94-775/95-865 Lecture 7: Clustering Part III

George Chen
Going from Similarities to Clusters

There's a whole zoo of clustering methods

Two main categories we'll talk about:

**Generative models**

1. Pretend data generated by specific model with parameters
2. Learn the parameters ("fit model to data")
3. Use fitted model to determine cluster assignments

**Hierarchical clustering**

Top-down: Start with everything in 1 cluster and decide on how to recursively split

Bottom-up: Start with everything in its own cluster and decide on how to iteratively merge clusters
Top-down: Divisive Clustering

0. Start with everything in the same cluster

1. Use a method to split the cluster (e.g., $k$-means, with $k = 2$)
Top-down: Divisive Clustering

0. Start with everything in the same cluster

1. Use a method to split the cluster (e.g., $k$-means, with $k = 2$)

2. Decide on next cluster to split (e.g., pick cluster with highest RSS)
Top-down: Divisive Clustering

1. Use a method to split the cluster (e.g., \(k\)-means, with \(k = 2\))

2. Decide on next cluster to split (e.g., pick cluster with highest RSS)

0. Start with everything in the same cluster
Top-down: Divisive Clustering

0. Start with everything in the same cluster

1. Use a method to split the cluster (e.g., $k$-means, with $k = 2$)

2. Decide on next cluster to split (e.g., pick cluster with highest RSS)
Top-down: Divisive Clustering

0. Start with everything in the same cluster

1. Use a method to split the cluster (e.g., $k$-means, with $k = 2$)

2. Decide on next cluster to split (e.g., pick cluster with highest RSS)
Top-down: Divisive Clustering

0. Start with everything in the same cluster

1. Use a method to split the cluster
   (e.g., \( k \)-means, with \( k = 2 \))

2. Decide on next cluster to split
   (e.g., pick cluster with highest RSS)
Top-down: Divisive Clustering

0. Start with everything in the same cluster

1. Use a method to split the cluster (e.g., $k$-means, with $k = 2$)

2. Decide on next cluster to split (e.g., pick cluster with highest RSS)
Top-down: Divisive Clustering

0. Start with everything in the same cluster

1. Use a method to split the cluster (e.g., $k$-means, with $k = 2$)

2. Decide on next cluster to split (e.g., pick cluster with highest RSS)

Stop splitting when some termination condition is reached (e.g., highest cluster RSS is small enough)
Top-down: Divisive Clustering

We can view the process in terms of a tree (colors are *not* important here and just help relate to the previous slide).

Each split is from $k$-means.
Top-down: Divisive Clustering

We can view the process in terms of a tree (colors are not important here and just help relate to the previous slide).

Each split is from $k$-means.

We could keep splitting until the leaves each have 1 point.
Top-down: Divisive Clustering

Divisive clustering uses *global* information and keeps splitting.

We could keep splitting until the leaves each have 1 point.

This tree is called a *dendrogram*, helpful for visualizing all the intermediate clustering stages.

Agglomerative clustering (bottom up) goes the other way.
Bottom-up: Agglomerative Clustering

0. Every point starts as its own cluster
Bottom-up: Agglomerative Clustering

0. Every point starts as its own cluster

1. Find the “most similar” two clusters
   (e.g., pick pair of clusters with closest cluster centers)
Bottom-up: Agglomerative Clustering

0. Every point starts as its own cluster

1. Find the “most similar” two clusters (e.g., pick pair of clusters with closest cluster centers)

2. Merge them
Bottom-up: Agglomerative Clustering

0. Every point starts as its own cluster

1. Find the “most similar” two clusters (e.g., pick pair of clusters with closest cluster centers)

2. Merge them
Bottom-up: Agglomerative Clustering

0. Every point starts as its own cluster

1. Find the “most similar” two clusters (e.g., pick pair of clusters with closest cluster centers)

2. Merge them
Bottom-up: Agglomerative Clustering

0. Every point starts as its own cluster

1. Find the “most similar” two clusters (e.g., pick pair of clusters with closest cluster centers)

2. Merge them
Bottom-up: Agglomerative Clustering

0. Every point starts as its own cluster

1. Find the “most similar” two clusters (e.g., pick pair of clusters with closest cluster centers)

2. Merge them
Bottom-up: Agglomerative Clustering

0. Every point starts as its own cluster

1. Find the “most similar” two clusters (e.g., pick pair of clusters with closest cluster centers)

2. Merge them
0. Every point starts as its own cluster.

1. Find the “most similar” two clusters (e.g., pick pair of clusters with closest cluster centers).

2. Merge them.
Bottom-up: Agglomerative Clustering

0. Every point starts as its own cluster

1. Find the “most similar” two clusters (e.g., pick pair of clusters with closest cluster centers)

2. Merge them
**Bottom-up: Agglomerative Clustering**

0. Every point starts as its own cluster.

1. Find the “most similar” two clusters (e.g., pick pair of clusters with closest cluster centers).

2. Merge them.
Bottom-up: Agglomerative Clustering

0. Every point starts as its own cluster

1. Find the “most similar” two clusters (e.g., pick pair of clusters with closest cluster centers)

2. Merge them
Bottom-up: Agglomerative Clustering

0. Every point starts as its own cluster

1. Find the “most similar” two clusters (e.g., pick pair of clusters with closest cluster centers)

2. Merge them
Bottom-up: Agglomerative Clustering

0. Every point starts as its own cluster

1. Find the “most similar” two clusters (e.g., pick pair of clusters with closest cluster centers)

2. Merge them
0. Every point starts as its own cluster

1. Find the “most similar” two clusters (e.g., pick pair of clusters with closest cluster centers)

2. Merge them
Bottom-up: Agglomerative Clustering

Don’t have to keep merging until there’s 1 cluster!
(e.g., stop when closest two clusters have distance between their centers exceed a threshold)

