Learning Kernel-based Approximate Isometries

2017

Mahlagha Sedghi
University of Central Florida

Find similar works at: https://stars.library.ucf.edu/etd

University of Central Florida Libraries http://library.ucf.edu

Part of the Electrical and Computer Engineering Commons

STARS Citation

https://stars.library.ucf.edu/etd/5930

This Masters Thesis (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of STARS. For more information, please contact lee.dotson@ucf.edu.
LEARNING KERNEL-BASED APPROXIMATE ISOMETRIES

by

MAHLAGHA SEDGHI
B.S. Sharif University of Technology, 2014

A thesis submitted in partial fulfilment of the requirements for the degree of Master of Science in the Department of Electrical and Computer Engineering in the College of Engineering and Computer Science at the University of Central Florida
Orlando, Florida

Summer Term
2017

Major Professor: Michael Georgioupolos
The increasing availability of public datasets offers an inexperienced opportunity to conduct data-driven studies. Metric Multi-Dimensional Scaling aims to find a low-dimensional embedding of the data, preserving the pairwise dissimilarities amongst the data points in the original space. Along with the visualizability, this dimensionality reduction plays a pivotal role in analyzing and disclosing the hidden structures in the data. This work introduces Sparse Kernel-based Least Squares Multi-Dimensional Scaling approach for exploratory data analysis and, when desirable, data visualization. We assume our embedding map belongs to a Reproducing Kernel Hilbert Space of vector-valued functions which allows for embeddings of previously unseen data. Also, given appropriate positive-definite kernel functions, it extends the applicability of our method to non-numerical data. Furthermore, the framework employs Multiple Kernel Learning for implicitly identifying an effective feature map and, hence, kernel function. Finally, via the use of sparsity-promoting regularizers, the technique is capable of embedding data on a, typically, lower-dimensional manifold by naturally inferring the embedding dimension from the data itself. In the process, key training samples are identified, whose participation in the embedding map’s kernel expansion is most influential. As we will show, such influence may be given interesting interpretations in the context of the data at hand. The resulting multi-kernel learning, non-convex framework can be effectively trained via a block coordinate descent approach, which alternates between an accelerated proximal average method-based iterative majorization for learning the kernel expansion coefficients and a simple quadratic program, which deduces the multiple-kernel learning coefficients. Experimental results showcase potential uses of the proposed framework on artificial data as well as real-world datasets, that underline the merits of our embedding framework. Our method discovers genuine hidden structure in the data, that in case of network data, matches the results of well-known Multi-level Modularity Optimization community structure detection algorithm.
Dedicated to my parents,

my sister Parisa,

and my beloved husband
ACKNOWLEDGMENTS

Mahlagha Sedghi acknowledges the support from the BOG grant (CSIT TEAm grant) from the State of Florida.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>LIST OF FIGURES</th>
<th>viii</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAPTER 1: INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Notation</td>
<td>4</td>
</tr>
<tr>
<td>CHAPTER 2: LITERATURE REVIEW</td>
<td>5</td>
</tr>
<tr>
<td>CHAPTER 3: MODEL FORMULATION</td>
<td>7</td>
</tr>
<tr>
<td>CHAPTER 4: ALGORITHM</td>
<td>12</td>
</tr>
<tr>
<td>CHAPTER 5: EXPERIMENTS</td>
<td>17</td>
</tr>
<tr>
<td>5.1 Artificial Datasets</td>
<td>20</td>
</tr>
<tr>
<td>5.2 Real-world Datasets</td>
<td>22</td>
</tr>
<tr>
<td>5.3 Experimental Results</td>
<td>22</td>
</tr>
<tr>
<td>5.3.1 Visualization</td>
<td>23</td>
</tr>
<tr>
<td>5.3.2 Multiple Kernel Learning</td>
<td>26</td>
</tr>
<tr>
<td>5.3.3 Effect of First Regularizer, $R_c$</td>
<td>29</td>
</tr>
<tr>
<td>5.3.4 Effect of Second Regularizer, $R_r$</td>
<td>37</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure 5.1: Artificial Datasets .................................................. 21

Figure 5.2: Embedding of Grid2D and Sphere datasets; both embeddings resemble their original form, showing that our embedding method preserves the structure in the data. ............................................... 23

Figure 5.3: Lenses dataset embedding demonstrates a nice linearly separable structure, mapping the data belonging to different categories to distinct spatial regions. . 24

Figure 5.4: Scientific-43 dataset embedding with different initializations; Both figures show a similar structure of 3 lobes emanating from a central point. Different initializations have been examined to observe the effect of different initial points. Our visualization discovers the same interesting structures in the data, independent of different initialization of the Iterative Majorization (IM) algorithm. .................................................. 25

Figure 5.5: Scientific-43 dataset embedding with community detection algorithm members demonstrated in different colors. In this figure the result of the MMO community detection algorithm is superimposed on our embedding. Each color represents a sub-community of the graph, and, visibly, this segmentation perfectly matches our detected lobe structures. ........................................ 26
Figure 5.6: *Scientific-121* dataset embedding with different initializations. As in Figure 5.4, two different initializations have been showed to avoid local minima. More sub-groups are observed in this bigger sub-cluster of the data, but there is a strong similarity with the subgroups identified for the smaller sub-cluster of the data-43.

Figure 5.7: *Grid2D* dataset embedding using an ensemble of 8 Gaussian kernels. Our method optimizes the kernel coefficients using the Multiple Kernel Learning (MKL) scheme. Only 3 kernels are chosen as dominant ones in the expansion.

Figure 5.8: *Grid2D* dataset embedding using an ensemble of 6 kernels including 4 polynomial kernels, one triangle, and one Gaussian kernel. The optimized values of kernel coefficients reveal that the linear kernel gets a considerable share reversing the dataset’s construction via random rotation. The embedding result shows a significant fidelity to the original form of the dataset.

Figure 5.9: *SwissRoll* dataset embedding using an ensemble of 8 Gaussian kernels. 3 out of 8 kernels have been almost zeroed out through the MKL procedure.

Figure 5.10: *SwissRoll* dataset embedding using an ensemble of 6 kernels including 4 polynomial kernels, one triangle, and one Gaussian kernel. The significance of the quadratic kernel coefficient is reflected as the unfolded structure of the roll in an approximate grid from.
Figure 5.11: Evolution of Grid2D dataset with increasing $\lambda$. Higher column-norms contribute more to the kernel expansion of the embedding and the corresponding samples are considered as more influential ones. More influential training samples are displayed by darker points, and the less influential ones, are shown as lighter points. The four corner points are the most influential ones, and moving towards the center of the grid decreases the influence of samples. Increasing $\lambda$ pushes more column-norms toward zero, and hence central points look whiter. With a high value of $\lambda$ there are less samples incorporating to the kernel expansion, and this results in an embedding where outer samples are moved towards the center of the embedding, where the most similar samples lie.

Figure 5.12: Evolution of Sphere dataset with increasing $\lambda$. Higher values of $\lambda$ causes less samples to have considerable contributions to the kernel expansion of the embedding. With $\lambda = 100$ the model uses the training samples lying on a great circle of the sphere, and ultimately, the ones on the diameter.

Figure 5.13: Embedding of Sphere dataset, with $\lambda = 100$, different view. The samples have been shaded with a blue tone, but in proportion to the magnitude of their column-norms. The four most influential samples are the darker ones accompanied with their corresponding indices. These points are located close to the four vertices and co-vertices of this ellipse-shaped embedding.

Figure 5.14: Scientific-43 dataset Bar chart of Column Norms. Higher values indicate more influential samples. The 10 most influential ones are indexed by 6, 5, 7, 35, 37, 38, 36, 43, 39, and 40, respectively.
Figure 5.15: Top-7 important samples of *Scientific*-43 dataset. As observed before, our visualization located the most influential samples at the boundaries which enables us to identify them by a radius from the embedding center. 7 samples are retrieved when this radius is set to 80% of the maximum distance from the mean. These 7 nodes are all amongst the 10 most influential samples.

Figure 5.16: Top-3 important samples of *Scientific*-43 dataset. Setting the radius to 85% of the maximum distance from the mean gives us 3 boundary points, which exactly match the first three influential points.

Figure 5.17: Evolution of *Grid2D* dataset embedding with increasing $\mu$. Higher value of $\mu$ decreases the dimensionality of the embedding and transforms the grid structure to a straight 1D line.

