Multiresolution Matrix Factorization and Wavelet Networks on Graphs

Truong Son Hy [1] **Risi Kondor** [1]

Abstract

Multiresolution Matrix Factorization (MMF) is unusual amongst fast matrix factorization algorithms in that it does not make a low rank assumption. This makes MMF especially well suited to modeling certain types of graphs with complex multiscale or hierarchical strucutre. While MMF promises to yields a useful wavelet basis, finding the factorization itself is hard, and existing greedy methods tend to be brittle. In this paper, we propose a "learnable" version of MMF that carfully optimizes the factorization with a combination of reinforcement learning and Stiefel manifold optimization through backpropagating errors. We show that the resulting wavelet basis far outperforms prior MMF algorithms and provides the first version of this type of factorization that can be robustly deployed on standard learning tasks. Furthermore, we construct the wavelet neural networks (WNNs) learning graphs on the spectral domain with the wavelet basis produced by our MMF learning algorithm. Our wavelet networks are competitive against other state-of-the-art methods in molecular graphs classification and node classification on citation graphs.

1. Introduction

In certain machine learning problems large matrices have complex hierarchical structures that traditional linear algebra methods based on the low rank assumption struggle to capture. Multiresolution matrix factorization (MMF) is a relatively little used alternative paradigm that is designed to capture structure at multiple different scales. MMF has been found to be particularly effective at compressing the adjacency or Laplacian matrices of graphs with complicated structure, such as social networks (Kondor et al., 2014).

MMF factorizations have a number of advantages, including the fact that they are easy to invert and have an interpretation as a form of wavelet analysis on the matrix and consequently on the underlying graph. The wavelets can be used e.g., for finding sparse approximations of graph signals. Finding the actual MMF factorization however is a hard optimization problem combining elements of continuous and combinatorial optimization. Most of the existing MMF algorithms just tackle this with a variety of greedy heuristics and are consequently brittle: the resulting factorizations typically have large variance and most of the time yield factorizations that are far from the optimal (Teneva et al., 2016; Ithapu et al., 2017; Ding et al., 2017).

The present paper proposes an alternative paradigm to MMF optimization based on ideas from deep learning. Specifically, we employ an iterative approach to optimizing the factorization based on backpropagating the factorization error and a reinforcement learning strategy for solving the combinatorial part of the problem. While more expensive than the greedy approaches, we find that the resulting "learnable" MMF produces much better quality factorizations and a wavelet basis that is smoother and better reflects the structure of the underlying matrix or graph. Unsurprisingly, this also means that the factorization performs better in downstream tasks.

To apply our learnable MMF algorithm to standard benchmark tasks, we also propose a wavelet extension of the Spectral Graph Networks algorithm of (Bruna et al., 2014) which we call the Wavelet Neural Network (WNN). Our experiments show that the combination of learnable MMF optimization with WNNs achieves state of the art results on several graph learning tasks. Beyond just benchmark performance, the greatly improved stability of MMF optimization process and the similarity of the hierarchical structure of the factorization to the architecture of deep neural networks opens up the possibility of MMF being tightly integrated with other learning algorithms in the future.

2. Related work

Compressing and estimating large matrices has been extensively studied from various directions, including (Drineas et al., 2006), (Halko et al., 2011), (Williams & Seeger, 2001) (Kumar et al., 2012), (Mahoney, 2011), (Jenatton et al., 2010). Many of these methods come with explicit guarantees but typically make the assumption that the matrix to be

[1]Department of Computer Science, University of Chicago. Correspondence to: Truong Son Hy <hytruongson@uchicago.edu>.

2nd AI4Science Workshop at the 39th International Conference on Machine Learning (ICML), 2022. Copyright 2022 by the author(s).

approximated is low rank.

MMF is more closely related to other works on constructing wavelet bases on discrete spaces, including wavelets defined based on diagonalizing the diffusion operator or the normalized graph Laplacian (Coifman & Maggioni, 2006) (Hammond et al., 2011) and multiresolution on trees (Gavish et al., 2010) (Lee et al., 2008). MMF has been used for matrix compression (Teneva et al., 2016), kernel approximation (Ding et al., 2017) and inferring semantic relationships in medical imaging data (Ithapu et al., 2017).

Graph neural networks (GNNs) utilizing the generalization of convolution concept to graphs have been popularly applied to many learning tasks such as estimating quantum chemical computation, and modeling physical systems, etc. Spectral methods such as (Bruna et al., 2014) provide one way to define convolution on graphs is via convolution theorem and graph Fourier transform (GFT). To address the high computational cost of GFT, (Xu et al., 2019) proposed to use the diffusion wavelet bases as previously defined by (Coifman & Maggioni, 2006) instead for a faster transformation.

3. Background on Multiresolution Matrix Factorization

The *Multiresolution Matrix Factorization* (MMF) of a matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is a factorization of the form

$$\boldsymbol{A} = \boldsymbol{U}_1^T \boldsymbol{U}_2^T \dots \boldsymbol{U}_L^T \boldsymbol{H} \boldsymbol{U}_L \dots \boldsymbol{U}_2 \boldsymbol{U}_1,$$

where the $\boldsymbol{H}$ and $\boldsymbol{U}_1, \dots, \boldsymbol{U}_L$ matrices conform to the following constraints: (i) Each $\boldsymbol{U}_\ell$ is an orthogonal matrix that is a k-point rotation for some small k, meaning that it only rotates k coordinates at a time; (ii) There is a nested sequence of sets $\mathbb{S}_L \subseteq \dots \subseteq \mathbb{S}_1 \subseteq \mathbb{S}_0 = [n]$ such that the coordinates rotated by $\boldsymbol{U}_\ell$ are a subset of $\mathbb{S}_\ell$; and (iii) $\boldsymbol{H}$ is an $\mathbb{S}_L$-core-diagonal matrix meaning that is diagonal with a an additional small $\mathbb{S}_L \times \mathbb{S}_L$ dimensional "core". Finding the best MMF factorization to a symmetric matrix $\boldsymbol{A}$ involves solving

$$\min_{\substack{\mathbb{S}_L \subseteq \dots \subseteq \mathbb{S}_1 \subseteq \mathbb{S}_0 = [n] \\ \boldsymbol{H} \in \mathbb{H}_n^{\mathbb{S}_L}; \boldsymbol{U}_1, \dots, \boldsymbol{U}_L \in \mathbb{O}}} \|\boldsymbol{A} - \boldsymbol{U}_1^T \dots \boldsymbol{U}_L^T \boldsymbol{H} \boldsymbol{U}_L \dots \boldsymbol{U}_1\|. \quad (1)$$

Assuming that we measure error in the Frobenius norm, (19) is equivalent to

$$\min_{\substack{\mathbb{S}_L \subseteq \dots \subseteq \mathbb{S}_1 \subseteq \mathbb{S}_0 = [n] \\ \boldsymbol{U}_1, \dots, \boldsymbol{U}_L \in \mathbb{O}}} \|\boldsymbol{U}_L \dots \boldsymbol{U}_1 \boldsymbol{A} \boldsymbol{U}_1^T \dots \boldsymbol{U}_L^T\|_{\text{resi}}^2, \quad (2)$$

where $\|\cdot\|_{\text{resi}}^2$ is the squared residual norm $\|\boldsymbol{H}\|_{\text{resi}}^2 = \sum_{i \neq j; (i,j) \notin \mathbb{S}_L \times \mathbb{S}_L} |\boldsymbol{H}_{i,j}|^2$. There are two fundamental difficulties in MMF optimization: finding the optimal nested sequence of $\mathbb{S}_\ell$ is a combinatorially hard (e.g., there are $\binom{d_\ell}{k}$ ways to choose k indices out of $\mathbb{S}_\ell$); and the solution for $\boldsymbol{U}_\ell$ must satisfy the orthogonality constraint such that $\boldsymbol{U}_\ell^T \boldsymbol{U}_\ell = \boldsymbol{I}$. The existing literature on solving this optimization problem (Kondor et al., 2014) (Teneva et al., 2016) (Ithapu et al., 2017) (Ding et al., 2017) has various heuristic elements and has a number of limitations:

- There is no guarantee that the greedy heuristics (e.g., clustering) used in selecting k rows/columns $\mathbb{I}_\ell = \{i_1, .., i_k\} \subset \mathbb{S}_\ell$ for each rotation return a globally optimal factorization.
- Instead of direct optimization for each rotation $\boldsymbol{U}_\ell \triangleq \boldsymbol{I}_{n-k} \oplus_{\mathbb{I}_\ell} \boldsymbol{O}_\ell$ where $\boldsymbol{O}_\ell \in \mathbb{SO}(k)$ globally and simultaneously with the objective (19), Jacobi MMFs (see Proposition 2 of (Kondor et al., 2014)) apply the greedy strategy of optimizing them locally and sequentially. Again, this does not necessarily lead to a *globally* optimal combination of rotations.
- Most MMF algorithms are limited to the simplest case of $k = 2$ where $\boldsymbol{U}_\ell$ is just a Given rotation, which can be parameterized by a single variable, the rotation angle θ_ℓ. This makes it possible to optimize the greed objective by simple gradient descent, but larger rotations would yield more expressive factorizations and better approximations.

In contrast, we propose an iterative algorithm to directly optimize the global MMF objective (19):

- We use gradient descent algorithm on the Stiefel manifold to optimize all rotations $\{\boldsymbol{U}_\ell\}_{\ell=1}^L$ *simultaneously*, whilst satisfying the orthogonality constraints. Importantly, the Stiefel manifold optimization is not limited to $k = 2$ case (Section 4).
- We formulate the problem of finding the optimal nested sequence $\mathbb{S}_L \subseteq \dots \subseteq \mathbb{S}_1 \subseteq \mathbb{S}_0 = [n]$ as learning a Markov Decision Process (MDP) that can be subsequently solved by the gradient policy method of Reinforcement Learning (RL), in which the RL agent (or stochastic policy) is modeled by graph neural networks (GNN) (Section 5).

We show that the resulting learning-based MMF algorithm outperforms existing greedy MMFs and other traditional baselines for matrix approximation in various scenarios (see Section 7).

4. Stiefel Manifold Optimization

The MMF optimization problem in (19) and (2) is equivalent to

$$\min_{\mathbb{S}_L \subseteq \dots \subseteq \mathbb{S}_1 \subseteq \mathbb{S}_0 = [n]} \min_{\boldsymbol{U}_1, \dots, \boldsymbol{U}_L \in \mathbb{O}} \|\boldsymbol{U}_L \dots \boldsymbol{U}_1 \boldsymbol{A} \boldsymbol{U}_1^T \dots \boldsymbol{U}_L^T\|_{\text{resi}}^2, \quad (3)$$

In order to solve the inner optimization problem of (3), we consider the following generic optimization with orthogonality constraints:

$$\min_{\boldsymbol{X}\in\mathbb{R}^{n\times p}} \mathcal{F}(\boldsymbol{X}), \text{ s.t. } \boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{I}_p, \tag{4}$$

where $\boldsymbol{I}_p$ is the identity matrix and $\mathcal{F}(\boldsymbol{X}) : \mathbb{R}^{n\times p} \to \mathbb{R}$ is a differentiable function. The feasible set $\mathcal{V}_p(\mathbb{R}^n) = \{\boldsymbol{X} \in \mathbb{R}^{n\times p} : \boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{I}_p\}$ is referred to as the Stiefel manifold of p orthonormal vectors in $\mathbb{R}^n$ that has dimension equal to $np - \frac{1}{2}p(p+1)$. We will view $\mathcal{V}_p(\mathbb{R}^n)$ as an embedded submanifold of $\mathbb{R}^{n\times p}$.

When there is more than one orthogonal constraint, (24) is written as

$$\min_{\boldsymbol{X}_1\in\mathcal{V}_{p_1}(\mathbb{R}^{n_1}),\ldots,\boldsymbol{X}_q\in\mathcal{V}_{p_q}(\mathbb{R}^{n_q})} \mathcal{F}(\boldsymbol{X}_1,\ldots,\boldsymbol{X}_q) \tag{5}$$

where there are q variables with corresponding q orthogonal constraints. For example, in the MMF optimization problem (19), suppose we are already given $\mathbb{S}_L \subseteq \cdots \subseteq \mathbb{S}_1 \subseteq \mathbb{S}_0 = [n]$ meaning that the indices of active rows/columns at each resolution were already determined, for simplicity. In this case, we have $q = L$ number of variables such that each variable $\boldsymbol{X}_\ell = \boldsymbol{O}_\ell \in \mathbb{R}^{k\times k}$, where $\boldsymbol{U}_\ell = \boldsymbol{I}_{n-k} \oplus_{\mathbb{I}_\ell} \boldsymbol{O}_\ell \in \mathbb{R}^{n\times n}$ in which $\mathbb{I}_\ell$ is a subset of k indices from $\mathbb{S}_\ell$, must satisfy the orthogonality constraint. The corresponding objective function is

$$\mathcal{F}(\boldsymbol{O}_1,\ldots,\boldsymbol{O}_L) = \|\boldsymbol{U}_L\ldots\boldsymbol{U}_1\boldsymbol{A}\boldsymbol{U}_1^T\ldots\boldsymbol{U}_L^T\|_{\text{resi}}^2. \tag{6}$$

Details about Stiefel manifold optimization is included in the Appendix.

