Clustering:
Similarity-Based Clustering

CS4780/5780 – Machine Learning
Fall 2013

Thorsten Joachims
Cornell University

Reading: Manning/Raghavan/Schuetze,
Chapters 16 (not 16.3) and 17
(http://nlp.stanford.edu/IR-book/)
Outline

• Supervised vs. Unsupervised Learning
• Hierarchical Clustering
  – Hierarchical Agglomerative Clustering (HAC)
• Non-Hierarchical Clustering
  – K-means
  – Mixtures of Gaussians and EM-Algorithm
Supervised Learning vs. Unsupervised Learning

- **Supervised Learning**
  - Classification: partition examples into groups according to pre-defined categories
  - Regression: assign value to feature vectors
  - Requires labeled data for training

- **Unsupervised Learning**
  - Clustering: partition examples into groups when no pre-defined categories/classes are available
  - Novelty detection: find changes in data
  - Outlier detection: find unusual events (e.g. hackers)
  - Only instances required, but no labels
Clustering

- Partition unlabeled examples into disjoint subsets of *clusters*, such that:
  - Examples within a cluster are similar
  - Examples in different clusters are different

- Discover new categories in an *unsupervised* manner (no sample category labels provided).
Applications of Clustering

• Cluster retrieved documents
  – to present more organized and understandable results to user → “diversified retrieval”
• Detecting near duplicates
  – Entity resolution
    • E.g. “Thorsten Joachims” == “Thorsten B Joachims”
  – Cheating detection
• Exploratory data analysis
• Automated (or semi-automated) creation of taxonomies
  – e.g. Yahoo, DMOZ
• Compression
Applications of Clustering
Clustering Example
Clustering Example
Clustering Example
Clustering Example
Similarity (Distance) Measures

• Euclidean distance ($L_2$ norm):

$$L_2(\vec{x}, \vec{x}') = \sqrt{\sum_{i=1}^{N} (x_i - x_i')^2}$$

• $L_1$ norm:

$$L_1(\vec{x}, \vec{x}') = \sqrt{\sum_{i=1}^{N} |x_i - x_i'|}$$

• Cosine similarity:

$$\cos(\vec{x}, \vec{x}') = \frac{\vec{x} \cdot \vec{x}'}{\|\vec{x}\| \|\vec{x}'\|}$$

• Kernels
Hierarchical Clustering

- Build a tree-based hierarchical taxonomy from a set of unlabeled examples.

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

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

- Assumes a *similarity function* for determining the similarity of two clusters.
- 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.
- Basic 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

• 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.
When computing cluster similarity, 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.
When computing cluster similarity, use minimum similarity of pairs:

$$\min (\sim (x, y))$$

→ Makes more “tight,” spherical clusters.

$$\text{sim}(c_i, c_j) = \min_{x \in c_i, y \in c_j} \text{sim}(x, y)$$
Computational Complexity of HAC

• In the first iteration, all HAC methods need to compute similarity of all pairs of $n$ individual instances which is $O(n^2)$.

• In each of the subsequent $O(n)$ merging iterations, must find smallest distance pair of clusters $\rightarrow$ Maintain heap $O(n^2 \log n)$

• In each of the subsequent $O(n)$ merging iterations, it must compute the distance between the most recently created cluster and all other existing clusters. Can this be done in constant time such that $O(n^2 \log n)$ overall?
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:
    \[
    \text{sim}((c_i \cup c_j), c_k) = \max(\text{sim}(c_i, c_k), \text{sim}(c_j, c_k))
    \]

  – Complete Link:
    \[
    \text{sim}((c_i \cup c_j), c_k) = \min(\text{sim}(c_i, c_k), \text{sim}(c_j, c_k))
    \]
### Single-Link Example

#### Table 1: Initial Data

<table>
<thead>
<tr>
<th></th>
<th>x1</th>
<th>x2</th>
<th>x3</th>
<th>x4</th>
<th>x5</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>1</td>
<td>0.8</td>
<td>0.2</td>
<td>0.7</td>
<td>0.3</td>
</tr>
<tr>
<td>x2</td>
<td>0.8</td>
<td>1</td>
<td>0.1</td>
<td>0.5</td>
<td>0.2</td>
</tr>
<tr>
<td>x3</td>
<td>0.2</td>
<td>0.1</td>
<td>1</td>
<td>0.9</td>
<td>0.5</td>
</tr>
<tr>
<td>x4</td>
<td>0.7</td>
<td>0.5</td>
<td>0.9</td>
<td>1</td>
<td>0.4</td>
</tr>
<tr>
<td>x5</td>
<td>0.3</td>
<td>0.2</td>
<td>0.5</td>
<td>0.4</td>
<td>1</td>
</tr>
</tbody>
</table>

#### Table 2: After Merge x3,x4

<table>
<thead>
<tr>
<th></th>
<th>x1</th>
<th>x2</th>
<th>c1</th>
<th>x5</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>1</td>
<td>0.8</td>
<td>0.7</td>
<td>0.3</td>
</tr>
<tr>
<td>x2</td>
<td>0.8</td>
<td>1</td>
<td>0.5</td>
<td>0.2</td>
</tr>
<tr>
<td>c1</td>
<td>0.7</td>
<td>0.5</td>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>x5</td>
<td>0.3</td>
<td>0.2</td>
<td>0.5</td>
<td>1</td>
</tr>
</tbody>
</table>

