Clustering

Class 14.  18 Oct 2011

How

- What is clustering
  - Clustering is the determination of naturally occurring grouping of data instances (with low within-group variability and high between-group variability)

Clustering

- What is clustering
  - Clustering is the determination of naturally occurring grouping of data instances (with low within-group variability and high between-group variability)

  - How is it done
    - Find groupings of data such that the groups optimize a "within-group variability" objective function of some kind
Clustering

What is clustering
- Clustering is the determination of naturally occurring grouping of data/instances (with low within-group variability and high between-group variability).

How is it done
- Find groupings of data such that the groups optimize a “within-group-variability” objective function of some kind.
- The objective function used affects the nature of the discovered clusters.
  - E.g. Euclidean distance and distance from center result in different clusters in this example.

Why Clustering

Automatic grouping into “Classes”
- Different clusters may show different behavior.

Quantization
- All data within a cluster are represented by a single point.

Preprocessing step for other algorithms
- Indexing, categorization, etc.

Clustering criteria

Compactness criterion
- Measure that shows how “good” clusters are.
  - The objective function.

Distance of a point from a cluster
- To determine the cluster a data vector belongs to.

“Compactness” criteria for clustering

Distance based measures
- Total distance between each element in the cluster and every other element in the cluster.
Distance based measures
- Total distance between each element in the cluster and every other element in the cluster
- Distance between the two farthest points in the cluster
- Total distance of every element in the cluster from the centroid of the cluster

Clustering: Distance from cluster
- How far is a data point from a cluster?
  - Euclidean or Minkowski distance from the centroid of the cluster

Distance measures are often weighted Minkowski metrics:
$$\text{dist} = \sqrt{\sum_{i=1}^{n} \left| x_i - y_i \right|^p}$$
Clustering: Distance from cluster

- How far is a data point from a cluster?
  - Euclidean or Minkowski distance from the centroid of the cluster
  - Distance from the closest point in the cluster
  - Distance from the farthest point in the cluster
  - Probability of data measured on cluster distribution

Optimal clustering: Exhaustive enumeration

- All possible combinations of data must be evaluated
  - If there are M data points, and we desire N clusters, the number of ways of separating M instances into N clusters is
    \[
    \frac{1}{M!} \sum_{i=1}^{M} (-1)^i \binom{M}{i} (M-i)^N
    \]
  - Exhaustive enumeration based clustering requires that the objective function (the "Goodness measure") be evaluated for every one of these, and the best one chosen
  - This is the only correct way of optimal clustering
  - Unfortunately, it is also computationally unrealistic

Not-quite non sequitur: Quantization

- Linear quantization (uniform quantization):
  - Each digital value represents an equally wide range of analog values
  - Regardless of distribution of data
  - Digital-to-analog conversion represented by a "uniform" table

<table>
<thead>
<tr>
<th>Signal Value</th>
<th>Bits</th>
<th>Mapped to</th>
</tr>
</thead>
<tbody>
<tr>
<td>S ≥ 3.75v</td>
<td>11</td>
<td>3 × const</td>
</tr>
<tr>
<td>3.75v &gt; S ≥ 2.5v</td>
<td>10</td>
<td>2 × const</td>
</tr>
<tr>
<td>2.5v &gt; S ≥ 1.25v</td>
<td>9</td>
<td>1 × const</td>
</tr>
<tr>
<td>1.25v &gt; S ≥ 0v</td>
<td>8</td>
<td>0</td>
</tr>
</tbody>
</table>

Analog value (arrows are quantization levels)
Not-quite non sequitur: Quantization

<table>
<thead>
<tr>
<th>Signal Value</th>
<th>Bits</th>
<th>Mapped to</th>
</tr>
</thead>
<tbody>
<tr>
<td>3v &gt; S &gt;= 4v</td>
<td>11</td>
<td>4.5</td>
</tr>
<tr>
<td>2.5v &gt; S &gt;= 3v</td>
<td>10</td>
<td>3.25</td>
</tr>
<tr>
<td>1.0v &gt; S &gt;= 0v</td>
<td>0</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Non-linear quantization:
- Each digital value represents a different range of analog values
- Finer resolution in high-density areas
- Mu-law / A-law assumes a Gaussian-like distribution of data
- Digital-to-analog conversion represented by a "non-uniform" table

Non-uniform quantization
- If data distribution is not Gaussianish?
  - Mu-law / A-law are not optimal
  - How to compute the optimal ranges for quantization
  - Or the optimal table

