Tsinghua University

media and network lab

AutoGraph Challenge Solution

-- KDD Cup 2020 AutoML Track

SmartMN-THU

Ke Tu, Ziwei Zhang, Dingyuan Zhu, Zeyang Zhang, Xin Wang

Tsinghua University

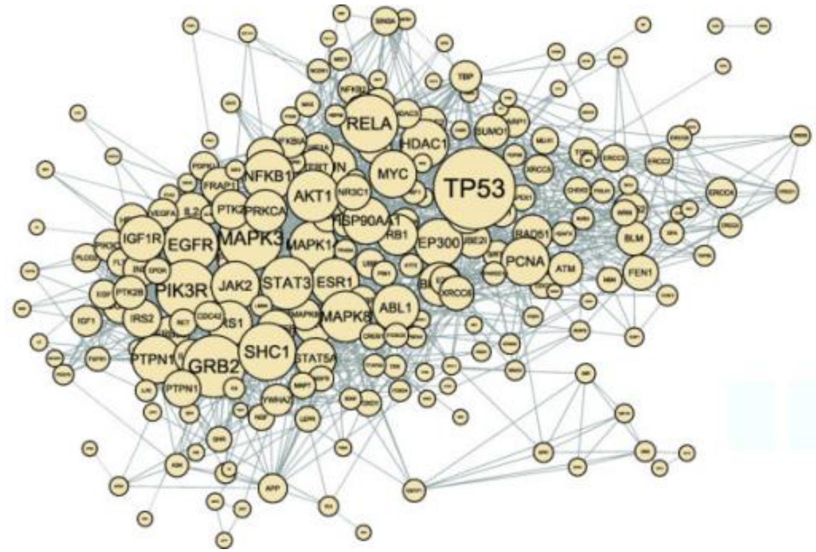
Advisor: Prof. Wenwu Zhu

Tsinghua University

Code: https://github.com/AutoGraphMaNlab/AutoGraph

Graphs are Ubiquitous

Social Network

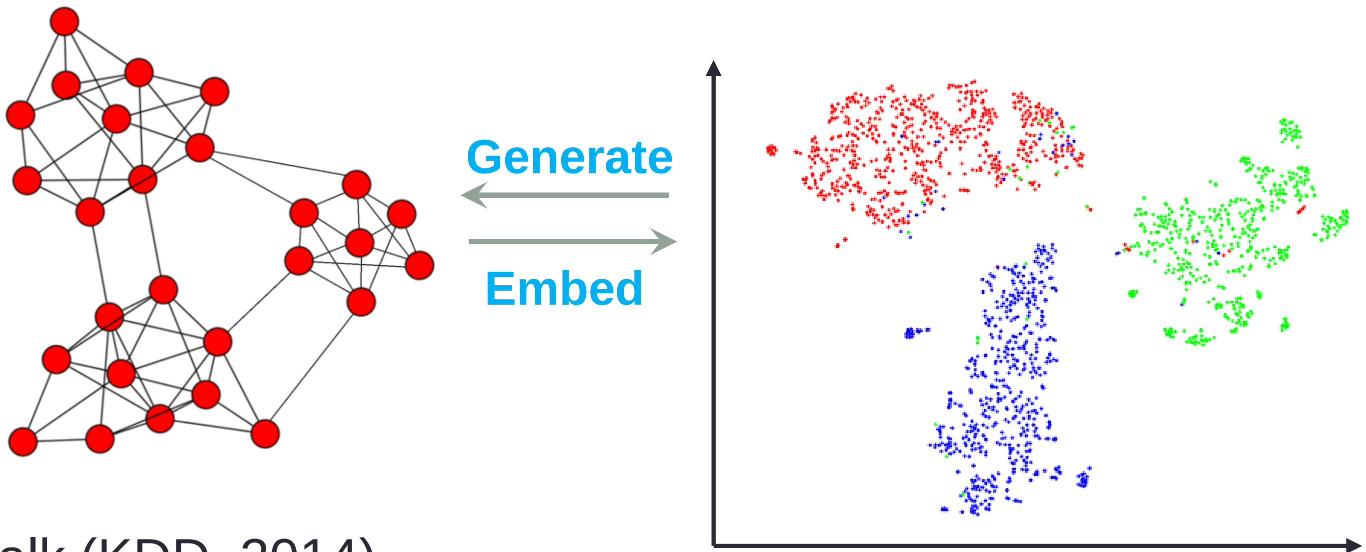


