Machine Learning for Graphs and Sequential Data

Graphs – Node Embeddings

Lecturer: Prof. Dr. Stephan Günnemann

www.daml.in.tum.de

Summer Term 2020

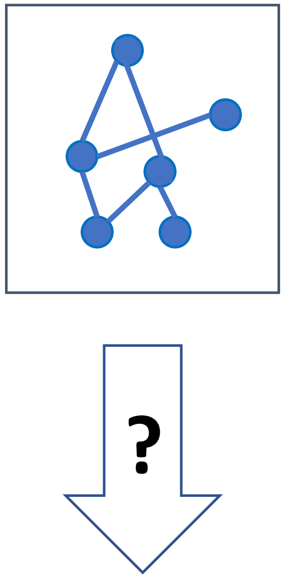
Data Analytics and
Machine Learning

Roadmap

- **Chapter: Graphs**

Challenge of ML on Graphs

- Difficult to apply traditional ML to graphs
  - How to encode graph structure?
  - Want to exploit it, so we need to handle it somehow
- Direct approaches violate basic properties and assumptions
  - Adjacency matrix as image is not invariant to node permutation
  - Concatenating features of neighbors produces variable length data
  - $O(N^2)$ scaling in runtime and feature size
- Traditional ML approaches have no concept of graphs
  - Would need to learn the graph structure implicitly from any encoding of the graph data
- We might even not have features - only structure

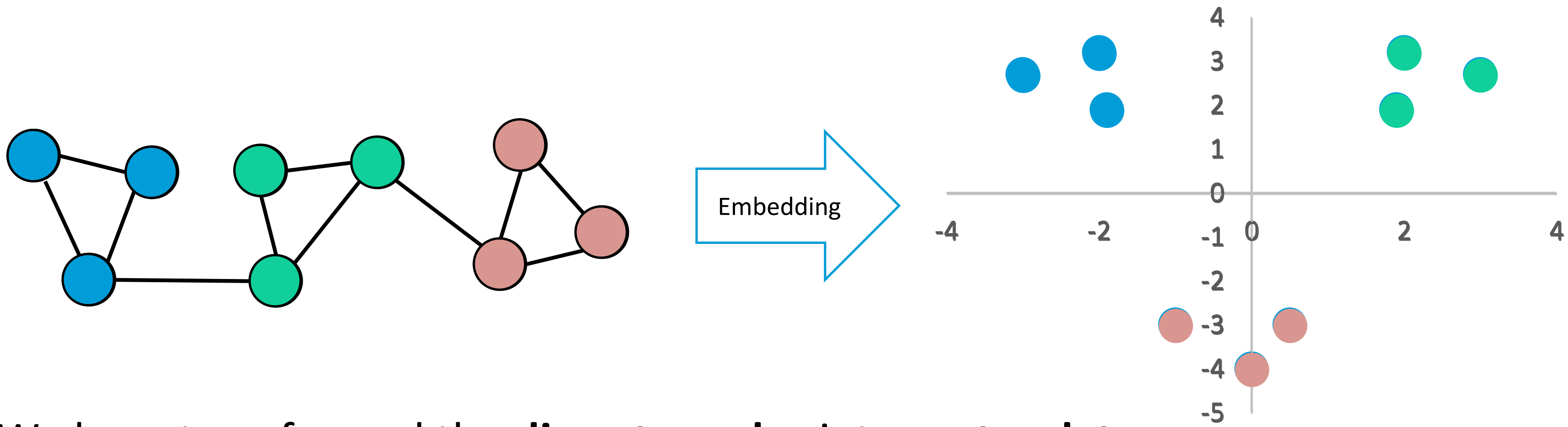


- k-means
- SVM
- etc.

