LEARNING OF STRUCTURED GRAPH DICTIONARIES

Xuan Zhang [†], *Xiaowen Dong* [†] [‡], *and Pascal Frossard* [†]

Signal Processing Laboratory ([†] LTS4 / [‡] LTS2)
Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland
{xuan.zhang, xiaowen.dong, pascal.frossard}@epfl.ch

ABSTRACT

We propose a method for learning dictionaries towards sparse approximation of signals defined on vertices of arbitrary graphs. Dictionaries are expected to describe effectively the main spatial and spectral components of the signals of interest, so that their structure is dependent on the graph information and its spectral representation. We first show how operators can be defined for capturing different spectral components of signals on graphs. We then propose a dictionary learning algorithm built on a sparse approximation step and a dictionary update function, which iteratively leads to adapting the structured dictionary to the class of target signals. Experimental results on synthetic and natural signals on graphs demonstrate the efficiency of the proposed algorithm both in terms of sparse approximation and support recovery performance.

***Index Terms*—** dictionary learning, signal processing on graphs, sparse approximations

1. INTRODUCTION

Traditional signal processing techniques deal mostly with signals residing in Euclidean space, such as the real line, where not much information could be extracted from their supports. Nowadays, however, we are increasingly interested in signals that are defined in irregular domains, such as graphs. Typical examples include signals defined in a traffic network or a social network. In these situations, the structure of the underlying graph contains useful information, and the support of the signals depends on the graph characteristics. We are interested in finding sparse representations for these signals where a few significant components are sufficient to describe the most relevant information in the data. Sparsity is often key in the construction of effective signal analysis or compression applications.

A signal on graph $\mathcal{G}$ with a vertex set V is defined as a function $f : V \rightarrow \mathbb{R}$, which assigns a real value to each vertex of the graph. The core idea is to analyze these signals through both the signal and spectrum domain representations, which are analogous to the time and frequency domain representations in classical signal processing literature. In the signal domain, each signal is a vector with its entries corresponding to the vertices of the graph. At the same time, the spectrum domain representation is dependent on the underlying graph structure and can be defined through some isotropic linear operators on graphs via Spectral Graph Theory. For example, the authors in [1] introduce the Spectral Graph Wavelet Transform (SGWT) for signals on graphs where the spectral representation of the signals is defined through the Graph Fourier Transform. The key observation is that we could define the scaling operation in the graph spectrum domain, which relates to the notion of scaling of signals on graphs. Another work that extends wavelets to graphs and manifolds is the Diffusion Wavelets (DW) [2], where the authors use powers of a diffusion operator as smoothing and scaling tools to define wavelet transforms on graphs. The powers of the diffusion operator act as dilation operators applied to the signals, which again enables an implicit definition of scaling of signals on graphs. These wavelet-based representations, however, do not guarantee the sparse representation of specific classes of signals on graphs.

Dictionary learning has been proposed as an attractive approach in the construction of sparse signal approximations. In this paper, we propose a dictionary learning method for the sparse representation of signals on graphs, where the graph Laplacian operator is used to explore the spectrum domain characteristics of the signals. We design a dictionary that is a concatenation of sub-dictionaries with a fixed structure driven by the graph. The structure corresponds to a kernel construction of each sub-dictionary by learning a set of associated variables in the graph spectrum domain. Our dictionary learning objective is two-fold: (1) we would like to have sparse representations for a class of signals as linear expansions of elements from the dictionary; (2) we look for an algorithm that is robust to the presence of noise in the signals. The proposed algorithm is built on two iterative steps, namely, sparse approximation and dictionary updating. We use Orthogonal Matching Pursuit [3] to obtain sparse coefficients for a fixed dictionary in the sparse approximation step. The dictionary is then updated by solving a convex optimization problem. We show by experiments that the proposed dictionary learning solution is effective in terms of signal support recovery and sparse approximation performance. To our knowledge, this paper is the first study about the dictionary learning problem on graphs.

This paper is organized as follows. In Section 2, we propose the structured dictionary for dictionary learning on graphs. In Section 3, we present the two-step optimization scheme, and introduce an algorithm for dictionary updating. We show simulation results in Section 4 with both synthetic and real world data. Finally, we conclude the paper in Section 5.

