Graph Mining on Static, Multiplex and Attributed Networks

Benedek Rozemberczki

The University of Edinburgh

June 3, 2021

Introduction

What is graph mining?

- What is a graph?
- Why do we care about graphs?
- What is graph mining?
- How does it relate to network science?
- Why do we care about graph mining?

Challenges for graph mining

- Web-scale data
- Extreme sparsity
- Autocorrelation
- Multimodal nature
- Information attribution
- Induction and transduction
- Deviations from the homogeneous - static graph model

Graph mining and this thesis

- Node embedding
- Graph fingerprinting techniques
- Spatial parametric machine learning
- Graph neural networks (message passing)
- Explainable embedding ensembles
- Tools to foster graph mining research

The philosophy of graph mining techniques in this thesis

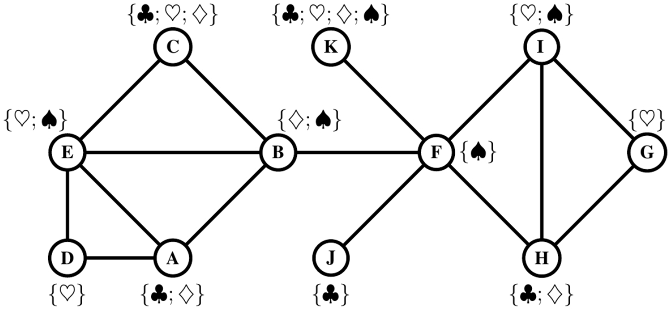


Figure 1: (a) Macro and micro hierarchy. (b) Multi-resolution view. (c) Feature distributions in neighbourhoods. (d) Multi-modality. (e) Attribution of gains by various graph mining algorithms.

Modern Graph Mining Techniques

GEMSEC: Graph Embedding with Self Clustering. Benedek Rozemberczki, Ryan Davies, Rik Sarkar, Charles Sutton. Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining.

Introduction

- The goal of our work is to design a community aware node embedding procedure.
- We want to extend the most general node embedding techniques such as:
  - **DeepWalk**
  - **Node2Vec**
  - **Walklets**
- Implicit factorization was never used so far for community aware node embeddings.
- Creating a clustered node embedding space might help in downstream tasks.

GEMSEC explained in a single image

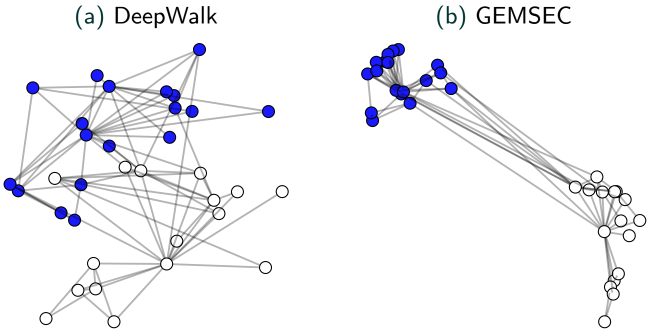


Figure 2: Zachary's Karate club graph. White nodes: instructor's group; blue nodes: president's group. GEMSEC produces embedding with more tightly clustered communities.

Community aware embeddings

- **M-NMF:** Factorizes a matrix that is a sum of the adjacency matrix and the neighbourhood overlap matrix.
- **ComE:** Models the random walk generated pointwise mutual information matrix with a Gaussian mixture model.
- **DANMF:** Creates a deep factorization of the adjacency matrix.

We want an algorithm which has the following characteristics:

1. Representation dimensions are disentangled from the community number.
2. It has a linear runtime.
3. Sequence sampling agnostic.

Model formulation

We assume that a sampling strategy S generates neighbourhoods $N_s(v)$ of nodes $v \in V$.

$$\min_f \sum_{v \in V} -\log P(N_S(v)|f(v)) \tag{1}$$

We assume conditional independence:

$$P(N_S(v)|f(v)) = \prod_{n_i \in N_S(v)} P(n_i \in N_S(v) \mid f(v), f(n_i)). \tag{2}$$

In addition, we assume a softmax and inner product parametrization:

$$P(n_i \in N_S(v) \mid f(v), f(n_i)) = \frac{\exp(f(n_i) \cdot f(v))}{\sum_{u \in V} \exp(f(u) \cdot (f(v))}. \tag{3}$$

A DeepWalk or Node2Vec type model has an objective function can be described by:

$$\min_{f} \sum_{v \in V} \left[\ln \left(\sum_{u \in V} \exp(f(v) \cdot f(u)) \right) - \sum_{n_i \in N_S(v)} f(n_i) \cdot f(v) \right] . \tag{4}$$

We augment Equation (4) with a clustering cost:

$$\mathcal{L} = \underbrace{\sum_{v \in V} \left[\ln \left(\sum_{u \in V} \exp(f(v) \cdot f(u)) \right) - \sum_{n_i \in N_S(v)} f(n_i) \cdot f(v) \right]}_{\text{Embedding cost}} \\ \underbrace{+ \gamma \cdot \sum_{v \in V} \min_{c \in C} \| f(v) - \mu_c \|_2}_{\text{Clustering cost}} . \tag{5}$$

A note on initializations

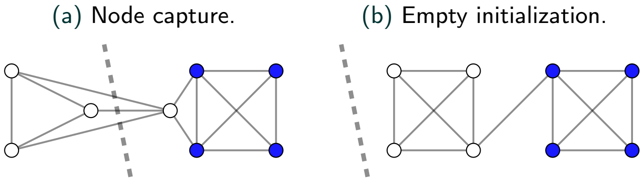


Figure 3: Potential issues with cluster cost weighting and cluster initialization. Colors denote the ground truth community memberships and the computed cluster boundary is denoted by the dashed line.

We set the clustering cost coefficient and learning rate dynamically:

$$\gamma = \gamma_0 \cdot \left(10^{\frac{-t \cdot \log_{10} \gamma_0}{w \cdot l \cdot |V| \cdot N}}\right) \tag{6}$$

$$\alpha = \alpha_0 - (\alpha_0 - \alpha_F) \cdot \frac{t}{w \cdot l \cdot |V| \cdot N} \tag{7}$$

A note on numerical stability

The gradient of a representation is given by:

$$\frac{\partial \mathcal{L}}{\partial f(v^*)} = \underbrace{\frac{\sum\limits_{u \in V} \exp(f(v^*) \cdot f(u)) \cdot f(u)}{\sum\limits_{u \in V} \exp(f(v^*) \cdot f(u))}}_{\text{Partition function gradient}} - \underbrace{\sum_{n_i \in N_S(v^*)} f(n_i)}_{\text{Neighbor gradient}}$$
$$+ \underbrace{\gamma \cdot \frac{f(v^*) - \mu_c}{\| f(v^*) - \mu_c \|_2}}_{\text{Closest cluster gradient}} \tag{8}$$

The gradient of a cluster representation is given by:

$$\frac{\partial \mathcal{L}}{\partial \mu_c} = -\gamma \cdot \sum_{v \in V_c} \frac{f(v) - \mu_c}{\| f(v) - \mu_c \|_2}. \tag{9}$$

Smoothness regularization

We add the following smoothness regularization term to the cost function:

$$\Lambda = \lambda \cdot \sum_{(v,u)\in E_S} w_{(v,u)} \cdot \|f(v) - f(u)\|_2$$

(a) A Barbell graph (b) Deepwalk reconstruction (c) Smooth reconstruction

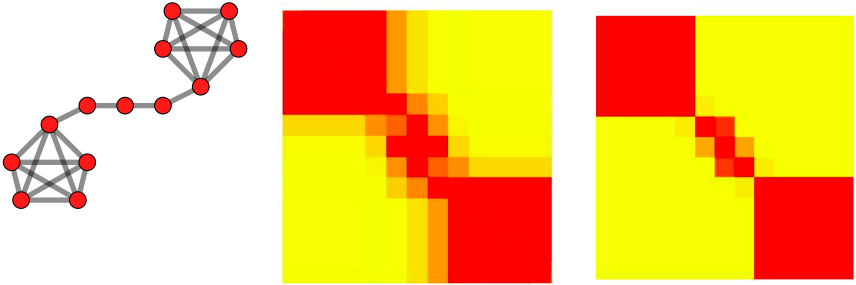


Figure 4: In the factorized target matrix we want separated communities.

