Learning Parametric Dictionaries for Signals on Graphs

Dorina Thanou, David I Shuman, and Pascal Frossard

***Abstract*—In sparse signal representation, the choice of a dictionary often involves a tradeoff between two desirable properties – the ability to adapt to specific signal data and a fast implementation of the dictionary. To sparsely represent signals residing on weighted graphs, an additional design challenge is to incorporate the intrinsic geometric structure of the irregular data domain into the atoms of the dictionary. In this work, we propose a parametric dictionary learning algorithm to design data-adapted, structured dictionaries that sparsely represent graph signals. In particular, we model graph signals as combinations of overlapping local patterns. We impose the constraint that each dictionary is a concatenation of subdictionaries, with each subdictionary being a polynomial of the graph Laplacian matrix, representing a single pattern translated to different areas of the graph. The learning algorithm adapts the patterns to a training set of graph signals. Experimental results on both synthetic and real datasets demonstrate that the dictionaries learned by the proposed algorithm are competitive with and often better than unstructured dictionaries learned by state-of-the-art numerical learning algorithms in terms of sparse approximation of graph signals. In contrast to the unstructured dictionaries, however, the dictionaries learned by the proposed algorithm feature localized atoms and can be implemented in a computationally efficient manner in signal processing tasks such as compression, denoising, and classification.**

***Index Terms*—Dictionary learning, graph Laplacian, graph signal processing, sparse approximation.**

I. Introduction

GRAPHS are flexible data representation tools, suitable for modeling the geometric structure of signals that live on topologically complicated domains. Examples of signals residing on such domains can be found in social, transportation, energy, and sensor networks [1]. In these applications, the vertices of the graph represent the discrete data domain, and the edge weights capture the pairwise relationships between the vertices. A graph signal is then defined as a function that assigns a real value to each vertex. Some simple examples of graph signals are the current temperature at each location in a sensor network and the traffic level measured at predefined points of the transportation network of a city. An illustrative example is given in Fig. 1.

We are interested in finding meaningful graph signal representations that (i) capture the most important characteristics of the graph signals, and (ii) are sparse. That is, given a weighted graph and a class of signals on that graph, we want to construct an overcomplete dictionary of atoms that can sparsely represent graph signals from the given class as linear combinations of only a few atoms in the dictionary. An additional challenge when designing dictionaries for graph signals is that in order to identify and exploit structure in the data, we need to account for the intrinsic geometric structure of the underlying weighted graph. This is because signal characteristics such as smoothness depend on the topology of the graph on which the signal resides (see, e.g., [1, Example 1]).

For signals on Euclidean domains as well as signals on irregular data domains such as graphs, the choice of the dictionary often involves a tradeoff between two desirable properties – the ability to adapt to specific signal data and a fast implementation of the dictionary [2]. In the *dictionary learning* or *dictionary training* approach to dictionary design, numerical algorithms such as K-SVD [3] and the Method of Optimal Directions (MOD) [4] (see [2, Section IV] and references therein) learn a dictionary from a set of realizations of the data (training signals). The learned dictionaries are highly adapted to the given class of signals and therefore usually exhibit good representation performance. However, the learned dictionaries are highly non-structured, and therefore costly to apply in various signal processing tasks. On the other hand, analytic dictionaries based on signal transforms such as the Fourier, Gabor, wavelet, curvelet and shearlet transforms are based on mathematical models of signal classes (see [5] and [2, Section III] for a detailed overview of transform-based representations in Euclidean settings). These structured dictionaries often feature fast implementations, but they are not adapted to specific realizations of the data. Therefore, their ability to efficiently represent the data depends on the accuracy of the mathematical model of the data.

The gap between the transform-based representations and the numerically trained dictionaries can be bridged by imposing a structure on the dictionary and learning the parameters of this structure. The structure generally incorporates desirable properties of the dictionary such as translation invariance [6], minimum coherence [7] or efficient implementation [8] (see [2, Section IV.E] for a complete list of references). Structured dictionaries represent a good trade-off between approximation performance and efficiency of the implementation.

In this work, we build on our previous work [9] and capitalize on the benefits of both numerical and analytical approaches by learning a dictionary that incorporates the graph structure and

Manuscript received December 18, 2013; revised April 14, 2014 and June 08, 2014; accepted June 08, 2014. Date of publication June 24, 2014; date of current version July 10, 2014. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Hing Cheung So. This work has been partially funded by the Swiss National Science Foundation under Grant 200021_135493. Part of the work reported here was presented at the IEEE Glob. Conf. Signal and Inform. Process., Austin, TX, USA, December 2013.

D. Thanou and P. Frossard are with Ecole Polytechnique Fédérale de Lausanne (EPFL), Signal Processing Laboratory-LTS4, CH-1015, Lausanne, Switzerland (e-mail: dorina.thanou@epfl.ch; pascal.frossard@epfl.ch).

D. I. Shuman is with the Department of Mathematics, Statistics, and Computer Science, Macalester College, Saint Paul, MN 55105 USA (e-mail: dshuman1@macalester.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSP.2014.2332441

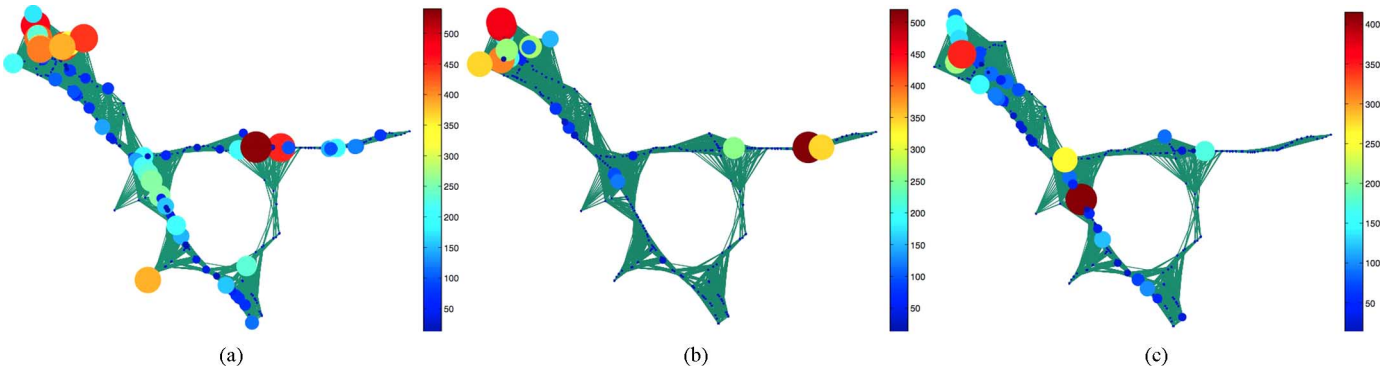

Fig. 1. Illustrative example: The three signals on the graph are the minutes of bottlenecks per day at different detector stations in Alameda County, California, on three different days. The detector stations are the nodes of the graph and the connectivity is defined based on the GPS coordinates of the stations. The size and the color of each ball indicate the value of the signal at each vertex of the graph. Note that all signals consist of a set of localized features positioned on different nodes of the graph. (a) Day 1. (b) Day 2. (c) Day 3.

can be implemented efficiently. We model the graph signals as combinations of overlapping local patterns, describing localized events or causes on the graph, that can appear in different vertices. That could be the case in graph signals for traffic data, brain data, or other type of networks. For example, the evolution of traffic on a highway might be similar to that on a different highway, at a different position in the transportation network. We incorporate the underlying graph structure into the dictionary through the graph Laplacian operator, which encodes the connectivity. In order to ensure the atoms are localized in the graph vertex domain, we impose the constraint that our dictionary is a concatenation of subdictionaries that are polynomials of the graph Laplacian [10]. We then learn the coefficients of the polynomial kernels via numerical optimization. As such, our approach falls into the category of parametric dictionary learning [2, Section IV.E]. The learned dictionaries are adapted to the training data, efficient to store, and computationally efficient to apply. Experimental results demonstrate the effectiveness of our scheme in the approximation of both synthetic signals and graph signals collected from real world applications. In addition to signal approximation, the localization of the atoms in the graph domain leads to an easier interpretation of the data from their atomic representations.

The structure of the paper is as follows. We first highlight some related work on the representation of graph signals in Section II. In Section III, we recall basic definitions related to graphs that are necessary to understand our dictionary learning algorithm. We describe the polynomial dictionary structure and the dictionary learning algorithms in Section IV. In Section V, we evaluate the performance of our algorithm on the approximation of both synthetic and real world graph signals. Finally, we discuss the benefits of the polynomial structure in Section VI.

II. Related Work

The design of overcomplete dictionaries to sparsely represent signals has been extensively investigated in the past few years. We restrict our focus here to the literature related to the problem of designing dictionaries for graph signals. Generic numerical approaches such as K-SVD [3] and MOD [4] can certainly be applied to graph signals, with these signals viewed as vectors in $\mathbb{R}^N$. However, the learned dictionaries will neither feature a fast implementation, nor explicitly incorporate the underlying graph structure.

Meanwhile, several transform-based dictionaries for graph signals have recently been proposed (see [1] for an overview and complete list of references). For example, the graph Fourier transform has been shown to sparsely represent smooth graph signals [11]; wavelet transforms such as diffusion wavelets [12], spectral graph wavelets [10], and critically sampled two-channel wavelet filter banks [13] target piecewise-smooth graph signals; the multiscale wavelets of [14], the critically sampled, generalized tree-based wavelets transform of [15] and its extension to a redundant wavelet transform in [16] exploit the tree structure of the data to represent signals defined on weighted graphs; and vertex-frequency frames [17]–[19] can be used to analyze signal content at specific vertex and frequency locations. These dictionaries feature pre-defined structures derived from the graph and some of them can be efficiently implemented; however, they generally are not adapted to the signals at hand. Some exceptions are the diffusion wavelet packets of [20], the wavelets on graphs via deep learning[21], and the tree-based wavelets [15], [16], which feature extra adaptivity.

The recent work in [22] tries to bridge the gap between the graph-based transform methods and the purely numerical dictionary learning algorithms by proposing an algorithm to learn structured graph dictionaries. The learned dictionaries have a structure that is derived from the graph topology, while its parameters are learned from the data. This work is the closest to ours in a sense that both graph dictionaries consist of subdictionaries that are based on the graph Laplacian. However, it does not necessarily lead to efficient implementations as the obtained dictionary is not necessarily a smooth matrix function (see, e.g., [23] for more on matrix functions) of the graph Laplacian matrix.

Finally, we remark that the graph structure is taken into consideration in [24], not explicitly into the dictionary but rather in the sparse coding coefficients. The authors use the graph Laplacian operator as a regularizer in order to impose that the obtained sparse coding coefficients vary smoothly along the geodesics of the manifold that is captured by the graph. However, the obtained dictionary does not have any particular structure. None of the previous works are able to design dictionaries that provide sparse representations, particularly adapted to a given class of graph signals, and have efficient implementations. This is exactly the objective of our work, where a structured graph signal

dictionary is composed of multiple polynomial matrix functions of the graph Laplacian.