Biology Network

Traffic Network

You can skip to page 8 if you are familiar with the competition!

Network Embedding: Vector Representation of Nodes



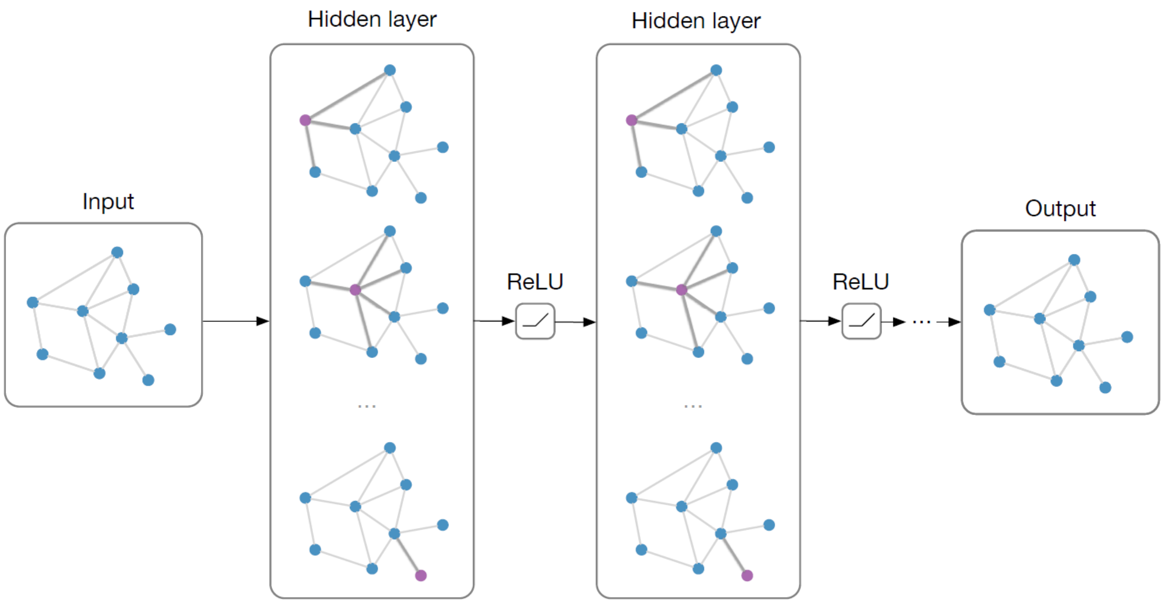
- DeepWalk (KDD, 2014)
- LINE (WWW, 2015)
- Node2vec (KDD, 2016)
- SDNE (KDD, 2016)
- HOPE (KDD, 2016)

…

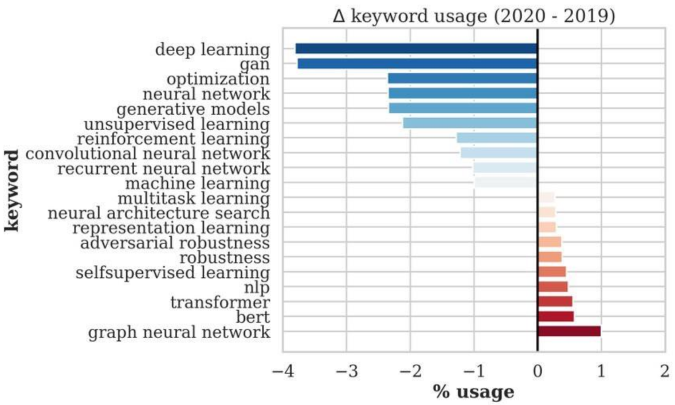
A Survey on Network Embedding. *IEEE TKDE*, 2018.

