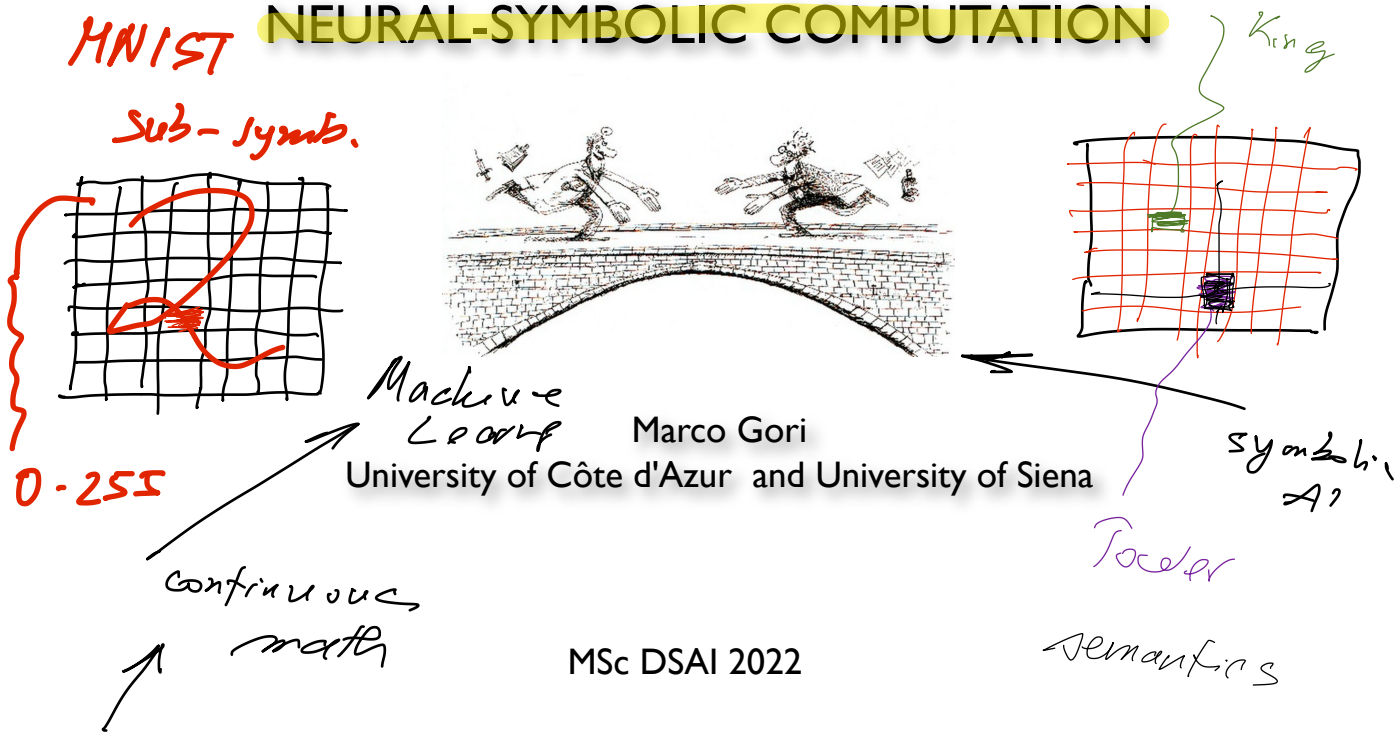
GRAPH NEURAL NETWORKS AND NEURAL-SYMBOLIC COMPUTATION



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

MSc DSAI 2022

UNIFIED VIEW

GRAPH NEURAL NETWORKS (GNN)

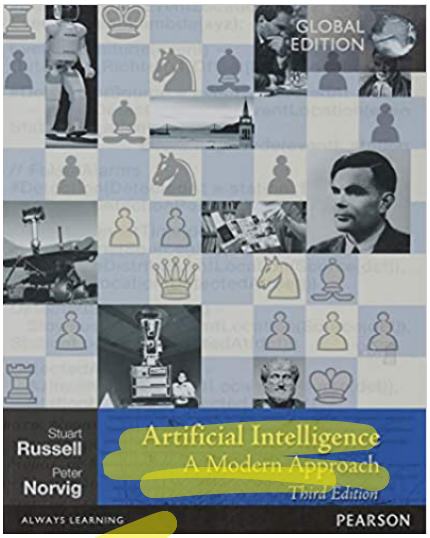
NEURAL-SYMBOLIC COMPUTATION (NESY)

EXPLAINABLE AI

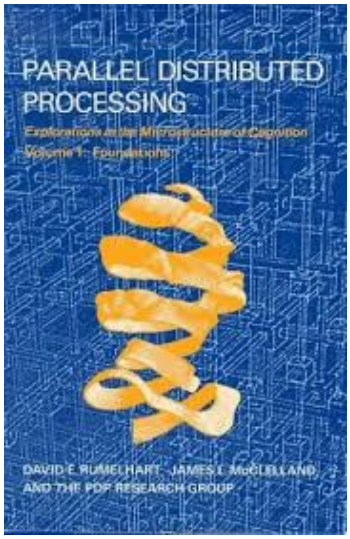
MSc DSAI 2022

COLLECTION OF SUBJECTS

UNIFIED (RESTRICTIVE) VIEW OF INTELLIGENCE



?



MSc DSAI 2022

OUTLINE

- Graph Neural Networks (GNN)
- Neural-Symbolic Computing

the next frontier?

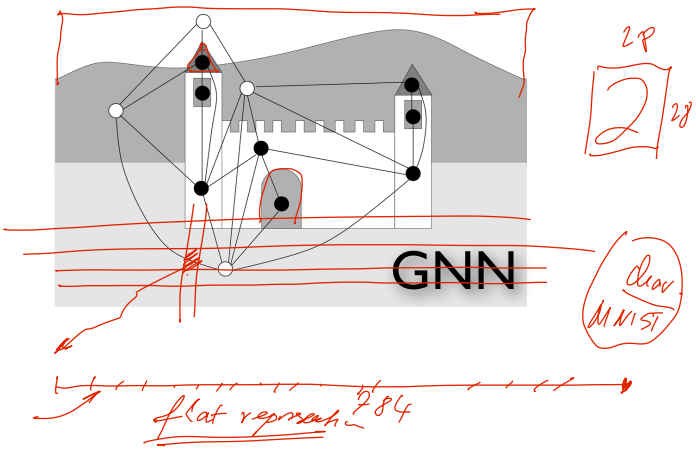
MSc DSAI 2022

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]

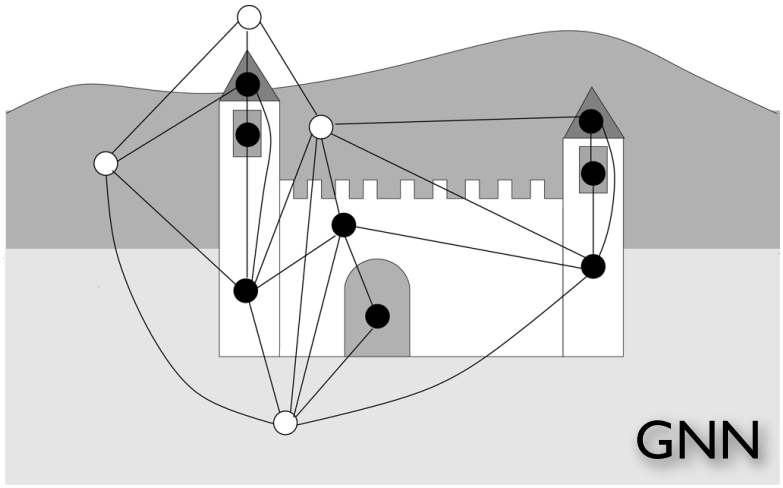
- COMBINE is bias + GRU
- Learning: **Backprop-through-time** after unrolling T steps

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

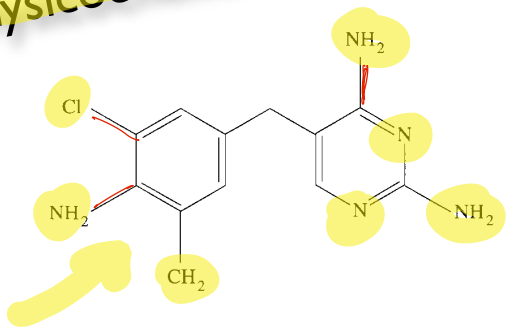
WHY MODELING THE ENVIRONMENT BY GRAPHS?



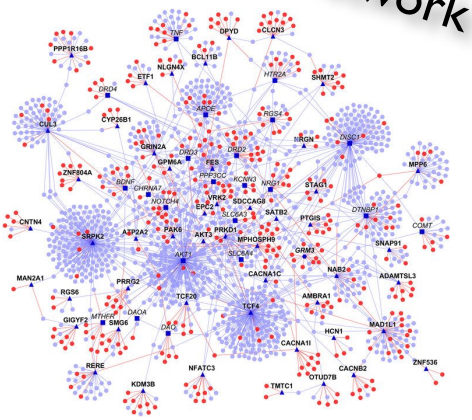
MSc DSAI 2022

CHEMISTRY AND BIOLOGY

physicochemical behavior



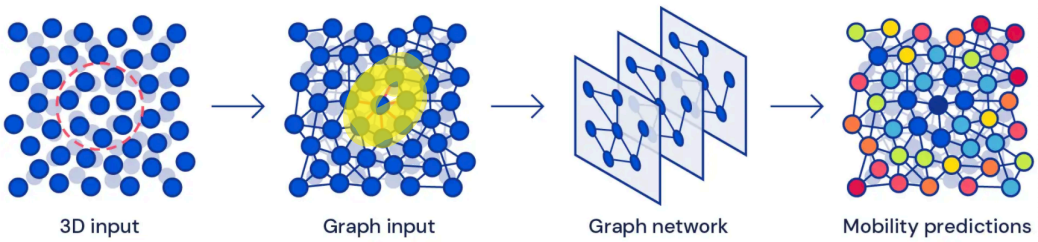
Protein Interaction Network

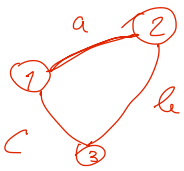


What are the features? The atoms, the bonds?

