Gradient Scarcity with Bilevel Optimization for Graph Learning

Hashem GHANEM[1] Samuel VAITER[2] Nicolas KERIVEN[3]

[1]Institut de Mathematiques de Bourgogne, 9 avenue Alain Savary BP 47870, 21078 Dijon Cedex, France

[2]Laboratoire Mathématiques & Interactions J.A. Dieudonné, Parc Valrose, 28 Avenue Valrose, 06108 Nice Cedex 02, France

[3]Institut de Recherche en Informatique et Systèmes Aléatoires, 263 avenue du Général Leclerc, 35042 RENNES cedex, France

Résumé – Dans le cadre de l'apprentissage semi-supervisé, la minimisation d'une function de coût pour apprendre *à la fois* le graphe de similarité entre les données et le classificateur peut conduire à un phénomène de *rareté* du gradient. Il s'agit d'arêtes éloignées des nœuds étiquetés qui reçoivent des gradients nuls. Initialement, le problème a été observé dans un contexte d'optimisation *jointe* et de *Graph Neural Networks* (GNN). Dans ce papier, nous démontrons que le problème se pose également pour l'optimisation *biniveau*, malgré une dépendance supplémentaire entre les paramètres du problème. Alors que la rareté du gradient avec les GNN résulte de leur champ réceptif fini, nous montrons qu'elle est également présente pour modèle de régularisation Laplacien, qui a pourtant un champ réceptif infini, avec des gradients diminuant exponentiellement avec la distance aux nœuds étiquetés. Pour résoudre ce problème, nous étudions plusieurs stratégies : utiliser un modèle de graphe latent, un procédé de régularisation ou un graphe de diamètre réduit. Nos expériences illustrent notre analyse et valident les solutions proposées.

Abstract – Under the semi-supervised learning setting, minimizing a classification loss while learning both the graph and the downstream classifier can lead to *gradient scarcity*. It refers to the fact that edges distant from labelled nodes receive zero gradients. Initially, the problem was observed while *jointly* optimizing the graph and Graph Neural Network (GNN) classifiers. This study demonstrates that the problem also arises in the *bilevel* optimization scheme, where additional dependency exists between the parameters of the problem. While gradient scarcity with GNNs is a result of their finite receptive field, we show that it is also present with the Laplacian regularization model, which has an infinite receptive field, with gradients diminishing exponentially with distance to labeled nodes. To address this issue, we suggest using latent graph learning, graph regularization, or optimizing on a graph with a reduced diameter. Our experiments support our analysis and validate the proposed solutions.

1 Introduction

Graphs model relations between points. In Semi-Supervised Learning (SSL) tasks, where only a small portion of points are labelled, graphs provide essential knowledge as linked points are likely to share the same label. Formally, a graph $\mathcal{G}$ is a pair (V, E), where V is a set of n nodes and $E \subseteq V \times V$ is a set of edges. We represent a graph by its adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, where $\boldsymbol{A}_{i,j}$ is the edge weight between i, j. We here look at transductive SSL problems. Given $(\boldsymbol{X},\ \boldsymbol{A}_{obs},\ \boldsymbol{Y}_{obs})$, where $\boldsymbol{X} \in \mathbb{R}^{n\times p}$ is the feature matrix whose rows include the features of corresponding nodes, $\boldsymbol{A}_{obs}$ is the observed graph, and $\boldsymbol{Y}_{obs} \in \mathbb{R}^n$ contains the labels of a subset of points at coordinates $i \in V_{tr} \subset V$ (its value outside V_{tr} does not matter), the goal is to predict labels on unlabelled points. There are roughly two main strategies to solve SSL problems. The first is to *propagate* known labels using a *regularization* process. A popular choice is the *Laplacian regularization* model, where predicted labels, while emphasizing the dependence on the adjacency matrix $\boldsymbol{A}$, write:

$$\boldsymbol{Y}_{\text{Reg}}(\boldsymbol{A}) \in \underset{\boldsymbol{Y}}{\arg\min}\ \tfrac{1}{|V_{tr}|}\sum_{i\in V_{tr}} \ell(\boldsymbol{Y}_i, (\boldsymbol{Y}_{obs})_i) + \tfrac{\lambda}{|E|}\boldsymbol{Y}^\top \boldsymbol{L}\boldsymbol{Y}, \tag{1}$$

where ℓ is a smooth loss function commonly chosen to be the Categorical Cross Entropy (CCE) loss for classification, and the Mean Square Error (MSE) for regression, $\boldsymbol{L} = \boldsymbol{L}(\boldsymbol{A}) = \boldsymbol{D} - \boldsymbol{A}$ is the graph Laplacian, and λ is a balancing parameter.

