Machine Learning with Graphs

K2I AI and ML Boot Camp, 2025

Arlei Silva

Rice University, Houston, TX

Welcome!

Instructor

Arlei Silva, Assistant Prof. of Computer Science

Research interests: graphs, data science, machine learning

Education:

1. Universidade Federal de Minas Gerais, Brazil (B.Sc., M.Sc.)
2. University of California, Santa Barbara (Ph.D.)

Research group

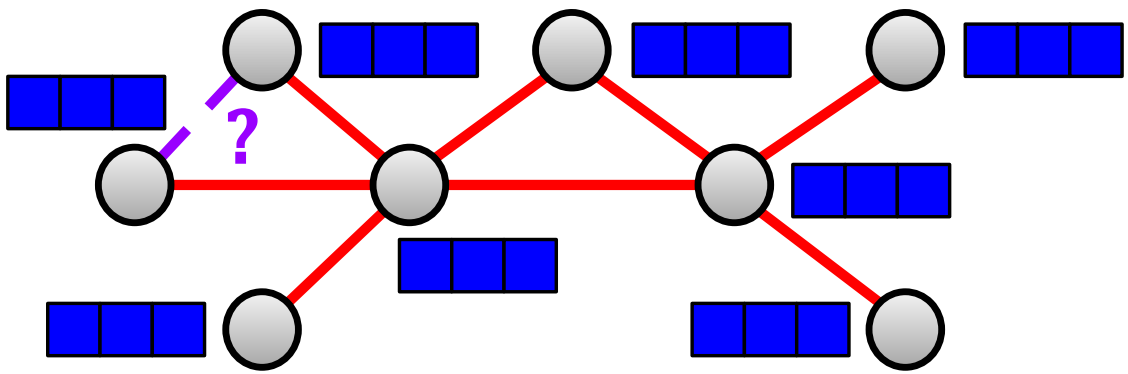
Networks as a framework for complex systems

Society, infrastructure, biological systems, content

Convergence between 3 areas:

1. Graph algorithms: combinatorial optimization
2. Network science: study of properties of real networks
3. Graph ML: representation learning

How can graph algorithms, netsci, and ML support prediction and optimization of networked systems?



The link prediction problem

Goals for this lecture

Learning goals:

1. Understand what are graphs and why graphML matters
2. Familiarize with real examples of graphML
3. Recognize graphML applications
4. Have an overview of the graphML toolkit

Topics:

- Intro to graphs
- Network science
- GraphML problems
- Graph matrices and spectral methods
- Label propagation
- Graphical models
- Graph embedding
- Graph neural networks
- Knowledge graphs

Introduction

Definition

An undirected graph (or network) is a tuple G(V,E)

$V = \{v_1, \ldots v_n\}$ is the set of vertices/nodes

$E = \{e_1, \ldots e_n\}$ is the set of edges/links, $e = \{v_i, v_j\}$

The set of neighbors of v is $N(v) = \{u | \{u, v\} \in E\}$

The degree of v is $d(v) = |N(v)|$

A complete graph has $|E| = \binom{n}{2}$

$d(v) = n - 1, \forall v \in V$

A graph is connected iff $|E| \geq n - 1$

Graph types

A simple graph is a graph that is undirected, unweighted and does not contain loops or multiple edges

Some generalizations:

Weighted: $W_e \in \mathbb{R}$ is the weight of e

Directed: $e = (v_i, v_j)$ is an ordered tuple

Signed: $sign(e) \in \{+, -\}$

Attributed: $F_V : V \to \mathbb{R}^K$, $F_E : E \to \mathbb{R}^{K'}$

Multigraph: E is a multiset

Hypergraph: hyper-edge e is non-empty subset of V

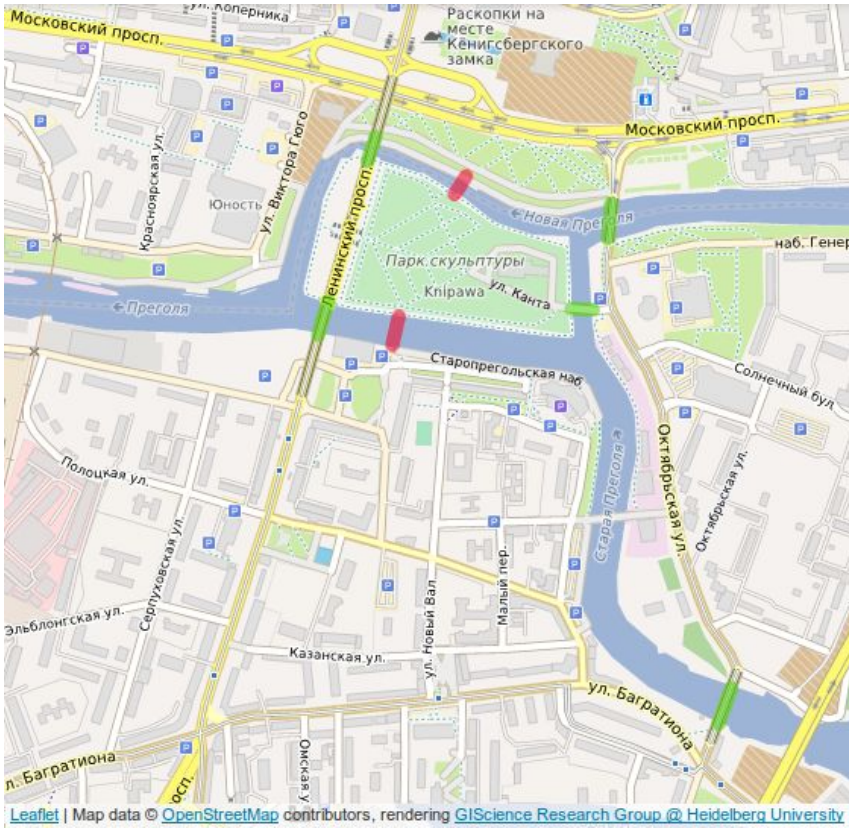
Heterogeneous: types $\tau_V : V \to K$, $\tau_E : E \to K'$

Dynamic: $\langle G_1, G_2, \ldots G_T \rangle$

The bridges of Konigsberg (Kaliningrad, Russia)

Problem given to Leonhard Euler in 1736

Is there a route that visits each of its 7 bridges exactly once?

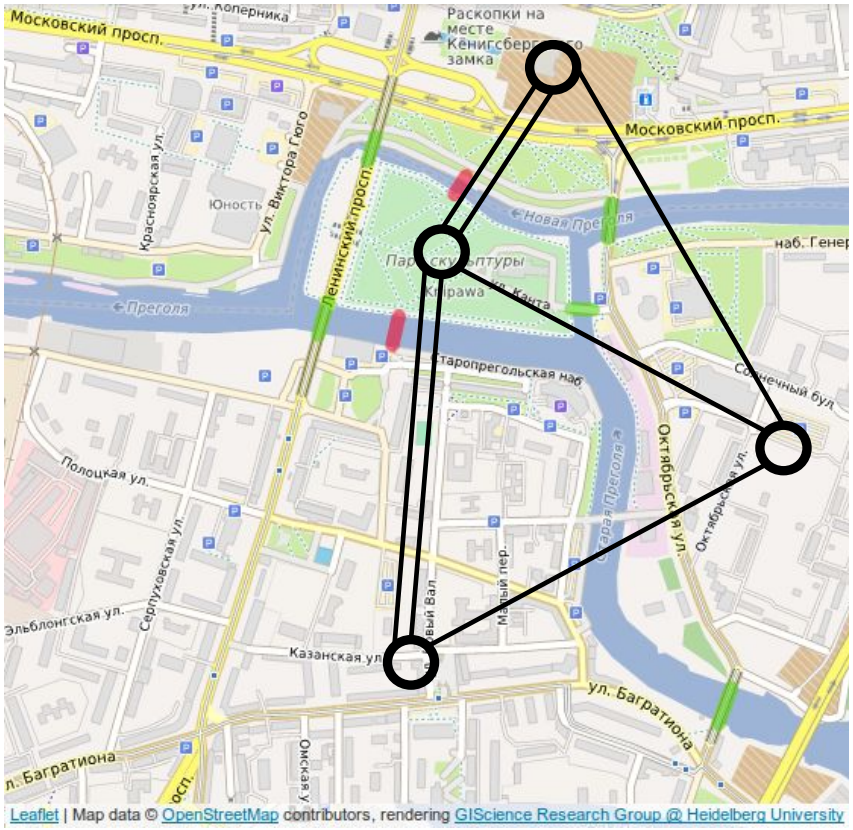


Bridges of Konigsberg (the red ones were destroyed in WWII). Source: `https://en.wikipedia.org/wiki/Seven_Bridges_of_K%C3%B6nigsberg`

The bridges of Konigsberg (Kaliningrad, Russia)

Problem given to Leonhard Euler in 1736

Is there a route that visits each of its 7 bridges exactly once?

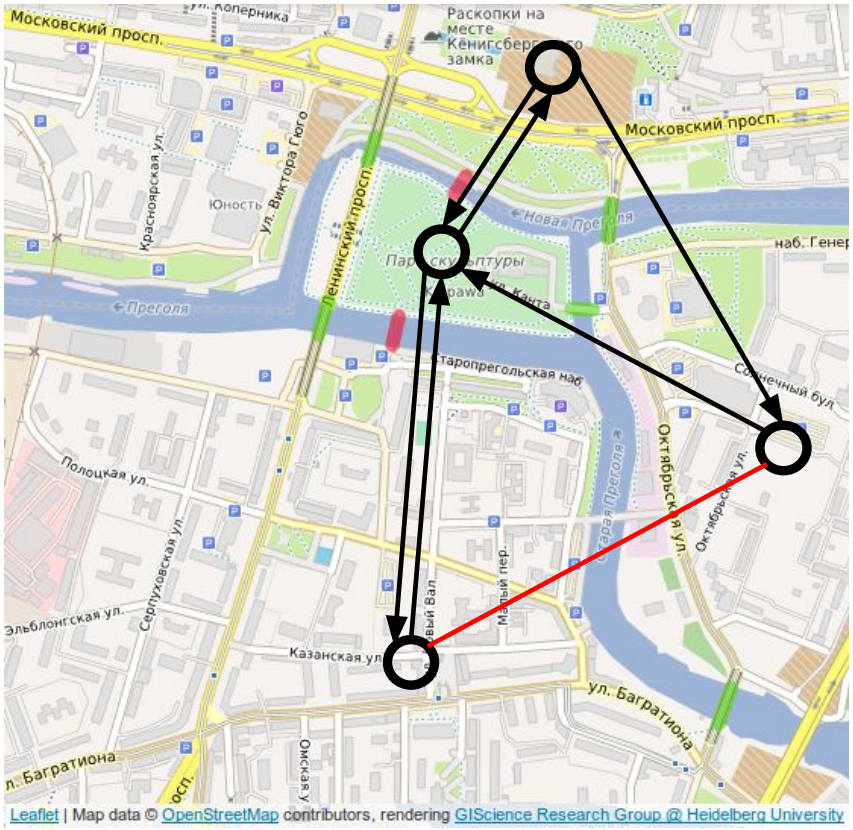


Bridges of Konigsberg (the red ones were destroyed in WWII). Source: `https://en.wikipedia.org/wiki/Seven_Bridges_of_K%C3%B6nigsberg`

The bridges of Konigsberg (Kaliningrad, Russia)

Problem given to Leonhard Euler in 1736

Is there a route that visits each of its 7 bridges exactly once?



Bridges of Konigsberg (the red ones were destroyed in WWII). Source: `https://en.wikipedia.org/wiki/Seven_Bridges_of_K%C3%B6nigsberg`

The Milgram experiment

Small-world experiment. Small-world experiment. Source: `https://en.wikipedia.org/wiki/Small-world_experiment`

Network science

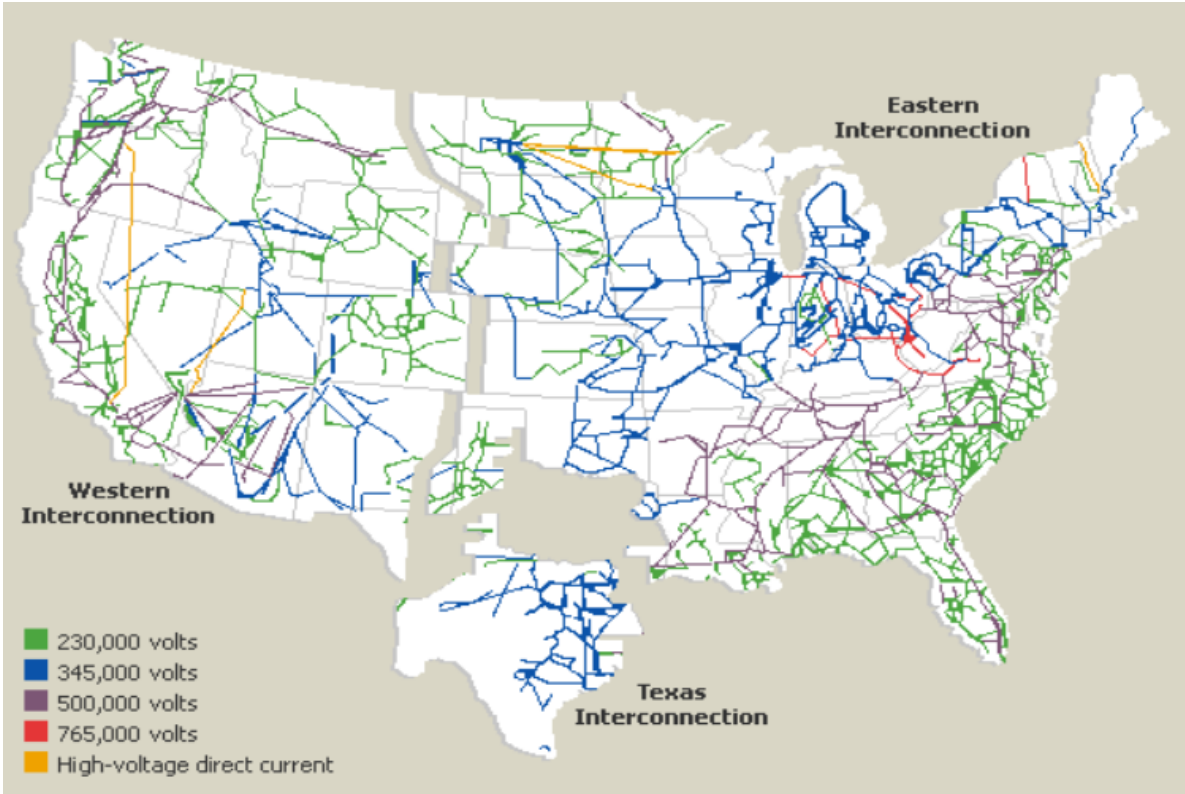
Technological networks

Network represents a man-made system

Examples:

- Internet
- Transportation network (road, airport, etc.)
- Power network
- Water distribution network

Power distribution



US power grid. Source: `https://www.ourworldofenergy.com/vignettes.php?type=electrical-power-generation&id=5`

Social networks

People as vertices

Social interactions as edges

Examples:

- Friendship network
- Collaboration network
- Opinion network
- Contact network

Facebook

Facebook network. Source: `http://www.facebook.com/notes/facebook-engineering/visualizing-friendships/469716398919`

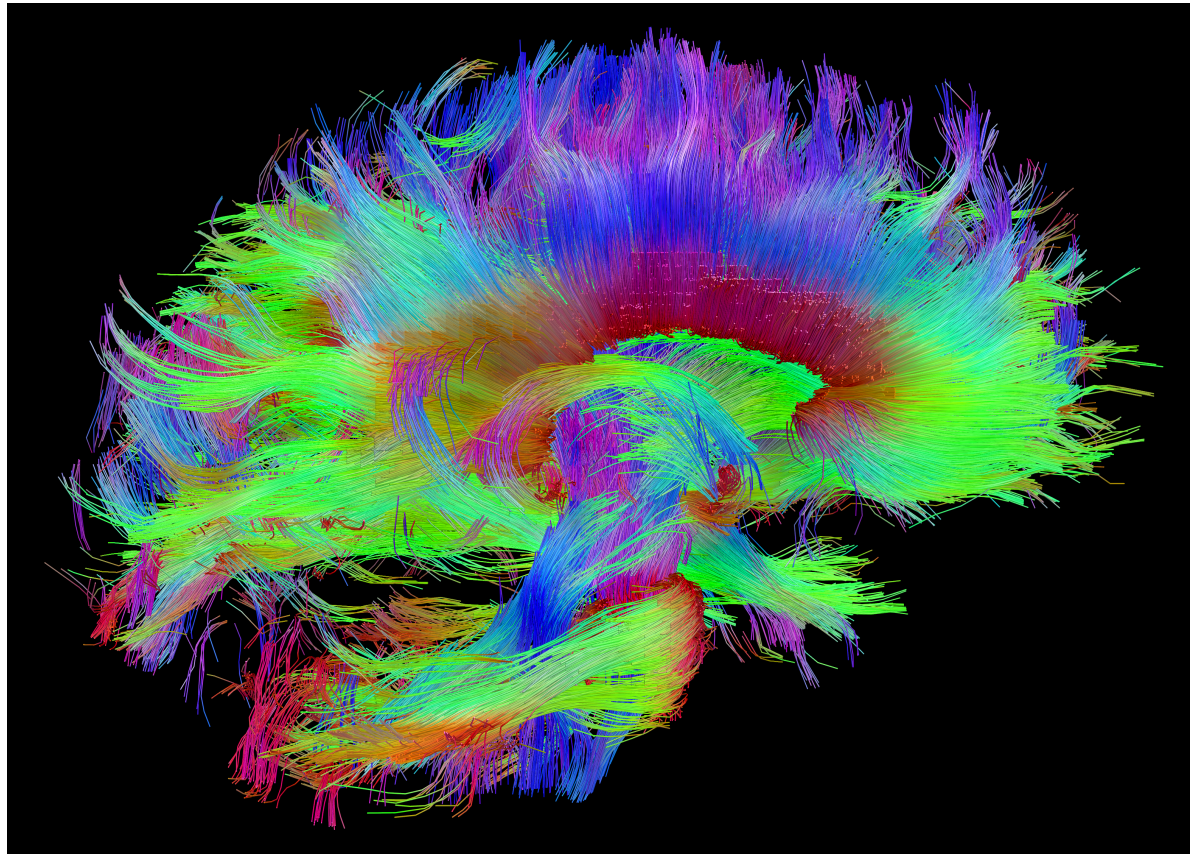
Biological networks

Represent biological systems or processes

Examples:

- Brain network
- Food web
- PPI network

The connectome



Human connectome (85B neurons). Source:
`https://directorsblog.nih.gov/2015/10/06/making-the-connections-study-links-brains-wiring-to-human-trait`

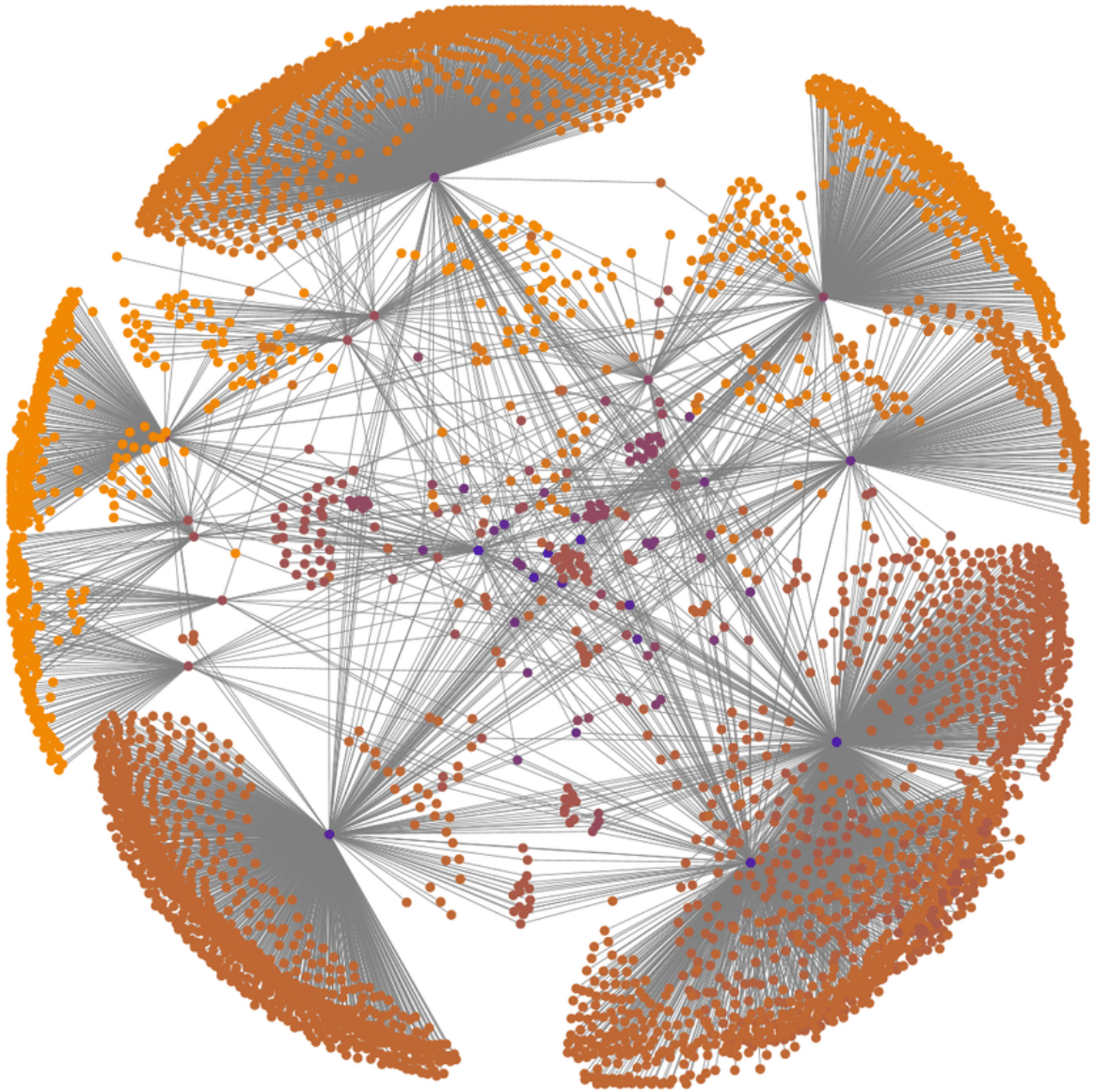
Information networks

Represent content/data and different types of interactions between them

Examples:

- The Web
- Citation networks
- Knowledge graphs

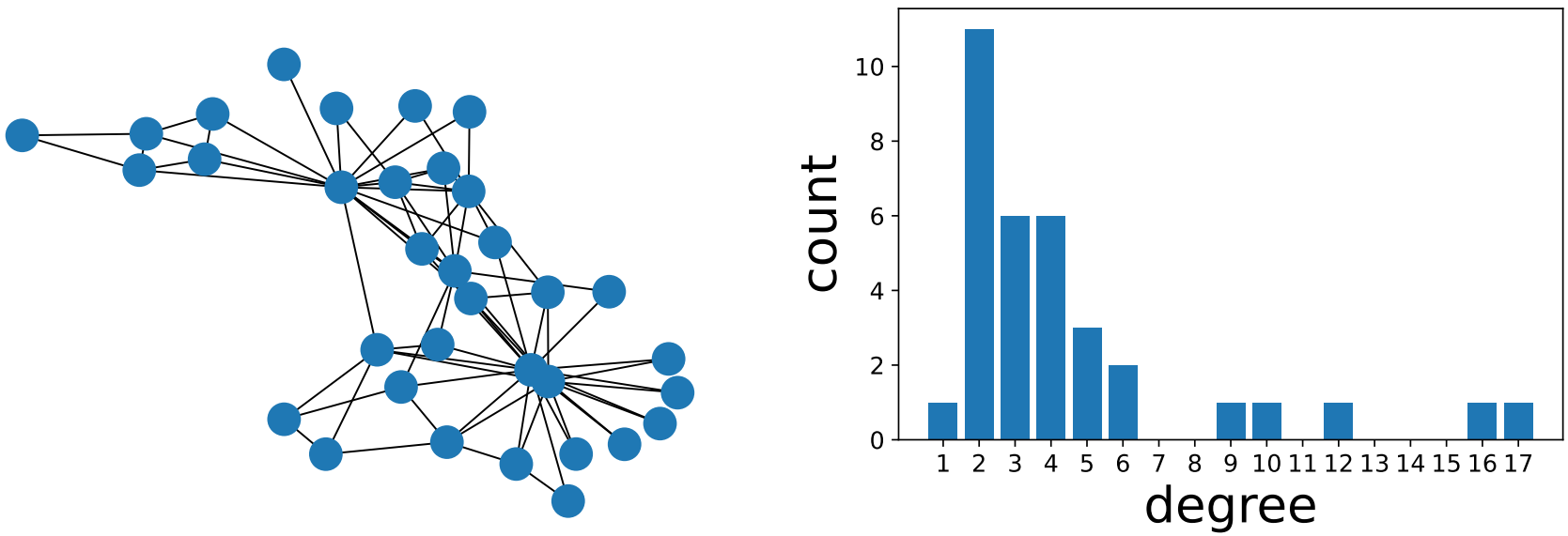
The Web



Small sample of the Web graph. Source: Dunne and Shneiderman. *Motif simplification: improving network visualization readability with fan,*

Network analysis: karate club network

34 nodes, 78 links



(a) Drawing (b) Degree distribution

Karate club network.

avg shortest path: 2.4, diameter: 5

Clustering coefficient: 0.26

GraphML Problems

GraphML problems

Abstractions that represent a class of real-world problems

E.g. classification (documents, proteins, users, images, etc.)

Help defining evaluation metrics and benchmarks

Four problems:

1. Node classification
2. Graph classification
3. Link prediction
4. Community detection

Node classification

A.k.a. semi-supervised learning on graphs

Predicting node labels based on training (labeled) nodes

Input graph $G = (V, E, F^V, F^E)$**:**

- Nodes V and edges E
- Node $F_V \in \mathbb{R}^D$ and edge features $F_E \in \mathbb{R}^{D'}$ (optional)

Label $C[v] \in \{1, 2, \ldots K\}$ **assigned to each node** $v \in V$

Labels known for $V^T \subset V$

Learn $f : V \to \{1, \ldots K\}$ **to predict labels for** $V^S \subset V \setminus V^T$

Node classification: applications

Product classification

- Graph: nodes are products, edges based on co-purchasing
- Node attributes: product descriptions
- Labels: product categories (e.g. books, toys & games, etc.)

Protein function prediction

- Graph: nodes are proteins, edges are protein interactions
- Edge attributes: interaction types (e.g. co-expression)
- Labels: protein functions (e.g. enzyme, transduction, etc.)

Node classification: datasets

Name	#Nodes	#Edges	#Classes	#Node features
Cora	2,708	5,429	7	1,433
Citeseer	3,327	4,732	6	3,703
PPI	14,755	225,270	121	50
PubMed	19,717	44,338	3	500
Nell	65,755	266,144	210	5,414
Products	2.4M	62M	47	100
Papers100M	111M	1.6B	172	128

Table: Examples of node classification datasets.

Graph classification

Predicting graph labels based on training (labeled) graphs

Graph database $\mathcal{G} = \{G_1, G_2, \ldots G_T\}$ **as input**

$G_t = (V_t, E_t, F_t^V, F_t^E)$ is a single graph

Label $C[G_t]$ **associated to each graph** $G_t \in \mathcal{G}$

Goal: learn a model $f : \mathcal{G} \to \{1, \ldots K\}$

Graph classification: applications

Molecular property prediction

- Graph: molecule
- Nodes are atoms, edges are bonds
- Features: atomic numbers, bond types
- Labels: properties (e.g. inhibits HIV replication)

Taxonomic group prediction

- Graph: protein association neighborhood
- Nodes are proteins, edges are protein interactions
- Labels: taxonomic group (e.g. mammal, bacterial family)

Graph classification: datasets

Name	#Graphs	Avg #Nodes	Avg #Edges	#Classes
Mutag	188	17.93	19.79	2
Fingerprint	2,800	5.42	4.42	15
Nci1	4,110	29.87	32.30	2
DBLP	19456	10.48	19.65	2
Molpcba	437,929	26.0	28.1	2
Ppa	158,100	243.4	2,266.1	37

Table: Examples of graph classification datasets.

Link prediction

Predicting links based on partially observed graph

Graph $G = (V, E, F^V, F^E)$ **as input**

Predicted links $E' \subseteq \{(u, v) | u, v \in V \wedge (u, v) \notin E\}$

Missing links, future links

Goal: learn f **such that** $f(u, v) = 1$ **if** $(u, v) \in E'$ **and** $f(u, v) = 0$**, otherwise**

Link prediction: applications

Friend recommendation in social networks

- Graph: nodes are users, edges are friendship
- Node attributes: user profile
- Future links: people who should meet

Missing citations and hyperlinks

- Graph: citation or hyperlink network
- Node attributes: content
- Missing links: missing citations, hyperlinks

Missing protein interactions

- Graph: protein interaction network
- Missing links: unknown interactions

Link prediction: datasets

Name	#Nodes	#Edges	Time
School	242	77,602	2 days
Conference	405	70261	2 days
AS-733	6,474	13,895	1997-2000
Arxiv HEP-TH	27,770	352,807	1993-2003
Patents	3.7M	16.5M	1975-1999

Table: Examples of (dynamic) link prediction datasets.

Community detection/partitioning

Group vertices from a graph $G(V, E, F^V, F^E)$ into communities (or clusters)

Settings:

- Unsupervised: no labels
- Semi-supervised: node community assignments available

Goal: learn f such that $f(u, v) = 1$ if $cluster(u) = cluster(v)$ and $f(u, v) = 0$

Most objectives are NP-hard

Community detection/partitioning: applications

Community detection in social networks

- Graph: nodes are users, edges are friendship
- Node attributes: user profile
- Communities: groups with common interests

Protein module discovery

- Graph: nodes are proteins, edges are protein interactions
- Edge attributes: interaction types (e.g. co-expression)
- Communities: protein modules

Other applications:

- High-performance computing
- VLSI design
- Divide-and-conquer

Graph matrices and spectral methods

The Laplacian matrix

Let G is undirected and unweighted

The adjacency matrix $A \in \mathbb{R}^{n \times n}$ of $G = (V, E)$ is defined as:

$$A_{ij} = \begin{cases} 1 & \text{if } (v_i, v_j) \in E \\ 0 & \text{otherwise} \end{cases}$$

The degree matrix of G is defined as:

$$D_{ij} = \begin{cases} deg(v_i) & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

The Laplacian $L \in \mathbb{R}^{n \times n}$ of G is:

$$L = D - A$$

Spectral clustering

Given a dataset $\mathcal{D} = \{\mathbf{x}_1, \ldots \mathbf{x}_n\}$

Goal: partition $\mathcal{D}$ into k non-overlapping clusters $C_1, C_2, \ldots C_k$

From dataset $\mathcal{D}$ to graph G:

- Pairwise edges, total $n(n-1)/2$
- Edge weights $W_{ij} = exp(-\gamma||\mathbf{x}_i - \mathbf{x}_j||)$

Algorithm:

1. Compute (standard or normalized) Laplacian $L = D - W$;
2. Compute first r eigenvectors $\mathbf{u}_1, \ldots \mathbf{u}_r$ of L;
3. Create new feature matrix from eigenvectors, where $\mathbf{x}'_i = [\mathbf{u}_1[i], \ldots \mathbf{u}_r[i]]$;
4. Cluster data using new features $\mathbf{x}'_1, \ldots \mathbf{x}'_n$ (e.g. k-means).

Label propagation

Node classification

Graph $G = (V, E)$ with labeled (V_L) and unlabeled (V_U) vertices, where $V = V_L \cup V_U$

Goal: predict labels of vertices in V_U

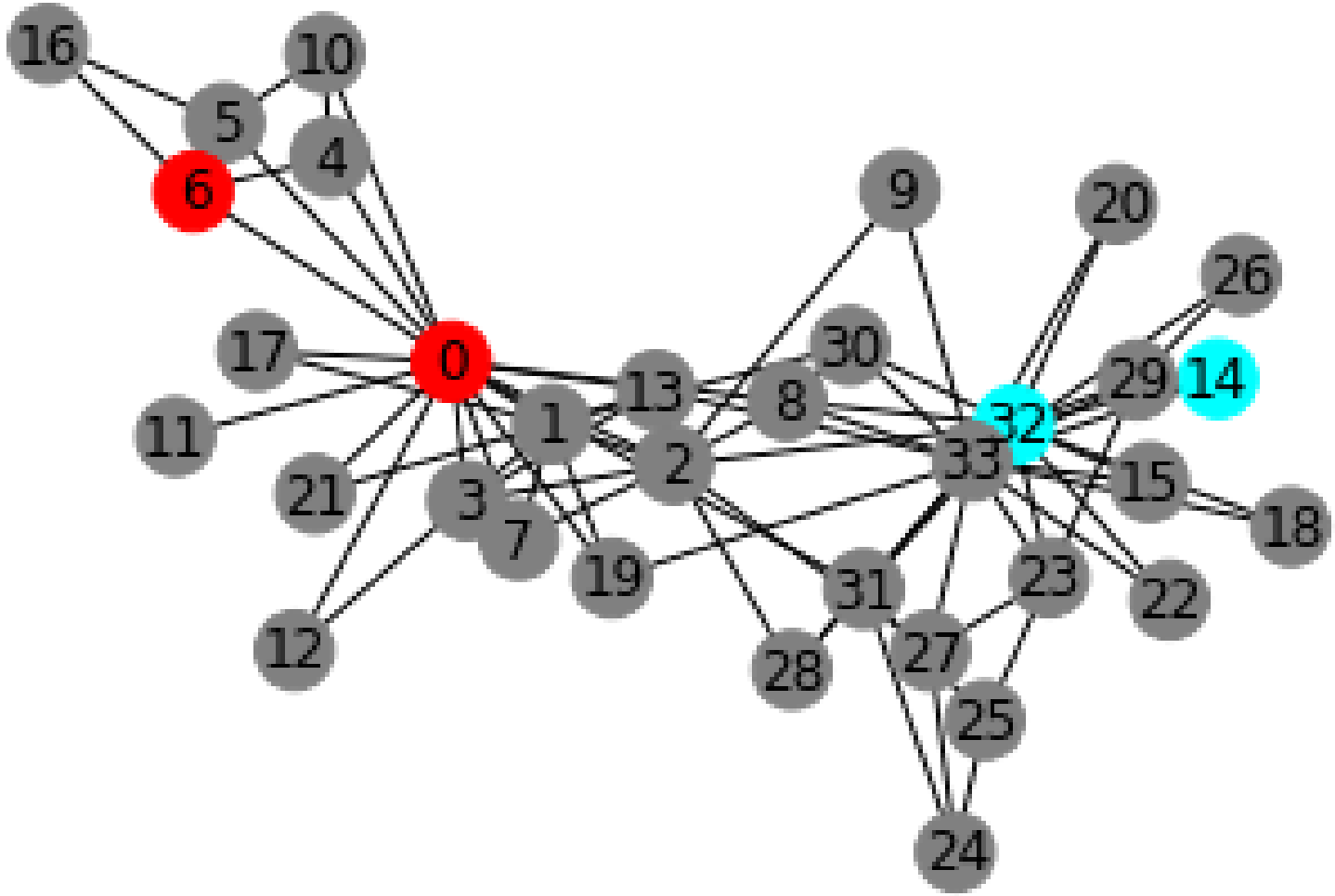
Unlabeled vertices: "stepping stones" for propagating labels

