into separate groups [8], it is sensitive to overfitting, and besides, it neglects the class information, which will impair the recognition accuracy [15]. However, the LLE problem can be rephrased to use class label information, and some supervised versions of LLE [2, 5, 16] were introduced. The intuition of these algorithms is obtain disjoint embedding for the individual classes.

In supervised LLE (or δ-LLE), the local neighborhood of a sample $\mathbf{x}_i$ (first step Algorithm 2) from class c ($1 \leq c \leq C$), should be composed of samples belonging to the same class only. This can be achieved by artificially increasing the pre-calculated distances between samples belonging to different classes, but leaving them unchanged if samples are from the same class:

$$D\left(\mathbf{x}_i, \mathbf{x}_j\right)' = \begin{cases} D\left(\mathbf{x}_i, \mathbf{x}_j\right) + \delta \max\left(D(\mathbf{x}_i, \mathbf{x}_j)\right), & if \ \{\mathbf{x}_i, \mathbf{x}_j\} \in \text{ the same class} \\ D\left(\mathbf{x}_i, \mathbf{x}_j\right), & \text{otherwise} \end{cases} \tag{3.14}$$

where $0 \leq \delta \leq 1$. If $\delta = 0$, one obtains unsupervised LLE; when $\delta = 1$, the result is the fully supervised LLE. Varying δ between 0 and 1 gives a partially supervised LLE [16]. Once the neighborhoods are found by means of (3.14) the LLE algorithm is computed with no changes.

3.3.1 Discussion about LLE for Classification

- Supervised LLE introduces an additional parameter δ, controlling how much of the class label information should be taken into account. This parameter is related to the generalization capacity of the algorithm. When $\delta = 1$ or nearly, the algorithm overfit the clusters on low dimensional space. On the other hand, a low value for δ implies that the class information labels are not taking into account.

- Supervised LLE does not work on low-dimensional data. Some simple classifiers often work well on the original data, as the number of parameters that need to be estimated is still reasonably low. In these cases, supervised LLE will not improve classification to the point where it is better than on the original data [2].

- Supervised LLE works well where k-NN classifier works well on the original data. It is a neighborhood-based method, like the k-NN classifier. In fact, Supervised LLE can be seen as a generalized k-NN method, where not only the neighbors labels play a role, but also the distances to these neighbors [16].

- The performance of the supervised LLE algorithm is very sensitive to the chosen value for δ and to the intrinsical local dimensionality computed as it is advised in (see Equation (4.33)). Besides, there are not objective methodologies for tuning δ.

3.4 Maximum Variance Unfolding

Maximum Variance Unfolding (MVU) [17] expresses the nonlinear dimensionality reduction as a semidefinite programming problem. This technique is building on earlier frameworks for analyzing high dimensional data that lies on or near a low dimensional manifold.

Let $\mathbf{X}$ be the high–dimensional input matrix of size $n \times p$, where the sample vectors $\mathbf{x}_i \in \mathbb{R}^p$, $i = 1, \ldots, n$ are mapped to a low–dimensional space, where the outputs $\{\mathbf{y}_i | \mathbf{y}_i \in \mathbb{R}^m\}_{i=1}^n$, and $m \ll p$.

The algorithm is based on a simple intuition. The inputs $\mathbf{x}_i$ are virtually connected to their k nearest neighbors by rigid rods. The MVU technique attempts to pull apart, maximizing the total sum of their pairwise distances without breaking the rigid rods that connect nearest neighbors. The outputs are obtained from the final state of this transformation.

Let $\zeta_{ij} \in \{0, 1\}$ denote whether inputs $\mathbf{x}_i$ and $\mathbf{x}_j$ are k nearest neighbors. The output $\mathbf{y}_i$ from maximum variance unfolding are those that solve the following optimization

$$\begin{aligned} &\max_{\mathbf{y}} \sum_{i=1}^{n} \sum_{j=1}^{n} \|\mathbf{y}_i - \mathbf{y}_j\|^2 \\ &\text{s.t.} \begin{cases} \|\mathbf{y}_i - \mathbf{y}_j\|^2 = \|\mathbf{x}_i - \mathbf{x}_j\|^2 \text{ if } \zeta_{ij} = 1 \\ \sum_{i=1}^{n} \mathbf{y}_i = 0 \end{cases} \end{aligned} \tag{3.15}$$

The optimization over the outputs $\mathbf{y}_i$ is not convex, meaning that it potentially suffers from spurious local maxima. Defining the inner product matrix $\mathbf{K}$, whose elements are $K_{ij} = \langle \mathbf{y}_i, \mathbf{y}_j \rangle$, the optimization as a semidefinite program (SDP) can be reformulated

$$\max_{\mathbf{y}} \operatorname{tr}(\mathbf{K}) \text{ s.t.} \begin{cases} K_{ii} - 2K_{ij} + K_{jj} = \|\mathbf{x}_i - \mathbf{x}_j\|^2 \text{ if } \zeta_{ij} = 1 \\ \sum_{i=1}^{n} \sum_{j=1}^{n} K_{ij} = 0 \\ \mathbf{K} \geq 0 \end{cases} \tag{3.16}$$

The last constraint requires the matrix $\mathbf{K}$ to be positive semidefinite, then the SDP is convex. We can derive outputs $\mathbf{y}_i$ satisfying $K_{ij} = \langle \mathbf{y}_i, \mathbf{y}_j \rangle$ by singular value decomposition. An m–dimensional representation that approximately satisfies $K_{ij} \approx \langle \mathbf{y}_i, \mathbf{y}_j \rangle$ can be obtained from the top m eigenvalues and eigenvectors of $\mathbf{K}$.

3.4.1 Discussion about MVU

- A notable disadvantage of MVU is the time required to solve a semidefinite program for $n \times n$ matrices when the number of data points n is large. As originally formulated, the size of the SDP scales linearly with the number of observations n.

- MVU is flexible to be adapted to particular applications. For example the distance–preserving constraints in SDP can be relaxed to handle noisy data or to yield more aggressive results in dimensionality reduction [17].

- MVU assumes that the data manifold is sufficiently smooth and densely sampled that it is locally approximately linear.

Chapter 4

Free parameters in the Locally Linear Embedding Algorithm

One of the main advantages of LLE algorithm is that just three parameters are needed to be set by user: the regularization parameter α, the number of nearest neighbors k, and the dimensionality of embedding space m (see section 3.2.2). This small amount of free parameters is quite desirable in contrast to other algorithms employed for nonlinear dimensionality reduction such as a mixtures of linear models, generative topographic mapping, etc., which need to set a lot of parameters [2]. Nevertheless, these parameter are highly important and they can not be ignored or poorly chosen, because they have a strong influence in the embedding results. On the other hand, empirical selection of each one of them can become a tortuous work, in this sense, it is essential to provide objective criteria for them selection.

4.1 Regularization Parameter Choice

The output low–dimensional solution obtained by means of LLE is very sensitive to regularization, since small variations of this parameter produce different outcomes in low dimensional space. In fact, the choice of the regularization parameter, as part of LLE training, has been considered frequently as an important issue. For example, an empirical approach for dealing with the ill-posed problem is formulated in [4], where the regularization is done by adding a small factor of the identity matrix to the Gram matrix. This amounts to penalizing large weights that exploit correlations beyond some level of precision in the data sampling process. Nevertheless, estimating precision of the regularization parameter is strongly remarked [2]: to find a good LLE mapping; the free parameters must be set. Later, they quoted, if the regularization parameter is set incorrectly the eigenanalysis may not converge. Consequently, another approach for automatic selection of the regularization parameter is discussed, and based in computing the average eigenvalues of the eigenvectors that had not been used in constructing the subspace. Although both approaches previously described have been employed [14, 16, 18], regularization parameter choice remains as an open issue, thus the paper [19] highlights that the mapping quality is quite sensitive to the regularization parameter. As a result, they discussed the

regularization method, demanding another parameter to be set by the user (a desirable determinant), that is, the method does not avoid empirical adjustment, and rather an a priori knowledge of data structure is needed.

Once again discussion on this issue is given in [20], where automatic selection approach, proposed in [2], is refuted, since it requires explicit eigenvalue decomposition for each local Gram matrix, which is computationally expensive. Consequently, another empirical regularization method is given that adds to the Gram matrix a small amount multiplied by identity matrix. But this fixed value is explicitly said to be suitable just for the particular database considered in that work.

We propose an objective way to choose the regularization parameter [21]; this makes possible that low–dimensional representations obtained by means of LLE to be consistent, such that data variations, caused by noise or sample randomness, do not substantially change the embedding results.

When Gram matrix in equation (3.6) is singular or close to singular, that is, $\mathbf{G}$ does not have full rank, for example, when there are more neighbors than input dimensions ($k > p$), the result of least squares problem for finding $\mathbf{W}$ matrix has not unique solution. In other words, (3.9) is no longer suitable to find weights from equation (5.18). Then, it is necessary to regularize $\mathbf{G}$ before finding $\mathbf{W}$. Nevertheless, the regularization parameter plays an important role for calculating a weighted linear combination using the neighbor points. What is more, optimal values for this parameter can vary over a wide range; it depends on particular application, which is partially explained by changes over input data scale [2].

In [3, 13] is proposed to calculate the regularization of $\mathbf{G}$ as

$$\mathbf{G} \leftarrow \mathbf{G} + \alpha_1 \mathbf{I}_{k\times k}, \tag{4.1}$$

where

$$\alpha_1 = \left(\frac{\Delta^2}{k}\right) \mathrm{tr}\left(\mathbf{G}\right),$$

Nevertheless, the parameter Δ must be empirically tuned, namely, it is advised to employ $\Delta = 0.1$ [3].

In [2], it is proposed to find the regularized version of $\mathbf{G}$ as

$$\mathbf{G} \leftarrow \mathbf{G} + \alpha_2 \mathbf{I}_{k\times k}, \tag{4.2}$$

being

$$\alpha_2 = \frac{1}{p-m} \sum_{i=m+1}^{p} \hat{\lambda}_i,$$

the mean of eigenvalues associated to eigenvectors discarded; this can be calculated using an spectral decomposition for each neighborhood by means of PCA.

Regardless, we propose a regularization of the form

$$\mathbf{G} \leftarrow \mathbf{G} + \alpha^2 \mathbf{I}_{k\times k}, \tag{4.3}$$

we propose to find the regularization parameter α in an automated way, taking into account the original formulation of optimization problem, given in (3.4), and the stability properties of the final computed solution.