Agglomerative clustering uses local information and keeps merging
Bottom-up: Agglomerative Clustering

Some ways to define what it means for two clusters to be “close” (needed to find most similar clusters):

**Centroid linkage:** what we saw already (distance between cluster means)

**Single linkage:** use distance between closest points across the two clusters
Example: Single Linkage

What would single linkage merge next?

Distance between blue and green: 1
Distance between blue and red: 3
Distance between green and red: 1

Single linkage would merge either blue with green, or green with red.
Bottom-up: Agglomerative Clustering

Some ways to define what it means for two clusters to be “close” (needed to find most similar clusters):

**Centroid linkage:** what we saw already (distance between cluster means)

**Single linkage:** use distance between closest points across the two clusters

**Complete linkage:** use distance between farthest points across the two clusters
Example: Complete Linkage

What would complete linkage merge next?

Distance between blue and green: 3
Distance between blue and red: 6
Distance between green and red: 4

Complete linkage would merge blue and green
Bottom-up: Agglomerative Clustering

Some ways to define what it means for two clusters to be “close” (needed to find most similar clusters):

**Centroid linkage:** what we saw already (distance between cluster means)

**Single linkage:** use distance between closest points across the two clusters

**Complete linkage:** use distance between farthest points across the two clusters

There are other ways as well: none are perfect
Hierarchical Clustering

Demo
Bottom-up: Agglomerative Clustering

Some ways to define what it means for two clusters to be “close” (needed to find most similar clusters):

**Centroid linkage:** what we saw already (distance between cluster means)
- Ignores # items in each cluster

**Single linkage:** use distance between closest points across the two clusters
- Has “chaining” behavior

**Complete linkage:** use distance between farthest points across the two clusters
- Has “crowding” behavior

There are other ways as well: none are perfect
Going from Similarities to Clusters

There’s a whole zoo of clustering methods

Two main categories we'll talk about:

**Generative models**
1. Pretend data generated by specific model with parameters
2. Learn the parameters ("fit model to data")
3. Use fitted model to determine cluster assignments

**Hierarchical clustering**
Top-down: Start with everything in 1 cluster and decide on how to recursively split
Bottom-up: Start with everything in its own cluster and decide on how to iteratively merge clusters
Going from Similarities to Clusters

Generative models

1. Pretend data generated by specific model with parameters
2. Learn the parameters ("fit model to data")
3. Use fitted model to determine cluster assignments

Hierarchical clustering

Top-down: Start with everything in 1 cluster and decide on how to recursively split
Bottom-up: Start with everything in its own cluster and decide on how to iteratively merge clusters

The most popular models effectively assume Euclidean distance…

You learn a model
➔ can predict cluster assignments for points not seen in training

Easily works with different distances (not just Euclidean)

Great for problems that don’t need to predict clusters for future points

Different split/merge criteria lead to clusters that look specific ways (e.g., chaining, crowding)
Example: Clustering on U.S. Counties
(using opioid death rate data across 37 years)

No need to predict which cluster new counties should belong to, since we’re already looking at all U.S. counties!

Image source: Amanda Coston
How to Choose a Clustering Method?

In general: not easy!

Some questions to think about:

• What features to even cluster on?
• For your application, what distance/similarity makes sense?
• Do you care about cluster assignments for new points?

It’s possible that several clustering methods give similar results (which is great! — it means that there are some reasonably “stable” clusters in your data)

• Example: tons of clustering methods can figure out from senate voting data who Democrats and Republicans are (of course, without knowing each senator’s political party)
Clustering Last Remarks

Ultimately, *you* have to decide on which clustering method and number of clusters make sense for your data.

- After you run a clustering algorithm, make visualizations to interpret the clusters *in the context of your application*!
- Do not just blindly rely on numerical metrics (e.g., CH index)
- Some times it makes more sense to define your own score function for how good a clustering assignment is

If you can set up a prediction task, then you can use the prediction task to guide the clustering
Is Clustering Structure Enough?

User clusters

1

2

\[ k \]

Items

1

2

3

4

5

\[ \cdots \]

m
Is Clustering Structure Enough?

