Chapter 9

Classification and Clustering
Classification and Clustering

Classification and clustering are classical *pattern recognition* and *machine learning* problems.

Classification, also referred to as categorization:
- Asks "what class does this item belong to?"
- *Supervised learning* task (automatically applies *labels* to data)

Clustering:
- Asks "how can I group this set of items?"
- *Unsupervised learning* task (*grouping* *related* items together)

Items can be documents, emails, queries, entities & images.

Useful for a wide variety of search engine tasks.
Classification

- **Classification** is the task of automatically applying labels to items.
- Useful for many search-related tasks:
  - Spam detection
  - Sentiment classification
  - Online advertising
- Two common approaches:
  - Probabilistic
  - Geometric
How to Classify?

- How do humans classify items?

- For example, suppose you had to classify the healthiness of a food
  - Identify set of features indicative of health: fat, cholesterol, sugar, sodium, etc.
  - Extract features from foods
    - Read nutritional facts, chemical analysis, etc.
  - Combine evidence from the features into a hypothesis
    - Add health features together to get “healthiness factor”
  - Finally, classify the item based on the evidence
    - If “healthiness factor” is above a certain value, then deem it healthy
Ontologies

- Ontology is a labeling or *categorization scheme*

**Examples**

- Binary (spam, not spam)
- Multi-valued (red, green, blue)
- Hierarchical (news/local/sports)

- Different classification tasks require different ontologies
Naïve Bayes Classifier

- Probabilistic classifier based on Bayes’ rule:

\[
P(C|D) = \frac{P(D|C)P(C)}{P(D)} = \frac{P(D|C)P(C)}{\sum_{c \in C} P(D|C)P(C|c)P(C|c)}
\]

- \(C(D)\) is a random variable corresponding to the class (input)

- Based on the term independence assumption, the Naïve Bayes’ rule yields:

\[
P(c | d) = \frac{P(d | c) P(c)}{\sum_{c \in C} P(d | c) P(c)} = \frac{\prod_{i=1}^{n} P(w_i | c) P(c)}{\sum_{c \in C} \prod_{i=1}^{n} P(w_i | c) P(c)} \quad \text{(Chain rule)}
\]
Naïve Bayes Classifier

Documents are classified according to

\[
\text{Class}(d) = \arg \max_{c \in \mathcal{C}} P(c | d)
\]

\[
= \arg \max_{c \in \mathcal{C}} \frac{P(d | c)P(c)}{\sum_{c \in \mathcal{C}} P(d | c)P(c)}
\]

Must estimate \(P(d | c)\) and \(P(c)\)

- \(P(c)\) is the probability of observing class \(c\)
- \(P(d | c)\) is the probability that document \(d\) is observed given the class is known to be \(c\)
Estimating $P(c)$

- $P(c)$ is the **probability** of observing class $c$
- Estimated as the *proportion* of training documents in class $c$:

$$P(c) = \frac{N_c}{N}$$

- $N_c$ is the number of *training documents* in class $c$
- $N$ is the total number of *training documents*
Estimating $P(d \mid c)$

- $P(d \mid c)$ is the **probability** that *document* $d$ is observed given the *class* is known to be $c$

- Estimate depends on the *event space* used to represent the documents

- **What is an event space?**
  - The set of all possible outcomes for a given random variable
  - e.g., for a coin toss random variable the event space is $S = \{ \text{heads, tails} \}$

- The **probability** of an event space $S$
  - A *probability* is assigned to each event/outcome in $S$
  - The *sum* of the *probabilities* over all the events in $S$ must equal to one
Multiple Bernoulli Event Space

- Documents are represented as *binary vectors*
  - One entry for every *word* in the vocabulary
  - Entry $i = 1$, if word $i$ occurs in the document; 0, otherwise

- Multiple Bernoulli distribution is a natural way to model distributions over binary vectors

- Same event space as used in the classical *probabilistic retrieval model*
## Multiple Bernoulli Document Representation