2. STRUCTURED DICTIONARY ON GRAPHS

In this section, we first recall the definition of the Graph Fourier Transform introduced in [1], and then propose a structure for the design of our dictionaries.

2.1. Graph Fourier Transform

Consider a weighted and undirected graph $\mathcal{G} = (V, E, \mathbf{w})$ with N vertices, where V and E represent the vertex and edge set of the graph and $\mathbf{w}$ gives a positive real weight to each edge of the graph. We assume that the graph is connected. Recall that the complex exponentials $e^{i\omega x}$ are eigenfunctions of the one-dimensional Laplace

This work has been partly funded by Nokia Research Center (NRC), Lausanne, Switzerland.

978-1-4673-0046-9/12/$26.00 ©2012 IEEE

operator $-\frac{d^2}{dx^2}$. They constitute an orthogonal basis in $\mathcal{L}_2$ space, and the classical Fourier transform is defined as scalar products of signals in $\mathcal{L}_2$ and these complex exponentials. The Graph Fourier Transform can be defined in a similar way. For a graph $\mathcal{G}$, the graph Laplacian operator $\mathbf{L}$ is defined as

$$\mathbf{L} = \mathbf{\Delta} - \mathbf{W} \tag{1}$$

where $\mathbf{W}$ is the adjacency matrix and $\mathbf{\Delta}$ is the degree matrix containing the degree of the vertices along diagonal. Since $\mathbf{L}$ is a real symmetric matrix, it has the following eigen-decomposition:

$$\mathbf{L} = \boldsymbol{\chi}\mathbf{\Sigma}\boldsymbol{\chi}^T \tag{2}$$

where $\boldsymbol{\chi}$ is the eigenbasis and $\mathbf{\Sigma}$ is the matrix containing the discrete spectrum (eigenvalues) along diagonal. Based on $\boldsymbol{\chi}$, we define the Graph Fourier Transform as the following:

Definition 1 (Graph Fourier Transform [1]). *For any function $\mathbf{y} \in \mathbb{R}^N$ defined on the vertices of a graph $\mathcal{G}$, the Graph Fourier Transform (GFT) $\hat{\mathbf{y}}$ is defined as*

$$\hat{y}(j) = \langle \boldsymbol{\chi}_j, \mathbf{y} \rangle = \sum_{n=1}^{N} \chi_j(n) y(n). \tag{3}$$

Similarly, the Inverse Graph Fourier Transform (IGFT) is defined as

$$y(n) = \sum_{j=1}^{N} \hat{y}(j) \chi_j(n). \tag{4}$$

By applying the graph Laplacian operator $\mathbf{L}$ to the signal $\mathbf{y}$, we have $\mathbf{L}\mathbf{y} = \boldsymbol{\chi}\mathbf{\Sigma}\boldsymbol{\chi}^T\mathbf{y}$. Notice that $\hat{\mathbf{y}} = \boldsymbol{\chi}^T\mathbf{y}$ is the GFT of $\mathbf{y}$, and $\boldsymbol{\chi}(\mathbf{\Sigma}\hat{\mathbf{y}})$ is the IGFT of $\hat{\mathbf{y}}$ filtered by $\mathbf{\Sigma}$ in the graph spectrum domain.

2.2. Structured dictionary

We have seen above that we can build filtering operators in the graph spectrum domain. This is exactly the property that we exploit for learning dictionaries that are able to capture meaningful spectral components of signals on graphs. Since an appropriate design of $\mathbf{\Sigma}$ permits to implement filtering of the signals in the spectrum domain, we propose to define a structured over-complete dictionary that includes several filtering functions to capture the different spectral components of the signal. We thus consider dictionaries that are defined as

$$\begin{aligned} \mathbf{D} &= [\mathbf{D}_1, \mathbf{D}_2, \ldots, \mathbf{D}_S] && (5) \\ &= [\boldsymbol{\chi}\mathbf{\Lambda}_1\boldsymbol{\chi}^T, \boldsymbol{\chi}\mathbf{\Lambda}_2\boldsymbol{\chi}^T, \ldots, \boldsymbol{\chi}\mathbf{\Lambda}_S\boldsymbol{\chi}^T] && (6) \end{aligned}$$

where $\{\mathbf{D}_i\}_{i=1}^S$ are S sub-dictionaries with a certain structure and $\{\mathbf{\Lambda}_i\}_{i=1}^S$ are some diagonal and positive semidefinite matrices ($\mathbf{\Lambda}_i \succcurlyeq \mathbf{0}$).

