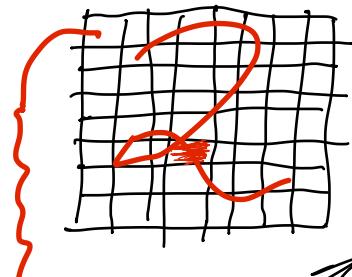


GRAPH NEURAL NETWORKS AND NEURAL-SYMBOLIC COMPUTATION

MNIST

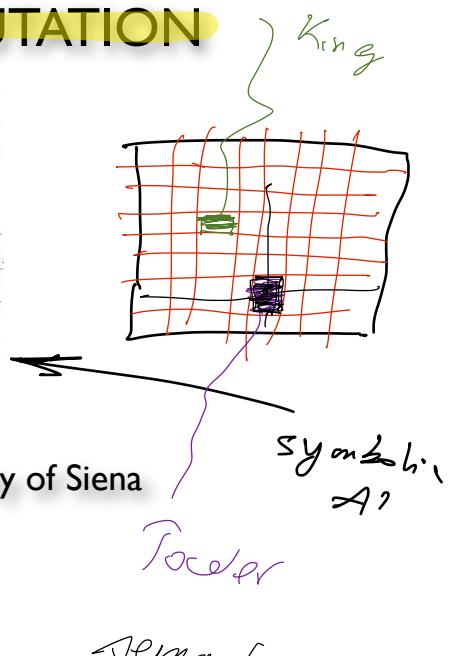
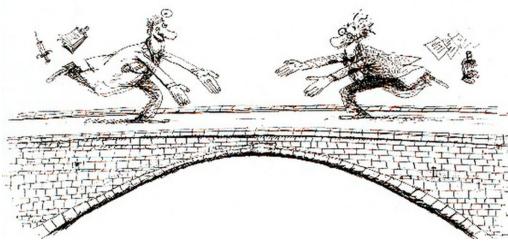
Sub-symb.



0-251

Machine
Learn

Marco Gori
University of Côte d'Azur and University of Siena



MSc DSAI 2022

continuous

math



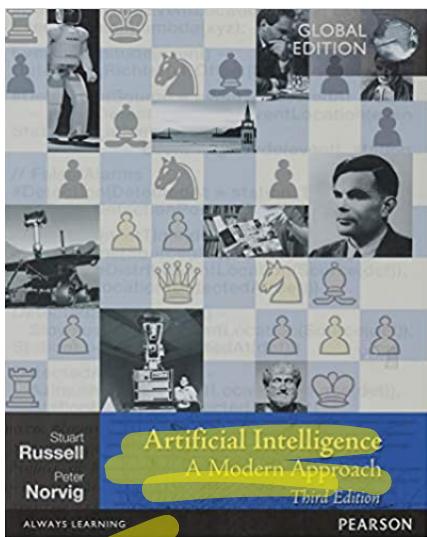
GRAPH NEURAL
NETWORKS (GNN)

NEURAL-SYMBOLIC
COMPUTATION (NESY)

EXPLAINABLE AI

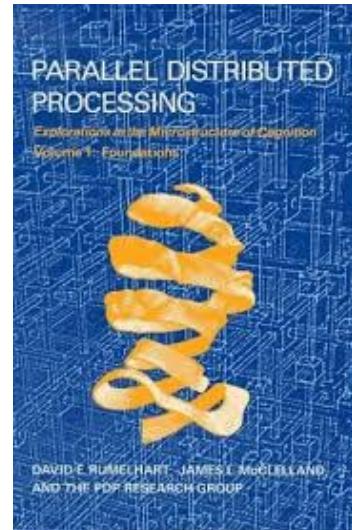
MSc DSAI 2022

COLLECTION OF SUBJECTS



UNIFIED (RESTRICTIVE) VIEW OF INTELLIGENCE

?



MSc DSAI 2022

OUTLINE

- Graph Neural Networks (GNN)
- Neural-Symbolic Computing the next frontier?

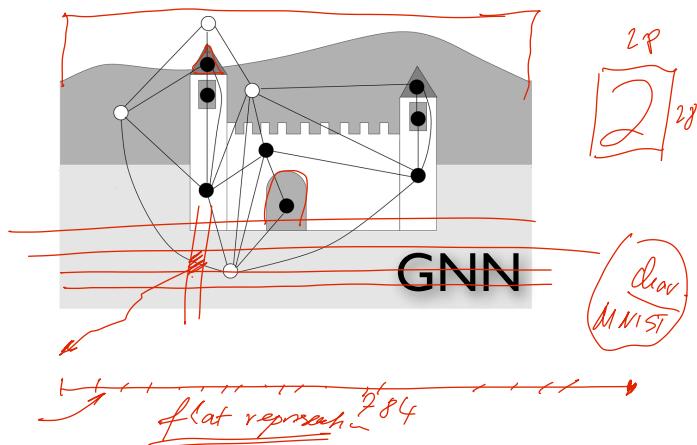
MSc DSAI 2022

Lecturers

- Graph Neural Networks (GNN), lectures,
Marco Gori
- GNN lab activities, Matteo Tiezzi, UNISI
- Neural-Symbolic Computation (NeSy), lectures,
Marco Gori
- NeSy lab activities (Giuseppe Marra, KULeuven)

MSc DSAI 2022

AN INTRODUCTION TO GRAPH NEURAL NETWORKS



MSc DSAI 2022

OUTLINE

- Graphs and internal representations in Machine Learning
- Graph Neural Networks (GNN)
- Data and knowledge diffusion by GNN

MSc DSAI 2022

WHERE DOES GNN COME FROM?

by Matej Balog (Univ. of Cambridge and Deep Mind)

Historical look: Scarselli et al. [2009]

[Scarselli et al. 2009]

- No explicit COMBINE function
- Learning: **Almeida-Pineida** learning algorithm

GGNN [Li et al., 2015]

- COMBINE is bias + GRU
- Learning: **Backprop-through-time** after unrolling T steps

IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 20, NO. 1, JANUARY 2009

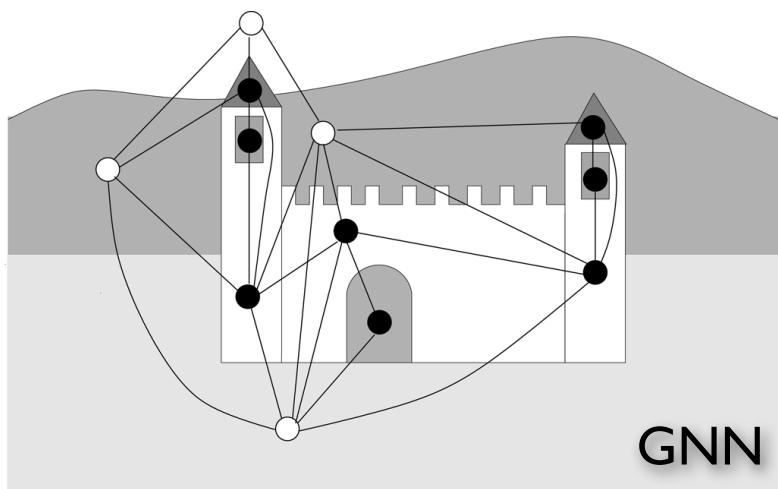
61

The Graph Neural Network Model

Franco Scarselli, Marco Gori, *Fellow, IEEE*, Ah Chung Tsoi, Markus Hagenbuchner, *Member, IEEE*, and Gabriele Monfardini

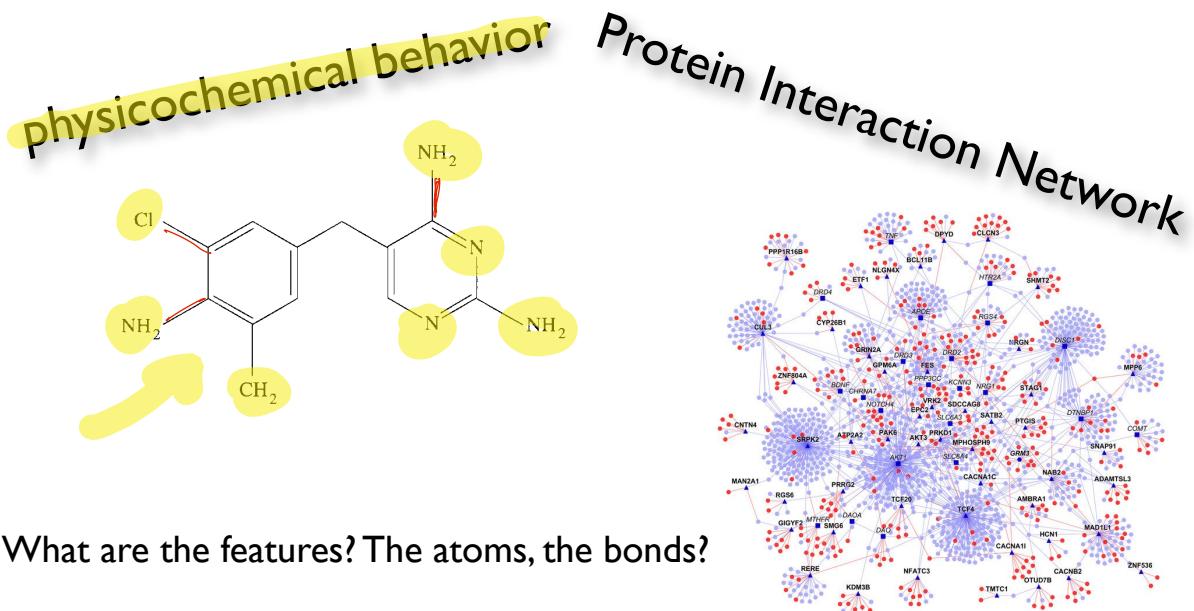
MSc DSAI 2022

WHY MODELING THE ENVIRONMENT BY GRAPHS?



MSc DSAI 2022

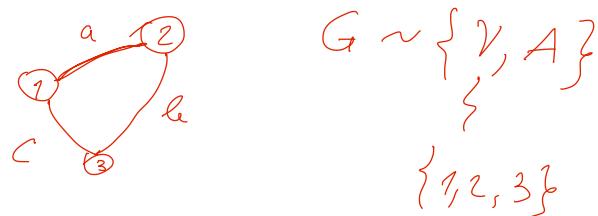
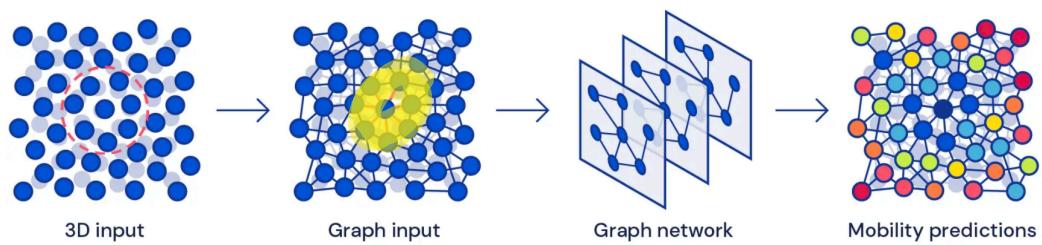
CHEMISTRY AND BIOLOGY



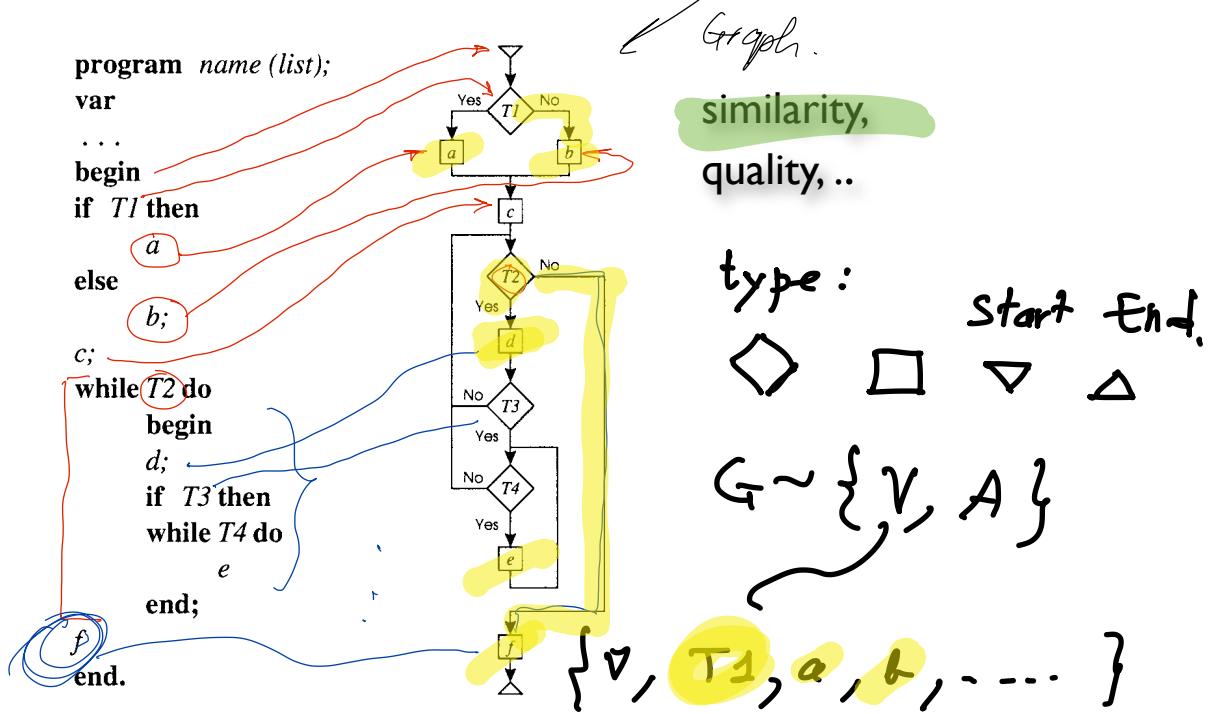
What are the features? The atoms, the bonds?

UNDERSTANDING GLASSES

DeepMind
Towards understanding glasses with graph neural networks

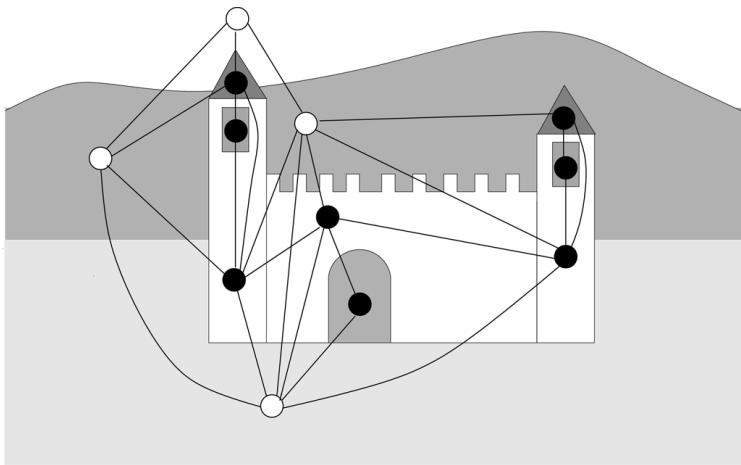


SOFTWARE ENGINEERING



PATTERN RECOGNITION

Structured representations



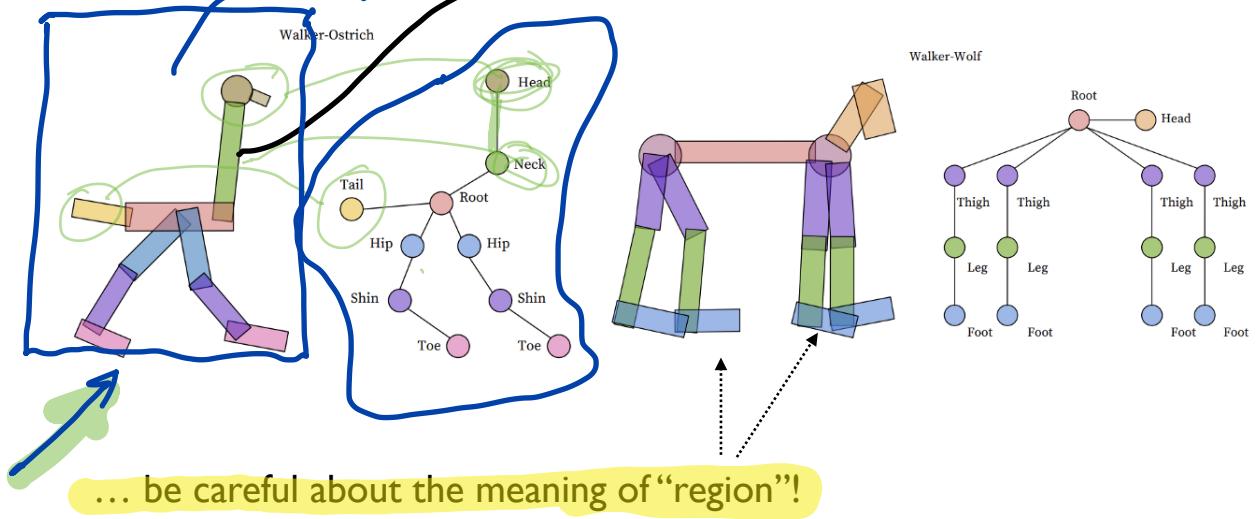
Pattern recognition community: enormous tradition
(e.g. syntactic pattern recognition, Horst Bunke, ...)

