Discriminative Interpolation for Classification of Functional Data

Rana Haber[†], Anand Rangarajan[‡], and Adrian M. Peter[*]

[†]Mathematical Sciences Department, Florida Institute of Technology,
[‡]Dept. of Computer and Information Science and Engineering, University of Florida,
[*]Systems Engineering Department, Florida Institute of Technology[⋆]
rhaber2010@my.fit.edu,anand@cise.ufl.edu,apeter@fit.edu

Abstract. The *modus operandi* for machine learning is to represent data as feature vectors and then proceed with training algorithms that seek to optimally partition the feature space $S \subset \mathbb{R}^n$ into labeled regions. This holds true even when the original data are functional in nature, i.e. curves or surfaces that are inherently varying over a continuum such as time or space. Functional data are often reduced to summary statistics, locally-sensitive characteristics, and global signatures with the objective of building comprehensive feature vectors that uniquely characterize each function. The present work directly addresses representational issues of functional data for supervised learning. We propose a novel *classification by discriminative interpolation (CDI)* framework wherein functional data in the same class are adaptively reconstructed to be more similar to each other, while simultaneously repelling nearest neighbor functional data in other classes. Akin to other recent nearest-neighbor metric learning paradigms like stochastic k-neighborhood selection and large margin nearest neighbors, our technique uses class-specific representations which gerrymander similar functional data in an appropriate parameter space. Experimental validation on several time series datasets establish the proposed discriminative interpolation framework as competitive or better in comparison to recent state-of-the-art techniques which continue to rely on the standard feature vector representation.

1 Introduction

The choice of data representation is foundational to all supervised and unsupervised analysis. The *de facto* standard in machine learning is to use feature representations that treat data as n-dimensional vectors in Euclidean space and then proceed with multivariate analysis in $\mathbb{R}^n$. This approach is uniformly adopted even for sensor-acquired data which naturally come in functional form—most sensor measurements are identifiable with real-valued functions collected over a discretized continuum like time or space. Familiar examples include time series data, digital images, video, LiDAR, weather, and multi-/hyperspectral volumes. The present work proposes a framework where we directly leverage functional

[⋆] This work is partially supported by NSF IIS 1065081 and 1263011.

representations to develop a k-nearest neighbor (kNN) supervised learner capable of discriminatively interpolating functions in the same class to appear more similar to each other than functions from other classes—we refer to this throughout as *Classification by Discriminative Interpolation* (CDI).

Over the last 25 years, the attention to statistical techniques with functional representations has resulted in the subfield commonly referred to as Functional Data Analysis (FDA) [1]. In FDA, the above mentioned data modalities are represented by their functional form: $f : \mathbb{R}^p \rightarrow \mathbb{R}^q$. To perform data analysis, one relies on the machinery of a Hilbert space structure endowed with a collection of functions. On a Hilbert space $\mathcal{H}$, many useful tools like bases, inner products, addition, and scalar multiplication allow us to mimic multivariate analysis on $\mathbb{R}^n$. However, since function spaces are in general infinite dimensional, special care must be taken to ensure theoretical concerns like convergence and set measures are defined properly. To bypass these additional concerns, some have resorted to treating real-valued functions of a real variable $f : \mathbb{R} \rightarrow \mathbb{R}$, sampled at m discrete points, $\mathbf{f} = \{f(t_i)\}_{i=1}^m$, as a vector $\mathbf{f} \in \mathbb{R}^m$ and subsequently apply standard multivariate techniques. Clearly, this is not a principled approach and unnecessarily strips the function properties from the data. The predominant method, when working with functional data, is to move to a feature representation where filters, summary statistics, and signatures are all derived from the input functions and then analysis proceeds per the norm. To avoid this route, several existing FDA works have revisited popular machine learning algorithms and reformulated them rigorously to handle functional data [2,3,4,5]—showing both theoretical and practical value in retaining the function properties. Here we also demonstrate how utilizing the well-known and simple interpolation property of functions can lead to a novel classification framework. Intuitively, given A classes and labeled exemplar functions $\left\{\mathbf{f}^j, y^j\right\}$, where labels $y^j = \{1, \ldots, A\}$ are available during training, we expand each $\mathbf{f}^j$ in an appropriate basis representation to yield a faithful reconstruction of the original curve. (Note: we use *function* and *curve* synonymously throughout the exposition.) In a supervised framework, we then allow the interpolants to morph the functions such that they resemble others from the same class, and incorporate a counter-force to repel similarities to other classes. The use of basis interpolants critically depends on the functional nature of the data and cannot be replicated efficiently in a feature vector representation where no underlying continuum is present. Though admittedly, this would become more cumbersome if for a function with domain dimension p and co-domain dimension q both were high dimensional, but for most practical datasets we have $p \leq 4$. In the current work, we detail the theory and algorithms for real-valued functions over the real line expanded in wavelet bases with extensions to higher dimensions following in a straightforward manner.

The motivation for a competing push-pull framework is built on several recent successful efforts in metric learning [6,7,8,9] where Mahalanobis-style metrics are learned by incorporating optimization terms that promote purity of local neighborhoods. The metric is learned using a kNN approach that is locally sen-

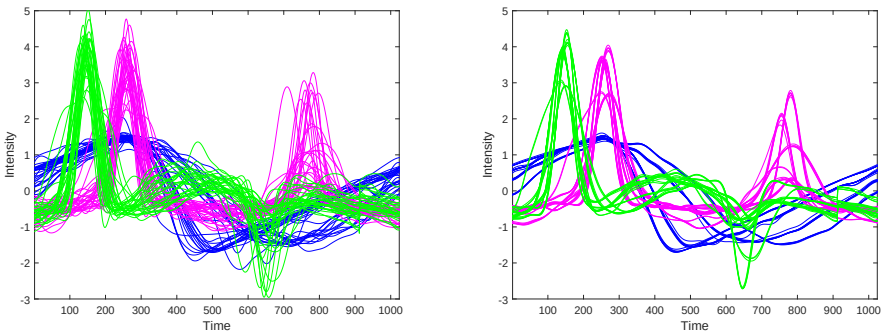


Fig. 1. Three-class *classification by discriminative interpolation* (CDI). (a) Original training functions from three different classes, showing 30 curves each. (b) Post training via the proposed CDI method, functions in each class morphed to resemble k-nearest neighbors [same 30 curves per class as in (a)].

