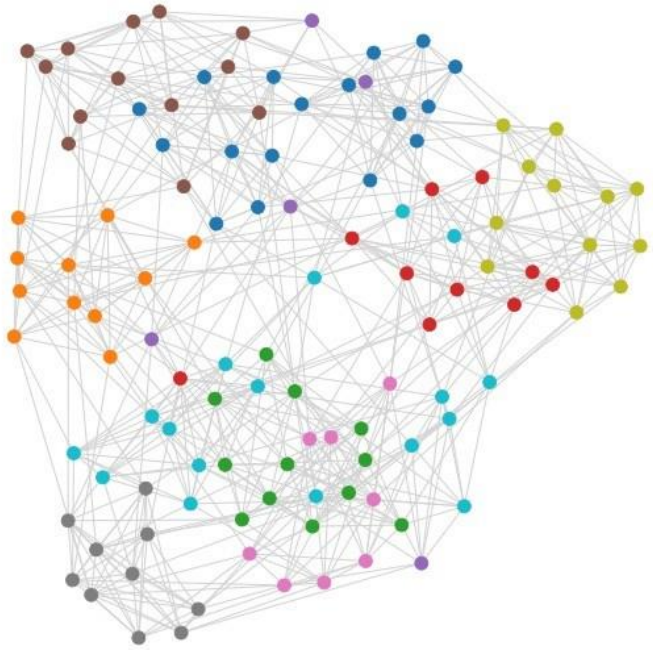
PART 2: Similarity based on paths

Assumption: Nodes connected by many paths of short distance are more likely to be connected

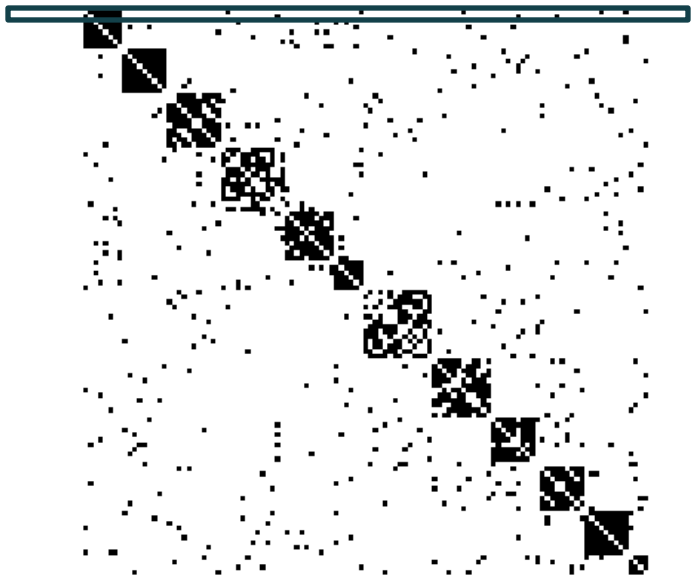
A: Presence of path between node *i* and *j*
A^2: Number of path between node *i* and *j* in two steps
A^3: Number of path between node *i* and *j* in three steps
…

Katz similarity: counts the number of walk at all distances between two nodes, giving shorter walks more weight

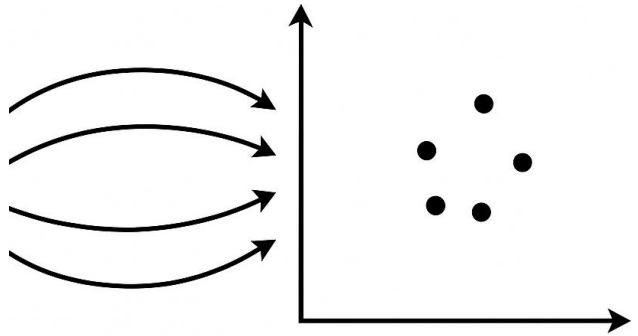
PART 3: Node embeddings



Adjacency matrix



Node embeddings



We can define each node by its connections

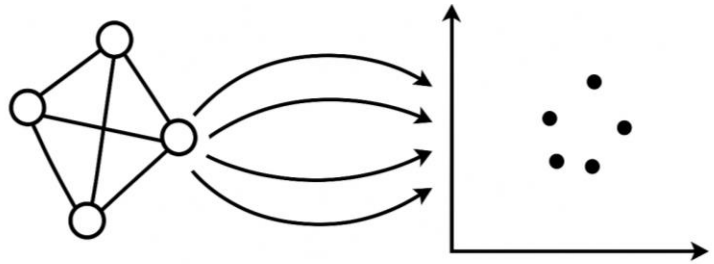
You can use it to predict something about the node:

- but that would mean thousands or millions of parameters!
- and it only provides information at the local level

Lower dimensionality representation

- That captures the network structure
- Where similar nodes should have similar embeddings

Node embeddings



What do we mean with similar nodes?

