HAL open science

Training recurrent Random Neural Networks: first and second-order techniques, reservoir models, numerical aspects

Gerardo Rubino

▶ To cite this version:

Gerardo Rubino. Training recurrent Random Neural Networks: first and second-order techniques, reservoir models, numerical aspects. Conference on Modelling Methods in Computer Systems, Networks and Bioinformatics, Oct 2019, Paris, France. pp.1-51. hal-02430427

HAL Id: hal-02430427

https://inria.hal.science/hal-02430427v1

Submitted on 8 Jan 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Training recurrent Random Neural Networks: first and second-order techniques, reservoir models, numerical aspects

G. Rubino

INRIA / IRISA
Rennes, France

based on joint work with S. Basterrech, P. Rodríguez-Bocca, and others

Conference on Modelling Methods
in Computer Systems, Networks and Bioinformatics,
Paris, October 15, 2019

1. The PSQA project
2. Reservoir Computing models
3. ESQN
4. Empirical results
5. Some conclusions

Outline

1. The PSQA project
2. Reservoir Computing models
3. ESQN
4. Empirical results
5. Some conclusions

- Consider any kind of application or service consisting basically in transporting audio and/or video signals over the Internet.
- The media can be sent one-way (e.g., a video streaming service) or two-ways (e.g., an IP-telephony application).
- For many reasons (compression techniques used, congestion in the network leading to delays, and/or to losses, external perturbation agents such as interferences in some networking technologies, etc.) transmission can suffer from content degradation.
- Perceptual quality (PQ) refers to the (subjective) perception (the view, the feeling) the user has on the quality of this media transmission system, on the quality of what she receives, on the impact of the degradations due to the transmission over the network.

- Two main characteristics:
  - It is, by definition, a subjective concept.
  - It lacks a formal definition.
- Somehow perceptual quality is at the heart of the designer's goals for these systems, it is her *ultimate target*, complemented by some other aspects of system's properties (e.g. security).
- It is measured daily using panels of humans, following specific norms. The area is called *subjective testing*, and the resulting PQ numerical value (we say a MOS value, for Mean Opinion Score) is very robust.
- At our team in INRIA, we decided to look for automatic and real time measuring techniques.

- There are *objective testing* methods, avoiding using panels:
  - some come from coding, and compare the received signal to the original one; we say that they are methods with reference (they need to access the original signal);
  - others consist of analyzing the received signal only, looking for impairments; they are no reference methods.
- Both have strong limitations:
  - accuracy, the most important one,
  - and lack of real time capabilities.
- Comment: new objective techniques have been developed where accuracy has been improved. More on this later.

- PSQA stands for Pseudo Subjective Quality Assessment.
- PSQA is a procedure
  - with no reference,
  - automatic,
  - accurate (so far, "optimally"),
  - and it works in real time if necessary or useful.
- It has been extensively tested in many contexts and for many applications, for video or voice (also for audio), working both one-way and two-ways communications.
- PSQA is network-dependent and application-dependent.
- It is a parametric approach (a "black-box" approach), mapping QoS parameters and source-based parameters into perceptual quality.

