Parametric Dictionary Learning for Graph Signals

Dorina Thanou, David I Shuman, and Pascal Frossard
Signal Processing Laboratory (LTS4)
Ecole Polytechnique Fédérale de Lausanne (EPFL)
CH-1015 Lausanne, Switzerland
Email:{dorina.thanou, david.shuman, pascal.frossard}@epfl.ch

***Abstract*—We propose a parametric dictionary learning algorithm to design structured dictionaries that sparsely represent graph signals. We incorporate the graph structure by forcing the learned dictionaries to be concatenations of subdictionaries that are polynomials of the graph Laplacian matrix. The resulting atoms capture the main spatial and spectral components of the graph signals of interest, leading to adaptive representations with efficient implementations. Experimental results demonstrate the effectiveness of the proposed algorithm for the sparse approximation of graph signals.**

I. Introduction

Graphs are flexible data representation tools, suitable for modeling the geometric structure of topologically complicated domains on which signals reside. Examples of such signals 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. A simple example of a graph signal is the current temperature at each location in a sensor network.

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, many numerical algorithms such as K-SVD and the Method of Optimal Directions (MOD) (see [2, Section IV] and references therein) have been constructed to 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, and curvelet transforms are based on mathematical models of signal classes. 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.

Returning specifically to the graph setting, numerical approaches such as K-SVD and MOD can certainly be applied to graph signals, which can be 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, a number of transform-based dictionaries for graph signals have recently been proposed. For example, the graph Fourier transform has been shown to sparsely represent smooth graph signals, wavelet transforms such as diffusion wavelets and spectral graph wavelets target piecewise-smooth graph signals, and the windowed graph Fourier transform can be used to analyze signal content at specific vertex and frequency locations (see [1] for an overview and complete list of references). 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 (one exception are the diffusion wavelet packets of [3], which feature extra adaptivity). The recent work in [4] tries to bridge the gap between the graph-based transform methods and the purely numerical dictionary learning algorithms. The learned dictionary in [4] has a structure derived from the graph topology, while its parameters are learned from the data. However, it does not necessarily lead to efficient implementations.

In this work, we capitalize on the benefits of both numerical and analytical approaches by learning a dictionary that incorporates the graph structure and can be implemented efficiently. We assume that the graph signals consist of local patterns, describing localized events on the graph. 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 [5]. 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].

II. Basic Definitions on Graphs

We consider a weighted and undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$ where $\mathcal{V}$, $\mathcal{E}$ represent the vertex and edge set of the graph and W the weights corresponding to each edge. We assume that the graph is connected. The graph Laplacian operator is defined as $L = D - W$, where D is the diagonal degree matrix. Its normalized form is defined as $\mathcal{L} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$. Both operators are real symmetric matrices and they have a complete set of orthonormal eigenvectors with corresponding non-negative eigenvalues. We denote the eigenvectors of the normalized Laplacian by $\chi = [\chi_1, \chi_2, ..., \chi_N]$ and its eigenvalues by $0 = \lambda_0 < \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_{N-1} = 2$.

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 $v \in \mathcal{V}$. In the graph setting, as is often the case in classical settings, the spectral domain representation can provide significant information about the characteristics of signals. It

has been shown that 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]. Throughout the paper, we use the normalized Laplacian eigenvectors as the Fourier basis in order to avoid some computational issues that arise when taking large powers of the unnormalized graph Laplacian. In particular, for any function y defined on the vertices of the graph, the graph Fourier transform $\hat{y}$ at frequency λ_ℓ is defined as

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

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 delta function centered at vertex n [1]:

$$T_n g = \sqrt{N}(g * \delta_n) = \sqrt{N} \sum_{\ell=0}^{N-1} \hat{g}(\lambda_\ell)\chi_\ell^*(n)\chi_\ell. \tag{1}$$

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, and the localization of $T_n g$ around the center vertex n is controlled by the smoothness of the kernel $\hat{g}(\cdot)$. 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:

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

In particular, the k^{th} power of the Laplacian $\mathcal{L}$ is exactly k-hop localized on the graph topology [5]. Combining (1) and (2), translating a polynomial kernel to each vertex in the graph yields a set of N localized atoms, which are the columns of

$$Tg = \sqrt{N}\hat{g}(\mathcal{L}) = \sqrt{N}\chi\hat{g}(\Lambda)\chi^T = \sqrt{N}\sum_{k=0}^{K} \alpha_k \mathcal{L}^k, \tag{3}$$

where Λ is the diagonal matrix of the eigenvalues.