sitive to the the k neighbors around each exemplar and optimized such that the metric moves data with the same class labels closer in proximity (pulling in good neighbors) and neighbors with differing labels from the exemplar are moved out of the neighborhood (pushing away bad neighbors). As these are kNN approaches, they can inherently handle nonlinear classification tasks and have been proven to work well in many situations [10] due to their ability to contextualize learning through the use of neighborhoods. In these previous efforts, the metric learning framework is well-suited for feature vectors in $\mathbb{R}^n$. In our current situation of working with functional data, we propose an analogue that allows the warping of the function data to visually resemble others in the same class (within a k-neighborhood) and penalize similarity to bad neighbors from other classes. We call this gerrymandered morphing of functions based on local neighborhood label characteristics *discriminative interpolation*. Figure 1 illustrates this neighborhood-based, supervised deforming on a three-class problem. In Fig. 1(a), we see the original training curves from three classes, colored magenta, green, and blue; each class has been sub-sampled to 30 curves for display purposes. Notice the high variability among the classes which can lead to misclassifications. Fig. 1(b) shows the effects of CDI post training. Now the curves in each class more closely resemble each other, and ultimately, this leads to better classification of test curves.

Learning and generalization properties have yet to be worked out for the CDI framework. Here, we make a few qualitative comments to aid better understanding of the formulation. Clearly, we can overfit during the training stage by forcing the basis representation of all functions belonging to the same class to be identical. During the testing stage, in this overfitting regime, training is irrelevant and the method devolves into a nearest neighbor strategy (using function distances). Likewise, we can underfit during the training stage by forcing the basis representation of each function to be without error (or residual). During testing,

in this underfitting regime, the basis representation coefficients are likely to be far (in terms of a suitable distance measure on the coefficients) from members in each class since no effort was made during training to conform to any class. We think a happy medium exists where the basis coefficients for each training stage function strike a reasonable compromise between overfitting and underfitting—or in other words, try to reconstruct the original function to some extent while simultaneously attempting to draw closer to nearest neighbors in each class. Similarly, during testing, the classifier fits testing stage function coefficients while attempting to place the function pattern close to nearest neighbors in each class with the eventual class label assigned to that class with the smallest compromise value. From an overall perspective, CDI marries function reconstruction with neighborhood gerrymandering (with the latter concept explained above).

To the best of our knowledge, this is the first effort to develop a FDA approach that leverages function properties to achieve kNN-margin-based learning in a *fully multi-class framework*. Below, we begin by briefly covering the requisite background on function spaces and wavelets (Section 2). Related works are detailed in Section 3. This is followed by the derivation of the proposed CDI framework in Section 4. Section 5 demonstrates extensive experimental validations on several functional datasets and shows our method to be competitive with other functional and feature-vector based algorithms—in many cases, demonstrating the highest performance measures to date. The article concludes with Section 6 where recommendations and future extensions are discussed.

2 Function Representations and Wavelets

Most FDA techniques are developed under the assumption that the given set of labeled functional data can be suitably approximated by and represented in an infinite dimensional Hilbert space $\mathcal{H}$. Ubiquitous examples of $\mathcal{H}$ include the space of square-integrable functions $\mathcal{L}^2([a,b] \subset \mathbb{R})$ and square-summable series $l^2(\mathbb{Z})$. This premise allows us to transition the analysis from the functions themselves to the coefficients of their basis expansion. Moving to a basis expansion also allows us to seamlessly handle irregularly sampled functions, missing data, and interpolate functions (the most important property for our approach). We now provide a brief exposition of working in the setting of $\mathcal{H}$. The reader is referred to many suitable functional analysis references [11] for further details.

The infinite dimensional representation of $f \in \mathcal{H}$ comes from the fact that there are a countably infinite number of basis vectors required to produce an exact representation of f, i.e.

$$f(t) = \sum_{l=1}^{\infty} \alpha_l \phi_l(t) \tag{1}$$

where ϕ is one of infinitely possible bases for $\mathcal{H}$. Familiar examples of ϕ include the Fourier basis, appropriate polynomials, and in the more contemporary setting—wavelets. In computational applications, we cannot consider infinite expansions and must settle for a projection into a finite d-dimensional subspace $\mathcal{P}$.

Given a discretely sampled function $\mathbf{f} = \{f(t_i)\}_{1 \le i \le m}$, the coefficients for this projection are given by minimizing the quadratic objective function

$$\min_{\{\alpha_l\}} \sum_{i=1}^{m} \left(f(t_i) - \sum_{l=1}^{d} \alpha_l \phi_l(t_i) \right)^2$$

or in matrix form

$$\min_{\alpha} \|\mathbf{f} - \boldsymbol{\phi}\boldsymbol{\alpha}\|^2, \tag{2}$$

where $\boldsymbol{\phi}$ is an $m \times d$ matrix with entries $\boldsymbol{\phi}_{i,l} = \phi_l(t_i)$ and $\boldsymbol{\alpha}$ is an $d \times 1$ column vector of the coefficients. Estimation is computationally efficient with complexity $O(md^2)$ [1]. For an orthonormal basis set, this is readily obtainable via the inner product of the space$\alpha_l = \langle f, \phi_i \rangle$. Once the function is represented in the subspace, we can shift our analysis from working directly with the function to instead working with the coefficients. Relevant results utilized later in our optimization framework include

$$\langle \mathbf{f}^h, \mathbf{f}^j \rangle = \left(\boldsymbol{\alpha}^h\right)^T \boldsymbol{\Phi} \boldsymbol{\alpha}^j, \tag{3}$$

where the $d \times d$ matrix $\boldsymbol{\Phi}$ is defined by $\boldsymbol{\Phi}_{r,s} = \langle \phi_r, \phi_s \rangle$ (not dependent on $\mathbf{f}^h$ and $\mathbf{f}^j$), and

$$\|\mathbf{f}^h - \mathbf{f}^j\|_2^2 = \left(\boldsymbol{\alpha}^h - \boldsymbol{\alpha}^j\right)^T \boldsymbol{\Phi} \left(\boldsymbol{\alpha}^h - \boldsymbol{\alpha}^j\right). \tag{4}$$

For orthonormal basis expansions, such as many useful wavelet families, eq. (4) reduces to $\|\boldsymbol{\alpha}^h - \boldsymbol{\alpha}^j\|_2^2$.

