Asymmetric Spectral clustering

Antoine Lejay

Asymmetric Spectral clustering

Antoine Lejay
Université de Lorraine, CNRS, Inria, IECL, F-54000 Nancy, France
antoine.lejay@univ-lorraine.fr

September 13, 2019 — Version 1

Abstract This report presents the mathematical foundation of the asymmetric spectral clustering approach given in the thesis of S.E. Atev (University of Wisconsin, 2011). This method, implemented in a companion `R` package `AsymmetricClustering`, is based on a Riemannian conjugate gradient algorithm to find a minimization problem based on an optimization problem involving unitary matrix.

Keywords Asymmetric Clustering — Riemannian Gradient Conjugate — Spectral Embedding

---

1 Introduction

A spectral clustering technique consists in assigning one cluster among k for a set of n data with $k \lll n$. More precisely, the clustering performed by combining 1) a dimension reduction based on the main eigenvalues of an affinity matrix A that encodes the similarities between any pairs of data, and 2) seeking the main direction of this reduced model. The main directions are specified as a basis of $\mathbb{R}^k$ (or $\mathbb{C}^k$), and the cluster associated to each data is given by selectionning the closest line among the k orthogonal lines specified by the basis.

Many spectral clustering techniques exists, coming from various fields. The survey [6] presents some of them with their rationale.

The affinity matrix A is usually symmetric as its coefficients are given by the distance between the pairs of data, this is not necessarily the case. In [2], S.E. Atev presents an algorithm to deal with asymmetric matrices A, with application to image analysis. He also gives a simplified `Matlab` code.

This report is the companion of the `R` package `AsymmetricClustering` which is a port of the above code. In particular, we present the mathematical foundation of the algorithm.

2 Notations

The notations given in Table 1 are used in this report.

3 Spectral clustering

The spectral clustering is performed through the following steps:

- We construct from n data $x_1, \dots, x_n$ a $n \times n$ matrix A for which each entry $A_{i,j}$ records a "similarity" between x_i and x_j.
- Through a spectral embedding step, we transform the matrix A into a $k \times n$ matrix Y that keeps the main features of A, where k is the number of clusters, $k \lll n$. This data reduction step uses the spectral information contained in A.

$[\![k, n]\!]$	Interval for integers	$\{k, \dots, n\}$
$\mathfrak{M}_{p\times q}(\mathbb{K})$	*Matrices* with p rows and q columns with coefficients in $\mathbb{K} = \mathbb{R}, \mathbb{C}$	
$\operatorname{Tr} M$	Trace of $M \in \mathfrak{M}_{n\times n}(\mathbb{K})$	$\sum_{i=1}^{n} M_{ii}$
$M_{\bullet i}$	i-th column	$M_{\bullet i} = \begin{bmatrix} M_{1,i} \\ \vdots \\ M_{n,i} \end{bmatrix}$
$M_{i\bullet}$	i-th row	$M_{i\bullet} = \begin{bmatrix} M_{1,i} & \cdots & M_{n,i} \end{bmatrix}$
M^T	*Transpose* of $M \in \mathfrak{M}_{p\times q}(\mathbb{K})$	$M^T = (M_{j,i})_{\substack{i\in[\![1,p]\!] \\ j\in[\![1,q]\!]}}$
M^H	*Hermitian transpose* of $M \in \mathfrak{M}_{p\times n}(\mathbb{C})$	$M^H = \overline{M}^T$
$u \cdot_{\mathbb{R}} v$	*Scalar product* of $u, v \in \mathbb{C}^d$	$\sum_{i=1}^{d} u_i v_i = u^T v$
$u \cdot v$	*Hermitian scalar product* of $u, v \in \mathbb{C}^d$	$u \cdot v = u^T \cdot_{\mathbb{R}} \overline{v}$
$u \otimes v$	*Tensor product* $u \otimes v \in \mathfrak{M}_{k\times n}(\mathbb{C})$	$u \otimes v = uv^T$
$\mathfrak{H}_n$	*Hermitian matrices* of $\mathfrak{M}_{n\times n}(\mathbb{C})$, characterized by $M = M^H$	
$\mathfrak{U}_n$	*Unitary matrices* of $\mathfrak{M}_{n\times n}(\mathbb{C})$, characterized by $U^H U = \mathrm{Id}$	
$\mathfrak{u}_n$	Lie algebra of *unitary matrices* of $\mathfrak{M}_{n\times n}(\mathbb{C})$, characterized by $U^H = -U$.	

