Graph neural networks

Marek Dědič

TZN-04, Monday 24th November, 2025

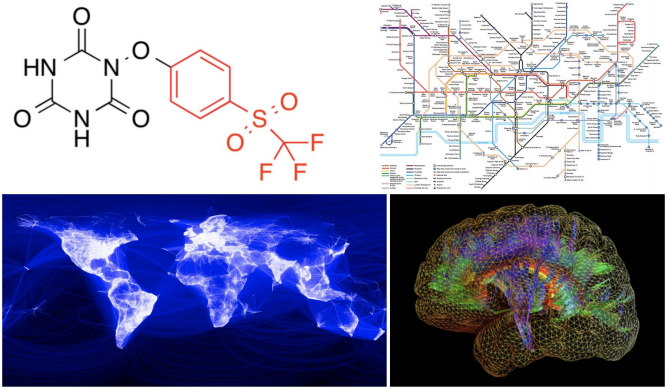
Contents

Motivation

- Motivation
- Tasks
- GNN
- DeepWalk
- node2vec
- GCN
- GraphSAGE

Motivation

- Graphs are a natural way to represent many different mathematical problems
- Examples: social networks, molecules, transportation networks, knowledge graphs, computer networks...



[1]

[1] Veličković et al., 2020.

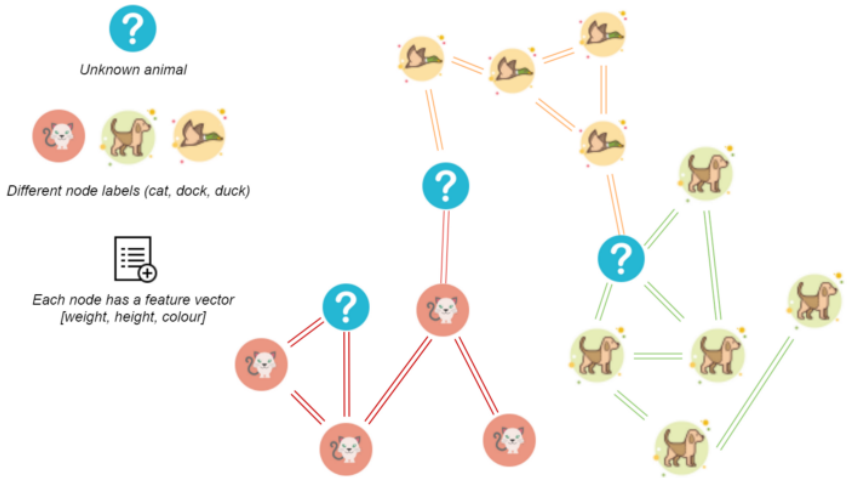
Definitions

- Graph $G = (V, E)$
- $ne[v]$ is the set of nodes connected to v by an edge
- $co[v]$ is the set of edges leading to or from v
- x_v are features of a node v
- x_e are features of an edge e

Tasks

- Motivation
- Tasks
- GNN
- DeepWalk
- node2vec
- GCN
- GraphSAGE

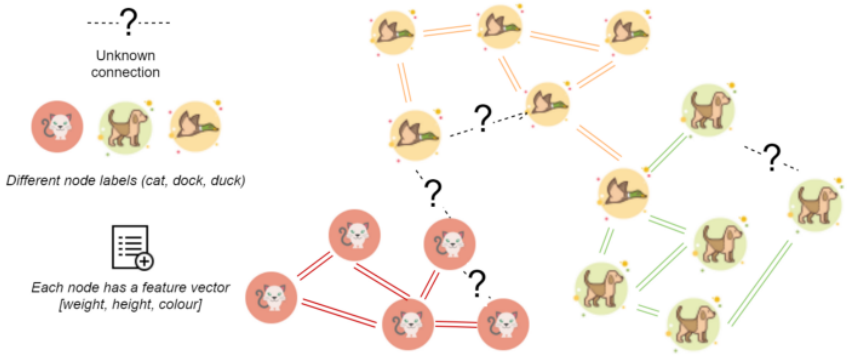
Node classification



2

[2] Kubara, 2020.

