Fisher-Bures Adversary Graph Convolutional Networks

Ke Sun[†][*] **Piotr Koniusz**[†][‡] **Zhen Wang**[†]
[†]CSIRO Data61 [‡]Australia National University
{Ke.Sun, Peter.Koniusz, Jeff.Wang}@data61.csiro.au

Abstract

In a graph convolutional network, we assume that the graph G is generated w.r.t. some observation noise. During learning, we make small random perturbations ΔG of the graph and try to improve generalization. Based on quantum information geometry, ΔG can be characterized by the eigendecomposition of the graph Laplacian matrix. We try to minimize the loss w.r.t. the perturbed $G + \Delta G$ while making ΔG to be effective in terms of the Fisher information of the neural network. Our proposed model can consistently improve graph convolutional networks on semi-supervised node classification tasks with reasonable computational overhead. We present three different geometries on the manifold of graphs: the intrinsic geometry measures the information theoretic dynamics of a graph; the extrinsic geometry characterizes how such dynamics can affect externally a graph neural network; the embedding geometry is for measuring node embeddings. These new analytical tools are useful in developing a good understanding of graph neural networks and fostering new techniques.

1 INTRODUCTION

Recently, neural network architectures are introduced [15, 36, 7, 12, 21, 17, 42] to learn high level features of objects based on a given graph among these objects. These graph neural networks, especially graph convolutional networks (GCNs), showed record-breaking scores on diverse learning tasks. Similar to the idea of data augmentation, this paper improves GCN generalization by minimizing the expected loss w.r.t. small random perturbations of the input graph. In order to do so, we must first have a rigorous definition of the *manifold of graphs* denoted by $\mathcal{M}$, which is the space of all graphs satisfying certain constraints. Then, based on the local geometry of $\mathcal{M}$ around a graph $G \in \mathcal{M}$, we can derive a compact parameterization of the perturbation so that it can be plugged into a GCN. We will show empirically that the performance of GCN can be improved and present theoretical insights on the differential geometry of $\mathcal{M}$.

[*]Corresponding author

Notations

We assume an undirected graph G without self-loops consisting of n nodes indexed as $1, \cdots, n$. $X_{n\times D}$ denotes the given node features, $H_{n\times d}$ denotes some learned high-level features, and $Y_{n\times O}$ denotes the one-hot node labels. All these matrices contain one sample per row. The graph structure is represented by the adjacency matrix $A_{n\times n}$ that can be binary or weighted, so that $a_{ij} \geq 0$, and $a_{ij} = 0$ indicates no link between nodes i and j. The neural networks weights are denoted by the matrix W^l, where $l = 1, \cdots, L$ indexes the layers. We use capital letters such as A, B, $\cdots$ to denote matrices and small letters such as a, b, $\cdots$ to denote vectors. We try to use Greek letters such as α, β, $\cdots$ to denote scalars. These rules have exceptions.

Problem Formulation

In a vanilla GCN model (see the approximations [21] based on [12]), the network architecture is recursively defined by

$$H^{l+1} = \sigma\left(\tilde{A} H^l W^l\right), \quad H^0 = X,$$

where $H^l_{n\times d^l}$ is the feature matrix of the l'th layer with its rows corresponding to the samples, $W^l_{d^l\times d^{l+1}}$ is the sample-wise feature transformation matrix, $\tilde{A}_{n\times n}$ is the normalized adjacency matrix so that $\tilde{A} = (D+I)^{-\frac{1}{2}}(A+$

$I)(D+I)^{-\frac{1}{2}}$, I is the identity matrix, $D = \mathrm{diag}(A1)$ is the degree matrix, 1 is the vector of all ones, $\mathrm{diag}()$ means a diagonal matrix w.r.t. the given diagonal entries, and σ is an element-wise nonlinear activation function. Based on a given set of samples X and optionally the corresponding labels Y, learning is implemented by $\min_W \ell(X, Y, A, W)$, where ℓ is a loss (*e.g.* cross-entropy), usually expressed in terms of Y and H^L, the feature matrix obtained by stacking multiple GCN layers.

Our basic assumption is that A is observed w.r.t. an underlying generative model as well as some random observation noise. In order to make learning robust to these noise and generalize well, we minimize the expected loss

$$\min_W \int q(\phi \mid \varphi)\ell(X, Y, A(\phi), W)\, d\phi, \qquad (1)$$

where $A(\phi)$ is a parameterization of graph adjacency matrices, $A(0) = A$ is the original adjacency matrix, $q(\phi \mid \varphi)$ is a zero-centered random perturbation so that $A(\phi)$ in a "neighborhood" of A, and φ is the freedom of this perturbation.

To implement this machinery, we must answer a set of fundamental questions: ① How to define the manifold $\mathcal{M}$ of graphs, *i.e.* the space of A? ② How to properly define the neighborhood $\{A(\phi) : \phi \sim q(\phi \mid \varphi)\}$? ③ What is the guiding principles to learn the neighborhood parameters φ? We will build a geometric solution to these problems, provide an efficient implementation of eq. (1), and test the empirical improvement in generalization. Our contributions are both theoretical and practical, which are summarized as follows:

- We bridge quantum information theory with graph neural networks, and provide Riemannian metrics in closed form on the manifold of graphs;
- We build a modified GCN [21] called the FisherGCN that can consistently improve generalization;
- We introduce algorithm 1 to pre-process the graph adjacency matrix for GCN so as to incorporate high order proximities.

The rest of this paper is organized as follows. We first review related works in section 2. Section 3 introduces basic quantum information geometry, based on which the following section 4 formulates the manifold of graphs $\mathcal{M}$ and graph neighborhoods. Sections 5 and 6 present the technical details and experimental results of our proposed FisherGCN. Sections 7 and 8 provide our theoretical analysis on different ways to define the geometry of $\mathcal{M}$. Section 9 concludes and discusses future extensions.

2 RELATED WORKS

Below, we related our work to deep learning on graphs (with a focus on sampling strategies), adversary learning, and quantum information geometry.

Graph Neural Networks

The graph convolutional network [21] is a state-of-the-art graph neural network [15, 36] which performs convolution on the graphs in the spectral domain. While the performance of GCNs is very attractive, spectral convolution is a costly operation. Thus, the most recent implementations, *e.g.* GraphSAGE [17], takes convolution from spectral to spatial domain defined by the local neighborhood of each node. The average pooling on the nearest neighbors of each node is performed to capture the contents of the neighborhood. Below we describe related works which, one way or another, focus on various sampling strategies to improve aggregation and performance.

