Chapter 5
Nonlinear Dimension Reduction and Diffusion Maps

Nonlinear dimension reduction: Diffusion maps, manifold learning, intrinsic geometry of data

5.0.7 Diffusion Maps

Diffusion Maps will allows us to represent (weighted) graphs $G = (V, E, W)$ in $\mathbb{R}^d$, i.e. associating, to each node, a point in $\mathbb{R}^d$. As we will see below, oftentimes when we have a set of data points $x_1, \ldots, x_n \in \mathbb{R}^p$ it will be beneficial to first associate to each a graph and then use Diffusion Maps to represent the points in d-dimensions, rather than using something like Principal Component Analysis.

Before presenting Diffusion Maps, we'll introduce a few important notions. Given $G = (V, E, W)$ we consider a random walk (with independent steps) on the vertices of V with transition probabilities:

$$\text{Prob}\{X(t+1) = j | X(t) = i\} = \frac{w_{ij}}{\deg(i)},$$

where $\deg(i) = \sum_j w_{ij}$. Let M be the matrix of these probabilities,

$$M_{ij} = \frac{w_{ij}}{\deg(i)}.$$

It is easy to see that $M_{ij} \geq 0$ and $M\mathbf{1} = \mathbf{1}$ (indeed, M is a transition probability matrix). Defining D as the diagonal matrix with diagonal entries $D_{ii} = \deg(i)$ we have

$$M = D^{-1}W.$$

If we start a random walker at node i ($X(0) = 1$) then the probability that, at step t, is at node j is given by

$$\text{Prob}\{X(t) = j | X(0) = i\} = \left(M^t\right)_{ij}.$$

In other words, the probability cloud of the random walker at point t, given that it started at node i is given by the row vector

$$\text{Prob}\{X(t)|X(0)=i\} = e_i^T M^t = M^t[i,:].$$

Remark 5.1. A natural representation of the graph would be to associate each vertex to the probability cloud above, meaning

$$i \to M^t[i,:].$$

This would place nodes i_1 and i_2 for which the random walkers starting at i_1 and i_2 have, after t steps, very similar distribution of locations. However, this would require $d = n$. In what follows we will construct a similar mapping but for considerably smaller d.

M is not symmetric, but a matrix similar to M, $S = D^{\frac{1}{2}} M D^{-\frac{1}{2}}$ is, indeed $S = D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$. We consider the spectral decomposition of S

$$S = V \Lambda V^T,$$

where $V = [v_1, \ldots, v_n]$ satisfies $V^T V = I_{n\times n}$ and Λ is diagonal with diagonal elements $\Lambda_{kk} = \lambda_k$ (and we organize them as $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$). Note that $S v_k = \lambda_k v_k$. Also,

$$M = D^{-\frac{1}{2}} S D^{\frac{1}{2}} = D^{-\frac{1}{2}} V \Lambda V^T D^{\frac{1}{2}} = \left(D^{-\frac{1}{2}} V\right) \Lambda \left(D^{\frac{1}{2}} V\right)^T.$$

We define $\Phi = D^{-\frac{1}{2}} V$ with columns $\Phi = [\varphi_1, \ldots, \varphi_n]$ and $\Psi = D^{\frac{1}{2}} V$ with columns $\Psi = [\psi_1, \ldots, \psi_n]$. Then

$$M = \Phi \Lambda \Psi^T,$$

and Φ, Ψ form a biorthogonal system in the sense that $\Phi^T \Psi = I_{n\times n}$ or, equivalently, $\varphi_j^T \psi_k = \delta_{jk}$. Note that φ_k and ψ_k are, respectively right and left eigenvectors of M, indeed, for all $1 \leq k \leq n$:

$$M\varphi_k = \lambda_k \varphi_k \quad \text{ and } \psi_k^T M = \lambda_k \psi_k^T.$$

