An Introduction to:

Reservoir Computing
and
Echo State Networks

Claudio Gallicchio

gallicch@di.unipi.it

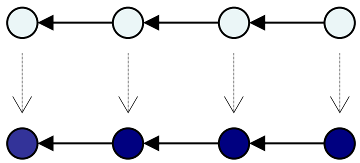
Outline

- **Focus:** Supervised learning in domain of sequences
- Recurrent Neural networks for supervised learning in domains of sequences
- Reservoir Computing: paradigm for efficient training of Recurrent Neural Networks
- Echo State Network model theory and applications

Sequence Processing

Notation used for Tasks on Sequence Domains (Supervised Learning)

Input Sequence → Output Sequence

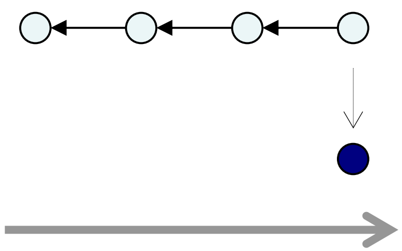


One output vector for each input vector

E.g. Sequence Transdution, Next Step Prediction

Example or sample: $(\mathbf{u}(n), \mathbf{y}_{target}(n))$

Input Sequence → Output Vector



One output vector for each input sequence

E.g. Sequence classification

Example or sample: $(\mathbf{s}(\mathbf{u}), \mathbf{y}_{target}(\mathbf{s}(\mathbf{u})))$

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

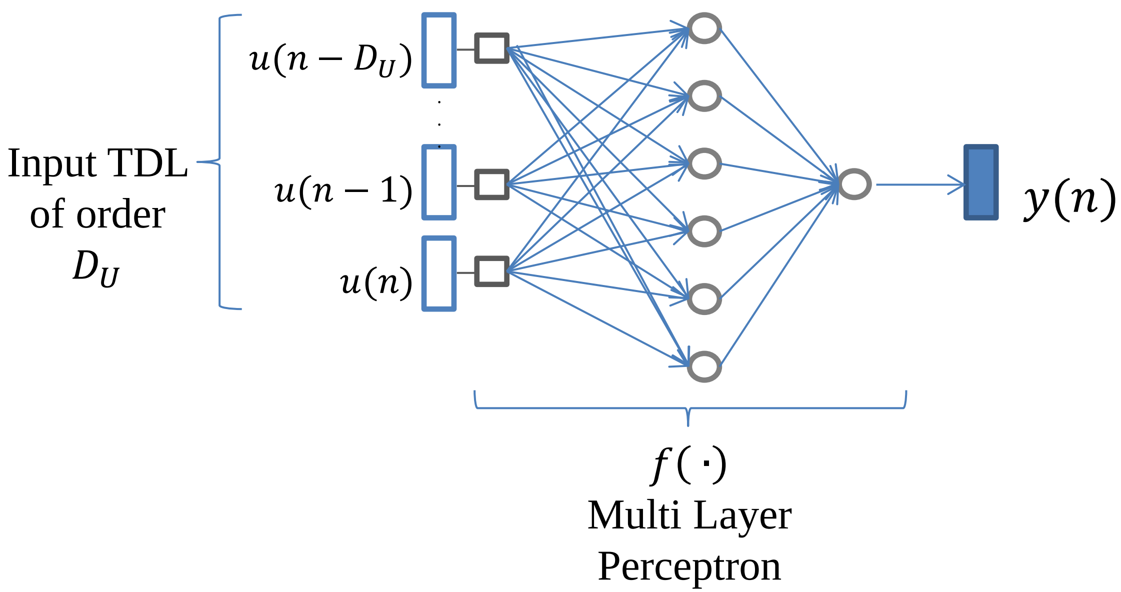
Notation: in the following slides, the variable *n* denotes the *time step*.

Neural Networks for Learning in Sequence Domains

- NN for processing temporal data exploit the idea of representing (more or less explicitely) the past input context in which new input information is observed
- Basic approaches: windowing strategies, feedback connections.

Input Delay Neural Networks (IDNN)

Temporal context represented using feed-forward neural networks + input windowing



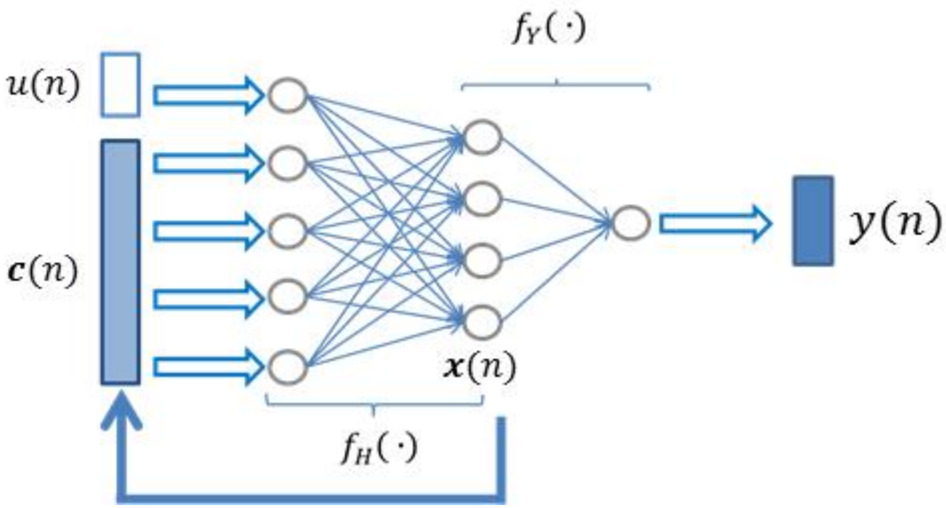
$$\boldsymbol{y}(n) = f(\boldsymbol{u}(n-D_U), \dots, \boldsymbol{u}(n-1), \boldsymbol{u}(n)),$$

- Pro: simplicity, training using Back-propagation
- Con: fixed window size.

