Machine Learning for Graphs and Sequential Data

Graphs – Limitations of GNNs

Lecturer: Prof. Dr. Stephan Günnemann

www.daml.in.tum.de

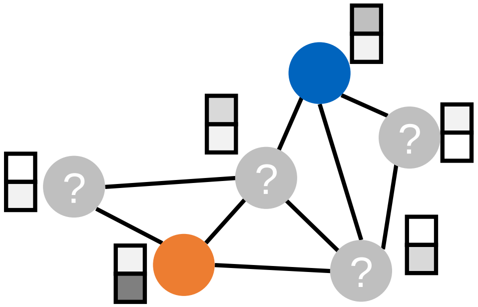
Summer Term 2020

Data Analytics and
Machine Learning

Roadmap

- **Chapter: Graphs**
  1. Graphs & Networks
  2. Generative Models
  3. Clustering
  4. Node Embeddings
  5. Ranking
  6. Semi-Supervised Learning
  7. **Limitations of GNNs**
     - Overview
     - **Robustness**

Adversarial Attacks on GNNs

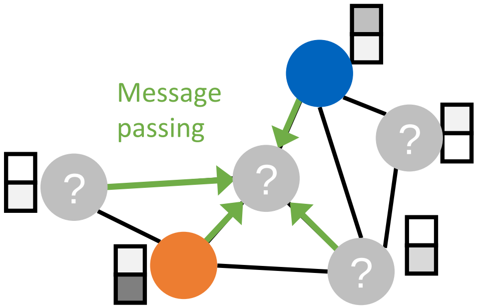


- Earlier in this course we have seen the problem of (adversarial) robustness of classifiers on “traditional” data, e.g. images.

- In contrast, graph neural networks (GNNs) use **both** the **node’s attributes** as well as their **connections** to make a prediction.
  - Therefore, adversarial attacks can happen through both the **node attributes** as well as the **graph structure**.
  - Structural attacks are indeed quite common in the real world (e.g. adding fake connections in a social network)

- Structure attacks are specifically challenging since they change the flow of messages passed through the GNN

Adversarial Attacks on GNNs



- Example: two-layer GCN in matrix form:

node attributes

$$\boldsymbol{Z} \in \mathbb{R}^{N\times C} = f_\theta(\boldsymbol{A},\boldsymbol{X}) = \mathrm{softmax}\Big(\hat{\boldsymbol{A}}\ \mathrm{ReLU}\big(\hat{\boldsymbol{A}}\boldsymbol{X}\boldsymbol{W}^{(1)} + \boldsymbol{b}^{(1)}\big)\boldsymbol{W}^{(2)} + \boldsymbol{b}^{(2)}\Big)$$

message passing

- $\theta = \left\{\boldsymbol{W}^{(1)}, \boldsymbol{b}^{(1)}, \boldsymbol{W}^{(2)}, \boldsymbol{b}^{(2)}\right\}$ are learnable model weights.
- **Adversarial attack**: Modify node attributes $\boldsymbol{X}$ and/or adjacency matrix $\boldsymbol{A}$ in order to maximize classification loss
  - of an individual target node or
  - on the whole dataset/test set (global attack).

GNN Adversarial Attacks: Challenges

1. Optimization over **discrete variables** (the graph structure). Perturbations are measured via non-convex L_0 norm.

2. **Relational dependencies** between the nodes: cannot view samples in isolation.

3. $(\boldsymbol{A}', \boldsymbol{X}') \approx (\boldsymbol{A}, \boldsymbol{X})$: What is a sensible measure of perturbations that do not change the semantics for (attributed) graphs?