5. Reinforcement Learning

5.1. Problem formulation

We formulate the problem of finding the optimal nested sequence of sets $\mathbb{S}_L \subseteq \cdots \subseteq \mathbb{S}_1 \subseteq \mathbb{S}_0 = [n]$ as learning an RL agent in order to solve the MMF optimization in (19). There are two fundamental parts to index selection for each resolution level $\ell \in \{1,..,L\}$:

- Select k indices $\mathbb{I}_\ell = \{i_1,..,i_k\} \subset \mathbb{S}_{\ell-1}$ to construct the corresponding rotation matrix $\boldsymbol{U}_\ell$ (see Section 4).
- Select the set of indices $\mathbb{T}_\ell \subset \mathbb{S}_{\ell-1}$ of rows/columns that are to be wavelets at this level, and then be eliminated by setting $\mathbb{S}_\ell = \mathbb{S}_{\ell-1} \setminus \mathbb{T}_\ell$. To reduce the computational cost, we assume that each resolution level has only one row/column to be selected as the wavelet (e.g., a single wavelet) such that $|\mathbb{T}_\ell| = 1$. That means the cardinality of $\mathbb{S}_\ell$ reduces by 1 after each level, $d_\ell = n - \ell$, and size of the core block of $\boldsymbol{H}$ is $(n-L)\times(n-L)$ that corresponds to exactly $n - L$ active rows/columns at the end.

5.2. Markov Decision Process

A key task for building our model is to specify our index selection procedure. We design an iterative index selection process and formulate it as a general decision process $M = (S, A, P, R, \gamma)$ as follows.

S is the set of states (or state space) that consists of all possible intermediate and final states in which each state $s \in S$ is a tuple of $(\overline{\boldsymbol{A}}, \mathbb{S}, \ell)$ where ℓ indicates the resolution level, $\mathbb{S}$ indicates the set of active row/column indices, and $\overline{\boldsymbol{A}} = \boldsymbol{A}_{\mathbb{S},\mathbb{S}}$ indicates the sub-matrix of $\boldsymbol{A}$ with indices of rows and columns are from $\mathbb{S}$ (e.g., $|\mathbb{S}| = d_\ell$, $\overline{\boldsymbol{A}} \in \mathbb{R}^{d_\ell\times d_\ell}$). We start at state $s_0 = (\boldsymbol{A}, [n], 0)$ where $\boldsymbol{A}$ is the input matrix that MMF tries to factorize (e.g., no rows/columns removed yet) and $[n]$ indicates all rows/columns are still active. The set of terminal (final) states $S^* \subset S$ includes every state s^* that has $\ell = L$.

A is the set of actions that describe the modification made to current state at each time step. An action $a \in A$ validly applied to a non-terminal state $s = (\boldsymbol{A}_{\mathbb{S},\mathbb{S}}, \mathbb{S}, \ell)$ is a tuple $(\mathbb{I}, \mathbb{T})$ where $\mathbb{I} = \{i_1,..,i_k\} \subset \mathbb{S}$ is the set of k indices corresponding to the rotation matrix $\boldsymbol{U}_{\ell+1}$, and $\mathbb{T} \subset \mathbb{S}$ is the set of wavelet indices to spit out at this level. This action transforms the state into the next one $s' = (\boldsymbol{A}_{\mathbb{S}',\mathbb{S}'}, \mathbb{S}', \ell+1)$ where $\mathbb{S}' = \mathbb{S} \setminus \mathbb{T}$ meaning the set of active indices gets shrinked further. The action is called invalid for the current state if and only if $\mathbb{I} \notin \mathbb{S}$ or $\mathbb{T} \notin \mathbb{S}$.

P is the transition dynamics that specifies the possible outcomes of carrying out an action at time t in which $p(a_\ell|s_\ell,..,s_0)$ is represented as a parameterized policy network π_θ with learnable parameters θ. The whole trajectory is always started by the same s_0 and finished by a terminal state after exactly L transitions:

$$s_0 \xrightarrow{a\sim\pi_\theta(\cdot|s_0)} s_1 \ldots s_{L-1} \xrightarrow{a\sim\pi_\theta(\cdot|s_{L-1})} s_L \in S^*,$$

that constructs the nested sequence.

$R(s^*)$ is the reward function that specifies the reward after reaching a terminal state s^*. The reward function is defined as negative of the MMF reconstruction loss such that

$$R(s^*) = -\|\boldsymbol{A} - \boldsymbol{U}_1^T\ldots\boldsymbol{U}_L^T\boldsymbol{H}\boldsymbol{U}_L\ldots\boldsymbol{U}_1\|_F. \tag{7}$$

We want to maximize this final reward that is equivalent to minimize error of MMF in Frobenius norm (as in problem (19)). Evaluation of the reward requires the Stiefel manifold optimization (see Section 4) for rotations $\{\boldsymbol{U}_\ell\}_{\ell=1}^L$. Obviously, the final reward in Eq. (7) is the most important. However, to improve the training quality of the policy, we can define the intermediate reward $R(s)$ for non-terminal states $s = (\boldsymbol{A}_{\mathbb{S},\mathbb{S}}, \mathbb{S}, \ell) \notin S^*$ as the immediate improvement of the ℓ-th resolution ($L > \ell > 0$):

$$R(s) = -\|[\boldsymbol{U}_\ell\boldsymbol{A}_{\ell-1}\boldsymbol{U}_\ell^T]_{\mathbb{S},\mathbb{S}}\|_{\text{resi}}^2. \tag{8}$$

Along the trajectory $(s_0, s_1, .., s_L)$, we generate the corresponding sequence of rewards $(r_1, r_2, .., r_L)$ based on (7, 8).

γ is the discount factor, a penalty to uncertainty of future rewards, $0 < \gamma \leq 1$. We define the return or discounted future reward g_ℓ for $\ell = 0, .., L-1$ as

$$g_\ell = \sum_{k=0}^{L-\ell-1} \gamma^k r_{\ell+k+1}, \tag{9}$$

which in the case of $\gamma = 1$ indicates simply accumulating all the immediate rewards and the final reward along the trajectory.

5.3. Graph convolutional policy network

In this section, we design our policy network π_θ as a graph neural network (GNN) with the message passing scheme. We consider the symmetric matrix $\boldsymbol{A}$ being represented by a weighted undirected graph $\mathcal{G} = (V, E)$ in which V is the set of nodes such that each node corresponds to a row/column of $\boldsymbol{A}$, and E is the set of edges such that the edge $(i, j) \in E$ has the weight $\boldsymbol{A}_{i,j}$. As defined in Section 5.2, a state $s \in S$ is a tuple $(\boldsymbol{A}_{\mathbb{S},\mathbb{S}}, \mathbb{S}, \ell)$ in which $\boldsymbol{A}_{\mathbb{S},\mathbb{S}}$ is the sub-matrix restricted to the active rows/columns $\mathbb{S}$, and an action $a \in A$ is a tuple $(\mathbb{I}, \mathbb{T})$ in which $\mathbb{I}$ is the set of k indices corresponding to the $(\ell+1)$-th rotation and $\mathbb{T}$ is the set of indices to spit out as wavelets. Practically, a state can be simply represented by a single binary vector such that if a bit is 1 then the corresponding index is active, without the need to explicitly storage matrix $\boldsymbol{A}_{\mathbb{S},\mathbb{S}}$ that can be efficiently constructed from $\boldsymbol{A}$ by any numerical toolkit. Our GNN policy network $\pi_\theta(a|s)$ learns to encode the underlying graph represented by $\boldsymbol{A}_{\mathbb{S},\mathbb{S}}$ and returns a sample of valid action such that $\mathbb{T} \subset \mathbb{I} \subset \mathbb{S}$. In Section 5.1, we assume that $\mathbb{T}$ contains only a single index that we will call as the *pivot* i^* (e.g., $\mathbb{T} = \{i^*\}$). Thus, the task of our GNN model is to learn to select the pivot i^* first, and then select the rest $k-1$ indices of $\mathbb{I}$ that are highly correlated to the pivot.

The simplest implementation of GNNs is Message Passing Neural Networks (MPNNs) (Gilmer et al., 2017). Suppose that the node embeddings (messages) $\boldsymbol{M}_0 \in \mathbb{R}^{N \times F}$ are initialized by the input node features where N is the number of nodes and F is the number of features for each node. Iteratively, the messages are propagated from each node to its neighborhood, and then transformed by a combination of linear transformations and non-linearities, e.g.,

$$\hat{\boldsymbol{M}}_t = \boldsymbol{A}\boldsymbol{M}_{t-1}, \quad \boldsymbol{M}_t = \sigma(\hat{\boldsymbol{M}}_t \boldsymbol{W}_{t-1}), \tag{10}$$

where $\boldsymbol{A}$ is the adjacency matrix; $\hat{\boldsymbol{M}}_t$ and $\boldsymbol{M}_t \in \mathbb{R}^{N \times D}$ are the aggregated messages (by summing over each node's neighborhood) and the output messages at the t'th iteration, respectively; σ is a element-wise non-linearity function (e.g., sigmoid, ReLU, etc.); and $\boldsymbol{W}$s are learnable weight matrices such that $\boldsymbol{W}_0 \in \mathbb{R}^{F \times D}$ and $\boldsymbol{W}_t \in \mathbb{R}^{D \times D}$ for $t > 0$. Basically, the set of learnable parameters θ of our policy network π includes all $\boldsymbol{W}$s. In some cases, a graph Laplacian $\boldsymbol{L}$ is used instead of the adjacency matrix $\boldsymbol{A}$ in model (10), for example, graph Laplacian $\boldsymbol{L} = \boldsymbol{D}^{-1}\boldsymbol{A}$ or its symmetric normalized version $\tilde{\boldsymbol{L}} = \boldsymbol{I} - \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$. One way to incorporate the set of active rows/columns/nodes $\mathbb{S}$ into our GNN model is by initializing the input node feature with a binary label ($f = 1$) such that a node v has label 1 if the v-th row/column is still active, otherwise 0. For a more efficient implementation, we can execute the message passing in the block $\boldsymbol{A}_{\mathbb{S},\mathbb{S}}$ only. Supposing that the message passing scheme is executed for T iterations, we concatenate messages from every iteration together into the final embedding:

$$\boldsymbol{M} = \bigoplus_{t=1}^{T} \boldsymbol{M}_t \in \mathbb{R}^{N \times DT}. \tag{11}$$

Model (10) produces the embedding for each node that allows us to define a sampling procedure to select the pivot i^*. Given the final embedding $\boldsymbol{M}$ from Eq. (11), we define the probability $\boldsymbol{P}_i$ that node $i \in \mathbb{S}$ is being selected as the pivot as:

$$P_i = \frac{\exp\left(\hat{P}_i\right)}{\sum_{j \in \mathbb{S}} \exp\left(\hat{P}_j\right)}, \quad \text{where} \quad \hat{P}_i = \sum_f \boldsymbol{M}_{i,f}.$$

In order to make the sampling procedure differentiable for backpropagation, we apply the Gumbel-max trick (Gumbel, 1954) (Maddison et al., 2014) (Jang et al., 2017) that provides a simple and efficient way to draw sample i^* as follows:

$$i^* = \text{one-hot}\Big(\arg\max_{i \in \mathbb{S}} \left[G_i + \log P_i\right]\Big),$$

where G_i are i.i.d samples drawn from Gumbel$(0, 1)$. Technically, the sample is represented by a one-hot vector such that the i^*-th element is 1. Similarly, $\mathbb{T}$, $\mathbb{I}$ and $\mathbb{S}$ are represented by vectors in $\{0, 1\}^N$ in which a 1-element indicates the existence of the corresponding index in the set. Furthermore, the set union and minus operations (e.g., $\mathbb{S} \setminus \mathbb{T}$) can be easily done by vector addition and subtraction, respectively.

