Joint Network and Edge-To-Vertex Graph Embedding

Ilya Makarov, Ksenia Korovina and L.E. Zhukov

HIGHER SCHOOL OF ECONOMICS
NATIONAL RESEARCH UNIVERSITY

Our survey

- We consider new formulation of **graph embedding algorithm**, while learning nodes and edges vector representation under common constraints.

- We evaluate our approach on **link prediction** stated as a binary classification on features of pairs of nodes, and show performance on other ML problems on graphs.

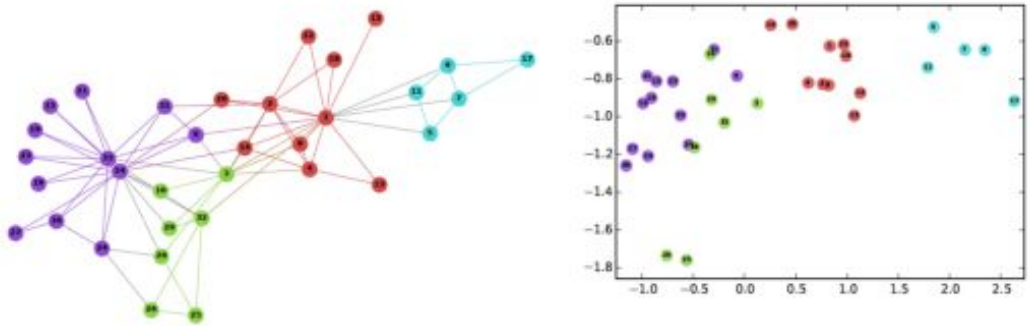
- We compare our model with existing **structural and attributive network embeddings**.

Node emdedding

- Machine learning on graphs require feature engineering of relational data.
- Mostly, manual feature engineering made by domain expert for particular problem was made.
- Nowadays, methods of automated task-independent graph feature engineering found wide broad of applications for machine learning problems on graphs.

Node emdedding

http://snap.stanford.edu/proj/embeddings-www/



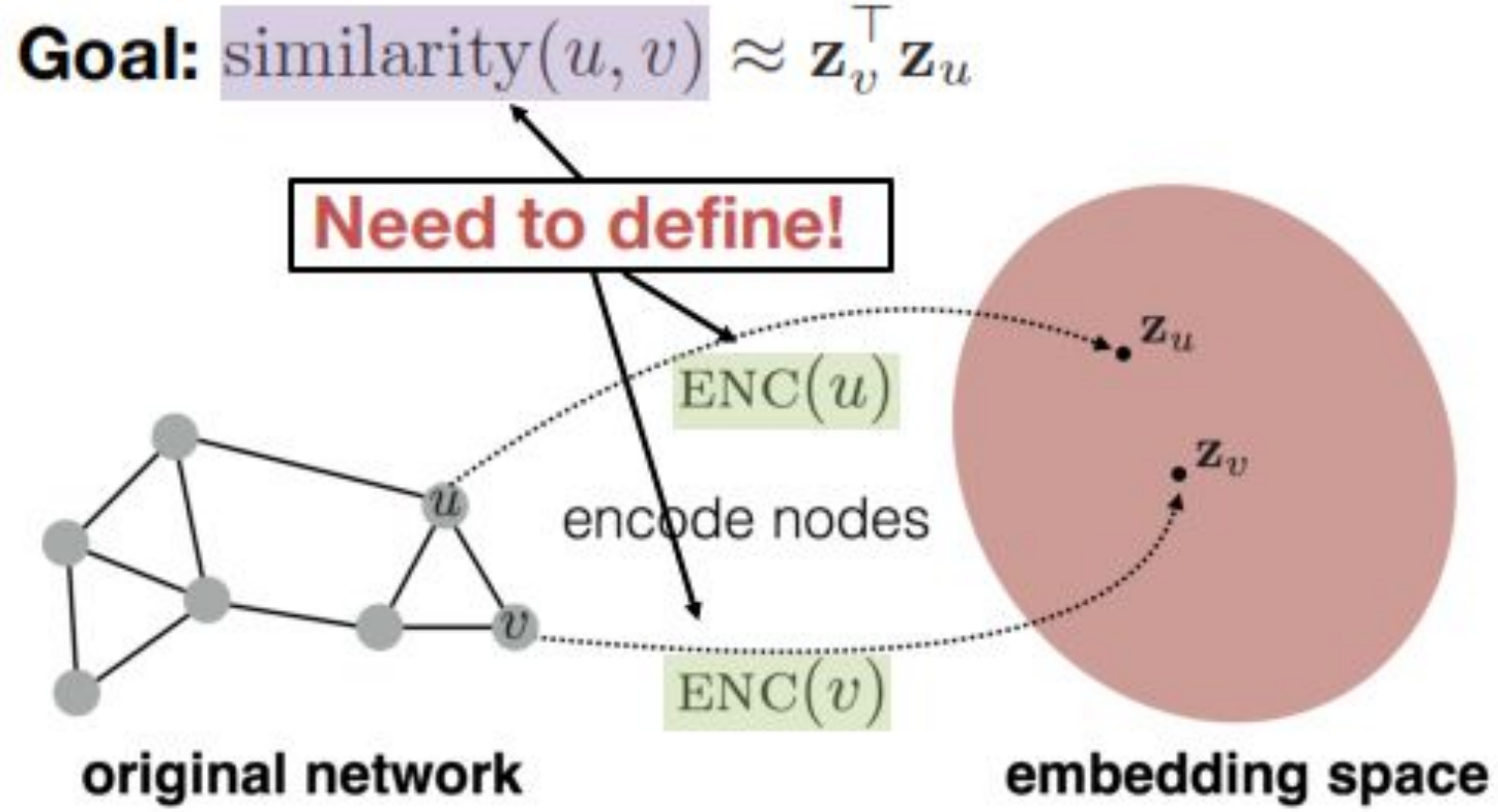
Intuition: Find embedding of nodes to d-dimensions so that "similar" nodes in the graph have embeddings that are close together.

