 0. In such a case, some default value may be used or one of the data points is removed.

\( d_f^{ij} \) is the distance contributed by attribute \( f \), and it is in the 0-1 range.
Validity of clusters

- Why validity of clusters?
  - Given *some* data, any clustering algorithm generates clusters
  - So we need to make sure the clustering results are valid and meaningful.
- Measuring the validity of clustering results usually involve
  - Optimality of clusters
  - Verification of meaning of clusters

Cluster Evaluation: hard problem

- The quality of a clustering is very hard to evaluate because
  - We do not know the correct clusters
- Some methods are used:
  - User inspection
    - Study centroids, and spreads
    - Rules from a decision tree.
    - For text documents, one can read some documents in clusters.
Cluster evaluation: ground truth

- We use some labeled data (for classification)
- **Assumption:** Each class is a cluster.
- After clustering, a confusion matrix is constructed. From the matrix, we compute various measurements, entropy, purity, precision, recall and F-score.
  - Let the classes in the data $D$ be $C = (c_1, c_2, \ldots, c_k)$. The clustering method produces $k$ clusters, which divides $D$ into $k$ disjoint subsets, $D_1, D_2, \ldots, D_k$.

Evaluation measures: Entropy

**Entropy:** For each cluster, we can measure its entropy as follows:

$$\text{entropy}(D_i) = -\sum_{j=1}^{k} \Pr_i(c_j) \log_2 \Pr_i(c_j), \quad (29)$$

where $\Pr_i(c_j)$ is the proportion of class $c_j$ data points in cluster $i$ or $D_i$. The total entropy of the whole clustering (which considers all clusters) is

$$\text{entropy}_{\text{total}}(D) = \sum_{i=1}^{k} \frac{|D_i|}{|D|} \times \text{entropy}(D_i), \quad (30)$$
Evaluation measures: purity

Purity: This again measures the extent that a cluster contains only one class of data. The purity of each cluster is computed with

$$purity(D_i) = \max_j (Pr(c_j))$$

(31)

The total purity of the whole clustering (considering all clusters) is

$$purity_{total}(D) = \sum_{i=1}^{k} \frac{|D_i|}{|D|} \times purity(D_i)$$

(32)

An example

Assume we have a text collection $D$ of 900 documents from three topics (or three classes), Science, Sports, and Politics. Each class has 300 documents. Each document in $D$ is labeled with one of the topics (classes). We use this collection to perform clustering to find three clusters. Note that class/topic labels are not used in clustering. After clustering, we want to measure the effectiveness of the clustering algorithm.

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Science</th>
<th>Sports</th>
<th>Politics</th>
<th>Entropy</th>
<th>Purity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>250</td>
<td>20</td>
<td>10</td>
<td>0.589</td>
<td>0.893</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>180</td>
<td>80</td>
<td>1.198</td>
<td>0.643</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>100</td>
<td>210</td>
<td>1.257</td>
<td>0.617</td>
</tr>
<tr>
<td>Total</td>
<td>300</td>
<td>300</td>
<td>300</td>
<td>1.031</td>
<td>0.711</td>
</tr>
</tbody>
</table>
A remark about ground truth evaluation

- Commonly used to compare different clustering algorithms.
- A real-life data set for clustering has no class labels.
  - Thus although an algorithm may perform very well on some labeled data sets, no guarantee that it will perform well on the actual application data at hand.
- The fact that it performs well on some label data sets does give us some confidence of the quality of the algorithm.
- This evaluation method is said to be based on external data or information.

Evaluation based on internal information

- **Intra-cluster cohesion** (compactness):
  - Cohesion measures how near the data points in a cluster are to the cluster centroid.
  - Sum of squared error (SSE) is a commonly used measure.
- **Inter-cluster separation** (isolation):
  - Separation means that different cluster centroids should be far away from one another.
- In most applications, expert judgments are still the key.
**Indirect evaluation**

- In some applications, clustering is not the primary task, but used to help perform another task.
- We can use the performance on the primary task to compare clustering methods.
- For instance, in an application, the primary task is to provide recommendations on book purchasing to online shoppers.
  - If we can cluster books according to their features, we might be able to provide better recommendations.
  - We can evaluate different clustering algorithms based on how well they help with the recommendation task.
  - Here, we assume that the recommendation can be reliably evaluated.

**Summary**

- Clustering is has along history and still active
  - There are a huge number of clustering algorithms
  - More are still coming every year.
- We only introduced several main algorithms. There are many others, e.g.,
  - density based algorithm, sub-space clustering, scale-up methods, neural networks based methods, fuzzy clustering, co-clustering, etc.
- Clustering is hard to evaluate, but very useful in practice. This partially explains why there are still a large number of clustering algorithms being devised every year.
- Clustering is highly application dependent and to some extent subjective.