Table 1: Notations

- We look for an orthogonal (if A is symmetric) or unitary matrix U whose axes encoding an orthonormal basis $\{u_1, \dots, u_k\}$ on $\mathbb{R}^k$ or $\mathbb{C}^k$. This matrix is selected to minimaize the overall distance between the columns vectors of Y, seen as elements of $\mathbb{R}^k$, and the rays emanating from 0 in one of the direction u_i. If the j-th column vector of Y is closest to the line in the direction u_i than in any other direction u_ℓ, $\ell \ni$, then we assign the data x_j to the i-th cluster.

3.1 Dealing with asymmetric matrices

We consider a $n \times n$ matrix A which is not necessarily symmetric.

Lemma 1. *The transform* $H : \mathfrak{M}_{n\times n}(\mathbb{R}) \to \mathfrak{H}_n$ *defined by*

$$H(A) = \frac{1}{2}(A + A^T) + \frac{1}{2}i(A - A^T)$$

is one-to-one with inverse $H^{-1}(M) = \mathfrak{R}M + \mathfrak{I}M$ *for* $M \in \mathfrak{H}_n$.

As the matrix $H(A)$ is Hermitian, we may consider the eigenvalue problem

$$H(A)x = \lambda x,\ x \in \mathbb{C}^d,\ \lambda \in \mathbb{R}.$$

The eigenvectors are orthogonal for the Hermitian scalar product. Therefore, we summarize the eigenvalue problem as

$$V \Lambda V^H = H(A) \tag{1}$$

where V is a unitary matrix and

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \lambda_d \end{bmatrix}$$

with $\lambda_1 \geq \dots \geq \lambda_n$. is a diagonal matrix containing the eigenvalues. The eigenvector of norm 1 corresponding to λ_i is encoded into $V_{\bullet i}$.

3.2 Spectral embedding

The spectral embedding step consists in transforming the coordinates of the $n \times n$ matrix A to a matrix Y of size $k \times n$, where k is the number of clusters. This is a dimensension reduction.

Following [2], the package `AsymmetricClustering` considers by default the following spectral embedding princple. Other choices are possible [2, Table 3.2, p. 23].

Principle 1 (Spectral embedding step)**.** Using Λ and V as in (1), we define

$$Y = \begin{bmatrix} \lambda_1^{-1/2} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \lambda_k^{-1/2} \end{bmatrix} \begin{bmatrix} V_{\bullet 1} & \dots & V_{\bullet k} \end{bmatrix}^H \in \mathfrak{M}_{n\times k}(\mathbb{C}).$$

As explained in [4], the spectral embedding step stems from the following principle: "Truncation of the eigenbasis amplifies any unevenness in the distribution of points on the d-dimensional hypersphere by causing points of high affinity to move toward each other and other to move apart."

Remark 1. With this matrix Y, we have

$$YV_{\bullet i} = \begin{cases} \lambda_i^{-1/2} & \text{if } 1 \leq i \leq k, \\ 0 & \text{otherwise.} \end{cases}$$

The matrix Y satisfies $YAY^* = \mathrm{Id}$.

3.3 Transformation into an optimization problem

The problem is now rewritten as an optimization problem.

Lemma 2. *Given a point $y \in \mathbb{C}^k$, $n \geq 1$ and a direction $u \in \mathbb{C}^k$, the distance $d(y,u)$ between y and its orthogonal projection on u is given by*

$$d(y,u) = \sqrt{\overline{y}^T(\mathrm{Id} - u \otimes \overline{u})y}. \tag{2}$$

Principle 2 (General principle of spectral clustering)**.** Given a matrix $Y \in \mathfrak{M}_{k\times n}(\mathbb{C})$, we consider finding a unitary matrix $U \in \mathfrak{U}_{k\times k}$ such that $d(Y_{\bullet i}, U_{\sigma(i)})$ is minimal for each $i \in [\![1,n]\!]$, where $\sigma : [\![1,n]\!] \to [\![1,k]\!]$. The cluster of the column i is given by $\sigma(i)$.