Figure 5.18: Evolution of *SwissRoll* dataset embedding with increasing $\mu$. As $\mu$ increases, the embedding is forced to lie in a lower dimension. $\mu = 100$ has nicely flattened out the roll structure to a 2D grid form. Increasing $\mu$ even more, squeezes the grid form inside, and eventually gets a 1D line.

Figure 5.19: *Scientific*-43 dataset embedding ($\mu = 100$), different views. As observed before, this embedding shows the effect of second regularization parameter, which decreases the embedding dimension. The 3D visualization has been projected to a 2-dimensional structure.
CHAPTER 1: INTRODUCTION

Metric Multi-Dimensional Scaling (Metric MDS) has been proven to be a valuable technique for visualizing and interpreting high-dimensional data (a good reference is [9]), especially in the case of mixed-type attributes, as it is often the case with phenomena, for example, in the social sciences (e.g. see [44]). In the case that, the visualization of the data is not aimed per se, one can be interested in preserving the structure of the data in a lower dimension, before applying any other developed methods, due to the curse of dimensionality ([21], [27], [19]) . Scientific visualization and data mining in a variety of fields such as biological sciences [15], bioinformatics [37], cognitive science [20], neuroimaging [39], psychological sciences [22], [28], psychophysics [24], psychometrics [44], and marketing [14] are some applications of Metric MDS. Furthermore, geostatistics is one of the other applications of Multi-Dimensional Scaling (MDS), where the spatial variability of the patterns of an image is modeled by representing them in a lower-dimensional space [18], as well as Isomap, a well known non-linear dimensionality technique [43], and natural language processing for modeling the semantic and affective relatedness concepts of natural language [7].

The main idea behind Metric MDS is to find a low-dimensional, Euclidean embedding space, in which the pairwise distances between the data’s images represent the pairwise dissimilarities of the data in the original space with as much fidelity as possible. The way these dissimilarities are defined is context- and problem-dependent and, ultimately, affect the data’s embedding and subsequent interpretations. A particular popular multi-dimensional scaling approach is Least Squares Multi-Dimensional Scaling (Least Squares MDS), which aims to minimize the average squared discrepancy between the original dissimilarities and the produced distances. The traditional version of the algorithm does not allow for embedding data outside the set used for constructing the embedding itself. This limitation has been overcome by a variety of interpolation methods...
(e.g. see [5, 48, 45, 1, 29]), of which one, Kernel-based Least Squares Multi-Dimensional Scaling (Kernel-based LS-MDS) ([46, 48]) is of particular interest, as it is based on the theory of positive definite kernels and their associated Reproducing Kernel Hilbert Spaces (RKHSs).

Kernel methods have been extensively used in a variety of machine learning problems due to their strong mathematical background and practical benefits. The idea behind kernel methods is to map the data into a possibly high-dimensional implicit feature space, where the problem can be handled by simple methods. For example, they enable linear discriminant methods to be applied to problems where the data are not linearly separable. The “kernel-trick” avoids the expensive, and in some cases intractable, calculation of the desirable feature vectors, and thus offers computational efficiency. Kernel methods also extend the applicability of many methods, initially designed for vectors, to non-numerical data such as strings, trees, and graphs. Aside from the classification methods, many others like clustering, regression, and ranking methods have used kernel methods to different extents, amongst which kernel-based metric MDS has not been fully investigated.

This paper explores Sparse Kernel-based LS-MDS, a novel Kernel-based LS-MDS approach. (i) We assume an embedding map to belong to a RKHS of vector-valued functions. This characteristic allows our framework to readily embed previously unseen, out-of-sample data points via meaningful interpolation. Furthermore, as a kernel-based method, it is also readily applicable to non-numerical data as well, given appropriate positive-definite kernel functions for such data. What clearly sets Sparse Kernel-based LS-MDS apart from previous relevant approaches are: (ii) via the use of sparsity-inducing regularizers, Sparse Kernel-based LS-MDS is naturally capable of identifying an appropriate embedding dimension, as well as (iii) specific training samples that play a key role in defining the embedded manifold’s specific structure. The importance of the latter samples is often correlated to insightful interpretation within the context of the data at hand, adding credibility to the value of the proposed model. (iv) Finally, in order to avoid the computational cost of tuning kernel hyper-parameters, the framework employs Multiple Kernel Learning (MKL)
for implicitly identifying an effective RKHS feature map and, hence, kernel function. In order to solve the resulting problem, a new algorithm has been designed to fit the associated non-convex and non-differentiable optimization problem, which is even more challenging because each of the constituent penalty terms have overlapping components.

In this thesis, we will start with a brief discussion about existent methods on kernel-based metric MDS, as well as other recent related literature and continue with notation associated with this thesis. In Section 3, we first review the standard Least Squares MDS model and then present a generalization of the Kernel-based LS-MDS model. The corresponding embedding components, which is derived from a Representer Theorem by [38], are given as a kernel expansion involving real-valued kernels. Next, a novel kernel-based Least Squares MDS model is developed that achieves benefits (ii) and (iii). Then, an Iterative Majorization (IM) algorithm is gradually presented that is capable of fitting our proposed model. Section 5 reports some experimental results that demonstrate the potential of the proposed model. Aside from the meaningful visualization of the data, our method is capable of discovering genuine hidden structures of the data that comply with the manifold hypothesis. Moreover, our observed structures of the graph-type data perfectly matches the sub-communities identified by the community detection algorithms of network science. Our embedding method maps the identified influential samples to the boundary points of the embedded map enabling them to be easily recognizable by means of 2/3-D visualization and facilitates interesting interpretations about the problem. We are also able to control the embedding dimension through learning procedure, as well as inferring the intrinsic dimensionality of the data. Finally, the conclusions are presented in Section 6.
1.1 Notation

The following notational conventions will be followed in this manuscript: \( \mathbb{R} \) denotes the set of real numbers and \( \mathbb{N} \) the set of natural numbers; \( \mathbb{S}^N \) denotes the set of \( N \times N \) real-valued symmetric matrices, while \( \mathbb{S}^N_+ \) will stand for the elements of \( \mathbb{S}^N \) with non-negative entries. On the other hand, if \( \mathbb{H}^N \) is the set of all hollow (zero diagonal entries) \( N \times N \) real-valued matrices, then we define \( \mathbb{S}^N_o \equiv \mathbb{S}^N \cap \mathbb{H}^N \) and \( \mathbb{S}^N_{o+} \equiv \mathbb{S}^N_+ \cap \mathbb{H}^N \). Finally, let \( \mathbb{N}_k \equiv \{1, \ldots, k\} \) for any \( k \in \mathbb{N} \).

Vector and matrices are denoted in boldface. Vectors are assumed to be column vectors. For a square matrix \( \mathbf{A} \in \mathbb{S}^N \) the notation \( \mathbf{A} \succeq 0 \) will mean that \( \mathbf{A} \) is positive semi-definite. \( \cdot^T \) denotes vector/matrix transposition of its argument, while trace \( \{\cdot\} \) denotes the trace of its argument (square matrix). vec\( (\cdot) \) will denote the matrix vectorization operator that stacks in order the columns of its argument into a single column vector. Moreover, diag\( (\cdot) \) will denote the operator that forms a diagonal matrix, whose diagonal entries equal the respective elements of its vector-valued argument. \( \|\cdot\| \) will denote a norm of a metric space and \( \|\cdot\|_2 \) will denote the usual Euclidean \( (L_2) \) norm. Additionally, \( \cdot^\dagger \) will denote the Moore-Penrose pseudo-inverse of its argument; note that if \( x \) is a real scalar, \( x^\dagger = 1/x \), if \( x \neq 0 \), and \( x^\dagger = 0 \), when \( x = 0 \). Also, \( \odot \) and \( \otimes \), denote the usual Hadamard and Kronecker products of matrices.