The second strategy is to train a *parametric model* $\boldsymbol{Y}_W(\boldsymbol{X}, \boldsymbol{A})$ parameterized by the weights W, such as Graph Neural Networks (GNNs):

$$\boldsymbol{Y}_{GNN}(\boldsymbol{A}) = \boldsymbol{Y}_{W^\star}(\boldsymbol{X}, \boldsymbol{A}),\ \text{where}$$
$$W^\star = \underset{W}{\arg\min}\ \frac{1}{|V_{tr}|}\sum_{i\in V_{tr}} \ell\big(\big(\boldsymbol{Y}_W(\boldsymbol{X}, \boldsymbol{A})\big)_i, (\boldsymbol{Y}_{obs})_i\big). \tag{2}$$

We here adopt classical message-passing GNNs with sum aggregation, where the output of the $(l+1)$-th layer equals

$$\boldsymbol{X}^{[l+1]} = \phi(\boldsymbol{X}^{[l]}\boldsymbol{W}_1^{[l+1]} + \boldsymbol{A}\boldsymbol{X}^{[l]}\boldsymbol{W}_2^{[l+1]} + \boldsymbol{1}_n(\boldsymbol{b}^{[l+1]})^\top),$$

where $\boldsymbol{W}_1^{[l+1]}, \boldsymbol{W}_2^{[l+1]} \in \mathbb{R}^{d_l \times d_{l+1}}, \boldsymbol{b}^{[l+1]} \in \mathbb{R}^{d_{l+1}}$ are learnable weights, d_l is the output dimensionality of the l-th layer, $\boldsymbol{1}_n = [1, \dots, 1]^\top \in \mathbb{R}^n$, and ϕ is a non-linear function. $\boldsymbol{Y}_W(\boldsymbol{X}, \boldsymbol{A}) = \boldsymbol{X}^{[k]}$ is the output after k rounds of message passing, and W is gathered as $W = \{\boldsymbol{W}_1^{[l]}, \boldsymbol{W}_2^{[l]}, \boldsymbol{b}^{[l]}\}_{l=1}^k$.

Unfortunately, real-world graphs are noisy, which significantly degrades the performance in SSL tasks. A mainstream approach in graph learning to overcome this issue seeks a graph that, when used by the classifier (either $\boldsymbol{Y}_{\text{Reg}}$ or $\boldsymbol{Y}_{GNN}$), minimizes the labelling loss on labeled nodes. However, the classifier itself requires an optimization process on its parameters. Two mathematical formulations are possible for this problem, which both are usually solved by gradient-based algorithms. In joint optimization, a *single* loss function F is used to optimize for both objects. For example, in the GNN scenario we solve: $\min_{\boldsymbol{A}\in\mathcal{A}, W} \sum_{i\in V_{tr}} \ell\big(\big(\boldsymbol{Y}_W(\boldsymbol{X}, \boldsymbol{A})\big)_i, (\boldsymbol{Y}_{obs})_i\big)$, where $\mathcal{A}$ is a set of admissible adjacency matrices. $\boldsymbol{A}, W$ in the joint setting are simultaneously updated at each iteration and considered *independent* of each other, *i.e.,* $\frac{\partial W}{\partial \boldsymbol{A}} = \boldsymbol{0}$. The formulation

of interest here, called *bilevel optimization*, associates a loss function F_{out} to the graph that is a function of the trained classifier. Assuming that there is another set of labelled nodes $V_{out} \subset V$ *distinct* from V_{tr}, we seek any

