arXiv:1903.04154v1 [cs.LG] 11 Mar 2019

Fisher-Bures Adversary Graph Convolutional Networks

Ke Sun[†] **Piotr Koniusz**[†⋆] **Jeff Wang**[†]
†CSIRO Data61 ⋆Australia National University
{First.Last}@data61.csiro.au

Abstract

In a graph convolutional network, we assume that the graph G is generated with respect to some observation noise. We make small random perturbations ΔG of the graph and try to improve generalization. Based on quantum information geometry, we can have quantitative measurements on the scale of ΔG. We try to maximize the intrinsic scale of the permutation with a small budget, while minimize the loss based on the perturbed $G + \Delta G$. Our proposed model can consistently improve graph convolutional networks on semi-supervised node classification tasks with reasonable computational overhead. We present two different types of geometry on the manifold of graphs: one is for measuring the intrinsic change of a graph; the other is for measuring how such changes can affect externally a graph neural network. These new analytical tools will be useful in development a good understanding of graph neural networks and fostering new techniques.

1 Introduction

Recently neural network architectures are introduced [BZSL14, DBV16, KW17, HYL17, VCC+18] to learn high level features of objects using "graph convolutions" w.r.t. a given graph among these objects. These graph convolutional networks (GCNs) show record-breaking scores on graph-based learning tasks. Similar to the idea of data augmentation, we propose to improve 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 rigorous mathematical definitions of the *manifold of graphs*, which includes all graphs satisfying certain parameterisation, with their parameters varying smoothly. Then, we perturb the graph along the directions where the intrinsic distance, that is also the information divergence, is maximized. We show empirically that the performance of GCN can be improved and present theoretical analysis of this geometry.

Notations

In this paper, 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 $Y_{n\times M}$ 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 i and j. We use capital letters such as A, B, $\cdots$ to denote matrix 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 [KW17] based on [DBV16]), the network architecture is recursively defined by

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

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. We use $W = \{W^l\}_{l=1}^L$ to denote all network weights. $\tilde{A}_{n\times n}$ is the normalized adjacency matrix so that $\tilde{A} = (D+I)^{-\frac{1}{2}}(A+I)(D+I)^{-\frac{1}{2}}$, $D = \mathrm{diag}(A1)$ is the degree matrix, 1 is the vector of all ones, $\mathrm{diag}()$ means a diagonal matrix using the given diagonal entries, and σ is an element-wise nonlinear activation function (it could be different

for different layers and the notation is slightly abused). Based on a set of given samples X and optionally the corresponding labels Y, learning is implemented by

$$\min_W \ell\left(X, Y, A, W\right), \tag{2}$$

where ℓ is a loss (*e.g.* cross-entropy), usually expressed in terms of Y and H^L, the feature matrix obtained by stacking L 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 replace the optimization problem in eq. (2) by

$$\min_W \int q(\phi \,|\, \varphi)\ell\left(X, Y, A(\phi), W\right) d\phi, \tag{3}$$

where $A(\phi)$ is a parameterisation of graph adjacency matrices, $A(0) = A$ is the original adjacency matrix, $q(\phi \,|\, \varphi)$ is a zero-centered density, defining a random $A(\phi)$ in a "neighbourhood" of A, φ is the parameters of this neighbourhood.

This brings up a set of fundamental questions: ① How to define the manifold of graphs $\{A(\phi)\}$? ② How to define the neighbourhood $q(\phi \,|\, \varphi)$? ③ How to choose the neighbourhood parameters φ? We will build a geometric solution of these problems, provide the exact formulations to implement eq. (3), and test the optimization can improve GCN generalization. Our contributions are both theoretical and practical, which are summarized as follows:

- We bridge new tools from quantum information theory into graph neural networks, and provide geometric formulations to make measurements of graphs;
- We propose a graph convolutional network that can consistently improve over GCN [KW17] on standard benchmarks (see table 2);
- We introduce an algorithm to perform polynomial transformation of the graph adjacency matrix so as to incorporate high order proximities in GCN.

The rest of this paper is organized as follows. We first review related works in section 2. Section 3 introduces some basics of quantum information geometry. Sections 4 and 5 build our theoretical formulations to solve the fundamental questions listed above. Section 6 presents our proposed method. Section 7 shows its performance on semi-supervised node classification. Section 8 provides more analytical results. 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

Graph Convolutional Network (GCN) [KW17] is a state-of-the-art network 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, GraphSAGE [HYL17] takes convolution from spectral to spatial domain defined by the local neighborhood of each node. The average pooling on nearest neighbourhoods 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 [PARS14, GL16] capture structural similarities in the graph. DeepWalk [PARS14] 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 [GL16] interpolates between breadth- and depth-first sampling strategies to aggregate over different types of neighborhood. MoNet [MBM$^+$17] generalizes the notion of coordinate spaces by learning a set of parameters of Gaussian functions to encode some distance for the node embedding, *e.g.* difference between degrees of a pair of nodes. GAT [VCC$^+$18] learns such weights via a self-attention mechanism. Jumping Knowledge Networks [XLT$^+$18] also target the notion of node locality. Experiments on JK-Net 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-Net aggregates over various neighborhoods thus considers multiple node localities.

By contrast, we compute FIM of the graph Laplacian and apply mild adversary perturbations to its parametrization to improve generalization. Thus, we infer a 'correction' of the Laplacian matrix while JK-Net aggregates multiple node localities.

Sampling strategy has also impact on the total size of receptive fields. In vanilla GCN [KW17], 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 oversmoothing of signal. Thus, stochastic GCN [CZS18] control the variance of the activation estimator by keeping the history/summary of activations in

the previous layer to be reused.

Our work is somewhat related to Deep Graph Infomax [VFH+19]. 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 on the parameters of FIM to flatten the most abrupt curvature directions. DGI relies on the notion of positive and negative samples. In contrast, we learn maximally perturbing parameters of our extrinsic graph representation which are the analogy to negative sampels.

Lastly, noteworthy are application driven pipelines such as ChebiNet [DMI+15], the molecule classification pipeline.

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 [MFF], 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 in the community, especially in the shallow setting [BNL11, KXC11, XBN+15].

Adversarial attacks under the FIM [ZFY+19] propose to carry out perturbations in the spectral domain. Given a quadratic form of FIM, the optimal adversarial perturbation is given by the first eigenvector corresponding to the largest eigenvalue. The larger the eigenvalues of 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 FIM for the purpose of extrinsically parameterize the graph Laplacian. We perform a maximization w.r.t. these parameters to flatten the FIM around its local optimum which corresponds to decreasing eigenvalues in [ZFY+19], thus making our approach well regularized in the sense of flattening the largest curvature associated with FIM (improved smoothness). With the smoothness, the classification performance typically degrades, *e.g*. see the impact of smoothness on kernel representations [MKHS14]. Indeed, study [TSE+19] 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 [ZFY+19]) 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 section 7.

