Reservoir Computing for Learning in Structured Domains

Machine Learning: Neural Networks and Advanced Models (AA2)

Claudio Gallicchio

gallicch@di.unipi.it

Department of Computer Science
University of Pisa

April 2015

Computational Intelligence and Machine Learning Group

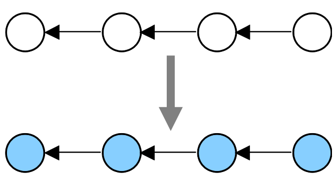
Overview

- Learning in Structured Domains (trees, graphs)
- Recurrent/Recursive Neural Networks
- Reservoir Computing
- Contractivity, Markovianity
- Reservoir computing for Structures, TreeESN, GraphESN
- Applications

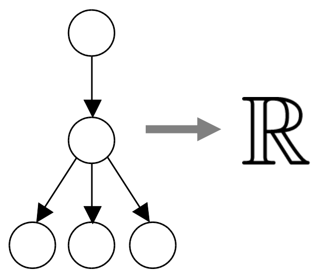
Learning in Structured Domains

- In many real-world application domains the information of interest can be naturally represented by the means of **structured data** representations.
- The problems of interest can be modeled as regression or classification **tasks on structured domains**.

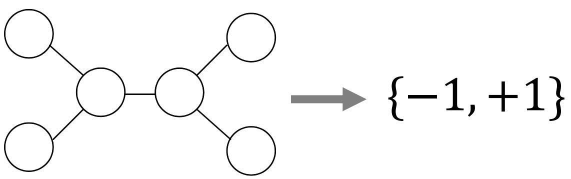
Sequences



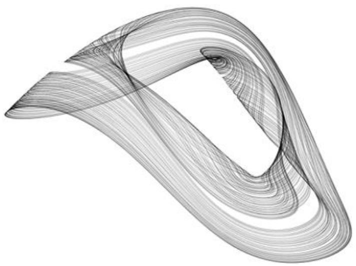
Trees



Graphs

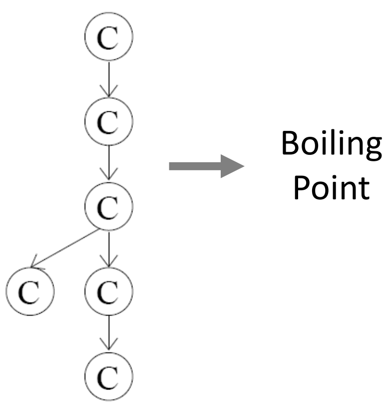


MG -Chaotic Time Series Prediction

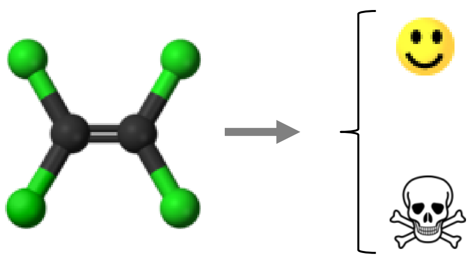


$$\frac{\partial u(t)}{\partial t} = \frac{0.2u(t-\tau)}{1+u(t-\tau)^{10}} - 0.1u(t)\alpha$$

QSPR analysis of Alkanes



Predictive Toxicology Challenge



Learning in Structured Domains

Learning in domains of trees and graphs opens up a wide range of research directions:

- Theoretical
- Experimental Analysis in interdisciplinary areas
  - QSAR/QSPR
  - Computational Toxicology, Cheminformatics
  - Social and Web information Processing
  - Document processing
  - Parallel Computation
  - …

Problems

Learning in Structured Domains entails a number of open research problems, mainly related to the increasing complexity of the data domains to treat

- **Efficiency**
- Generalization of **class of data structures** supported
- **Adaptivity**
- **Generalization** ability

Neural Networks for Structured Domains

Models

- Neural Networks for structured domains: Recurrent Neural Networks (RNNs), Recursive Neural Networks (RecNNs), Neural Networks for Graphs (NN4Gs), Graph Neural Networks (GNNs)
- **Reservoir Computing** – extension of RC to structured domain processing
  Tree Echo State Networks (TreeESNs) , Graph Echo State Networks (GraphESNs)

- Kernel Methods for structures

General Framework for Processing Structured Domains

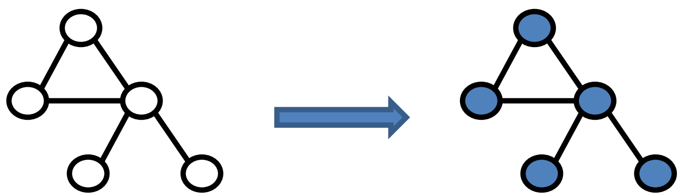
Transductions on Structured Domains

$$\mathcal{T} : \mathcal{U}^{\#} \rightarrow \mathcal{Y}^{\#}$$

label input space — label output space

$$\mathcal{U} = \mathbb{R}^{N_U}$$
$$\mathcal{Y} = \mathbb{R}^{N_Y}$$

Structure-to-structure Transductions

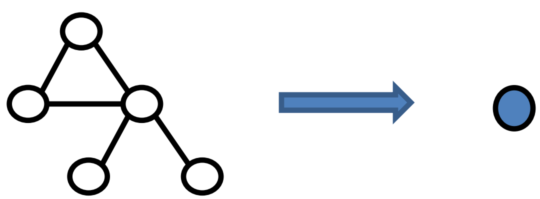


$\mathbf{g}$ $\quad$ $\mathbf{y}(\mathbf{g})$

$\mathbf{y}(\mathbf{g})$ is isomorphic to $\mathbf{g}$

$$skel(\mathbf{y}(\mathbf{g})) = skel(\mathbf{g})$$

Structure-to-element Transductions



$\mathbf{g}$ $\quad$ $\mathbf{y}(\mathbf{g})$

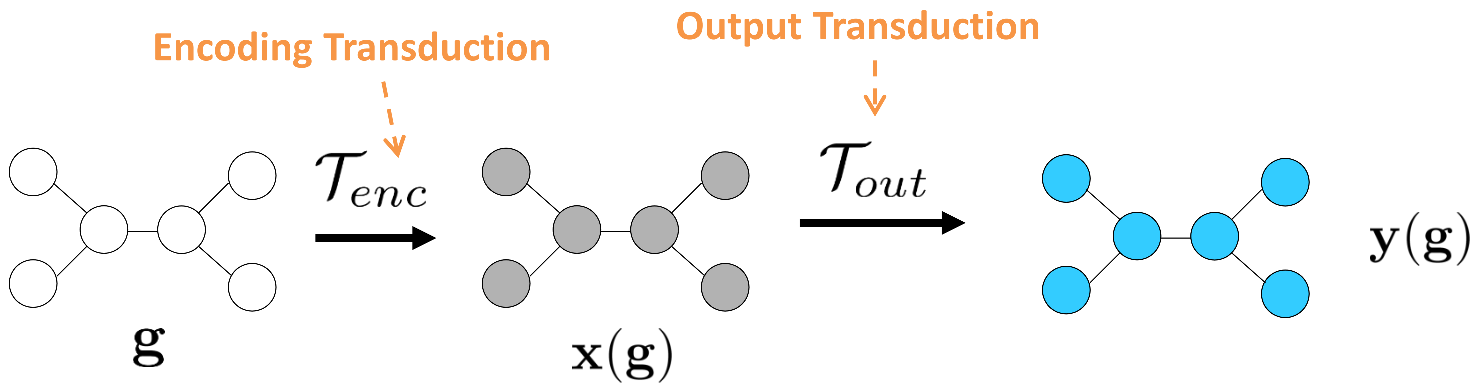
$\mathbf{y}(\mathbf{g})$ is a vector

General Framework for Processing Structured Domains

Computing Structural Transductions

Structure-to-structure Transductions

$$\mathcal{T} = \mathcal{T}_{out} \circ \mathcal{T}_{enc}$$



$$\mathcal{T}_{enc} : (\mathbb{R}^{N_U})^{\#} \rightarrow (\mathbb{R}^{N_R})^{\#}$$

$$\tau : \mathbb{R}^{N_U} \times \mathbb{R}^{k\, N_R} \rightarrow \mathbb{R}^{N_R}$$

$$\hat{\tau} : (\mathbb{R}^{N_U})^{\#} \times \mathbb{R}^{N_R} \rightarrow (\mathbb{R}^{N_R})^{\#}$$

$$\mathcal{T}_{out} : (\mathbb{R}^{N_R})^{\#} \rightarrow (\mathbb{R}^{N_Y})^{\#}$$

