A regression approach for explaining manifold embedding coordinates

Marina Meila[*]   Samson Koelle   Hanyu Zhang

Department of Statistics

University of Washington

Seattle, WA 98195-4322

{mmp2,sjkoelle,hanyuz6}@uw.edu

November 30, 2018

Abstract

Manifold embedding algorithms map high dimensional data, down to coordinates in a much lower dimensional space. One of the aims of the dimension reduction is to find the *intrinsic coordinates* that describe the data manifold. However, the coordinates returned by the embedding algorithm are abstract coordinates. Finding their physical, domain related meaning is not formalized and left to the domain experts. This paper studies the problem of recovering the domain-specific meaning of the new low dimensional representation in a semi-automatic, principled fashion. We propose a method to explain embedding coordinates on a manifold as *non-linear* compositions of functions from a user-defined dictionary. We show that this problem can be set up as a sparse *linear Group Lasso* recovery problem, find sufficient recovery conditions, and demonstrate its effectiveness on data.

Manifold learning algorithms, also known as embedding algorithms, map data from high dimensional, possibly infinite dimensional spaces down to coordinates in a much lower dimensional space. In the sciences, one of the goals of dimension reduction is the discovery of descriptors of the data generating process. Both linear dimension reduction algorithms like Principal Component Analysis (PCA) and non-linear algorithms such as Diffusion Maps [CL06] are used in applications from genetics to chemistry to uncover *collective coordinates* describing large scale properties of the interrogated system. For example, in chemistry, a common objective is to discover so-called *reaction coordinates* describing evolution of molecular configurations [CNO00, NC17] .

For example, Figure 1 shows the toluene molecule (C_7H_8), consisting of $N_a = 15$ atoms. By *Molecular Dynamics (MD)* simulation, configurations of this molecule are generated in $D = 3 \times N_a = 45$ dimensions. They are then mapped into $m = 2$ dimensions by a manifold learning algorithm. The figure shows that this configuration space is well approximated by a one-dimensional manifold, and this manifold is parametrized by a geometric quantity, the *torsion* of the bond connecting the CH_3 *methyl* group with the C_6H_5 *benzene* group. Torsions are dihedral angles in tetrahedra with vertices at atoms of the molecule. In a molecule, any two atoms which are not too distant interact; hence the toluene molecule has many more interactions than the graph edges presented in Figure

---
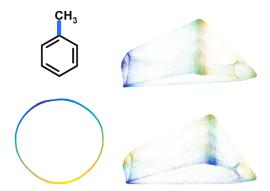
[*]www.stat.washington.edu/mmp

Figure 1: Explaining the collective coordinates in Molecular Dynamics (MD) simulations. In such simulations, a molecular configuration is represented by the $D = 3N_a$ vector of spatial locations of the N_a atoms comprising the molecule (not shown). Left, top: The toluene molecule, with the bond defining the important torsion τ marked in blue. Left, bottom: Embedding of the configurations of toluene into $m = 2$ dimensions, showing a manifold of $d = 1$. The color corresponds to the values of τ. Right: Embedding of the configurations of the ethanol (C_2H_5OH) in $m = 2$ dimensions, showing a manifold of $d = 2$. This manifold is colored by two bond torsions in the ethanol molecule; it can be seen that each torsion explains one of the circles generating the torus.

1. Our problem is to select the torsion τ that explains the one-dimensional manifold of toluene configurations out of the many torsions to be considered.

Finding the meaning behind the coordinates output by an embedding algorithm through comparison to problem-relevant covariates is usually done via visual inspection by human experts. This paper introduces a regression method for automatically establishing such relationships between the abstract coordinates output by a manifold learning algorithm and functions of the data that have meaning in the domain of the problem.

The next section defines the problem formally, Section 2 presents useful background in manifold estimation, while Sections 3, 4 and 5 develop our method. Section 3 is of independent interest as it introduces *Functional Lasso (*FLasso*)*, a novel method for sparsely representing a function f as a non-linear composition of functions from a dictionary. Section 4 introduces a new differential-geometric method for estimating the gradient of a function defined on a manifold. These two methods are combined to obtain our main algorithm, ManifoldLasso in Section 5. The relationship to previous work is discussed in Section 6. Section 8 presents experiments with FLasso and ManifoldLasso, respectively. Recovery results are presented in Section 7.

1   Problem formulation, assumptions and challenges

We make the standard assumption that the observed data $\mathcal{D} = \{\xi_i \in \mathbb{R}^D : i \in 1 \ldots n\}$ are sampled i.i.d. from a *smooth* manifold [1] $\mathcal{M}$ of intrinsic dimension d *smoothly embedded* in $\mathbb{R}^D$ by the inclusion map. In this paper, we will call *smooth* any function or manifold of class at least $\mathcal{C}^3$. We assume that the intrinsic dimension d of $\mathcal{M}$ is known; for example, by having been estimated previously

---

[1]The reader is referred to [Lee03] for the definitions of the differential geometric terms used in this paper.

by one method in [KvL15]. The manifold $\mathcal{M}$ is a *Riemannian manifold* with *Riemannian metric* inherited from the ambient space $\mathbb{R}^D$. Furthermore, we assume the existence of a smooth *embedding map* $\phi : \mathcal{M} \to \phi(\mathcal{M}) \subset \mathbb{R}^m$, where typically $m << D$. We call the coordinates $\phi(\xi_i)$ in this m dimensional ambient space the *embedding coordinates*; let $\Phi = [\phi(\xi_i)^T]_{i=1:n} \in \mathbb{R}^{n \times m}$. In practice, the mapping of the data $\mathcal{D}$ onto $\phi(\mathcal{D})$ represents the output of an embedding algorithm, and we only have access to $\mathcal{M}$ and ϕ via $\mathcal{D}$ and its image $\phi(\mathcal{D})$.

In addition, we are given a *dictionary* of user-defined and domain-related smooth functions $\mathcal{G} = \{g_1, \dots g_p$, with $g_j : \mathbb{R}^D \to \mathbb{R}\}$. Our goal is to express the embedding coordinate functions $\phi_1 \dots \phi_m$ in terms of functions in $\mathcal{G}$.

More precisely, we assume that $\phi(x) = h(g_{j_1}(x), \dots g_{j_s}(x))$ with $h : O \subseteq \mathbb{R}^s \to \mathbb{R}^m$ a smooth function of s variables, defined on a open subset of $\mathbb{R}^s$ containing the range of $g_{j_1}, \dots g_{j_s}$. Let $S = \{j_1, \dots j_s\}$, and $g_S = [g_{j_1}(x), \dots g_{j_s}]^T$. The problem is to discover the set $S \subset [p]$ such that $\phi = h \circ g_S$. We call S the *functional support* of h, or the *explanation* for the manifold $\mathcal{M}$ in terms of $\mathcal{G}$. For instance, in the toluene example, the functions in $\mathcal{G}$ are all the possible torsions in the molecule, $s = 1$, and $g_S = \tau$ is the explanation for the 1-dimensional manifold traced by the configurations.

Indeterminacies In differential geometric terms, the explanation g_S is strongly related to finding a *parametrization* of $\mathcal{M}$. Hence, $|S| \geq d$. Since the function ϕ given by the embedding algorithm is not unique, the function h cannot be uniquely determined. Moreover, whenever $g_1 = t(g_2)$ where t is a smooth monotonic function, the support S may not be unique either.

In Section 7 we give sufficient and necessary conditions under which S can be recovered uniquely; intuitively, they consist of functional independencies between the functions in $\mathcal{G}$. For instance, it is sufficient to assume that that the dictionary $\mathcal{G}$ is a *functionally independent* set, i.e. there is no $g \in \mathcal{G}$ that can be obtained as a smooth function of other functions in $\mathcal{G}$.

Hence, in this paper we resolve the indeterminacies w.r.t. the support S, and we limit our scope to recovering S. We leave to future work the problem of recovering information on how the functions g_S combine to explain $\mathcal{M}$.

2 Background on manifold estimation: neighborhood graph, Laplacian and estimating tangent subspaces

Manifold learning and intrinsic geometry Suppose we observe data points $\xi_i \in \mathbb{R}^D$ that are sampled from a smooth d-dimensional submanifold $\mathcal{M} \subset \mathbb{R}^D$. The task of manifold learning is to provide a diffeomorphism $\phi : \mathcal{M} \to \phi(\mathcal{M}) \subset \mathbb{R}^m$ where $m \ll D$. The Whitney Embedding Theorem [Lee03] guarantees the existence of a map satisfying this property with $m \leq 2d$. Hence, a good manifold learner will identify a smooth map $\phi : \mathcal{M} \to \mathbb{R}^m$ with $d \leq m \leq 2d \ll D$.

The neighborhood graph and kernel matrix The *neighborhood graph* is a data structure that associates to each data point $\xi_i \in \mathcal{D}$ its set of *neighbors* $\mathcal{N}_i = \{i' \in [n], \text{with} \, ||\xi_{i'} - \xi_i|| \leq r_N$, where r_N is a *neighborhood radius* parameter. The neighborhood relation is symmetric, and determines an undirected graph with nodes represented by the data points $\xi_{1:n}$.

Closely related to the neighborhood graph are the local position matrices $\Xi_i = \{\xi_{i'} : i' \in \mathcal{N}_i\} \in \mathbb{R}^{|\mathcal{N}_i| \times d}$, local embedding coordinate matrices $\Phi_i = \{\phi(\xi_{i'}) : i' \in \mathcal{N}_i\} \in \mathbb{R}^{|\mathcal{N}_i| \times d}$, and the *kernel*

matrix $K \in \mathbb{R}^{n \times n}$ whose elements are

$$K_{ii'} = \begin{cases} \exp\left(-\frac{||\xi_i - \xi_{i'}||}{\epsilon_N^2}\right) & \text{if } i' \in \mathcal{N}_i \\ 0 \text{ otherwise.} \end{cases} \tag{1}$$

Typically, the radius r_N and the *bandwidth* parameter ϵ_N are related by $r_N = c\epsilon_N$ with c a small constant greater than 1. This ensures that K is close to its limit when $r_N \to \infty$ while remaining sparse, with sparsity structure induced by the neighborhood graph. Rows of this matrix will be denoted $K_{i,\mathcal{N}_i}$ to emphasize that when a particular row is passed to an algorithm, only $|\mathcal{N}_i|$ values need to be passed. The neighborhood graph, local position matrices, and kernel matrix play crucial roles in our manifold estimation tasks.