To set up Principle 2 in practive, we need to rewrite it by defining a cost function. For that, σ is replaced by $W \in \mathfrak{M}_{k\times n}$ containing exactly one and only one 1 on each row. The *cost function*[1] is

$$J(Y,U,W) = \sum_{i=1}^{n}\sum_{j=1}^{k} W_{j,i}D_{j,i} \text{ with } D_{j,i} = d(Y_{\bullet i}, U_{\bullet j})^2 \tag{3}$$

for $Y \in \mathfrak{M}_{k\times n}(\mathbb{C})$, $U \in \mathfrak{U}_{k\times k}$ and $W \in \mathfrak{M}_{k\times n}(\mathbb{R})$ with the constraint

$$W_{j,i} \in \{0,1\} \text{ and } \sum_{j=1}^{k} W_{j,i} = 1 \text{ for } i = 1,\dots,n. \tag{4}$$

Principle 3. Fix $1 \leq k \leq n$. Given $Y \in \mathfrak{M}_{k,n}(\mathbb{R})$, find $U \in \mathfrak{M}_{k\times k}(\mathbb{C})$ and $W \in \mathfrak{M}_{k\times n}(\mathbb{R})$ satisfying (4) such that

$$\begin{gathered} J(Y,U,W) \text{ is minimal} \\ \text{with } U \in \mathfrak{U}_k \text{ and } W \text{ satisfies (4).} \end{gathered} \tag{5}$$

The constraint (4) is relaxed into

$$0 \leq W_{i,j} \leq 1 \text{ and } \sum_{j=1}^{k} W_{j,i} = 1 \text{ for } i = 1,\dots,n. \tag{6}$$

We coefficients $W_{i,j}$ are then seen as *weights* that quantify the probability to belong to a class k.

Principle 4 (Minimization of the cost function, relaxed version of Principle 3)**.** Fix $1 \leq k \leq n$. Given $Y \in \mathfrak{M}_{k,n}(\mathbb{R})$, find $U \in \mathfrak{M}_{k\times k}(\mathbb{C})$ and $W \in \mathfrak{M}_{k\times n}(\mathbb{R})$ satisfying (4) such that

$$\begin{gathered} J(Y,U,W) \text{ is minimal} \\ \text{with } U \in \mathfrak{U}_k \text{ and } W \text{ satisfies (6).} \end{gathered} \tag{7}$$

The cluster for the column i of Y is given by $\arg\max_{1\leq j\leq k} W_{j,i}$

The minimization problem 7 is solved using an iterative approach in which we alternatively optimize over the weights and over unitary matrices, up to reaching a state in which the update are small. A "temperature" parameter is updated at each global step.

[1] In [2], the matrix W is defined as the transpose of ours.

4 Numerical algorithms

4.1 Algorithm 1: updating the weights

When U is fixed, the weight matrix W is updated through

$$W_{j,i}(\sigma, Y, U) = \frac{\exp(-D_{j,i}/\sigma)}{\sum_{\ell=1}^{k} \exp(-D_{\ell,i}/\sigma)} \text{ for } j \in [\![1, k]\!],\ i \in [\![1, n]\!],$$

with $D_{j,i} = d(Y_{\bullet i}, U_{\bullet j})^2$ (see (3)) for a scale parameter σ. Thus, the column $W_{\bullet,i}$ is the output the softmax function

$$\mathcal{S}\left(\begin{bmatrix} x_1 \\ \vdots \\ x_k \end{bmatrix}, \sigma\right) = \frac{1}{\sum_{i=1}^{k} \exp(x_i/\sigma)} \left(\begin{bmatrix} \exp(x_1/\sigma) \\ \vdots \\ \exp(x_k/\sigma) \end{bmatrix}\right),$$

applied to the vector $D_{\bullet,i}$. The function $\sigma \mapsto J(Y, U, W(\sigma, Y, U))$ is non-decreasing. Reducing σ reduces the cost function. At the end of the m-th step, we update the scale parameter σ as

$$\sigma_{m+1} = \frac{J(Y, U_m, W(\sigma_m, Y, U_m))}{n \cdot m}.$$

To avoid numerical problems with the softmax function $\mathcal{S}$, we actually use the formula

$$\mathcal{S}\left(\begin{bmatrix} x_1 \\ \vdots \\ x_k \end{bmatrix}, \sigma\right) = \frac{1}{\sum_{i=1}^{k} \exp\left(\frac{x_i - m}{\sigma}\right)} \left(\begin{bmatrix} \exp\left(\frac{x_1 - m}{\sigma}\right) \\ \vdots \\ \exp\left(\frac{x_k - m}{\sigma}\right) \end{bmatrix}\right) \text{ with } m = \max_{j=1,\dots,k} x_j.$$

See [3] for numerical considerations on the computation of the softmax function.

