Intro to Learning in SD -2

Alessio Micheli
E-mail: micheli@di.unipi.it

2- Neural Networks for Graphs
Apr 2021

DRAFT, please do not circulate!

www.di.unipi.it/groups/ciml

Dipartimento di Informatica
Università di Pisa

Computational Intelligence &
Machine Learning Group
Learning in Structured Domain
Plan in 2 lectures

1. Recurrent and Recursive Neural Networks
   Extensions of models for supervised and unsupervised learning in structured domains
   - 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
   - Recap SD1
   - Causality for Recurrent and Recursive models &
   - Contextual approaches (CRCC and DPAGs)
   - Neural Networks for graphs
Recup SD-1: Adaptive processing of SD

- **The problem:** there has been no systematic way to extract features or metrics relations between examples for SD

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

- **Recursive** and parametric realization of the transduction function
- **Adaptive** by Neural Networks: RecNN
  - Pro: RecNN adapts the model to the hierarchical data
  - Cons: Causality issue (*): it affects the computational power of RecNN and the class of graphs! → new models!
The scenario, terms (and trends)

<table>
<thead>
<tr>
<th>Input Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flat (vectors) (e.g. F.F.NN)</td>
</tr>
</tbody>
</table>

Layering

- Shallow
- Deep

Focus

Representation of the input at multiple-levels of abstraction

“Deep and wide” approaches
Our graphs (in the following)

Labeled graphs $g$

Node/Vertex $v$  
Arc/Edge/Link (can be oriented/directed)

Node vector label $l(v) = l_d = [1,0,1,0.7]$
Arc vector label (e.g. a position)
Cycle
Graph Representations

The problem: there has been no systematic way (of general validity for any task) to extract features or metrics relations between examples for SD

- **Features based** representations are incomplete (or strongly task-dependent, e.g. topological indexes)

- **Adjacent/incident matrix** representations (or other fixed-sizes representations). Issues:
  - Over-dim./incomplete (wasteful by padding/lose inf.)
  - Alignment among different graphs
  - Topological order (make difficult the generalization)

- ML issues for the high proportion between combinatorial number of possible data examples and available data

- “The ability to treat the proper inherent nature of the input data is the key feature for a successful application of the machine learning methodologies.”
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 supervised and unsupervised learning in structured domains
   - 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
   - Recap SD1
     - Causality for Recurrent and Recursive models &
     - Contextual approaches (CRCC and DPAGs)
   - Neural Networks for graphs

By a journey through the causality assumption! *
CRCC: introduction

- Analysis of the **causality** assumption for Recurrent and Recursive neural computing models

- Partial relaxation (or **extension**) of the causality assumption

- First approach to deal with *contextual information* in SD by Recursive models
Causal Systems

- Recurrent NN models are based on the **Causality** assumption, i.e. RNN are only able to memorize past information

A system is causal if the output at time $t_0$ only depends on inputs at time $t \leq t_0$

necessary and sufficient for internal state

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

Graphical model

Shift operator $q^{-1}$
Causal Systems in Structured Domain (RecNN)

- The causality concept can be generalized to structured data transductions as follows.

A system is causal if its output for a node $v$ only depends on $v$ and its descendants.

Unfolding the encoding process through structure:

\[
\begin{align*}
x(v) &= \tau(x(ch[v]), l(v)) \\
y(v) &= g(x(v), l(v))
\end{align*}
\]
Drawbacks of Causal Systems for sequence domain

- Several prediction tasks involving **sequences** require past and “future” information (**on known sequences**)
  - DNA and Protein analysis / Language understanding / ...