$$\widehat{\boldsymbol{A}} \in \arg\min_{\boldsymbol{A}\in\mathcal{A}} F_{out}(\boldsymbol{A}) = \frac{1}{|V_{out}|} \sum_{i\in V_{out}} \ell(\boldsymbol{Y}(\boldsymbol{A})_i, \boldsymbol{Y}_i) \ , \quad (3)$$

such that $\boldsymbol{Y}(\boldsymbol{A}) = \boldsymbol{Y}_{GNN}(\boldsymbol{A})$ or $\boldsymbol{Y}(\boldsymbol{A}) = \boldsymbol{Y}_{\text{Reg}}(\boldsymbol{A})$. That is, the minimization of the objective function F_{out}, called the *outer* problem, involves $\boldsymbol{Y}(\boldsymbol{A})$, which is itself the result of an optimization problem called the *inner* problem. We refer to the outer gradient ∇F_{out} as *hypergradient*. A common model for $\mathcal{A}$ is edge refinement $\mathcal{A} = \{\boldsymbol{A} \in \mathbb{R}^{n\times n} | \boldsymbol{A}_{ij} = 0 \text{ when } (\boldsymbol{A}_{obs})_{ij} = 0\}$. We will see that other models are capable of resolving the problem of interest in this work, which emerges in edge refinement tasks: *gradient scarcity*.

Remark: min-min problems are an instance of bilevel problems which are usually easier to analyze. They take place when the inner and the outer objectives are identical, *e.g.,* when $\boldsymbol{Y}(\boldsymbol{A}) = \boldsymbol{Y}_{GNN}(\boldsymbol{A})$ and $V_{tr} = V_{out}$. Therefore, the analysis in this paper encompasses the min-min setting as it does not necessitate V_{tr}, V_{out} being disjoint sets. Recall that we consider this assumption to achieve a better generalization.

Unlike bilevel optimization, joint optimization lead to the trivial solution $\boldsymbol{A} = \boldsymbol{0}$ when adopting the Laplacian regularization. Moreover, joint optimization promotes overfitting with GNNs, as GNNs usually have the capacity to fit the small subset of labelled data V_{tr} with an arbitrary graph. By introducing a distinct training set for the graph in the outer problem, which can be seen as a validation set, bilevel optimization leads to a graph that generalizes better on test nodes.

The gradient scarcity phenomenon [2] was observed solving the joint optimization problem using a gradient-based algorithm with the classifier being a GNN. Indeed, a k-layer GNN computes the label of a node using information from nodes up to k-hop away, hence due to this limited receptive field, the gradient of this label becomes zero when calculated with respect to edges that connect nodes outside of this range. However, it is not clear how to extend this argument to the bilevel optimization setting where the weights of the GNN themselves depend on the graph, *i.e.,* $\frac{\partial W}{\partial \boldsymbol{A}} \neq \boldsymbol{0}$. It is also not clear whether using graph-based models with an infinite receptive field, such as the Laplacian regularization, can solve this problem.

Contribution: we prove that by solving edge refinement tasks with bilevel optimization while using a k-layer GNN classifier, the hypergradient on edges between nodes at least k-hop from nodes in $V_{tr} \cup V_{out}$ are null. The distance to a subset of nodes is defined as the shortest distance to any of its member nodes. Similarly, we prove that hypergradient scarcity *persists in the Laplacian regularization case*, with hypergradients being exponentially damped as distance from labeled nodes increases. Our empirical validation supports these results. To address this issue, we explore three possible strategies: latent graph learning, graph regularization, and refining a power of the given adjacency matrix.

2 Related work

The method in [3] learns graphs by optimizing the parameters of Bernoulli probability distributions over random edges using bilevel optimization; however, it learns n^2 parameters which limits scalability. Other works adopted *joint* optimization, as in [14] or in attention mechanisms [12], where edge weights are refined after each GNN layer based on similarity between node representations. The method in [13], called GAM, learns graphs by penalizing edge absence between nodes with the same label, hence GAM does not adopt the bilevel nor the joint setting.