Learning Node Representations

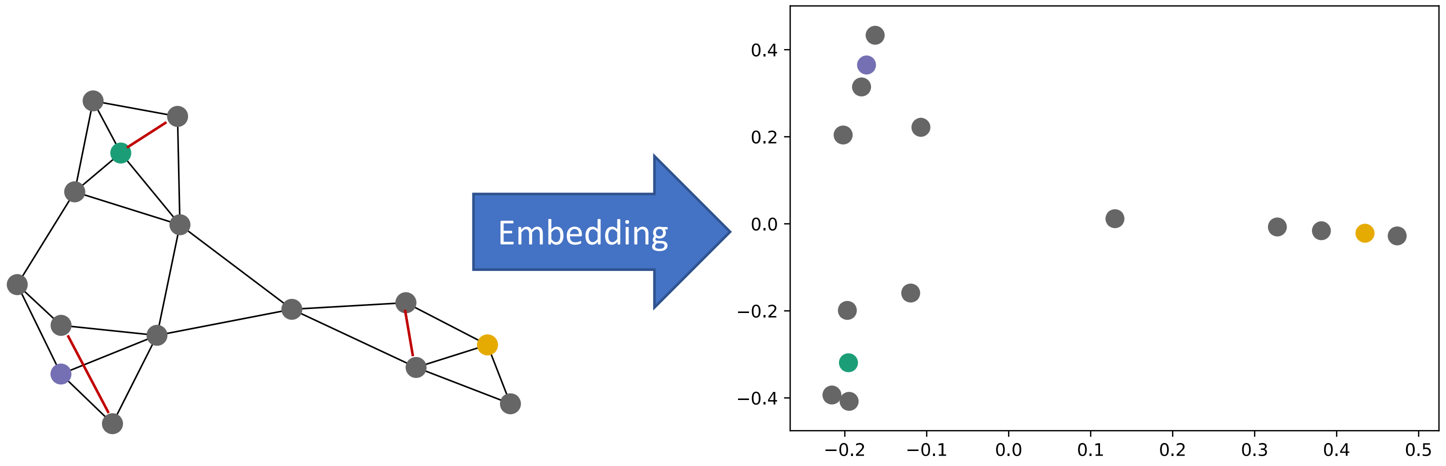
General approach:

- Transform the graph such that each *vertex* is represented by a vector
  - Node embedding function $\Phi: V \rightarrow \mathbb{R}^d$ maps each node to a point in $\mathbb{R}^d$ (e.g., the smallest d eigenvectors for spectral clustering).
  - Nodes close in the embedding space are “similar” w.r.t. graph structure



- We have transformed the **discrete nodes** into **vector data**.
  - We can now use **standard tools** for vector data to perform "downstream" tasks such as clustering or classification.

Applications



- **Clustering**

  Group the nodes into a set of clusters in an unsupervised way, e.g. using K-Means.

- **Semi-supervised classification**

  Given a small set of **labeled** nodes for which we know their class, classify the remaining nodes in the graph based on their representations.

- **Link prediction**
  - Predict likely (unobserved) links in the graph, e.g. friendship recommendation or “other users have watched/bought” lists.
  - $\Pr\big((i,j) \in E\big) \propto \Phi(v_i)^T \Phi(v_j)$

Types of Embeddings

- Many possible ways to define an embedding
  - Differences come from how we define similarity between nodes in the embedding space and which properties of the graph we are trying to capture
  - For example in role-based embeddings nodes with similar role (e.g. hubs) should be close together in the embedding even if they are far in the graph
  - We will see an NLP inspired approach in the deep embeddings section

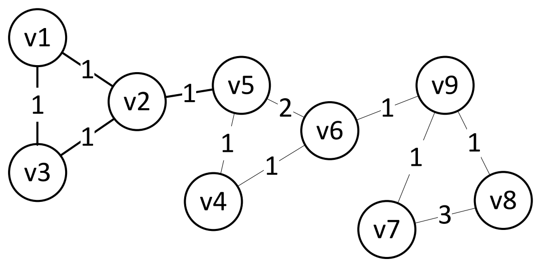
- Spectral embeddings → nodes are similar if they belong to the same clusters
- Deep embeddings → similarity depends on loss function and architecture

Roadmap

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

Recap: Spectral Clustering

- Construct the graph Laplacian L
- Compute the first k eigenvectors v_i of L in columns of H, i.e. $H_i^T = v_i$
- Represent the i-th node as the i-th row of H
- Cluster the vector representations, for example with k-means



2	-1	-1	0	0	0	0	0	0
-1	3	-1	0	0	-1	0	0	0
-1	-1	2	0	0	0	0	0	0
0	0	0	2	-1	-1	0	0	0
0	-1	0	-1	4	-2	0	0	0
0	0	0	-1	-2	4	0	0	-1
0	0	0	0	0	0	4	-3	-1
0	0	0	0	0	0	-3	4	-1
0	0	0	0	0	-1	-1	-1	3