Vector \( \mathbf{1}_N \in \mathbb{R}^N \) will denote the all-ones column vector with \( N \) elements. Matrix \( \mathbf{I}_N \in \mathbb{R}^{N \times N} \) will denote an identity matrix. Moreover, \( \mathcal{L} \{\cdot\} \) will stand for the ordinary graph Laplacian operator; if \( \mathbf{A} \in \mathbb{R}^{N \times N} \) is a symmetric, weighted adjacency matrix of a given graph, then \( \mathcal{L} \{\mathbf{A}\} \equiv \text{diag} \{\mathbf{A}\mathbf{1}_N\} - \mathbf{A} \). Finally, we consider the group LASSO norm (\( \ell_{2,1} \) block norm) \( \|\cdot\|_{2,1} \); if \( \mathbf{A} \equiv [\mathbf{a}_1 \cdots \mathbf{a}_N] \), then \( \|\mathbf{A}\|_{2,1} \equiv \sum_{n=1}^N \|\mathbf{a}_n\|_2 \).
CHAPTER 2: LITERATURE REVIEW

Some of the existent literature is related to our work. After reviewing kernel-based metric MDS methods, we will go over other recent papers in this area.

Seminal work on kernel-based models for metric MDS has been done in [46], which considered a Radial Basis Function model to implement the embedding. The use of Gaussian basis functions in this work renders a kernel-based model, albeit outside the context of RKHSs. The model itself was fitted by an IM algorithm, which generalized the work originally done in [12]. This line of research was furthered by [48], in which the embedding was \textit{a priori} assumed to be a linear combination of pre-specified basis functions, rather than been derived from fundamental principles. Again in [48], Gaussian radial basis functions were used, and an IM algorithm was presented to fit the model parameters (expansion coefficients).

More recently, [13] considered a kernel-based setting for robust applications of MDS to combat the presence of outliers in the data. For their outlier-aware dissimilarity model the authors in [13] employed regularization techniques to discount potential outliers, which led to the development of a specialized IM algorithm. This work is singled out, since our paper also proposes a regularization approach to kernel-based MDS.

[1] modifies the basic MDS approach with an IM algorithm that addresses the interpolation problem, so that MDS can be extended to out-of-sample data. In this paper MDS is combined with K-nearest neighbor method to address the interpolation issue. In particular, first the data is sampled and the dimensionality of the sampled data is reduced using a well-known MDS method called Scaling by MAjorizing a COmplicated Function (SMACOF), with a modified majorizer. Then, for a new data point, its k-nearest neighbors are chosen amongst the sample data and this time the MDS is run using the new data point and its k-nearest neighbors. This way, the computational
costs and memory usage are reduced, although the resulting embedding is an approximation of the actual mapping via MDS. Furthermore, for each data point the computations do not depend on all the data but only on the point’s nearest neighbors and as such the algorithm can be parallelized when multiple out-of-sample points are considered. But note that doing so may bias the test outcomes toward similar training samples. In our proposed method, we present a similar benefaction, such that new out-of-sample data points can be mapped using interpolation; but with the difference that we obtain the exact mapping of the unseen test samples via the learned embedding function, rather than approximating them. As there is no need for running the training algorithm for the new data point added to the dataset, time and space complexities are reduced considerably.

In [3] the authors have applied the Deterministic Annealing (DA) method to overcome the local minima problem of MDS. In this method an extra parameter $T$, referred to as temperature, is added to the objective function. To avoid getting trapped in local optima, one starts the experimentation with a high value for $T$ and slowly decreases the temperature in an effort of finding a solution. The authors have shown that the running time for this method is less than the original MDS formulation or a smoothed version of it. Furthermore, they end up with lower values for the objective function when DA is used. In another recent paper [2], the authors try to parallelize the SMACOF. In their effort they distribute the data and the processes into several blocks, to overcome the limitations of memory size and runtime cost, and utilize an existing iterative majorizing algorithm to solve the MDS problem.
Assume an arbitrary set $\mathcal{X}$, which we will refer to as the native (input) space, although it may not be a bonafide vector space. Furthermore, assume a sample $\mathcal{T} \triangleq \{x_n \in \mathcal{X}\}_{n \in \mathbb{N}}$, each of which is a $D$-tuple. We will abuse terminology once more by referring to $D$ as the dimensionality of $\mathcal{X}$. Finally, assume that we are in possession of $N(N-1)/2$ dissimilarities $\delta_{m,n}$ between the aforementioned $N$ samples of $\mathcal{X}$. These dissimilarities are, typically, organized into a matrix $\Delta \in \mathbb{S}_o^N$ and have some qualities of a metric: $\delta_{m,n} = \delta_{n,m}$ and $\delta_{m,m} = 0$ for every $m, n = 1, \ldots, N$. No additional generative or distributional assumptions are made. Finally, assume that $\mathcal{Y}$ is a $P$-dimensional metric space equipped with a norm $\|\cdot\|$. In the most general case, Metric MDS aims to identify a point configuration $\{f_n \in \mathcal{Y}\}_{n=1,\ldots,N}$, such that $d_{m,n} \approx \delta_{m,n}$ $\forall$ $m, n \in \mathbb{N}_N$, where $d_{m,n} \triangleq \|f_m - f_n\|$. One can envision the data lying on an implied manifold in the native space, where geodesic distances on that manifold are given by these dissimilarities. In that sense, minimizing the discrepancies would correspond to learning an isometry, form the manifold to an Euclidean embedding space, where discrepancies are defined as the difference between the original dissimilarities and embedded distances ($d_{m,n} - \delta_{m,n}$). This can be regarded as the main difference between Metric MDS methods and Johnson-Lindenstrauss lemma ([23]) where it only considers Euclidean spaces. This task does not necessarily presuppose access to $\mathcal{T}$, but only to $\Delta$. Typically, $\mathcal{Y} \equiv \mathbb{R}^P$ and $\|\cdot\| \equiv \|\cdot\|_2$, i.e. the Euclidean ($\ell_2$) norm. Then, Metric MDS’ learning task amounts to

$$\min_{\mathbf{F}} \sum_{m<n} \ell(d_{m,n}, \delta_{m,n})$$

(3.1)

where $\mathbf{F} \triangleq [f_1 \ f_2 \ \ldots \ f_N]^T \in \mathbb{R}^{N \times P}$ is the matrix containing all $N$ configuration points arranged as rows. The $U$ matrix can be used to assign more or less weight to individual discrepancies between
dissimilarities and distances in the embedding space. We also used here the notational convention

$$\sum_{m<n} \equiv \sum_{m=1}^{N-1} \sum_{n=m+1}^N$$

(3.2)

In Equation (3.1), $\ell$ is an appropriate loss function, which tends to zero, when $d_{m,n} \approx \delta_{m,n} \quad \forall \ m, n \in \mathbb{N}_N$. A typical choice for the loss function is $\ell (d_{m,n}, \delta_{m,n}) \triangleq (d_{m,n} - \delta_{m,n})^2$, which gives rise to Least Squares MDS:

$$\min_{\mathbf{F}} \sum_{m<n} u_{m,n} (d_{m,n} - \delta_{m,n})^2$$

(3.3)

$$\mathcal{E} (\mathbf{F}) \triangleq$$

If we define

$$\mathbf{S}_A \triangleq \mathcal{L} \{ \mathbf{U} \}$$

(3.4)

$$\mathbf{Z} (\mathbf{F}) \in \mathbb{S}^N_{++} : z_{m,n} \triangleq u_{m,n} \delta_{m,n} d_{m,n}^T$$

(3.5)

$$\mathbf{S}_B (\mathbf{F}) \triangleq \mathcal{L} \{ \mathbf{Z} (\mathbf{F}) \}$$

(3.6)

$$\kappa \triangleq \frac{1}{2} \mathbf{1}^T \left( \mathbf{U} \circ \Delta^{(2)} \right) \mathbf{1} = \sum_{m<n} u_{m,n} \delta_{m,n}^2$$

(3.7)

where $\circ$ denotes the Hadamard (element-wise) matrix product and $\Delta^{(2)} \triangleq \Delta \circ \Delta$, the Least Squares MDS objective can be compactly written as

$$\mathcal{E} (\mathbf{F}) = \text{trace} \left\{ \mathbf{F}^T \left[ \mathbf{S}_A - 2 \mathbf{S}_B (\mathbf{F}) \right] \mathbf{F} \right\} + \kappa$$

(3.8)

Minimizing $\mathcal{E} (\mathbf{F})$, referred to as stress in the literature, with respect to $\mathbf{F}$ corresponds to the classic Least Squares MDS problem, which was originally considered in [25] and [17].

