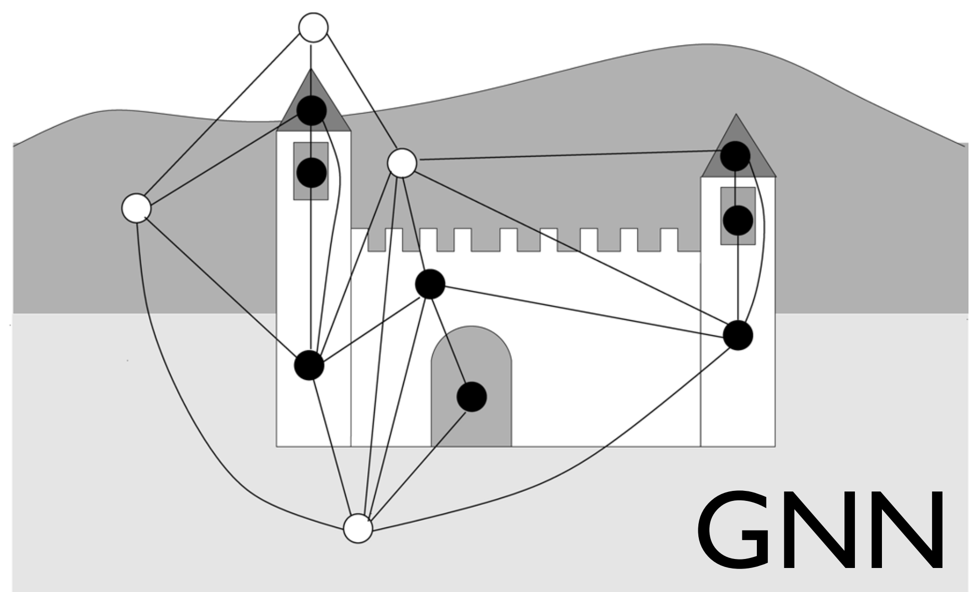
LOCAL PROPAGATION IN GRAPH NEURAL NETWORKS



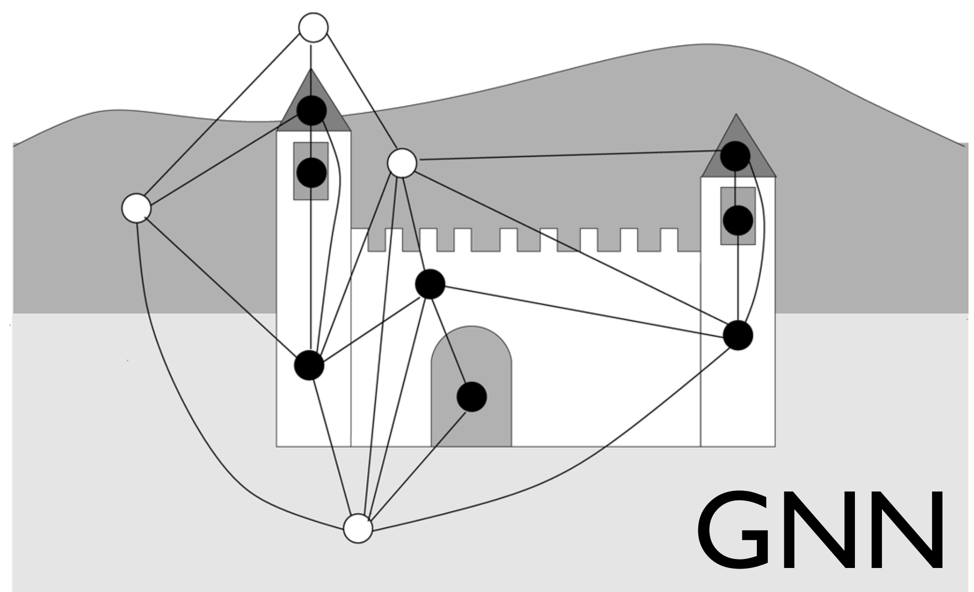
Marco Gori
Department of Information Engineering
and Mathematics

GbR 2019

OUTLINE

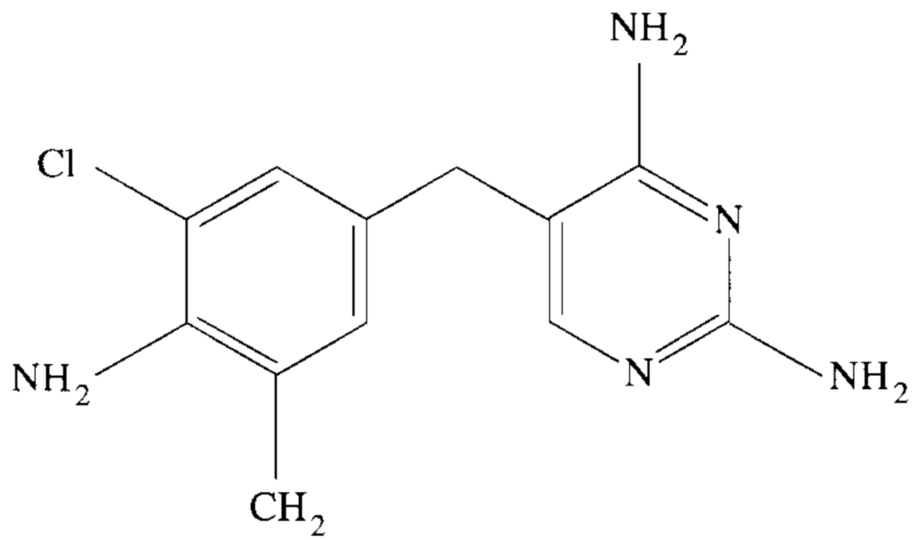
- Graphical domains
- Graph Neural Networks (GNN) and recent evolutions
- Local Propagation in NN and in GNN
- The perspective of "learning from constraints"

GRAPHICAL DOMAINS

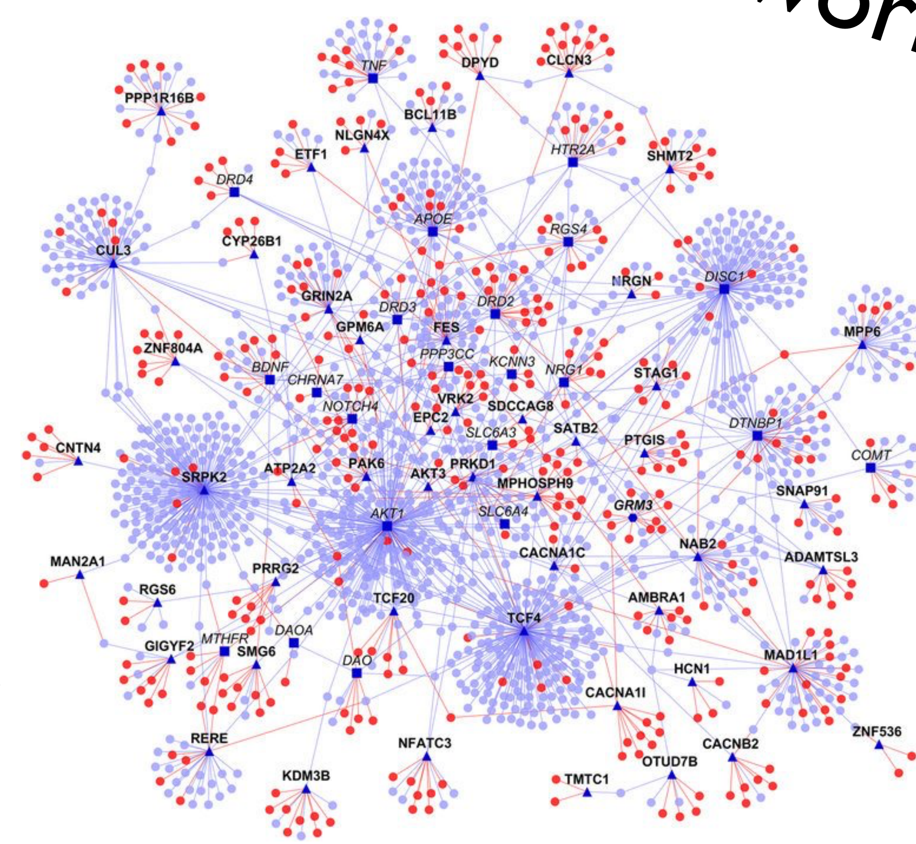


GbR 2019

physicochemical behavior



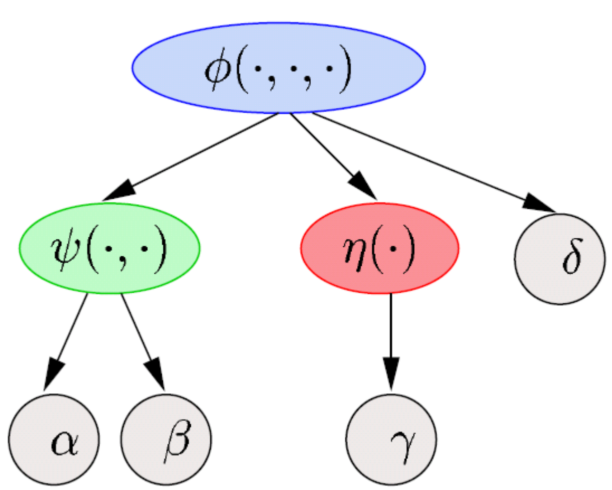
Protein Interaction Network



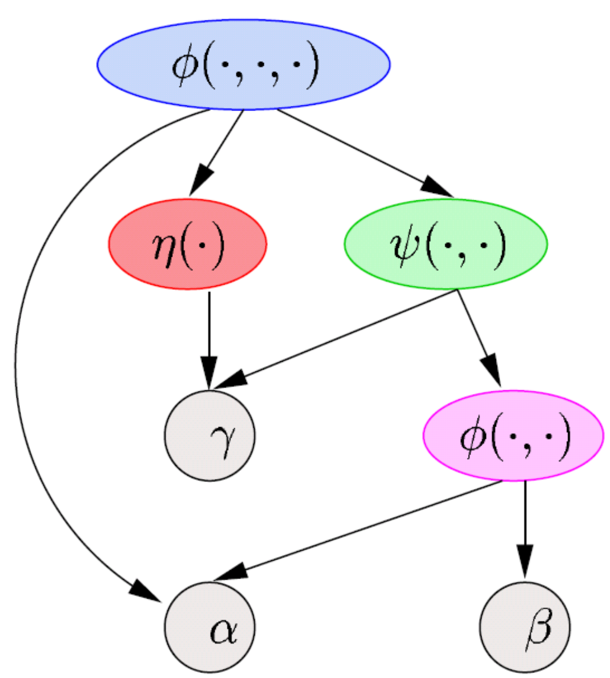
What are the features? The atoms, the bonds?

On the truth of logic statements

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

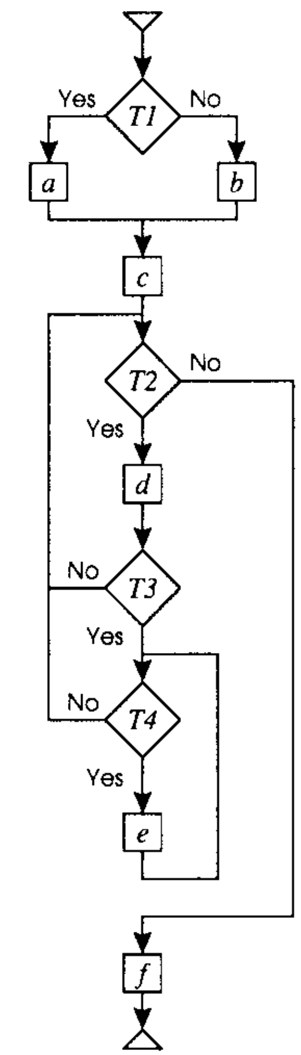


$\phi(\alpha, \eta(\gamma), \psi(\gamma, \phi(\alpha, \beta)))$

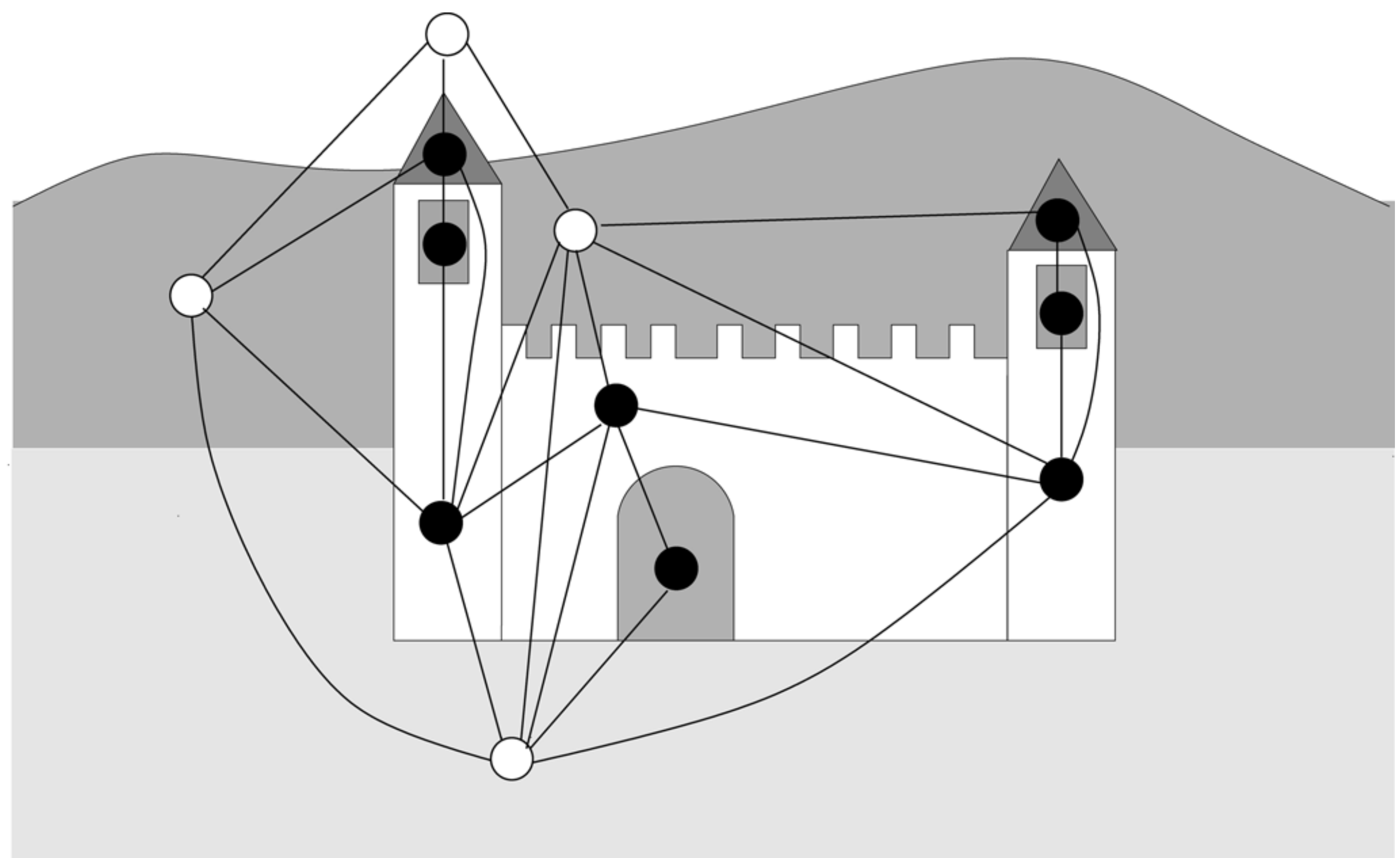


Program behavior

```
program  name (list);
var
. . .
begin
if  T1 then
        a
else
        b;
c;
while T2 do
        begin
        d;
        if  T3 then
        while T4 do
                e
        end;
f
end.
```

