NETWORK EMBEDDING

NAMES

- Graph embedding / Network embedding
- Representation learning on networks
  - Wikipedia: **Representation learning = feature learning**, as opposed to **manual feature engineering**
- Embedding => Latent space

IN CONCRETE TERMS

- A graph is composed of
  - Nodes (possibly with labels)
  - Edges (possibly directed, weighted, with labels)

- A graph embedding technique in **d** dimension will assign a vector of length **d** to each node, that will be useful for **what we want to do with the graph**.

- A vector can be assigned to an edge *(u,v)* by combining vectors of *u* and *v*

WHAT TO DO WITH EMBEDDINGS?

- Two possible ways to use an embedding:
  - Supervised learning:
    - Algorithm learn to predict *something* from the features in the embedding
  - Unsupervised learning:
    - The *distance* between vectors in the embedding is used for *something*

WHAT CAN WE DO WITH EMBEDDINGS ?

EMBEDDING TASKS

- Common tasks:
  - Link prediction (supervised)
  - Graph reconstruction (unsupervised link prediction ? / ad hoc)
  - Community detection (unsupervised)
  - Node classification (supervised community detection ?)
  - Visualisation (distances, like unsupervised)
  - Role definition (unsupervised, some special embeddings)

OVERVIEW OF MOST POPULAR METHODS

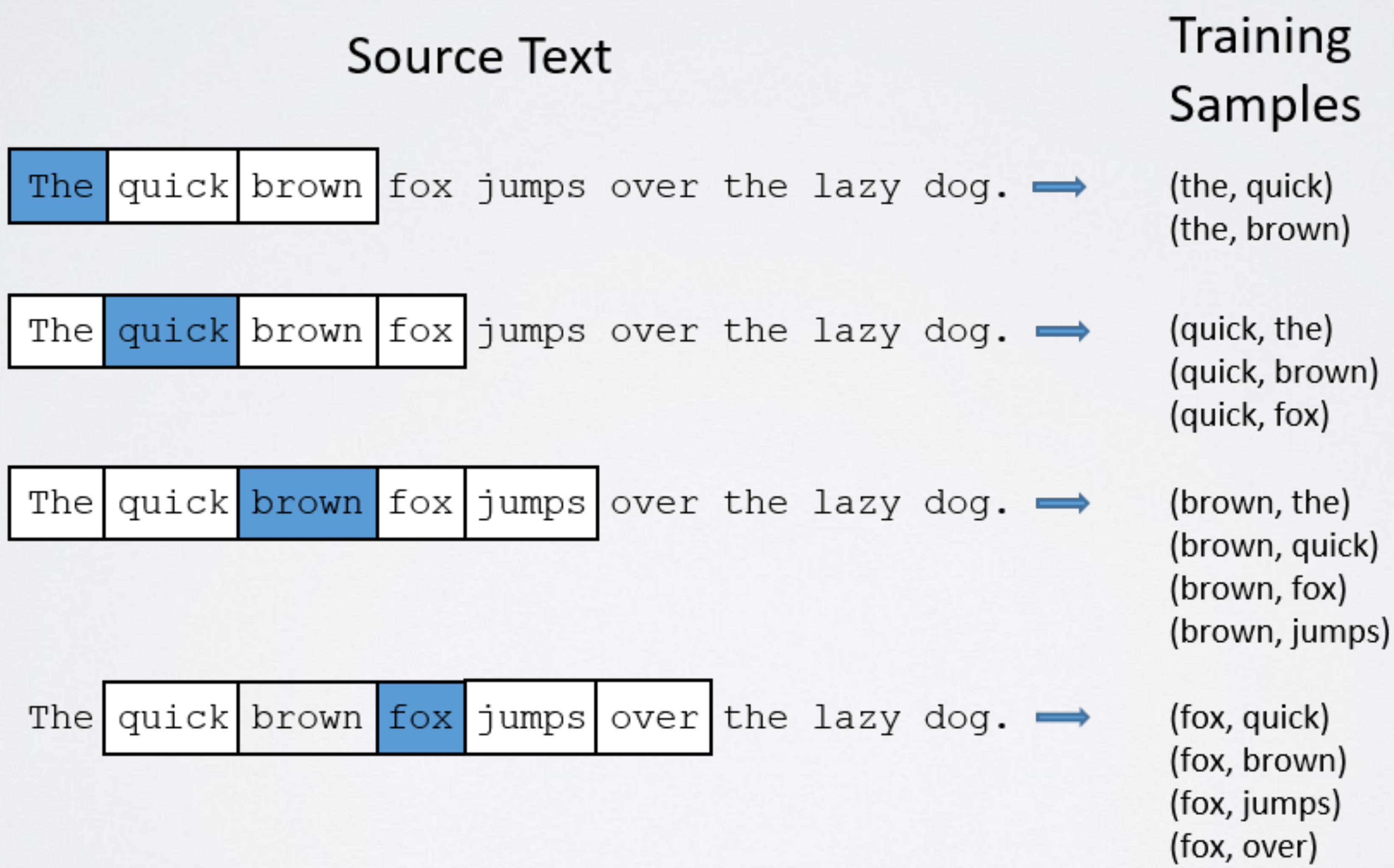
1. RANDOM WALK BASED

DEEPWALK

- The first "modern" graph embedding method
- Adaptation of **word2vec/skipgram** to graphs

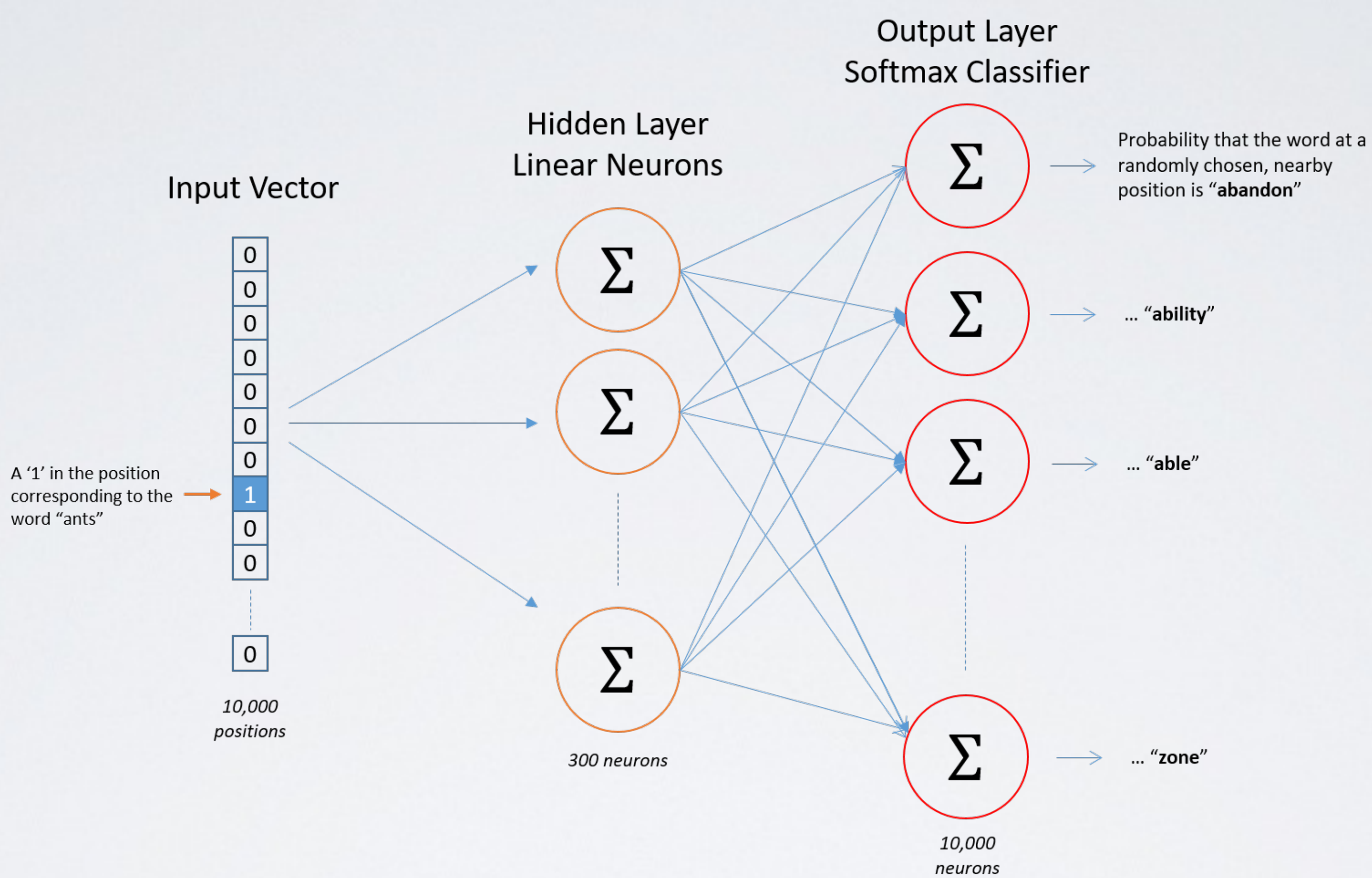
SKIPGRAM

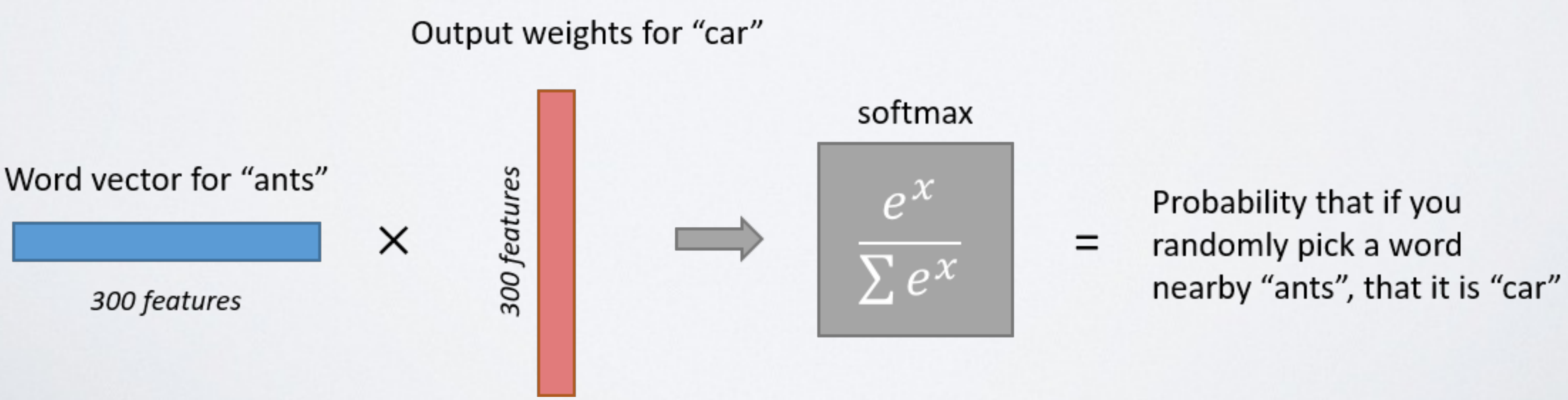
Word embedding
Natural language => vectors



[http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/]