Quantum Information Geometry

Natural gradient [Ama16, Ama19, PB14, ZSDG17, SN17] 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 [SKZ14], 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 parameterisation also draws on this view.

Tools from quantum information geometry are applied to machine learning [BJL18, MC18] but not yet ported to the domain of graph neural networks. In information geometry one can have different matrix divergences [NB13] that can be applied on p.s.d. matrices. We point the reader to related definitions of the discrete Fisher information [CLZ17] without illuminate the details.

3 Prerequisites

Fisher Information Metric

The discipline of information geometry [Ama16] 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

$$\mathcal{G}_{ij}(\Theta) = \int p(X \mid \Theta) \frac{\log p(X \mid \Theta)}{\partial \Theta} \frac{\log p(X \mid \Theta)}{\partial \Theta^\top} \mathrm{d}X. \tag{4}$$

By definition, we must have $\mathcal{G}(\Theta) \succeq 0$. Following H. Hotelling and C. R. Rao, this $\mathcal{G}(\Theta)$ is used (see *e.g*. section 3.5 [Ama16]) to define the Riemannian metric of a statistical model $\mathcal{M} = \{\Theta : p(X \mid \Theta)\}$, which is known as the Fisher information metric (FIM) $ds^2 = d\Theta^\top \mathcal{G}(\Theta) d\Theta$. Intuitively, ds^2 is small when $d\Theta$ makes less *intrinsic* change of the model. The metric is invariant to reparameterisation and is the unique Riemannian metric in the space of probability distributions under certain conditions [Čen82, Ama16].

Bures Metric

In quantum mechanics, a quantum state is represented by a graph (see *e.g.* [BGS16]). 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 definition of Fisher information to define a geometry of the Θ space. The quantum version of the Fisher information matrix is

$$\mathcal{G}_{ij}(\Theta) = \frac{1}{2}\mathrm{tr}\left[\rho(\Theta)(\partial L_i \partial L_j + \partial L_i \partial L_j)\right], \quad (5)$$

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. (5) will recover the traditional Fisher information defined in eq. (4). The quantum Fisher information metric $ds^2 = d\Theta^\top \mathcal{G}(\Theta) d\Theta$, upto constant scaling, is known as the Bures metric [Bur69]. We use BM to denote these equivalent metrics, and use $\mathcal{G}$ 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 ported to the graph domain [BGS16].

4 Intrinsic Geometry of Graphs

In this section, we define an intrinsic geometry of graphs based on the BM, so that we can measure distances on the manifold consisting of all graphs and have the notion of *neighbourhood*.

We parameterize a graph by its density matrix

$$\rho = U\,\mathrm{diag}(\lambda)U^\top = \sum_{i=1}^{n} \lambda_i u_i u_i^\top \succeq 0, \quad (6)$$

where $UU^T = U^TU = I$ 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 parameterisation was already derived in closed form (see eq.(10) [H$\ddot{\text{O}}$92]), given by

$$ds^2 = \frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{n} \frac{(u_j^\top d\rho u_k)^2}{\lambda_j + \lambda_k}. \quad (7)$$

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. (7), we get the following theorem.

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

$$ds^2 = \sum_{i=1}^{n}\left[\frac{1}{4\lambda_i}d\lambda_i^2 + d\lambda_i c_i^\top du_i + \frac{1}{2}du_i^\top \sum_{j=1}^{n}\left(\frac{(\lambda_i-\lambda_j)^2}{\lambda_i+\lambda_j}u_j u_j^\top\right)du_i\right],$$

where c_i are some coefficients which we do not care about.

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 reparameterisation, and we can write it in the following equivalent form.

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

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

This parameterisation 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 $d\Theta$ 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

$$\mathrm{tr}(\mathcal{G}(\lambda)) = \frac{1}{4}(1^\top\lambda^{-1})(1^\top\lambda) \geq \frac{n^2}{4}. \quad (8)$$

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 parameters. Therefore, informally, the λ parameters carry more information than 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 a p.s.d. matrix can be expressed by its largest eigenvalues and their corresponding eigenvectors. This is also true, although not straightforward, for approximating density matrix based on the BM. While BM is defined on an infinitesimal patch, its corresponding non-local distance is known as the Bures distance

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

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*

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

Our proof in the supplementary material is based on Theorem 3 [MMPidZ08]. We may simply denote $\bar{\rho}_0^k$ as $\bar{\rho}_0$, which has a low rank spectrum decomposition $\bar{\rho}_0 = \bar{U}_0 \operatorname{diag}(\bar{\lambda}_0) \bar{U}_0^\top$.

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

$$L(A(\phi)) = \operatorname{tr}(L(A))\rho(A(\phi)) \tag{9}$$

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

$$\rho(A(\phi)) = \rho(A) + \bar{U} \operatorname{diag}\left(\frac{\exp(\bar{\theta} + \phi)}{1^\top \exp(\bar{\theta} + \phi)} - \bar{\lambda}\right) \bar{U}^\top, \tag{10}$$

where the second 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, and the random variable ϕ follows

$$q_{\text{iso}}(\phi) = \mathcal{U}\left(\phi \,|\, 0, \mathcal{G}^{-1}(\bar{\theta})\right), \tag{11}$$

which can be either a Gaussian distribution or a uniform distribution[1], which has zero mean and its precision matrix defined by $\mathcal{G}(\bar{\theta})$. Intuitively, it has smaller variance on the directions with a large $\mathcal{G}$, so that $q(\phi)$ is intrinsically isotropic. In our experiments $\mathcal{U}$ is the uniform distribution.

In summary, our neighbourhood of a graph with adjacency matrix A has k most informative dimensions selected by the BM, and is defined by eqs. (9) to (11). To compute this neighbourhood one needs to perform a matrix factorization to extract the k largest eigenvectors of $\rho(A)$. One may alternatively parameterize a neighbourhood by corrupt the links. However this approach is empirical and it may be tricky to have a compact parameterisation.

[1] 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 of ϕ.

5 Extrinsic Geometry of Graphs

In this section we derive an "extrinsic" geometry of a parametric graph that is embedded into a neural network model, so that we can capture the curved directions of the loss surface and make more effective perturbations than the isotropic perturbation in eq. (10). This extrinsic geometry measures how varying the parameters of the graph will change the external model, rather than how much the graph itself has changed due to the movement, which is measured by the intrinsic geometry. Intuitively, if a dynamic ΔG causes little change based on the intrinsic geometry, one may also expect ΔG has little affect on the external neural network. However, in general, these two geometries impose different Riemannian metrics on the same manifold of graphs.

