Interpretable Multiple-Kernel Prototype Learning for Discriminative Representation and Feature Selection

Abstract

Prototype-based methods are of the particular interest for domain specialists and practitioners as they summarize a dataset by a small set of representatives. Therefore, in a classification setting, interpretability of the prototypes is as significant as the prediction accuracy of the algorithm. Nevertheless, the state-of-the-art methods make inefficient trade-offs between these concerns, especially if the given data has a (multiple-)kernel-based representation. In this paper, we propose a novel interpretable multiple-kernel prototype learning (IMKPL) to construct highly interpretable prototypes in the feature space which are also efficient for discriminative representation of the data. Our method focuses on the local discrimination of the classes in the feature space and shaping the prototypes based on condense class-homogeneous neighborhoods of data. Besides, IMKPL learns a combined embedding in the feature space in which the above objectives are better fulfilled. When the base kernels coincide with the data dimensions, this embedding results in a discriminative features selection. We evaluate IMKPL on several benchmarks from different domains which demonstrate its superiority to the related state-of-the-art methods regarding both interpretability and discriminative representation.

1 Introduction

The cognitive psychology claims that humans categorize the data in their minds based on a set of common prototypes (or examples) for each class of data [Rosch, 1975]. Inspired by that, prototype-based models are among popular machine learning methods used by domain specialists regarding many applications. A supervised prototype-based algorithm finds representatives in the input space such that the class label of the data samples are predicted based on their distances to these prototypes [Friedman *et al.*, 2001]. Apart from this simplicity, their decisions are highly explainable (e.g., by a practitioner) throughout the direct inspection of the prototypes to which each test data is assigned [Hammer *et al.*, 2014]. The most popular prototype-based approaches are the self-organizing map (SOM) [Kohonen, 1995], the nearest prototype classifier [Friedman *et al.*, 2001], the learning vector quantization (LVQ) [Biehl *et al.*, 2007], and the prototype selection algorithm (PS) [Bien *et al.*, 2011], where the first one belongs to the unsupervised methods and the rest to the supervised problems.

Besides the discriminative performance of these methods, it is always a main concern to learn interpretable representatives such that each prototype can represent a specific class or a condense neighborhood of data in the input space [Friedman *et al.*, 2001; Biehl *et al.*, 2007; Bien *et al.*, 2011]. Usually, this concern induces a trade-off between the discriminative quality and the interpretative value of the prototypes [Bien *et al.*, 2011] In addition, regardless of the reported efficiency and simplicity of these algorithms, their designs generally face difficulties when the classes contain distinct sub-classes and are linearly inseparable (e.g., XOR dataset).

In nowadays applications, it is common to observe non-Euclidean data settings, such as time-series, sequences, or strings. As a an effective solution, a relational representation of the data is obtained using non-Euclidean pairwise dissimilarities between the data points. Consequently, the kernel variants of prototype-based learning methods are designed by assuming a corresponding implicit mapping to a high-dimensional reproducing kernel Hilbert space (RKHS). Among those methods, kernel K-means [Shawe-Taylor and Cristianini, 2004] and kernelized-generalized LVQ (KGLVQ) [Schleif *et al.*, 2011] represent the well-known unsupervised and supervised prototype-based algorithms respectively. Nevertheless, kernel-based methods generally have weak interpretations of the prototypes as they may be constructed by a wide inter-class set of data points in the space [Nova and Estévez, 2014]. In contrast, some dissimilarity-based algorithms such as PS [Bien *et al.*, 2011] try to select class-specific exemplars based on their distances to the members of their corresponding classes. Although this choice increases the interpretability of the prototypes, it limits their discriminative performance in comparison.

From another point of view, it is shown in [Bach *et al.*, 2004] that applying different kernels on the inputs can result in multiple-kernel (MK) representation of the data which carries non-redundant pieces of information about significant properties of the data. Consequently, multiple-kernel learning

(MKL) approaches are designed to find an effective weighted combination of these base kernels which enhances the classification performance. Besides, by taking into account only the non-zero weights that construct this combined RKHS, we can obtain a discriminative feature selection [Dileep and Sekhar, 2009; Wang *et al.*, 2014]. Nevertheless, the majority of MKL methods have a non-realistic assumption about existing a combined RKHS in which the data classes are linearly (globally) separable [Shawe-Taylor and Cristianini, 2004]. To our knowledge, there is no MKL which is designed in particular for prototype-based models and with a focus on local separation of the classes.

Dictionary learning (DL) methods try to find a set of dictionary atoms in the input space which reconstruct each data sample via a sparse weighted combination (sparse code) of these atoms [Rubinstein *et al.*, 2008]. These sparse representations can capture essential intrinsic characteristics of the dataset [Kim *et al.*, 2010] that are consistent between the training and testing distributions. A popular stream in DL belongs to the approaches which also respect the label information in the data [Guo *et al.*, 2016]. Furthermore, some recent works similar to [Wang *et al.*, 2014; Zhu *et al.*, 2017] joined MKL with DL in order to improve the reconstruction and discrimination quality of the dictionary by learning it on an efficient combined RKHS. Although one can consider the dictionary atoms as a set of representative prototypes, none of the multiple-kernel (or single-kernel) dictionary learning (MKDL) algorithms are designed with an explicit focus on that goal. Therefore, they would either suffer from the interpretation of their dictionary atoms, or these learned atoms cannot efficiently represent the classes in a discriminative way.

Our Contributions: In this paper, we propose the interpretable multiple-kernel prototype learning algorithm (IMKPL) to obtain a discriminative prototype-based model for multiple-kernel data representations. We construct our framework upon the basic model of multiple-kernel dictionary learning, such that each data sample can be represented by a set of prototypes. However, beyond the typical classification objective, IMKPL has a specific focus on the interpretability of the learned prototypes and the local representation of the data. More specifically, our work contributes to the current state-of-the-art in the following aspects:

- We extend the application of prototype-based learning to MK data representations.
- The prototypes in our method are efficiently interpretable based on the class-specific local neighborhoods they represent.
- Our prototype-learning framework focuses on local discrimination of the classes in the combined RKHS to mitigate their global overlapping.

In the next section, we discuss the relevant work. Our proposed IMKPL algorithm is introduced in Section 3, and its optimization scheme is explained in Section 4. The empirical evaluations are presented in Section 5, and the conclusion of work is provided in the final section.

2 Preliminaries

We define the data matrix $\mathbf{X} = [\vec{x}_1, ..., \vec{x}_N] \in \mathbb{R}^{d\times N}$ denoting N training samples, and $\mathbf{L} = [\vec{l}_1, \dots, \vec{l}_N] \in \mathbb{R}^{p\times N}$ indicates the corresponding label matrix in a p-class setting. Each $\vec{l}_i$ is a zero vector except $l_{qi} = 1$ if $\vec{x}_i$ is in class q. As a convention, $\vec{v}_i$ denotes the i-th column in a given matrix $\mathbf{V}$, v_{ji} and v_j refer to the j-th entries in vectors $\vec{v}_i$ and $\vec{v}$ respectively, and $\vec{v}^j$ denotes the j-th row in $\mathbf{V}$.