III. Preliminaries

In this section, we briefly review a few basic definitions for signals on graphs. A more complete description of the graph signal processing framework can be found in [1]. We consider a weighted and undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$ where $\mathcal{V}$ and $\mathcal{E}$ represent the vertex and edge sets of the graph, and W represents the matrix of edge weights, with $W_{ij} = W_{ji}$ denoting the positive weight of an edge connecting vertices i and j; otherwise $W_{ij} = 0$. We assume that the graph is connected. The graph Laplacian operator is defined as $L = D - W$, where D is the diagonal degree matrix whose i^{th} diagonal element is equal to the sum of the weights of all the edges incident to vertex i [25]. The normalized graph Laplacian is defined as $\mathcal{L} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$. Both operators are real symmetric and positive semidefinite matrices and they have a complete set of real orthonormal eigenvectors with corresponding nonnegative eigenvalues. We denote the eigenvectors of the normalized graph Laplacian by $\chi = [\chi_0, \chi_1, \ldots, \chi_{N-1}]$, and the spectrum of eigenvalues by

$$\sigma(\mathcal{L}) := \Big\{0 = \lambda_0 < \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_{N-1} \leq 2\Big\}.$$

A graph signal y in the vertex domain is a real-valued function defined on the vertices of the graph $\mathcal{G}$, such that $y(v)$ is the value of the function at vertex $v \in \mathcal{V}$. The spectral domain representation can also provide significant information about the characteristics of graph signals. In particular, the eigenvectors of the Laplacian operators can be used to perform harmonic analysis of signals that live on the graph, and the corresponding eigenvalues carry a notion of frequency [1]. The normalized Laplacian eigenvectors are a Fourier basis, so that for any function y defined on the vertices of the graph, the graph Fourier transform $\hat{y}$ at frequency λ_ℓ is defined as

$$\hat{y}(\lambda_\ell) = \langle y, \chi_\ell \rangle = \sum_{n=1}^{N} y(n) \chi_\ell^*(n),$$

while the inverse graph Fourier transform is

$$y(n) = \sum_{\ell=0}^{N-1} \hat{y}(\lambda_\ell)\, \chi_\ell(n), \quad \forall n \in \mathcal{V}.$$

Besides its use in harmonic analysis, the graph Fourier transform is also useful in defining the translation of a signal on the graph. The generalized translation operator can be defined as a generalized convolution with a Kronecker δ function centered at vertex n [10], [17], [18]:

$$T_n g = \sqrt{N}(g * \delta_n) \overset{(a)}{=} \sqrt{N} \sum_{\ell=0}^{N-1} \hat{g}(\lambda_\ell) \chi_\ell^*(n) \chi_\ell, \qquad (1)$$

where the normalizing constant $\sqrt{N}$ ensures that the translation operator preserves the mean of a signal. Moreover, (a) follows from the property that convolution in the vertex domain is equivalent to multiplication in the graph spectral domain, where the eigenvectors of the Laplacian are used as the Fourier basis in graph settings. The right-hand side of (1) allows us to interpret the generalized translation as an operator acting on the kernel $\hat{g}(\cdot)$, which is defined directly in the graph spectral domain. The localization of $T_n g$ around the center vertex n is controlled by the smoothness of the kernel $\hat{g}(\cdot)$ [10], [18]. One can thus design atoms $T_n g$ that are localized around n in the vertex domain by taking the kernel $\hat{g}(\cdot)$ in (1) to be a smooth polynomial function of degree K:

$$\hat{g}(\lambda_\ell) = \sum_{k=0}^{K} \alpha_k \lambda_\ell^k, \quad \ell = 0, \ldots, N-1. \qquad (2)$$

Combining (1) and (2), we can translate a polynomial kernel g to a vertex n in the graph as

$$\begin{aligned} T_n g &= \sqrt{N}(g * \delta_n) = \sqrt{N} \sum_{\ell=0}^{N-1} \sum_{k=0}^{K} \alpha_k \lambda_\ell^k \chi_\ell^*(n) \chi_\ell \\ &= \sqrt{N} \sum_{k=0}^{K} \alpha_k \sum_{\ell=0}^{N-1} \lambda_\ell^k \chi_\ell^*(n) \chi_\ell = \sqrt{N} \sum_{k=0}^{K} \alpha_k (\mathcal{L}^k)_n, \end{aligned} \qquad (3)$$

where $(\mathcal{L}^k)_n$ denotes the n^{th} column of the matrix $\mathcal{L}^k$. The concatenation of N such columns allows us to generate a set of N localized atoms, which are the columns of

$$T g = \sqrt{N} \hat{g}(\mathcal{L}) = \sqrt{N} \chi \hat{g}(\Lambda) \chi^T = \sqrt{N} \sum_{k=0}^{K} \alpha_k \mathcal{L}^k, \qquad (4)$$

where Λ is the diagonal matrix of the eigenvalues. In short, if $\hat{g}(\cdot)$ is a degree K polynomial, then $(T_n g)(i) = 0$ for all vertices i more than K hops from the center vertex n; that is, in the vertex domain, the support of the kernel translated to center vertex n is contained in a ball of K hops from vertex n [10, Lemma 5.2], [18, Lemma 2]. Furthermore, within this ball, the smoothness properties of the polynomial kernel can be used to estimate the decay of the magnitude $|(T_n g)(i)|$ as the distance from n to i increases [18, Section 4.4].

IV. Parametric Dictionary Learning on Graphs

Given a set of training signals on a weighted graph, our objective is to learn a structured dictionary that sparsely represents classes of graph signals. We consider a general class of graph signals that are linear combinations of (overlapping) graph patterns positioned at different vertices on the graph. The latter implies that the signal model is not necessarily the same across all the vertices but it can differ across the different neighborhoods. We aim to learn a dictionary that is capable of capturing all possible translations of a set of patterns. We use the definition (1) of generalized translation, and we learn a set of polynomial generating kernels (i.e., patterns) of the form (2) that capture the main characteristics of the signals in the spectral domain. Learning directly in the spectral domain enables us to detect spectral components that exist in our training signals, such as atoms that are supported on selected frequency components. In this section, we describe in detail the structure of our dictionary and the learning algorithm.

A. Dictionary Structure

We design a structured graph dictionary $\mathcal{D} = [\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_S]$ that is a concatenation of a set of S subdictionaries of the form

$$\mathcal{D}_s = \widehat{g_s}(\mathcal{L}) = \chi \left(\sum_{k=0}^{K} \alpha_{sk} \Lambda^k \right) \chi^T = \sum_{k=0}^{K} \alpha_{sk} \mathcal{L}^k, \qquad (5)$$

where $\widehat{g_s}(\cdot)$ is the generating kernel or pattern of the subdictionary $\mathcal{D}_s$. Note that the atom given by column n of subdictionary $\mathcal{D}_s$ is equal to $\frac{1}{\sqrt{N}} T_n g_s$; i.e., the polynomial $\widehat{g_s}(\cdot)$ of

order K translated to the vertex n. The polynomial structure of $\widehat{g}_s(\cdot)$ ensures that the resulting atom given by column n of subdictionary $\mathcal{D}_s$ has its support contained in a K-hop neighborhood of vertex n [10, Lemma 5.2].

The polynomial constraint guarantees the localization of the atoms in the vertex domain, but it does not provide any information about the spectral representation of the atoms. In order to control their frequency behavior, we impose two constraints on the spectral representation of the kernels $\{\widehat{g}_s(\cdot)\}_{s=1,2,\ldots,S}$. First, we require that the kernels are nonnegative and uniformly bounded by a given constant c. In other words, we impose that $0 \leq \widehat{g}_s(\lambda) \leq c$ for all $\lambda \in [0, \lambda_{\max}]$, or, equivalently,

$$0 \preceq \mathcal{D}_s \preceq cI, \ \ \forall s \in \{1, 2, \ldots, S\}, \tag{6}$$

where I is the $N \times N$ identity matrix. Each subdictionary $\mathcal{D}_s$ has to be a positive semi-definite matrix whose maximum eigenvalue is upper bounded by c.

Second, since the classes of signals under consideration usually contain frequency components that are spread across the entire spectrum, the learned kernels $\{\widehat{g}_s(\cdot)\}_{s=1,2,\ldots,S}$ should also cover the full spectrum. We thus impose the constraint $c - \epsilon_1 \leq \sum_{s=1}^{S} \widehat{g}_s(\lambda) \leq c + \epsilon_2$, for all $\lambda \in [0, \lambda_{\max}]$, or equivalently

$$(c - \epsilon_1)I \preceq \sum_{s=1}^{S} \mathcal{D}_s \preceq (c + \epsilon_2)I, \tag{7}$$

where ϵ_1, ϵ_2 are small positive constants. Note that both (6) and (7) are quite generic and do not assume any particular prior on the spectral behavior of the atoms. If we have additional prior information, we can incorporate that prior into our optimization problem by modifying these constraints. For example, if we know that our signals' frequency content is restricted to certain parts of the spectrum, by choosing ϵ_1 close to c, we relax the constraint on the coverage of the entire spectrum, and we give the flexibility to our learning algorithm to learn filters covering only part of it.

Finally, the spectral constraints increase the stability of the dictionary. From the constants c, ϵ_1 and ϵ_2, we can derive frame bounds for $\mathcal{D}$, as shown in the following proposition.

Proposition 1: Consider a dictionary $\mathcal{D} = [\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_S]$, where each $\mathcal{D}_s$ is of the form of $\mathcal{D}_s = \sum_{k=0}^{K} \alpha_{sk} \mathcal{L}^k$. If the kernels $\{\widehat{g}_s(\cdot)\}_{s=1,2,\ldots,S}$ satisfy the constraints $0 \leq \widehat{g}_s(\lambda) \leq c$ and $c - \epsilon_1 \leq \sum_{s=1}^{S} \widehat{g}_s(\lambda) \leq c + \epsilon_2$, for all $\lambda \in [0, \lambda_{\max}]$ then the set of atoms $\{d_{s,n}\}_{s=1,2,\ldots,S, n=1,2,\ldots,N}$ of $\mathcal{D}$ form a frame. Namely, for every signal $y \in \mathbb{R}^N$,

$$\frac{(c - \epsilon_1)^2}{S} \|y\|_2^2 \leq \sum_{n=1}^{N} \sum_{s=1}^{S} |\langle y, d_{s,n} \rangle|^2 \leq (c + \epsilon_2)^2 \|y\|_2^2.$$

Proof: From [19, Lemma 1], which is a slight generalization of [10, Theorem 5.6], we have

$$\sum_{n=1}^{N} \sum_{s=1}^{S} |\langle y, d_{s,n} \rangle|^2 = \sum_{\ell=0}^{N-1} |\hat{y}(\lambda_\ell)|^2 \sum_{s=1}^{S} |\widehat{g}_s(\lambda_\ell)|^2, \ \ \forall \lambda \in \sigma(\mathcal{L}). \tag{8}$$

From the constraints on the spectrum of kernels $\{\widehat{g}_s(\cdot)\}_{s=1,2,\ldots,S}$ we have

$$\sum_{s=1}^{S} |\widehat{g}_s(\lambda_\ell)|^2 \leq \left(\sum_{s=1}^{S} \widehat{g}_s(\lambda_\ell) \right)^2 \leq (c + \epsilon_2)^2, \ \ \forall \lambda \in \sigma(\mathcal{L}). \tag{9}$$

Moreover, from the left side of (7) and the Cauchy-Schwarz inequality, we have

$$\frac{(c - \epsilon_1)^2}{S} \leq \frac{\left(\sum_{s=1}^{S} \widehat{g}_s(\lambda_\ell) \right)^2}{S} \leq \sum_{s=1}^{S} |\widehat{g}_s(\lambda_\ell)|^2, \ \ \forall \lambda \in \sigma(\mathcal{L}). \tag{10}$$

Combining (8), (9) and (10) yields the desired result. ■

We remark that if we alternatively impose that $\sum_{s=1}^{S} |\widehat{g}_s(\lambda)|^2$ is constant for all $\lambda \in [0, \lambda_{\max}]$, the resulting dictionary $\mathcal{D}$ would be a tight frame. However, such a constraint leads to a dictionary update step that is non-convex and requires optimization techniques that are different from the one described in the next section. To summarize, the polynomial dictionary $\mathcal{D}$ is a parametric dictionary that depends on the parameters $\{\alpha_{sk}\}_{s=1,2,\ldots,S;\ k=1,2,\ldots,K}$, and the constraints (6) and (7) can be viewed as constraints on these parameters. They provide some control on the spectral representation of the atoms and the stability of signal reconstruction with the learned dictionary. Finally, we note here that for the design of the dictionary, we have used the normalized graph Laplacian eigenvectors as the Fourier basis. Given the polynomial structure of our dictionary, the upper bound of $\lambda_{N-1} \leq 2$ on the spectrum of the normalized Laplacian makes it more appropriate for our framework. The unnormalized Laplacian contains eigenvectors that have similar interpretation in terms of frequency. However, its eigenvalues can have a large magnitude, causing some numerical instabilities when taking large powers.

