

Laplacian Matrix for Dimensionality Reduction and Clustering

— Lecture Notes —

Laurenz Wiskott & Fabian Schönfeld
Institut für Neuroinformatik
Ruhr-Universität Bochum, Germany, EU

18 September 2019

Contents

1	Introduction	3
2	Intuition	3
2.1	Heat diffusion analogy of Laplacian eigenmaps	3
2.2	Heat diffusion analogy of spectral clustering	4
2.3	Heat diffusion equation for connected heat reservoirs	5
2.4	Laplacian matrix	5
2.5	Solution of the heat diffusion equation	6
3	Formalism	7
3.1	Simple graphs	7
3.2	Matrix representation	8
3.3	Optimization problem	9
3.4	Associated eigenvalue problem	9
3.5	The role of the weighted normalization constraint	10
3.6	Symmetric normalized Laplacian matrix	11
3.7	Random walk normalized Laplacian matrix +	12
3.8	Summary of mathematical properties	13

© 2017–2019 Laurenz Wiskott (ORCID <http://orcid.org/0000-0001-6237-740X>, homepage <https://www.ini.rub.de/PEOPLE/wiskott/>). This work (except for all figures from other sources, if present) is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License, see <http://creativecommons.org/licenses/by-sa/4.0/>. If figures are not included for copyright reasons, they are uni colored, but the word 'Figure', 'Image', or the like in the reference is often linked to a freely available copy.

Core text and formulas are set in dark red, one can repeat the lecture notes quickly by just reading these; ♦ marks important formulas or items worth remembering and learning for an exam; ◇ marks less important formulas or items that I would usually also present in a lecture; + marks sections that I would usually skip in a lecture.

More teaching material is available at <https://www.ini.rub.de/PEOPLE/wiskott/Teaching/Material/>.

4 Algorithms	14
4.1 Similarity graphs	14
4.2 Laplacian eigenmaps (LEM)	16
4.2.1 Motivation	16
4.2.2 Objective	16
4.2.3 Algorithm	17
4.2.4 Sample applications	17
4.3 Locality preserving projections (LPP)	19
4.3.1 Linear LPP	19
4.3.2 Sample application	20
4.3.3 Nonlinear LPP	21
4.4 Spectral clustering	21
4.4.1 Objective	21
4.4.2 Algorithm	21
4.4.3 Sample application	22

Requirements: I assume the student can already ...

- ... apply basic concepts from linear algebra, such as *vector*, *matrix*, *matrix product*, *inverse matrix*.
- ... solve an *ordinary eigenvalue equation* in linear algebra and explain intuitively what *eigenvalues* and *eigenvectors* are.
- ... relate the *eigenvalues* and *eigenvectors* of a symmetric matrix to the solutions of the minimization/maximization problem of the corresponding *quadratic form*.
- ... interpret a *system of linear differential equations with constant coefficients*.

Learning objectives: The learning objective of this unit is that the student can ...

- ... define basic notions of graph theory, namely *graph*, *node*, *edge*, and *simple graph* (Sec. 3.1).
- ... explain matrix representations of graphs, namely *adjacency matrix*, *degree matrix*, and *Laplacian matrix* (Sec. 3.2).
- ... reproduce and interpret the generalized eigenvalue equation (45) of the Laplacian matrix and weighted degree matrix and describe how it relates to the optimization problem of Laplacian eigenmaps and spectral clustering (Eqs. 46–48).
- ... summarize and motivate mathematical properties $\langle 6,7,9,11 \rangle$ (Sec. 3.8) of the eigenvalues and eigenvectors of the generalized eigenvalue equation (45).
- ... discuss the role of the normalization constraint (47) vs. (42) (Sec. 3.5).
- ... explain how a similarity graph can be generated from a set of data points (Sec. 4.1).
- ... explain how the Laplacian eigenmaps (LEM) algorithm (Sec. 4.2) and spectral clustering (Sec. 4.4) work.
- ... name a limitation of LEM and sketch how locality preserving projections (LPP) overcome it (Sec. 4.3).

1 Introduction

Many problems in machine learning can be expressed by means of a graph with nodes representing training samples and edges representing the relationship between samples in terms of similarity, temporal proximity, or label information. Graphs can in turn be represented by matrices. A special example is the Laplacian matrix, which allows us to assign each node a value that varies only little between strongly connected nodes and more between distant nodes. Such an assignment can be used to extract a useful feature representation, find a good embedding* of data in a low dimensional space, or perform clustering on the original samples. In the following we first introduce the Laplacian matrix and then present a small number of algorithms designed around it.

2 Intuition

This section is meant to give an intuitive introduction into the Laplacian matrix, Laplacian eigenmaps, and spectral clustering. It is not necessary to understand the remainder of the lecture notes but hopefully makes it easier. If you are short on time and rich in math and machine learning background, you might prefer to skip it.

The Laplacian matrix can be used to model heat diffusion in a graph. Its theory can thus be understood intuitively with the help of the heat diffusion analogy.

2.1 Heat diffusion analogy of Laplacian eigenmaps

First consider a very simple heat diffusion analogy for nonlinear dimensionality reduction from 2D to 1D with the Laplacian eigenmap algorithm. Figure 1 (left) shows seven points in 2D, labeled A through G. Their position might not be very meaningful but we assume that we have some similarity function that induces relationships between these points. This results in a simple undirected graph with seven nodes and six edges in this example. We see already that the graph is a simple linear graph, a chain, but in high dimensions with many more nodes and a slightly more complicated structure, this might not be so obvious anymore.

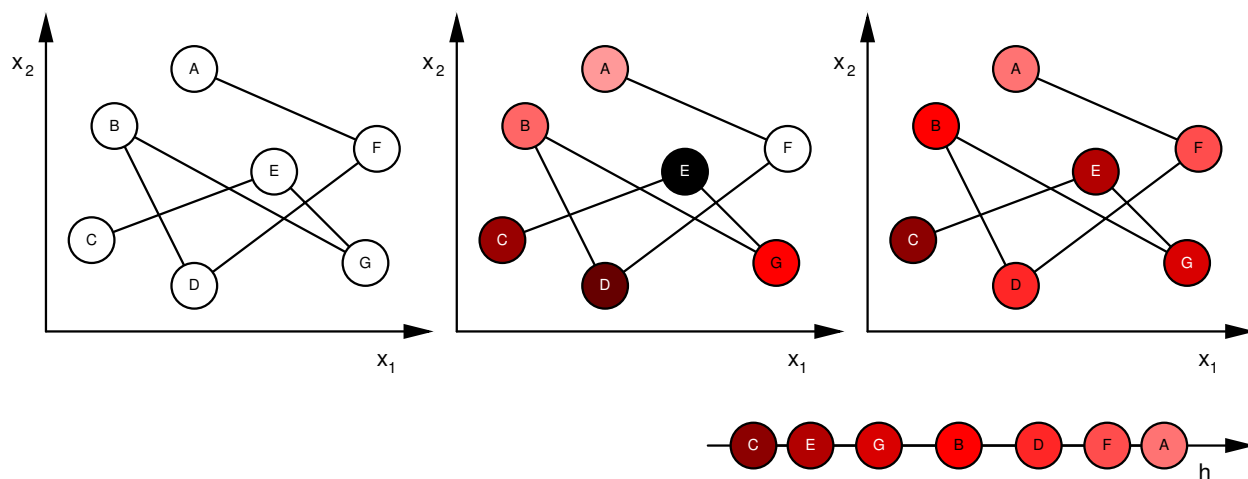


Figure 1: Heat diffusion analogy of the Laplacian eigenmaps algorithm.

The heat diffusion analogy now says that nodes are considered heat reservoirs and heat can diffuse from one node to neighboring nodes via the edges, but no heat gets lost or added. So, let

*A remark on terminology: We use *assign/assignment* for giving data samples an associated value. These values implicitly define a *mapping* from (possibly high-dimensional or non-vectorial) data samples to points in a low-dimensional space, the *mapped space*. In LPP the mapping is defined more explicitly by a linear function. The collection of points in mapped space form an *embedding*. Thus, all these terms refer to the same process.

us randomly initialize the nodes with arbitrary temperatures, Figure 1 (middle). What happens if we wait? Well, it is obvious that heat diffuses from warmer to colder nodes until temperature has balanced out completely. It is also obvious that local temperature differences balance out quickly, while global temperature differences between distant nodes (distant in terms of the graph connectivity) take more time to balance out. So **if one measures the temperatures quite late in the process, one finds a distribution like the one shown in Figure 1 (right)**. One end of the chain is slightly warmer than the other end, and from one end to the other there is a monotonic decrease of temperature. This is interesting, because **if one now plots the seven points again, but now in a 1D space according to their temperature, one gets the plot in Figure 1 (bottom right)**. The points are nicely ordered by their position in the linear graph. This is much better for visualization and interpretation and possibly further processing of the points, since the position in space now reflects similarity relations well. (The details of the spacing reveal a flattening of the temperature profile towards the ends, an effect that takes more effort to understand intuitively and is beyond the scope of this introduction.)

This is essentially how the Laplacian eigenmaps algorithm works, except that one does not really use heat diffusion but **finds the resulting heat distribution analytically** in a more efficient and robust way. It is also possible to map the points into a 2D or even higher-dimensional space by taking more than one heat diffusion mode into account.

2.2 Heat diffusion analogy of spectral clustering

For a heat diffusion analogy of spectral clustering consider a different connectivity of the graph, like the one shown in Figure 2 (left). The difference to the example above is that now **the graph has two disconnected subgraphs**. No heat can diffuse from one subgraph to the other. **If one waits long enough, the temperature within each subgraph has completely balanced out, but the two subgraphs have different temperature**, because there is no edge between them, Figure 2 (right). **If one now plots the seven points in a 1D space according to their temperature, Figure 2 (bottom right)**, all points of one subgraph cluster at one value and the points of the other subgraph cluster at another value. Thus, in this space **separating the two subgraphs is trivial**.

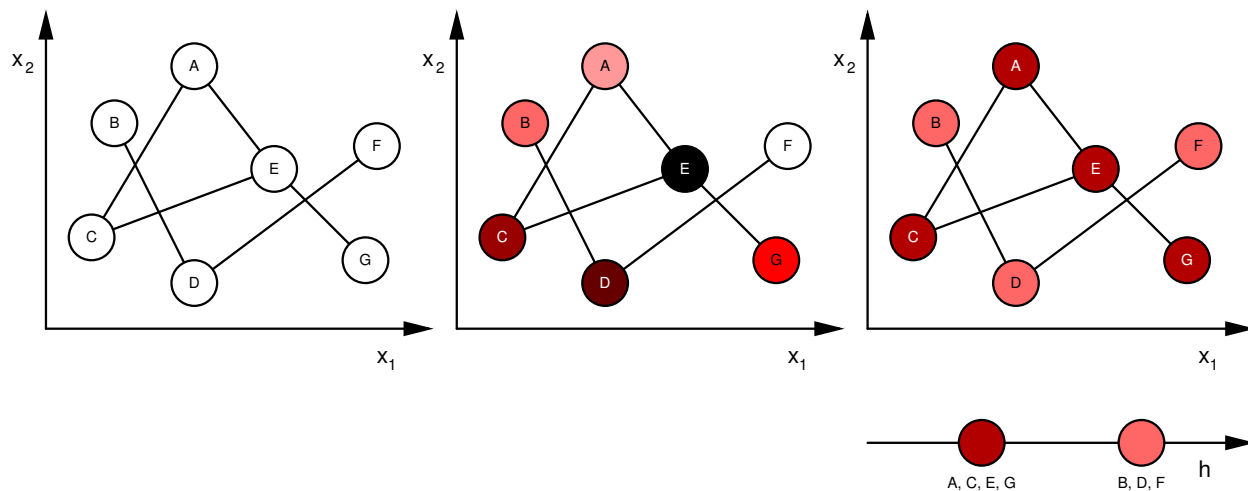


Figure 2: Heat diffusion analogy of spectral clustering.

This is essentially how spectral clustering works. In real data the clusters, i.e. subgraphs, might not be completely disconnected, but with some tricks one can also deal with that.