![Diagram showing past and future sequence parts](image)

- Contextual information for **structured** domains: whenever the meaning of a sub-structure depends on the context in which it is found
  - some classes of transductions **cannot** be computed by causal models (also some causal transduction !!!!)
  - extension of the class of graphs
  - **Properties in flat domains cannot be trivially “exported” in SD!**
Premise: Overcome Causality for Sequences

- A possible bi-causal model: however, this is not easily implementable
  - Cycles: State equations and enc. net. (unfolding) become dynamical systems due to mutual dependencies
  - Different solutions are available (e.g. bidirectional approaches for RNN using a different state for left-to-right o right-to-left encoding). See CNS course
  - *It is not easy to extend them to structures* (many possible paths, *not just* left-to-right o right-to-left)
Overcome the Causality Assumption for SD: CRCC

The Structure Domain

- Context processing relevance
- CRCC: Cascade **Recursive** Cascade Correlation:
  - Effects over sequences and trees
  - Moving to trees and DPAGs
  - Examples of Results

```
Supersource
```
```
Rooted Tree
```
```
DPAG
```
Contextual Target Functions

Relevance of contextual processing (I)

*contextual* IO-isomorphic transductions

*(where causal models fail)*

\[
\text{Target}(t_1) \neq \text{Target}(t_2) \\
\text{out}_{\text{RecNN}}(t_1) = \text{out}_{\text{RecNN}}(t_2) \\
C(x_k(c_1)) \neq C(x_k(c_2))
\]
Example on the PCA Code Plot

Causal mapping

Unique code for each occurrence

Contextual mapping

Each fragment can be represented in different ways depending on the context (position)

Internal Encoded Subgraphs Space

More expressive sub-structure encoding
DPAG representation: a counter-example

Relevance of contextual processing (II)

- Two different DPAG necessarily mapped into the same output by RecNN (supersource causal transductions) (i.e. causal models fail)

- CRCC can distinguish $G_1/G_2$ (context for node "a" is different)
- RecNN cannot ($b$ and $c$ see the same state values)
DPAGs are not trees!

Relevance of contextual processing (III)

- Causal models allow to rewrite a DPAG as an equivalent tree

![Diagram showing the equivalence between a DPAG and a tree with shared nodes](attachment:image.png)

- CRCC distinguish them!
- *We (really) extended the domain from trees to DPAGs!*
Overcome the Causality Assumption for SD: CRCC

The Structure Domain

- Context processing relevance
- CRCC: Cascade **Recursive** Cascade Correlation:
  - Effects over sequences and trees
  - Moving to trees and DPAGs
  - Examples of Results

Supersource

Rooted Tree

DPAG
Premise: recall of the RCC Approach

By a Recursive Cascade Correlation we can realize the recurrent/recursive network by a constructive approach: The hidden units are added to the network, and frozen, during the training.
CRCC Approach

- We proposed an instance of Contextual RecCC suitable for implementation with Recursive Cascade Correlation.
- Each time a unit is frozen, the portion of its (memorized) state encodes knowledge of the whole input structure.
- The following units can access to $pa[v]$ (parents of each $v$) through the $q^{+1}$ operator without introducing cycles in the state equations and encoding networks.
The CRCC Contextual Approach

- Each time a unit is frozen, the portion of its (memorized) state encodes knowledge of “the whole” structure

\[
x_1(v) = \tau_1(x_1(ch[v]), l(v))
\]
\[
x_2(v) = \tau_2(x_2(ch[v]), x_1(ch[v]), x_1(pa[v]), l(v))
\]
\[
\vdots
\]
\[
x_m(v) = \tau_m(x_m(ch[v]), x_{m-1}(ch[v]), x_{m-1}(pa[v]), \ldots, x_1(ch[v]), x_1(pa[v]), l(v))
\]

- \(pa[v]\): set of parents of \(v\)

Alessio Micheli

Micheli et al. IEEE TNN, 2004
Overcome the Causality Assumption for SD: CRCC

The Structure Domain

- Context processing relevance
- CRCC: Cascade **Recursive** Cascade Correlation:
  - Effects over sequences and trees
  - Moving to trees and DPAGs
  - Examples of Results (theoretical, experimental)

- Diagrams:
  - Supersource
  - Rooted Tree
  - DPAG

Alessio Micheli
Example: unfolding CRCC on a sequence

We can gain information on the “future” proportionally to the number of hidden units.
Example: \( C(\bullet) \) for DPAGs

The context grows (via in_set) including all sub-DPAG met along the (inverse) path \( v \rightarrow s \) and \( \downarrow v \rightarrow s \) descendants
Overcome the Causality Assumption for SD: CRCC

The Structure Domain

- Context processing relevance
- CRCC: Cascade **Recursive** Cascade Correlation:
  - Effects over sequences and trees
  - Moving to trees and DPAGs
- Examples of Results (theoretical, experimental)
Theory

