Intro to Learning in SD -1

Alessio Micheli

E-mail: micheli@di.unipi.it

1- Introduction to RecNN

Apr 2021

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)]

An introductive aim for a research field in ML
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 in 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/networks**

- **Terminologies**:
  - Structured domain learning/ Learning in Structured Domains
  - Graph Representation Learning, Learning with/for/on graphs
  - Relational Learning
  - Structures/Graphs Mining
    - Also sub-areas, e.g. Molecule Mining
  - **Deep learning for graphs**: (deep) NN for Graphs (NN4G/DNNG), NN on Graphs, Graph NN (GNN), Graph Convolutional Networks (GCN), RecGNN*, ConvGNN*, Graph Nets, **Deep Graph Networks (DGN)***, …

* A Comprehensive Survey on Graph Neural Networks – TNNLS Jan. 2021
** A gentle introduction to deep learning for graphs – Neural Networks Sep. 2020
Increasing Attention in ML

• Dozens of new research papers in all the top-tier ML venues (literature is rapidly growing)

• And concrete real-world applications
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.

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 most of the ML course)
- **Structured**: Sequences, trees, graphs, multi-relational data

---

**stringa_in_italiano**

- Strings
- Series/temporal stream
- Proteins
- Small molecules
- Network data
Example: logo recognition

Alessio Micheli
Example: Terms in 1\textsuperscript{st} order logic

\[
A(B(\alpha, \beta), Z(\gamma, \delta), \varepsilon)
\]

\[
B(N(\gamma, \alpha, \beta), Z(\varepsilon, K(\gamma, \alpha)))
\]
It has no bearing on our work force.
Example: Social networks
Example: Biological Networks

- Node for protein
- Link for interaction or similarity
Example: Google maps – traffic estimates

- **DeepMind** – Google Maps. Sept. 2020
- **Aim**: accurate traffic predictions and estimated times of arrival (ETAs)
- **Graphs**: networks of roads
  - Node: each route segment
  - Edges: between segments that are consecutive on the same road or connected through an intersection
- **Task**: predicts the travel time for each supersegment (multiple adjacent segments of road that share significant traffic volume - subgraphs)
- **Results**: improve the accuracy of real time ETAs by up to 50% in places like Berlin, Jakarta, São Paulo, Sydney, Tokyo, and Washington D.C. by using Graph Neural Networks (indeed we will see *Convolutional NN for graphs*).
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 = $T(\text{Structure})$

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 known couples as in the molecules example

- Given a set of examples $(graph_i, target_i)$ (in the training set)
- Learn a 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
  - We are looking to the automatic extraction of the salinet topological information directly from data
  - A representation learning instances (extended to SD)!

Our “mantra”

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*
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 $T \in G$:
  - The children of $v$ are the node successors of $v$, each with a position $j=1,...,k$;
  - $k$ is the maximum out-degree over $G$, i.e. the maximum number of children for each node;
  - $l(v)$ 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 $v$. 

\[
T = \begin{array}{c}
\quad \text{root} \\
T^{(1)} & \ldots & T^{(k)}
\end{array}
\]
Structured Domains

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

We start considering sets of DPAGs (recursive structures), i.e.
- \(G\) as labeled direct positional acyclic graphs with:
  - \textit{Super-source}: a vertex \(s\) such that every vertex can be reached by a direct path starting from \(s\).
  - A total order (or the position) on the edges leaving from each vertex
  - \textit{Bounded out-degree and in-degree} (the number of edges leaving and entering from a vertex \(v\))
Data Domains \( G \)

- Include sub-classes:
  - DPAGs \( \supset \) DOAGs \( \supset \) k-ary trees \( \supset \) sequences \( \supset \) vectors.