Gradient scarcity was detected in [2] where for tasks adopting a 2-layer GNN classifier, the authors show that edges between unlabelled nodes do not receive supervision if they are at least 2-hop from labelled nodes. This issue cannot be resolved by adding more GNN layers due to the oversmoothing issue [8]. To provide more supervision on the graph level, authors regularize the graph using a penalty term that optimizes its performance in denoising node features. That said, this study assumed no dependence between the GNN weights and the graph as it considered joint optimization. Gradient scarcity has not been studied under the bilevel optimization setting, and when considering other graph-based models.

In [10], authors state that optimizing the graph and a GNN model under the supervision of a classification task introduces reliance on available labels, and bias in the edge distribution. This, however, is not theoretically justified. To overcome this problem, authors proposed an *unsupervised* graph learning framework. Although the proposed framework competed state-of-the-art methods, we believe that labels contain informative knowledge that is not exploited by unsupervised learners.

3 Hypergradient scarcity with GNNs

We consider bilevel optimization (3) for *edge refinement*, adopting the GNN classifier $\boldsymbol{Y}(\boldsymbol{A}) = \boldsymbol{Y}_{GNN}(\boldsymbol{A})$.

Theorem 3.1. *Let $\boldsymbol{Y}_W$ be a k-layer GNN parametrized by the set of weights W. Assume that the inner optimization problem is solved with a gradient-based algorithm. Then, for any pair of nodes i, j at least k-hop from nodes in $V_{out} \cup V_{tr}$, we have $\frac{\partial F_{out}}{\partial \boldsymbol{A}_{i,j}} = \boldsymbol{0}$.*

Sketch of Proof. By induction on gradient-based updates on W, we prove that the weights W_t at any iteration t are not a function of edges connecting nodes at least k-hop from nodes in V_{tr}. We then prove for all u in V_{out}, given the previous result and the GNN's finite receptive field, that $(\boldsymbol{Y}_{W_t})_u$ is not a function of $\boldsymbol{A}_{i,j}$, which completes the proof as with the chain rule one gets $\frac{\partial F_{out}}{\partial \boldsymbol{A}_{i,j}} = \boldsymbol{0}$. Full proof is available in [4]. □

We point out that the proof does not assume that the gradient-based inner optimizer converges to a minimizer of the inner problem. Thus, hypergradient scarcity occurs even when this optimizer converges to a local minimum, which is the case in practice as the inner problem is not convex.

4 Hypergradient scarcity with the Laplacian regularization

Since the finite receptive field of GNNs leads to gradient scarcity, we now investigate hypergradient scarcity in the case where $\boldsymbol{Y}(\boldsymbol{A}) = \boldsymbol{Y}_{\text{Reg}}(\boldsymbol{A})$ with the Laplacian regularization.

In this scenario, the inner problem (1) does not induce a finite receptive field because in general $\frac{\partial \boldsymbol{Y}(\boldsymbol{A})}{\partial \boldsymbol{A}ij} \neq 0$ for all i, j.

We show that although to a lesser degree, this issue still arises. Specifically, we establish that the hypergradient's magnitude diminishes *exponentially* as the sum of the two distances to V_{tr} and V_{out} increases. Our study is focused on regression tasks, where ℓ is the MSE loss function in Eqs. (1) and (3). Let $\boldsymbol{S}_{in} \in \mathbb{R}^{n \times n}$ be the diagonal matrix with entries equal to 1 for nodes in V_{tr} and 0 otherwise, the solution $\boldsymbol{Y}(\boldsymbol{A})$ enjoys a closed-form expression:

$$\boldsymbol{Y}(\boldsymbol{A}) = \boldsymbol{B}^{-1} \tilde{\boldsymbol{S}}_{in} \boldsymbol{Y}_{obs} \ ,$$

where $\tilde{\boldsymbol{S}}_{in} = \frac{\boldsymbol{S}_{in}}{|V_{tr}|}$ and $\boldsymbol{B} = \frac{\boldsymbol{S}_{in}}{|V_{tr}|} + \lambda \frac{\boldsymbol{L}}{|E|}$. Given that, we now state the main result of this section.

