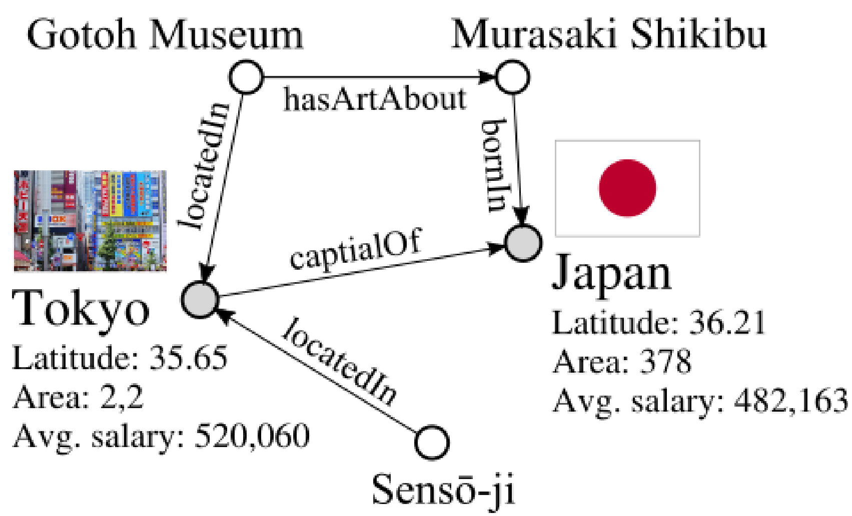
Neural Representation Learning for Graphs

Mathias Niepert

NEC Labs Europe
Heidelberg



Leuven, Belgium
December 11th

NEC Labs Europe: What do we do?

- ~ 80 researchers, 22 nationalities
- Research lab, no product development
- Main objectives:
  1. Research output for top tier conferences
  2. Stable prototypes for technology transfer
  3. Patent applications
- Product prototypes based on lab’s research

Research Collaborations

NEC Japan (business units and central labs)

- Digital Health
- Retail
- Finance
- Networked Systems

EU Projects

- Exploration of applications not coming from NEC
- Foster collaborations with research community
- Understand trends and problems in the SME market

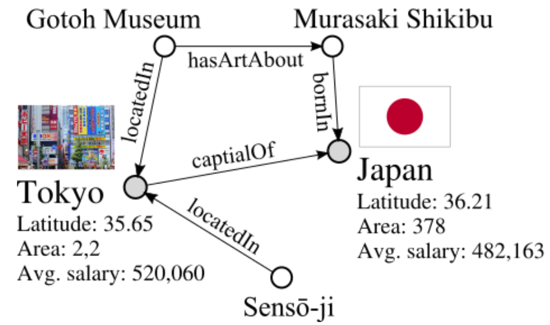
Third party Collaborations

- DKFZ
- University of Heidelberg medical school

Main Research Themes

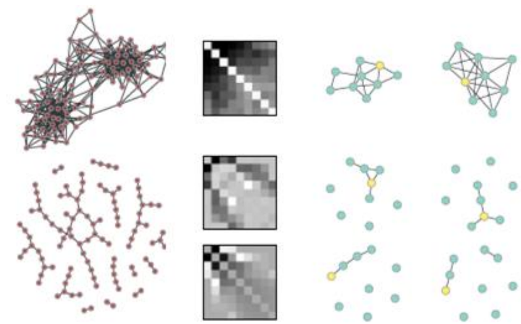
Multi-Modal Learning and Reasoning

- Combining different attribute types and modalities
- **Knowledge graphs for multi-modal learning** (combining deep learning and logical reasoning)



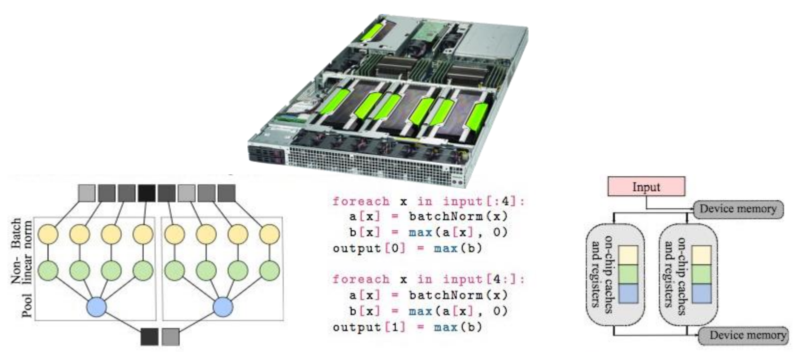
Graph-based (Relational) Machine Learning

- Learning graph representations
- **Unsupervised and semi-supervised learning**



Systems and ML

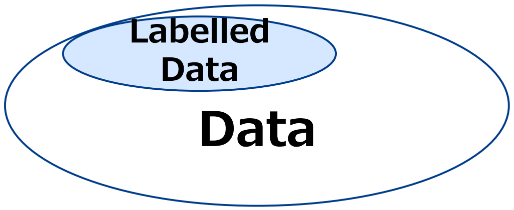
- ML for Systems and Systems for ML
- CPU/GPU/network optimizations etc.
- Deep learning for data networks



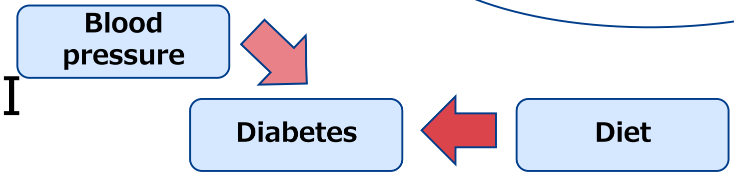
NEC

Technological Challenges

- ML that works without much labelled data (unsupervised and semi-supervised learning)



- Interpretable and Explainable AI



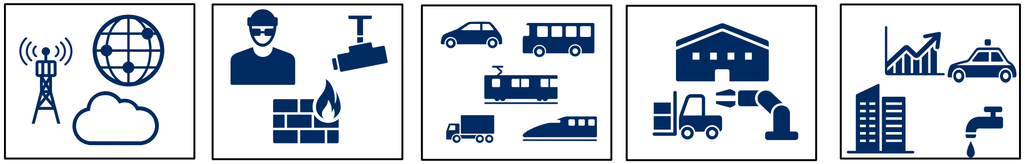
- Ability to combine different data modalities (data integration, multi-modal learning)

Latitude: 35.65

Tokyo (Japanese: [toːkjoː] (listen), English: /ˈtoʊki.oʊ/), officially **Tokyo Metropolis**,[6] is the capital of Japan and one of its 47 prefectures.[7]

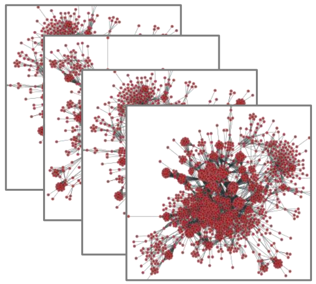
- Efficiency and support of real time predictions (network speed if required)

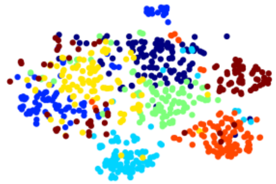
- Applicable to several business use cases (horizontal technology)



Graph-Based Machine Learning

Learn representations for entire graphs

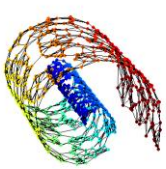


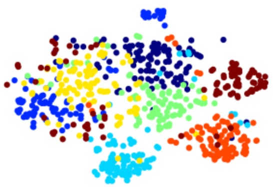


Graph classification/
regression problems

Learn representations for nodes

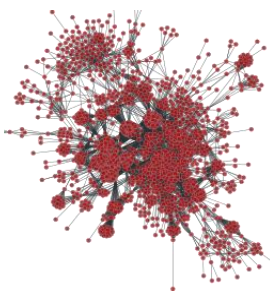
Induce
graph

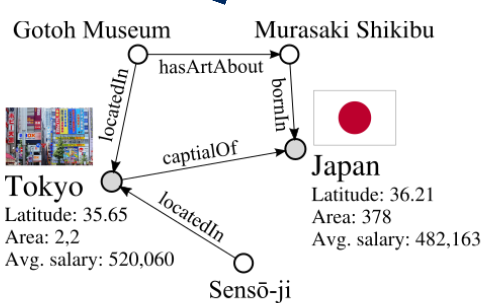




Node classification/
regression problems

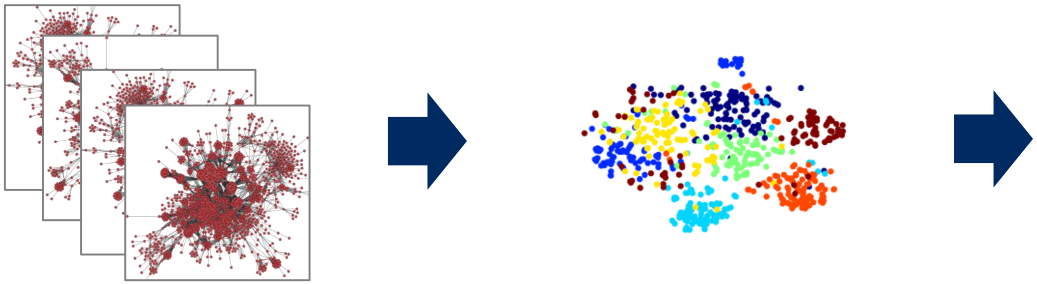
Link prediction
problems



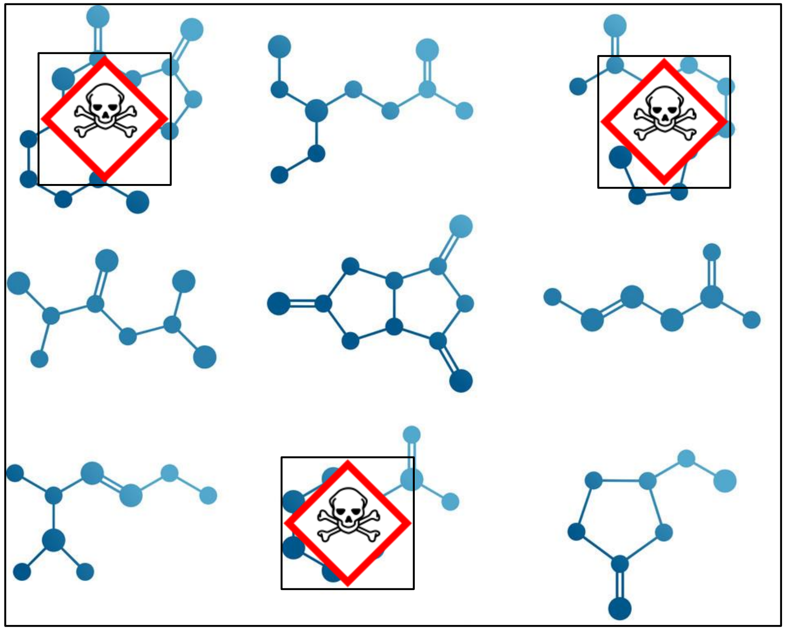


Example Applications – Drug Discovery

Learn representations for entire graphs

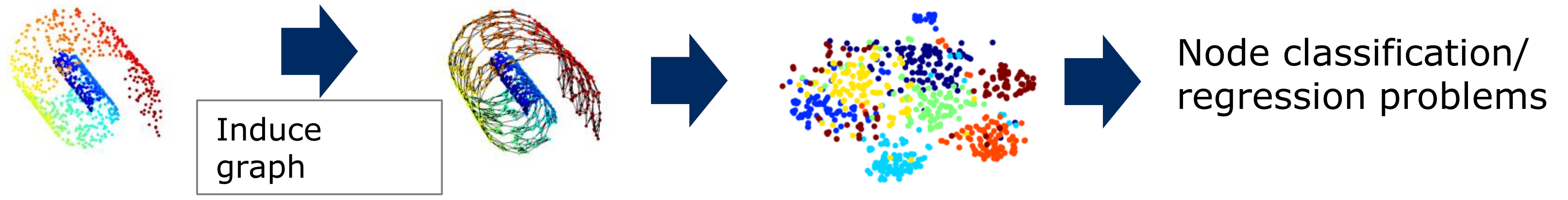


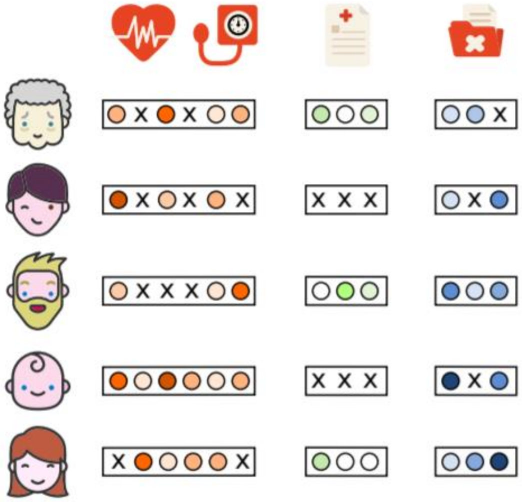
Graph classification/
regression problems