As one can see here, the set of matrices $\{\mathbf{\Lambda}_i\}_{i=1}^S$ plays an important role for filtering in the spectrum domain. Another interpretation is that we embed each vertex into S different scalar product spaces with dimension N. In the k-th such space, a vertex v_i is mapped to $(\chi_{1,i}, \chi_{2,i}, \ldots, \chi_{N,i})$ where the metric tensor $\mathbf{g}_k = \mathbf{\Lambda}_k$ is learned to promote features leading to sparsity. $\mathbf{D}_k$ can then be considered as a kernel (covariance matrix) in the embedded space measuring the similarity of the embedded vertices. As remarks, we also noticed that the Spectral Graph Wavelets and Diffusion Wavelets (before orthogonalization) share a similar structure as that in (6).

3. DICTIONARY LEARNING ON GRAPHS

Adopting the proposed structure, we need to properly design $\{\mathbf{\Lambda}_i\}_{i=1}^S$ such that the resulting dictionary $\mathbf{D} \in \mathbb{R}^{N \times K}$ could well represent the signals $\{\mathbf{y}_j\}_{j=1}^M \in \mathbb{R}^N$ as linear combinations of its elements ($K = NS$ where S is the number of sub-dictionaries). Here $\mathbf{D}$ is overcomplete (i.e., $K > N$) and each column $\{\mathbf{d}_j\}_{j=1}^K \in \mathbb{R}^N$ is called an atom. Dictionary learning can be formulated as the following optimization problem:

$$\min_{\mathbf{D},\mathbf{X}} \|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 \qquad \text{subject to} \quad \|\mathbf{x}_j\|_0 \le T_0, \quad \forall j \tag{7}$$

where $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_M]$ are the training signals, $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M]$ are the sparse coefficients, and $\|\cdot\|_0$ denotes the l^0 pseudo-norm counting the non-zero entries of a vector. Such a problem is unfortunately quite complex to solve directly. One possible approximate solution relies on a two-step optimization scheme where a sparse approximation step and a dictionary updating step are performed iteratively. In the sparse approximation step, we find $\mathbf{x}_j$ under a fixed $\mathbf{D}$; in the dictionary updating step, we update $\mathbf{D}$ while fixing $\mathbf{x}_j$. We describe these two steps in the rest of this section.

3.1. Sparse approximation

The sparse approximation step aims at finding approximation of training signals with a few atoms of a fixed dictionary. We use Orthogonal Matching Pursuit (OMP) to find sparse representations of the training signals $\mathbf{Y}$ with respect to the fixed dictionary $\mathbf{D}$. OMP employs a greedy optimization strategy where atoms are picked successively in order to best approximate the signals. We further normalize each atom in the dictionary so that they all have a unit l^2 norm before running OMP.

3.2. Dictionary updating

Based on the proposed structure, the update of the dictionary is equivalent to computing the operators $\{\mathbf{\Lambda}_i\}_{i=1}^S$. Since a change of any entry in $\mathbf{\Lambda}_i$ affects all atoms in the corresponding sub-dictionary, we update one sub-dictionary at a time. Specifically, we update each $\mathbf{D}_k$ while fixing all other sub-dictionaries, with the aim of finding a better $\mathbf{\Lambda}_k$ that minimizes the approximation error. Suppose that, after the sparse approximation step, we have obtained a sparse coefficient matrix $\mathbf{X} \in \mathbb{R}^{K \times M}$ for training signals $\mathbf{Y} \in \mathbb{R}^{N \times M}$. For each sub-dictionary $\mathbf{D}_k$ in the dictionary $\mathbf{D}$, the corresponding coefficients are denoted by $\mathbf{X}_k \in \mathbb{R}^{N \times M}$. Therefore, the approximation error due to the dictionary $\mathbf{D}_k$ can be written as