Theorem 4.1. *Let nodes i, j be at least k-hop from V_{out}, and q-hop from V_{tr}. Then we have:*

$$\left| \frac{\partial F_{out}}{\partial \boldsymbol{A}_{ij}} \right| \lesssim \lambda \frac{\sqrt{|V_{out}|} + \mu_{\min} \sqrt{|V_{tr}||V_{out}|}}{\mu_{\min}^3 |V_{tr}||E|} y_\infty^2 (1-\mu)^{q+k} \ ,$$

s.t. $\mu_{\min}$ ($\mu_{\max}$) is the smallest (largest) eigenvalue of $\boldsymbol{B}$ satisfying $0 < \mu_{\min} < \mu_{\max}$, $\mu = \frac{\mu_{\min}}{\mu_{\max}}$, and $y_\infty = \|\boldsymbol{Y}_{obs}\|_\infty$.

Sketch of Proof. We express $\boldsymbol{Y}(\boldsymbol{A})$ as a Neumann series, then bound the derivative of terms in the resulted series, and by extension the gradient of F_{out}. Full proof is available in [4]. □

Since $0 < 1 - \mu < 1$, Theorem 4.1 states that the hypergradient is exponentially damped as $q + k$ increases.

5 Alleviating hypergradient scarcity

We review 3 strategies to mitigate hypergradient scarcity:

— *Generalized edge refinement*: refining a power of $\boldsymbol{A}_{obs}$ as $\boldsymbol{A}_{obs}^r$ yields a potential edge between neighbors less than r-hop from each other in $\boldsymbol{A}_{obs}$. Hence, the distance between unlabelled and labelled nodes is reduced.

— *Graph regularization* provides another source of hypergradients than the classification loss. Here, we adopt the penalty $-\gamma \mathbf{1}_n^\top \log \boldsymbol{A} \mathbf{1}_n$ proposed in [7], where γ is the trade-off hyper-parameter, which penalizes graphs with low node degrees and achieves state-of-the-art results on graph learning from smooth signals.

— *Latent graph learning*: training a model f_θ parameterized by the weights θ on edge refinement, *i.e.,* $\mathcal{A} = \{\boldsymbol{A}_\theta = f_\theta(\boldsymbol{A}_{obs}, \boldsymbol{X})\}$. f_θ takes as input node features and the observed graph: we refer to such models as *Graph-to-Graph* (G2G). Note that G2G outputs weights only on observed edges. The adopted $G2G$ model is:

$$(\boldsymbol{A}_\theta)_{i,j} = \alpha\big((\boldsymbol{X}_i - \boldsymbol{X}_j)^2\big) \ , \qquad (4)$$

where the square function is applied entrywise, $\alpha : \mathbb{R}^p \to \mathbb{R}$ is a Multi-Layer Perceptron (MLP) model of k_{G2G} layers of the form: $\boldsymbol{X}^{[l+1]} = \phi^{[l+1]}(\boldsymbol{X}^{[l]} \boldsymbol{W}_1^{[l+1]} + \mathbf{1}_n (\boldsymbol{b}^{[l+1]})^\top)$, where $\boldsymbol{W}_1^{[l+1]} \in \mathbb{R}^{d_l \times d_{l+1}}, \boldsymbol{b}^{[l+1]} \in \mathbb{R}^{d_{l+1}}$ are learnable parameters, and d_l is the output dimensionality of the l-th layer. θ is gathered as $\theta = \{\boldsymbol{W}_1^{[l]}, \boldsymbol{b}^{[l]}\}_{l=1}^{k_{G2G}}$.

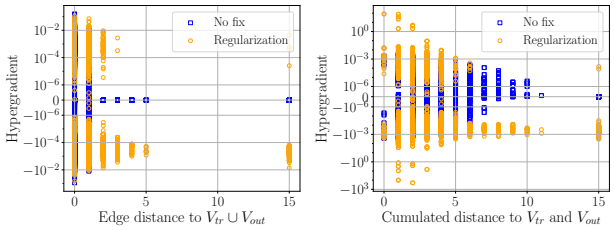

Figure 1 – Observing hypergradient scarcity and the effect of graph regularization on Cora. Left: adopting the GNN classifier. Right: adopting the Laplacian regularization model. We plot the hypergradient against edge distance. When this distance is not defined, *e.g.,* in connected components without a node from $V_{tr} \cup V_{out}$, we assign the distance 15 to edges in such components in order to be able to visualize them.

