ProGCL: Rethinking Hard Negative Mining in Graph Contrastive Learning

Jun Xia\(^1\)\(^2\)  Lirong Wu\(^1\)\(^2\)  Ge Wang\(^1\)\(^2\)  Jintao Chen\(^3\)  Stan Z. Li\(^1\)\(^2\)

Abstract

Contrastive Learning (CL) has emerged as a dominant technique for unsupervised representation learning which embeds augmented versions of the anchor close to each other (positive samples) and pushes the embeddings of other samples (negatives) apart. As revealed in recent studies, CL can benefit from hard negatives (negatives that are most similar to the anchor). However, we observe limited benefits when we adopt existing hard negative mining techniques of other domains in Graph Contrastive Learning (GCL). We perform both experimental and theoretical analysis on this phenomenon and find it can be attributed to the message passing of Graph Neural Networks (GNNs). Unlike CL in other domains, most hard negatives are potentially false negatives (negatives that share the same class with the anchor) if they are selected merely according to the similarities between anchor and themselves, which will undesirably push away the samples of the same class. To remedy this deficiency, we propose an effective method, dubbed ProGCL, to estimate the probability of a negative being true one, which constitutes a more suitable measure for negatives’ hardness together with similarity. Additionally, we devise two schemes (i.e., ProGCL-weight and ProGCL-mix) to boost the performance of GCL. Extensive experiments demonstrate that ProGCL brings notable and consistent improvements over base GCL methods and yields multiple state-of-the-art results on several unsupervised benchmarks or even exceeds the performance of supervised ones. Also, ProGCL is readily pluggable into various negatives-based GCL methods for performance improvement. We release the code at [https://github.com/junxia97/ProGCL](https://github.com/junxia97/ProGCL).

\(^1\)Westlake University  \(^2\)Westlake Institute for Advanced Study  \(^3\)School of Computer Science, Zhejiang University. Correspondence to: Jun Xia <xiajun@westlake.edu.cn>.

Proceedings of the 39\textsuperscript{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

Figure 1. Similarity (cosine similarity between normalized embeddings of anchor and negatives) histograms of negatives. We adopt SimCLR (Chen et al., 2020) and GCA (Zhu et al., 2021c) for CIFAR-10 and Coauthor-CS respectively. False (or true) negatives denote the samples that are (or not) from the same class with the anchor. Unlike CL, most negatives with larger similarities to the anchor are false ones in GCL. More examples are in the appendix.

1. Introduction

Recently, Contrastive Learning (CL) has demonstrated unprecedented unsupervised performance in computer vision (He et al., 2020; Chen et al., 2020), natural language processing (Gao et al., 2021; Zheng et al., 2022) and graph representation learning (Velickovic et al., 2019; Xia et al., 2022b; Zhu et al., 2021c). As with metric learning (Kaya & Bilge, 2019), existing works both theoretically and practically validate that hard negatives contribute more to CL. For example, HCL (Robinson et al., 2021) develops a family of unsupervised sampling methods for selecting hard negatives and MoCHi (Kalantidis et al., 2020) mixes the hard negatives to synthesize more hard negative ones. However, we observe minor improvement or even significant performance drop when we adopt these negative mining techniques in GCL (the results can be seen in Table 5).

Concurrent to our work, Zhu et al. (2021b) also observe that existing hard negative mining techniques that work well in other domains bring limited benefits to GCL. Unfortunately, they didn’t provide any solution to tackle this issue.

To explain these phenomena, we first plot the negatives’ distributions over similarity of various datasets in Figure 1. Note that we do not observe significant changes of negatives’ distribution (keep unimodal as shown in Figure 1(a)) during the training process of SimCLR (Chen et al., 2020) on CIFAR-10 and other image datasets. However, for the
ProGCL: Rethinking Hard Negative Mining in Graph Contrastive Learning

first stage of GCL, the negatives’ distribution is bimodal for a long period as Figure 1(b) and then progressively transit to unimodal distribution as CL at the second stage. Similar phenomena for more datasets can be found in the appendix. The difference of negatives’ distribution between CL and GCL can be attributed to the unique message passing of Graph Neural Networks, which we further discuss in section 3.2. These provide explanations for the poor performance of existing negative mining techniques in GCL. Specifically, they regard the negatives that are most similar to anchor points as hard ones across all the training process. However, as shown in Figure 1(b), most selected “hard” negatives in this way are false negatives for GCL indeed, which will undesirably push away the semantically similar samples and thus degrade the performance. The existence of false negatives is termed as sampling bias in DCL (Chuang et al., 2020). However, as reported in Table 5 and Zhu et al. (2021b), DCL brings performance drop for GCL because the sampling bias is severer in GCL. Now, we are naturally motivated to ask following questions: Are there better alternatives to measure negatives’ hardness considering the issue of false negatives in GCL? Can we devise more suitable methods to eliminate severer bias in GCL?

To answer these questions, we argue that true and false negatives can be distinguished by fitting a two-component (true-false) beta mixture model (BMM) on the similarity. The posterior probability of a negative being true one under BMM can constitute a more suitable measure for negatives’ hardness accompanied with similarity. With the novel measure, we devise two schemes (ProGCL-weight and ProGCL-mix) for further improvement of negatives-based GCL methods. To the best of our knowledge, our work makes one of the pioneering attempts to study hard negative mining in node-level GCL. We highlight the following contributions:

- We demonstrate the difference of negatives’ distribution between GCL and CL and explain why existing hard negative mining techniques cannot work well in GCL with both theoretical and experimental analysis.
- We propose to utilize BMM to estimate the probability of a negative being true one relative to a specific anchor. Combined with the similarity, we obtain a more suitable measure for negatives’ hardness.
- We devise two schemes (i.e., ProGCL-weight and ProGCL-mix) that are more suitable for hard negative mining in GCL.
- ProGCL brings notable and consistent improvements over base GCL methods and yields multiple state-of-the-art results on several unsupervised benchmarks or even exceeds the performance of supervised ones. Also, it can boost various negatives-based GCL methods for further improvements.