The graphs in Figures 1 and 2 are drawn in a way that **the position of the nodes actually has no meaning at all**. This is to emphasize that the edges are the only thing that matters for the result of Laplacian eigenmaps and spectral clustering. **In real world examples**, however, spatial proximity often plays an important role and **edges are preferably inserted between neighboring data points**.

2.3 Heat diffusion equation for connected heat reservoirs

How can we model heat diffusion mathematically, and how can we figure out the relevant temperature distributions analytically? Heat diffusion is a continuous process, so we need a differential equation (DE) for it. Since we consider heat diffusion between a discrete set of heat reservoirs rather than on a continuous medium, the DE is a system of ordinary DEs and not a partial DE. It is linear, e.g. if you have twice as much heat, diffusion will be twice as strong. And it is homogeneous, because if there is no heat, then there is no diffusion. Thus we **consider the following system of ordinary linear DEs**

$$\diamond \quad \dot{\mathbf{h}}(t) = -\mathbf{L}\mathbf{h}(t) \quad (1)$$

$$\diamond \iff \dot{h}_1(t) = -L_{11}h_1(t) - L_{12}h_2(t) - L_{13}h_3(t) \quad (2)$$

$$\wedge \dot{h}_2(t) = -L_{21}h_1(t) - L_{22}h_2(t) - L_{23}h_3(t) \quad (3)$$

$$\wedge \dot{h}_3(t) = -L_{31}h_1(t) - L_{32}h_2(t) - L_{33}h_3(t) \quad (4)$$

spelled out for three heat reservoirs, **where $\mathbf{h}(t)$ is a nonnegative vector representing the temperatures of the nodes** as a function of time. **\mathbf{L} is a matrix representing the heat diffusion** between the nodes, and it will be explained in a moment.

Readers not so familiar with differential equations might find it easier to consider the temporally discretized version of it,

$$\frac{\mathbf{h}(t + \Delta t) - \mathbf{h}(t)}{\Delta t} = \dot{\mathbf{h}}(t) \quad (\text{for } \Delta t \rightarrow 0) \quad (5)$$

$$\stackrel{(1)}{=} -\mathbf{L}\mathbf{h}(t) \quad (6)$$

$$\iff \mathbf{h}(t + \Delta t) = \mathbf{h}(t) - \Delta t \mathbf{L}\mathbf{h}(t) \quad (7)$$

$$= (\mathbf{I} - \Delta t \mathbf{L})\mathbf{h}(t) \quad (8)$$

$$\iff h_1(t + \Delta t) = h_1(t) - \Delta t(L_{11}h_1(t) + L_{12}h_2(t) + L_{13}h_3(t)) \quad (9)$$

$$\wedge h_2(t + \Delta t) = h_2(t) - \Delta t(L_{21}h_1(t) + L_{22}h_2(t) + L_{23}h_3(t)) \quad (10)$$

$$\wedge h_3(t + \Delta t) = h_3(t) - \Delta t(L_{31}h_1(t) + L_{32}h_2(t) + L_{33}h_3(t)) \quad (11)$$

which is an approximation of the differential equation $\dot{\mathbf{h}}(t) = -\mathbf{L}\mathbf{h}(t)$, which is exact for $\Delta t \rightarrow 0$.

2.4 Laplacian matrix

In either case, it is clear that \mathbf{L} is responsible for any change of \mathbf{h} and that **the physics of the heat diffusion process imposes constraints on \mathbf{L}** . If $\mathbf{L} = \mathbf{0}$ then $\mathbf{h}(t)$ is constant, which would correspond to three disconnected nodes (= heat reservoirs) that do not exchange any heat. A negative L_{ij} indicates that h_i increases proportional to h_j with factor $-L_{ij}$. A positive L_{ij} indicates that h_i decreases proportional to h_j with factor $-L_{ij}$.

We want that no heat gets lost or added to the system, thus $\sum_i L_{ij} = 0$ must be fulfilled, as one can easily verify by setting $\dot{h}_1(t) + \dot{h}_2(t) + \dot{h}_3(t) = 0$ or $h_1(t + \Delta t) + h_2(t + \Delta t) + h_3(t + \Delta t) = \text{const}$ for any values of $h_1(t), h_2(t)$, and $h_3(t)$. Since the heat one node gains must come from some other nodes, one can say that $-L_{ij}h_j$ (with negative L_{ij}) indicates the amount of heat node i gains from node j for $i \neq j$. The term $-L_{jj}h_j$ (with positive L_{jj}) indicates how much heat node j loses to the other nodes.

If we consider the situation that all three nodes are connected and one node, say Node 1, is hot and the other two nodes are absolutely freezing, i.e. $h_2 = h_3 = 0$ (Kelvin not Celsius) then initially only L_{11}, L_{21} , and L_{31} matter. It is intuitively clear that in this situation heat diffuses from Node 1 to Nodes 2 and 3, i.e. h_1 decreases and h_2 as well as h_3 increase proportionally to h_1 . This implies $0 < L_{11}$, indicating that Node 1 loses heat, and $L_{21}, L_{31} < 0$, indicating that Nodes 2 and 3 gain heat from Node 1. If a connection would be absent, e.g. between Nodes 2 and 1, then no heat diffuses between these two nodes and the corresponding entry is zero, $L_{21} = 0$. If a node, let say Node 1, is not connected to any other node, then it cannot gain or lose heat at all, resulting in $L_{11} = 0$. Thus, by symmetry arguments we have $0 \leq L_{ii}$ and $L_{ij} \leq 0 \forall j \neq i$.

Finally, it is clear that if two different nodes i and j have same temperature, $h_i = h_j$, then the heat $-L_{ij}h_j$ diffusing from node j to node i equals the heat $-L_{ji}h_i$ diffusing from node i to node j , because otherwise one

node would spontaneously become warmer and the other cooler, which would allow us to build a perpetual mobile. This implies $L_{ij} = L_{ji}$. Please notice here that if two connected nodes have same temperature, it does not mean that no heat diffuses from one to the other, it only means that the heat flows cancel out each other.

If we summarize the insights above **we find that**

$$L_{ij} = L_{ji} \quad (\mathbf{L} \text{ is symmetric}) \quad (12)$$

$$\sum_i L_{ij} \stackrel{(12)}{=} \sum_j L_{ij} = 0 \quad (\text{rows and columns add up to zero}) \quad (13)$$

$$L_{ii} \geq 0 \quad (\text{diagonal elements are non-negative}) \quad (14)$$

$$L_{ij} \leq 0 \quad \forall j \neq i \quad (\text{off-diagonal elements are non-positive}) \quad (15)$$

An example of a matrix with all these properties is

$$\mathbf{L} = \begin{pmatrix} 0.2 & -0.2 & 0 \\ -0.2 & 1.0 & -0.8 \\ 0 & -0.8 & 0.8 \end{pmatrix} \quad (16)$$

The corresponding graph is shown in Figure 3.

2.5 Solution of the heat diffusion equation

Assume the eigenvectors \mathbf{u}_α and eigenvalues γ_α of the Laplacian matrix are known with

$$\mathbf{L}\mathbf{u}_\alpha = \gamma_\alpha\mathbf{u}_\alpha \quad (17)$$

and ordered such that $\gamma_1 \leq \gamma_2 \leq \dots \leq \gamma_I$. It turns out that **all eigenvalues are non-negative** and from (13) follows directly that **one can chose $\mathbf{u}_1 = (1, 1, \dots, 1)^T$** (usually normalized to norm one by convention) **with $\gamma_1 = 0$** as the first eigenvector and -value.

For the discretized version of the differential equation it is interesting to see that

$$(8) = \underbrace{(\mathbf{I} - \Delta t\mathbf{L})}_{=: \mathbf{P}} \mathbf{u}_\alpha = \mathbf{I}\mathbf{u}_\alpha - \Delta t\mathbf{L}\mathbf{u}_\alpha \quad (18)$$

$$\stackrel{(17)}{=} \mathbf{u}_\alpha - \Delta t\gamma_\alpha\mathbf{u}_\alpha \quad (19)$$

$$= \underbrace{(1 - \Delta t\gamma_\alpha)}_{=: \xi_\alpha} \mathbf{u}_\alpha \quad (20)$$

Thus the \mathbf{u}_α are also eigenvectors of \mathbf{P} but with eigenvalues $\xi_\alpha = (1 - \Delta t\gamma_\alpha)$ with $1 = \xi_1 \geq \xi_2 \geq \dots \geq \xi_I > 0$ for small enough Δt .

Because the Laplacian matrix is symmetric and real, the set of eigenvectors is complete, and any initial temperature vector $\mathbf{h}(t=0)$ can be written as a linear combination of the eigenvectors

$$\mathbf{h}(t=0) = \sum_\alpha \omega_\alpha \mathbf{u}_\alpha \quad (21)$$

with some appropriate prefactors ω_α .

From the theory of systems of homogeneous linear differential equations we know that **the general solution of (1)** for this $\mathbf{h}(t=0)$ is

$$\diamond \quad \mathbf{h}(t) = \sum_\alpha \omega_\alpha \exp(-\gamma_\alpha t) \mathbf{u}_\alpha \quad (22)$$

For those who prefer the discretized version of the differential equation one can show that

$$\mathbf{h}(t = N\Delta t) \stackrel{(8,18)}{=} \mathbf{P}^N \mathbf{h}(0) \tag{23}$$

$$\stackrel{(21)}{=} \mathbf{P}^N \sum_{\alpha} \omega_{\alpha} \mathbf{u}_{\alpha} \tag{24}$$

$$= \sum_{\alpha} \omega_{\alpha} \mathbf{P}^N \mathbf{u}_{\alpha} \tag{25}$$

$$\stackrel{(20)}{=} \sum_{\alpha} \omega_{\alpha} \xi_{\alpha}^N \mathbf{u}_{\alpha} \tag{26}$$

In either case, if one waits long enough, only the first eigenvectors with eigenvalue $\gamma_{\alpha} = 0$ respectively $\xi_{\alpha} = 1$ will still contribute to $\mathbf{h}(t)$, and one can show that if the graph is connected, only the contribution of \mathbf{u}_1 **survives indefinitely long**, because $\exp(-\gamma_1 t) = \exp(-0t) = 1$ and $\xi_1^N = 1^N = 1$ for any t . **The last eigenvector fading away is \mathbf{u}_2 , and that is exactly the vector we are interested in for the Laplacian eigenmaps algorithm, see Figure 1 (right).**

If the graph is disconnected then it is intuitively clear that each subgraph balances out its heat over time, but there is no heat exchange between subgraphs. **The corresponding Laplacian matrix becomes a block matrix** with as many blocks on the diagonal as there are subgraphs. In the example above in Figure 2, there are two subgraphs, and because of the block structure of the Laplacian matrix and the fact that rows add up to zero, one can verify that **the second eigenvector $\mathbf{u}_2 = (1/4, -1/3, 1/4, -1/3, 1/4, -1/3, 1/4)^T$** (usually normalized to norm one by convention) **is constant within each subgraph and has eigenvalue $\gamma_2 = 0$. This again** reflects the temperature distribution that remains if one waits for a long time, and that **is exactly the vector we are interested in** the spectral clustering algorithm, see Figure 2 (right).

In summary, the second eigenvector of the Laplacian matrix provides a nice 1D arrangement of the nodes of a similarity graph. In practice one often also uses the third and possibly the fourth eigenvector to get visualizations in 2D or 3D, but that is not so easy to understand with this intuitive explanation.

3 Formalism

After the intuitive explanation we **now consider Laplacian eigenmaps and spectral clustering** more directly and **more formally**. For both algorithms data must first be represented as a graph. Nodes represent data samples and edges represent similarities between data samples. The samples could be anything, e.g. words, persons, or melodies, they need not be vectors in a vector space. We just need a non-negative function that measures similarity between two data samples. And this function does not even need to be consistent with a metric. We first introduce some notions from graph theory and then consider the optimization problem.