Experimental results

Table 1: Statistics of the social networks used in the paper.

Source	Dataset	$\lvert V \rvert$	Density	Transitivity
Facebook	Politicians	5,908	0.0024	0.3011
	Companies	14,113	0.0005	0.1532
	Athletes	13,866	0.0009	0.1292
	Media	27,917	0.0005	0.1140
	Celebrities	11,565	0.0010	0.1666
	Artists	50,515	0.0006	0.1140
	Government	7,057	0.0036	0.2238
	TV Shows	3,892	0.0023	0.5906
Deezer	Croatia	54,573	0.0004	0.1146
	Hungary	47,538	0.0002	0.0929
	Romania	41,773	0.0001	0.0752

Node classification

Table 2: Multi-label node classification performance on the Deezer genre likes datasets.

	Croatia	Hungary	Romania
DeepWalk	0.207 (±**0.004**)	0.228 (±**0.002**)	0.186 (±**0.006**)
Node2Vec	0.235 (±**0.010**)	0.267 (±**0.011**)	0.229 (±**0.008**)
Walklets	0.270 (±**0.012**)	0.307 (±**0.006**)	0.281 (±**0.012**)
ComE	0.217 (±**0.009**)	0.246 (±**0.007**)	0.212 (±**0.006**)
M-NMF	0.217 (±**0.003**)	0.239 (±**0.011**)	0.209 (±**0.013**)
DANMF	0.210 (±**0.002**)	0.242 (±**0.008**)	0.210 (±**0.012**)
Smooth DeepWalk	0.215 (±**0.006**)	0.244 (±**0.004**)	0.204 (±**0.006**)
GEMSEC	0.212 (±**0.004**)	0.244 (±**0.004**)	0.213 (±**0.006**)
Smooth GEMSEC	0.215 (±**0.004**)	0.250 (±**0.004**)	0.215 (±**0.006**)
GEMSEC$_2$	**0.287** (±**0.005**)	0.310 (±**0.007**)	**0.289** (±**0.007**)
Smooth GEMSEC$_2$	0.276 (±**0.006**)	**0.314** (±**0.006**)	0.287 (±**0.007**)

Runtime - a synthetic example

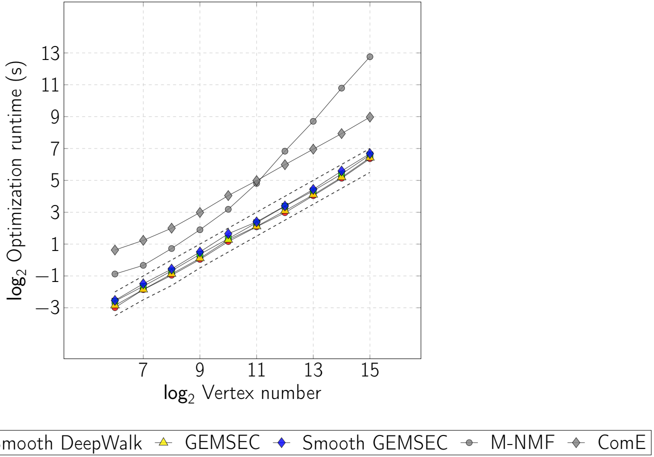


Figure 5: Sensitivity of optimization runtime to graph size measured by seconds.

Contributions and Impact

1. First community aware method to disentangle dimension number and cluster count.
2. Sequence agnostic embedding.
3. Smoothness regularization.
4. Node classification datasets released.
5. Model is widely used in graph mining education - e.g. University of Passau.

Multi-scale Attributed Node Embedding. Benedek Rozemberczki, Carl Allen, Rik Sarkar. Journal of Complex Networks. 2021.

Multi-scaling explained

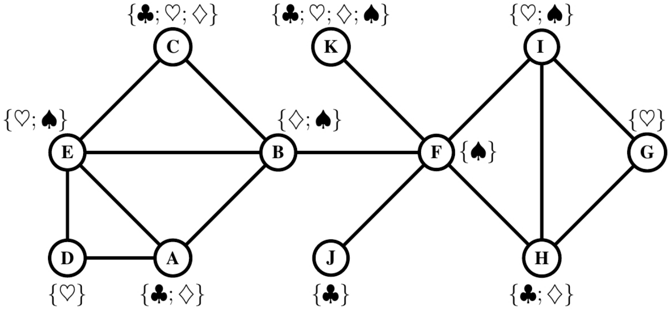


Figure 6: Attributed nodes D and G have the same feature set and their nearest neighbours also exhibit equivalent sets of features, whereas features at higher order neighbourhoods differ.

Low effective diameter

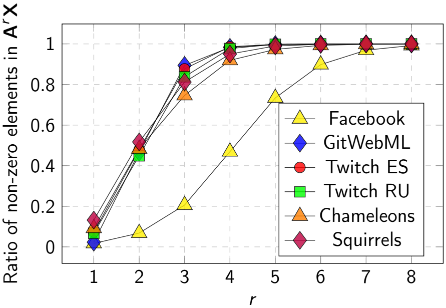


Figure 7: As the order of neighbourhoods considered (r) increases, the product of the adjacency matrix power and the feature matrix becomes less sparse.

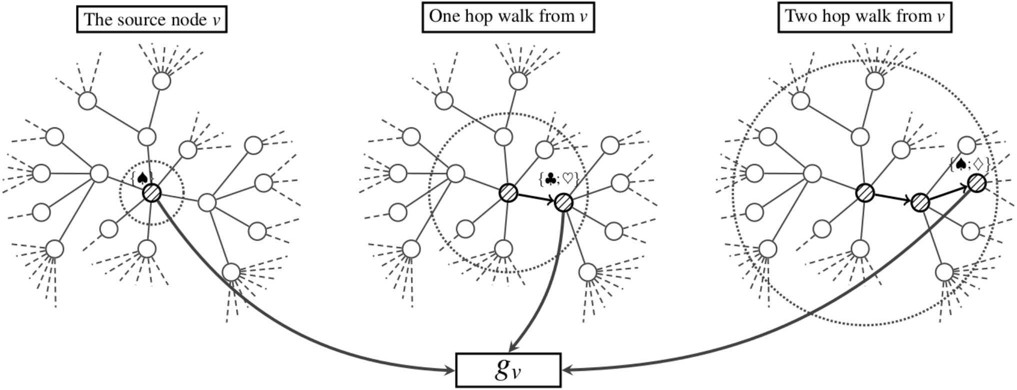


Figure 8: AE learns a pooled node embedding g_v of the source node $v \in V$ using features from the 1^{st} and 2^{nd} order proximity.

