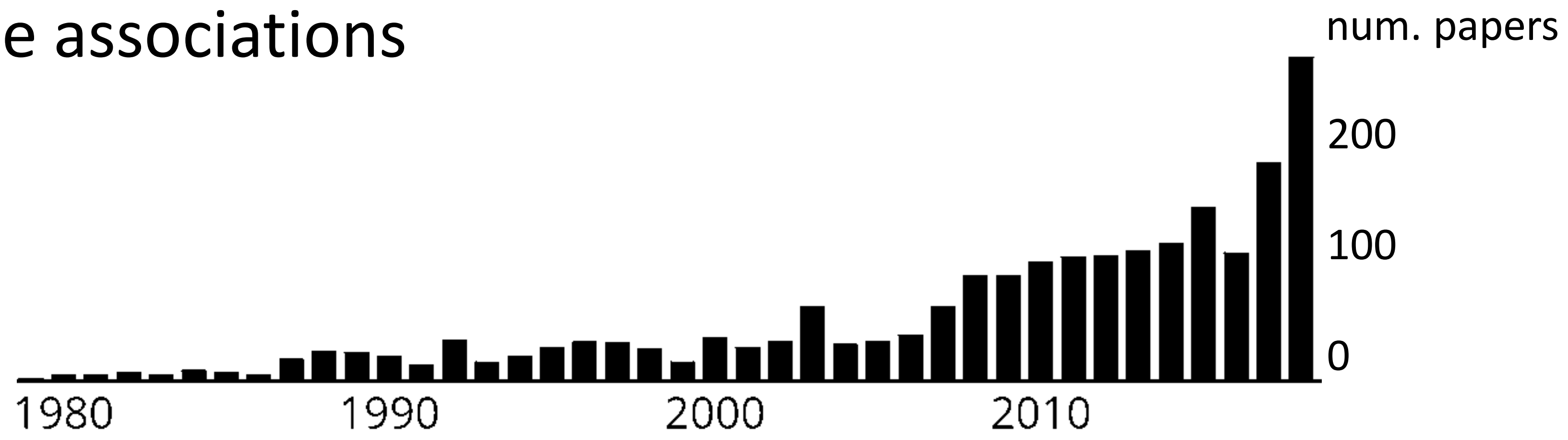
Adversarial Attacks on Node Embeddings via Graph Poisoning

Aleksandar Bojchevski, Stephan Günnemann
Technical University of Munich

ICML 2019

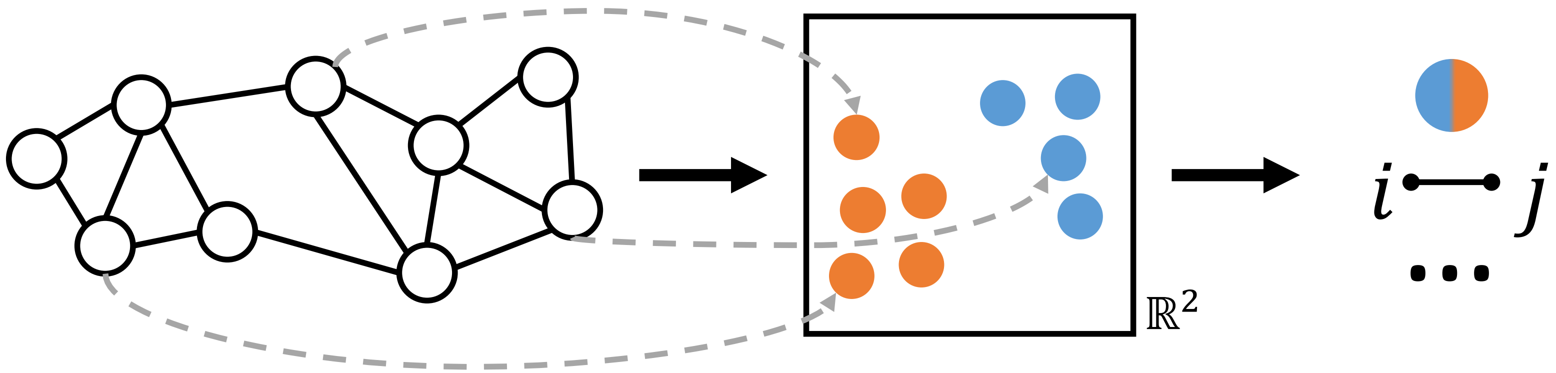
Node embeddings are used to

- Classify scientific papers
- Recommend items
- Classify proteins
- Detect fraud
- Predict disease-gene associations
- Spam filtering
- .....



