Link Prediction

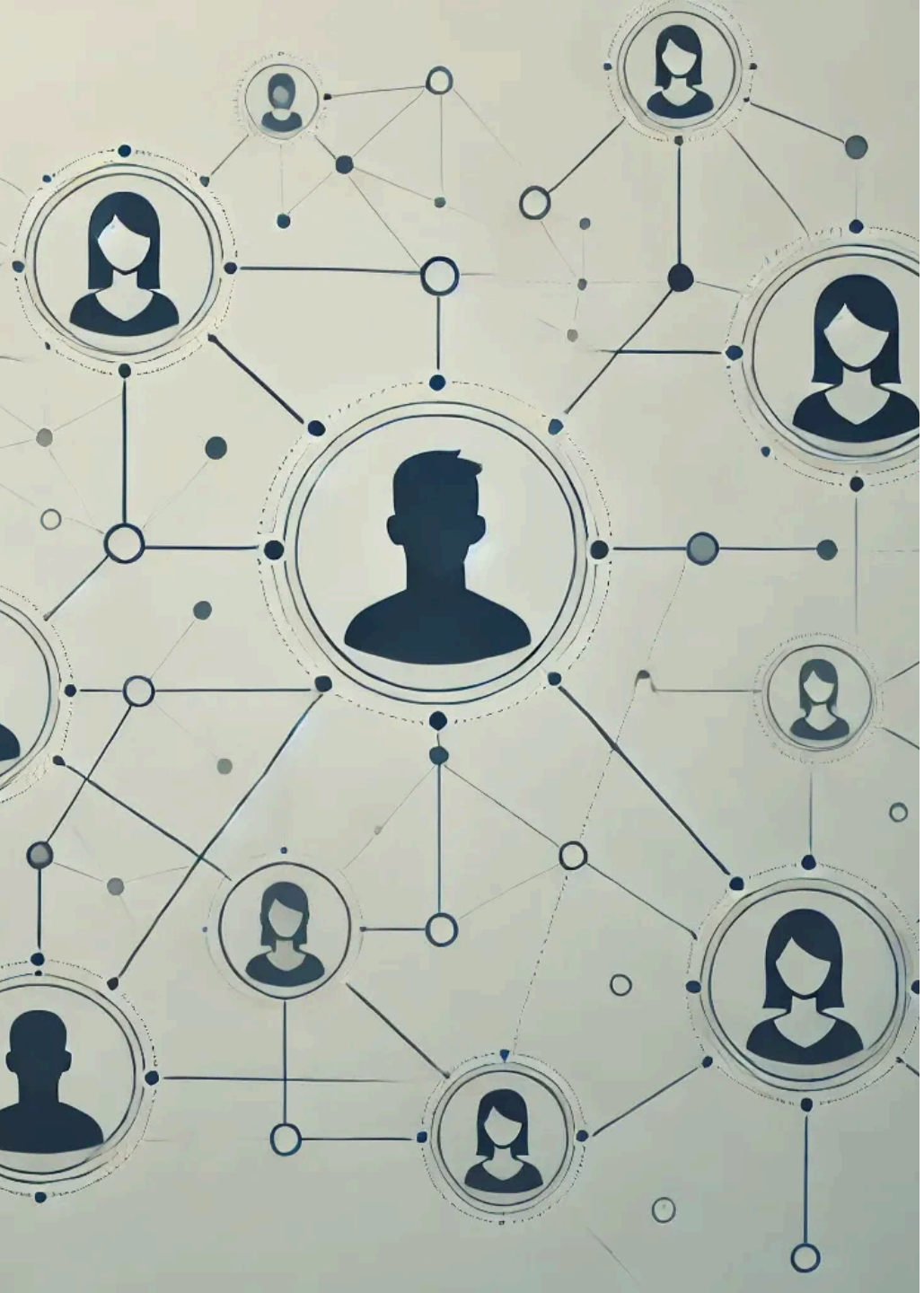
Agenda

- Introduction to link prediction problems
- Exploration of methods
- Comparisons and conclusions

Link prediction problem

Link prediction is to predicting whether two nodes in a network are likely to have a link

Applications



- **Friend Suggestions**
  Predicts potential connections based on mutual friends and shared interests.
- **Fraud Detection**
  Identifies suspicious patterns, such as clusters of fake profiles that attempt to connect to real users.

- **Product Recommendations** Suggests items based on previous purchases and browsing history.
- **Personalized Experience** Enhances user engagement by predicting relevant content.

- **Protein Interactions**
  Predicts potential interactions between proteins, aiding in functional discovery.
- **Genetic Disease Understanding**
  Assists in understanding genetic diseases by predicting gene-gene interactions.

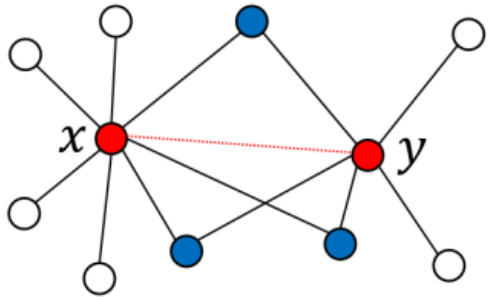
Traditional Methods

Categories of link prediction methods

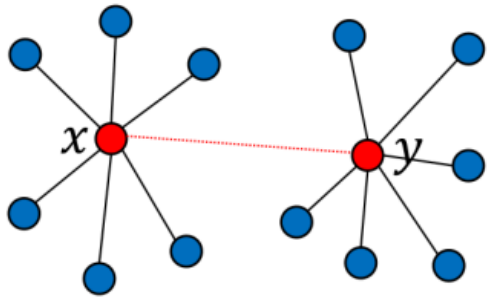
- **Heuristic methods**
  use simple node similarity scores as the likelihood of links
- **Embedding (latent-feature) methods**
  produce latent representations of nodes using a variety of optimization algorithms
- **Content-based methods**
  use node-only features, disregarding the graph structure

Local heuristics

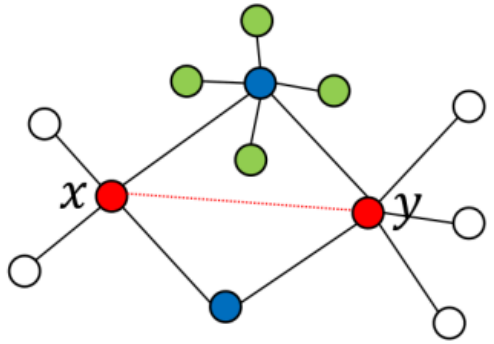
This class of heuristics considers only the local subgraphs of the nodes in question.



common neighbors (CN):
$|\Gamma(x) \cap \Gamma(y)|$



preferential attachment (PA):
$|\Gamma(x)| \cdot |\Gamma(y)|$



Adamic-Adar (AA):
$\sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{\log |\Gamma(z)|}$

Here $\Gamma(x)$ denotes the neighbors of x.