B. Dictionary Learning Algorithm

Given a set of training signals $Y = [y_1, y_2, \ldots, y_M] \in \mathbb{R}^{N \times M}$, all living on the weighted graph $\mathcal{G}$, our objective is to learn a graph dictionary $\mathcal{D} \in \mathbb{R}^{N \times NS}$ with the structure described in Section IV-A that can efficiently represent all of the signals in Y as linear combinations of only a few of its atoms. Since $\mathcal{D}$ has the form (5), this is equivalent to learning the parameters $\{\alpha_{sk}\}_{s=1,2,\ldots,S;\ k=1,2,\ldots,K}$ that characterize the set of generating kernels, $\{\widehat{g}_s(\cdot)\}_{s=1,2,\ldots,S}$. We denote these parameters in vector form as $\alpha = [\alpha_1; \ldots; \alpha_S]$, where α_s is a column vector with $(K + 1)$ entries.

Therefore, the dictionary learning problem can be cast as the following optimization problem:

$$\begin{aligned}
\underset{\alpha \in \mathbb{R}^{(K+1)S},\ X \in \mathbb{R}^{SN \times M}}{\operatorname{argmin}} \quad & \left\{ \|Y - \mathcal{D}X\|_F^2 + \mu \|\alpha\|_2^2 \right\} \\
\text{subject to} \quad & \|x_m\|_0 \leq T_0, \quad \forall m \in \{1, \ldots, M\}, \\
& \mathcal{D}_s = \sum_{k=0}^{K} \alpha_{sk} \mathcal{L}^k, \ \forall s \in \{1, 2, \ldots, S\} \\
& 0 \preceq \mathcal{D}_s \preceq c, \quad \forall s \in \{1, 2, \ldots, S\} \\
& (c - \epsilon_1) I \preceq \sum_{s=1}^{S} \mathcal{D}_s \preceq (c + \epsilon_2) I,
\end{aligned} \tag{11}$$

where $\mathcal{D} = [\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_S]$, x_m corresponds to column m of the coefficient matrix X, and T_0 is the sparsity level of the coefficients of each signal. Note that in the objective of the optimization problem (11), we penalize the norm of the polynomial coefficients α in order to (i) promote smoothness in the learned polynomial kernels, and (ii) improve the numerical stability of the learning algorithm. In practice, a small value of μ is enough to guarantee the stability of the solution while preserving large values in the polynomial coefficients. The value of the parameter c does not affect the frequency behavior nor the localization of the atoms. It simply scales the magnitude of the kernel coefficients. Finally, the values of ϵ_1, ϵ_2 are generally chosen to be arbitrarily small, unless prior information, like frequency spread information, indicates otherwise.

The optimization problem (11) is not convex, but it can be approximately solved in a computationally efficient manner by alternating between the sparse coding and dictionary update steps. In the first step, we fix the parameters α (and accordingly fix the dictionary $\mathcal{D}$ via the structure (5)) and solve

$$\underset{X}{\operatorname{argmin}} \|Y - \mathcal{D}X\|_F^2 \quad \text{subject to } \|x_m\|_0 \leq T_0, \tag{12}$$

for all $m \in \{1, \ldots, M\}$, using orthogonal matching pursuit (OMP) [26], [27], which has been shown to perform well in the dictionary learning literature. Before applying OMP, we normalize the atoms of the dictionary so that they all have a unit norm. This step is essential for the OMP algorithm in order to treat all of the atoms equally. After computing the coefficients X, we renormalize the atoms of our dictionary to recover our initial polynomial structure [28, Chapter 3.1.4] and the sparse coding coefficients in such a way that the product $\mathcal{D}X$ remains constant.

In the second step, we fix the coefficients X and update the dictionary by finding the vector of parameters, α, that solves

$$\begin{aligned} \underset{\alpha \in \mathbb{R}^{(K+1)S}}{\operatorname{argmin}} \quad & \left\{\|Y - \mathcal{D}X\|_F^2 + \mu\|\alpha\|_2^2\right\} \\ \text{subject to} \quad & \mathcal{D}_s = \sum_{k=0}^{K} \alpha_{sk}\mathcal{L}^k, \quad \forall s \in \{1, 2, \ldots, S\} \\ & 0 \preceq \mathcal{D}_s \preceq cI, \quad \forall s \in \{1, 2, \ldots, S\} \\ & (c - \epsilon_1)I \preceq \sum_{s=1}^{S} \mathcal{D}_s \preceq (c + \epsilon_2)I. \end{aligned} \tag{13}$$

The optimization problem (13) is a quadratic program [29] as it consists of a quadratic objective function and a set of affine constraints. In particular, the objective function is written as

$$\begin{aligned} &\|Y - \mathcal{D}X\|_F^2 + \mu\|\alpha\|_2^2 \\ &= \sum_{n=1}^{N}\sum_{m=1}^{M} (Y - \mathcal{D}X)_{nm}^2 + \mu\alpha^T\alpha \\ &= \sum_{n=1}^{N}\sum_{m=1}^{M} \left(Y - \sum_{s=1}^{S}\sum_{k=0}^{K} \alpha_{sk}\mathcal{L}^k X_s\right)_{nm}^2 + \mu\alpha^T\alpha, \end{aligned} \tag{14}$$

where $X_s \in \mathbb{R}^{N \times M}$ denotes the rows of the matrix X corresponding to the atoms in the subdictionary $\mathcal{D}_s$. Let us define the column vector $P_{nm}^s \in \mathbb{R}^{(K+1)}$ as

$$P_{nm}^s = [(\mathcal{L}^0)_{(n,:)} X_{s(:,m)}; (\mathcal{L}^1)_{(n,:)} X_{s(:,m)}; \ldots; (\mathcal{L}^K)_{(n,:)} X_{s(:,m)}],$$

where $(\mathcal{L}^k)_{(n,:)}$ is the n^{th} row of the k^{th} power of the Laplacian matrix and $X_{s(:,m)}$ is the m^{th} column of the matrix X_s. We then stack these column vectors into the column vector $P_{nm} \in \mathbb{R}^{S(K+1)}$, which is defined as $P_{nm} = [P_{nm}^1; P_{nm}^2; \ldots; P_{nm}^S]$. Using this definition of P_{nm}, (14) can be written as

$$\begin{aligned} &\|Y - \mathcal{D}X\|_F^2 + \mu\|\alpha\|_2^2 \\ &= \sum_{n=1}^{N}\sum_{m=1}^{M} (Y_{nm} - P_{nm}^T\alpha)^2 + \mu\alpha^T\alpha \\ &= \sum_{n=1}^{N}\sum_{m=1}^{M} Y_{nm}^2 - 2Y_{nm}P_{nm}^T\alpha + \alpha^T P_{nm}P_{nm}^T\alpha + \mu\alpha^T\alpha \\ &= \|Y\|_F^2 - 2\left(\sum_{n=1}^{N}\sum_{m=1}^{M} Y_{nm}P_{nm}^T\right)\alpha \\ &\quad + \alpha^T\left(\sum_{n=1}^{N}\sum_{m=1}^{M} P_{nm}P_{nm}^T + \mu I_{S(K+1)}\right)\alpha, \end{aligned}$$

where $I_{S(K+1)}$ is the $S(K+1) \times S(K+1)$ identity matrix. The matrix $\sum_{n=1}^{N}\sum_{m=1}^{M} P_{nm}P_{nm}^T + \mu I$ is positive definite, which implies that our objective is quadratic.

Finally, the optimization constraints (6), (7) can be expressed as affine functions of α with

$$\begin{aligned} 0 \leq I_S \otimes B\alpha \leq c\mathbf{1}, \\ (c - \epsilon)\mathbf{1} \leq \mathbf{1}^T \otimes B\alpha \leq (c + \epsilon)\mathbf{1}\,, \end{aligned}$$

where the inequalities are component-wise inequalities, $\mathbf{1}$ is the vector of ones, $\otimes$ is the Kronecker product, I_S is the $S \times S$ identity matrix, and B is the Vandermonde matrix

$$B = \begin{bmatrix} 1 & \lambda_0 & \lambda_0^2 \ldots \lambda_0^K \\ 1 & \lambda_1 & \lambda_1^2 \ldots \lambda_1^K \\ \vdots & \vdots & \vdots \\ 1 & \lambda_{N-1} & \lambda_{N-1}^2 \ldots \lambda_{N-1}^K \end{bmatrix}.$$

Thus, the coefficients of the polynomials can be found by solving the following quadratic optimization problem:

$$\begin{aligned} \underset{\alpha \in \mathbb{R}^{(K+1)S}}{\operatorname{argmin}} \quad & \alpha^T\left(\sum_{n=1}^{N}\sum_{m=1}^{M} P_{nm}P_{nm}^T + \mu I_{S(K+1)}\right)\alpha \\ & - 2\left(\sum_{n=1}^{N}\sum_{m=1}^{M} Y_{nm}P_{nm}^T\right)\alpha \\ \text{subject to} \quad & 0 \leq I_S \otimes B\alpha \leq c\mathbf{1} \\ & (c - \epsilon)\mathbf{1} \leq \mathbf{1}^T \otimes B\alpha \leq (c + \epsilon)\mathbf{1}. \end{aligned} \tag{15}$$

Algorithm 1 contains a summary of the basic steps of our dictionary learning algorithm. We can initialize the dictionary by either generating a set of polynomial kernels that satisfy the constraints imposed in the learning or simply generating for each kernel $\widehat{g}_s$, a set of discrete values $\widehat{g}_s(\lambda_0), \widehat{g}_s(\lambda_1), \ldots, \widehat{g}_s(\lambda_{N-1})$

uniformly distributed in the range between 0 and c. Each subdictionary is then set to be $\mathcal{D}_s = \chi \widehat{g}_s(\Lambda)\chi^T$. Since the optimization problem (11) is solved by alternating between the two steps, the polynomial dictionary learning algorithm is not guaranteed to converge to the optimal solution; practically, we observed in most of our experiments that the total representation error $\|Y - \mathcal{D}X\|_F^2$ either reduced or remained constant over the iterations, which implies that the algorithm tends to converge to a local optimum. Finally, the overall complexity of the algorithm at each iteration depends on the complexity of both the sparse coding algorithm, and the quadratic program for the dictionary update step. In the dictionary update step, the quadratic program (line 10 of Algorithm 1) can be efficiently solved in polynomial time using optimization techniques such as interior point methods [29] or operator splitting methods (e.g., Alternating Direction Method of Multipliers [30]). The former methods lead to more accurate solutions, while the latter are better suited to solve large scale problems. In applications where the computational time is crucial and the graph is sparse, it would be interesting to employ an optimized OMP implementation, or rely on first order methods such as the iterative soft thresholding, by exploiting the polynomial structure of the dictionary, as described in Section VI. For the numerical examples in this paper, we generally use OMP and interior point methods to solve the sparse coding step and the quadratic optimization problem respectively.