2. Related Work

2.1. Graph Contrastive Learning (GCL)

Recently, GCL has gained popularity in unsupervised graph representation learning, which can get rid of the resource-intensive annotations (Xia et al., 2021; Tan et al., 2021; Xia et al., 2022a). Initially, DGI (Velickovic et al., 2019) and InfoGraph (Sun et al., 2020) are proposed to obtain expressive representations for graphs or nodes via maximizing the mutual information between graph-level representations and substructure-level representations of different granularity. Similarly, GMI (Peng et al., 2020) adopts two discriminators to directly measure mutual information between input and representations of both nodes and edges. Additionally, MVGRL (Hassani & Khasahmadi, 2020) proposes to learn both node-level and graph-level representation by performing node diffusion and contrasting node representation to augmented graph representation. Different from our work, GraphCL (You et al., 2020) and its variants (Suresh et al., 2021; Xu et al., 2021; You et al., 2021; 2022; Li et al., 2021) adopt SimCLR framework and design various augmentations for graph-level representation learning. For node-level representations, GRACE (Zhu et al., 2020) and its variants (Zhu et al., 2021c; Tong et al., 2021) maximize the agreement of node embeddings across two corrupted views of the graph. More recently, SimGRACE (Xia et al., 2022b), BGRL (Thakoor et al., 2021) and CCA-SSG (Zhang et al., 2021) try to simplify graph contrastive learning via discarding the negatives, parameterized mutual information estimator or even data augmentations. In this paper, we consider hard negatives mining to further boost GCL on node-level representation learning, which is still under-explored. For the limited space, we recommend readers refer to a recent review (Xia et al., 2022d) for more relevant literatures.

2.2. Hard Negative Mining in Contrastive Learning

For contrastive learning, HCL (Robinson et al., 2021) and MoChi (Kalantidis et al., 2020) are proposed to emphasize or synthesize hard negatives. Additionally, Ring (Wu et al., 2021) introduces a family of mutual information estimators that sample negatives in a “ring” around each positive. However, as shown in Table 5, these techniques which emphasize hard negatives don’t work well in GCL. For GCL, Zhao et al. (Zhao et al., 2021) utilize the clustering pseudo labels to alleviate false negative issue, which suffers from heavy computational overhead and will degrade the performance when confronted with multi-class datasets. CuCo (Chu et al., 2022) sorts the negatives from easy to hard with similarity for graph-level contrastive learning and proposes to automatically select and train negative samples with curriculum learning. Instead, we study node-level GCL with message passing among instances. Additionally, Zhu et al. (Zhu et al., 2022) propose a structure-enhanced negative mining scheme.
which discovers hard negatives in each metapath-induced view in heterogeneous graph. However, it is not suitable for homogeneous graphs without metapath.

3. Methodology

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the graph, where $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}, \mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denote the node set and edge set respectively. Additionally, $X \in \mathbb{R}^{N \times F}$ and $A \in \{0, 1\}^{N \times N}$ are the feature matrix and the adjacency matrix. $f_i \in \mathbb{R}^F$ is the feature of $v_i$, and $A_{ij} = 1$ iff $(v_i, v_j) \in \mathcal{E}$. Our objective is to learn a GNN encoder $f(X, A) \in \mathbb{R}^{N \times F'}$ to embed nodes in low dimensional space without label information. These low dimensional representations can be used in downstream tasks including node classification. As shown in Figure 2, we sample two augmentation functions $t \sim \mathcal{T}$ and $t' \sim \mathcal{T}$, here $\mathcal{T}$ is the set of all augmentation functions. We can then obtain two views for $\mathcal{G}$, $\mathcal{G}_1 = t(\mathcal{G})$ and $\mathcal{G}_2 = t'(\mathcal{G})$. Given $\mathcal{G}_1 = (\hat{X}_1, \hat{A}_1)$ and $\mathcal{G}_2 = (\hat{X}_2, \hat{A}_2)$, we denote node embeddings in the two views as $U = f(\hat{X}_1, \hat{A}_1)$ and $V = f(\hat{X}_2, \hat{A}_2)$. For any node $v_i$, its embedding in one view $v_i$ is regarded as the anchor. The embedding $u_i$ in the other view is the positive sample and the embeddings of other nodes in both views are negatives. Similar to the InfoNCE (Oord et al., 2018), the training objective for each positive pair $(u_i, v_i)$ is,

$$J = -\frac{1}{2N} \sum_{i=1}^{N} [\ell(u_i, v_i) + \ell(v_i, u_i)].$$

Many GCL methods are built on this framework (Zhu et al., 2021b; Qiu et al., 2020; Zhu et al., 2021c; Jin et al., 2021), which motivates us to study hard negative mining on it.

3.2. Experimental and Theoretical Analysis

In Figure 3, we study the message passing’s role in GCL via replacing 2-layer GCN (Kipf & Welling, 2016a) that is composed of message passing and multi-layer perceptron (MLP) with a 2-layer MLP encoder only. As can be observed, the similarity histograms of GCL will be similar to that of CL without message passing, which verifies that message passing of GNN encoder is the key factor for the difference of negatives’ distribution between CL and GCL. More specifically, for the first bimodal stage of GCL, the message passing in GCL enlarges the similarities between neighbour nodes which often share the same class. For the second unimodal stage, instance discrimination of GCL occupies a position of prominence and pushes away all the other samples regardless of their semantic class. Theoretically, for embeddings of any nodes pair in the graph $\mathcal{G}$, we can compare their distance before and after message passing and show that the distance will be decreased with the process, as formally induced in Theorem 3.1.

**Theorem 3.1.** $\mathcal{G}$ is a non-bipartite and connected graph with $N$ nodes $\mathcal{V} = \{v_1, \ldots, v_N\}$, and $X^{(\tau)}$ is the embedding of node $v_i$ after $\tau$ times message passing ($X^{(0)} = f_i$). For large enough $\tau$, $||X_i^{(\tau)} - X_j^{(\tau)}||_2 \leq ||X_i^{(0)} - X_j^{(0)}||_2$.