SKIPGRAM

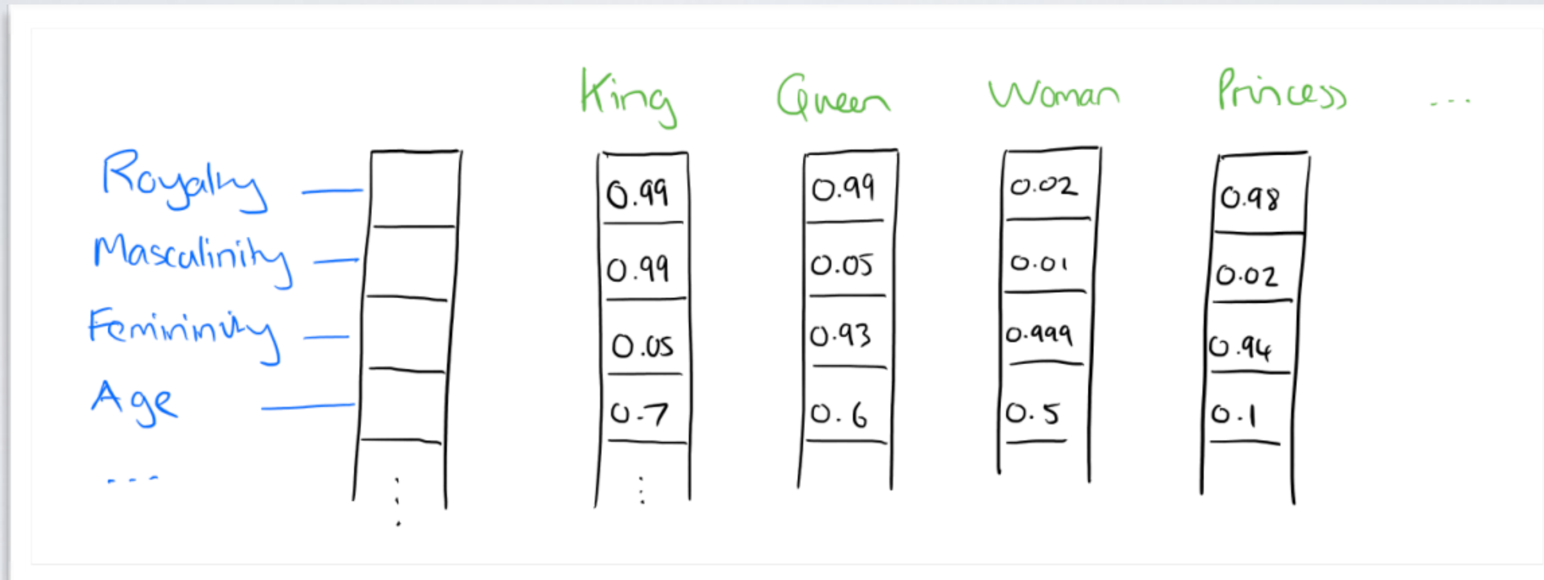

Output Layer
Softmax Classifier
Hidden Layer
Linear Neurons
Input Vector
0
0
0
0
0
0
0
1
0
0
0
A '1' in the position corresponding to the word "ants"
10,000 positions
300 neurons
10,000 neurons
Probability that the word at a randomly chosen, nearby position is "**abandon**"
... "**ability**"
... "**able**"
... "**zone**"

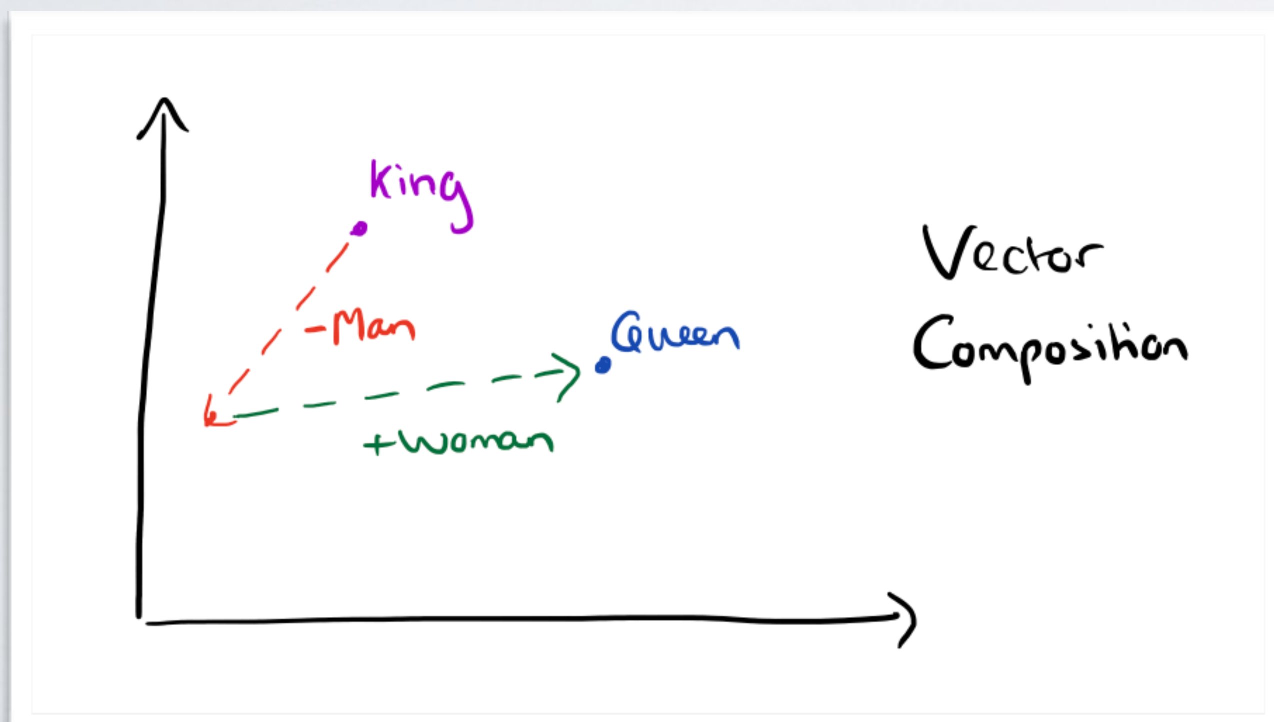

Output weights for "car"
Word vector for "ants"
300 features
×
300 features
softmax
$\frac{e^x}{\sum e^x}$
=
Probability that if you randomly pick a word nearby "ants", that it is "car"

GENERIC "SKIPGRAM"

- Algorithm that takes an input:
  - The element to embed
  - A list of "context" elements
- Provide as output:
  - An embedding with nice properties
    - Works well for machine learning
    - Similar elements are close in the embedding
    - Somewhat preserves the overall structure

GENERIC “SKIPGRAM”





[https://blog.acolyer.org/2016/04/21/the-amazing-power-of-word-vectors/]

GENERIC "SKIPGRAM"

Table 8: *Examples of the word pair relationships, using the best word vectors from Table 4 (Skip-gram model trained on 783M words with 300 dimensionality).*

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

[https://blog.acolyer.org/2016/04/21/the-amazing-power-of-word-vectors/]

DEEPWALK

- Skipgram for graphs:
  - 1)Generate "sentences" using random walks
  - 2)Apply Skipgram
- Parameters: dimensions *d*, RW length *k*

NODE2VEC

- Use biased random walk to tune the context to capture what we want
  - "Breadth first" like RW => local neighborhood (edge probability ?)
  - "Depth-first" like RW => global structure ? (Communities ?)
  - 2 parameters to tune:
    - p: likelihood revisiting node
    - q: bias towards neighbors of the previous nodes (BFS)

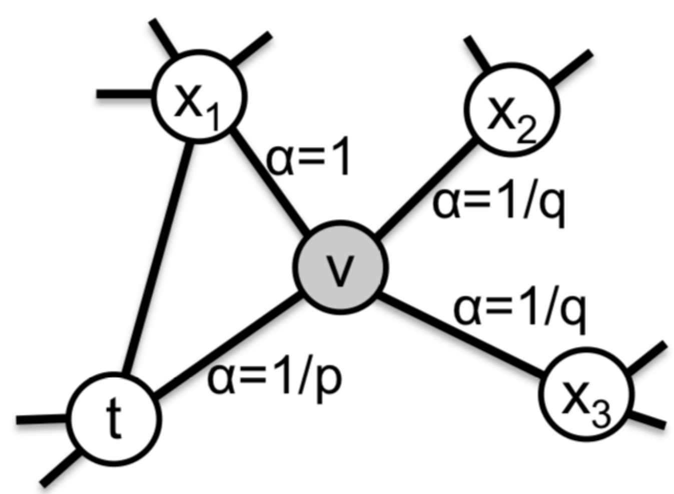


Figure 2: Illustration of the random walk procedure in *node2vec*. The walk just transitioned from t to v and is now evaluating its next step out of node v. Edge labels indicate search biases α.

RANDOM WALK METHODS

- What is the objective function ?
- How to interpret the distance between nodes in the embedding ?
- =>Dot product/cosine distance (u,v) *inversely related to* probability of reaching v from u with a RW of length *k*

II. "OLD" METHODS

LE

- Laplacian Eigenmaps (2001)

Laplacian Eigenmaps [25] aims to keep the embedding of two nodes close when the weight W_{ij} is high. Specifically, they minimize the following objective function

$$\phi(Y) = \frac{1}{2} \sum_{i,j} |Y_i - Y_j|^2 W_{ij}$$
$$= tr(Y^T L Y),$$

where L is the Laplacian of graph G. The objective function is subjected to the constraint $Y^T D Y = I$ to eliminate trivial solution. The solution to this can be obtained by taking the eigenvectors corresponding to the d smallest eigenvalues of the normalized Laplacian, $L_{norm} = D^{-1/2} L D^{-1/2}$.

Main idea:
"High weights must be close"

LLE

- Locally linear embedding (published: 2000)

LLE [26] assumes that every node is a linear combination of its neighbors in the embedding space. If we assume that the adjacency matrix element W_{ij} of graph G represents the weight of node j in the representation of node i, we define

$$Y_i \approx \sum_j W_{ij} Y_j \qquad \forall i \in V.$$

Hence, we can obtain the embedding $Y^{N \times d}$ by minimizing

$$\phi(Y) = \sum_i |Y_i - \sum_j W_{ij} Y_j|^2,$$

To remove degenerate solutions, the variance of the embedding is constrained as $\frac{1}{N} Y^T Y = I$. To further remove translational invariance, the embedding is centered around zero: $\sum_i Y_i = 0$. The above constrained optimization problem can be reduced to an eigenvalue problem, whose solution is to take the bottom $d + 1$ eigenvectors of the sparse matrix $(I - W)^T (I - W)$ and discarding the eigenvector corresponding to the smallest eigenvalue.

GRAPH FACTORIZATION

- (published: 2013)

To the best of our knowledge, Graph Factorization [21] was the first method to obtain a graph embedding in $O(|E|)$ time. To obtain the embedding, GF factorizes the adjacency matrix of the graph, minimizing the following loss function

$$\phi(Y, \lambda) = \frac{1}{2} \sum_{(i,j)\in E} (W_{ij} - < Y_i, Y_j >)^2 + \frac{\lambda}{2} \sum_i \|Y_i\|^2,$$

where λ is a regularization coefficient. Note that the summation is over the observed edges as opposed to all possible edges. This is an approximation in the interest of scalability, and as such it may introduce noise in the solution. Note that as the adjacency matrix is often not positive semidefinite, the minimum of the loss function is greater than 0 even if the dimensionality of embedding is $|V|$.

Simple main idea:
Minimize difference between
Weight and cosine similarity

DEEP-LEARNING BASED

SDNE

- Intuitive definition: a "deep neural network" "autoencoder" learns embedding in order to minimize 2 objectives:
  - Nodes with similar neighbors should be close
  - Connected nodes should be close

GENERIC METHOD

VERSE

- Input: a (normalized) matrix of "similarity" between nodes (adjacency, #common nb, personalized PageRank, …)
- Function to minimize: $\sum_{v \in V} \mathrm{KL}\left(sim_{\mathrm{G}}(v, \cdot) \,||\, sim_{\mathrm{E}}(v, \cdot)\right)$
- Kullback-Leibler divergence between the original distribution of similarities (of v to other nodes) and the reconstructed cosine distance between v and other nodes.