Inspiring example ...

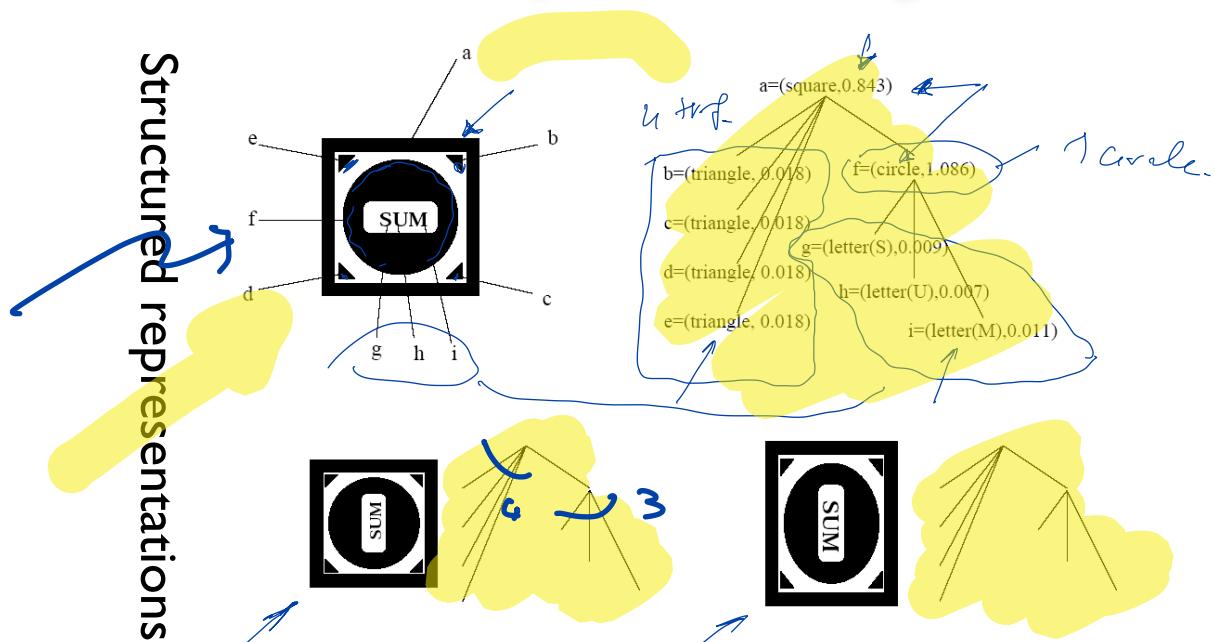
Wang et al 2017

flat repr.

homogeneous

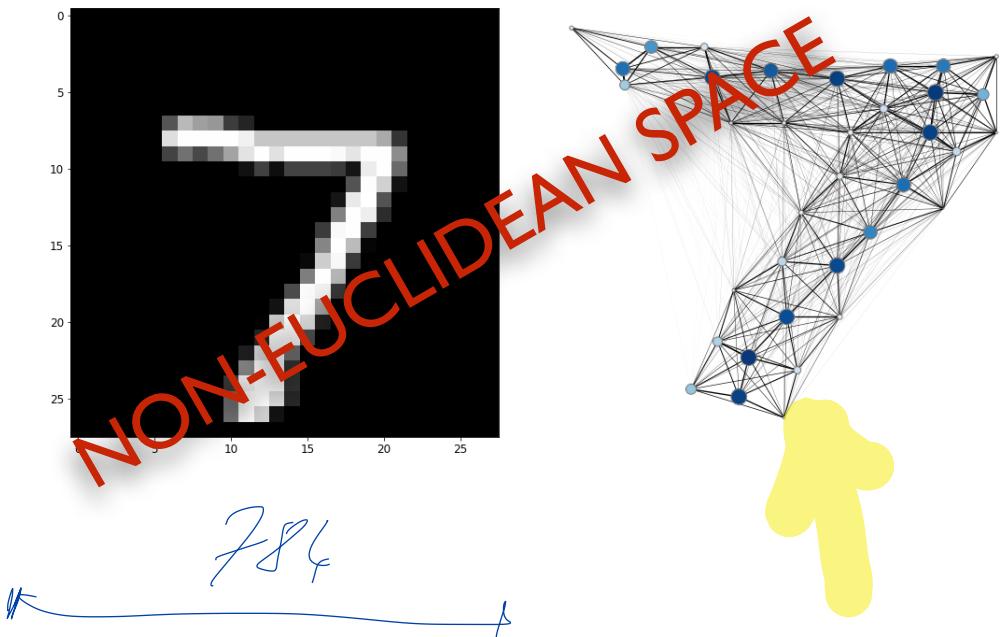


Document Analysis and Recognition



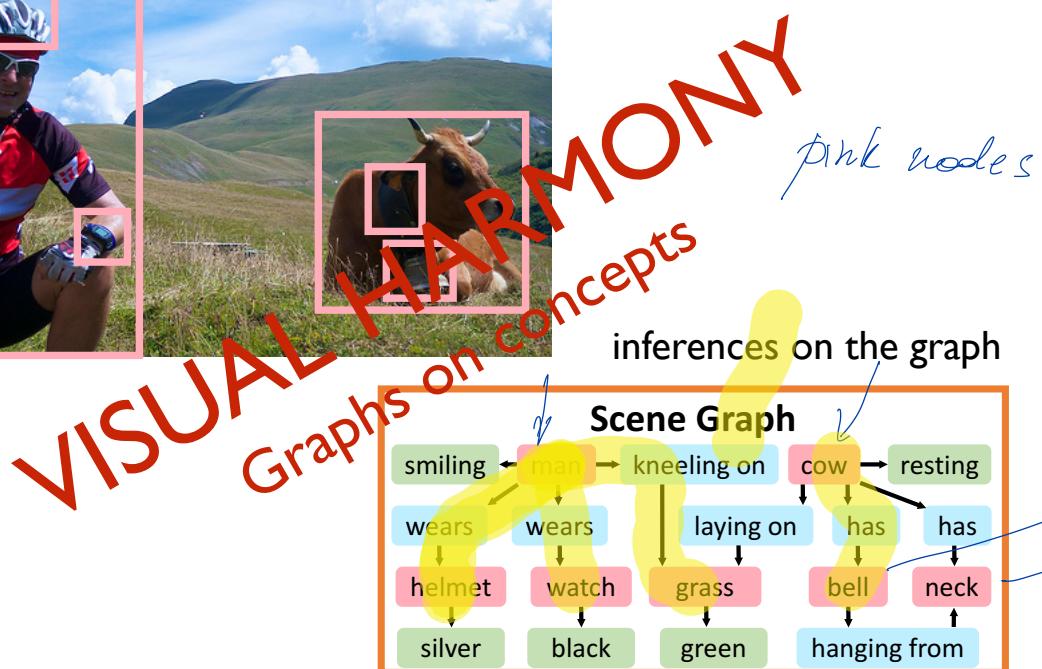
Another example: XY-trees for Document Analysis and Recognition

MNIST Handwritten Char Recognition

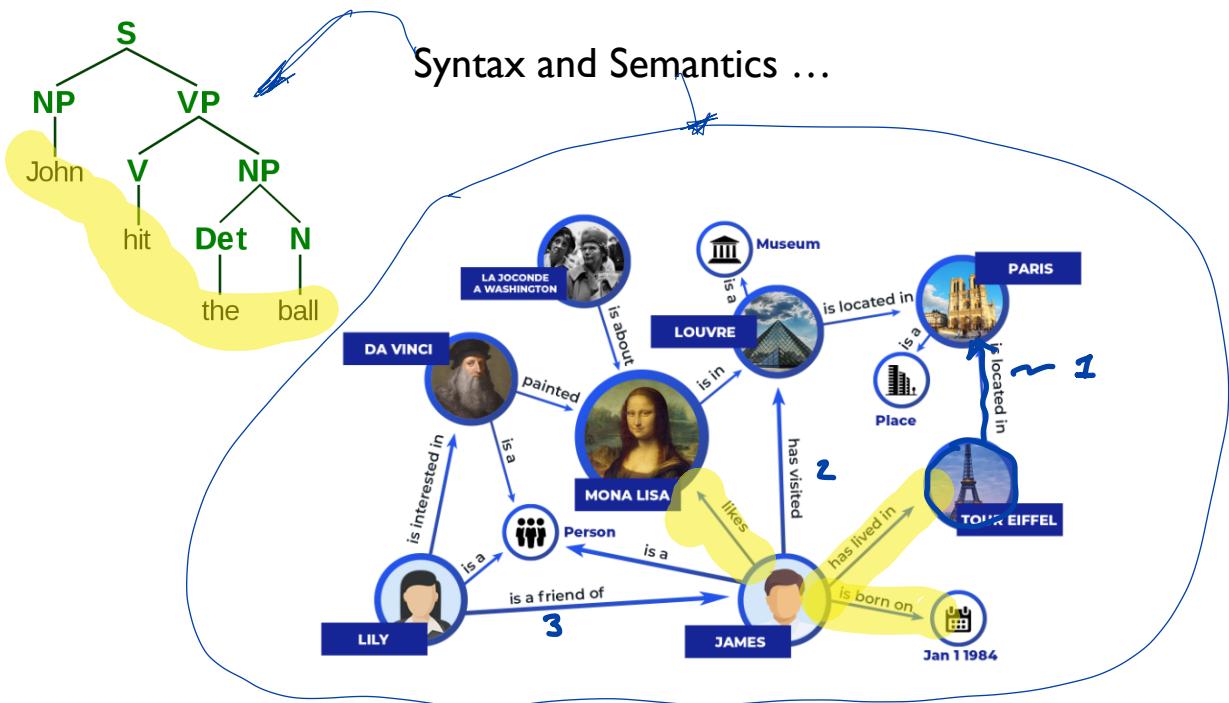


Scene Graphs

Johnson, Krishna, Stark, Li, Shamma, Bernstein, and Fei-Fei, "Image Retrieval with Scene Graphs", CVPR 2015



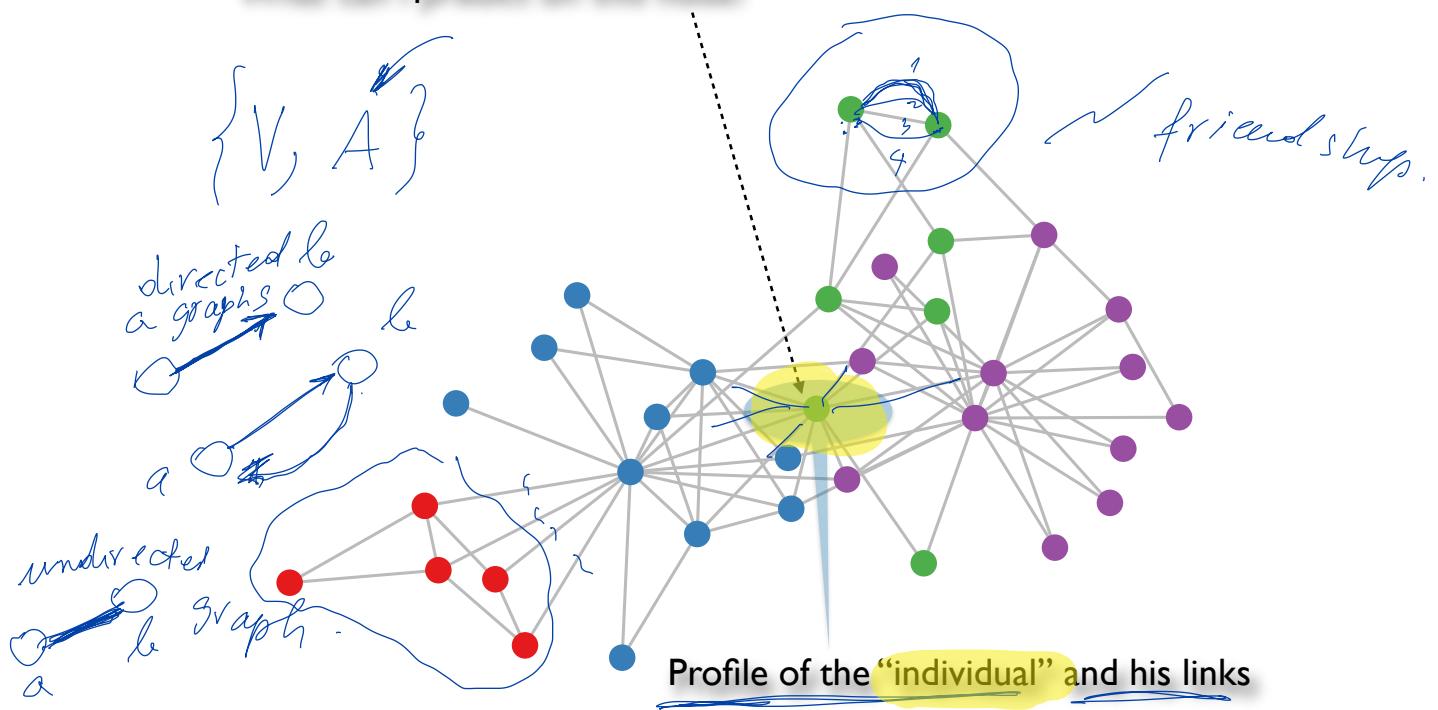
In Natural Language ...



SOCIAL NETWORKS

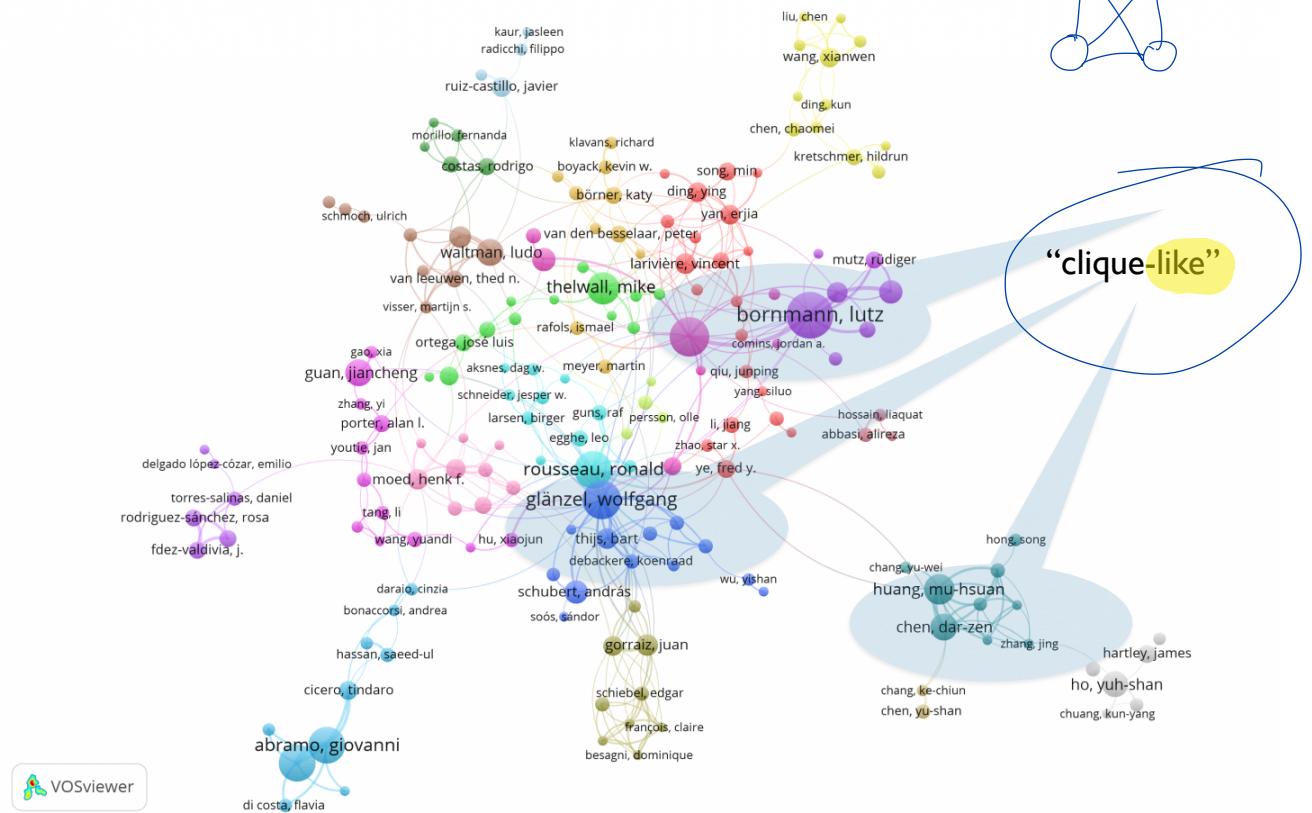
a unique graph!

