Learning from the Dark: Boosting Graph Convolutional Neural Networks with Diverse Negative Samples

Wei Duan[1], **Junyu Xuan**[1], **Maoying Qiao** [2], **Jie Lu**[1]

[1] The Australian Artificial Intelligence Institute (AAII), University of Technology Sydney
[2] Australian Catholic University
Wei.Duan@student.uts.edu.au, Junyu.Xuan@uts.edu.au, maoying.qiao@acu.edu.au, Jie.Lu@uts.edu.au

Abstract

Graph Convolutional Neural Networks (GCNs) has been generally accepted to be an effective tool for node representations learning. An interesting way to understand GCNs is to think of them as a message passing mechanism where each node updates its representation by accepting information from its neighbours (also known as positive samples). However, beyond these neighbouring nodes, graphs have a large, dark, all-but forgotten world in which we find the non-neighbouring nodes (negative samples). In this paper, we show that this great dark world holds a substantial amount of information that might be useful for representation learning. Most specifically, it can provide negative information about the node representations. Our overall idea is to select appropriate negative samples for each node and incorporate the negative information contained in these samples into the representation updates. Moreover, we show that the process of selecting the negative samples is not trivial. Our theme therefore begins by describing the criteria for a good negative sample, followed by a determinantal point process algorithm for efficiently obtaining such samples. A GCN, boosted by diverse negative samples, then jointly considers the positive and negative information when passing messages. Experimental evaluations show that this idea not only improves the overall performance of standard representation learning but also significantly alleviates over-smoothing problems.

Introduction

Graphs are powerful structures for modelling specific kinds of data such as molecules, social networks, citation networks, traffic networks, etc. (Chakrabarti and Faloutsos 2006). However, the representation power of graphs is not a free lunch; it brings with it issues of incompatibility with some of the strongest and most popular deep learning algorithms, as these can often only handle regular data structures like vectors or arrays. Hence, to lend learning power to these great tools of representation, graph neural networks (GNNs) have emerged as a congruent deep learning architecture (Wu et al. 2020). Such has been their success that, today, there are many different GNN variants, each adapted for a specific task – for instance, graph sequence neural networks (Li et al. 2016), graph convolutional neural networks (Kipf and Welling 2017), and spatio-temporal graph convolutional networks (Yu, Yin, and Zhu 2018).

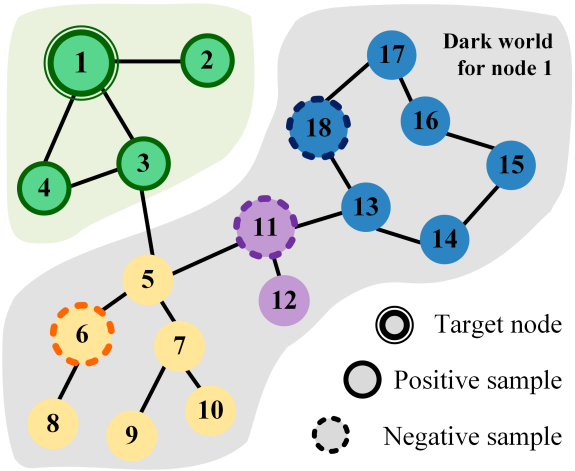

Figure 1: Illustration of the motivation of this work, including the dark world (gray shadow), different semantic clusters (green, yellow, purple, blue nodes), positive samples (nodes 2, 3, 4) and selected diverse negative samples (nodes 6, 11, and 18) of a given node (node 1).

Among all the varieties of GNNs, graph convolutional neural networks (GCN) (Kipf and Welling 2017) is a simple but representative and salient one, which introduces the concept of convolution to GNNs that means to share weights for nodes within a layer. An easy and intuitive way to understand GCNs is to think of them as a message passing mechanism (Geerts, Mazowiecki, and Pérez 2021) where each node accepts information from its neighbouring nodes to update its representation. The basic idea is that the representations of nodes with edge between them should be positively correlated. Hence, the neighboring nodes are also named as positive samples[1]. This message-passing mechanism is highly effective in many scenarios, but it does lead to an annoying over-smoothing problem where the node representations become more and more similar as the number of layers of the graph neural network increases. This is hardly sur-

[1]Hereafter we refer to neighbouring nodes and positive samples interchangeably.

prising when each node only updates its representation according to its neighbours. Yet, beyond these observed edges, there is a dark world that could provide diverse and useful information to the representation updates and help to overcome the over-smoothing problem at the same time, as illustrated in Fig. 1. The reasoning is this: two nodes without edge between them should have different representations, so we can understand this as a negative link. However, in contrast to the commonly used positive links, these negative links are rarely used in the existing various GCNs.

Selecting appropriate negative samples in a graph is not trivial. To our knowledge, only three studies have explored procedures to this end. The first is Kim and Oh (Kim and Oh 2021), who propose the natural and easy approach of uniformly randomly selecting some negative samples from non-neighbouring nodes. However, as we all know, a large graph is normally has the small-world property (Watts and Strogatz 1998) which means the graph will tend to contain some clusters. Moreover, nodes in the same cluster will tend to have similar representations while nodes within different clusters will tend to have different representations, as illustrated in Fig. 1. Hence, under uniform random selection, the large clusters will overwhelm the smaller ones, and the final converged node representations will be short on information contained in the small clusters. Of the other two methods, one is based on Monte Carlo chains (Yang et al. 2020) and the other on personalised PageRank (Ying et al. 2018), and neither covers diverse information as well. Further, the selected negative samples should not be redundant; each of them should not be overlapping and hold distinct information. Hence, as a definition, a good negative sample should *contribute negative information to the give node contrast to its positive samples and include as much information as possible to reflect the variety of the dark world.*