Also, we can rewrite this decomposition as

$$M = \sum_{k=1}^{n} \lambda_k \varphi_k \psi_k^T.$$

and it is easy to see that

$$M^t = \sum_{k=1}^{n} \lambda_k^t \varphi_k \psi_k^T. \tag{5.1}$$

Let's revisit the embedding suggested on Remark 5.1. It would correspond to

$$v_i \to M^t[i,:] = \sum_{k=1}^{n} \lambda_k^t \varphi_k(i) \psi_k^T,$$

it is written in terms of the basis ψ_k. The Diffusion Map will essentially consist of the representing a node i by the coefficients of the above map

$$v_i \to M^t[i,:] = \begin{bmatrix} \lambda_1^t \varphi_1(i) \\ \lambda_2^t \varphi_2(i) \\ \vdots \\ \lambda_n^t \varphi_n(i) \end{bmatrix}, \tag{5.2}$$

Note that $M\mathbf{1} = \mathbf{1}$, meaning that one of the right eigenvectors φ_k is simply a multiple of $\mathbf{1}$ and so it does not distinguish the different nodes of the graph. We will show that this indeed corresponds to the the first eigenvalue.

Proposition 5.1. *All eigenvalues λ_k of M satisfy $|\lambda_k| \leq 1$.*

Proof.

Let φ_k be a right eigenvector associated with λ_k whose largest entry in magnitude is positive $\varphi_k(i_{\max})$. Then,

$$\lambda_k \varphi_k(i_{\max}) = M\varphi_k(i_{\max}) = \sum_{j=1}^{n} M_{i_{\max},j} \varphi_k(j).$$

This means, by triangular inequality that, that

$$|\lambda_k| = \sum_{j=1}^{n} \left|M_{i_{\max},j}\right| \frac{|\varphi_k(j)|}{|\varphi_k(i_{\max})|} \leq \sum_{j=1}^{n} \left|M_{i_{\max},j}\right| = 1.$$

□

Remark 5.2. It is possible that there are other eigenvalues with magnitude 1 but only if G is disconnected or if G is bipartite. Provided that G is disconnected, a natural way to remove potential periodicity issues (like the graph being bipartite) is to make the walk lazy, i.e. to add a certain probability of the walker to stay in the current node. This can be conveniently achieved by taking, e.g.,

$$M' = \frac{1}{2}M + \frac{1}{2}I.$$

By the proposition above we can take $\varphi_1 = \mathbf{1}$, meaning that the first coordinate of (5.2) does not help differentiate points on the graph. This suggests removing that coordinate:

Definition 5.1 (Diffusion Map). Given a graph $G = (V,E,W)$ construct M and its decomposition $M = \Phi\Lambda\Psi^T$ as described above. The Diffusion Map is a map $\varphi_t : V \to \mathbb{R}^{n-1}$ given by

$$\varphi_t(v_i) = \begin{bmatrix} \lambda_2^t \varphi_2(i) \\ \lambda_3^t \varphi_3(i) \\ \vdots \\ \lambda_n^t \varphi_n(i) \end{bmatrix}.$$

This map is still a map to $n-1$ dimensions. But note now that each coordinate has a factor of λ_k^t which, if λ_k is small will be rather small for moderate values of t. This motivates truncating the Diffusion Map by taking only the first d coefficients.

Definition 5.2 (Truncated Diffusion Map). Given a graph $G = (V,E,W)$ and dimension d, construct M and its decomposition $M = \Phi\Lambda\Psi^T$ as described above. The Diffusion Map truncated to d dimensions is a map $\varphi_t : V \to \mathbb{R}^d$ given by

$$\varphi_t^{(d)}(v_i) = \begin{bmatrix} \lambda_2^t \varphi_2(i) \\ \lambda_3^t \varphi_3(i) \\ \vdots \\ \lambda_{d+1}^t \varphi_{d+1}(i) \end{bmatrix}.$$

In the following theorem we show that the euclidean distance in the diffusion map coordinates (called diffusion distance) meaningfully measures distance between the probability clouds after t iterations.

Theorem 5.1. *For any pair of nodes v_{i_1}, v_{i_2} we have*

$$\|\varphi_t(v_{i_1}) - \varphi_t(v_{i_2})\|^2 = \sum_{j=1}^{n} \frac{1}{\deg(j)} \left[\text{Prob}\{X(t) = j | X(0) = i_1\} - \text{Prob}\{X(t) = j | X(0) = i_2\}\right]^2.$$

Proof.