4. **Transductive setting**: unlabeled data is **used during training**; most realistic scenario is a **poisoning attack**, where the attacker modifies the training data, which corresponds to a challenging **bilevel optimization problem:**

$$\max_{\boldsymbol{A},\boldsymbol{X}} \mathcal{L}_{test}\big(f_{\theta^*}(\boldsymbol{A},\boldsymbol{X})\big) \quad s.t.\ \theta^* = \arg\min_{\theta} \mathcal{L}_{train}(f_{\theta}\,(\boldsymbol{A},\boldsymbol{X}))$$

GNN Adversarial Attack: Nettack

- One of the earliest GNN attack algorithms [Zügner+ 2018].
- Targets a **single node's prediction**.

$$\boldsymbol{Z} = f_\theta(\boldsymbol{A}, \boldsymbol{X}) = \cancel{softmax}\left(\widehat{\boldsymbol{A}}\ \cancel{ReLU}\left(\widehat{\boldsymbol{A}}\boldsymbol{X}\boldsymbol{W}^{(1)}\right)\boldsymbol{W}^{(2)}\right)$$

Linearize classifier

$$\log \boldsymbol{Z}' \quad = \widehat{\boldsymbol{A}}^2 \boldsymbol{X}\boldsymbol{W}'$$

Structure perturbations: $\max_{\widehat{\boldsymbol{A}}} \mathcal{L}'\left(\log \boldsymbol{Z}'_v\right)\ where\ \log Z'_v = [\widehat{\boldsymbol{A}}^{\mathbf{2}}\boldsymbol{C}]_v$

Feature perturbations: $\max_{\boldsymbol{X}} \mathcal{L}'(\log \boldsymbol{Z}'_v)\ where\ \log Z'_v = [\boldsymbol{C}_1\boldsymbol{X}\boldsymbol{C}_2]_v$

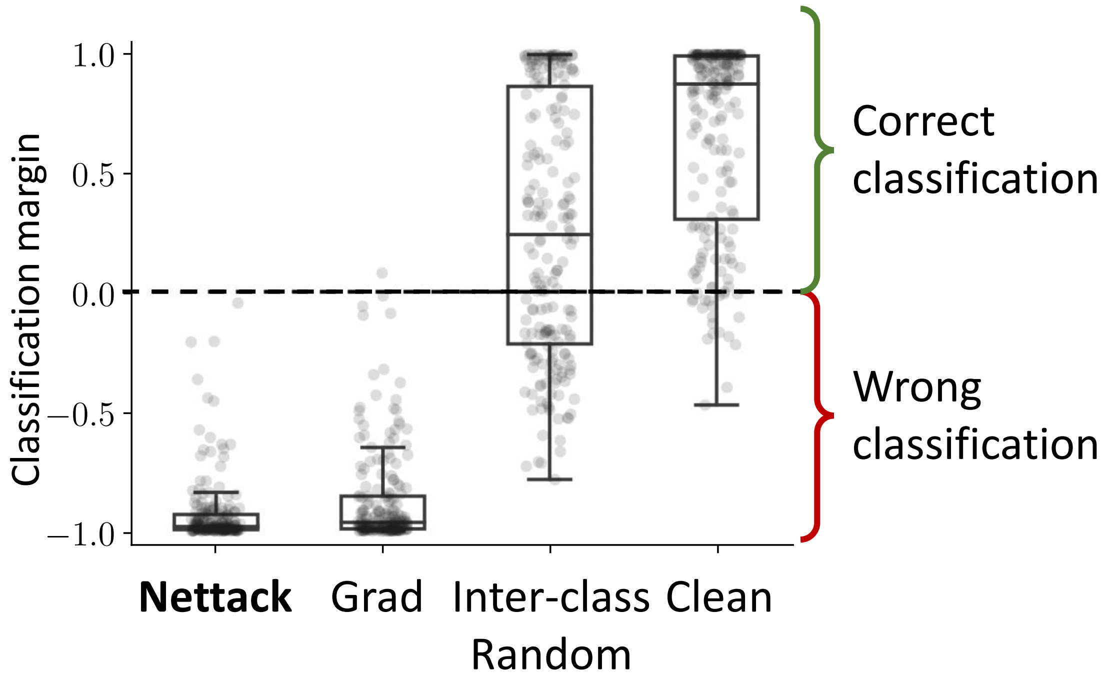
Constants

➔ **Greedily** pick the **optimal perturbation** at each step.