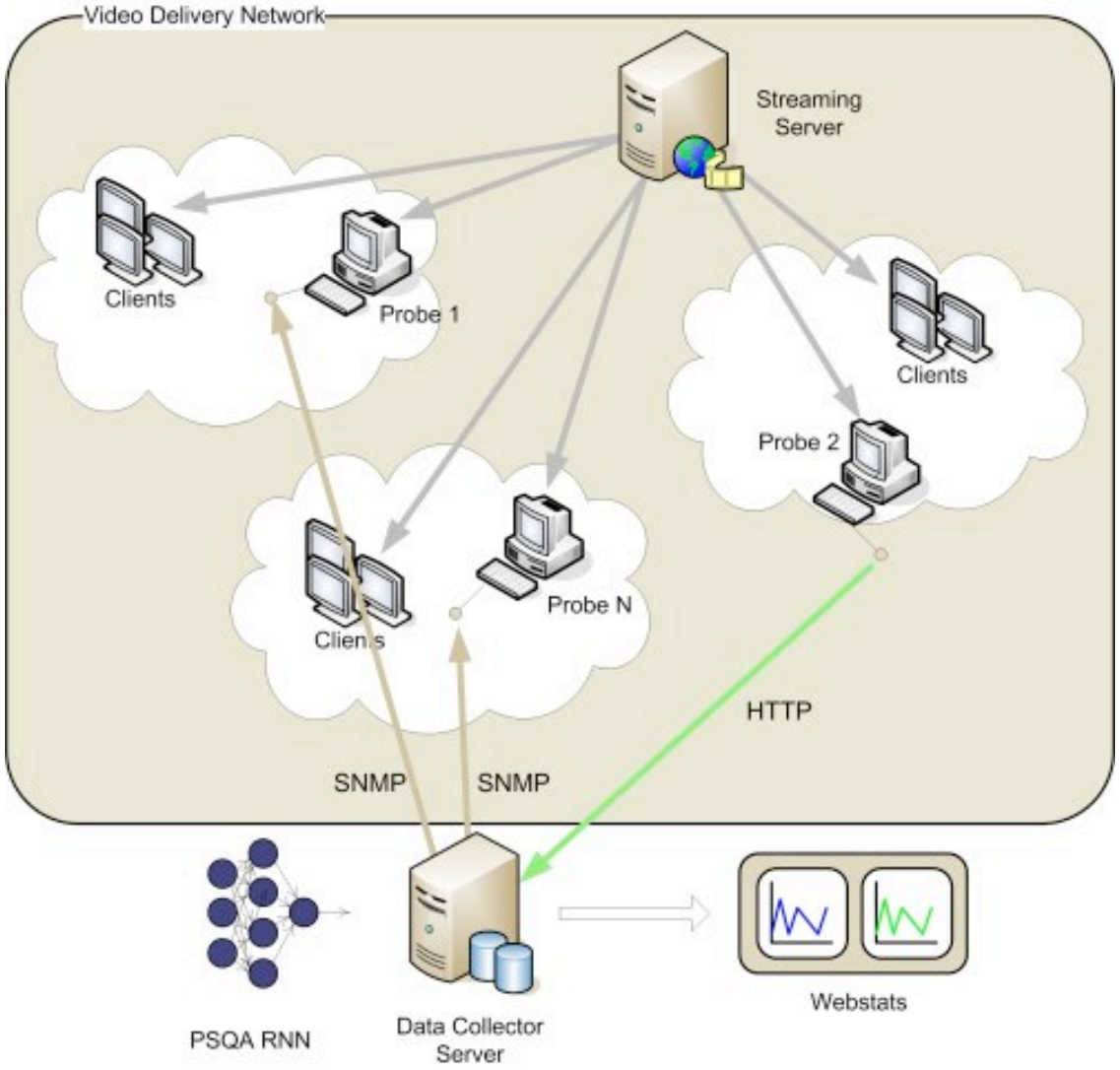
How PSQA works

- First of all, we select
  - (i) measurable QoS metrics characterizing the state of the network (logically speaking, instantaneous measures), and assumed, a priori, to have an impact on the Perceptual Quality,
  - (ii) and metrics related to the channel or to the source, again, expected to have impact on the PQ.
- Example in video: (i) the instantaneous packet loss rate LR, the average size of a burst of loss packets $MLBS$, (ii) the Bit Rate BR and the Frame Rate FR.
- Example in VoIP: (i) the instantaneous packet loss rate LR, the average size of a burst of loss packets $MLBS$, (ii) the Bit Rate BR and the Offset of the FEC (Forward Error Correction) $OFEC$.
- The selected variables will be the input ones in the function whose output will be the PQ measure.