UNDERSTANDING GLASSES

DeepMind
Towards understanding glasses with graph neural networks

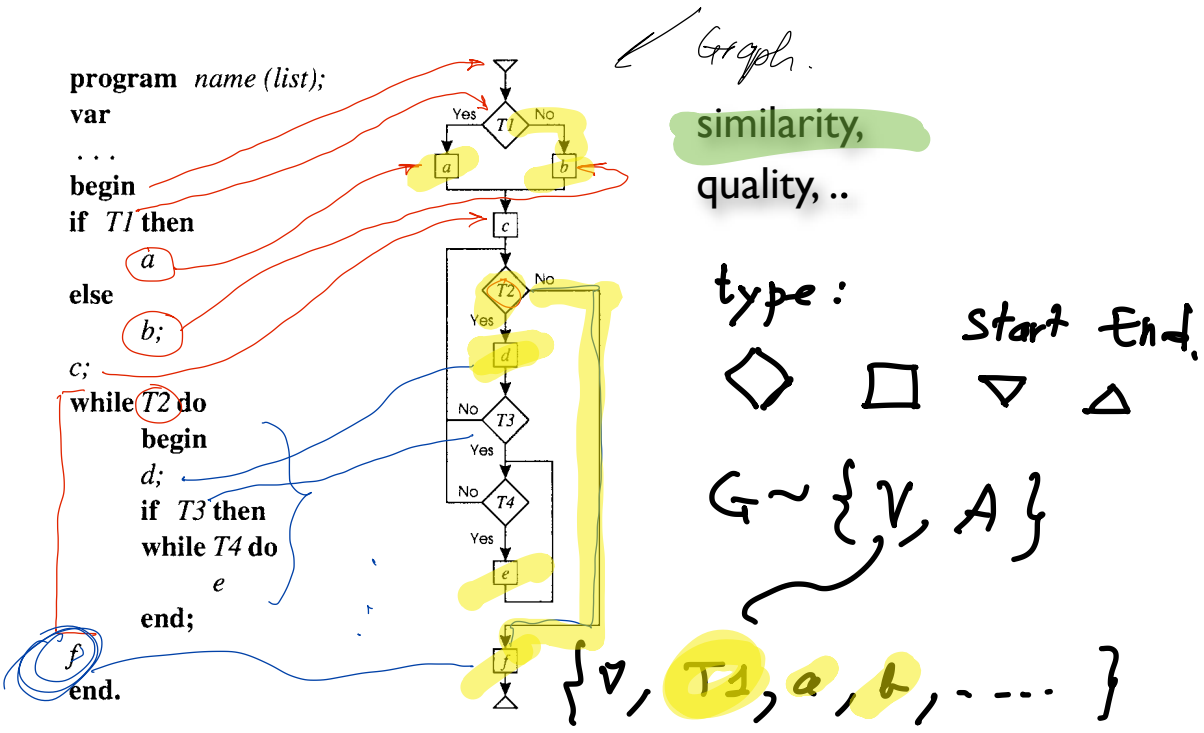




$G \sim \{V, A\}$

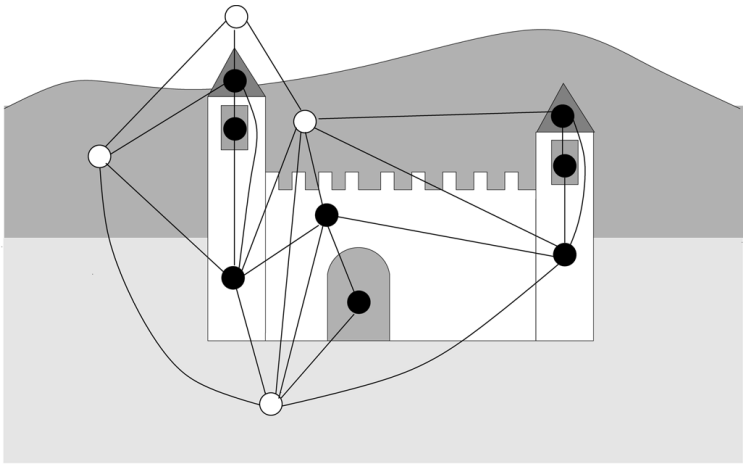
$\{1, 2, 3\}$

SOFTWARE ENGINEERING



PATTERN RECOGNITION

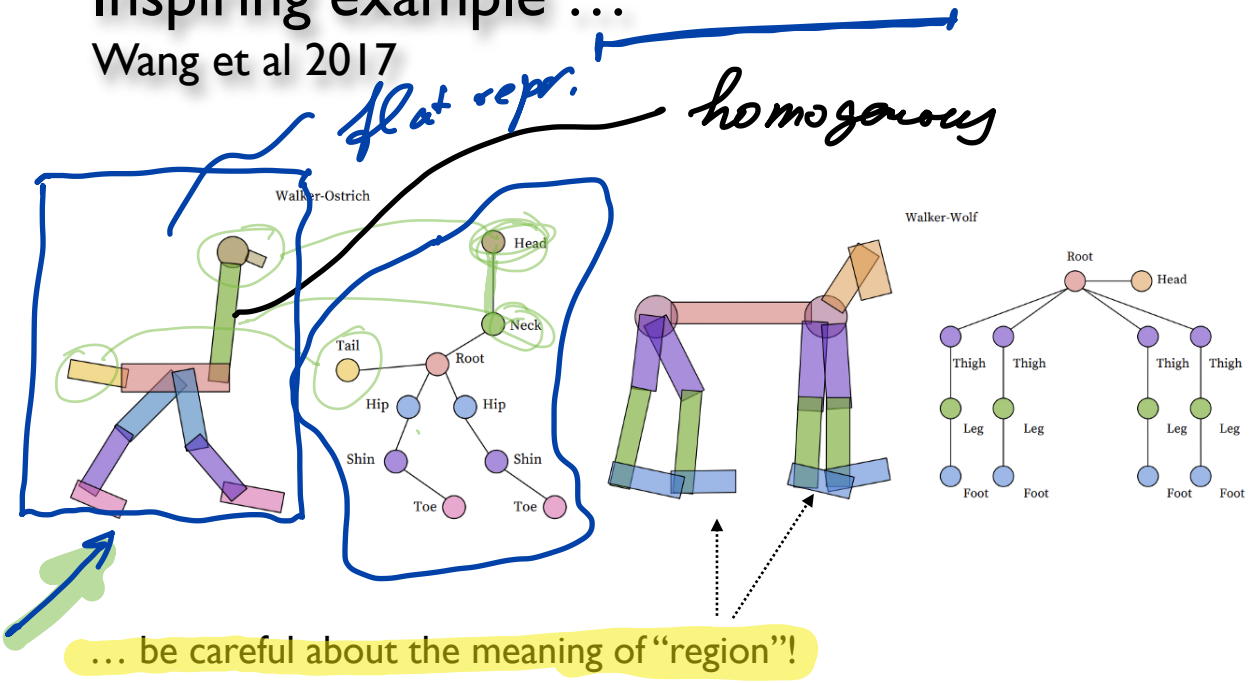
Structured representations



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

Inspiring example …

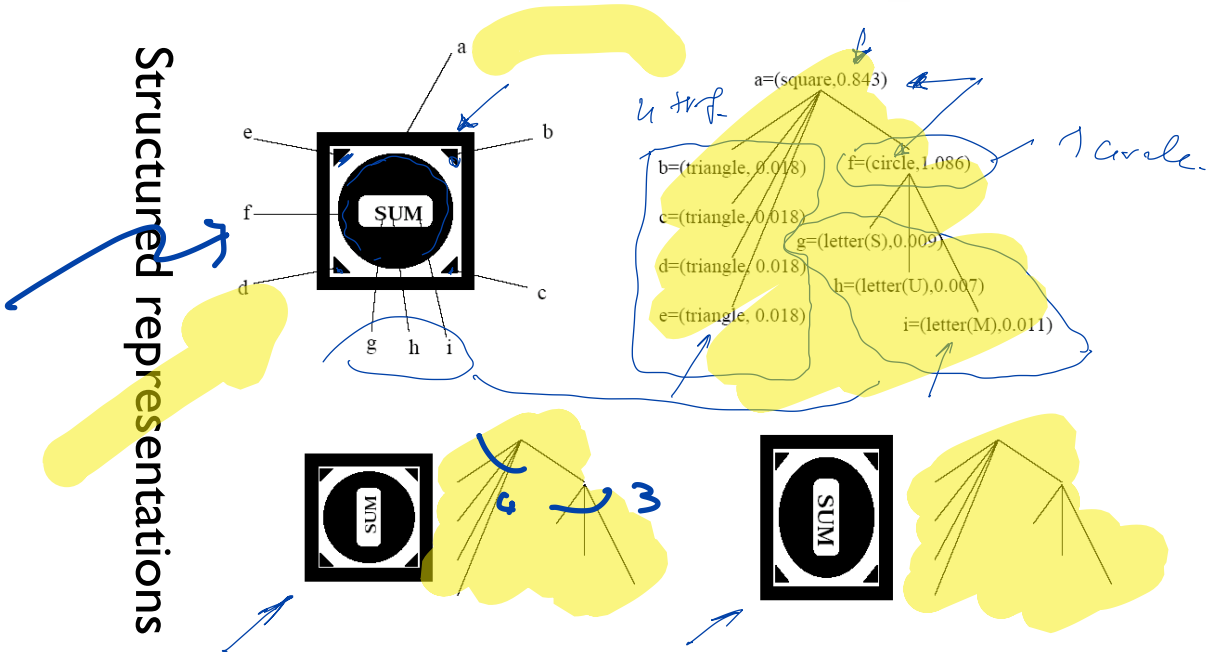
Wang et al 2017



… be careful about the meaning of "region"!

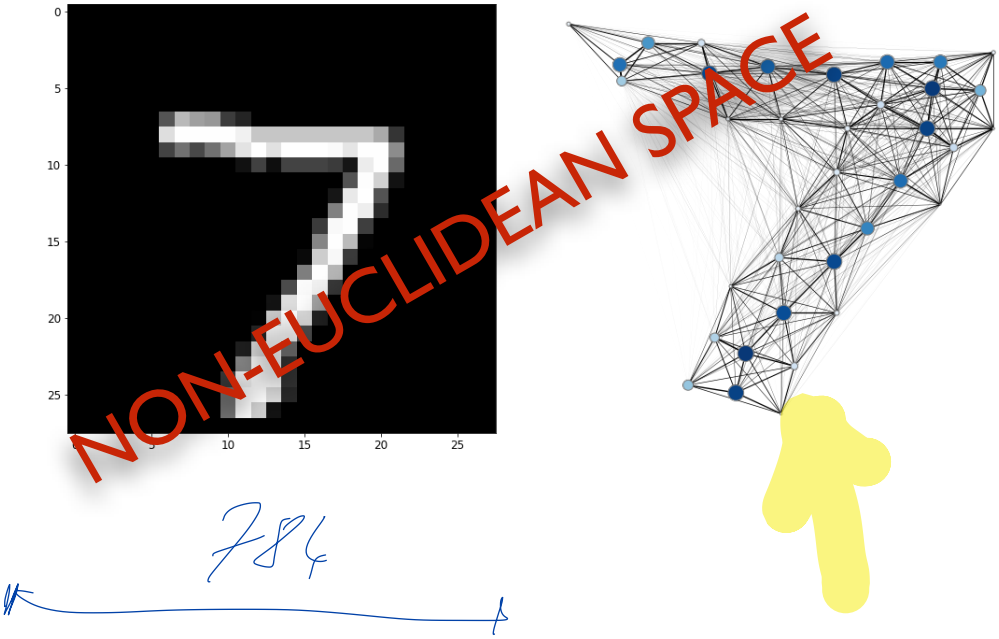
Document Analysis and Recognition

Structured representations



Another example: XY-trees for Document Analysis and Recognition

MNIST Handwritten Char Recognition



Scene Graphs

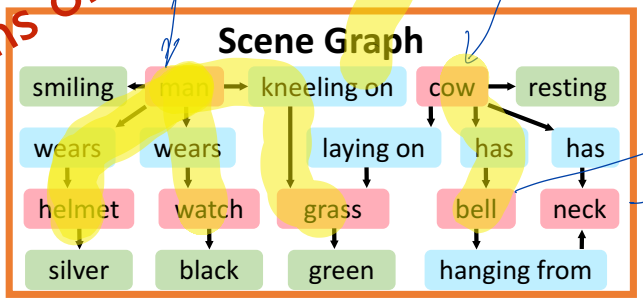
Johnson, Krishna, Stark, Li, Shamma, Bernstein, and Fei-Fei, "Image Retrieval with Scene Graphs", CVPR 2015

VISUAL HARMONY

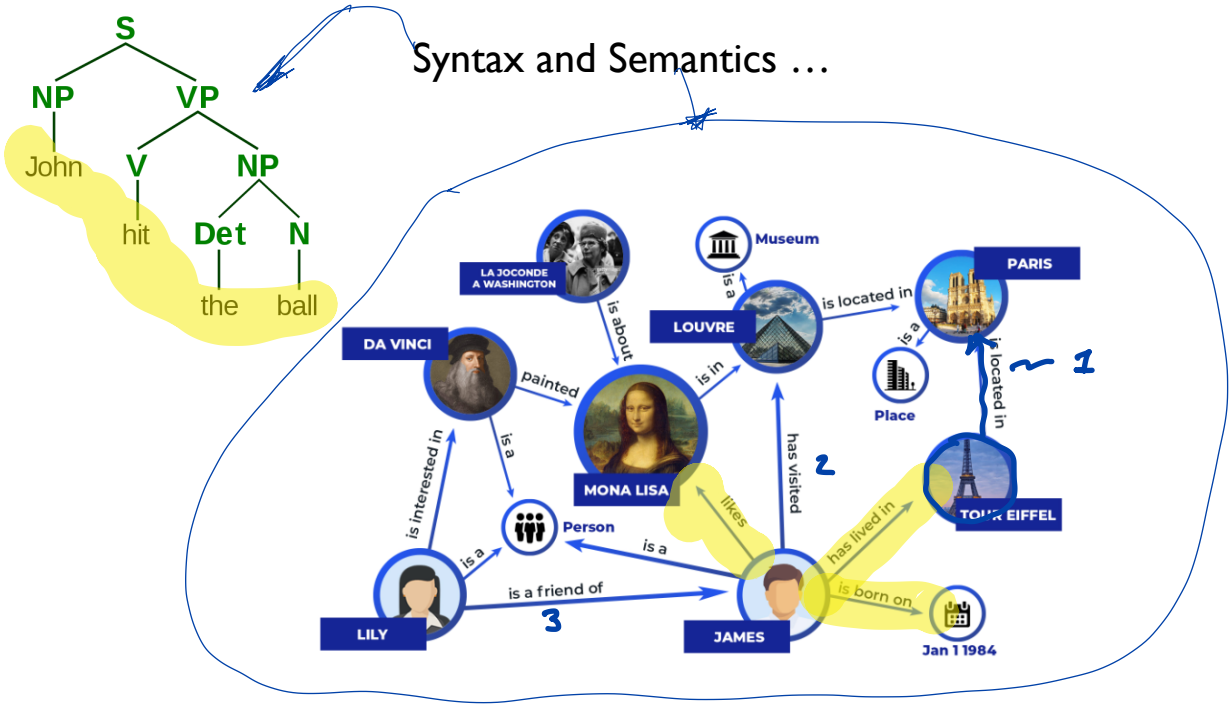
Graphs on concepts

pink nodes

inferences on the graph



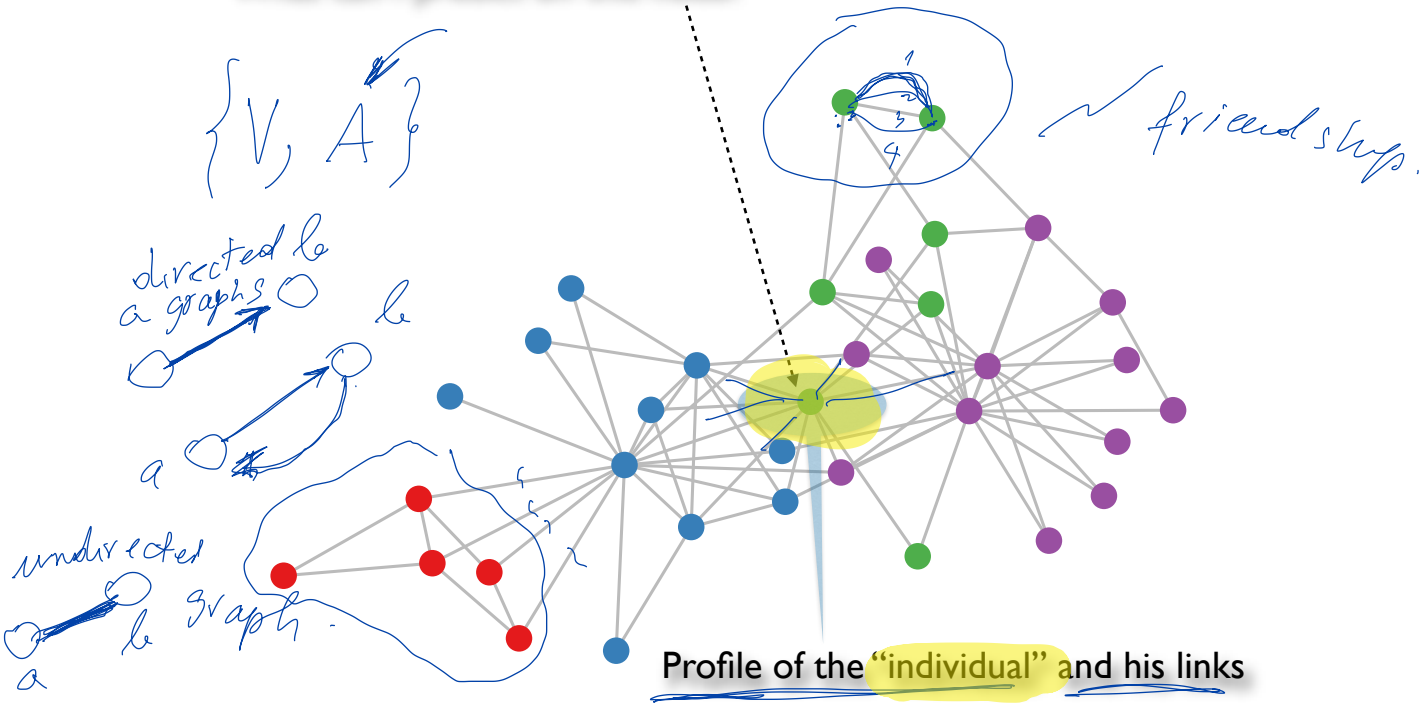
In Natural Language …



SOCIAL NETWORKS

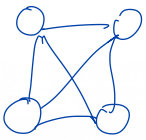
a unique graph!