Though many different bases exist for $\mathcal{H} = \mathcal{L}^2([a,b])$, wavelet expansions are now widely accepted as the most flexible in providing compact representations and faithful reconstructions. Functions $f \in \mathcal{L}^2([a,b])$ can be represented as a linear combination of wavelet bases[12]

$$f(t) = \sum_{k} \alpha_{j_0,k} \phi_{j_0,k}(t) + \sum_{j \ge j_0,k}^{\infty} \beta_{j,k} \psi_{j,k}(t) \tag{5}$$

where $t \in \mathbb{R}$, $\phi(x)$ and $\psi(x)$ are the *scaling* (a.k.a. father) and *wavelet* (a.k.a. mother) basis functions respectively, and $\alpha_{j_0,k}$ and $\beta_{j,k}$ are scaling and wavelet basis function coefficients; the j-index represents the current resolution level and the k-index the integer translation value. (The translation range of k can be computed from the span of the data and basis function support size at the different resolution levels. Also, when there is no need to distinguish between scaling or wavelet coefficients, we simply let $\mathbf{c} = \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{bmatrix}$.) The linear combination in eq. (5) is known as a multiresolution expansion. The key idea behind multiresolution theory is the existence of a sequence of nested subspaces V_j $j \in \mathbb{Z}$ such that

$$\cdots V_{-2} \subset V_{-1} \subset V_0 \subset V_1 \subset V_2 \cdots \tag{6}$$

and which satisfy the properties $\bigcap V_j = \{0\}$ and $\overline{\bigcup V_j} = \mathcal{L}^2$ (completeness). The resolution increases as $j \to \infty$ and decreases as $j \to -\infty$ (some references show

this order reversed due to the fact they invert the scale [12]). At any particular level $j+1$, we have the following relationship

$$V_j \bigoplus W_j = V_{j+1} \tag{7}$$

where W_j is a space orthogonal to V_j, i.e. $V_j \bigcap W_j = \{0\}$. The father wavelet $\phi(x)$ and its integer translations form a basis for V_0. The mother wavelet $\psi(x)$ and its integer translates span W_0. These spaces decompose the function into its smooth and detail parts; this is akin to viewing the function at different scales and at each scale having a low pass and high pass version of the function. The primary usefulness of a full multiresolution expansion given by eq. (5) is the ability to threshold the wavelet coefficients and obtain a sparse representation of the signal. If sparse representations are not required, functions can be expanded using strictly scaling functions ϕ. Our CDI technique adopts this approach and selects an appropriate resolution level j_0 based on empirical cross-validation on the training data. Given a set of functional data, coefficients for each of the functions can be computed using eq. (2). Once the coefficients are estimated, as previously discussed, all subsequent analysis can be transitioned to working directly with the coefficients. There are also a number wavelet families from which we can select ϕ. With our desire to work with orthonormal bases, we limit ourselves to the compactly supported families of Daubechies, Symlets, and Coiflets [12].

3 Related Work

Several authors have previously realized the importance of taking the functional aspect of the data into account. These include representing the data in a different basis such as B-Splines [13], Fourier [14] and wavelets [15,16] or utilizing the continuous aspects [17] and differentiability of the functional data [18]. Abraham *et al.* [13] fit the data using B-Splines and then use the K-means algorithm to cluster the data. Biau *et al.* [14] reduce the infinite dimension of the space of functions to the first d dimensional coefficients of a Fourier series expansion of each function and then they apply a k-nearest neighborhood for classification. Antoniadis *et al.* [15] use the wavelet transform to detect clusters in high dimensional functional data. They present two different measures, a similarity and a dissimilarity, that are then used to apply the k-centroid clustering algorithm. The similarity measure is based on the distribution of energy across scales generating a number of features and the dissimilarity measure are based on wavelet-coherence tools. In [16], Berlinet *et al.* expand the observations on a wavelet basis and then apply a classification rule on the non-zero coefficients that is not restricted to the k-nearest neighbors. Alonso *et al.* [17] introduce a classification technique that uses a weighted distance that utilizes the first, second and third derivative of the functional data. López-Pintado and Romo [18] propose two *depth*-based classification techniques that take into account the continuous aspect of the functional data. Though these previous attempts have

demonstrated the utility of basis expansions and other machine learning techniques on functional data, none of them formulate a neighborhood, margin-based learning technique as proposed by our current CDI framework.

In the literature, many attempts have been made to find the *best* neighborhood and/or define a good metric to get better classification results [6,8,9,19,20,21]. Large Margin Nearest Neighbor (LMNN) [6,19] is a locally adaptive metric classification method that uses margin maximization to estimate a local flexible metric. The main intuition of LMNN is to learn a metric such that at least k of its closest neighbors are from the same class. It pre-defines k neighbors and identifies them as *target* neighbors or *impostors*—the same class or different class neighbors respectively. It aims at readjusting the margin around the data such that the *impostors* are outside that margin and the k data points inside the margin are of the same class. Prekopcsák and Lemire [21] classify times series data by learning a Mahalanobis distance by taking the pseudoinverse of the covariance matrix, limiting the Mahalanobis matrix to a diagonal matrix or by applying covariance shrinkage. They claim that these metric learning techniques are comparable or even better than LMNN and Dynamic Time Warping (DTW) [22] when one nearest neighbor is used to classify functional data. We show in the experiments in Section 5 that our CDI method performs better. In [8], Trivedi *et al.* introduce a metric learning algorithm by selecting the neighborhood based on a gerrymandering concept; redefining the margins such that the majority of the nearest neighbors are of the same class. Unlike many other algorithms, in [8], the choice of neighbors is a latent variable which is learned at the same time as it is learning the optimal metric—the metric that gives the best classification accuracy. These neighborhood-based approaches are pioneering approaches that inherently incorporate context (via the neighbor relationships) while remaining competitive with more established techniques like SVMs [23] or deep learning [24]. Building on their successes, we adapt a similar concept of redefining the class margins through pushing away *impostors* and pulling *target* neighbors closer.