A vector $\mathbf{w}$ that minimizes the error ε is searched, then it is possible to write (3.7) as

$$\varepsilon_{reg} = \mathbf{w}^\top \mathbf{G}\mathbf{w} + \alpha^2 \mathbf{w}^\top \mathbf{w} \quad \text{s.t.} \quad \sum_{j=1}^{k} w_j = 1. \tag{4.4}$$

The solution of (4.4) by means of Lagrange multipliers is

$$2\mathbf{G}\mathbf{w} + 2\alpha^2 \mathbf{w} = \lambda \mathbf{1} \quad \text{s.t.} \quad \mathbf{1}^\top \mathbf{w} = 1. \tag{4.5}$$

In this way, (4.5) is a linear system with two equations and two variables

$$\begin{aligned} 2\left(\mathbf{G} + \alpha^2 \mathbf{I}\right)\mathbf{w} - \lambda \mathbf{1} &= \mathbf{0} \\ \mathbf{1}^\top \mathbf{w} - \lambda 0 &= 1 \end{aligned} \tag{4.6}$$

that is

$$\begin{bmatrix} 2\left(\mathbf{G} + \alpha^2 \mathbf{I}\right) & \mathbf{1} \\ \mathbf{1}^\top & 0 \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ -\lambda \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix}. \tag{4.7}$$

Before stating the way of computing α, some further analysis of the stability properties of systems of the form (4.6) is needed.

4.1.1 Sensitivity Analysis

In this section, we study how sensitive is the solution (3.9) to the presence of perturbations in the matrix $\mathbf{G}$. For such a purpose, let $\tilde{\mathbf{G}} = \mathbf{G} + \mathbf{E}$ be a noisy version of $\mathbf{G}$. We will assume that $\mathbf{E}$ is small enough for keeping $\tilde{\mathbf{G}}$ non singular. So, working with $\tilde{\mathbf{G}}$ we get the solution

$$\tilde{\lambda} = \frac{2}{\mathbf{1}^\top \tilde{\mathbf{G}}^{-1} \mathbf{1}}, \tilde{\mathbf{w}} = \frac{\tilde{\lambda}}{2} \tilde{\mathbf{G}}^{-1} \mathbf{1}. \tag{4.8}$$

Now, we calculate the distance between the computed and the wanted solution. We begin by noting

$$\begin{aligned} \lambda \mathbf{1} - \tilde{\lambda} \mathbf{1} &= 2\mathbf{G}\mathbf{w} - 2\tilde{\mathbf{G}}\tilde{\mathbf{w}} \\ &= 2\mathbf{G}\mathbf{w} - 2\mathbf{G}\tilde{\mathbf{w}} + 2\mathbf{G}\tilde{\mathbf{w}} - 2\tilde{\mathbf{G}}\tilde{\mathbf{w}} \\ &= 2\mathbf{G}\left(\mathbf{w} - \tilde{\mathbf{w}}\right) + 2\left(\mathbf{G} - \tilde{\mathbf{G}}\right)\tilde{\mathbf{w}} \end{aligned}$$

Also

$$\lambda = \frac{2}{\mathbf{1}^\top \mathbf{G}^{-1} \mathbf{1}} = \frac{2}{\mathbf{1}^\top \tilde{\mathbf{G}}^{-1} \mathbf{1}} \frac{\mathbf{1}^\top \tilde{\mathbf{G}}^{-1} \mathbf{1}}{\mathbf{1}^\top \mathbf{G}^{-1} \mathbf{1}} = \tilde{\lambda} \frac{\mathbf{1}^\top \tilde{\mathbf{G}}^{-1} \mathbf{1}}{\mathbf{1}^\top \mathbf{G}^{-1} \mathbf{1}}.$$

So, we can write

$$\begin{aligned} \left(\lambda - \tilde{\lambda}\right) &= \tilde{\lambda} \frac{\mathbf{1}^\top \left(\tilde{\mathbf{G}}^{-1} - \mathbf{G}^{-1}\right) \mathbf{1}}{\mathbf{1}^\top \mathbf{G}^{-1} \mathbf{1}}, \\ \left(\mathbf{w} - \tilde{\mathbf{w}}\right) &= \mathbf{G}^{-1} \mathbf{E} \tilde{\mathbf{w}} + \frac{\tilde{\lambda}}{2} \frac{\mathbf{1}^\top \left(\tilde{\mathbf{G}}^{-1} - \mathbf{G}^{-1}\right) \mathbf{1}}{\mathbf{1}^\top \mathbf{G}^{-1} \mathbf{1}} \mathbf{G}^{-1} \mathbf{1}. \end{aligned}$$

Finally, we use the following first order approximation

$$\tilde{\mathbf{G}}^{-1} = (\mathbf{G}+\mathbf{E})^{-1} \approx \mathbf{G}^{-1} - \mathbf{G}^{-1}\mathbf{E}\mathbf{G}^{-1},$$

which is valid for small perturbations $\mathbf{E}$ [22]. Then, we can approximate the error on the computing of $\mathbf{w}$ as

$$\begin{aligned} \boldsymbol{\varepsilon} &= \mathbf{G}^{-1}\mathbf{E}\tilde{\mathbf{w}} - \frac{\tilde{\lambda}}{2}\frac{\mathbf{G}^{-1}\mathbf{E}\mathbf{G}^{-1}}{\mathbf{1}^\top\mathbf{G}^{-1}\mathbf{1}}\mathbf{G}^{-1}\mathbf{1} \\ &= \begin{bmatrix} \mathbf{G}^{-1}\mathbf{E} & \frac{1}{2}\frac{\mathbf{1}^\top\mathbf{G}^{-1}\mathbf{E}\mathbf{G}^{-1}\mathbf{1}}{\mathbf{1}^\top\mathbf{G}^{-1}\mathbf{1}}\mathbf{G}^{-1}\mathbf{1} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{w}} \\ -\tilde{\lambda} \end{bmatrix}. \end{aligned} \tag{4.9}$$

So, for avoiding big errors we must compute $\tilde{\mathbf{w}}$ and $\tilde{\lambda}$ by bounding their size.

4.1.2 Regularized Solution

Now, we reconsider the very common situation when the Gram matrix is ill-posed. Maybe for being rank-deficient or for having to close to zero singular values. In this case, some kind of regularization procedure has to be implemented. In this work, we avoid the original problem

$$\mathcal{A}\mathbf{x} = \mathbf{b}, \tag{4.10}$$

where

$$\mathcal{A} = \begin{bmatrix} 2\mathbf{G} & \mathbf{1} \\ \mathbf{1}^\top & 0 \end{bmatrix}, \mathbf{x} = \begin{bmatrix} \mathbf{w} \\ -\lambda \end{bmatrix} \text{ and } \mathbf{b} = \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix}.$$

And we replace it for the regularized version

$$\mathcal{A}_\alpha\mathbf{x} = \mathbf{b}, \tag{4.11}$$

with

$$\mathcal{A}_\alpha = \begin{bmatrix} 2\mathbf{G} + 2\alpha^2\mathbf{I} & \mathbf{1} \\ \mathbf{1}^\top & 0 \end{bmatrix}. \tag{4.12}$$

The unique solution to this last problem is

$$\lambda_\alpha = \frac{2}{\mathbf{1}^\top\left(\mathbf{G}+\alpha^2\mathbf{I}\right)^{-1}\mathbf{1}}, \mathbf{w}_\alpha = \frac{\lambda_\alpha}{2}\left(\mathbf{G}+\alpha^2\mathbf{I}\right)^{-1}\mathbf{1}. \tag{4.13}$$

At this point, two big issues rise up. First, what is the relationship between (4.13) and the optimal solution for least squares of (4.10)?, and second, how can we choose a suitable value for the parameter α?

Relationship with Classical Least Square Solution

From classical theory of Tikhonov regularization theory, it is well known that

$$\left(\mathbf{G}+\alpha^2\mathbf{I}\right)^{-1} \to \mathbf{G}^\dagger \text{ as } \alpha \to 0.$$

Here, $\mathbf{G}^\dagger$ stands for the Penrose's Generalized Inverse of $\mathbf{G}$. So, by guessing $\mathbf{1}^\top\mathbf{G}^\dagger\mathbf{1} \neq 0$,

$$\lambda_\alpha \to \lambda_0 = \frac{2}{\mathbf{1}^\top\mathbf{G}^\dagger\mathbf{1}} \text{ and } \mathbf{w}_\alpha \to \mathbf{w}_0 = \frac{\lambda_0}{2}\mathbf{G}^\dagger\mathbf{1}, \text{ as } \alpha \to 0. \tag{4.14}$$

Now, we denote the classical least square solution of (4.10) for $\mathbf{w}_{ls}$ and λ_{ls}. This solution is characterized for the properties

$$\mathcal{A}^\top\mathcal{A}\mathbf{x}_{ls} = \mathcal{A}^\top\mathbf{b} \text{ and } \mathbf{x}_{ls} \in \mathcal{R}(\mathcal{A}^\top). \tag{4.15}$$

Here, $\mathcal{R}(\mathcal{A}^\top)$ denotes the row space of matrix $\mathcal{A}$. The following lemmas make clear the relation between $\mathbf{x}_{ls}$ and $\mathbf{x}_0 = \begin{bmatrix} \mathbf{w}_0 & -\lambda_0 \end{bmatrix}^\top$.

Lemma 4.1. *If* $\mathbf{1} \notin \mathcal{N}(\mathbf{G})$, *then* $\mathbf{1}^\top\mathbf{G}^\dagger\mathbf{1} \neq 0$.