- **DPAG**: superclass of the **DOAGs**: 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** (\( K=2 \))

- **Notations**:
  - \( \text{ch}[\nu] \) set of successors of \( \nu \)
  - \( \text{ch}_j[\nu] \) is the \( j \)-th child of the node \( \nu \)

Mostly used in this lecture
Structured Data: Examples

- Labeled Sequences, Trees, DOAGs- DPAGs, graphs

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 vertices exponential 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)

Children are far, distant relatives are close
Trees and DPGAs

Exercise after the lecture SD-2

b: shared node
Positional versus Ordered (details)

- DPAG: for each vertex $\nu$ in $\text{vert}(G)$, an injective function $S_\nu: \text{edg}(\nu) \rightarrow [1,2,...,K]$ is defined on the edges leaving from $\nu$

$$\text{in_set}(V) = \{u \mid \nu \in V \text{ and } u \rightarrow \nu\}$$

**Predecessors**

$$\text{out_set}(V) = \{u \mid \nu \in V \text{ and } \nu \rightarrow u\}$$

**Successors**
Models:
Restarting from the beginning:
Recurrent and Recursive models
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)]
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

Fractal tree: a recursive structure

We will see how RecNNs extend RNNs *matching* the recursive nature of trees (moving the model to the nature of the data)
## Summary of the Neural Computing Approach

<table>
<thead>
<tr>
<th>Domain</th>
<th>Neural Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static fixed-dim patterns</td>
<td>Feedforward</td>
</tr>
<tr>
<td>(vectors, records, ...)</td>
<td></td>
</tr>
<tr>
<td>Dynamical patterns</td>
<td>Recurrent</td>
</tr>
<tr>
<td>(temporal sequences, sequences...)</td>
<td></td>
</tr>
<tr>
<td>Structured patterns</td>
<td>Recursive</td>
</tr>
<tr>
<td>(trees, DPAGs, ...)</td>
<td></td>
</tr>
</tbody>
</table>
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:**
- Introduced in the ML course
- They will be the subject (further developed) ISPR/CNS courses (see later).
Recurrent Neural Networks (resume)

- Up to now:

Given \( x(0) = 0 \)

\[
\begin{align*}
\text{internal state} & \quad \begin{cases} x(t) = \tau(x(t-1), l(t)) \\ y(t) = g(x(t), l(t)) \end{cases} \\
\end{align*}
\]

E.g. also HMM

- Tau is the state transition (next-state) function realized by a NN (say \( \tau_{NN} \))
RNN training and properties (resume)

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

  - **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 $\tau$ is independent on node $v$ (the same in any time)
Recursive Neural Networks (overview)

- Now:

\[
\begin{align*}
    x(v) &= \tau(x(\text{ch}[v]), l(v)) \\
    y(v) &= g(x(v), l(v))
\end{align*}
\]

\[
x(\text{ch}[v]) = x(\text{ch}_1[v]), ..., x(\text{ch}_k[v])
\]

State transition system (for trees)

# feedbacks = # children
Recursive Neural Networks (overview)
More precisely (with initial conditions)

Now:

\[ \begin{align*}
  x(v) &= \tau(x(ch[v]), l(v)) \\
  y(v) &= g(x(v), l(v))
\end{align*} \]

\[ x(ch[v]) = x(ch_{l}[v]), \ldots, x(ch_{k}[v]) \]

Given \( x(nil) = 0 \)

State transition system

nil or \( x_0 \) or empty node

\# feedbacks = \# children
Generalized Shift Operators

- Standard shift operator (time):

\[ q^{-1} S_t = S_{t-1} \]

- Generalized shift operators (structure):

\[ q_j^{-1} G_v = G_{\text{ch}_j[v]} \]

where \( \text{ch}_j[v] \) is the \( j \)-th child of \( v \)

RecNN with \( q \):

\[
\begin{align*}
\mathbf{x}(v) &= \tau(\mathbf{l}(v), q^{-1} \mathbf{x}(v)) \\
\mathbf{y}(v) &= g(\mathbf{x}(v))
\end{align*}
\]

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

Used to extend the states for children of vertexes of the tree
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**
     • Properties of the recursive transduction
     • Unfolding
     • Domain details and implementation by NN
     • Applications
     • Learning algorithms

   • Recursive Cascade Correlation & other recursive approaches

2. Moving to DPAG and Graphs: the role of causality
   [Next Lecture (SD-2)]
Recursive Processing *(a different point of view)*

Recursive definition of $\tau_E$ *(encoding function)*

**Node/Graph embedding**

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

$$G = \begin{cases} 
\text{s} \\
G^{(1)} & \cdots & G^{(k)} 
\end{cases}$$

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

$G^{(j)}$ is a subgraph rooted in the children $j$ of $s$.