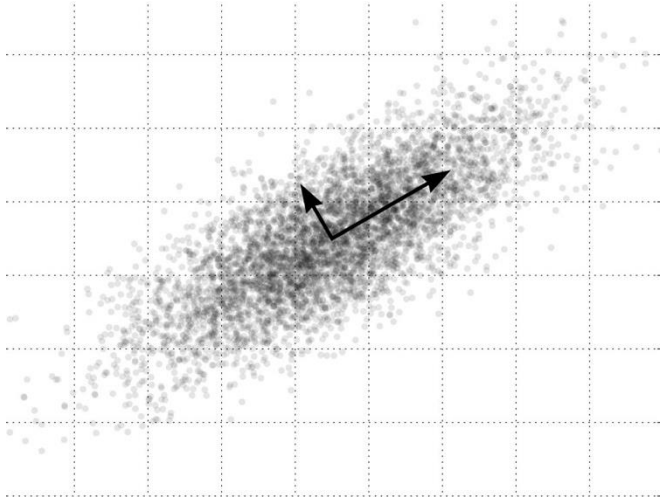
- Nodes with similar position in the network
  → **Unsupervised learning:** Spectral methods and Shallow neural networks

- Nodes with the same outcome (e.g. criminal nodes in a financial network)
  → **Node classification**
    X = Embedding
    Y = outcome to be predicted

- Nodes that are connected (for link prediction)
  → **Link prediction**
    X = combination of the two embeddings X_1 and X_2 (e.g. $X_1 * X_2$, or np.abs(X_1 - X_2))
    Y = link/no link

Part 3.1: Spectral methods

Related to characteristic eigenvectors of matrices associated with the network



Principal Component Analysis (PCA)

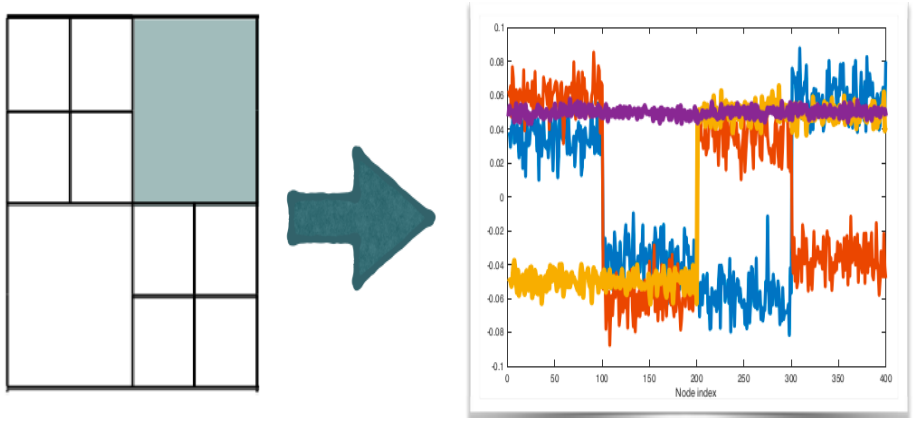
- Correlated variables → Linear combination of orthogonal variables
- Eigenvectors corresponding to the largest k eigenvalues of A^TA (centered)

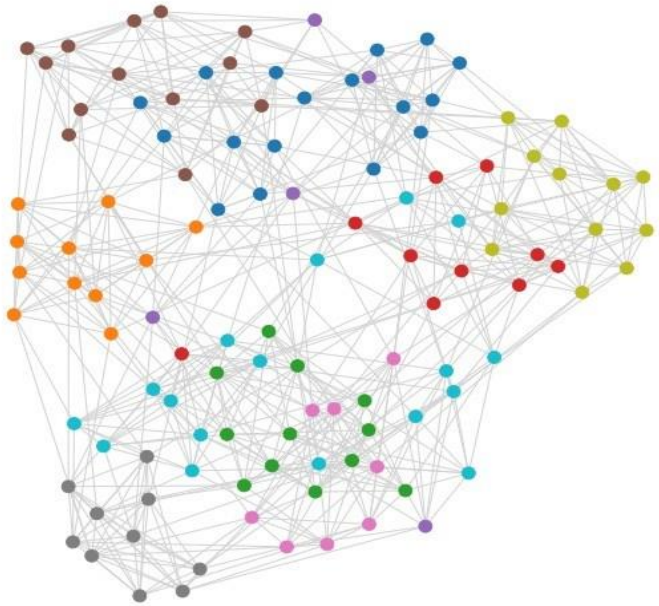
Singular Value Decomposition (SVD)

- Eigenvectors corresponding to the largest k eigenvalues of A^TA (uncentered)

Laplacian Eigenmaps (LE)

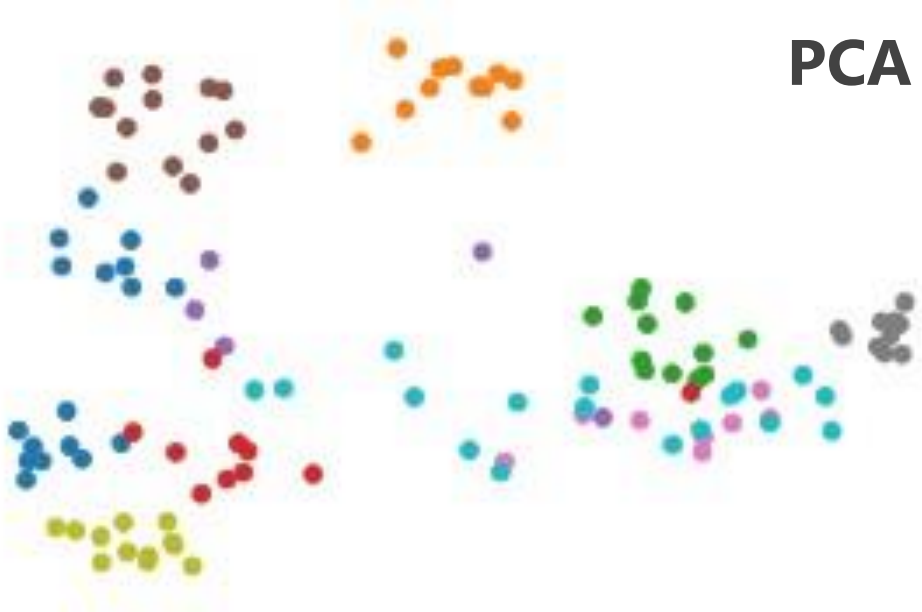
- Assumes that the nodes lie on a low-dimensional space
- Nodes close in that space are also close in the network
- That space = the eigenvectors corresponding to the smallest *k* eigenvectors of the normalized Laplacian matrix $D^{-1}(D-A)$