4.2 Algorithm 2: minimization over the unitary matrices

During this optimisation step, the matrix W is fixed. The problem is then to optimize over unitary matrices. For this, a conjugate gradient method is used in the framework of a Riemannian algorithm [5] that takes profits from the geometry of the Lie group of unitary matrices. The algorithm implements the method of [1].

At each point U of $\mathfrak{U}_n$, the tangent space $T_U\,\mathfrak{U}_n$ is the set

$$T_U\,\mathfrak{U}_n = \{SU \mid S \in \mathfrak{u}_n\} \text{ with } \mathfrak{u}_n = \{S \in \mathfrak{M}_{n\times n}(\mathbb{C}) \mid S^H = -S\}.$$

The tangent space is also identified with the set

$$T_U\,\mathfrak{U}_n = \{S \in \mathfrak{M}_{n\times n}(\mathbb{C}) \mid U^H S + S^H U = 0\}.$$

On $\mathfrak{M}_{n\times n}(\mathbb{C})$, we define a scalar product as

$$\langle\!\langle M, N \rangle\!\rangle = \frac{1}{2}\Re\,\mathrm{Tr}(MN^H).$$

Lemma 3. *For $M \in \mathfrak{M}_{n\times n}(\mathbb{C})$, write $M_{\mathrm{a}} = \frac{1}{2}(M - M^H)$. The map $p^{\perp} : M \mapsto M_{\mathrm{a}}$ is the orthogonal projection from $\mathfrak{M}_{n\times n}(\mathbb{C})$ to $\mathfrak{u}_n$ for $\langle\!\langle \cdot, \cdot \rangle\!\rangle$.*

Proof. For this scalar product, Hermitian and anti-Hermitian matrices are orthogonal. Any matrix M is decomposed as $M = M_{\mathrm{s}} + M_{\mathrm{a}}$ with $M_{\mathrm{s}} = \frac{1}{2}(M + M^H)$ which is Hermitian and M_{a} which is anti-Hermitian. □

As any element of $T_U\,\mathfrak{U}_n$ is writen SU for some $S \in \mathfrak{u}_n$ and $U^{-1} = U^*$, a scalar product is naturally on $T_U\,\mathfrak{U}_n$ for any $U \in \mathfrak{U}_n$ as

$$\langle V, W\rangle_U = \langle\!\langle VU^{-1}, WU^{-1}\rangle\!\rangle = \frac{1}{2}\Re\,\mathrm{Tr}(VU^HUW^H) = \langle\!\langle V, W\rangle\!\rangle.$$

This way, $\langle\cdot,\cdot\rangle$ gives rise to a Riemannian metric.

Lemma 4. *The orthogonal projection of $M \in \mathfrak{M}_{n\times n}(\mathbb{C})$ onto $T_U\,\mathfrak{U}$ for $U \in \mathfrak{U}_n$ is*

$$p_U^{\perp}(M) = M - UM^HU. \tag{8}$$

Proof. We set $p_U^{\perp} : M \mapsto p^{\perp}(MU^{-1})U$. It is easily checked that $\langle p_U^{\perp}(M), W\rangle_U = 0$ for any $W \in W \in T_U\,\mathfrak{U}_n$. Using Lemma 3, this leads to (8). □

The geodesics emanating from U in the direction $SU \in T_U\,\mathfrak{U}_n$, that is for $S \in \mathfrak{u}_n$, are then

$$\gamma(t) = \exp(tS)U$$

where $S \mapsto \exp(S)$ is the matrix exponential.