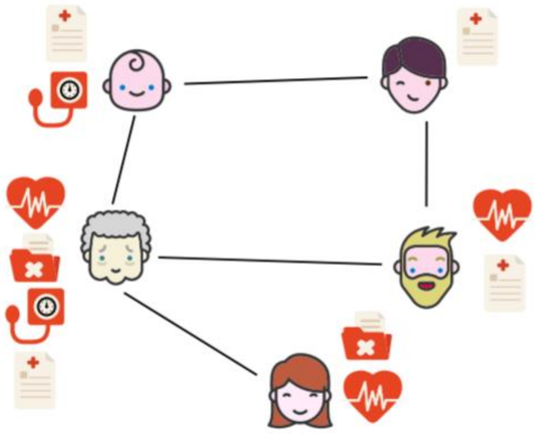


Example Applications – Patient Outcome Prediction

Learn representations for nodes

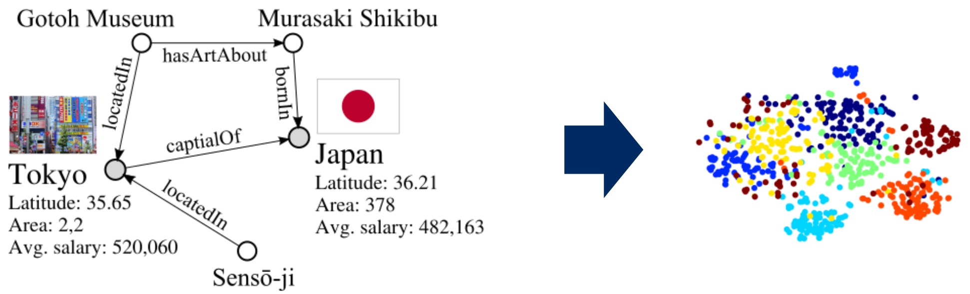




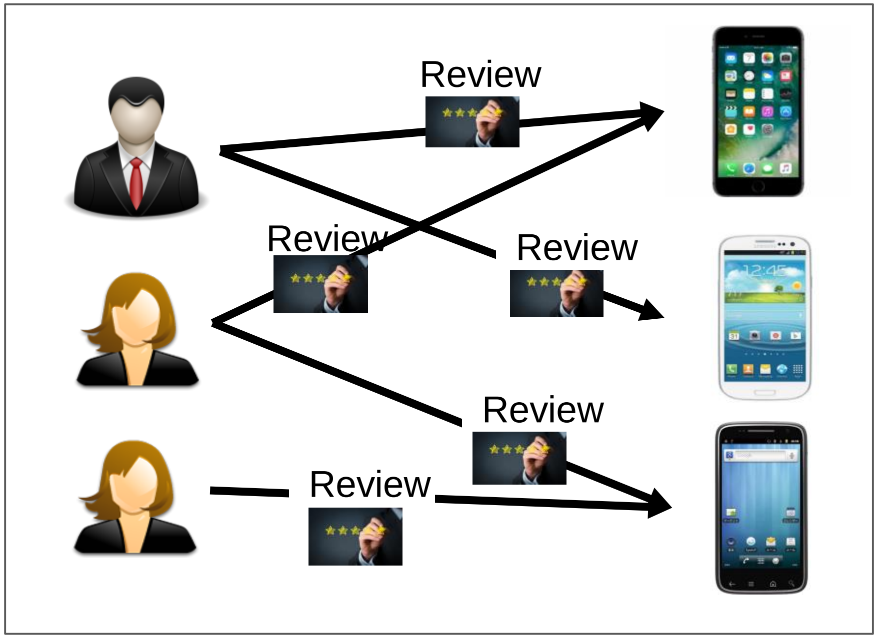


Example Applications – Recommender Systems

Learn representations for nodes

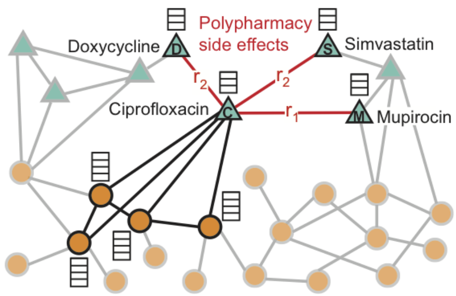


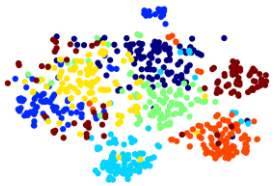
Link prediction problem



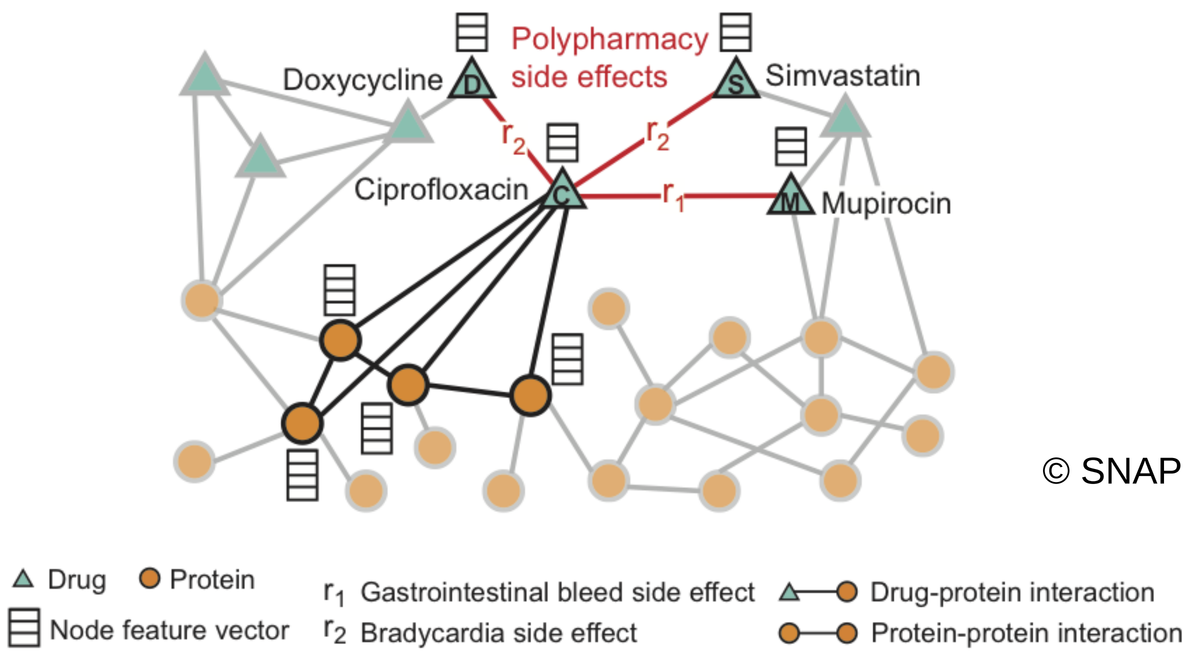
Example Applications – Polypharmacy Prediction

Learn representations for nodes



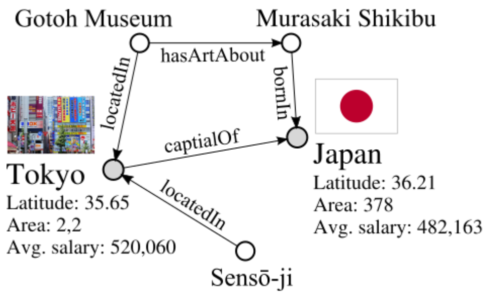


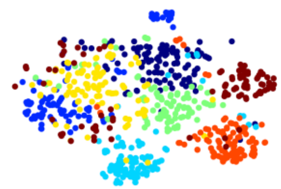
Link prediction problem



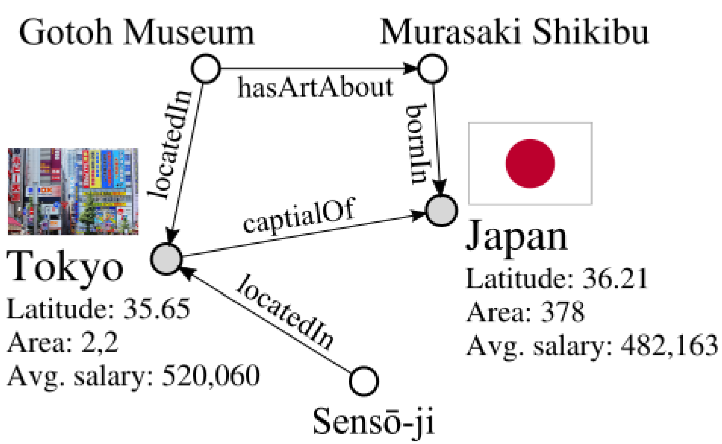
Example Applications – Knowledge Base Completion

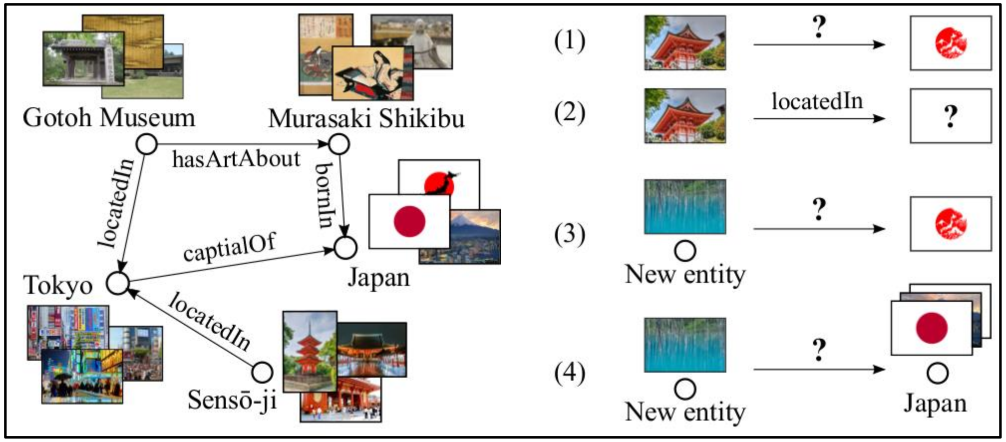
Learn representations for nodes





Link prediction problem





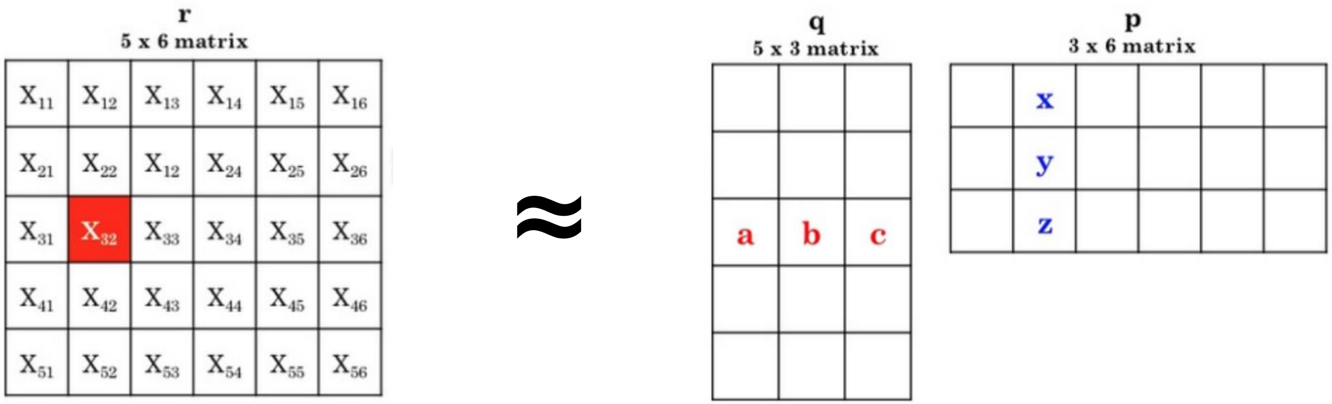
Outline of the First Part of our Lecture

1. Basic Concepts

2. Two Perspectives on Learning from Graphs
   - Knowledge Graph = Tensor (KB completion, evaluation, etc.)
   - Learning from Local Structure (learning from paths and neighborhoods)

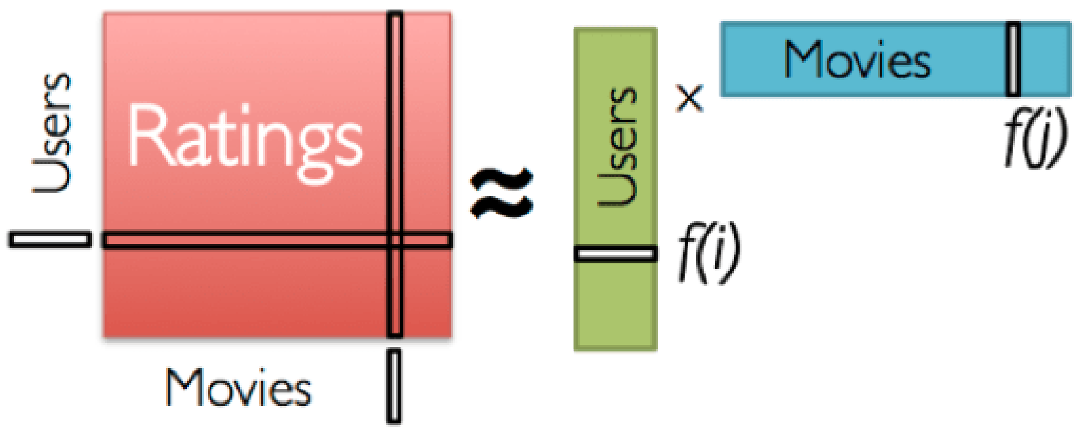
3. Some Practical Observations

Matrix Factorization



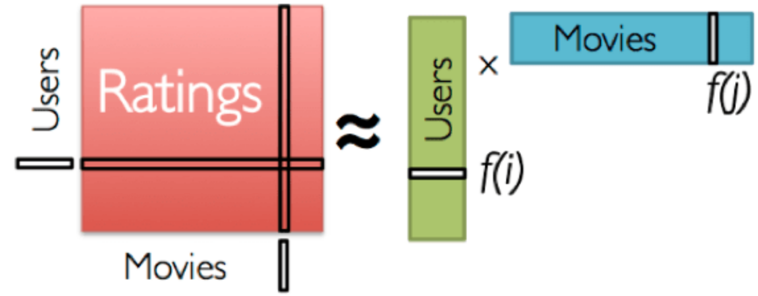
NETFLIX

amazon



The Differential Programming Approach

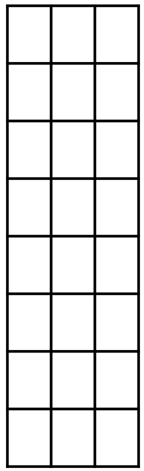
- **Step 1:** Assume users and movies are represented with one-hot encoding and define **encoding** function f for users and movies



Parameters

One-hot encoding

Embedding (dimension size=3)

[0 0 1 0 0 0 0 0] ×  = [0.2 0.9 -1.6]

- **Step 2:** Define **scoring function** between user-movie pairs

$$\text{Score} = \begin{bmatrix} 0.2 \\ 0.9 \\ -1.6 \end{bmatrix} \bullet \begin{bmatrix} 0.8 \\ -1.2 \\ 0.5 \end{bmatrix} = [-1.72]$$

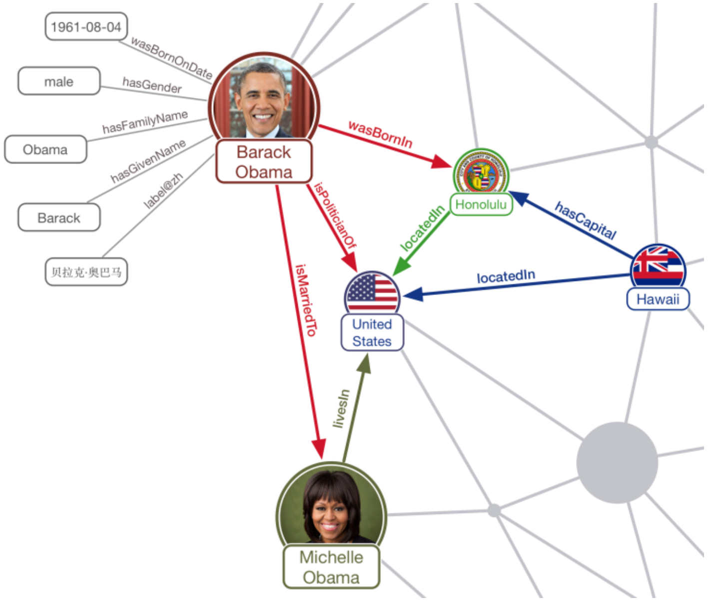
- **Step 3:** Define a **loss** between scorings and actual existing user ratings