SOME REMARKS ON WHAT ARE EMBEDDINGS

ADJACENCY MATRIX

- An adjacency matrix is an embedding… (in high dimension)
- That captures… the structural equivalence
  - 2 nodes have similar "embeddings" if they have similar neighborhoods
- Traditional dimensionality reduction of this matrix can be meaningful

GRAPH LAYOUT

- Graph layouts are also embeddings !
  - Force layout, kamada-kawai ….
- They try to put connected nodes close to each other and non-connected ones "not close"
- Problem: they try to avoid overlaps
- Usually not scalable

VISUALLY ?

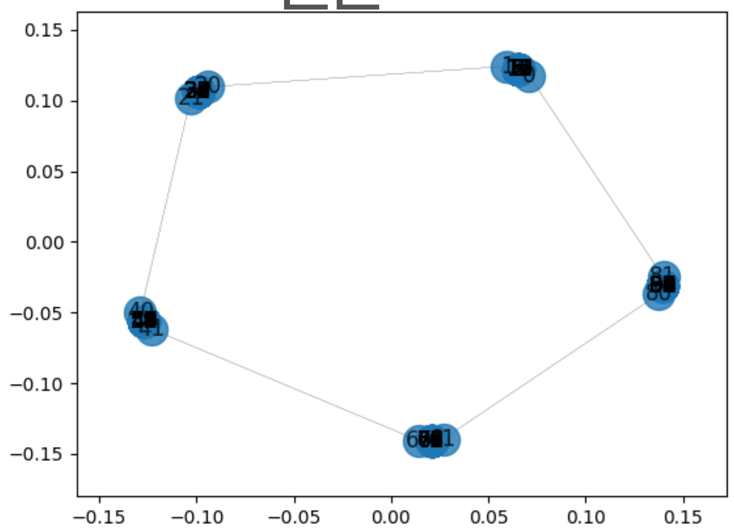
VISUALIZATION

- Be careful, embedding in 2 dimensions
- Usually: embedding in 128 dimensions
- Just to give intuitive idea