3.1 Simple graphs

A graph $G = (\mathbb{V}, \mathbb{E})$ is a set of nodes (or vertices or points) $\mathbb{V} = \{v_1, \dots, v_I\}$ **and a set of edges $\mathbb{E} = \{e_1, \dots, e_L\}$.** **An edge e_l connects two nodes v_i and v_j** and is therefore defined by a pair of nodes. Edges may be *directed*, going from node v_i to node v_j , indicated by $e_l = (v_i, v_j)$. Edges may also be *undirected*, in which case the order of the vertices does not matter and we can write $e_l := \{v_i, v_j\}$, where the curly brackets imply that the order does not matter. **Simple graphs are undirected graphs without loops**, which are edges that connect a node with itself, **and no parallel edges**, which are edges that connect the same pair of nodes. **Here we consider mainly simple graphs.**

Further reading: (Wikipedia, 2017a).

3.2 Matrix representation

Graphs can be conveniently represented by real matrices. **The adjacency matrix $A = (A_{ij})$ of an undirected graph is $I \times I$ and defined as**

$$\blacklozenge \quad A_{ij} := \begin{cases} 1 & \text{if } \{v_i, v_j\} \in \mathbb{E} \\ 0 & \text{otherwise} \end{cases} \quad (27)$$

i.e. it has a one in entry A_{ij} if and only if nodes v_i and v_j are connected with each other. Matrix **A is naturally symmetric**, since the edges are not directed.

The degree matrix $D = (D_{ij})$ of an undirected graph is a diagonal matrix, where the diagonal entries D_{ii} indicate the number of edges connected to node v_i .

In context of the Laplacian matrix, **we generalize these definitions to weighted graphs**, where the edges are labeled with a real (positive) number indicating their weight W_{ij} . If one simply replaces the 1 values in (27) by these weights, then **A becomes the (edge) weight matrix W , and the weighted degree matrix $D = (D_{ij})$ gets the sum over all weights of the edges converging on a node in their diagonal entries.**

$$\blacklozenge \quad D_{ii} := \sum_j W_{ij} = \sum_j W_{ji} \quad (28)$$

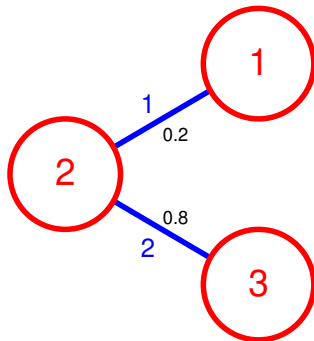


Figure 3: Example of a simple, weighted, undirected graph. Edges are numbered in blue, their weights are shown in black. Weights do not need to add up to one, like here for Node 2.

Figure 3 shows a simple weighted graph. The weighted adjacency matrix, or weight matrix, of the undirected graph is

$$\mathbf{W} = \begin{pmatrix} & v_1 & v_2 & v_3 \\ v_1 & 0 & 0.2 & 0 \\ v_2 & 0.2 & 0 & 0.8 \\ v_3 & 0 & 0.8 & 0 \end{pmatrix} \quad (29)$$

The weighted degree matrix of the undirected graph is

$$\mathbf{D} = \begin{pmatrix} & v_1 & v_2 & v_3 \\ v_1 & 0.2 & 0 & 0 \\ v_2 & 0 & 1.0 & 0 \\ v_3 & 0 & 0 & 0.8 \end{pmatrix} \quad (30)$$

The Laplacian matrix L is defined as the difference between weighted degree matrix D and weight matrix W

$$\blacklozenge \quad \mathbf{L} = \mathbf{D} - \mathbf{W}. \quad (31)$$

It is easy to verify that it has all the properties (12–15) derived in Section 2.4 from the heat diffusion analogy. The Laplacian matrix for the example above is

$$\mathbf{L} = \begin{pmatrix} & 0.2 & -0.2 & 0 \\ -0.2 & & 1.0 & -0.8 \\ 0 & -0.8 & & 0.8 \end{pmatrix} \quad (32)$$

3.3 Optimization problem

The objective of Laplacian eigenmaps as well as spectral clustering is to assign similar values to similar nodes, i.e. strongly connected nodes, and dissimilar values to nodes that are not similar. This is a non-trivial operation, since similarity is a property of a pair of nodes, or an edge, while value is a property of a single node. It is not guaranteed that there is a good solution at all. Consider, for instance, three nodes A , B , and C . If A and B are very similar as well as B and C , but A and C are very dissimilar, then there are no values that could reflect that. However, reasonable similarity measures usually do not lead to such conflicts, definitely not those inducing a proper metric. In any case, the objective is to

$$\blacklozenge \quad \text{minimize} \quad \frac{1}{2} \sum_{ij} (u_i - u_j)^2 W_{ij} \quad (33)$$

$$\diamond \quad \text{subject to} \quad \mathbf{1}^T \mathbf{u} = 0 \quad (\text{zero mean}) \quad (34)$$

$$\diamond \quad \text{and} \quad \mathbf{u}^T \mathbf{u} = 1 \quad (\text{unit variance}) \quad (35)$$

$$\blacklozenge \quad \text{or subject to} \quad \mathbf{1}^T \mathbf{D} \mathbf{u} = 0 \quad (\text{weighted zero mean}) \quad (36)$$

$$\blacklozenge \quad \text{and} \quad \mathbf{u}^T \mathbf{D} \mathbf{u} = 1 \quad (\text{weighted unit variance}) \quad (37)$$

with $\mathbf{u} = (u_1, u_2, \dots, u_I)^T$ and $\mathbf{1} = (1, 1, 1, \dots, 1)^T$ indicating the one-vector. Objective (33) favors solutions where strongly connected nodes with a large edge weight W_{ij} have similar values u_i and u_j . Constraints (34) and (35) in conjunction avoid the trivial constant solution, which implicitly guarantees that nodes that are not similar get assigned dissimilar values. Constraints (36) and (37) have the same function but imply some normalization, see Section 3.5.

If we need more than one solution in order to map the nodes into a higher dimensional space, we add a subscript index to \mathbf{u} and solve the same optimization problem multiple times subject to the additional constraint

$$\diamond \quad \mathbf{u}_\beta^T \mathbf{u}_\alpha = 0 \quad \forall \beta < \alpha \quad (\text{decorrelation to previous solutions}) \quad (38)$$

$$\blacklozenge \quad \text{or} \quad \mathbf{u}_\beta^T \mathbf{D} \mathbf{u}_\alpha = 0 \quad \forall \beta < \alpha \quad (\text{decorrelation to previous solutions}) \quad (39)$$

for the second and later solutions \mathbf{u}_α to make them different (orthogonal) to the previous solutions \mathbf{u}_β .

3.4 Associated eigenvalue problem

It is known that **the normalized eigenvectors \mathbf{u}_α of the ordinary eigenvalue equation**

$$\diamond \quad \mathbf{L} \mathbf{u}_\alpha = \gamma_\alpha \mathbf{u}_\alpha \quad (40)$$

ordered by increasing eigenvalues γ_α **solve the optimization problem**

$$\diamond \quad \text{minimize} \quad \mathbf{u}_\alpha^T \mathbf{L} \mathbf{u}_\alpha = \frac{1}{2} \sum_{ij} (u_{\alpha,i} - u_{\alpha,j})^2 W_{ij} \quad (41)$$

$$\diamond \quad \text{subject to} \quad \mathbf{u}_\alpha^T \mathbf{u}_\alpha = 1 \quad (\text{unit norm}) \quad (42)$$

$$\diamond \quad \text{and} \quad \mathbf{u}_\beta^T \mathbf{u}_\alpha = 0 \quad \forall \beta < \alpha \quad (\text{order and orthogonality}) \quad (43)$$

where constraint (43) induces an order such that \mathbf{u}_1 is the optimal solution without any orthogonality constraint (only the unit norm constraint), \mathbf{u}_2 is the optimal solution with the additional constraint of being orthogonal to \mathbf{u}_1 , \mathbf{u}_3 is the optimal solution with the additional constraint of being orthogonal to \mathbf{u}_1 and \mathbf{u}_2 , etc. Constraints (42, 43) can be combined to $\mathbf{u}_\beta^T \mathbf{u}_\alpha = \delta_{\beta\alpha} \forall \beta \leq \alpha$. Identity (41) is left to the reader as an exercise. If one orders the eigenvalues by ascending rather than descending value, the corresponding eigenvectors solve the maximization rather than minimization problem. The rest should be known, for instance from principal component analysis.

The zero mean constraint (34) is implicit here. Since the first solution \mathbf{u}_1 is a scaled version of $\mathbf{1}$, Constraint (43) with $\beta = 1$ is equivalent to (34). The solutions of interest thus start with index 2 rather than 1.

Since

$$\mathbf{u}_\alpha^T \mathbf{L} \mathbf{u}_\alpha \stackrel{(40)}{=} \mathbf{u}_\alpha^T \gamma_\alpha \mathbf{u}_\alpha = \gamma_\alpha \mathbf{u}_\alpha^T \mathbf{u}_\alpha \stackrel{(42)}{=} \gamma_\alpha \quad (44)$$

the eigenvalues are the optimal values of the objective function.

In the algorithms below the constraint is usually $\mathbf{w}^T \mathbf{D} \mathbf{w} = 1$ rather than $\mathbf{u}^T \mathbf{u} = 1$ (we switch here from \mathbf{u} to \mathbf{w} to indicate solutions with this weighted normalization). Thus we note that **the appropriately normalized eigenvectors \mathbf{w}_α of the generalized eigenvalue equation**

$$\blacklozenge \quad \mathbf{L} \mathbf{w}_\alpha = \lambda_\alpha \mathbf{D} \mathbf{w}_\alpha \quad (45)$$

ordered by increasing eigenvalues λ_α **solve the optimization problem**

$$\blacklozenge \quad \text{minimize} \quad \mathbf{w}_\alpha^T \mathbf{L} \mathbf{w}_\alpha = \frac{1}{2} \sum_{ij} (w_{\alpha,i} - w_{\alpha,j})^2 W_{ij} \quad (46)$$

$$\blacklozenge \quad \text{subject to} \quad \mathbf{w}_\alpha^T \mathbf{D} \mathbf{w}_\alpha = 1 \quad (\text{weighted unit norm}) \quad (47)$$

$$\blacklozenge \quad \text{and} \quad \mathbf{w}_\beta^T \mathbf{D} \mathbf{w}_\alpha = 0 \quad \forall \beta < \alpha \quad (\text{order and weighted orthogonality}) \quad (48)$$

The derivation (44) does not hold here, since the eigenvectors must have weighted unit norm, not standard unit norm. But still we find analogously

$$\mathbf{w}_\alpha^T \mathbf{L} \mathbf{w}_\alpha \stackrel{(45)}{=} \mathbf{w}_\alpha^T \lambda_\alpha \mathbf{D} \mathbf{w}_\alpha = \lambda_\alpha \mathbf{w}_\alpha^T \mathbf{D} \mathbf{w}_\alpha \stackrel{(47)}{=} \lambda_\alpha \quad (49)$$

Thus, the eigenvalues are the value of the objective function for the different eigenvectors. It is intuitively clear that **eigenvectors with small eigenvalue are smooth in the sense that connected nodes tend to have similar values** while eigenvectors with large eigenvalue are more rugged, i.e. connected nodes tend to have different values.

Further reading: (Wikipedia, 2017c).

3.5 The role of the weighted normalization constraint

What is the difference between the constraints $\mathbf{u}_\alpha^T \mathbf{u}_\alpha = 1$ (42) and $\mathbf{w}_\alpha^T \mathbf{D} \mathbf{w}_\alpha = 1$ (47)? Since \mathbf{D} is a diagonal matrix, this simply means that in the constraint the components of the generalized eigenvectors get weighted by $\sqrt{D_{ii}}$ (28) (the square root comes from the fact that in $\mathbf{w}_\alpha^T \mathbf{D} \mathbf{w}_\alpha$ the D_{ii} has to be equally distributed over the two \mathbf{w}_α). For the term $w_i D_{ii} w_i$ to have the same effect size in the constraint, a component w_i with large D_{ii} must be smaller than one with a small D_{ii} . This is illustrated in Figure 4 by the green solid ellipse vs the blue dashed circle. The latter is the set of points with $\mathbf{u}_\alpha^T \mathbf{u}_\alpha = 1$, the former the set with $\mathbf{w}_\alpha^T \mathbf{D} \mathbf{w}_\alpha = 1$ with large D_{ii} and small D_{jj} .