Definition Graph embedding is a mapping from a collection of substructures (most commonly either all nodes, or all edges, or certain subgraphs) to $\mathbb{R}^d$

We will mostly consider node embeddings:

$$f : V \to \mathbb{R}^d, \; d \ll |V|.$$

Node emdedding



Node embedding

1. **Define an encoder** (i.e., a mapping from nodes to embeddings)
2. **Define a node similarity function** (i.e., a measure of similarity in the original network).
3. **Optimize the parameters of the encoder so that:**

$$\text{similarity}(u, v) \approx \mathbf{z}_v^\top \mathbf{z}_u$$

Random-walk Node emdedding

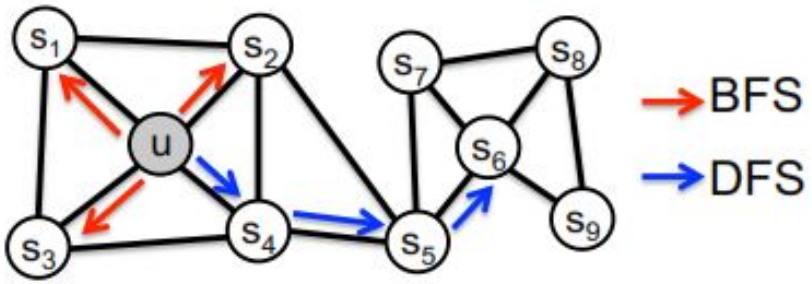
$$\mathbf{z}_u^\top \mathbf{z}_v \approx$$

probability that u and v co-occur on a random walk over the network

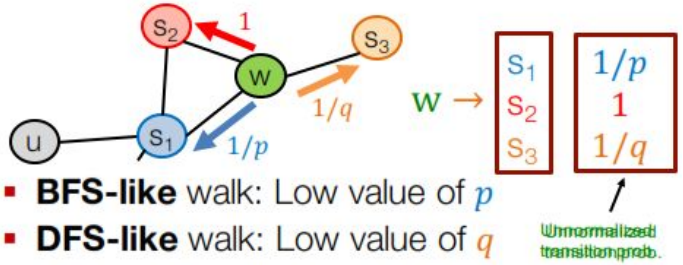
1. Run short random walks starting from each node on the graph using some strategy R.
2. For each node u collect $\mathbf{N}_R(u)$, the multiset* of nodes visited on random walks starting from u.
3. Optimize embeddings to according to:

$$\mathcal{L} = \sum_{u \in V} \sum_{v \in N_R(u)} -\log(P(v|\mathbf{z}_u))$$

Two classic strategies to define a neighborhood $N_R(u)$ of a given node



- Walker is at w. Where to go next?