Estimating tangent spaces in the ambient space $\mathbb{R}^D$ The tangent subspace at point ξ_i in the data can be estimated by *Weighted Local Principal Component Analysis*, as described in the LOCALPCA algorithm. The output of this algorithm is an orthogonal matrix $T_i \in \mathbb{R}^{D \times d}$, representing a basis for $\mathcal{T}_{\xi_i}\mathcal{M}$. For this algorithm and others we define the SVD algorithm $\text{SVD}(X, d)$ of a symmetrical matrix X as outputting V, Λ, where Λ and V are the largest d eigenvalues and their eigenvectors, respectively, and denote a column vector of ones of length k by $\mathbf{1}_k$. We denote $|\mathcal{N}_i|$ by k_i.

LOCALPCA   (local data Ξ_i, kernel row $K_{i,\mathcal{N}_i}$, intrinsic dimension d)

1: Compute weighted mean $\bar{\xi}_i = (K_{i,\mathcal{N}_i}\mathbf{1}_{k_i})^{-1}K_{i,\mathcal{N}_i}\Xi_i$
2: Compute weighted local difference matrix $Z_i = (K_{i,\mathcal{N}_i}\mathbf{1}_{k_i})^{-1}K_{i,\mathcal{N}_i}(\Xi_i - \mathbf{1}_{k_i}\bar{\xi}_i)$
3: Compute $T_i, \Lambda \leftarrow \text{SVD}(Z_i^T Z_i, d)$
4: **Output** T_i

The renormalized graph Laplacian   The *renormalized graph Laplacian*, also known as the *sample Laplacian*, or *Diffusion Maps Laplacian L*, constructed by LAPLACIAN , converges to the manifold Laplace operator $\Delta_{\mathcal{M}}$; [CL06] shows that this estimator is unbiased w.r.t. the sampling density on $\mathcal{M}$ (see also [HAvL05, HAvL07, THJ10]). The Laplacian L is a sparse matrix; row i of L contains non-zeros only for $i' \in \mathcal{N}_i$. Thus, as for K, rows of this matrix will be denoted $L_{i,\mathcal{N}_i}$. Again, sparsity pattern is given by the neighborhood graph; construction of the neighborhood graph is therefore the computationally expensive component of this algorithm.

LAPLACIAN   (neighborhoods $\mathcal{N}_{i:n}$, local data $\Xi_{1:n}$, bandwidth ϵ_N)

1: Compute kernel matrix K using (1)
2: Compute normalization weights $w_i \leftarrow K_{i,\mathcal{N}_i}\mathbf{1}_{k_i}$, $i = 1, \ldots n$, $W \leftarrow \text{diag}(w_i \ i = 1 : n)$
3: Normalize $\tilde{L} \leftarrow W^{-1}KW^{-1}$
4: Compute renormalization weights $\tilde{w}_i \leftarrow \tilde{L}_{i,\mathcal{N}_i}\mathbf{1}_{k_i}$, $i = 1, \ldots n$, $\tilde{W} = \text{diag}(\tilde{w}_i \ i = 1 : n)$
5: Renormalize $L \leftarrow \frac{4}{\epsilon_N^2}(\tilde{W}^{-1}\tilde{L} - I_n)$
6: **Output** Kernel matrix K, Laplacian L, [optionally $\tilde{w}_{1:n}$]

The m principal eigenvectors of the Laplacian L (or alternatively of the matrix $\tilde{L}$ of Algorithm LAPLACIAN), corresponding to its smallest eigenvalues, are sometimes used as embedding coordinates

Φ of the data; the embedding obtained is known as the *Diffusion Map* [CL06] or the *Laplacian Eigenmap* [BN02] of $\mathcal{D}$. We use this embedding approach for convenience, but in general, any algorithm which asymptotically generates a smooth embedding is acceptable.

The pushforward Riemannian metric Geometric quantities such as angles and lengths of vectors in the tangent bundle $\mathcal{TM}$, distances along curves in $\mathcal{M}$, etc., are captured by Riemannian geometry. We assume that $(\mathcal{M}, \mathbf{id})$ is a *Riemannian manifold*, with the metric $\mathbf{id}$ induced from $\mathbb{R}^D$. Furthermore, we associate with $\phi(\mathcal{M})$ a Riemannian metric $\mathbf{g}$ which preserves the geometry of $(\mathcal{M}, \mathbf{id})$. This metric is called the *pushforward Riemannian metric* and is defined by

$$\langle u, v \rangle_{\mathbf{g}} \;=\; \langle D\phi^{-1}(\xi)u, D\phi^{-1}(\xi)v \rangle \quad \text{for all } u, v \in \mathcal{T}_{\phi(\xi)}\phi(\mathcal{M}). \tag{2}$$

In the above, $D\phi^{-1}(\xi)$ maps vectors from $\mathcal{T}_{\phi(\xi)}\phi(\mathcal{M})$ to $\mathcal{T}_{\xi}\mathcal{M}$, and $\langle , \rangle$ is the Euclidean scalar product.

For each $\phi(\xi_i)$, the associated push-forward Riemannian metric expressed in the coordinates of $\mathbb{R}^m$, is a symmetric, semi-positive definite $m \times m$ matrix G_i of rank d. The scalar product $\langle u, v \rangle_{\mathbf{g}}$ takes the form $u^T G_i v$.

The matrices G_i can be estimated by the algorithm RMETRIC of [PM13]. The algorithm uses only local information, and thus can be run efficiently. For notational simplicity, we refer to the m embedding coordinates of the points in a neighborhood of a point i as Φ_i.

RMETRIC (Laplacian row $L_{i,\mathcal{N}_i}$, local embedding coordinates Φ_i, intrinsic dimension d)

1: Compute centered local embedding coordinates $\tilde{\Phi}_i \leftarrow \Phi_i - \phi(\xi_i)\mathbf{1}_{k_i}^T$
2: $H_{ikk'} = L_{i,\mathcal{N}_i}(\tilde{\Phi}_{i,k} \odot \tilde{\Phi}_{i,k'})$ for $k, k' = 1 : m$. $H_i \leftarrow [H_{ikk'}]_{k,k' \in 1:m}$
3: Compute $V_i, \Lambda_i \leftarrow \text{SVD}(H_i, d)$
4: $G_i \leftarrow V_i \Lambda_i^{-1} V_i^T$.
5: **Output** G_i

---

3 Flasso: sparse non-linear functional support recovery

Our approach for explaining manifold coordinates is predicated on the following general method for decomposing a real-valued function f over a given dictionary $\mathcal{G}$. The main idea is that when $f = h \circ g$, their differentials Df, Dh, Dg at any point are in the *linear* relationship $Df = DhDg$. This is expressed more precisely as follows.

Proposition 1 (Leibniz Rule) *Let $f = h \circ g_S$ with $g_S = [g_1, g_2, \cdots, g_s]^T : \mathbb{R}^D \to \mathbb{R}^s, h : \mathbb{R}^s \to \mathbb{R}$. All maps are assumed to be smooth. Then at every point,*

$$\nabla f \;=\; \frac{\partial h}{\partial g_1}\nabla g_1 + \frac{\partial h}{\partial g_2}\nabla g_2 + \dots \frac{\partial h}{\partial g_s}\nabla g_s, \tag{3}$$

or, in matrix notation

$$Df = DhDg_S.$$

Given samples $\xi_{1:n} \in \mathbb{R}^D$, dictionary $\mathcal{G}$, and function f, we would like to discover that f is a function of g_S without knowing h. The usefulness of this decomposition is that even if f is a non-linear function of $g_{1:s}$, its gradient is a *linear combination* of the gradients $\nabla g_{1:s}$ at every point. That is, proposition 1 transforms the *functional, non-linear* sparse recovery problem into a set of *linear* sparse recovery problems, one for each data point.

Formulating the problem as Group Lasso Let $\nabla f(\xi_i) = y_i$ and $\nabla g_j(\xi_i)/||\nabla g_j(\xi)||_2 = x_{ij}$, for $i = 1 : n$. These vectors in $\mathbb{R}^D$ could be estimated, or available analytically. Then, by (3) we construct the following linear model

$$y_i = \sum_{j=1}^{p} \beta_{ij} x_{ij} + \epsilon_i = \mathbf{X}_i \beta_{i:} + \epsilon_i, \quad \text{for } i = 1 : n. \tag{4}$$

In the above, $\beta_{i:} \in \mathbb{R}^p$ is the vector $[\beta_{ij}]_{j=1:p}$. If there exists some h such that $f = h \circ g_S$ holds, then non-zero β_{ij}s are estimations of $\frac{\partial h}{g_j}(\xi_i)||\nabla g_j(\xi_i)||$ for $j \in S$ and zeros β_{ij} indicate $j \notin S$. Hence, in $\beta_{i:}$, only s elements are non-zero. The term ϵ_i is added to account for noise or model misspecification.

The key characteristic of the functional support that we leverage is that the same set of elements will be non-zero for all i's. Denote by $\beta_j \in \mathbb{R}^n$ the vector $[\beta_{ij}]_{i=1:n}$. Since finding the set $S \subset [p]$ for which $\beta_j \not\equiv 0$, given $y_{1:n}$ and $\mathbf{X}_{1:n}$ is underdetermined when for example $s > d$, we use *Group Lasso* [YL06] regularization in order to zero out as many β_j vectors as possible. In particular, each group has size n. Thus, the support recovery can be reformulated as the globally convex optimization problem

$$(\text{FLasso}) \quad \min_{\beta} J_\lambda(\beta) = \tfrac{1}{2} \sum_{i=1}^{n} ||y_i - \mathbf{X}_i \boldsymbol{\beta}_i||_2^2 + \lambda/\sqrt{n} \sum_j ||\beta_j||, \tag{5}$$

with $\lambda > 0$ a regularization parameter. This framework underscores why we normalize gradients by $||\nabla g_j||_2$; else we would favor dictionary functions that are scaled by large constants, since their coefficients would be smaller. Note that normalization of f is not necessary since it rescales all estimated partials the same. We call problem (5) *Functional Lasso (*FLasso*)*. Experiments on synthetic data in Section 8 illustrate the effectiveness of this algorithm.