III. 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 similar graph signals. In this section, we describe the dictionary structure and learning algorithm, as well as computational issues related to the use of the learned dictionary.

A. Dictionary Structure

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

$$\mathcal{D}_s = \widehat{g_s}(\mathcal{L}) = \chi \left(\sum_{k=0}^{K} \alpha_{sk} \Lambda^k \right) \chi^T = \sum_{k=0}^{K} \alpha_{sk} \mathcal{L}^k. \tag{4}$$

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 order K polynomial $\widehat{g_s}(\cdot)$ translated to vertex n. The polynomial structure of the kernel $\widehat{g_s}(\cdot)$ 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.

We also impose two constraints on the kernels $\{\widehat{g_s}(\cdot)\}_{s=1,2,...,S}$. First, we require that the kernels are non-negative and uniformly bounded by a given constant c; i.e., $0 \leq \widehat{g_s}(\lambda) \leq c$ for all $\lambda \in [0, \lambda_{\max}]$, or, equivalently,

$$0 \preceq \mathcal{D}_s \preceq cI, \quad \forall s \in \{1, 2, ..., S\}, \tag{5}$$

where I is the $N \times N$ identity matrix; i.e., each subdictionary $\mathcal{D}_s$ is a positive semi-definite matrix whose maximum eigenvalue is upper bounded by c.

Second, since the training signals usually contain frequency components that are spread across the entire spectrum, the learned kernels $\{\widehat{g_s}(\cdot)\}_{s=1,2,...,S}$ should also cover the spectrum. One way to ensure this would be to impose a constraint that $\sum_{s=1}^{S} |\widehat{g_s}(\lambda)|^2$ is constant for all $\lambda \in [0, \lambda_{\max}]$, which, following a slight generalization of [5, Theorem 5.6], also guarantees that the resulting dictionary $\mathcal{D}$ is a tight frame. However, such a constraint would lead to a significantly more difficult optimization problem for the learning phase discussed in the next subsection. Therefore, we instead impose the constraint $c - \epsilon \leq \sum_{s=1}^{S} \widehat{g_s}(\lambda) \leq c + \epsilon$, for all $\lambda \in [0, \lambda_{\max}]$, or equivalently

$$(c-\epsilon)I \preceq \sum_{s=1}^{S} \mathcal{D}_s \preceq (c+\epsilon)I, \tag{6}$$

where ϵ is a small positive constant. To summarize, the dictionary $\mathcal{D}$ is a parametric dictionary that depends on the parameters $\{\alpha_{sk}\}_{s=1,2,...,S;\ k=1,2,...,K}$, and the constraints (5) and (6) can be viewed as constraints on these parameters.

B. Dictionary Learning Algorithm

Given a set of training signals $Y = [y_1, y_2, ..., y_M] \in \mathbb{R}^{N \times M}$, all living on the graph $\mathcal{G}$, we aim at learning a graph dictionary $\mathcal{D} \in \mathbb{R}^{N \times NS}$ with the structure described in Section III-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 (4), this is equivalent to learning the parameters $\{\alpha_{sk}\}_{s=1,2,...,S;\ k=1,2,...,K}$ that characterize the set of generating kernels, $\{\widehat{g_s}(\cdot)\}_{s=1,2,...,S}$. We denote these parameters in vector form as $\alpha = [\alpha_1; ...; \alpha_S]$, where α_s is a column vector with $(K+1)$ entries.

To learn the dictionary parameters, we solve the following optimization problem:

$$\begin{aligned} \underset{\alpha \in \mathbb{R}^{(K+1)S},\ X \in \mathbb{R}^{SN \times M}}{\operatorname{argmin}} \quad & \left\{ ||Y - \mathcal{D}X||_F^2 + \mu \|\alpha\|_2^2 \right\} \\ \text{subject to} \quad & \|x_m\|_0 \leq T_0, \quad \forall m \in \{1, ..., M\}, \\ & 0 \preceq \mathcal{D}_s \preceq c, \forall s \in \{1, 2, ..., S\} \\ & (c-\epsilon)I \preceq \sum_{s=1}^{S} \mathcal{D}_s \preceq (c+\epsilon)I, \end{aligned} \tag{7}$$

where x_m corresponds to column m of the coefficient matrix X, and T_0 is the maximum sparsity level of the coefficients of each training signal. Note that in the objective of (7), 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.

