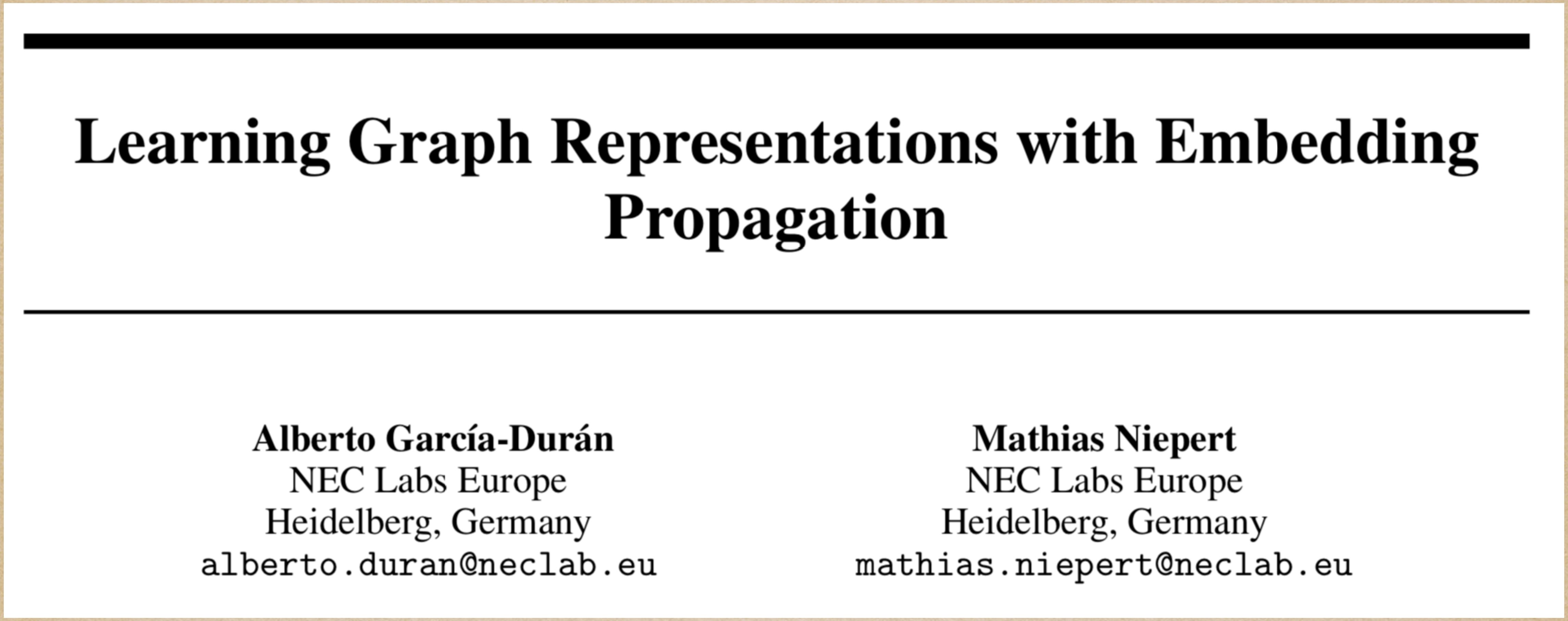
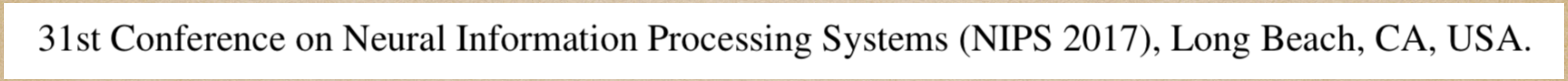


Learning Graph Representations with Embedding Propagation

Alberto García-Durán
NEC Labs Europe
Heidelberg, Germany
alberto.duran@neclab.eu

Mathias Niepert
NEC Labs Europe
Heidelberg, Germany
mathias.niepert@neclab.eu



31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

An Overview of Embedding Propagation (EP)

- **Motivation:** Graph-structured data occurs in numerous application domains such as social networks, bioinformatics and relational knowledge bases. Nodes may have an arbitrary number of label types, and these label types may be heterogeneous (node entities, text, images...).
- **Contributions:** We propose **embedding propagation** (EP), an unsupervised learning framework that supports arbitrary label types.
- an unsupervised learning framework
- EP learns vector representations (embeddings) of graphs by passing messages between neighboring nodes.

