Reservoir Computing Methods

Basics and Recent Advances

Claudio Gallicchio

Contact Information

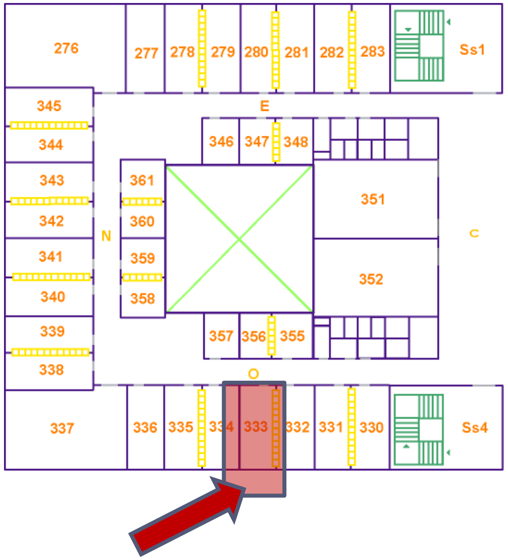
Claudio Gallicchio, Ph.D.
Assistant Professor
Computational Intelligence and Machine Learning Group
http://www.di.unipi.it/groups/ciml/

Dipartimento di Informatica - Universita' di Pisa
Largo Bruno Pontecorvo 3, 56127 Pisa, Italy – Polo Fibonacci

Research on Reservoir Computing

Chair of the IEEE Task Force on RC

https://sites.google.com/view/reservoir-computing-tf/home



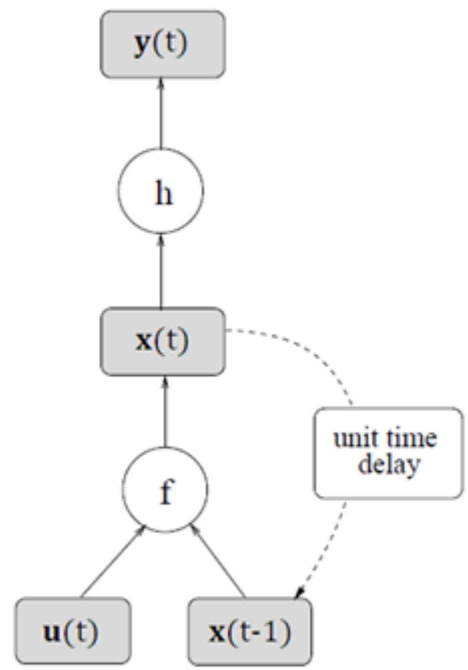
web: www.di.unipi.it/~gallicch
email: gallicch@di.unipi.it
tel.:+39 050 2213145

Dynamical Recurrent Models

- Neural network architectures with feedback connections are able to deal with *temporal data* in a natural fashion
- Computation is based on dynamical systems

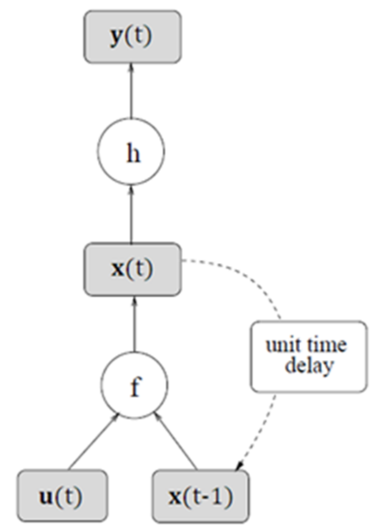
feed-forward component

dynamical component



Recurrent Neural Networks (RNNs)

- Feedbacks allows the representation of the temporal context in the state (neural memory)
- Discrete-time non-autonomous dynamical system
- Potentially the input history can be maintained for arbitrary periods of time
- Theoretically very powerful
  - Universal approximation through learning



Learning with RNNs *(repetita)*

- Universal approximation of RNNs (e.g. SRN, NARX) *through learning*
- Training algorithms involve some downsides that you already know
  - Relatively high computational training costs and potentially slow convergence
  - Local minima of the error function (which is generally non-convex)
  - Vanishing of the gradients and problem of learning long-term dependencies
    - Alleviated by gated recurrent architectures (although training is made quite complex in this case)

Dynamical Recurrent Networks trained easily

Question:

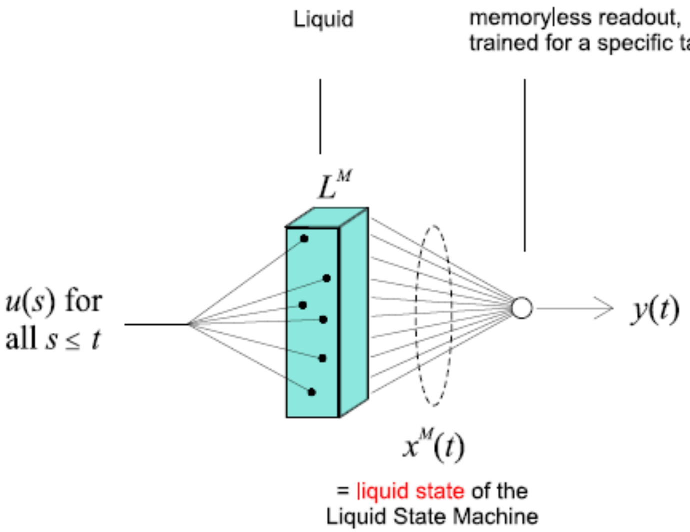
- Is it possible to train RNN architectures more efficiently?

- We can shift the focus from training algorithms to the study of initialization conditions and *stability* of the input-driven system
- To ensure stability of the dynamical part we must impose a contractive property to the system dynamics

Liquid State Machines

- W. Maas, T. Natschlaeger, H. Markram (2002)

  `W. Maass, T. Natschlaeger, and H. Markram, Real-time computing without stable states: A new framework for neural computation based on perturbations, Neural Computation. 14(11), 2531-2560, (2002)`



$$x^M(t) = (L^M u)(t)$$
$$y(t) = f^M(x^M(t))$$

Integrate-and-fire

$$\tau_m \frac{du}{dt} = -u(t) + RI(t)$$

Izhikevich

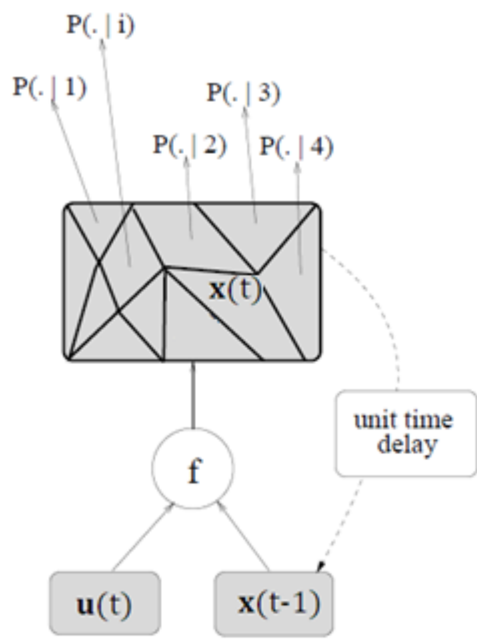
$$\frac{dv}{dt} = 0.04v^2 + 5v + 140 - u + I,$$
$$\frac{du}{dt} = a(bv - u)$$

- Originated from the study of biologically inspired spiking neurons
- The liquid should satisfy a pointwise separation property
- Dynamics provided by a pool of spiking neurons with bio-inspired arch.

Fractal Prediction Machines

- P. Tino, G. Dorffner (2001)

  Tino, P., Dor®ner, G.: Predicting the future of discrete sequences from fractal representations of the past. Machine Learning 45 (2001) 187-218



$$\begin{aligned}\mathbf{x}(t) &= f(\mathbf{u}(t) + \mathbf{x}(t-1)) \\ &= \rho\mathbf{x}(t-1) + (1-\rho)\mathbf{u}(t)\end{aligned}$$

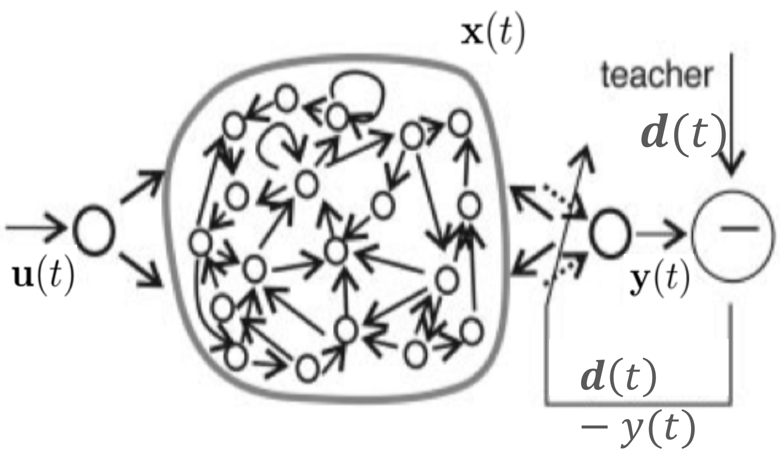
- Contractive Iterated Function Systems
- Fractal Analysis

Echo State Networks

- H. Jaeger (2001)

Jaeger, H.: The "echo state" approach to analysing and training recurrent neural networks. Technical Report GMD Report 148, German National Research Center for Information Technology (2001)

Jaeger, H., Haas, H.: Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. Science 304 (2004) 78-80



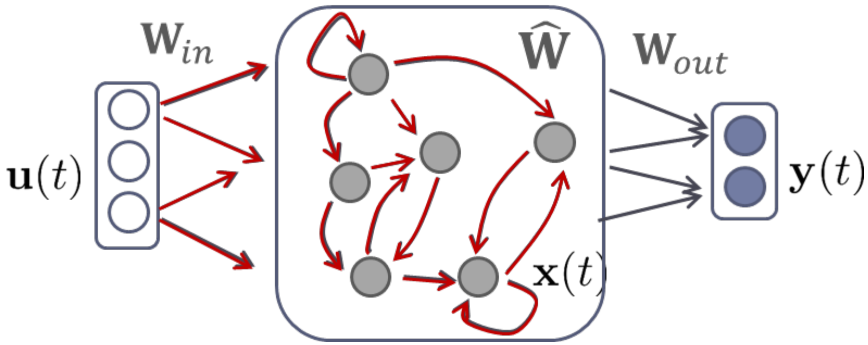
$$\mathbf{x}(t) = tanh(\mathbf{W}_{in}\mathbf{u}(t) + \hat{\mathbf{W}}\mathbf{x}(t-1))$$

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

- Control the spectral properties of the recurrence matrix
- Echo State Property

Reservoir Computing

- Reservoir: untrained non-linear recurrent hidden layer
- Readout: (linear) output layer



$$\mathbf{x}(\mathrm{t}) = \tanh(\mathbf{W}_{in}\boldsymbol{u}(t) + \widehat{\mathbf{W}}\,\mathbf{x}(\mathrm{t}-1))$$

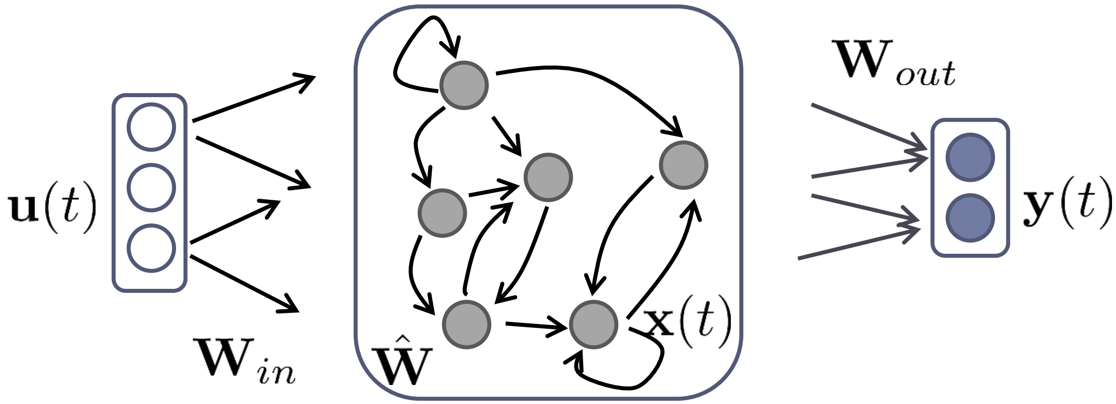
$$\mathbf{y}(\mathrm{t}) = \mathbf{W}_{out}\boldsymbol{x}(t)$$

- Initialize $\mathbf{W}_{in}$ and $\widehat{\mathbf{W}}$ randomly
- Scale $\widehat{\mathbf{W}}$ to meet the contractive/stability property
- Drive the network with the input signal
- Discard an initial transient
- Train the readout

Verstraeten, David, et al. "An experimental unification of reservoir computing methods." *Neural networks* 20.3 (2007): 391-403.

Echo State Networks

Echo state Network: Architecture



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

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

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

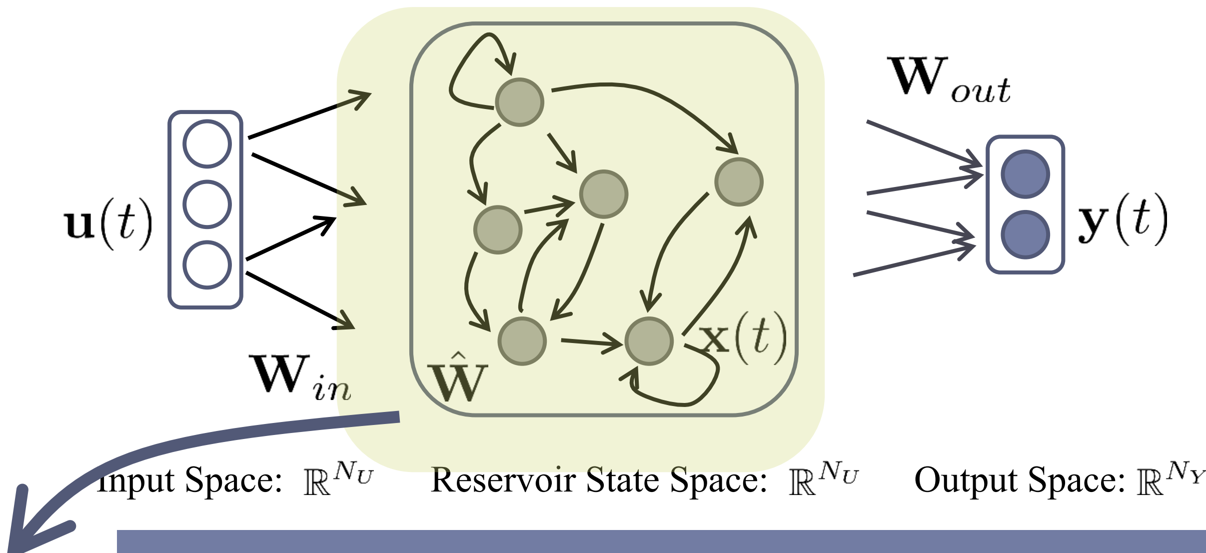
  $$\mathbf{x}(t) = tanh(\mathbf{W}_{in}\mathbf{u}(t) + \hat{\mathbf{W}}\mathbf{x}(t-1))$$

- Readout: trained, linear layer

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

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

Echo state Network: Architecture

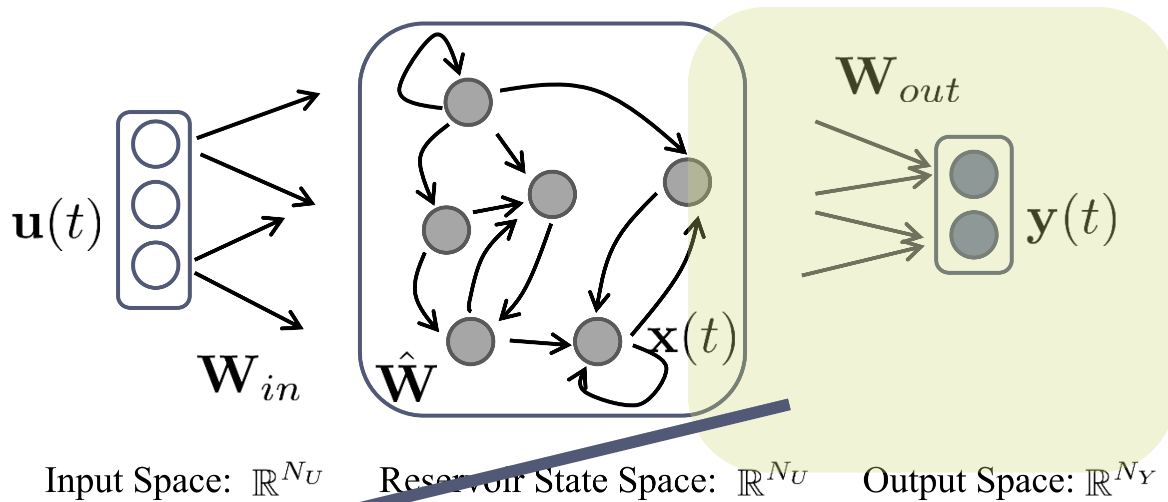


Efficient: the recurrent part is left completely untrained

Reservoir

- Non-linearly embed the input into a higher dimensional feature space where the original problem is more likely to be solved linearly (Cover's Th.)
- Randomized basis expansion computed by a pool of randomized filters
- Provides a "rich" set of input-driven dynamics
- Contextualize each new input given the previous state: memory