Multidimensional FLasso The FLasso extends easily to vector valued functions $f : \mathbb{R}^D \to \mathbb{R}^m$. To promote finding a common support S for all m embedding coordinates, groups will overlap the m embedding coordinates. Let

$$y_{ik} = \nabla f_k(\xi_i) \quad \beta_{ijk} = \frac{\partial h_k}{\partial g_j}(\xi_i) \tag{6}$$

and

$$\beta_j = \text{vec}(\beta_{ijk}, \ i = 1 : n, k = 1 : m) \in \mathbb{R}^{mn}, \quad \beta_{ik} = \text{vec}(\beta_{ijk}, \ j = 1 : p) \in \mathbb{R}^p. \tag{7}$$

Then the FLasso objective function becomes

$$J_\lambda(\beta) = \frac{1}{2} \sum_{i=1}^{n} \sum_{k=1}^{m} ||y_{ik} - \mathbf{X}_i \beta_{ik}||^2 + \frac{\lambda}{\sqrt{mn}} \sum_{j=1}^{p} ||\beta_j||. \tag{8}$$

In the above, β_j is the vector of regression coefficients corresponding to the effect of function g_j, which form group j in the multivariate group Lasso problem. The size of each group is mn.

6

4 Explaining manifold coordinates with FLASSO on the tangent bundle

To explain the coordinates $\phi_{1:m}$ with the FLASSO method of Section 3, we need (i) to extend FLASSO to functions defined on a manifold $\mathcal{M}$, and (ii) to reliably estimate their differentials $D\phi_{1:m}$ in an appropriate coordinate system.

These tasks require bringing various "gradients" values into the same coordinate system. For function g defined on a neighborhood of $\mathcal{M}$ in $\mathbb{R}^D$, we denote $\nabla_\xi g$ the usual gradient $\frac{\partial g}{\partial \xi}$. Similarly for a function f defined on a neighborhood of $\phi(\mathcal{M})$ in $\mathbb{R}^m$, the gradient is denoted $\nabla_\phi f$. To define the gradient of a function on a manifold, we recall that the differential of a scalar function f on $\mathcal{M}$ at point $\xi \in \mathcal{M}$ is a linear functional $Df(\xi) : \mathcal{T}_\xi \mathcal{M} \to \mathbb{R}$. The *gradient* of f at ξ, denoted $\operatorname{grad} f(\xi) \in \mathcal{T}_\xi \mathcal{M}$ is defined by $\langle \operatorname{grad} f(\xi), v \rangle = Df(\xi)v$ for any $v \in \mathcal{T}_\xi \mathcal{M}$. To distinguish between gradients in the tangent bundles of $\mathcal{M}$, respectively $\phi(\mathcal{M})$, we denote the former by $\operatorname{grad}_\mathcal{T}$ and the latter by grad_ϕ.

In this section, we will be concerned with the coordinate functions $\phi_1, \ldots \phi_m$ seen as functions on $\mathcal{M}$, and with the gradients $\nabla_\xi g_j$ and $\operatorname{grad}_{T_i} g_j$ of the dictionary functions. It is easy to see that for a choice of basis $T_i \in \mathbb{R}^{D \times d}$ in $\mathcal{T}_{\xi_i} \mathcal{M}$,

$$\operatorname{grad}_\mathcal{T} g_j(\xi) \;=\; T_i^T \nabla_\xi g_j(\xi_i). \tag{9}$$

4.1 FLASSO for vector-valued functions on $\mathcal{M}$

We start from the FLASSO objective function J_λ for m-dimensional vector valued functions (8). Since differentials of functions on a manifold $\mathcal{M}$ operate on the tangent bundle $\mathcal{T}\mathcal{M}$, the definitions of y_i and $\mathbf{X}_i$ from (FLASSO) are replaced with the values of the respective gradients in $\mathcal{T}\mathcal{M}$. For the moment we assume that these are given. In Section 4.2 we will concern ourselves with estimating the parameters y_{ik} of J_λ from the data, while $\mathbf{X}_i$ is given in (12) below.

For each i, we fix an orthogonal basis in $\mathcal{T}_{\xi_i} \mathcal{M}$, and denote the orthogonal $D \times d$ matrix representing the respective basis vectors by T_i. Hence, any vector in $\mathcal{T}_{\xi_i} \mathcal{M}$ can be expressed as $v \equiv [v_1 \ldots v_d]^T$ by its d coordinates in the chosen basis, or as $T_i v \in \mathbb{R}^D$. For any vector w in $\mathbb{R}^D$, $T_i^T \xi \in \mathbb{R}^d$ represents the coordinates of its projection onto $\mathcal{T}_{\xi_i} \mathcal{M}$; in particular, if $w \in \mathcal{T}_{\xi_i} \mathcal{M}$, $T_i^T T_i w = w$. These elementary identities are reflected in (9) above. Let

$$y_{ik} = \nabla_\mathcal{T} \phi_k(\xi_i) \quad Y_i = [y_{ik}]_{k=1}^m \in \mathbb{R}^{d \times m} \quad \beta_{ijk} = \frac{\partial h_k}{\partial g_j}(\xi_i) \tag{10}$$

and

$$\beta_j = \operatorname{vec}(\beta_{ijk}, \; i = 1:n, k = 1:m) \in \mathbb{R}^{mn}, \quad \beta_{ik} = \operatorname{vec}(\beta_{ijk}, \; j = 1:p) \in \mathbb{R}^p. \tag{11}$$

In the above, the columns of Y_i are the coordinates of $\operatorname{grad}_\mathcal{T} \phi_k(\xi_i)$ in the chosen basis of $\mathcal{T}_{\xi_i} \mathcal{M}$; β_j is the vector of regression coefficients corresponding to the effect of function g_j, which form group j in the multivariate group Lasso problem. The size of each group is mn. By comparing (6) and (7) with (10) and (11) we see, first, the change in the expression of y_{ik} due to projection. We also note that in the definition of β_{ijk} the normalization is omitted. For the present, we will assume that the dictionary functions $g_{1:p}$ have been appropriately normalized; how to do this is deferred to Section 5.

To obtain the matrices $\mathbf{X}_i$, we project the gradients of the dictionary functions g_j onto the tangent bundle, i.e.

$$\mathbf{X}_i \;=\; T_i^T \nabla_\xi g_j(\xi_i)]_{j=1:p} \in \mathbb{R}^{d \times p}. \tag{12}$$

The FLASSO objective function is

$$J_\lambda(\beta) \;=\; \frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{m}||y_{ik} - \mathbf{X}_i\beta_{ik}||^2 + \frac{\lambda}{\sqrt{mn}}\sum_{j=1}^{p}||\beta_j||, \tag{13}$$

an expression identical to (8), but in which $y_i, \mathbf{X}_i$ denote the quantities in (10) and (12). To note that $J_\lambda(\beta)$ is invariant to the change of basis T_i. Inded, let $\tilde{T}_i = T_i\Gamma$ be a different basis, with $\Gamma \in \mathbb{R}^{d\times d}$ a unitary matrix. Then, $\tilde{y}_{ik} = \Gamma^T y_{ik}$, $\tilde{\mathbf{X}}_i = \Gamma^T\mathbf{X}_i$, and $||\tilde{y}_{ik} - \tilde{\mathbf{X}}_i\beta||^2 = ||y_{ik} - \mathbf{X}_i\beta||^2$. The next section is concerned with estimating the gradients y_{ik} from the data.

4.2 Estimating the coordinate gradients by pull-back

Since ϕ is implicitly determined by a manifold embedding algorithm, the gradients of ϕ_k are in general not directly available, and ϕ_k is known only through its values at the data points. We could estimate these gradients naively from differences $\phi_k(\xi_i) - \phi_k(\xi_{i'})$ between neighboring points, but we choose a differential geometric method inspired by [LSW09] and [PM13], that *pulls back* the differentials of $\phi_{1:m}(\xi_i)$ from $\mathbb{R}^m$ to $\mathcal{T}_{\xi_i}\mathcal{M}$, thus enabling them to be compared in the same coordinate system with the derivatives $\mathrm{grad}_\mathcal{T}\, g_j$.

If the range of ϕ were $\mathbb{R}^m$, then the gradient of a coordinate function ϕ_k would be trivially equal to the k-th basis vector, and $\nabla_\phi\phi = I_d$. If $d < m$, by projecting the rows of I_m onto the tangent subspace $\mathcal{T}_{\phi(\xi_i)}\phi(\mathcal{M})$, we get $\mathrm{grad}_{\mathcal{T}_\phi}\,\phi$. We then *pull back* the resulting vectors to $\mathcal{T}_{\xi_i}\mathcal{M}$ to get $\mathrm{grad}_\mathcal{T}\,\phi(\xi_i) \equiv Y_i$, as defined in (10).

Pulling back $\mathrm{grad}_\phi\,\phi = I$ **into** $\mathcal{T}_{\xi_i}\mathcal{M}$ If the embedding induced by ϕ were an isometry, the estimation of $\mathcal{T}_{\phi(\xi_i)}\phi(\mathcal{M})$ could be performed by LOCALPCA, and the pull-back following [PM13]. Here we do *not* assume that ϕ is isometric. The method we introduce exploits the fact that, even when ϕ is not an isometry, the distortion induced by ϕ can be estimated and corrected [PM13]. More precisely, we make use of the RMETRIC algorithm described in Section |refsec:background to estimate for each $\phi(\xi_i)$, the associated push-forward Riemannian metric, expressed in the coordinates of $\mathbb{R}^m$ by G_i, a symmetric, semi-positive definite $m \times m$ matrix of rank d. Additionally, since the theoretical rank of G_i equals d, the d principal eigenvectors of G_i represent (an estimate of) an orthonormal basis of $\mathcal{T}_{\phi(\xi_i)}\phi(\mathcal{M})$.