6 Experiments

We validate our findings on Cora [11], which is a common SSL benchmark dataset for node classification. Cora consists of one graph, where nodes correspond to publications represented by a bag of words, and edges represent citations. The objective is to categorize the given articles based on their topic. Further experiments on different datasets can be found in [4].

Models: the function α in the $G2G$ model is an MLP with 2 hidden layers, each having 32 neurons and followed by the $ReLu$ activation function. The GNN we consider has 1 hidden layer of 128 neurons that is followed by $ReLu$, while the output layer is followed by the $softmax$ function.

Setup: we unroll [6] τ_{in} iterations of the inner optimizer, and deploy automatic differentiation [1], implemented in the `Higher` package [5], to compute hypergradients. For the inner and the outer solvers, we consider the Adaptive Moment estimation algorithm (Adam) [9]. We set with a grid search $\eta_{in} = 10^{-2}$ with the GNN, $\eta_{in} = 10^{-1}$ with the Laplacian regularization, $\eta_{out} = 10^{-2}$ in all experiments without a $G2G$ model, otherwise $\eta_{out} = 10^{-4}$ with the GNN classifier, and $\eta_{out} = 10^{-3}$ with the Laplacian regularization. Also $\tau_{in} = 500$ for the Laplacian regularization, and 100 with the GNN classifier. We set the number of outer iterations to 150 and select the model with the highest validation accuracy. We set $\lambda = 1$ in training as we expect the bilevel algorithm to learn λ by scaling the learned adjacency matrix. When applying the Laplacian regularization fed with $\boldsymbol{A}_{obs}$, we set $\lambda = 0.1$ after a grid search. γ in the graph regularization term is tuned to 1.

6.1 Results

The task at hand is a multi-label classification problem with ℓ set to the CCE function. In Fig. 1, we show the hypergradient received on edges at outer iteration 9 as a function of their distance to labelled nodes. For the Laplacian regularization case, this is the edge cumulative distance to V_{tr} and V_{out} defined as follows: we compute $q + k$, the sum of distances to V_{tr} and V_{out}, respectively, for its both endpoint nodes, then we take the minimum of the two results. Whereas in the GNN case, we compute for each of its endpoint nodes its distance to $V_{tr} \cup V_{out}$, then we take the minimum. The figure showcases our analysis, and shows that the GNN classifier manifests null hypergradients for distances exceeding 2, while the Laplacian regularization scenario displays a hypergradient

Table 1 – Accuracies obtained on Cora when the classifier is trained using the output graph of the Bilevel Optimization (BO) framework, BO equipped with graph regularization, BO equipped with the G2G model. We also benchmark against GAM [13] (the result is reported from the according paper). For each classifier, we report test accuracy in the according first line and training accuracy on V_{out} in the second one. Training accuracy on V_{tr} equals 100% for all methods.

Graph	$\boldsymbol{A}_{obs}$	BO	BO+Reg	BO+G2G	GAM
GNN	77.0	76.2	80.3	82.0	84.8
	77.4	94.9	94.1	97.4	-
Laplacian	71.7	76.2	78.3	76.2	-
	71.0	81.9	83.2	83.5	-

that diminishes exponentially as the distance increases.

Regarding the generalization capacity, Table 1 shows that the learned graph exhibits lower generalization capacity in the GNN case compared to $\boldsymbol{A}_{obs}$. This suggests that hypergradient scarcity leads to overfitting, which is expected given the extreme scarcity in the GNN scenario. Specifically, edges with distances exceeding 2 retain their random initialization after training. This is, however, not the case in the Laplacian regularization scenario where most edges have distances less than 11 thereby they do not display severely damped hypergradients, thus not affecting generalization.