- Theoretical results have introduced to characterize the *computational power* of CRCC (class of computable functions/transductions vs causal models)

- Solving the examples before:
  - extension to *contextual* IO-isomorphic transductions,
  - e.g. $\text{Target}(v)=f$ (*whole structure*): future dependencies.
  - extension to the class of supersource (causal) transductions involving DPAGs that cannot be computed by causal models
  - while supporting all the function computable by RCC

- And also:
  - Formal compact expression of the “context window”
  - Proof of computational power of CRCC (*abstracting from neural realization*)
Example: $C(\cdot)$ for Sequences

$$C(x_k(v)) = \bigcup_{i=1}^{k-1} x_i \downarrow v_{t+k-i} \cup x_k \downarrow v_{t-1}$$

It is possible to formalize the context giving formal expression of state functional dependencies
Example here for sequences.
Universal Approximation


- RecCC can approximate every measurable functions form sequences and trees to real values (in spite of their restricted recurrent architecture) for finite sets.

- **CRCC**: Universal approximation capability extended to classes of labeled DPAGs

\[ f \text{ approximated up to any desired degree of accuracy} \]
\[ \text{(up to inputs of arbitrary small probability)} \]

\[ P(x \in DPAG : |f(x) - \text{CRCC}(x)| > \delta) < \varepsilon \]
Context in a CRCC Application

PCA of the representation of the sub-structures developed by CRCC for a chemical regression task

CRCC exploits the opportunity to develop different embeddings according to the context.
CRCC Conclusions

- Show advantages of including “context” (including parents)
  - Extension of the computation capability
  - Extension of the classes of data to DPAGs
  - Expressive encoding of substructures
  - Performance where causality assumption is unknown

- However, CRCC still requires topological order and supersource, still recursive dynamics: DPAGs/DAGs

- ... New approaches: by retaining and extending context, removing causality/recursion?
  - Yes, Move to graphs!
Learning in Structured Domain
Plan in 2 lectures

1. Recurrent and Recursive Neural Networks
   Extensions of models for supervised and unsupervised learning in structured domains
   - 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
   - Recap SD1
   - Causality for Recurrent and Recursive models &
   - Contextual approaches (CRCC and DPAGs)
   - Neural Networks for graphs

By a journey through the causality assumption!
Graphs by NN: **Cycles**

Causality assumption in RecNN introduce issues in processing *cycles* (due to the mutual dependencies among state values)

Cyclic graph

How to deal with cycles and causality?
Main approaches for graphs by NN

Different classes of approaches:

1. **Rewriting the graph:**
   - Atomic representation of cycles: e.g. functional groups in chemistry
   - To trees/DAGs (e.g. SMILES representation in chemistry)

2. **RecNN** by explicitly treating the cyclic dynamics by contractive constraints (GNN, GraphESN) [1,2]

3. **Layering:** contextual non-recursive approaches (NN4G [3] /Conv. NN for graphs [4]) → **Deep NN for graphs** (or Conv/spatial approaches)

2. GNN/GraphESN (2009-2010)

- In GraphESN and GNN the equation are similar to RecNN
- Cycles are allowed (in state computation), the state/embedding for each \( v \) is computed iterating the state transition function until convergence
- Stability of the recursive encoding process is guaranteed by resorting to contractive state dynamics (Banach theorem for fixed point)
  - In GNN imposing constrains in the loss function (alternating learning and convergence)
  - In GraphESN the condition is inherited by contractivity of the reservoir dynamics (see ESP conditions): very efficient!

GraphESN state transitions
\[
x(v) = \tanh(W_{in}u(v) + \sum_{v' \in N(v)} \hat{W}x(v'))
\]

Context evolution, with the iteration, of the state for the vertex in the center (not just local, by diffusion on graph)

- Gallicchio, Micheli. IJCNN, 2010.

Also extended to GRU (Li et al. 2015)
2. GraphESN details (I)

Iterative \((t)\) computation of the local encoding

\[
x_t(v) = \tau(u(v), x_{t-1}(\mathcal{N}(v)))
\]

**Reservoir** \((\tau)\)

\[
x_t(v) = \tanh(W_{in} u(v) + \hat{W}_N x_{t-1}(\mathcal{N}(v)))
\]

A state value is computed for every vertex of each graph.