Background: Node embeddings

Every node $v \in \mathcal{V}$ is mapped to a low-dimensional vector $z_v \in \mathbb{R}^d$ such that the graph structure is captured.



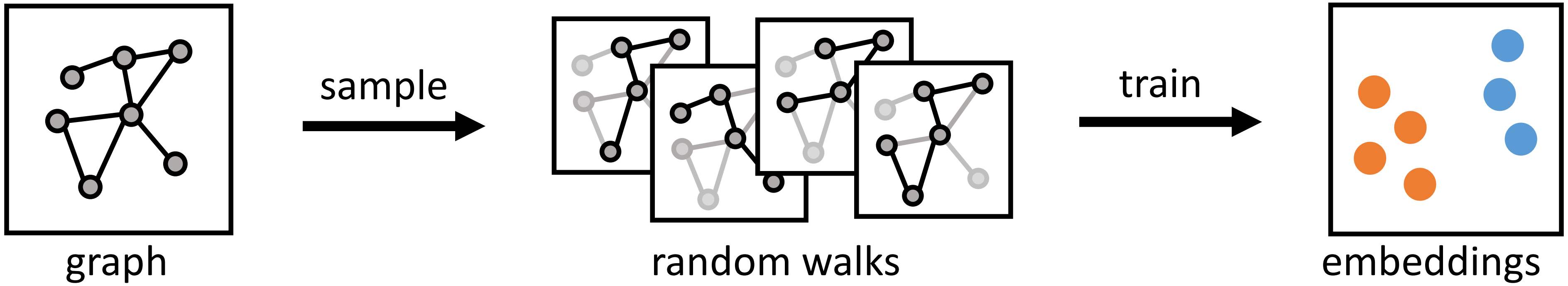
node classification
link prediction
other tasks

Similar nodes are close to each other in the embedding space.

Background: Random walk based embeddings

Let nodes = words and random walks = sentences.

Train a language model, e.g. Word2Vec.

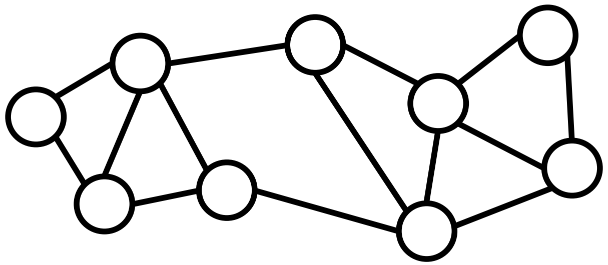


Nodes that co-occur in the random-walks have similar embeddings.

Are node embeddings robust to adversarial attacks?

In domains where graph embeddings are used (e.g. the Web) adversaries are common and false data is easy to inject.

