Intro to Learning in SD -1

Alessio Micheli

E-mail: **micheli@di.unipi.it**

1- Introduction to RecNN

Apr 2019

DRAFT, please do not circulate!

www.di.unipi.it/groups/ciml

Dipartimento di Informatica
Università di Pisa - Italy

Computational Intelligence & Machine Learning Group

Learning in Structured Domain
Plan in 2 lectures

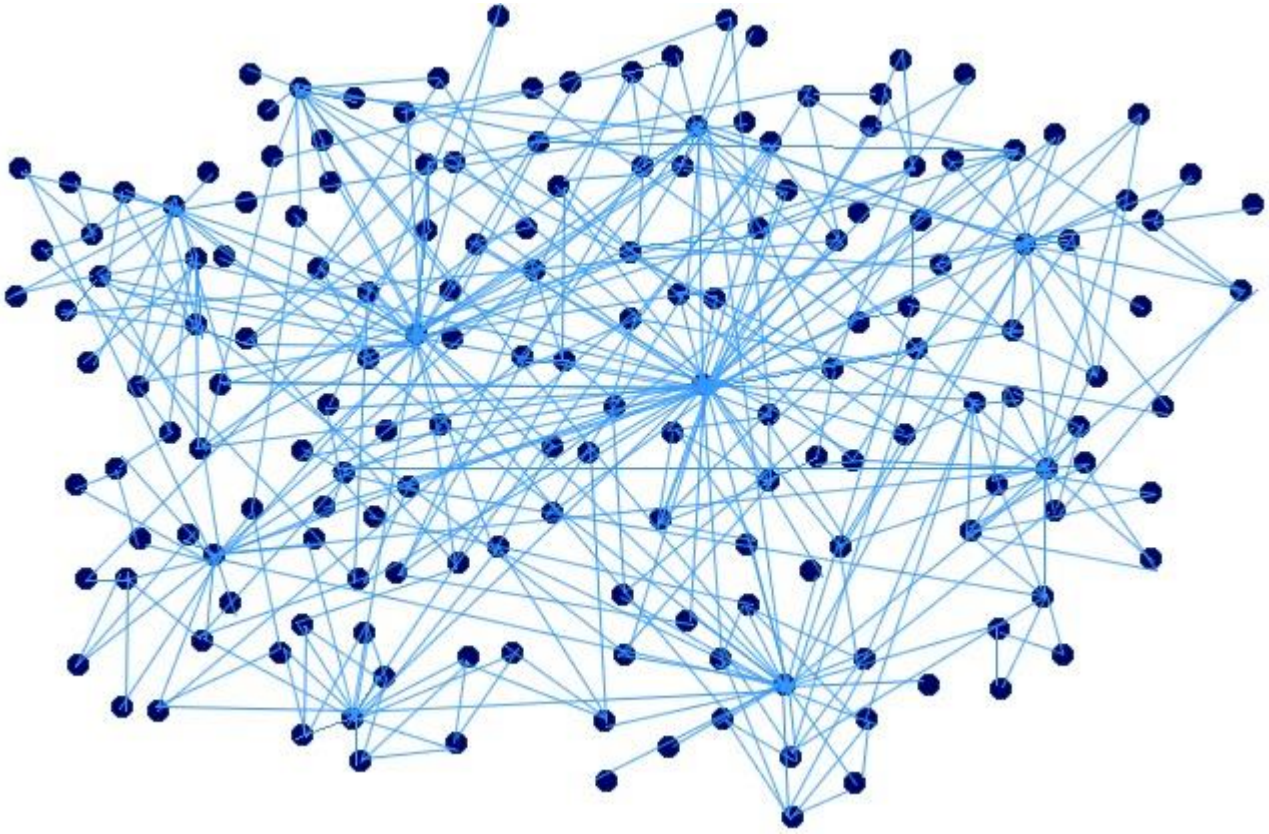
1. Recurrent and Recursive Neural Networks

   Extensions of models for learning in structured domains

   - Motivation and examples (structured data)
   - The structured data (recursive)
   - Recursive models: RNN and RecNN
   - Recursive Cascade Correlation & other recursive approaches

2. Moving to DPAG and Graphs: the role of causality [Next Lecture (SD-2)]

Why structured data?



Because data have relationships

Introduction: Motivation of ML for SD

- Most of known ML methods are limited to the use of flat and fixed form of the data (vectors or sequences)

  fixed-length attribute-value vectors

- Central: *data representation*
- Graph: very useful abstraction for real data

- **Labeled graphs** = vector patterns + relationships
  - **natural**: for structured domain
  - **richness**
  - **efficiency**: repetitive nature inherent the data

- **SD + ML =** adaptive processing of structured information

Introduction: Research Area

- **SD + ML =** adaptive processing of structured information

- **General Aim**: investigation of ML models for the adaptive processing of structured information: **sequences, trees, graphs**:
  - Structured domain learning/ Learning in Structured Domain
  - Relational Learning
  - Structure/Graph Mining
    - Molecule Mining
  - ***… Deep Learning for Graphs***

Advancements from ML to SDL

Learning in Structured Domains (SD) in Pisa/CIML: Pioneering since the 90's the development of

- Theoretical analysis
- New approaches
- Applications

Especially on the basis of Recursive approaches.

Computational Intelligence & Machine Learning Group

And for you?

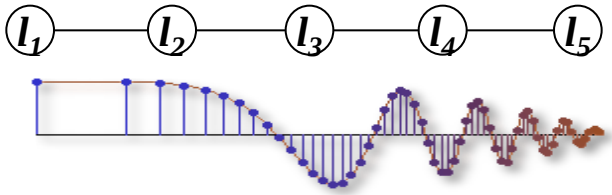
- To build and advanced background for
  - Analysis/development of innovative models
  - Applications in the are of interdisciplinary projects (@ ciml)
- Practical: **thesis are possible for the design of new models/applications for the extension of the input domains toward:**
  - Extension of the applicative domain
  - Adaptivity and accuracy
  - Efficiency

From flat to structured data

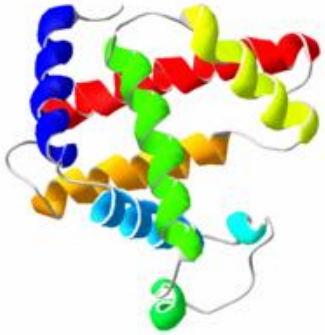
- **Flat: vectors** (as in the rest of AA1)
- **Structured:** Sequences, trees, graphs, multi-relational data

stringa_in_italiano

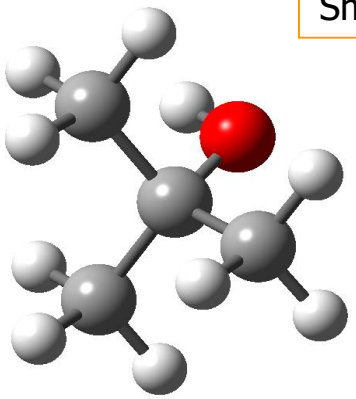
Strings



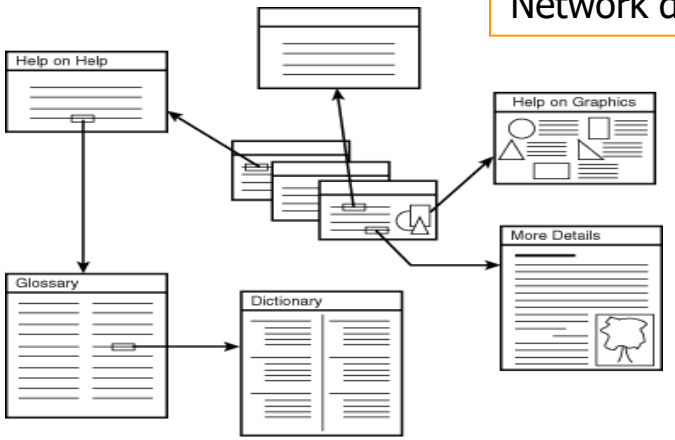
Series/
temporal stream



Proteins

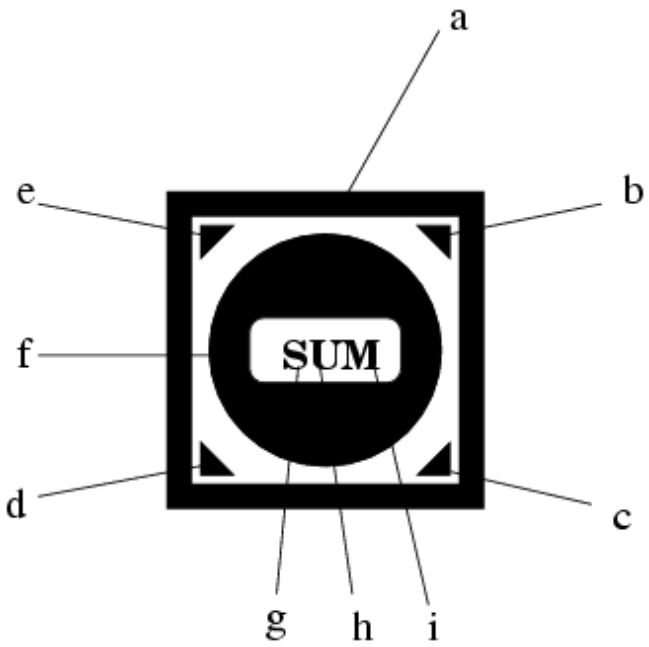


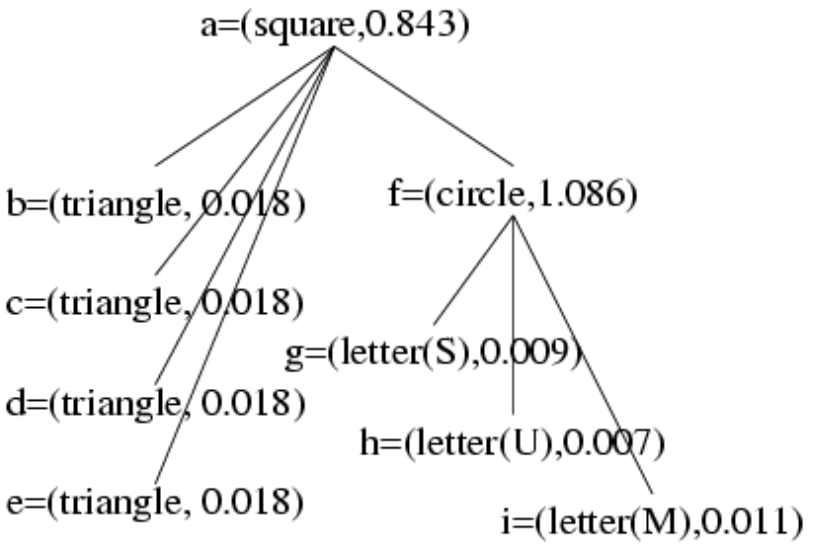
Small molecules

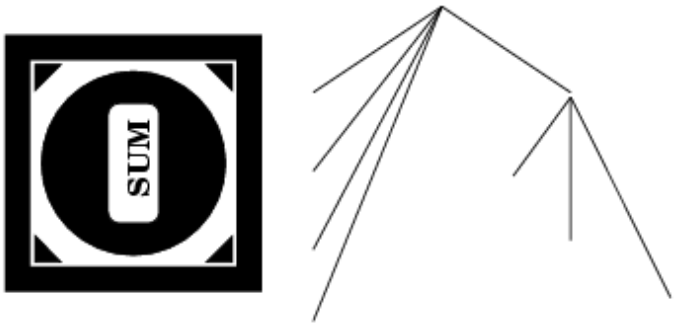


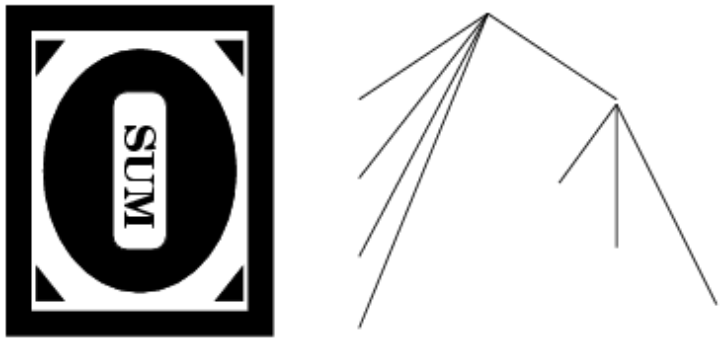
Network data

Example: logo recognition

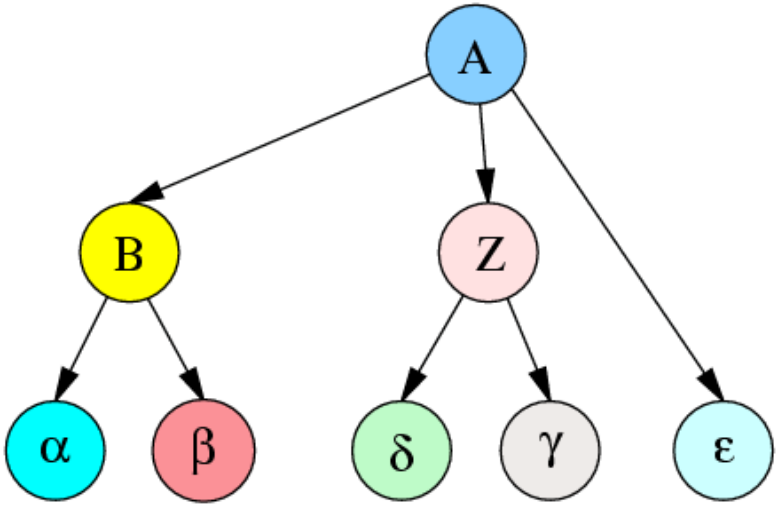




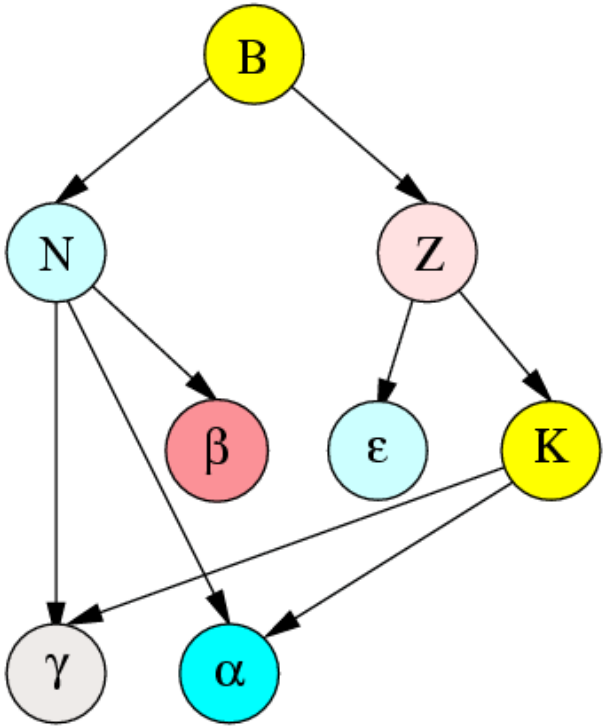




Example: Terms in 1st order logic

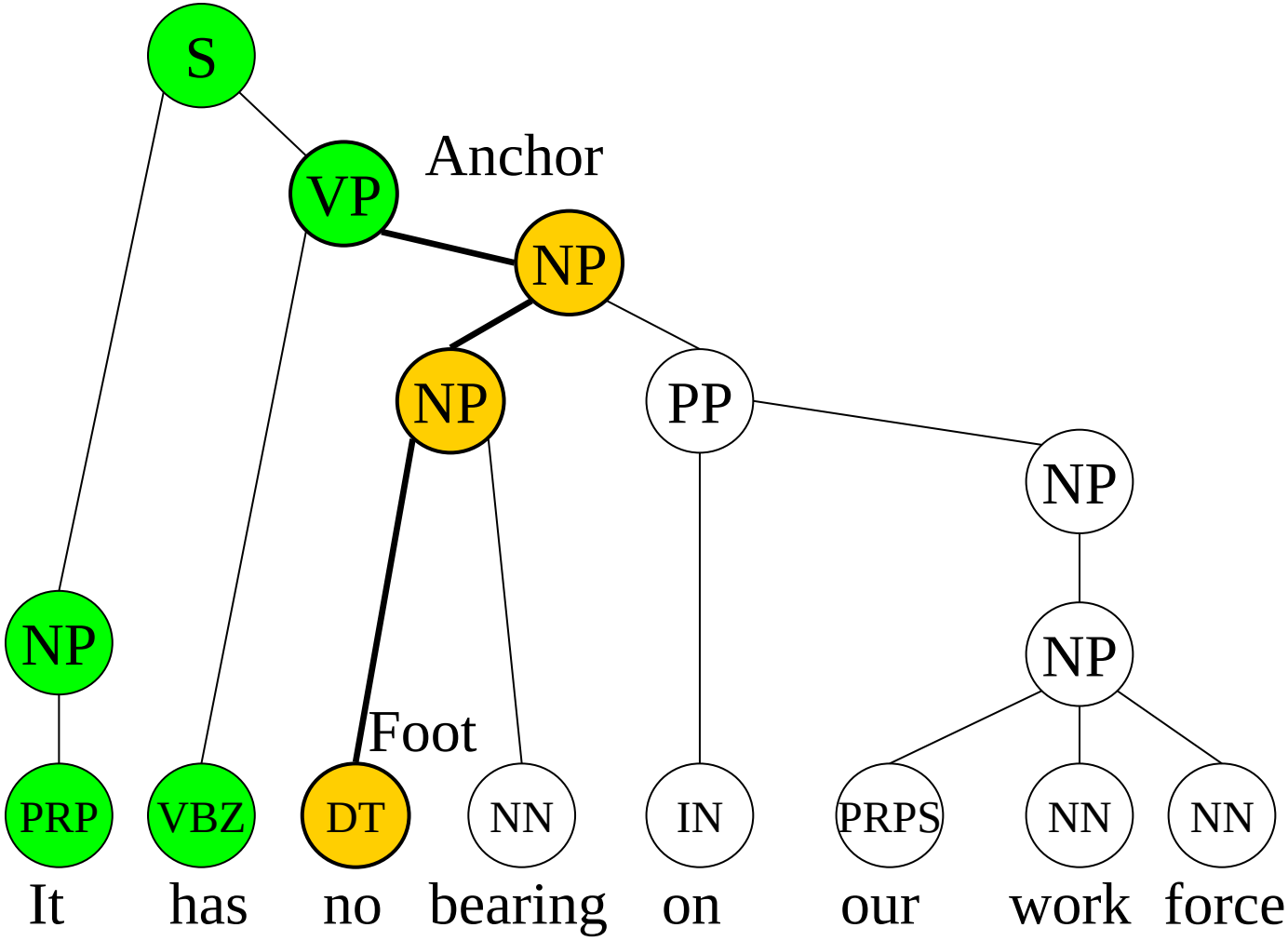


A(B(α,β),Z(γ,δ),ε)