**Proof.** Given the message passing is,

$$X^{(t+1)} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} X^{(t)},$$
where $\hat{A} = A + I$ and $\hat{D}_i = \sum_j \hat{A}_{ij}$. We can then rewrite the message passing as,

$$X^{(t+1)} = (I - L_{sym})X^{(t)},$$

where $L_{sym} = \hat{D}^{-\frac{1}{2}}\hat{L}\hat{D}^{-\frac{1}{2}}$. Let $(\lambda_1, \ldots, \lambda_N)$ and $(e_1, \ldots, e_N)$ denote the eigenvalue and eigenvector of matrix $I - L_{sym}$, respectively. With the property of symmetric Laplacian matrix for non-bipartite and connected graph,

$$-1 < \lambda_1 < \lambda_2 < \cdots < \lambda_N = 1,$$

we can rewrite $X^{(\tau)}_i - X^{(\tau)}_j$ as

$$X^{(\tau)}_i - X^{(\tau)}_j = [(I - L_{sym})^\tau X]_i - [(I - L_{sym})^\tau X]_j = [\lambda_1^{\tau}(e_1^{(i)} - e_1^{(j)}), \ldots, \lambda_N^{\tau}(e_N^{(i)} - e_N^{(j)})]\hat{X},$$

where $e_k^{(i)}$ is the $i$th element of eigenvector $e_k$, $\hat{X}$ is the coordinate matrix of $X$ in the space spanned by eigenvectors $(e_1, \ldots, e_N)$. Thus,

$$||X^{(\tau)}_i - X^{(\tau)}_j||_2 = \sqrt{\sum_{m=1}^{N} \sum_{k=1}^{N-1} \lambda_k^{\tau}(e_k^{(i)} - e_k^{(j)})^2 ||\hat{X}||_{km}^2}.$$

Because $1 < \lambda_1 < \lambda_2 < \cdots < \lambda_N < 1$, thus for a large $\tau$, $||X^{(\tau)}_i - X^{(\tau)}_j||_2 \leq ||X^{(0)}_i - X^{(0)}_j||_2$. \hfill $\square$

### 3.3. ProGCL

We aim to estimate the probability of a negative being true one. As can be observed in Figure 4, there is a significant difference between the false negatives and true negatives’ distributions in GCL, allowing the two types of negatives can be distinguished from the similarity distribution. Here we propose to utilize mixture model to estimate the probability. Gaussian Mixture Model (GMM) is the most popular mixture model (Lindsay, 1995). However, in Figure 4, the distribution of false negatives is skew and thus symmetric Gaussian distribution can not fit this well. To circumvent this issue, we resort to beta distribution (Gupta & Nadarajah, 2004; Ji et al., 2005) which is flexible enough to model various distributions (symmetric, skewed, arched distributions and so on) over $[0, 1]$. As can be observed in Figure 4, Beta Mixture Model (BMM) can fit the empirical distribution better than GMM. Also, we compare the performance of ProGCL with BMM and GMM in Table 4 and find that BMM consistently outperforms GMM. The probability density function (pdf) of beta distribution is,

$$p(s \mid \alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} s^{\alpha-1}(1-s)^{\beta-1},$$

where $\alpha, \beta > 0$ are the parameters of beta distribution and $\Gamma(\cdot)$ is gamma function. The pdf of beta mixture model of

### Figure 4.

Empirical distribution and estimated GMM and BMM on Amazon-Photo and Coauthor-CS datasets. The BMM fits the distributions better. Here $C_1, C_2$ denote the estimated distributions of two components respectively. Note that BMM is defined on the interval $[0, 1]$ while GMM is defined over $[\infty, +\infty]$.

$C$ components on $s$ (Min-Max normalized cosine similarity between normalized embeddings of anchors and negatives) can be defined as:

$$p(s) = \sum_{c=1}^{C} \lambda_c p(s \mid \alpha_c, \beta_c),$$

where $\lambda_c$ are the mixture coefficients. Here we can fit a two-component BMM (i.e., $C = 2$) to model the distribution of true and false negatives. We then utilize Expectation Maximization (EM) algorithm to fit BMM to the observed distribution. In E-step, we fix the parameters of BMM $(\lambda_c, \alpha_c, \beta_c)$ and update $p(c \mid s)$ with Bayes rule,

$$p(c \mid s) = \frac{\lambda_c p(s \mid \alpha_c, \beta_c)}{\sum_{j=1}^{C} \lambda_j p(s \mid \alpha_j, \beta_j)}.$$

In practice, fitting BMM with all the similarities will incur high computational cost. Instead, we can fit BMM well by only sampling $M$ ($M \ll N^2$) similarities for simplification. Larger $M$ bring limited benefits, which we will discuss in the appendix. We can then obtain the weighted average $\bar{s}_c$ and variance $\bar{v}_c^2$ over $M$ similarities,

$$\bar{s}_c = \frac{\sum_{i=1}^{M} p(c \mid s_i) s_i}{\sum_{i=1}^{M} p(c \mid s_i)}, \quad \bar{v}_c^2 = \frac{\sum_{i=1}^{M} p(c \mid s_i) (s_i - \bar{s}_c)^2}{\sum_{i=1}^{M} p(c \mid s_i)}.$$

For M-step, the component model parameters $\alpha_c, \beta_c$ can be estimated using the method of moments in statistics,

$$\alpha_c = \bar{s}_c \left( \frac{\bar{s}_c (1 - \bar{s}_c)}{\bar{v}_c^2} - 1 \right), \quad \beta_c = \frac{\alpha_c (1 - \bar{s}_c)}{\bar{s}_c},$$

and coefficients $\lambda_c$ for mixing can be calculated as,

$$\lambda_c = \frac{1}{M} \sum_{i=1}^{M} p(c \mid s_i).$$

The above E and M-steps are iterated until convergence or the maximum of iterations $I$ are reached. In our experiments, we set $I = 10$ and we study the influence of $I$ in
the appendix. Finally, we can obtain the probability of a negative being true or negative one relative to the anchor with their similarity $s$,

$$p(c | s) = \frac{\lambda_c p(s | \alpha_c, \beta_c)}{p(s)} .$$ (9)

Note that we can regard the fitted distribution with larger $\lambda_c$ as true negatives’ distribution because there are more true negatives than false ones relative to each anchor in multi-class datasets. Also, we can regard the fitted distribution with smaller mean as true negatives’ distribution because the anchor and false negatives share the same class and they are more similar to each other in general. With the posterior probabilities, we devise two schemes to boost the performance of existing GCL as we elaborate below.