Structural Similarity and Sampling Strategies

Graph embeddings [33, 16] capture structural similarities in the graph. DeepWalk [33] takes advantage of simulated localized walks in the node proximity which are then forwarded to the language modeling neural network to form the node context. Node2Vec [16] interpolates between breadth- and depth-first sampling strategies to aggregate different types of neighborhood.

MoNet [26] generalizes the notion of coordinate spaces by learning a set of parameters of Gaussian functions to encode some distance for the node embedding, *e.g.* the difference between degrees of a pair of nodes. Graph attention networks [42] learn such weights via a self-attention mechanism. Jumping Knowledge Networks (JK-Nets) [46] also target the notion of node locality. Experiments on JK-Nets show that depending on the graph topology, the notion of the subgraph neighborhood varies, *e.g.* random walks progress at different rates in different graphs. Thus, JK-Nets aggregate over various neighborhoods and considers multiple node localities. By contrast, we apply mild adversary perturbations of the graph Laplacian based on quantum Fisher information so as to improve generalization. Thus, we infer a "correction" of the Laplacian matrix while JK-Net aggregates multiple node localities.

Sampling strategy has also an impact on the total size of receptive fields. In the vanilla GCN [21], the receptive field of a single node grows exponentially w.r.t. the number of layers which is computationally costly and results in so-called over smoothing of signals [22]. Thus, stochastic GCN [10] controls the variance of the activation estimator by keeping the history/summary of activations in the previous layer to be reused.

Both our work and Deep Graph Infomax (DGI) [43] take an information theoretic approach. DGI maximizes the mutual information between representations of local subgraphs (a.k.a. patches) and high-level summaries of graphs while minimizing the mutual information between negative samples and the summaries. This "contrasting" strategy is somewhat related to our approach as we generate adversarial perturbation of the graph to flatten the most abrupt curvature directions. DGI relies on the notion of positive and negative samples. In contrast, we learn maximally perturbed parameters of our extrinsic graph representation which are the analogy to negative samples.

Lastly, noteworthy are application driven pipelines, *e.g.* for molecule classification [13].

Adversarial Learning

The role of adversarial learning is to generate difficult-to-classify data samples by identifying them along the decision boundary and "pushing" them over this boundary. In a recent DeepFool approach [27], a cumulative sparse adversarial pattern is learned to maximally confuse predictions on the training dataset. Such an adversarial pattern generalizes well to confuse prediction on test data. Adversarial learning is directly connected to sampling strategies, *e.g.* sampling hard negatives (obtaining the most difficult samples), and it has been long investigated [14, 37] in the community, especially in the shallow setting [5, 19, 45].

Adversarial attacks under the Fisher information metric (FIM) [50] propose to carry out perturbations in the spectral domain. Given a quadratic form of the FIM, the optimal adversarial perturbation is given by the first eigenvector corresponding to the largest eigenvalue. The larger the eigenvalues of the FIM are, the larger is the susceptibility of the classification approach to attacks on the corresponding eigenvectors.

Our work is related in that we also construct a quantum version of the FIM w.r.t. a parameterization of the graph Laplacian. We perform a maximization w.r.t. these parameters to condition the FIM around the local optimum, thus making our approach well regularized in the sense of flattening the most curved directions associated with the FIM. With the smoothness constraint, the classification performance typically degrades, *e.g.* see the impact of smoothness on kernel representations [24]. Indeed, study [41] further shows there is a fundamental trade-off between high accuracy and the adversarial robustness.

However, our min-max formulation seeks the most effective perturbations (according to [50]) which thus simultaneously prevents unnecessary degradation of the decision boundary. With robust regularization for medium size datasets, we avoid overfitting which boosts our classification performance, as demonstrated in the following section 6.

Quantum Information Geometry

Natural gradient [2, 3, 32, 49, 40] is a second-order optimization procedure which takes the steepest descent w.r.t. the Riemannian geometry defined by the FIM, which takes small steps on the directions with a large scale of FIM. This is also suggestive that the largest eigenvectors of the FIM are the most susceptible to attacks.

Bethe Hessian [35], or deformed Laplacian, was shown to improve the performance of spectral clustering on a par with non-symmetric and higher dimensional operators, yet, drawing advantages of symmetric positive-definite representation. Our graph Laplacian parameterization also draws on this view.

Tools from quantum information geometry are applied to machine learning [4, 29] but not yet ported to the domain of graph neural networks. In information geometry, one can have different matrix divergences [30] that can be applied on the cone of p.s.d. matrices. We point the reader to related definitions of the discrete Fisher information [11] without illuminating the details.

3 PREREQUISITES

Fisher Information Metric

The discipline of information geometry [2] studies the space of probability distributions based on the Riemannian geometry framework. As the most fundamental concept, the Fisher information matrix is defined w.r.t. a given statistical model, *i.e.* a parametric form of the conditional probability distribution $p(X \mid \Theta)$, by

$$\mathcal{G}(\Theta) = \int p(X \mid \Theta) \frac{\log p(X \mid \Theta)}{\partial \Theta} \frac{\log p(X \mid \Theta)}{\partial \Theta^\top} \mathrm{d}X. \tag{2}$$

By definition, we must have $\mathcal{G}(\Theta) \succeq 0$. Following H. Hotelling and C. R. Rao, this $\mathcal{G}(\Theta)$ is used (see section 3.5 [2] for history) to define the Riemannian metric of a statistical model $\mathcal{M} = \{\Theta : p(X \mid \Theta)\}$, which is known as the Fisher information metric $ds^2 = d\Theta^\top \mathcal{G}(\Theta) d\Theta$. Intuitively, the scale of ds^2 corresponds to the *intrinsic* change of the model w.r.t. the movement $d\Theta$. The FIM is invariant to reparameterization and is the unique Riemannian metric in the space of probability distributions under certain conditions [9, 2].

Bures Metric

In quantum mechanics, a quantum state is represented by a graph (see *e.g.* [6]). Denote a parametric graph Laplacian matrix as $L(\Theta)$, and the trace-normalized Laplacian $\rho(\Theta) = \frac{1}{\mathrm{tr}(L(\Theta))}L(\Theta)$ is known as the *density matrix*, where $\mathrm{tr}(\cdot)$ means the trace. One can therefore generalize the FIM to define a geometry of the Θ space. In analogy to eq. (2), the quantum version of the Fisher information matrix is

$$\mathcal{G}_{ij}(\Theta) = \frac{1}{2}\mathrm{tr}\left[\rho(\Theta)(\partial L_i \partial L_j + \partial L_j \partial L_i)\right], \quad (3)$$

where ∂L_i is the symmetric logarithmic derivative that generalizes the notation of the derivative of logarithm:

$$\frac{\partial \rho}{\partial \theta_i} = \frac{1}{2}(\rho \cdot \partial L_i + \partial L_i \cdot \rho).$$

Let $\rho(\Theta)$ be diagonal, then $\partial L_i = \partial \log \rho / \partial \theta_i$. Plugging into eq. (3) will recover the traditional Fisher information defined in eq. (2). The quantum Fisher information metric $ds^2 = d\Theta^\top \mathcal{G}(\Theta) d\Theta$, up to constant scaling, is known as the Bures metric [8]. We use BM to denote these equivalent metrics and abuse $\mathcal{G}$ to denote both the BM and the FIM. We develop upon the BM without considering its meanings in quantum mechanics. This is because ① it can fall back to classical Fisher information; ② its formulations are well-developed and can be useful to develop deep learning on graphs.