Given the pivot i^*, we compute the similarity score between i^* and other nodes $i \in \mathbb{S}$ as $C_i = \langle \boldsymbol{M}_{i^*,:}, \boldsymbol{M}_{i,:} \rangle$. Finally, we sample $k-1$ nodes with the highest similarity scores to i sequentially (one-by-one) without replacement by the Gumbel-max trick, that completes our sampling procedure for action $a = (\mathbb{I}, \mathbb{T})$.

The REINFORCE (Williams, 1988) (Williams, 1992) (Sutton et al., 2000) update rule for policy parameters is

$$\theta \leftarrow \theta + \eta \gamma^\ell g_\ell \nabla_\theta \log \pi_\theta(a_\ell | s_\ell) \quad \text{for} \quad \ell = 0, .., L-1; \tag{12}$$

where η is the learning rate, that is used in training our policy network π_θ in Algorithm 1 (see section *Policy gradient methods* in the Appendix).

5.4. The learning algorithm

Putting everything together, our MMF learning algorithm is sketched in Algorithm 1. Iteratively: (1) we sample a trajectory by running the policy network that indicates the indices for rotation and wavelet for each resolution, (2) we apply the Stiefel manifold optimization to find the rotations, and (3) we compute the future rewards and update the parameters of the policy network by REINFORCE accordingly. The learning terminates when we complete ω iterations or episodes. In all our experiments, the learning algorithm can get to sufficient precision in matrix approximation after $\omega = 256$ episodes.

6. Wavelet Networks on Graphs

6.1. Motivation

The eigendecomposition of the normalized graph Laplacian operator $\tilde{\boldsymbol{L}} = \boldsymbol{U}^T \boldsymbol{H} \boldsymbol{U}$ can be used as the basis of a graph Fourier transform. (Shuman et al., 2013) defines graph Fourier transform (GFT) on a graph $\mathcal{G} = (V, E)$ of a graph signal $\boldsymbol{f} \in \mathbb{R}^n$ (that is understood as a function $f : V \to \mathbb{R}$ defined on the vertices of the graph) as $\hat{\boldsymbol{f}} = \boldsymbol{U}^T \boldsymbol{f}$, and the inverse graph Fourier transform as $\boldsymbol{f} = \boldsymbol{U} \hat{\boldsymbol{f}}$. Analogously to the classical Fourier transform, GFT provides a way to represent a graph signal in two domains: the vertex domain and the graph spectral domain; to filter graph signal according to smoothness; and to define the graph convolution operator, denoted as $*_{\mathcal{G}}$:

$$\boldsymbol{f} *_{\mathcal{G}} \boldsymbol{g} = \boldsymbol{U}\big((\boldsymbol{U}^T \boldsymbol{g}) \odot (\boldsymbol{U}^T \boldsymbol{f})\big), \tag{13}$$

where $\boldsymbol{g}$ denotes the convolution kernel, and $\odot$ is the element-wise Hadamard product. If we replace the vector $\boldsymbol{U}^T \boldsymbol{g}$ by a diagonal matrix $\tilde{\boldsymbol{g}}$, then we can rewrite the Hadamard product in Eq. (13) to matrix multiplication as $\boldsymbol{U} \tilde{\boldsymbol{g}} \boldsymbol{U}^T \boldsymbol{f}$ (that is understood as filtering the signal $\boldsymbol{f}$ by the filter $\tilde{\boldsymbol{g}}$). Based on GFT, (Bruna et al., 2014) and (Defferrard et al., 2016) construct convolutional neural networks (CNNs) learning on spectral domain for discrete structures such as graphs. However, there are two fundamental limitations of GFT:

- High computational cost: eigendecomposition of the graph Laplacian has complexity $O(n^3)$, and "Fourier transform" itself involves multiplying the signal with a dense matrix of eigenvectors.
- The graph convolution is not localized in the vertex domain, even if the graph itself has well defined local communities.

To address these limitations, we propose a modified spectral graph network based on the MMF wavelet basis rather than the eigenbasis of the Laplacian. This has the following advantages: (i) the wavelets are generally localized in both vertex domain and frequency, (ii) the individual basis transforms are sparse, and (iii) MMF provides a computationally efficient way of decomposing graph signals into components at different granularity levels and an excellent basis for sparse approximations.

6.2. Network construction

In the case $\boldsymbol{A}$ is the normalized graph Laplacian of a graph $\mathcal{G} = (V, E)$, the wavelet transform (up to level L) expresses a graph signal (function over the vertex domain) $f : V \to \mathbb{R}$, without loss of generality $f \in \mathbb{V}_0$, as:

$$f(v) = \sum_{\ell=1}^{L} \sum_m \alpha_m^\ell \psi_m^\ell(v) + \sum_m \beta_m \phi_m^L(v), \quad \text{for each } v \in V,$$

where $\alpha_m^\ell = \langle f, \psi_m^\ell \rangle$ and $\beta_m = \langle f, \phi_m^L \rangle$ are the wavelet coefficients. At each level, a set of coordinates $\mathbb{T}_\ell \subset \mathbb{S}_{\ell-1}$ are selected to be the wavelet indices, and then to be eliminated from the active set by setting $\mathbb{S}_\ell = \mathbb{S}_{\ell-1} \setminus \mathbb{T}_\ell$ (see Section 5.1). Practically, we make the assumption that we only select 1 wavelet index for each level (see Section 5.1) that results in a single mother wavelet $\psi^\ell = [\boldsymbol{A}_\ell]_{i^*,:}$ where i^* is the selected index (see Section 5.3). We get exactly L mother wavelets $\overline{\psi} = \{\psi^1, \psi^2, \ldots, \psi^L\}$. On the another hand, the active rows of $\boldsymbol{H} = \boldsymbol{A}_L$ make exactly $N - L$ father wavelets $\overline{\phi} = \{\phi_m^L = \boldsymbol{H}_{m,:}\}_{m \in \mathbb{S}_L}$. In total, a graph of N vertices has exactly N wavelets (both mothers and fathers). Analogous to the convolution based on GFT (Bruna et al., 2014), each convolution layer $k = 1, .., K$ of our wavelet network transforms an input vector $\boldsymbol{f}^{(k-1)}$ of size $|V| \times F_{k-1}$ into an output $\boldsymbol{f}^{(k)}$ of size $|V| \times F_k$ as

$$\boldsymbol{f}_{:,j}^{(k)} = \sigma\left(\boldsymbol{W} \sum_{i=1}^{F_{k-1}} \boldsymbol{g}_{i,j}^{(k)} \boldsymbol{W}^T \boldsymbol{f}_{:,i}^{(k-1)}\right) \quad \text{for } j = 1, \ldots, F_k, \tag{14}$$

where $\boldsymbol{W}$ is our wavelet basis matrix as we concatenate $\overline{\phi}$ and $\overline{\psi}$ column-by-column, $\boldsymbol{g}_{i,j}^{(k)}$ is a parameter/filter in the form of a diagonal matrix learned in spectral domain, and σ is an element-wise linearity (e.g., ReLU, sigmoid, etc.).

For example, in node classification tasks, assume the number of classes is C, the set of labeled nodes is V_{label}, and we are given a normalized graph Laplacian $\tilde{L}$ and an input node feature matrix $\boldsymbol{f}^{(0)}$. First of all, we apply our MMF learning algorithm 1 to factorize $\tilde{L}$ and produce our wavelet basis matrix $\boldsymbol{W}$. Then, we construct our wavelet network as a multi-layer CNNs with each convolution is defined as in Eq. (14) that transforms $\boldsymbol{f}^{(0)}$ into $\boldsymbol{f}^{(K)}$ after K layers. The top convolution layer K-th returns exactly $F_K = C$ features and uses softmax instead of the nonlinearity σ for each node.

Algorithm 1 MMF learning algorithm optimizing problem (19)

Input: Matrix $\boldsymbol{A}$, number of resolutions L, and constants k, γ, η, and ω.
Initialize the policy parameter θ at random.
while true **do**
  Start from state s_0 ▷ $s_0 \triangleq (\boldsymbol{A}, [n], 0)$
  Initialize $\mathbb{S}_0 \leftarrow [n]$ ▷ All rows/columns are active at the beginning.
  for $\ell = 0, .., L-1$ **do**
    Sample action $a_\ell = (\mathbb{I}_{\ell+1}, \mathbb{T}_{\ell+1})$ from $\pi_\theta(a_\ell|s_\ell)$. ▷ See Section 5.3.
    $\mathbb{S}_{\ell+1} \leftarrow \mathbb{S}_\ell \setminus \mathbb{T}_{\ell+1}$ ▷ Eliminate the wavelet index (indices).
    $s_{\ell+1} \leftarrow (\boldsymbol{A}_{\mathbb{S}_{\ell+1},\mathbb{S}_{\ell+1}}, \mathbb{S}_{\ell+1}, \ell+1)$ ▷ New state with a smaller active set.
  end for
  Given $\{\mathbb{I}_\ell\}_{\ell=1}^L$, minimize objective (6) by Stiefel manifold optimization to find $\{\boldsymbol{O}_\ell\}_{\ell=1}^L$. ▷ $\boldsymbol{U}_\ell = \boldsymbol{I}_{n-k} \oplus_{\mathbb{I}_\ell} \boldsymbol{O}_\ell$
  for $\ell = 0, .., L-1$ **do**
    Estimate the return g_ℓ based on Eq. (7), Eq. (8), and Eq. (9).
    $\theta \leftarrow \theta + \eta\gamma^\ell g_\ell \nabla_\theta \log \pi_\theta(a_\ell|s_\ell)$ ▷ REINFORCE policy update in Eq. (12)
  end for
  Terminate if we already completed ω episodes.
end while

The loss is the cross-entropy error over all labeled nodes as:

$$\mathcal{L} = -\sum_{v \in V_{\text{label}}} \sum_{c=1}^{C} \boldsymbol{y}_{v,c} \ln \boldsymbol{f}_{v,c}^{(K)}, \tag{15}$$

where $\boldsymbol{y}_{v,c}$ is a binary indicator that is equal to 1 if node v is labeled with class c, and 0 otherwise. The set of weights $\{\boldsymbol{g}^{(k)}\}_{k=1}^K$ are trained using gradient descent optimizing the loss in Eq. (15).

7. Experiments

7.1. Matrix factorization

We evaluate the performance of our MMF learning algorithm in comparison with the original greedy algorithm (Kondor et al., 2014) and the Nyström method (Gittens & Mahoney, 2013) in the task of matrix factorization on 3 datasets: (i) normalized graph Laplacian of the Karate club network ($N = 34$, $E = 78$) (Zachary, 1976); (ii) a Kronecker product matrix ($N = 512$), $\mathcal{K}_1^n$, of order $n = 9$, where $\mathcal{K}_1 = ((0,1),(1,1))$ is a 2×2 seed matrix (Leskovec et al., 2010); and (iii) normalized graph Laplacian of a Cayley tree or Bethe lattice with coordination number $z = 4$ and 4 levels of depth ($N = 161$). The rotation matrix size K are 8, 16 and 8 for Karate, Kronecker and Cayley, respectively. Meanwhile, the original greedy MMF is limited to $K = 2$ and implements an exhaustive search to find an optimal pair of indices for each rotation. For both versions of MMF, we drop $c = 1$ columns after each rotation, which results in a final core size of $d_L = N - c \times L$. The exception is for the Kronecker matrix ($N = 512$), our learning algorithm drops up to 8 columns (for example, $L = 62$ and $c = 8$ results into $d_L = 16$) to make sure that the number of learnable parameters $L \times K^2$ is much smaller the matrix size N^2. Our learning algorithm compresses the Kronecker matrix down to $6-7\%$ of its original size. The details of efficient training reinforcement learning with the policy networks implemented by GNNs are included in the Appendix.

For the baseline of Nyström method, we randomly select, by uniform sampling without replacement, the same number d_L columns $\boldsymbol{C}$ from $\boldsymbol{A}$ and take out $\boldsymbol{W}$ as the corresponding $d_L \times d_L$ submatrix of $\boldsymbol{A}$. The Nyström method approximates $\boldsymbol{A} \approx \boldsymbol{C}\boldsymbol{W}^\dagger\boldsymbol{C}^T$. We measure the approximation error in Frobenius norm. Figure 1 shows our MMF learning algorithm consistently outperforms the original greedy algorithm and the Nyström baseline given the same number of active columns, d_L. Figure 2 depicts the wavelet bases at different levels of resolution.