State transition eq. (reservoir units): convergence to a fixed point.
After a neural message passing operation, a vertex acquires information from its neighborhood. Here 1 embedding step is shown for the central vertex, and then it can be iterated.

Note: this is useful for both 2. Rec. and 3. Layering approaches
2. GNN/GraphESN

Pro/Cons:

+ Extend the domain of RecNN to general graphs

+ Theoretical approximation capability and VC dimension have been proved

- [GNN] elongate training time with the convergence (double mutual iteration)

- Constraints of the weight values $\rightarrow$ bias to contractive transduction

+ GraphESN does not require training time of the recursive part $\rightarrow$ efficient!

+ A deep (multi recurrent layers) version has been developed, called FDGNN (see the references)
**Toward 3: Layering/ Convolutional approaches: the problem**

CNN: Convolutional kernel can be applied on a 2D regular grid (image matrix). The CNN takes a weighted average of the pixel values in the neighbor window. The neighbor of a vertex are ordered and have a fixed size.

CNN cannot be immediately applied: For graphs the neighbors of a vertex are unordered and variable in size (*not Euclidian geometry*)

How to deal with it?
How to extend from local neighbor?
3. Layering
Contextual Multi-Layered approaches for graphs

Layering basic idea:

• the mutual dependencies are managed (architecturally) through different layers (i.e. by a deep architecture)
  – Instead of iterating at the same layer, each vertex can take the context of the other vertices computed in the previous layers, accessing progressively to the entire graph/network
  – And each vertex takes information from all the others, including the mutual influences: Collective inferencing

• NN4G since 2005-2009: a pioneer approach following the RecNN/ CRCC line (completely relaxing the recursive causality assumption)
  – In the following

• CNN for graphs since 2015: using the same concepts which moves the idea for 2D processing (images) to graph processing through many layers
NN4G: Motivations
Restarting from Causality

- Is it possible to find more general and simpler solutions removing *causality* without introducing cycles dependencies in the states definition?

NN4G: Neural Network for Graphs

- Two main ingredients:
  1. constructive (feedforward) neural network approach
  2. Local and contextual information of each vertex of a graph

  But *recursive causality* is removed

- Micheli, Sestito. WIRN 2005
1) **Constructive Approach**

Cascade Correlation (*RecCC* in the picture):
The hidden units are progressively added to the network during training, and **frozen** after insertion.

![Diagram](image.png)
2) Local Context and Structured Domain

- We assume a fairly general class of labeled graphs \( g \in G \)
- \( \text{Vert}(g) \): set of vertices of \( g \); \( l(v) \): label of \( v \)
- \( \text{edg}(v) \): set of edges incident on \( v \)
- **Neighbors of** \( v \):

\[
N(v) = \{u \in \text{Vert}(g) \mid (u, v) \lor (v, u) \in \text{edg}(v)\}
\]

**Directed**

\[
N(v) = \{u \in \text{Vert}(g) \mid (u, v) \in \text{edg}(v)\}
\]

**Undirected**

- Context of \( v \) is the set of vertices with a path to/from \( v \) affecting the output of \( v \).
NN4G: 2) Hidden Units and Context

- NN4G compute a state variable for each vertex

\[
x(v) = \begin{cases} 
  x_1(v) = f \left( \sum_{j=0}^{L'} \overline{w}_{ij} l_j(v) \right) \\
  x_i(v) = f \left( \sum_{j=0}^{L'} \overline{w}_{ij} l_j(v) + \sum_{j=1}^{i-1} \hat{w}_{ij} \sum_{u \in N(v)} x_j(u) \right) 
\end{cases}
\]

- **Current Label**

- **Context**

  It was the sum over the children 1..k for RecNN

  Nowadays, called aggregate and combine operator

- **Frozen States**

  \( j < i \)

- **New state** \( i \) for \( v \)

- **Note: Not Recursive** (no feedbacks): \( x_i(v) \) depends only on frozen values \( (j < i) \)
  - No cyclic dependencies are introduced in the definition of the state transition system
  - No topological order to follow: \( x_i(v) \) can be computed in parallel for vertices of \( g \)
NN4G: Hidden Units Generalization (edges)