Title / Author	Cited by	Year
XGBoost: A Scalable Tree Boosting System T Chen, C Guestrin Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge ...	2832	2016
DeepWalk: online learning of social representations B Perozzi, R Al-Rfou, S Skiena Proceedings of the 20th ACM SIGKDD international conference on Knowledge ...	1818	2014
node2vec: Scalable Feature Learning for Networks A Grover, J Leskovec Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge ...	1622	2016
Why Should I Trust You?: Explaining the Predictions of Any Classifier MT Ribeiro, S Singh, C Guestrin Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge ...	1528	2016
Knowledge vault: a web-scale approach to probabilistic knowledge fusion X Dong, E Gabrilovich, G Heitz, W Horn, N Lao, K Murphy, T Strohmann, ... Proceedings of the 20th ACM SIGKDD international conference on Knowledge ...	904	2014
Collaborative Deep Learning for Recommender Systems H Wang, N Wang, DY Yeung Proceedings of the 21th ACM SIGKDD International Conference on Knowledge ...	626	2015
Structural Deep Network Embedding D Wang, P Cui, W Zhu Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge ...	563	2016

Graph Neural Networks



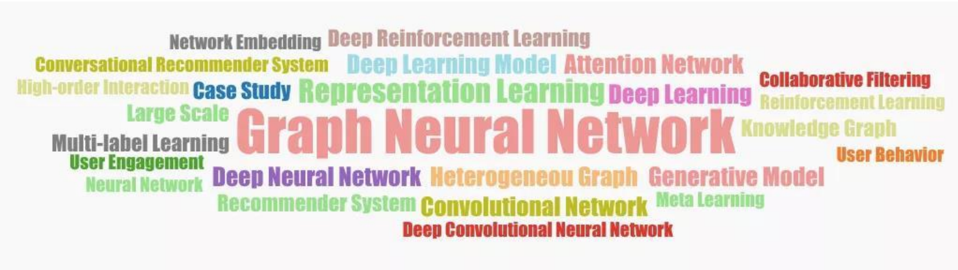
Picture credit to Thomas Kipf.



ICLR20 Keyword Change

- Spectral GCN (ICLR, 2014)
- ChebNet (NeurIPS 2016)
- GCN (ICLR, 2017)
- GraphSAGE (NeurIPS, 2017)
- GAT (ICLR, 2018)