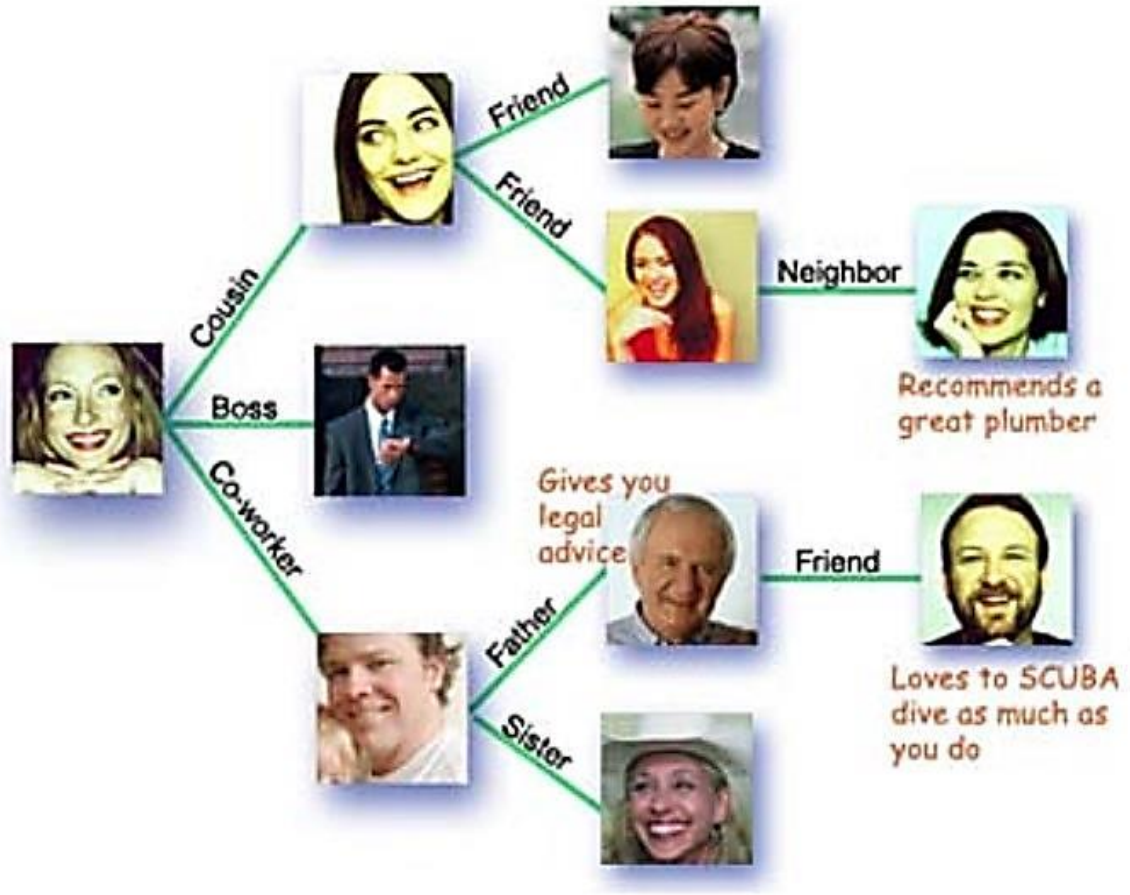


B(N(γ,α,β),Z(ε,K(γ,α)))

Example (trees): language parsing



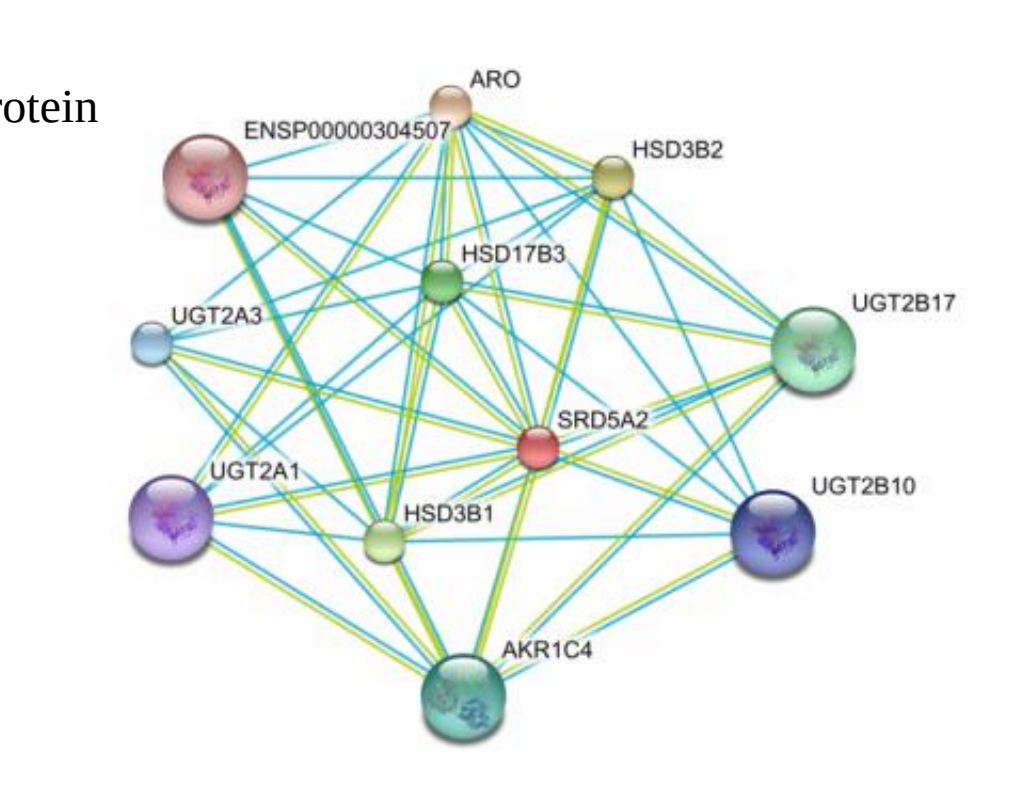
Example: Social networks



Example: Biological Networks

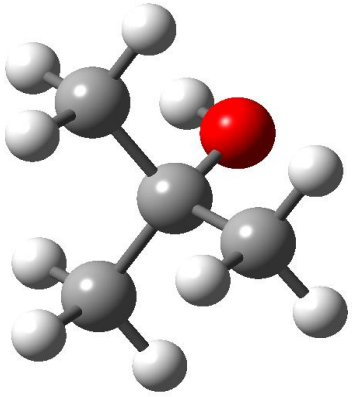
- Node for protein
- Link for **interaction** or similarity

Protein



Example (graphs): Molecules

- A fundamental problem in Chemistry: correlate chemical structure of molecules with their properties (e.g. physico-chemical properties, or biological activity of molecules) in order to be able to predict these properties for new molecules
  - Quantitative Structure-Property Relationship (QSPR)
  - Quantitative Structure-Activity Relationship (QSAR)



*Property/Activity = **T**(Structure)*

T

Property Value (regression)
Toxic (yes/no) (classification)

QSPR: Correlate chemical structure of molecules with their properties

Molecules are not vectors !

Molecules can be more naturally represented by varying size ***structures***

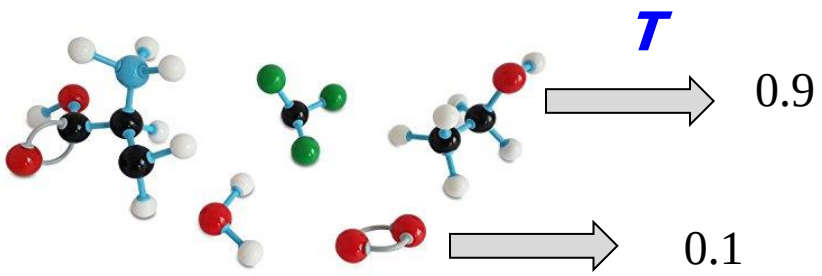
Can we predict directly from structures ?

Learn a transduction

- **Goal**: to learn a mapping between a structured information domain (SD) and a discrete or continuous space (*transduction* ***T***).

- Start with this problem: classify variable size graphs
  - For instance, classify different graphs starting from a training set of know couples as in the molecules example

- Given a set of examples $(graph_i, target_i)$
- Learn an hypothesis mapping $T(graph)$



Introduction: Learning Model for SD

- The problem: there has been no systematic way to extract features or metrics relations between examples for SD
  - A representation learning instances (extended to SD)!
- What we mean for *adaptive* processing of SD:

  extraction of the topological information directly from data
  - $\mathcal{H}$ has to be able to represent hierarchic relationships
  - **adaptive** measure of similarity on structures + apt **learning rule**
  - efficient handling of structure variability
  - Classical:
    - efficient learning
    - good generalization performance
    - knowledge extraction capabilities

Learning in Structured Domain
Plan in 2 lectures

1. Recurrent and Recursive Neural Networks

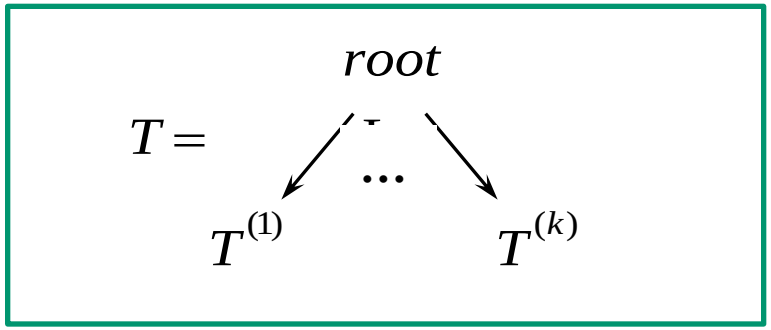
   Extensions of models for learning in structured domains

   - Motivation and examples (structured data)
   - **The structured data (recursive)**
   - Recursive models: RNN and RecNN
   - Recursive Cascade Correlation & other recursive approaches

2. Moving to DPAG and Graphs: the role of causality [Next Lecture (SD-2)]

K-ary Trees

- ***k-ary trees*** (*trees* in the following) are rooted positional trees with finite out-degree k.
- Given a node v in the tree $T \in \boldsymbol{G}$:
  - The children of v are the node successors of v, each with a position $j=1,\ldots k$;
  - k is the maximum out-degree over G, i.e. the maximum number of children for each node;
  - $L(v)$ in is the input label associated with v, and $L_i(v)$ is the i-th element of the label;
  - The *subtree* $T^{(j)}$ is a tree rooted at the j-th children of v.



Structured Domains

- ***L***: set of attribute vectors $l(v)$
- *Structure G*: vertexes labels ***L*** + topology (skeleton of *G*)
- Sequences, Trees, DOAGs- DPAGs, graphs:

- ***G*** : labeled direct ordered/positional acyclic graphs with super-source
  - A total order (or the position) on the edges leaving from each vertex
  - *Super-source*: a vertex *s* such that every vertex can be reached by a direct path starting from *s.*
  - Bounded out-degree and in-degree (the number of edges leaving and entering from a vertex *v*)

- **DPAG**: supeclass of the **DOAG**s: besides ordering, a distinctive positive integer can be associated to each edge, allowing some position to be absent.

- Trees: labeled rooted order trees, or positional (K-ary trees).
  - Super-source: the root of the tree.
  - **Binary tree (K=2)**

Mostly used in this lecture

Data Domains *G*

- We consider sets of **DPAG**s: labeled direct positional acyclic graphs with *super-source*, bounded *in-degree* and *out-degree* (*k*).

- Include sub-classes:

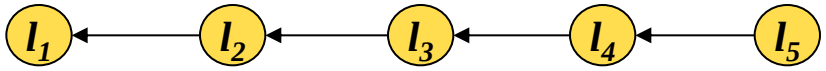
  DPAGs $\supset$ DOAGs $\supset$ k-ary trees $\supset$ sequences $\supset$ vectors.

- Notations:
  - ch[*v*] set of successors of *v*
  - ch_j[*v*] is the j-th child of the node *v*

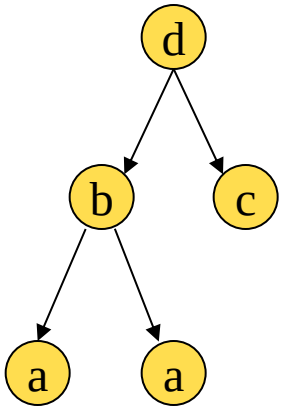
Structured Data: Examples

- Labeled Sequences, Trees, DOAGs- DPAGs, graphs

Single labeled vertex

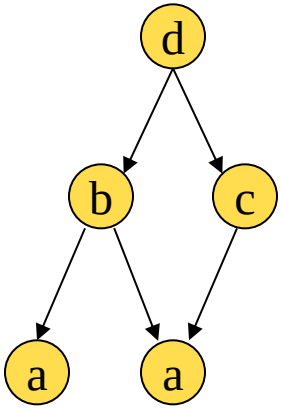


Sequence

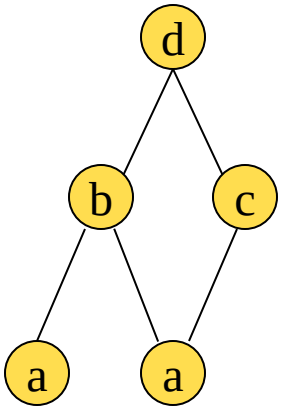


Rooted Tree

Supersource



DPAG



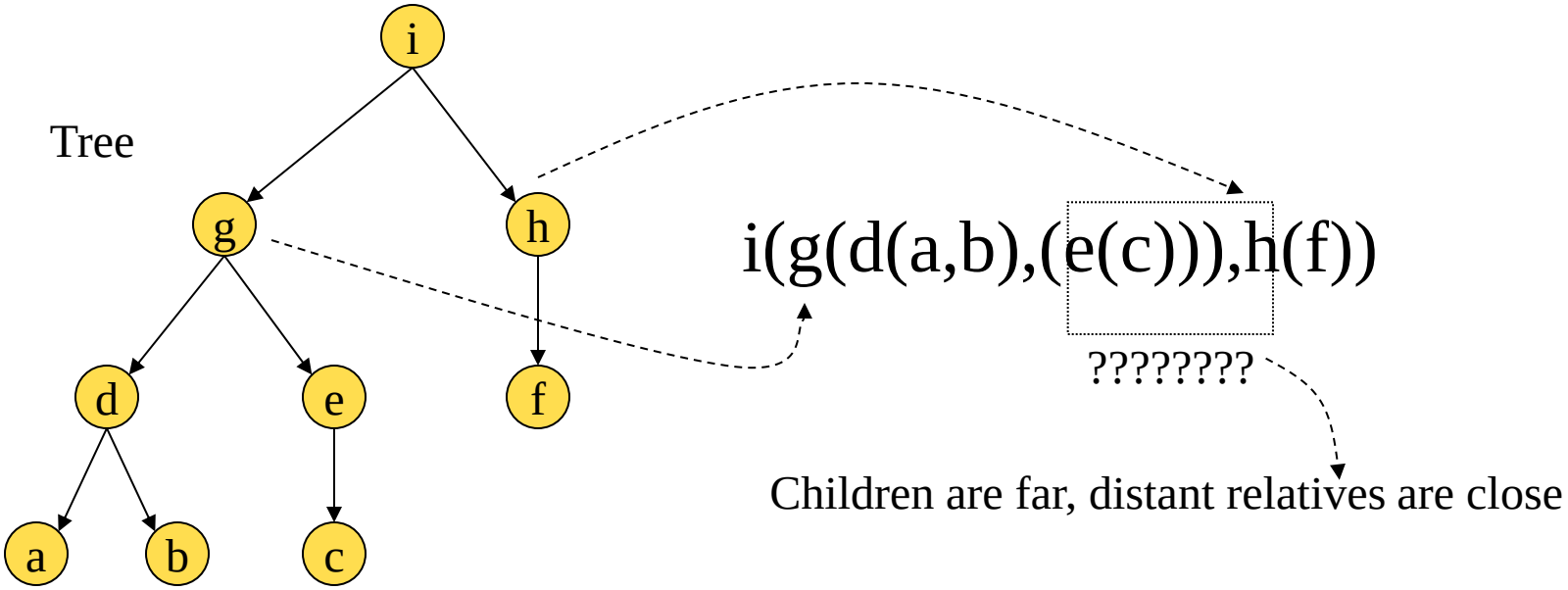
Graph (undirected)

DPAG: labeled direct positional acyclic graphs with *super-source*, bounded *in-degree* and *out-degree* (*k*).