Definition

$G = (V, E)$

$E \subseteq \{(v, w) \mid v, w \in V\}$

k label classes:

$\mathbf{L} = \{L_1, ..., L_k\}$

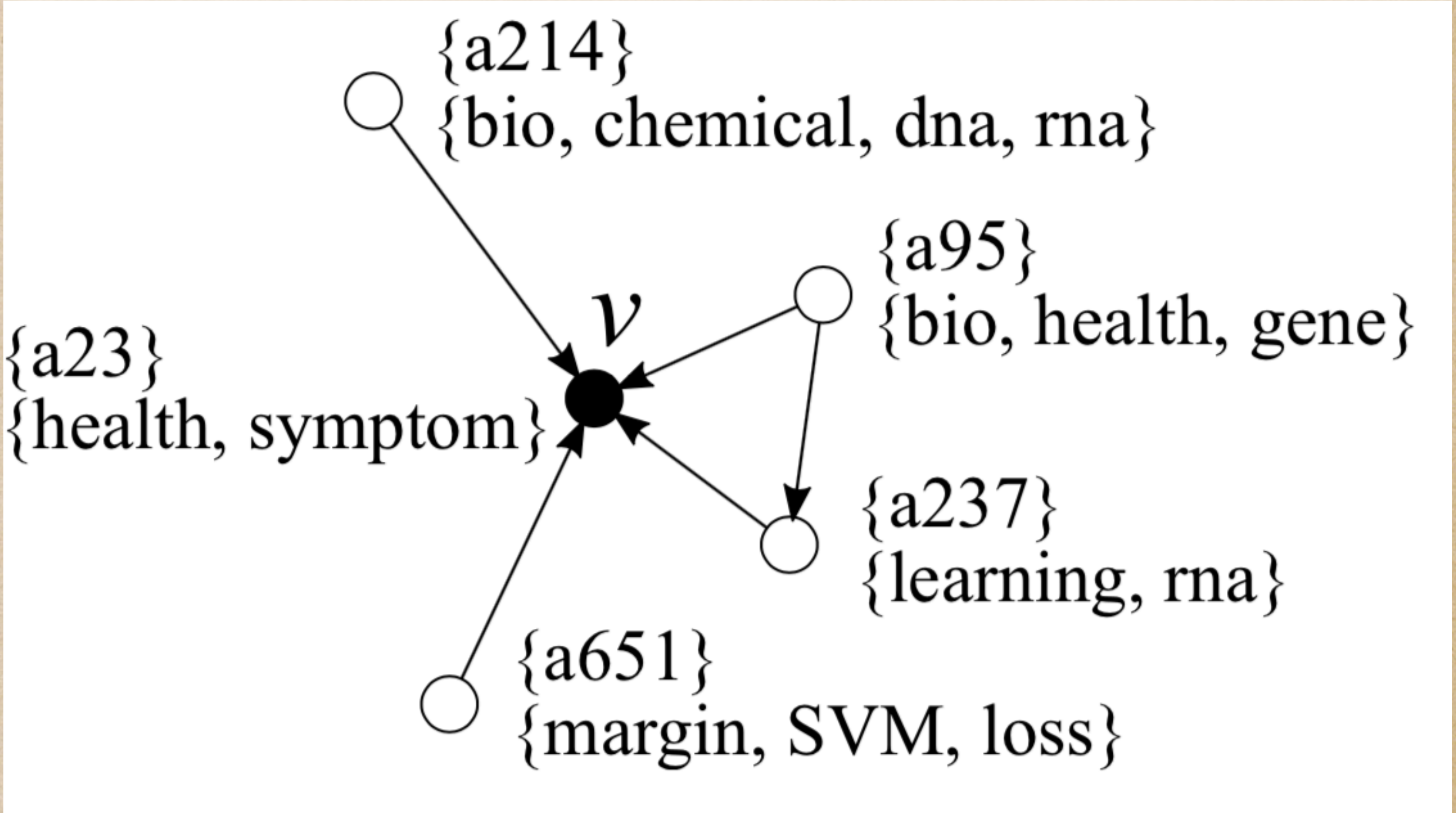
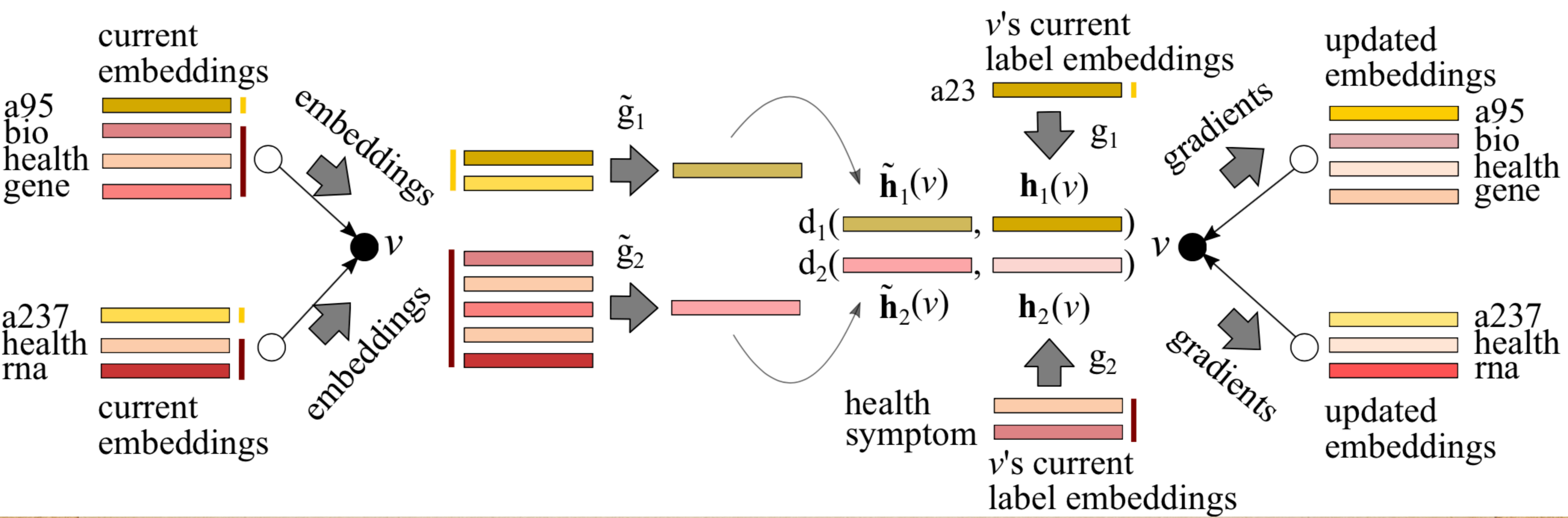