SVD

PCA

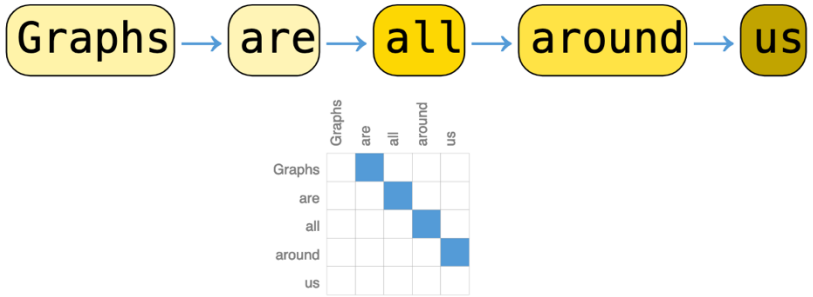


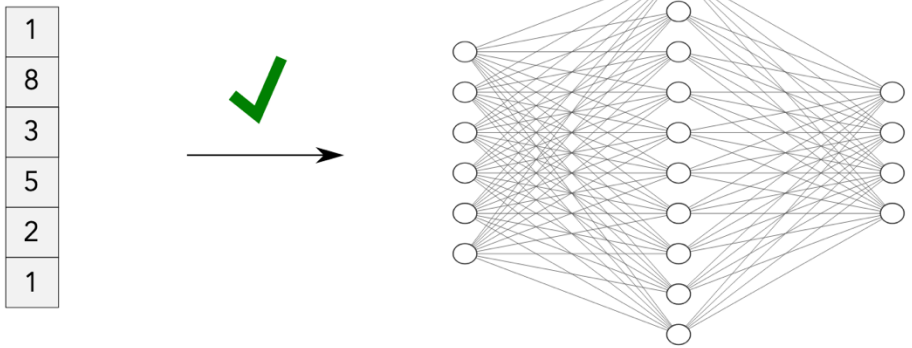
LE

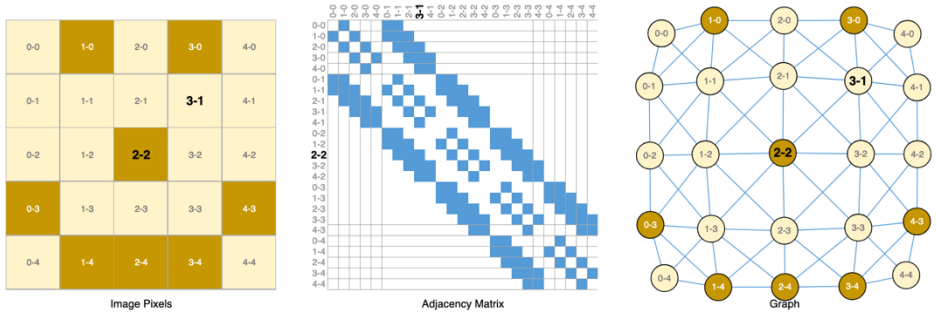
Part 3.2: Shallow Neural Networks

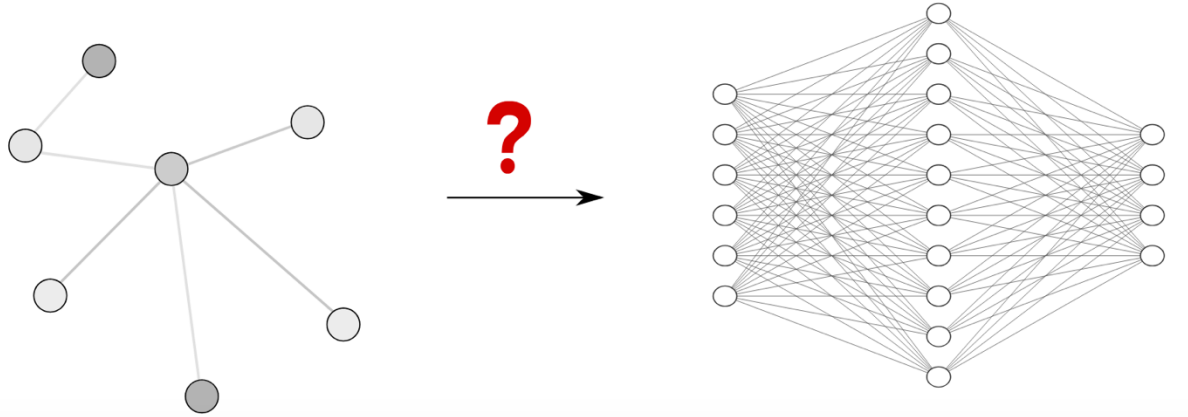
Regular networks: Text, images

- Text analysis: Chain (nodes = words)
- Images: Lattices (nodes = pixels)







