Self-Tuning Spectral Clustering

Sam Bowyer

Statistical Methods 2

24th February 2023

Table of Contents

Basic Spectral Clustering Algorithm

To cluster n data points $\{\mathbf{x}_i\}_{i=1}^{n}$ into K clusters:

1. Form the affinity/similarity matrix $W \in \mathbb{R}^{n\times n}$ where $W_{ij} = \exp\left(-\frac{d^2(\mathbf{x}_i,\mathbf{x}_j)}{\sigma^2}\right)$ for $i \neq j$ and $W_{ii} = 0$.

[Ng et al., 2001]

Basic Spectral Clustering Algorithm

To cluster n data points $\{\mathrm{x}_i\}_{i=1}^{n}$ into K clusters:

1. Form the affinity/similarity matrix $W \in \mathbb{R}^{n\times n}$ where $W_{ij} = \exp\left(-\frac{d^2(\mathrm{x}_i, \mathrm{x}_j)}{\sigma^2}\right)$ for $i \neq j$ and $W_{ii} = 0$.
2. Let G be the diagonal matrix with $G_{ii} = \sum_{j=1}^{n} W_{ij}$ (the degree of x_i). Let $\tilde{L} := G^{-1/2} W G^{-1/2}$ be the symmetric normalized Laplacian.

[Ng et al., 2001]

Basic Spectral Clustering Algorithm

To cluster n data points $\{\mathrm{x}_i\}_{i=1}^n$ into K clusters:

1. Form the affinity/similarity matrix $W \in \mathbb{R}^{n \times n}$ where $W_{ij} = \exp\left(-\frac{d^2(\mathrm{x}_i, \mathrm{x}_j)}{\sigma^2}\right)$ for $i \neq j$ and $W_{ii} = 0$.
2. Let G be the diagonal matrix with $G_{ii} = \sum_{j=1}^n W_{ij}$ (the degree of x_i). Let $\tilde{L} := G^{-1/2} W G^{-1/2}$ be the symmetric normalized Laplacian.
3. Form the matrix $Z \in \mathbb{R}^{n \times K}$ by stacking the eigenvectors corresponding to the K largest eigenvalues of $\tilde{L}$.

[Ng et al., 2001]

Basic Spectral Clustering Algorithm

To cluster n data points $\{\mathrm{x}_i\}_{i=1}^n$ into K clusters:

1. Form the affinity/similarity matrix $W \in \mathbb{R}^{n \times n}$ where $W_{ij} = \exp\left(-\frac{d^2(\mathrm{x}_i, \mathrm{x}_j)}{\sigma^2}\right)$ for $i \neq j$ and $W_{ii} = 0$.
2. Let G be the diagonal matrix with $G_{ii} = \sum_{j=1}^n W_{ij}$ (the degree of x_i). Let $\tilde{L} := G^{-1/2} W G^{-1/2}$ be the symmetric normalized Laplacian.
3. Form the matrix $Z \in \mathbb{R}^{n \times K}$ by stacking the eigenvectors corresponding to the K largest eigenvalues of $\tilde{L}$.
4. Cluster the rows of Z using K-means: assign x_i to cluster $k \in [K] := \{1, \ldots, K\}$ if and only if row i of Z was assigned to cluster k.

[Ng et al., 2001]

Parameters To Tune

Problem: we have to choose K and σ.

Parameters To Tune

Problem: we have to choose K and σ.

Self-Tuning Spectral Clustering [Zelnik-Manor and Perona, 2004] contributions:

(i) Selection of the appropriate scale to analyse the data.

(ii) Clustering data distributed at different scales.

(iii) Clustering with irregular background clutter.

(iv) Estimating the number of clusters.

Parameters To Tune

Problem: we have to choose K and σ.

Self-Tuning Spectral Clustering [Zelnik-Manor and Perona, 2004] contributions:

(i) Selection of the appropriate scale to analyse the data.

(ii) Clustering data distributed at different scales.

(iii) Clustering with irregular background clutter.