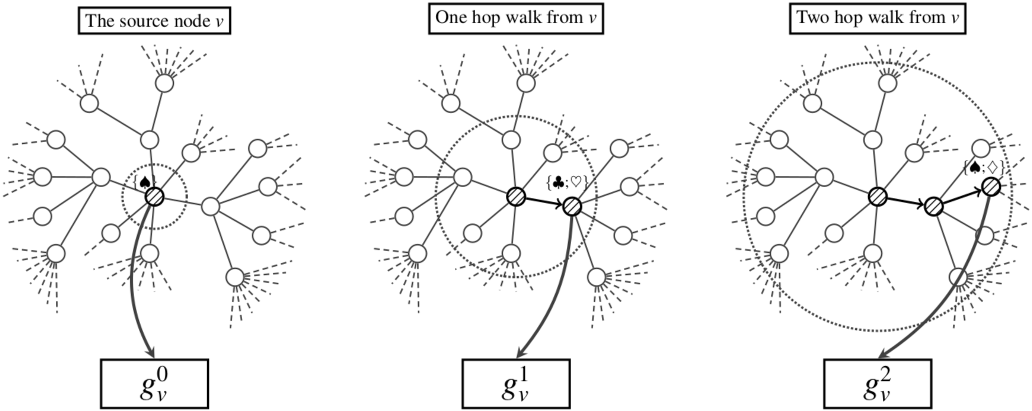


Figure 9: MUSAE learns a multi-scale embedding of the node by learning individual embeddings of for each proximity noted by g_v^0, g_v^1 and g_v^2.

Data: $\mathscr{G} = (\mathbb{V}, \mathbb{E})$ – Graph to be embedded.
$\{\mathbb{F}_v\}_{\mathbb{V}}$ – Set of node feature sets.
s – Number of sequence samples.
l – Length of sequences.
t – Context size.
d – Embedding dimension.
b – Number of negative samples.

Result: Node embedding components g^r and feature embeddings component h^r, for $r \in \{1, \ldots, t\}$.

for i in $1 : s$ **do**
  Pick $v_1 \in \mathbb{V}$ according to $P(v) \sim deg(v)/\mathrm{vol}(\mathscr{G})$.
  $(v_1, v_2, \ldots, v_l) \leftarrow$ Sample Nodes$(\mathscr{G}, v_1, l)$
  for j in $1 : l - t$ **do**
    for r in $1 : t$ **do**
      for f in $\mathbb{F}_{v_{j+r}}$ **do**
        Add the tuple (v_j, f) to multiset $\mathbb{D}_{\overrightarrow{r}}$.
      end
      for f in $\mathbb{F}_{v_j}$ **do**
        Add the tuple (v_{j+r}, f) to multiset $\mathbb{D}_{\overleftarrow{r}}$.
      end
    end
  end
end
for r in $1 : t$ **do**
  Create $\mathbb{D}_r$ by unification of $\mathbb{D}_{\overrightarrow{r}}$ and $\mathbb{D}_{\overleftarrow{r}}$.
  Run SGNS on $\mathbb{D}_r$ with b negative samples and $\frac{d}{t}$ dimensions.
  Output g_v^r, $\forall v \in \mathbb{V}$, and h_f^r, $\forall f \in \mathbb{F} = \cup_{\mathbb{V}} \mathbb{F}_v$.
end

Figure 10: The MUSAE sampling and training algorithm.

Transfer Learning

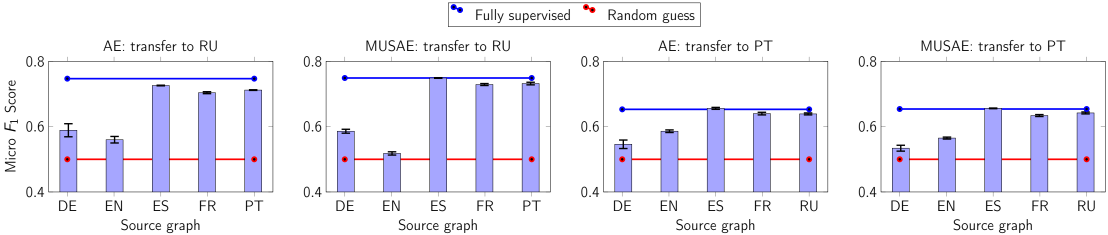


Figure 11: Transfer learning: mean micro F_1 score and standard error over 10 runs with the Twitch Portugal and Russia datasets as target and each other dataset as the source graph.

Node level regression performance

Table 3: Node attribute regression with embedding features: average test R^2 and standard error calculated from a 100 splits for predicting monthly website traffic.

	Wikipedia Chameleons	Wikipedia Crocodiles	Wikipedia Squirrels
DeepWalk	$.375 \pm .004$	$.553 \pm .002$	$.170 \pm .001$
$LINE_2$	$.381 \pm .003$	$.586 \pm .001$	$.232 \pm .002$
Node2Vec	$.414 \pm .003$	$.574 \pm .001$	$.174 \pm .002$
Walklets	$.426 \pm .003$	$.625 \pm .001$	$.249 \pm .002$
NetMF	$.440 \pm .003$	$.629 \pm .002$	$.099 \pm .002$
HOPE	$.380 \pm .002$	$.571 \pm .001$	$.175 \pm .001$
GraRep	$.520 \pm .004$	$.696 \pm .002$	$.301 \pm .001$
TADW	$.527 \pm .003$	$.636 \pm .001$	$.271 \pm .002$
AANE	$.598 \pm .007$	$.732 \pm .002$	$.287 \pm .002$
ASNE	$.440 \pm .009$	$.572 \pm .003$	$.229 \pm .005$
BANE	$.464 \pm .003$	$.617 \pm .001$	$.168 \pm .002$
TENE	$.494 \pm .020$	$.701 \pm .003$	$.321 \pm .007$
AE	$.642 \pm .006$	$\mathbf{.743} \pm \mathbf{.003}$	$.291 \pm .006$
MUSAE	$\mathbf{.658} \pm \mathbf{.004}$	$.736 \pm .003$	$\mathbf{.338} \pm \mathbf{.007}$

MUSAE and AE runtime

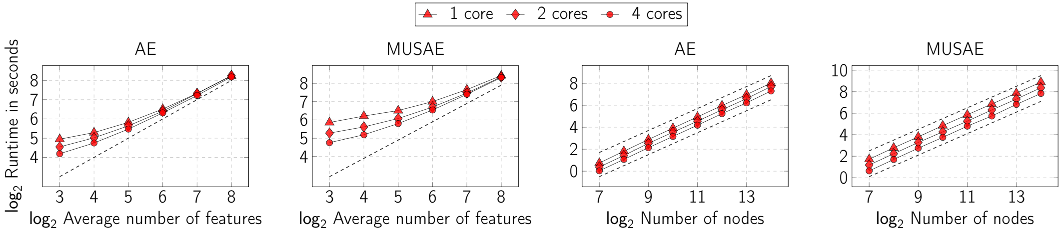


Figure 12: Training time as a function of average feature count and number of nodes. Dashed lines are linear runtime references.