…

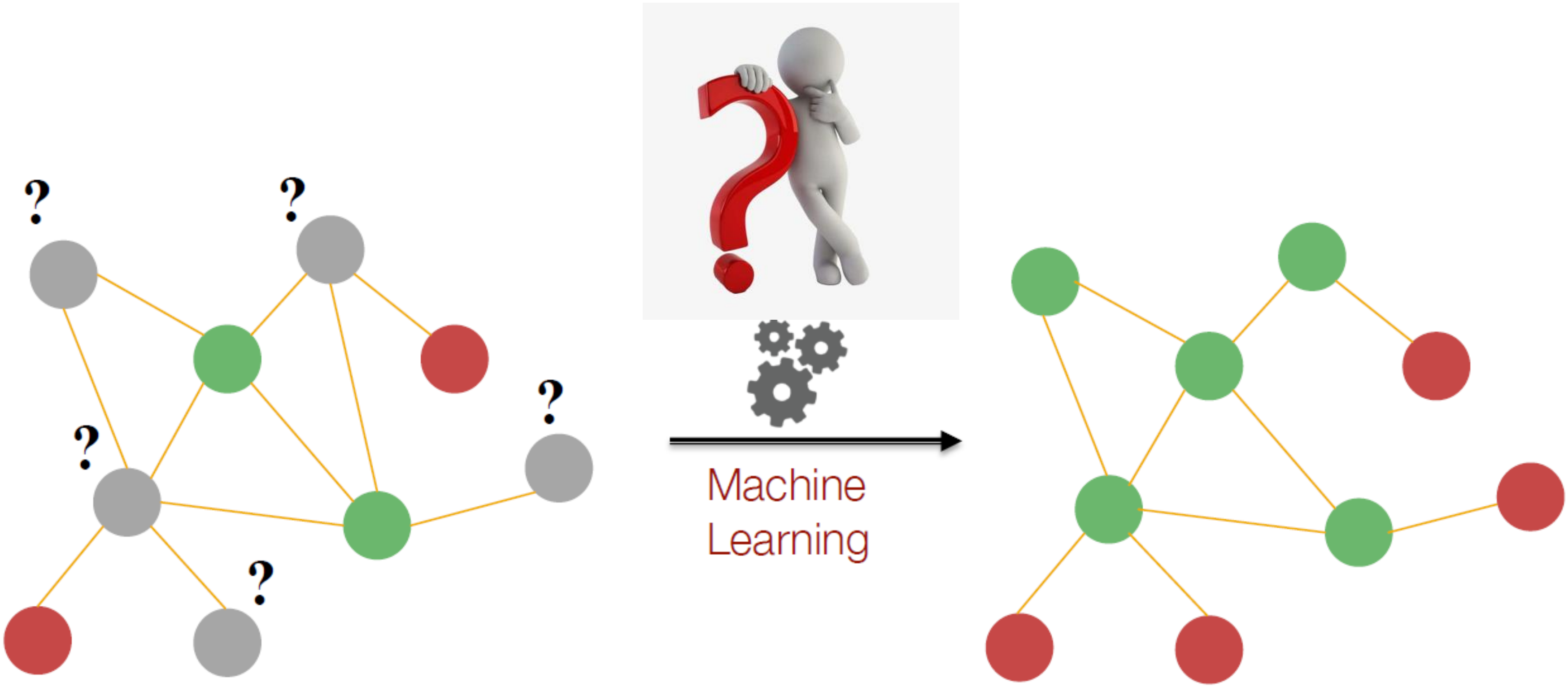


KDD 20 Keywords

Deep Learning on Graphs, A Survey.

IEEE TKDE, 2020.