Assume there exists f implicit non-linear mappings $\{\Phi_i(\vec{x})\}_{i=1}^{f}$ projecting $\mathbf{X}$ onto f individual RKHSs [Bach *et al.*, 2004]. Then, it is possible to compute the weighted kernel function $\hat{\Phi}(\vec{x})$ based on the following scaling of the feature space

$$\hat{\Phi}(\vec{x}) = [\sqrt{\beta_1}\Phi_1^\top(\vec{x}), \cdots, \sqrt{\beta_f}\Phi_f^\top(\vec{x})]^\top, \tag{1}$$

where $\vec{\beta}$ is the combination vector which results in a combined RKHS corresponding to $\hat{\Phi}(\vec{x})$. By assuming each $\Phi_i(\vec{x})$ is related to a base kernel function $\mathcal{K}_i(\vec{x}_t, \vec{x}_s) = \Phi_i^\top(\vec{x}_t)\Phi_i(\vec{x}_s)$, we compute the weighted kernel $\hat{\mathcal{K}}$ corresponding to $\hat{\Phi}(\vec{x})$ as

$$\hat{\mathcal{K}}(\vec{x}_t, \vec{x}_s) = \sum_{i=1}^{f} \beta_i \mathcal{K}_i(\vec{x}_t, \vec{x}_s) = \hat{\Phi}(\vec{x}_t)^\top \hat{\Phi}(\vec{x}_s). \tag{2}$$

By applying a non-negativity constraint on the entries of $\vec{\beta}$, the resulted kernel weights can be interpreted as the relative importance of each base kernel in the obtained MK representation $\hat{\Phi}(\vec{x})$ [Gönen and Alpaydın, 2011]. For the ease of reading, we denote the Gram matrix $\hat{\mathcal{K}}(\mathbf{X}, \mathbf{X})$ and the vector $\hat{\mathcal{K}}(\vec{x}_i, \mathbf{X})$ simply by $\hat{\mathcal{K}}$ and $\hat{\mathcal{K}}(i, :)$ respectively.

The goal of a multiple-kernel dictionary learning method is to find an optimal MK dictionary $\hat{\Phi}(\mathbf{D})$ in the combined RKHS to reconstruct the input signals as $\hat{\Phi}(\mathbf{X}) \approx \hat{\Phi}(\mathbf{D})\mathbf{\Gamma}$ in this space. The columns of $\mathbf{\Gamma} = [\vec{\gamma}_1, \dots, \vec{\gamma}_N] \in \mathbb{R}^{c\times N}$ contain the corresponding sparse codes which are desired to have sparse non-zeros entries [Aharon *et al.*, 2006]. A basic MKDL framework can be formulated as a variant of the following

$$\begin{aligned} \min_{\mathbf{\Gamma},\mathbf{U}} \quad & \|\hat{\Phi}(\mathbf{X}) - \hat{\Phi}(\mathbf{X})\mathbf{U}\mathbf{\Gamma}\|_F^2 \\ \text{s.t.} \quad & \|\vec{\beta}\|_1 = 1,\ \beta_i \in \mathbb{R}_+,\ \|\vec{\gamma}_i\|_0 \le T_0, \end{aligned} \tag{3}$$

where the objective term $\mathcal{J}_{rec} = \|\hat{\Phi}(\mathbf{X}) - \hat{\Phi}(\mathbf{X})\mathbf{U}\mathbf{\Gamma}\|_F^2$ measures the reconstruction quality of the data in RKHS. In Eq. (3), $\|.\|_F^2$ denotes the F-norm, and the dictionary is modeled as $\hat{\Phi}(\mathbf{D}) = \hat{\Phi}(\mathbf{X})\mathbf{U}$ where $\mathbf{U} \in \mathbb{R}^{N\times c}$ is the dictionary matrix [Nguyen *et al.*, 2013]. Hence, each column of $\mathbf{U}$ defines a linear combination of data points in the feature space while it can be optimized in the input space. The constraint $\|\vec{\beta}\|_1 = 1$ in Eq. (3) applies an affine combination of base kernel which prevents $\|\vec{\beta}\|_1$ from shrinking in favor of the trivial solution of Eq. (3). The constraint $\|\vec{\gamma}_i\|_0 \le T_0$ applies a T_0 sparsity limit on the cardinality of each encoding $\vec{\gamma}_i$. The role of $\vec{\beta}$ in Eq. (1) can enhance the discriminative power of the learned dictionary atoms $\{\hat{\Phi}(\mathbf{X})\vec{u}_i\}_{i=1}^c$ by increasing the dissimilarity between the different-label columns in $\hat{\Phi}(\mathbf{X})$.

Although $\mathcal{J}_{rec}$ is a common term among most of the existing MKDL methods, they have significant differences based on the formulation of their MKL or DL parts. In approaches similar to [Thiagarajan *et al.*, 2014], the $\vec{\beta}$ in Eq. (3) is optimized to improve the linear separability of the classes in RKHS. In contrast, [Shrivastava *et al.*, 2015] jointly optimize $\{\mathbf{U}, \vec{\beta}\}$ in Eq. (3) by pre-defining class-isolated sub-dictionaries in $\mathbf{U}$ and enforcing the orthogonality of one class to the dictionaries of other classes in RKHS. [Zhu *et al.*, 2017] utilized an analysis-synthesis formulation of Eq. (3) and also applies a low-rank constraint on $\mathbf{\Gamma}$.

Distinguishing from the MKDL frameworks, our work specifically focuses on shaping the dictionary atoms as interpretable prototypes which can represent local parts of the data and can discriminate them regarding their class labels. The empirical evaluations in Section 5 show that neither of these major MKDL algorithms can adequately fulfill the above prototype-based objectives.

3 Interpretable Multiple-Kernel Prototype Learning

We want to learn a MK dictionary that its constituent prototypes (atoms) reconstruct the given data while presenting discriminative, interpretable characteristics regarding the class labels. In more explicit terms, we want to fulfill the following specific objectives:

O1: Assigning prototypes to the local neighborhoods in the classes to efficiently discriminate them in RKHS regarding their class labels (Figure 2-(d)).

O2: Learning prototypes which are interpretable based on the condense class-specific neighborhoods that they represent (Figure 1-(b))

O3: Obtaining an efficient MK representation of the data to assist the above objectives and to improve the local separation of the classes in the resulted RKHS (Figure 2).