- **BFS-like** walk: Low value of p
- **DFS-like** walk: Low value of q

$N_S(u)$ are the nodes visited by the walker

$N_{BFS}(u) = \{s_1, s_2, s_3\}$ Local microscopic view

$N_{DFS}(u) = \{s_4, s_5, s_6\}$ Global macroscopic view

Graph CNN

- V is the vertex set.
- A is the adjacency matrix (assume binary).
- $\mathbf{X} \in \mathbb{R}^{m \times |V|}$ **is a matrix of node features.**
  - Categorical attributes, text, image data

Basic Neighborhood Aggregation

$$\mathbf{h}_v^k = \sigma\left(\mathbf{W}_k \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|} + \mathbf{B}_k \mathbf{h}_v^{k-1}\right)$$

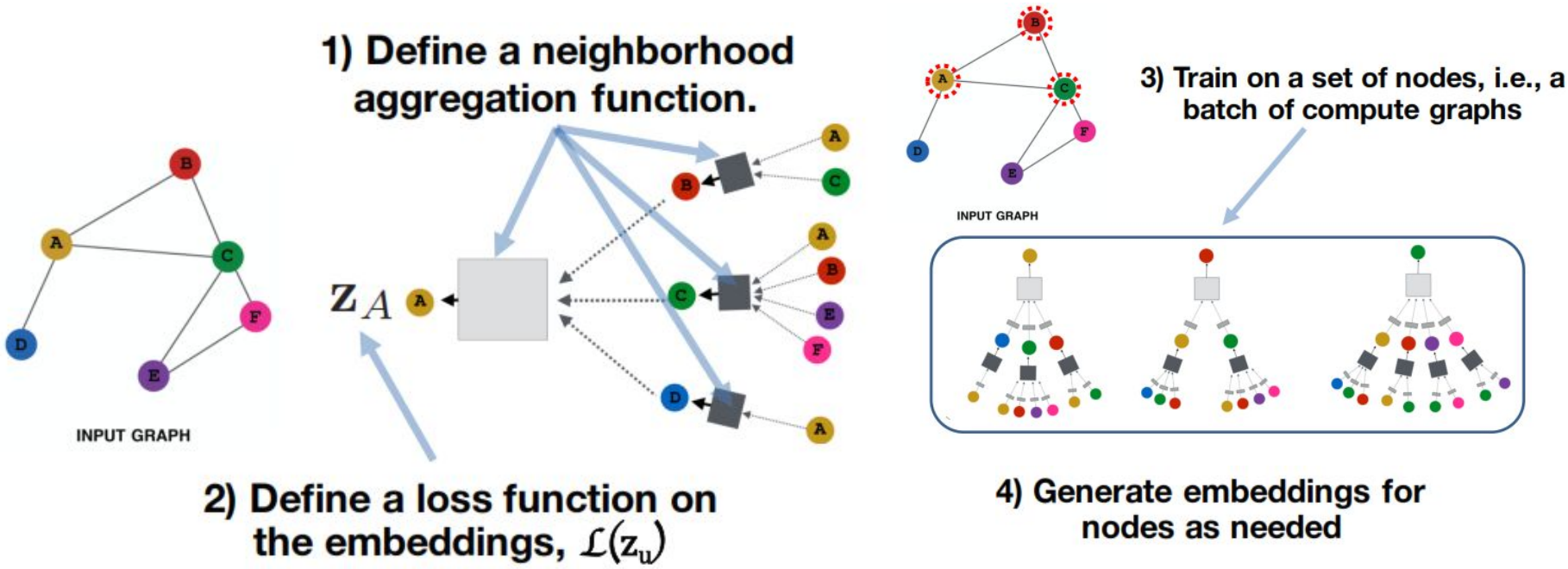
VS.

GCN Neighborhood Aggregation

$$\mathbf{h}_v^k = \sigma\left(\mathbf{W}_k \sum_{u \in N(v) \cup v} \frac{\mathbf{h}_u^{k-1}}{\sqrt{|N(u)||N(v)|}}\right)$$

same matrix for self and neighbor embeddings

per-neighbor normalization



Edge Embeddings

For edge embedding authors **applied specific component-wise functions representing edge to node embeddings** for source and target nodes of a given edge.