We now apply (2) to the set of vectors $\mathrm{Proj}_{\mathcal{T}_{\xi_i}\mathcal{M}}(\xi_{i'} - \xi_i)$ for $i' \in \mathcal{N}_i$. Let

$$A_i \;=\; \left[\mathrm{Proj}_{\mathcal{T}_{\xi_i}\mathcal{M}}(\xi_{i'} - \xi_i)\right]_{i'\in\mathcal{N}_i} \in \mathbb{R}^{d\times k_i} \quad B_i = [\phi(\xi_{i'}) - \phi(\xi_i)]_{i'\in\mathcal{N}_i} \in \mathbb{R}^{m\times k_i} \tag{14}$$

and $Y_i = \mathrm{grad}_\mathcal{T}\,\phi(\xi_i) \in \mathbb{R}^{m\times d}$ as defined in the previous section. Then, (2) implies

$$A_i^T Y_i \;\approx\; B_i^T G_i I. \tag{15}$$

The equality is approximate to first order because the $\mathrm{Proj}_{\mathcal{T}_{\xi_i}}\,\mathcal{M}$ operator above is a first order approximation to the *logarithmic map* ; the error tends to 0 with the neighborhood radius [dC92]. If A is full rank d, for $k_i \geq d$, we can obtain Y_i by least-squares as

$$Y_i = (A_i A_i^T)^{-1} A_i B_i^T G_i. \tag{16}$$

This equation recovers $\mathrm{grad}_\mathcal{T}\,\phi(\xi_i)$ in the local basis T_i of $\mathcal{T}_{\xi_i}\mathcal{M}$.

8

If we desired to obtain $\text{grad}_x\, i\phi(\xi_i)$ in the global $\mathbb{R}^D$ coordinates, it suffices to express the columns of A_i as vectors in $\mathbb{R}^D$ (or equivalently to apply the appropriate linear transformation to Y_i).

Algorithm PULLBACKDPHI summarizes the above steps. For the estimation of G_i, the Laplacian must be available. Note also that in obtaining (15) explicit projection on $\mathcal{T}_{\phi(\xi_i)}\phi(\mathcal{M})$ is not necessary, because any component orthogonal to this subspace is in the null space of G_i.

PULLBACKDPHI (basis T_i, local data Ξ_i, local embedding coordinates Φ_i, Laplacian row $L_{i,\mathcal{N}_i}$, intrinsic dimension d

1: Compute pushforward metric $G_i \leftarrow \text{RMETRIC}(L_{i,\mathcal{N}_i}, \Phi_i, d)$
2: Project $A_i = T_i^T(\Xi_i - \xi_i \mathbf{1}_{k_i}^T)$.
3: Compute $B_i = \Phi_i - \phi(\xi_i)\mathbf{1}_{k_i}^T \in \mathbb{R}^{|\mathcal{N}_i| \times m}$.
4: Solve linear system $A_i^T Y_i = B_i^T G_i$ as in (16)
5: **Output** Y_i

---

5 The full MANIFOLDLASSO algorithm

We are now ready to combine the algorithms of Section 2 with the results of Sections 3 and 4 into the main algorithm of this paper. Algorithm MANIFOLDLASSO takes as input data $\mathcal{D}$ sampled near an unknown manifold $\mathcal{M}$, a dictionary $\mathcal{G}$ of functions defined on $\mathcal{M}$ (or alternatively on an open subset of the ambient space $\mathbb{R}^D$ that contains $\mathcal{M}$) and an embedding $\phi(\mathcal{D})$ in $\mathbb{R}^m$. The output of MANIFOLDLASSO is a set S of indices in $\mathcal{G}$, representing the functions $\mathcal{G}$ that explain $\mathcal{M}$.

MANIFOLDLASSO (Dataset $\mathcal{D}$, dictionary $\mathcal{G}$, embedding coordinates $\phi(\mathcal{D})$, intrinsic dimension d, kernel bandwidth ϵ_N, neighborhood cutoff size r_N, regularization parameter λ)

1: Construct $\mathcal{N}_i$ for $i = 1 : n$; $i' \in \mathcal{N}_i$ iff $||\xi_{i'} - \xi_i|| \le r_N$, and local data matrices $\Xi_{1:n}$
2: Construct kernel matrix and Laplacian $K, L \leftarrow \text{LAPLACIAN}(\mathcal{N}_{1:n}, \Xi_{1:n}, \epsilon_N)$
3: [Optionally compute embedding: $\phi(\xi_{1:n}) \leftarrow \text{EMBEDDINGALG}(\mathcal{D}, \mathcal{N}_{1:n}, m, \ldots)$]
4: **for** $i = 1, 2, \ldots n$ *(Prepare gradients for group lasso)* **do**
5:    Compute basis $T_i \leftarrow \text{LOCALPCA}(\Xi_i, K_{i,\mathcal{N}_i}, d)$
6:    Compute $\nabla_\xi g_j(\xi_i)$ for $j = 1, \ldots p$
7:    Project $\mathbf{X}_i \leftarrow T_i^T \nabla_\xi g_{1:p}$
8:    Compute $Y_i \leftarrow \text{PULLBACKDPHI}(T_i, \Xi_i, \Phi_i, L_{i,\mathcal{N}_i}, K_{i,\mathcal{N}_i}, d)$
9: **end for**
10: $\beta, S \leftarrow \text{FLASSO}(\mathbf{X}_{i:n}, Y_{i:n}, \lambda)$
11: **Output** S

Computation The first two steps of MANIFOLDLASSO are construction of the neighborhood graph, and estimation of the Laplacian L. As shown in Section 2, L is a sparse matrix, hence RMETRIC can be run efficiently by only passing values corresponding to one neighborhood at a time. Note that in our examples and experiments, Diffusion Maps is our chosen embedding algorithm, so the neighborhoods and Laplacian are already available, though in general this is not the case.

The second part of the algorithm estimates the gradients and constructs matrices $Y_{1:n}, \mathbf{X}_{1:n}$. The gradient estimation runtime is $O(qd^2 + nd^3)$ where q is the number of edges in the neighborhood graph,

using Cholesky decomposition-based solvers. Finally, the last step is a call to the GROUPLASSO, which estimates the support S of ϕ. The computation time of each iteration in GROUPLASSO is $O(n^2 m^2 pd)$. For large data sets, one can perform the "for" loop over a subset $\mathcal{I} \subset [n]$ of the original data while retaining the geometric information from the full data set. This replaces the n in the computation time with the smaller factor $|\mathcal{I}|$.

Normalization In Group Lasso, the columns of the $\mathbf{X}$ matrix corresponding to each group are rescaled to have unit Euclidean norm. This ensures that the FLASSO algorithm will be invariant to rescaling of dictionary functions by a constant. Since any multiplication of g_j by a non-zero constant, simultaneously with dividing its corresponding β_j by the same constant leaves the reconstruction error of all y's invariant, but affects the norm $||\beta_j||$, rescaling will favor the dictionary functions that are scaled by large constants. Therefore, the relative scaling of the dictionary functions g_j can influence the support S recovered.

When the dictionary functions g_j are defined on $\mathcal{M}$, but not outside $\mathcal{M}$. In this case we follow the standard Group Lasso recipe. We calculate the *normalizing constant*

$$\gamma_j^2 \;=\; \frac{1}{n} \sum_{i=1}^{n} \| \operatorname{grad}_{\mathcal{T}} g_j(\xi_i)\|^2. \tag{17}$$

The above is the finite sample version of $\| \operatorname{grad}_{\mathcal{T}} g_j \|_{L_2(\mathcal{M})}$, integrated w.r.t. the data density on $\mathcal{M}$. Then we set $g_j \leftarrow g_j/\gamma_j$ which has the same effect as the normalization for FLASSO in Section 3.

In the case when the dictionary functions are defined on a neighborhood around $\mathcal{M}$ in $\mathbb{R}^D$, we compute the normalizing constant with respect to $\nabla_\xi g_j$, that is

$$\gamma_j^2 \;=\; \frac{1}{n} \sum_{i=1}^{n} \| \nabla_\xi g_j(\xi_i)\|^2. \tag{18}$$

Then, once again, we set $g_j \leftarrow g_j/\gamma_j$. Doing this favors the dictionary functions whose gradients are parallel to the manifold $\mathcal{M}$, and penalizes the g_j's which have large gradient components perpendicular to $\mathcal{M}$.

Tuning As the tuning parameter λ is increased, the cardinality of the support decreases. Tuning parameters are often selected by cross-validation in Lasso-type problems, but this is not possible in our unsupervised setting. We base our choice of λ on matching the cardinality of the support to d. The recovery results proved in Section 7.2 state that for a single chart, $s = d$ functionally independent dictionary functions suffice.

In manifolds which cannot be represented by a single chart, the situation is more complicated, and it depends on the topology of the co-domains of the functions g_j. For example, in the toluene data presented in Section 8, the manifold is a circle. This cannot be covered by a single chart, but it can be parametrized by a single function in the dictionary, the angle of the bond torsion in Figure 5. Another case when $s > d$ is the case when no single dictionary function can parametrize the manifold. When $\mathcal{G}$ is not a functionally independent set, it is possible that the parametrization of $\mathcal{M}$ by $\mathcal{G}$ is not unique. Hence, in general, the support size s may be equal to d or larger.

5.1   Variants and extensions

The approach utilized here can be extended in several interesting ways. First, our current approach explains the embedding coordinates ϕ produced by a particular embedding algorithm. However, the

same approach can be used to directly explain the tangent subspace of $\mathcal{M}$, independently of any embedding.

Second, one could set up FLASSO problems that explain a single coordinate function. In general, manifold coordinates do not have individual meaning, so it will not be always possible to find a good explanation for a single ϕ_k. However, Figure 6 shows that for the ethanol molecule, whose manifold is a torus, there exists a canonical system of coordinates in which each coordinate is explained by one torsion.

Third, manifold codomains of $\mathcal{G}$ can be used, eliminating certain issues of continuity and chart-counting.

Finally, when the gradients of the dictionary functions are not analytically available, they can also be estimated from data.

6 Related work

To our knowledge, ours is the first solution to estimating a function f as a *non-linear* sparse combination of functions in a dictionary. Below we cite some of the closest related work.

Group Lasso has been widely used in sparse regression and variable selection. In [OPV+14] the *Functional Sparse Shrinkage and (FuSSO)* is introduced to solve similar problem in Euclidean space setting. FuSSO uses Group Lasso to recover F as a sparse *linear* combination of functions from an infinite dictionary given implictly by the decomposition over the Fourier basis. More importantly, this method imposes an additive model between the response and the functional covariates, which is not true in our method.