Recurrent Neural Networks (RNNs)

- Neural network architectures with explicit recurrent connections
- Feedback allows the representation of temporal context of state information (neural memory) – implement dynamical systems
- Potentially maintain input history information for arbitrary periods of time

Elman Network (Simple Recurrent Network)

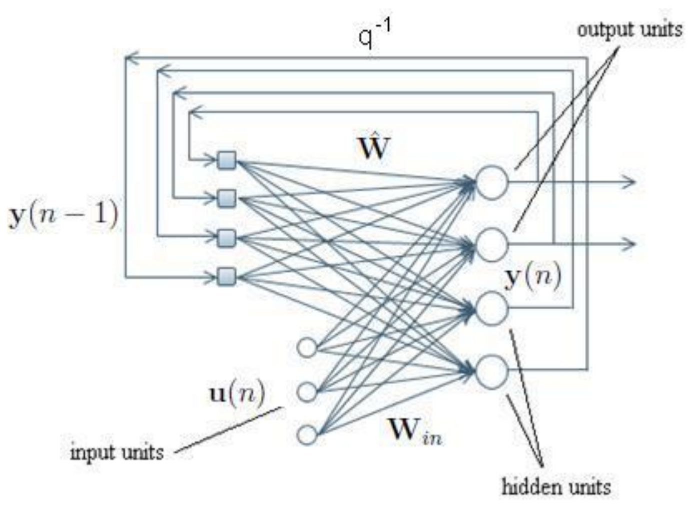


- Pro: theoretically very powerful; Universal approximation through training
- Con: drawbacks related to training

$$\boldsymbol{y}(n) = f_Y(\boldsymbol{x}(n))$$
$$\boldsymbol{x}(n) = f_H(\boldsymbol{u}(n), \boldsymbol{c}(n))$$
$$\boldsymbol{c}(n) = \boldsymbol{x}(n-1)$$

Learning with RNNs

- Universal approximation of RNNs (e.g. Elman, NARX) through learning
- However, training algorithms for RNNs involve some known drawbacks:
    - High computational training costs and slow convergence
    - Local minima (error function is generally a non convex function)
    - Vanishing of the gradient and problem in learning long-term dependencies



Markovian Bias of RNNs

- Properties of RNNs state dynamics in the early stages of training
- RNNs initialized with small weights result in contractive state transition functions and can discriminate among different input histories even prior to learning
- Markovian characterization of the state dynamics is a bias for RNN architectures

- Computational tasks with characterization compatible to such Markovian characterization can be approached by RNNs in which recurrent connections are not trained
- Reservoir Computation paradigm exploits this **fixed** Markovian characterization

Reservoir Computing (RC)

• Paradigm for efficient RNN modeling – state of the art for efficient learning in sequential domains

• Implements dynamical system

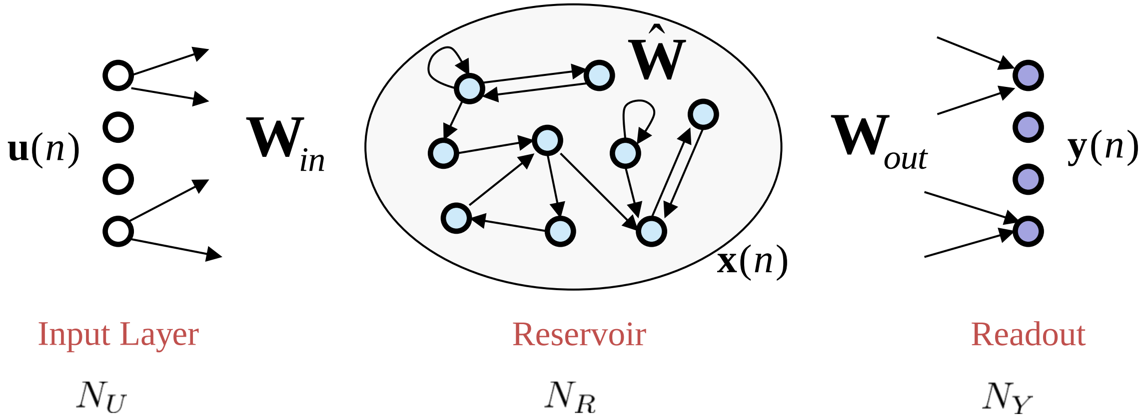
• Conceptual separation: dynamical/recurrent non-linear part (reservoir)
feed-forward output tool (readout)

• **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



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

Reservoir Computation

The reservoir implements the state transition function:

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

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

Iterated version of the state transition function (application of the reservoir to an input sequence):

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

initial state

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

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

Echo State Property (ESP)

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

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

Echo State Property

- Holds if the state of the network is determined uniquely by the left-inifinite input history
- **State contractive, state forgetting, input forgetting**
- The state of the network asymptotically depends only on the driving input signal
- Dependencies on the initial conditions are progressively lost

Conditions for the Echo State Property

Conditions on $\hat{\mathbf{W}}$

Sufficient: maximum singular value is less than 1

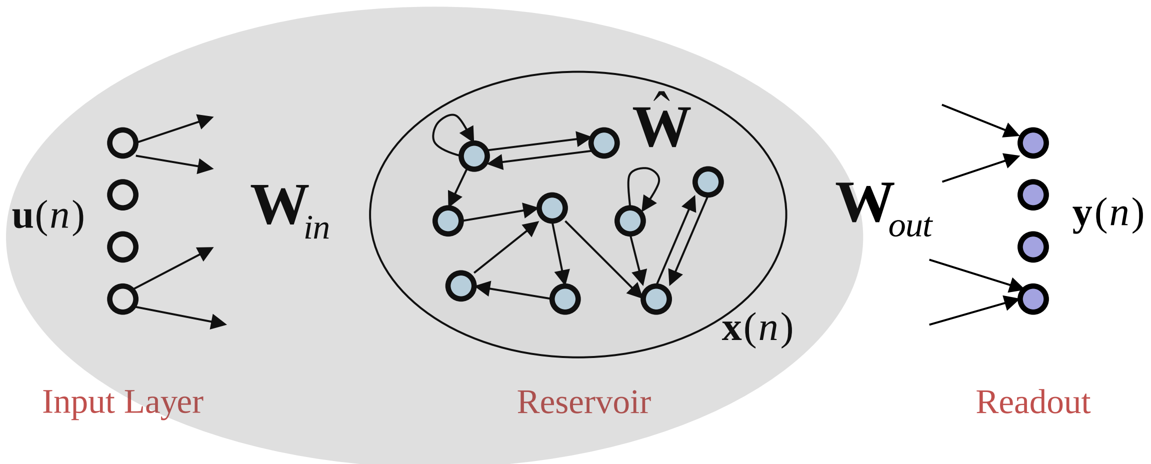
$\sigma(\hat{\mathbf{W}}) = \|\hat{\mathbf{W}}\|_2 < 1$ (contractive dynamics)