Here we need to make prediction at node level!
What can I predict on this node?



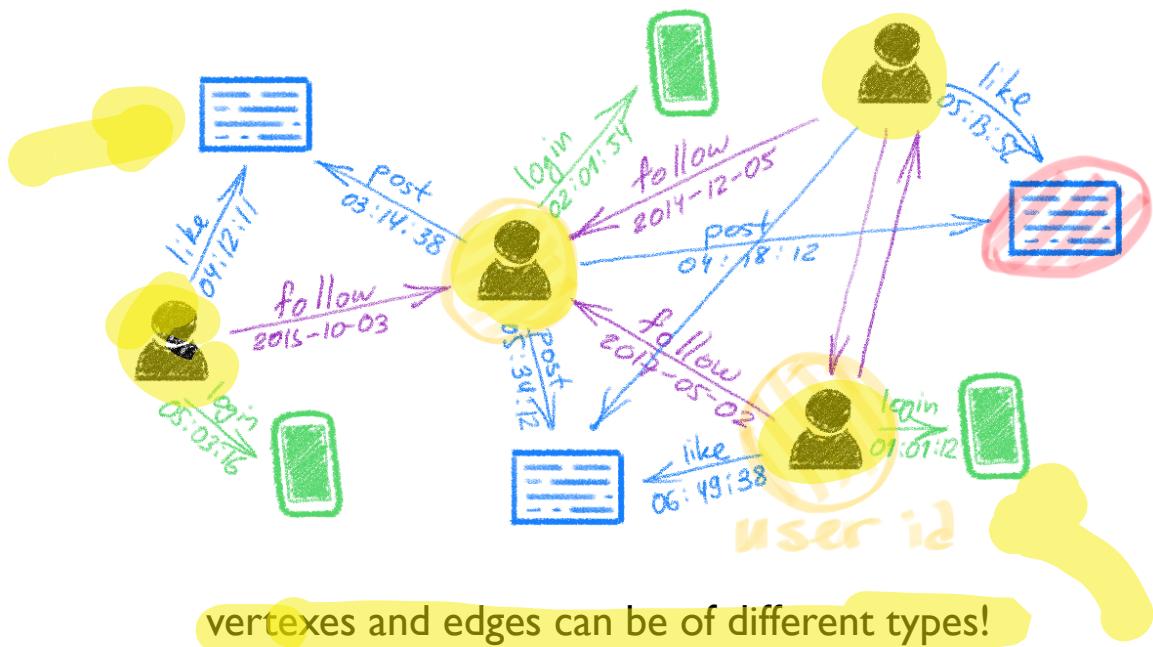
CITATION NETWORKS

by CitNetExplorer



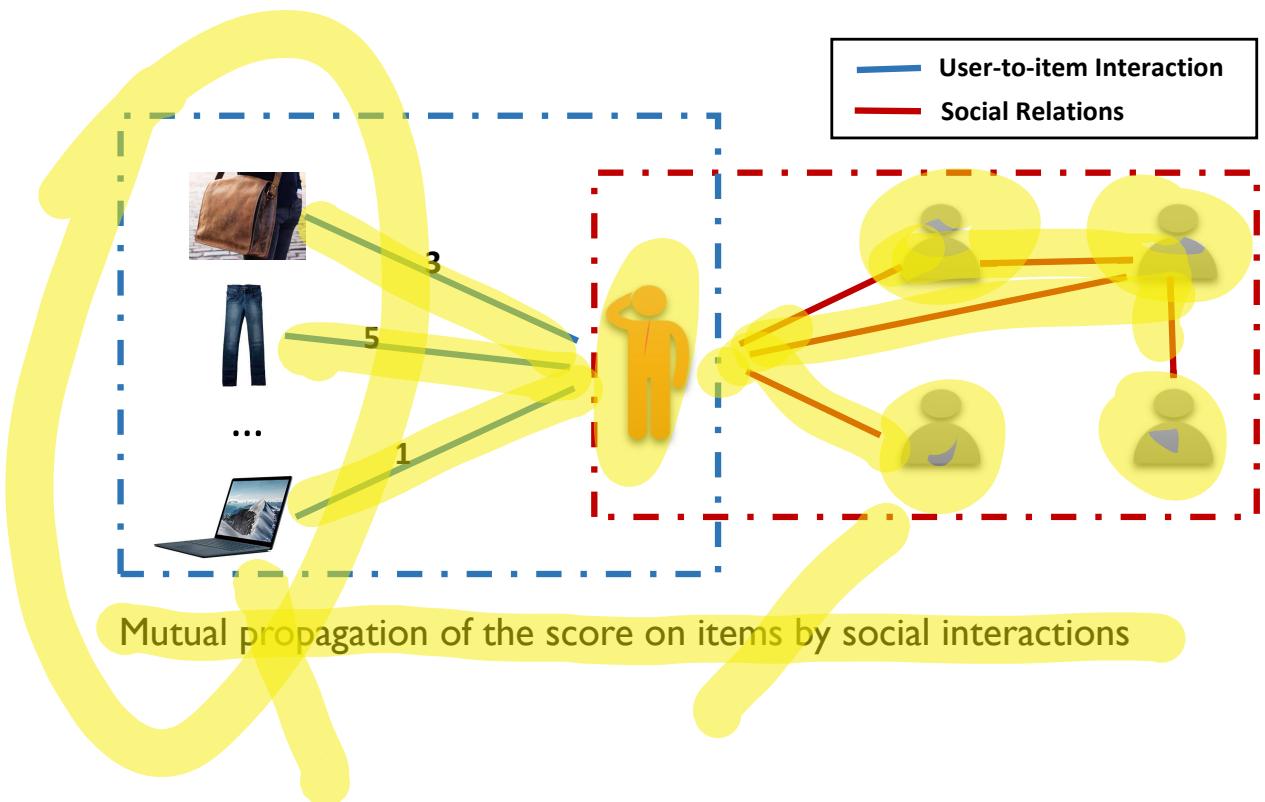
SOCIAL NETWORKS

Michael Bronstein - Twitter

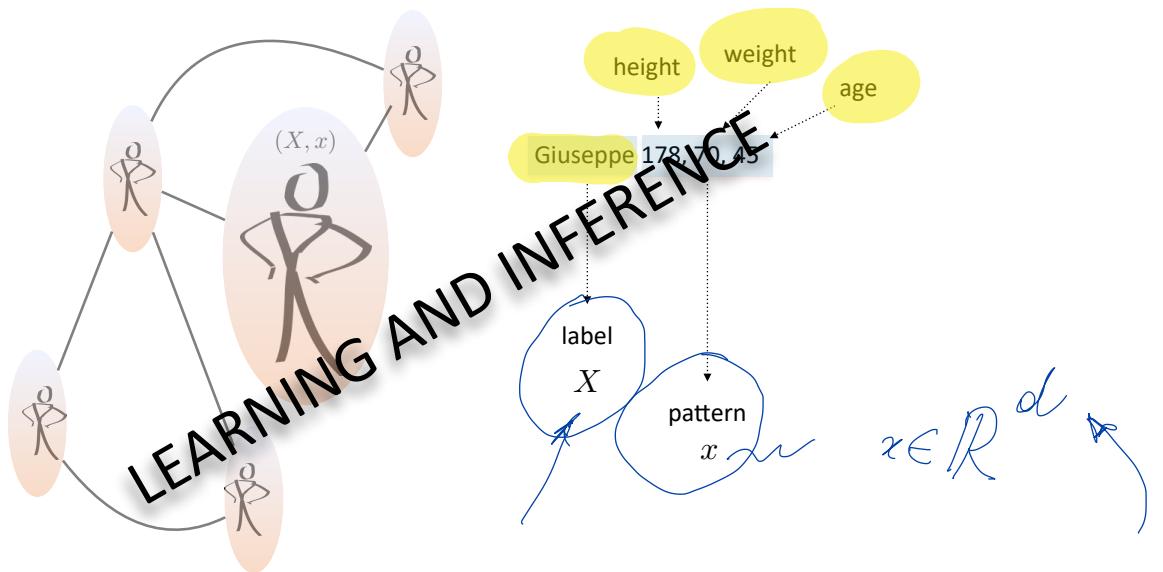


RECOMMENDATION SYSTEMS

W. Fan et al. 2019

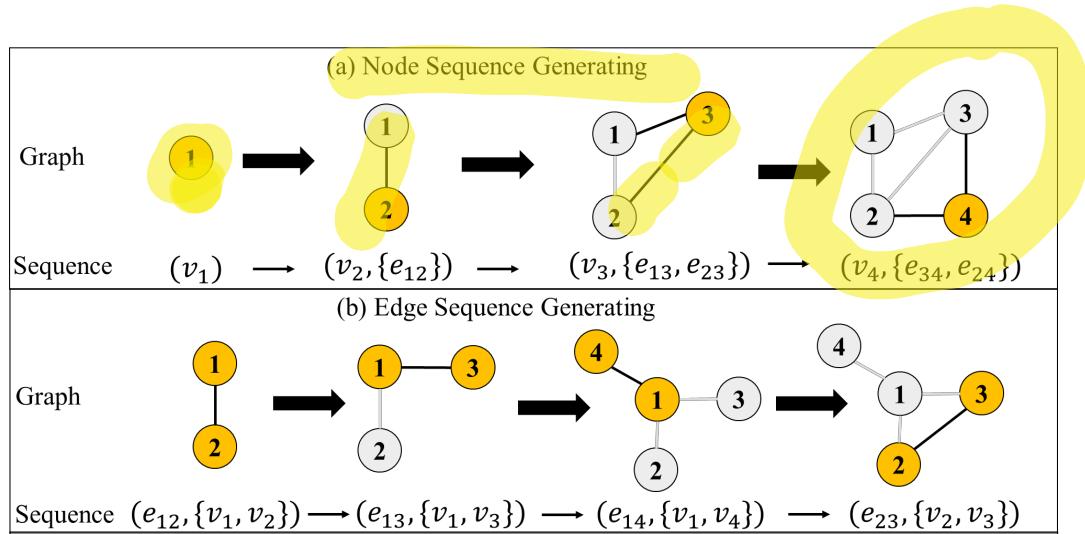


NET WITH SYMBOLS AND SUB-SYMBOLS



GRAPH GENERATION

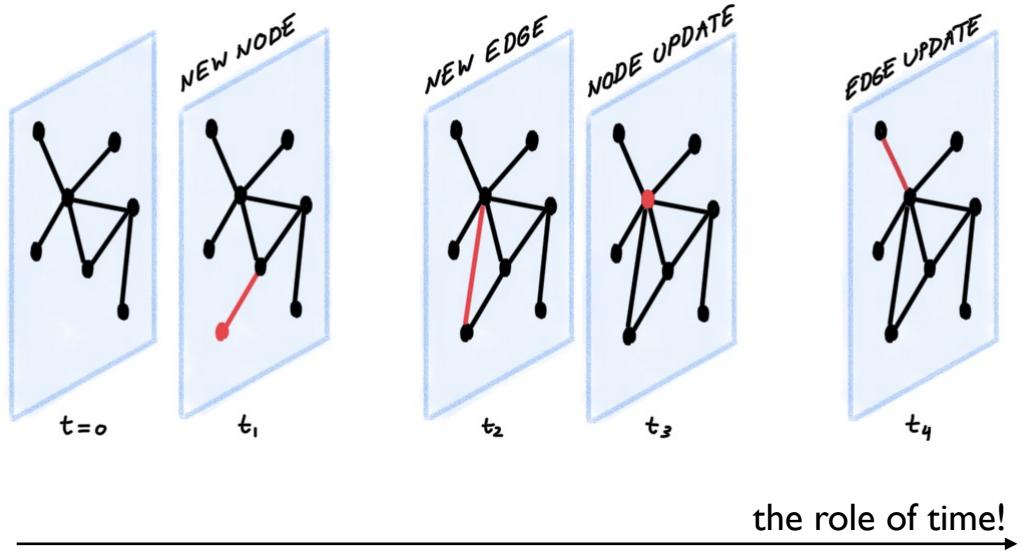
GHUO & ZHAO, 2020



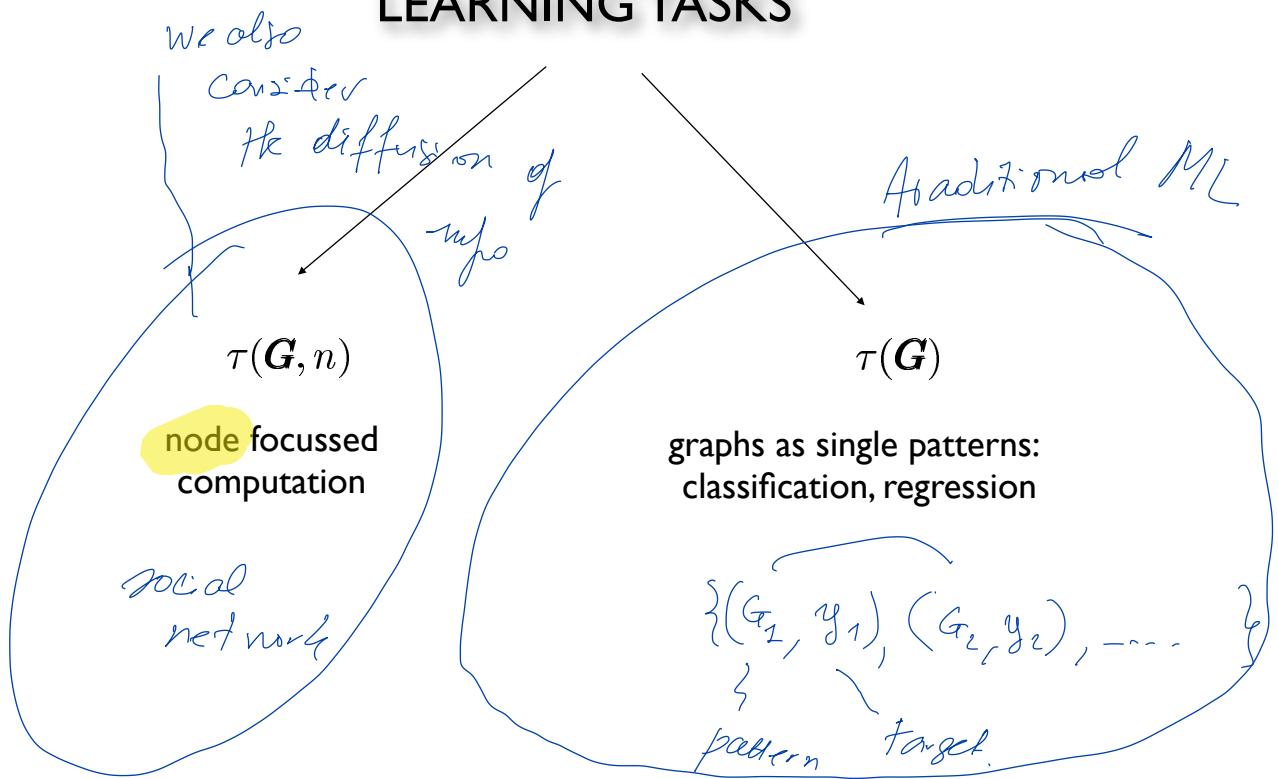
as new nodes/arcs come we need a generation scheme
to define the construction of the graph