Note that $\sum_{j=1}^{n} \frac{1}{\deg(j)} \left[\text{Prob}\{X(t) = j | X(0) = i_1\} - \text{Prob}\{X(t) = j | X(0) = i_2\}\right]^2$ can be rewritten as

$$\sum_{j=1}^{n} \frac{1}{\deg(j)} \left[\sum_{k=1}^{n} \lambda_k^t \varphi_k(i_1)\psi_k(j) - \sum_{k=1}^{n} \lambda_k^t \varphi_k(i_2)\psi_k(j)\right]^2 = \sum_{j=1}^{n} \frac{1}{\deg(j)} \left[\sum_{k=1}^{n} \lambda_k^t \left(\varphi_k(i_1) - \varphi_k(i_2)\right)\psi_k(j)\right]^2$$

and

$$\begin{aligned} \sum_{j=1}^{n} \frac{1}{\deg(j)} \left[\sum_{k=1}^{n} \lambda_k^t \left(\varphi_k(i_1) - \varphi_k(i_2)\right)\psi_k(j)\right]^2 &= \sum_{j=1}^{n} \left[\sum_{k=1}^{n} \lambda_k^t \left(\varphi_k(i_1) - \varphi_k(i_2)\right) \frac{\psi_k(j)}{\sqrt{\deg(j)}}\right]^2 \\ &= \left\| \sum_{k=1}^{n} \lambda_k^t \left(\varphi_k(i_1) - \varphi_k(i_2)\right) D^{-\frac{1}{2}} \psi_k \right\|^2. \end{aligned}$$

Note that $D^{-\frac{1}{2}}\psi_k = v_k$ which forms an orthonormal basis, meaning that

$$\left\| \sum_{k=1}^{n} \lambda_k^t \left(\varphi_k(i_1) - \varphi_k(i_2)\right) D^{-\frac{1}{2}} \psi_k \right\|^2 = \sum_{k=1}^{n} \left(\lambda_k^t \left(\varphi_k(i_1) - \varphi_k(i_2)\right)\right)^2$$
$$= \sum_{k=2}^{n} \left(\lambda_k^t \varphi_k(i_1) - \lambda_k^t \varphi_k(i_2)\right)^2,$$

where the last inequality follows from the fact that $\varphi_1 = \mathbf{1}$ and concludes the proof of the theorem.

□

5.0.7.1 A couple of examples

The ring graph is a graph on n nodes $\{1, \ldots, n\}$ such that node k is connected to $k-1$ and $k+1$ and 1 is connected to n. Figure 5.1 has the Diffusion Map of it truncated to two dimensions

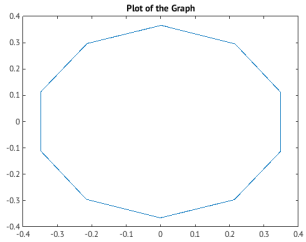

Fig. 5.1: The Diffusion Map of the ring graph gives a very natural way of displaying (indeed, if one is asked to draw the ring graph, this is probably the drawing that most people would do). It is actually not difficult to analytically compute the Diffusion Map of this graph and confirm that it displays the points in a circle.

Another simple graph is K_n, the complete graph on n nodes (where every pair of nodes share an edge), see Figure 5.2.

5.0.7.2 Diffusion Maps of point clouds

Very often we are interested in embedding in $\mathbb{R}^d$ a point cloud of points $x_1, \ldots, x_n \in \mathbb{R}^p$ and necessarily a graph. One option (as discussed before in the course) is to use Principal Component Analysis (PCA), but PCA is only designed to find linear structure of the data and the low dimensionality of the dataset may be non-linear. For example, let's say our dataset is images of the face of someone taken from different

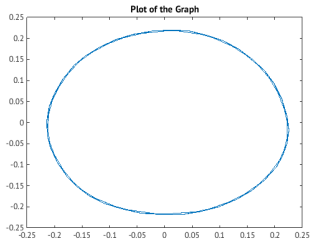

Fig. 5.2: The Diffusion Map of the complete graph on 4 nodes in 3 dimensions appears to be a regular tetrahedron suggesting that there is no low dimensional structure in this graph. This is not surprising, since every pair of nodes is connected we don't expect this graph to have a natural representation in low dimensions.

angles and lighting conditions, for example, the dimensionality of this dataset is limited by the amount of muscles in the head and neck and by the degrees of freedom of the lighting conditions (see Figure **??**) but it is not clear that this low dimensional structure is linearly apparent on the pixel values of the images.