In the figure it is assumed that the determinant of \mathbf{D} is one. That does not need to be the case. It could be any other positive value, depending on how strong the weights of the edges are. However, a consistent scaling of the weights does not change the solution, so we can assume w.l.o.g. that they are scaled such that $|\mathbf{D}| = 1$.

While the constraint differs, the objective function (41, 46) is the same in both cases. It takes the form of an unisotropic paraboloid, like a squeezed champagne glass, indicated in Figure 4 by dotted ellipses. Minimizing it under the constraint means finding the point on the blue circle or green ellipse that comes closest to the inner ellipses. To the extent the D_{ii} differ, the components with larger D_{ii} are favored over components with smaller D_{ii} , because they allow the vector \mathbf{w}_α to move closer to the origin, where the true minimum of the objective function with value 0 lies.

However, this does not mean that all components of \mathbf{w}_α with large D_{ii} become larger relative to those with small D_{ii} . That depends also on the objective function. But the general tendency is that the change from constraint $\mathbf{u}_\alpha^T \mathbf{u}_\alpha = 1$ to constraint $\mathbf{w}_\alpha^T \mathbf{D} \mathbf{w}_\alpha = 1$ makes the values of highly connected nodes (with large D_{ii}) larger relative to less connected nodes (with small D_{ii}).

Why might that be useful? Imagine a square lattice of 7×7 nodes, connected with their four nearest neighbors with equal edge weights one. This looks like a pretty good connectivity to represent the 2D layout of the

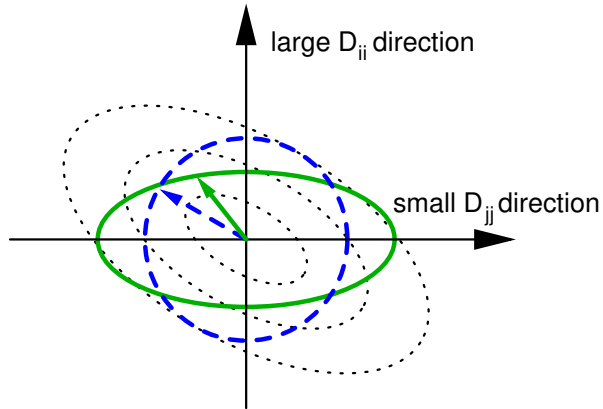


Figure 4: Visualization of the role of the constraint on the optimization problem. The dotted ellipses illustrate the quadratic form being minimized (41, 46), which is the same for both problems. The blue dashed circle and green solid ellipse illustrate the constraints (42) and (47), respectively. The corresponding arrow indicates the optimal solution, which is the point on the circle or ellipse that comes closest to the inner dashed ellipses.

grid. Now, imagine in the right half of the grid, each node is connected to its eight nearest neighbors instead of four. Both, the four- as well as the eight-neighbor connectivity, are perfectly fine representations of the 2D layout. But because the nodes on the right side have more edges, heat would diffuse faster and temperature would equalize more quickly, leading to more similar values, the nodes would move closer together in the embedding. If one uses constraint $\mathbf{w}_\alpha^T \mathbf{D} \mathbf{w}_\alpha = 1$ this advantage of the more densely connected half would be somewhat compensated by scaling up the values, which also leads to larger differences. This leads to a value distribution that better reflects the 2D layout and is less influenced by the different density of connections between left and right half.

It is probably also possible to construct examples where the constraint $\mathbf{u}_\alpha^T \mathbf{u}_\alpha = 1$ gives more desirable results. But at least it should be clear now what the effect of **the constraint $\mathbf{w}_\alpha^T \mathbf{D} \mathbf{w}_\alpha = 1$** is, it **somewhat counteracts the effect of systematically strong (or weak) connections in a region of the graph.** This does not tell much about the effects on a more microscopic level. But it is clear that it makes no sense to change the value of a single highly connected node and make it too different from the values of its neighbors, because that really contributes to a bad value in the objective function.

3.6 Symmetric normalized Laplacian matrix

For the algorithms below, we consider the eigenvalues and -vectors of the generalized eigenvalue equation $L \mathbf{w}_\alpha = \lambda_\alpha \mathbf{D} \mathbf{w}_\alpha$. Since most of us are more familiar with the ordinary eigenvalue equation, it is interesting to note that **one can convert the generalized eigenvalue equation into an ordinary one and back again.** This allows us to transfer what we know about ordinary eigenvalue equations to the generalized ones.

First assume $D_{ii} \neq 0 \forall i$ ($0 \leq D_{ii}$ is true in any case) and define

$$\diamond \quad \mathbf{d} := (D_{11}, \dots, D_{II})^T \tag{50}$$

$$\diamond \quad \bar{\mathbf{d}} := (\sqrt{D_{11}}, \dots, \sqrt{D_{II}})^T \tag{51}$$

$$\diamond \quad \underline{\mathbf{d}} := (1/\sqrt{D_{11}}, \dots, 1/\sqrt{D_{II}})^T \tag{52}$$

$$\diamond \quad \mathbf{D} := \text{diag}(\mathbf{d}) = \mathbf{D}^T \tag{53}$$

$$\diamond \quad \bar{\mathbf{D}} := \text{diag}(\bar{\mathbf{d}}) = \bar{\mathbf{D}}^T \tag{54}$$

$$\diamond \quad \underline{\mathbf{D}} := \text{diag}(\underline{\mathbf{d}}) = \underline{\mathbf{D}}^T \tag{55}$$

so that, for instance, $\bar{\mathbf{D}} \mathbf{D} = \underline{\mathbf{D}} \bar{\mathbf{D}} = \mathbf{I}$ and $\bar{\mathbf{D}} \underline{\mathbf{D}} = \mathbf{D}$.

Now we convert the generalized eigenvalue equation into an ordinary one.

$$\blacklozenge \quad \mathbf{L}\mathbf{w}_\alpha \stackrel{!}{=} \lambda_\alpha \mathbf{D}\mathbf{w}_\alpha \quad | \quad \mathbf{D}. \quad (56)$$

$$\diamond \quad \iff \underbrace{\mathbf{D}\mathbf{L}\mathbf{D}\mathbf{D}}_{=\mathbf{I}}\mathbf{w}_\alpha = \underbrace{\mathbf{D}\lambda_\alpha\mathbf{D}\mathbf{D}}_{=\mathbf{D}}\mathbf{w}_\alpha \quad (\text{since } \mathbf{D} \text{ is invertible}) \quad (57)$$

$$\diamond \quad \iff \underbrace{\mathbf{D}\mathbf{L}\mathbf{D}}_{=: \hat{\mathbf{L}}}\mathbf{D}\mathbf{w}_\alpha = \lambda_\alpha \underbrace{\mathbf{D}\mathbf{D}\mathbf{D}}_{=\mathbf{I}}\mathbf{w}_\alpha \quad (58)$$

$$\blacklozenge \quad \iff \hat{\mathbf{L}}\hat{\mathbf{w}}_\alpha = \lambda_\alpha \hat{\mathbf{w}}_\alpha \quad (59)$$

with

$$\blacklozenge \quad \hat{\mathbf{w}}_\alpha = \mathbf{D}\mathbf{w}_\alpha \quad (60)$$

$$\blacklozenge \quad \iff \mathbf{w}_\alpha = \mathbf{D}\hat{\mathbf{w}}_\alpha \quad (61)$$

and the *symmetric normalized Laplacian matrix*

$$\blacklozenge \quad \hat{\mathbf{L}} := \mathbf{D}\mathbf{L}\mathbf{D} \quad (62)$$

Thus, **if and only if \mathbf{w}_α is an eigenvector of the generalized eigenvalue equation with eigenvalue λ_α , then $\hat{\mathbf{w}}_\alpha$ is an eigenvector of the ordinary eigenvalue equation with same eigenvalue λ_α .** It is sometimes helpful to switch back and forth between these two views.

For the example above we find

$$\hat{\mathbf{L}} = \begin{pmatrix} & \div\sqrt{0.2} & \div\sqrt{1.0} & \div\sqrt{0.8} \\ \div\sqrt{0.2} \rightarrow & \downarrow & \downarrow & \downarrow \\ \div\sqrt{1.0} \rightarrow & 0.2 & -0.2 & 0 \\ \div\sqrt{0.8} \rightarrow & -0.2 & 1.0 & -0.8 \\ & 0 & -0.8 & 0.8 \end{pmatrix} = \begin{pmatrix} & 1.0 & -\sqrt{0.2} & 0 \\ -\sqrt{0.2} & & 1.0 & -\sqrt{0.8} \\ 0 & -\sqrt{0.8} & & 1.0 \end{pmatrix} \quad (63)$$

where $\div\sqrt{\cdot}$ indicates multiplication with \mathbf{D} from the left along the rows and from the right along the columns. It is easy to see that $\hat{\mathbf{L}}_{ii} = 1$ by construction, since $\mathbf{D}\mathbf{L}\mathbf{D} = \mathbf{D}(\mathbf{D} - \mathbf{W})\mathbf{D} = (\mathbf{I} - \mathbf{D}\mathbf{W}\mathbf{D})$ and $\mathbf{D}\mathbf{W}\mathbf{D}$ has only zeroes on the diagonal. But the rows and columns do not add up to zero anymore.

The objective function related to the eigenvalue equation of the symmetric normalized Laplacian matrix is

$$\hat{\mathbf{w}}_\alpha^T \hat{\mathbf{L}}\hat{\mathbf{w}}_\alpha \stackrel{(62)}{=} \hat{\mathbf{w}}_\alpha^T \mathbf{D}\mathbf{L}\mathbf{D}\hat{\mathbf{w}}_\alpha \quad (64)$$

$$= (\mathbf{D}\hat{\mathbf{w}}_\alpha)^T \mathbf{L}\mathbf{D}\hat{\mathbf{w}}_\alpha \quad (\text{since } \mathbf{D} \text{ is diagonal, thus } \mathbf{D} = \mathbf{D}^T) \quad (65)$$

$$\stackrel{(41)}{=} \frac{1}{2} \sum_{ij} ((\mathbf{D}\hat{\mathbf{w}}_\alpha)_i - (\mathbf{D}\hat{\mathbf{w}}_\alpha)_j)^2 W_{ij} \quad (66)$$

$$\stackrel{(55,52)}{=} \frac{1}{2} \sum_{ij} \left(\frac{\hat{w}_{\alpha,i}}{\sqrt{D_{ii}}} - \frac{\hat{w}_{\alpha,j}}{\sqrt{D_{jj}}} \right)^2 W_{ij} \quad (\text{since } \mathbf{D} \text{ is diagonal}) \quad (67)$$

3.7 Random walk normalized Laplacian matrix +

Another possibility to convert the generalized eigenvalue equation into an ordinary one is simply to multiply (45) from the left with the inverse of the weighted degree matrix.

$$\diamond \quad \mathbf{L}\mathbf{w}_\alpha \stackrel{(45)}{=} \lambda_\alpha \mathbf{D}\mathbf{w}_\alpha \quad | \quad \mathbf{D}^{-1}. \quad (68)$$

$$\diamond \quad \iff \underbrace{\mathbf{D}^{-1}\mathbf{L}}_{=: \hat{\mathbf{L}}^{\text{rw}}}\mathbf{w}_\alpha = \lambda_\alpha \mathbf{w}_\alpha \quad (\text{since } \mathbf{D} \text{ is invertible}) \quad (69)$$

$$\diamond \quad \iff \hat{\mathbf{L}}^{\text{rw}}\mathbf{w}_\alpha = \lambda_\alpha \mathbf{w}_\alpha \quad (70)$$

$\hat{\mathbf{L}}^{\text{rw}} := \mathbf{D}^{-1}\mathbf{L}$ is the *random walk normalized Laplacian matrix* and has the same eigenvalues and eigenvectors as the generalized eigenvalue equation of the Laplacian matrix. Its main disadvantage is that it is non-symmetric.

