The paper with augmentation techniques for graphs, a review of different methods (<https://arxiv.org/pdf/2202.08871.pdf> )

**Mixup:**

1. Mixup (<https://arxiv.org/pdf/1710.09412.pdf> ) – the first paper about mixup for traditional machine learning, images, text. Mixup on input data.

2. Manifold mixup (<http://proceedings.mlr.press/v97/verma19a/verma19a.pdf> ) - mixup on k-layer of a NN. Mixup on inner representation of data.

Mixup with graphs:

1. ifMixup: Towards Intrusion-Free Graph Mixup (<https://arxiv.org/pdf/2110.09344.pdf> ) - mixup on adjacency matrices with two graphs.

2. GraphMix (Bengio’s paper) (<https://ojs.aaai.org/index.php/AAAI/article/view/17203> ) - two models: GNN and FCN are used together and their weights are shared.

3. G-Mixup (<https://arxiv.org/pdf/2202.07179.pdf> ) - find a graphon for every graph class, then mix two graphons, after that generate new graphs from a mixed graphon and classify generated graphs.

4. GRAPHMAD (<https://arxiv.org/pdf/2210.15721.pdf> ) - very short paper where the authors applied nonlinear graph mixup, convex clustering, and different mixup functions for data samples and their labels. We didn’t discuss this paper a lot.

5. GraphMixup (<https://dl.acm.org/doi/pdf/10.1145/3442381.3449796> ) - the one that was printed during the meeting, mixup is applied next to the final representation of a graph.

**New methods:**

6. GAUG ( <https://arxiv.org/pdf/2006.06830.pdf> ) - first estimate the probabilities of edges, then mix these probabilities with real adjacency matrix to get new connection and remove some existing connections.

7. Mixup for node classification for imbalanced classes (<https://arxiv.org/pdf/2106.11133.pdf> ) - need to read more about it, but it has Feature Mixup, Edge Mixup, and Reinforcement Mixup.

8. Graph Transplant: Node Saliency-Guided Graph Mixup (<https://arxiv.org/pdf/2111.05639.pdf> ) - define core nodes, their neighbors and then mix them up.

The idea of graphon was used in G-Mixup paper (we mix up graphons of different classes). **Graphon estimation**:

1. USVT (Universal Singular Value Thresholding) (<https://arxiv.org/abs/1212.1247> ) - originally was proposed for matrices (not graphs spesific). The idea is to find the singular value decomposition of initial matrix and then simplify it using a threshold. I didn't go into details. In the paper the most useful chapters are 1.1, 1.2 - algorithm itself; 2.2, 2.6 - application for graphs.

2. MC (Matrix Completion) (<https://arxiv.org/abs/0901.3150> ) - was created for recommendation systems mostly, the idea is based on the singular value decomposition with some post-processing (cleaning). I didn't go into details. Chapter 1 describes the algorithm, other chapters are more theoretical with proofs.

3. LG (Largest gap) (<https://hal.archives-ouvertes.fr/hal-01190224/document> ):

Chapter 2 (2.2):

    - take a graph

    - calculate the number of neighbours for every node - a degree of a node (they also apply some normalization)

    - sort nodes based on their degree

    - calculate gaps between consecutive degrees

    - set a number of desired blocks Q

    - find Q-1 largest gaps

    - split nodes into Q blocks based on gaps from the previous step

I didn't go into theorems and equations.

In conclusion the authors say that LG method should be used for large graphs, not small ones (chapter 6.3, 8).

4. SBA (Stochastic blockmodel approximation) (<https://proceedings.neurips.cc/paper/2013/file/b7b16ecf8ca53723593894116071700c-Paper.pdf> )

    - set a threshold delta

    - take a node ni of a graph and assign in to a block Bk (k=1, ..., K)

        - take another node nj

        - calculate the distance d between ni and nj

        - if d <= delta squared -> add nj to Bk

        - repeat it for all nodes without a block

    - increase the index of a block

    - repeat it for all nodes without a block

5. SAS (Sorting and smoothing) (<https://arxiv.org/abs/1402.1888> ) - an extension of SBA, the same author

    - calculate degrees of the nodes

    - sort nodes based on their degrees

    - compute a histogram: set binwidth h and group nodes into bins - it reduces the size of a adjacency matrix

    - smooth the histogram (minimize total variation of a histogram)

Basically, in these methods we try to align nodes based on their degree and group nodes to several bins/blocks/groups (or kinda cluster nodes) based on node degree. Then we create a matrix for these groups and calculate the probability of two nodes from different groups to be connected based on what we’ve seen in an original graph.

Validation of graphons:

    - take a known graphon g

    - generate a graph with a given number of nodes

    - find a graphon g' of the generated graph

    - calculate the difference between g and g'

    - repeat previous steps several times (generate new graphs) to calculate the average error and std of an error

(easy to see in SAS paper, chapter 5).

There are more ways to find a graphon,  also recent papers about it. The one that can be interesting for our discussion: Learning Graphons via Structured Gromov-Wasserstein Barycenters (2020) (<https://arxiv.org/abs/2012.05644> )

**New methods:**

6. Neighborhood smoothing (NBS) (<https://arxiv.org/pdf/1509.08588v1.pdf> ) - calculate distance between nodes, then select neighbors based on this distance and group nodes.

7. NBSE: Neighbourhood Smoothing, Extended (<https://arxiv.org/pdf/1906.00494.pdf> ) - extended version of NBS. Sub-graphs with intersection are selected, then they are analyzed all together.

8. ICE - Iterative Connecting Probability Estimation (<https://proceedings.neurips.cc/paper/2021/file/0919b5c38396c3f0c41f1112d538e42c-Paper.pdf> ) - extension of NBS with two and more iterations.