Consider the predictive model represented by the conditional distribution $p(Y \,|\, X, A(\phi), W)$. Wlog consider ϕ is a scalar, which serves as a coordinate system of graphs. w.r.t. a set of observations $\{X_i, Y_i\}_{i=1}^N$, and based on eq. (4), we have

$$\mathcal{G}^E(\phi) = \frac{1}{N} \sum_{i=1}^N \left(\frac{\partial \log p(Y_i \,|\, X_i, A(\phi), W)}{\partial \phi}\right)^2. \tag{12}$$

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 in section 1, we can get an explicit expression of $\mathcal{G}^E$.

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

vation function. Then

$$\mathcal{G}^E(\phi) = \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{l=0}^{L-1}\operatorname{tr}\left(H^l W^l(\Delta_{l+1}\circ\Sigma'_{l+1})^\top\frac{\partial\tilde{A}}{\partial\phi}\right)\right)^2 = \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{l=0}^{L-1}\operatorname{tr}\left(\frac{\partial\tilde{A}}{\partial\phi}H^l\Delta_l^\top\right)\right)^2;$$

$$\mathcal{G}^E(W^l) = \frac{1}{N}\sum_{i=1}^{N}\left(\operatorname{vec}\left((\Delta_{l+1}\circ\Sigma'_{l+1})^\top\tilde{A}H^l\right) \times \operatorname{vec}^\top\left((\Delta_{l+1}\circ\Sigma'_{l+1})^\top\tilde{A}H^l\right)\right),$$

where "$\circ$" is element-wise product, $\operatorname{vec}()$ *means rearranging a matrix into a column vector.*

The information geometry of neural networks is mostly used to develop the second order optimization [PB14, Ama19], where $\mathcal{G}^E(W)$ is used. In contrast, we are mostly interested in $\mathcal{G}^E(\phi)$, and our target is not for better optimization but to find a neighbourhood of a given graph with large intrinsic variations. A movement with large scale of $\mathcal{G}^E$ can most effectively change the predictive model $p(Y\,|\,X)$.

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 be positively correlated with $(h_i^{l\top}\Delta_{lj}+\Delta_{li}^\top h_j^l)$, which means how the feature of node i (node j) is correlated with the increment of node j (node i).

The meaning of theorem 5 is mainly theoretical, giving an explicit expression of $\mathcal{G}$ for the GCN model, which, to the best of the authors' knowledge, was not given 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, it is unfortunately hard to directly work on $\mathcal{G}^E(\phi)$. We will therefore seek for an empirical approximation which is explained as follows.

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. We introduce some shape parameters φ of the perturbation and re-formulate eq. (11) as

$$q(\phi\,|\,\varphi)=\mathcal{U}\left(\phi\,|\,0,\mathcal{G}^{-1/2}(\bar{\theta})\operatorname{diag}(\varphi\circ\varphi)\mathcal{G}^{-1/2}(\bar{\theta})\right), \tag{13}$$

where $0<\varphi\le\epsilon$ element-wisely (one can implement the constraint through reparameterisation $\varphi=\epsilon/(1+\exp(-\xi))$), $\varphi\circ\varphi$ is just the element-wise square, and ϵ is a hyper-parameter specifying the radius of the perturbation. If $\varphi=\epsilon 1$, then $q(\phi\,|\,\varphi)=\mathcal{U}(\phi\,|\,0,\epsilon^2\mathcal{G}(\bar{\theta})^{-1})$ falls back to the isotropic $q_{\text{iso}}(\phi)$. Letting φ free allows the neighbourhood to deform. Then, we *maximize* the expected loss $E_{\phi\sim\,q(\phi\,|\,\varphi)}-\log(Y\,|\,X,A(\phi),W)$ w.r.t. φ, so that the density $q(\phi)$ will focus on the neighbourhood of the original graph A where the loss surface is most upcurved (see fig. 1). This approach has the advantage that the original optimization in eq. (3) are replaced with a min-max problem of the same loss, which can be solved conveniently.

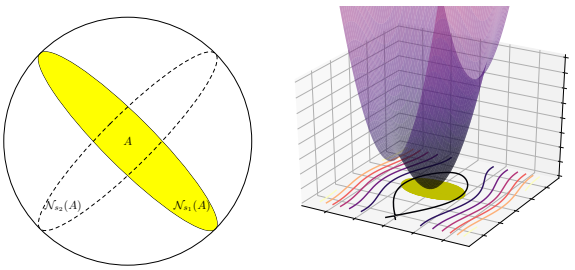

Figure 1: Learning a neighbourhood (yellow region) of a graph where the loss surface is most curved corresponding to large $\mathcal{G}^E$.

6 Fisher-Bures Adversary GCN

Based on the previous sections 4 and 5, we have the explicit expressions of $q(\phi\,|\,\varphi)$ and $A(\phi)$, and we know how to optimize the neighbourhood parameters φ. Now we are ready to implement our perturbed GCN which we call the FisherGCN.

To solve the expectation in eq. (3), we apply the reparameterisation trick [KW14] and express a random sample of $q(\phi\,|\,\varphi)$ based on eq. (13) as

$$\phi(\varphi,\varepsilon)=\mathcal{G}^{-1/2}(\bar{\theta})\operatorname{diag}(\varphi)\varepsilon=\exp\left(-\frac{\bar{\theta}}{2}\right)\circ\varphi\circ\varepsilon, \tag{14}$$

where 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 $\mathcal{G}(\bar{\theta})$. Then one can compute $\rho(A(\phi))$ and corresponding Laplacian matrix $L(A(\phi))$ based on eqs. (9) and (10).

Our optimization problem can be expressed as

$$\min_W\max_\varphi -\frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}\log p(Y_i\,|\,X_i,A(\phi(\varphi,\varepsilon_j)),W). \tag{15}$$

Similar to the training procedure of a GAN [GPAM+14], one can solve the optimization problem by alternatingly

updating φ along $\bigtriangledown\varphi$, the gradient of ℓ w.r.t. φ, and updating W along $-\bigtriangledown W$. These gradients can be solved conveniently using an auto-differentiation framework.

We highlight the key equations and steps (instead of a full workflow) of the proposed method as follows.

① normalize A (use the renormalization trick on page 3 [KW17] or our algorithm 1);

② compute $\bar{\rho}^k(A)$ (theorem 4) by sparse matrix factorization (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. (14) to get the perturbation ϕ;
(b) Use eqs. (9) and (10) to get the perturbed density matrix $\rho(A(\phi))$ and the 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 before computing the density matrix, so that the multiple multiplications with A in different layers does not cause numerical instability. This is only an implementation detail.