Figure 1: A fragment of a citation network.

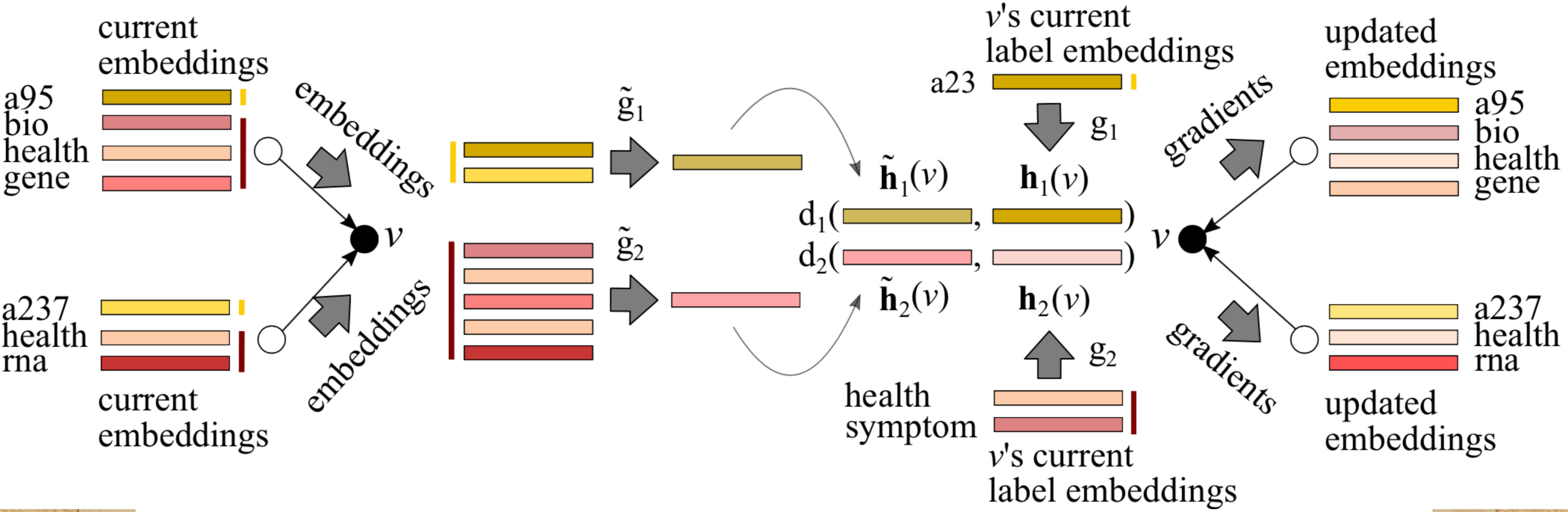
$\mathbf{N}(v)$ is the set of neighbors of v if G is undirected and the set of in-neighbors if G is directed.

EP message passing



- First, EP learns a vector representation for every label by passing messages along the edges of the input graph.
- Second, EP computes a vector representation for each vertex v from the vector representations of v’s labels.

EP message passing



Label Representations

- Let $\boldsymbol{\ell} \in \Re^d$ be the representation of label ℓ, and $\mathtt{f}$ be a differentiable embedding function
- For labels of label type i, we apply a learnable embedding function $\boldsymbol{\ell} = \mathtt{f}_i(\ell)$
- $\mathbf{h}_i(v)$ is the *embedding of label type i* for vertex v:

$$\mathbf{h}_i(v) = \mathtt{g}_i\left(\{\boldsymbol{\ell} \mid \ell \in \text{labels of type } i \text{ associated with vertex } v\}\right)$$

- $\widetilde{\mathbf{h}}_i(v)$ is the *reconstruction of the embedding of label type i* for vertex v:

$$\widetilde{\mathbf{h}}_i(v) = \widetilde{\mathtt{g}}_i\left(\{\boldsymbol{\ell} \mid \ell \in \text{labels of type } i \text{ associated with the neighbors of vertex } v\}\right)$$

$$\min \mathcal{L}_i = \min \sum_{v \in V} \mathtt{d}_i\left(\widetilde{\mathbf{h}}_i(v), \mathbf{h}_i(v)\right)$$

