Unsupervised Learning

CS534
Supervised vs Unsupervised Learning

• So far we have assumed that the training examples were labeled with their class membership --- supervised learning

• We assume now that all one has is a collection of examples without being told their categories --- unsupervised learning
What can we learn from unlabeled data?

- Group of clusters in the data
- Low dimensional structure

PCA

Nonlinear embedding
Clustering

• Are there any groups in the data?
• How to group?
• How many groups?
What is clustering

• Clustering: the process of grouping a set of objects into classes of similar objects
  – high intra-class similarity
  – low inter-class similarity

• It is the most common form of unsupervised learning
Example Applications

• Find genes that are similar in their functions
• Group documents based on topics
• Categorize customers based on their buying habit
• Group images based on their contents
Issues for clustering

• What is a natural grouping among these objects?
  – Definition of "groupness"
• What makes objects “related”?
  – Definition of "similarity/distance"
• Representation for objects
  – Vector, normalization?
• How many clusters?
  – Fixed a priori?
  – Completely data driven?
  – Avoid “trivial” clusters - too large or small
• Clustering Algorithms
  – – Partition algorithms
  – – Hierarchical algorithms
What is a natural grouping among these objects?

By color? By pattern? By weight?

The definition of natural grouping is subjective.

This is why we call clustering exploratory data analysis.
What is similarity

Hard to define but
We know it when we see it

- The real meaning of similarity is a philosophical question. We will take a more pragmatic approach
  - Depends on representation and algorithm. For many rep./alg., easier to think in terms of a distance (rather than similarity) between vectors
What properties should a distance measure have?

- **Symmetry**: For any pair of objects \( A, B \), the distance from \( A \) to \( B \) should equal the distance from \( B \) to \( A \).
  
  \[ D(A, B) = D(B, A) \]

  Otherwise, we can say \( A \) looks like \( B \) but \( B \) does not look like \( A \).

- **Positivity and self-similarity**: The distance between any two distinct objects \( A, B \) must be positive, and the distance between an object and itself must be zero.
  
  \[ D(A, B) \geq 0, \quad D(A, A) = 0 \text{ iff } A = B \]

  Otherwise, there will be different objects that we cannot tell apart.

- **Triangle inequality**: For any three objects \( A, B, C \), the distance from \( A \) to \( C \) must be less than or equal to the sum of the distances from \( A \) to \( B \) and from \( B \) to \( C \).
  
  \[ D(A, B) + D(B, C) \geq D(A, C) \]

  Otherwise, one can say “\( A \) is like \( B \), \( B \) is like \( C \), but \( A \) is not like \( C \) at all.”
Distance Measures: Minkowski Metric

• Suppose two object $x$ and $y$ both have $d$ features
  $x = (x_1, \ldots, x_d), y = (y_1, \ldots, y_d)$
• The Minkowski metric of order $r$ is defined by
  $$d(x, y) = \sqrt[r]{\sum_i |x_i - y_i|^r}$$
• Common Minkowski metrics:
  
  – Euclidean ($r=2$): $d(x, y) = \sqrt{\sum_i (x_i - y_i)^2}$, also called $L_2$ distance
  
  – Manhattan distance ($r=1$): $d(x, y) = \sum_i |x_i - y_i|$, also called $L_1$ distance
  
  – “Sup” distance ($r = +\infty$): $d(x, y) = \max_i |x_i - y_i|$, also called $L_\infty$ distance
An Example

1: Euclidean distance: \( \sqrt{4^2 + 3^2} = 5 \).
2: Manhattan distance: \( 4 + 3 = 7 \).
3: "sup" distance: \( \max\{4,3\} = 4 \).
Other Distances

- Hamming distance (Manhattan distance on binary features)
  - # of features that differ
  - e.g.: distance of two sites based on their species composition

- Mahalanobis distance (assuming $x, y$ follows a Gaussian distribution with covariance matrix $\Sigma$

$$D(x, y) = \sqrt{(x - y)^T \Sigma^{-1} (x - y)}$$

```
sp1 sp2 sp3 sp4 sp5 sp6 sp7 sp8 sp9
Site A: 1 0 1 1 0 0 1 0 1
Site B: 0 0 1 0 1 1 1 0 1

D(A, B) = 4
```
Similarities

• Cosine similarities – commonly used to measure document similarity

\[
    \cos(x, x') = \frac{\langle x \cdot x' \rangle}{|x| \cdot |x'|}
\]

• Kernels – e.g., RBF (Gaussian) Kernel

\[
    S(X, X') = \exp \left( \frac{-|X - X'|^2}{2\sigma^2} \right)
\]
Clustering algorithms

• Hierarchical algorithms
  – Bottom up – agglomerative
  – Top down – divisive

• Partition algorithms (Flat)
  – K-means
  – Mixture of Gaussian
  – Spectral Clustering
Hierarchical Clustering

- Given a set of objects, build a tree-based taxonomy.

- Hierarchies are convenient way for organizing information, used frequently by web-portals.
Hierarchical Agglomerative Clustering (HAC)

- Starts with each object in a separate cluster
- Repeatedly joins the closest pair of clusters
- Until there is only one cluster

The history of merging forms a tree of hierarchy

**Question**: how to measure the “closeness” of two clusters?
Closest Pair of Clusters

The distance between two clusters is defined as the distance between:

- **Single-link**
  - The nearest pair of points

- **Complete-link**
  - The furthest pair of points

- **Centroid**
  - The center of gravity

- **Average-link**
  - Average of all cross-cluster pairs
Single Link Method

1. Diagram showing points a, b, c, d and a table showing distances between them.
2. Diagram showing points a, b, c, d and a table showing distances between them.
3. Diagram showing points a, b, c, d and a table showing distances between them.