Assumption: binary labels $y_v = \mathbf{y}[v] \in 0, 1$,$\mathbf{y} \in \{0, 1\}^n$

Notation: label vector in the form $\mathbf{y} = [\mathbf{y}_L; \mathbf{y}_U]$

$\mathbf{y}_L$ and $\mathbf{y}_U$ are associated with V_L and V_U, respectively

Example


16
10
5
4
6
9
20
26
17
0
30
29
14
13
8
32
11
1
2
33
15
18
21
3
7
19
31
23
22
12
28
27
25
24

Label propagation

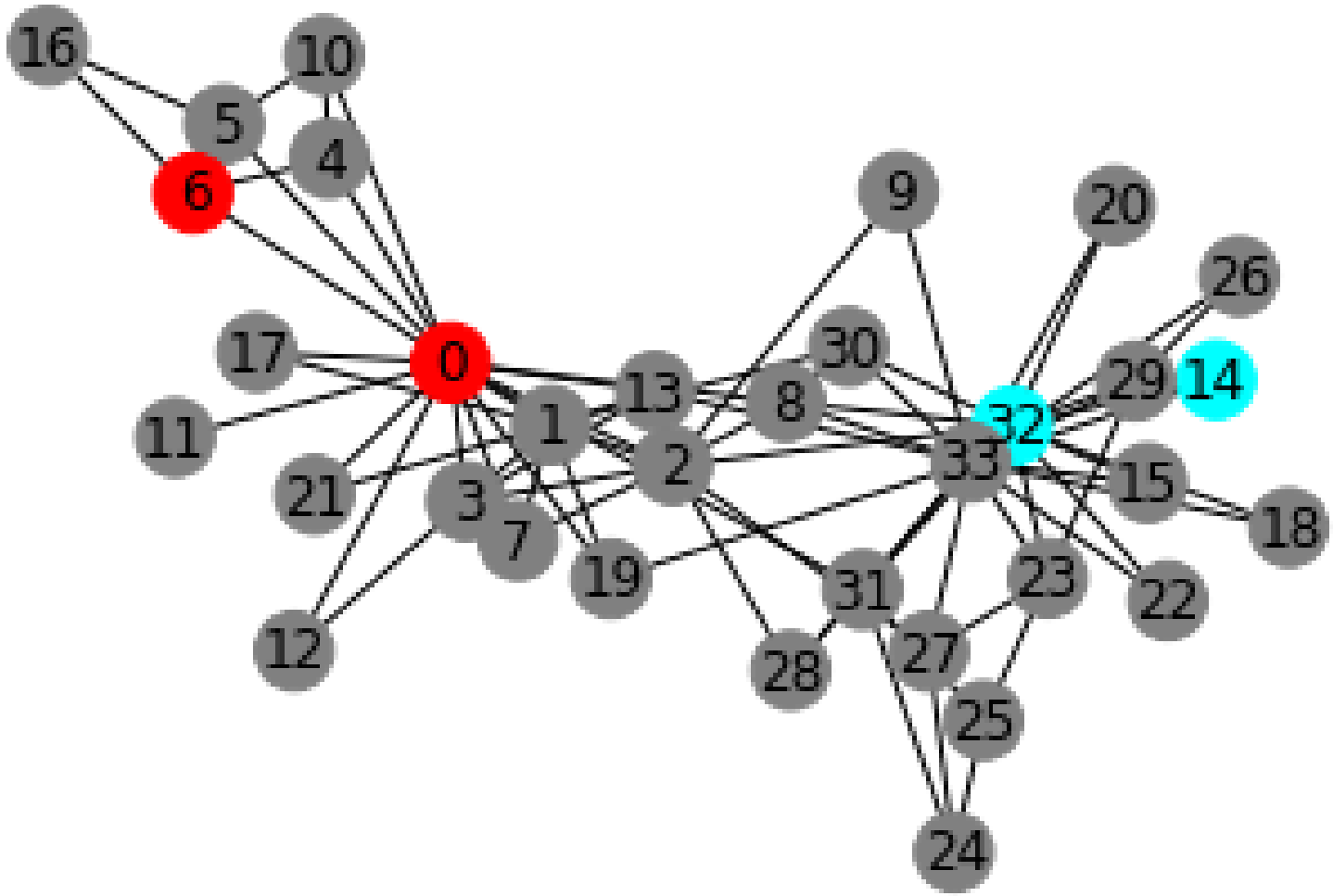
Transition matrix $P = D^{-1}A$

- $D = diag(A\mathbf{1}_n)$ is the degree matrix
- A is the adjacency matrix
- $P \in [0,1]^{n \times n}$ is such that $P_{uv} = A_{uv}/deg(u)$

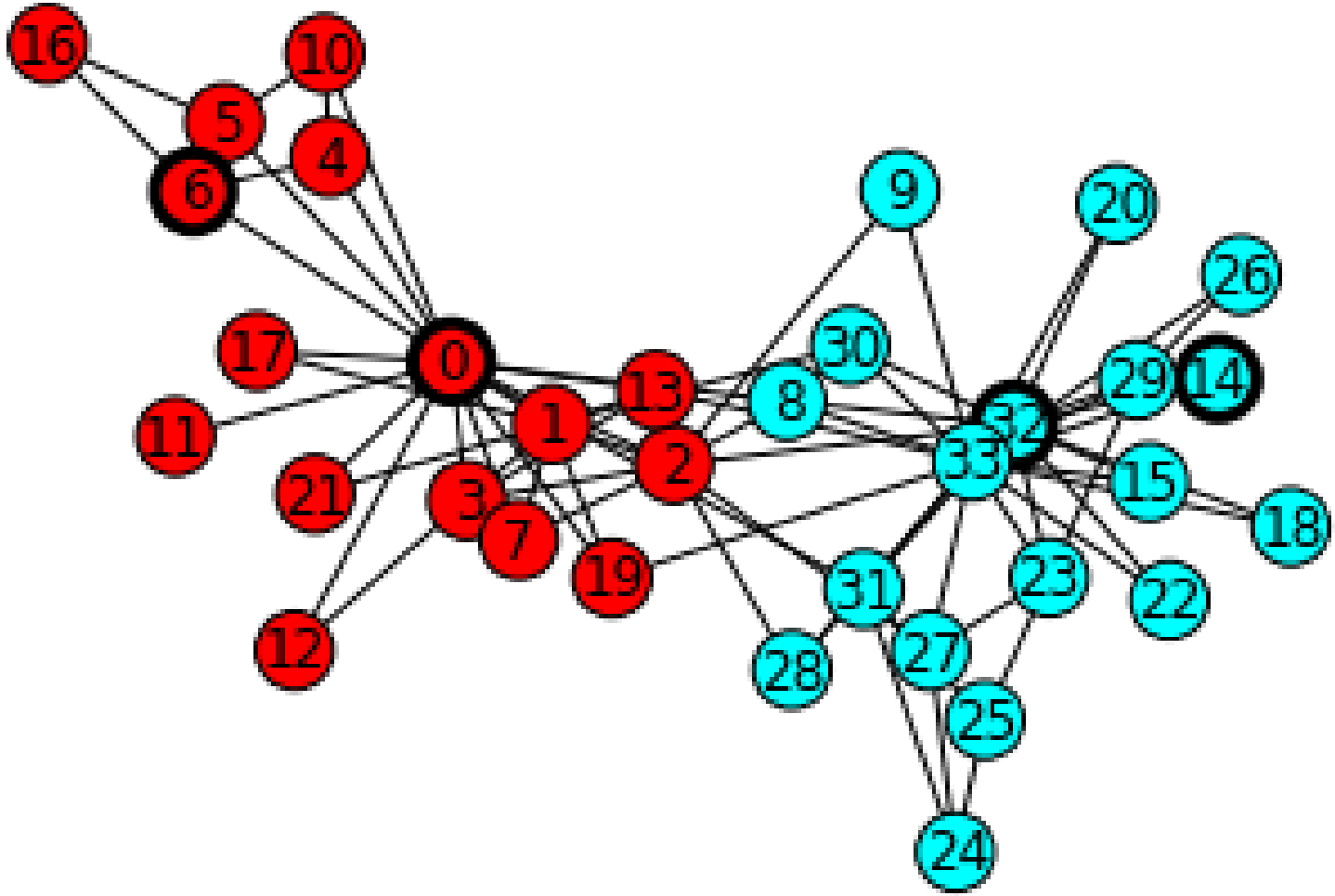
Algorithm:

1. $P = D^{-1}A$
2. $\mathbf{y}^0 = [\mathbf{y}_L; 0]$
3. for $t \in 1, \ldots T$
   - 3.1 $\mathbf{y}^{t+1} = P\mathbf{y}^t$
   - 3.2 $\mathbf{y}_L^{t+1} = \mathbf{y}_L$

Example


16
10
5
4
6
9
20
26
17
0
30
29
14
13
8
32
11
1
2
33
15
18
21
3
7
19
31
23
22
12
28
27
25
24

Example: label propagation



Graphical models

The causality ladder

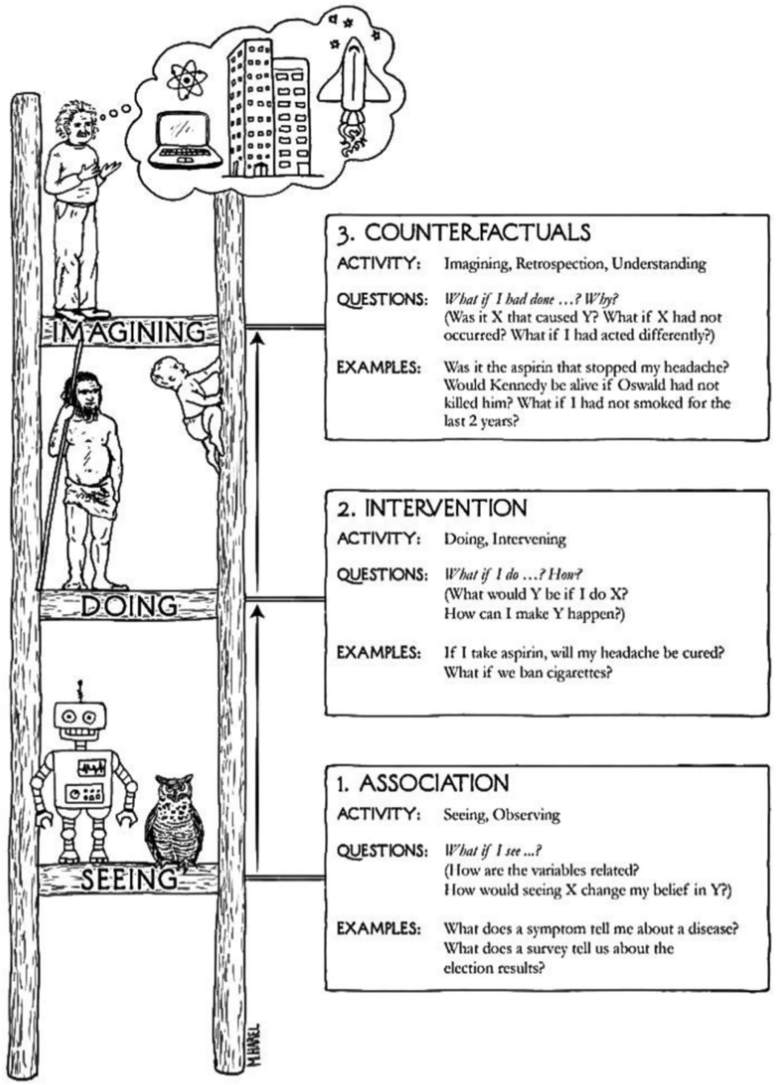


Figure: Source: "The Book of Why" by Pearl and Mackenzie

Causal inference

Examples:

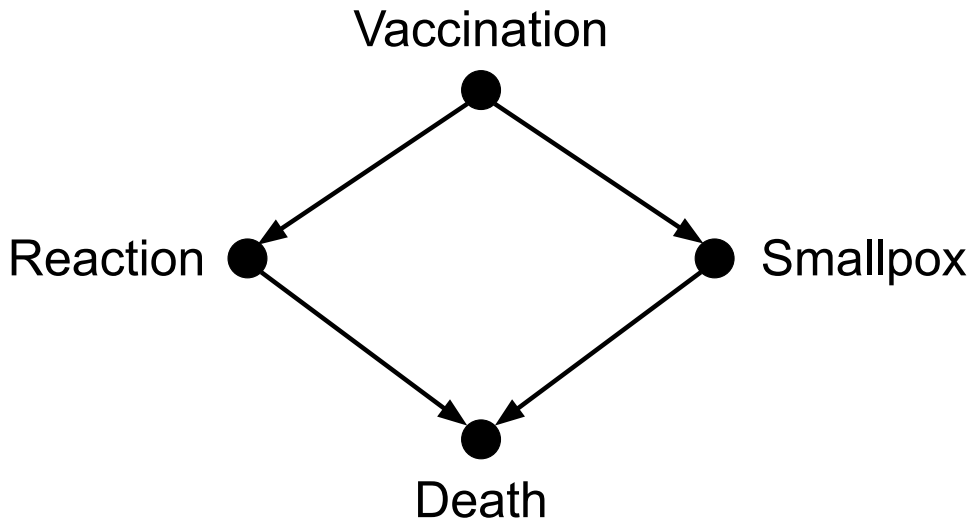
1. How effective is a given treatment in preventing a disease?
2. What cause our sales to go up? Lower taxes? Advertising?
3. I'm about to quit my job. Should I?

"Data can tell you that people who took a medicine recovered faster than those who did not take it, but they can't tell you why"

"Bayesian networks hold the key that enables causal diagrams to interface with the data"

E.g. difference between $P(X|Y)$ **and** $P(X|do(Y))$

Causal inference: example



Parameters:

- $p(Reaction|Vaccination)$
- $p(Smallpox|Vaccination)$
- $p(Death|Reaction)$
- $p(Death|Smallpox)$

Graphical models

Framework for solving learning and inference problems

Graph representation of a joint distribution:

- Vertices: random variables
- Edges: probabilistic relationships (e.g. dependencies)

Examples:

- Directed acyclic graphs (Bayesian networks)
- Undirected graphs (Markov Random Fields)

Problems:

- Learning the parameters of the model
- Inferring the probabilities given the model

Example

We are given readings from two (accurate) sensors

1. Checks whether the sky is cloudy or not
2. Checks if it is raining or not

Sensor readings as Boolean random variables:

"Cloudy" and "Rain"

Goal: model relationships between variables

To Predict one variable given another

Example

Collecting data/observations (counts):

1. Cloudy and raining: $\#(Cloudy = 1 \wedge Rain = 1)$
2. Cloudy but not raining: $\#(Cloudy = 1 \wedge Rain = 0)$
3. Raining but not cloudy: $\#(Cloudy = 0 \wedge Rain = 1)$
4. Neither raining or cloudy: $\#(Cloudy = 0 \wedge Rain = 0)$

We can use the data to estimate:

- The joint probability $p(Cloudy, Rain)$
- Conditional probabilities, e.g. $p(Rain|Cloudy)$

Cloudy\Rain	0	1
0	.8	.2
1	.2	.8

Table: Conditional distribution $p(Rain|Cloudy)$.

Example

Now consider additional variables capturing the following:

1. "Sprinkler" (on/off)
2. "Wet-Grass" (yes/no)

Examples of problems in this setting:

1. What is the prob. that the sprinkler is on and it is raining?
2. Given that it is cloudy, what is the prob. that the grass is wet?

Collecting data:

1. $\#(\mathit{Cloudy} = 1 \wedge \mathit{Rain} == 1 \wedge \mathit{Sprinkler} = 1 \wedge \mathit{Wet\text{-}Grass} = 1)$
2. $\#(\mathit{Cloudy} = 1 \wedge \mathit{Rain} == 1 \wedge \mathit{Sprinkler} = 1 \wedge \mathit{Wet\text{-}Grass} = 0)$
3. ...

16. $\#(\mathit{Cloudy} = 0 \wedge \mathit{Rain} == 0 \wedge \mathit{Sprinkler} = 0 \wedge \mathit{Wet\text{-}Grass} = 0)$

Example

For 4 variables, the joint probability table 4-dimensional

- Is this scalable?
- Are there alternatives?

Example

For 4 variables, the joint probability table 4-dimensional

- Is this scalable?
- Are there alternatives?

Assume that variables are independent, then:

$$p(\mathit{Cloudy}, \mathit{Rain}, \mathit{Sprinkler}, \mathit{Wet\text{-}Grass}) = p(\mathit{Cloudy}) \times p(\mathit{Rain}) \times p(\mathit{Sprinkler}) \times p(\mathit{Wet\text{-}Grass})$$

Joint probability based on 1-dimensional tables

Cloudy	0	1
	.5	.5

Table: Probability $p(\mathit{Cloudy})$.