7.2. Molecular graphs classification

We trained and evaluated our wavelet networks (WNNs) on standard graph classification benchmarks including four bioinformatics datasets: (1) MUTAG, which is a dataset of 188 mutagenic aromatic and heteroaromatic nitro compounds with 7 discrete labels (Debnath et al., 1991); (2) PTC, which consists of 344 chemical compounds with 19 discrete labels that have been tested for positive or negative toxicity in lab rats (Toivonen et al., 2003); (3) PROTEINS, which contains 1,113 molecular graphs with binary labels, where nodes are secondary structure elements (SSEs) and there is an edge between two nodes if they are neighbors in the amino-acid sequence or in 3D space (Borgwardt et al., 2005); (4) NCI1, which has 4,110 compounds with binary labels, each screened for activity against small cell lung cancer and ovarian cancer lines (Wale et al., 2008). Each molecule is represented by an adjacency matrix, and we represent each atomic type as a one-hot vector and use them

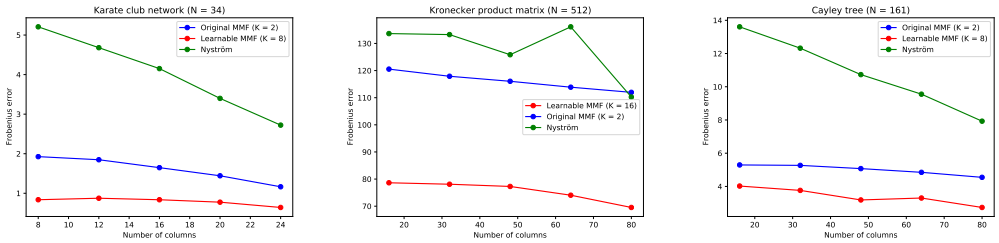
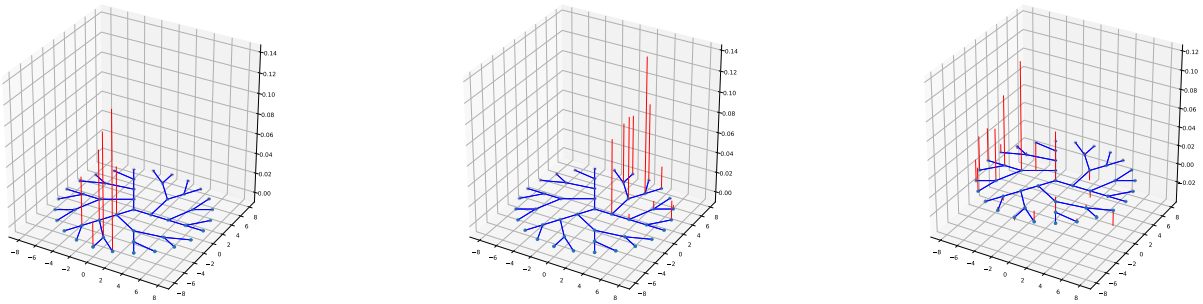


Figure 1. Matrix factorization for the Karate network (left), Kronecker matrix (middle), and Cayley tree (right). Our learnable MMF consistently outperforms the classic greed methods.



$\ell = 1$ $\ell = 20$ $\ell = 39$

Figure 2. Visualization of some of the wavelets on the Cayley tree of 46 vertices. The low index wavelets (low ℓ) are highly localized, whereas the high index ones are smoother and spread out over large parts of the graph.

as the node features.

We factorize all normalized graph Laplacian matrices in these datasets by MMF with $K = 2$ to obtain the wavelet bases. Again, MMF wavelets are **sparse** and suitable for fast transform via sparse matrix multiplication, with the following average percentages of non-zero elements for each dataset: 19.23% (MUTAG), 18.18% (PTC), 2.26% (PROTEINS) and 11.43% (NCI1).

Our WNNs contain 6 layers of spectral convolution, 32 hidden units for each node, and are trained with 256 epochs by Adam optimization with an initial learning rate of 10^{-3}. We follow the evaluation protocol of 10-fold cross-validation from (Zhang et al., 2018). We compare our results to several deep learning methods and popular graph kernel methods. Baseline results are taken from (Maron et al., 2019). Our WNNs outperform 7/8, 7/8, 8/8, and 2/8 baseline methods on MUTAG, PTC, PROTEINS, and NCI1, respectively (see Table 1).

7.3. Node classification on citation graphs

To further evaluate the wavelet bases returned by our learnable MMF algorithm, we construct our wavelet networks (WNNs) as in Sec. 6 and apply it to the task of node classification on two citation graphs, Cora ($N = 2,708$) and Citeseer ($N = 3,312$) (Sen et al., 2008) in which nodes and edges represent documents and citation links. Each document in Cora and Citeseer has an associated feature vector (of length $1,433$ resp. $3,703$) computed from word frequencies, and is classified into one of 7 and 6 classes, respectively. We factorize the normalized graph Laplacian by learnable MMF with $K = 16$ to obtain the wavelet bases. The resulting MMF wavelets are sparse, which makes it possible to run a fast transform on the node features by sparse matrix multiplication: only 4.69% and 15.25% of elements are non-zero in Citeseer and Cora, respectively. In constrast, Fourier bases given by eigendecomposition of the graph Laplacian are completely dense (100% of elements are non-zero). We evaluate our WNNs with 3 different random splits of train/validation/test: (1) $20\%/20\%/60\%$ denoted as MMF_1, (2) $40\%/20\%/40\%$ denoted as MMF_2, and (3) $60\%/20\%/20\%$ denoted as MMF_3. The WNN learns to encode the whole graph with 6 layers of spectral convolution and 100 hidden dimensions for each node. During training, the network is only trained to predict the node labels in the training set. Hyperparameter searching is done on the validation set. The number of epochs is 256 and we use the Adam optimization method (Kingma & Ba, 2015) with learning rate $\eta = 10^{-3}$. We report the final test accuracy for each split in Table 2.

We compare with several traditional methods and deep learning methods including other spectral graph convolution net-

Table 1. Molecular graphs classification. Baseline results are taken from (Maron et al., 2019).

Method	MUTAG	PTC	PROTEINS	NCI1
DGCNN (Zhang et al., 2018)	85.83 ± 1.7	58.59 ± 2.5	75.54 ± 0.9	74.44 ± 0.5
PSCN (Niepert et al., 2016)	88.95 ± 4.4	62.29 ± 5.7	75 ± 2.5	76.34 ± 1.7
DCNN (Atwood & Towsley, 2016)	N/A	N/A	61.29 ± 1.6	56.61 ± 1.0
CCN (Hy et al., 2018)	**91.64 ± 7.2**	**70.62 ± 7.0**	N/A	76.27 ± 4.1
GK (Shervashidze et al., 2009)	81.39 ± 1.7	55.65 ± 0.5	71.39 ± 0.3	62.49 ± 0.3
RW (Vishwanathan et al., 2010)	79.17 ± 2.1	55.91 ± 0.3	59.57 ± 0.1	N/A
PK (Neumann et al., 2016)	76 ± 2.7	59.5 ± 2.4	73.68 ± 0.7	82.54 ± 0.5
WL (Shervashidze et al., 2011)	84.11 ± 1.9	57.97 ± 2.5	74.68 ± 0.5	**84.46 ± 0.5**
IEGN (Maron et al., 2019)	84.61 ± 10	59.47 ± 7.3	75.19 ± 4.3	73.71 ± 2.6
MMF	86.31 ± 9.47	67.99 ± 8.55	**78.72 ± 2.53**	71.04 ± 1.53

Table 2. Node classification on citation graphs. Baseline results are taken from (Xu et al., 2019).

Method	Cora	Citeseer
MLP	55.1%	46.5%
ManiReg (Belkin et al., 2006)	59.5%	60.1%
SemiEmb (Weston et al., 2008)	59.0%	59.6%
LP (Zhu et al., 2003)	68.0%	45.3%
DeepWalk (Perozzi et al., 2014)	67.2%	43.2%
ICA (Getoor, 2005)	75.1%	69.1%
Planetoid (Yang et al., 2016)	75.7%	64.7%
Spectral CNN (Bruna et al., 2014)	73.3%	58.9%
ChebyNet (Defferrard et al., 2016)	81.2%	69.8%
GCN (Kipf & Welling, 2017)	81.5%	70.3%
MoNet (Monti et al., 2017)	81.7%	N/A
GWNN (Xu et al., 2019)	82.8%	71.7%
MMF_1	**84.35%**	68.07%
MMF_2	**84.55%**	**72.76%**
MMF_3	**87.59%**	**72.90%**

works such as Spectral CNN, and graph wavelet neural networks (GWNN). Baseline results are taken from (Xu et al., 2019). Our wavelet networks perform competitively against state-of-the-art methods in the field.

8. Conclusions

In this paper we introduced a general algorithm based on reinforcement learning and Stiefel manifold optimization to optimize Multiresolution Matrix Factorization (MMF). We find that the resulting learnable MMF consistently outperforms the existing greedy and heuristic MMF algorithms in factorizing and approximating hierarchical matrices. Based on the wavelet basis returned from our learning algorithm, we define a corresponding notion of spectral convolution and construct a wavelet neural network for graph learning problems. Thanks to the sparsity of the MMF wavelets, the wavelet network can be efficiently implemented with sparse matrix multiplication. We find that this combination of learnable MMF factorization and spectral wavelet network yields state of the art results on standard node classification and molecular graph classification.

References

Atwood, J. and Towsley, D. Diffusion-convolutional neural networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pp. 20012009, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.

Belkin, M., Niyogi, P., and Sindhwani, V. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7(85):2399–2434, 2006. URL http://jmlr.org/papers/v7/belkin06a.html.

Borgwardt, K. M., Ong, C. S., Schönauer, S., Vishwanathan, S. V. N., Smola, A., and Kriegel, H.-P. Protein function prediction via graph kernels. *Bioinformatics*, 21 Suppl 1: i47–56, 2005.

Bruna, J., Zaremba, W., Szlam, A., and Lecun, Y. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations (ICLR2014), CBLS, April 2014*, 2014.

Coifman, R. R. and Maggioni, M. Diffusion wavelets. *Applied and Computational Harmonic Analysis*, 21(1):53–94, 2006. ISSN 1063-5203. doi: https://doi.org/10.1016/j.acha.2006.04.004. URL https://www.sciencedirect.com/science/article/pii/S106352030600056X. Special Issue: Diffusion Maps and Wavelets.

Debnath, A. K., Lopez de Compadre, R. L., Debnath, G., Shusterman, A. J., and Hansch, C. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of Medicinal Chemistry*, 34 (2):786–797, 1991. doi: 10.1021/jm00106a046. URL https://doi.org/10.1021/jm00106a046.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized

spectral filtering. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/04df4d434d481c5bb723be1b6df1ee65-Paper.pdf.

Ding, Y., Kondor, R., and Eskreis-Winkler, J. Multiresolution kernel approximation for gaussian process regression. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/850af92f8d9903e7a4e0559a98ecc857-Paper.pdf.

Drineas, P., Kannan, R., and Mahoney, M. W. Fast monte carlo algorithms for matrices ii: Computing a low-rank approximation to a matrix. *SIAM J. Comput.*, 36:158–183, 2006.

Gavish, M., Nadler, B., and Coifman, R. R. Multiscale wavelets on trees, graphs and high dimensional data: Theory and applications to semi supervised learning. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pp. 367374, Madison, WI, USA, 2010. Omnipress. ISBN 9781605589077.

Getoor, L. *Link-based Classification*, pp. 189–207. Springer London, London, 2005. ISBN 978-1-84628-284-3. doi: 10.1007/1-84628-284-5_7. URL https://doi.org/10.1007/1-84628-284-5_7.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1263–1272. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/gilmer17a.html.

Gittens, A. and Mahoney, M. Revisiting the nystrom method for improved large-scale machine learning. In Dasgupta, S. and McAllester, D. (eds.), *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pp. 567–575, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. URL https://proceedings.mlr.press/v28/gittens13.html.

Gumbel, E. J. Statistical theory of extreme values and some practical applications: a series of lectures. *US Govt. Print. Office*, Number 33, 1954.

Halko, N., Martinsson, P. G., and Tropp, J. A. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, 2011. doi: 10.1137/090771806. URL https://doi.org/10.1137/090771806.