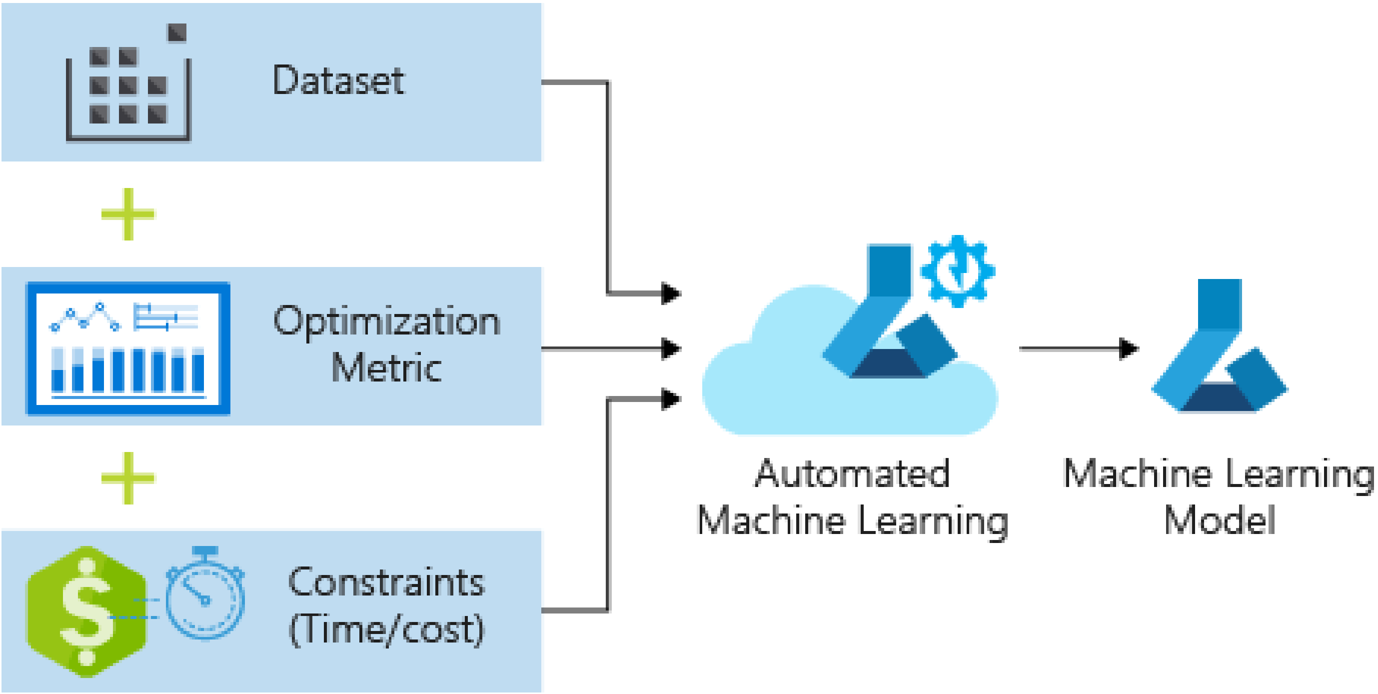
However, human efforts are still needed…



- Different tasks and different datasets may be completed different!

Picture credit to CS224W, Stanford University, by Jure Leskovec.