Global Heuristics

These heuristics, also known as high order heuristics, make their judgement based on the entire network.

Katz index

This index calculates a score based on all possible paths between two nodes, where longer paths contribute less to the score than shorter paths. This captures the idea that while two nodes might not be directly connected, they could still be "close" in terms of the network structure.

$$K_{ij} = \sum_{l=1}^{\infty} \beta^l \cdot \left(A^l\right)_{ij}$$

where $0 < \beta < 1$ is a decay factor and l is the path length

Rooted PageRank

Let π_x be a stationary distribution representing the long-term behavior of a random walker on a graph that following a set of rules:

- move to one of its neighboring nodes with probability α
- return to x with probability $1 - \alpha$

Therefore, $[\pi_x]_y$ essentially estimates how likely it is to reach node y from node x, which can be interpreted as the strength or likelihood of a connection (link) from x to y.

$$f_{\mathrm{RPR}}(x, y) = [\boldsymbol{\pi}_x]_y + [\boldsymbol{\pi}_y]_x$$

γ-Decaying Heuristic Theory

$$\mathscr{H}(x,y) = \eta \sum_{l=1}^{\infty} \gamma^{l} f(x,y,l)$$

> It can be shown that proves that under certain conditions, any γ-decaying heuristic can be approximated from an h-hop enclosing subgraph, and the approximation error decreases at least exponentially with h. (...) Most existing link prediction heuristics inherently share the same γ-decaying heuristic form, and thus can be effectively approximated from an h-hop enclosing subgraph.

Name	Formula	Order
common neighbors	$\lvert\Gamma(x)\cap\Gamma(y)\rvert$	first
Jaccard	$\frac{\lvert\Gamma(x)\cap\Gamma(y)\rvert}{\lvert\Gamma(x)\cup\Gamma(y)\rvert}$	first
preferential attachment	$\lvert\Gamma(x)\rvert\cdot\lvert\Gamma(y)\rvert$	first
Adamic-Adar	$\sum_{z\in\Gamma(x)\cap\Gamma(y)}\frac{1}{\log\lvert\Gamma(z)\rvert}$	second
resource allocation	$\sum_{z\in\Gamma(x)\cap\Gamma(y)}\frac{1}{\lvert\Gamma(z)\rvert}$	second
Katz	$\sum_{l=1}^{\infty}\beta^{l}\lvert\text{walks}^{\langle l\rangle}(x,y)\rvert$	high
rooted PageRank	$[\pi_x]_y+[\pi_y]_x$	high
SimRank	$\gamma\frac{\sum_{a\in\Gamma(x)}\sum_{b\in\Gamma(y)}\text{score}(a,b)}{\lvert\Gamma(x)\rvert\cdot\lvert\Gamma(y)\rvert}$	high

Heuristic limitations

- Most heuristics only work for homogeneous graphs
- Work well only when the network formation aligns with the heuristic
- They only capture a small subset of all possible structure patterns

Embedding methods

The second class of traditional link prediction methods, known as embedding methods (or latent-feature methods), compute hidden node representations. These features, derived from network structures like adjacency or Laplacian matrices, are not directly interpretable.

Matrix Factorization

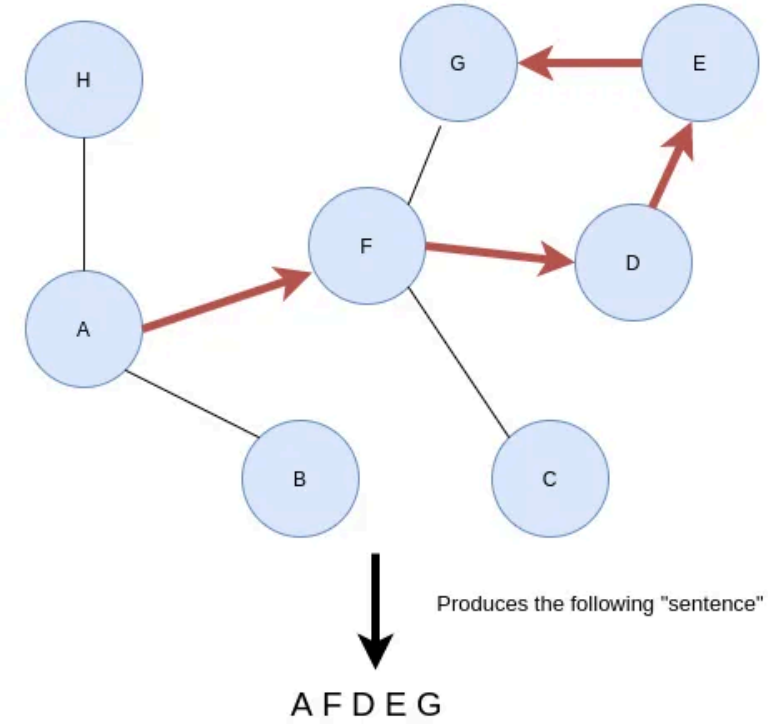
Matrix factorization decomposes the observed adjacency matrix A into the product of a low-rank latent-embedding matrix Z and its transpose, allowing for the edges reconstruction:

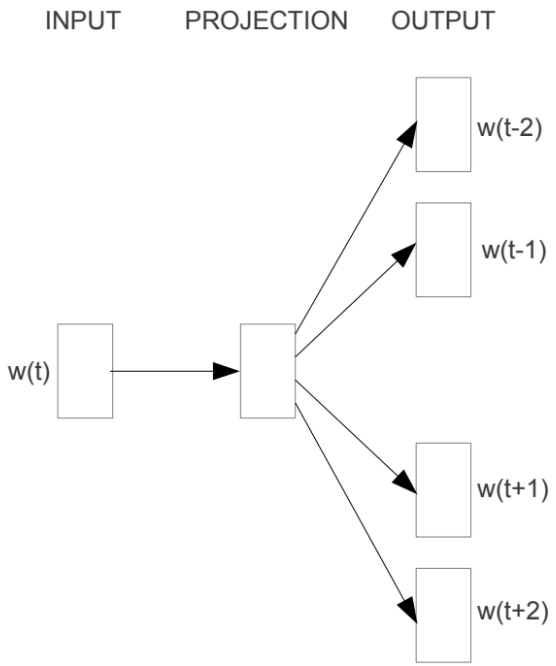
$$\hat{A}_{i,j} = \mathbf{z}_i^\top \mathbf{z}_j,$$

The latent embeddings are learned by minimizing the mean-squared error between the reconstructed and true adjacency matrices:

$$\mathscr{L} = \frac{1}{|\mathscr{E}|} \sum_{(i,j)\in\mathscr{E}} (A_{i,j} - \hat{A}_{i,j})^2.$$

Node2Vec / DeepWalk





Skip-gram

$$\log\left(\mathrm{vol}(\mathcal{G})\left(\frac{1}{w}\sum_{r=1}^{w}(D^{-1}A)^{r}\right)D^{-1}\right)-\log(b)$$

Content-based methods

> Content-based methods leverage explicit content features associated with nodes for link prediction (...) However, they usually have worse performance than heuristic and latent-feature methods (...) Thus, they are usually used together with the other two types of methods.

