Reduced Kernel Dictionary Learning

Denis C. ILIE-ABLACHIM
Faculty of Automatic Control and Computers
University Politehnica of Bucharest
denis.ilie_ablachim@upb.ro

Bogdan DUMITRESCU
Faculty of Automatic Control and Computers
University Politehnica of Bucharest
bogdan.dumitrescu@upb.ro

***Abstract*—In this paper we present new algorithms for training reduced-size nonlinear representations in the Kernel Dictionary Learning (KDL) problem. Standard KDL has the drawback of a large size of the kernel matrix when the data set is large. There are several ways of reducing the kernel size, notably Nyström sampling. We propose here a method more in the spirit of dictionary learning, where the kernel vectors are obtained with a trained sparse representation of the input signals. Moreover, we optimize directly the kernel vectors in the KDL process, using gradient descent steps. We show with three data sets that our algorithms are able to provide better representations, despite using a small number of kernel vectors, and also decrease the execution time with respect to KDL.**

I. Introduction

Dictionary Learning (DL) is a representation learning method that aims to find a sparse representation for a set of signals, $\boldsymbol{Y}$, represented as a matrix with N columns (signals) of size m. The representation is achieved by computing a dictionary $\boldsymbol{D}$ of size $m \times n$ and a sparse representation $\boldsymbol{X}$ of size $n \times N$ such that a good approximation $\boldsymbol{Y} \approx \boldsymbol{DX}$ is obtained. Most applications with dictionary learning are in image denoising, inpainting, signal reconstruction, clustering or classification.

In this paper we present new methods of dictionary learning that produce sparse representations in both linear and nonlinear spaces, starting from the Kernel Dictionary Learning (KDL) idea [1], [2]. In a first approach, this is done in two stages; the linear representation is built and then it is used unchanged in the nonlinear space as kernel vectors during the training procedure. In the second approach, the kernel vectors are optimized alongside with the nonlinear representation. The main advantage of this method is the use of a reduced matrix, $\boldsymbol{D}$, containing the kernel vectors, which is also the dictionary in a standard DL problem. By the use of the reduced matrix, the built-in kernel is smaller and thus the problem complexity is reduced.

As KDL has a high complexity when N is large (and so the kernel matrix is large), solutions have been adopted from other problems where kernels appear. In Large-Scale Kernel Machines, various kernel enhancement or resizing strategies have been used, such as Nyström Sampling [3], [4] or Random Fourier Features (RFF) [5]. The first one computes a rank $\hat{m}$ approximation $\hat{\boldsymbol{K}}$ of the kernel matrix $\boldsymbol{K}$. The whole procedure consists in approximating the nonlinear mapping function $\varphi(\boldsymbol{Y})$ with a matrix $\hat{\boldsymbol{Y}}$, containing a compressed version of the original signals. Compared to the original problem, the new problem no longer requires a high computational cost. The Random Fourier Features method proposes to map the input signals to a randomized low-dimensional feature space. More exactly, the inner product, used in the kernel trick, is replaced with a randomized map $k(\boldsymbol{x}, \boldsymbol{y}) = \varphi(\boldsymbol{x})^\top \varphi(\boldsymbol{y}) = z(\boldsymbol{x})^\top z(\boldsymbol{y})$, where $z : \mathbb{R}^m \mapsto \mathbb{R}^{\hat{m}}$ and $\hat{m} \ll m$. Both methods enable the use of fast linear methods, which will further use the resulting features. The use of reduced kernels dates from Support Vector Machine times, significant contributions being [6], [7], with applications in classification [8]. Other learning methods where a reduced kernel appears can be found in [9]. In these early works, kernel vectors are usually selected (randomly or with some heuristic) as a subset of input signals. Finally, there are KDL substitutes like [10], where the nonlinear transformation is performed by a neural encoder-decoder, with standard DL on the encoded signals.

The paper is organized as follows. In Section II we review the standard DL and KDL problems and the most common algorithm for solving them. In Section III we present our own contribution, named the Reduced Kernel Dictionary Learning (RKDL) problem, under three different scenarios, in which the kernel vectors are: a) the result of a DL problem, solved a priori; b) optimized together with the other KDL variables, using gradient descent, with an objective that is directly related to that of the KDL problem; c) optimized like before, but using a mixed objective that combines the nonlinear representation of the signals with the linear representation of the kernel vectors. The algorithms for b) and c) are new. Section IV contains the experimental results, obtained on three public data sets: Digits, MNIST [11] and CIFAR-10 [12], under the three proposed scenarios.