Our loss ℓ only imposes k (that is fixed to $k = 10$ in our experiments) 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 (which is fixed to $M = 5$ in our experiments). Notice that $\rho(A)$ is sparse and the eigen-decomposition of sparse matrix only need to be performed once. The matrix multiplication

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

introduced by the perturbation term can be solved efficiently in $O(knD)$ time as $\mathrm{rank}(\bar{U}) = k$. This computational cost of this multiplication can be ignored (with no increase in the complexity) as AX has $O(md)$ complexity (m is the number of edges). Overall, our optimization is several times slower than GCN with roughly the same number of free parameters and computational complexity. Because the sampling may cause additional variations of the gradient, we implement a stricter early stopping rule to avoid under-fitting.

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

7 Experiments

In this section, we perform an experimental study on semi-supervised transductive node classification tasks. We will use three popularly adopted benchmark datasets, namely, the Cora, CiteSeer and PubMed citation networks. The statistics of these datasets are displayed in table 1. As suggested recently [SMBG18], we use both the "canonical" split of training:validation:testing datasets by Planetoid [YCS16], as well as random splits of these datasets using the same ratio as in the "Train:Valid:Test" column.

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 methods in section 9 and refer the reader to [SMBG18] for the scores of 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 [YHC$^+$18]. We found that pre-processing the graph adjacency matrix A (with detailed steps listed in algorithm 1) can improve the performance of GCN on semisupervised node classification tasks. This processing is based on DeepWalk similarities [PARS14] that are explicitly formulated in Table 1 [QDM$^+$18]. The processing 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 set $T = 5$ and $\nu = 10^{-4}$. These procedures correspond to a polynomial filter with hand-crafted high order 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 the original A (after normalization) and the pre-processed A by algorithm 1.

The testing accuracy is reported in table 2. Our GCN (in the row "GCN") is adapted based on the codes by the original authors [KW17]. One can observe that FisherGCN and GCN^T can both improve over GCN, which means our perturbation and the pre-processing by algorithm 1 both help to improve generalization with additional computational cost. The best results are given by FisherGCN^T with both techniques added. One can also observe that FisherGCN (FisherGCN^T) presents a smaller variance than GCN (GCN^T), meaning the performance is more stable with our perturbations. On the random split datasets, the performance goes down as the "quality" of the training

Table 1: Dataset statistics. Note the number of edges reported in previous works [KW17] counts duplicate edges 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	#Edges	#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: Classification accuracy in percentage. The best average testing accuracy over a configuration grid is reported. The "Dataset" columns show the mean±std scores on 20 different initializations of the neural network weights based on the best configuration and the "canonical" split of training:validation:testing subsets used in the literature. "Dataset(R)" show the best mean±std scores on 100 runs over 10 random split of these datasets (10 different initializations per split), except FisherGCN^T on PubMed(R) that is computed based on 60 instead of 100 runs.

	Cora	CiteSeer	PubMed	Cora(R)	CiteSeer(R)	PubMed(R)
GCN [KW17]	81.5	70.3	79.0	–	–	–
GCN	81.90±0.8	71.08±0.7	79.10±0.3	79.27±2.2	70.08±1.9	78.20±2.2
FisherGCN	82.78±0.5	71.24±0.5	79.29±0.1	79.82±1.8	70.60±1.8	78.31±2.1
GCN^T	82.31±0.7	71.81±0.5	79.42±0.3	80.32±2.5	70.80±1.8	79.29±1.8
FisherGCN^T	**82.85±0.5**	**72.03±0.4**	**79.80±0.3**	**81.22±2.3**	**71.22±1.9**	**79.66±1.8**

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

$A \leftarrow \text{diag}^{-1}(A1)A$;
$S, B \leftarrow A$;
for $t \leftarrow 2$ ***to*** T **do**
  $B \leftarrow BA$;
  $S \leftarrow S + B$;
$A \leftarrow \frac{1}{T} S \circ (I - \text{diag}(1))$;
$A \leftarrow A \circ (A > \nu)$;
$A \leftarrow A + A^\top + I$;
$A \leftarrow \text{diag}^{-\frac{1}{2}}(A1)\ A\ \text{diag}^{-\frac{1}{2}}(A1)$;

set degrades. The observations are consistent.

8 Analysis

Based on [KW17], we express a graph convolution operation on an input signal $x = \sum_{i=1}^n \alpha_i u_i \in \Re^n$ as

$$\begin{aligned}[I - \text{tr}(L)\rho(A)]\, x &= x - \text{tr}(L) U\, \text{diag}(\lambda) U^\top x \\ &= x - \text{tr}(L) E_{i\sim\lambda}(\alpha_i u_i), \end{aligned} \quad (16)$$

where E denotes the expectation. The convolution term $\rho(A)x$ defined by the density operator $\rho(A)$ is an expectation of x's coordinates under the orthogonal frame defined by U.

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

$$\rho^\omega(A)x = \frac{1}{\lambda^\omega 1} U\, \text{diag}(\lambda)^\omega U^\top x = E_{i\sim \frac{\lambda^\omega}{\lambda^\omega 1}}(\alpha_i u_i). \quad (17)$$

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 largest eigenvalue of $\rho(A)$ without multiplicity). Therefore, high order convolutions will enhance the signal along the largest eigenvectors of ρ. Therefore our perturbation in the low order GCN [KW17] is equivalent to add high order polynomial filters. This also demonstrate the usefulness of the introduced formulations.

To make this study more systematical, we present an alternative intrinsic geometry of graphs (different from section 4) which is constructed in the spatial domain and is closely related to graph embeddings [PARS14]. 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. The assumption is these similarities are generated 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 $\text{KL}(W : P(Y))$ evaluated at the maximum likelihood estimation $Y^\star = \arg\min_Y \text{KL}(W : P(Y))$. 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

$$\hat{\mathcal{G}}(y^k) = 4L(W - P(Y)) + 8L(P(Y)D^k) - 4(B^k)^\top B^k.$$

where y_k is the k'th column of Y, $L(W - P(Y))$ is the Laplacian matrix computed based on the indefinite weights $(W - P(Y))$, $D^k = (y_{ik} - y_{jk})^2$, 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 [SMM14]. To understand this result, we ignore the first term because $P(Y) \to W$ as the number of observations increase. Then

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

is in the form of a variance of $(y_{ik} - y_{jk})(dy_{ik} - dy_{jk}) = d(y_{ik} - y_{jk})^2$ 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 the most variance in the neighbour's distance. For example, a rigid motion, or an uniform expansion/shrinking of the latent network embedding will cause little or no effect on the variance $d(y_{ik} - y_{jk})^2$, and hence corresponds to a small distance in this geometry.