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



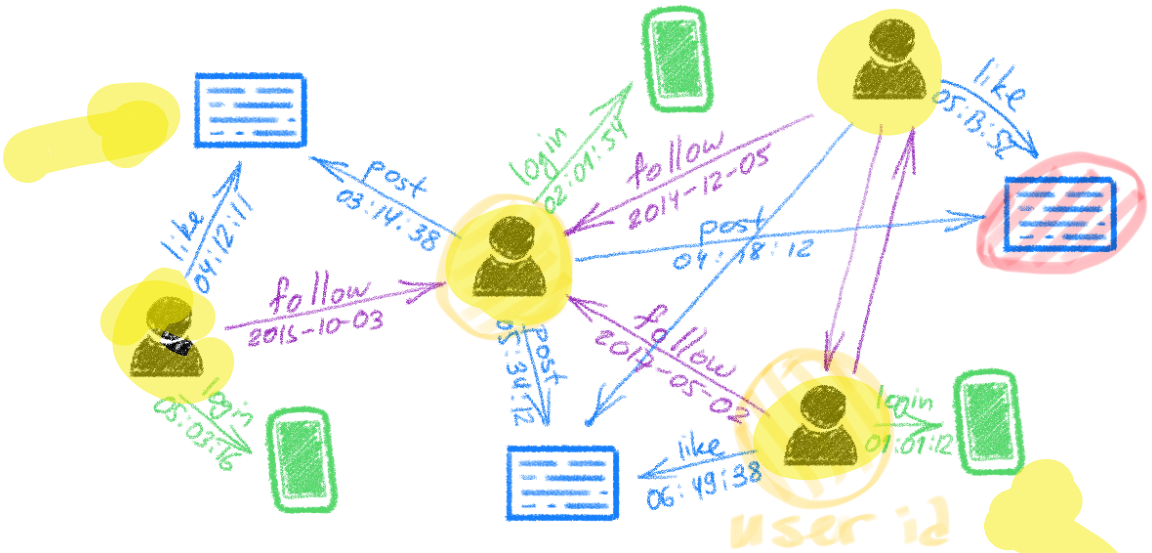
CITATION NETWORKS

by CitNetExplorer



SOCIAL NETWORKS

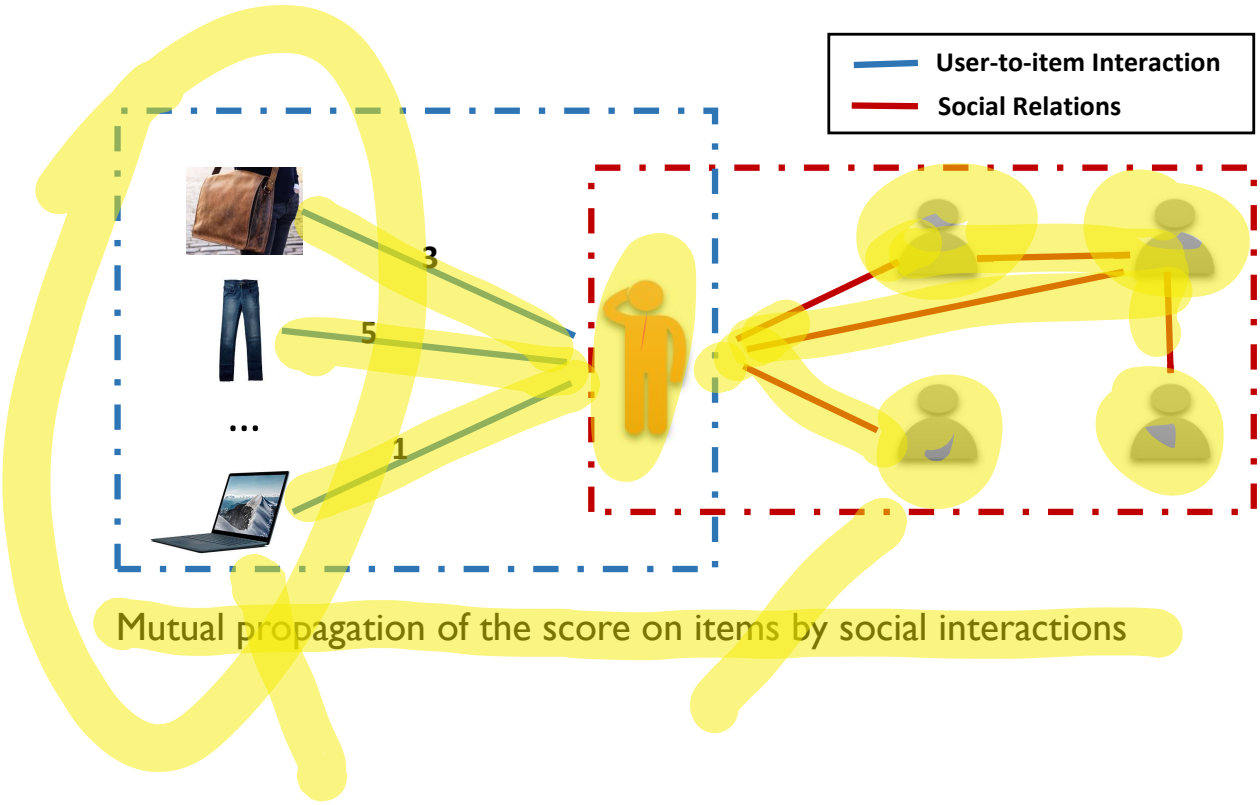
Michael Bronstein - Twitter



vertexes and edges can be of different types!

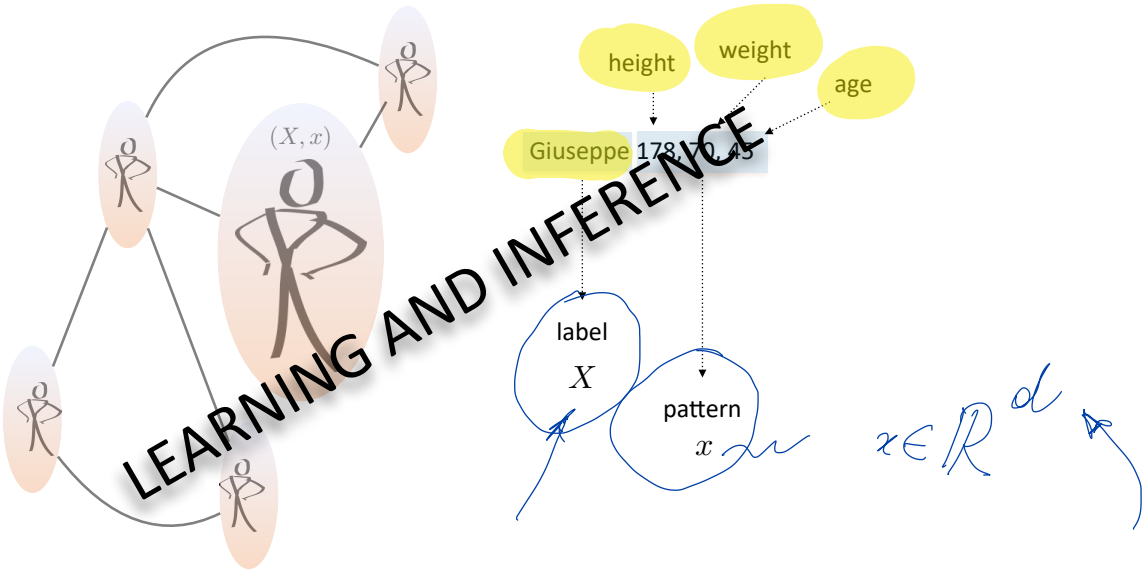
RECOMMENDATION SYSTEMS

W. Fan et al. 2019



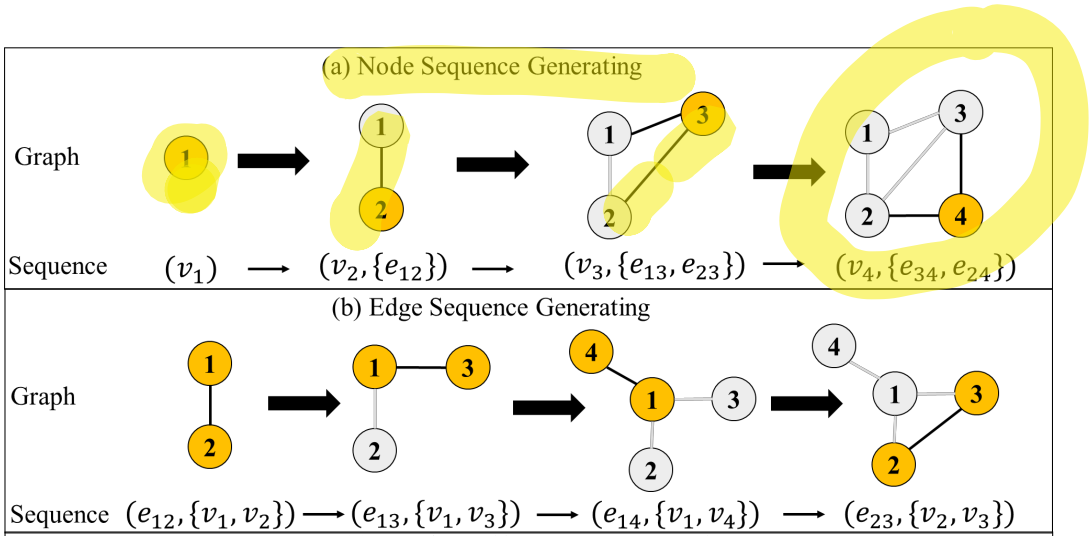
Mutual propagation of the score on items by social interactions

NET WITH SYMBOLS AND SUB-SYMBOLS



GRAPH GENERATION

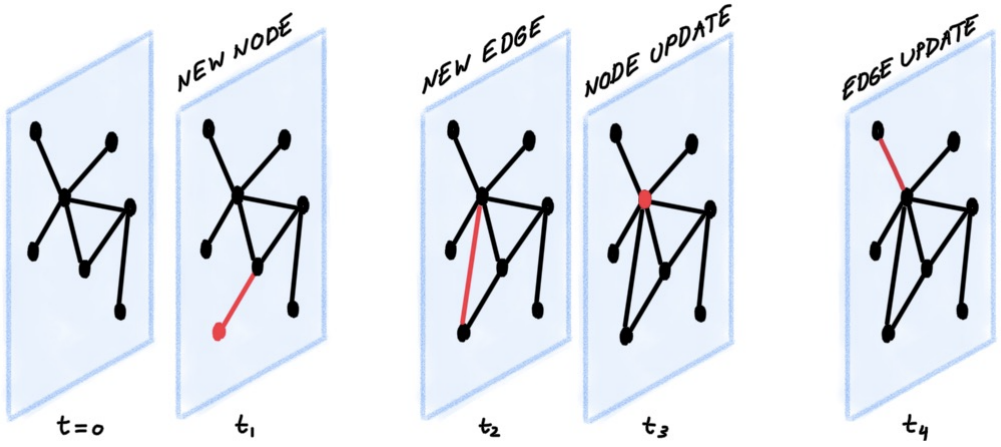
GHUO & ZHAO, 2020



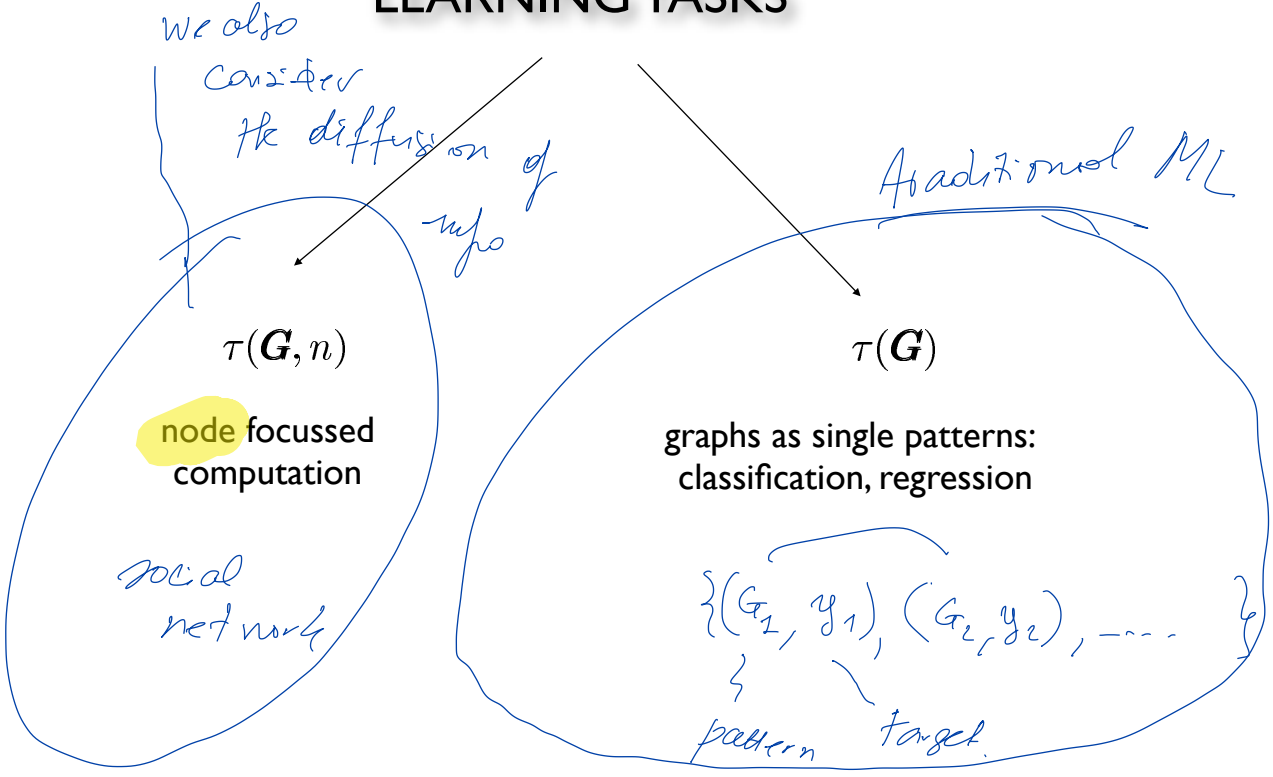
as new nodes/arcs come we need a generation scheme
to define the construction of the graph

GRAPH GENERATION

Emanuele Rossi, Imperial College



LEARNING TASKS



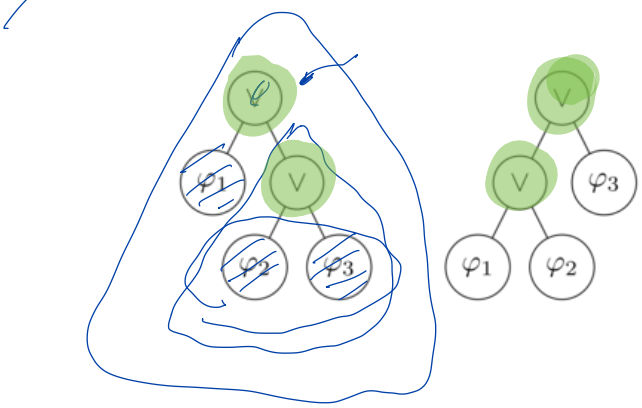
WHAT IF WE START EXPRESSING
RELATIONS ON CATEGORIES?

