IncreGNN: Incremental Graph Neural Network Learning by Considering Node and Parameter Importance

Di Wei, Yu Gu(✉), Yumeng Song, Zhen Song, Fangfang Li, and Ge Yu

School of Computer Science and Engineering, Northeastern University, Shenyang, Liaoning, China
{guyu,lifangfang,yuge}@mail.neu.edu.cn

Abstract. Graph Neural Network (GNN) has shown powerful learning and reasoning ability. However, graphs in the real world generally exist dynamically, i.e., the topological structure of graphs is constantly evolving over time. On the one hand, the learning ability of the networks declines since the existing GNNs cannot process the graph streaming data. On the other hand, the cost of retraining GNNs from scratch becomes prohibitively high with the increasing scale of graph streaming data. Therefore, we propose an online incremental learning framework IncreGNN based on GNN in this paper, which solves the problem of high computational cost of retraining GNNs from scratch, and prevents catastrophic forgetting during incremental training. Specifically, we propose a sampling strategy based on node importance to reduce the amount of training data while preserving the historical knowledge. Then, we present a regularization strategy to avoid over-fitting caused by insufficient sampling. The experimental evaluations show the superiority of IncreGNN compared to existing GNNs in link prediction task.

Keywords: Graph neural networks · Dynamic graph · Catastrophic forgetting · Incremental learning

1 Introduction

Graphs are ubiquitous in the real world, which have been used for processing deep learning and data mining tasks. Through various analysis of graphs, we can have a deep understanding of complex social relationships and different communication modes. Since graph data is often high-dimensional and difficult to be processed by graph analysis tasks, Graph Embedding (GE) has been widely used as an effective dimensionality reduction technique [6]. As a deep learning-based GE method, GNN gradually shows its advantages in processing graph analysis tasks. The embedding obtained by GNN can not only capture the local neighborhood information, but also enables the embedding to characterize the global neighborhood through multiple iterations.

A. Bhattacharya et al. (Eds.): DASFAA 2022, LNCS 13245, pp. 739–746, 2022.
https://doi.org/10.1007/978-3-031-00123-9_59

At present, most GNNs are designed for static graph data. However, most real-world graphs exist dynamically, that is, as time goes by, new graph streaming data will continue to arrive, which is called the dynamic graph. There are some studies on dynamic graphs [5,10]. However, these dynamic graph embedding methods cannot process the incoming graph data in real time. As a result, such methods must retrain the model from scratch to obtain the embeddings of new nodes. Incremental learning provides ideas for solving this problem, which uses the latest data to update the current model and can prevent catastrophic forgetting. This technology significantly improves the efficiency of the model through reducing the number of nodes involved by retraining, and maintains the similar performance. ContinualGNN [12] is the incremental learning method based on GNN. However, it may reduce the expressive ability of the model and is not suitable for link prediction tasks.

To solve the problems of the existing incremental learning methods, we propose an online incremental learning framework IncreGNN based on GNN, which combines the experience replay-based and regularization-based methods. Motivated by experience replay, we design a sampling strategy based on node importance to sample affected and unaffected nodes separately and learn knowledge of new task on incremental data. Simultaneously, to prevent the historical knowledge from being weaken and overcome the over-fitting problem, we propose a regularization strategy by constraining the modification on the important model parameters. The main contributions of this work are as follows:

- We propose a GNN-based online incremental learning framework IncreGNN, which can efficiently generate node embedding representations in a dynamic environment.
- We design a sampling strategy based on node importance to sample the affected and unaffected nodes separately. At the same time, a regularization strategy based on the importance of model parameters is designed to constrain model parameters.
- We conduct comparative experiments on various datasets, and the results show that IncreGNN can efficiently perform incremental calculations with less accuracy loss.