Definition 1 *Each $\vec{x}$ is represented by the prototype set $\{\hat{\Phi}(\mathbf{X})\vec{u}_i\}_{i\in I}$ in the combined RKHS, if $\|\hat{\Phi}(\vec{x}) - \hat{\Phi}(\mathbf{X})\mathbf{U}\vec{\gamma}\|_2^2 < \epsilon$ for a small $\epsilon > 0$ and $\gamma_i \neq 0, \forall i \in I$.*

Based on the the Definition 1, we call $\{\vec{u}_i\}_{i=1}^c$ as the prototype vectors which represent the columns of $\hat{\Phi}(\mathbf{X})$. Accordingly, we propose the interpretable multiple-kernel prototype learning algorithm addressing the above objectives. IMKPL has the the novel optimization scheme of:

$$\begin{aligned} \min_{\vec{\beta},\mathbf{\Gamma},\mathbf{U}} \quad & \|\hat{\Phi}(\mathbf{X}) - \hat{\Phi}(\mathbf{X})\mathbf{U}\mathbf{\Gamma}\|_F^2 + \lambda\mathcal{J}_{dis} + \mu\mathcal{J}_{ls} + \tau\mathcal{J}_{ip} \\ \text{s.t.} \quad & \|\vec{\gamma}_i\|_0 < T_0,\ \|\vec{\beta}\|_1 = 1,\ \|\hat{\Phi}(\mathbf{X})\vec{u}_i\|_2^2 = 1, \\ & \|\vec{u}_i\|_0 \leq T_0,\ u_{ji}, \beta_i, \gamma_{ji} \in \mathbb{R}_+, \end{aligned} \tag{4}$$

in which λ and μ are trade-off scalars, and $\forall i$, the cardinalities of $\vec{\gamma}_i$ and $\vec{u}_i$ are limited by the constant T_0. The cardinality and non-negativity constraints on $\{\mathbf{U}, \mathbf{\Gamma}\}$ coincide with the dictionary structure $\hat{\Phi}(\mathbf{X})\mathbf{U}$ [Hosseini and Hammer, 2018]. They motivate each prototype $\hat{\Phi}(\mathbf{X})\vec{u}_i$ to be formed by sparse contributions from similar training samples in $\hat{\Phi}(\mathbf{X})$ to increase their interpretability [Bien *et al.*, 2011]. Although each $\vec{u}_i$ is loosely shaped from the local neighborhoods in RKHS, it cannot fulfill the objectives **O1** and **O2** on its own (Figure 1-(a)). Also, having $\|\hat{\Phi}(\mathbf{X})\vec{u}_i\|_2^2 = 1$ prevents the solution of $\vec{u}_i$ from being degenerated [Rubinstein *et al.*, 2008].

In Eq. (4), we introduce the novel terms $\mathcal{J}_{dis}$, $\mathcal{J}_{ls}$, and $\mathcal{J}_{ip}$ in order to fulfill the the objectives **O1-O3**. In the following sub-sections, we discuss these terms in detail.

3.1 Discriminative Loss $\mathcal{J}_{dis}(\mathbf{U}, \mathbf{\Gamma}, \vec{\beta})$

We can rewrite the reconstruction of $\vec{x}$ in the combined RKHS as $\hat{\Phi}(\mathbf{X})\mathbf{U}\vec{\gamma} = \hat{\Phi}(\mathbf{X})\vec{h}$, where $\vec{h} \in \mathbb{R}^N$ reconstructs $\hat{\Phi}(\vec{x})$ based on other samples in $\hat{\Phi}(\mathbf{X})$. Hence, aiming for **O1**, we learn the prototype vectors $\vec{u}_i$ such that they represent each $\hat{\Phi}(\vec{x})$ with a corresponding vector $\vec{h}$ that has non-zero values only regarding the local same-class neighbors of $\hat{\Phi}(\vec{x})$. Accordingly, we define the loss term $\mathcal{J}_{dis}$ as:

$$\begin{aligned} & \mathcal{J}_{dis}(\mathbf{U}, \mathbf{\Gamma}, \vec{\beta}) = \\ & \tfrac{1}{2}\sum_{i=1}^{N}[\sum_{s=1}^{N} \vec{u}^s\vec{\gamma}_i(\vec{l}_i^\top\vec{l}_s\|\hat{\Phi}(\vec{x}_i) - \hat{\Phi}(\vec{x}_s)\|_2^2 + \|\vec{l}_i - \vec{l}_s\|_2^2)]. \end{aligned} \tag{5}$$

Proposition 1 *$\mathcal{J}_{dis}$ has its minima if each $\hat{\Phi}(\vec{x})$ is reconstructed by a corresponding $\vec{\gamma}$ with non-zero entries only related to columns of $\mathbf{U}$ that are formed by the same-label neighbors of $\vec{x}$ in RKHS.*

Proof 1 *The detailed proof is provided in the supplementary materials[1]. However, as a sketch proof, Eq. (5) consists of two parts, one motivates $\hat{\Phi}(\vec{x})$ to be related to its close same-label neighbors (throughout $\mathbf{U}$) while the other prevents it from being related to data samples from other classes.*

Based on the Proposition 1, $\mathcal{J}_{dis}$ enforces the optimization of Eq. (4) to learn $\mathbf{U}$ such that each prototype $\hat{\Phi}(\mathbf{X})\vec{u}_i$ is shaped by a concatenated neighborhood in one class and efficiently represents (Definition 1) its nearby samples of that class. This task directly coincides with the objective **O1**. However, the formulation of $\mathcal{J}_{dis}$ is still flexible to tolerate small cross-class contributions in each $\vec{u}_i$ to facilitate the possible inter-class overlapping. For example, in Figure 1-(b), a square sample has contributed to the formation of $\vec{u}_1$ (with a small corresponding u_{j1}) based on its similarity to $\hat{\Phi}(\vec{x})$. Nevertheless, $\vec{u}_1$ can still be interpreted as the representative of a small neighborthood in the *circle* class.

3.2 Interpretability Loss $\mathcal{J}_{ip}(\mathbf{U})$

Definition 2 *A prototype $\hat{\Phi}(\mathbf{X})\vec{u}_i$ can be interpreted as a local representative of the class q, if $\forall t|u_{ti} \neq 0$, $\vec{x}_t$ belongs to a condense neighborhood in RKHS and also $\frac{\vec{l}^q\vec{u}_i}{\|\mathbf{L}\vec{u}_i\|_1} \approx 1$.*

Definition 2 mathematically Although $\mathcal{J}_{dis}$ can result in such interpretable prototypes provided if the neighborhoods in RKHS are smooth labeling, when we have overlapping between the neighborhoods of different classes, it is common to

[1]?? dropbox JCAI2019-supplementaries

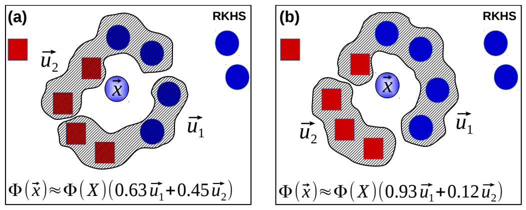
(a)
RKHS
$\vec{u}_2$
$\vec{x}$
$\vec{u}_1$
$\Phi(\vec{x})\approx\Phi(X)(0.63\vec{u}_1+0.45\vec{u}_2)$
(b)
RKHS
$\vec{x}$
$\vec{u}_2$
$\vec{u}_1$
$\Phi(\vec{x})\approx\Phi(X)(0.93\vec{u}_1+0.12\vec{u}_2)$