**Example.**

<table>
<thead>
<tr>
<th>document id</th>
<th>cheap</th>
<th>buy</th>
<th>banking</th>
<th>dinner</th>
<th>the</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>not spam</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>spam</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>not spam</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>spam</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>spam</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>not spam</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>not spam</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>not spam</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>not spam</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>not spam</td>
</tr>
</tbody>
</table>
Multiple-Bernoulli: Estimating $P(d \mid c)$

- $P(d \mid c)$ is computed (in the Multiple-Bernoulli model) as

$$P(d \mid c) = \prod_{w \in \mathcal{V}} P(w \mid c)^{\delta(w, d)} (1 - P(w \mid c))^{1-\delta(w, d)}$$

where $\delta(w, d) = 1$ iff term $w$ occurs in $d$; $P(d \mid c) = 0$ if $\exists w \in d$ never occurred in $c$ in the training set, the “data sparseness” problem, which can be solved by the “smoothing” methods.

- Laplacian smoothed estimate:

$$P(w \mid c) = \frac{df_{w,c} + 1}{N_c + 1}$$

where $df_{w,c}$ denotes the number of documents in $c$ including term $w$

$N_c$ is the number of documents belonged to class $c$

- Collection smoothed estimate:

$$P(w \mid c) = \frac{df_{w,c} + \mu \frac{N_w}{N}}{N_c + \mu}$$

where $\mu$ is a tunable parameter and $N_w$ is the no. of doc. including $w$
Documents are represented as vectors of term frequencies

- One entry for every word in the vocabulary
- Entry $i = \text{number of times that term } i \text{ occurs in the document}$

Multinomial distribution is a natural way to model distributions over frequency vectors

Same event space as used in the language modeling retrieval model
### Multinomial Document Representation

**Example.**

<table>
<thead>
<tr>
<th>document id</th>
<th>cheap</th>
<th>buy</th>
<th>banking</th>
<th>dinner</th>
<th>the</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>not spam</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>spam</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>not spam</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>spam</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>spam</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>not spam</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>not spam</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>not spam</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>not spam</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>not spam</td>
</tr>
</tbody>
</table>
Multinomial: Estimating $P(d \mid c)$

- $P(d \mid c)$ is computed as:

$$P(d \mid c) = P(|d|) \left( t f_{w_1,d}, t f_{w_2,d}, \ldots, t f_{w_{|V|},d} \right)! \prod_{w \in V} P(w \mid c)^{t f_{w,d}}$$

- Laplacian smoothed estimate:

$$P(w \mid c) = \frac{t f_{w,c} + 1}{|c| + |V|}$$

where $|c|$ is the number of terms in the training documents of class $c$ and $|V|$ is the number of distinct terms in the training documents.

- Collection smoothed estimate:

$$P(w \mid c) = \frac{t f_{w,c} + \mu c f_{w}}{|c| + \mu}$$

Number of term $w$ in a training set $C$ and Number of terms in all training documents.
Multinomial Versus Multiple-Bernoulli Model

- The Multinomial model is consistently outperform the Multiple-Bernoulli model
- Implementing both models is relatively straightforward
- Both classifiers are
  - efficient, since their statistical data can be stored in memory
  - accurate in document classification
  - popular and attractive choice as a general-purpose classifier
Support Vector Machines (SVM)

- A vector-space-based machine-learning method
- Goal: find a decision boundary between two classes that is *maximally* far from any point in the training data
- Two-class data sets are separable by a linear classifier

An infinite number of hyperplanes that separate two *linearly separable* classes (right up against the margin of the classifier)
Support Vector Machines (SVM)

- Based on geometric principles
- Documents are represented as $N$-dimensional vectors with (non-)binary feature weights
- Given a set of inputs labeled ‘+’ & ‘-’, find the best hyperplane in an $N$-dimensional space that separates the ‘+’s and ‘-’s, i.e., a binary-categorization method

Questions

- How is “best” defined?
- What if no hyperplane exists such that the ‘+’s and ‘-’s can be perfectly separated?
“Best” Hyperplane?