LOGIC STATEMENTS

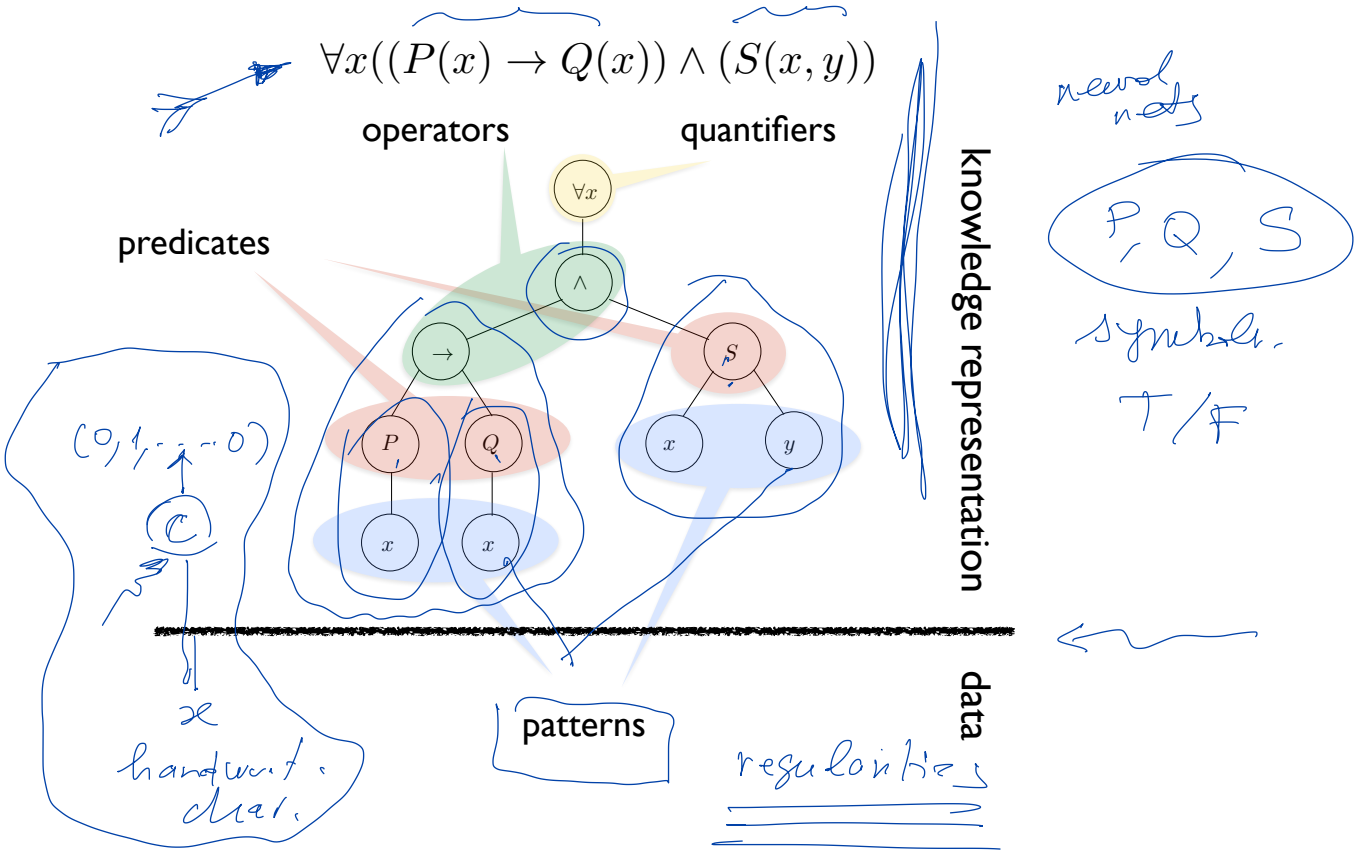
logic formulas represented by parse trees

$$\varphi_1 \vee (\varphi_2 \vee \varphi_3) = (\varphi_1 \vee \varphi_2) \vee \varphi_3$$

associativity - graphical counterpart



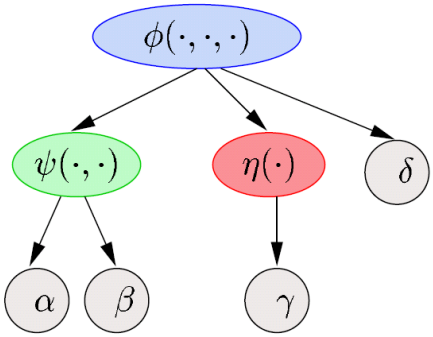
FOL LOGIC STATEMENTS



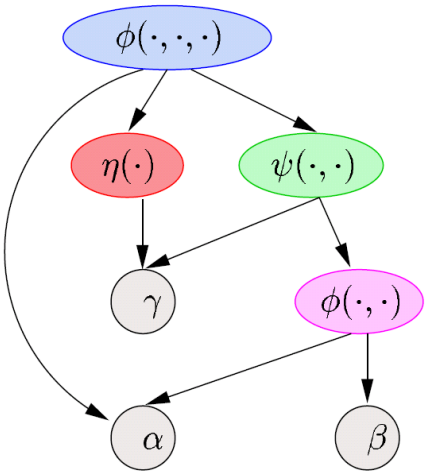
LOGIC STATEMENTS

… by Directed Acyclic Graphs

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



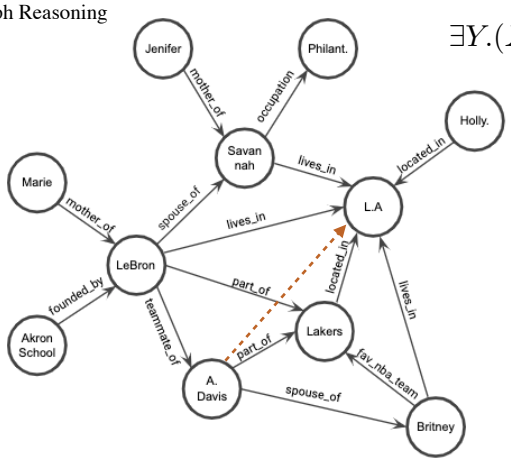
$\phi(\alpha, \eta(\gamma), \psi(\gamma, \phi(\alpha, \beta)))$



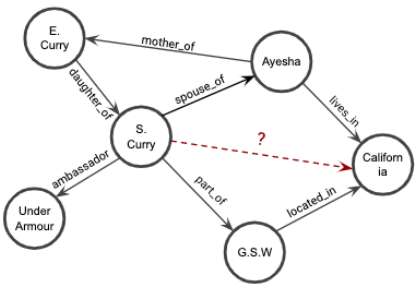
KNOWLEDGE GRAPHS

W. Hamilton et al (MILA)
ICML-2020

... Subgraph Reasoning

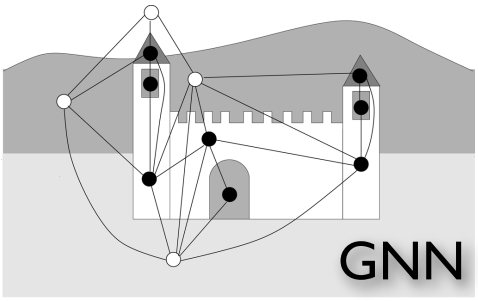


$$\exists Y.(X, \texttt{spouse\_of}, Y) \wedge (Y, \texttt{lives\_in}, Z) \rightarrow (X, \texttt{lives\_in}, Z)$$



GRAPH NEURAL NETS

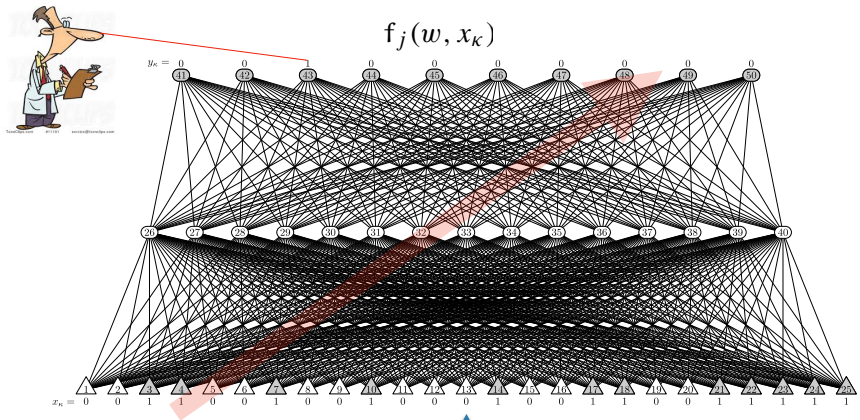
The viewpoint of diffusion

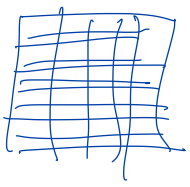


MSc DSAI 2022

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

A Graph (Neural Network) processes a vector …



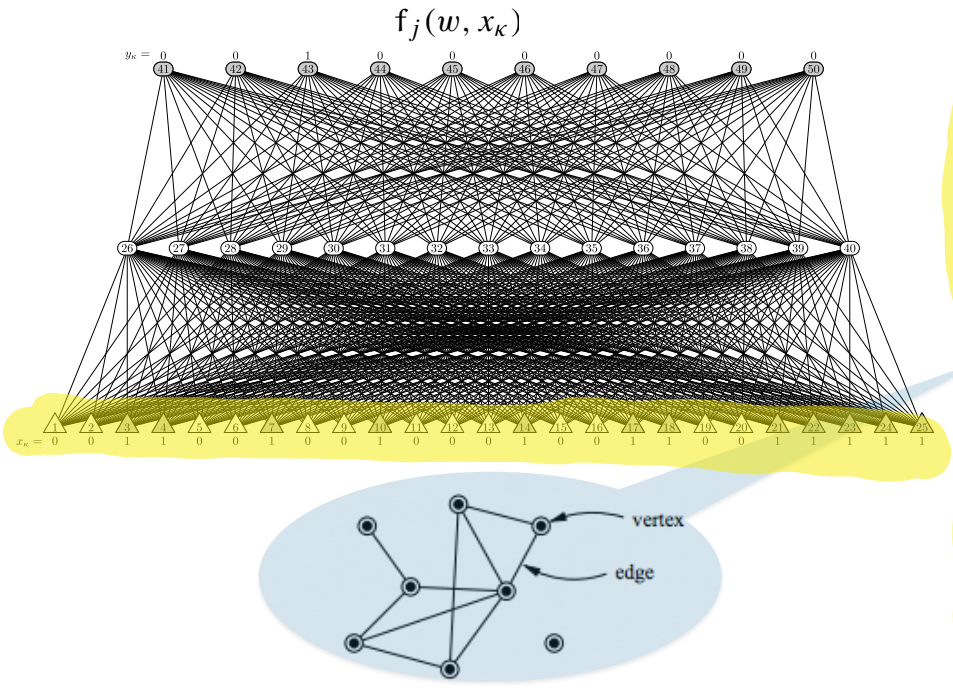


$\sim$ 00011000001001000000001000000010000000101000010001111100000000011.

MSc DSAI 2022

flat - based representation.

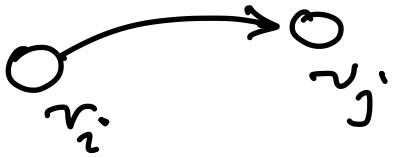
How Can Graph (ANN) Process a Graph?



how to represent it as a "flat" input?

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

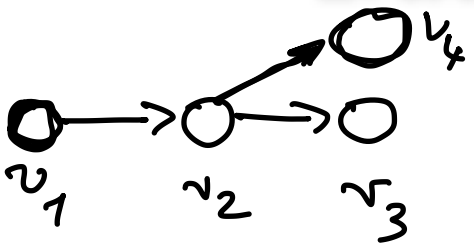
$v_i \prec v_j$



partial ordering relation.

DIFFUSION-BASED PROCESSING

LET US BEGIN WITH DAGs …



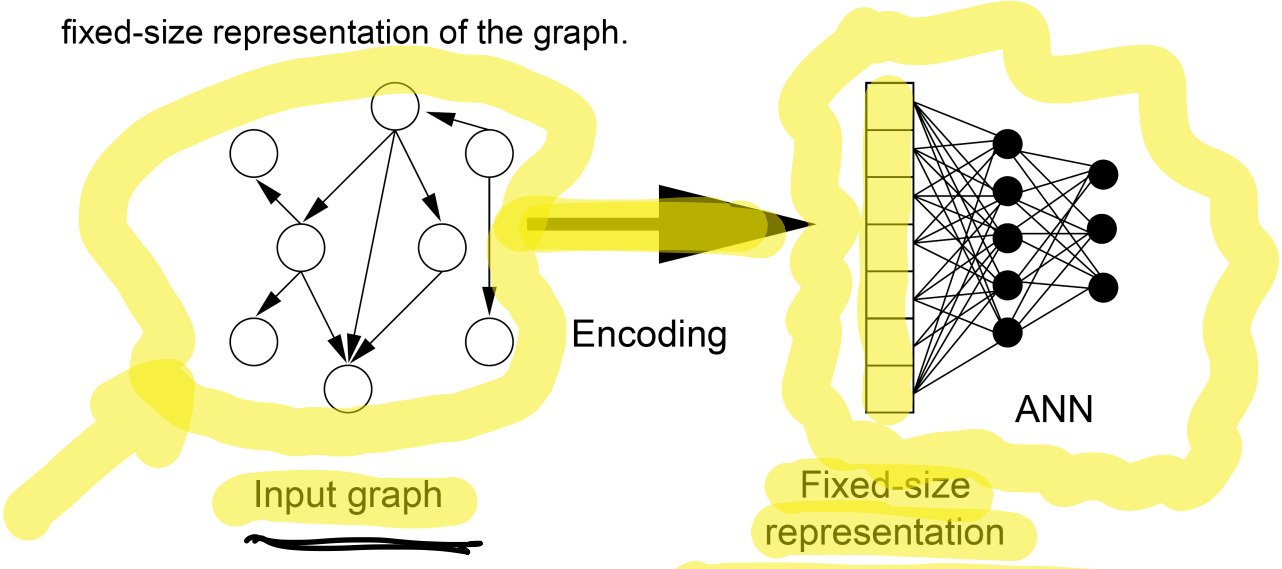
$$\begin{cases} v_1 \prec v_2 \prec v_3 \\ v_2 \prec v_4 \end{cases}$$

Directed Acyclic Graph.