(iv) Estimating the number of clusters.

Selecting a suitable σ will resolve (i)–(iii), whilst (iv) is just choosing K.

Selecting σ

Note that σ can be very sensitive.

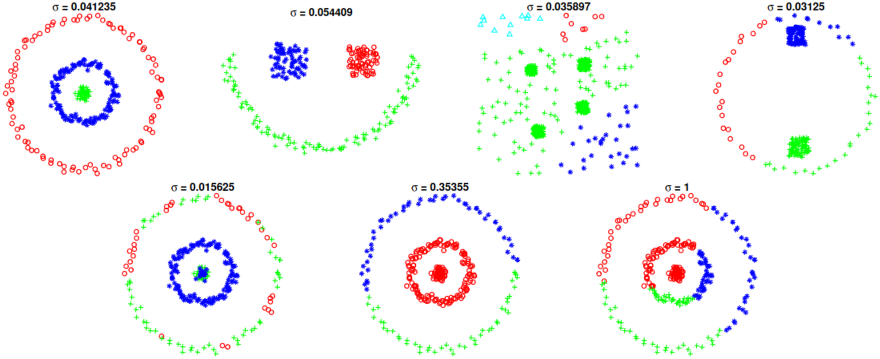


Figure: Fig. 1 from [Zelnik-Manor and Perona, 2004].

Local Scaling

Furthermore, different bandwidths σ may be needed for different clusters which are distributed at different scales. So we introduce local scaling:

Local Scaling

Furthermore, different bandwidths σ may be needed for different clusters which are distributed at different scales. So we introduce local scaling:

1. For each data point $\mathtt{x}_i$, define $\sigma_i = d(x_i, x_P)$, i.e. the distance from x_i to its Pth nearest neighbour, x_P. ($P = 7$ suggested.)

Local Scaling

Furthermore, different bandwidths σ may be needed for different clusters which are distributed at different scales. So we introduce local scaling:

1. For each data point $\mathbf{x}_i$, define $\sigma_i = d(x_i, x_P)$, i.e. the distance from x_i to its Pth nearest neighbour, x_P. ($P = 7$ suggested.)
2. The *locally scaled* affinity matrix's non-diagonal entries are given by $\hat{W}_{ij} = \exp\left(-\frac{d^2(x_i, x_j)}{\sigma_i \sigma_j}\right)$.

Local Scaling

Furthermore, different bandwidths σ may be needed for different clusters which are distributed at different scales. So we introduce local scaling:

1. For each data point x_i, define $\sigma_i = d(x_i, x_P)$, i.e. the distance from x_i to its Pth nearest neighbour, x_P. ($P = 7$ suggested.)
2. The *locally scaled* affinity matrix's non-diagonal entries are given by $\hat{W}_{ij} = \exp\left(-\frac{d^2(x_i,x_j)}{\sigma_i\sigma_j}\right)$.

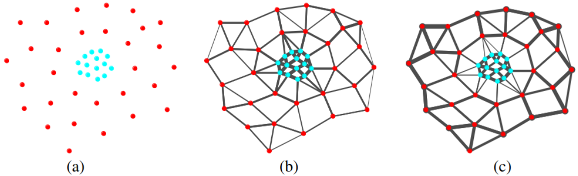


Figure: Fig. 2 from [Zelnik-Manor and Perona, 2004]: (a) Input data; (b) unscaled affinity; (c) locally-scaled affinity (edge thickness represents weight).

Selecting K

In the ideal case that there are some K completely disconnected clusters (i.e. $\hat{W}_{ij} > 0$ if and only if x_i and x_j are in the same cluster), then the eigenvalues of $\tilde{L}$, given by $\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_n$, will be such that:

$$1 = \lambda_1 = \cdots = \lambda_K > \lambda_{K+1} \geqslant \cdots \geqslant \lambda_n \geqslant 0.$$

This leads to the "eigengap heuristic" for choosing K: choose K to be the smallest integer such that $\lambda_{K+1} - \lambda_K$ is large.