Necessary: spectral radius is less than 1 $\rho(\hat{\mathbf{W}}) = max(|eig(\hat{\mathbf{W}})|)$

$\rho(\hat{\mathbf{W}}) < 1$ (asymptotically stable around the **0** state)

How to Initialize ESNs

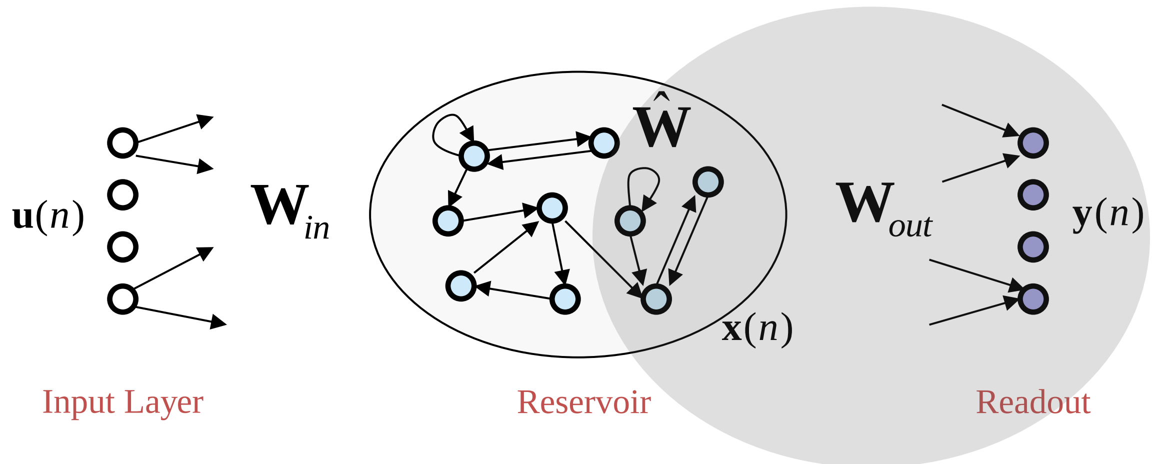
Reservoir Initialization



- $\mathbf{W}_{in}$ initialized randomly in $[-w_{in}, w_{in}]$
- $\hat{\mathbf{W}}$ initialization procedure:
  - Start with a randomly generated matrix $\hat{\mathbf{W}}_{random}$
  - Scale to meet the condition for the ESP $\hat{\mathbf{W}} = \hat{\mathbf{W}}_{random} \dfrac{\rho_{desired}}{\rho(\hat{\mathbf{W}}_{random})}$

Training ESNs

Training Phase



- Discard an *initial transient* (washout)
- Collect the reservoir states and target values for each *n*

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

- Train the linear readout:

$$min \|\mathbf{W}_{out}\mathbf{X} - \mathbf{Y}_{target}\|_2^2$$

Training the Readout

- Off-line training: standard in most applications

Moore-Penrose pseudo-inversion

$$\mathbf{W}_{out} = \mathbf{Y}_{target}\mathbf{X}^{+}$$

(possible regularization using random noise)

Ridge Regression

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

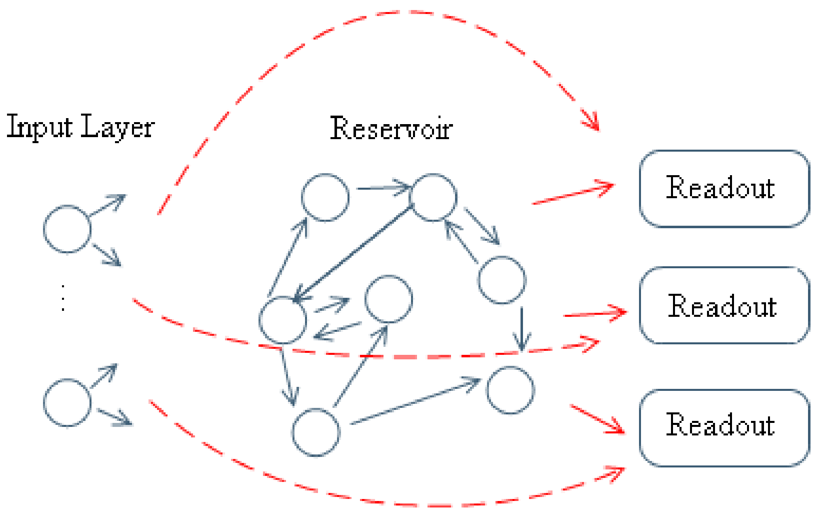
λ_r is the *regularization coefficient* (tipically < 1)

- On-line training
    - Least Mean Squares typically not suitable (ill posed problem)
    - Recursive Least Squares more suitable

Training the Readout

- **Other readouts:**
  - MLPs, SVMs, kNN, etc...