Adversarial attacks in the graph domain

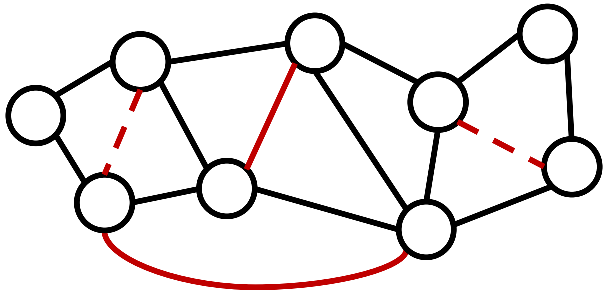


clean graph

\+

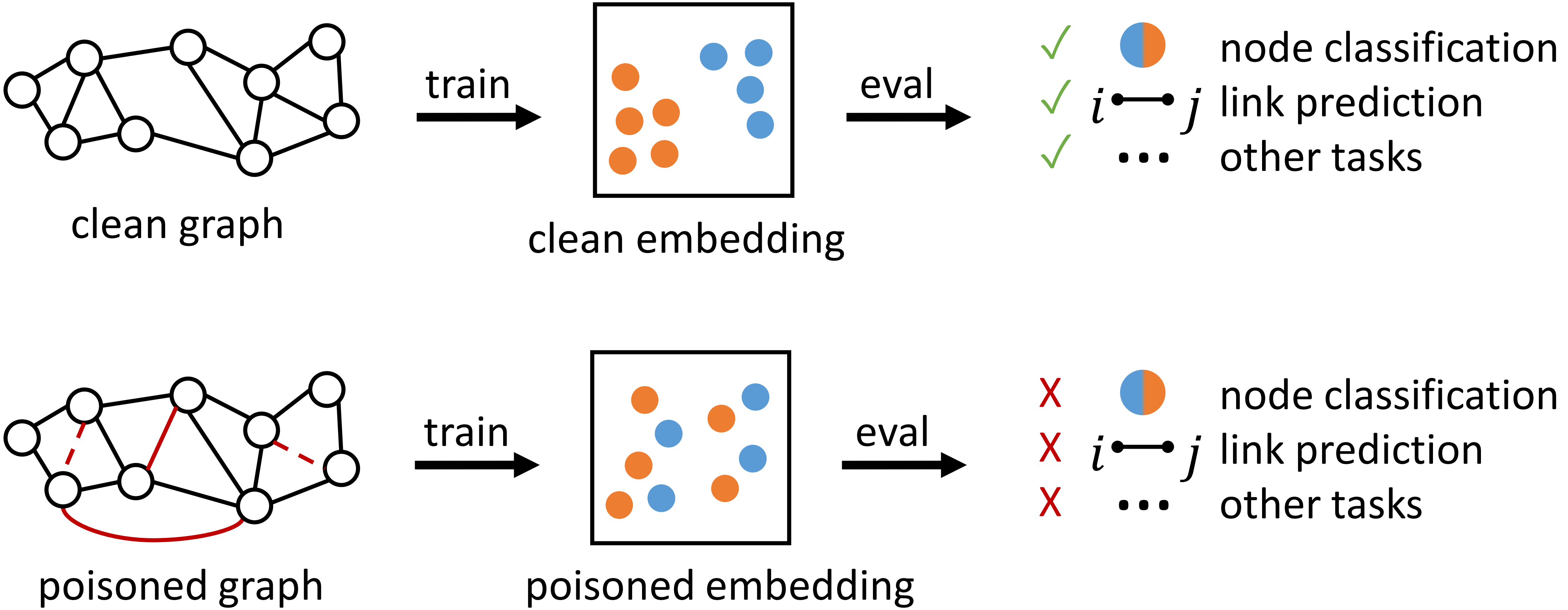
adversarial flips:
add ( ) and/or
remove ( ) edges

=



poisoned graph

Poisoning: train after the attack



Poisoning attack formally

The graph after perturbing some edges

$$G_{pois.} = \underset{\substack{G \in all\ graphs \\ |G_{clean} - G| \leq budget}}{\operatorname{argmax}} \mathcal{L}(G, Z^*(G))$$

$$Z^*(G) = \underset{Z}{\operatorname{argmin}} \mathcal{L}(G, Z)$$

The optimal embedding from the to be optimized graph G

Poisoning attack for random walk models

The graph after perturbing some edges

$$G_{pois.} = \underset{\substack{G \in all\ graphs \\ |G_{clean} - G| \leq budget}}{\operatorname{argmax}} \mathcal{L}(G, Z^*(G))$$

$$Z^*(G) = \underset{Z}{\operatorname{argmin}} \mathcal{L}(\{r_1, r_2, \ldots\}_G, Z) \quad r_i = rnd\_walk(G)$$