- NN4G define a very general computational framework, e.g.

\[
x(v) = \begin{cases} 
x_1(v) = f \left( \sum_{j=0}^{L'} \overline{w}_{ij} l_j(v) \right) \\
x_i(v) = f \left( \sum_{j=0}^{L'} \overline{w}_{ij} l_j(v) + \sum_{j=1}^{i-1} \sum_{u \in \mathcal{N}(v)} \hat{w}_{ij}^{(v,u)} x_j(u) \right) & i = 2,\ldots, N 
\end{cases}
\]

- \((v,u)\) is unordered (for undirected graphs)
- **Stationarity** (weight sharing) strategy: association between weights and edges
  - Entering/leaving edges for directed graphs
  - Position for positional/ordered graphs
  - **Label of the arc** more in general: \(W^{(u,v)} = W^{(t,v)}\) if \(L(u,v)=L(t,v)\) (w is the same for edges sharing the same label, i.e. a different \(w\) for each different edge type)

- First trials: full stationarity: 1 weights for each edges:
  - unordered and undirected graphs and
  - strong parameters reduction
NN4G: 2) Hidden Units and Context

- Is NN4G just a relational approach taking only a local neighborhood (for each hidden unit)?

  - **No**, because through layering NN4G extends the context of each vertex to all the vertices in graph
  - Because progressively, **by composition**, the model extends the context of influence to other vertices through the context developed in the previous (frozen) hidden units (layers) \( \rightarrow \) **see the next slide**
Evolution of the Context (Compositional)

Context of radius 2 for $x_3(v)$

Hidden Unit 1

Hidden Unit 2

Hidden Unit 3

State variables

$x_1(v)$

$x_2(g)$

$x_3(g)$

Hidden Unit 2

Hidden Unit 3

$w_1$

$w_2$

$w_3$

$y$
1. From states to the output layer
   - IO- isomorphic transduction (an output for each vertex) or
   - A scalar value for a whole graph can be emitted, using an operator $X$, e.g.:

   $y(g) = f\left( \sum_{j=0}^{N} w_j X_j(g) \right)$

   States mapping function $x(g)$

   It can be a simple permutation invariant function e.g. a simple global sum or average or a selection from relevant vertices etc.

2. Output layer: e.g. A single standard neural unit

   • Learning: as in (feedforward) Cascade Correlation: adding hidden units and interleaving min. of error at the output layer and max. of the correlation score for each hidden unit.
Summary: NN4G Algorithm
(compute the output for a set of graphs)

1. For \( i = 1 \) to \( N \)
2. For all \( g \) in \( G \)
3. For all \( v \) in \( \text{Vert}(g) \)
4. Compute \( x_i(v) \) (even in parallel *)
5. Compute \( X_i(g) \)
6. For all \( g \) in \( G \)
7. Compute \( y(g) \)

(*) I.e. a traversal of the input graph: the result does not depend on the visiting order
Weight sharing: the same model is applied for each vertex
Evolution of the Context (Compositional)

Context of radius 2 for $x_3(v)$

Hidden Unit 1

Hidden Unit 2

Hidden Unit 3

State variables $x_1(v)$ for $x_3(v)$

$X_1(g)$

$X_2(g)$

$X_3(g)$

$y$

$w_1$

$w_2$

$w_3$
**NN4G: Context Growth**

- The growth of the context is symmetric in each direction starting from each vertex, and grow with layers.
- In such a way, the size of the context window can grow and we do not need to fix it prior to learning.
- *The depth of networks is functional to context development*
Context Scope:

Formal Properties relating $h$ and $C$

- It has been **formally proved** that the context $C(x_h(v))$ grows one step ahead, for each added unit (layer $h$), as $N^h(v)$:
  - the dimension of the context is proportional to the number of units,
  - and the structure of the composition is given by the topology of the input graph

- And that $C(x_h(v))$ can involve all the vertices of the graph:

**Theorem [NN4G]:** Given a finite size graph $G$, there exists a finite number $h$ of state variables (hidden layers) such that for each $v$ in $G$ the context of $v$ involves all the vertices of $G$.
  - In particular, $h > \text{ “diameter” }$ of the graph satisfy the proposition.