Adaptive vs fixed encoding

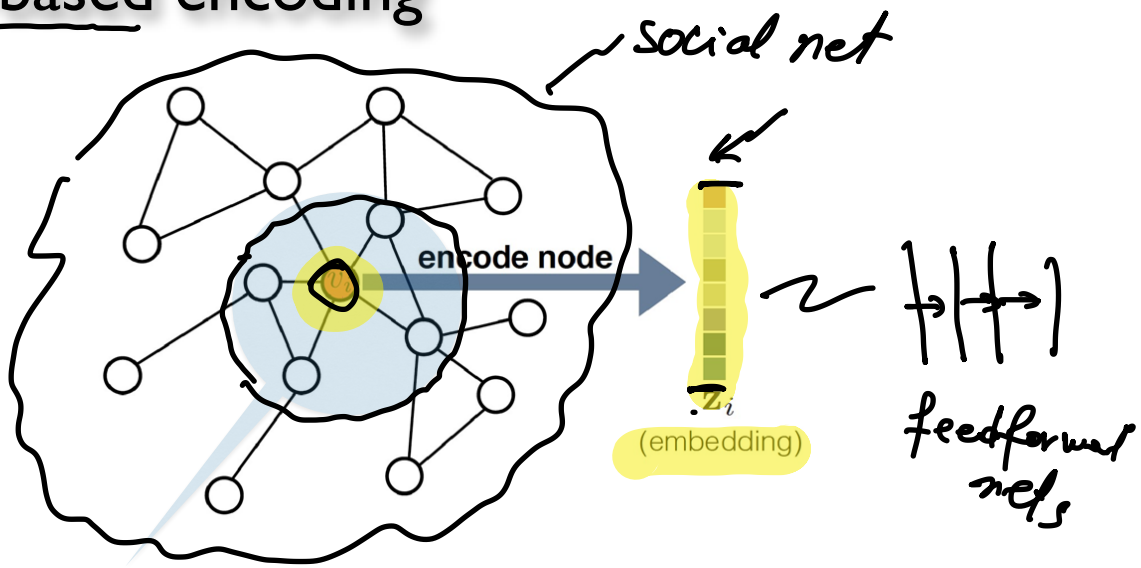
graph-focussed computation.

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



How can we provide an appropriate encoding?

Node-based encoding

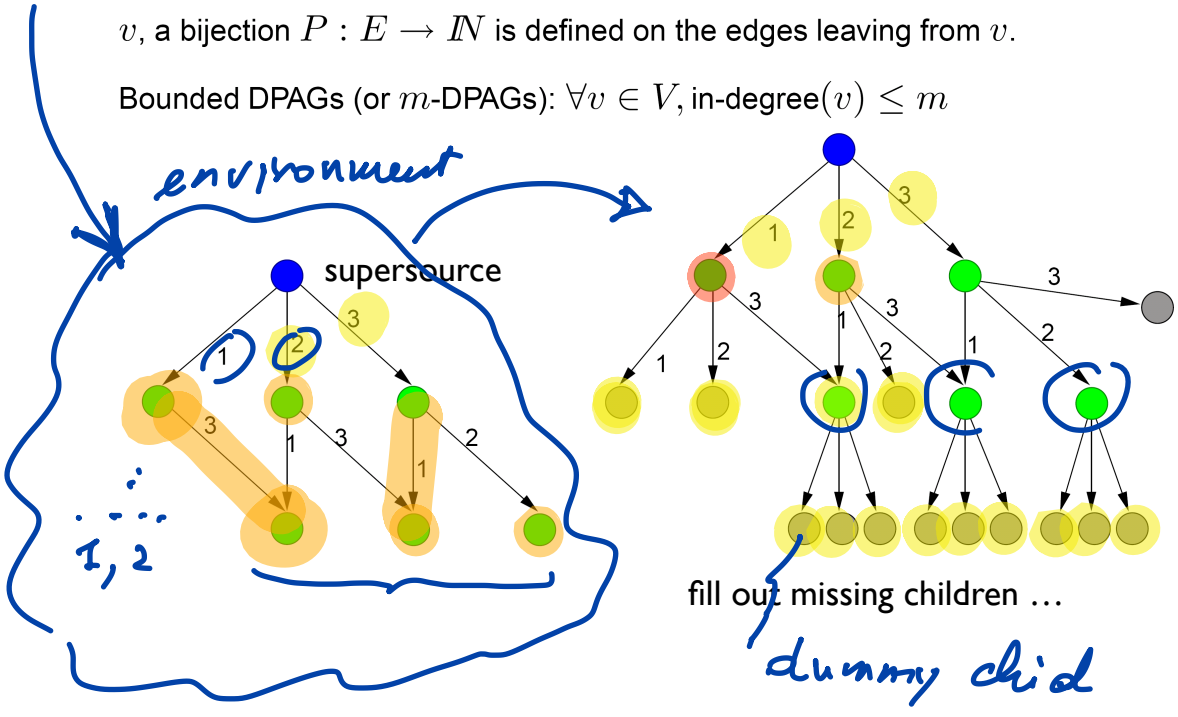


we could choose an "appropriate window"

Directed Positional Acyclic Graphs

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

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



Graph Transductions

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

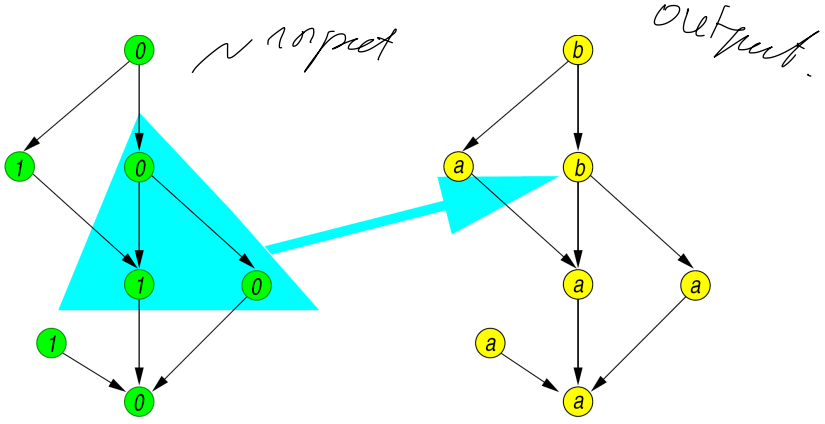
A transduction $\mathcal{T}$ is a subset of $\mathcal{U}^{\#} \times \mathcal{Y}^{\#}$.

Restrictions:

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

from graphs to graphs

Choice of the "window" …



$\boldsymbol{U} \in \mathcal{U}^{\#}$ $\mathcal{T}(\boldsymbol{U}) \in \mathcal{Y}^{\#}$

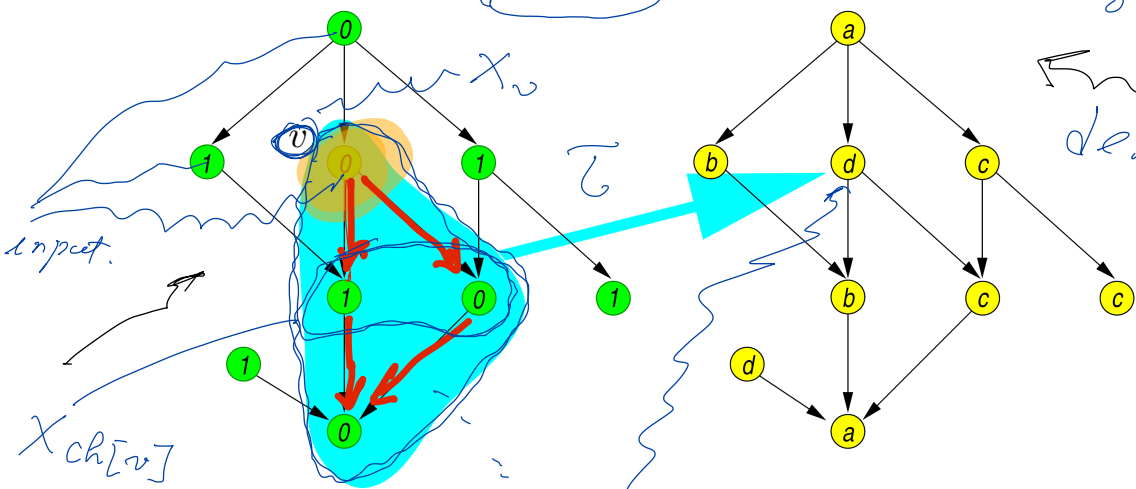
Compare to sequences: here we have to decide both:

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

Information diffusion and causality

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

node v along with the descendant $\text{de}[v]$



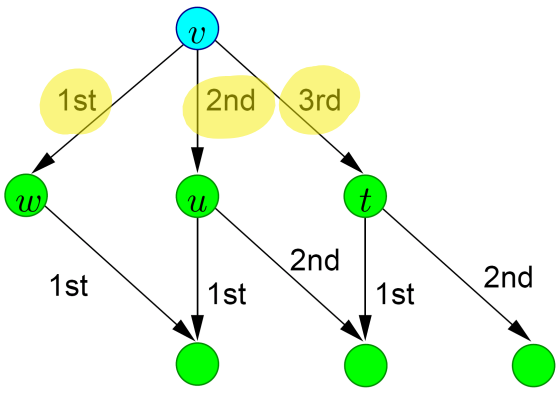
x_v is all the info I need to perform the comp. = past information

node-based computation.

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



State-based representation

IMPORTANCE OF THE POSITION

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

state of node v

$$\begin{cases} \boldsymbol{X}_v &= f(\boldsymbol{X}_{\mathrm{ch}[v]}, \boldsymbol{U}_v) \\ \boldsymbol{Y}_v &= g(\boldsymbol{X}_v, \boldsymbol{U}_v) \end{cases}$$

state trans.

output

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

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

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

1-D

Compare to temporal dynamical systems:

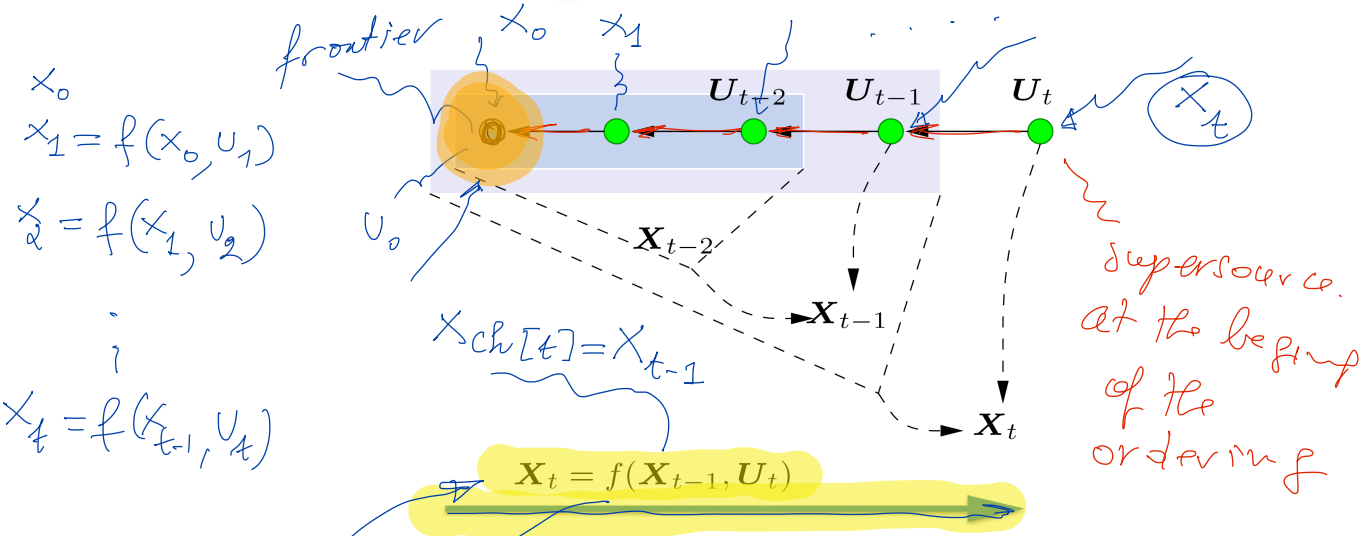
$$\begin{cases} \boldsymbol{X}_t &= f(\boldsymbol{X}_{t-1}, \boldsymbol{U}_t) \\ \boldsymbol{Y}_t &= g(\boldsymbol{X}_t, \boldsymbol{U}_t) \end{cases}$$

ordering of children state does matter!