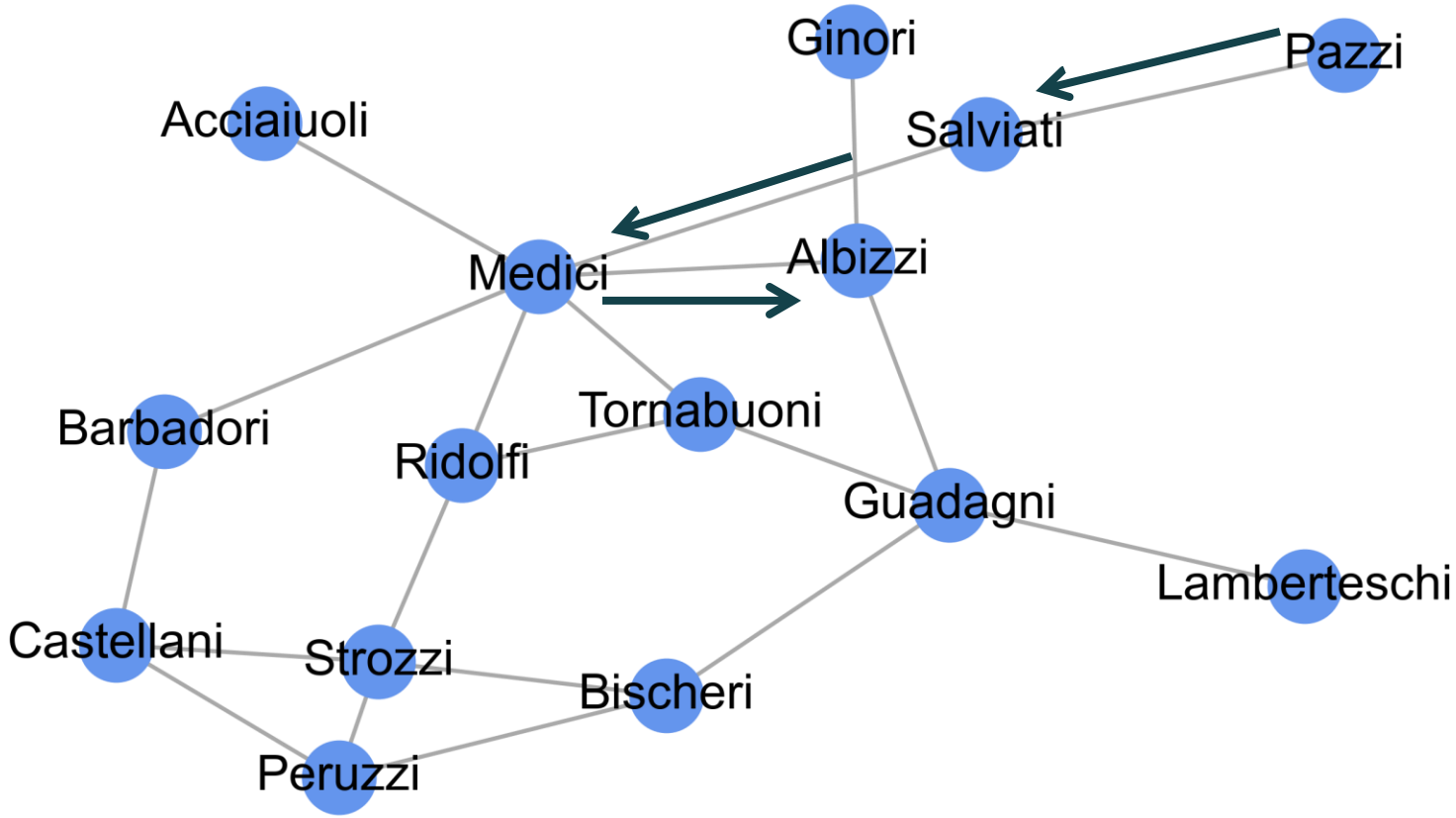


https://distill.pub/2021/understanding-gnns/

Node2vec / deepwalk

Idea:

1. Generate “sentences” using random walks.
2. Use methods from text analysis



Pazzi -> Salviati -> Medici -> Albizzi

…

Text analysis: Word2vec

Word2vec

Distributional hypothesis: similar words will be surrounded by similar words (you will know a word by the company it keeps)

What words tend to appear around “Network”?