Echo state Network: Architecture



Readout

- Compute the features in the reservoir state space for the output computation
- Typically implemented by using linear models

Reservoir: State Computation

- The reservoir layer implements the state transition function of the dynamical system

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

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

- It is also useful to consider the iterated version of the state transition function
  - the reservoir state after the presentation of an entire input sequence

$\hat{F} : (\mathbb{R}^{N_U})^* \times \mathbb{R}^{N_R} \to \mathbb{R}^{N_R}$

$\forall \mathbf{s} \in (\mathbb{R}^{N_U})^*, \quad \forall \mathbf{x} \in \mathbb{R}^{N_R} \text{ initial state :}$

$$\hat{F}(\mathbf{s}, \mathbf{x}) = \begin{cases} \mathbf{x} & \text{if } \mathbf{s} = [\,] \\ F(\mathbf{u}(n), \hat{F}([\mathbf{u}(1), \ldots, \mathbf{u}(n-1)], \mathbf{x})) & \text{if } \mathbf{s} = [\mathbf{u}(1), \ldots, \mathbf{u}(n)] \end{cases}$$

Echo State Property (ESP)

A valid ESN should satisfy the "Echo State Property" (ESP)

- **Def**. An ESN satisfies the ESP whenever:

$$\forall \mathbf{s} = [\mathbf{u}(1), \ldots, \mathbf{u}(n)] \in (\mathbb{R}^{N_U})^n \text{ input sequence of length } n$$

$$\forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^{N_R} \text{initial states}:$$

$$\|\hat{F}(\mathbf{s}, \mathbf{x}) - \hat{F}(\mathbf{s}, \mathbf{x}')\| \to 0 \quad \text{as } n \to \infty$$

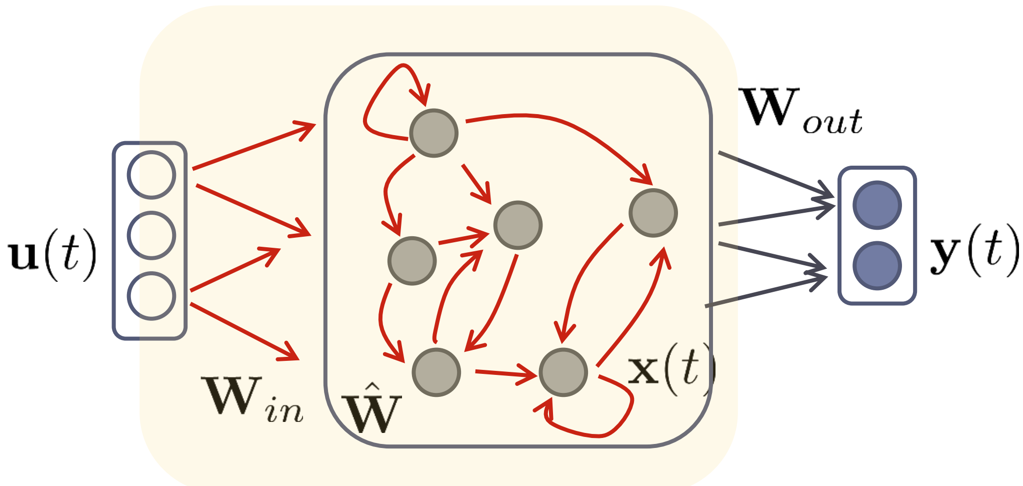
- The state of the network asymptotically depends only on the driving input signal
- Dependencies on the initial conditions are progressively lost
- Equivalent definitions: state contractivity, state forgetting and input forgetting

Conditions for the ESP

The ESP can be inspected by controlling the algebraic properties of the recurrent weight matrix $\hat{\mathbf{W}}$

- **Theorem.** If the maximum singular value of $\hat{\mathbf{W}}$ is less than 1 then the ESN satisfies the ESP.
  - Sufficient condition for the ESP (contractive dynamics for every input)
    $\sigma(\hat{\mathbf{W}}) = \|\hat{\mathbf{W}}\|_2 < 1$
- **Theorem.** If the spectral radius of $\hat{\mathbf{W}}$ is greater than 1 than (under mild assumptions) the ESN does not satisfy the ESP.
  - Necessary condition for the ESP (stable dynamics)
    $\rho(\hat{\mathbf{W}}) < 1$
  - recall: the spectral radius is the maximum among the absolute values of the eigenvalues
    $\rho(\hat{\mathbf{W}}) = max(|eig(\hat{\mathbf{W}})|)$

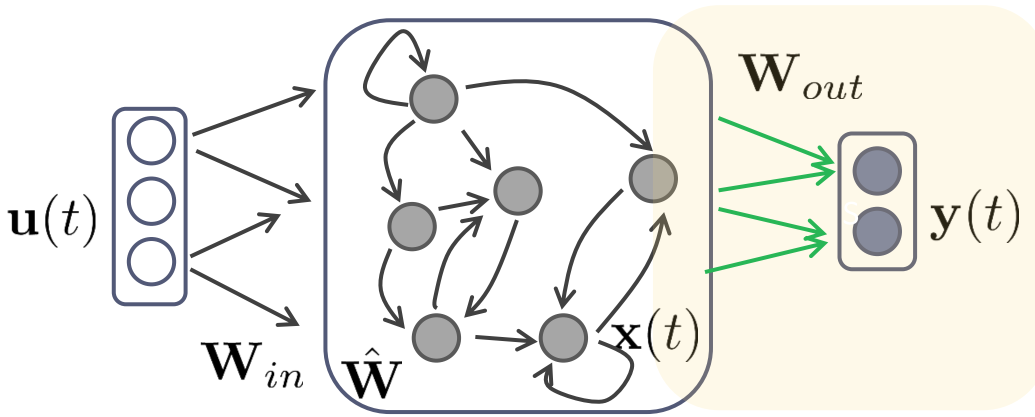
ESN Initialization: How to setup the Reservoir



- Elements in $\mathbf{W}_{in}$ are selected randomly in $[-scale_{in}, scale_{in}]$
- $\widehat{\mathbf{W}}$ initialization procedure:
  - Start with a randomly generated matrix $\widehat{\mathbf{W}}_{rand}$
  - Scale $\widehat{\mathbf{W}}_{rand}$ to meet the condition for the ESP (usually: the necessary one)

$$\hat{\mathbf{W}} = \hat{\mathbf{W}}_{rand} \frac{\rho_{desired}}{\rho(\hat{W}_{rand})}$$

ESN Training



- Run the network on the whole input sequence and collect the reservoir states

$$\mathbf{X} = [\mathbf{x}(1) \ldots \mathbf{x}(N)] \qquad \mathbf{Y}_{tg} = [\mathbf{y}_{tg}(1) \ldots \mathbf{y}_{tg}(N)]$$

- Discard an initial transient (washout)
- Solve the least squares problem defined by

$$min_{\mathbf{W}_{out}} \|\mathbf{W}_{out}\mathbf{X} - \mathbf{Y}_{tg}\|_2^2$$

Training the Readout

- On-line training is not the standard choice for ESNs
  - Least Mean Squares is typically not suitable
    - High eigenvalue spread (i.e. large condition number) of $\mathbf{X}$
  - Recursive Least Squares is more suitable
- Off-line training is standard in most applications
- Closed form solution of the least squares problem by direct methods
  - Moore-Penrose pseudo-inversion

    $$\mathbf{W}_{out} = \mathbf{Y}_{tg}\mathbf{X}^{+} = \mathbf{Y}_{tg}\mathbf{X}^{T}(\mathbf{X}\mathbf{X}^{T})^{-1}$$

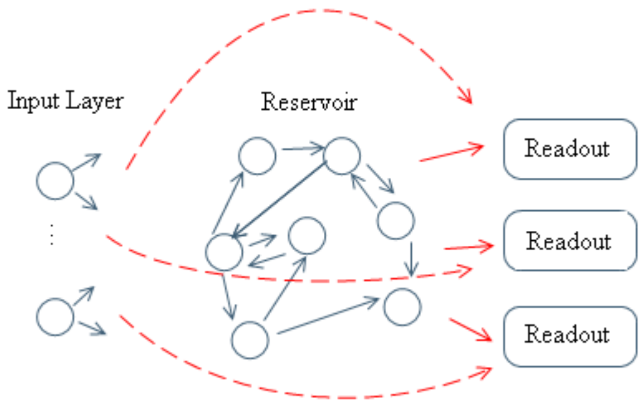
    - Possible regularization using random noise in the states
  - Ridge-regression

    $$\mathbf{W}_{out} = \mathbf{Y}_{tg}\mathbf{X}^{T}(\mathbf{X}\mathbf{X}^{T} + \lambda_r\mathbf{I})^{-1}$$

    - $\boldsymbol{\lambda_r}$ is a regularization coefficient (the higher, the more the readout is regularized)

Training the Readout/2

- Multiple readouts for the same reservoir
  - Solving more than 1 task with the same reservoir dynamics



- Other choices for the readout:
  - Multi-layer Perceptron
  - Support Vector Machine
  - K-Nearest Neighbor
  - ...

ESN – Algorithmic Description: Training

- Initialization
  - `Win = 2*rand(Nr,Nu) - 1; Win = scale_in * Win;`
  - `Wh = 2*rand(Nr,Nr) - 1; Wh = rho * (Wh / max(abs(eig(Wh))));`
  - `state = zeros(Nr,1);`
- Run the reservoir on the input stream
  - ```
    for t = 1:trainingSteps
      state = tanh(Win * u(t) + Wh * state);
      X(:,end+1) = state;
    end
    ```
- Discard the washout
  - `X = X(:,Nwashout+1:end);`
- Train the readout
  - `Wout = Ytarget(:,Nwashout+1:end)*X'*inv(X*X'+lambda_r*eye(Nr));`

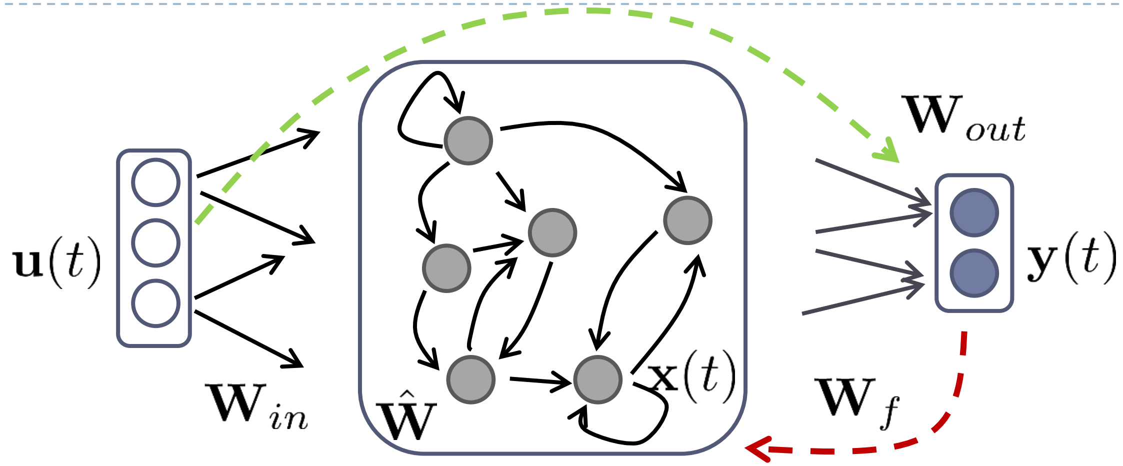
- The ESN is now ready for operation (estimations/predictions)
    ```

ESN – Algorithmic Description: Operation Phase

- Run the reservoir on the input stream (test part)
  - ```
    for t = testSteps
      state = tanh(Win * u(t) + Wh * state);
      output(:,end+1) = Wout * state;
    end
    ```
- Note: when working on a single time-series you do not need to
  - re-initialize the state
  - discard the initial transient
    ```

ESN Hyper-parameterization & Model Selection

Implement ESNs following a good practice for model selection (like for any other ML/NN model)

- Careful selection of network's hyper-parameters
  - reservoir dimension N_R
  - spectral radius ρ
  - input scaling $scale_{in}$
  - readout regularization λ_r
  - …

ESN Major Architectural Variants



- direct connections from the input to the readout

$$\mathbf{y}(t) = \mathbf{W}_{out} \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{u}(t) \end{bmatrix}$$

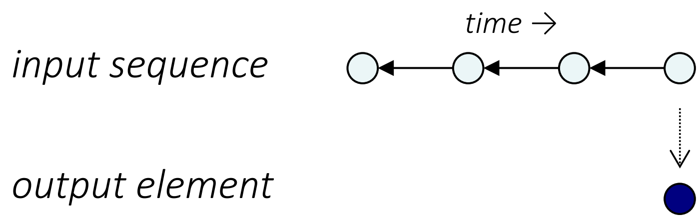
- feedback connections from the output to the reservoir

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

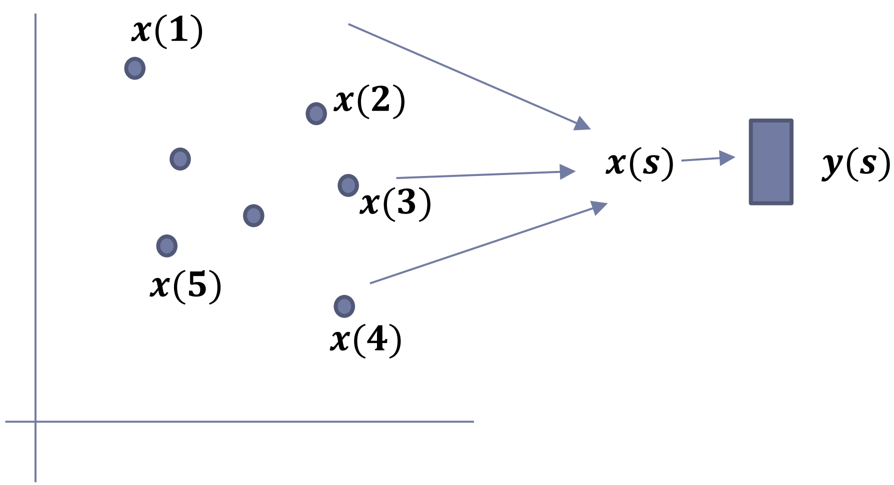
  - might affect the stability of the network's dynamics
  - small values are typically used

ESN for sequence-to-element tasks

- The learning problem requires one single output for each input sequence
- Granularity of the task is on entire sequences (not on time-steps)
  - example: sequence classification

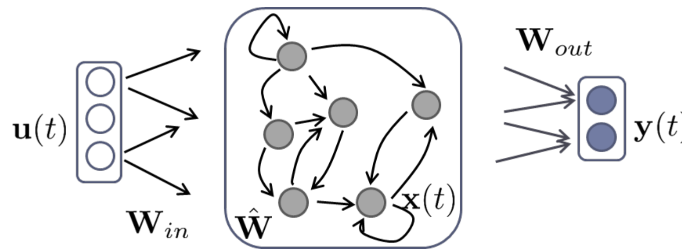


$$\boldsymbol{s} = [\boldsymbol{u}(\mathbf{1}), \ldots, \boldsymbol{u}(\boldsymbol{N_s})]$$



- Last state
  - $\boldsymbol{x}(s) = \boldsymbol{x}(N_s)$
- Mean state
  - $\boldsymbol{x}(s) = {}^{1}\!/_{N_s} \sum_{t=1}^{N_s} \boldsymbol{x}(t)$
- Sum state
  - $\boldsymbol{x}(s) = \sum_{t=1}^{N_s} \boldsymbol{x}(t)$

Leaky Integrator ESN (LI-ESN)



- Use leaky integrator reservoir units

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

- Apply an exponential moving average to reservoir states
  - low-pass filter to better handle input signals that change slowly with respect to the sampling frequency
- the leaking rate parameter $a \in [0,1]$
  - controls the speed of reservoir dynamics in reaction to the input
  - smaller values imply reservoir that react more slowly to the input changes
  - if $a = 1$ then standard ESN dynamics are obtained

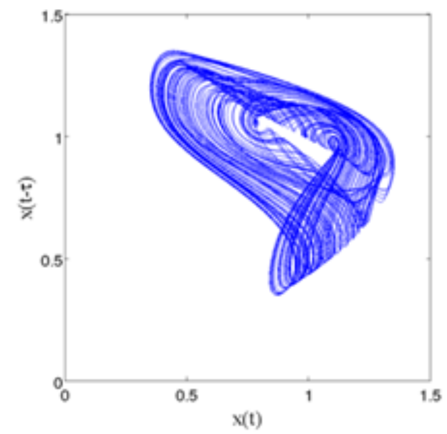
Examples of Applications

Applications of ESNs: Examples /1

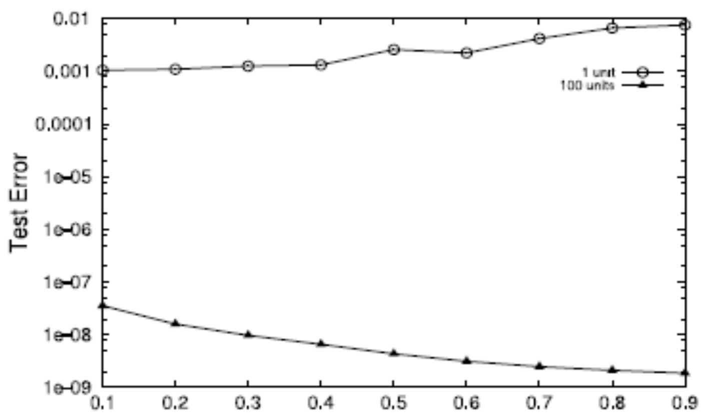
- ESNs for modeling chaotic time series
- Mackey-Glass time series