Table 1: Binary operators for computing vectorized (u, v)-edge representation based on node attribute embeddings $f(x)$ for ith component for $f(u, v)$. Parameter $\alpha \geqslant 1$

Symmetry Operator	Definition
Average (AVG)	$\frac{f_i(u)+f_i(v)}{2}$
Hadamard (MULT)	$f_i(u) \cdot f_i(v)$
Weighted-L_α (WL_α)	$\lvert f_i(u) - f_i(v)\rvert^\alpha$
Neighbor Weighted-L_α (NWL_α)	$\left\lvert \frac{\sum_{w\in N(u)\cup\{u\}} f_i(w)}{\lvert N(u)\rvert+1} - \frac{\sum_{t\in N(v)\cup\{v\}} f_i(t)}{\lvert N(v)\rvert+1} \right\rvert^\alpha$
Neighbor Weighted-L^0_α (NWL^0_α)	$\left\lvert \frac{\sum_{w\in N(u)} f_i(w)}{\lvert N(u)\rvert} - \frac{\sum_{t\in N(v)} f_i(t)}{\lvert N(v)\rvert} \right\rvert^\alpha$
Neighbor Double Weighted-L_α ($NDWL_\alpha$)	$\left\lvert \frac{\sum_{w\in N(u)\cup\{u\}} f_i(w)\cdot\omega(u,w)}{\sum_{p\in N(u)\cup\{u\}} \omega(u,p)} - \frac{\sum_{t\in N(v)\cup\{v\}} f_i(t)\cdot\omega(v,t)}{\sum_{s\in N(v)\cup\{v\}} \omega(v,s)} \right\rvert^\alpha$

Link prediction

Usually formulated as **binary classification task**, we take certain percentage of edges as positive examples, and use **negative sampling** for negative ones.

Baseline in this task is shown in the Table:

Similarity metric	Definition
Common Neighbors	$\lvert N(u) \cap N(v) \rvert$
Jaccard Coefficient	$\frac{\lvert N(u) \cap N(v) \rvert}{\lvert N(u) \cup N(v) \rvert}$
Adamic-Adar Score	$\sum_{w \in N(u) \cap N(v)} \frac{1}{\ln \lvert N(w) \rvert}$
Preferential Attachment	$\lvert N(u) \rvert \cdot \lvert N(v) \rvert$
Graph Distance	length of shortest path between u and v
Metric score	$\frac{1}{1+\lvert\lvert x-y \rvert\rvert}$
Cosine score	$\frac{(x,y)}{\lvert\lvert x \rvert\rvert \lvert\lvert y \rvert\rvert}$
Pearson Coefficient	$\frac{cov(x,y)}{\sqrt{cov(x,x)} \cdot \sqrt{cov(y,y)}}$
Generalized Jaccard	$\frac{\sum min(x_i,y_i)}{\sum max(x_i,y_i)}$

Our Model: Preliminaries

Definition Given a graph G = (V; E) defined as a set of vertices V and a set of edges E, we denote by G* = (V*; E*) Edge-to-vertex Dual (Line) graph, the nodes of which are the edges of G and edges are nodes, so that two adjacent nodes are connected by an edge if corresponding edges have a common node incident to them.