Proof. Let $\mathbf{G} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$ an orthogonal diagonalization of the symmetric matrix $\mathbf{G}$, and set

$$\mathbf{\Lambda} = \text{diag}(\lambda_1, \lambda_2, \cdots, \lambda_r, 0, \cdots, 0).$$

Here, $\lambda_1, \lambda_2, \cdots, \lambda_r$ are the positive eigenvalues of $\mathbf{G}$ (Gram matrix is positive semidefinite) written in non-increasing way. Then $\mathbf{G}^\dagger = \mathbf{V}\mathbf{\Lambda}^\dagger\mathbf{V}^\top$, where

$$\mathbf{\Lambda}^\dagger = \text{diag}(\lambda_1^{-1}, \lambda_2^{-1}, \cdots, \lambda_r^{-1}, 0, \cdots, 0).$$

By setting $\mathbf{t} = \mathbf{V}^\top\mathbf{1} = \begin{bmatrix} t_1 & t_2 & \cdots & t_k \end{bmatrix}^\top$, then

$$\mathbf{G1} = \mathbf{V}\mathbf{\Lambda}\mathbf{t} = \mathbf{V}\begin{bmatrix} \lambda_1 t_1 & \cdots & \lambda_r t_r & 0 & \cdots & 0 \end{bmatrix}^\top.$$

So, at least one $t_i \neq 0$, for $i = 1, 2, \cdots, r$. Then,

$$\mathbf{1}^\top\mathbf{G}^\dagger\mathbf{1} = \mathbf{t}^\top\mathbf{\Lambda}^\dagger\mathbf{t} = \sum_{i=1}^{r}\lambda_i^{-1}t_i^2 > 0.$$

□

Lemma 4.2. *If* $\mathbf{1} \notin \mathcal{N}(\mathbf{G})$, *then the vectors* $\mathbf{x}_{ls}$ *and* $\mathbf{x}_0$ *satisfy*

1. $\mathbf{x}_0 \in \mathcal{R}(\mathcal{A}^\top)$.
2. $\mathbf{x}_{ls} = \mathbf{x}_0$ *if and only if* $\mathbf{1} \in \mathcal{R}(\mathbf{G})$.

Proof. For proving $\mathbf{x}_0 \in \mathcal{R}(\mathcal{A}^\top)$, we look for a vector $\mathbf{z} = \begin{bmatrix} \mathbf{z}_1 & z_2 \end{bmatrix}^\top$ such that

$$\mathcal{A}^\top\mathbf{z} = \mathbf{x}_0. \tag{4.16}$$

We begin observing that by definition $\mathbf{G}^\dagger\mathbf{1} \in \mathcal{R}(G^\top)$. But $\mathbf{G}$ is symmetric, so $\mathcal{A}$ is symmetric, $\mathbf{w}_0 \in \mathcal{R}(G)$ and $\mathbf{w}_0 = \mathbf{G}\left(\mathbf{G}^\dagger\mathbf{w}_0\right)$. Hence, the system (4.16) states

$$\begin{cases} \mathbf{G}\mathbf{z}_1 + z_2\mathbf{1} & = \mathbf{G}\left(\mathbf{G}^\dagger\mathbf{w}_0\right) \\ \mathbf{1}^\top\mathbf{z}_1 & = -\lambda_0 \end{cases} \tag{4.17}$$

From first equation in (4.17), we need

$$z_2\mathbf{1} = \mathbf{G}\left(\mathbf{G}^\dagger\mathbf{w}_0 - \mathbf{z}_1\right). \tag{4.18}$$

Suppose firstly that $\mathbf{1} \notin \mathcal{R}(\mathbf{G})$, then from (4.18) we need $z_2 = 0$. So, the system forces

$$\mathbf{G}\mathbf{z}_1 = \mathbf{G}\left(\mathbf{G}^\dagger\mathbf{w}_0\right).$$

This is true for any choice of the scalar β, if

$$\mathbf{z}_1 = \mathbf{G}^\dagger\mathbf{w}_0 + \beta\mathbf{1}_N,$$

and $\mathbf{1}_N$ is the ortogonal projection of $\mathbf{1}$ over the null space of $\mathbf{G}$. As $\mathbf{1} \notin \mathcal{R}(\mathbf{G})$, it is necessary $\mathbf{1}_N \neq \mathbf{0}$. Now we apply the second condition

$$\begin{aligned} -\lambda_0 &= \mathbf{1}^\top\mathbf{z}_1 \\ &= \mathbf{1}^\top\mathbf{G}^\dagger\mathbf{w}_0 + \beta\mathbf{1}^\top\mathbf{1}_N \\ &= \mathbf{1}^\top\mathbf{G}^\dagger\mathbf{w}_0 + \beta\mathbf{1}_N^\top\mathbf{1}_N. \end{aligned}$$

So, it is enough taking

$$\beta = \frac{-1}{\|\mathbf{1}_N\|^2}\left(\lambda_0 + \mathbf{1}^\top\mathbf{G}^\dagger\mathbf{w}_0\right).$$

Now, suppose $\mathbf{1} \in \mathcal{R}(\mathbf{G})$. Then $\mathbf{G}\left(\mathbf{G}^\dagger\mathbf{1}\right) = \mathbf{1}$. And the first equation on (4.17) reads

$$\mathbf{0} = \mathbf{G}\left(\mathbf{G}^\dagger\mathbf{w}_0 - \mathbf{z}_1 - z_2\mathbf{G}^\dagger\mathbf{1}\right). \tag{4.19}$$

So, for any choice of z_2, it is enough to do

$$\mathbf{z}_1 = \mathbf{G}^\dagger\mathbf{w}_0 - z_2\mathbf{G}^\dagger\mathbf{1}.$$

But the second equation needs

$$\begin{aligned} -\lambda_0 &= \mathbf{1}^\top\mathbf{z}_1 \\ &= \mathbf{1}^\top\mathbf{G}^\dagger\mathbf{w}_0 - z_2\mathbf{1}^\top\mathbf{G}^\dagger\mathbf{1}. \end{aligned}$$

Note that, as $\mathbf{1}^\top\mathbf{G}^\dagger\mathbf{1} \neq 0$. Then we can take

$$z_2 = \frac{1}{\mathbf{1}^\top\mathbf{G}^\dagger\mathbf{1}}\left(\lambda_0 + \mathbf{1}^\top\mathbf{G}^\dagger\mathbf{w}_0\right).$$

Finally, we compute the residual $\mathcal{A}^\top\mathcal{A}\mathbf{x}_0 - \mathcal{A}^\top\mathbf{b}$,

$$\begin{aligned} \mathcal{A}^\top\mathcal{A}\mathbf{x}_0 - \mathcal{A}^\top\mathbf{b} &= \begin{bmatrix} 4\mathbf{G}^2 + \mathbf{1}\mathbf{1}^\top & 2\mathbf{G}\mathbf{1} \\ 2\mathbf{1}^\top\mathbf{G} & \mathbf{1}^\top\mathbf{1} \end{bmatrix}\begin{bmatrix} \mathbf{w}_0 \\ -\lambda_0 \end{bmatrix} - \begin{bmatrix} \mathbf{1} \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} \left(4\mathbf{G}^2 + \mathbf{1}\mathbf{1}^\top\right)\mathbf{w}_0 - \lambda_0 2\mathbf{G}\mathbf{1} \\ 2\mathbf{1}^\top\mathbf{G}\mathbf{w}_0 - \lambda_0\mathbf{1}^\top\mathbf{1} \end{bmatrix} - \begin{bmatrix} \mathbf{1} \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} 2\lambda_0\mathbf{G}^2\mathbf{G}^\dagger\mathbf{1} + \left(\mathbf{1}^\top\mathbf{w}_0\right)\mathbf{1} - \lambda_0 2\mathbf{G}\mathbf{1} \\ \lambda_0\mathbf{1}^\top\mathbf{G}\mathbf{G}^\dagger\mathbf{1} - \lambda_0\mathbf{1}^\top\mathbf{1} \end{bmatrix} - \begin{bmatrix} \mathbf{1} \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} 2\lambda_0\mathbf{G}^2\mathbf{G}^\dagger\mathbf{1} + \left(\mathbf{1}^\top\mathbf{w}_0\right)\mathbf{1} - \lambda_0 2\mathbf{G}\mathbf{1} \\ \lambda_0\mathbf{1}^\top\mathbf{G}\mathbf{G}^\dagger\mathbf{1} - \lambda_0\mathbf{1}^\top\mathbf{1} \end{bmatrix} - \begin{bmatrix} \mathbf{1} \\ 0 \end{bmatrix} \end{aligned}$$

But, since $\mathbf{1} \in \mathcal{R}(\mathbf{G})$, then $\mathbf{G}\mathbf{G}^{\dagger}\mathbf{1} = \mathbf{1}$, and $\left(\mathbf{1}^{\top}\mathbf{w}_0\right) = 1$ by construction. So,

$$\mathcal{A}^{\top}\mathcal{A}\mathbf{x}_0 - \mathcal{A}^{\top}\mathbf{b} = \begin{bmatrix} \mathbf{0} \\ 0 \end{bmatrix}.$$

□

Remark 1. *The presence of the term* $\mathbf{1}^{\top}\mathbf{G}^{\dagger}\mathbf{1}$ *on the ratio for defining* λ_0 *warns us about problem regarding to the growing of the size of the regularized solution as* $\alpha \to 0$. *Therefore, a proper value for* α *has to be choice. Again the size of the same solution* $\mathbf{x}_{\alpha} = \begin{bmatrix} \mathbf{w}_{\alpha} & -\lambda_{\alpha} \end{bmatrix}^{\top}$ *is recommended.*

4.1.3 Parameter Choice

The analysis of the preceding section warns us about possible complications of the norm of the error due to the growing in the size of the computed solution. So, for implementing this kind of regularization, in an automated way for realistic conditions, special attention has to be paid to the size of the solution. This is a common situation to other regularization procedures [23].

With this in mind, we propose to choose the regularization parameter by

$$\alpha_{opt} = \arg\min_{\alpha} g(\alpha), \tag{4.20}$$

where

$$g(\alpha) = \|\mathbf{x}_{\alpha}\|^2 = \|\mathbf{w}_{\alpha}\|^2 + |\lambda_{\alpha}|^2. \tag{4.21}$$

Now, we explore some properties of this function g as $\alpha \to 0$. We begin by rewriting it as

$$\begin{aligned} g(\alpha) &= \|\mathbf{w}_{\alpha}\|^2 + |\lambda_{\alpha}|^2 \\ &= \frac{|\lambda_{\alpha}|^2}{2}\left\{\left\|\left(\mathbf{G}+\alpha^2\mathbf{I}\right)^{-1}\mathbf{1}\right\|^2 + 2\right\}. \end{aligned}$$

But, following the ideas on Lemma 4.1, we found

$$\left(\mathbf{G}+\alpha^2\mathbf{I}\right)^{-1}\mathbf{1} = \mathbf{V}\begin{bmatrix} \left(\lambda_1+\alpha^2\right)^{-1}t_1 & \cdots & \left(\lambda_r+\alpha^2\right)^{-1}t_r & \alpha^{-2}t_{r+1} & \cdots & \alpha^{-2}t_k \end{bmatrix}^{\top}.$$

$$\left\|\left(\mathbf{G}+\alpha^2\mathbf{I}\right)^{-1}\mathbf{1}\right\|^2 = \sum_{i=1}^{r}\left(\lambda_1+\alpha^2\right)^{-2}t_i^2 + \sum_{i=r+1}^{k}\alpha^{-4}t_i^2, \tag{4.22}$$

where, we use the fact that $\mathbf{V}$ is orthogonal, so it preserves norms.