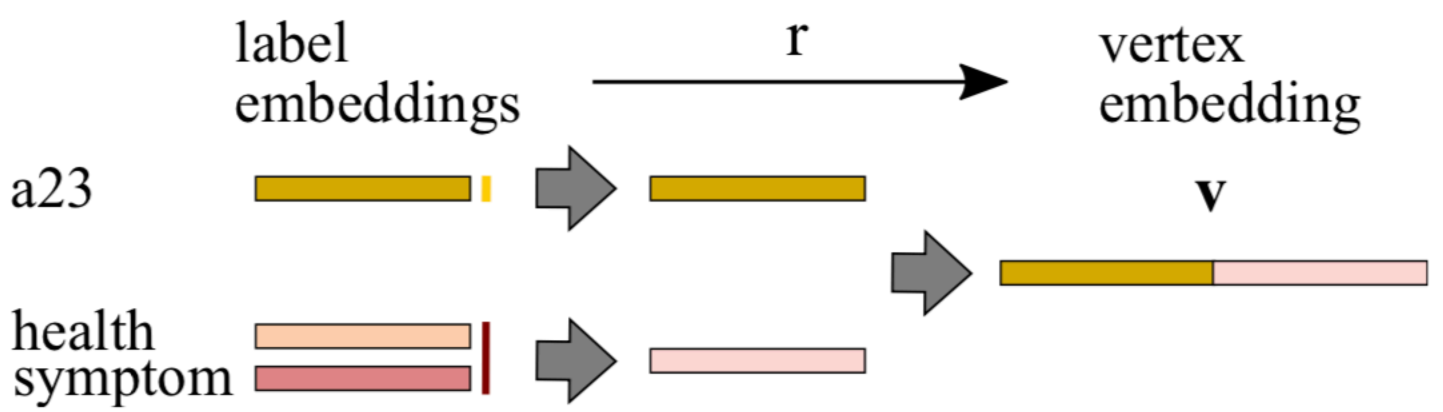


Figure 3: For each vertex v, the function $\mathtt{r}$ computes a vector representation of the vertex based on the vector representations of v's labels.

Node Representations

- Let $\mathbf{v}$ be the representation of a vertex v computed by applying a function $\mathtt{r}$ on the representations of v's labels:

$$\mathbf{v} = \mathtt{r}\left(\{\boldsymbol{\ell} \mid \ell \in \text{all labels associated with vertex } v\}\right)$$

Learning objective for each label type $i \in \{1, \ldots, k\}$:

$$\boxed{\min \mathcal{L}_i = \min \sum_{v \in V} \sum_{u \in V \setminus \{v\}} \left[\gamma + \mathtt{d}_i\left(\widetilde{\mathbf{h}}_i(v), \mathbf{h}_i(v)\right) - \mathtt{d}_i\left(\widetilde{\mathbf{h}}_i(v), \mathbf{h}_i(u)\right)\right]_+}$$

where $\mathtt{d}_i$ is the Euclidean distance, $[x]_+$ is the positive part of x, and $\gamma > 0$ is a margin hyperparameter.

A simple instance of EP

EP-B

- $\mathtt{f}_i$ are embedding lookup tables
- $\mathrm{g}_i(\mathbf{H}) = \widetilde{\mathrm{g}}_i(\mathbf{H}) = \frac{1}{|\mathbf{H}|}\sum_{\mathbf{h}\in\mathbf{H}} \mathbf{h}$ for all label types i
- $\mathbf{v} = \mathtt{concat}\left[\mathbf{h}_1(v), ..., \mathbf{h}_k(v)\right]$
- **Inductive** setting: $\mathbf{v} = \mathtt{concat}\left[\widetilde{\mathbf{h}}_1(v), ..., \widetilde{\mathbf{h}}_k(v)\right]$

For a graph without node attributes.

Method	#params	#hyperparams
DEEPWALK [Perozzi et al.,2014]	$2d\|V\|$	4
NODE2VEC [Grover et al.,2016]	$2d\|V\|$	6
LINE [Tang et al.,2015]	$2d\|V\|$	2
PLANETOID [Yang et al., 2016]	$\gg 2d\|V\|$	≥ 6
EP-B	$d\|V\|$	2

Advantages of EP

- Unsupervised
- Focus on Attributes but handle it easier
- Applied to large scale graphs
- an update of a label embedding affects neighboring label embeddings which, in other updates, affects their neighboring label embeddings

Experiment

EVALUATION

Node classification:

- The input is a graph (with or without attributes)
- A fraction of the nodes is assigned a class label
- The output is an assignment of class labels to the test nodes
- EP is classifier agnostic
- In fairness to existing unsupervised works, we opt for logistic regression trained with node representations **v**