In this paper, we propose a graph convolutional neural network boosted by diverse negative samples selected through a determinant point process (DPP). DPP is special point process for defining a probability distribution on a set where more diverse subset has a higher probability (Hough et al. 2009). Our idea is to find diverse negative samples for a given node by firstly defining a DPP-based distribution on diverse subsets of all non-neighbouring nodes of this node, and then outputting a diverse subset from this distribution. The number of subsets is limited because the samples are mutually exclusive. However, when applying this algorithm to large graphs, the computational cost can be as high as $O(N^3)$, where N is the number of non-neighbouring nodes. Therefore, we propose a method based on a depth-first search (DFS) that collects diverse negative samples sequentially along the search path for a node. The motivation is that a DFS path is able to go through all different clusters in the graph and, therefore, the local samples collected will form a good, diverse approximation of all the information in the dark world. Further, the computational cost is approximately $O(\overline{\text{pa}} \cdot \overline{\text{deg}}^3)$ where $\overline{\text{pa}} \ll N$ is the path length (normally smaller than the diameter of the graph) and $\overline{\text{deg}} \ll N$ is the average degree of the graph. As an example, consider a Cora graph (Sen et al. 2008) with 3,327 nodes, $\overline{\text{pa}} = 19$, and $\overline{\text{deg}} = 3.9$. Thus, $O(N^3) = 3.6e10 \gg 1,127 = O(\overline{\text{pa}} \cdot \overline{\text{deg}}^3)$.

Symbol	Meaning
G	a graph
G_n	the node set of graph G
$\mathcal{Y}$	a ground set
Y	a node subset
k	the number of negative samples of a given node in a graph
$\mathcal{N}(i)$	neighbours (positive samples) of node i
$\overline{\mathcal{N}}(i)$	negative samples of node i
$x_i^{(l)}$	the representation of node i at layer l
$\deg(i)$	the degree of node i
$\mathbb{L}$	the $\mathbb{L}$-ensemble of DPP
λ	the eigenvalues of $\mathbb{L}$
$e_k^{\lvert\mathcal{Y}\rvert}$	the k^{th} elementary symmetric polynomial on eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_{\lvert\mathcal{Y}\rvert}$ of $\mathbb{L}$
$\boldsymbol{v}$	the eigenvectors of the $\mathbb{L}$-ensemble
V	the set $\boldsymbol{v}$

Table 1: Key notations

In evaluating these ideas, we conducted empirical experiments on different tasks, including node classification and over-smoothing tests. The results show that our approach produces superior results.

Thus, the contributions of this study include:

- A new negative sampling method based on a DPP that selects diverse negative samples to better represent the dark information;
- A DFS-based negative sampling method that sequentially collects locally diverse negative samples along the DFS path to greatly improve computational efficiency;
- We are the first to fuse negative samples into the graph convolution, yielding a new GCN boosted by diverse negative samples (D2GCN) to improve the quality of node representations and alleviate the over-smoothing problem.

Preliminaries

This section briefly introduces the basic concepts of graph convolutional neural networks, message passing, and determinantal point process.

Graph Convolutional Neural Networks (GCNs)

GCNs introduce traditional convolution from classical neural networks to graph neural networks. Like a message-passing mechanism, the representation of a node is updated using its neighbours' representations through

$$x_i^{(l)} = \sum_{j \in \mathcal{N}_i \cup \{i\}} \frac{1}{\sqrt{\deg(i)} \cdot \sqrt{\deg(j)}} \left(\Theta^{(l)} \cdot x_j^{(l-1)}\right) \quad (1)$$

where $x_i^{(l)}$ are the representation of node i at layer l, $\mathcal{N}(i)$ is the neighbours of node i (i.e., positive samples), $\deg(i)$ is

the degree of node i, and $\Theta^{(l)}$ is a feature transition matrix. The goal of this equation is to aggregate (sum) the weighted representations of all neighbours. Note that although we use GCNs to demonstrate our approach throughout the paper, the same ideas can be easily applied to other variants as well.

Determinantal Point Processes (DPP)

Given a ground set $\mathcal{Y} = \{1, 2, \ldots, |\mathcal{Y}|\}$, a determinantal point process $\mathcal{P}$ is a probability measure on all possible subsets of $\mathcal{Y}$ with size $2^{|\mathcal{Y}|}$. For every $Y \subseteq \mathcal{Y}$, a DPP (Hough et al. 2009) defined via an $\mathbb{L}$-ensemble is formulated as

$$\mathcal{P}_{\mathbb{L}}(Y) = \frac{\det(\mathbb{L}_Y)}{\det(\mathbb{L} + I)} \tag{2}$$

where $\det(\cdot)$ denotes the determinant of a given matrix, $\mathbb{L}$ is a real and symmetric $|\mathcal{Y}| \times |\mathcal{Y}|$ matrix indexed by the elements of $\mathcal{Y}$, and $\det(\mathbb{L} + I)$ is a normalisation term that is constant once the ground dataset $\mathcal{Y}$ is fixed. Given a Gram decomposition $\mathbb{L}_Y = \bar{\mathbb{L}}^T\bar{\mathbb{L}}$, a determinantal operator can be interpreted geometrically as

$$\mathcal{P}_{\mathbb{L}}(Y) \propto \det(\mathbb{L}_Y) = \text{vol}^2\left(\{\bar{\mathbb{L}}_i\}_{i \in Y}\right) \tag{3}$$

where the right hand side is the squared volume of the parallelepiped spanned by the columns in $\bar{\mathbb{L}}$ corresponding to elements in Y. Intuitively, to get a parallelepiped of greater volume, the columns should be as repulsive as possible to each other. Hence, DPP assigns a higher probability to a subset of Y whose elements span a greater volume.