Figure 1: The effect of $\mathcal{J}_{dis}$ in Eq. (4). (a): When $\lambda = 0$, the prototypes $(\vec{u}_1, \vec{u}_2)$ (the hatched selections) are shaped and reconstruct $\hat{\Phi}(\vec{x})$ by its neighboring samples from both of the classes (circles and squares). (b): When $\lambda \neq 0$, these prototypes are formed s.t. $\hat{\Phi}(\vec{x})$ is approximately represented by $\vec{u}_1$ mostly using its local, same-class neighbors (circles).

observe more than one large entries in $\vec{s}$ (practically 2 or 3). Therefore, to explicitly aim for the class-based interpretation of the prototypes we define

$$\mathcal{J}_{ip}(\mathbf{U}) = \|\mathbf{L}\mathbf{U}\|_1, \tag{6}$$

which tries to minimize $\mathbf{L}\vec{u}_i$ for each prototype vector. This additional term, along with the effect of $\mathcal{J}_{dis}$ in Eq. (4), results in a significantly sparse $\vec{s} = \mathbf{L}\vec{u}_i$, such that $\frac{\vec{l}^q \vec{u}_i}{\|\mathbf{L}\vec{u}_i\|_1}$ in Definition 2 obtains a value close to 1. Hence, based on Definition 2, $\mathcal{J}_{ip}$ improves the interpretation of each $\hat{\Phi}(\mathbf{X})\vec{u}_i$.

3.3 Local-Separation Loss $\mathcal{J}_{ls}(\vec{\beta})$

According to Eq. (1) and Eq. (4), the weighting vector $\vec{\beta}$ is already incorporated into $\mathcal{J}_{rec}$ and $\mathcal{J}_{dis}$ via its role in $\hat{\Phi}(\mathbf{X})$. Hence, minimizing Eq. (4) w.r.t to $\vec{\beta}$ optimizes the combined embedding in the features spaces to better fulfill the objectives **O1** and **O2**. Besides, as an efficient complement to the above, we optimize $\vec{\beta}$ to locally separate the classes in k-size neighborhoods in the feature space. To that aim we propose $\mathcal{J}_{ls}$ as the following novel, convex loss term:

$$\mathcal{J}_{ls}(\vec{\beta}) = \sum_{i=1}^{N} \big[\textstyle\sum_{s\in\mathcal{N}_i^k} \|\hat{\Phi}(\vec{x}_i) - \hat{\Phi}(\vec{x}_s)\|_2^2 + \sum_{s\in\overline{\mathcal{N}_i^k}} \hat{\Phi}(\vec{x}_i)^\top \hat{\Phi}(\vec{x}_s) \big], \tag{7}$$

in which $\mathcal{N}_i^k$ is the set of indexes specifying the k-nearest neighbors of $\vec{x}_i$ in RKHS with labels equal to $\vec{l}_i$, and $\overline{\mathcal{N}_i^k}$ is its corresponding k-size set for the samples $\vec{x}_s$ such that $\vec{l}_s \neq \vec{l}_i$.

From the mathematical perspective, Eq. (7) obtains its minima for each $\vec{x}_i$ when: (1) the summation of its distances to the nearby same-label points is minimized, and (2) it is dissimilar from the nearby data of other classes (Figure 2-b). Therefore, optimizing $\vec{\beta}$ w.r.t $\mathcal{J}_{mk}$ in conjunction with other terms in Eq. (4) makes the classes locally condensed and separated from each other. Such a solution results in leaning better interpretable, discriminative prototypes (Figure 2-(d)). In the next section, we explain how to solve the optimization problem of Eq. (4) efficiently.

4 Optimization Scheme of IMKPL

To optimize the parameters $\{\mathbf{U}, \mathbf{\Gamma}, \vec{\beta}\}$ of the problem in Eq. (4), we adopt a 3-step alternating optimization scheme.

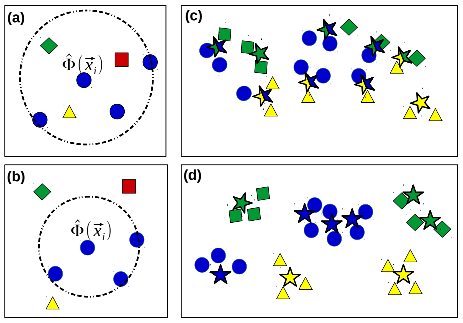

Figure 2: Effect of $\mathcal{J}_{ls}$ on local separation of each $\hat{\Phi}(\vec{x}_i)$ from its different-label neighbors in RKHS when $k = 4$ (**b** compared to **a**), which concentrates the classes locally (**d** compared to **c**) and improves the interpretation of the prototypes $\{\hat{\Phi}(\mathbf{X})\vec{u}_i\}_{i=1}^c$ (the stars) by the class-neighborhood to which they are assigned (their colors).

Hence, at each of the following steps, we update only one of the parameters while fixing the two others. The details regarding the derivation of the corresponding optimization schemes are provided in the supplementary material.

4.1 Updating the Matrix of Sparse Codes $\mathbf{\Gamma}$

By fixing $\{\mathbf{U}, \vec{\beta}\}$, using Eq. (2), and removing the constant terms, we rewrite the problem in Eq. (4) w.r.t. each $\vec{\gamma}_i$ as:

$$\begin{aligned} \min_{\vec{\gamma}_i} \quad & \vec{\gamma}_i^\top (\mathbf{U}^\top \hat{\mathcal{K}} \mathbf{U}) \vec{\gamma}_i + [\lambda \tilde{\mathcal{K}}(i,:) - 2\hat{\mathcal{K}}(i,:)]\mathbf{U}\vec{\gamma}_i \\ \text{s.t.} \quad & \|\vec{\gamma}_i\|_0 < T_0, \quad \gamma_{ji} \in \mathbb{R}_+, \end{aligned} \tag{8}$$

where $\tilde{\mathcal{K}} = \mathbf{1} - (\mathbf{L}^\top \mathbf{L}) \odot \hat{\mathcal{K}}$ while " $\odot$ " denotes the Hadamard product operator. This optimization problem is a non-negative quadratic programming with a cardinality constraint on $\vec{\gamma}_i$. The matrix $\mathbf{U}^\top \hat{\mathcal{K}} \mathbf{U}$ is positive semidefinite (PSD) because $\hat{\mathcal{K}}$ is PSD and $\mathbf{U}$ is non-negative, hence Eq. (8) is a convex problem and can be optimized by the Non-Negative Quadratic Pursuit (NQP) algorithm [2]. NQP generalizes the Matching Pursuit approach [Aharon *et al.*, 2006] to non-negative quadratic problems similar to Eq. (8). Therefore, we approximate the columns of $\mathbf{\Gamma}$ individually by the NQP algorithm.