In this work, we aim at an inductive kernel-based Least Squares MDS model that utilizes a learn-
able map $f : \mathcal{X} \to \mathbb{R}^P$ to embed test samples via interpolation into a $P$-dimensional Euclidean space. In particular, we choose $f \triangleq [f_1 \cdots f_P]^T$, where the component functions $f_p$ belong to a RKHS $\mathcal{H}$ of real-valued functions on $\mathcal{X}$ with reproducing kernel $k$, feature mapping $\phi$ and inner product $\langle \cdot, \cdot \rangle_\mathcal{H}$, such that $\langle f_p, \phi(x) \rangle_\mathcal{H} = f_p(x)$ and $\langle \phi(x), \phi(x') \rangle_\mathcal{H} = k(x, x')$ for all $x, x' \in \mathcal{X}$.

Given these assumptions, the configuration point (embedding) $f_n$ of the $n$th training sample is given as

$$f_n = f(x_n) = \begin{bmatrix} f_1(x_n) \\ \vdots \\ f_P(x_n) \end{bmatrix} = \begin{bmatrix} \langle f_1, \phi(x_n) \rangle_\mathcal{H} \\ \vdots \\ \langle f_P, \phi(x_n) \rangle_\mathcal{H} \end{bmatrix} \quad (3.9)$$

Problem 3.3, where the sample embeddings are given by (3.9), can be recast as minimization of $\mathcal{E}(F)$ with respect to $\{f_p \in \mathcal{H}\}_{p=1}^P$. Along the lines of the Representer theorem in [38], noticing that the cost function depends only on inner products of the component functions with the images $\{\phi(x_n)\}_{n=1}^N$, it suffices to only consider functions $f_p$ of $\mathcal{H}$ that lie in span $\{\phi(x_1), \ldots, \phi(x_N)\}$. This implies that each $f_p$ can be expressed as $f_p = \sum_{n=1}^N c_{p,n} \phi(x_n)$ for some coefficients $c_{p,n} \in \mathbb{R}$. Substituting this form into (3.9) yields

$$f_n = f(x_n) = \begin{bmatrix} \sum_{m=1}^N c_{1,n} k(x_n, x_m) \\ \vdots \\ \sum_{m=1}^N c_{P,n} k(x_n, x_m) \end{bmatrix} \Rightarrow F = KC^T \quad (3.10)$$

where the (symmetric) kernel matrix $K \in \mathbb{R}^{N \times N}$ consists of all pairwise kernel function evaluations on the training set and $C \in \mathbb{R}^{P \times N}$ contains all of the kernel expansion coefficients. Using
Problem 3.3 becomes

\[
\min_{C \in \mathbb{R}^{P \times N}} \text{trace} \left\{ CK [S_A - 2S_B(C)] KC^T \right\} + \kappa
\]  

(3.11)

where the matrix \(S_B\) is now an implicit function of \(C\). Given the optimal coefficients \(C\), the learned embedding map takes the form \(f(x) = Ck(x)\), where \(k(x) \triangleq [k(x, x_1) \cdots k(x, x_N)]^T\).

In this work, we consider a further variant of Problem 3.11 with the following characteristics: (i) it promotes sparsity both in the kernel expansions of each component function \(f_p\) and the resulting embedding dimension. In this manner, it is hoped that only a subset of the training data will be influential in determining high-quality component functions, as well as that a suitable embedding dimension can be identified as implied by the training data. The desired sparsities can be achieved by penalizing the group LASSO norms of the columns and rows of \(C\) respectively. And, (ii) instead of committing to the use of a single kernel function, whose hyper-parameter must be tuned, we employ a typical MKL approach, where we employ a conic combination of kernel functions, i.e., we let \(k(x, x') = \sum_{\ell=1}^{L} \theta_\ell k_\ell(x, x')\), where \(\theta \in \Omega \triangleq \{\theta \in \mathbb{R}^L : \theta \succeq 0, \|\theta\|_1 \leq 1\}\). In particular, we consider the problem

\[
\min_{C \in \mathbb{R}^{P \times N}, \theta \in \Omega} \mathcal{E}(C, \theta) + R(C)
\]  

(3.12)

where

\[R(C) \triangleq R_c(C) + R_r(C)\]  

(3.13)

\[R_c(C) \triangleq \lambda \|C\|_{2,1}\]  

(3.14)

\[R_r(C) \triangleq \mu \|C^T\|_{2,1}\]  

(3.15)
and where $\lambda, \mu \geq 0$ are penalty parameters that influence the sparsity of $C$’s columns (to identify influential training samples) and rows (to affect the embedding dimension) respectively and the kernel matrix employed within $E(C)$ takes the form $K(\theta) = \sum_{\ell=1}^{L} \theta_{\ell} K_\ell$ and, hence, becomes a function of $\theta$. If we define $c \triangleq \text{vec } C$, then our stress function can be expressed as

$$E(c, \theta) \triangleq c^T [ (KS_A K) \otimes I_P ] c - 2 c^T [ (KS_B (c, \theta) K) \otimes I_P ] c + \kappa$$

(3.16)

where, for simplicity, we defined $K \triangleq K(\theta)$. Again, $E_R(C, \theta)$ is neither convex in $C$, nor in $\theta$. In order to solve this problem, an IM approach is proposed, which hinges on the availability of a suitable, ancillary objective function $\bar{E}(c, \theta|c', \theta')$ that majorizes $E_R(c, \theta)$, i.e., $E_R(c, \theta) \leq \bar{E}(c, \theta|c', \theta')$ for any $(c, \theta), (c', \theta')$ and attaining equality, when $(c', \theta') = (c, \theta)$. One can demonstrate that the function $E$ of Equation (3.16) is majorized by the function $\bar{E}$ as given below

$$\bar{E}(c, \theta|c', \theta') \triangleq c^T [ (KS_A K) \otimes I_P ] c - 2 c^T [ (KS_B (c', \theta') K') \otimes I_P ] c' + \kappa$$

(3.17)

Proof. The majorizer $\bar{E}(c, \theta|c', \theta')$ of $E(c, \theta)$ as given in Equation (3.17) is derived from Equation (3.18) introduced in [12] as the surrogate loss function of Problem 3.8.

$$\bar{E}(F|F') = \text{trace} \{ F^T S_A F \} - 2 \text{trace} \{ F^T S_B (F') F' \} + \text{const}$$

(3.18)

The derivation is proceeded by using the fact that $F = K(C)^T$, and similarly, letting $F' \triangleq K'(C')^T$, where, for simplicity, we defined $K' \triangleq K(\theta')$. The result is derived using vector-matrix algebraic manipulations.
CHAPTER 4: ALGORITHM

Clearly, our regularized stress function $\mathcal{E}_R(c, \theta)$ is majorized by $\tilde{\mathcal{E}}_R(c, \theta|c', \theta') \equiv \tilde{\mathcal{E}}(c, \theta|c', \theta') + R(c)$. The last fact facilitates an IM algorithm to minimize our regularized stress function, $\mathcal{E}_R$. Apart from the historical motivation of using IM algorithms for MDS methods, it is particularly a suitable choice for our problem, through which our non-convex objective function is transformed to a convex one, and, we can use proximal methods inside IM algorithm, that can deal with non-differentiability and also enjoys a predictable convergence behavior. A block coordinate descent framework can be applied to decompose Problem 3.12 into two subproblems. Thus, the optimization can be carried out by repeatedly minimizing this cost function with respect to $c$ and $\theta$ in an alternating fashion. The first subproblem which is minimizing $\tilde{\mathcal{E}}_R$ with respect to $c$, while holding $\theta$ fixed to $\theta'$, yields a convex, yet non-differentiable problem due to the non-smooth nature of the regularizers we employ. Nevertheless, this sub-problem nicely fits the conditions taken into account in [47], which introduced the Accelerated Proximal Average method. Proximal algorithms can be considered as one of the standard tools to solve constrained, non-smooth and large-scale or distributed versions of optimization problems[33]. The proximal mapping (or proximal operator) of a convex function $h$ scaled with parameter $\rho$ is defined as:

$$\text{prox}_{\rho h}(x) \triangleq \arg \min_u \left[ h(u) + \frac{\|u - x\|^2}{2\rho} \right]$$

(4.1)

In general, to solve a minimization problem using a proximal gradient method, the objective is split into two terms $f$ and $g$, one of which ($f$ in Equation (4.2)) should be differentiable, but both convex. Then the proximal gradient update is given as

$$x^{k+1} \triangleq \text{prox}_{\rho g}(x^k - \lambda^k \nabla f(x^k))$$

(4.2)
where, \( \lambda^k > 0 \) is the step size for the \( k^{th} \) iteration.