4 AN INTRINSIC GEOMETRY

In this section, we define an intrinsic geometry of graphs based on the BM, so that one can measure distances on the manifold of all graphs with a given number of nodes and have the notion of *neighborhood*.

We parameterize a graph by its density matrix

$$\rho = U\,\mathrm{diag}(\lambda)U^\top = \sum_{i=1}^n \lambda_i u_i u_i^\top \succeq 0, \quad (4)$$

where $UU^T = U^T U = I$ so that U is on the unitary group, *i.e.* the manifold of unitary matrices, u_i is the i'th column of U, λ satisfies $\lambda \geq 0$, $\lambda^\top 1 = 1$ and is on the closed probability simplex. Notice that the $\tau \geq 1$ smallest eigenvalue(s) of the graph Laplacian (and ρ which shares the same spectrum up to scaling) are zero, where τ is the number of connected components of the graph.

Fortunately for us, the BM w.r.t. this canonical parameterization was already derived in closed form (see eq.(10) [18]), given by

$$ds^2 = \frac{1}{2}\sum_{j=1}^n \sum_{k=1}^n \frac{(u_j^\top d\rho u_k)^2}{\lambda_j + \lambda_k}. \quad (5)$$

For simplicity, we are mostly interested in the diagonal blocks of the FIM. Plugging

$$d\rho = \sum_{i=1}^n \left[d\lambda_i u_i u_i^\top + \lambda_i du_i u_i^\top + \lambda_i u_i du_i^\top\right]$$

into eq. (5), we get the following theorem.

Theorem 1. *In the canonical parameterization $\rho = U\,\mathrm{diag}(\lambda)U^\top$, the BM is*

$$\begin{aligned} ds^2 &= d\lambda^\top \mathcal{G}(\lambda) d\lambda + \sum_{i=1}^n du_i^\top \mathcal{G}(u_i) du_i \\ &= \sum_{i=1}^n \Bigg[\frac{1}{4\lambda_i} d\lambda_i^2 + d\lambda_i c_i^\top du_i \\ &\quad + \frac{1}{2} du_i^\top \sum_{j=1}^n \left(\frac{(\lambda_i - \lambda_j)^2}{\lambda_i + \lambda_j} u_j u_j^\top\right) du_i\Bigg], \end{aligned}$$

where c_i are some coefficients which we do not care about that can be ignored in this paper.

One can easily verify that the first term in theorem 1 coincides with the simplex geometry induced by the FIM. Note that the BM is invariant to reparameterization, and we can write it in the following equivalent form.

Corollary 2. *Under the reparameterization $\lambda_i = \exp(\theta_i)$ and $\rho(\theta, U) = U\,\mathrm{diag}(\exp(\theta))U^\top$, the BM is*

$$\begin{aligned} ds^2 = \sum_{i=1}^n &\Bigg[\frac{\exp(\theta_i)}{4} d\theta_i^2 + \exp(\theta_i) d\theta_i c_i^\top du_i \\ &+ \frac{1}{2} du_i^\top \sum_{j=1}^n \left[\frac{(\exp(\theta_i) - \exp(\theta_j))^2}{\exp(\theta_i) + \exp(\theta_j)} u_j u_j^\top\right] du_i\Bigg]. \end{aligned}$$

This parameterization is favored in our implementation because after a small movement in the θ-coordinates, the density matrix is still p.s.d.

The BM allows us to study *quantatively* the intrinsic change of the graph measured by ds^2. For example, a constant scaling of the edge weights results in $ds^2 = 0$ because the density matrix does not vary. The BM of the eigenvalue λ_i is proportional to $1/\lambda_i$, therefore as the network scales up and $n \to \infty$, the BM of the spectrum will scales up. By the Cauchy-Schwarz inequality, we have

$$\mathrm{tr}(\mathcal{G}(\lambda)) = \frac{1}{4}(1^\top \lambda^{-1})(1^\top \lambda) \geq \frac{n^2}{4}. \quad (6)$$

It is, however, not straightforward to see the scale of $\mathcal{G}(u_i)$, that is the BM w.r.t. the eigenvector u_i. We therefore have the following result.

Corollary 3. $\mathrm{tr}(\mathcal{G}(u_i)) = \frac{1}{2}\sum_{j=1}^n \frac{(\lambda_i - \lambda_j)^2}{\lambda_i + \lambda_j} \leq \frac{1}{2}$; $\mathrm{tr}(\mathcal{G}(U)) = \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n \frac{(\lambda_i - \lambda_j)^2}{\lambda_i + \lambda_j} \leq \frac{n}{2}$.

Remark 3.1. *The scale (measured by trace) of $\mathcal{G}(U)$ is $O(n)$ and U has $O(n^2)$ parameters. The scale of $\mathcal{G}(\lambda)$ is $O(n^2)$ and λ has $(n-1)$ parameters.*

Therefore, informally, the λ parameters carry more information than U. Moreover, it is computationally more expensive to parameterize U. We will therefore make our perturbations on the spectrum λ.

We need to make a low-rank approximation of ρ so as to reduce the degree of freedoms, and make our perturbation cheap to compute. Based on the Frobenius norm, the best low-rank approximation of any given matrix can be expressed by its largest singular values and their corresponding singular vectors. Similar results hold for approximating density matrix based on the BM. While BM is defined on an infinitesimal neighborhood, its corresponding non-local distance is known as the Bures distance $D_B(\rho_1, \rho_2)$ given by

$$D_B^2(\rho_1, \rho_2) = 2\left(1 - \operatorname{tr}\left(\sqrt{\rho_1^{\frac{1}{2}} \rho_2 \rho_1^{\frac{1}{2}}}\right)\right).$$

For diagonal matrices, the Bures distance reduces to the Hellinger distance up to constant scaling.

We have the following low-rank projection of a given density matrix.