- First, what is a hyperplane?
  - A generalization of a line to higher dimensions
  - Defined by vector $w$ that is learned from the training data
  - Avoiding the overfitting problem, i.e., working well with the training data, but fails at classifying the test data

- To avoid overfitting, SVM chooses a hyperplane with the maximum margin that separates ‘+’s & ‘-’s
  - Necessarily since points near the decision surface represent very uncertain classification decisions with almost a 50% chance deciding either way
  - Correctly generalize to test data is increased
Support Vector Machines

\[ w \cdot x > 0 \]

\[ w \cdot x < 0 \]

\[ w \cdot x = 0 \]

Vector \( w \) defines the hyperplane \((H)\)

Distance from \( x^- \) to \( H \) plus the distance from \( x^+ \) to \( H \)

Maximizes the separation of the ‘+’ and ‘-’ data points
“Best” Hyperplane?

- $w \cdot x$ is a scalar, i.e., a single number, the projection of $w$ onto $x$

- $w \cdot x = 0$ specifies each point $x$ that lies on the line perpendicular to $w$

- $w \cdot x = 1$ is the line parallel to $w \cdot x = 0$, shifted by $1 / \|w\|$

- $2 / \|w\|$ is thus the maximal margin, which is the objective boundary function
“Best” Hyperplane?

- If $x^+ \& x$ are the closest ‘+’ & ‘-’ inputs to the hyperplane, called support vectors, then the margin is:

$$Margin(w) = \frac{|w \cdot x^-| + |w \cdot x^+|}{||w||}$$

- It is typically assumed that $|w \cdot x^-| = |w \cdot x^+| = 1$, which does not change the solution to the problem.

- Thus, to find the hyperplane with the largest (maximal) margin, it requires $Margin(w) = 2 / (||w|| = (w \cdot w)^{1/2})$
Separable vs. Non-Separable Data

**Separable**

*Linearly separable data sets are well-handed*

**Non-Separable**

Must map the original feature space to some *higher dimensional* feature space where the dataset is separable.
Linear Separable Case

- **In math:**
  \[
  \text{minimize : } \frac{1}{2}||w||^2 \\
  \text{subject to :}
  \]
  \[
  w \cdot x_i \geq 1 \quad \forall i \text{ s.t. Class}(i) = + \\
  w \cdot x_i \leq -1 \quad \forall i \text{ s.t. Class}(i) = -
  \]

- **In English:**
  - Find the *largest margin* hyperplane that separates the ‘+’s and ‘-’s
  - Can be solved using *quadratic programming*
  - An unseen document \(d\) can be classified using
    \[
    \text{Class}(d) = \begin{cases} 
    + & \text{if } w \cdot x_d > 0 \\
    - & \text{otherwise}
    \end{cases}
    \]
Feature Selection for Text Classification

- Document classifiers can have a very large number of features, such as indexed terms
  - *Not* all features are useful
  - Excessive features can increase computational cost of training and testing
- Feature selection methods *reduce* the number of features by choosing the most useful features
  - which can significantly *improve efficiency* (in terms of storage and processing time) while *not hurting* the effectiveness much (in addition to *eliminating noisy*)
Information Gain

- Information gain (IG) is a commonly used feature selection measure based on information theory
  - It tells how much “information” is gained (about the class labels) if we observe some feature
  - Entropy characterizes the (im)purity of a collection of examples
  - The information gain is the expected reduction in entropy caused by partitioning the examples according to an attribute (word)
  - Rank features by information gain and then train model using the top $K$ ($K$ is typically small) attributes (words)

- The information gain for a MNB classifier is computed as

$$IG(w) = \frac{1}{C} \sum_{c \in C} P(c) \log P(c) - \sum_{c \in C} P(w) \sum_{c \in C} P(c|w) \log P(c|w)$$

**Entropy of $P(c)$**  
**Conditional Entropy**
Feature Selection for Text Classification

- Feature selection is based on entropy/information gain
  - The law of large numbers indicates that the symbol $a_j$ will, on the average, be selected
    
    $$n \times p(a_j)$$
  
  times in a total of $n$ selections
  - The average amount of information obtained from $n$ different source outputs for each $a_j$ ($j \geq 1$) with $-\log_2 p(a_j)$ bits is
    
    $$n \times p(a_1) \log_2 p(a_1)^{-1} + \ldots + n \times p(a_j) \log_2 p(a_j)^{-1}$$