This metric can be useful for develop theoretical perspectives of network embeddings, or build spatial perturbations of graphs, that is in contrast with our proposed spectral perturbation. We leave this theoretical result here for future investigations.

9 Conclusion and Discussion

We import quantum information geometry to define distance and projections on the manifold of graphs. We provide the Riemannian metric in closed form and adapted the notations of quantum information theory. The results and adaptations are novel and useful to develop new deep learning methods on graphs. We demonstrate their usage by perturb graph structures in a graph convolutional network, showing consistent improvements on transductive node classification experiments.

Our method can be generalized to a scalable setting, where a mini-batch only contains a sub-graph [HYL17] of $m \ll n$ nodes. This is because our perturbation has a low rank factorization given by the second term in eq. (10). One can reuse this spectrum factorization of the global matrix to build sub-graph perturbations.

If A has free-parameters [VCC+18], one can compute the low rank projection $\bar{\rho}^k(A)$ based on the original graph that is parameter free, or periodically save the graph and recompute $\bar{\rho}^k(A)$ during optimization.

References

[Ama16] S.-I. Amari. *Information Geometry and Its Applications*, volume 194 of *Applied Mathematical Sciences*. Springer-Verlag, Berlin, 2016.

[Ama19] S.-I. Amari. Fisher Information and Natural Gradient Learning in Random Deep Networks. In *AISTATS*, 2019. (to appear).

[BGS16] Samuel L. Braunstein, Sibasish Ghosh, and Simone 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.

[BJL18] Rajendra Bhatia, Tanvi Jain, and Yongdo Lim. On the bures–wasserstein distance between positive definite matrices. *Expositiones Mathematicae*, 2018.

[BNL11] Battista Biggio, Blaine Nelson, and Pavel Laskov. Support Vector Machines Under Adversarial Label Noise. In *ACML*, volume 20 of *PMLR*, pages 97–112. PMLR, 2011.

[Bur69] Donald Bures. An Extension of Kakutani's Theorem on Infinite Product Measures to the Tensor Product of Semifinite w^*-Algebras. *Transactions of the American Mathematical Society*, 135:199–212, 1969.

[BZSL14] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. In *ICLR*, 2014.

[Čen82] 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).

[CLZ17] Shui-Nee Chow, Wuchen Li, and Haomin Zhou. A Discrete Schrödinger Bridge Problem via Optimal Transport on Graphs. *CoRR*, abs/1705.07583, 2017.

[CZS18] Jianfei Chen, Jun Zhu, and Le Song. Stochastic Training of Graph Convolutional Networks with Variance Reduction. In *ICML*, volume 80 of *PMLR*, pages 942–950, 2018.

[DBV16] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *NIPS*, pages 3844–3852. Curran Associates, Inc., 2016.

[DMI+15] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alan Aspuru-Guzik, and Ryan P Adams. Convolutional Networks on Graphs for Learning Molecular Fingerprints. In *NIPS 28*, pages 2224–2232. Curran Associates, Inc., 2015.

[GL16] Aditya Grover and Jure Leskovec. Node2Vec: Scalable Feature Learning for Networks. In *KDD*, pages 855–864, 2016.

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *NIPS 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[H92] Matthias Hübner. Explicit computation of the Bures distance for density matrices. *Physics Letters A*, 163(4):239 – 242, 1992.

[HYL17] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. In *NIPS*, pages 1024–1034. Curran Associates, Inc., 2017.

[KW14] D. P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *ICLR*, 2014.

[KW17] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*, 2017.

[KXC11] Murat Kantarcıoğlu, Bowei Xi, and Chris Clifton. Classifier Evaluation and Attribute Selection Against Active Adversaries. *Data Mining and Knowledge Discovery*, 22(1):291–335, Jan 2011.

[MBM+17] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, and Michael M. Bronstein. Geometric Deep Learning on Graphs and Manifolds Using Mixture Model CNNs. In *CVPR*, pages 5425–5434, 2017.

[MC18] Boris Muzellec and Marco Cuturi. Generalizing point embeddings using the wasserstein space of elliptical distributions. In *NIPS*, pages 10237–10248. 2018.

[MFF] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. deepfool: A simple and accurate method to fool deep neural networks.

[MKHS14] Julien Mairal, Piotr Koniusz, Zaid Harchaoui, and Cordelia Schmid. Convolutional Kernel Networks. In *NIPS*, pages 2627–2635. Curran Associates, Inc., 2014.

[MMPidZ08] Damian Markham, Jarosław Adam Miszczak, Zbigniew Puchała, and Karol Życzkowski. Quantum State Discrimination: a Geometric Approach. *Phys. Rev. A*, 77, 2008.

[NB13] Frank Nielsen and Rajendra Bhatia. *Matrix Information Geometry*. Springer-Verlag Berlin Heidelberg, 2013.

[PARS14] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. DeepWalk: Online Learning of Social Representations. In *KDD*, pages 701–710, 2014.

[PB14] Razvan Pascanu and Yoshua Bengio. Revisiting Natural Gradient for Deep Networks. In *ICLR*, 2014.

[QDM+18] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18, pages 459–467, 2018.

[SKZ14] Alaa Saade, Florent Krzakala, and Lenka Zdeborová. Spectral Clustering of graphs with the Bethe Hessian. In *NIPS 27*, pages 406–414. Curran Associates, Inc., 2014.

[SMBG18] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Gnnemann. Pitfalls of Graph Neural Network Evaluation. In *NeurIPS Workshop on Relational Representation Learning*, 2018.

[SMM14] Ke Sun and Stphane Marchand-Maillet. An Information Geometry of Statistical Manifold Learning. In *ICML 31*, volume 32 of *PMLR*, pages 1–9, 2014.

[SN17] Ke Sun and Frank Nielsen. Relative Fisher Information and Natural Gradient for Learning Large Modular Models. In *ICML 34*, volume 70, pages 3289–3298. PMLR, 2017.

[TSE+19] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *ICLR*, 2019.

[VCC+18] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *ICLR*, 2018.

[VFH+19] Petar Velikovi, William Fedus, William L. Hamilton, Pietro Li, Yoshua Bengio, and R Devon Hjelm. Deep Graph Infomax. In *ICLR*, 2019.

[XBN+15] 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.

[XLT+18] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation Learning on Graphs with Jumping Knowledge Networks. In *ICML 35*, volume 80 of *PMLR*, pages 5453–5462, Stockholmsmssan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[YCS16] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *ICML*, pages 40–48, 2016.

[YHC+18] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *KDD*, pages 974–983. ACM, 2018.