Theorem 4. *Given $\rho_0 = U \operatorname{diag}(\lambda) U^\top$, where $\lambda_1, \cdots \lambda_n$ are monotonically non-increasing, its* rank-k *projection is*

$$\bar{\rho}_0^k = \mathop{\arg\min}_{\rho:\operatorname{rank}(\rho)=k} D_B(\rho, \rho_0) = \frac{\sum_{i=1}^k \lambda_i u_i u_i^\top}{\sum_{i=1}^k \lambda_i}.$$

Our proof in the supplementary material[1] is based on Theorem 3 [25]. We may simply denote $\bar{\rho}_0^k$ as $\bar{\rho}_0$ with the spectrum decomposition $\bar{\rho}_0 = \bar{U}_0 \operatorname{diag}(\bar{\lambda}_0) \bar{U}_0^\top$.

Hence, we can define a neighborhood of A by varying the spectrum of $\bar{\rho}^k(A)$. Formally, the graph Laplacian of the perturbed A w.r.t. the perturbation ϕ is

$$L(A(\phi)) = \operatorname{tr}(L(A))\ \rho(A(\phi)) \tag{7}$$

so that its trace is not affected by the perturbation, and the perturbed density matrix is

$$\rho(A(\phi)) = \rho(A) + \bar{U} \operatorname{diag}\left(\frac{\exp(\bar{\theta} + \phi)}{1^\top \exp(\bar{\theta} + \phi)} - \bar{\lambda}\right) \bar{U}^\top, \tag{8}$$

where the second low-rank term on the rhs is a perturbation of $\bar{\rho}^k(A)$ whose trace is 0 so that $\rho(A(\phi))$ is still a density matrix. The random variable ϕ follows

$$q_{\text{iso}}(\phi) = \mathcal{U}\left(\phi \,|\, 0, \mathcal{G}^{-1}(\bar{\theta})\right), \tag{9}$$

which can be either a Gaussian distribution or a uniform distribution[2], which has zero mean and precision matrix $\mathcal{G}(\bar{\theta})$ up to constant scaling. Intuitively, it has smaller variance on the directions with a large $\mathcal{G}$, so that $q(\phi)$ is intrinsically isotropic w.r.t. the BM.

In summary, our neighborhood of a graph with adjacency matrix A has k most informative dimensions selected by the BM, and is defined by eqs. (7) to (9). To compute this neighborhood, one needs to pre-compute the k largest eigenvectors of $\rho(A)$, which can be performed efficiently [28] for small k. LanczosNet [23] also utilizes an eigendeompositiona sub-module for a low-rank approximation of the graph Laplacian. Their focus is on building spectral filters rather than geometric perturbations. An empirical range of k is $10 \sim 50$.

One may alternatively parameterize a neighborhood by corrupting the graph links. However, it is hard to control the scale of the perturbation based on information theory and to have a compact parameterization.

5 FISHER-BURES ADVERSARY GCN

Based on the previous section 4, we know how to define the graph neighborhood. Now we are ready to implement our perturbed GCN, which we call the "FisherGCN".

We parameterize the perturbation as

$$\phi(\varphi, \varepsilon) = \mathcal{G}^{-1/2}(\bar{\theta}) \operatorname{diag}(\varphi)\varepsilon = \exp\left(-\frac{\bar{\theta}}{2}\right) \circ \varphi \circ \varepsilon, \tag{10}$$

where "$\circ$" means element-wise product, ϵ follows the uniform distribution over $[-\frac{1}{2}, \frac{1}{2}]^k$ or the multivariate Gaussian distribution, and corollary 2 is used here to get $\mathcal{G}(\bar{\theta})$. The vector $0 < \varphi \le \epsilon$ contains shape parameters (One can implement the constraint through reparameterization $\varphi = \epsilon/(1 + \exp(-\xi))$), where ϵ is a hyper-parameter specifying the radius of the perturbation. If $\varphi = \epsilon 1$, then ϕ follows $q_{\text{iso}}(\phi)$ in eq. (9). Then, one can compute the randomly perturbed density matrix $\rho(A(\phi))$ and corresponding Laplacian matrix $L(A(\phi))$ based on eqs. (7) and (8).

Our learning objective is to make predictions that is robust to such graph perturbations by solving the following

[1]The supplementary material is in the appendix of `https://arxiv.org/abs/1903.04154`. Our codes to reproduce all reported experimental results are available at `https://github.com/stellargraph/FisherGCN`.

[2]Strictly speaking, $\mathcal{U}(\phi \,|\, \mu, \Sigma)$ should be the pushforward distribution w.r.t. the Riemannian exponential map, which maps the distribution on the tangent space to the parameter manifold.

minimax problem

$$\min_{W} \max_{\varphi} -\frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} \log p(Y_i \mid X_i, A(\phi(\varphi, \varepsilon_j)), W), \tag{11}$$

where M (*e.g.* $M = 5$) is the number of perturbations. Similar to the training procedure of a GAN [14], one can solve the optimization problem by alternatingly updating φ along $\bigtriangledown\varphi$, the gradient w.r.t. φ, and updating W along $-\bigtriangledown W$.

For brevity, we highlight the key equations and steps (instead of a full workflow) of FisherGCN as follows:

① Normalize A (use the renormalization trick [21] or our algorithm 1 that will be introduced in section 6);

② Compute $\bar{\rho}^k(A)$ (theorem 4) by sparse matrix factorization [28] (only the top k eigenvectors of $\rho(A)$ is needed, and this needs only to be done for once);

③ Perform regular GCN optimization

  (a) Use eq. (10) to get the perturbation ϕ;
  (b) Use eqs. (7) and (8) to get the perturbed density matrix $\rho(A(\phi))$ and the graph Laplacian matrix $L(A(\phi))$;
  (c) Plug $\tilde{A} = I - \mathrm{tr}(L(A))\rho(A(\phi))$ into eq. (1).

Notice that the A matrix (and the graph Laplacian) is normalized in step ① before computing the density matrix, so that the multiple multiplications with A in different layers do not cause numerical instability. This can be varied depending on the implementation.

Our loss only imposes k (*e.g.* $k = 10 \sim 50$) additional free parameters (the rank of the projected $\bar{\rho}^k(A)$), while W contains the majority of the free parameters. As compared to GCN, we need to solve the k leading eigenvectors of $\rho(A)$ before training, and multiply the computational cost of training by a factor of M. Notice that $\rho(A)$ is sparse and the eigendecomposition of sparse matrix only need to be performed once. Instead of computing the perturbed density matrix explicitly, which is not sparse anymore, one only need to compute the correction term

$$\left[\bar{U}_{n\times k}\,\mathrm{diag}\left(\frac{\exp(\bar{\theta}+\phi)}{1^\top \exp(\bar{\theta}+\phi)} - \bar{\lambda}\right)\bar{U}_{k\times n}^\top\right] X_{n\times D}$$

which can be solved efficiently in $O(knD)$ time. If k is small, this computational cost can be ignored (with no increase in the overall complexity) as computing AX has $O(md)$ complexity (m is the number of links). In summary, our FisherGCN is several times slower than GCN with roughly the same number of free parameters and complexity.

