Clustering in low dimensions

Jeremy Magland

1 November 2019
Flatiron-wide Algorithms and Mathematics (FWAM) conference
Objectives

● Focus on 2-d (for visualization purposes)
● Provide overview of popular clustering algorithms
  ○ Algorithmic concepts
  ○ Example performance
  ○ Instructions on how to run
  ○ Pros / cons
● Web application for comparison and evaluation
● Motivate further research - there is room for improvement
How many clusters do you see?
How many clusters do you see?
How many clusters do you see?
How many clusters do you see?
How many clusters do you see?
How many clusters do you see?
What is clustering?

Given points $x_1, \ldots, x_n$ in $R^d$, determine the number of clusters $K$ and integer labels $l_1, \ldots, l_n$, where $l_i \in \{1, \ldots, K\}$ such that...

Loosely speaking: similar points should be in the same cluster.

The classification should agree with the human eye in the obvious cases.
Spoiler

In two dimensions, I’m not sure if any existing clustering techniques can do as well as an eight-year-old human.

Good news: there is room for improvement
Step 1: Optimize the K cluster representatives.

Step 2: Optimize assignment to the K clusters.

Repeat until convergence

Objective:

For a fixed number of clusters K, minimize the sum of the squared distances to the cluster centroids.

Both steps reduce the following quantity:

$$\sum_{i=1}^{N} \|x_i - \mu_{k_i}\|^2$$

In practice, convergence is rapid, even though it is technically NP-hard in general.
K-means

Problem:

How to choose K?

In general, we don’t know ahead of time the number of clusters.
K-means

Step 1: Optimize the K cluster representatives.
Step 2: Optimize assignment to the K clusters.

Problem:
Unequal variances

K-means can fail to give the correct decision boundaries.
K-means

Problem:

Unequal populations

K-means can fail to separate sparse clusters.
I could have done better
How to run k-means?

From Python:

```python
from sklearn.cluster import KMeans
A = KMeans(n_clusters=n_clusters).fit(X)
labels = A.labels_
```

Notes:
- Fast, robust, simple
- Need to choose K ahead of time
- Assumptions about cluster shapes and sizes
Web application on colab notebook

Jeremy Magland, Ph.D.
Senior Data Scientist, Center for Computational Mathematics, Flatiron Institute

Miscellaneous links
- Style setting changes: [Style typedefs](https://users.flatironinstitute.org/~magland/)
- Example style: [matplotlib_magnolias](https://users.flatironinstitute.org/~magland/)
- Colab notebook: [Clustering demo](https://users.flatironinstitute.org/~magland/)

Miscellaneous projects under construction
- Style setting tools for Python, [Style typedefs](https://users.flatironinstitute.org/~magland/)
- Matplotlib style: [matplotlib.magnolias](https://users.flatironinstitute.org/~magland/)
- Custom matplotlib manager: [guppy](https://users.flatironinstitute.org/~magland/)
- Custom matplotlib style: [magnolias](https://users.flatironinstitute.org/~magland/)
- Style setting algorithm: [Magnetocentric](https://users.flatironinstitute.org/~magland/)

Try out the various algorithms with different parameters using google’s compute resources.
Public website

Evaluates 19 clustering algorithms on 25 datasets.
DBSCAN

Two parameters: $\epsilon$, minPts
A core neighborhood is an $\epsilon$-neighborhood containing at least minPts.
A cluster is the points in a connected union of core neighborhoods.
DBSCAN
DBSCAN

Has trouble with clusters of varying densities
No single choice of parameter works for all clusters
DBSCAN

I could have done better
How to run DBSCAN?

From Python:

```python
from sklearn.cluster import DBSCAN
A = DBSCAN(eps=3, min_samples=2).fit(X)
labels = A.labels_
```

**Notes:**

- Fast - even for large numbers of points
- Good for irregular cluster shapes
- Choice of neighborhood size and minPts is a problem
- Not good for higher # dimensions
Rodriguez-Laio

Distance to nearest point with higher density

Rodriguez and Laio, Science, 2014
Rodriguez-Laio
Rodriguez-Laio

No single choice of cutoff parameter works for all clusters

This technique looks promising. That example was cherry-picked to make R-L fail, so I think further investigation is required to determine whether I could do better in general.
How to run Rodriguez-Laio?

Not sure the best way to run Rodriguez-Laio in Python.

Some methods require input of the full distance matrix.

Also known as dpclust or density-peak clustering.

Someone should write a nice Python implementation for points in low dimensions.

Notes:
Further exploration required
Minimal assumptions about cluster shapes
How to choose the cutoff rule?
Certain cluster shapes are not handled well
Affinity propagation

Messages are passed between nearby data points to determine optimal “exemplars” for clusters.

Affinity propagation

Advantages

- Works with similarities between data points (no need to embed in vector space)
- No need to specify # clusters a priori

Affinity propagation

<table>
<thead>
<tr>
<th>chang_pathbased</th>
<th>chang_spiral</th>
<th>gionis_aggregation</th>
<th>synthetic_cassini</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL</td>
<td>URL</td>
<td>URL</td>
<td>URL</td>
</tr>
<tr>
<td>Num. points</td>
<td>Num. points</td>
<td>Num. points</td>
<td>Num. points</td>
</tr>
<tr>
<td>301</td>
<td>312</td>
<td>788</td>
<td>250</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>traenti_s3</th>
<th>fu_flame</th>
<th>synthetic_cuboid</th>
<th>synthetic_spirals</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL</td>
<td>URL</td>
<td>URL</td>
<td>URL</td>
</tr>
<tr>
<td>Num. points</td>
<td>Num. points</td>
<td>Num. points</td>
<td>Num. points</td>
</tr>
<tr>
<td>1000</td>
<td>240</td>
<td>250</td>
<td>250</td>
</tr>
</tbody>
</table>

I could have done better
How to run affinity propagation?