4.2 Updating Prototype Matrix $\mathbf{U}$

Similar to the approximation of $\mathbf{\Gamma}$, the prototype vectors $\vec{u}_i$ are updated sequentially. We re-formulate the reconstruction objective $\mathcal{J}_{rec}$ in Eq. (4) as

$$\|\hat{\Phi}(\mathbf{X})\mathbf{E}_i - \hat{\Phi}(\mathbf{X})\vec{u}_i\vec{\gamma}^i\|_F^2, \quad \mathbf{E}_i = (\mathbf{I} - \textstyle\sum_{j\neq i} \vec{u}_j \vec{\gamma}^j), \tag{9}$$

where $\mathbf{I} \in \mathbb{R}^{N\times N}$ is an identity matrix. By using Eq. (9) and writing $\mathcal{J}_{dic}$ in terms of $\vec{u}_i$, we reformulate Eq. (4) as

$$\begin{aligned} \min_{\vec{u}_i} \quad & \vec{u}_i^\top (\vec{\gamma}^i \vec{\gamma}^{i\top} \hat{\mathcal{K}}) \vec{u}_i + [\vec{\gamma}^i (-2\mathbf{E}_i^\top \hat{\mathcal{K}} + \lambda \tilde{\mathcal{K}}) + \tau \vec{1}^\top \mathbf{L}] \vec{u}_i \\ \text{s.t.} \quad & \|\vec{u}_i\|_0 < T_0, \quad \|\hat{\Phi}(\mathbf{X})\vec{u}_i\|_2^2 = 1, \quad u_{ji} \in \mathbb{R}_+. \end{aligned} \tag{10}$$

Analogous to Eq. (8), this is a convex non-negative quadratic problem in terms of $\vec{u}_i$ with a hard limit on $\|\vec{u}_i\|_0$. Hence, we

[2]This is a part of an under review article, but the description of the NQP algorithm is included in the supplementary materials.

update the prototype vectors $\{\vec{u}_i\}_{i=1}^{c}$ by solving Eq. (10) using the NQP algorithm. After updating each $\vec{u}_i$, we normalize it as $\vec{u}_i = \frac{\vec{u}_i}{\|\Phi(\mathbf{X})\vec{u}_i\|_2} = \frac{\vec{u}_i}{\sqrt{\vec{u}_i^\top \hat{\mathcal{K}}\vec{u}_i}}$.

4.3 Updating Kernel Weights $\vec{\beta}$

By normalizing each base kernel $\mathcal{K}_i(\mathbf{X},\mathbf{X})$ in advance, we can simplify Eq. (4) to the following linear program (LP)

$$\begin{aligned} \min_{\vec{\beta}} \quad & (\vec{\mathcal{E}}_{rec} + \lambda\vec{\mathcal{E}}_{dis} + \mu\vec{\mathcal{E}}_{mk})^\top \vec{\beta} \\ \text{s.t.} \quad & \textstyle\sum_{i=1}^{f} \beta_i = 1,\ \beta_i \in \mathbb{R}^+, \end{aligned} \tag{11}$$

where the entries of $\vec{\mathcal{E}}_{rec}$, $\vec{\mathcal{E}}_{dis}$, and $\vec{\mathcal{E}}_{mk}$ are derived by incorporating Eq. (2) into the terms $\mathcal{J}_{rec}$, $\mathcal{J}_{dis}$, and $\mathcal{J}_{mk}$ respectively. We compute their l-th entries ($l = 1,\dots,f$) as

$$\begin{aligned} &\mathcal{E}_{rec}(l) = \mathrm{Tr}[\mathcal{K}_l(\mathbf{I} - 2\mathbf{U}\mathbf{\Gamma}) + \mathbf{\Gamma}^\top\mathbf{U}^\top\mathcal{K}_l\mathbf{U}\mathbf{\Gamma}], \\ &\mathcal{E}_{dis}(i) = \mathrm{Tr}(\tilde{\mathcal{K}}_l\mathbf{U}\mathbf{\Gamma}), \\ &\mathcal{E}_{mk}(l) = \sum_{i=1}^{N}\sum_{s\in\mathcal{N}_i^k}[2 - 2\mathcal{K}_l(\vec{x}_i,\vec{x}_s)] + \sum_{s\in\mathcal{N}_i^k}\mathcal{K}_l(\vec{x}_i,\vec{x}_s), \end{aligned} \tag{12}$$

where $\tilde{\mathcal{K}}_l$ is derived by computing $\tilde{\mathcal{K}}$ while replacing $\mathcal{K}$ with $\mathcal{K}_l$. Therefore, we can efficiently solve the LP in Eq. (11) using linear solvers [Strayer, 2012]. Algorithm 1 provides the overview of all the optimization steps for our IMKPL framework.

Algorithm 1 Interpretable Multiple-Kernel Prototype Learning

```
Parameters: Trade-off weights $(\lambda,\mu,\tau)$, sparsity limit $T_0$, neighborhood size $k$
Input: Label matrix $\mathbf{L}$, kernel functions $\{\mathcal{K}_i(\mathbf{X},\mathbf{X})\}_{i=1}^{f}$
Output: Prototype vectors $\{\vec{u}_i\}_{i=1}^{c}$ and kernel weights $\vec{\beta}$
Initialization: Computing $\{\tilde{\mathcal{K}}, \{\tilde{\mathcal{K}}_i\}_{i=1}^{f}, \vec{\mathcal{E}}_{mk}\}$, $\vec{\beta} = \vec{\mathbf{1}}$
1: repeat
2:     Computing $\hat{\mathcal{K}}(\mathbf{X},\mathbf{X}) = \sum_{i=1}^{f}\beta_i\mathcal{K}_i(\mathbf{X},\mathbf{X})$
3:     Updating $\mathbf{\Gamma}$ based on Eq. (8) using NQP
4:     Updating $\mathbf{U}$ based on Eq. (10) using NQP
5:     Updating $\vec{\beta}$ based on Eq. (11) using an LP solver
6: until convergence
7: return $\mathbf{U}, \mathbf{\Gamma}, \vec{\beta}$
```

4.4 Representation of the Test Data

To represent each test data $\vec{x}_{test}$ using the trained $\mathbf{U}$ and $\vec{\beta}$, the sparse code $\vec{\gamma}_{test}$ is computed using Eq. (8) while setting $\lambda = 0$. The relational values of the entries in $\vec{\gamma}_{test}$ show the main prototypes which are used to represent $\vec{x}_{test}$.

4.5 Complexity and Convergence of IMKPL

In order to calculate the computational complexity of IMKPL per iteration, we analyze the update of each $\{\mathbf{\Gamma},\mathbf{U},\vec{\beta}\}$ individually. In each iteration, the update of $\mathbf{\Gamma}$ and $\mathbf{U}$ are done using the NQP algorithm which has the time complexity of $\mathcal{O}(nT_0^2)$ (provided in the supplementary material), where n is the number of dimensions in the quadratic problem. Therefore, optimizing $\mathbf{\Gamma}$ and $\mathbf{U}$ cost $\mathcal{O}(N(cT_0^2 + cN))$ and $\mathcal{O}(c(NT_0^2 + N^2 + pN)$ respectively. Optimizing $\vec{\beta}$ with a LP solver has the computational complexity of $\mathcal{O}(2tf + f(N^2 + kN))$, where t is the number of iterations in which the LP-solver converges. Also, computing $\hat{\mathcal{K}}(\mathbf{X},\mathbf{X})$ in each iteration costs $\mathcal{O}(cN^2)$. As a common observation in practice, each $p, f, k << N$, and we choose $c = pT_0$ and usually $T_0 < N/p$ in the implementations. Hence, the computational complexity of IMKPL in each iteration is approximately $\mathcal{O}(T_0N^2)$.