FisherGCN can be intuitively understood as running multiple GCN in parallel, each based on a randomly perturbed graph. To implement the method does not require understanding our geometric theory but only to follow the list of pointers ①②③ shown above.

6 EXPERIMENTS

In this section, we perform an experimental study on semi-supervised transductive node classification tasks. We use three benchmark datasets, namely, the Cora, CiteSeer and PubMed citation networks [47, 21]. The statistics of these datasets are displayed in table 1. As suggested recently [38], we use random splits of training:validation:testing datasets based on the same ratio as the Planetoid split [47], as given in the "Train:Valid:Test" column in table 1.

We will mainly compare against GCN which can represent the state-of-the-art on these datasets, because our method serves as an "add-on" of GCN. We will discuss how to adapt this add-on to other graph neural networks in section 9 and refer the reader to [38] for how the performance of GCN compares against the other methods. Nevertheless, we introduce a stronger baseline called GCN^T. It was known that random walk similarities can help improve learning of graph neural networks [48]. We found that pre-processing the graph adjacency matrix A (with detailed steps listed in algorithm 1) can improve the performance of GCN on semisupervised node classification tasks[3] This processing is based on DeepWalk similarities [33] that are explicitly formulated in Table 1 [34]. Algorithm 1 involves two hyperparameters: the order $T \geq 1$ determines the order of the proximities (the larger, the denser the resulting A; $T = 1$ falls back to the regular GCN); the threshold $\nu > 0$ helps remove links with small probabilities to enhance sparsity. In the experiments we fix $T = 5$ and $\nu = 10^{-4}$. These procedures correspond to a polynomial filter with hand-crafted coefficients. One can look at table 1 and compare the sparsity of the processed adjacency matrix by algorithm 1 (in the "Sparsity^T" column) v.s. the original sparsity (in the "Sparsity" column) to have a rough idea on the computational overhead of GCN^T v.s. GCN.

Our proposed methods are denoted as FisherGCN and FisherGCN^T, which are respectively based on GCN and GCN^T. We fix the perturbation radius parameter $\epsilon = 0.1$ and the rank parameter $k = 10$.

The testing accuracy and loss are reported in table 2.

[3] During the review period of this paper, related works appeared [44, 1] which build a high order GCN. Comparatively, our GCN^T is closely based on DeepWalk similarities [33] instead of power transformations of the adjacency matrix.

Table 1: Dataset statistics. Note the number of links reported in previous works [21] counts duplicate links and some self-links, which is corrected here. "#Comps" means the number of connected components. "Sparsity" shows the sparsity of the matrix $\tilde{A}$. "Sparsity^T" shows its sparsity in GCN^T (with mentioned settings of T and ϵ).

Dataset	#Nodes	#Links	#Comps	#Features	#Classes	Train:Valid:Test	Sparsity	Sparsity^T
Cora	2,708	5,278	78	1,433	7	140:500:1000	0.18%	9.96%
CiteSeer	3,327	4,552	438	3,703	6	120:500:1000	0.11%	3.01%
PubMed	19,717	44,324	1	500	3	60:500:1000	0.03%	3.31%

Table 2: Testing loss and accuracy in percentage. The hyperparameters (learning rate 0.01; 64 hidden units; dropout rate 0.5; weight decay 5×10^{-4}) are selected based on the best overall testing accuracy of GCN on Cora and CiteSeer. Then we use these hyperparameters across all the four methods and three datasets. The reported mean±std scores are based on 200 runs (20 random splits; 10 different initializations per split). The splits used for hyperparameter selection and testing are different.

	Testing Accuracy			Testing Loss		
	Cora	CiteSeer	PubMed	Cora	CiteSeer	PubMed
GCN	80.52 ± 2.3	69.59 ± 2.0	78.17 ± 2.4	1.07 ± 0.04	1.36 ± 0.03	0.75 ± 0.04
FisherGCN	80.70 ± 2.2	69.80 ± 2.0	78.43 ± 2.4	1.06 ± 0.04	1.35 ± 0.03	0.74 ± 0.04
GCN^T	81.20 ± 2.3	70.31 ± 1.9	78.99 ± 2.6	1.04 ± 0.04	1.33 ± 0.03	0.70 ± 0.05
FisherGCN^T	$\mathbf{81.46} \pm 2.2$	$\mathbf{70.48} \pm 1.7$	$\mathbf{79.34} \pm 2.7$	$\mathbf{1.03} \pm 0.03$	$\mathbf{1.32} \pm 0.03$	$\mathbf{0.69} \pm 0.04$

Algorithm 1: Pre-process A to capture high-order proximities ($T \geq 2$ is the order; $\nu > 0$ is a threshold)

```
$A \leftarrow \mathrm{diag}^{-1}(A\mathbf{1})A$;
$S, B \leftarrow A$;
for $t \leftarrow 2$ to $T$ do
    $B \leftarrow BA$;
    $S \leftarrow S + B$;
$A \leftarrow \frac{1}{T} S \circ (1_{n\times n} - I)$;
$A \leftarrow A \circ (A > \nu)$;
$A \leftarrow A + A^\top + 2I$;
$A \leftarrow \mathrm{diag}^{-\frac{1}{2}}(A\mathbf{1})\, A\, \mathrm{diag}^{-\frac{1}{2}}(A\mathbf{1})$;
```

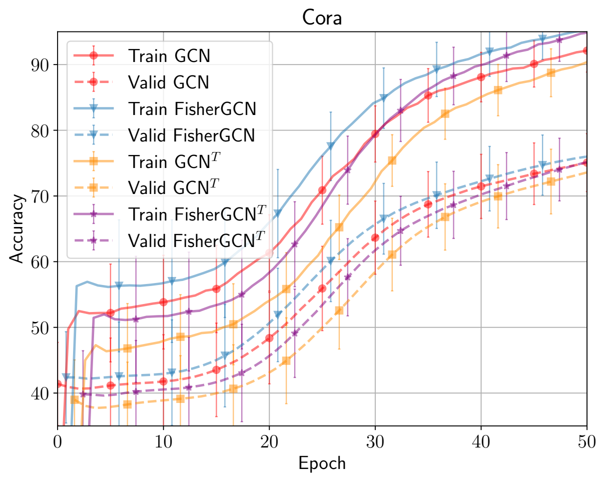

Figure 1: Learning curves (averaged over 200 runs) in accuracy on the Cora dataset.