II. Kernel Dictionary Learning

The DL problem is formulated as following

$$\begin{array}{ll} \min\limits_{\boldsymbol{D},\boldsymbol{X}} & \|\boldsymbol{Y} - \boldsymbol{DX}\|_F^2 \\ \text{s.t.} & \|\boldsymbol{x}_\ell\|_0 \le s_x, \ell = 1 : N \\ & \|\boldsymbol{d}_j\| = 1, j = 1 : n, \end{array} \tag{1}$$

where $\|\cdot\|_0$ represents the 0-pseudo-norm, s_x is the sparsity level and $\boldsymbol{d}_j$ is a column (named also atom) of $\boldsymbol{D}$.

This work was supported by a grant of the Romanian Ministry of Research, Innovation and Digitization, CCCDI - UEFISCDI, project number PN-III-P2-2.1-PED-2019-3248, within PNCDI III.

The standard dictionary learning problem can be solved by using simple strategies. In order to overcome the nonconvexity and the huge dimension of the problem, the most usual optimization procedure iterates two basic steps and is also known as DL by Alternate Optimization. In this way, the problem is divided in two subproblems: sparse coding and dictionary update. By alternating these two stages for a given number of iterations, good local solutions can be obtained. A simple iteration consists of computing the sparse representation $\boldsymbol{X}$, using Orthogonal Matching Pursuit (OMP) [13] while the dictionary $\boldsymbol{D}$ is fixed, and then updating the dictionary atoms successively while the sparse representation is fixed. There are several methods of dictionary learning [14], but the one of interest to us is AK-SVD [15], [16] due to its low complexity and good performance.

In the DL problem, the input data are modeled through a linear representation, which in some cases may be seen as a limitation. In order to overcome this drawback, kernel representations can be used for a better quantification of similarities or differences between input vectors.

The kernel representation is an extension to nonlinearity. We do this by associating to a data vector $\boldsymbol{y} \in \mathbb{R}^m$ a feature vector $\varphi(\boldsymbol{y}) \in \mathbb{R}^{\tilde{m}}$, where $\varphi : \mathbb{R}^m \to \mathbb{R}^{\tilde{m}}$ is a nonlinear function. Typically, Mercer kernels are used, which can be expressed as a scalar product of feature vector functions $k(\boldsymbol{x}, \boldsymbol{y}) = \varphi(\boldsymbol{x})^\top \varphi(\boldsymbol{y})$. In the last form, the scalar product can be replaced with a specific function definition, such as radial basis function (RBF) $k(\boldsymbol{x}, \boldsymbol{y}) = \exp(-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|_2^2}{2\sigma^2})$ or polynomial kernel $k(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}^\top \boldsymbol{y} + \alpha)^\beta$.

The Kernel Dictionary Learning (KDL) [1], [2] problem is

$$\begin{array}{ll} \min\limits_{\boldsymbol{A},\boldsymbol{Z}} & \|\varphi(\boldsymbol{Y}) - \varphi(\boldsymbol{Y})\boldsymbol{A}\boldsymbol{Z}\|_F^2 \\ \text{s.t.} & \|\boldsymbol{z}_\ell\|_0 \le s_z, \ell = 1 : N \\ & \|\varphi(\boldsymbol{Y})\boldsymbol{a}_j\| = 1, j = 1 : n, \end{array} \tag{2}$$

where $\varphi(\boldsymbol{Y})$ represents the nonlinear extension of data and $\varphi(\boldsymbol{Y})\boldsymbol{A}$ is the nonlinear dictionary, where $\boldsymbol{A}$ is the coefficients matrix of the dictionary. Depending on the used data set, the problem can be difficult to solve due the large kernel matrix $\varphi(\boldsymbol{Y})^\top \varphi(\boldsymbol{Y})$ that results from the trace form of the objective function. In this case, the problems with large data sets can involve large volume of memory and long execution times. The KDL problem can also be solved by alternate optimization. The sparse representation is computed according to the Kernel OMP algorithm [1]. The columns of the matrix $\boldsymbol{A}$ are sequentially update with an algorithm inspired by AK-SVD, while the representation matrix $\boldsymbol{X}$ is fixed.

Both algorithms, AK-SVD and Kernel AK-SVD, alongside with the sparse representation computation are presented in [14].

III. Reduced Kernel Dictionary Learning

Nonlinear space can extend the horizon of data representation. However, KDL has disadvantages when the number of available signals is large. The size of the kernel increases in proportion to the size of the data. Thus, the problem becomes more complex from a numerical point of view.