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

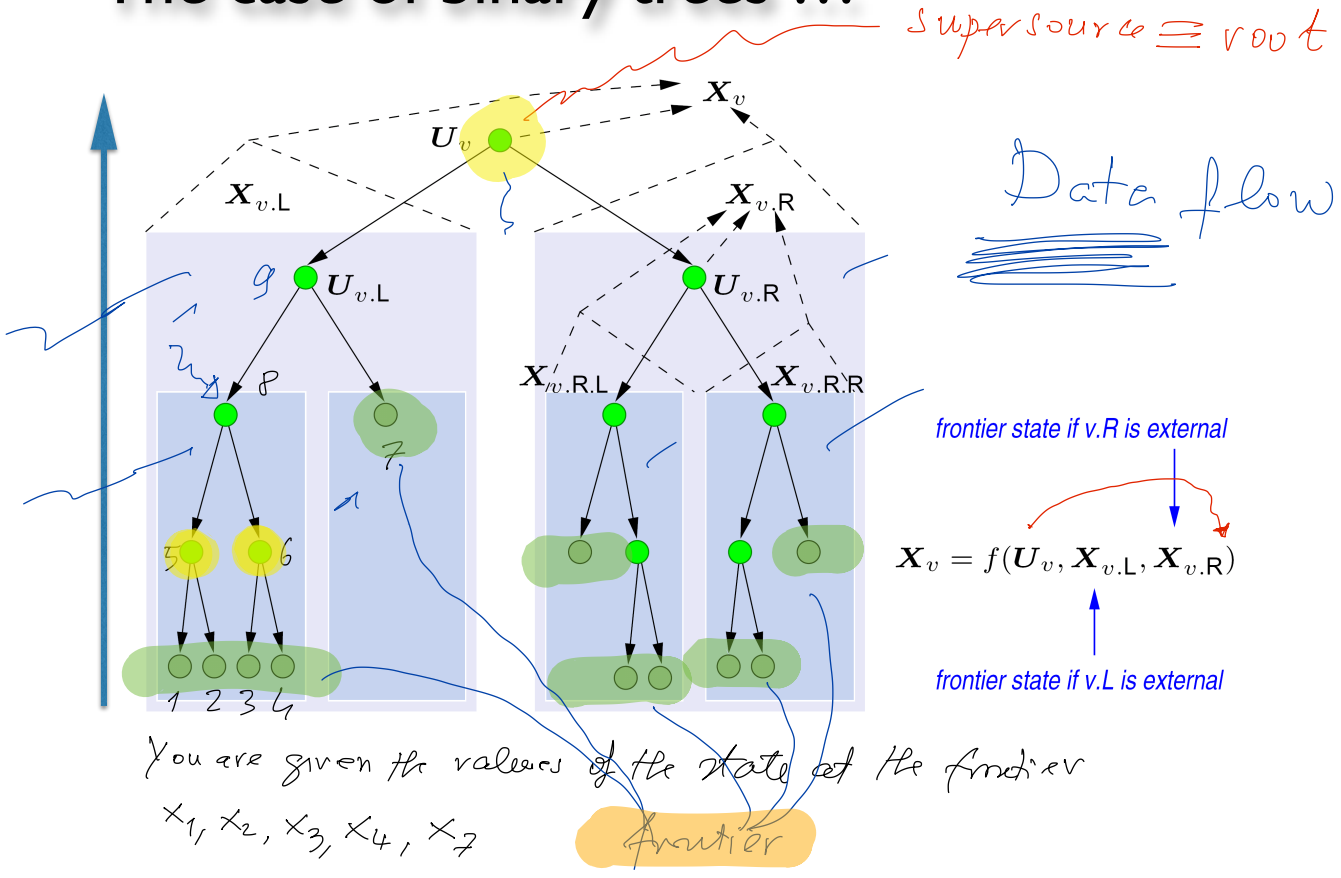
Reduction to sequences

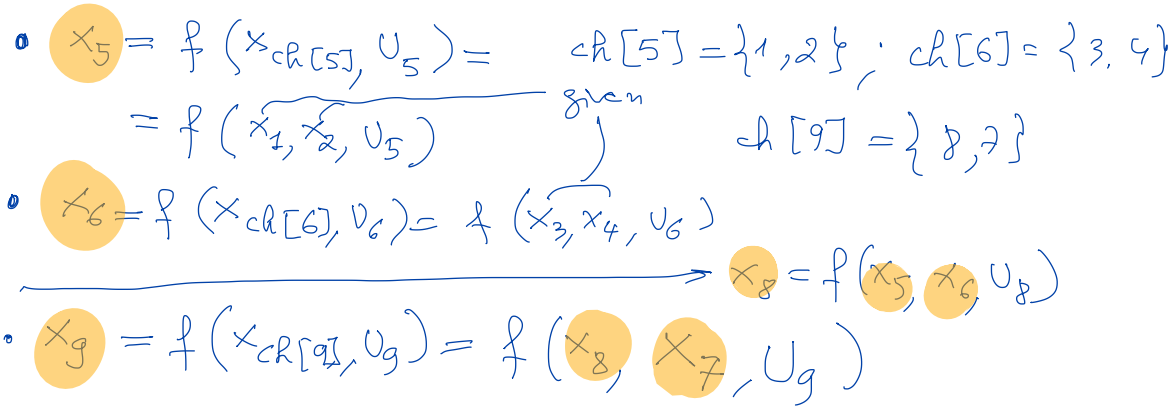


NB: for $t = 0$, $\boldsymbol{X} = \boldsymbol{X}_0$ is the **initial state**

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

The case of binary trees …





Supersource transductions

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

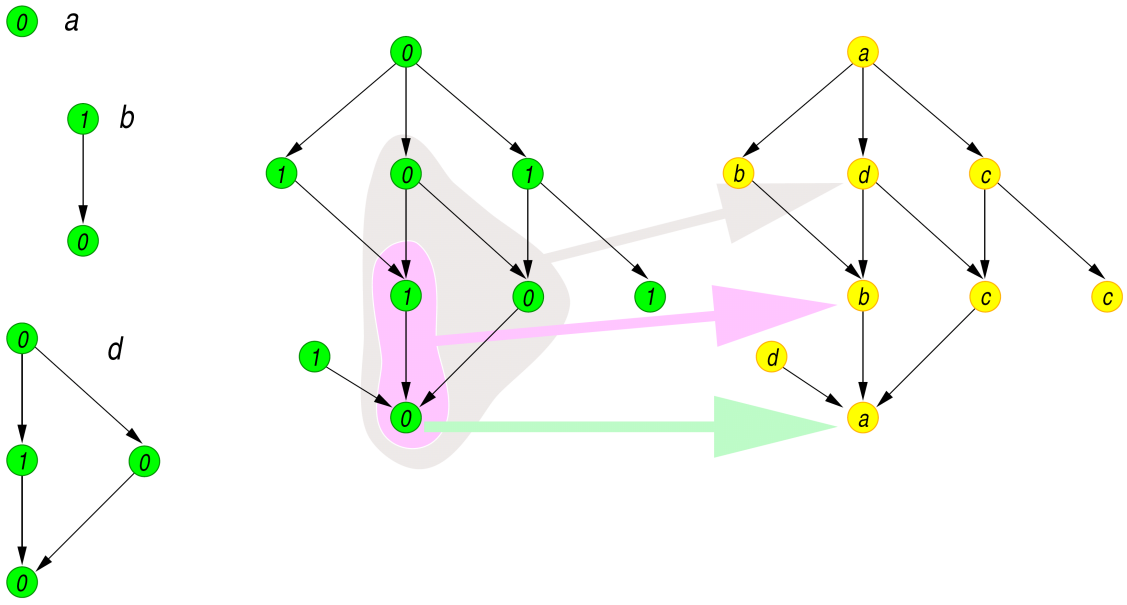
$$\boldsymbol{X}_v = f(\boldsymbol{X}_{\mathsf{ch}[v]}, \boldsymbol{U}_v)$$

- The output label is “emitted” at s:

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

IO-isomorph transductions

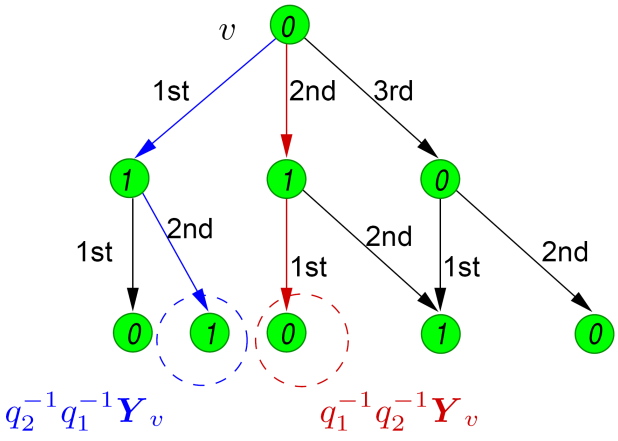
An IO-isomorph transduction is like many supersource transductions:



Generalized shift-operator

- Sequences: $q^{-1}\boldsymbol{Y}_t = \boldsymbol{Y}_{t-1}$ (unitary time delay).
- DOAGs: $q_k^{-1}\boldsymbol{Y}_v$ is the label attached to the k-th child of vertex v.

  NB: $q_k^{-1}\boldsymbol{Y}_v = \emptyset$ if the k-th child of v belongs to the frontier.
- Composition is not commutative:

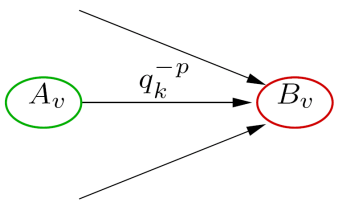


Recursive networks

Hypotheses:
- $\mathcal{T}()$ admits a recursive state space representation;
- input, state, and output are uniformly labeled.

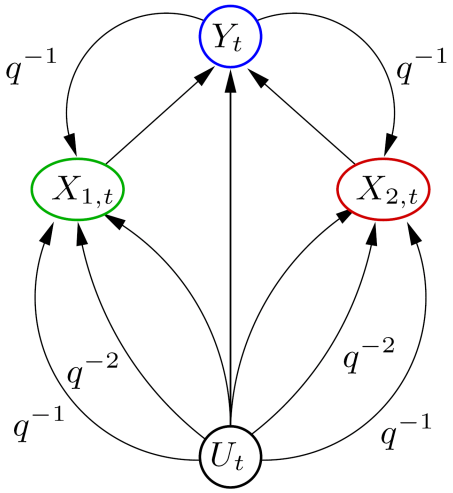
The recursive network of $\mathcal{T}()$ is a directed graph where:

- 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



$$Y_t = g(X_{1,t}, X_{2,t}, U_t)$$

$$X_{1,t} = f(U_t, q^{-1}U_t, q^{-2}U_t, q^{-1}Y_t)$$

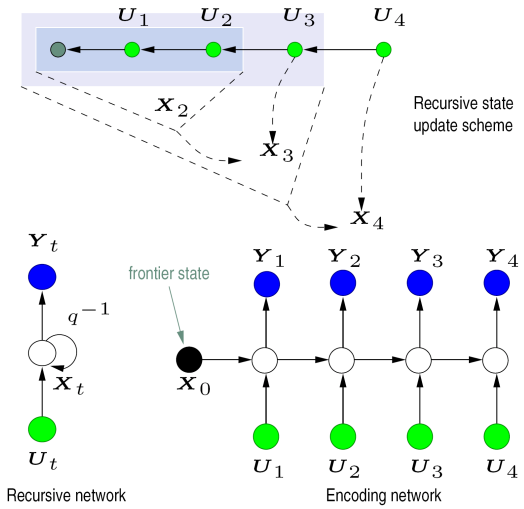
$$X_{2,t} = f(U_t, q^{-1}U_t, q^{-2}U_t, q^{-1}Y_t)$$

Encoding networks

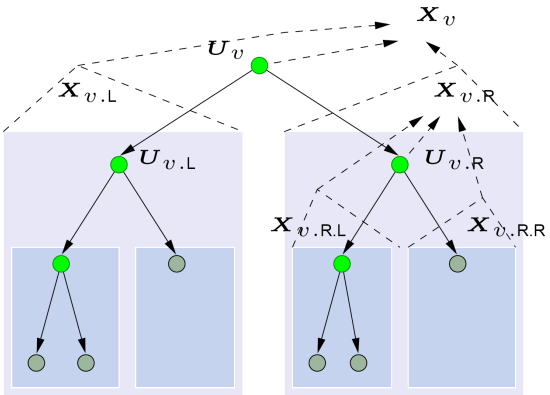
Given a graph $\boldsymbol{U} \in \mathcal{U}^{\#}$ and a recursive transduction $\mathcal{T}$.

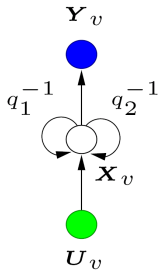
The *encoding network* associated to $\boldsymbol{U}$ and $\mathcal{T}$ is formed by unrolling the recursive network of $\mathcal{T}$ through the input graph $\boldsymbol{U}$.

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

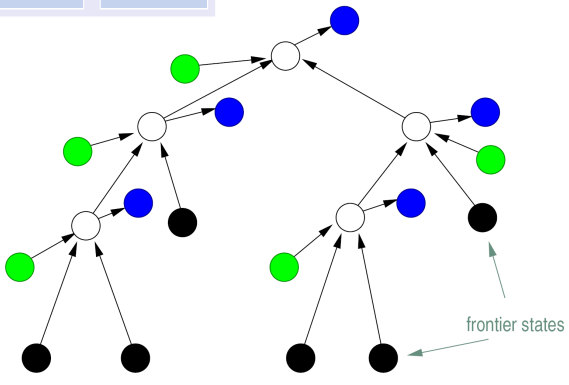


Encoding nets for binary trees



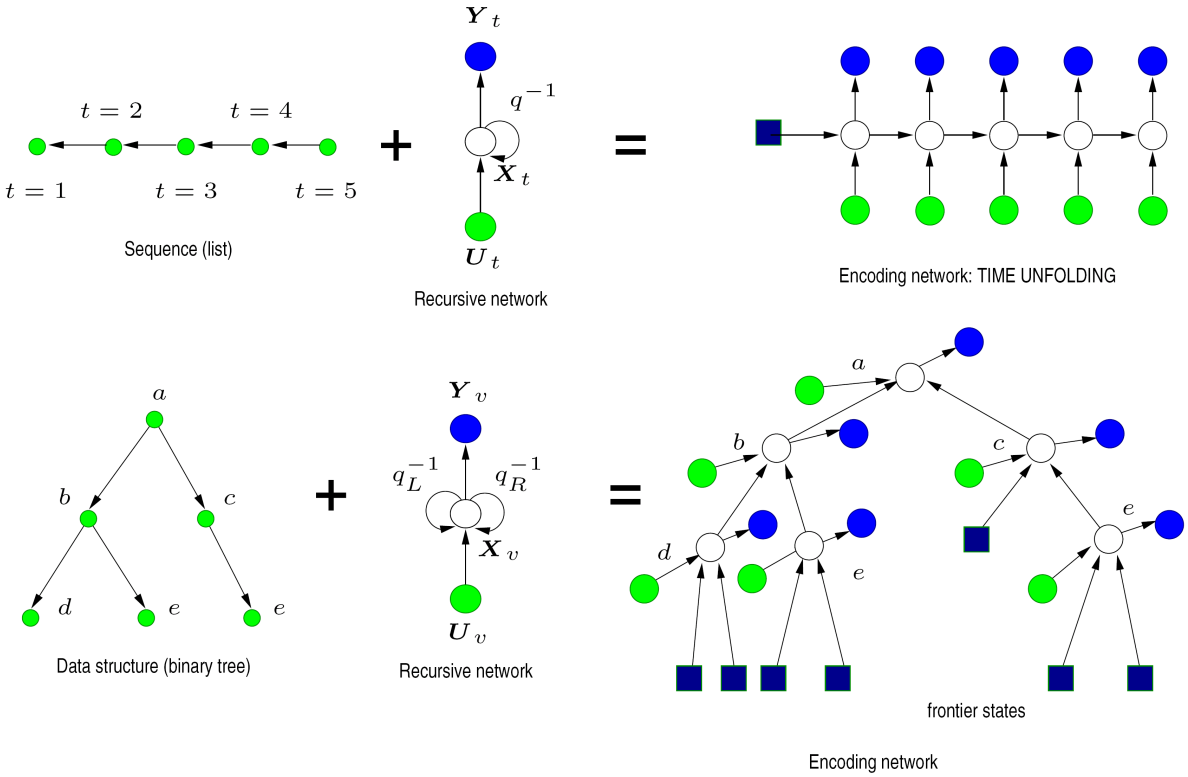


Recursive network



Encoding network

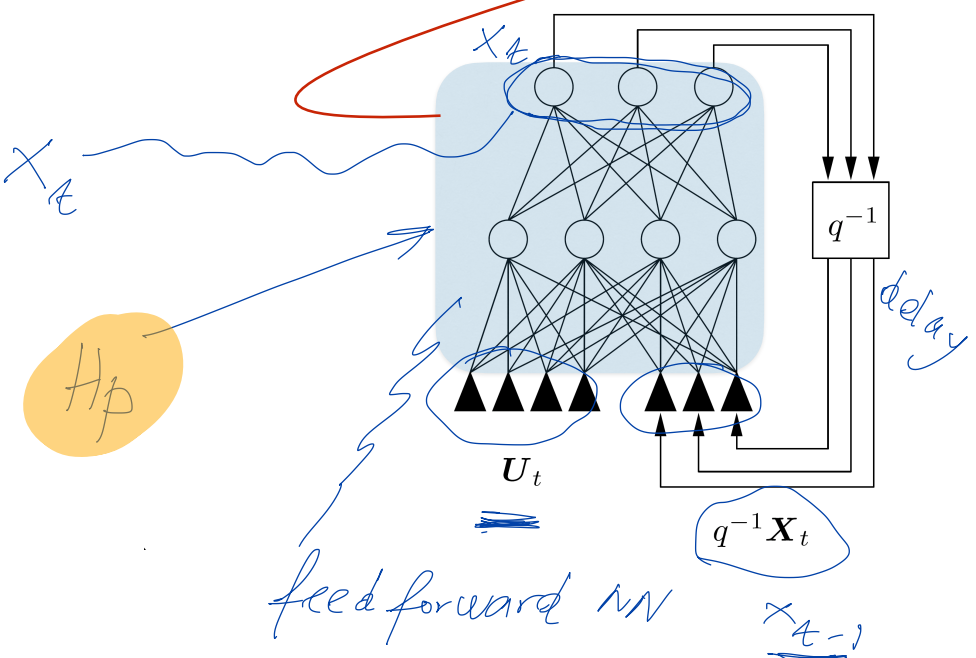
Data structures + recursive nets = encoding nets