Example

Independence assumption is quite convenient

for computation, storage, data collection, etc.

However, all dependencies were lost

Example

Independence assumption is quite convenient

for computation, storage, data collection, etc.

However, all dependencies were lost

Is there any midterm solution?

Graphical models

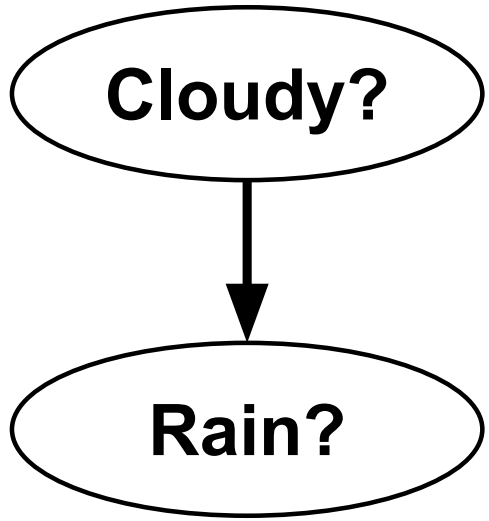


Figure: Graphical model for our first example.

Edge from Cloudy to Rain means that clouds "cause" rain

Graphical models

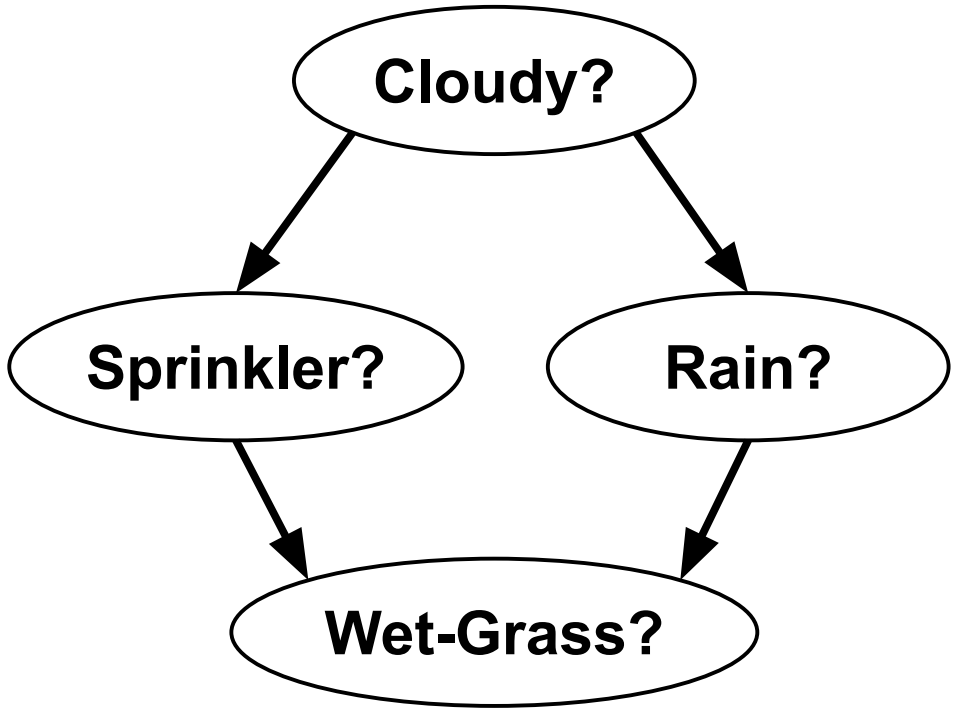


Figure: Graphical model for our second example.

We have applied our knowledge to reduce the dependencies

Graph encodes our assumptions about the real world

Graphical models

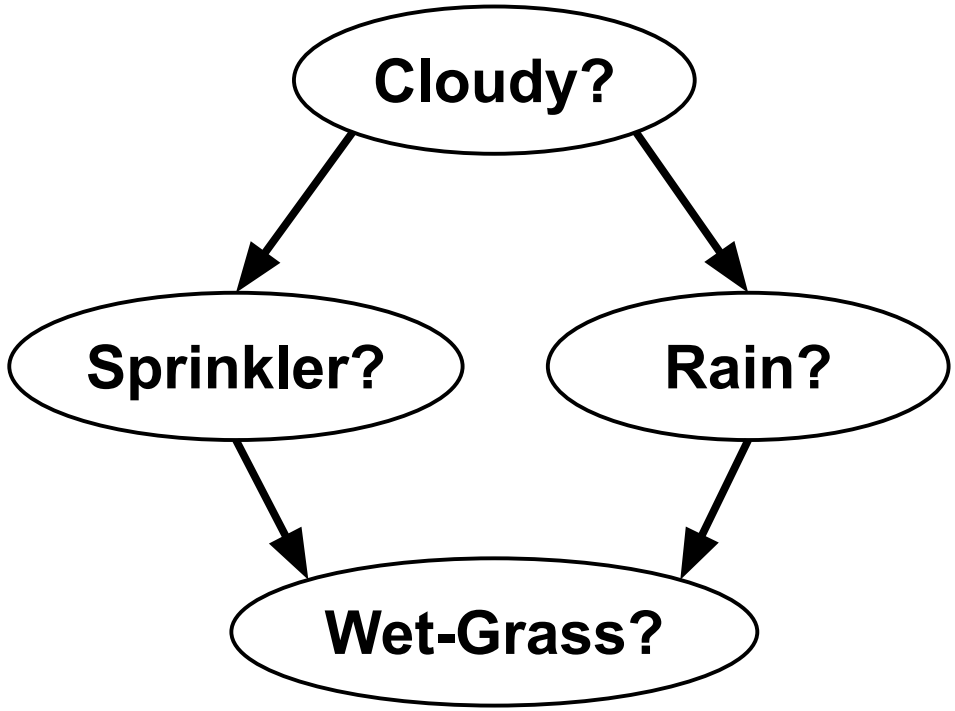


Figure: Graphical model for our second example.

Using the GM to decompose the joint distribution:

$$p(Cloudy, Rain, Sprinkler, Wet\text{-}Grass) = p(Cloudy) \times p(Rain|Cloudy) \times p(Sprinkler|Cloudy) \times p(Wet\text{-}Grass|Sprinkler, Rain)$$

Graphical models

Cloudy\Rain	0	1
0	.8	.2
1	.2	.8

(a) $p(Rain|Cloudy)$

Cloudy\Sprinkler	0	1
0	.5	.5
1	.9	.1

(b) $p(Sprinkler|Cloudy)$

Cloudy	0	1
	.5	.5

(c) $p(Cloudy)$

Rain	Sprinkler \ Wet-Grass	0	1
0	0	.1	.0
0	1	.1	.9
1	0	.1	.9
1	1	.01	.99

(d) $p(Wet\text{-}Grass|Rain, Sprinkler)$

Figure: Conditional probability tables for graphical model from Figure 4.

GM sparsity translates to factorization of the distribution

Graphical model

i is a parent of j in the GM if there exists an edge (i, j)

Joint distribution of a GM with random variables $x_1, x_2, \ldots x_n$:

$$p(x_1, \ldots x_n) = \prod_{i=1}^{n} p(x_i | pa_i)$$

where pa_i is the set of parents of the node i.

More examples: HMM

Hidden Markov Model

- Observations $\langle y_1, y_2, \ldots y_T \rangle$, $y_t \in \{1, \ldots m\}$
- Hidden states $\langle x_0, x_1, \ldots x_T \rangle$,$x_t \in \{1, \ldots n\}$
- Conditional distribution $p(y_t|x_t)$
- Hidden state transition probabilities $p(x_t|x_{t-1})$

More examples: HMM

Hidden Markov Model

- Observations $\langle y_1, y_2, \ldots y_T \rangle$, $y_t \in \{1, \ldots m\}$
- Hidden states $\langle x_0, x_1, \ldots x_T \rangle$,$x_t \in \{1, \ldots n\}$
- Conditional distribution $p(y_t|x_t)$
- Hidden state transition probabilities $p(x_t|x_{t-1})$

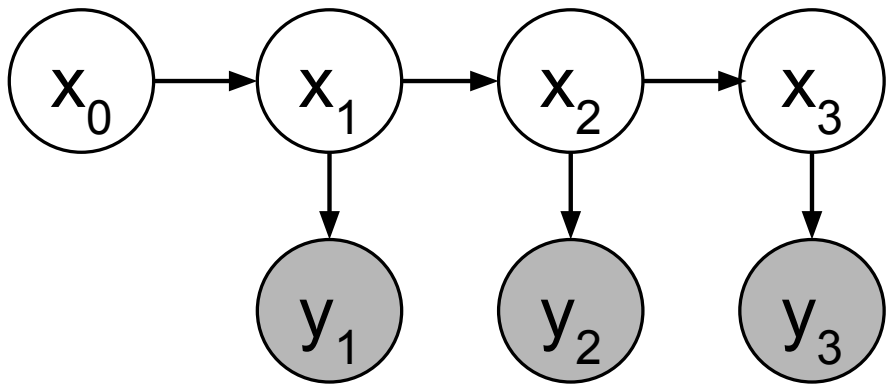


Figure: Hidden Markov Model. Grey nodes are observations.

Joint probability:

More examples: HMM

Hidden Markov Model

- Observations $\langle y_1, y_2, \ldots y_T \rangle$, $y_t \in \{1, \ldots m\}$
- Hidden states $\langle x_0, x_1, \ldots x_T \rangle$,$x_t \in \{1, \ldots n\}$
- Conditional distribution $p(y_t|x_t)$
- Hidden state transition probabilities $p(x_t|x_{t-1})$

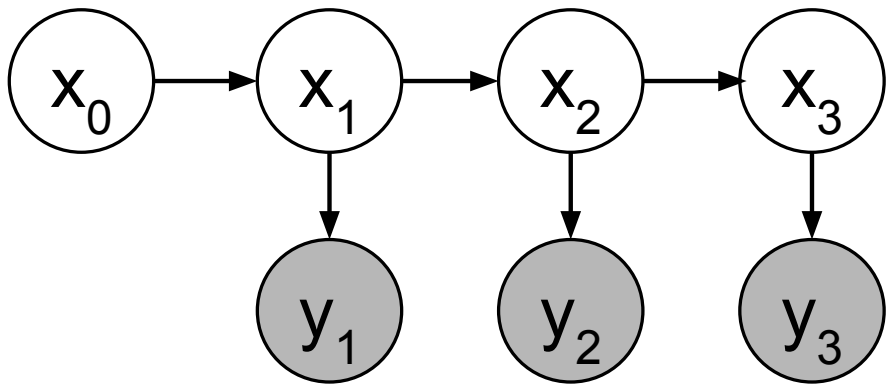


Figure: Hidden Markov Model. Grey nodes are observations.

Joint probability:

$$p(x_0, \ldots x_T; y_1, \ldots y_T) = p(x_0) \prod_{t=1}^{T} p(y_t|x_t) p(x_t|x_{t-1})$$

Factorial Hidden Markov Model

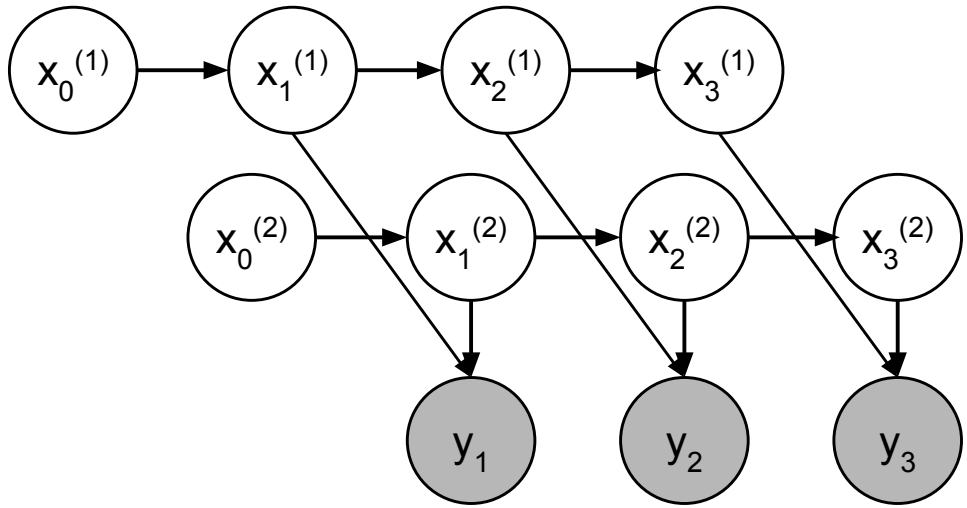


Figure: Factorial Hidden Markov Model.

Extension of HMMs:

- Multiple hidden state sequences in parallel
- Single observation that is a function of all sequences

Factorial Hidden Markov Model

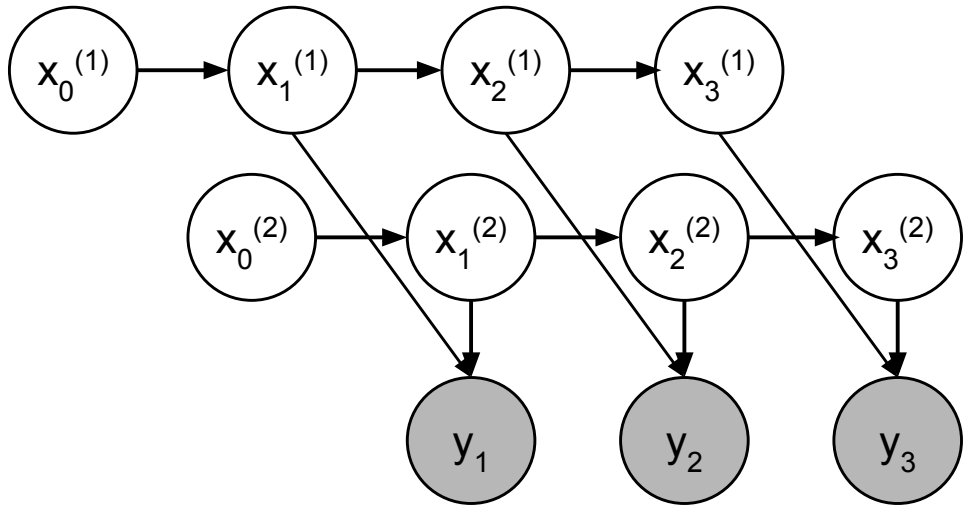


Figure: Factorial Hidden Markov Model.

Extension of HMMs:

- Multiple hidden state sequences in parallel
- Single observation that is a function of all sequences

$$p(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{y}) = p(x_0^{(1)})p(x_0^{(2)}) \prod_{t=1}^{T} p(y_t|x_t^{(1)}, x_t^{(2)})p(x_t^{(1)}|x_{t-1}^{(1)})p(x_t^{(2)}|x_{t-1}^{(2)})$$

where $\mathbf{x}^{(k)} = (x_0^{(k)}, \ldots x_T^{(k)})$ and $\mathbf{y} = (y_1, \ldots y_T)$.