Graph Laplacian L

-0.3333	-0.4376	0.2939
-0.3333	-0.3370	0.0890
-0.3333	-0.4376	0.2939
-0.3333	0.0000	-0.5878
-0.3333	-0.0584	-0.3829
-0.3333	0.0584	-0.3829
-0.3333	0.4376	0.2939
-0.3333	0.4376	0.2939
-0.3333	0.3370	0.0890

Eigenvectors of L

- Smallest eigenvalues of L: 0 ; 0.23 ; 0.7

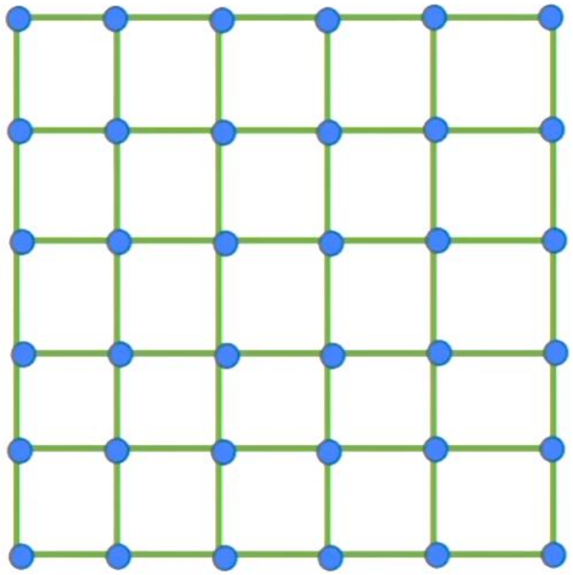
Spectral Clustering: Embedding View

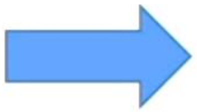
- Spectral embedding is based on the **eigenvectors of the graph Laplacian $\boldsymbol{L}$**
  - L encodes the structural behavior of the graph G
  - $|V| \times |V|$ adjacency matrix is transformed and reduced to $|V| \times k$ matrix H

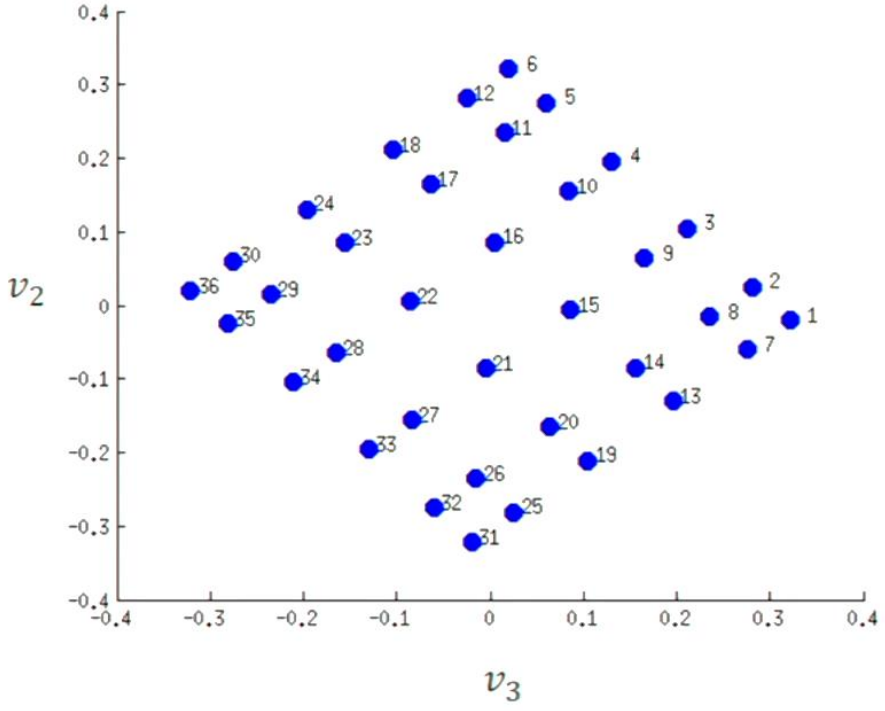
- Relation to PCA and dimensionality reduction
  - transformation is based on eigenvectors of the data-matrix and one retains only eigenvectors with smallest/largest eigenvalue