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

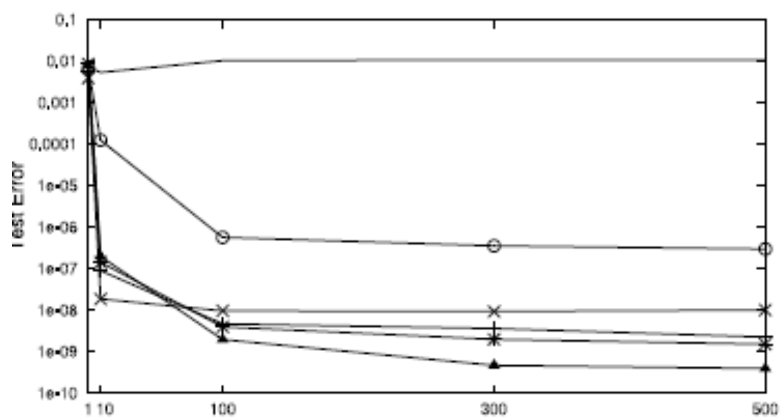
  - for $\alpha > 16.8$ the system has a chaotic attractor
  - most used values are 17 and 30



ESN performance on the MG17 task



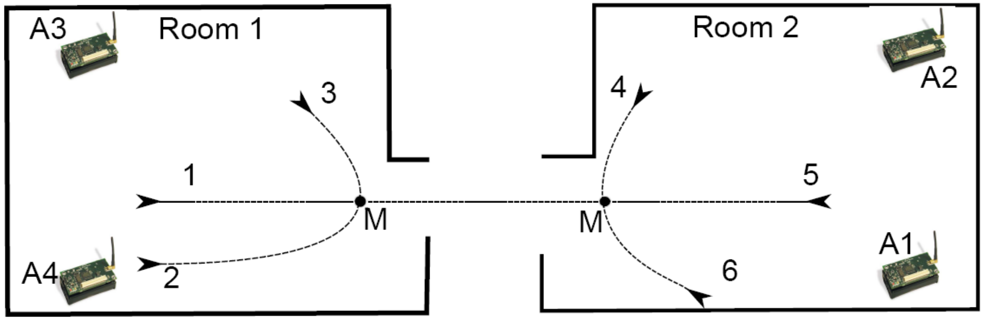
contraction coefficient



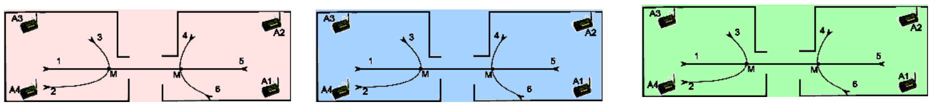
reservoir dimension

Applications of ESNs: Examples /2

- Forecasting of indoor user movements



Generalization of predictive performance to unseen environments



Homogeneous	Heterogeneous
$95.95\% (\pm 3.54)$	$89.52\% (\pm 4.48)$

- ❑ Deployed WSN: 4 fixed sensors (anchors) & 1 sensor worn by the user (mobile)
- ❑ Predict if the user will change room when she is in position M

- ❑ Input: received signal strength (RSS) data from the 4 anchors (10 dimensional vector for each time step, noisy data)
- ❑ Target: binary classification (change environmental context or not)

rubicon

http://fp7rubicon.eu/

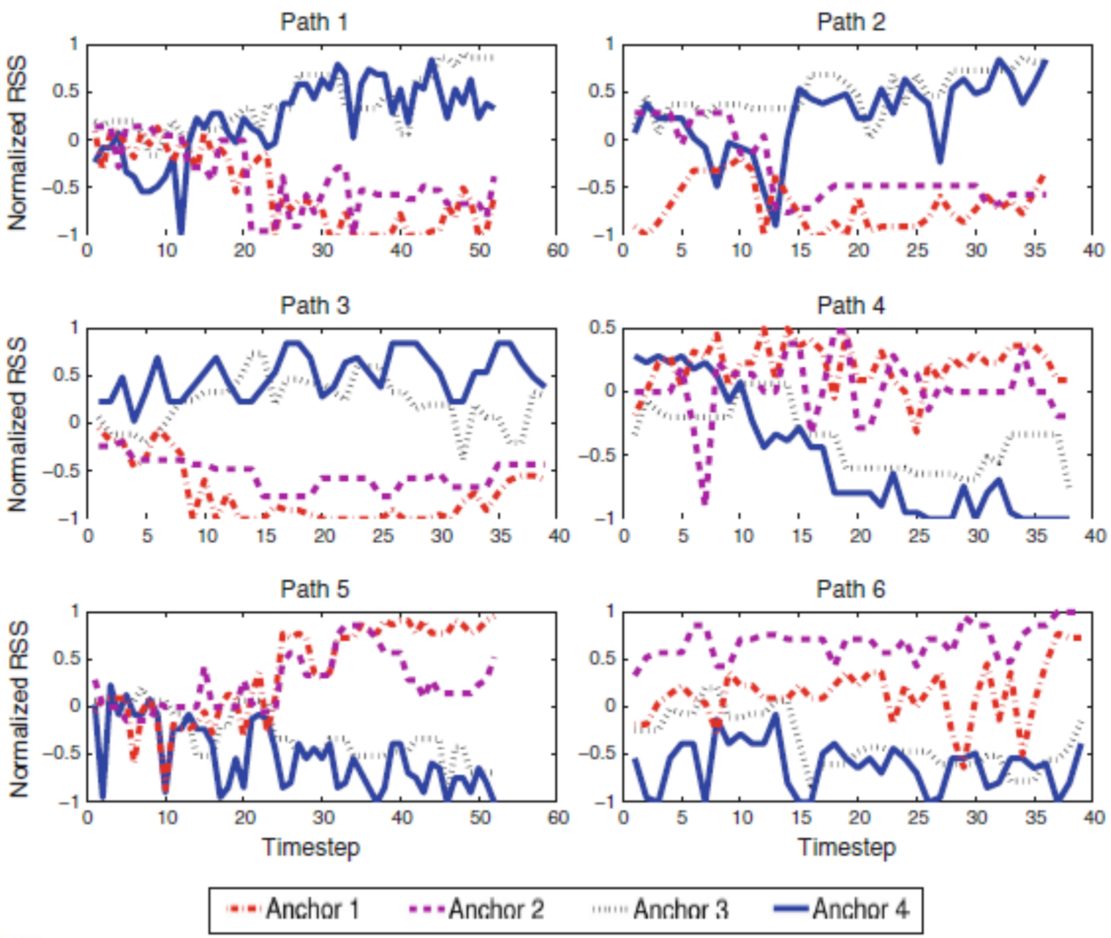
UCI Machine Learning Repository

Dataset is available online on the UCI repository

https://archive.ics.uci.edu/ml/datasets/Indoor+User+Movement+Prediction+from+RSS+data

Applications of ESNs: Examples /2

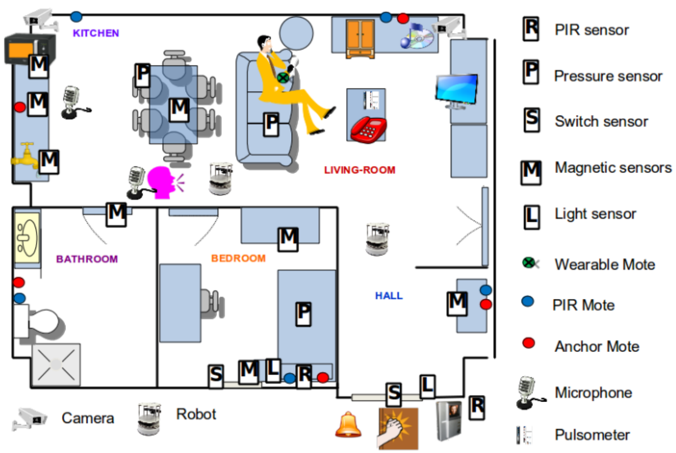
- Forecasting of indoor user movements – Input data



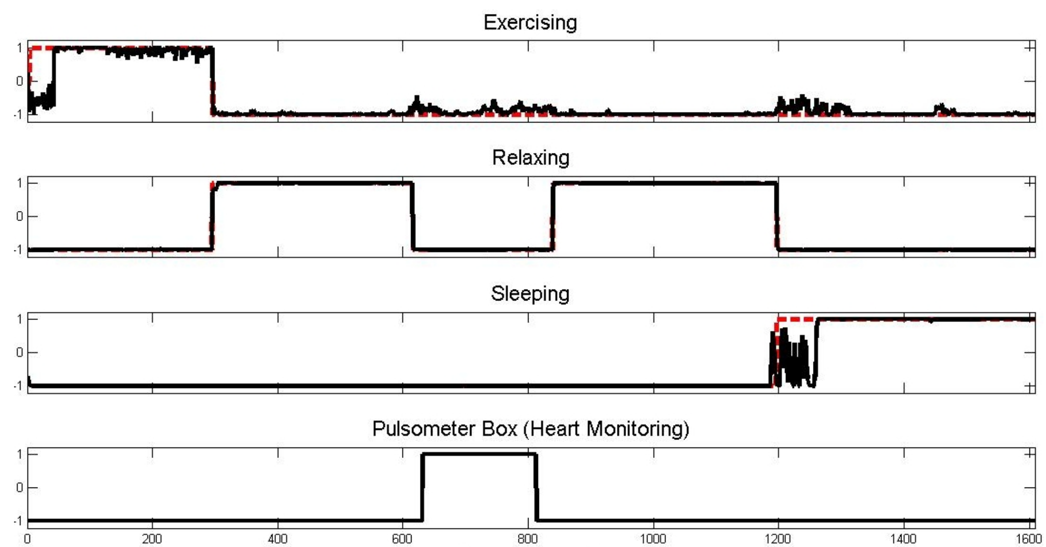
example of the RSS traces gathered from all the 4 anchors in the WSN, for different possible movement paths

Applications of ESNs: Examples /3

- Human Activity Recognition (HAR) and Localization

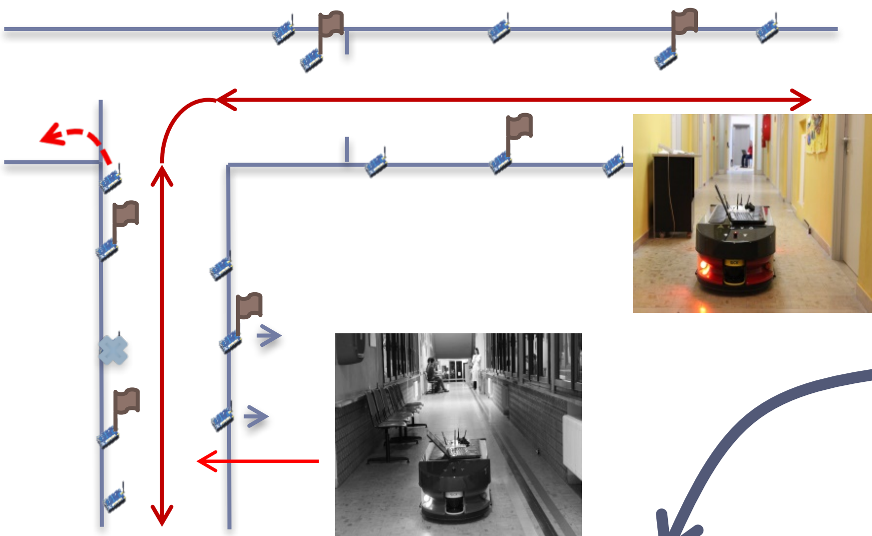


- Input from heterogeneous sensor sources (data fusion)
- Predicting event occurrence and confidence
- High accuracy of event recognition/indoor localization > 90 % on test data
- Effectiveness in learning a variety of HAR tasks
- Effectiveness in training on new events



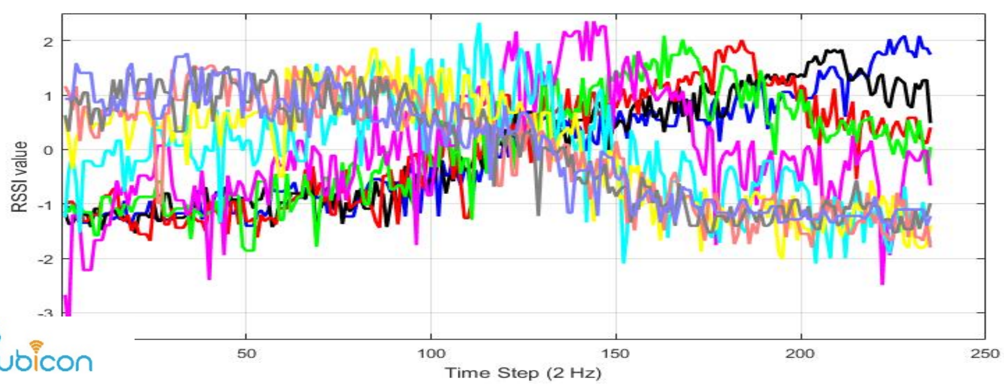
Applications of ESNs: Examples /4

- Robotics



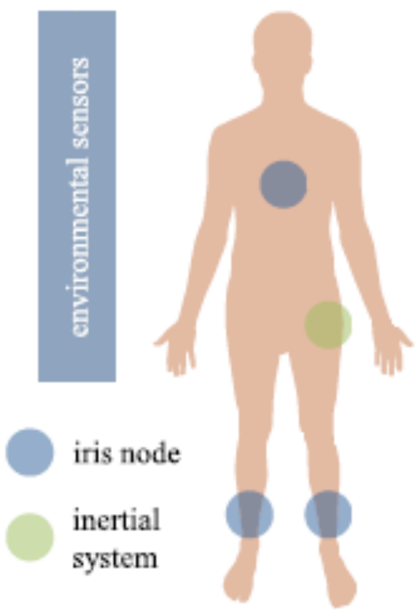
- Indoor localization estimation in critical environment (Stella Maris Hospital)
- Precise robot localization estimation using noisy RSSI data (35 cm)
- Recalibration in case of environmental alterations or sensor malfunctions

- Input: temporal sequences of RSSI values (10 dimensional vector for each time step, noisy data)
- Target: temporal sequences of laser-based localization (x,y)



Applications of ESNs: Examples /7

- Human Activity Recognition



- ❑ Classification of human daily activities from RSS data generated by sensors worn by the user
- ❑ Input: temporal sequences of RSS values (6 dimensional vector for each time step, noisy data)
- ❑ Target: classification of human activity (bending, cycling , lying, sitting, standing, walking)
- ❑ Extremely good accuracy (≈ 0,99) and F1 score (≈ 0,96)
- ❑ 2nd Prize at 2013 EvAAL International Competition

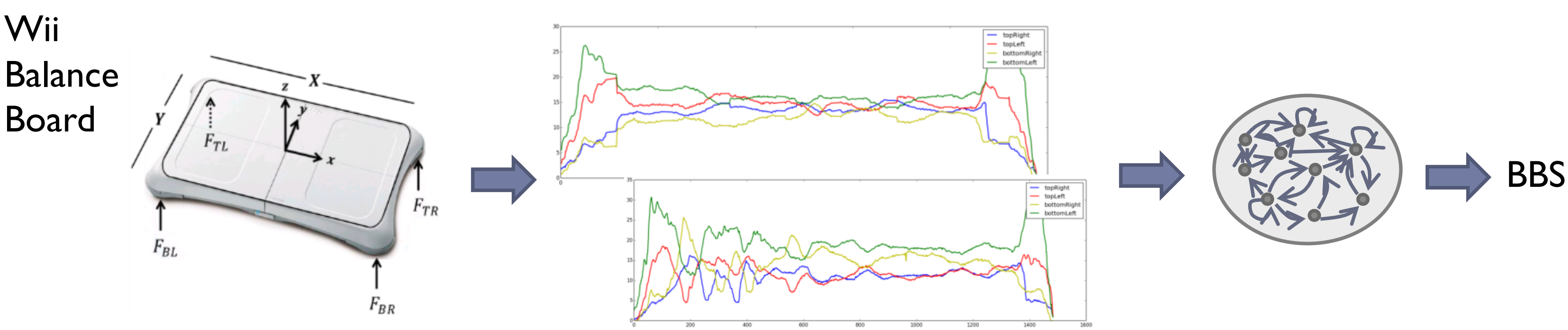
EvAAL

UCI Machine Learning Repository Dataset is available online on the UCI repository

http://archive.ics.uci.edu/ml/datasets/Activity+Recognition+system+based+on+Multisensor+data+fusion+%28AReM%29

Applications of ESNs: Examples /8

- Autonomous Balance Assessment
  - An unobtrusive automatic system for balance assessment in elderly
  - Berg Balance Scale (BBS) test: 14 exercises/items (30 min.)