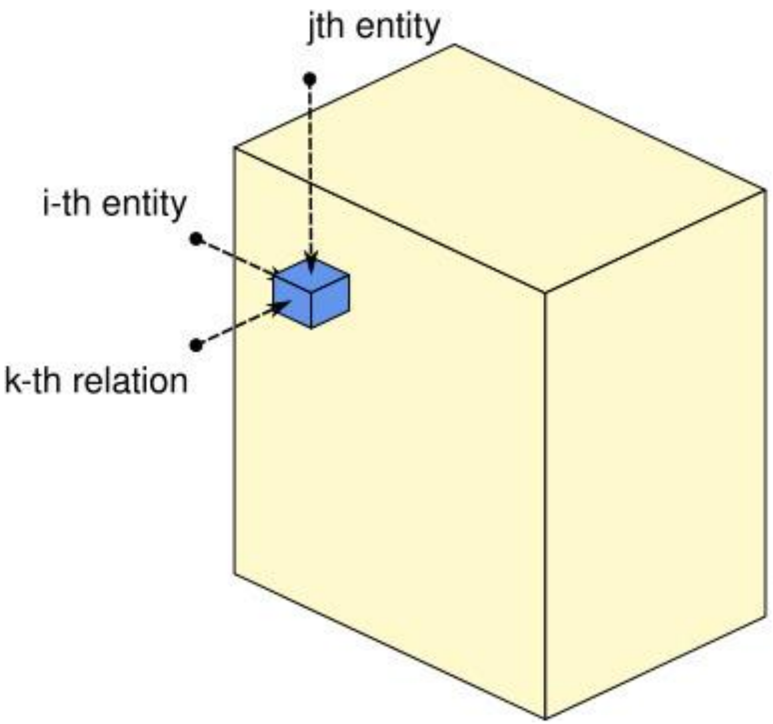
$$\text{Loss} = (-1.72 - 3)^2$$

Observed rating

- **Step 4:** Apply **gradient decent** to train the model “end-to-end”

1. The multi-relational graph as a **3D tensor**

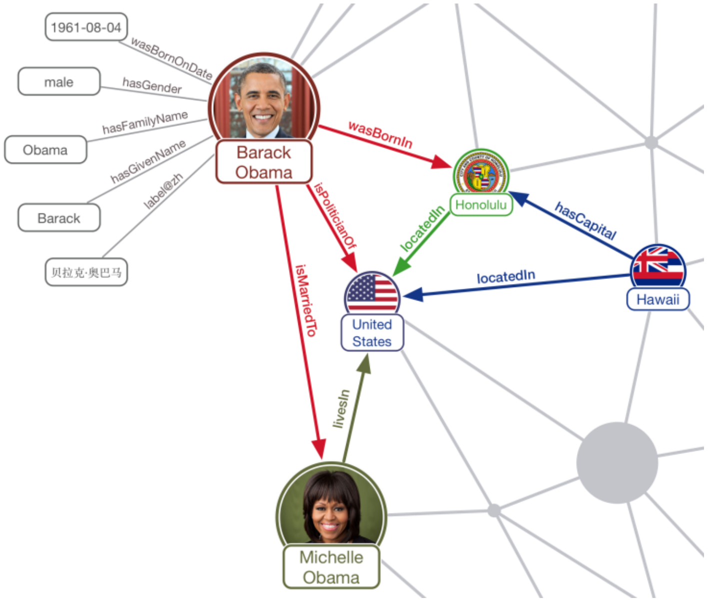


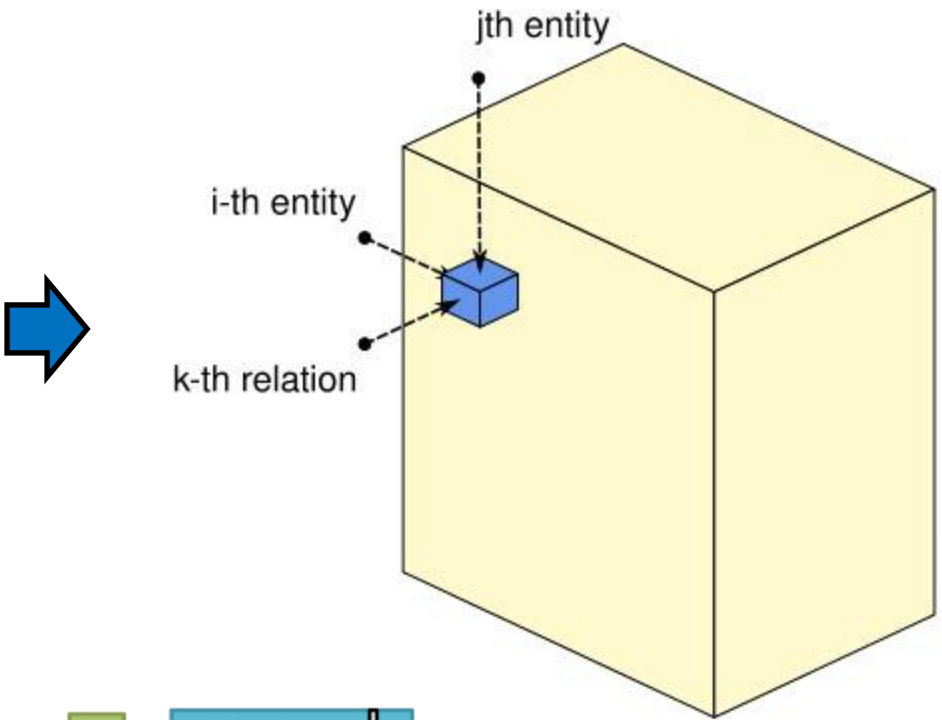


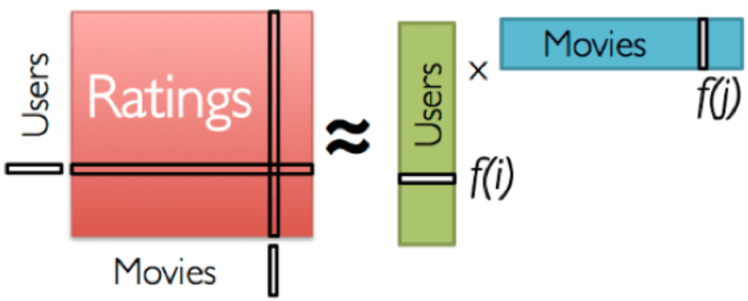
© Maximilian Nickel

Two Perspectives on Learning from Graph Data

1. The multi-relational graph as a **3D tensor**



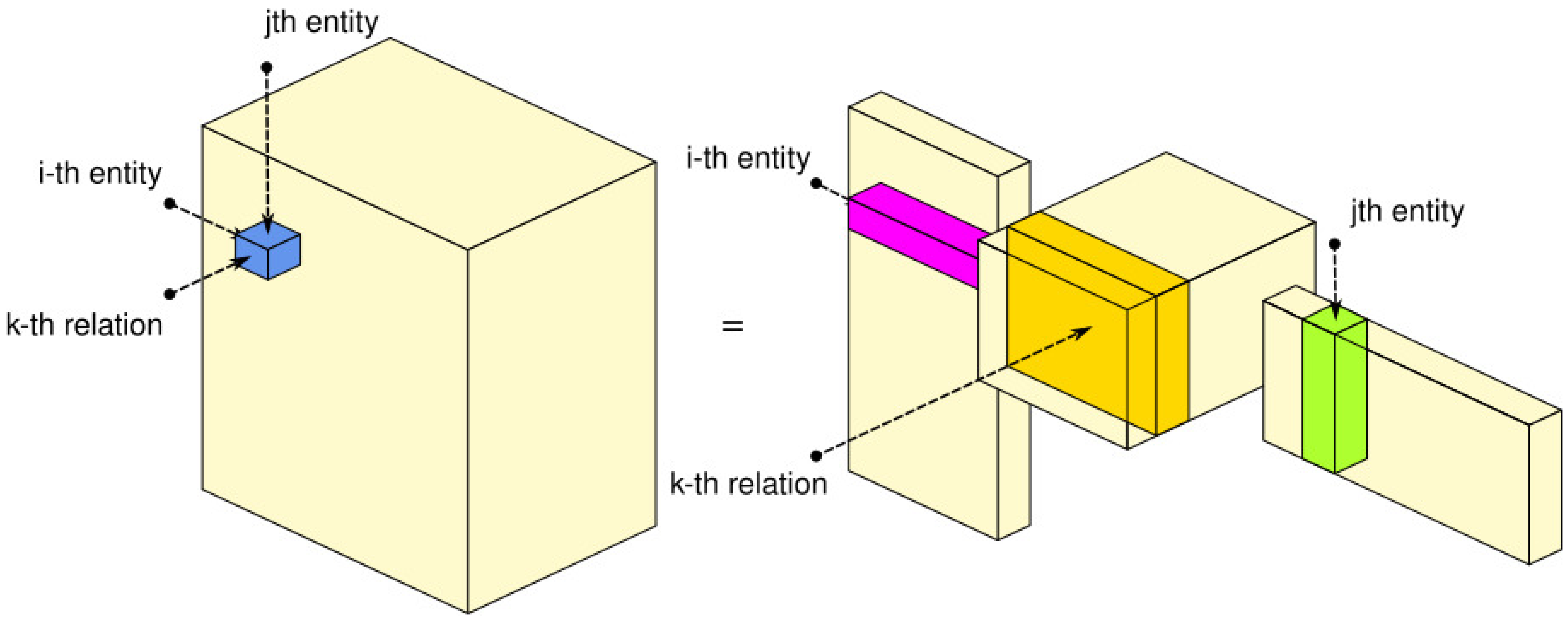




RESCAL

Nickel et al., 2011

1. The multi-relational graph as a **3D tensor**

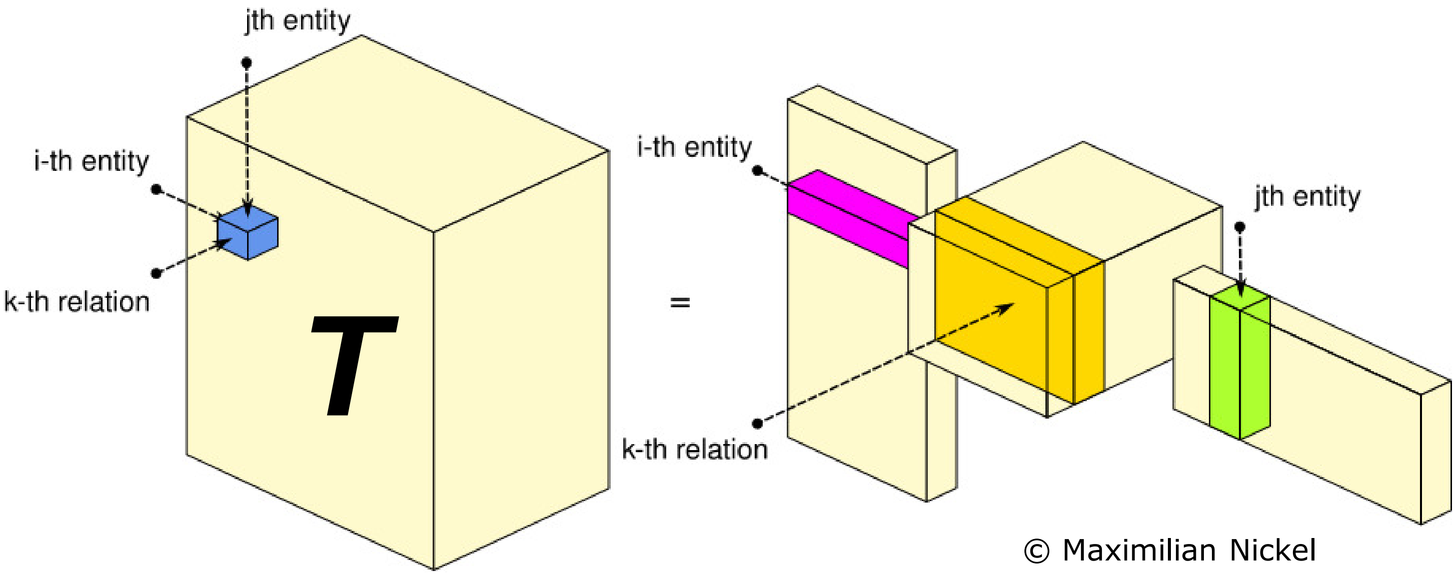


© Maximilian Nickel

Nickel et al, A Three-Way Model for Collective Learning on Multi-Relational Data, 2011

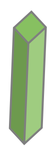
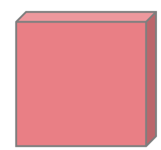
RESCAL

Nickel et al., 2011



© Maximilian Nickel

- **Step 1:** Choose the representation (encoding) for entities and relations

Entities: $\boldsymbol{e}_i =$  Relation types: $\boldsymbol{W}_r =$ 

- **Step 2:** Choose scoring function for triples (h, r, t) = coordinates in the 3D tensor

$$s(h,r,t) = \boldsymbol{e}_h^T \cdot \boldsymbol{W}_r \cdot \boldsymbol{e}_t$$

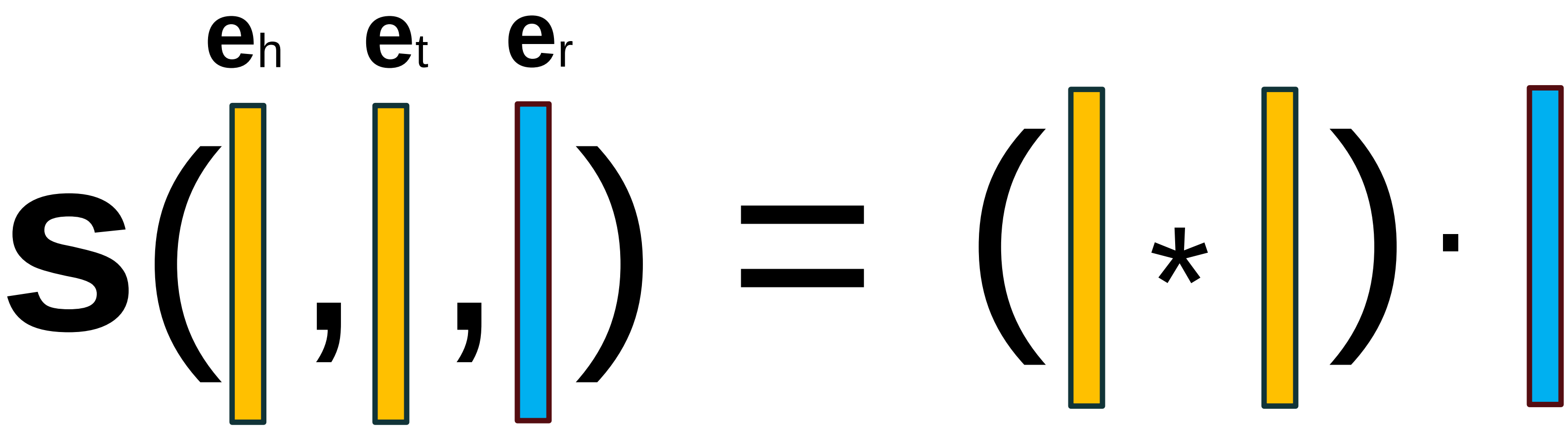
- **Step 3:** Choose loss function

$$\sum_{h,r,t} (\boldsymbol{T}_{\{h,r,t\} - s(h,r,t)})^2$$

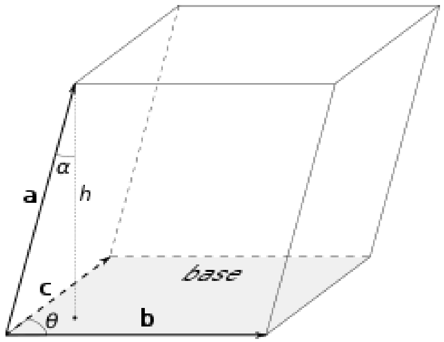
Yang et al., 2015

- **DistMult:** well-performing KB embedding method
- Simplifies RESCAL; relation matrix only non-zero in diagonal