GNN Methods

Can graph neural network compete with the other methods?

Link prediction paradigms

- **Node-based**
  aggregate the pairwise node representations learned by a GNN as the link representation
- **Subgraph-based**
  extract a local subgraph around each link and use the subgraph representation learned by a GNN as the link representation.

Graph Convolutional Network (GCN)

$$\hat{A}_{i,j} = \sigma(\mathbf{z}_i^\top \mathbf{z}_j), \quad \text{where } \mathbf{z}_i = Z_{i,:}, Z = \text{GCN}(X, A)$$

The convolution is neighborhood aggregation and resembles that of the regular convolutional networks if we think of the closest pixels as a neighborhood subgraph.

Graph AutoEncoder (GAE)

$$\hat{A}_{i,j} = \sigma(\mathbf{z}_i^\top \mathbf{z}_j), \quad \text{where } \mathbf{z}_i = Z_{i,:}, Z = \text{GCN}(X, A)$$

Such node-based model is trained to minimize the binary cross entropy between the true adjacency matrix and its reconstruction.

$$\mathscr{L} = \sum_{i \in \mathscr{V}, j \in \mathscr{V}} (-A_{i,j} \log \hat{A}_{i,j} - (1 - A_{i,j}) \log(1 - \hat{A}_{i,j}))$$

- Identity matrix may be substituted for X should there be no node feature data to consider.
- The probability of there being a link is defined as a sigmoid of the similarity between the representations of the two considered nodes with respect to their dot product.
- In practice, the loss from existing edges is usually upscaled by the ratio between the existing and missing edges to balance out the data.

Variational AutoEncoders (VAEs)

Variational AutoEncoders are a bayesian inference approach to data generation. We presuppose the existance of a some hidden variable z which generates an observation x and we would like to compute $p(z|x)$.

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)}$$

This is troublesome to compute and so we approximate the $p(z|x)$ with $q(z|x)$ and minimize

$$E_{q(z|x)} \log p(x|z) - KL(q(z|x)||p(z))$$

Penalizing reconstruction loss encourages the distribution to describe the input

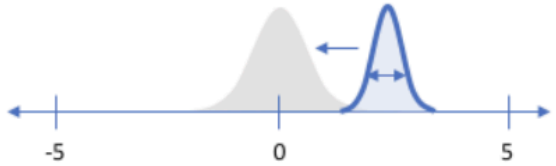
Our distribution deviates from the prior to describe some characteristic of the data

Without regularization, our network can "cheat" by learning narrow distributions

With a small enough variance, this distribution is effectively only representing a single value

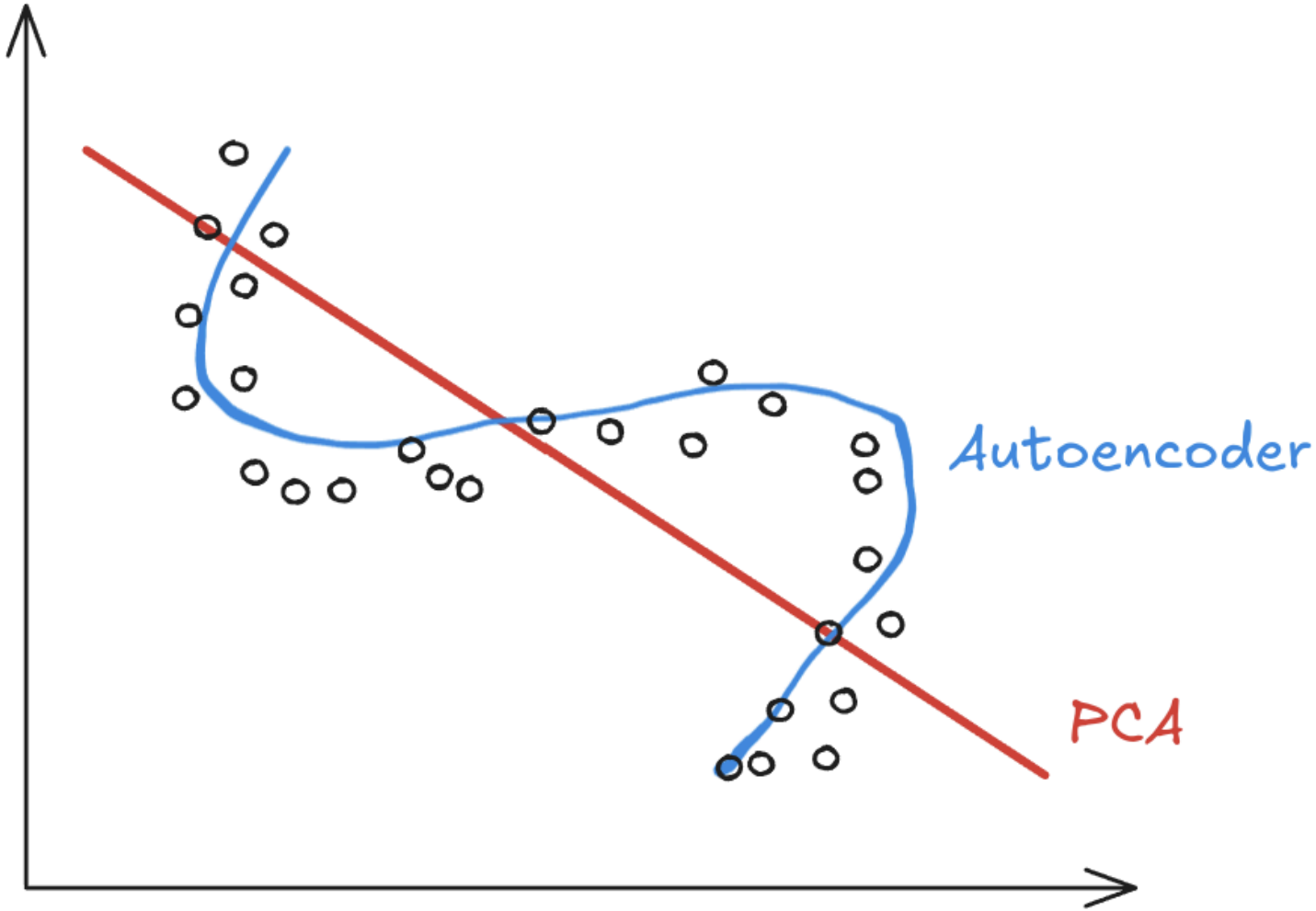
Penalizing KL divergence acts as a regularizing force