➔ Uses closed-form solutions for the **optimal perturbation** at each step

GNN Adversarial Attack: Nettack results

- Poisoning attack scenario (model is trained on perturbed data)
- Each point represents one attacked node
- Attack budget per node: $\Delta(i) = \deg(i) + 2$



% Correct: **1.0%** 2.7% 60.8% 90.3%

Improving Robustness

- GNNs are not robust under adversarial perturbations
  - specifically graph structure perturbations are harmful
- Heuristic defenses:
  - E.g. adjacency low-rank approximation via truncated Singular Value Decomposition (Entezari et al., 2020); filtering of malicious edges via attribute similarity (Wu et al., 2019)
  - However: equivalent/similar defenses for CNNs have been proven to be non-robust against worst-case perturbations
- Robust Training:
  - In form of Adversarial Training, e.g., via Projected Gradient Descent (Xu et al., 2019)
  - Or proposed together with a certification technique (upcoming topic)

Recall: Certification (via Convex Relaxation)

Rephrase the original **goal**: develop an algorithm that answers the question:

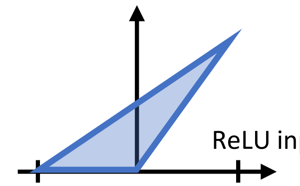
"Is the GNN f_θ around the features $\mathbf{X}$ and adjacency matrix $\mathbf{A}$ adversarial-free (within an ϵ-ball(s) measured by some norm)?"

Allowed answers in the relaxed setting:

- **`YES`**: If for all $\tilde{\mathbf{x}} \in \mathcal{P}_X(\mathbf{x}), \widetilde{\mathbf{A}} \in \mathcal{P}_A(\mathbf{A})$: $\arg\max F\left(\tilde{\mathbf{x}}, \widetilde{\mathbf{A}}\right) = \arg\max F(\mathbf{x}, \mathbf{A})$
- **`POTENTIALLY NOT / MAYBE`**: In this case we have no guarantees.
- [**`NO`**: If any $\tilde{\mathbf{x}} \in \mathcal{P}_X(\mathbf{x}), \widetilde{\mathbf{A}} \in \mathcal{P}_A(\mathbf{A})$: $\arg\max F\left(\tilde{\mathbf{x}}, \widetilde{\mathbf{A}}\right) \neq \arg\max F(\mathbf{x}, \mathbf{A})$]

1. Graph and Attributes may change simultaneously
2. The nodes of a graph are non i.i.d.
3. L_0-ball perturbations is natural for discrete data

Exact / Relaxed Certification



Already challenging if we are only allowed to perturb $\mathbf{X}$

Proposed approaches so far are focusing on specific architectures and/or only attribute or structure perturbations:

- One can generalize the relaxed certification setting via linear programs to **attribute perturbations** on a **GCN** (Zügner and Günnemann, 2019).
- Certifying a **GCN** against **structure perturbations** can be formulized via a Jointly Constraint Bilinear Program (Zügner and Günnemann, 2020).
- To certify a **PPNP** model w.r.t. **structure perturbations**, we may solve a Quadratically Constrained Linear Program (Bojchevski and Günnemann, 2019).
  - under specific perturbation models ("local budget"; max x perturbations per node) one can perform certification exactly in polynomial time; for a global budget (max x perturbations overall), the problem becomes NP-hard and, thus, requires relaxation for efficiency

Randomized Smoothing

Recall: Smooth classifier $g(\mathbf{x})_c$ returns the probability that the base classifier f classifies a smoothed sample $\tilde{\mathbf{x}} \sim \phi(\mathbf{x})$ as class c

$$g(\mathbf{x})_c := \mathbb{P}\big(\mathrm{f}\big(\phi(\mathbf{x})\big) = c\big) = \mathbb{E}_{\tilde{\mathbf{x}} \sim \phi(\mathbf{x})}(\mathbb{I}[\mathrm{f}(\tilde{\mathbf{x}}) = c])$$

with a randomization scheme $\phi(\mathbf{x})$. We denote with $c^* = \arg\max_c g(\mathbf{x})_c$ the most likely class.