One important variant of DPP is k-DPP (Kulesza and Taskar 2012). k-DPP measures only k-sized subsets of $\mathcal{Y}$ rather than all of them including an empty subset. It is formally defined as

$$\mathcal{P}_{\mathbb{L}}(Y) = \frac{\det(\mathbb{L}_Y)}{e_k^{|\mathcal{Y}|}} \tag{4}$$

with the cardinality of the subset Y being a fixed size k, i.e., $|Y| = k$. $e_k^{|\mathcal{Y}|}$ is the k^{th} elementary symmetric polynomial on eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_{|\mathcal{Y}|}$ of $\mathbb{L}$, i.e., $e_k(\lambda_1, \lambda_2, \ldots, \lambda_{|\mathcal{Y}|})$.

Both DPP and k-DPP can be used to sample a diverse subset, and both have been well studied in the machine learning area (Kulesza and Taskar 2012). The popularity of DPP (and also of k-DPP) for modeling diversity is because of its great modelling power.

Related Work

GCN and Its Variants

Motivated by the achievements of convolutional neural networks (CNNs), many GNNs approaches have been actively studied to model graph data. These are classified into two types, spectral-based and spatial-based. (Bruna et al. 2014) first created a graph convolution based on spectral graph theory. Although their paper was conceptually important, it had major computational flaws, which prevented it from being a genuinely useful tool. Since then, a growing number of enhancements, extensions, and approximations of spectral-based GCNs have been made to overcome these flaws. Based on these, (Kipf and Welling 2017) proposed GCNs, which is a localized first-order approximation of spectral graph convolutions as a generalised method for semi-supervised learning on graph-structured data. Their model acquires implicit representations, encodes the region graph structure and node attributes, and expands linearly in terms of the number of graph edges.

A representative work in a spatial-based way is GraphSAGE (Hamilton, Ying, and Leskovec 2017), which is a general framework for generating node embedding by sampling and aggregating features from neighbourhood of a node. In order to theoretically analyze the representational power of GNNs, (Xu et al. 2019) formally characterised how expressive different GNNs variants are at learning to represent different graph structures based on the graph isomorphism test (Weisfeiler and Leman 1968). They proposed that, since modern GNNs follow a neighbourhood aggregation strategy, the network at the k-th layer can be summarised formally in two steps AGGREGATE and COMBINE. In addition, they further proposed GINs, which can distinguish different graph structures and capture dependencies between graph structures to achieve better classification results (Xu et al. 2019).

All the above GNNs can be considered to use positive sampling when generating new node feature vectors, because the neighbour aggregation on $j \in \mathcal{N}(i)$ leads to a high correlation between the central node and its neighbours. Extremely few studies use negative sampling with a GNN. The only three works (Ying et al. 2018; Kim and Oh 2021; Yang et al. 2020) that do use it do not satisfy the our criteria of good negative samples: good negative samples should include as many information of the dark world as possible and, at the same time, without much overlapping and redundant information. Moreover, the above approaches only use the results of negative sampling in the loss function, while the direct application of to convolution operations remaining unexplored.

DPP and Its Applications

Determinantal point processes (DPPs) (Hough et al. 2009) are statistical models and provide probability measures over every configuration of subsets on data points. DPPs were first introduced to machine learning area by (Kulesza and Taskar 2012), and they have since been extended to include closed-form normalisation, marginalisation (Kulesza and Taskar 2012), sampling (Kang 2013), dual representation, maximising a posterior (MAP) (Gillenwater, Kulesza, and Taskar 2012b)) and parameter learning (Affandi et al. 2014; Gillenwater et al. 2014), and its structural (Gillenwater, Kulesza, and Taskar 2012a) and Markov (Affandi, Kulesza, and Fox 2012) variants to name just two. This repulsive characteristic has been successfully applied to prior modelling in a variety of scenarios, such as clustering (Kang 2013), inhibition in neural spiking data (Snoek, Zemel, and Adams 2013), sequential labelling (Qiao et al. 2015), document summarisation, video summarisation, tweet timeline generation, and so on. However, to the best of our knowledge, DPP has not been used for negative sampling with GNNs.

Algorithm 1: Diverse negative sampling

Input: A graph G
Output: $\overline{\mathcal{N}}(i)$ for all $i \in G$

Let $\overline{\mathcal{N}} = \mathbf{0}$.
for each node i **do**
  Compute $\mathbb{L}_{G\setminus i}$ using Eq. (5);
  Define a distribution on all possible subsets $\mathcal{P}_{\mathbb{L}}(Y_{G\setminus i})$ using Eq. (6);
  Obtain a sample of $\mathcal{P}_{\mathbb{L}}(Y_{G\setminus i})$ using Algorithm 8 in (Kulesza and Taskar 2012);
  Save nodes in the sample as $\overline{\mathcal{N}}(i)$;
end for
return $\overline{\mathcal{N}}$

Proposed Model

This section first introduces a method for obtaining good negative samples for a given node, including two negative sampling algorithms. We then integrate the algorithms with a GCN to obtain a new graph convolutional neural network boosted by diverse negative samples (D2GCN).