- **Multiple readouts** for the same reservoir: solving more tasks with the same reservoir dynamics



ESN Hyper-parametrization (Model Selection)

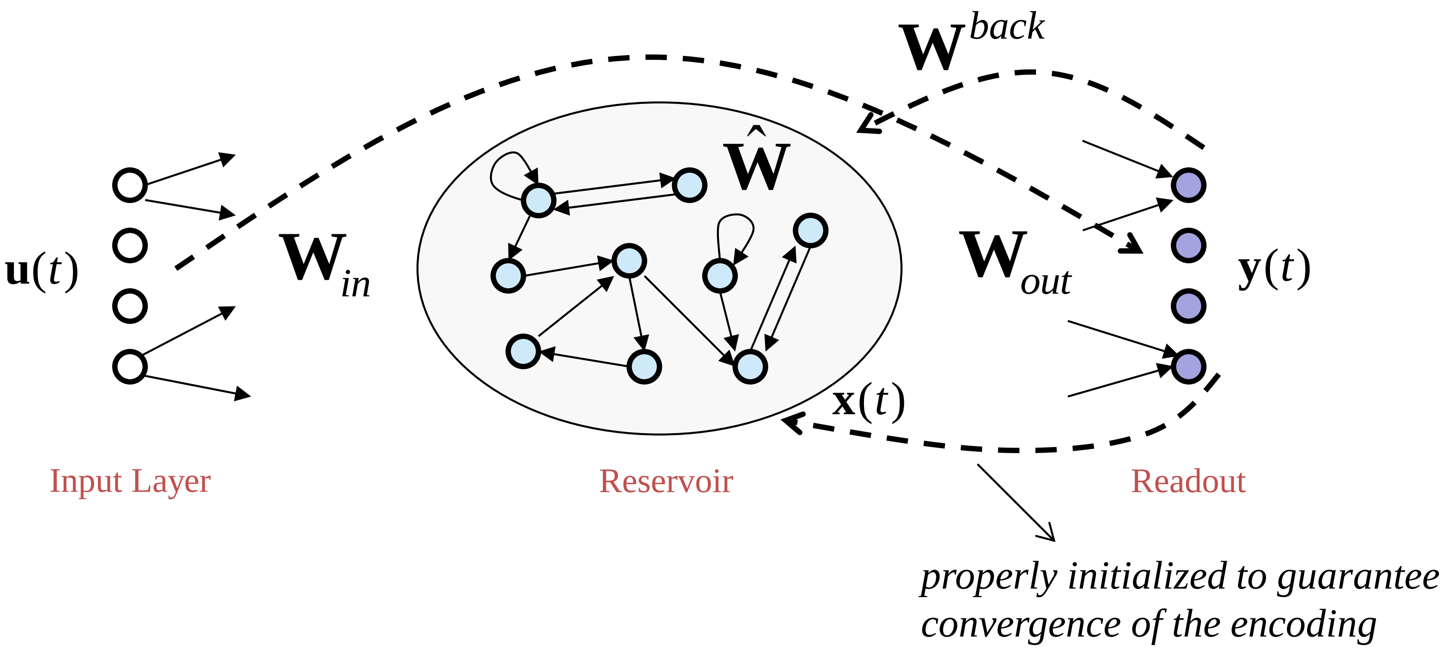
Easy, Efficient, but many *fixed* hyper-parameters to set...

- Reservoir dimension N_R
- Spectral radius ρ
- Input Scaling w_{in}
- Readout Regularization λ_r

- Reservoir sparsity
- Non-linearity of reservoir activation function
- Input bias
- Architectural design
- Length of the transient (settling time)

<u>ESN hyper-parametrization should be chosen carefully through an appropriate model selection procedure</u>

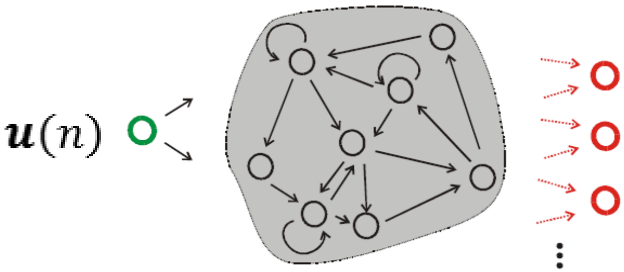
ESN Architectural Variants



- direct input-to-readout connections
- output feedback connections (stability issues)

Memory Capacity

How long is the effective short-term memory of ESNs?



$$\boldsymbol{y}_1(n) = \boldsymbol{u}(n-1)$$
$$\boldsymbol{y}_2(n) = \boldsymbol{u}(n-2)$$
$$\boldsymbol{y}_3(n) = \boldsymbol{u}(n-3)$$
$$\vdots$$

$$MC = \sum_{k=1}^{\infty} \rho^2(\boldsymbol{u}(n-k), \boldsymbol{y}_k(n))$$

Squared correlation coefficient

Some results:

The MC is bounded by the reservoir dimension

$$MC \leq N_R$$

The MC is maximum for linear reservoirs

$MC = N_R$ $\Longrightarrow$ Dilemma: memory capacity VS non-linearity

- Longer delays cannot be learnt better than short delays ("fading memory")
- It is impossible to train ESNs on tasks which require unbounded-time memory

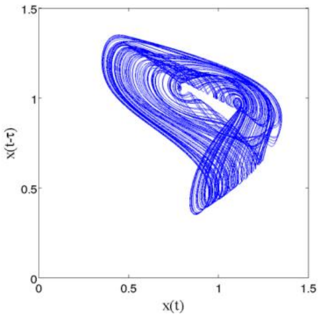
Applications of ESNs

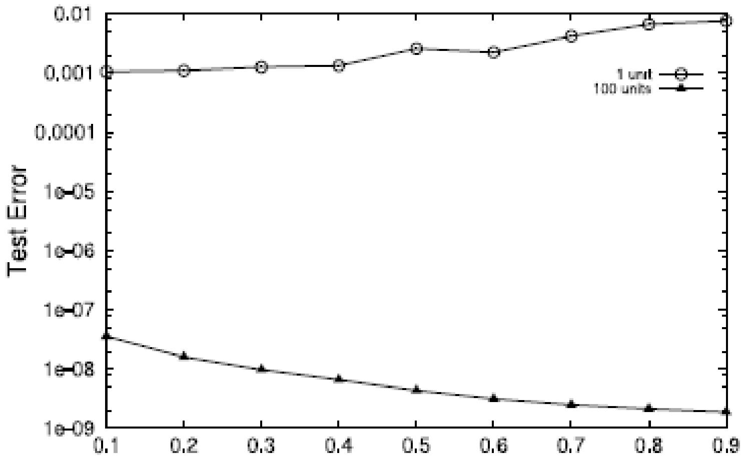
- Several hundreds of relevant applications of ESNs are reported in literature
  - (Chaotic) Time series prediction
  - Non-linear system identification
  - Speech recognition
  - Sentiment analysis
  - Robot localization & control
  - Financial forecasting
  - Bio-medical applications
  - Ambient Assited Living
  - Human Activity Recognition
  - ...