Graphs for processing Graphs: Using neural nets for mapping

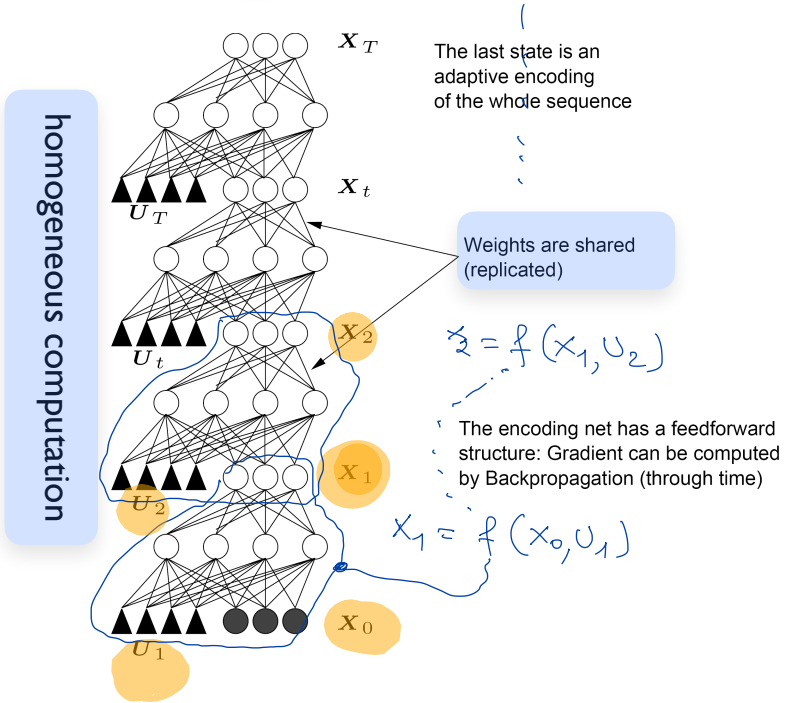
The state transition function is implemented by a MLP.

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



Time unfolding

BPTT

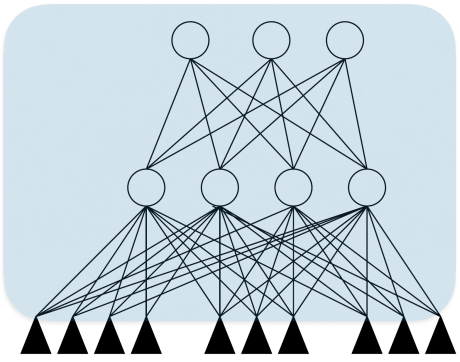


Using neural nets
for binary trees …

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

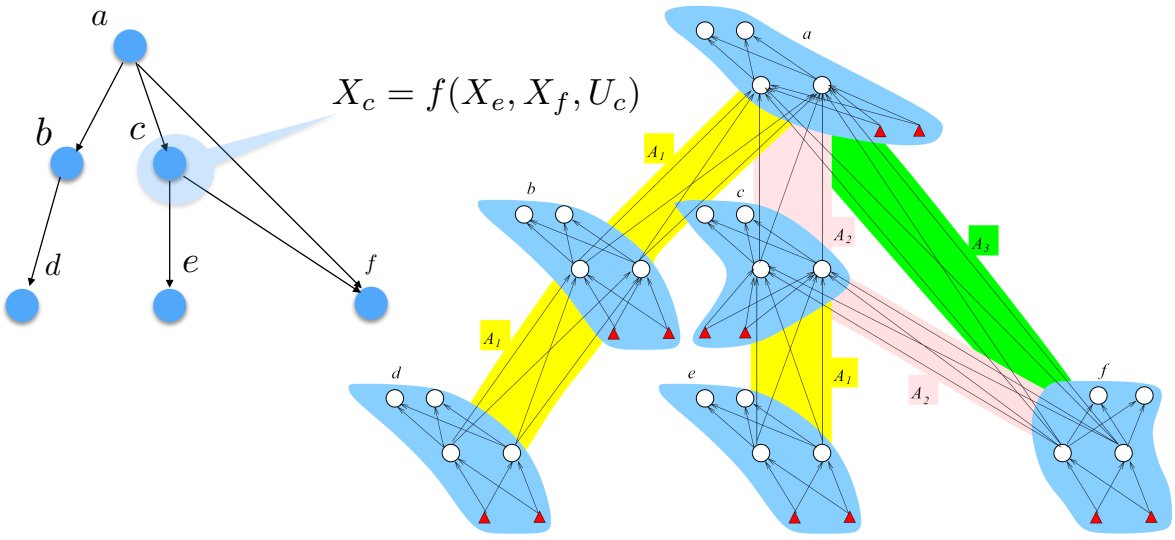
The state transition function is implemented by a MLP (e.g. case of binary trees)

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



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

Structure (graph) unfolding



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

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

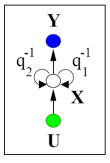
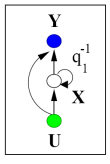
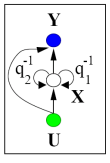
BE CAREFUL: WEIGHT SHARING WHEN TYPES ARE UNIFORM!

… just like in recurrent neural networks for sequences

Non-stationary transductions

Linguistic specification of the recursive network

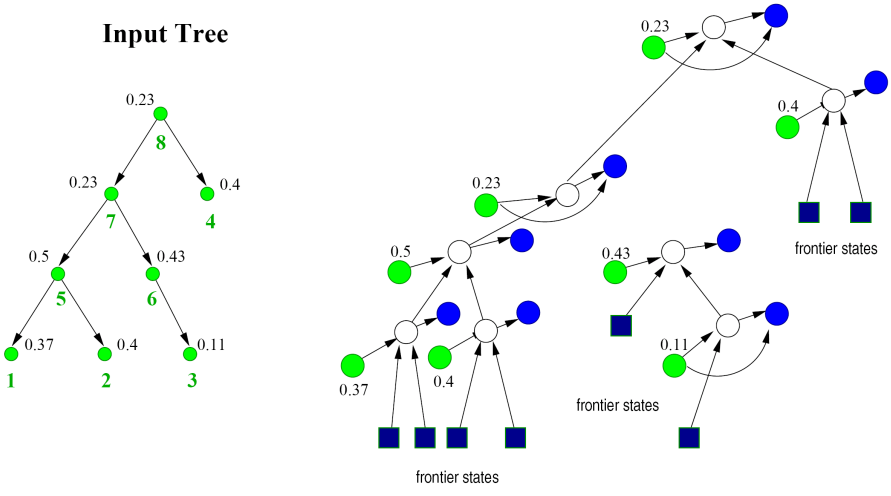
knowledge-based map description

```
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 
                                 
      ;
  }
  else 
                                 
  ;
}
```

Compiling …

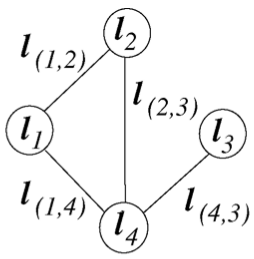
The input tree is mapped to one with different structure!

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



What if DOAG assumption is lost?

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

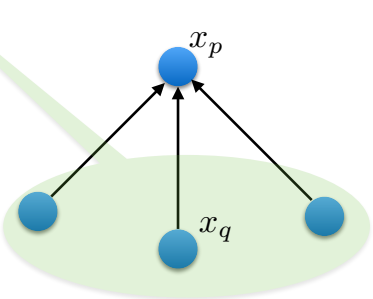


When ordering is lost, the previous data flow computational scheme cannot be established:
We need a different diffusion process!

PageRank:

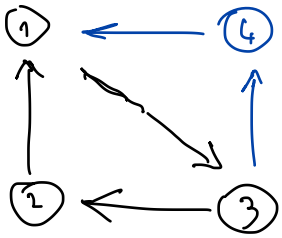
A Related Diffusion Issue

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



over all the graph

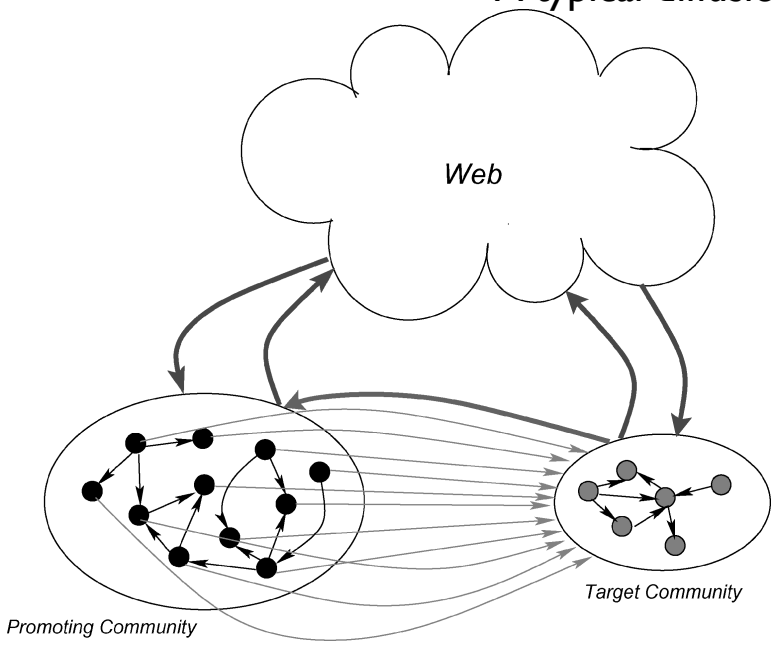
$$\boldsymbol{x} = d\boldsymbol{W}\boldsymbol{x} + (1-d)\mathbb{1}_N$$



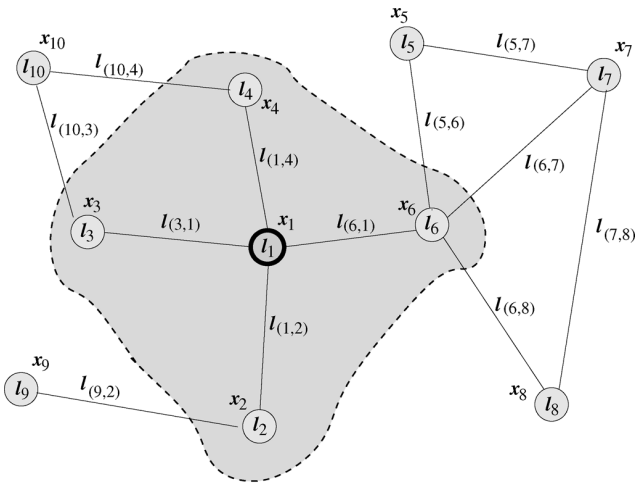
PageRank:

The popular "Web-spam" issue!

A typical diffusion process



Neighbor-based computation



node label connection label neighbor state neighbor label

$$\boldsymbol{x}_n = f_{\boldsymbol{w}}(\boldsymbol{l}_n, \boldsymbol{l}_{\mathrm{co}[n]}, \boldsymbol{x}_{\mathrm{ne}[n]}, \boldsymbol{l}_{\mathrm{ne}[n]})$$

$$\boldsymbol{o}_n = g_{\boldsymbol{w}}(\boldsymbol{x}_n, \boldsymbol{l}_n)$$

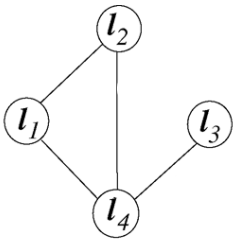
$$\boldsymbol{x} = F_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{l})$$

$$\boldsymbol{o} = G_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{l}_N)$$

equilibrium configuration!

Non-positional graphs

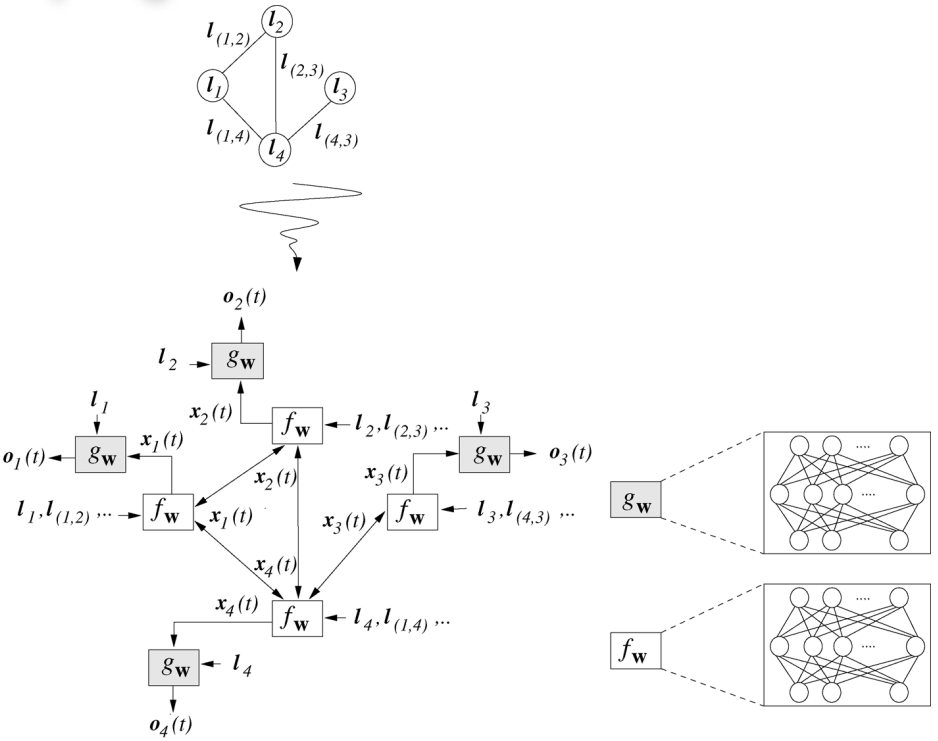
diffusion-based computation similar to PageRank



$$\boldsymbol{x}_n = \sum_{u \in \mathrm{ne}[n]} h_{\boldsymbol{w}}(\boldsymbol{l}_n, \boldsymbol{l}_{(n,u)}, \boldsymbol{x}_u, \boldsymbol{l}_u), \qquad n \in \boldsymbol{N}$$

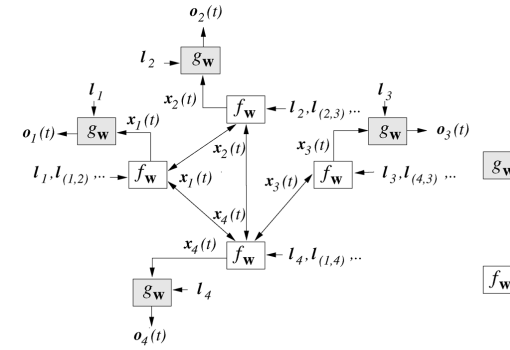
permutation-independent

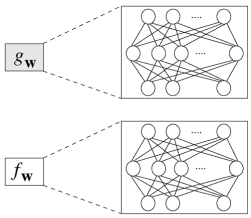
Graph compiling …



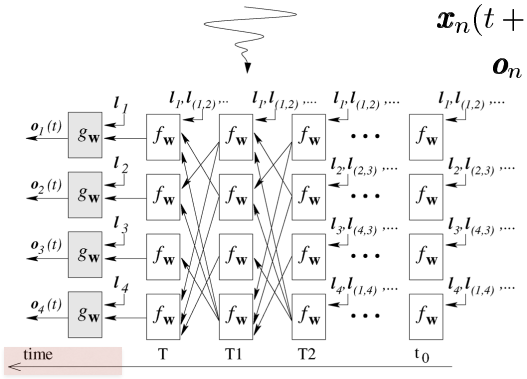
How we get the equilibrium points?