2 Related Work

Incremental learning is also called continuous learning, or lifelong learning, which is first introduced in Neural Networks to solve multi-task learning problems. The current mainstream incremental learning methods can be divided into three categories including experience replay-based methods [7], regularization-based methods [1,3] and parameter isolation-based methods [4,9]. Due to the complex structure of graphs, most of the current incremental learning methods are applied to image processing and cannot directly process graph data. ContinualGNN [12] is the method to apply incremental learning to graphs. ContinualGNN proposes a novel approximate scoring algorithm to detect the emergence of new patterns. However, ContinualGNN performs sampling by calculating the probability, which will cause the loss of the local structure information of graphs. There are also incremental learning studies [2,13] that focus on improving model accuracy by retaining past knowledge rather than training efficiency, which is different from our concerns.

3 Overview of IncreGNN

Figure 1 shows the architecture of IncreGNN. A new task is defined as training a GNN on incremental data $\triangle G^t$ for the corresponding graph analysis tasks. Historical tasks are defined as training GNNs on historical incremental data $\{\triangle G^1, \triangle G^2, ..., \triangle G^t\}$ before time t to process graph analysis tasks at different time. There is usually only one graph analysis task involved for a GNN during model training. Therefore, we merge the historical incremental data $\{\triangle G^1, \triangle G^2, ..., \triangle G^t\}$ into an intergrated graph G^{t-1}, so that we only need to consider a single historical task when performing incremental learning.

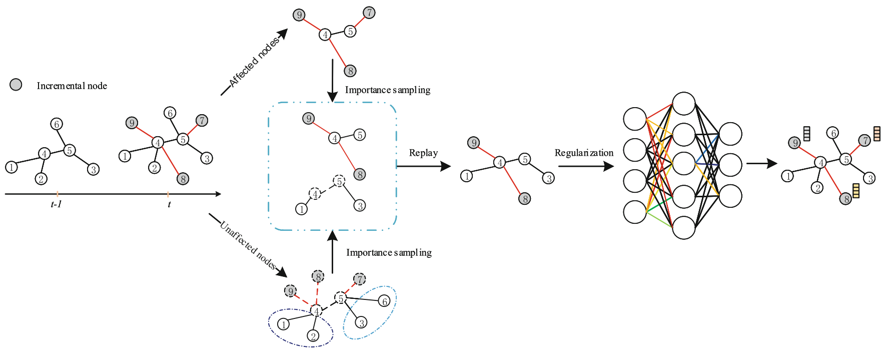


Fig. 1. Overview of IncreGNN

Our goal is to incrementally learn a GNN parameterized by θ with an optimization goal on graph G^t. Based on the above analysis, we explain IncreGNN from the perspective of probability. From this point of view, optimizing the parameters θ is tantamount to finding their most probable values given graph G^t. Therefore, according to Bayes rule, we can obtain the conditional probability by using the prior probability $p(\theta)$ of the parameter and the probability $p(G^t|\theta)$ of graph G^t:

$$logp(\theta|G^t) = logp(G^t|\theta) + logp(\theta) - logp(G^t) \tag{1}$$

Theorem 1. Graph G^t at time t is composed of historical graph G^{t-1} at time $t-1$ and incremental data $\triangle G^t$ at time t. The conditional probability $logp(\theta|G^t)$ can be calculated using historical graph G^{t-1} and incremental data $\triangle G^t$ as conditions:

$$logp(\theta|G^{t-1}, \triangle G^t) = logp(\triangle G^t|\theta) + logp(\theta|G^{t-1}) - logp(\triangle G^t) \tag{2}$$

where the first term $logp(\triangle G^t|\theta)$ is obviously the log-likelihood value of incremental data $\triangle G^t$, that is, the loss function of the learning task at time t is negative $-L_{\triangle G^t}(\theta)$, the second term $logp(\theta|G^{t-1})$ is a posterior distribution of all information learned on the data G^{t-1} related to the task at time $t-1$, and the third term $logp(\triangle G^t)$ is a constant value.

Therefore, according to (2), the total loss function of the IncreGNN at time t can be expressed as:

$$L_{IncreGNN} = L_{\triangle} + L_{pre} = L_{\triangle} + L_{olddata} + L_{param} = L_{data} + L_{param} \tag{3}$$

where $L_{\triangle}$ represents the loss function of the incremental change of the graph at time t, and L_{pre} represents the review of the old knowledge learned on the graph data before time t, so as to avoid catastrophic forgetting. The IncreGNN combines two strategies of experience replay and regularization, where the experience replay strategy is a combination of new and old data. Therefore, L_{pre} can be further decomposed into two parts: $L_{olddata}$, a review of knowledge learned from a part of the old data, and L_{param}, a constraint on the parameters of the old model. Combining $L_{\triangle}$ and $L_{olddata}$, L_{data} represents the loss function of using the experience replay strategy to process new and old data.