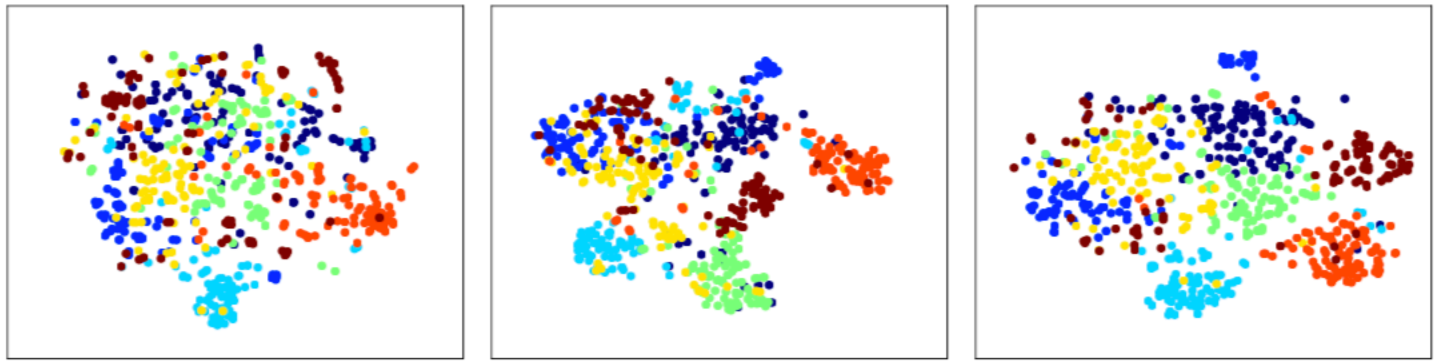
Transductive setting

Method	Cora	Citeseer	Pubmed
EP-B	78.0% ± 1.5%	**71.0% ± 1.3%**	**79.6% ± 2.1%**
DW+BoW	76.1% ± 2.1%	61.9% ± 2.3%	77.8% ± 2.2%
Planetoid-T	71.9% ± 5.3%	58.6% ± 6.3%	74.5% ± 4.9%
GCN	**79.6% ± 2.0%**	69.2% ± 1.2%	77.3% ± 2.7%
DeepWalk	71.1% ± 2.7%	47.6% ± 2.3%	73.5% ± 3.0%
BoW Feat	58.6% ± 0.7%	58.1% ± 1.7%	70.5% ± 2.9%

Inductive setting

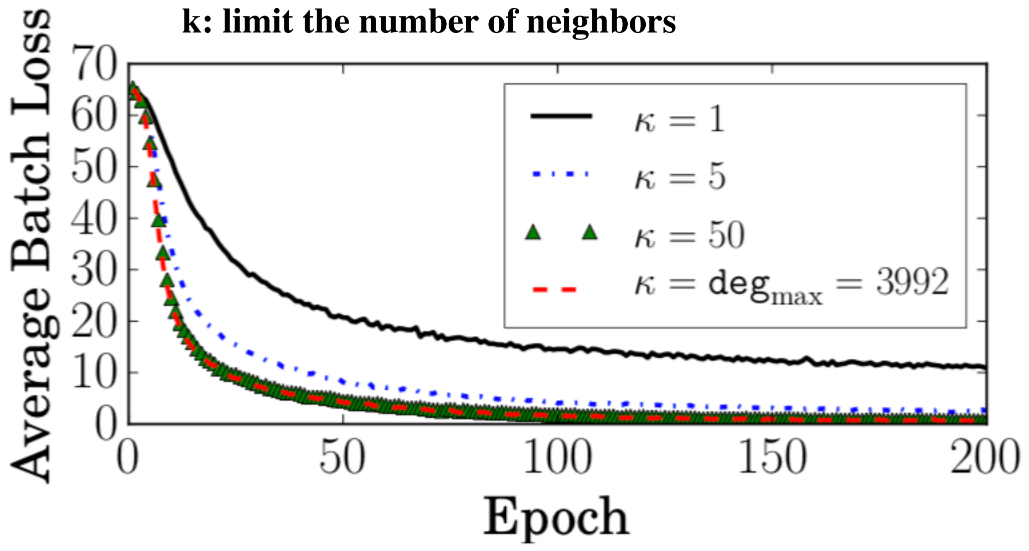
Method	Cora	Citeseer	Pubmed
EP-B	**73.1% ± 1.7%**	**68.6% ± 1.7%**	**79.9% ± 2.3%**
DW-I+BoW	68.3% ± 1.7%	59.5% ± 2.5%	74.9% ± 1.2%
Planetoid-I	64.8% ± 3.7%	62.0% ± 3.8%	75.7% ± 4.2%
GCN-I	67.7% ± 2.1%	63.4% ± 1,0%	73.5% ± 2.5%
BoW Feat	58.6% ± 0.7%	58.1% ± 1.7%	70.5% ± 2.9%

The transductive setting is the setting where all nodes of the input graph are present during training. In the inductive setting, a certain percentage of the nodes are not part of the graph during unsupervised learning.



(a)

k: limit the number of neighbors



(b)

Thanks and QA