In order to overcome this limitations we use a reduced space on which the kernel matrix is built. This strategy is implemented by using as kernel vectors not the full set of signals, $\boldsymbol{Y}$, but a smaller set of vectors, $\boldsymbol{D}$, trained with DL as a dictionary for linear representations, thus replacing (2) with

$$\begin{array}{ll} \min\limits_{\boldsymbol{A},\boldsymbol{Z}} & \|\varphi(\boldsymbol{Y}) - \varphi(\boldsymbol{D})\boldsymbol{A}\boldsymbol{Z}\|_F^2 \\ \text{s.t.} & \|\boldsymbol{z}_\ell\|_0 \le s_z, \ell = 1 : N \\ & \|\varphi(\boldsymbol{D})\boldsymbol{a}_j\| = 1, j = 1 : n. \end{array} \tag{3}$$

This problem has two advantages that can be used as needed. If the number of training signals is very large, a dictionary with a much smaller number of atoms can be used. On the other hand, we have problems where the number of training signals is small or the given signals are not representative enough for the representation problem. In this case it is recommended to use large dictionaries. In our work the case of interest is the first one.

A. Standard Reduced Kernel Dictionary Learning

A first approach to (3) consists of solving it in two steps. In the first step we design $\boldsymbol{D}$ by solving a DL problem and thus obtaining a trained dictionary. We call this method RKDL-D, due the use of matrix $\boldsymbol{D}$; it was introduced in [17]. Here is a brief reminder of the atom update step of RKDL-D. (Sparse representation can be easily derived from Kernel OMP.)

Expressing the objective of (3) in its trace form and isolating the current atom a_j, we can write

$$\operatorname{Tr}\left[\left(\varphi^\top(\boldsymbol{Y}) - \sum_{i\neq j} \boldsymbol{z}_i \boldsymbol{a}_i^\top \varphi^\top(\boldsymbol{D}) - \boldsymbol{z}_j \boldsymbol{a}_j^\top \varphi^\top(\boldsymbol{D})\right) \left(\varphi(\boldsymbol{Y}) - \varphi(\boldsymbol{D}) \sum_{i\neq j} \boldsymbol{a}_i \boldsymbol{z}_i^\top - \varphi(\boldsymbol{D})\boldsymbol{a}_j \boldsymbol{z}_j^\top\right)\right].$$

We compute the partial derivatives with respect to the current atom $\boldsymbol{a}_j$

$$\frac{\partial \operatorname{Tr}(\cdot)}{\partial \boldsymbol{a}_j} = 2\|\boldsymbol{z}_j\|^2 \boldsymbol{K}_{DD}\boldsymbol{a}_j + 2\boldsymbol{K}_{DD}\boldsymbol{R}\boldsymbol{z}_j - 2\boldsymbol{K}_{YD}\boldsymbol{z}_j$$

and the current sparse representation vector $\boldsymbol{z_j}$

$$\frac{\partial \operatorname{Tr}(\cdot)}{\partial \boldsymbol{z}_j} = 2\boldsymbol{a}_j^\top \boldsymbol{K}_{DD}\boldsymbol{a}_j \boldsymbol{z}_j + 2\boldsymbol{R}^\top \boldsymbol{K}_{DD}\boldsymbol{a}_j - 2\boldsymbol{K}_{YD}\boldsymbol{a}_j.$$

where we have used the notations $\boldsymbol{K}_{YD} = K(\boldsymbol{Y}, \boldsymbol{D})$, $\boldsymbol{K}_{DD} = K(\boldsymbol{D}, \boldsymbol{D})$ and $\boldsymbol{R} = \sum\limits_{i\neq j} \boldsymbol{a}_i \boldsymbol{z}_i^\top$.

Setting these derivatives to zero, we obtain the optimal atom (for fixed representation)

$$\boldsymbol{a}_j = \left(\|\boldsymbol{z}_j\|_2^2 \boldsymbol{K}_{DD}\right)^{-1} (\boldsymbol{K}_{YD} + \boldsymbol{K}_{DD}\boldsymbol{R})\boldsymbol{z}_j \tag{4}$$

and optimal representation (for fixed atom)

$$\boldsymbol{z}_j = \left(\boldsymbol{K}_{YD} - \boldsymbol{R}^\top \boldsymbol{K}_{DD}\right) \boldsymbol{a}_j. \tag{5}$$

The RKDL-D procedure of updating atoms is summarized in Algorithm 1. For brevity, we did not use special notations, but only the signals from $\boldsymbol{Y}$ where $\boldsymbol{a}_j$ appears in the representation are involved in the computation.