Applications of ESN – Examples/1

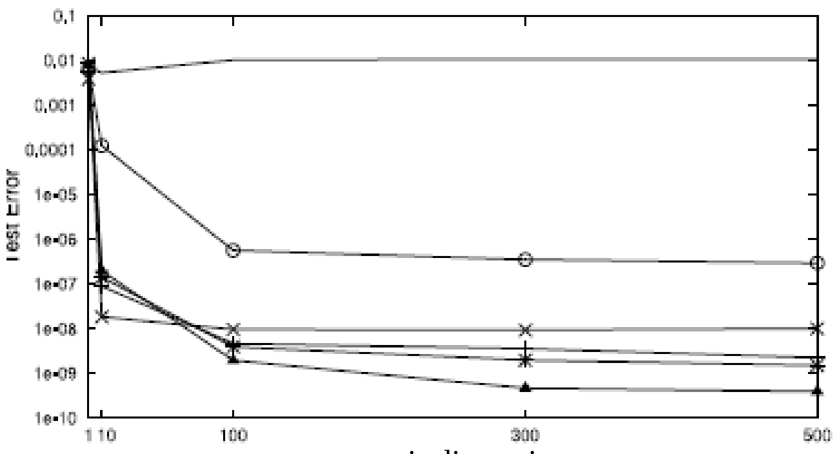
Mackey-Glass time series

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





contraction coefficient

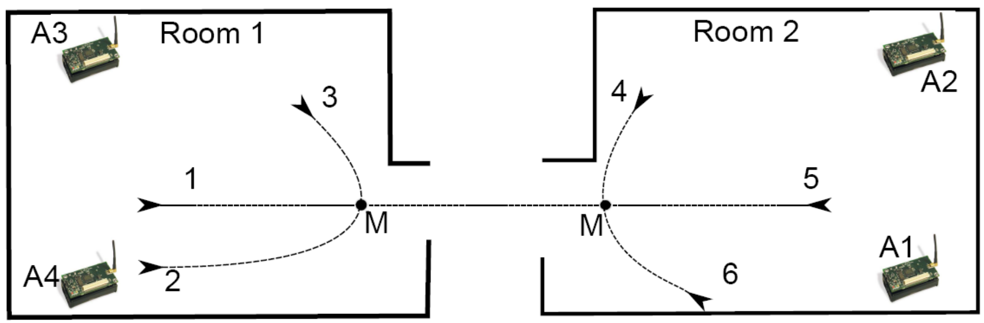


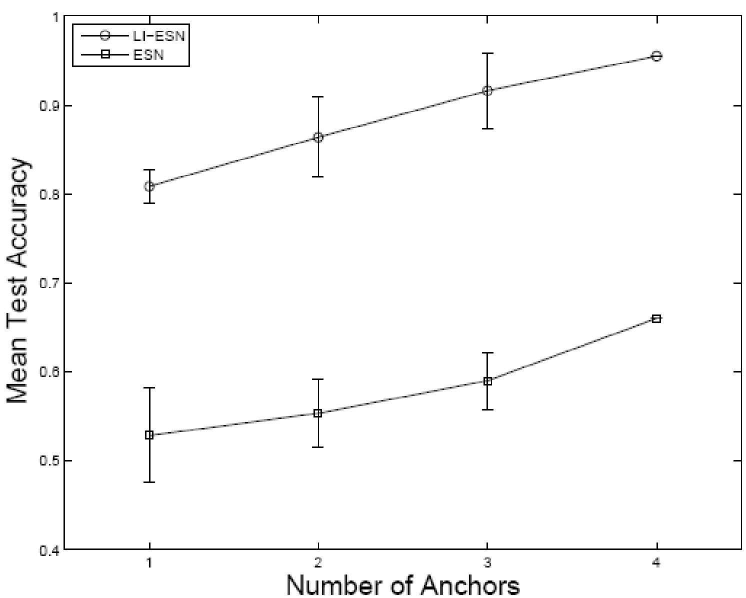
reservoir dimension

Extremely good approximation performance!

Applications of ESN – Examples/2

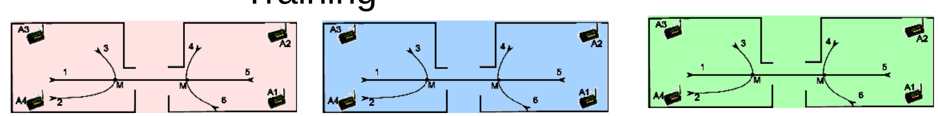
Forecasting of user movements





- Generalization of predictive performance to unseen environments

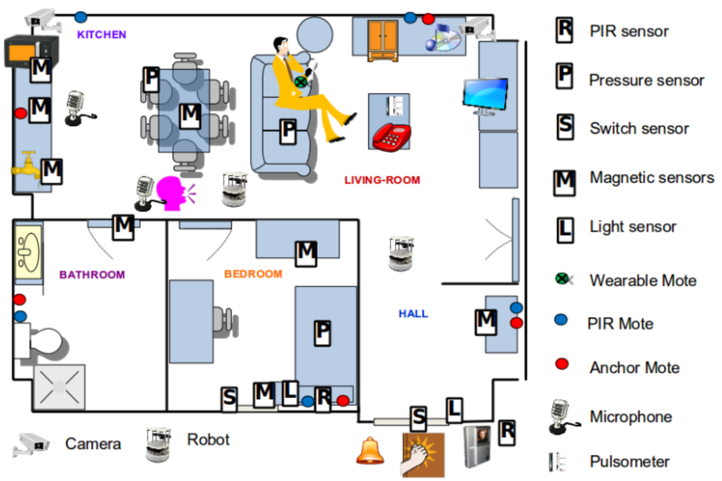
Training

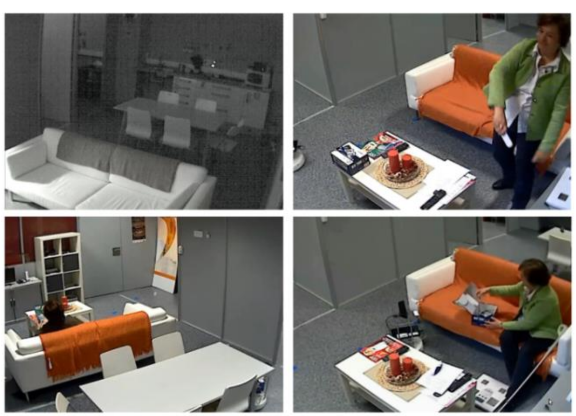


Homogeneous	Heterogeneous
95.95%(±3.54)	89.52%(±4.48)