The optimization problem (7) 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 dictionary $\mathcal{D}$ and solve

$$\min_X ||Y - \mathcal{D}X||_F^2 \quad \text{subject to } \|x_m\|_0 \leq T_0 \quad \forall m \in \{1, ..., M\},$$

using orthogonal matching pursuit (OMP). Note that other methods for solving the sparse coding step such as matching pursuit (MP) or iterative soft thresholding could be used as well. In the second step, we fix the coefficients X and update the dictionary $\mathcal{D}$ by finding the

vector of parameters, α, that solves

$$\begin{aligned} &\underset{\alpha\in\mathbb{R}^{(K+1)S}}{\operatorname{argmin}} \left\{||Y-\mathcal{D}X||_F^2+\mu\|\alpha\|_2^2\right\} \\ &\text{subject to} \quad 0\preceq\mathcal{D}_s\preceq cI, \quad \forall s\in\{1,2,...,S\} \\ &\qquad (c-\epsilon)I\preceq\sum_{s=1}^{S}\mathcal{D}_s\preceq(c+\epsilon)I. \end{aligned} \tag{8}$$

Note that α determines $\mathcal{D}_s$ and $\mathcal{D}$ according to (4). Problem (8) is convex and can be written as a quadratic program.

C. Computational Efficiency of the Learned 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 non-structured dictionaries, the dictionary forward and adjoint operators can be efficiently computed when the graph is sparse, as is usually the case in practice. Recall from (4) that $\mathcal{D}^T y=\sum_{s=1}^{S}\sum_{k=0}^{K}\alpha_{sk}\mathcal{L}^k$. The computational cost of the iterative sparse matrix-vector multiplication required to compute $\{\mathcal{L}^k y\}_{k=1,2,...,K}$ is $O(K|\mathcal{E}|)$. 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 [5, 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. Therefore, these efficient implementations are very useful in numerous signal processing tasks, and comprise one of the main advantages of learning structured parametric dictionaries.

IV. Experimental results

In the following experiments, we quantify the performance of the proposed dictionary on both synthetic and real data.

A. Synthetic Examples

We generate a graph by randomly placing $N=80$ vertices in the unit square. We design the edge weights based on the thresholded Gaussian kernel function in such a way 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 $\kappa=0.5$.

To generate training and testing signals, we divide the spectrum of the graph into four bands: $[\lambda_0:\lambda_{19}]$, $[(\lambda_{20}:\lambda_{29})\cup(\lambda_{70}:\lambda_{79})]$, $[\lambda_{30}:\lambda_{49}]$, and $[\lambda_{50}:\lambda_{69}]$. We then generate a synthetic dictionary of $J=320$ atoms, each with a spectral representation that is concentrated exclusively in one of the four bands. In particular, each atom j is of the form

$$d_j=\widehat{h}_j(\mathcal{L})\delta_n=\chi\widehat{h}_j(\Lambda)\chi^T\delta_n, \tag{9}$$

and is independently generated in the following way. We randomly pick one of the four bands and we generate 20 uniformly random coefficients that cover the diagonal entries of $\widehat{h}_j(\Lambda)$ corresponding to the indices of the chosen band. The rest of the entries of $\widehat{h}_j(\Lambda)$ are set to zero. The center vertex n in (9) is then chosen randomly as well, so that the synthetic atoms $\{d_j\}_{j=1,2,...,J}$ are centered at random areas in the graph. Note that the obtained atoms are not guaranteed to be well localized in the graph since the discrete values of $\widehat{h}_j(\Lambda)$ are chosen randomly and are not necessarily derived from a smooth kernel. We generate a set of 1000 training signals by linearly combining (with random coefficients) $T_0=4$ random atoms from the dictionary.