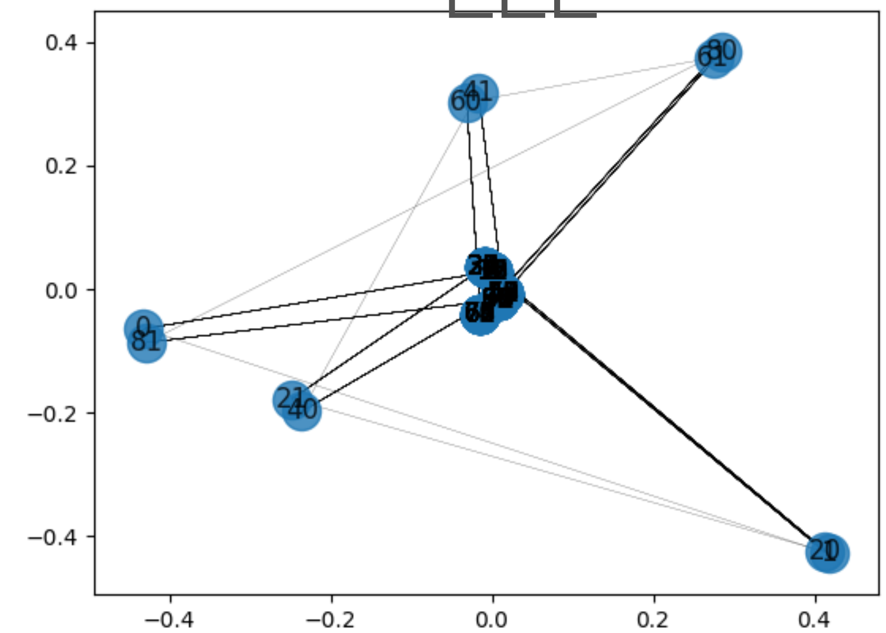
CLIQUE RING

5 cliques or size 20 with 1 edge between them

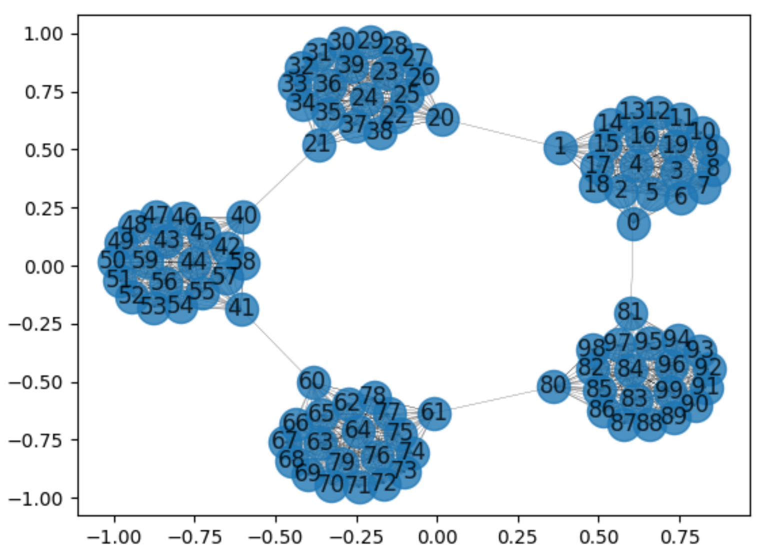
LE



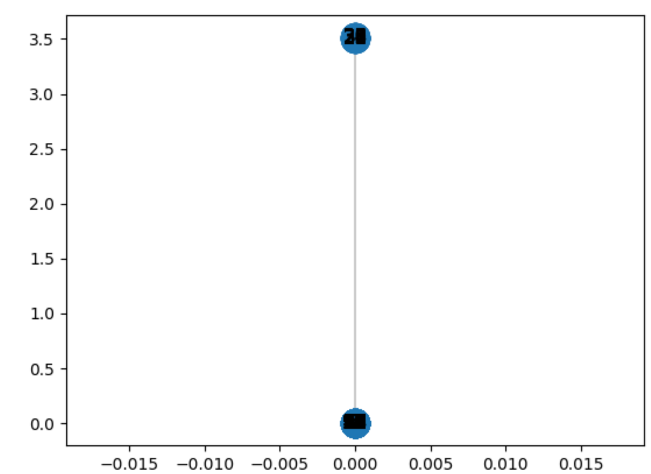
LLE



Spring layout



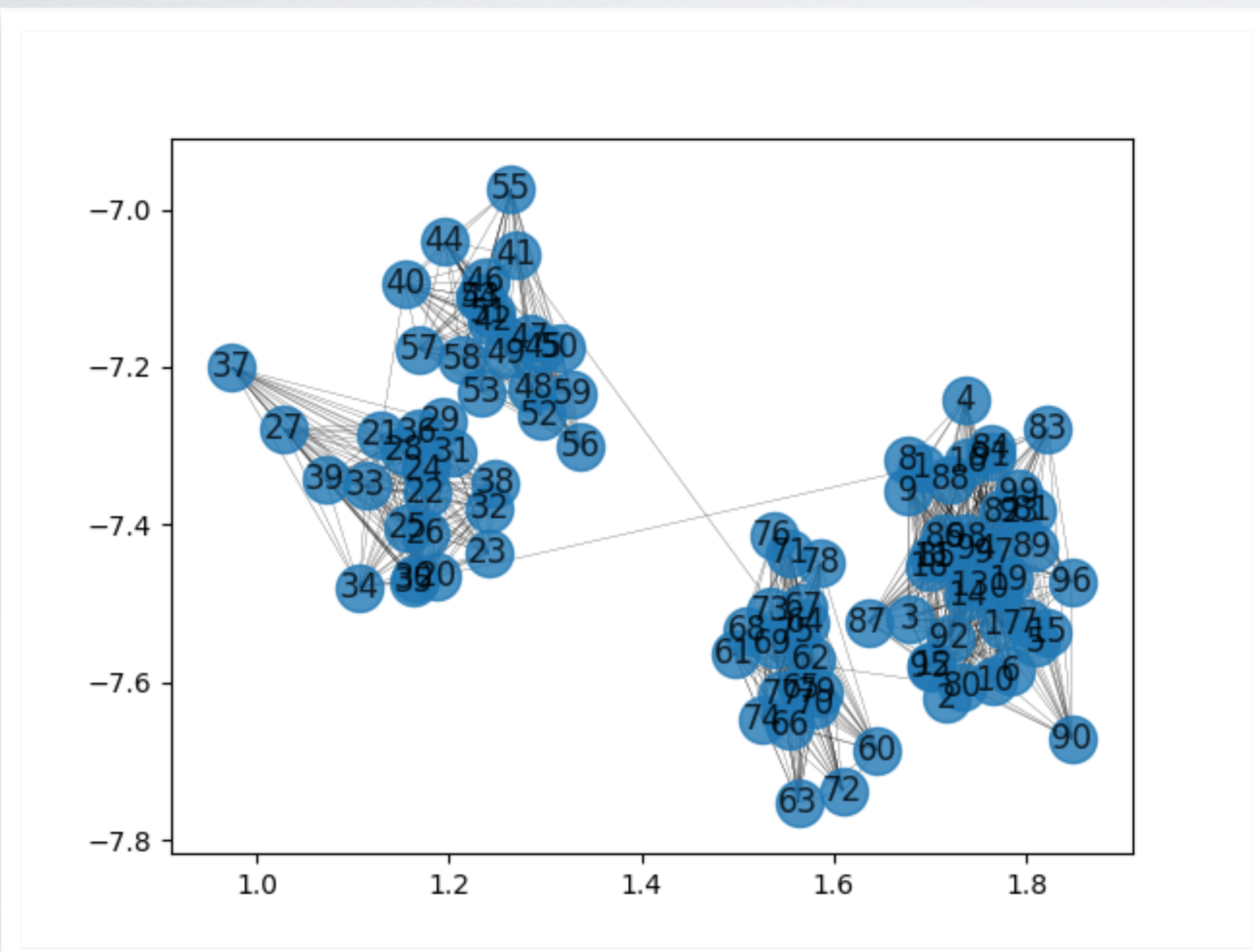
SDNE

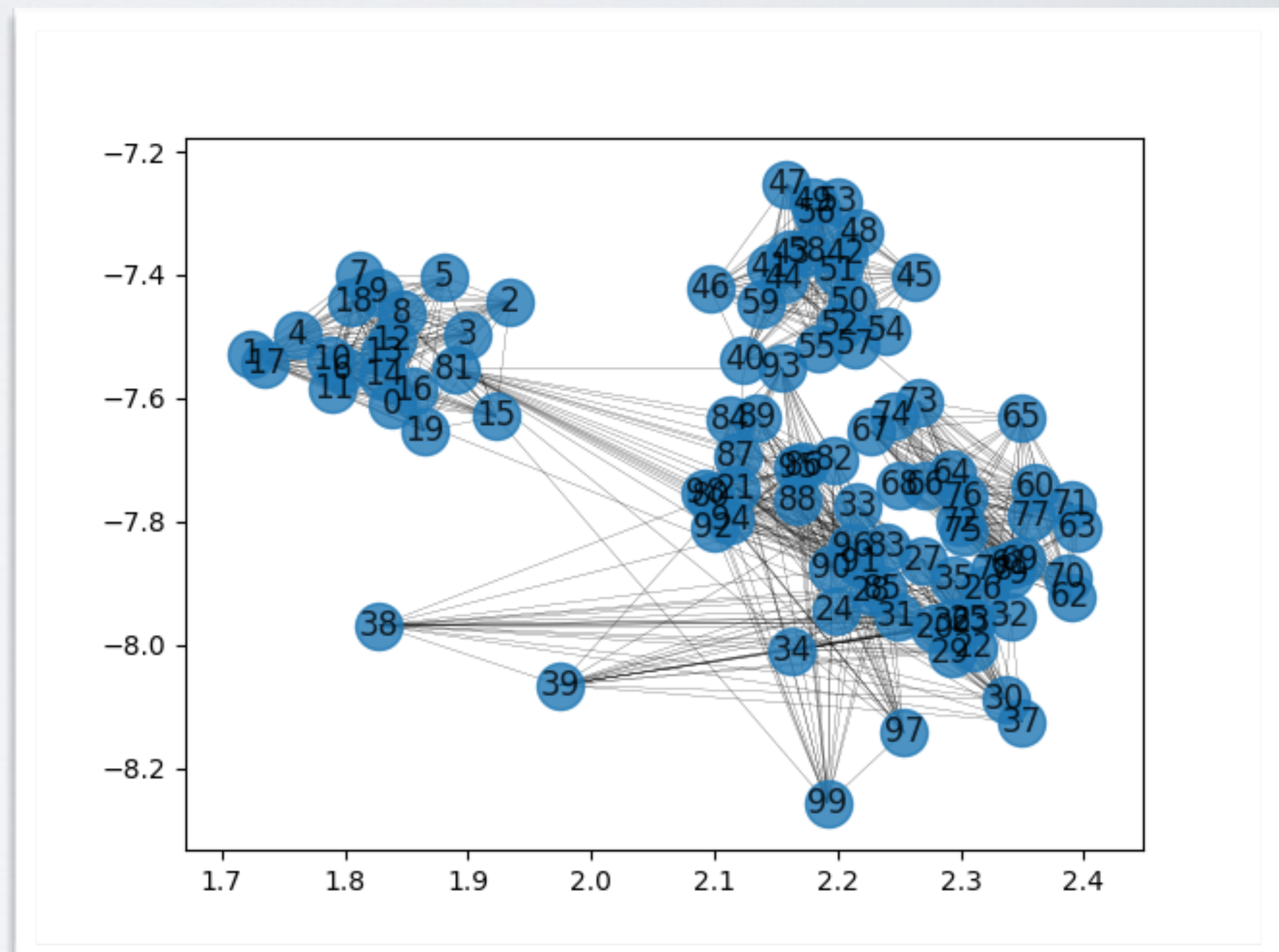


CLIQUE RING

5 cliques or size 20 with 1 edge between them

n2v



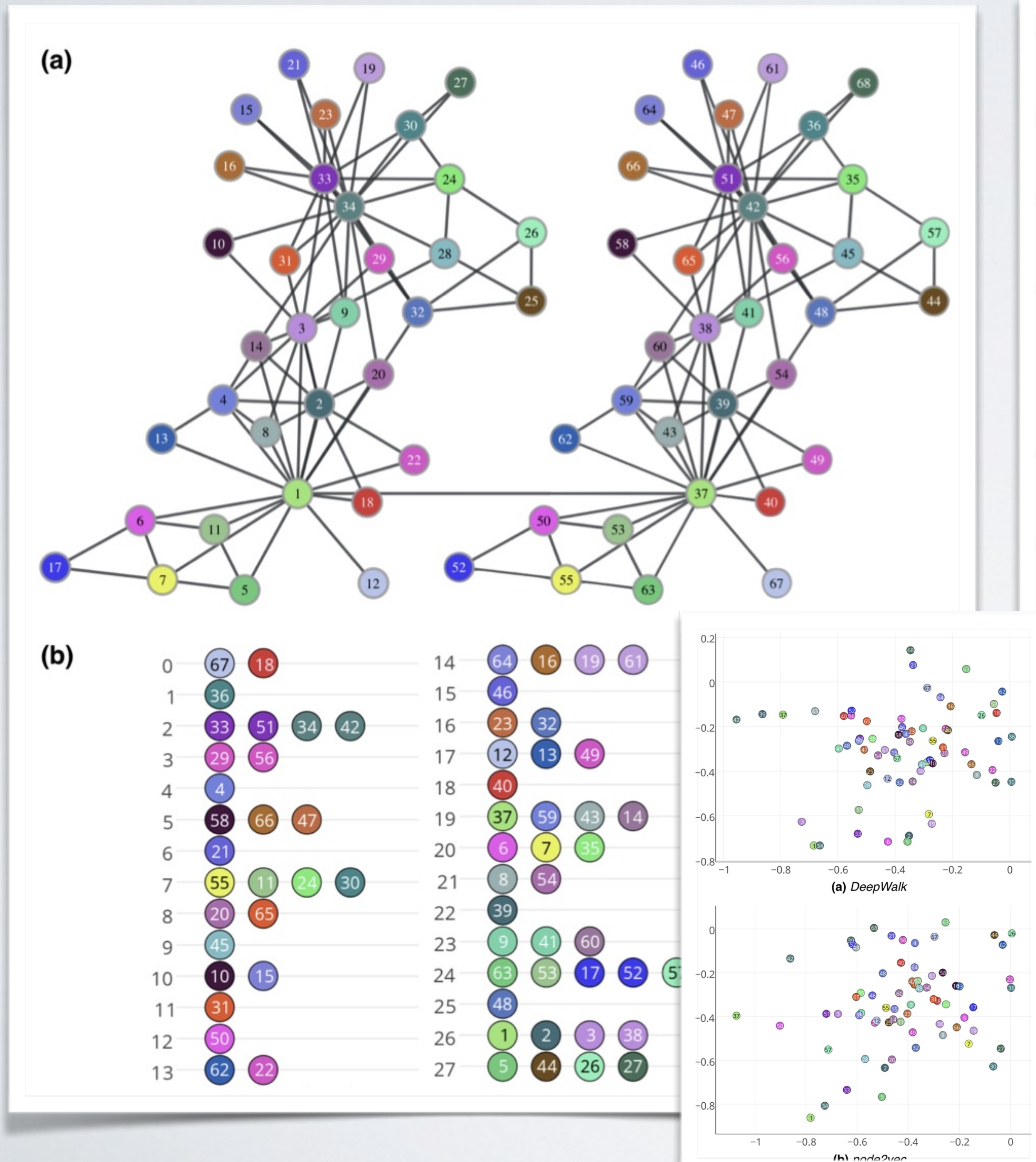


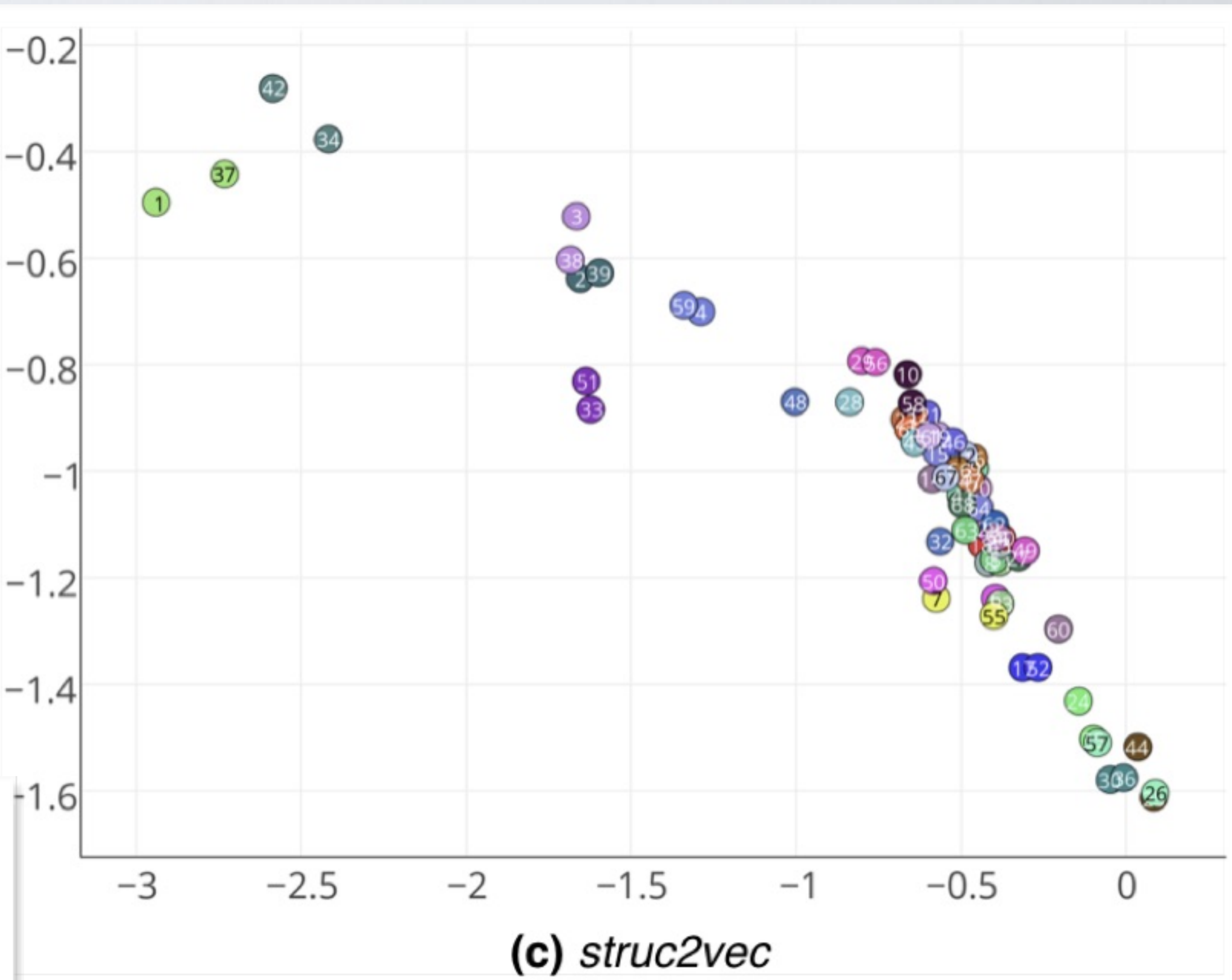
EMBEDDING ROLES

STRUCT2VEC

- In node2vec/Deepwalk, the context collected by RW contain the **labels** of encountered nodes
- Instead, we could memorize the **properties** of the nodes: attributes if available, or computed attributes (degrees, CC, …)
- =>Nodes with a same context will be nodes in a same "position" in the graph
- =>Capture the role of nodes instead of proximity

STRUCT2VEC : DOUBLE ZKC





NOTION OF DISTANCE IN EMBEDDINGS

DISTANCE IN EMBEDDINGS

- In embeddings, each node has an associated vector
- We can compute the distance between vectors
  - Euclidean distance (L2 norm)
  - Manhattan distance (L1 norm)
  - Cosine distance
  - Dot product
- Does this distance means something?
  - What does it means?