We adapt the GCN codes [21] so that the four methods are compared in exactly the same settings and only differ in the matrix A that is used for computing the graph convolution. One can observe that FisherGCN and GCN^T can both improve over GCN, which means that our perturbation and the pre-processing by algorithm 1 both help to improve generalization. The best results are given by FisherGCN^T with both techniques added. Th large variation is due to different splits of the training:validation:testing datasets [38], and therefore these scores vary with the split. In repeated experiments, we observed a consistent improvement of the proposed methods as compared to the baselines.

Figure 1 shows the learning curves on the Cora dataset (see the supplementary material for the other cases). We can observe that the proposed perturbation presents higher training and testing scores during learning. The performance boost of FisherGCN is more significant if the number of epochs is limited to a small value.

7 AN EXTRINSIC GEOMETRY

In this section and the following section section 8, we present analytical results on the geometry of the manifold of graphs. These results are useful to interpret the proposed FisherGCN and are useful to understand graph-based machine learning.

We first derive an *extrinsic* geometry of a parametric graph embedded in a neural network. Based on this geometry, the learner can capture the curved directions of the loss surface and make more effective perturbations than the isotropic perturbation in eq. (8). While the intrinsic geometry in section 4 measures how much the graph itself has changed due to a movement on $\mathcal{M}$, the extrinsic geometry measures how varying the parameters of the graph will change the external model. Intuitively, if a dynamic ΔG causes little change based on the intrinsic geometry, one may also expect ΔG has little effect on the external neural network. However, in general, these two geometries impose different Riemannian metrics on the same manifold $\mathcal{M}$ of graphs.

Consider the predictive model represented by the conditional distribution $p(Y \mid X, A(\phi), W)$. Wlog consider ϕ is a scalar, which serves as a coordinate system of graphs. We use $\mathcal{G}^E$ to denote the extrinsic Riemannian metric (the upper script "E" is for extrinsic) that is to be distinguished with the intrinsic $\mathcal{G}$. Based on the GCN computation introduced in section 1, we can get an explicit expression of $\mathcal{G}^E$.

Theorem 5. *Let $\ell = -\log p(Y \mid X, A(\phi), W)$, $\Delta_l = \partial\ell/\partial H^l$ denote the back-propagated error of layer l's output H^l, and Σ_l' denote the derivative of layer l's activation function. Then*

$$\begin{aligned}
&\mathcal{G}^E(\phi)\\
&= \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{l=0}^{L-1}\left(H^l W^l(\Delta_{l+1}\circ\Sigma'_{l+1})^\top\frac{\partial\tilde{A}}{\partial\phi}\right)_{ii}\right)^2\\
&= \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{l=0}^{L-1}\left(H^l\Delta_l^\top\frac{\partial\tilde{A}}{\partial\phi}\right)_{ii}\right)^2;\\
&\mathcal{G}^E(W^l)\\
&= \frac{1}{N}\sum_{i=1}^{N}\Bigg(\mathrm{vec}\left((\Delta_{l+1}\circ\Sigma'_{l+1})^\top\tilde{A}H^l\right)\\
&\qquad\times\mathrm{vec}^\top\left((\Delta_{l+1}\circ\Sigma'_{l+1})^\top\tilde{A}H^l\right)\Bigg),
\end{aligned}$$

where vec() *means rearranging a matrix into a column vector.*

The information geometry of neural networks is mostly used to develop the second order optimization [32, 3], where $\mathcal{G}^E(W)$ is used. Here we are mostly interested in $\mathcal{G}^E(\phi)$, and our target is not for better optimization but to find a neighborhood of a given graph with large intrinsic variations. A movement with a large scale of $\mathcal{G}^E$ can most effectively change the predictive model $p(Y \mid X)$.

Let us develop some intuitions based on the term inside the trace on the rhs of $\mathcal{G}^E(\phi)$. In order to change the predictive model, the most effective edge increment da_{ij} should be positively correlated with $(h_i^{l\top}\Delta_{lj} + \Delta_{li}^\top h_j^l)$, which means how the hidden feature of node i (node j) is correlated with the increment of the hidden feature of node j (node i). This makes intuitive sense.

The meaning of theorem 5 is mainly theoretical, giving an explicit expression of $\mathcal{G}^E$ for the GCN model, which, to the best of the authors' knowledge, was not derived before (most literature studies the FIM of a feed-forward model such as a multi-layer perceptron). This could be useful for future works for natural gradient optimizers specifically tailored for GCN. On the practical side,

Theorem 5 also helps to understand the proposed minimax optimization. On the manifold $\mathcal{M}$ of graphs, we make the rough assumption that $A(0) = A$ is a local minimum of ℓ along the ϕ coordinate system, that is, adding a small noise to A will always cause an increment in the loss. The random perturbation in eq. (10) corresponds to the distribution

$$q(\phi \mid \varphi) = \mathcal{U}\left(\phi \mid 0, \mathcal{G}^{-1/2}(\bar{\theta})\,\mathrm{diag}(\varphi\circ\varphi)\mathcal{G}^{-1/2}(\bar{\theta})\right),$$

and our loss function in eq. (11) is obtained by applying the reparameterization trick [20] to solve the expectation in eq. (1). If $\varphi = \epsilon 1$, then $q(\phi \mid \varphi) = \mathcal{U}(\phi \mid 0, \epsilon^2\mathcal{G}(\bar{\theta})^{-1})$ falls back to the isotropic $q_{\mathrm{iso}}(\phi)$. Letting φ free allows the neighborhood to deform (see fig. 2 left). Then, through the maximization in eq. (11) w.r.t. φ, the density $q(\phi)$ will focus on the neighborhood of the original graph A where the loss surface is most upcurved (see fig. 2 right). These directions have large $\mathcal{G}^E$ and make the perturbation effective in terms of the FIM of the graph neural network. Consider the reverse case, when the density $q(\phi)$ corresponds to small values of $\mathcal{G}^E$. Such perturbations are long the flat directions of the loss surface and will have little effect on learning the predictive model.

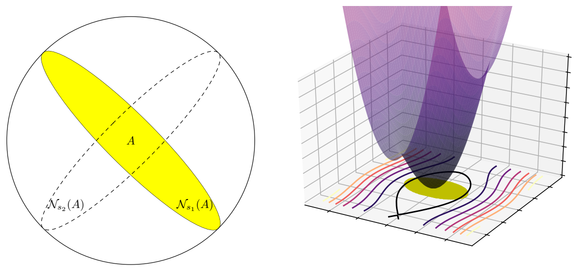

Figure 2: Learning a neighborhood (yellow region) of a graph where the loss surface is most curved corresponding to large $\mathcal{G}^E$.