V. Experimental Results

In the following experiments, we quantify the performance of the proposed dictionary learning method in the approximation of both synthetic and real data. First, we study the behavior of our algorithm in the synthetic scenario where the signals are linear combinations of a few localized atoms that are placed on different vertices of the graph. Then, we study the performance of our algorithm in the approximation of graph signals collected from real world applications. In all experiments, we compare the performance of our algorithm to the performance of (i) graph-based transform methods such as the spectral graph wavelet transform (SGWT)[10], (ii) purely numerical dictionary learning methods such as K-SVD [3] that treat the graph signals as vectors in $\mathbb{R}^N$ and ignore the graph structure, and (iii) the graph-based dictionary learning algorithm presented in [22]. The kernel bounds in (11), if not otherwise specified, are chosen as $c = 1$ and $\epsilon_1 = \epsilon_2 = 0.01$, and the number of iterations in the learning algorithm is fixed to 25. Moreover, we set $\mu = 10^{-4}$ and we initialize the dictionary by generating for each kernel $\widehat{g}_s$, a set of discrete values $\widehat{g}_s(\lambda_0), \widehat{g}_s(\lambda_1), \ldots, \widehat{g}_s(\lambda_{N-1})$ uniformly distributed in the range between 0 and c. Each subdictionary is then set to $\mathcal{D}_s = \chi \widehat{g}_s(\Lambda)\chi^T$. The sparsity level in the learning phase is set to $T_0 = 4$ for all the synthetic experiments. We use the *sdpt3* solver [31] in the *yalmip* optimization toolbox [32] to solve the quadratic problem (13) in the learning algorithm. In order to directly compare the methods mentioned above, we always use orthogonal matching pursuit (OMP) for the sparse coding step in the testing phase, where we first normalize the dictionary atoms to a unit norm. Finally, the average normalized approximation error is defined as $\frac{1}{|Y_{test}|}\sum_{m=1}^{|Y_{test}|} \|Y_m - \mathcal{D}X_m\|_2^2/\|Y_m\|^2$, where $|Y_{test}|$ is the cardinality of the testing set.

Algorithm 1: Parametric Dictionary Learning on Graphs

1: **Input:** Signal set Y, initial dictionary $\mathcal{D}^{(0)}$, target signal sparsity T_0, polynomial degree K, number of subdictionaries S, number of iterations $iter$

2: **Output:** Sparse signal representations X, polynomial coefficients α

3: **Initialization:** $\mathcal{D} = \mathcal{D}^{(0)}$

4: **for** $i = 1, 2, \ldots, iter$ **do:**

5: **Sparse Approximation Step:**

6: (a) Scale each atom in $\mathcal{D}$ to a unit norm

7: (b) Update X using (12)

8: (c) Rescale X, $\mathcal{D}$ to recover the polynomial structure

9: **Dictionary Update Step:**

10: Compute the polynomial coefficients α by solving (15), and update the dictionary according to (5)

11: **end for**

A. Synthetic Signals

We first study the performance of our algorithm for the approximation of synthetic signals. We generate a graph by randomly placing $N = 100$ vertices in the unit square. We set the edge weights based on a thresholded Gaussian kernel function so that $W(i,j) = e^{-\frac{[dist(i,j)]^2}{2\theta^2}}$ if the physical distance between vertices i and j is less than or equal to κ, and zero otherwise. We fix $\theta = 0.9$ and $\kappa = 0.5$ in our experiments, and ensure that the graph is connected.

1) Polynomial Generating Dictionary: In our first set of experiments, to construct a set of synthetic training signals consisting of localized patterns on the graph, we use a generating dictionary that is a concatenation of $S = 4$ subdictionaries that comply with the constraints of our dictionary learning algorithm. Each subdictionary is a fifth order ($K = 5$) polynomial of the graph Laplacian according to (5) and captures one of the four constitutive components of our signal class. The generating kernels $\{\widehat{g}_s(\cdot)\}_{s=1,2,\ldots,S}$ of the dictionary are shown in Fig. 2(a). We generate the graph signals by linearly combining $T_0 \leq 4$ random atoms from the dictionary with random coefficients. We then learn a dictionary from the training signals, and we expect this learned dictionary to be close to the known generating dictionary.

We first study the influence of the size of the training set on the dictionary learning outcome. Collecting a large number of training signals can be infeasible in many applications. Moreover, training a dictionary with a large training set significantly increases the complexity of the learning phase, leading to intractable optimization problems. Using our polynomial dictionary learning algorithm with training sets of $M = \{400, 600, 2000\}$ signals, we learn a dictionary of $S = 4$ subdictionaries. To allow some flexibility into our learning algorithm, we fix the degree of the learned polynomials to

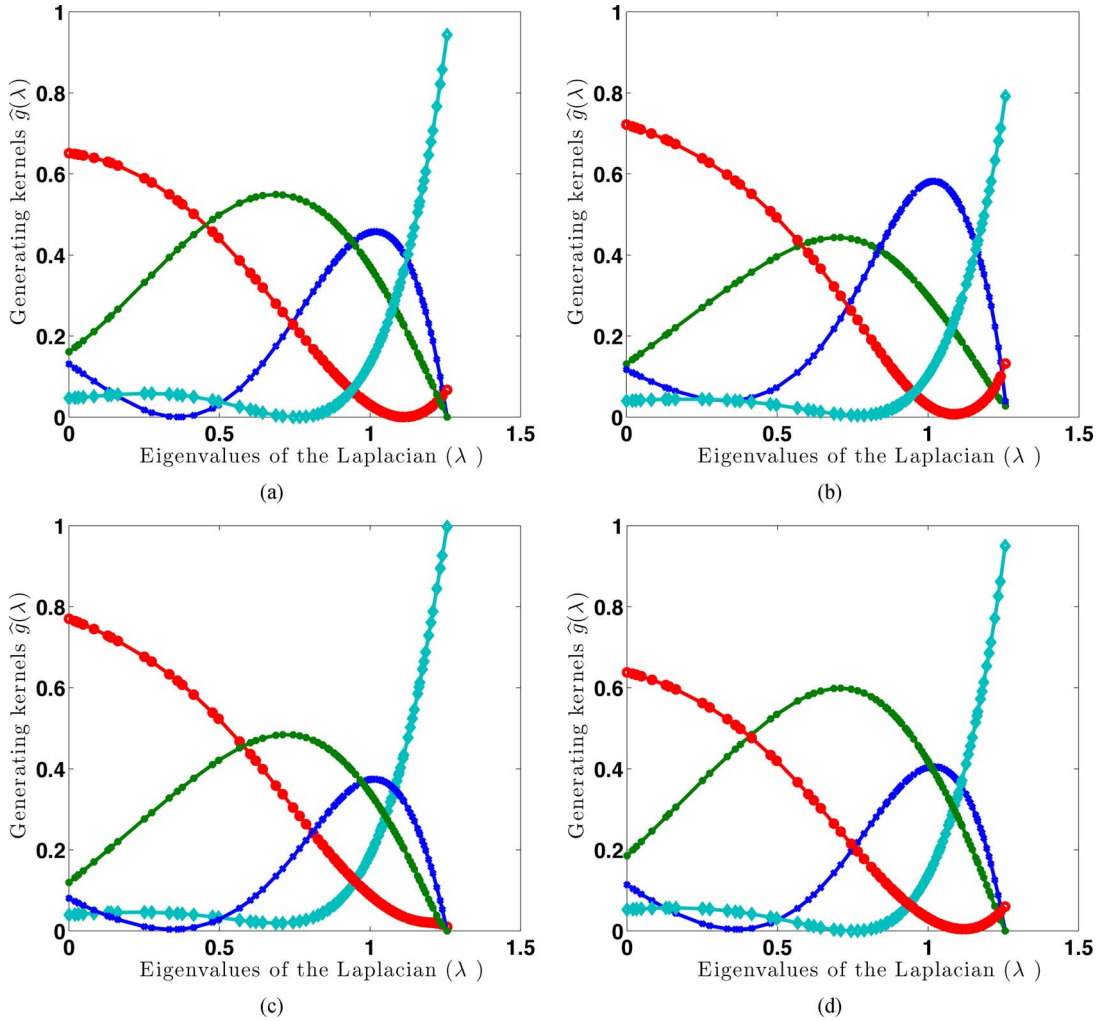

Fig. 2. Comparison of the kernels learned by the polynomial dictionary learning algorithm to the generating kernels $\{\widehat{g}_s(\cdot)\}_{s=1,2,\ldots,S}$ (shown in (a)) for $M = 400$, $M = 600$ and $M = 2000$ training signals. (a) Kernels of the generating dictionary. (b) Learned kernels with $M = 400$. (c) Learned kernels with $M = 600$. (d) Learned kernels with $M = 2000$.

$K = 20$. Comparing Fig. 2(a) to Figs. 2(b)–(d), we observe that our algorithm is able to recover the shape of the kernels used for the generating dictionary, even with a very small number of training signals. However, the accuracy of the recovery improves as we increase the size of the training set. To quantify the improvement, we define the mean SNR of the learned kernels as $\frac{1}{S}\sum_{s=1}^{S} -20\log(\|\widehat{g}_s(\Lambda) - \widehat{g}_s{}'(\Lambda)\|_2)$, where $\widehat{g}_s(\Lambda)$ is the true pattern of Fig. 2(a) for the subdictionary $\mathcal{D}_s$ and $\widehat{g}_s{}'(\Lambda)$ is the corresponding pattern learned with our learning algorithm. The SNR values that we obtain are $\{4.9, 5.3, 14.9\}$ for $M = \{400, 600, 2000\}$, respectively.

Next, we generate 2000 testing signals using the same method as for the construction of the training signals. We then study the effect of the size of the training set on the approximation of the testing signals with atoms from our learned dictionary. Fig. 3 illustrates the results for three different sizes of the training set and compares the approximation performance to that of other learning algorithms. Each point in the figure is the average of 20 random runs with different realizations of the training and testing sets. We first observe that the approximation performance of the polynomial dictionary is always better than that of SGWT, which demonstrates the benefits of the learning process. The improvement is attributed to the fact that the SGWT kernels are designed *a priori*, while our algorithm learns the shape of the kernels from the data.