Attract distribution to have zero mean

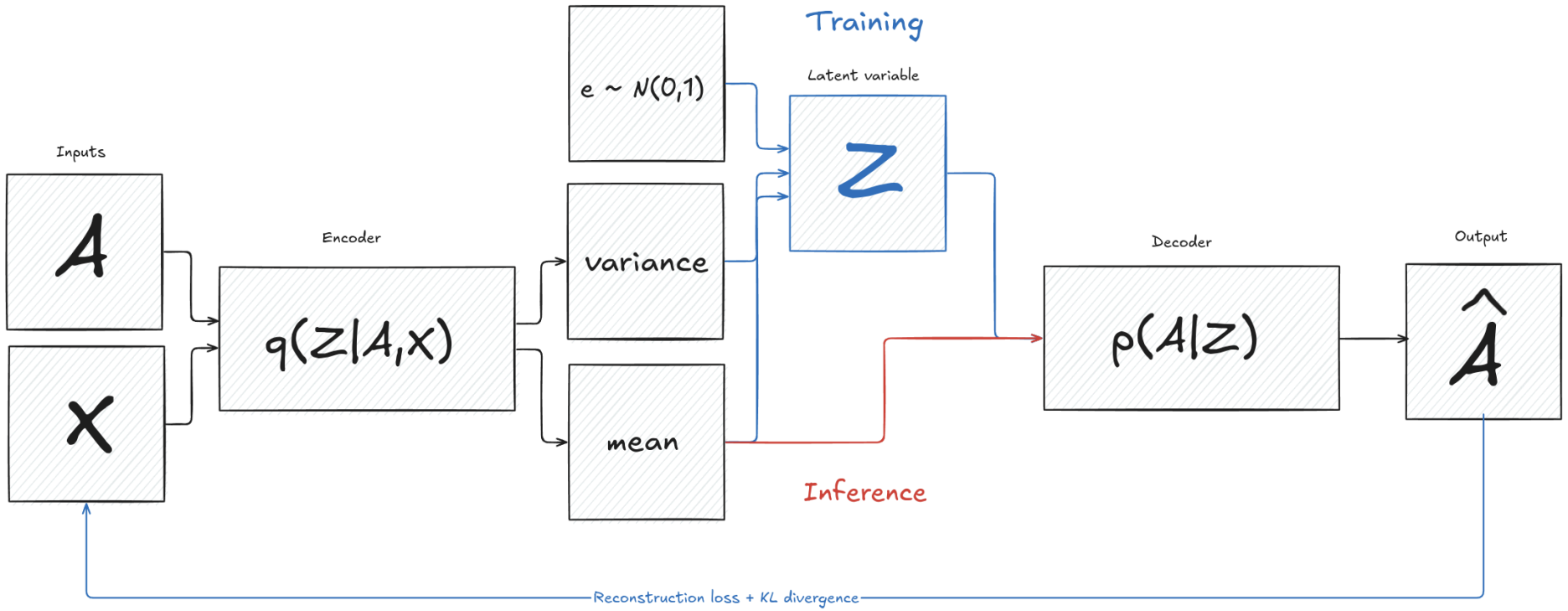


Ensure sufficient variance to yield a smooth latent space

Linear vs nonlinear dimensionality reduction



Variational Graph AutoEncoder (VGAE)



Just like in GAE a GCN encoder and a simple dot product decoder are proposed.

Discussion Results for the link prediction task in citation networks are summarized in Table 1. GAE* and VGAE* denote experiments without using input features, GAE and VGAE use input features. We report *area under the ROC curve* (AUC) and *average precision* (AP) scores for each model on the test set. Numbers show mean results and standard error for 10 runs with random initializations on fixed dataset splits.

Table 1: Link prediction task in citation networks. See [1] for dataset details.

Method	**Cora**		**Citeseer**		**Pubmed**	
	AUC	AP	AUC	AP	AUC	AP
SC [5]	84.6 ± 0.01	88.5 ± 0.00	80.5 ± 0.01	85.0 ± 0.01	84.2 ± 0.02	87.8 ± 0.01
DW [6]	83.1 ± 0.01	85.0 ± 0.00	80.5 ± 0.02	83.6 ± 0.01	84.4 ± 0.00	84.1 ± 0.00
GAE*	84.3 ± 0.02	88.1 ± 0.01	78.7 ± 0.02	84.1 ± 0.02	82.2 ± 0.01	87.4 ± 0.00
VGAE*	84.0 ± 0.02	87.7 ± 0.01	78.9 ± 0.03	84.1 ± 0.02	82.7 ± 0.01	87.5 ± 0.01
GAE	91.0 ± 0.02	92.0 ± 0.03	89.5 ± 0.04	89.9 ± 0.05	$\mathbf{96.4} \pm 0.00$	$\mathbf{96.5} \pm 0.00$
VGAE	$\mathbf{91.4} \pm 0.01$	$\mathbf{92.6} \pm 0.01$	$\mathbf{90.8} \pm 0.02$	$\mathbf{92.0} \pm 0.02$	94.4 ± 0.02	94.7 ± 0.02

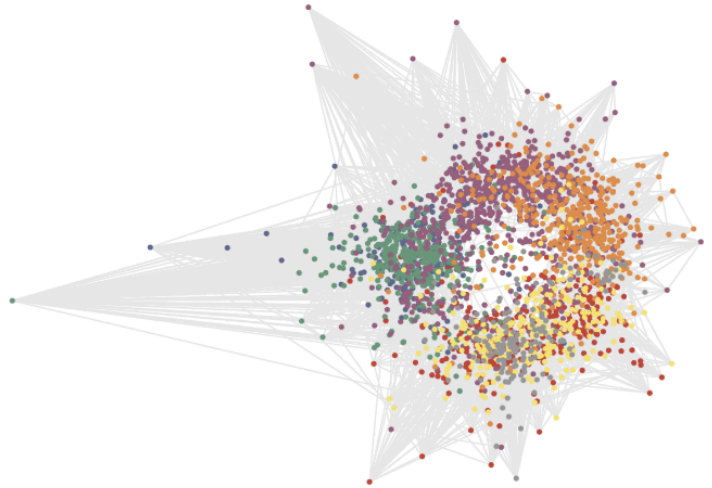


Figure 1: Latent space of unsupervised VGAE model trained on Cora citation network dataset [1]. Grey lines denote citation links. Colors denote document class (not provided during training). Best viewed on screen.