  - Input: stream of pressure data gathered from the 4 corners Nintendo Wii board during the execution of just 1 (over the 14) BBS exercises
  - Target: global BBS score of the user (0-56)
  - The use of RNNs allow to automatically exploit the richness of the signal dynamics

Applications of ESNs: Examples /8

- Autonomous Balance Assessment
  - Excellent prediction performance using LI-ESNs

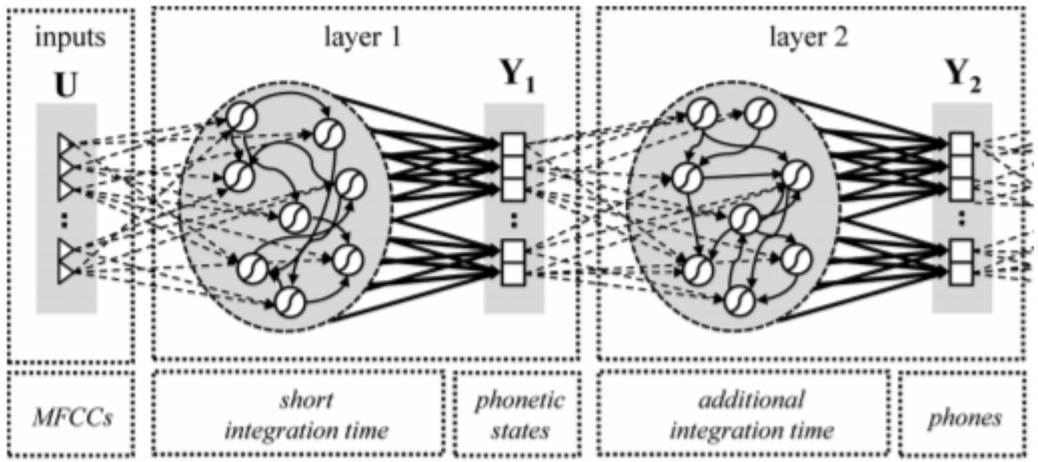
LI-ESN model	Test MAE (BBS points)	Test R
standard	4,80 ± 0,40	0,68
+ weight	4,62 ± 0,30	0,69
LR weight sharing (ws)	4,03 ± 0,13	0,71
ws + weight	**3,80 ± 0,17**	**0,76**

- ❑ Very good comparison
  - ❑ with related models (MLPs, TDNN, RNNs, NARX, …)
  - ❑ with literature approaches

- Practical example of how performance can be improved in a real-world case
  - By an appropriate design of the task
    e.g. inclusion of clinical parameters in input
  - By an appropriate choices for the network design
    e.g. by using a weight sharing approach on the input-to-reservoir connections

Applications of ESNs: Examples /9

- Phones recognition with reservoir networks
  - 2-layered ad-hoc reservoir architecture



  - layers focus on different ranges of frequencies (using appropriate leaky parameters) and focus on different sub-problems

Triefenbach, Fabian, et al. "Phoneme recognition with large hierarchical reservoirs." *Advances in neural information processing systems*. 2010.

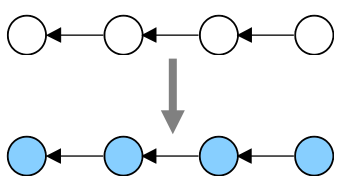
Triefenbach, Fabian, et al. "Acoustic modeling with hierarchical reservoirs." IEEE Transactions on Audio, Speech, and Language Processing 21.11 (2013): 2439-2450.

Extensions to Structured Data

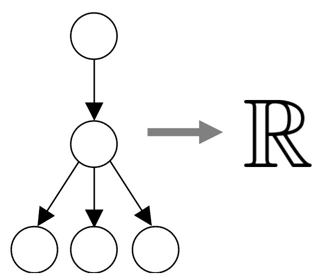
Learning in Structured Domains

- 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.

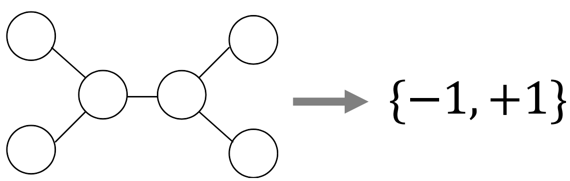
Sequences



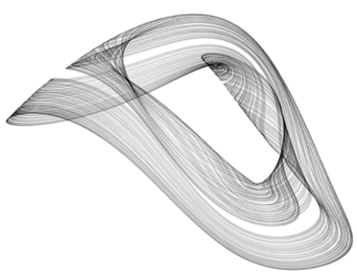
Trees



Graphs

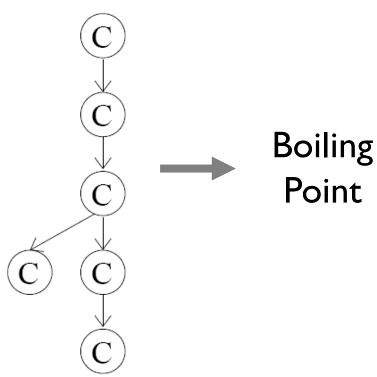


MG -Chaotic Time Series Prediction

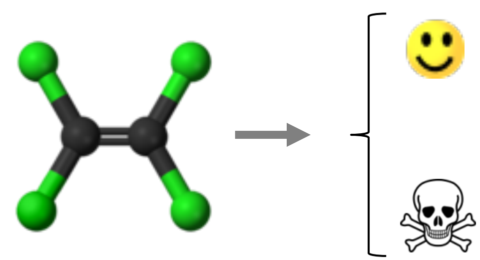


$$\frac{\partial u(t)}{\partial t} = \frac{0.2u(t-\tau)}{1+u(t-\tau)^{10}} - 0.1u(t)\alpha$$

QSPR analysis of Alkanes



Predictive Toxicology Challenge

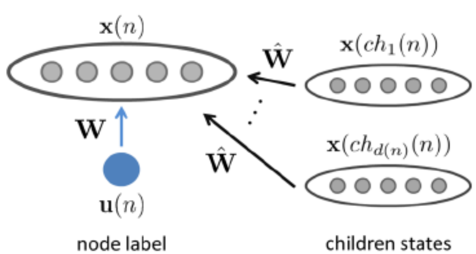


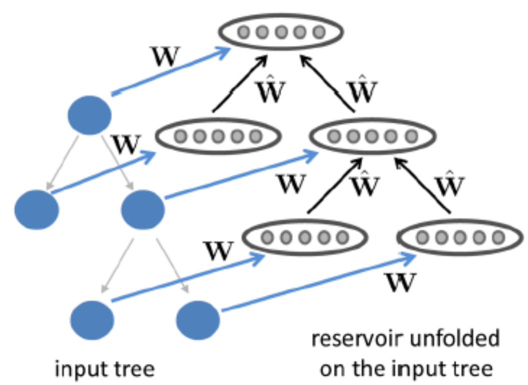
Learning in Structural Domains

- Recursive Neural Networks extend the applicability of RNN methodologies to learning in domains of trees and graphs
- Randomized approaches enable efficient training and state-of-the art performance
- Echo State Networks extended to discrete structures:
  Tree and Graph Echo State Networks

  [C. Gallicchio, A. Micheli, Priceedings of IJCNN 2010] [C. Gallicchio, A. Micheli, Neurocomputing, 2013]

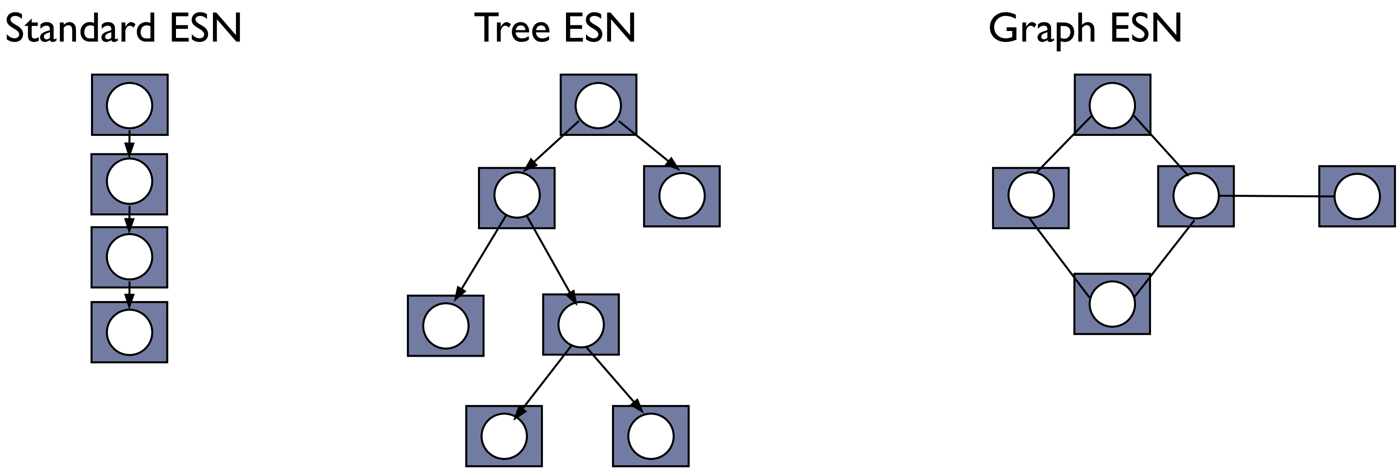
- Basic Idea: the reservoir is applied to each node/vertex of the input structure





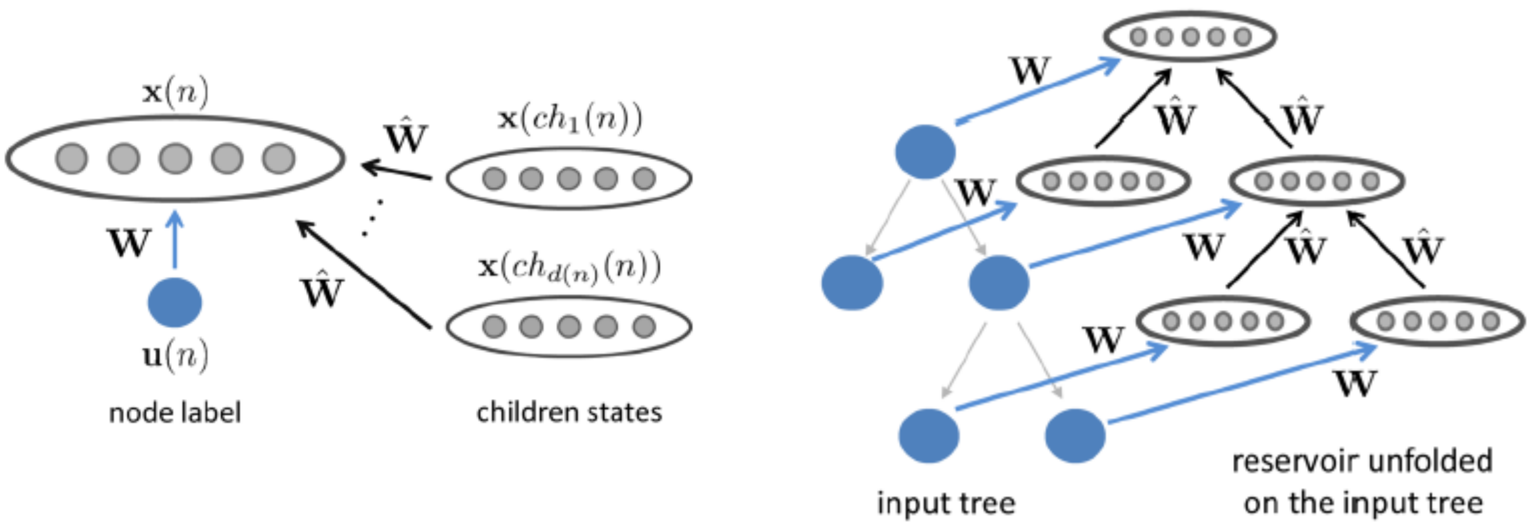
Learning in Structured Domains

- The reservoir operation is generalized from temporal sequences to discrete structures
- State transition systems on discrete tree/graph structures



TreeESN: Reservoir

- Large, sparsely connected, untrained layer of non-linear recursive units
- Input driven dynamical system on discrete tree structures

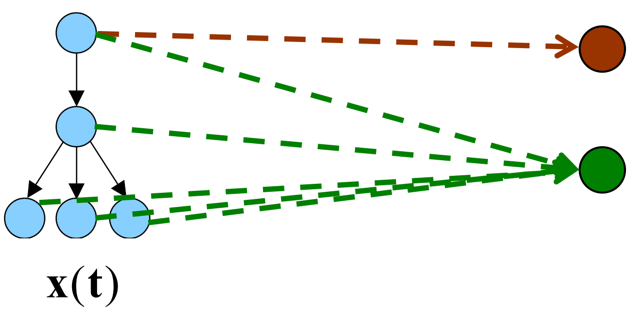


$$\tau : \mathbb{R}^{N_U} \times \mathbb{R}^{k\ N_R} \rightarrow \mathbb{R}^{N_R}$$

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

TreeESN: State mapping and Readout

- State Mapping function for tree-to-element tasks
  - one single output vector (unstructured) is required for each input tree
  - example: document classification



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 computation and training is as in the case of standard Reservoir Computing approaches
  - tree-to-tree (isomorphic transductions) $\mathbf{y}(n) = \mathbf{W}_{out}\mathbf{x}(n)$
  - tree-to-element $\mathbf{y}(\mathbf{t}) = \mathbf{W}_{out}\chi(\mathbf{x}(\mathbf{t}))$

TreeESN: Echo State Property

- The recursive reservoir dynamics can be left untrained provided that a stability property is satisfied
- Tree ESP: asymptotic stability conditions on tree structures

  [C. Gallicchio, A. Micheli, Information Sciences, 2019]

  - for two any initial states, the state computed for the root of the input tree should converge for increasing height of the tree
  - the influence of a perturbation in the label of a node will progressively fade away
- Sufficient condition for the ESP for trees
  - being contractive $\|\hat{\mathbf{W}}\|_2 < 1/k$
- Necessary condition
  - being stable $\rho(\hat{\mathbf{W}}) < 1/k$

degree

TreeESN: Efficiency

Computational Complexity

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

Encoding Process

For each tree **t**

$$O(\underbrace{|N(\mathbf{t})|}_{\text{number of nodes}}\ \underbrace{k}_{\text{max degree}}\ \underbrace{R}_{\text{degree of connectivity}}\ \underbrace{N_R}_{\text{number of reservoir units}})$$

- 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)

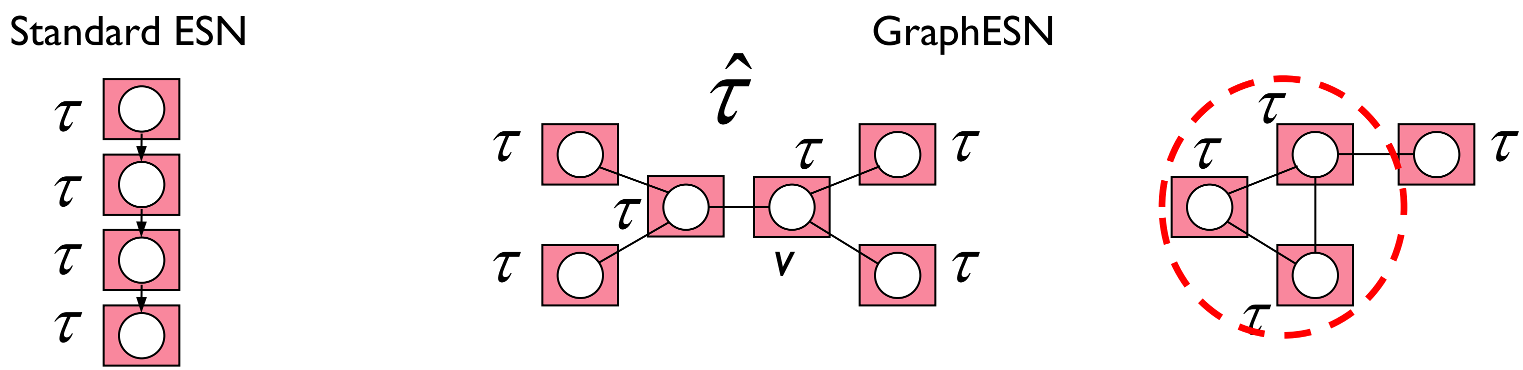
Output Computation

- 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)

Reservoir in Graph Echo State Networks

- Input-driven dynamical system on discrete graphs
- The same reservoir architecture is applied to all the vertices in the structure

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



- Stability of the state update ensures a solution even in case of cyclic dependencies among state variables

GraphESN: Echo State Property

- A stability constraint is required to achieve usable dynamics
- Foundational idea: resort to contractive dynamics
  - Contractive dynamics ensure the convergence of the encoding process (Banach Th.)
  - Enables an iterative computation of the encoding process

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

- Sufficient condition $\|\hat{\mathbf{W}}\|_2 < 1/k$
- Necessary condition $\rho(\hat{\mathbf{W}}) < 1/k$

Research on Reservoir Computing (In our group)

- Applications to complex real-world tasks
  - NLP, earthquake time-series, human monitoring, …
- Gated Reservoir Computing Models
- RC-based analysis of fully trained RNNs
- Unsupervised adaptation of reservoirs
  - edge of stability/chaos
  - Bayesian optimization
- Deep Echo State Networks
  - Advanced mathematical analysis
  - Architectural construction of hierarchical RC
- Deep RC for Structures

Echo State Network Dynamics

Echo State Property

- Assumption: Input and state spaces are compact sets
- A reservoir network whose state update is ruled by

