Proceedings of Machine Learning Research vol 145:1–33, 2022 3rd Annual Conference on Mathematical and Scientific Machine Learning

SpecNet2: Orthogonalization-free Spectral Embedding by Neural Networks

Ziyu Chen ZIYU@MATH.DUKE.EDU
Department of Mathematics, Duke University

Yingzhou Li YINGZHOULI@FUDAN.EDU.CN
School of Mathematical Sciences, Fudan University

Xiuyuan Cheng XIUYUAN.CHENG@DUKE.EDU
Department of Mathematics, Duke University

Editors: Bin Dong, Qianxiao Li, Lei Wang, Zhi-Qin John Xu

Abstract

Spectral methods which represent data points by eigenvectors of kernel matrices or graph Laplacian matrices have been a primary tool in unsupervised data analysis. In many application scenarios, parametrizing the spectral embedding by a neural network that can be trained over batches of data samples gives a promising way to achieve automatic out-of-sample extension as well as computational scalability. Such an approach was taken in the original paper of SpectralNet (Shaham et al. 2018), which we call SpecNet1. The current paper introduces a new neural network approach, named SpecNet2, to compute spectral embedding which optimizes an equivalent objective of the eigen-problem and removes the orthogonalization layer in SpecNet1. SpecNet2 also allows separating the sampling of rows and columns of the graph affinity matrix by tracking the neighbors of each data point through the gradient formula. Theoretically, we show that any local minimizer of the new orthogonalization-free objective reveals the leading eigenvectors. Furthermore, global convergence for this new orthogonalization-free objective using a batch-based gradient descent method is proved. Numerical experiments demonstrate the improved performance and computational efficiency of SpecNet2 on simulated data and image datasets.

Keywords: Spectral embedding, orthogonalization-free, neural network

1. Introduction

Spectral embedding, namely representing data points in lower dimensional space using eigenvectors of a kernel matrix or graph Laplacian matrix, plays a crucial role in unsupervised data analysis. It can be used, for example, for dimension reduction, spectral clustering, and revealing the underlying topological structure of a dataset. A known challenge in the use of spectral embedding is the out-of-sample extension. Another shortcoming in practice is the possible high computational cost due to the involvement of constructing a kernel matrix and solving an eigen-problem. To overcome these challenges, previously, the original SpectralNet (Shaham et al., 2018), which we call SpecNet1 in this paper, adopted a neural network to embed data into the eigenspace of its associated graph Laplacian matrix. To be able to enforce the orthogonality among eigenvectors, an additional orthogonalization layer is appended to the neural network and updated after each optimization step. Accurate computation of the orthogonalization layer requires evaluation of the neural network model on the whole dataset, which would be computationally too expensive for large datasets. To reduce the computational cost in practice, in SpecNet1 the computation of the orthogonalization layer is approximated by only using mini-batches of data samples, see more in Section 2.3. However, when

a small batch is being used, the approximation error therein leads to unsatisfactory convergence behavior in practice. This is further elaborated in Remark 2. This work develops SpecNet2, which removes the orthogonalization layer and will compute the neural network spectral embedding more efficiently.

From the perspective of linear algebra eigen-problems, SpecNet1 adopts the projected gradient descent method to optimize an *orthogonally-constrained* quadratic objective of the eigenvalue problem, where the orthogonalization layer is updated to conduct the orthogonalization projection step through a QR decomposition or a Cholesky decomposition. Meanwhile, the past decade has witnessed a trend in employing *unconstrained* optimization to address the eigenvalue problem without the need for the orthogonalization step (Liu et al., 2015; Lei et al., 2016; Li et al., 2019; Gao et al., 2020, 2021), especially in the field of computational chemistry (Mauri et al., 1993; Ordejon et al., 1993; Wang et al., 2019). These unconstrained optimization techniques are also known as "orthogonalization-free optimization" for solving eigen-problems. All of these methods adopt various forms of quadratic polynomials as their objective functions. Some works (Liu et al., 2015; Lei et al., 2016; Li et al., 2019; Gao et al., 2020; Wang et al., 2019) are equivalent to applying the penalty method to the orthogonally constrained optimization problem. There are two major reasons behind moving from constrained optimization to unconstrained optimization: 1) explicit orthogonalization requires solving a reduced size eigenvalue problem, which is not of high parallel efficiency; 2) explicit orthogonalization requires accessing the entire vectors, which is incompatible with batch updating scheme. As a result, in dealing with large-scale eigenvalue problems where parallelization and/or batch updating scheme are needed, an unconstrained optimization approach is preferred. This naturally suggests the use of unconstrained optimization for the eigen-problem in neural network spectral embedding methods.

In the current paper, we modify the orthogonalization-free objective in (Li et al., 2019) for the graph Laplacian matrix and use it under the spectral network framework (Shaham et al., 2018) so as to compute neural network parametrized spectral embedding of data. The contribution of the work includes

- The proposed spectral network, SpecNet2, is trained with an orthogonalization-free training objective, which can be optimized more efficiently than SpecNet1. In particular, the new optimization objective we proposed allows updating the graph neighbors of the samples in a mini-batch at each iteration, which memory-wise only requires loading part of the affinity matrix restricted to that graph neighborhood. Thus the method better scales to large graphs.

- Theoretically, it is proved that the global minimum of the orthogonalization-free objective function (unique up to a rotation) reveals the leading eigenvectors of the graph Laplacian matrix and the iterative update scheme is guaranteed to converge to the global minimizer for any initial point up to a measure-zero set.

- The efficiency and advantage of SpecNet2 over SpecNet1 with neighbor evaluation scheme are demonstrated empirically on several numerical examples. The network embedding also shows better stability and accuracy in some cases.

The rest of the paper is organized as follows. In Section 2, we introduce notations used throughout the paper, as well as the background of the spectral embedding problem we aim to solve. In Section 3, we propose an orthogonalization-free iterative eigen-problem solver from a numerical

linear algebra point of view with three updating schemes. In Section 4, we introduce the neural network parametrization as well as the updating scheme implementations in neural network training. Theoretical results are analyzed in Section 5. Numerical results are shown in Section 6. Finally, we conclude our paper with discussions in Section 7.

2. Background

2.1. Graph Laplacian and spectral embedding

Given a dataset of n samples $X = \{x_i\}_{i=1}^n$ in $\mathbb{R}^m$, an *affinity matrix* W is constructed such that $W_{i,j}$ measures the similarity between x_i and x_j. By construction, W is a real symmetric matrix of size $n \times n$, and $W_{i,j} \geq 0$. One could also view W as the weights on edges of an undirected graph $G = (V, E)$, where $V = [n] = \{1, 2, \ldots, n\}$, $E = \{(i, j), W_{i,j} > 0\}$. In many scenarios, W is constructed to be a sparse matrix. For example, in Laplacian eigenmap (Belkin and Niyogi, 2003) and Diffusion map (Coifman and Lafon, 2006), the affinity matrix can be constructed as

$$W_{i,j} = h\left(\frac{\|x_i - x_j\|^2}{\sigma^2}\right), \tag{1}$$

where h is a non-negative function on $[0, \infty)$, is compactly supported or decays exponentially. As a result, when the kernel bandwidth σ is chosen to be the scale of the size of a local neighborhood, then $W_{i,j}$ is significantly non-zero only when x_i is a nearby neighbor of x_j. Typical non-negative examples of the function h include the indicator function on $[0, 1)$, Gaussian function e^{-r^2}, truncated Gaussian function, etc. Other examples of W which differs from the form of (1) include kNN graphs and kernels with self-tuned bandwidth (Cheng and Wu, 2021a). These constructions also produce a sparse real symmetric matrix W.

Given an affinity matrix W, the *degree matrix* D of W is a diagonal matrix with diagonal entries defined by $D_{i,i} = \sum_{j=1}^n W_{i,j}$. Note that $D_{i,i} > 0$ whenever the graph has no isolated node. The matrix $P := D^{-1}W$ is row-stochastic and can be viewed as the transition matrix of a random walk on the graph. The matrix $L_{rw} = I - P$ is called the "random-walk graph Laplacian" and L_{rw} has real eigenvalues and eigenvectors $L_{rw}\psi_k = \lambda_k\psi_k$, starting from $\lambda_1 = 0$ and ψ_1 is a constant eigenvector. Throughout this paper, we call the zero eigenvalue the "trivial" eigenvalue and eigenvectors associated with zero eigenvalue the "trivial" eigenvectors of L_{rw}; "nontrivial" refers to non-zero eigenvalues and eigenvectors associated with non-zero eigenvalues of L_{rw}. When the graph is connected, the trivial eigenvalue zero is of multiplicity one. The first $K - 1$ nontrivial eigenvectors $\psi_2, \ldots, \psi_K$ associated with the smallest eigenvalues $0 < \lambda_2 \leq \cdots \leq \lambda_K$ of L_{rw}, can provide a low-dimensional embedding of the dataset X, known as the Laplacian Eigenmap (Belkin and Niyogi, 2003), where each sample is mapped to

$$x_i \mapsto \Psi(x_i) = [\psi_2(i), \ldots, \psi_K(i)] \in \mathbb{R}^{K-1}. \tag{2}$$

Diffusion maps (Coifman and Lafon, 2006) are closely related and map

$$x_i \mapsto \Psi_t(x_i) = [\lambda_2^t\psi_2(i), \ldots, \lambda_K^t\psi_K(i)] \in \mathbb{R}^{K-1}, \tag{3}$$

where $t > 0$ is the diffusion time. These embeddings using the eigenvectors of graph Laplacians are called *spectral embedding*. The eigenvectors of the unnormalized graph Laplacian $D - W$ have also been used for embedding and spectral clustering.

2.2. Out-of-sample extension and limiting eigenfunctions

Note that in (2), the mapping Ψ is defined on the discrete points $x_i \in X$ but not on the whole space yet, since it is provided by the eigenvectors of a discrete graph Laplacian matrix. The problem of *out-of-sample extension* for kernel methods and spectral methods refers to the problem of efficiently generalizing the spectral embedding to new samples not in X. Recomputing the eigenvalue decomposition on the extended dataset is computationally too expensive to be practical. Ideally, we would like to generalize the spectral embedding without such recomputation. Classical methods include Nyström extension (Nyström, 1930; Belabbas and Wolfe, 2009; Williams and Seeger, 2000) and its variants (Bermanis et al., 2013). More recently, a neural network-based approach has been proposed in (Shaham et al., 2018) to parametrize the eigenvectors of the Laplacian that automatically gives an out-of-sample extension.

Theoretically, it is thus natural to ask when the mapping $\Psi(x_i)$ is the restriction of an underlying eigenfunction in the continuous space on the dataset X. An answer has been provided by the theory of *spectral convergence* in a manifold data setting: when data are sampled on a sub-manifold $\mathcal{M}$ which can be of lower dimensionality than the ambient space, the eigenvectors and eigenvalues of the graph Laplacian L_{rw} built from n samples with kernel bandwidth parameter σ converge to the eigenfunctions and eigenvalues of a limiting differential operator $\mathcal{L}$ when $n \to \infty$ and $\sigma \to 0$ (Coifman and Lafon, 2006). The expression of $\mathcal{L}$ depends on the affinity construction and the kernel matrix normalization, e.g., when data points are uniformly sampled on the manifold with respect to the Riemannian volume then $\mathcal{L} = -\Delta_{\mathcal{M}}$ (the Laplace-Beltrami operator up to a sign); and when density is non-uniform, $\mathcal{L}$ is a certain infinitesimal generator of the manifold diffusion process. The spectral convergence on finite samples requires σ to scale with n in a proper way, and in practice, the low-lying eigenvectors, namely those with smaller eigenvalues of $\mathcal{L}$ near zero, converge faster than the high-frequency (high-lying) ones.

As a result, in applications where the data samples can be viewed as lying on or near to a low-dimensional submanifold, it is natural to parametrize the first $K-1$ nontrivial eigenvectors ψ_k of the large kernel matrix, evaluated at sample x_i, by a neural network, that gives us $\psi_{k,\theta}(x_i)$, $k = 2, \ldots, K$, where θ stands for network parameters.

2.3. Summary of SpecNet1

SpecNet1 (Shaham et al., 2018) adopts neural network parametrizations of eigenvectors of a normalized graph Laplacian, and the network is trained by minimizing an objective which is the variational form of the eigen-problem with an orthogonality constraint. Here we briefly review the three ingredients of the method of SpecNet1: the linear algebra optimization objective, the batch-based gradient evaluation scheme, and the neural network parametrization (including the orthogonalization layer).

Optimization objective. From a linear algebra point of view, SpecNet1 aims to find the first K eigenvectors of the symmetrically normalized Laplacian $L_{sym} := I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$ via solving the following orthogonally constrained optimization problem

$$\min_{\substack{Y^\top Y = nI \\ Y \in \mathbb{R}^{n \times K}}} f_1(Y) = \frac{1}{n} \mathrm{tr}\left(Y^\top (I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}) Y \right). \tag{4}$$

Note that in (4), Y is a real array as in the classical variational form of eigen-problem. It will be parametrized by a neural network below.

Mini-batch gradient evaluation. When the graph is large, memory constraint in practice usually prevents loading the full graph affinity matrix W into the memory or solving the full matrix Y over iterations. Thus, mini-batch is used in the training of SpecNet1. Given a batch of data points $\mathcal{B} \subset X$, SpecNet1 performs a single projected gradient descent step of a surrogate constrained optimization problem

$$\min_{\substack{Y_{\mathcal{B}}^\top Y_{\mathcal{B}} = bI \\ Y_{\mathcal{B}} \in \mathbb{R}^{b\times K}}} \tilde{f}_1(Y_{\mathcal{B}}) = \frac{1}{b}\mathrm{tr}\left(Y_{\mathcal{B}}^\top (I - \tilde{D}_{\mathcal{B}}^{-\frac{1}{2}} W_{\mathcal{B},\mathcal{B}} \tilde{D}_{\mathcal{B}}^{-\frac{1}{2}}) Y_{\mathcal{B}}\right), \tag{5}$$

where $W_{\mathcal{B},\mathcal{B}}$ is a submatrix of W with row and column index associated to data points in $\mathcal{B}$, $\tilde{D}_{\mathcal{B}}$ is the diagonal degree matrix of $W_{\mathcal{B},\mathcal{B}}$, and b is the number of data points in $\mathcal{B}$. We call (5) the "local evaluation scheme" of SpecNet1, as it only uses $W_{\mathcal{B},\mathcal{B}}$ retrieved from the matrix W when updating $Y_{\mathcal{B}}$. In this paper, we will propose and study three different mini-batch evaluation schemes in the training SpecNet2, and local scheme is one of the three. Corresponding to the other two mini-batch evaluation schemes of SpecNet2, which are called "neighbor" and "full" schemes respectively, we also study the counterpart schemes for SpecNet1. The details are explained in Section 4.2 (for neural network training) and Section 3.2 (for linear algebra optimization problem). Figure 3 gives a comparison of the different mini-batch schemes used to train SpecNet1 and SpecNet2. It can be seen that the performance of the local scheme is inferior to the other mini-batch evaluation schemes. Actually, in the linear algebra iterative solver (without neural network parametrization) of the variational eigen-problem, the relatively worse performance of the local scheme already presents, c.f. Figure 2. This is because only using the submatrix $W_{\mathcal{B},\mathcal{B}}$ may drastically lose the information of W when the batch size is small, especially when the graph is sparse. In contrast, the neighbor and full schemes use more information of W. See more in later sections.