Weisfeiler-Lehman Neural Machine

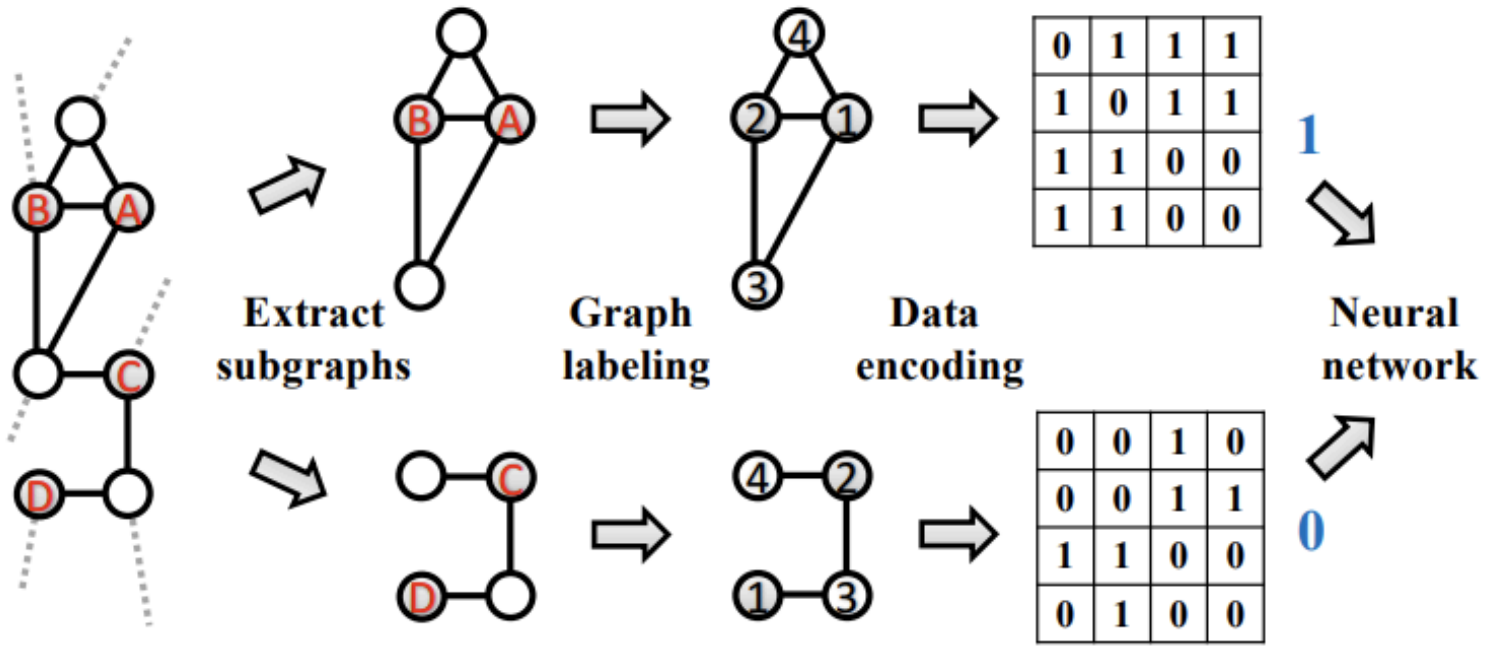


Figure 1: An illustration of WLNM. Given a network, WLNM first samples a set of positive links (illustrated as link (A, B)) and negative links (illustrated as link (C, D)) as training links, and extracts an enclosing subgraph for each link. Graph labeling is used to decide the vertex ordering and adjacency matrix. The resulting (matrix, label) pairs are fed into a neural network for link prediction.