User clusters

\[ \text{k} \]

Items

\[ 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad \cdots \quad m \]

\[ \text{\ding{55}} \quad \text{\ding{55}} \quad \text{\ding{55}} \quad \text{\ding{55}} \quad \text{\ding{55}} \quad \cdots \quad \text{\ding{55}} \]

\[ \text{\ding{55}} \quad \text{\ding{55}} \quad \text{\ding{55}} \quad \text{\ding{55}} \quad \text{\ding{55}} \quad \cdots \quad \text{\ding{55}} \]

\[ \vdots \]

\[ \text{\ding{55}} \quad \text{\ding{55}} \quad \text{\ding{55}} \quad \text{\ding{55}} \quad \text{\ding{55}} \quad \cdots \quad \text{\ding{55}} \]
Is Clustering Structure Enough?

User clusters

1

2

k

Items

1 2 3 4 5 ••• m

1 2 3 4 5 ••• m

1 2 3 4 5 ••• m

What if these two users shared a Netflix account (and used the same user profile)?
Is Clustering Structure Enough?

In general: How do we handle when a user appears to belong to multiple clusters?

What if these two users shared a Netflix account (and used the same user profile)?
Topic Modeling

Movie recommendation
Each user is part of multiple “clusters”/topics
Each cluster/topic consists of a bunch of movies
(example clusters: “sci-fi epics”, “cheesy rom-coms”)

Text
Each document is part of multiple topics
Each topic consists of a bunch of regularly co-occurring words
(example topics: “sports”, “medicine”, “movies”, “finance”)

Health care
Each patient’s health records explained by multiple “topics”
Each topic consists of co-occurring “events”
(example topics: “heart condition”, “severe pancreatitis”)
Topic Modeling

Movie recommendation
Each user is part of multiple “clusters”/topics
Each cluster/topic consists of a bunch of movies
(example clusters: “sci-fi epics”, “cheesy rom-coms”)

In all of these examples:
• Each data point (a feature vector) is part of multiple topics
• Each topic corresponds to specific feature values in the feature vector likely appearing

Health care
Each patient’s health records explained by multiple “topics”
Each topic consists of co-occurring “events”
(example topics: “heart condition”, “severe pancreatitis”)

Each user is part of multiple “clusters”/topics
Each cluster/topic consists of a bunch of movies
(example clusters: “sci-fi epics”, “cheesy rom-coms”)

In all of these examples:
• Each data point (a feature vector) is part of multiple topics
• Each topic corresponds to specific feature values in the feature vector likely appearing

Health care
Each patient’s health records explained by multiple “topics”
Each topic consists of co-occurring “events”
(example topics: “heart condition”, “severe pancreatitis”)

Each patient’s health records explained by multiple “topics”
Each topic consists of co-occurring “events”
(example topics: “heart condition”, “severe pancreatitis”)
Latent Dirichlet Allocation (LDA)

- Easy to describe in terms of text (but works for not just text)
- A generative model
- Input: “document-word” matrix, and pre-specified # topics $k$
- Output: what the $k$ topics are (details on this shortly)

<table>
<thead>
<tr>
<th>Document</th>
<th>1</th>
<th>2</th>
<th>…</th>
<th>$d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>…</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$n$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$i$-th row, $j$-th column: # times word $j$ appears in doc $i$

- Output: what the $k$ topics are (details on this shortly)
Each word in Alice’s text is generated by:

1. Flip 2-sided coin for Alice
2. If weather: flip 4-sided coin for weather
   If food: flip 4-sided coin for food
Each word in Bob’s text is generated by:

1. Flip 2-sided coin for Bob
2. If weather: flip 4-sided coin for weather
   If food: flip 4-sided coin for food
Each word in doc. \( i \) is generated by:

1. Flip 2-sided coin for doc. \( i \)
2. If weather: flip 4-sided coin for weather
   If food: flip 4-sided coin for food

“Learning the topics” means figuring out these 4-sided coin probabilities.

LDA Example

<table>
<thead>
<tr>
<th>Topic</th>
<th>Document</th>
<th>Alice’s text</th>
<th>Bob’s text</th>
</tr>
</thead>
<tbody>
<tr>
<td>weather</td>
<td>0.1</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>food</td>
<td>0.9</td>
<td>0.5</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Topic</th>
<th>Word</th>
<th>cold</th>
<th>hot</th>
<th>apple</th>
<th>pie</th>
</tr>
</thead>
<tbody>
<tr>
<td>weather</td>
<td>0.3</td>
<td>0.7</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>food</td>
<td>0.1</td>
<td>0.3</td>
<td>0.5</td>
<td>0.1</td>
<td>0.1</td>
</tr>
</tbody>
</table>
LDA models each word in document $i$ to be generated as:

- Randomly choose a topic $Z$ (use topic distribution for doc $i$)
- Randomly choose a word (use word distribution for topic $Z$)

Goal: Learn these distributions
**LDA**

- Easy to describe in terms of text (but works for not just text)
- A generative model
- Input: “document-word” matrix, and pre-specified # topics $k$

<table>
<thead>
<tr>
<th>Word</th>
<th>1</th>
<th>2</th>
<th>…</th>
<th>$d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>…</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$n$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$i$-th row, $j$-th column: # times word $j$ appears in doc $i$

- Output: the $k$ topics’ distribution of words