Contributions and Impact

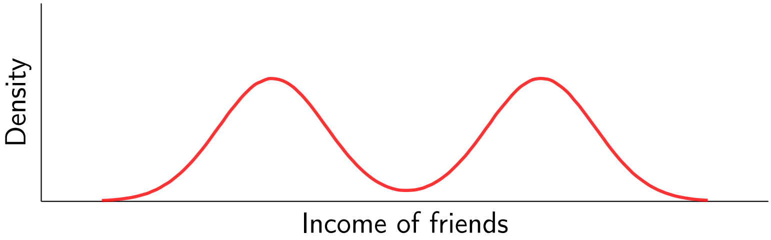
1. **Implicit attributed node embedding.**
2. **Unsupervised attributed node embedding at multiple scales.**
3. **Theoretical connection of proximity-based and attributed embeddings.**
4. **Transfer learning datasets (not from the biological domain).**
5. **Datasets introduced fostered theory research - e.g. smoothness assumptions.**

Characteristic Functions on Graphs: Birds of a Feather, from Statistical Descriptors to Parametric Models. Benedek Rozemberczki and Rik Sarkar. Proceedings of the 29th ACM International Conference on Information & Knowledge Management. 2020.

How do we extract vertex features?

- **Location**
  - Global location
  - Proximity to landmark
- **Structural features**
  - Centrality
  - Clustering coefficient
  - Presence of cyclic patterns (motifs) and trees
- **Generic vertex metadata**
  - Raw features
  - Average, median and mode of features in neighbourhood
  - Moments of features
  - Characterization of distribution

How can we characterize complicated distributions?



- **Solutions:**
  - Quantiles
  - Histogram
- **Shortcomings:**
  - Manual feature engineering (e.g. bins)
  - Non-adaptive features

The characteristic function defined on neighbourhoods

$$\mathrm{E}\left[e^{i\theta\mathbf{x}}|u\right] = \sum_{w\in V} P(w|u)\cdot e^{i\theta\mathbf{x}_w}$$

- i – Imaginary unit
- θ – Evaluation point
- $\mathbf{x}$ – Node feature
- u – Source node
- V – Set of nodes
- $P(w \mid u)$ – Affiliation probability

Weighting the characteristic function

We can exploit Euler's identity – we have to calculate:

$$\mathrm{Re}\left(\mathrm{E}\left[e^{i\theta\mathbf{x}}|u\right]\right) = \sum_{w\in V} P(w|u)\cos(\theta\mathbf{x}_w)$$

$$\mathrm{Im}\left(\mathrm{E}\left[e^{i\theta\mathbf{x}}|u\right]\right) = \sum_{w\in V} P(w|u)\sin(\theta\mathbf{x}_w)$$

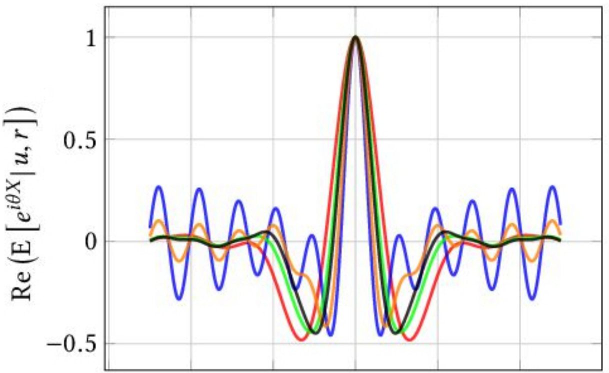
We also have to choose $P(w|u)$ – the affiliation probability:

- Jaccard similarity of neighbours
- Higher order neighbourhood overlap
- Uniform weights
- Random walks

The r-scale random walk weighted characteristic function

Let us look at a very specific example with random walk weighting:

$$\text{Re}\left(\text{E}\left[e^{i\theta\mathbf{x}}|u,r\right]\right) = \sum_{w\in V} \underbrace{P(v_{j+r}=w|v_j=u)}_{\text{Affiliation probability}}\cos(\theta\mathbf{x}_w) = \sum_{w\in V}\widehat{\mathbf{A}}^r_{u,w}\cos(\theta\mathbf{x}_w)$$



Defining node embeddings and parametric models

Node embeddings (sketches)

We evaluate the characteristic function at multiple points for all nodes:

$$\mathbf{H} = \underbrace{\widehat{\mathbf{A}}^r}_{|V|\times|V|} \cdot \underbrace{\cos(\mathbf{x} \otimes \Theta)}_{|V|\times d}$$

Parametric models (e.g. graph neural network layer)

We learn Θ with supervision – what are optimal evaluation points?

$$\mathbf{H} = f_{\Theta}(\widehat{\mathbf{A}}, r, \mathbf{x}) = \widehat{\mathbf{A}}^r \cdot \cos(\mathbf{x} \otimes \Theta)$$

Pooling

We can define *graph level* vector representations with pooling:

$$\mathbf{g} = \textbf{Invariant Pooling}(\mathbf{H})$$

Are characteristic functions useful at all?

Characteristic functions are predictive - Wikipedia

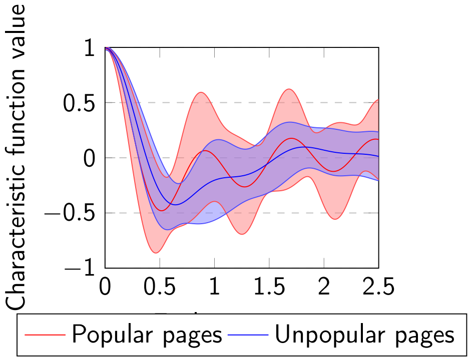


Figure 13: The real part of class dependent mean characteristic functions with standard deviations around the mean for the log transformed degree on the Wikipedia Crocodiles dataset.

Manipulating scale and granularity – Facebook

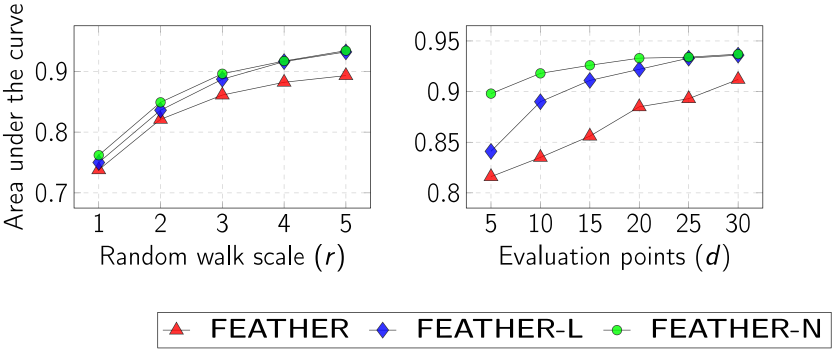


Figure 14: Mean AUC values on the Facebook page-page test set (10 seeded splits) achieved by FEATHER model variants as a function of random walk scale and characteristic function evaluation point count.