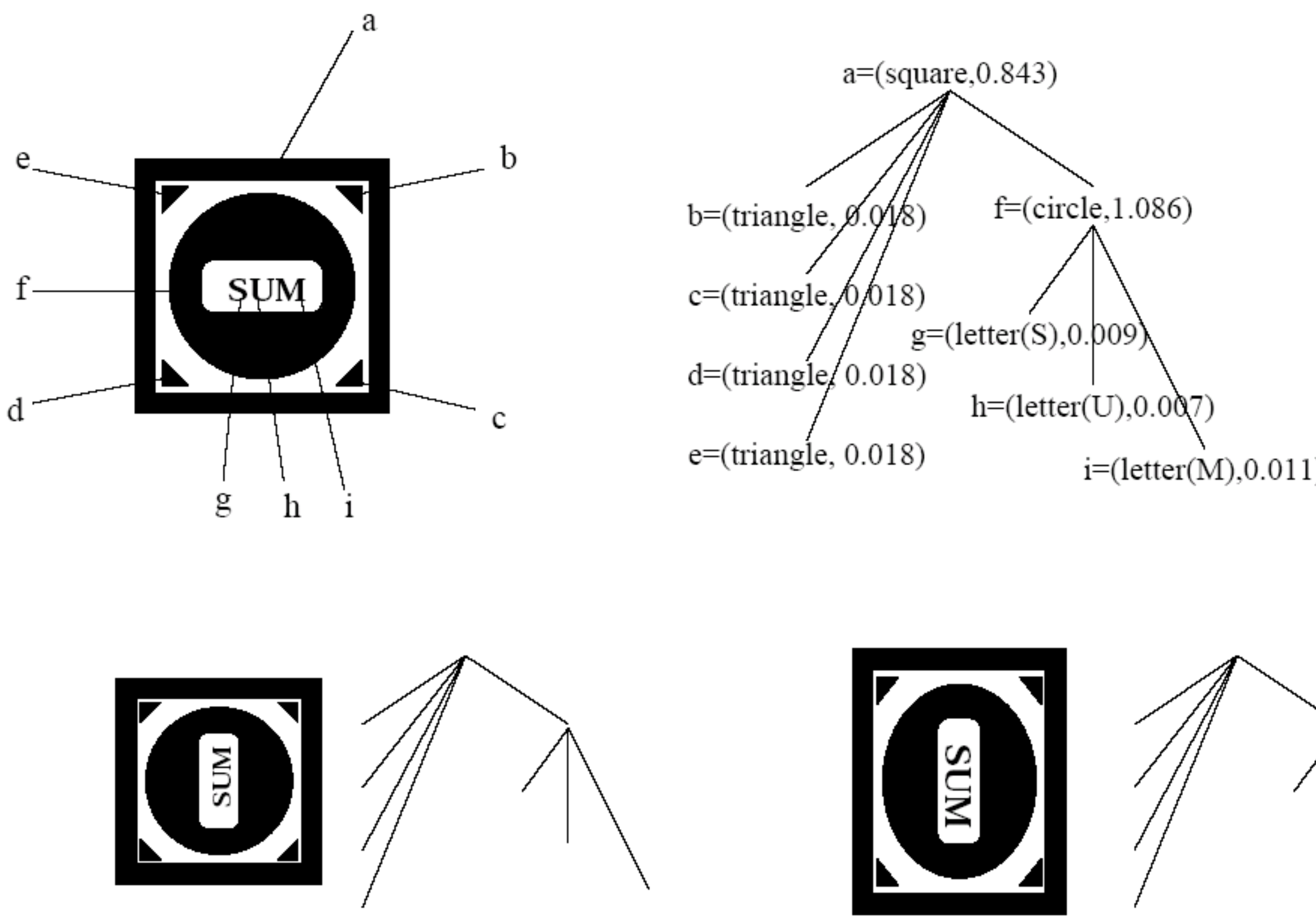


Structured patterns …



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

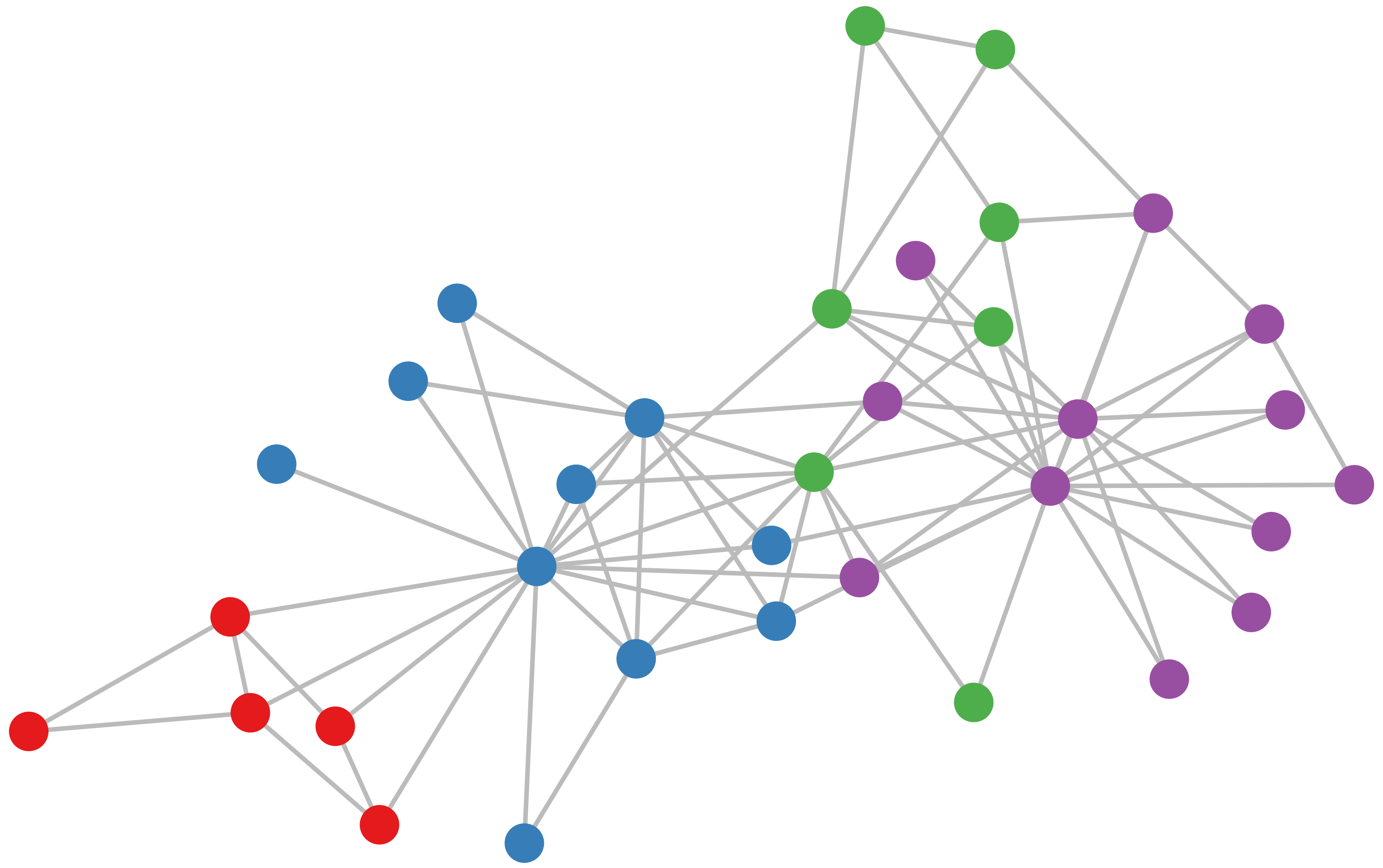
Yet another one …



Another example: XY-trees for Document Analysis and Recognition

Social nets

here we need to make prediction at node level!



Formulation of Learning Tasks

$\tau(\boldsymbol{G}, n)$

node focussed
computation

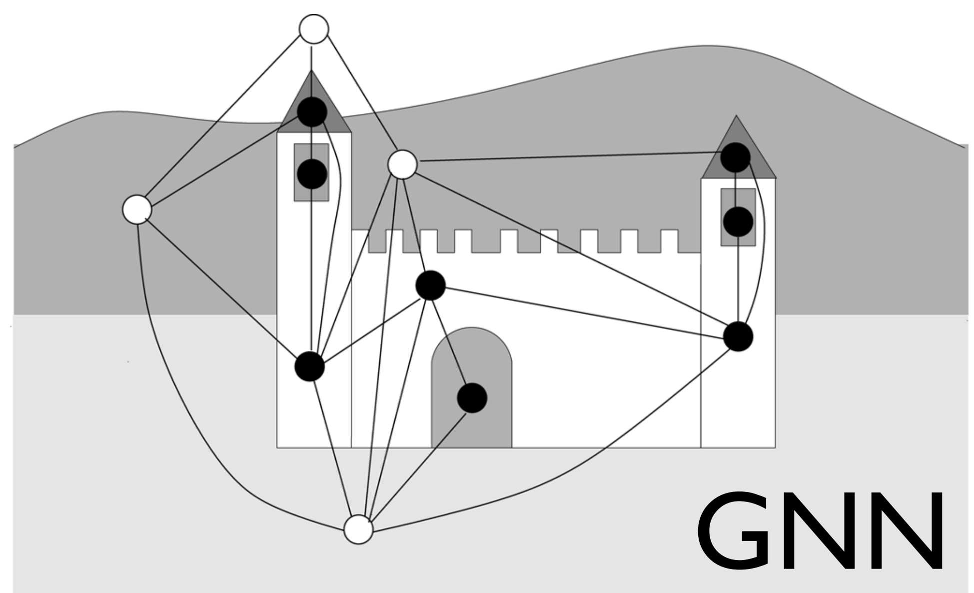
$\tau(\boldsymbol{G})$

graphs as single patterns:
classification, regression

GRAPH NEURAL NETS

METHODS AND HISTORICAL ISSUES

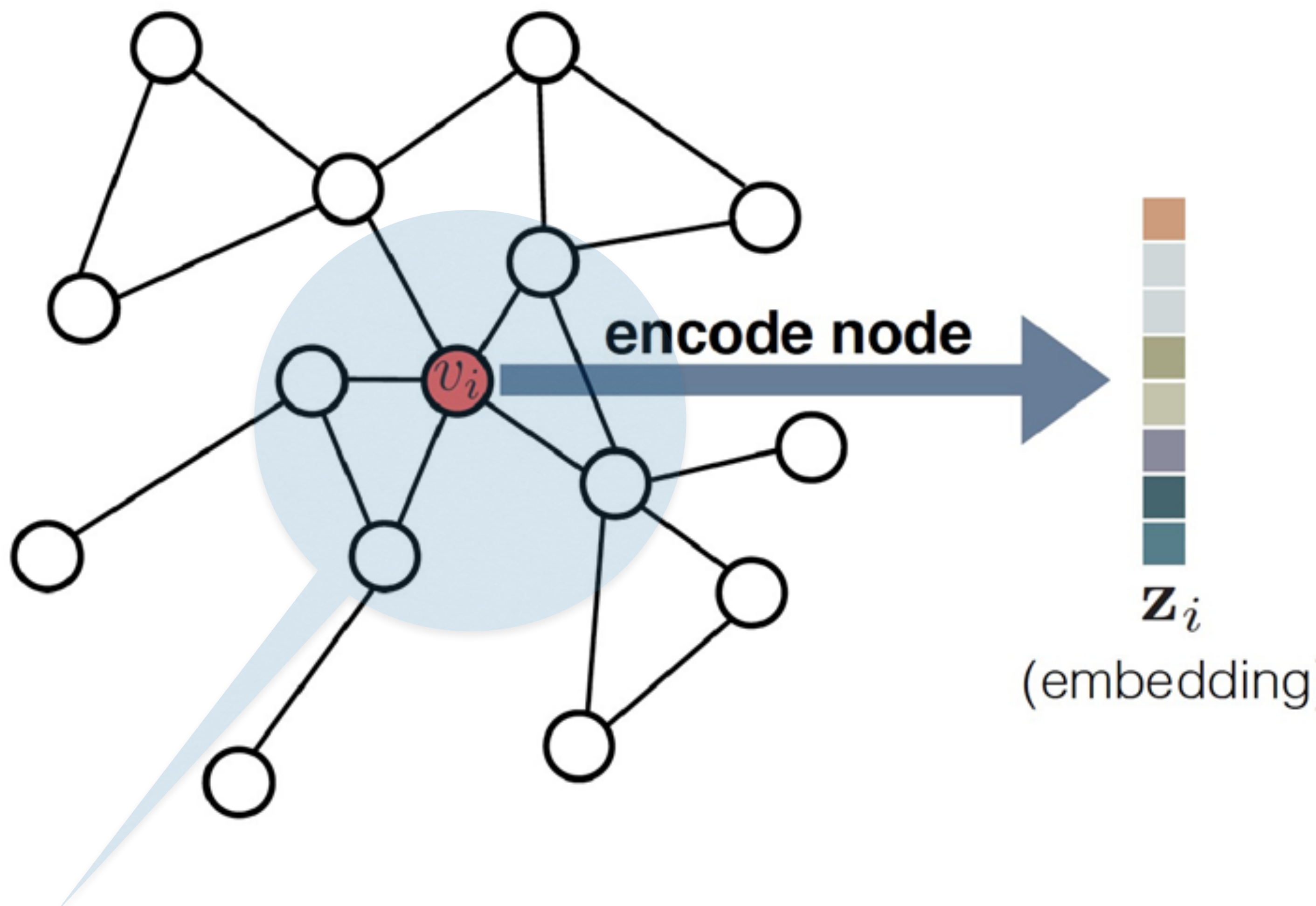
Where do they come from?



"diffusion" machines …

GbR 2019

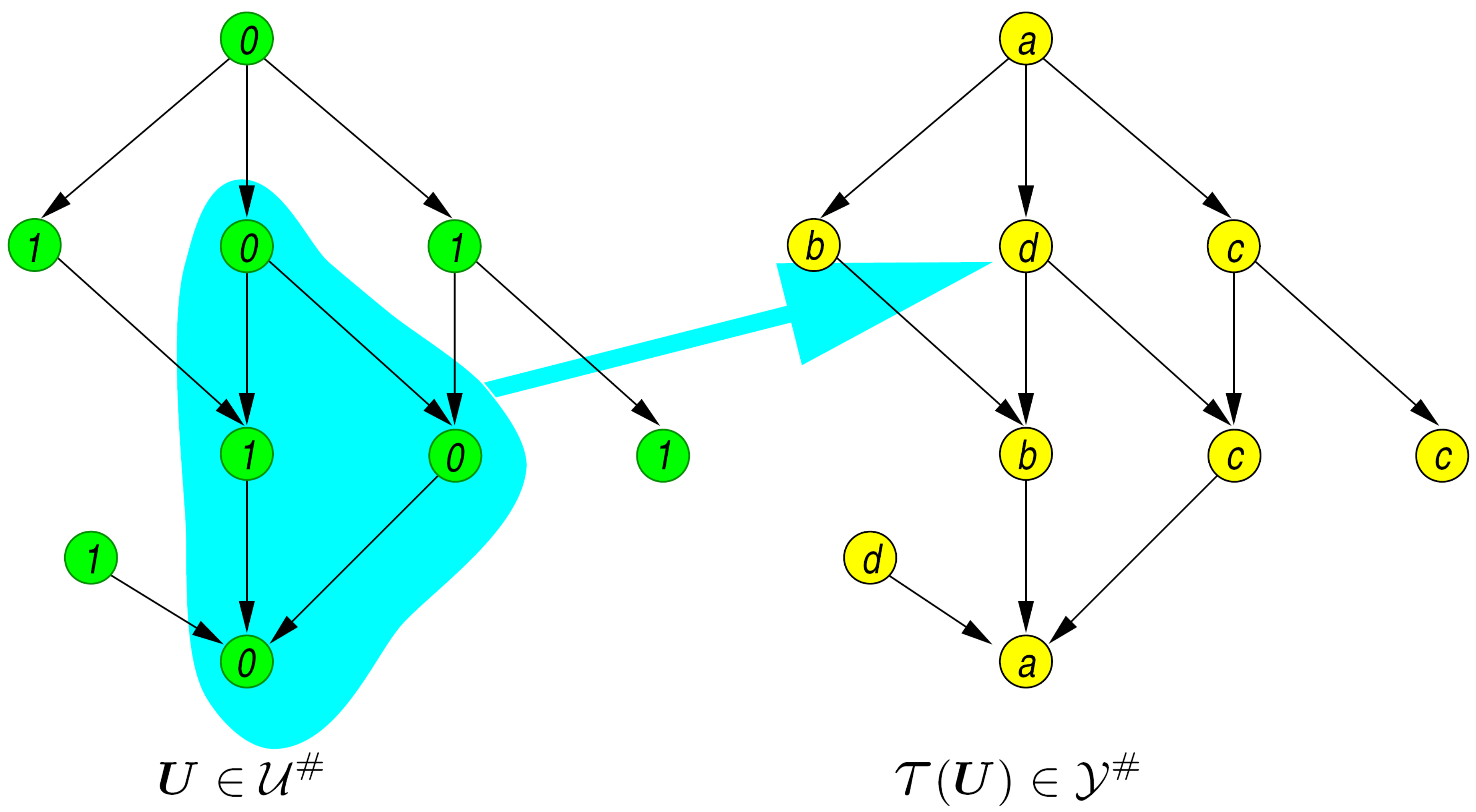
Node-based encoding



we could choose an “appropriate window”