Example of experimental assessment: Cyclic versus Acyclic Undirected Graphs

- Artificial task: test NN4G capability to learn a relevant topological feature, i.e. the occurrence of cycles in the input undirected graphs, which cannot be directly treated by RNN.

- **Input domain:** 150 cyclic graphs and 150 acyclic graphs with 3 up to 10 vertices (total 2670 vertices)

- **100% test classification accuracy** over all the folds of 10-fold cross-validation with 5 trials for each fold.

- **Just 2 hidden units.**

- In fact, the second unit is able to distinguish the ratio between the number of edges and vertices in the graph, which is a sufficient feature to discriminate the input graphs on the basis of the occurrence of cycles in its topology.
**NN4G Recap**

- **NN4G:** *A deep model for graphs*

- **Characteristics:**
  - Direct/undirected cyclic/acyclic labeled graphs
    - W.r.t. RecNN does not assume *causality* over directed structure; in particular, no assumption on the topological order is needed;

  - Incremental, layer by layer learning & automatic model design
  - **Depth** functional to contextual encoding: Dimension of context grows with layers (*formally proved*)

  - **Efficient:** no cyclic def. of state var., divide et impera on the task
    - Scaling: Current model (full stationarity): $O(|G|Vh^2\text{ epochs})$: Linear in the number $V$ of vertices

  - Generality: No constraints on weights values (vs GNN)
    - Pool strategy (in cascade corr. training): local minima avoidance, supervised architecture optimization
A first comparison
NN4G / Conv.NN for Graphs

Concepts in common:

- **Traversal of the input graph:** Visiting (the nodes of) input graphs through units with weight sharing (stationarity)
  - This corresponds for CNN to the *convolution* over (the nodes of) input graphs,
  - i.e. constrained to graph topology instead of 2D matrix

- **Layering** and hence *moving to deep architecture* (functional to *contextual processing*)

- **Composition** for the (no causal) context learning, parsimony, and adaptivity are achieved and extend to *any kind of graphs*

- Node-centric learning can exploit the *Collective inference*
A first comparison
NN4G / Conv.NN for Graphs

Main differences are more related to the training:

- **CNN-Gs** typically use CNN architecture/training approaches,
  - Fixed architecture (few hidden layers)
  - Top-down back-prop (end-to-end): can be quite computational demanding using many layers

- **NN4G**: Incremental, layer by layer learning & automatic model design
  - Advantages: No gradient vanish issue, *divide et impera*, automatic number of layers, etc.
Many models and variations ...

Differences among Convolutional models e.g. for:

- **Neighbors aggregation** (or other slight variations)
  - E.g. NN4G, GCN, DCNN, PATCHY-SAN, GIN, ECC, GAT, ...
    - also with sampling, e.g. FastGCN, GraphSAGE, ...

- **Different filters for spectral approaches**
  - E.g. SpectralCNN, ChebNet, ARMA-GCN, ...

- **Pooling among layers**, including decimation of graph nodes and pyramidal schemas
  - E.g. Graclus, DiffPool, NDP, ...

- **Hybrid models**
Learning in Structured Domain
Plan in 2 lectures

1. Recurrent and Recursive Neural Networks
   Extensions of models for supervised and unsupervised learning in structured domains
   - 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
   - Recap SD1
   - Causality for Recurrent and Recursive models &
   - Contextual approaches (CRCC and DPAGs)
   - Neural Networks for graphs
   - Other models and looking ahead
Terminology again (recap)

See lect. SD1: Deep learning for graphs: (deep) NN for/on graphs (NN4G/DNNG), graph NN (GNN), graph convolutional networks (GCN), Graph nets, ...