Let's say that we are given a point cloud that is sampled from a two dimensional swiss roll embedded in three dimension (see Figure 5.3). In order to learn the two dimensional structure of this object we need to differentiate points that are near eachother because they are close by in the manifold and not simply because the manifold is curved and the points appear nearby even when they really are distant in the manifold (see Figure 5.3 for an example). We will achieve this by creating a graph from the data points.

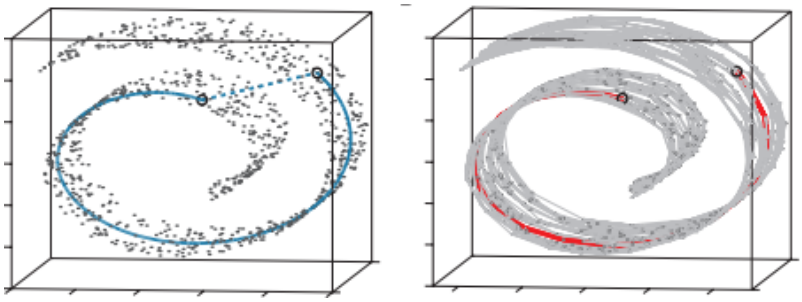

Fig. 5.3: A swiss roll point cloud (see, for example, [150]). The points are sampled from a two dimensional manifold curved in $\mathbb{R}^3$ and then a graph is constructed where nodes correspond to points.

Our goal is for the graph to capture the structure of the manifold. To each data point we will associate a node. For this we should only connect points that are

close in the manifold and not points that maybe appear close in Euclidean space simply because of the curvature of the manifold. This is achieved by picking a small scale and linking nodes if they correspond to points whose distance is smaller than that scale. This is usually done smoothly via a kernel K_ε, and to each edge (i, j) associating a weight

$$w_{ij} = K_\varepsilon\left(\|x_i - x_j\|_2\right),$$

a common example of a Kernel is $K_\varepsilon(u) = \exp\left(-\frac{1}{2\varepsilon}u^2\right)$, that gives essentially zero weight to edges corresponding to pairs of nodes for which $\|x_i - x_j\|_2 \gg \sqrt{\varepsilon}$. We can then take the the Diffusion Maps of the resulting graph.

5.0.7.3 A simple example

A simple and illustrative example is to take images of a blob on a background in different positions (image a white square on a black background and each data point corresponds to the same white square in different positions). This dataset is clearly intrinsically two dimensional, as each image can be described by the (two-dimensional) position of the square. However, we don't expect this two-dimensional structure to be directly apparent from the vectors of pixel values of each image; in particular we don't expect these vectors to lie in a two dimensional affine subspace!

Let's start by experimenting with the above example for one dimension. In that case the blob is a vertical stripe and simply moves left and right. We think of our space as the in the arcade game Asteroids, if the square or stripe moves to the right all the way to the end of the screen, it shows up on the left side (and same for up-down in the two-dimensional case). Not only this point cloud should have a one dimensional structure but it should also exhibit a circular structure. Remarkably, this structure is completely apparent when taking the two-dimensional Diffusion Map of this dataset, see Figure 5.4.

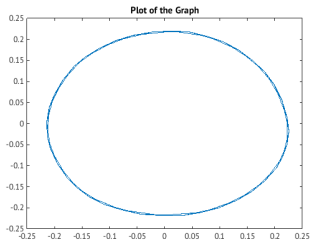

Fig. 5.4: The two-dimensional diffusion map of the dataset of the datase where each data point is an image with the same vertical strip in different positions in the x-axis, the circular structure is apparent.

For the two dimensional example, we expect the structure of the underlying manifold to be a two-dimensional torus. Indeed, Figure 5.5 shows that the three-dimensional diffusion map captures the toroidal structure of the data.