Gradient Learning [YX12]is also a similar work trying to recover non-zero partial derivatives via Group Lasso type regression as methods of variable selection and dimension reduction. Their work does not have functional covariate like us as input. Moreover, the goal of [YX12] is not explaining the coordinates but instead predicting a response given the covariates.

The *Active Subspace* method of [CDW14] uses the information in the gradient to discover a subspace of maximum variation of f. This subspace is given by the principal subspace of the matrix $C = E_\rho[\nabla f \nabla f^T]$, where ρ is a weighting function averaging over a finite or infinite set of points in the domain of f. While this method uses the gradient information, it can only find a global subspace, which would not be adequate for function composition, or for functions defined on non-linear manifolds.

The work of [BPK16] is similar to ours in that it uses a dictionary. The goal is to identify the functional equations of non-linear dynamical systems by regressing the time derivatives of the state variables on a subset of functions in the dictionary, with a sparsity inducing penalty. The recovered functional equation is *linear* in the dictionary functions, hence any non-linearity in the state variables must be explicitly included in the dictionar. On the other hand, when the functional equation can be expressed as a sum of dictionary functions, then the system is completely identified.

With respect to parametrizing manifolds, the early work of [SR03, TR02] (and references therein) proposes parametrizing the manifold by finite mixtures of local linear models, aligned in a way that provides global coordinates, in a way reminiscent of LTSA[ZZ04].

A point of view different from ours views a set of d eigenvectors of the Laplace-Beltrami operator $\Delta_\mathcal{M}$ as a parametrization of $\mathcal{M}$. Hence, the Diffusion Maps coordinates could be considered such a parametrization [CL06, CLL+05, Gea12]. With [MN17] it was shown that principal curves and surfaces can provide an approximate manifold parametrization. Our work differs from these in two ways (1) first, obviously, the explanations we obtain are endowed with the physical meaning of

the domain specific dictionaries, (2) less obviously, descriptors like principal curves or Laplacian eigenfunctions are generally still non-parametric (i.e exist in infinite dimensional function spaces), while the parametrizations by dictionaries we obtain (e.g the torsions) are in finite dimensional spaces. [DTCK18] tackles a related problem: choosing among the infinitely many Laplacian eigenfunctions d which provide a d-dimensional parametrization of the manifold; the approach is to solve a set of Local Linear Embedding [RS00] problems, each aiming to represent an eigenfunction as a combination of the preceding ones.

7 Uniqueness and recovery results

7.1 Functional dependency and uniqueness of representation

Here we study the conditions under which $f = h \circ g_S$ is uniquely represented over a dictionary $\mathcal{G}$ that contains g_S. Not surprisingly, we will show that these are functional (in)dependency conditions on the dictionary.

We first study when a group of functions on an open $U \subset \mathbb{R}^d$ can be represented with a subset of functionally independent functions. The following lemma implies that if a group of non-full-rank smooth functions has a constant rank in a neighborhood, then it can be showed that locally we can choose a subset of these functions such that the others can be smoothly represented by them. This is a direct result from the constant rank theorem.

Lemma 2 (Remark 2 after Zorich[Zor04] Theorem 2 in Section 8.6.2) *Let $f : U \to \mathbb{R}^m$ be a mapping defined in an neighborhood $U \subset \mathbb{R}^d$ of a point $x \in \mathbb{R}^d$. Suppose $f \in C^\ell$ and the rank of the mapping f is k at every point of a neighborhood U and $k < m$. Moreover assume that the principal minor of order k of the matrix Df is not zero. Then in some neighborhood of $x \in U$ there exist $m - k$ C^ℓ functions $g^i, i = k + 1, \cdots, m$ such that*

$$f^i(x_1, x_2, \cdots, x_d) = g^i(f^1(x_1, x_2, \cdots, x_d), f^2(x_1, x_2, \cdots, x_d), \cdots, f^k(x_1, x_2, \cdots, x_d)) \tag{19}$$

Applying this lemma we can construct a local representation of a subset in g_S. The following classical result in differential geometry lets us expand the above lemma beyond local to global.

We start with a definition. A *smooth partition of unity subordiante to* $\{U_\alpha\}$ is an indexed family $(\phi_\alpha)_{\alpha \in A}$ of smooth functions $\phi_\alpha : M \to \mathbb{R}$ with the following properties:

1. $0 \leq \phi_\alpha(x)$ for all $\alpha \in A$ and all $x \in M$;

2. supp $\phi_\alpha \subset U_\alpha$ for each $\alpha \in A$;

3. Every $\xi \in \mathcal{M}$ has a neighborhood that intersects supp ϕ_α for only finitely many values of α;

4. $\sum_{\alpha \in A} \phi_\alpha(x) = 1$ for all $\xi \in \mathcal{M}$.

Lemma 3 (John 2013[Lee03] Theorem 2.23) *Suppose M is a smooth manifold, and $\{U_\alpha\}_{\alpha \in A}$ is any indexed open cover of M. Then there exists a smooth partition of unity subordinate to $\{U_\alpha\}$*

Now we state our main results.

Theorem 4 *Assume $\mathcal{G}$ and g_S are defined as in Section 3, with $g_{1:p}$ are C^ℓ functions in open set $U \subset \mathbb{R}^d$ and let $S' \subset [p], S' \neq S, |S'| < d$ be another subset of C^ℓ functions. Then there exists a C^ℓ mapping $\tau : \mathbb{R}^{s'} \to \mathbb{R}^s$ on U such that $g_S = \tau \circ g_{S'}$ iff*

$$\text{rank}\begin{pmatrix} Dg_S \\ Dg_{S'} \end{pmatrix} = \text{rank}\, Dg_{S'} \ \text{on}\ U \tag{20}$$

Proof If $Dg_{S'}$ is not full rank on U, we replace S' with a subset of S' so that $Dg_{S'}$ is full rank globally on U. The existence of such subsets are guaranteed by step-wise eliminating functions in $g_{S'}$, which will result in a zero matrix in r.h.s if such a subset does not exist and thus leads to a contradiction. Since $s' = |S'| \leq d$, rank $Dg_{S'} = s'$. Let

$$g_{S' \cup S}(\xi) = \begin{pmatrix} g_{S'}(\xi) \\ g_S(\xi) \end{pmatrix} \tag{21}$$

and $Dg_{S' \cup S}$ denote the l.h.s. matrix in (20). When the rank of $Dg_{S \cup S'}$ equals the rank of $D_{g_{S'}}$, according to Lemma 2, there exists some neighborhood $U_x \in \mathbb{R}^d$ of x and C^ℓ functions $\tau_x^i, i = s'+1, s'+2, \cdots, s'+s$

$$g_{S' \cup S}^i(\xi) = \tau_x^i(g_1(\xi), \cdots, g_{s'}(\xi)), \ \text{for}\ i = s'+1, s'+2, \cdots, s'+s, \ \xi \in U_x \tag{22}$$

Here we should notice that τ_x^i is defined only on a neighborhood of x. We denote such a neighborhood by U_x and then since this holds for every $x \in U$, therefore we can find an open cover $\{U_x\}$ of the original open set U. Since each open set in $\mathbb{R}^d$ is a manifold, the classical result of partition of unity in Lemma 3 holds, that U admits a smooth partition of unity subordinate to the cover $\{U_x\}$. We denote this partition of unity by $\phi_x(\cdot)$.

Hence we can define

$$\tau^i(\xi) = \sum_x \phi_x(\xi)\tau_x^i(g_{j_1}(\xi), \cdots, g_{j_{|S \cup S'|}}(\xi)), \tag{23}$$

where the functions in $g_{S \cup S'}$ are taken without repetition. For each $x \in U$, the product $\phi_x \tau_x^i$ is C^ℓ on the neighborhood U_x. According to the properties of partition of unity, this is a locally finite sum, which means that we do not have to deal with the problem of convergence. Also this will be a C^ℓ function.

Therefore, globally in U we have

$$g_{S \cup S'}^i(\xi) = \tau^i(g_1(\xi), \cdots, g_{s'}(\xi)), \ \text{for}\ i = s'+1, s'+2, \cdots, s'+s, \ \xi \in U \tag{24}$$

Now we prove the converse implication. If rank $Dg_{S \cup S'} > s'$, then there is $j \in S$, so that $Dg_j \notin \text{rowspan}\, Dg_{S'}$. Pick $\xi^0 \in U$ such that $Dg_j(\xi^0) \neq 0$; such an ξ^0 must exist because otherwise it will be in rowspan $Dg_{S'}$. By the theorem's assumption, $Dg_S = D\tau Dg_{S'}$. This implies that $(Dg_S)^T$ is in rowspan$(Dg_{S'})^T$ for any ξ. But this is impossible at ξ^0. ∎

A direct corollary of this theorem is that in a single chart scenario, we can use exactly d functionally independent functions in the dictionary to give the explanation.