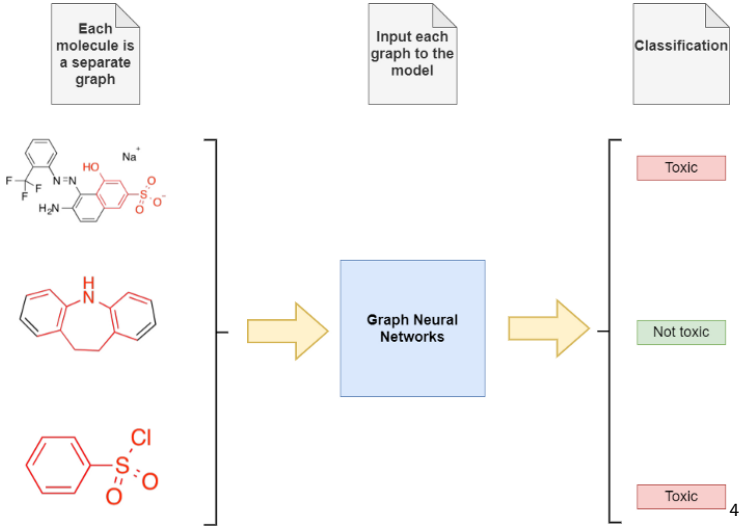
Link prediction



3

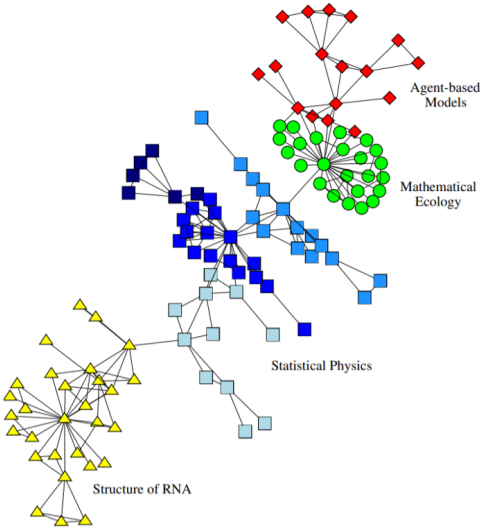
[3] Kubara, 2020.

Graph learning



[4] Kubara, 2020.

Community detection



[5]

[5] Girvan et al., 2002.

GNN

- Motivation
- Tasks
- GNN
- DeepWalk
- node2vec
- GCN
- GraphSAGE

GNN 1

- We're solving "node classification"
- We adapt neural networks to graph data
- Formally, classification on the space $\mathcal{G} \times \mathcal{N}$ – pairs graph + one of its nodes
- We assume features $\mathbf{x}_v$ and $\mathbf{x}_e$ for each node v and edge e
- We model for each node v a hidden state $\mathbf{h}_v$ capturing information about v and its neighbourhood

GNN 2

- We model for each node v a hidden state $\mathbf{h}_v$ capturing information about v and its neighbourhood as

$$\mathbf{h}_v = f\left(\mathbf{x}_v, \mathbf{x}_{co[v]}, \mathbf{h}_{ne[v]}, \mathbf{x}_{ne[v]}\right)$$