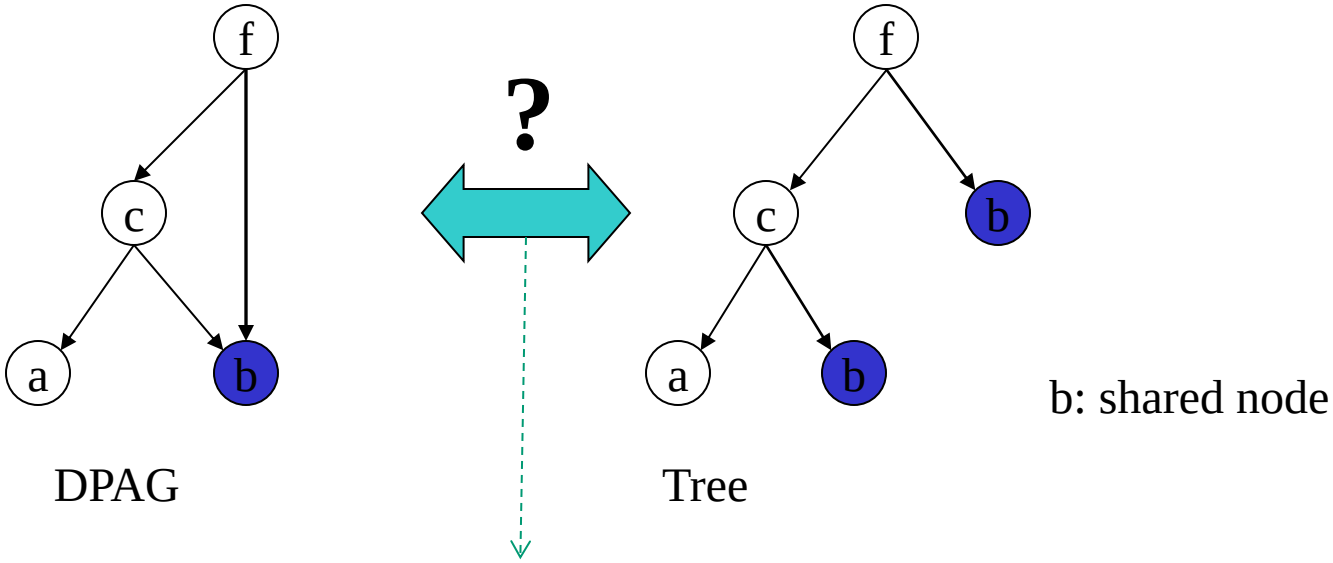
Structures: just use sequence ?

- Can we process structures like they are sequences?
- E.g. any tree can be converted into a sequence (no information loss) but:
  - Sequences may be long: number of vertex exp w.r.t. height of tree

    (aka the paths are *log* of #nodes for the tree, so the dependencies are much shorter)
  - Dependencies are blurred out (arbitrary depending on the visit)



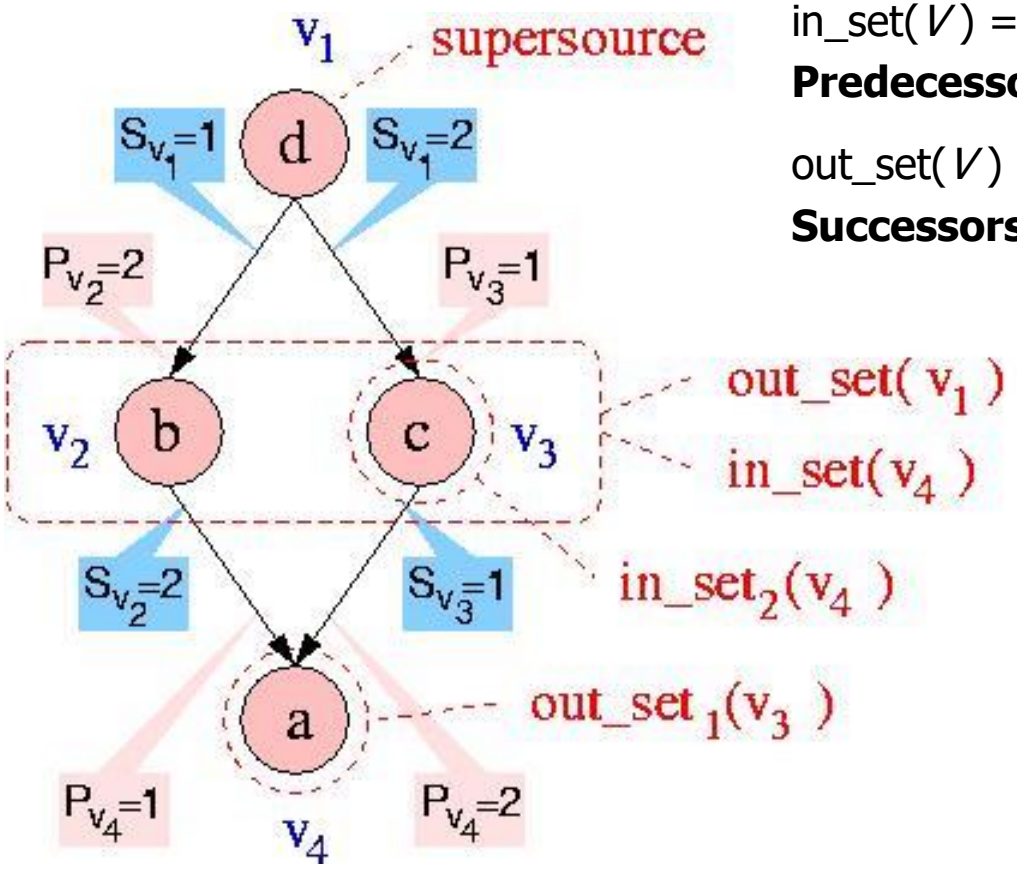
Trees and DPGAs



Exercise after the lecture SD-2

Positional versus Ordered

- DPAG: for each vertex *v in vert(G)*, an injective function S_v: *edg(v)* → [*1,2,..,K*] is defined on the edges leaving from *v*



in_set(V) = {u | $v \in V$ and $u \rightarrow v$}
Predecessors

out_set(V) = {u | $v \in V$ and $v \rightarrow u$}
Successors

Learning in Structured Domain
Plan in 2 lectures

1. Recurrent and Recursive Neural Networks

   Extensions of models for learning in structured domains

   - Motivation and examples (structured data)
   - The structured data (recursive)
   - **Recursive models: RNN and RecNN**
   - Recursive Cascade Correlation & other recursive approaches

2. Moving to DPAG and Graphs: the role of causality [Next Lecture (SD-2)]

The models

- Instead of moving *data to models*
  (e.g. Graphs into vectors or trees into sequences, with alignment problems, loose of information, etc.)
  we move *models to data*

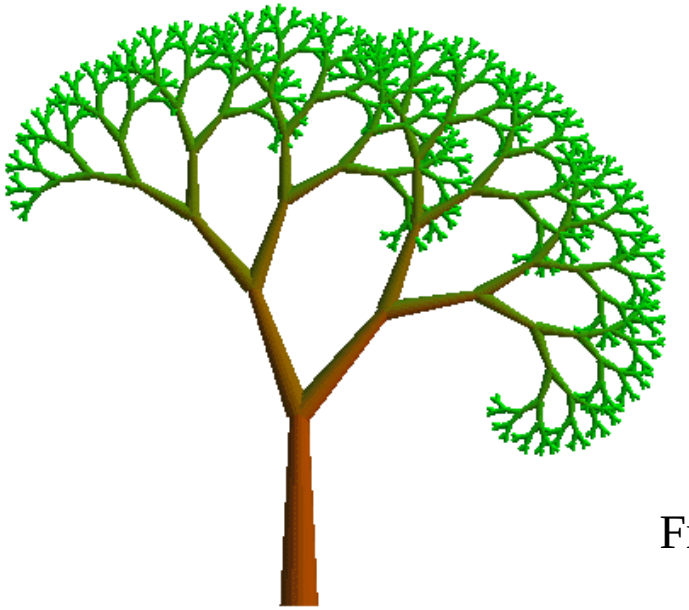
SD Learning scenario

Model / Data Type	Symbolic	Connectionist	Probabilistic
STATIC attribute/value, real vectors	Rule induction, Decision trees	NN, SVM	Mixture models, Naïve Bayes
SEQUENTIAL serially ordered entities	Learning finite state automata	Recurrent NN	Hidden Markov Models
STRUCTURAL relations among domain variables	Inductive logic programming	Recursive NN (Kernels forSD)	Recursive Markov models

Preview: The RecNN idea

Recursive NN:

- **Recursive** and parametric realization of the transduction function
  - In other words: Node embedding by a neural state machine
- *Adaptive* by Neural Networks



We will see how RecNNs extend RNNs matching the recursive nature of trees

Fractal tree: a recursive structure

Neural Computing Approach

- NN are universal approximators (Theorem of Cybenko)
- NN can learn from example (automatic inference)
- NN can deal with noise and incomplete data
- NN can handle continuos real and discrete data
- Simple gradient descent technique for training
- Successful model in ML due to the flexibility in applications

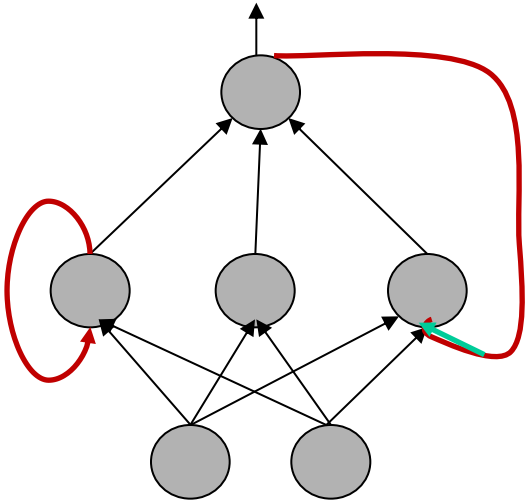
Domain	**Neural Network**
Static fixed-dim patterns (vectors, records, ...)	Feedforward
Dynamical patterns (temporal sequences, sequences ...)	Recurrent
Structured patterns (DPAGs, trees ...)	**Recursive**

Feedforward versus Recurrent *(memento)*

- **Feedforward**: direction: input→ output
- **Recurrent** neural networks: A different category of architecture, based on the addition of *feedback loops* connections in the network topology,
  - The presence of **self-loop** connections provides the network with dynamical properties, letting a memory of the past computations in the model.
  - This allows us to extend the representation capability of the model to the processing of sequences (and structured data).

Recurrent neural networks:

- They will be the subject
  (further developed) ISPR/CNS courses (see later).

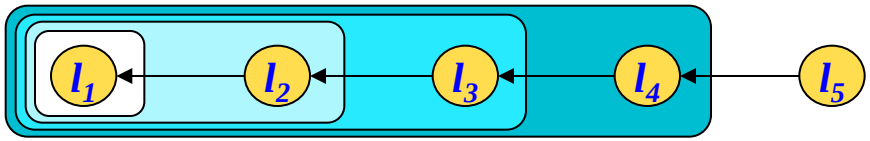


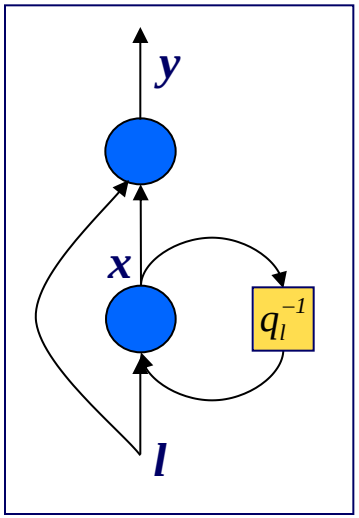
Recurrent Neural Networks (*resume*)

- Up to now:

Given ***x(0) =0***

internal state $\begin{cases} \boldsymbol{x}(t) = \tau(\boldsymbol{x}(t-1), \boldsymbol{l}(t)) \\ \boldsymbol{y}(t) = g(\boldsymbol{x}(t), \boldsymbol{l}(t)) \end{cases}$



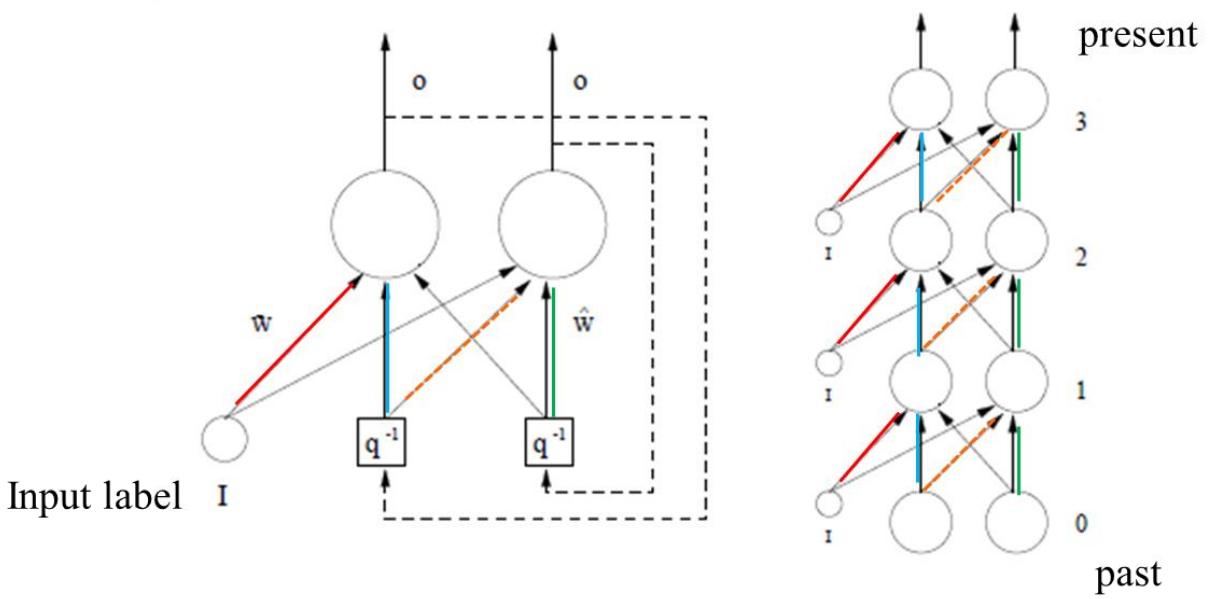


Graphical model

E.g. also **HMM**

RNN training and properties ***resume***

- BPTT/RTRL [see CNS course]
- Unfolding [see ML lecture RNN]:



Back-Prop Through Time: Backprop on this enrolled version

- Causality: A system is causal if the output at time t_0 (or vertex v) only depends on inputs at time $t<t_0$ (depends only on v and its descendants)
  - necessary and sufficient for *internal state*
- Stationarity: time invariance, state transition function τ is independent on node v (the same in any time)

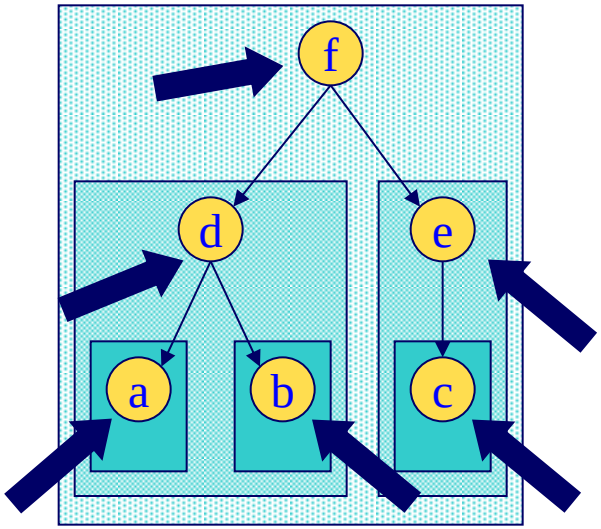
Recursive Neural Networks (overview)

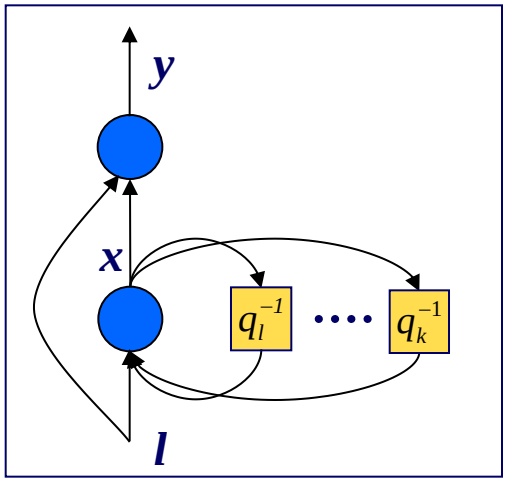
- Now:

$$\begin{cases} \boldsymbol{x}(v) = \tau(\boldsymbol{x}(\mathrm{ch}[v]), \boldsymbol{l}(v)) \\ \boldsymbol{y}(v) = g(\boldsymbol{x}(\mathrm{v}), \boldsymbol{l}(v)) \end{cases}$$

$$\boldsymbol{x}(\mathrm{ch}[v]) = \boldsymbol{x}(\mathrm{ch}_1[v]), \ldots, \boldsymbol{x}(\mathrm{ch}_k[v])$$

State transition sytem





 # feedbacks = # children

Recursive Neural Networks (overview)

More precisely (with initial conditions)

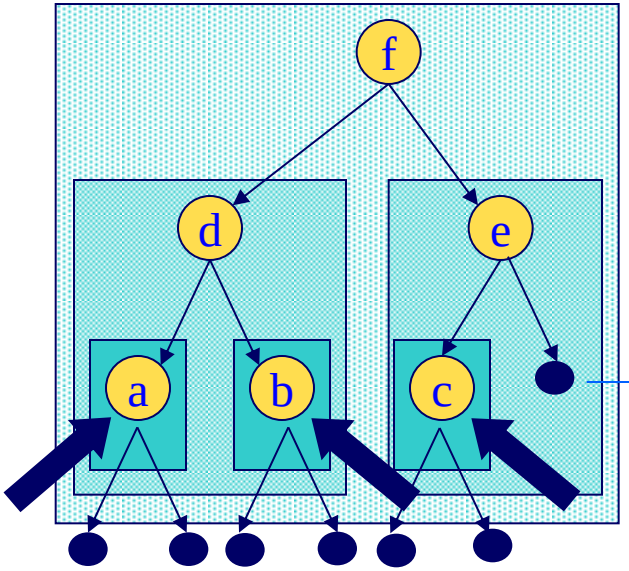
- Now:

Given $\boldsymbol{x}(nil) = 0$

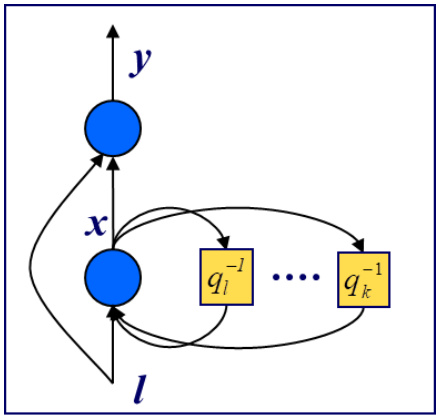
$$\begin{cases} \boldsymbol{x}(v) = \tau(\boldsymbol{x}(\mathrm{ch}[v]), \boldsymbol{l}(v)) \\ \boldsymbol{y}(v) = g(\boldsymbol{x}(\mathrm{v}), \boldsymbol{l}(v)) \end{cases}$$

$$\boldsymbol{x}(\mathrm{ch}[v]) = \boldsymbol{x}(\mathrm{ch}_1[v]), \ldots, \boldsymbol{x}(\mathrm{ch}_k[v])$$

State transition sytem



nil or $\boldsymbol{x}_0$
or empty node



 # feedbacks = # children

Generalized Shift Operators

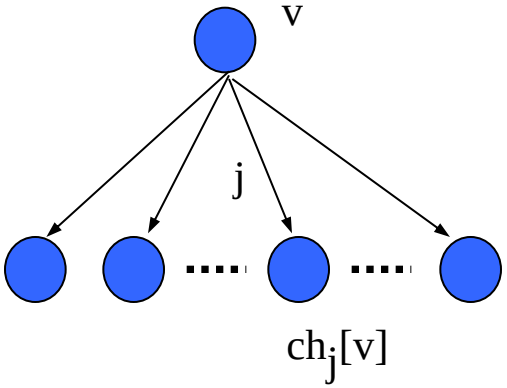
- Standard shift operator (time):

$$q^{-1}\, S_t = S_{t-1}$$

- Generalized shift operators (structure):

$$q_j^{-1}\, G_v = G_{ch_j[v]}$$

where $ch_j[v]$ is the j-th child of v



RecNN with q:

$$\begin{cases} \boldsymbol{x}(v) = \tau(\boldsymbol{l}(v), \boldsymbol{q}^{-1}\boldsymbol{x}(v)) \\ \boldsymbol{y}(v) = g(\boldsymbol{x}(v)) \end{cases}$$

$$\begin{cases} q_j^{-1} x_i(v) = x_0 & \text{if } \mathrm{ch}_j[v] = nil \\ q_j^{-1} x_i(v) = x_i(\mathrm{ch}_j[v])) & \text{otherwise} \end{cases}$$

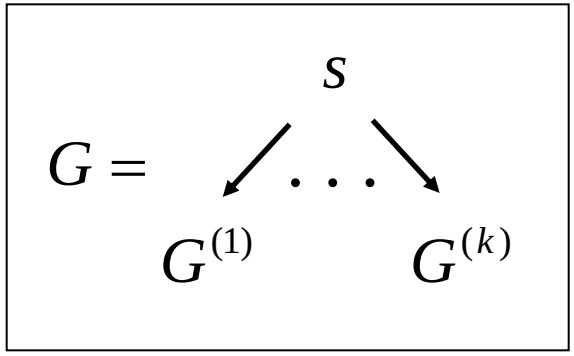
Used to extend the states for children of vertexes of the tree

Recursive Processing

Recursive definition of τ_E (encoding function)

Node/Graph embedding

$\boldsymbol{x}(root) = \tau_E(G)$



s can be either a root for a tree or a super-source for a DPAG

$$\tau_E(G) = \begin{cases} \boldsymbol{0} & \text{if } G \text{ is empty} \\ \tau_{NN}(L_{root}, \tau_E(G^{(1)}),\ldots,\tau_E(G^{(k)})) \end{cases}$$

τ_E : systematic visit of $G \rightarrow$ it guides the application of τ_{NN} to each node of tree (bottom-up). *Causality* and *stationary* assumption.

Properties or RecNN (I)

Extension of *causality* and *stationarity* defined for RNN:

- **Causality:** the output for a vertex v only depends on v and its descendants (induced subgraphs)
  - Compositionality!

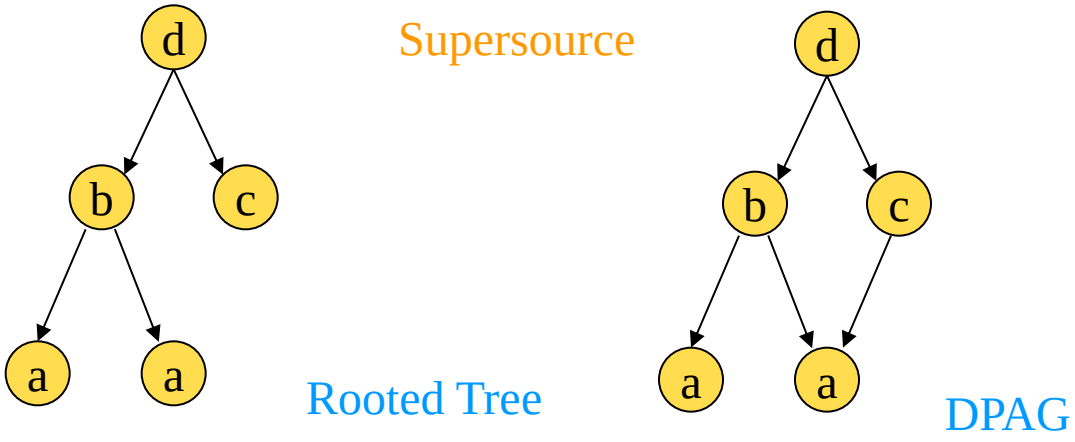
- **Stationary**: state transition function τ_{NN} is independent on vertex v
  - Parsimony: we use the same NN for each vertex

Recurrent/recursive NN transductions admit a recursive state representation with such properties

- **Adaptivity** (NN. learn. Alg.) **+ Universal approximation** over the tree domain [Hammer 2005-2007]

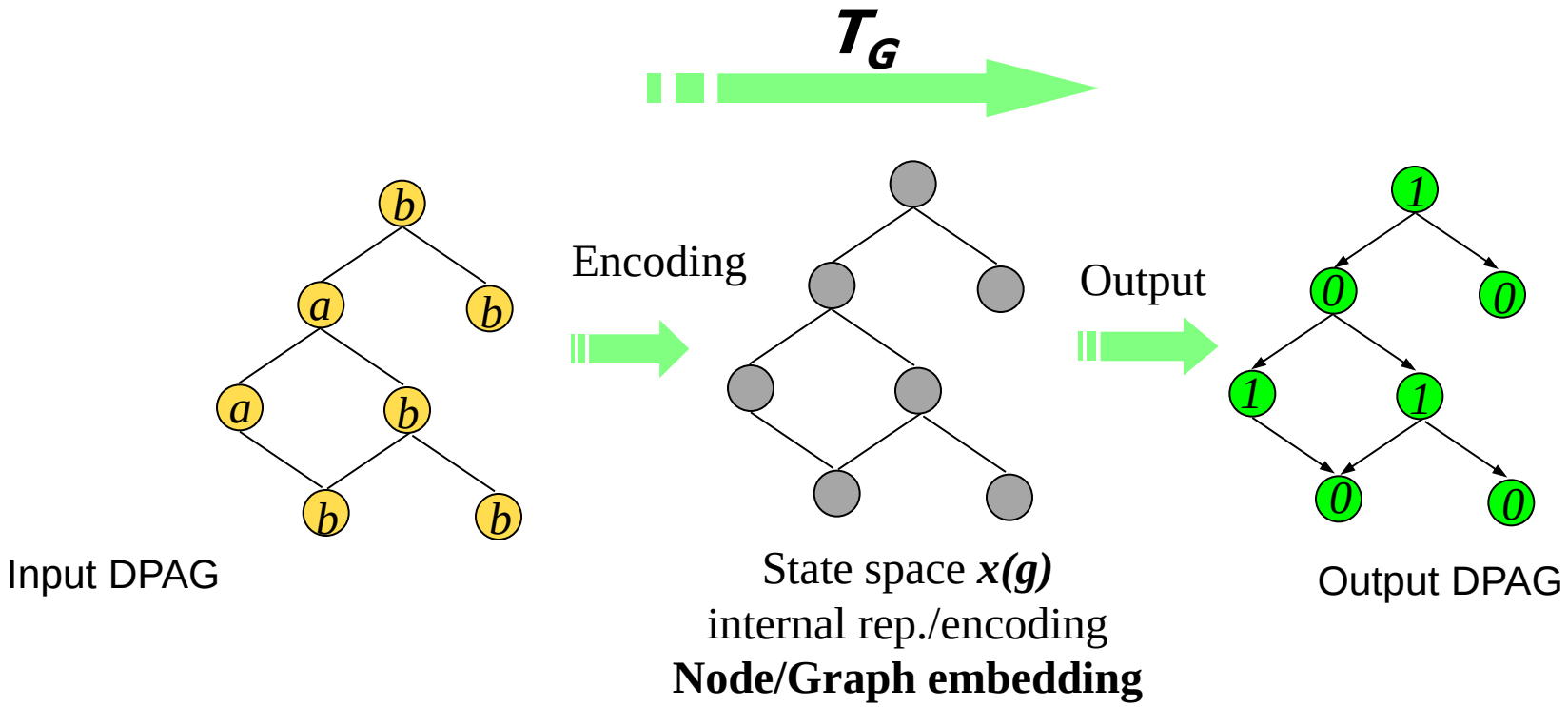
Kind of graphs for RecNN

- RecNN can in principle treats both Trees and DOAGs/DPAGs
  - If it can discriminates completely also the DOAG/DPAGs will treated later
  - But if there are no cycles the *recursive* model can visit the input DOAG/DAPG without special care



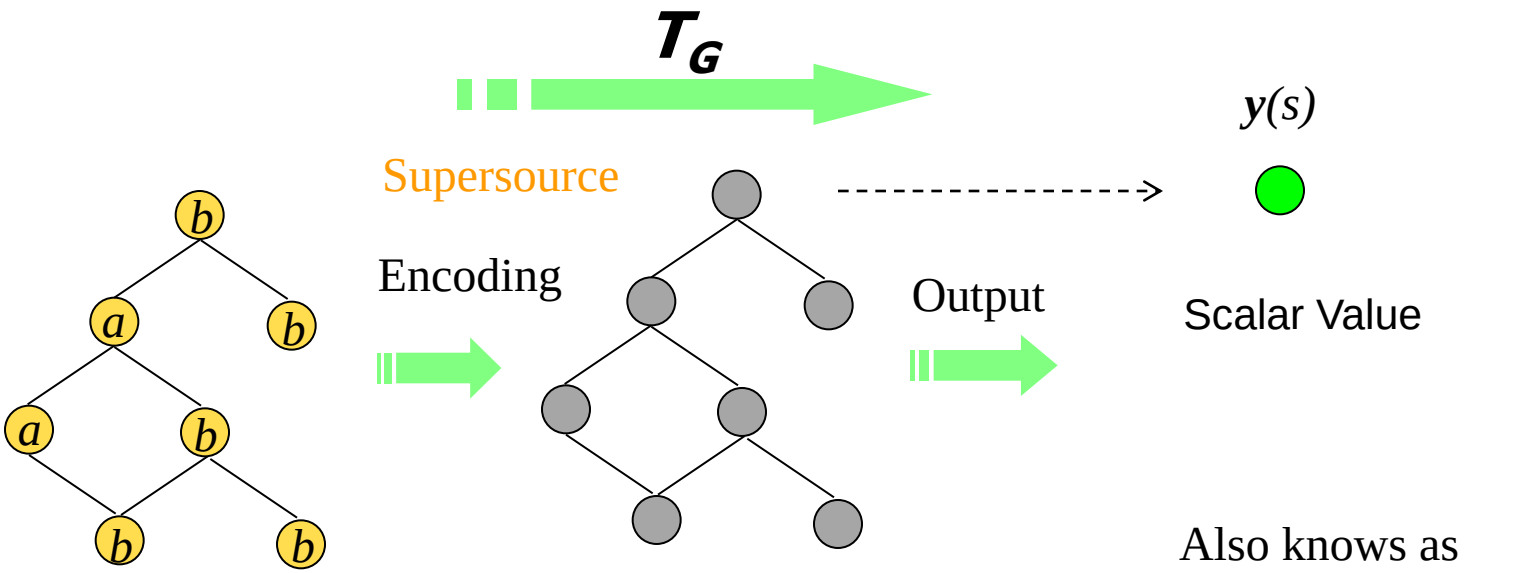
Properties (II): graphical view

- T_G is *IO-isomorph* if G and $T_G(G)$ have the same skeleton (graph after removing labels)