Corollary 5 *Let $\mathcal{G}, g_S$ defined as before. $\mathcal{M}$ is a smooth manifold with dimension d embedded in $\mathbb{R}^D$. Suppose that $\phi : \mathcal{M} \subset \mathbb{R}^D \to \mathbb{R}^m$ is also an embedding of $\mathcal{M}$ and has a decomposition $\phi(\xi) = h \circ g_S(\xi)$ for each $\xi \in \mathcal{M}$. If there is a C^ℓ diffeomorphism $\varphi : \mathbb{R}^d \to \mathcal{M}$, then there is a subset of functions $g_{S'} \subset g_S$ with $|S'| = d$ such that for some C^ℓ function $\widetilde{h}$ we can find $\phi = \widetilde{h} \circ g_{S'}$ on each $\xi \in \mathcal{M}$*

Proof Since φ is a diffeomorphism, we can write uniquely $\xi = \varphi(\eta)$ for each $\xi \in \mathcal{M}$. And also $\varphi^{-1}(\mathcal{M}) = \mathbb{R}^d$. Then the original decomposition of $\phi(\xi) = h \circ g_S$ on each $\xi \in \mathcal{M}$ is actually

$$\phi \circ \varphi(\eta) = h \circ g_S \circ \varphi(\eta) \tag{25}$$

and $\widetilde{g}_S = g_S \circ \varphi$ is a C^ℓ mapping $\mathbb{R}^d \to \mathbb{R}^s$. Now we choose $S' \subseteq S$ such that $D\widetilde{g}_{S'}$ is full rank and $\operatorname{rank} D\widetilde{g}_S = \operatorname{rank} D\widetilde{g}_{S'}$. From

$$\operatorname{rank}\begin{pmatrix} D\widetilde{g}_S \\ D\widetilde{g}_{S'} \end{pmatrix} = \operatorname{rank} D\widetilde{g}_{S'} \tag{26}$$

and the previous theorem, we know that there exists a C^ℓ function τ such that $\widetilde{g}_S = \tau \circ \widetilde{g}_{S'}$ on each $\eta \in \mathbb{R}^d$. Finally let $\widetilde{h} = h \circ \tau$ then

$$\phi(\xi) = \phi \circ \varphi \circ \varphi^{-1}(\xi) = h \circ \widetilde{g}_S \circ \varphi^{-1}(\xi) = h \circ \tau \circ \widetilde{g}_{S'} \circ \varphi^{-1}(\xi) = \widetilde{h} \circ g_{S'}(\xi) \tag{27}$$

holds for each $\xi \in \mathcal{M}$. Now we determine the number of functions in S'. On one hand we can select S' so that $D\widetilde{g}_{S'} = Dg_{S'}D\varphi$ is full rank, hence $|S'| = \operatorname{rank} Dg_{S'} = s' \leq d$. On the other hand, since ϕ is an embedding, which means that $\phi(\mathcal{M})$ is also an d−dimension manifold, we could consider another diffeomorphism $\psi : \phi(\mathcal{M}) \to \mathbb{R}^d$ such that

$$\psi \circ \phi \circ \varphi(\eta) = \psi \circ \widetilde{h} \circ \widetilde{g}_{S'}(\eta) \tag{28}$$

is $\mathbb{R}^d \to \mathbb{R}^d$ and one-to-one.Therefore the Jacobian of l.h.s. is of rank d, which should be less than the rank of each of the r.h.s. differentials. Therefore $s' \geq d$. To combine these results we have $s' = d$.
∎

We say that S itself is functionally independent when that every $g \in \mathcal{G}_S$ is functionally independent of $\mathcal{G}_S \setminus \{g\}$. Corollary 5 states that S can be uniquely recovered iff S is a functionally independent subset of $\mathcal{G}$.

In a finite sample setting, Theorem 4 states that S and S' are equivalent explanations for f whenever (20) holds on open sets around the sample points. For example, Theorem 4 does not have to hold globally.

For low dimensional settings, i.e. when $s \geq d$, the Theorem implies that in general there will be many equivalent explanations of f in $\mathcal{G}$. Assuming no noise, the solution to (13) will be the subset S that minimizes the penalty term $\frac{\lambda}{\sqrt{mn}} \sum_{j=1}^{p} \|\beta_j\|$ on the data set.

7.2 Recovery guarantees for FLASSO

We now give recovery guarantees for the (FLASSO) problem. The guarantees are deteriministic, but they depend on the noise sample covariance, hence they can lead to statistical guarantees holding w.h.p. in the usual way. The first theorem deals with support recovery, proving that all coefficients outside the support are zeroed out, under conditions that depend only on the dictionary, the true support S and the noise. The second result completes the previous with error bounds on the estimates $\hat{\beta}_j$, assuming an additional condition on the magnitude of the true β_j coefficients. Since these coefficients are partial derivatives w.r.t. the dictionary functions, the condition implies that the the dependence on each function must be strong enough to allow the accurate estimation of the partial derivatives.

Introduce the following quantitites

$$S\text{-incoherence in } \mathcal{G} \quad \mu = \max_{i=1:n, j\in S, j'\notin S} |x_{ij}^T x_{ij'}| \tag{29}$$

$$\text{noise level } \sigma \text{ defined by } \quad \sum_{i=1}^{n} ||\epsilon_i||^2 = nd\sigma^2, \tag{30}$$

and *internal colinearity* ν, defined as follows. Let

$$\Sigma_i \ = \ \left[x_{ij}^T x_{ij'} \right]_{j,j' \in S} = \ \mathbf{X}_{iS}^T \mathbf{X}_{iS}, \quad \text{and} \quad \Sigma \ = \ \text{diag}\{\Sigma_{1:n}\} \tag{31}$$

and denote by $\nu \leq 1$ the maximum eigenvalue of Σ^{-1}; a smaller ν means that the x_{ij} gradients are closer to being orthogonal at each datapoint i.

Theorem 6 (Support recovery) *Assume that equation (4) holds, and that $||x_{ij}|| = 1$ for all $i = 1 : n, j = 1 : p$. Denote by $\bar{\beta}$ the solution of (5) for some $\lambda > 0$. If $\mu\nu\sqrt{s} + \frac{\sigma\sqrt{nd}}{\lambda} < 1$, then $\bar{\beta}_{ij} = 0$ for $j \notin S$ and all $i = 1, \dots n$.*

Proof We structure equation (4) in the form

$$y \ = \ \bar{\bar{X}}\bar{\beta}^* + \bar{\epsilon} \quad \text{with } y = [y_i]_{i=1:n} \in \mathbb{R}^{nd}, \ \bar{\beta} = [\beta_i]_{i=1:n} \in \mathbb{R}^{np}, \tag{32}$$

$\tilde{X}_{ij} \in \mathbb{R}^{nd}$ is obtained from x_{ij} by padding with zeros for the entries not in the i-th segment, $\bar{\bar{X}}_j = [[\tilde{X}_{tj}]_{j=1:p}]_{i=1:n} \in \mathbb{R}^{nd \times np}$, and $\bar{\bar{X}}_j = [\tilde{X}_{ij}]_{i=1:n} \in \mathbb{R}^{nd \times n}$ collects the colums of $\bar{\bar{X}}$ that correspond to the j-th dictionary entry. Note that

$$\tilde{X}_{ij}^T \tilde{X}_{ij'} \ = \ x_{ij}^T x_{ij'} \quad \text{and} \quad \tilde{X}_{ij}^T \tilde{X}_{i'j'} \ = \ 0 \text{ whenever } i \neq i'. \tag{33}$$

The proof is by the *Primal Dual Witness* method, following [ESJ$^+$17, OWJ11]. It can be shown [ESJ$^+$17, Wai09] that $\bar{\beta}$ is a solution to (FLASSO) iff, for all $j = 1 : p$,

$$\bar{\bar{X}}_j^T \bar{\bar{X}}(\bar{\beta} - \bar{\beta}^*) - \bar{\bar{X}}_j^T \bar{\epsilon} + \lambda z_j \ = \ 0 \ \in \mathbb{R}^n \text{ with } z_j = \frac{\beta_j}{||\beta_j||} \text{ if } \beta_j \neq 0 \text{ and } ||z_j|| < 1 \text{ otherwise.} \tag{34}$$

The matrix $\bar{\bar{X}}_j^T \bar{\bar{X}}$ is a diagonal matrix with n blocks of size $1 \times p$, hence the first term in (34) becomes

$$[x_{ij}^T \mathbf{X}_i(\bar{\beta}_{i:} - \bar{\beta}_{i:}^*)]_{i=1:n} \in \mathbb{R}^n. \tag{35}$$

Similarly $\bar{\bar{X}}_j^T \bar{\epsilon} = [x_{ij}^T w_i]_{i=1:n} \in \mathbb{R}^n$.

We now consider the solution $\hat{\beta}$ to problem (5) (FLASSO) under the additional constraint that $\beta_{tj'} = 0$ for $j' \notin S$. In other words, $\hat{\beta}$ is the solution we would obtain if S was known. Let $\hat{z}$ be the optimal dual variable for this problem, and let $\hat{z}_S = [\hat{z}_j]_{j \in S}$.

We will now complete $\hat{z}_S$ to a $z \in \mathbb{R}^{np}$ so that the pair $(\hat{\beta}, z)$ satisfies (34). If we succeed, then we will have proved that $\hat{\beta}$ is the solution to the original FLASSO problem, and in particular that the support of $\hat{\beta}$ is included in S.

From (34) we obtain values for z_j when $j \notin S$.

$$z_j \ = \ \frac{-1}{\lambda} \bar{\bar{X}}_j^T \left[\bar{\bar{X}}^T(\hat{\beta} - \bar{\beta}^*) - \bar{\epsilon} \right]. \tag{36}$$

In the same time, if we consider all $j \in S$, we obtain from (34) that $\bar{\bar{X}}_S = [\bar{\bar{X}}_j]_{j \in S}$ (here the vectors β_S, β_{S*} and all other vectors are size ns, with entries sorted by j, then by i).

$$\bar{\bar{X}}_S^T \bar{\bar{X}}_S(\hat{\beta}_S - \beta_S^*) - \bar{\bar{X}}_S^T \bar{\epsilon} + \lambda \hat{z}_S \ = \ 0. \tag{37}$$

15

Solving for $\hat\beta_S - \beta_S^*$ in (37), we obtain

$$\hat\beta_S - \beta_S^* \;=\; (\bar{\bar X}_S^T \bar{\bar X}_S)^{-1}\left(\bar{\bar X}_S^T \bar\epsilon - \lambda \hat z_S\right) \;=\; \Sigma^{-1}\left(\bar{\bar X}_S^T \bar\epsilon - \lambda \hat z_S\right). \tag{38}$$

After replacing the above in (36) we have

$$z_j \;=\; \frac{-1}{\lambda}\bar{\bar X}_j^T\left[\bar{\bar X}_S \Sigma^{-1}\bar{\bar X}_S^T w - \bar{\bar X}_S \Sigma^{-1}\lambda \hat z_S - \bar\epsilon\right] \;=\; \bar{\bar X}_j^T \bar{\bar X}_S \Sigma^{-1}\hat z_S + \frac{1}{\lambda}\bar{\bar X}_j^T (I - \bar{\bar X}_S \Sigma^{-1}\bar{\bar X}_S^T)\bar\epsilon. \tag{39}$$

Finally, by noting that $\Pi = I - \bar{\bar X}_S \Sigma^{-1}\bar{\bar X}_S^T$ is the projection operator on the subspace $\mathrm{span}(\bar{\bar X}_S)^\perp$, we obtain that

$$z_j \;=\; (\bar{\bar X}_j^T \bar{\bar X}_S)\Sigma^{-1}\hat z_S + \frac{1}{\lambda}\bar{\bar X}_j^T \Pi\bar\epsilon, \quad \text{for } j \notin S. \tag{40}$$

We must show that $||z_j|| < 1$ for $j \notin S$. To bound the first term, we note that $\bar{\bar X}_j^T \bar{\bar X}_S$ is $n \times ns$, block diagonal, with blocks of size $1 \times s$, and with all non-zero entries bounded in absolute value by μ. Hence, for any vector $v = [v_i]_{i=1:n} \in \mathbb{R}^{ns}$, $||\bar{\bar X}_j^T \bar{\bar X}_S v||^2 = ||[(x_{ij}^T x_{iS})v_i]_{i=1:n}||^2 = \sum_{i=1}^n ||(x_{ij}^T x_{iS})v_i||^2 \le \mu^2 \sum_{i=1}^n ||v_i||^2 = \mu^2 ||v||^2$. But in our case $v = \Sigma^{-1}\hat z_S$, hence $||v|| \le ||\Sigma^{-1}||\,||\hat z_S|| = \nu\sqrt{s}$.