Similarly,

$$\mathbf{1}^{\top}\left(\mathbf{G}+\alpha^2\mathbf{I}\right)^{-1}\mathbf{1} = \sum_{i=1}^{r}\left(\lambda_1+\alpha^2\right)^{-1}t_i^2 + \sum_{i=r+1}^{k}\left(\alpha^2\right)^{-1}t_i^2. \tag{4.23}$$

At this point, we remember that the entries t_i for $i = r+1, \cdots, k$ are zero when $\mathbf{1} \in \mathcal{R}(\mathbf{G})$. Anyway, the expressions (4.22) and (4.23) are non–increasing as function of α.

Additionally, if some $t_i \neq 0$ with i between $r+1$ and k, both of them go to infinity as $\alpha \to 0$. Finally, as

$$\lambda_\alpha = \frac{2}{\mathbf{1}^\top \left(\mathbf{G}+\alpha^2\mathbf{I}\right)^{-1}\mathbf{1}},$$

then λ_α grows as a function of α.

In summary, the function $g(\alpha)$ is the product of the following functions: the increasing one λ_α and the non–increasing one $\left\|\left(\mathbf{G}+\alpha^2\mathbf{I}\right)^{-1}\mathbf{1}\right\|^2$. Furthermore, if some $t_i \neq 0$ with i between $r+1$ and k, then the value of $g(\alpha)$ could be too high for both small α values and too big α values. This motivates the search of a tradeoff by properly choosing their values (4.20).

4.1.4 Discussion about Regularization Parameter Choice

- The idea behind our technique for choosing automatic regularization parameter is to search a tradeoff between the stability and the accuracy of the solution, because the increase of these parameters leads to augment the error (4.9) in the solution of the cost function (4.4). If we set a high value for α then the value of $|\lambda_\alpha|^2$ will be high, and if we choose a low value for α then the value of $\|\mathbf{w}_\alpha\|^2$ will be high too.

- The empirical regularization method proposed in [3, 13] yields better embedding results than automatic regularization methods, when using the artificial data sets. Nonetheless, the main drawback of heuristic approaches is that each database, used in different contexts, needs its own empirical adjustment, which is far from being easy, particularly when the input space dimensionality is greater than three, because there is not a reference or control about embedding results, that is, a blind embedding.

- The problem with the automatic regularization, proposed in [2], is that, when neighborhoods are unconnected or contain discontinuities, the embedding result turns to be poor. In these cases, residual variance calculated in equation (4.2) can not be proper as regularization value for $\mathbf{G}$.

- Our proposed automatic regularization method is effective for finding out a suitable value of regularization parameter for both artificial and real-world data sets.

4.2 Automatic Choice of the Number of Nearest Neighbors

The crux of the LLE algorithm is a suitable selection of the number of nearest neighbors k, because this parameter allow to establish the underlying structure of the manifold. That is, if k is set too small, the mapping will not reflect any global properties; if it is too high, the mapping will lose its nonlinear character and behave like traditional PCA [2].

In this work we analyze two previous approaches for choosing k. Kouropteva et. al [24] presente a hierarchical method for automatic selection of an optimal parameter value based

on the Minimization of the Residual Variance. Goldberg and Ritov [25] display a novel measure based on Procrustes Rotation that enables quantitative comparison of the output of manifold-based embedding algorithms, this measure also serves as a natural tool for choosing k.

Besides, we propose two new more efficient techniques for automatically choosing the number of nearest neighbors. The first one called Preservation Neighborhood Error (PNE) computes a cost function that quantifies the quality of embedding space [26]. This function takes into account local and global geometry preservation. Nevertheless, just a single value k for the whole manifold is obtained, which may yield misleading embedding results when the intrinsic dimensionality of the neighborhoods varies across the data set. Therefore, we propose another technique, it locally computes a suitable number of neighbors for each sample point in the manifold.

4.2.1 Residual Variance

In [24], the residual variance is employed as a quantitative measure of the embedding results. It is defined as

$$\sigma_R^2(D_{\mathbf{X}}, D_{\mathbf{Y}}) = 1 - \rho^2_{D_{\mathbf{X}} D_{\mathbf{Y}}}, \tag{4.24}$$

where ρ^2 is the standard linear correlation coefficient, taken over all entries of $D_{\mathbf{X}}$ and $D_{\mathbf{Y}}$; $D_{\mathbf{X}}$ and $D_{\mathbf{Y}}$ are the matrices for Euclidean distances in $\mathbf{X}$ and $\mathbf{Y}$, respectively. $D_{\mathbf{Y}}$ depends on the number of neighbors selected k. According to [24], the lowest residual variance corresponds to the best high-dimensional data representation in the embedded space. Hence, the number of neighbors can be computes as

$$k_{\sigma_R^2} = \arg\min_k (\sigma_R^2(D_{\mathbf{X}}, D_{\mathbf{Y}})). \tag{4.25}$$

4.2.2 Procrustes Rotation

Now, in [25] is proposed to compare a neighborhood on the manifold (in $\mathbb{R}^p$) and its embedding (in $\mathbb{R}^m$) using the Procrustes statistic as a measure for qualifying the transformation. Assume that input manifold is dense and isometrically embedded in $\mathbb{R}^m$.

The Procrustes statistic measures the distance between two configurations of points, it computes the sum of squares between pairs of corresponding points after one of the configurations is rotated and translated to best match the other. The measure is defined as

$$P(\mathbf{X}, \mathbf{Y}) = \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{A}\mathbf{y}_i - b\|^2, \tag{4.26}$$

being $\mathbf{A}^\top \mathbf{A} = \mathbf{I}$ and $\mathbf{b} \in \mathbb{R}^m$. The rotation matrix $\mathbf{A}$ can be computed from $\mathbf{Z} = \mathbf{X}^\top \mathbf{H} \mathbf{Y}$, where $\mathbf{H} = \mathbf{I} - \frac{1}{k}\mathbf{1}\mathbf{1}^\top$, $\mathbf{1}$ is a $n \times 1$ column vector, and $\mathbf{H}$ is the centering matrix. Let $\mathbf{U}\mathbf{L}\mathbf{V}^\top$ be the singular-value decomposition of $\mathbf{Z}$, then $\mathbf{A} = \mathbf{U}\mathbf{V}^\top$, the translation vector $\mathbf{b} = \overline{\mathbf{x}} - \mathbf{A}\overline{\mathbf{y}}$, where $\overline{\mathbf{x}}$ and $\overline{\mathbf{y}}$ are the sample means of $\mathbf{X}$ and $\mathbf{Y}$, respectively. Finally, we have

$$P(\mathbf{X}, \mathbf{Y}) = \left\|\mathbf{H}(\mathbf{X} - \mathbf{Y}\mathbf{A}^\top)\right\|_F^2, \tag{4.27}$$

where $\|\cdot\|_F$ is the Frobenius norm.

In order to define how well an embedding preserves the local neighborhoods using the Procrustes statistic $P_L(\mathbf{X}_i, \mathbf{Y}_i)$ of each neighborhood-embedding pair $(\mathbf{X}_i, \mathbf{Y}_i)$. $P_L(\mathbf{X}_i, \mathbf{Y}_i)$ estimates the relation between the entire input neighborhood and its embedding as one entity, instead of comparing angles and distances within the neighborhood with those within its embedding [25]. A global embedding that preserves the local structure can be found by minimizing the sum of the Procrustes statistics of all neighborhood-embedding pairs [25].

Let $\mathbf{X}_i$ be the neighborhood of $\mathbf{x}_i (i = 1, .., n)$ and $\mathbf{Y}_i$ its corresponding mapping, then

$$R(\mathbf{X}, \mathbf{Y}) = \frac{1}{n} \sum_{i=1}^{n} P_L(\mathbf{X}_i, \mathbf{Y}_i). \tag{4.28}$$

The function $R(\mathbf{X}, \mathbf{Y})$ measures the average quality of the neighborhood embeddings. Embedding $\mathbf{Y}$ is considered better than embedding $\tilde{\mathbf{Y}}$ in the local-neighborhood-preserving sense if $R(\mathbf{X}, \mathbf{Y}) < R\left(\mathbf{X}, \tilde{\mathbf{Y}}\right)$. This means that on the average, $\mathbf{Y}$ preserves the structure of the local neighborhoods better than $\tilde{\mathbf{Y}}$. Besides, $R(\mathbf{X}, \mathbf{Y})$ is sensitive to scaling, therefore normalization should be considered. A reasonable normalization is

$$R_N(\mathbf{X}, \mathbf{Y}) = \frac{1}{n} \sum_{i=1}^{n} P_L(\mathbf{X}_i, \mathbf{Y}_i) / \left\| \mathbf{H_L} \mathbf{X}_i \right\|_F^2, \tag{4.29}$$

where $\mathbf{H_L} = \mathbf{I} - \frac{1}{k} \mathbf{1}\mathbf{1}^\top$; $\mathbf{1}$ is a $k \times 1$ vector. Therefore, the number of nearest neighbors can be calculated as

$$k_{R_N} = \arg\min_k (R_N(\mathbf{X}, \mathbf{Y})). \tag{4.30}$$

4.2.3 Preservation Neighborhood Error

We propose an alternative measure for quantifying the embedding quality [26]. This measure attempts to preserve the local geometry and the neighborhood co–location, identifying possible overlaps on the low dimensional space.

We are trying to measure the embedding quality of each neighborhood, such that, nearby points to $\mathbf{x_i}$ in the input space, remain nearby to $\mathbf{y}_i$ in the output space, besides, points outside the high–dimensional neighborhood should not be taken into account as part of the low–dimensional neighborhood. Our criterion is defined as

$$PNE(\mathbf{X}, \mathbf{Y}) = \frac{1}{2n} \sum_{i=1}^{n} \left\{ \frac{1}{k} \sum_{j=1}^{k} \left(D_{(\mathbf{x}_i, \boldsymbol{\eta}_j)} - D_{(\mathbf{y}_i, \boldsymbol{\phi}_j)} \right)^2 + \frac{1}{k_n} \sum_{j=1}^{k_n} \left(D_{(\mathbf{x}_i, \boldsymbol{\theta}_j)} - D_{(\mathbf{y}_i, \boldsymbol{\gamma}_j)} \right)^2 \right\}, \tag{4.31}$$

where D is an standardized Euclidean distance to obtain a maximum value equal to one. For example, $D_{(\mathbf{x}_i, \boldsymbol{\eta}_j)}$ is the distance calculated between the observation $\mathbf{x}_i$ and each one of its k neighbors on the input space.