Characteristic function features are transferable – Twitch

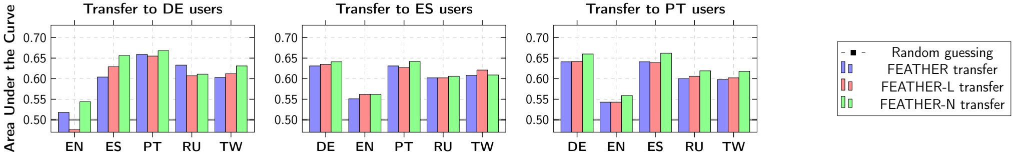


Figure 15: Transfer learning performance of FEATHER variants on the Twitch Germany, Spain and Portugal datasets as target graphs. The transfer performance was evaluated by mean AUC values calculated from 10 seeded experimental repetitions.

Node classification tasks

Table 4: Mean micro-averaged AUC values on the test set with standard errors on the node level datasets calculated from 10 seed train-test splits.

	Wikipedia Crocodiles	**Facebook Page-Page**	**LastFM Asia**	**Deezer Europe**
GCN	.924 ± .001	**.984 ± .001**	.962 ± .001	.632 ± .001
GAT	.917 ± .002	**.984 ± .001**	.956 ± .001	.611 ± .002
GraphSAGE	.916 ± .001	**.984 ± .001**	.955 ± .001	.618 ± .001
ClusterGCN	.922 ± .001	.977 ± .001	.944 ± .002	.594 ± .002
APPNP	.900 ± .001	**.986 ± .001**	**.968 ± .001**	.667 ± .001
MixHop	.928 ± .001	.976 ± .001	.956 ± .001	**.682 ± .001**
SGConv	.889 ± .001	.966 ± .001	.957 ± .001	.647 ± .001
FEATHER	**.943 ± .001**	**.981 ± .001**	**.954 ± .001**	**.651 ± .001**
FEATHER-L	**.944 ± .002**	**.984 ± .001**	.960 ± .001	.656 ± .001
FEATHER-N	**.947 ± .001**	**.987 ± .001**	**.970 ± .001**	.673 ± .001

Contributions and Impact

1. **Generalization of characteristic functions.**
2. **Trainable evaluation points.**
3. **Definition of parametric models.**
4. **Proofs of robustness and permutation invariance.**
5. **Release of node classification datasets.**
6. **State-of-the-art on MalNet. See: Scott Freitas, Yuxiao Dong, Joshua Neil, Duen Horng Chau. A Large-Scale Database for Graph Representation Learning. (2020)**

Pathfinder Discovery Networks for Neural Message Passing. Benedek Rozemberczki, Peter Englert, Amol Kapoor, Martin Blais, Bryan Perozzi. Proceedings of the Web Conference. 2021.

The problem that we solve

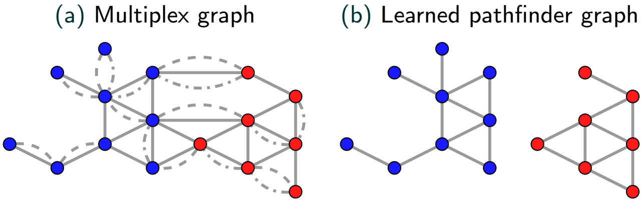


Figure 16: Pathfinder discovery networks take multiple sets of weighted edges and learn a graph specifically suited to a downstream predictive task.

A general overview

- We define a neural network layer which can aggregate edges in multiplex graphs.
- We look at theoretical properties of the layer.
- We show how existing layers can be seen as a special case of this layer.
- We evaluate the models by node classification tasks.

Pathfinder neuron

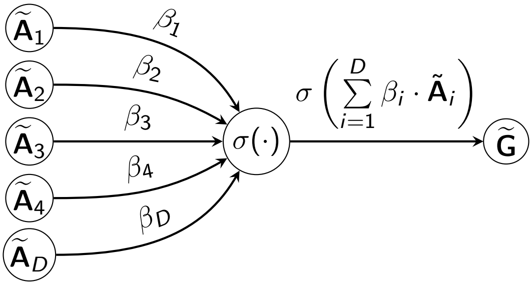


Figure 17: Architecture of a single pathfinder neuron.

Pathfinder layer and pathfinder discovery network (PDN)

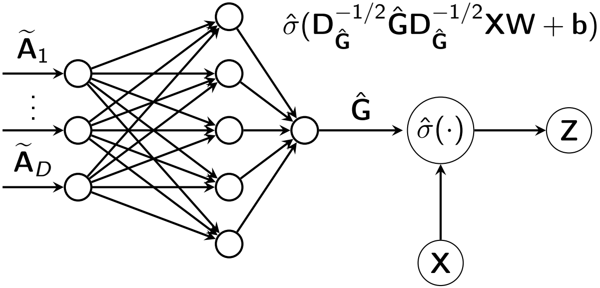


Figure 18: A PDN consisting of one hidden pathfinder layer and a GCN

The expressiveness of the model

Edge State		PDN activations		
$\mathbf{A}'_{u,v}$	$\mathbf{A}''_{u,v}$	h_1	h_2	$\alpha_{u,v}$
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	2	1	0

Let us look at a forward pass example:

$$h_1 = \mathrm{ReLU}(\mathbf{A}'_{u,v} + \mathbf{A}''_{u,v})$$

$$h_2 = \mathrm{ReLU}(\mathbf{A}'_{u,v} + \mathbf{A}''_{u,v} - 1)$$

$$\alpha_{u,v} = h_1 - 2 \cdot h_2$$

Non-diminishing edge weights

The Graph Attention edge weights will diminish:

$$\lim_{|N(u)|\to\infty} \alpha_{u,v} = \lim_{|N(u)|\to\infty} \frac{\exp(f_\theta(\mathbf{H}_{u,:};\mathbf{H}_{v,:}))}{\sum\limits_{w\in N(u)} \exp(f_\theta(\mathbf{H}_{u,:};\mathbf{H}_{w,:}))} = 0. \tag{10}$$

The PDN edge weights do not diminish:

$$\lim_{|N(u)|\to\infty} \alpha_{u,v} = \beta_1 \cdot \mathbf{A}'_{u,v} + \beta_2 \cdot \mathbf{A}''_{u,v}. \tag{11}$$

Diffusion convolutions are PDNs

Data: $\widetilde{\mathbf{A}}$ - Normalized adjacency matrix
$\mathbf{X}$ - Feature matrix
D - Order of adjacency matrix powers
d - Number of filters

Result: $\mathbf{Z}$ – Hidden state matrix

$\mathbf{Z} \leftarrow$ Initialize representations(d).
$\mathbf{Z}_0 \leftarrow \mathbf{XW}$
for $i \in \{1, \ldots, D\}$ **do**
    $\mathbf{Z}_i \leftarrow \widetilde{\mathbf{A}}\mathbf{Z}_{i-1}$
    $\mathbf{Z} \leftarrow \mathbf{Z} + P_i \cdot \mathbf{Z}_i$
end

Algorithm 1: Efficient sparsity aware forward pass multi-scale mixing with a softmax learned graph and a linear graph convolutional activation function.