Neural network parametrization. The neural network architecture in SpecNet1 (Shaham et al., 2018) contains two parts: First, a network mapping $\Phi_\theta : \mathbb{R}^m \to \mathbb{R}^K$, parametrized by θ, which maps an input data point $x_i \in \mathbb{R}^m$ to the K-dimensional space of spectral embedding coordinates; Second, an additional linear layer $\Xi \in \mathbb{R}^{K\times K}$, mapping from $\mathbb{R}^K$ to $\mathbb{R}^K$ and parametrized by the matrix Ξ, such that the composed mapping $\Phi_\theta(x_i)\Xi$ approximates the spectral embedding (eigenvectors), i.e.,

$$\Psi(x_i) \approx \Phi_\theta(x_i)\Xi. \tag{6}$$

The linear layer parametrized by Ξ is called the "orthogonalization layer". The neural network embedding of the entire dataset X is then represented as

$$Y(X) = \begin{pmatrix}(\Phi_\theta(x_1)\Xi)^\top & (\Phi_\theta(x_2)\Xi)^\top & \cdots & (\Phi_\theta(x_n)\Xi)^\top\end{pmatrix}^\top \in \mathbb{R}^{n\times K}, \tag{7}$$

where $(\Phi_\theta(x_i)\Xi)^\top$ is a column vector in $\mathbb{R}^K$ for each $i = 1, \ldots, n$.

Influence on the orthogonality constraint. We now explain a crucial difference when parametrizing Y by a neural network on the maintenance of the orthogonality constraint when using mini-batch. Note that the network representation (7) differs from a real array Y in that all rows of Y in (7) are related via network parametrization θ and Ξ. Using mini-batch, in a linear-algebra update of Y in (5), an update on $Y_{\mathcal{B}}$ would only change $Y_{\mathcal{B}}$ and leave the rest entries $Y_{\mathcal{B}^c}$ unchanged, where $\mathcal{B}^c = X \setminus \mathcal{B}$. In contrast, using the back-propagated gradient to update network parameters in (7), any update on θ and Ξ would change the embedding of all data points in $\mathcal{B}$ and $\mathcal{B}^c$.

In the training of SpecNet1, the neural network parameters θ and Ξ are updated separately in a mini-batch iteration. Specifically, at each mini-batch iteration, SpecNet1 first computes an overlapping matrix $\left((\Phi_\theta(x_i))^\top \Phi_\theta(x_j)\right)_{x_i,x_j\in\mathcal{B}}$ and its Cholesky factor L. Then the orthogonalization layer parameter Ξ is updated as $\Xi = \sqrt{b}(L^{-1})^\top$ to enforce the orthogonality constraint in (5). In the second step, it takes a gradient descent step or an equivalent optimization step of $\tilde{f}_1(Y_\mathcal{B})$ with respect to θ to update weights θ and keep the orthogonalization layer unchanged. Due to the dependence among rows of Y as in (7), we emphasize that such a mini-batch iteration also changes $Y_{\mathcal{B}^c}$ and the orthogonality constraint as in (4) cannot be exactly maintained.

We see in Figure 2 that in the linear algebra setting, SpecNet1 achieves good convergence with both the full and neighbor evaluation schemes; however, in the neural network setting, SpecNet1 with the neighbor scheme performs significantly worse than the full scheme, as shown in Figure 3. This is because in the neural network, at each iteration, the orthogonalization is computed based on the update only on the neighborhood of $\mathcal{B}$ for the neighbor scheme, while for the full scheme, the orthogonalization is computed on the updated output on the whole dataset X. On the other hand, due to memory constraints, we do not want to perform orthogonalization over all data samples at each iteration, we thus want to find a way such that we can still obtain good convergence with light memory budget. This motivates our development of SpecNet2 in this paper.

2.4. Other related works

The convergence of graph Laplacian eigenvectors to the limiting eigenfunctions of the manifold Laplacian operator has been proved in a series of works (Belkin and Niyogi, 2007; Von Luxburg et al., 2008; Burago et al., 2014; Singer and Wu, 2016) and recently in (García Trillos et al., 2020; Calder and Garcia Trillos, 2019; Dunson et al., 2021; Calder et al., 2020; Cheng and Wu, 2021b). The result shows that in the i.i.d. manifold data setting, the empirical graph Laplacian eigenvectors approximate the eigenfunctions evaluated on the data points in the large sample limit, where the kernel bandwidth is properly chosen to decrease to zero. The robustness of spectral embedding with input data noise has been shown in (Shen and Wu, 2020), among others. Based on these theories, the current work utilizes neural networks to approximate eigenfunctions so as to generalize to test data samples, due to that the eigenfunctions are the consistent limit of the eigenvectors of a properly constructed graph Laplacian.

For neural network methods to obtain dimension-reduced embedding, neural network embedding guided by pairwise relation was explored earlier in SiameseNet (Hadsell et al., 2006), where the training objective is heuristic. Using kernel affinity and spectral embedding to overcome the topological constraint in neural network embedding has been explored in (Mishne et al., 2019), and under the Variational Auto-encoder framework in (Li et al., 2020). The current paper differs from these auto-encoder methods in that SpecNet2, analogous to SpecNet1, outputs a dimension-reduced representation of data in a low-dimensional space, from which the training objective is computed via the graph Laplacian matrix.

3. Orthogonalization-free Iterative Eigensolver

We first investigate an orthogonalization-free iterative eigensolver, which serves as the loss function of SpecNet2 from a linear algebra point of view. Then three updating schemes incorporated with the coordinate descent method are proposed and compared, which later will be turned into the

mini-batch technique in the neural network in Section 4. Finally, the computational costs of three updating schemes are analyzed.

3.1. Unconstrained optimization

Recall that the spectral embedding is by computing the leading eigenvectors of the graph Laplacian $L_{rw} = I - D^{-1}W$. Equivalently, it aims to find K eigenvectors corresponding to the K largest eigenvalues of a generalized eigenvalue problem (GEVP) with the matrix pencil (W, D), where $K-1$ is the dimension of embedded space and D is the diagonal degree matrix associated with W. More explicitly, the generalized eigenvalue problem is of the form,

$$\begin{aligned} WU &= DU\Lambda, \\ U^\top DU &= n^2 I, \end{aligned} \tag{8}$$

where $\Lambda \in \mathbb{R}^{K\times K}$ is a diagonal matrix with its diagonal entries being the largest K eigenvalues of (W, D), $U \in \mathbb{R}^{n\times K}$ is the corresponding eigenvector matrix, and I denotes the identity matrix of size K. Throughout this paper, we assume the eigenvalue problem (8) has a nonzero eigengap between the K-th and $(K+1)$-th eigenvalues. Such a GEVP has been extensively studied and many efficient algorithms can be found in (Golub and Van Loan, 2013) and references therein.

In contrast to the constrained optimization problem as in SpecNet1, we propose to solve an unconstrained optimization problem to find the eigenpairs of (8). Many previous works (Liu et al., 2015; Lei et al., 2016; Li et al., 2019; Wang et al., 2019) adopt an unconstrained optimization problem for solving the standard eigenvalue problem, i.e., with $D = I$ in (8). The optimization problem therein minimizes $\left\|W - YY^\top\right\|_{\mathrm{F}}^2$ without any constraint on Y.

Extending the optimization problem to GEVP, we propose the following unconstrained optimization problem,

$$\min_{Y\in\mathbb{R}^{n\times K}} f_2(Y) = \frac{1}{n^2}\mathrm{tr}\left(-2Y^\top WY + \frac{1}{n^2}Y^\top DYY^\top DY\right). \tag{9}$$

The gradient of $f_2(Y)$ with respect to Y is

$$\nabla_Y f_2(Y) = -4\frac{W}{n}Y + 4\frac{D}{n^3}YY^\top DY. \tag{10}$$

Note that $\nabla_Y f_2(Y)$ in (10) is n times the actual gradient of $f_2(Y)$ in (9). The reason of normalizing $f_2(Y)$ and $\nabla_Y f_2(Y)$ in the way above is due to that we want to ensure an $O(1)$ limit, corresponding to the continuous limit of the eigen-problem, as $n \to \infty$. Details are explained in Appendix C.

Once we obtain the solution $\hat{Y}$ to (9), we can retrieve the approximations to eigenvectors of $D^{-1}W$, denoted as $\hat{U}$, by a single step of Rayleigh-Ritz method. More specifically, $\hat{U}$ is calculated as $\hat{U} = \hat{Y}O$, where $O \in \mathbb{R}^{K\times K}$ satisfies

$$\hat{Y}^\top W\hat{Y}O = \hat{Y}^\top D\hat{Y}O\hat{\Lambda}, \tag{11}$$

for diagonal matrix $\hat{\Lambda}$ as a refined approximation of the eigenvalues of (W, D).

Since the first trivial constant eigenvector of $D^{-1}W$ is typically not useful, one can skip solving for that in (9) by deflation, i.e., replacing W by $W - \eta\eta^\top$, where $\eta = \frac{d}{\|\sqrt{d}\|_2}$, and $d \in \mathbb{R}^n$ is a

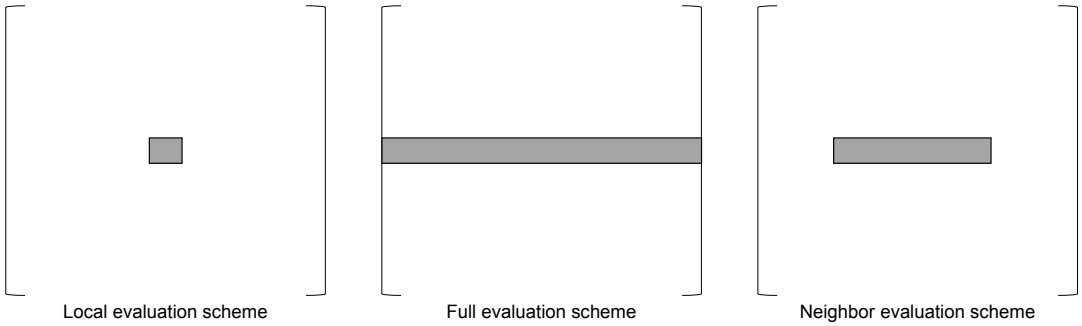

Figure 1: Entries of $W - \eta\eta^\top$ used (in gray block) in different gradient evaluation schemes at each batch step. See section 3.2 for detail.

column vector with $d_i = D_{i,i}$. Since D is positive-definite, Theorem 4 and other analysis results still hold except that we will skip the first trivial eigenvector in $Y^\star$, where $Y^\star$ is the minimizer of (9). Hence, for the rest of the paper beside Section 5, we will use

$$\min_{Y\in\mathbb{R}^{n\times K}} f_2(Y) = \frac{1}{n^2}\mathrm{tr}\left(-2Y^\top(W-\eta\eta^\top)Y + \frac{1}{n^2}Y^\top DYY^\top DY\right). \tag{12}$$

The gradient of $f_2(Y)$ is then

$$\nabla f_2(Y) = -4\frac{W-\eta\eta^\top}{n}Y + 4\frac{D}{n^3}YY^\top DY. \tag{13}$$

3.2. Different gradient evaluation schemes

In this subsection, we introduce efficient optimization methods of loss (12) by mini-match. Mini-batch is a mandatory technique in dealing with big datasets. Traditional mini-batch techniques randomly sample a mini-batch of data points $\mathcal{B} \subset X$, and solve the reduced problem on $\mathcal{B}$. Due to the fact that the computational cost to evaluate the term $Y^\top DY$ in (13) is very expensive for large n, we study different approximations to the gradient $\nabla f_2(Y)$, which yields three different gradient evaluation schemes. The visualization of these schemes in terms of the corresponding entries of $W - \eta\eta^\top$ is shown in Figure 1.

- ***Local evaluation scheme:*** One can evaluate the gradient on each mini-batch as

$$\nabla_{\mathcal{B}}\tilde{f}_2(Y_{\mathcal{B}}) = -\frac{4}{b}(W_{\mathcal{B},\mathcal{B}} - \tilde{\eta}_{\mathcal{B}}\tilde{\eta}_{\mathcal{B}}^\top)Y_{\mathcal{B}} + \frac{4}{b^3}\tilde{D}_{\mathcal{B}}Y_{\mathcal{B}}Y_{\mathcal{B}}^\top\tilde{D}_{\mathcal{B}}Y_{\mathcal{B}}, \tag{14}$$

  where $\tilde{f}_2(Y) = \frac{1}{b^2}\mathrm{tr}\left(-2Y^\top(W_{\mathcal{B},\mathcal{B}} - \tilde{\eta}_{\mathcal{B}}\tilde{\eta}_{\mathcal{B}}^\top)Y + \frac{1}{b^2}Y_{\mathcal{B}}^\top\tilde{D}_{\mathcal{B}}Y_{\mathcal{B}}Y_{\mathcal{B}}^\top\tilde{D}_{\mathcal{B}}Y_{\mathcal{B}}\right)$ is the objective function on $\mathcal{B}$, $b = |\mathcal{B}|$ is the cardinality of $\mathcal{B}$ and $\tilde{\eta} = \frac{\tilde{d}}{\|\sqrt{\tilde{d}}\|_2}$, and $\tilde{d} \in \mathbb{R}^{|\mathcal{B}|}$ is a column vector with $\tilde{d}_i = \tilde{D}_{\mathcal{B},i,i}$, i.e., the i-th diagonal entry of $\tilde{D}_{\mathcal{B}}$. The iterative algorithm then conducts the update as,

$$Y_{\mathcal{B}} = Y_{\mathcal{B}} - \alpha\nabla_{\mathcal{B}}\tilde{f}_2(Y_{\mathcal{B}}), \tag{15}$$

where $\alpha > 0$ is the stepsize.

Consider an example, where data points are relatively well-separated and the affinity matrix is very sparse. Such a mini-batch sampling is difficult to capture the neighbor points effectively and $W_{\mathcal{B},\mathcal{B}}$ for most $\mathcal{B}$ is nearly diagonal. Comparing (13) and (14), (14) is not a good approximation of (10) unless $\mathcal{B}$ is sufficiently large to capture the asymptotic behavior of the continuous limit. Therefore, optimizing the loss function using such a mini-batch technique requires either a big batch size or many iterations to achieve reasonable results.

- ***Full evaluation scheme:*** We evaluate the gradient on batch $\mathcal{B}$ as

$$\nabla_{\mathcal{B}} f_2(Y) = -\frac{4}{n}(W_{\mathcal{B},X} - \eta_{\mathcal{B}}\eta^\top)Y + \frac{4}{n^3} D_{\mathcal{B}} Y_{\mathcal{B}} Y^\top DY, \tag{16}$$

where $D_{\mathcal{B}}$ is the principle submatrix of D restricting to rows and columns in $\mathcal{B}$. And the update is then conducted as

$$Y_{\mathcal{B}} = Y_{\mathcal{B}} - \alpha \nabla_{\mathcal{B}} f_2(Y), \tag{17}$$

where $\alpha > 0$ is the stepsize.

This update is the block coordinate descent method applied to the proposed optimization problem. The computational burden lies in evaluating $\eta^\top Y$ and $Y^\top DY$ every iteration.