AutoML



Design ML methods → Design AutoML methods

Picture credit to Microsoft Azure Machine Learning AutoML

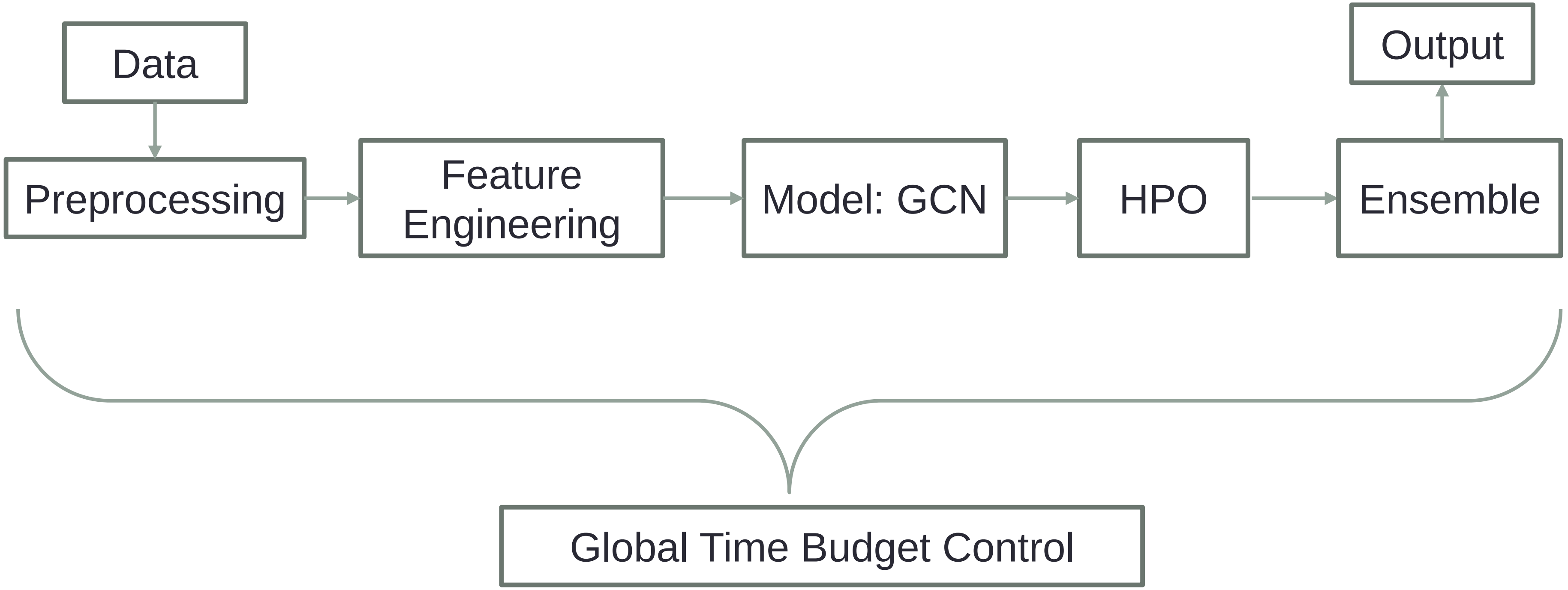
AutoGraph: KDD Cup 2020 AutoML Track

- Task: node classification
- 15 datasets: 5 for training, 5 for validation, and 5 for final testing
  - Validation/testing datasets are not accessible to participants
  - The AutoML algorithm should automatically handle them!
  - Leaderboard on validation and no any feedback on final testing
- Different graph types: directed/undirected, weighted/unweighted, with/without node features…
- Time budget

https://www.4paradigm.com/competition/kddcup2020

https://www.automl.ai/competitions/3

Framework


Data
Output
Preprocessing
Feature Engineering
Model: GCN
HPO
Ensemble
Global Time Budget Control

Preprocessing

- Graph structure
  - Whether graph is directed/undirected, weighted/unweighted, signed/unsigned
- Node features
  - Drop constant features
  - Drop one-hot encoding of node-IDs
- Node labels
  - Whether the training labels are balanced

Feature Engineering (1)

- Selection of original features
  - Select up to top 2,000 important features by LightGBM [1]
- Feature Generation
  - Eigen-GNN [2]
  - In/out degree (one-hot encodings)
  - Other methods tried
    - PageRank
    - Graphlets [3]
    - GDC [4]
    - DeepFM [5]
    - DeepWalk

[1] https://lightgbm.readthedocs.io/en/latest/
[2] Ziwei Zhang, et al., Eigen-GNN: A Graph Structure Preserving Plug-in for GNNs. *arXiv, 2006.04330.*
[3] Network motifs: simple building blocks of complex networks, *Science 2002.*
[4] Diffusion Improves Graph Learning, *NeurIPS 2019.*
[5] DeepFM: A Factorization-Machine based Neural Network for CTR Prediction, *IJCAI 2017.*

Feature Engineering (2)

- Modeling the interactions among node features
- DeepGL [1]:
  - Feature Generation Aggregators: sum, mean, max, min
  - Feature Selection
    - Lightgbm
    - Other methods tried: connected components
  - Procedure
    - Generate features per aggregator
    - Select top K=200 important features for concatenation and future generation
    - Repeat N=5 times

Feature engineering is more important if no node feature is available

[1] Deep Inductive Graph Representation Learning, *TKDE 2018.*

Model

- GCN: incorporate graph topology and node attributes, trained end-to-end
- Additional “tricks”:
  - Jumping connections [1]: aggregate information from different layers
  - Batch normalization: shown empirically to be important [2]

- Have tried other methods:
  - GAT: slightly better results, but much more time consuming, e.g. 5x time
    - Better results when allocating these time budgets to HPO + ensemble
  - GraphSAGE: similar results with GCN
  - GIN: similar results with GCN
  - Tried other tricks such as residual connections [3] or removing non-linearity [4], but not observing consistent improvements

