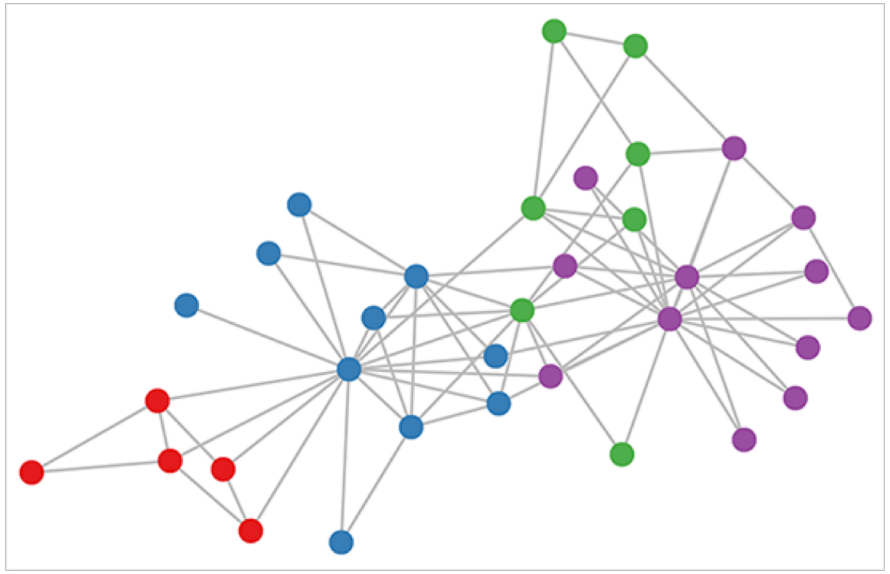
Deep learning on graphs:

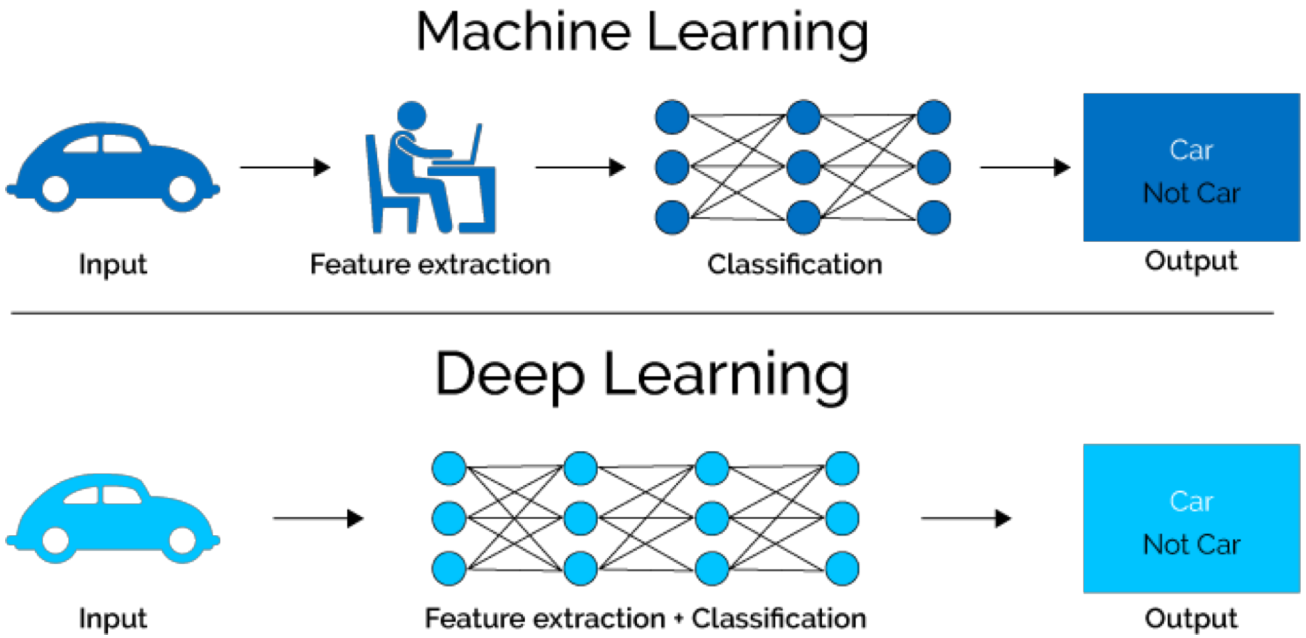
Learning from graph-structured data



Kishan KC

Oct 15, 2018

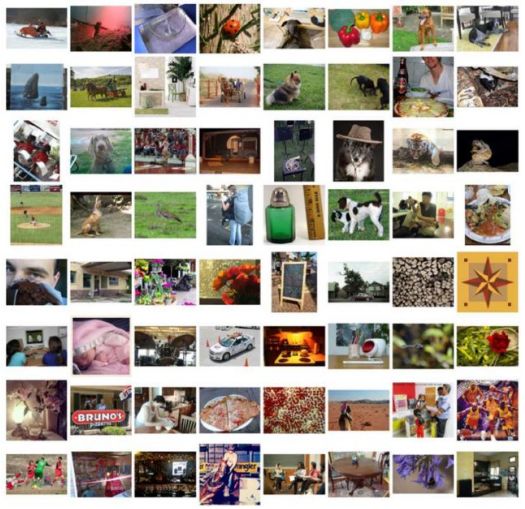
Deep Learning vs Machine Learning



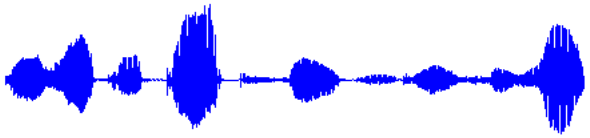
From medium.com

Deep Learning

IMAGENET

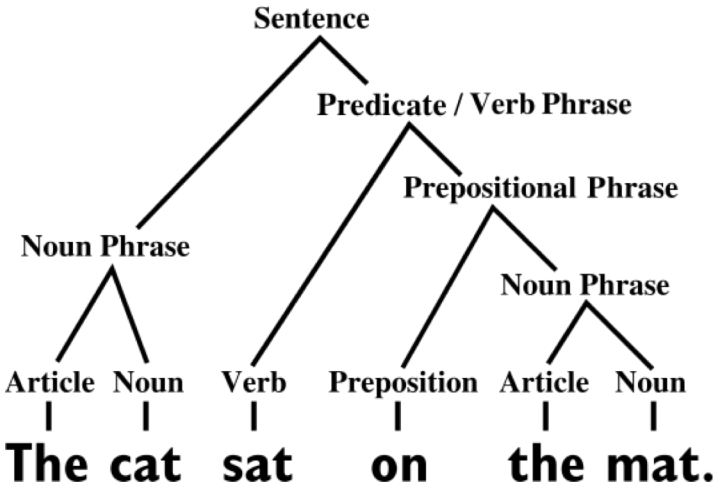


Speech data



DNA Sequence

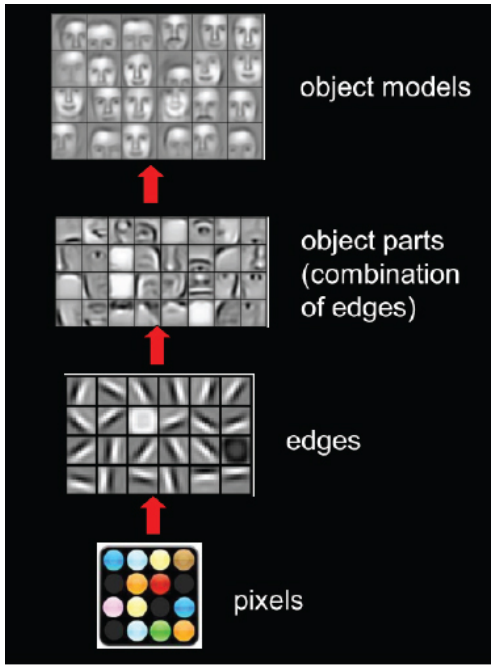
Natural Language Processing



And so on……..