- ***Neighbor evaluation scheme:*** We introduce another way to conduct mini-batch on the gradient directly, which is block coordinate gradient descent with dynamic updating and plays an important role in the later neural network part. Given a sampled mini-batch $\mathcal{B}$, we define the neighborhood of $\mathcal{B}$ as,

$$\mathcal{N}(\mathcal{B}) = \{x_j \mid W_{i,j} \neq 0, x_i \in \mathcal{B}\}, \tag{18}$$

and we abbreviate it as $\mathcal{N}$. The gradient of batch $\mathcal{B}$ is evaluated as

$$\nabla_{\mathcal{B}} \bar{f}_2(Y) = -\frac{4}{n} W_{\mathcal{B},\mathcal{N}} Y_{\mathcal{N}} + \frac{4}{n} \eta_{\mathcal{B}} \eta^\top Y + \frac{4}{n^3} D_{\mathcal{B}} Y_{\mathcal{B}} Y^\top DY. \tag{19}$$

Note that $\eta^\top Y = \eta_{\mathcal{B}}^\top Y_{\mathcal{B}} + \eta_{\mathcal{B}^c}^\top Y_{\mathcal{B}^c}$ and $Y^\top DY = Y_{\mathcal{B}}^\top D_{\mathcal{B}} Y_{\mathcal{B}} + Y_{\mathcal{B}^c}^\top D_{\mathcal{B}^c} Y_{\mathcal{B}^c}$, where $\mathcal{B}^c = [n] \backslash \{i : x_i \in \mathcal{B}\}$. At each iteration, we only update $\eta^\top Y$ and $Y^\top DY$ on batch $\mathcal{B}$ in (19); that is, we update $\eta_{\mathcal{B}}^\top Y_{\mathcal{B}}$ for $\eta^\top Y$ and $Y_{\mathcal{B}}^\top D_{\mathcal{B}} Y_{\mathcal{B}}$ for $Y^\top DY$ using $Y_{\mathcal{B}}$ without touching the $\mathcal{B}^c$ part. The iterative algorithm then conducts the update as,

$$Y_{\mathcal{B}} = Y_{\mathcal{B}} - \alpha \nabla_{\mathcal{B}} \bar{f}_2(Y) \tag{20}$$

for α being the stepsize.

Similarly, we can evaluate the gradient of $f_1(Y)$ using three different evaluation schemes, whose detail can be found in Appendix A.1.

Remark 1 *In the linear algebra sense, both the gradient in* (16) *and the gradient in* (19) *are the same as the exact gradient of $f_2(Y)$ restricted to batch $\mathcal{B}$. When the neural network gets involved, the full and neighbor gradient evaluation schemes become different, which will be discussed in section 4.2. The gradient in* (14)*, however, is the gradient of $\tilde{f}_2$, which is not $\nabla_{\mathcal{B}} f_2$ unless $\mathcal{B} = X$.*

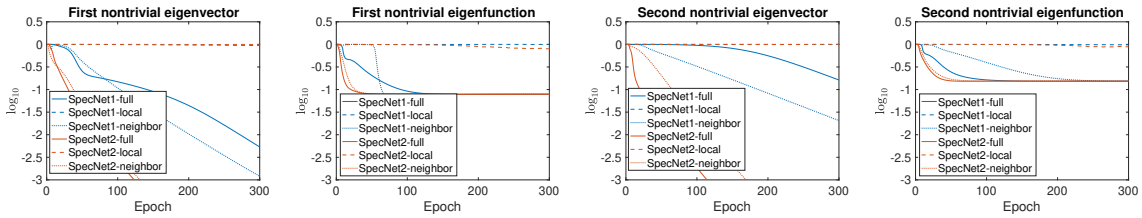

Figure 2: In the linear algebra problems of SpecNet1 and SpecNet2, relative errors of eigenvector (or eigenfunction, see titles of subfigures) approximations by different evaluation schemes on the one moon training dataset. The affinity matrix is 2000×2000 and detail about how the data is generated can be found in Appendix B. Legend refers to the update: SpecNet1-full (28), SpecNet1-local (26), SpecNet1-neighbor (30), SpecNet2-full (17), SpecNet2-local (15), SpecNet2-neighbor (20). The relative error for eigenfunction approximations is defined in (31), and the relative error for eigenvector approximation is defined as $\frac{\|\psi-\tilde{\psi}\|_2}{\|\psi\|_2}$, where ψ is the true eigenvector of (D, W) and $\tilde{\psi}$ is the corresponding column in $\hat{U}$ obtained through (11) that approximates ψ at each iteration.

We illustrate the convergence of three gradient evaluation schemes of f_1 and f_2 on a one moon dataset, the visualization of which is shown in Figure 7, and the results are shown in Figure 2. Here we choose constant stepsize for each method. The formulation for computing relative errors can be found in Appendix B.1.

Remark 2 *As shown in Figure 2, the full and neighbor gradient evaluation schemes of f_1 and f_2 can achieve good convergence, but the local scheme of either f_1 or f_2 does not converge. See Figure 3 for the illustration in the neural network setting.*

3.3. Computational cost of different schemes

We study the computational cost of different gradient evaluation schemes, taking f_2 as an example. Consider a sparse affinity matrix with on average s nonzeros on rows and columns.

For the local evaluation scheme, the computational cost for the first part in (14) is $O(|\mathcal{B}|^2 K)$ for $|\mathcal{B}|$ being the cardinality of $\mathcal{B}$ and the cost for the second part is $O(|\mathcal{B}| K^2)$. The overall computational cost per batch step is then $O(|\mathcal{B}|^2 K)$, assuming $|\mathcal{B}| \geq K$.

For the full evaluation scheme, the computational cost for the first part in (16) is $O(n |\mathcal{B}| K)$ and the cost for the second part is $O(nK^2)$. The overall computational cost per batch step is $O(n |\mathcal{B}| K)$, again assuming $|\mathcal{B}| \geq K$.

For the neighbor evaluation scheme, the computational cost for the first part in (19) is $O(s |\mathcal{B}| K)$, where s is the number of neighbors of $\mathcal{B}$. While the naïve computation of the third part in (19) costs $O(nK^2)$ operations, same as the full update scheme. When dynamic updating is taken into consideration at each step, only Y restricted to $\mathcal{B}$ is updated, and the matrix $Y^\top DY$ can be efficiently updated in $O(|\mathcal{B}| K^2)$ operations. Hence we can dynamically update the matrix $Y^\top DY$ throughout iterations and the computation of the third part in (19) is reduced to $O(|\mathcal{B}| K^2)$. Similarly, we can dynamically update the vector $\eta^\top Y$, and only those restricted to $\mathcal{B}$ is updated, and the second term can be updated in $O(|\mathcal{B}| K)$ operations. The overall computational cost per batch step is then $O(s |\mathcal{B}| K)$, assuming $s \geq K$.

4. Neural network parametrization and training

Inspired by the convergence results by two gradient evaluation schemes (16) and (19) as shown in Figure 2 as well as the theoretical guarantee for their convergence that we will prove later in Section 5, we propose a neural network that can incorporate the linear algebra formulations in Section 3.2.

4.1. Network parametrization of eigenfunctions

In section 2.2 we mention that eigenvectors of the graph Laplacian matrix can be viewed as the restriction of underlying eigenfunctions of a limiting operator on the dataset X. (Shaham et al., 2018) suggests we approximate those eigenfunctions by a neural network. In this paper, we use a feedforward fully-connected neural network, and it can be extended to other types of neural networks, for example, convolutional neural network. Suppose the neural network computes a map $G_\theta : \mathbb{R}^m \to \mathbb{R}^K$, where θ denotes the network weights. Let $Y = G_\theta(X)$, so that each coordinate of G_θ, $(G_\theta)_i$, $i = 1, \ldots, K$ is an approximation to an eigenfunction, and each column of $G_\theta(X)$ approximates an eigenvector of the graph Laplacian matrix. Our goal is to find a good approximation by training the neural network, SpecNet2, with the orthogonalization-free objective function $L(\theta) = f_2(Y)$.

4.2. Network Training for SpecNet2

In this subsection, we introduce the training of SpecNet2; that is, how to update θ to minimize $L(\theta) = f_2(Y)$. We have proposed three different gradient evaluation schemes in section 3.2 to calculate the gradients in the block coordinate descent method to minimize $f_2(Y)$ in the linear algebra setup. In the neural network setting, note that $\frac{\partial L(\theta)}{\partial \theta} = \nabla_Y f_2(Y) \cdot \frac{\partial G_\theta(X)}{\partial \theta}$, we can also incorporate these gradient evaluation schemes to evaluate $\nabla_Y f_2(Y)$ in the training of a neural network. Let $\mathcal{B} \subset X$ be the randomly sampled mini-batch, and $\mathcal{N}$ be the neighborhood of $\mathcal{B}$. Note that unlike in the linear algebra setup where we can only update Y on $\mathcal{B}$, we are updating θ for the neural network, such that once θ is updated, not only $G_\theta(\mathcal{B})$ is different but also $G_\theta(\mathcal{B}^c)$. We follow the notations as in section 3.2, and we have different gradient evaluation schemes for SpecNet2 as follows:

Local evaluation scheme: At each batch step, we can compute the neural network mapping of batch $\mathcal{B}$ as $Y_\mathcal{B} = G_\theta(\mathcal{B})$, so that we can obtain $\nabla_\mathcal{B} \tilde{f}_2(Y_\mathcal{B})$ by plugging $Y_\mathcal{B}$ into (14). Then we want to minimize $\mathrm{tr}\left(Y_\mathcal{B}(\theta)^\top \nabla_\mathcal{B} \tilde{f}_2(Y_\mathcal{B})\right)$ and update θ using the gradient of $\mathrm{tr}\left(Y_\mathcal{B}(\theta)^\top \nabla_\mathcal{B} \tilde{f}_2(Y_\mathcal{B})\right)$ with respect to θ through the chain rule, where inside the trace we write the first term $Y_\mathcal{B}$ as $Y_\mathcal{B}(\theta)$ to emphasize it is a function of θ; and the second term $\nabla_\mathcal{B} \tilde{f}_2(Y_\mathcal{B})$ is detached and viewed as constant.

Full evaluation scheme: At each batch step, we can compute $\nabla_\mathcal{B} f_2(Y_\mathcal{B})$ by plugging $Y_\mathcal{B}$ and Y into (16). Then we want to minimize $\mathrm{tr}\left(Y_\mathcal{B}(\theta)^\top \nabla_\mathcal{B} f_2(Y_\mathcal{B})\right)$ and update θ using the gradient of $\mathrm{tr}\left(Y_\mathcal{B}(\theta)^\top \nabla_\mathcal{B} f_2(Y_\mathcal{B})\right)$ through the chain rule. And similarly, inside the trace we only view $Y_\mathcal{B}(\theta)$ as a function of θ but $\nabla_\mathcal{B} f_2(Y_\mathcal{B})$ as constant when computing the gradient.

Neighbor evaluation scheme: We keep a record of two matrices $(YDY)_\star$ and Y_0 throughout the training, where they are initialized at the first iteration: $(YDY)_\star = Y^\top DY$ and $Y_0 = Y$, and detach both of them. At each batch step, we compute $Y_\mathcal{N} = G_\theta(\mathcal{N})$. Then we update $(YDY)_\star = (YDY)_\star - Y_0(\mathcal{N})^\top D_\mathcal{N} Y_0(\mathcal{N}) + Y_\mathcal{N}^\top D_\mathcal{N} Y_\mathcal{N}$ followed by an update of Y_0 on $\mathcal{N}$ as $Y_0(\mathcal{N}) = Y_\mathcal{N}$. Both matrices are again detached. The gradient of $f_2(Y)$ on $\mathcal{B}$ is then evaluated as

$$\nabla_\mathcal{B} \bar{f}_2(Y_\mathcal{B}) = -\frac{4}{n} W_{\mathcal{B},\mathcal{N}} Y_\mathcal{N} + \frac{4}{n} \eta_\mathcal{B} \eta^\top Y_0 + \frac{4}{n^3} D_\mathcal{B} Y_\mathcal{B} (YDY)_\star.$$

Then we minimize $\mathrm{tr}\left(Y_{\mathcal{B}}(\theta)^\top \nabla_{\mathcal{B}} \bar{f}_2(Y_{\mathcal{B}})\right)$ and update θ by computing the gradient of the quantity $\mathrm{tr}\left(Y_{\mathcal{B}}(\theta)^\top \nabla_{\mathcal{B}} \bar{f}_2(Y_{\mathcal{B}}))\right)$ by the chain rule. Similarly, inside the trace we only view $Y_{\mathcal{B}}(\theta)$ as a function of θ but $\nabla_{\mathcal{B}} \bar{f}_2(Y_{\mathcal{B}})$ as constant when computing the gradient.

Details about the network training for SpecNet1 can be found in Appendix A.2. With those different learning objective functions from different evaluation schemes, we can choose an optimizer, for example, SGD or Adam, with some user-selected learning rate to update the network weights θ. Detail for the choice in our experiments is introduced in Appendix B.

Remark 3 *Note that while $\nabla_{\mathcal{B}} \bar{f}_2(Y_{\mathcal{B}})$ in (19) is the exact gradient of $f_2(Y)$ on $\mathcal{B}$, the gradient $\nabla_{\mathcal{B}} \bar{f}_2(Y_{\mathcal{B}})$ we evaluate here in the neural network setting is no longer the exact gradient of $f_2(Y)$ of Y on $\mathcal{B}$, but only an approximation.*

5. Theoretical Analysis

In this section, we provide a theoretical guarantee for the performance of SpecNet2 by analyzing the optimization iterations to minimize (9). In Section 5.1, we discuss the energy landscape of (9). Through our analysis, we show that (9) is a nonconvex function whose local minima are global minima. In addition, we also give the explicit expression of the global minima of (9), which span the same space as that of the leading eigenvectors of matrix pencil (W, D), assuming D is positive definite. All analysis in this section holds for general symmetric matrix W and diagonal positive definite matrix D such that (W, D) has at least K positive eigenvalues. Hence our results apply to deflated matrix pencil $(W - \eta\eta^\top, D)$ as well. In Section 5.2, based on the energy landscape, we prove the global convergence of the gradient descent method with full and neighbor evaluation schemes for all initial points in a giant ball except a measure-zero set.

5.1. Analysis of energy landscape

The explicit form of the local minimizers of (9) are explicitly given in Theorem 4.

Theorem 4 *The local minimizers of (9) are of the form,*

$$Y^\star = U\Lambda^{\frac{1}{2}}Q, \tag{21}$$

where U and Λ are defined as in (8), and $Q \in \mathbb{R}^{K\times K}$ denotes an arbitrary orthogonal matrix.

The proof of Theorem 4 can be found in Appendix D.1. Through the analysis, we find that $f_2(Y)$ is nonconvex and all local minimizers span the same space as the eigenvectors of (W, D) associated with the K largest eigenvalues.

Corollary 5 *All local minimizers of (9) are global minimizers.*

The proof of Corollary 5 can be found in Appendix D.2. According to Theorem 4 and Corollary 5, the unconstrained optimization problem (9) does not have any spurious local minima and all local minimizers are global minimizers. Furthermore, the target of our problem, leading K eigenpairs of (W, D), can be extracted from the global minimizers through a single step Rayleigh-Ritz method, as mentioned in (11).

5.2. Global convergence

In this section, we prove the global convergence for the iterative schemes, (17) and (20), with full and neighbor gradient evaluation schemes, respectively. The energy landscape analysis in the previous section already hints at the global convergence from the gradient flow perspective. Here, we give a rigorous statement and its proof for the global convergence of our iterative scheme (17), which can be applied to (20) directly.