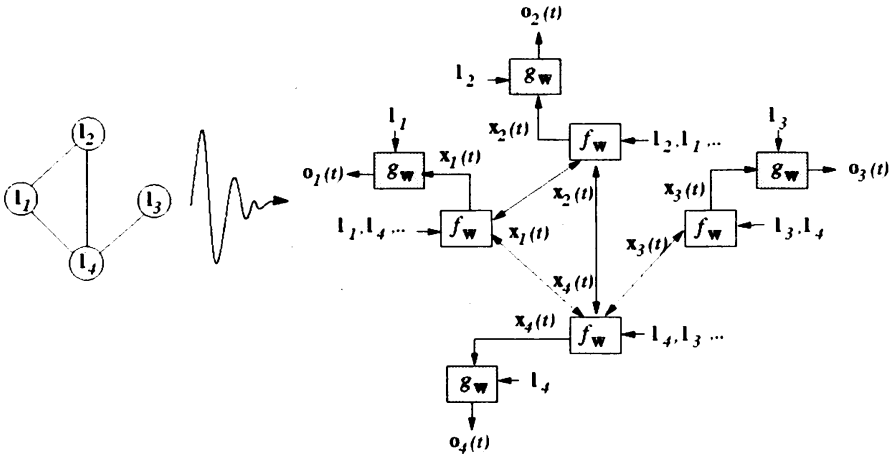
- ...and an output for node v as

$$\mathbf{o}_v = g\left(\mathbf{h}_v, \mathbf{x}_v\right)$$

- We can calculate the hidden state $\mathbf{h}_v$ iteratively for all nodes as

$$\mathbf{H}^{t+1} = F\left(\mathbf{H}^t, \mathbf{X}\right)$$

GNN 3



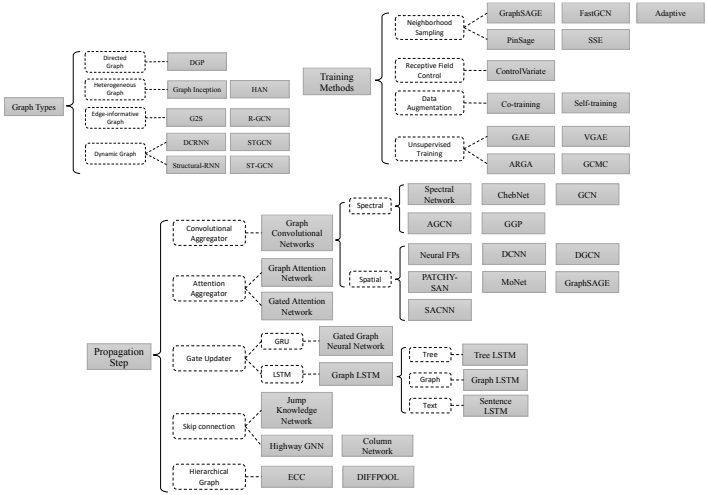
6

[6] Gori et al., 2005.

GNN – conclusion

- We can view the iterative application of function F as an RNN
- We may replace the iterative process with an MLP
- A GNN doesn't use edge hidden states, only node hidden states

GNN – modifications



[7]

[7] Zhou et al., 2019.

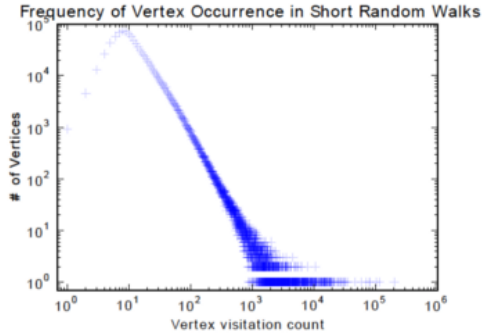
DeepWalk

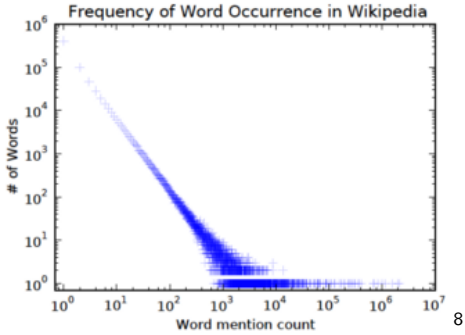
- Motivation
- Tasks
- GNN
- DeepWalk
- node2vec
- GCN
- GraphSAGE

DeepWalk 1

- Inspired by word2vec and skip-gram methods from NLP
- M random walks of length L from a given node to generate sequences of nodes, analogous to sentences

DeepWalk 2

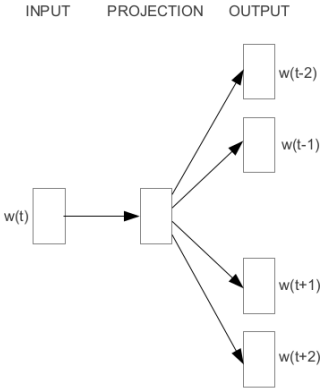




[8]

[8] Perozzi et al., 2014.

DeepWalk – skip-gram



Skip-gram

9

[9] Mikolov et al., 2013.

DeepWalk 4

- For each vector v we learn its representation $f : \mathcal{V} \to \mathbb{R}^{|\mathbb{V}| \times d}$
- We try to maximize the value

$$\mathrm{P}\left(\{v_{i-w}, \ldots, v_{i+w}\} \setminus v_i | f\left(v_i\right)\right)$$

- Assuming independence, we simplify

$$\mathrm{P}\left(\{v_{i-w}, \ldots, v_{i+w}\} \setminus v_i | f\left(v_i\right)\right) = \prod_{\substack{j=i-w \\ j \neq i}}^{i+w} \mathrm{P}\left(v_j | f\left(v_i\right)\right)$$

- This is a problem solved by word2vec
- It's further necessary to optimize using hierarchical softmax