Graph embedding

Book embedding: example

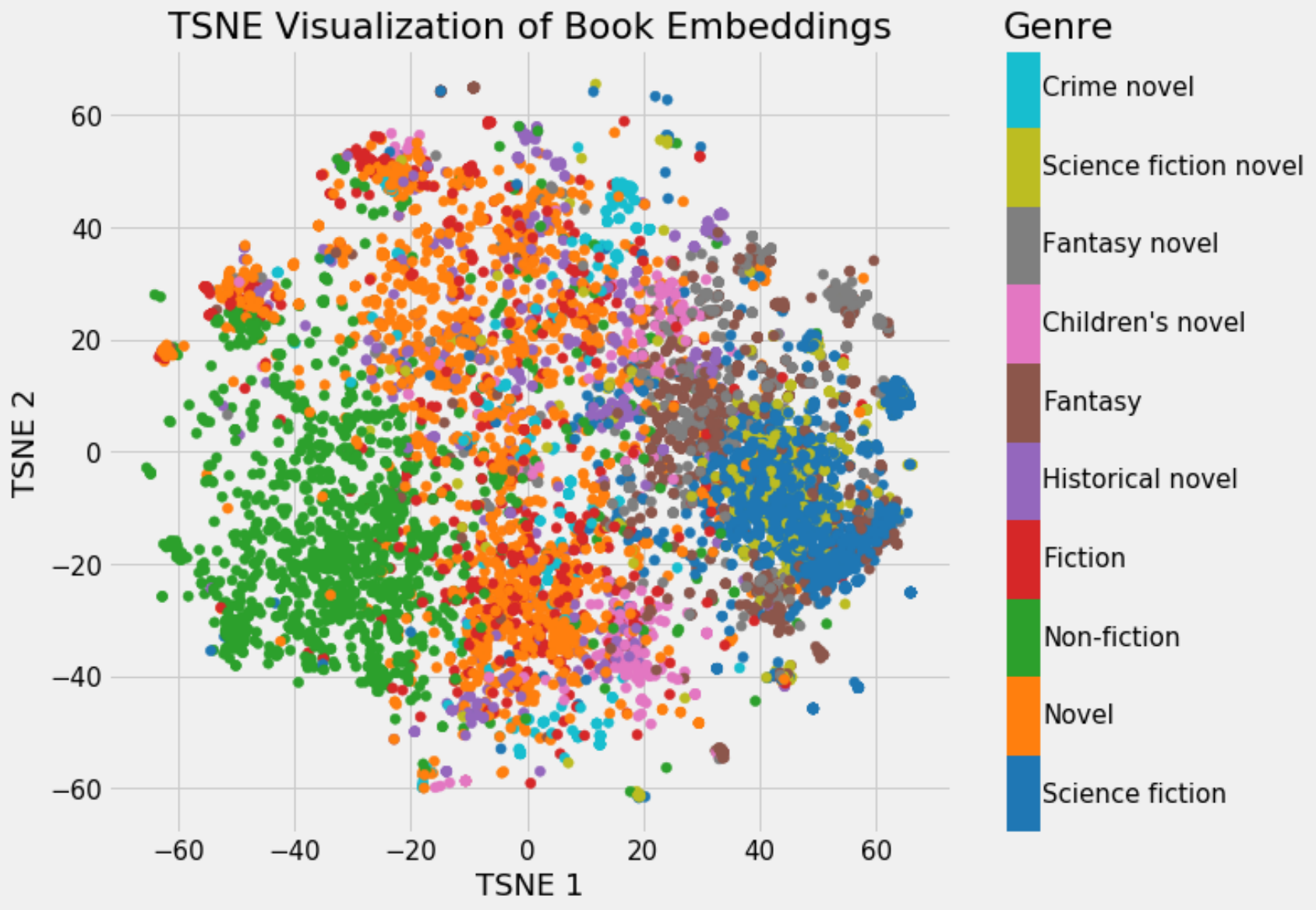


Figure: `https://devopedia.org/word-embedding`

Proteins: example

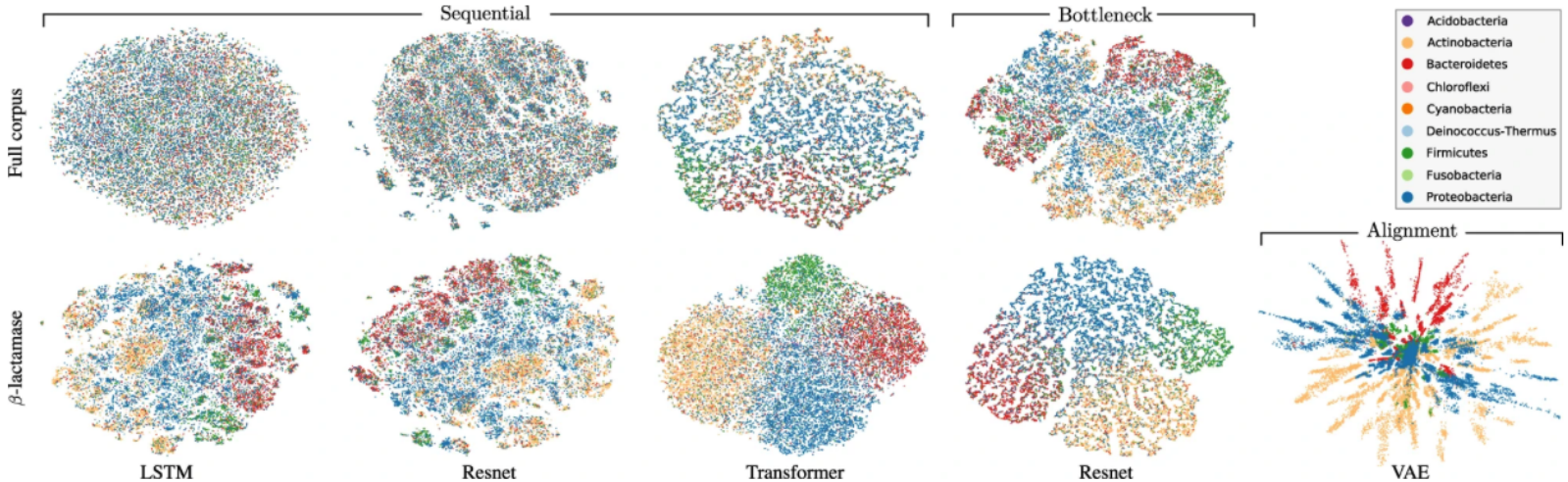
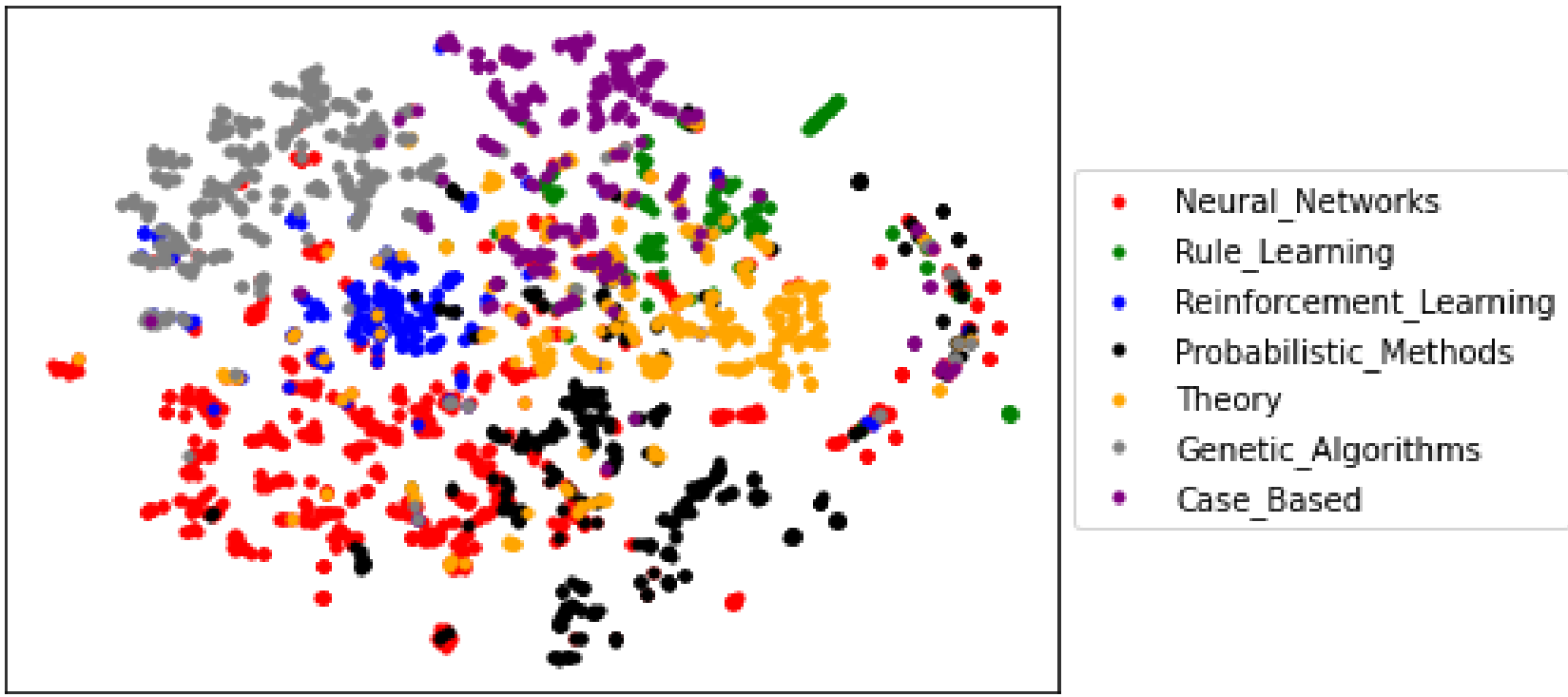


Figure: Detlefsen et al. Learning meaningful representations of protein sequences. Nature Communications. 2022

Graph embedding: example



Random-walk based graph embedding

Family of graph embedding methods inspired by skipgram

Deepwalk, node2vec, etc.

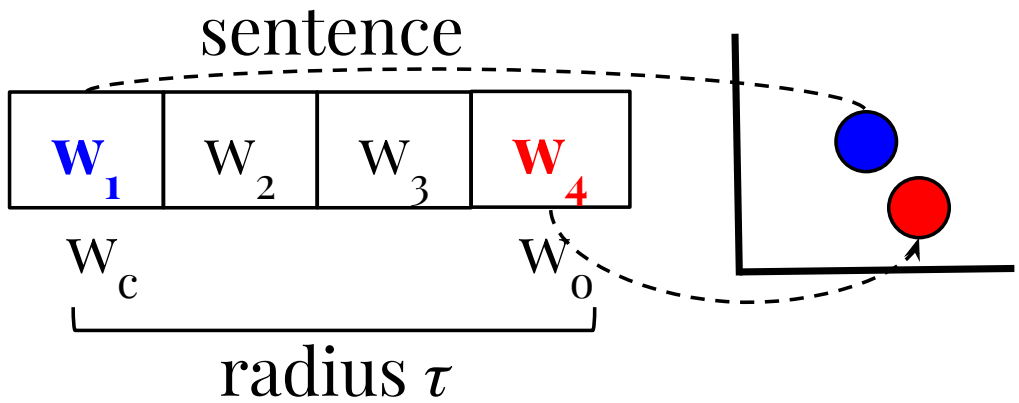


Figure: Skipgram model

Random-walks treated as sentences

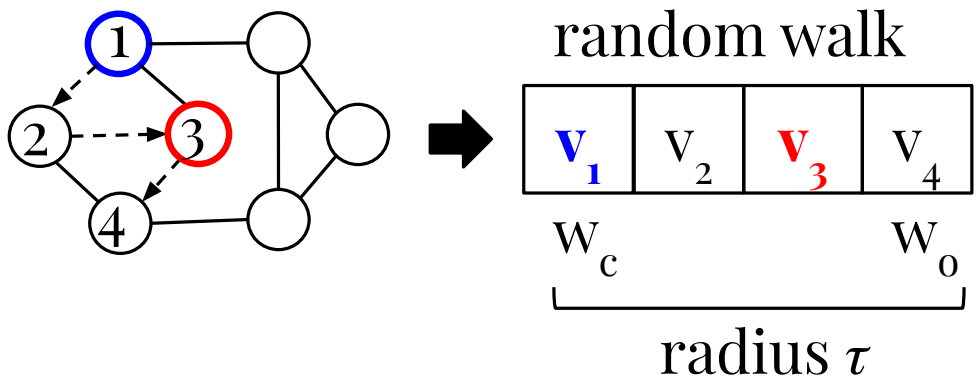


Figure: RW-based graph embedding

Random-walk based graph embedding

Algorithm:

1. Sample walks $\langle v_1^{(i)}, v_2^{(i)} \dots v_L^{(i)} \rangle$ from G
2. Build corpus with pairs $(v_t^{(i)}, v_{t+\tau}^{(i)})$ from each walk
3. Apply skipgram model to the corpus
4. Return embeddings of "word" v for corresponding vertex

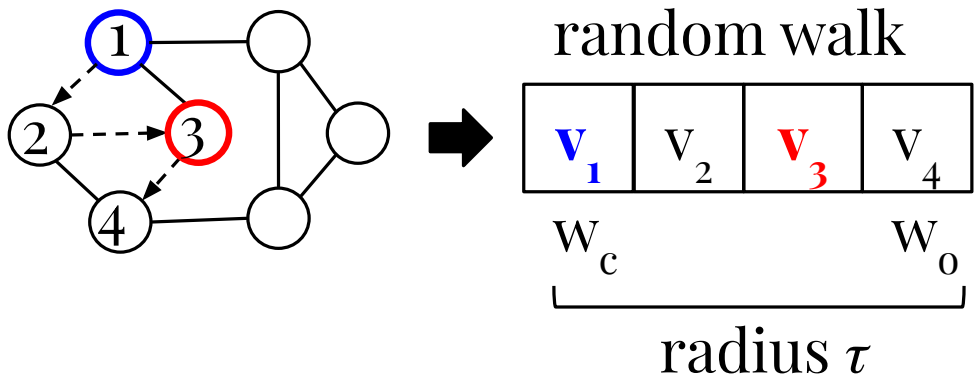
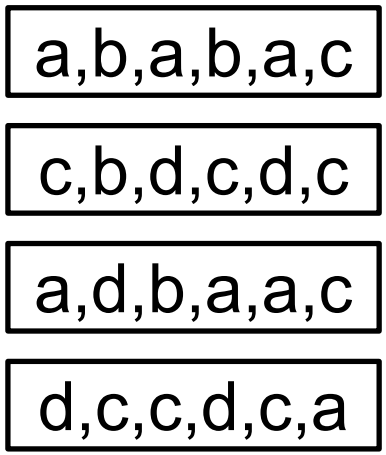