Information diffusion and causality

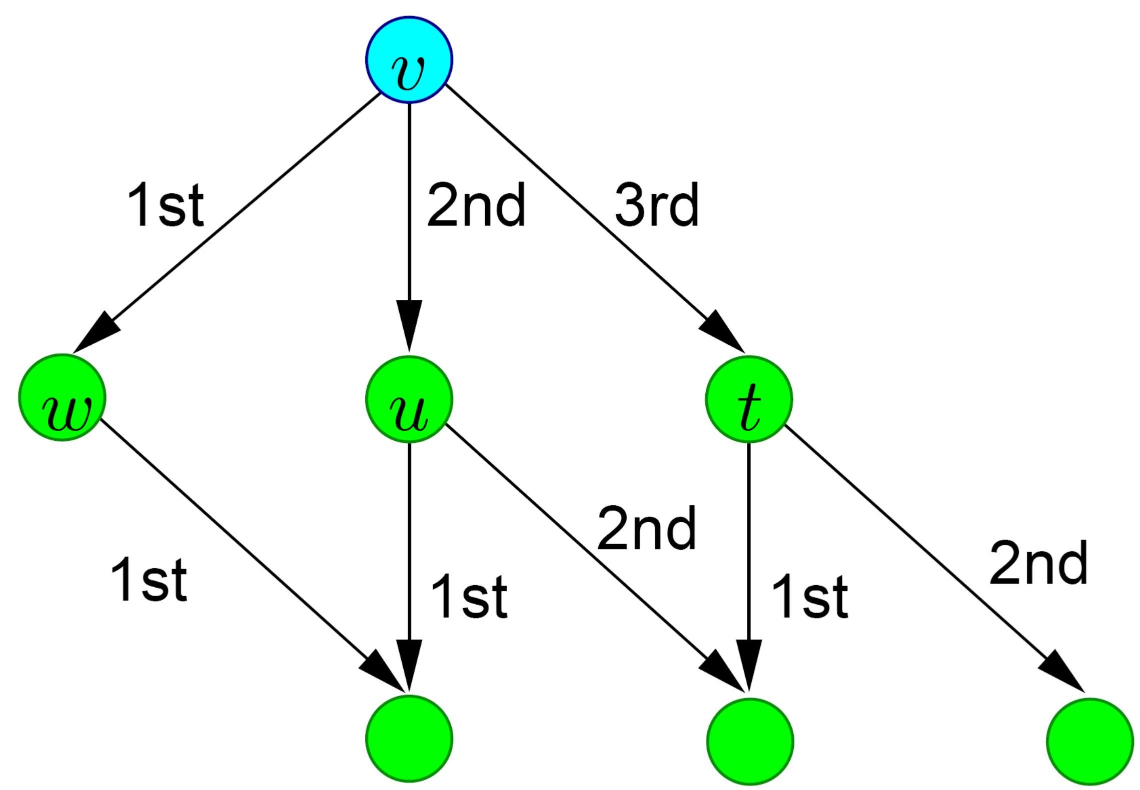
A transduction $\mathcal{T}(\cdot)$ is **causal** if $\forall v \in$ vert$(\boldsymbol{U})$ $\mathcal{T}(\boldsymbol{U})_v$ only depends on the subgraph of $\boldsymbol{U}$ induced by $\{v\} \cup$ de$[v]$.



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



State-based representation

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

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

ordering of children state does matter!

where 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*

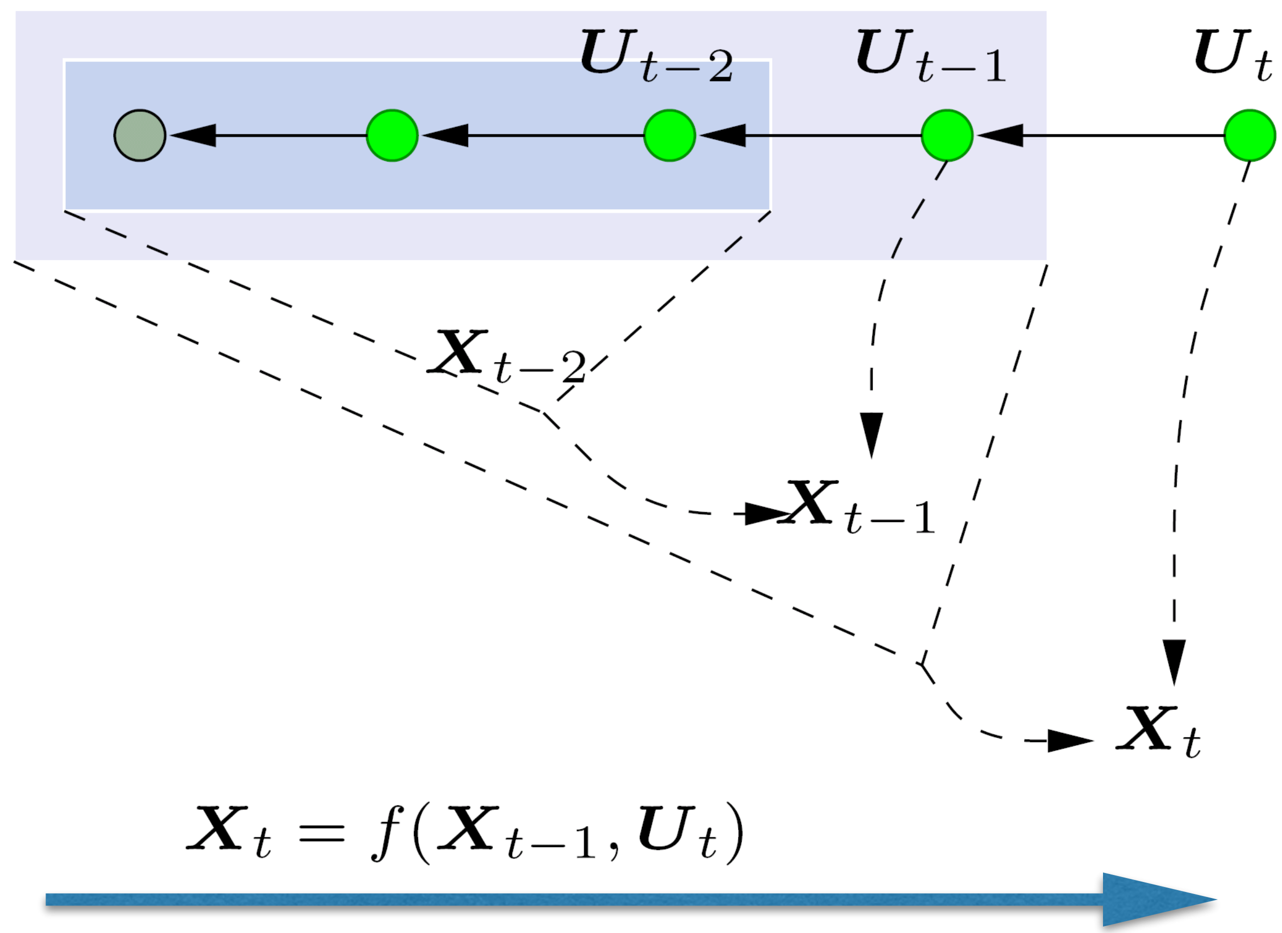
Compare to temporal dynamical systems:

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

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

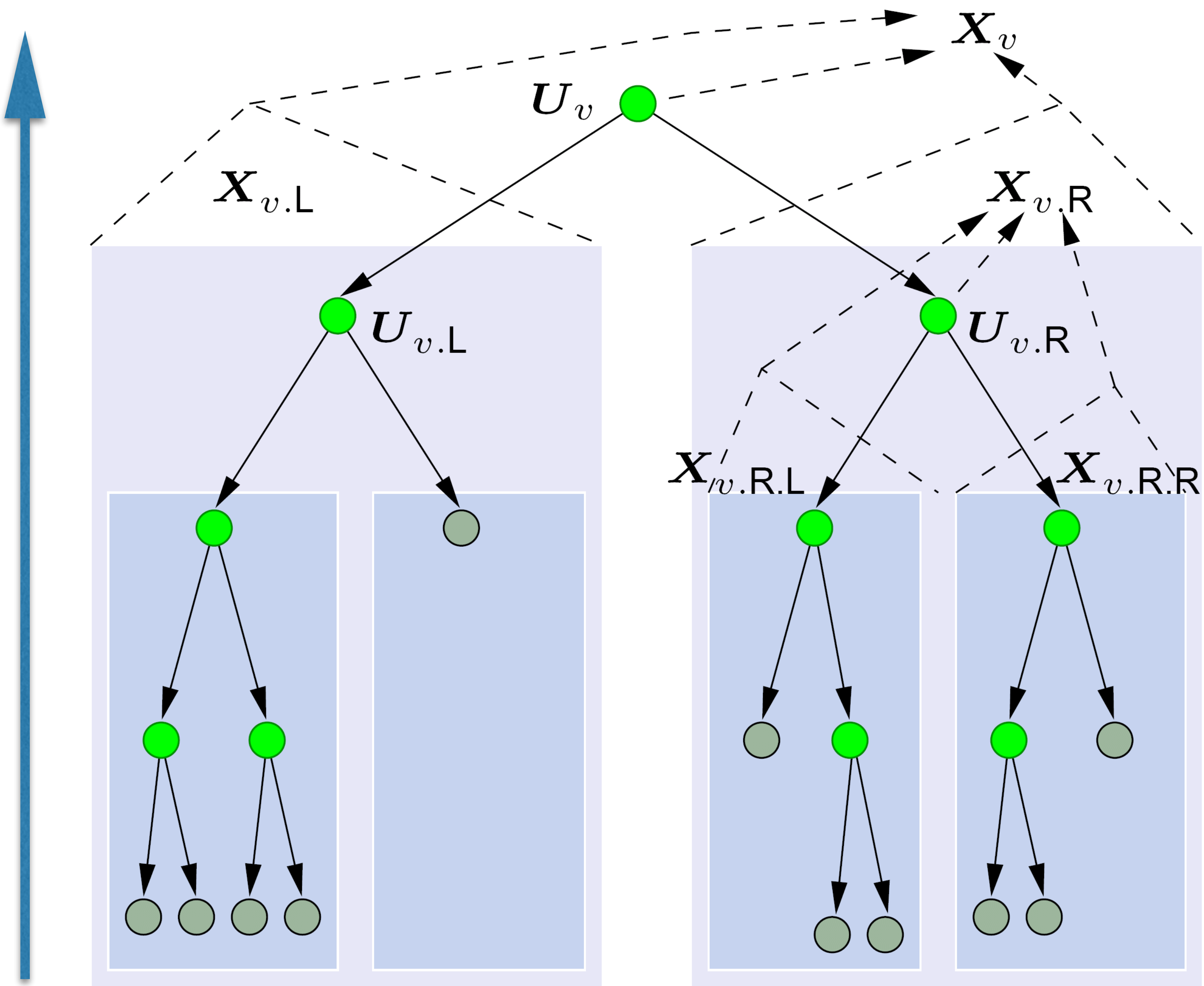
Reduction to sequences



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 …



frontier state if v.R is external

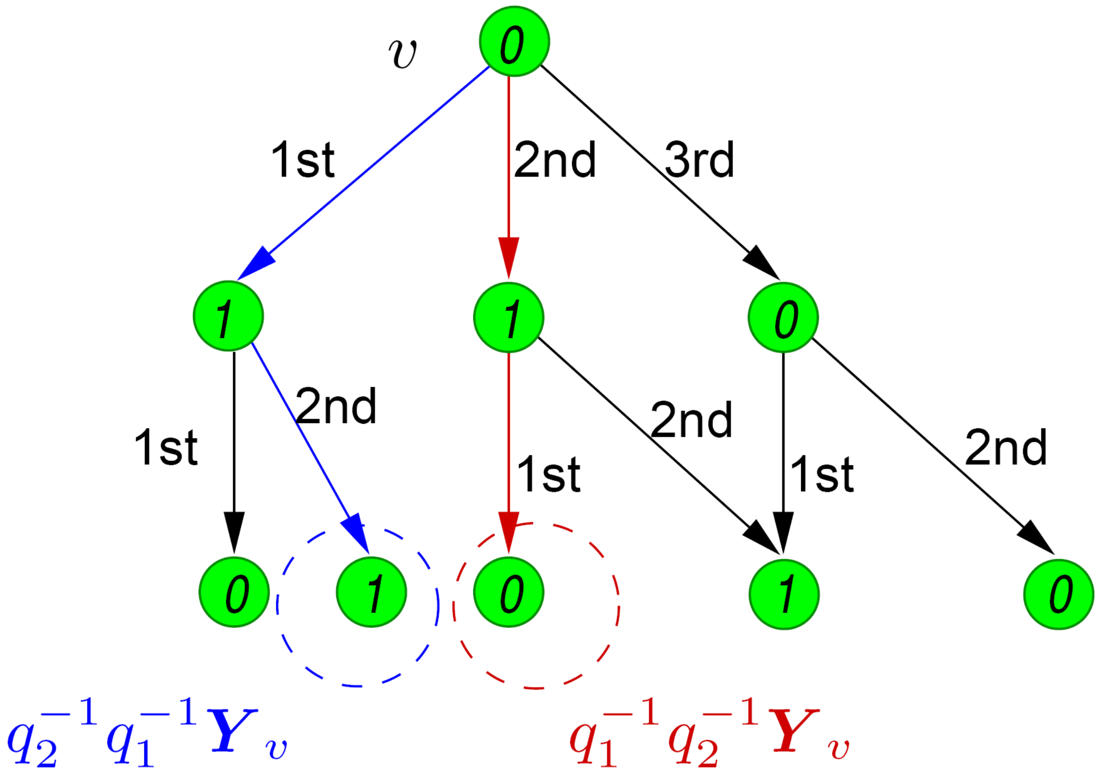
$$\boldsymbol{X}_v = f(\boldsymbol{U}_v, \boldsymbol{X}_{v.\mathsf{L}}, \boldsymbol{X}_{v.\mathsf{R}})$$

frontier state if v.L is external

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:

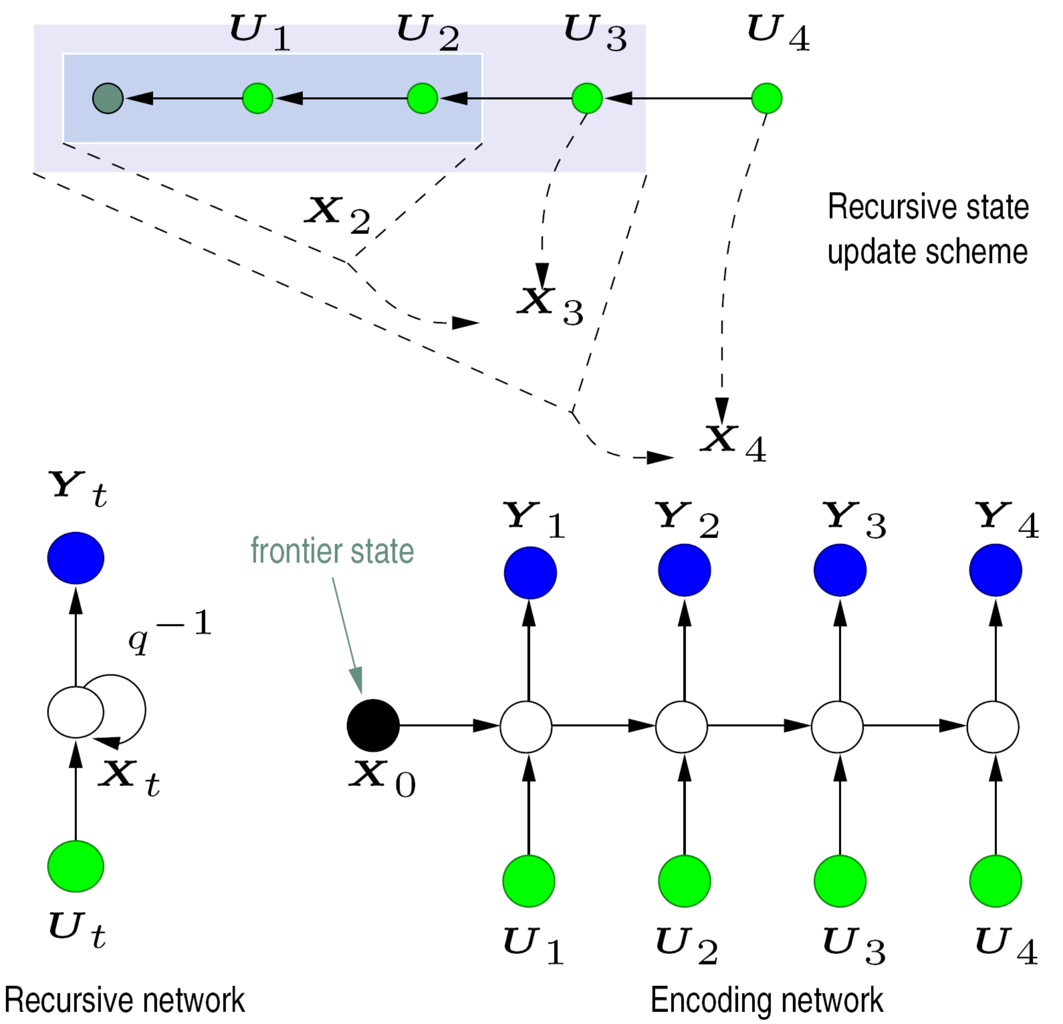


Encoding networks

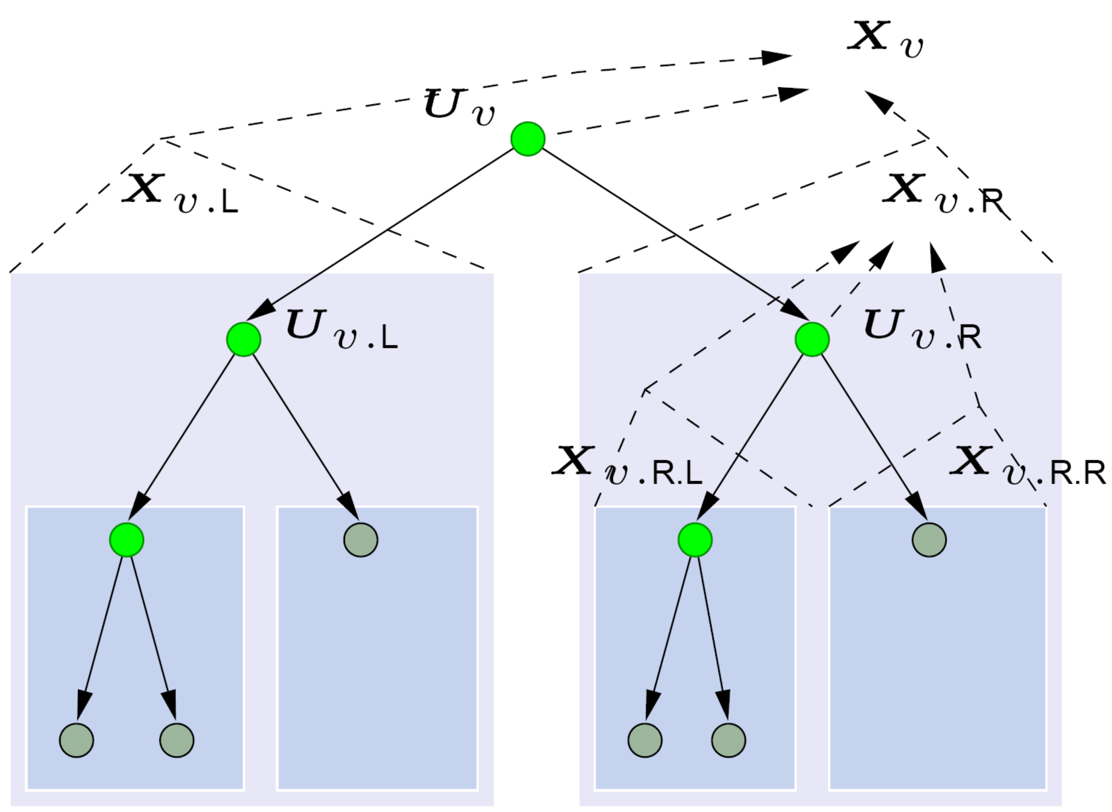
Given a graph $\boldsymbol{U} \in \mathcal{U}^{\#}$ and a recursive transduction $\mathcal{T}$.

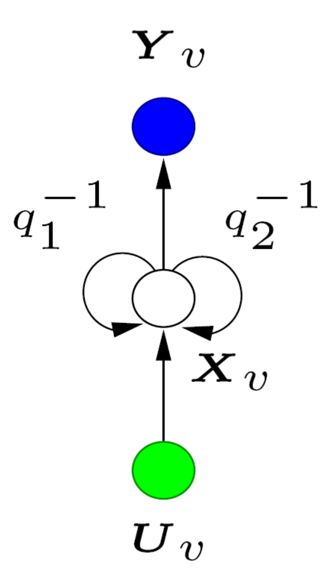
The *encoding network* associated to $\boldsymbol{U}$ and $\mathcal{T}$ is formed by unrolling the recursive network of $\mathcal{T}$ through the input graph $\boldsymbol{U}$.

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

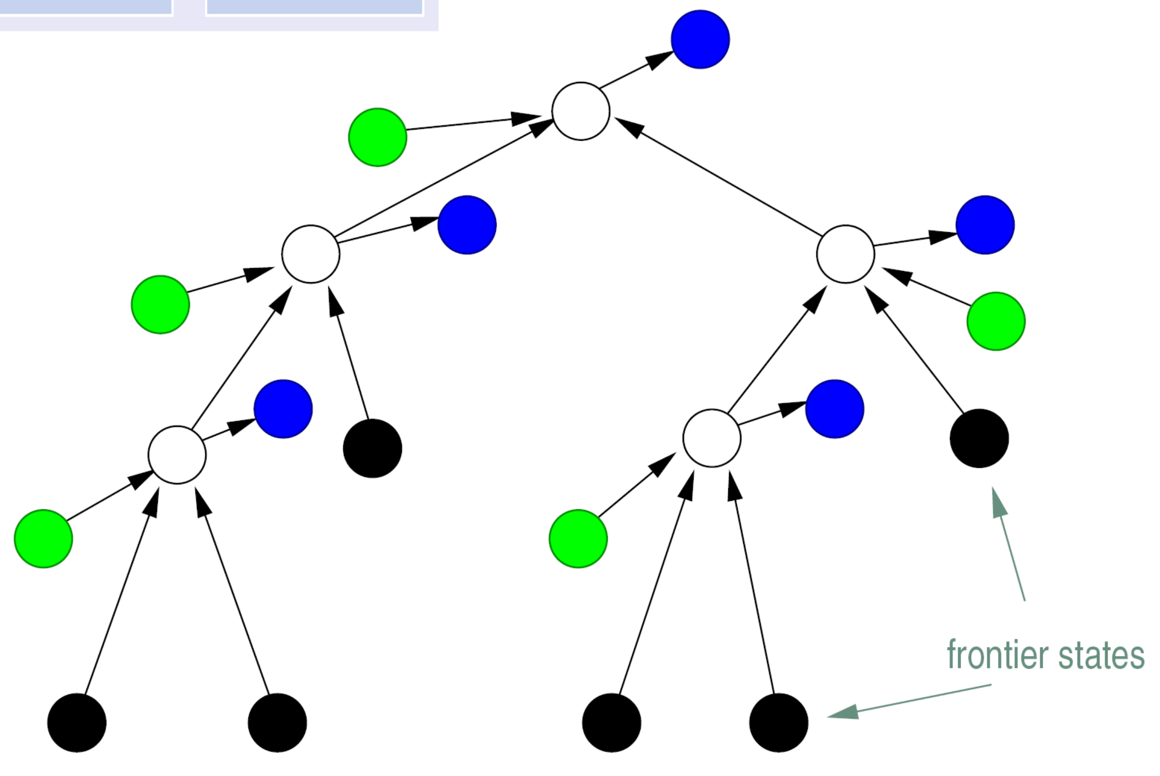


Encoding nets for binary trees



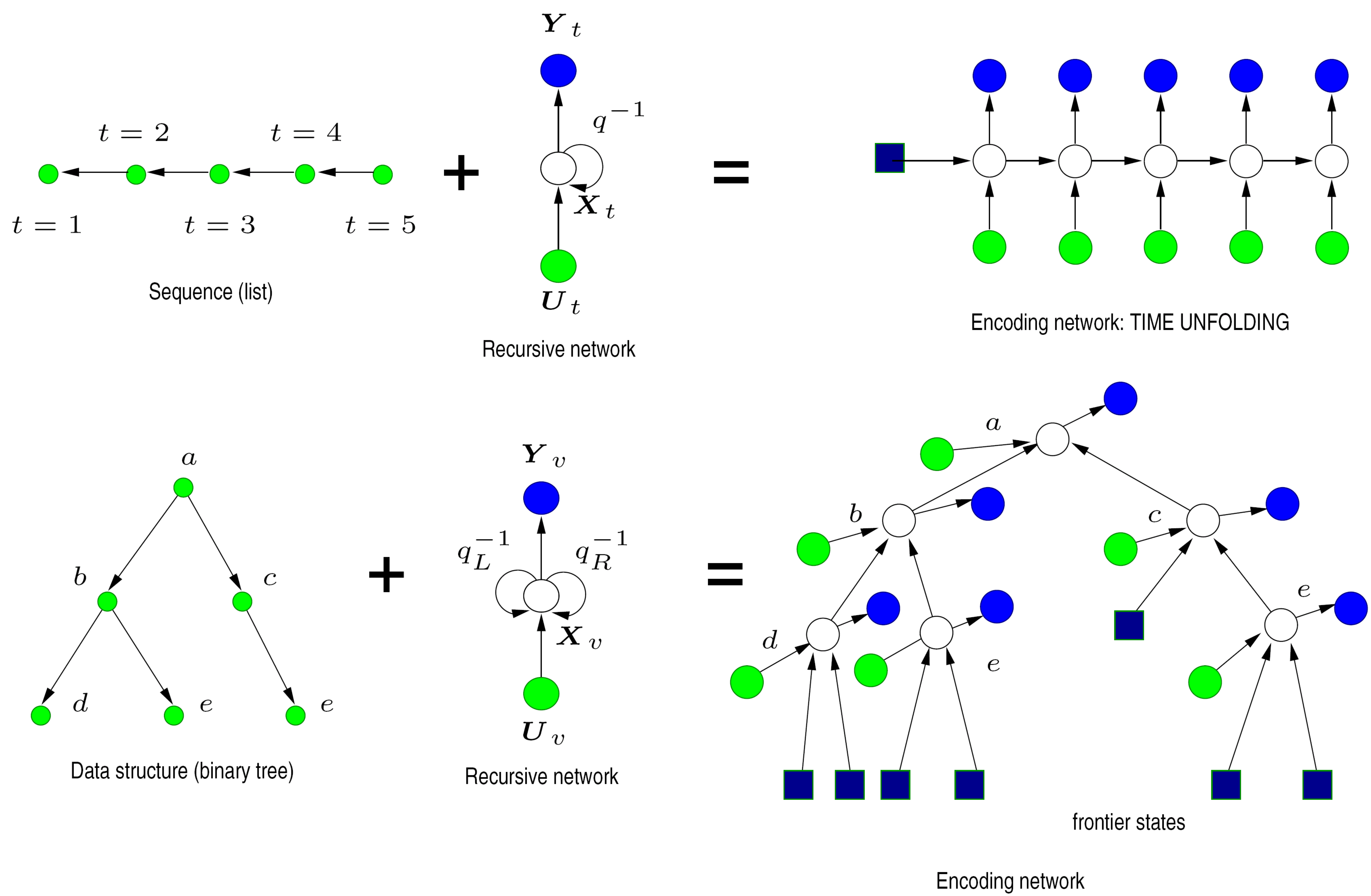


Recursive network



Encoding network

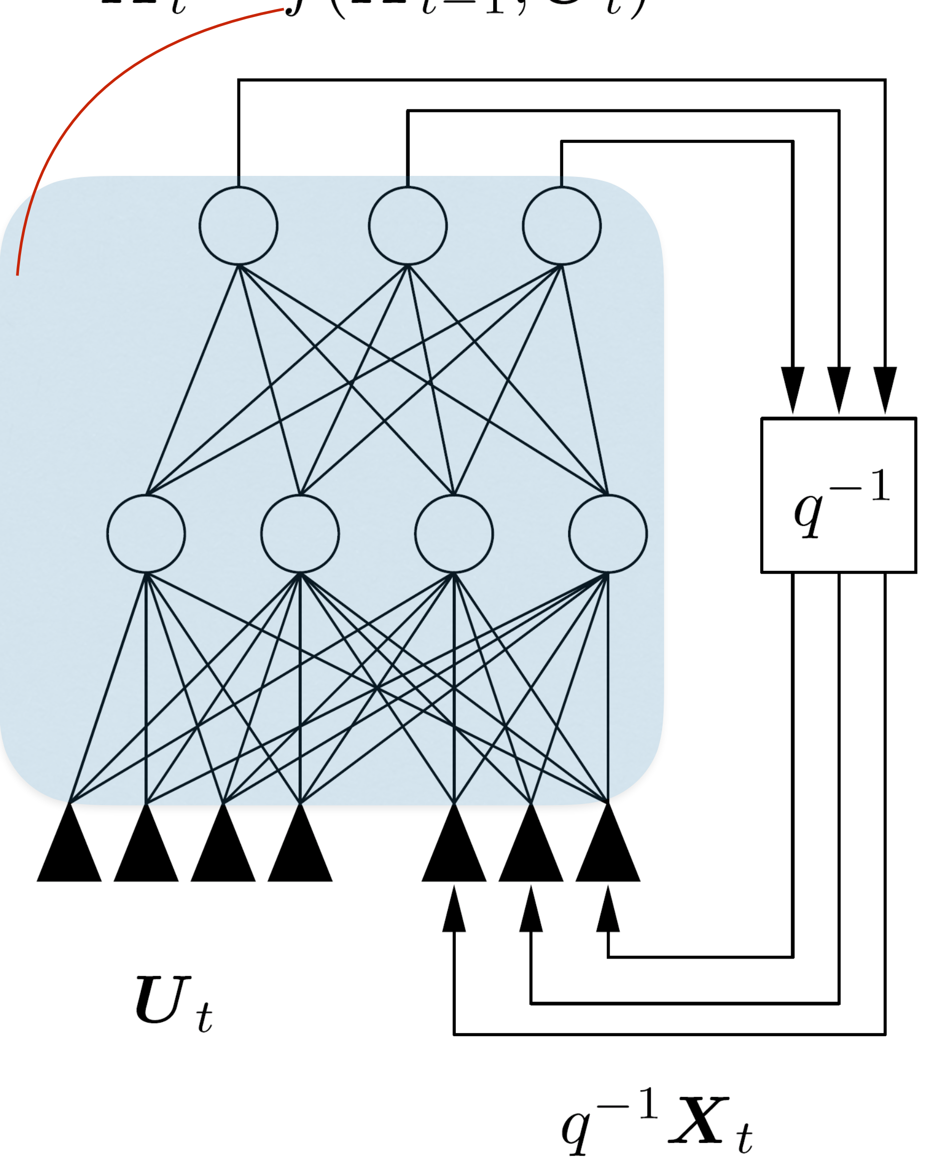
Data structures + recursive nets = encoding nets



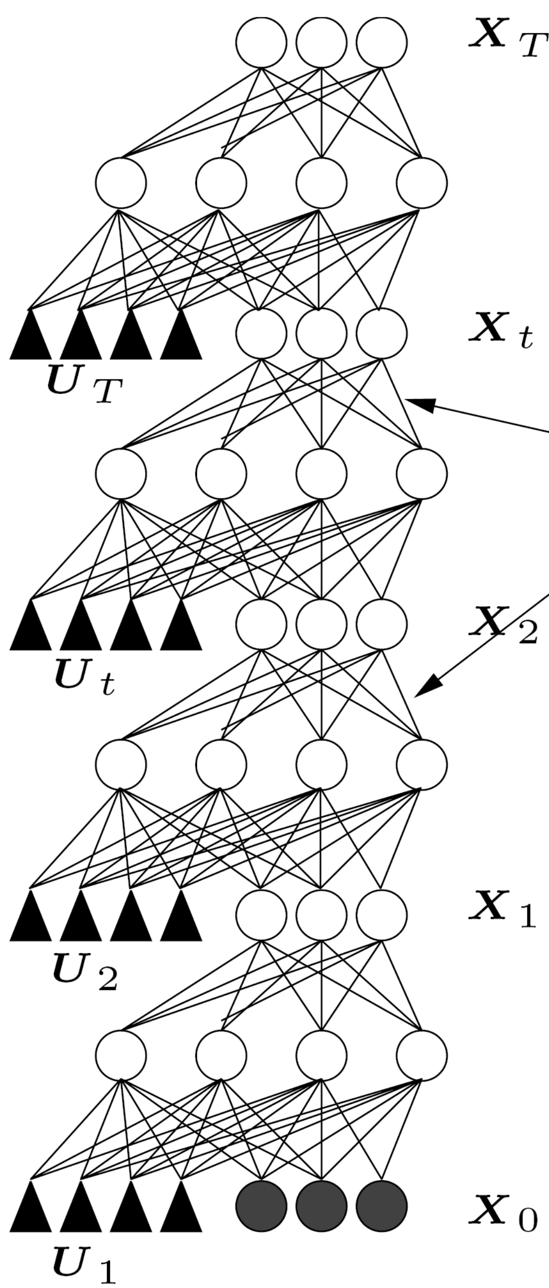
Using neural nets
for sequences …

The state transition function is implemented by a MLP:

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



Time unfolding



The last state is an adaptive encoding of the whole sequence

Weights are shared (replicated)