$s$ has a label $l(s) = L_{\text{root}}$

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

$\tau_E$ : **systematic visit** of $G \rightarrow$ it guides the application of $\tau$ (in the form of a NN $\tau_{NN}$) 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 $\nu$ only depends on $\nu$ and its descendants (induced subgraphs)
  - Compositionality over the input structure!

- **Stationarity:** state transition function $\tau_{NN}$ is independent on vertex $\nu$
  - Parsimony: we use the same NN for each vertex, regardless of the dimension of sequences (useful to process structures of variable size with a fixed size of the model)

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]
Property (II): Kind of graphs for RecNN

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

![Graphs Diagram]
Properties (III): Transduction: graphical view

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

\[ T_G \]

Input DPAG

$\tau_E$

Encoding

State space $x(G)$

internal rep./encoding

Node/Graph embedding

Output DPAG

$g$

Output

Structure-to-Structure transductions
Properties (IV): Transduction: graphical view

- $T_G$ Supersource transductions

Supersource ($s$) → Encoding → State space $x(G)$ → Internal rep./encoding → Node/Graph embedding 

Input DPAG

Structure-to-Element (regression/classification) 
Or Supersource transductions
Properties (IV)

- IO-isomorph causal transduction

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

Only the sub-structure is considered
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
       • Properties of the recursive transduction
       • Unfolding
       • Domain details and implementation by NN
       • Applications
       • Learning algorithms
     • Recursive Cascade Correlation & other recursive approaches

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

- We will see (again) the 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** isomorph 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
Alessio Micheli

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**
Unfolding & Encoding Process: Unfolding view (1)

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

- We will see later how to make it with $\tau_{NN}$ (and hence by NN) for each step: to build an encoding network
RecNN over different structures: unfolding 2

Examples on different trees for chemical compounds:
Unfolding through structures:
The same process applies 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)

\[ \tau \rightarrow \tau_{NN} \]

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

\( \tau_{NN} \) (and weight sharing) : units are the same for the all the vertices of a tree and for all the trees in the data set!!!
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**
     • Properties of the recursive transduction
     • Unfolding
     • **Domain details and implementation by NN**
     • Applications
     • Learning algorithms
   • Recursive Cascade Correlation & other recursive approaches

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

- \( X = \mathbb{IR}^m \) continuous (real values) state (code) space (encoded subgraph space)
- \( L = \mathbb{IR}^n \) vertex label space
- \( O = \mathbb{IR}^z \) or \( \{0,1\}^z \)
- \( \tau = \tau_{NN} : \mathbb{IR}^n \times \mathbb{IR}^m \times \ldots \times \mathbb{IR}^m \rightarrow \mathbb{IR}^m \)
- \( g \) output function
- \( x_0 = 0 \)

\( T_G : G \rightarrow O \)

\( \tau \) and \( g \) realized by NN with free parameters \( W \)

RNN realize \( g \circ \tau_E \)
Realization of $\tau_{NN}(\text{RecNN})$ unit

$$\tau_{NN} : IR^n \times IR^m \times \ldots \times IR^m \rightarrow IR^m$$

$\text{Dim } m$

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

where $x^{(j)} = x(\text{ch}_j[v])$

$1$ recursive neuron

$(\tau_{NN} \text{ with } m=1)$

Process a vertex

$\# \text{ feedbacks} = \# \text{ children}$

$(\text{max } k)$
Fully-connected RecNN

- Standard Neurons
- Recursive Neurons
- Inputs

Output Network

$g$ function

Encoder

$\tau_E$

$x$

Labels

$q_1^{-1}x$

$q_2^{-1}x$

Copy made according to the graph topology

E.g. for a binary tree
In details: Encoding Network with NN (I)

Start here!