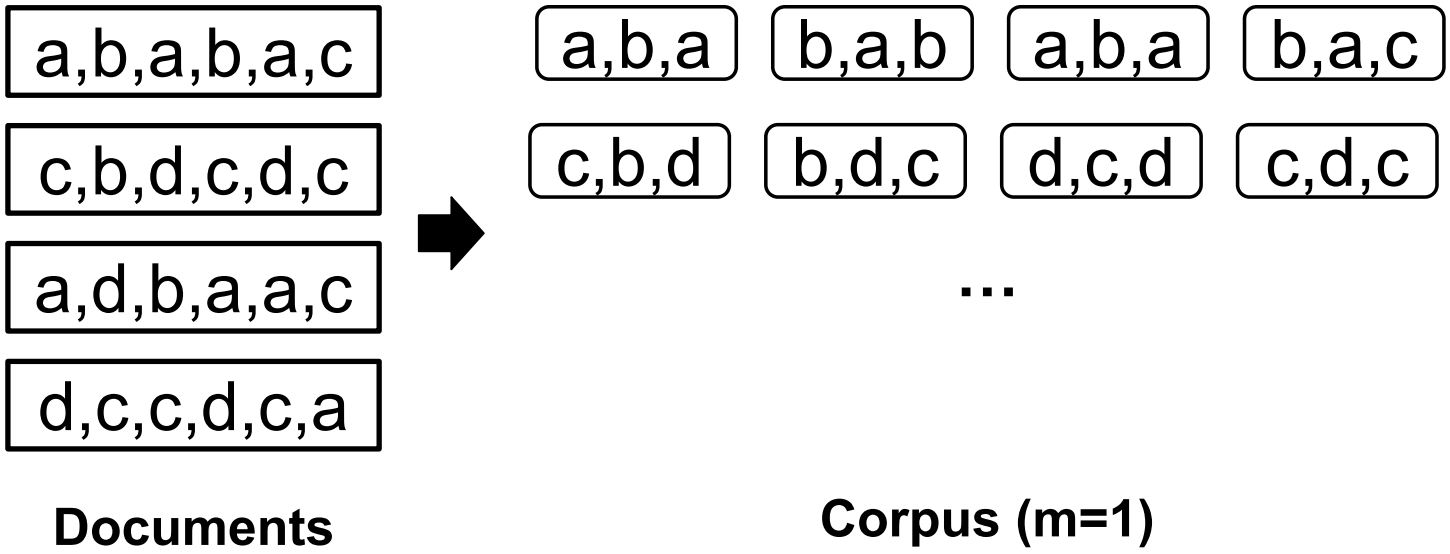
Figure: RW-based graph embedding

Skipgram model

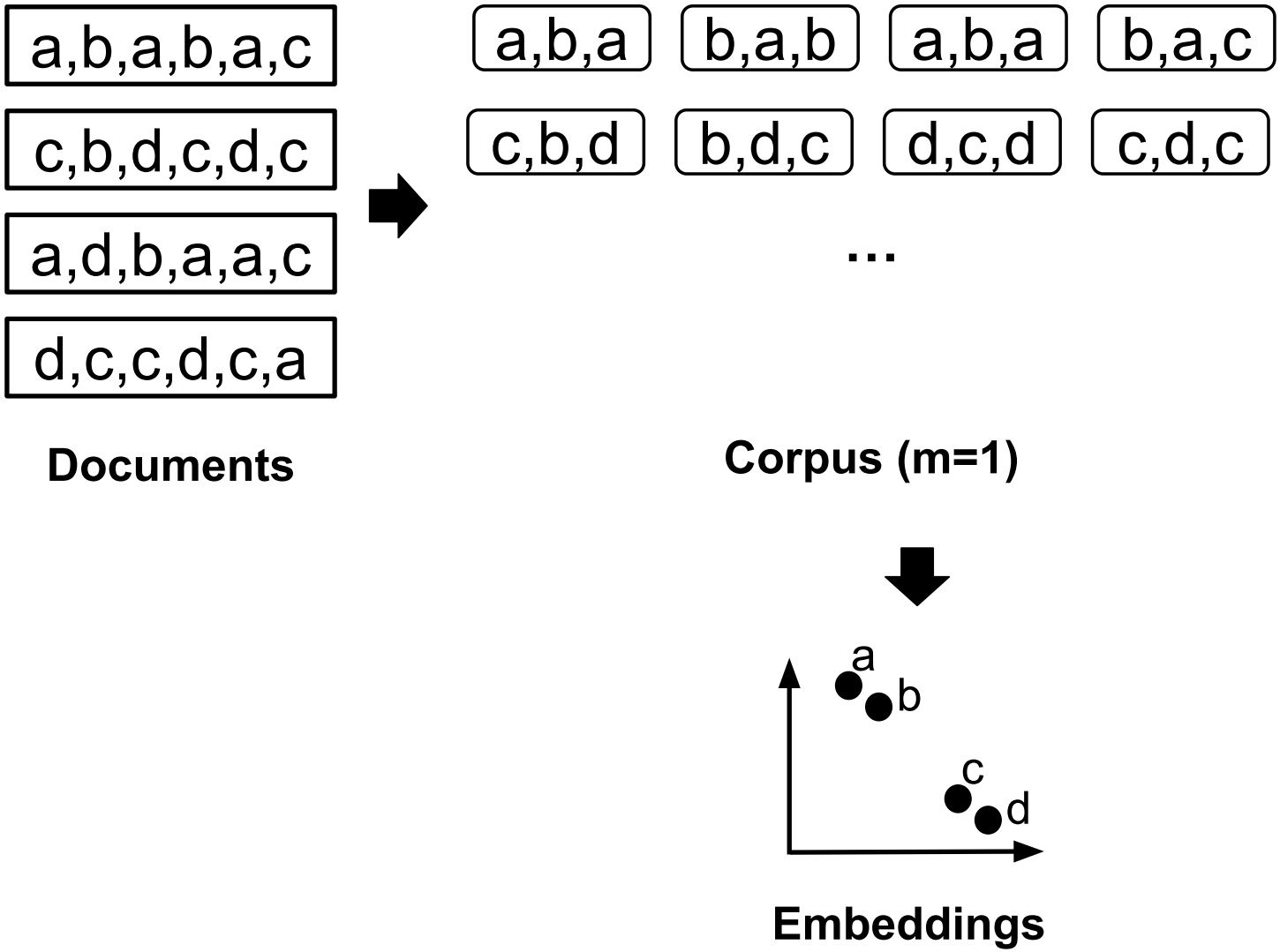


Documents

Skipgram model



Skipgram model



Skipgram model

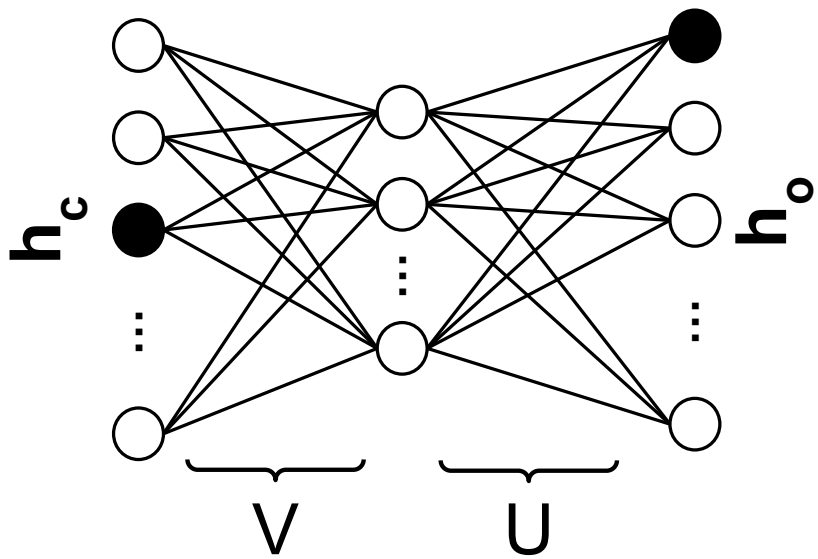


Figure: Skipgram as a neural net.

Loss function as cross-entropy over pairs in the corpus:

$$\mathcal{L}(\Theta) = -\sum_{t=1}^{T} \sum_{-m \leq j \leq m} \log(\langle \sigma(U(V^T \mathbf{h}_t)), \mathbf{h}_{t+j} \rangle)$$

Graph neural networks

Graph Neural Networks (GNNs)

Generalizing the deep learning framework to graph data

$G(V, E, F_V, F_E)$

Graph Neural Networks (GNNs)

Generalizing the deep learning framework to graph data

$G(V, E, F_V, F_E)$

Key tasks (semi-supervised or supervised):

Node classification;

Link prediction;

Graph classification.

Graph Neural Networks (GNNs)

Generalizing the deep learning framework to graph data

$G(V, E, F_V, F_E)$

Key tasks (semi-supervised or supervised):

Node classification;

Link prediction;

Graph classification.

Other successful applications:

Chemistry (e.g. drug design);

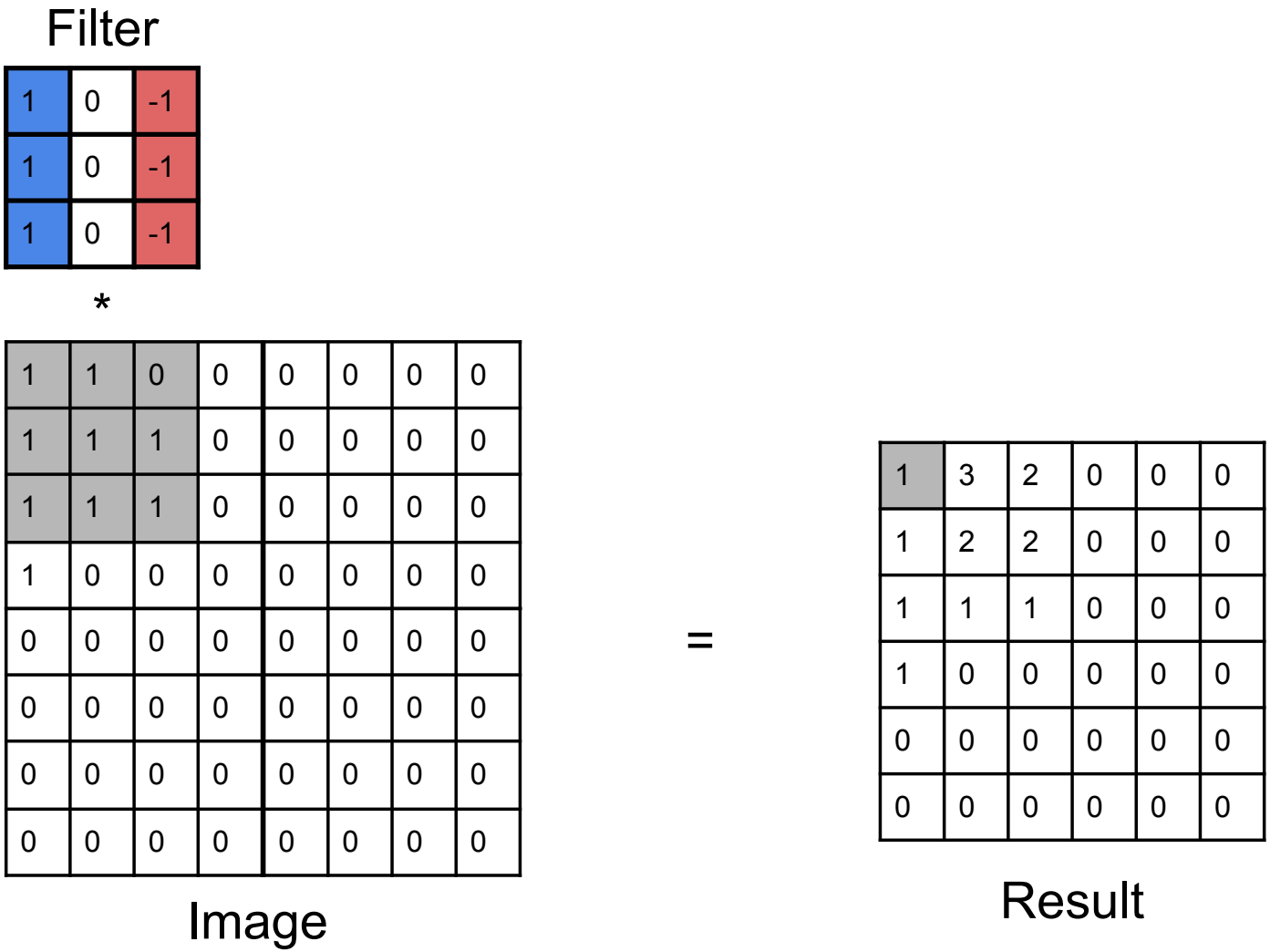
Traffic management (e.g. flow prediction);

Program analysis (e.g. bug discovery);

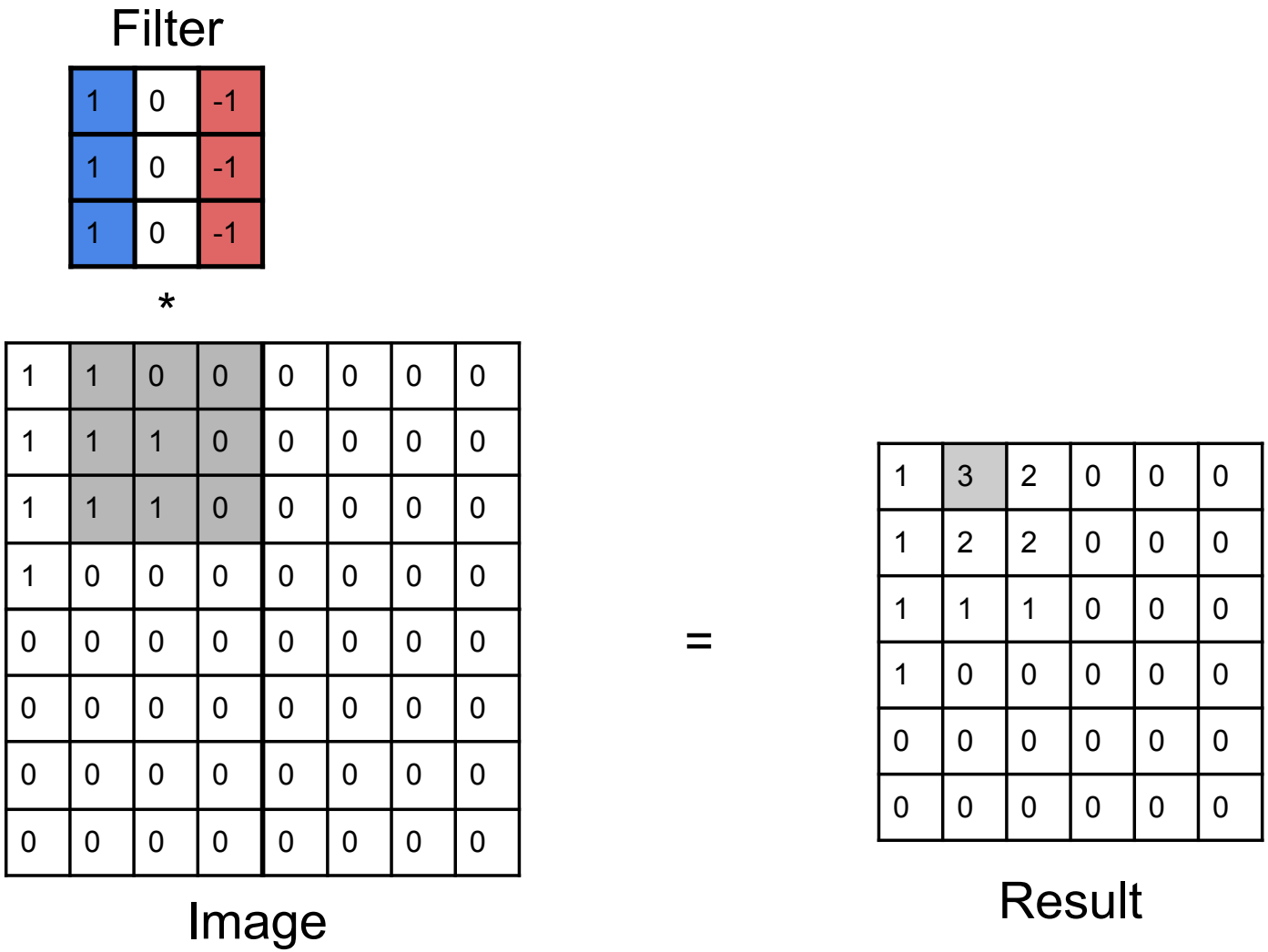
NLP (e.g. question answering);

Recommender systems (e.g. product recommendation).

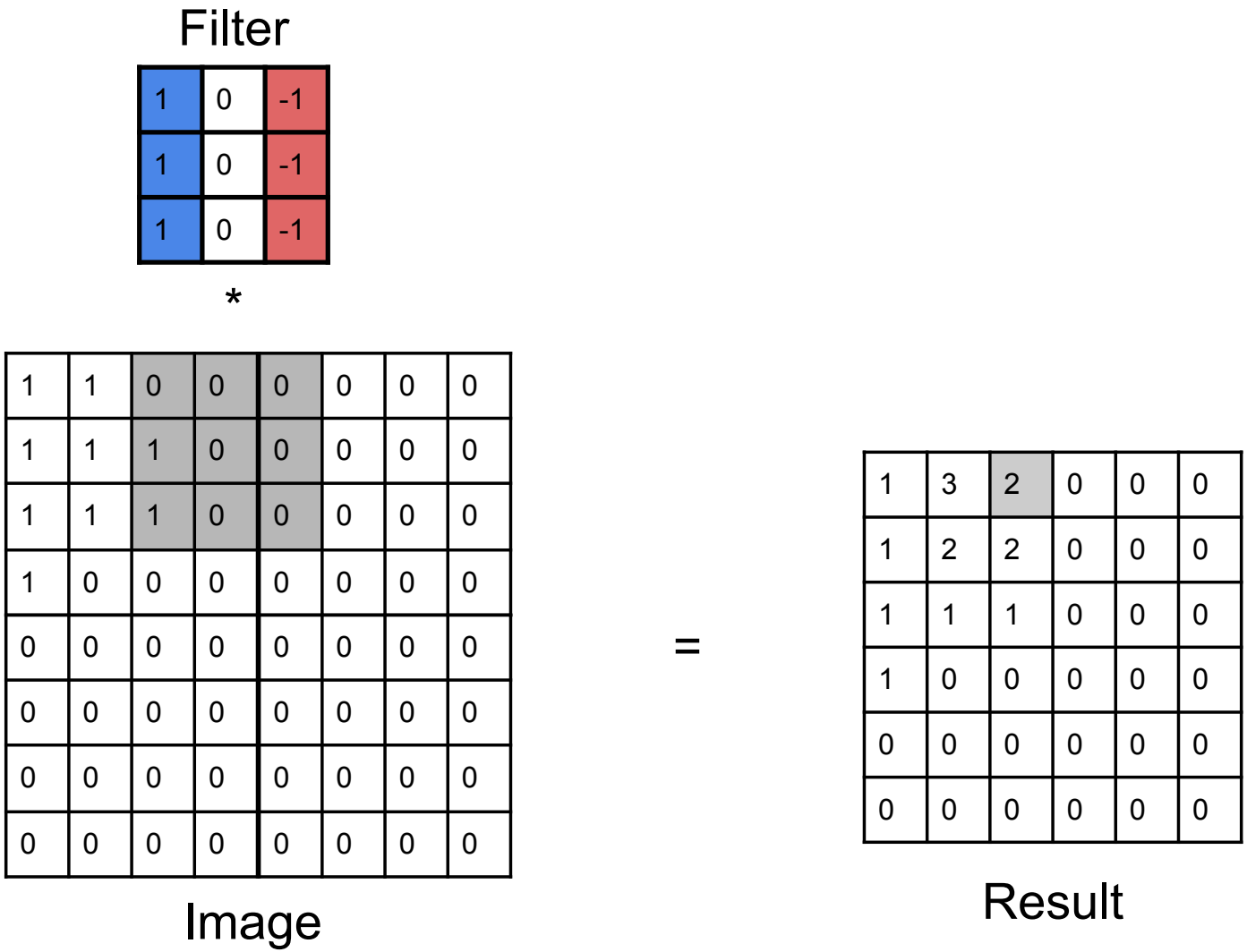
Message-passing neural networks



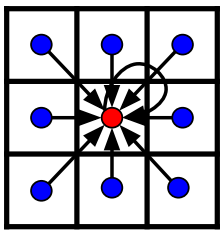
Message-passing neural networks

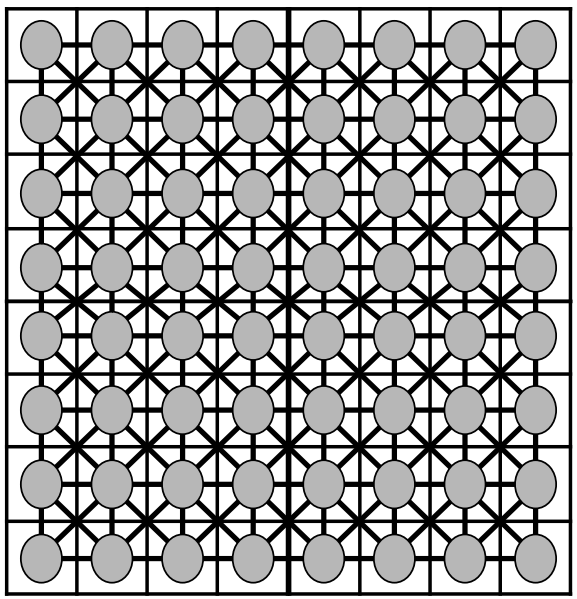


Message-passing neural networks



Message-passing neural networks





Message-passing neural networks

Messages: $\mathbf{m}_{uv}^t = f_e(\mathbf{h}_u^t, \mathbf{h}_v^t, \mathbf{e}_{uv})$

f_e is a learnable function (neural net);

$\mathbf{h}_u \in \mathbb{R}^k$ is a node representation at layer t;

$\mathbf{e}_{uv}$ are edge features.

Message-passing neural networks

Messages: $\mathbf{m}_{uv}^t = f_e(\mathbf{h}_u^t, \mathbf{h}_v^t, \mathbf{e}_{uv})$

f_e is a learnable function (neural net);

$\mathbf{h}_u \in \mathbb{R}^k$ is a node representation at layer t;

$\mathbf{e}_{uv}$ are edge features.