The optimal embedding from the to be optimized graph G

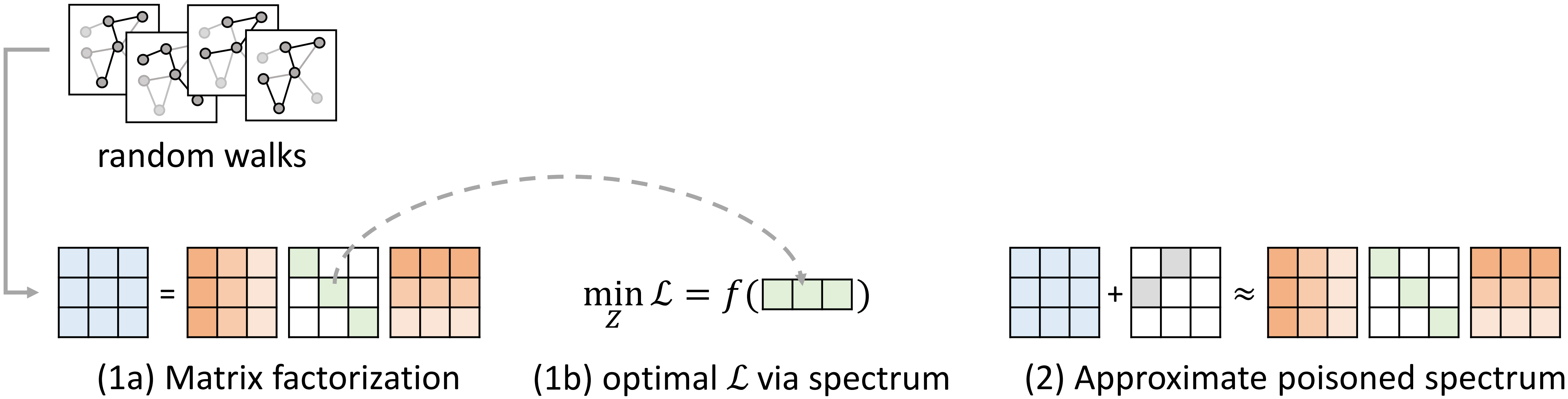
$$\mathrm{G}_{pois.} = \underset{\substack{G \in all\ graphs \\ |G_{clean} - G| \leq budget}}{\operatorname{argmax}} \; \min_{Z} \mathcal{L}(\{r_1, r_2, \dots\}_G, Z)$$

Challenges

Bi-level optimization problem.

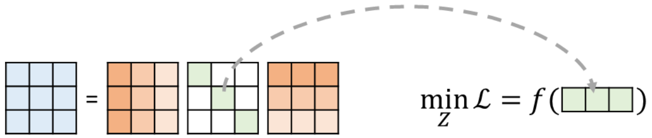
Combinatorial search space.

Inner optimization includes non-differentiable sampling.



Overview

1. Reduce the bi-level problem to a single-level
   a) DeepWalk as Matrix Factorization
   b) Express the optimal $\mathcal{L}$ via the graph spectrum
2. Approximate the poisoned graph's spectrum



1. Reduce bi-level problem to a single-level

a) DeepWalk corresponds to factorizing the PPMI matrix.

transition/degree matrix

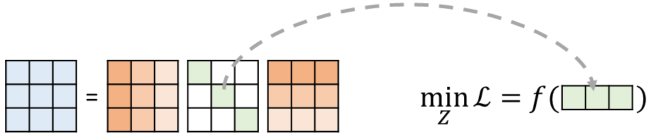
$$M_{ij} = \log \max\{cS_{ij}, 1\} \qquad S = \left(\sum_{r=1}^{T} P^r\right) D^{-1}$$

Get the embeddings Z via SVD of M

Rewrite S in terms of the generalized spectrum of A.

$$Au = \lambda D u \qquad S = U \left(\sum_{r=1}^{T} \Lambda^r\right) U^T$$

generalized eigenvalues/vectors



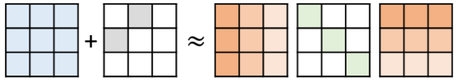
1. Reduce bi-level problem to a single-level

b) The optimal loss is now a simple function of the eigenvalues.

$$\min_{Z} \mathcal{L}(G, Z) = f(\lambda_i, \lambda_{i+1}, \dots)$$

Training the embedding is replaced by computing eigenvalues.

$$G_{pois.} = \operatorname*{argmax}_{G} \min_{Z} \mathcal{L}(G, Z) \quad \Rightarrow \quad G_{pois.} = \operatorname*{argmax}_{G} f(\lambda_i, \lambda_{i+1}, \dots)$$



