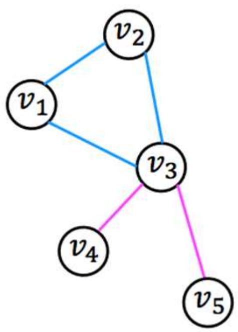
From Graph to Knowledge Graph: Algorithms and Applications

Module 3: Graph Representation Learning

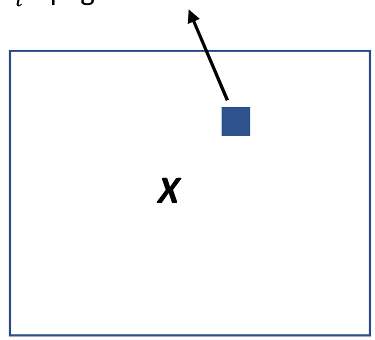
Module 3: Graph Representation Learning

- Representation learning
- Skip-gram based graph representation learning
  - Homogeneous network embedding
  - Understanding network embedding
  - Heterogeneous network embedding
- Deep learning for graph representation learning
  - Graph convolutional networks

The graph mining paradigm so far



x_{ij}: node v_i's j^{th} feature, e.g., v_i's pagerank value



hand-crafted feature matrix

Graph & network applications

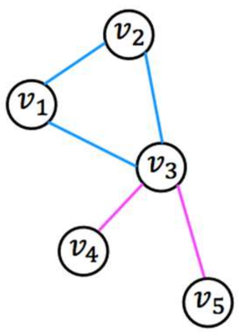
- Node label inference;
- Node clustering;
- Link prediction;
- … …

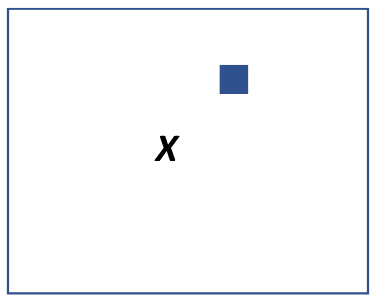
feature **engineering**

machine learning models

- **Classification algorithms,** such as logistic regression, SVM, and random forest;
- **Regression algorithms,** such as linear regression;
- **Clustering algorithms**, such as k-means.

Representation learning for graph mining?





~~hand-crafted~~ **latent** feature matrix

Graph & network applications

- Node label inference;
- Node clustering;
- Link prediction;
- … …

Feature ~~engineering~~ **learning**

machine learning models

- **Classification algorithms,** such as logistic regression, SVM, and random forest;
- **Regression algorithms,** such as linear regression;
- **Clustering algorithms**, such as k-means.

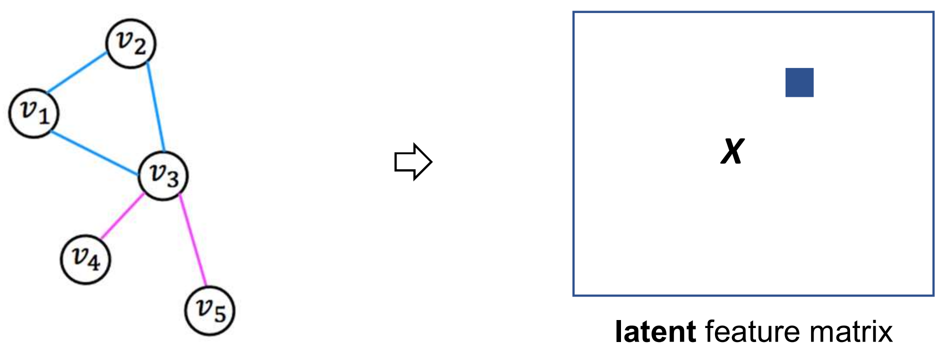
- Bengio, Courville, Vincent. Representation learning: A review and new perspectives. *IEEE TPAMI* 2013.
- LeCun, Bengio, Hinton. Deep learning. *Nature,* 521(7553):436–444, 2015.

Representation learning for graph mining?

Problem (Graph representation learning, network embedding, graph embedding)

- Input: a network $G = (V, E)$
- Output: $\boldsymbol{X} \in R^{|V| \times k}, k \ll |V|$, k-dim vector $\boldsymbol{X}_v$ for each node *v*.

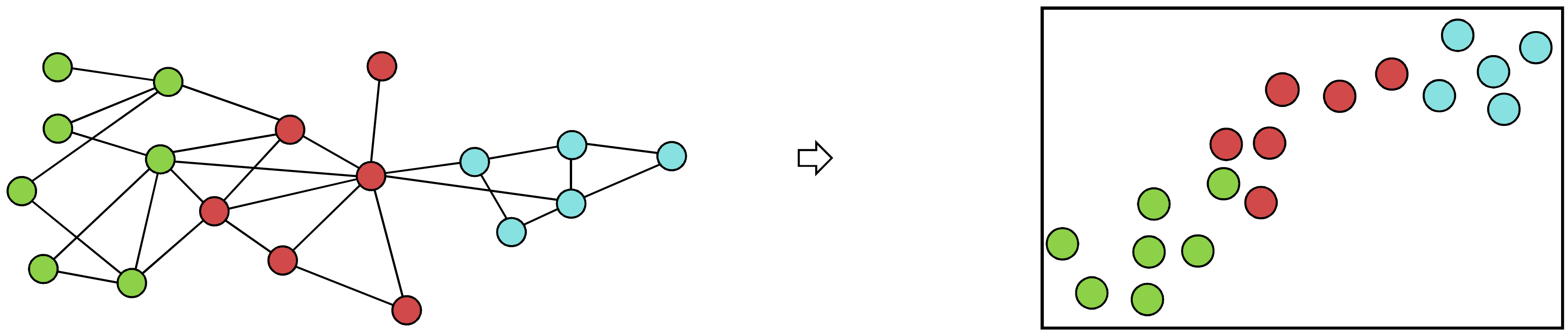
The goal is to map each node into a latent low-dimension space such that network structure information is encoded into distributed node representations



latent feature matrix

Feature **learning**

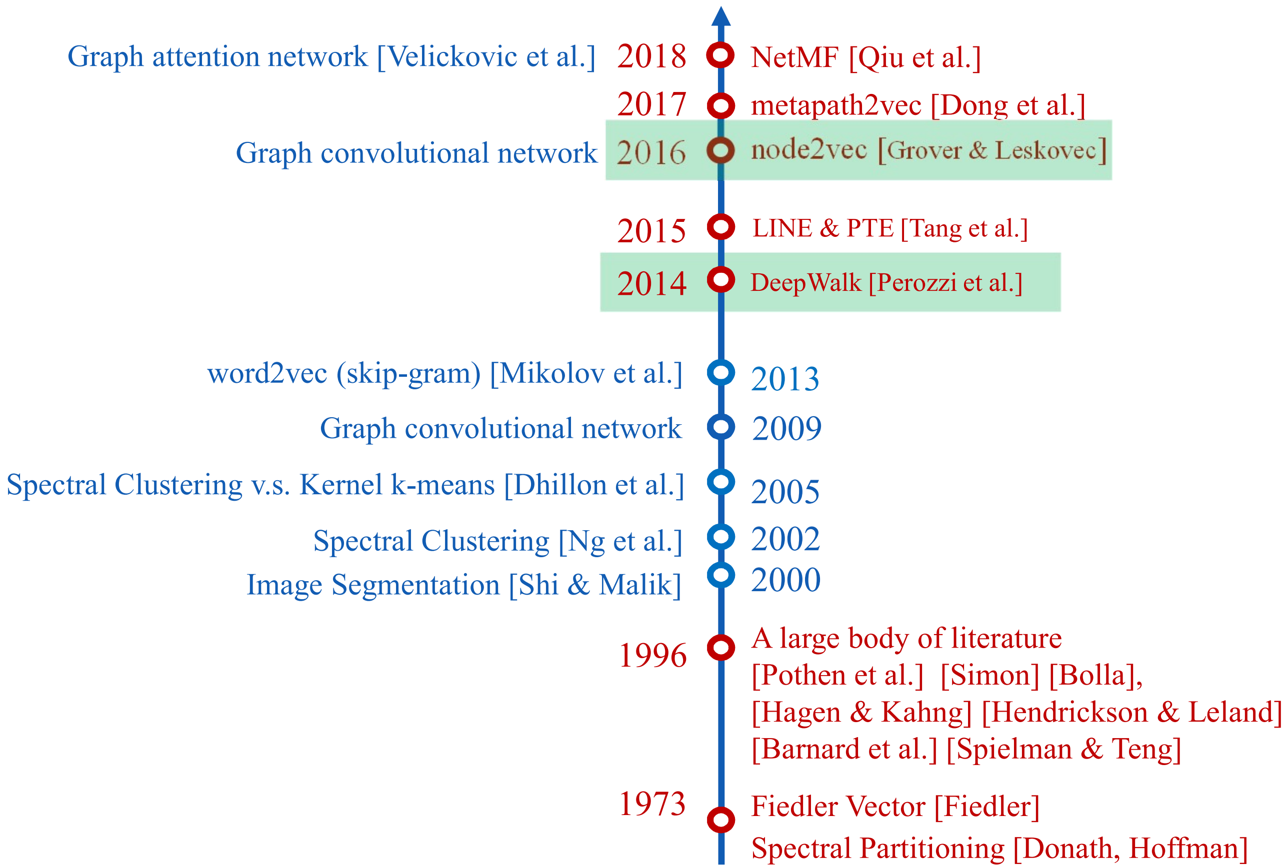
Graph representation learning



A brief history of graph embedding

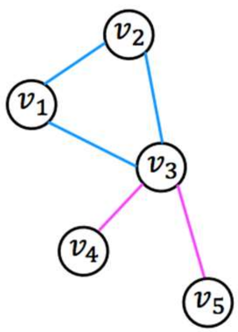
- 2018 — Graph attention network [Velickovic et al.]; NetMF [Qiu et al.]
- 2017 — metapath2vec [Dong et al.]
- 2016 — Graph convolutional network; node2vec [Grover & Leskovec]
- 2015 — LINE & PTE [Tang et al.]
- 2014 — DeepWalk [Perozzi et al.]
- 2013 — word2vec (skip-gram) [Mikolov et al.]
- 2009 — Graph convolutional network
- 2005 — Spectral Clustering v.s. Kernel k-means [Dhillon et al.]
- 2002 — Spectral Clustering [Ng et al.]
- 2000 — Image Segmentation [Shi & Malik]
- 1996 — A large body of literature [Pothen et al.] [Simon] [Bolla], [Hagen & Kahng] [Hendrickson & Leland] [Barnard et al.] [Spielman & Teng]
- 1973 — Fiedler Vector [Fiedler]; Spectral Partitioning [Donath, Hoffman]

A brief history of graph embedding



Graph embedding

- Input: a graph $G = (V, E)$
- Output: $\boldsymbol{X} \in R^{|V| \times k}, k \ll |V|$, k-dim vector $\boldsymbol{X}_v$ for each node v.



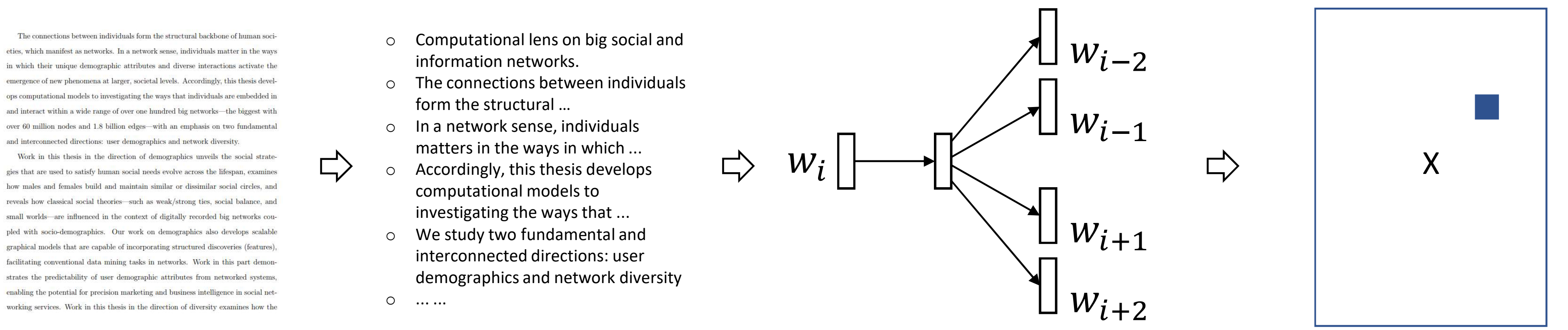
?

Feature **learning**

latent feature matrix

Word embedding in NLP

- Input: a text corpus $D = \{W\}$
- Output: $\boldsymbol{X} \in R^{|W| \times d}, d \ll |W|$, d-dim vector $\boldsymbol{X}_w$ for each word *w.*



sentences

Word embedding models

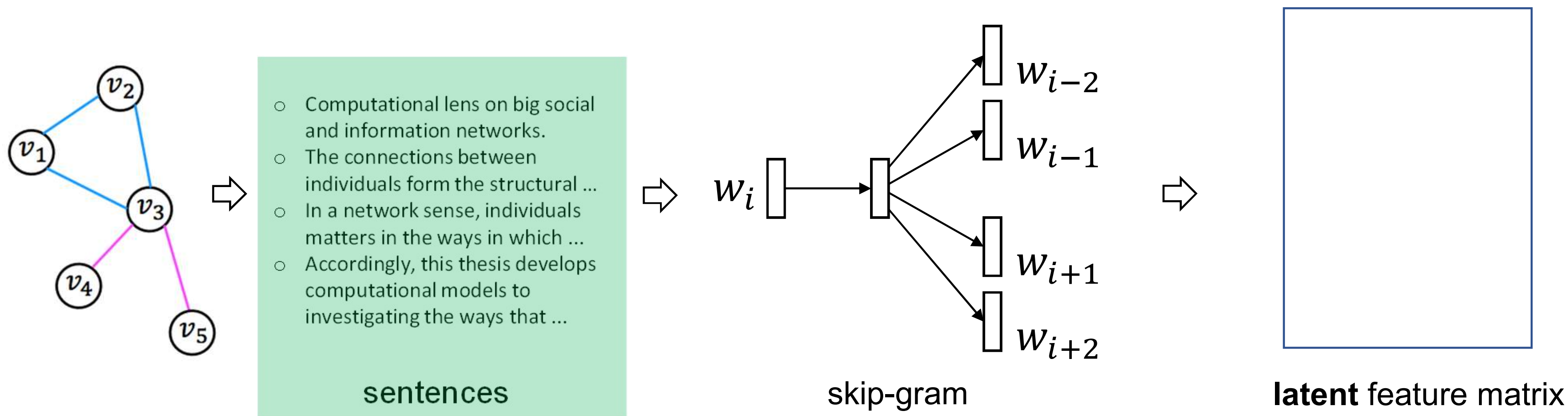
latent feature matrix

- Basic assumption: geographically close words---a word and its context words---in a sentence or document exhibit interrelations in human natural language.
- Key idea: try to predict the words that surrounding each one.

- Bengio, et al. Representation learning: A review and new perspectives. In *IEEE TPAMI 2013.*
- Mikolov, et al. Efficient estimation of word representations in vector space. In *ICLR 2013*.