Algorithm 1: RKDL-D – update step

```
Data: complementary kernel matrix $K_{DD} \in \mathbb{R}^{p\times p}$
      partial kernel matrix $K_{YD} \in \mathbb{R}^{N\times p}$
      current dictionary $A \in \mathbb{R}^{N\times n}$
      representation matrix $Z \in \mathbb{R}^{n\times N}$
Result: updated dictionary $A$, representation $Z$
1 Compute sum $S = \sum_{i=1}^{n} Z_i^\top a_i^\top$
2 for $j = 1$ to $n$ do
3     Modify sum: $R = S - Z_j a_j^\top$
4     Update atom:
      $a_j = \left(\|z\|_2^2 K_{DD}\right)^{-1} \left(K_{YD}^\top + K_{DD} R\right) Z_j$
5     Normalize atom: $a_j \leftarrow a_j / \left(a_j^\top K_{DD} a_j\right)^{\frac{1}{2}}$
6     Update representation: $Z_j^\top \leftarrow (K_{YD} - R K_{DD}) a_j$
7     Recompute error: $S = R + Z_j a_j^\top$
```

B. Optimized Reduced Kernel Dictionary Learning

The improvement that we propose here is to update the dictionary $\boldsymbol{D}$, containing the kernel vectors, during the non-linear optimization procedure. We keep the idea of alternate optimization. The matrices $\boldsymbol{Z}$, $\boldsymbol{A}$ and $\boldsymbol{D}$ are updated successively. As above, for $\boldsymbol{Z}$ we use Kernel OMP and the atoms of $\boldsymbol{A}$ are updated as described by Algorithm 1. Updating the dictionary $\boldsymbol{D}$ must be done with a different procedure, detailed below. Since the dictionary $\boldsymbol{D}$ is updated together with the nonlinear representation, we call the resulting method Optimized Reduced Kernel Dictionary Learning (ORKDL-D).

In order to solve the optimization problem for $\boldsymbol{D}$, we update each column $\boldsymbol{d_j}$ independently, by the use of the trace form of the objective function of (3):

$$\operatorname{Tr}\left[\boldsymbol{K}_{YY} - 2\boldsymbol{K}_{YD}\boldsymbol{A}\boldsymbol{Z} + \boldsymbol{Z}^\top\boldsymbol{A}^\top\boldsymbol{K}_{DD}\boldsymbol{A}\boldsymbol{Z}\right].$$

We compute the partial derivatives with respect to the ith element of the current column $\boldsymbol{d}_j$, for both nonlinear terms

$$\begin{aligned}\frac{\partial \operatorname{Tr}[\boldsymbol{K}_{YD}\boldsymbol{A}\boldsymbol{Z}]}{\partial \boldsymbol{d_{ij}}} &= \operatorname{Tr}\left[\left(\frac{\partial \operatorname{Tr}[\boldsymbol{K}_{YD}\boldsymbol{A}\boldsymbol{Z}]}{\partial \boldsymbol{K}_{YD}}\right)^\top \cdot \frac{\partial \boldsymbol{K_{YD}}}{\partial \boldsymbol{d_{ij}}}\right]\\ &= \operatorname{Tr}\left[\boldsymbol{A}\boldsymbol{Z}\cdot\frac{\partial \boldsymbol{K_{YD}}}{\partial \boldsymbol{d_{ij}}}\right]\end{aligned}$$

and

$$\begin{aligned}\frac{\partial \operatorname{Tr}[\boldsymbol{Z}^\top\boldsymbol{A}^\top\boldsymbol{K}_{DD}\boldsymbol{A}\boldsymbol{Z}]}{\partial \boldsymbol{d_{ij}}} &= \operatorname{Tr}\left[\left(\frac{\partial \operatorname{Tr}[\boldsymbol{Z}^\top\boldsymbol{A}^\top\boldsymbol{K}_{DD}\boldsymbol{A}\boldsymbol{Z}]}{\partial \boldsymbol{K}_{DD}}\right)^\top \cdot \frac{\partial \boldsymbol{K_{DD}}}{\partial \boldsymbol{d_{ij}}}\right]\\ &= \operatorname{Tr}\left[\boldsymbol{A}\boldsymbol{Z}\boldsymbol{Z}^\top\boldsymbol{A}^\top\cdot\frac{\partial \boldsymbol{K_{DD}}}{\partial \boldsymbol{d_{ij}}}\right].\end{aligned}$$