- Network Science
- Network Analysis

→ Words *science* and *analysis* are similar

Text analysis: Word2vec

Step 1: Create co-occurrance matrix

- I like deep learning
- I like NLP
- I enjoy flying

Context word

Target word: counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

Word Sense Disambiguation Using Vector Space Models

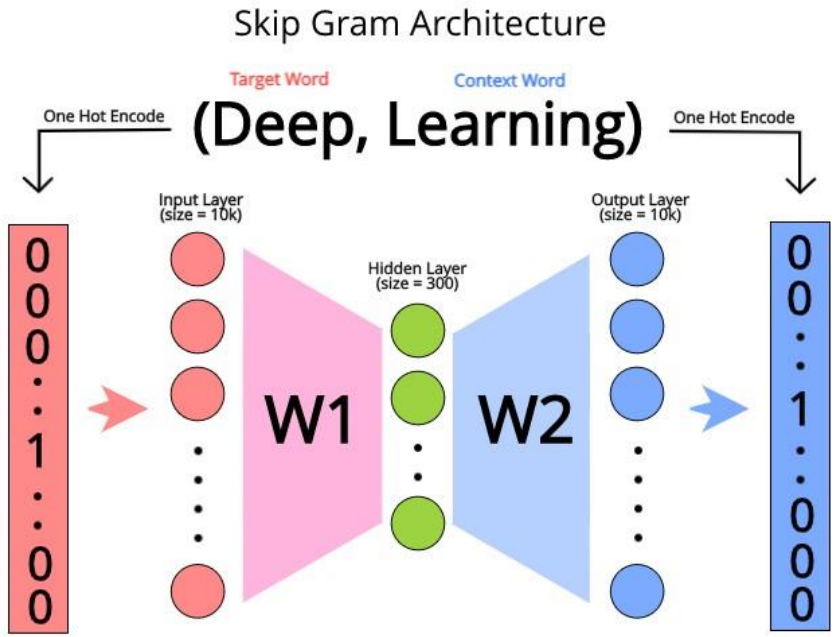
Text analysis: Word2vec

Step 2: Train a classification model

- *Positive examples*: target word should predict context words (Skip-Gram)
- *Negative examples*: target word should not predict random words (i.e., not context words)

Two vectors are similar if they have a high dot product (~cosine similarity)

- Vector associated to “deep”: w1[“deep”,:]
- Vector associated to “learning”: w2[:,”learning”]



Intuition:

- Modify W1 and W2 so target embeddings are close (have a high dot product) to context embeddings for nearby words and further from context embeddings for noise words that don’t occur nearby

Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition (Jurafsky and Dan, 2009)

In networks: DeepWalk / Node2Vec

Distributional hypothesis: The more often two nodes appear near the same nodes in the same random walk, the more similar their embeddings will be.

Step 1: Create co-occurrance matrix

Pazzi -> Salviati -> Medici -> Albizzi

Medici -> Albizzi -> Guadagni -> Medici

Context node

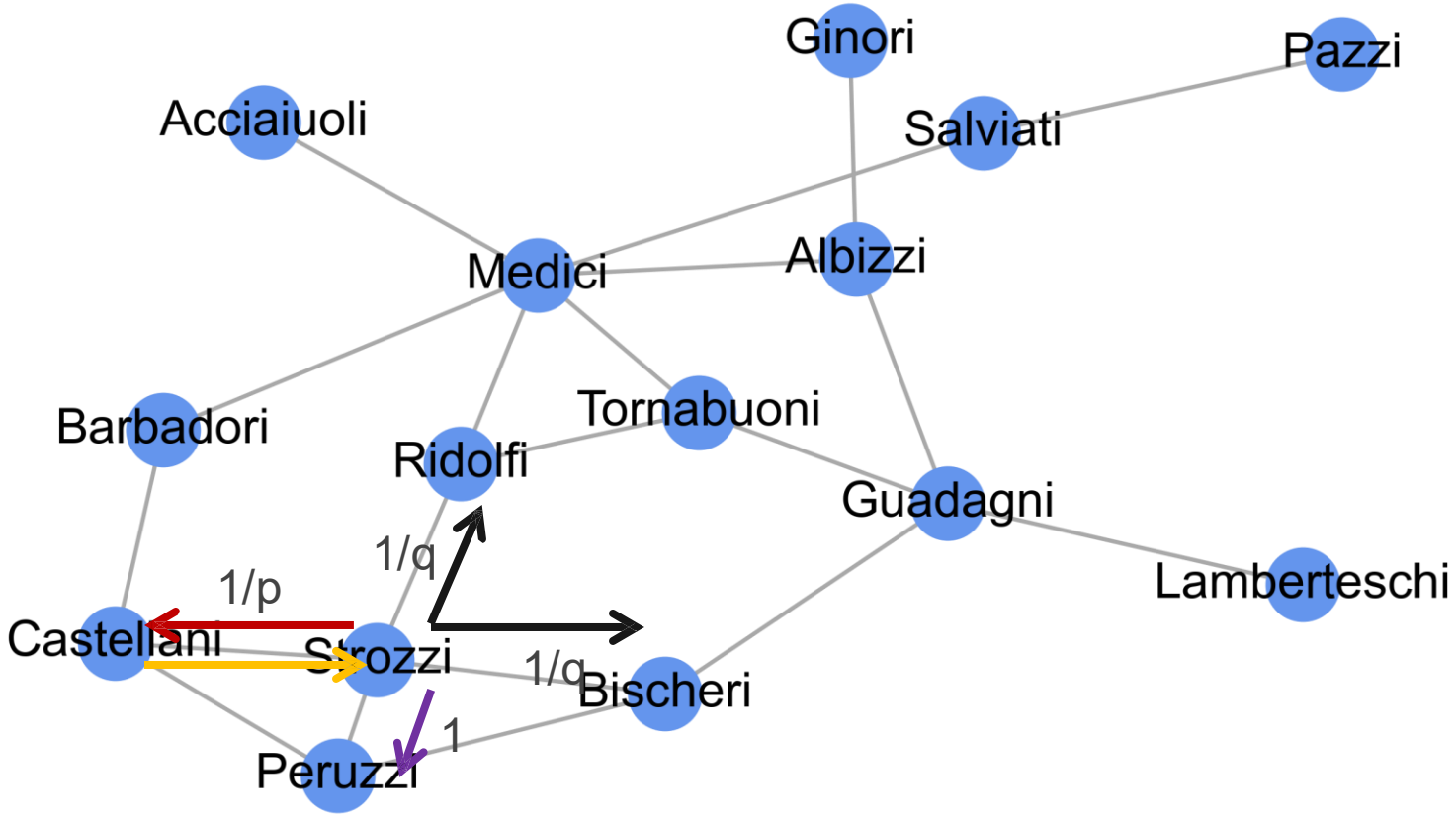
Target node

…	Pazzi	Salviati	Guadagni	Medici	Albizzi
Pazzi		1		1	1
Salviati	1				
Guadagni					
Medici	1		1	1	1
Albizzi	1			1	