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

  satisfies the ESP if initial conditions are asymptotically forgotten

$$\forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^{N_R} \text{initial states :}$$

$$\forall \mathbf{s} = [\mathbf{u}(1), \ldots, \mathbf{u}(n)] \in (\mathbb{R}^{N_U})^n \text{ input sequence of length } n$$

$$\|\hat{F}(\mathbf{s}, \mathbf{x}) - \hat{F}(\mathbf{s}, \mathbf{x}')\| \rightarrow 0 \quad \text{as } n \rightarrow \infty$$

- The state dynamics provides a pool of "echoes" of the driving input
- Essentially, this is an **stability** condition (global asymptotic stability in the sense of Lyapunov)

Echo State Property: Stability

- *Why a stable regime is so important?*
- An unstable network exhibits sensitivity to input perturbations
  - Two slightly different (long) input sequences drive the network into (asymptotically very) different states
- Good for training
  - The state vectors tend to be more and more linearly separable (for any given task)
- Bad for generalization: overfitting!
  - No generalization ability if a temporal sequence similar to one in the training set drives the network into completely different states

ESP: Sufficient Condition

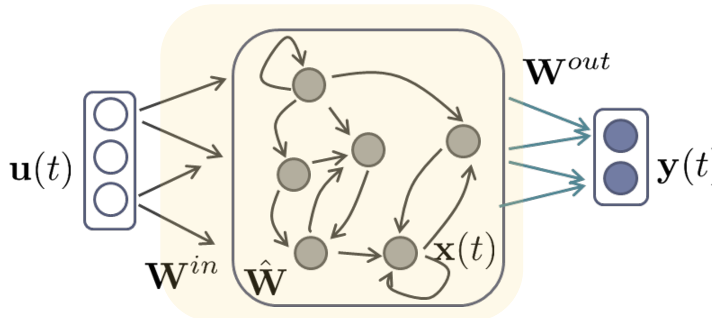
- The sufficient condition for the ESP analyzes the case of *contractive* dynamics of the state transition function
- Whatever is the driving input signal:

  If the system is contractive then it will exhibit stability
- In what follows, we assume state transition functions of the form:

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

$$\underbrace{\mathbf{x}(t)}_{\text{new state}} = tanh(\overbrace{\mathbf{W}_{in}}^{\text{input weight matrix}}\underbrace{\mathbf{u}(t)}_{\text{input}} + \overbrace{\hat{\mathbf{W}}}^{\text{recurrent weight matrix}}\underbrace{\mathbf{x}(t-1)}_{\text{previous state}})$$

Contractivity



The reservoir state transition function rules the evolution of the corresponding dynamical system

$F : \mathbb{R}^{N_U} \times \mathbb{R}^{N_R} \rightarrow \mathbb{R}^{N_R}$ $\quad\quad$ $\mathbf{x}(t) = tanh(\mathbf{W}_{in}\mathbf{u}(t) + \hat{\mathbf{W}}\mathbf{x}(t-1))$

- **Def**. The reservoir has contractive dynamics whenever its state transition function *F* is Lipschitz continuous with constant $C < 1$

$$\exists C \in \mathbb{R}, 0 \leq C < 1, \ \ \forall \mathbf{u} \in \mathbb{R}^{N_U}, \ \ \forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^{N_R} :$$

$$\|F(\mathbf{u}, \mathbf{x}) - F(\mathbf{u}, \mathbf{x}')\| \leq C\|\mathbf{x} - \mathbf{x}'\|$$

Contractivity and the ESP

- **Theorem.** If an ESN has a contractive state transition function *F (and bounded state space)*, then it satisfies the Echo State Property
  - Assumption: *F* is contractive with parameter *C < 1*

- Given this condition:

$$\exists C \in \mathbb{R}, 0 \leq C < 1, \ \ \forall \mathbf{u} \in \mathbb{R}^{N_U}, \ \ \forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^{N_R} :$$
$$\|F(\mathbf{u},\mathbf{x}) - F(\mathbf{u},\mathbf{x}')\| \leq C\|\mathbf{x}-\mathbf{x}'\| \qquad \text{(Contractivity)}$$

- We want to show that the ESP holds true:

$$\forall \mathbf{s} = [\mathbf{u}(1), \ldots, \mathbf{u}(n)] \in (\mathbb{R}^{N_U})^n \quad \text{input sequence of length } n,$$
$$\forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^{N_R} :$$
$$\|\hat{F}(\mathbf{s},\mathbf{x}) - \hat{F}(\mathbf{s},\mathbf{x}')\| \to 0 \ \text{ as } \ n \to \infty$$
$$\text{(ESP)}$$

Contractivity and the ESP

- **Theorem.** If an ESN has a contractive state transition function *F*, then it satisfies the Echo State Property
  - Assumption: *F* is contractive with parameter *C < 1*

$$
\begin{aligned}
&\|\hat{F}([\mathbf{u}(1),\ldots,\mathbf{u}(n)],\mathbf{x}) - \hat{F}([\mathbf{u}(1),\ldots,\mathbf{u}(n)],\mathbf{x}')\| \\
&= \|F(\mathbf{u}(n),\hat{F}([\mathbf{u}(1),\ldots,\mathbf{u}(n-1)],\mathbf{x})) - F(\mathbf{u}(n),\hat{F}([\mathbf{u}(1),\ldots,\mathbf{u}(n-1)],\mathbf{x}'))\| \\
&\leq C\|\hat{F}([\mathbf{u}(1),\ldots,\mathbf{u}(n-1)],\mathbf{x}) - \hat{F}([\mathbf{u}(1),\ldots,\mathbf{u}(n-1)],\mathbf{x}')\|
\end{aligned}
$$

$$\leq \ldots$$

$$
\begin{aligned}
&\leq C^{n-1}\|\hat{F}([\mathbf{u}(1)],\mathbf{x}) - \hat{F}([\mathbf{u}(1)],\mathbf{x}')\| \\
&= C^{n-1}\|F(\mathbf{u}(1),\hat{F}([\,],\mathbf{x})) - F(\mathbf{u}(1),F([\,],\mathbf{x}'))\| \\
&= C^{n-1}\|F(\mathbf{u}(1),\mathbf{x}) - F(\mathbf{u}(1),\mathbf{x}')\| \\
&\leq C^{n}\|\mathbf{x}-\mathbf{x}'\|
\end{aligned}
$$

goes to 0 as *n* goes to infinity → the ESP holds

Contractivity and Reservoir Initialization

- If the reservoir is initialized to implement a contractive mapping than the ESP is guaranteed (in any norm, for any input)
- Formulation of a sufficient condition for the ESP $\sigma(\hat{\mathbf{W}}) = \|\hat{\mathbf{W}}\|_2 < 1$
  - Assumptions:
    - Euclidean distance as metric in the state space (use L2-norm)
    - Reservoir units with *tanh* activation function (note: squashing nonlinearities bound the state space)

$$\|F(\mathbf{u},\mathbf{x}) - F(\mathbf{u},\mathbf{x}')\|_2$$

$$= \|tanh(\mathbf{W}_{in}\mathbf{u} + \hat{\mathbf{W}}\mathbf{x}) - tanh(\mathbf{W}_{in}\mathbf{u} + \hat{\mathbf{W}}\mathbf{x}')\|_2$$

$$\leq max(|tanh'|)\|\hat{\mathbf{W}}(\mathbf{x} - \mathbf{x}')\|_2$$

$$\leq \|\hat{\mathbf{W}}\|_2\|\mathbf{x} - \mathbf{x}'\|_2$$

$$\|\hat{\mathbf{W}}\|_2 < 1 \Rightarrow F \text{ is contractive} \Rightarrow \text{the ESP holds}$$

Markovian Nature of state space organizations

- Contractive dynamical systems are related to suffix-based state space organizations
- States assumed in correspondence of different input sequences sharing a common suffix are close to each other proportionally to the length of the common suffix
  - similar sequences are mapped to close states
  - different sequences are mapped to different states
  - similarities and dissimilarities are intended in a suffix-based fashion
- RNNs initialized with small weights (with contractive state transition function) and bounded state space implement (approximate arbitrarily well) definite memory machines

Hammer, B., Tino, P.: Recurrent neural networks with small weights implement definite memory machines. Neural Computation 15 (2003) 1897-1929

Markovian Nature of state space organizations

- Markovian Architectural bias of RNNs
  - recurrent weights are typically initialized with small values
  - this leads to a typically contractive initialization of recurrent dynamics
  - Iterated Function Systems, fractal theory, *architectural bias* of RNNs
  - RNNs initialized with small weights (with contractive state transition function) and bounded state space implement (approximate arbitrarily well) definite memory machines

    Hammer, B., Tino, P.: Recurrent neural networks with small weights implement definite memory machines. Neural Computation 15 (2003) 1897-1929
  - This characterization is a *bias* for fully trained RNNs: holds in the early stages of learning

Markovianity and ESNs

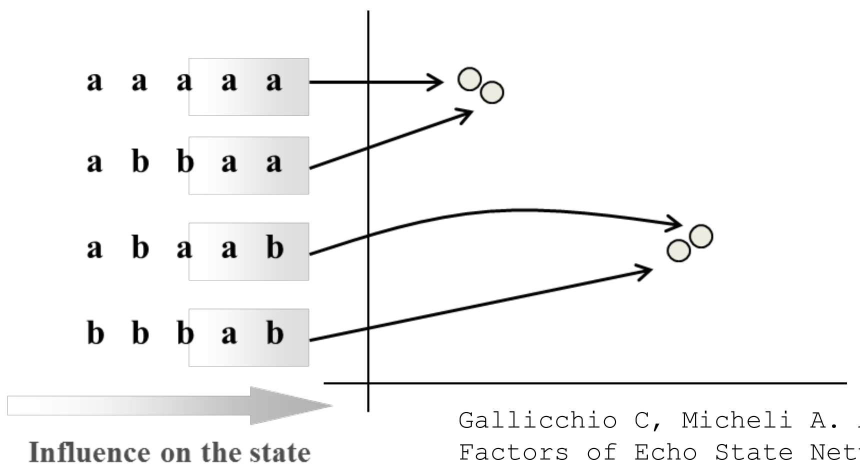
- Using dynamical systems with contractive state transition functions (in any norm) implies the Echo State Property (for any input)
- ESNs featured by fixed contractive dynamics
  - Relations with the universality of RC for bounded memory computation (LSMs theory)

    Maass, W., Natschlager, T., Markram, H.: Real-time computing without stable states: A new framework for neural computation based on perturbations. Neural Computation 14 (2002) 2531-2560
  - **ESNs with untrained contractive reservoirs are already able to distinguish input sequences on a suffix-based fashion**
  - In the RC framework this is no longer a bias, it is a fixed characterization of the RNN model

    Gallicchio C, Micheli A. Architectural and Markovian Factors of Echo State Networks. Neural Networks 2011;24(5):440–456.

Why do Echo State Networks work?

- Because they exploit the Markovian state space organization
- The reservoir constructs a high-dimensional Markovian state space representation of the input history
- Input sequences sharing a common suffix drive the system into close states
  - The states are close to each other proportionally to the length of the common suffix
  - A simple output (readout) tool can then be sufficient to separate the different cases

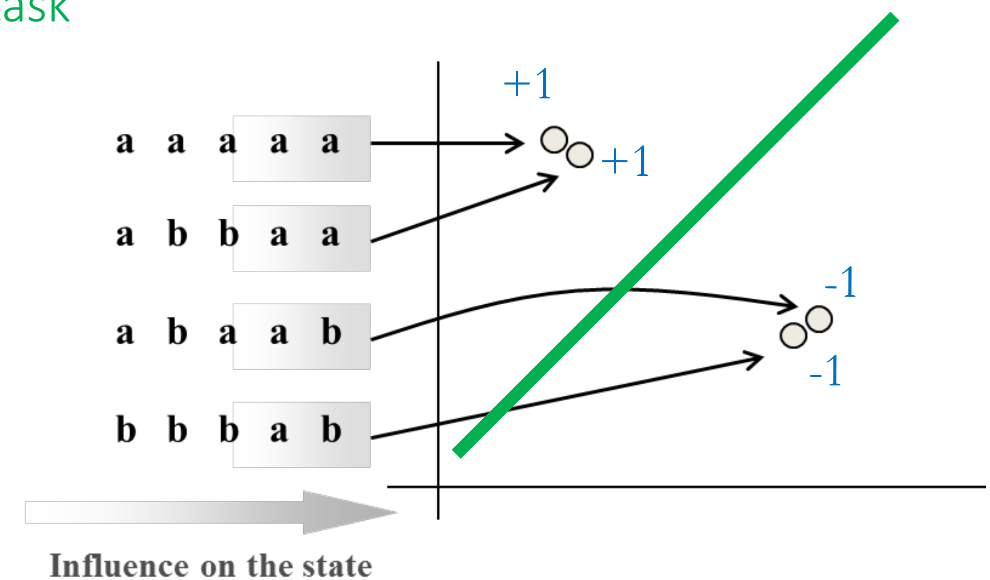


Gallicchio C, Micheli A. Architectural and Markovian Factors of Echo State Networks. Neural Networks 2011;24(5):440-456.

When do Echo State Networks work?

- When the target matches the Markovian assumption behind the reservoir state space organization
- Markovianity can be used to characterize easy/hard tasks for ESNs

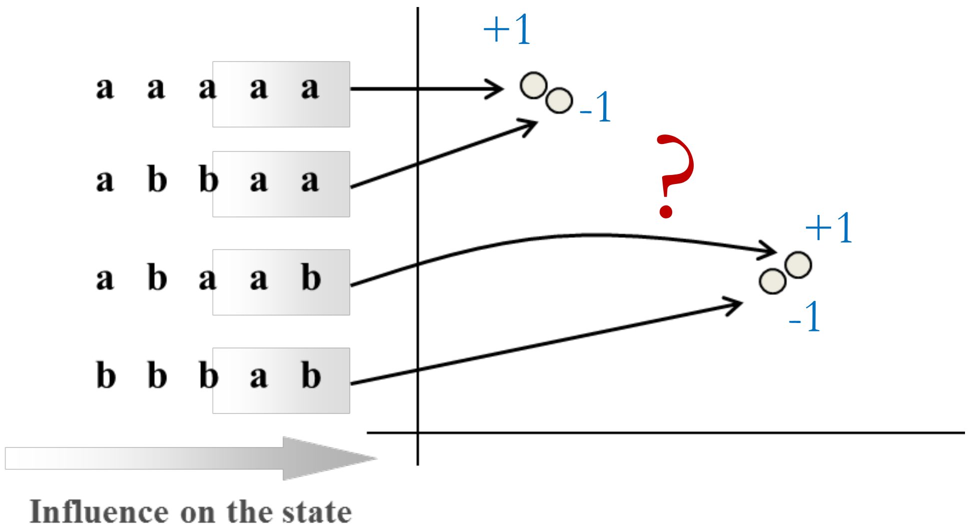
Example: easy task



When do Echo State Networks work?