4 Experience Replay and Regularization Strategy

4.1 Experience Replay Strategy Based on Node Importance

In order to obtain more important historical nodes as much as possible, we introduce the experience replay strategy based on node importance. We associate the importance of a historical node with the degree of influence. Then, the experience replay strategy based on node importance is divided into the following two steps:

Step 1: Sampling the important neighbor nodes of the affected nodes in order. The set of incremental data is defined as base change group, where the incremental data includes incremental nodes and edges. Then, the neighbor nodes of the nodes in base change group are regarded as first-order change group, and the neighbor nodes of the nodes in first-order change group are regarded as second-order change group, and so on. It is not difficult to find that the lower the order of the node, the greater the contribution to the incremental node, that is, the greater the impact of the incremental node. So it is necessary to sample as many nodes as possible in low-order change group. Here, we gradually reduce the number of samples as the order increases, and assign the number of node samples at the ratio of $\frac{1}{i}/(1+\frac{1}{2}+...+\frac{1}{K})$ for each order, where K represents a total of K-order change groups of samples, and i represents the i-th change group currently sampled. Node importance of v is measured by the personalized PageRank value between the node and its neighbors in G^{t-1} and denoted as π_v^T.

By calculating node importance, we can get the nodes with high degree of correlation with the incremental nodes according to the number of sampling nodes in each order n_k, which are more important for the knowledge of the old task:

$$I(G^t) = \bigcup_{k=1}^{K} \{v_i | \pi_{v_i}^T > \pi_{v_j}^T, i \in [1, n_k], j \notin [1, n_k]\} \tag{4}$$

Step 2: Performing importance sampling on unaffected nodes. To prevent over-fitting and catastrophic forgetting, it is also necessary to perform partial sampling on unaffected nodes. Since random sampling may sample nodes of the same category or the embeddings are too similar, the incremental learning will be skewed and part of the old knowledge will be covered. Therefore, the unaffected nodes are divided using K-means according to the label or the learned embeddings for the unlabeled nodes in G^{t-1}, and K clusters are obtained. Within each cluster, the degree of the node is used as the criterion to measure node importance. The greater the degree of the node, the higher the node importance within the cluster. Finally, we uniformly sample the same number of k height nodes from each cluster by degree:

$$UI(G^t) = \bigcup_{k=1}^{K} \{v_i | deg_{v_i}^T > deg_{v_j}^T, i \in [1, n_k], j \notin [1, n_k]\} \tag{5}$$

Through node importance sampling strategies, the total number of sampled node sets is M, including the sum of the affected node set $I(G^t)$ and the unaffected node set $UI(G^t)$ sampled in K clusters. These M nodes will be used as the training data for incremental learning of GNN at time t. From the perspective of experience replay, the optimization goal for both retaining historical knowledge and learning new task knowledge is:

$$L_{data} = \sum_{v_i \in I(G^t) \cup UI(G^t)} l(\theta; v_i) \tag{6}$$

4.2 Regularization Strategy Based on Parameter Importance

The purpose of the regularization strategy is to avoid updating the parameters drastically related to the old knowledge. Specifically, a corresponding importance coefficient δ_{ij} is calculated for each parameter θ_{ij}, and the update extent of the parameter is restricted by this importance coefficient. For the parameter θ_{ij} with a large δ_{ij}, the magnitude of its change should be minimized during the gradient descent process, because a large importance coefficient δ_{ij} can indicate that this parameter θ_{ij} is more important to the old knowledge of model learning. Thus this parameter needs to be retained to avoid catastrophic forgetting.