    bits. Divided by $n$ obtain the average amount of information per source output symbol, which is known as uncertainty, or the entropy, $E$, where $\sigma$ is the number of symbols

    $$E = - \sum_{i=1}^{\sigma} p_i \log_2 p_i$$
Information Gain

- Example. The information gain for the term “cheap”, using

$$IG(w) = -\sum_{c \in C} P(c) \log P(c) + \sum_{w \in \{0,1\}} P(w) \sum_{c \in C} P(c|w) \log P(c|w)$$

where $P(\text{cheap})$ denotes $P(\text{cheap} = 0)$,

$P(\text{spam})$ denotes $P(\text{not spam})$,

$0 \log 0 = 0$, and

$IG(\text{buy}) = 0.0008$, $IG(\text{banking}) = 0.04$, $IG(\text{dinner}) = 0.36$, $IG(\text{the}) = 0$

\[
IG(\text{cheap}) = -P(\text{spam}) \log P(\text{spam}) - P(\text{spam}) \log P(\text{spam}) + \\
P(\text{cheap})P(\text{spam|cheap}) \log P(\text{spam|cheap}) + \\
P(\text{cheap})P(\text{spam|cheap}) \log P(\text{spam|cheap}) + \\
P(\text{cheap})P(\text{spam|cheap}) \log P(\text{spam|cheap}) + \\
P(\text{cheap})P(\text{spam|cheap}) \log P(\text{spam|cheap}) = \\
= \frac{3}{10} \log \frac{3}{10} - \frac{7}{10} \log \frac{7}{10} + \frac{4}{10} \cdot \frac{3}{4} \log \frac{3}{4} \\
+ \frac{4}{10} \cdot \frac{6}{4} \log \frac{6}{4} + \frac{6}{10} \cdot \frac{0}{6} \log \frac{6}{6} + \frac{6}{10} \cdot \frac{0}{6} \log \frac{6}{6} = 0.2749
\]
Clustering

- A set of unsupervised algorithms that attempt to find latent structure in a set of items

- Goal is to identify groups (clusters) of similar items, given a set of unlabeled instances

- Suppose I gave you the shape, color, vitamin C content and price of various fruits and asked you to cluster them
  - What criteria would you use?
  - How would you define similarity?

- Clustering is very sensitive to (i) how items are represented and (ii) how similarity is defined
Clustering

- General outline of clustering algorithms
  
  1. Decide how items will be represented (e.g., feature vectors)
  2. Define similarity measure between pairs or groups of items (e.g., cosine similarity)
  3. Determine what makes a “good” clustering (e.g., using intra- & inter-cluster similarity measures)
  4. Iteratively construct clusters that are increasingly “good”
  5. Stop after a local/global optimum clustering is found

- Steps 3 and 4 differ the most across algorithms
Hierarchical Clustering

- Constructs a *hierarchy of clusters*
  - Starting with some initial clustering of data & iteratively trying to improve the “quality” of clusters
  - The *top* level of the hierarchy consists of a *single* cluster with *all* items in it
  - The *bottom* level of the hierarchy consists of *N* (number of items) *singleton* clusters

- Different objectives lead to different types of clusters

- Two types of hierarchical clustering
  - Divisive (“*top down*”)
  - Agglomerative (“*bottom up*”)

- Hierarchy can be visualized as a *dendogram*
Example Dendrogram

Height indicates the similarity of the clusters involved
Divisive & Agglomerative Hierarchical Clustering

- **Divisive**
  - Start with a *single* cluster consisting of all of the items
  - Until only *singleton clusters* exist …
    - *Divide* an existing cluster into two (or more) new clusters

- **Agglomerative**
  - Start with *N* (number of items) singleton clusters
  - Until a single cluster exists …
    - *Combine* two (or more) existing cluster into a new cluster

- **How do we know how to divide or combine clusters?**
  - Define a *division* or *combination cost*
  - Perform the division or combination with the *lowest cost*
Divisive Hierarchical Clustering
Agglomerative Hierarchical Clustering
Clustering Costs

- **Cost**: a measure of how *expensive* to merge 2 clusters

- **Single linkage**
  \[
  COST(C_i, C_j) = \min \{dist(X_i, X_j) | X_i \in C_i, X_j \in C_j\}
  \]