For the example above we find

$$\hat{\mathbf{L}}^{\text{rw}} = \begin{pmatrix} \div 0.2 \rightarrow & 0.2 & -0.2 & 0 \\ \div 1.0 \rightarrow & -0.2 & 1.0 & -0.8 \\ \div 0.8 \rightarrow & 0 & -0.8 & 0.8 \end{pmatrix} = \begin{pmatrix} 1.0 & -1.0 & 0 \\ -0.2 & 1.0 & -0.8 \\ 0 & -1.0 & 1.0 \end{pmatrix} \quad (71)$$

where $\div \cdot$ indicates multiplication with \mathbf{D}^{-1} from the left along the rows. Notice that $\hat{L}_{ii}^{\text{rw}} = 1$ and that the rows, but not the columns, add up to zero. $\mathbf{P} := \mathbf{I} - \hat{\mathbf{L}}^{\text{rw}}$ is a *right stochastic matrix* (Wikipedia, 2017e), which can be interpreted as a transition matrix for a random walk between the nodes of the graph. Therefore the name. We are not sure how useful this intuition is, since the right stochastic matrix has to be multiplied from the right, in order to simulate a random walk, but in the eigenvalue equation $\hat{\mathbf{L}}^{\text{rw}}$ is multiplied from the left.

In what follows we focus on $\hat{\mathbf{L}}$ rather than $\hat{\mathbf{L}}^{\text{rw}}$, because the non-symmetry makes the latter more difficult to deal with.

3.8 Summary of mathematical properties

The Laplacian matrix appears in a multitude of different algorithms, three of which will be discussed in this lecture: *Laplacian eigenmaps (LEM)*, *locality preserving projections (LPP)*, and *spectral clustering*. When using the Laplacian matrix in an algorithm, we are usually interested in its eigenvectors and eigenvalues. The set of eigenvalues of a matrix is referred to as its *spectrum*.

The Laplacian matrix, its eigenvectors, and its spectrum have the following properties:

1. **\mathbf{L} and $\hat{\mathbf{L}}$ are both symmetric (and real).** The symmetry of \mathbf{L} follows directly from equation (31) since \mathbf{D} is diagonal and \mathbf{W} is symmetric. The symmetry of $\hat{\mathbf{L}}$ follows from equation (62) and the symmetry of \mathbf{L} . See (32) and (63) for the example above.
2. **\mathbf{L} and $\hat{\mathbf{L}}$ each have a complete set of orthogonal eigenvectors \mathbf{u}_α and $\hat{\mathbf{w}}_\alpha$, respectively, with real eigenvalues.** This is true for any real symmetric matrix, see Property ⟨1⟩.
3. **\mathbf{L} and $\hat{\mathbf{L}}$ are both positive semi-definite.** For \mathbf{L} this follows directly from (41) and the fact that all weights are positive; for $\hat{\mathbf{L}}$ this follows from equation (62) and the fact that it holds for \mathbf{L} .
4. **\mathbf{L} and $\hat{\mathbf{L}}$ have only non-negative eigenvalues.** This follows from Property ⟨3⟩. Note, however, that the eigenvalues of \mathbf{L} and $\hat{\mathbf{L}}$ may be different. We indicate the eigenvalues of \mathbf{L} by γ_α and those of $\hat{\mathbf{L}}$ by λ_α .
5. **$\hat{\mathbf{L}}\hat{\mathbf{w}}_\alpha = \lambda_\alpha\hat{\mathbf{w}}_\alpha$ and $\mathbf{L}\mathbf{w}_\alpha = \lambda_\alpha\mathbf{D}\mathbf{w}_\alpha$ have the same set of eigenvalues λ_α and their eigenvectors are related by $\mathbf{w}_\alpha = \mathbf{D}\hat{\mathbf{w}}_\alpha \Leftrightarrow \hat{\mathbf{w}}_\alpha = \overline{\mathbf{D}}\mathbf{w}_\alpha$,** see Section 3.6.
6. The generalized eigenvalue equation **$\mathbf{L}\mathbf{w}_\alpha = \lambda_\alpha\mathbf{D}\mathbf{w}_\alpha$ has only non-negative eigenvalues λ_α and a full set of eigenvectors \mathbf{w}_α that are orthogonal with respect to the inner product $\mathbf{w}_\beta\mathbf{D}\mathbf{w}_\alpha$** for $\beta \neq \alpha$. This follows from Properties ⟨2,4⟩ with Property ⟨5⟩, since $\forall \beta \neq \alpha : 0 \stackrel{(2)}{=} \hat{\mathbf{w}}_\beta^T \hat{\mathbf{w}}_\alpha \stackrel{(60)}{=} \mathbf{w}_\beta^T \overline{\mathbf{D}}\mathbf{w}_\alpha = \mathbf{w}_\beta\mathbf{D}\mathbf{w}_\alpha$.
7. **$\mathbf{1} := (1, 1, \dots, 1)^T$ (the one-vector) is a solution of the ordinary eigenvalue equation $\mathbf{L}\mathbf{u}_\alpha = \gamma_\alpha\mathbf{u}_\alpha$ as well as the generalized eigenvalue equation $\mathbf{L}\mathbf{w}_\alpha = \lambda_\alpha\mathbf{D}\mathbf{w}_\alpha$ with eigenvalue 0.** This follows directly from the definition of \mathbf{L} (31), since its rows sum up to zero, and because the two eigenvalue equations are identical for $\gamma_\alpha = \lambda_\alpha = 0$. We chose the appropriately normalized one-vector to be the first eigenvectors $\mathbf{u}_1 = 1/\sqrt{\mathbf{1}^T\mathbf{1}}$ and $\mathbf{w}_1 = 1/\sqrt{\mathbf{1}^T\mathbf{D}\mathbf{1}}$ with $\gamma_1 = \lambda_1 = 0$.
8. **$\bar{\mathbf{d}}$, see (51), is a solution of the ordinary eigenvalue equation $\hat{\mathbf{L}}\hat{\mathbf{w}}_\alpha = \lambda_\alpha\hat{\mathbf{w}}_\alpha$ with eigenvalue 0.** This follows from Property ⟨7⟩ and equation (61) since $\mathbf{D}\bar{\mathbf{d}} = \mathbf{1} \stackrel{(7)}{=} \mathbf{w}_1$. We chose this 'square-root degree-vector' normalized to norm one to be the first eigenvector $\hat{\mathbf{w}}_1 = \bar{\mathbf{d}}/\sqrt{\bar{\mathbf{d}}^T\bar{\mathbf{d}}}$ with $\lambda_1 = 0$.

9. Property [\(7\)](#) generalizes to several eigenvalues with eigenvalue 0 for disconnected graphs (the proof is left to the reader as an exercise). **If a graph has C subgraphs** that are intrinsically connected but not mutually, **then L has C orthogonal eigenvectors with eigenvalue 0**. Each of these eigenvectors has identical values within each of the connected subgraphs and possibly different values between subgraphs. Since it is possible to arbitrarily rotate a set of eigenvectors with identical eigenvalue and still get a set of eigenvectors, **it is possible to chose the eigenvectors with eigenvalue 0 such that each one has the value 1 within a subgraph and value 0 on all other nodes**. Such vectors are **referred to as indicator vectors** ([Wikipedia, 2016](#)). These indicator vectors can then be normalized to fulfill the convention of normalized eigenvectors.
10. If we do not perform the rotation mentioned in Property [\(9\)](#) to get indicator vectors, but rather choose the first eigenvector to be the one-vector, then **all higher eigenvectors of the ordinary eigenvalue equation $L\mathbf{u}_\alpha = \gamma_\alpha \mathbf{u}_\alpha$ have zero mean**, since $\forall \alpha \neq 1 : 0 \stackrel{\langle 2 \rangle}{=} \mathbf{u}_1^T \mathbf{u}_\alpha \stackrel{\langle 7 \rangle}{\iff} 0 = \mathbf{1}^T \mathbf{u}_\alpha = \sum_j u_{\alpha j}$ by Properties [\(2,7\)](#).
11. Similarly, if the first eigenvector is the one-vector **all higher eigenvectors of the generalized eigenvalue equation $L\mathbf{w}_\alpha = \lambda_\alpha D\mathbf{w}_\alpha$ have weighted zero mean** since $\forall \alpha \neq 1 : 0 \stackrel{\langle 6 \rangle}{=} \mathbf{w}_1^T D\mathbf{w}_\alpha \stackrel{\langle 7 \rangle}{\iff} 0 = \mathbf{1}^T D\mathbf{w}_\alpha = \sum_j w_{\alpha j} D_{jj}$ by Properties [\(6,7\)](#).
12. **The eigenvectors are solutions to the optimization problems and the eigenvalues are the values that the objective functions assume for the optimal solutions**, see Section [3.4](#). Equation [\(44\)](#) yields $\mathbf{u}_\alpha^T L\mathbf{u}_\alpha = \gamma_\alpha$, and $\hat{\mathbf{w}}_\alpha^T \hat{L}\hat{\mathbf{w}}_\alpha = \lambda_\alpha$ holds analogously. For the generalized eigenvalue equation, we find [\(49\)](#) $\mathbf{w}_\alpha^T L\mathbf{w}_\alpha = \lambda_\alpha$.

Further reading: ([Wikipedia, 2017b](#)).

4 Algorithms

4.1 Similarity graphs

The algorithms presented in the following are all based on the properties of the Laplacian matrix discussed above. In order to take advantage of the Laplacian matrix, though, any **input data first has to be represented as a graph**, commonly **referred to as a similarity graph**: A simple graph where the nodes represent individual data samples and edge weights denote the similarity (or distance) between two connected nodes, i.e. data samples. Appropriate similarity metrics depend on the problem and can be as simple as the Euclidean or Manhattan distance between two points.

There are different ways to construct a similarity graph, depending on the problem at hand (e.g. [Belkin and Niyogi, 2003](#), Sec. 2; [He and Niyogi, 2004](#), Sec. 2.2; [Von Luxburg, 2007](#), Sec. 2). Three common methods are ϵ -neighborhood, k -nearest neighbors, and fully connected graphs:

- **ϵ -neighborhood: Two nodes are connected if the distance between them is smaller than a given threshold ϵ** . Often ϵ is chosen so small that the distance values within an ϵ -neighborhood do not carry much useful information. In this case **edges are often weighted binary**, i.e., **with 1 or 0** depending on whether the data samples in question are close enough or not, respectively.
- **k -nearest neighbors: Node v_i is connected to v_j if v_j is among the k nearest neighbors of v_i** . Note that **this neighborhood relation is not symmetric** and yields a directed graph, thus **some cleanup is required**. To arrive at a simple graph we take each unilateral edge that has no mirrored counterpart and either remove it or keep it and set it as bilateral. Removal results in a graph where each node has at most k neighbors (*mutual k -nearest neighbor graph*), while setting unilateral edges to bilateral results in a graph where each node has at least k neighbors (*k -nearest neighbor graph*). **All edges are weighted by the similarity between the two nodes they connect**. Binary weighting, as in the preceding method, is more dangerous here, because it cannot be guaranteed that connected nodes are close to each other.