GRAPH GENERATION

Emanuele Rossi, Imperial College



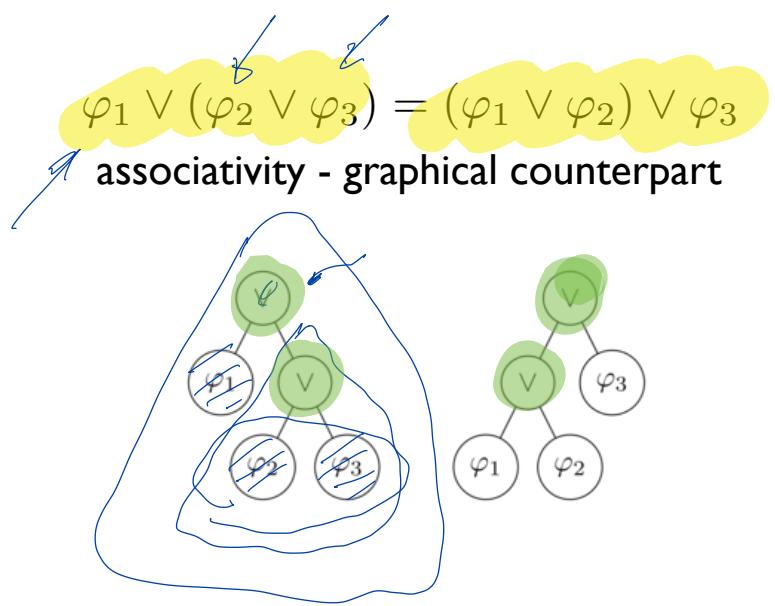
LEARNING TASKS



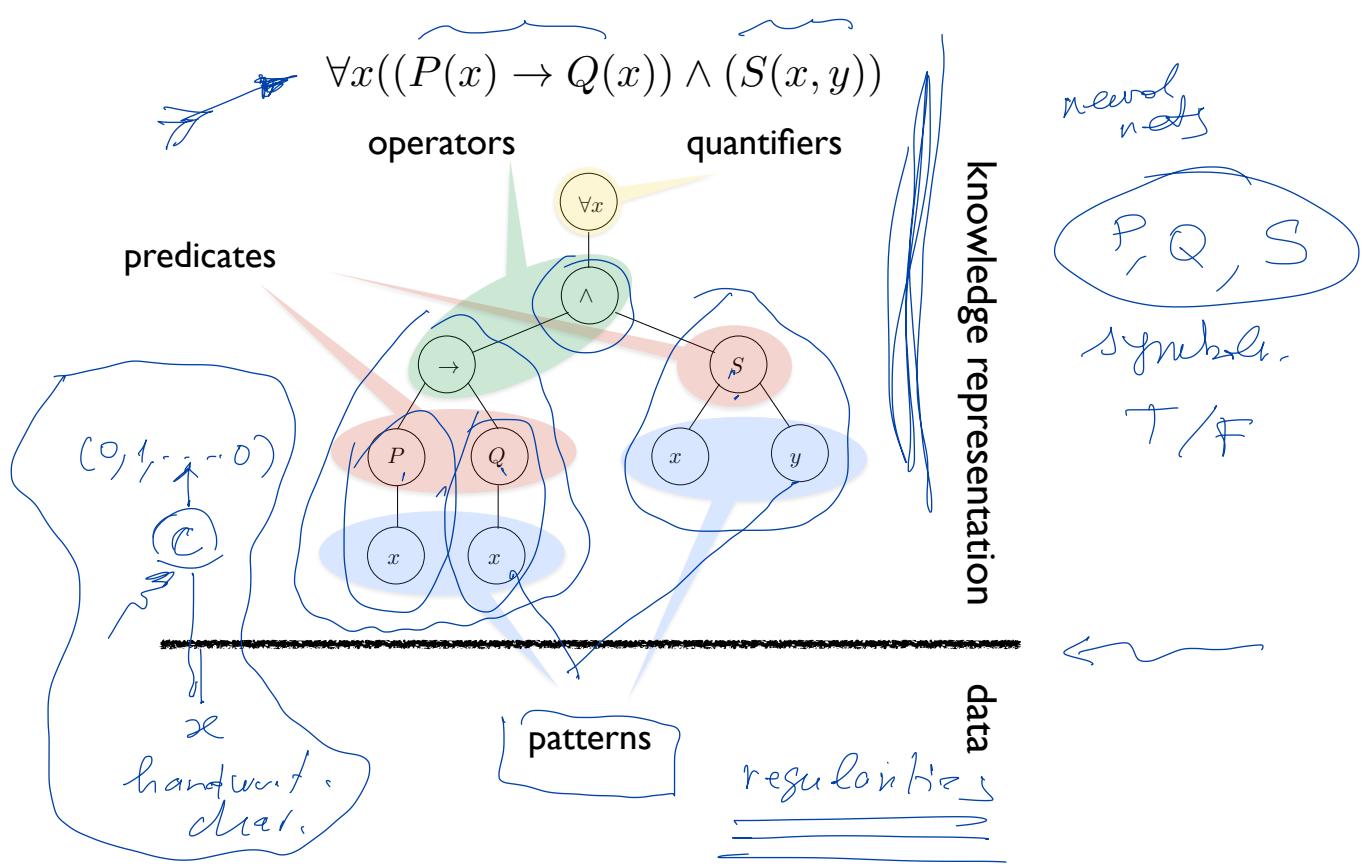
**WHAT IF WE START EXPRESSING
RELATIONS ON CATEGORIES?**

LOGIC STATEMENTS

logic formulas represented by parse trees



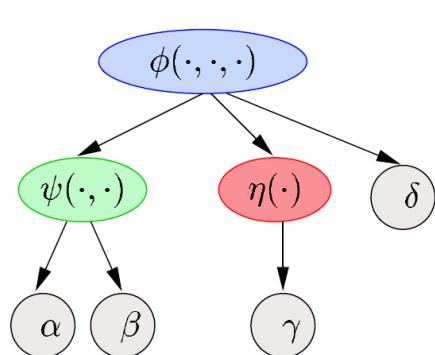
FOL LOGIC STATEMENTS



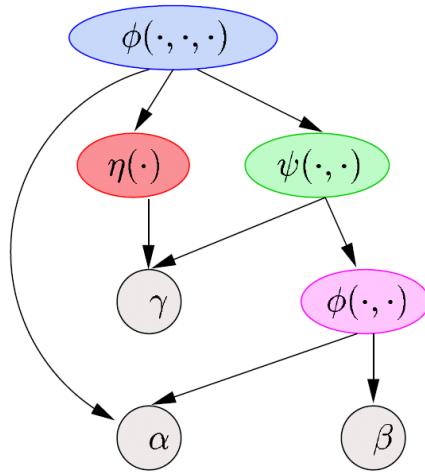
LOGIC STATEMENTS

... by Directed Acyclic Graphs

$\phi(\psi(\alpha, \beta), \eta(\gamma), \delta)$



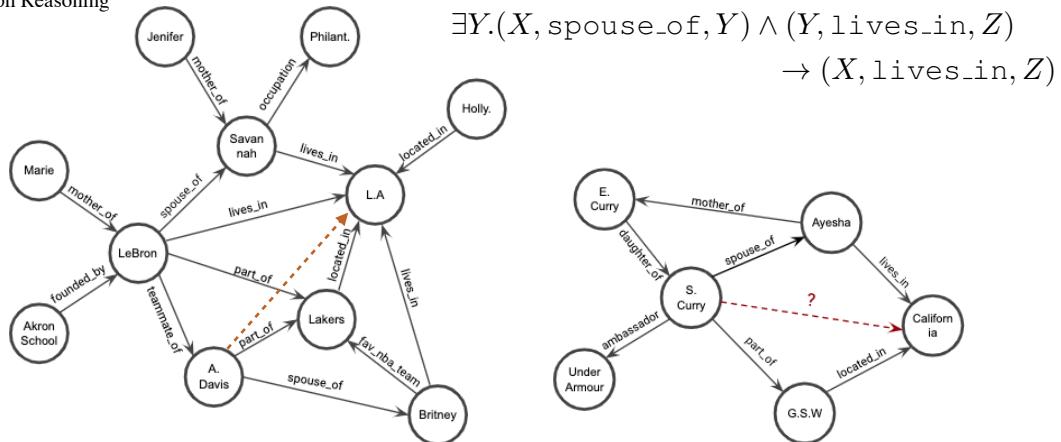
$\phi(\alpha, \eta(\gamma), \psi(\gamma, \phi(\alpha, \beta)))$



KNOWLEDGE GRAPHS

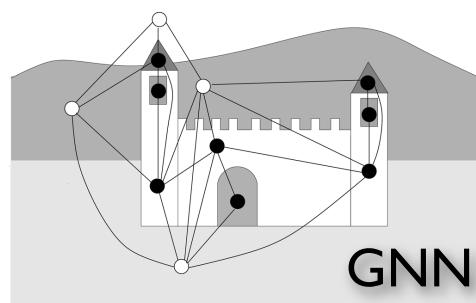
W. Hamilton et al (MILA)
ICML-2020

... Subgraph Reasoning



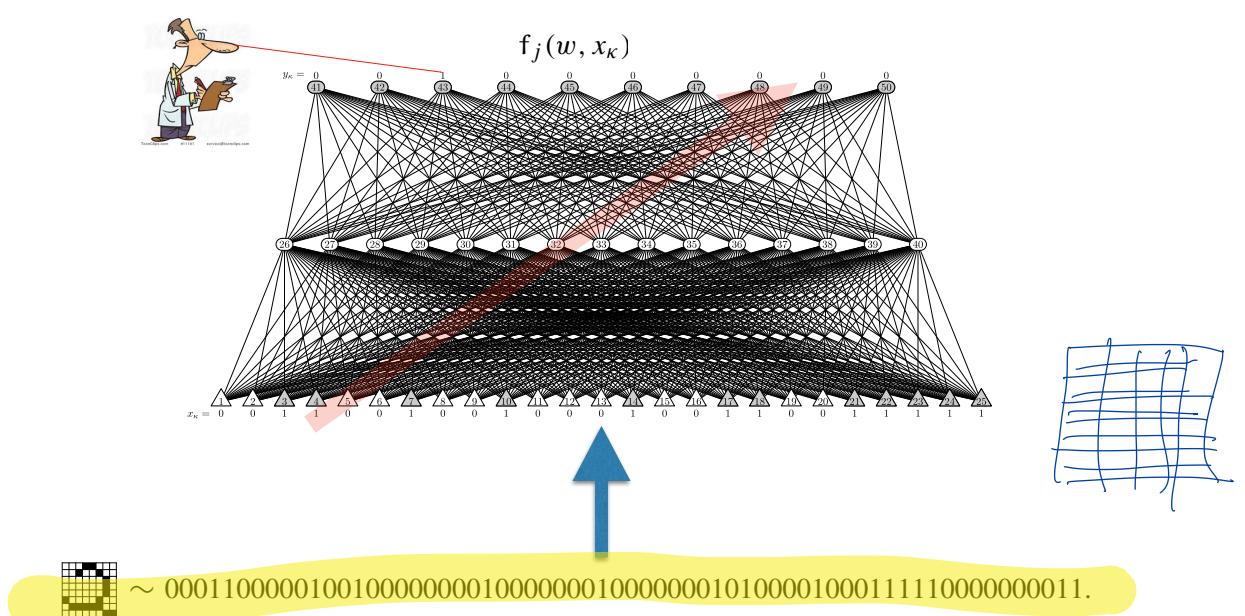
GRAPH NEURAL NETS

The viewpoint of diffusion



**HOW CAN A NEURAL NET (GRAPH)
PROCESS A GRAPH?**

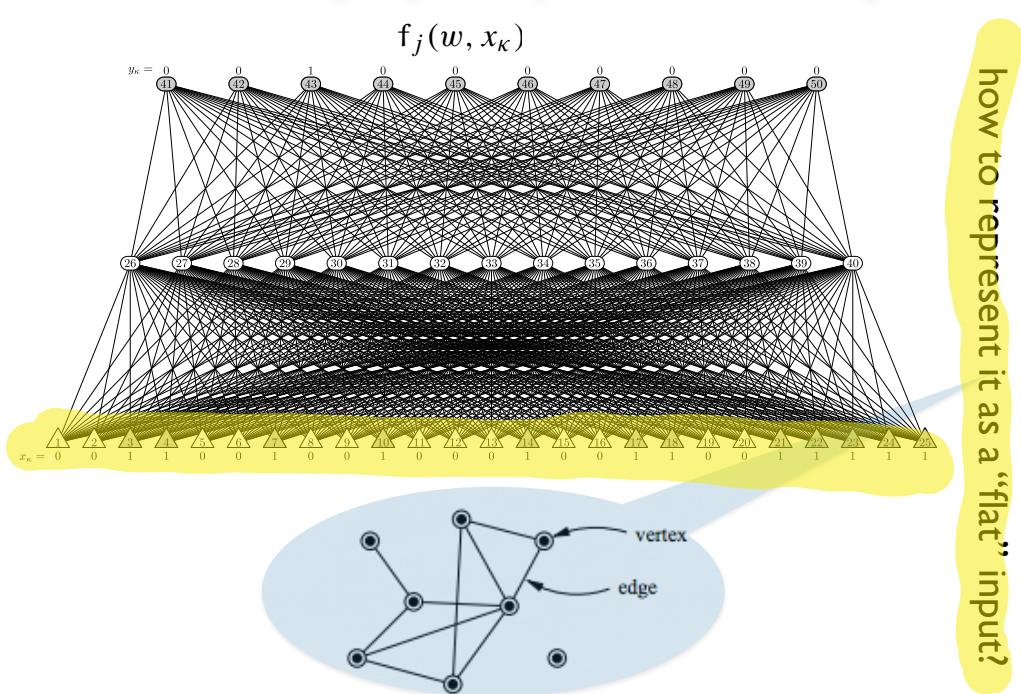
A Graph (Neural Network) processes
a vector ...



MSc DSAI 2022

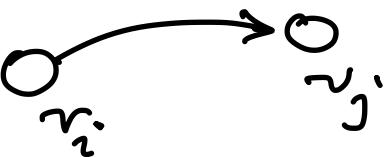
flat-based representation.

How Can Graph (ANN) Process a Graph?



$$G \sim \{V, A\};$$

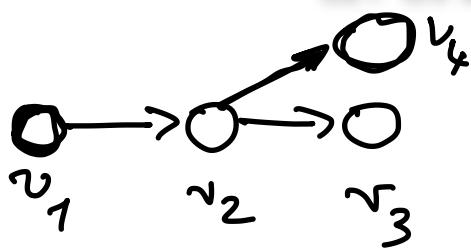
$$v_i < v_j$$



partial ordering relation

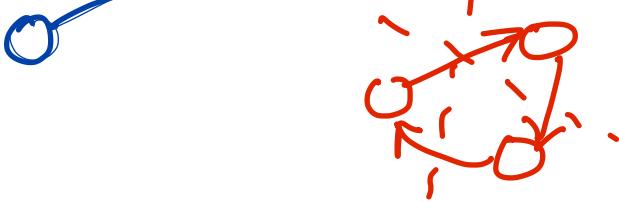
DIFFUSION-BASED PROCESSING

LET US BEGIN WITH DAGs ...



Directed Acyclic Graph.