- **Complete linkage**
  \[
  COST(C_i, C_j) = \max \{dist(X_i, X_j) | X_i \in C_i, X_j \in C_j\}
  \]

- **Average linkage**
  \[
  COST(C_i, C_j) = \frac{\sum_{X_i \in C_i, X_j \in C_j} dist(X_i, X_j)}{|C_i||C_j|}
  \]

- **Average group linkage**
  \[
  COST(C_i, C_j) = dist(\mu_{C_i}, \mu_{C_j})
  \]
  where $\mu_C = (\sum_{X \in C} X) / |C|$ is the centroid of cluster $C$
Clustering Strategies

* Generally, *Average-Link* Clustering yields the best effectiveness
Clustering Costs

- The choice of the best clustering technique/strategy requires experiments & evaluation

- Single linkage
  - Could result in “very long” or “spread-out” clusters

- Complete linkage
  - Clusters are more compact than Single Linkage

- Average linkage
  - A compromise between Single & Complete Linkage

- Average group linkage
  - Closely related to the Average Linkage approach
Hierarchical clustering constructs a hierarchy of clusters

K-means always maintains exactly $K$ clusters
  - Clusters are represented by their centroids ("centers of mass")

Basic algorithm:
  - Step 0: Choose $K$ cluster centroids
  - Step 1: Assign points to closest centroid
  - Step 2: Re-compute cluster centroids
  - Step 3: Goto Step 1

Tends to converge quickly

Can be sensitive to choice of initial centroids

Must choose $K$ to begin with!
K-Means Clustering

- **Goal:** find the cluster assignments (for the assignment vectors $A[1], \ldots, A[N]$) that **minimize** the cost function:

$$
\text{COST}(A[1], \ldots, A[N]) = \sum_{k=1}^{K} \sum_{i:A[i]=k} \text{dist}(X_i, C_k)
$$

where $\text{dist}(X_i, C_k) = \| X_i - \mu_{C_k} \|^2$, where $\mu_{C_k}$ is the **centroid** of $C_k$

$$
= (X_i - \mu_{C_k}) \cdot (X_i - \mu_{C_k}), \text{ the Euclidean Distance}
$$

- **Strategy:**
  1. Randomly select $K$ initial cluster centers (instances) as **seeds**
  2. Move the cluster centers around to **minimize** the cost function
     1. Re-assign instances to the cluster with the **closest** centroid
     2. Re-compute the cost value of each centroid based on the current members of its cluster
K-Means Clustering

Example.
K-Means Clustering

The $K$-means optimization problem:

- A naïve approach is to try every possible combination of cluster assignments, which is *infeasible* for large data sets.
- The $K$-means algorithm should find an *approximate, heuristic solution* that iteratively tries to minimize the cost.

Anticipated results:
- The solution is *not* guaranteed to be *globally (nor locally) optimal*.
- Despite the heuristic nature, the $K$-means algorithm tends to work very well in practice.

In practice, *$K$-means clustering* tends to converge quickly:

- Compared to hierarchical clustering $H$, $K$-means is more *efficient* and produces clusters of similar quality to $H$.
- Implementing $K$-means requires $O(KN)$, rather than $O(N^2)$ for $H$.
K-Means Clustering Algorithm

Algorithm 1 K-Means Clustering

1: procedure KMEANSCluster($X_1, \ldots, X_N, K$)
2: \hspace{1em} $A[1], \ldots, A[N] \leftarrow$ initial cluster assignment (* Either randomly or using some knowledge of the data *)
3: \hspace{1em} repeat
4: \hspace{2em} change $\leftarrow$ false
5: \hspace{2em} for $i = 1$ to $N$ do
6: \hspace{3em} $\hat{k} \leftarrow \arg \min_k \text{dist}(X_i, C_k)$ (* Each instance is assigned to the closest cluster *)
7: \hspace{3em} if $A[i]$ is not equal $\hat{k}$ then
8: \hspace{4em} $A[i] \leftarrow \hat{k}$ (* The cluster of an instance changes; proceeds *)
9: \hspace{4em} change $\leftarrow$ true
10: \hspace{3em} end if
11: \hspace{2em} end for
12: \hspace{1em} until change is equal to false return $A[1], \ldots, A[N]$  
13: end procedure
- *Hierarchical* and *K-Means* clustering partition items into clusters
  - Every item is in exactly one cluster