To obtain a desired proximal operator, one needs to solve Equation (4.1). In our problem setting, the regularization is the sum of two overlapping group-LASSO norms, whose closed-form proximal operator is hard to obtain. According to [47], for a composite minimization problem of the form Equation (4.3), one can “naively” assume that the proximal map is linear and, therefore, can approximate it by the weighted sum of proximal operators of each summation term, \( f_k \), as shown in Equation (4.4).

\[
\min_x \ell(x) + \bar{f}(x) \tag{4.3}
\]

where \( \bar{f}(x) \triangleq \sum_{k=1}^{K} \alpha_k f_k(x) \).

\[
\text{prox}_{\rho \bar{f}}(x) \approx \sum_{k=1}^{K} \alpha_k \text{prox}_{\rho f_k}(x) \tag{4.4}
\]

The two main assumptions here are that, each \( f_k \) is convex and \( L_k \)-Lipschitz continuous, and that each corresponding proximal map can be computed “easily”. Using this average proximal operator, a fast scheme, namely an Accelerated Proximal Gradient (APG) method [4], can be utilized to solve Problem 4.3. For an accuracy of \( \epsilon > 0 \) and a choice of \( \rho = \min \{1/L_0, 2\epsilon/\bar{L}^2\} \), after at most \( \sqrt{2/\rho \epsilon} \|x_0 - x\| \) steps, the output of APG algorithm, say \( \tilde{x} \), satisfies \( \ell(\tilde{x}) + \bar{f}(\tilde{x}) \leq \ell(x) + \bar{f}(x) + 2\epsilon \), where \( L_0 \) is the Lipschitz constant of \( \nabla \ell \) and \( \bar{L} \) is defined as \( \sum_{k=1}^{K} \alpha_k L^2_k \).

In particular, the proximal operators of \( R_c \) and \( R_r \) for the \( n^{th} \) column, \( C_{\cdot, n} \), and \( p^{th} \) row, \( C_{p, \cdot} \), of \( C \)
respectively have a closed form solution [34] as:

\[
(\text{prox}_{\rho R_c}(C))_n = [1 - \lambda \rho/\|C_n\|_2^2]_+ C_n
\]  

\[
(\text{prox}_{\rho R_r}(C))_p = [1 - \mu \rho/\|C_p\|_2^2]_+ C_p
\]  

where, \([\cdot]_+ \triangleq \max\{\cdot, 0\}\) denotes the hinge function. This implies that the proximal average for the regularizer \(R\) is computed as

\[
\text{prox}_{\rho R}(C) = \frac{(\text{prox}_{\rho R_c}(C) + \text{prox}_{\rho R_r}(C))}{2}
\]  

Using the previous operator, the APG method can be applied as shown in Algorithm 2, to optimize the majorizer of the regularized stress for \(c\). The gradient of the \(\bar{E}\) function with respect to \(c\), used in step 6 of Algorithm 2, is calculated through the equation below:

\[
\nabla \bar{E}(c, \theta | c', \theta') = 2 \left[ (\text{KS}_A K) \otimes I_P \right] c - 2 \left[ (\text{KS}_B (c', \theta') K') \otimes I_P \right] c'
\]  

where \(c'\) and \(\theta'\) are the parameter values derived in the previous iteration of the IM algorithm.

It can be shown that in our case, \(L_0 \leq \mathcal{O}(N^3)\), \(L_1 \leq \mathcal{O}(\sqrt{N})\) and \(L_2 \leq \mathcal{O}(\sqrt{P})\), where \(L_0\), \(L_1\), and \(L_2\) correspond to the Lipschitz constants of \(\nabla \bar{E}\), \(R_c\), and \(R_r\), respectively. Following the convergence analysis in [4], \(\rho\) should be chosen as

\[
\min\{1/L0, 2\epsilon/\bar{L}^2\} = \min\{1/N^3, 4\epsilon/(N + P)\}
\]  

Since typically \(N >> P\), so \(\rho = 1/N^3\). Hence, maximum number of iterations for Algorithm 2 to be satisfy the convergence condition is \(\mathcal{O}(N^{3/2})\). Also note that, each iteration of Algorithm 2 requires \(\mathcal{O}(NP)\) time complexity. Thus, the over all complexity of the first block minimization
can be expressed by $O(N^{5/2}P)$.

On the other hand, the second block is to minimize the regularized stress with respect to $\theta$, by holding the values of the coefficients in $c$ fixed. If $\theta'$ denotes the MKL coefficients obtained from a previous iteration of our algorithm, then one has to consider solving the following convex optimization problem:

\[
\min_{\theta \in \Omega} \theta^T G \theta - 2\theta^T h'
\]

(4.10)

\[
G \in \mathbb{R}^{L \times L} : g_{i,j} \triangleq c^T [(K_i S_A K_j) \otimes I_P] c
\]

(4.11)

\[
h' \in \mathbb{R}^L : h'_i \triangleq c^T [(K_i S_B(c, \theta') \otimes K') \otimes I_P] c
\]

(4.12)

As the previous problem does not have a closed-form solution, it has to be solved iteratively. In our implementation we used the \texttt{cvx} toolkit [11, 16] for that purpose. In particular, the \textit{SeDuMi} [41] solver has been utilized, which for a fixed accuracy $\epsilon$, stops after at most $O(\sqrt{L \log(1/\epsilon)})$ iterations.

**Algorithm 1** Iterative Majorization for solving Problem 3.12

**Input:** $c_{\text{init}} \in \mathbb{R}^{NP}$, $\theta_{\text{init}} \in \Omega$

**Output:** $c_{\text{final}}, \theta_{\text{final}}$

1: $t \leftarrow 1$
2: loop
3: $c_t \leftarrow \arg \min_c \tilde{E}_R(c, \theta_{t-1} | c_{t-1}, \theta_{t-1})$ via Alg. 2 // Perform IM step w.r.t. $c$
4: $\theta_t \leftarrow \arg \min_{\theta \in \Omega} \tilde{E}_R(c_t, \theta | c_t, \theta_{t-1})$ // Perform IM step w.r.t. $\theta$
5: if converged then
6: $c_{\text{final}} \leftarrow c_t$, $\theta_{\text{final}} \leftarrow \theta_t$
7: break
8: end if
9: $t \leftarrow t + 1$
10: end loop
11: return $c_{\text{final}}, \theta_{\text{final}}$
Algorithm 2 Accelerated Proximal Average-Proximal Gradient used in Algorithm 1

Input: $c_{t-1}, \rho$

Output: $c_t$

1: $c_{\text{trial}}(0) \leftarrow c_{t-1}$
2: $y_{\text{trial}}(1) \leftarrow c_{\text{trial}}(0)$
3: $\eta_{\text{trial}}(0) \leftarrow 1$
4: $l \leftarrow 1$
5: loop
6: $z_{\text{trial}}(l) \leftarrow y_{\text{trial}}(l) - \rho \nabla \bar{E}(c, \theta | c', \theta') | c' = y_{\text{trial}}(l)$
7: $c_{\text{trial}}(l) \leftarrow \text{prox}_{\rho R}(z_{\text{trial}}(l))$
8: $\eta_{\text{trial}}(l+1) \leftarrow \frac{1 + \sqrt{1 + 4\eta_{\text{trial}}(l)^2}}{2}$
9: $y_{\text{trial}}(l + 1) \leftarrow c_{\text{trial}}(l) + (c_{\text{trial}}(l) - c_{\text{trial}}(l - 1))\frac{\eta_{\text{trial}}(l) - 1}{\eta_{\text{trial}}(l+1)}$
10: $l \leftarrow l + 1$
11: end loop
12: return $c_t$
CHAPTER 5: EXPERIMENTS

In order to assess the performance of the proposed method, a series of experiments are conducted on both artificial datasets, as well as real-world data. The artificial datasets are chosen since their characteristics and perfect embeddings are known in advance, and this enables us to draw sound conclusions of how our model operates. The benchmark datasets are chosen to evaluate the model on real-world data. We designed 4 sets of experiments to showcase characteristics of the proposed method.