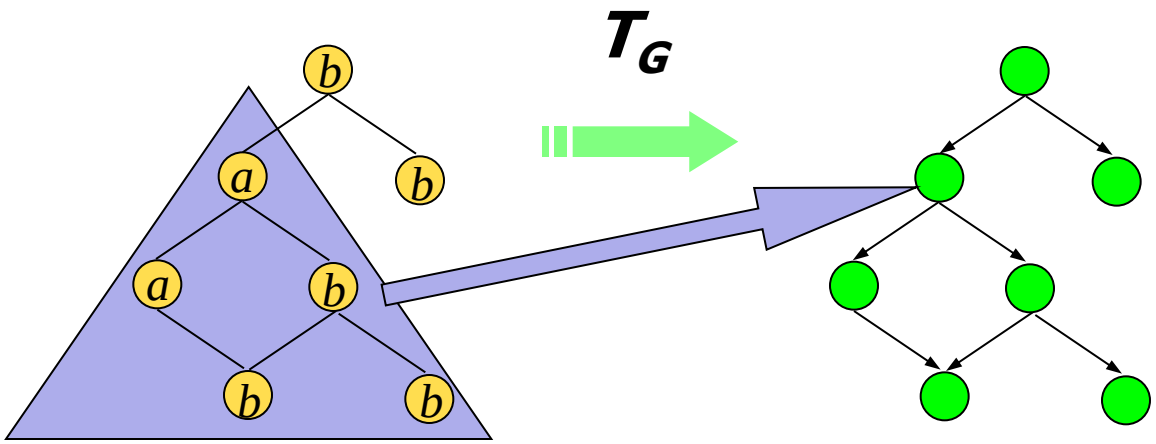
Properties (II): graphical view

- T_G *Supersource* transductions



Properties (III)

- IO-isomorph **causal** transduction



Only the sub-structure is considered

Unfolding and Enc. Network

- We will see through RecNN data flow process with two points of view:

1. Unfolding by "Walking on structures" model (*stationarity*), according to *causal* assumption (inverse topological order*).
   - The model visits the structures

2. Building an Encoding network isomorphic to the input structure (same skeleton, inverse arrows, again with *stationarity*) and *causal* assumption):
   - We build a different encoding network for each input structure

* see next slide

Topological Order

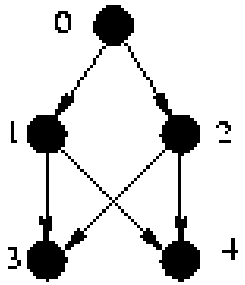
- A linear ordering of its nodes s.t. each node comes before all nodes to which it has edges. Every DAG has at least one topological sort, and may have many.

- A numbering of the *vertices* of a *directed acyclic graph* such that every *edge* from a vertex numbered i to a vertex numbered j satisfies i<j.

According to a Partial order

For RNN:

Inverse topological order

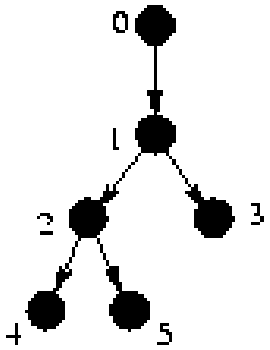


0,1,2,3,4

0,1,2,4,3

0,2,1,3,4

0,2,1,4,3



0,1,3,2,4,5

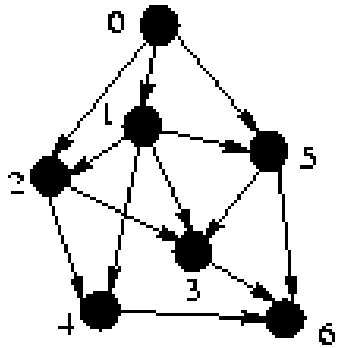
0,1,3,2,5,4

0,1,2,3,4,5

0,1,2,3,5,4

...

0,1,2,5,4,3



0,1,2,4,5,3,6

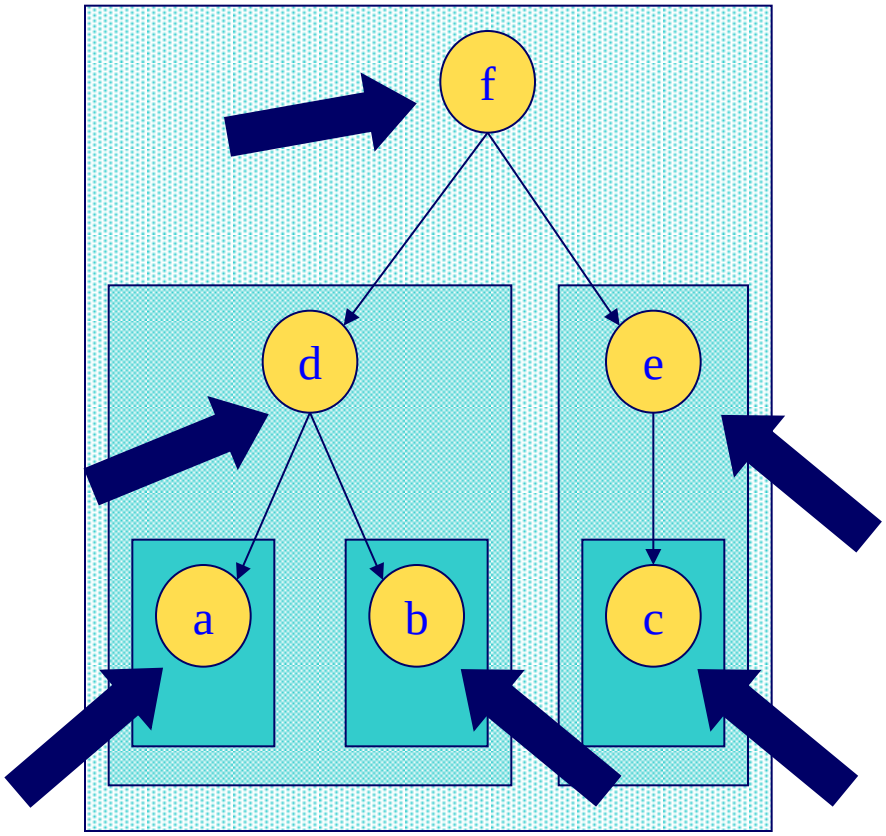
0,1,2,5,3,4,6

0,1,2,5,4,3,6

0,1,5,2,3,4,6

0,1,5,2,4,3,6

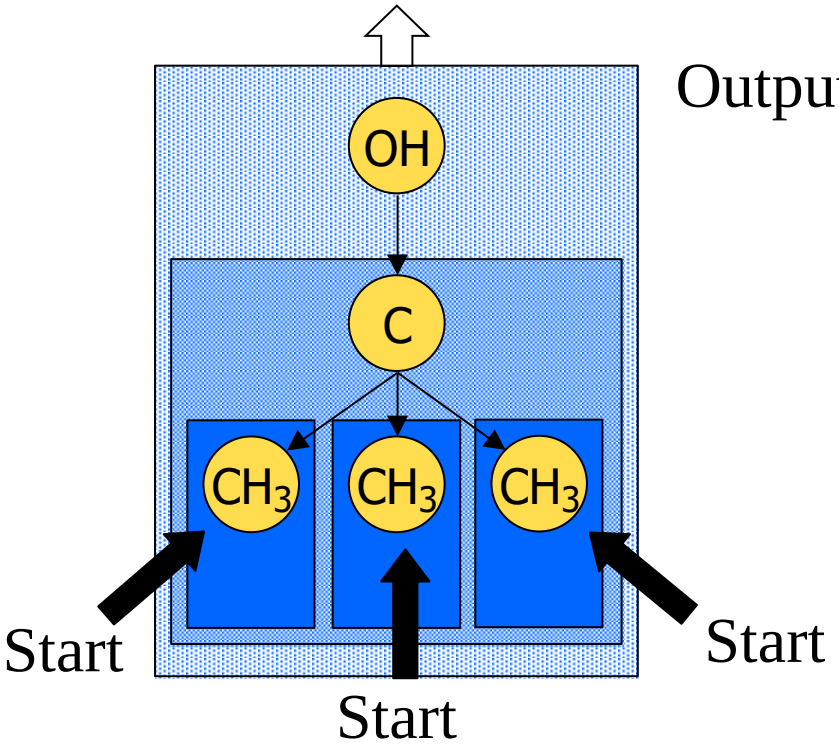
Unfolding & Encoding Process:
Unfolding view (1)



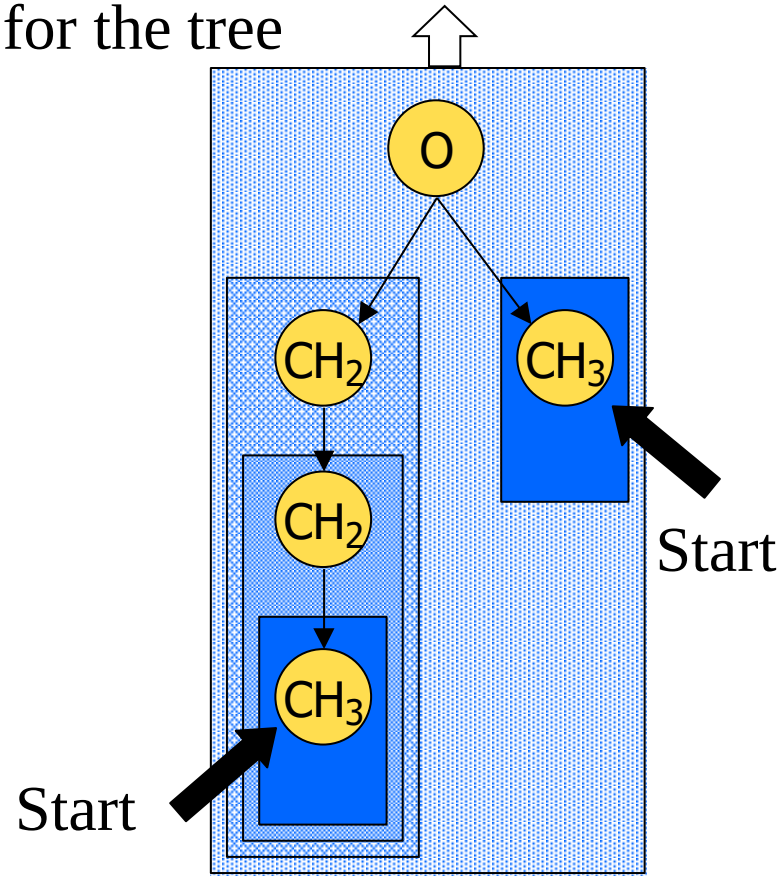
- Unfolding the encoding process trough structures
- Bottom-up process for visiting

- We will see later how to made it with τ_{NN} (and hence by NN) for each step: to build an *encoding network*

RecNN over different structures: unfolding 2



Output for the tree

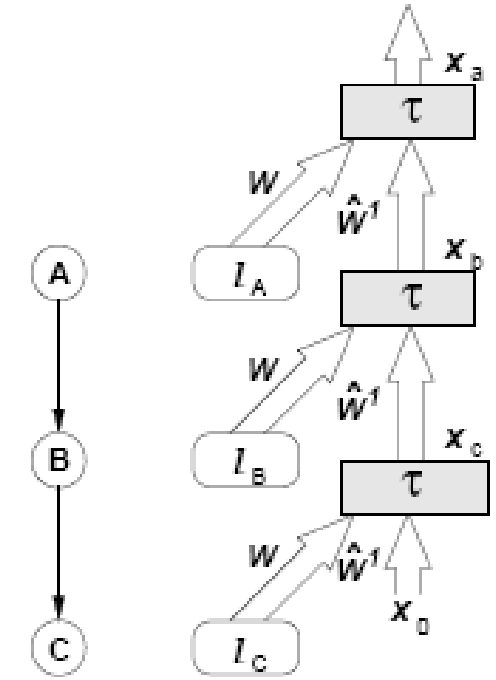


Examples on different trees for chemical compounds:

Unfolding through structures:

The same process apply to all the vertices of a tree and for all the trees in the data set

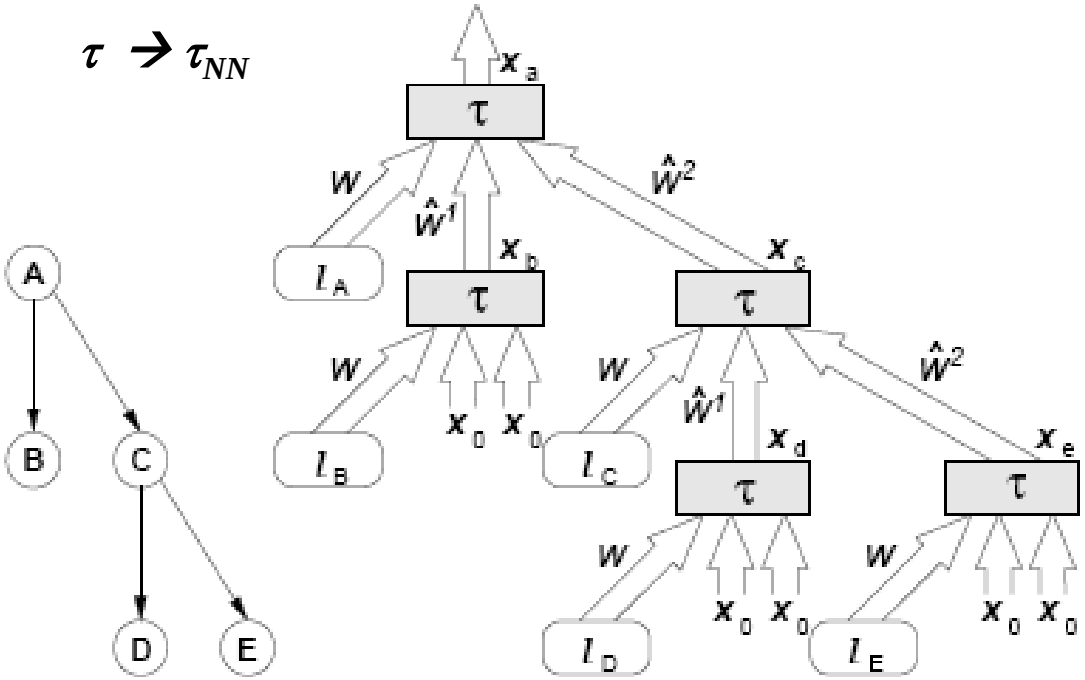
Unfolding (3) & Enc. Net. For Recurrent and Recursive NN
(or Recursive unfold. for two different structures)



Structured Data

Encoding Network

$\tau \rightarrow \tau_{NN}$



Structured Data

Encoding Network

- τ_{NN} for each step (with weight sharing): ***encoding network***
- Adaptive encoding via the free parameters of τ_{NN}

τ_{NN} (and weight) sharing : units are the same for the all the vertices of a tree and for all the trees in the data set!!!