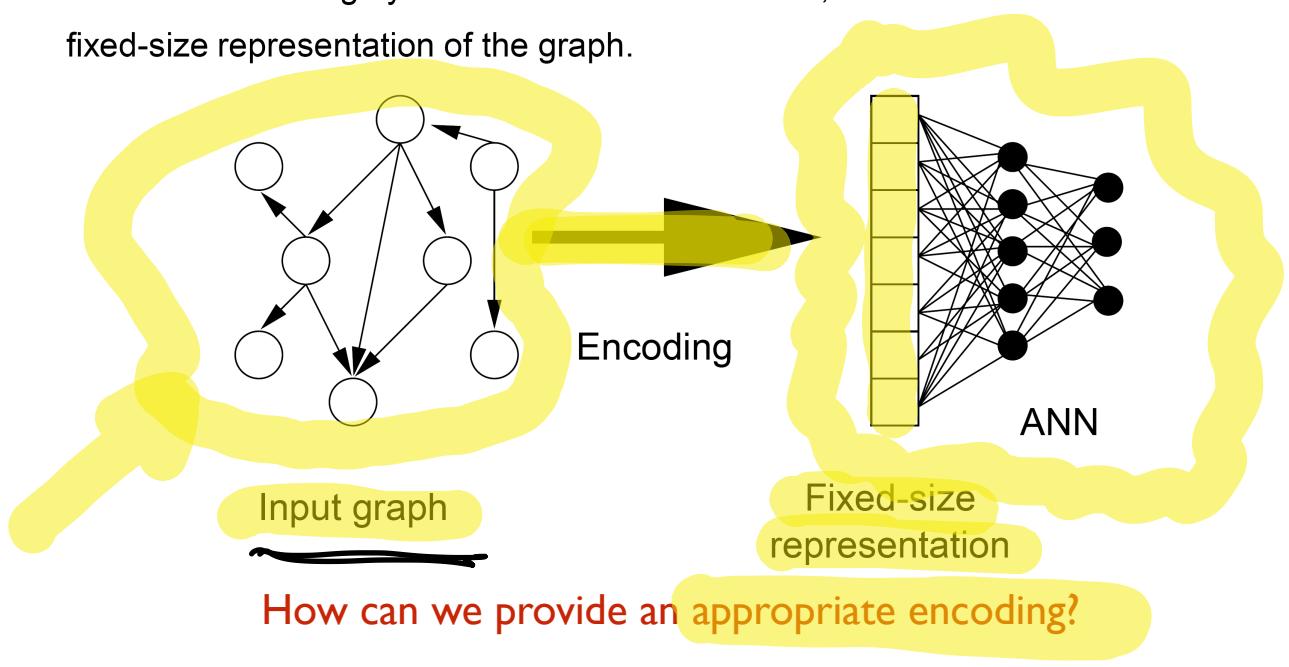
$$\begin{cases} v_1 < v_2 < v_3 \\ v_2 < v_4 \end{cases}$$



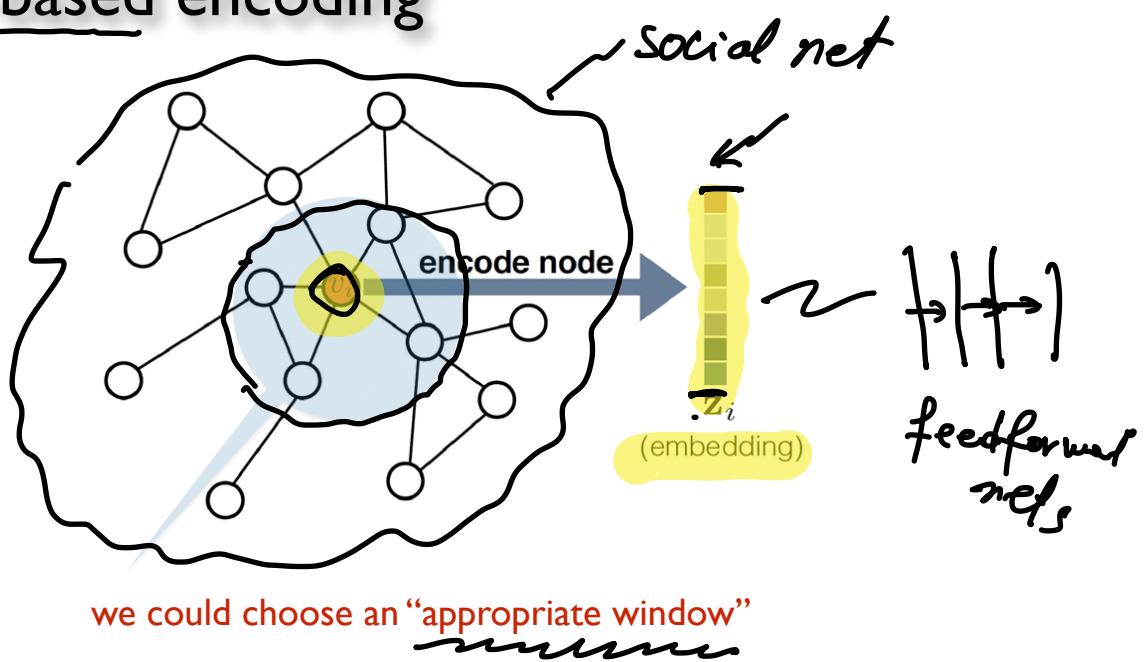
Adaptive vs fixed encoding

graph-focussed computation.

Instead of selecting by hand a fixed set of features, let a network to learn a fixed-size representation of the graph.



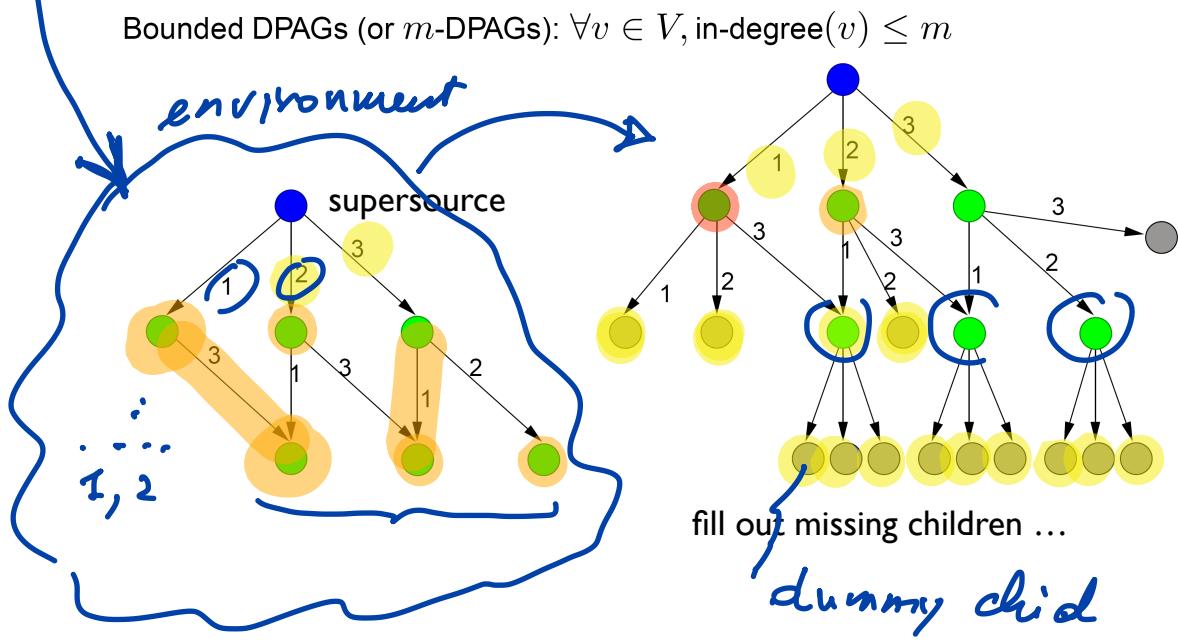
Node-based encoding



Directed Positional Acyclic Graphs

The class of DPAGs is formed by directed acyclic graphs such that, for each vertex v , a bijection $P : E \rightarrow \mathbb{N}$ is defined on the edges leaving from v .

Bounded DPAGs (or m -DPAGs): $\forall v \in V, \text{in-degree}(v) \leq m$



Graph Transductions

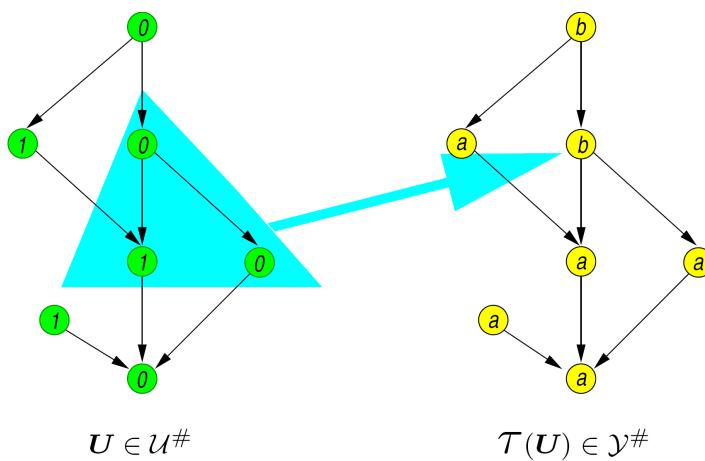
Let $\mathcal{U}^\#$ and $\mathcal{Y}^\#$ be two DOAG spaces.

A transduction \mathcal{T} is a subset of $\mathcal{U}^\# \times \mathcal{Y}^\#$.

Restrictions:

- \mathcal{T} is a function $\mathcal{T} : \mathcal{U}^\# \rightarrow \mathcal{Y}^\#$.
- \mathcal{T} is *IO-isomorph*, i.e. $\text{skel}(\mathcal{T}(\mathbf{U})) = \text{skel}(\mathbf{U})$.
- $\#$ is an ordered class of DAGs.

Choice of the “window” ...

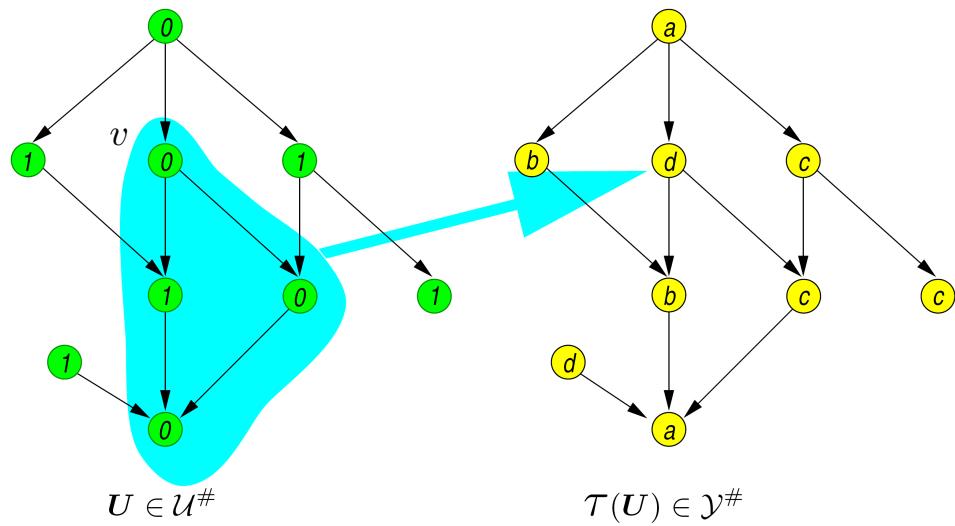


Compare to sequences: here we have to decide both:

- The number of nodes in the window.
- The *shape* of the window.

Information diffusion and causality

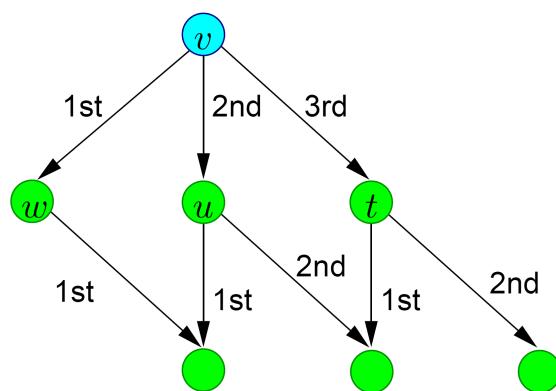
A transduction $\mathcal{T}(\cdot)$ is **causal** if $\forall v \in \text{vert}(\mathbf{U}) \quad \mathcal{T}(\mathbf{U})_v$ only depends on the subgraph of \mathbf{U} induced by $\{v\} \cup \text{de}[v]$.



Directed Ordered Acyclic Graphs

The class of DOAGs is formed by directed acyclic graphs such that, for each vertex v , a total order \prec is defined on the edges leaving from v .

E.g.: $(v, w) \prec (v, u) \prec (v, t)$



ordering of children state does matter!

State-based representation

Given an input graph \mathcal{U} , for each vertex v :

$$\mathbf{X}_v = f(\mathbf{X}_{\text{ch}[v]}, \mathcal{U}_v)$$

$$\mathbf{Y}_v = g(\mathbf{X}_v, \mathcal{U}_v)$$

where $\text{ch}[v]$ are the (ordered) children of v and

$f : \mathcal{X}^m \times \mathcal{U} \rightarrow \mathcal{X}$ state transition function

$g : \mathcal{X} \times \mathcal{U} \rightarrow \mathcal{Y}$ output function

Compare to temporal dynamical systems:

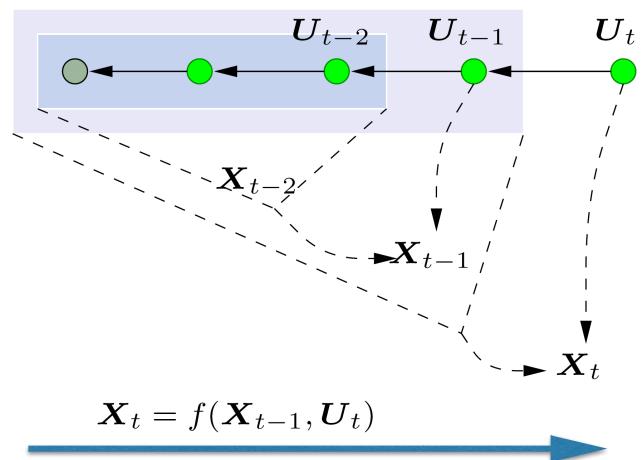
$$\mathbf{X}_t = f(\mathbf{X}_{t-1}, \mathcal{U}_t)$$

$$\mathbf{Y}_t = g(\mathbf{X}_t, \mathcal{U}_t)$$

a recursive state representation exists only if $\mathcal{T}(\cdot)$ is causal

$\mathcal{T}(\cdot)$ is stationary: $f(\cdot)$ and $g(\cdot)$ do not depend on v

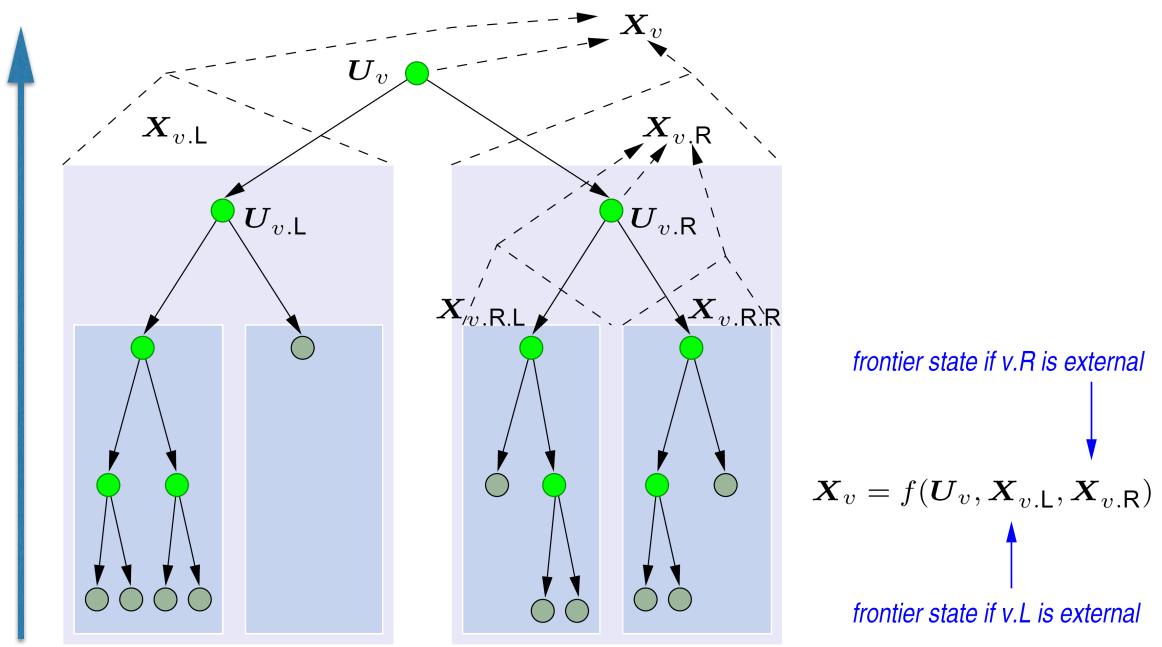
Reduction to sequences



NB: for $t = 0$, $\mathbf{X} = \mathbf{X}_0$ is the **initial state**

The initial state is associated with the external vertex (frontier)

The case of binary trees ...