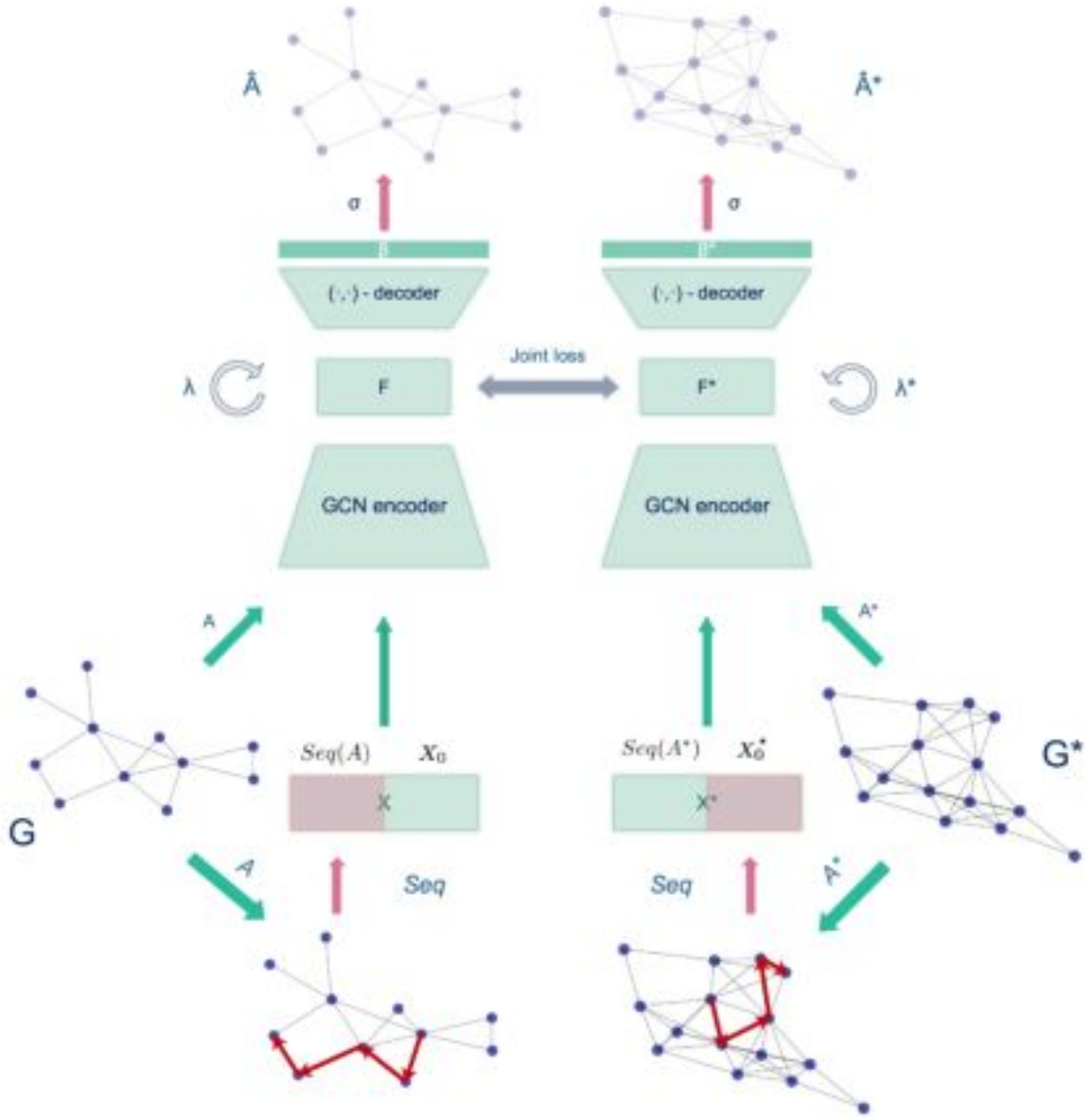
Definition The **first-order proximity** describes the proximity between vertices presented by edge weight **Aij**.

A **neighborhood** of vertex is defined as a set of vertices adjacent to it (vertex itself is not a part of its neighborhood).

The **second-order proximity** between a pair of vertices describes the similarity measure between their neighborhood structures with respect to a selected proximity measure.

Our Model

- Coupled graph autoencoders for G and G*
- Structural feature generation
- Joint loss constraint for embeddings for G and G*
- Regularizations
- Support weights, attributes, labels for both, nodes and edges



Our Model: Input and Feature Generation

As input to JONNEE, a graph $G = (V, E)$, represented with an adjacency matrix $A \in R_+^{|V|\times|V|}$ (binary or non-negative valued for weighted graphs) and optionally node attributes $X \in \mathbb{R}^{|V|\times d_0}$ and edge attributes $X^* \in \mathbb{R}^{|E|\times d_0^*}$ are available. Prior to learning, we obtain the graph dual $G^* = (E, E')$, $E' \subseteq E \times E$

During the first stage, the algorithm linearizes the graph using an algorithm $Seq \in \{\text{node2vec}, \text{diff2vec}\}$, which can be either random walks or diffusion. Seq learns features $X_1 = Seq(G, u)$ and $X_1^* = Seq(G^*, u)$ of a size u for G and u^* for G^* separately. If node and dual node (edge) features X_0 and X_0^* are available from the outset, they are concatenated with these learned sequence embeddings to form augmented feature sets:

$$X = [X_0, X_1] \in \mathbb{R}^{V\times D},\ X^* = [X_0^*, X_1^*] \in \mathbb{R}^{E\times D^*}$$

Our Model: Graph Autoencoders

Succeeding feature generation, the second stage receives as input the data A, X, and dual data A^*, X^*. Two graph autoencoders GAE and GAE^* are then trained on this data to learn hidden representations $F \in \mathbb{R}^{|V| \times d}$ and $F^* \in \mathbb{R}^{|E| \times d}$. These representations should necessarily have the same dimensionality d, so that node and edge embeddings live in the same vector space. The autoencoders have the same simple architecture that consists of an encoder GCN with two graph convolutional layers, regularized with dropout, and an β-**masked** inner-product decoder obtaining reconstruction $\hat{A}$ of the adjacency matrix A from graph embedding representations.