Graph embedding

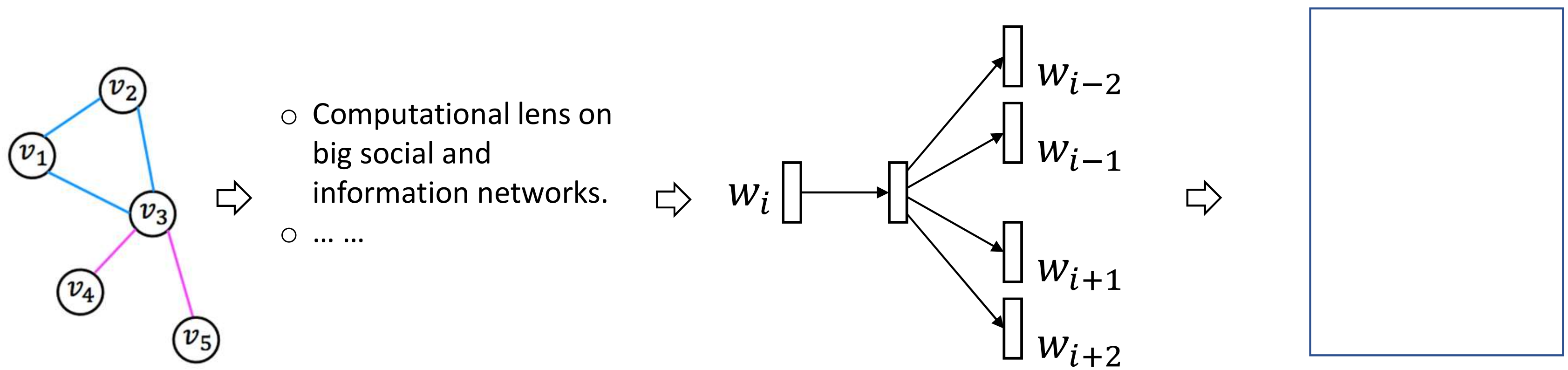
- Input: a network $G = (V, E)$
- Output: $\boldsymbol{X} \in R^{|V| \times k}, k \ll |V|$, k-dim vector $\boldsymbol{X}_v$ for each node v.



Feature **learning**

Graph embedding

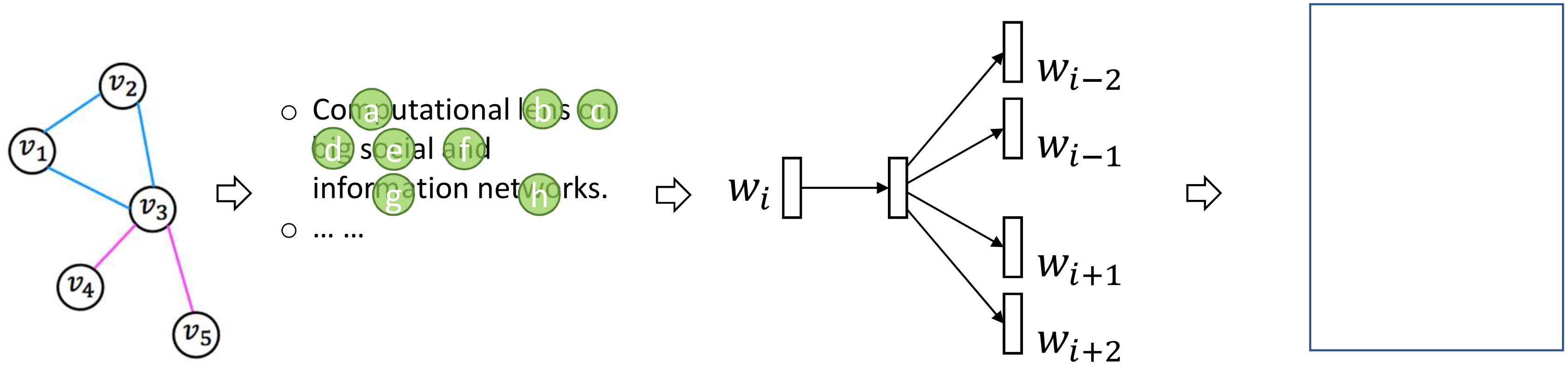
- Input: a network $G = (V, E)$
- Output: $\boldsymbol{X} \in R^{|V| \times k}, k \ll |V|$, k-dim vector $\boldsymbol{X}_v$ for each node v.



Feature ~~engineering~~ **learning**

Graph embedding

- Input: a network $G = (V, E)$
- Output: $\boldsymbol{X} \in R^{|V| \times k}, k \ll |V|$, k-dim vector $\boldsymbol{X}_v$ for each node v.



sentences

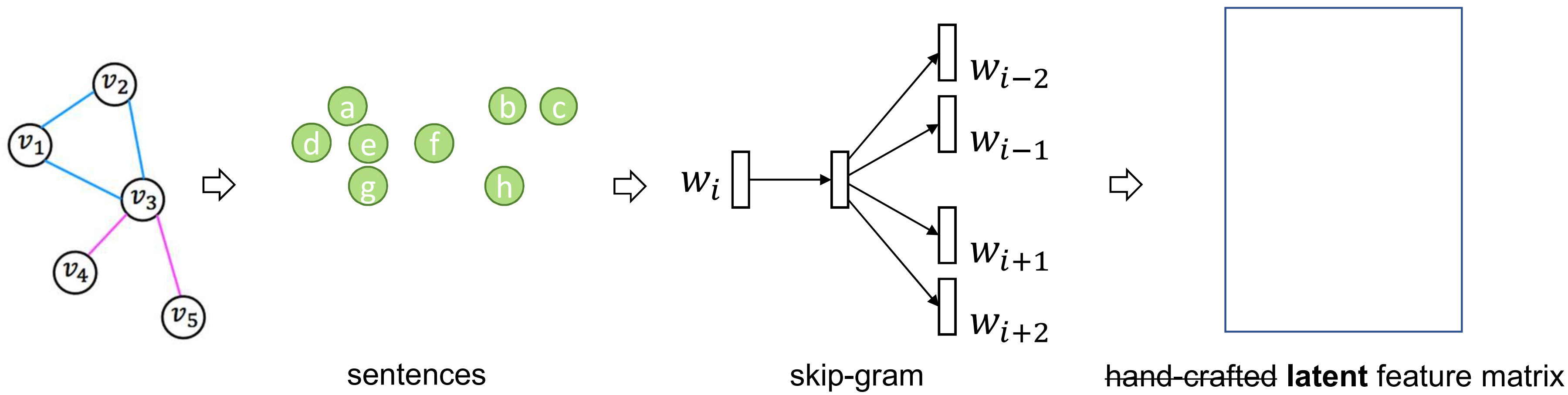
skip-gram

~~hand-crafted~~ **latent** feature matrix

Feature ~~engineering~~ **learning**

Graph embedding

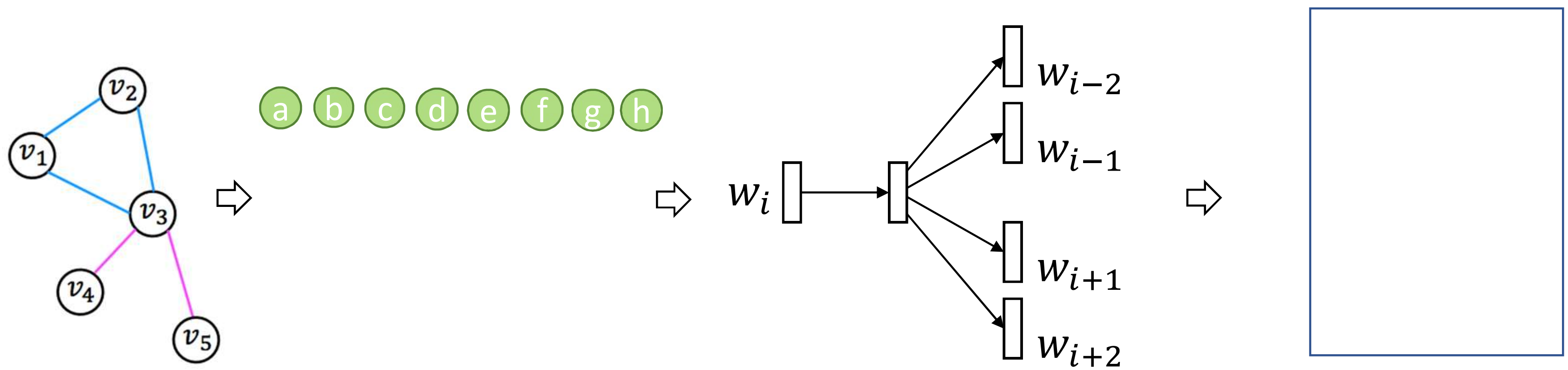
- Input: a network $G = (V, E)$
- Output: $\boldsymbol{X} \in R^{|V| \times k}, k \ll |V|$, k-dim vector $\boldsymbol{X}_v$ for each node v.



Feature ~~engineering~~ **learning**

Graph embedding

- Input: a network $G = (V, E)$
- Output: $\boldsymbol{X} \in R^{|V| \times k}, k \ll |V|$, k-dim vector $\boldsymbol{X}_v$ for each node v.

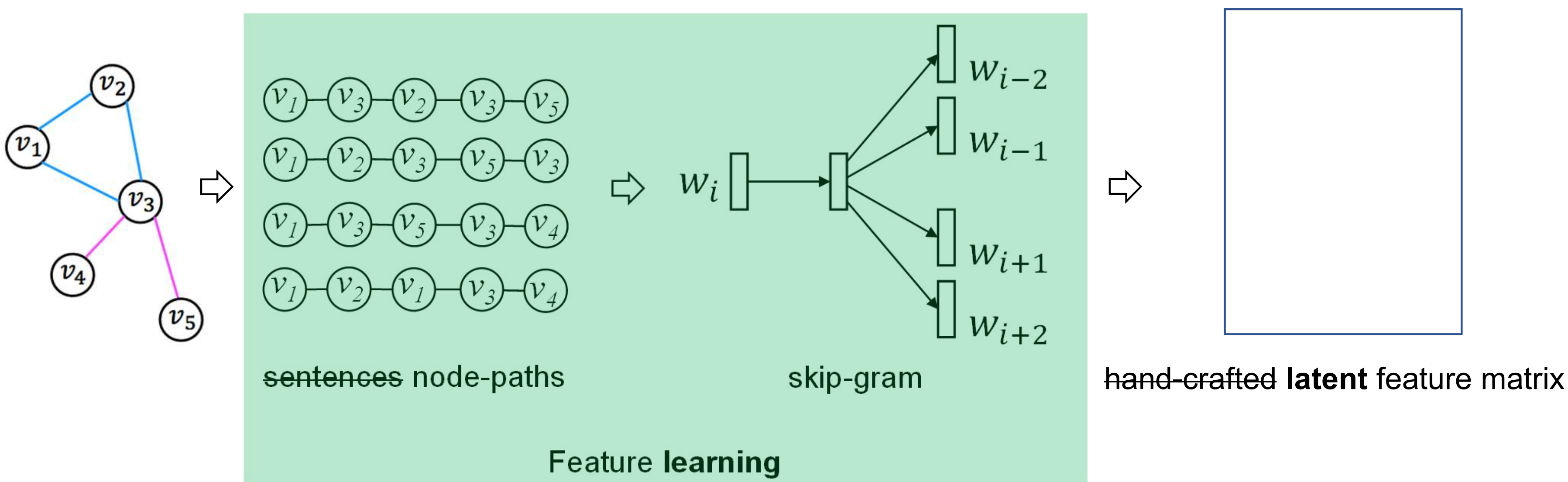


~~sentences~~ node-paths | skip-gram | ~~hand-crafted~~ **latent** feature matrix

Feature ~~engineering~~ **learning**

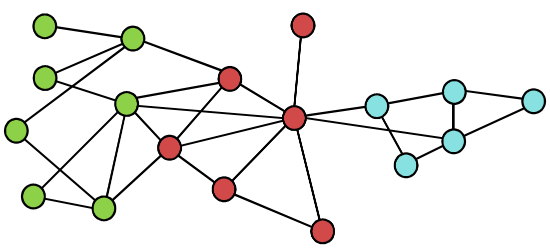
Graph embedding: DeepWalk & node2vec

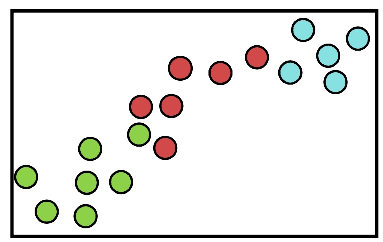
- Input: a network $G = (V, E)$
- Output: $\boldsymbol{X} \in R^{|V| \times k}, k \ll |V|$, k-dim vector $\boldsymbol{X}_v$ for each node v.



- Perozzi et al. DeepWalk: Online learning of social representations. In *KDD' 14*, pp. 701–710.
- Grover and Leskovec. node2vec: Scalable Feature Learning for Networks. in *KDD '16*, pp. 855—864.

Graph embedding: DeepWalk & node2vec

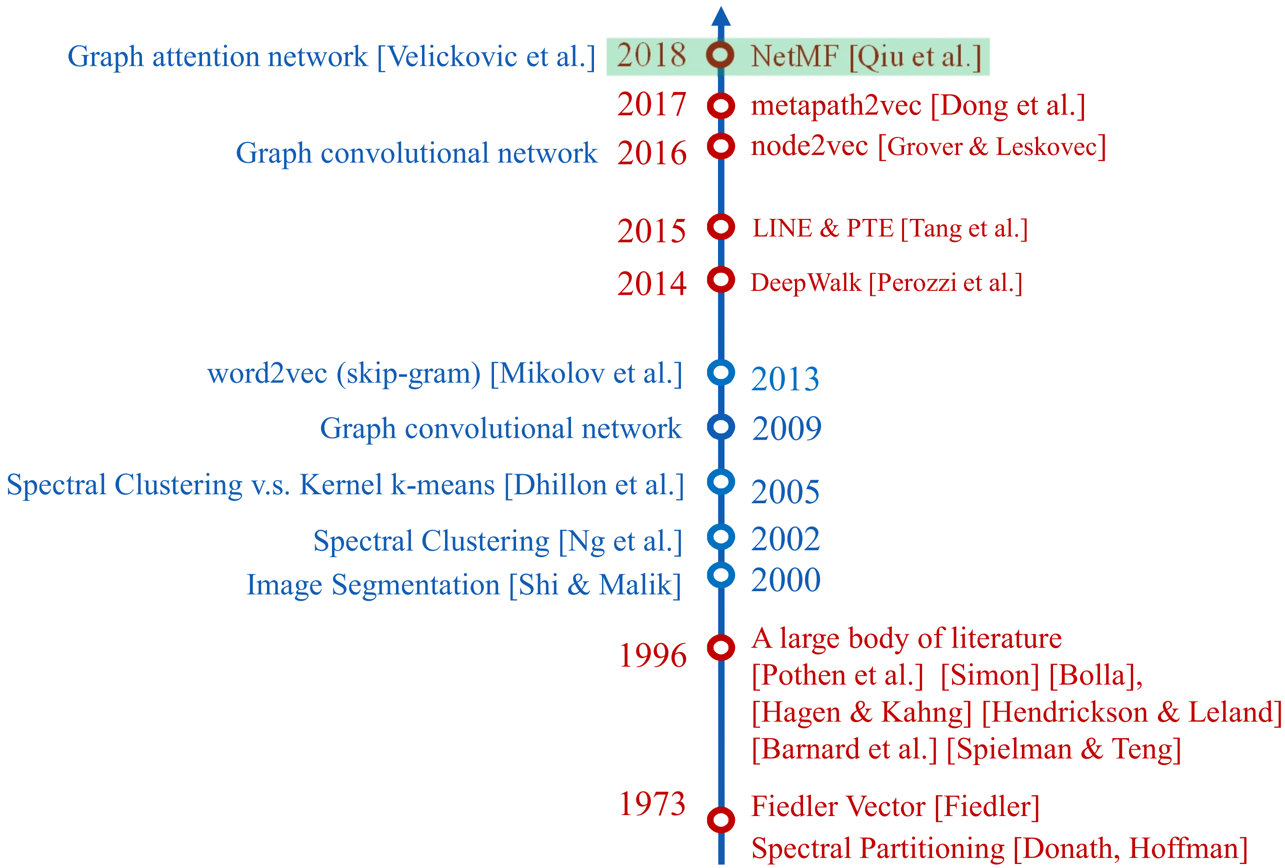




Graph & network applications

- Node label inference;
- Node clustering;
- Link prediction;
- ... ...