Once the embedding, for each point $\mathbf{y}_i \in \mathbb{R}^m$ a set $\boldsymbol{\beta}$ of k nearest neighbors is calculated, and the projection $\boldsymbol{\phi}$ of $\boldsymbol{\eta}$ is found. The neighbors computed in $\boldsymbol{\beta}$ that are not neighbors

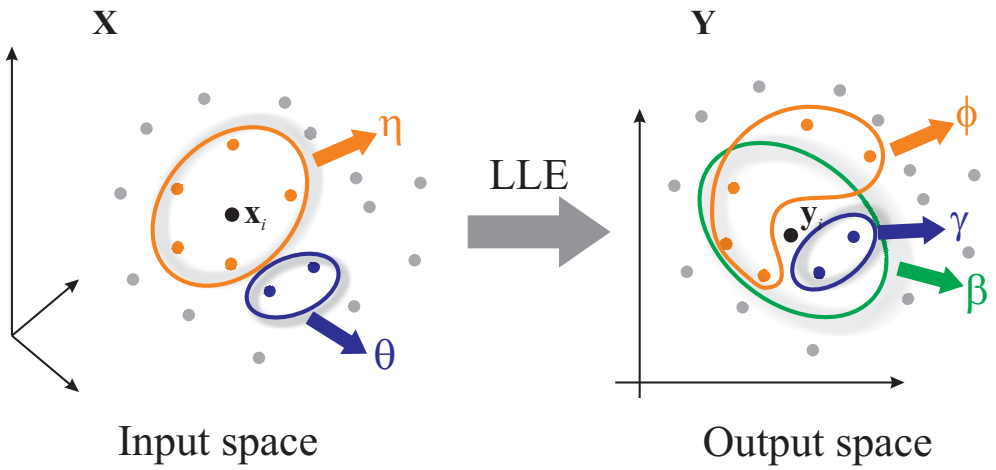


Figure 4.1: Input and output space neighbor sets

in $\boldsymbol{\eta}$ conform a new set $\boldsymbol{\gamma}$, that is, $\boldsymbol{\gamma} = \boldsymbol{\beta} - (\boldsymbol{\beta} \cap \boldsymbol{\phi})$. The size of $\boldsymbol{\gamma}$ is k_n. These sets are shown in Fig. 4.1.

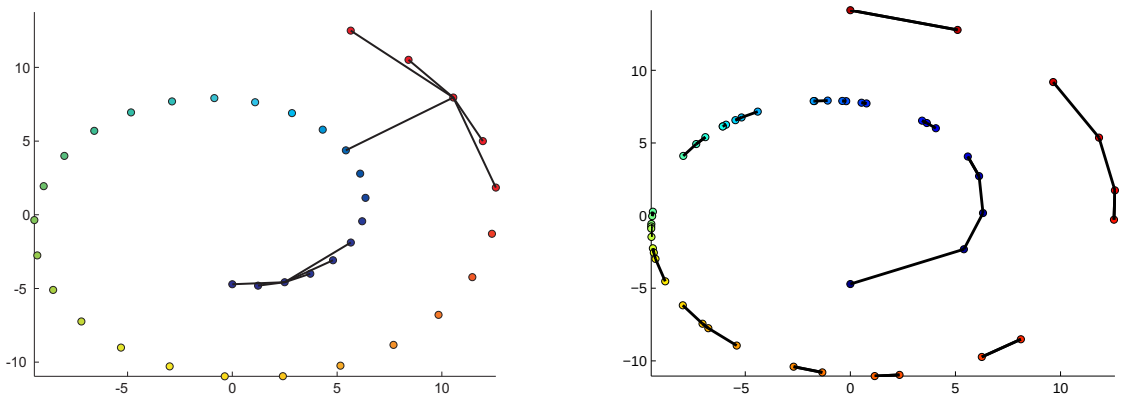
The first term in (4.31) quantifies the local geometry preservation. The k nearest neighbors of $\mathbf{x}_i$ chosen on the high dimensional space $\mathbf{X}$ are compared against their representations in the embedded space $\mathbf{Y}$. The second term computes the error produced by possible overlaps in the embedded results, which frequently occurs when the number of neighbors is strongly increased, and global properties of the manifold are lost. In an ideal embedding $PNE\,(\cdot) = 0$, it means that computed neighbors in the input space $\mathbf{X}$ are the same neighbors in the output space $\mathbf{Y}$. The number of nearest neighbors can be found as

$$k_{PNE} = \arg\min_{k}(PNE(\mathbf{X}, \mathbf{Y})). \tag{4.32}$$

4.2.4 Local Number of Nearest Neighbors

Each point $\mathbf{x}_i$ lies in a neighborhood, which has two main characteristics: density, and intrinsic dimensionality. If these characteristics are very different among neighborhoods, to compute just one global k for the whole manifold could not be right, because it is not possible to ensure that these neighborhoods are locally linear patches, and then, the assumptions of the LLE algorithm are not fulfilled. The nearest neighbors for each sample are measured by Euclidean distance, if k is not correctly set, faraway points are identified as neighbors (short circuits, Figure 4.2(a)), or patches are not well–sampled (Figure 4.2(b)), which leads to unacceptable embedding results.

To address the above outlined problems, we propose a new methodology for finding out a suitable number of nearest neighbors for each point in the manifold. The goal is to represent each observation by computing their nearest neighbors, which are obtained analyzing each neighborhood in $\mathbf{X}$ using Euclidean and geodesic distances. It is important to emphasize that this methodology finds a neighborhood of size k_i for each input $\mathbf{x}_i$. If the region around a point is linear and dense, the Euclidean and geodesic distances should obtain a similar set of nearest neighbors for each $\mathbf{x}_i$, else the neighborhood computed using Euclidean distance will have short circuits while the geodesic distance will be able to

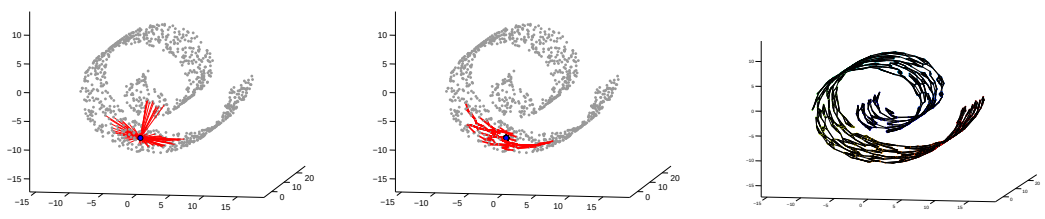


(a) Wrong neighborhood structure, $k = 5$ is too high (b) Low density neighborhood, $k = 1$ is too low

Figure 4.2: k wrongly chosen

identify correctly the neighbors of each sample taking into account the manifold structure. Figures 4.3(a) and 4.3(b) show the neighborhoods of a point according to the Euclidean and geodesic distances, respectively. Euclidean distance exhibits a neighborhood with short circuits while geodesic distance presents a better approximation of the neighborhood.

The idea is to establish a prior set of values k that could be suitable to satisfy the LLE assumptions for all inputs. A lower bound for this set is a value of k that allows to construct a complete graph G over $\mathbf{X}$. The graph G gives an approximation of the manifold structure and it allows to compute the pairwise geodesic distance for all the samples (Figure 4.3(c) presents an example of a complete graph). An upper bound for this set is a value of k computed according to the relation between data density and number of observations. The upper bound pretends to limit the maximum neighborhood size in order to satisfy the local analysis in LLE. Using Euclidean distance, geodesic distance and a suitable set of values for k, it is possible to establish a methodology that computes a number of nearest neighbors k_i for each $\mathbf{x}_i$, finding the largest patch around the point, avoiding short circuits and taking into account the size of the other patches in order to preserve the global structure of the manifold. The procedure for finding out the number of nearest neighbors needed per each point is presented in Algorithm 3.



(a) Euclidean distance (b) Geodesic distance (c) Connected graph

Figure 4.3: Changes in neighborhoods according to the distance

Algorithm 3 – Obtaining k_i in LLE

1: Compute the pairwise Euclidean distance D for all points in $\mathbf{X}$.
2: Construct the minimal connected neighborhood graph G of the given data set $\mathbf{X}$ by the k-nearest neighbors method with k_{min}, initializing $k_{min} = 1$. Check the full connectivity of the graph (by using the breath-first search algorithm). If the graph is not full connected then update $k_{min} = k_{min} + 1$ and start again this step.
3: Compute the geodesic distance D_G over G by using Dijistra's algorithm.
4: Define $k_{max} = n^2/(k_{min}N_E)$, where N_E is the number of edges in G and n the number of samples in $\mathbf{X}$.
5: Set the vector $\mathbf{k}_s = [k_{min} + 1, ..., k_{max}]$, with $\mathbf{k}_s \in \mathbb{R}^b$. The vector $\mathbf{k}_s$ contains the possible values of k for every $\mathbf{x}_i$.
6: For each $\mathbf{x}_i$ define the sets $\boldsymbol{\eta}_D^{(c)}$ and $\boldsymbol{\eta}_{D_G}^{(c)}$, with $c = 1, ..., b$. Each element in $\boldsymbol{\eta}_D^{(c)}$ and $\boldsymbol{\eta}_{D_G}^{(c)}$ corresponds to the $\mathbf{k_{s_c}}$ nearest neighbors $\mathbf{x}_j$ of $\mathbf{x}_i$ ($j = 1, ..., \mathbf{k_{s_c}}$) according to D and D_G respectively.
7: Calculate the linearity conservation matrix $\mathbf{V}$ of size $n \times b$, which analyzes the similarity of the neighborhoods obtained by D and D_G, taking into account the patch size. Each element of $\mathbf{V}$ can be computed as, $\mathbf{V}_{ic} = |\{\boldsymbol{\eta}_D^{(c)} \cap \overline{\boldsymbol{\eta}_{D_G}^{(c)}}\}|/\mathbf{k_{s_c}}$, where $|\cdot|$ calculates the cardinality of a set and $\overline{\{\cdot\}}$ the complement.
8: Initially, for each $\mathbf{x}_i$ define the set $\boldsymbol{k}_o = \emptyset$. Verify the equality $\mathbf{V}_{ic} = \min\{\mathbf{v}_i\}$, where $\mathbf{v}_i$ is a row vector of $\mathbf{V}$ of size $1 \times b$. If the equality is fulfilled update $\boldsymbol{k}_o = \boldsymbol{k}_o \cup \mathbf{k_{s_c}}$.
9: Find the number of nearest neighbors k_i for each $\mathbf{x}_i$ as $k_i = \max\{\boldsymbol{k}_o\}$.
10: Smooth the vector k_i to obtain similar properties in near neighborhoods as, $k_i = (k_i + \mathbf{k}_{\boldsymbol{\eta}}\mathbf{1})\,/\,(k_i + 1)$, where $\mathbf{k}_{\boldsymbol{\eta}}$ is a vector of size $1 \times k_i$, with the sizes of the neighborhoods of each element in $\boldsymbol{\eta}$ (set with the $\mathbf{x}_j$ nearest neighbors of $\mathbf{x}_i$ using Euclidean distance and with $j = 1, .., k_i$), and $\mathbf{1} = \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}^\top$ of size $k_i \times 1$.
11: Store all the values k_i, previously calculated, in a vector $\mathbf{k}$ of size $1 \times n$.
12: Remove the outliers of $\mathbf{k}$ (see [27]). The outliers of $\mathbf{k}$ are replaced by the average of the elements of $\mathbf{k}$, which were not identified as outliers.
13: Each element of $\mathbf{k}$ contains the number of nearest neighbors k_i for each $\mathbf{x}_i$.

