UNSUPERVISED PRE-TRAINING OF GRAPH CONVOLUTIONAL NETWORKS

Ziniu Hu, Changjun Fan, Ting Chen, Kai-Wei Chang, Yizhou Sun
University of California, Los Angeles
{bull, cjfan2017, tingchen, kwchang, yzsun}@cs.ucla.edu

ABSTRACT

Recently, graph convolutional networks (GCNs) have been shown to be successful in modeling applications with graph structure. However, training an accurate GCN model often requires a large collection of labeled data and expressive features, which might not be accessible for certain applications. In this paper, we show that a pre-trained GCN model can capture generic structural information of graphs and benefit downstream applications by providing useful features or parameter initiation in a low-resource setting. We further explore three unsupervised tasks: 1) denoising graph reconstruction, 2) centrality score ranking, and 3) cluster detection, for building the pre-trained GCN model without human annotations. Comprehensive experiments demonstrate that our proposed pre-training framework can significantly enhance the performance for both node classification and link classification tasks.

1 INTRODUCTION

Graph convolution networks (GCNs) (Kipf & Welling, 2017) are a powerful tool for modeling graph-structured data. GCN takes a graph with arbitrary features as input and applies convolution filters to generate hierarchical node embeddings layer by layer. The embedding vectors of nodes in the final layer are then passed to a task module to perform the downstream task. The whole GCN framework can be trained in an end-to-end manner and it has been shown to achieve very competitive performance in many graph-based applications, such as semi-supervised node classification (Kipf & Welling, 2017), recommendation systems (Ying et al., 2018) and knowledge graphs (Schlichtkrull et al., 2018). Despite the success, like other deep neural network models, GCNs often have numerous model parameters. Therefore, to train an accurate GCN model, it may require a large collection of labeled data or expressive features. However, annotating and feature engineering for graph data are often expensive and thus limit the applications of GCNs.

Recently, pre-trained language models (e.g., BERT (Devlin et al., 2018)) and image encoders (e.g., VGG Nets (Girshick et al., 2014)) have been proposed to learn generic representations of inputs. These encoders can provide a strong initialization for model parameters and improve the performance of downstream tasks. As these pre-trained encoders can be learned from unannotated corpora or data from related tasks, they reduce the burden of collecting large task-specific annotations and features.

In this paper, we tackle the following research questions: *Can GCNs benefit from unsupervised pre-training? Which unsupervised tasks can be used to pre-train GCNs?* Ideally, the pre-trained graph encoders should capture task-agnostic structural information of graphs. Based on this principle, we study pre-trained graph encoders (GCNs) learned from three unsupervised tasks with different levels of abstractions:

- **Denoising Graph Reconstruction**: We mask out some edges in a graph as noise, and train a model to reconstruct the original graph.
- **Centrality Score Ranking**: We calculate different centrality scores of each node, including PageRank, Betweenness and Closeness, and train GCNs to rank nodes by these scores.
- **Cluster Detection**: We first detect graph clusters using Clauset-Newman-Moore algorithm (Clauset et al., 2004) and then train GCNs to learn node representations that preserve these clusters.

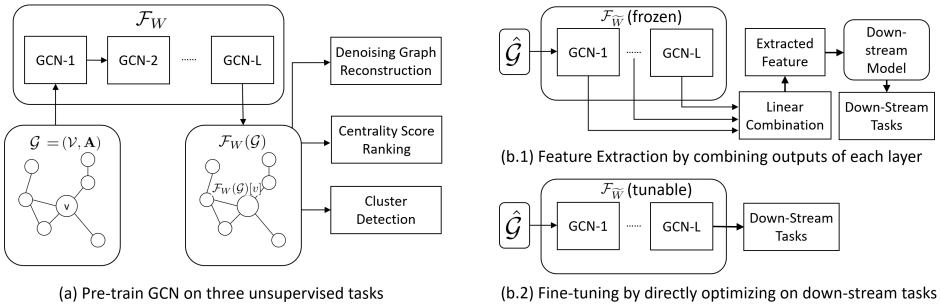

Figure 1: The overall framework for pre-training and adapting GCN.

These three tasks reflect different aspects of graph structural information, from local to global. As a result, GCNs that are pre-trained on a large collection of graphs are able to capture multi-scale structural information for any given graph and thus provide a good initialization for model parameters or regularization for learning loss in various downstream tasks.