A brief history of graph embedding



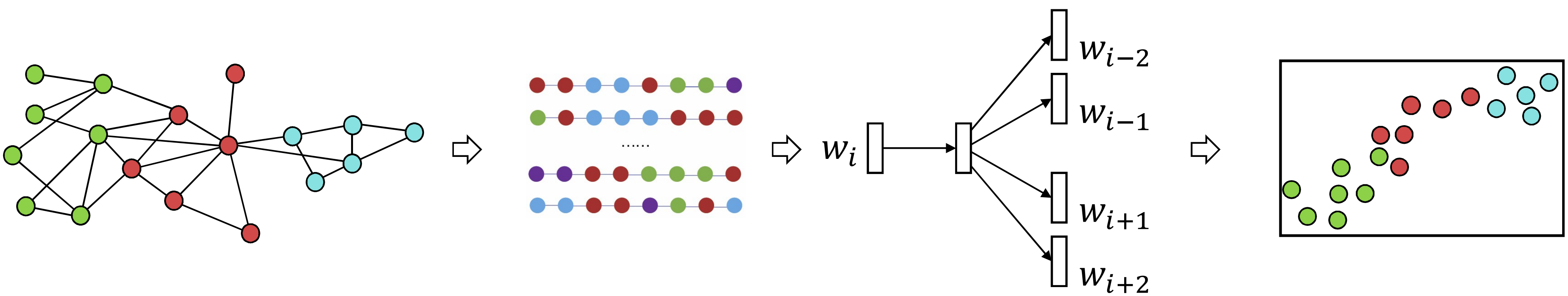
Skip-gram based graph embedding

- Input: an undirected weighted network $G = (V, E)$ with $|V| = n$ & $|E| = \mathrm{m}$
  - Adjacency matrix $\boldsymbol{A} \in \mathbb{R}_{+}^{n\times \mathrm{n}}$

$$A_{i,j} = \begin{cases} a_{i,j} > 0 & (i,j) \in E \\ 0 & (i,j) \notin E \end{cases}$$

  - Degree matrix $\boldsymbol{D} = diag(d_1, d_2, \cdots, d_n)$
  - Volume of G: $vol(G) = \sum_i \sum_j A_{ij}$
- Output: for each node, its k-dimension latent feature representation vector $\mathbf{Z}^{n\times k}$
  - Latent feature embedding matrix $\mathbf{Z} \in \mathbb{R}^{n\times k}$

Understanding random walk + skip gram



Skip gram with negative sampling

Skip-gram with negative sampling (SGNS)

- SGNS maintains a multiset $\boldsymbol{\mathcal{D}}$ that counts the occurrence of each word-context pair (w, c)
- Objective

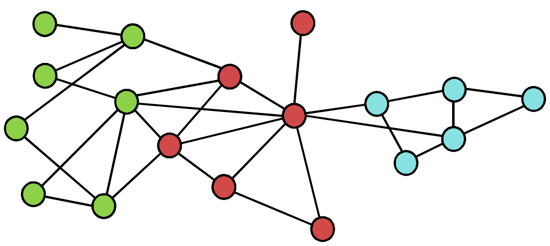
$$\mathcal{L} = \sum_{w} \sum_{c} (\#(w, c) \log g(x_w^T x_c) + \frac{b\#(w)\#(c)}{|\mathcal{D}|} \log g(-x_w^T x_c))$$

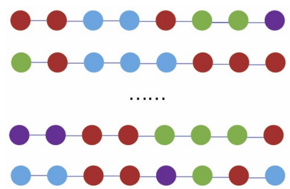
- For sufficiently large dimension *d*, the objective above is equivalent to factorizing the PMI matrix

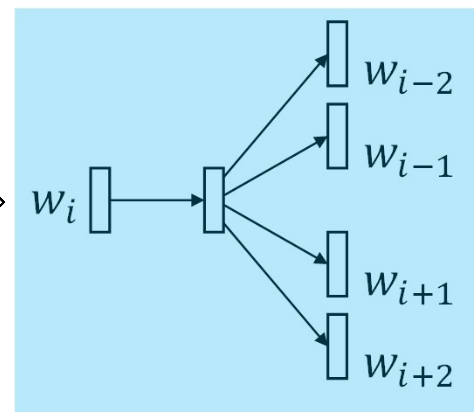
$$\log \frac{\#(w, c)|\mathcal{D}|}{b\#(w)\#(c)}$$

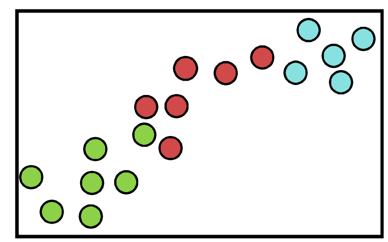
Levy and Goldberg. Neural word embeddings as implicit matrix factorization. In *NIPS 2014*

Understanding random walk + skip gram









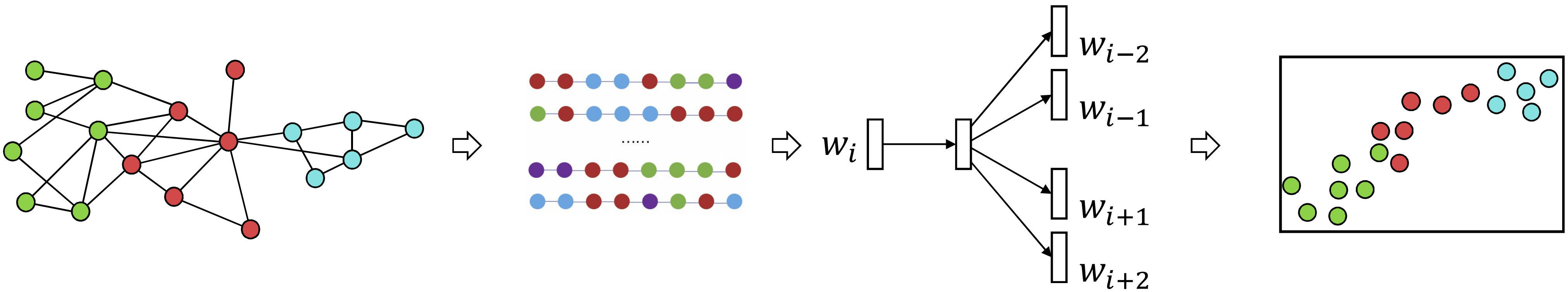
$G = (V, E)$

- Adjacency matrix $\boldsymbol{A}$
- Degree matrix $\boldsymbol{D}$
- Volume of G: $vol(G)$

?

$$\log\left(\frac{\#(w,c)|\mathcal{D}|}{b\#(w)\#(c)}\right)$$

Understanding random walk + skip gram



$G = (V, E)$

- Adjacency matrix $\boldsymbol{A}$
- Degree matrix $\boldsymbol{D}$
- Volume of G: $vol(G)$

?

$$\log\left(\frac{\#(w,c)|\mathcal{D}|}{b\#(w)\#(c)}\right)$$

Understanding random walk + skip gram

$$\log\left(\frac{\#(w,c)\,|\mathcal{D}|}{b\#(w)\cdot\#(c)}\right) = \log\left(\frac{\frac{\#(w,c)}{|\mathcal{D}|}}{b\frac{\#(w)}{|\mathcal{D}|}\frac{\#(c)}{|\mathcal{D}|}}\right)$$

$$\frac{\#(w,c)}{|\mathcal{D}|} = \frac{1}{2T}\sum_{r=1}^{T}\left(\frac{\#(w,c)_{\overrightarrow{r}}}{|\mathcal{D}_{\overrightarrow{r}}|} + \frac{\#(w,c)_{\overleftarrow{r}}}{|\mathcal{D}_{\overleftarrow{r}}|}\right)$$

$$\frac{\#(w,c)_{\overrightarrow{r}}}{|\mathcal{D}_{\overrightarrow{r}}|} \xrightarrow{p} \frac{d_w}{\mathrm{vol}(G)}\left(\boldsymbol{P}^r\right)_{w,c}$$

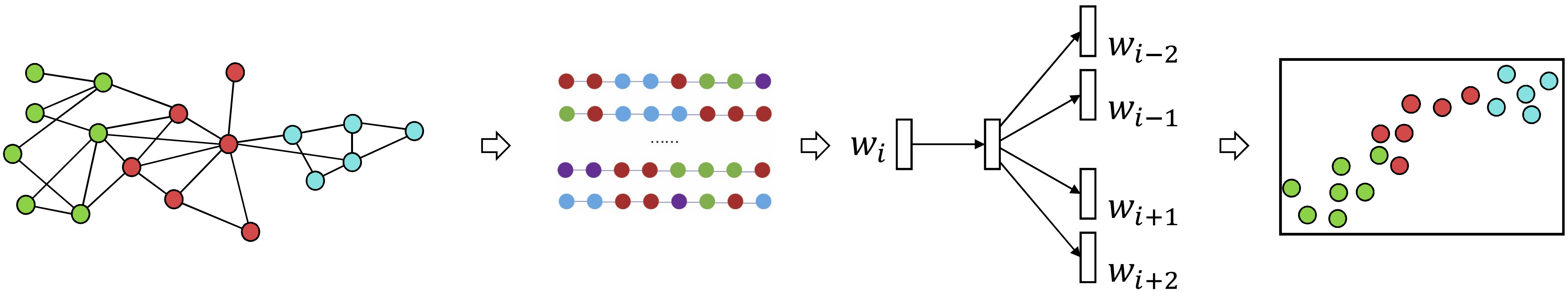
$$\frac{\#(w,c)_{\overleftarrow{r}}}{|\mathcal{D}_{\overleftarrow{r}}|} \xrightarrow{p} \frac{d_c}{\mathrm{vol}(G)}\left(\boldsymbol{P}^r\right)_{c,w}$$

$$\frac{\#(w,c)}{|\mathcal{D}|} \xrightarrow{p} \frac{1}{2T}\sum_{r=1}^{T}\left(\frac{d_w}{\mathrm{vol}(G)}\left(\boldsymbol{P}^r\right)_{w,c} + \frac{d_c}{\mathrm{vol}(G)}\left(\boldsymbol{P}^r\right)_{c,w}\right)$$

$$\boldsymbol{P} = \boldsymbol{D}^{-1}\boldsymbol{A}$$

$$\frac{\#(w,c)\,|\mathcal{D}|}{\#(w)\cdot\#(c)} \xrightarrow{p} \frac{\mathrm{vol}(G)}{2T}\left(\frac{1}{d_c}\sum_{r=1}^{T}\left(\boldsymbol{P}^r\right)_{w,c} + \frac{1}{d_w}\sum_{r=1}^{T}\left(\boldsymbol{P}^r\right)_{c,w}\right)$$

Understanding random walk + skip gram



DeepWalk is asymptotically and implicitly factorizing

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

$\boldsymbol{A}$ Adjacency matrix

$\boldsymbol{D}$ Degree matrix

$$vol(G) = \sum_i \sum_j A_{ij}$$

b: #negative samples

T: context window size

Skip-gram based graph embedding

- DeepWalk $$\log\left(\frac{\mathrm{vol}(G)}{b}\left(\frac{1}{T}\sum_{r=1}^{T}\left(\boldsymbol{D}^{-1}\boldsymbol{A}\right)^{r}\right)\boldsymbol{D}^{-1}\right)$$
- LINE $$\log\left(\frac{\mathrm{vol}(G)}{b}\boldsymbol{D}^{-1}\boldsymbol{A}\boldsymbol{D}^{-1}\right)$$
- PTE $$\log\left(\begin{bmatrix}\alpha\,\mathrm{vol}(G_{\mathsf{ww}})(\boldsymbol{D}_{\mathsf{row}}^{\mathsf{ww}})^{-1}\boldsymbol{A}_{\mathsf{ww}}(\boldsymbol{D}_{\mathsf{col}}^{\mathsf{ww}})^{-1}\\ \beta\,\mathrm{vol}(G_{\mathsf{dw}})(\boldsymbol{D}_{\mathsf{row}}^{\mathsf{dw}})^{-1}\boldsymbol{A}_{\mathsf{dw}}(\boldsymbol{D}_{\mathsf{col}}^{\mathsf{dw}})^{-1}\\ \gamma\,\mathrm{vol}(G_{\mathsf{lw}})(\boldsymbol{D}_{\mathsf{row}}^{\mathsf{lw}})^{-1}\boldsymbol{A}_{\mathsf{lw}}(\boldsymbol{D}_{\mathsf{col}}^{\mathsf{lw}})^{-1}\end{bmatrix}\right)-\log b$$
- node2vec $$\log\left(\frac{\frac{1}{2T}\sum_{r=1}^{T}\left(\sum_{u}\boldsymbol{X}_{w,u}\underline{\boldsymbol{P}}_{c,w,u}^{r}+\sum_{u}\boldsymbol{X}_{c,u}\underline{\boldsymbol{P}}_{w,c,u}^{r}\right)}{b\left(\sum_{u}\boldsymbol{X}_{w,u}\right)\left(\sum_{u}\boldsymbol{X}_{c,u}\right)}\right)$$

- Qiu et al. Network embedding as matrix factorization: unifying deepwalk, line, pte, and node2vec. In *WSDM'18.*
- Perozzi et al. DeepWalk: Online learning of social representations. In *KDD'14.*
- Tang et al. LINE: Large-scale information network embedding. In *WWW'15.*
- Tang et al. PTE: Predictive text embedding through large-scale heterogeneous text networks. In *KDD'15.*
- Grover and Leskovec. Node2vec: scalable feature learning for networks. In *KDD'16.*

Graph embedding: NetMF

Factorize the DeepWalk matrix explicitly, e.g.,
using singular-value decomposition (SVD)

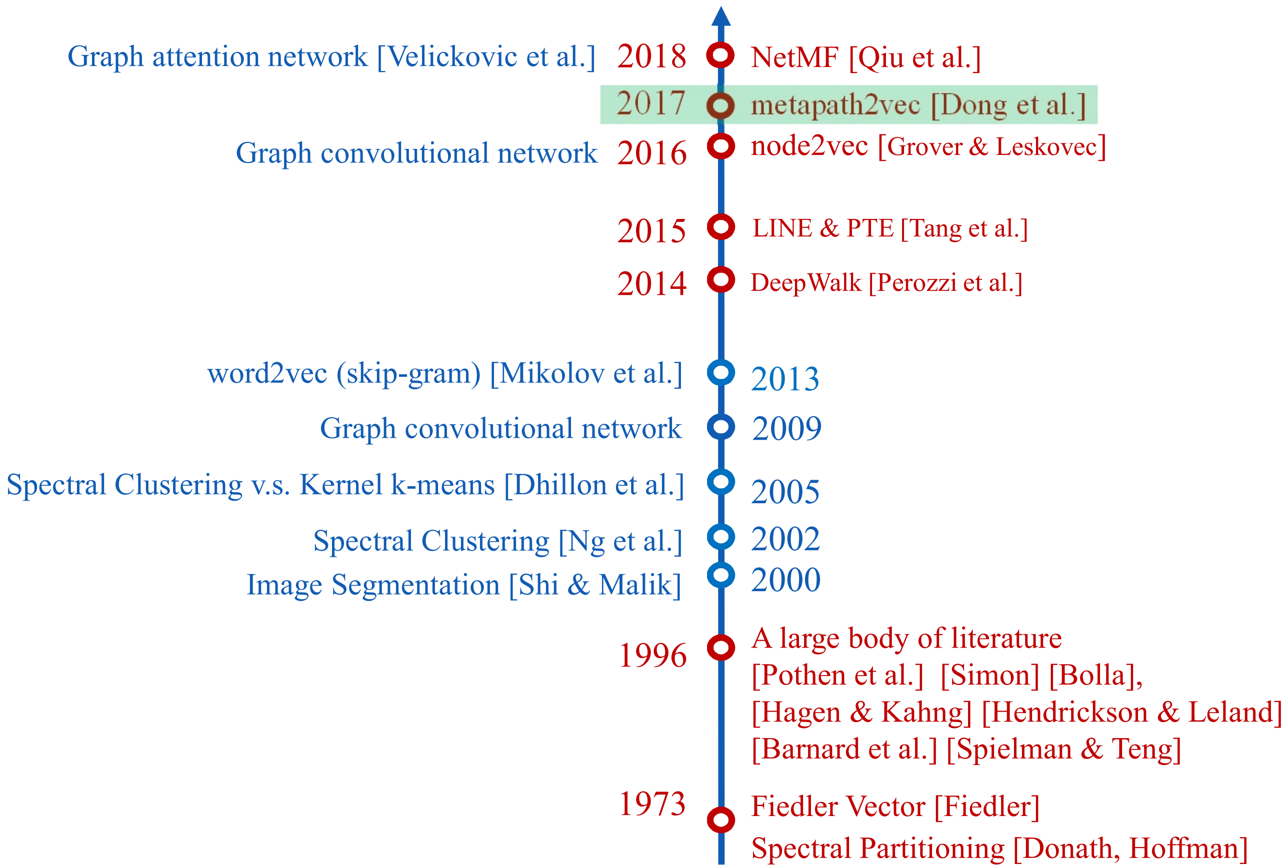
$$\log\left(\frac{\operatorname{vol}(G)}{b}\left(\frac{1}{T}\sum_{r=1}^{T}\left(\boldsymbol{D}^{-1}\boldsymbol{A}\right)^{r}\right)\boldsymbol{D}^{-1}\right)$$