Goal: We want to certify the smooth classifier g. That is we aim to show that for a radius r it holds:

$$\text{for all } \mathbf{x}' \text{ with } \|\mathbf{x}' - \mathbf{x}\|_0 \leq r: \quad c^* = \arg\max_c g(\mathbf{x}')_c$$

For simplicity, we assume binary data (e.g. an unweighted graph)

→ **$\mathbf{L_0}$-norm radius certification.**

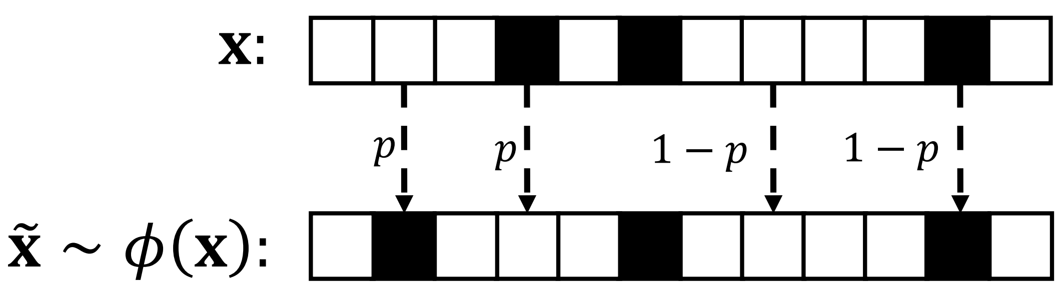
How to Smooth Graph Data?

Challenge: Adding **Gaussian noise** to the **discrete graph structure** is not suitable

Solution: We model the n^2 possible edges in the adjacency matrix as a **Bernoulli random variable**

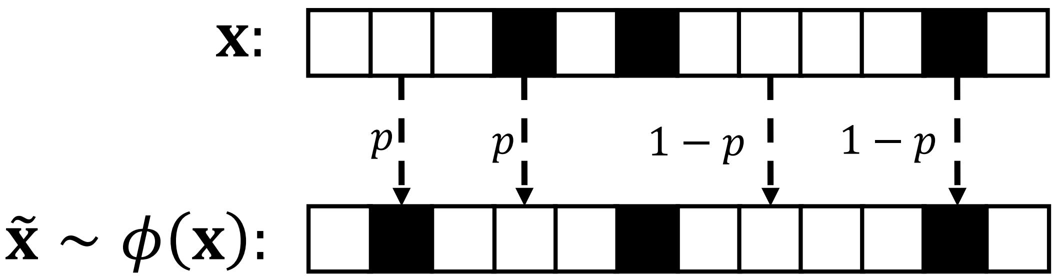
First idea: Same "flip probability" for every element

$$\tilde{\mathbf{x}} \sim \phi(\mathbf{x}) \text{ defined via } \mathbb{P}(\tilde{\mathbf{x}}_i \mid \mathbf{x}) = \begin{cases} p & , \tilde{\mathbf{x}}_i = 1 - \mathbf{x}_i \\ 1 - p & , \tilde{\mathbf{x}}_i = \mathbf{x}_i \end{cases}$$



How to Smooth Graph Data?