To bound the second term, we note that $||\bar{\bar X}_j|| = ||x_{ij}|| = 1$, and that $||\Pi\bar\epsilon|| \le ||\bar\epsilon||$ because Π is a projection. Hence, the norm squared of this term is bounded above $\sum_{i=1}^n ||\epsilon_i||^2 ||x_{ij}||^2/\lambda^2 = nd\sigma^2/\lambda^2$.

Replacing these bounds in (40) we obtain that

$$||z_j|| \le ||\bar{\bar X}_j^T \bar{\bar X}_S \Sigma^{-1}\hat z_S|| + ||\frac{1}{\lambda}\bar{\bar X}_j^T \Pi\bar\epsilon|| \;\le\; \mu\nu\sqrt{s} + \frac{\sigma\sqrt{dn}}{\lambda} \text{ for any } j \notin S. \tag{41}$$

$\square$

Theorem 7 *Assume that equation (5) holds, and that $||x_{ij}|| = 1$ for all $i = 1:n, j = 1:p$. Denote by $\hat\beta$ the solution to problem (FLASSO) for some $\lambda > 0$. If (1) $\mu\nu\sqrt{s} < 1$, (2) $\lambda = (c-1)\sigma\sqrt{dn}$ with $c > 1 + \frac{1}{1-\mu\nu\sqrt{s}}$, (3) $||\beta_j^*|| > c\sigma\sqrt{dn}(1 + \sqrt{s})$ for all $j \in S$, then the support S is recovered exactly and $||\hat\beta_j - \beta_j^*|| < c\sigma\sqrt{dn}(1 + \sqrt{s})$ for all $j \in S$.*

Proof According to Theorem 6, $\hat\beta_j = 0$ for $j \notin S$. It remains to prove the error bound for $j \in S$. According to Lemma V.2 of [ESJ$^+$17], for any $j \in S$,

$$||\hat\beta_j - \beta_j^*|| \;\le\; ||\bar{\bar X}_j^T \bar\epsilon|| + ||\bar{\bar X}_S^T \bar\epsilon|| + \lambda(1 + \sqrt{s}) \tag{42}$$

$$\le\; (||\bar{\bar X}_j|| + ||\bar{\bar X}_S||)||\bar\epsilon|| + \sigma(c-1)\sqrt{dn}(1 + \sqrt{s}) \tag{43}$$

$$\le\; (1 + \sqrt{s})\sigma\sqrt{dn} + \sigma(c-1)\sqrt{dn}(1 + \sqrt{s}) \;=\; c\sigma\sqrt{dn}(1 + \sqrt{s}) \tag{44}$$

Hence, if $||\beta_j^*|| > c\sigma\sqrt{dn}(1 + \sqrt{s})$, $\hat\beta_j \ne 0$ and the support is recovered. $\square$

Note that for this value of c, $\frac{\sigma\sqrt{dn}}{\lambda} + \mu\nu\sqrt{s} = \frac{1}{c-1} + \mu\nu\sqrt{s} < 1$.

The assumptions we make are not probabilistic, while those in [ESJ$^+$17] are. In spite of this difference, the assumptions in our work bear comparison with [ESJ$^+$17]. Specifically, our conditions only concern the internal collinearity, for functions $j \in S$, while [ESJ$^+$17] requires bounds on the *intra-block coherence* (the equivalent condition) for the whole dictionary. Second, we only require incoherence between g_S and the functions not in S; this property is most similar to the *inter-block coherence* bound from [ESJ$^+$17]. In other words, in a deterministic framework incoherence between functions outside the support is not necessary. This conclusion can be extended in a straightforward manner to other group Lasso recovery proofs.

8 Experimental results

8.1 FLasso on synthetic data

We illustrate the effectiveness of FLasso for sparse functional support recovery where the output function is composed of products of dictionary functions. In such situations, methods which only consider linear combinations of dictionary functions will not be able to recover the support. The ability to distinguish such multiplied signals is therefore a key advantage of our approach. These examples have analytically known true solutions in the absence of noise, and also show the robustness of FLasso to noise.

8.1.1 Example 1

Consider the following case where $D = 4$ and $m = 1$ and ξ is drawn from a standard normal distribution with covariance .2.

$$f(\xi) = \xi_1 \xi_2$$
$$\nabla f = [\xi_2, \xi_1, 0, 0]$$

We consider the algorithm output on the following two dictionaries.

$$\mathcal{G}_1 : \xi \mapsto \{\xi_1, \xi_2, \xi_3, \xi_4\}$$

$$\nabla \mathcal{G}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathcal{G}_2 : \xi \mapsto \{\xi_1, \xi_2, \xi_1\xi_2, 0\}$$

$$\nabla \mathcal{G}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \xi_2 & \xi_1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

For the first dictionary, the only loss zero partials are $\frac{\partial h}{\partial g} = [\xi_2, \xi_1, 0, 0]$, while for the second dictionary, $\frac{\partial h_1}{\partial g} = [x_2, x_1, 0, 0]$ and $\frac{\partial h_2}{\partial g} = [0, 0, \gamma_3, 0]$ both result in loss-zero solutions. However, $\|\frac{\partial h_2}{\partial g}\|_{1,2} < \|\frac{\partial h_1}{\partial g}\|_{1,2}$ for all distributions of x, since $\int \|\frac{\partial h_2}{\partial g}\|_{1,2}\mu(dx) = \gamma_3 = \int (x_2^2 + x_1^2)^{1/2}\mu(dx) = \int \|\frac{\partial h_1}{\partial g}\|_{1,2}\mu(dx) = \int (x_2 + x_1)\mu(dx)$ and $1-$norm is bounded by $2-$norm. . Figure 2 shows that FLasso recovers the correct support in both dictionaries.
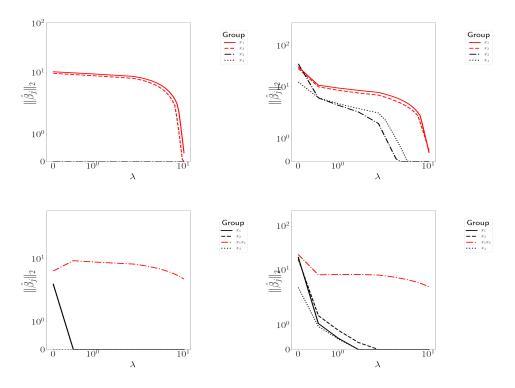
Figure 2: Regularization path of FLasso for Example 1 with 100 points. The first row shows support recovery over a range of λ values for dictionary $\mathcal{G}_1$, and the second for $\mathcal{G}_2$. On the left are results with no noise, and on the right, independent Gaussian noise with $\sigma^2 = 0.01$ was added to the differentials of g and f.

8.1.2 Example 2

This example shows that FLasso is effective at selecting between more complex functional covariates. As in the previous example, the function is a product of two dictionary functions. ξ is drawn from a standard normal distribution with covariance .2 and $d = 8, 20$. We have experimented with other distributions and found that the sampling of ξ has little or no effect, as long as it is not restricted to specific low dimensional sets. Let

$$f = g_1 g_{d+2} = \sin(\xi_1 + \xi_2)(\xi_3^2 + \xi_5^2)$$
$$\nabla f = [\cos(\xi_1 + \xi_2)(\xi_3^2 + \xi_5^2), \cos(\xi_1 + \xi_2)(\xi_3^2 + \xi_5^2), 2\xi_3 \sin(\xi_1 + \xi_2), 0, 2\xi_5 \sin(\xi_1 + \xi_2), 0, \dots]$$
$$\mathcal{G}_1 = \{g_k = \sin(\xi_k + \xi_{k+1}), \ k = 1 : d-1\}$$
$$\mathcal{G}_2 = \{g_{d-1+k} = \xi_k^2 + \xi_{k+2}^2, \ k = 1 : d-2\}$$
$$\mathcal{G} = \mathcal{G}_1 \cup \mathcal{G}_2.$$

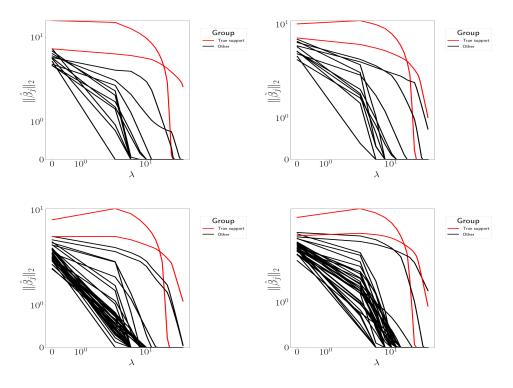Here, we have not included $\nabla\mathcal{G}$ for brevity, but it can be readily derived.

18

Figure 3: Regularization path of FLASSO for Example 2 with 100 points. The first row shows support recovery over a range of λ values for $d = 8$, and the second for $d = 20$. To produce the results on the left, independent Gaussian noise with $\sigma^2 = 0.05$ was added to the differentials of g and f, while on the right $\sigma^2 = 0.1$.