4 Classification by Discriminative Interpolation

In this section, we introduce the CDI method for classifying functional data. Before embarking on the detailed development of the formulation, we first provide an overall sketch.

In the training stage, the principal task of discriminative interpolation is to learn a basis representation of all training set functions while simultaneously pulling each function representation toward a set of nearest neighbors in the same class and pushing each function representation away from nearest neighbors in the other classes. This can be abstractly characterized as

$$E_{\mathrm{CDI}}(\mathbf{c}^i) = \sum_i \left[D_{\mathrm{rep}}(\mathbf{f}^i, \boldsymbol{\phi}\mathbf{c}^i) + \lambda \sum_{j \in \mathcal{N}(i)} D_{\mathrm{pull}}(\mathbf{c}^i, \mathbf{c}^j) - \mu \sum_{k \in \mathcal{M}(i)} D_{\mathrm{push}}(\mathbf{c}^i, \mathbf{c}^k) \right] \quad (8)$$

where D_{rep} is the representation error between the actual data (training set function sample) and its basis representation, D_{pull} is the distance between the

coefficients of the basis representation in the same class and D_{push} the distance between coefficients in different classes (with the latter two distances often chosen to be the same). The parameters λ and μ weigh the pull and push terms respectively. $\mathcal{N}(i)$ and $\mathcal{M}(i)$ are the sets of nearest neighbors in the same class and different classes respectively.

Upon completion of training, the functions belonging to each class have been discriminatively interpolated such that they are more similar to their neighbors in the same class. This contextualized representation is reflected by the coefficients of the wavelet basis for each of the curves. We now turn our attention to classifying incoming test curves. Our labeling strategy focuses on selecting the class that is able to best represent the test curve under the pulling influence of its nearest neighbors in the same class. In the testing stage, the principal task of discriminative interpolation is to learn a basis representation for just the test set function while simultaneously pulling the function representation toward a set of nearest neighbors in the chosen class. This procedure is repeated for all classes with class assignment performed by picking that class which has the lowest compromise between the basis representation and pull distances. This can be abstractly characterized as

$$\hat{a} = \arg\min_{a} \min_{\mathbf{c}} D_{\text{rep}}(\mathbf{f}, \boldsymbol{\phi}\mathbf{c}) + \lambda \sum_{k \in \mathcal{K}(a)} D_{\text{pull}}(\mathbf{c}, \mathbf{c}^{k}_{(a)}) \tag{9}$$

where $\mathcal{K}(a)$ is the set of nearest neighbors in class a of the incoming test pattern's coefficient vector $\mathbf{c}$. Note the absence of the push mechanism during testing. Further, note that we have to solve an optimization problem during the testing stage since the basis representation of each function is *a priori* unknown (for both training and testing).

4.1 Training Formulation

Given labeled functional data $\{(\mathbf{f}^i, y^i)\}_{i=1}^N$ where $\mathbf{f}^i \in \mathcal{H}$ and $y^i = \{1, \ldots, A\}$ are the labels, A is the number of classes. We can express $\mathbf{f}^i$ as a series expansion

$$\mathbf{f}^i = \sum_{l=1}^{\infty} c_l^i \phi_l \tag{10}$$

where $\{\phi_d\}_{d=1}^{\infty}$ form a complete, orthonormal system of $\mathcal{H}$. As mentioned in Section 2, we approximate the discretized data $\mathbf{f}^i = \{f^i(t_j)\}_{1 \leq j \leq m}$ in a d-dimensional space. Let $\mathbf{c}^i = [c_1^i, c_2^i, \cdots, c_d^i]^T$ be the $d \times 1$ vector of coefficients associated with the approximation $\hat{\mathbf{f}}^i$ of $\mathbf{f}^i$ and let $\boldsymbol{\phi} = [\phi_1, \phi_2, \cdots, \phi_d]$ be the $m \times d$ matrix of the orthonormal basis. Then the approximation to eq. 10 can be written in matrix form as

$$\hat{\mathbf{f}}^i = \boldsymbol{\phi}\mathbf{c}^i. \tag{11}$$

Getting the best approximation to $\mathbf{f}^i$ requires minimizing eq. 2, but in CDI we want to find a weighted approximation $\hat{\mathbf{f}}^i$ such that the function resembles more

the functions in its class; i.e. we seek to minimize

$$\sum_{j \text{ s.t. } y_i = y_j} M_{ij} \|\hat{\mathbf{f}}^i - \hat{\mathbf{f}}^j\|^2 \tag{12}$$

while reducing the resemblance to functions in other classes; i.e. we seek to maximize

$$\sum_{j \text{ s.t. } y_i \neq y_j} M'_{ij} \|\hat{\mathbf{f}}^i - \hat{\mathbf{f}}^j\|^2 \tag{13}$$

where M_{ij} and M'_{ij} are some weight functions—our implementation of the adapted push-pull concept. Combining the three objective function terms, we attempt to get the best approximation by *pulling* similarly labeled data together while *pushing* different labeled data away. This yields the following objective function and optimization problem:

$$\min_{\hat{\mathbf{f}}, M, M'} E = \min_{\hat{\mathbf{f}}, M, M'} \sum_{i=1}^{N} \|\mathbf{f}^i - \hat{\mathbf{f}}^i\|^2 + \lambda \sum_{i,j \text{ s.t. } y_i = y_j} M_{ij} \|\hat{\mathbf{f}}^i - \hat{\mathbf{f}}^j\|^2 - \mu \sum_{i,j \text{ s.t. } y_i \neq y_j} M'_{ij} \|\hat{\mathbf{f}}^i - \hat{\mathbf{f}}^j\|^2 \tag{14}$$