We also study two widely used adaptation methods for transferring graph structural information learned from pre-training to downstream tasks: 1) feature extraction and 2) fine-tuning, and compare their performance empirically. Experimental results on two tasks of node classification and link prediction, show that our proposed framework can significantly benefit the GCNs with degree as the input feature.

2 Pre-training Framework

Our pre-training framework is summarized in Figure 1. A GCN model is trained via three unsupervised tasks, and the pre-trained model will then be adapted to down-stream tasks by feature extraction or fine-tuning. The details of pre-training and adaptation are elaborated below.

2.1 Unsupervised Pre-training Tasks

We apply the following encoder-decoder framework to pre-train GCNs. For any given graph $\mathcal{G} = (\mathcal{V}, \mathbf{A})$, where $\mathcal{V}$ denotes the set of all nodes and $\mathbf{A}$ denotes the adjacency matrix, we use GCN, denoted by $\mathcal{F}_W$, as the encoder to get representation vectors $\mathcal{F}_W(\mathcal{G})$ for each node in the graph, which are fed into three decoders to conduct tasks reflecting different levels of structural properties. In the following we use $\mathcal{F}_W(\mathcal{G})[v]$ to denote the representation for node v. The three unsupervised tasks are described as follow:

Task 1: Denoising Graph Reconstruction Vincent et al. (2010) proposed denoising autoencoders, which aim to learn a robust representation of image by reconstructing randomly corrupted input images (i.e., denoising). Motivated by their success, we adopt the auto-encoder framework into the graph setting and refer it as denoising graph reconstruction. Specifically, we add noise to an input graph $\mathcal{G}$ and obtain its noised version $\mathcal{G}^*$ by randomly removing a fraction of existing edges. A GCN model $\mathcal{F}_W$ takes the noised graph $\mathcal{G}^*$ as input and learns to represent the graph. The learned representation is then passed to a neural tensor network Socher et al. (2013) $\mathcal{D}_{denoise}$, which predicts if two nodes are connected or not $\hat{\mathbf{A}}_{u,v} = \mathcal{D}_{denoise}\Big(\mathcal{F}_W(\mathcal{G}^*)[u], \mathcal{F}_W(\mathcal{G}^*)[v]\Big)$. $\mathcal{F}_W$ and $\mathcal{D}_{denoise}$ are optimized jointly with a binary cross-entropy loss:

$$\mathcal{L}_{denoise} = -\sum_{u,v \in \mathcal{V}} \Big(\mathbf{A}_{u,v} \log(\hat{\mathbf{A}}_{u,v}) + (1 - \mathbf{A}_{u,v}) \log(1 - \hat{\mathbf{A}}_{u,v})\Big). \tag{1}$$

In this way, the pre-trained $\mathcal{F}_{\widetilde{W}}$ can learn a robust representation to well preserve local graph structure, which is especially useful for incomplete or noisy graphs.

Task 2: Centrality Score Ranking Node centrality is an important metrics for graphs (Bonacich, 1987). It measures the importance of nodes based on their structural roles in the whole graph. Based

on this, we propose to pre-train GCN $\mathcal{F}_W$ to rank nodes by their centrality scores, so as to enable $\mathcal{F}_W$ to capture structural roles of each node. Specifically, three centrality scores are used:

i *PageRank* (Page et al., 1999) measures nodes' influences based on the idea that high-score nodes contribute more to their neighbors. It describes the 'hub' roles in a whole graph.
ii *Betweenness* (Freeman, 1977) measures the number of times a node lies on the shortest path between other nodes. It describes the 'bridge' roles in a whole graph.
iii *Closeness* (Sabidussi, 1966) measures the total length of the shortest paths between one node and the others. It describes the 'broadcaster' roles in a whole graph.

The above three centrality scores concentrate on the different role of a node in the whole graph. Thus, a GCN that can estimate these scores is able to preserve multi-perspective global information of the graph. Since centrality scores are not comparable among different graphs with different scales, we resort to rank the relative orders between nodes in the same graph regarding the three centrality scores. For a node pair (i, j) and a centrality score s, with relative order as $\mathbf{O}^s_{u,v} = (s_u > s_v)$, a decoder $\mathcal{D}^s_{rank}$ will estimate the rank score by $\hat{S}_v = \mathcal{D}^s_{rank}(\mathcal{F}_W(\mathcal{G})[v])$. Following the pairwise ranking setting defined in (Burges et al., 2005), the probability of estimated rank order is defined by $\hat{\mathbf{O}}^s_{u,v} = \frac{exp(\hat{S}_u - \hat{S}_v)}{1+exp(\hat{S}_u - \hat{S}_v)}$. Thus, we can optimize $\mathcal{F}_W$ and $\mathcal{D}^s_{rank}$ for each centrality score s:

$$\mathcal{L}_{rank} = -\sum_s \sum_{u,v \in \mathcal{V}} \Big(\mathbf{O}^s_{u,v} \log(\hat{\mathbf{O}}^s_{u,v}) + (1 - \mathbf{O}^s_{u,v}) \log(1 - \hat{\mathbf{O}}^s_{u,v}) \Big) \tag{2}$$

In this way, the pre-trained $\mathcal{F}_{\widetilde{W}}$ can learn the global roles of each node in the whole graph.