What does deep neural nets learn?

- Learns representations of raw data
- Translational equivariance (weight sharing)
- Compositional hierarchy



Regular grids

A lot of real-world data does not “exist” on regular grids

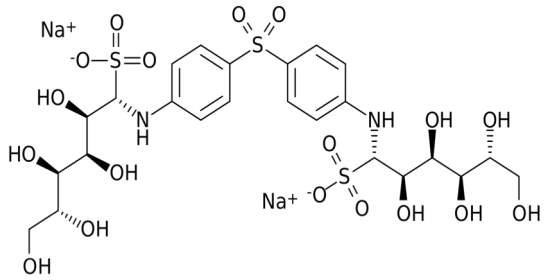
Graph-structured data

From indico.io

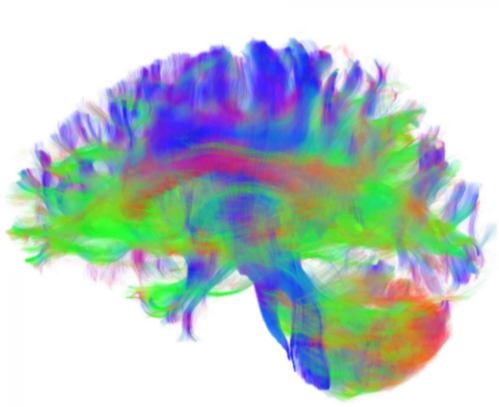
Graphs

- Ubiquitous data structure
  - Social Networks
  - Molecular Graphs
  - Biological Protein-Protein Networks
  - Recommender Systems …….