Triple: (h, r, t)

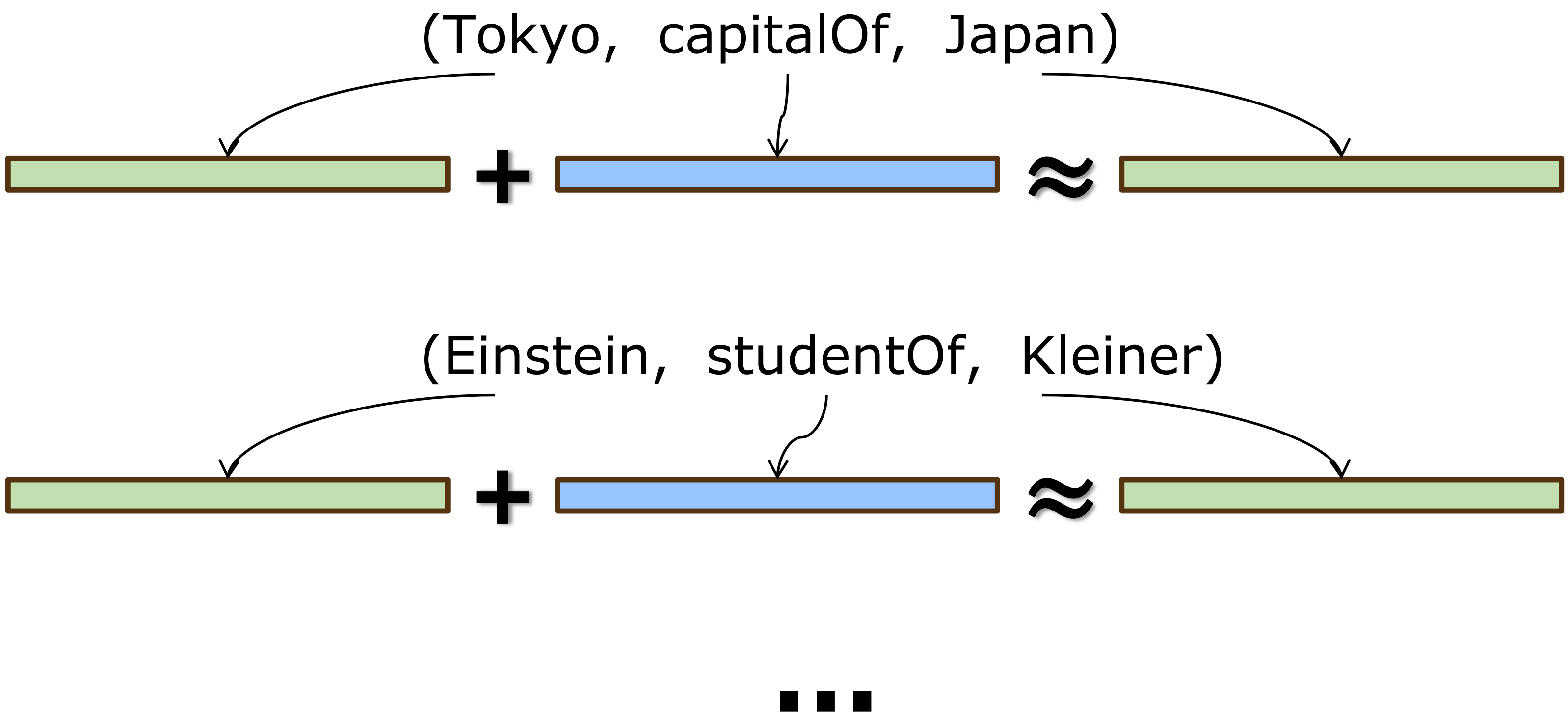


- **Geometric interpretation:** Absolute value is the volume of the 3D parallelogram spanned by the three vectors



TransE – Relation Types as Translations

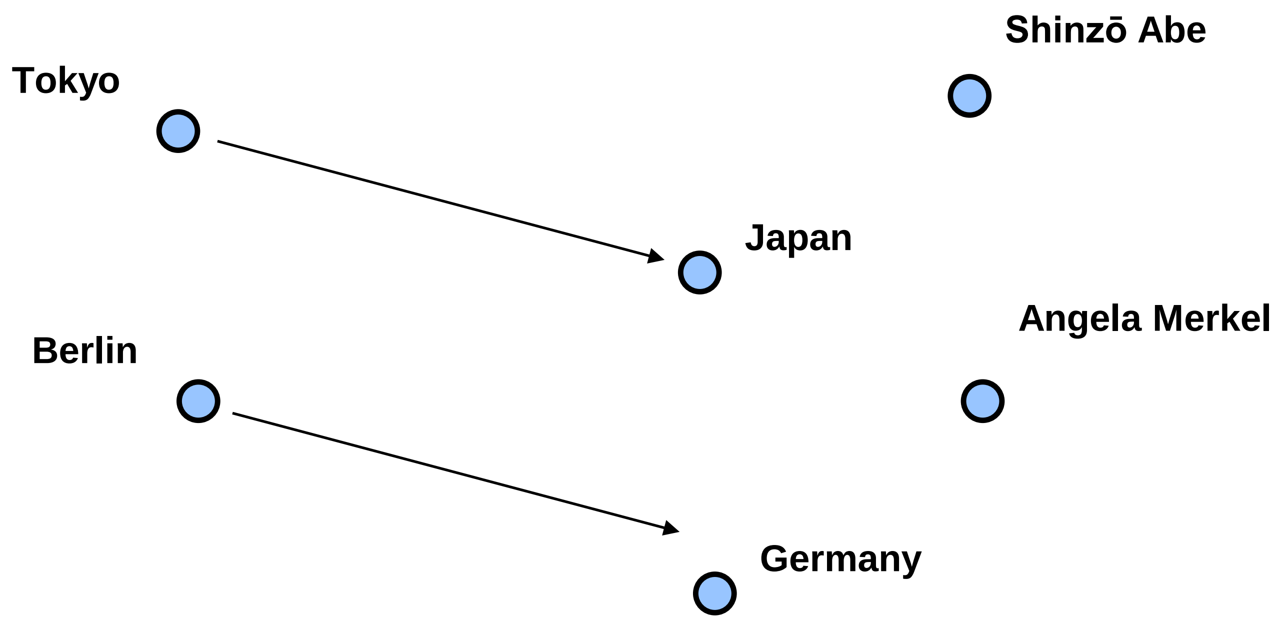
- **TransE** learns embeddings of entities and relations



TransE – Relation Types as Translations

Bordes et al., 2013

- **TransE** learns embeddings of entities and relations

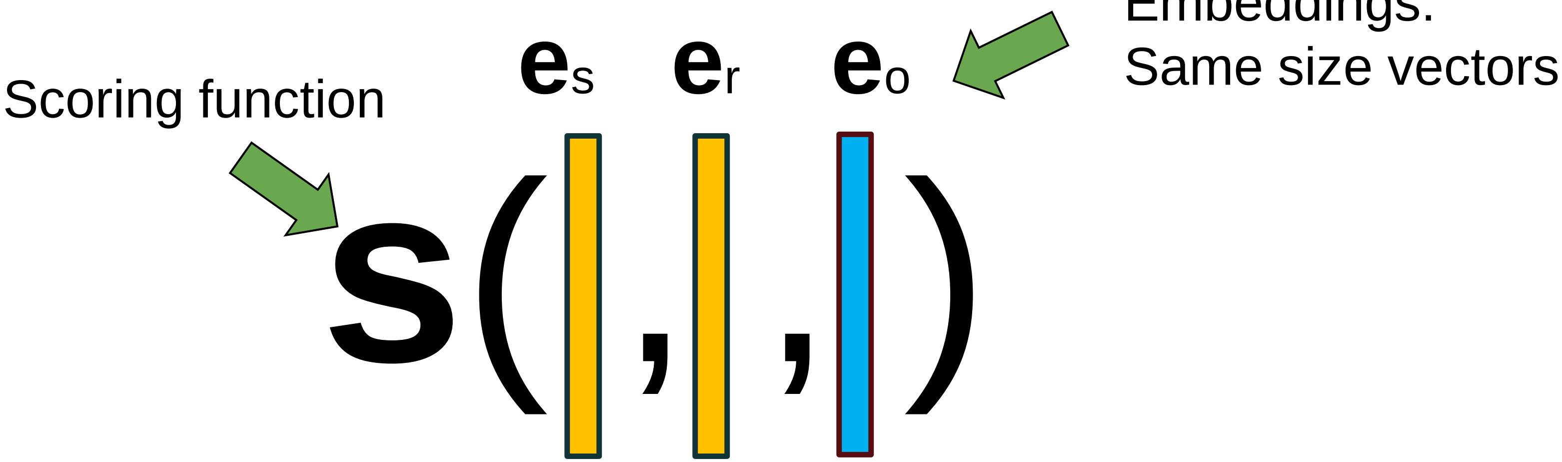


- **Geometric interpretation:** Relation vector translates (moves) head entity embedding to tail entity embedding

Knowledge Graph Representations

- Many alternative scoring functions have been proposed

Triple: (s, r, o)



Model	Scoring Function	Relation parameters
RESCAL (Nickel et al., 2011)	$e_s^T W_r e_o$	$W_r \in \mathbb{R}^{K^2}$
TransE (Bordes et al., 2013b)	$\|\|(e_s + w_r) - e_o\|\|_p$	$w_r \in \mathbb{R}^K$
NTN (Socher et al., 2013)	$u_r^T f(e_s W_r^{[1..D]} e_o + V_r \begin{bmatrix} e_s \\ e_o \end{bmatrix} + b_r)$	$W_r \in \mathbb{R}^{K^2 D}, b_r \in \mathbb{R}^K$ $V_r \in \mathbb{R}^{2KD}, u_r \in \mathbb{R}^K$
DistMult (Yang et al., 2015)	$< w_r, e_s, e_o >$	$w_r \in \mathbb{R}^K$
HolE (Nickel et al., 2016b)	$w_r^T(\mathcal{F}^{-1}[\overline{\mathcal{F}[e_s]} \odot \mathcal{F}[e_o]])$	$w_r \in \mathbb{R}^K$
ComplEx	$\mathrm{Re}(< w_r, e_s, \bar{e}_o >)$	$w_r \in \mathbb{C}^K$

Trouillon et al. 2016

Knowledge Graph Embeddings

What do they actually learn?

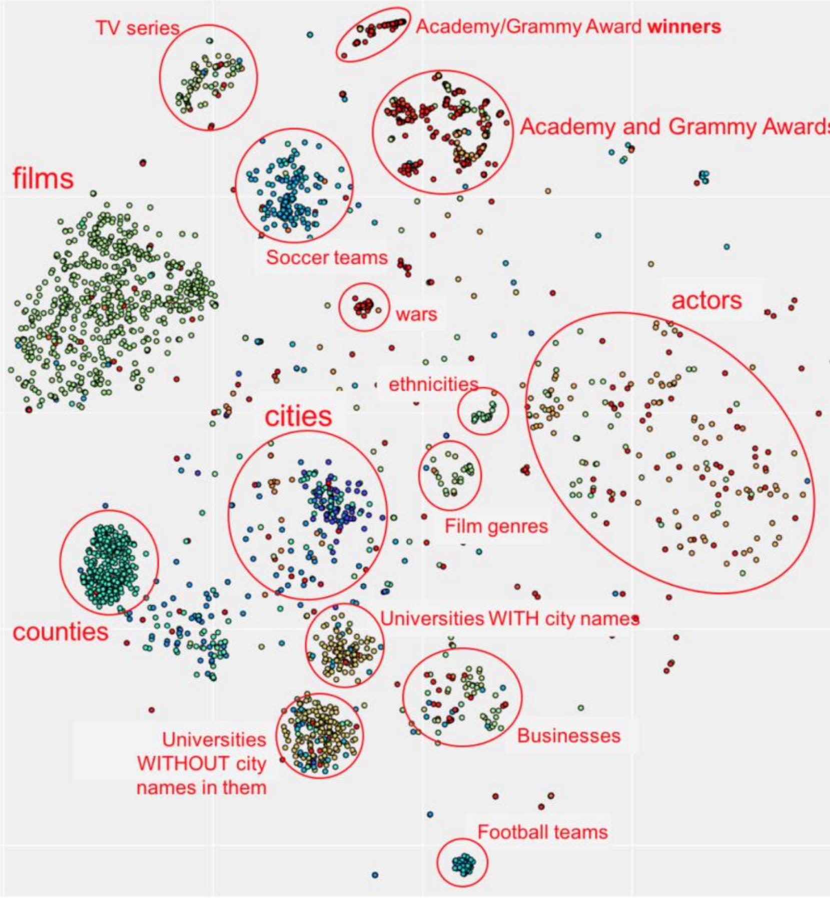
- Fine grained **latent types** of entities
- Latent representation of relation types

What do they not learn?