$$\begin{aligned} &\|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 = \|\mathbf{Y} - \sum_{s=1}^{S} \mathbf{D}_s\mathbf{X}_s\|_F^2 = \|\mathbf{E}_k - \mathbf{D}_k\mathbf{X}_k\|_F^2 \\ =&\|\mathbf{E}_k - \boldsymbol{\chi}\mathbf{\Lambda}_k\boldsymbol{\chi}^T\mathbf{X}_k\|_F^2 \end{aligned} \tag{8}$$

where $\mathbf{E}_k = \mathbf{Y} - \sum_{s=1, s\neq k}^{S} \mathbf{D}_s\mathbf{X}_s$ is the approximation error that does not depend on $\mathbf{D}_k$. Rewriting the Frobenius norm in a trace form and removing terms that are constant with respect to $\mathbf{\Lambda}_k$, the minimization of the approximation error becomes equivalent to

$$\begin{aligned} &\min_{\mathbf{\Lambda}_k} \|\mathbf{E}_k - \boldsymbol{\chi}\mathbf{\Lambda}_k\boldsymbol{\chi}^T\mathbf{X}_k\|_F^2 \\ =&\min_{\mathbf{\Lambda}_k} Tr[\mathbf{E}_k^T\mathbf{E}_k - \mathbf{E}_k^T\boldsymbol{\chi}\mathbf{\Lambda}_k\boldsymbol{\chi}^T\mathbf{X}_k \\ &- (\mathbf{E}_k^T\boldsymbol{\chi}\mathbf{\Lambda}_k\boldsymbol{\chi}^T\mathbf{X}_k)^T + \mathbf{X}_k^T\boldsymbol{\chi}\mathbf{\Lambda}_k^2\boldsymbol{\chi}^T\mathbf{X}_k] \\ \equiv&\min_{\mathbf{\Lambda}_k} Tr[-2\boldsymbol{\chi}^T\mathbf{X}_k\mathbf{E}_k^T\boldsymbol{\chi}\mathbf{\Lambda}_k + \boldsymbol{\chi}^T\mathbf{X}_k\mathbf{X}_k^T\boldsymbol{\chi}\mathbf{\Lambda}_k^2] \end{aligned} \tag{9}$$

where $Tr(\cdot)$ denotes the trace of a matrix. Since $\mathbf{\Lambda}_k$ lies on a differential manifold, we could take derivatives directly. The second derivative of the objective function in (9) turns out to be positive semidefinite indicating that the optimization problem (9) is convex with respect to $\mathbf{\Lambda}_k$. Therefore, we need to solve the following convex optimization problem to update $\mathbf{\Lambda}_k$:

$$\begin{aligned}&\min_{\mathbf{\Lambda}_k} Tr[-2\boldsymbol{\chi}^T\mathbf{X}_k\mathbf{E}_k^T\boldsymbol{\chi}\mathbf{\Lambda}_k + \boldsymbol{\chi}^T\mathbf{X}_k\mathbf{X}_k^T\boldsymbol{\chi}\mathbf{\Lambda}_k^2]\\ &\text{subject to} \qquad \forall k = 1, 2, \ldots, S \\ &\qquad\qquad\qquad \mathbf{\Lambda}_k \text{ is diagonal and } \mathbf{\Lambda}_k \succcurlyeq \mathbf{0}.\end{aligned} \tag{10}$$