We also see that the performance of K-SVD depends on the size of the training set. Recall that K-SVD is blind to the graph structure, and is therefore unable to capture translations of similar patterns. In particular, we observe that when the size of the training set is relatively small, as in the case of $M = \{400, 600\}$, the approximation performance of K-SVD significantly deteriorates. It improves when the number of training signals increases (i.e., $M = 2000$). Our polynomial dictionary however shows much more stable performance with respect to the size of the training set. We note three reasons that may explain the better performance of our algorithm, as compared to K-SVD. First, we recall that the number of unknowns parameters for K-SVD is $N^2S = 40000$, while for the polynomial dictionary this number is reduced to $(K+1)S = 84$. Thus, due to the lack of structure, the number of training signals needed for K-SVD usually grows linearly with the size of the dictionary, and is greater than the number needed to effectively train the polynomial dictionary. This fact explains the improved performance of K-SVD with $M = 2000$

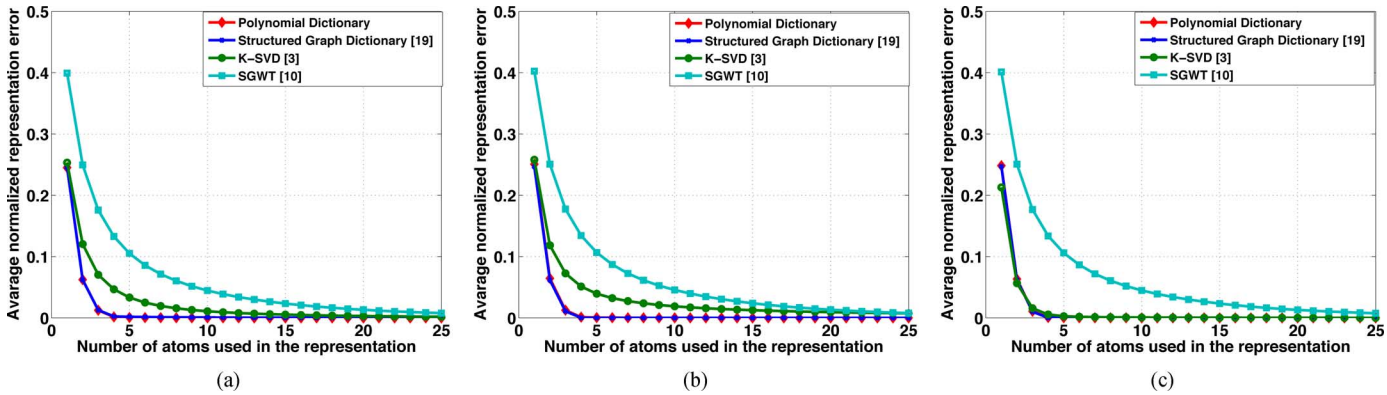

Fig. 3. Comparison of the learned polynomial dictionary to the SGWT[10], K-SVD [3] and the graph structured dictionary [22] in terms of approximation performance on test data generated from a polynomial generating dictionary, for different sizes of the training set. (a) $\mathrm{M} = 400$. (b) $\mathrm{M} = 600$. (c) $\mathrm{M} = 2000$.

training signals. Second, due to the limited size of the training set, K-SVD tends to learn atoms that sparsely approximate the signal on the whole graph, rather than to extract common features that appear in different neighborhoods. As a result, the atoms learned by K-SVD tend to have a global support on the graph, and K-SVD shows poor performance in the datasets containing many localized signals. Third, even when K-SVD does learn a localized pattern appearing in the training data, it does not take into account that similar patterns may appear at other areas of the graph. Of course, as we increase the number of training signals, translated instances of the pattern are more likely to appear in other areas of the graph in the training data, and K-SVD is then more likely to learn atoms containing such patterns in different areas of the graph. On the other hand, our polynomial dictionary learning algorithm learns the patterns in the graph spectral domain, and then includes translated versions of the patterns to all locations in the graph in the learned dictionary, even if some specific instances of the translated patterns do not appear in the training set. Thus, for a smaller number of training examples, our polynomial dictionary shows significantly better performance with respect to K-SVD due to reduced overfitting.

The algorithm proposed in [22] represents some sort of intermediate solution between K-SVD and our algorithm. It learns a dictionary that consists of subdictionaries of the form $\chi\widehat{g_s}(\Lambda)\chi^T$, where the specific values $\widehat{g_s}(\lambda_0), \widehat{g_s}(\lambda_1), \ldots, \widehat{g_s}(\lambda_{N-1})$ are learned, rather than learning the coefficients of a polynomial kernel $\widehat{g_s}(\cdot)$ and evaluating it at the N discrete eigenvalues as we do. Thus, the overall number of unknowns of this algorithm is NS, which is usually larger the number required by the polynomial dictionary ($(K+1)S$) and smaller than that of K-SVD (N^2S). The obtained dictionary is adapted to the graph structure and it contains atoms that are translated versions of the same pattern on the graph. However, the obtained atoms are not in general guaranteed to be well localized in the graph since the learned discrete values of $\widehat{g_s}$ are not necessarily derived from a smooth kernel. Moreover, the unstructured construction of the kernels in the method of [22] leads to more complex implementations, as discussed in Section VI.

2) Non-Polynomial Generating Dictionary: In the next set of experiments, we depart from the idealistic scenario and study the performance of our polynomial dictionary learning algorithm in the more general case when the signal components are not exactly polynomials of the Laplacian matrix. In order to generate training and testing signals, we divide the spectrum of the graph into four frequency bands, defined by the eigenvalues of the graph: $[\lambda_0 : \lambda_{24}]$, $[(\lambda_{25} : \lambda_{39}) \cup (\lambda_{90} : \lambda_{99})]$, $[\lambda_{40} : \lambda_{64}]$, and $[\lambda_{65} : \lambda_{89}]$. We then construct a generating dictionary of $J = 400$ atoms, with each atom having a spectral representation that is concentrated exclusively in one of the four bands. In particular, atom j is of the form

$$d_j = \widehat{h_j}(\mathcal{L})\delta_n = \chi\widehat{h_j}(\Lambda)\chi^T\delta_n. \tag{16}$$

Each atom is generated independently of the others as follows. We randomly pick one of the four bands, randomly generate 25 coefficients uniformly distributed in the range $[0, 1]$, and assign these random coefficients to be the diagonal entries of $\widehat{h_j}(\Lambda)$ corresponding to the indices of the chosen spectral band. The rest of the values in $\widehat{h_j}(\Lambda)$ are set to zero. The atom is then centered on a vertex n that is also chosen randomly. Note that the obtained atoms are not guaranteed to be well localized in the vertex domain since the discrete values of $\widehat{h_j}(\Lambda)$ are chosen randomly and are not derived from a smooth kernel. Therefore, the atoms of the generating dictionary do not exactly match the signal model assumed by our dictionary design algorithm, but rather are closer to the signal model assumed by [22]. Finally, we generate the training signals by linearly combining (with random coefficients) $T_0 \leq 4$ random atoms from the generating dictionary.

We first verify the ability of our dictionary learning algorithm to recover the spectral bands that are used in the synthetic generating dictionary. We fix the number of training signals to $M = 600$ and run our dictionary learning algorithm for three different degree values of the polynomial, i.e., $K = \{5, 10, 20\}$. The kernels $\{\widehat{g_s}(\cdot)\}_{s=1,2,3,4}$ obtained for the four subdictionaries are shown in Fig. 4 and the boundaries between the different frequency bands are indicated with the vertical dashed lines. We observe that for higher values of K, the learned kernels are more localized in the graph spectral domain and each kernel approximates one of the four bands defined in the generating dictionary, similarly to the behavior of classical frequency filters.

In Fig. 5, we illustrate the four learned atoms centered at the vertex $n = 1$ (one atom for each subdictionary), with $K = 20$. We can see that the support of the atoms adapts to the graph

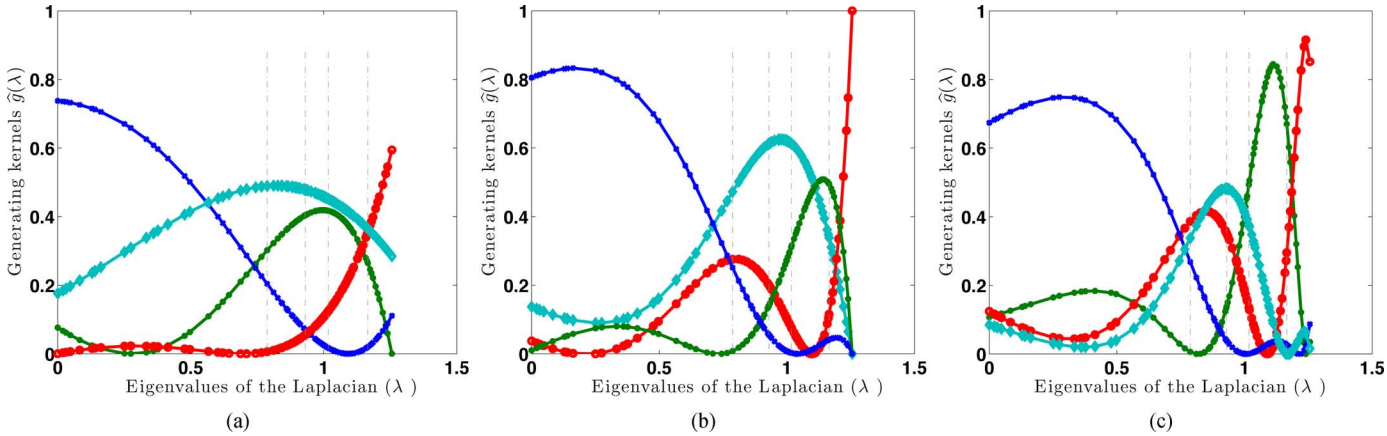

Fig. 4. Kernels $\{\widehat{g_s}(\cdot)\}_{s=1,2,3,4}$ learned by the polynomial dictionary algorithm for (a) $K = 5$, (b) $K = 10$, and (c) $K = 20$.

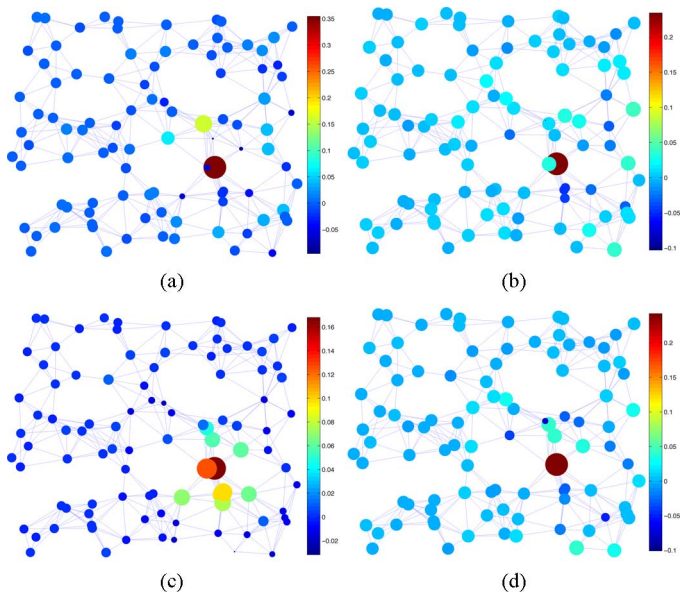

Fig. 5. Learned atoms centered on vertex $n = 1$, from each of the subdictionaries. (a) $\widehat{g_1}(\mathcal{L})\delta_1$. (b) $\widehat{g_2}(\mathcal{L})\delta_1$. (c) $\widehat{g_3}(\mathcal{L})\delta_1$. (d) $\widehat{g_4}(\mathcal{L})\delta_1$.

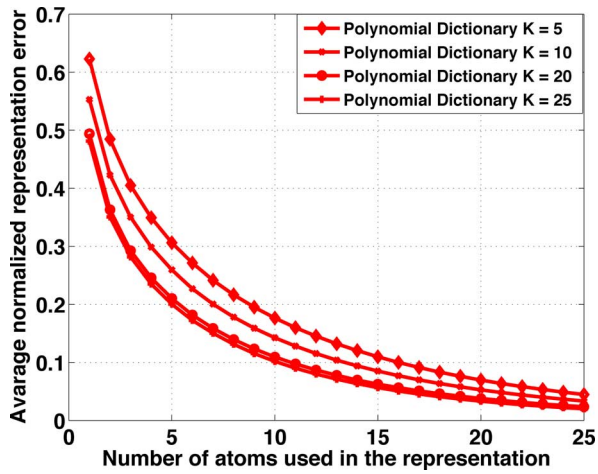

Fig. 6. Comparison of the average approximation performance of our learned dictionary on test signals generated by the non-polynomial synthetic generating dictionary, for $K = \{5, 10, 20, 25\}$.

topology. The atoms can be either smoother around a particular vertex, as for example in Fig. 5(c), or more localized, as in Fig. 5(a). Comparing Figs. 4 and 5, we observe that the localization of the atoms in the graph domain depends on the spectral behavior of the kernels. Note that the smoothest atom on the graph (Fig. 5(c)) corresponds to the subdictionary generated from the kernel that is concentrated on the low frequencies (i.e., $\widehat{g_3}(\cdot)$). This is because the graph Laplacian eigenvectors associated with the lower frequencies are smoother with respect to the underlying graph topology, while those associated with the larger eigenvalues oscillate more rapidly [1]. Apart from the polynomial degree, a second parameter that influences the support of the atoms on the graph is the sparsity level T_0 imposed in the leaning phase. A large T_0 implies that the learning algorithm has the flexibility to approximate the signals with many atoms. In the extreme case where T_0 is very large, the atoms of the dictionary tend to look like impulse functions. On the other hand, if T_0 is chosen to be small, the algorithm learns a dictionary that approximates the signals with only a few atoms. It implicitly guides the algorithm to learn atoms that are more spread on the graph, in order to cover it fully.