The two partial derivatives of kernel matrices with respect to the current atom are sparse matrices with non-zero columns or rows only where their index is equal to the index of the current atom. The two matrices are computed as follows:

$$\frac{\partial \boldsymbol{K}_{YD}}{\partial \boldsymbol{d}_{ij}} = \begin{bmatrix} 0 & \cdots & \frac{\partial k(\boldsymbol{y}_1,\boldsymbol{d}_j)}{\partial \boldsymbol{d}_{ij}} & \cdots & 0\\ \vdots & & \frac{\partial k(\boldsymbol{y}_2,\boldsymbol{d}_j)}{\partial \boldsymbol{d}_{ij}} & & \vdots\\ \vdots & & \vdots & & \vdots\\ 0 & \cdots & \frac{\partial k(\boldsymbol{y}_N\boldsymbol{d}_j)}{\partial \boldsymbol{d}_{ij}} & \cdots & 0\end{bmatrix} \tag{6}$$

and

$$\frac{\partial \boldsymbol{K}_{DD}}{\partial \boldsymbol{d}_{ij}} = \begin{bmatrix} 0 & \cdots & \frac{\partial k(\boldsymbol{d}_1,\boldsymbol{d}_j)}{\partial \boldsymbol{d}_{ij}} & \cdots & 0\\ 0 & \cdots & \frac{\partial k(\boldsymbol{d}_2,\boldsymbol{d}_j)}{\partial \boldsymbol{d}_{ij}} & \cdots & 0\\ \vdots & & \vdots & & \vdots\\ \frac{\partial k(\boldsymbol{d}_j,\boldsymbol{d}_1)}{\partial \boldsymbol{d}_{ij}} & \cdots & \frac{\partial k(\boldsymbol{d}_j,\boldsymbol{d}_j)}{\partial \boldsymbol{d}_{ij}} & \cdots & \frac{\partial k(\boldsymbol{d}_j,\boldsymbol{d}_m)}{\partial \boldsymbol{d}_{ij}}\\ \vdots & & \vdots & & \vdots\\ 0 & \cdots & \frac{\partial k(\boldsymbol{d}_n,\boldsymbol{d}_j)}{\partial \boldsymbol{d}_{ij}} & \cdots & 0\end{bmatrix}. \tag{7}$$

Depending on the kernel function of interest, the partial derivatives are computed via

$$\frac{\partial k(\boldsymbol{x},\boldsymbol{y})}{\partial \boldsymbol{x}} = -\exp\frac{-\|\boldsymbol{x}-\boldsymbol{y}\|_2^2}{2\sigma^2}\frac{(\boldsymbol{x}-\boldsymbol{y})}{\sigma^2}$$

for the radial basis function kernel $k(\boldsymbol{x},\boldsymbol{y}) = \exp(\frac{-\|\boldsymbol{x}-\boldsymbol{y}\|_2^2}{\sigma^2})$, and

$$\frac{\partial k(\boldsymbol{x},\boldsymbol{y})}{\partial \boldsymbol{x}} = \beta\left(\boldsymbol{x}^\top\boldsymbol{y}+\alpha\right)^{\beta-1}\boldsymbol{y}$$

for the polynomial kernel $k(\boldsymbol{x},\boldsymbol{y}) = (\boldsymbol{x}^\top\boldsymbol{y}+\alpha)^\beta$.

Since explicit solutions to the optimization problem on $\boldsymbol{d}_j$ do not seem to exist, we update each column $\boldsymbol{d}_j$ through gradient descent procedure applied on each element. At gradient descent iteration ℓ, the next element value is computed via

$$\boldsymbol{d}_{ij}^{(\ell+1)} = \boldsymbol{d}_{ij}^{(\ell)} - \gamma\boldsymbol{g}_{ij}^{(\ell)},$$

where $\gamma \in \mathbb{R}_+$ represents the chosen learning rate and $\boldsymbol{g}_{ij}^{(\ell)}$ is the gradient

$$\boldsymbol{g}_{ij}^{(\ell)} = \operatorname{Tr}\left[\boldsymbol{A}\boldsymbol{Z}\left(\boldsymbol{Z}^\top\boldsymbol{A}^\top\cdot\frac{\partial \boldsymbol{K}_{DD}}{\partial \boldsymbol{d_{ij}}} - 2\cdot\frac{\partial \boldsymbol{K}_{YD}}{\partial \boldsymbol{d_{ij}}}\right)\right],$$

where the matrices (6) and (7) are computed for the dictionary at iteration ℓ. The update step of the ORKDL-D algorithm is obtained by introducing a few steps of gradient descent as described above for updating the dictionary $\boldsymbol{D}$, after the update of $\boldsymbol{A}$ described by Algorithm 1.