- *K Nearest neighbor* clustering forms one cluster per item
  - The cluster for item \( j \) consists of \( j \) and the *K nearest neighbors of* \( j \)
  - Clusters now overlap
5 Nearest Neighbor Clustering
K Nearest Neighbor Clustering

**Drawbacks** of the $K$ Nearest Neighbor Clustering method:

- Often fails to find meaningful clusters
  - In *spare areas* of the input space, the instances assigned to a cluster are farther far away (e.g., $D$ in the 5-NN example)
  - In *dense areas*, some related instances may be missed if $K$ is not large enough (e.g., $B$ in the 5-NN example)

- **Computational expensive** (compared with $K$-means), since it computes distances between *each pair* of instances

**Applications** of the $K$ Nearest Neighbor Clustering method

- Emphasize finding a small number (rather than all) of closely related instances, i.e., precision over recall
How to Choose K?

- *K*-means and *K* nearest neighbor clustering require us to choose *K*
  - No theoretically appealing way of choosing *K*

- Depends on the *application* and *data*; often chosen experimentally to evaluate the *quality* of the resulting clusters for various values of *K*

- Can use *hierarchical clustering* and choose the best level

- Can use *adaptive* *K* for *K*-nearest neighbor clustering
  - Larger (Smaller) *K* for *dense* (spare) areas
  - Challenge: choosing the *boundary* size

- Difficult problem with no clear solution
Adaptive Nearest Neighbor Clustering
Evaluation of Clustering

- Typical objective functions/\textit{internal criterion} in clustering
  - Attaining high \textit{intra-cluster similarity} (documents within a cluster are \textit{similar})
  - Achieving low \textit{inter-cluster similarity} (documents from different clusters are \textit{dissimilar})

- \textit{External criterion} evaluates how well the clustering matches the \textit{gold standard} classes
  - The gold standard is ideally produced by human judges with a good level of inter-judge agreement
  - A set of \textit{classes} are used in the evaluation benchmark
  - Four external criteria of clustering quality: \textit{Purity}, \textit{Normalized mutual information}, \textit{Rand index}, & \textit{F-measure}
Evaluation of Clustering

- **Purity**, a simple & transparent evaluation measure
  - Each *cluster* is assigned to the *class* which is **most frequent** in the cluster
  - The *accuracy* of the assignment is measured by counting the number of correctly assigned documents divided by \( N \), the total number of documents to be clustered

\[
Purity(\Omega, C) = \frac{1}{N} \sum_{i=1}^{K} \max_{j} | w_i \cap C_j |
\]

where \( \Omega = \{ w_1, w_2, \ldots w_K \} \) is the set of clusters

\( C = \{ C_1, C_2, \ldots C_J \} \) is the set of classes

\( w_n, 1 \leq n \leq K \ (C_m, 1 \leq m \leq J) \) is a set of documents

- **Bad** clusterings have purity values close to 0 & a **perfect** clustering has a purity of 1
Example. An external evaluation criterion for cluster quality where majority class & no. of member of the majority class for the 3 classes are: x (5), o (4), and ♦ (3)

- Purity is \( \frac{1}{17} \times (5 + 4 + 3) \approx 0.71 \)

<table>
<thead>
<tr>
<th></th>
<th>Purity</th>
<th>NMI</th>
<th>RI</th>
<th>F_5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Maximum</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Value for Example</td>
<td>0.71</td>
<td>0.36</td>
<td>0.68</td>
<td>0.46</td>
</tr>
</tbody>
</table>
Evaluation of Clustering

- Normalized Mutual Information

  - High purity is easy to achieve when the number of clusters is large, in particular, purity is 1 if each doc is assigned its own cluster