Examples (I)

- Spectral embedding of a grid



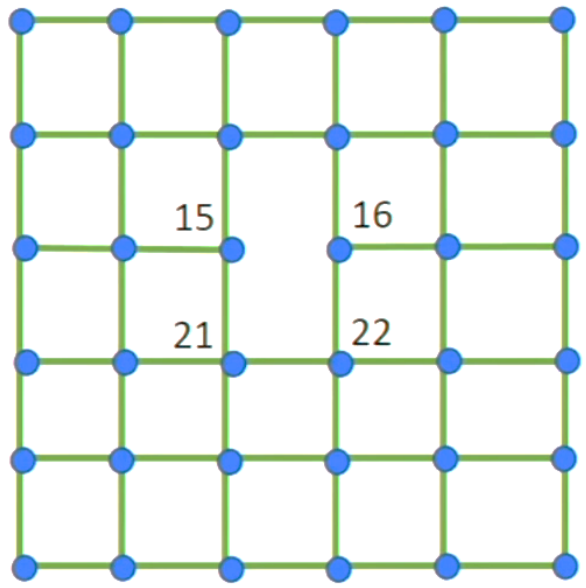


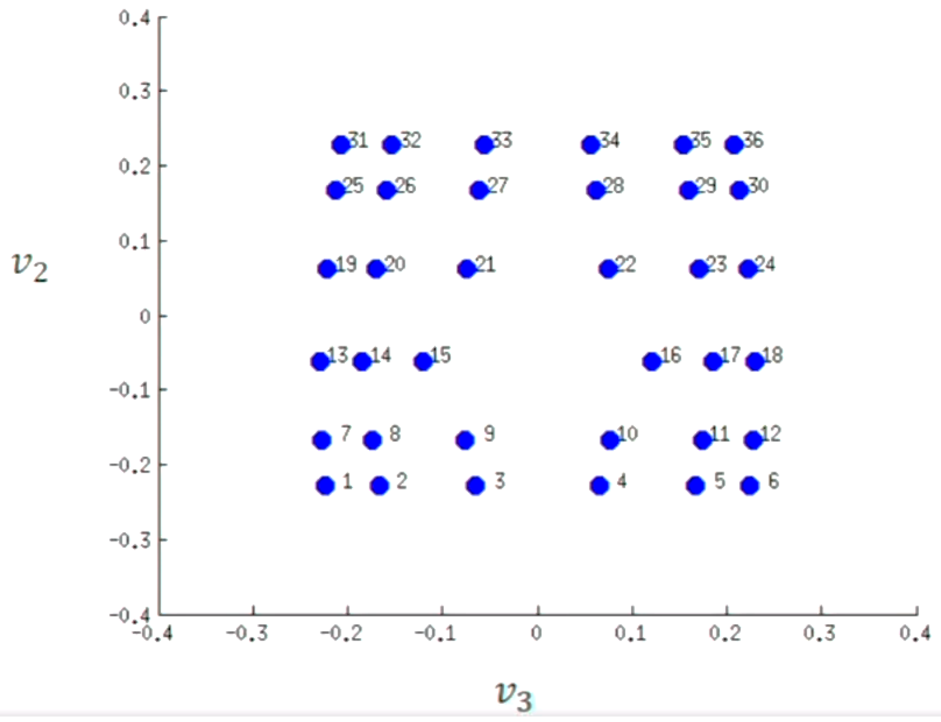


Data Analytics and Machine Learning

Examples (II)