The difficulties of the convergence analysis come from two aspects. First, our objective function $f_2(Y)$ is a fourth-order polynomial of Y, and its Hessian is unbounded from above for $Y \in \mathbb{R}^{n\times K}$ and so is the Lipschitz constant. Second, the iterative scheme updates Y on different batches for different iterations. Hence the iterative mapping is not fixed across iterations.

We first prove a few lemmas to overcome these difficulties and then conclude the global convergence in Theorem 9. In Lemma 6, we define a giant ball with radius R and prove that our iterative scheme never leaves the ball. Given the bounded ball, we then have a bounded Lipschitz constant being defined in Lemma 7 and a nonempty set for stepsize α. Lemma 8 shows that our iterative scheme converges to first-order points of $f_2(Y)$. Combining these lemmas together with results in (Lee et al., 2019), we prove the global convergence.

We define a set of notations to simplify the statements of lemmas and theorem. The mini-batch technique partitions the dataset X into disjoint b batches. We denote the index set of mini-batch partitions as $\{S_1, S_2, \ldots, S_b\}$ such that $S_p \cap S_q = \emptyset$ for $p \neq q$ and $\cup_p S_p = [n]$. For an index i, i^c denotes the complement indices, i.e., $i^c = [n]\backslash\{i\}$. D_i denotes the i-th diagonal entry of D and Y_i denotes the i-th row of Y. $Y^{(\ell)}$ denotes the iteration variable at ℓ-th iteration. Further, we define two constants and a function depending on entries of W and D,

$$M_1 := \max_i \frac{W_{i,i} + \sqrt{W_{i,i}^2 + D_i \left\| W_{i,i^c} D_{i^c}^{-\frac{1}{2}} \right\|_2^2 + \frac{D_i}{2}}}{2D_i}, \; M_2 := \max_i \frac{W_{i,i}^2}{4D_i} + \frac{\left\| W_{i,i^c} D_{i^c}^{-\frac{1}{2}} \right\|_2^2}{4},$$

and $M(R) := 3\left(\max_i W_{i,i}^2 R^2 + \max_i D_i^2 \cdot n^2 K^2 R^6 + \max_i D_i \left\| W_{i,i^c} D_{i^c}^{-\frac{1}{2}} \right\|_2^2 \cdot nR^2\right)$, where the R will be the radius of the giant ball.

Lemma 6 *Let R be a constant such that $R \geq 2\sqrt{M_1}$ and α be the stepsize such that*

$$\alpha < \min\{\frac{-2M_2 + \sqrt{4M_2^2 + 3M(R)R^2}}{8M(R)}, \frac{1}{16M(R)}\}.$$

Then for any $Y^{(\ell)} \in W_0 = \{Y \in \mathbb{R}^{n\times K} : \max_i \left\| D_i^{\frac{1}{2}} Y_i \right\|_2 < R\}$, we have $Y^{(\ell+1)} \in W_0$.

Lemma 7 *For any $1 \leq i_1, i_2 \leq n$, $1 \leq k_1, k_2 \leq K$ and $Y \in W_0$ with $R \geq 2\sqrt{M_1}$, we have*

$$\left| \frac{\partial^2 f_2}{\partial Y_{i_1,k_1} \partial Y_{i_2,k_2}} \right| \leq 4 \max_{i,j} W_{i,j} + 4(n+K)R^2 \max_i D_i.$$

We define the upper bound in Lemma 7 as

$$L := 4 \max_{i,j} W_{i,j} + 4(n+K)R^2 \max_i D_i, \tag{22}$$

which is a Lipschitz constant of ∇f_2 in the coordinate sense.

We denote the iterative mapping as $Y^{(\ell+1)} = g_p(Y^{(\ell)})$, which is the block coordinate update at the ℓ-th iteration on batch S_p. Our iterative scheme then applies $g_1, \ldots, g_b$ in a cyclic way. When contiguous b iterations of our iterative scheme are applied, we could view it as a composed iterative mapping as,

$$g = g_b \circ g_{b-1} \circ \cdots \circ g_1, \tag{23}$$

and the corresponding iteration is

$$Y^{((i+1)b)} = g(Y^{(ib)}), \quad i = 0, 1, 2, \ldots. \tag{24}$$

Though mapping g is not explicitly shown in the statements of Lemma 8 and Theorem 9, their proofs rely on the detailed analysis of g.

Lemma 8 *Suppose α is sufficiently small such that $\alpha < \frac{1}{L}$. Then the iteration converges to first-order points, i.e.,*

$$\lim_{\ell \to \infty} \left\| \nabla f_2(Y^{(\ell)}) \right\| = 0.$$

With all these lemmas available, we then show the global convergence of our iterative scheme with full gradient evaluation scheme (17), in Theorem 9. The proof is based upon the stable manifold theorem (Lee et al., 2019).

Theorem 9 (Global Convergence) *Let $R \geq 2\sqrt{M_1}$ be a constant and suppose the stepsize satisfies that*

$$\alpha < \min \left\{ \frac{-2M_2 + \sqrt{4M_2^2 + 3M(R)R^2}}{8M(R)}, \frac{1}{16M(R)}, \frac{1}{KL \max_{i \in [b]} |S_i|} \right\}.$$

Then the iteration (17) converges to global minimizers of (9) for all $Y^{(0)} \in W_0$ up to an initial point set of measure zero.

Proofs of Lemma 6, Lemma 7, Lemma 8 and Theorem 9 are provided in Appendix D.3. We emphasize that the iterative scheme with full gradient evaluation scheme and neighbor gradient evaluation scheme are identical in the linear algebra sense. Hence the iterative scheme with the neighbor gradient evaluation scheme, (20), also admits the same global convergence property.

6. Numerical Experiments

We compare the performance of SpecNet2 with SpecNet1 through an ablation study: That is, all the setup of SpecNet1 is the same as SpecNet2 except that SpecNet1 has one additional orthogonalization layer appended to the output layer of SpecNet2. Details about the data generation, network architecture and parameters can be found in Appendix B. The code is available at https://github.com/ziyuchen7/SpecNet2.

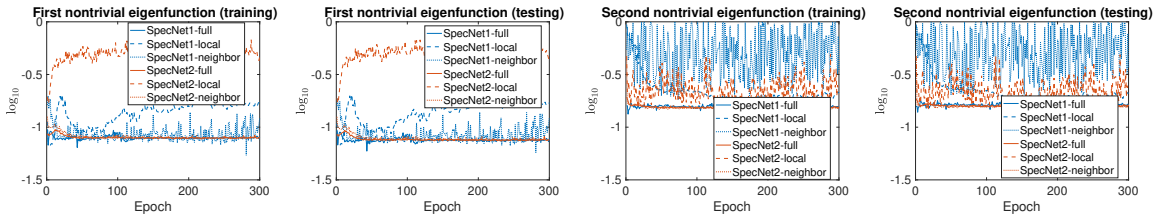

Figure 3: In the neural network setting of SpecNet1 and SpecNet2, relative errors of eigenfunction approximations by different evaluation schemes. SpecNet1-full, SpecNet1-local and SpecNet1-neighbor are introduced in Section A.2; SpecNet2-full, SpecNet2-local and SpecNet2-neighbor are introduced in Section 4.2. The relative error for training and testing is defined below (31).

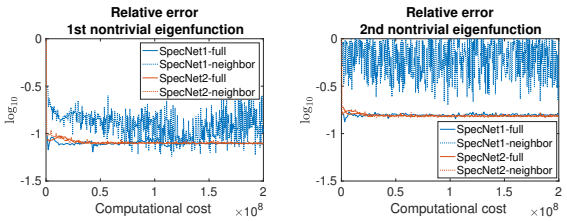

Figure 4: Results of full and neighbor schemes of SpecNet1 and SpecNet2 on the training data, where we rescale the x-axis to the computational cost.

6.1. One moon data

The visualization of one moon data can be found in Figure 7. Training data and testing data both consists of 2000 samples. Figure 3 demonstrates the performance of both methods with all three gradient evaluation schemes on the one moon data. We also compare the computational efficiency of full and neighbor schemes in Figure 4, and its detail can be found in Appendix B.1. We observe in Figure 3 that SpecNet1-full, SpecNet2-full and SpecNet2-neighbor can provide good approximations to the first two nontrivial eigenfunctions; SpecNet1-local, SpecNet2-local, and SpecNet1-neighbor give poor approximations as their relative errors are significantly larger. In Figure 4, we see that the relative error for the first nontrivial eigenfunction by SpecNet2-neighbor reaches the plateau earlier than SpecNet2-full in terms of the computational cost, while they can achieve similar accuracy. We also show the embedding results provided by different methods in Figure 8 in the Appendix.

6.2. Two moons data

We compare the performance and stability of SpecNet2 with SpecNet1 through an unsupervised clustering task on a two moons dataset (visualized in Figure 7) that contains 2000 training samples and 2000 testing samples. Due to the savings in memory and computational cost, we only compare SpecNet2-neighbor with SpecNet1-neighbor in this example. Figure 5 shows the classification performance of SpecNet1-neighbor and SpecNet2-neighbor over 10 different realizations of the neural network. We observe that though the average curves provided by SpecNet1-neighbor and SpecNet2-neighbor are close to each other, the variance of SpecNet1-neighbor is much larger than

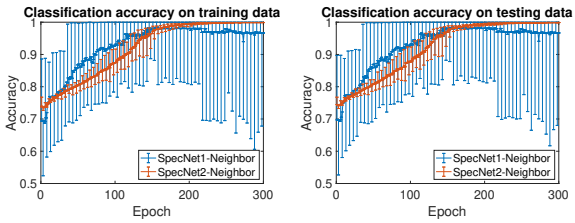

Figure 5: Classification results on two moons dataset. Average classification accuracy and error-bars are plotted over 10 different network initializations. At each epoch, the classification and its accuracy is computed in an unsupervised way.

that of SpecNet2-neighbor. Hence, we conclude that SpecNet2-neighbor is able to achieve similar average classification accuracy as SpecNet1-neighbor but with much higher reliability.

6.3. MNIST data

In this experiment, we use 20,000 samples of MNIST data (gray-scale images of hand-written digits which are of size 28×28) as the training set and 10,000 samples for testing. We construct the adjacency matrix A of an kNN graph on the training set by setting $A_{i,j} = 1$ if the j-th training sample is within k nearest neighbors of the i-th training sample and $A_{i,j} = 0$ otherwise, and we use $k = 16$. The affinity matrix W is obtained by setting $W = \frac{1}{2}(A + A^\top)$. We compare the performance of SpecNet1-local with SpecNet2-neighbor with different batch sizes. Specifically, the batch sizes for SpecNet2-neighbor are 2, 4 and 8 and those for SpecNet1-local are 45, 90, 180 (the average numbers of neighbors of a batch of size 2, 4 and 8 are about 45, 90 and 180 respectively).

Figure 6 shows the losses f_1 and f_2 (defined in (4) and (12) respectively) over the training epochs. We observe that though SpecNet2-neighbor has larger variance compared to SpecNet1-local, SpecNet2-neighbor achieves better performance in average when the batch size is small, e.g., comparing SpecNet2-neighbor with batch size 2 with SpecNet1-local with batch size 45. See Figure 11 in Appendix B.3 for the embedding result.

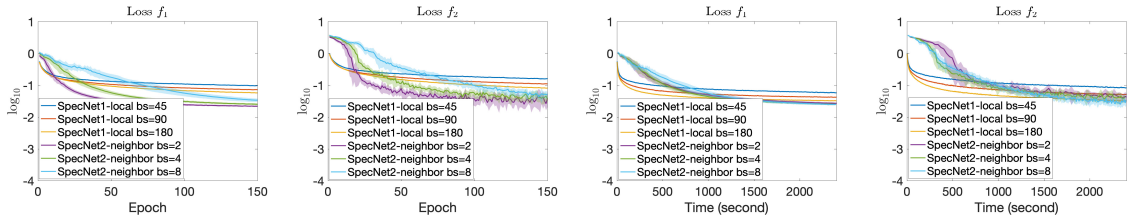

Figure 6: MNIST datset: plot of two different losses $\log_{10}(f_1(Y) - f_1^\star)$ and $\log_{10}(f_2(Y) - f_2^\star)$ over epochs (left two subfigures) and over time (right two subfigures), with f_1 and f_2 defined in (4) and (12). Networks are trained on 20000 MNIST images on a 2021 14-inch Macbook Pro with an 8-core CPU. See Appendix B.3 for the detail.

7. Discussion

The current paper develops a new spectral network approach, which removes the orthogonalization layer in the original SpectralNet (Shaham et al., 2018). We first proposed an unconstrained orthogonalization-free optimization problem to reveal the leading K eigenvectors of a given matrix pencil (W, D). Iterative algorithms with three different mini-batch gradient evaluation schemes, namely local scheme, full scheme, and neighbor scheme, are proposed and extended to the neural network training setting. The energy landscape of the optimization problem is analyzed, and the global convergence to the minimizer is guaranteed for all initial points up to a measure zero set. Numerically, SpecNet2-neighbor achieves almost the same accuracy as SpecNet1-full and SpecNet2-full while its computational cost is significantly lower due to the neighborhood tracking trick.

There are several directions to extend the work. Theoretically, the current analysis is in the sense of linear algebra. Further analysis is needed to obtain optimization guarantee with the neural network parametrization. Method-wise, the current approach assumes a graph affinity matrix is provided, while in practice when only data samples are provided one also needs to explore how to efficiently construct the graph affinity, which can be used by the SpecNet2 neural network. Finally, application to other real-world datasets could be explored, which would potentially leads to more efficient implementations.

Acknowledgement

The work is supported by NSF DMS-2031849. Z.C. is supported by Simons Foundation Award and Simons Foundation - Math+X Investigators; X.C. is partially supported by NSF DMS-2007040, NIH and the Alfred P. Sloan Foundation. We thank the anonymous reviewers for their careful reading of our manuscript and their many insightful comments and suggestions.

References

Mohamed-Ali Belabbas and Patrick J Wolfe. On landmark selection and sampling in high-dimensional data analysis. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 367(1906):4295–4312, 2009.

Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

Mikhail Belkin and Partha Niyogi. Convergence of laplacian eigenmaps. In *Advances in Neural Information Processing Systems*, pages 129–136, 2007.

Amit Bermanis, Amir Averbuch, and Ronald R Coifman. Multiscale data sampling and function extension. *Applied and Computational Harmonic Analysis*, 34(1):15–29, 2013.

Dmitri Burago, Sergei Ivanov, and Yaroslav Kurylev. A graph discretization of the laplace-beltrami operator. *Journal of Spectral Theory*, 4(4):675–714, 2014.

Jeff Calder and Nicolas Garcia Trillos. Improved spectral convergence rates for graph laplacians on epsilon-graphs and k-nn graphs. *arXiv preprint arXiv:1910.13476*, 2019.

Jeff Calder, Nicolas Garcia Trillos, and Marta Lewicka. Lipschitz regularity of graph laplacians on random data clouds. *arXiv preprint arXiv:2007.06679*, 2020.

Xiuyuan Cheng and Hau-Tieng Wu. Convergence of graph laplacian with knn self-tuned kernels. *Information and Inference: A Journal of the IMA*, 2021a.

Xiuyuan Cheng and Nan Wu. Eigen-convergence of gaussian kernelized graph laplacian by manifold heat interpolation. *arXiv preprint arXiv:2101.09875*, 2021b.