DPP-Based Negative Sampling

Given a node, we believe that its good negative samples should include as much information about the dark world as possible and, at the same time, without much overlap and redundancy. The repulsive property of DPP inspired us to use it to select negative samples. However, applying DPP to different scenarios is not trivial; modifications are required to make it work. Hence, for a given node i in a graph G, we need to first define a $\mathbb{L}$-ensemble for our problem:

$$\mathbb{L}_{G_n\setminus i}(j, j') = \exp\left(\cos\left(x_j, x_{j'}\right) - 1\right), \tag{5}$$

where $G_n \setminus i$ denotes the node set of G_n excluding node i, j and j' are two nodes within $G_n \setminus i$, and $\cos(\cdot,\cdot)$ is the cosine similarity between two node representations x. The node feature x is used here because, when defining the distance between nodes, nodes with similar information are expected to be further apart. x_j is expected to encode the information of node j in terms of both features and network structure. Feature information may dominate at the beginning but both types of information will be balanced after a number of training steps. The reason we chose cosine similarity is because the following prediction (label and link) is normally based on cosine similarity. We used only x to simplify the explanation. However, it would be straightforward to include more information here, like the network distance between nodes j and j', the similarity with node i or node degree of each node.

With this $\mathbb{L}$-ensemble, we can obtain a distribution on all possible subsets of all nodes in the graph except for i,

$$\mathcal{P}_{\mathbb{L}}(Y_{G_n\setminus i}) = \frac{\det(\mathbb{L}_{Y_{G_n\setminus i}})}{\det(\mathbb{L}_{G_n\setminus i} + I)} \tag{6}$$

where $Y_{G_n\setminus i}$ is a set of all subsets of $G_n \setminus i$ and $\det(\cdot)$ is the matrix determinant operator. Comparing other ordinary probability distributions with same support, this distribution has a nice and unique property that the more diverse the subsets, the higher their probability values, the easier it is to obtain a diverse set from this distribution by sampling. To ensure the scale of negative samples is similar to the positive samples, we use k-DPP to fix the number of negative samples as $k = |\mathcal{N}_i| + 1$. Here, we use a sampling method based on eigendecomposition (Hough et al. 2006; Kulesza and Taskar 2012). Eq. (6) can be rewritten as the k-DPP distribution in terms of the corresponding DPP

$$\mathcal{P}^k_{\mathbb{L}}(Y_{G_n\setminus i}) = \frac{1}{e_k^{|G_n\setminus i|}} \det(\mathbb{L}_{G_n\setminus i} + I)\mathcal{P}_{\mathbb{L}}(Y_{G_n\setminus i}) \tag{7}$$

whenever $|Y_{G_n\setminus i}| = k$ and $e_k^{|G_n\setminus i|}$ denotes the k-th elementary symmetric polynomial. Following the (Kulesza and Taskar 2012), Eq. (7) can be decomposed into elementary parts

$$\mathcal{P}^k_{\mathbb{L}}(Y_{G_n\setminus i}) = \frac{1}{e_k^{|G_n\setminus i|}} \sum_{|J|=k} \mathcal{P}^{V_J}(Y_{G_n\setminus i}) \prod_{m\in J} \lambda_m \tag{8}$$

where $V_{Y_{G_n\setminus i}}$ denotes the set $\{\boldsymbol{v}_m\}_{m\in Y_{G_n\setminus i}}$ and $\boldsymbol{v}_m$ and λ_m are the eigenvectors and eigenvalues of the $\mathbb{L}$-ensemble, respectively. Based on Eq. (8), the complete process of sampling from k-DPP is in Algorithm 8 in (Kulesza and Taskar 2012).

Note that comparing with a sample, the mode of this distribution is a more rigorous output (Gillenwater, Kulesza, and Taskar 2012b), but it is usually with an unbearable complexity, so we use a sample rather than mode here. The experimental evaluation have shown that the sample has been able to achieve satisfactory results. Algorithm 1 shows a diverse negative sampling method based on DPP, but its computation cost is above the standard for a DPP on $G_n \setminus i$ is $O(|G_n \setminus i|^3)$ for each node. This totals an exorbitant $O(|G_n \setminus i|^4)$ for all nodes! Although Algorithm 1 is able to explore the whole dark world and find the best diverse negative samples, the large number of candidates makes it an impractical solution for even a moderately-sized graph. Hence, we propose the approximate heuristic method below.

Our belief is that the good negative samples are the ones with different semantics and complete knowledge of the whole graph, such as nodes 6, 11, and 18 in Fig. 2. In this figure, we hope the selected negative samples could each belong to the different 'cluster' and all 'clusters' are represented by the negative samples. Hence, given a node i, we use a depth-first-search (DFS) method to build a fixed-length path from node i to the other nodes. We then collect the first-order neighbours of the nodes on this path to form a candidate set. Finally, we select diverse negative samples from this candidate set using the same DPP idea as outlined above. This DFS-based method can be considered as using a local diverse negative samples to approximate the global diverse negative samples. The reason is that DFS has the capability of breaking through the 'cluster' of the current node i to reach the semantics of other 'clusters'. So the first-order neighbours of nodes on the path are able to supply sufficient information from many 'clusters' but at a much smaller size. Although only first-order neighbours are collected here, collecting a higher order may further increase approximation