RecNN: going to details for the domains

- $\boldsymbol{X} = IR^m$ continuos state (code) space (encoded subgraph space)
- $\boldsymbol{L} = IR^n$ vertex label space
- $\boldsymbol{O} = IR^z$ or $\{0,1\}^z$
- $\tau =$ $\tau_{NN} : IR^n \times \underbrace{IR^m \times \ldots \times IR^m}_{k \text{ times}} \rightarrow IR^m$ Subgraph code
- g output function
- $x_0 = \mathbf{0}$

$$T_G : G \rightarrow O$$

$$G \xrightarrow{\tau_E} IR^m \xrightarrow{g} IR \qquad (T_G)$$

- τ and g realized by NN with free parameters **W**

RNN realize $g \circ \tau_E$

Realization of τ_{NN}

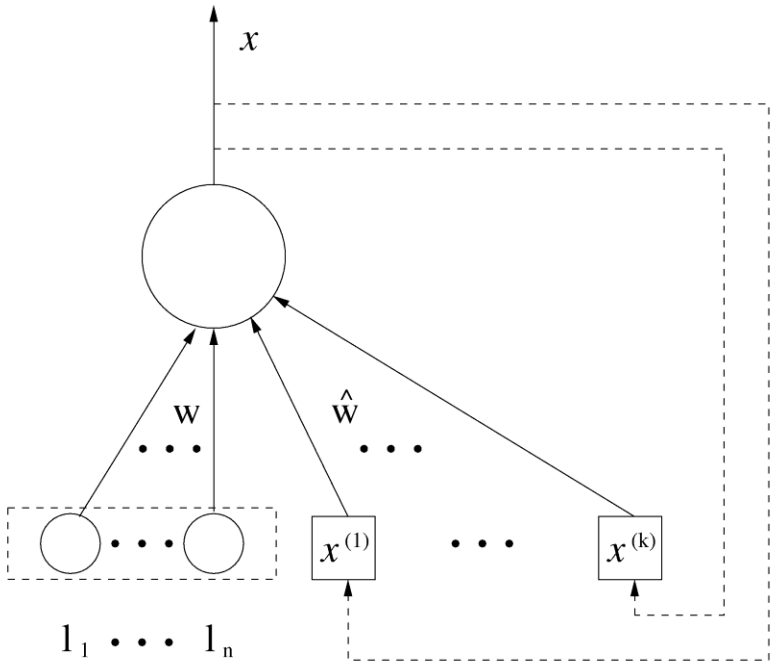
$$\tau_{NN} : IR^n \times \underbrace{IR^m \times \ldots \times IR^m}_{k \text{ times}} \rightarrow \boxed{IR^m}$$

$$\boldsymbol{x} = \tau_{NN}(\boldsymbol{l}, \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(k)}) = \sigma(\boldsymbol{W}\boldsymbol{l} + \sum_{j=1}^{k} \hat{\boldsymbol{W}}_j \boldsymbol{x}^{(j)} + \theta)$$

Subgraph code

m x m

Free parameters



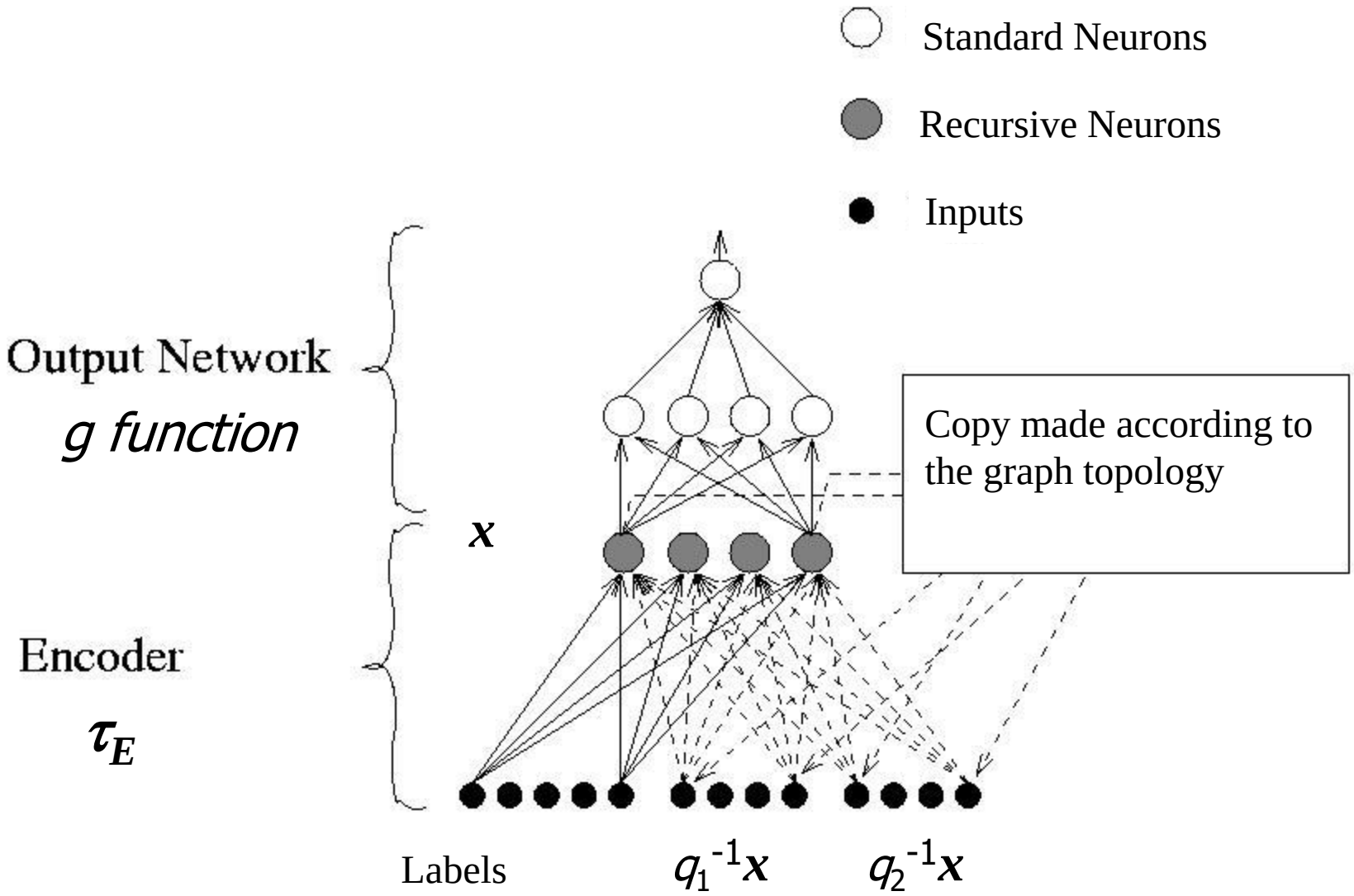
Recursive neuron (τ_{NN} with m=1)

Process a vertex

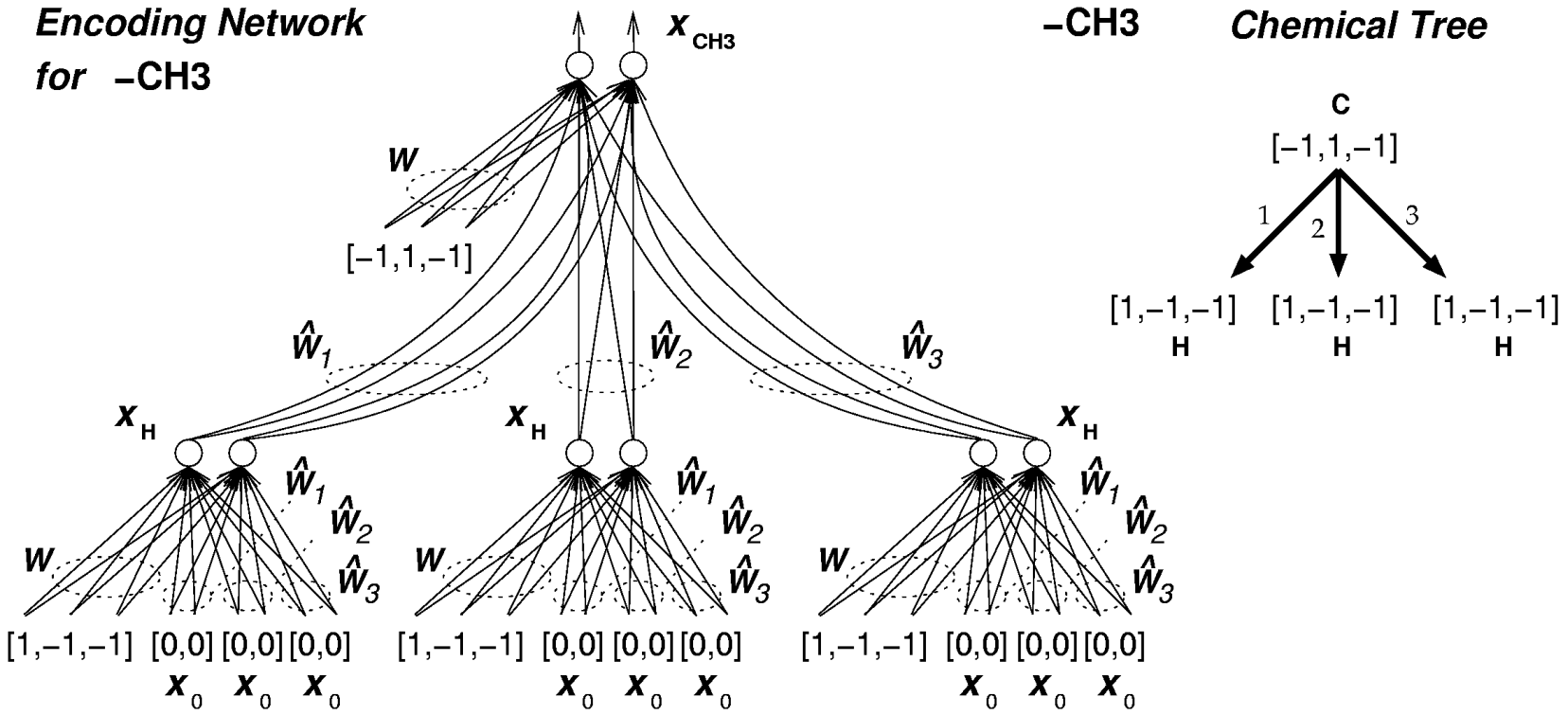
 # feedbacks = # children

(max *k*)

Fully-connected RNN



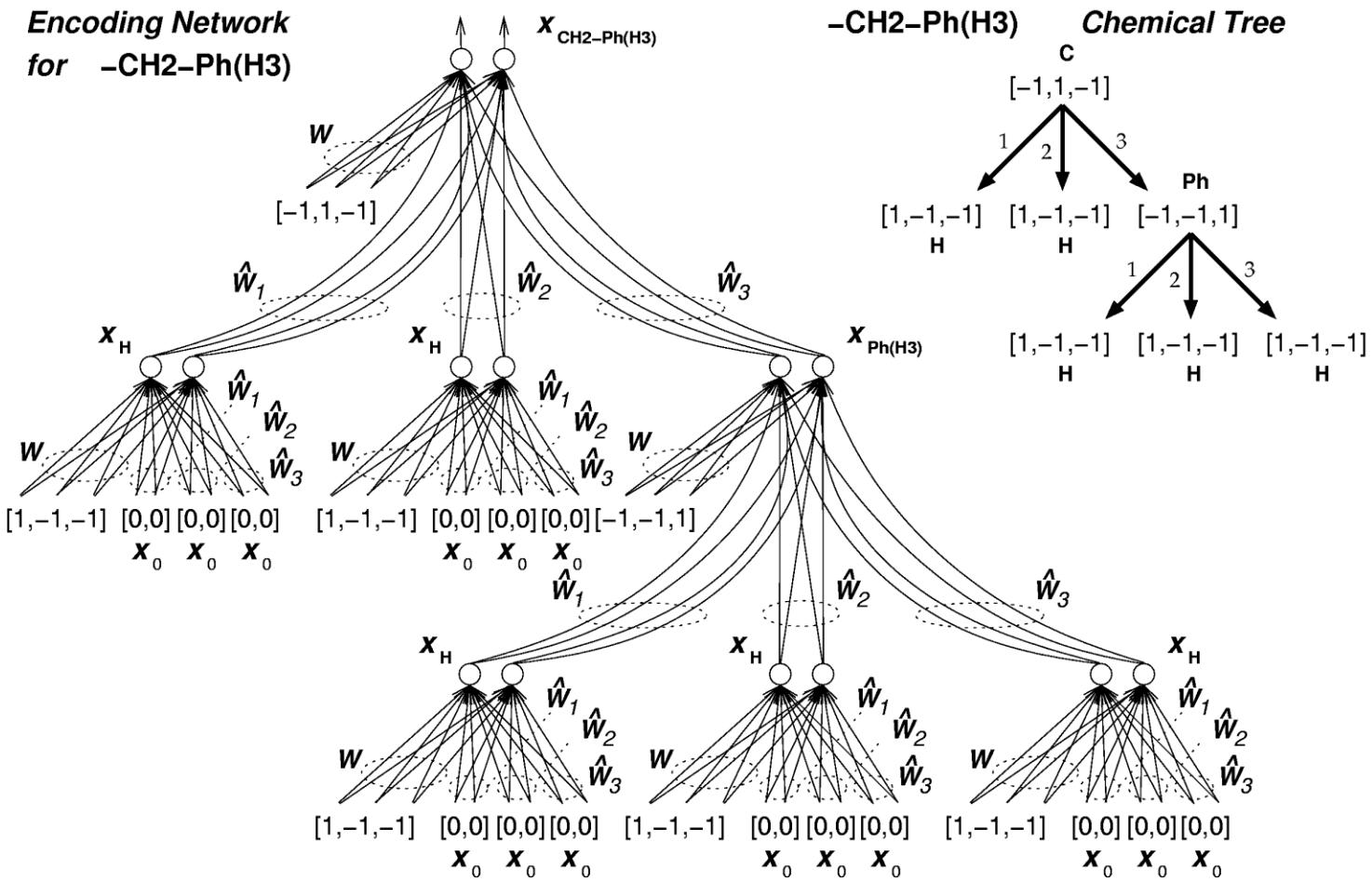
In details: Encoding Network (I)



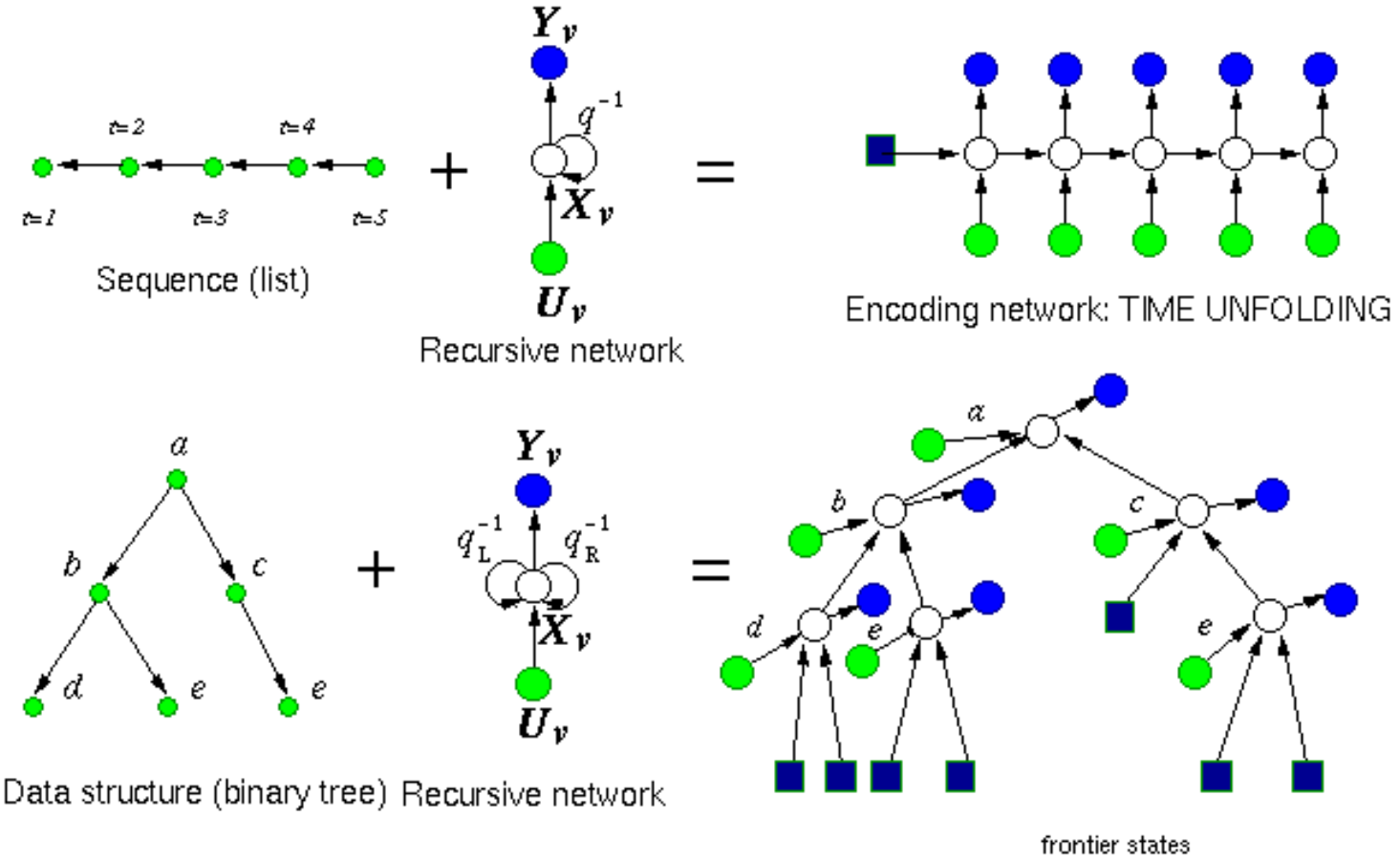
Start here !