Ronald R Coifman and Stéphane Lafon. Diffusion maps. *Applied and computational harmonic analysis*, 21(1):5–30, 2006.

David B Dunson, Hau-Tieng Wu, and Nan Wu. Spectral convergence of graph laplacian and heat kernel reconstruction in l_∞ from random samples. *Applied and Computational Harmonic Analysis*, 55:282–336, 2021.

Weiguo Gao, Yingzhou Li, and Bichen Lu. Triangularized orthogonalization-free method for solving extreme eigenvalue problems, may 2020. http://arxiv.org/abs/2005.12161.

Weiguo Gao, Yingzhou Li, and Bichen Lu. Global convergence of triangularized orthogonalization-free method, 2021. URL https://arxiv.org/abs/2110.06212v1.

Nicolás García Trillos, Moritz Gerlach, Matthias Hein, and Dejan Slepčev. Error estimates for spectral convergence of the graph laplacian on random geometric graphs toward the laplace–beltrami operator. *Foundations of Computational Mathematics*, 20(4):827–887, 2020.

Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 4th edition, 2013. ISBN 9781421407944.

Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006.

Jason D Lee, Ioannis Panageas, Georgios Piliouras, Max Simchowitz, Michael I Jordan, and Benjamin Recht. First-order methods almost always avoid strict saddle points. *Mathematical programming*, 176(1):311–337, 2019.

Qi Lei, Kai Zhong, and Inderjit S. Dhillon. Coordinate-wise power method. In D D Lee, M Sugiyama, U V Luxburg, I Guyon, and R Garnett, editors, *Adv. Neural Inf. Process. Syst. 29*, pages 2064–2072. Curran Associates, Inc., 2016.

Henry Li, Ofir Lindenbaum, Xiuyuan Cheng, and Alexander Cloninger. Variational diffusion autoencoders with random walk sampling. In *European Conference on Computer Vision*, pages 362–378. Springer, 2020.

Yingzhou Li, Jianfeng Lu, and Zhe Wang. Coordinatewise descent methods for leading eigenvalue problem. *SIAM J. Sci. Comput.*, 41(4):A2681–A2716, jan 2019. doi: 10.1137/18M1202505.

Xin Liu, Zaiwen Wen, and Yin Zhang. An efficient Gauss-Newton algorithm for symmetric low-rank product matrix approximations. *SIAM J. Optim.*, 25(3):1571–1608, 2015. ISSN 10526234. doi: 10.1137/140971464.

Francesco Mauri, Giulia Galli, and Roberto Car. Orbital formulation for electronic-structure calculations with linear system-size scaling. *Phys. Rev. B*, 47(15):9973, apr 1993. ISSN 01631829. doi: 10.1103/PhysRevB.47.9973.

Gal Mishne, Uri Shaham, Alexander Cloninger, and Israel Cohen. Diffusion nets. *Applied and Computational Harmonic Analysis*, 47(2):259–285, 2019.

Evert J Nyström. Über die praktische auflösung von integralgleichungen mit anwendungen auf randwertaufgaben. *Acta Mathematica*, 54:185–204, 1930.

Pablo Ordejon, David A. Drabold, Matthew P. Grumbach, Richard M. Martin, Pablo Ordejón, David A. Drabold, Matthew P. Grumbach, and Richard M. Martin. Unconstrained minimization approach for electronic computations that scales linearly with system size. *Phys. Rev. B*, 48 (19):14646, nov 1993. ISSN 01631829. doi: 10.1103/PhysRevB.48.14646.

Uri Shaham, Kelly Stanton, Henry Li, Ronen Basri, Boaz Nadler, and Yuval Kluger. Spectralnet: Spectral clustering using deep neural networks. In *International Conference on Learning Representations*, 2018.

Chao Shen and Hau-Tieng Wu. Scalability and robustness of spectral embedding: landmark diffusion is all you need. *arXiv preprint arXiv:2001.00801*, 2020.

Amit Singer and Hau-Tieng Wu. Spectral convergence of the connection laplacian from random samples. *Information and Inference: A Journal of the IMA*, 6(1):58–123, 2016.

Ulrike Von Luxburg, Mikhail Belkin, and Olivier Bousquet. Consistency of spectral clustering. *The Annals of Statistics*, pages 555–586, 2008.

Zhe Wang, Yingzhou Li, and Jianfeng Lu. Coordinate descent full configuration interaction. *J. Chem. Theory Comput.*, 15(6):3558–3569, jun 2019. doi: 10.1021/acs.jctc.9b00138. URL http://pubs.acs.org/doi/10.1021/acs.jctc.9b00138.

Christopher Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. *Advances in neural information processing systems*, 13, 2000.

Appendix A. Gradient evaluation schemes for SpecNet1

A.1. Gradient evaluation schemes for f_1

The gradient descent of the formulation in (4) can be written as $Y = Y - 2\alpha(I - D^{-\frac{1}{2}}WD^{-\frac{1}{2}})Y$, where α is the stepsize. We multiply both sides by $D^{-\frac{1}{2}}$ on the left and we have $D^{-\frac{1}{2}}Y = D^{-\frac{1}{2}}Y - 2\alpha(I - D^{-1}W)D^{-\frac{1}{2}}Y$. So instead of updating Y, we update $\tilde{Y} := D^{-\frac{1}{2}}Y$ at each iteration, i.e., $\tilde{Y} = \tilde{Y} - 2\alpha(I - D^{-1}W)\tilde{Y}$. And the constraint will be $\tilde{Y}^\top D\tilde{Y} = n^2 I$. To keep the consistency of the notation, we will abuse the notion of gradient and still call that $2(I - D^{-1}W)\tilde{Y}$ as the gradient of f_1 for the rest of the paper, while keeping in mind that we are updating $D^{-\frac{1}{2}}Y$ in (4). Then the gradient evaluation schemes for f_1 with orthogonalization constraint works as follows

- ***Local evaluation scheme:*** One can evaluate the gradient on each mini-batch as

$$\nabla_{\mathcal{B}}\tilde{f}_1(Y_{\mathcal{B}}) = 2(I - \tilde{D}_{\mathcal{B}}^{-1}W_{\mathcal{B},\mathcal{B}})Y_{\mathcal{B}}. \tag{25}$$

  The iterative algorithm then conducts the update as,

$$Y_{\mathcal{B}} = Y_{\mathcal{B}} - \alpha\nabla_{\mathcal{B}}\tilde{f}_1(Y_{\mathcal{B}}), \tag{26}$$

  where $\alpha > 0$ is the stepsize, which is followed by an orthogonalization step $Y = bYR^{-1}$, where $\tilde{D}_{\mathcal{B}}^{\frac{1}{2}}Y_{\mathcal{B}} = QR$ is the QR decomposition of $\tilde{D}_{\mathcal{B}}^{\frac{1}{2}}Y_{\mathcal{B}}$.

- ***Full evaluation scheme:*** We evaluate the gradient on batch $\mathcal{B}$ as

$$\nabla_{\mathcal{B}}f_1(Y) = 2(I - D_{\mathcal{B}}^{-1}W_{\mathcal{B},X})Y, \tag{27}$$

  and the update is then conducted as

$$Y_{\mathcal{B}} = Y_{\mathcal{B}} - \alpha\nabla_{\mathcal{B}}f_1(Y), \tag{28}$$

  where $\alpha > 0$ is the stepsize. It follows by an orthogonalization step $Y = nYR^{-1}$, where $D^{\frac{1}{2}}Y = QR$ is the QR decomposition of $D^{\frac{1}{2}}Y$. The full scheme of f_1 is equivalent to the power method with mini-batch and dynamic shift.

- ***Neighbor evaluation scheme:*** The gradient of batch $\mathcal{B}$ is evaluated as

$$\nabla_{\mathcal{B}}\bar{f}_1(Y_{\mathcal{N}}) = 2(Y_{\mathcal{B}} - D_{\mathcal{B}}^{-1}W_{\mathcal{B},\mathcal{N}}Y_{\mathcal{N}}). \tag{29}$$

  The iterative algorithm then conduct the update as,

$$Y_{\mathcal{B}} = Y_{\mathcal{B}} - \alpha\nabla_{\mathcal{B}}\bar{f}_1(Y_{\mathcal{N}}) \tag{30}$$

  for α being the stepsize.

  It follows by an orthogonalization step such that $Y = nY(L^{-1})^\top$, where $Y^\top DY = LL^\top$ is the Cholesky decomposition of $Y^\top DY$, and as in (19), we only update $Y^\top DY$ on $\mathcal{B}$ at each iteration.

A.2. Network Training for SpecNet1

Different from SpecNet2, we have one additional orthogonalization layer, denoted by $R \in \mathbb{R}^{K\times K}$, appended to G_θ for SpecNet1. Therefore, the mapping given by SpecNet1 is $x \mapsto G_\theta(x)\cdot R$ for any $x \in \mathbb{R}^m$. We also introduce the training of SpecNet1 that incorporates those gradient evaluation schemes in section A.1.

Local evaluation scheme: At each batch step, we compute $Y_{\mathcal{B}} = G_\theta(\mathcal{B})$. The orthogonalization layer is computed as in the QR factorization $\tilde{D}_{\mathcal{B}}^{\frac{1}{2}}Y_{\mathcal{B}} = QR$, and the output after that is then $\tilde{Y}_{\mathcal{B}} = bY_{\mathcal{B}}R^{-1}$. So we can obtain $\nabla_{\mathcal{B}}\tilde{f}_1(\tilde{Y}_{\mathcal{B}})$ by plugging $\tilde{Y}_{\mathcal{B}}$ into (25). Then we minimize $\mathrm{tr}\left(\tilde{Y}_{\mathcal{B}}(\theta)^\top\nabla_{\mathcal{B}}\tilde{f}_1(\tilde{Y}_{\mathcal{B}}))\right)$ and update θ using the gradient of $\mathrm{tr}\left(\tilde{Y}_{\mathcal{B}}(\theta)^\top\nabla_{\mathcal{B}}\tilde{f}_2(Y_{\mathcal{B}})\right)$ with respect to θ through the chain rule, where inside the trace we write the first term $\tilde{Y}_{\mathcal{B}}$ as $\tilde{Y}_{\mathcal{B}}(\theta)$ to emphasize it is a function of θ; and the second term $\nabla_{\mathcal{B}}\tilde{f}_2(\tilde{Y}_{\mathcal{B}})$ is detached and viewed as constant. We shall

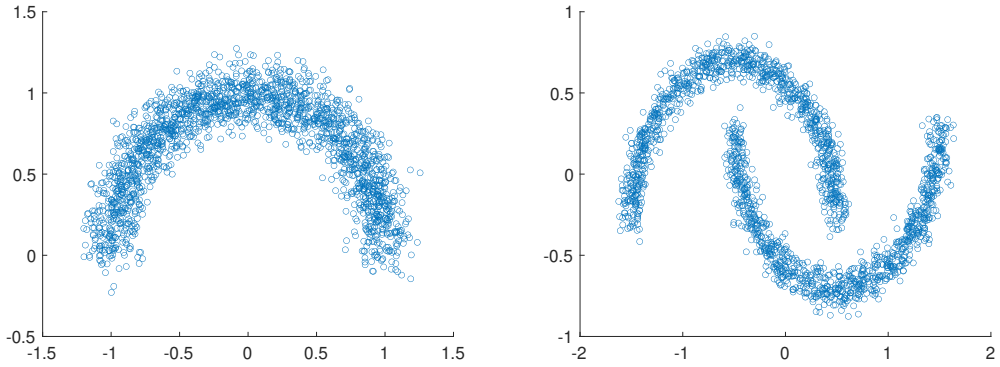

Figure 7: Visualizations of the one moon and two moons training dataset. For each example, the testing set are i.i.d. sampled from the same distribution as the training set.

mention that SpecNet1 with local evaluation scheme is the method in the original SpecNet1 paper (Shaham et al., 2018), except that here we also update weights of the orthogonalization layer using the gradient by back-propagation, which turns out to improve the performance of the original SpecNet1 significantly.

Full evaluation scheme: At each batch step, we compute $Y = G_\theta(X)$. The orthogonalization layer is computed as in the QR factorization $\tilde{D}^{\frac{1}{2}}Y = QR$, and the output after that is then $\tilde{Y} = nYR^{-1}$. So we can obtain $\nabla_{\mathcal{B}} f_1(\tilde{Y})$ by plugging $\tilde{Y}$ into (27). Then we want to minimize $\mathrm{tr}\left(\tilde{Y}_{\mathcal{B}}(\theta)^\top \nabla_{\mathcal{B}} f_1(\tilde{Y}))\right)$ and update θ using the gradient of $\mathrm{tr}\left(\tilde{Y}_{\mathcal{B}}(\theta)^\top \nabla_{\mathcal{B}} f_1(\tilde{Y})\right)$ through the chain rule. And similarly, inside the trace we only view $\tilde{Y}_{\mathcal{B}}(\theta)$ as a function of θ but $\nabla_{\mathcal{B}} f_1(\tilde{Y})$ as constant when computing the gradient.

Neighbor evaluation scheme: We keep a record of two matrices $(YDY)_\star$ and Y_0 throughout the training, where they are initialized at the first iteration: $(YDY)_\star = Y^\top DY$ and $Y_0 = Y$, and detach both of them. At each batch step, we compute $Y_{\mathcal{N}} = G_\theta(\mathcal{N})$. Then we update $(YDY)_\star = (YDY)_\star - Y_0(\mathcal{N})^\top D_{\mathcal{N}} Y_0(\mathcal{N}) + Y_{\mathcal{N}}^\top D_{\mathcal{N}} Y_{\mathcal{N}}$ followed by an update of Y_0 on $\mathcal{N}$ as $Y_0(\mathcal{N}) = Y_{\mathcal{N}}$. Both matrices are again detached. The orthogonalization layer is computed as in the Cholesky factorization $(YDY)_\star = LL^\top$, and the output after that is then $\tilde{Y}_{\mathcal{N}} = nY_{\mathcal{N}}(L^{-1})^\top$. So we can obtain $\nabla_{\mathcal{B}} \bar{f}_1(\tilde{Y}_{\mathcal{N}})$ by plugging $\tilde{Y}_{\mathcal{N}}$, which includes $\tilde{Y}_{\mathcal{B}}$, into (29). Then we minimize $\mathrm{tr}\left(\tilde{Y}_{\mathcal{B}}(\theta)^\top \nabla_{\mathcal{B}} \bar{f}_1(\tilde{Y}_{\mathcal{N}})\right)$ and update θ by computing the gradient of $\mathrm{tr}\left(\tilde{Y}_{\mathcal{B}}(\theta)^\top \nabla_{\mathcal{B}} \bar{f}_1(\tilde{Y}_{\mathcal{N}})\right)$ by the chain rule. Similarly, inside the trace we only view $\tilde{Y}_{\mathcal{B}}(\theta)$ as a function of θ but $\nabla_{\mathcal{B}} \bar{f}_1(\tilde{Y}_{\mathcal{N}})$ as constant when computing the gradient.

Appendix B. Details of the numerical examples in Section 6

B.1. One moon data

Data generation: The training set consists of $n = 2000$ points in $\mathbb{R}^2$, and is generated by $x_i = (\cos\eta_i, \sin\eta_i) + \xi_i$, $i = 1, \ldots, 2000$, where η_i are i.i.d. uniformly sampled on $[0, \pi]$ and ξ_i are i.i.d. Gaussian random variables of dimension two drawn from $\mathcal{N}(0, 0.01 I_2)$. The testing set consists of 2000 points and is generated in the same way as the training set with a different realization.