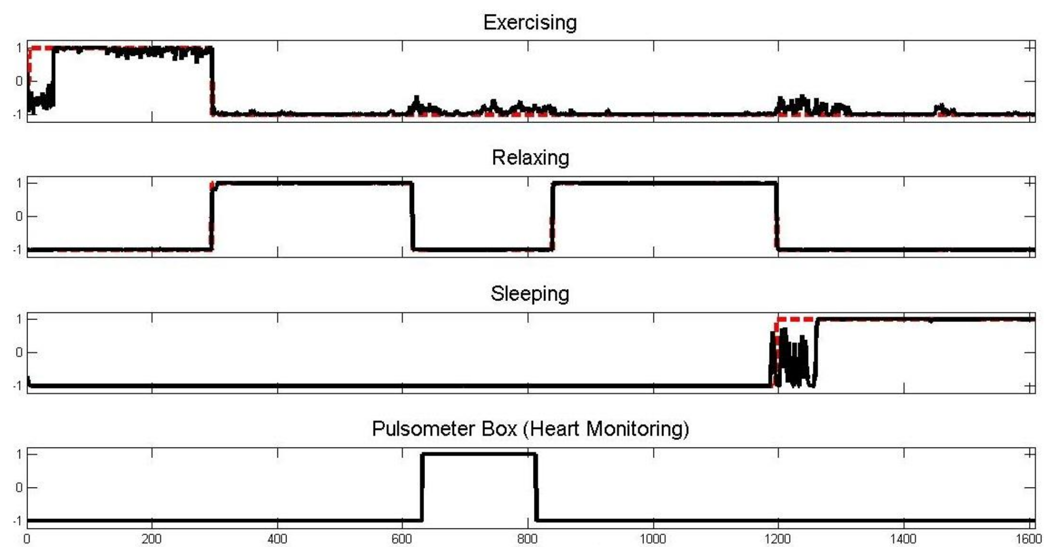
Applications of ESN – Examples/3

Human Activity Recognition and Localization



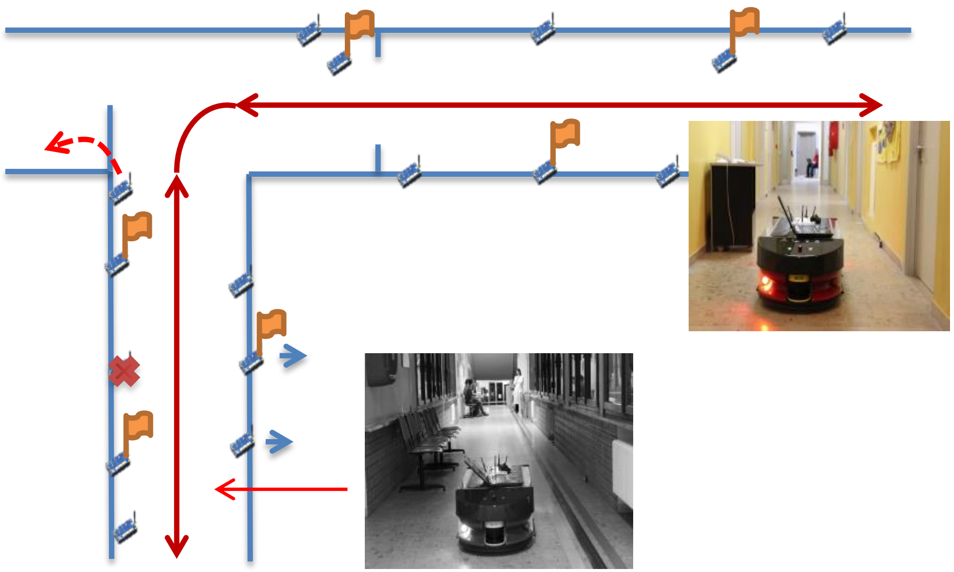


- Input from heterogeneous sensor sources (data fusion)
- Predicting event occurrence and confidence
- Effectiveness in learning a variety of HAR tasks
- Training on new events
- Average test accuracy is 91.32%($\pm$0.80)