- When the target matches the Markovian assumption behind the reservoir state space organization
- Markovianity can be used to characterize easy/hard tasks for ESNs

Example: hard task



ESP: Necessary Condition

- Investigating the stability of reservoir dynamics from a dynamical system perspective
- **Theorem.** If an ESN has unstable dynamics around the zero state and the zero sequence is an admissible input, then the ESP is not satisfied.
- Approach this study by linearizing the state transition function

$$\mathbf{x}(t) = \mathbf{J}_F(\mathbf{u}(t), \mathbf{x}_0)(\mathbf{x}(t-1) - \mathbf{x}_0) + F(\mathbf{u}(t), \mathbf{x}_0)$$

$$\mathbf{J}_{F,\mathbf{x}}(\mathbf{u}(t), \mathbf{x}_0) = \begin{pmatrix} \mathbf{J}_{F^{(1)},\mathbf{x}^{(1)}}(\mathbf{u}(t), \mathbf{x}_0) & \mathbf{J}_{F^{(1)},\mathbf{x}^{(2)}}(\mathbf{u}(t), \mathbf{x}_0) & \dots & \mathbf{J}_{F^{(1)},\mathbf{x}^{(N_L)}}(\mathbf{u}(t), \mathbf{x}_0) \\ \mathbf{J}_{F^{(2)},\mathbf{x}^{(1)}}(\mathbf{u}(t), \mathbf{x}_0) & \mathbf{J}_{F^{(2)},\mathbf{x}^{(2)}}(\mathbf{u}(t), \mathbf{x}_0) & \dots & \mathbf{J}_{F^{(2)},\mathbf{x}^{(N_L)}}(\mathbf{u}(t), \mathbf{x}_0) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{J}_{F^{(N_L)},\mathbf{x}^{(1)}}(\mathbf{u}(t), \mathbf{x}_0) & \mathbf{J}_{F^{(N_L)},\mathbf{x}^{(2)}}(\mathbf{u}(t), \mathbf{x}_0) & \dots & \mathbf{J}_{F^{(N_L)},\mathbf{x}^{(N_L)}}(\mathbf{u}(t), \mathbf{x}_0) \end{pmatrix}$$

Jacobian matrix

ESP: Necessary Condition

- Linearization around the zero state and for null input

$$\mathbf{x}(t) = \mathbf{J}_F(\mathbf{0}, \mathbf{0})\mathbf{x}(t-1)$$

- Remember:

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

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

- The Jacobian with *tanh* neurons is given by

$$\mathbf{J}_F = \begin{bmatrix} 1 - x_1(t-1)^2 & 0 & \dots & 0 \\ 0 & 1 - x_2(t-1)^2 & \dots & 0 \\ \dots 0 & 0 & \dots & 1 - x_{N_R}(t-1)^2 \end{bmatrix} \hat{\mathbf{W}}$$

ESP: Necessary Condition

- Linearization around the zero state and for null input

$$\mathbf{x}(t) = \mathbf{J}_F(\mathbf{0}, \mathbf{0})\mathbf{x}(t-1)$$

- Remember:

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

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

- The Jacobian with *tanh* neurons is given by

$$\mathbf{J}_F = \begin{bmatrix} 1 - \cancel{x_1(t-1)^2} & 0 & \dots & 0 \\ 0 & 1 - \cancel{x_2(t-1)^2} & \dots & 0 \\ \dots 0 & 0 & \dots & 1 - \cancel{x_{N_R}(t-1)^2} \end{bmatrix} \hat{\mathbf{W}}$$

Null input assumption

ESP: Necessary Condition

- The linearized system now reads:

$$\mathbf{x}(t) = \hat{\mathbf{W}}\mathbf{x}(t-1)$$

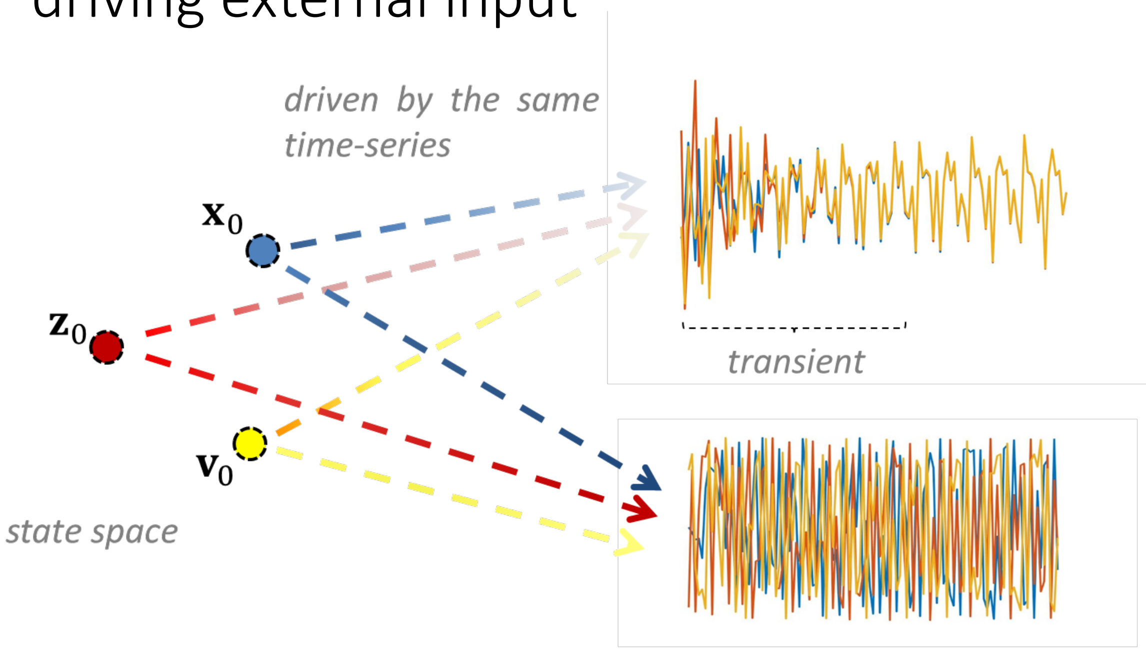
- 0 is a fixed point. Is it stable?
- Linear dynamical systems theory tells us that
  If $\rho(\widehat{\mathbf{W}}) < 1$ then the fixed point is stable
- Otherwise: **0** is not stable
  if we start from a state near **0** and we drive the network with a (infinite-length) null sequence we do not end up in **0**
- The null sequence is a counter-example: the ESP does not hold!
  - There are at least two different orbits resulting from the same input sequence

ESP: Necessary Condition

- A sufficient condition (under our assumptions) for the absence of the ESP is that $\rho(\hat{\mathbf{W}}) \geq 1$
- Hence, a necessary condition for the ESP is that $\rho(\hat{\mathbf{W}}) < 1$
- In general: $\rho(\hat{\mathbf{W}}) \leq \|\hat{\mathbf{W}}\|_2$
- Typically, in applications:
  - The sufficient condition is often too restrictive
  - The necessary condition for the ESP is often used to scale the recurrent weight matrix (e.g. to a value of the spectral radius of 0.9)
- However, note that scaling the spectral radius below 1 *in presence of driving input* is neither sufficient nor necessary for ensuring echo states

ESP Index: a concrete perspective

- Driving input is not properly taken into account in conventional reservoir initialization strategies
- Idea: calculate average deviations of reservoir state orbits from different initial conditions and under the influence of the same driving external input



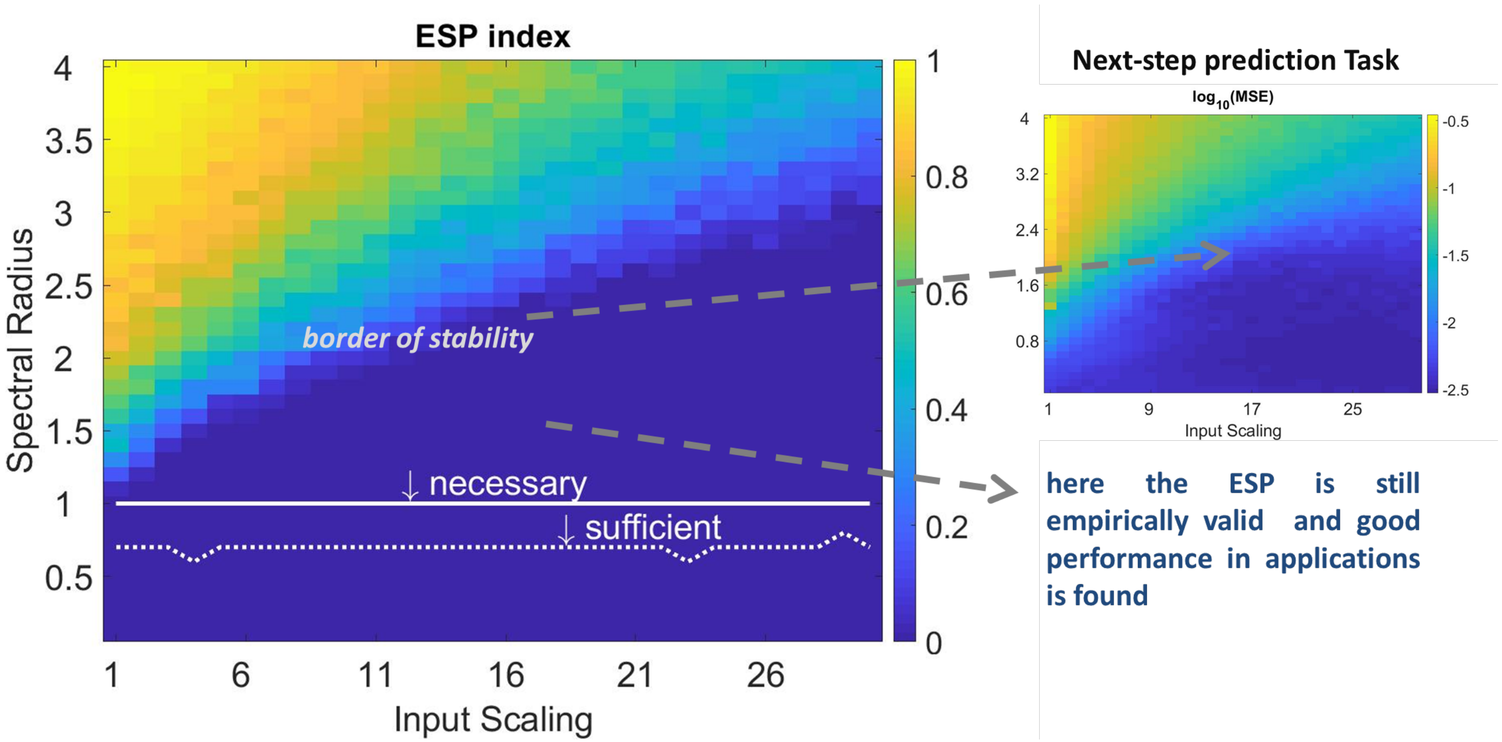
stable dynamics

- all orbits synchronize
- unique attracting input-dependent orbit
- **the ESP is empirically satisfied**

unstable dynamics

- orbits are sensitive to initial conditions
- dynamics are prone to overfitting

ESP Index: a concrete perspective



C. Gallicchio, "Chasing the Echo State Property", ESANN 2019.

- The set of configurations that satisfy the ESP in real cases are well beyond those commonly adopted in ESN practice
- A large portion of "good" reservoirs are usually neglected in common practice

Advances on Echo State Networks

Research on Echo State Networks

- The research on ESNs follows two major complementary objectives:
  - Study the intrinsic properties of RNNs, taking aside the aspects related to training of the recurrent connections
  - Develop efficiently trained RNNs
- Advances…
  - Theoretical analysis
  - Quality of reservoir dynamics
  - Architectural studies
  - Deep Echo State Networks
  - Reservoir Computing for Structures

Echo State Property: Advances

- Much of the theoretical advances in the study of ESNs aim to establish simple conditions for reservoir initialization
- Focus of the theoretical investigations is shifted to the study of stability constraint
- Analysis of the system stability *given the input*

  I.B. Yildiz, H. Jaeger, and S.J. Kiebel. Re-visiting the echo state property. Neural networks, 35:1-9, 2012.

  - Non-autonomous dynamical systems

    G. Manjunath and H. Jaeger. Echo state property linked to an input: Exploring a fundamental characteristic of recurrent neural networks. Neural computation, 25(3):671-696, 2013.

  - Mean Field Theory

    M. Massar and S. Massar. Mean-field theory of echo state networks. Physical Review E, 87(4):042809, 2013.

  - Local Lyapunov exponents

    G. Wainrib and M.N. Galtier. A local echo state property through the largest lyapunov exponent. Neural Networks, 76:39-45, 2016.

Applications to Real-world Problems

- Successful applications in several real-world problems
  - Chaotic time-series modeling
  - Non-linear system identification
  - Speech recognition
  - Financial forecasting
  - Bio-medical applications
  - Robot localization & control
  - …
- High dimensional reservoirs are often needed to achieve excellent performance in complex real-world tasks
- Question: can we combine training efficiency with compact (i.e. small size) ESNs?

Quality of Reservoir Dynamics

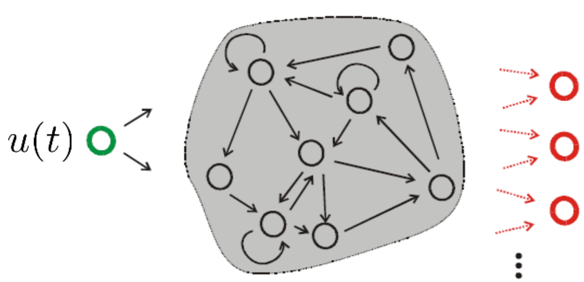
- How to establish the quality of a reservoir?
- If I can find a suitable way to characterize how good a reservoir is I can try to optimize it
- Entropy of recurrent units activations
  - Unsupervised adaptation of reservoirs using Intrinsic Plasticity
- Study the short-term memory ability of the system
  - Memory Capacity and relations to linearity
- Edge of stability/chaos: reservoir dynamics at the border of stability
  - Recurrent systems close to instability show optimal performances whenever the task at hand requires long short-term memory

Short-term Memory Capacity

- An aspect of great importance in the study of dynamical systems is the analysis of their memory abilities
- Jaeger introduced a learning task, called Memory Capacity (MC) to quantify it

  Jaeger, Herbert. *Short term memory in echo state networks*. Vol. 5. GMD-Forschungszentrum Informationstechnik, 2001.

- Train individual output units to recall increasingly delayed versions of a univariate i.i.d. input signal



$y_1(t) = u(t-1)$

$y_2(t) = u(t-2)$

$y_3(t) = u(t-3)$

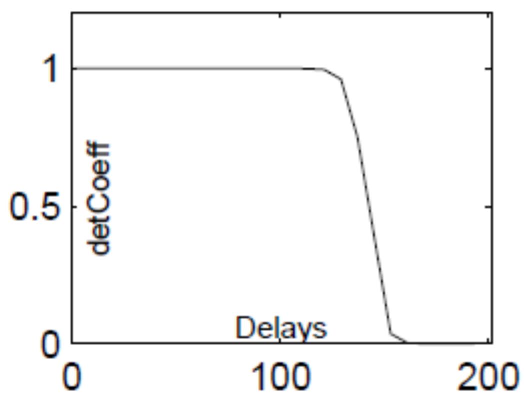
$\vdots$

$$MC = \sum_{k=1}^{\infty} r^2(u(t-k), y_k(t))$$

MC is the sum of squared correlation coefficients between the delayed signals and the outputs

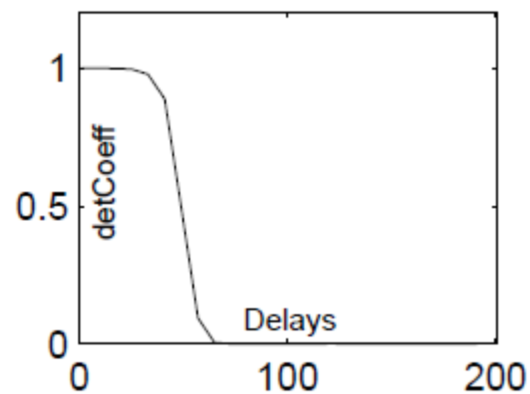
Short-term Memory Capacity

- Forgetting curves to study the memory structure
- Plot the squared correlation (Y-axis, i.e. *detCoeff*) with respect to individual delays (X-axis)



linear reservoir units

B



tanh reservoir units

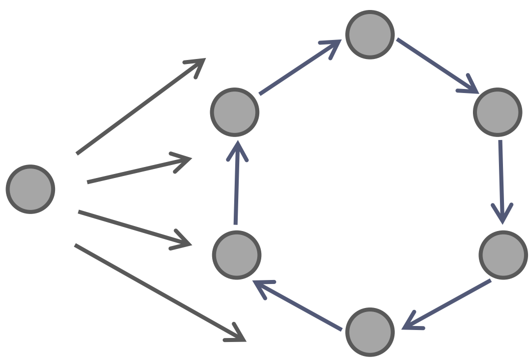
Short-term Memory Capacity

Some fundamental theoretical results:

- The MC of a network with N recurrent units is upper-bounded by N
  - MC ≤ N
  - It is impossible to train an ESN on tasks which require unbounded-time memory
- Linear reservoirs can achieve the maximum bound
  - e.g. sufficient condition: the matrix $\hat{\mathbf{W}}^1\mathbf{w}_{in}\hat{\mathbf{W}}^2\mathbf{w}_{in} \dots \hat{\mathbf{W}}^N\mathbf{w}_{in}$ has full rank
  - example: unitary recurrent matrices (i.e. **orthogonal matrices** in the real case)
- Memory versus Non-linearity dilemma
  - Linear reservoirs are featured by longer short-term memories, but non-linear reservoirs are required to solve complex real-world problems....
- Memory Capacity vs Predictive Capacity

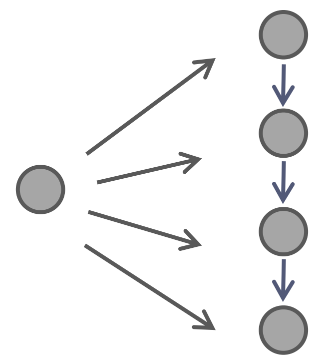
Architectural Setup

- How to construct "better" reservoirs than just random reservoirs?
- Critical Echo State Networks: relevance of orthogonal recurrence weight matrices (e.g. permutation matrices)
  [M.A. Hajnal, A. Lorincz, 2006]



Cyclic Reservoirs

[A. Rodan, P. Tino, 2011]
[T. Strauss et al., 2012]
[J. Boedecker et al., 2009]

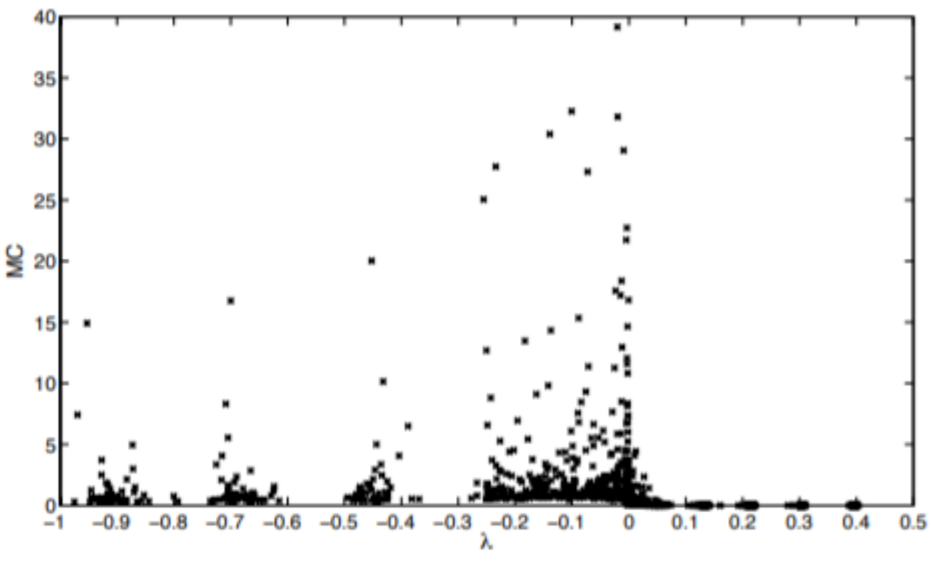


Delay Line Reservoirs

[M. Cernansky, P. Tino 2008]
[A. Rodan, P. Tino 2012]

Edge of Stability

- Reservoir dynamical regime close to the transition between stable and unstable dynamics
  - E.g., study of Lyapunov exponents
  - $\lambda = 0$ identifies the transition between (locally) stable ($\lambda < 0$) and unstable ($\lambda > 0$) dynamics



[J. Boedecker, O. Obst, J.T. Lizier, N.M. Mayer, and M. Asada. Information processing in echo state networks at the edge of chaos. Theory in Biosciences, 131(3):205–213, 2012.]