DeepWalk – conclusion

- Using ideas from word2vec to learn on graphs
- DeepWalk cannot classify new nodes – it is necessary to retrain the model

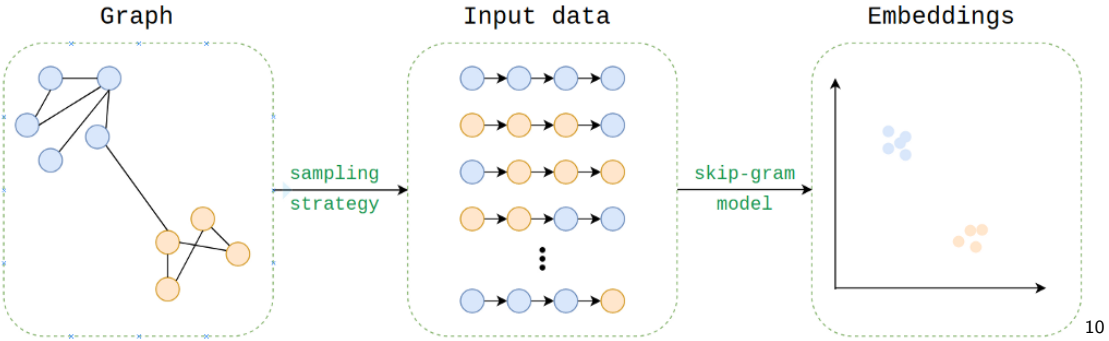
node2vec

- Motivation
- Tasks
- GNN
- DeepWalk
- node2vec
- GCN
- GraphSAGE

node2vec 1

- Also inspired by word2vec, but in a different way
- We are looking for an embedding $\mathrm{node2vec} : \mathcal{G} \to \mathbb{R}^n$

node2vec 2



10

[10] Cohen, 2018.

node2vec 3

- For each vector v we learn its representation $f : \mathcal{V} \rightarrow \mathbb{R}^{|\mathbb{V}| \times d}$
- We try to maximize the value

$$\mathrm{P}\left(ne\left[v\right]|f\left(v\right)\right)$$

- Assuming independence, we can simplify

$$\mathrm{P}\left(ne\left[v\right]|f\left(v\right)\right) = \prod_{v_i \in ne[v]} \mathrm{P}\left(v_i|f\left(v\right)\right)$$

- It's further necessary to optimize using hierarchical softmax and negative sampling

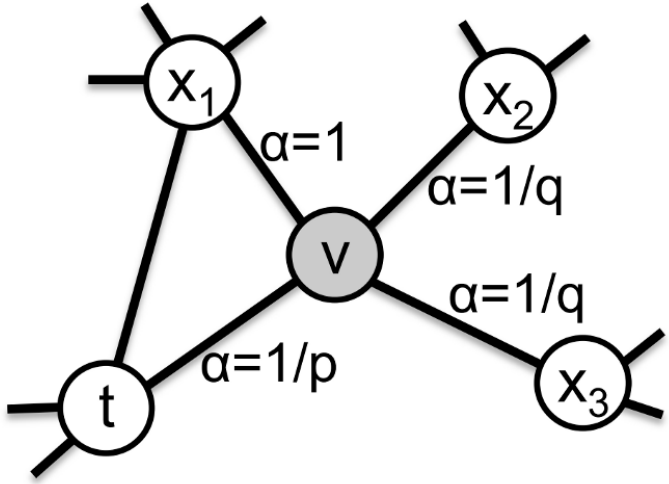
node2vec 4

- How to find $ne[v]$?
- 2 extrema: BFS, DFS
- Regular random walk: If I'm at node v, I go to some node connected to v with equal probability for all such nodes
- We modify the random walk to interpolate between BFS and DFS
- If coming to v from t, we assign weights to candidates for the next node x as

$$\alpha(t,x) = \begin{cases} \frac{1}{p} & \text{for } d_{tx} = 0 \\ 1 & \text{for } d_{tx} = 1 \\ \frac{1}{q} & \text{for } d_{tx} = 2 \end{cases}$$

where d_{tx} is the length of the shortest path from t to x and p, q are model parameters

node2vec 5



11

[11] Grover et al., 2016.

node2vec – conclusion

- Using ideas from word2vec to learn on graphs
- An improvement over DeepWalk
- node2vec cannot classify new nodes – it is necessary to retrain the model