Supersource transductions

- Assumption: The input graph \mathcal{U} is an m DOAG and has a supersource s .
- The output is a single label \mathbf{Y} .
 - Classification: a categorical variable.
 - Scalar regression: a (multivariate) numerical variable.
- Usual state updating scheme:

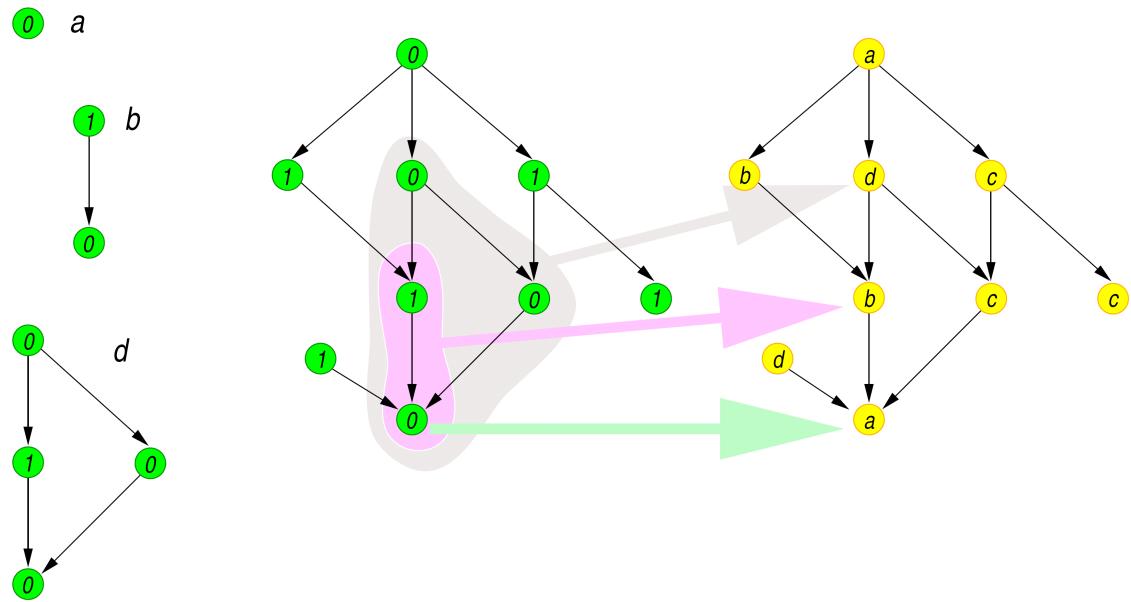
$$\mathbf{X}_v = f(\mathbf{X}_{\text{ch}[v]}, \mathcal{U}_v)$$

- The output label is “emitted” at s :

$$\mathbf{Y} = g(\mathbf{X}_s)$$

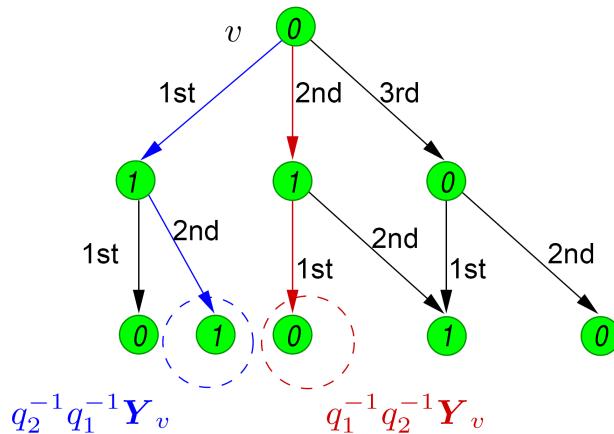
IO-isomorph transductions

An IO-isomorph transduction is like many supersource transductions:



Generalized shift-operator

- Sequences: $q^{-1}\mathbf{Y}_t = \mathbf{Y}_{t-1}$ (unitary time delay).
- DOAGs: $q_k^{-1}\mathbf{Y}_v$ is the label attached to the k -th child of vertex v .
NB: $q_k^{-1}\mathbf{Y}_v = \emptyset$ if the k -th child of v belongs to the frontier.
- Composition is not commutative:

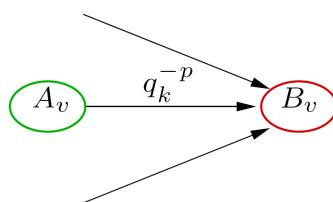


Recursive networks

Hypotheses: $\left\{ \begin{array}{l} \bullet \quad \mathcal{T}(\cdot) \text{ admits a recursive state space representation;} \\ \bullet \quad \text{input, state, and output are uniformly labeled.} \end{array} \right.$

The recursive network $\mathcal{T}(\cdot)$ is a directed graph where:

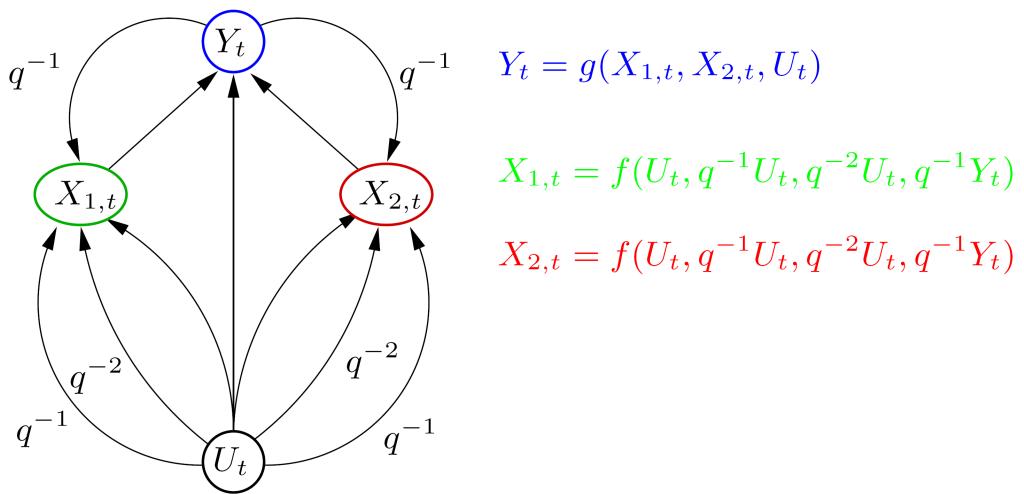
- vertices are marked with representative variables;
- edges are marked with generalized shift operators;
- An edge (A_v, B_v) , with label q_k^{-p} , means that for each v in the vertex set of the input structure B_v is a function of $q_k^{-p} A_v$.



\

NARX nets

The Nonlinear AutoRegressive network with eXogenous inputs

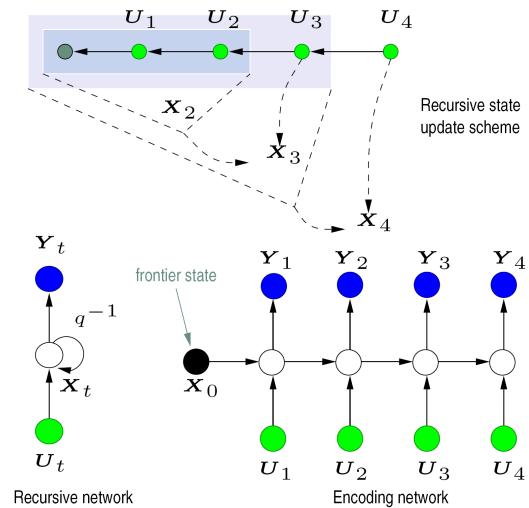


Encoding networks

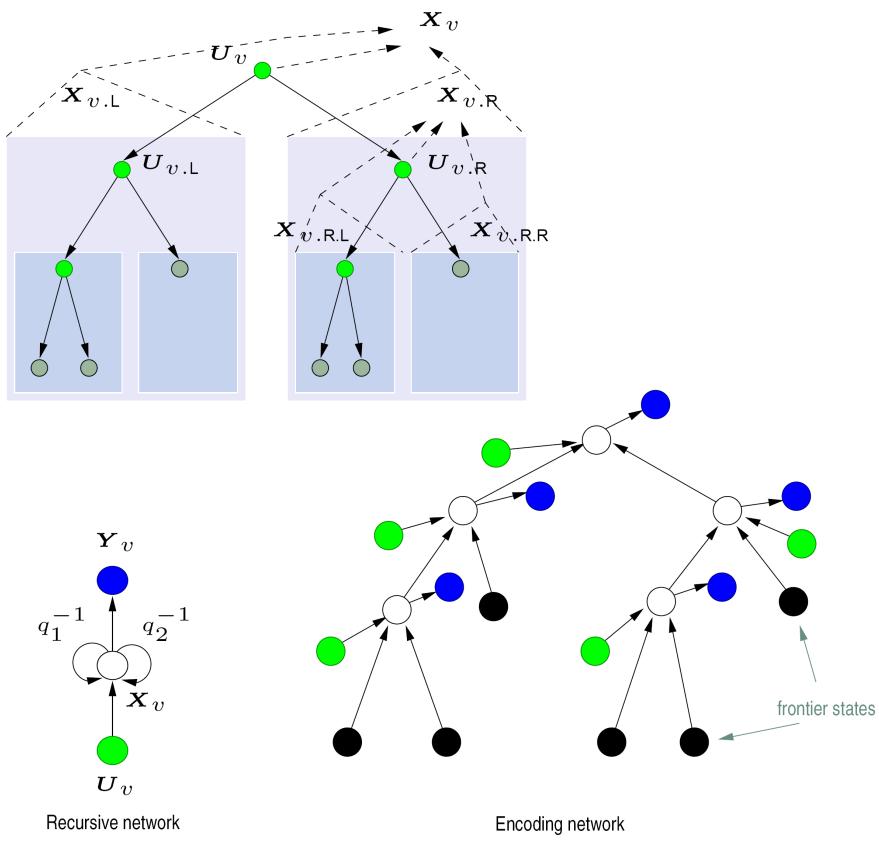
Given a graph $\mathbf{U} \in \mathcal{U}^\#$ and a recursive transduction \mathcal{T} .

The *encoding network* associated to \mathbf{U} and \mathcal{T} is formed by unrolling the recursive network of \mathcal{T} through the input graph \mathbf{U} .

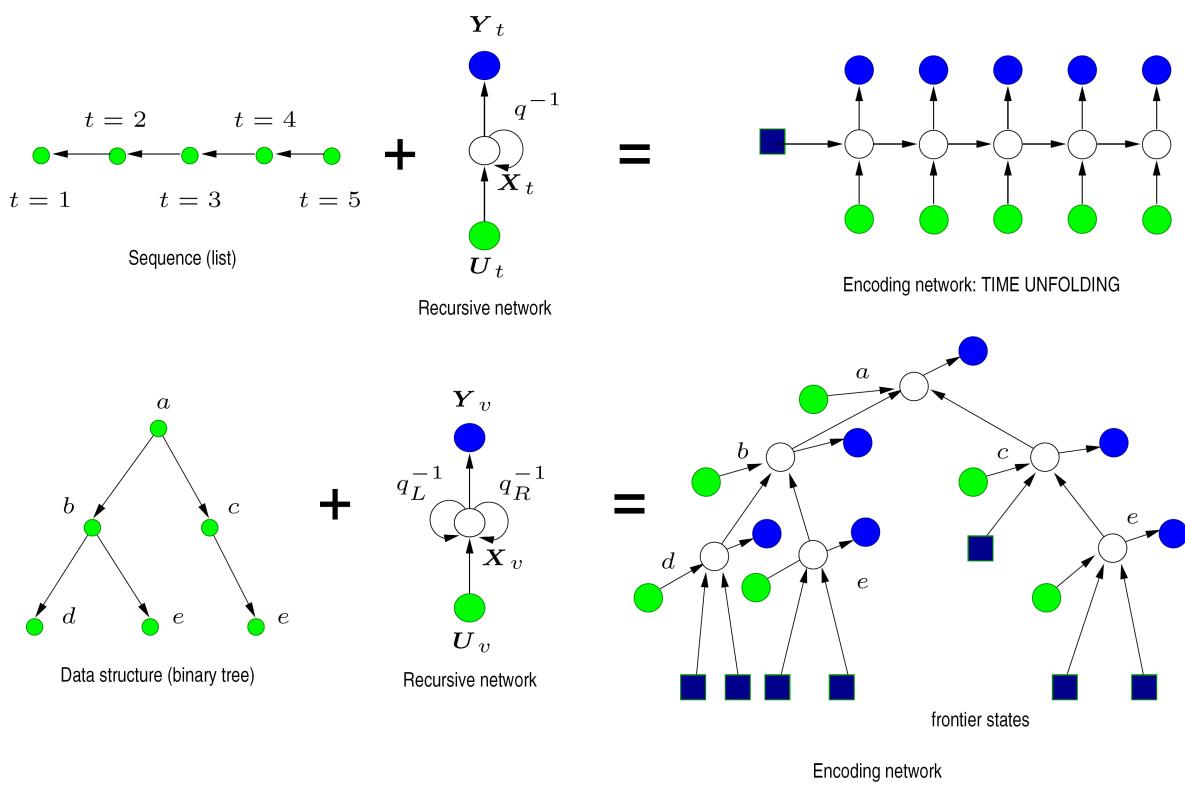
Special case (*time-unfolding*): $\#$ is the class of sequences:



Encoding nets for binary trees

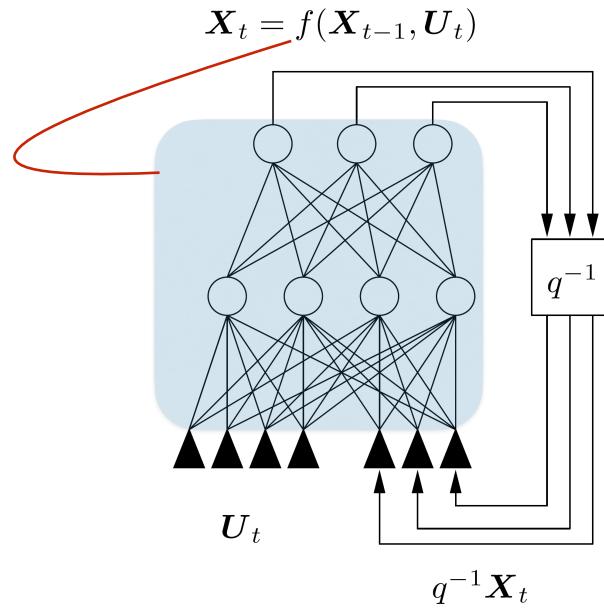


Data structures + recursive nets = encoding nets

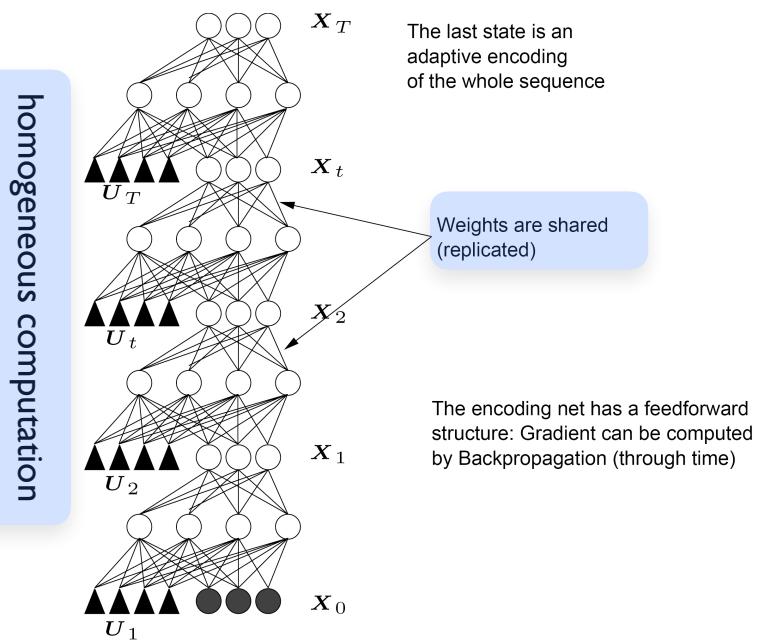


Graphs for processing Graphs: Using neural nets for mapping

The state transition function is implemented by a MLP:



Time unfolding

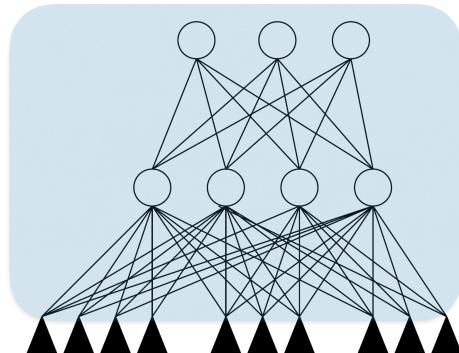


Using neural nets for binary trees ...