- Spectral embedding of an incomplete grid with one edge removed





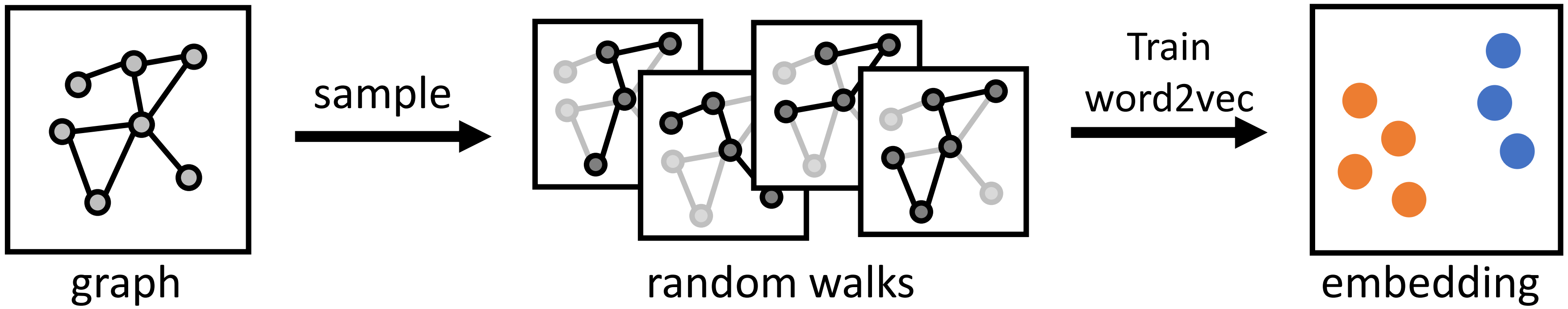
"Deep" Node Embedding Approaches

- Representation learning for graphs has become a very active field of research in recent years
  - specifically exploiting neural networks/deep learning etc.
- Goal: Try to capture more complex structure than spectral embeddings
  - thus, hopefully getting better results for specific downstream task
  - or capturing different notions of "similarity"

- Natural language processing (NLP) also deals with discrete data (words), and we can try to adapt successful techniques to graphs
  - **DeepWalk** [Perozzi2014] is a popular node embedding algorithm based on the word embedding model **word2vec** [Mikolov2013]

DeepWalk

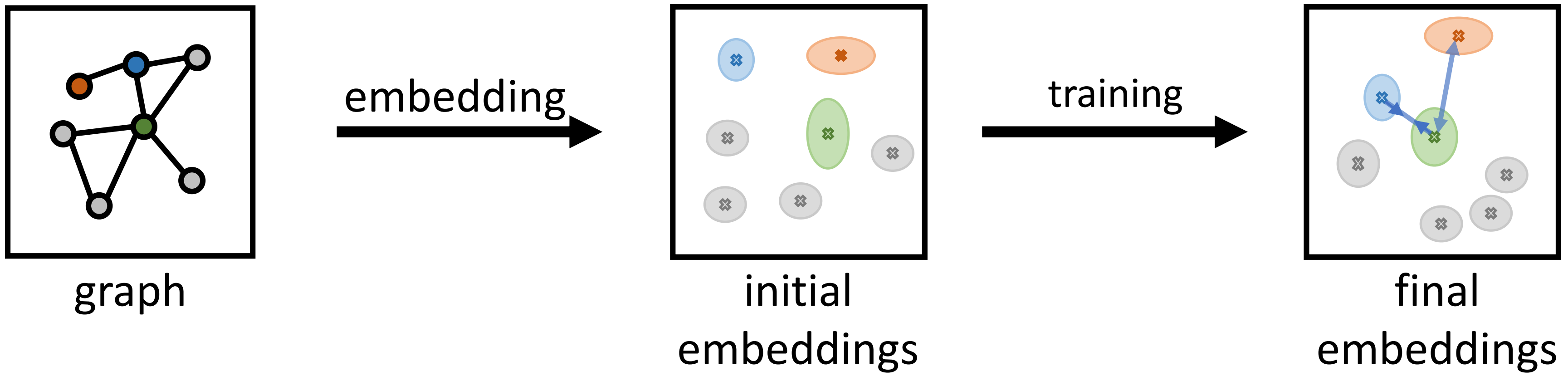
- **Idea**: transform the graph into a set of random walks and learn a **word2vec** model
- For every node v_i sample multiple random walks
- **Random walks** correspond to **sentences** in word2vec; both are sequences of discrete tokens (node-ids ≙ words)
- Train word2vec on the collection of all these "sentences"



- **Result**: nodes that are close to each other in the graph and share many neighbors get similar vector representations (embeddings)