### 3.4. Scheme 1: ProGCL-weight

As revealed above, GCL suffers from severe sampling bias which will undermine the performance. To tackle this problem, we propose a novel measure for negatives’ hardness considering the hardness and the probability of a negative being true one simultaneously,

$$w(i, k) = \frac{p(c_i | s_{ik}) s_{ik}}{\sum_{j \neq i} [p(c_i | s_{ij}) s_{ij}]} .$$ (10)

where $s_{ik}$ is the similarity between anchor $u_i$ and its inter-view negative sample $v_k$ and $p(c_i | s_{ij})$ denotes the probability of $v_j$ being true negative relative to anchor $u_i$. Note $w(i, k)$ can be utilized to weight both inter-view ($u_i, v_k$) and intra-view ($u_i, u_k$) negative pair,

$$\ell_u (u_i, v_i) = \log \left( \frac{e^{\theta(u_i, v_i)}}{\sum_{j \neq i} w(i, k) e^{\theta(u_i, v_j)} + \sum_{k \neq i} w(i, k) e^{\theta(u_i, u_k)}} \right) .$$ (11)

we can then define the new overall loss as the average of all the positive pairs,

$$J_u = -\frac{1}{2N} \sum_{i=1}^{N} [\ell_u (u_i, v_i) + \ell_u (v_i, u_i)] .$$ (12)

### 3.5. Scheme 2: ProGCL-mix

Recently, MoCHi (Kalantidis et al., 2020) proposes to synthesize more negatives with “hard” negatives selected only with similarity. However, as analysed above, many synthesized hard negatives in GCL are positive samples indeed, which will undermine the performance. To remedy this deficiency, we propose ProGCL-mix which synthesizes more hard negatives considering the probability of a negative being true one. The comparison between MoCHi and ProGCL-mix can be seen in Figure 5. More specifically, for each anchor point $u_i$, we synthesize $m$ hard negatives by convex linear combinations of pairs of its “hardest” existing negatives. Here, “hardest” existing negatives refers to $N'$ negatives that are selected with the measure in Eq. (10). Instead of mixing $N'$ samples randomly, we mix them by emphasizing samples that are more likely to be true negative. Formally, for each anchor $u_i$, a synthetic point $\tilde{u}_k (k \in [1, m])$ would be given by,

$$\tilde{u}_k = \alpha_k u_p + (1 - \alpha_k) u_q ,$$ (13)

where $u_p, u_q$ are selected from $N'$ “hardest” existing negatives measured by Eq (10) and $\alpha_k$ can be calculated as,

$$\alpha_k = \frac{p(c_i | s_{ip}) p(c_i | s_{iq}) + p(c_i | s_{ip}) + p(c_i | s_{iq})}{p(c_i | s_{ip}) p(c_i | s_{iq})} ,$$ (14)

we can then define the training objective for each positive pair $(u_i, v_i)$ with the synthetic negatives,

$$\ell_m (u_i, v_i) = \log \left( \frac{e^{\theta(u_i, v_i)}}{\sum_{j \neq i} \sum_{k \neq i} e^{\theta(u_i, v_j)} + \sum_{k \neq i} e^{\theta(u_i, u_k)} + \sum_{k=1}^{m} e^{\theta(u_i, \tilde{u}_k)}} \right) .$$ (15)

we can then define the new overall loss as the average of all the positive pairs,

$$J_m = -\frac{1}{2N} \sum_{i=1}^{N} [\ell_m (u_i, v_i) + \ell_m (v_i, u_i)] .$$ (16)

### 3.6. Time Complexity Analysis

It is noteworthy that the estimation of the posterior probabilities introduces light computational overhead over the base model. Firstly, we only have to fit BMM once during the training process instead of once per epoch. Secondly, we can fit BMM well with $M (M \ll N^2)$ similarities and the time complexity of EM algorithm for fitting in ProGCL
Algorithm 1 ProGCL-weight & -mix (Transductive)

Input: $T, \mathcal{G}, f, g, N$, normalized cosine similarity $s$, epoch for fitting BMM $E$, mode (‘weight’ or ‘mix’).

for epoch = 0, 1, 2, ... do
  Draw two augmentation functions $t \sim T$, $t' \sim T$
  $\mathcal{G}_1 = t(\mathcal{G})$, $\mathcal{G}_2 = t'(\mathcal{G})$
  $U = f(\mathcal{G}_1)$, $V = f(\mathcal{G}_2)$
  for $u_i \in U$ and $v_i \in V$ do
    $s_{ij} = s(g(u_i), g(v_i))$
    if epoch = $E$ then
      Compute $p(c_i | s_{ij})$ with Eq. (4) to Eq. (9).
    end if
  end for
  if epoch $\geq E$ then
    if mode = ‘weight’ then
      Compute $J_w$ with Eq. (10) to Eq. (12).
      Update the parameters of $f, g$ with $J_w$;
    end if
    if mode = ‘mix’ then
      Compute $J_m$ with Eq. (13) to Eq. (16).
      Update the parameters of $f, g$ with $J_m$.
    end if
  else
    Compute $J$ with Eq. (1) to Eq. (2).
    Update the parameters of $f, g$ with $J$.
  end if
end for

Output: $f, g$.

4. Experiments

4.1. Experimental Setup & Baselines

Following previous work (Velickovic et al., 2019), we train the model in an unsupervised manner. The resulting embeddings are utilized to train and test a simple $\ell_2$-regularized logistic classifier. We train the classifier for 20 runs. Additionally, we adopt GRACE as base model and measure performance using micro-averaged F1-score on inductive tasks. For transductive tasks, we adopt GCA as base model and report the test accuracy.

Transductive learning. We adopt a two-layer GCN (Kipf & Welling, 2016a) as the encoder for transductive learning following previous works (Velickovic et al., 2019; Zhu et al., 2020). We consider our ProGCL with multiple baselines including DeepWalk (Perozzi et al., 2014) and node2vec (Grover & Leskovec, 2016). Additionally, we also consider recent methods including Graph AutoEncoders (GAE, VGAE) (Kipf & Welling, 2016b), DGI, GMI, MVGRL, MERIT, BGRL and GCA as introduced in the related work. We report the best performance of three variants of GCA. We also compare ProGCL with supervised counterparts including GCN (Kipf & Welling, 2016a) and Graph Attention Networks (GAT) (Velickovic et al., 2017).

Inductive learning on large graphs. Considering the large scale of some graph datasets, we adopt a three-layer GraphSAGE-GCN with residual connections as the encoder following DGI. We adopt the sub-sampling strategy in GraphSAGE where we first select a mini-batch of nodes and then a subgraph centered around each selected node is obtained by sampling node neighbors with replacement. More specifically, we sample 10, 10 and 25 neighbors at the first, second and third level, respectively as DGI. The batchsize of our experiments is 256. Also, we estimate the posterior with pairwise similarities among each mini-batch instead of total training set, which we elaborate on in the appendix. We set traditional methods DeepWalk and deep learning based methods unsupervised GraphSAGE, DGI, a recent block-contrastive method COLES (Zhu et al., 2021a) and GMI as baselines. To compare ProGCL with supervised counterparts, we report the performance of two supervised methods FastGCN (Chen et al., 2018) and GraphSAGE. More details can be seen in the appendix.