DISTANCE IN EMBEDDINGS

- What the distance means is often determined by what cost function the embedding tries to minimize
- => LE: $$\phi(Y) = \frac{1}{2} \sum_{i,j} |Y_i - Y_j|^2 W_{ij}$$
- =>node2vec/DeepWalk: Probability to reach after random walk of distance k

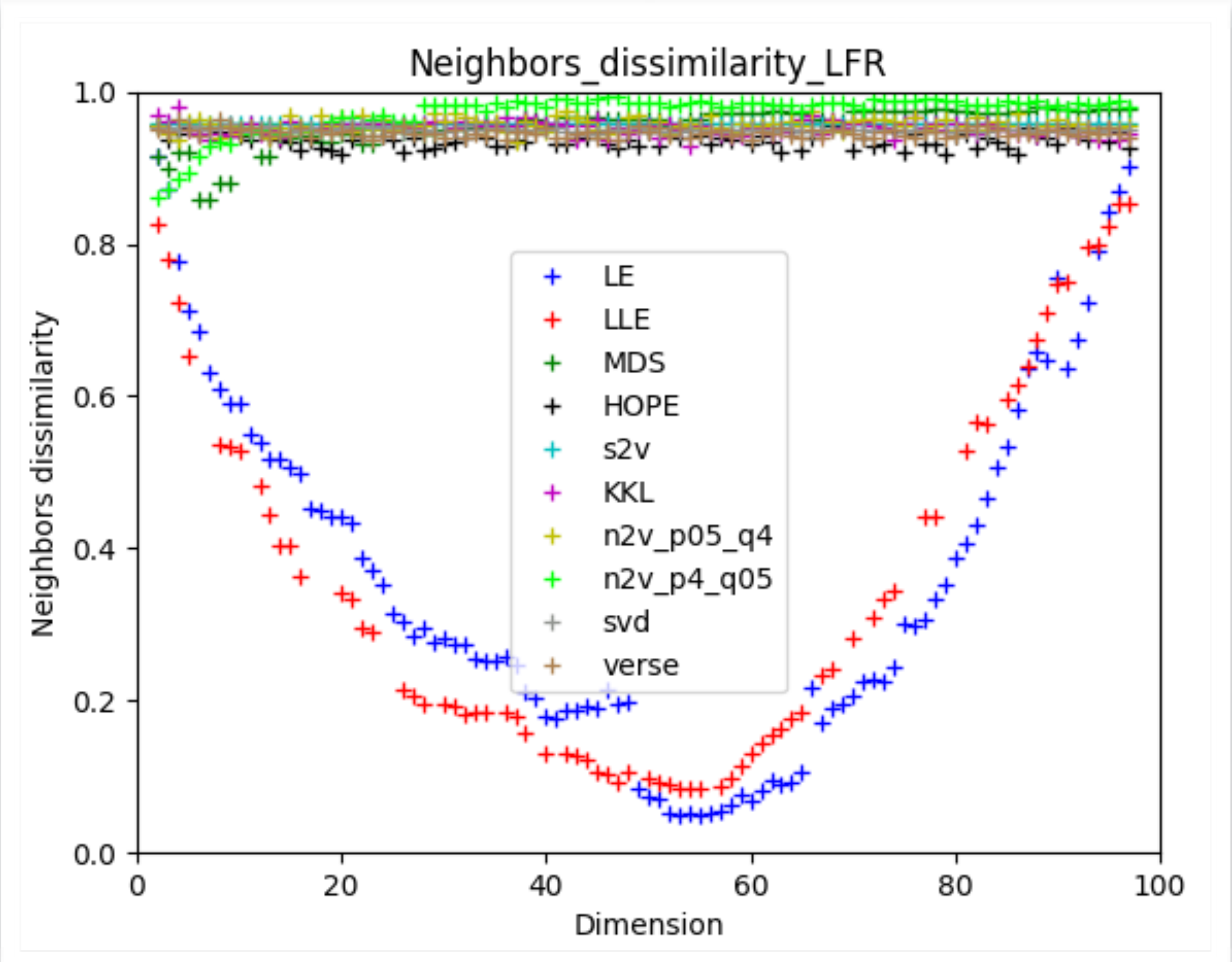
DISTANCE IN EMBEDDINGS

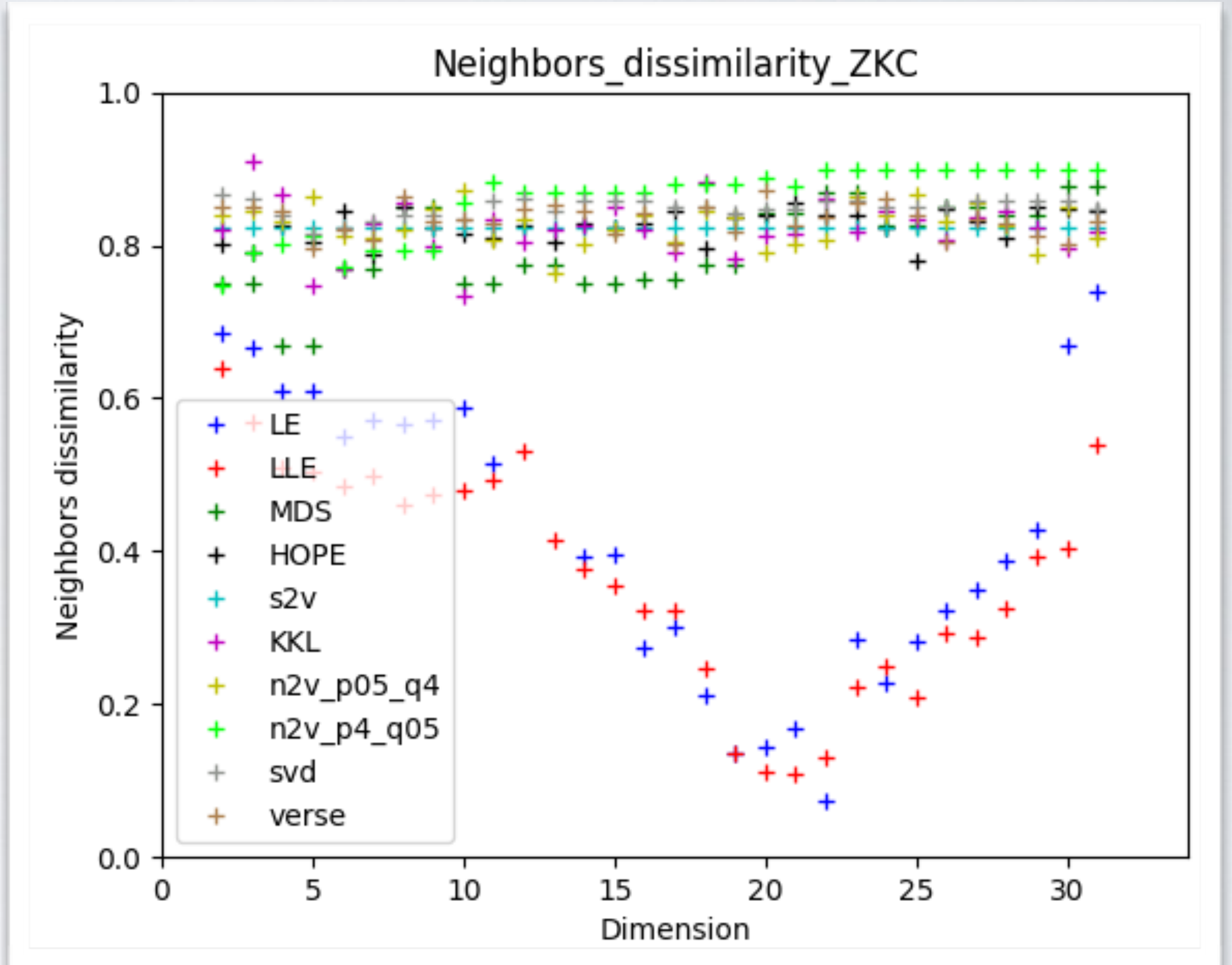
- Several possibilities:
  - Distance preserves the **probability of having an edge**
    - We can reconstruct the network from distances
  - Distance preserves the **similarity of neighborhood**
    - Called Structural equivalence
  - Distance preserves the **role in the network**
    - Hard to define
  - Distance preserves the **community structure**
    - Or another type of mesoscopic organization?

DISTANCE IN EMBEDDINGS

- Distance <=> having an edge?
- For each node:
  - 1)Find the neighbors in the graph. Number of N is k
  - 2)Find the k closest nodes in the embedding
  - 3)Compute the fraction of nodes in common in 1) and 2)
- Compute the average over all nodes
- Dissimilarity = 1-simmilarity