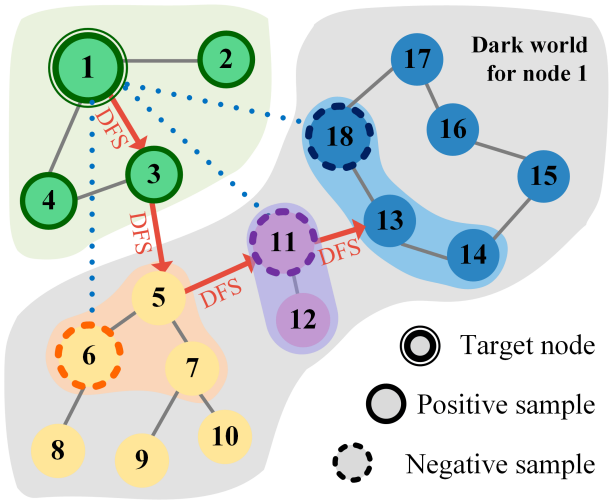

Figure 2: The concept of DPP-based negative sampling. The target node is Node 1. Nodes 2, 3 and 4 are positive samples. Nodes 5-18 are the dark world of Node 1. The 4-length DFS path of Node 1 is $\{3, 5, 11, 13\}$, where $\{5, 11, 13\}$ are the central nodes on the path in the dark world. With their first-order neighbouring nodes, they form the candidate set of DPPs, i.e.$\{5, 6, 7, 11, 12, 13, 14, 18\}$. The selected negative samples from this set are 6, 11, and 18, which can be seen as virtual negative links to Node 1.

Algorithm 2: DFS-based diverse negative sampling

Input: A graph G
Output: $\overline{\mathcal{N}}(i)$ for all $i \in G$

Let $\overline{\mathcal{N}} = \mathbf{0}$;
for each node i **do**
  Build a DFS path pa_i in $G \setminus i$;
  Let $C_i = []$;
  for each node $j \in pa_i$ **do**
    Collect first-order neighbors $\mathcal{N}(j)$ of j;
    Expand $C_i = [C_i, \mathcal{N}(j)]$;
  end for
  Compute $\mathbb{L}_{C_i}$ using Eq. (5);
  Define a distribution on all possible subsets $\mathcal{P}_{\mathbb{L}}(Y_{C_i})$ using Eq. (6);
  Obtain a sample of $\mathcal{P}_{\mathbb{L}}(Y_{C_i})$;
  Save nodes in the sample as $\overline{\mathcal{N}}(i)$.
end for
return $\overline{\mathcal{N}}$

performance; however, it would also increase the computational cost. In terms of path length, we found that performance is sufficiently good when the path length is set to around a quarter of the graph's diameter. The full procedure is summarised in Algorithm 2.

Here, the overall compuational cost is $O(N \cdot |\overline{\text{pa}}| \cdot \overline{\text{deg}}^3)$ where $|\overline{\text{pa}}| \ll N$ is the path length (normally smaller than the diameter of the graph) and $\overline{\text{deg}} \ll N$ is the average degree of the graph. That is much smaller than $O(|G_n \setminus i|^4)$

Algorithm 3: D2GCN

Input: A graph G
Output: $x_i^{(L)}$ for all $i \in G$

Build DFS path for all nodes;
for each level l **do**
  for each node i in l **do**
    Find negative samples for i using Algorithm 2;
    Update node representation $x_i{}^{(l)}$ using Eq. (9);
  end for
end for
return $x^{(L)}$

GCN Boosted by Diverse Negative Samples

The classical GCN is only based on positive samples as shown in Eq. (1), which will inevitably lead to over-smoothing issues. With the negative samples from Algorithms 1 or 2, we propose the following new graph convolutional operation as

$$x_i{}^{(l)} = \sum_{j \in \mathcal{N}_i \cup \{i\}} \frac{1}{\sqrt{\deg(i)} \cdot \sqrt{\deg(j)}} \left(\Theta^{(l)} \cdot x_j^{(l-1)}\right) - \omega \sum_{\bar{j} \in \overline{\mathcal{N}_i}} \frac{1}{\sqrt{\deg(i)} \cdot \sqrt{\deg(\bar{j})}} \left(\Theta^{(l)} \cdot x_{\bar{j}^{(l-1)}}\right) \tag{9}$$

where $\overline{\mathcal{N}_i}$ is the negative samples of node i and ω is a hyper-parameter to balance the contribution of the negative samples. It is also interesting to consider that all these negative samples form a virtual graph with the same nodes as before but with negative links between the nodes. When using message-passing framework for the mode learning, there are in fact two messages from each node: one is positive from neighbouring nodes and the other is negative from the negative samples. The positive messages push all the nodes with same semantics to have similar representations, while the negative messages push all nodes with different semantics to have different representations. This strategy is similar to clustering, where samples within same cluster are a small distance apart and large distances between samples indicate the sample is sitting in a different cluster. We believe that these negative messages are precisely what is missing from GCNs and a significant element in their advancement.

The final GCN boosted by diverse negative samples (D2GCN) with L layers, is given in Algorithm 3. Excluding negative sampling, the computational cost is the double that of the original GCN. Note that, although GCNs are used as the base model here, these idea can be easily applied to other GNNs as well.

Experiments

Datasets

The datasets we used are benchmark graph datasets in the literature: Citeseer, Cora and Pubmed (Sen et al. 2008). The datasets include sparse bag-of-words feature vectors for each document as well as a list of document-to-document citation connections. Datasets are downloaded from Pytorch

geometric[2]. To better perform the experiments using DFS, we chose the maximum connected subgraph of each graph data. The datasets were split strictly in accordance with (Kipf and Welling 2017).