This may be computationally challenging with a large T

$$\begin{aligned}\left(\frac{1}{T}\sum_{r=1}^{T}\left(\boldsymbol{D}^{-1}\boldsymbol{A}\right)^{r}\right)\boldsymbol{D}^{-1} &= \left(\boldsymbol{D}^{-1/2}\right)\left(\frac{1}{T}\sum_{r=1}^{T}\left(\boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}\right)^{r}\right)\left(\boldsymbol{D}^{-1/2}\right)\\ &= \left(\boldsymbol{D}^{-1/2}\right)\left(\boldsymbol{U}\underbrace{\left(\frac{1}{T}\sum_{r=1}^{T}\boldsymbol{\Lambda}^{r}\right)}_{\text{A polynomial}}\boldsymbol{U}^{\top}\right)\left(\boldsymbol{D}^{-1/2}\right)\end{aligned}$$

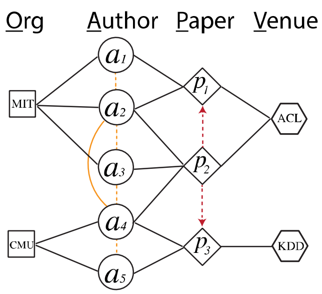
- Qiu et al. Network embedding as matrix factorization: unifying deepwalk, line, pte, and node2vec. In *WSDM'18.*

A brief history of network embedding



Heterogeneous graph embedding

- Input: a heterogeneous graph $G = (V, E)$
- Output: $\boldsymbol{X} \in R^{|V|\times k}, k \ll |V|$, k-dim vector $\boldsymbol{X}_v$ for each node *v.*

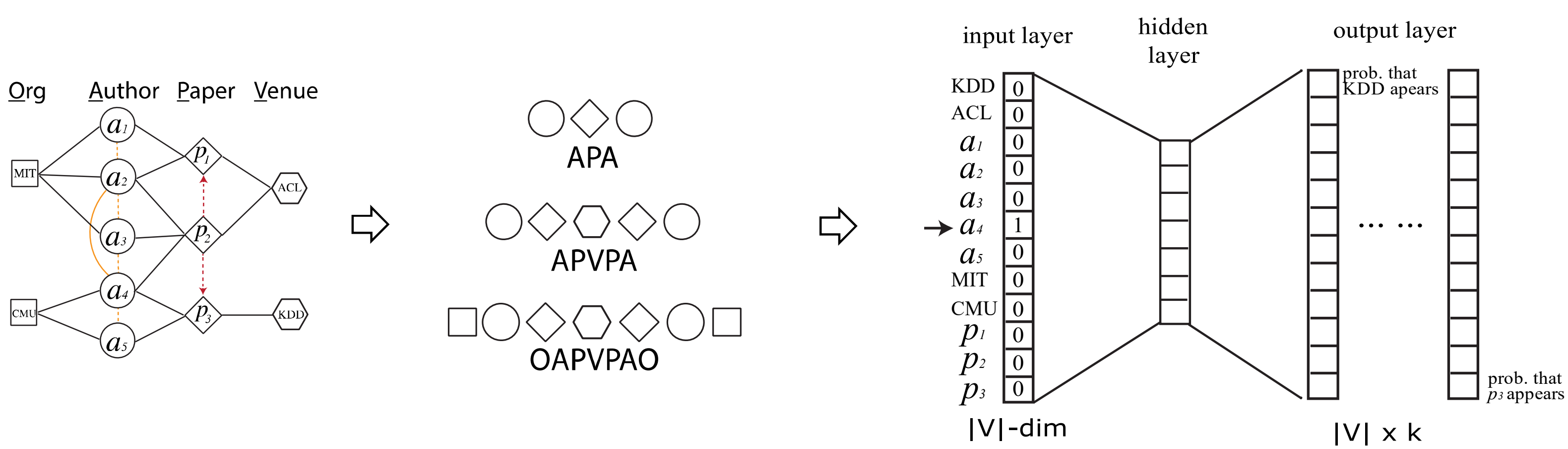


- How do we random walk over heterogeneous networks?
- How do we apply skip-gram over different types of nodes?

~~hand-crafted~~ **latent** feature matrix

Feature **learning**

Heterogeneous graph embedding

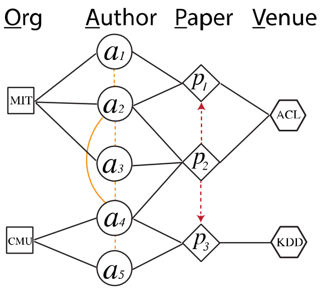


meta-path-based random walks

skip-gram

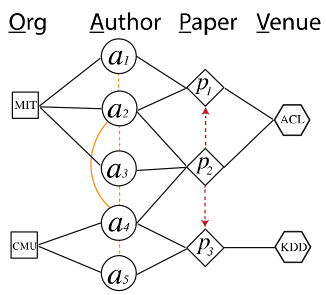
- Sun and Han. Mining heterogeneous information networks: Principles and Methodologies. Morgan & Claypool Publishers, 2012.
- Dong et al. metapath2vec: scalable representation learning for heterogeneous networks. In *ACM KDD* 2017.

Heterogeneous graph embedding



Goal: to generate paths that are able to capture both the semantic and structural correlations between different types of nodes, facilitating the transformation of heterogeneous network structures into skip-gram.

Heterogeneous graph embedding: Meta-Path-Based Random Walks



- Given a meta-path scheme

$$\mathcal{P}: V_1 \xrightarrow{R_1} V_2 \xrightarrow{R_2} \cdots V_t \xrightarrow{R_t} V_{t+1} \cdots \xrightarrow{R_{l-1}} V_l$$

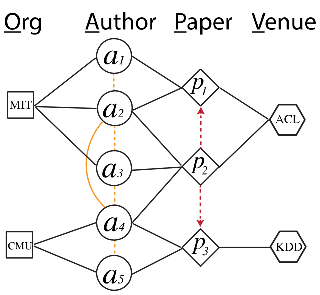
- The transition probability at step *i* is defined as

$$p(v^{i+1}|v_t^i, \mathcal{P}) = \begin{cases} \frac{1}{|N_{t+1}(v_t^i)|} & (v^{i+1}, v_t^i) \in E, \phi(v^{i+1}) = t{+}1 \\ 0 & (v^{i+1}, v_t^i) \in E, \phi(v^{i+1}) \neq t{+}1 \\ 0 & (v^{i+1}, v_t^i) \notin E \end{cases}$$

- Recursive guidance for random walkers, i.e.,

$$p(v^{i+1}|v_t^i) = p(v^{i+1}|v_1^i), \text{ if } t = l$$

Heterogeneous graph embedding: Meta-Path-Based Random Walks



- Given a meta-path scheme (Example)

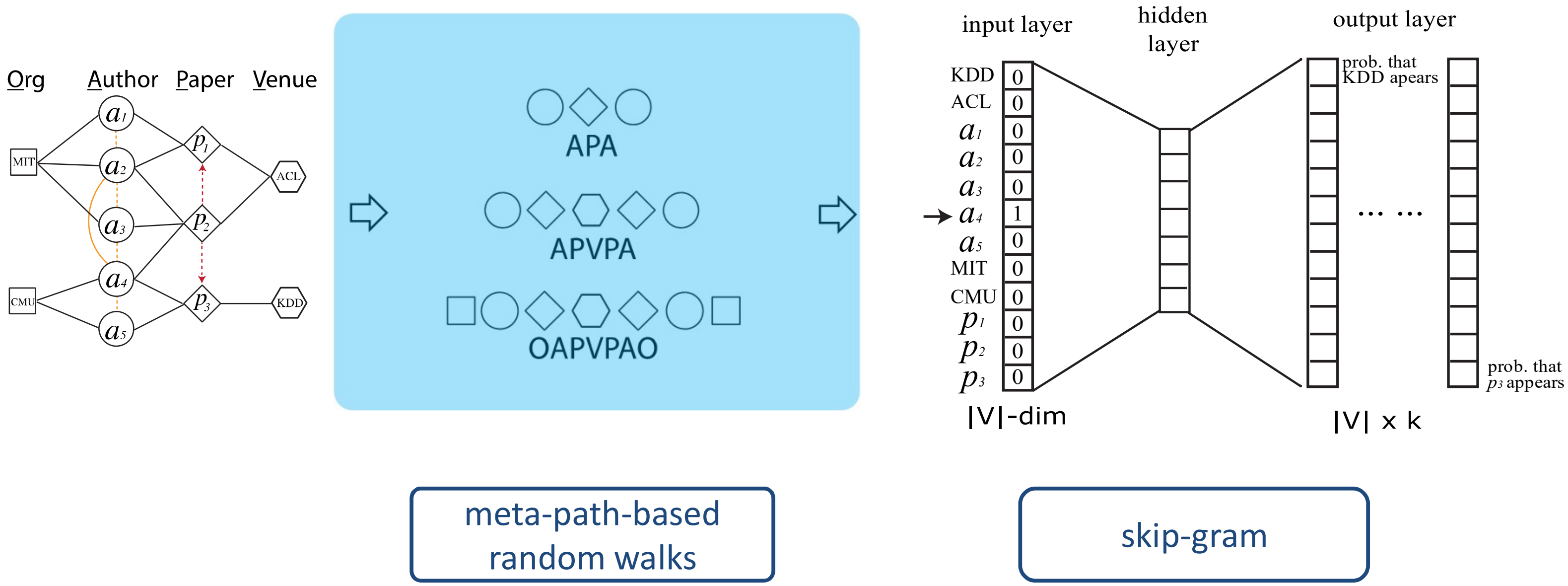
  OAPVPAO

- In a traditional random walk procedure, in the toy example, the next step of a walker on node a_4 transitioned from node O_{CMU} can be all types of nodes surrounding it— a_2, a_3, a_5, p_2, p_3 and O_{CMU}.

- Under the meta-path scheme 'OAPVPAO', for example, the walker is biased towards paper nodes (P) given its previous step on an organization node O_{CMU}(O), following the semantics of this meta-path.

Heterogeneous graph embedding

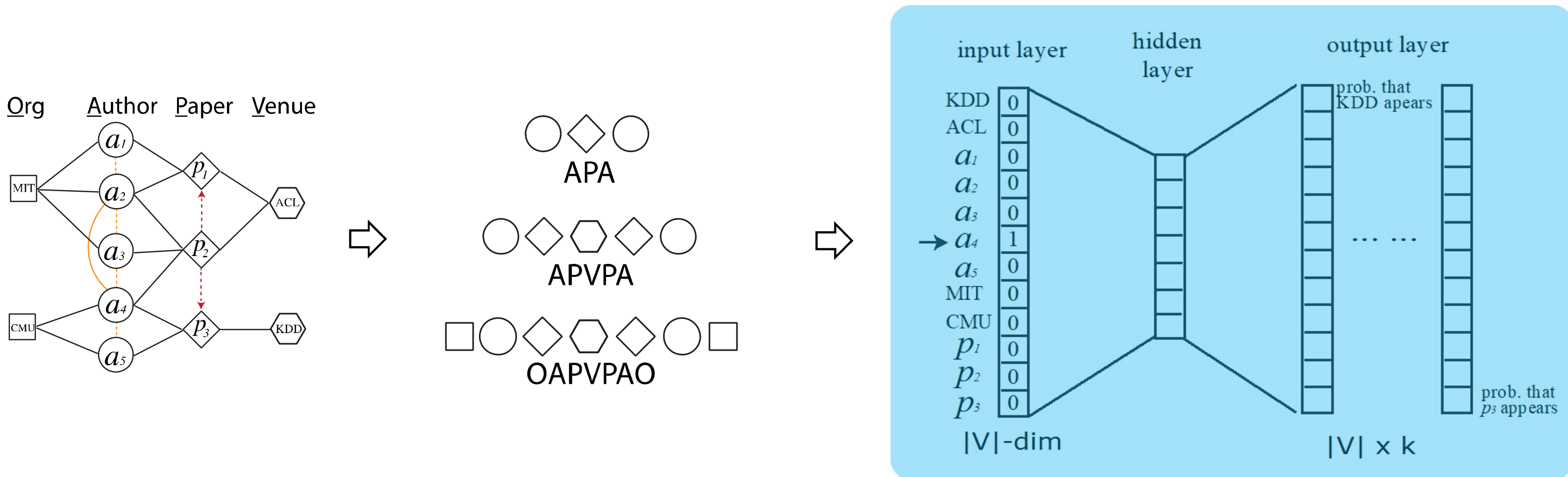
- Input: a heterogeneous graph $G = (V, E)$
- Output: $\boldsymbol{X} \in R^{|V| \times k}, k \ll |V|$, k-dim vector $\boldsymbol{X}_v$ for each node v.



- Sun and Han. Mining heterogeneous information networks: Principles and Methodologies. Morgan & Claypool Publishers, 2012.
- Dong et al. metapath2vec: scalable representation learning for heterogeneous networks. In *ACM KDD* 2017.

Heterogeneous graph embedding

- Input: a heterogeneous graph $G = (V, E)$
- Output: $\boldsymbol{X} \in R^{|V| \times k}, k \ll |V|$, k-dim vector $\boldsymbol{X}_v$ for each node v.



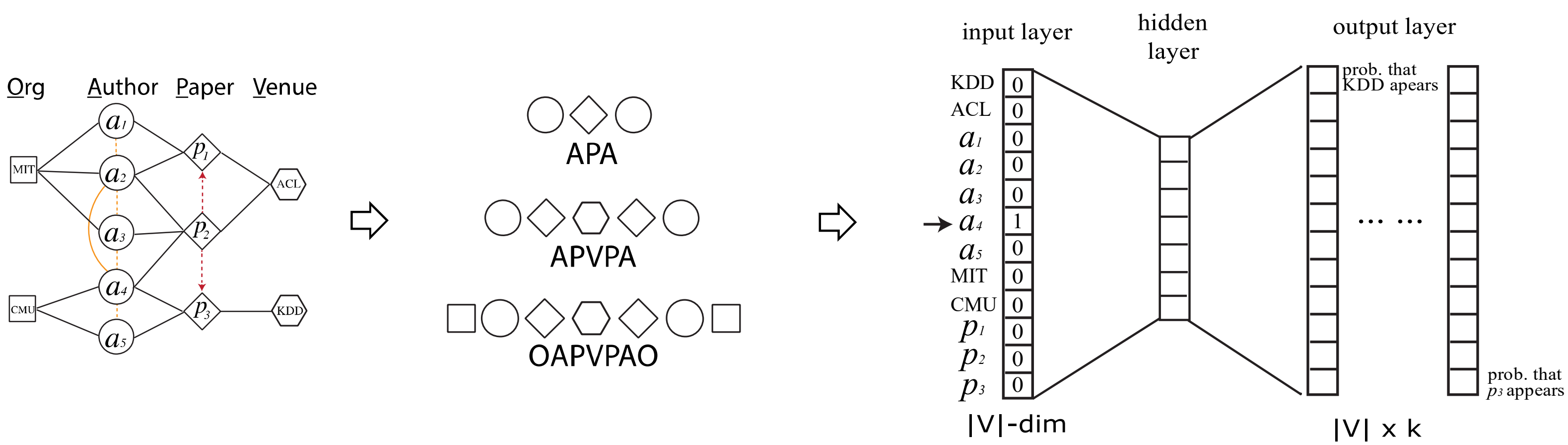
meta-path-based random walks

skip-gram

- Sun and Han. Mining heterogeneous information networks: Principles and Methodologies. Morgan & Claypool Publishers, 2012.
- Dong et al. metapath2vec: scalable representation learning for heterogeneous networks. In *ACM KDD* 2017.