Gradients are computed with respect to the complex conjugate derivative operator

$$\frac{\partial}{\partial U^H} = \frac{1}{2}\left(\frac{\partial}{\partial \Re U} + \mathrm{i}\frac{\partial}{\partial \Im U}\right).$$

The Riemannian gradient of $f : \mathfrak{U}_n \to \mathbb{R}$ at U is the orthogonal projection of $\frac{\partial f}{\partial U^H}$ onto $T_U\,\mathfrak{U}_n$, so that with (8),

$$\nabla_{\mathfrak{U}} f(U) = \frac{\partial f}{\partial U^H} - U\left(\frac{\partial f}{\partial U^H}\right)^H U. \tag{9}$$

4.2.1 The conjugate gradient

Notation 1. The gradient of the cost function $J(Y, U, W)$ at point $U \in \mathfrak{U}_n$ is denoted by $\Gamma(U) = \frac{\partial J(Y,U,W)}{\partial U^H}$.

The *conjugate gradient algorithm* consists in finding successive points U_k in $\mathfrak{U}_n$ by

- At U_k, the next point U_{k+1} is computed by following the geodesic through a *search direction* $-H_k \in T_{U_k}\,\mathfrak{U}_n$, as
$$U_{k+1} = \exp(-\mu H_k)U_k \text{ with } \mu = \arg\min J(Y, \exp(-\mu H_k)U_k, W).$$
The scalar μ is choosen according to a line search algorithm presented in § 4.2.3.
- At U_{k+1}, the search direction is updated by combining the direction given by steepest descent gradient
$$G_{k+1} = \Gamma(U_{k+1})U_{k+1}^H - U_{k+1}\Gamma(U_k)^H$$
and H_k. As $G_{k+1} \in T_{U_{k+1}}\,\mathfrak{U}_n$ and $H_k \in U_{T_k}\,\mathfrak{U}_n$, H_k a parallel transport is used, so that H_k is transformed into
$$\widetilde{H}_k = H_k \exp(-\mu H_k)U_k = H_kU_{k+1}.$$

The new search direction is defined so that

$$-H_{k+1} = -G_{k+1} - \theta \widetilde{H}_k \tag{10}$$

where θ is selected so that H_{k+1} and $\widetilde{H}_k$ are Hessian conjugate, that is

$$H_{k+1}^T \operatorname{Hess} J(Y, U_{k+1}, W) \widetilde{H}_k = 0.$$

The value of θ is actually computed using the Polak-Ribière approximation [1, Eq. (10), § 2.4],

$$\theta = \frac{\Re \operatorname{Tr}((G_{k+1} - G_k)^H G_{k+1})}{\Re \operatorname{Tr} G_k^H G_k}. \tag{11}$$

4.2.2 The gradient

Let us compute first the gradient of the distance $d(y,u)^2$ given by (2) for two vectors y and u in $\mathbb{C}^k$. We note first that

$$\frac{\partial u_i}{\partial u_i^H} = 0 \text{ for } i \neq j, \ \frac{\partial \overline{u}_j}{\partial u_i^H} = 1 \text{ for } i \neq j \text{ and } \frac{\partial u_i \overline{u}_i}{\partial u_i^H} = u_i.$$

Therefore,

$$\frac{\partial d(y,u)^2}{\partial u_i^H} = -\sum_{p=1}^{k} \overline{y}_p u_p y_i = -(y^H \cdot u) y_i.$$

Applied to the cost function,

$$\frac{\partial J(Y,U,W)}{\partial U_{r,c}^H} = -\sum_{i=1}^{n} W_{c,i} \big((Y_{\bullet i})^H \cdot U_{\bullet c} \big) Y_{r,i} = -\sum_{i=1}^{n} \sum_{j=1}^{k} W_{c,i} (Y^H U)_{i,c} Y_{r,i}.$$

4.2.3 The line search

The line search algorithm consists in finding the minimal point of the cost function along a geodesic curve. More precisely, of

$$G(\mu) = J(Y, \exp(-\mu H)U, W)$$

for a direction $H \in \mathfrak{u}_n$, as $\mu \mapsto \exp(-\mu H)U$ is a geodesic passing through U in the direction HU.

Here, H is the *search direction*: it is either the steepest descent (computed from the Riemannian Gradient $\nabla_{\mathfrak{U}} J(Y,U,W)$) given by (9)) or the conjugate gradient (see (10) and (11)).