Selecting K

In the ideal case that there are some K completely disconnected clusters (i.e. $\hat{W}_{ij} > 0$ if and only if x_i and x_j are in the same cluster), then the eigenvalues of $\tilde{L}$, given by $\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_n$, will be such that:

$$1 = \lambda_1 = \cdots = \lambda_K > \lambda_{K+1} \geqslant \cdots \geqslant \lambda_n \geqslant 0.$$

This leads to the "eigengap heuristic" for choosing K: choose K to be the smallest integer such that $\lambda_{K+1} - \lambda_K$ is large.

However, with real, noisy data, the above may not hold ("lacks a theoretical justification" [Zelnik-Manor and Perona, 2004])—so instead of looking at the eigenvalues, we'll look further into the structure of $\tilde{L}$'s eigenvectors.

Analysing the Eigenvectors

In our ideal case (i.e. with K completely disconnected clusters), $\tilde{L}$ is block diagonal, with each block $\tilde{L}^{(k)}$ corresponding to a cluster $k \in [K]$.

Analysing the Eigenvectors

In our ideal case (i.e. with K completely disconnected clusters), $\tilde{L}$ is block diagonal, with each block $\tilde{L}^{(k)}$ corresponding to a cluster $k \in [K]$. Therefore when we construct $Z \in \mathbb{R}^{n \times K}$ by stacking the eigenvectors of $\tilde{L}$ corresponding to the K largest eigenvalues of $\tilde{L}$, we obtain

$$Z = \begin{bmatrix} \mathrm{v}^{(1)} & \vec{0} & \vec{0} \\ \vec{0} & \dots & \vec{0} \\ \vec{0} & \vec{0} & \mathrm{v}^{(K)} \end{bmatrix} \in \mathbb{R}^{n \times K}$$

where $\mathrm{v}^{(k)}$ is the eigenvector corresponding to the largest eigenvalue (i.e. 1) of the submatrix $\tilde{L}^{(k)}$.

Analysing the Eigenvectors

$$Z = \begin{bmatrix} \mathrm{v}^{(1)} & \vec{0} & \vec{0} \\ \vec{0} & \dots & \vec{0} \\ \vec{0} & \vec{0} & \mathrm{v}^{(K)} \end{bmatrix} \in \mathbb{R}^{n \times K}$$

1. Each row i of Z has only one nonzero entry, lying in the column j corresponding to x_i's cluster—making the remaining task of cluster allocation trivial.

Analysing the Eigenvectors

$$Z = \begin{bmatrix} \mathrm{v}^{(1)} & \vec{0} & \vec{0} \\ \vec{0} & \dots & \vec{0} \\ \vec{0} & \vec{0} & \mathrm{v}^{(K)} \end{bmatrix} \in \mathbb{R}^{n \times K}$$

1. Each row i of Z has only one nonzero entry, lying in the column j corresponding to x_i's cluster—making the remaining task of cluster allocation trivial.
2. Taking more than K eigenvectors would result in more than one nonzero entry in some rows of Z, whilst taking fewer would result in some rows of Z having no nonzero entries.

Analysing the Eigenvectors

$$Z = \begin{bmatrix} \mathrm{v}^{(1)} & \vec{0} & \vec{0} \\ \vec{0} & \dots & \vec{0} \\ \vec{0} & \vec{0} & \mathrm{v}^{(K)} \end{bmatrix} \in \mathbb{R}^{n \times K}$$

1. Each row i of Z has only one nonzero entry, lying in the column j corresponding to x_i's cluster—making the remaining task of cluster allocation trivial.
2. Taking more than K eigenvectors would result in more than one nonzero entry in some rows of Z, whilst taking fewer would result in some rows of Z having no nonzero entries.

With noisy, non-ideal data the rows of Z may have more than one nonzero entry, but there should hopefully be one entry significantly larger than the others.
Assign x_i to the cluster $k = \operatorname{argmax}_j Z_{ij}^2$.