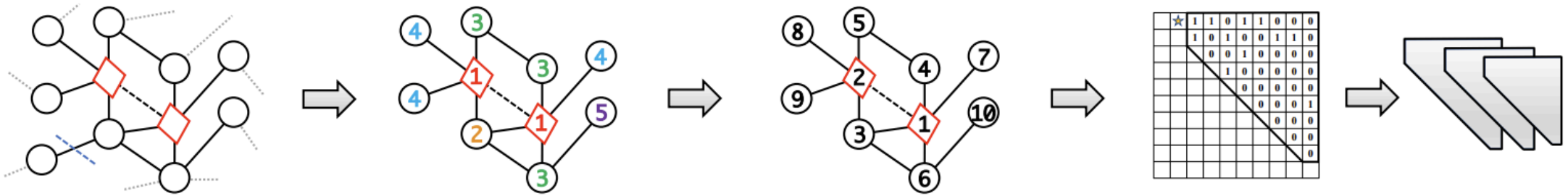
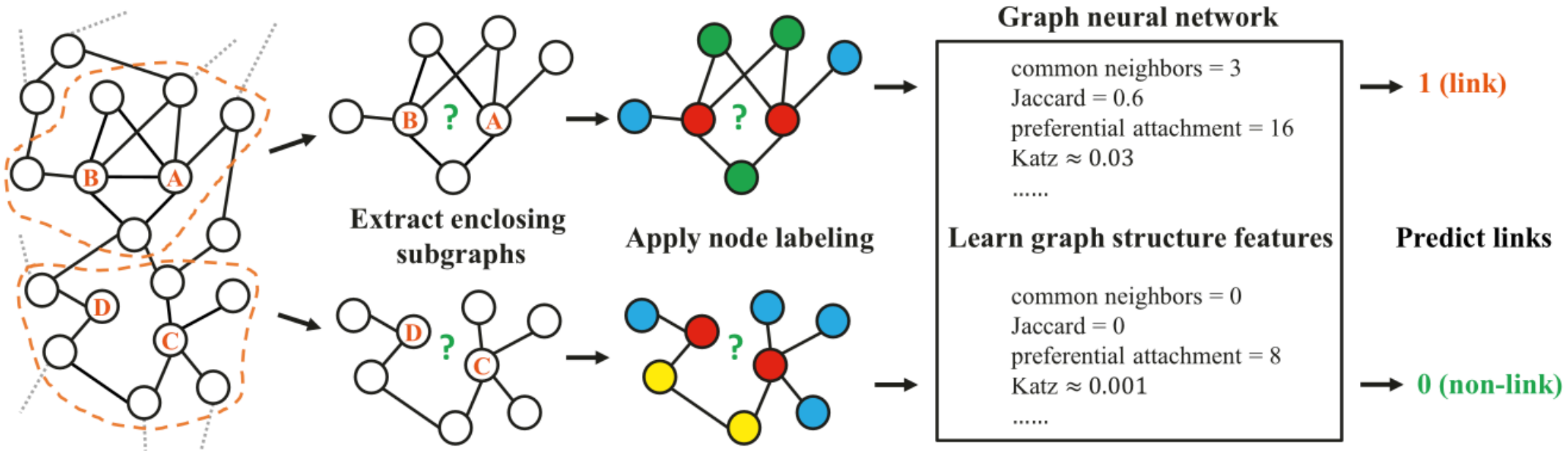
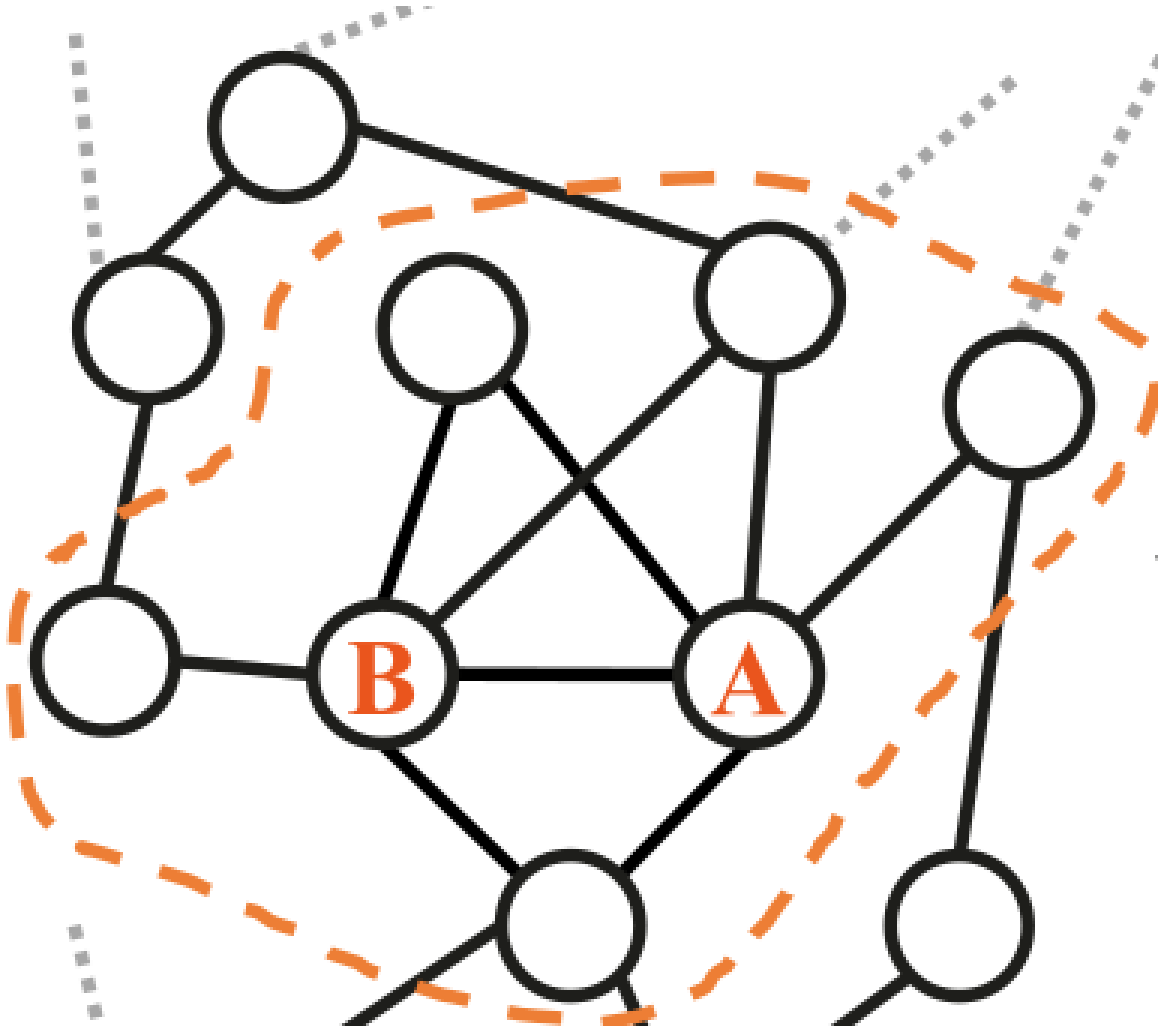


Figure 4: Illustration of the WLNM procedure: enclosing subgraph extraction (the leftmost figure), subgraph pattern encoding (the middle three figures), and neural network training (the rightmost figure).

SEAL

(learning from Subgraphs, Embeddings, and Attributes for Link prediction)