Hammond, D. K., Vandergheynst, P., and Gribonval, R. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011. ISSN 1063-5203. doi: https://doi.org/10.1016/j.acha.2010.04.005. URL https://www.sciencedirect.com/science/article/pii/S1063520310000552.

Hy, T. S., Trivedi, S., Pan, H., Anderson, B. M., , and Kondor, R. Predicting molecular properties with covariant compositional networks. *The Journal of Chemical Physics*, 148, 2018.

Ithapu, V. K., Kondor, R., Johnson, S. C., and Singh, V. The incremental multiresolution matrix factorization algorithm. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 692–701, 2017. doi: 10.1109/CVPR.2017.81.

Jacobi, C. ber ein leichtes verfahren die in der theorie der scularstrungen vorkommenden gleichungen numerisch aufzulsen*).:. 1846(30):51–94, 1846. doi: doi: 10.1515/crll.1846.30.51. URL https://doi.org/10.1515/crll.1846.30.51.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. In *ICLR*, 2017.

Jenatton, R., Obozinski, G., and Bach, F. Structured sparse principal component analysis. In Teh, Y. W. and Titterington, M. (eds.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pp. 366–373, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL https://proceedings.mlr.press/v9/jenatton10a.html.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *Proc. ICLR*, San Diego, 2015.

Kipf, T. N. and Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR '17, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Kondor, R., Teneva, N., and Garg, V. Multiresolution matrix factorization. In Xing, E. P. and Jebara, T. (eds.), *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pp. 1620–1628, Bejing, China, 22–24 Jun 2014. PMLR. URL https://proceedings.mlr.press/v32/kondor14.html.

Kumar, S., Mohri, M., and Talwalkar, A. Sampling methods for the nystrom method. *Journal of Machine Learning Research*, 13(34):981–1006, 2012. URL http://jmlr.org/papers/v13/kumar12a.html.

Lee, A. B., Nadler, B., and Wasserman, L. TreeletsAn adaptive multi-scale basis for sparse unordered data. *The Annals of Applied Statistics*, 2(2):435 – 471, 2008. doi: 10.1214/07-AOAS137. URL https://doi.org/10.1214/07-AOAS137.

Leskovec, J., Chakrabarti, D., Kleinberg, J., Faloutsos, C., and Ghahramani, Z. Kronecker graphs: An approach to modeling networks. *Journal of Machine Learning Research*, 11(33):985–1042, 2010. URL http://jmlr.org/papers/v11/leskovec10a.html.

Maddison, C. J., Tarlow, D., and Minka, T. A* sampling. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

Mahoney, M. W. Randomized algorithms for matrices and data. *Found. Trends Mach. Learn.*, 3(2):123224, February 2011. ISSN 1935-8237. doi: 10.1561/2200000035. URL https://doi.org/10.1561/2200000035.

Mallat, S. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, 1989. doi: 10.1109/34.192463.

Maron, H., Ben-Hamu, H., Shamir, N., and Lipman, Y. Invariant and equivariant graph networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Syx72jC9tm.

Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., and Bronstein, M. M. Geometric deep learning on graphs and manifolds using mixture model cnns. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5425–5434, Los Alamitos, CA, USA, jul 2017. IEEE Computer Society. doi: 10.1109/CVPR.2017.576.

Neumann, M., Garnett, R., Baukhage, C., and Kersting, K. Propagation kernels: Efficient graph kernels from propagated information. *Machine Learning*, 102:209–245, 2 2016.

Niepert, M., Ahmed, M., and Kutzkov, K. Learning convolutional neural networks for graphs. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 2014–2023, New York, New York, USA, 20–22 Jun 2016. PMLR.

Nocedal, J. and Wright, S. J. *Numerical Optimization*. Springer, New York, NY, USA, second edition, 2006.

Perozzi, B., Al-Rfou, R., and Skiena, S. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pp. 701710, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450329569. doi: 10.1145/2623330.2623732. URL https://doi.org/10.1145/2623330.2623732.

Sen, P., Namata, G. M., Bilgic, M., Getoor, L., Gallagher, B., , and Eliassi-Rad, T. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.

Shervashidze, N., Vishwanathan, S., Petri, T., Mehlhorn, K., and Borgwardt, K. Efficient graphlet kernels for large graph comparison. In van Dyk, D. and Welling, M. (eds.), *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pp. 488–495, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 16–18 Apr 2009. PMLR. URL https://proceedings.mlr.press/v5/shervashidze09a.html.

Shervashidze, N., Schweitzer, P., van Leeuwen, E. J., Mehlhorn, K., and Borgwardt, K. M. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(77):2539–2561, 2011.

Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., and Vandergheynst, P. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98, 2013. doi: 10.1109/MSP.2012.2235192.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018. ISBN 0262039249.

Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In Solla, S., Leen, T., and Müller, K. (eds.), *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 2000. URL https://proceedings.neurips.cc/paper/1999/file/

464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf.

Tagare, H. Notes on optimization on stiefel manifolds. 2011.

Teneva, N., Mudrakarta, P. K., and Kondor, R. Multiresolution matrix compression. In Gretton, A. and Robert, C. C. (eds.), *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pp. 1441–1449, Cadiz, Spain, 09–11 May 2016. PMLR. URL https://proceedings.mlr.press/v51/teneva16.html.

Toivonen, H., Srinivasan, A., King, R. D., Kramer, S., and Helma, C. Statistical evaluation of the Predictive Toxicology Challenge 20002001. *Bioinformatics*, 19(10):1183–1193, 07 2003. ISSN 1367-4803. doi: 10.1093/bioinformatics/btg130. URL https://doi.org/10.1093/bioinformatics/btg130.

Vishwanathan, S. V. N., Schraudolph, N. N., Kondor, R., and Borgwardt, K. M. Graph kernels. *J. Mach. Learn. Res.*, 11:12011242, August 2010. ISSN 1532-4435.

Wale, N., Watson, I., and Karypis, G. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowl. Inf. Syst.*, 14:347–375, 03 2008. doi: 10.1109/ICDM.2006.39.

Wen, Z. and Yin, W. A feasible method for optimization with orthogonality constraints. *Mathematical Programming*, 142, 12 2010. doi: 10.1007/s10107-012-0584-1.

Weston, J., Ratle, F., and Collobert, R. Deep learning via semi-supervised embedding. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pp. 11681175, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605582054. doi: 10.1145/1390156.1390303. URL https://doi.org/10.1145/1390156.1390303.

Williams, C. and Seeger, M. Using the nyström method to speed up kernel machines. In Leen, T., Dietterich, T., and Tresp, V. (eds.), *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2001. URL https://proceedings.neurips.cc/paper/2000/file/19de10adbaa1b2ee13f77f679fa1483a-Paper.pdf.

Williams, R. J. Toward a theory of reinforcement-learning connectionist systems. Technical Report NU-CCS-88-3, Northeastern University, College of Computer Science, 1988.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(34):229256, May 1992. ISSN 0885-6125. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.

Xu, B., Shen, H., Cao, Q., Qiu, Y., and Cheng, X. Graph wavelet neural network. In *International Conference on Learning Representations*, 2019.

Yang, Z., Cohen, W. W., and Salakhutdinov, R. Revisiting semi-supervised learning with graph embeddings. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pp. 4048. JMLR.org, 2016.

Zachary, W. An information flow model for conflict and fission in small groups1. *Journal of anthropological research*, 33, 11 1976. doi: 10.1086/jar.33.4.3629752.

Zhang, M., Cui, Z., Neumann, M., and Chen, Y. An end-to-end deep learning architecture for graph classification. In *AAAI*, 2018.

Zhu, X., Ghahramani, Z., and Lafferty, J. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, pp. 912919. AAAI Press, 2003. ISBN 1577351894.

A. Notation

We define $[n] = \{1, 2, \ldots, n\}$ as the set of the first n natural numbers. We denote $\boldsymbol{I}_n$ as the n dimensional identity matrix. The group of n dimensional orthogonal matrices is $\mathbb{SO}(n)$. $\mathbb{A} \uplus \mathbb{B}$ will denote the disjoint union of two sets $\mathbb{A}$ and $\mathbb{B}$, therefore $\mathbb{A}_1 \uplus \mathbb{A}_2 \uplus \cdots \uplus \mathbb{A}_k = \mathbb{S}$ is a partition of $\mathbb{S}$.

Given a matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ and two sequences of indices $\boldsymbol{i} = (i_1, \ldots, i_k) \in [n]^k$ and $\boldsymbol{j} = (j_1, \ldots, j_k) \in [n]^k$ assuming that $i_1 < i_2 < \cdots < i_k$ and $j_1 < j_2 < \cdots < j_k$, $\boldsymbol{A}_{\boldsymbol{i},\boldsymbol{j}}$ will be the $k \times k$ matrix with entries $[\boldsymbol{A}_{\boldsymbol{i},\boldsymbol{j}}]_{x,y} = \boldsymbol{A}_{i_x, j_y}$. Furthermore, $\boldsymbol{A}_{i,:}$ and $\boldsymbol{A}_{:,j}$ denote the i-th row and the j-th column of $\boldsymbol{A}$, respectively. Given $\boldsymbol{A}_1 \in \mathbb{R}^{n_1 \times m_1}$ and $\boldsymbol{A}_2 \in \mathbb{R}^{n_2 \times m_2}$, $\boldsymbol{A}_1 \oplus \boldsymbol{A}_2$ is the $(n_1 + n_2) \times (m_1 + m_2)$ dimensional matrix with entries

$$[\boldsymbol{A}_1 \oplus \boldsymbol{A}_2]_{i,j} = \begin{cases} [\boldsymbol{A}_1]_{i,j} & \text{if } \ i \leq n_1 \ \text{ and } \ j \leq m_1 \\ [\boldsymbol{A}_2]_{i-n_1, j-m_1} & \text{if } \ i > n_1 \ \text{ and } \ j > m_1 \\ 0 & \text{otherwise.} \end{cases}$$

A matrix $\boldsymbol{A}$ is said to be block diagonal if it is of the form

$$\boldsymbol{A} = \boldsymbol{A}_1 \oplus \boldsymbol{A}_2 \oplus \cdots \oplus \boldsymbol{A}_p \tag{16}$$

for some sequence of smaller matrices $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_p$. For the generalized block diagonal matrix, we remove the restriction that each block in (16) must involve a contiguous set of indices, and introduce the notation

$$\boldsymbol{A} = \oplus_{(i_1^1, \ldots, i_{k_1}^1)} \boldsymbol{A}_1 \oplus_{(i_1^2, \ldots, i_{k_2}^2)} \boldsymbol{A}_2 \cdots \oplus_{(i_1^p, \ldots, i_{k_p}^p)} \boldsymbol{A}_p$$

in which

$$\boldsymbol{A}_{a,b} = \begin{cases} [\boldsymbol{A}_u]_{q,r} & \text{if } \ i_q^u = a \ \text{ and } \ i_r^u = b \ \text{ for some } \ u, q, r, \\ 0 & \text{otherwise.} \end{cases}$$

The Kronecker tensor product $\boldsymbol{A}_1 \otimes \boldsymbol{A}_2$ is an $n_1 n_2 \times m_1 m_2$ matrix whose elements are

$$[\boldsymbol{A}_1 \otimes \boldsymbol{A}_2]_{(i_1-1)n_2+i_2, (j_1-1)m_2+j_2} = [\boldsymbol{A}_1]_{i_1,j_1} \cdot [\boldsymbol{A}_2]_{i_2,j_2},$$

with the obvious generalization to p-fold products $\boldsymbol{A}_1 \otimes \boldsymbol{A}_2 \otimes \cdots \otimes \boldsymbol{A}_p$. We denote $\boldsymbol{A}^{\otimes p}$ as the p-fold product $\boldsymbol{A} \otimes \boldsymbol{A} \otimes \cdots \otimes \boldsymbol{A}$.

A matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is called skew-symmetric (or anti-symmetric) if $\boldsymbol{A}^T = -\boldsymbol{A}$. The Euclidean inner product between two matrices $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{B} \in \mathbb{R}^{m \times n}$ is defined as

$$\langle \boldsymbol{A}, \boldsymbol{B} \rangle = \sum_{j,k} \boldsymbol{A}_{j,k} \boldsymbol{B}_{j,k} = \text{trace}(\boldsymbol{A}^T \boldsymbol{B}).$$

The Frobenius norm of $\boldsymbol{A}$ is defined as $\|\boldsymbol{A}\|_F = \sqrt{\sum_{i,j} \boldsymbol{A}_{i,j}^2}$.