[ZFY+19] Chenxiao Zhao, P. Thomas Fletcher, Mixue Yu, Yaxin Peng, Guixu Zhang, and Chaomin Shen. The Adversarial Attack and Detection under the Fisher Information Metric. In *AAAI*, 2019.

[ZSDG17] Guodong Zhang, Shengyang Sun, David K. Duvenaud, and Roger B. Grosse. Noisy Natural Gradient as Variational Inference. *CoRR*, abs/1712.02390, 2017.

Supplementary Material of "Fisher-Bures Adversary Graph Convolutional Networks"

A Proof of Theorem 1

By eq. (7), we have

$$ds^2 = \frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{n}\frac{(u_j^\top d\rho u_k)^2}{\lambda_j + \lambda_k}. \tag{18}$$

We also have

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

Because $\{u_i\}$ are orthonormal, we have

$$\left(u_j^\top \sum_{i=1}^{n}\left[d\lambda_i u_i u_i^\top + \lambda_i du_i u_i^\top + \lambda_i u_i du_i^\top\right] u_k\right) = d\lambda_j\delta_{jk} + \lambda_k u_j^\top du_k + \lambda_j du_j^\top u_k$$

Wrt the λ parameters, we have

$$ds^2 = \frac{1}{2}\sum_j \frac{(d\lambda_j)^2}{2\lambda_j} = \frac{1}{4}\sum_i \frac{1}{\lambda_i}d\lambda_i^2 \tag{19}$$

The first term on the rhs is proved. Now we consider the U parameters. On the unitary group, we have $\forall j, k$,

$$u_j^\top u_k = \text{constant}, \tag{20}$$

therefore

$$d(u_j^\top u_k) = du_j^\top u_k + u_j du_k^\top = 0. \tag{21}$$

Therefore

$$\begin{aligned}(\lambda_k u_j^\top du_k + \lambda_j du_j^\top u_k)^2 &= (\lambda_k u_j^\top du_k + \lambda_k du_j^\top u_k + (\lambda_j - \lambda_k)du_j^\top u_k)^2 \\ &= (\lambda_j - \lambda_k)^2 (du_j^\top u_k)^2.\end{aligned} \tag{22}$$

Plugging back into eq. (18), we get the Riemannian metric in the U coordinates. Notice that the cross terms $d\lambda_i du_i$ are ignored.

B Proof of corollary 2

The result is straightforward by plugging

$$d\lambda_i = \exp(\theta_i)d\theta_i \tag{23}$$

into the statement of theorem 1.

C Proof of corollary 3

We only need to prove the first part of corollary 3, that leads to the second part.

By theorem 1, we have

$$\begin{aligned}\mathrm{tr}(\mathcal{G}(u_i)) &= \frac{1}{2}\mathrm{tr}\left(\sum_{j=1}^{n}\left(\frac{(\lambda_i-\lambda_j)^2}{\lambda_i+\lambda_j}u_j u_j^\top\right)\right)\\ &= \frac{1}{2}\sum_{j=1}^{n}\mathrm{tr}\left(\frac{(\lambda_i-\lambda_j)^2}{\lambda_i+\lambda_j}u_j u_j^\top\right)\\ &= \frac{1}{2}\sum_{j=1}^{n}\left(\frac{(\lambda_i-\lambda_j)^2}{\lambda_i+\lambda_j}\mathrm{tr}(u_j u_j^\top)\right)\\ &= \frac{1}{2}\sum_{j=1}^{n}\frac{(\lambda_i-\lambda_j)^2}{\lambda_i+\lambda_j}. \end{aligned} \tag{24}$$

Because

$$\frac{|\lambda_i-\lambda_j|}{\lambda_i+\lambda_j} \leq 1, \tag{25}$$

we got a stronger result

$$\mathrm{tr}(\mathcal{G}(u_i)) \leq \frac{1}{2}\sum_{j=1}^{n}|\lambda_i-\lambda_j|. \tag{26}$$

Note that for density matrix the trace are normalized and we have $0 \leq \lambda_i \leq 1$, Therefore

$$\mathrm{tr}(\mathcal{G}(u_i)) \leq \frac{1}{2}\sum_{j=1}^{n}|\lambda_i-\lambda_j| \leq \frac{1}{2}\sum_{j=1}^{n}|0-\lambda_j| = \frac{1}{2}\sum_{j=1}^{n}\lambda_j = \frac{1}{2}. \tag{27}$$

D Proof of theorem 4

We first notice that D_B is invariant to unitary transformations: for any unitary U, we have

$$D_B(U\rho_1 U^\top, U\rho_2 U^\top) = D_B(\rho_1,\rho_2). \tag{28}$$

Therefore

$$D_B(\rho,\rho_0) = D_B(\rho, U\Lambda U^\top) = D_B(U^\top\rho U,\Lambda), = D_B(U^\top V R V^\top U,\Lambda) \tag{29}$$

where $\Lambda = \mathrm{diag}(\lambda)$, and $R = \mathrm{diag}(r_1,\cdots,r_n)$. By Theorem 3 [MMPidZ08], the optimal $V^\star = U$ so that the first density matrix on the rhs is diagonal, and the optimal R must have the same order as Λ. The problem reduces to

$$\min 2(1-\sum_i\sqrt{r_i\lambda_i}) \tag{30}$$

with respect to the constraints

$$\forall i, r_i \geq 0 \tag{31}$$

$$\sum_i r_i = 1 \tag{32}$$

$$r \text{ has } k \text{ non-zero entries} \tag{33}$$

The optimal $r^\star$ must be composed of the largest k eigenvalues of the given density matrix, i.e., λ_1, λ_k after re-scaling, that is,

$$\begin{cases} r_i &= \gamma\lambda_i \quad (\text{if } i=1,\cdots k)\\ r_i &= 0 \quad (\text{otherwise}) \end{cases} \tag{34}$$

We have

$$\sum_i r_i = \gamma\sum_i\lambda_i = 1. \tag{35}$$

Therefore $\gamma = 1/\sum_i\lambda_i$. Now we have both R and V and can express the optimal low-rank projection, which is given by theorem 4.