4.2.5 Discussion about the Automatic Choice of the Number of Nearest Neighbors

– The residual variance measure (4.25) does not quantify the local geometric structure of the manifold, then it can not identify a suitable number of neighbors for LLE. One possible for this deficient behavior could be attributed to the fact that Euclidean distance becomes an unreliable indicator for proximity.

– The local procrustes measure (4.30) preserves local geometric structure but does not consider the global behavior of the manifold. For this reason, far neighborhoods can be overlapped in the low-dimensional space and this measure shall not detect it.

– The proposed approach called *Preservation Neighborhood Error* preserves the local geometry of data and the global behavior of the manifold. The first term in (4.31)

quantifies the local geometry preservation, while the second one computes the error produced by possible overlaps in the embedded results, which frequently occurs when the number of neighbors is strongly increased, and global properties of the manifold are lost.

- Preservation neighborhood error also can be used as a measure for quantifying the embedding quality for LLE. Where an ideal embedding corresponds to $PNE(\Delta) = 0$.

- The second proposed approach computes an specific number of neighbors for each point in the manifold, taking into account the linearity and density of each patch. This methodology calculates each k_i preserving the linearity of every patch, taking into account the density of each region and the neighborhoods size of the other regions in the manifold, in order to retain both local geometry and global structure.

- Global optimization techniques, such as a residual variance, Procrustes rotation, and preservation neighborhood error, need to compute an embedding for each possible value of k, solving the eigenvalue problem in LLE many times, resulting in a large computational cost process. The local optimization methodology solves just once the eigenvalue problem, reducing greatly the computational cost compared with global optimization techniques.

- The local selection of the number of neighbors outperforms global selection techniques, specially when intrinsic properties of the neighborhoods change throughout the manifold. In these cases, the size of neighborhoods lying in linear patches is not regular. Then, the local algorithm proposed can adapt to this kind of fluctuations.

- Nevertheless, local criterion requires special attention when it is used on manifolds with a large amount of outliers. In this sense, the local cost function can overfit the number of neighbors, producing an unconnected manifold, thus the embedding results can be inconsistent. In this case it is preferable to use the proposed algorithm for global selection.

4.3 Intrinsic Dimensionality

Once again, we remark that mapping quality is quite sensitive to free parameters of the LLE algorithm. Particularly, if the intrinsic dimensionality m is set too high, the mapping will enhance noise (due to the constraint that removes the scale as a degree of freedom $\sum_{i=1}^{n} \mathbf{y}_i \mathbf{y}_i^\top / n = \mathbf{I}_{m \times m}$), if m is set too low, distinct parts of the data set might be mapped on top of each other [2]. In previous works [2, 3, 8] this problem have been referred.

In this sense, a first strategy [8] consists of estimating m by the number of eigenvalues comparable in magnitude to the smallest nonzero eigenvalues of the cost matrix $\mathbf{M}$ from equation (3.12). A second strategy [2, 3] suggest to estimate the intrinsic dimensionality m by examining the eigenvalue spectra of local covariance matrices $\mathbf{G}$ computed using (3.6). In practice, one often specifies an amount ν of variance that should be retained by

projecting the data set, such that

$$\nu \leq \frac{\sum_{j=1}^{m} \varsigma_j}{\sum_{j=1}^{p} \varsigma_j} \tag{4.33}$$

where ς_j are the eigenvalues of the matrix $\mathbf{G}$ sorted in descend order. Then, for a specified ν, one can calculate a minimal m.

On the other hand, we propose an approach based on self-similarity concepts [28] for figuring out the dimensionality of the underlying structure. Particularly, we redefine the correlation dimension, which is a tool to quantify self-similarity when it is known to be presented, in terms of samples of the manifold. First, let $\mathcal{C}$ be the correlation sum

$$\mathcal{C}(\varepsilon) = \frac{2}{n(n-1)} \sum_{i=1}^{n} \sum_{j=1}^{n} \Theta(\varepsilon - \|\mathbf{x}_i - \mathbf{x}_j\|), \tag{4.34}$$

where $\Theta(\cdot)$ is the Heaviside step function

$$\Theta(z) = \begin{cases} 0, & \text{if } z \leq 0 \\ 1, & \text{if } z > 0 \end{cases} \tag{4.35}$$

The correlation sum just counts the pairs $(\mathbf{x}_i, \mathbf{x}_j)$ whose distance is smaller than ε. In the limit $n \to \infty$ and for small ε, we expect $\mathcal{C}$ to scale like a power law

$$\mathcal{C}(\varepsilon) \propto \varepsilon^{\mathcal{D}}. \tag{4.36}$$

Then, $\mathcal{D}$ is called the correlation dimension, which can be found as

$$\mathcal{D} = \lim_{\varepsilon \to 0} \lim_{n \to \infty} d(n, \varepsilon), \tag{4.37}$$

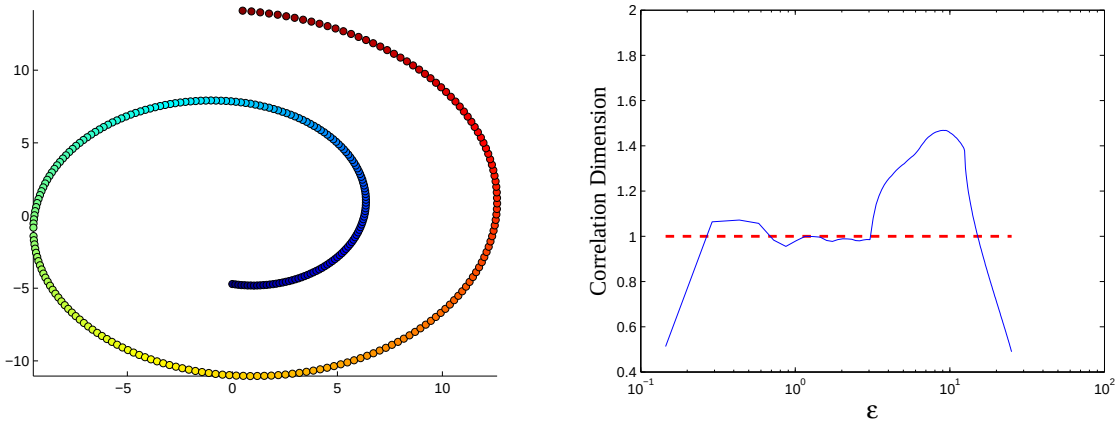
where

$$d(n, \varepsilon) = \frac{\partial \ln \mathcal{C}(n, \varepsilon)}{\partial \ln \varepsilon} \tag{4.38}$$

The straight forward estimator (4.34) turns out to be biased towards too small dimensions when the pairs entering the sum are not statistically independent. If you plot $\mathcal{C}(\varepsilon)$ versus ε, it is possible to find something like a power law for $\mathcal{C}(\varepsilon)$. Nevertheless, the power law behavior of $\mathcal{C}(\varepsilon)$ as the signature of self-similarity can best be found by plotting the slope $d(\varepsilon)$ of a doble logarithmic plot of $\mathcal{C}(\varepsilon)$ versus ε, the latter still on a logarithmic scale. This representation shows much more clearly the plateau of (4.38), which corresponds to the desired power law for $\mathcal{C}$, that is, the plateau denotes the correlation dimension, such that $m = \lceil \mathcal{D} \rceil$. For practical applications we can consider

$$d(\varepsilon) = \frac{\ln(\mathcal{C}(2\varepsilon)) - \ln\left(\mathcal{C}\left(\frac{\varepsilon}{2}\right)\right)}{\ln(2\varepsilon) - \ln\left(\frac{\varepsilon}{2}\right)} = \frac{1}{2} \log_2 \frac{\mathcal{C}(2\varepsilon)}{\mathcal{C}\left(\frac{\varepsilon}{2}\right)} \tag{4.39}$$

In Figure 4.4 it is presented an example of the correlation dimension found for a two dimensional roll. The plateau, in Figure 4.4(b), determines the correlation dimension, and then the intrinsical dimensionality of the manifold.



(a) Roll, $p = 2$, $n = 315$

(b) Correlation Dimension, $m = \lceil \mathcal{D} \rceil = 1$

Figure 4.4: Intrinsical dimensionality using correlation dimension

4.3.1 Discussion about Intrinsic Dimensionality

- The first strategy above mentioned (number of the smallest nonzero eigenvalues) works only for contrived examples, such as data that lies on an essentially linear manifold, or data that has been sampled in an specially uniform way, so that the lowest nonzero eigenvalues are equal or nearly equal due to symmetry. In more general cases, this technique could not be reliable.

- The problem in the second proposal is to estimate just one value for the intrinsic dimensionality. It is posible to think about to compute the expected intrinsic dimensionality of the whole data set as the mean, median, mode (or another estimator) of the intrinsic dimensionality computed from local covariance matrices. When the local dimensionality is very changing. Which kind of estimator is consistent?

- From (4.37), the two limits we have to take will get us into trouble whenever we have a finite sample instead of a full distribution: n is limited by the sample size, and the range of the meaningful choices for ε is limited by the finite accuracy of the data and the inevitable lack of near neighbors at small length scales.