The sparse affinity matrix associated with the training set is generated via Gaussian kernel with bandwidth $\sigma = 0.1$, and truncated at threshold 0.6.

Network training: We use a fully-connected feedforward neural network with a single 128-unit hidden layer:

SpecNet1: $2 \xrightarrow{\text{fc}} 128 - \text{ReLU} \xrightarrow{\text{linear}} 3 \xrightarrow{\text{orthogonal}} 3$;

SpecNet2: $2 \xrightarrow{\text{fc}} 128 - \text{ReLU} \xrightarrow{\text{linear}} 2$,

where "fc" stands for fully-connected layers. The batch size is 4, and we use Adam as the optimizer with learning rate 10^{-3} for SpecNet2 and 10^{-4} for SpecNet1.

Error evaluation: We evaluate the network approximation of the first two nontrivial eigenfunctions by computing the relative errors of the output functions of the trained network with the underlying true eigenfunctions. The true eigenfunctions are constructed via a fine grid discretization of the continuous operator. We introduce how the relative error is calculated. Suppose $\psi \in \mathbb{R}^n$ is the limiting eigenfunction evaluated at $\{x_i\}$, and $\tilde{\psi} \in \mathbb{R}^n$ is the network output function, which approximates ψ, evaluated at $\{x_i\}$. The relative error $\tau(\tilde{\psi}, \psi)$ of $\tilde{\psi}$ with respect to ψ is defined as

$$\tau(\tilde{\psi}, \psi) := \frac{\left\|\psi - \beta\tilde{\psi}\right\|_2}{\|\psi\|_2}, \tag{31}$$

where $\beta = \frac{\psi^\top \tilde{\psi}}{\|\tilde{\psi}\|_2^2}$ is the number that minimizes $\left\|\psi - \beta\tilde{\psi}\right\|_2$ serving the role of aligning two eigenfunctions. To evaluate the relative error on the training set, ψ will be the limiting eigenfunction evaluated at training samples and $\tilde{\psi}$ is the corresponding network output function evaluated at training samples; the relative error on the testing set can be defined similarly on testing samples.

To further compare the computational efficiency of gradient evaluation schemes, we plot the relative errors against the leading computational cost in Figure 4, where the leading computational costs are estimated as: $\frac{n^2}{|\mathcal{B}|} \cdot$ epoch for the full gradient evaluation scheme; $\frac{n|\mathcal{N}|}{|\mathcal{B}|} \cdot$ epoch for the neighbor gradient evaluation scheme, where the averaged number of neighbors of a batch of size 4 is about 620. We also show the embedding results provided by different methods in Figure 8.

Moreover, we seek to solve the generalized eigenvalue problem (W, D), corresponding to the random walk Laplacian $D^{-1}W$ in both the SpecNet1 and SpecNet2 implementation here. We can also approximate the eigenvalue problem of $D^{-\frac{1}{2}}WD^{-\frac{1}{2}}$, the symmetrically normalized Laplacian, in our implementation: that is, setting $W = D^{-\frac{1}{2}}WD^{-\frac{1}{2}}$ and $D = I$ in (9) for SpecNet2 and multiply the output by $D^{-\frac{1}{2}}$ on the left to get back to the generalized eigenvalue problem (W, D), which approximates eigenfunctions of a continuous limiting operator. We show the result in Figure 9 for the full scheme for the relative errors of approximations of first two nontrivial eigenfunctions on the training set. We see that the performance is similar if we switch from using (W, D) to $D^{-\frac{1}{2}}WD^{-\frac{1}{2}}$ for the training objective (we may need to change the learning rate after switch, but we did not tune it here).

B.2. Two moons data

Data generation: The two moons training set consists of $n = 2000$ points in $\mathbb{R}^2$. One piece of moons is generated by the equation $x_i = (\cos\eta_i - 0.5, \sin\eta_i - 0.3) + \xi_i$, $i = 1, \ldots, 1000$, and the other piece is generated by $x_i = (-\cos\eta_i + 0.5, -\sin\eta_i + 0.3) + \xi_i$, $i = 1001, \ldots, 2000$, where η_i are i.i.d. uniformly sampled on $[0, \pi]$ and ξ_i are i.i.d. Gaussian random variables of dimension two

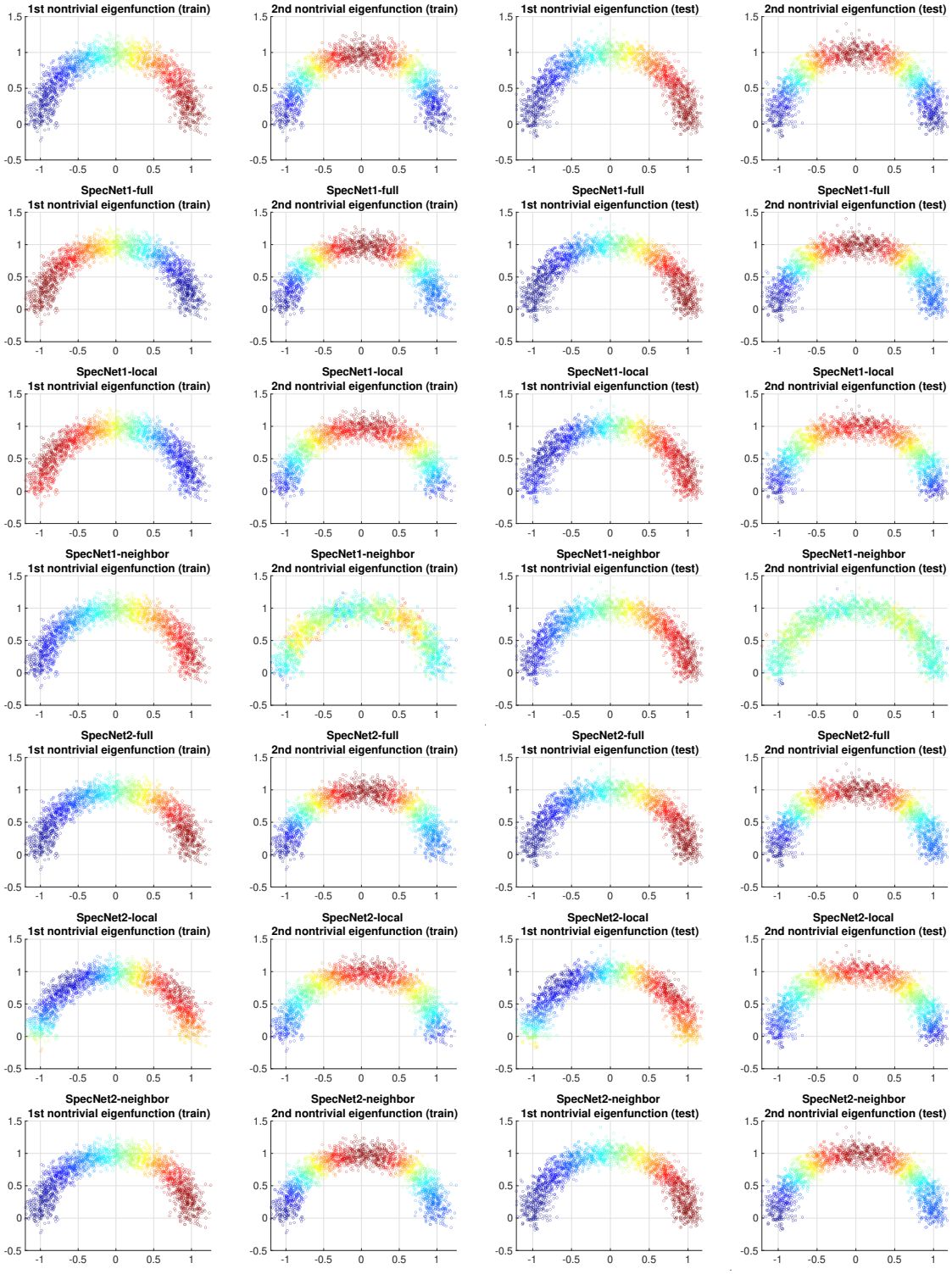

Figure 8: One moon datset: embeddings by different methods using the first two nontrivial eigenfunctions. The first row is the ground truth.

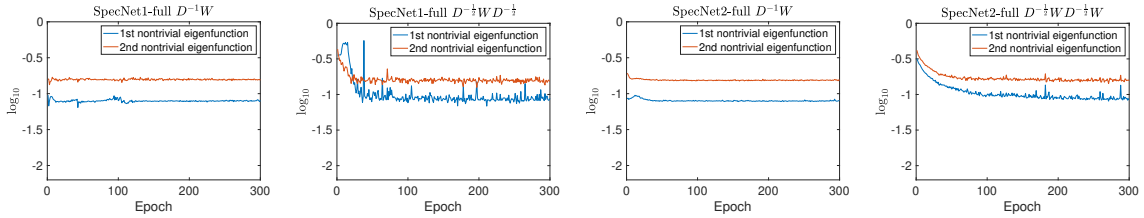

Figure 9: One moons datset: network approximations of eigenfunctions using different Laplacian matrix on the training set.

drawn from $\mathcal{N}(0, 0.0036I_2)$. The testing set consists of 2000 points and is generated in the same way as the training set with a different realization. The sparse affinity matrix associated with the traning set is generated via Gaussian kernel with bandwidth $\sigma = 0.15$, and truncated at threshold 0.08.

Network training: We use a fully-connected feedforward neural network with single 128-unit hidden layer:

SpecNet1: $2 \xrightarrow{\text{fc}} 128 - \text{ReLU} \xrightarrow{\text{linear}} 2 \xrightarrow{\text{orthogonal}} 2$;

SpecNet2: $2 \xrightarrow{\text{fc}} 128 - \text{ReLU} \xrightarrow{\text{linear}} 1$.

The batch size is 4 (the average number of neighbors of batches of size 4 in the sparse affinity matrix is about 670), and we use the Adam as the optimizer with learning rate 10^{-3} for SpecNet2 and 10^{-5} for SpecNet1.

Error evaluation: The classification is done in an unsupervised way. Specifically, we label the training and testing samples that are generated from one piece of moons as 1; label those samples generated from the other piece of moons as 2; use them as the ground truth and train the network on the training data without labels. Let us take the classification accuracy on the training data as an example, we evaluate the network output function corresponding to the first nontrivial eigenvector on the training set and perform the standard K-means algorithm ($K = 2$) to split their one-dimensional embedding into two clusters also labeled as number 1 or 2, denoted as $\tilde{\gamma} \in \mathbb{R}^n$. Denote the ground truth of labels on the training set as $\gamma \in \mathbb{R}^n$. The classification accuracy is computed by $\max\{\frac{\sum_{i=1}^n |\gamma_i - \tilde{\gamma}_i|}{n}, 1 - \frac{\sum_{i=1}^n |\gamma_i - \tilde{\gamma}_i|}{n}\}$. The classification accuracy on the testing set can be computed in a similar way.

B.3. MNIST data

Data preprocessing: Our training set consists of 20000 sample images randomly selected from the MNIST training dataset and our testing set contains 10000 sample images from the MNIST testing dataset. Every sample in the training and testing set is vectorized as a vector in $\mathbb{R}^{784}$.

Network training: We use a fully-connected feedforward neural network with two 256-unit hidden layers:

SpecNet1: $784 \xrightarrow{\text{fc}} 256 - \text{ReLU} \xrightarrow{\text{fc}} 256 - \text{ReLU} \xrightarrow{\text{linear}} 7 \xrightarrow{\text{orthogonal}} 7$;

SpecNet2: $784 \xrightarrow{\text{fc}} 256 - \text{ReLU} \xrightarrow{\text{fc}} 256 - \text{ReLU} \xrightarrow{\text{linear}} 6$.

We want to embed the training set using first six nontrivial eigenvectors of $D^{-1}W$, so the output dimension for SpecNet1 is 7 and that for SpecNet2 is 6, and we use the Adam as the optimizer with learning rate 10^{-4} for both SpecNet1 and SpecNet2.

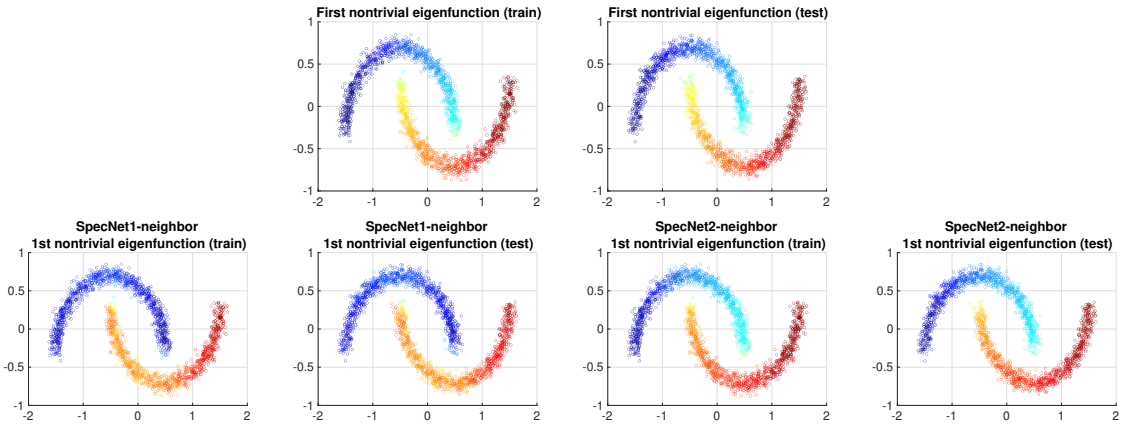

Figure 10: Two moons datset: embeddings by neighbor schemes using the first nontrivial eigenfunctions. The first row is the ground truth.

In Figure 6, since the minimum of f_1 and f_2 are not necessarily zero, we plot the quantities $\log_{10}(f_1(Y) - f_1^\star)$ and $\log_{10}(f_2(Y) - f_2^\star)$, where $f_1^\star = K - \sum_{i=1}^K \lambda_i$ and $f_2^\star = \sum_{i=2}^K \lambda_i^2$ are the global minimums (over matrix Y) of f_1 and f_2 respectively. (In the definition of $f_1^\star$ and $f_2^\star$, $\lambda_1 \geq \lambda_2 \cdots \geq \lambda_K$ are the K largest eigenvalues of $D^{-1}W$, D being the degree matrix of W.) The values of $(f_i(Y) - f_i^\star)$, $i = 1, 2$, in the plots are computed over 10 replicas of random initialization of the neural network. The solid curve shows the average over the replicas, and the shaded area around each curve reveals the standard deviation.

Figure 11 shows the embeddings (on both training and testing sets) at the 50-th training epoch, computed by SpecNet2-neighbor (with batch size 2) and SpecNet1-local (with batch size 45) respectively. By comparing to the true spectral embeddings (by linear algebra eigenvectors) plotted in the top panel, we can see that SpecNet2-neighbor gives a better result, and this is consistent with the lower value of losses of SpecNet2-neighbor in Figure 6. As shown in Figure 11, the embedding on test set is close to that on the training set, and this demonstrates the out-of-sample extension ability of SpecNet2.