Next, we test the approximation performance of our learned dictionary on a set of 2000 testing signals generated in exactly the same way as the training signals, for four different degree values of the polynomial, i.e., $K = \{5, 10, 20\}$. Fig. 6 shows that the approximation performance obtained with our algorithm improves as we increase the polynomial degree. There are two main reasons for this improvement: (i) by increasing the polynomial degree, we allow more flexibility in the learning process; (ii) a small K implies that the atoms are localized in a small neighborhood and thus more atoms are needed to represent signals with support in different areas of the graph. However, we have empirically observed that, in practice, the improvement in the performance saturates as the value of K increases ($K = 20$ is usually enough to capture the frequency characteristics of the signals).

In Fig. 7, we fix $K = 20$, and compare the approximation performance of our learned dictionary to that of other dictionaries, with exactly the same setup as we used in Fig. 3. We again observe that K-SVD is the most sensitive to the size of the training data, and it clearly achieves the best performance when the size of the training set is large ($M = 2000$). Since the kernels used in the generating dictionary in this case do not match our polynomial model, the structured graph dictionary learning algorithm

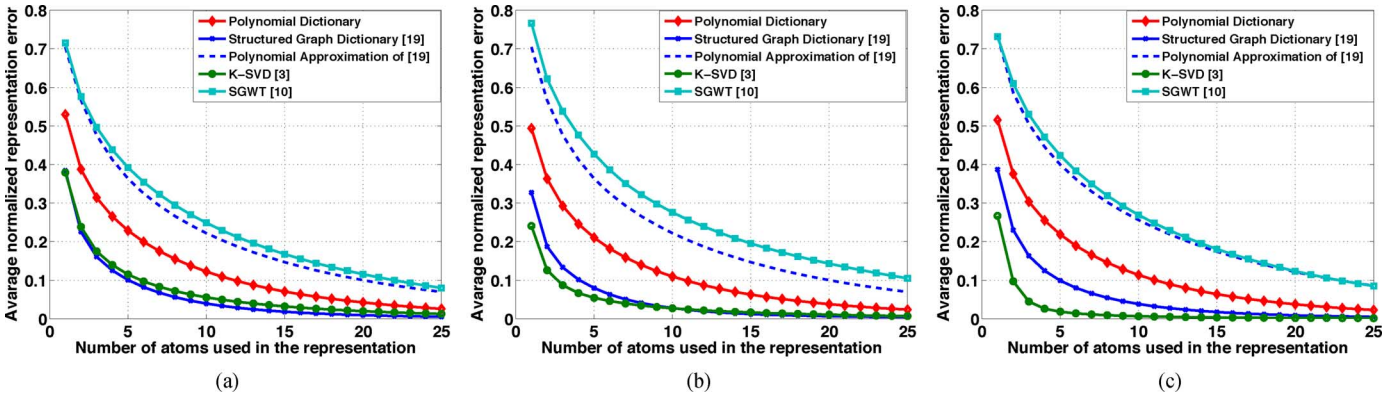

Fig. 7. Comparison of the learned polynomial dictionary to the SGWT[10], K-SVD [3] and the graph structured dictionary [22] in terms of approximation performance on test data generated from a non-polynomial generating dictionary, for different sizes of the training set. (a) $M = 400$. (b) $M = 600$. (c) $M = 2000$.

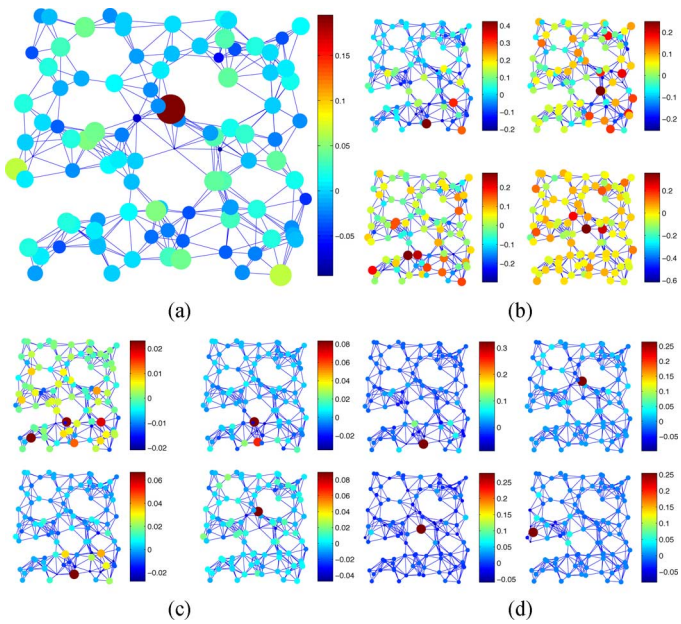

Fig. 8. (a) An example of a graph signal from the testing set and its atomic decomposition with respect to (b) the K-SVD dictionary, (c) the dictionary learned by [22] and (d) the learned polynomial graph dictionary.

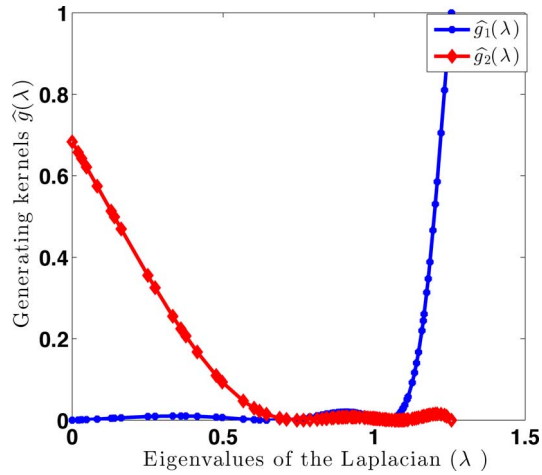

Fig. 9. Kernels $\{\widehat{g}_s(\cdot)\}_{s=1,2}$ learned by the polynomial dictionary algorithm from a set of training signals that are supported in only two particular bands of the spectrum: $[\lambda_0 : \lambda_9]$ and $[\lambda_{89} : \lambda_{99}]$, which correspond to the values $[0 : 0.275]$ and $[1.174 : 1.256]$ respectively.

of [22] has more flexibility to learn non-smooth generating kernels and therefore generally achieves better approximation. For a fairer comparison of approximation performance, we fit an order $K = 20$ polynomial function to the discrete values $\widehat{g}_s$ learned with the algorithm of [22]. We observe that our polynomial dictionary outperforms the polynomial approximation of the dictionary learned by [22] in terms of approximation performance. An example of the atomic decomposition of a graph testing signal with respect to the K-SVD dictionary, the structured graph dictionary of [22] and the polynomial graph dictionary is illustrated in Fig. 8. Note that the K-SVD atoms have a more global support in comparison to the other two graph dictionaries, while the polynomial dictionary atoms are the most localized in specific neighborhoods of the graph. Nonetheless, the approximation performance of our learned dictionary is competitive, especially for smaller training sets.

3) Generating Dictionary Focused on Specific Frequency Bands: In the final set of experiments, we study the behavior of our algorithm in the case when we have the additional prior information that the training signals do not cover the entire spectrum, but are concentrated only in some bands that are not known *a priori*. In order to generate the training signals, we choose only two particular frequency bands, defined by the eigenvalues of the graph: $[\lambda_0 : \lambda_9]$ and $[\lambda_{89} : \lambda_{99}]$, which correspond to the values $[0 : 0.275]$ and $[1.174 : 1.256]$, respectively. We construct a generating dictionary of $J = 400$ atoms, with each atom concentrated in only one of the two bands and generated according to (16). The training signals ($M = 600$) are then constructed by linearly combining $T_0 \leq 4$ atoms from the generating dictionary. We set $K = 20$ and $\epsilon_1 = c$ in order to allow our polynomial dictionary learning algorithm the flexibility to learn kernels that are supported only on specific frequency bands. The learned kernels are illustrated in Fig. 9. We observe that the algorithm is able to detect the spectral components that exist in the training signals since the learned kernels are concentrated only in the two parts of the spectrum to which the atoms of the generating dictionary belong.

B. Approximation of Real Graph Signals

After examining the behavior of the polynomial dictionary learning algorithm for synthetic signals, we illustrate the performance of our algorithm in the approximation of localized graph

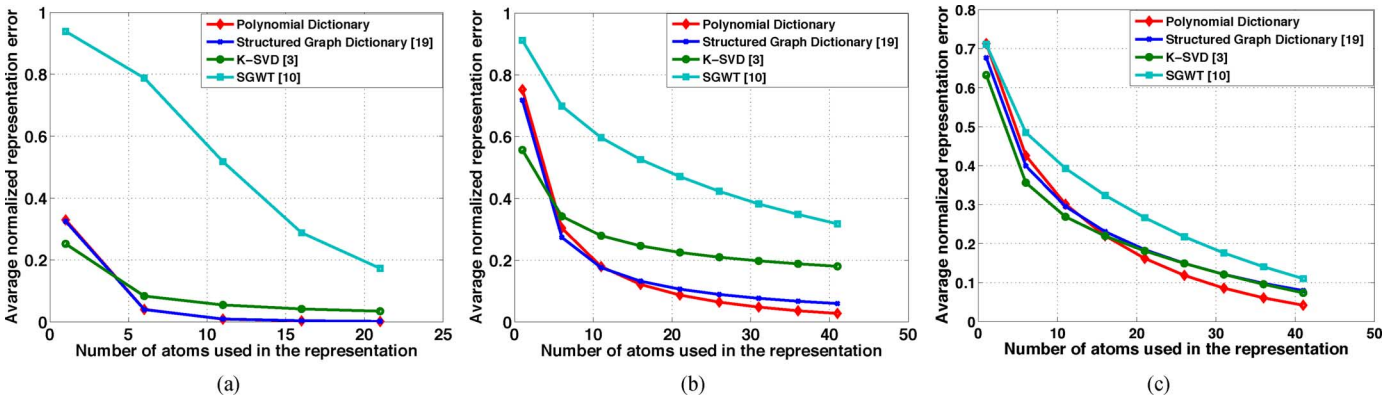

Fig. 10. Comparison of the learned polynomial dictionaries to the SGWT, K-SVD, and graph structured dictionaries [22] in terms of approximation performance on testing data generated from the (a) Flickr, (b) traffic, and (c) brain datasets, for $T_0 = 6$.

signals from real world datasets. In particular, we examine the following three datasets.

1) Flickr Dataset: We consider the daily number of distinct Flickr users that took photos at different geographical locations around Trafalgar Square in London, between January 2010 and June 2012 [33]. Each vertex of the graph represents a geographical area of 10×10 meters and it corresponds to the centroid of the area. We measure the pairwise distance between the nodes and we set the cutoff distance of the graph to 30 meters. We assign an edge between two locations when the distance between them is smaller than the cutoff distance, and we set the edge weight to be inversely proportional to the distance. By following this procedure, we obtain a sparse graph. The number of vertices of the graph is $N = 245$. The signal on the graph is the total number of distinct Flickr users that have taken photos at each location during a specific day. We have a total of 913 signals, and we use 700 of them for training and the rest for testing. We set $S = 2$ and $K = 10$ in our learning algorithm.