The Lloyd Quantizer
- Lloyd quantizer: An iterative algorithm for computing optimal quantization tables for non-uniformly distributed data
- Learned from "training" data

Lloyd Quantizer
- Randomly initialize quantization points
- Right column entries of quantization table
- Assign all training points to the nearest quantization point
- Draw boundaries
- Reestimate quantization points
Lloyd Quantizer

- Randomly initialize quantization points
- Right column entries of quantization table
- Assign all training points to the nearest quantization point
- Draw boundaries
- Reestimate quantization points
- Iterate until convergence

Generalized Lloyd Algorithm: K–means clustering

- K means is an iterative algorithm for clustering vector data
- General procedure:
  - Initially group data into the required number of clusters somehow (initialization)
  - Assign each data point to the closest cluster
  - Once all data points are assigned to clusters, redefine clusters
  - Iterate

K–means

- Problem: Given a set of data vectors, find natural clusters
- Clustering criterion is scatter: distance from the centroid
  - Every cluster has a centroid
  - The centroid represents the cluster
- Definition: The centroid is the weighted mean of the cluster
  - Weight = 1 for basic scheme

K–means

1. Initialize a set of centroids randomly
2. For each data point \( x \), find the distance from the centroid for each cluster
   - \( d_{\text{cluster}} = \text{distance}(x, m_{\text{cluster}}) \)
3. Put data point in the cluster of the closest centroid
   - Cluster for which \( d_{\text{cluster}} \) is minimum
K–means

1. Initialize a set of centroids randomly
2. For each data point \( x \), find the distance from the centroid for each cluster
   \[ d_{cluster} = \text{distance}(x, m_{cluster}) \]
3. Put data point in the cluster of the closest centroid
   • Cluster for which \( d_{cluster} \) is minimum
4. When all data points clustered, recompute cluster centroid
5. If not converged, go back to 2

\[
\sum_{i} x_i \]

37 18 Oct 2011

38 18 Oct 2011

39 18 Oct 2011

40 18 Oct 2011

41 18 Oct 2011
K-Means

1. Initialize a set of centroids randomly
2. For each data point $x$, find the distance from the centroid for each cluster
   - $d_{cluster} = \text{distance}(x, m_{cluster})$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $d_{cluster}$ is minimum
4. When all data points are clustered, recompute centroids
5. If not converged, go back to 2

**K-Means comments**

- The distance metric determines the clusters
  - In the original formulation, the distance is L2 distance
  - Euclidean norm, $w_i = 1$
  - If we replace every $x$ by $m_{cluster}(x)$, we get Vector Quantization
  - K-means is an instance of generalized EM
  - Not guaranteed to converge for all distance metrics

**Initialization**

- Random initialization
- Top-down clustering
  - Initially partition the data into two (or a small number of) clusters using K means
  - Partition each of the resulting clusters into two (or a small number of) clusters, also using K means
  - Terminate when the desired number of clusters is obtained

**K-Means for Top–Down clustering**

1. Start with one cluster
2. Split each cluster into two:
   - Perturb centroid of cluster slightly (by < 5%) to generate two centroids
K-Means for Top–Down clustering

1. Start with one cluster
2. Split each cluster into two:
   - Perturb centroid of cluster slightly (by < 5%) to generate two centroids
3. Initialize K means with new set of centroids
4. Iterate Kmeans until convergence
5. If the desired number of clusters is not obtained, return to 2

Non-Euclidean clusters

- Basic K-means results in good clusters in Euclidean spaces
  - Alternately stated, will only find clusters that are "good" in terms of Euclidean distances
- Will not find other types of clusters
For any vector $p \in \mathbb{R}^d$ we must modify the distance measure

- E.g. distance from a circle
- May be viewed as a distance in a higher dimensional space
  - I.e Kernel distances
  - Kernel K-means

Other related clustering mechanisms:
- Spectral clustering
- Non-linear weighting of adjacency
- Normalized cuts...