First idea: Same ”flip probability” for every element



Problem: Real-world graphs are typically very **sparse** ($m \ll n^2$) and hence picking a meaningful p almost impossible

- Large flip probability p: most certainly we will add more random edges than original edges exist
- Small flip probability p: usually only a very few edges would be deleted

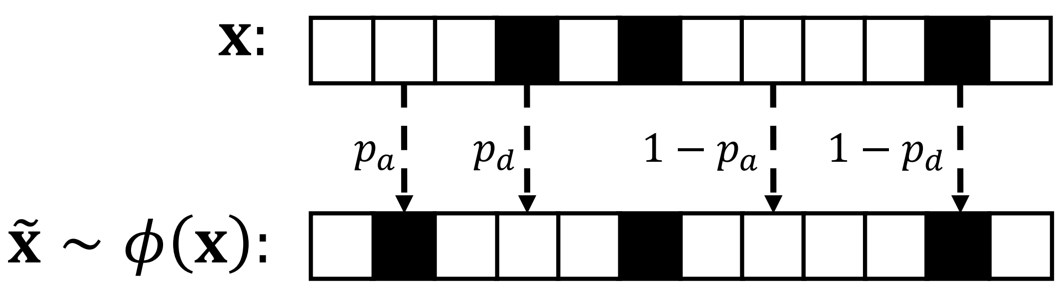
How to Smooth Graph Data? Sparsity Matters!

Sparsity-aware random sampling $\tilde{\mathbf{x}} \sim \phi(\mathbf{x})$:

Every element of the adjacency matrix is modelled via a **Bernoulli random variable** with **data dependent probability**:

$$\mathbb{P}(\tilde{\mathbf{x}}_i \mid \mathbf{x}) = \begin{cases} p_d^{\mathbf{x}_i} p_a^{1-\mathbf{x}_i} & , \tilde{\mathbf{x}}_i = 1 - \mathbf{x}_i \\ (1-p_d)^{\mathbf{x}_i}(1-p_a)^{1-\mathbf{x}_i} & , \tilde{\mathbf{x}}_i = \mathbf{x}_i \end{cases}$$

That is, each of the n^2 elements in the adjacency matrix is flipped with probability p_a if the value was previously 0 (no edge) or with p_d if previously an edge existed:

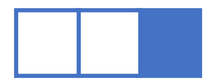


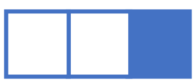
Smoothed Classifier for Discrete Data

With this randomization scheme we can write:

$$g(\mathbf{x})_c \coloneqq \mathbb{P}\big(\mathrm{f}(\phi(\mathbf{x})) = c\big) = \mathbb{E}_{\tilde{\mathbf{x}}\sim\phi(\mathbf{x})}(\mathbb{I}[\mathrm{f}(\tilde{\mathbf{x}}) = c])$$

$$= \sum_{\tilde{\mathbf{x}}} \mathbb{P}(\tilde{\mathbf{x}}\,|\mathbf{x})\mathbb{I}[\mathrm{f}(\tilde{\mathbf{x}}) = c] = \sum_{\tilde{\mathbf{x}}\text{ s.t. }\mathrm{f}(\tilde{\mathbf{x}})=c} \mathbb{P}(\tilde{\mathbf{x}}\,|\mathbf{x}) = \sum_{\tilde{\mathbf{x}}\text{ s.t. }\mathrm{f}(\tilde{\mathbf{x}})=c} \prod_{i=1}^{n^2} \mathbb{P}(\tilde{\mathbf{x}}_i\,|\mathbf{x})$$

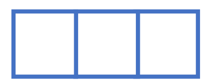
We illustrate $\mathbb{P}(\tilde{\mathbf{x}}\,|\mathbf{x})$ with a hypothetical subgraph: $\mathbf{x} =$ 

$\mathbb{P}($  $) = (1-p_a)^2(1-p_d)$

$\mathbb{P}($  $) = p_a(1-p_a)(1-p_d)$

$\mathbb{P}($  $) = p_a(1-p_a)(1-p_d)$

$\mathbb{P}($  $) = p_a^2(1-p_d)$

$\mathbb{P}($  $) = (1-p_a)^2 p_d$

$\mathbb{P}($  $) = p_a(1-p_a)p_d$

$\mathbb{P}($  $) = p_a(1-p_a)p_d$

$\mathbb{P}($  $) = p_a^2 p_d$

Worst-Case Base Classifier

Let's assume we know the value of $g(\mathbf{x})_{c^*}$ for the original sample $\mathbf{x}$