Pooling: $\mathbf{h}_v^{t+1} = f_v(\mathbf{h}_v^t, \cup_{u \in N(v)} \mathbf{m}_{uv}^t)$

f_v is also a learnable function

Permutation invariance

Message-passing neural networks

Messages: $\mathbf{m}_{uv}^t = f_e(\mathbf{h}_u^t, \mathbf{h}_v^t, \mathbf{e}_{uv})$

f_e is a learnable function (neural net);

$\mathbf{h}_u \in \mathbb{R}^k$ is a node representation at layer t;

$\mathbf{e}_{uv}$ are edge features.

Pooling: $\mathbf{h}_v^{t+1} = f_v(\mathbf{h}_v^t, \cup_{u \in N(v)} \mathbf{m}_{uv}^t)$

f_v is also a learnable function

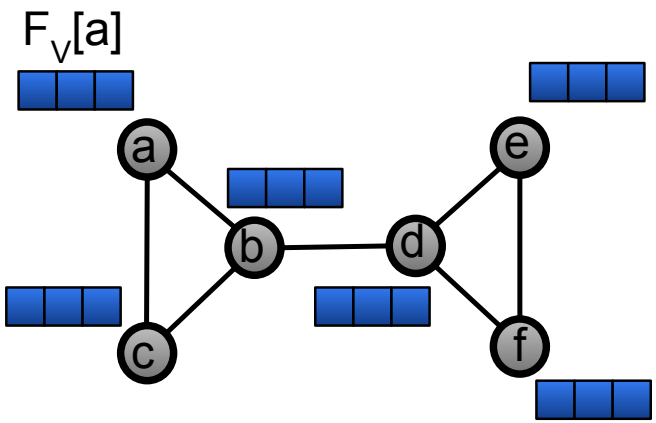
Permutation invariance

$\mathbf{h}_v^0$: node features/attributes

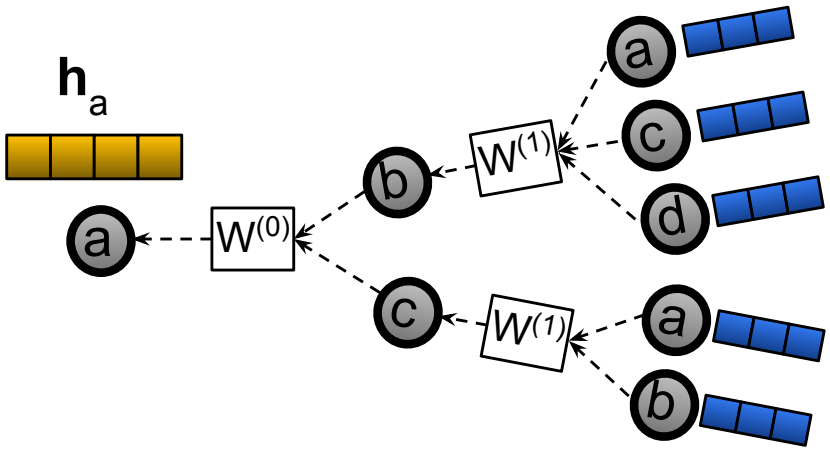
Output: $\mathbf{h}_1^T, \ldots \mathbf{h}_{|V|}^T$

For T layers

Message-passing neural networks



Attributed graph



GNN computational graph

Graph Convolutional Networks and MPNNs

$$Z = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(t)}W^{(t)}$$

where $\tilde{A} = A + I$, $\tilde{D} = \mathbf{diag}(\tilde{A}\mathbf{1})$, $H^{(0)} = F_V$

Graph Convolutional Networks and MPNNs

$$Z = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(t)}W^{(t)}$$

where $\tilde{A} = A + I$, $\tilde{D} = \mathbf{diag}(\tilde{A}\mathbf{1})$, $H^{(0)} = F_V$

Messages: $\mathbf{m}_{uv}^t = f_e(\mathbf{h}_u^t, \mathbf{h}_v^t, \mathbf{e}_{uv}) = \mathbf{h}_u^t$

Graph Convolutional Networks and MPNNs

$$Z = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(t)}W^{(t)}$$

where $\tilde{A} = A + I$, $\tilde{D} = \mathbf{diag}(\tilde{A}\mathbf{1})$, $H^{(0)} = F_V$

Messages: $\mathbf{m}_{uv}^t = f_e(\mathbf{h}_u^t, \mathbf{h}_v^t, \mathbf{e}_{uv}) = \mathbf{h}_u^t$

Pooling:

$$\mathbf{h}_v^{t+1} = f_v(\mathbf{h}_v^t, \cup_{u \in N(v)} \mathbf{m}_{uv}^t)$$

$$= \sigma\left(\frac{1}{\sqrt{|N(v)|+1}} W^t \left(\frac{\mathbf{h}_v^t}{\sqrt{|N(v)|+1}} + \sum_{u \in N(v)} \frac{\mathbf{h}_u^t}{\sqrt{|N(u)|+1}}\right)\right)$$

Graph Convolutional Networks and MPNNs

$$Z = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(t)}W^{(t)}$$

where $\tilde{A} = A + I$, $\tilde{D} = \mathbf{diag}(\tilde{A}\mathbf{1})$, $H^{(0)} = F_V$

Messages: $\mathbf{m}_{uv}^t = f_e(\mathbf{h}_u^t, \mathbf{h}_v^t, \mathbf{e}_{uv}) = \mathbf{h}_u^t$

Pooling:

$$\mathbf{h}_v^{t+1} = f_v(\mathbf{h}_v^t, \cup_{u \in N(v)} \mathbf{m}_{uv}^t)$$

$$= \sigma\left(\frac{1}{\sqrt{|N(v)|+1}} W^t \left(\frac{\mathbf{h}_v^t}{\sqrt{|N(v)|+1}} + \sum_{u \in N(v)} \frac{\mathbf{h}_u^t}{\sqrt{|N(u)|+1}}\right)\right)$$

More scalable, no edge features

GCN for node classification

Goal: predict the labels of nodes based on a few labeled ones

Formulation:

- $X \in \mathbb{R}^{n \times d}$ is the matrix of node attributes
- Labels/classes be $1, \ldots C$
- $\hat{A} = \tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2}$

GCN for node classification

Goal: predict the labels of nodes based on a few labeled ones

Formulation:

- $X \in \mathbb{R}^{n \times d}$ is the matrix of node attributes
- Labels/classes be $1, \ldots C$
- $\hat{A} = \tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2}$

$$Z = \text{softmax}(\hat{A}\text{ReLU}(\hat{A}XW^{(0)})W^{(1)})$$

where:

- $W^{(0)} \in \mathbb{R}^{d \times h}$ and $W^{(1)} \in \mathbb{R}^{h \times C}$ are learnable weights
- $\text{ReLU}(x) = \max(0, x)$
- Softmax is as follows:

$$\text{softmax}(x_c) = \frac{\exp(x_c)}{\sum_{i=1}^{C} \exp(x_i)}$$

GCN for node classification

$W^{(0)}$ and $W^{(1)}$ learned by minimizing cross-entropy:

$$\mathcal{L} = -\sum_{v \in Y_L} \sum_{c=1}^{C} [label(v) = c] \log(Z[v, c])$$

where:

- Y_L is the set of labeled nodes
- $[exp]$ returns 1 if exp is TRUE and 0, otherwise

We minimize $\mathcal{L}$ using your favorite optimization algorithm

GCNs are inductive as long as attributes remain the same

Knowledge graphs

Knowledge graphs

Graph of data intended to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose edges represent potentially different relations between these entities

Graph-based data model

Supports extraction, management, and integration of knowledge

First KG created by Google in 2012

Why knowledge graphs?

Concise and intuitive abstraction

More flexible than a relational database

- They don't require a schema

Support navigational operators on the graph

Supported by scalable graph analytics frameworks

Examples

DBPedia: KG based on Wikipedia

More than 6M entities, 10B relations

Google Knowledge Graph

5B entities, 500B relations

FreeBase: owned by Google (discontinued)

50M entities, 3B relations

Graph-based data models: DEL

Directed edge-labelled graphs: nodes with directed labelled edges between them

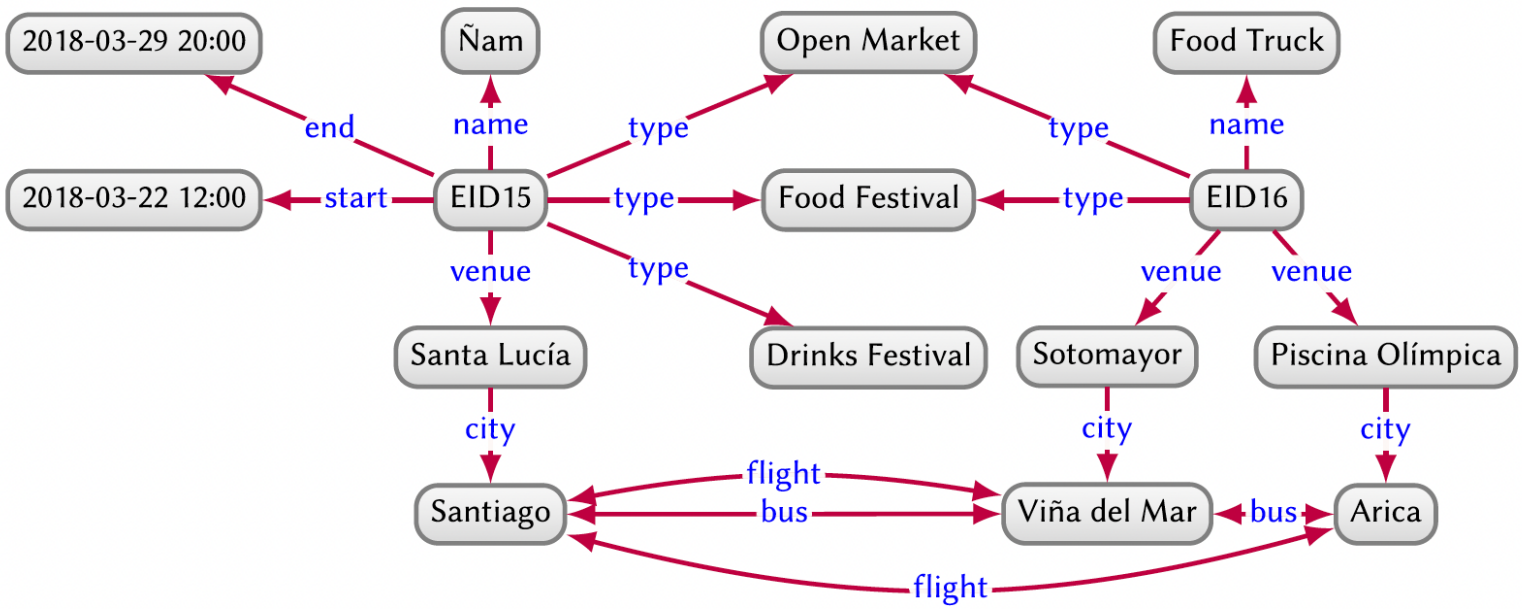
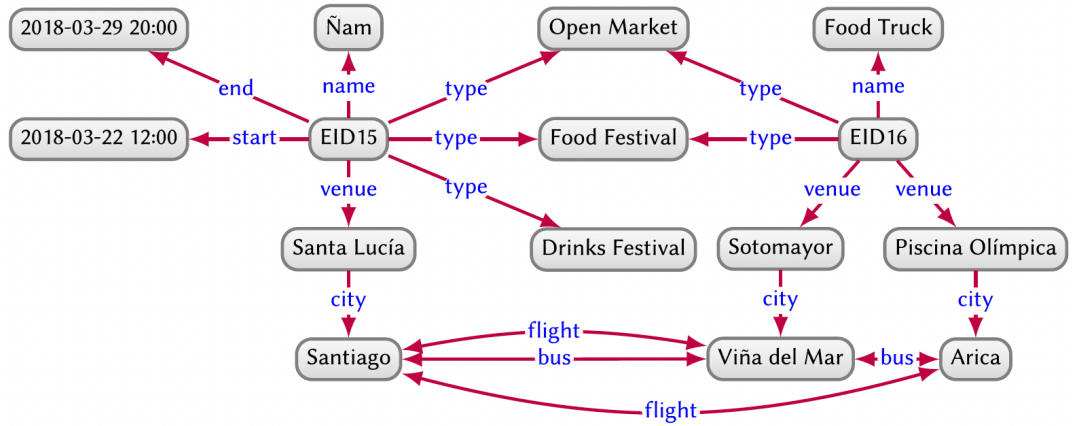


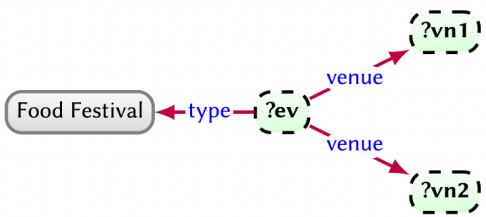
Figure: Directed edge-labelled graph.

Querying KGs

Simple graph pattern: graph with variables and constants



(a) DEL graph



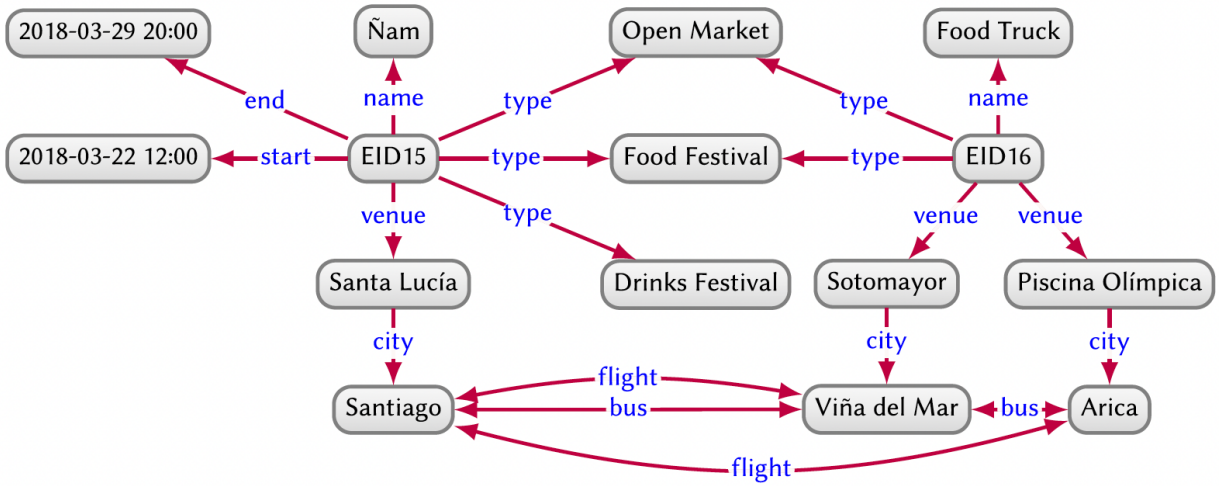
?ev	?vn1	?vn2
EID16	Piscina Olímpica	Sotomayor
EID16	Sotomayor	Piscina Olímpica
EID16	Piscina Olímpica	Piscina Olímpica
EID16	Sotomayor	Sotomayor
EID15	Santa Lucía	Santa Lucía

(b) Pattern and its mappings

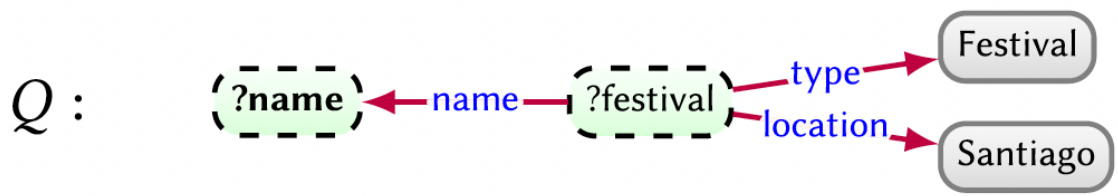
Figure: DEL graph, example of a pattern with mappings.

Deductive knowledge

Going beyond what edges indicate



(a) DEL graph



(b) Query

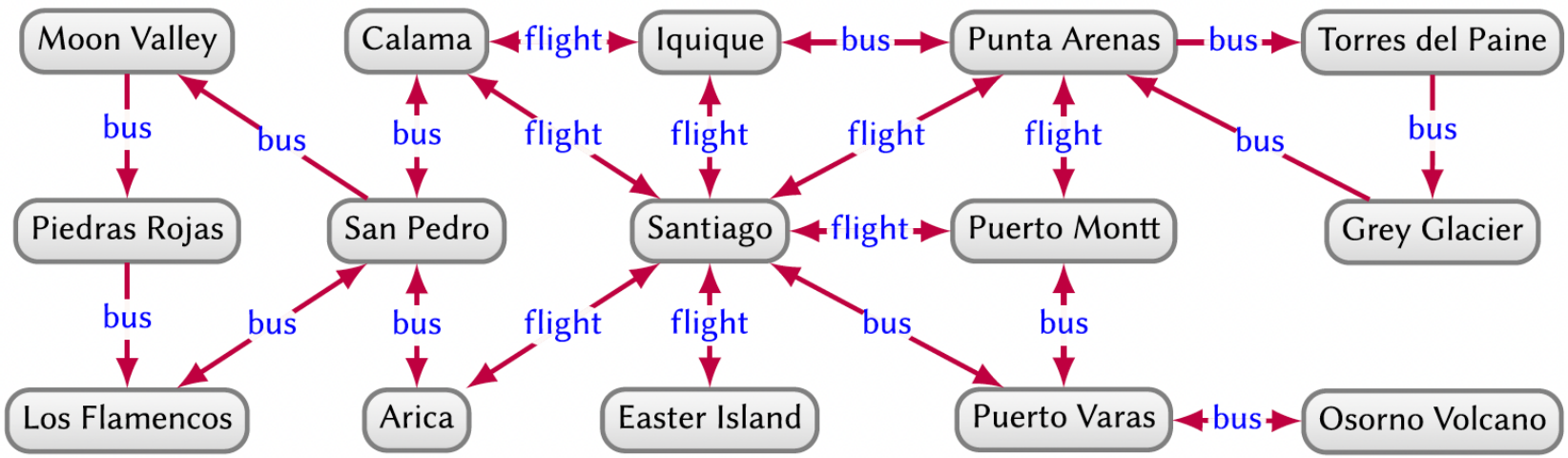
Figure: DEL graph and query.