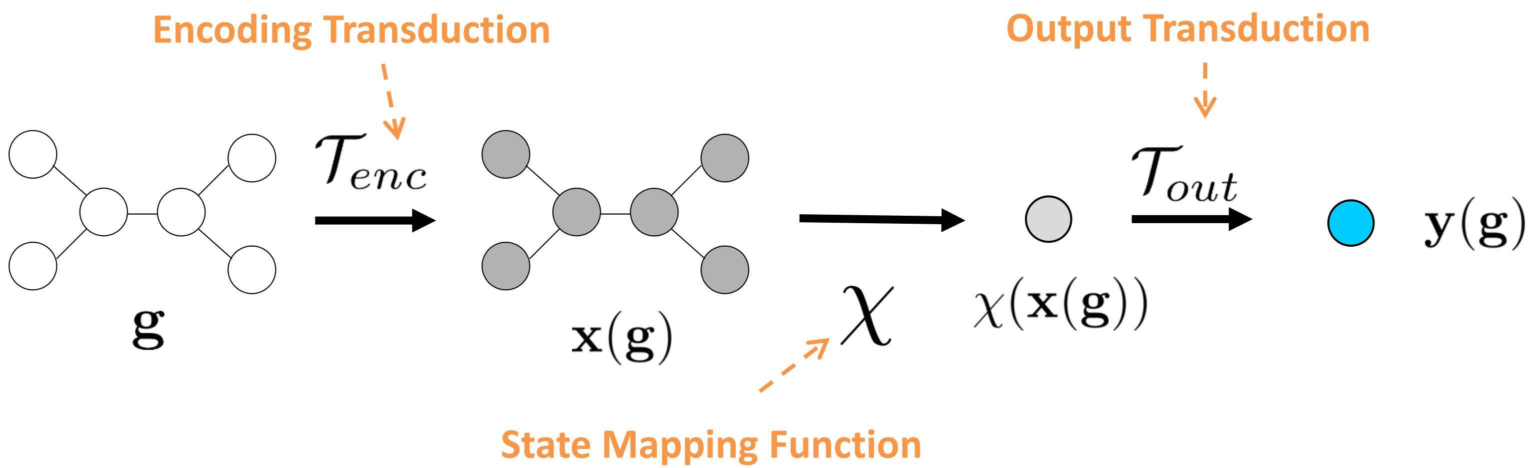
$$g_{out} : \mathbb{R}^{N_R} \rightarrow \mathbb{R}^{N_Y}$$

General Framework for Processing Structured Domains

Computing Structural Transductions

Structure-to-element Transductions

$$\mathcal{T} = \mathcal{T}_{out} \circ \chi \circ \mathcal{T}_{enc}$$



$$\mathcal{T}_{enc} : (\mathbb{R}^{N_U})^{\#} \to (\mathbb{R}^{N_R})^{\#}$$

$$\tau : \mathbb{R}^{N_U} \times \mathbb{R}^{k\ N_R} \to \mathbb{R}^{N_R}$$

$$\hat{\tau} : (\mathbb{R}^{N_U})^{\#} \times \mathbb{R}^{N_R} \to (\mathbb{R}^{N_R})^{\#}$$

$$\chi : (\mathbb{R}^{N_R})^{\#} \to \mathbb{R}^{N_R}$$

$$\mathcal{T}_{out} : (\mathbb{R}^{N_R})^{\#} \to (\mathbb{R}^{N_Y})^{\#}$$

$$g_{out} : \mathbb{R}^{N_R} \to \mathbb{R}^{N_Y}$$

Characterizations of Structural Transductions

- **Causality**
  the function computed in correspondence of a vertex *v* depends only on *v* and its descendants

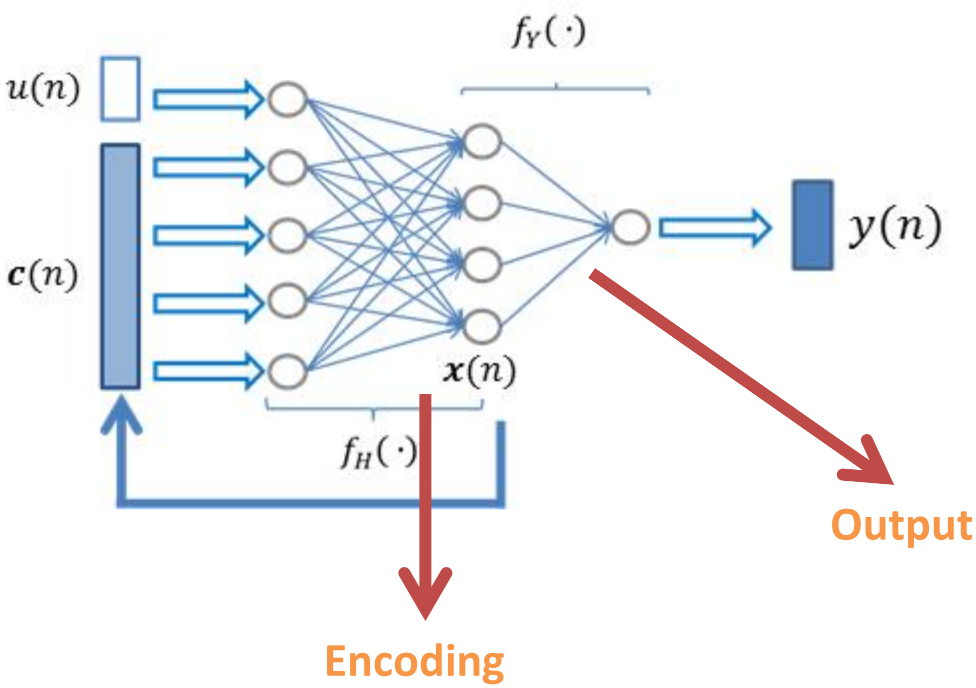
- **Stationarity**
  the function computed in correspondence of a vertex *v* does not depend on the particular vertex *v*

- **Adaptivity**
  the function is learnt from observed data

Recurrent Neural Networks

- Neural networks for learning sequence transductions
- Local encoding function τ and output function g_{out} implemented by layers of units.

Elman Network (Simple Recurrent Network)

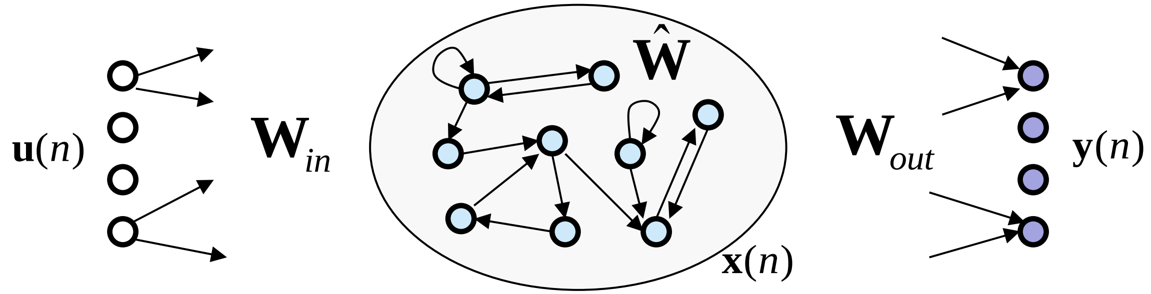


- Pro: theoretically very powerful; Universal approximation through training
- **Con**: drawbacks related to training

Reservoir Computing

- Paradigm for efficient RNN modeling – state of the art for efficient learning in sequential domains
- Implements dynamical system
- Conceptual separation: dynamical/recurrent non-linear part, reservoir
  feed-forward output tool, readout
- **Efficiency**:
  - training is restricted to the linear readout
  - exploits Markovian characterization resulting from (untrained) contractive dynamics
- Includes several classes: Echo State Networks (ESNs), Liquid State Machines, Backpropagation Decorrelation, Evolino, ...

Echo State Networks - Architecture



Input Layer N_U | Reservoir N_R | Readout N_Y

Input Space: $\mathbb{R}^{N_U}$ Reservoir State Space: $\mathbb{R}^{N_U}$ Output Space: $\mathbb{R}^{N_U}$

- Reservoir: **untrained** large, sparsely and randomly connected, non-linear layer

$\tau : \mathbb{R}^{N_U} \times \mathbb{R}^{N_R} \rightarrow \mathbb{R}^{N_R}$

$$\mathbf{x}(n) = tanh(\mathbf{W}_{in}(\mathbf{u}(n)) + \hat{\mathbf{W}}\mathbf{x}(n-1))$$

encoding of the input sequence

- linear units
- leaky-integrators
- spiking neurons

- Readout: **trained** linear layer

$g_{out} : \mathbb{R}^{N_R} \rightarrow \mathbb{R}^{N_Y}$

$$\mathbf{y}(n) = \mathbf{W}_{out}\mathbf{x}(n)$$

Train only the connections to the readout

Echo State Networks - Properties

Echo State Property

- A valid ESN satisfies the Echo State Property (ESP)
- The state of the network aymptotically depends on the input history only
- The influence of initial conditions gradually fades out

$$\forall\ \mathbf{s}(\mathbf{u}) = [\mathbf{u}(1), \ldots, \mathbf{u}(n)] \in (\mathbb{R}^{N_U})^n$$
$$\forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^{N_R} :$$
$$\|\hat{\tau}(\mathbf{s}(\mathbf{u}), \mathbf{x}) - \hat{\tau}(\mathbf{s}(\mathbf{u}), \mathbf{x}')\| \to 0$$

Initialization Conditions

Sufficient condition $\|\hat{\mathbf{W}}\|_2 < 1$

Necessary condition $\rho(\hat{\mathbf{W}}) < 1$ (asymptotical stability around **0**)