In particular:

- **RecGNN** → the recursive approach (e.g. originally GNN and GraphESN)
- **ConvGNN** → for spectral-based approaches and the spatial-based approaches (e.g. NN4G and all the convolutional approaches)
- **DGN** → An extend class to include also Bayesian and Generative approaches

* A Comprehensive Survey on Graph Neural Networks – TNNLS Jan. 2021
** A gentle introduction to deep learning for graphs – Neural Networks Sep. 2020
DBGN: by NN4G+ HTMM

- We can extend the contextual ideas also to RC and HTMM approaches making them **deep** and **for graphs**

  - *E.g.* ICML 2018 A NN4G realized by a *generative* approach (a DBGN)
  - trained by a mix of unsupervised (Markov models for hidden layer) and supervised (output layer) approaches
  - Also for unsupervised / semi-supervised probabilistic learning

- **Contextual Graph Markov Models**

Bacciu, Errica, Micheli, ICML 2018
Bacciu, Errica, Micheli, JMLR 2020
Generative Graph Networks

- **Generative**: NN for graphs can learn how to generate new data preserving the statistical properties of the training set → generate new graphs

Podda, Bacciu, Micheli. A Deep Generative Model for Fragment-Based Molecule Generation - AISTATS (2020)
Further discussion: properties

- **Models for graphs** support different graph structures as well as a flexible representations of global, node, and edge attributes, customizable according to specific demands of tasks.

- **Non-Euclidean geometry**: they treat directly the structured nature of graphs.

- **Modularity and compositionality**: they can learn independent mechanisms that can be reused in several parts of the graph.

- **Cross-modality**: they can learn how to combine structured unstructured data sources. This is particularly relevant when integrating data/signals at different levels of complexity.

- **Multiscale**: the graph representation has the capability of integrating granular information organized as networks at different layers of complexity.
  - E.g. patterns in higher-order structures such as motifs, pathways, tissues (as compositions of cells), organs (as composition of tissues), processes and apparatus (as composition of organs), stratification (as composition of individuals).
And Kernels?

- Large set of proposals (out of the scope)
- Examples of kernels for SD from RC [1] and HTMM [2] for SD, e.g.
  (by CIML):

- Kernels for SD from RC

- Kernels for SD from HTMM (adaptive kernels + generative & discriminative)
  2. D. Bacciu, A. Micheli, A. Sperduti. *Generative Kernels for Tree-Structured Data*, IEEE TNNLS (Transactions on Neural Networks and Learning Systems), 2018
Summarizing the MODELS panorama for SD (examples)

Standard ML models for flat data

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

DPAG:
- CRCC

See references for models in the bibliography slides (later)

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

- GNN/GraphESN
- NN4G, CGN
- DGN
- Graph Kernels
- SRL
- …
Software

- PyTorch geometric
- DGL (agnostic: PyTorch, TensorFlow or MXNet)
- Spektral (TF)
- Jraph (in jax)
- ...
- In CIML (Software): PyDGN
Data Sets: Large Collections

- OGB (Stanford)
- TUDataset (TU-Dortmund - ICML 2020)
- inside PyTorch geometric and the other tools
Future – just ahead

• New models for SD (discussed so far)

• New applications

• ...

A. Micheli
Applications

• See the first lecture SD1 (initial slides) for examples for molecules, social, biological networks, maps ....

• And also recommendation systems (Amazon!), image processing, games, medical records, ...

• But all the data have relationships !!!

• Let me introduce some running projects in CIML

Computational Intelligence & Machine Learning Group
SDL for parallel computing

- Mapping and configuration problem in *parallel computing* (on *tree* data representations): estimation of execution time and energy [with M. Danelutto, UNIPI]

Skeleton application description

```
Farm
```

```
Pipeline
```

```
Seq.
```

Execution time (sec) or Energy consumption

NN for Trees

On going: Thesis available!

Service time, parallel degree, ….
Cybersecurity and Trustworthy

Under construction (with ISTI CNR):

• Starting for example from malware detection/classification from a «call graph» or «control flow graph» of the code

• Also ... toward trustworthy learning for graphs (a new field!)
Bioinformatics

Under construction! Ready to apply for tree/graph data on:

- **System biology** (graphs/networks) (with P. Milazzo) → next slides

- **Bioinformatics:**
  - Prediction of proteins function → next slides
  - Pan-genome analysis (by graph representation)
  - Coarse grained models for proteins → next slides
  - Bio-molecular Networks → next slides
  - Coarse grained models for Proteins → next slides

*On going: Thesis available!*
Biochemical Pathways (BPs) are sets of chemical reactions among biomolecules.

They model **cell functionality**.