In the first set of experiments, we visualize the embedding results in 2D or 3D and analyze the structural properties of the embeddings. In addition to the visualization of the data, the embedding results allow us to discover meaningful structures in the data. Our findings match the so-called manifold hypothesis ([30], [8]) which theorizes that, although high-dimensional data with complicated structures may appear to lie in a high dimensional space in their original representation, these data points are actually samples from low dimensional sub-manifolds embedded in that high-dimensional space.

The second set of experiments examines the MKL capabilities of the framework, through which optimized values of kernel coefficients are determined. This feature assures the best subset of a pre-specified ensemble of kernels that obtains the lowest stress value. This alleviates the need for an engineered choice of a suitable kernel, requiring domain knowledge, and also the computational cost of tuning kernel hyper-parameters. We present some examples from this set of experiments accompanied with appropriate discussion points.

In the third set of experiments, we investigate the impact of the first regularizer, $R_c$. As mentioned earlier, $R_c$ corresponds to the column norms of matrix $C$, and thus affects the contribution of each data point in the embedding. Higher values of $\lambda$ weigh this term more in the objective function,
and hence it results in a more accentuated effect of the influential samples. Influence of a sample is defined here as the magnitude of the corresponding column norm of the model parameter matrix \( C \). Evolution of the embedding in a range of this parameter \( \lambda \), shows that boundary points play an influential role in the embedding. Consequently, there is no direct relation between \( N \), number of training samples, and the performance of the proposed method, in the sense that, if the added training points are the ones that will get mapped to the boundary points of the embedded structure, it will affect the embedding considerably, but if no boundary points are added, there would not be a notable difference in the corresponding embedding. Increased values of \( \lambda \) cause less number of samples to be considered as influential points, and forces the data to be embedded closer to the interior samples of the dataset. Interior samples are the most similar ones to all others, on average, while corner/edge points are the most different ones, with respect to the rest. When you want to use less data points for embedding, common sense suggests to embed the points further from the edge points, and closer to the most similar samples. Our method reaffirms this intuition and maps the data closer to the central points, when a large value of \( \lambda \) is used.

In the fourth set of experiments, we study the effect of the second regularizer, \( R_r \), which controls the embedding dimension. We expect that increased values of \( \mu \), the parameter for this regularization, will enforce a lower-dimensional embedding, different than the explicit embedding dimension set by the user. For example, one can initially set the embedding dimension to 4, but a large value of \( \mu \) may impose the embedding result to lie in a 2D space. We also showcase how our model discovers the intrinsic dimension of the data by identifying the suitable embedding dimension. As in the first part, these results agree with the manifold hypothesis, inferring lower dimensions as the suitable embedding dimension for high-dimensional datasets.

As mentioned before, this work is supposed to be an exploratory data analysis tool in various scientific fields of research. Reasonably, the quality of the produced embedding can only be assessed within the context of a particular application domain, such as natural language processing and cog-
nitive sciences [42, 36, 40]. In other words, one can have an embedding with low stress values, but it might not convey a meaningful message for the proficients, and hence, comparing different methods according to just the stress values would not be helpful in this context. On the other hand, if we were to look into the other methods having different objective functions, then it would not be a like-to-like comparison, and one cannot drive any well-reasoned conclusion of this comparison.

In all the experiments, except for the Scientific dataset, an ensemble of 8 Gaussian kernels with spreads $s = 0.01, 0.1, 1, 2, 3, 4, 5, 6$ is used for MKL. For Scientific dataset, a different kernel, suitable to the nature of the data is used, which will be explained more in the corresponding part. 30% of the data is used as a validation set to determine the values of the regularization parameters $\lambda$, and $\mu$; The possible values of these parameters are chosen over the set of \{0, $10^i; i = -2, -1, 0, 0.3, 0.6, \ldots, 3$\} via cross-validation. In all reported results, except the Grid2D dataset, $P$ is set to 3, to allow for a 3-dimensional visualization. For the Grid2D dataset, $P = 2$. In practice, the most-often encountered choice for $U$ is

$$U_1 \triangleq 1_N 1_N^T - I_N$$

and is referred to as the all-ones case; In all of the experiments, $U$ is set to $U_1$, so that all discrepancies between pairwise distances in the embedding space and the pairwise dissimilarities in the native space are deemed equally important. In reality, one can choose different $U$ matrices based on domain knowledge about the data and the problem at hand. This parameter can also be viewed as the confidence level of fidelity of pairwise dissimilarities. For instance using $U_1$ as the $U$ matrix, indicates that the user has equal confidence on all pairwise dissimilarities.

We evaluate the performance of our model on both artificial and benchmark datasets described in the following section. Dissimilarities are chosen based on each dataset’s characteristics.
5.1 Artificial Datasets

The *Grid2D* dataset is chosen as it is 2-dimensional with implicit dimensionality of 2, therefore, its resulting embedding can be visualized. This dataset consists of 64 samples located on a regular 2-dimensional grid as shown in Figure 5.1a; the training samples are indicated as red markers, while the test samples appear as continuous blue lines. The dataset was further processed by transforming it into 6-dimensional dataset, using zero padding and a subsequent application of a random rotation matrix in order to test the embedding result. The dissimilarity is defined as the pairwise Euclidean distance in the 6-dimensional native space. Hence the resulting embedding is from an Euclidean space to another Euclidean space.

The *SwissRoll* is a 3-dimensional dataset and is obtained by equally-spaced sampling of an arithmetic spiral surface, whose cross-section for any height is given by the parametric equation $[x_1, x_2] = [\phi \cos \phi, \phi \sin \phi]$, where $0 \leq \phi \leq 4\pi$. For each of 5 different height values, 20 training points were sampled, such that consecutive points have equal arc-length of $\alpha \triangleq 4.0410$ (see Figure 5.1b). The height values are chosen as multiples of $\alpha$. An additional 500 test points were randomly sampled from the same surface. The training data are depicted with red markers in Figure 5.1b. The pairwise dissimilarities between training samples for this 3-dimensional dataset were calculated as follows: first, an undirected graph was constructed using the $\epsilon$-method, *i.e.* vertices (corresponding to samples) were considered connected by an edge, if their Euclidean distance did not exceed $\epsilon = 5$. This threshold is set such that every point is directly connected to its two neighbors on the spiral and the immediate neighbors above and below. Each existing edge was weighed by inverse of the corresponding pairwise Euclidean distance. Based on the resulting weighted graph, the shortest graph distance for each pair of training samples was calculated as an estimate of the geodesic distance between them. *SwissRoll* is a classical benchmark dataset for visualization, and also a good example where we can use the geodesic distances on the original manifold as dissimilarities.
The *Sphere* dataset consists of 30 training points, chosen randomly, lying on a unit sphere with parametric equations \([x_1, x_2, x_3] = [\cos \theta \sin \phi, \sin \theta \sin \phi, \cos \phi]\) (Figure 5.1c). Then 1000 points are randomly selected on the sphere as testing samples. This dataset was also further transformed into 5-dimensional samples by zero-padding and random rotation. The pairwise dissimilarities are taken to be geodesic distances between points on the sphere, for which, the pairwise angle differences of the points are calculated and then multiplied by the radius of the sphere. This distance is also known as *orthodromic* distance. *Sphere* is not topologically flat as *SwissRoll*, and it is considered a closed manifold.
5.2 Real-world Datasets

The *Lenses* dataset is a subset of a database from the UCI repository [26] for fitting contact lenses. In this dataset, patients are classified into 3 different categories: ones who need hard contact lenses, ones who need soft contacts, and ones who do not need contacts. *Lenses* has 24 samples. The 4 attributes of every sample are: the age group of the patient, the spectacle prescription, the astigmatism, and the tear production rate. Dissimilarities are calculated by pairwise Euclidean distances of the data.

The *Newman’s scientific collaboration network* is a co-authorship network of a section of arXive between 1995 and 1999 [32]. We have used the projected one-mode network where, the authors are considered as nodes and the sum of the jointly co-authored papers become weights for the connecting edges. Because of the large number of nodes in the original dataset, we study a smaller part of this data. To have a meaningful partition of this graph data, the Multi-level Modularity Optimization (MMO) method [6] (implemented in Python igraph package [10]) is utilized and two relatively smaller sub-communities having 43 and 121 nodes are chosen. We will refer to these data as Scientific-43, and Scientific-121 throughout the paper. As in the *SwissRoll* case, based on the resulting weighted graph, the shortest graph distance for each pair of training samples is taken as their dissimilarity. As we do not have access to the point configurations in the original space, we have used a kernel suitable for graph data, with the parametric equation of $\gamma^\delta$, where $\delta$ denotes the graph’s pairwise dissimilarities, and $\gamma$ is set to 0.5.