Tau (and **weight**) sharing : units are the same for the all the vertices of a tree and for all the trees in the data set!!!

In details: Encoding Network (II)



Recap: Unfolding 4. A different view by graphical models of the Encoding Networks for Seq. and Structures



Note that the use of graphical models make uniform the cases of NN (RNN & RecNN) and generative approaches (HMM/HTreeMM)

RecNN applications

- Representing **hierarchical** information in many real-world domains
- Many examples:
  - Molecular Biology
  - Document (**XML**) Processing
  - **Natural Language Processing**
    - E.g. Stanford NLP group shown the effectives of RecNN applied to tree representation of language (and images) data and tasks.
    - Sentiment Tree Bank
    - Next slides

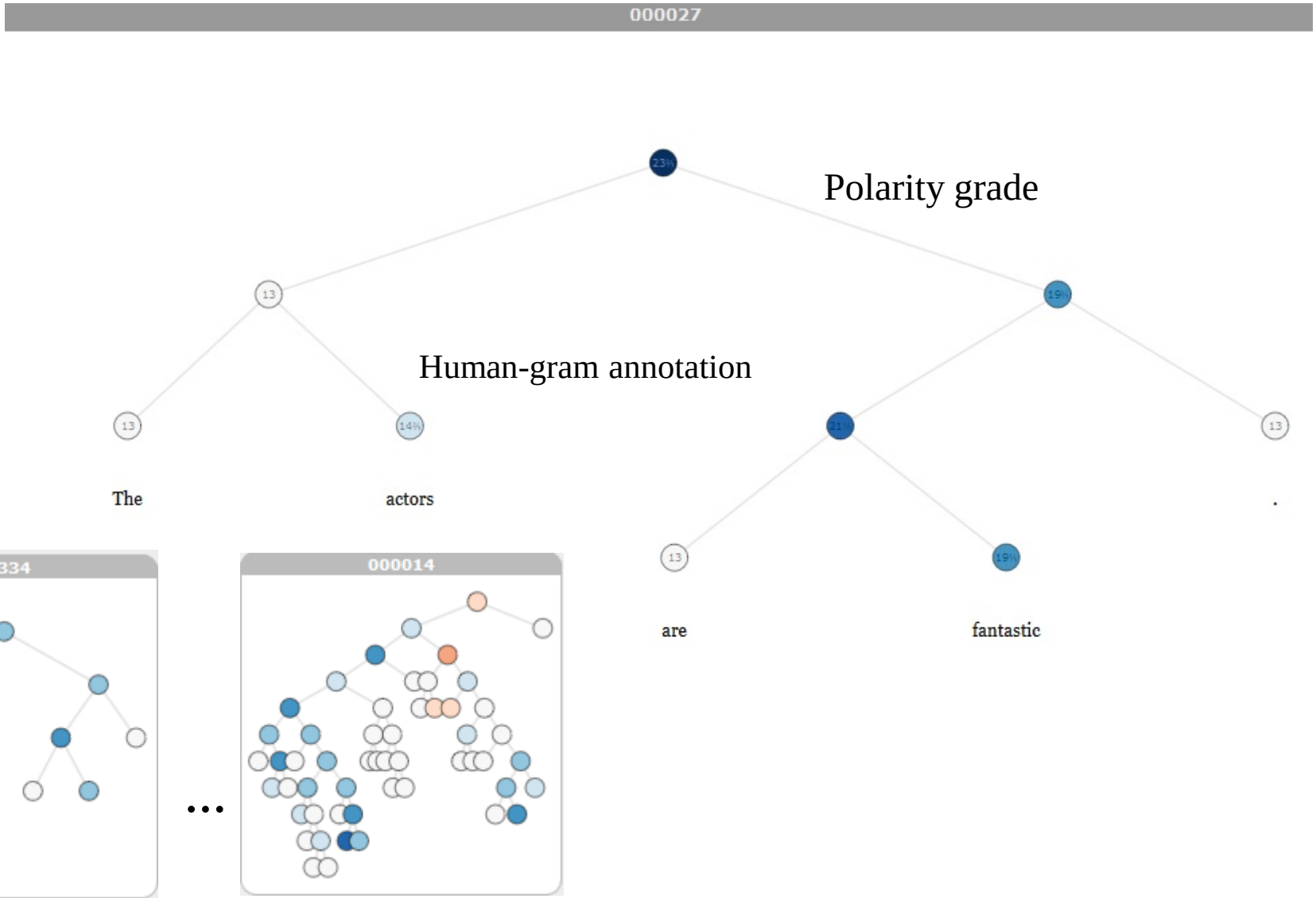
Recent Applications: Repetita (da ML): RecNN for NLP recent exploitment

- Currently wide successful application in NLP (e.g. by the Stanford NLP group)

- Shown the effectives of RecursiveNN applied to tree representation of language (and images) data and tasks. Started in 2011-13

- E.g. Sentiment Treebank
  - Sentiment labels (movies reviews) for 215,154 phrases in the **parse trees** of **11,855** sentences
  - Recursive NN pushes the state of the art in single sentence positive/negative classification from 80% up to 85.4%.

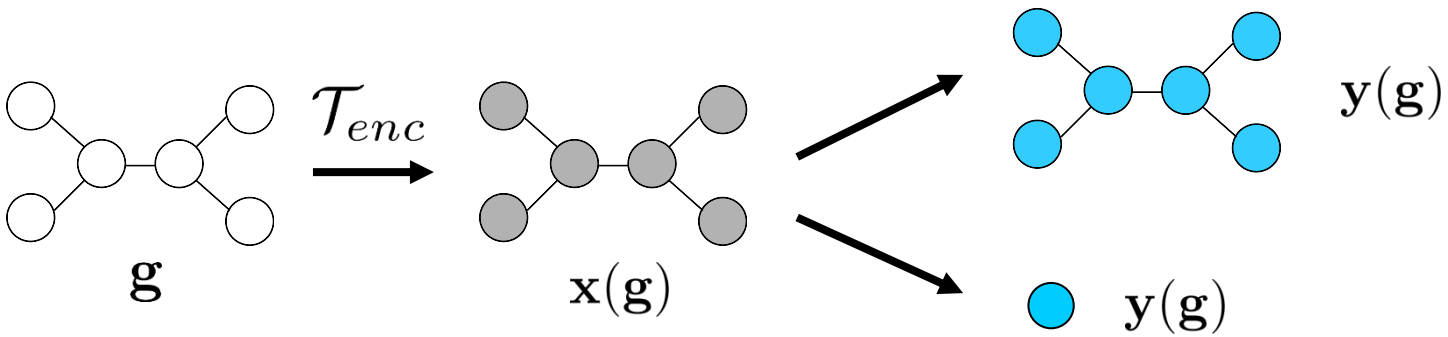
Repetita (da ML): Examples



Learning Aims

- ***Parametric : T_G*** depends on tunable parameters ***W.***
- With different possible aims:

Structure-to-Structure
(Input-Output isomorphic)



Input graph

Possible internal representation/encoding

Node/Graph embedding

Structure-to-Scalar/Element
(regression/classification)

Or (in general) it can also be **non-isomorphic**

RNN Learning Algorithms

- Backpropagation through structure: Extension of BPTT
  Goller & Küchler (1996)
  - Simple to understand using graphical formalism (backprop+weight sharing on the unfolded net) :
  - *The notation is adapted to the case of delta form the fathers*

- RTRL Sperduti & Starita (1997),

✓ Equations: See Cap 19 in

Kolen, Kremer , *A Field Guide to* ***Dynamical Recurrent Networks***.
IEEE press 2001

- RCC family based: next slides

Learning in Structured Domain
Plan in 2 lectures

1. Recurrent and Recursive Neural Networks

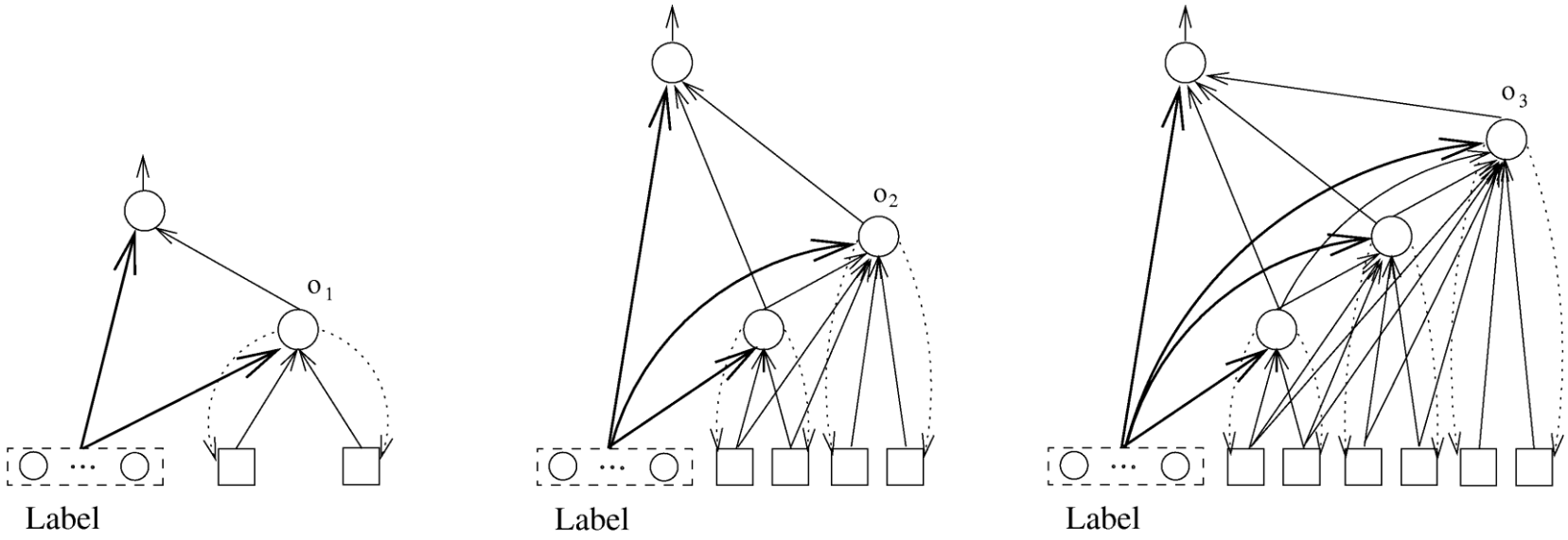
   Extensions of models for learning in structured domains

   - Motivation and examples (structured data)
   - The structured data (recursive)
   - Recursive models: RNN and RecNN
   - **Recursive Cascade Correlation & other recursive approaches**

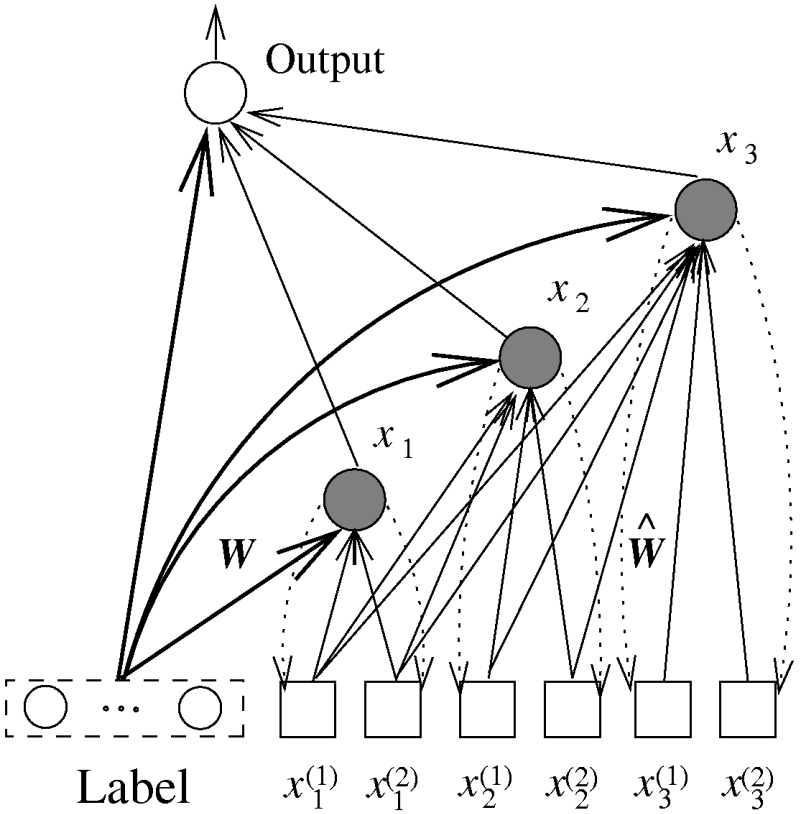
2. Moving to DPAG and Graphs: the role of causality [Next Lecture (SD-2)]

RCC (I)

- Architecture of Cascade Correlation for Structure (Recursive Cascade Correlation - RCC)
- We realize RNN by RCC: **constructive approach** $\Rightarrow$ *m* is automatically computed by the training algorithm.
- *A deep neural network!*



RCC (II)



Architecture of a RCC with 3 hidden units (**m=3**) and **k=2** . Recursive hidden units (shadowed) generate the code of the input graph (function τ_E). The hidden units are added to the network during the training. The box elements are used to store the output of the hidden units, i.e. the code $x_i^{(j)}$ that represent the context according to the graph topology.

The output unit realize the function **g** and produce the final prediction value.

E.G. A.M. Bianucci, A. Micheli, A. Sperduti, A. Starita.
Application of Cascade Correlation Networks for Structures to Chemistry,
Applied Intelligence Journal. 12 (1/2): 117-146, 2000

RCC (III) Learning

Gradient descent: interleaving LMS (output) and maximization of correlation between new hidden units and residual error. Main difference with CC: calculation of the following derivatives by recurrent equations:

$$\frac{\partial x_h}{\partial w_{hi}^j} = \frac{\partial \tau_h(\boldsymbol{l}, \boldsymbol{x}^{(1)}, \cdots, \boldsymbol{x}^{(k)})}{\partial w_{hi}^j} = f'\left(l_i + \sum_{t=1}^{k} \hat{w}_{hh}^t \frac{\partial \boldsymbol{x}_h^{(t)}}{\partial w_{hi}^j} \right)$$

$$\frac{\partial x_h}{\partial \hat{w}_{hi}^j} = \frac{\partial \tau_h(\boldsymbol{l}, \boldsymbol{x}^{(1)}, \cdots, \boldsymbol{x}^{(k)})}{\partial \hat{w}_{hi}^j} = f'\left(\boldsymbol{x}_i^{(j)} + \sum_{t=1}^{k} \hat{w}_{hh}^t \frac{\partial \boldsymbol{x}_h^{(t)}}{\partial \hat{w}_{hi}^j} \right)$$

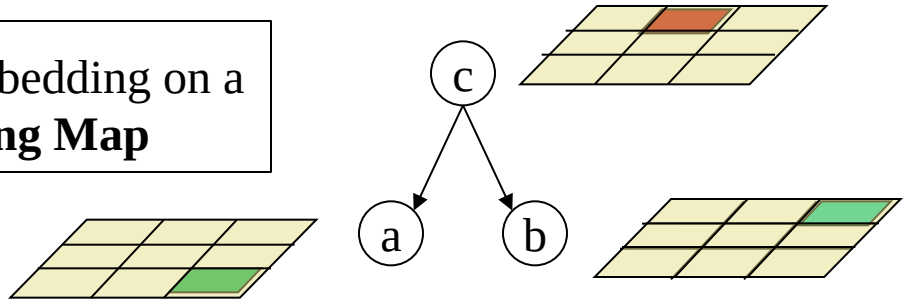
Note: simplification (of the sum on other units) due to the architecture compared to full RTRL! But compared to the recurrent case it appears the summation on children appears!