<p>(28) Degree matrix: $\mathbf{D} : D_{ij} := \delta_{ij} \sum_j W_{ij}$</p> <p>(31) Laplacian matrix: $\mathbf{L} = \mathbf{D} - \mathbf{W}$</p> <p>(1) Is symmetric: $\mathbf{L} = \mathbf{L}^T$</p> <p>(3) Is positive semi-definite: $\mathbf{x}^T \mathbf{L} \mathbf{x} \geq 0 \forall \mathbf{x}$</p>	<p>Weight matrix: \mathbf{W}</p>	<p>(55) $\underline{\mathbf{D}} := \text{diag}(1/\sqrt{D_{11}}, \dots, 1/\sqrt{D_{II}})$</p> <p>(62) Sym. norm. Lapl. matrix: $\hat{\mathbf{L}} := \underline{\mathbf{D}} \mathbf{L} \underline{\mathbf{D}}$</p> <p>(1) Is symmetric: $\hat{\mathbf{L}} = \hat{\mathbf{L}}^T$</p> <p>(3) Is positive semi-definite: $\mathbf{x}^T \hat{\mathbf{L}} \mathbf{x} \geq 0 \forall \mathbf{x}$</p>
<p>(40) Ordinary eigenvalue equation: $\mathbf{L} \mathbf{u}_\alpha = \gamma_\alpha \mathbf{u}_\alpha$</p> <p>Optimization problem: minimize</p> <p>(41) $\mathbf{u}_\alpha^T \mathbf{L} \mathbf{u}_\alpha = \frac{1}{2} \sum_{ij} (u_{\alpha,i} - u_{\alpha,j})^2 W_{ij}$</p> <p>(42,43) subject to $\mathbf{u}_\beta^T \mathbf{u}_\alpha = \delta_{\beta\alpha} \forall \beta \leq \alpha$</p> <p>Trivial first solution:</p> <p>(7) $\mathbf{u}_1 = \mathbf{1}/\sqrt{\mathbf{1}^T \mathbf{1}}$ with $\gamma_1 = 0$</p> <p>Objective function value:</p> <p>(12),(44) $\mathbf{u}_\alpha^T \mathbf{L} \mathbf{u}_\alpha = \gamma_\alpha$</p>	<p>(45) Generalized eigenvalue equation: $\mathbf{L} \mathbf{w}_\alpha = \lambda_\alpha \mathbf{D} \mathbf{w}_\alpha$</p> <p>Optimization problem: minimize</p> <p>(46) $\mathbf{w}_\alpha^T \mathbf{L} \mathbf{w}_\alpha = \frac{1}{2} \sum_{ij} (w_{\alpha,i} - w_{\alpha,j})^2 W_{ij}$</p> <p>(47,48) subject to $\mathbf{w}_\beta^T \mathbf{D} \mathbf{w}_\alpha = \delta_{\beta\alpha} \forall \beta \leq \alpha$</p> <p>Trivial first solution:</p> <p>(7) $\mathbf{w}_1 = \mathbf{1}/\sqrt{\mathbf{1}^T \mathbf{D} \mathbf{1}}$ with $\lambda_1 = 0$</p> <p>Objective function value:</p> <p>(12),(49) $\mathbf{w}_\alpha^T \mathbf{L} \mathbf{w}_\alpha = \lambda_\alpha$</p>	<p>(59) Ordinary eigenvalue equation: $\hat{\mathbf{L}} \hat{\mathbf{w}}_\alpha = \lambda_\alpha \hat{\mathbf{w}}_\alpha$</p> <p>Optimization problem: minimize</p> <p>(67) $\hat{\mathbf{w}}_\alpha^T \hat{\mathbf{L}} \hat{\mathbf{w}}_\alpha = \frac{1}{2} \sum_{ij} \left(\frac{\hat{w}_{\alpha,i}}{\sqrt{D_{ii}}} - \frac{\hat{w}_{\alpha,j}}{\sqrt{D_{jj}}} \right)^2 W_{ij}$</p> <p>subject to $\hat{\mathbf{w}}_\beta^T \hat{\mathbf{w}}_\alpha = \delta_{\beta\alpha} \forall \beta \leq \alpha$</p> <p>Trivial first solution:</p> <p>(8) $\hat{\mathbf{w}}_1 = \underline{\mathbf{d}}/\sqrt{\underline{\mathbf{d}}^T \underline{\mathbf{d}}}$ with $\lambda_1 = 0$</p> <p>Objective function value:</p> <p>(12) $\hat{\mathbf{w}}_\alpha^T \hat{\mathbf{L}} \hat{\mathbf{w}}_\alpha = \lambda_\alpha$</p>
<p>Relation between two solutions:</p> <p>$\lambda_\alpha = \lambda_\alpha$</p> <p>$\mathbf{w}_\alpha = \underline{\mathbf{D}} \hat{\mathbf{w}}_\alpha$</p>		

Table 1: Overview over different Laplacian matrices, eigenvalue equations, optimization problems, and solutions.

- **Fully connected:** To construct a fully connected graph each data sample is simply connected to all others. In this case, using binary weights renders the graph entirely meaningless. A fully connected graph **always requires weighting the edges with a similarity function** (e.g. a Gaussian similarity function for vectorial data $w_{ij} = w_{ji} = s(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / (2\sigma^2))$ where σ defines the extent of local neighborhoods).

4.2 Laplacian eigenmaps (LEM)

4.2.1 Motivation

Many algorithms work only on vectorial data and are limited in the dimensionality they can process efficiently. This causes problems if one has data that is either not vectorial, such as text, or too high dimensional, such as images, or both. If one can define a similarity function on the data, yielding a scalar similarity value for each pair of data samples, the Laplacian eigenmaps algorithm can provide a low-dimensional vectorial embedding of the data that tends to preserve similarity relationships and allows to apply other algorithms to the data that would not be applicable directly (Belkin and Niyogi, 2003). Laplacian eigenmaps are also very good for a 2- or 3-dimensional visualization of data.

Example: Imagine a drone hovering through the air while equipped with a downward facing camera. Using the high dimensional pictures from its camera, we could, in theory, precisely compute the drone's current position and elevation. Unfortunately, the space of all possible high dimensional images is effectively intractable. Luckily though, we are merely interested in a small subset of this space, namely only those images the drone's camera can actually produce in a particular environment. And while each data point of this vastly smaller subset still is of the original, high dimensionality, it can be fully described by six dimensions alone: the position and orientation of the drone in 3D space. Laplacian eigenmaps can be used to find a low dimensional embedding of the images that still permits extracting positional and orientation information.

4.2.2 Objective

The objective of the Laplacian eigenmaps algorithm is to find an embedding of a set of I data samples (do not need to be vectors, but there must be a similarity function) **in a low-dimensional vector space $\{\mathbf{y}_1, \dots, \mathbf{y}_I\}$ such that samples with high similarity are close to each other in the embedding.** For dimensionality $M = 1$, i.e. an embedding in only a 1-dimensional space, this objective translates into minimizing

$$\frac{1}{2} \sum_{ij} (y_i - y_j)^2 W_{ij} \quad (72)$$

where the y_i are the values assigned to the samples and W_{ij} indicates the similarity between two samples. We have already seen above how this optimization problem is solved by the second eigenvector of the Laplacian matrix, (41) or (46) depending on the constraint. Each additional eigenvector adds one orthogonal (meaning the values are uncorrelated) dimension to the embedding provided by the other eigenvectors already. The quality of the embedding induced by each eigenvector is given by its associated eigenvalue, which directly relates to the actual value of sum (72). **The best M -dimensional embedding is thus given by the first M eigenvectors w_α of the Laplacian matrix with smallest eigenvalues (excluding the first one).**

Please notice that the dimension of the eigenvectors corresponds to the number I of data points, because the Laplacian matrix is $I \times I$ by construction. Thus, if you arrange the first M eigenvectors as rows in a matrix, this matrix will be $M \times I$ and the column vectors are the data points \mathbf{y}_i in the M -dimensional embedding. For instance, three data samples embedded in a 2-dimensional space with LEM using the ordinary eigenvalue problem (for simplicity) could yield

$$\left(\begin{array}{ccc} & \mathbf{y}_1 & \mathbf{y}_2 & \mathbf{y}_3 \\ & \downarrow & \downarrow & \downarrow \\ \mathbf{u}_2 \rightarrow & -1/\sqrt{2} & 0 & +1/\sqrt{2} \\ \mathbf{u}_3 \rightarrow & -1/\sqrt{6} & +2/\sqrt{6} & -1/\sqrt{6} \end{array} \right) \quad (73)$$

As usual, we have dropped \mathbf{u}_1 , because it has equal components throughout, e.g. $(1, 1, 1)^T/\sqrt{3}$; \mathbf{u}_2 and \mathbf{u}_3 have zero mean, because they need to be orthogonal to \mathbf{u}_1 ; and \mathbf{u}_2 and \mathbf{u}_3 are orthogonal to each other as well.

We now have all the required components to formulate the Laplacian eigenmaps algorithm.

4.2.3 Algorithm

Laplacian eigenmaps algorithm (Belkin and Niyogi, 2003)

1. Given a set of I data samples, **construct a similarity graph** G according to one of the methods described in Section 4.1.
2. **Construct the** $I \times I$ weight matrix \mathbf{W} , degree matrix \mathbf{D} (28), and **Laplacian matrix** \mathbf{L} (31) for G .
3. **Compute the first** $M + 1$ **eigenvectors** w_α **of the generalized eigenvalue problem**

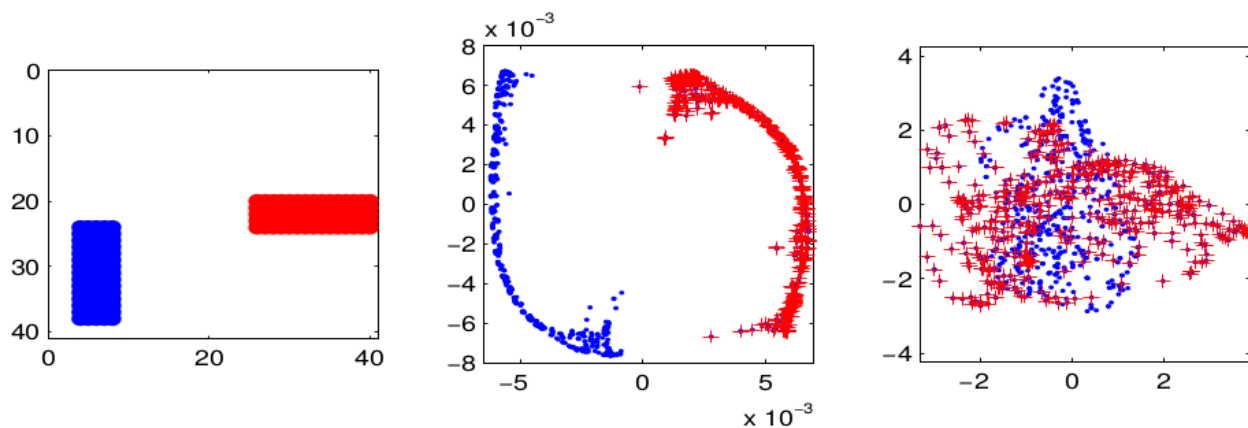
$$\diamond \quad \mathbf{L}w_\alpha = \lambda_\alpha \mathbf{D}w_\alpha \quad (74)$$

ordered by increasing eigenvalues.

4. **An** M -**dimensional representation of data sample** i **is now given by** $(w_{2,i}, \dots, w_{M+1,i})^T$.

4.2.4 Sample applications

Figure 5 shows a toy example of **dimensionality reduction of 1000 images of size 40×40 with either a vertical or a horizontal bar** (Belkin and Niyogi, 2002). One can clearly see how the images with the horizontal bar are separate from the images with the vertical bar. It would be interesting to see a three dimensional Laplacian eigenmap, because presumably the red and blue points would each form a square manifold representing x - and y -position. A projection onto the first two principal components is shown for comparison.



(Belkin and Niyogi, 2002, Fig. 1, URL)¹

Figure 5: Dimensionality reduction of 1000 40×40 images with either a vertical or a horizontal bar (plotted together but distinguished by color for illustrative purposes, the original is in grayscale). Left: Two input images superimposed, one with a horizontal bar (red) one with a vertical bar (blue). Middle: Result of LEM. Right: Result of PCA for comparison.

Figures 6 and 7 show an application of Laplacian eigenmaps to a set of 300 frequently used words (Belkin and Niyogi, 2003). Each word was represented by a 600-dimensional vector indicating how often any of the other words was found to the left or to the right of the considered word. Similarity was defined based on these 600-dimensional vectors. **Zooming into Figure 7** shows that **grammatically closely related words are grouped together.**

Further reading: (Belkin and Niyogi, 2003).