where $M_{ij} \in (0,1)$ and $\sum_j M_{ij} = 1$ is the nearest neighbor constraint for $y_i = y_j$ where $j \neq i$, similarly, $M'_{ij} \in (0,1)$ and $\sum_j M'_{ij} = 1$ is the nearest neighbor constraint for $y_i \neq y_j$ where $j \neq i$. From Section 2, we showed that given an orthonormal basis, $\|\hat{\mathbf{f}}^i - \hat{\mathbf{f}}^j\|^2 = \|\boldsymbol{c}^i - \boldsymbol{c}^j\|^2$, eq. 14 can be expressed in terms of the coefficients as

$$\min_{\mathbf{c}, M, M'} E = \min_{\mathbf{c}, M, M'} \sum_{i=1}^{N} \|\mathbf{f}^i - \boldsymbol{\phi}\mathbf{c}^i\|^2 + \lambda \sum_{i,j \text{ s.t. } y_i = y_j} M_{ij} \|\mathbf{c}^i - \mathbf{c}^j\|^2 - \mu \sum_{i,j \text{ s.t. } y_i \neq y_j} M'_{ij} \|\mathbf{c}^i - \mathbf{c}^j\|^2. \tag{15}$$

As we saw in Section 3, there are many different ways to set the nearest neighbor. The simplest case is to find only one nearest neighbor from the same class and one from the different classes. Here, we set $M_{ij} \in \{0,1\}$ and likewise for M'_{ij}, which helps us obtain an analytic update for $\mathbf{c}^i$ (while keeping M, M' fixed). An extension to this approach would be to find the k-nearest neighbors allowing $k > 1$. A third alternative would be to have graded membership of neighbors with a free parameter deciding the degree of membership. We adopt this strategy—widely prevalent in the literature [25]—as a softmax winner-take-all. In this approach, the M_{ij} and M'_{ij} are "integrated out" to yield

$$\min_{\mathbf{c}} \sum_{i=1}^{N} \|\mathbf{f}^i - \boldsymbol{\phi}\mathbf{c}^i\|^2 - \frac{\lambda}{\beta} \sum_i \log \sum_{j \text{ s.t. } y_i = y_j} e^{-\beta \|\mathbf{c}^i - \mathbf{c}^j\|^2} + \frac{\mu}{\beta} \sum_i \log \sum_{r \text{ s.t. } y_i \neq y_r} e^{-\beta \left\|\mathbf{c}^i - \mathbf{c}^r\right\|^2} \tag{16}$$

where β is a free parameter deciding the degree of membership. This will allow curves to have a weighted vote in the CDI of $\mathbf{f}^i$ (for example).

Algorithm 1 Functional Classification by Discriminative Interpolation (CDI)

Training

Input: $\mathbf{f}_{\text{train}} \in \mathbb{R}^{N\times m}$, $y_{\text{train}} \in \mathbb{R}^N$, λ (CV*), μ (CV), k, k_p, η (stepsize)
Output: $\mathbf{c}^i$(optimal interpolation)
1. **For** $i \leftarrow 1$ to N
2. **Repeat**
3. Find $\mathcal{N}(i)$ and $\mathcal{M}(i)$ using kNN
4. Compute $M_{ij}\ \forall\, i,j$ pairs (eq. 18)
5. Compute $\nabla E^i_{\text{interp}} = \frac{\partial E}{\partial \mathbf{c}^i}$ (eq. 17)
6. $\mathbf{c}^i \leftarrow \mathbf{c}^i - \eta \nabla E^i_{\text{interp}}$
8. **Until** convergence
7. **End For**
*Obtained via cross validation (CV).

Testing

Input: $\mathbf{f}_{\text{test}} \in \mathbb{R}^{L\times m}$, $\mathbf{c}^i$(from training), λ (CV), μ (CV), k, η' (stepsize)
Output: $\hat{y}$ (labels for all testing data)
1. **For** $l \leftarrow 1$ to L
2. **For** $a \leftarrow 1$ to A
3. **Repeat**
4. Find $\mathcal{N}(a)$ for $\tilde{\mathbf{c}}^l$using kNN
5. Compute $M^a_i\ \forall\, i$ neighborhood (22)
6. Compute $\nabla E^l_a = \frac{\partial E^a}{\partial \tilde{\mathbf{c}}^l}$ (eq. 19)
7. $\tilde{\mathbf{c}}^l \leftarrow \tilde{\mathbf{c}}^l - \eta' \nabla E^l_a$
8. **Until** convergence
9. compute E^l_a (eq. 21)
10. **End For**
11. $\tilde{y}^l \leftarrow \{a | \min E^l_a\ \forall a\}$
12. **End For**

An update equation for the objective in eq. 16 can be found by taking the gradient with respect to $\mathbf{c}^i$, which yields

$$\frac{\partial E}{\partial \mathbf{c}^i} = -2\left(\boldsymbol{\phi}^T\mathbf{f}^i - \boldsymbol{\phi}^T\boldsymbol{\phi}\mathbf{c}^i - \lambda\sum_{j\,\text{s.t.}\,y_i=y_j} M_{ij}(\mathbf{c}^i-\mathbf{c}^j) + \lambda\sum_{k\,\text{s.t.}\,y_k=y_i} M_{ki}(\mathbf{c}^k-\mathbf{c}^i)\cdots \right.$$
$$\left. +\mu\sum_{r\,\text{s.t.}\,y_i\neq y_r} M_{ir}(\mathbf{c}^i-\mathbf{c}^r) - \mu\sum_{s\,\text{s.t.}\,y_s\neq y_i} M_{si}(\mathbf{c}^s-\mathbf{c}^i)\right) \quad (17)$$

where

$$M_{ij} = \frac{\exp\left(-\beta\|\mathbf{c}^i-\mathbf{c}^j\|^2\right)}{\sum\limits_{t,\,\text{s.t.}\,y_i=y_t}\exp\left(-\beta\|\mathbf{c}^i-\mathbf{c}^t\|^2\right)}\,. \quad (18)$$

The computational complexity of this approach can be prohibitive. Since the gradient w.r.t. the coefficients involves a graded membership to *all* the datasets, the worst case complexity is $O(Nd)$. In order to reduce this complexity, we use an approximation to eq. 17 via an expectation-maximization (EM) style heuristic. We first find the nearest neighbor sets, $\mathcal{N}(i)$ (same class) and $\mathcal{M}(i)$ (different classes), compute the softmax in eq. 18 using these nearest neighbor sets and then use gradient descent to find the coefficients. Details of the procedure are found in Algorithm 1. Future work will focus on developing efficient *and* convergent optimization strategies that leverage these nearest neighbor sets.