Training

Solve the least squares linear regression problem: $\min \|\mathbf{W}_{out}\mathbf{X} - \mathbf{Y}_{target}\|_2^2$

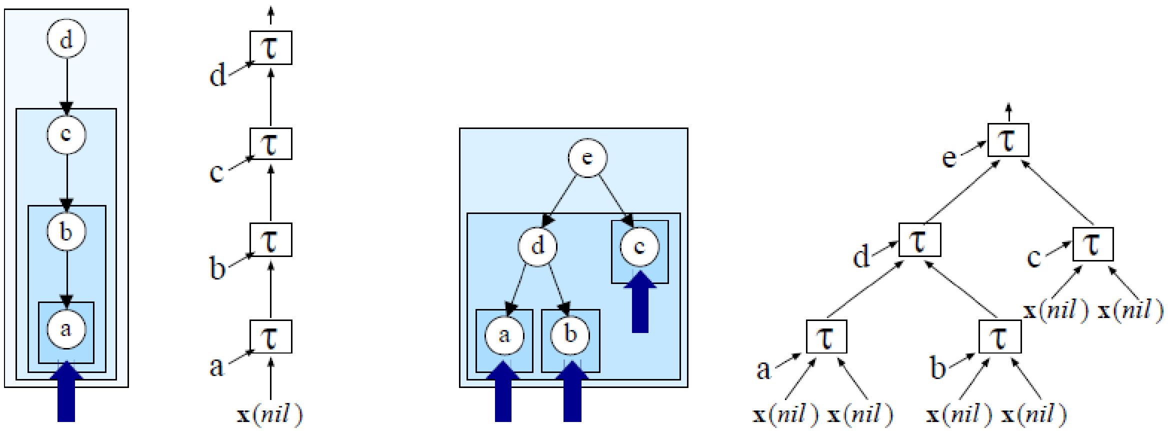
- Moore-Penrose pseudo-inversion $\mathbf{W}_{out} = \mathbf{Y}_{target}\mathbf{X}^+$
- Ridge regression $\mathbf{W}_{out} = \mathbf{Y}_{target}\mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \lambda_r\mathbf{I})^{-1}$

ESN Hyper-parametrization

Reservoir dimension, input scaling, spectral radius, readout regularization, ...

Recursive Neural Networks (RecNNs) for Structured Data

- Generalization of RNNs for processing hierarchical structures
- Bottom-up recursive encoding



$$\mathbf{x}(n) = \tau(\mathbf{u}(n), \mathbf{x}(ch_1(n)), \ldots, \mathbf{x}(ch_k(n)))$$
$$= f(\mathbf{W}_{in}\mathbf{u}(n) + \sum_{i=1}^{k} \hat{\mathbf{W}}_i \mathbf{x}(ch_i(n)))$$

$$\mathbf{y}(n) = f_{out}(\mathbf{W}_{out}\mathbf{x}(n))$$

Recursive Neural Networks (RecNNs) for Structured Data

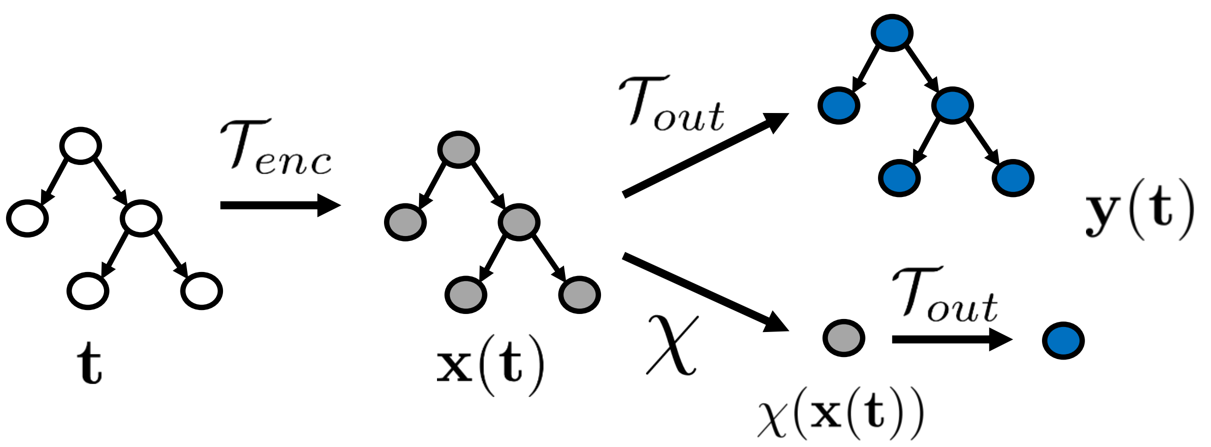
- Powerful class of learning models, universal approximation for tree domains processing (through training)
- Training RecNNs involves similar drawbacks to those encountered for RNNs
  - Local minima
  - Slow convergence
  - Vanishing of the gradients

- Reservoir Computing represents a natural candidate for investigating efficient approaches to RecNNs modeling
- Extension of the Reservoir Computing approach to structured domains

Tree Echo State Networks (TreeESNs)

- Extend the applicability of the RC/ESN approach to tree structured data
- Extremely efficient way of modeling RecNNs
- Architectural and experimental performance baseline for trained RecNN models

  - Generalized Reservoir: bottom-up recursive encoding process (untrained)
  - Readout: output computation (trained)
  - State Mapping Function

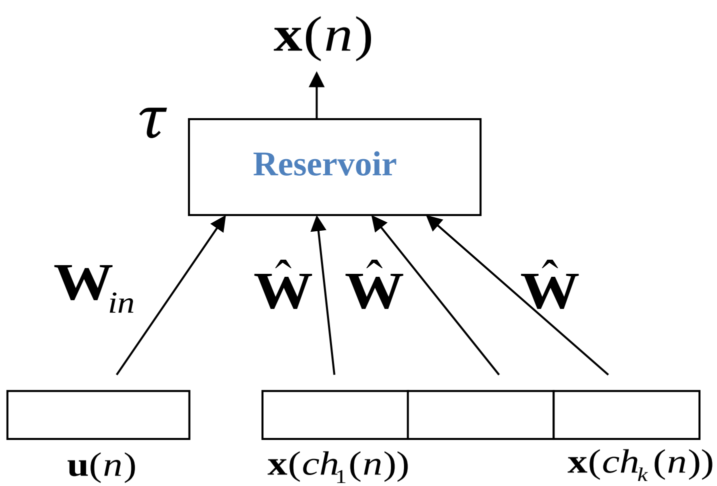


Tree Echo State Networks

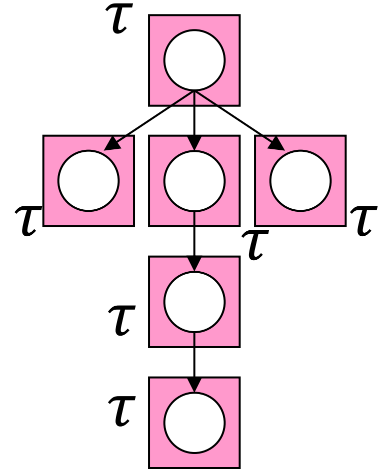
Reservoir

- Large, sparsely connected, untrained layer of non-linear recursive units
- Implements the local encoding function τ
- Contractive state transition system on trees

Reservoir Application to an Input Node



Bottom-up Recursive Processing of Trees



- The same reservoir architecture is applied to each node
- Run only once: from the leaves to the root

- Each reservoir unit is fed by: node label and states already computed for children
- Connection between two reservoir units carries all the state information for the children

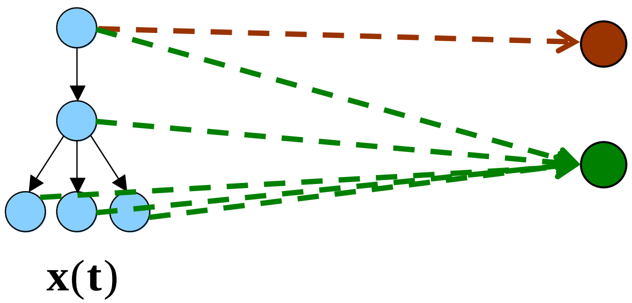
$$\tau : \mathbb{R}^{N_U} \times \mathbb{R}^{k\ N_R} \rightarrow \mathbb{R}^{N_R}$$

$$\boldsymbol{x}(n) = \tanh(\boldsymbol{W}_{in}\boldsymbol{u}(n) + \sum_{i=1}^{k} \widehat{\boldsymbol{W}}\boldsymbol{x}(ch_i(n)))$$

Tree Echo State Networks

State Mapping Function

- Maps the tree structured state into a fixed-size state
- Influence on the characterization of the model dynamics



Root State Mapping $\chi(\mathbf{x}(\mathbf{t})) = \mathbf{x}(root(\mathbf{t}))$

Mean State Mapping $\chi(\mathbf{x}(\mathbf{t})) = (1/|N(\mathbf{t})|) \sum_{n \in N(\mathbf{t})} \mathbf{x}(n)$

Readout

$$\mathbf{y}(\mathbf{t}) = \mathbf{W}_{out}\mathbf{x}(\mathbf{t})$$

- The linear readout implements the local output function
- Training as in ESN case (e.g. off-line by pseudo-inversion or ridge regression)

Tree Echo State Networks

A tree suffix of $\boldsymbol{t}$ of height h is denoted by $S_h(\boldsymbol{t})$

$$S_h(\boldsymbol{t}) = \begin{cases} nil, \ t = nil \ or \ h = 0 \\ n(S_{h-1}(\boldsymbol{t}(ch_1(n))), \dots, S_{h-1}\big(\boldsymbol{t}(ch_k(n))\big)), \ \boldsymbol{t} = n(\boldsymbol{t}(ch_1(n)), \dots, \boldsymbol{t}(ch_k(n))) \end{cases}$$

Markovianity

A state model on tree domains is characterized by a state space organization of a Markovian nature whenever the states it assumes in correspondence of different input trees sharing a common suffix, are close to each other proportionally to the height of the suffix.