Step 2: Create embeddings representing how similar the neighbors of each node are

node2vec

Difference with deepwalk: generate “sentences” using biased random walks.
p = controls probability of going back to previous node
1 = weight of going to a node adjacent to the previous one
q = controls probability of going to new nodes



Using node2vec

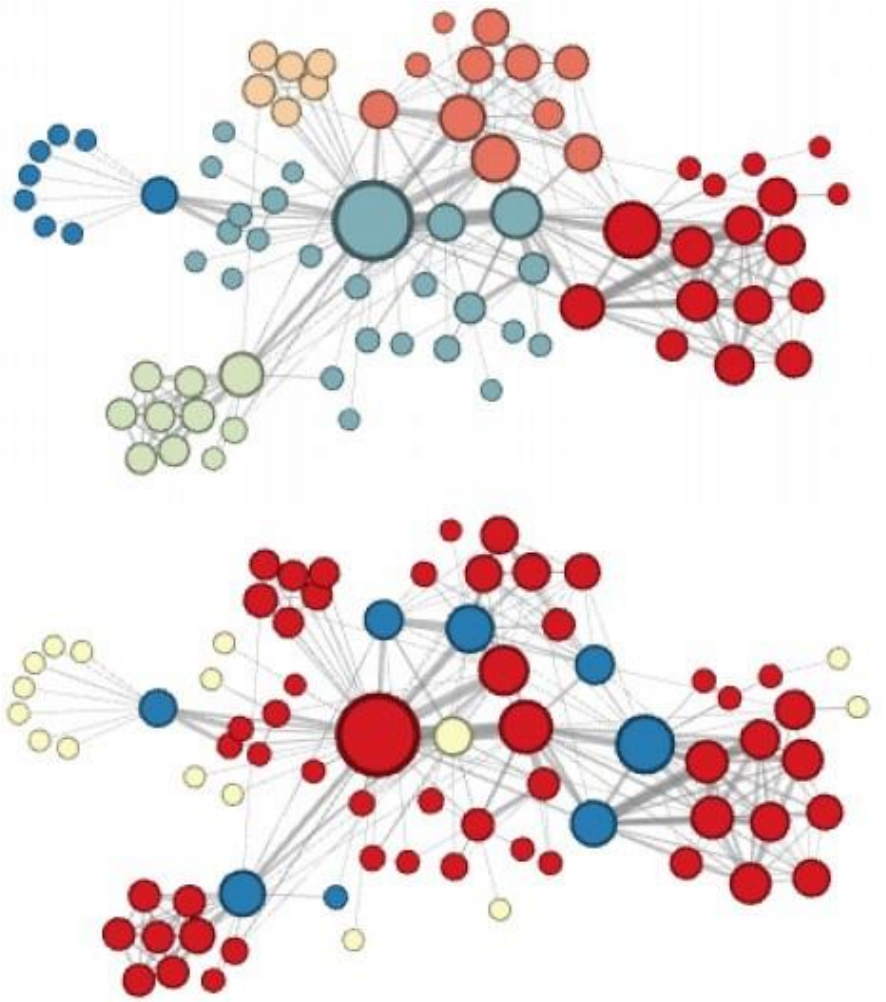


Figure 3: Complementary visualizations of Les Misérables coappearance network generated by *node2vec* with label colors reflecting homophily (top) and structural equivalence (bottom).

Depending on q

~ similiarity reflecting clusters

~ similiarity reflecting "structural roles"

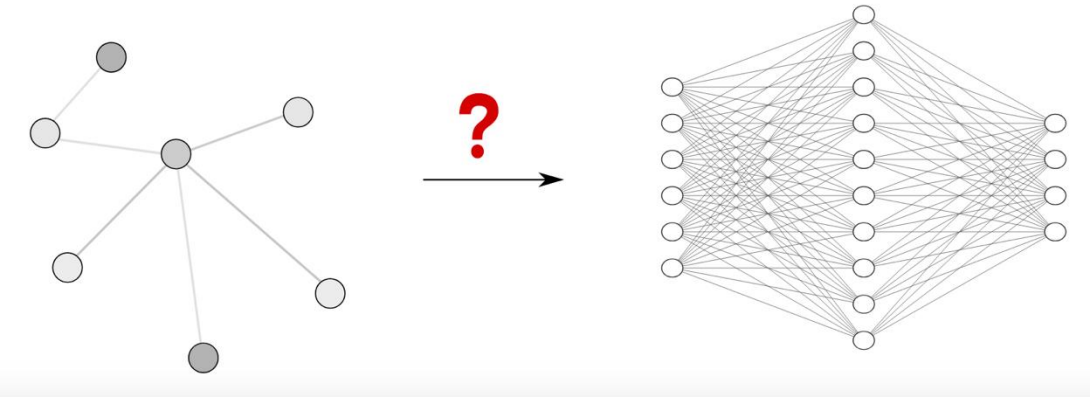
Big problem: how to set up q and p?