5.3 Experimental Results

To assess the performance of our method, we utilized our algorithm on the aforementioned datasets. Results are presented in the following sections.
5.3.1 Visualization

Figure 5.2 shows the embedding result of the Grid2D, and Sphere dataset when the aforementioned ensemble of Gaussian kernels are used. The obtained embeddings highly resemble the original 2D and 3D versions of these datasets (Figure 5.1a and Figure 5.1c) which shows our embedding preserves the structure of the data. The test samples are well interpolated in both datasets, especially the ones in the middle of the square for Grid2D data, since in that vicinity there are many terms in the kernel expansion that facilitate good interpolation.

![Grid2D data embedding](image)

(a) Grid2D data embedding

![Sphere data embedding](image)

(b) Sphere data embedding

Figure 5.2: Embedding of Grid2D and Sphere datasets; both embeddings resemble their original form, showing that our embedding method preserves the structure in the data.

Figure 5.3 shows the result of the Lenses embedding under similar settings. One can easily recognize that our embedding is helpful to discriminate between different classes.
Figure 5.3: Lenses dataset embedding demonstrates a nice linearly separable structure, mapping the data belonging to different categories to distinct spatial regions.

Similarly, the 3D embedding for the first sub-cluster of the Scientific data with different initializations is displayed in Figure 5.4. The embedding exhibits an interesting structure, in the sense that it consists of author sub-groups shown as lobes emanating from a central point. Our understanding of the Scientific data suggests that these lobes mirror sub-communities in which authors build closer collaboration networks. In network science, the community detection problem refers to the segmentation of a network into some densely inter-connected components. Modularity optimization based algorithms ([31], [6], [35]) are one of the popular methods to tackle these segmentation problems. As used in the preprocessing step, we’ve run MMO method on the Scientific-43 dataset, to get the sub-community membership of the nodes. In Figure 5.5 we code members of each resulted sub-community with different colors on our embedding result previously shown in Figure 5.4b. Notably, these sub-communities, identified by the MMO method, match our earlier detected lobe structures in the embedding. This reaffirms that our 3D visualization facilitates the discoveries of
interesting structures in the data.

Figure 5.4: *Scientific*-43 dataset embedding with different initializations; Both figures show a similar structure of 3 lobes emanating from a central point. Different initializations have been examined to observe the effect of different initial points. Our visualization discovers the same interesting structures in the data, independent of different initialization of the IM algorithm.

The other sub-cluster of this dataset (*Scientific*-121 dataset), exhibits the same data structure, but with more lobes originated from a central point. Figure 5.6 shows two results of the algorithm on this sub-cluster, corresponding to different initializations.
5.3.2 Multiple Kernel Learning

With regards to the MKL capabilities of the framework, a few experiments were conducted that considered two ensembles of kernels for each dataset; The first ensemble consisted of 8 Gaussian kernels with spreads $s = 0.01, 0.1, 1, 2, 3, 4, 5, 6$, and the second one is a collection of polynomial kernels of powers $p = 1, 2, 3, 4$, a triangle kernel of spread $s = 3$ and a Gaussian kernel of spread $s = 6$. No regularization was considered in this part to avoid possible interference in drawing
sound conclusions.

Figure 5.6: Scientific-121 dataset embedding with different initializations. As in Figure 5.4, two different initializations have been showed to avoid local minima. More sub-groups are observed in this bigger sub-cluster of the data, but there is a strong similarity with the subgroups identified for the smaller sub-cluster of the data-43.

Figure 5.7a shows the resulting embedding of the first ensemble for the Grid2D dataset, while Figure 5.7b shows the values of $\theta$; out of 8 kernels the MKL procedure identified only 3 dominant ones. When using the second ensemble, the obtained embedding is shown in Figure 5.8a and the $\theta$ values in Figure 5.8b. It is hardly a surprise that the only overwhelmingly prominent kernel is the linear kernel ($p = 1$), as it effects a linear mapping from the dataset’s 6-dimensional native space onto the plane, thus, reversing its random rotation (a linear mapping). Embedding results for the SwissRoll dataset are shown in Figure 5.9 for the Gaussian kernel ensemble and in Figure 5.10 for the polynomial-triangle-Gaussian kernel ensemble. Of some interest is the latter figure, which shows a prominent role of the quadratic kernel and, to a much lesser extend, of the quartic kernel. The use of a quadratic kernel implies the mapping of the data into a 3-dimensional RKHS ($\mathbb{R}^3$), which allows the points of the SwissRoll to unfold and then being projected as an approximate
grid-form on the plane.

Figure 5.7: *Grid2D* dataset embedding using an ensemble of 8 Gaussian kernels. Our method optimizes the kernel coefficients using the MKL scheme. Only 3 kernels are chosen as dominant ones in the expansion.

Figure 5.8: *Grid2D* dataset embedding using an ensemble of 6 kernels including 4 polynomial kernels, one triangle, and one Gaussian kernel. The optimized values of kernel coefficients reveal that the linear kernel gets a considerable share reversing the dataset’s construction via random rotation. The embedding result shows a significant fidelity to the original form of the dataset.
Figure 5.9: *SwissRoll* dataset embedding using an ensemble of 8 Gaussian kernels. 3 out of 8 kernels have been almost zeroed out through the MKL procedure.

Figure 5.10: *SwissRoll* dataset embedding using an ensemble of 6 kernels including 4 polynomial kernels, one triangle, and one Gaussian kernel. The significance of the quadratic kernel coefficient is reflected as the unfolded structure of the roll in an approximate grid from.

5.3.3 Effect of First Regularizer, $R_c$

To better understand how our algorithm identifies the “influential” samples of data, we increment the $\lambda$ regularizer parameter logarithmically and observe the embedding evolution of the data. In the following figures, we reflect the degree of influence of the training points with different shades of black; the darker the shade, the more influential the sample. Figure 5.11 illustrates how *Grid2D*
evolves with changing $\lambda$ from a small value to a large one. Considering a moderate value of this parameter, Figure 5.11b facilitates some interesting observations. First, the four corner points and the edge points are deemed as the first, and the second most influential samples, respectively. Hence, these boundary points, being the most dissimilar samples with respect to the rest affect the objective function the most. This is commensurate with their influence in the kernel expansion. The influential samples chosen by the model also remind us of the boundary point property of a convex set which, by definition, is a point that can not be constructed with any other points of that convex set. Furthermore, it is also worth noting the movement of the test samples on the edges towards the center, as $\lambda$ increases. For example, by increasing $\lambda$ to 100.00, we can see the outer test samples squeezed inside the grid of Figure 5.11c. This figure along with (Figure 5.11d) also demonstrate that the influence of the points remains intact for a wide range of $\lambda$ values, and the test points are moving towards the direction of the center of the embedded structure. The center of the embedded structure contains the samples that are most similar to each other. As a result, when it comes to embedding, this model accentuates the effect of the samples which are the most similar ones, on average, to all other points of the data. Predictably, with extreme values of this parameter even the boundary points get mapped to the center of the grid as well and all samples collapse into the centroid region.
Figure 5.11: Evolution of Grid2D dataset with increasing $\lambda$. Higher column-norms contribute more to the kernel expansion of the embedding and the corresponding samples are considered as more influential ones. More influential training samples are displayed by darker points, and the less influential ones, are shown as lighter points. The four corner points are the most influential ones, and moving towards the center of the grid decreases the influence of samples. Increasing $\lambda$ pushes more column-norms toward zero, and hence central points look whiter. With a high value of $\lambda$ there are less samples incorporating to the kernel expansion, and this results in an embedding where outer samples are moved towards the center of the embedding, where the most similar samples lie.

As expected, when $\lambda$ increases, the embedding is enforced to use less samples. Consequently, as shown in Figure 5.12 it appears that with higher $\lambda$s the model uses the samples which lie in the great circle of the sphere, and ultimately, the diameter. It’s interesting to observe that in this case, by increasing $\lambda$ the dimension is decreased implicitly, since the important samples lie in a lower-dimensional space. Here, the median of the pairwise Euclidean distances is considered as
the spread value for the Gaussian kernel for simplicity.