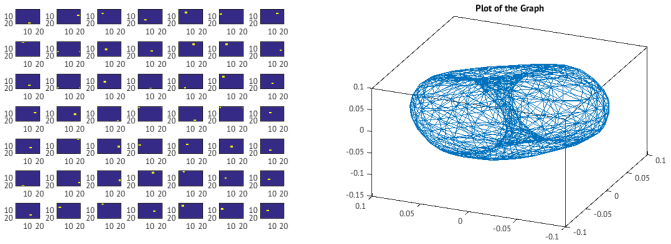

Fig. 5.5: On the left the data set considered and on the right its three dimensional diffusion map, the fact that the manifold is a torus is remarkably captured by the embedding.

5.0.7.4 Similar non-linear dimensional reduction techniques

There are several other similar non-linear dimensional reduction methods. A particularly popular one is ISOMAP [**?**]. The idea is to find an embedding in $\mathbb{R}_d$ for which euclidean distances in the embedding correspond as much as possible to geodesic distances in the graph. This can be achieved by, between pairs of nodes v_i, v_j finding their geodesic distance and then using, for example, Multidimensional Scaling to find points $y_i \in \mathbb{R}^d$ that minimize (say)

$$\min_{y_1,\ldots,y_n \in \mathbb{R}^d} \sum_{i,j} \left(\|y_i - y_j\|^2 - \delta_{ij}^2 \right)^2,$$

which can be done with spectral methods (it is a good exercise to compute the optimal solution to the above optimization problem).

5.0.8 Semi-supervised learning

Classification is a central task in machine learning. In a supervised learning setting we are given many labelled examples and want to use them to infer the label of a new, unlabeled example. For simplicity, let's say that there are two labels, $\{-1, +1\}$.

Let's say we are given the task of labeling point "?" in Figure 5.9 given the labeled points. The natural label to give to the unlabeled point would be 1.

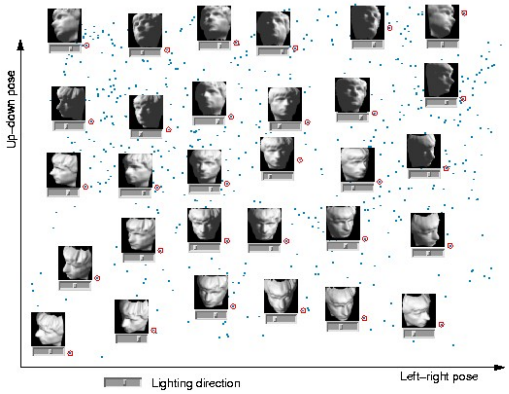

Fig. 5.6: The two dimensional represention of a data set of images of faces as obtained in [150] using ISOMAP. Remarkably, the two dimensionals are interpretable

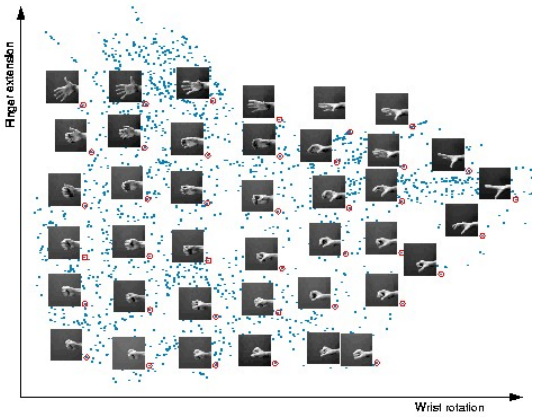

Fig. 5.7: The two dimensional represention of a data set of images of human hand as obtained in [150] using ISOMAP. Remarkably, the two dimensionals are interpretable

However, let's say that we are given not just one unlabeled point, but many, as in Figure 5.10; then it starts being apparent that -1 is a more reasonable guess.

Intuitively, the unlabeled data points allowed us to better learn the geometry of the dataset. That's the idea behind Semi-supervised learning, to make use of the fact that often one has access to many unlabeled data points in order to improve classification.