Applications of ESN – Examples/4

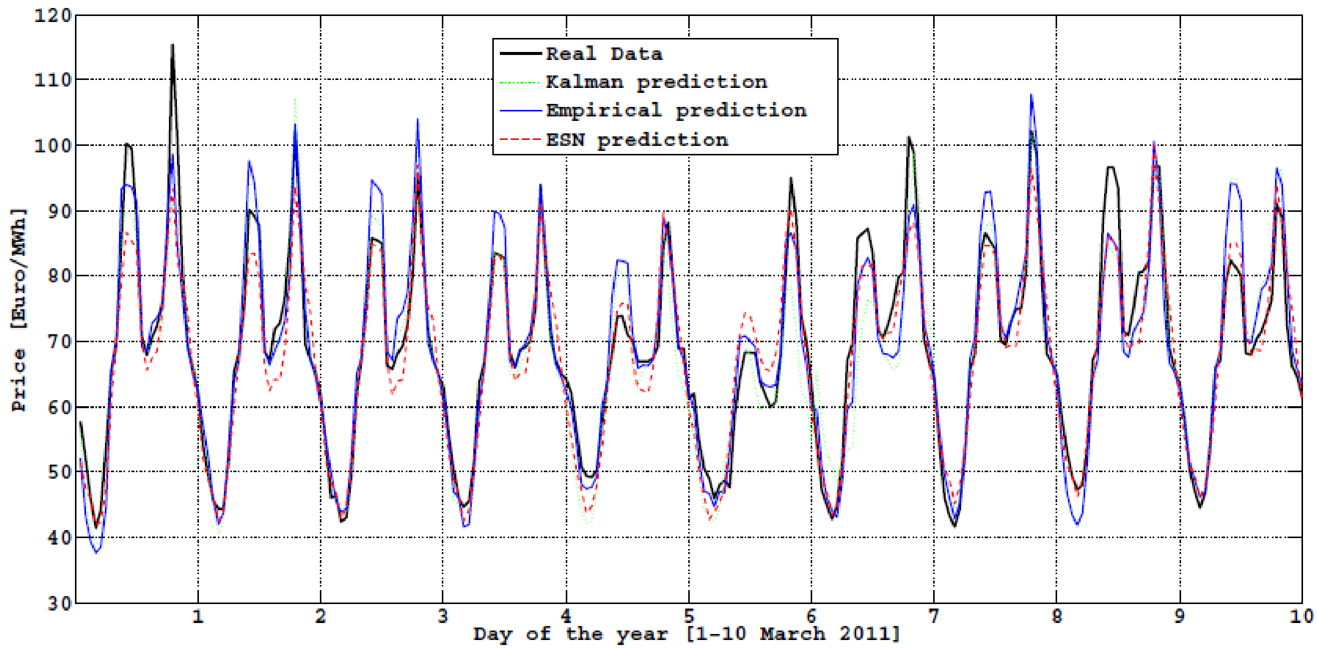
Robotics



- 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

Applications of ESN – Examples/5

Prediction of Electricity Price on the Italian Market

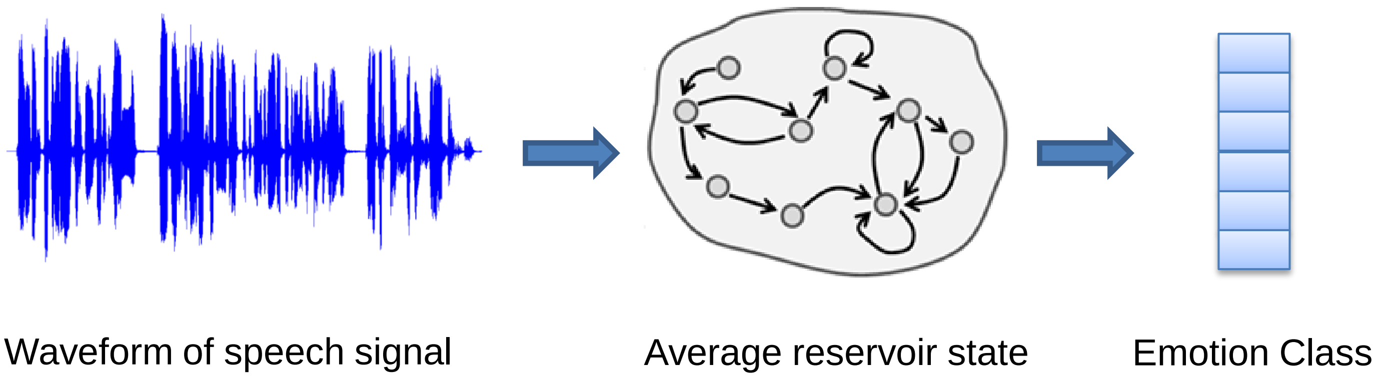


Accurate prediction of hourly electricity price (less than 10% MAPE error)

Applications of ESN – Examples/6

Speech and Text Processing

EVALITA 2014 - Emotion Recognition Track (Sentiment Analysis)



- **Challenge**: the reservoir encodes the temporal input signals avoiding the need of explicitly resorting to fixed-size feature extraction
- **Promising performances** already in line with the state of the art

Markovianity

- **Markovian nature**: 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

Contractivity

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

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

τ is **contractive** if the following property is satisfied

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

Markovianity

- Markovianity and contractivity: Iterated Function Systems, fractal theory, *architectural bias* of RNNs
  RNNs initialized with small weigths (with contractive state transition function) and bounded state space implement (approximate arbitrarily well) definite memory machines
- RNN dynamics constrained in region of state space with Markovian characterization
- Contractivity (in any norm) of state transition function **impies** Echo States (<u>next slide</u>)
- ESNs featured by *fixed* contractive dynamics
- Relations with the universality of RC for bounded memory computation

Contractivity and ESP

A contractive setting of the state transition function τ (in any norm) implies the ESP.

Assumption: τ is contractive with parameter *C*.

$$\|\hat{\tau}([\mathbf{u}(1),\ldots,\mathbf{u}(n)],\mathbf{x})-\hat{\tau}([\mathbf{u}(1),\ldots,\mathbf{u}(n)],\mathbf{x}')\|$$

$$=\|\tau(\mathbf{u}(n),\hat{\tau}([\mathbf{u}(1),\ldots,\mathbf{u}(n-1)],\mathbf{x}))-\tau(\mathbf{u}(n),\hat{\tau}([\mathbf{u}(1),\ldots,\mathbf{u}(n-1)],\mathbf{x}'))\|$$

$$\leq C\|\hat{\tau}([\mathbf{u}(1),\ldots,\mathbf{u}(n-1)],\mathbf{x})-\hat{\tau}([\mathbf{u}(1),\ldots,\mathbf{u}(n-1)],\mathbf{x}')\|$$

$$\leq \ldots$$

$$\leq C^{n-1}\|\hat{\tau}([\mathbf{u}(1)],\mathbf{x})-\hat{\tau}([\mathbf{u}(1)],\mathbf{x}')\|$$

$$=C^{n-1}\|\tau(\mathbf{u}(1),\hat{\tau}([\,],\mathbf{x}))-\tau(\mathbf{u}(1),\hat{\tau}([\,],\mathbf{x}'))\|$$

$$=C^{n-1}\|\tau(\mathbf{u}(1),\mathbf{x})-\tau(\mathbf{u}(1),\mathbf{x}')\|$$

$$\leq C^{n}\|\mathbf{x}-\mathbf{x}'\|$$

Approaches 0 as *n* goes to infinity

Contractivity and Reservoir Initialization

Reservoir is initialized to implement a *contractive* state transition function, so that the ESP is guaranteed.