FLASSO coefficients for the true support are highest. However, one of the elements of the true support decays rapidly along the solution path, so our variable selection procedure is not quite robust enough to detect the functional support.

8.2 Manifold FLASSO on synthetic data

8.2.1 Articulated ethanol skeleton

This simulation is on an ethanol skeleton with simplified behavior. This skeleton articulates in a full rotation around the $C - C$ and $C - O$ bonds. Configurations are distributed uniformly around these two angles of rotation. Thus, each configuration is determined by angles of rotation around the $C - C$ and $C - O$ bonds, and the noise (independent Gaussian with $\sigma^2 = .01$) that is added to the position of each atom. The learned manifold is a twisted torus explained by these two angles. Our dictionary consists of four torsions, two which correspond to the articulated angles, and two corresponding to noise. FLASSO is able to detect which of these torsions explain the embedding coordinates with respect to the manifold.
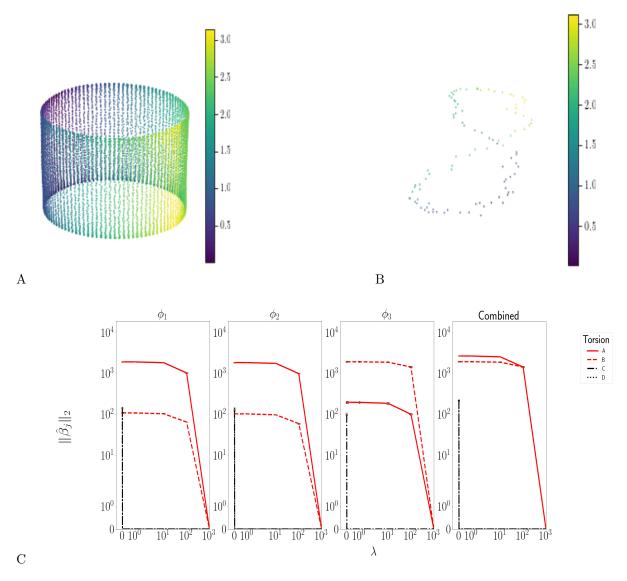
Figure 4: A) The learned manifold on the entire data set colored by the first angle of rotation. B) A radial slice of the learned manifold colored by the other angle of rotation. C) Regularization path of FLASSO for 8 repeats of 200 randomly selected points. The first two embedding coordinates are explained by torsion A (corresponding to A), while the last coordinate is explained by torsion B (corresponding to B). The final panel corresponds to the combined norm used for variable selection.

8.3 Manifold FLASSO on molecular dynamics data

These experiments are based off of molecular dynamics simulations of two small molecules. In molecular dynamics simulations, individual atoms are moved according to the interatomic force fields acting upon them. Each data point corresponds to a position along such a trajectory. We

20

apply a manifold learning algorithm to identify the low-dimensional manifold these trajectories define, seek an explanation of this manifold in terms of bond torsions. A major complicating factor in this is that the true configuration space is $\mathbb{R}^3 \backslash SE(3)$ (the $3 - D$ special Euclidean group), since configurations which differ by rotations and translations (but not mirror images) are equivalent. To respect this symmetry, we parametrize each configuration by the angles of the triangles formed by each combination of three atoms. Torsions are computed with respect to the angular representation.

8.3.1 Toluene

The toluene molecule has one rotational degree of freedom. The manifold learning algorithm therefore discovers that the data manifold is a circle. Out of a functional dictionary consisting of the putatively significant dihedral angle and several other dihedral angles, FLASSO then identifies the bond torsion that explains the circle.
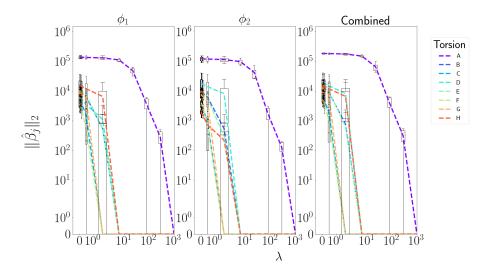

Figure 5: Regularization path of FLASSO for molecular dynamics simulation of toluene with 3 repeats selecting 25 random points out of 50000 used to learn the manifold.

8.3.2 Ethanol

The ethanol molecule has two rotational degree of freedom. The manifold learning algorithm therefore discovers that the data manifold is a torus. Out of a functional dictionary consisting of the two putatively significant dihedral angles and two other dihedral angles, FLASSO then identifies the bond torsions that explain this torus.
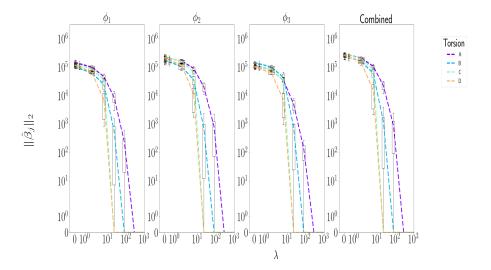
Figure 6: Regularization path of FLASSO for molecular dynamics simulation ethanol with 8 repeats of selecting 200 random points out of 50000 used to learn the manifold.

References

[BN02]   M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

[BPK16]   Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016.

[CDW14]   P. Constantine, E. Dow, and Q. Wang. Active subspace methods in theory and practice: Applications to kriging surfaces. *SIAM Journal on Scientific Computing*, 36(4):A1500–A1524, 2014.

[CL06]   R. R. Coifman and S. Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 30(1):5–30, 2006.

[CLL+05]   R. R. Coifman, S. Lafon, A. Lee, Maggioni, Warner, and Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. In *Proceedings of the National Academy of Sciences*, pages 7426–7431, 2005.

[CNO00]   C. Clementi, H. Nymeyer, and J.N. Onuchic. Topological and energetic factors: what determines the structural details of the transition state ensemble and "en-route" intermediates for protein folding? an investigation for small globular proteins. *Journal of molecular biology*, 2000. says topology (of protein) more important than energy wells.

[dC92]   Manfredo do Carmo. *Riemannian Geometry*. Springer, 1992.

[DTCK18]   Carmeline J Dsilva, Ronen Talmon, Ronald R Coifman, and Ioannis G Kevrekidis. Parsimonious representation of nonlinear dynamical systems through manifold learning: A chemotaxis case study. *Appl. Comput. Harmon. Anal.*, 44(3):759–773, May 2018.

[ESJ⁺17] M.K. Elyaderani, S.Jain, J.M.Druce, S.Gonella, and J.D.Haupt. Improved support recovery guarantees for the group lasso with applications to structural health monitoring. *CoRR*, abs/1708.08826, 2017.

[Gea12] C W Gear. Parameterization of non-linear manifolds. August 2012.

[HAvL05] Matthias Hein, Jean-Yves Audibert, and Ulrike von Luxburg. From graphs to manifolds - weak and strong pointwise consistency of graph laplacians. In *Learning Theory, 18th Annual Conference on Learning Theory, COLT 2005, Bertinoro, Italy, June 27-30, 2005, Proceedings*, pages 470–485, 2005.

[HAvL07] Matthias Hein, Jean-Yves Audibert, and Ulrike von Luxburg. Graph laplacians and their convergence on random neighborhood graphs. *Journal of Machine Learning Research*, 8:1325–1368, 2007.

[KvL15] Matthäus Kleindessner and Ulrike von Luxburg. Dimensionality estimation without distances. In *AISTATS*, 2015.

[Lee03] John M. Lee. *Introduction to Smooth Manifolds.* Springer-Verlag New York, 2003.

[LSW09] Chuanjiang Luo, Issam Safa, and Yusu Wang. Approximating gradients for meshes and point clouds via diffusion metric. *Comput. Graph. Forum*, 28(5):1497–1508, July 2009.

[MN17] Kitty Mohammed and Hariharan Narayanan. Manifold learning using kernel density estimation and local principal components analysis. *arxiv*, 1709.03615, 2017.

[NC17] Frank Noé and Cecilia Clementi. Collective variables for the study of long-time kinetics from molecular trajectories: theory and methods. *Curr. Opin. Struct. Biol.*, 43:141–147, April 2017.

[OPV⁺14] Junier Oliva, Barnabas Poczos, Timothy Verstynen, Aarti Singh, Jeff Schneider, Fang-Cheng Yeh, and Wen-Yih Tseng. FuSSO: Functional Shrinkage and Selection Operator. In *International Conference on AI and Statistics(AISTATS)*, 2014.

[OWJ11] Guillaume Obozinski, Martin J. Wainwright, and Michael I. Jordan. Support union recovery in high-dimensional multivariate regression. *The Annals of Statistics*, 39(1):1–47, 2011.

[PM13] D. Perraul-Joncas and M. Meila. Non-linear dimensionality reduction: Riemannian metric estimation and the problem of geometric discovery. *ArXiv e-prints*, May 2013.

[RS00] Sam Roweis and Lawrence Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, December 2000.

[SR03] Lawrence K. Saul and Sam T. Roweis. Think globally, fit locally: Unsupervised learning of low dimensional manifolds. *J. Mach. Learn. Res.*, 4:119–155, December 2003.

[THJ10] Daniel Ting, Ling Huang, and Michael I. Jordan. An analysis of the convergence of graph laplacians. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 1079–1086, 2010.

[TR02] Yee Whye Teh and Sam T. Roweis. Automatic alignment of local representations. In *NIPS*, 2002.

[Wai09] Martin J. Wainwright. Sharp thresholds for high-dimensional and noisy sparsity recovery using ℓ_1 -constrained quadratic programming (lasso). *IEEE Transactions on Information Theory*, 55:2183–2202, 2009.

[YL06] M Yuan and Y Lin. Model selection and estimation in regression with grouped variables. *J. R. Stat. Soc. Series B Stat. Methodol.*, 2006.

[YX12] Gui-Bo Ye and Xiaohui Xie. Learning sparse gradients for variable selection and dimension reduction. *Machine Learning*, 87(3):303–355, Jun 2012.

[Zor04] Vladimir A. Zorich. *Mathematical Analysis I.* Springer-Verlag Berlin Heidelberg, 2004.

[ZZ04] Zhenyue Zhang and Hongyuan Zha. Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. *SIAM J. Scientific Computing*, 26(1):313–338, 2004.