Heterogeneous graph embedding

- Input: a heterogeneous graph $G = (V, E)$
- Output: $\boldsymbol{X} \in R^{|V| \times k}, k \ll |V|$, k-dim vector $\boldsymbol{X}_v$ for each node v.



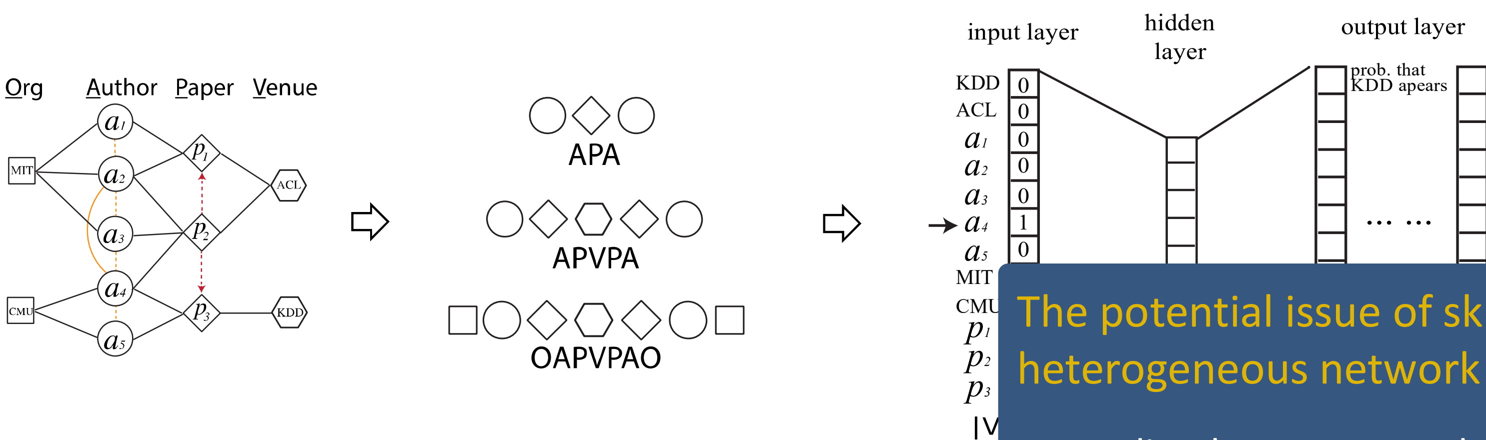
meta-path-based random walks

skip-gram

- Sun and Han. Mining heterogeneous information networks: Principles and Methodologies. Morgan & Claypool Publishers, 2012.
- Dong et al. metapath2vec: scalable representation learning for heterogeneous networks. In *ACM KDD* 2017.

Heterogeneous graph embedding

- Input: a heterogeneous graph $G = (V, E)$
- Output: $\boldsymbol{X} \in R^{|V| \times k}, k \ll |V|$, k-dim vector $\boldsymbol{X}_v$ for each node v.



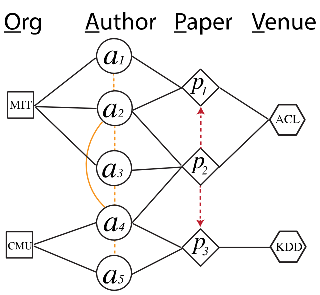
meta-path-based random walks

The potential issue of skip-gram for heterogeneous network embedding:

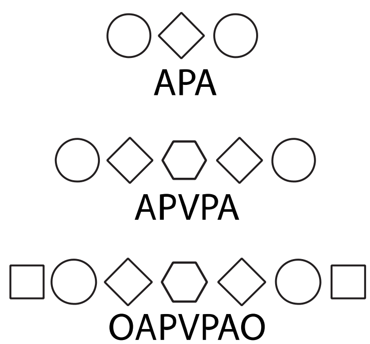
To predict the context node c_t (type t) given a node v, *metapath2vec* encourages all types of nodes to appear in this context position

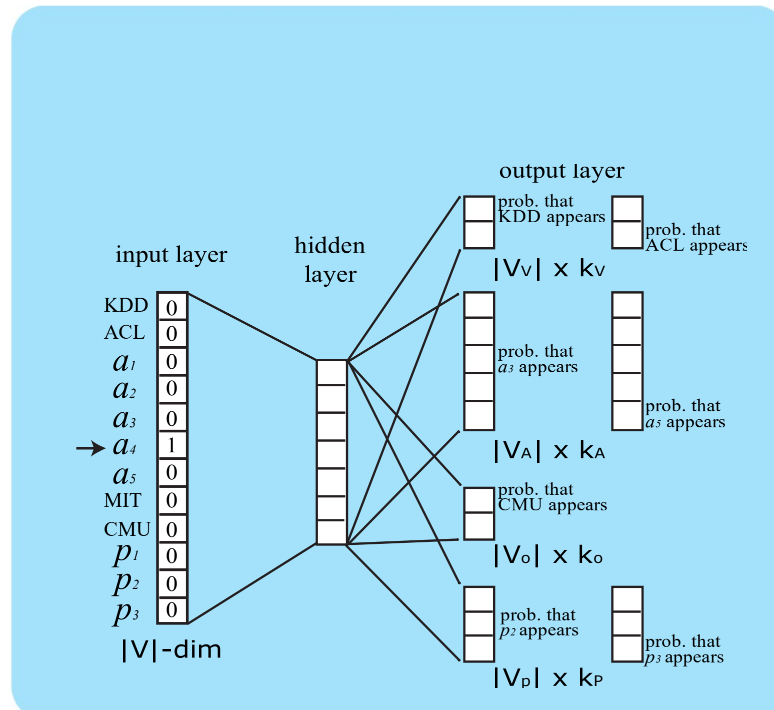
- Sun and Han. Mining heterogeneous information networks: Principles and Methodologies. Morgan & Claypool Publishers, 2012.
- Dong et al. metapath2vec: scalable representation learning for heterogeneous networks. In *ACM KDD* 2017.

Heterogeneous graph embedding



meta paths

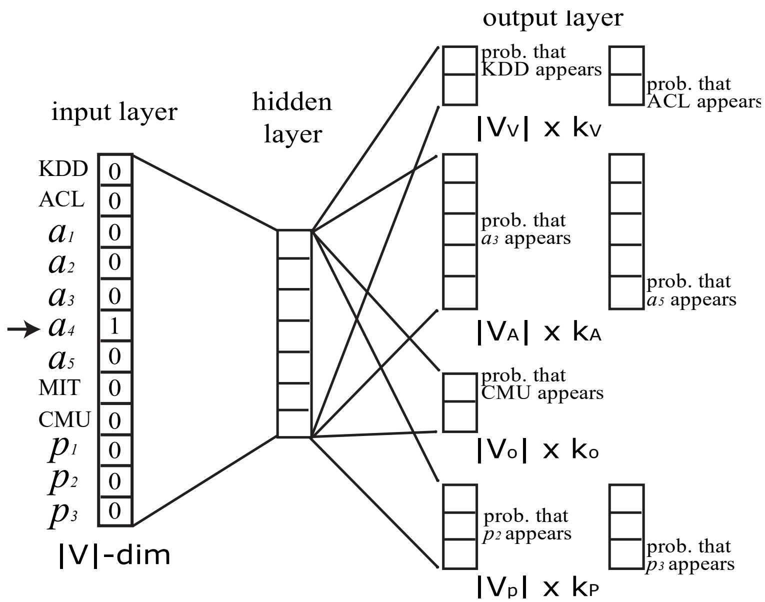




meta-path-based random walks

heterogeneous skip-gram

Heterogeneous Skip-Gram



- softmax in *metapath2vec*

$$p(c_t|v;\theta) = \frac{e^{X_{c_t}} \cdot e^{X_v}}{\sum_{u \in V} e^{X_u} \cdot e^{X_v}}$$

- softmax in *metapath2vec++*

$$p(c_t|v;\theta) = \frac{e^{X_{c_t}} \cdot e^{X_v}}{\sum_{u_t \in V_t} e^{X_{u_t}} \cdot e^{X_v}}$$

- objective function (heterogeneous negative sampling)

$$\mathcal{O}(\mathbf{X}) = \log \sigma(X_{c_t} \cdot X_v) + \sum_{k=1}^{K} \mathbb{E}_{u_t^k \sim P_t(u_t)} [\log \sigma(-X_{u_t^k} \cdot X_v)]$$

- stochastic gradient descent

$$\frac{\partial \mathcal{O}(\mathbf{X})}{\partial X_{u_t^k}} = (\sigma(X_{u_t^k} \cdot X_v - \mathbb{I}_{c_t}[u_t^k]))X_v$$

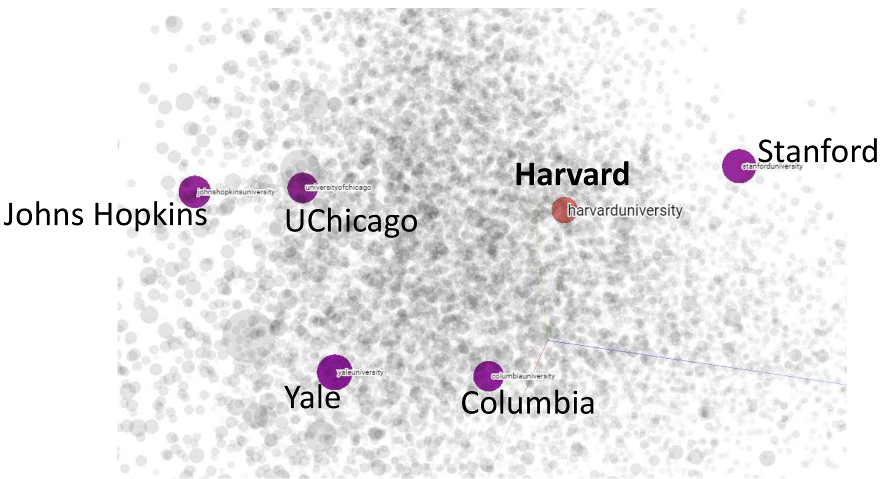
$$\frac{\partial \mathcal{O}(\mathbf{X})}{\partial X_v} = \sum_{k=0}^{K} (\sigma(X_{u_t^k} \cdot X_v - \mathbb{I}_{c_t}[u_t^k]))X_{u_t^k}$$

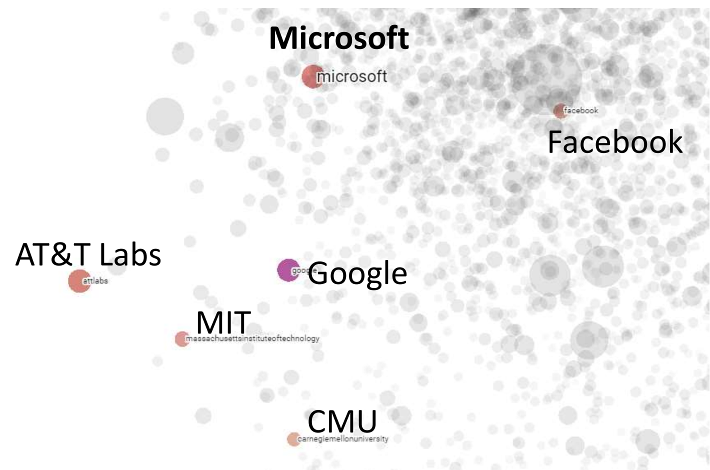
Heterogeneous graph embedding

Application: Embedding Heterogeneous Academic Graph

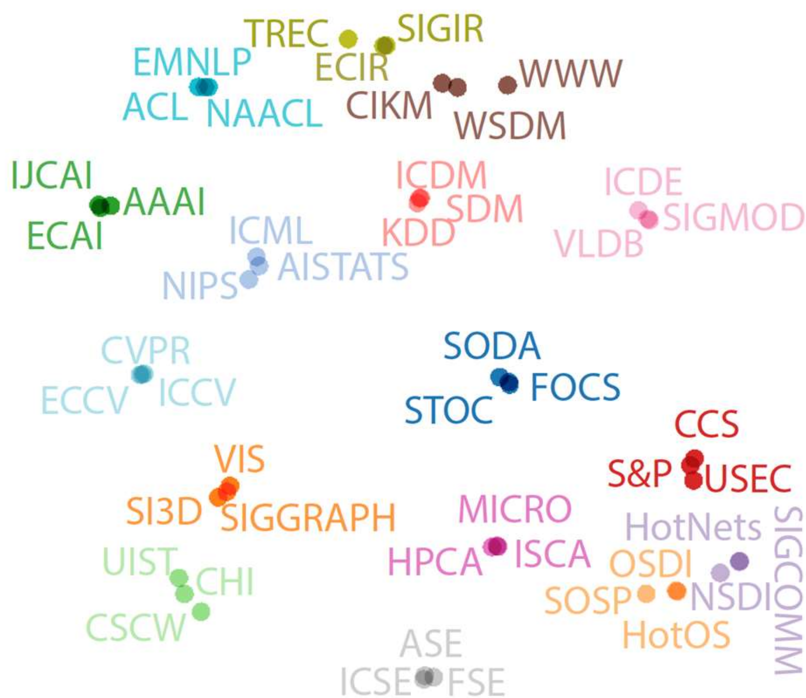
Microsoft Academic Graph

Application: Embedding Heterogeneous Academic Graph

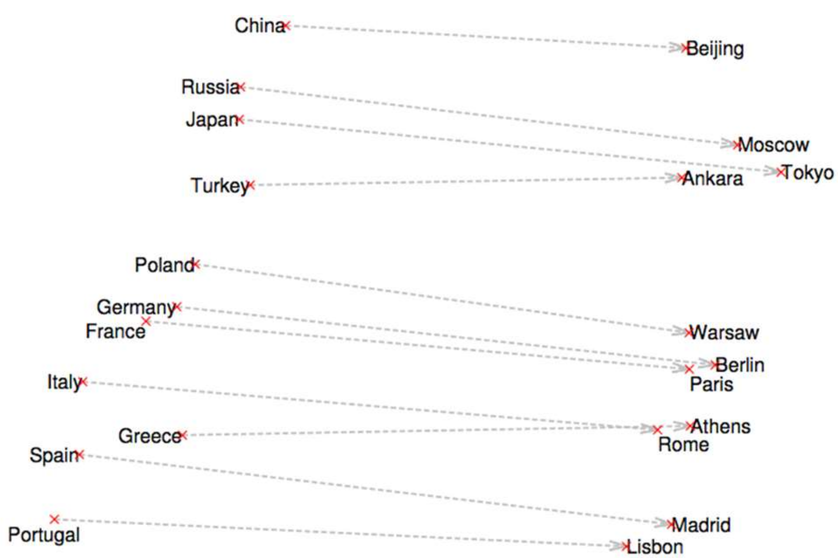




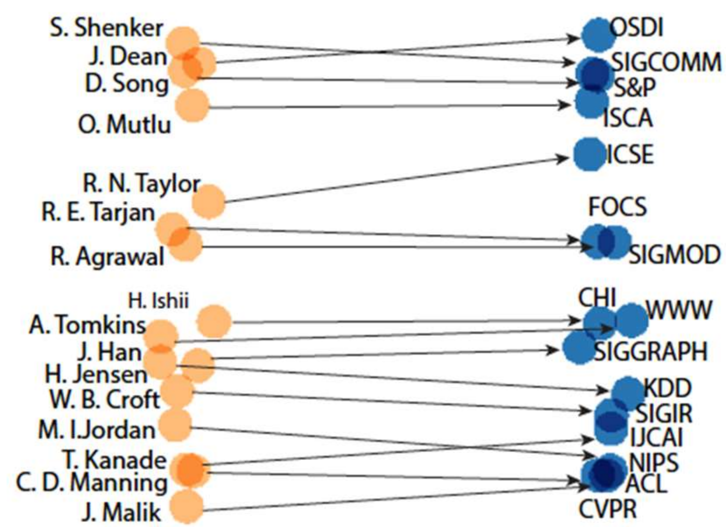
Application: Embedding Heterogeneous Academic Graph