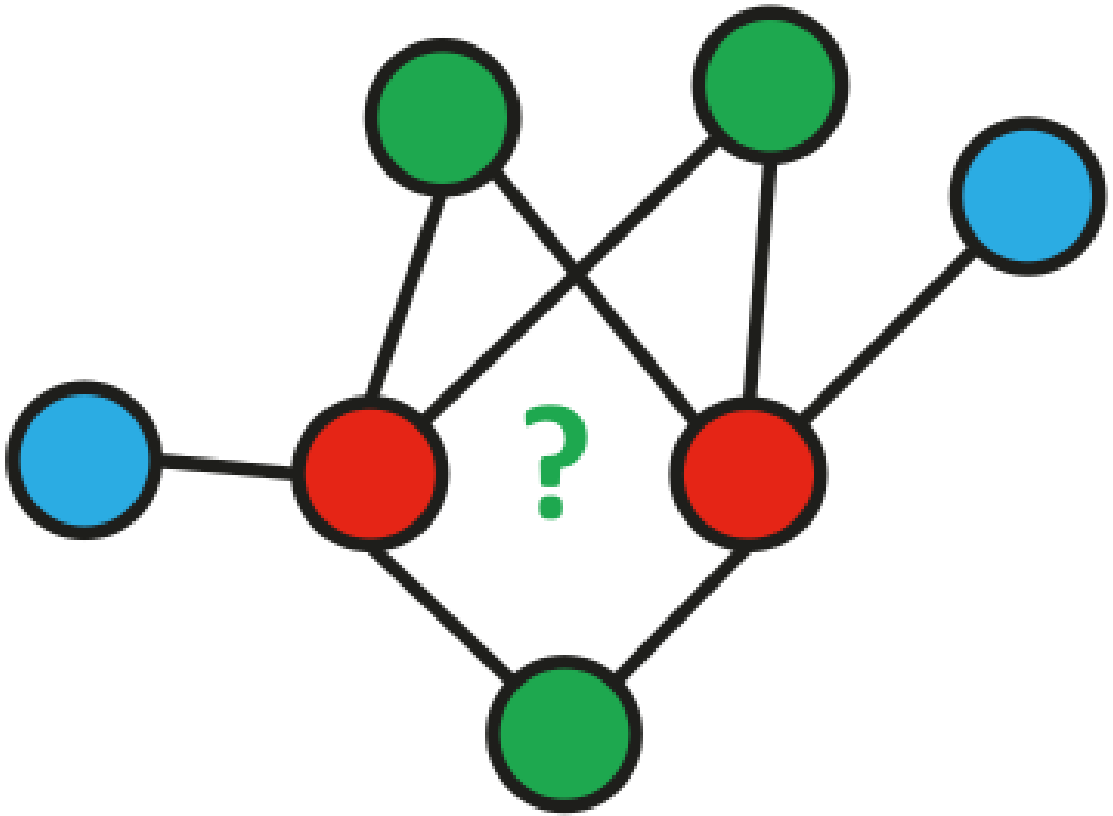




Subgraph Extraction

- **Purpose:** extracting subgraphs simplifies the prediction task by reducing the problem size and focusing on the local structure, which is more directly relevant to the potential link.
- **Process:** for each pair of nodes u and v use k-hop neighborhood (nodes within k steps from u and v) to extract the subgraph.

Definition 10.1. (Enclosing subgraph) For a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, given a set of nodes $S \subseteq \mathcal{V}$, the h-hop enclosing subgraph for S is the subgraph $\mathcal{G}_S^h$ induced from $\mathcal{G}$ by the set of nodes $\cup_{j \in S}\{i \mid d(i,j) \leq h\}$, where $d(i,j)$ is the shortest path distance between nodes i and j.



Node Labeling

- **Purpose:** give an integer label to each node in the subgraph as its additional feature to differentiate nodes of different roles within a subgraph
- **Process:** Double Radius Node Labeling (assign 1 to x and y and hash all else)

$$f_l(i) = 1 + min(d_x, d_y) + (d/2)[(d/2) + (d\%2) - 1]$$

where $d_x := d(i, x)$, $d_y := d(i, y)$, $d := d_x + d_y$

> We iteratively assign larger labels to nodes with a larger radius w.r.t. both center nodes

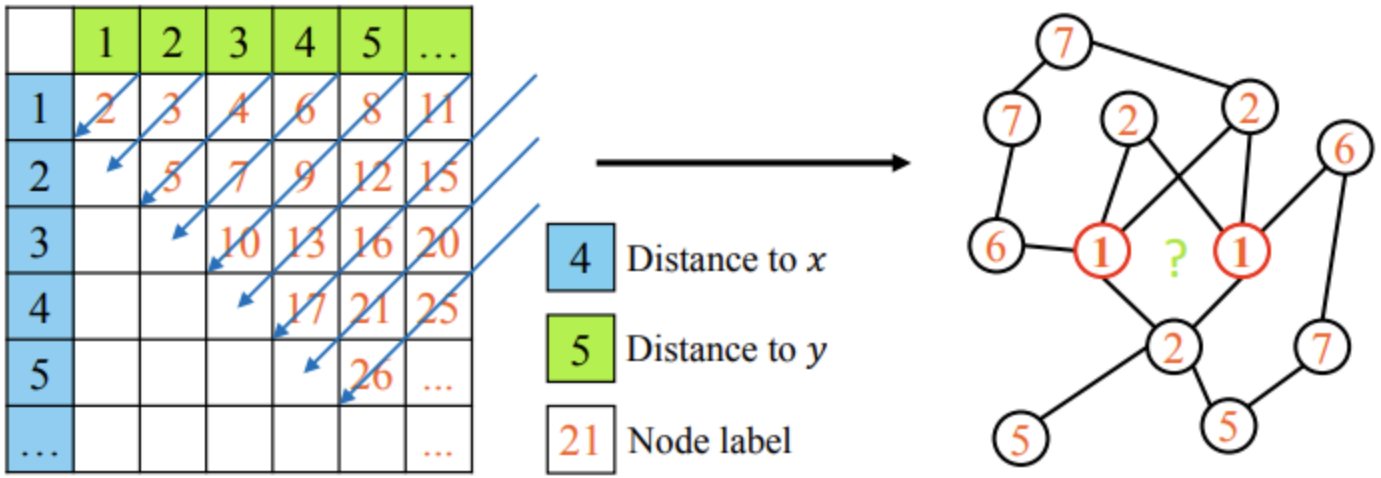


Figure 2: Double-Radius Node Labeling.

Representation learning and prediction

> SEAL feeds these enclosing subgraphs as well as their new node feature vectors into a graph-level GNN, DGCNN (Zhang et al, 2018g), to learn a graph classification function. The groundtruth of each subgraph is whether the two center nodes really have a link. To train this GNN, SEAL randomly samples N existing links from the network as positive training links, and samples an equal number of unobserved links (random node pairs) as negative training links.

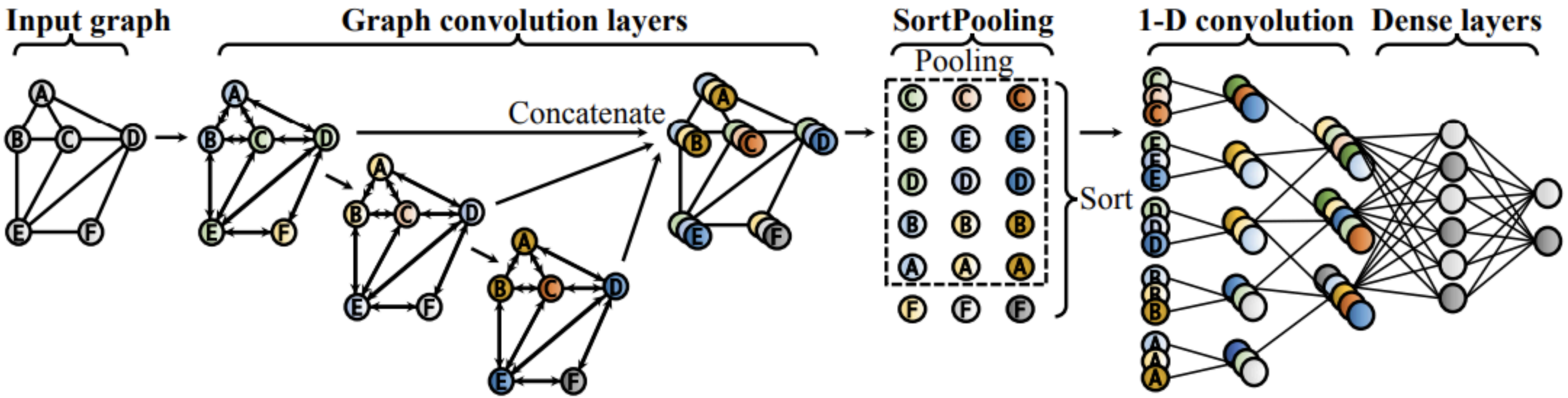


Figure 3: The DGCNN architecture.