Each of the sub-problems defined by Eqs. (8),(10),(11) are convex. Hence, the optimization problem of Eq. (4) is convex w.r.t each of $\{\mathbf{\Gamma},\mathbf{U},\vec{\beta}\}$ when other two parameters are fixed. As a result, the alternating optimization scheme of Algorithm 1 converges in a limited number of steps.

Dataset	T_0	λ	μ	τ	Size	Dim	Cls
CLL_SUB	15	0.2	0.1	0.2	111	11340	3
TOX_171	18	0.1	0.3	0.1	171	5748	4
Isolet	20	0.2	0.2	0.3	1560	617	26
UTKinect	5	0.1	0.1	0.2	200	60	10
PEMS	20	0.2	0.3	0.1	440	963	7
AUSLAN	12	0.3	0.1	0.2	2500	128	95

Table 1: Information and implementation settings regarding the selected datasets. **Dim**: dimensions, **Cls**: number of classes.

5 Experiments

For our experiments, we implement IMKPL on 6 datasets with different structures. We choose the vectorial datasets CLL_SUB_111, TOX_171, and Isolet from a feature selection repository [3]; and, PEMS and AUSLAN from the UCI machine learning repository [4] along with Utkiect Human action dataset [Xia *et al.*, 2012] form our multivariate time-series (MTS) datasets. The detailed characteristics of these datasets are reported in Table 1. Besides, the code of the IMKPL algorithm is publicly available online [5].

5.1 Experimental Setup

To compute the base kernels for vectorial datasets, we use the Gaussian kernel $\mathcal{K}_i(\vec{x}_s,\vec{x}_t) = exp(-\mathcal{D}(\vec{x}_s^i,\vec{x}_t^i)^2/\delta_i)$ where $\vec{x}^i$ denotes the i-th dimension of $\vec{x}$, and $\mathcal{D}(\vec{x}_s^i,\vec{x}_t^i) = \|\vec{x}_s^i - \vec{x}_t^i\|_2$ as the Euclidean distance between each pair of $\{\vec{x}_s,\vec{x}_t\}$ in that dimension. The scalar δ_i is equal to the average of $\mathcal{D}(\vec{x}_s^i,\vec{x}_t^i)^2$ over all data points. For MTS datasets, we compute each $\mathcal{K}_i$ via employing the global alignment kernel (GAK) [Cuturi *et al.*, 2007] for the i-th dimension of the time-series. Exceptionally for Utkiect, prior to the application of GAK, we use the pre-processing from [Vemulapalli *et al.*, 2014] to obtain the Lie Group representation.

We compare our proposed method to the following major prototype-based learning or multiple-kernel dictionary learning methods: AKGLVQ [Schleif *et al.*, 2011], PS [Bien *et al.*, 2011], MKLDPL [Zhu *et al.*, 2017], DKMLD [Thiagarajan *et al.*, 2014], and MIDL [Shrivastava *et al.*, 2015]. The

[3]http://featureselection.asu.edu/datasets.php

[4]https://archive.ics.uci.edu/ml/index.php

[5]https://github.com/XXX

AKGLVQ algorithm is the sparse variant of the kernelized-generalized LVQ [Schleif *et al.*, 2011], and for the PS algorithm, we use its distance-based implementation. These two algorithms are implemented by using the average-kernel as the input ($\vec{\beta} = \vec{1}$). Hence, we also implement ISKPL as the single-kernel variant of IMKPL on that input representation.
Note: We elusively selected only the baseline methods which can be evaluated according to our specific research objectives (**O1-O3**).

We perform 5-fold cross-validation on the training set to tune the hyper-parameters $\{\lambda, \mu, T_0, \tau\}$ in Eq. (4) which are reported in Table 1. We carry out a similar procedure regarding the parameter tuning of other baselines. For IMKPL, we determine the dictionary size $c = pT_0$ and the neighborhood radius $k = T_0$. As the rationale, the constraint $\|\vec{u}_i\|_0 \leq T_0$ and the term $\mathcal{J}_{dis}$ in Eq. (4) make each $\vec{u}_i$ effective mostly on its T_0-radius neighborhood. In practice, choosing $\lambda = \mu = \tau \in [0.2\ 0.4]$ is a good working setting for IMKPL to initiate the parameter tuning (e.g., Figure 3).

5.2 Evaluation Measures

In order to evaluate the quality of the learned prototypes in the resulted RKHS (based on $\{\mathbf{U}, \vec{\beta}\}$), we utilize the following measures which coincide with our objectives **O1-O3**.

Interpretability of the Prototypes (IP)
As discussed in Section 3, we have two main preferences regarding the interpretability of each prototype $\hat{\Phi}(\mathbf{X})\vec{u}_i$: 1. Its formation based on class-homogeneous data samples. 2. Its connection to local neighborhoods in the feature space. Therefore we use the following IP term to evaluate the above criteria base on the values of the prototype vectors $\{\vec{u}_i\}_{i=1}^c$:

$$IP = \frac{1}{c}\sum_{i=1}^{c} \frac{\vec{l}^q \vec{u}_i}{\|\mathbf{L}\vec{u}_i\|_1} e^{-\sum\limits_{s,t} u_{si}u_{ti}\|\hat{\Phi}(\vec{x}_s)-\hat{\Phi}(\vec{x}_t)\|_2^2}, \tag{13}$$

where $q = \arg\max\limits_{q} \vec{l}^q \vec{u}_i$ is the class to which the i-th prototype is assigned. The first part of this equation obtains the maximum value of 1 if each $\vec{u}_i$ has its non-zero entries related to only one class of data, while the exponential term becomes 1 (maximum) if those entries correspond to a condense neighborhood of points in RKHS. Hence, IP becomes close to 1 if both of the above concerns are sufficiently fulfilled. For the PS algorithm, we measure IP based on the samples inside ϵ-radius of each prototype [Bien *et al.*, 2011].

Discriminative Representation (DR)
In order to properly evaluate how discriminative each prototype $\hat{\Phi}(\mathbf{X})\vec{u}_i$ is we define the DR term:

$$DR = \frac{1}{c}\sum_{i=1}^{c} \frac{\sum_{s:\vec{l}_s=q} \gamma_{is}}{\|\vec{\gamma}^i\|_1}, \tag{14}$$

where q is the same as in Eq. (13), and $\mathbf{\Gamma}$ is computed based on the test set. Hence, DR becomes 1 (maximum) if each prototype i which is assigned to class q only represents (reconstructs) that class of data; in other words, the prototypes provide exclusive representation of the classes. The DR measure does not fit the models of AKGLVQ and PS algorithms.