A Problem With Z

Problem: The eigensolver we use may not return the K eigenvectors of $\tilde{L}$ in the standard basis—it could return any set of orthonormal vectors spanning the same space as Z's columns.

A Problem With Z

Problem: The eigensolver we use may not return the K eigenvectors of $\tilde{L}$ in the standard basis—it could return any set of orthonormal vectors spanning the same space as Z's columns.

Solution: Find the rotation matrix $R \in \mathbb{R}^{K \times K}$ that minimises the cost function

$$J_K = \sum_{i=1}^{n} \sum_{j=1}^{K} \frac{\hat{Z}_{ij}^2}{\max_l (\hat{Z}_{il}^2)}$$

where $\hat{Z} = ZR$ is the rotated matrix of eigenvectors.

A Problem With Z

Problem: The eigensolver we use may not return the K eigenvectors of $\tilde{L}$ in the standard basis—it could return any set of orthonormal vectors spanning the same space as Z's columns.
Solution: Find the rotation matrix $R \in \mathbb{R}^{K \times K}$ that minimises the cost function

$$J_K = \sum_{i=1}^{n} \sum_{j=1}^{K} \frac{\hat{Z}_{ij}^2}{\max_l(\hat{Z}_{il}^2)}$$

where $\hat{Z} = ZR$ is the rotated matrix of eigenvectors.

Minimising this attempts to make the rows of $\hat{Z}$ as close to the standard basis as possible (i.e. with only one nonzero entry) and can be done via gradient descent (see [Zelnik-Manor and Perona, 2004] for details).

Selecting K from $\hat{Z}$

Choosing some large K' we can calculate the value of J_K for all $K \in [K']$ efficiently as follows:

Selecting K from $\hat{Z}$

Choosing some large K' we can calculate the value of J_K for all $K \in [K']$ efficiently as follows:

1. Let $Z \in \mathbb{R}^{n \times 2}$ be the matrix of eigenvectors of $\tilde{L}$ corresponding to the two largest eigenvalues of $\tilde{L}$.

Selecting K from $\hat{Z}$

Choosing some large K' we can calculate the value of J_K for all $K \in [K']$ efficiently as follows:

1. Let $Z \in \mathbb{R}^{n \times 2}$ be the matrix of eigenvectors of $\tilde{L}$ corresponding to the two largest eigenvalues of $\tilde{L}$.
2. Find the rotation R that minimises J_K on $Z \in \mathbb{R}^{n \times 2}$ to obtain $\hat{Z}$.

Selecting K from $\hat{Z}$

Choosing some large K' we can calculate the value of J_K for all $K \in [K']$ efficiently as follows:

1. Let $Z \in \mathbb{R}^{n \times 2}$ be the matrix of eigenvectors of $\tilde{L}$ corresponding to the two largest eigenvalues of $\tilde{L}$.
2. Find the rotation R that minimises J_K on $Z \in \mathbb{R}^{n \times 2}$ to obtain $\hat{Z}$.
3. Add the eigenvector corresponding to the next largest eigenvalue of $\tilde{L}$ as a column to $\hat{Z}$ and find the rotation R that minimises J_K for $K = 3$.

Selecting K from $\hat{Z}$

Choosing some large K' we can calculate the value of J_K for all $K \in [K']$ efficiently as follows:

1. Let $Z \in \mathbb{R}^{n\times 2}$ be the matrix of eigenvectors of $\tilde{L}$ corresponding to the two largest eigenvalues of $\tilde{L}$.
2. Find the rotation R that minimises J_K on $Z \in \mathbb{R}^{n\times 2}$ to obtain $\hat{Z}$.
3. Add the eigenvector corresponding to the next largest eigenvalue of $\tilde{L}$ as a column to $\hat{Z}$ and find the rotation R that minimises J_K for $K = 3$.
4. Repeat until we've found $J_{K'}$.