Task 3: Cluster Detection One important characteristics of real graphs is the cluster (or community) structure (Schaeffer, 2007), which means nodes in a graph can be grouped into different clusters with denser connections within clusters and sparser connections inter clusters. In this paper, we choose Clauset-Newman-Moore algorithm proposed by Clauset et al. (2004) that greedily maximize modularity. Using this algorithm, we can group the nodes in a graph into K different non-overlapping clusters $\mathcal{C} = \{C_i | C_i \subset \mathcal{V}, C_i \cap C_j = \emptyset\}_{i=1}^K$, and a set of indicator functions $\{\mathbf{I}_c(\cdot) \mid C \in \mathcal{C}\}$ to tell whether a given node belongs to cluster C.

We pre-train GCN to learn node representations that these clusters can be easily detected. Firstly we use an attention-based aggregator $\mathcal{A}$ to get a cluster representation by $\mathcal{A}\big(\{\mathcal{F}_W(\mathcal{G})[v] \mid v \in C\}\big)$. Then, a decoder $\mathcal{D}_{cluster}$ estimates the probability that node v belongs to cluster C by: $\hat{\mathbf{I}}_C(v) = \mathcal{D}_{cluster}\Big(\mathcal{F}_W(\mathcal{G})[v],\ \mathcal{A}\big(\{\mathcal{F}_W(\mathcal{G})[v] \mid v \in C\}\big)\Big)$. We then optimize $\mathcal{F}_W$ and $\mathcal{D}_{cluster}$ by:

$$\mathcal{L}_{cluster} = -\sum_{C \in \mathcal{C}} \sum_{v \in \mathcal{V}} \Big(\mathbf{I}_C(v) \log\big(\hat{\mathbf{I}}_C(v)\big) \Big) \tag{3}$$

In this way, the pre-trained $\mathcal{F}_{\widetilde{W}}$ can learn to embed nodes in a graph to a representation space that can preserve the cluster information, which can be easily detected by $\mathcal{D}_{cluster}$.

2.2 Adaptation Procedure

By pre-training $\mathcal{F}_W$ on the above three tasks with $\mathcal{L}_{denoise}$, $\mathcal{L}_{rank}$ and $\mathcal{L}_{cluster}$, we can empower the pre-trained GCN $\mathcal{F}_{\widetilde{W}}$ to capture generic structural information for any given graph. The next step is to adapt $\mathcal{F}_{\widetilde{W}}$ to new tasks. As is shown in igure 1b, there are two main paradigms for adaptation:

i ***F**eature **E**xtraction* (FE) frozes the pre-trained model and extracts the hidden representation as input features for downstream tasks. In this paper, we aggregate node representations from each GCN layer through a weighted combination. Then the extracted features are fed into another task-specific GCN that is trained from scratch.
ii ***F**ine **T**uning* (FT) directly uses the pre-trained model as a backbone model. Here we add an output layer related to the downstream task on top of the pre-trained model, and then fine-tune it in a task-specific way.

After such an adaptation, the framework can benefit the structural information learned from pre-training, and then better deal with the downstream tasks, even with few labels. Details of the architecture implementation, training and adaptation procedure are elaborated more in Appendix 4.

Table 1: Comparison of Micro F1-score (%) results on transductive and inductive node classification and link classification, with training from scratch and different pre-training tasks.