Predictive performance on synthetic graphs

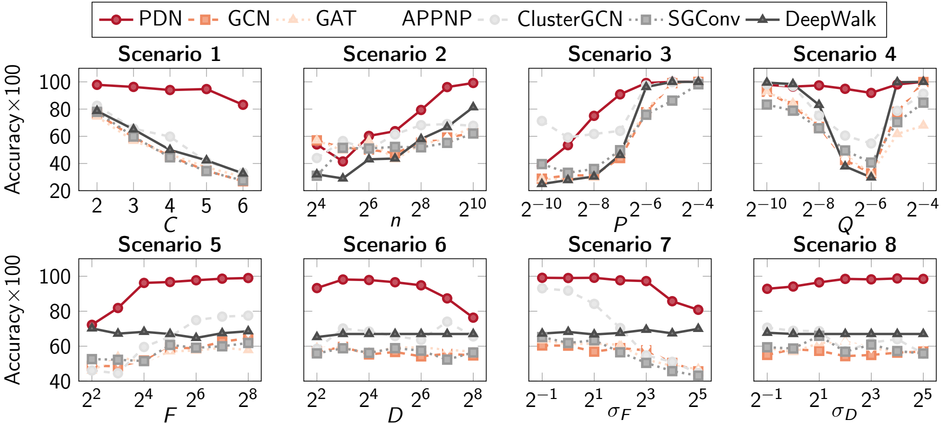


Figure 19: Node classification performance measured by average test set accuracy (10 experimental repetitions) on the synthetically generated attributed graphs

Relative runtime

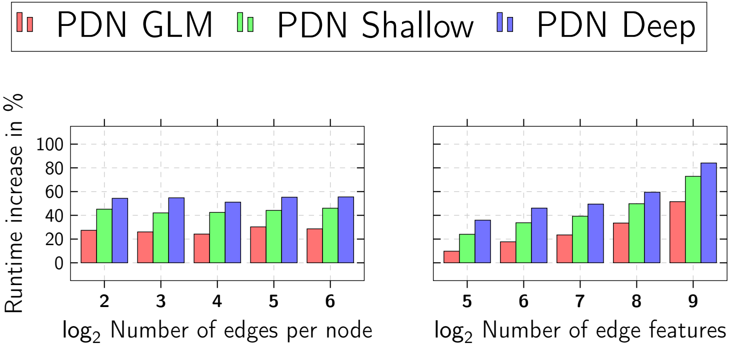


Figure 20: The relative runtime increase (compared to spectral graph convolutions) needed for training PDN models on synthetic datasets.

Contributions and Impact

1. **Pathfinder neuron.**
2. **Pathfinder layer.**
3. **Pathfinder discovery network.**
4. **Model variations and theoretical properties.**
5. **Part of the Google AI Research graph convolutions library.**

The Shapley Value of Classifiers in Ensemble Games. Benedek Rozemberczki and Rik Sarkar. Under Review in the 30th ACM International Conference on Information & Knowledge Management. 2021.

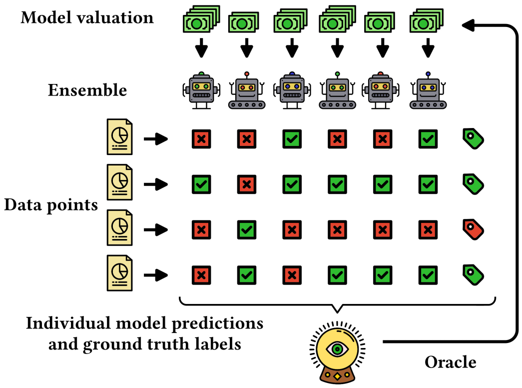


Figure 21: An overview of the model valuation problem. Models in the ensemble receive a set of data points and score those. Using the predictions and ground truth labels the oracle quantifies the worth of models.

Table 5: Comparison of Shapley value computation and approximation techniques in terms of having (✔) and missing (✘) desiderata; complexities with respect to the number of players m and permutations p.

Method	Voting	Bound	Non-Random	Space	Time
Explicit	✔	✔	✔	$\mathcal{O}(m)$	$\mathcal{O}(m!)$
MC	✘	✔	✘	$\mathcal{O}(m)$	$\mathcal{O}(mp)$
TMC	✘	✔	✘	$\mathcal{O}(m)$	$\mathcal{O}(mp)$
MLE	✘	✘	✔	$\mathcal{O}(m)$	$\mathcal{O}(m)$
MMLE	✘	✘	✔	$\mathcal{O}(m)$	$\mathcal{O}(m!)$
EMC	✔	✔	✔	$\mathcal{O}(m)$	$\mathcal{O}(m^2)$

Table 6: Comparison of ensemble pruning and building techniques in terms of having (✔) and missing (✘) desiderata.

Method	Agnostic	Set based	Diverse	Bidirectional
Greedy	✔	✔	✘	✔
WV-LP	✔	✔	✘	✘
RE	✔	✔	✘	✘
DREP	✔	✔	✔	✘
COMEP	✘	✔	✔	✘
EP-SDP	✔	✔	✘	✘
CART SySM	✘	✘	✔	✘
Troupe (ours)	✔	✔	✔	✔

Ensemble games.

Definition. ***Ensemble game.*** Let $\mathcal{M}$ be a set of binary classifiers. An ensemble game for a labeled data point $(y, \mathbf{x})$ is then a co-operative game $G = (\mathcal{M}, v)$ in which:

$$v(\mathcal{S}) = \begin{cases} 1 & \text{if } w(\mathcal{S}) \geq \gamma, \\ 0 & \text{otherwise.} \end{cases}$$

where $w(\mathcal{S}) = \sum_{M \in \mathcal{S}} w_M$ for any sub-ensemble $\mathcal{S} \subseteq \mathcal{M}$ and threshold $0 \leq \gamma \leq 1$.

Definition. ***Shapley value.*** The Shapley value of binary classifier M in the ensemble $\mathcal{M}$, for the data point level ensemble game $G = (\mathcal{M}, v)$ is defined as

$$\Phi_M(v) = \sum_{\mathcal{S} \subseteq \mathcal{M} \setminus \{M\}} \frac{|\mathcal{S}|!\ (|\mathcal{M}| - |\mathcal{S}| - 1)!}{|\mathcal{M}|!} (v(\mathcal{S} \cup \{M\}) - v(\mathcal{S})).$$

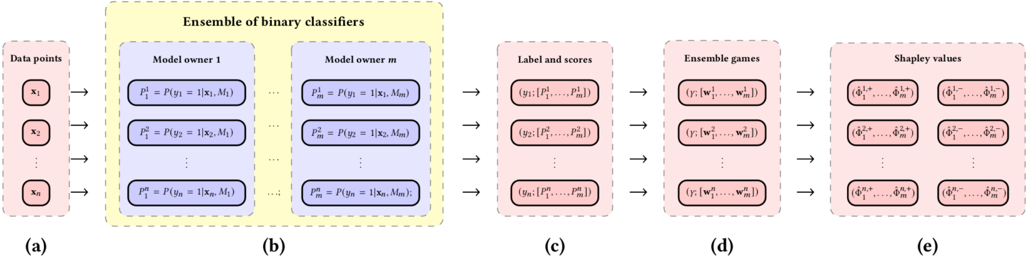


Figure 22: The approximate average Shapley value calculation pipeline.

Approximation quality

Table 7: Absolute percentage error of average conditional Shapley values obtained by approximation techniques (rows) for the graph classifiers (columns) in the ensemble game. Bold numbers note the lowest error on each dataset – classifier pair.

