SYNERISE AT KDD CUP 2021: NODE CLASSIFICATION IN MASSIVE HETEROGENEOUS GRAPHS

A PREPRINT

Michał Daniluk
Synerise
Warsaw University of Technology
michal.daniluk@synerise.com

Jacek Dąbrowski
Synerise
jack.dabrowski@synerise.com

Barbara Rychalska
Synerise
Warsaw University of Technology
barbara.rychalska@synerise.com

Konrad Gołuchowski
Synerise
konrad.goluchowski@synerise.com

ABSTRACT

We describe our winning solution to the KDD Cup 2021 Open Benchmark Challenge. We tackle the task of academic paper classification within a heterogeneous graph containing paper, author and institution nodes. We present an efficient model based on our previously introduced algorithms: EMDE and Cleora, on top of a simplistic feed-forward neural network. Our solution can be trained on a single commodity 16 GB GPU, taking around 40 minutes per model. We achieve the 1st place with 0.7454 test accuracy on the initial leaderboard, and 0.7460 test accuracy on the final evaluation set. We release the source code at: https://github.com/Synerise/kdd-cup-2021

Keywords KDD Cup Challenge · node classification · neural networks · network embeddings

1 Introduction

This paper describes one of the winning systems in the KDD Cup 2021 Open Graph Benchmark Challenge, MAG240M-LSC task. The task consists of predicting subject areas of papers in a heterogeneous academic graph. The dataset is extracted from the Microsoft Academic Graph (MAG) and contains 121M academic papers written in English. The paper set is written by 122M author entities, who are affiliated with 26K academic institutions. There are 3 kinds of relations in the graph: **cites** (between 2 paper nodes), **writes** (between author node and paper node), and **affiliated with** (between author node and instutition node). Each paper is represented by a RoBERTa sentence encoder embedding its title and abstract. Out of 121M paper nodes, only approximately 1.4M nodes are arXiv papers annotated with their subject areas, e.g., cs.LG (Machine Learning).

The task is to predict the primary subject areas of the given arXiv papers, which is cast as an ordinary multi-class classification problem among 153 classes. The evaluation metric is the classification accuracy.

Data is split by time, with the trainset composed of arXiv papers published until 2018, validation set - of the 2019 papers, and testset - of the 2020 papers. The split reflects the practical scenario of helping the authors and moderators annotate the subject areas of the newly-published arXiv papers.

We tackle the task with an efficient model based on our previously introduced algorithms: EMDE and Cleora, on top of a simplistic feed-forward neural network. We use EMDE to represent nodes in the form of *sketches* - structures representing local similarity, which additionally allow for easy accumulation of multiple object values (e.g. for a joint representation of the contents of multiple cited papers). We use Cleora for label propagation, i.e. representing nodes with sets of labels observed in the training data. We achieve the 1st place with 0.7454 test accuracy on the initial leaderboard, and 0.7460 test accuracy on the final evaluation set.