Inductive knowledge: GNNs

Supervised setting

E.g. Which locations need tourist information offices?

Node classification problem



More examples

Example: weather forecasting

Problem: Given the current weather conditions, predict future weather in a given location.

Conditions: temperature, wind (3D), precipitation, humidity, etc.

State-of-the-art: Integrated Forecasting System (IFS)

- By the European Centre for Medium-Range Weather Forecasts (ECMWF)
- HRES: 10-day forecast, at 0.1-lat lat-long resolution
- ENS: 50 stochastically perturbed 15-day forecasts, 0.2-deg resolution, several times/day

Example: GraphCast

Encode-process-decode auto-regressive model[1]

- Encode: two input frames $\rightarrow$ internal mesh representation
- Process: message-passing on the mesh
- Decode: mesh representation $\rightarrow$ output frame
- $X^{t+1} = \mathit{GraphCast}(X^t, X^{t-1})$

ERA5 dataset

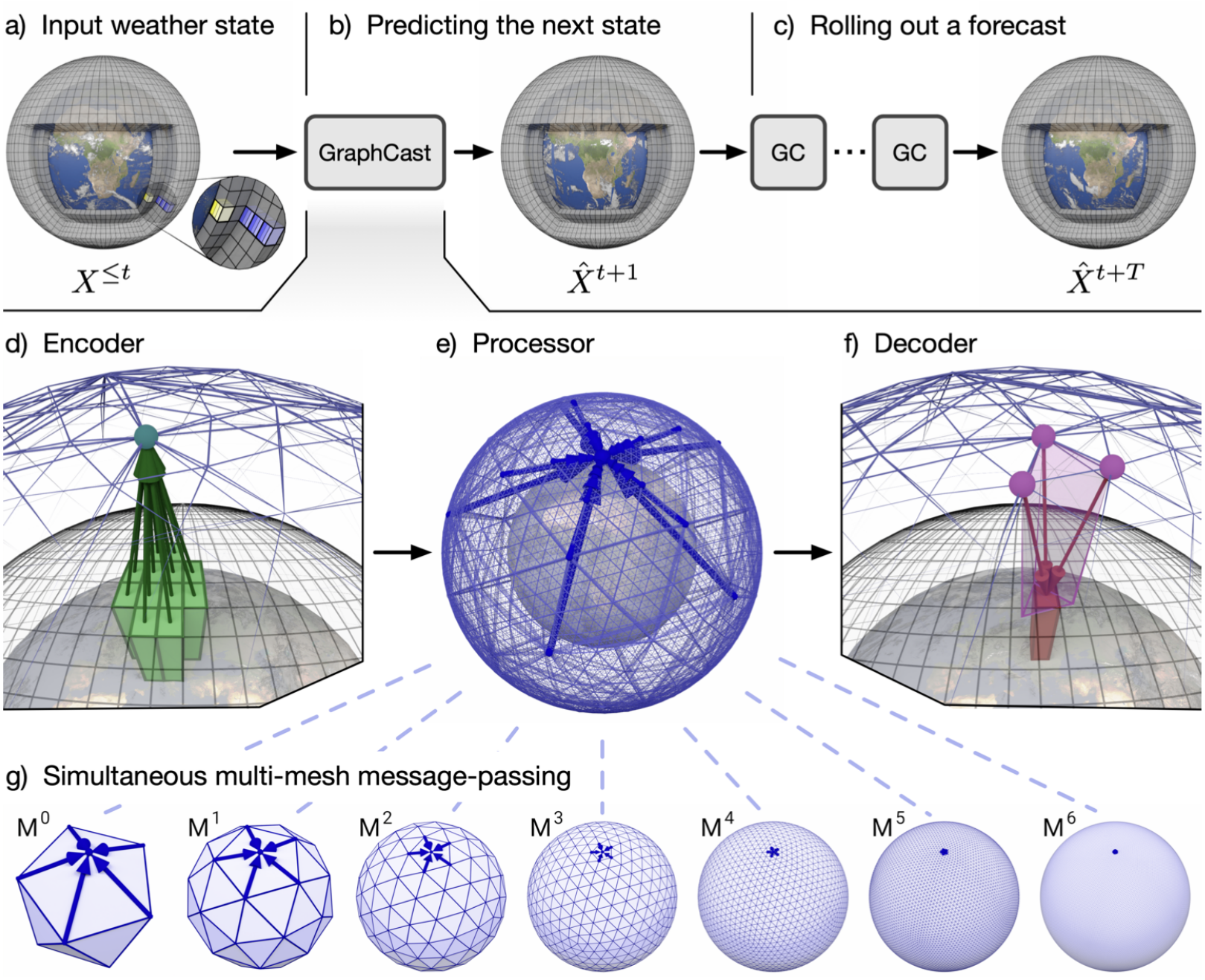
- 1979-2018, 6-hour intervals, 0.25-deg, 37 pressure levels
- 227 target variables/cell (temperature, wind, precipitation)

Multi-mesh message-passing

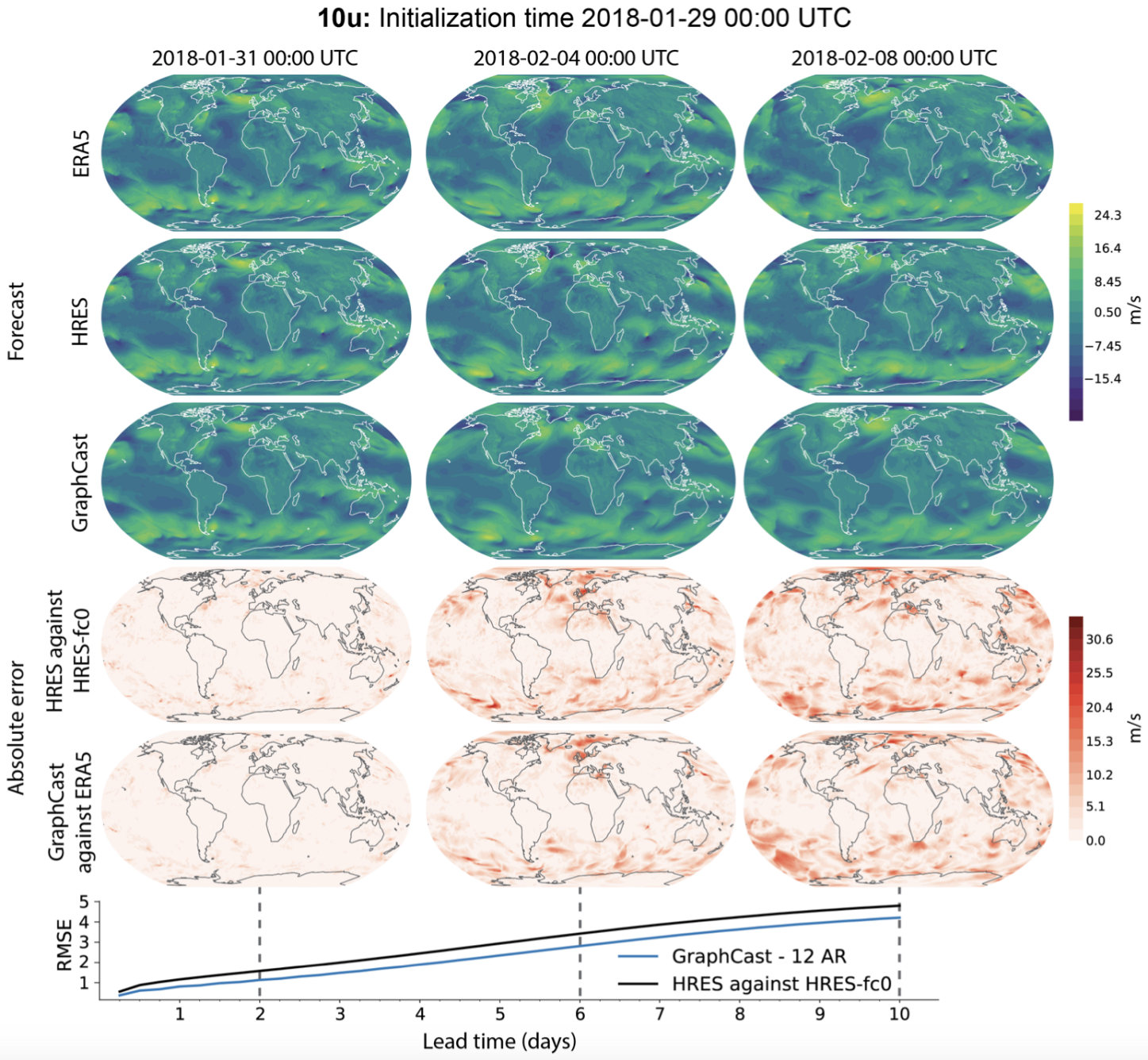
- Earth's surface triangulated at multiple scales
- Encoder: Earth regions to mesh cells (directional)
- Processor: Neighbor mesh cells (bi-directional)
- Decoder: Mesh cells to earth regions (directional)

[1] Lam et al. *Learning skillful medium-range global weather forecasting*. Science, 2023.

Example: GraphCast



Example: GraphCast results



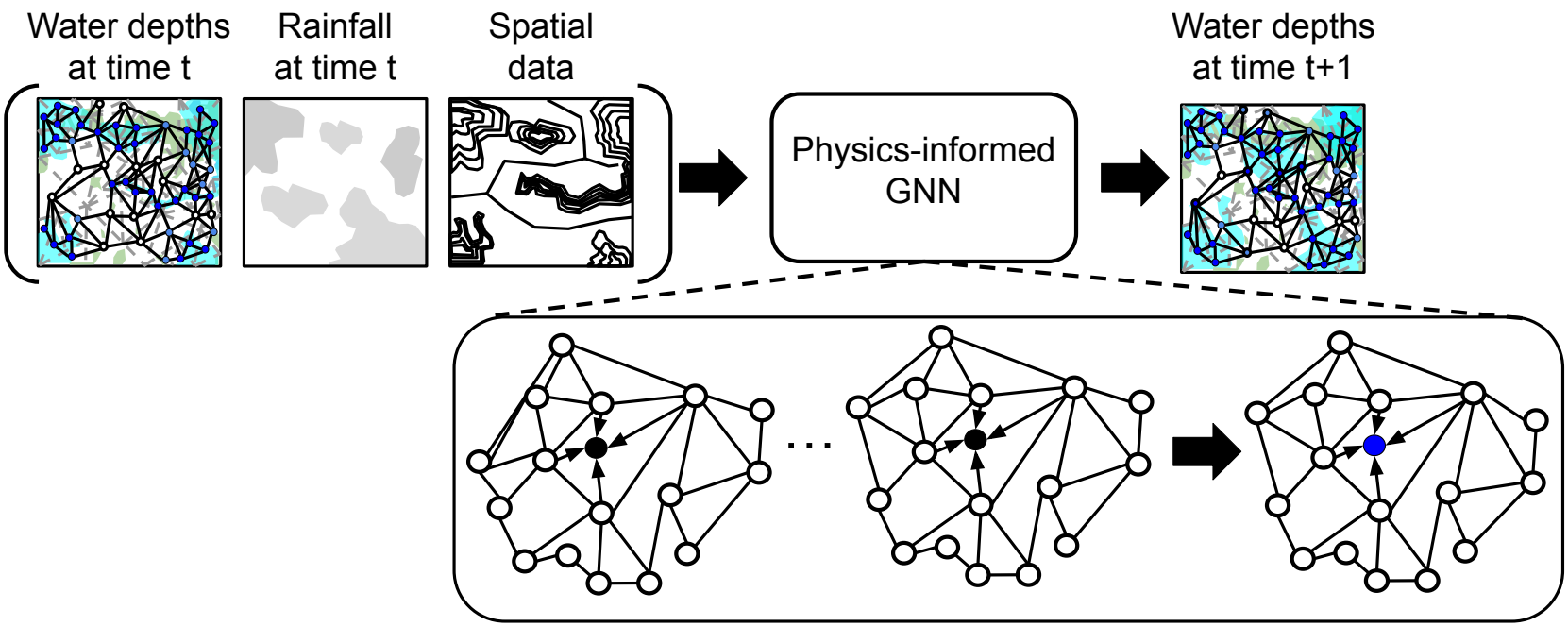
Scientific Machine Learning

Improving (e.g. flooding, traffic) simulations with ML?

Physics-inspired graph neural network for flooding:

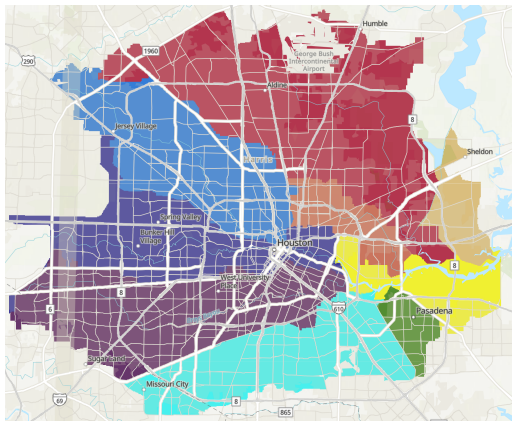
$$\mathbf{q}_i^t = \mathbf{q}_i^{t-1} + \sigma(\mathbf{e}_i^t) \odot \mathrm{MLP}((\mathbf{e}_j^t + \mathbf{z}_j) - (\mathbf{e}_i^t + \mathbf{z}_i))$$

$$h_i^t = h_i^t + \mathrm{MLP}(\sum_k \mathbf{q}_k^t - \mathbf{q}_i^t)$$



Experiments

9 sub-watersheds from Harris County, 7 historical events



Method	RMSE ↓		NSE ↑		r ↑		CSI ↑ ($t_r = 40$)	
	$t_r = 20$	$t_r = 40$	$t_r = 20$	$t_r = 40$	$t_r = 20$	$t_r = 40$	0.001m	0.01m
MLP	0.0256	0.0749	0.1050	0.5446	-0.0028	0.5416	0.2524	0.3499
GCN	0.0315	0.0807	0.0721	0.5077	0.0662	0.5530	0.2370	0.2195
GAT	0.0285	0.0807	0.0866	0.5077	0.0375	0.4926	0.2345	0.1848
U-net	0.0246	0.1281	0.1132	0.2900	0.6083	0.6839	0.2294	0.6142
GNN	0.0263	0.0822	0.1002	0.4983	0.1752	0.5077	0.2476	0.3462
Ours	**0.0046**	**0.0503**	**0.7877**	**0.7264**	**0.8659**	**0.7928**	**0.8275**	**0.8665**

Proposed solution approximates simulator better than baselines but is 1000× faster

Conclusion

To learn more: online courses

Stanford: Machine Learning with Graphs

- Instructor: Jure Leskovec
- Link: `https://www.youtube.com/watch?v=JAB_plj2rbA&list=PLoROMvodv4rPLKxIpqhjhPgdQy7imNkDn`
- Mostly GNNs

UPenn: Machine Learning on Graphs

- Instructor: Alejandro Ribeiro
- Link: `https://www.youtube.com/@alelabalelab5337/playlists`
- GNNs+signal processing

To learn more: online courses

CMU: Probabilistic Graphical Models

- Instructor: Eric Xing
- Link: `https://www.youtube.com/watch?v=oqvdH_8lmCA&list=PLoZgVqqHOumTqxIhcdcpOAJOOimrRCGZn`
- Only graphical models

HSE: Network Science

- Instructor: Leonid Zhukov
- Link: `https://www.youtube.com/watch?v=1T5-BG6yngM&list=PLriUvS7IljvkGesFRuYjqRz4lKgodJgh2`
- Not focused on learning

To learn more: online courses

National University of Singapore: Graph Machine learning

- Instructor: Xavier Bresson
- Link: `https://storage.googleapis.com/xavierbresson/index.html`
- Very compressed!

To learn more: books

Christopher M Bishop. Pattern recognition and machine learning. Springer, 2006

John A Lee and Michel Verleysen. Nonlinear dimensionality reduction. Springer Science & Business Media, 2007

Fan RK Chung and Fan Chung Graham. Spectral graph theory. American Mathematical Soc., 1997

Mark Newman. Networks. Oxford University press, 2018

Yao Ma and Jiliang Tang. Deep Learning on Graphs. Cambridge University Press, 2021

Summary of this lecture

Learning goals:

1. Understand what are graphs and why graphML matters
2. Familiarize with real examples of graphML
3. Recognize graphML applications
4. Have an overview of the graphML toolkit

Topics:

- Intro to graphs
- Network science
- GraphML problems
- Graph matrices and spectral methods
- Label propagation
- Graphical models
- Graph embedding
- Graph neural networks
- Knowledge graphs

Machine Learning with Graphs

K2I AI and ML Boot Camp, 2025

Arlei Silva

Rice University, Houston, TX