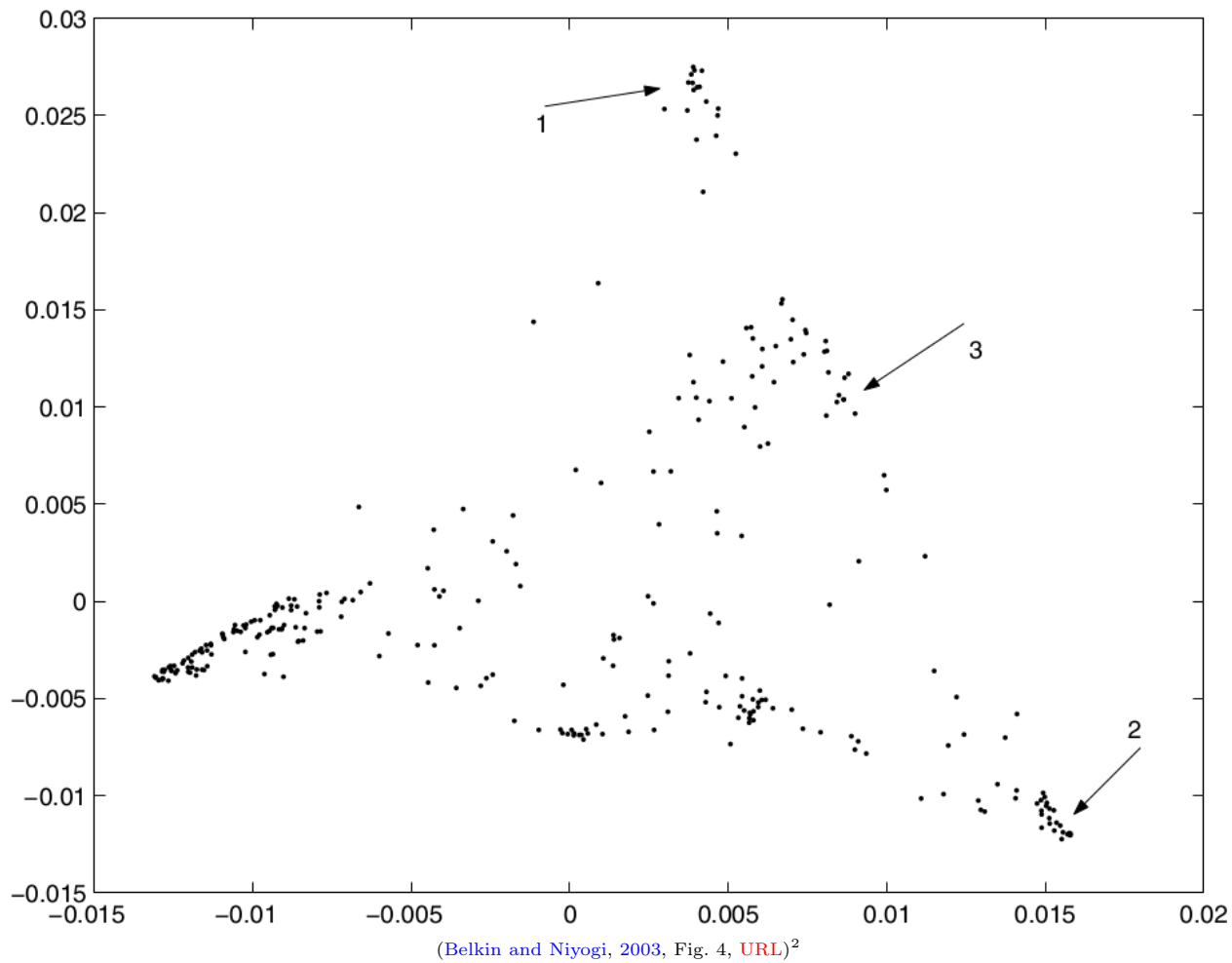
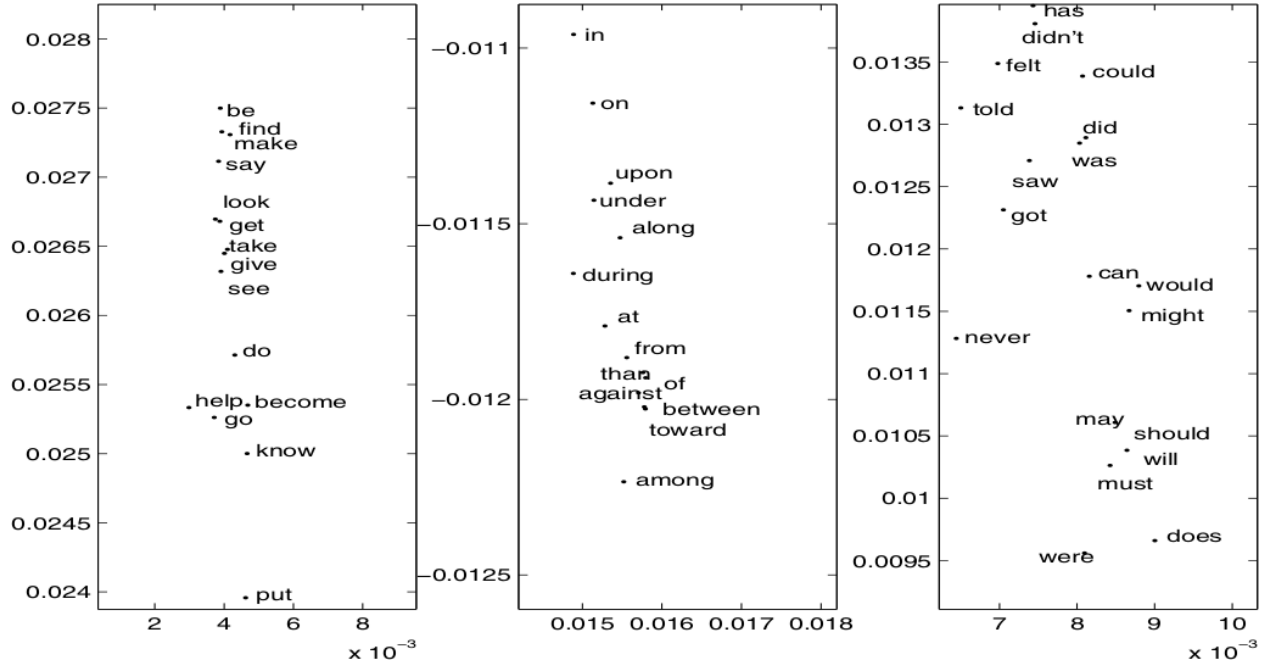


Figure 6: Dimensionality reduction for 300 frequently used words from their word context data.



(Belkin and Niyogi, 2003, Fig. 5, URL)³

Figure 7: Zoom-in into the three subregions marked in Figure 6. Left infinitives, middle prepositions, and right mostly modal and auxiliary verbs.

4.3 Locality preserving projections (LPP)

4.3.1 Linear LPP

Laplacian eigenmaps have the disadvantage that they **only provide values for the data used during training**. There is no straight forward way to process new data. **This can be changed if** the nodes v_i are data points in Euclidean space $v_i = \mathbf{x}_i \in \mathbb{R}^N$ and **the values of the eigenvectors \mathbf{w}_α are approximated by linear functions in the data points** (He and Niyogi, 2004). Since the values of the nodes are now computed with a linear function rather than assigned freely, new data can be processed by applying the same linear function. On the training data the linear function yields the values of the nodes as follows

$$\diamond \quad \mathbf{w}_{\alpha,i} = \mathbf{x}_i^T \mathbf{z}_\alpha \quad (75)$$

$$\blacklozenge \quad \iff \quad \mathbf{w}_\alpha = \mathbf{X}^T \mathbf{z}_\alpha \quad (76)$$

$$\blacklozenge \quad \text{with data} \quad \mathbf{X} := (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_I) \quad (77)$$

The vectors \mathbf{z}_α are the variables to be optimized. Inserting this in (46) and the corresponding constraints (47,48) yields

$$\blacklozenge \quad \text{minimize} \quad \mathbf{w}_\alpha^T \mathbf{L} \mathbf{w}_\alpha \stackrel{(76)}{=} \mathbf{z}_\alpha^T \underbrace{\mathbf{X} \mathbf{L} \mathbf{X}^T}_{=: \mathbf{L}'} \mathbf{z}_\alpha = \mathbf{z}_\alpha^T \mathbf{L}' \mathbf{z}_\alpha \quad (78)$$

$$\diamond \quad \text{subject to} \quad 1 = \mathbf{w}_\alpha^T \mathbf{D} \mathbf{w}_\alpha \stackrel{(76)}{=} \mathbf{z}_\alpha^T \underbrace{\mathbf{X} \mathbf{D} \mathbf{X}^T}_{=: \mathbf{D}'} \mathbf{z}_\alpha = \mathbf{z}_\alpha^T \mathbf{D}' \mathbf{z}_\alpha \quad (79)$$

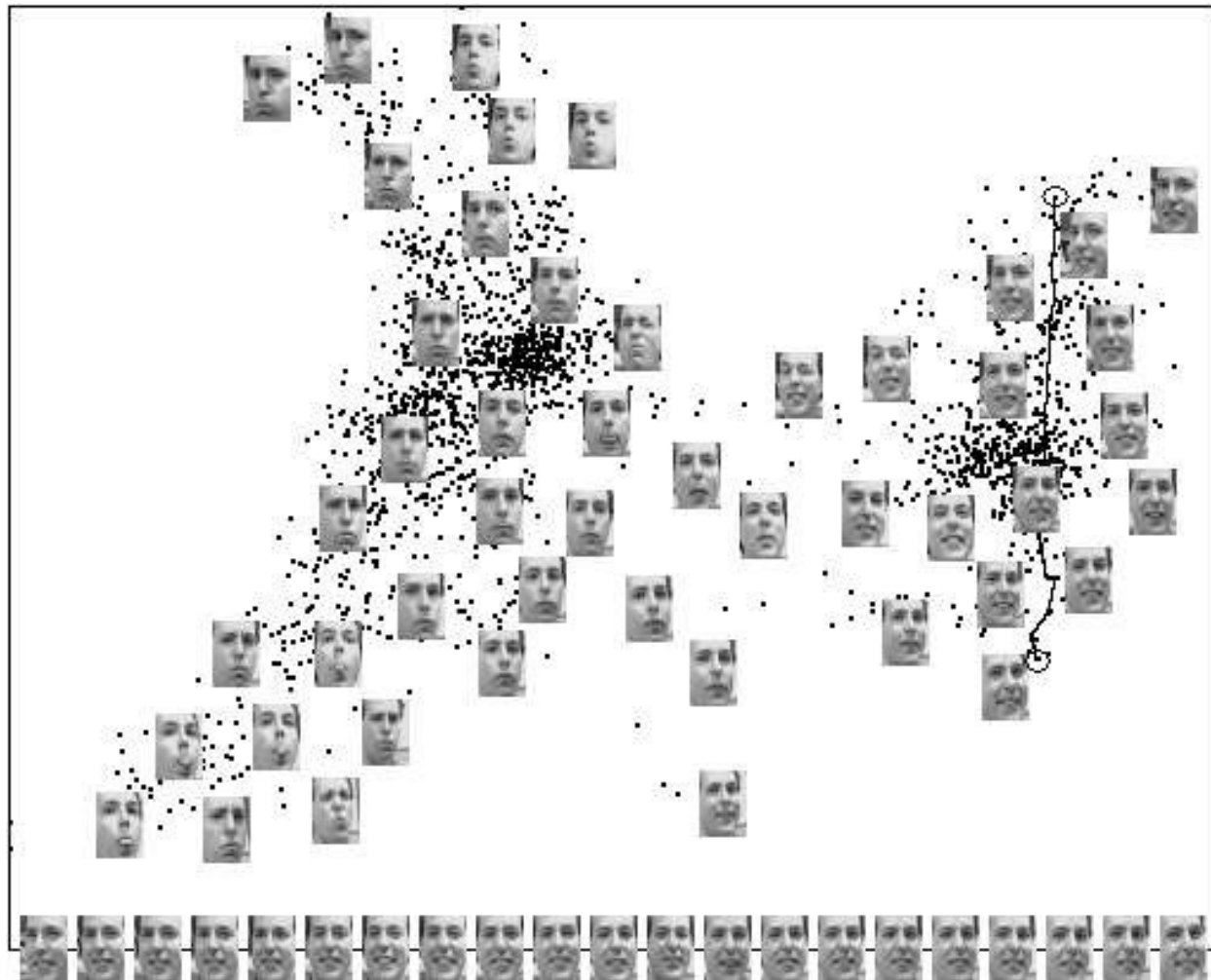
$$\diamond \quad \text{and} \quad 0 = \mathbf{w}_\beta^T \mathbf{D} \mathbf{w}_\alpha \stackrel{(76)}{=} \mathbf{z}_\beta^T \underbrace{\mathbf{X} \mathbf{D} \mathbf{X}^T}_{=: \mathbf{D}'} \mathbf{z}_\alpha = \mathbf{z}_\beta^T \mathbf{D}' \mathbf{z}_\alpha \quad \forall \beta < \alpha \quad (80)$$