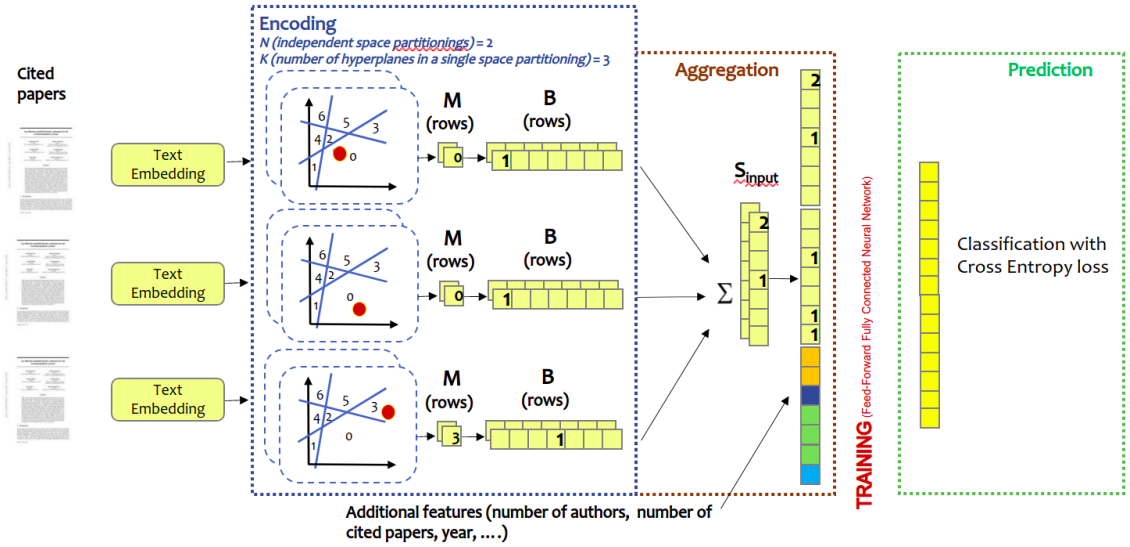

Figure 1: EMDE architecture overview. Example of creating an aggregated sketch of cited papers as input to a feed forward neural network.

2 Preliminaries

Our solution is composed of multiple features computed with our previously introduced algorithms: the Efficient Manifold Density Estimator (EMDE) [Dąbrowski et al., 2021], Cleora [Rychalska et al., 2021], and Fourier Feature Encoding. We briefly introduce each of them in this section and then proceed to describe the full model in Section 3.

2.1 EMDE

Efficient Manifold Density Estimator (EMDE) introduced in [Dąbrowski et al., 2021] is a probability density estimator inspired by Count-Min Sketch algorithm (CMS) and local sensitive hashing (LSH). The overview of the algorithm is shown in Figure 1. EMDE ingests input data represented by vectors embedded on manifolds spanned by various upstream representation learning methods. The manifolds are then partitioned with the data-dependent LSH method (DLSH). The partitioning method divides the manifold into regions, analogous to CMS *buckets*. The purpose of manifold partitioning follows the logic of LSH: similar data points should be mapped to the same region. While a single region is large (typically 64-256 regions form a single partitioning covering the whole manifold, as described in [Dąbrowski et al., 2021]), multiple independent partitionings allow to obtain a high resolution map of the manifold via intersection or ensembling. The number of regions on a single manifold is further denoted as $sketch\_dim$ (2^K in Figure 1), and the number of independent divisions - as $sketch\_depth$ (N in Figure 1).

The resulting region assignment vectors (*sketches*) can be thought of as a form of a histogram. Each position within a sketch corresponds to a region, and each value represents the number of data points in the given region. For example, a sketch of an item (e.g. of a single academic paper) can take the form of $I = [0, 0, 1, 0, 0]$, which means that the data point in question is located in region number 2 out of 5 total regions. All items are represented with sketches of the same dimensionality, and representations of item sets are computed by simple summation of individual item sketches. For example, a unordered collection of 4 academic papers can be represented as sketch $S = [1, 0, 2, 0, 1]$ containing 4 papers in total, one located in region number 0, two in region number 2, and one in region number 4. EMDE precomputes sketch representations of all items within the inventory, and computes aggregate representations of desired item sets.

2.2 Cleora

Cleora [Rychalska et al., 2021] is a fast and efficient node embedding technique. Cleora combines intuitions from both random walk-based methods as well as graph convolutional networks (GCNs) into a simple solution. Instead of sampling individual random walks, we consider all possible random-walk transitions in a Markov transition matrix. Multiplication of vertex embeddings by this matrix allows to perform all possible random walks in parallel, with one large step. Cleora is further motivated by the surprising observation that in GCNs, the effect imposed by non-linearities and dense layers

in between convolutions does not increase the expressive power, but rather normalizes and stabilizes the embeddings. Graph convolutions perform a neighborhood smoothing operation and for many purposes all other GCN operations are redundant, as shown in [Wu et al., 2019, Oono and Suzuki, 2020]. This operation is emulated within Cleora with the iterative matrix multiplications. Cleora marks significant gains in speed due to its simple algorithm and efficient implementation in Rust. At the same time, it is shown to achieve competitve results compared to other recent methods on tasks such as link prediction and node classification. Cleora is publicly available at https://github.com/Synerise/cleora.