Our Model: Graph Autoencoders

A two-layer GCN architecture is defined as

$$H_1 = \mathrm{ReLU}(\widetilde{A}_{\mathrm{degree}}^{-\frac{1}{2}} \widetilde{A} \widetilde{A}_{\mathrm{degree}}^{-\frac{1}{2}} X W_1)$$

$$GCN(\Theta; X, A) = H_2 = \mathrm{ReLU}(\widetilde{A}_{\mathrm{degree}}^{-\frac{1}{2}} \widetilde{A} \widetilde{A}_{\mathrm{degree}}^{-\frac{1}{2}} H_1 W_2)$$

where $\widetilde{A} = A + I, A_{\mathrm{degree}_{ii}} = \sum_j \widetilde{A}_{ij}$ and $\Theta = \{W_1 \in \mathbb{R}^{d_0 \times h}, W_2 \in \mathbb{R}^{h \times d}, b_2 \in \mathbb{R}^d\}$ are trainable parameters of GCN.

Then graph and dual graph embeddings and reconstructions are specified as:

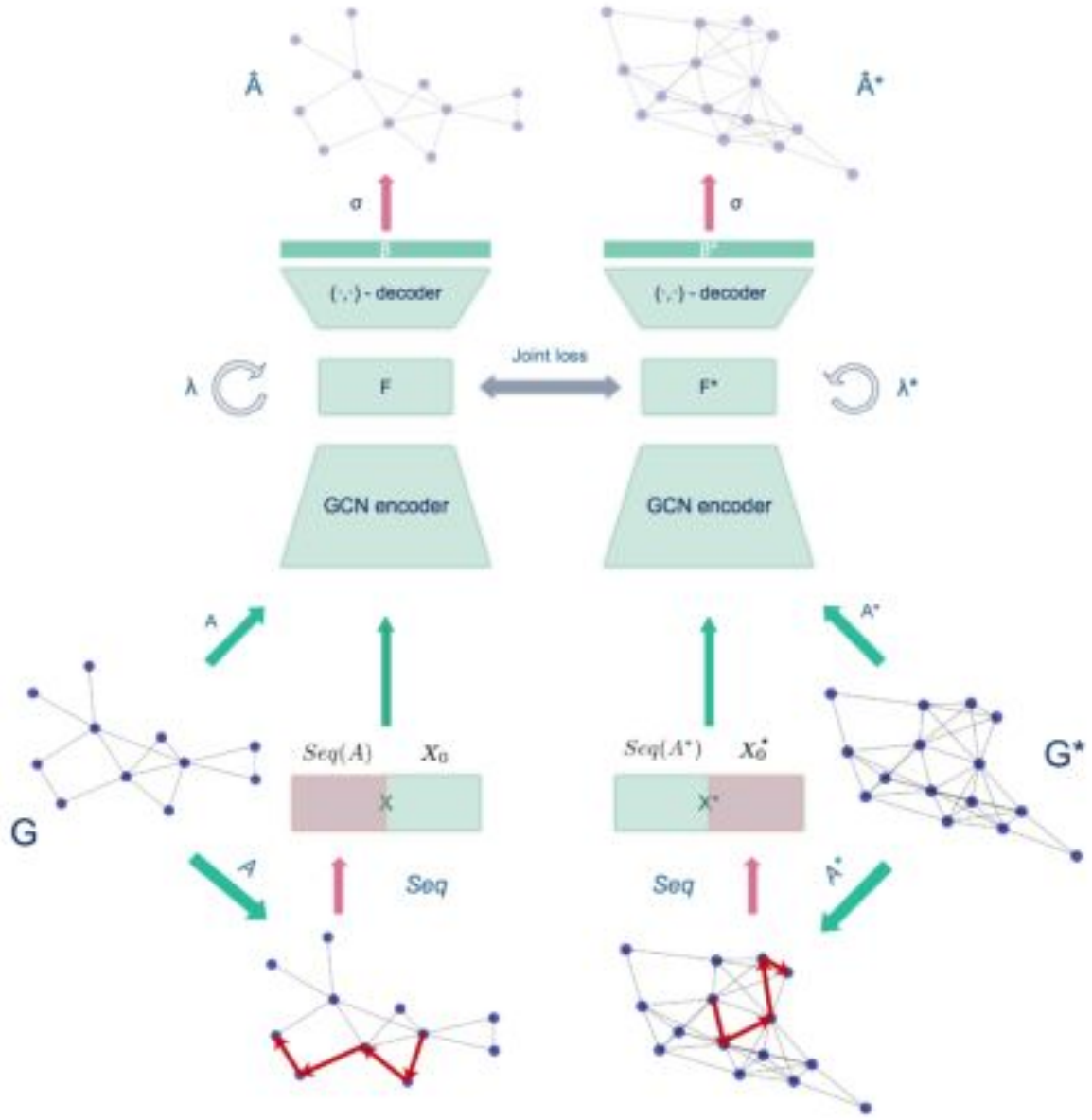
$$F = GCN(X, A), \quad \widehat{A} = \sigma\left(F F^T \odot B\right)$$