DataSet	Adapt	No Pre-train	Denoise	Centrality	Cluster	All Tasks
Cora	FE	42.5 ± 1.9	55.7 ± 1.1	49.4 ± 1.3	61.2 ± 1.5	$\mathbf{61.9 \pm 1.8}$
	FT	41.6 ± 0.6	48.6 ± 1.7	47.3 ± 0.9	60.8 ± 0.6	$\mathbf{61.5 \pm 0.7}$
Pubmed	FE	44.2 ± 0.9	53.8 ± 0.7	54.6 ± 0.8	$\mathbf{61.8 \pm 0.3}$	61.1 ± 0.7
	FT	43.8 ± 1.0	48.1 ± 1.2	50.2 ± 1.0	$\mathbf{53.3 \pm 1.5}$	53.0 ± 0.9
PPI	FE	42.1 ± 1.3	45.9 ± 1.1	46.5 ± 1.0	$\mathbf{48.4 \pm 1.5}$	47.7 ± 1.1
	FT	40.7 ± 0.7	43.2 ± 0.6	43.1 ± 0.6	46.9 ± 1.0	$\mathbf{47.4 \pm 0.9}$
MovieLens	FE	60.3 ± 1.2	74.3 ± 1.1	68.5 ± 0.9	73.2 ± 1.0	$\mathbf{75.2 \pm 1.4}$
	FT	63.5 ± 0.8	$\mathbf{72.1 \pm 1.1}$	66.7 ± 0.6	70.4 ± 0.8	71.7 ± 0.8

3 EXPERIMENTS

For node classification, we follow the same settings used in Hamilton et al. (2017), including the following benchmark tasks: (1) classify papers in Cora and Pubmed datasets into their subject categories, which is a transductive setting that deals with a single graph. (2) classify protein roles in various protein-protein interaction (PPI) graphs, which is an inductive setting that requires generalizing to unseen graphs. For link classification, we follow the same setting in van den Berg et al. (2017) to classify the ratings that users write to movies on MovieLens 100K dataset.

Since we aim to learn the generic structural feature extractor that can be transferred to different graphs even without expressive features, we choose to use degree of a node as the sole input features. Noted since most existing work (Kipf & Welling, 2017) utilizes extra node features, we do not compare with them directly.

We pre-train the GCNs on synthetic graphs generated with the powerlaw-cluster model by Holme & Kim (2002), and then fine-tune the pre-trained GCNs on downstream tasks via FE (**F**eature **E**xtraction) and FT (**F**ine **T**uning). We comprehensively compare the results: directly train from random initialization (No Pre-train), the one pre-trained by each of the three tasks (Denoise, Rank, and Cluster), and the one pre-trained by all three tasks (All Tasks). We independently run 10 times for each downstream task and report the mean and standard variance results.

The detailed results are shown in Table 1. It is obvious that our proposed pre-training framework can benefit various downstream tasks. Compared with the purely supervised setting without pre-training (No Pre-train + FT), our method (All Tasks + FE) has achieved 48.8%, 39.5%, 17.2% and 18.4% gain on Cora, Pubmed, PPI, and MovieLens dataset. We can also see that different dataset can be benefited by different pre-training tasks. For example, both the transductive and inductive node classification tasks are benefited most by cluster detection task, indicating that cluster is the most useful signal in these tasks. While link classification is benefited most by Denoising graph reconstruction, as they both rely on robust representation of node pairs. Overall, combining all the three tasks together can get the best results. Also, we can see that feature extraction (FE) in most case is a better choice for adaptation, as the datasets we consider have so few data for robust fine-tuning.

4 IMPLEMENTATION DETAILS

4.1 GRAPH CONVOLUTIONAL NETWORKS

In the network based tasks, we are given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathbf{A})$, where $\mathcal{V}$ denotes the set of all nodes and $\mathbf{A}$ denotes the adjacency matrix. Different from the conventional convolutional neural networks (CNN), the convolution operation for graph is defined as the weighted average of neighbors of one particular node. Mathematically, the graph convolution layer is defined as

$$\mathbf{H}^{(l+1)} = \sigma\big(\mathbf{P}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\big) \tag{4}$$

where $\mathbf{P} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{A} + I)\mathbf{D}^{-\frac{1}{2}}$ is the normalized Laplacian matrix of the graph $\mathcal{G}$. $\mathbf{H}^{(l)}$ is the input of the l-th hidden layer of GCN, i.e., the output of the $(l-1)$-th hidden layer. $\mathbf{W}^{(l)}$ is the weight matrix in the l-th hidden layer that would be trained, and $\sigma(\cdot)$ is the activation function.