- The correlation dimension concept is less suited and has to be used with caution if self-similarity still has not be established, that is, if it is uncertain if the data are low dimensional deterministic data at all.

- Correlation dimension leads to serious underestimation unless the necessary care is taken.

Chapter 5

A New Proposal for Locally Linear Embedding

Dimensionality reduction and other pattern recognition tasks employ *static features*, that is, scalar representations of some particular attribute, which are assumed constant with respect to some associated dimension or dimensions, for example: time, space, etc. Nevertheless, we can also represent objects by means of measures that do change over some associated dimension (*dynamic features*). Thus, instead of representing observations as single vectors whose entries are constant features, we may turn these entries into arrays that properly introduce the dynamic relation. Commonly, dynamic features are called curves, stochastic processes, or functional data [29].

In particular for biological systems, changes of its dynamics are usually connected with changes in the observed status, which for biomedical signals means changes in patient health [30]. A characterization of the information flow of the biological process underlying a given set of measured biomedical time series can lead to the detection of changes as alarm marks for episodes at risk and to the classification of pathologies based on the corresponding hidden dynamics, when other methods based on more traditional signal features fail [30].

Previous works [31–33] that employ functional data for pattern classification are limited because they constrain the analysis to the linear case, particularly, to PCA, which can not correctly discover underlying structures that lie on a nonlinear manifold.

This limiting representation is overcome in [4], where Locally Linear Embedding (LLE) technique is presented, which allows to find out underlying data structures from nonlinear manifolds, because LLE assumes data lie on manifolds locally behave in a linear way. Although LLE have shown to be an appropriate technique for NLDR, specially in visualization, it has some limitations when data proceed from different manifolds or when data is divided into separated groups [8], which are common cases in pattern recognition. Besides, LLE does not consider class label information, which can be helpful for improving data representation on this kind of data.

It is well known that when the LLE algorithm is used for finding out an underlying structure, it is necessary to ensure that chosen neighborhoods appropriately represent the manifold, these neighborhoods actually must be well-sampled and lying in locally linear patches. The suitable selection of the neighborhoods fundamentally depends on the

distance measure employed inside the algorithm and its associated parameters.

Particularly, functional data that are coming from biological and industrial systems are usually corrupted by artifacts or missing values, these observations are considered as outliers for the LLE algorithm when employing Euclidean distance, producing low-density and unconnected neighborhoods, which distorts the relations among neighbors and leads to unappropriated embeddings. Actually, the problem with using Euclidean distance is that artifacts in the observations get much importance and missing values are intractable. This reason lead us to think about a measure for comparing observations that would recover the relevant information, and at the same time it would not be disturbed by artifacts or missing values.

Some recent publications describe efforts to extract relevant information from functional data, they propose the use of informative cost functions as similarity measures for improving the performance of learning systems [9]. We are particularly interested in [34] where a new generalized correlation measure (correntropy) is developed, from an information theoretic learning (ITL) point of view, this measure quantifies the shape and size of the group of points in feature space. Besides, in [35] the probabilistic and geometric meaning of the correntropy function is established. The correntropy function can be understood as a localized similarity measure, it is directly related to the probability of how similar two random variables are in a neighborhood of the joint space controlled by a kernel bandwidth. Moreover, correntropy induces a metric which is equivalent to the 2-norm if points are close, outside of this zone it behaves similarly to 1-norm, eventually, as two points are further apart, the metric saturates and becomes insensitive to distance approaching to a 0-norm.

Inspired by information-theoretic learning (ITL) [9], we are interested in exploiting the advantages of the correntropy function as a similarity measure that can consider the geometrical and the statistical behavior of the data, and specially its particular metric that allows it to conform to suitable neighborhoods avoiding outliers and distortions. We propose a new technique for NLDR called correntropy locally linear embedding (Correntropy-LLE), that improves the performance of LLE, when functional data are employed. This kind of embedding makes further analysis as visualization or classification easier.

On the other hand, we extend the LLE approach for NLDR to deal with several manifolds (classes) employing class labels as extra information for guiding the embedding, this allows to figure out a suitable representation in the low dimensional space. That is, we are trying to find a low dimensional output conformed by underlying variables from the high dimensional input data, which are closely related to the class label information.

5.1 Correntropy Locally Linear Embedding

As mentioned above, the problem with traditional NLDR schemes like LLE is that they traditionally employ the Euclidean distance measure to determine neighbors within local patches on the manifold. While the Euclidean distance measure is appropriate for measuring the distance between objects characterized by static features, it is less appropriate for measuring the distance between functional data. Non-Euclidean distance measures such as mutual information, entropy correlation coefficient, and the relative entropy have been

shown to be more appropriate for measuring the similarity between signals compared to the L2 norm [7]. We propose to use a generalized correlation function called cross-correntropy [35] (case of two arbitrary random variables), which contains higher-order moments of the probability density function (pdf), but it is much simpler to estimate directly from samples than conventional moment expansions.

Considering a functional data representation, let $\mathbf{X}$ be the input data matrix of size $n \times L$, where the given object $\{x_i[l],\ l = 1, \ldots, L\}$ is an univariate stochastic process of L samples, and $i = 1, \ldots, n$ indexes the objects (input data). Similarly as in the LLE algorithm, observations can be approximated as combinations of their nearest neighbors and then be mapped to a lower dimensional space $(m, m \leq L)$ which preserves data local geometry.

At the beginning, it is necessary to find the k nearest neighbors of each observation, as measured by cross-correntropy $\mathcal{V}_\sigma$. As it was stated cross-correntropy is a generalized similarity measure between two arbitrary scalar random variables X_i and X_j controlled by a kernel bandwidth [35], so that

$$\mathcal{V}_\sigma(X_i, X_j) = \mathbf{E}\left[\kappa_\sigma(X_i - X_j)\right]. \tag{5.1}$$

This kernel bandwidth acts as a zoom lens, controlling the observation window in which similarity is assessed, providing an effective mechanism to draw aside artifacts, and then, to avoid distortions in the estimation of the neighborhoods. In practice, the joint pdf is unknown and only a finite number of samples $\{\mathbf{x}_i, \mathbf{x}_j | x_i[l], x_j[l]\}_{l=1}^{L}$ are available, leading to the sample estimator of correntropy

$$\hat{\mathcal{V}}_{n,\sigma}(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{L}\sum_{l=1}^{L}\kappa_\sigma(x_i[l] - x_j[l]). \tag{5.2}$$

Employing a Gaussian kernel, it is possible rewrite (5.2) as

$$\hat{\mathcal{V}}_{n,\sigma}(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{L}\sum_{l=1}^{L}\frac{1}{\sigma\sqrt{2\pi}}\exp\left(\frac{-(x_i[l] - x_j[l])^2}{2\sigma^2}\right), \tag{5.3}$$

where σ is the kernel bandwidth for a Gaussian kernel, which is a free parameter that must be chosen by the user using concepts of density estimation, such as Silverman's rule

$$\sigma = 0.9AL^{-1/5} \text{ where } A = \min\left\{\hat{\sigma}, \frac{R}{1.34}\right\}, \tag{5.4}$$

where $\hat{\sigma}$ is the empirical standard deviation of data and R is the data interquartile range.

Once the neighborhoods are established, each object is represented as a weighted combination of its neighbors, in this case, we calculate weights $\mathbf{W}$ that maximize the cost function

$$J(W) = \sum_{i=1}^{n}\frac{1}{L}\sum_{l=1}^{L}\kappa_\sigma\left(x_i[l] - \sum_{j=1}^{n} w_{ij}x_j[l]\right), \tag{5.5}$$

subject to a sparseness constraint $w_{ij} = 0$ if $\mathbf{x}_j$ is not one of the k-nearest neighbors of $\mathbf{x}_i$, and an invariance constraint $\sum_{j=1}^{n} w_{ij} = 1$. Now, considering a particular object $\{\mathbf{x}|x[l], l = 1, \ldots, L\}$, and let $\mathbf{V}$ be a matrix of size $k \times L$

$$\mathbf{V} = \begin{bmatrix} v_1[1] & v_1[2] & \cdots & v_1[l] & \cdots & v_1[L] \\ v_2[1] & v_2[2] & \cdots & v_2[l] & \cdots & v_2[L] \\ \vdots & \vdots & & \vdots & & \vdots \\ v_j[1] & v_j[2] & \cdots & v_j[l] & \cdots & v_j[L] \\ \vdots & \vdots & & \vdots & & \vdots \\ v_k[1] & v_k[2] & \cdots & v_k[l] & \cdots & v_k[L] \end{bmatrix}, \tag{5.6}$$

which contains the k-nearest neighbors of $\mathbf{x}$, where each row corresponds to a neighbor object. Using column vectors for writing $\mathbf{V}$

$$\mathbf{V} = \begin{bmatrix} \mathbf{v}[1] & \mathbf{v}[2] & \cdots & \mathbf{v}[l] & \cdots & \mathbf{v}[L] \end{bmatrix}, \tag{5.7}$$

where

$$\mathbf{v}[l] = \begin{bmatrix} v_1[l] & v_2[l] & \cdots & v_j[l] & \cdots & v_k[l] \end{bmatrix}^\top. \tag{5.8}$$

Let be $\mathbf{w}$ a column vector of size $k \times 1$, which contains the representation weights of the data $\mathbf{x}$ as a function of its k nearest neighbors $\mathbf{V}$, then we wish

$$\max_{\mathbf{w}} J(\mathbf{w}) = \max_{\mathbf{w}} \left(\frac{1}{L} \sum_{l=1}^{L} \kappa_\sigma \left(x[l] - \mathbf{w}^\top \mathbf{v}[l] \right) \right) \text{ s.t. } \mathbf{1}^\top \mathbf{w} = 1, \tag{5.9}$$

where $\mathbf{1}^\top = \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}$. Using a Gaussian kernel, it is possible to rewrite (5.9) as

$$\max_{\mathbf{w}} J(\mathbf{w}) = \max_{\mathbf{w}} \left(\frac{1}{L} \sum_{l=1}^{L} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-\left(x[l] - \mathbf{w}^\top \mathbf{v}[l] \right)^2}{2\sigma^2} \right) \right) \text{ s.t. } \mathbf{1}^\top \mathbf{w} = 1. \tag{5.10}$$