Baselines

GCN is the base model. We compare our sampling method with the only three negative sampling methods available to date, then put the selected samples into the convolution operation using Eq. (9). The first method is to select negative samples in a purely random way, named **RGCN** (Kim and Oh 2021). The second one is based on Monte Carlo chains, named **MCGCN** (Yang et al. 2020). The last one is based on personalised PageRank, named **PGCN** (Ying et al. 2018) To ensure the consistency and fairness of the experiments, all the graph convolution models had the same structure and were initialised and trained with the same methods. Note there is no other related negative sampling works in the field.

Setup

The experimental task was standard node classification. We set the length of DFS to 5 and the negative rate as a trainable parameter and trained all models with different numbers of layers in the range $\{2, \cdots, 6\}$ to test behaviour at increasing depths. Each model was trained for 200 epochs on Cora and Citeseer and for 100 epochs with Pubmed, using an Adam optimiser with a learning rate of 0.01. Tests for each model at each depth with each dataset were conducted 10 times. Moreover, all experiments were conducted on an Intel(R) Xeon(R) CPU @ 2.00GHz and NVIDIA Tesla T4 GPU. The code was implemented in PyTorch[3].

Metrics

It was our goal to verify two capabilities of the proposed model: one is ability to improve the prediction performance and the other is to alleviate the over-smoothing problem. Hence, we used the following two metrics:

- **Accuracy** is the cross-entropy loss for the node label prediction on test nodes (the larger is the better);
- **Mean Average Distance (MAD)** (Chen et al. 2020) reflects the smoothness of graph representation (the larger is the better):

$$\text{MAD} = \frac{\sum_i D_i}{\sum_i 1(D_i)}, \quad D_i = \frac{\sum_j D_{ij}}{\sum_j 1(D_{ij})} \tag{10}$$

  where $D_{ij} = 1 - \cos(x_i, x_j)$ is the cosine distance between the nodes i and j.

Results

The results of five models with different numbers of layers on three datasets are shown in Fig. 3. The performances of all five models on Cora and Citeseer were very similar in terms of both Accuracy and MAD at 2 and 3 layers. On Pubmed, our model was marginally better than the others. When the depth increased from 3 to 6, both the Accuracy and MAD of all models decreased to some extent. This is a consistent observation with the literature that increasing the depth of the network leads to a performance drop due to over-smoothing. We also observed that the decreasing trends of RGCN, MCGCN and PGCN on Accuracy were very similar, while PGCN are slightly better than the other two on both Cora and Citeseer datasets, especially at layers 5 and 6. As for MAD, their trends were similar on Pubmed, but on the other two datasets GCN are much worse than the others. Moreover, although the MAD of RGCN, MCGCN and PGCN is close to our D2GCN on Cora and Citeseer, especially at layers 5 and 6 on Citeseer RGCN and PGCN even surpass our method, they are much less accurate than D2GCN. This observation suggests that adding negative samples to the convolution does reduce the over-smoothing to some extent, but choosing the appropriate negative samples is is not trivial. Additional effort needs to be given to the procedure for selecting negative samples.

As shown in the Fig. 3, our proposed model achieved consistently the best performance in terms of accuracy on all datasets. For MAD, it also performs outstandingly in particular on the Cora and Pubmed datasets. At first, we observed that the performance of our model also decreased along with the increasing depth. However, the rate at which performance decreased was much slower than for the other two models – and almost flat on Pubmed. Secondly, in terms of MAD, the performance of our model generally exceeds the others, especially on Pubmed by a large margin. These observations verify that our idea is able to significantly alleviate the over-smoothing problem and in turn improve the prediction accuracy. As a final observation, we found that the variance of our model was smaller than the other four. One possible reason is that the diverse negative samples may quickly change the current node representation rather than keep sticking around the initialisation when only positive samples are used.

The sensitivity of hyper-parameters is analysed and shown in Fig. 4 where we trained 5-layer D2GCN on Citeseer dataset in varying length of DFS and scale of negative rate. We observe that as negative rate or DFS-length goes up, the trends of both Accuracy and MAD are roughly the same, increasing first and then decreasing. For the length of DFS, the outstanding results are obtained when it is equal to 5 or 6. For the negative rate, it achieves the same performance as the trainable parameters, when it is equal to $\{1, 2, 3\}$.

We further show results on three different types of datasets in Tab.**??**: protein graph, gamer graph, paper graph. We compared two SOTA methods and two latest methods for over-smoothing include: GATv2 (Brody, Alon, and Yahav 2021) , GraphSage (Hamilton, Ying, and Leskovec 2017) and MAD (Chen et al. 2020) and DGN (Zhou et al. 2020). Setting is same with the above and all models are with 4 layers. Our method has consistently achieved better performance.

Conclusion and Future Work

In this paper, we identified the importance of negative samples to GCNs and described the criteria for what constitutes

[2]https://pytorch-geometric.readthedocs.io/en/latest/modules/datasets.html

[3]The code is available at https://github.com/Wei9711/D2GCN.