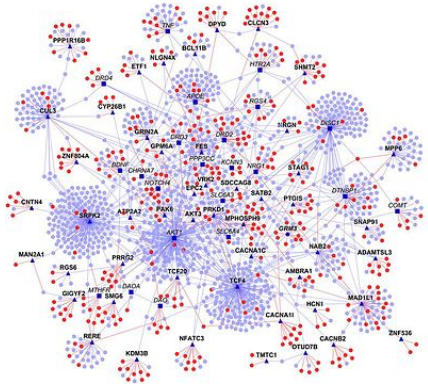
Social network



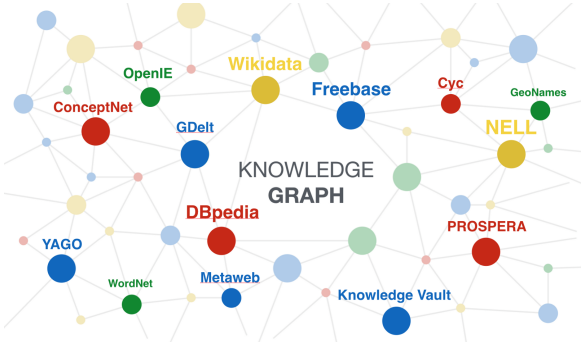
Molecules



Brain networks



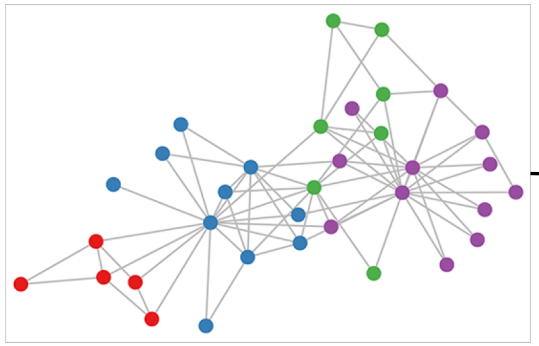
Protein interaction networks



Knowledge graph

Road maps

Applications



Graph-structured data

Deep Learning

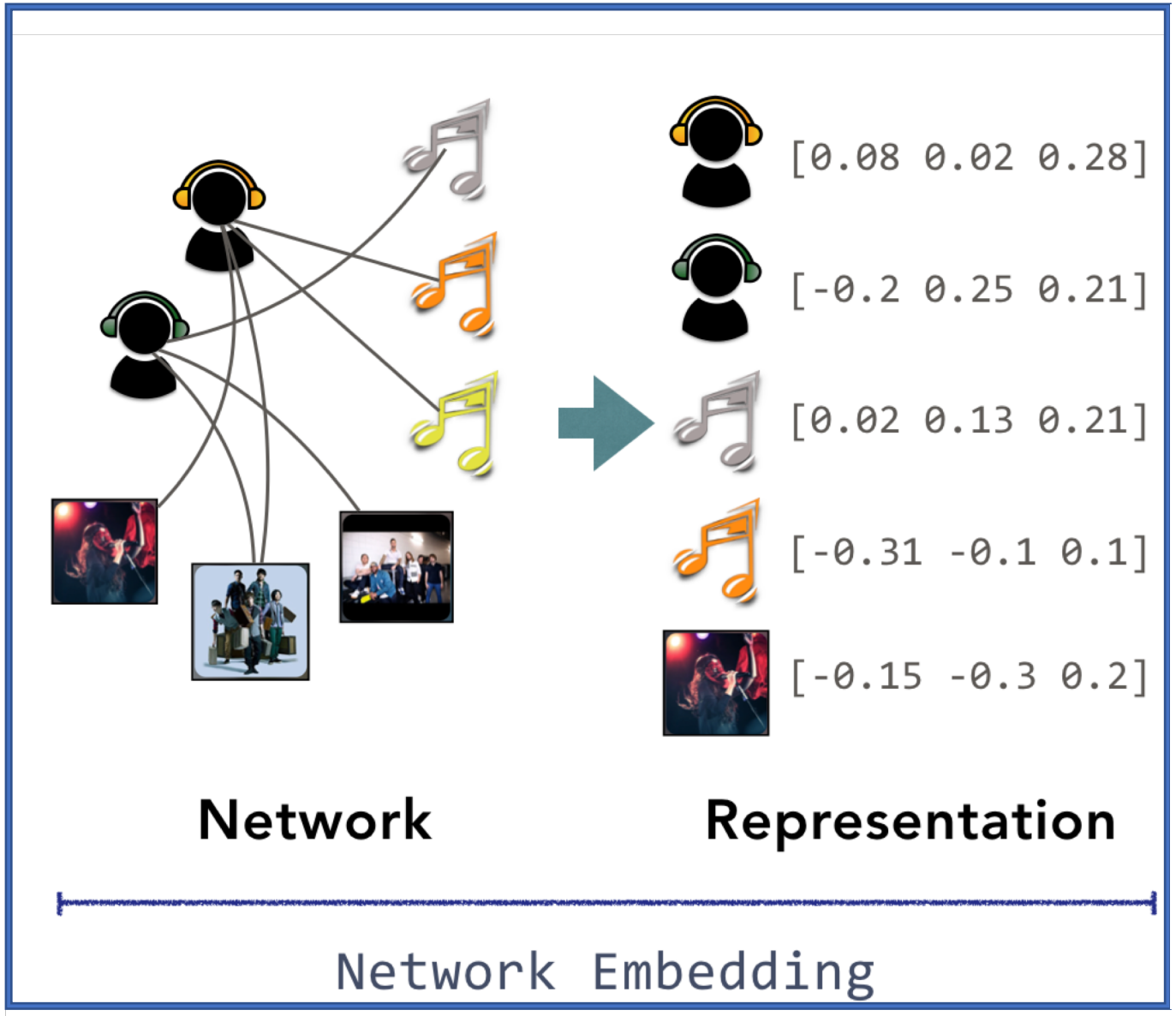
Node Classification

Graph Classification

Link Prediction

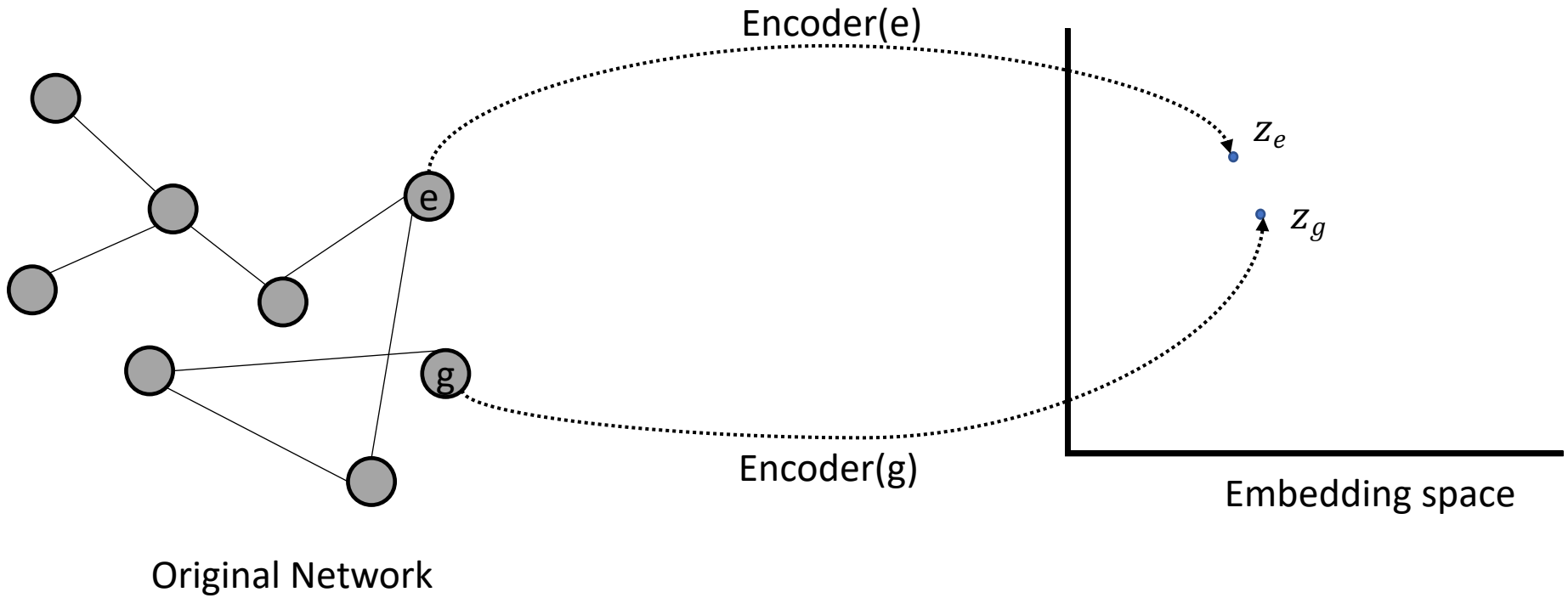
Network Visualization

Representation Learning on graphs



Embedding nodes

Goal is to encode nodes to lower dimensional representation so that the similarity in the embedding space approximates the similarity in the original network.

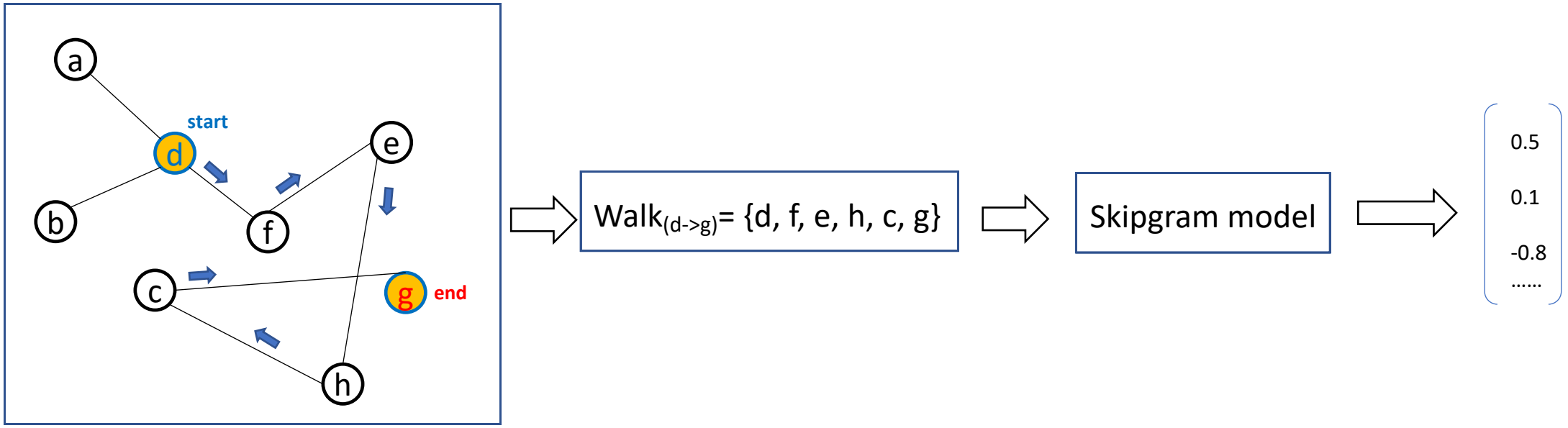


Encoder maps each node to a low dimensional vector:

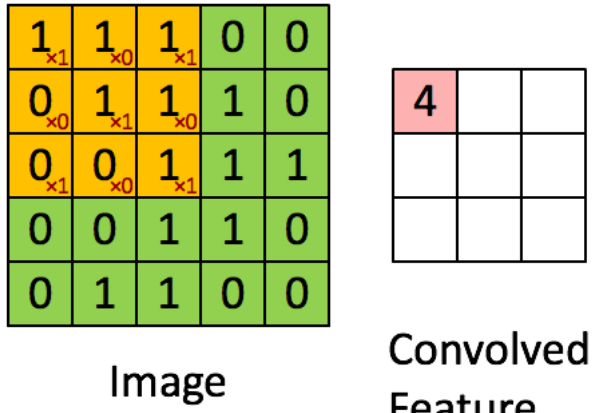
Encoder(a) = z_a

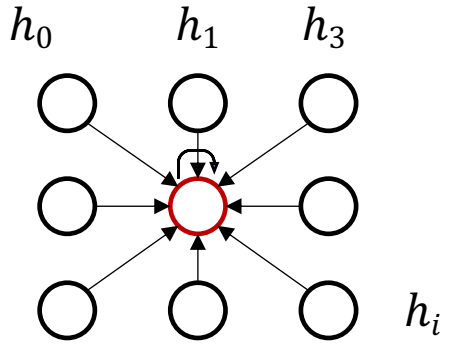
Early methods

- Random walk based approach
  - DeepWalk
  - Node2vec
- Strong baselines



Review: convolution on regular grid





$h_i \in \mathbb{R}^F$ are activations of a pixel or node

Update for a single pixel:

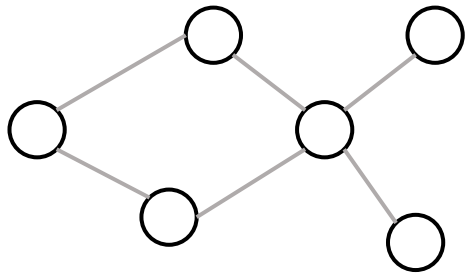
- Transform messages individually $\boldsymbol{W_i h_i}$
- Add everything up $\sum_i \boldsymbol{W_i h_i}$