DISTANCE IN EMBEDDINGS





Lowest is better

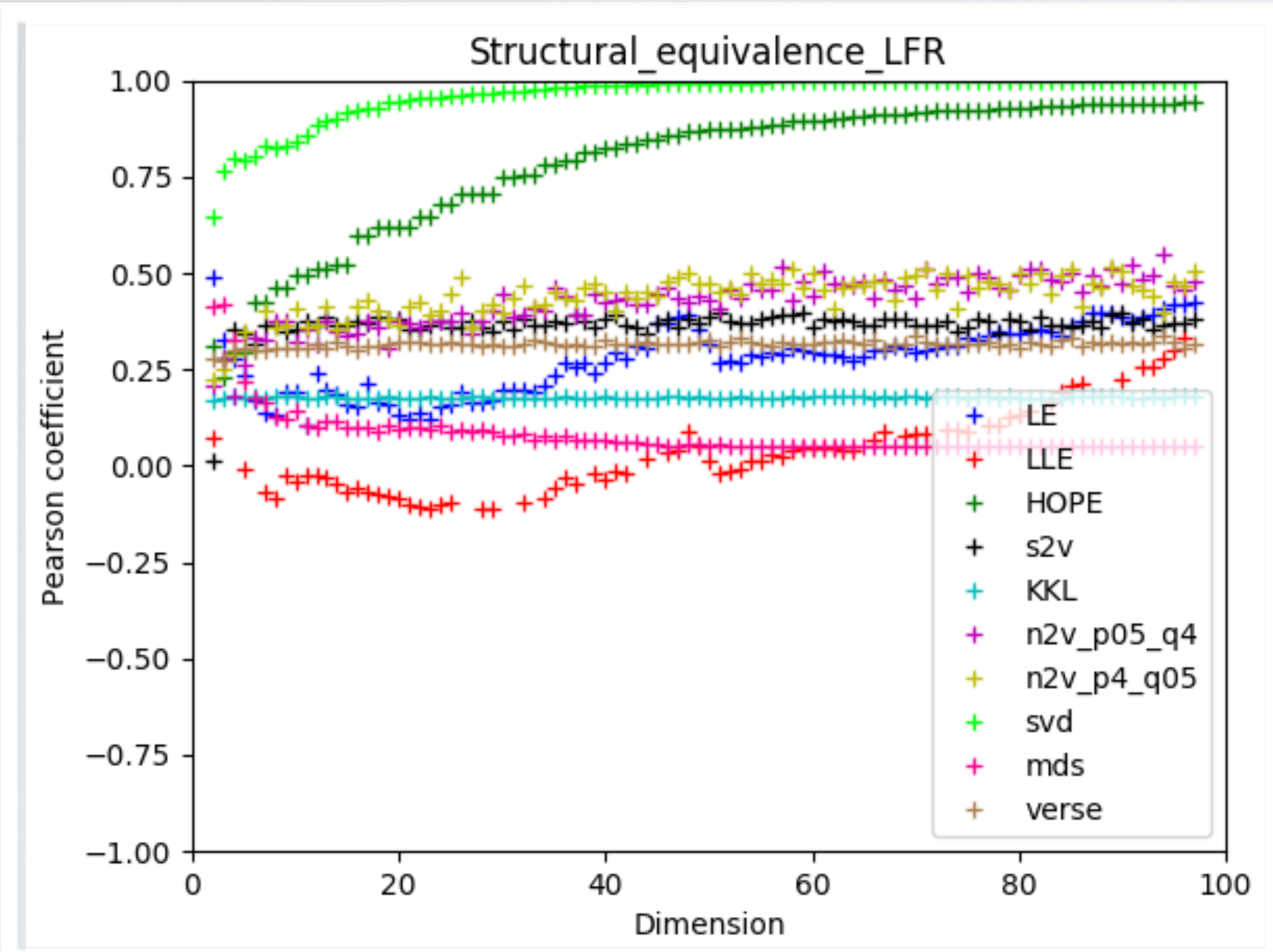
DISTANCE IN EMBEDDINGS

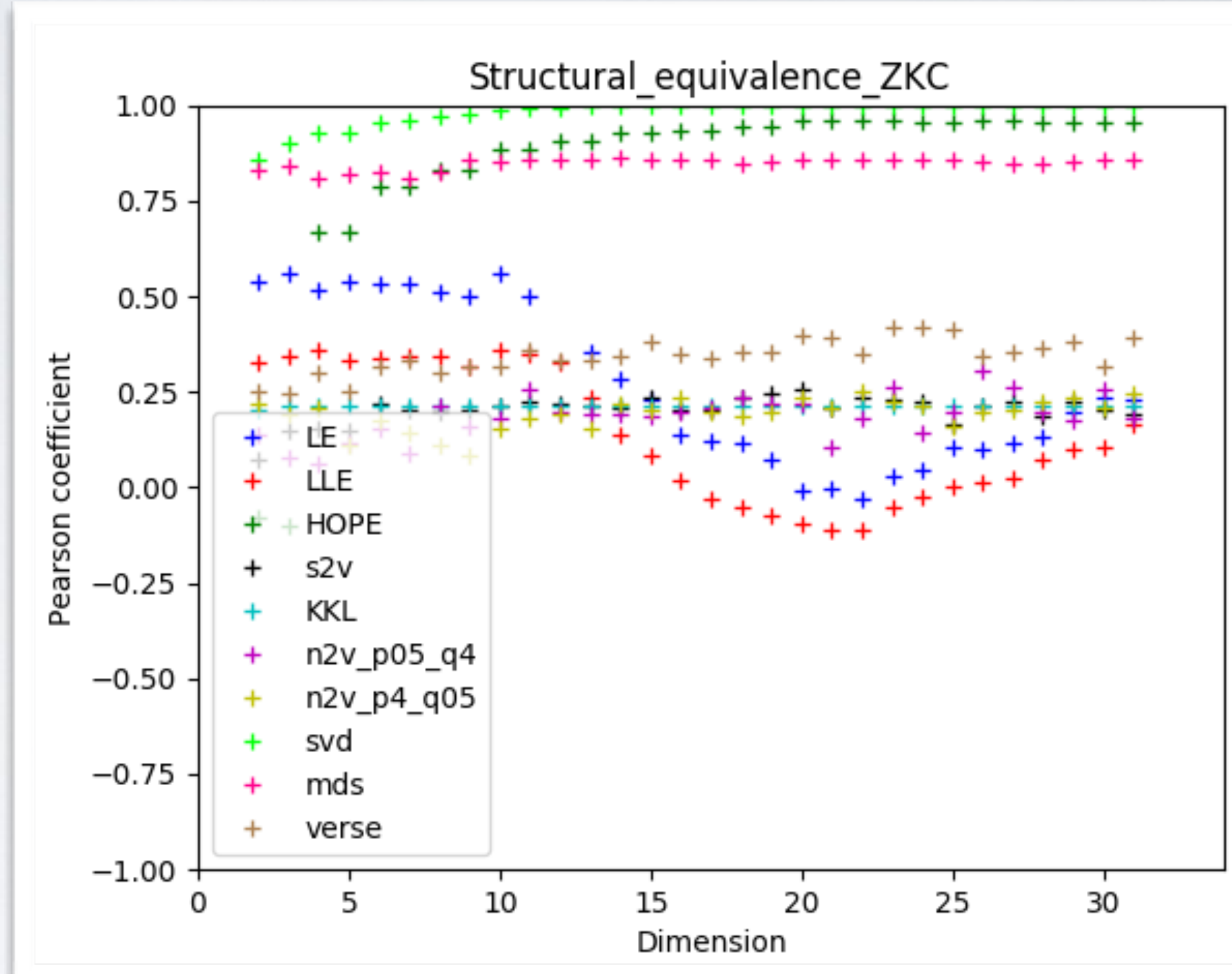
- Conclusion:
- Most algorithms do not preserve this property
- Some of them do it for some number of dimensions

STRUCTURAL EQUIVALENCE

- For each pair of nodes:
  - 1)Compute cosine distance between row of the adjacency matrix
    - Distance between neighborhoods
  - 2)Compute distance in the embedding
  - 3)Compute Correlation (Pearson) between both ordered sets of values
- =>How strongly both distances are correlated

STRUCTURAL EQUIVALENCE





STRUCTURAL EQUIVALENCE

- Conclusion:
- Many algorithms do not preserve this property
- Some algorithms do it
  - And in that case, the most dimensions, the better

COMMUNITY STRUCTURE

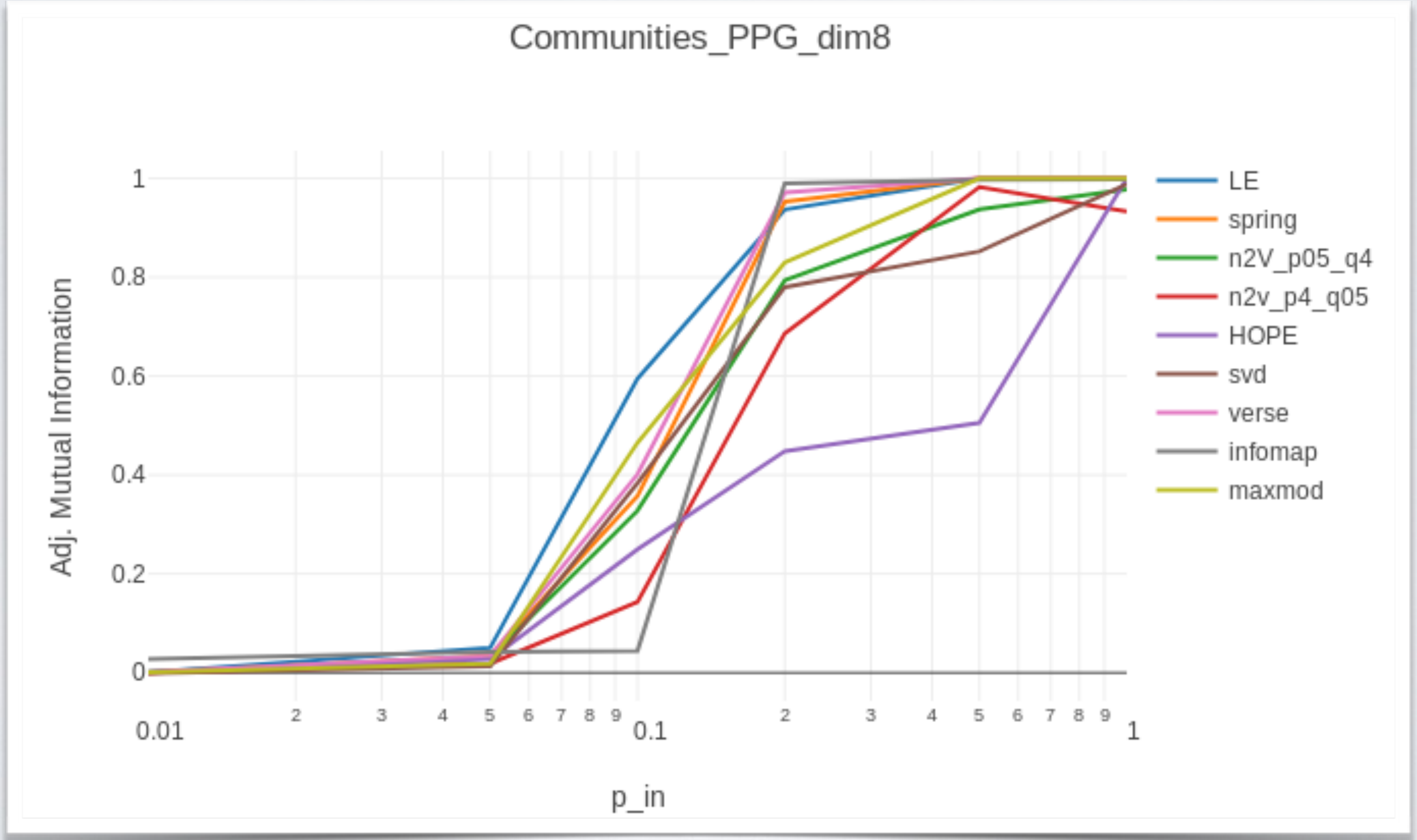
- Idea: if distance preserves community structure:
  - Nodes belonging to the same community should be close in the embedding
- We can use clustering algorithms (k-means…) to discover the communities

COMMUNITY STRUCTURE

- 1)Create a network with a community structure
- 2)Use k-means clustering on embedding to detect the community structure
- 3)Compare expected to k-means using the aNMI