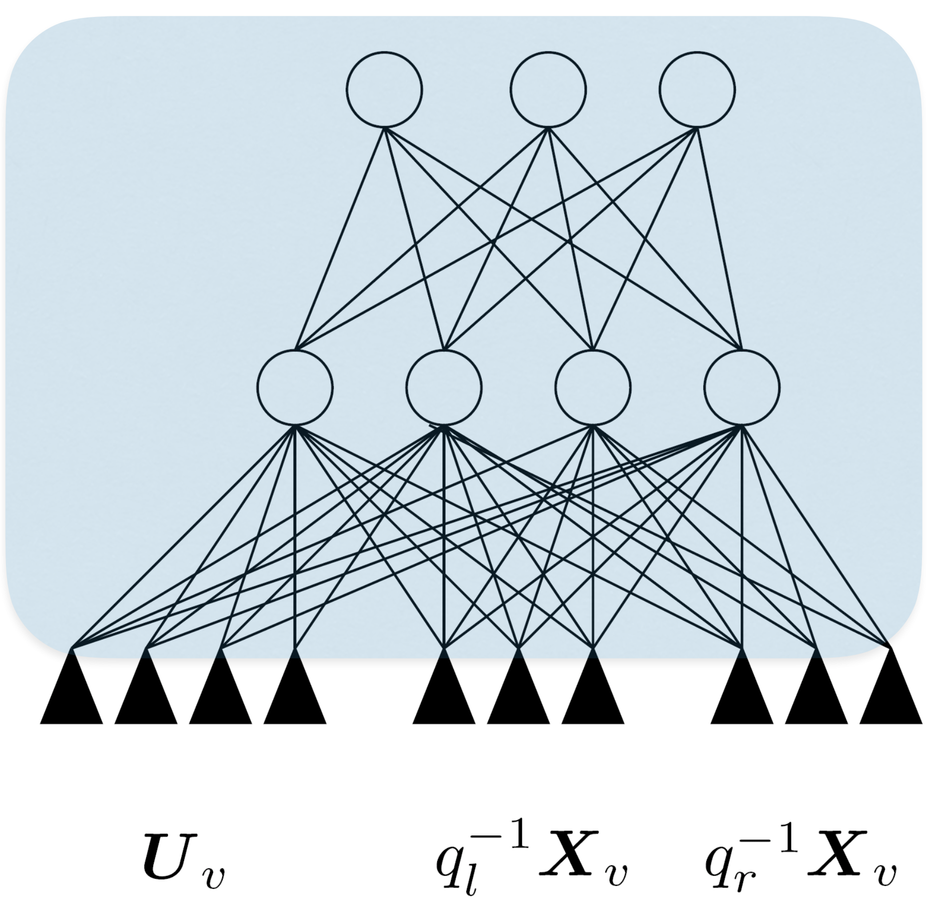
The encoding net has a feedforward structure: Gradient can be computed by Backpropagation (through time)

Using neural nets
for binary trees …

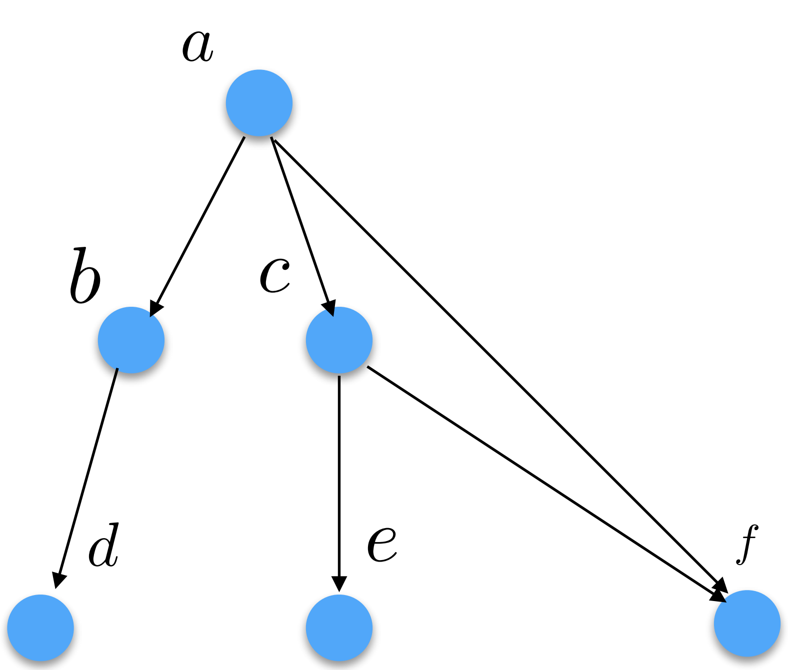
State labels are real vectors: $\boldsymbol{X}_v \in \mathbb{R}^n$.

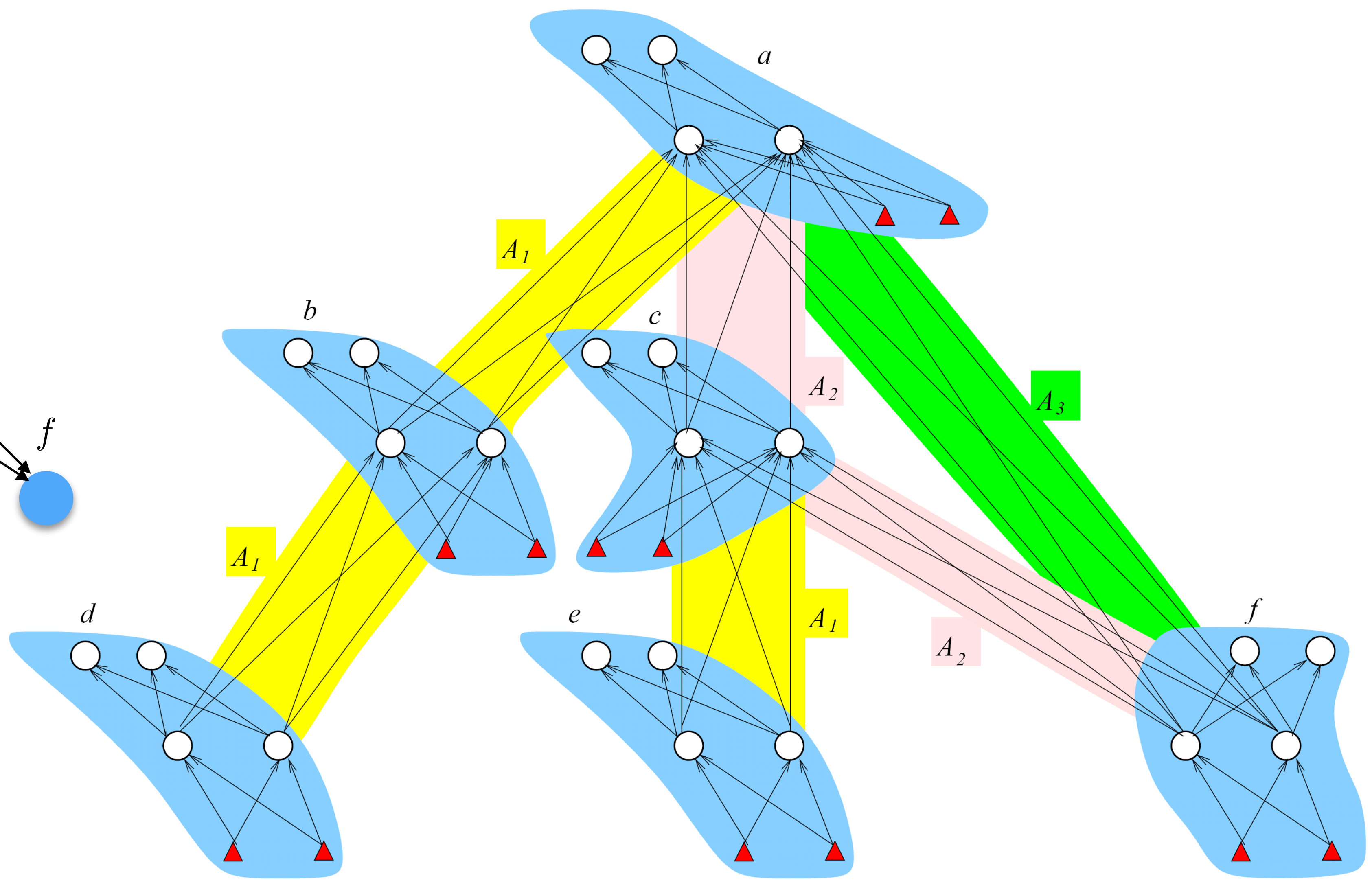
The state transition function is implemented by a MLP (e.g. case of binary trees)

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



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

$\texttt{Initialize}(\Theta)$;

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

$\texttt{Backpropagation}(\mathbf{N})$;

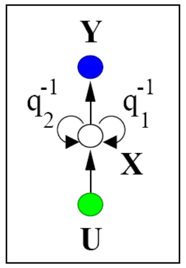
$\texttt{Average}(\Theta)$. ← Weight sharing …

end

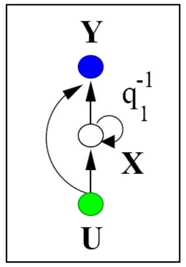
Non-stationary transductions

Linguistic specification of the recursive network

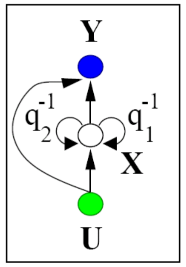
```
Sequence_of_vertices Seq;
Vertex v;
Seq <- sort_vertices_by( dist_from(frontier), <);
foreach(v, Seq) {
  if (dist_from(frontier)<3) then {
        if ( U  in [0.3,0.55] ) then
```




```
;
         else
```




```
;
        }
        else
```

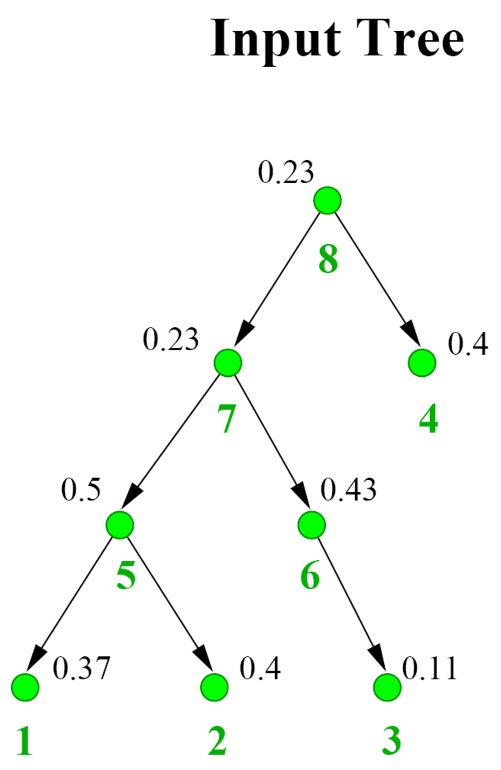


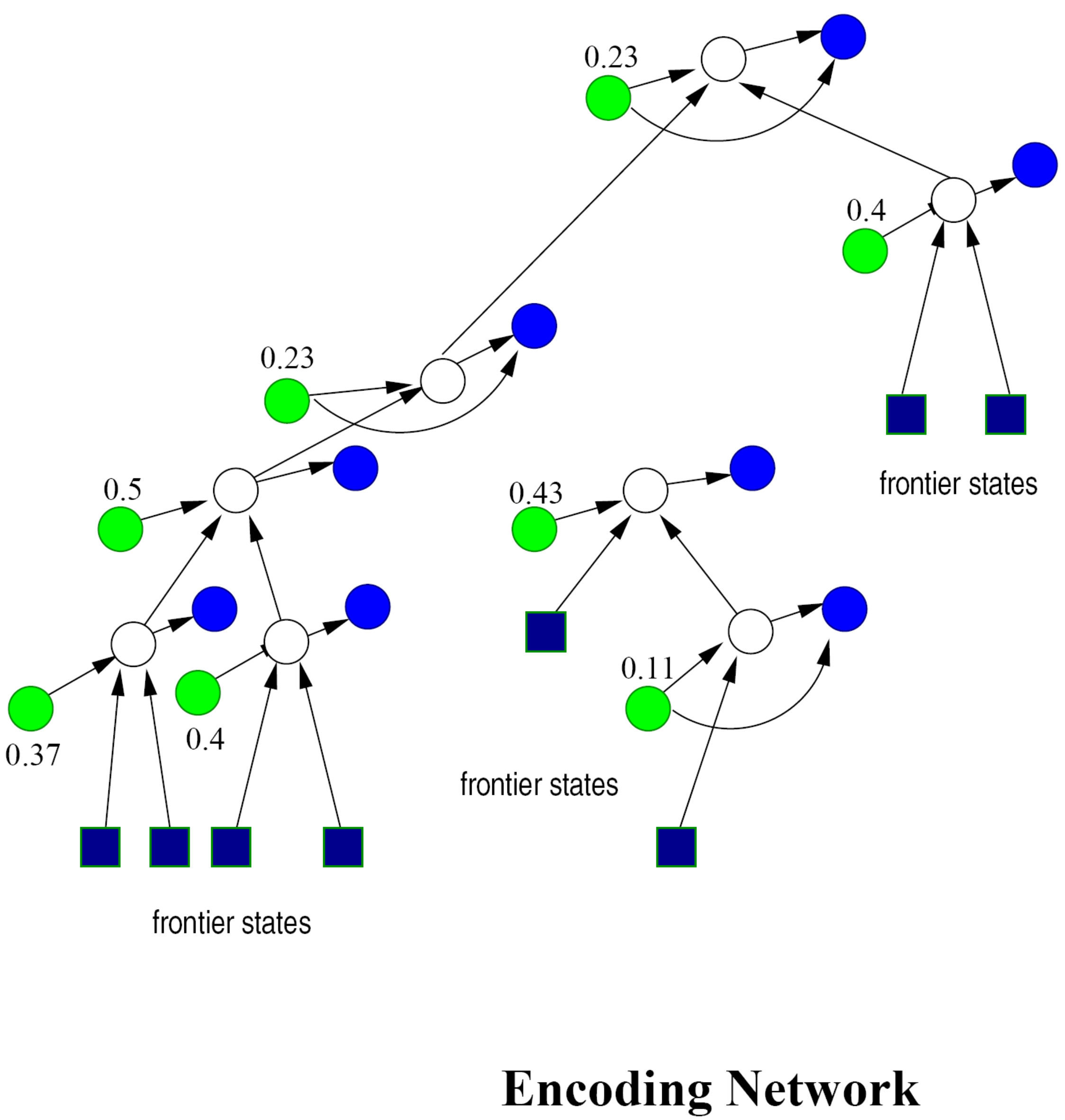

```
;
}
```

Compiling ...

The input tree is mapped to one with different structure!

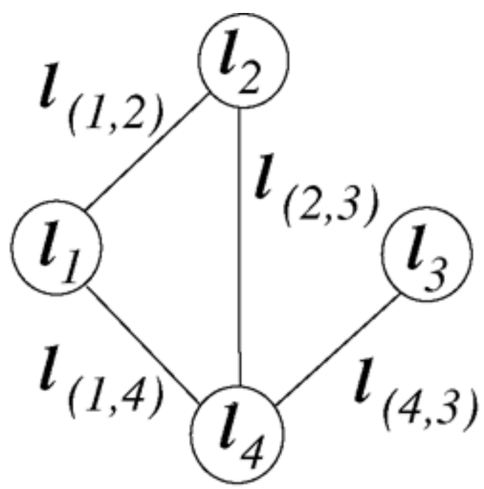
From the previous linguistic specification the encoding network is compiled. Finally, in the last step the encoding neural network is created.





What if DOAG assumption is lost?