![Image](image_url)

Figure 5.12: Evolution of Sphere dataset with increasing $\lambda$. Higher values of $\lambda$ causes less samples to have considerable contributions to the kernel expansion of the embedding. With $\lambda = 100$ the model uses the training samples lying on a great circle of the sphere, and ultimately, the ones on the diameter.

In Figure 5.13 the samples have been shaded with blue tone, but in proportion to the magnitude of their column-norms, and the four most influential data points have been encoded with integer indices for identification purposes. As expected, with this high value of $\lambda$ only few samples are deemed as influential ones and thus are displayed as darker points in the figure. More specifically, the 23rd point on the top left corner, the 25th on the bottom right corner, the 28th on the front edge, and the 10th on the back edge have darker colors, and thus are of higher importance.
Figure 5.13: Embedding of Sphere dataset, with $\lambda = 100$, different view. The samples have been shaded with a blue tone, but in proportion to the magnitude of their column-norms. The four most influential samples are the darker ones accompanied with their corresponding indices. These points are located close to the four vertices and co-vertices of this ellipse-shaped embedding.

Analogously, if we turn to the column-norms of the training data in the Scientific collaboration graph to identify influential samples, we see that the 10 largest values of the column norms are for data points indexed by 6, 5, 7, 35, 37, 38, 36, 43, 39, and 40, respectively (Figure 5.14).
Figure 5.14: *Scientific-43* dataset Bar chart of Column Norms. Higher values indicate more influential samples. The 10 most influential ones are indexed by 6, 5, 7, 35, 37, 38, 36, 43, 39, and 40, respectively.

In a similar fashion as in the *Grid2D* experiment, the most influential samples of the collaboration graph are the points further than a certain radius from the embedding center. Embedding center is calculated as the mean of the embedded points, and two different radii have been examined, one with the 80% of the maximum distance from the mean, which amounts to 7 most important samples, and one with 85% of the maximum distance from the mean, corresponding to top-3 important samples. Figure 5.15 and Figure 5.16 show these embeddings with the highlighted sphere indicating the aforementioned radii. For legibility purposes we have only colored the top-3 and top-7 influential points, and all other training data are shown as open circles.
Figure 5.15: Top-7 important samples of Scientific-43 dataset. As observed before, our visualization located the most influential samples at the boundaries which enables us to identify them by a radius from the embedding center. 7 samples are retrieved when this radius is set to 80% of the maximum distance from the mean. These 7 nodes are all amongst the 10 most influential samples.

This discussion is another confirmation that the most influential samples identified by our model lie exclusively on the fringe of the mapped embedding and we could retrieve both top-3 and top-7 results correctly.

Furthermore, looking into the influential data points more carefully, we observe that these samples are indeed the ones whose sum of pairwise dissimilarities are the highest ones. In other words, the Scientific visualization not only provides us with a node proximity, based on each nodes coordinates, but also reveals the level of their collaboration and thus enables us to infer the most distinguished nodes of the graph. These observations lead us to some interesting interpretations for any network-based data.
Figure 5.16: Top-3 important samples of Scientific-43 dataset. Setting the radius to 85% of the maximum distance from the mean gives us 3 boundary points, which exactly match the first three influential points.

Consider a company whose employees form the nodes of a graph, and their collaborations are reflected by the edges amongst nodes. To draw useful conclusions about the employees, management could be interested in visualizing the data rather than looking at distinct employee records. The visualized data allows one to observe the relations between employees, identifying similarities between them, and monitoring their collaborations.

This visualized representation can in turn suggest interesting policy decisions. As shown in this experiment, the most distinctive employees are identifiable by being located at the boundary of the embedding, and are the ones that have unique skills. If the company’s strategy is in support of its distinct aspects, then the management should retain this group of people, and/or hire other employees with similar unique traits. On the other hand, if a company values more collaborative activities, the management should try to retain the employees whose characteristics place them at
the center of the embedding, and are more collaborative. Again, policy making is totally strategy-dependent, but in all cases these policies can be assisted by meaningful visualizations of the data.

5.3.4 Effect of Second Regularizer, $R_r$

As the second regularization parameter $\mu$ increases, the high row-norm values of $C$ are penalized. This leads the model to practically zero out some rows of the $C$ matrix, and hence decreases the intrinsic dimensionality of the embedding from the preset dimension $P$. Figure 5.17 illustrates the effect of a large $\mu$ on the embedding. As expected, this regularizer controls the embedding dimension, and setting $\mu$ to higher values results in lower dimensions. For a large enough $\mu$ value, the 2D grid structure is embedded as a straight line in a 1-dimensional space.

![Figure 5.17](image)

Figure 5.17: Evolution of Grid2D dataset embedding with increasing $\mu$. Higher value of $\mu$ decreases the dimensionality of the embedding and transforms the grid structure to a straight 1D line.

Similarly, in Figure 5.18 we have studied how increasing $\mu$ affects the embedding of the SwissRoll data. As expected, the larger $\mu$ forces the embedding to be projected into a lower dimension, 2, and in this case we can interestingly observe that the manifold has been adequately unfolded to a flat grid in 2D. When $\mu$ is increased further, the flat rectangular embedding is forced to get
even more compact, eventually ending up being a straight line. It’s noteworthy that our model infers the suitable embedding dimension of this data to be 2, which maps the unfolded form of this manifold in 2-dimensional space. This confirms our claim that the model can capture the intrinsic dimensionality of the data.

Figure 5.18: Evolution of SwissRoll dataset embedding with increasing $\mu$. As $\mu$ increases, the embedding is forced to lie in a lower dimension. $\mu = 100$ has nicely flattened out the roll structure to a 2D grid form. Increasing $\mu$ even more, squeezes the grid form inside, and eventually gets a 1D line.
In a similar fashion, as we increase $\mu$ to 100, the embedding space of the collaboration graph data tends to collapse to lower dimensions in Figure 5.19.

![Figure 5.19: Scientific-43 dataset embedding ($\mu = 100$), different views. As observed before, this embedding shows the effect of second regularization parameter, which decreases the embedding dimension. The 3D visualization has been projected to a 2-dimensional structure.](image)
CHAPTER 6: CONCLUSION

In this paper, we proposed a novel Sparse Kernel-based Least-Squares MDS approach, which assumes the embedding to belong to a RKHS of vector-valued functions, yielding a solution that is not a point configuration, but a vector-valued function, capable of embedding unseen out-of-sample data-points. Framing the formulation as a MKL problem, enables us to infer the best ensemble kernel from the available data. We have introduced two regularization terms, of which one is to promote embedding maps, whose appropriate range dimensionality could be estimated by training and the other promotes maps, whose kernel expansions are dominated by key training samples; the importance of these samples are also inferred from the training process. A new IM algorithm has been developed to fit the associated non-convex and non-smooth optimization problem, which introduces significant new computational challenges because of the incorporated overlapping structured sparsity regularization. The derivation of this iterative process hinged on finding a suitable surrogate function that majorizes the framework’s stress function, to obtain a convex objective, and a Proximal operator for the the composite regularizers to deal with the overlapping non-smooth regularization. Finally, an extensive experimentation of the framework on a variety of dimensionality reduction and visualization tasks showcased the benefits of the proposed model including hidden structure discoveries through visualization, inferring an effective RKHS feature map, identification of critical samples, and also inferring a suitable embedding dimension. Future work on this subject will address a parallelization of this method to offer more computational efficiency.
LIST OF REFERENCES


[27] R. B. MARIMONT and M. B. SHAPIRO. Nearest neighbour searches and the curse of


[29] Alistair Morrison, Greg Ross, and Matthew Chalmers. Fast multidimensional scaling through

[30] Hariharan Narayanan and Sanjoy Mitter. Sample complexity of testing the manifold hypoth-


[34] Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in optimiza-

[35] Xinyu Que, Fabio Checconi, Fabrizio Petrini, Teng Wang, and Weikuan Yu. Lightning-fast
community detection in social media: A scalable implementation of the louvain algorithm.
*Department of Computer Science and Software Engineering, Auburn University, Tech. Rep.

[36] Lili Duan Robin L. Pinkley, Michele J. Gelfand. When, where and how: The use of mul-
tidimensional scaling methods in the study of negotiation and social conflict. *International


[40] Mark Steyvers. Multidimensional scaling. ENCYCLOPEDIA OF COGNITIVE SCIENCE.