- Note: Of course, like in the Gaussian/continuous case, we do **not** compute this term exactly (far too expensive) but rather derive a bound based on MC samples

How far can we deviate from $\mathbf{x}$, e.g. obtaining $\mathbf{x}'$, and still **guarantee** that we do not change the prediction, i.e. still have $\arg\max_c g(\mathbf{x})_c = c^* = \arg\max_c g(\mathbf{x}')_c$?

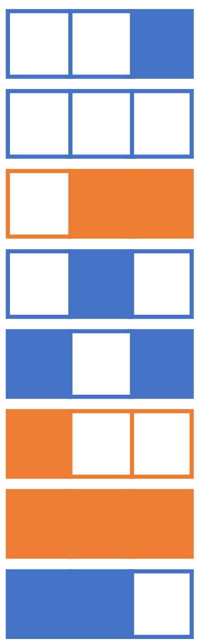
$\longrightarrow$ Similarly to the Gaussian/continuous randomized smoothing, to answer this question, we can inspect the **worst-case base classifier.**

Since the worst-case base classifier has a simple form (e.g. linear in the Gaussian case), once we know it, it is "rather simple" to obtain the certification radius

The Space of Base Classifiers

- Recall: we only assumed knowledge about the value of $g(\mathbf{x})_{c^*}$
  - We do **not** know the output of the actual base classifier f at every possible input
- Various base classifiers h fulfill the property $\sum_{\tilde{\mathbf{x}} \text{ s.t. } h(\tilde{\mathbf{x}})=c^*} \mathbb{P}(\tilde{\mathbf{x}} \,|\mathbf{x}) = g(\mathbf{x})_{c^*}$
  - Let $\mathcal{H}$ denote the set of all these base classifiers; clearly $f \in \mathcal{H}$

$h_1 \in \mathcal{H}$



$\mathbb{P}(\ )$ $= (1-p_a)^2(1-p_d) =$ $\mathbb{P}(\ )$

$\mathbb{P}(\ )$ $= (1-p_a)^2 p_d =$ $\mathbb{P}(\ )$

$\mathbb{P}(\ )$ $= p_a(1-p_a)(1-p_d) =$ $\mathbb{P}(\ )$

$\mathbb{P}(\ )$ $= p_a(1-p_a)p_d =$ $\mathbb{P}(\ )$

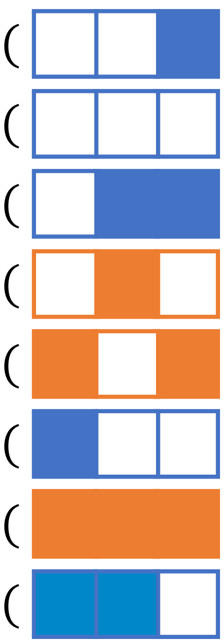
$\mathbb{P}(\ )$ $= p_a(1-p_a)(1-p_d) =$ $\mathbb{P}(\ )$

$\mathbb{P}(\ )$ $= p_a(1-p_a)p_d =$ $\mathbb{P}(\ )$

$\mathbb{P}(\ )$ $= p_a^2(1-p_d) =$ $\mathbb{P}(\ )$

$\mathbb{P}(\ )$ $= p_a^2 p_d =$ $\mathbb{P}(\ )$

$h_2 \in \mathcal{H}$



How to Find the Worst-Case Base Classifier?