- Relational rules with **constants**
- E.g., relation true if married to PersonX
- Approximate vs. exact entity type

Majority of KB embedding approaches are outperformed by simple relational baselines

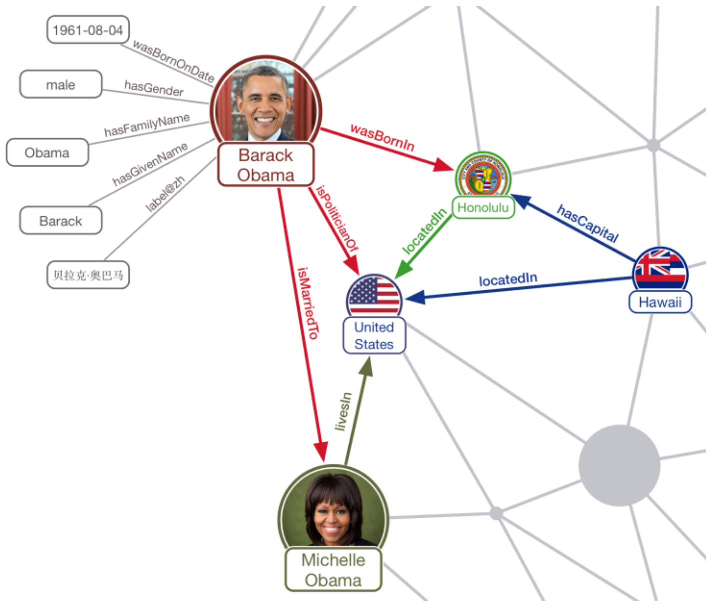
- First observed by Toutanova et al, 2015
- Holds true for dense KBs (e.g. FB15k) but not for sparser ones (e.g., FB15k-237)
- Embedding methods outperform purely relational models on sparse KBs



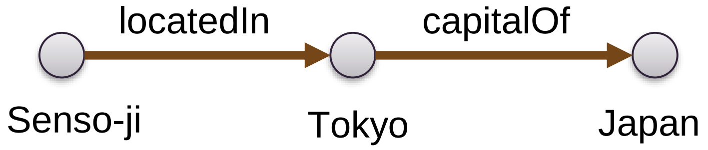
© Corby Rosset

Two Perspectives on Learning from Graph Data

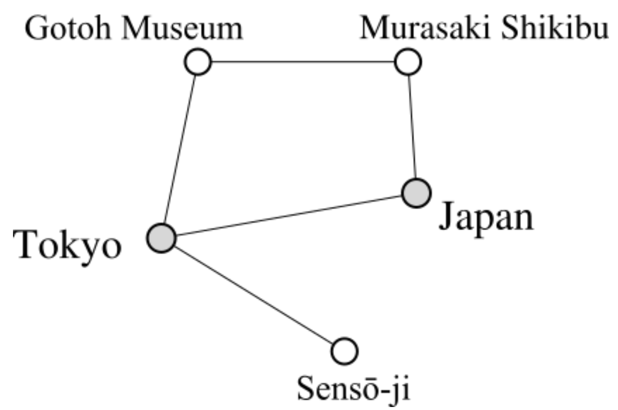
2. Learning from Local Graph Structures



Paths / random walks



Local Neighborhoods



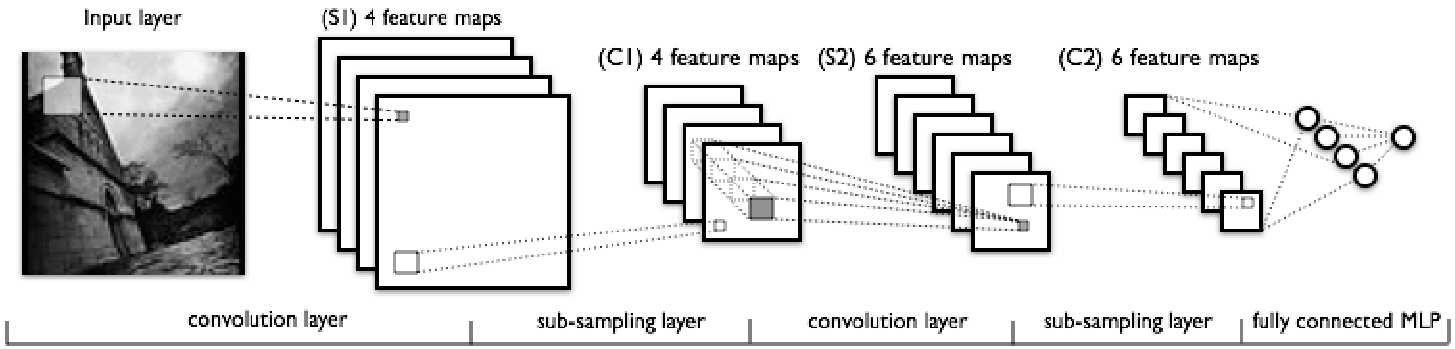
NB: Learning from local structures can capture global properties through a recursive propagation process between nodes

Representation Learning for Knowledge Graphs

Observation: Effective representations are often composed bottom-up from **local** representations

- Weight sharing
- Hierarchical features
- Model tractability

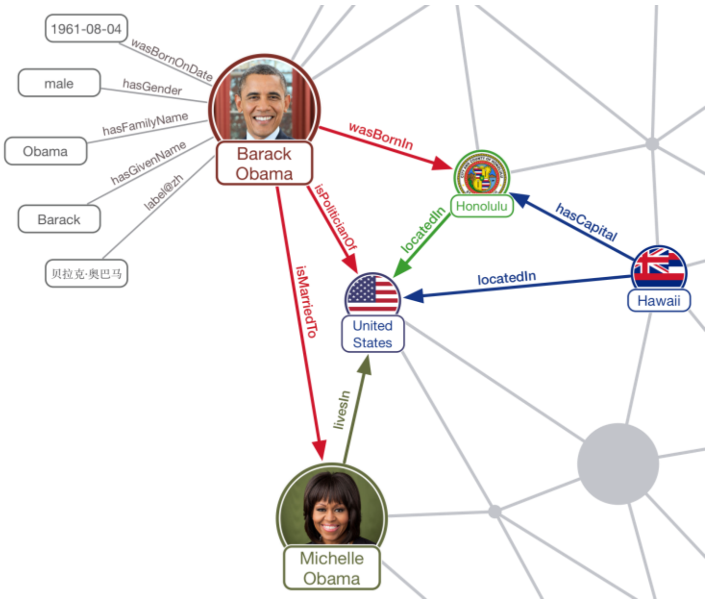
Example: Convolutional neural networks



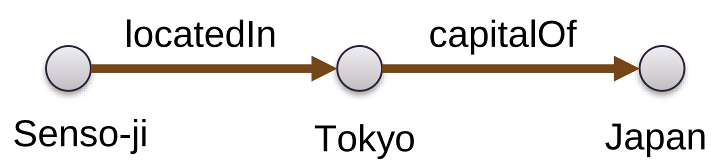
Question: What is a suitable notion of **locality** in knowledge graphs?

Learning From Random Walks and Paths

- Basic idea: **Mine frequent paths** in the graph and use these paths as features for some learning method

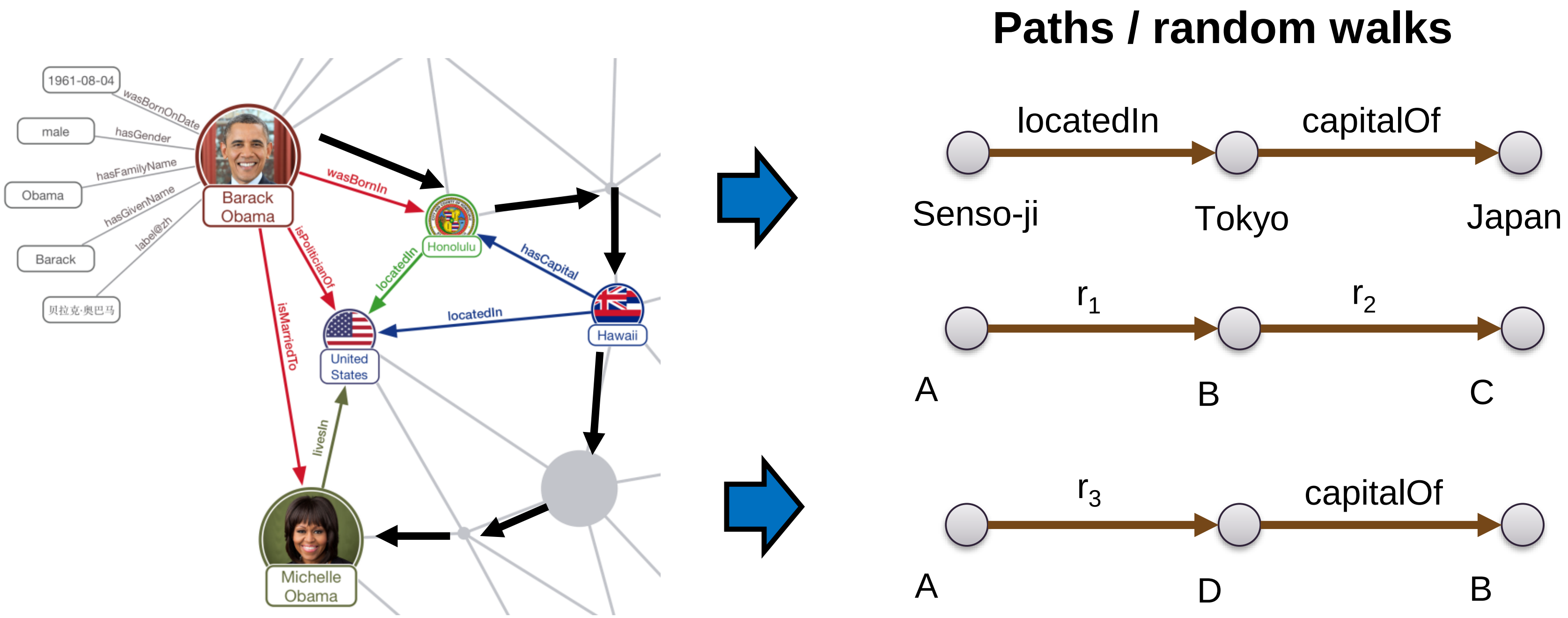


Paths / random walks



Methods for Path Extraction

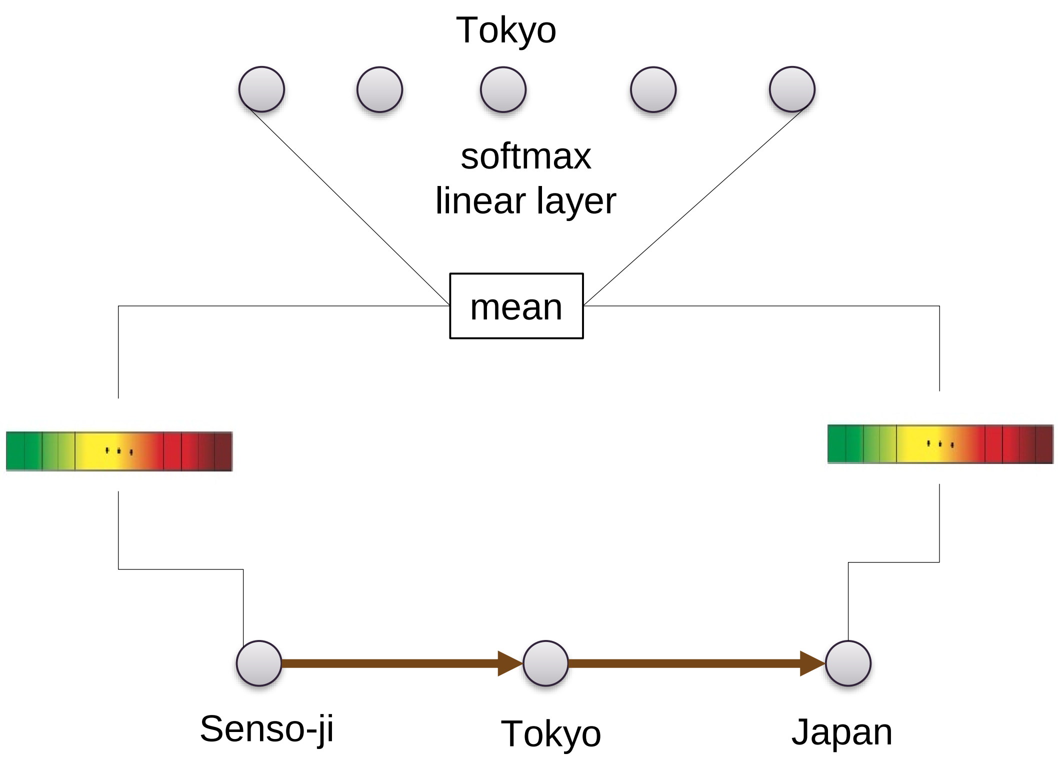
- Perform a large number of **Random Walks**



- Keep the paths most frequently encountered

Methods for Learning from Single-Relational Paths

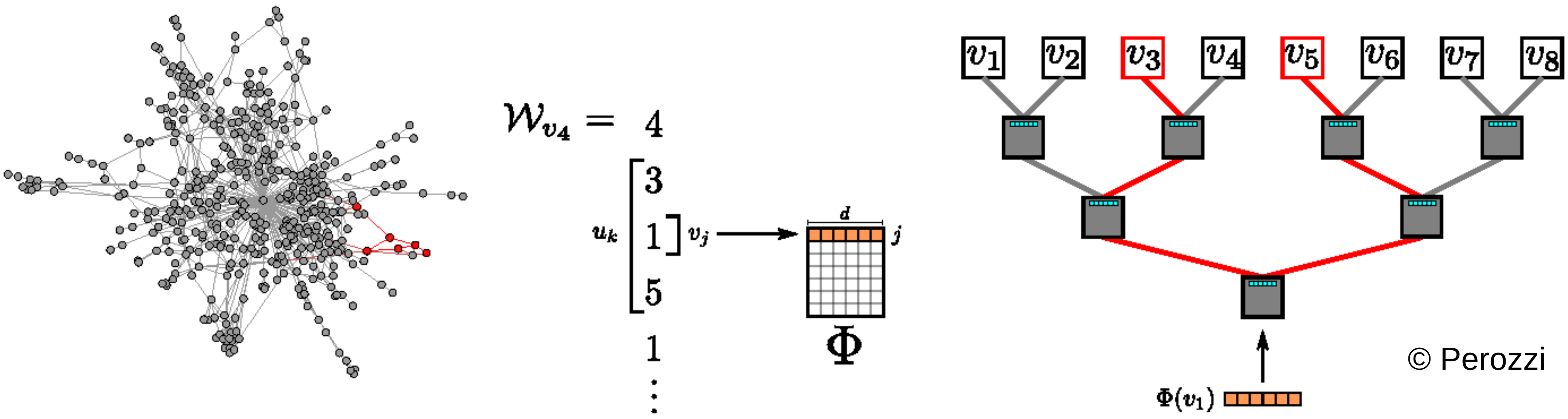
- Interpret every walk as a sentence (sequence of nodes visited)
- Train word embedding method such as Word2vec



Continuous bag of nodes

DeepWalk

Perozzi et al., 2014



(a) Random walk generation.

(b) Representation mapping.

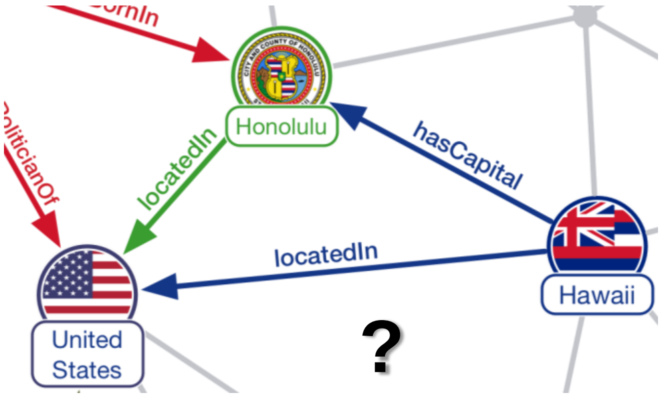
(c) Hierarchical Softmax.