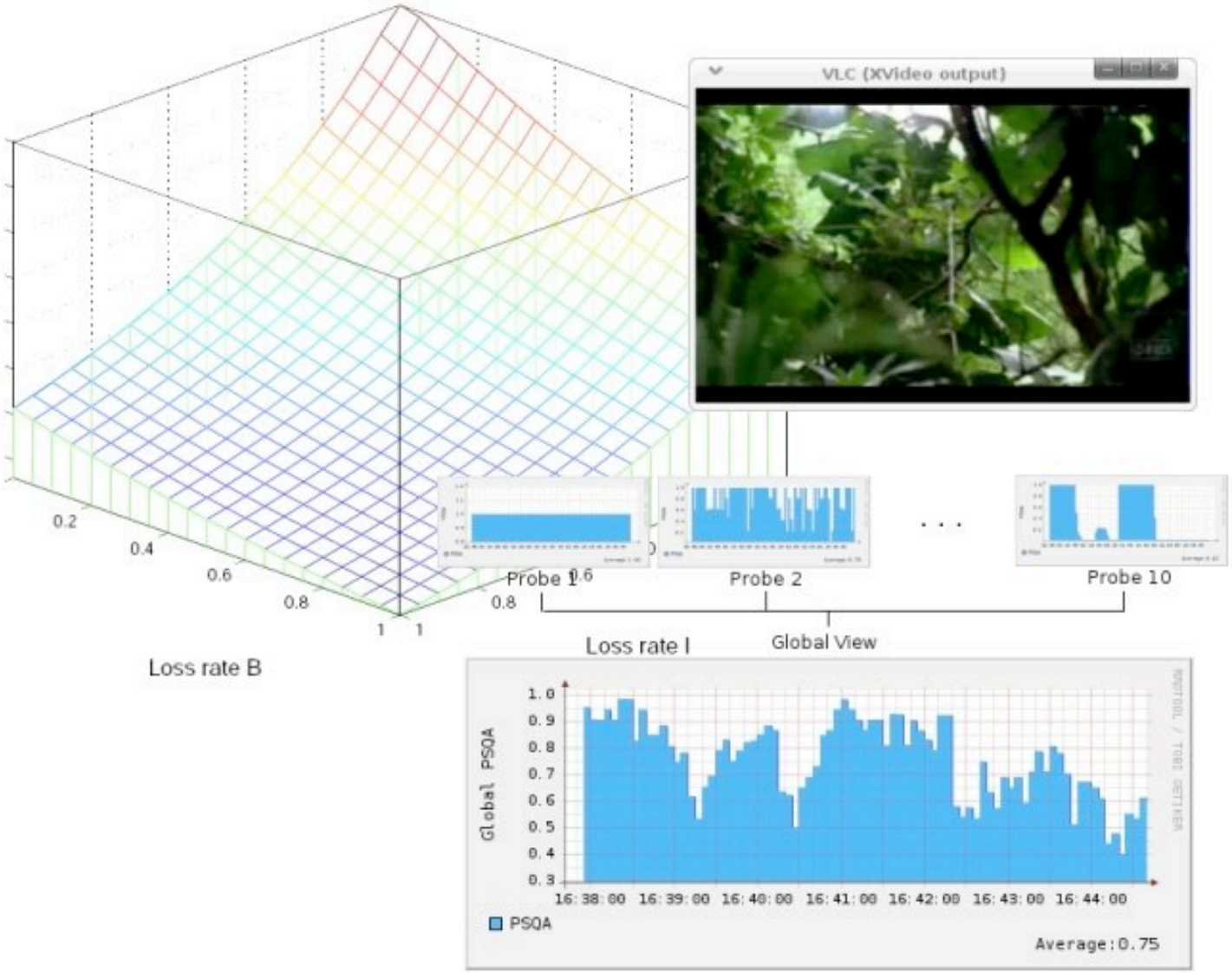
- Let $\vec{x} = (x_1, \ldots, x_n)$ denote the vector whose components are the n chosen metrics. We call it configuration.
- Configurations live in some product space $S_1 \times \cdots \times S_n$.
- Our postulate: PQ depends only on $\vec{x}$ (when "things go well") and not on signal content.
- We select K signals representative of the application or service target, $\sigma_1, \ldots, \sigma_K$. Typically, $K = 100, 200, \ldots$. Some are in general repeated in the set.
- We then build a set of K configurations by a mix of random sampling and quasi-Monte Carlo (or weak discrepancy sequences). Call them $\vec{\gamma}_1, \ldots, \vec{\gamma}_K$.
- Last, we build a platform allowing to send signal σ_i through a simulated or deployed network where we can simultaneously control all components of the configurations.