Full update:

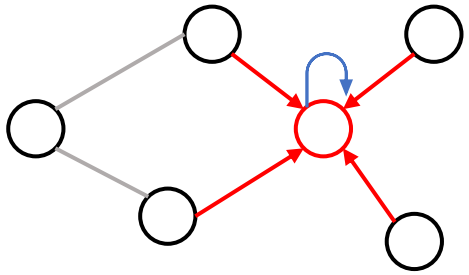
$h_4 = \sigma(W_0 h_0 + W_1 h_1 + \ldots + W_8 h_8)$

Generalization to graphs (irregular grid)

Consider an undirected graph



Aggregate neighborhood information for red node

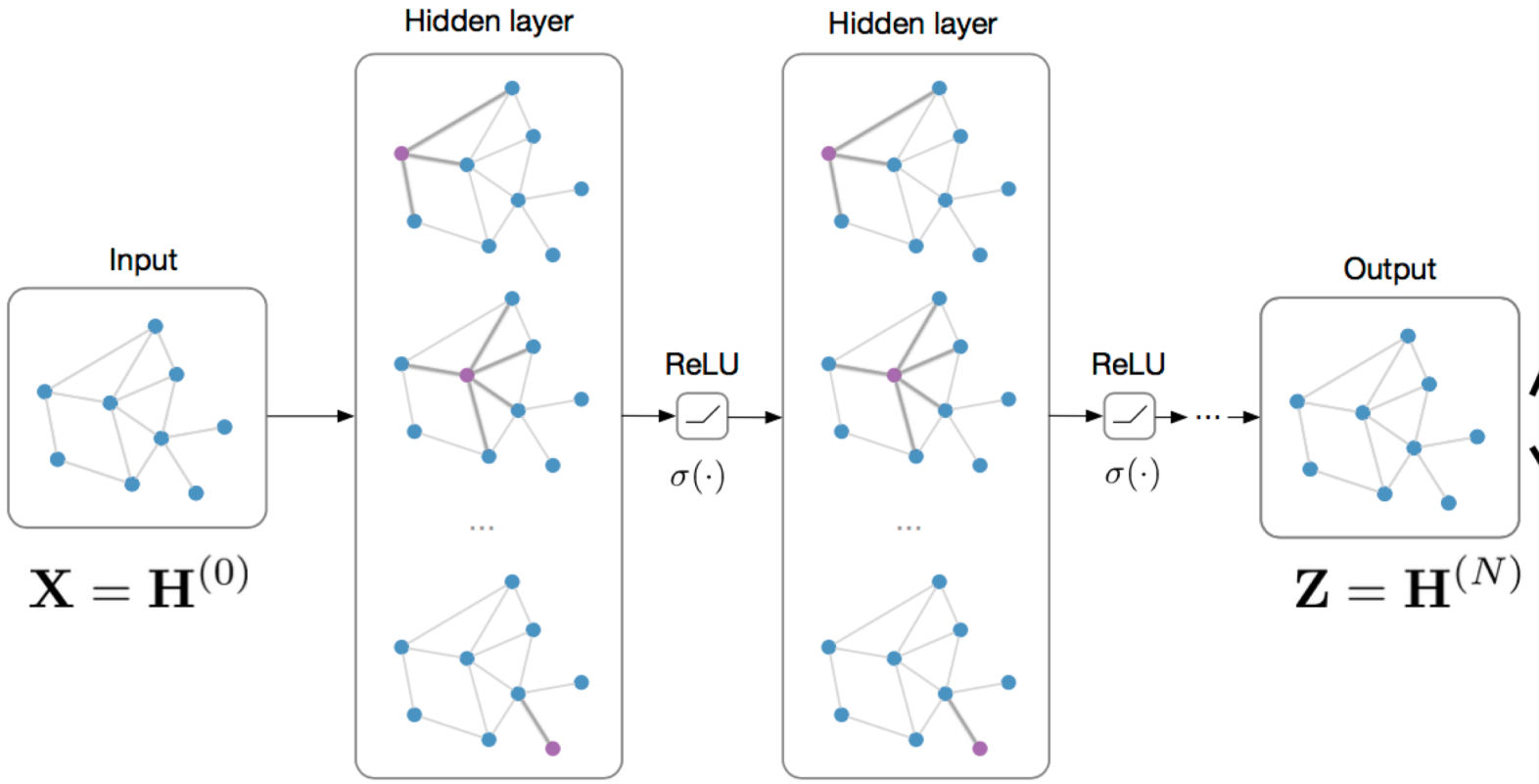


Update rule:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{h}_i^{(l)}\mathbf{W}_0^{(l)} + \sum_{j\in\mathcal{N}_i}\frac{1}{c_{ij}}\mathbf{h}_j^{(l)}\mathbf{W}_1^{(l)}\right)$$

Graph Convolutional networks (GCN)

Input: Feature matrix $\mathbf{X} \in \mathbb{R}^{N \times E}$, preprocessed adjacency matrix $\hat{\mathbf{A}}$



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

Node classification:

$\mathrm{softmax}(\mathbf{z_n})$

e.g. Kipf & Welling (ICLR 2017)

Graph classification:

$\mathrm{softmax}(\sum_n \mathbf{z_n})$

e.g. Duvenaud et al. (NIPS 2015)

Link prediction:

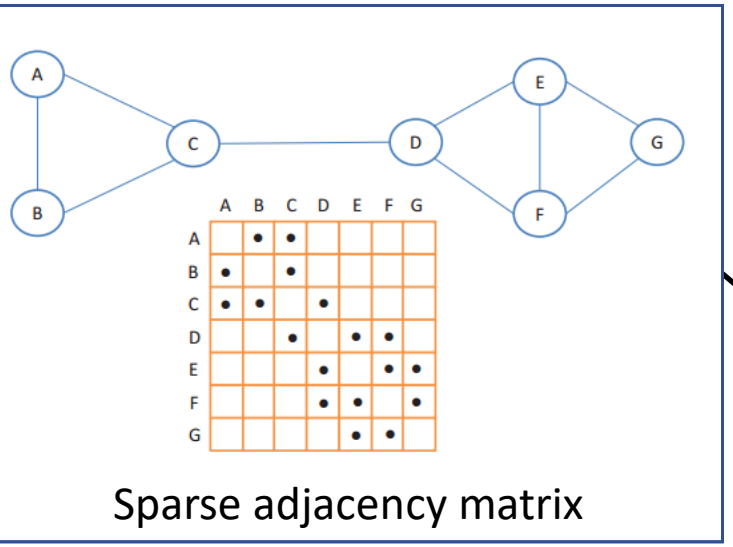
$p(A_{ij}) = \sigma(\mathbf{z_i^T z_j})$

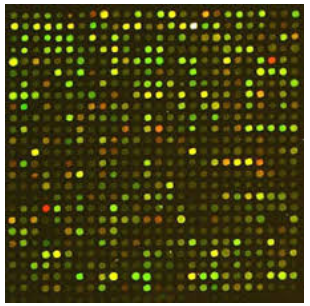
Kipf & Welling (NIPS BDL 2016)

"Graph Auto-Encoders"

Gene Network Embedding (GNE):