Taking derivative with respect to $\mathbf{\Lambda}_k$ for the objective, we have:

$$\begin{aligned}&\frac{\partial Tr[-2\boldsymbol{\chi}^T\mathbf{X}_k\mathbf{E}_k^T\boldsymbol{\chi}\mathbf{\Lambda}_k + \boldsymbol{\chi}^T\mathbf{X}_k\mathbf{X}_k^T\boldsymbol{\chi}\mathbf{\Lambda}_k^2]}{\partial\mathbf{\Lambda}_k}\\ &= -2[(\boldsymbol{\chi}^T\mathbf{X}_k\mathbf{E}_k^T\boldsymbol{\chi}) \circ \mathbf{I}] + 2\mathbf{\Lambda}_k[(\boldsymbol{\chi}^T\mathbf{X}_k\mathbf{X}_k^T\boldsymbol{\chi}) \circ \mathbf{I}]\end{aligned} \tag{11}$$

where $\circ$ denotes the Hadamard product and $\mathbf{I}$ is the identity matrix. We then write $\boldsymbol{\chi}^T\mathbf{X}_k\mathbf{E}_k^T\boldsymbol{\chi} = \text{diag}(a_1, a_2, \ldots, a_N)$ and $\boldsymbol{\chi}^T\mathbf{X}_k\mathbf{X}_k^T\boldsymbol{\chi} = \text{diag}(b_1, b_2, \ldots, b_N)$, and the solution to (10) is thus

$$\lambda_{k,i} = \begin{cases} \max(0, \frac{a_i}{b_i}) & b_i \neq 0 \\ \lambda_{k,i} & b_i = 0. \end{cases} \tag{12}$$

Based on the two steps described above, the complete dictionary learning algorithm is given in Algorithm 1.

Algorithm 1 Structured Dictionary Learning

```
INITIALIZATION: Set $\lambda_{k,j}$ randomly in $\mathbf{D}$ from Half-normal distribution.
repeat
  1. Sparse Approximation Step:
  a. Normalize each atom in $\mathbf{D}$;
  b. Use OMP to compute the coefficients matrix $\mathbf{X}$ for $\mathbf{Y}$ under the current dictionary $\mathbf{D}$;
  c. Rescale the coefficients and atoms to recover the structure;
  2. Dictionary Updating Step:
  for $k = 1 \to S$ do
    Compute the overall approximation error matrix $\mathbf{E}_k$ as
        $$\mathbf{E_k} = \mathbf{Y} - \sum_{s=1, s\neq k}^{S} \mathbf{D}_s\mathbf{X}_s$$
    $\text{diag}(b_1, b_2, \ldots, b_N) = \boldsymbol{\chi}^T\mathbf{X}_k\mathbf{X}_k^T\boldsymbol{\chi}$
    $\text{diag}(a_1, a_2, \ldots, a_N) = \boldsymbol{\chi}^T\mathbf{X}_k\mathbf{E}_k^T\boldsymbol{\chi}$
    Update $\mathbf{D}_k$ through the following loop
    for $i = 1 \to N$ do
      if $b_i \neq 0$ then
            $$\lambda_{k,i} = \max(0, \frac{a_i}{b_i})$$
      else
            $\lambda_{k,i}$ unchanged
      end if
    end for
  end for
until the maximum number of iterations is reached
```

4. EXPERIMENTAL RESULTS

4.1. Synthetic signals on graphs

We first evaluate the performance of the dictionary learning algorithm on synthetic signals and show that it is able to recover the main components of the signals of interest. Our goal is to recover a synthetic dictionary $\mathbf{D}^t$ from noisy signals built as linear combinations of its elements. We examined the influence of the graph size, the number of training signals, and the noise on the performance of our algorithm. The graph consists of N vertices, each of which has $\log N$ incident edges on average with weights being uniformly distributed in the unit interval. After obtaining $\boldsymbol{\chi}$ from the graph Laplacian, $\mathbf{D}^t$ of $K = NS$ atoms are generated by setting $S = 5$ and initializing $\{\mathbf{\Lambda}_i^t\}_{i=1}^S$ as Half-normal random variables. The sparsity level is chosen as $L = \lfloor(1 + \sqrt{N})/2\rfloor$ and the training signals are generated by randomly combining L atoms in $\mathbf{D}^t$ with normally distributed coefficients. In each scenario, the learned dictionary $\mathbf{D}$ is compared with $\mathbf{D}^t$ to show the average percentage of original atoms that are recovered by the learning algorithms, over 100 runs. The comparison is done by first normalizing all atoms in $\mathbf{D}$ and $\mathbf{D}^t$ and then matching an atom $\mathbf{d}_i$ in $\mathbf{D}$ to one atom $\mathbf{d}_j^t$ in $\mathbf{D}^t$ under the criteria $1 - \mathbf{d}_i\mathbf{d}_j^t < 0.01$. The number of iterations for the sparse approximation and dictionary updating steps is limited to 100.