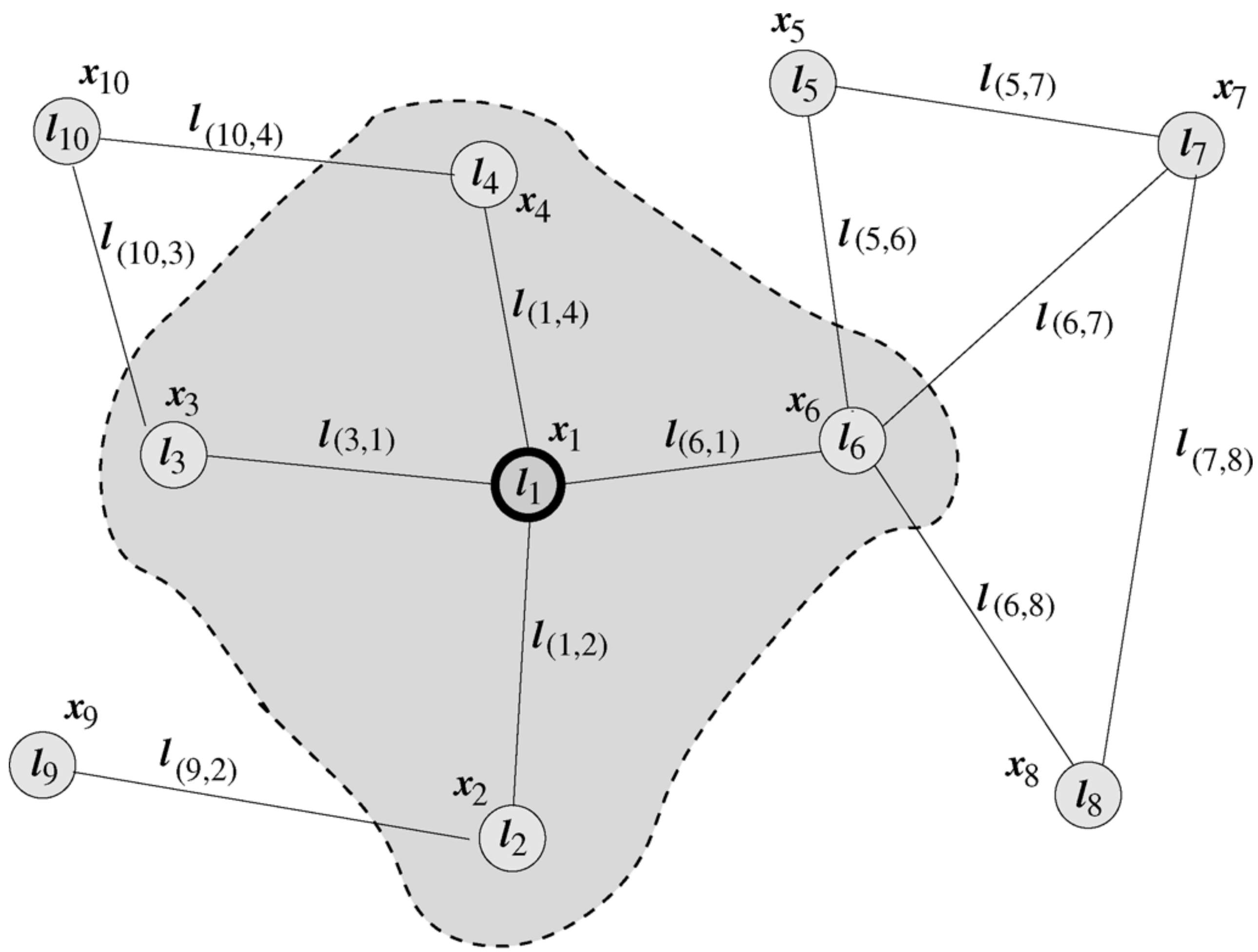
It's the general case which originated the term GNN!



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

We need a different diffusion process!

Neighbor-based computation



$$x_1 = f_w(\underbrace{l_1, l_{(1,2)}, l_{(3,1)}, l_{(1,4)}, l_{(6,1)}}_{l_{co[1]}}, \underbrace{x_2, x_3, x_4, x_6}_{x_{ne[1]}}, \underbrace{l_2, l_3, l_4, l_6}_{l_{ne[n]}})$$

node label connection label neighbor state neighbor label

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

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

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

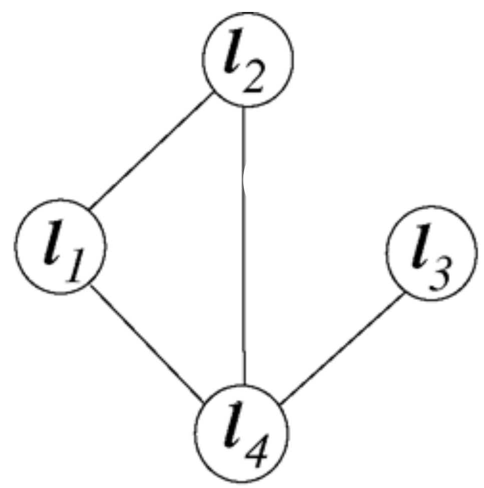
$$\boldsymbol{o} = G_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{l}_{\boldsymbol{N}})$$

equilibrium configuration!

Non-positional graphs

in many cases …

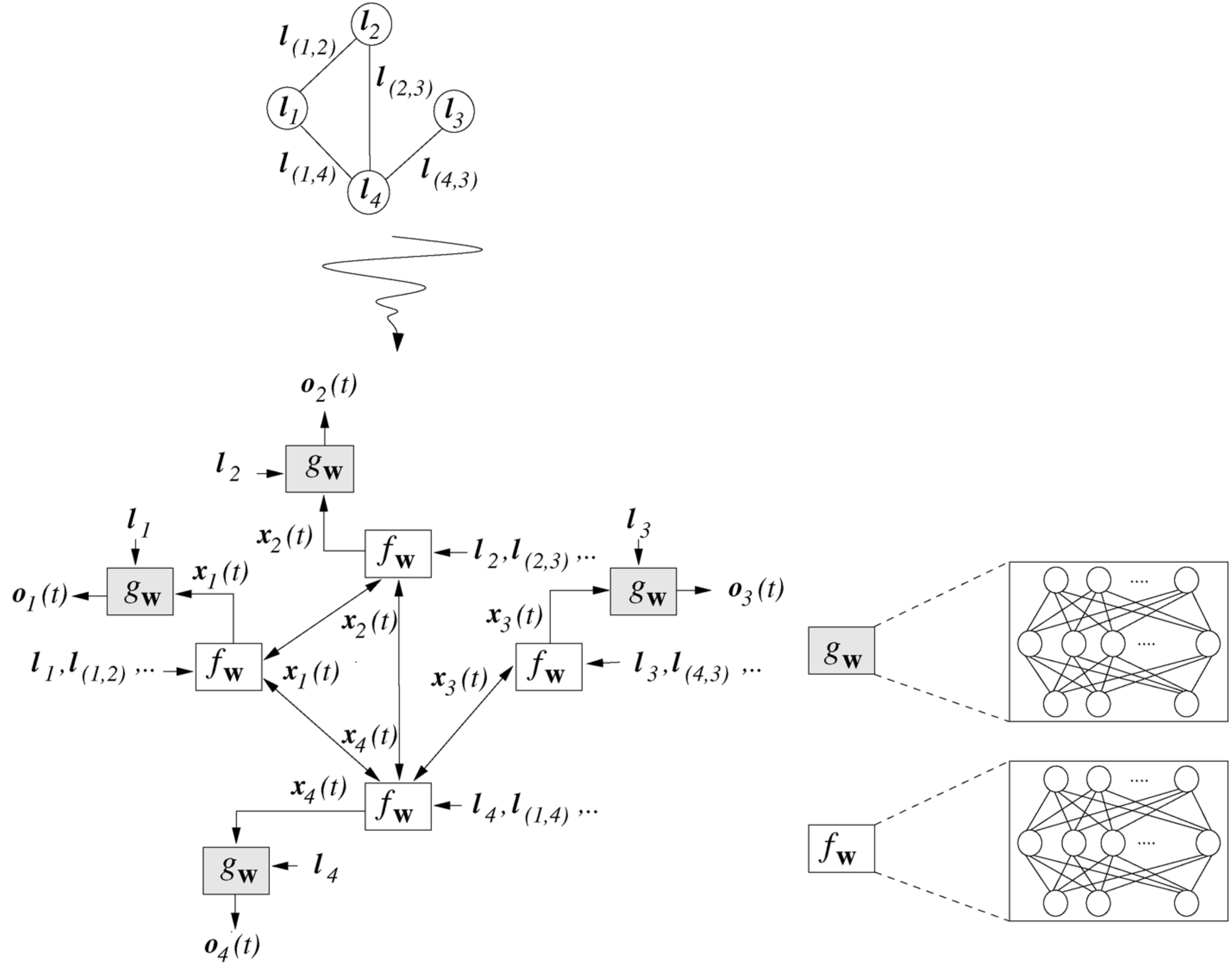
diffusion-based computation similar to PageRank



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

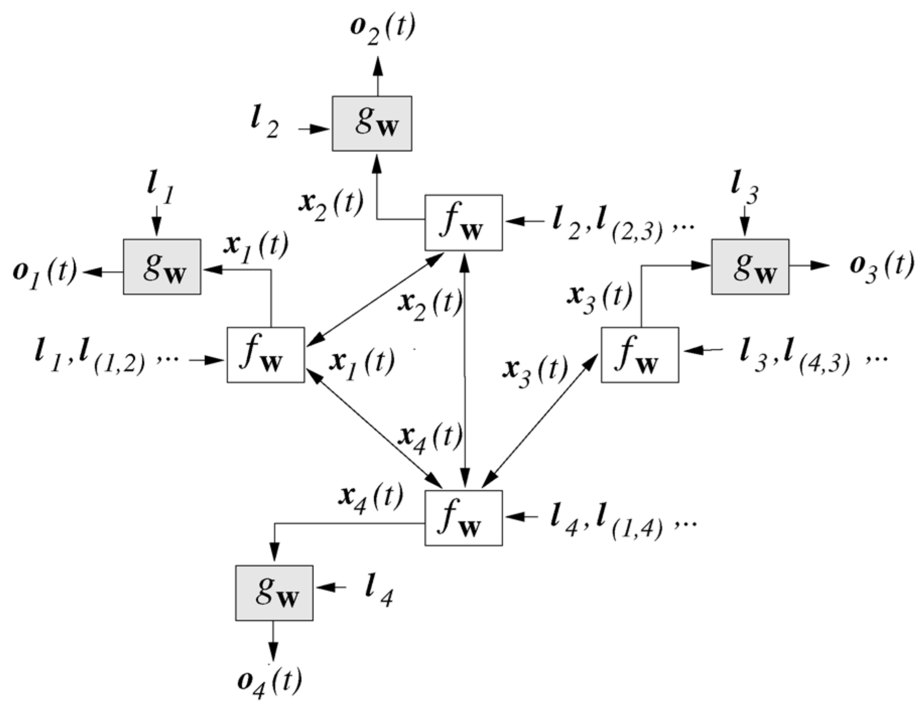
permutation-independent

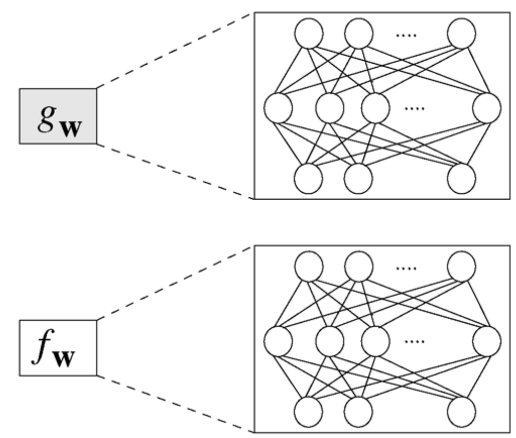
Graph compiling …



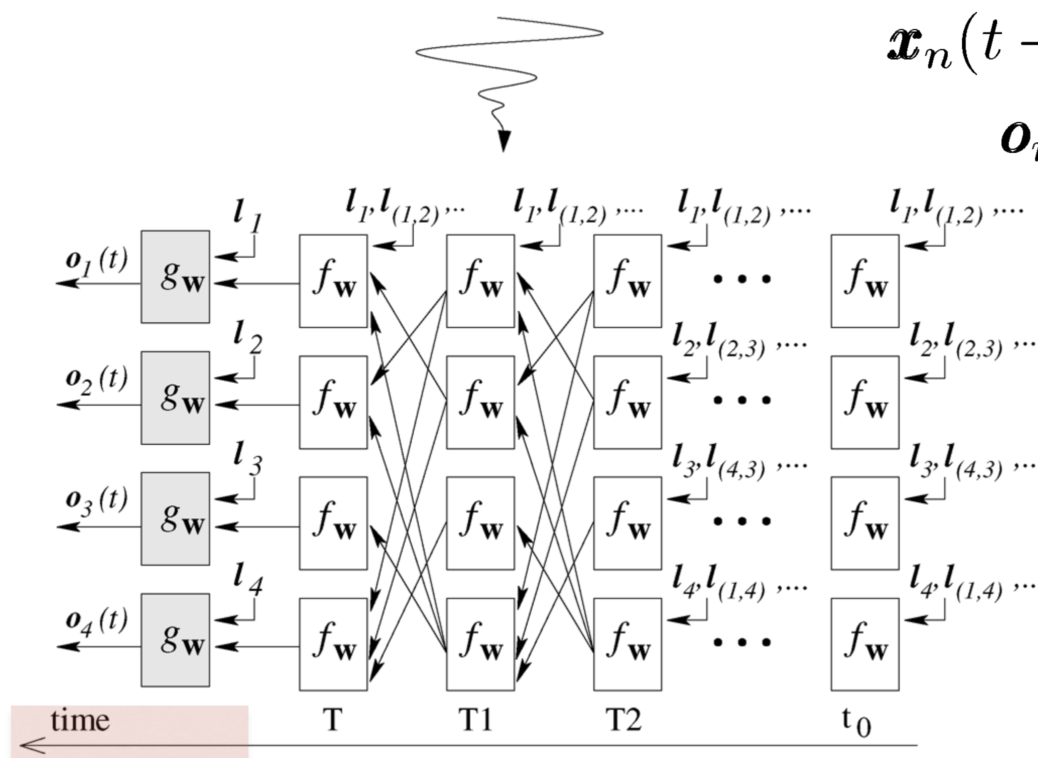
How we get the equilibrium points?

Relaxation to an equilibrium





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



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

GNN Learning

Gori et al IJCNN2005, TNN2009

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

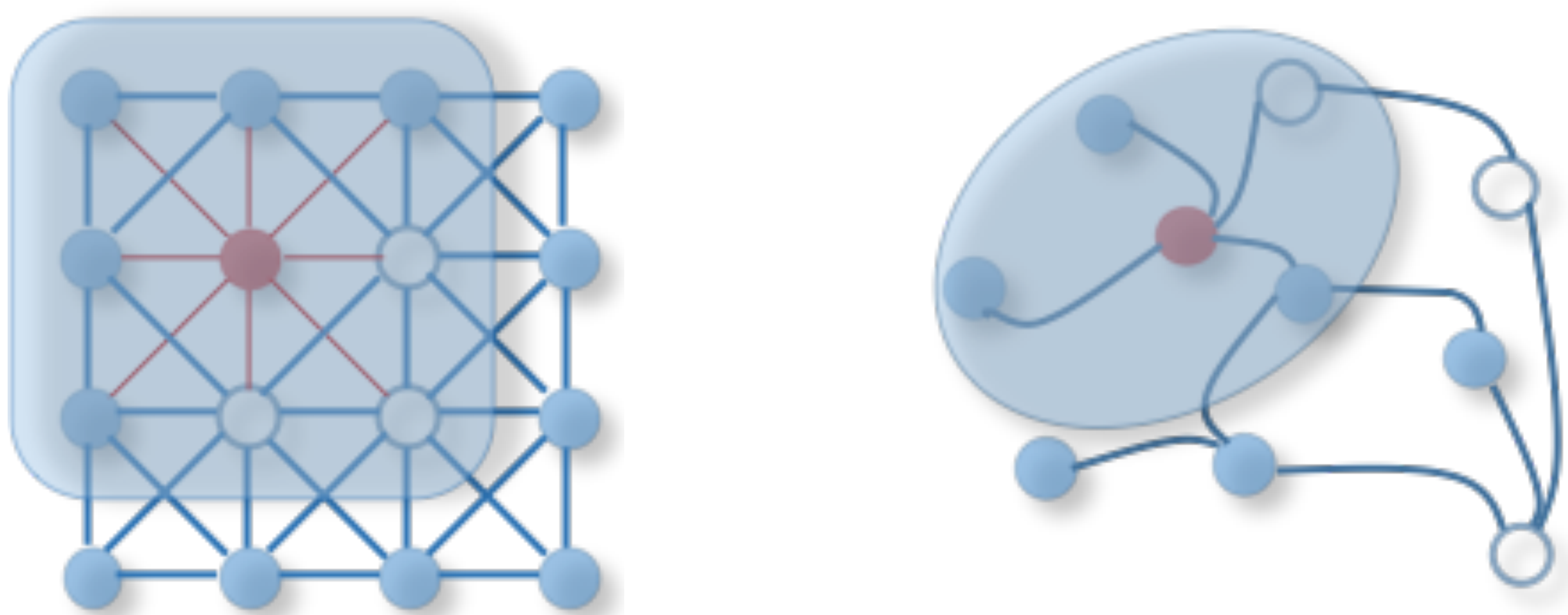
a) The states $\boldsymbol{x}_n(t)$ are iteratively updated by ↑ until at time T they approach the fixed point solution of $\boldsymbol{x}(T) \approx \boldsymbol{x}$.
b) The gradient $\partial e_{\boldsymbol{w}}(T)/\partial \boldsymbol{w}$ is computed.
c) The weights $\boldsymbol{w}$ are updated according to the gradient computed in step b).