Deep model for gene interaction network inference



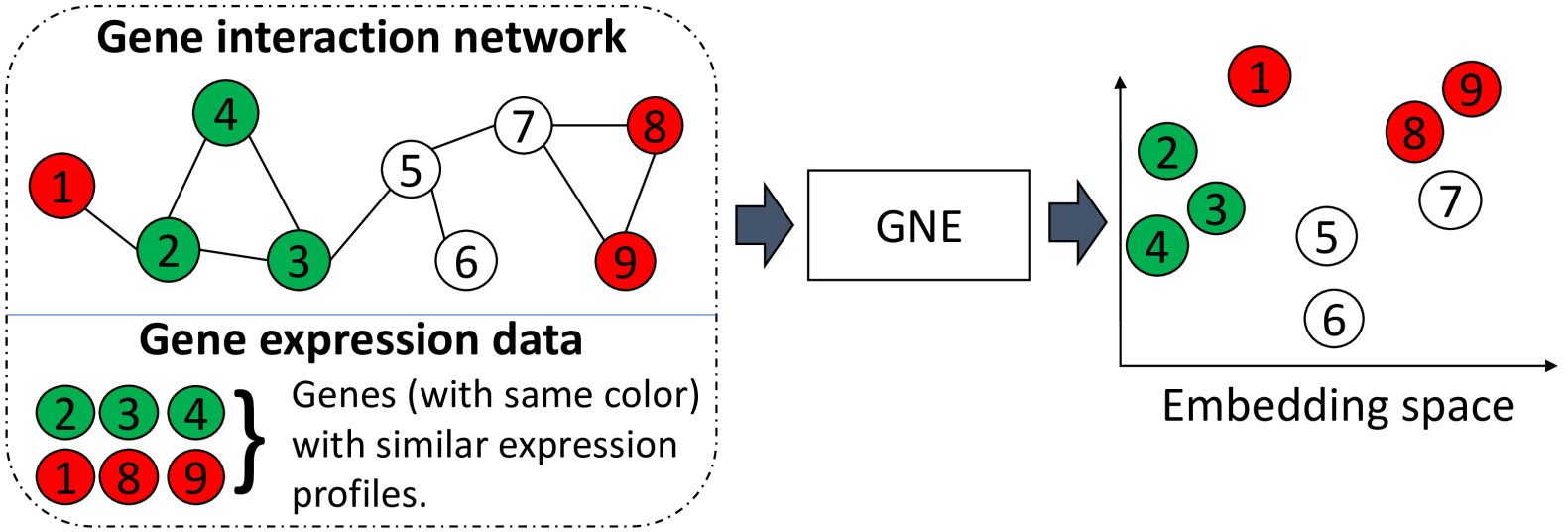


GNE

Gene interaction prediction

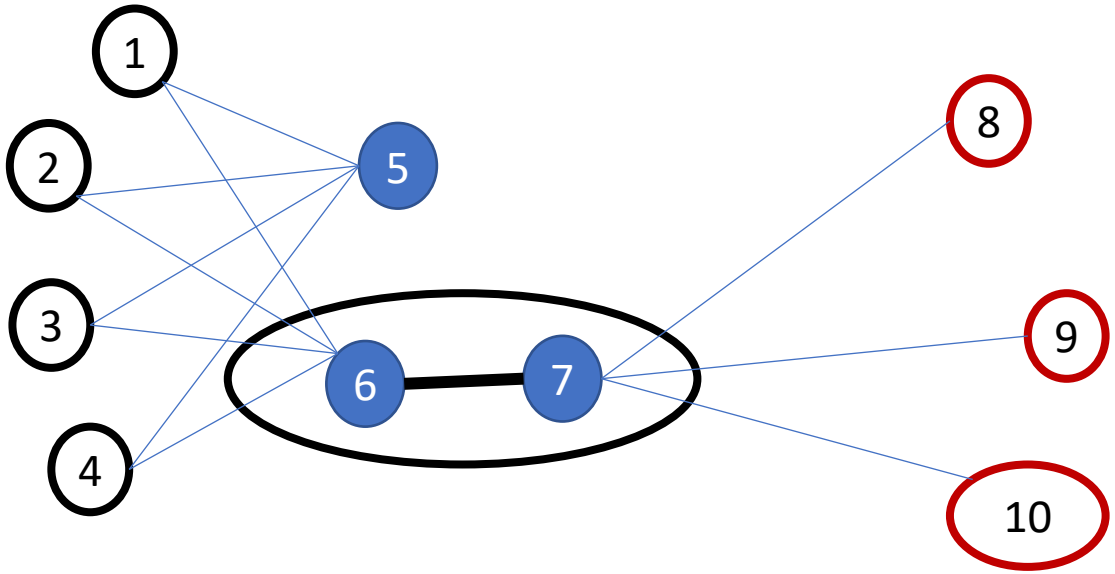
GNE Overview

Given a gene network denoted as $\boldsymbol{G = (V, E, A)}$, gene network embedding aims to learn a function $\boldsymbol{f}$ that maps gene network structure and their attribute information to a $\boldsymbol{d}$-dimensional space where a gene is represented by a vector $\boldsymbol{y_i \, \epsilon \, \mathbb{R}^d}$ where $\boldsymbol{d \ll M}$. The low dimensional vectors $\boldsymbol{y_i}$ and $\boldsymbol{y_j}$ for genes $\boldsymbol{v_i}$ and $\boldsymbol{v_j}$ preserve their relationships in terms of the network topological structure and attribute proximity.



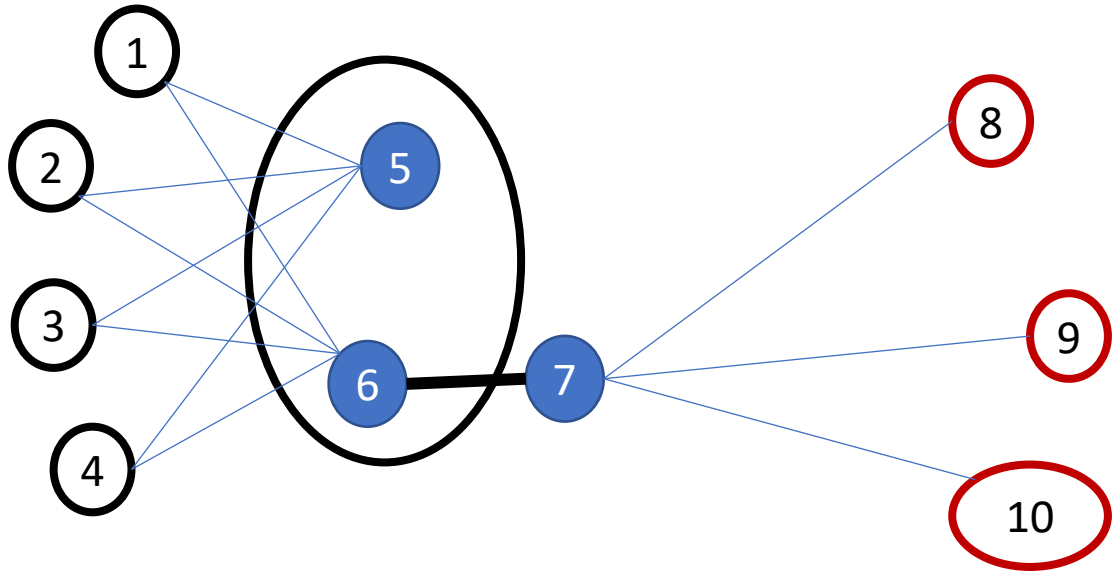
Background: First Order Proximity

- Pairwise proximity between vertices
- For any pair of vertices,
  - If $A_{i,j} > 0$, first order proximity between v_i and v_j is positive
  - Otherwise, first order proximity between v_i and v_j is 0



Background: Second Order Proximity

- Proximity of the pair's neighborhood structure
- N_u is neighborhood of vertex u and N_v is neighborhood of vertex v
- Similarity between N_u and N_v gives second order proximity