Tree Echo State Networks

Contractivity

The node-wise encoding function τ is a contraction with respect to the state space $\mathbb{R}^{N_R}$

$$\exists\, C \in \mathbb{R},\, 0 \leq C < 1$$
$$\forall \boldsymbol{u} \in \mathbb{R}^{N_U}, \forall \boldsymbol{x}_1, \dots, \boldsymbol{x}_k, \boldsymbol{x_1}', \dots, \boldsymbol{x_k}' \in \mathbb{R}^{N_R}$$
$$\|\tau(\boldsymbol{u}, \boldsymbol{x}_1, \dots, \boldsymbol{x}_k) - \tau(\boldsymbol{u}, \boldsymbol{x}_1', \dots, \boldsymbol{x}_k')\| \leq C \max_{i=1,\dots,k} \|\boldsymbol{x}_i - \boldsymbol{x}_i'\|$$

Contractivity + Bounded state space: Markovian characterization of TreeESN dynamics

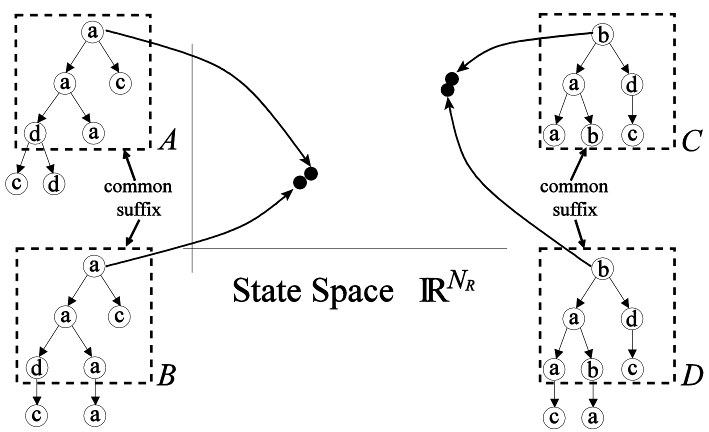
Tree Echo State Networks

Markovianity

Contractivity of Reservoir Dynamics

- Inherited from ESN for sequences
- Ensures stability of the encoding process
- Markovian organization of TreeESN state space

Markovian Characterization of TreeESN Dynamics



$\forall \mathbf{t}, \mathbf{t}' \in (\mathbb{R}^{N_U})^{\#k}$ sharing a common suffix of height *h*

$$\forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^{N_R}$$

$$\|\hat{\tau}(\mathbf{t}, \mathbf{x}) - \hat{\tau}(\mathbf{t}', \mathbf{x}')\| \leq C^h \; diam$$

- Implies a tree version of the Echo State Property
- The reservoir of TreeESN is able to discriminate among input trees in a Markovian tree suffix –based way without any training
- Suitable for tasks with target functions compatible with Markovianity

Contractive Initialization

$$\sigma = k \; \|\hat{\mathbf{W}}\|_2 < 1$$

Assuming Euclidean distance as metric in the reservoir space

Tree Echo State Networks

Computational Complexity

Extremely efficient RC approach: only the linear readout parameters are trained

Encoding Process

For each tree **t**

$$O(|N(\mathbf{t})| \; k \; R \; N_R)$$

number of nodes — max degree — degree of connectivity — number of reservoir units

- Scales linearly with the number of nodes and the reservoir dimension
- The same cost for training and test
- Compares well with state of art methods for trees:
    - RecNNs: extra cost (time + memory) for gradient computations
    - Kernel methods: higher cost of encoding (e.g. Quadratic in PT kernels)

Output Computation

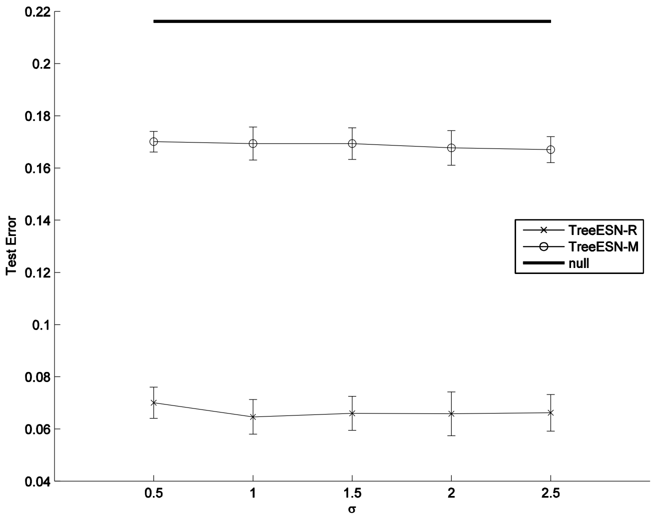
- Depends on the method used (e.g. Direct using SVD or iterative)
- The cost of training the linear TreeESN readout is generally inferior to the cost of training MLPs or SVMs (used in RecNNs and Kernels)

Tree Echo State Networks

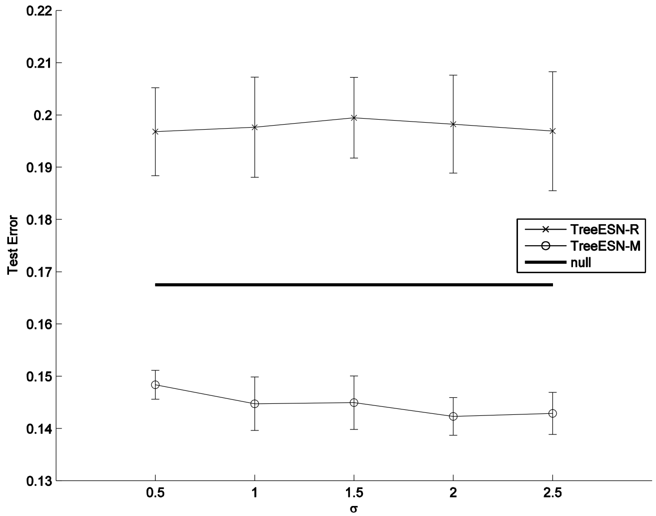
Experiments

Markovian/anti-Markovian Tasks

- Target functions with Markovian/anti-Markovian characterization (tight control on Markovianity)
- Relevant influence of the choice of the state mapping function



Markovian task



Anti-Markovian task

Root State Mapping

- Better than mean state mapping on Markovian task (independently on the degree of contractivity)
- Worse than *null model* on the anti-Markovian task

Mean State Mapping

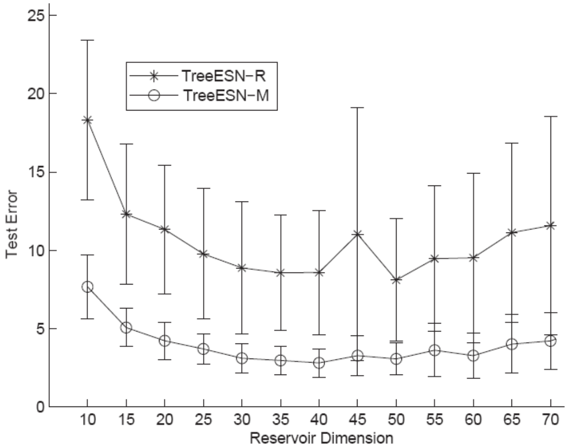
- Outperforms TreeESN with root state mapping on anti-Markovian Task (but not sufficient to solve it)
- Almost the same performance on the two tasks (prefixes and suffixes are merged together)