Following Memory Aware Synapses (MAS) [1], for all nodes in the training set, we accumulate the mean value of the gradient calculated on the feature vectors of all nodes to obtain the importance weight δ_{ij} of the parameter θ_{ij}:

$$\delta_{ij} = \frac{1}{N} \sum_{v=1}^{N} ||g_{ij}(x_v)|| \tag{7}$$

where $g_{ij}(x_v) = \frac{\partial(F'(x_v;\theta))}{\partial\theta_{ij}}$ is the partial derivative of the function F' to the parameter θ_{ij}, F' is an approximate function mapping to the real function F, x_v is the feature of each node v and N is the number of nodes in the training set. However, calculating δ requires traversing the entire graph snapshot at the previous time, which will incur a lot of cost to storage and calculation. Therefore, we use the important nodes sampled by experience replay strategy to estimate the importance weight.

Moreover, in order to consolidate more historical knowledge of important parameters, it is necessary to accumulate the parameter importance weight δ calculated by each task at all previous times. Finally, the importance parameter δ_{ij} corresponding to the parameter θ_{ij} at time t is:

$$\delta_{ij} = \frac{1}{tN}\sum_{T=0}^{t-1}\sum_{v=1}^{N}||g_{ij}(x_v)|| \tag{8}$$

Through approximate calculation to obtain the estimated parameter importance weight, the optimization goal of consolidating historical knowledge from the perspective of constrained model parameters can be obtained:

$$L_{param} = \frac{\lambda}{2}\sum_{i,j}\delta_{ij}(\theta_{ij} - \theta_{ij}^{t-1})^2 \tag{9}$$

where λ is the degree of importance of historical knowledge to new graph.

5 Experiments

5.1 Experiment Setup

We conduct experiments on Enron[1], UCI[2], BC-Alpha[3] and ML-10M[4], which are all divide into 13 graph snapshots following [8]. For link prediction task, 40%, 20% and 40% of the edges are taken as training set, validation set and test set, respectively. We compare the proposed method with five baselines, including EvolveGCN [5], Retrained GAT [11], Pretrained GAT [11], Online GAT [11] and ContinualGNN [12]. Retrained GAT, Pretrained GAT and Online GAT are GATs for retraining, pre-training and dynamic graph online training respectively. The experiment uses all the data at $t = 0$ to train a general model, and then incrementally learns the sampled data at $t = 1...T$. The regularization term λ is (80, 800, 80, 320).

[1] https://www.cs.cmu.edu/~./enron/.
[2] http://networkrepository.com/opsahl_ucsocial.php.
[3] http://www.btc-alpha.com.
[4] http://networkrepository.com/ia-movielens-user2tags-10m.php.

Table 1. Performance of link prediction.

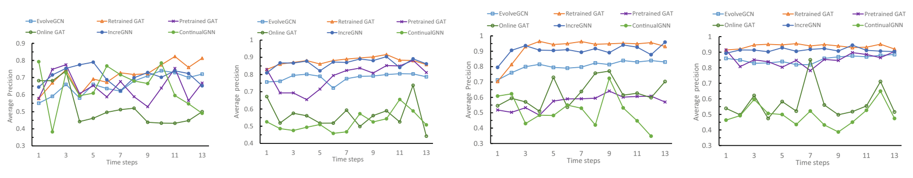
Method	Enron			UCI			BC-Alpha			ML-10M		
	AUC	AP	Time(s)	AUC	AP	Time(s)	AUC	AP	Time(s)	AUC	AP	Time(s)
EvolveGCN	65.4	66.0	0.691	77.1	78.9	1.744	79.3	81.0	1.30	85.0	85.6	5.174
RetrainedGAT	**68.9**	**71.5**	0.057	**87.5**	**87.8**	0.677	**91.7**	**91.9**	0.422	**93.1**	**93.9**	3.329
PretrainedGAT	62.5	64.3	0.000	81.9	78.8	0.000	42.0	57.0	0.000	87.6	84.9	0.000
OnlineGAT	44.6	52.2	0.033	58.1	56.2	0.115	64.9	63.0	0.064	61.9	57.3	0.249
ContinualGNN	57.1	64.3	0.006	49.7	52.4	0.061	49.3	53.5	0.147	47.9	49.5	7.469
IncreGNN	**67.3**	**71.2**	0.019	**82.6**	**84.7**	0.143	**90.6**	**90.8**	0.045	**89.9**	**91.4**	0.070