#### Table 3: After Merge x1,x2

<table>
<thead>
<tr>
<th></th>
<th>c2</th>
<th>c1</th>
<th>x5</th>
</tr>
</thead>
<tbody>
<tr>
<td>c2</td>
<td>1</td>
<td>0.7</td>
<td>0.3</td>
</tr>
<tr>
<td>c1</td>
<td>0.7</td>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>x5</td>
<td>0.3</td>
<td>0.5</td>
<td>1</td>
</tr>
</tbody>
</table>

#### Table 4: After Merge c1,c2

<table>
<thead>
<tr>
<th></th>
<th>c3</th>
<th>x5</th>
</tr>
</thead>
<tbody>
<tr>
<td>c3</td>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>x5</td>
<td>0.5</td>
<td>1</td>
</tr>
</tbody>
</table>
Group Average Agglomerative Clustering

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

\[
sim(c_i, c_j) = \frac{1}{|c_i \cup c_j| (|c_i \cup c_j| - 1)} \sum_{\bar{x} \in (c_i \cup c_j)} \sum_{\bar{y} \in (c_i \cup c_j): \bar{y} \neq \bar{x}} \text{sim}(\bar{x}, \bar{y})
\]

• Compromise between single and complete link.
Computing Group Average Similarity

• Assume cosine similarity and normalized vectors with unit length.

• Always maintain sum of vectors in each cluster.

\[ \tilde{s}(c_j) = \sum_{\tilde{x} \in c_j} \tilde{x} \]

• Compute similarity of clusters in constant time:

\[ \text{sim}(c_i, c_j) = \frac{(\tilde{s}(c_i) + \tilde{s}(c_j)) \cdot (\tilde{s}(c_i) + \tilde{s}(c_j)) - (|c_i| + |c_i|)}{(|c_i| + |c_i|)(|c_i| + |c_i| - 1)} \]
Non-Hierarchical Clustering

- K-means clustering ("hard")
- Mixtures of Gaussians and training via Expectation maximization Algorithm ("soft")
Clustering Criterion

• Evaluation function that assigns a (usually real-valued) value to a clustering
  – Clustering criterion typically function of
    • within-cluster similarity and
    • between-cluster dissimilarity

• Optimization
  – Find clustering that maximizes the criterion
    • Global optimization (often intractable)
    • Greedy search
    • Approximation algorithms
Centroid-Based Clustering

• Assumes instances are real-valued vectors.
• Clusters represented via centroids (i.e. average of points in a cluster) $c$:

$$
\bar{\mu}(c) = \frac{1}{|c|} \sum_{\bar{x} \in c} \bar{x}
$$

• Reassignment of instances to clusters is based on distance to the current cluster centroids.
K-Means Algorithm

• Input: $k = \text{number of clusters, distance measure } d$
• 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 min.
  • For each cluster $c_j$ //update the centroid of each cluster
    • $s_j = \mu(c_j)$
K-means Example

(k=2)

Pick seeds
Reassign clusters
Compute centroids
Reassign clusters
Compute centroids
Reassign clusters

Converged!
Time Complexity

• Assume computing distance between two instances is $O(N)$ where $N$ is the dimensionality of the vectors.
• Reassigning clusters for $n$ points: $O(kn)$ distance computations, or $O(knN)$.
• Computing centroids: Each instance gets added once to some centroid: $O(nN)$.
• Assume these two steps are each done once for $i$ iterations: $O(iknN)$.
• Linear in all relevant factors, assuming a fixed number of iterations, more efficient than HAC.
Buckshot Algorithm

Problem
- Results can vary based on random seed selection, especially for high-dimensional data.
- Some seeds can result in poor convergence rate, or convergence to sub-optimal clusterings.

- First randomly take a sample of instances of size \( n^{1/2} \)
- Run group-average HAC on this sample
- Use the results of HAC as initial seeds for K-means.
- Overall algorithm is efficient and avoids problems of bad seed selection.
Clustering as Prediction

• Setup
  – Learning Task: $P(X)$
  – Training Sample: $S = (\tilde{x}_1, ..., \tilde{x}_n)$
  – Hypothesis Space: $H = \{h_1, ..., h_{|H|}\}$ each describes $P(X|h_i)$ where $h_i$ are parameters
  – Goal: learn which $P(X|h_i)$ produces the data

• What to predict?
  – Predict where new points are going to fall
Gaussian Mixtures and EM

• Gaussian Mixture Models
  – Assume
  \[
P(X = \hat{x} | h_i) = \sum_{j=1}^{k} P(X = \hat{x} | Y = j, h_i) P(Y = j)
  \]
  where \( P(X = \hat{x} | Y = j, h) = \mathcal{N}(X = \hat{x} | \mu_j, \Sigma_j) \)
  and \( h = (\mu_1, ..., \mu_k, \Sigma_1, ..., \Sigma_k) \).

• EM Algorithm
  – Assume \( P(Y) \) and \( k \) known and \( \Sigma_i = 1 \).
  – REPEAT
    • \( \hat{\mu}_j = \frac{\sum_{i=1}^{n} P(Y = j | X = \hat{x}_i, \mu_j) \hat{x}_i}{\sum_{i=1}^{n} P(Y = j | X = \hat{x}_i, \mu_j)} \)
    • \( P(Y = j | X = \hat{x}_i, \mu_j) = \frac{P(X = \hat{x}_i | Y = j, \mu_j) P(Y = j)}{\sum_{l=1}^{k} P(X = \hat{x}_i | Y = l, \mu_l) P(Y = l)} = \frac{e^{-0.5(\hat{x}_i - \mu_j)^2} P(Y = j)}{\sum_{l=1}^{k} e^{-0.5(\hat{x}_i - \mu_l)^2} P(Y = l)} \)