Tree Echo State Networks

Experiments

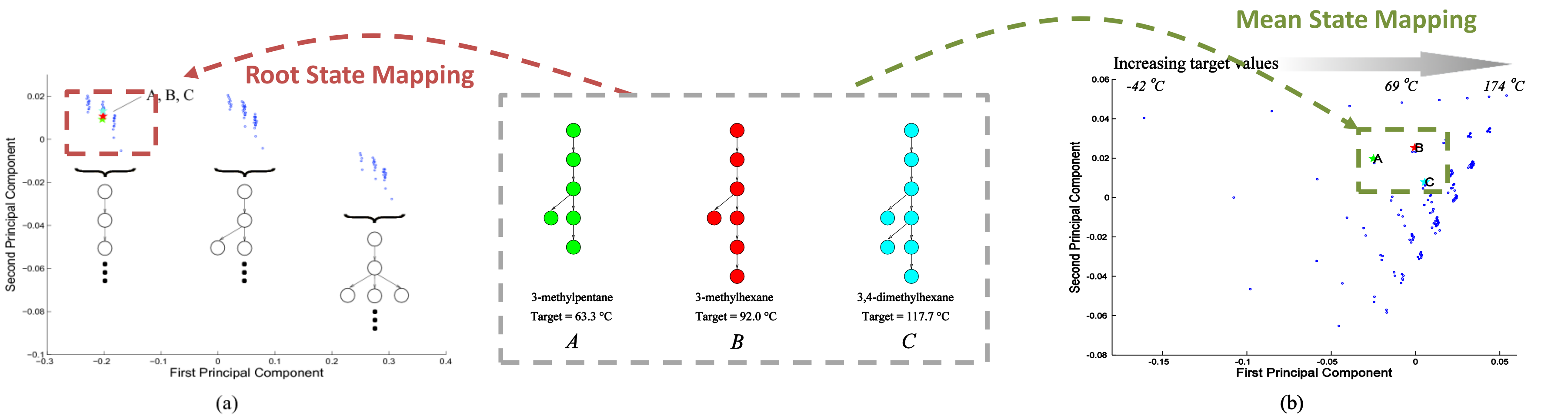
QSPR Analysis of Alkanes

- Predict the boiling point of alkanes
- Target is related to global properties of the molecules (num of carbons + branching pattern): non Markovian



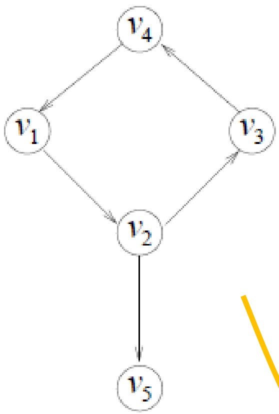
Model	ϵ_t	Test Set MAE
TreeESN-R	*best*	8.09(±3.91)
TreeESN-R	8^oC	15.01(±9.24)
TreeESN-R	5^oC	13.18(±8.58)
TreeESN-M	*best*	2.78(±0.90)
TreeESN-M	8^oC	3.09(±0.93)
TreeESN-M	5^oC	3.05(±1.05)
RCC	8^oC	2.87(±0.91)
CRCC	8^oC	2.56(±0.80)
SST	8^oC	2.93(±0.92)
NN4G	8^oC	2.34(±0.31)
NN4G	5^oC	1.74(±0.23)

- Performance is sensible to the choice of state mapping function
- Though analysis aim: reasonable results respect to state-of-the-art

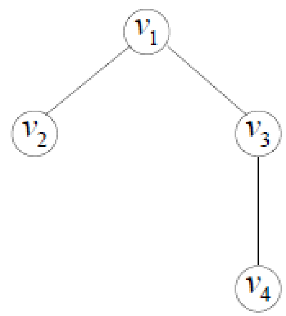


Dealing with Cycles and Undirected Graphs

- Dealing with cyclic/undirected structures represents an issue due to the causal assumption



$\mathbf{g}_1$



$\mathbf{g}_2$

- In case of undirected graphs, the state computed for each vertex depends on the state computed for its *neighbors*

$$\mathcal{N}(v) = \{u \in V(\mathrm{g}) | \exists (u, v) \in E(\mathrm{g})\}$$

- Mutual dependencies among the states

$$\begin{aligned}
& \mathrm{x}(v_5) = f(\mathbf{W}_{in}\mathbf{u}(v_5) + \hat{\mathbf{W}}\mathrm{x}(nil)) \\
* \quad & \mathrm{x}(v_2) = f(\mathbf{W}_{in}\mathbf{u}(v_2) + \hat{\mathbf{W}}\mathrm{x}(v_3)) \\
* \quad & \mathrm{x}(v_3) = f(\mathbf{W}_{in}\mathbf{u}(v_3) + \hat{\mathbf{W}}\mathrm{x}(v_4)) \\
* \quad & \mathrm{x}(v_4) = f(\mathbf{W}_{in}\mathbf{u}(v_4) + \hat{\mathbf{W}}\mathrm{x}(v_1)) \\
* \quad & \mathrm{x}(v_1) = f(\mathbf{W}_{in}\mathbf{u}(v_1) + \hat{\mathbf{W}}\mathrm{x}(v_2))
\end{aligned}$$

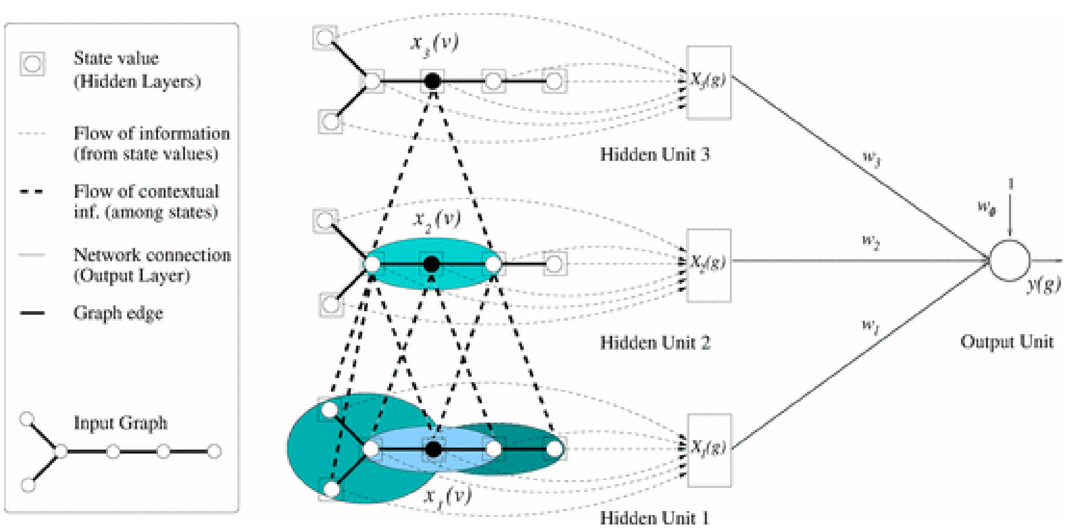
- RecNNs traditionally unsuitable for processing cyclic and undirected graphs

- Two approaches: explicitly treat the cycles constraining state dynamics (GraphESN, GNN), or contextual non-recursive approach (NN4Gs)

Neural Networks for Graphs (NN4Gs)

- Recently proposed model for processing general classes of graphs
- Encoding transduction implemented by a non-recursive state transition function
- The encoding process is non-recursive and can be computed without stability issues
- Overcome the causal assumption: directly deal with cyclic/acyclic, directed/undirected graphs
- Contextual, constructive approach

$$x_l(v) = \begin{cases} f(\mathbf{W}_{in}\mathbf{u}(v)) & if \;\; l = 1 \\ f(\mathbf{W}_{in}\mathbf{u}(v) + \sum_{j=1}^{l-1} \sum_{v' \in \mathcal{N}(v)} \hat{w}_{lj} \, x_j(v')) & otherwise \end{cases}$$



- The context window is incrementally extended when the number of hidden units is increased
- The output function is implemented by a layer of linear units
- For structure-to-element transduction a state-mapping-function is used

Kernel Methods for Graphs

- Extension of kernel methods for dealing with structured data directly
- Idea is to define a kernel function on the product space of the structured input domain

$$k : \mathcal{U}^{\#} \times \mathcal{U}^{\#} \rightarrow \mathbb{R}$$

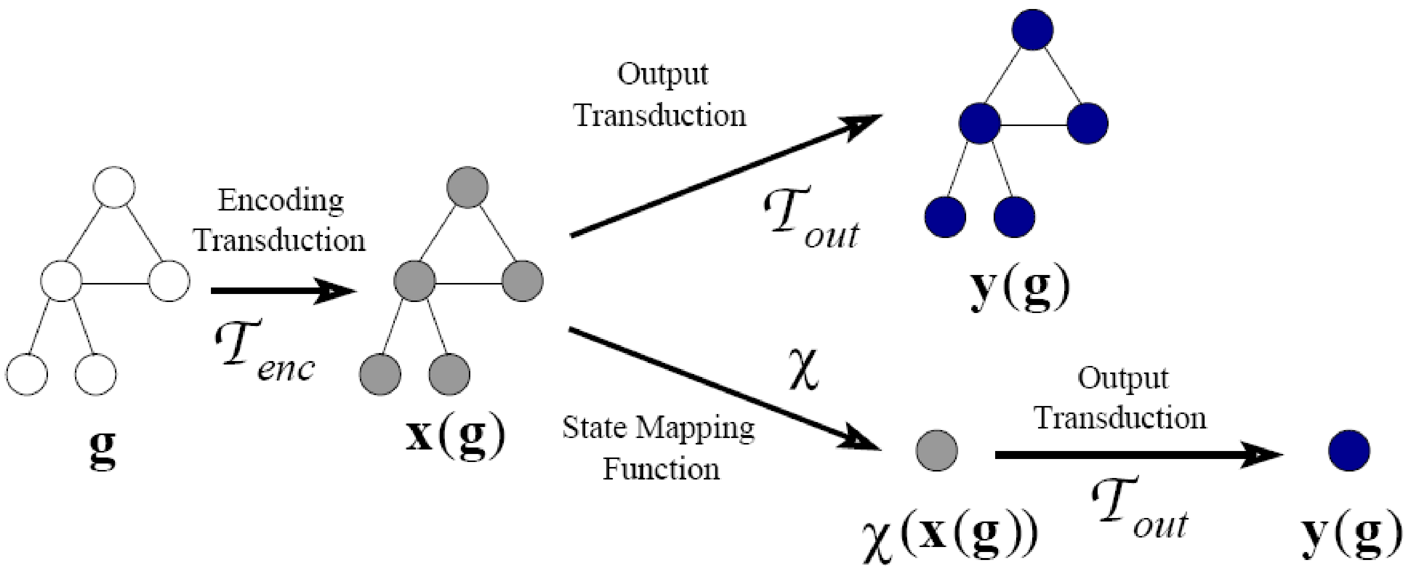
- Corresponds to the definition of a similarity measure on couples of instances in the structured input space
- The encoding transduction is implicitly computed by the kernel function, the output transduction is computed by a SVM

Examples: Marginalized Kernel, Optimal Assignment Kernel, EM Kernel, ….