8 AN EMBEDDING GEOMETRY

We present a geometry of graphs which is constructed in the spatial domain and is closely related to graph embeddings [33]. Consider representing a graph by a node similarity matrix $W_{n\times n}$ (*e.g.* based on algorithm 1), which is row-normalized and has zero-diagonal entries. These similarities are assumed to be based on a latent graph embedding $Y_{n\times d}$: $p_{ij}(Y) = \frac{1}{Z_i}\exp\left(-\|y_i - y_j\|^2\right)$, where $P_{n\times n}$ is the generative model with the same constraints as the W matrix, and Z_i is the partition function. Then, the observed FIM (that leads to the FIM as the number of observations increase) is given by the Hessian matrix of $\mathrm{KL}(W : P(Y))$ evaluated at the maximum likelihood estimation $Y^\star = \arg\min_Y \mathrm{KL}(W : P(Y))$, where KL denotes the Kullback-Leibler divergence. We have the following result.

Theorem 6. *W.r.t. the generative model $p_{ij}(Y)$, the diagonal blocks of the observed FIM $\hat{\mathcal{G}}$ of a graph represented by the similarity matrix W is*

$$\begin{aligned}\hat{\mathcal{G}}(y^k) =& 4L(W - P(Y)) + 8L(P(Y)\circ D^k)\\ &- 4(B^k)^\top B^k,\end{aligned}$$

where y^k is the k'th column of Y, $L(W - P(Y))$ is the Laplacian matrix computed based on the indefinite weights $(W - P(Y))$ after symmetrization, $D^k = (y_{ik} - y_{jk})^2$, and $B^k = L(p_{ij}(y_{ik} - y_{jk}))$.

The theorem gives the observed FIM, while the expected FIM (the 2nd and 3rd terms in theorem 6) can be alternatively derived based on [39]. To understand this result, we can assume that $P(Y) \to W$ as the number of observations increase. Then

$$\begin{aligned}dy^{k\top}\hat{\mathcal{G}}(y^k)dy^k = 4\sum_{i=1}^{n}\Bigg[&\sum_{j=1}^{n} p_{ij}(y_{ik}-y_{jk})^2(dy_{ik}-dy_{jk})^2\\ &-\left(\sum_{j=1}^{n} p_{ij}(y_{ik}-y_{jk})(dy_{ik}-dy_{jk})\right)^2\Bigg]\end{aligned}$$

is in the form of a variance of $(y_{ik}-y_{jk})(dy_{ik}-dy_{jk}) = \frac{1}{2}d(y_{ik} - y_{jk})^2$ w.r.t. p_{ij}. Therefore a large Riemannian metric $dy^{k\top}\hat{\mathcal{G}}(y^k)dy^k$ corresponds to a motion dy^k which cause a large variance of neighbor's distance increments. For example, a rigid motion, or a uniform expansion/shrinking of the latent network embedding will cause little or no effect on the variance of $d(y_{ik} - y_{jk})^2$, and hence corresponds to a small distance in this geometry.

This metric can be useful for developing theoretical perspectives of network embeddings, or build spatial perturbations of graphs (instead of our proposed spectral perturbation). As compared to the intrinsic geometry in section 4, the embedding geometry is based on a generative model instead of the BM. As compared to the extrinsic geometry in section 7, the embedding geometry is not related to a neural network model.

9 CONCLUSION AND DISCUSSIONS

We imported new tools and adapted the notations from quantum information geometry to the area of geometric deep learning. We discussed three different geometries on the ambient space of graphs, with their Riemannian metrics provided in closed form. The results and adaptations are useful to develop new deep learning methods. We demonstrated their usage by perturbing graph structures in a GCN, showing consistent improvements in transductive node classification tasks.

It is possible to generalize FisherGCN to a scalable setting, where a mini-batch only contains a sub-graph [17] of $m \ll n$ nodes. This is because our perturbation has a low-rank factorization given by the second term in eq. (8). One can reuse this spectrum factorization of the global matrix to build sub-graph perturbations.

If A contains free-parameters [42], one can compute the low-rank projection $\bar{\rho}^k(A)$ using the original graph that is parameter free, based on which the perturbation term can be constructed. Alternatively, one can periodically save the graph and recompute $\bar{\rho}^k(A)$ during learning.

Based on [21], we express a graph convolution operation on an input signal $x = \sum_{i=1}^n \alpha_i u_i \in \Re^n$ as

$$[I - \mathrm{tr}(L)\rho(A)]\,x = x - \mathrm{tr}(L)E_{i\sim\lambda}(\alpha_i u_i),$$

where E denotes the expectation. The von Neumann entropy of the quantum state ρ is defined by the Shannon entropy of λ, that is $-\sum_{i=1}^n \lambda_i \log\lambda_i$. If we consider a higher order convolutional operator (in plain polynomial), given by

$$\rho^\omega(A)x = \frac{1}{\lambda^\omega 1}U\,\mathrm{diag}(\lambda)^\omega U^\top x = E_{i\sim\frac{\lambda^\omega}{\lambda^\omega 1}}(\alpha_i u_i).$$

The von Neumann entropy is monotonically decreasing as $\omega \geq 1$ increases. As $\omega\to\infty$, we have $\rho^\omega(A)x \to \alpha_1 u_1$ (if λ_1 is the largest eigenvalue of $\rho(A)$ without multiplicity). Therefore, high order convolutions enhance the signal w.r.t. the largest eigenvectors of ρ. Therefore our perturbation is equivalent to adding high order polynomial filters. It is interesting to explore alternative perturbations based on other distances, *e.g.* matrix Bregman divergence [31]. An empirical stay on comparing different types of perturbations in the GCN setting is left as future work.

Acknowledgements

The authors gratefully thank the anonymous UAI reviewers and Frank Nielsen for their valuable and constructive comments. We thank the authors [47, 21, 38] for making their processed datasets and codes public. This work is supported by StellarGraph@Data61.

References

[1] S. Abu-El-Haija, B. Perozzi, A. Kapoor, N. Alipourfard, K. Lerman, H. Harutyunyan, G. V. Steeg, and A. Galstyan. MixHop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *ICML*, volume 97, pages 21–29. PMLR, 2019.

[2] S. Amari. *Information Geometry and Its Applications*, volume 194 of *Applied Mathematical Sciences*. Springer-Verlag, Berlin, 2016.

[3] S. Amari, R. Karakida, and M. Oizumi. Fisher information and natural gradient learning in random deep networks. In *AISTATS*, volume 89 of *PMLR*, pages 694–702, 2019.

[4] R. Bhatia, T. Jain, and Y. Lim. On the Bures–Wasserstein distance between positive definite matrices. *Expositiones Mathematicae*, 2018.