State labels are real vectors: $\mathbf{X}_v \in I\!\!R^n$.

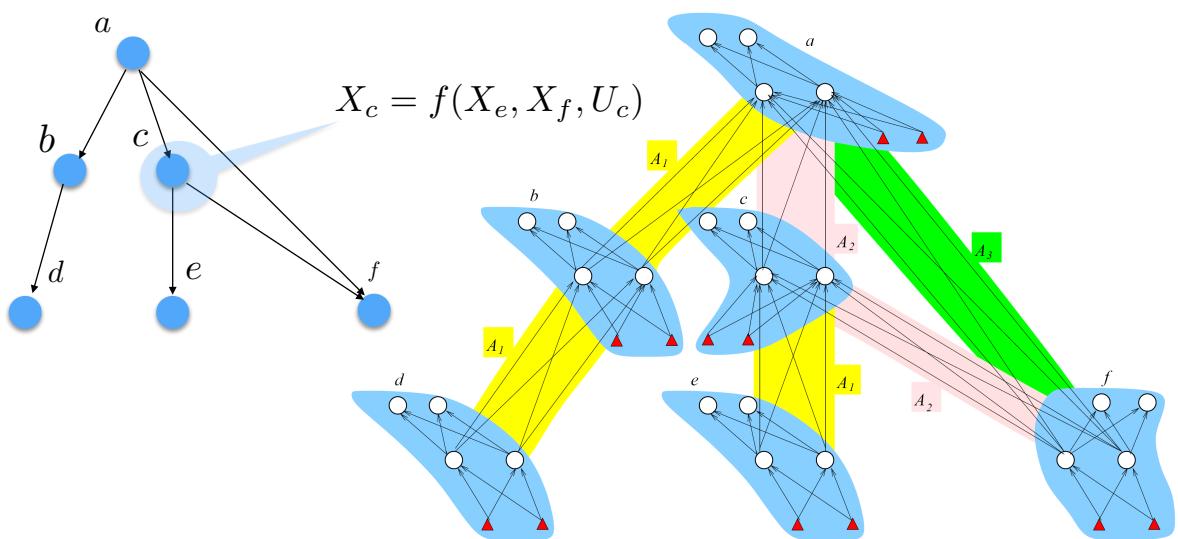
The state transition function is implemented by a MLP (e.g. case of binary trees)

$$\mathbf{X}_v = f(\mathbf{X}_{\text{ch}[v]}, \mathbf{U}_v) = f(q_l^{-1} \mathbf{X}_v, q_r^{-1} \mathbf{X}_v, \mathbf{U}_v)$$



$$\mathbf{U}_v \quad q_l^{-1} \mathbf{X}_v \quad q_r^{-1} \mathbf{X}_v$$

Structure (graph) unfolding



From the encoding network to the encoding neural network ...

... just like in recurrent neural networks for sequences

Backpropagation Through Structure

Algorithm 1 BPTS

Input:

The graph \mathbf{U} ;
A recursive neural network \mathbf{N} .

Output:

The gradient $\nabla_{\Theta} \ell_U(\Theta)$

begin

 Initialize(Θ);

 Encoding-Neural-Network(\mathbf{U}, \mathbf{N});

 Backpropagation(\mathbf{N});

 Average(Θ). ← Weight sharing ...

end

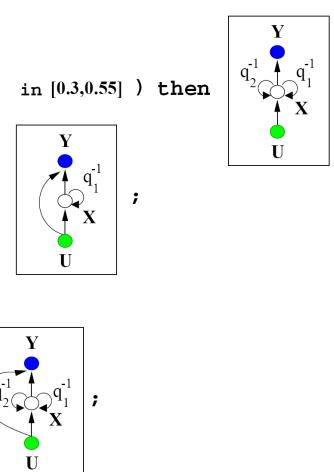
BE CAREFUL: WEIGHT SHARING WHEN TYPES ARE UNIFORM!

Non-stationary transductions

Linguistic specification of the recursive network

knowledge-based map description

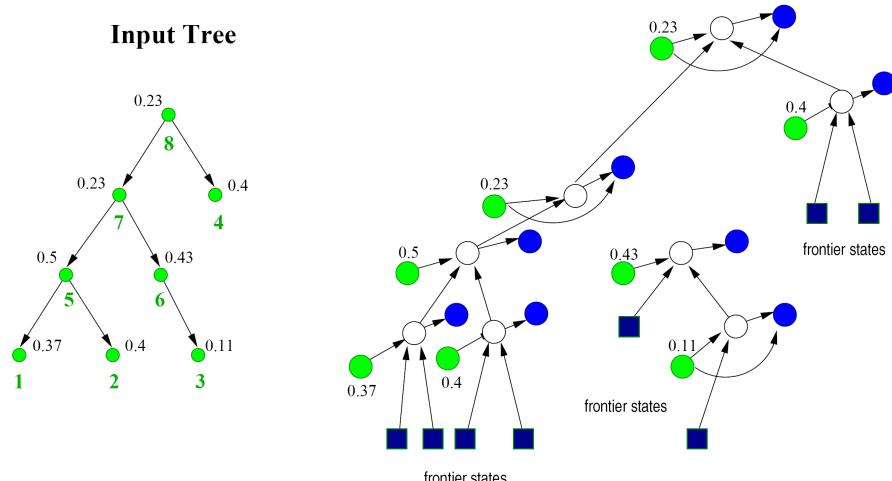
```
Sequence_of_vertices Seq;
Vertex v;
Seq <- sort_vertices_by( dist_from(frontier), < );
foreach(v, Seq) {
    if (dist_from(frontier)<3) then {
        if (U in [0.3,0.55] ) then
            else
        }
    }
}
```



Compiling ...

The input tree is mapped to one with different structure!

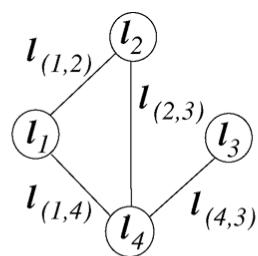
From the previous linguistic specification the encoding network is compiled. Finally, in the last step the encoding neural network is created.



Encoding Network

What if DOAG assumption is lost?

It's the general case which originated the term GNN!

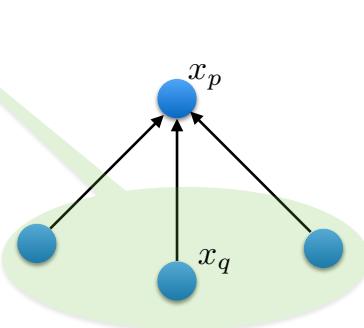


When ordering is lost, the previous data flow computational scheme cannot be established:

We need a different diffusion process!

PageRank: A Related Diffusion Issue

$$x_p = d \sum_{q \in pa[p]} \frac{x_q}{h_q} + (1 - d)$$

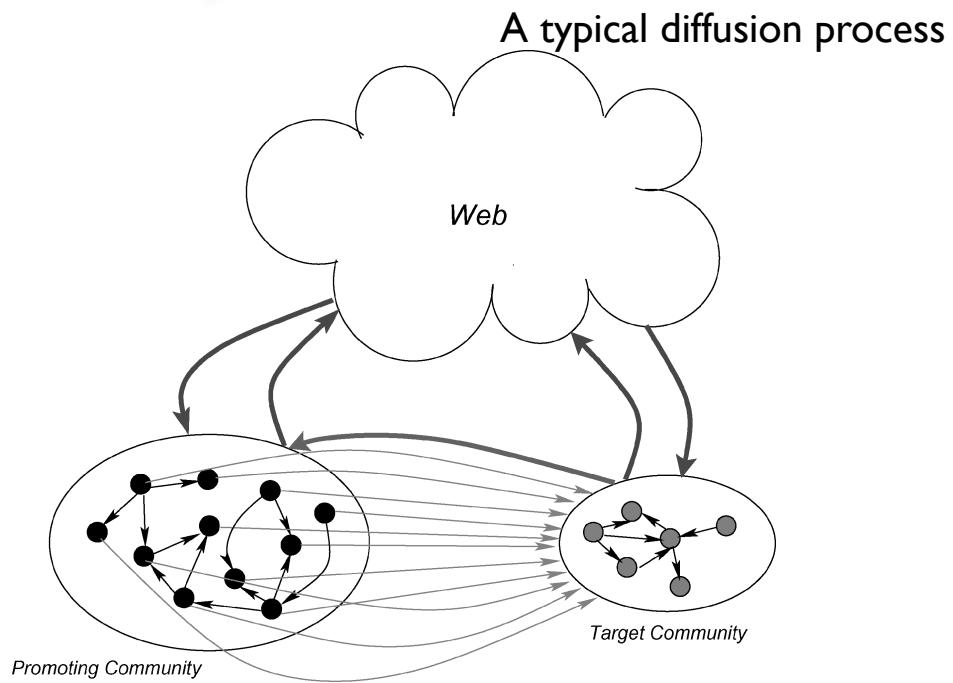


over all the graph

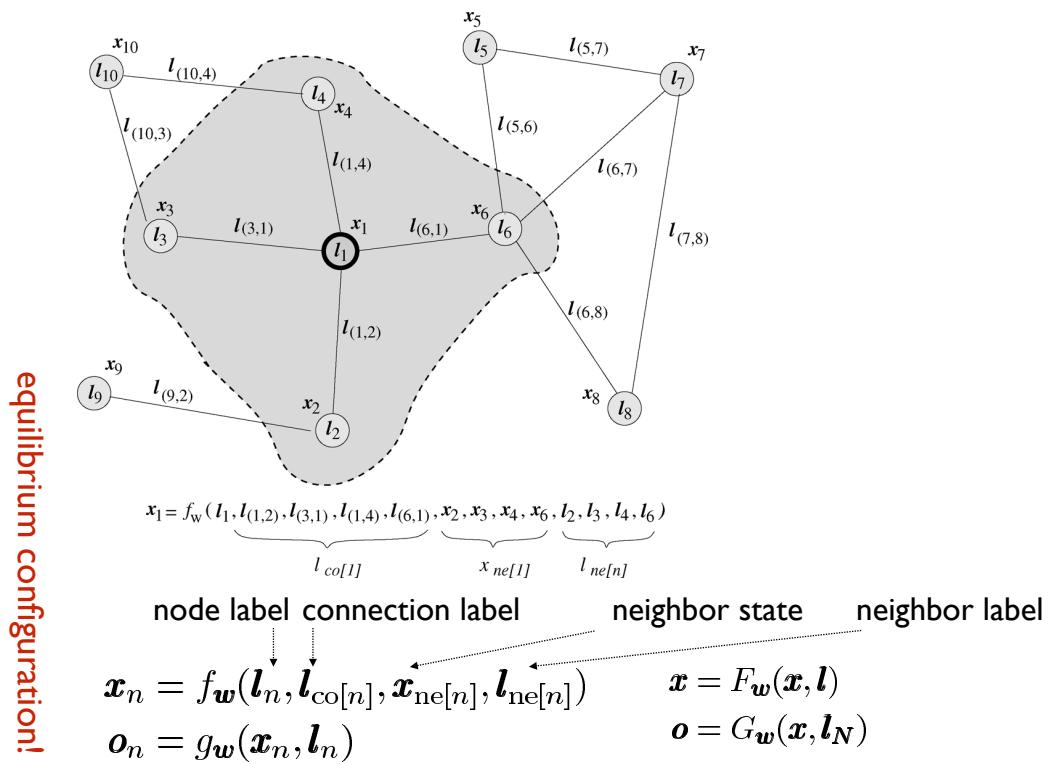
$$\mathbf{x} = d \mathbf{Wx} + (1 - d) \mathbf{1}\mathbf{1}_N$$

PageRank:

The popular “Web-spam” issue!

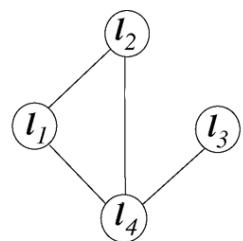


Neighbor-based computation



Non-positional graphs

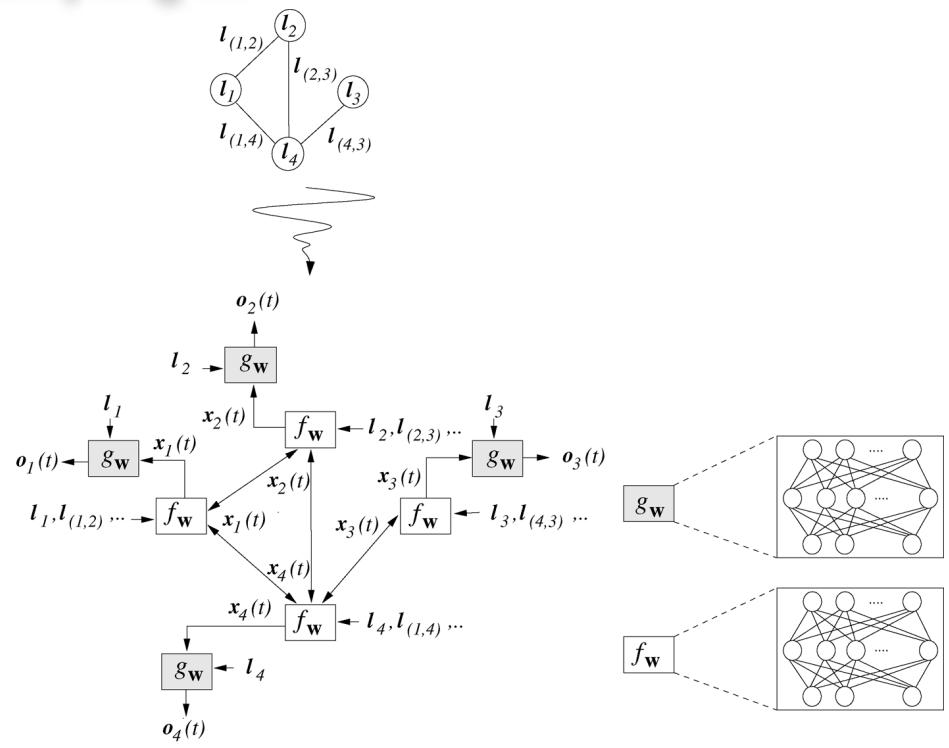
diffusion-based computation similar to PageRank



$$\mathbf{x}_n = \sum_{u \in \text{ne}[n]} h_{\mathbf{w}}(\mathbf{l}_n, \mathbf{l}_{(n,u)}, \mathbf{x}_u, \mathbf{l}_u), \quad n \in N$$

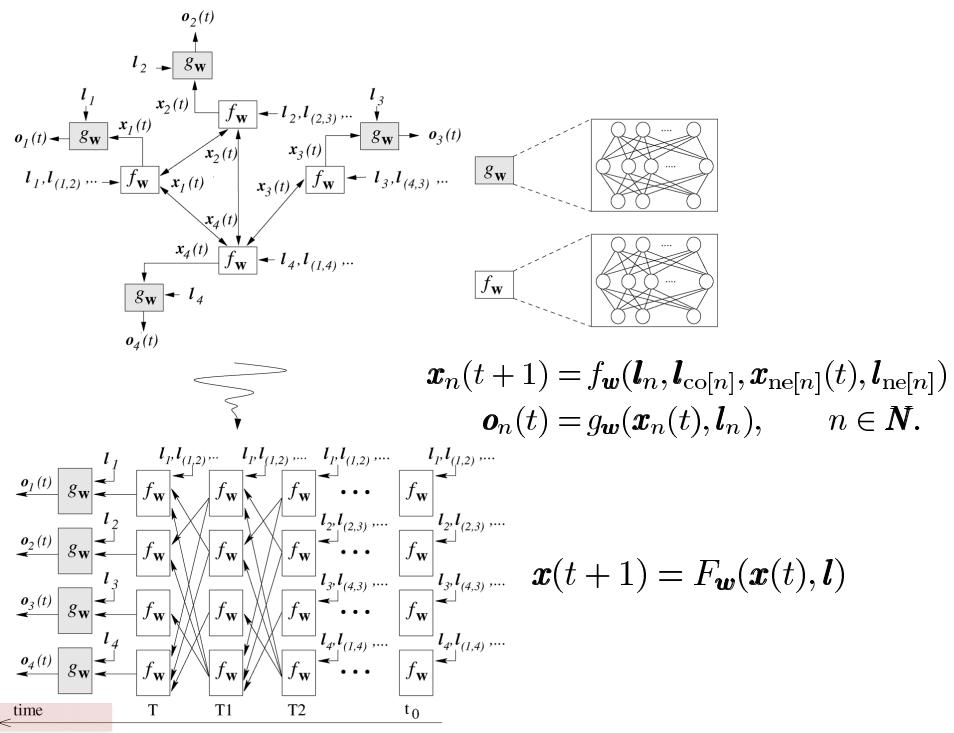
← permutation-independent

Graph compiling ...