Graph Echo State Networks

- GraphESN extends the applicability of RC to general graphs
- Dealing with general graphs brings expressive potential but possible explosion of computational cost with respect to the size of input

  - Generalized Reservoir: contractive encoding process (untrained)
  - Readout: output computation (trained)
  - State Mapping Function



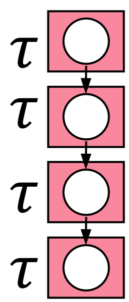
Graph Echo State Networks

Reservoir

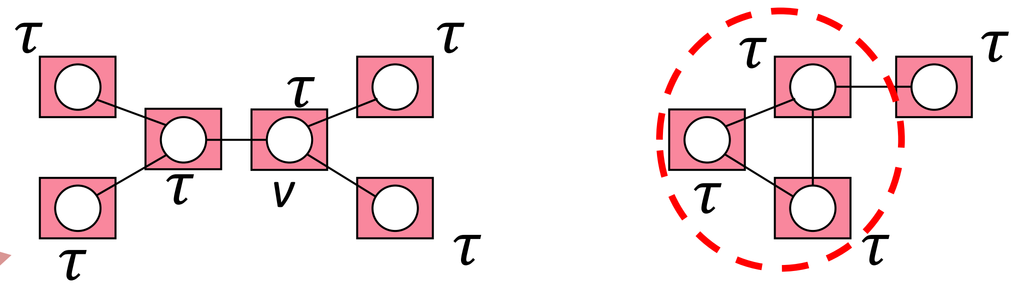
- Implements the local encoding function on graph patterns

$$\mathbf{x}(v) = tanh(\mathbf{W}_{in}\mathbf{u}(v) + \sum_{v' \in V(\mathbf{g})} \hat{\mathbf{W}}\mathbf{x}(\mathcal{N}_i(v')))$$

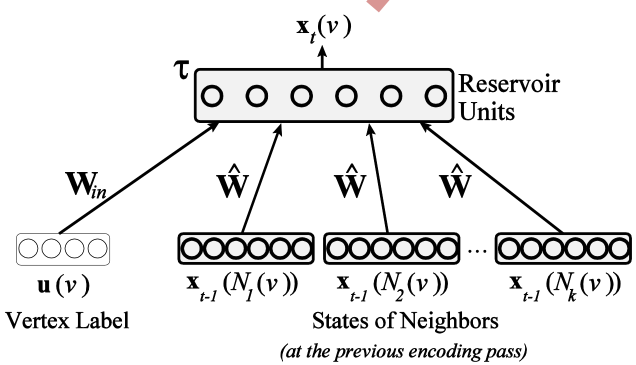
Standard ESN



GraphESN



State transition system on graphs



Contractivity of Reservoir Dynamics

- Inherited from ESNs and TreeESNs
- Guarantees stability of the encoding process (Banach Th.)
- Extends applicability to cyclic/undirected graphs
- Markovian nature of reservoir space organization
- Iterative encoding process

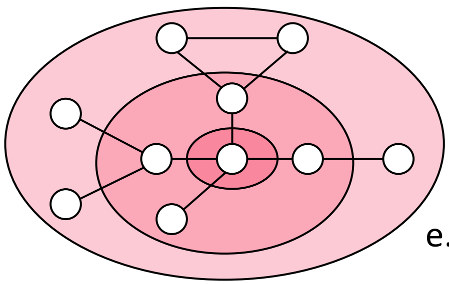
$$\mathbf{x}_t(v) = tanh(\mathbf{W}_{in}\mathbf{u}(v) + \sum_{v' \in V(\mathbf{g})} \hat{\mathbf{W}}\mathbf{x}_{t-1}(\mathcal{N}_i(v)))$$

Initialization $\sigma = k \, \|\hat{\mathbf{W}}\|_2 < 1$

Graph Echo State Networks

Markovianity

- Contractivity of the state transition function implies reservoir dynamics with Markovian flavour
- Suffix: the concept is extended to the set of d-neighbors of a vertex *v*, i.e. $N^{(d)}(v)$



e.g. d = 2

Markovian Characterization of GraphESN Dynamics

$$\forall \mathbf{g}, \mathbf{g}' \in (\mathbb{R}^{N_U})^{\#k}$$

$$\forall v \in V(\mathbf{g}), v' \in V(\mathbf{g}') \quad \text{such that } N^{(d)}(v) = N^{(d)}(v')$$

$$\|\boldsymbol{x}(v) - \boldsymbol{x}(v')\|_2 \leq C^d diam$$

- Ability to discriminate among graph patterns in a suffix-based Markovian way without learning of the recursive connections (untrained reservoir)
- Architectural baseline
- Tasks within Markovian characterization can be approached very efficiently by GraphESNs
- Limit of the model, unsuitableness for tasks with no Markovian assumptions

Computational Complexity

Exploits extreme efficiency of RC approach

Encoding Process

$$O(|V(\mathbf{g})|\ k\ R\ N_R)$$

For each graph **g**,
for each pass of the encoding process

- $|V(\mathbf{g})|$: number of nodes
- k: max degree
- R: degree of connectivity
- N_R: number of reservoir units

- Scales linearly with the number of nodes and the reservoir dimension
- The same cost for training and testing
- Compares well with state of art methods for graph domains:
  - GNN: (as in GraphESN + learning) x number of epochs
  - Kernel methods: quadratic (e.g. EM Kernel), cubic (e.g. OA Kernel)

Output Computation

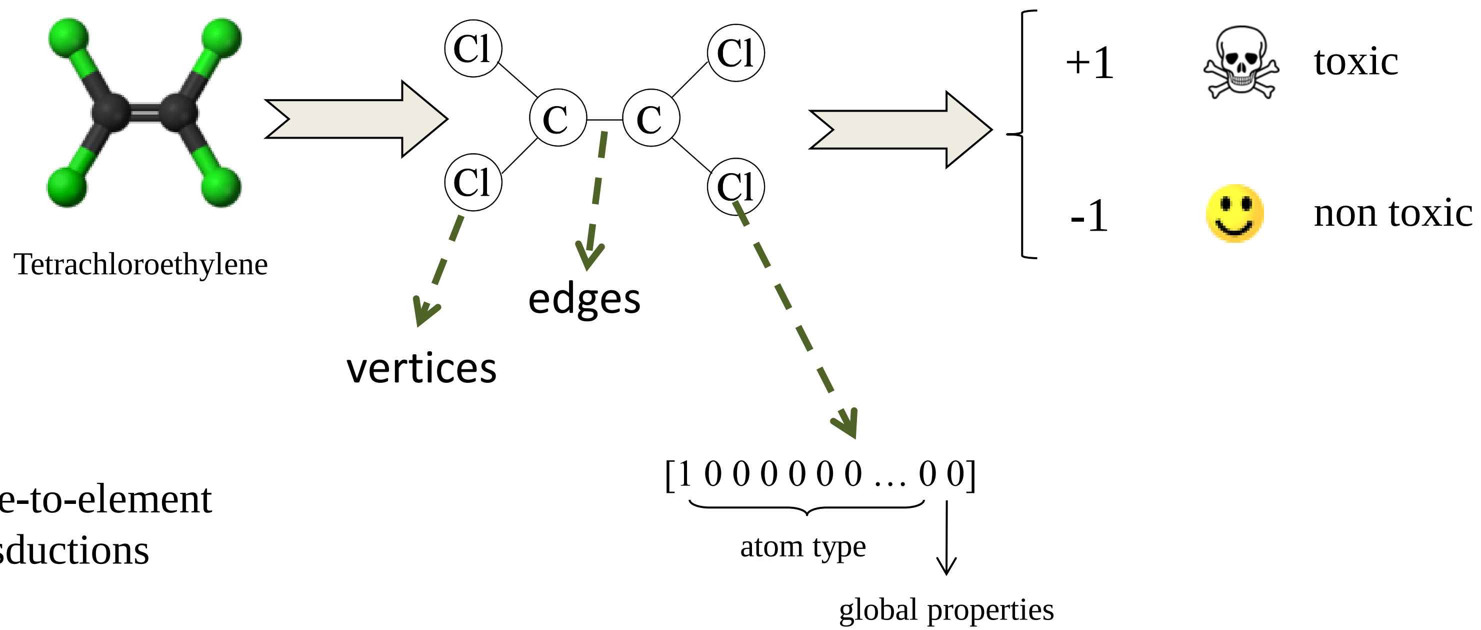
- Depends on the method used
- Inferior to the cost of training MLPs or SVMs (used in RecNNs and Kernels)