From Python:

```python
from sklearn.cluster import AffinityPropagation
A = AffinityPropagation(bandwidth=0.5).fit(X)
labels = A.labels_
```

Notes:

- Does not require embedding in vector space
- Good for irregular cluster shapes and variable sizes
- SLOW - not scalable with number of datapoints
- Seems to fail in simple cases
Spectral clustering

Was discussed earlier by Marina.
Spectral clustering (true K)

I could have done better
How to run spectral clustering?

From Python:

```python
from sklearn.cluster import SpectralClustering
A = SpectralClustering(n_clusters=5).fit(X)
labels = A.labels_
```

Notes:
 Mostly an embedding - requires a second step for labeling
 Medium scalability
Agglomerative clustering

- Begin with each point in its own cluster
- Merge clusters one at a time based on a measure of similarity
- Different options for similarity
  - **Single linkage** - based on closest pair of points
  - **Complete linkage** - based on furthest pair of points
  - **Average linkage** - based on average distances between all pairs
  - **Ward** - global objective function (minimizes within-cluster variances)
  - etc.
- Construct a dendrogram
- A second step is needed to obtain labels from dendrogram (e.g., specify K)
I could have done better
How to run agglomerative clustering?

From Python:

```python
from sklearn.cluster import AgglomerativeClustering
A = AgglomerativeClustering(n_clusters=5).fit(X)
labels = A.labels
```

Notes:
- Fast and scalable
- Does not require embedding in vector space
- Good for irregular cluster shapes and variable sizes
- Need to specify number of clusters or some other criteria for cutting dendrogram
- Many choices for linkage / merge criteria
Mean shift

- Clusters are basins of attraction for the mean shift operation
- Mean shift iteration:
  - Compute weighted centroid of nearby points
  - Move to centroid
- Requires choice of bandwidth / kernel function
Mean shift (bandwidth = auto)

I could have done better
How to run mean shift clustering?

From Python:

```python
from sklearn.cluster import MeanShift
A = MeanShift(bandwidth=3).fit(X)
labels = A.labels_
```

Notes:
Slow and not very good, it seems
ISO-SPLIT (1-d)

Data points with ISO-SPLIT hyperplane

Projection histogram with best unimodal fit and cut point

Empirical CDF with best unimodal fit

Dip score = 0.6  Dip score = 0.8  Dip score = 2.7  Dip score = 4.0  Dip score = 5.3

Accept unimodality hypothesis
Reject unimodality hypothesis and find optimal cutpoint
ISO-SPLIT (2-d)

Looks good to me!

But I think you were biased in picking those examples to make your own method look good.
## ISO-SPLIT

<table>
<thead>
<tr>
<th>Dataset</th>
<th>URL</th>
<th>Num. points</th>
</tr>
</thead>
<tbody>
<tr>
<td>chang_pathbased</td>
<td><a href="https://ckanexample.edu/d1/dataset/445">link</a></td>
<td>300</td>
</tr>
<tr>
<td>chang_spiral</td>
<td><a href="https://ckanexample.edu/d1/dataset/31">link</a></td>
<td>312</td>
</tr>
<tr>
<td>gionis_aggregation</td>
<td><a href="https://ckanexample.edu/d1/dataset/2149">link</a></td>
<td>788</td>
</tr>
<tr>
<td>synthetic_cassini</td>
<td><a href="https://ckanexample.edu/d1/dataset/2562">link</a></td>
<td>350</td>
</tr>
<tr>
<td>fraedi_s3</td>
<td><a href="https://ckanexample.edu/d1/dataset/1641">link</a></td>
<td>5000</td>
</tr>
<tr>
<td>fu_flame</td>
<td><a href="https://ckanexample.edu/d1/dataset/1954">link</a></td>
<td>240</td>
</tr>
<tr>
<td>synthetic_cuboid</td>
<td><a href="https://ckanexample.edu/d1/dataset/3551">link</a></td>
<td>250</td>
</tr>
<tr>
<td>synthetic_spirals</td>
<td><a href="https://ckanexample.edu/d1/dataset/2566">link</a></td>
<td>250</td>
</tr>
</tbody>
</table>

![I could have done better](image)

---

**Source:** FLATIRON INSTITUTE
Center for Computational Mathematics
How to run ISO-SPLIT?

From Python:

```python
from isosplit5 import isosplit5
labels = isosplit5(X)
```

**Notes:**
- Unimodal clusters
- No adjustable parameters
- Fast
- Not great for small datasets (statistical test benefits from large number of points)
- Cannot handle non-unimodal clusters
Did not cover mixture models

- Mixture models require *a priori* assumptions about cluster distributions
- Clustering fails when assumptions are not met
- Typically there are many adjustable parameters
- Overfitting via overlapping clusters
Summary

• Clustering in low dimensions is difficult to define
• A variety of techniques attempt to tackle the problem
• It appears that in simple 2-d examples, an eight-year-old child can do a better job than any of the techniques we explored
• There is room for improvement
• There is a web app to help decide which technique and parameters are suitable
• Future directions
  • Expand the web app (more datasets, more algorithms)
  • Improve ISO-SPLIT or create yet another alternative
Thank you for listening!

- [Colab notebook: Clustering](https://users.flatironinstitute.org/~magland/)