Part 3.3: Graph Neural Networks (GNNs)

Node2vec created the embeddings in an unsupervised (or self-supervised) way. But we can do it in a *supervised* way, so **node embeddings are similar if nodes have the same outcome**

In the context of link prediction → The same outcome = being connected

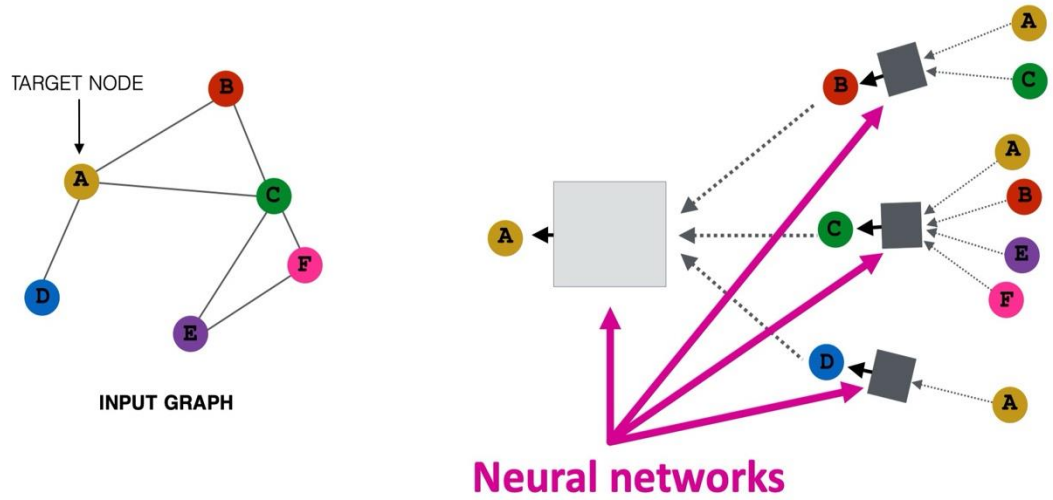
Problem:



https://distill.pub/2021/understanding-gnns/

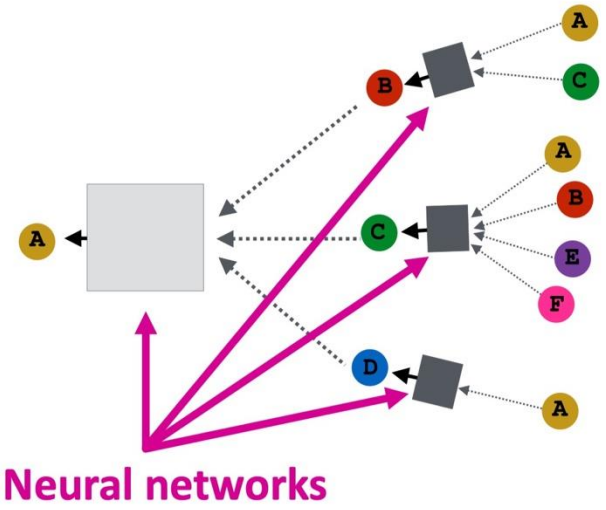
Solution:

- **Intuition:** Nodes aggregate information from their neighbors using neural networks



https://web.stanfod.edu/class/cs224w/

Example (Graph Convolutional Network)



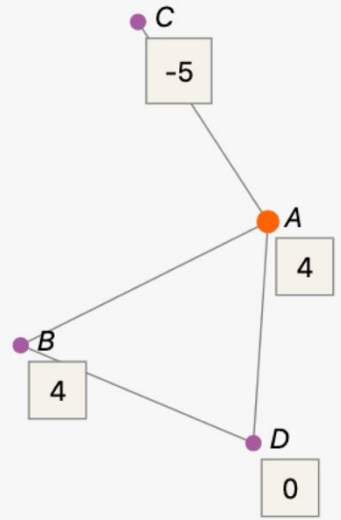
https://web.stanfod.edu/class/cs224w/

$$h^{(k)} = f((D^{-1}A + I) \cdot h^{(k-1)}W^{(k)^T})$$

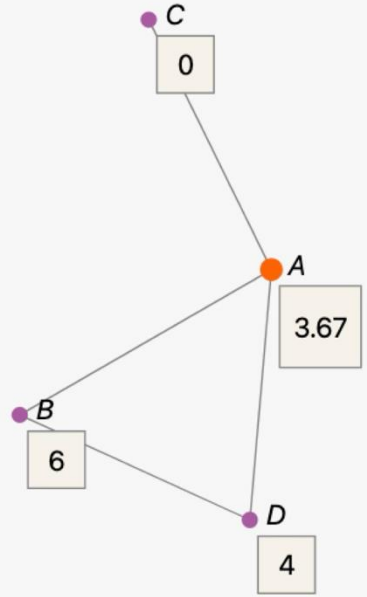
- $h^{(k)}$: Node embedding (at layer k)
- $(D^{-1}A + I)$: Normalized adjacency matrix
- $h^{(k-1)}$: Node embedding (layer k-1)
- $W^{(k)^T}$: Trainable weights