Graph Neural Networks (GNNs)

- Stability of the recursive encoding process is guaranteed by resorting to contractive state dynamics (like in GraphESN)
- The error function in the gradient descent learning algorithm includes a penalty term (to penalize non-contractive state transition functions)
- State relaxation – gradient computation phases are alternated
- Reduced efficiency with respect to GraphESNs

Predictive Toxicology Challenge (PTC) Dataset

- Carcinogenicity information for 417 molecules
- Data concerns 4 classes of rodents: Male Rats (MR), Female Rats (FR), Male Mice (MM), Female Mice (FM)
- Classificaton Task (carcinogenic molecule +1, non-carcinogenic molecule -1)
- Molecules are represented as undirected graphs



Structure-to-element transductions

PTC Dataset– SDF Format

```
WItclserve11290013443D 0   0.00000     0.00000cramer

 25 26  0  0  0  0  0  0  0  0  2 V2000

 0.7143    0.6231   -0.1367 C   0  0  0  0  0  0  0  0  0  0  0  0
 1.6445    1.6447   -0.1115 C   0  0  0  0  0  0  0  0  0  0  0  0
   ...
  3.2657    0.0876    2.1403 H   0  0  0  0  0  0  0  0  0  0  0  0
 -1.0455   -0.9737    2.0776 H   0  0  0  0  0  0  0  0  0  0  0  0
```

Atom block

Atom element

```
  1  2  1  0  0  0  0
  2  3  2  0  0  0  0
  3  4  1  0  0  0  0
     ...
 15 25  1  0  0  0  0
```

Bond block

Edges information

```
M  CHG  2  16   1  18  -1
M  END
```

Property Block

Global information for the atom label

```
> <RecNN.name>
TR026

> <PTC.CLASS.FR>
+1

> <PTC.CLASS.MM>
+1

> <PTC.CLASS.FM>
+1
```

Additional Information

Target Information

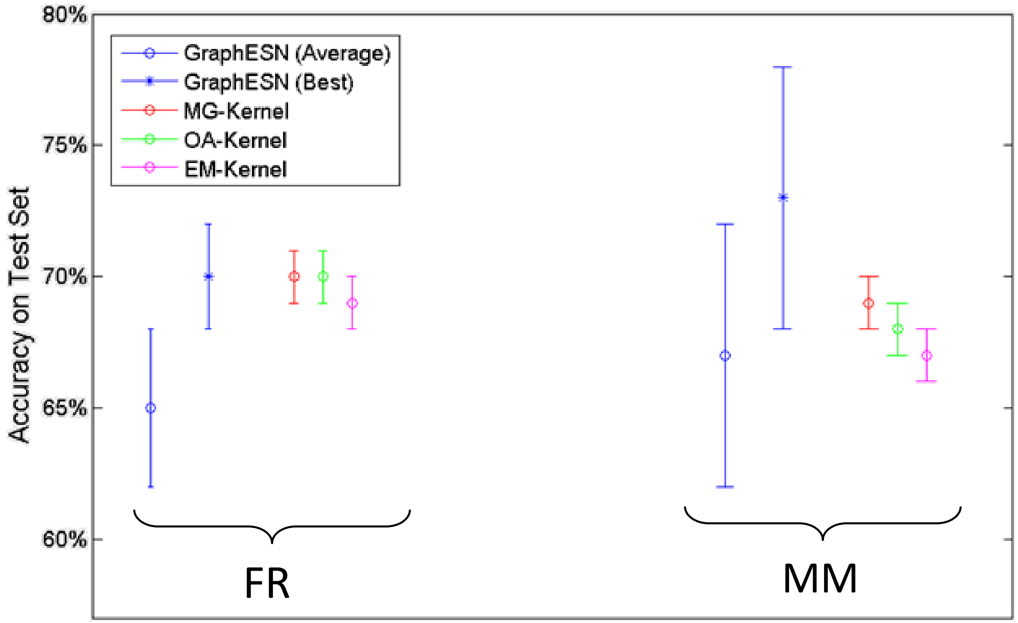
Molecule Name

Graph Echo State Networks

Experiments

Predictive Toxicology Challenge (PTC) Dataset

- Model selection on GraphESN hyper-parameters (by cross fold validation)



	MR	FR	MM	FM
Average TS	57% (± 4%)	65% (± 3%)	67% (± 5%)	58% (± 4%)
Best TS	63% (± 4%)	70% (± 2%)	73% (± 5%)	65% (± 5%)

Graph Echo State Networks

Experiments

Mutagenesis Dataset

- Mutagenicity of nitroaromatic compounds
- Classificaton Task
- Different descriptions of the molecules are available (AB, C, PS)

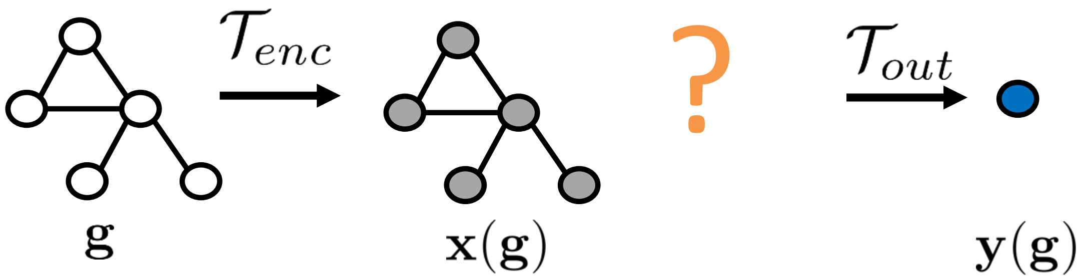
Model		AB	AB+C	AB+C+PS
RDBC		83%	82%	
TILDE		77%	82%	
1nn(d*m*)		81%	88%	
GNN				86%
GraphESN	*Average*	72%(±4%)	82%(±7%)	82%(±7%)
Supersource S.M.	*Best*	81%(±3%)	89%(±7%)	88%(±8%)
GraphESN	*Average*	76%(±9%)	80%(±6%)	80%(±6%)
Mean S.M.	*Best*	86%(±7%)	88%(±8%)	87%(±6%)

- State-of-the-art results within the range of GraphESN performance
- Relevance of the contractive assumption

Adaptivity of State Mappings for GraphESN

Motivations

- State transition systems naturally unsuitable for graph-to-element transductions

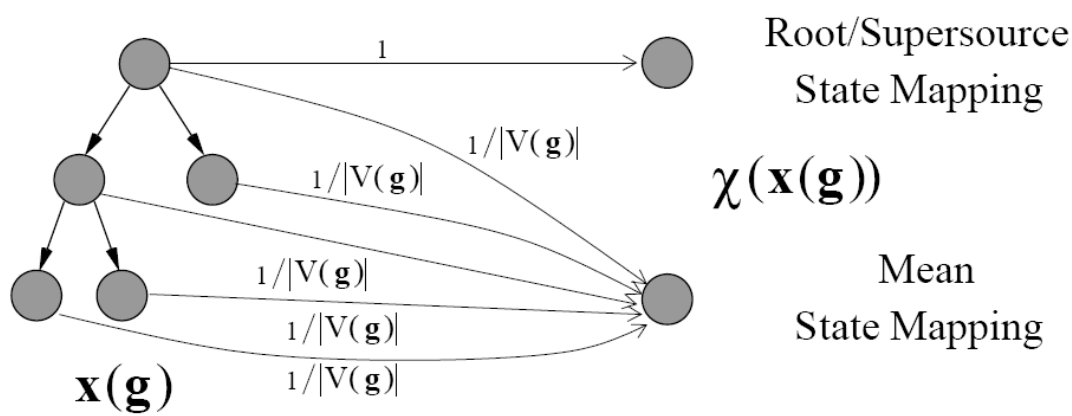


- Extract the relavant information from structured state spaces
- Weight the relevance of each vertex on the output
- Deal with general graphs with variable size and topology (no vertices alignments)