2.3 Fourier Feature Encoding

Fourier Feature Encoding is a way to sidestep the necessity of explicit normalization of model inputs. Instead of feeding a single input feature, we transform it into a 16-dimensional vector. First, an input numeric value is divided by a few numbers representing increasing scale levels (in our case, 8 scale levels represented by powers of 2). Then, each of the 8 results is fed to sin and cos functions. The numeric significance of the feature is thus represented on multiple levels, and in a numerically stable way as any number is brought to a small numeric interval defined by trigonometric functions. Below we present a code snippet of the Fourier Feature Encoding.

```
import numpy as np

def multiscale(x, scales):
    return np.hstack([x.reshape(-1,1)/pow(2., i) for i in scales])

def encode_scalar_column(x, scales=[-1, 0, 1, 2, 3, 4, 5, 6]):
    return np.hstack([np.sin(multiscale(x, scales)), np.cos(multiscale(x, scales))
    ])
```

3 The Full Challenge Solution

The model architecture itself is very simple - it is a 4-layer residual feed forward neural network with 3500 neurons in each hidden layer, with leaky ReLU activations and batch normalization, and with 153 outputs representing the possible arXiv classes. The most elaborate part of our solution are the input features, which are precomputed and fed as model input in the form of a large concatenated vector.

Feature computation starts with precomputing some models which will be needed later:

1. **Fitting EMDE on RoBERTa embeddings.** We run EMDE on the RoBERTa embeddings of all papers, using $sketch\_dim = 256$ and $sketch\_depth = 40$. This way we partition the embedding space into 256 regions, repeating the procedure 40 times independently. Papers similar in terms of RoBERTa embeddings are expected to fall into the same regions often, as per the Locality-Sensitive Hashing paradigm. The resulting per-paper sketches have dimensionality $40 \times 256 = 10240$.
2. **Precomputing Cleora embeddings for papers and authors.** We form a single large adjacency matrix, treating author and paper nodes in exactly the same way. A value of 1 is inserted for pairs of nodes citing each other and for author-paper pairs.
   (a) **Label propagation in Cleora inputs.** Embedding computation proceeds exactly as in Cleora, with the exception of node embedding initialization. Concretely, the papers which do have assigned labels are initialized with a one-hot-encoded vector expressing their labels (a value of 1 at the index denoting specific label assignment). Similarly, author node embeddings, when the author has some papers with labels, are initialized with a summed and normalized one-hot encoded vector of paper labels. Nonlabeled nodes have the embedding initialized to all-zero vectors. Thus, the embedding dimensionality is 153 for all entities, with most of them having been initialized randomly, and some based on their label assignment.
   (b) **Cleora training.** Cleora training is done on a half of all labelled nodes in the trainset, the other half is used for full model training, with its labels used as output. This is to prevent information leaks between Cleora embeddings and model outputs. Cleora is trained with 2 iterations.
3. **Precomputing Cleora embeddings for institutions.** Institutions are embedded with Cleora using the clique expansion scheme [Rychalska et al., 2021]. A single clique is formed per author, so whenever an author belongs to multiple institutions, all these institution nodes are connected with each other. The graph consists of only one node type: the instutution node and embeddings are computed with 3 Cleora iterations.
4. **Fitting EMDE on institution embeddings from Cleora.** Similarly as with RoBERTa, a new instance of EMDE is fitted on Cleora institution embeddings. This time the sketch dimensions can be smaller, with the

embedding space divided into 128 regions and the procedure repeated 40 times independently. Resulting sketches have dimensionality $40 \times 128 = 5120$.

Having completed the aforementioned steps, we have Cleora embeddings for two types of nodes: papers and authors. We also have two fitted EMDE models for computing sketches of papers and institutions. We then proceed to compute input features for the neural network:

1. **Paper vector representation.** RoBERTa paper embedding.
2. Three width-wise L2-normalized, concatenated papers sketches: **cited papers sketch** that represents a summary content value of the cited papers, **citing papers sketch** represents papers which cited the given paper, and **authors papers sketch** that is an aggregation of all other papers of given paper authors. We aggregate each sketches with a simple element-wise summation followed by L2 normalization across $sketch\_dim$. This can be done as EMDE sketches are piecewise-additive and the resulting structure retains sketch properties and shape. This aggregate sketch forms a $3 * 40 * 256 = 30720$-element component of the input vector to the neural network. We take care to always include papers from the same year or older that the current paper in order not to break the temporal logic
3. **Institutions sketch.** An aggregate L2-normalized sketch of institutions to which the paper authors belong.
4. **Label representation of neighbors.** We treat the following 3 types of nodes as neighbors: cited papers, citing papers, another papers of the same authors. For each neighbor node with labels, a one-hot-encoded label vector is created, and all such vectors of all adjacent papers are summed and L2-normalized. This results in a 153-item vector for each of the 3 neighbor types. Furthermore, we calculated such vectors for papers which are 2 hops away in the graph from the current paper - i.e. "papers cited by papers cited by the current paper". We only include papers older than the current paper. Additionally, these vectors are precomputed separately for different timespans in 3 versions: 1) including older examples, 2) including examples from 1 year ago, 3) including examples from 2 years ago. This serves to increase model robustness against the variability of labels over time.
5. **Cleora-propagated label representation of neighbors.** Similar to point 4), but instead of true labels, utilizing labels propagated by Cleora from the unmasked to the masked part of the dataset. We computed these vectors separately for papers from the same year as current paper and for all not newer publications.
6. **Cleora representation of paper.** Cleora embedding representation of given paper.
7. **Cleora representation of paper authors.** Summation of L2-normalized Cleora vectors of all authors of the paper.
8. **Numerical features.** We also include simple features expressed with a single number, which are: the number of cited papers, the number of papers which cite the given paper, the number of another papers of the authors, the number of authors, publication year. Additionally, we include all L2 norms of all label representation vectors before normalization computed in points 4-6. All of these are encoded with our Fourier Feature Encoding.

In summary, we obtain a concatenated input vector of length 41381 per example.

3.1 Model Training

The model is written in PyTorch. We train on a single NC24s v3 Azure instance with Intel Xeon CPU ES-2690 v4 @ 2.60 GHz (441 GB memory), using 1 Tesla V100 16 GB GPU. Training a single model takes around 42 minutes, and inference takes around 7 minutes. We use AdamW optimizer [Loshchilov and Hutter, 2017] with first momentum coefficient of 0.9 and second momentum coefficient of 0.999 (Standard configuration recommended by [Kingma and Ba, 2014]) with weight decay of 0.01 and a mini-batch size of 512 for optimization. The model was trained for 2 epochs, the first with a greater learning rate of 0.0001 and the second with a smaller learning rate of 0.00005. We observe performance gains after sorting the dataset according to date, so we apply this in the final submission.

To achieve maximal performance, we train 60 independent ensemble models. Training of a single ensemble requires independent sampling of the 1/2 labeled nodes for Cleora computation, while the rest is used for training. For final submission, we train on train set combined with the validation set.

3.2 Results

The metric used in the challenge is the classification accuracy. Our score on hidden test set on initial leaderbord is 0.7454 compared to the best baseline model R-GAT [Veličković et al., 2017] which achieved 0.6949 accuracy score.

4 Conclusions

In this report we have described our solution to the KDD Cup 2021 Open Graph Benchmark Challenge, MAG240M-LSC task. Our system achieves 0.7454 accuracy on the initial leaderboard (1st place) and 0.7460 on the final testset. We have shown that a combination of EMDE and Cleora algorithms builds a very successful architecture for node classification in massive heterogeneous graphs.

References

Jacek Dąbrowski, Barbara Rychalska, Michał Daniluk, Dominika Basaj, Konrad Gołuchowski, Piotr Babel, Andrzej Michałowski, and Adam Jakubowski. An efficient manifold density estimator for all recommendation systems. 2021. URL `https://arxiv.org/abs/2006.01894`.

Barbara Rychalska, Piotr Bąbel, Konrad Gołuchowski, Andrzej Michałowski, and Jacek Dąbrowski. Cleora: A simple, strong and scalable graph embedding scheme, 2021.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6861–6871. PMLR, 09–15 Jun 2019. URL `http://proceedings.mlr.press/v97/wu19e.html`.

Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=S1ldO2EFPr`.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.