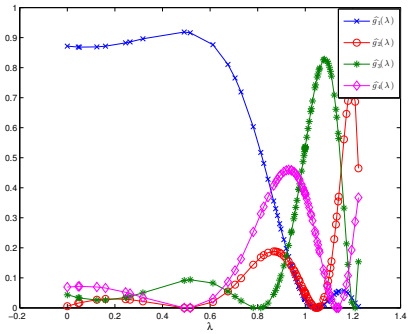
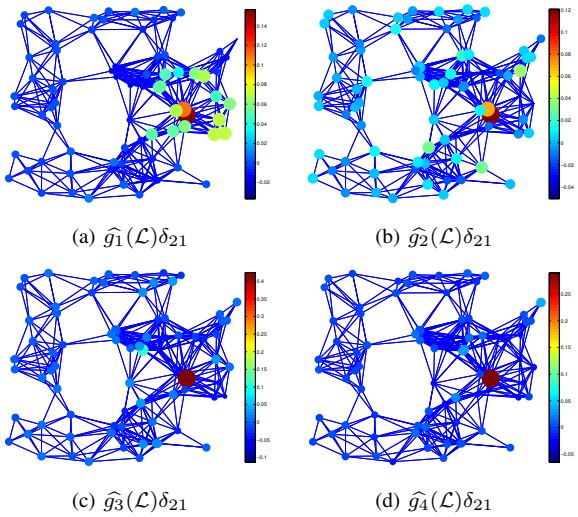


Fig. 1. Learned kernels $\{\widehat{g}_s(\cdot)\}_{s=1,2,3,4}$ in the synthetic examples.



(a) $\widehat{g}_1(\mathcal{L})\delta_{21}$ (b) $\widehat{g}_2(\mathcal{L})\delta_{21}$

(c) $\widehat{g}_3(\mathcal{L})\delta_{21}$ (d) $\widehat{g}_4(\mathcal{L})\delta_{21}$

Fig. 2. Learned atoms centered on vertex $n=21$, from each of the subdictionaries.

Next, we use these training signals to learn a dictionary of $S=4$ subdictionaries using our graph polynomial dictionary learning algorithm, with a polynomial degree of $K=20$ and a sparsity level of $T_0=4$.[1] The kernels bounds in (7) are chosen as $c=1$ and $\epsilon=0.01$. The obtained kernels $\{\widehat{g}_s(\cdot)\}_{s=1,2,3,4}$ are shown in Fig. 1. We observe that each learned kernel captures one of the four bands defined in the synthetic dictionary, following a filter-like behavior. Our algorithm leads to kernels that can be selective in the spectral domain since they combine different parts of the spectrum (see e.g., $\widehat{g}_2(\lambda)$). In Fig. 2, we illustrate the atoms obtained in each of the four subdictionaries and positioned at the same location. We can see that the support of the atoms adapts to the graph topology. The atoms can either be smooth around a particular vertex, as for example in Fig. 2(a), or highly localized, as in Fig. 2(d).

We test the approximation performance of the learned dictionary on a set of 2000 testing signals, generated in the same way as the training signals. We compare the approximation performance of our algorithm for a polynomial degree of $K=\{5,10,20\}$ to that obtained with (i) graph-based transform methods such as the spectral graph wavelet transform (SGWT) [5], (ii) purely numerical dictionary learning methods such as K-SVD [6], and (iii) the graph-based dictionary learning algorithm presented in [4]. The sparse coding

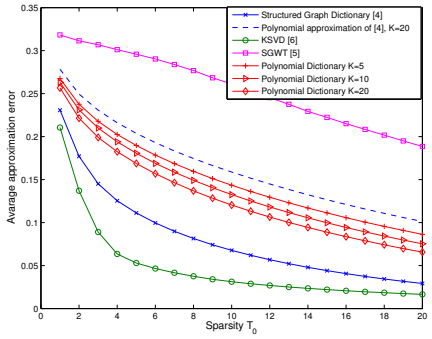
[1] To solve (8), we use the *sdpt3* solver in the *yalmip* optimization toolbox, which is publicly available at http://users.isy.liu.se/johanl/yalmip/