https://distill.pub/2021/gnn-intro/
https://distill.pub/2021/understanding-gnns/

GNN: Graph Neural Networks ~ message passing + feature mixing



$$
\begin{aligned}
h_A^{(1)} &= f\left(W^{(1)} \times \frac{h_B^{(0)} + h_C^{(0)} + h_D^{(0)}}{3} + B^{(1)} \times h_A^{(0)}\right) \\
&= f\left(1 \times \frac{4 + -5 + 0}{3} + 1 \times 4\right) \\
&= f\left(-0.33 + 4\right) \\
&= f\left(3.67\right) \\
&= \mathrm{ReLU}\left(3.67\right) = 3.67.
\end{aligned}
$$



$$
\begin{aligned}
h_A^{(2)} &= f\left(W^{(2)} \times \frac{h_B^{(1)} + h_C^{(1)} + h_D^{(1)}}{3} + B^{(2)} \times h_A^{(1)}\right) \\
&= f\left(1 \times \frac{6 + 0 + 4}{3} + 1 \times 3.67\right) \\
&= f\left(3.33 + 3.67\right) \\
&= f\left(7\right) \\
&= \mathrm{ReLU}\left(7\right) = 7.
\end{aligned}
$$

$$\mathbf{H}^{(\ell+1)} = \sigma\left(\hat{\mathbf{A}}\mathbf{H}^{(\ell)}\mathbf{W}^{(\ell)}\right)$$

Final embedding used to predict college completion

https://distill.pub/2021/understanding-gnns/

Recap node embeddings

We want to create low-dimensional embeddings:

- Spectral methods → Based on the eigenvectors of the network
  - Unsupervised, similarity = similar position in the network
- Shallow neural networks → Based on random walks on the network
  - Unsupervised/self-supervised, similarity = similar neighborhood
- Graph Neural Networks → Based on message passing between nodes + node features
  - Supervised, similarity = similar position in the network and similar outcome
  - Many GNNs (GCN, GAT, SAGE, Transformer…)

Stacking classifiers

Stacking = Combining multiple models to build a stronger one.

Each individual method (Jaccard, Katz, SVD, etc.) makes predictions for link probabilities, but each has its own strengths and weaknesses.

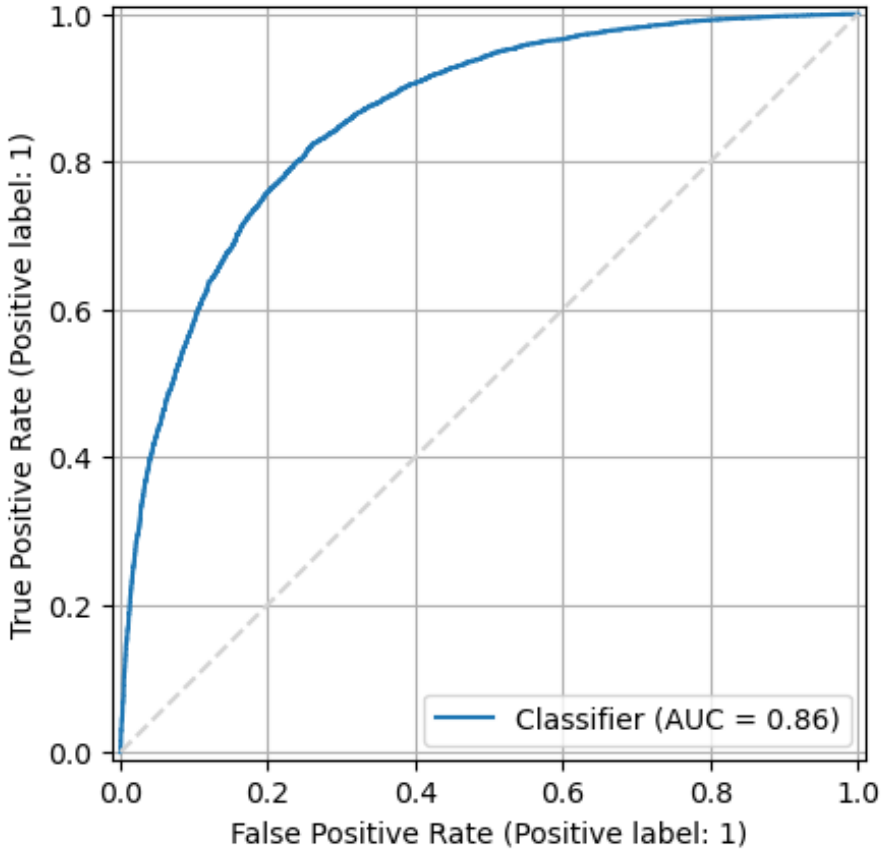
Instead of choosing the best method, we let a new model learn how to combine them effectively.

How It Works:

* Train several base classifiers using different features (e.g. Jaccard, Katz).
* Each base model outputs a prediction score (probability).
* Combine these predictions into a new dataset.
* Train a meta-classifier (e.g. Logistic Regression) on these outputs.

Predict links in the PPI network

Evaluated using AUC



- Based on common neighbors
  - → change nx.jaccard_coefficient
- Based on paths
  - → Try different path lengths and katz decay
- Based on Spectral methods
  - → Try different methods, different embedding size
  - → Try different ways to combine embeddings (pairwise_L1)
- Based on node2vec (if it works on your computer)
  - → Try different dimension/walk_length/context size (window)
- Based on GNN
  - → Try different dimension/learning rate/dropout/convolutional layer