For any **chosen location** $\mathbf{x}'$, we can express the **worst-case base classifier** as a minimization problem:

$$\min_{h \in \mathcal{H}} \sum_{\tilde{\mathbf{x}} \text{ s.t. } \mathrm{h}(\tilde{\mathbf{x}})=c^*} \mathbb{P}(\tilde{\mathbf{x}} \,|\mathbf{x}') \qquad \left(\leq \sum_{\tilde{\mathbf{x}} \text{ s.t. } \mathrm{f}(\tilde{\mathbf{x}})=c} \mathbb{P}(\tilde{\mathbf{x}} \,|\mathbf{x}') = g(\mathbf{x}')_{c^*} \right),$$

Intuition:

We search for a base classifier h such that the resulting smooth classifier

- maintains the probability mass for the clean sample $\mathbf{x}$,
  i.e. $\mathbb{P}\big(h\big(\phi(\mathbf{x})\big) = c^*\big) = g(\mathbf{x})_{c^*}$ // $h \in \mathcal{H}$
- and simultaneously minimizes the probability mass at the perturbed sample $\mathbf{x}'$, i.e. $\mathbb{P}\big(h\big(\phi(\mathbf{x}')\big) = c^*\big) = \sum_{\tilde{\mathbf{x}} \text{ s.t. } \mathrm{h}(\tilde{\mathbf{x}})=c^*} \mathbb{P}(\tilde{\mathbf{x}} \,|\mathbf{x}')$

Solution for the Worst-Case Base Classifier

The previous minimization problem can be formulated as a linear program!

Denote with $\tilde{\mathbf{x}}^{(i)}$ the enumeration of all possible $\tilde{\mathbf{x}}$.

$$\min_{\mathbf{h}} \sum_i \mathbf{h}_i \, \mathbb{P}\left(\tilde{\mathbf{x}}^{(i)} | \mathbf{x}'\right)$$

$$\text{subject to} \sum_i \mathbf{h}_i \, \mathbb{P}\left(\tilde{\mathbf{x}}^{(i)} | \mathbf{x}\right) = g(\mathbf{x})_{c^*} \;\text{ and }\; 0 \leq \mathbf{h}_i \leq 1$$

The vector $\mathbf{h}$ represents the worst-case base classifier: $\mathbf{h}_i$ indicates whether $h(\tilde{\mathbf{x}}^{(i)})$ outputs c^* ($\mathbf{h}_i = 1$) or some other class

- Technically it is a soft classifier (like logistic regression) since $0 \leq \mathrm{h}_i \leq 1$

Solution for the Worst-Case Base Classifier

Interesting fact: This is a very special LP, which can efficiently and exactly be solved with a greedy approach

- Initialize all $\mathbf{h}_i$ with zero
- Compute likelihood ratios $\eta_i = {}^{\mathbb{P}(\tilde{\mathbf{x}}^{(i)}|\mathbf{x})}/_{\mathbb{P}(\tilde{\mathbf{x}}^{(i)}|\mathbf{x}')}$ and sort them
  - i.e. get indices $j_1, j_2, j_3, \ldots$ such that $\eta_{j_1} \geq \eta_{j_2} \geq \eta_{j_3} \geq \cdots$
- For $k = 1, \ldots$ set $\mathbf{h}_{j_k} = 1$ while budget $\sum_i \mathbf{h}_i \, \mathbb{P}\left(\tilde{\mathbf{x}}^{(\mathrm{i})}|\mathbf{x}\right) = g(\mathbf{x})_{c^*}$ is not used up
  - i.e. process the sorted indices from left to right and assign a 1 (again: at the "switch point" we might have $0 < \mathbf{h}_{j.} < 1$ to consume the budget fully).