C. Mixed Optimized Reduced Kernel Dictionary Learning

The previous subsection presents an update procedure for the kernel vectors $\boldsymbol{D}$. A possible drawback is that direct optimization of the objective of (3) may lead to a dictionary $\boldsymbol{D}$ with weaker representation power for the entire set of signals, $\boldsymbol{Y}$. We present here a further improvement. To keep the representation significance of $\boldsymbol{D}$, which is a natural property to require in the context of kernel methods, we introduce the standard DL objective (1) into the ORKDL-D problem, obtaining

$$\begin{array}{cl}\min\limits_{\boldsymbol{A},\boldsymbol{X},\boldsymbol{Z}} & \|\varphi(\boldsymbol{Y}) - \varphi(\boldsymbol{D})\boldsymbol{A}\boldsymbol{Z}\|_F^2 + \lambda\|\boldsymbol{Y}-\boldsymbol{D}\boldsymbol{X}\|_F^2\\ \text{s.t.} & \|\boldsymbol{z}_\ell\|_0 \le s_z, \ell = 1:N\\ & \|\varphi(\boldsymbol{D})\boldsymbol{a}_j\| = 1, j = 1:n\\ & \|\boldsymbol{x}_\ell\|_0 \le s_x, \ell = 1:N\\ & \|\boldsymbol{d}_j\| = 1, j = 1:n,\end{array} \tag{8}$$

where $\lambda \in \mathbb{R}_+$ is a penalty constant. Since a mixed objective is proposed, we name this method Mixed Optimized Reduced Kernel Dictionary Learning (MORKDL-D).

By following two directions of optimization, we update the nonlinear representation while conserving the representation power of the linear space. The current optimization problem is solved similarly with the previous problems. The update of a column of $\boldsymbol{D}$ is slightly different. The gradient of the current atom uses also the partial derivative of the linear term and is

$$\tilde{\boldsymbol{g}}_{ij} = \boldsymbol{g}_{ij} - \boldsymbol{e}_{ij}$$

where $\boldsymbol{e}_{ij}$ represents the ith element of the vector $(\boldsymbol{Y}-\boldsymbol{D}\boldsymbol{X})\cdot \boldsymbol{X}_j^\top$; here, $\boldsymbol{X}_j^\top$ is the jth row of $\boldsymbol{X}$. On the other hand, the linear sparse representation matrix $\boldsymbol{X}$ is computed with the OMP algorithm. As before, the nonlinear sparse representation matrix $\boldsymbol{Z}$ is computed using the Kernel OMP algorithm, since $\boldsymbol{Z}$ and $\boldsymbol{X}$ can be optimized independently when $\boldsymbol{A}$ and $\boldsymbol{D}$ are fixed.

IV. Experiments

In this section we present the main results obtained with the proposed methods. We used three different data sets: Digits, MNIST and CIFAR-10. For each experiment we trained a nonlinear dictionary for representing the whole data set or a subset extracted by its corresponding label. For example, for the Digits data set we used all the signals during the training procedure, while for the MNIST and CIFAR-10 data sets we used signals specific to a selected label. For the MNIST data set we used label 5, while for the CIFAR-10 data set we used the first label.

All the algorithms were implemented in Matlab and Python and were run on a Desktop PC with Ubuntu 20.04 as operating system. The PC has a processor with 16 cores, with a base frequency of 2.90 GHz (Max Turbo Frequency 4.80 GHz), and 80 GB RAM memory. During the experiments, we computed the objective function of (3) (we report the values $\|\varphi(\boldsymbol{Y}) - \varphi(\boldsymbol{D})\boldsymbol{A}\boldsymbol{Z}\|_F/\sqrt{mN}$ of the error per signal element) at each iteration and measured the overall execution time. Note that for MORKDL-D we report the values of the same error, not of the objective of (8). Of course, for KDL we compute the objective of (2), which corresponds to the standard case $\boldsymbol{D} = \boldsymbol{Y}$. The error and execution time were computed by the mean on ten different rounds. As kernel function we used the radial basis function $k(\boldsymbol{x}, \boldsymbol{y}) = \exp(\frac{-|\boldsymbol{x}-\boldsymbol{y}|_2^2}{\sigma^2})$. The kernel parameters were chosen according to a grid search, and for all data sets we chose $\sigma = 10$. All the algorithms trained a nonlinear dictionary $\boldsymbol{A}$ of size 20, with sparsity level 4, using 10 iterations. For the reduced methods we initially trained a linear dictionary $\boldsymbol{D}$ of size 50, having a sparsity level of 5, with 10 iterations of the AK-SVD algorithm. The dictionary $\boldsymbol{D}$ dimensions ensures a reduced kernel by having a smaller number of atoms compared to the number of data signals. For example, the Digits data set contains 5000 signals of size 784, while the MNIST data set consists of approximately 6000 signals of the same size, of the same class. From the CIFAR-10 data set we used 5000 signals, specific to the interest class, of size 1024.

