Text Clustering

Clustering

- Partition unlabeled examples into disjoint subsets of clusters, such that:
  - Examples within a cluster are very similar
  - Examples in different clusters are very different
- Discover new categories in an unsupervised manner (no sample category labels provided).
Hierarchical Clustering

- Build a tree-based hierarchical taxonomy \((dendrogram)\) from a set of unlabeled examples.

- Recursive application of a standard clustering algorithm can produce a hierarchical clustering.
Aglommerative vs. Divisive Clustering

- **Aglommerative** (*bottom-up*) methods start with each example in its own cluster and iteratively combine them to form larger and larger clusters.
- **Divisive** (*partitional, top-down*) separate all examples immediately into clusters.

Direct Clustering Method

- **Direct clustering** methods require a specification of the number of clusters, $k$, desired.
- A **clustering evaluation function** assigns a real-value quality measure to a clustering.
- The number of clusters can be determined automatically by explicitly generating clusterings for multiple values of $k$ and choosing the best result according to a clustering evaluation function.
Hierarchical Agglomerative Clustering (HAC)

- Assumes a *similarity function* for determining the similarity of two instances.
- Starts with all instances in a separate cluster and then repeatedly joins the two clusters that are most similar until there is only one cluster.
- The history of merging forms a binary tree or hierarchy.

HAC Algorithm

Start with all instances in their own cluster.
Until there is only one cluster:
   - Among the current clusters, determine the two clusters, $c_i$ and $c_j$, that are most similar.
   - Replace $c_i$ and $c_j$ with a single cluster $c_i \cup c_j$.
Cluster Similarity

- Assume a similarity function that determines the similarity of two instances: $sim(x,y)$.
  - Cosine similarity of document vectors.
- How to compute similarity of two clusters each possibly containing multiple instances?
  - **Single Link**: Similarity of two most similar members.
  - **Complete Link**: Similarity of two least similar members.
  - **Group Average**: Average similarity between members.

Single Link Agglomerative Clustering

- Use maximum similarity of pairs:
  \[
  sim(c_i,c_j) = \max_{x \in c_i, y \in c_j} sim(x, y)
  \]
- Can result in “straggly” (long and thin) clusters due to *chaining effect*.
  - Appropriate in some domains, such as clustering islands.
Single Link Example

Complete Link Agglomerative Clustering

- Use minimum similarity of pairs:

\[ \text{sim}(c_i, c_j) = \min_{x \in c_i, y \in c_j} \text{sim}(x, y) \]

- Makes more “tight,” spherical clusters that are typically preferable.
Computing Cluster Similarity

- After merging $c_i$ and $c_j$, the similarity of the resulting cluster to any other cluster, $c_k$, can be computed by:
  - Single Link:
    \[
    sim((c_i \cup c_j), c_k) = \max(sim(c_i, c_k), sim(c_j, c_k))
    \]
  - Complete Link:
    \[
    sim((c_i \cup c_j), c_k) = \min(sim(c_i, c_k), sim(c_j, c_k))
    \]
Group Average Agglomerative Clustering

- Use average similarity across all pairs within the merged cluster to measure the similarity of two clusters.

\[
sim(c_i, c_j) = \frac{1}{|c_i \cup c_j|} \sum_{\tilde{x} \in (c_i \cup c_j)} \sum_{\tilde{y} \in (c_i \cup c_j), \tilde{y} \neq \tilde{x}} \sum \sim(x, y)
\]

- Compromise between single and complete link.

Non-Hierarchical Clustering

- Typically must provide the number of desired clusters, \( k \).
- Randomly choose \( k \) instances as seeds, one per cluster.
- Form initial clusters based on these seeds.
- Iterate, repeatedly reallocated instances to different clusters to improve the overall clustering.
- Stop when clustering converges or after a fixed number of iterations.
K-Means

- Assumes instances are real-valued vectors.
- Clusters based on centroids, center of gravity, or mean of points in a cluster, \( c \):
  \[
  \bar{\mu}(c) = \frac{1}{|c|} \sum_{x \in c} \bar{x}
  \]
- Reassignment of instances to clusters is based on distance to the current cluster centroids.

Distance Metrics

- Euclidian distance (L_2 norm):
  \[
  L_2(\bar{x}, \bar{y}) = \sum_{i=1}^{m} (x_i - y_i)^2
  \]
- L_1 norm:
  \[
  L_1(\bar{x}, \bar{y}) = \sum_{i=1}^{m} |x_i - y_i|
  \]
- Cosine Similarity (transform to a distance by subtracting from 1):
  \[
  1 - \frac{\bar{x} \cdot \bar{y}}{||\bar{x}|| \cdot ||\bar{y}||}
  \]
K-Means Algorithm

Let $d$ be the distance measure between instances.
Select $k$ random instances $\{s_1, s_2, \ldots, s_k\}$ as seeds.
Until clustering converges or other stopping criterion:
  For each instance $x_i$:
    Assign $x_i$ to the cluster $c_j$ such that $d(x_i, s_j)$ is minimal.
    *Update the seeds to the centroid of each cluster:*
  For each cluster $c_j$
    $s_j = \mu(c_j)$

K Means Example
(K=2)

Pick seeds
Reassign clusters
Compute centroids
Reassign clusters
Compute centroids
Reassign clusters
Converged!
Seed Choice

- Results can vary based on random seed selection.
- Some seeds can result in poor convergence rate, or convergence to sub-optimal clusterings.
- Select good seeds using a heuristic or the results of another method.

Text Clustering

- HAC and K-Means have been applied to text in a straightforward way.
- Typically use normalized, TF/IDF-weighted vectors and cosine similarity.
- Optimize computations for sparse vectors.
- Applications:
  - During retrieval, add other documents in the same cluster as the initial retrieved documents to improve recall.
  - Clustering of results of retrieval to present more organized results to the user (à la Northernlight folders).
  - Automated production of hierarchical taxonomies of documents for browsing purposes (à la Yahoo & DMOZ).
Soft Clustering

- Clustering typically assumes that each instance is given a “hard” assignment to exactly one cluster.
- Does not allow uncertainty in class membership or for an instance to belong to more than one cluster.
- *Soft clustering* gives probabilities that an instance belongs to each of a set of clusters.
- Each instance is assigned a probability distribution across a set of discovered categories (probabilities of all categories must sum to 1).

Exercise

Cluster to following documents using K-means with K=2 and cosine similarity.

- Doc1: “go monster go”
- Doc2: “go karting”
- Doc3: “karting monster”
- Doc4: “monster monster”

Assume Doc1 and Doc3 are chosen as initial seeds. Use tf (no idf). Show the clusters and their centroids for each iteration. The algorithm should converge after 2 iterations.