Application: Embedding Heterogeneous Academic Graph

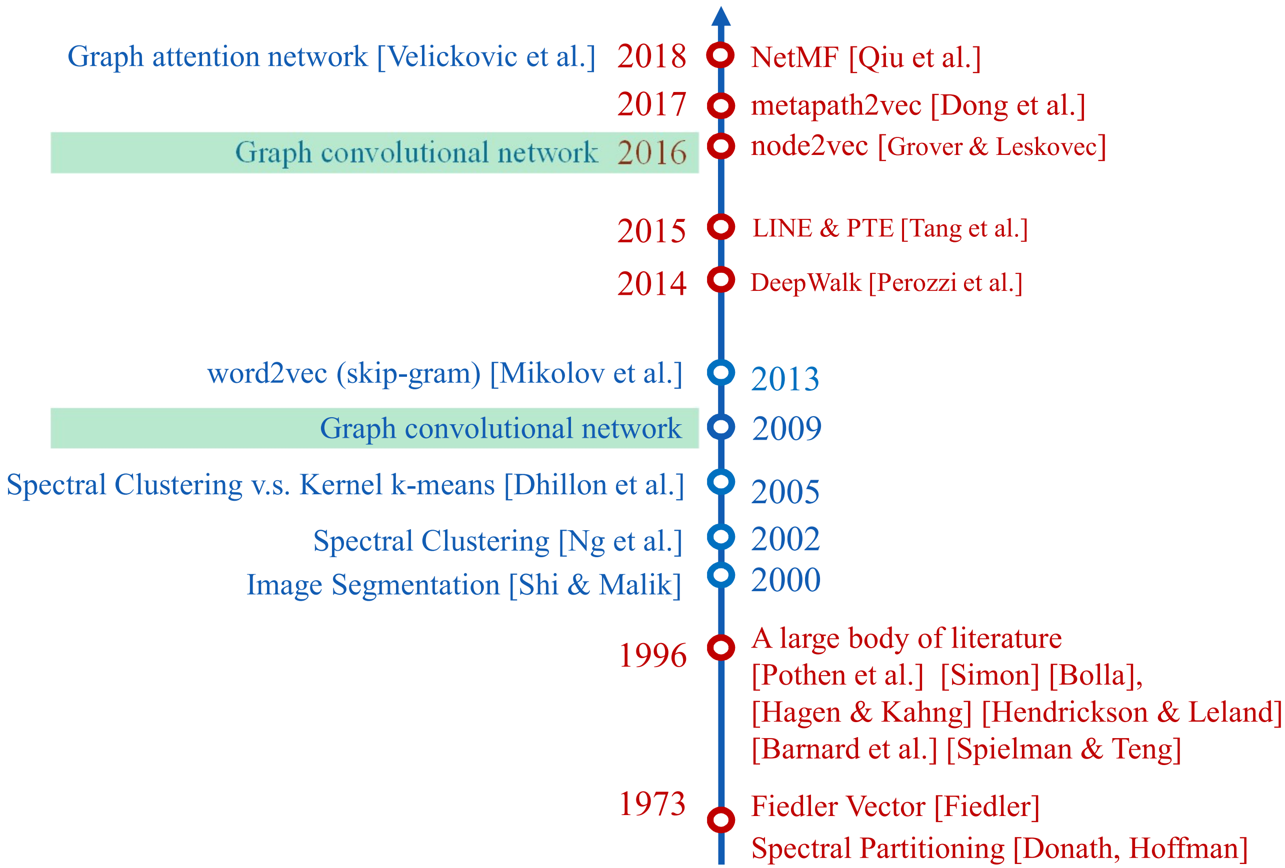


word2vec [Mikolov, 2013]

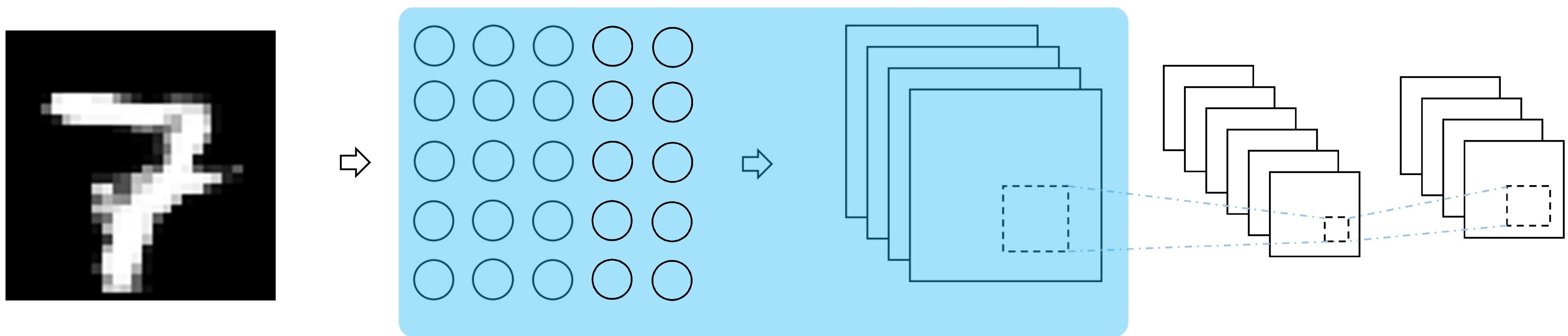


Metapath2vec++ [Dong et al., 2017]