  - A tradeoff is the \textit{normalized mutual information (NMI)}

    \[
    \text{NMI}(\Omega, C) = \frac{I(\Omega; C)}{[H(\Omega) + H(C)] / 2}
    \]

    where \( I \) is mutual information, the knowledge about the classes

    \[
    I(\Omega; C) = \sum_{n=1}^{k} \sum_{m=1}^{J} P(w_n \cap C_m) \log \frac{P(w_n \cap C_m)}{P(w_n) P(C_m)}
    \]

    \[
    H(\Omega) = - \sum_{n=1}^{k} P(w_n) \log P(w_n) = - \sum_{n=1}^{k} \frac{|w_n|}{N} \log \frac{|w_n|}{N}, \text{ the entropy}
    \]
Evaluation of Clustering

- Normalized Mutual Information
  - $I(\Omega; C)$ is 0 if the clustering is random w.r.t. class membership
  - Maximum mutual information (MI) is reached for a clustering $\Omega_{\text{exact}}$ that perfectly recreates the classes
  - A clustering with $K = N$, i.e., generate one document clusters, has the maximum MI, the same problem as purity, which is fixed by $[H(\Omega) + H(C)] / 2$
  - Entropy tends to increase with the number of clusters, i.e., $H(\Omega)$ reaches its maximum log $N$ for $K = N$
  - $[H(\Omega) + H(C)] / 2$ is a tight upper bound on $I(\Omega; C)$
Evaluation of Clustering

- Rand Index (RI), which measures the decisions that are correct on assigning two documents to the same cluster, assuming that they are similar

\[
RI = \frac{TP + TN}{TP + FP + FN + TN}
\]

where

TP occurs if two similar docs are assigned to the same cluster
TN occurs when two dissimilar docs are assigned to different clusters
FP occurs if two dissimilar docs are assigned to the same cluster
FN occurs when two similar docs are assigned to different clusters
Evaluation of Clustering

- **Example (RI).** Given the following clusters

![Cluster Diagram]

\[ TP + FP = \binom{6}{2} + \binom{6}{2} + \binom{5}{2} = 40 \]

\[ TP = \binom{5}{2} + \binom{4}{2} + \binom{3}{2} + \binom{2}{2} = 20 \]

\[ FN = \binom{5}{1} + \binom{5}{2} + \binom{4}{1} + \binom{3}{1} + \binom{2}{1} \]

\[ = 5 + 10 + 4 + 3 + 2 = 24 \]

\[ TN = 25+2 \times (C1-C2) + 15+5 \times (C1-C3) + 20+3+2 \times (C2-C3) = 72 \]

\[ RI = \frac{(20 + 72)}{(20 + 20 + 24 + 72)} \approx 0.68 \]
Evaluation of Clustering

- **F-Measure**
  - RI gives \textit{equal} weight to FPs and FNs
  - Separating \textit{similar} docs is sometimes worse than putting pairs of \textit{dissimilar} docs in the same cluster
  - F-measure penalizes FNs more strongly than FPs by selecting a $\beta > 1$, thus giving more weight to \textit{recall}

\[
P = \frac{TP}{TP + FP}, \quad R = \frac{TP}{TP + FN}, \quad F_\beta = \frac{(\beta^2 + 1) \cdot PR}{\beta^2 P + R}
\]

- **Example.** Based on $TP = 20$, $FP = 20$, $FN = 24$, and $TN = 72$

  \[
P = \frac{20}{40} = 0.5, \quad R = \frac{20}{44} \approx 0.455, \quad F_1 \approx 0.48, \quad F_5 \approx 0.456
\]
Evaluating Clustering

- Evaluating clustering is challenging, since it is an unsupervised learning task.

- If labels exist, can use standard IR metrics, e.g., precision/recall.

- If not, then can use measures such as “cluster precision”, which is defined as:

\[
\text{Cluster Precision} = \frac{\sum_{i=1}^{K} |\text{MaxClass}(C_i)|}{N}
\]

where \( K (= |C|) \) is the total number of resultant clusters.

\(|\text{MaxClass}(C_i)|\) is the number of instances in cluster \( C_i \) with the most instances of (human-assigned) class label \( C_i \).

\( N \) is the total number of instances.

- Another option is to evaluate clustering as part of an end-to-end system, e.g., clusters to improve web ranking.