2) Traffic Dataset: We consider the daily bottlenecks in Alameda County in California between January 2007 and May 2013. The data are part of the Caltrans Performance Measurement System (PeMS) dataset that provides traffic information throughout all major metropolitan areas of California [34].[1] In particular, the nodes of the graph consist of $N = 439$ detector stations where bottlenecks were identified over the period under consideration. The graph is designed by connecting stations when the distance between them is smaller than a threshold of $\theta = 0.08$, which corresponds to approximately 13 kilometers. The distance is set to be the Euclidean distance of the GPS coordinates of the stations and the edge weights are set to be inversely proportional to the distance. A bottleneck could be any location where there is a persistent drop in speed, such as merges, large on-ramps, and incidents. The signal on the graph is the average length of the time in minutes that a bottleneck is active for each specific day. In our experiments, we fix the maximum degree of the polynomial to $K = 10$ and we learn a dictionary consisting of $S = 2$ subdictionaries. We use the signals in the period between January 2007 and December 2010 for training and the rest for testing. For computational issues, we normalize all the signals with respect to the norm of the signal with maximum energy.

3) Brain Dataset: We consider a set of fMRI signals acquired on five different subjects [35], [36]. For each subject, the signals have been preprocessed into timecourses of $N = 88$ brain regions of contiguous voxels, which are determined from a fixed anatomical atlas, as described in [36]. The timecourses for each subject correspond to 1290 different graph signals that are measured while the subject is in two different states, either completely relaxing, in the absence of any stimulation, or passively watching small movie excerpts. For the purpose of this paper, we treat the measurements at each time as an independent signal on the 88 vertices of the brain graph. The anatomical distances between regions of the brain are approximated by the Euclidean distance between the coordinates of the centroids of each region, the connectivity of the graph is determined by assigning an edge between two regions when the anatomical distance between them is shorter than 40 millimeters, and the edge weight is set to be inversely proportional to the distance. We then apply our polynomial dictionary learning algorithm in order to learn a dictionary of atoms representing brain activity across the network at a fixed point in time. We use the graph signals from the timecourses of two subjects as our training signals and we learn a dictionary of $S = 2$ subdictionaries and a maximum polynomial degree of $K = 15$. We use the graph signals from the remaining three timecourses to validate the performance of the learned dictionary. As in the previous dataset, we normalize all of the graph signals with respect to the norm of the signal with maximum energy. A summary of the main parameters of the three datasets is shown in Table I.

TABLE I
PARAMETERS OF REAL GRAPH SIGNALS

Dataset	N	S	K	Training set	Testing set
Flickr	245	2	10	700	213
Traffic	439	2	10	1459	882
Brain	88	2	15	2580	3870

Fig. 10 shows the approximation performance of the learned polynomial dictionaries for the three different datasets, for a sparsity constraints in the learning phase of $T_0 = 6$. The behavior is similar in all three datasets, and also similar to the results on the synthetic datasets in the previous section. In particular, the data-adapted dictionaries clearly outperform the SGWT dictionary in terms of approximation error on test signals, and the localized atoms of the learned polynomial dictionary effectively represent the real graph signals. It can even achieve better performance than K-SVD when sparsity

[1]The data are publicly available at http://pems.dot.ca.gov.

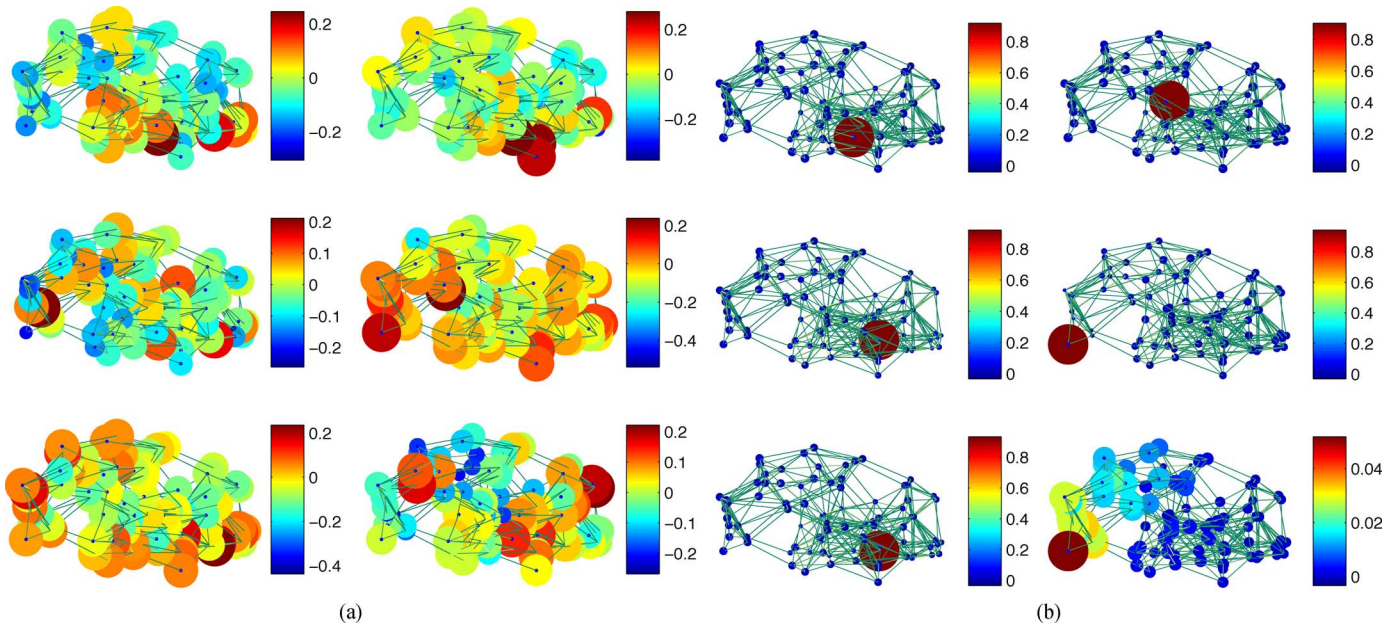

Fig. 11. Examples of atoms learned from training data from the brain dataset with (a) K-SVD and (b) the polynomial dictionary. The six atoms illustrated here are the ones that were most commonly included by OMP in the sparse decomposition of the testing signals.

increases. In particular, we observe that K-SVD outperforms both graph structured algorithms for a small sparsity level as it learns atoms that can smoothly approximate the whole signal. Comparing our algorithm with the one of [22], we observe that the performance of the latter is comparable. Apart from the differences between the two algorithms that we have already discussed in the previous subsections, one drawback of [22] is the way the dictionary is updated. Specifically, the update of the dictionary is performed block by block, which leads to a local optimum in the dictionary update step. This can lead to worse performance when compared to our algorithm, where all subdictionaries are updated simultaneously.

In Fig. 11, we illustrate the six most used atoms after applying OMP for the sparse decomposition of the testing signals from the brain dataset in the learned K-SVD dictionary and our learned polynomial dictionary. Note that in Fig. 11(b), the polynomial dictionary consists of localized atoms with support concentrated on small neighborhoods of vertices. These atoms capture the activation of particular regions of the brain. Interestingly, we observe that one of the most frequently chosen atoms is the one capturing the visual cortex, which is found in the back of the brain (second figure in the third row). The result of the sparse coding in this case is consistent with the pattern that we expect to appear in the brain, as the visual cortex is activated during visual stimuli. The price to pay for the interpretability and the localization of the atoms, is the poor approximation performance at low sparsity levels. However, as the sparsity tolerance increases, the localization property clearly becomes beneficial. Detecting the activated patterns in the brain using our polynomial dictionary is a very promising research direction.

C. Illustrative Application: Image Segmentation

As an illustrative application of the proposed dictionary, we provide some results in image segmentation. We emphasize that the particular application is provided just to illustrate the use of structured graph dictionaries in different signal processing tasks. We take the 128×128 house and 128×129 cameraman images and from each of them we extract overlapping block patches of size 5×5 pixels, covering all the pixels of the original image. Each patch is centered in one pixel and, for the sake of simplicity, we ignore the pixels on the boundary that do not have both horizontal and vertical neighbors. For each of the two images, the training signals are constructed as a collection of 15376 and 15625 such patches respectively. We fix the number of subdictionaries to $S = 4$, the polynomial degree to $K = 15$ and the sparsity level to $T_0 = 4$. The graph for each patch is the binary graph defined by connecting each pixel to its horizontal and vertical neighbors. For each of the images, we apply our polynomial dictionary learning algorithm, training a dictionary of dimensionality 25×100. Since the number of training signals is large, we apply ADMM to solve the quadratic program in the learning phase.

In order to extract the features for the segmentation of the image, we compute the inner product of each patch with the atoms of the learned dictionary. If y_j is the patch corresponding to pixel j, then $\mathcal{D}_s^T y_j = \sum_{\ell=0}^{N-1} \widehat{y}_j(\lambda_\ell)\widehat{g}_s(\lambda_\ell)\chi_\ell$, which implies that we filter each patch with all four filters $\{\widehat{g}_s(\cdot)\}_{s=1,2,3,4}$ in order to modify its frequency characteristics. For each filtered version of the patch, we compute the mean and the variance. We define as feature for each patch a vector in $\mathbb{R}^{2S}$ that contains the mean and variance of each filtered versions. The features, and consequently the nodes, are then clustered in $S = 4$ clusters, using K-means. The obtained segmentations and some of the learned atoms are shown in Fig. 12. We observe that the segmentation results in both images are quite promising as the edges of the images are preserved most of the time. This is mainly due to the localization of the atoms. Further work and more extensive studies are required to deploy the proposed algorithm in image segmentation applications.

Fig. 12. Learned atoms (a) and segmentation results (d,e) obtained using the polynomial dictionary on the house (b) and cameraman (c) images.

VI. Computational Efficiency of the Learned Polynomial Dictionary

The structural properties of the proposed class of dictionaries lead to compact representations and computationally efficient implementations, which we elaborate on briefly in this section. First, the number of free parameters depends on the number S of subdictionaries and the degree K of the polynomials. The total number of parameters is $(K+1)S$, and since K and S are small in practice, the dictionary is compact and easy to store. Second, contrary to the unstructured dictionaries learned by algorithms such as K-SVD and MOD, the dictionary forward and adjoint operators can be efficiently applied when the graph is sparse, as is usually the case in practice. Recall from (5) that $\mathcal{D}^T y = \sum_{s=1}^{S}\sum_{k=0}^{K} \alpha_{sk}\mathcal{L}^k y$. The computational cost of the iterative sparse matrix-vector multiplication required to compute $\{\mathcal{L}^k y\}_{k=0,2,\ldots,K}$ is $O(K|\mathcal{E}|)$, where $|\mathcal{E}|$ is the cardinality of the edge set of the graph. Therefore, the total computational cost to compute $\mathcal{D}^T y$ is $O(K|\mathcal{E}|+NSK)$. We further note that, by following a procedure similar to the one in [10, Section 6.1], the term $\mathcal{D}\mathcal{D}^T y$ can also be computed in a fast way by exploiting the fact that $\mathcal{D}\mathcal{D}^T y = \sum_{s=1}^{S} \widehat{g_s}^2(\mathcal{L})y$. This leads to a polynomial of degree $K' = 2K$ that can be efficiently computed. Both operators $\mathcal{D}^T y$ and $\mathcal{D}\mathcal{D}^T y$ are important components of most sparse coding techniques. In turn, these efficient implementations are therefore useful in numerous signal processing tasks, and comprise one of the main advantages of learning structured parametric dictionaries. For example, to find sparse representations of different signals with the learned dictionary, rather than using OMP, we can use iterative soft thresholding [37] to solve the lasso regularization problem [38]. The two main operations required in iterative soft thresholding, $\mathcal{D}^T y$ and $\mathcal{D}^T\mathcal{D}x$, can both be approximated by the Chebyshev approximation method of [10], as explained in more detail in [39, Section IV.C]. The same procedure could be applied to compute efficiently the forward and adjoint operators of the dictionary learned in [22]. In that case however, we need to first approximate the discrete values of the kernel with a polynomial function, which as shown in Fig. 7, can deteriorate the approximation performance.