Relaxation to an equilibrium



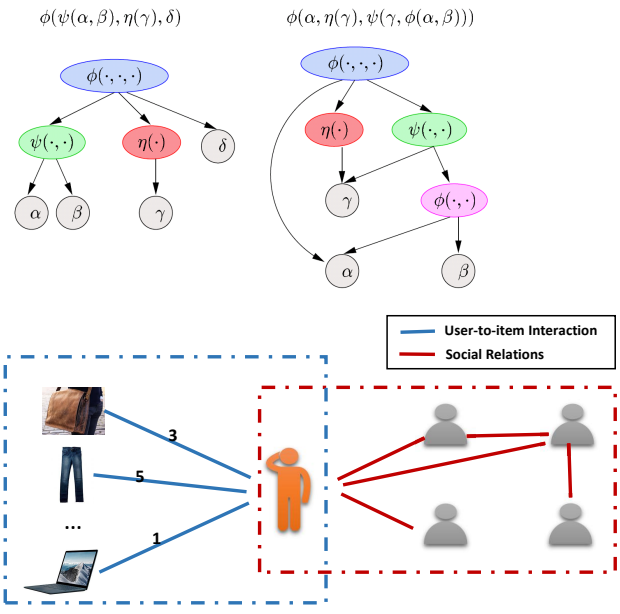


$$\boldsymbol{x}_n(t+1) = f_{\boldsymbol{w}}(\boldsymbol{l}_n, \boldsymbol{l}_{\mathrm{co}[n]}, \boldsymbol{x}_{\mathrm{ne}[n]}(t), \boldsymbol{l}_{\mathrm{ne}[n]})$$
$$\boldsymbol{o}_n(t) = g_{\boldsymbol{w}}(\boldsymbol{x}_n(t), \boldsymbol{l}_n), \qquad n \in \boldsymbol{N}.$$

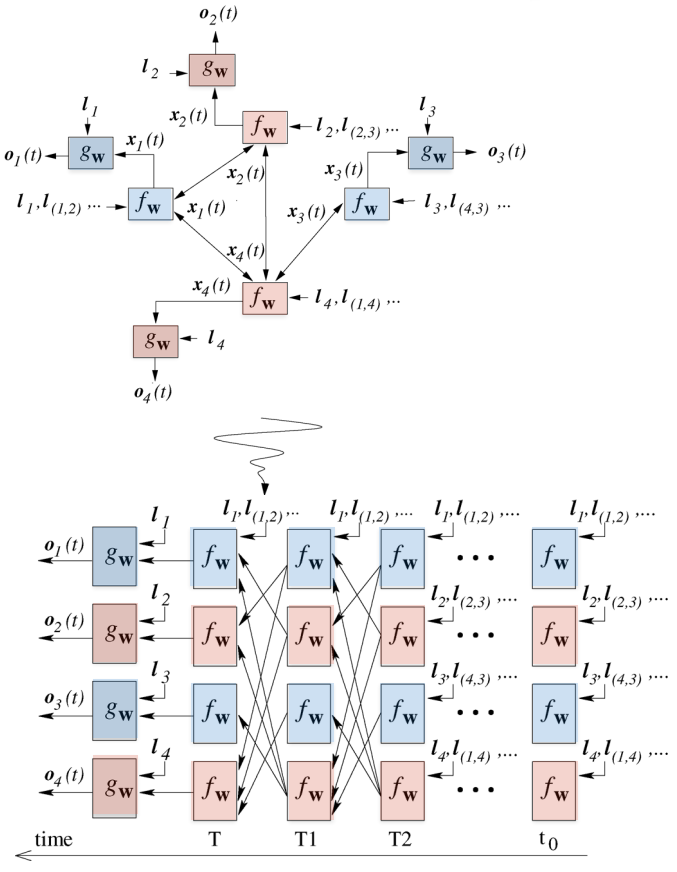


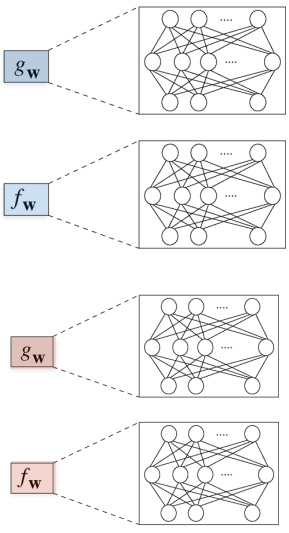
$$\boldsymbol{x}(t+1) = F_{\boldsymbol{w}}(\boldsymbol{x}(t), \boldsymbol{l})$$

WHAT IF NODES ARE OF DIFFERENT TYPE?



Nodes of different types





weight constraints on red/blue blocks are different!

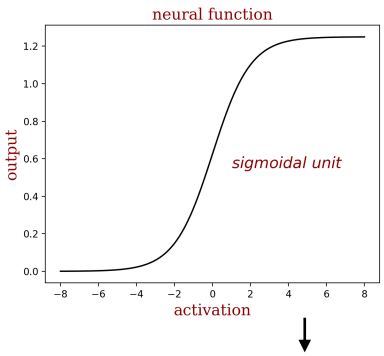
Graph Attention Networks
Node/arc types are generated on-line

How we get the equilibrium?

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

$$\boldsymbol{x}_n(t+1) = f_{\boldsymbol{w}}(\boldsymbol{l}_n, \boldsymbol{l}_{\mathrm{co}[n]}, \boldsymbol{x}_{\mathrm{ne}[n]}(t), \boldsymbol{l}_{\mathrm{ne}[n]})$$
$$\boldsymbol{o}_n(t) = g_{\boldsymbol{w}}(\boldsymbol{x}_n(t), \boldsymbol{l}_n), \qquad n \in \boldsymbol{N}.$$

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



When do we reach an equilibrium?

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

Contraction map!

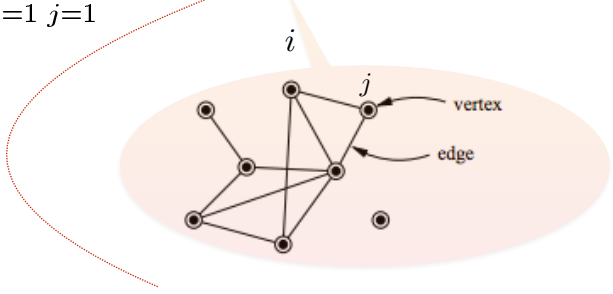
Supervised Learning

$$\mathcal{L} = \{(\boldsymbol{G}_i, n_{i,j}, \boldsymbol{t}_{i,j})|, \boldsymbol{G}_i = (\boldsymbol{N}_i, \boldsymbol{E}_i) \in \mathcal{G};$$ no of graphs

$$n_{i,j} \in \boldsymbol{N}_i; \boldsymbol{t}_{i,j} \in I\!R^m, 1 \leq i \leq p, 1 \leq j \leq q_i\}$$

target — no of nodes

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



no supervision is required on all the vertexes!

Can we learn by classic gradient descent?

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

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

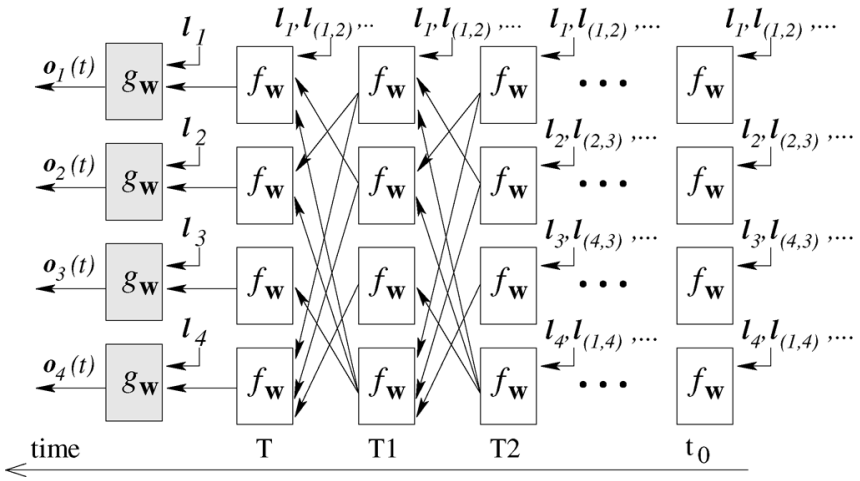
$$e_{\boldsymbol{w}} = \sum_{i=1}^{p} \sum_{j=1}^{q_i} (\boldsymbol{t}_{i,j} - \varphi_{\boldsymbol{w}}(\boldsymbol{G}_i, n_{i,j}))^2$$

$F_{\boldsymbol{w}}$ is a contraction map

Theorem 1 (Differentiability): Let $F_{\boldsymbol{w}}$ and $G_{\boldsymbol{w}}$ be the global transition and the global output functions of a GNN, respectively. If $F_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{l})$ and $G_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{l_N})$ are continuously differentiable w.r.t. $\boldsymbol{x}$ and $\boldsymbol{w}$, then $\varphi_{\boldsymbol{w}}$ 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 $\boldsymbol{w}$;
$\boldsymbol{x}$=**Forward**($\boldsymbol{w}$);
repeat
$\frac{\partial e_{\boldsymbol{w}}}{\partial \boldsymbol{w}}$=BACKWARD($\boldsymbol{x}, \boldsymbol{w}$);
$\boldsymbol{w}$=$\boldsymbol{w} - \lambda \cdot \frac{\partial e_{\boldsymbol{w}}}{\partial \boldsymbol{w}}$;
$\boldsymbol{x}$=FORWARD($\boldsymbol{w}$);
until (a stopping criterion);
return $\boldsymbol{w}$;
end

gradient descent

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

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

BACKWARD($\boldsymbol{x}$,$\boldsymbol{w}$)
$\boldsymbol{o} = G_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{l}_N)$;
$\boldsymbol{A} = \frac{\partial F_{\boldsymbol{w}}}{\partial \boldsymbol{x}}(\boldsymbol{x}, \boldsymbol{l})$;
$\boldsymbol{b} = \frac{\partial e_{\boldsymbol{w}}}{\partial \boldsymbol{o}} \cdot \frac{\partial G_{\boldsymbol{w}}}{\partial \boldsymbol{x}}(\boldsymbol{x}, \boldsymbol{l}_N)$;
initialize $\boldsymbol{z}(0)$, t=0;
repeat
$\boldsymbol{z}(t) = \boldsymbol{z}(t+1) \cdot \boldsymbol{A} + \boldsymbol{b}$;
t=$t - 1$;
until $\|\boldsymbol{z}(t-1) - \boldsymbol{z}(t)\| \leq \varepsilon_b$;
$\boldsymbol{c} = \frac{\partial e_{\boldsymbol{w}}}{\partial \boldsymbol{o}} \cdot \frac{\partial G_{\boldsymbol{w}}}{\partial \boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{l}_N)$;
$\boldsymbol{d} = \boldsymbol{z}(t) \cdot \frac{\partial F_{\boldsymbol{w}}}{\partial \boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{l})$;
$\frac{\partial e_{\boldsymbol{w}}}{\partial \boldsymbol{w}} = \boldsymbol{c} + \boldsymbol{d}$;
return $\frac{\partial e_{\boldsymbol{w}}}{\partial \boldsymbol{w}}$;
end

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

Backpropagation

Formal proof

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

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

$$\frac{\partial e_{\boldsymbol{w}}}{\partial \boldsymbol{w}} = \frac{\partial e_{\boldsymbol{w}}}{\partial \boldsymbol{o}} \cdot \frac{\partial G_{\boldsymbol{w}}}{\partial \boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{l_N}) + \boldsymbol{z} \cdot \frac{\partial F_{\boldsymbol{w}}}{\partial \boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{l}) \qquad (8)$$

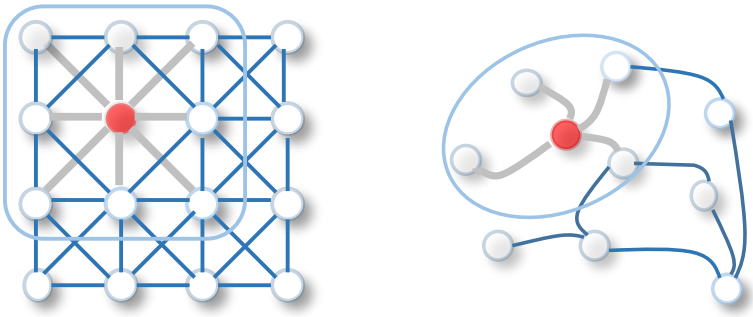
holds, where $\boldsymbol{x}$ is the stable state of the GNN.

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



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

(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

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

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

$$\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T, \text{ where } \mathbf{U} = [\mathbf{u_0}, \mathbf{u_1}, \cdots, \mathbf{u_{n-1}}] \in \mathbf{R}^{n \times n}$$

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

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

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

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