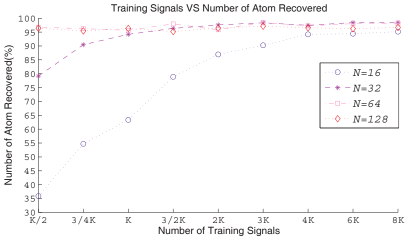

Fig. 1: Percentage of atoms recovered for different graph sizes N and different numbers of training signals M (100 runs, S=5, SNR=20dB).

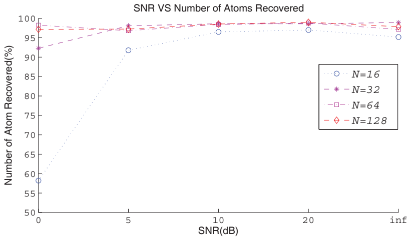

Fig. 2: Percentage of atoms recovered for different graph sizes N and different levels of SNR (100 runs, S=5, M=8K).

We initially study the impact of the graph size N and the number of training signals M on the learned dictionary. We add White Gaussian Noise (WGN) to the training signals with Signal to Noise Ratio (SNR) 20dB. The experiments are performed for graphs of size $N = 16, 32, 64, 128$. For each N, the number of training signals M is chosen as $K/2, 3/4K, K, 3/2K, 2K, 3K, 4K, 6K, 8K$. In Fig. 1, X-axis indicates the number of training signals available and Y-axis is the corresponding percentage of atoms recovered. As we can see, the algorithm performed quite well with over 90% of the atoms being recovered on average when the number of training signals exceeds $4K$. This shows that, empirically, our algorithm converged and recovered most atoms with $\Omega(K)$ training signals which is only linearly proportional to K. We also notice that the larger the graph, the better the support recovery performance in general. This can be explained by the fact that we generate $\mathbf{D}^t$ randomly and larger graphs tend to lead to less correlation among atoms in this case.

Next, we investigate the impact of the noise for different graph sizes. We choose $N = 16, 32, 64, 128$, SNR(dB) $=$

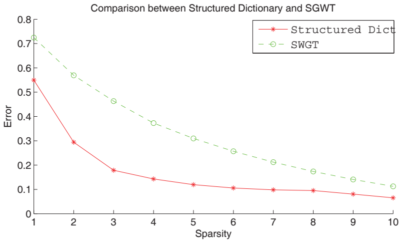

Fig. 3: Comparison of Structured Dictionary with SGWT in terms of approximation error (100 runs, S=5, N=20, L=3, M=8K).

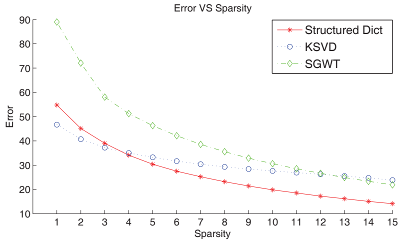

Fig. 4: Comparison of Structured Dictionary Learning with K-SVD and SGWT in terms of approximation error in the Twitter simulation (100 runs, S=4, N=63, M=4032).

$0, 5, 10, 20, +\infty$ and $M = 8K$. The result is shown in Fig. 2 where it can be seen that our algorithm is also robust to the noise contamination as the performance is reasonably stable when SNR $\geq$ 5dB. This is easy to explain as random noise does not match the graphs. Thus, our algorithm seems to be capable of distinguishing the signals from the added random noise thanks to the utilization of the graph information.