4.2 Testing Formulation

We have estimated a set of coefficients which in turn give us the best approximation to the training curves in a discriminative setting. We now turn to the testing stage. In contrast to the feature vector-based classifiers, this stage is not straightforward. When a test function appears, we don't know its wavelet coefficients. In order to determine the best set of coefficients for each test function, we minimize an objective function which is very similar to the training stage objective function. To test membership in each class, we minimize the sum of the wavelet reconstruction error and a suitable distance between the unknown coefficients and its nearest neighbors in the chosen class. The test function is assigned to the class that yields the minimum value of the objective function. This overall testing procedure is formalized as

$$\arg\min_a \left(\min_{\tilde{\mathbf{c}}} E^a(\tilde{\mathbf{c}}) \right) = \arg\min_a \left(\min_{\tilde{\mathbf{c}}} \|\tilde{\mathbf{f}} - \boldsymbol{\phi}\tilde{\mathbf{c}}\|^2 + \lambda \sum_{i \text{ s.t. } y_i = a} M_i^a \|\tilde{\mathbf{c}} - \mathbf{c}^i\|^2 \right) \quad (19)$$

where $\tilde{\mathbf{f}}$ is the test set function and $\tilde{\mathbf{c}}$ is its vector of reconstruction coefficients. M_i^a is the nearest neighbor in the set of class a patterns. As before, the membership can be "integrated out" to get

$$E^a(\tilde{\mathbf{c}}) = \|\tilde{\mathbf{f}} - \boldsymbol{\phi}\tilde{\mathbf{c}}\|^2 - \frac{\lambda}{\beta} \log \sum_{i \text{ s.t. } y_i = a} \exp\left\{-\beta \|\tilde{\mathbf{c}} - \mathbf{c}^i\|^2\right\}. \quad (20)$$

This objective function can be minimized using methods similar to those used during training. The testing stage algorithm comprises the following steps.

1. Solve $\Gamma(a) = \min_{\tilde{\mathbf{c}}} E^a(\tilde{\mathbf{c}})$ for every class a using the objective function gradient
$$\frac{\partial E^a}{\partial \tilde{\mathbf{c}}} = -2\boldsymbol{\phi}^T\tilde{\mathbf{f}} + 2\boldsymbol{\phi}^T\boldsymbol{\phi}\tilde{\mathbf{c}} + \lambda \sum_{i \text{ s.t. } y_i = a} M_i^a \left(2\tilde{\mathbf{c}} - 2\mathbf{c}^i\right) \quad (21)$$
where
$$M_i^a = \frac{\exp\left\{-\beta\|\tilde{\mathbf{c}} - \mathbf{c}^i\|^2\right\}}{\sum_{j \in \mathcal{N}(a)} \exp\left\{-\beta\|\tilde{\mathbf{c}} - \mathbf{c}^j\|^2\right\}}. \quad (22)$$
2. Assign the label $\tilde{y}$ to $\tilde{\mathbf{f}}$ by finding the class with the smallest value of $\Gamma(a)$, namely $\arg\min_a(\Gamma(a))$.

Table 1. Functional Datasets. Datasets contain a good mix of multiple classes, class imbalances, and varying number of training versus testing curves.

Dataset	Number of Classes	Size of Training Set	Size of Testing Set	Length
Synthetic Control	6	300	300	60
Gun Point	2	50	150	150
ADIAC	37	390	391	176
Swedish Leaf	15	500	625	128
ECG200	2	100	100	96
Yoga	2	300	3000	426
Coffee	2	28	28	286
Olive Oil	4	30	30	570

5 Experiments

In this section, we discuss the performance of the CDI algorithm using publicly available functional datasets, also known as time series datasets from the "UCR Time Series Data Mining Archive" [26]. The multi-class datasets are divided into training and testing sets with detailed information such as the number of classes, number of curves in each of the testing sets and training sets and the length of the curves shown in Table 1. The datasets that we have chosen to run the experiments on range from 2 class datasets—the Gun Point dataset, and up to 37 classes—the ADIAC dataset. Learning is also exercised under a considerable mix of balanced and unbalanced classes, and minimal training versus testing exemplars, all designed to rigorously validate the generalization capabilities of our approach.

For comparison against competing techniques, we selected four other leading methods based on reported results on the selected datasets. Three out of the four algorithms are classification techniques based on support vector machines (SVM) with extensions to Dynamic Time Warping (DTW). DTW has shown to be a very promising similarity measurement for functional data, supporting warping of functions to determine closeness. Gudmundsson *et al.* [27] demonstrate the feasibility of the DTW approach to get a positive semi-definite kernel for classification with SVM. The approach in Zhang *et al.* [28] is one of many that use a vectorized method to classify functional data instead of using functional properties of the dataset. They develop several kernels for SVM known as elastic kernels—Gaussian elastic metric kernel (GEMK) to be exact and introduce several extensions to GEMK with different measurements. In [29], another SVM classification technique with a DTW kernel is employed but this time a weight is added to the kernel to provide more flexibility and robustness to the kernel function. Prekopcsák *et al.* [21] do not utilize the functional properties of the data. Instead they learn a Mahalanobis metric followed by standard nearest neighbors. For brevity, we have assigned the following abbreviations to these techniques: SD [27], SG [28], SW [29], and MD [21]. In addition to these published works, we also evaluated a standard kNN approach directly on the

Table 2. Experimental parameters. Free parameter settings via cross-validation for each of the datasets. $|\cdot|$ represents set cardinality.