Graph2Gauss

- **Idea**: map each node to a Gaussian distrib. in embedding space such that a node‘s 1st neighbors are closer than its 2nd neighbors and so forth (i.e. preserve ranking)
- Learn a mapping $f_\theta(\boldsymbol{v}) \rightarrow \mathcal{N}\left(\mu_\theta(\boldsymbol{v}), \Sigma_\theta(\boldsymbol{v})\right)$
- For each node $\boldsymbol{u}$ define a loss $\sum_{(v,w)} E_{uv}^2 + e^{-E_{uw}}$ where $E_{uv} = KL(f_\theta(\boldsymbol{u})||f_\theta(\boldsymbol{v}))$ and $\boldsymbol{v}$ and $\boldsymbol{w}$ are all node pairs such that $\boldsymbol{v}$ is closer to $\boldsymbol{u}$ than $\boldsymbol{w}$ (ranking loss)

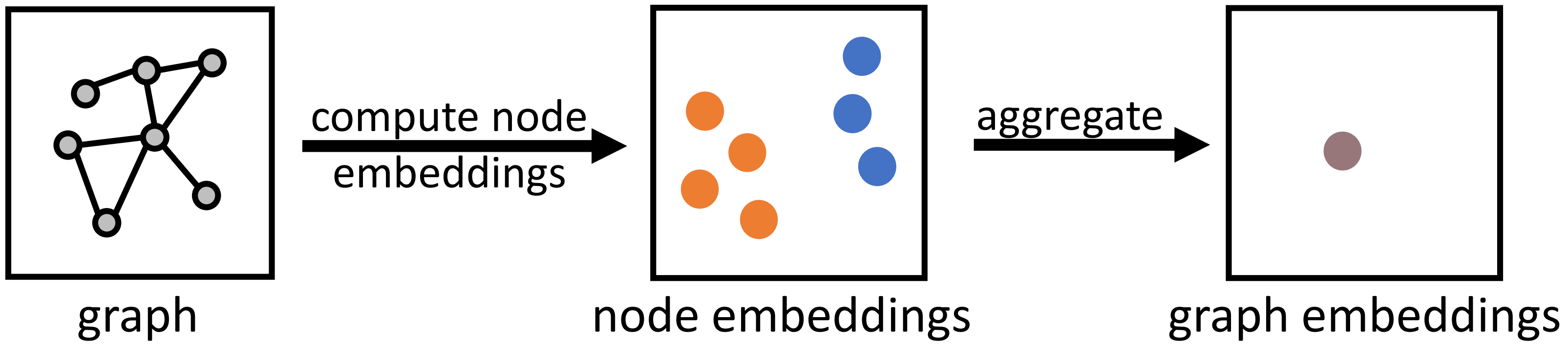


- **Result**: distances in embedding space correlate with distances in the graph and the Gaussian variances express how certain the model is about the embedding

[Bojchevski2018]

Graph Embeddings (vs. Node Embeddings)

- Tasks such as predicting molecule properties work on the graph-level instead of with individual nodes
- **Idea**: Leverage node embeddings to embed graphs
- **Problem**: Graphs have different numbers of nodes
  - simple concatenation produces incompatible dimensionalities
- **Solution**: Aggregate all node embeddings into one graph embedding, for example with mean pooling
  - more advanced principles available



Summary

- Node embeddings translate discrete graph structure information into continuous data
- With node embeddings we can even find fixed-length representations for the graph as a whole
- Classic spectral methods only use graph structure, deep methods can easily combine structural and attribute information
- Embeddings can be fed into ML methods for vector data, e.g.
  - k-means
  - SVMs, NNs
  - visualization in 2D/3D

Questions

- How can you use node embeddings to visualize the structure of a graph?

- Consider two nodes u and v in a graph that share the same node attributes x but are far apart in the graph. What can you say about the embeddings that Graph2Gauss would find for these nodes? Why do the other methods work better?

Reading Material

- [Perozzi2014] Perozzi, B., Al-Rfou, R., & Skiena, S. (2014, August). Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 701-710)
- [Mikolov2013] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111-3119)
- [Bojchevski2018] Bojchevski, A., Günnemann, S. (2018). Deep Gaussian Embedding of Graphs: Unsupervised Inductive Learning via Ranking. In *International Conference on Learning Representations* (ICLR)