Finally, we compare our learned dictionary with SGWT. $\mathbf{D}^t$ is used to generate both training signals and testing signals. We set $S = 5$, $N = 20$, $L = 3$ and add WGN only to the training signals with SNR 20dB. The number of training signals and that of testing signals are $8K$. We use our learned dictionary and SGWT to represent testing signals under different sparsity levels and the comparison is purely based on the average approximation error per signal. From Fig. 3 we can see that our algorithm achieves a smaller error than SGWT under the same sparsity level, which demonstrates the benefit of the learning process. Notice that when sparsity exceeds 3, our algorithm is able to recover almost all atoms in $\mathbf{D}^t$, and the approximation error is due to the failure of recovering the exact atoms used to generate some of the testing signals. The two main reasons are: (1) OMP used in the sparse approximation step is a very greedy algorithm; (2) the correlation among atoms might be high when N is relatively small in view of the randomized way of creating $\mathbf{D}^t$.

4.2. Real world data

We now compare our learned dictionary with the one learned by K-SVD [4] and SGWT using real world data. In K-SVD, the dictionary is learned only from the training signals, while in SGWT the wavelets are predefined without any learning process involved. All three dictionaries are of the same size and S is chosen to be 4. The comparison is based on the approximation errors under fixed sparsity level. For each sparsity level, we first obtain the dictionaries, and then apply OMP to the testing signals to calculate the approximation errors between the testing signals and the approximated signals.

In our simulation, we generate a small social network of 63 Twitter users. The graph is directly constructed from the social relationship between these users on Twitter. Specifically, if user A follows user B or vice versa, we add an edge between them with weight 1. We collect 50 signals on this graph, each of which counts the number of tweets containing predefined keywords that each user has posted during a fixed time window. Due to the limited number of signals, we randomly pick up 25 out of the 50 signals to generate 4032 training signals by linear combinations with normally distributed coefficients. The number of signals used for the linear combinations is 5. In exactly the same manner, the remaining 25 signals are used to generate the testing signals. We also add noise to all signals such that the SNR is 20dB. We run the simulation 100 times and average the errors.

The result is shown in Fig. 4. X-axis is indexed by sparsity level and Y-axis is the corresponding error per signal. Compared to SGWT, our algorithm always achieves lower approximation errors, which again demonstrates the benefit of the learning process on the graph. In terms of comparison with K-SVD which is blind to the graph information, our algorithm shows a faster error declining rate by creating more patterns in the dictionary. When sparsity is below 4, K-SVD outperformed with relatively large error. It indicates that the tolerance of large error offsets the benefits contributed by the graph. Moreover, as we have seen in our synthetic simulations, our algorithm can efficiently distinguish the signals from the noise. The error for our algorithm showing up here thus contains the noise added to the testing signals, which degraded its performance. Our method starts to outperform K-SVD when sparsity exceeds 3, which shows the advantage of the utilization of the graph information. After the sparsity reaches 13, another transition point occurs where SGWT beats K-SVD, which again illustrates the significance of the graph in representing these signals. We also notice that K-SVD only tends to sparsely represent the training signals rather than extracting the common features of the class of signals on graphs, inevitably leading to overfitting the training signals, which is a major drawback of K-SVD. By contrast, the structure in our dictionary imposed by the graph prevents the algorithm from overfitting. Therefore, we believe that this is another reason for our algorithm to perform better than K-SVD in approximating the testing signals.

5. CONCLUSION

In this paper, we have proposed a method for dictionary learning on graphs. We make use of the graph Laplacian operator to exploit the spectral representation of signals on graphs. The learned dictionary is able to capture the spectral features of the signals, in return providing sparse representations for the signals on graphs. The simulation results demonstrate that our algorithm is able to efficiently exploit the graph information and that it is robust to noise in the training setup.

6. REFERENCES

[1] D. K. Hammond, P. Vandergheynst, and R. Gribonval, “Wavelets on graphs via spectral graph theory,” *Applied and Computational Harmonic Analysis*, vol. 30, pp. 129–150, 2010.

[2] R. R. Coifman and M. Maggioni, “Diffusion wavelets,” *Applied and Computational Harmonic Analysis*, vol. 21, pp. 53–94, 2006.

[3] Z. Zhang, “Matching pursuit,” Ph.D. dissertation, Courant Institute, New York University, 1993.

[4] M. Aharon, M. Elad, and A. M. Bruckstein, “K-svd: An algorithm for designing overcomplete dictionaries for sparse representation,” *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.