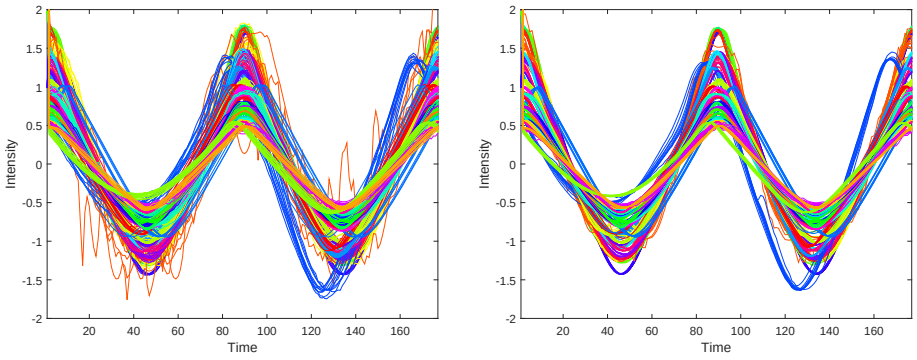
Dataset	λ	μ	$\lvert\mathcal{N}\rvert$	$\lvert\mathcal{M}\rvert$
Synthetic Control	2	0.01	10	5
Gun Point	0.1	0.001	7	2
ADIAC	0.8	0.012	5	5
Swedish Leaf	0.1	0.009	7	2
ECG200	0.9	0.01	3	3
Yoga	0.08	0.002	2	2
Coffee	0.1	0.005	1	1
Olive Oil	0.1	0.005	1	1

wavelet coefficients obtained from eq. 2, i.e. direct representation of functions in a wavelet basis without neighborhood gerrymandering. This was done so that we can comprehensively evaluate if the neighborhood adaptation aspect of CDI truly impacted generalization (with this approach abbreviated as kNN).

A k-fold cross-validation is performed on the training datasets to find the optimal values for each of our free parameters (λ and μ being the most prominent). Since the datasets were first standardized (mean subtraction, followed by standard deviation normalization), the free parameters λ and μ became more uniform across all the datasets. λ ranges from (0.05, 2.0) while μ ranges from $(10^{-3}, 0.01)$. Table 2 has detailed information on the optimal parameters found for each of the datasets. In all our experiments, β is set to 1 and Daubechies 4 (DB4) at $j_0 = 0$ was used as the wavelets basis (i.e. only scaling functions used). We presently do not investigate the effects of β on the classification accuracy as we perform well with it set at unity. The comprehensive results are given in Table 3, with the error percentage being calculated per the usual:

$$\text{Error} = 100\frac{\#\text{ of misclassified curves}}{\text{Total Number of Curves}}. \tag{23}$$

The experiments show very promising results for the proposed CDI method in comparison with the other algorithms, with our error rates as good or better in most datasets. CDI performs best on the ADIAC dataset compared to the other techniques, with an order of magnitude improvement over the current state-of-the-art. This is a particularly difficult dataset having 37 classes where the class sizes are very small, only ranging from 4 to 13 curves. Figure 2(a) illustrates all original curves from the 37 classes which are very similar to each other. Having many classes with only a few training samples in each presents a significant classification challenge and correlates with why the competing techniques have a high classification error. In Figure 2(b), we show how CDI brings curves within the same class together making them more "pure" (such as the orange curves) while also managing to separate classes from each other. Regularized, discriminative learning also has the added benefit of smoothing the functions. The competing SVM-based approaches suffer in accuracy due to the heavily unbalanced classes. In some datasets (e.g. Swedish Leaf or Yoga) where we are not the leader, we



(a) Original ADIAC Curves

(b) Discriminatively Interpolated ADIAC Curves

Fig. 2. ADIAC dataset 37 classes - 300 Curves. The proposed CDI method is an order of magnitude better than the best reported competitor. Original curves in (a) are uniquely colored by class. The curves in (b) are more similar to their respective classes and are smoothed by the neighborhood regularization—unique properties of the CDI.

Table 3. Classification Errors. The proposed CDI method achieves state-of-the-art performance on half of the datasets, and is competitive in almost all others. The ECG200 exception, with kNN outperforming everyone, is discussed in text.

Dataset	SD [27]	SG [28]	SW [29]	MD [21]	kNN	CDI
Synthetic Control	0.67 (2)	0.7 (4)	0.67 (2)	1 (5)	9.67 (6)	0.33 (1)
Gun Point	4.67 (3)	0 (1)	2 (2)	5 (4)	9 (6)	6 (5)
ADIAC	32.48 (5)	24(3)	24.8 (4)	23 (2)	38.87 (6)	3.84 (1)
Swedish Leaf	14.72 (3)	5.3 (1)	22.5 (6)	15 (4)	16.97 (5)	8.8 (2)
ECG200	16 (6)	7 (2)	14 (5)	8 (3)	0 (1)	8 (3)
Yoga	16.37 (5)	11 (2)	18 (6)	16 (4)	10 (1)	15 (3)
Coffee	10.71 (4)	0 (1)	23 (5)	-	8.67 (3)	0 (1)
Olive Oil	13.33 (4)	10 (1)	17.3 (5)	13 (3)	20.96 (6)	10 (1)

are competitive with the others. Comparison with the standard kNN resulted in valuable insights. CDI fared better in 6 out of the 8 datasets, solidifying the utility of our push-pull neighborhood adaptation (encoded by the learned μ and λ), clearly showing CDI is going beyond vanilla kNN on the coefficient vectors. However, it is interesting that in two of the datasets that kNN beat not only CDI but all other competitors. For example, kNN obtained a perfect score on ECG200. The previously published works never reported a simple kNN score on this dataset, but as it can occur, a simple method can often beat more advanced methods on particular datasets. Further investigation into this dataset showed that the test curves contained variability versus the training, which may have contributed to the errors. Our error is on par with all other competing methods.

6 Conclusion

The large margin k-nearest neighbor functional data classification framework proposed in this work leverages class-specific neighborhood relationships to *discriminatively interpolate* functions in a manner that morphs curves from the same class to become more similar in their appearance, while simultaneously pushing away neighbors from competing classes. Even when the data naturally occur as functions, the norm in machine learning is to move to a feature vector representation, where the burden of achieving better performance is transferred from the representation to the selection of discriminating features. Here, we have demonstrated that such a move can be replaced by a more principled approach that takes advantage of the functional nature of data.