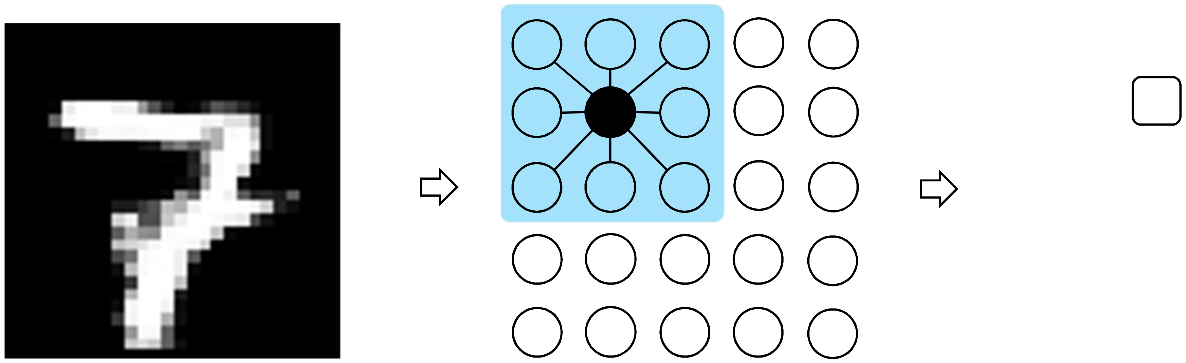
A brief history of graph embedding



Convolutional neural network

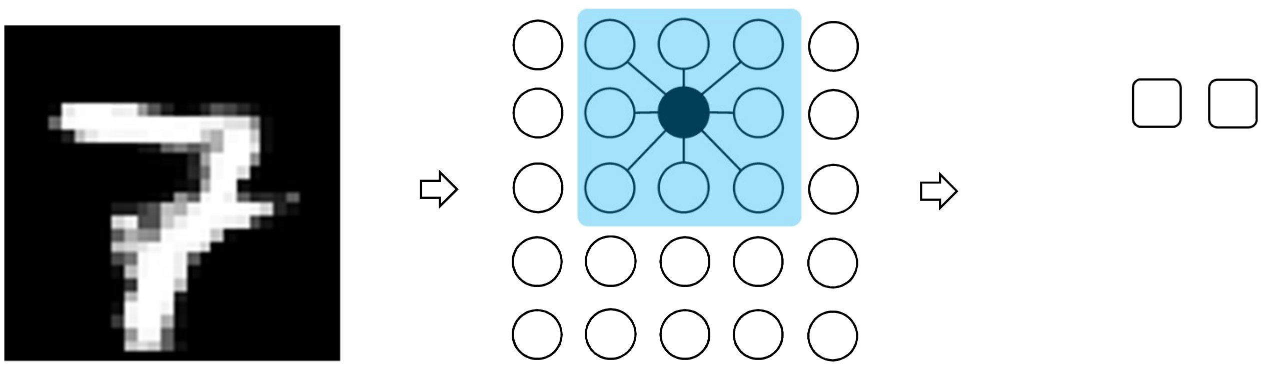


Convolutional neural network



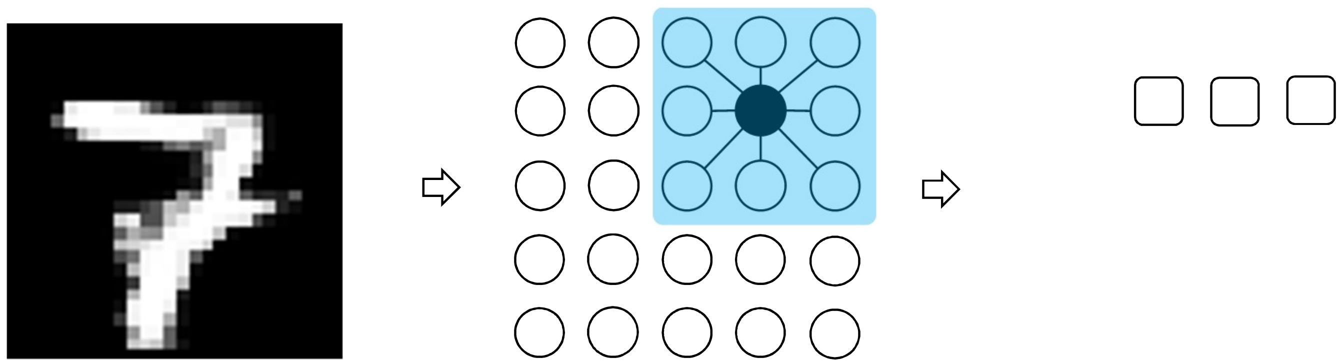
$$\sum WX$$

Convolutional neural network



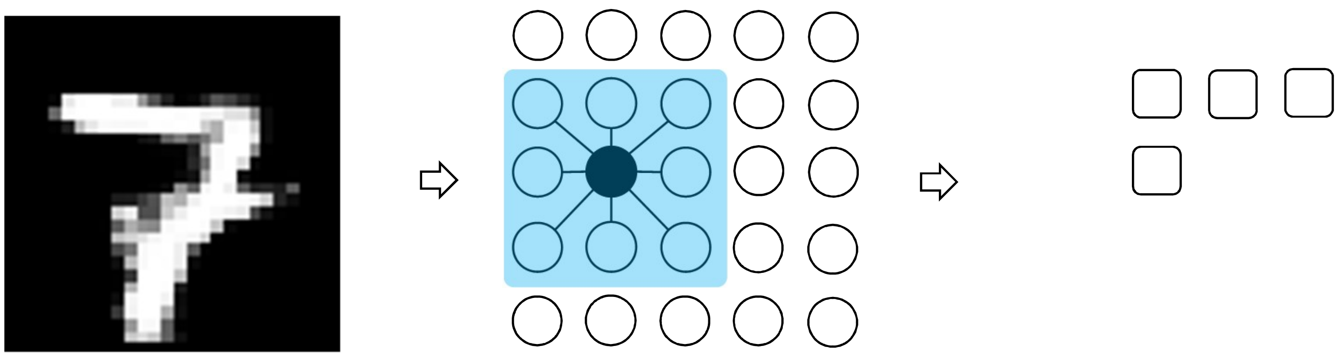
$\sum WX$

Convolutional neural network



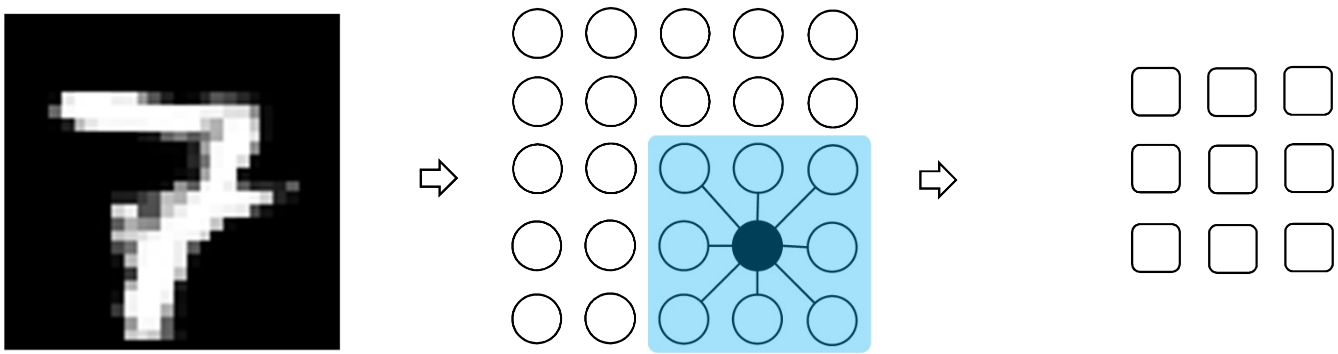
$\sum WX$

Convolutional neural network



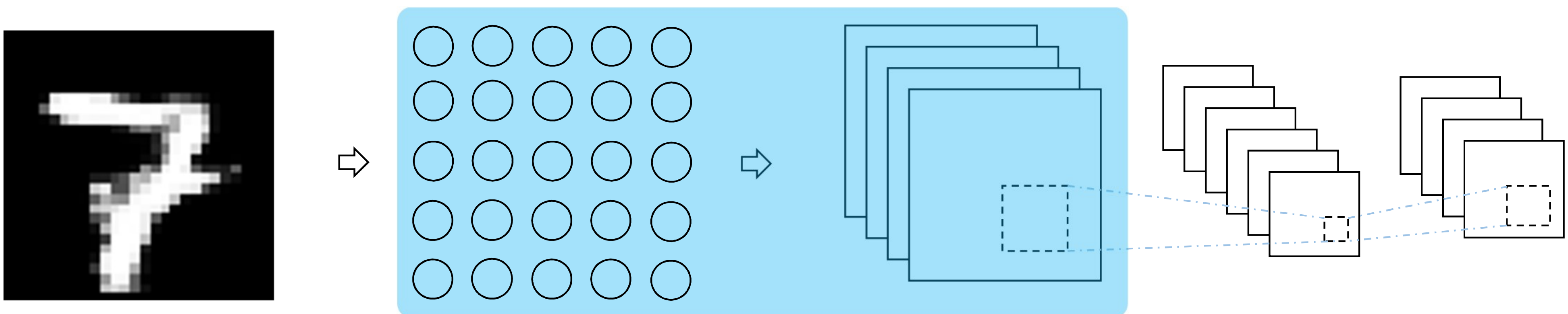
$\sum WX$

Convolutional neural network



$$\sum WX$$

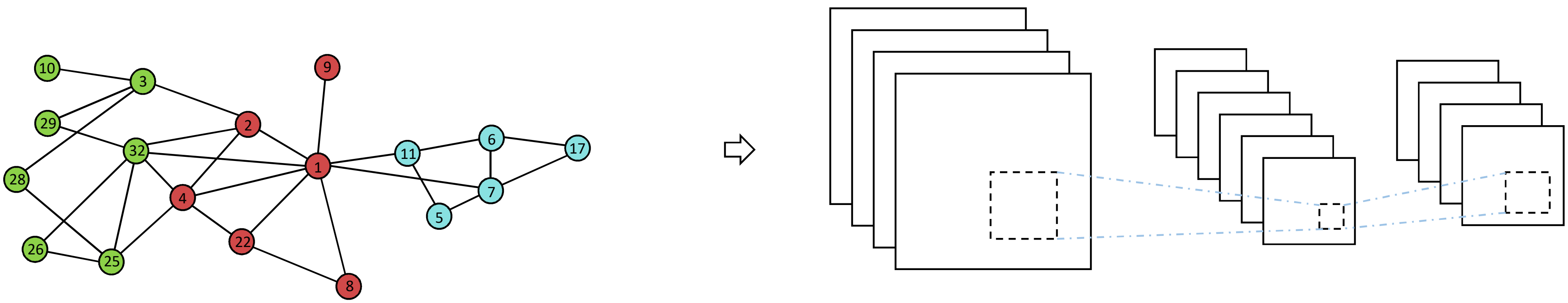
Convolutional neural network



$$\sum WX$$

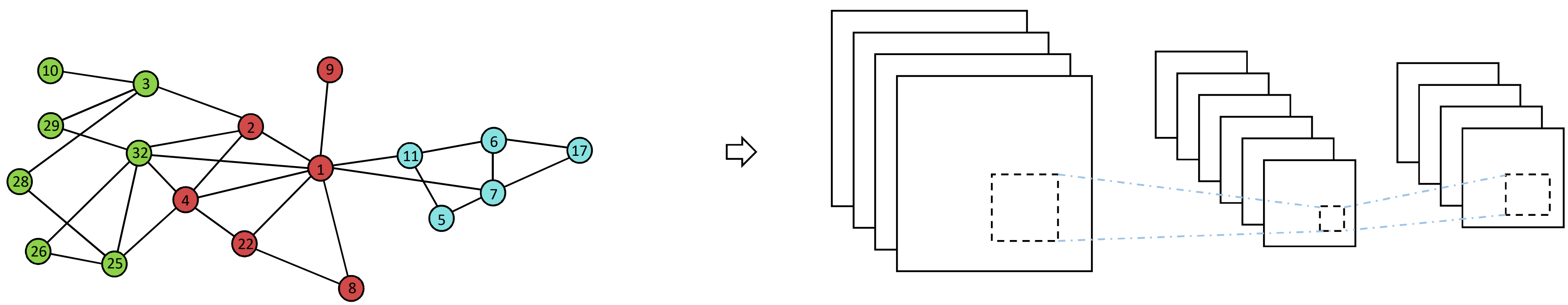
It is straightforward to define convolutions over images with fixed 2D structures

Convolutional neural networks on graphs



How to define convolutions over graphs with arbitrary structures and size?

Convolutional neural networks on graphs



How to define convolutions over graphs with arbitrary structures and size?

Graph convolutional networks

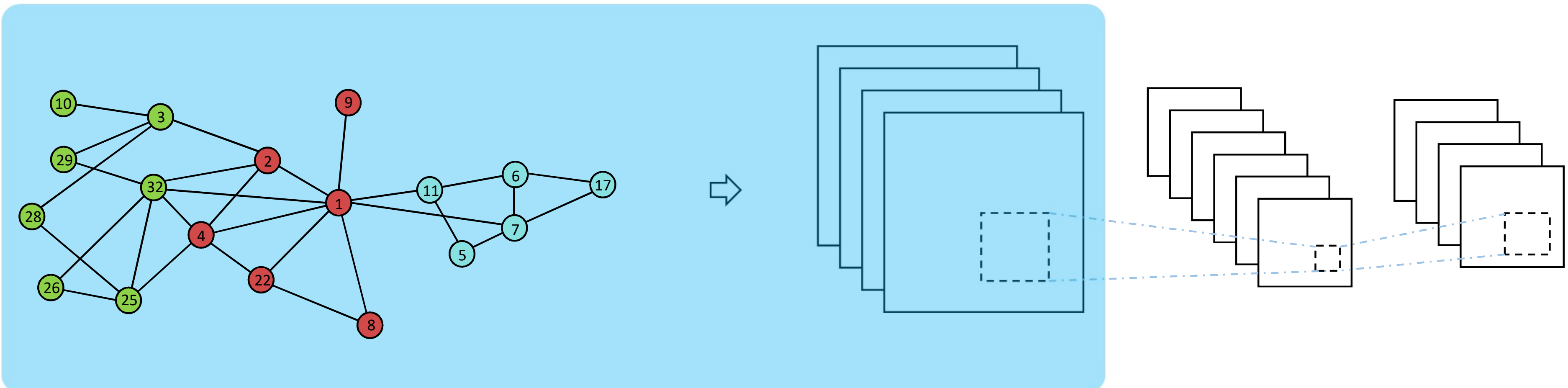
Input: a graph $G = (V, E)$ with $|V| = n$ & $|E| = \mathrm{m}$ and its feature matrix X

- Feature matrix $\boldsymbol{X} \in \mathbb{R}^{n \times \mathrm{d}}$
- Adjacency matrix $\boldsymbol{A} \in \mathbb{R}_{+}^{n \times \mathrm{n}}$
- Degree matrix $\boldsymbol{D} = diag(d_1, d_2, \cdots, d_n)$

Output: for each node, its k-dimension latent feature representation vector $\mathbf{Z}^{n \times k}$

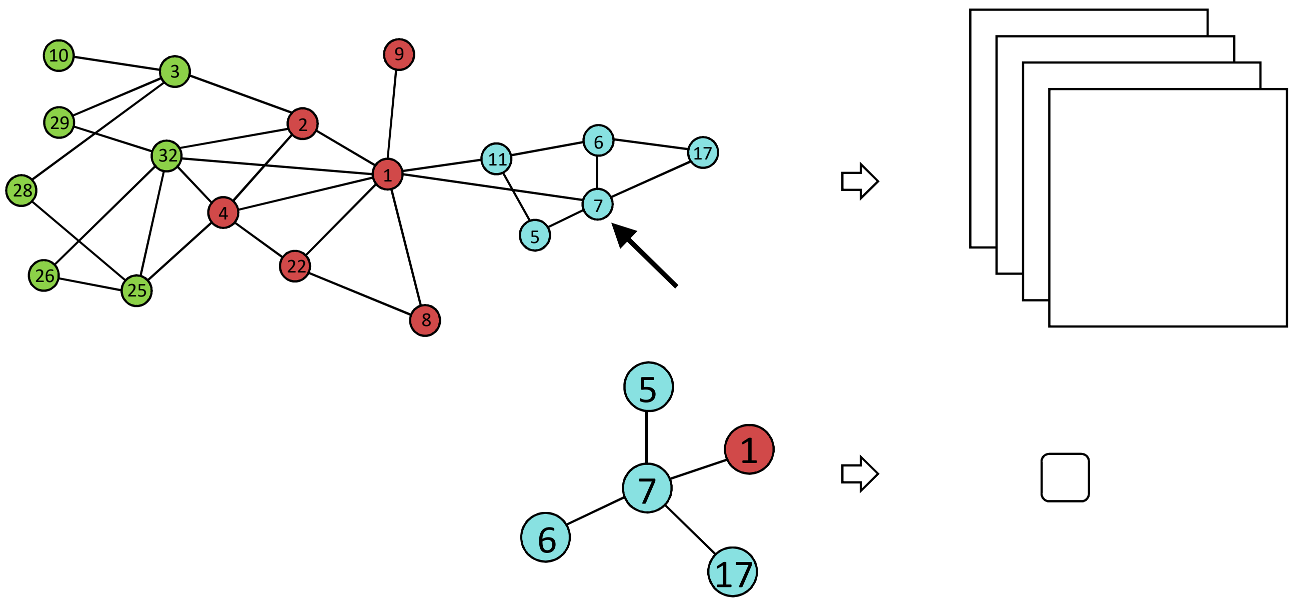
- Latent feature embedding matrix $\mathbf{Z} \in \mathbb{R}^{n \times k}$

Graph convolutional networks



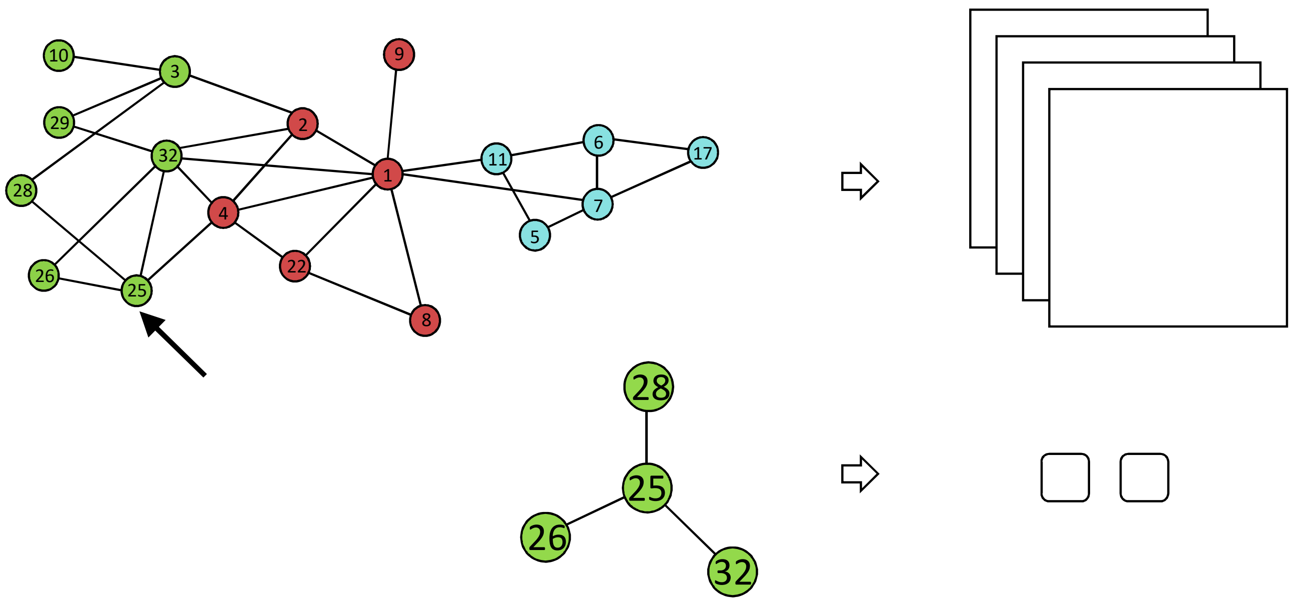
$$Z^{(1)} = f(X, A, W^{(0)})$$

Graph convolutional networks



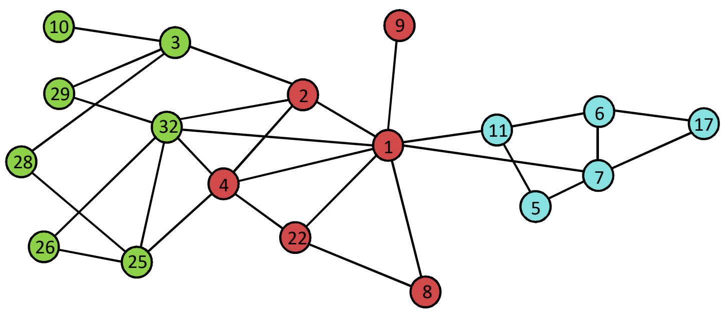
$$\boldsymbol{Z}^{(1)} = f(\boldsymbol{X}, \boldsymbol{A}, \boldsymbol{W}^{(0)})$$

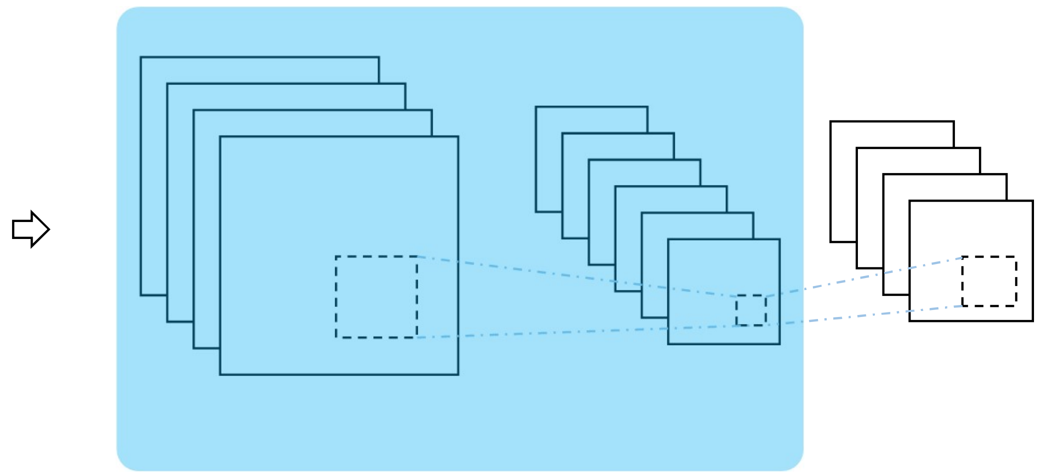
Graph convolutional networks



$$\boldsymbol{Z}^{(1)} = f(\boldsymbol{X}, \boldsymbol{A}, \boldsymbol{W}^{(0)})$$

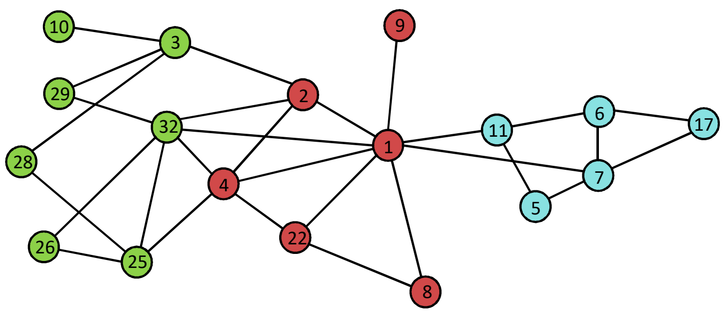
Graph convolutional networks

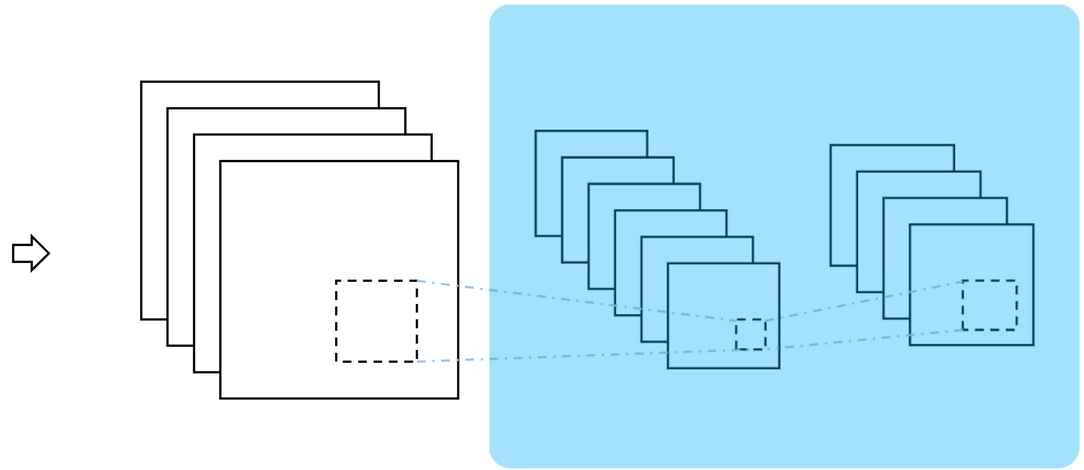




$$Z^{(1)} = f(X, A, W^{(0)}) \qquad Z^{(2)} = f(Z^{(1)}, A, W^{(1)})$$

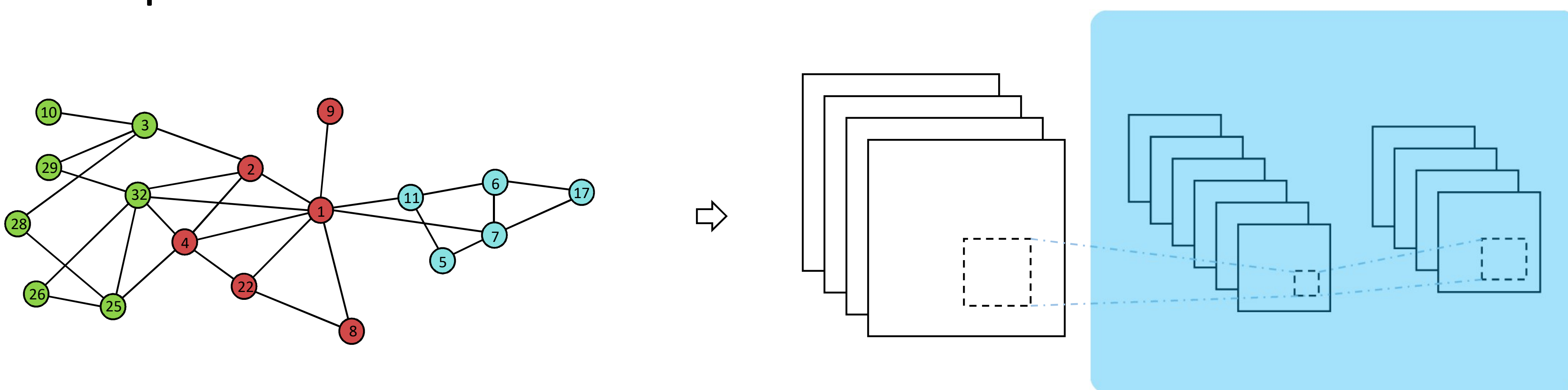
Graph convolutional networks





$$Z^{(1)} = f(X, A, W^{(0)}) \quad Z^{(2)} = f(Z^{(1)}, A, W^{(1)}) \quad \boldsymbol{Z}^{(3)} = f(\boldsymbol{Z}^{(2)}, \boldsymbol{A}, \boldsymbol{W}^{(2)})$$