2. Approximate the poisoned graph's spectrum

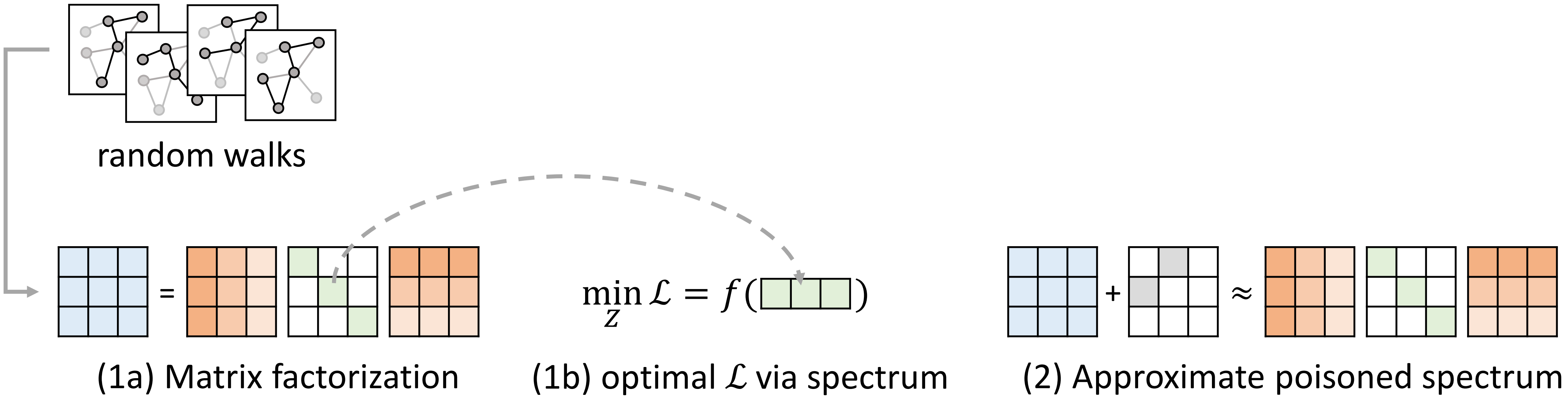
Compute the change using Eigenvalue Perturbation Theory.

$$A_{poisoned} = A_{clean} + \Delta A$$

$$\lambda_{poisoned} = \lambda_{clean} + u_{clean}^T(\Delta A + \lambda_{clean}\Delta D)u_{clean}$$

simplifies for a single edge flip (i, j)

$$\lambda_p = \lambda_c + \Delta \mathrm{A}_{ij}\left(2u_{ci} \cdot u_{cj} - \lambda_c(u_{ci}^2 + u_{cj}^2)\right)$$ # compute in $O(1)$

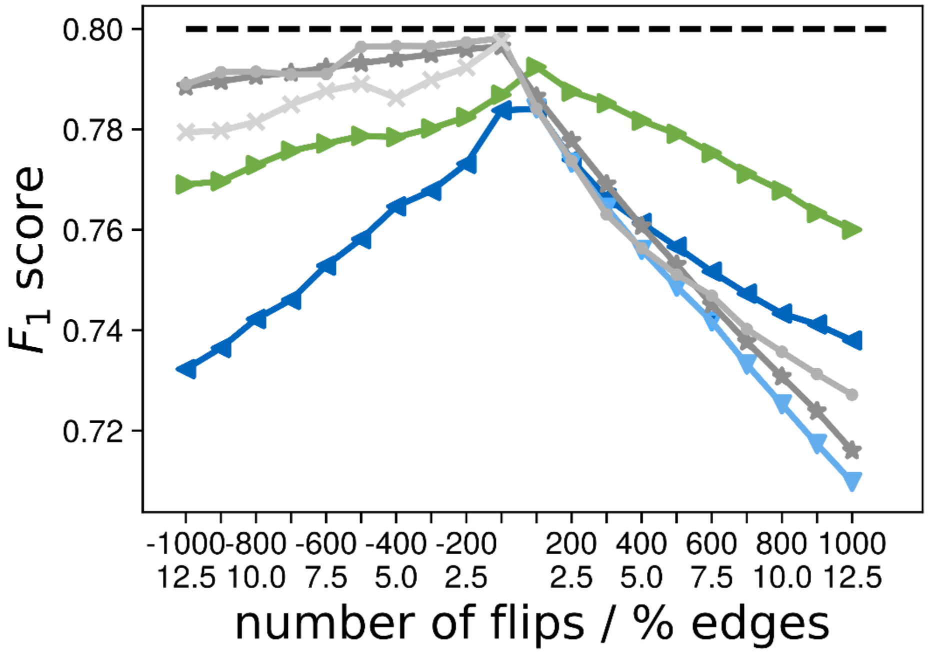


Overall algorithm

1. Compute generalized eigenvalues/vectors (Λ/U) of the graph
2. For all candidate edge flips (i,j) compute the change in λ_i
3. Greedily pick the top candidates leading to largest optimal loss

General attack

Poisoning decreases the overall quality of the embeddings.