Our CDI objective uses a wavelet expansion to produce faithful approximations of the original dataset and concurrently incorporates localized push-pull terms that promote neighborhood class purity. The detailed training optimization strategy uses a familiar iterative, alternating descent algorithm whereby first the coefficients of the basis expansions are adapted in the context of their labeled, softmax-weighted neighbors, and then, the curves' k-neighborhoods are updated. Test functional data are classified by the cost to represent them in each of the morphed training classes, with a minimal cost correct classification reflecting both wavelet basis reconstruction accuracy and nearest-neighbor influence. We have extensively validated this simple, yet effective, technique on several datasets, achieving competitive or state-of-the-art performance on most of them. In the present work, we have taken advantage of the interpolation characteristic of functions. In the future, we intend to investigate other functional properties such as derivatives, hybrid functional-feature representations, and extensions to higher dimensional functions such as images. We also anticipate improvements to our optimization strategy which was not the focus of the present work.

References

1. Ramsay, J., Silverman, B.: Functional Data Analysis. 2nd edn. Springer (2005)
2. Hall, P., Hosseini-Nasab, M.: On properties of functional principal components analysis. Journal of the Royal Statistical Society: Series B (Statistical Methodology) **68**(1) (2006) 109–126
3. Garcia, M.L.L., Garcia-Rodenas, R., Gomez, A.G.: K-means algorithms for functional data. Neurocomputing **151, Part 1** (March 2015) 231–245
4. Rossi, F., Villa, N.: Support vector machine for functional data classification. Neurocomputing **69**(7-9) (2006) 730–742
5. Rossi, F., Delannay, N., Conan-Guez, B., Verleysen, M.: Representation of functional data in neural networks. Neurocomputing **64** (March 2005) 183–210
6. Weinberger, K.Q., Blitzer, J., Saul, L.K.: Distance metric learning for large margin nearest neighbor classification. In: Advances in Neural Information Processing Systems 18 (NIPS), MIT Press (2006) 1473–1480
7. Tarlow, D., Swersky, K., Charlin, L., Sutskever, I., Zemel, R.S.: Stochastic k-neighborhood selection for supervised and unsupervised learning. Proc. of the 30th International Conference on Machine Learning (ICML) **28**(3) (2013) 199–207

8. Trivedi, S., McAllester, D., Shakhnarovich, G.: Discriminative metric learning by neighborhood gerrymandering. In: Advances in Neural Information Processing Systems (NIPS) 27. (2014) 3392–3400
9. Zhu, P., Hu, Q., Zuo, W., Yang, M.: Multi-granularity distance metric learning via neighborhood granule margin maximization. Inf. Sci. **282** (October 2014) 321–331
10. Keysers, D., Deselaers, T., Gollan, C., Ney, H.: Deformation models for image recognition. PAMI, IEEE Transactions on **29**(8) (Aug 2007) 1422–1435
11. Kreyszig, E.: Introductory Functional Analysis with Applications. John Wiley and Sons (1978)
12. Daubechies, I.: Ten Lectures on Wavelets. CBMS-NSF Reg. Conf. Series in Applied Math. SIAM (1992)
13. Abraham, C., Cornillon, P.A., Matzner-Lober, E., Molinari, N.: Unsupervised curve clustering using B-splines. Scandinavian J. of Stat. **30**(3) (2003) 581–595
14. Biau, G., Bunea, F., Wegkamp, M.: Functional classification in Hilbert spaces. Information Theory, IEEE Transactions on **51**(6) (June 2005) 2163–2172
15. Antoniadis, A., Brossat, X., Cugliari, J., Poggi, J.M.: Clustering functional data using wavelets. International Journal of Wavelets, Multiresolution and Information Processing **11**(1) (2013) 1350003 (30 pages)
16. Berlinet, A., Biau, G., Rouviere, L.: Functional supervised classification with wavelets. Annale de l'ISUP **52** (2008)
17. Alonso, A.M., Casado, D., Romo, J.: Supervised classification for functional data: A weighted distance approach. Computational Statistics & Data Analysis **56**(7) (2012) 2334–2346
18. López-Pintado, S., Romo, J.: Depth-based classification for functional data. DIMACS Series in Discrete Mathematics and Theoretical Comp. Sci. **72** (2006) 103
19. Domeniconi, C., Gunopulos, D., Peng, J.: Large margin nearest neighbor classifiers. Neural Networks, IEEE Transactions on **16**(4) (July 2005) 899–909
20. Globerson, A., Roweis, S.T.: Metric learning by collapsing classes. In: Advances in Neural Information Processing Systems 18. MIT Press (2006) 451–458
21. Prekopcsák, Z., Lemire, D.: Time series classification by class-specific Mahalanobis distance measures. Adv. in Data Analysis and Classification **6**(3) (2012) 185–200
22. Berndt, D., Clifford, J.: Using dynamic time warping to find patterns in time series. In: AAAI Workshop on Knowledge Discovery in Databases. (1994) 229–248
23. Vapnik, V.N.: The nature of statistical learning theory. Springer, New York, NY, USA (1995)
24. Hinton, G., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural Computation **18**(7) (2006) 1527–1554
25. Yuille, A., Kosowsky, J.: Statistical physics algorithms that converge. Neural Computation **6**(3) (1994) 341–356
26. Keogh, E., Xi, X., Wei, L., Ratanamahatana, C.: The UCR time series classification/clustering homepage. `http://www.cs.ucr.edu/~eamonn/time_series_data/` (2011)
27. Gudmundsson, S., Runarsson, T., Sigurdsson, S.: Support vector machines and dynamic time warping for time series. In: IEEE International Joint Conference on Neural Networks (IJCNN). (June 2008) 2772–2776
28. Zhang, D., Zuo, W., Zhang, D., Zhang, H.: Time series classification using support vector machine with Gaussian elastic metric kernel. In: 20th International Conference on Pattern Recognition (ICPR). (2010) 29–32
29. Jeong, Y.S., Jayaramam, R.: Support vector-based algorithms with weighted dynamic time warping kernel function for time series classification. Knowledge-Based Systems **75** (December 2014) 184–191