We now evaluate the effectiveness of the proposed solutions for mitigating hypergradient scarcity. Due to memory constraints, we do not attempt to learn a power of $\boldsymbol{A}_{obs}$; however, we applied this approach to other datasets in [4]. As demonstrated in Fig. 1, graph regularization is a viable solution, producing non-zero hypergradients on all edges with magnitudes comparable to those of edges with small distances. Note that hypergradients are received on the G2G weights when it is used, not on edges, hence they are not depicted in the figure. With respect to generalization, Table 1 illustrates that both fixes result in significant improvements in test accuracy compared to $\boldsymbol{A}_{obs}$ when used with the GNN classifier. In the Laplacian regularization scenario, graph regularization results in higher test accuracy, whereas the G2G model exhibits similar generalization performance to learning edge weights directly. Moreover, we observe that the GNN classifier consistently outperforms the Laplacian regularization, particularly when using the G2G model and graph regularization, as well as when operating directly on $\boldsymbol{A}_{obs}$. This is expected since the Laplacian regularization promotes similarity between connected nodes, but it is not a supervised technique like GNNs, which is why it generally yields lower performance.

We finally point out that the bilevel optimization framework with either fix does not achieve state-of-the-art results produced by GAM. This suggests that the label agreement model produces graphs that generalize better on citation networks.

7 Conclusion

We examined the hypergradient scarcity issue when utilizing bilevel optimization to solve edge refinement tasks under the SSL setting. This problem occurs when optimizing both the graph and the classifier to improve the classification performance, where edges that are far from labeled nodes receive scarce hypergradients. We showed that using GNNs leads to this phenomenon. We also investigated hypergradient scarcity with the Laplacian regularization model, which has an infinite receptive field. While the phenomenon still occurs, it is less severe, and we proved that the magnitude of hypergradients is exponentially damped with distance to labeled nodes. To mitigate hypergradient scarcity, we proposed three solutions: latent graph learning, graph regularization, and refining the edges in a power of the observed adjacency matrix. Our experiments confirmed our findings and proved the effectiveness of the first two solutions. Future works include examining other solutions, such as spectral graph convolutional networks while setting its filter width large to have a larger receptive field.

References

[1] Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *J. Mach. Learn. Res.*, 18, 2018.

[2] Bahare Fatemi, Layla El Asri, and Seyed Mehran Kazemi. Slaps: Self-supervision improves structure learning for graph neural networks. *NeurIPS*, 2021.

[3] Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. Learning discrete structures for graph neural networks. In *ICML*, pages 1972–1982, 2019.

[4] Hashem Ghanem, Samuel Vaiter, and Nicolas Keriven. Gradient scarcity with bilevel optimization for graph learning. *arXiv preprint arXiv:2303.13964*, 2023.

[5] Edward Grefenstette, Brandon Amos, Denis Yarats, Phu Mon Htut, Artem Molchanov, Franziska Meier, Douwe Kiela, Kyunghyun Cho, and Soumith Chintala. Generalized inner loop meta-learning. *arXiv preprint arXiv:1910.01727*, 2019.

[6] Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *Proceedings of the 27th international conference on international conference on machine learning*, pages 399–406, 2010.

[7] Vassilis Kalofolias. How to learn a graph from smooth signals. In *Artificial Intelligence and Statistics*, pages 920–929. PMLR, 2016.

[8] Nicolas Keriven. Not too little, not too much: a theoretical analysis of graph (over) smoothing. In *NeurIPS*, 2022.

[9] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[10] Yixin Liu, Yu Zheng, Daokun Zhang, Hongxu Chen, Hao Peng, and Shirui Pan. Towards unsupervised deep graph structure learning. In *WWW*, 2022.

[11] Qing Lu and Lise Getoor. Link-based classification. In *ICML*, 2003.

[12] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *EMNLP*, 2015.

[13] Otilia Stretcu, Krishnamurthy Viswanathan, Dana Movshovitz-Attias, Emmanouil Platanios, Sujith Ravi, and Andrew Tomkins. Graph agreement models for semi-supervised learning. *NeurIPS*, 2019.

[14] Hongwei Wang and Jure Leskovec. Unifying graph convolutional neural networks and label propagation. *arXiv preprint arXiv:2002.06755*, 2020.