B. Multiresolution Matrix Factorization

B.1. Background

Most commonly used matrix factorization algorithms, such as principal component analysis (PCA), singular value decomposition (SVD), or non-negative matrix factorization (NMF) are inherently single-level algorithms. Saying that a symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is of rank $r \ll n$ means that it can be expressed in terms of a dictionary of r mutually orthogonal unit vectors $\{u_1, u_2, \ldots, u_r\}$ in the form

$$\boldsymbol{A} = \sum_{i=1}^{r} \lambda_i u_i u_i^T,$$

where $u_1, \ldots, u_r$ are the normalized eigenvectors of A and $\lambda_1, \ldots, \lambda_r$ are the corresponding eigenvalues. This is the decomposition that PCA finds, and it corresponds to factorizing $\boldsymbol{A}$ in the form

$$\boldsymbol{A} = \boldsymbol{U}^T \boldsymbol{H} \boldsymbol{U}, \tag{17}$$

where $\boldsymbol{U}$ is an orthogonal matrix and $\boldsymbol{H}$ is a diagonal matrix with the eigenvalues of $\boldsymbol{A}$ on its diagonal. The drawback of PCA is that eigenvectors are almost always dense, while matrices occuring in learning problems, especially those related to graphs, often have strong locality properties, in the sense that they are more closely couple certain clusters of nearby coordinates than those farther apart with respect to the underlying topology. In such cases, modeling A in terms of a basis of global eigenfunctions is both computationally wasteful and conceptually unreasonable: a localized dictionary would be more appropriate. In contrast to PCA, (Kondor et al., 2014) proposed *Multiresolution Matrix Factorization*, or MMF for short, to construct a sparse hierarchical system of L-level dictionaries. The corresponding matrix factorization is of the form

$$\boldsymbol{A} = \boldsymbol{U}_1^T \boldsymbol{U}_2^T \dots \boldsymbol{U}_L^T \boldsymbol{H} \boldsymbol{U}_L \dots \boldsymbol{U}_2 \boldsymbol{U}_1,$$

where $\boldsymbol{H}$ is close to diagonal and $\boldsymbol{U}_1, \dots, \boldsymbol{U}_L$ are sparse orthogonal matrices with the following constraints:

1. Each $\boldsymbol{U}_\ell$ is k-point rotation for some small k, meaning that it only rotates k coordinates at a time. Formally, Def. B.1 defines and Fig. 4 shows an example of the k-point rotation matrix.
2. There is a nested sequence of sets $\mathbb{S}_L \subseteq \dots \subseteq \mathbb{S}_1 \subseteq \mathbb{S}_0 = [n]$ such that the coordinates rotated by $\boldsymbol{U}_\ell$ are a subset of $\mathbb{S}_\ell$.
3. $\boldsymbol{H}$ is an $\mathbb{S}_L$-core-diagonal matrix that is formally defined in Def. B.2.

Definition B.1. We say that $\boldsymbol{U} \in \mathbb{R}^{n \times n}$ is an **elementary rotation of order** k (also called as a k-point rotation) if it is an orthogonal matrix of the form

$$\boldsymbol{U} = \boldsymbol{I}_{n-k} \oplus_{(i_1, \dots, i_k)} \boldsymbol{O}$$

for some $\mathbb{I} = \{i_1, \dots, i_k\} \subseteq [n]$ and $\boldsymbol{O} \in \mathbb{SO}(k)$. We denote the set of all such matrices as $\mathbb{SO}_k(n)$.

The simplest case are second order rotations, or called Givens rotations, which are of the form

$$\boldsymbol{U} = \boldsymbol{I}_{n-2} \oplus_{(i,j)} \boldsymbol{O} = \begin{pmatrix} \cdot & & & \\ & \cos(\theta) & & -\sin(\theta) \\ & & \cdot & \\ & \sin(\theta) & & \cos(\theta) \\ & & & & \cdot \end{pmatrix}, \tag{18}$$

where the dots denote the identity that apart from rows/columns i and j, and $\boldsymbol{O} \in \mathbb{SO}(2)$ is the rotation matrix of some angle $\theta \in [0, 2\pi)$. Indeed, Jacobi's algorithm for diagonalizing symmetric matrices (Jacobi, 1846) is a special case of MMF factorization over Givens rotations.

Definition B.2. Given a set $\mathbb{S} \subseteq [n]$, we say that a matrix $\boldsymbol{H} \in \mathbb{R}^{n \times n}$ is $\mathbb{S}$-core-diagonal if $\boldsymbol{H}_{i,j} = 0$ unless $i, j \in \mathbb{S}$ or $i = j$. Equivalently, $\boldsymbol{H}$ is $\mathbb{S}$-core-diagonal if it can be written in the form $\boldsymbol{H} = \boldsymbol{D} \oplus_{\mathbb{S}} \overline{\boldsymbol{H}}$, for some $\overline{H} \in \mathbb{R}^{|\mathbb{S}| \times |\mathbb{S}|}$ and $\boldsymbol{D}$ is diagonal. We denote the set of all $\mathbb{S}$-core-diagonal symmetric matrices of dimension n as $\mathbb{H}_n^{\mathbb{S}}$.

In general, finding the best MMF factorization to a symmetric matrix $\boldsymbol{A}$ requires solving

$$\min_{\substack{\mathbb{S}_L \subseteq \dots \subseteq \mathbb{S}_1 \subseteq \mathbb{S}_0 = [n] \\ \boldsymbol{H} \in \mathbb{H}_n^{\mathbb{S}_L}; \boldsymbol{U}_1, \dots, \boldsymbol{U}_L \in \mathbb{O}}} \|\boldsymbol{A} - \boldsymbol{U}_1^T \dots \boldsymbol{U}_L^T \boldsymbol{H} \boldsymbol{U}_L \dots \boldsymbol{U}_1\|. \tag{19}$$

B.2. Multiresolution analysis

Definition B.3. Given an appropriate subset $\mathbb{O}$ of the group $\mathbb{SO}(n)$ of n-dimensional rotation matrices, a depth parameter $L \in \mathbb{N}$, and a sequence of integers $n = d_0 \geq d_1 \geq d_2 \geq \dots \geq d_L \geq 1$, a **Multiresolution Matrix Factorization (MMF)** of a symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ over $\mathbb{O}$ is a factorization of the form

$$\boldsymbol{A} = \boldsymbol{U}_1^T \boldsymbol{U}_2^T \dots \boldsymbol{U}_L^T \boldsymbol{H} \boldsymbol{U}_L \dots \boldsymbol{U}_2 \boldsymbol{U}_1, \tag{20}$$

where each $\boldsymbol{U}_\ell \in \mathbb{O}$ satisfies $[\boldsymbol{U}_\ell]_{[n] \setminus \mathbb{S}_{\ell-1}, [n] \setminus \mathbb{S}_{\ell-1}} = \boldsymbol{I}_{n-d_\ell}$ for some nested sequence of sets $\mathbb{S}_L \subseteq \dots \subseteq \mathbb{S}_1 \subseteq \mathbb{S}_0 = [n]$ with $|\mathbb{S}_\ell| = d_\ell$, and $\boldsymbol{H} \in \mathbb{H}_n^{\mathbb{S}_L}$ is an $\mathbb{S}_L$-core-diagonal matrix.

Definition B.4. We say that a symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is **fully multiresolution factorizable** over $\mathbb{O} \subset \mathbb{SO}(n)$ with $(d_1, \dots, d_L)$ if it has a decomposition of the form described in Def. B.3.

$$\begin{tikzcd}
L_2(\mathbb{X}) \arrow[r] & \cdots \arrow[r] & \mathbb{V}_0 \arrow[r] \arrow[dr] & \mathbb{V}_1 \arrow[r] \arrow[dr] & \mathbb{V}_2 \arrow[r] \arrow[dr] & \cdots \\
& & & \mathbb{W}_1 & \mathbb{W}_2 & \mathbb{W}_3
\end{tikzcd}$$

Figure 3. Multiresolution analysis splits each function space $\mathbb{V}_0, \mathbb{V}_1, \dots$ into the direct sum of a smoother part $\mathbb{V}_{\ell+1}$ and a rougher part $\mathbb{W}_{\ell+1}$.

We formally define MMF in Defs. B.3 and B.4. Furthermore, (Kondor et al., 2014) has shown that MMF mirrors the classical theory of multiresolution analysis (MRA) on the real line (Mallat, 1989) to discrete spaces. The functional analytic view of wavelets is provided by MRA, which, similarly to Fourier analysis, is a way of filtering some function space into a sequence of subspaces

$$\cdots \subset \mathbb{V}_{-1} \subset \mathbb{V}_0 \subset \mathbb{V}_1 \subset \mathbb{V}_2 \subset \dots \tag{21}$$

However, it is best to conceptualize (21) as an iterative process of splitting each $\mathbb{V}_\ell$ into the orthogonal sum $\mathbb{V}_\ell = \mathbb{V}_{\ell+1} \oplus \mathbb{W}_{\ell+1}$ of a smoother part $\mathbb{V}_{\ell+1}$, called the *approximation space*; and a rougher part $\mathbb{W}_{\ell+1}$, called the *detail space* (see Fig. 3). Each $\mathbb{V}_\ell$ has an orthonormal basis $\Phi_\ell \triangleq \{\phi_m^\ell\}_m$ in which each ϕ is called a *father* wavelet. Each complementary space $\mathbb{W}_\ell$ is also spanned by an orthonormal basis $\Psi_\ell \triangleq \{\psi_m^\ell\}_m$ in which each ψ is called a *mother* wavelet. In MMF, each individual rotation $\boldsymbol{U}_\ell : \mathbb{V}_{\ell-1} \to \mathbb{V}_\ell \oplus \mathbb{W}_\ell$ is a sparse basis transform that expresses $\Phi_\ell \cup \Psi_\ell$ in the previous basis $\Phi_{\ell-1}$ such that:

$$\phi_m^\ell = \sum_{i=1}^{\dim(\mathbb{V}_{\ell-1})} [\boldsymbol{U}_\ell]_{m,i} \phi_i^{\ell-1},$$

$$\psi_m^\ell = \sum_{i=1}^{\dim(\mathbb{V}_{\ell-1})} [\boldsymbol{U}_\ell]_{m+\dim(\mathbb{V}_{\ell-1}),i} \phi_i^{\ell-1},$$

in which Φ_0 is the standard basis, i.e. $\phi_m^0 = e_m$; and $\dim(\mathbb{V}_\ell) = d_\ell = |\mathbb{S}_\ell|$. In the $\Phi_1 \cup \Psi_1$ basis, $\boldsymbol{A}$ compresses into $\boldsymbol{A}_1 = \boldsymbol{U}_1 \boldsymbol{A} \boldsymbol{U}_1^T$. In the $\Phi_2 \cup \Psi_2 \cup \Psi_1$ basis, it becomes $\boldsymbol{A}_2 = \boldsymbol{U}_2 \boldsymbol{U}_1 \boldsymbol{A} \boldsymbol{U}_1^T \boldsymbol{U}_2^T$, and so on. Finally, in the $\Phi_L \cup \Psi_L \cup \dots \cup \Psi_1$ basis, it takes on the form $\boldsymbol{A}_L = \boldsymbol{H} = \boldsymbol{U}_L \dots \boldsymbol{U}_2 \boldsymbol{U}_1 \boldsymbol{A} \boldsymbol{U}_1^T \boldsymbol{U}_2^T \dots \boldsymbol{U}_L^T$ that consists of four distinct blocks (supposingly that we permute the rows/columns accordingly):

$$\boldsymbol{H} = \begin{pmatrix} \boldsymbol{H}_{\Phi,\Phi} & \boldsymbol{H}_{\Phi,\Psi} \\ \boldsymbol{H}_{\Psi,\Phi} & \boldsymbol{H}_{\Psi,\Psi} \end{pmatrix},$$

where $\boldsymbol{H}_{\Phi,\Phi} \in \mathbb{R}^{\dim(\mathbb{V}_L) \times \dim(\mathbb{V}_L)}$ is effectively $\boldsymbol{A}$ compressed to $\mathbb{V}_L$, $\boldsymbol{H}_{\Phi,\Psi} = \boldsymbol{H}_{\Psi,\Phi}^T = 0$ and $\boldsymbol{H}_{\Psi,\Psi}$ is diagonal. MMF approximates $\boldsymbol{A}$ in the form

$$\boldsymbol{A} \approx \sum_{i,j=1}^{d_L} h_{i,j} \phi_i^L {\phi_j^L}^T + \sum_{\ell=1}^{L} \sum_{m=1}^{d_\ell} c_m^\ell \psi_m^\ell {\psi_m^\ell}^T,$$

where $h_{i,j}$ coefficients are the entries of the $\boldsymbol{H}_{\Phi,\Phi}$ block, and $c_m^\ell = \langle \psi_m^\ell, \boldsymbol{A} \psi_m^\ell \rangle$ wavelet frequencies are the diagonal elements of the $\boldsymbol{H}_{\Psi,\Psi}$ block.