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

Beyond GNN

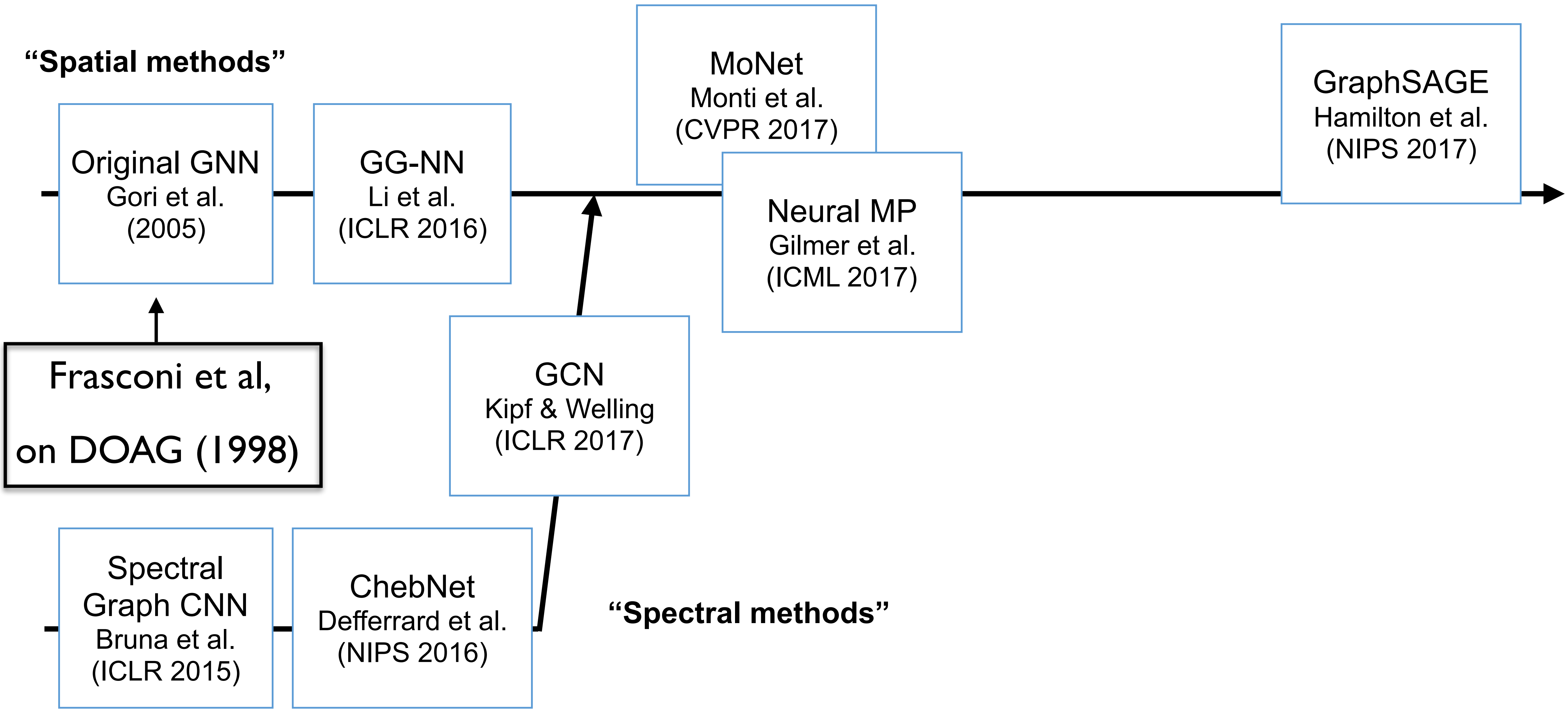
Graph convolutional networks

Layers are not shared!



pictures from Z. Wu et al

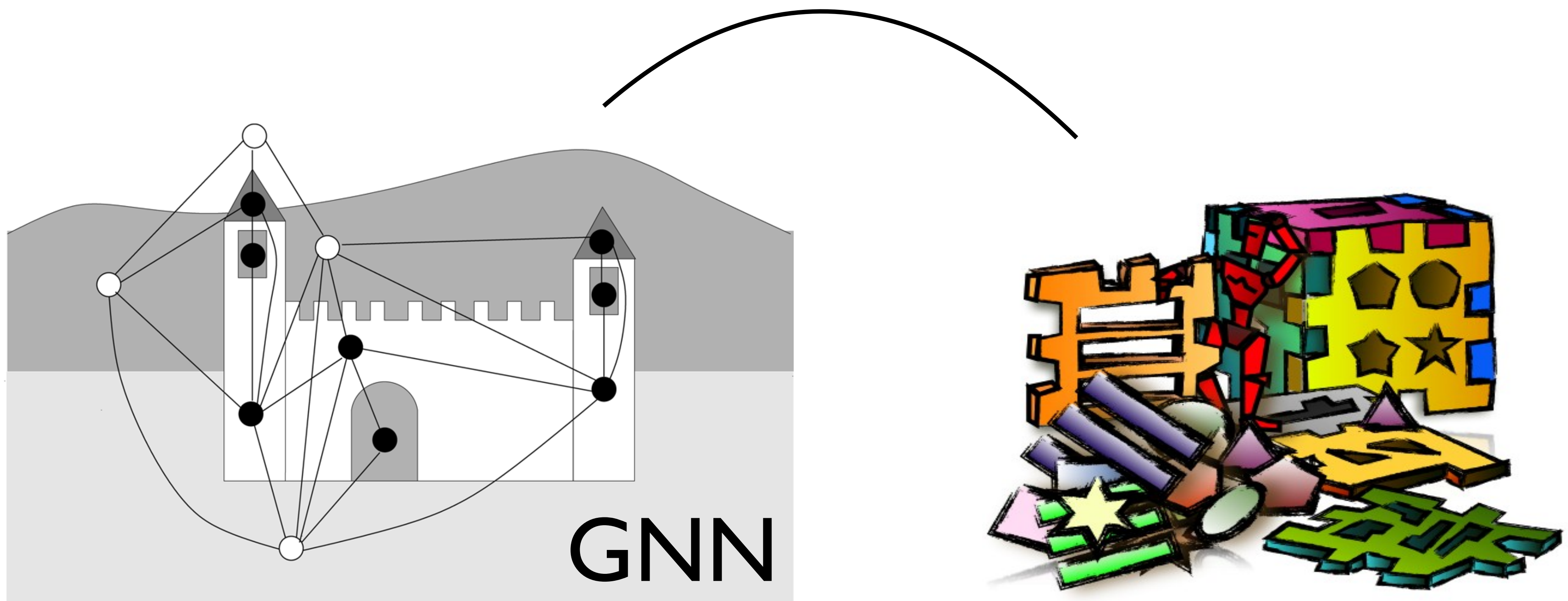
A brief history of graph neural networks



(slide inspired by Alexander Gaunt's talk on GNNs)

GbR 2019

THE FRAMEWORK OF CONSTRAINED-BASED LEARNING



GbR 2019

Constraint-based learning

external "rules"

$$\phi(f(x)) = 0$$

given

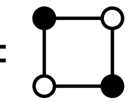
task to be learned

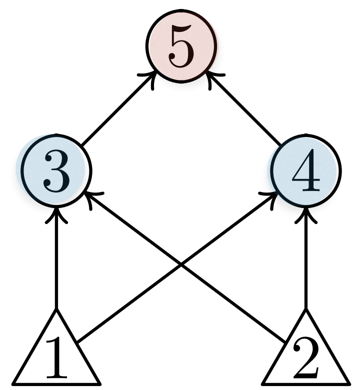
everything revolves around this compositional structure

Gnecco et al, Neural Computation 2015

Supervised Learning

architectural and environmental constraints

$$\mathcal{L} = \{((0,0),0), ((0,1),1), ((1,0),1), ((1,1),0)\} =$$ 



1 2

$$\mathsf{y}((0,1),(1,0))$$

$$\neg\mathsf{y}((0,0),(1,1)).$$

4 3

Lagrangian framework

"hard" architectural constraints

$$x_{\kappa 3} - \sigma(w_{31}x_{\kappa 1} + w_{32}x_{\kappa 2} + b_3) = 0$$
$$x_{\kappa 4} - \sigma(w_{41}x_{\kappa 1} + w_{42}x_{\kappa 2} + b_4) = 0 \qquad \kappa = 1,2,3,4$$
$$x_{\kappa 5} - \sigma(w_{53}x_{\kappa 3} + w_{54}x_{\kappa 4} + b_4) = 0$$

training set constraints

$$x_{15} = 1,\ x_{25} = 1,\ x_{35} = 0,\ x_{45} = 0$$

GbR 2019

Architectural constraints

Supervised learning, Lagrangian formulation

minimize
$$E(w) = \sum_{\kappa=1}^{\ell} \sum_{i \in O} V(x_{\kappa i}, y_{\kappa i})$$

subject to
$i \in H \cup O$
$\kappa = 1, \ldots, \ell$

$$g_{\kappa i} = x_{\kappa i} - \sigma\left(\sum_{j \in pa(i)} w_{ij} x_{\kappa j} \right) = 0$$

hard constraint

$$L(\lambda, w) = \sum_{\kappa=1}^{\ell} \sum_{i \in O} V(x_{\kappa i}, y_{\kappa i}) + \sum_{i \in H \cup O} \sum_{\kappa=1}^{\ell} \lambda_{\kappa i} \left(x_{\kappa i} - \sigma\Big(\sum_{j \in pa(i)} w_{ij} x_{kj} \Big) \right)$$

"Saddle moves": gradient descent/ascent

A more biologically plausibile solution than Backpropagation

saddle points of the Lagrangian

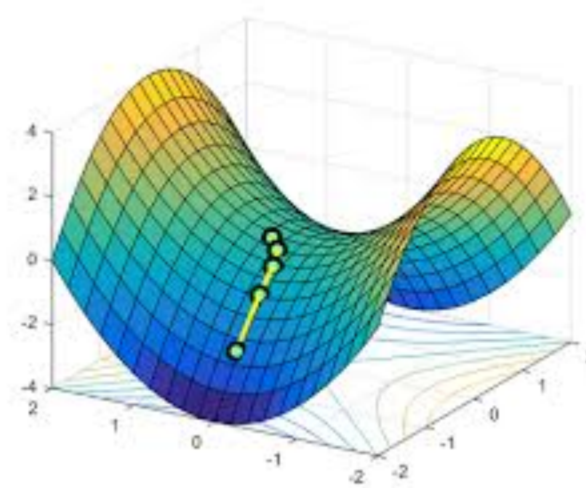
$$w_{ij} \leftarrow w_{ij} - \eta_w \partial_{w_{ij}} L$$

$$x_{\kappa i} \leftarrow x_{\kappa i} - \eta_x \partial_{x_{\kappa i}} L$$

gradient descent

$$\lambda_{\kappa i} \leftarrow \lambda_{\kappa i} + \eta_\lambda \partial_{\lambda_{\kappa i}} L$$

gradient ascent



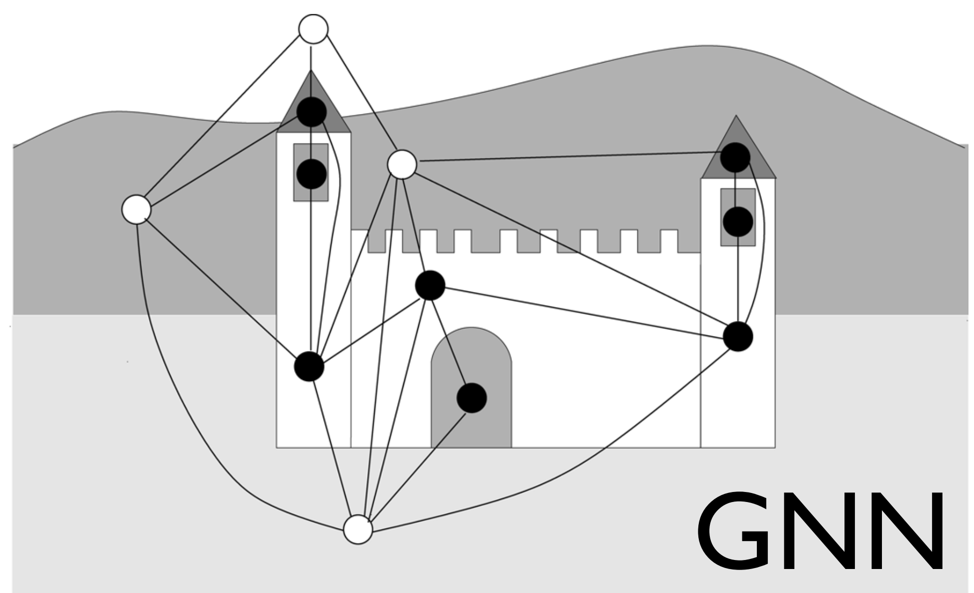
$$g_{\kappa i} = x_{\kappa i} - \sigma\left(\sum_{j \in pa(i)} w_{ij} x_{\kappa j} \right) = 0$$

saddle points of the Lagrangian

Lagrangian multipliers, straw and support neurons!

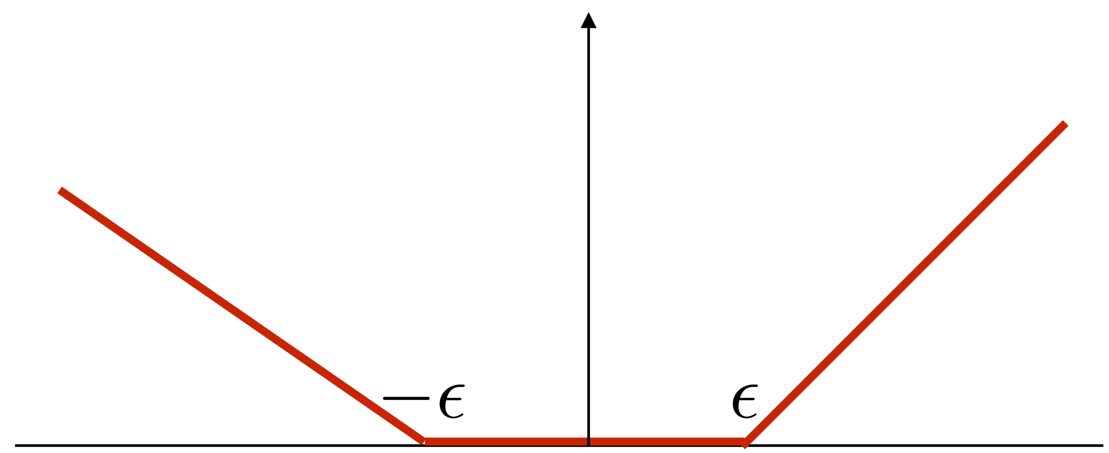
Network growing and constraint selection …

LOCAL PROPAGATION
IN GRAPHIC NEURAL NETWORKS



GbR 2019

Constrained-based expression of data structures



$$\mathcal{G}(x) = \max(||x||_1 - \epsilon, 0)$$

$$\forall v \in V,\ \mathcal{G}\left(x_v - f_a(x_{\mathrm{ne}[v]}, l_{\mathrm{ne}[v]}, l_{(v,\mathrm{ch}[v])}, l_{(\mathrm{pa}[v],v)}, x_v, l_v | \theta_{f_a})\right) = 0$$

transition function

$$v \in S \subseteq V$$

supervised nodes

output function

$$\min_{\theta_{f_a}, \theta_{f_r}, X} \sum_{v \in S} L(f_r(x_v | \theta_{f_r}), y_v)$$

$$\text{subject to} \quad \mathcal{G}\left(x_v - f_a(x_{\mathrm{ne}[v]}, l_{\mathrm{ne}[v]}, l_{(v,\mathrm{ch}[v])}, l_{(\mathrm{pa}[v],v)}, x_v, l_v | \theta_{f_a})\right) = 0, \quad \forall\, v \in V$$

Constrained-based expression of data structures (con't)

$$\mathcal{L}(\theta_{f_a}, \theta_{f_r}, X, \Lambda) = \sum_{v \in S} [L(f_r(x_v|\theta_{f_r}), y_v) +$$
$$+\lambda_v \mathcal{G}\left(x_v - f_a(x_{\mathrm{ne}[v]}, l_{\mathrm{ne}[v]}, l_{(v,\mathrm{ch}[v])}, l_{(\mathrm{pa}[v],v)}, x_v, l_v|\theta_{f_a})\right)]$$

$$\min_{\theta_{f_a}, \theta_{f_r}, X} \max_{\Lambda} \mathcal{L}(\theta_{f_a}, \theta_{f_r}, X, \Lambda)$$

$$\frac{\partial \mathcal{L}}{\partial x_v} = L' f'_{r,v} + \lambda_v \mathcal{G}'_v (1 - f'_{a,v}) - \sum_{w: v \in ne[w]} \lambda_w \mathcal{G}'_w f'_{a,w}$$

$$\frac{\partial \mathcal{L}}{\partial \theta_{f_a}} = -\sum_{v \in S} \lambda_v \mathcal{G}'_v f'_{a,v}$$

