Graph Clustering Algorithms
Affinity Propagation and Spectral Clustering

Graph Representations of Datasets

Notation. Let $D = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\}$ be a set of d-dimensional data points: $\mathbf{x_i} = (x_{i1}, \ldots, x_{id})$ over a list of attributes: $\mathbf{X} = \{X_1, \ldots, X_d\}$.
We *make no assumptions* about the types of individual attributes X_i - they can be *numeric or categorical*.

Similarity function. We postulate the existence of a **similarity function**

$$s : (dom(X_1) \times \ldots \times dom(X_d)) \times (dom(X_1) \times \ldots \times dom(X_d)) \longrightarrow \mathbb{R},$$

which produces a *similarity score* for each pair of data points $\mathbf{x_i}, \mathbf{x_j}$. We require that s is symmetric:

$$s(\mathbf{x_i}, \mathbf{x_j}) = s(\mathbf{x_j}, \mathbf{x_i}).$$

We have no other hard contstraints, but under most circumstances the similarity function should also ensure that

$$s(\mathbf{x_i}, \mathbf{x_i}) > s(\mathbf{x_i}, \mathbf{x_j}) \text{ if } \mathbf{x_i} \neq \mathbf{x_j}.$$

If all attributes are numeric, this turns into

$$s : \mathbb{R}^d \times \mathbb{R}^d \longrightarrow \mathbb{R}.$$

Datasets as Graphs

Multidimensional datasets may be hard to work with:

- For $d > 4$, d-dimensional dataasets are hard to visualize even if all attributes are numeric.
- Mixed datasets (i.e., datasets with both numeric and categorical attributes) are even harder to visualize.
- Often the exact positioning of data points in d-dimensional space is irrelevant, but **proximity** between points is important.
- Some clustering algorithms (and some classification algorithms too!) do not require the exact dataset as input, they work with the matrix of pairwise similarities (or distances).

In all such cases, the dataset $\mathbf{X}$ can be represented in a form of a graph.

Definition 1 *Given a dataset $\mathbf{X} = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\}$ of d-dimensional data points, and a similarity function s, the graph representation of $\mathbf{X}$, denoted $G_{mathbfX}$ is a graph:*

$$G_{\mathbf{X}} = \langle \mathbf{X}, E, l \rangle,$$

where $E = \mathbf{X} \times \mathbf{X} - \{(\mathbf{x}, \mathbf{x}) | \mathbf{x} \in \mathbf{X}\}$ and $l : E \longrightarrow \mathbb{R}$ is the edge label computed as: $l((\mathbf{x_i}, \mathbf{x_j})) = s(\mathbf{x_i}, \mathbf{x_j})$.

Where clear, we omit the $\mathbf{X}$ subscript from $G_{\mathbf{X}}$ and simply write G. Additionally, without loss of generality we do not distinguish between $l()$ (technically a function defined on the set of edges) and $s()$ (technically a function defined on pairs of data points).

Informally, a **full graph representaion** of a dataset $\mathbf{X}$ is a graph that has an edge between every pair of **different** data points in $\mathbf{X}$[1] with a label equal to the similarity between the two data points.

Graph $G_{\mathbf{X}}$ is undirected. In some applications, this graph is turned into a directed graph in which each edge is arbitrarily ordered (in such applications, the actual ordering will not matter, but the mechanics of representing an ordered graph as a matrix will play a major role in the computations).

In some scenarios, the fully-connected graphs are not necessary. In these cases, graphs can be truncated.

Definition 2 *Given a dataset* $\mathbf{X} = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\}$ *of* d*-dimensional data points, a similarity function* s*, and a value* $\varepsilon > 0$*, the* ε*-truncated graph representation of* $\mathbf{X}$*, denoted* $G_{\mathbf{X},\varepsilon}$ *is a graph:*

$$G_{\mathbf{X},\varepsilon} = \langle \mathbf{X}, E, l \rangle,$$

where $E = \{(\mathbf{x_i}, \mathbf{x_j}) | \mathbf{x_i}, \mathbf{x_j} \in \mathbf{X}, s(\mathbf{x_i}, \mathbf{x_j}) \geq \varepsilon\}$ *and* $l : E \longrightarrow \mathbb{R}$ *is the edge label computed as:* $l((\mathbf{x_i}, \mathbf{x_j})) = s(\mathbf{x_i}, \mathbf{x_j})$.