Skip-gram model

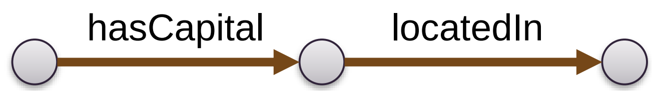
Results in node embeddings to be used for other tasks

Methods for Learning from Multi-Relational Paths

- Interpret every walk as a logical rule:
  "If path is present, then set feature to 1"
- Combine these features with simple classifier such as logistic regression



Common path

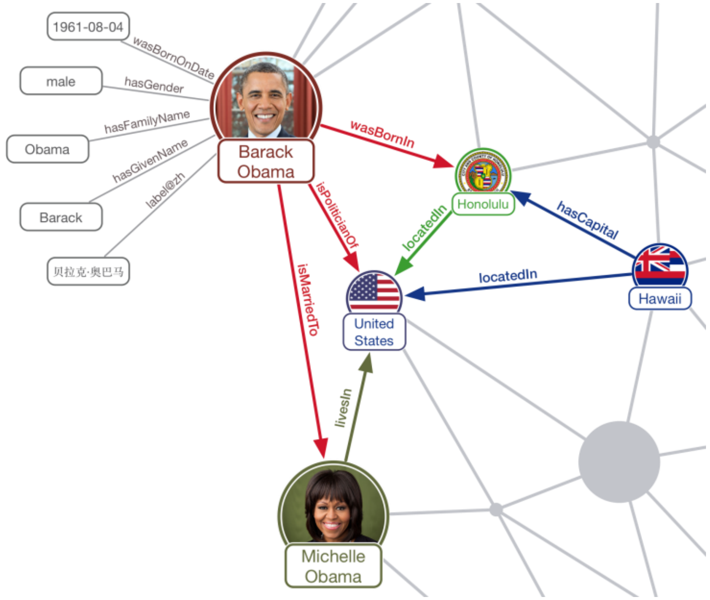


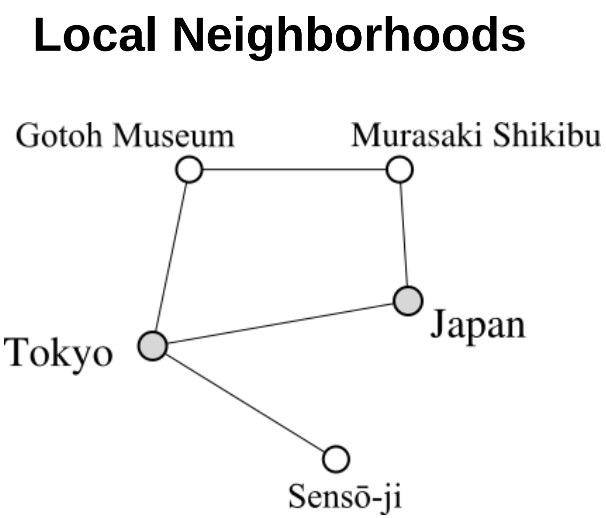
Good feature to predict "locatedIn"

Lao and Cohen, Path Ranking Algorithm, 2010

Two Perspectives on Learning from Graph Data

2. Learning from Local Graph Structures



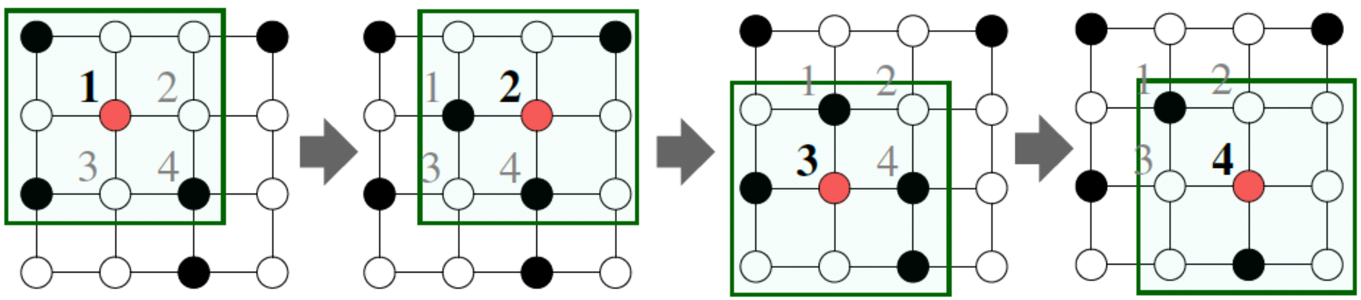


NB: Learning from local structures can capture global properties through a recursive propagation process between nodes

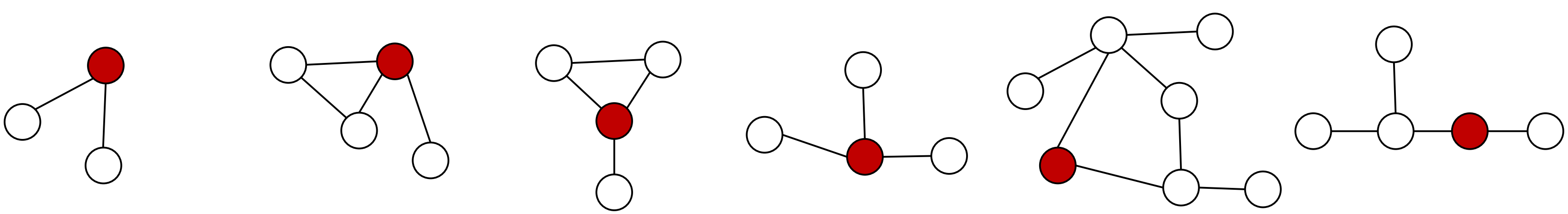
Strengths of CNNs

- Implicit feature hierarchy based on **local features**
- **Parameter sharing** across data points

Straightforward for regular graphs

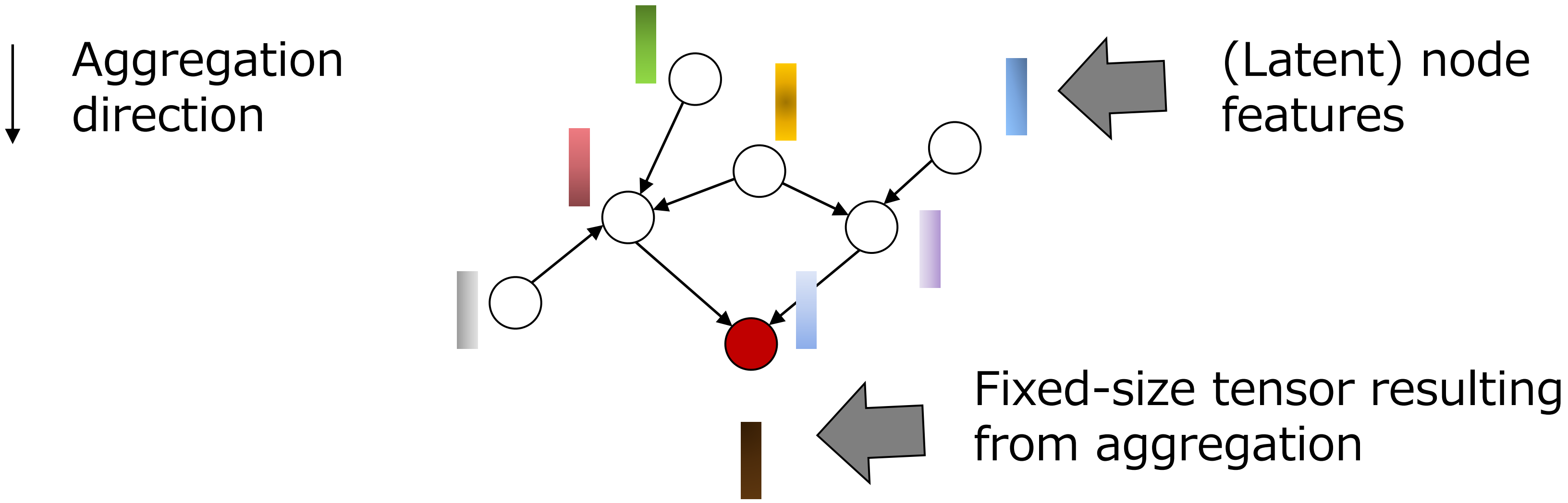


Challenging for irregular graphs



The Big Question of Graph CNNs

- How do we **aggregate neighborhood information** into **fixed-size** representations? → requirement for **weight sharing**

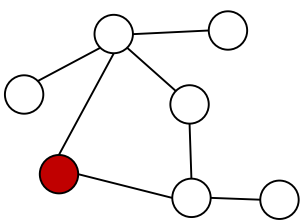


- Feature transformations are applied **locally** for each node on its neighborhood
- Requires ability to work with **highly heterogeneous** neighborhood structures