Another benefit is that in settings where the data is distributed and communication between nodes of the graph is costly (e.g., a sensor network), the polynomial structure of the learned dictionary enables quantities such as $\mathcal{D}^T y$, $\mathcal{D}x$, $\mathcal{D}\mathcal{D}^T y$, and $\mathcal{D}^T\mathcal{D}x$ to be efficiently computed in a distributed fashion using the techniques of [39]. We consider, as an illustration, the distributed processing scenario where each node n of the graph knows only its own component of a signal $y \in \mathbb{R}^N$ and the n^{th} row of the corresponding weight matrix $\mathcal{L}$. The polynomial coefficients used in the dictionary are further known to the nodes all over the network. Each node n can communicate only with its neighbors and after some simple computations, it can compute the components $\{(\mathcal{D}^T y)_{(s-1)N+n}\}_{s=1,\ldots,S}$. The basic steps of this operation are shown in Algorithm 2. As discussed in [39], the concise representation of the dictionary in terms of the polynomial coefficients makes it possible to implement many signal processing algorithms in a distributed fashion.

Algorithm 2: Distributed computation of $\mathcal{D}^T y$

Inputs at node n**:** $y(n), \mathcal{L}_{n,:}, \alpha = [\alpha_1; \ldots; \alpha_S]$
Output at node n**:** $\{(\mathcal{D}^T y)_{(s-1)N+n}\}_{s=1,\ldots,S}$
Transmit $y(n)$ to all neighbors $\mathcal{N}_n$
Receive $y(m)$ from neighbors $\mathcal{N}_n$
Compute and store $c_n^1 = (\mathcal{L}^T y)_n$.
for $k = 2, \ldots, K$ **do:**
    Transmit $c_n^{l-1} = (\mathcal{L}^T c^{l-2})_n$ to all the neighbors
    Receive c_m^{l-1} from all the neighbors $m \in \mathcal{N}_n$.
end for
for $s = 1, \ldots, S$ **do**
    Compute $(\mathcal{D}^T y)_{(s-1)N+n} = \alpha_{0s} y(n) + \sum_{k=1}^{K} \alpha_{ks} c_n^l$
end for

VII. Conclusion

We proposed a parametric family of structured dictionaries – namely, unions of polynomial matrix functions of the graph Laplacian – to sparsely represent signals on a given weighted graph, and an algorithm to learn the parameters of a dictionary belonging to this family from a set of training signals on the graph. When translated to a specific vertex, the learned polynomial kernels in the graph spectral domain correspond to localized patterns on the graph. Translating each of these patterns to different areas of the graph led to sparse approximation performance that was clearly better than that of non-adapted graph wavelet dictionaries such as the SGWT, and comparable to or better than that of dictionaries learned by state-of-the art numerical algorithms such as K-SVD. The approximation performance of our learned dictionaries was also more robust to the size of training data. At the same time, because our learned dictionaries are unions of polynomial matrix functions of the graph

Laplacian, they can be efficiently stored and implemented in both centralized and distributed signal processing tasks.

Although we have provided some preliminary results in signal approximation, the potential of the proposed dictionary structure is yet to be explored in other applications. Additional work is required to apply the polynomial structure in other graph signal processing and data analysis tasks such as classification, clustering, community detection, or source localization where we expect that the localization properties of the dictionary can be beneficial.

ACKNOWLEDGMENT

The authors would like to thank Prof. D. Van De Ville for providing the brain data, Prof. A. Ortega, Prof. P. Vandergheynst, and X. Dong for their useful feedbacks about this work, G. Stathopoulos for helpful discussions about the optimization problem, and the anonymous reviewers for their constructive comments.

REFERENCES

[1] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, May 2013.

[2] R. Rubinstein, A. M. Bruckstein, and M. Elad, "Dictionaries for sparse representation modeling," *Proc. IEEE*, vol. 98, no. 6, pp. 1045–1057, Apr. 2010.

[3] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.

[4] K. Engan, S. O. Aase, and J. H. Husoy, "Method of optimal directions for frame design," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Phoenix, AZ, USA, Mar. 1999, vol. 5, pp. 2443–2446.

[5] S. Mallat, *A Wavelet Tour of Signal Processing: The Sparse Way*, 3rd ed. New York, NY, USA: Academic, 2008.

[6] P. Jost, P. Vandergheynst, S. Lesage, and R. Gribonval, "Motif: An efficient algorithm for learning translation invariant dictionaries," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Toulouse, France, May 2006, vol. 5, pp. 857–860.

[7] M. Yaghoobi, L. Daudet, and M. E. Davies, "Parametric dictionary design for sparse coding," *IEEE Trans. Signal Process.*, vol. 57, no. 12, pp. 4800–4810, Dec. 2009.

[8] R. Rubinstein, M. Zibulevsky, and M. Elad, "Double sparsity: Learning sparse dictionaries for sparse signal approximation," *IEEE Trans. Signal Process.*, vol. 58, no. 3, pp. 1553–1564, Mar. 2010.

[9] D. Thanou, D. I Shuman, and P. Frossard, "Parametric dictionary learning for graph signals," presented at the IEEE Glob. Conf. Signal Inf. Process., Austin, TX, USA, Dec. 2013.

[10] D. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Appl. Comput. Harmon. Anal.*, vol. 30, no. 2, pp. 129–150, Mar. 2010.

[11] X. Zhu and M. Rabbat, "Approximating signals supported on graphs," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Kyoto, Japan, Mar. 2012, pp. 3921–3924.

[12] R. R. Coifman and M. Maggioni, "Diffusion wavelets," *Appl. Comput. Harmon. Anal.*, vol. 21, pp. 53–94, Mar. 2006.

[13] S. K. Narang and A. Ortega, "Perfect reconstruction two-channel wavelet filter banks for graph structured data," *IEEE Trans. Signal Process.*, vol. 60, no. 6, pp. 2786–2799, Jun. 2012.

[14] M. Gavish, B. Nadler, and R. R. Coifman, "Multiscale wavelets on trees, graphs and high dimensional data: Theory and applications to semi supervised learning," presented at the Int. Conf. Mach. Learn., Haifa, Israel, Jun. 2010.

[15] I. Ram, M. Elad, and I. Cohen, "Generalized tree-based wavelet transform," *IEEE Trans. Signal Process.*, vol. 59, no. 9, pp. 4199–4209, Sep. 2011.

[16] I. Ram, M. Elad, and I. Cohen, "Redundant wavelets on graphs and high dimensional data clouds," *IEEE Signal Process. Lett.*, vol. 19, no. 5, pp. 291–294, May 2012.

[17] D. I. Shuman, B. Ricaud, and P. Vandergheynst, "A windowed graph Fourier transform," presented at the IEEE Statist. Signal Process. Workshop, Ann Arbor, MI, USA, Aug. 2012.

[18] D. I. Shuman, B. Ricaud, and P. Vandergheynst, "Vertex-frequency analysis on graphs," *Appl. Comput. Harmon. Anal.*, Jul. 2013, submitted for publication.

[19] D. I. Shuman, C. Wiesmeyr, N. Holighaus, and P. Vandergheynst, "Spectrum-adapted tight graph wavelet and vertex-frequency frames," *IEEE Trans. Signal Process.*, Nov. 2013, submitted for publication.

[20] J. C. Bremer, R. R. Coifman, M. Maggioni, and A. D. Szlam, "Diffusion wavelet packets," *Appl. Comput. Harmon. Anal.*, vol. 21, no. 1, pp. 95–112, Jun. 2006.

[21] R. M. Rustamov and L. Guibas, "Wavelets on graphs via deep learning," in *Adv. Neural Inf. Process. Syst. (NIPS)*, Dec. 2013.

[22] X. Zhang, X. Dong, and P. Frossard, "Learning of structured graph dictionaries," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Kyoto, Japan, Mar. 2012, pp. 3373–3376.

[23] N. J. Higham, *Functions of Matrices*. Philadelphia, PA, USA: SIAM, 2008.

[24] M. Zheng, J. Bu, C. Chen, C. Wang, L. Zhang, G. Qiu, and D. Cai, "Graph regularized sparse coding for image representation," *IEEE Trans. Image Process.*, vol. 20, no. 5, pp. 1327–1336, May 2011.

[25] F. Chung, *Spectral Graph Theory*. Providence, RI, USA: Amer, Math. Soc., 1997.

[26] J. A. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Trans. Inf. Theory*, vol. 50, no. 10, pp. 2231–2242, Oct. 2004.

[27] M. Bruckstein, D. Donoho, and M. Elad, "From sparse solutions of systems of equations to sparse modeling of signals and images," *SIAM Rev.*, vol. 51, no. 1, pp. 34–81, Feb. 2009.

[28] M. Elad, *Sparse and Redundant Representations – From Theory to Applications in Signal and Image Processing*. New York, NY, USA: Springer, 2010.

[29] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge Univ. Press, 2004.

[30] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.

[31] K. C. Toh, M. J. Todd, and R. H. Tutuncu, "SDPT3—A MATLAB software package for semidefinite programming," in *Optim. Methods Softw.*, 1999, vol. 11, pp. 545–581.

[32] J. Lofberg, "YALMIP: A toolbox for modeling and optimization in MATLAB," presented at the CACSD Conf., Taipei, Taiwan, Sep. 2004.

[33] X. Dong, A. Ortega, P. Frossard, and P. Vandergheynst, "Inference of mobility patterns via spectral graph wavelets," presented at the Proc. IEEE Int. Conf. Acoust., Speech, Signal Process., Vancouver, Canada, May 2013.

[34] T. Choe, A. Skabardonis, and P. P. Varaiya, "Freeway performance measurement system (PeMS): An operational analysis tool," presented at the Annu. Meet. of Transportat. Res. Board, Washington, DC, USA, Jan. 2002.

[35] H. Eryilmaz, D. Van De Ville, S. Schwartz, and P. Vuilleumier, "Impact of transient emotions on functional connectivity during subsequent resting state: A wavelet correlation approach," *NeuroImage*, vol. 54, no. 3, pp. 2481–2491, Feb. 2011.

[36] J. Richiardi, H. Eryilmaz, S. Schwartz, P. Vuilleumier, and D. Van De Ville, "Decoding brain states from fMRI connectivity graphs," *NeuroImage*, vol. 56, no. 2, pp. 616–626, May 2011.

[37] I. Daubechies, M. Defrise, and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Commun. Pure Appl. Math.*, vol. 57, no. 11, pp. 1413–1457, Nov. 2004.

[38] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. R. Statist. Soc. Ser. B*, vol. 58, pp. 267–288, 1994.

[39] D. I Shuman, P. Vandergheynst, and P. Frossard, "Chebyshev polynomial approximation for distributed signal processing," presented at the Int. Conf. Distrib. Comput. Sensor Sys., Barcelona, Spain, Jun. 2011.

Dorina Thanou, photograph and biography not available at the time of publication.

David I Shuman, photograph and biography not available at the time of publication.

Pascal Frossard, photograph and biography not available at the time of publication.