5.2 Experimental Results

Table 1 shows the average AUC, AP values and training time of all methods. Note that the training time does not include the time to calculate personalized PageRank and K-means, which can be calculated offline before the next snapshot coming. The importance parameter δ_{ij} is updated online. Experimental results show that our proposed algorithm IncreGNN achieves the best performance on both AUC and AP compared to other comparison methods in addition to Retrained GAT. At the same time, IncreGNN can reach experimental results that are similar to the theoretically most accurate Retrained GAT, which shows that the method we propose has high superiority in incremental learning.

Figure 2 shows the performance results of each method across multiple time steps. On the UCI, BCAlpha and ML-10M datasets, our method has reached a very high level and are very stable, indicating that the method we proposed can effectively preserve old knowledge and learn new knowledge. The results of Online GAT on the 4 datasets are very poor and fluctuate greatly. The reason is that Online GAT does not preserve old knowledge, which leads to catastrophic forgetting.

IncreGNN also performs well on the node classification task, but the experiments are omitted due to limited space.



(a) Enron (b) UCI (c) BC-Alpha (d) ML-10M

Fig. 2. Link prediction across multiple time steps

6 Conclusion

In this work, we propose a GNN-based incremental learning framework to process dynamic graphs. We design a sampling strategy based on node importance to sample the affected and unaffected nodes. At the same time, we design a regularization strategy based on the importance of model parameters to constrain the important parameters. Through these two strategies, the problem of catastrophic forgetting in incremental learning can be avoided, and the knowledge of the new task can be learned while preserving the knowledge of the old task. Experimental results verify the effectiveness and efficiency of IncreGNN.

Acknowledgements. This work is supported by the National Natural Science Fondation of China (62072083 and U1811261).

References

1. Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., Tuytelaars, T.: Memory aware synapses: Learning what (not) to forget. In: ECCV, pp. 139–154 (2018)
2. Galke, L., Franke, B., Zielke, T., Scherp, A.: Lifelong learning of graph neural networks for open-world node classification. In: IJCNN, pp. 1–8 (2021)
3. Li, Z., Hoiem, D.: Learning without forgetting. IEEE Trans. Pattern Analysis and Machine Intelligence **40**(12), 2935–2947 (2017)
4. Mallya, A., Lazebnik, S.: PackNet: adding multiple tasks to a single network by iterative pruning. In: CVPR, pp. 7765–7773 (2018)
5. Pareja, A., et al.: EvolveGCN: evolving graph convolutional networks for dynamic graphs. In: AAAI, vol. 34, pp. 5363–5370 (2020)
6. Peng, Y., Choi, B., Xu, J.: Graph learning for combinatorial optimization: a survey of state-of-the-art. Data Sci. Eng. **6**(2), 119–141 (2021)
7. Rebuffi, S.A., Kolesnikov, A., Sperl, G., Lampert, C.H.: iCaRL: incremental classifier and representation learning. In: CVPR, pp. 2001–2010 (2017)
8. Sankar, A., Wu, Y., Gou, L., Zhang, W., Yang, H.: DySAT: deep neural representation learning on dynamic graphs via self-attention networks. In: WSDM, pp. 519–527 (2020)
9. Serra, J., Suris, D., Miron, M., Karatzoglou, A.: Overcoming catastrophic forgetting with hard attention to the task. In: International Conference on Machine Learning, pp. 4548–4557. PMLR (2018)
10. Trivedi, R., Farajtabar, M., Biswal, P., Zha, H.: DyRep: learning representations over dynamic graphs. In: ICLR (2019)
11. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
12. Wang, J., Song, G., Wu, Y., Wang, L.: Streaming graph neural networks via continual learning. In: CIKM, pp. 1515–1524 (2020)
13. Xu, Y., Zhang, Y., Guo, W., Guo, H., Tang, R., Coates, M.: GraphSAIL: graph structure aware incremental learning for recommender systems. In: CIKM, pp. 2861–2868 (2020)