4.2. Datasets

We conduct experiments on seven widely-used datasets including Amazon-Photo, Amazon-Computers, Wiki-CS, Coauthor-CS, Reddit, Flickr and ogbn-arXiv. To keep fair, for transductive tasks, we split Amazon-Photo, Amazon-Computers, Wiki-CS and Coauthor-CS for the training, validation and testing following (Zhu et al., 2021c). For inductive task, we split Reddit and Flickr following (Velickovic et al., 2019; Zeng et al., 2019). The experimental setting of ogbn-arXiv is the same as BGRL (Thakoor et al., 2021). More information of the datasets is in the appendix.

4.3. Comparison with State-of-the-art Results

For transductive classification, as can be observed in Table 1, ProGCL consistently performs better than previous unsupervised baselines or even the supervised baselines, which validates the superiority of our ProGCL. We provide more observations as following. Firstly, traditional methods node2vec and DeepWalk only using adjacency matrix outperform the simple logistic regression classifier that only uses raw features (“Raw features”) on Amazon datasets. However, the latter can perform better on Coauthor-CS.
Table 1. Summary of the accuracies (± std) on transductive node classification. The ‘Available Data’ refers to data we can obtain for training, where X, A and Y denotes feature matrix, adjacency matrix and label matrix respectively. ‘OOM’: out of memory on a 32GB GPU. We highlight the performance of ProGCL with gray background. The highest performance of unsupervised models is highlighted in boldface; the highest performance of supervised models is underlined. The baselines marked with *’ are reproduced with the same experimental settings (20 random dataset splits and model initializations). The other results are taken from previously published reports.

<table>
<thead>
<tr>
<th>Method</th>
<th>Available Data</th>
<th>Amazon-Photo</th>
<th>Amazon-Computers</th>
<th>Coauthor-CS</th>
<th>Wiki-CS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw features</td>
<td>X</td>
<td>78.53 ± 0.00</td>
<td>73.81 ± 0.00</td>
<td>90.37 ± 0.00</td>
<td>71.98 ± 0.00</td>
</tr>
<tr>
<td>node2vec</td>
<td>A</td>
<td>89.67 ± 0.12</td>
<td>84.39 ± 0.08</td>
<td>85.08 ± 0.03</td>
<td>71.79 ± 0.05</td>
</tr>
<tr>
<td>DeepWalk</td>
<td>A</td>
<td>89.44 ± 0.11</td>
<td>85.68 ± 0.06</td>
<td>84.61 ± 0.22</td>
<td>74.35 ± 0.06</td>
</tr>
<tr>
<td>DeepWalk + features</td>
<td>X, A</td>
<td>90.05 ± 0.08</td>
<td>86.28 ± 0.07</td>
<td>87.70 ± 0.04</td>
<td>77.21 ± 0.03</td>
</tr>
<tr>
<td>GAE</td>
<td>X, A</td>
<td>91.62 ± 0.13</td>
<td>85.27 ± 0.19</td>
<td>90.01 ± 0.17</td>
<td>70.15 ± 0.01</td>
</tr>
<tr>
<td>VGAE</td>
<td>X, A</td>
<td>92.20 ± 0.11</td>
<td>86.37 ± 0.21</td>
<td>92.11 ± 0.09</td>
<td>75.35 ± 0.14</td>
</tr>
<tr>
<td>DGI</td>
<td>X, A</td>
<td>91.61 ± 0.22</td>
<td>83.95 ± 0.47</td>
<td>92.15 ± 0.63</td>
<td>75.35 ± 0.14</td>
</tr>
<tr>
<td>GMI</td>
<td>X, A</td>
<td>90.68 ± 0.17</td>
<td>82.21 ± 0.31</td>
<td>92.18 ± 0.15</td>
<td>77.43 ± 0.17</td>
</tr>
<tr>
<td>MVGRL*</td>
<td>X, A</td>
<td>92.08 ± 0.01</td>
<td>87.45 ± 0.21</td>
<td>92.72 ± 0.03</td>
<td>78.41 ± 0.09</td>
</tr>
<tr>
<td>BRGRL*</td>
<td>X, A</td>
<td>92.95 ± 0.07</td>
<td>87.89 ± 0.10</td>
<td>92.51 ± 0.14</td>
<td>78.35 ± 0.05</td>
</tr>
<tr>
<td>MERIT*</td>
<td>X, A</td>
<td>92.53 ± 0.15</td>
<td>88.01 ± 0.12</td>
<td>92.40 ± 0.07</td>
<td>78.26 ± 0.06</td>
</tr>
<tr>
<td>GCA*</td>
<td>X, A</td>
<td>92.55 ± 0.03</td>
<td>87.82 ± 0.11</td>
<td>93.51 ± 0.06</td>
<td>78.68 ± 0.12</td>
</tr>
<tr>
<td>ProGCL-weight</td>
<td>X, A</td>
<td>93.30 ± 0.09</td>
<td>89.28 ± 0.15</td>
<td>93.51 ± 0.06</td>
<td>78.68 ± 0.12</td>
</tr>
<tr>
<td>ProGCL-mix</td>
<td>X, A</td>
<td>93.64 ± 0.13</td>
<td>89.55 ± 0.16</td>
<td>93.67 ± 0.12</td>
<td>78.45 ± 0.04</td>
</tr>
<tr>
<td>Supervised GCN</td>
<td>X, A, Y</td>
<td>92.42 ± 0.22</td>
<td>86.51 ± 0.54</td>
<td>93.03 ± 0.31</td>
<td>77.19 ± 0.12</td>
</tr>
<tr>
<td>Supervised GAT</td>
<td>X, A, Y</td>
<td>92.56 ± 0.35</td>
<td>86.93 ± 0.29</td>
<td>92.31 ± 0.24</td>
<td>77.65 ± 0.11</td>
</tr>
</tbody>
</table>

Table 2. Summary of the micro-averaged $F_1$ scores (± std) on inductive node classification.