For any given node, a GCN layer aggregates the previous layer's embedding of its neighbor with $\mathbf{P}$, followed by linear transformation W and nonlinear activation σ, so as to obtain a contextualized node representation. We denote $\mathcal{F}_W(\cdot)$ as L-layer GCNs, paramatrized by $\{W_i\}_{i=1}^{L}$. For each given graph $\mathcal{G} = (\mathcal{V}, \mathbf{A})$ with input features $W^{(1)}$, $F(\cdot)$ can get node representations by:

$$\mathcal{F}_W(\mathcal{G}) = \sigma\Big(\mathbf{P}\big(...\sigma(\mathbf{P}\mathbf{H}^{(1)}\mathbf{W}^{(1)})...\big)\mathbf{W}^{(L)}\Big) \tag{5}$$

In this paper, we use 5-layer GCNs with 128 hidden dimension to accomplish the pre-training and adaptation. The input feature is degree, we encode it with the sinusoid function previously used for positional encoding by Vaswani et al. (2017). Experiments show that such procedure is better than directly using a digit to represent degree. For better training, we also add the layer-wise skip-connection and layer normalization, and they both show to be effective.

4.2 Node Centrality

- PageRank Centrality: the PageRank centrality value $C_{pr}(w)$ of a node w is defined as:

$$C_{pr}(w) = \alpha \sum_{y \to w} \frac{C_{pr}(y)}{d(y)} + \frac{1-\alpha}{N} \tag{6}$$

  where α is a damping factor that controls the probability of whether a surfer follows a link at random or just jump to a random page. It is generally assumed to be set around 0.85. $d(y)$ is the degree of node y. The above formula iteratively defines the PageRank centrality, and we can calculate it repeatedly until convergence. The time complexity of PageRank is $\mathcal{O}(|\mathcal{V}| + ||\mathbf{A}||_1)$.

- Betweenness Centrality: the betweenness centrality value $C_b(w)$ of a node w is defined as:

$$C_b(w) = \frac{1}{|V|(|V|-1)} \sum_{u \neq w \neq v} \frac{\sigma_{uv}(w)}{\sigma_{uv}} \tag{7}$$

  where $|V|$ denotes the number of nodes in the network, σ_{uv} denotes the number of shortest paths from u to v, and $\sigma_{uv}(w)$ denotes the number of shortest paths from u to v that pass through w. The time complexity of Betweenness is $\mathcal{O}(|\mathcal{V}| \cdot ||\mathbf{A}||_1)$.

- Closeness Centrality: the closeness centrality value $C_c(w)$ of a node w is defined as:

$$C_c(w) = \frac{1}{\sum_y d(y, w)} \tag{8}$$

  where $d(y, w)$ is the distance between nodes y and w. The time complexity of Closeness is $\mathcal{O}(|\mathcal{V}| \cdot ||\mathbf{A}||_1)$.

5 Conclusion and Future Work

In this paper, we propose pre-train GCNs to learn graph structural information. To do so, we propose to use three unsupervised tasks, which are denoising graph reconstruction, centrality score ranking, and cluster detection, for building the pre-trained GCN without any labeled data. The learned GCN can be then adapted to downstream tasks using feature extraction or fine-tuning. Experimental results prove the effectiveness of this pre-training framework.

Despite the significant enhancement of our pre-training framework against purely supervised setting, our current results using degree as input features are not comparable to the state-of-the-art supervised results with informative features. We will use our framework to enhance the model that take network embeddings as input (with alignment), and consider combining the learned structural features with informative features to reach state-of-the-art results.

References

Phillip Bonacich. Power and centrality: A family of measures. *American Journal of Sociology*, 92 (5):1170–1182, 1987. ISSN 00029602, 15375390.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *ICML*, 2005.

Aaron Clauset, M. E. J. Newman, , and Cristopher Moore. Finding community structure in very large networks. *Physical Review E*, pp. 1– 6, 2004. doi: 10.1103/PhysRevE.70.066111.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Linton C Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pp. 35–41, 1977.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.

William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 1025–1035, 2017.

Petter Holme and Beom Kim. Growing scale-free networks with tunable clustering. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 65:026107, 03 2002. doi: 10.1103/PhysRevE.65.026107.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations (ICLR-17)*, 2017.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

Gert Sabidussi. The centrality index of a graph. *Psychometrika*, pp. 581603, 1966.

Satu Elisa Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007. doi: 10.1016/j.cosrev.2007.05.001. URL https://doi.org/10.1016/j.cosrev.2007.05.001.

Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*, pp. 593–607, 2018.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pp. 926–934, 2013.

Rianne van den Berg, Thomas N. Kipf, and Max Welling. Graph convolutional matrix completion. *CoRR*, abs/1706.02263, 2017. URL http://arxiv.org/abs/1706.02263.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 6000–6010, 2017. URL http://papers.nips.cc/paper/7181-attention-is-all-you-need.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11:3371–3408, 2010.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pp. 974–983, 2018.