We also show another example in Figure 12 where we construct the adjacency matrix A of an kNN graph on the training set by setting $A_{i,j} = 1$ if the j-th training sample is within k nearest neighbors of the i-th training sample and $A_{i,j} = 0$ otherwise, and we use $k = 10$. As a result, the average numbers of neighbors of a batch of size 2, 4 and 8 are about 28, 56 and 112 respectively. We observe that SpecNet2-neighbor with batch size 2 has higher variance, but it can still achieve better performance compared to SpecNet1-local with batch size 28 in the long run.

Appendix C. Scaling of the losses and gradients

We consider the generalized eigenvalue problem $WU = DU\Lambda$, where $W, D \in \mathbb{R}^{n\times n}$, $X \in \mathbb{R}^{n\times K}$, $W_{i,j} = k_\sigma(x_i, x_j) := e^{\frac{-\|x_i - x_j\|^2}{2\sigma^2}}$, and σ is fixed; D is a diagonal matrix such that $D_{i,i} = \sum_{j=1}^n W_{i,j}$.

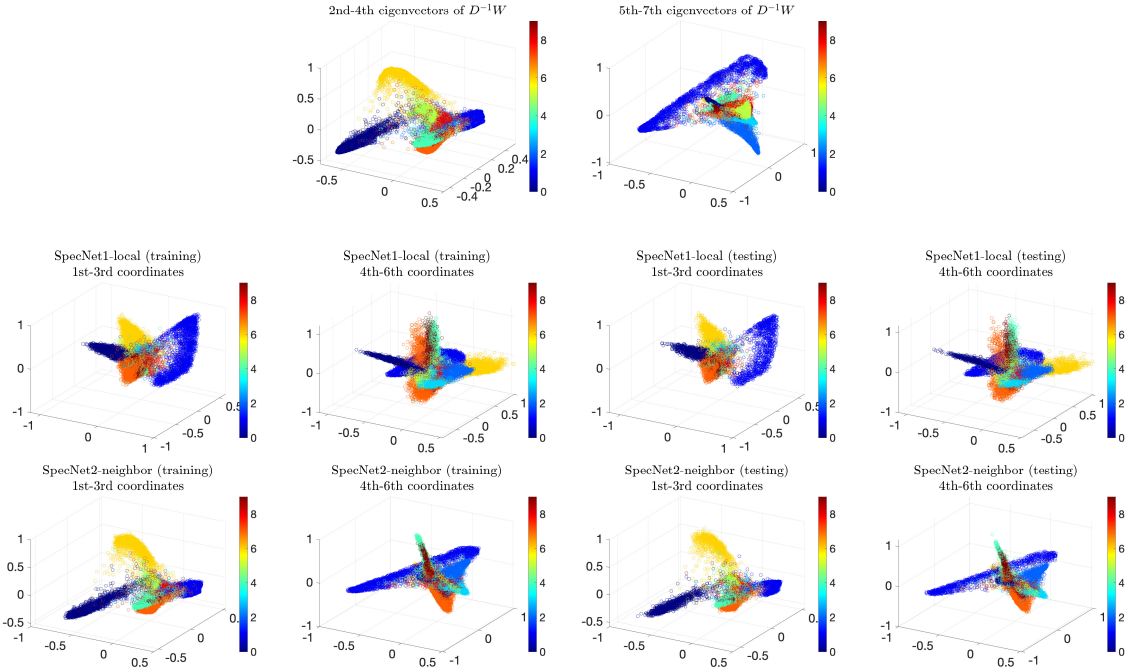

Figure 11: Embeddings of the MNIST datset. Top row: embeddings of the training set using the first six nontrivial eigenvectors of $D^{-1}W$; Middle row: embeddings computed by SpecNet1-local with batch size 45 at the 50-th epoch; Bottom row: embeddings computed by SpecNet2-neighbor with batch size 2 at the 50-th epoch.

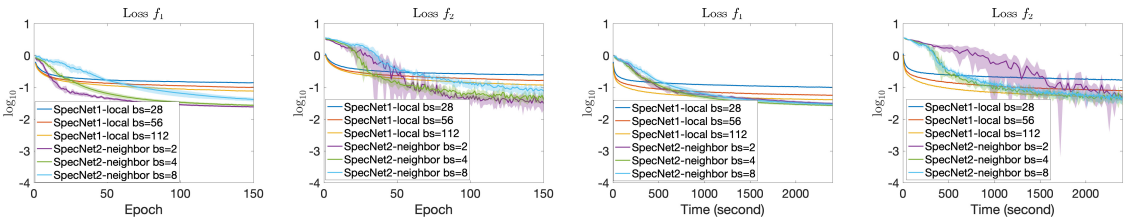

Figure 12: MNIST datset: plot of two different losses $\log_{10}(f_1(Y) - f_1^\star)$ and $\log_{10}(f_2(Y) - f_2^\star)$ over epochs (left two subfigures) and over time (right two subfigures), with f_1 and f_2 defined in (4) and (12). Networks are trained on 20000 MNIST images on a 2021 14-inch Macbook Pro with an 8-core CPU.

This generalized eigenvalue problem can be viewed as the discretization of the following continuous eigenvalue problem:

$$\int k_\sigma(x,y)\psi_k(y)p(y)\,\mathrm{d}y = \lambda_k u_\sigma(x)\psi_k(x), \tag{32}$$

where $p(x)$ is the density function, and $u_\sigma(x) := \int k_\sigma(x,y)p(y)\,\mathrm{d}y \approx m_0 p(x)\sigma^d + O_{d,p,k_\sigma}(\sigma^{d+2})$, where m_0 depends on the dimension d. And ψ_i satisfies the normalization condition:

$$\int \psi_i(x)\psi_j(x)u_\sigma(x)p(x)\,\mathrm{d}x = \delta_{ij}. \tag{33}$$

Note that $\lim_{n\to\infty} \frac{D_{i,i}}{n} = u_\sigma(x_i)$ by the law of large numbers.

Consider the loss function

$$g(Y) = \operatorname{tr}\left(-2Y^\top AY + Y^\top BYY^\top BY\right), \tag{34}$$

where $Y \approx [y_1(x), \ldots, y_k(x)] = [\sqrt{\lambda_1}\psi_1(x), \ldots, \sqrt{\lambda_k}\psi_k(x)]$ and entries of Y is $O(1)$. We need to know a proper scaling of A and B in terms of W and D so that g is $O(1)$ and does not scale with n. Recall that $\operatorname{tr}\left(Y^\top WY\right)$ and $\operatorname{tr}\left(Y^\top DYY^\top DY\right)$ are discretization of integrals,

$$\begin{aligned}
(Y^\top WY)_{k,k} &= \sum_{i=1}^n \sum_{j=1}^n y_k(x_i) y_k(x_j) k_\sigma(x_i, x_j) \\
&\approx n^2 \int\int k_\sigma(x,y) y_k(x) y_k(y) p(x) p(y)\,\mathrm{d}x\,\mathrm{d}y, \\
(Y^\top DY)_{k,l} &= \sum_{i=1}^n y_k(x_i) y_l(x_i) D_{i,i} \\
&\approx \sum_{i=1}^n y_k(x_i) y_l(x_i) n u_\sigma(x_i) \\
&\approx n^2 \int y_k(x) y_l(x) u_\sigma(x) p(x)\,\mathrm{d}x.
\end{aligned}$$

Hence, the proper scaling for A and B are $A = \frac{W}{n^2}$ and $B = \frac{D}{n^2}$ respectively.

Now let us look at the functional (variational) derivative of g with respect to y_s. We first split f into $g_1(Y) = \operatorname{tr}\left(-2Y^\top AY\right)$ and $g_2(Y) = \operatorname{tr}\left(Y^\top BYY^\top BY\right)$.

$$g_1(Y) \approx -2\sum_k \int\int k_\sigma(x,y) y_k(x) y_k(y) p(x) p(y)\,\mathrm{d}x\,\mathrm{d}y.$$

Replacing $y_s(x)$ by $y_s(x) + \epsilon\eta(x)$, and taking derivative with repsect to ϵ at 0, we have

$$\frac{\mathrm{d}g_1}{\mathrm{d}\epsilon}\Big|_{\epsilon=0} = -4\int\int k_\sigma(x,y) y_s(x) \eta(y) p(x) p(y)\,\mathrm{d}x\,\mathrm{d}y,$$

and the variational derivative of g_1 with respect to $y_s(x)$ is

$$\frac{\partial g_1}{\partial y_s}(y) = -4\int k_\sigma(x,y) y_s(x) p(x)\,\mathrm{d}x \approx \frac{1}{n}(-4\sum_{i=1}^n k_\sigma(x_i, y) y_s(x_i)).$$

Therefore, the $O(1)$ scaling of the gradient of g_1 is $\nabla_Y g_1 = -4\frac{W}{n}Y$.

On the other hand, a similar procedure can be applied to analyze the scaling of the gradient of g_2. Recall,

$$g_2(Y) = \sum_k (\sum_{i=1}^n (Y^\top BY)_{k,i} (Y^\top BY)_{i,k}) \approx \sum_k \sum_i (\int y_k(x) y_i(x) u_\sigma(x) p(x)\,\mathrm{d}x)^2.$$

Replacing $y_s(x)$ by $y_s(x) + \epsilon\eta(x)$, taking derivative with respect to ϵ at 0, and using the orthogonality condition (33), we have,

$$\frac{\mathrm{d}g_2}{\mathrm{d}\epsilon}\Big|_{\epsilon=0} = 4[(\int y_s^2(x)u_\sigma(x)p(x)\,\mathrm{d}x)(\int y_s(x)\eta(x)u_\sigma(x)p(x)\,\mathrm{d}x)],$$

and

$$\begin{aligned}\frac{\partial g_2}{\partial y_s}(x_j) &= 4(\int y_s^2(y)u_\sigma(y)p(y)\,\mathrm{d}y)y_s(x_j)u_\sigma(x_j)\\ &\approx 4\frac{1}{n}(\sum_{i=1}^n y_s^2(x_i)\frac{D_{i,i}}{n})y_s(x_j)\frac{D_{j,j}}{n} = \frac{4}{n^3}(\sum_{i=1}^n y_s^2(x_i)D_{i,i})y_s(x_j)D_{j,j}.\end{aligned}$$

Hence the $O(1)$ scaling of the gradient of g_2 is $\nabla_Y g_2 = \frac{4}{n^3}DYY^\top DY$.

Appendix D. Proofs

D.1. Proof of Theorem 4

We prove Theorem 4 in three steps. First we explicitly give the expressions for all stationary points of (9). Then we show that many of these stationary points are strict saddle points, i.e., there exists decay direction at these points. Finally, we prove the rest stationary points are of form as (21) and are global minimizers.

Recall the gradient of the objective function $f_2(Y)$ is of form (10). We can also derive the Hessian of the objective function and its bilinear form satisfies,

$$\begin{aligned}S^\top\nabla^2 f_2(Y)S =& -4\mathrm{tr}\left(S^\top\frac{W}{n}S\right) + 4\mathrm{tr}\left(S^\top\frac{D}{n}SY^\top\frac{D}{n}Y\right)\\ &+ 4\mathrm{tr}\left(S^\top\frac{D}{n}YS^\top\frac{D}{n}Y\right) + 4\mathrm{tr}\left(S^\top\frac{D}{n}YY^\top\frac{D}{n}S\right),\end{aligned}$$

where $S^\top\nabla^2 f_2(Y)S$ is a symbolic notation.

Stationary points of (9) satisfy the first order condition, i.e.,

$$\nabla f_2(Y) = 0 \Leftrightarrow WY = DYY^\top\frac{D}{n}Y \Leftrightarrow \left(D^{-\frac{1}{2}}WD^{-\frac{1}{2}}\right)\left(D^{\frac{1}{2}}Y\right) = \left(D^{\frac{1}{2}}Y\right)Y^\top\frac{D}{n}Y. \tag{35}$$

The right most equality in (35) implies that $D^{\frac{1}{2}}Y$ lies in an invariant subspace of $D^{-\frac{1}{2}}WD^{-\frac{1}{2}}$, which is formed by eigenvectors of $D^{-\frac{1}{2}}WD^{-\frac{1}{2}}$. Denote the invariant subspace by eigenvectors $V_r \in \mathbb{R}^{n\times r}$, where $r \le K$ is the dimension. The corresponding eigenvalues are denoted by a diagonal matrix $\Lambda_r \in \mathbb{R}^{r\times r}$. In connection to (8), Λ_r consists r eigenvalues of (W, D) and V_r consists of r eigenvectors of (W, D) transformed by $D^{\frac{1}{2}}$. $D^{\frac{1}{2}}Y$ then can be written as $D^{\frac{1}{2}}Y = V_rA$ for $A \in \mathbb{R}^{r\times K}$ being a full row rank matrix. Substituting the expression back into (35), we obtain,

$$V_r\Lambda_rA = V_rAA^\top A \Leftrightarrow \Lambda_rA = AA^\top A \Leftrightarrow \Lambda_r = AA^\top, \tag{36}$$

where the equivalences are due to the orthogonality of V_r and full-rankness of A. Therefore, A admit the follow expression,

$$A = \Lambda_r^{\frac{1}{2}}Q, \tag{37}$$

where $Q \in \mathbb{R}^{r\times K}$ is a unitary matrix such that $QQ^\top = I$. Putting the above analysis together, we conclude that the stationary points of (9) are of form,

$$Y = U\Lambda^{\frac{1}{2}}PQ, \tag{38}$$

where Λ and U are the eigenvalue and the corresponding eigenvector matrix of (W, D), $P \in \mathbb{R}^{n\times r}$ is the first r columns of an arbitrary permutation matrix for $r \leq K$, and $Q \in \mathbb{R}^{r\times K}$ is an arbitrary orthogonal matrix.

Next we will show that many of these stationary points are saddle points. Consider a stationary point Y_0 which does not include one of the leading K eigenvectors, e.g., $Y_0^\top DU_i = 0$ and i is an index smaller than K. If $r < K$, then we have a unitary vector $Q_\perp \in \mathbb{R}^{1\times K}$ such that $Q_\perp Q^\top = 0$. Selecting a direction $S_0 = U_i Q_\perp$, the Hessian at Y_0 evaluated at S_0 is,

$$S_0^\top \nabla^2 f_2(Y_0)S_0 = -4\mathrm{tr}\left(S_0^\top \frac{W}{n} S_0\right) + 4\mathrm{tr}\left(S_0^\top \frac{D}{n} S_0 Y_0^\top \frac{D}{n} Y_0\right) = -4\lambda_i < 0. \tag{39}$$

If $r = K$, then there are K eigenvectors selected by P and one of them must have index greater than K. Without loss of generality, we assume the first column of $U\Lambda^{\frac{1}{2}}P$ is eigenvector with index $K+1$. Then we choose a specific $S_0 = \begin{bmatrix} U_i & 0 & \cdots & 0 \end{bmatrix} Q \in \mathbb{R}^{n\times K}$ and obtain,

$$S_0^\top \nabla^2 f_2(Y_0)S_0 = -4\mathrm{tr}\left(S_0^\top \frac{W}{n} S_0\right) + 4\mathrm{tr}\left(S_0^\top \frac{D}{n} S_0 Y_0^\top \frac{D}{n} Y_0\right) = -4\lambda_i + 4\lambda_{K+1} < 0, \tag{40}$$

where the last inequality is due to the assumption on the nonzero eigengap between the K-th and $(K+1)$-th eigenvalues. Therefore, we conclude that when any of the leading K eigenvectors is not selected in (38), the stationary point is a strict saddle point. Besides these strict saddle points, the rest stationary points are of form,

$$Y = U_K \Lambda_K^{\frac{1}{2}} Q, \tag{41}$$

where Λ_K consists K leading eigevalues and U_K consists the corresponding K eigenvectors, $Q \in \mathbb{R}^{K\times K}$ is an arbitrary orthogonal matrix.