<table>
<thead>
<tr>
<th>Method</th>
<th>Available Data</th>
<th>Flickr</th>
<th>Reddit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw features</td>
<td>X</td>
<td>20.3</td>
<td>58.5</td>
</tr>
<tr>
<td>DeepWalk</td>
<td>A</td>
<td>27.9</td>
<td>32.4</td>
</tr>
<tr>
<td>GraphSAGE</td>
<td>X, A</td>
<td>36.5</td>
<td>90.8</td>
</tr>
<tr>
<td>DGI</td>
<td>X, A</td>
<td>42.9 ± 0.1</td>
<td>94.0 ± 0.1</td>
</tr>
<tr>
<td>GMI</td>
<td>X, A</td>
<td>44.5 ± 0.2</td>
<td>94.8 ± 0.0</td>
</tr>
<tr>
<td>COLES-S²GC</td>
<td>X, A</td>
<td>46.8 ± 0.5</td>
<td>95.2 ± 0.3</td>
</tr>
<tr>
<td>GRACE</td>
<td>X, A</td>
<td>48.0 ± 0.1</td>
<td>94.2 ± 0.0</td>
</tr>
<tr>
<td>ProGCL-weight</td>
<td>X, A</td>
<td>49.2 ± 0.6</td>
<td>95.1 ± 0.2</td>
</tr>
<tr>
<td>ProGCL-mix</td>
<td>X, A</td>
<td>50.0 ± 0.3</td>
<td>95.6 ± 0.1</td>
</tr>
<tr>
<td>Supervised FastGCN</td>
<td>X, A, Y</td>
<td>48.1 ± 0.5</td>
<td>89.5 ± 1.2</td>
</tr>
<tr>
<td>Supervised GraphSAGE</td>
<td>X, A, Y</td>
<td>50.1 ± 1.3</td>
<td>92.1 ± 1.1</td>
</tr>
</tbody>
</table>

Table 3. Performance on the ogbn-arXiv measured in accuracy along with standard deviations. The results of baselines are taken from published reports. ‘-’ means that the results are unavailable.

<table>
<thead>
<tr>
<th>Method</th>
<th>Validation</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP</td>
<td>57.65 ± 0.12</td>
<td>55.50 ± 0.23</td>
</tr>
<tr>
<td>node2vec</td>
<td>71.29 ± 0.13</td>
<td>70.07 ± 0.13</td>
</tr>
<tr>
<td>Random-Init</td>
<td>69.90 ± 0.11</td>
<td>68.94 ± 0.15</td>
</tr>
<tr>
<td>DGI</td>
<td>71.26 ± 0.11</td>
<td>70.34 ± 0.16</td>
</tr>
<tr>
<td>GRACE-Subsampling</td>
<td>72.61 ± 0.15</td>
<td>71.51 ± 0.11</td>
</tr>
<tr>
<td>BGRG</td>
<td>72.53 ± 0.09</td>
<td>71.64 ± 0.12</td>
</tr>
<tr>
<td>COLES-S²GC</td>
<td>-</td>
<td>72.48 ± 0.25</td>
</tr>
<tr>
<td>ProGCL-weight</td>
<td>72.45 ± 0.21</td>
<td>72.18 ± 0.09</td>
</tr>
<tr>
<td>ProGCL-mix</td>
<td>72.82 ± 0.08</td>
<td>72.56 ± 0.20</td>
</tr>
<tr>
<td>Supervised GCN</td>
<td>73.00 ± 0.17</td>
<td>71.74 ± 0.29</td>
</tr>
</tbody>
</table>

4.4. Results on Large-Scale OGB Dataset

We conduct experiments on another large graph datasets ogbn-arxiv from OGB benchmark (Hu et al., 2020). We adopt GRACE with sub-sampling as our base model and a 3-layer GCN as the encoder following Hu et al. (2020). As can observed in Table 3, ProGCL outperforms other unsupervised baselines, which verifies that ProGCL can achieve good tradeoff between performance and complexity. We report the results on both validation and test sets, as is convention for this task since the dataset is split based on a chronological ordering.
4.5. Improving Various GCL Methods

Figure 6. The performance of ProGCL for another graph contrastive learning method MERIT (Jin et al., 2021).

In addition to GCA and GRACE, we also evaluate the performance of ProGCL on another GCL method MERIT. The results shown in Figure 6 demonstrate that ProGCL brings consistent improvements over the base method, which verifies that our ProGCL is readily pluggable into various negatives-based GCL methods to improve their performance.

4.6. Why ProGCL Can Alleviate the Bias?

To verify that our ProGCL can alleviate the sampling bias, we plot estimated probability histograms of negatives in Figure 7. Compared with similarity, the estimated probability serves as a more discriminative measure to distinguish true and false negatives, which can help us select hard and true negatives together with similarity as in Eq. (10).

4.7. Ablation Study

In this section, we replace or remove various parts of ProGCL to study the impact of each component.

Table 4. Comparison between BMM and GMM.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Amazon-Photo</th>
<th>Amazon-Computers</th>
<th>Coauthor-CS</th>
</tr>
</thead>
<tbody>
<tr>
<td>GMM</td>
<td>weight mix</td>
<td>weight mix</td>
<td>weight mix</td>
</tr>
<tr>
<td>BMM</td>
<td>92.71</td>
<td>92.83</td>
<td>89.35</td>
</tr>
</tbody>
</table>

BMM vs. GMM. We replace BMM in our ProGCL with GMM and report the performance in Table 4. BMM consistently outperforms GMM in both weight and mix schemes. The reason is that BMM can fit the negatives distribution better than GMM as shown in Figure 4.

4.8. Hyperparameters Sensitivity Analysis

Here we study the number of most hard negatives $N'$ and synthetic negatives $m$ of ProGCL-mix. The results shown in Figure 8 illustrate that more samples mixing bring consistent performance gains in general. However, oversized $N'$ demonstrated no significant advantages in both accuracy and efficiency.
Figure 8. Accuracy when varying $N'$ (x-axis) and $m$ on Amazon-Photo. We study more hyperparameters in the appendix.

5. Conclusions

In this paper, we explain why existing hard negative mining methods can not work well in GCL and contrapuntally introduce BMM to estimate the probability of a negative being true one. Also, we devise two schemes to further boost GCL. Interesting directions of future work include (1) applying GCL to more real-world tasks including social analysis and drug discovery (Sun et al., 2021; Xia et al., 2022c); (2) exploring the theoretical explanations for the immense success of contrastive learning.