We rewrite

$$G(\mu) = J(Y, (1+Z)U, W) \text{ with } Z = \exp(-\mu H) - 1.$$

The cost function J is quadratic in U, so that there exists J_1 linear and J_2 bilinear on $\mathfrak{M}_{k\times k}(\mathbb{C})$ such that

$$G(\mu) = G(0) + J_1 \cdot (\exp(-\mu H) - 1) + J_2 \cdot (\exp(-\mu H) - 1) \otimes (\exp(-\mu H) - 1). \tag{12}$$

Let us recall a standard result on the spectral decomposition of anti-Hermitian matrices.

Lemma 5. *The spectrum of H is purely imaginary, so that* $\exp(-\mu H) = Q \operatorname{Diag}(e^{-\mu i \omega_1}, \dots, e^{-\mu i \omega_k}) Q^H$ *for $\omega_1 \leq \cdots \leq \omega_k$ and Q a unitary matrix.*

It follows from the above property and (12) that $G(\mu)$ is the linear superposition of function of type $\mu \mapsto \exp(-\mu i\omega_j)$ and $\mu \mapsto \exp(-2\nu i\omega_j)$.

With $\omega = \max_{i=1,\dots,k}|\omega_i|$, we then restrict the search of the minimum of $G(\mu)$ to the interval $[0, 2\pi/q\omega]$ with $q = 2$, as $G(\mu + 2\pi\omega)$ is close to $G(\mu)$.

With the chain rule and the definition of the Riemannian gradient,

$$G'(\mu) = -2\Re \operatorname{Tr}\left(\nabla_{\mathfrak{U}} J(Y, \exp(-\mu H)U, W)U^H \exp(-\mu H)^H H^H\right)$$

since the derivative of $\mu \mapsto \exp(-\mu H)U$ is $H \exp(-\mu H)U \in T_{\exp(-\mu H)U}\,\mathfrak{U}_n$.

As the search direction imposes that $G'(0) < 0$, to find the minimum of $G(\mu)$, we look for the smallest zero-crossing of the derivative.

To avoid the high cost of computing the exponential $\exp(-\mu H)$ too frequently, we use a polynomial approximation of $G'(\mu)$. Thus, $G'(\mu)$ is evaluated at the P spaced point $\mu_k = \frac{k\pi}{P\omega}$, $k = 1, \dots, P$. Hence,

$$\exp(-\mu_k G) = \exp(-\mu_1 G)^k \text{ for } k = 1, \dots, P.$$

With the approximation

$$G'(\mu) \approx G'(0) + \sum_{k=1}^{P} a_k \mu^k,$$

we obtain

$$G'(\mu_i) \approx G'(0) + \sum_{k=1}^{P} a_k \mu_i^k \text{ for } i = 1, \dots, P.$$

The a_i are then found by solving the linear system

$$\begin{bmatrix} G'(\mu_1) - G'(0) \\ \vdots \\ G'(\mu_P) - G'(0) \end{bmatrix} = \begin{bmatrix} \mu_1 & \cdots & \mu_1^P \\ \vdots & \vdots & \vdots \\ \mu_k & \cdots & \mu_k^P \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_P \end{bmatrix}.$$

The line search algorithm returns the smallest positive root of $a_0 + a_1\mu + \cdots + a_P\mu^P = 0$ if it exist. Otherwise, it returns 0, and a new direction is computed.

References

[1] T. Abrudan, J. Eriksson, and V. Koivunen. "Conjugate gradient algorithm for optimization under unitary matrix constraint". In: *Signal Processing* 89.9 (Sept. 2009), pp. 1704–1714. DOI: 10.1016/j.sigpro.2009.03.015.

[2] S. E. Atev. "Using Asymmetry in the Spectral Clustering of Trajectories". University of Minnesota, 2011.

[3] P. Blanchard, D. J. Higham, and N. J. Higham. *Accurate Computation of the Log-Sum-Exp and Softmax Functions*. 2019. arXiv: 1909.03469.

[4] M. Brand and K. Huang. "A unifying theorem for spectral embedding and clustering". In: *AISTATS*. 2003.

[5] A. Edelman, T. A. Arias, and S. T. Smith. "The Geometry of Algorithms with Orthogonality Constraints". In: *SIAM Journal on Matrix Analysis and Applications* 20.2 (Jan. 1998), pp. 303–353. DOI: 10.1137/s0895479895290954.

[6] U. von Luxburg. "A tutorial on spectral clustering". In: *Statistics and Computing* 17.4 (Dec. 2007), pp. 395–416. DOI: 10.1007/s11222-007-9033-z.