GCN

- Motivation
- Tasks
- GNN
- DeepWalk
- node2vec
- GCN
- GraphSAGE

GCN 1

- We are solving “node classification”
- Inspired by CNNs

GCN 2

- Expressed as a neural network operating on the whole graph, similar to GNN
- Each layer has the form

$$\mathbf{H}^{l+1} = f\left(\mathbf{H}^{(l)}, \mathbf{A}\right)$$

  where $\mathbf{A}$ is the adjacency matrix
- We can express a simple layer as

$$f\left(\mathbf{H}^{(l)}, \mathbf{A}\right) = \sigma\left(\mathbf{A}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right)$$

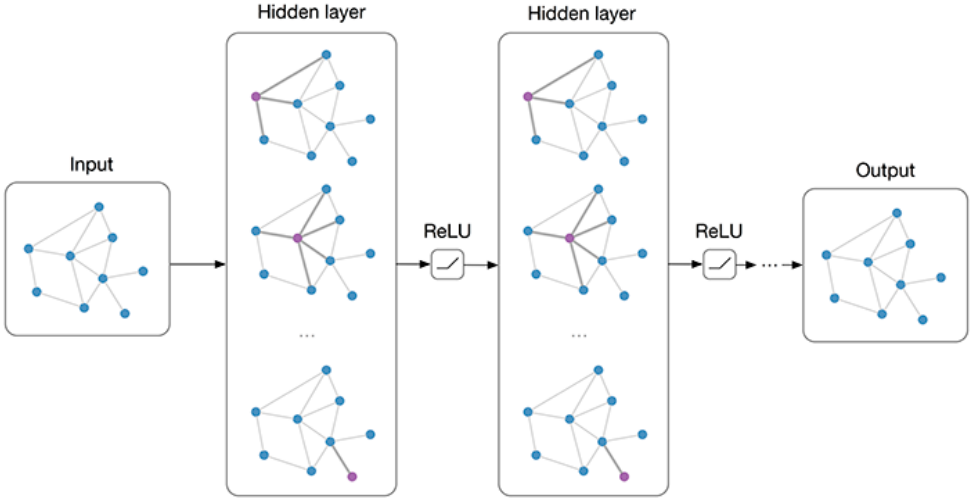
  where $\mathbf{W}$ are the weights and σ is the activation function

GCN 3

- If we multiplied by $\mathbf{A}$, we would forget the information in the given node at each step. We solve this by using instead the matrix $\widehat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$
- Because the matrix $\mathbf{A}$ is not normalized, the scale would change after each layer. To prevent this, we use instead the symmetrically normalized matrix $\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ where $\mathbf{D}$ is the diagonal matrix of degrees of individual nodes
- We thus obtain one layer of the network as

$$f\left(\mathbf{H}^{(l)}, \mathbf{A}\right) = \sigma\left(\widehat{\mathbf{D}}^{-\frac{1}{2}}\widehat{\mathbf{A}}\widehat{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right)$$

GCN 4



12

[12] Kipf, 2016.

GCN – conclusion

- Using ideas from CNNs to learn on graphs
- GCNs are a first-order approximation of localized spectral filters on graphs – see Kipf; Welling, 2017

GraphSAGE

- Motivation
- Tasks
- GNN
- DeepWalk
- node2vec
- GCN
- GraphSAGE

GraphSAGE 1

- GraphSAGE is and inductive framework for generating node embeddings – there is no need to have the whole graph during training
- We learn a representation for each node v using the representations of its neighbors, i.e.

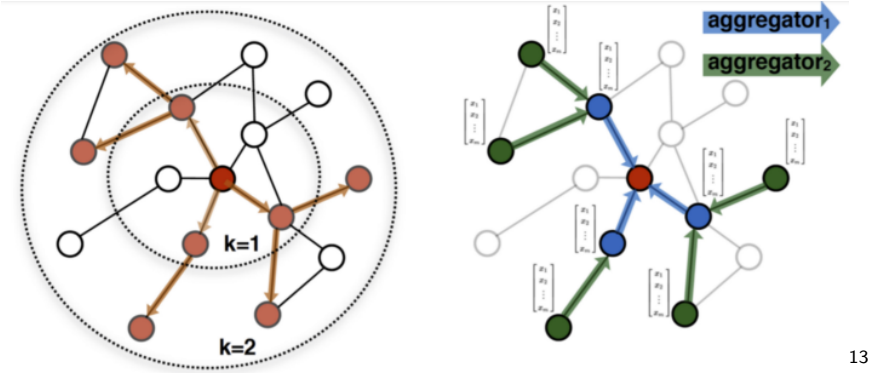
$$\mathbf{h}_v^{(l+1)} = \sigma\left(\mathbf{W}^{(l)} \cdot \text{concat}\left(\mathbf{h}_v^{(l)}, \mathbf{h}_{ne[v]}^{(l)}\right)\right)$$

where

$$\mathbf{h}_{ne[v]}^{(l)} = \text{AGGREGATE}_l\left(\left\{\mathbf{h}_u^{(l)} \middle| u \in ne\,[v]\right\}\right)$$