Graph convolutional networks



$$\underbrace{\boldsymbol{Z}^{(1)} = f(\boldsymbol{X}, \boldsymbol{A}, \boldsymbol{W}^{(0)}) \quad \boldsymbol{Z}^{(2)} = f(\boldsymbol{Z}^{(1)}, \boldsymbol{A}, \boldsymbol{W}^{(1)}) \quad \boldsymbol{Z}^{(\mathbf{3})} = f(\boldsymbol{Z}^{(2)}, \boldsymbol{A}, \boldsymbol{W}^{(2)})}_{\boldsymbol{Z}^{(l+1)} = f(\boldsymbol{Z}^{(l)}, \boldsymbol{A}, \boldsymbol{W}^{(l)}) \text{ with } \boldsymbol{Z}^{(0)} = \boldsymbol{X}}$$

Graph convolutional networks

$$\boldsymbol{Z}^{(l+1)} = f(\boldsymbol{Z}^{(l)}, \boldsymbol{A}, \boldsymbol{W}^{(l)})$$

Could be any non-linear activation function.

$$f(\boldsymbol{Z}^{(l)}, \boldsymbol{A}, \boldsymbol{W}^{(l)}) = \mathit{ReLU}(\boldsymbol{A}\boldsymbol{Z}^{(l)}\boldsymbol{W}^{(l)})$$

$\boldsymbol{W}^{(l)}$: parameters over layer l

Kipf & Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR 2017*.

Graph convolutional networks

$$\boldsymbol{Z}^{(l+1)} = f(\boldsymbol{Z}^{(l)}, \boldsymbol{A}, \boldsymbol{W}^{(l)})$$

Could be any non-linear activation function.

$$f(\boldsymbol{Z}^{(l)}, \boldsymbol{A}, \boldsymbol{W}^{(l)}) = ReLU(\boldsymbol{A}\boldsymbol{Z}^{(l)}\boldsymbol{W}^{(l)})$$

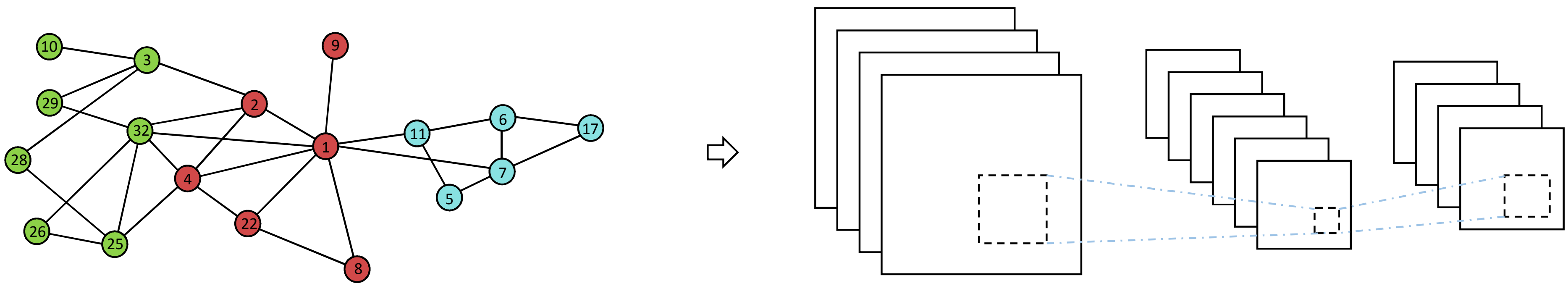
$$\boldsymbol{D}^{-1}\boldsymbol{A}$$

$$\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}$$

$$\boldsymbol{D}^{-\frac{1}{2}}(\boldsymbol{A}+\boldsymbol{I})\boldsymbol{D}^{-\frac{1}{2}}$$

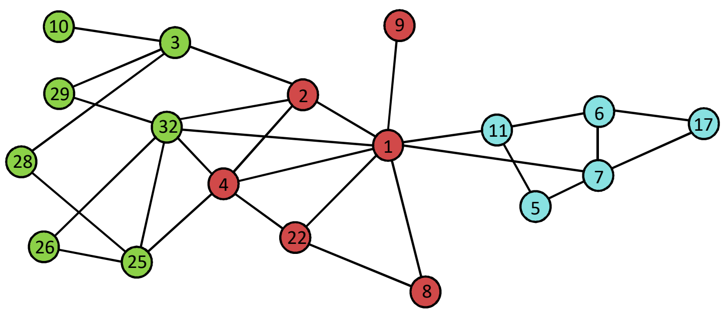
Kipf & Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR 2017*.

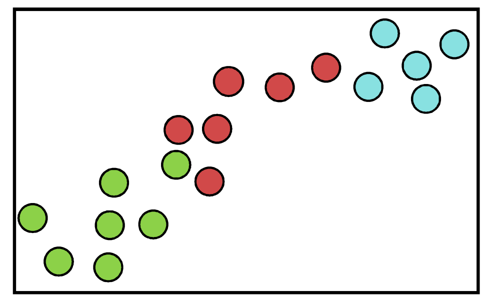
Graph convolutional networks



$$\boldsymbol{Z}^{(l+1)} = f(\boldsymbol{Z}^{(l)}, \boldsymbol{A}, \boldsymbol{W}^{(l)}) = ReLU(\boldsymbol{D}^{-\frac{1}{2}}(\boldsymbol{A} + \boldsymbol{I})\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{Z}^{(l)}\boldsymbol{W}^{(l)})$$

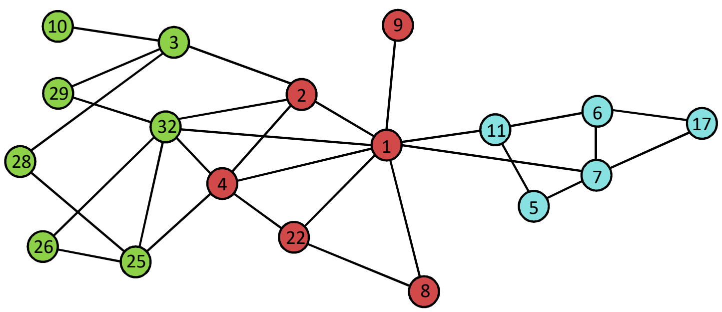
Graph convolutional networks

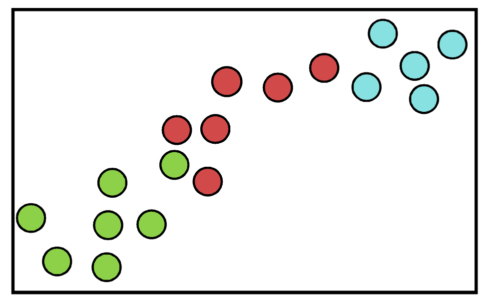




$$\boldsymbol{Z}^{(l+1)} = f(\boldsymbol{Z}^{(l)}, \boldsymbol{A}, \boldsymbol{W}^{(l)}) = ReLU(\boldsymbol{D}^{-\frac{1}{2}}(\boldsymbol{A} + \boldsymbol{I})\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{Z}^{(l)}\boldsymbol{W}^{(l)})$$

Graph convolutional networks





Graph & network applications

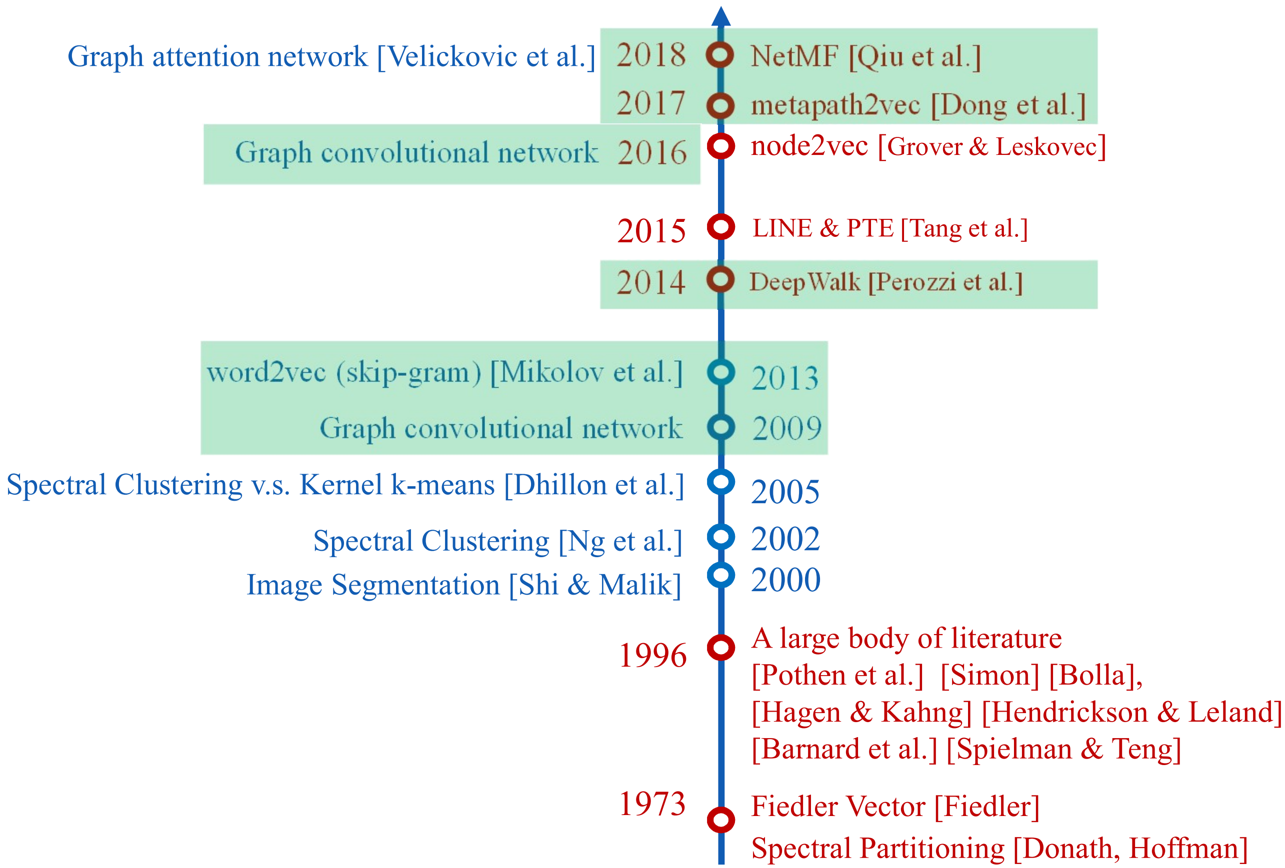
- Node label inference;
- Node clustering;
- Link prediction;
- … …

$$\boldsymbol{Z}^{(l+1)} = f(\boldsymbol{Z}^{(l)}, \boldsymbol{A}, \boldsymbol{W}^{(l)}) = ReLU(\boldsymbol{D}^{-\frac{1}{2}}(\boldsymbol{A}+\boldsymbol{I})\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{Z}^{(l)}\boldsymbol{W}^{(l)})$$

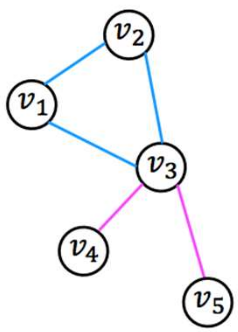
Recent advances in deep learning on graphs

- GraphSAGE:
  - Hamilton, Ying, Leskovec. Inductive representation learning on large graphs. NIPS 2017.
- Graph Attention:
  - Velickovic, et al. Graph Attention Networks. ICLR 2018.

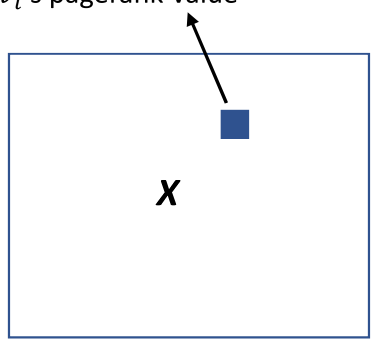
A brief history of network embedding



The conventional graph mining paradigm



x_{ij}: node v_i's j^{th} feature, e.g., v_i's pagerank value



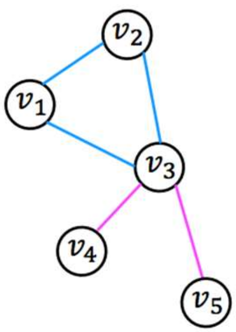
hand-crafted feature matrix

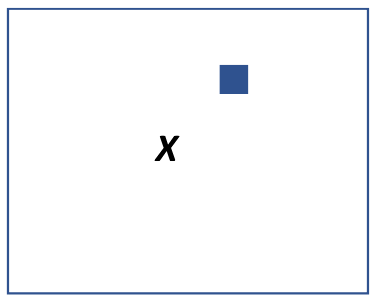
Graph & network applications
- Node label inference;
- Node clustering;
- Link prediction;
- … …

feature **engineering**

machine learning models
- **Classification algorithms,** such as logistic regression, SVM, and random forest;
- **Regression algorithms,** such as linear regression;
- **Clustering algorithms**, such as k-means.

Representation learning for graph mining





~~hand-crafted~~ **latent** feature matrix

Graph & network applications
- Node label inference;
- Node clustering;
- Link prediction;
- … …

Feature ~~engineering~~ **learning**

machine learning models
- **Classification algorithms,** such as logistic regression, SVM, and random forest;
- **Regression algorithms,** such as linear regression;
- **Clustering algorithms**, such as k-means.

- Bengio, Courville, Vincent. Representation learning: A review and new perspectives. *IEEE TPAMI* 2013.
- LeCun, Bengio, Hinton. Deep learning. *Nature,* 521(7553):436–444, 2015.

Module 3: Graph Representation Learning

- Representation learning
- Skip-gram based graph representation learning
  - Homogeneous network embedding
  - Understanding network embedding
  - Heterogeneous network embedding
- Deep learning for graph representation learning
  - Graph convolutional networks