Methods	IP	DR	IP	DR	IP	DR
	CLL_SUB		TOX_171		Isolet	
IMKPL (ours)	**91**	**75**	**95**	**89**	**96**	**90**
ISKPL (ours)	88	70	93	79	94	81
MKLDPL	75	57	86	66	84	63
DKMLD	67	51	71	60	72	57
MIDL	66	50	69	60	69	54
AKGLVQ	69	–	76	–	78	–
PS	78	–	80	–	84	–
	UTKinect		PEMS		AUSLAN	
IMKPL (ours)	**96**	**91**	**95**	**88**	**97**	**87**
ISKPL (ours)	92	82	91	80	95	79
MKLDPL	83	60	77	59	86	70
DKMLD	74	52	70	56	73	59
MIDL	69	50	67	58	71	61
AKGLVQ	77	–	72	–	75	–
PS	79	–	76	–	81	–

Table 2: Comparison of the baselines regarding IP and DR.

Classification Accuracy of Test Data (Acc)
For each test data $\vec{x}_{test}$, we predict its class q as

$$q = \arg\max_{q}\ \vec{l}^q \mathbf{U}\vec{\gamma}_{test}, \tag{15}$$

meaning that the q-th class provides the most contributions in the reconstruction of $\vec{x}_{test}$. Accordingly, we compute the average of $Acc = \frac{\#\text{correct predictions}}{N} \times 100$ over 10 randomized test / train selections for each dataset.

5.3 Results: Efficiency of the Prototypes

In Table 2, we compare the baselines regarding the interpretability and discriminative qualities of their trained prototypes. Considering the IP values, IMKPL significantly outperforms both the MKDL and prototype-based learning algorithms. For the PEMS dataset, our method has a margin of 18% compared to the best baseline algorithm (MKLDPL). Also, the ISKPL algorithm obtains higher interpretability performances than the single-kernel and multiple-kernel baselines, which shows the effectiveness of the prototype leaning parts of the design ($\mathcal{J}_{dis}$ and $\mathcal{J}_{ip}$). Besides, the difference between the IP values of ISKPL and IMKPL signifies the role of the $\mathcal{J}_{ls}$ objective in enhancing the interpretation of IMKPL's prototypes by learning a suitable multiple-kernel representation. Other algorithms show weak results in learning class-specific and locally concentrated prototypes.

We observe similar behaviors by comparing the algorithms based on the DR measure. Table 2 shows that the prototypes learned by IMKPL are more efficient regarding the exclusive representation of the classes in a combined RKHS. For instance, IMKPL outperforms MKLDPL (best baseline) with the DR margin of 31$\%$ on UTKinect dataset.

5.4 Results: Discriminative Feature Selection

Each base kernel $\mathcal{K}_i$ is derived from one dimension of the data. Therefore, we evaluate the discriminative feature selection performance of the algorithms by comparing their $\|\vec{\beta}\|_0$ and Acc values. As presented in Table 3, IMKPL has the best prediction accuracies on all datasets. It outperforms other baselines with relatively significant Acc-margins (e.g., 4.21% compared to MKLDPL on `PEMS`).

Regarding $\|\vec{\beta}\|_0$, IMKPL obtains the smallest set of selected features on three of the datasets. It particularly shows a significant feature selection performance on `TOX_171` by obtaining 97.21% accuracy while selecting 72 features out of the total 5748 dimensions. Regarding other datasets (`CLL_SUB`, `Isolet`, and `PEMS`), considering also the values of Acc reveals that the multiple-kernel optimization of IMKPL (role of $\vec{\beta}$ in Eq. (4)) finds a set of features (not necessarily the smallest set) which leads to an efficient discriminative prototype-based model. This role is more noticeable when comparing IMKPL to ISKPL.

On the other hand, comparing the prediction accuracy of ISKPL to AKGLVQ and PS (as the major prototype-based learning methods) demonstrates the significant discriminative performance of our algorithm in this domain. Besides, even though ISKPL obtained lower Acc values than MKLDPL and DKMLD (as it does not optimize $\vec{\beta}$), its higher DR values show the effectiveness of its design ($\mathcal{J}_{dis}$ and $\mathcal{J}_{ip}$) regarding our expectations from a prototype-based representation.

5.5 Effect of Parameter Settings

We study the effect of the hyper-parameters in the Acc and Ip performance of the IMKPL algorithm. We conduct 4 individual experiments on the `Isolet` dataset (Figure 3), at each of which we change one parameter from $\{\lambda, \mu, \tau, T_0\}$ while having others fixed to the values given in Table 1.

As a general observation in Figure 3-(left), the algorithm maintains an acceptable performance when $\{\lambda, \mu, \tau\}$ are chosen in the range of $[0.1\ 0.5]$, but its accuracy and interpretability may decrease outside of this range. To be more specific, τ has a slight effect on Acc, but can increase the value of IP almost monotonically. In comparison, μ and λ influence Acc more significantly. Nevertheless, they have small effects on IP when they are small (in $[0\ 0.6]$), but λ has a constructive effect and μ a slight destructive role when their values become larger. Regarding the effect of μ on IP and Acc, Figure 3 shows that when the classes have high overlapping in the RKHS, focusing only on $\mathcal{J}_{mk}$ (choosing a large μ) does not necessarily provide the best prototype-based solution.

Figure 3-(right) shows that increasing T_0 generally improves Acc up to a point after which the value of Acc has slight changes. Since T_0 also controls the total number of learned prototypes, its large values only increases the redundancy of the learned prototypes. Besides, increase of T_0 generally degrades the IP value by expanding the neighborhoods for constructing each $\vec{u}_i$ in Eq. 4. However, IP almost settles down to a lower bound value for large T_0 ($\approx 87\%$ for `Isolet`) because of the minimum interpretability affected by the non-negativity constraint $u_{ji} \in \mathbb{R}_+$ in Eq. (4).

Methods	Acc	$\|\vec{\beta}\|_0$	Acc	$\|\vec{\beta}\|_0$	Acc	$\|\vec{\beta}\|_0$
	`CLL_SUB`		`TOX_171`		`Isolet`	
IMKPL(ours)	**81.73**	204	**97.21**	**72**	**97.75**	141
ISKPL(ours)	77.95	–	88.07	–	90.11	–
MKLDPL	79.39	310	94,72	347	95,21	224
DKMLD	78.13	**101**	90,49	230	92,74	**126**
MIDL	77,24	452	87,63	571	91,66	236
AKGLVQ	74.66	–	86.21	–	88.32	–
PS	74.03	–	82.47	–	86.36	–
	`UTKinect`		`PEMS`		`AUSLAN`	
IMKPL(ours)	**98.82**	**14**	**95.32**	89	**95.81**	**21**
ISKPL(ours)	90.32	–	88.65	–	88.92	–
MKLDPL	94.32	32	91,11	65	92.39	40
DKMLD	91.64	30	90,23	**62**	89.21	36
MIDL	91,01	47	87,51	158	88.45	79
AKGLVQ	88.75	–	85.13	–	85.35	–
PS	85.89	–	84.54	–	83.65	–

The best result (**bold**) is according to a two-valued t-test at a 5% significance level.

Table 3: Comparison of the baselines regarding Acc (%) and $\|\vec{\beta}\|_0$.