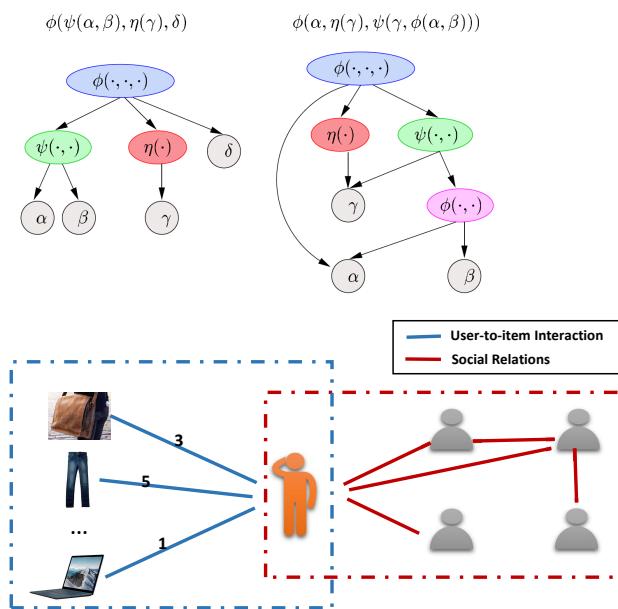


Relaxation to an equilibrium

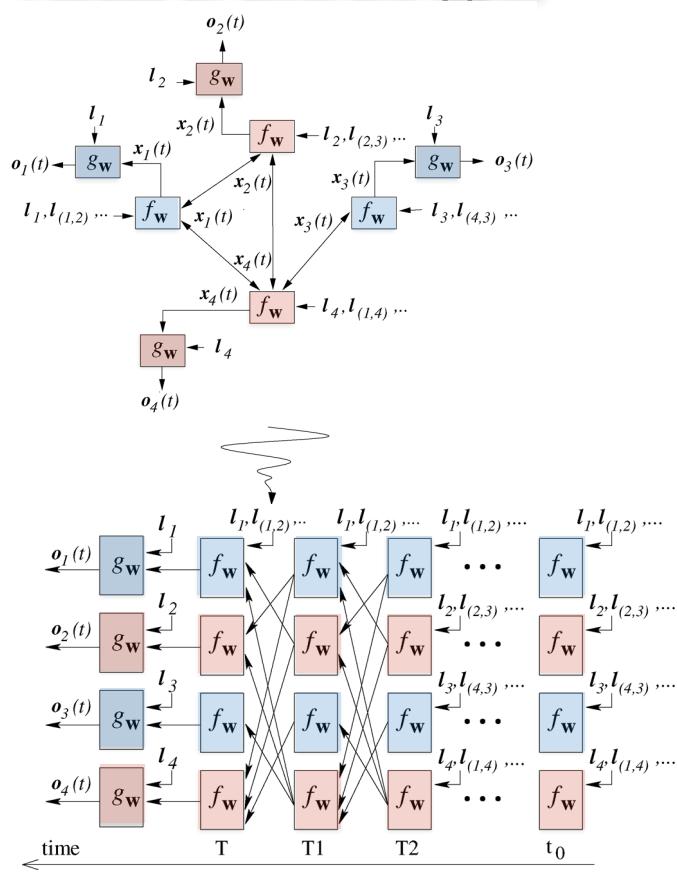
How we get the equilibrium points?



WHAT IF NODES ARE OF DIFFERENT TYPE?



Nodes of different types



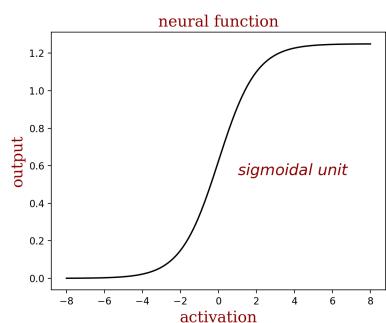
How we get the equilibrium?

Don't forget we are in front of non-linear dynamics!

$$\mathbf{x}_n(t+1) = f_{\mathbf{w}}(\mathbf{l}_n, \mathbf{l}_{co[n]}, \mathbf{x}_{ne[n]}(t), \mathbf{l}_{ne[n]})$$

$$\mathbf{o}_n(t) = g_{\mathbf{w}}(\mathbf{x}_n(t), \mathbf{l}_n), \quad n \in \mathbf{N}.$$

$$\mathbf{x}(t+1) = F_{\mathbf{w}}(\mathbf{x}(t), \mathbf{l})$$



When do we reach an equilibrium?

$$\|\mathbf{F}_{\mathbf{w}}(\mathbf{x}, \mathbf{l}) - \mathbf{F}_{\mathbf{w}}(\mathbf{y}, \mathbf{l})\| \leq \mu \|\mathbf{x} - \mathbf{y}\|, \quad 0 \leq \mu < 1$$

Contraction map!

Supervised Learning

$$\mathcal{L} = \{(\mathbf{G}_i, n_{i,j}, \mathbf{t}_{i,j}) |, \mathbf{G}_i = (\mathbf{N}_i, \mathbf{E}_i) \in \mathcal{G}; \text{ no of graphs } \\ n_{i,j} \in \mathbf{N}_i; \mathbf{t}_{i,j} \in I\!\!R^m, 1 \leq i \leq p, 1 \leq j \leq q_i\}$$

no supervision is required on all the vertexes!

Can we learn by classic gradient descent?

$$\varphi_{\mathbf{w}} \text{ differentiability}$$

$\partial_{\mathbf{w}} e_{\mathbf{w}}$

$$e_{\mathbf{w}} = \sum_{i=1}^p \sum_{j=1}^{q_i} (\mathbf{t}_{i,j} - \varphi_{\mathbf{w}}(\mathbf{G}_i, n_{i,j}))^2$$

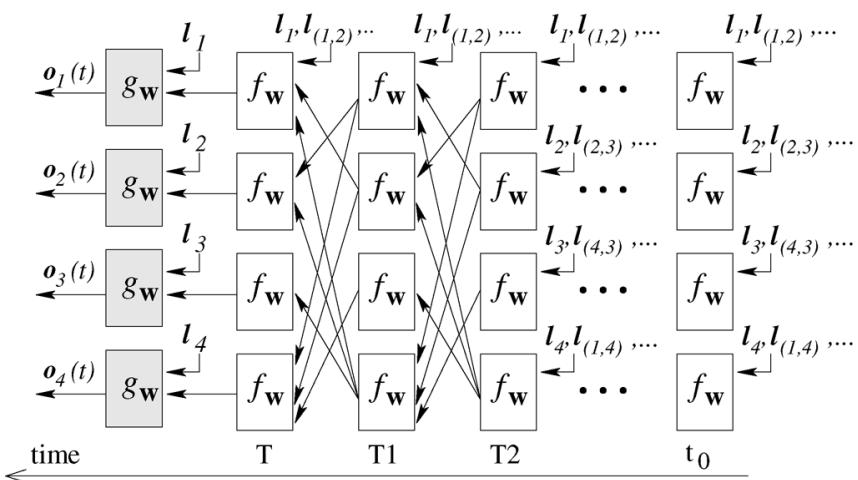
$F_{\mathbf{w}}$ is a contraction map.

Theorem 1 (Differentiability): Let $F_{\mathbf{w}}$ and $G_{\mathbf{w}}$ be the global transition and the global output functions of a GNN, respectively. If $F_{\mathbf{w}}(\mathbf{x}, \mathbf{l})$ and $G_{\mathbf{w}}(\mathbf{x}, \mathbf{l}_N)$ are continuously differentiable w.r.t. \mathbf{x} and \mathbf{w} , then $\varphi_{\mathbf{w}}$ is continuously differentiable

It is not a trivial issue.

In general, this doesn't hold for non-linear dynamics

Backprop over an infinite net



BPTT / BPTS require the storing of the neural information at different layers! **How can we overcome this problem?**

Intuition: since we reach an equilibrium such a storing is not required ...

Backpropagation

Formulation on Nets with Infinite Depth!

```

MAIN
    initialize  $w$ ;
     $x = \text{Forward}(w)$ ;
    repeat
         $\frac{\partial e_w}{\partial w} = \text{BACKWARD}(x, w)$ ;
         $w = w - \lambda \cdot \frac{\partial e_w}{\partial w}$ ;
         $x = \text{FORWARD}(w)$ ;
    until (a stopping criterion);
    return  $w$ ;
end

```

gradient descent

```

FORWARD( $w$ )
    initialize  $x(0)$ ,  $t = 0$ ;
    repeat
         $x(t+1) = F_w(x(t), l)$ ;
         $t = t + 1$ ;
    until  $\|x(t) - x(t-1)\| \leq \varepsilon_f$ 
    return  $x(t)$ ;
end

```

forward step computes the state
on a net with “infinite depth”:
relaxation to an equilibrium

```

BACKWARD( $x, w$ )
     $o = G_w(x, l_N)$ ;
     $A = \frac{\partial F_w}{\partial x}(x, l)$ ;
     $b = \frac{\partial e_w}{\partial o} \cdot \frac{\partial G_w}{\partial x}(x, l_N)$ ;
    initialize  $z(0)$ ,  $t=0$ ;
    repeat
         $z(t) = z(t-1) \cdot A + b$ ;
         $t = t - 1$ ;
    until  $\|z(t-1) - z(t)\| \leq \varepsilon_b$ ;
     $c = \frac{\partial e_w}{\partial o} \cdot \frac{\partial F_w}{\partial w}(x, l_N)$ ;
     $d = z(t) \cdot \frac{\partial F_w}{\partial w}(x, l)$ ;
     $\frac{\partial e_w}{\partial w} = c + d$ ;
    return  $\frac{\partial e_w}{\partial w}$ ;
end

```

backward step computes the state
on a net with “infinite depth”:
relaxation to an equilibrium

Backpropagation

Formal proof

$$\mathbf{z}(t) = \mathbf{z}(t+1) \cdot \frac{\partial F_{\mathbf{w}}}{\partial \mathbf{x}}(\mathbf{x}, \mathbf{l}) + \frac{\partial e_{\mathbf{w}}}{\partial \mathbf{o}} \cdot \frac{\partial G_{\mathbf{w}}}{\partial \mathbf{x}}(\mathbf{x}, \mathbf{l}_N)$$

Then, the sequence $\mathbf{z}(T), \mathbf{z}(T-1), \dots$ converges to a vector $\mathbf{z} = \lim_{t \rightarrow -\infty} \mathbf{z}(t)$ and the convergence is exponential and independent of the initial state $\mathbf{z}(T)$. Moreover

$$\frac{\partial e_{\mathbf{w}}}{\partial \mathbf{w}} = \frac{\partial e_{\mathbf{w}}}{\partial \mathbf{o}} \cdot \frac{\partial G_{\mathbf{w}}}{\partial \mathbf{w}}(\mathbf{x}, \mathbf{l}_N) + \mathbf{z} \cdot \frac{\partial F_{\mathbf{w}}}{\partial \mathbf{w}}(\mathbf{x}, \mathbf{l}) \quad (8)$$

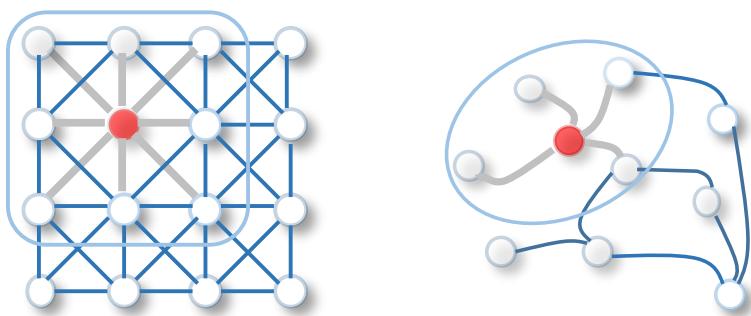
holds, where \mathbf{x} is the stable state of the GNN.

ALTERNATIVE VIEWS ON GNN COMPUTATIONAL MODEL

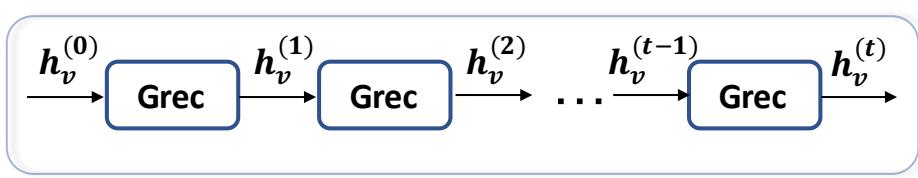
A Comprehensive Survey on Graph Neural Networks

Zonghan Wu, Shirui Pan, *Member, IEEE*, Fengwen Chen, Guodong Long,
Chengqi Zhang, *Senior Member, IEEE*, Philip S. Yu, *Fellow, IEEE*

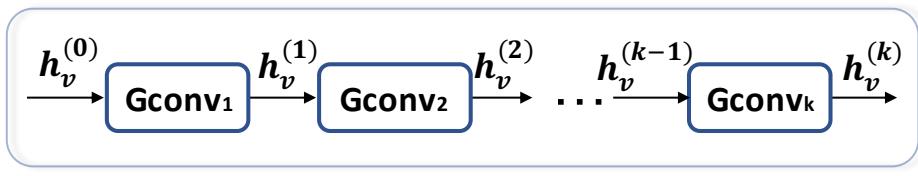
Convolution on Graphs



RecGNNs vs ConvGNNs



(a) Recurrent Graph Neural Networks (RecGNNs). RecGNNs use the same graph recurrent layer (Grec) in updating node representations.



(b) Convolutional Graph Neural Networks (ConvGNNs). ConvGNNs use a different graph convolutional layer (Gconv) in updating node representations.

Spectral Approach

$$\mathbf{L} = \mathbf{I}_n - \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$$

$$\mathbf{D}_{ii} = \sum_j (\mathbf{A}_{i,j})$$

$$\mathbf{L} = \mathbf{U} \boldsymbol{\Lambda} \mathbf{U}^T, \text{ where } \mathbf{U} = [\mathbf{u}_0, \mathbf{u}_1, \dots, \mathbf{u}_{n-1}] \in \mathbf{R}^{n \times n}$$

$$\mathbf{U}^T \mathbf{U} = \mathbf{I}$$

$$\mathbf{x} \in \mathbf{R}^n$$

$$\mathcal{F}(\mathbf{x}) = \mathbf{U}^T \mathbf{x}$$

$$\mathcal{F}^{-1}(\hat{\mathbf{x}}) = \mathbf{U} \hat{\mathbf{x}} \quad \mathbf{x} = \sum_i \hat{x}_i \mathbf{u}_i$$