Our attacks:

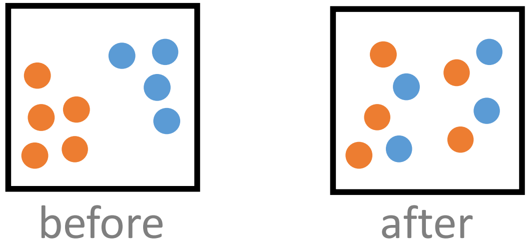
Gradient baseline:

Simple baselines:

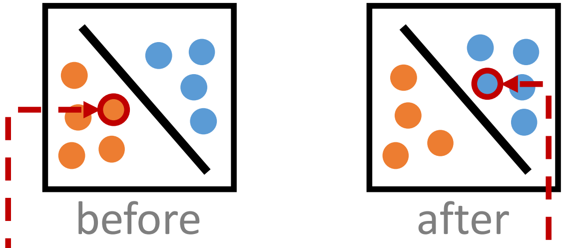
Clean graph:

Targeted attack

Goal: attack a specific node and/or a specific downstream task.



General Attack



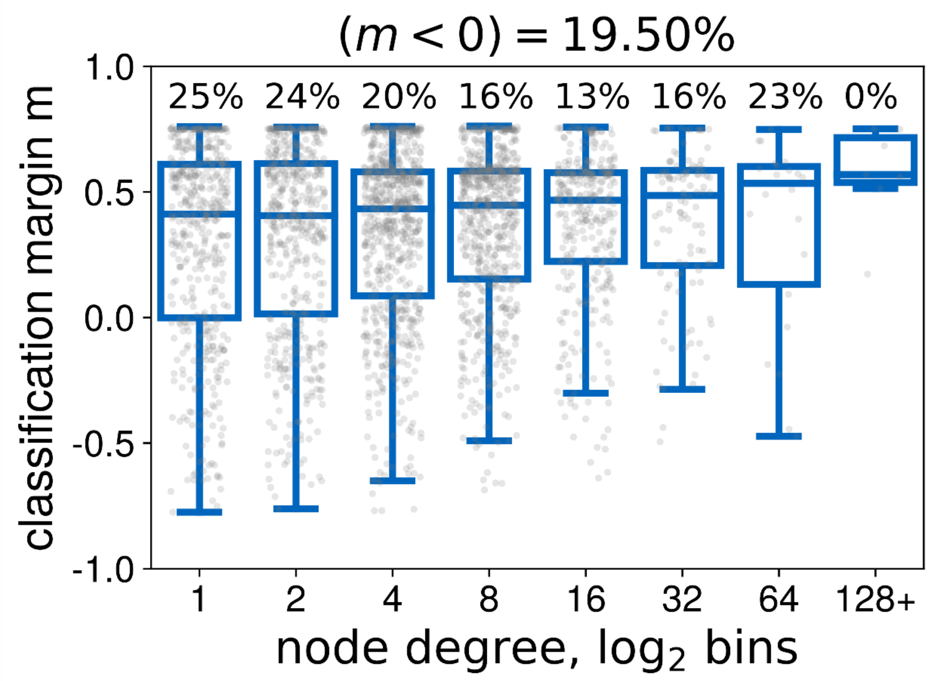
Targeted Attack

Examples:

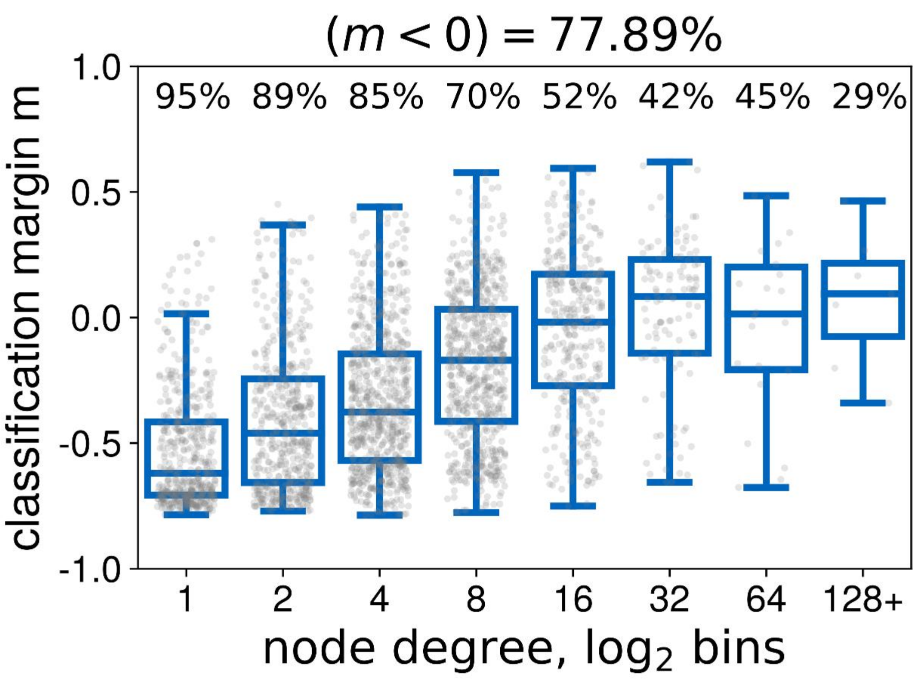
- Misclassify a single given target node t
- Increase/decrease the similarity of a set of node pairs $\mathcal{T} \subset \mathcal{V} \times \mathcal{V}$

Targeted attack

Most nodes can be misclassified with few adversarial edges.



Before attack



After attack

Transferability

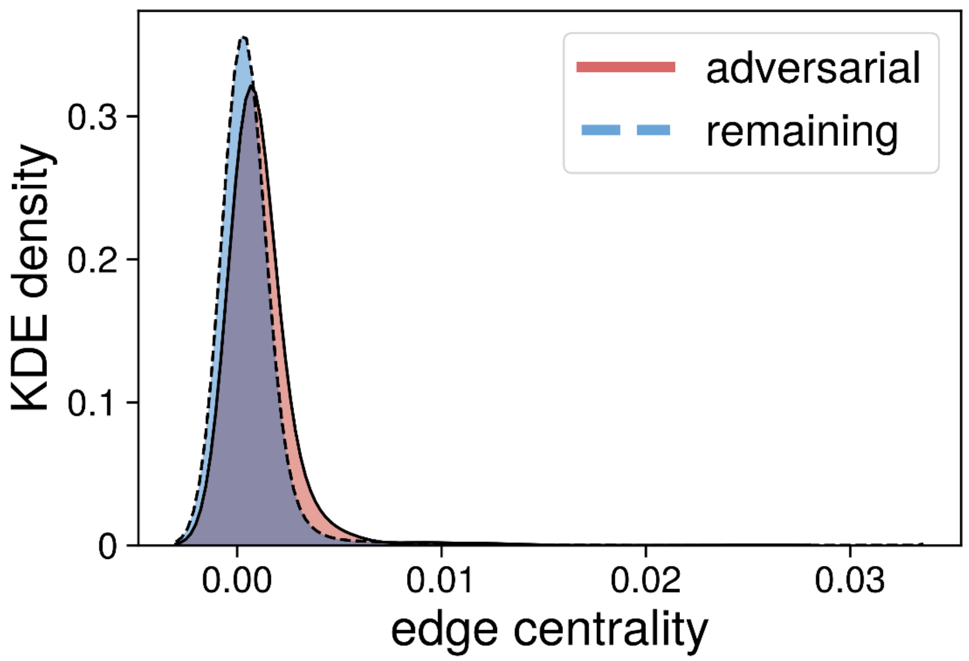
Our selected adversarial edges transfer to other (un)supervised methods.