Table:

<table>
<thead>
<tr>
<th></th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>2</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>b</td>
<td>3</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>c</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>d</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(1) Diagram shows points a, b, c, d connected by lines.
(2) Diagram shows points a, b, c, d connected by lines.
(3) Diagram shows points a, b, c, d connected by lines.
Complete Link Method

1. Start with four points labeled $a$, $b$, $c$, $d$.
2. Connect $a$ with $b$, $c$, and $d$.
3. Connect $c$ with $d$.

<table>
<thead>
<tr>
<th></th>
<th>$b$</th>
<th>$c$</th>
<th>$d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>2</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>$b$</td>
<td>3</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>$c$</td>
<td></td>
<td></td>
<td>4</td>
</tr>
</tbody>
</table>

(1) Connect $a$ with $b$, $c$, $d$.
(2) Connect $c$ with $d$.
(3) Complete the links between $a$, $b$, $c$, $d$. 

Diagram shows the progression from (1) to (3).
Visualization of the hierarchy: Dendrogram

- The distance between two clusters is represented as the length of the node that joins the two objects.
- Can be used to identify the number of clusters in data.
  - A horizontal cut will create a unique clustering.
  - Moving the cut from root down creates more clusters.
  - Large gaps between the merging nodes indicate a good cutting point.
Dendrograms

Single-Link

Complete-Link
Another example
Single Link vs. Complete Link

- Single-link creates straggly clusters due to chaining effect.
Computational Complexity

• All hierarchical clustering methods need to compute distance of all pairs of n individual instances which is $O(n^2)$

• There are $n-1$ iterations, at each iteration after the merge we must compute the distance between new cluster and all other clusters

\[ \sum_{i=2}^{n-1} n - i = O(n^2) \]

• In order to maintain an overall $O(n^2)$ performance, distance update must be done in constant time – trivial for complete-link and single-link
Partitional Clustering

• Given a data set of n points, we know that there are k clusters in the data, how to find these clusters?
• Roughly speaking there are $O(k^n)$ ways to partition the data, Which one is better?
• One intuition says that we want tight clusters, i.e., points should be in a tight ball
• This leads to the following objective function
  $$\sum_{i=1}^{k} \sum_{x \in C_i} |x - \mu_i|^2$$  --- squared distance between data point x and its cluster center
• Optimizing this objective is a combinatorial optimization problem
  – Exhaustive search for an optimal solution is not feasible
Combinatorial optimization: An iterative solution

- **Initialization**: Start with a random partition of the data
- **Iterative step**: the cluster assignments and cluster centers are updated to improve the objective
- **Stopping criterion**: if no improvement can be achieved.

Iterative greedy descent
- convergence is guaranteed, but to local optimal
K-Means

Algorithm

Input – Desired number of clusters, $k$

Initialize – the $k$ cluster centers (randomly if necessary)

Iterate –

1. Assigning each of the $N$ data points to its nearest cluster centers
2. Re-estimate the cluster center by assuming that the current assignment is correct

\[ \mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x \]

Termination –

If none of the data points changed membership in the last iteration, exit. Otherwise, go to 1
K-Means Example (K=2)

1. Pick seeds
2. Reassign clusters
3. Compute centroids
4. Reassign clusters
5. Compute centroids
6. Reassign clusters
7. Converged!
Computational Complexity

• At each iteration:
  – Reassigning clusters: $O(kn)$ distance computations
  – Computing centroids: Each instance vector gets added once to some centroid: $O(n)$

• Assume these two steps are each done once for $I$ iterations: $O(Ikn)$.

• Linear in all relevant factors, assuming a fixed number of iterations, more efficient than $O(n^2)$ HAC

• Does it always converge?
Kmeans Convergence

Objective

\[
\min_{\mu} \min_{c} \sum_{i=1}^{k} \sum_{x \in C_i} |x - \mu_i|^2
\]

1. Fix \( \mu \), optimize \( C \):

\[
\min_{c} \sum_{i=1}^{k} \sum_{x \in C_i} |x - \mu_i|^2 = \min_{c} \sum_{i} |x_i - \mu_{x_i}|^2
\]

2. Fix \( C \), optimize \( \mu \):

\[
\min_{\mu} \sum_{i=1}^{k} \sum_{x \in C_i} |x - \mu_i|^2
\]

– Take partial derivative of \( \mu_i \) and set to zero, we have

\[
\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x
\]

Kmeans takes an alternating optimization approach, each step is guaranteed to decrease the objective – thus guaranteed to converge.
Impact of Initial Seeds

- Highly sensitive to the initial seeds

- Multiple random trials: choose the one with best sum of squared loss (important!)

- Heuristics for choosing better centers
  - choose initial centers to be far apart – furthest first traversal
  - Initialize with results of other clustering method
More Comments

• K-Means is exhaustive:
  – Cluster every data point, no notion of outlier
  – Outliers cause problems
    • Become singular clusters
    • Bias the centroid estimation

• K-medoids methods is more robust to outliers
  – Cluster medoid: the point that has minimum sum squared distance to all data points in the cluster
  – More expensive to compute
    • For each pt: sum squared dist with all other pts in cluster \(O(|C|^2)\)

• Need to specify \(k\): difficult in practice
  – Automatically deciding \(k\)? more on this later...