In order to optimize (5.10) it is possible to employ a Sequential Quadratic Optimization (SQO) strategy, in particular we use the Lagrange-Newton method [36]. In this case, to maximize $J(\mathbf{w})$ in (5.10) is equivalent to minimize $f(\mathbf{w})$ being

$$f(\mathbf{w}) = -\frac{1}{L} \sum_{l=1}^{L} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-\left(x[l] - \mathbf{w}^\top \mathbf{v}[l] \right)^2}{2\sigma^2} \right) \tag{5.11}$$

then

$$\min_{\mathbf{w}} f(\mathbf{w}) = \min_{\mathbf{w}} \left(-\frac{1}{L} \sum_{l=1}^{L} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-\left(x[l] - \mathbf{w}^\top \mathbf{v}[l] \right)^2}{2\sigma^2} \right) \right) \text{ s.t. } \mathbf{1}^\top \mathbf{w} = 1. \tag{5.12}$$

It is necessary to solve the problem via a series of approximations of the form

$$f(\mathbf{w} + \boldsymbol{\delta}) \simeq q(\boldsymbol{\delta}) = \frac{1}{2} \boldsymbol{\delta}^\top \Omega \boldsymbol{\delta} + f'(\mathbf{w})^\top \boldsymbol{\delta} + f(\mathbf{w}), \tag{5.13}$$

subject to

$$c\left(w+\boldsymbol{\delta}\right) \simeq r\left(\boldsymbol{\delta}\right) = \mathbf{J}_{c1}\left(\mathbf{w}\right)^{\top}\boldsymbol{\delta} + c_1\left(\mathbf{w}\right), \tag{5.14}$$

where $c_1\left(\mathbf{w}\right)$ is the equality constraint in (5.12) and $\mathbf{J}_{c1}\left(\mathbf{w}\right)$ is the Jacobian of the constraint

$$\left(\mathbf{J}_{c1}\right)_j = \frac{\partial c_1}{\partial w_j}. \tag{5.15}$$

The method is iterative and each step includes the solution of a general quadratic optimization problem, which allows to update the parameter $\mathbf{w}$ in each iteration. First it is necessary to calculate

$$\mathbf{h} = \arg\min_{\boldsymbol{\delta}} \left\{q\left(\boldsymbol{\delta}\right)\right\}, \tag{5.16}$$

It is possible to reformulate (5.16) as a nonlinear equation system, such that $(\boldsymbol{\delta}, \lambda)$ can be found

$$L'\left(\boldsymbol{\delta}, \lambda\right) = 0, \tag{5.17}$$

being

$$L\left(\boldsymbol{\delta}, \lambda\right) = \frac{1}{2}\boldsymbol{\delta}^{\top}\Omega\boldsymbol{\delta} + f'\left(\mathbf{w}\right)^{\top}\boldsymbol{\delta} + f\left(\mathbf{w}\right) - \lambda c_1'\left(\mathbf{w}\right)\boldsymbol{\delta} - \lambda c_1\left(\mathbf{w}\right). \tag{5.18}$$

Then, we have

$$\begin{aligned} L'_{\delta}\left(\boldsymbol{\delta}, \lambda\right) &= \Omega^{\top}\boldsymbol{\delta} + \nabla f\left(\mathbf{w}\right) - \lambda\nabla c_1\left(\mathbf{w}\right) \\ L'_{\lambda}\left(\boldsymbol{\delta}, \lambda\right) &= -\nabla c_1\left(\mathbf{w}\right)^{\top}\boldsymbol{\delta} - c_1\left(\mathbf{w}\right), \end{aligned} \tag{5.19}$$

and equating to zero

$$\begin{bmatrix} \Omega^{\top} & -\nabla c_1\left(\mathbf{w}\right) \\ -\nabla c_1\left(\mathbf{w}\right)^{\top} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\delta} \\ \lambda \end{bmatrix} = \begin{bmatrix} -\nabla f\left(\mathbf{w}\right) \\ c_1\left(\mathbf{w}\right) \end{bmatrix}, \tag{5.20}$$

where we have initial values for $\mathbf{w}$ and Ω, besides

$$c_1\left(\mathbf{w}\right) = \mathbf{1}^{\top}\mathbf{w} - 1, \tag{5.21}$$

and

$$\begin{aligned} \nabla f\left(\mathbf{w}\right) &= \begin{bmatrix} \frac{\partial f(\mathbf{w})}{\partial w_1} & \frac{\partial f(\mathbf{w})}{\partial w_2} & \cdots & \frac{\partial f(\mathbf{w})}{\partial w_k} \end{bmatrix}^{\top} \\ \nabla c_1\left(\mathbf{w}\right) &= \begin{bmatrix} \frac{\partial c_1(\mathbf{w})}{\partial w_1} & \frac{\partial c_1(\mathbf{w})}{\partial w_2} & \cdots & \frac{\partial c_1(\mathbf{w})}{\partial w_k} \end{bmatrix}^{\top}. \end{aligned} \tag{5.22}$$

In this way, from the equation (5.20) $\boldsymbol{\delta}$ and λ are found, and $\mathbf{h} = \boldsymbol{\delta}$, therefore the next approximation for $\mathbf{w}$ can be found as

$$\mathbf{w}_{\alpha} := \mathbf{w} + \alpha\mathbf{h}. \tag{5.23}$$

It is necessary to compute the step parameter α, which requires another iterative algorithm. Initially $\alpha = 1$, $\mu = 0$ and later redefined as

$$\mu = \max\left\{\left|\lambda\right|, \frac{1}{2}\left(\mu + \left|\lambda\right|\right)\right\}. \tag{5.24}$$

Now calculate

$$\begin{aligned} \pi(\alpha) &= f(\mathbf{w}_\alpha) + \mu \left|\mathbf{1}^\top \mathbf{w}_\alpha - 1\right| \\ \pi(0) &= f(\mathbf{w}) + \mu \left|\mathbf{1}^\top \mathbf{w} - 1\right| \end{aligned} \tag{5.25}$$

$$\Delta = h^\top \nabla f(\mathbf{w}) - \mu \left|\mathbf{1}^\top \mathbf{w} - 1\right|, \tag{5.26}$$

$$\beta = \frac{-\Delta\alpha^2}{2(\pi(\alpha) - \pi(0) - \Delta\alpha)}, \tag{5.27}$$

and finally

$$\alpha := \min\{0.9\alpha, \max\{\beta, 0.1\alpha\}\}. \tag{5.28}$$

This process is repeated while

$$\pi(\alpha) \geq \pi(0) + 0.1\Delta\alpha \tag{5.29}$$

On the other hand, the algorithm for updating $\mathbf{w}$ requires in each iteration to recompute the value of Ω. For this purpose is necessary calculate

$$\mathbf{r} = \nabla f(\mathbf{w}_\alpha) - \nabla f(\mathbf{w}) - (\nabla c_1(\mathbf{w}_\alpha) - \nabla c_1(\mathbf{w}))\lambda, \tag{5.30}$$

and then to verify the curvature condition

$$\mathbf{r}^\top (\mathbf{w}_\alpha - \mathbf{w}) > 0, \tag{5.31}$$

if this is satisfied, then Ω is positive definite with respect to the step direction $\mathbf{h}$, and so is Ω_{new} found by

$$\Omega_{new} = \Omega + \frac{1}{\alpha \mathbf{h}^\top \mathbf{r}} \mathbf{r}\mathbf{r}^\top - \frac{1}{\mathbf{h}^\top \mathbf{u}} \mathbf{u}\mathbf{u}^\top \text{ where } \mathbf{u} = \Omega\mathbf{h}, \tag{5.32}$$

if the curvature condition is not satisfied, then we let

$$\Omega_{new} = \Omega. \tag{5.33}$$

Finally, the stopping criterion is a measure of goodness of the approximate solution

$$\eta(\mathbf{w}, \lambda) = |q(\alpha\mathbf{h}) - f(\mathbf{w})| + \lambda |c_1(\mathbf{w})| < \varepsilon, \tag{5.34}$$

where

$$q(\alpha\mathbf{h}) = f(\mathbf{w}) + \alpha\mathbf{h}^\top \nabla f(\mathbf{w}) + \frac{1}{2}\alpha^2 \mathbf{h}^\top \Omega \mathbf{h}. \tag{5.35}$$

The algorithm for compute the weights $\mathbf{w}$ is summarized in Algorithm 4.

Once $\mathbf{w}$ weights for all the input objects are calculated, a matrix $\mathbf{W}$ of size $n \times n$ is obtained and it is possible to find the data embedding to the low dimensional space. The low dimensional output $\mathbf{Y}$ can be found by minimizing (3.10), which is done in the same way as in the third step of the LLE algorithm (section 3.2).

Algorithm 4 – Lagrange-Newton Method for compute $\mathbf{w}$ weights

Require: Choose $\mathbf{w}_0$, i_{max}, ε. Set $\Omega = \mathbf{I}$ and $\mu = 0$

$\mathbf{w} := \mathbf{w}_0$
repeat
  $i = i + 1$
  Compute $\boldsymbol{\delta}$ and λ by (5.20)
  $\mathbf{h} = \boldsymbol{\delta}$
  Set $\alpha = 1$
  Update μ by (5.24)
  Compute Δ by (5.26)
  while $\pi(\alpha) \geq \pi(0) + 0.1\Delta\alpha$ **do**
    Compute β by (5.27)
    Compute α by (5.28)
  end while
  Update $\mathbf{w}$ by (5.23)
  Compute $\mathbf{r}$ by (5.30)
  if $\mathbf{r}^\top(\mathbf{w}_\alpha - \mathbf{w}) > 0$ **then**
    Compute Ω_{new} by (5.32)
  else
    Compute Ω_{new} by (5.33)
  end if
  Compute $\eta(\mathbf{w}, \lambda)$ by (5.34)
until $\eta(\mathbf{w}, \lambda) < \varepsilon$ **or** $i > i_{max}$

5.1.1 Discussion about Correntropy-LLE

- The kernel bandwidth controls how similar two objects are in a neighborhood, this property of the correntropy provides an appropriate way to remove the effects of the non-gaussian noise on the unfolded representation. Nonetheless, to adjust this parameter is necessary to employ density estimation techniques such as Silverman's rule, but because suitable unfolded structures are not always obtained, automatic estimation of the kernel bandwidth remains an open problem.

- When the correntropy similarity measure is employed instead of Euclidean distance in the LLE algorithm, the unfolded structure describes more clearly the underlying processes behind the input data. Besides, the low dimensional representations are smooth, without distortions, and generally, the neighborhoods are conformed for very similar objects

- The algorithm takes advantage of the metric induced by the correntropy similarity measure, which reduces the effects of artifacts, distortions and non-gaussian present in the objects.

- This new proposal allows to deal with missing values. It is possible to replace missing values by very high values, that is, one could change missing values by the mean of