The approach we'll take is to use the data points to construct (via a kernel K_ε) a graph $G = (V, E, W)$ where nodes correspond to points. More precisely, let l denote

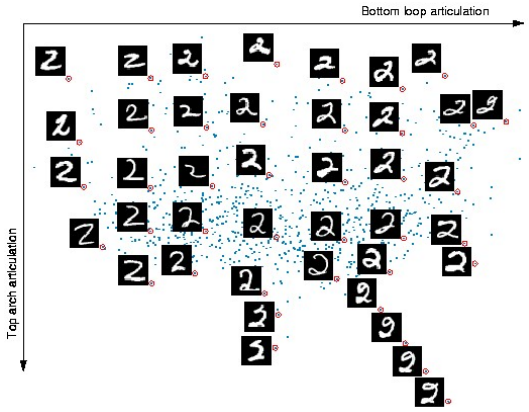

Fig. 5.8: The two dimensional represention of a data set of handwritten digits as obtained in [150] using ISOMAP. Remarkably, the two dimensionals are interpretable

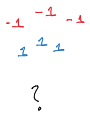

Fig. 5.9: Given a few labeled points, the task is to label an unlabeled point.

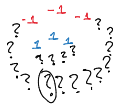

Fig. 5.10: In this example we are given many unlabeled points, the unlabeled points help us learn the geometry of the data.

the number of labeled points with labels $f_1,\ldots,f_l$, and u the number of unlabeled points (with $n = l+u$), the first l nodes $v_1,\ldots,v_l$ correspond to labeled points and the rest $v_{l+1},\ldots,v_n$ are unlabaled. We want to find a function $f : V \to \{-1,1\}$ that agrees on labeled points: $f(i) = f_i$ for $i = 1,\ldots,l$ and that is "as smooth as possible" the graph. A way to pose this is the following

$$\min_{f:V\to\{-1,1\}:\, f(i)=f_i\, i=1,\ldots,l} \sum_{i<j} w_{ij}\,(f(i)-f(j))^2\,.$$

Instead of restricting ourselves to giving $\{-1,1\}$ we allow ourselves to give real valued labels, with the intuition that we can "round" later by, e.g., assigning the sign of $f(i)$ to node i.

We thus are interested in solving

$$\min_{f:V\to\mathbb{R}:\, f(i)=f_i\, i=1,\ldots,l} \sum_{i<j} w_{ij}\left(f(i)-f(j)\right)^2.$$

If we denote by f the vector (in $\mathbb{R}^n$ with the function values) then we are can rewrite the problem as

$$\begin{aligned}\sum_{i<j} w_{ij}\left(f(i)-f(j)\right)^2 &= \sum_{i<j} w_{ij}\left[(e_i-e_j)f\right]\left[(e_i-e_j)f\right]^T \\ &= \sum_{i<j} w_{ij}\left[(e_i-e_j)^T f\right]^T\left[(e_i-e_j)^T f\right] \\ &= \sum_{i<j} w_{ij} f^T (e_i-e_j)(e_i-e_j)^T f \\ &= f^T\left[\sum_{i<j} w_{ij}(e_i-e_j)(e_i-e_j)^T\right] f\end{aligned}$$

The matrix $\sum_{i<j} w_{ij}(e_i-e_j)(e_i-e_j)^T$ will play a central role throughout this course, it is called the graph Laplacian [51].

$$L_G := \sum_{i<j} w_{ij}(e_i-e_j)(e_i-e_j)^T.$$

Note that the entries of L_G are given by

$$(L_G)_{ij} = \begin{cases} -w_{ij} & \text{if } i\neq j \\ \deg(i) & \text{if } i=j, \end{cases}$$

meaning that

$$L_G = D - W,$$

where D is the diagonal matrix with entries $D_{ii}=\deg(i)$.

Remark 5.3. Consider an analogous example on the real line, where one would want to minimize

$$\int f'(x)^2 dx.$$

Integrating by parts

$$\int f'(x)^2 dx = \text{Boundary Terms} - \int f(x)f''(x)dx.$$

Analogously, in $\mathbb{R}^d$:

$$\int \|\nabla f(x)\|^2 dx = \int \sum_{k=1}^{d} \left(\frac{\partial f}{\partial x_k}(x)\right)^2 dx = \text{B. T.} - \int f(x) \sum_{k=1}^{d} \frac{\partial^2 f}{\partial x_k^2}(x) dx = \text{B. T.} - \int f(x) \Delta f(x) dx,$$

which helps motivate the use of the term graph Laplacian.