Selecting K from $\hat{Z}$

Choosing some large K' we can calculate the value of J_K for all $K \in [K']$ efficiently as follows:

1. Let $Z \in \mathbb{R}^{n \times 2}$ be the matrix of eigenvectors of $\tilde{L}$ corresponding to the two largest eigenvalues of $\tilde{L}$.
2. Find the rotation R that minimises J_K on $Z \in \mathbb{R}^{n \times 2}$ to obtain $\hat{Z}$.
3. Add the eigenvector corresponding to the next largest eigenvalue of $\tilde{L}$ as a column to $\hat{Z}$ and find the rotation R that minimises J_K for $K = 3$.
4. Repeat until we've found $J_{K'}$.

Finally, we choose $K_{\text{best}} = \text{argmin}_{K \in [K']} J_K$. (Although, if several Ks have very similar costs—e.g. within 0.01% of each other—then choose the largest of these Ks.)

Cluster Allocation

Now that we have K_{best}, using the rotated matrix $\hat{Z} \in \mathbb{R}^{n \times K_{\text{best}}}$ we can allocate each data point x_i to a cluster k in one of two ways:

Cluster Allocation

Now that we have K_{best}, using the rotated matrix $\hat{Z} \in \mathbb{R}^{n \times K_{\text{best}}}$ we can allocate each data point x_i to a cluster k in one of two ways:

1. Assign x_i to the cluster $k = \text{argmax}_j \, \hat{Z}_{ij}^2$.

Cluster Allocation

Now that we have K_{best}, using the rotated matrix $\hat{Z} \in \mathbb{R}^{n \times K_{\text{best}}}$ we can allocate each data point x_i to a cluster k in one of two ways:

1. Assign x_i to the cluster $k = \text{argmax}_j \hat{Z}_{ij}^2$.
2. As in the old algorithm, perform K-means on the rows of $\hat{Z}$ to find the clusters (this should converge fairly quickly since $\hat{Z}$ is likely to be a good initialisation). This is particularly useful for very noisy data.

Self-Tuning Spectral Clustering Algorithm

1. Compute the local scale σ_i for each data point x_i.

[Zelnik-Manor and Perona, 2004]

Self-Tuning Spectral Clustering Algorithm

1. Compute the local scale σ_i for each data point x_i.
2. Form the locally scaled affinity matrix $\hat{W} \in \mathbb{R}^{n \times n}$ where $\hat{W}_{ij} = \exp\left(-\frac{d^2(\mathbf{x}_i, \mathbf{x}_j)}{\sigma_i \sigma_j}\right)$ for $i \neq j$ and $\hat{W}_{ii} = 0$.

[Zelnik-Manor and Perona, 2004]

Self-Tuning Spectral Clustering Algorithm

1. Compute the local scale σ_i for each data point x_i.
2. Form the locally scaled affinity matrix $\hat{W} \in \mathbb{R}^{n \times n}$ where $\hat{W}_{ij} = \exp\left(-\frac{d^2(\mathbf{x}_i, \mathbf{x}_j)}{\sigma_i \sigma_j}\right)$ for $i \neq j$ and $\hat{W}_{ii} = 0$.
3. Let G be the diagonal matrix with $G_{ii} = \sum_{j=1}^{n} \hat{W}_{ij}$ (the degree of x_i) and $\tilde{L} := G^{-1/2}\, \hat{W}\, G^{-1/2}$ be the symmetric normalized Laplacian.

[Zelnik-Manor and Perona, 2004]

Self-Tuning Spectral Clustering Algorithm

1. Compute the local scale σ_i for each data point x_i.
2. Form the locally scaled affinity matrix $\hat{W} \in \mathbb{R}^{n \times n}$ where $\hat{W}_{ij} = \exp\left(-\frac{d^2(\mathbf{x}_i, \mathbf{x}_j)}{\sigma_i \sigma_j}\right)$ for $i \neq j$ and $\hat{W}_{ii} = 0$.
3. Let G be the diagonal matrix with $G_{ii} = \sum_{j=1}^{n} \hat{W}_{ij}$ (the degree of x_i) and $\tilde{L} := G^{-1/2} \hat{W} G^{-1/2}$ be the symmetric normalized Laplacian.
4. For some large K', form the matrix $Z \in \mathbb{R}^{n \times K'}$ by stacking the eigenvectors corresponding to the K' largest eigenvalues of $\tilde{L}$ and use gradient descent to find the rotation matrix $R \in \mathbb{R}^{K' \times K'}$ that minimizes $J_{K'} = \sum_{i=1}^{n} \sum_{j=1}^{K'} (Z_{ij}^2 / \max_l Z_{il}^2)$.