A Spectrum Of Methods

Patchy [ICML 2016]
Neighborhood
Normalization

High variance
Low bias

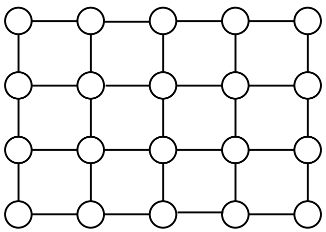


Learning CNNs for Graphs

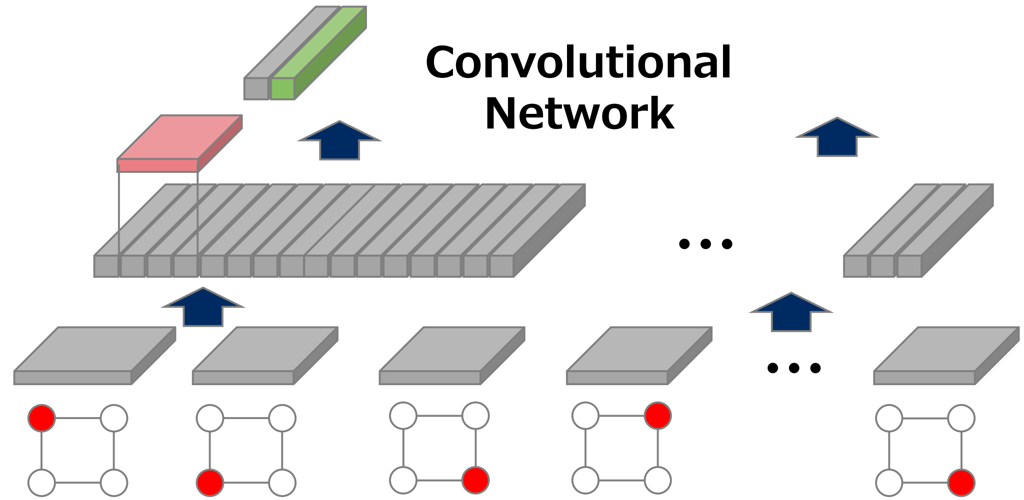
Niepert et al, 2016

Image CNN

- Grid graph required (spatial order)
- Works <u>only</u> for images

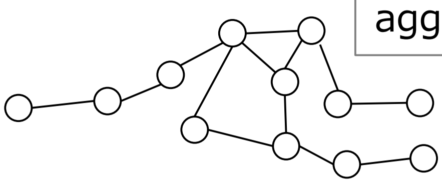


Standard CNN *moves* over image

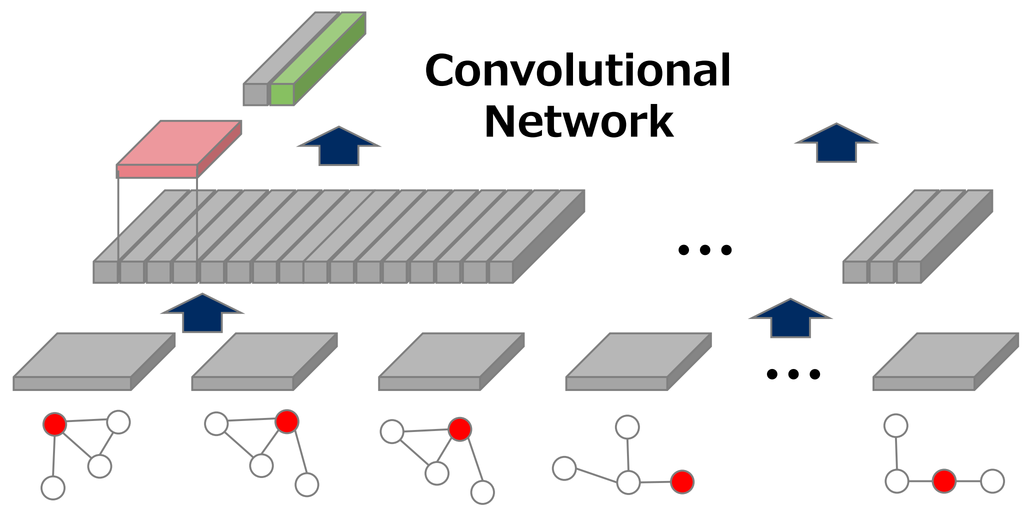


Graph CNN

- Arbitrary input graph
- Node attributes
- Edge attributes

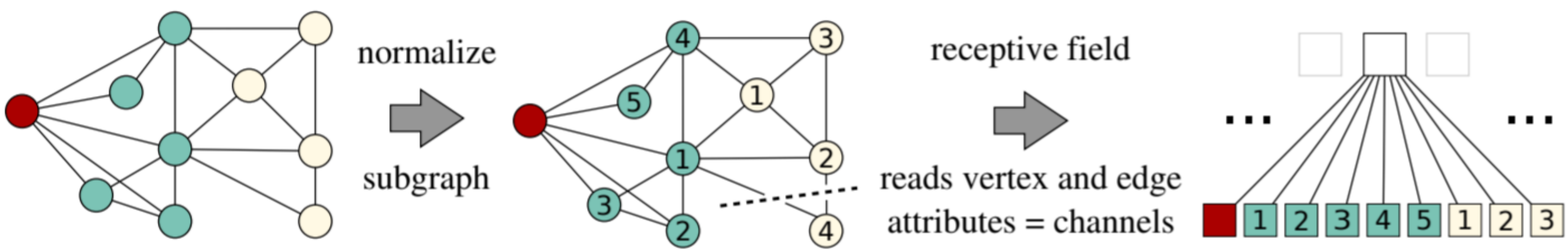


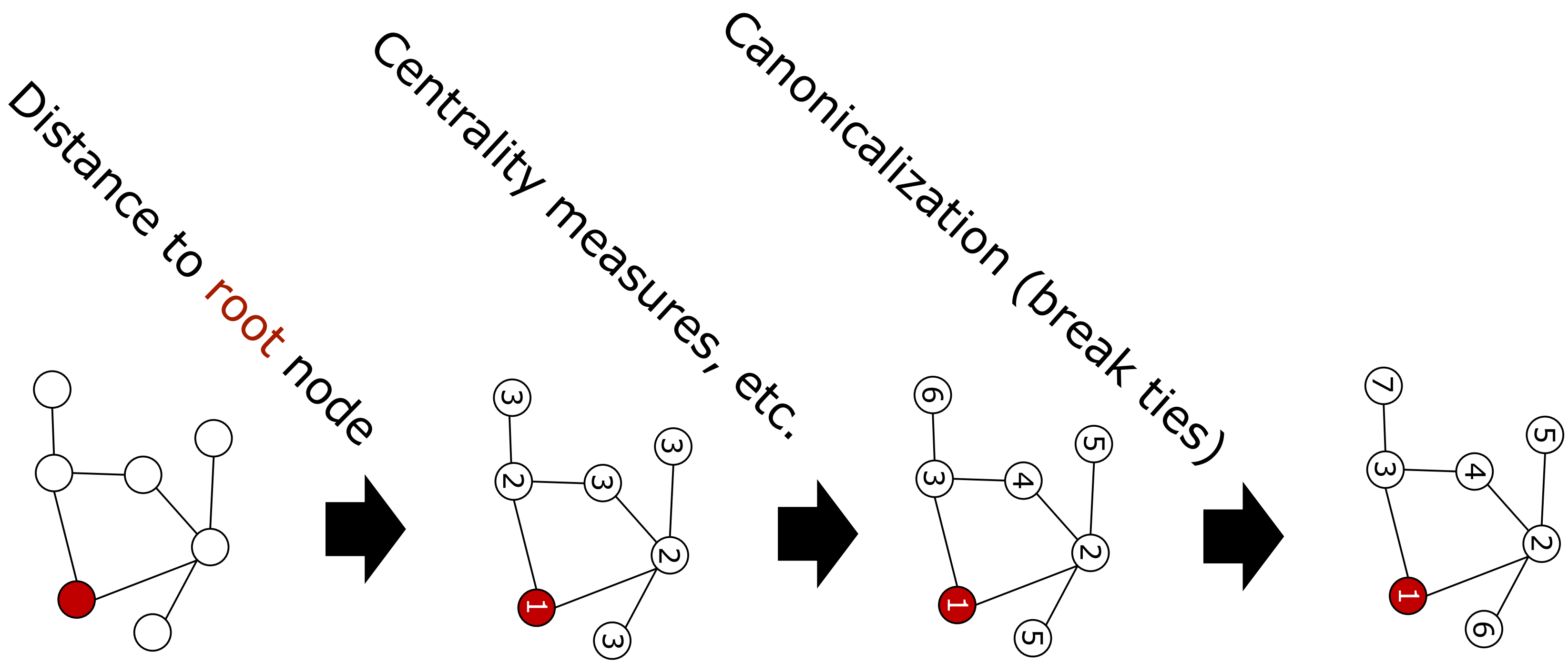
What are good local aggregations?



Neighborhood Normalization

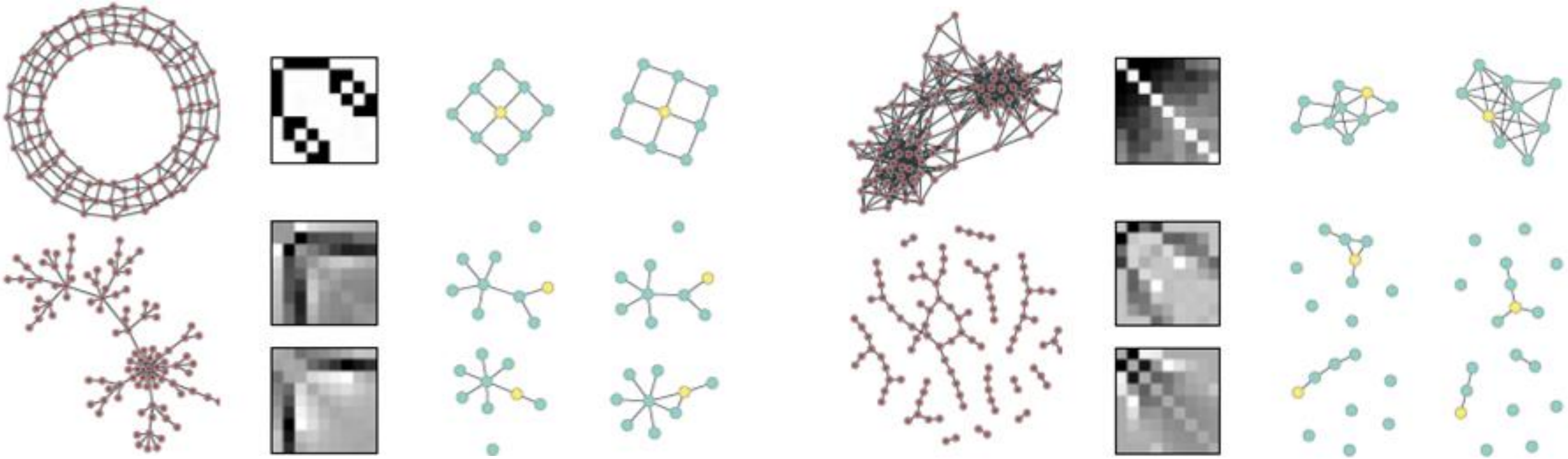
Niepert et al, 2016





Feature Visualization

Niepert et al, 2016



motifs *learned by the* model

small instances of input graphs

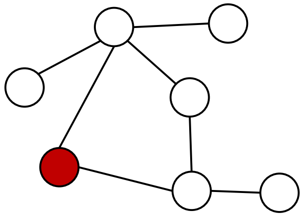
A Spectrum Of Methods

Patchy [ICML 2016]
Neighborhood
Normalization

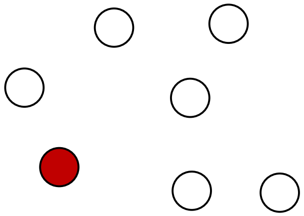
GCN [ICLR 2017]
Average Pooling

High variance
Low bias

Low variance
High bias



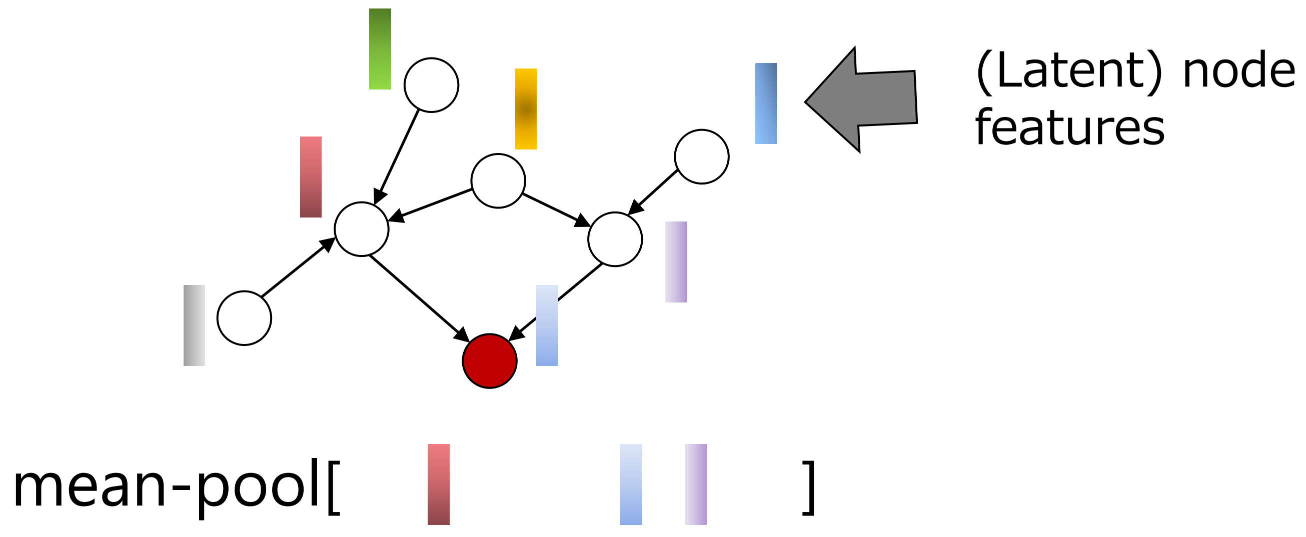
Leverage adjacency structure



Treat neighboring nodes as *exchangeable*

Graph Convolutional Networks

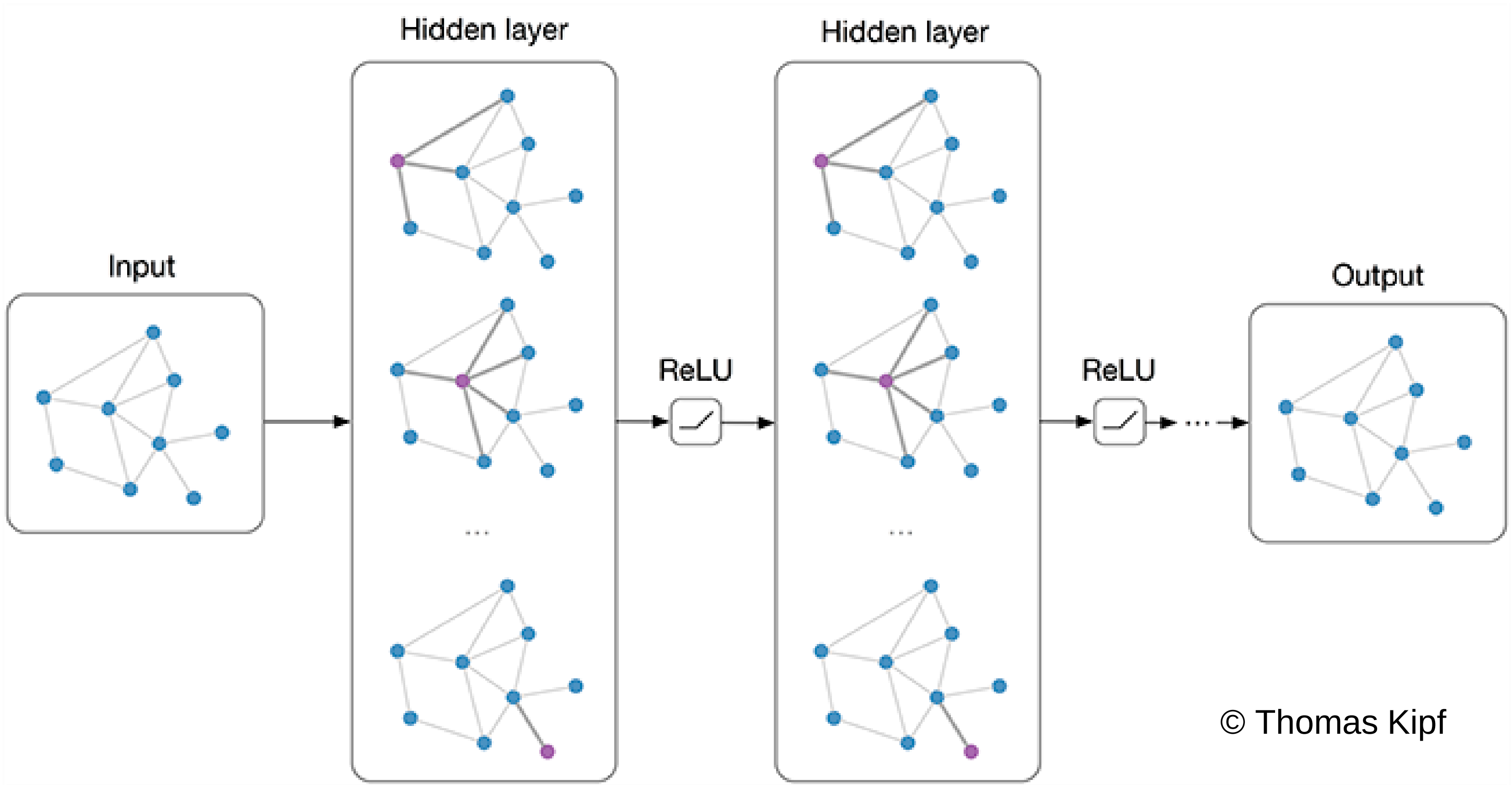
Kipf and Welling, 2017



- Compute a **weighted sum** of the node features where weights are determined by **global node adjacency** information
- Essentially **average pooling** of the (latent) node features
- Similar to message passing algorithm, aggregation and parameter updates performed **in each iteration**

Graph Convolutional Networks

Kipf and Welling, 2017



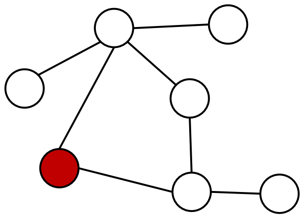
A Spectrum Of Methods

Patchy [ICML 2016]
Neighborhood
Normalization

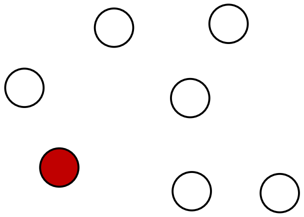
GCN [ICLR 2017]
Average Pooling

High variance
Low bias

Low variance
High bias



?



Leverage complete
adjacency structure

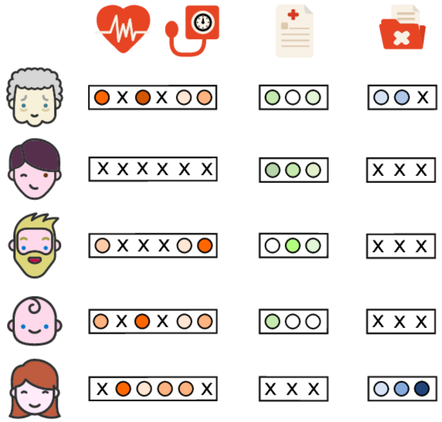
Approximate lifted learning
= clustering of structurally
similar entities in the graphs

Treat neighboring nodes as
exchangeable

Embedding propagation (EP)

Input data

Initial (incomplete) data



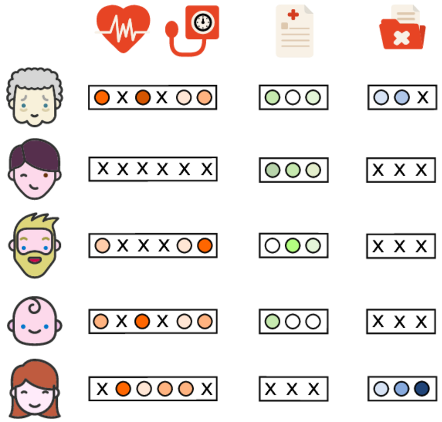
```
df_patients = ep_utils.get_small_patient_df()
```

[Learning Graph Representations with Embedding Propagation. Alberto García-Durán and Mathias Niepert. NIPS 2017.]