Tau (and weight) sharing: units are the same for all the vertices of a tree and for all the trees in the data set!!!
In details: Encoding Network with NN (II)
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**
       • Properties of the recursive transduction
       • Unfolding
       • Domain details and implementation by NN
     • Applications
     • Learning algorithms
   • Recursive Cascade Correlation & other recursive approaches

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

- 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 (about movies)
  - 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

Polarity grade

Human-gram annotation

The actors are fantastic

Other instances in the dataset
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**
       • Properties of the recursive transduction
       • Unfolding
       • Domain details and implementation by NN
       • Applications
       • **Learning algorithms**
       • Recursive Cascade Correlation & other recursive approaches

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

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)
- RecNN with recursive units within a **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 \( \tau_E \)). 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.
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:

\[
\frac{\partial x_h(v)}{\partial w_{hi}} = \frac{\partial \tau_h(l, x^{(1)}, \ldots, x^{(k)})}{\partial w_{hi}} = f'(l_i + \sum_{t=1}^{k} \hat{w}_{hh}^t \frac{\partial x_h^{(t)}}{\partial w_{hi}})
\]

\[
\frac{\partial x_h(v)}{\partial \hat{w}_{hi}} = \frac{\partial \tau_h(l, x^{(1)}, \ldots, x^{(k)})}{\partial \hat{w}_{hi}} = f'(x^{(j)}_i + \sum_{t=1}^{k} \hat{w}_{hh}^t \frac{\partial x_h^{(t)}}{\partial \hat{w}_{hi}})
\]

where \(x^{(j)} = x(ch_j[v])\)

**Note:** simplification (of the sum on other units) due to the architecture difference with the RTRL for a fully connected RNN! But compared to the recurrent case it appears the summation on children!
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

<table>
<thead>
<tr>
<th></th>
<th>$l$</th>
<th>$C_1$</th>
<th>$C_2$</th>
</tr>
</thead>
</table>

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

- Generative process from the leaves to the root
- Markov assumption (conditional dependence) 
  \[ Q_{ch1}(u), \ldots, Q_{ch_K}(u) \rightarrow Q_u \]

**Children to parent hidden state transition**

\[
P(Q_u | Q_{ch1}(u), \ldots, Q_{ch_K}(u))
\]

Where (from a different paper)

\[ Q_v \leftrightarrow x(v) \]

Bayesian network unfolding graphical model over the input trees; y: observed elements

Q: hidden states variables with discrete values

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

![Diagram of TreeESN]

The recursive process of RecNN made by efficient RC approaches


More in a NEXT LECTURE
Hierarchical abstraction both through the input structure and architectural layers

Progressively more abstract deep reservoir representations of trees

- Improve efficiency (giving same #units) by a factor = num. of layers
- Improve results

Pro:
1. Rec NN adv.s
2. Deep learning abstraction (even before training: arch. bias)
3. Extremely efficient

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 parameters) without reducing expressive power (universal approximation theorem)

- **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?

We will see:
Or by relaxing the causality assumption
Or by imposing constraints on the recursive processing
Summary of MODELS for SD (examples)

Standard ML models for flat data

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

DPAG:
- CRCC

• Recurrent NN/ESN
• HMM
• Kernel for strings ...

See references for models in the bibliography slides (later)
Bibliography: aims

Different parts in the following:

- Basic/Fundamentals
- To go ahead

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

- **More bibliography on demand:** micheli@di.unipi.it
Bibliografia (Basic, origins of RecNN)

RecNN


Bibliography: NN approaches-2

* UNSUPERVISED RecursiveNN


* TreeESN: efficient RecNN


* HTMM: further developments (generative)

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)

Bibliography: basic items for the next lecture

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

* NN for GRAPH DATA: how to extend the domain (II)
  
  
  
  - C. Gallicchio, A. Micheli. *Graph Echo State Networks*, Proceedings of the International Joint Conference on Neural Networks (IJCNN), pages 1–8, 2010.
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