E Proof of theorem 5

By eq. (1), we have

$$dH^{l+1} = \Sigma \circ (\tilde{A} dH^l W^l) + \Sigma \circ (\tilde{A} H^l dW^l) + \Sigma \circ (d\tilde{A} H^l W^l). \tag{36}$$

and

$$\begin{aligned}
& d\ell \\
& = \mathrm{tr}(\frac{\partial \ell}{\partial H^{l+1}}^\top dH^{l+1}) \\
& = \mathrm{tr}\left(\frac{\partial \ell}{\partial H^{l+1}}^\top \left(\Sigma \circ (\tilde{A} dH^l W^l) + \Sigma \circ (\tilde{A} H^l dW^l) + \Sigma \circ (d\tilde{A} H^l W^l)\right)\right) \\
& = \mathrm{tr}\left((\frac{\partial \ell}{\partial H^{l+1}}^\top \circ \Sigma^\top)(\tilde{A} dH^l W^l)\right) + \mathrm{tr}\left((\frac{\partial \ell}{\partial H^{l+1}}^\top \circ \Sigma^\top)(d\tilde{A} H^l W^l)\right) \\
& + \mathrm{tr}\left((\frac{\partial \ell}{\partial H^{l+1}}^\top \circ \Sigma^\top)(\tilde{A} H^l dW^l)\right) \\
& = \mathrm{tr}\left(W^l (\frac{\partial \ell}{\partial H^{l+1}} \circ \Sigma)^\top \tilde{A} dH^l\right) + \mathrm{tr}\left((\frac{\partial \ell}{\partial H^{l+1}} \circ \Sigma)^\top \tilde{A} H^l dW^l\right) \\
& + \mathrm{tr}\left(H^l W^l (\frac{\partial \ell}{\partial H^{l+1}} \circ \Sigma)^\top d\tilde{A}\right)
\end{aligned} \tag{37}$$

Therefore

$$\begin{aligned}
\frac{\partial \ell}{\partial H^l} &= \tilde{A}\left(\frac{\partial \ell}{\partial H^{l+1}} \circ \Sigma\right) W^{l\top}; \\
\frac{\partial \ell}{\partial W^l} &= H^{l\top} \tilde{A} (\frac{\partial \ell}{\partial H^{l+1}} \circ \Sigma); \\
\frac{\partial \ell}{\partial \tilde{A}} &= H^l W^l (\frac{\partial \ell}{\partial H^{l+1}} \circ \Sigma).
\end{aligned} \tag{38}$$

Note only all layers contributes to the gradient w.r.t. $\tilde{A}$, and the above expression has to be corrected accordingly. Strictly speaking, this gradient has to be projected to be symmetric based on the constraint of the $\tilde{A}$ matrix.

The stated results are straightforward from the definition of the FIM (see [Ama16]) in eq. (4), and the above chain-rule equations.

F Experimental Settings

Our experiments are performed on Python 3.7 and Tensorflow 1.12 (GPU version).

For all investigated methods, we used the following configurations for the standard split of the citation datasets

- learning rate $\{0.01, 0.005\}$;
- Dropout rates $\{0.5, 0.7\}$;
- L^2 regularization strength $\{0.002, 0.001, 0.0005\}$;
- Number of layers 2;
- Hidden layer dimensionality 16;
- FisherGCN noise level $\epsilon \in \{0.001, 0.003, 0.01, 0.03\}$;
- #epochs: 500 (with early stopping);

We did not choose a dense configuration grid, due to limited computational capacity and to avoid overfiting the testing data. For the random split experiments, we fix the learning rates to 0.01 and dropout rates to 0.5, and remain the other configurations.

For GCN and GCN^T, we stop training if the validation error is greater than the average validation error of the last 10 epochs. For FisherGCN and FisherGCN^T, we stop training if the average validation error of the last 10 epochs becomes greater than then average over the last 100 epochs.

G Proof of theorem 6

We denote the KL divergence as $\mathcal{E}$.

$$\begin{aligned} d\mathcal{E} &= \sum_i \left[\sum_j w_{ij} dD_{ij} + \frac{1}{Z_i} \sum_j \exp(-D_{ij})(-dD_{ij}) \right] \\ &= \sum_i \sum_j \left(w_{ij} - p_{ij}(\boldsymbol{Y}) \right) dD_{ij}. \end{aligned} \tag{39}$$

As

$$dD_{ij} = d\|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2 = 2(\boldsymbol{y}_i - \boldsymbol{y}_j)^\top (d\boldsymbol{y}_i - d\boldsymbol{y}_j), \tag{40}$$

we have

$$\begin{aligned} d\mathcal{E} &= \sum_i \sum_j \left(w_{ij} - p_{ij}(\boldsymbol{Y}) \right) 2(\boldsymbol{y}_i - \boldsymbol{y}_j)^\top (d\boldsymbol{y}_i - d\boldsymbol{y}_j) \\ &= 2 \sum_i \sum_j \left(w_{ij} - p_{ij}(\boldsymbol{Y}) \right) (\boldsymbol{y}_i - \boldsymbol{y}_j)^\top (d\boldsymbol{y}_i - d\boldsymbol{y}_j) \\ &= 2\mathrm{tr} \left(d\boldsymbol{Y}^\top \mathrm{diag} \left((\boldsymbol{W} - \boldsymbol{P}(\boldsymbol{Y}))\mathbf{1} \right) \boldsymbol{Y} \right) \\ &+ 2\mathrm{tr} \left(d\boldsymbol{Y}^\top \mathrm{diag} \left((\boldsymbol{W}^\top - \boldsymbol{P}^\top(\boldsymbol{Y}))\mathbf{1} \right) \boldsymbol{Y} \right) \\ &- 2\mathrm{tr} \left(d\boldsymbol{Y}^\top \left(\boldsymbol{W} - \boldsymbol{P}(\boldsymbol{Y}) \right) \boldsymbol{Y} \right) \\ &- 2\mathrm{tr} \left(d\boldsymbol{Y}^\top \left(\boldsymbol{W}^\top - \boldsymbol{P}^\top(\boldsymbol{Y}) \right) \boldsymbol{Y} \right). \end{aligned} \tag{41}$$

Therefore

$$\begin{aligned} \frac{\partial \mathcal{E}}{\partial \boldsymbol{Y}} =& 2 \mathrm{diag} \left((\boldsymbol{W} - \boldsymbol{P}(\boldsymbol{Y}))\mathbf{1} \right) \boldsymbol{Y} \\ &+ 2 \mathrm{diag} \left((\boldsymbol{W}^\top - \boldsymbol{P}^\top(\boldsymbol{Y}))\mathbf{1} \right) \boldsymbol{Y} \\ &+ 2 \left(\boldsymbol{W} - \boldsymbol{P}(\boldsymbol{Y}) \right) \boldsymbol{Y} \\ &+ 2 \left(\boldsymbol{W}^\top - \boldsymbol{P}^\top(\boldsymbol{Y}) \right) \boldsymbol{Y} \\ =& 2\boldsymbol{L}(\boldsymbol{W} - \boldsymbol{P}(\boldsymbol{Y}))\boldsymbol{Y}, \end{aligned} \tag{42}$$

where $\boldsymbol{L}(\boldsymbol{W} - \boldsymbol{P}(\boldsymbol{Y}))$ is the Laplacian matrix with the indefinite weights $\boldsymbol{W} - \boldsymbol{P}(\boldsymbol{Y})$.