Embedding propagation (EP)

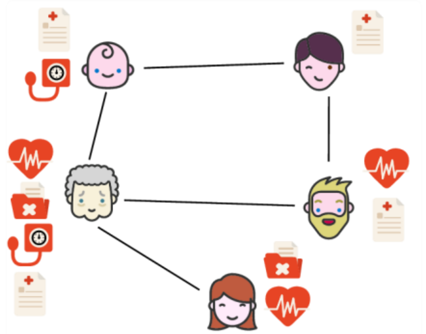
Input data

Initial (incomplete) data



Induce graph

Knowledge graph



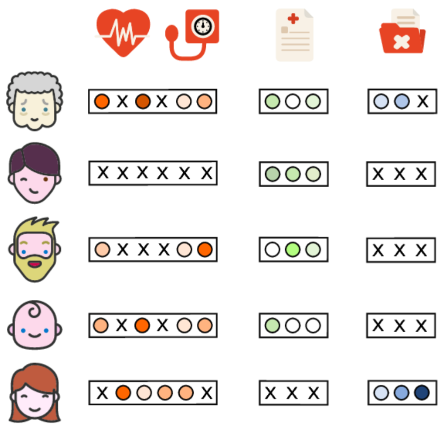
```
demographic_cols = ['gender', 'race']
graph = GraphCreator.get_graph(
    df_patients,
    demographic_cols,
    identity_index='index'
)
```

[Learning Graph Representations with Embedding Propagation. Alberto García-Durán and Mathias Niepert. NIPS 2017.]

Embedding propagation (EP)

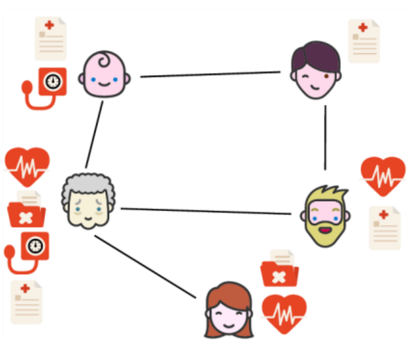
Input data

Initial (incomplete) data

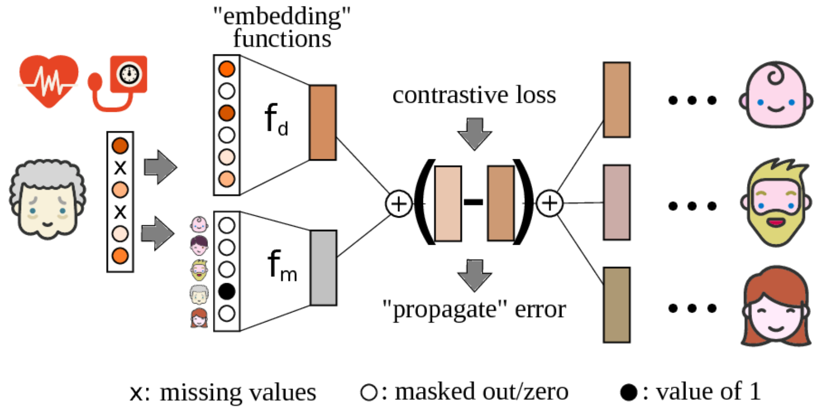


Induce graph

Patient graph



Embedding propagation (EP)



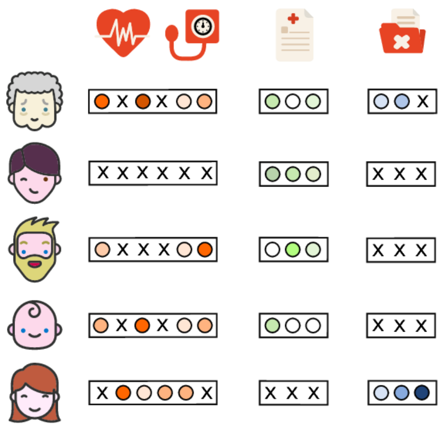

```
target_column = [
    'has_cm' # this is the target variable
]

ep = EP.get_ep(
    df_patients,
    graph,
    ignore_cols=target_column
)

ep_fit = ep.fit()
```

[Learning Graph Representations with Embedding Propagation. Alberto García-Durán and Mathias Niepert. NIPS 2017.]

GraphAI: Embedding propagation (EP)

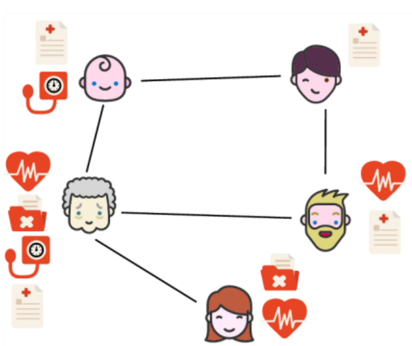
Input data

Initial (incomplete) data

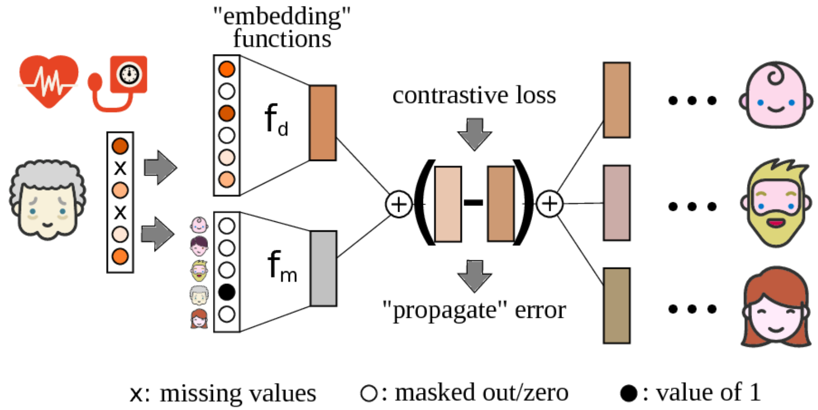


Induce graph

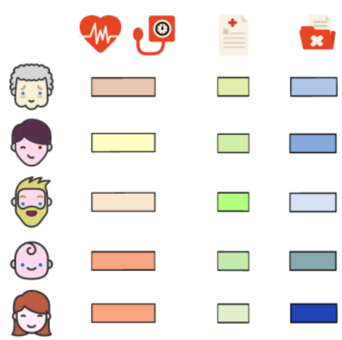
Patient graph



Embedding propagation (EP):



Complete data

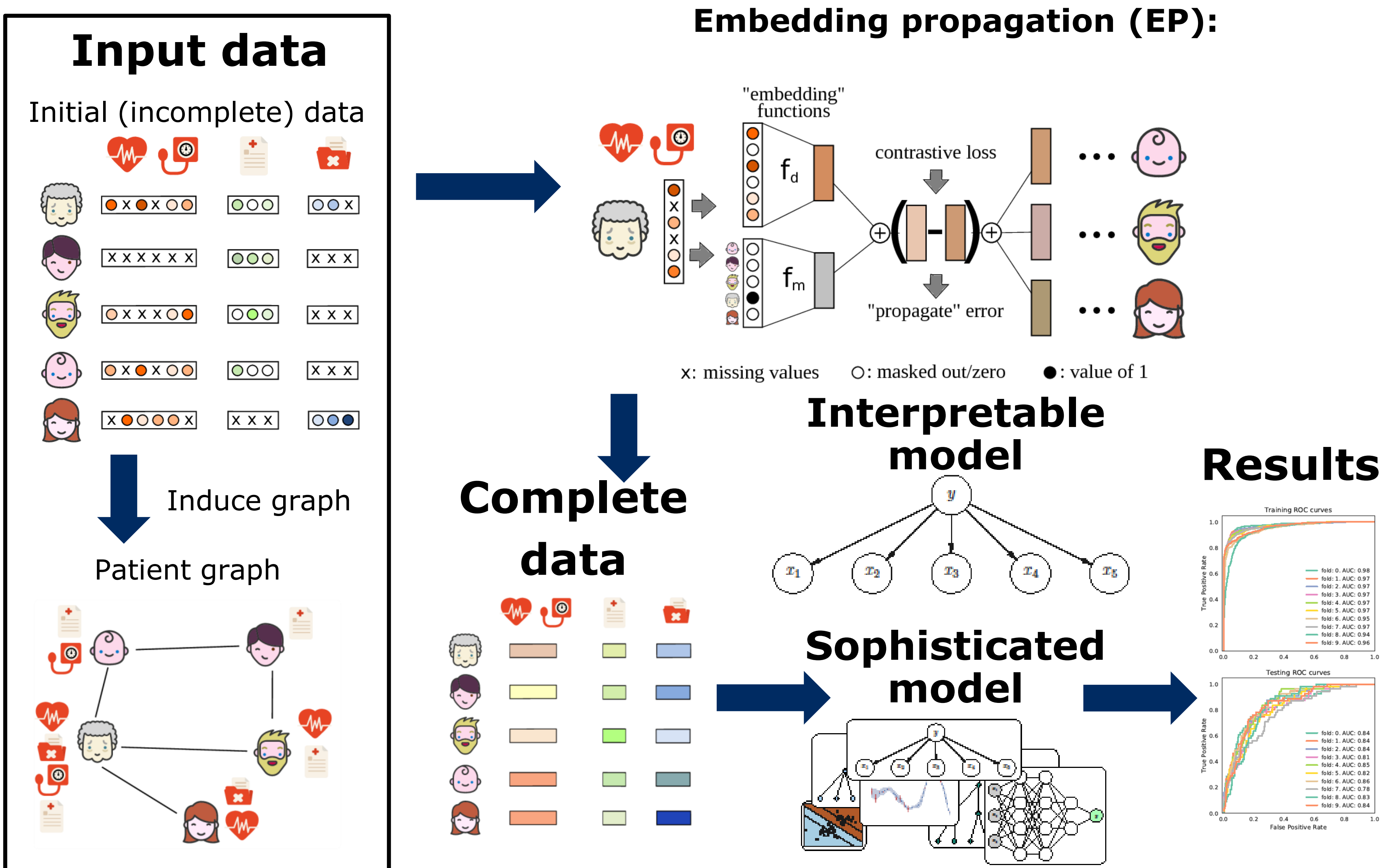


```
complete_data = ep_fit.transform(df_patients['index'])
```

[Learning Graph Representations with Embedding Propagation. Alberto García-Durán and Mathias Niepert. NIPS 2017.]

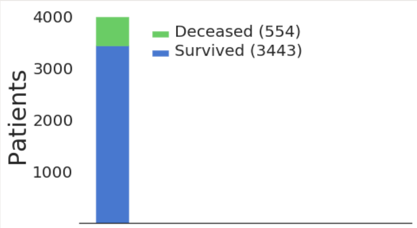
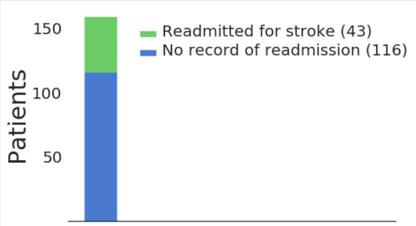
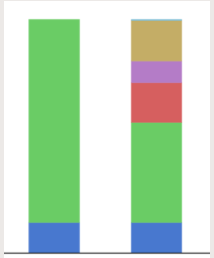
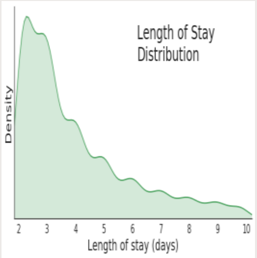
NEC

Embedding propagation (EP) workflow

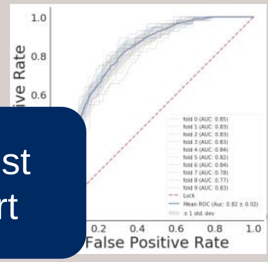
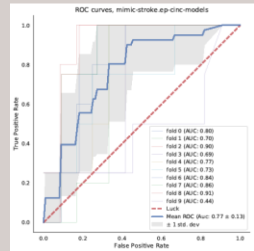
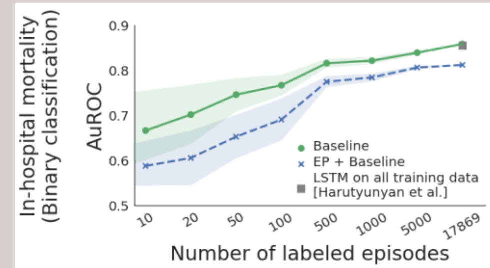
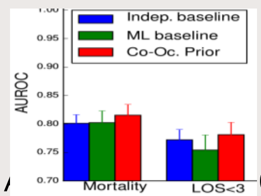
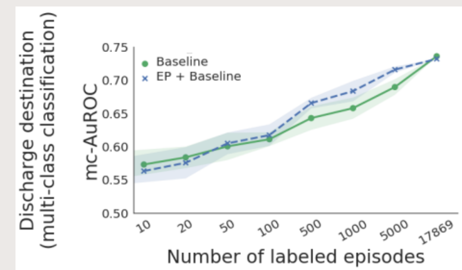
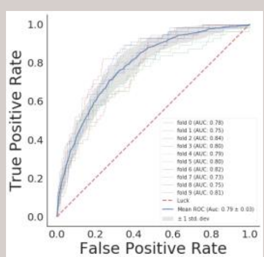
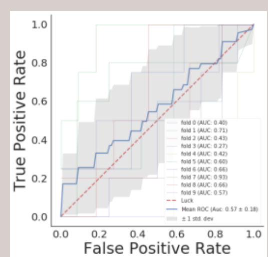
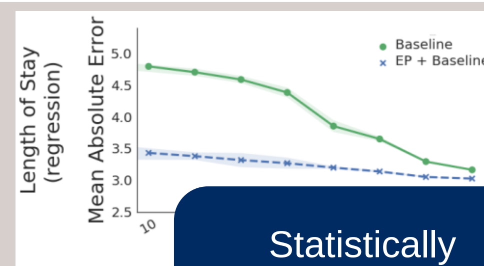


[Learning Graph Representations with Embedding Propagation. Alberto García-Durán and Mathias Niepert. NIPS 2017.]

EP use case: patient outcome prediction (datasets)

Name		**Cardiac** (Computation in Cardiology Challenge, 2012)	**Stroke** (Custom MIMIC benchmark)	**General** (MIMIC benchmarks from [Harutyunyan et al., 2017])
Task(s)		• **In-hospital mortality** (binary classification)	• **Long-term (10 year) stroke readmission** (binary classification)	• **In-hospital mortality** (binary classification) • **Length of stay** (regression) • **Discharge destination** (multiclass classification)
Time of prediction		2-days after ICU admission	End of ICU admission	2-days after ICU admission
Number of Patients		4 000	159	21 102
Outcomes				 
Modalities	Time series	37	65	17
	Demographics	0	0	5
	Free text	0	0	6
	Other	2 (SOFA and SAPS-I)	1 (primary ICD diagnosis)	3 (admission type, location, and diagnosis)
Graph construction		Sequential Organ Failure Assessment and Severity of Disease	International Classification of Diseases (ICD) – Medical Diagnosis Codes	Similarity of admission descriptions

EP use case: patient outcome prediction (results)

	Cardiac (missing data ~35%)	**Stroke** (missing data ~80%)	**General** (missing data ~35%, multimodal)
EP + Logistic regression	 2	 13	 Multimodal baseline outperforms existing state of the art.
State of the art	 03 [Uni. South California, KDD 2015]		 EP outperforms the (strong) baseline for most sample sizes.
Logistic regression	 AuROC: 0.79 ± 0.03	 AuROC: 0.57 ± 0.18	 EP is much ... acros...

Better than best state of the art

EP works best when missing data is large (+20 vs. +3), Many modalities available

Statistically significantly better for regression

Detailed Results