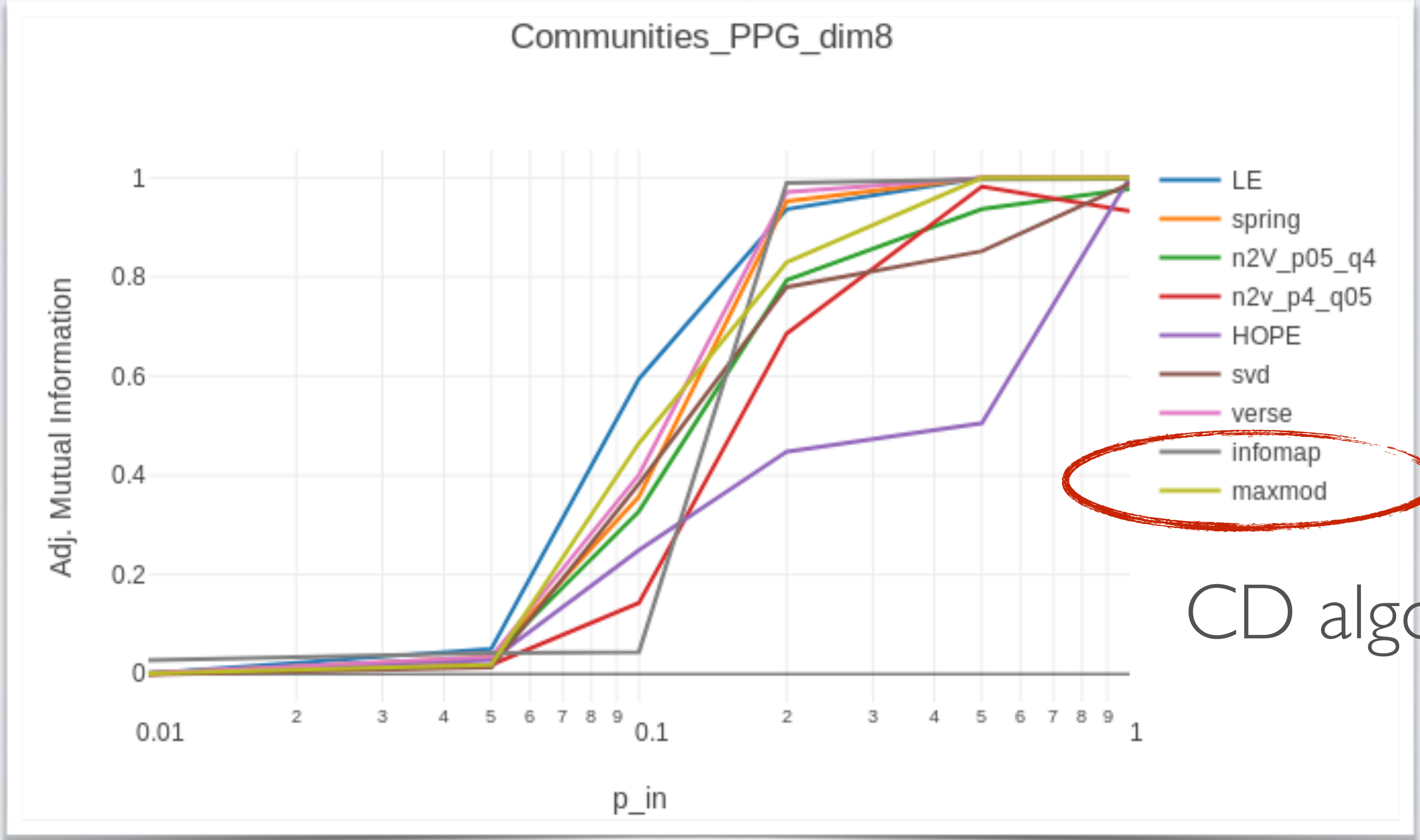
COMMUNITY STRUCTURE

Planted partitions. 8 dimensions



COMMUNITY STRUCTURE

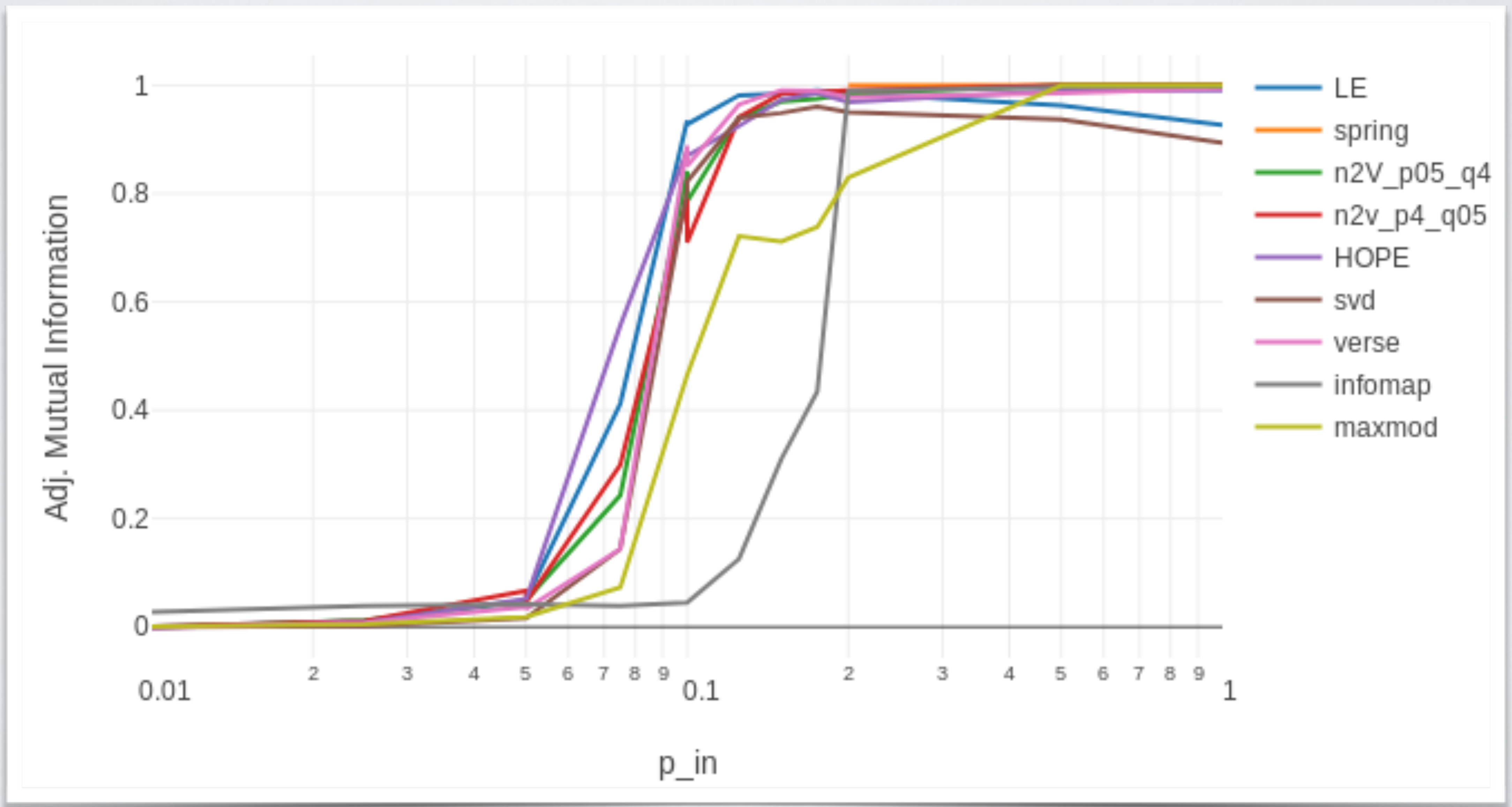
Planted partitions. 8 dimensions



CD algorithms

COMMUNITY STRUCTURE

Planted partitions. 128 dimensions



COMMUNITY STRUCTURE

- Conclusion (to be verified)
- If we know the number of clusters to find
- And we can use a large number of dimensions
- =>Embeddings can beat traditional algorithms

LINK PREDICTION WITH EMBEDDINGS

LINK PREDICTION

- Reminder:

- Unsupervised link prediction
  - Compute a score of similarity between pairs of nodes
  - =>Highest score: more probable links

- Supervised link prediction
  - Compute several features about pairs of nodes
  - Train a classifier to learn edges from features

LINK PREDICTION

- Unsupervised link prediction **from embeddings**
- =>Compute the distance between nodes in the embedding
- =>Use it as a similarity score

LINK PREDICTION

- Supervised link prediction **from embeddings**
- =>embeddings provide features for nodes (nb features: dimensions)
  - Combine nodes features to obtain edge features
- =>Train a classifier to predict edges based on features from the embedding

LINK PREDICTION

Operator	Result
Average	$(\mathbf{a} + \mathbf{b})/2$
Concat	$[\mathbf{a}_1, \ldots, \mathbf{a}_d, \mathbf{b}_1, \ldots, \mathbf{b}_d]$
Hadamard	$[\mathbf{a}_1 * \mathbf{b}_1, \ldots, \mathbf{a}_d * \mathbf{b}_d]$
Weighted L1	$[\lvert\mathbf{a}_1 - \mathbf{b}_1\rvert, \ldots, \lvert\mathbf{a}_d - \mathbf{b}_d\rvert]$
Weighted L2	$[(\mathbf{a}_1 - \mathbf{b}_1)^2, \ldots, (\mathbf{a}_d - \mathbf{b}_d)^2]$

Combining nodes vectors into edge vectors

LINK PREDICTION

- How well does it works ?

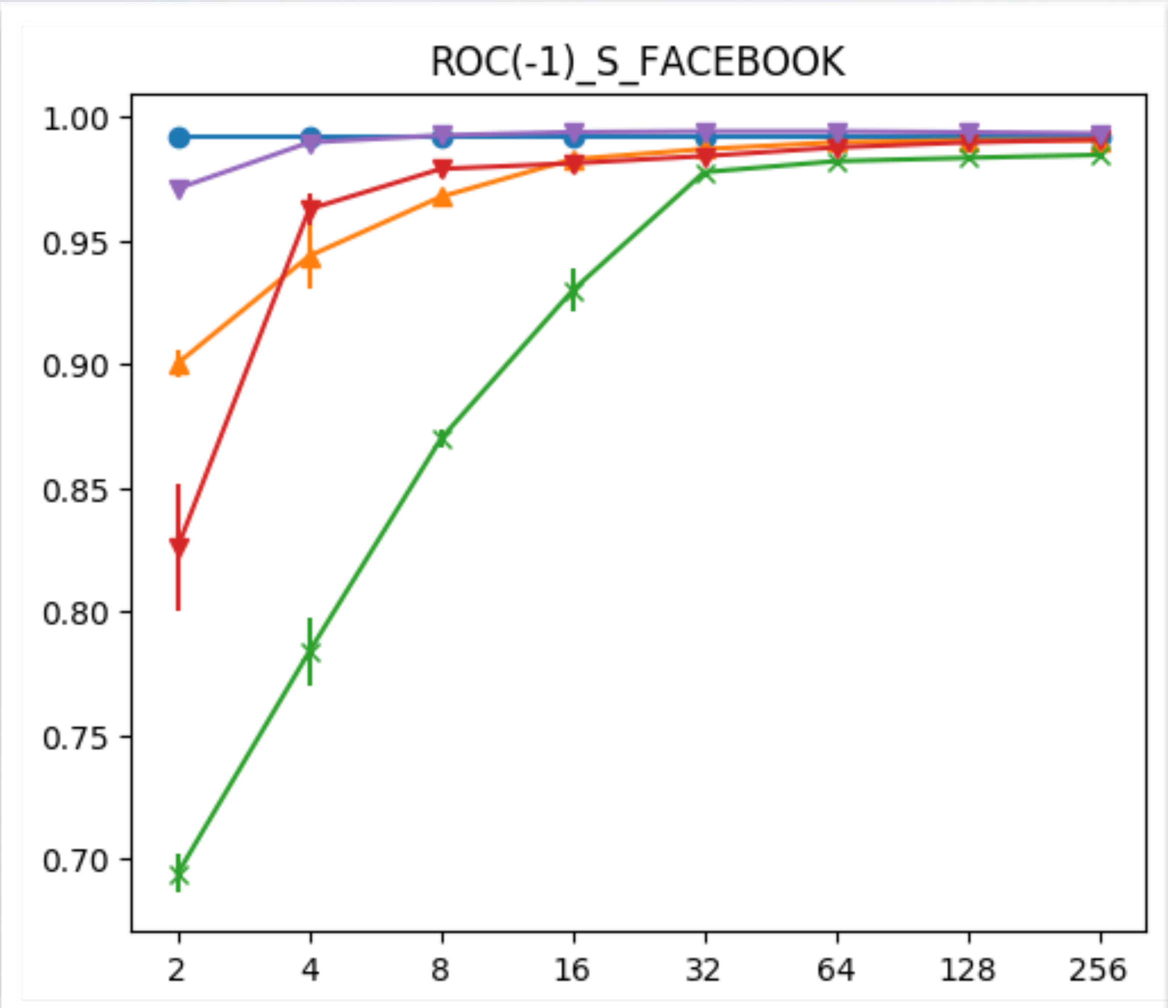
- According to recent articles
  - Node2vec (2016)
  - VERSE (2018)