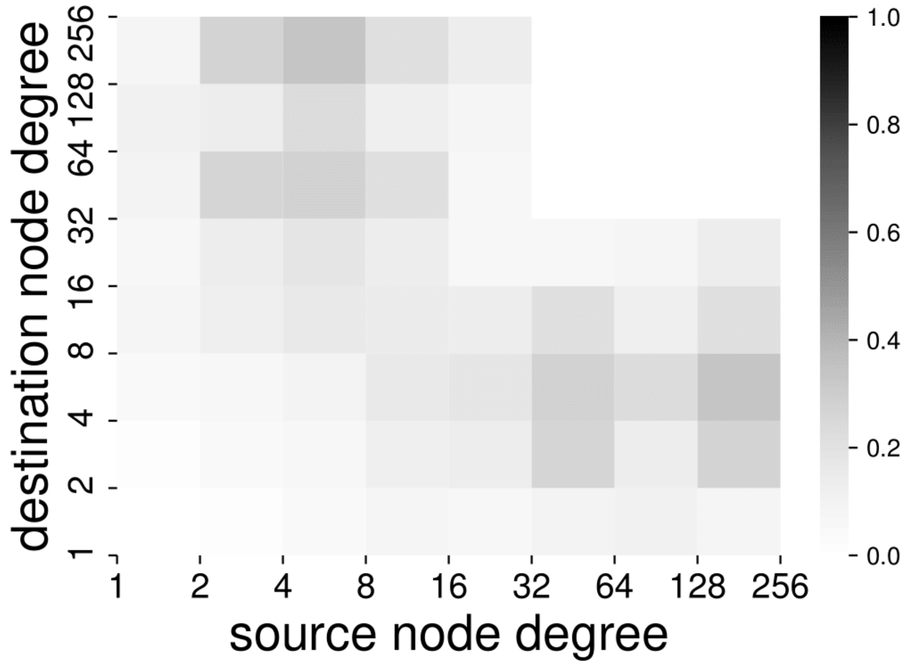
budget	DeepWalk SVD	DeepWalk Sampling	node 2vec	Spectral Embed.	Label Prop.	Graph Conv.
250	-7.59	-5.73	-6.45	-3.58	-4.99	-2.21
500	-9.68	-11.47	-10.24	-4.57	-6.27	-8.61

The change in F_1 score (in percentage points) compared to the clean graph. Lower is better.

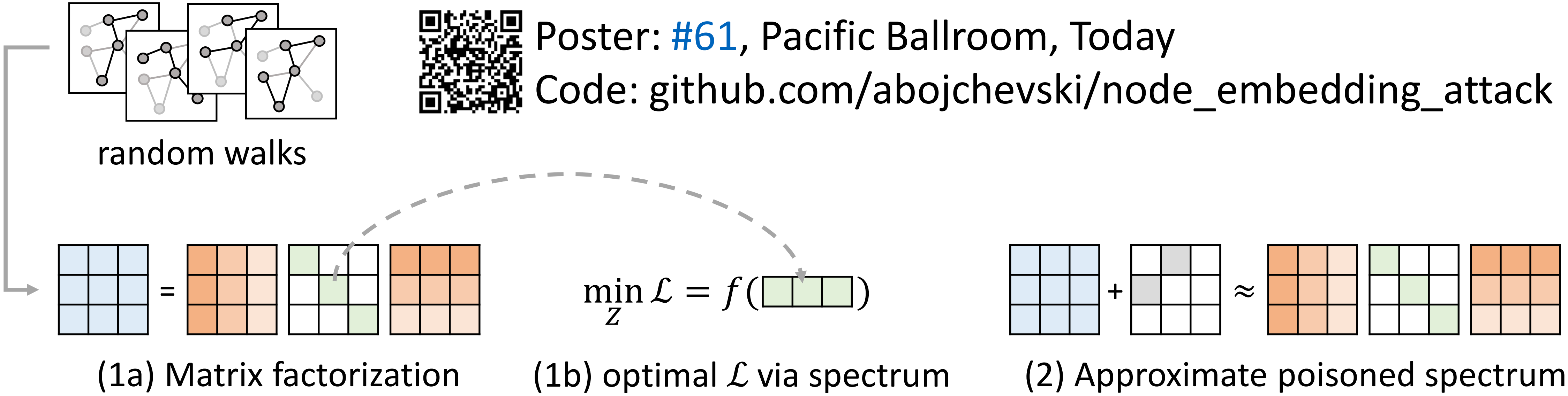
Analysis of adversarial edges

There is no simple heuristic that can find the adversarial edges.





Poster: #61, Pacific Ballroom, Today
Code: github.com/abojchevski/node_embedding_attack



Summary

- ❑ Node embeddings are vulnerable to adversarial attacks.
- ❑ Find adversarial edges via matrix factorization and the graph spectrum.
- ❑ Relatively few perturbations degrade the embedding quality and the performance on downstream tasks.