Let us consider our problem

$$\min_{f:V\to\mathbb{R}:\, f(i)=f_i\, i=1,\ldots,l} f^T L_G f.$$

We can write

$$D = \begin{bmatrix} D_l & 0 \\ 0 & D_u \end{bmatrix}, \quad W = \begin{bmatrix} W_{ll} & W_{lu} \\ W_{ul} & W_{uu} \end{bmatrix}, \quad L_G = \begin{bmatrix} D_l - W_{ll} & -W_{lu} \\ -W_{ul} & D_u - W_{uu} \end{bmatrix}, \quad \text{and } f = \begin{bmatrix} f_l \\ f_u \end{bmatrix}.$$

Then we want to find (recall that $W_{ul} = W_{lu}^T$)

$$\min_{f_u \in \mathbb{R}^u} f_l^T \left[D_l - W_{ll}\right] f_l - 2 f_u^T W_{ul} f_l + f_u^T \left[D_u - W_{uu}\right] f_u.$$

by first-order optimality conditions, it is easy to see that the optimal satisfies

$$\left(D_u - W_{uu}\right) f_u = W_{ul} f_l.$$

If $D_u - W_{uu}$ is invertible[1] then

$$f_u^* = \left(D_u - W_{uu}\right)^{-1} W_{ul} f_l.$$

Remark 5.4. The function f function constructed is called a harmonic extension. Indeed, it shares properties with harmonic functions in euclidean space such as the mean value property and maximum principles; if v_i is an unlabeled point then

$$f(i) = \left[D_u^{-1} \left(W_{ul} f_l + W_{uu} f_u\right)\right]_i = \frac{1}{\deg(i)} \sum_{j=1}^{n} w_{ij} f(j),$$

which immediately implies that the maximum and minimum value of f needs to be attained at a labeled point.

5.0.8.1 An interesting experience and the Sobolev Embedding Theorem

Let us try a simple experiment. Let's say we have a grid on $[-1,1]^d$ dimensions (with say m^d points for some large m) and we label the center as $+1$ and every node that is at distance larger or equal to 1 to the center, as -1. We are interested in understanding how the above algorithm will label the remaining points, hoping

[1] It is not difficult to see that unless the problem is in some form degenerate, such as the unlabeled part of the graph being disconnected from the labeled one, then this matrix will indeed be invertible.

that it will assign small numbers to points far away from the center (and close to the boundary of the labeled points) and large numbers to points close to the center.

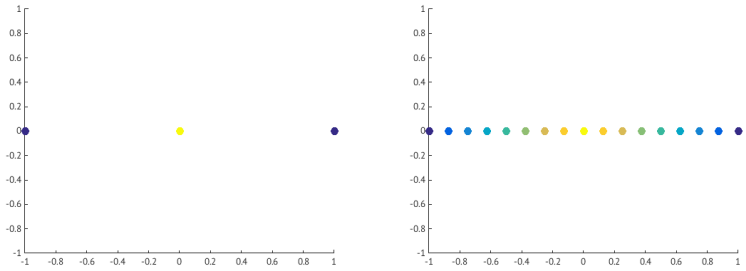

Fig. 5.11: The $d = 1$ example of the use of this method to the example described above, the value of the nodes is given by color coding. For $d = 1$ it appears to smoothly interpolate between the labeled points.

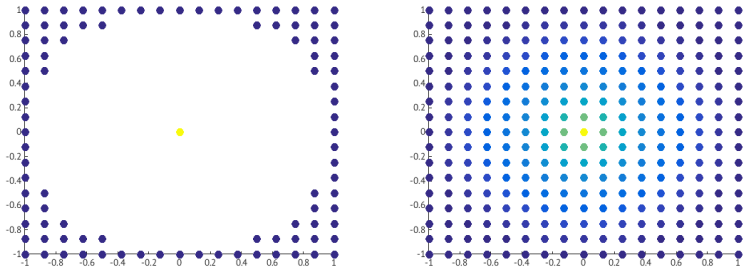

Fig. 5.12: The $d = 2$ example of the use of this method to the example described above, the value of the nodes is given by color coding. For $d = 2$ it appears to smoothly interpolate between the labeled points.