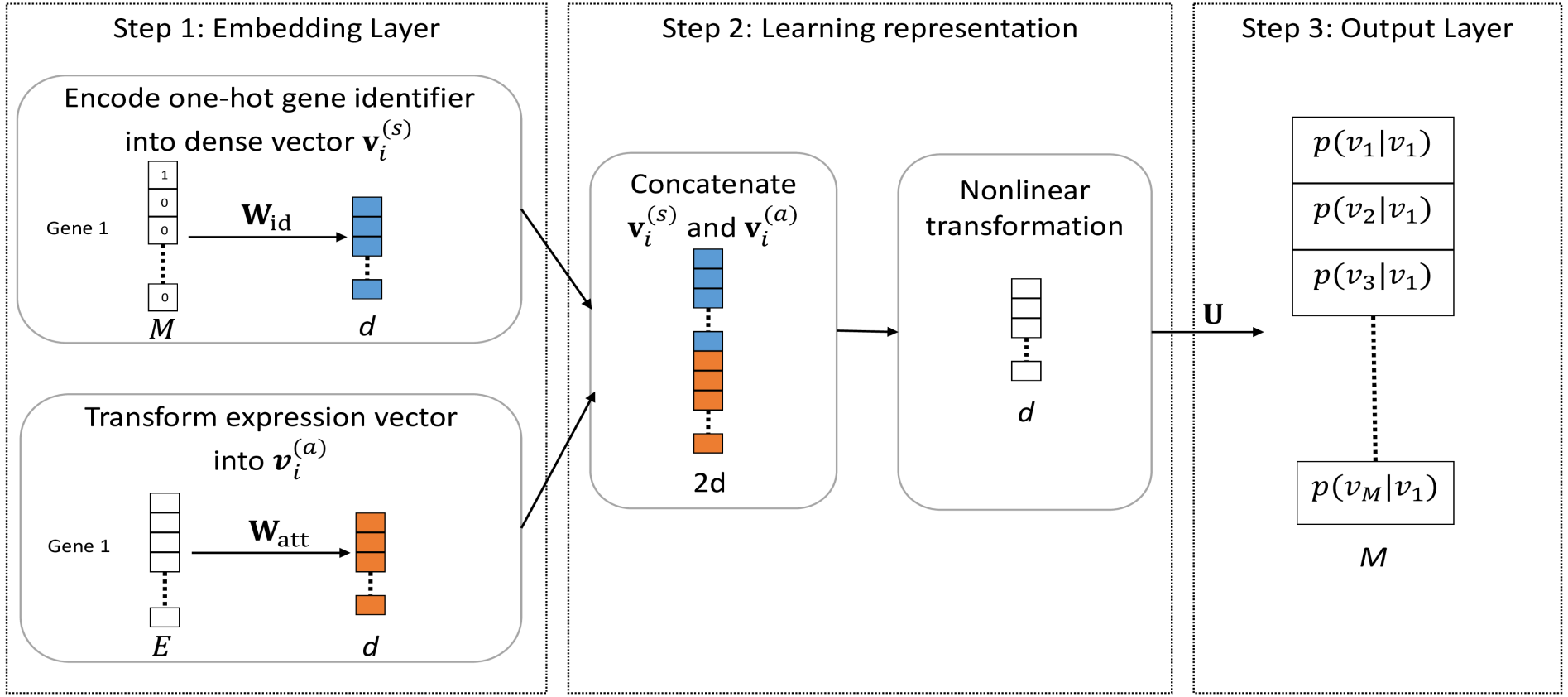
Background: Attribute Proximity

- Similarity in attributes between vertices u and v
- Vertices with similar attributes will be placed close to each other in embedding space

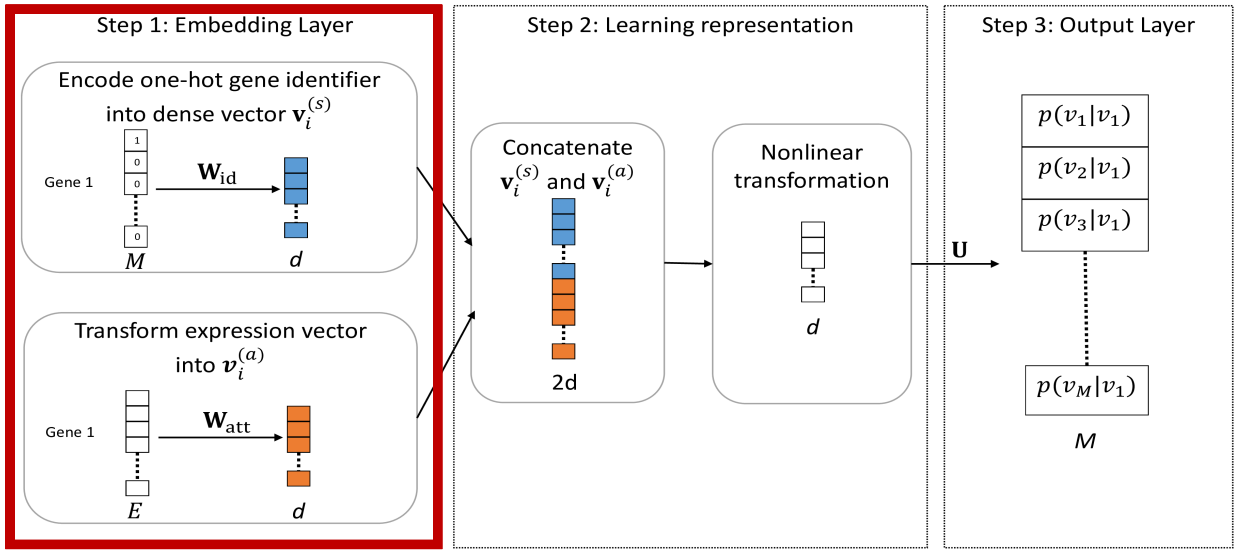
- Example

Vertex	Place	Major	College	Gender
A	Nepal	CS	RIT	Male
B	Germany	HCI	UR	Female
C	Nepal	SE	RIT	Male
D	USA	IMGS	MIT	Male

GNE Architecture



GNE: Embedding



GNE Network Structure Modeling

Encode one-hot encoded representation of a gene v_i via embedding lookup.

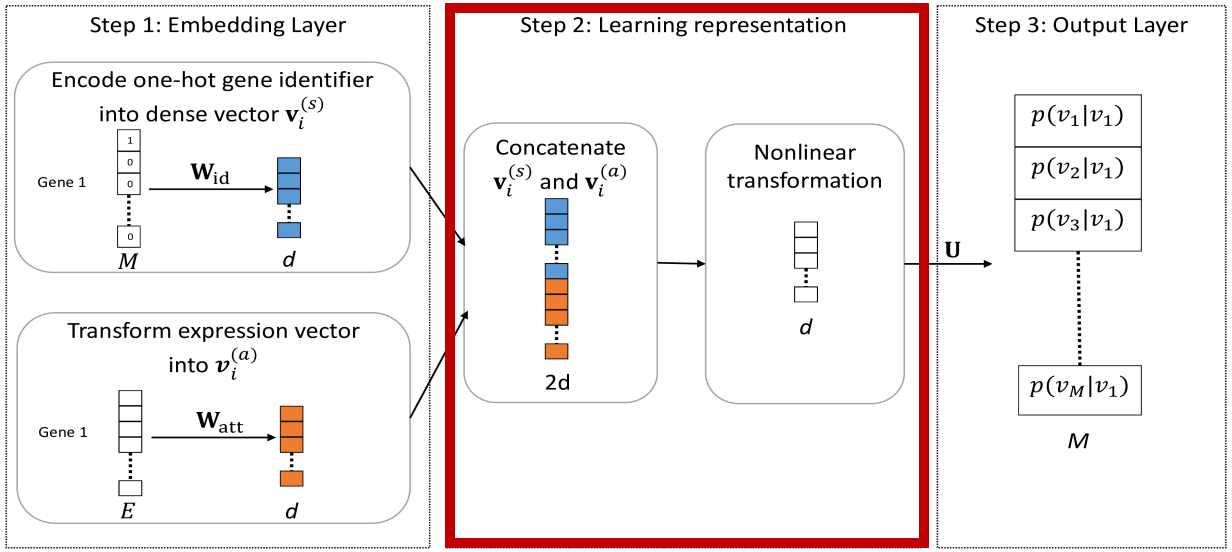
$$\mathbf{v}_i^{(s)} = \boldsymbol{W_{id}}\, v_i$$