Distance in higher-dimensional space

- Transform the data into a synthetic higher-dimensional space
  - Where the desired patterns become natural clusters
- E.g. the quadratic transform above
- Problem: What is the function/space?
- Problem: Distances in higher dimensional space are more expensive to compute
- Yet only carry the same information in the lower-dimensional space

Distance in lower-dimensional space: A combination of dot products

- $|\|s_i - x_j\|_2^2 = (x_i - x_j)^T (x_i - x_j) = x_i x_j + x_j x_i - 2 x_i x_j$
- $d(x_i, x_j) = |\|\phi(x_i) - \phi(x_j)\|_2^2$
- $d(x_i, x_j)$ can be computed without computing $z$
  - Since it is a direct function of $x_i$ and $x_j$

$\Phi(x) = \{\phi_1(x), \phi_2(x), \phi_3(x), \ldots\}$

- $\phi_i(x)$ can be computed for any $x_i$ and $x_j$ without knowing $\phi_i()$
- $\phi_i(x)$, $\phi_j(x)$ can be computed without knowing $\phi_i()$

A property of the dot product

- For any vector $v$, $v^Tv = ||v||^2 > 0$
  - This is just the length of $v$ and is therefore non-negative
- For any vector $u = \sum_i a_i v_i$, $||u||^2 > 0$
  - $\Rightarrow (\sum_i a_i v_i)^T (\sum_i a_i v_i) > 0$
  - $\Rightarrow \sum_i a_i v_i v_i^T > 0$
- This holds for ANY real $\{a_1, a_2, \ldots\}$

A kernel function $K(x_i, x_j)$ is a function such that:

- $K(x_i, x_j) = \Phi(x_i), \Phi(x_j)$
- $d(x_i, x_j) = |\|\phi(x_i) - \phi(x_j)\|_2^2$
- But what is $K(x_i, x_j)$?
For each data point
Initialize the clusters with a
\( K \)-means

**Typical Kernel Functions**
- **Linear**: \( K(x, y) = x^T y + c \)
- **Polynomial**: \( K(x, y) = (ax^T y + c)^p \)
- **Gaussian**: \( K(x, y) = \exp(-||x-y||^2/\sigma^2) \)
- **Exponential**: \( K(x, y) = \exp(-||x-y||/\lambda) \)
- Several others
  - Choosing the right Kernel with the right parameters for your problem is an artform

**K–means**
- Initialize the clusters with a random set of \( K \) points
  - Cluster has 1 point
- For each data point \( x \), find the closest cluster
  \[
  \text{cluster}(x) = \min_{\text{cluster}} \left\{ d(x, \text{cluster}) = \min_{\text{cluster}} \| \Phi(x) - m_{\text{cluster}} \|^2 \right\}
  \]
  \[
  d(x, \text{cluster}) = \left\| \phi(x) - \Phi(m_{\text{cluster}}) \right\|^2 = \left\| \phi(x) - \Phi(C) \sum w_i \phi(x_i) \right\|^2
  \]
  \[
  = \phi(x)^T \phi(x) - 2 \Phi(C) \sum w_i \phi(x_i) \phi(x) + \Phi(C) \sum w_i \Phi(x_i)^T \Phi(x_i) \Phi(C)
  \]
  \[
  = K(x, x) - 2 \Phi(C) \sum w_i K(x_i, x) + \Phi(C) \sum w_i \Phi(x_i)^T \Phi(x_i) \Phi(C)
  \]
  \[
  = K(x, x) - 2 \sum \sum w_i w_j K(x_i, x_j) + \sum w_i \Phi(x_i)^T \Phi(x_i) \Phi(C)
  \]
  \[
  = K(x, x) - 2 \sum \sum w_i w_j K(x_i, x_j) + \sum w_i \Phi(x_i)^T \Phi(x_i) \Phi(C)
  \]
  Computed entirely using only the kernel function!

**Kernel K–means**
- Perform the K-mean in the Kernel space
  - The space of \( z = \phi(x) \)
  - The algorithm..

**The Mercer Condition**
- If \( \Phi(x) \) is a high-dimensional vector derived from \( x \) then for all real \{\( a_1, a_2, \ldots \)\} and any set \( \{x_1, x_2, \ldots \} = \{\phi(x_1), \phi(x_2), \ldots \} \)
  - \( \sum a_i a_j \phi(x_i) \phi(x_j) \geq 0 \)
  - \( \sum a_i a_j K(x_i, x_j) \geq 0 \)
- If \( K(x_1, x_2) = \phi(x_1), \phi(x_2) \)
  - \( \sum a_i a_j K(x_i, x_j) \geq 0 \)
- Any function \( K() \) that satisfies the above condition is a valid kernel function