$$F^* = GCN(X^*, A^*), \quad \widehat{A^*} = \sigma\left(F^* F^{*T} \odot B^*\right)$$

Here $\odot$ denotes the Kronecker (point-wise) product of matrices. Matrices $B = (1 + \beta a_{ij})_{i,j}$ and $B^* = (1 + \beta a^*_{ij})_{i,j}$ with $\beta, \beta^* > 0$ allow us to soften learning and regularizes the model, especially on unweighted graphs.

Our Model

- **X0,X*0 -** original node/edge features
- **X1,X*1 -** learned sequential features
- **F, F* -** embedding vectors
- **A^,A*^ -** reconstructed adjacency matrices



Our Model: Loss Function

Reconstruction loss:

$$L_G = \|A - \widehat{A}\|_F^2 \sim \frac{1}{|V|^2} \sum_{i,j} (\hat{a}_{ij} - a_{ij})^2$$

$$L_{G^*} = \|A^* - \widehat{A^*}\|_F^2 \sim \frac{1}{|E|^2} \sum_{i,j} (\hat{a}^*_{ij} - a^*_{ij})^2$$

Laplacian Regularization:

$$L_{reg}(F) = \sum_{i,j} a_{ij} \left\| \frac{f_i}{d_i} - \frac{f_j}{d_j} \right\|^2 = 2\,\mathrm{tr}(F^T L F)$$

$$L^*_{reg}(F^*) = \sum_{i,j} a^*_{ij} \left\| \frac{f^*_i}{d^*_i} - \frac{f^*_j}{d^*_j} \right\|^2 = 2\,\mathrm{tr}(F^{*T} L F^*)$$

Joint constraint:

$$L_{G^* \to G} = \sum_{v \in V} \|f(v) - \frac{1}{|N^G(v)|} \sum_{u \in N(v)} f^*((u,v))\|_2^2$$

$$L_{G \to G^*} = \sum_{e=(u,v) \in E} \|f^*(e) - \frac{1}{|N^{G^*}(e)|} \sum_{t \in N^G(u) \cup N^G(v)} f(t)\|_2^2$$

Model loss (C*=0):

$$L(\Theta, \Theta^*) = L_G(\Theta) + L_{G^*}(\Theta^*) + \lambda L_{reg}(\Theta) + \lambda^* L^*_{reg}(\Theta) + C \cdot L_{G^* \to G}(\Theta, \Theta^*)$$

Datasets

Table 1: Summary of datasets

Network \ Parameters	$\mid V \mid$	$\mid E \mid$	# classes
Blog-Catalog [24]	10312	333983	2 classes, 39 labels/node
Ca-GrQc [24]	5242	14496	-
Cit-HepTh [24]	27770	352807	-
HSE [25]	1491	2947 weighted	-
Cora [9]	2708, features	5429	7 classes
PPI [24]	56944, features	818716	121 classes, 50 labels/node