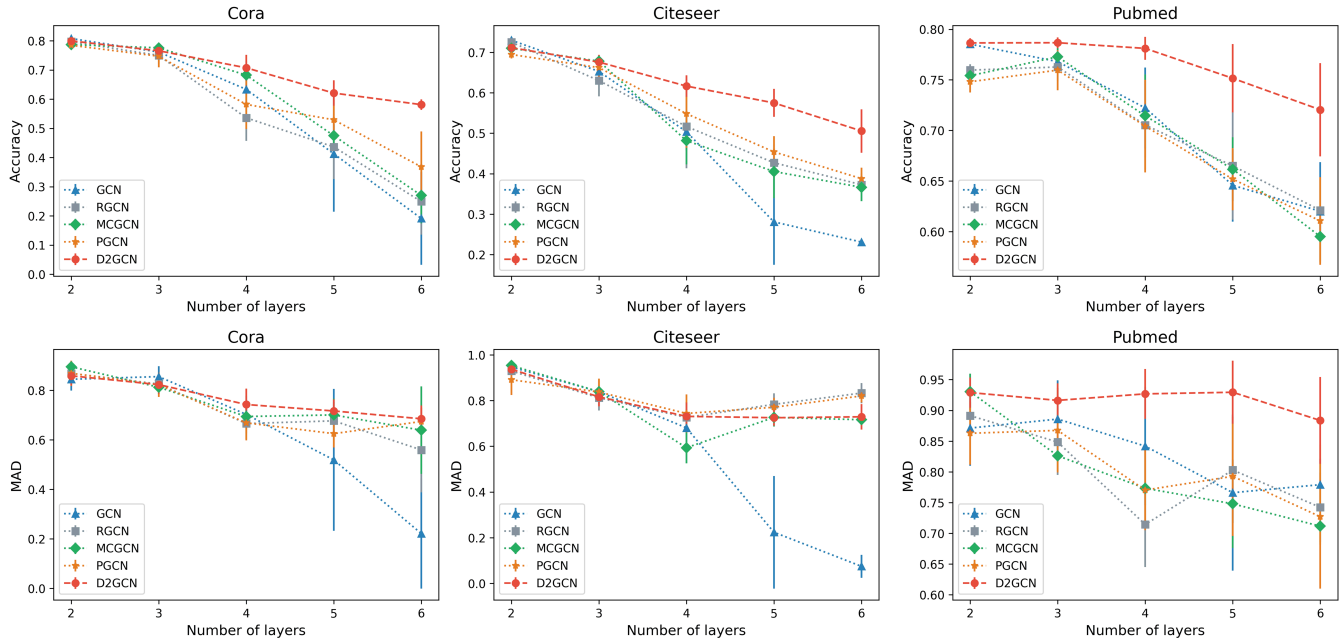

Figure 3: The Accuracy and MAD of five models on three datasets with varying number of layers from 2 to 6. The x-axis denotes the layer number, and the mean and standard deviation of 10 runs are given for each model with each layer number.

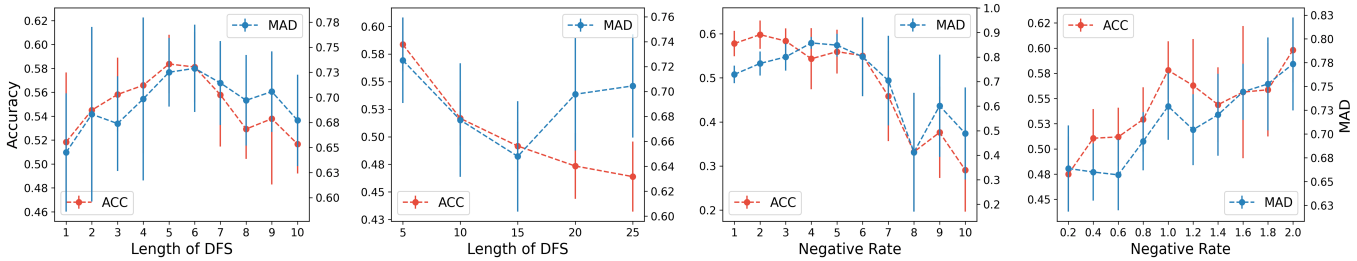

Figure 4: The Accuracy and MAD of D2GCN with 5 layers on Citeseer datasets in varying length of DFS and scale of negative rate. The mean and standard deviation of 10 runs are given for each setting.

Method	Proteins	Twitch(EN)	ogbn-arxiv
D2GCN	**.74 ±.02**	**.58 ±.01**	**.66 ±.01**
GATv2	.66 ±.02	.54 ±.03	.31 ±.06
GraphSAGE	.66 ±.02	.55 ±.01	.58 ±.02
MAD	.66 ±.02	.55 ±.02	*
DGN	.64 ±.01	.52 ±.01	.60 ±.01

* MAD on obgn-arxiv didn't finish one run after more than 12h.

Table 2: Accuracy on other three different types of datasets.

a good negative sample. We introduced DPP to select meaningful negative samples from the dark world of the whole graph. To the best of our knowledge, we are the first to introduce DPP to GCNs for negative sampling and the first to fuse the negative samples into the graph convolution. We further presented a DFS-based heuristic approximation method to greatly reduce the computational cost. The experimental evaluations shows that the proposed D2GCN consistently delivers better performance than alternative methods. In addition to greater predictive accuracy, the method also helps to prevent over-smoothing. With this study, we identify that negative samples are important to graph neural networks and should be considered in future works. Note that the proposed idea can be applied to other graph neural networks apart from GCN.

In future research, we will continue to investigate how to speed-up the DPP sampling process in the algorithm. Another interesting follow-up work would be to investigate more effective aggregation of positive and negative samples as the current solution may lose some information when summing the samples together – especially when the samples are diverse.

Acknowledgements

This work is supported by the Australian Research Council under Australian Laureate Fellowships FL190100149 and Discovery Early Career Researcher Award DE200100245.