By eq. (39),

$$d^2\mathcal{E} = \sum_i \sum_j (w_{ij} - p_{ij}(\boldsymbol{Y})) d^2 D_{ij} - \sum_i \sum_j dp_{ij}(\boldsymbol{Y}) dD_{ij}. \tag{43}$$

By noticing

$$\begin{aligned} d^2 D_{ij} &= 2 \left(d\boldsymbol{y}_i - d\boldsymbol{y}_j \right)^\top \left(d\boldsymbol{y}_i - d\boldsymbol{y}_j \right) \\ &= 2 \sum_l \left(dy_{il} - dy_{jl} \right)^2. \end{aligned} \tag{44}$$

The first term on the rhs turns out to be

$$\begin{aligned} d^2\mathcal{E}_1 =& \sum_i \sum_j (w_{ij} - p_{ij}(\boldsymbol{Y}))2 \left(d\boldsymbol{y}_i - d\boldsymbol{y}_j\right)^\top \left(d\boldsymbol{y}_i - d\boldsymbol{y}_j\right) \\ =& 2 \sum_i \sum_j \sum_l (w_{ij} - p_{ij}(\boldsymbol{Y})) \left(dy_{il} - dy_{jl}\right)^2 . \end{aligned} \tag{45}$$

Therefore,

$$\begin{aligned} \frac{\partial^2 \mathcal{E}_1}{\partial y_{il} \partial y_{jl}} &= \begin{cases} -4(w_{ij} - p_{ij}(\boldsymbol{Y})) - 4(w_{ji} - p_{ji}(\boldsymbol{Y})) & \text{if } i \neq j; \\ 4 \sum_i (w_{ij} - p_{ij}(\boldsymbol{Y})) & \text{if } i = j. \end{cases} \\ \frac{\partial^2 \mathcal{E}_1}{\partial \boldsymbol{y}_l \partial \boldsymbol{y}_l} &= 4\boldsymbol{L}(\boldsymbol{W} - \boldsymbol{P}(\boldsymbol{Y})). \end{aligned} \tag{46}$$

The second term yields

$$\begin{aligned} d^2\mathcal{E}_2 =& - \sum_i \sum_j dp_{ij}(\boldsymbol{Y}) dD_{ij} \\ =& - \sum_i \sum_j d\left(\frac{1}{Z_i} \exp(-D_{ij})\right) dD_{ij} \\ =& \sum_i \sum_j \frac{1}{Z_i} \exp(-D_{ij}) \left(dD_{ij}\right)^2 \\ & - \sum_i \sum_j \frac{1}{Z_i^2} \exp(-D_{ij}) dD_{ij} \sum_j \exp(-D_{ij}) dD_{ij} \\ =& \sum_i \left[\sum_j p_{ij}(\boldsymbol{Y}) \left(dD_{ij}\right)^2 - \left(\sum_j p_{ij}(\boldsymbol{Y}) dD_{ij} \right)^2 \right]. \end{aligned} \tag{47}$$

We have

$$\begin{aligned} dD_{ij} &= 2(\boldsymbol{y}_i - \boldsymbol{y}_j)^\top (d\boldsymbol{y}_i - d\boldsymbol{y}_j) \\ (dD_{ij})^2 &= 4 \sum_k \sum_l (y_{ik} - y_{jk})(y_{il} - y_{jl})(dy_{ik} - dy_{jk})(dy_{il} - dy_{jl}). \end{aligned} \tag{48}$$

and therefore

$$\begin{aligned} & - \sum_i \sum_j dp_{ij}(\boldsymbol{Y}) dD_{ij} \\ =& 4 \sum_i \sum_j \sum_k \sum_l p_{ij}(\boldsymbol{Y})(y_{ik} - y_{jk})(y_{il} - y_{jl})(dy_{ik} - dy_{jk})(dy_{il} - dy_{jl}) \\ & - 4 \sum_i \left(\sum_j p_{ij}(\boldsymbol{Y}) \sum_k (y_{ik} - y_{jk})(dy_{ik} - dy_{jk}) \right)^2 \\ =& 4 \sum_i \sum_j \sum_k p_{ij}(\boldsymbol{Y})(y_{ik} - y_{jk})^2 (dy_{ik} - dy_{jk})^2 \\ & - 4 \sum_i \sum_k \left(\sum_j p_{ij}(\boldsymbol{Y})(y_{ik} - y_{jk})(dy_{ik} - dy_{jk}) \right)^2 \text{ (ignoring all terms with } k \neq l.) \end{aligned} \tag{49}$$

For the first term, we have

$$\frac{\partial^2 \mathcal{E}_{21}}{\partial y_{ik} \partial y_{jk}} = \begin{cases} -8p_{ij}(\boldsymbol{Y})(y_{ik} - y_{jk})^2 - 8p_{ji}(\boldsymbol{Y})(y_{ik} - y_{jk})^2 & \text{if } i \neq j \\ 8\sum_i p_{ij}(\boldsymbol{Y})(y_{ik} - y_{jk})^2 + 8\sum_j p_{ij}(\boldsymbol{Y})(y_{ik} - y_{jk})^2 & \text{otherwise} \end{cases}$$

$$\frac{\partial^2 \mathcal{E}_{21}}{\partial \boldsymbol{y}_k \partial \boldsymbol{y}_k} = 8\boldsymbol{L}(\boldsymbol{P}\boldsymbol{D}^k). \tag{50}$$

where $D_{ij}^k = (y_{ik} - y_{jk})^2$ means the pair-wise distance along the k'th dimension.

For the second term, we have

$$\frac{\partial^2 \mathcal{E}_{22}}{\partial \boldsymbol{y}_k \partial \boldsymbol{y}_k^\top} = -4(\boldsymbol{B}^k)^\top (\boldsymbol{B}^k), \tag{51}$$

where

$$b_{ij}^k = \begin{cases} \sum_j p_{ij}(\boldsymbol{Y})(y_{ik} - y_{jk}) & \text{if } i = j \\ -p_{ij}(\boldsymbol{Y})(y_{ik} - y_{jk}) & \text{otherwise} \end{cases} \tag{52}$$

Putting everything together, we get

$$\frac{\partial^2 \mathcal{E}}{\partial \boldsymbol{y}_k \partial \boldsymbol{y}_k^\top} = 4\boldsymbol{L}(\boldsymbol{W} - \boldsymbol{P}(\boldsymbol{Y})) + 8\boldsymbol{L}(\boldsymbol{P}(\boldsymbol{Y})\boldsymbol{D}^k) - 4(\boldsymbol{B}^k)^\top \boldsymbol{B}^k. \tag{53}$$