step in the testing phase is performed using orthogonal matching pursuit (OMP) in all cases. The average approximation error is set to $\|Y_{test} - \mathcal{D}X\|_F^2/|Y_{test}|$, where $|Y_{test}|$ is the size of the testing set and is measured for a fixed sparsity level. Fig. 3(a) shows that the approximation performance obtained with our algorithm is consistently better than the one of SGWT, which demonstrates the benefits of the learning process. The improvement is attributed to the fact that the spectral graph wavelet kernels are designed a priori to cover the whole spectrum, while with our dictionary learning algorithm, we learn the shape of the kernels from the data. As expected, the gap in the gain increases as we increase the polynomial degree. On the other hand, we see that K-SVD, which is blind to the graph structure, outperforms our algorithm in terms of approximation performance. This is not surprising as the K-SVD algorithm has more flexibility to adapt the learned dictionary to the set of signals at hand, especially in the case when the number of the training signals is relatively high. However, the K-SVD dictionary is highly non-structured and quite complex to apply. These results illustrate the tradeoff between the adaptability and the complexity of a dictionary. Finally, the algorithm proposed in [4] is an intermediate step 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 $\widehat{g}_s$ does not follow any particular function model, but rather consists of some discrete values. Thus, the obtained dictionary is better in terms of approximation, but it does not benefit from the polynomial structure described in Section III-C. In order to obtained an efficiently implementable dictionary, we fit a polynomial function to the discrete values $\widehat{g}_s$ learned with [4]. We observe that our polynomial dictionary outperforms the polynomial approximation of [4] in terms of approximation performance.

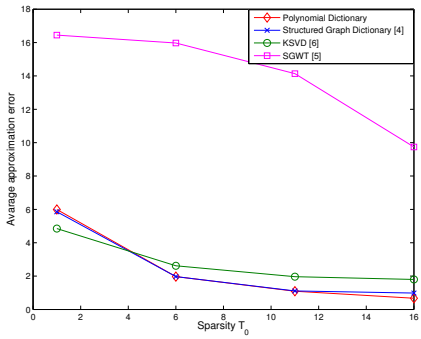
B. Social Network Graph Signals

We now provide experiments with signals from the real world that are well localized in the graph. We consider the daily number of distinct Flickr users that have taken photos at different geographical locations around Trafalgar Square in London, between January 2010 and June 2012. Each vertex of the graph represents a geographical area of 10×10 meters. The graph is computed by assigning an edge between two locations when the distance between them is shorter than 30 meters, and the edge weight is set to be inversely proportional to the distance. We threshold the weights in order to obtain a sparse graph. The number of vertices of the graph is $N = 245$. We set $S = 2$ and $K = 10$. We have a total of 913 signals, and we use 700 of them for training and the rest for testing.

Fig. 3(b) compares the approximation performance of the polynomial dictionary with the dictionaries of SGWT, K-SVD, and the structured graph dictionary. We notice that the polynomial dictionary again outperforms the SGWT. It can even achieve better performance than K-SVD when sparsity increases. In particular, we observe that K-SVD outperforms both graph structured algorithms for a small sparsity level as it learns atoms of global support that can smoothly approximate the whole signal. On the other hand, the polynomial dictionary consists of localized atoms with a support concentrated on the close neighborhood of a vertex, which implies that only a few atoms may not be able to cover the whole graph. However, as the sparsity level increases, the localization property clearly becomes beneficial. Thus, in datasets that are characterized by spatially localized events, the polynomial dictionary can capture important points of interests and landmarks.



(a)



(b)

Fig. 3. Comparison of the polynomial dictionary with SGWT [5], K-SVD [6], the structured graph dictionary [4] in terms of approximation performance in (a) the synthetic examples and (b) the Flickr dataset.

V. Conclusions

We have proposed a structured dictionary model and an algorithm for learning the dictionary parameters for the approximation of signals on graphs. We incorporate the graph structure into the dictionary by considering polynomials of the Laplacian operator. The learned dictionary consists of localized atoms that capture the important spectral and spatial components of the graph signals. Finally, its polynomial structure leads to dictionaries that are efficient in terms of storage and computational complexity, while at the same time are well-adapted to the signals at hand.

Acknowledgement

The authors would like to thank Xiaowen Dong for the fruitful discussions and for providing the Flickr data.

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. of the IEEE*, vol. 98, no. 6, pp. 1045 –1057, Apr. 2010.

[3] 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, 2006.

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

[5] D. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Applied and Computational Harmonic Analysis*, vol. 30, no. 2, pp. 129–150, March 2010.

[6] 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, 2006.