References

Affandi, R. H.; Fox, E. B.; Adams, R. P.; and Taskar, B. 2014. Learning the parameters of determinantal point process kernels. In *Proceedings of the 31th International Conference on Machine Learning (ICML), Beijing, China*, 1224–1232.

Affandi, R. H.; Kulesza, A.; and Fox, E. B. 2012. Markov determinantal point processes. In *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence (UAI), Catalina Island, CA, USA*, 26–35.

Brody, S.; Alon, U.; and Yahav, E. 2021. How Attentive are Graph Attention Networks? arXiv:2105.14491.

Bruna, J.; Zaremba, W.; Szlam, A.; and LeCun, Y. 2014. Spectral networks and locally connected networks on graphs. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR), Banff, AB, Canada*.

Chakrabarti, D.; and Faloutsos, C. 2006. Graph mining: laws, generators, and algorithms. *ACM Computing Surveys*, 38(1): 1–69.

Chen, D.; Lin, Y.; Li, W.; Li, P.; Zhou, J.; and Sun, X. 2020. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of The 34th AAAI Conference on Artificial Intelligence (AAAI), New York, NY, USA*, 3438–3445.

Geerts, F.; Mazowiecki, F.; and Pérez, G. A. 2021. Let's agree to degree: comparing graph convolutional networks in the message-passing framework. In *Proceedings of the 38th International Conference on Machine Learning (ICML), Virtual Event*, 3640–3649.

Gillenwater, J.; Kulesza, A.; Fox, E. B.; and Taskar, B. 2014. Expectation-maximization for learning determinantal point processes. In *Proceedings of 27th Annual Conference on Neural Information Processing Systems (NIPS), Montreal, Quebec, Canada*, 3149–3157.

Gillenwater, J.; Kulesza, A.; and Taskar, B. 2012a. Discovering diverse and salient threads in document collections. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), Jeju Island, Korea*, 710–720.

Gillenwater, J.; Kulesza, A.; and Taskar, B. 2012b. Near-optimal MAP inference for determinantal point processes. In *Proceedings of 26th Annual Conference on Neural Information Processing Systems (NIPS), Lake Tahoe, Nevada, United States*, 2744–2752.

Hamilton, W. L.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS), Long Beach, CA, United States*, 1024–1034.

Hough, J. B.; Krishnapur, M.; Peres, Y.; and Virág, B. 2006. Determinantal processes and independence. *Probability Surveys*, 3: 206–229.

Hough, J. B.; Krishnapur, M.; Peres, Y.; and Virág, B. 2009. *Zeros of gaussian analytic functions and determinantal point processes*, volume 51 of *University Lecture Series*. American Mathematical Society.

Kang, B. 2013. Fast determinantal point process sampling with application to clustering. In *Proceedings of 27th Annual Conference on Neural Information Processing Systems (NIPS), Lake Tahoe, Nevada, United States*, 2319–2327.

Kim, D.; and Oh, A. H. 2021. How to find your friendly neighborhood: graph attention design with self-Supervision. In *Proceedings of the 9th International Conference on Learning Representations (ICLR), Virtual Event, Austria*.

Kipf, T. N.; and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations (ICLR), Toulon, France*.

Kulesza, A.; and Taskar, B. 2012. Determinantal point processes for machine learning. *Foundations and Trends in Machine Learning*, 5(2-3): 123–286.

Li, Y.; Tarlow, D.; Brockschmidt, M.; and Zemel, R. S. 2016. Gated graph sequence neural networks. In *Proceedings of the 4th International Conference on Learning Representations (ICLR), San Juan, Puerto Rico*.

Qiao, M.; Bian, W.; Da Xu, R. Y.; and Tao, D. 2015. Diversified hidden models for sequential labeling. *IEEE Transactions on Knowledge and Data Engineering*, 27(11): 2947–2960.

Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Gallagher, B.; and Eliassi-Rad, T. 2008. Collective classification in network data. *AI Magazine*, 29(3): 93–106.

Snoek, J.; Zemel, R. S.; and Adams, R. P. 2013. A determinantal point process latent variable model for inhibition in neural spiking data. In *Proceedings of the 27th Annual Conference on Neural Information Processing Systems (NIPS), Lake Tahoe, Nevada, United States*, 1932–1940.

Watts, D. J.; and Strogatz, S. H. 1998. Collective dynamics of 'small-world'networks. *Nature*, 393(6684): 440–442.

Weisfeiler, B.; and Leman, A. 1968. The reduction of a graph to canonical form and the algebra which appears therein. *NTI, Series*, 2(9): 12–16.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Yu, P. S. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1): 4–24.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How powerful are graph neural networks? In *Proceedings of the 7th International Conference on Learning Representations (ICLR), New Orleans, LA, US*.

Yang, Z.; Ding, M.; Zhou, C.; Yang, H.; Zhou, J.; and Tang, J. 2020. Understanding negative sampling in graph representation learning. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD), Virtual Event, CA, USA*, 1666–1676.

Ying, R.; He, R.; Chen, K.; Eksombatchai, P.; Hamilton, W. L.; and Leskovec, J. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD), London, UK*, 974–983.

Yu, B.; Yin, H.; and Zhu, Z. 2018. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI), Stockholm, Sweden*, 3634–3640.

Zhou, K.; Huang, X.; Li, Y.; Zha, D.; Chen, R.; and Hu, X. 2020. Towards Deeper Graph Neural Networks with Differentiable Group Normalization. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020 (NIPS), virtual*.