In the ORKDL-D and MORKDL-D algorithms, the dictionary $\boldsymbol{D}$ is updated with three gradient descent iterations. For each of the three data sets we chose a learning rate that ensures a smooth decrease of the objective function. We took $\gamma = 5 \cdot 10^{-4}$ for the Digits and MNIST data sets and $\gamma = 6 \cdot 10^{-4}$ for the CIFAR-10 data set. For all our experiments we used $\lambda = 1$.

For a better visualization of the results, we show the evolution of the nonlinear error (objective function of (3)) during the training procedure for each algorithm using the three data sets Digits (Figure 1), MNIST (Figure 2) and CIFAR-10 (Figure 3). As we can see, the improvements are visible from iteration two of training. More results are given in two tables, containing the value of the error at the last iteration (Table I) and the execution time (Table II). All the reduced methods achieve a smaller error compared to the KDL method. At first glance we notice that very large kernel spaces are not necessary to obtain good results. The introduction of a small dimensional space is enough to obtain satisfactory results. For situations where an improvement of the nonlinear representation space is needed, several iterations of gradient descent can be run as needed. According to the results obtained on the three data sets it can be seen that the biggest advantage of the proposed methods is the reduction of the execution time. For example, the non-optimized reduced form obtains an execution time at least six times shorter than the KDL method. In the problems where we try to train the kernel space, an additional time will be introduced (for dictionary $\boldsymbol{D}$ update), but the execution times are still shorter than those of KDL. For reduced kernel methods the execution time is reduced by approximately 20-45% with respect to the KDL problem. All the implementations are available at https://github.com/denisilie94/rkdl.

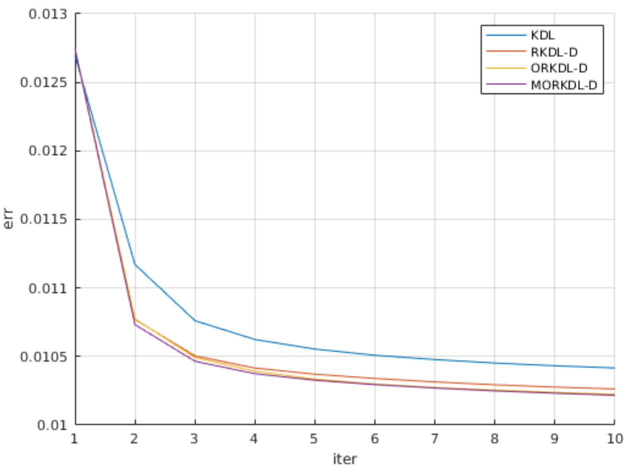

Fig. 1. Digits results

V. Conclusions

In this paper we have presented a new approach to the Kernel Dictionary Learning problem by introducing a reduced kernel and thus obtaining a new algorithm, named RKDL. This method is suitable for problems with large data sets, where standard KDL requires large memory sizes and long execution times. Moreover, we have demonstrated that most of the time