D.2. Proof of Corollary 5

Proof $f_2(Y)$ is a smooth function of Y and note that the second term inside the trace of f_2 is $Y^\top DYY^\top DY$, which is a fourth-order term of Y, and D is positive-definite, so $f_2(Y) \to +\infty$ as $\|Y\| \to +\infty$ and $f_2(Y)$ is bounded from below. Hence global minimizers of $f_2(Y)$ exist and are among local minimizers. Substituting all local minimizers as shown in Theorem 4 into $f_2(Y)$, we have,

$$f_2(Y^\star) = -\sum_{i=1}^{K} \lambda_i^2, \tag{42}$$

which means all local minimizers are of the same objective function value. They are all global minimizers. ■

D.3. Proof of Theorem 9

Proof [Proof of Lemma 6]

Let $Y^+ = g_p(Y)$ for any $p = 1, \dots, b$. By the definition of W_0, it suffices to show $\left\| D_i^{\frac{1}{2}} Y_i^+ \right\|_2 < R$ for $i \in S_p$ to prove the lemma.

First, recall the iterative expression for i-th coordinate,

$$\begin{aligned} Y_i^+ &= Y_i + 4\alpha(W_{i,:}Y - D_i Y_i (Y^\top D Y)) \\ &= Y_i + 4\alpha(W_{i,i}Y_i + W_{i,i^c}Y_{i^c} - D_i Y_i (Y_i^\top D_i Y_i) - D_i Y_i (Y_{i^c}^\top D_{i^c} Y_{i^c})). \end{aligned}$$

Left multiplying $D_i^{\frac{1}{2}}$ for rescaling purpose, we obtain,

$$\begin{aligned} D_i^{\frac{1}{2}} Y_i^+ =& D_i^{\frac{1}{2}} Y_i + 4\alpha \Big(W_{i,i} D_i^{\frac{1}{2}} Y_i + D_i^{\frac{1}{2}} W_{i,i^c} D_{i^c}^{-\frac{1}{2}} (D_{i^c}^{\frac{1}{2}} Y_{i^c}) \\ & - D_i (D_i^{\frac{1}{2}} Y_i)(Y_i^\top D_i Y_i) - D_i (D_i^{\frac{1}{2}} Y_i)(Y_{i^c}^\top D_{i^c} Y_{i^c}) \Big) \\ =:& D_i^{\frac{1}{2}} Y_i + 4\alpha T_i, \end{aligned}$$

where T_i denotes all terms in the parentheses. Denote $X := DY$, $X_i := D_i^{\frac{1}{2}} Y_i$, $X_{i^c} := D_{i^c}^{\frac{1}{2}} Y_{i^c}$ and $X_i^+ := D_i^{\frac{1}{2}} Y_i^+$. Then we have

$$\begin{aligned} & \left\| X_i^+ \right\|_2^2 \\ =& \left\| X_i \right\|_2^2 + 16\alpha^2 \left\| T_i \right\|_2^2 + 8\alpha \left(W_{i,i} \left\| X_i \right\|_2^2 - D_i \left\| X_i \right\|_2^4 + D_i^{\frac{1}{2}} W_{i,i^c} D_{i^c}^{-\frac{1}{2}} X_{i^c} X_i^\top - D_i \left\| X_i X_{i^c}^\top \right\|_2^2 \right) \\ \le& \left\| X_i \right\|_2^2 + 16\alpha^2 \left\| T_i \right\|_2^2 \\ & + 8\alpha \left(W_{i,i} \left\| X_i \right\|_2^2 - D_i \left\| X_i \right\|_2^4 + D_i^{\frac{1}{2}} \left\| W_{i,i^c} D_{i^c}^{-\frac{1}{2}} \right\| \left\| X_{i^c} X_i^\top \right\| - D_i \left\| X_i X_{i^c}^\top \right\|_2^2 \right). \end{aligned}$$

First, we bound $\left\| T_i \right\|_2^2$ as,

$$\begin{aligned} \left\| T_i \right\|_2^2 \le& 3 \left(\max_i W_{i,i}^2 R^2 + \left\| D_i X_i \right\|_2^2 \left\| X^\top X \right\|_2^2 + \left\| D_i^{\frac{1}{2}} W_{i,i^c} D_{i^c}^{-\frac{1}{2}} \right\|_2^2 \left\| X_{i^c} \right\|_2^2 \right) \\ \le& 3 \left(\max_i W_{i,i}^2 R^2 + \max_i D_i^2 \cdot n^2 K^2 R^6 + \max_i D_i \left\| W_{i,i^c} D_{i^c}^{-\frac{1}{2}} \right\|_2^2 \cdot n R^2 \right) = M(R), \end{aligned}$$

where we adopts $\max_i \left\| X_i \right\|_2 < R$, $\left\| X^\top X \right\|_2^2 \le \left\| X^\top X \right\|_F^2 \le K^2 (nR^2)^2$, and $\left\| X_{i^c} \right\|_2^2 < nR^2$.

Then, we estimate the coefficient of linear term in α. By the argument of second order polynomial, we have,

$$\begin{aligned} W_{i,i} \left\| X_i \right\|_2^2 - D_i \left\| X_i \right\|_2^4 \le& \frac{W_{i,i}^2}{4D_i}, \\ D_i^{\frac{1}{2}} \left\| W_{i,i^c} D_{i^c}^{-\frac{1}{2}} \right\|_2 \left\| X_i X_{i^c}^\top \right\|_2 - D_i \left\| X_i X_{i^c}^\top \right\|_2^2 \le& \frac{\left\| W_{i,i^c} D_{i^c}^{-\frac{1}{2}} \right\|_2^2}{4}. \end{aligned}$$

Next we discuss the inequality of $\left\|X_i^+\right\|_2^2$ in two cases: $\|X_i\|_2 \leq \frac{R}{2}$ and $\frac{R}{2} < \|X_i\|_2 < R$. When $\|X_i\|_2 \leq \frac{R}{2}$, we have

$$\left\|X_i^+\right\|_2^2 \leq \frac{R^2}{4} + 16\alpha^2 M(R) + 8\alpha M_2 < R^2,$$

where the last inequality can be verified using $\alpha < \frac{-2M_2+\sqrt{4M_2^2+3M(R)R^2}}{8M(R)}$.

When $\frac{R}{2} < \|X_i\|_2 < R$, again by the argument of second order polynomial, we have

$$W_{i,i}\|X_i\|_2^2 - D_i\|X_i\|_2^4 + \frac{\left\|W_{i,i^c}D_{i^c}^{-\frac{1}{2}}\right\|_2^2}{4} < -\frac{1}{8}$$

due to the fact that $R \geq 2\sqrt{M_1}$. Substituting into the inequality of $\left\|X_i^+\right\|_2^2$, we have

$$\left\|X_i^+\right\|_2^2 \leq \|X_i\|_2^2 + 16\alpha^2 M(R) - \alpha < \|X_i\|_2^2 < R^2,$$

where the second inequality can be verified using $\alpha < \frac{1}{16M(R)}$. ■

Proof [Proof of Lemma 7]

First, through a direct calculation, we have

$$\frac{\partial f_2}{\partial Y_{i_1,k_1}} = -4\sum_{j=1}^n W_{i_1,j}Y_{j,k_1} + 4D_{i_1}\sum_{k=1}^K Y_{i_1,k}\left(\sum_{\ell=1}^n Y_{\ell,k}D_\ell Y_{\ell,k_1}\right).$$

And the second order partial derivative admits,

$$\begin{aligned}\frac{\partial^2 f_2}{\partial Y_{i_1,k_1}\partial Y_{i_2,k_2}} =& -4\delta_{k_1k_2}W_{i_1,i_2} + 4D_{i_1}\delta_{i_1i_2}\left(\sum_{\ell=1}^n Y_{\ell,k_2}D_\ell Y_{\ell,k_1}\right)\\ &+ 4D_{i_1}Y_{i_1,k_2}D_{i_2}Y_{i_2,k_1} + 4D_{i_1}\sum_{k=1}^K Y_{i_1,k}Y_{i_2,k}D_{i_2}\delta_{k_1k_2},\end{aligned}$$

where $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ otherwise. By assumption that $\max_i \left\|D_i^{\frac{1}{2}}Y_i\right\|_2 < R$, we have $\max_{i,j}\left|D_i^{\frac{1}{2}}Y_{i,j}\right| < R$, and $\max_j \left\|D^{\frac{1}{2}}Y_{:,j}\right\|_2^2 < nR^2$. Therefore,

$$\begin{aligned}\left|\frac{\partial^2 f_2}{\partial Y_{i_1,k_1}\partial Y_{i_2,k_2}}\right| \leq& 4\max_{i,j}W_{i,j} + 4D_{i_1}nR^2 + 4\max_i D_iKR^2\\ \leq& 4\max_{i,j}W_{i,j} + 4\max_i D_i(n+K)R^2.\end{aligned}$$

■

Proof [Proof of Lemma 8] Applying the updating expression, we have

$$f_2(Y^{(\ell+1)}) \le f_2(Y^{(\ell)}) - \alpha \sum_{i\in S_\ell}\sum_{j=1}^{K}\left(\nabla_{i,j} f_2(Y^{(\ell)})\right)^2 + \alpha^2 L \sum_{i\in S_\ell}\sum_{j=1}^{K}\left(\nabla_{i,j} f_2(Y^{(\ell)})\right)^2,$$

where we abuse noation S_ℓ to denote the batch at ℓ-th iteration. Since $1 - \frac{\alpha L}{2} > 0$, we have

$$\sum_{i\in S_\ell}\sum_{j=1}^{K}\left(\nabla_{i,j} f_2(Y^{(\ell)})\right)^2 \le \frac{1}{\alpha(1-\alpha L)}\left(f_2(Y^{(\ell)}) - f_2(Y^{(\ell+1)})\right).$$

Summing over all ℓ from 0 to $T-1$, for $T = bP$ and any large integer P, we have

$$\begin{aligned}\sum_{p=0}^{P-1}\sum_{\ell=bp}^{b(p+1)-1}\left[\sum_{i\in S_\ell}\sum_{j=1}^{K}\left(\nabla_{i,j} f_2(Y^{(\ell)})\right)^2\right] &\le \frac{1}{\alpha(1-\alpha L)}\left(f_2(Y^{(0)}) - f_2(Y^{(T)})\right)\\ &\le \frac{1}{\alpha(1-\alpha L)}\left(f_2(Y^{(0)}) - f_2^*\right),\end{aligned}$$

where f_2^* denotes the minimum of f_2. Hence

$$\lim_{\ell\to\infty}\sum_{i\in S_\ell}\sum_{j=1}^{K}\left(\nabla_{i,j} f_2(Y^{(\ell)})\right)^2 = 0.$$

That is, for any $\epsilon > 0$, there exists an integer $P_0 > 0$, such that for any $p \ge P_0$, we have

$$\sum_{i\in S_\ell}\sum_{j=1}^{K}\left(\nabla_{i,j} f_2(Y^{(\ell)})\right)^2 \le \epsilon^2, \quad \text{for } \ell = pb, \ldots, (p+1)b-1.$$

For any two iterations, ℓ_1 and ℓ_2 such that $pb \le \ell_1 \le \ell_2 < (p+1)b$, and for any $i \in S_{\ell_1}$, $1 \le j \le K$, we have

$$\begin{aligned}\left|\nabla_{i,j} f_2(Y^{(\ell_1)}) - \nabla_{i,j} f_2(Y^{(\ell_2)})\right| &\le \sum_{\ell=\ell_1}^{\ell_2-1}\left|\nabla_{i,j} f_2(Y^{(\ell)}) - \nabla_{i,j} f_2(Y^{(\ell+1)})\right|\\ &\le L\sum_{\ell=\ell_1}^{\ell_2-1}\left\|Y^{(\ell)} - Y^{(\ell+1)}\right\|_2\\ &\le L\sum_{\ell=\ell_1}^{\ell_2-1}\alpha\sqrt{\sum_{i\in S_\ell}\sum_{j=1}^{K}\left(\nabla_{i,j} f_2(Y^{(\ell)})\right)^2}\\ &< b\epsilon,\end{aligned}$$

where the last inequality is due to $\alpha L < 1$.

Let ℓ_0 be an iteration within pb and $(p+1)b-1$, $p \geq P_0$. Note that $\cup_{\ell=pb}^{(p+1)b-1} S_\ell = [n]$. Then we have

$$\begin{aligned}
\left\|\nabla f_2(Y^{(\ell_0)})\right\|_2^2 =& \sum_{\ell=pb}^{(p+1)b-1} \sum_{i\in S_\ell} \sum_{j=1}^{K} \left(\nabla_{i,j} f_2(Y^{(\ell_0)})\right)^2 \\
=& \sum_{\ell=pb}^{(p+1)b-1} \sum_{i\in S_\ell} \sum_{j=1}^{K} \left(\nabla_{i,j} f_2(Y^{(\ell_0)}) - \nabla_{i,j} f_2(Y^{(\ell)}) + \nabla_{i,j} f_2(Y^{(\ell)})\right)^2 \\
\leq& \sum_{\ell=pb}^{(p+1)b-1} \sum_{i\in S_\ell} \sum_{j=1}^{K} \Big[\left(\nabla_{i,j} f_2(Y^{(\ell)})\right)^2 + 2\epsilon \left|\nabla_{i,j} f_2(Y^{(\ell_0)}) - \nabla_{i,j} f_2(Y^{(\ell)})\right| \\
&+ \left|\nabla_{i,j} f_2(Y^{(\ell_0)}) - \nabla_{i,j} f_2(Y^{(\ell)})\right|^2 \Big] \\
<& (b + 2nKb + nKb^2)\epsilon^2.
\end{aligned}$$

Since ϵ can be arbitrarily small, we proved the lemma.

■

Proof [Proof of Theorem 9]

Lemma 6 states that for any $Y \in W_0$ and $1 \leq i \leq b$, we have $g_i(Y) \in W_0$. Hence we have for any $Y \in W_0$, $g(Y) \in W_0$. Lemma 7 states that f_2 has bounded Lipschitz coordinate gradient in W_0, and the stepsize α satisfies $\alpha < \frac{1}{KL \max_{i\in[b]} |S_i|}$. Note that $\max_{i\in[b]} \left\|\nabla^2 f_2(Y)_{S_i}\right\|_2 \leq \max_{i\in[b]} \left\|\nabla^2 f_2(Y)_{S_i}\right\|_F \leq \sqrt{(K \cdot \max_{i\in[b]} |S_i|)^2 L^2} = KL \max_{i\in[b]} |S_i|$, Proposition 6 in (Lee et al., 2019) shows that under these conditions, we have $\det(Dg(x)) \neq 0$. Corollary 5 in (Lee et al., 2019) tells us that $\mu(\{Y^{(0)} : \lim_{j\to\infty} g^j(Y^{(0)}) \in \chi^s\}) = 0$ for χ^s being the set of unstable stationary points and local maximizers. Combining with the conclusion of Lemma 8, we obtain the conclusion of Theorem 9.

■