➢ Result: We do not even have to solve an optimization problem! We just sort based on the **likelihood ratio** and assign class c^* to the "left part"

- Note the similarity to the linear classifier in the Gaussian/continuous case

➢ The worst-case base classifier has a very simple form

Some Details We Skip

- Knowing the worst-case base classifier, enables us to find the certification radius r (technically we even have two radii: addition/deletion)
  - Core insight: The general form of the worst-case classifier is always the same, independent of which $\mathbf{x}'$ we consider; similar to the Gaussian/continuous case
- In the linear program the dimensionality of $\mathbf{h}$ would be enormous (all possible graphs). We can use the fact that only the likelihood ratio $\eta_i = {}^{\mathbb{P}(\tilde{\mathbf{x}}^{(i)}|\mathbf{x})}/_{\mathbb{P}(\tilde{\mathbf{x}}^{(i)}|\mathbf{x}')}$ matters for the solution.
  - Intuition: Group together all graphs that have the same value for η_i into one large region → dimensionality of $\mathbf{h}$ equals to the number of regions
  - Indeed we have only a small number of regions: linear in the radius/dimensionality of the input; very fast certification possible
- For further details we refer to (Bojchevski et al., 2020).

Most importantly, this randomized smoothing technique works for all models with binary input data: GNNs, CNNs, SVMs, Decision Trees, …

Questions

1. Is a projected-gradient-descent (PGD) attack on a GNN via the graph structure a good idea? Why or why not?

2. Suppose you want to determine the worst-case structure perturbation Δ, which is limited to (i) insert or (ii) remove at most k edges. How many possible perturbations are there (in big-O notation w.r.t. the number of nodes N and number of edges E)?

3. Given a graph with 2810 nodes and 7336 edges. What value of p_a do we need to choose if in expectation we want to sample 7336 further edges?

Summary

- GNNs are **not robust** to adversarial attacks.

- GNN robustness/certification is a highly **active research area**.
  - To date there exists **no defense** against **structure attacks** that consistently improves results; standard methods such as **adversarial training** do not seem to work well.
- **Robustness certification** of GNNs is challenging but possible
  - specialized approaches enable to exploit structure of the GNN models
- **Randomized smoothing** can be adapted to **discrete input** data via **Bernoulli random variables**

  $\rightarrow$ We draw Monte Carlo samples for $g(\mathbf{x})$ and obtain the certified radii analytically.

  $\rightarrow$ Most importantly, **randomized smoothing with the proposed noise model works for all models with binary input data**: GNNs, CNNs, SVMs, Decision Trees, …

References: Attacks

- Bojchevski, Aleksandar, and Stephan Günnemann. "Adversarial attacks on node embeddings via graph poisoning." ICML 2019.
- Zügner, Daniel, Amir Akbarnejad, and Stephan Günnemann. "Adversarial attacks on neural networks for graph data." *KDD* 2018.
- Zügner, Daniel and Stephan Günnemann. "Adversarial attacks on Graph Neural Networks via Meta Learning." *ICLR 2019*.

References: Improving Robustness

- Negin Entezari, Saba A. Al-Sayouri, Amirali Darvishzadeh, and Evangelos E. Papalexakis. All you need is Low (rank): Defending against adversarial attacks on graphs. WSDM 2020 - Proceedings of the 13th International Conference on Web Search and Data Mining, pages 169–177, 2020.
- Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. Adversarial examples for graph data: Deep insights into attack and defense. IJCAI International Joint Conference on Artificial Intelligence, 2019.
- Kaidi Xu, Hongge Chen, Sijia Liu, Pin Yu Chen, Tsui Wei Weng, Mingyi Hong, and Xue Lin. Topology attack and defense for graph neural networks: An optimization perspective. IJCAI International Joint Conference on Artificial Intelligence, 2019.

References: Certification

- Daniel Zügner and Stephan Günnemann. Certifiable robustness and robust training for graph convolutional networks. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 246–256, 2019.
- Daniel Zügner and Stephan Günnemann. Certifiable Robustness of Graph Convolutional Networks under Structure Perturbations. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2020.
- Aleksandar Bojchevski and Stephan Günnemann. Certifiable Robustness to Graph Perturbations. (NeurIPS), 2019
- Aleksandar Bojchevski, Johannes Klicpera and Stephan Günnemann. Efficient Robustness Certificates for Discrete Data: Sparsity-Aware Randomized Smoothing for Graphs, Images and More, ICML 2020