6. Acknowledgements

This work is supported in part by the Science and Technology Innovation 2030 - Major Project (No. 2021ZD0150100) and National Natural Science Foundation of China (No. U21A20427).

References


Appendix of ProGCL

A. Datasets

Table 7. Statistics of datasets used in experiments.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Task</th>
<th>Nodes</th>
<th>Edges</th>
<th>Features</th>
<th>Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon-Photo</td>
<td>Transductive</td>
<td>7,650</td>
<td>119,081</td>
<td>745</td>
<td>8</td>
</tr>
<tr>
<td>Amazon-Computers</td>
<td>Transductive</td>
<td>13,752</td>
<td>245,861</td>
<td>767</td>
<td>10</td>
</tr>
<tr>
<td>Coauthor-CS</td>
<td>Transductive</td>
<td>18,333</td>
<td>81,894</td>
<td>6,805</td>
<td>15</td>
</tr>
<tr>
<td>Wiki-CS</td>
<td>Transductive</td>
<td>11,701</td>
<td>216,123</td>
<td>300</td>
<td>10</td>
</tr>
<tr>
<td>Flickr</td>
<td>Inductive</td>
<td>89,250</td>
<td>899,756</td>
<td>500</td>
<td>7</td>
</tr>
<tr>
<td>Reddit</td>
<td>Inductive</td>
<td>231,443</td>
<td>11,606,919</td>
<td>602</td>
<td>41</td>
</tr>
<tr>
<td>Ogbn-arXiv</td>
<td>Inductive</td>
<td>169,343</td>
<td>1,166,243</td>
<td>128</td>
<td>40</td>
</tr>
</tbody>
</table>

We introduce the datasets used in our experiments as follows:

- **WikiCS** (Mernyei & Cangea, 2020) is a reference network constructed based on Wikipedia. The nodes correspond to articles about computer science and edges are hyperlinks between the articles. Nodes are labeled with ten classes each representing a branch of the field. Node features are calculated as the average of pre-trained GloVe (Pennington et al., 2014) word embeddings of words in each article.

- **Amazon-Computers** and **Amazon-Photo** (Shchur et al., 2018) are two networks of co-purchase relationships constructed from Amazon, where nodes are goods and two goods are connected when they are frequently bought together. Each node has a sparse bag-of-words feature encoding product reviews and is labeled with its category.

- **Coauthor-CS** (Shchur et al., 2018) is an academic network, which contain co-authorship graphs based on the Microsoft Academic Graph from the KDD Cup 2016 challenge. In the graph, nodes represent authors and edges indicate co-authorship relationships; that is, two nodes are connected if they have co-authored a paper. Each node has a sparse bag-of-words feature based on paper keywords of the author. The label of an author corresponds to their most active research field.

- **Flickr** originates from NUS-wide\(^1\). The SNAP website\(^2\) collected Flickr data from four different sources including NUS-wide, and generated an undirected graph. One node in the graph represents one image uploaded to Flickr. If two images share some common properties (e.g., same geographic location, same gallery, comments by the same user, etc.), there is an edge between the nodes of these two images. The node features are the 500-dimensional bag-of-word representation of the images provided by NUS-wide. For labels, we scan over the 81 tags of each image and manually merged them to 7 classes. Each image belongs to one of the 7 classes.

- **Reddit** (Hamilton et al., 2017) contains Reddit posts created in September 2014, where posts belong to different communities (subreddit). In the dataset, nodes correspond to posts, and edges connect posts if the same user has commented on both. Node features are constructed from post title, content, and comments, using off-the-shelf GloVe word embeddings, along with other metrics such as post score and the number of comments.

- **Ogbn-arXiv** is a large-scale graph from the Open Graph Benchmark (Hu et al., 2020). This is another citation network, where nodes represent CS papers on arXiv indexed by the Microsoft Academic Graph (Sinha et al., 2015). In our experiments, we symmetrize this graph and thus there is an edge between any pair of nodes if one paper has cited the

---

\(^1\)https://lms.comp.nus.edu.sg/research/NUS-WIDE.htm

\(^2\)https://snap.stanford.edu/data/web-flickr.html
other. Papers are classified into 40 classes based on arXiv subject area. The node features are computed as the average word-embedding of all words in the paper, where the embeddings are computed using a skip-gram model (Mikolov et al., 2013) over the entire corpus.

B. More Similarity Histograms of Negativess in SimCLR (CL) and GCA (GCL).

![Figure 9. More similarity histograms of negative samples in and GCL.](image_url)

As demonstrated in Figure 9, the negatives distribution over similarity varies significantly across GCL and CL. These phenomena provide explanations for the failure of existing techniques that emphasize hard negatives in GCL. More specifically, they regard the negatives that are most similar to anchor points as hard ones, which is feasible in CL. However, for graph-structured data (see Figure 9(a), Figure 9(b) and Figure 9(c)), many “hard” ones are false negatives indeed, which will undesirably push away the semantically similar samples.

C. More Experimental Details

C.1. Transductive learning

We adopt a two-layer GCN (Kipf & Welling, 2016a) as the encoder for transductive learning following previous works (Velickovic et al., 2019; Zhu et al., 2020). We can describe the architecture of the encoder as,

\[
GC_i(X, A) = \sigma \left( \hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}} X W_i \right),
\]

\[
f(X, A) = GC_2(GC_1(X, A), A),
\]

where \( \hat{A} = A + I \) is the adjacency matrix with self-loops and \( \hat{D} = \sum_i \hat{A}_i \) is the degree matrix, \( \sigma(\cdot) \) is a nonlinear activation function, \( W_i \) is the learnable weight matrix.