In particular, the dictionary vectors corresponding to certain rows of $\boldsymbol{U}_1$ are interpreted as level one wavelets, the dictionary vectors corresponding to certain rows of $\boldsymbol{U}_2 \boldsymbol{U}_1$ are interpreted as level two wavelets, and so on (see Section B.2). One thing that is immediately clear is that whereas Eq. (17) diagonalizes $\boldsymbol{A}$ in a single step, multiresolution analysis will involve a sequence of basis transforms $\boldsymbol{U}_1, \boldsymbol{U}_2, \dots, \boldsymbol{U}_L$, transforming $\boldsymbol{A}$ step by step as

$$\boldsymbol{A} \to \boldsymbol{U}_1 \boldsymbol{A} \boldsymbol{U}_1^T \to \boldsymbol{U}_2 \boldsymbol{U}_1 \boldsymbol{A} \boldsymbol{U}_1^T \boldsymbol{U}_2^T \to \dots \to \boldsymbol{U}_L \dots \boldsymbol{U}_2 \boldsymbol{U}_1 \boldsymbol{A} \boldsymbol{U}_1^T \boldsymbol{U}_2^T \dots \boldsymbol{U}_L^T, \tag{22}$$

so the corresponding matrix factorization must be a multilevel factorization

$$\boldsymbol{A} \approx \boldsymbol{U}_1^T \boldsymbol{U}_2^T \dots \boldsymbol{U}_\ell^T \boldsymbol{H} \boldsymbol{U}_\ell \dots \boldsymbol{U}_2 \boldsymbol{U}_1. \tag{23}$$

Fig. 5 depicts the multiresolution transform of MMF as in Eq. (22). Fig. 6 illustrates the corresponding factorization as in Eq. (23).

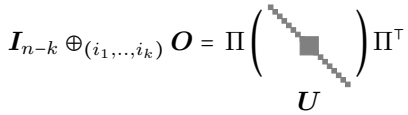


Figure 4. A rotation matrix of order k. The purpose of permutation matrix Π is solely to ensure that the blocks of the matrices appear contiguous in the figure. In this case, $n = 17$ and $k = 4$.

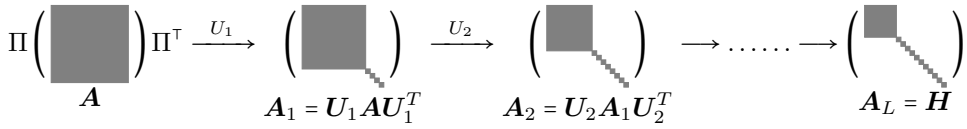


Figure 5. MMF can be thought of as a process of successively compressing $\boldsymbol{A}$ to size $d_1 \times d_1$, $d_2 \times d_2$, etc. (plus the diagonal entries) down to the final $d_L \times d_L$ core-diagonal matrix $\boldsymbol{H}$ (see Def. B.3). The role of permutation matrix Π is purely for the ease of visualization (as in Fig. 4).

B.3. Optimization by heuristics

Heuristically, factorizing $\boldsymbol{A}$ can be approximated by an iterative process that starts by setting $\boldsymbol{A}_0 = \boldsymbol{A}$ and $\mathbb{S}_1 = [n]$, and then executes the following steps for each resolution level $\ell \in \{1, \dots, L\}$:

1. Given $\boldsymbol{A}_{\ell-1}$, select k indices $\mathbb{I}_\ell = \{i_1, \dots, i_k\} \subset \mathbb{S}_{\ell-1}$ of rows/columns of the active submatrix $[\boldsymbol{A}_{\ell-1}]_{\mathbb{S}_{\ell-1},\mathbb{S}_{\ell-1}}$ that are highly correlated with each other.
2. Find the corresponding k-point rotation $\boldsymbol{U}_\ell$ to $\mathbb{I}_\ell$, and compute $\boldsymbol{A}_\ell = \boldsymbol{U}_\ell \boldsymbol{A}_{\ell-1} \boldsymbol{U}_\ell^T$ that brings the submatrix $[\boldsymbol{A}_{\ell-1}]_{\mathbb{I}_\ell,\mathbb{I}_\ell}$ close to diagonal. In the last level, we set $\boldsymbol{H} = \boldsymbol{A}_L$ (see Fig. 5).
3. Determine the set of coordinates $\mathbb{T}_\ell \subseteq \mathbb{S}_{\ell-1}$ that are to be designated wavelets at this level, and eliminate them from the active set by setting $\mathbb{S}_\ell = \mathbb{S}_{\ell-1} \setminus \mathbb{T}_\ell$.

C. Stiefel Manifold Optimization

In order to solve the MMF optimization problem, we consider the following generic optimization with orthogonality constraints:

$$\min_{\boldsymbol{X} \in \mathbb{R}^{n \times p}} \mathcal{F}(\boldsymbol{X}), \quad \text{s.t.} \quad \boldsymbol{X}^T \boldsymbol{X} = \boldsymbol{I}_p, \tag{24}$$

We identify tangent vectors to the manifold with $n \times p$ matrices. We denote the tangent space at $\boldsymbol{X}$ as $\mathcal{T}_{\boldsymbol{X}} \mathcal{V}_p(\mathbb{R}^n)$. Lemma C.1 characterizes vectors in the tangent space.

Lemma C.1. *Any $\boldsymbol{Z} \in \mathcal{T}_{\boldsymbol{X}} \mathcal{V}_p(\mathbb{R}^n)$, then $\boldsymbol{Z}$ (as an element of $\mathbb{R}^{n \times p}$) satisfies*

$$\boldsymbol{Z}^T \boldsymbol{X} + \boldsymbol{X}^T \boldsymbol{Z} = 0,$$

where $\boldsymbol{Z}^T \boldsymbol{X}$ is a skew-symmetric $p \times p$ matrix.

Proof. Let $\boldsymbol{Y}(t)$ be a curve in $\mathcal{V}_p(\mathbb{R}^n)$ that starts from $\boldsymbol{X}$. We have:

$$\boldsymbol{Y}^T(t) \boldsymbol{Y}(t) = \boldsymbol{I}_p. \tag{25}$$

We differentiate two sides of Eq. (25) with respect to t:

$$\frac{d}{dt}(\boldsymbol{Y}^T(t) \boldsymbol{Y}(t)) = 0$$

that leads to:

$$\left(\frac{d\boldsymbol{Y}}{dt}(0)\right)^T \boldsymbol{Y}(0) + \boldsymbol{Y}(0)^T \frac{d\boldsymbol{Y}}{dt}(0) = 0$$

at $t = 0$. Recall that by definition, $\boldsymbol{Y}(0) = \boldsymbol{X}$ and $\frac{d\boldsymbol{Y}}{dt}(0)$ is any element of the tangent space at $\boldsymbol{X}$. Therefore, we arrive at $\boldsymbol{Z}^T \boldsymbol{X} + \boldsymbol{X}^T \boldsymbol{Z} = 0$. □

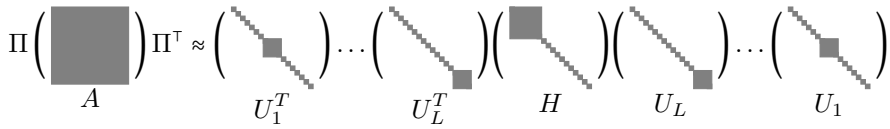


Figure 6. Matrix approximation as in Eq. 20. In this figure, the core block size of each rotation matrix $\boldsymbol{U}_\ell$ and $\boldsymbol{H}$ are $k \times k = 4 \times 4$ and $d_L \times d_L = 8 \times 8$, respectively. Permutation matrix Π is only for visualization (as in Figs. 4 5).

Suppose that $\mathcal{F}$ is a differentiable function. The gradient of $\mathcal{F}$ with respect to $\boldsymbol{X}$ is denoted by $\boldsymbol{G} \triangleq \mathcal{D}\mathcal{F}_{\boldsymbol{X}} \triangleq \left(\frac{\partial \mathcal{F}(\boldsymbol{X})}{\partial \boldsymbol{X}_{i,j}}\right)$. The derivative of $\mathcal{F}$ at $\boldsymbol{X}$ in a direction $\boldsymbol{Z}$ is

$$\mathcal{D}\mathcal{F}_{\boldsymbol{X}}(\boldsymbol{Z}) \triangleq \lim_{t \to 0} \frac{\mathcal{F}(\boldsymbol{X} + t\boldsymbol{Z}) - \mathcal{F}(\boldsymbol{X})}{t} = \langle \boldsymbol{G}, \boldsymbol{Z} \rangle$$

Since the matrix $\boldsymbol{X}^T\boldsymbol{X}$ is symmetric, the Lagrangian multiplier Λ corresponding to $\boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{I}_p$ is a symmetric matrix. The Lagrangian function of problem (24) is

$$\mathcal{L}(\boldsymbol{X}, \boldsymbol{\Lambda}) = \mathcal{F}(\boldsymbol{X}) - \frac{1}{2}\text{trace}(\boldsymbol{\Lambda}(\boldsymbol{X}^T\boldsymbol{X} - \boldsymbol{I}_p)) \tag{26}$$

Lemma C.2. *Suppose that $\boldsymbol{X}$ is a local minimizer of problem (24). Then $\boldsymbol{X}$ satisfies the first-order optimality conditions $\mathcal{D}_{\boldsymbol{X}}\mathcal{L}(\boldsymbol{X}, \boldsymbol{\Lambda}) = \boldsymbol{G} - \boldsymbol{X}\boldsymbol{G}^T\boldsymbol{X} = 0$ and $\boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{I}_p$ with the associated Lagrangian multiplier $\boldsymbol{\Lambda} = \boldsymbol{G}^T\boldsymbol{X}$. Define $\nabla\mathcal{F}(\boldsymbol{X}) \triangleq \boldsymbol{G} - \boldsymbol{X}\boldsymbol{G}^T\boldsymbol{X}$ and $\boldsymbol{A} \triangleq \boldsymbol{G}\boldsymbol{X}^T - \boldsymbol{X}\boldsymbol{G}^T$. Then $\nabla\mathcal{F} = \boldsymbol{A}\boldsymbol{X}$. Moreover, $\nabla\mathcal{F} = 0$ if and only if $\boldsymbol{A} = 0$.*

Proof. Since $\boldsymbol{X} \in \mathcal{V}_p(\mathbb{R}^n)$, we have $\boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{I}_p$. We differentiate both sides of the Lagrangian function:

$$\mathcal{D}_{\boldsymbol{X}}\mathcal{L}(\boldsymbol{X}, \boldsymbol{\Lambda}) = \mathcal{D}\mathcal{F}(\boldsymbol{X}) - \boldsymbol{X}\boldsymbol{\Lambda} = 0.$$

Recall that by definition, $\boldsymbol{G} \triangleq \mathcal{D}\mathcal{F}(\boldsymbol{X})$, we have

$$\mathcal{D}_{\boldsymbol{X}}\mathcal{L}(\boldsymbol{X}, \boldsymbol{\Lambda}) = \boldsymbol{G} - \boldsymbol{X}\boldsymbol{\Lambda} = 0. \tag{27}$$

Multiplying both sides by $\boldsymbol{X}^T$, we get $\boldsymbol{X}^T\boldsymbol{G} - \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\Lambda} = 0$ that leads to $\boldsymbol{X}^T\boldsymbol{G} - \boldsymbol{\Lambda} = 0$ or $\boldsymbol{\Lambda} = \boldsymbol{X}^T\boldsymbol{G}$. Since the matrix $\boldsymbol{X}^T\boldsymbol{X}$ is symmetric, the Lagrangian multiplier $\boldsymbol{\Lambda}$ correspoding to $\boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{I}_p$ is a symmetric matrix. Therefore, we obtain $\boldsymbol{\Lambda} = \boldsymbol{\Lambda}^T = \boldsymbol{G}^T\boldsymbol{X}$ and $\mathcal{D}_{\boldsymbol{X}}\mathcal{L}(\boldsymbol{X}, \boldsymbol{\Lambda}) = \boldsymbol{G} - \boldsymbol{X}\boldsymbol{G}^T\boldsymbol{X} = 0$. By definition, $\boldsymbol{A} \triangleq \boldsymbol{G}\boldsymbol{X}^T - \boldsymbol{X}\boldsymbol{G}^T$. We have $\boldsymbol{A}\boldsymbol{X} = \boldsymbol{G} - \boldsymbol{X}\boldsymbol{G}^T\boldsymbol{X} = \nabla\mathcal{F}$. The last statement is trivial. □