[1] Representation Learning on Graphs with Jumping Knowledge Networks, *ICML 2018.*
[2] Benchmarking Graph Neural Networks, *arXiv 2003.00982.*
[3] Predict then Propagate: Graph Neural Networks meet Personalized PageRank, *ICLR 2019.*
[4] Simplifying Graph Convolutional Networks, *ICML 2019.*

Hyper-Parameter Optimization (HPO)

- Search hyper-parameters for 20 times (if there is enough time)
  - Each time we re-split the datasets into training and validation
  - Each time we search 5 group of hyper-parameters and choose the best one
- Consider some model choices as hyper-parameters
  - E.g. the jumping connection function
- HPO method: Tree Parzen Estimation [1]
  - Better compatibility with discrete hyper-parameters compared to Gaussian Process Regression used in [2]

[1] Algorithms for hyper-parameter optimization. *NIPS 2011*.
[2] Ke Tu, et al. AutoNE: Hyperparameter Optimization for Massive Network Representation Learning*. KDD, 2019.*

Hyper-parameters and Their Ranges

- We manually set the default hyper-parameters and their search ranges
  - Number of GCN layers: {1,2}, default 2
  - Hidden size in the first layer: [8, 128], default 64
  - Hidden size in the second layer: [8, 64], default 32
  - Dropout rate: [0.1, 0.9], default 0.5
  - Learning rate: [1e-4,1e0], default 0.005
  - Number of epochs: [100, 300], default 300
  - Weight decay: [1e-4, 1e-1], default 5e-4
  - Jumping connection function: {sum, concat, none}, default concat
- Empirically, we find that the most important hyper-parameters are number of layers, learning rate, and jumping connection function

Ensemble

- A simple voting-like method:
  - Sort the models by their accuracies (on the validation set)
  - Filter models showing significantly poor results
    - Make sure that the variance of top models are smaller than a threshold
  - Compute the weighted sum of the predicted probabilities as final results

- Important in getting stable predictions

Global Time Budget Control

- Goal of the global timer: ensure valid results and estimate running time

- Stop and return results whenever the remaining time < 5 seconds
- Control the time used in feature engineer
  - Stop generating features when it had cost 1/3 budget to save time for models
  - If the number of edges is too large, does not run EigenGNN
- Estimate the time of training one model to better allocate resources

Further Discussions

- AutoML algorithms may also suffer from over-fitting
  - E.g., only focusing on getting better results on the validation set
  - We prepare a dozen off-line datasets for additional validation (all publicly datasets from PyTorch Geometric)

- Why not doing Neural Architecture Search (NAS):
  - Time budget
  - Most existing GCN architectures are relatively simple
    - E.g., the number of layers is usually no more than 2 due to over-smoothing

- Another strategy we tried but not worked: use network embedding methods to extract topology features, concatenate with node features, and adopt a non-linear classifier (e.g., LightGBM)

Results

Validation

1	supergx	3.2
2	daydayup	4.8
3	common	5.0
4	qqerret	5.8
5	SmartMN-THU	7.2
6	shiqitao	7.4
7	JunweiSun	7.8
8	aister	9.0
9	Qitian	10.2
10	PostDawn	10.8
11	Alpha	13.4
12	PASA_NJU	14.4

Test

1	aister	4.8
2	PASA_NJU	5.2
3	qqerret	5.4
4	common	6.6
5	PostDawn	7.4
6	SmartMN-THU	7.8
7	JunweiSun	7.8
8	u1234x1234	9.2
9	shiqitao	9.6
10	supergx	11.8

Thanks!

Ziwei Zhang, Tsinghua University

zw-zhang16@mails.tsinghua.edu.cn

https://zw-zhang.github.io/

https://github.com/AutoGraphMaNlab/AutoGraph