	Approximation	FEATHER	Graph2Vec	GL2Vec	NetLSD	SF	LDP	GeoScatter	IGE	FGSD
Reddit	Troupe	**1.23**	**2.35**	8.18	**0.99**	**2.64**	**2.31**	**1.64**	**4.85**	**1.49**
	MLE	3.20	23.61	32.62	4.19	5.34	7.97	7.12	5.42	7.61
	MC $p = 10^3$	12.57	30.94	13.26	8.67	32.76	12.32	11.62	16.36	12.78
	MC $p = 10^3$	4.71	5.38	**3.41**	1.67	3.03	5.51	4.34	4.97	3.82
Twitch	Troupe	**0.28**	**3.33**	**1.18**	**2.53**	**1.62**	**0.59**	**1.48**	**0.25**	1.19
	MLE	5.22	5.44	3.05	8.32	2.38	1.92	3.14	4.85	5.77
	MC $p = 10^2$	2.37	10.40	6.76	7.07	15.79	6.36	13.99	23.96	**0.39**
	MC $p = 10^3$	2.32	4.60	2.96	2.67	2.53	2.73	6.31	0.27	3.89
GitHub	Troupe	**2.68**	**0.18**	**2.61**	**1.41**	**1.49**	1.23	**1.88**	**2.36**	1.04
	MLE	9.22	5.12	3.76	3.27	9.71	7.46	5.08	4.04	0.73
	MC $p = 10^2$	5.91	9.37	4.67	9.82	8.78	8.34	13.66	28.95	**0.76**
	MC $p = 10^3$	3.35	6.09	7.70	3.26	2.84	**0.79**	6.67	2.51	1.12

Normalized value of complex models

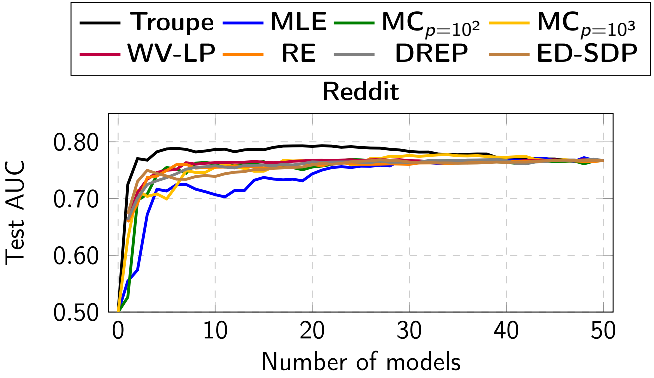


Figure 23: The classification performance of ensembles as a function of ensemble size selected by our forward model building procedure and baselines.

Adversarial model valuation

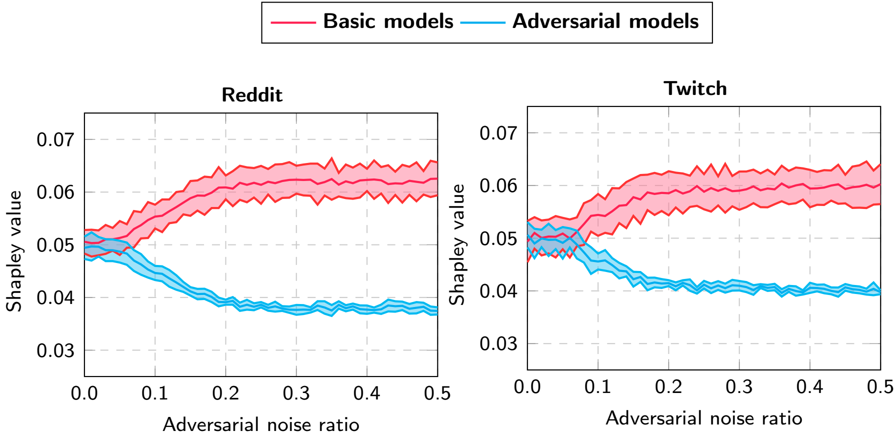


Figure 24: The mean Shapley value (standard errors added) of adversarial and base classifiers in ensemble games as a function of adversarial noise ratio mixed to predictions.

Normalized value of complex models

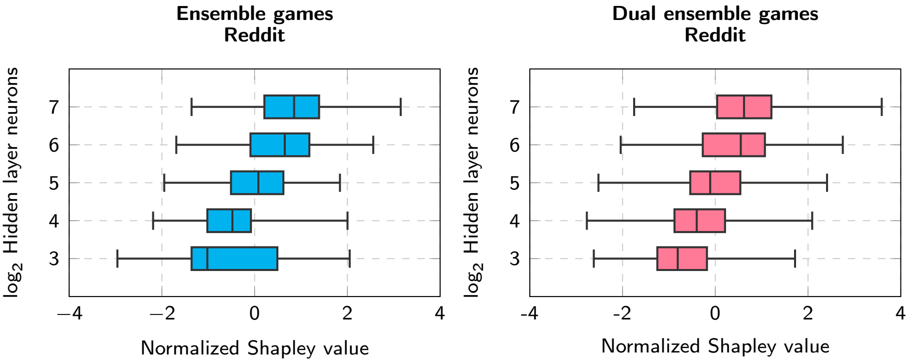


Figure 25: The distribution of normalized Shapley values for neural networks in ensemble and dual ensemble games conditional on the number of neurons.

Approximation runtime

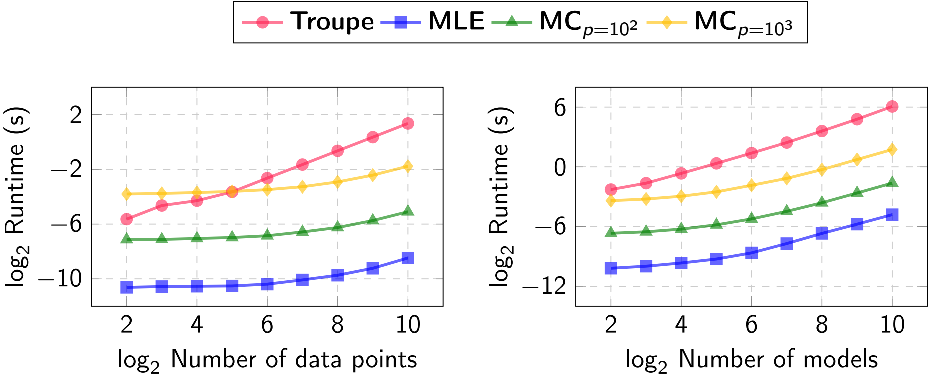


Figure 26: The runtime of Shapley value approximation in ensemble games as a function of dataset size and number of classifiers in the ensemble.

Contributions

1. **Ensemble games.**
2. **Model valuation in the ensemble.**
3. **Average Shapley value algorithm.**
4. **Ensemble pruning and model selection.**
5. **Model complexity and relative value relationship.**

Software for Graph Mining

Karate Club: An API Oriented Open-source Python Framework for Unsupervised Learning on Graphs. Benedek Rozemberczki, Oliver Kiss, Rik Sarkar. Proceedings of the 29th ACM International Conference on Information & Knowledge Management. 2020.