This leads to the formulation of the sufficient condition on the maximum singular value of the recurrent reservoir weight matrix:

$$\sigma(\hat{\mathbf{W}}) = \|\hat{\mathbf{W}}\|_2 < 1$$

Assumption: Euclidean distance as metric in the reservoir space, *tanh* as reservoir activation function

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

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

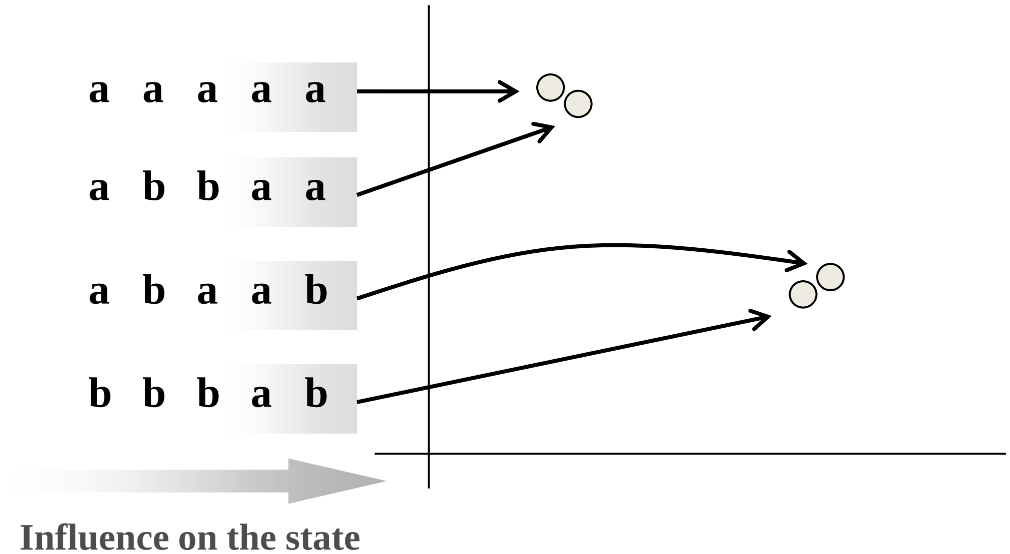
$$\leq max(|tanh'|)\|\hat{\mathbf{W}}(\mathbf{x} - \mathbf{x}')\|_2$$

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

$$\|\hat{\mathbf{W}}\|_2 < 1 \Rightarrow \text{Contractivity of } \tau$$

(sufficient condition for the ESN)

Markovianity



- Input sequences sharing a common suffix drive the ESN into close states, proportionally to the length of the suffix
- Ability to intrinsically discriminate among different input sequences in a suffix-based fashion without adaptation of the reservoir parameters
- Target task should match Markovianity of reservoir state space

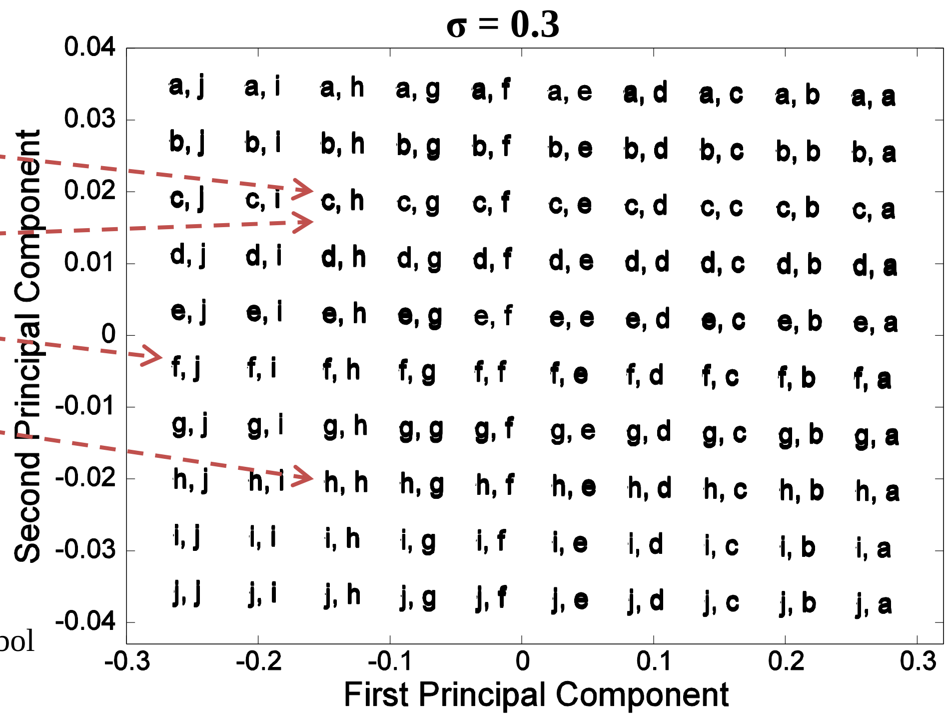
Markovianity

b e j a a c h

a e i e c c h

g j e e g f j

g j e e g h h



1st PC: last input symbol

2nd PC: next-to-last input symbol

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 (tasks compatible with Markovian characterization)
- Model Selection: many hyper-parameters to be set

Research Issues

- Optimization of reservoirs: supervised or unsupervised reservoir adaptation (e.g. Intrinsic Plasticity)
- Architectural Studies: e.g. Minimum complexity ESNs, φ-ESNs, ...
- Non-linearity vs memory capacity
- Stability analysis in case of output feedbacks
- <u>**Reservoir Computing for Learning in Structured Domains**</u>
  TreeESN, GraphESN
- **Applications, applications, applications, applications ...**