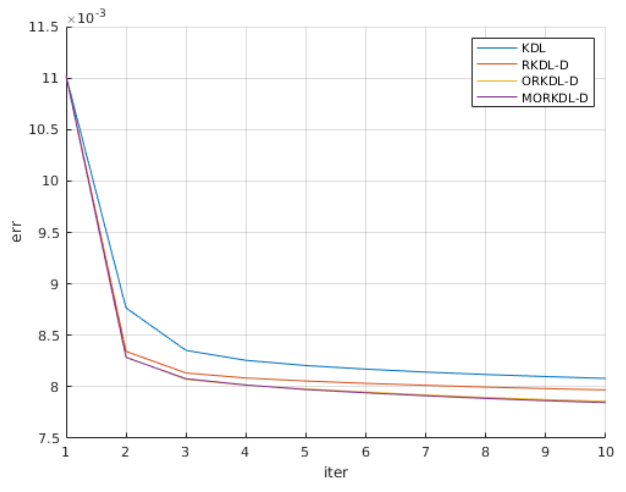

Fig. 2. MNIST results

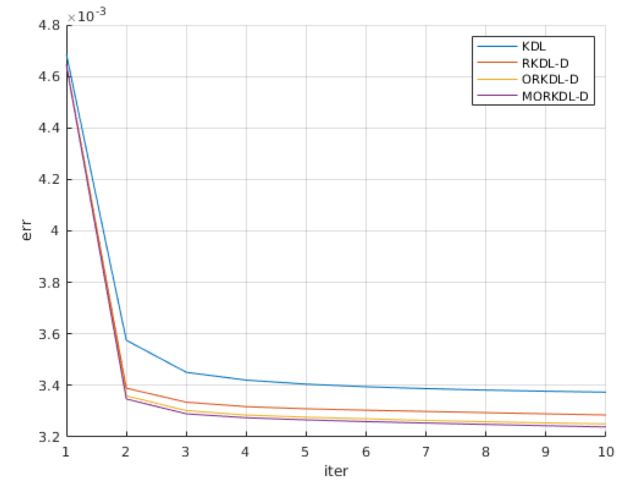

Fig. 3. CIFAR-10 results

TABLE I
Last error value

Error / Data set	Digits	MNIST	CIFAR-10
KDL	$1.041 \cdot 10^{-2}$	$8.081 \cdot 10^{-3}$	$3.373 \cdot 10^{-3}$
RKDL-D	$1.026 \cdot 10^{-2}$	$7.970 \cdot 10^{-3}$	$3.285 \cdot 10^{-3}$
ORKDL-D	$1.022 \cdot 10^{-2}$	$7.859 \cdot 10^{-3}$	$3.250 \cdot 10^{-3}$
MORKDL-D	$1.021 \cdot 10^{-2}$	$7.846 \cdot 10^{-3}$	$3.238 \cdot 10^{-3}$

TABLE II
Execution time

Time / Data set	Digits	MNIST	CIFAR-10
KDL	61.7	92.5	65.1
RKDL-D	9.2	10.9	9.6
ORKDL-D	39.2	48.4	51.6
MORKDL-D	38.9	49	51.4

a large representation space for the kernel matrix is not always needed to obtain satisfactory results. The reduced form of the kernel is enough to get similar results or even better. Since the kernel matrix is smaller, the required execution time is also much shorter.

The RKDL algorithm was presented under three different forms: standard RKDL-D, optimized RKDL-D and mixed optimized RKDL-D, the latter two including optimization of the kernel vectors. All of them obtain competitive results with the standard KDL problem.

References

[1] Hien Van Nguyen, Vishal M Patel, Nasser M Nasrabadi, and Rama Chellappa. Design of non-linear kernel dictionaries for object recognition. *IEEE Transactions on Image Processing*, 22(12):5123–5135, 2013.

[2] Jayaraman J Thiagarajan, Karthikeyan Natesan Ramamurthy, and Andreas Spanias. Multiple kernel sparse representations for supervised and unsupervised learning. *IEEE Transactions on Image Processing*, 23(7):2905–2915, 2014.

[3] Alona Golts and Michael Elad. Linearized kernel dictionary learning. *IEEE Journal of Selected Topics in Signal Processing*, 10(4):726–739, 2016.

[4] Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar. Sampling methods for the Nyström method. *The Journal of Machine Learning Research*, 13(1):981–1006, 2012.

[5] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

[6] Yuh-Jye Lee and Olvi L Mangasarian. RSVM: Reduced support vector machines. In *Proceedings of the 2001 SIAM International Conference on Data Mining*, pages 1–17. SIAM, 2001.

[7] Kuan-Ming Lin and Chih-Jen Lin. A study on reduced support vector machines. *IEEE Transactions on Neural Networks*, 14(6):1449–1459, 2003.

[8] Glenn M Fung, Olvi L Mangasarian, and Alexander J Smola. Minimal kernel classifiers. *Journal of Machine Learning Research*, 3(Nov):303–321, 2002.

[9] Wanyu Deng, Qinghua Zheng, and Kai Zhang. Reduced kernel extreme learning machine. In *Proceedings of the 8th international conference on computer recognition systems CORES 2013*, pages 63–69. Springer, 2013.

[10] Junlin Hu and Yap-Peng Tan. Nonlinear dictionary learning with application to image classification. *Pattern Recognition*, 75:282–291, 2018.

[11] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[12] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[13] Yagyensh Chandra Pati, Ramin Rezaiifar, and Perinkulam Sambamurthy Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Proceedings of 27th Asilomar conference on signals, systems and computers*, pages 40–44. IEEE, 1993.

[14] Bogdan Dumitrescu and Paul Irofti. *Dictionary learning algorithms and applications*. Springer, 2018.

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

[16] Ron Rubinstein, Michael Zibulevsky, and Michael Elad. Efficient implementation of the K-SVD algorithm using batch orthogonal matching pursuit. Technical report, Computer Science Department, Technion, 2008.

[17] Denis C. Ilie-Ablachim and Bogdan Dumitrescu. Anomaly detection with selective dictionary learning. In *International Conference on Control, Decision and Information Technologies*. IEEE, 2022.