A general overview

- Karate Club is a machine learning framework for graphs
- It includes approximately 40 models
- We designed it with a scikit-learn like API
- We released new datasets, a documentation, and tutorials

The methods covered

- We only included unsupervised methods
- Community detection
  - Non-overlapping
  - Overlapping
- Node embedding
  1. Proximity preserving
  2. Attributed
  3. Structural
- Whole graph embedding and statistical graph fingerprints

What were the design principles?

- Encapsulated hyperparameters
- API oriented design
- Limited number of public methods
  - Standardized data ingestion
  - Standardized output generation
- Type safety
- Easy interfacing with downstream libraries
- Python package index inclusion
- Continuous integration
- Perfect 100% test coverage

Standardized dataset ingestion and internal model mechanics

Dataset ingestion:

- Graph – NetworkX graph
- Node features – Numpy array / SciPy array
- Graphs – List of NetworkX graphs

Model mechanics:

- Graph operations – NetworkX
- Linear algebra – NumPy
- Sparse linear algebra – SciPy
- Implicit factorization – gensim

Encapsulated hyperparameters

```python
import networkx as nx
from karateclub import DeepWalk

graph = nx.gnm_random_graph(100, 1000)

model = DeepWalk()
print(model.dimensions)

model = DeepWalk(dimensions=64)
print(model.dimensions)
```

API consistency

```python
import networkx as nx
from karateclub import DeepWalk

graph = nx.gnm_random_graph(100, 1000)

model = DeepWalk()
model.fit(graph)
embedding = model.get_embedding()
```

Downstream interfacing

```
import community
import networkx as nx
from karateclub import LabelPropagation

graph = nx.gnm_random_graph(100, 1000)

model = LabelPropagation()
model.fit(graph)
lp_memberships = model.get_memberships()

print(community.modularity(scd_memberships, graph))
```

Graph level datasets

Table 8: Statistics of graph datasets used for graph level algorithms.

Dataset	Graphs	Nodes Min	Nodes Max	Density Min	Density Max	Diameter Min	Diameter Max
Reddit Threads	203,088	11	97	0.021	0.382	2	27
Twitch Egos	127,094	14	52	0.038	0.967	1	2
GitHub StarGazers	12,725	10	957	0.003	0.561	2	18
Deezer Egos	9,629	11	363	0.015	0.909	2	2

Graph classification

Table 9: Mean AUC values with standard errors on the graph level datasets calculated from 100 seed train-test splits.

	Reddit Threads	Twitch Egos	GitHub StarGazers	Deezer Egos
GL2Vec	$.753 \pm .002$	$.664 \pm .002$	$.551 \pm .001$	$.504 \pm .001$
Graph2Vec	$.804 \pm .002$	$.702 \pm .003$	$.585 \pm .001$	$.512 \pm .001$
SF	$.814 \pm .002$	$.678 \pm .003$	$.558 \pm .001$	$.501 \pm .001$
NetLSD	$\mathbf{.827 \pm .001}$	$.631 \pm .002$	$.632 \pm .001$	$.522 \pm .001$
FGSD	$.825 \pm .002$	$.705 \pm .003$	$.656 \pm .001$	$.526 \pm .001$
GeoScattering	$.800 \pm .001$	$.697 \pm .001$	$.546 \pm .003$	$.522 \pm .003$
FEATHER	$\mathbf{.830 \pm .002}$	$\mathbf{.720 \pm .003}$	$\mathbf{.748 \pm .002}$	$\mathbf{.540 \pm .001}$

How can I access Karate Club?

pip install karateclub

https://karateclub.readthedocs.io/

Contributions and Impact

1. **Coverage of unsupervised techniques.**
2. **First library to provide graph fingerprints.**
3. **New datasets for graph classification.**
4. **Used for education in graduate schools e.g. University of Michigan.**

Little Ball of Fur: A Python Library for Graph Sampling. Benedek Rozemberczki, Oliver Kiss, Rik Sarkar. Proceedings of the 29th ACM International Conference on Information & Knowledge Management. 2020.

A general overview

- Little Ball of Fur is a Python library for graph sampling
- It includes more than 20 graph sampling algorithms
- We designed it with a scikit-learn like API
- Little Ball of Fur supports NetworkX and NetworKit

What were the design principles?

- Encapsulated hyperparameters
- Backed wrapper based, API oriented design
- Limited number of public methods
  - Standardized data ingestion
  - Standardized output generation
- Easy interfacing with downstream libraries
- Python package index inclusion
- Continuous integration
- Perfect 100% test coverage

Encapsulated hyperparameters

```python
from littleballoffur import RandomWalkSampler

sampler = RandomWalkSampler()
print(sampler.seed)

sampler = RandomWalkSampler(seed=41)
print(sampler.seed)
```

API consistency and downstream interfacing

```
import networkx as nx
from littleballoffur import RandomWalkSampler

graph = nx.watts_strogatz_graph(1000, 10, 0)

sampler = RandomWalkSampler()
sampled_graph = sampler.sample(graph)

print(nx.transitivity(sampled_graph))
```

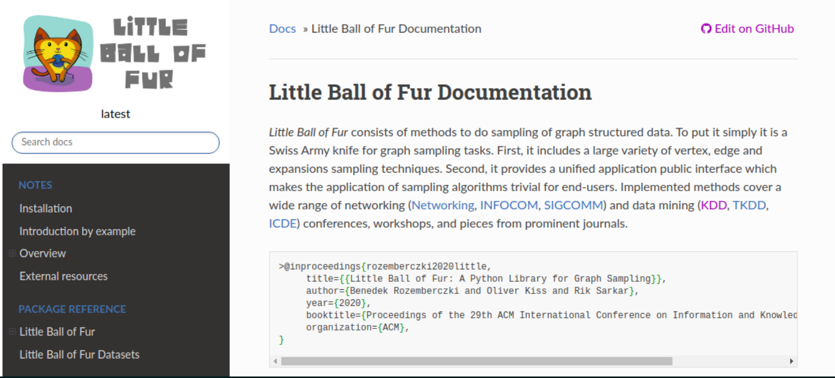
API consistency and downstream interfacing

```python
import networkx as nx
from littleballoffur import ForestFireSampler

graph = nx.watts_strogatz_graph(1000, 10, 0)

sampler = ForestFireSampler()
sampled_graph = sampler.sample(graph)

print(nx.transitivity(sampled_graph))
```

How can I access the documentation of Little Ball of Fur?

pip install littleballoffur

https://little-ball-of-fur.readthedocs.io/



Contributions and Impact

1. **First sampling library.**
2. **Node sampling.**
3. **Edge sampling.**
4. **Exploration sampling.**
5. **Became research tool standard e.g. WWW' 21.**

Summary

General Summary of Contributions

1. Enforcing clustering in proximity-based node embedding spaces.
2. Unsupervised learning of multi-scale attributed node representations.
3. Generalization of characteristic functions.
4. Supervised aggregation of multiplex graphs for message passing.
5. Class of ensemble games and solution by Shapley value.
6. Karate Club - software for graph mining.
7. Little Ball of Fur - software for graph sampling.

Outlook and Extensions

- Multi-resolution latent space clustering.
- Multi-scale attributed embeddings with feature space subsampling.
- Large-scale use of pathfinder like models.
- Characteristic function for permutation invariant pooling.
- Hierarchical ensemble games and market design.
- Software for spatiotemporal learning.

Thank you for the kind attention!