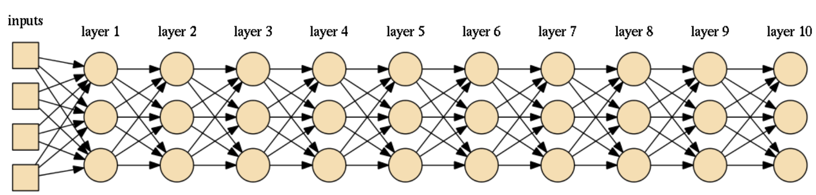
Deep Neural Networks

Deep Learning is an attractive research area

- Hierarchy of many non-linear models
- Ability to learn data representations at different (higher) levels of abstraction

Deep Neural Networks (DeepNNs)

- Feed-forward hierarchy of multiple hidden layers of non-linear units



- Impressive **performance** in real-world problems (especially in the cognitive area)
- Remember: deep learning has a strong biological plausibility

Deep Recurrent Neural Networks

Extension of the deep learning methodology to temporal processing.

Aim at naturally capture temporal feature representations at different time-scales

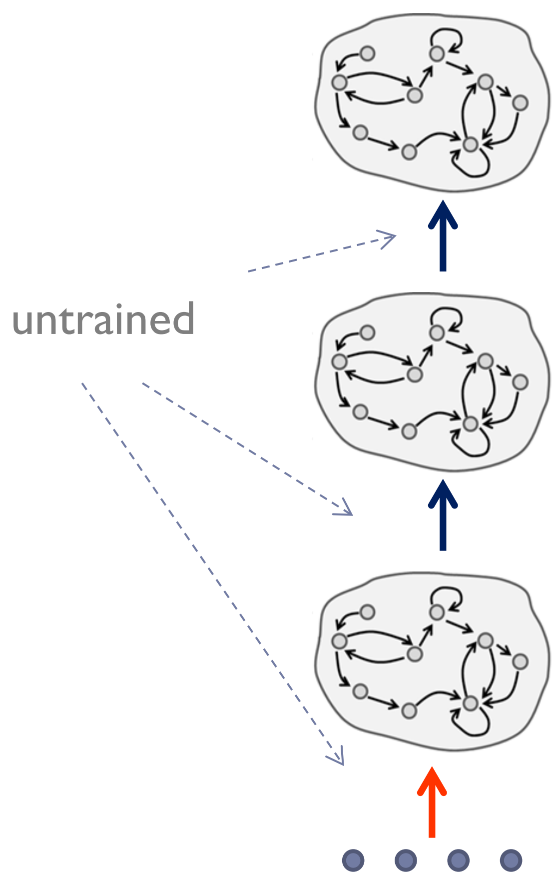
- Text, speech, language processing
- Multi-layered processing of temporal information with feedbacks has a strong biological plausibility

The analysis of deep RNNs is still young

- **Deep Reservoir Computing: Investigate the actual role of layering in deep recurrent architectures**
- Stability: characterize the dynamics of hierarchically organized recurrent models

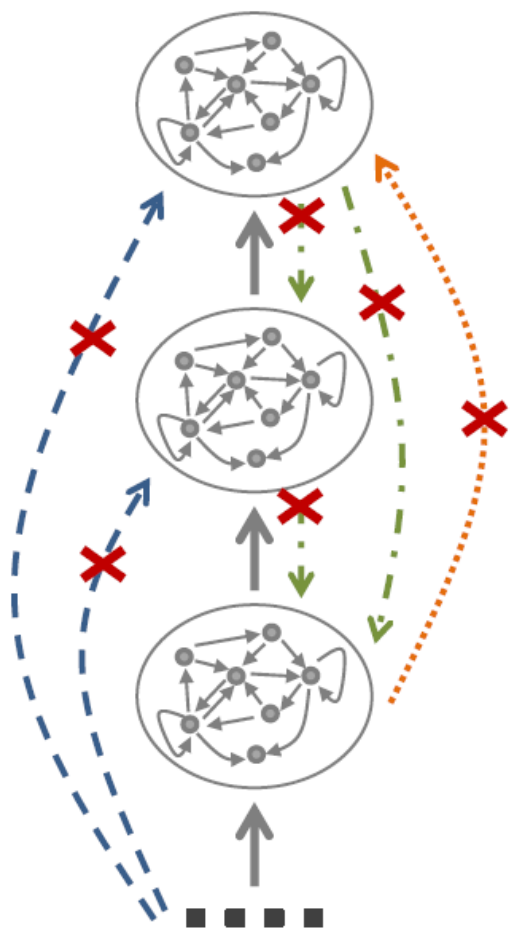
Deep Echo State Network

C. Gallicchio, A. Micheli, L. Pedrelli, "Deep Reservoir Computing: A Critical Experimental Analysis", Neurocomputing, 2017



- What is the **intrinsic role** of **layering** in recurrent architectures?
- Develop **novel efficient approaches** to exploit
  - **Multiple time-scales** representations
  - Extreme efficiency of training

Deep RNN Architecture: The role of Layering

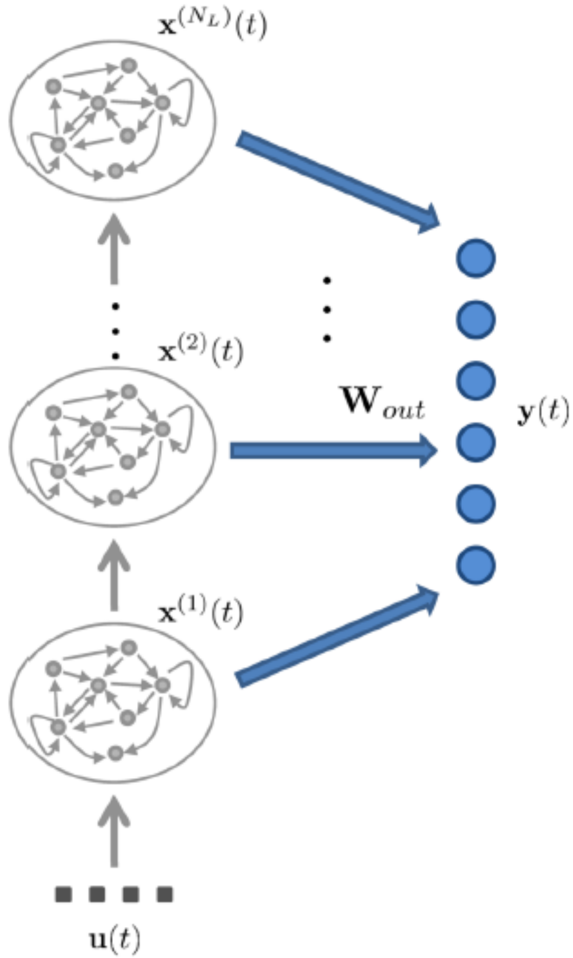


Constraints to the architecture of a fully connected RNN, by:

- Removing connection from input to higher layers
- Removing connections from higher layers to lower ones
- Removing connections to layers at levels higher than +1

Less weights to store than a fully connected RNN

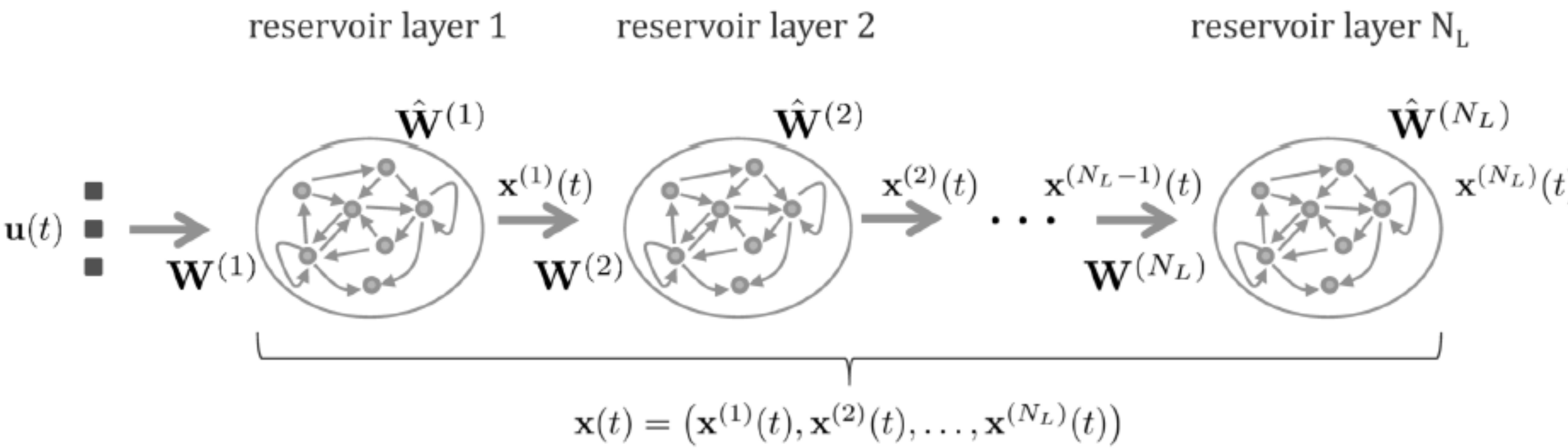
DeepESN: Output Computation



$$\mathbf{y}(t) = \mathbf{W}_{out}\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(N_L)}\right)$$

- The readout can modulate the (qualitatively different) temporal features developed at the different layers

DeepESN: Architecture and Dynamics



first layer

$$\begin{aligned}\mathbf{x}^{(1)}(t) &= F(\mathbf{u}(t), \mathbf{x}^{(1)}(t-1)) \\ &= (1-a^{(1)})\mathbf{x}^{(1)}(t-1) + \mathbf{f}(\mathbf{W}^{(1)}\mathbf{u}(t) + \\ &\qquad \hat{\mathbf{W}}^{(1)}\mathbf{x}^{(1)}(t-1)),\end{aligned}$$

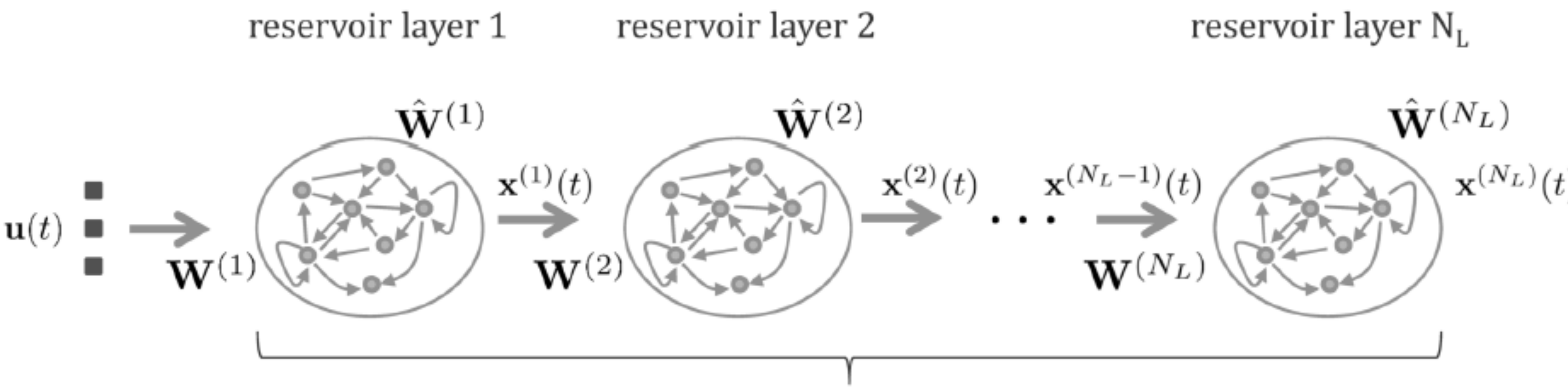
l-th layer (l>1)

$$\begin{aligned}\mathbf{x}^{(l)}(t) &= F(\mathbf{x}^{(l-1)}(t), \mathbf{x}^{(l)}(t-1)) \\ &= (1-a^{(l)})\mathbf{x}^{(l)}(t-1) + \mathbf{f}(\mathbf{W}^{(l)}\mathbf{x}^{(l-1)}(t) + \\ &\qquad \hat{\mathbf{W}}^{(l)}\mathbf{x}^{(l)}(t-1)).\end{aligned}$$

Each layer has its own:
- leaky integration constant
- Input scaling
- Spectral radius
- Inter-layer scaling

* seminar topic

DeepESN: Architecture and Dynamics



$$\mathbf{x}(t) = \left(\mathbf{x}^{(1)}(t), \mathbf{x}^{(2)}(t), \ldots, \mathbf{x}^{(N_L)}(t)\right)$$

The recurrent part of the system is hierarchically structured. Interestingly, this naturally entails a structure into the developed system dynamics

$$\hat{\mathbf{W}}^{(1)}\mathbf{x}^{(1)}(t-1)),$$

own:

- leaky integration constant
- Input scaling
- Spectral radius
- Inter-layer scaling

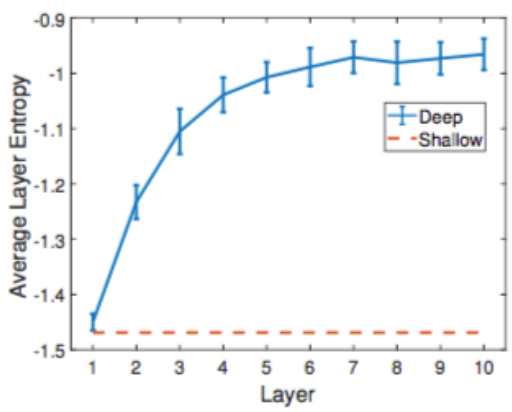
l-th layer (l>1)

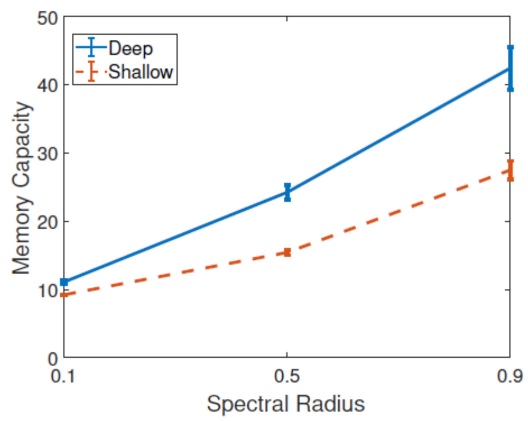
$$\begin{aligned}\mathbf{x}^{(l)}(t) &= F(\mathbf{x}^{(l-1)}(t), \mathbf{x}^{(l)}(t-1)) \\ &= (1-a^{(l)})\mathbf{x}^{(l)}(t-1) + \mathbf{f}(\mathbf{W}^{(l)}\mathbf{x}^{(l-1)}(t) + \\ &\qquad \hat{\mathbf{W}}^{(l)}\mathbf{x}^{(l)}(t-1)).\end{aligned}$$

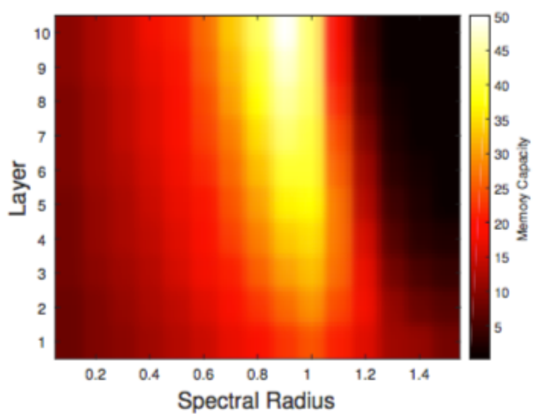
Intrinsically Richer Dynamics

Layering in RNN: a convenient way of architectural setup

- Multiple time-scales representations
- Richer dynamics closer to the edge of stability
- Longer short-time memory