- We need a special loss function

GraphSAGE 2



13

[13] Hamilton et al., 2017.

GraphSAGE 3

- We can choose an aggregator for each layer independently
- The mean aggregator yields a GCN-like layer
- An LSTM aggregator can capture more complex relationships (we need to randomly permute the neighbors)
- Pooling:

$$\text{AGGREGATE}_l = \max \left\{ \sigma \left(\mathbf{W}_{\text{pool}} \mathbf{h}_u^{(l)} + \mathbf{b} \right) \middle| u \in ne\,[v] \right\}$$

GraphSAGE – conclusion

- An inductive approach is suitable for graphs where nodes are added after training
- We don't need to re-train the model, it will work with new data


That's all Folks!

Bibliography I

COHEN, Elior, 2018. *node2vec: Embeddings for Graph Data* [Medium] [online]. 2018-04-23. [visited on 2020-11-03]. Available from: `https://towardsdatascience.com/node2vec-embeddings-for-graph-data-32a866340fef`.

GIRVAN, M.; NEWMAN, M. E. J., 2002. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences* [online]. Vol. 99, no. 12, pp. 7821–7826 [visited on 2020-11-03]. ISSN 0027-8424, ISSN 1091-6490. Available from DOI: 10.1073/pnas.122653799. Publisher: National Academy of Sciences Section: Physical Sciences.

GORI, M.; MONFARDINI, G.; SCARSELLI, F., 2005. A new model for learning in graph domains. In: *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* Vol. 2, 729–734 vol. 2. Available from DOI: 10.1109/IJCNN.2005.1555942. ISSN: 2161-4407.

GROVER, Aditya; LESKOVEC, Jure, 2016. node2vec: Scalable feature learning for networks. In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864.

HAMILTON, Will; YING, Zhitao; LESKOVEC, Jure, 2017. Inductive representation learning on large graphs. In: *Advances in neural information processing systems*, pp. 1024–1034.

Bibliography II

KIPF, Thomas N., 2016. *How powerful are Graph Convolutional Networks?* [online]. 2016-09-30. [visited on 2020-11-04]. Available from: http://tkipf.github.io/graph-convolutional-networks/.

KIPF, Thomas N.; WELLING, Max, 2017. Semi-Supervised Classification with Graph Convolutional Networks. In: *5th International Conference on Learning Representations, {ICLR} 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. Toulon, France: OpenReview.net. Available also from: https://openreview.net/forum?id=SJU4ayYgl.

KUBARA, Kacper, 2020. *Machine Learning Tasks on Graphs* [Medium] [online]. 2020-09-28. [visited on 2020-11-03]. Available from: https://towardsdatascience.com/machine-learning-tasks-on-graphs-7bc8f175119a.

MIKOLOV, Tomas; CHEN, Kai; CORRADO, Greg; DEAN, Jeffrey, 2013. Efficient Estimation of Word Representations in Vector Space [online] [visited on 2020-11-03]. Available from: https://openreview.net/forum?id=idpCdOWtqXd60.

PEROZZI, Bryan; AL-RFOU, Rami; SKIENA, Steven, 2014. Deepwalk: Online learning of social representations. In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710.

Bibliography III

VELIČKOVIĆ, Petar; DEAC, Andreea, 2020. *Opening Remarks*. Vienna. Available also from: `https://slideslive.at/icml-2020/graph-representation-learning-and-beyond-grl`. ICML 2020.

ZHOU, Jie; CUI, Ganqu; ZHANG, Zhengyan; YANG, Cheng; LIU, Zhiyuan; WANG, Lifeng; LI, Changcheng; SUN, Maosong, 2019. Graph Neural Networks: A Review of Methods and Applications. *arXiv:1812.08434 [cs, stat]* [online] [visited on 2020-11-03]. Available from arXiv: `1812.08434`.