Let $\boldsymbol{X} \in \mathcal{V}_p(\mathbb{R}^n)$, and $\boldsymbol{W}$ be any $n \times n$ skew-symmetric matrix. We consider the following curve that transforms $\boldsymbol{X}$ by $\left(\boldsymbol{I} + \frac{\tau}{2}\boldsymbol{W}\right)^{-1}\left(\boldsymbol{I} - \frac{\tau}{2}\boldsymbol{W}\right)$:

$$\boldsymbol{Y}(\tau) = \left(\boldsymbol{I} + \frac{\tau}{2}\boldsymbol{W}\right)^{-1}\left(\boldsymbol{I} - \frac{\tau}{2}\boldsymbol{W}\right)\boldsymbol{X}. \tag{28}$$

This is called as the *Cayley transformation*. Its derivative with respect to τ is

$$\boldsymbol{Y}'(\tau) = -\left(\boldsymbol{I} + \frac{\tau}{2}\boldsymbol{W}\right)^{-1}\boldsymbol{W}\left(\frac{\boldsymbol{X} + \boldsymbol{Y}(\tau)}{2}\right). \tag{29}$$

The curve has the following properties:

1. It stays in the Stiefel manifold, i.e. $\boldsymbol{Y}(\tau)^T\boldsymbol{Y}(\tau) = \boldsymbol{I}$.
2. Its tangent vector at $\tau = 0$ is $\boldsymbol{Y}'(0) = -\boldsymbol{W}\boldsymbol{X}$. It can be easily derived from Lemma C.1 that $\boldsymbol{Y}'(0)$ is in the tangent space $\mathcal{T}_{\boldsymbol{Y}(0)}\mathcal{V}_p(\mathbb{R}^n)$. Since $\boldsymbol{Y}(0) = X$ and $\boldsymbol{W}$ is a skew-symmetric matrix, by letting $\boldsymbol{Z} = -\boldsymbol{W}\boldsymbol{X}$, it is trivial that $\boldsymbol{Z}^T\boldsymbol{X} + \boldsymbol{X}^T\boldsymbol{Z} = 0$.

Lemma C.3. *If we set $\boldsymbol{W} \triangleq \boldsymbol{A} \triangleq \boldsymbol{G}\boldsymbol{X}^T - \boldsymbol{X}\boldsymbol{G}^T$ (see Lemma C.2), then the curve $\boldsymbol{Y}(\tau)$ (defined in Eq. (28)) is a decent curve for $\mathcal{F}$ at $\tau = 0$, that is*

$$\mathcal{F}'_\tau(\boldsymbol{Y}(0)) \triangleq \left.\frac{\partial \mathcal{F}(\boldsymbol{Y}(\tau))}{\partial \tau}\right|_{\tau=0} = -\frac{1}{2}\|\boldsymbol{A}\|_F^2.$$

Proof. By the chain rule, we get

$$\mathcal{F}'_\tau(\boldsymbol{Y}(\tau)) = \text{trace}(\mathcal{D}\mathcal{F}(\boldsymbol{Y}(\tau))^T \boldsymbol{Y}'(\tau)).$$

At $\tau = 0$, $\mathcal{D}\mathcal{F}(\boldsymbol{Y}(0)) = \boldsymbol{G}$ and $\boldsymbol{Y}'(0) = -\boldsymbol{A}\boldsymbol{X}$. Therefore,

$$\mathcal{F}'_\tau(\boldsymbol{Y}(0)) = -\text{trace}(\boldsymbol{G}^T(\boldsymbol{G}\boldsymbol{X}^T - \boldsymbol{X}\boldsymbol{G}^T)\boldsymbol{X}) = -\frac{1}{2}\text{trace}(\boldsymbol{A}\boldsymbol{A}^T) = -\frac{1}{2}\|\boldsymbol{A}\|_F^2.$$

□

It is well known that the steepest descent method with a fixed step size may not converge, but the convergence can be guaranteed by choosing the step size wisely: one can choose a step size by minimizing $\mathcal{F}(\boldsymbol{Y}(\tau))$ along the curve $\boldsymbol{Y}(\tau)$ with respect to τ (Wen & Yin, 2010). With the choice of $\boldsymbol{W}$ given by Lemma C.3, the minimization algorithm using $\boldsymbol{Y}(\tau)$ is roughly sketched as follows: Start with some initial $\boldsymbol{X}^{(0)}$. For $t > 0$, we generate $\boldsymbol{X}^{(t+1)}$ from $\boldsymbol{X}^{(t)}$ by a curvilinear search along the curve $\boldsymbol{Y}(\tau) = \left(\boldsymbol{I} + \frac{\tau}{2}\boldsymbol{W}\right)^{-1}\left(\boldsymbol{I} - \frac{\tau}{2}\boldsymbol{W}\right)\boldsymbol{X}^{(t)}$ by changing τ. Because finding the global minimizer is computationally infeasible, the search terminates when then Armijo-Wolfe conditions that indicate an approximate minimizer are satisfied. The Armijo-Wolfe conditions require two parameters $0 < \rho_1 < \rho_2 < 1$ (Nocedal & Wright, 2006) (Wen & Yin, 2010) (Tagare, 2011):

$$\mathcal{F}(\boldsymbol{Y}(\tau)) \leq \mathcal{F}(\boldsymbol{Y}(0)) + \rho_1 \tau \mathcal{F}'_\tau(\boldsymbol{Y}(0)) \tag{30}$$

$$\mathcal{F}'_\tau(\boldsymbol{Y}(\tau)) \geq \rho_2 \mathcal{F}'_\tau(\boldsymbol{Y}(0)) \tag{31}$$

where $\mathcal{F}'_\tau(\boldsymbol{Y}(\tau)) = \text{trace}(\boldsymbol{G}^T\boldsymbol{Y}'(\tau))$ while $\boldsymbol{Y}'(\tau)$ is computed as Eq. (29) and $\boldsymbol{Y}'(0) = -\boldsymbol{A}\boldsymbol{X}$. The gradient descent algorithm on Stiefel manifold to optimize the generic orthogonal-constraint problem (24) with the curvilinear search submodule is described in Algorithm 2, which is used as a submodule in part of our learning algorithm to solve the MMF in (19). The algorithm can be trivially extended to solve problems with multiple variables and constraints.

Algorithm 2 Stiefel manifold gradient descent algorithm

Given $0 < \rho_1 < \rho_2 < 1$ and $\epsilon > 0$.
Given an initial point $\boldsymbol{X}^{(0)} \in \mathcal{V}_p(\mathbb{R}^n)$.
$t \leftarrow 0$
while true **do**
  $\boldsymbol{G} \leftarrow \left(\frac{\partial \mathcal{F}(\boldsymbol{X}^{(t)})}{\partial \boldsymbol{X}^{(t)}_{i,j}}\right)$ ▷ Compute the gradient of $\mathcal{F}$ w.r.t $\boldsymbol{X}$ elemense-wise
  $\boldsymbol{A} \leftarrow \boldsymbol{G}\boldsymbol{X}^{(t)^T} - \boldsymbol{X}^{(t)}\boldsymbol{G}^T$ ▷ See Lemma 2, 3
  Initialize τ to a non-zero value. ▷ Curvilinear search for the optimal step size
  while (30) and (31) are **not** satisfied **do** ▷ Armijo-Wolfe conditions
    $\tau \leftarrow \frac{\tau}{2}$ ▷ Reduce the step size by half
  end while
  $\boldsymbol{X}^{(t+1)} \leftarrow \boldsymbol{Y}(\tau)$ ▷ Update by the Cayley transformation
  if $\|\nabla \mathcal{F}(\boldsymbol{X}^{(t+1)})\| \leq \epsilon$ **then** ▷ Stopping check. See Lemma 2.
    STOP
  else
    $t \leftarrow t + 1$
  end if
end while

D. Reinforcement Learning

D.1. Policy gradient methods

Policy gradient has been a widely used approach to solve reinforcement learning problems that targets at modeling and optimizing the policy directly (Sutton & Barto, 2018). Monte-Carlo policy gradient (REINFORCE) (Williams, 1988) (Williams, 1992) (Sutton et al., 2000) depends on an estimated return by Monte-Carlo methods using episode samples to update the learnable parameters θ of the policy network π_θ. We define the value of state s when we follow a policy

π as $V^\pi(s) = \mathbb{E}_{a\sim\pi}[g_\ell | s_\ell = s]$. The value of (state, action) pair when we follow a policy π is defined similarly as $Q^\pi(s,a) = \mathbb{E}_{a\sim\pi}[g_\ell | s_\ell = s, a_\ell = a]$. The value of the reward objective function depends on the policy and is defined as

$$J(\theta) = \sum_{s\in S} d^\pi(s) V^\pi(s) = \sum_{s\in S} d^\pi(s) \sum_{a\in A} \pi_\theta(a|s) Q^\pi(s,a) \tag{32}$$

where $d^\pi(s) = p(s_L = s | s_0, \pi_\theta)$ is the stationary distribution of Markov chain for π_θ. It is important to remark that our MDP process terminates after a finite number of transitions (e.g., L), so $d^\pi(s)$ is the probability that we end up at state s when starting from s_0 and following policy π_θ for L steps. (Sutton et al., 2000) has shown that an unbiased estimate of the gradient of (32) can be obtained from experience using an approximate value function. The expectation of the sample gradient is equal to the actual gradient:

$$\nabla_\theta J(\theta) = \mathbb{E}_\pi[Q^\pi(s,a) \nabla_\theta \log \pi_\theta(a|s)] = \mathbb{E}_\pi[g_\ell \nabla_\theta \log \pi_\theta(a_\ell | s_\ell)] \tag{33}$$

that allows us to update our policy gradient by measuring g_ℓ from real sample trajectories. Based on (33), the update rule for policy parameters is simply as

$$\theta \leftarrow \theta + \eta \gamma^\ell g_\ell \nabla_\theta \log \pi_\theta(a_\ell | s_\ell) \quad \text{for } \ell = 0, .., L-1;$$

where η is the learning rate, that is used in training our policy network π_θ.

D.2. 2-phase process

The learning algorithm is expensive due to the Stiefel manifold optimization in line 11 to find the optimal rotations $\boldsymbol{O}_\ell$ that are used to compute the rewards g_ℓ. In practice, we propose a 2-phase process that is more efficient:

- **Phase 1:** Reinforcement learning to find the sequence of indices, but instead of manifold optimization, we just use the closed-form solutions for $\boldsymbol{O}_\ell$ as the eigenvectors of $\boldsymbol{A}_{\mathbb{S}_\ell,:}\boldsymbol{A}^T_{\mathbb{S}_\ell,:}$ to estimate the rewards. In all our experiments, we implement the policy network by two graph neural networks, one to select the pivot (wavelet index) and the another one to select $K-1$ indices, with 4 layers of message passing and hidden dimension of 10. The input node feature for node v (or the v-th row) is binary: 1 if $v \in \mathbb{S}_\ell$, otherwise 0. We use $\gamma = 1$ as the discount factor and learning rate $\eta = 10^{-3}$.
- **Phase 2:** Given a sequence of indices found by the previous phase, we apply Stiefel manifold optimization to actually find the optimal rotations accordingly.

D.3. Transfer learning

Ideally, we want our policy network π_θ to be *universal* in the sense that the same trained policy can be applied to different graphs with little adaptation or without any further training. However, the search space is gigantic with large graphs such as social networks, and the cost of training the policy is computationally expensive. Therefore, we apply the idea of *transfer learning* that is to reuse or transfer information from previously learned tasks (source tasks) into new tasks (target tasks). The source task here is to train our GNN policy on a dataset of small graphs (e.g., possibly synthetic graphs) that are much faster to train on, and the target task is to run the trained policy on the large actual graph. For example, given citation networks with thousands of nodes such as Cora and Citeseer (Sen et al., 2008), we generate the dataset for policy training by partitioning the big graph into many smaller connected clusters. The learning algorithm can be easily modified for such a purpose (e.g., training multiple graphs simultaneously).