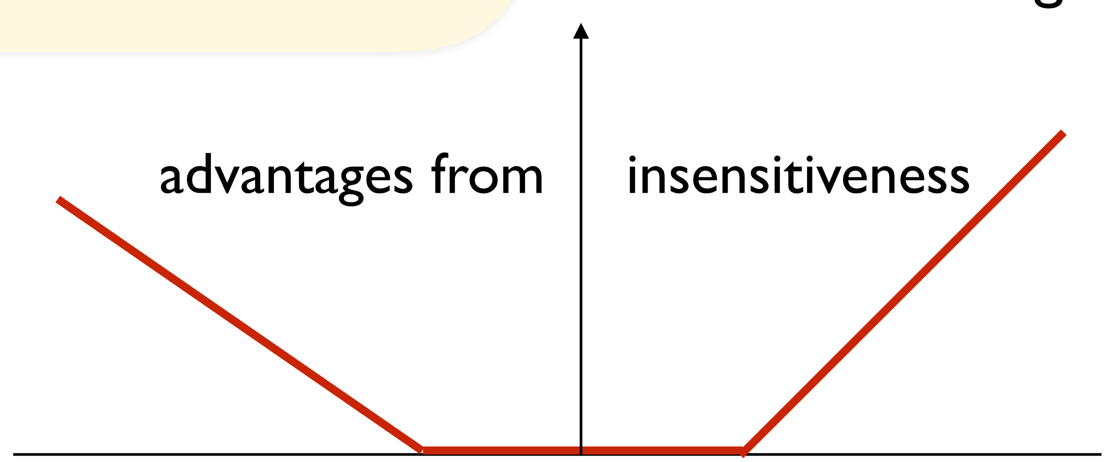
$$\frac{\partial \mathcal{L}}{\partial \theta_{f_r}} = \sum_{v \in S} L' f'_{r,v}$$

gradient descent

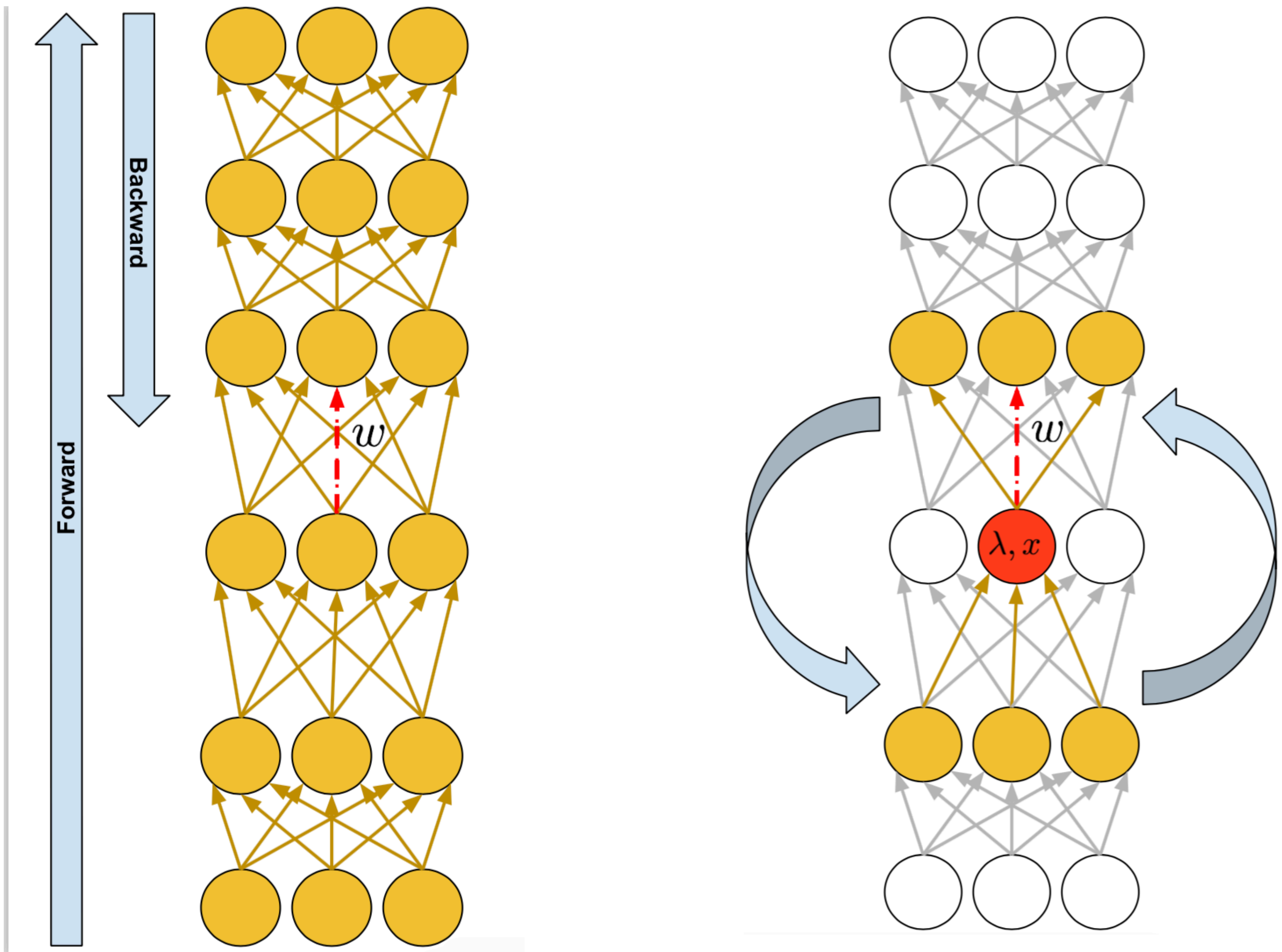
$$\frac{\partial \mathcal{L}}{\partial \lambda_v} = \mathcal{G}_v$$

gradient ascent

better suited for generalization

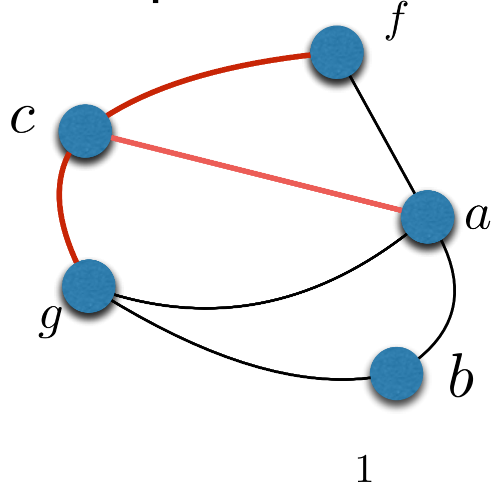


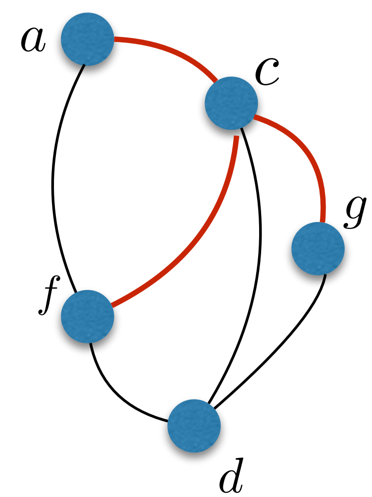
Backpropagation vs Local Propagation

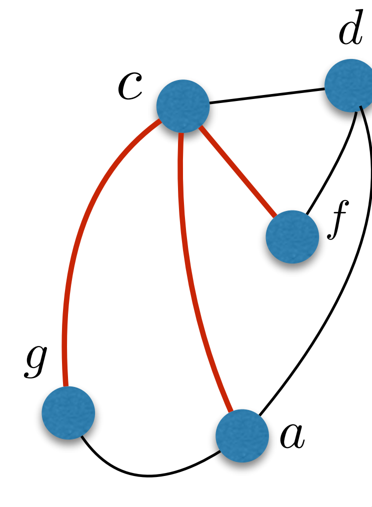


When new graphs come …

constraints on state propagation
constraints on supervision

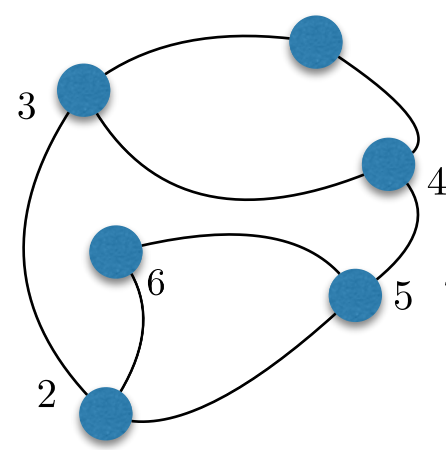






semi-supervised learning

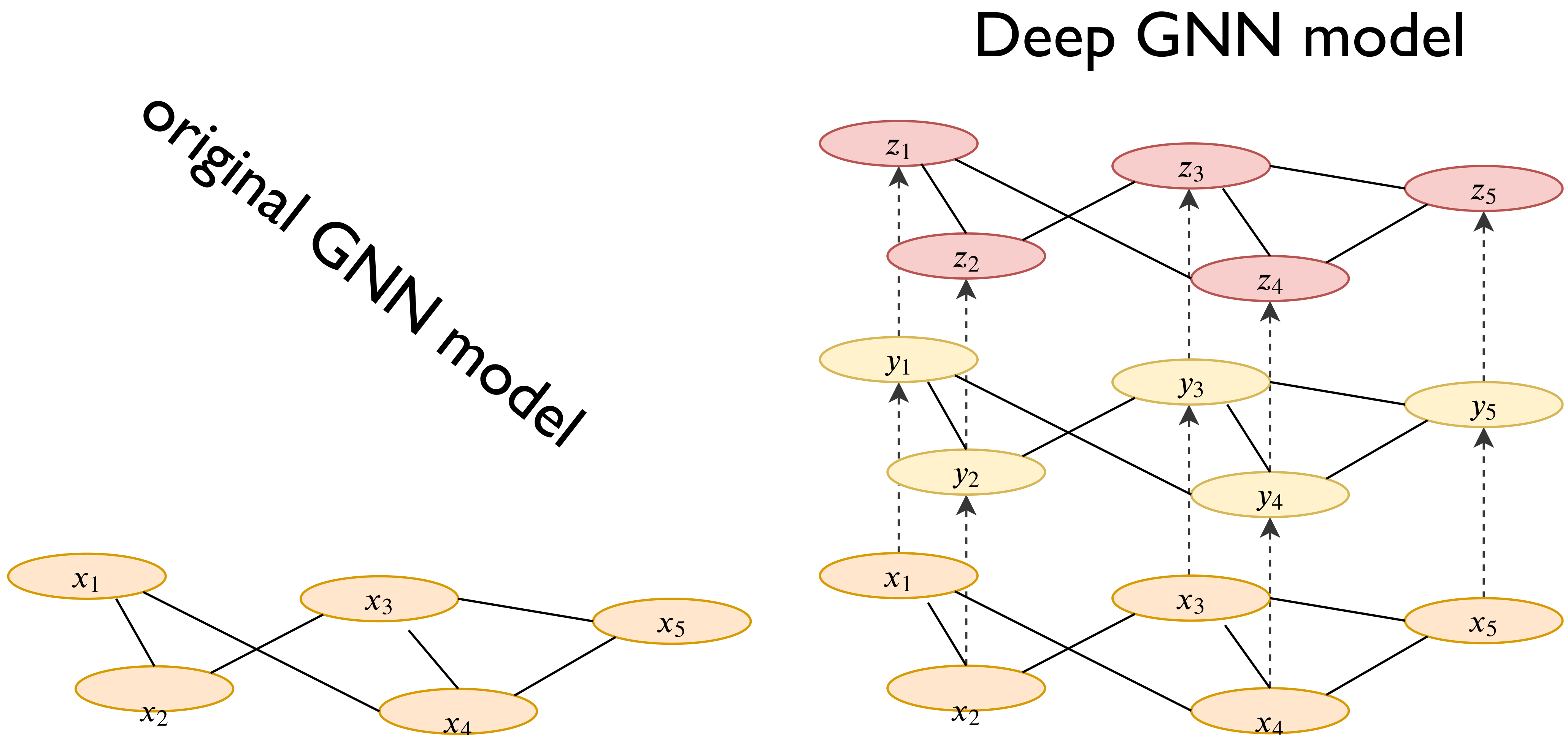
constraints on state propagation



you can always enforce state propagation

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

Deep GNN



(a) A shallow GNN.

(b) A deep GNN.

Conclusions

A unified framework for learning and reasoning

- GNN as diffusion machines and its evolution
- Constrained-based learning
- Saddle move algorithms and local computation
- The natural links with logic
- Learning of constraints and explanation

Special Issue on Non-Euclidean Deep Learning

Paper submission due: 15 July 2019
First Notification: 1 November 2019
Revision: 1 January 2020
Final Decision: 1 March 2020
Publication date: June 2020 (tentative)

Guest Editors

Michael Bronstein*, Imperial College London (UK), michael.bronstein@imperial.ac.uk

Joan Bruna, New York University (USA), bruna@cims.nyu.edu

Taco Cohen, Qualcomm AI Research (Netherlands), tacos@qti.qualcomm.com

Marco Gori, University of Siena (Italy), marco@diism.unisi.it

Pietro Lio', University of Cambridge (UK), pl219@cam.ac.uk

Jure Leskovec, Stanford University (USA), jure@cs.stanford.edu

Le Song, Georgia Institute of Technology (USA), lsong@cc.gatech.edu

Oriol Vinyals, DeepMind (UK), vinyals@google.com

Stefanos Zafeiriou*, Imperial College London (UK), s.zafeiriou@imperial.ac.uk

Surveys

- P.W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner et al., "Relational inductive biases, deep learning, and graph networks," arXiv preprint arXiv:1806.01261, 2018.

- Graph Neural Networks: A Review of Methods and Applications. Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Maosong Sun. 2018
- Geometric Deep Learning: Going beyond Euclidean data. Bronstein, Michael M and Bruna, Joan and LeCun, Yann and Szlam, Arthur and Vandergheynst, Pierre. IEEE SPM 2017

Software resources at SAILAB

- https://sailab.diism.unisi.it/gnn/

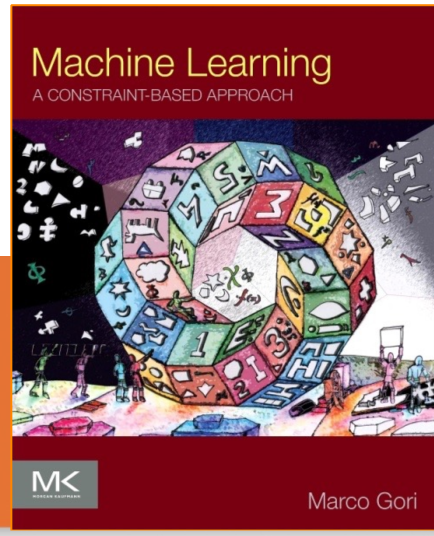
- https://github.com/GiuseppeMarra/CLAREecml

Machine Learning

A CONSTRAINT-BASED APPROACH

MK
MORGAN KAUFMANN

Marco Gori



ISBN: 978-0-08-100659-7
PUB DATE: November 2017
LIST PRICE: £59.99/€70.95/$99.95
FORMAT: Paperback
PAGES: c. 580

AUDIENCE

Upper level undergraduate and graduate students taking a machine learning course in computer science departments and professionals involved in relevant areas of artificial intelligence

A focused approach that covers the deep ideas of machine learning through a variety of specific techniques

KEY FEATURES

- **It** is an introductory book for all readers who love in-depth explanations of fundamental concepts.
- It is intended to stimulate questions and help a gradual conquering of basic methods, more than offering "recipes for cooking."
- It proposes the adoption of the notion of constraint as a truly unified treatment of nowadays most common machine learning approaches, while combining the strength of logic formalisms dominating in the AI community.
- It contains a lot of exercises along with the answers, according to a slight modification of Donald Knuth's difficulty ranking.
- It comes with a companion Web site to assist more on practical issues.

QUOTES

A fairly comprehensive and original book on machine learning, including deep learning, written from a constraint-based perspective where Marco Gori shares his passion for the topic with his reader. The book comes also with a set of useful problems, exercises, solutions, as well as a companion web site.

Pierre Baldi, University of California Irvine

This very interesting book brings a fresh look at machine learning and deep learning from the broad point of view in which learning corresponds to satisfying constraints, encompassing the perceptual as well as the symbolic, soft as well as hard constraints.

Yoshua Bengio, Université de Montréal

A real tour-de-force across the landscape of a field -- machine learning -- which is developing very rapidly and is transforming a large swath of today's science and engineering of intelligence.

Tomaso Poggio, MIT

9 780081 006597