Definition 3 *Given a dataset* $\mathbf{X} = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\}$ *of* d*-dimensional data points, a similarity function* s*, and an integer number* $k > 0$*, the KNN-truncated graph representation of* $\mathbf{X}$*, denoted* $G_{\mathbf{X},k}$ *is a directed graph:*

$$G_{\mathbf{X}}, k = \langle \mathbf{X}, E, l \rangle,$$

where $E = \{(\mathbf{x_i}, \mathbf{x_j}) |$ $\mathbf{x_i}$ *is included in* k *nearest neighbors of* $\mathbf{x_k}$*, and* $\mathbf{x_j}$ *is included in* k *nearest neighbors of* $\mathbf{x_i}\}$ *and* $l : E \longrightarrow \mathbb{R}$ *is the edge label computed as:* $l((\mathbf{x_i}, \mathbf{x_j})) = s(\mathbf{x_i}, \mathbf{x_j})$.

Graphs As Datasets

In some situations, the **inverse problem** is of interest:

> given a graph $G = \langle V, E, l \rangle$ with edge labels representing similarity between the nodes of the graph, can we find a mapping $f : V \longrightarrow \mathbb{R}^d$ for **some** integer value $d > 0$, that preserves the similarities between the points: i.e.,
>
> $$(\forall x_i, x_j \in V) l(x_i, x_j) \approx s(f(x_i), f(x_j)),$$
>
> for some predetermined similarity function $s : \mathbb{R}^d \times \mathbb{R}^d \longrightarrow \mathbb{R}$.

This problem has a number of important practical applications, including, but not limited to, the problem of visualizing a given graph in 2D or 3D space - something that numerous graph rendering packages need to solve.

This problem can be expressed as an optimization problem.

Given $V = (v_1, \ldots, v_n)$, find $d > 0$ and parameters

$$\mathbf{x_1} = (x_{11}, \ldots x_{1d})$$
$$\mathbf{x_2} = (x_{21}, \ldots x_{2d})$$
$$\ldots$$
$$\mathbf{x_n} = (x_{n1}, \ldots x_{nd})$$

that **minimize** the sum of squared error:

$$Error_G = \sum_{(v_i, v_j) \in E} (l(v_i, v_j) - s(\mathbf{x_i}, \mathbf{x_j}))^2 \longrightarrow \min$$

[1]Note, that $\mathbf{X}$ is *not a set*, and therefore duplicate data points can occur - such data points will be connected by an edge.

Clustering via Affinity Propagation

Affinity Propagation Clustering is a clustering algorithm that relies on *message-passing* between data points in the dataset $\mathbf{X}$.

While in the most general case, the message passing can take place between every pair of data points, we often prefer to restrict the message passing to the specific *vicinities* of each point, and use a graph $G_{\mathbf{X},\varepsilon}$ or $G_{\mathbf{X},k}$ that restricts the edges.

Idea. The Affinity Propagation clustering can be viewed as a version of K-means clustering that (a) does not require K as an input parameter[2], and (b) upgrades the "negotiation" process between points and cluster centroids.

In case of affinity propagation, each cluster is represented by some point $\mathbf{x} \in \mathbf{X}$ (i.e., no points outside of $\mathbf{X}$, like centroids, are constructed).

Affinity propagation is an iterative technique. On each iteration, two "complementary" messages are exchanged between each pair of connected data points. That is, for each pair of connected data point $\mathbf{x_i}, \mathbf{x_j}$ two values: $r(i,j)$ and $a(i,j)$ are computed representing respectively:

- $r(i,j)$: responsibility of data point $\mathbf{x_j}$ for representing, as a cluster center, the data point $\mathbf{x_i}$. The higher the value, the more desirable is it for the algorithm to assign $\mathbf{x_j}$ to be a cluster representative and to have $\mathbf{x_i}$ in its cluster. (you can think of this informally as a message from $\mathbf{x_i}$ to $\mathbf{x_j}$ asking $\mathbf{x_j}$ to responsibility for representing it)
- $a(i,j)$: availability of data point $\mathbf{x_j}$ to represent $\mathbf{x_i}$ in a cluster. This can be thought of as the message from $\mathbf{x_j}$ to $\mathbf{x_i}$ in response to the "responsibility" request.

Affinity Propagation. On iteration $t+1$ assume that values $a^{(t)}(i,j)$ are computed and available. The affinity propagation computation proceeds as follows.

- **Step 1.** Propagate responsibility:

$$r^{(t+1)}(i,j) \leftarrow s(\mathbf{x_i}, \mathbf{x_j}) - \max_{k \neq j} \left(a^{(t)}(i,k) + s(i,k) \right)$$

- **Step 2.** Propagate availability for pairs of different points:

$$a^{(t+1)}(i,j) \leftarrow \min \left(0, r^{(t+1)}(j,j) + \sum_{k \neq i} \max(0, r(k,j) \right)$$

- **Step 2.5.** Update self-availability for the point:

$$a^{(t+1)}(j,j) \leftarrow \sum_{k \neq i} \max(0, r(k,j)$$

[2]The number of clusters to generate is controlled via a different set of input parameters, that act in more subtle ways.