[Zelnik-Manor and Perona, 2004]

Self-Tuning Spectral Clustering Algorithm

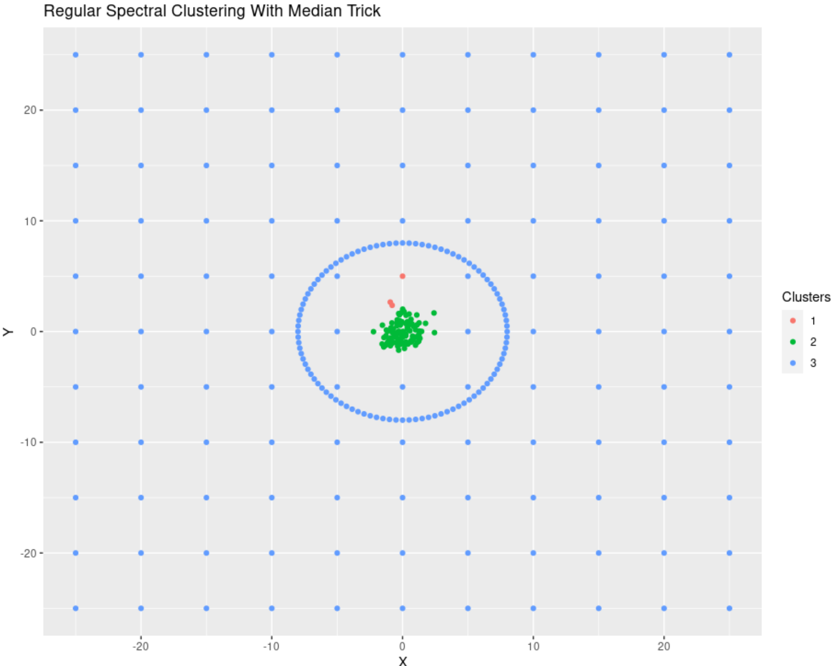
1. Compute the local scale σ_i for each data point x_i.
2. Form the locally scaled affinity matrix $\hat{W} \in \mathbb{R}^{n \times n}$ where $\hat{W}_{ij} = \exp\left(-\frac{d^2(\mathbf{x}_i, \mathbf{x}_j)}{\sigma_i \sigma_j}\right)$ for $i \neq j$ and $\hat{W}_{ii} = 0$.
3. Let G be the diagonal matrix with $G_{ii} = \sum_{j=1}^{n} \hat{W}_{ij}$ (the degree of x_i) and $\tilde{L} := G^{-1/2} \hat{W} G^{-1/2}$ be the symmetric normalized Laplacian.
4. For some large K', form the matrix $Z \in \mathbb{R}^{n \times K'}$ by stacking the eigenvectors corresponding to the K' largest eigenvalues of $\tilde{L}$ and use gradient descent to find the rotation matrix $R \in \mathbb{R}^{K' \times K'}$ that minimizes $J_{K'} = \sum_{i=1}^{n} \sum_{j=1}^{K'} (Z_{ij}^2 / \max_l Z_{il}^2)$.
5. Choose $K_{\text{best}} = \text{argmin}_K J_K$ (or largest such K if several give very similar costs).