GNE Expression Modeling

Exponential Linear unit (ELU) to model non-linearity of gene expression x_i and capture underlying patterns.

$$\mathbf{v}_i^{(a)} = \text{elu}(\ \boldsymbol{W_{att}} \cdot x_i)$$

GNE: Learning representation



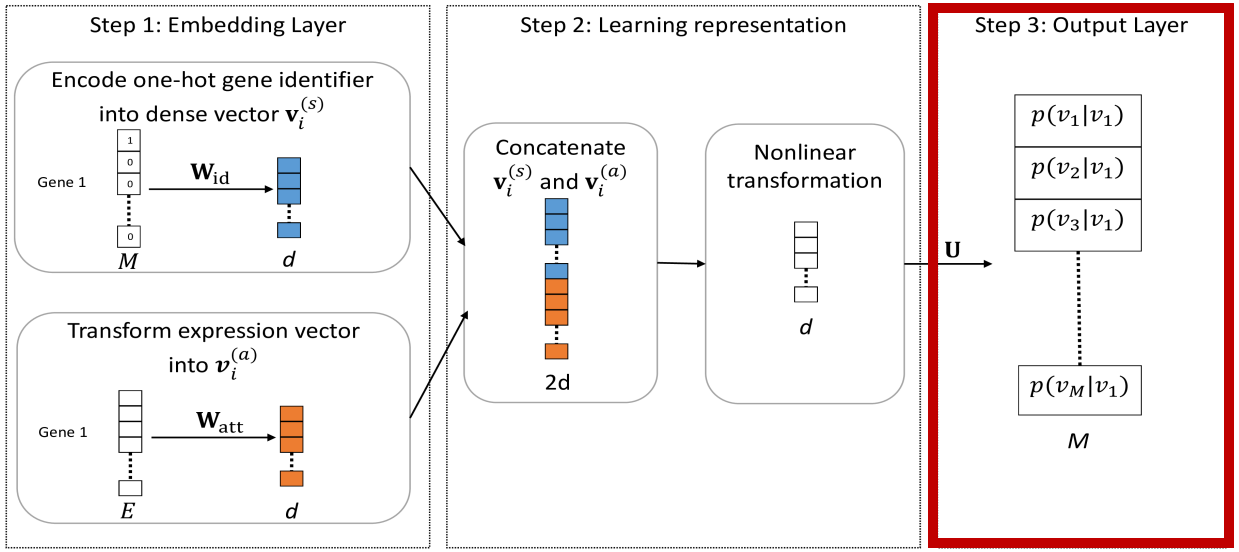
Concatenation of structural and attribute representation

$$\mathbf{v}_i = [\mathbf{v}_i^{(s)} \quad \lambda \mathbf{v}_i^{(a)}]$$

Transformation of concatenated representation via $\boldsymbol{k}$-hidden layers with hyperbolic tangent activation.

$$\mathbf{h}_i^{(k)} = \delta_k(\boldsymbol{W_k}\, \mathbf{h}_i^{(k-1)} + b^{(k)})$$

GNE: Predicting probabilities



Last layer outputs the probability vector which contains conditional probability of all other genes to gene v_i

$$\mathbf{o}_i = [p(v_1|v_i), p(v_2|v_i), \ldots, p(v_M|v_i)]$$

where

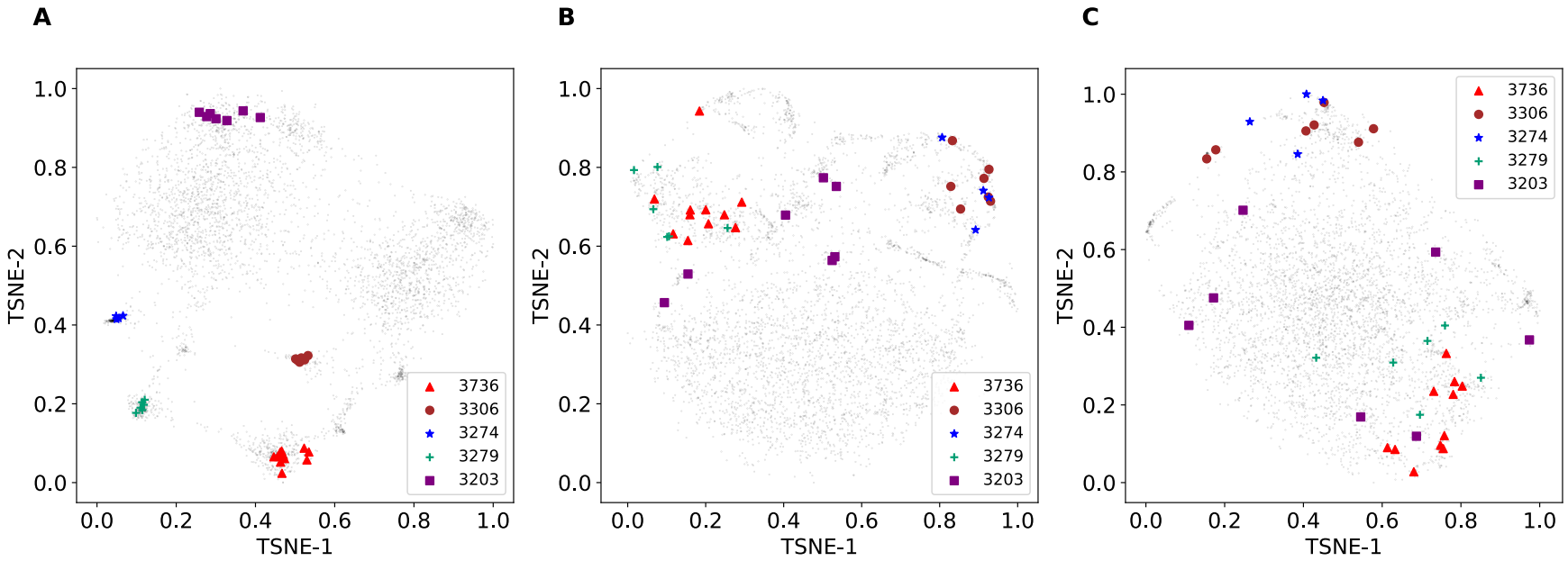
$$p(v_j|v_i) = \frac{\exp(\widetilde{\mathbf{v}}_j \cdot \mathbf{h}_i^{(k)})}{\sum_{j'=1}^{M} \exp(\widetilde{\mathbf{v}}_{j'} \cdot \mathbf{h}_i^{(k)})}$$

Optimization:

$$\Theta^* = \underset{\Theta}{\operatorname{argmax}} \left[\sum_{i=1}^{M} \sum_{v_j \in N_i} \log \frac{\exp(\widetilde{\mathbf{v}}_j \cdot \mathbf{h}_i^{(k)})}{\sum_{j'=1}^{M} \exp(\widetilde{\mathbf{v}}_{j'} \cdot \mathbf{h}_i^{(k)})} \right]$$