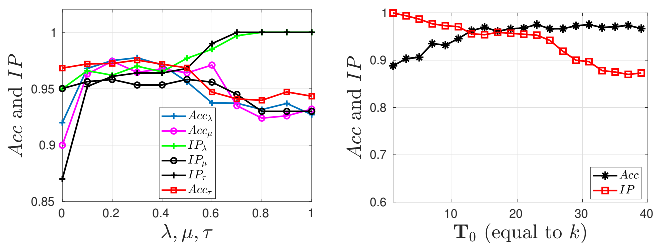

Figure 3: Isolated effects of changing the parameters $\{\lambda, \mu, T_0, \tau\}$ on the performance measures Acc and IP for the `Isolet` dataset.

6 Conclusion

We have proposed an optimization framework as an efficient prototype-based learning to obtain discriminative representation of a dataset in the feature space. We explicitly formulated our expectations from an interpretable prototype-based representation into specific objectives upon which we designed our IMKPL algorithm. Our method learns prototypes which represent the local subsets within the classes with small overlapping between different classes. Besides, each prototype is specifically constructed to be interpreted by the condensed class-homogeneous neighborhood of data points in RKHS to which it belongs (e.g., a subset of similar *walking* sequences in the data).

Additionally, our method can find an efficient combined RKHS in the feature space to better fulfill the above prototype-based concerns there. Hence, IMKPL also performs discriminative feature selection for multiple-kernel feature representations. Our optimization framework is constructed by adding novel objective terms to the basic scheme of multiple-kernel dictionary learning and we solve it using convex alternating optimization.

We implemented our algorithm on relatively large benchmarks (w.r.t. the number of dimensions or data samples) in the vectorial and time-series domains. The empirical results validate the efficiency of IMKPL compared to other prototype-based baselines.

References

[Aharon *et al.*, 2006] M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, 2006.

[Bach *et al.*, 2004] Francis R Bach, Gert RG Lanckriet, and Michael I Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *ICML'04*, 2004.

[Biehl *et al.*, 2007] Michael Biehl, Anarta Ghosh, and Barbara Hammer. Dynamics and generalization ability of lvq algorithms. *JMLR*, 8(Feb):323–360, 2007.

[Bien *et al.*, 2011] Jacob Bien, Robert Tibshirani, et al. Prototype selection for interpretable classification. *The Annals of Applied Statistics*, 5(4):2403–2424, 2011.

[Cuturi *et al.*, 2007] Marco Cuturi, Jean-Philippe Vert, Oystein Birkenes, and Tomoko Matsui. A kernel for time series based on global alignments. In *ICASSP 2007*, volume 2, pages II–413. IEEE, 2007.

[Dileep and Sekhar, 2009] Aroor Dinesh Dileep and C Chandra Sekhar. Representation and feature selection using multiple kernel learning. In *IJCNN 2009*, pages 717–722. IEEE, 2009.

[Friedman *et al.*, 2001] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[Gönen and Alpaydın, 2011] Mehmet Gönen and Ethem Alpaydın. Multiple kernel learning algorithms. *Journal of machine learning research*, 12(Jul):2211–2268, 2011.

[Guo *et al.*, 2016] Jun Guo, Yanqing Guo, Xiangwei Kong, Man Zhang, and Ran He. Discriminative analysis dictionary learning. In *AAAI*, pages 1617–1623, 2016.

[Hammer *et al.*, 2014] Barbara Hammer, Daniela Hofmann, Frank-Michael Schleif, and Xibin Zhu. Learning vector quantization for (dis-) similarities. *Neurocomputing*, 131:43–51, 2014.

[Hosseini and Hammer, 2018] B. Hosseini and B. Hammer. Confident kernel sparse coding and dictionary learning. In *ICDM'18*, 2018.

[Johnson and Robinson, 1981] Charles R Johnson and Herbert A Robinson. Eigenvalue inequalities for principal submatrices. *Linear Algebra and its Applications*, 37:11–22, 1981.

[Kim *et al.*, 2010] Taehwan Kim, Gregory Shakhnarovich, and Raquel Urtasun. Sparse coding for learning interpretable spatio-temporal primitives. In *NIPS'10*, pages 1117–1125, 2010.

[Kohonen, 1995] T. Kohonen. *Self-organizing maps*. Springer, 1995.

[Lee *et al.*, 2006] H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. In *Advances in neural information processing systems*, pages 801–808, 2006.

[Nesterov, 2012] Yu Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

[Nguyen *et al.*, 2013] H. V. Nguyen, V. M. Patel, N. M. Nasrabadi, and R. Chellappa. Design of non-linear kernel dictionaries for object recognition. *IEEE Transactions on Image Processing*, 22(12):5123–5135, 2013.

[Nova and Estévez, 2014] David Nova and Pablo A Estévez. A review of learning vector quantization classifiers. *Neural Computing and Applications*, 25(3-4):511–524, 2014.

[Rosch, 1975] Eleanor Rosch. Cognitive reference points. *Cognitive psychology*, 7(4):532–547, 1975.

[Rubinstein *et al.*, 2008] Ron Rubinstein, Michael Zibulevsky, and Michael Elad. Efficient implementation of the k-svd algorithm using batch orthogonal matching pursuit. *Cs Technion*, 40(8):1–15, 2008.

[Schleif *et al.*, 2011] F-M Schleif, Thomas Villmann, Barbara Hammer, and Petra Schneider. Efficient kernelized prototype based classification. *International Journal of Neural Systems*, 21(06):443–457, 2011.

[Shawe-Taylor and Cristianini, 2004] John Shawe-Taylor and Nello Cristianini. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[Shrivastava *et al.*, 2015] Ashish Shrivastava, Jaishanker K Pillai, and Vishal M Patel. Multiple kernel-based dictionary learning for weakly supervised classification. *Pattern Recognition*, 48(8):2667–2675, 2015.

[Strayer, 2012] James K Strayer. *Linear programming and its applications*. Springer Science & Business Media, 2012.

[Thiagarajan *et al.*, 2014] J. J. Thiagarajan, K. N. Ramamurthy, and A. Spanias. Multiple kernel sparse representations for supervised and unsupervised learning. *IEEE TIP*, 23(7):2905–2915, 2014.

[Van Loan, 1996] Charles F Van Loan. Matrix computations (johns hopkins studies in mathematical sciences), 1996.

[Vemulapalli *et al.*, 2014] Raviteja Vemulapalli, Felipe Arrate, and Rama Chellappa. Human action recognition by representing 3d skeletons as points in a lie group. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 588–595, 2014.

[Von. Luxburg, 2007] Ulrike Von. Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[Wang *et al.*, 2014] J. Wang, J. Yang, H. Bensmail, and X. Gao. Feature selection and multi-kernel learning for sparse representation on a manifold. *Neural Networks*, 51:9–16, 2014.

[Xia *et al.*, 2012] Lu Xia, Chia-Chih Chen, and JK Aggarwal. View invariant human action recognition using histograms of 3d joints. In *CVPRW'12 Workshops*, pages 20–27. IEEE, 2012.

[Zhu *et al.*, 2017] X. Zhu, X.-Y. Jing, F. Wu, D. Wu, L. Cheng, S. Li, and R. Hu. Multi-kernel low-rank dictionary pair learning for multiple features based image classification. In *AAAI*, pages 2970–2976, 2017.