State Mapping Function

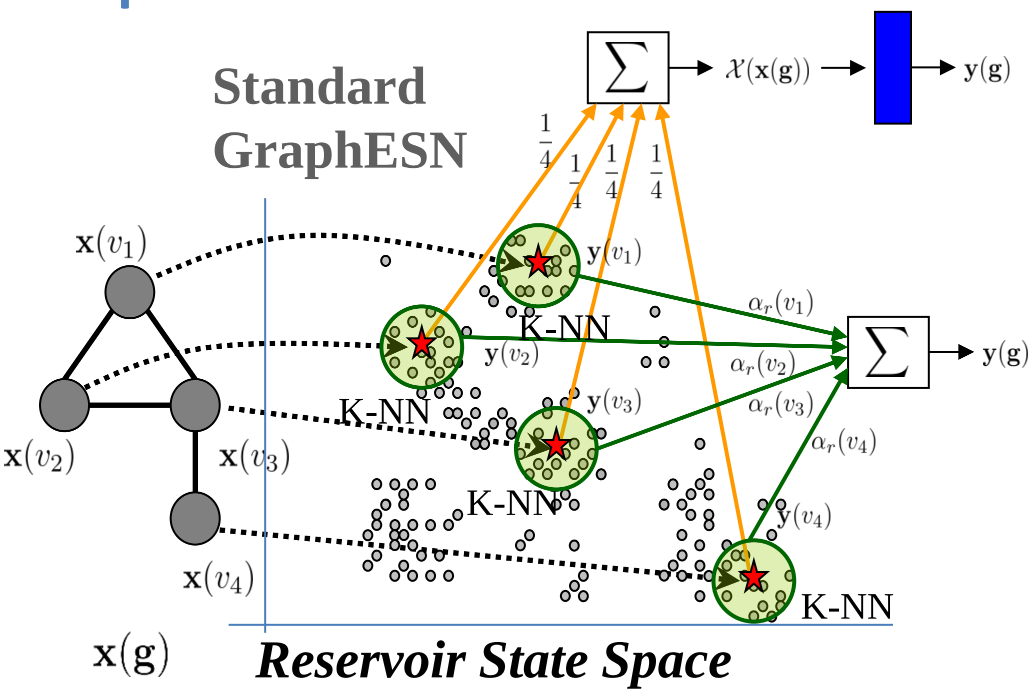
$$\chi : (\mathbb{R}^{N_R})^{\#} \rightarrow \mathbb{R}^{N_R}$$

- Relevant effect
- Critical role in applications (in relation to the target properties)
- **Flexible/adaptive state mapping functions**



Adaptivity of State Mappings for GraphESN

GraphESN-wnn



- Readout implemented using distance-weighted K-neares neighbor
- Weights the contribution of each vertex according to a fixed scheme
- Flexible/supervised extraction of information from the reservoir state space
- Stronger influence of vertices whose states are in regions corresponding to more uniform target information

$$\mathbf{y}(v) = \frac{\sum_{i=1}^{K} w_i^{(v)} \mathbf{y}_{tg}(v_i^N)}{\sum_{i=1}^{K} w_i^{(v)}} \qquad w_i^{(v)} = \frac{1}{\|\mathbf{x}(v) - \mathbf{x}(v_i^N)\|_2^2}$$

$$\alpha_r(v) = \frac{\sum_{i=1}^{K} w_i^{(v)}}{\sum_{i=1}^{K} w_i^{(v)} (\mathbf{y}(v) - \mathbf{y}_{tg}(v_i^N))^2}$$

$$\mathbf{y}(\mathbf{g}) = \frac{\sum_{v \in V(\mathbf{g})} \alpha_r(v) \mathbf{y}(v)}{\sum_{v \in V(\mathbf{g})} \alpha_r(v)}$$

PTC Dataset

Model selection on reservoir parameters, K, readout reg.

Model	*MM*	*FM*	*MR*	*FR*
GraphESN	62.87(±1.2)	60.40(±1.7)	59.43(±1.9)	64.44(±0.9)
GraphESN-wnn	63.04(±2.7)	63.32(±2.6)	58.02(±2.1)	67.37(±2.5)

$$K \in \{1, 5, 15, 30, 50\}$$

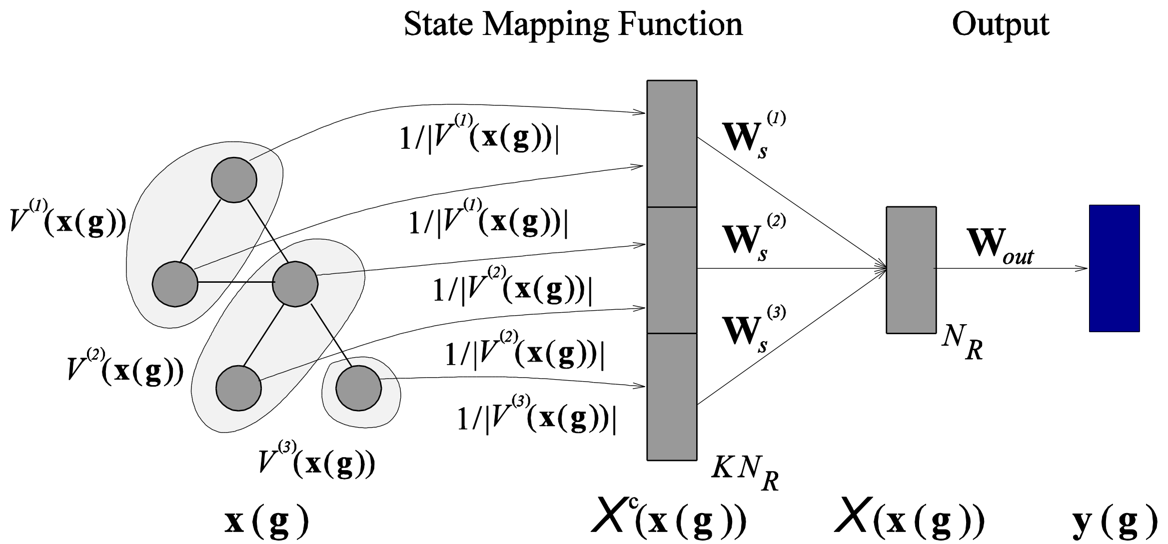
Best reservoir setting after model selection on the readout

Model	*MM*	*FM*	*MR*	*FR*
GraphESN	68.45(±2.4)	64.77(±3.5)	65.99(±2.6)	68.95(±2.2)
GraphESN-wnn	69.65(±2.7)	67.91(±4.8)	67.43(±4.5)	69.25(±3.1)
MG-Kernel	69.05(±1.5)	64.76(±1.2)	62.50(±1.2)	70.09(±0.6)
OA-Kernel	67.87(±1.7)	65.33(±0.9)	63.39(±2.1)	70.37(±1.1)
EM-Kernel	66.97(±1.1)	64.47(±1.2)	60.84(±1.7)	68.95(±0.7)

Adaptivity of State Mappings for GraphESN

GraphESN-NG

- Fully adaptively weight (through readout learning) the relevance of the states of each vertex in the state mapping computation



- Neural Gas (NG) clustering algorithm is used to cluster the reservoir space
- For each graph g, average the state information locally to each cluster and then combined with free parameters for the output computation

$$\chi(\mathbf{x}(\mathbf{g})) = \sum_{i=1}^{K} \mathbf{W}_s^{(i)} \chi^{(i)}(\mathbf{x}(\mathbf{g}))$$

- Supervised approach for the adaptation of the state mapping computation
- For K = 1 GraphESN is obtained

Adaptivity of State Mappings for GraphESN

GraphESN-NG - Experiments

Effectiveness of the adaptive approach for state mapping functon computation

Model selection on the hyper-parameters by double cross fold validation

PTC Dataset

Task	K = 1 (baseline)	K = 5	K = 10	K = 30
MR	57.27(±3.33)	59.24(±2.88)	60.00(±3.13)	61.18(±2.20)
FR	67.12(±0.14)	66.76(±1.67)	64.44(±1.76)	65.06(±2.10)
MM	65.00(±0.66)	64.86(±1.38)	62.67(±2.26)	64.40(±3.13)
FM	60.42(±0.86)	62.49(±1.71)	60.75(±3.57)	57.38(±2.16)

Performance comparable to MG and OA kernels

Bursi Dataset

Mutagenicity of chemicals. Large, high quality dataset.

K = 1 (baseline)	K = 5	K = 10	K = 30
75.82(±0.55)	77.20(±0.58)	78.11(±0.72)	79.24(±0.64)

GraphESN-NG outperforms competitive state-of-the-art methods (*lazar*, Benigni/Bossa structural alerts)

Conclusions

- Learning in Structured Domains: opens up a wide range of research directions, applications + research issues
- Transductions on trees and graphs
- Extension of the Reservoir Computing paradigm for trees: TreeESN
- Extension of the Reservoir Computing paradigm for graphs: GraphESN
- Reservoir: non-linear dynamic component, untrained after contractive initialization used ot implement the vertex-wise encoding function
- Readout: linear feed-forward component, trained used to implement the vertex-wise output function
- State Mapping Function: influences the organization of the resulting state space
- Markovian flavour of reservoir state dynamics extended to the case of state transition systems on trees and graphs
- Successful applications
- Model Selection: many hyper-parameters to be set

References

- Tree Echo State Networks

    - C. Gallicchio, A. Micheli, Tree echo state networks, *Neurocomputing*, vol.101, pag. 319-337, 2013

- Graph Echo State Networks

    - C. Gallicchio, A. Micheli, Graph echo state networks, *International Joint Conference on Neural Networks (IJCNN)*, IEEE, pag. 1-8, 2010
    - C. Gallicchio, A. Micheli, Supervised State Mapping of Clustered GraphESN States, *Proceedings of the 21st Italian Workshop on Neural Nets (Wirn)*, Vol. 234, pag. 28-35. IOS Press, 2011

- Neural Network for Graphs

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