Visualizing the embeddings

- Visualize embeddings on 2D space using t-SNE package
- **Operons**: genes that interact with each other and are co-regulated.
  - Colored the points in 2D space with operons



GNE LINE node2vec

- Significant test to see if genes within same operons are likely to have similar representation

Comparison with other methods

- Randomly removed 50% of interactions as test set to predict missing interactions
- Experimental results with and without expression data

Methods	**Yeast**		**E. coli**	
	AUROC	**AUPR**	**AUROC**	**AUPR**
Isomap	0.507	0.588	0.559	0.672
LINE	0.726	0.686	0.897	0.851
node2vec	0.739	0.708	0.912	0.862
GNE*	0.787	0.784	0.930	0.931
GNE	**0.825**	**0.821**	**0.940**	**0.939**

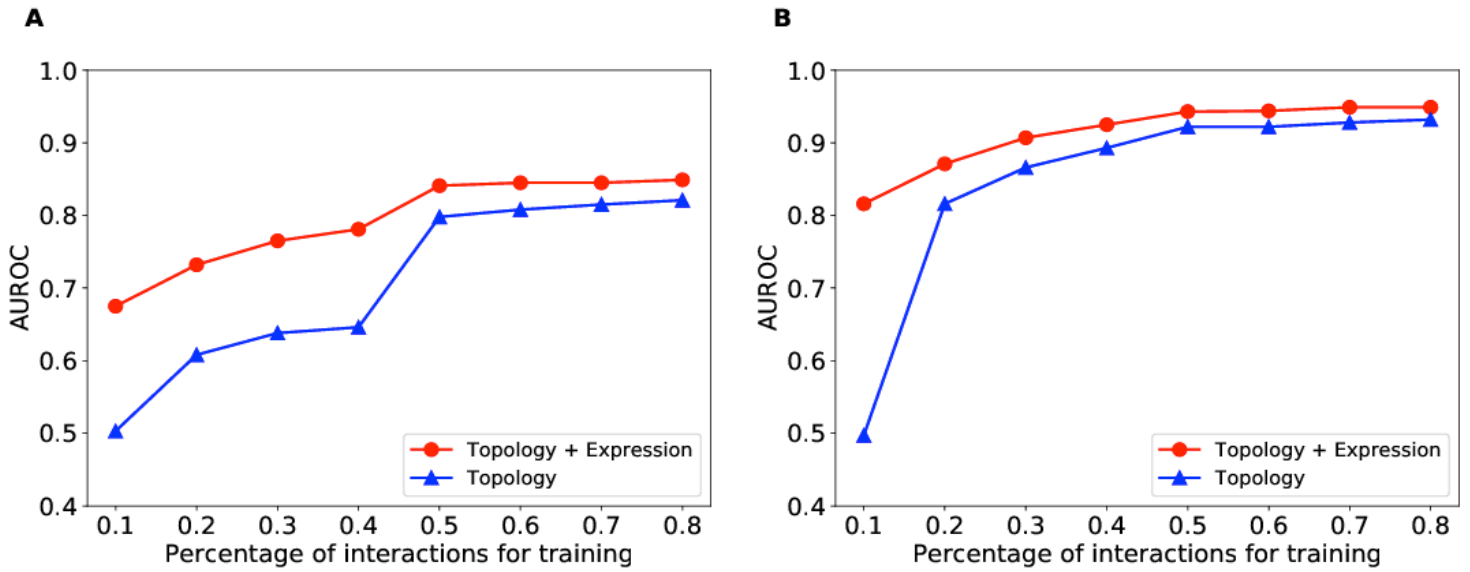
Temporal holdout validation

- Two version of interaction dataset: 2017 and 2018 version
  - 2018 version has **12,835** new interactions for yeast and **11,185** new interactions for E. coli
- Randomly selected 50% of interactions from 2017 version as training data to predict new interactions in 2018 version
- Experimental results with and without expression data

Methods	**Yeast**		**E. coli**	
	AUROC	**AUPR**	**AUROC**	**AUPR**
LINE	0.620	0.611	0.569	0.598
node2vec	0.640	0.609	0.587	0.599
GNE	**0.710**	**0.683**	**0.653**	**0.658**

Impact of network sparsity

- Hold out 10% interactions as test dataset
- Change the sparsity of training data by randomly removing a portion of remaining interactions
- Evaluation with and without expression data

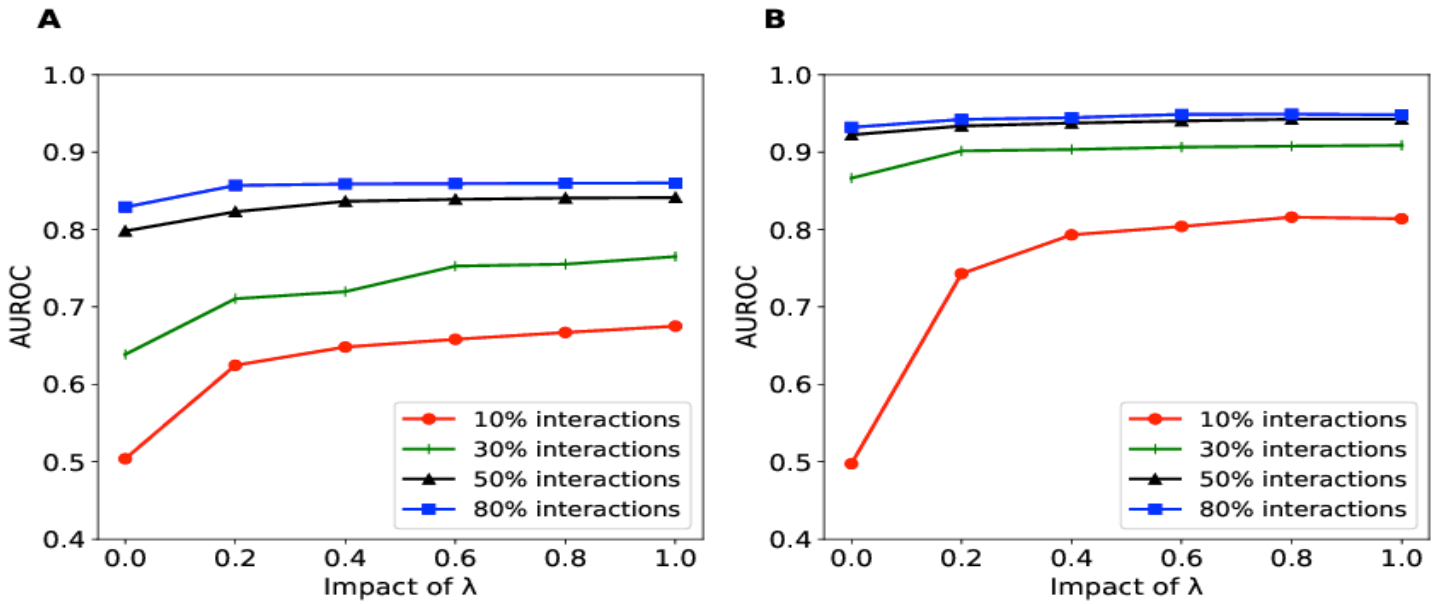


Yeast

E. coli

Relative importance of topology and expression data

- Evaluation of parameter λ to see the impact on model's performance
- Values of λ used in experiment: [0, 0.2, 0.4, 0.6, 0.8, 1, 10, 100, 1000]
- Integration of expression data improves model's performance



Yeast

E. coli

Future works

- Integration of multi-omics data
- Exploring advanced graph-based deep learning approach for gene network inference

Any questions?