This optimization problem can again be solved through a generalized eigenvalue problem, **much like the original one**. Notice, however, that the eigenvalues and the approximated eigenvectors \mathbf{w}_α are not necessarily identical to those of the original eigenvalue problem, because $\mathbf{w}_\alpha \in \mathbb{R}^I$ is not free but constrained to be a linear function in the $\mathbf{x}_i \in \mathbb{R}^N$. Notice also that this problem is not of the dimensionality of the

number I of data points as before but only of the dimension N of the data points, which is usually much smaller and, consequently makes this approximation more computationally efficient. For instance, if you have 100 data points in 3D, the problem is 3-dimensional not 100-dimensional as for the LEM algorithm. The main advantage, however, is that new data points \mathbf{x}_j can easily be mapped into the low-dimensional space by applying the linear function $\mathbf{x}_j^T \mathbf{z}_\alpha$. Performing Laplacian eigenmaps with **this** linear approximation is referred to as *locality preserving projections (LPP)*.

4.3.2 Sample application

An application of LPP to face images of a single person is shown in Figure 8 (He and Niyogi, 2004). Even though the mapping is only linear, LPP still captures some prominent variations and orders the images nicely in 2D. The person looks to the left (or right) at the top (or bottom) of the plot, and it smiles on the right side while it makes faces on the left.



(He and Niyogi, 2004, Fig. 3, [URL](#))⁴

Figure 8: Dimensionality reduction of face images of a single person down to two dimensions with linear LPP. Face images in the plot indicate what some points stand for and the line of faces at the bottom corresponds to the line of data points on the right.

4.3.3 Nonlinear LPP

LPP can be generalized to nonlinear functions by adding a nonlinear expansion prior to the algorithm. Assume $\mathbf{f}(\mathbf{x})$ is such a nonlinear expansion from $\mathbb{R}^N \rightarrow \mathbb{R}^P$ with $N \ll P$, then one can define

$$w_{\alpha,i} = \mathbf{f}(\mathbf{x}_i)^T \mathbf{z}_\alpha \quad (81)$$

$$\iff \mathbf{w}_\alpha = \mathbf{F}^T \mathbf{z}_\alpha \quad (82)$$

$$\text{with } \mathbf{F} := (\mathbf{f}(\mathbf{x}_1), \mathbf{f}(\mathbf{x}_2), \dots, \mathbf{f}(\mathbf{x}_I)) \quad (83)$$

and then run the algorithm as before. Notice that now $\mathbf{z}_\alpha \in \mathbb{R}^P$ rather than \mathbb{R}^N .

Further reading: (He and Niyogi, 2004).

4.4 Spectral clustering

4.4.1 Objective

Spectral clustering is an umbrella term for a number of algorithms that use the eigenvectors of the Laplacian matrix to perform clustering on a given set of data points. In particular, spectral clustering is often used in image processing to identify connected parts of a given image and, ideally, identify the extent of the individual components of an image, a process called *image segmentation*.

As illustrated intuitively in Figure 2 **the eigenvectors of the Laplacian matrix place the nodes of connected subgraphs at the same location**, even in two, three, or higher dimensions, if the graph has several subgraphs. This also holds for the eigenvectors of the generalized eigenvalue problem, and this also holds approximately if the subgraphs are not completely separate from each other. **Given this representation it is much easier than on the original data to cluster the nodes with some standard clustering algorithm.**

Remember that for C intrinsically connected but mutually disconnected subgraphs, i.e. clusters, there are exactly C eigenvectors with constant values on each of the clusters. For extracting C clusters one would therefore use the first C eigenvectors, this time including also the first one, see Property (9).

4.4.2 Algorithm

Normalized spectral clustering algorithm (Ng et al., 2002)

1. Given a set of I data samples, **construct a similarity graph G** according to one of the methods described in Section 4.1. For instance, when performing segmentation on a single image, each pixel becomes a node of the graph with similarity between nodes usually being a function of color and spatial distance.
2. **Compute the** weight matrix \mathbf{W} , degree matrix \mathbf{D} (28), and **Laplacian matrix L** (31) **for G .**
3. **Compute the first C eigenvectors of the generalized eigenvalue problem**

$$\diamond \quad L\mathbf{w}_\alpha = \lambda_\alpha \mathbf{D}\mathbf{w}_\alpha \quad (84)$$

ordered by increasing eigenvalue.

4. **Arrange the eigenvectors w_1, \dots, w_C in the rows[†] of a matrix U and normalize its columns to one to get matrix T with**

$$T_{ij} = U_{ij} / \left(\sum_{i'} U_{i'j}^2 \right)^{1/2} \quad (85)$$

A C -dimensional representation \mathbf{y}_i of data sample i is now given by the i -th column vector of T .

5. **Perform the k -means algorithm** on the set of embedded data points $\{\mathbf{y}_1, \dots, \mathbf{y}_I\}$ to partition the data into C clusters.

[†]In the original formulation (Ng et al., 2002), the vectors were arranged in columns. We use rows here for consistency with the LEM algorithm, see Sec. 4.2.3.

4.4.3 Sample application

Figure 9 shows an example of applying spectral clustering to an old data set collected by Edgar Anderson (Wikipedia, 2017d). He measured length and width of the sepal and petal from 50 exemplars of three types of iris. One species (red in the left plot) is well separated from the other two, which in turn are hard to distinguish in the 2D plots. Spectral clustering performs fairly well on this task in 4D as one can see by comparing ground truth on the left with the clustering result on the right.

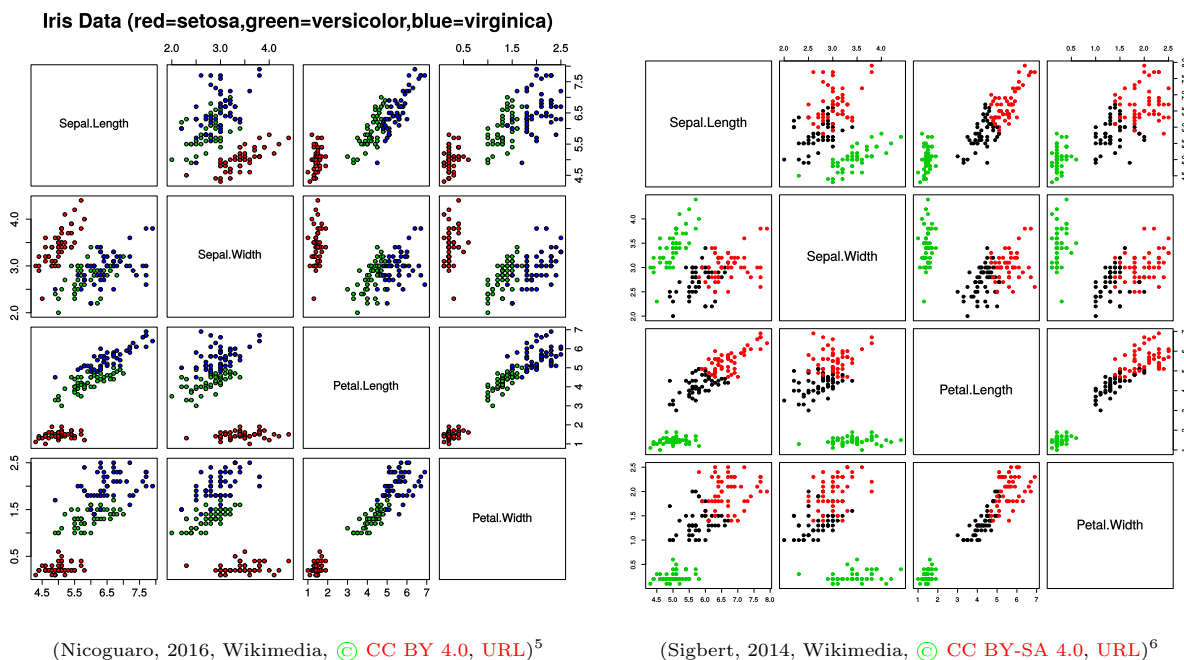


Figure 9: Spectral clustering on iris (the plant, not the eye) data. Left: length and width of the sepal and petal from 50 exemplars of three types of iris as indicated by the three colors. Right: Result of spectral clustering on the 150 four-dimensional data points.

Further reading: (Von Luxburg, 2007), an excellent tutorial on spectral clustering.

Acknowledgments: We thank Jan Melchior and Merlin Schüler for valuable feedback on an earlier version of these lecture notes.

References

- Belkin, M. and Niyogi, P. (2002). Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems*, volume 14, pages 585–591. 17
- Belkin, M. and Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396. 14, 16, 17, 18, 19
- He, X. and Niyogi, P. (2004). Locality preserving projections. In Thrun, S., Saul, L., and Schölkopf, B., editors, *Advances in neural information processing systems*, pages 153–160. MIT Press. 14, 19, 20, 21
- Ng, A. Y., Jordan, M. I., and Weiss, Y. (2002). On spectral clustering: analysis and an algorithm. *Advances in Neural Information Processing Systems*, 14:849–856. 21
- Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416. 14, 22
- Wikipedia (2016). Indicator vector — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Indicator_vector&oldid=743797854, accessed 3 December 2017. 14

Wikipedia (2017a). Graph (discrete mathematics) — Wikipedia, The Free Encyclopedia. [https://en.wikipedia.org/w/index.php?title=Graph_\(discrete_mathematics\)&oldid=800782160](https://en.wikipedia.org/w/index.php?title=Graph_(discrete_mathematics)&oldid=800782160), accessed 3 December 2017. 7

Wikipedia (2017b). Laplacian matrix — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Laplacian_matrix&oldid=812863352, accessed 3 December 2017. 14

Wikipedia (2017c). Rayleigh quotient — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Rayleigh_quotient&oldid=808561799, accessed 3 December 2017. 10

Wikipedia (2017d). Spektrales Clustering — Wikipedia, Die freie Enzyklopädie. https://de.wikipedia.org/w/index.php?title=Spectral_Clustering&oldid=170428156, accessed 2 December 2017. 22

Wikipedia (2017e). Stochastic matrix — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Stochastic_matrix&oldid=813141273, accessed 3 December 2017. 13


Notes

¹Belkin & Niyogi, 2002, NIPS, Fig. 1, <http://papers.nips.cc/paper/1961-laplacian-eigenmaps-and-spectral-techniques-for-embedding-and-clustering.pdf>

²Belkin & Niyogi, 2003, Neur. Comp., Fig. 4, <https://pdfs.semanticscholar.org/989a/f45f8242b96cecb91d48b85620e7322e4aa7.pdf>

³Belkin & Niyogi, 2003, Neur. Comp., Fig. 5, <https://pdfs.semanticscholar.org/989a/f45f8242b96cecb91d48b85620e7322e4aa7.pdf>

⁴He and Niyogi, 2004, NIPS, Fig. 3, <http://papers.nips.cc/paper/2359-locality-preserving-projections.pdf>

⁵Nicoguardo, 2016, Wikimedia,  CC BY 4.0, https://commons.wikimedia.org/wiki/File:Iris_dataset_scatterplot.svg

⁶Sigbert, 2014, Wikimedia,  CC BY-SA 4.0, https://commons.wikimedia.org/wiki/File:Specclus_iriscluster.svg