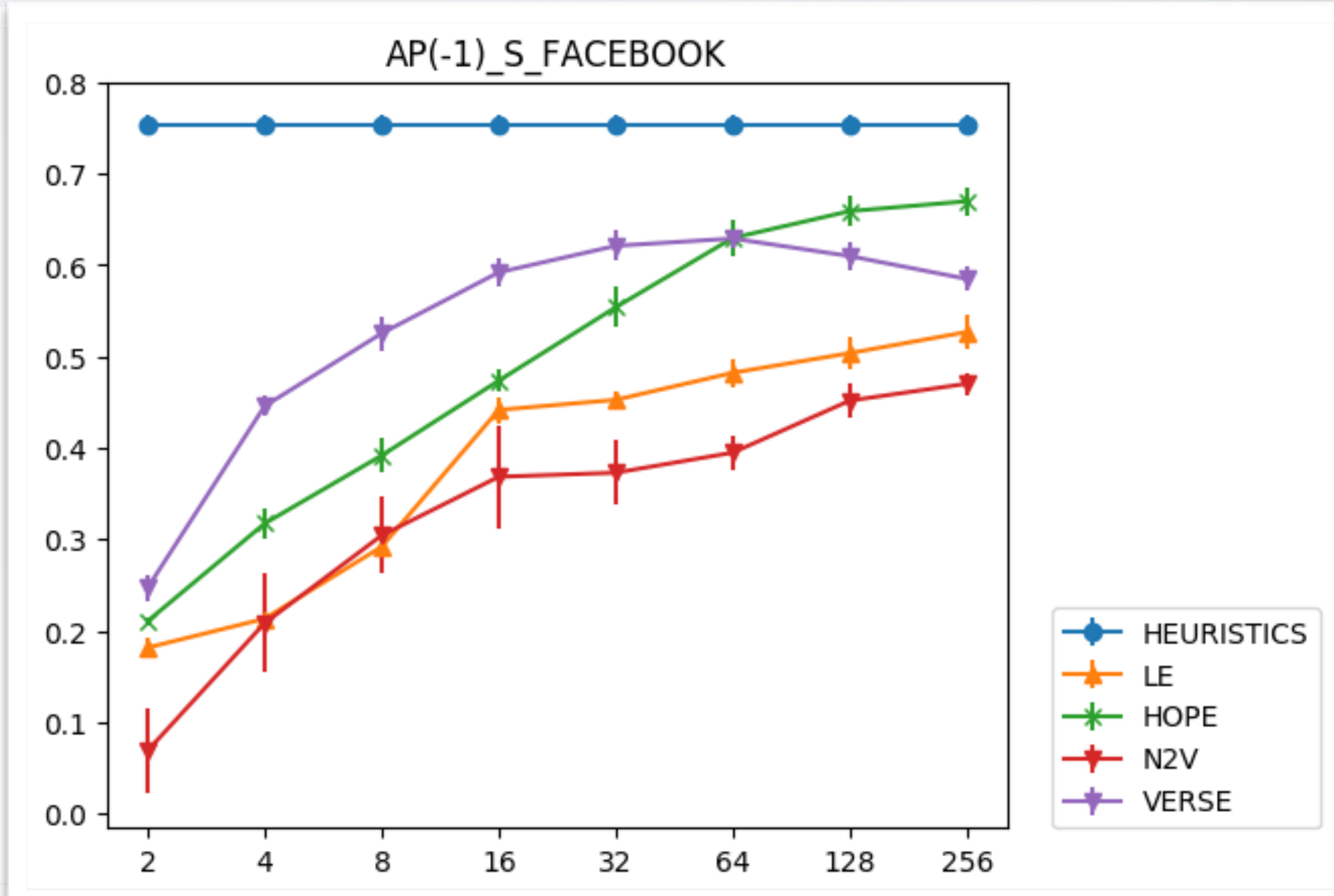
- =>These methods are better than state of the art

LINK PREDICTION

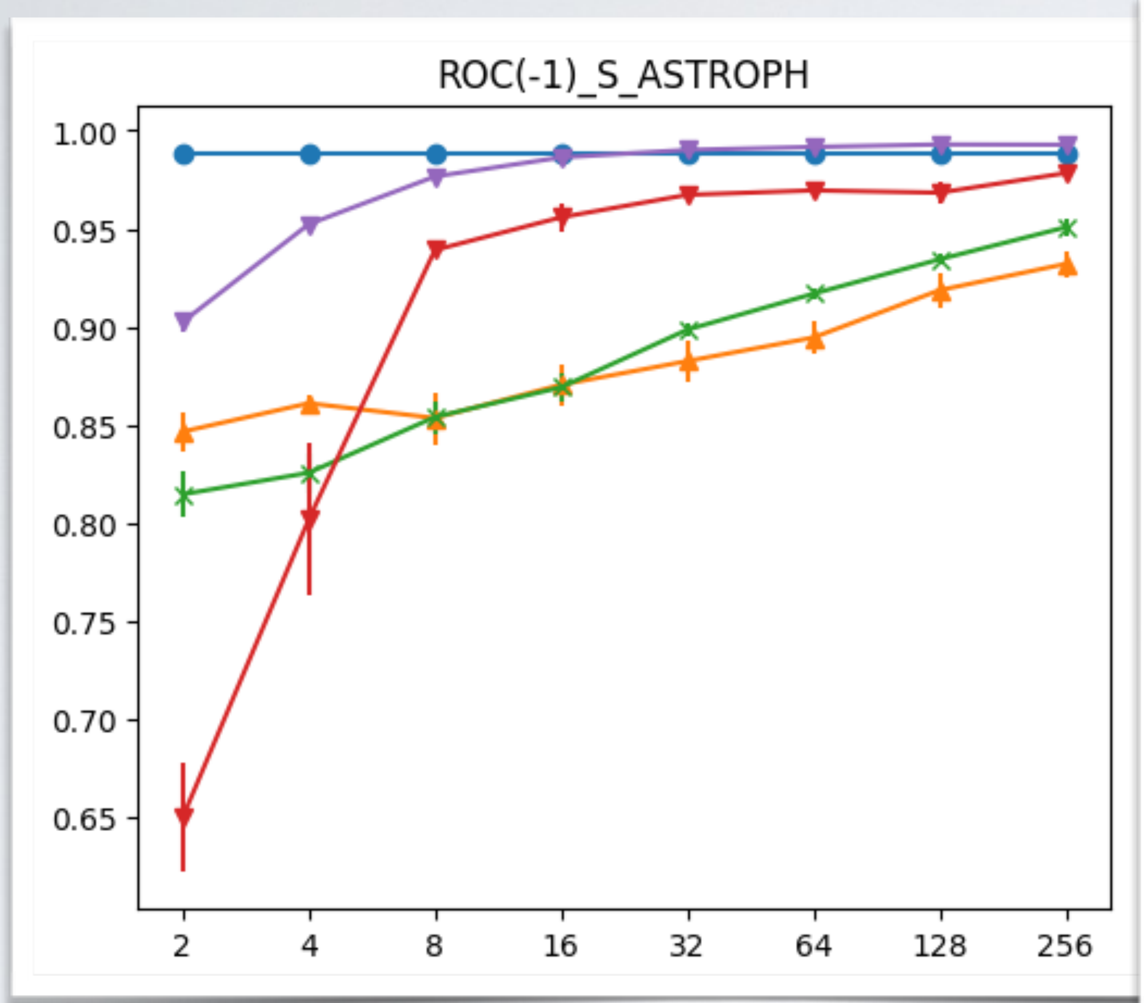
- Our tests: not really
- Embeddings are better only if we use some particular tests settings
  - Accuracy score on balanced test sets (WRONG)
  - Supervised LP for embeddings compared with unsupervised heuristics

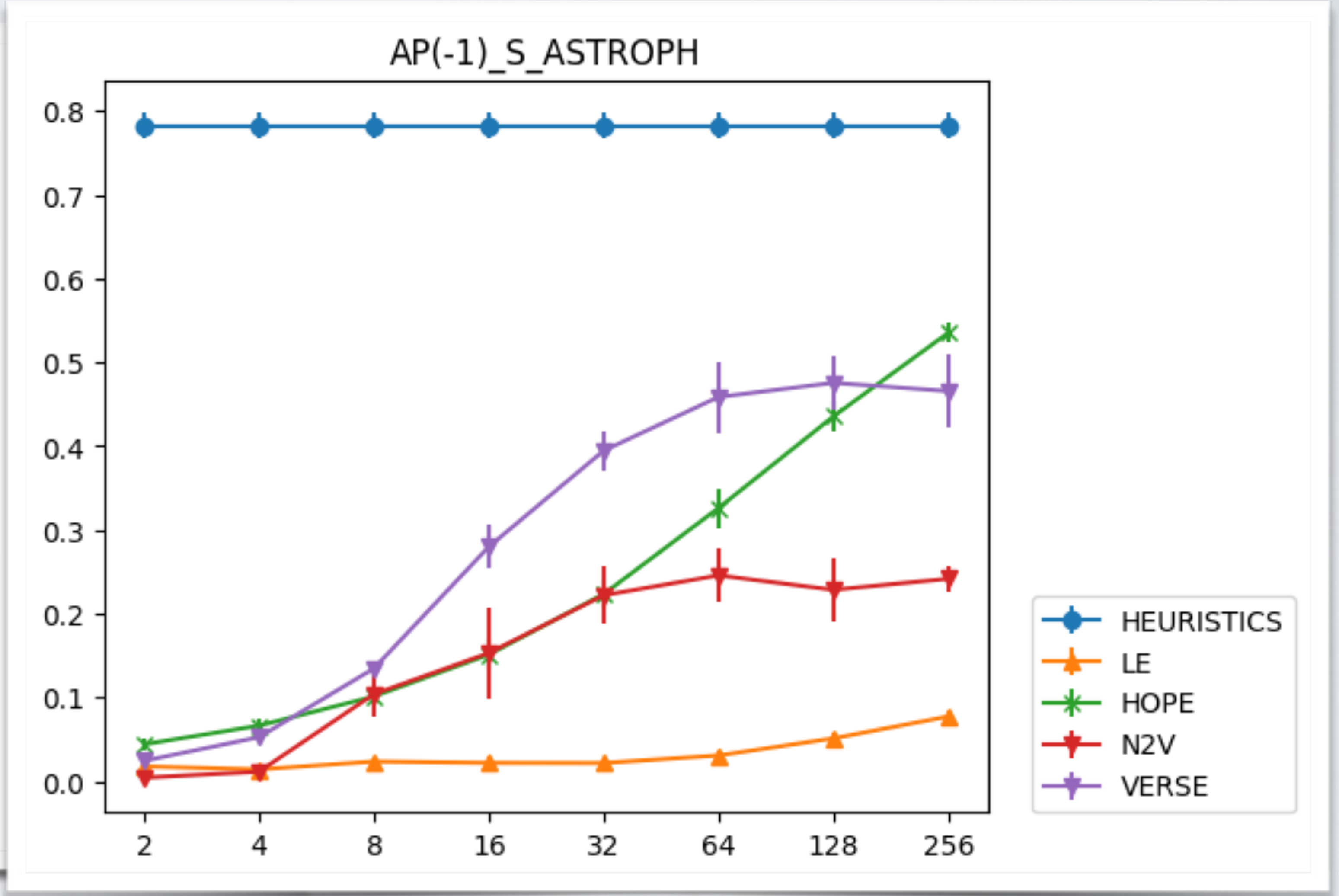
LINK PREDICTION





LINK PREDICTION





LINK PREDICTION

- So, embeddings not interesting?
  - =>Not at all!
- Different embeddings capture different things
  - Being directly linked
  - Having same neighbors
  - Community structure
  - Roles
- They all provides “features” we can use as input to classifiers
- =>Our current project: combine them to do better link prediction

PRACTICALS

- 1)Continue last week practicals (link prediction)
- 2)Compute embeddings using layouts from networkx
  - Function spring_layout (options: dimensions, iterations…)
- 3)Evaluate quality of link prediction using AUC/AP
  - Compare with other methods
- 4) (Advanced) Use "real" embeddings and compare the results
  - https://github.com/palash1992/GEM
  - https://github.com/xgfs/verse

PROJECT: EXAMPLE DATASETS

- Music:
  - LastFM: https://labrosa.ee.columbia.edu/millionsong/lastfm
  - Million Song dataset: https://labrosa.ee.columbia.edu/millionsong/
- Movies:
  - Internet Movie Database: https://www.imdb.com/interfaces/
  - https://www.kaggle.com/carolzhangdc/imdb-5000-movie-dataset
  - MovieLens: https://grouplens.org/datasets/movielens/
- Food:
  - Open food facts: https://world.openfoodfacts.org
- List of datasets:
  - https://www.kdnuggets.com/datasets/index.html
- List of list of datasets:
  - https://towardsdatascience.com/cool-data-sets-ive-found-adc17c5e55e1
- Colombia open data:
  - https://www.datos.gov.co/browse?sortBy=newest