[5] B. Biggio, B. Nelson, and P. Laskov. Support vector machines under adversarial label noise. In *ACML*, volume 20 of *PMLR*, pages 97–112, 2011.

[6] S. L. Braunstein, S. Ghosh, and S. Severini. The Laplacian of a graph as a density matrix: a basic combinatorial approach to separability of mixed states. *Annals of Combinatorics*, 10(3):291–317, 2016.

[7] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs. In *ICLR*, 2014.

[8] D. Bures. An extension of Kakutani's theorem on infinite product measures to the tensor product of semifinite w^*-algebras. *Transactions of the AMS*, 135:199–212, 1969.

[9] N. N. Čencov. *Statistical Decision Rules and Optimal Inference*, volume 53 of *Translations of Mathematical Monographs*. American Mathematical Society, 1982. (Published in Russian in 1972).

[10] J. Chen, J. Zhu, and L. Song. Stochastic training of graph convolutional networks with variance reduction. In *ICML*, volume 80 of *PMLR*, pages 942–950, 2018.

[11] S. Chow, W. Li, and H. Zhou. A discrete Schrödinger bridge problem via optimal transport on graphs. *Journal of Functional Analysis*, 276(8):2440–2469, 2019.

[12] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, pages 3844–3852. Curran Associates, Inc., 2016.

[13] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *NIPS 28*, pages 2224–2232. Curran Associates, Inc., 2015.

[14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[15] M. Gori, G. Monfardini, and F. Scarselli. A new model for learning in graph domains. In *IJCNN*, volume 2, pages 729–734, 2005.

[16] A. Grover and J. Leskovec. Node2Vec: Scalable feature learning for networks. In *KDD*, pages 855–864, 2016.

[17] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *NIPS*, pages 1024–1034. Curran Associates, Inc., 2017.

[18] M. Hübner. Explicit computation of the Bures distance for density matrices. *Physics Letters A*, 163(4):239 – 242, 1992.

[19] M. Kantarcıoğlu, B. Xi, and C. Clifton. Classifier evaluation and attribute selection against active adversaries. *Data Mining and Knowledge Discovery*, 22(1):291–335, 2011.

[20] D. P. Kingma and M. Welling. Auto-encoding variational Bayes. In *ICLR*, 2014.

[21] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[22] Q. Li, Z. Han, and X.-M. Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*, 2018.

[23] R. Liao, Z. Zhao, R. Urtasun, and R. S. Zemel. LanczosNet: Multi-scale deep graph convolutional networks. In *ICLR*, 2019.

[24] J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid. Convolutional kernel networks. In *NIPS*, pages 2627–2635. Curran Associates, Inc., 2014.

[25] D. Markham, J. Adam Miszczak, Z. Puchała, and K. Życzkowski. Quantum state discrimination: a geometric approach. *Phys. Rev. A*, 77:042111, 2008.

[26] F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, and M. M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model CNNs. In *CVPR*, pages 5425–5434, 2017.

[27] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. DeepFool: A simple and accurate method to fool deep neural networks. In *CVPR*, pages 2574–2582, 2016.

[28] C. Musco and C. Musco. Randomized block Krylov methods for stronger and faster approximate singular value decomposition. In *NIPS 28*, pages 1396–1404. Curran Associates, Inc., 2015.

[29] B. Muzellec and M. Cuturi. Generalizing point embeddings using the Wasserstein space of elliptical distributions. In *NeurIPS 31*, pages 10237–10248. Curran Associates, Inc., 2018.

[30] F. Nielsen and R. Bhatia. *Matrix Information Geometry*. Springer-Verlag Berlin Heidelberg, 2013.

[31] R. Nock, B. Magdalou, E. Briys, and F. Nielsen. Mining matrix data with Bregman matrix divergences for portfolio selection. In F. Nielsen and R. Bhatia, editors, *Matrix Information Geometry*, pages 373–402. Springer Berlin Heidelberg, 2013.

[32] R. Pascanu and Y. Bengio. Revisiting natural gradient for deep networks. In *ICLR*, 2014.

[33] B. Perozzi, R. Al-Rfou, and S. Skiena. DeepWalk: Online learning of social representations. In *KDD*, pages 701–710, 2014.

[34] J. Qiu, Y. Dong, H. Ma, J. Li, K. Wang, and J. Tang. Network embedding as matrix factorization: Unifying DeepWalk, LINE, PTE, and Node2Vec. In *WSDM*, pages 459–467, 2018.

[35] A. Saade, F. Krzakala, and L. Zdeborová. Spectral clustering of graphs with the Bethe Hessian. In *NIPS 27*, pages 406–414. Curran Associates, Inc., 2014.

[36] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[37] J. Schmidhuber. Learning factorial codes by predictability minimization. *Neural Computation*, 4(6):863–879, 1992.

[38] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann. Pitfalls of graph neural network evaluation. In *NeurIPS Workshop on Relational Representation Learning*, 2018.

[39] K. Sun and S. Marchand-Maillet. An information geometry of statistical manifold learning. In *ICML 31*, volume 32 of *PMLR*, pages 1–9, 2014.

[40] K. Sun and F. Nielsen. Relative Fisher information and natural gradient for learning large modular models. In *ICML 34*, volume 70 of *PMLR*, pages 3289–3298, 2017.

[41] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry. Robustness may be at odds with accuracy. In *ICLR*, 2019.

[42] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. *ICLR*, 2018.

[43] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R D. Hjelm. Deep graph infomax. In *ICLR*, 2019.

[44] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger. Simplifying graph convolutional networks. In *ICML*, volume 97 of *PMLR*, pages 6861–6871, 2019.

[45] H. Xiao, B. Biggio, B. Nelson, H. Xiao, C. Eckert, and F. Roli. Support vector machines under adversarial label contamination. In *Neurocomputing*, volume 160, pages 53–62, 2015.

[46] K. Xu, C. Li, Y. Tian, T. Sonobe, K. Kawarabayashi, and S. Jegelka. Representation learning on graphs with jumping knowledge networks. In *ICML 35*, volume 80 of *PMLR*, pages 5453–5462, 2018.

[47] Z. Yang, W. W. Cohen, and R. Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *ICML*, volume 48 of *PMLR*, pages 40–48, 2016.

[48] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *KDD*, pages 974–983, 2018.

[49] G. Zhang, S. Sun, D. Duvenaud, and R. Grosse. Noisy natural gradient as variational inference. In *ICML*, volume 80 of *PMLR*, pages 5852–5861, 2018.

[50] C. Zhao, P. T. Fletcher, M. Yu, Y. Peng, G. Zhang, and C. Shen. The adversarial attack and detection under the Fisher information metric. In *AAAI*, 2019.