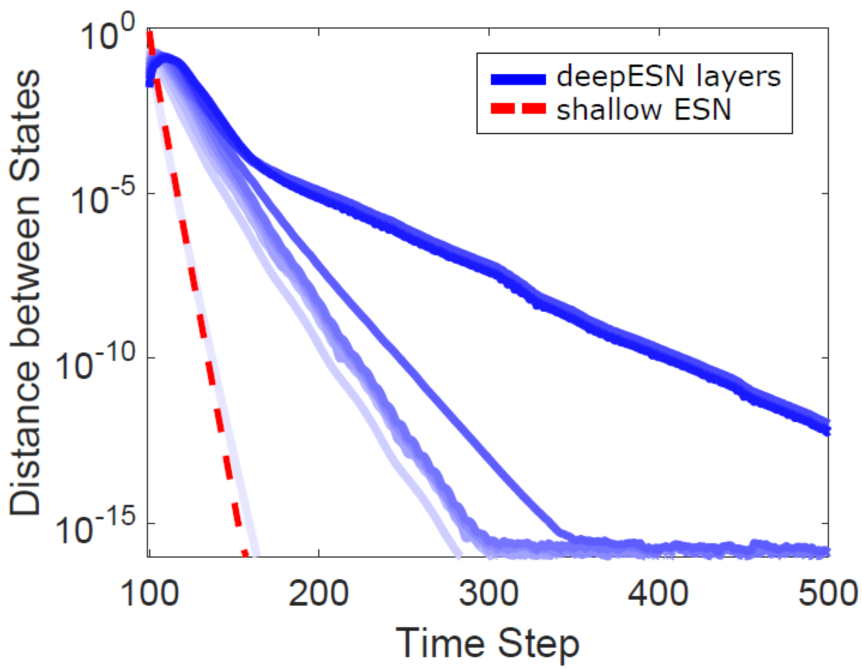




DeepESN: Hierarchical Temporal Features

Structured representation of temporal data through the deep architecture

Empirical Investigations



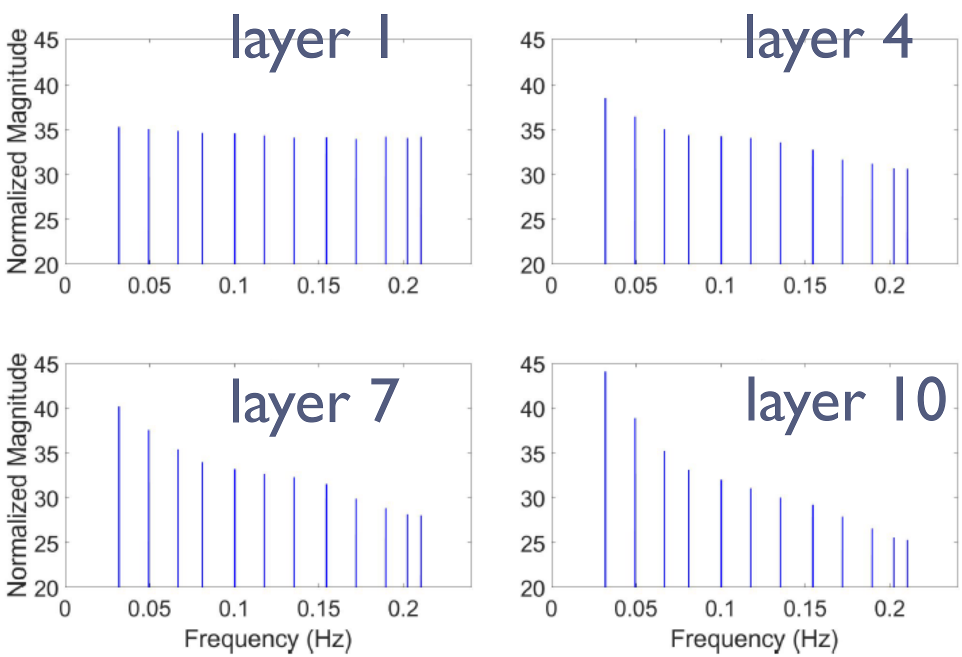
- Effects of input perturbations lasts longer in higher layers
- Multiple time-scales representation
- Ordered along the network's hierarchy

C. Gallicchio, A. Micheli, L. Pedrelli, "Deep Reservoir Computing: A Critical Experimental Analysis", Neurocomputing, 2017

DeepESN: Hierarchical Temporal Features

Structured representation of temporal data through the deep architecture

Frequency Analysis



- Diversified magnitudes of FFT components
- Multiple frequency representation
- Ordered along the network's hierarchy
- Higher layers tend to focus on lower frequencies

[C. Gallicchio, A. Micheli, L. Pedrelli, WIRN 2017]

DeepESN: Mathematical Background

Structured representation of temporal data through the deep architecture

Theoretical Analysis [C. Gallicchio, A. Micheli. Cognitive Computation (2017).]

- Higher layers intrinsically implement less contractive dynamics

$$C^{(i)} = (1 - a^{(i)}) + a^{(i)}\big(C^{(i-1)}\|\mathbf{W}_{in}^{(i)}\| + \|\hat{\mathbf{W}}^{(i)}\|\big) < 1$$

- Echo State Property for Deep ESNs

- Deeper networks naturally develop richer dynamics, closer to the edge of stability [C. Gallicchio, A. Micheli, L. Silvestri. Neurocomputing 2018.]

$$\lambda_{max} = \max_{i,k} \frac{1}{N_s} \sum_{t=1}^{N_s} \ln\left(\left|eig_k\left((1 - a^{(i)})\mathbf{I} + a^{(i)}\mathbf{D}^{(i)}(t)\hat{\mathbf{W}}^{(i)}\right)\right|\right)$$

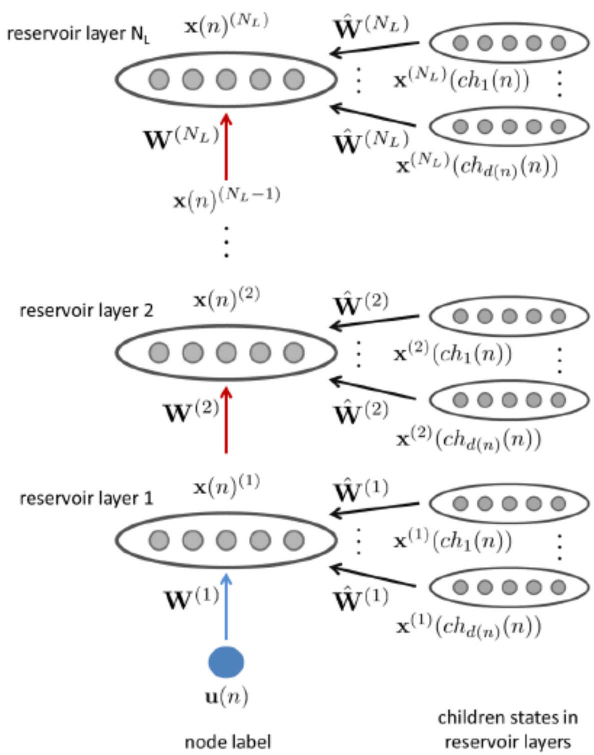
DeepESN: Performance in Applications

Formidable trade-off between performance and computational time

Model	total recurrent units	free-parameters	test ACC	computation time
Piano-midi.de				
DeepESN	6000	540088	**33.33 (0.11) %**	**386**
ESN	6000	540088	30.43 (0.06) %	748
SRN	652	540596	29.48 (0.35) %	3185
LSTM	316	539816	28.98 (2.93) %	2333
GRU	369	539566	31.38 (0.21) %	2821
MuseData				
DeepESN	6000	504082	**36.32 (0.06) %**	**789**
ESN	6000	504082	35.95 (0.04) %	997
SRN	632	503786	34.02 (0.28) %	8825
LSTM	307	504176	34.71 (1.17) %	18274
GRU	358	503072	35.89 (0.17) %	18104
JSBchorales				
DeepESN	6000	324052	**30.82 (0.12) %**	**83**
ESN	6000	324052	29.14 (0.09) %	140
SRN	519	323908	29.68 (0.17) %	341
LSTM	254	325172	29.80 (0.38) %	532
GRU	295	323372	29.63 (0.64) %	230
Nottingham				
DeepESN	6000	360058	69.43 (0.05) %	**677**
ESN	6000	360058	69.12 (0.08) %	1473
SRN	545	360848	65.89 (0.49) %	2252
LSTM	266	361286	70.00 (0.24) %	26175
GRU	309	359116	**71.50 (0.77) %**	11844

C. Gallicchio, A. Micheli, L. Pedrelli, "Comparison between DeepESNs and gated RNNs on multivariate time-series prediction", ESANN 2019.

Deep Tree Echo State Networks



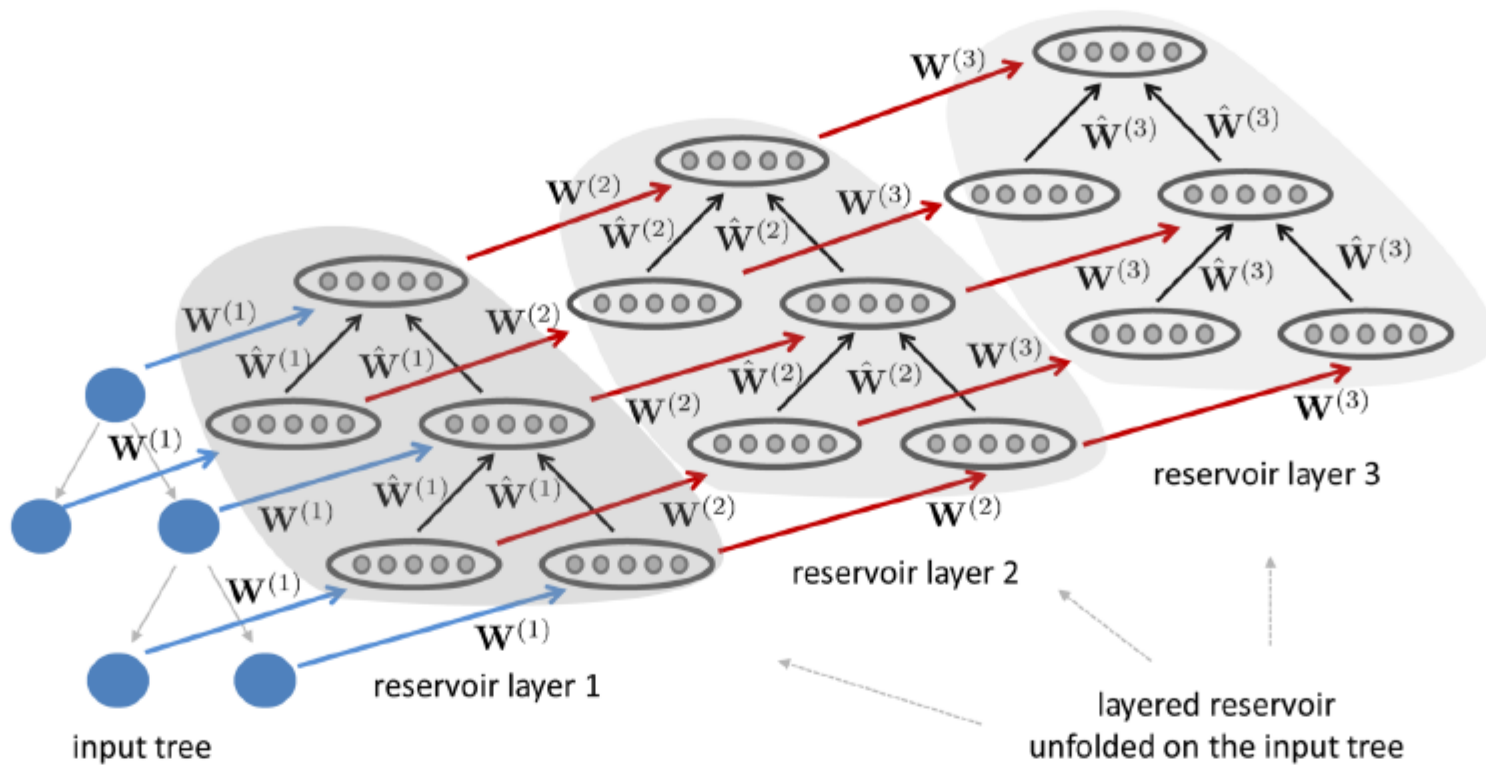
- Deep Tree Echo State Networks
- Untrained multi-layered recursive neural network

$$\mathbf{x}^{(1)}(n) = \tanh\left(\mathbf{W}^{(1)}\mathbf{u}(n) + \frac{1}{d(n)}\sum^{d(n)} \hat{\mathbf{W}}^{(1)}\mathbf{x}^{(1)}(ch_i(n))\right)$$

$$\mathbf{x}^{(l)}(n) = \tanh\left(\mathbf{W}^{(l)}\mathbf{x}^{(l-1)}(n) + \frac{1}{d(n)}\sum_{i=1}^{d(n)} \hat{\mathbf{W}}^{(l)}\mathbf{x}^{(l)}(ch_i(n))\right)$$

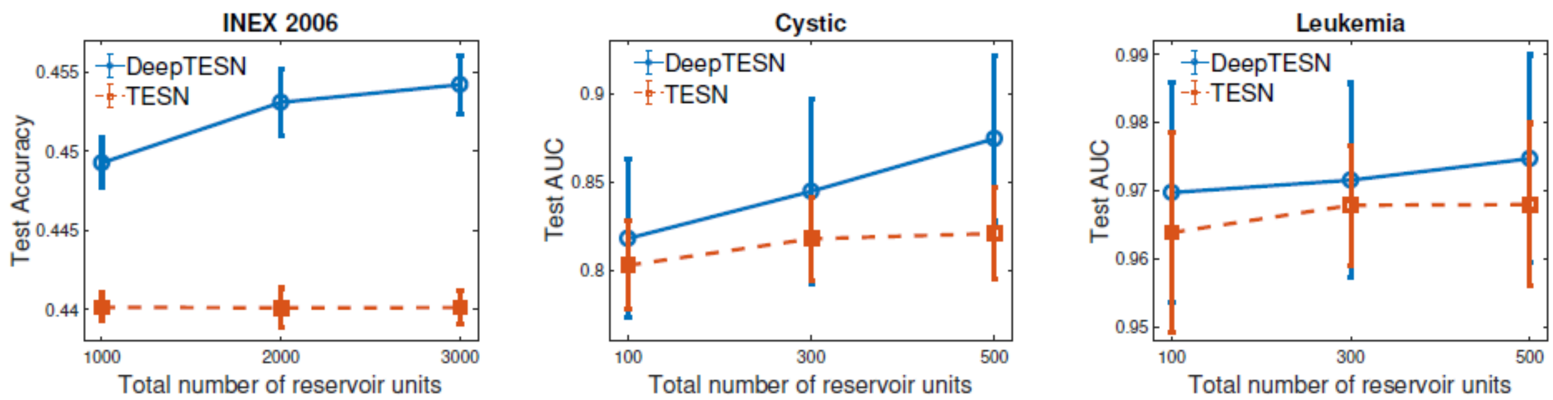
Gallicchio, Micheli, IJCNN, 2018.
Gallicchio, Micheli, Information Sciences, 2019.

Deep Tree Echo State Networks: Unfolding



Deep Tree Echo State Networks: Advantages

- Effective in applications



- Extremely efficient
  - Layered recursive architecture $\mathcal{O}(|\mathcal{N}(\mathbf{t})|\, k\, N_L\, N_R^2)$
  - Shallow case $\mathcal{O}(|\mathcal{N}(\mathbf{t})|\, k\, N_L^2\, N_R^2)$

Model	*INEX 2006*		*Cystic*		*Leukemia*	
	TR	TS	TR	TS	TR	TS
DeepTESN	4.18'	4.20'	0.43"	0.04"	1.65"	0.18"
TESN	39.91'	40.37'	1.60"	0.17"	7.55"	0.83"

Conclusions

- Reservoir Computing: paradigm for efficient modeling of RNNs
- Reservoir: non-linear dynamic component, untrained after contractive initialization
- Readout: linear feed-forward component, trained
- Easy to implement, fast to train
- Markovian flavour of reservoir state dynamics
- Successful applications Recent extensions toward:
  - Deep Learning architecture
  - Structured Domains

References

- Jaeger H. The “echo state” approach to analysing and training recurrent neural networks - with an erratum note. Tech. rep. GMD - German National Research Institute for Computer Science, Tech. Rep. 2001.
- Jaeger H, Haas H. Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. Science. 2004;304(5667):78–80.
- Gallicchio C, Micheli A. Architectural and Markovian Factors of Echo State Networks. Neural Networks 2011;24(5):440–456.
- Lukoševičius, Mantas, and Herbert Jaeger. "Reservoir computing approaches to recurrent neural network training." *Computer Science Review* 3.3 (2009): 127-149.
- Lukoševičius, Mantas. "A practical guide to applying echo state networks." *Neural networks: Tricks of the trade*. Springer Berlin Heidelberg, 2012. 659-686.

References

- Gallicchio, Claudio, Alessio Micheli, and Luca Pedrelli. "Deep reservoir computing: A critical experimental analysis." *Neurocomputing* 268 (2017): 87-99.
- Gallicchio, Claudio, Alessio Micheli, and Luca Pedrelli. "Design of deep echo state networks." *Neural Networks* 108 (2018): 33-47.
- Gallicchio, Claudio, and Alessio Micheli. "Deep echo state network (deepesn): A brief survey." *arXiv preprint arXiv:1712.04323* (2017).
- Gallicchio, Claudio, and Alessio Micheli. "Deep Reservoir Neural Networks for Trees." *Information Sciences* 480 (2019): 174-193.
- Gallicchio, Claudio, and Alessio Micheli. "Graph echo state networks." *The 2010 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2010.