Unsupervised recursive models (2003-2005)

- Transfer *recursive* idea to unsupervised learning
- No prior metric/pre-processing (but still bias!)
- Evolution of the similarity measure through *recursive comparison* of sub-structures

- Iteratively compared via bottom-up encoding process

Recursive nodes embedding on a **Self-Organizing Map**

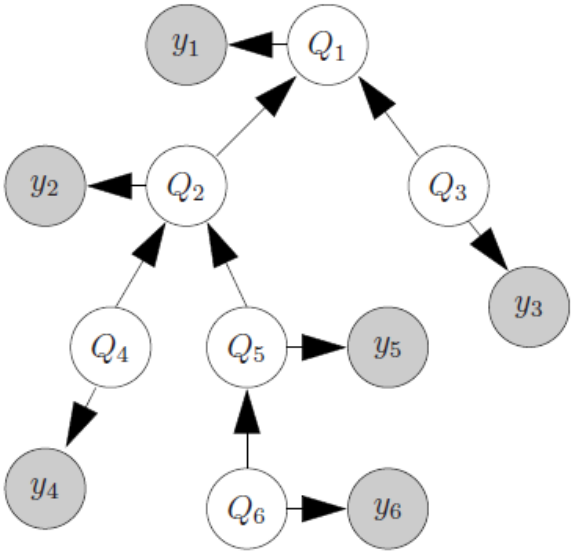


It uses e.g. the SOM coordinates for node embedding

M. Hagenbuchner et al. IEEE TNN, 2003
B. Hammer et al. Neural Networks, 2005

Generative: HTMM (2012-2018)

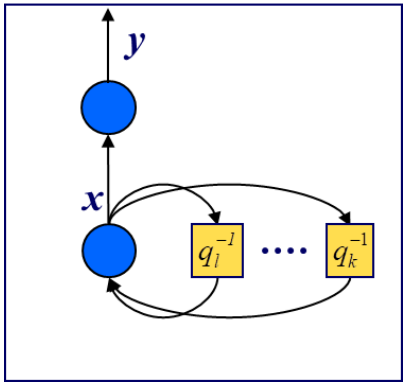
- E.g Bottom-up Hidden Tree Markov Models extend HMM to trees exploiting the recursive approach



Bayesian network unfolding graphical model over the input trees; y: observed elements
Q: hidden states variables with discrete values

- Generative process from the leaves to the root
- Markov assumption (conditional dependence)
  $Qch1(u), \ldots, Qch_K(u) \rightarrow Qu$

Children to parent hidden state transition
$P(Qu \mid Qch1(u), \ldots, Qch_K(u))$

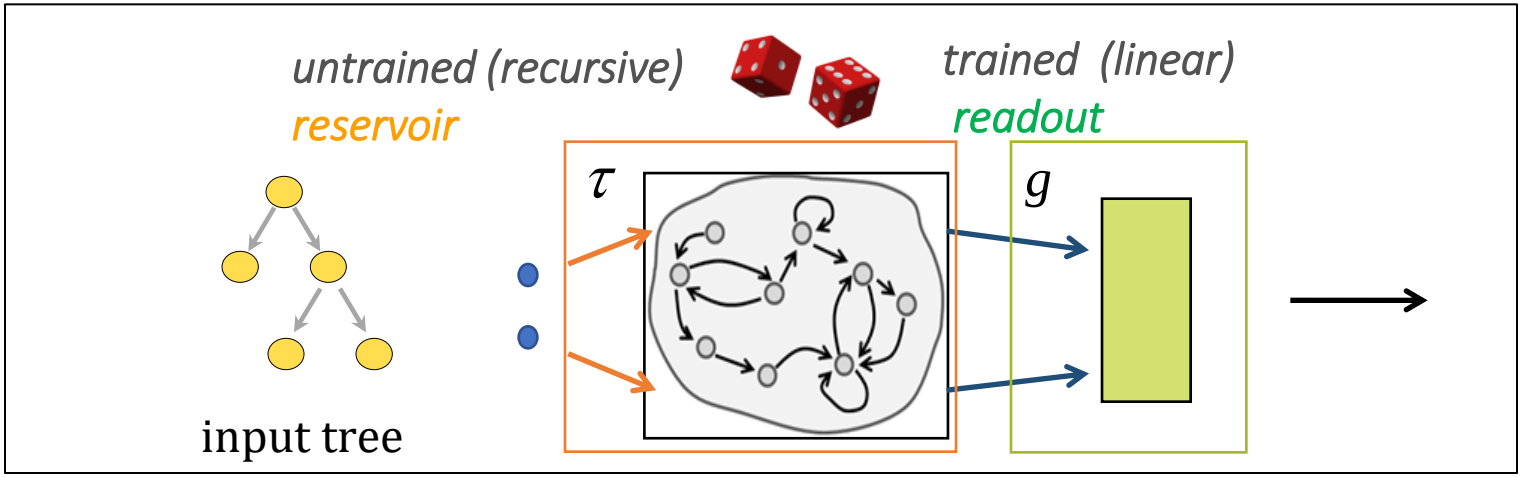


Issue: how decompose this joint state transition? (see ref.).

Bacciu, Micheli, Sperduti. IEEE TNNLS, 2012

Efficient: TreeESN (2010-13)

- Combine Reservoir Computing (un-trained layer of recurrent units with linear redout) and recursive modeling
  - Extend the applicability of the RC/ESN approach to tree structured data
  - Extremely efficient way of modeling RecNNs (randomized approaches)
  - Architectural and experimental performance baseline for trained RecNN models with often comptetive results.



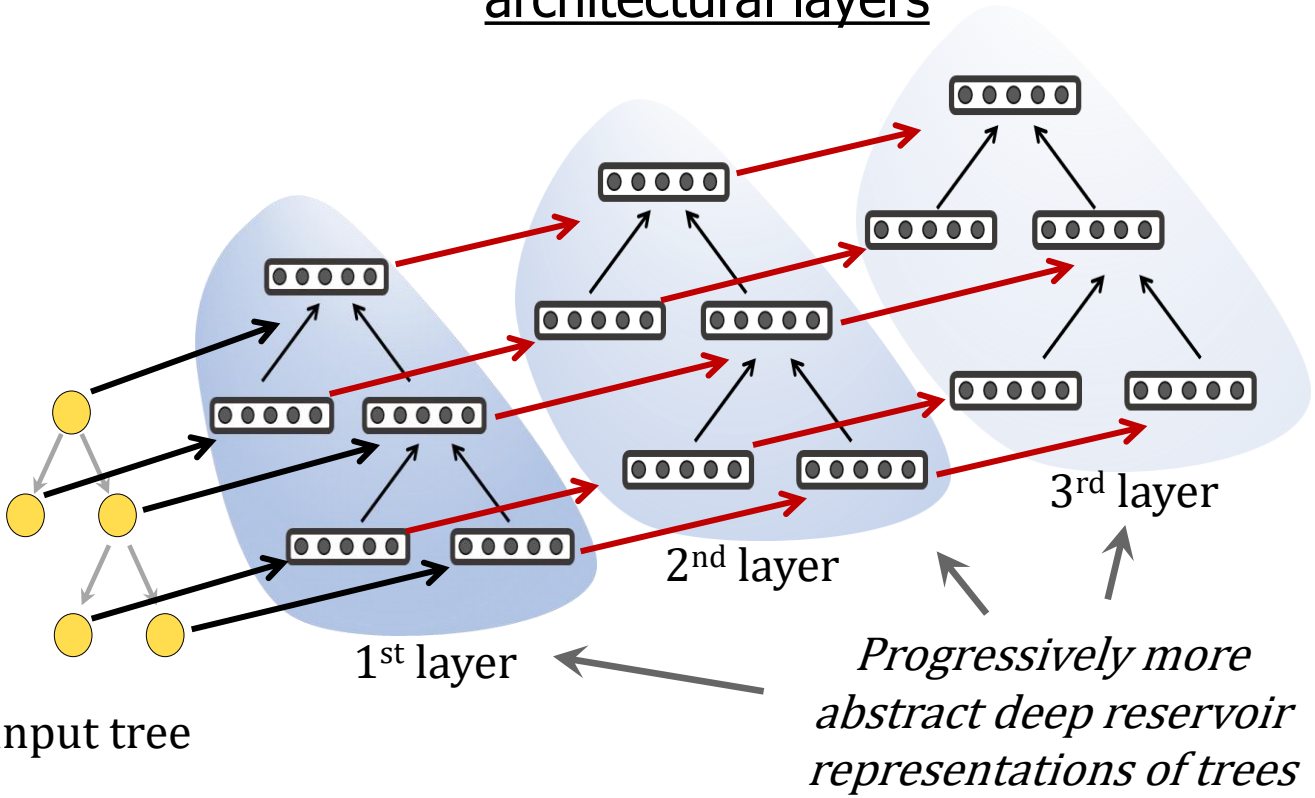
The recursive process of RecNN made by efficient RC approaches

C. Gallicchio, A. Micheli. Neurocomputing, 2013.

More in a NEXT LECTURE

Deep: Deep Tree ESN (2018)

Hierarchical abstraction **both** through the input structure and architectural layers



- Improve efficiency (giving same #units)
- Improve results

C.Gallicchio, A.Micheli IEEE IJCNN 2018
C.Gallicchio, A.Micheli Information Sciences 2019

Learning in Structured Domain
Plan in 2 lectures

1. Recurrent and Recursive Neural Networks

   Extensions of models for learning in structured domains
   - Motivation and examples (structured data)
   - The structured data (recursive)
   - Recursive models: RNN and RecNN
   - Recursive Cascade Correlation & other recursive approaches

2. Moving to DPAG and Graphs: the role of causality [Next Lecture (SD-2)]

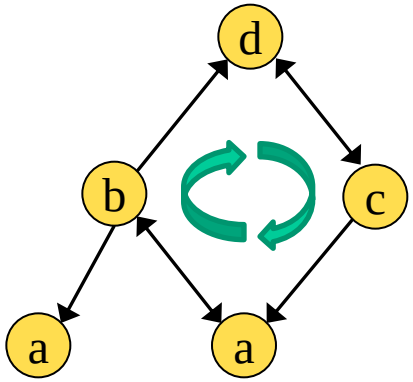
Toward next lecture

RecNN Analysis

- RecNNs allow adaptive representation of SD
  - handling of variability by causality and stationarity
  - **Adaptive** transduction: BPTS, RTRL, ...
  -
- **Stationarity**:
  - efficacy solution to parsimony (reducing the number of paratmetrs) without reducing expressive power
- **Causality:** affects the computational power !
  - RNN are only able to memorize past information (sub-sequences)
  - RecNN outputs depend only on sub-structures
  - The domain is restricted to sequences and trees due to causality
  - Toward partial relaxation (or **extension**) of the causality assumption

Graphs by NN?

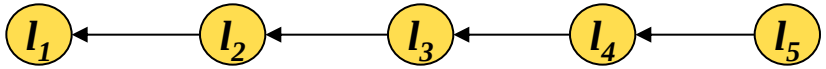
- For Graphs by NN: see next lecture!
- Following a journey through the causality assumption!



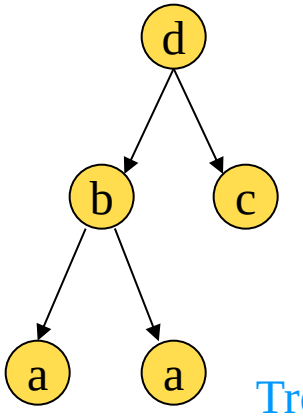
How to deal with cycles and causality?

MODELS panorama for SD (examples)

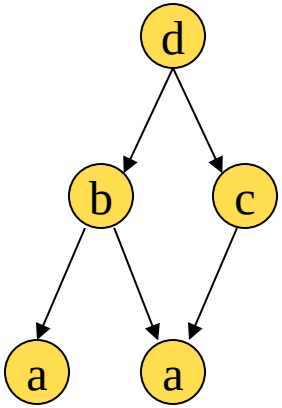
Standard ML models for flat data



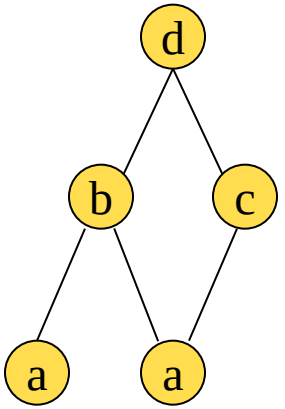
- Recurrent NN/ESN
- HMM
- Kernel for strings …



Tree:
- Recursive NN
- Tree ESN
- HTMM
- Tree Kernels
- …



DPAG:
- CRCC



- GNN/GraphESN
- NN4G
- Graph Kernels
- SRL
- …

See references for models in the bibliography slides (later)

Bibliography: aims

Different parts in the following:

- Basic/Fundamentals
* Possible topic for seminars
- May be useful also for future studies

  - Many topics can be subject of study and development
  - Many many works in literature (arrive continuously)!
  - Many possible topics for demand and possible thesis

  - **More bibliography on demand:** micheli@di.unipi.it

Bibliografia (Basic, origins of RecNN)

RecNN

- A. Sperduti, A. Starita. *Supervised Neural Networks for the Classification of Structures*,IEEE Transactions on Neural Networks. Vol. 8, n. 3, pp. 714-735, 1997.

- P. Frasconi, M. Gori, and A. Sperduti, *A General Framework for Adaptive Processing of Data Structures*, IEEE Transactions on Neural Networks. Vol. 9, No. 5, pp. 768-786, 1998.

- A.M. Bianucci, A. Micheli, A. Sperduti, A. Starita. *Application of Cascade Correlation Networks for Structures to Chemistry*, Applied Intelligence Journal (Kluwer Academic Publishers), Special Issue on "Neural Networks and Structured Knowledge" Vol. 12 (1/2): 117-146, 2000.

- A. Micheli, A. Sperduti, A. Starita, A.M. Bianucci. *A Novel Approach to QSPR/QSAR Based on Neural Networks for Structures,*
Chapter in Book : "Soft Computing Approaches in Chemistry", pp. 265-296, H. Cartwright, L. M. Sztandera, Eds., Springer-Verlag, Heidelberg, March 2003.

Bibliography: NN approaches-2

* UNSUPERVISED RecursiveNN

- B. Hammer, A. Micheli, M. Strickert, A. Sperduti.
  A General Framework for Unsupervised Processing of Structured Data,
  Neurocomputing (Elsevier Science) Volume 57, Pages 3-35, March 2004.
- B. Hammer, A. Micheli, A. Sperduti, M. Strickert.
  Recursive Self-organizing Network Models. Neural Networks, Elsevier Science. Volume 17, Issues 8-9, Pages 1061-1085, October-November 2004.

* TreeESN: efficient RecNN

- C. Gallicchio, A. Micheli.
  Tree Echo State Networks, Neurocomputing, volume 101, pag. 319-337, 2013.
- C. Gallicchio, A. Micheli.
  Deep Reservoir Neural Networks for Trees. Information Sciences 480, 174-193, 2019.

* HTMM: further developments (generative)

- D. Bacciu, A. Micheli and A. Sperduti.
  Compositional Generative Mapping for Tree-Structured Data - Part I: Bottom-Up Probabilistic Modeling of Trees, IEEE Transactions on Neural Networks and Learning Systems, vol. 23, no. 12, pp. 1987-2002, 2012

Bibliography: RecNN applications (example)

* NLP applications (that you can extend with recent instances, and relate them to the general RecNN framework present in this lecture and the basic RecNN bibliography references)

- R. Socher, C.C. Lin, C. Manning, A.Y. Ng,
  Parsing natural scenes and natural language with recursive neural networks,
  Proceedings of the 28th international conference on machine learning (ICML-11)
- R. Socher, A. Perelygin, J.Y. Wu, J. Chuang, C.D. Manning, A.Y. Ng, C.P. Potts,
  Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank
  Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1631–1642, Seattle, Washington, USA, 18-21 October 2013

Bibliography: Next lecture

* RecNN for DPAGs : how to extend the domain (I)

- A. Micheli, D. Sona, A. Sperduti.
  Contextual Processing of Structured Data by Recursive Cascade Correlation. IEEE Transactions on Neural Networks. Vol. 15, n. 6, Pages 1396- 1410, November 2004.
- Hammer, A. Micheli, and A. Sperduti.
  Universal Approximation Capability of Cascade Correlation for Structures.
  Neural Computation. Vol. 17, No. 5, Pages 1109-1159, (C) 2005 MIT press.

* NN for GRAPH DATA: how to extend the domain (II)

- * A. Micheli. *Neural network for graphs: a contextual constructive approach*, IEEE Transactions on Neural Networks, volume 20 (3), pag. 498-511, doi: 10.1109/TNN.2008.2010350, 2009.

- C. Gallicchio, A. Micheli. *Graph Echo State Networks*, Proceedings of the International Joint Conference on Neural Networks (IJCNN), pages 1–8, 2010.

- F. Scarselli, M. Gori, A.C.Tsoi, M. Hagenbuchner, G. Monfardini. *The graph neural network model*, IEEE Transactions on Neural Networks, 20(1), pag. 61–80, 2009.

DRAFT, please do not circulate!

For information

Alessio Micheli
micheli@di.unipi.it

www.di.unipi.it/groups/ciml

Dipartimento di Informatica
Università di Pisa - Italy

Computational Intelligence & Machine Learning Group