**A corollary**: If any kernel \( K() \) satisfies the Mercer condition
\( d(x_1, x_2) = K(x_1, x_2) + K(x_2, x_2) - 2K(x_1, x_2) \)
satisfies the following requirements for a “distance”
- \( d(x, x) = 0 \)
- \( d(x, y) = 0 \)
- \( d(x, y) = d(x, y) \)

**K–means**
- Initialize a set of centroids

1. Initialize a set of centroids randomly

- Cluster has 1 point
- For each data point \( x \), find the closest cluster
  \[
  \text{cluster}(x) = \min_{\text{cluster}} \left\{ d(x, \text{cluster}) = \min_{\text{cluster}} \| \phi(x) - m_{\text{cluster}} \|^2 \right\}
  \]
  \[
  d(x, \text{cluster}) = \left\| \phi(x) - \Phi(m_{\text{cluster}}) \right\|^2 = \left\| \phi(x) - \Phi(C) \sum w_i \phi(x_i) \right\|^2
  \]
  \[
  = \phi(x)^T \phi(x) - 2 \Phi(C) \sum w_i \phi(x_i) \phi(x) + \Phi(C) \sum w_i \Phi(x_i)^T \Phi(x_i) \Phi(C)
  \]
  \[
  = K(x, x) - 2 \Phi(C) \sum w_i K(x_i, x) + \Phi(C) \sum w_i \Phi(x_i)^T \Phi(x_i) \Phi(C)
  \]
  \[
  = K(x, x) - 2 \sum \sum w_i w_j K(x_i, x_j) + \sum w_i \Phi(x_i)^T \Phi(x_i) \Phi(C)
  \]
  \[
  = K(x, x) - 2 \sum \sum w_i w_j K(x_i, x_j) + \sum w_i \Phi(x_i)^T \Phi(x_i) \Phi(C)
  \]
  Computed entirely using only the kernel function!
K–means

1. Initialize a set of centroids randomly

2. For each data point \( x \), find the distance from the centroid for each cluster
   - \( d_{\text{cluster}} = \text{distance}(x, m_{\text{cluster}}) \)

3. Put data point in the cluster of the closest centroid
   - Cluster for which \( d_{\text{cluster}} \) is minimum

4. When all data points clustered, recompute cluster centroid

5. If not converged, go back to 2

\[ \sum \in \text{cluster} \]
K–means
1. Initialize a set of centroids randomly
2. For each data point $x$, find the distance from the centroid for each cluster
   • $d_{\text{cluster}} = \text{distance}(x, m_{\text{cluster}})$
3. Put data point in the cluster of the closest centroid
   • Cluster for which $d_{\text{cluster}}$ is minimum
4. When all data points are clustered, recompute centroids
   $$m_{\text{new}} = \frac{1}{n_{\text{cluster}}} \sum_{x \in \text{cluster}} x$$
5. If not converged, go back to 2

Kernel K–means
1. Initialize a set of centroids randomly
2. For each data point $x$, find the distance from the centroid for each cluster
   • $d_{\text{cluster}} = \text{distance}(x, m_{\text{cluster}})$
3. Put data point in the cluster of the closest centroid
   • Cluster for which $d_{\text{cluster}}$ is minimum
4. When all data points clustered, recompute cluster centroid
$$m_{\text{new}} = \frac{1}{\sum_{i} n_{i}} \sum_{i} m_{i} n_{i}$$
5. If not converged, go back to 2

How many clusters?
- Assumptions:
  - Dimensionality of kernel space $>$ no. of clusters
  - Clusters represent separate directions in Kernel spaces
- Kernel correlation matrix $K$
  - $K_{ij} = K(x_i, x_j)$
- Find Eigen values $\Lambda$ and Eigen vectors $e$ of matrix $K$
  - No. of clusters = no. of dominant $\lambda_i(1e_i)$ terms

Spectral Methods
- “Spectral” methods attempt to find “principal” subspaces of the high-dimensional kernel space
- Clustering is performed in the principal subspaces
  - Normalized cuts
  - Spectral clustering
- Involves finding Eigenvectors and Eigen values of Kernel matrix
- Fortunately, provably analogous to Kernel K–means
Other clustering methods

- Regression based clustering
- Find a regression representing each cluster
- Associate each point to the cluster with the best regression
  - Related to kernel methods

Clustering..

- Many many other variants
- Many applications...

Important: Appropriate choice of feature
  - Appropriate choice of feature may eliminate need for kernel trick...
  - Google is your friend.