C.2. Inductive learning

Considering the large scale of some graph datasets, we adopt a three-layer GraphSAGE-GCN with residual connections (He et al., 2016) as the encoder following DGI (Velickovic et al., 2019) and GRACE (Zhu et al., 2020). The architecture of the encoder can be formulated as,

\[
\hat{M}P_i(X, A) = \sigma \left( \hat{D}^{-1} \hat{A}X; X \right) W_i,
\]

\[
f(X, A) = \hat{M}P_3 \left( \hat{M}P_2 \left( \hat{M}P_1(X, A), A \right), A \right).
\]

D. Hyper-parameters Analysis

Following GRACE (Zhu et al., 2020) and GCA (Zhu et al., 2021c), we initialize the model parameters with Glorot initialization (Glorot & Bengio, 2010) and train the model using Adam SGD optimizer for all datasets. The \( \ell_2 \) weight decay factor is set as \( 10^{-5} \) and the dropout rate is set to zero. The parameters which control the sampling process of two views are the
Table 8. Hyperparameters specifications. “Starting epoch” $E$ refers to the epoch for fitting BMM; Initial weight $w_{\text{init}}$ is the initial weight for false negative component of mixture distribution; “Iterations” is the iterations $I$ of EM algorithm.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$\tau$</th>
<th>learning rate</th>
<th>Training epochs</th>
<th>Hidden dimension</th>
<th>Activation function</th>
<th>Starting epoch</th>
<th>Initial weight</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon-Photo</td>
<td>0.3</td>
<td>0.01</td>
<td>2500</td>
<td>128</td>
<td>RRelu</td>
<td>400</td>
<td>0.15</td>
<td>10</td>
</tr>
<tr>
<td>Amazon-Computers</td>
<td>0.2</td>
<td>0.01</td>
<td>2000</td>
<td>128</td>
<td>RRelu</td>
<td>400</td>
<td>0.05</td>
<td>10</td>
</tr>
<tr>
<td>Coauthor-CS</td>
<td>0.2</td>
<td>0.0001</td>
<td>1000</td>
<td>256</td>
<td>RRelu</td>
<td>400</td>
<td>0.05</td>
<td>10</td>
</tr>
<tr>
<td>Wiki-CS</td>
<td>0.4</td>
<td>0.01</td>
<td>4000</td>
<td>256</td>
<td>PRelu</td>
<td>50</td>
<td>0.05</td>
<td>10</td>
</tr>
<tr>
<td>Reddit</td>
<td>0.4</td>
<td>0.0001</td>
<td>80</td>
<td>512</td>
<td>ELU</td>
<td>40</td>
<td>0.01</td>
<td>10</td>
</tr>
<tr>
<td>Flickr</td>
<td>0.4</td>
<td>0.0001</td>
<td>80</td>
<td>512</td>
<td>ELU</td>
<td>20</td>
<td>0.15</td>
<td>10</td>
</tr>
<tr>
<td>Ogbn-arXiv</td>
<td>0.4</td>
<td>0.0001</td>
<td>100</td>
<td>512</td>
<td>ELU</td>
<td>20</td>
<td>0.02</td>
<td>10</td>
</tr>
</tbody>
</table>

![Figure 10](image_url) Accuracy when varying $E$, $w_{\text{init}}$, $I$ and $M'$ ($M = NM'$) for BMM.

same as GRACE and GCA. Other hyper-parameters of ProGCL can be seen in Table 8. For transductive task, the two hyper-parameters were chosen in a grid $E \in \{50, 100, 200, 300, 400, 600, 800\}$ and $w_{\text{init}} \in \{0.01, 0.05, 0.10, 0.15, 0.20, 0.25\}$.

We further study the influence of $E$, $w_{\text{init}}$, $I$ and $M$ in Figure 10. Firstly, the accuracy varies significantly across various starting epoch $E$ in Amazon-Photo while varies slightly in Coauthor-CS, which illustrates the importance of tuning $E$ varies across datasets. However, generally speaking, ProGCL's performance does not see sharp drop when varying $E$. This validates that BMM is flexible enough to fit various distributions of different epochs. As shown in figure 10(b), the initial weight $w_{\text{init}}$ does not influence much, which illustrates that BMM can split the two component well to an ideal ratio even if the initial weight is far from intuitive one (the reciprocal of classes number). As shown in figure 10(c), we can observe minor improvement when we iterate EM algorithm more times. However, this will introduce more computational overhead and thus we set $I = 10$ in all experiments for convenience. As shown in Figure 9(c), sampling more similarities for fitting the BMM can bring minor improvements, however, it will introduce much more computational overhead. In our experiments, we only sample $M' = 100$ samples for each anchor point. Thus, the number of total selected samples $M = NM'$.

E. Pseudo Codes of Inductive Learning.

Different from transductive learning, it is not feasible to compute all the pairwise similarities for large-scale graph datasets. Thus, we extend ProGCL to inductive setting. The algorithm of ProGCL for inductive learning can be seen as follows. The equations mentioned in the algorithm can be seen in the main text.
Algorithm 2 ProGCL-weight & ProGCL-mix (Inductive)

**Input:** $T, G, f, g, N$, normalized cosine similarity $s$, epoch for fitting BMM $E$, mode (‘weight’ or ‘mix’), empty list $\mathcal{P}$ for storing the estimated probabilities, Batchsize $B$.

**for** epoch = 0, 1, 2, ... **do**

$k = 0$;

**for** each mini-batch do

$G_s(V_s, E_s) \leftarrow$ Sampled sub-graph of $G$ with sampling rules of GraphSAGE;

Draw two augmentation functions $t \sim T, t' \sim T$;

$\tilde{G}^{(1)}_s = t(G_s), \tilde{G}^{(2)}_s = t'(G_s)$;

$U_s = f(\tilde{G}^{(1)}_s), V_s = f(\tilde{G}^{(2)}_s)$;

**for** all $u_i \in U_s$ and $v_i \in V_s$ do

$s_{ij} = s(g(u_i), g(v_i))$;

if epoch $= E$ then

Compute $\mathcal{M}_{i,j} = p(c_t | s_{ij})$ with Eq. (4) to Eq. (9);

$\mathcal{P}.append(M)$.

end if

if epoch $\geq E$ then

if mode $= \text{‘weight’}$ then

Compute $\mathcal{J}_w$ with Eq. (10) to Eq. (12); ($\mathcal{P}[k]$ as the estimated probabilities for $k$-th minibatch)

Update the parameters of $f, g$ with $\mathcal{J}_w$.

end if

if mode $= \text{‘mix’}$ then

Compute $\mathcal{J}_m$ with Eq. (13) to Eq. (16); ($\mathcal{P}[k]$ as the estimated probabilities for $k$-th minibatch)

Update the parameters of $f, g$ with $\mathcal{J}_m$.

end if

else if epoch $< E$ then

Compute $\mathcal{J}$ with Eq. (1) to Eq. (2)

Update the parameters of $f, g$ with $\mathcal{J}$.

end if

end for

$k = k + 1$.

end for

end for

**Output:** $f, g$. 

---