See the results for $d = 1$ in Figure 5.11, $d = 2$ in Figure 5.12, and $d = 3$ in Figure 5.13. While for $d \leq 2$ it appears to be smoothly interpolating between the labels, for $d = 3$ it seems that the method simply learns essentially -1 on all points, thus not being very meaningful. Let us turn to $\mathbb{R}^d$ for intuition:

Let's say that we want to find a function in $\mathbb{R}^d$ that takes the value 1 at zero and -1 at the unit sphere, that minimizes $\int_{B_0(1)} \|\nabla f(x)\|^2 dx$. Let us consider the following function on $B_0(1)$ (the ball centered at 0 with unit radius)

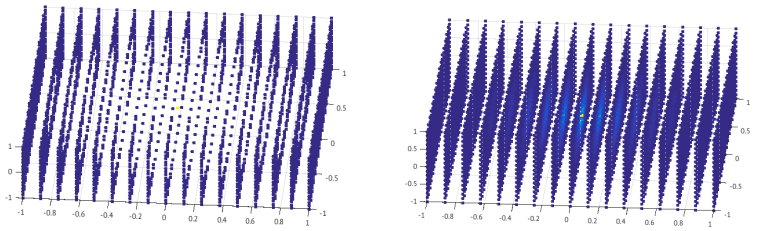

Fig. 5.13: The $d = 3$ example of the use of this method to the example described above, the value of the nodes is given by color coding. For $d = 3$ the solution appears to only learn the label -1.

$$f_\varepsilon(x) = \begin{cases} 1 - 2\frac{|x|}{\varepsilon} & \text{if} |x| \leq \varepsilon \\ -1 & \text{otherwise.} \end{cases}$$

A quick calculation suggest that

$$\int_{B_0(1)} \|\nabla f_\varepsilon(x)\|^2 dx = \int_{B_0(\varepsilon)} \frac{1}{\varepsilon^2} dx = \text{vol}(B_0(\varepsilon)) \frac{1}{\varepsilon^2} dx \approx \varepsilon^{d-2},$$

meaning that, if $d > 2$, the performance of this function is improving as $\varepsilon \to 0$, explaining the results in Figure 5.13.

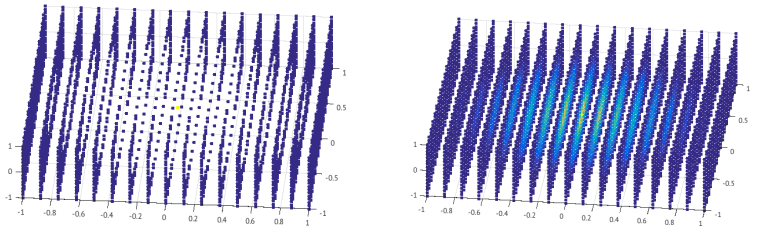

Fig. 5.14: The $d = 3$ example of the use of this method with the extra regularization $f^T L^2 f$ to the example described above, the value of the nodes is given by color coding. The extra regularization seems to fix the issue of discontinuities.

One way of thinking about what is going on is through the Sobolev Embedding Theorem. $H^m\left(\mathbb{R}^d\right)$ is the space of function whose derivatives up to order m are square-integrable in $\mathbb{R}^d$, Sobolev Embedding Theorem says that if $m > \frac{d}{2}$ then, if $f \in H^m\left(\mathbb{R}^d\right)$ then f must be continuous, which would rule out the behavior observed in Figure 5.13. It also suggests that if we are able to control also second derivates of f then this phenomenon should disappear (since $2 > \frac{3}{2}$). While we will not describe

it here in detail, there is, in fact, a way of doing this by minimizing not $f^T Lf$ but $f^T L^2 f$ instead, Figure 5.14 shows the outcome of the same experiment with the $f^T Lf$ replaced by $f^T L^2 f$ and confirms our intuition that the discontinuity issue should disappear (see, e.g., [113] for more on this phenomenon).