Evaluation on Link Prediction

Table 11: Link Prediction evaluation

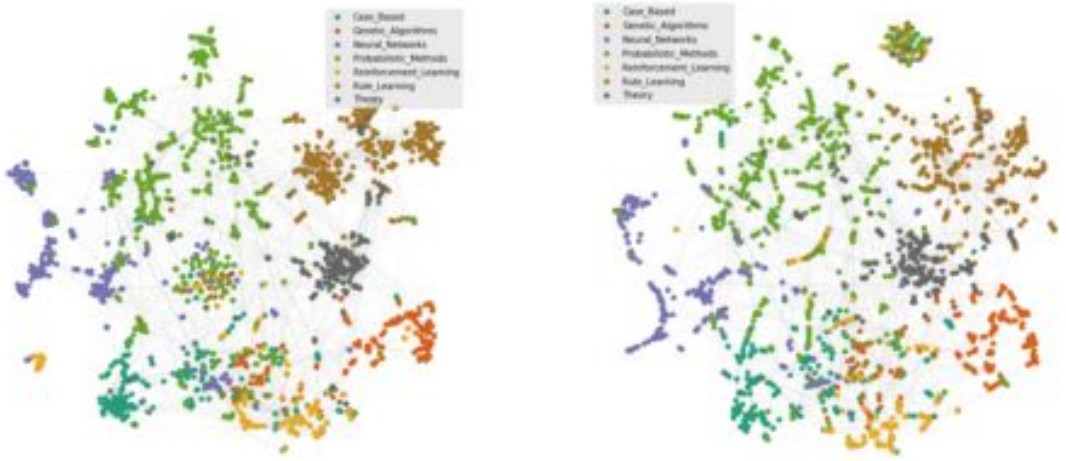
Dataset \ Model	GrF	HOPE	N2v	D2v	GAE	Ours	Metric
Cit-HepTh	0.555	0.741	0.616	0.720	0.633	**0.751**	Acc
	0.564	0.730	0.758	0.799	0.778	**0.817**	ROC
	0.554	0.759	0.757	0.817	**0.845**	0.841	AP
	0.583	0.682	0.698	0.730	0.677	**0.748**	F1
Ca-GrQc	0.596	**0.689**	0.557	0.615	0.602	0.642	Acc
	0.661	0.690	0.732	0.735	0.670	**0.797**	ROC
	0.682	0.693	0.800	0.795	0.718	**0.852**	AP
	0.631	0.557	0.657	0.644	0.63	**0.700**	F1
HSE	0.665	0.777	0.477	0.463	0.566	**0.793**	Acc
	0.745	0.781	0.674	0.751	0.678	**0.876**	ROC
	0.766	0.783	0.743	0.842	0.706	**0.906**	AP
	0.715	0.717	0.631	0.601	0.558	**0.748**	F1
PPI	0.534	0.626	0.567	0.669	0.618	**0.677**	Acc
	0.519	0.620	0.694	0.698	0.592	**0.704**	ROC
	0.510	0.620	**0.768**	0.727	0.581	0.730	AP
	0.546	0.383	**0.633**	0.627	0.562	0.624	F1
Blog -Catalog	0.598	**0.734**	0.601	0.698	0.662	0.699	Acc
	0.539	0.798	0.703	0.733	0.830	**0.890**	ROC
	0.477	0.839	0.631	0.672	0.880	**0.904**	AP
	0.660	0.750	0.707	0.752	0.719	**0.753**	F1

Evaluation on Node Classification

Table 12: Node classification evaluation for Accuracy and F1-micro

Dataset \ Model	GrF	HOPE	N2v	D2v	GAE	Ours	Metric
Cora	0.087	0.577	0.714	0.681	0.625	**0.760**	Acc
	0.154	0.689	0.804	0.779	0.734	**0.831**	F1
Blog	0.060	0.060	0.065	0.065	0.063	**0.085**	Acc
-Catalog	**0.021**	0.010	0.020	0.020	0.020	0.017	F1

Evaluation on Clustering and Visaulizaiton



(a) JONNEE with d=100 (b) diff2vec with d=100

Figure 4: Comparison of visualization for diffusion and JONNEE (400 epochs)

Conclusion

In our research we

- ✓ present **new graph embedding model** for join node and edge optimization problem;
- ✓ we verify the quality of our approach on **link prediction** problem and conduct experiments on node classification, clustering and visualization;
- ✓ we will evaluate our model in semi-supervised framework: joint constraint prove to improve basic GCN performance;
- ✓ we aim to further study methods of **joint network node and edge embedding** methods and compare with existing models, such as ELAINE, LANE, Dual-Primal GCN and Edge-to-vec.

THANK YOU FOR YOUR ATTENTION!

iamakarov@hse.ru