Structure described by graphs (e.g. Petri Nets), dynamical properties analyzed with Ordinary Differential Equations (ODEs).

Entities (nodes): species (circle) or reactions (square).

Relations (arcs): produce, enhance, block.

- There are also Input/Output nodes: induced sub-graphs of paths for different I/O nodes.

UNIPI funded research project - PRA 2020
METODI INFORMATICI INTEGRATI PER LA BIOMEDICA
Classification of Biochemical Pathway Robustness with Neural Networks for Graphs

• **Input:** Pathway Petri nets (their correspondent *directed* graph), for each pair of species in a given net
• **Output:** an associated *robustness* value (0/1)

• Direct prediction for the Petri nets topology, avoiding costly ODE simulations
• Does the networks include enough information to predict robustness? Or other properties?

*On going: Thesis available!*

• Best paper award @ **BIOSTEC-BIOINFORMATICS 2020**
Prediction of proteins function

- Prediction of proteins function
- Using gene ontology graphs
- Possibility to participate to CAFA challenge
  - There are many proteins in the databases for which the sequence is known, but the function is not.
  - The Critical Assessment of Functional Annotation (CAFA) is an ongoing, global, community-driven effort to evaluate and improve the computational annotation of protein function.

Approaches can include:

- SVM/Kernel building by efficient computation of common sub-graphs between pairs of proteins (with R. Grossi)
- Use of NN for graphs
Bio-molecular Networks

A Bio-molecular network is a graph representation of:

- **Relationships** (of which there are many types) amongst a group of biomolecules.
- **Vertices** or nodes represent biomolecules, including macromolecules such as genes, **proteins**, and RNAs, or small biomolecules like amino acids, sugars, and nucleic acids.

In particular, other networks data/problems to extend the application of ML/NN for graphs are:

- Proteins (interactions or similarities or ontologies) networks for drug or oncology research (with other institutes)
  - These graphs contain a relevant information that can be exploited for conducting different kinds of analysis, such as automatic function prediction, disease gene prioritization, drug repositioning or discovery, etc.

- Knowledge graphs for COVID related analysis (with Univ. of Padova, under construction)
Coarse grained models for Proteins (Biophysics with Univ. of Trento)

By NN for graphs!

On-going: Thesis available!
A more general aim (CIML -Pisa)

• Adaptive processing of SD

• A theoretical and practical framework for the automatic design of efficient models for sequences, trees and graphs (both generative and discriminative) exploiting DL approaches
  – Able to answer the main issue of DL frameworks: how many layers? How many units? Which hyper.? Etc.
  – Open to semi-supervised learning and different graph and network tasks
  – Efficient by incremental NN and RC approaches

Computational Intelligence & Machine Learning Group
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 (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: RecNN 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: This lecture (I)**

**Main references**

- **Bidirectional RNN**

- *RecNN for DPAGs: how to extend the domain (I)*
  - A. Micheli, D. Sona, A. Sperduti.
  - Hammer, A. Micheli, and A. Sperduti.
**Main references**

* 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.


* C. Gallicchio, A. Micheli *Fast and Deep Graph Neural Network*, accepted for AAAI 2020. Pre-print at: https://arxiv.org/abs/1911.08941
Bibliography: This lecture (III)

Other References

Convolutional Neural Networks for Graphs

- Kearnes et al., Molecular Graph Convolutions: Moving Beyond Fingerprints, J Comput Aided Mol Des, 2016, arxiv.org/abs/1603.00856v3
- ..., ..., ... continuosly coming!
Bibliography: This lecture (IV)

Other References

Probalistic models for Graphs


• ..., ..., ... continuosly coming!
Bibliography: This lecture (V)

Other References

NN for graph data: recent surveys


by the CIML group (Pisa)

- D. Bacciu, A. Micheli. *Deep learning for graphs*. Proceedings of INNSBDDL 2019 - book series Studies in Computational Intelligence (Springer), in press. Available upon request ([micheli@di.unipi.it](mailto:micheli@di.unipi.it)).

- ... , ... , ... continuously coming!
DRAFT, please do not circulate!

For information

Alessio Micheli
micheli@di.unipi.it

www.di.unipi.it/groups/ciml

Computational Intelligence & Machine Learning Group