Table 5: Comparison with heuristic methods (AP), 90% training links.

Data	CN	Jaccard	PA	AA	RA	Katz	PR	SR	ENS	WLK	WLNM	SEAL
USAir	93.45±1.19	87.54±2.07	91.22±1.28	95.36±1.00	96.27±0.79	94.07±1.18	95.08±1.16	69.24±2.61	91.33±1.27	**96.82**±0.84	95.95±1.13	**96.80**±0.55
NS	94.39±0.96	94.44±0.93	72.85±1.88	94.46±0.93	94.46±0.93	95.05±1.08	95.11±1.04	94.98±1.02	97.68±0.36	98.79±0.40	98.81±0.49	**99.06**±0.37
PB	91.47±0.45	84.78±0.71	89.33±0.72	92.36±0.46	92.37±0.57	93.07±0.46	92.97±0.77	64.33±0.95	89.35±0.71	93.34±0.89	92.69±0.64	**94.31**±0.56
Yeast	89.34±0.62	89.15±0.67	85.36±0.85	89.53±0.63	89.55±0.63	95.23±0.39	95.47±0.43	93.42±0.64	85.54±0.85	96.82±0.35	96.40±0.38	**98.33**±0.37
C.ele	82.62±1.51	77.06±2.55	75.49±1.86	86.46±1.43	87.10±1.53	85.93±1.69	**89.56**±1.57	68.61±2.31	75.69±1.86	88.96±2.06	85.08±2.05	89.48±1.85
Power	58.77±0.88	58.77±0.89	51.93±1.16	58.76±0.89	58.76±0.90	79.82±0.91	80.56±0.91	77.02±0.93	83.63±1.37	83.02±3.19	87.16±0.77	**89.55**±1.29
Router	56.39±0.53	55.84±0.80	69.03±0.95	56.50±0.51	56.51±0.50	64.52±0.81	64.91±0.85	58.82±1.12	69.25±0.96	86.59±2.23	93.53±1.09	**96.23**±1.71
E.coli	93.49±0.38	82.42±0.59	94.04±0.33	96.05±0.25	96.72±0.25	94.83±0.30	96.41±0.33	55.01±0.86	94.11±0.33	97.25±0.42	97.50±0.23	**98.03**±0.20

Table 6: Comparison with latent feature methods (AP), 90% training links.

Data	MF	SBM	N2V	LINE	SPC	VGAE	SEAL
USAir	94.36±0.79	95.08±1.10	89.71±2.97	79.70±11.76	78.07±2.92	89.27±1.29	**97.13**±0.80
NS	78.41±3.85	92.13±2.36	94.28±0.91	85.17±1.65	90.83±2.16	95.83±1.04	**98.12**±0.77
PB	93.56±0.71	93.35±0.52	84.79±1.03	78.82±2.71	86.57±0.61	90.38±0.72	**94.55**±0.43
Yeast	92.01±0.47	92.73±0.44	94.90±0.38	90.55±2.39	94.63±0.56	95.19±0.36	**97.95**±0.35
C.ele	83.63±2.09	84.66±2.95	83.12±1.90	67.51±2.72	62.07±2.40	78.32±3.49	**88.81**±2.32
Power	53.50±1.22	65.48±1.85	81.49±0.86	56.66±1.43	**91.00**±0.58	75.91±1.56	86.69±1.50
Router	82.59±1.38	84.67±1.89	68.66±1.49	71.92±1.53	73.53±1.47	70.36±0.85	**95.66**±1.23
E.coli	95.59±0.31	95.30±0.27	90.87±1.48	86.45±1.82	96.08±0.37	92.77±0.65	**97.83**±0.20

Node-based vs Subgraph-methods

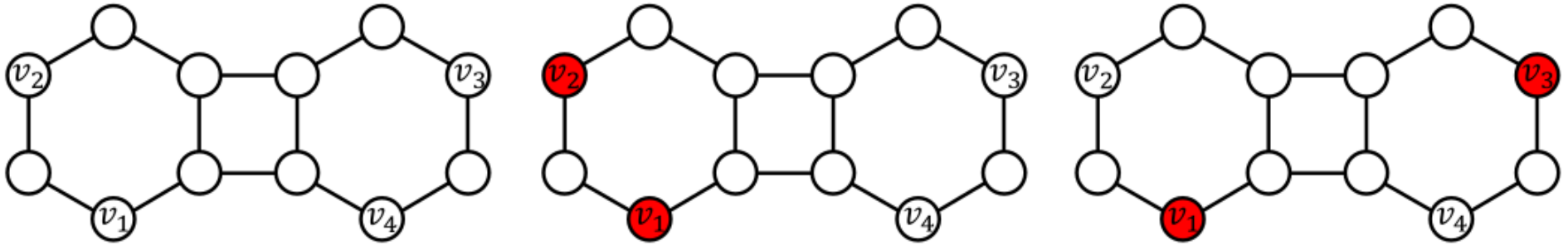


Fig. 10.3: The different link representation ability between node-based methods and subgraph-based methods. In the left graph, nodes v_2 and v_3 are isomorphic; links (v_1, v_2) and (v_4, v_3) are isomorphic; link (v_1, v_2) and link (v_1, v_3) are **not** isomorphic. However, a node-based method cannot differentiate (v_1, v_2) and (v_1, v_3). In the middle graph, when we predict (v_1, v_2), we label these two nodes differently from the rest, so that a GNN is aware of the target link when learning v_1 and v_2's representations. Similarly, when predicting (v_1, v_3), nodes v_1 and v_3 will be labeled differently (shown in the right graph). This way, the representation of v_2 in the left graph will be different from the representation of v_3 in the right graph, enabling GNNs to distinguish (v_1, v_2) and (v_1, v_3).

End