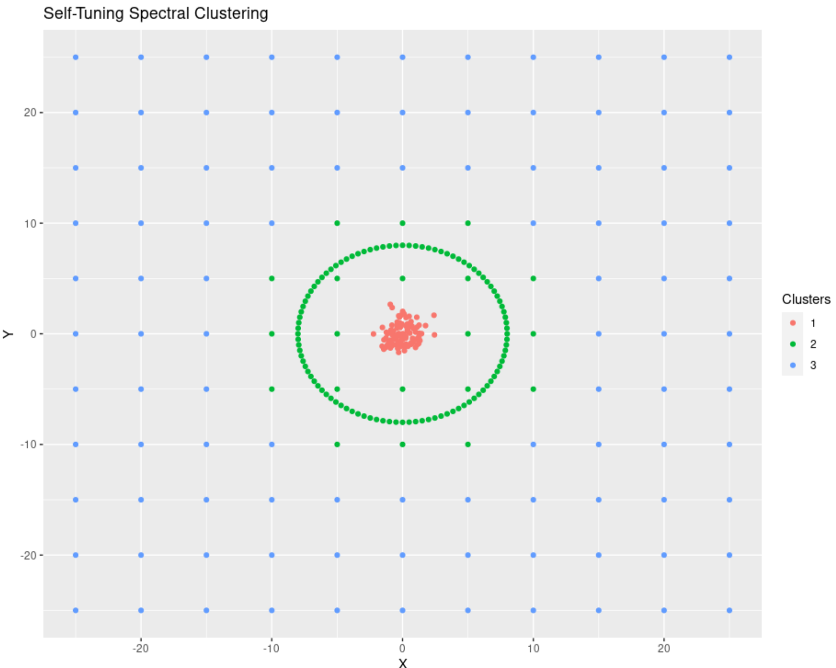
[Zelnik-Manor and Perona, 2004]

Self-Tuning Spectral Clustering Algorithm

1. Compute the local scale σ_i for each data point x_i.
2. Form the locally scaled affinity matrix $\hat{W} \in \mathbb{R}^{n \times n}$ where $\hat{W}_{ij} = \exp\left(-\frac{d^2(\mathbf{x}_i, \mathbf{x}_j)}{\sigma_i \sigma_j}\right)$ for $i \neq j$ and $\hat{W}_{ii} = 0$.
3. Let G be the diagonal matrix with $G_{ii} = \sum_{j=1}^{n} \hat{W}_{ij}$ (the degree of x_i) and $\tilde{L} := G^{-1/2}\, \hat{W}\, G^{-1/2}$ be the symmetric normalized Laplacian.
4. For some large K', form the matrix $Z \in \mathbb{R}^{n \times K'}$ by stacking the eigenvectors corresponding to the K' largest eigenvalues of $\tilde{L}$ and use gradient descent to find the rotation matrix $R \in \mathbb{R}^{K' \times K'}$ that minimizes $J_{K'} = \sum_{i=1}^{n} \sum_{j=1}^{K'} (Z_{ij}^2 / \max_l Z_{il}^2)$.
5. Choose $K_{\text{best}} = \text{argmin}_K\, J_K$ (or largest such K if several give very similar costs).
6. Assign x_i to cluster k if and only if $\max_j Z_{ij}^2 = Z_{ik}^2$. (Or, for very noisy data, use K-means to cluster the rows of Z as in the standard algorithm.)

[Zelnik-Manor and Perona, 2004]


Regular Spectral Clustering With Median Trick
Y
X
20
10
0
-10
-20
-20
-10
0
10
20
Clusters
1
2
3



Self-Tuning Spectral Clustering
20
10
0
-10
-20
Y
-20
-10
0
10
20
X
Clusters
1
2
3

Conclusion

- Self-tuning spectral clustering allows us to determine suitable values of σ and K automatically.
- We do have to choose P and K', but these are much simpler to tune (and we can usually just set K' to be 'big enough' in some sense).
- The local scaling implemented to deal with σ also allows us to perform clustering noisy, multi-scale data.

References

Ng, A. Y., Jordan, M. I., and Weiss, Y. (2001).
On Spectral Clustering: Analysis and an algorithm.

Zelnik-Manor, L. and Perona, P. (2004).
Self-Tuning Spectral Clustering.

Thank you

Any questions?