- Then we send the different original σ_is using the platform, when the configuration is $\vec{\gamma}_i$ using one configuration at a time, and obtain a possibly degraded sequence σ'_i. We show it to a panel of humans and we obtain its PQs Q_i.
- We thus obtain a set of K received sequences, with variable but known PQ (MOS values) coming from subjective testing sessions, and for each, we know which configuration was used to obtain it.
- Then, we use a RNN (a G-network), of the classical feedforward type with 3 layers, to learn the mapping from configurations to PQ (a mapping from QoS$\times$"channel state" to MOS values). Data: the M pairs $\left(\gamma_j, Q_j\right)$.
- In the example of video, we obtain a function $\nu(LR, MLBS, BR, FR)$. In the VoIP example, we obtain a function $\nu(LR, MLBS, BR, OFEC)$.

Example: auditing the network with PSQA (cont.)



"Graphical" overview (from twice-awarded prototype):



Example in analytical models

- Consider the bottleneck of a video transmission system modeled as an $M/M/1/N$ with load $\varrho \neq 1$.
- Classical approach: choose N and ϱ such that $p_L < \varepsilon$, using

$$p_L = \frac{1-\varrho}{1-\varrho^{N+1}}\varrho^N$$

- PSQA-based approach: choose N and ϱ such that *the perceived quality* Q satisfies $Q \geq Q_0$, using

$$Q = \frac{13.3 + 2.01\varrho + 6.74\varrho^N - 20.0\varrho^{N+1} - 2.01\varrho^{N+2}}{16.6 + 3.99\varrho + 93.3\varrho^N - 110\varrho^{N+1} - 3.99\varrho^{N+2}}$$

oo

Two ongoing projects

1/ Instead of measuring PQ at present time t, we want to predict it at time $t + \Delta$, with Δ "small".

2/ We want to eliminate the use of humans panels when building PSQA measuring modules.

- For the second topic, we must move to Big Data problems and to the exploration of Deep Learning tools. Not covered here.
- The first project, so far worked with Sebastian Basterrech, researcher at the VSB Technical University of Ostrava, Czech Republic, leads to
  - to use recurrent topologies (so far, PSQA uses FeedForward ones), and the necessary specific related optimization procedures for learning,
  - then, to face the possible difficulties in training them,
  - to explore Reservoir Computing techniques,
  - to explore the case of vector time series with *time* and *spatial* correlated components.

Two ongoing projects

1/ Instead of measuring PQ at present time t, we want to predict it at time $t + \Delta$, with Δ "small".

2/ We want to eliminate the use of humans panels when building PSQA measuring modules.

- For the second topic, we must move to Big Data problems and to the exploration of Deep Learning tools. Not covered here.
- The first project, so far worked with Sebastian Basterrech, researcher at the VSB Technical University of Ostrava, Czech Republic, leads to
  - to use recurrent topologies (so far, PSQA uses FeedForward ones), and the necessary specific related optimization procedures for learning,
  - then, to face the possible difficulties in training them,
  - to explore Reservoir Computing techniques,
  - to explore the case of vector time series with *time* and *spatial* correlated components.

Outline

1. The PSQA project
2. Reservoir Computing models
3. ESQN
4. Empirical results
5. Some conclusions

Reservoir Computing (RC)

Definitions

- They are an attempt to develop models that uses the potential for *memorization* of recurrent neural networks without the difficulties in the training process of these networks.
- They appeared at the beginning of the 2000s, and they are known today under the name of *Reservoir Computing* (RC) paradigm.
- The two most popular RC models are
  - the *Echo State Network (ESN)*
    (see H. Jaeger, "The *echo state* approach to analysing and training recurrent neural networks, "German National Research Centre for Information Technology, Tech. Rep. 148, 2001)
  - and the *Liquid State Machine (LSM)*
    (see W. Maass, "Liquid state machines: Motivation, theory, and applications," in *Computability in Context: Computation and Logic in the Real World*, Imperial College Press, 2010, pp. 275-296).

The origin of the terminology:

- In the LSM literature, the reservoir is often referred to as the liquid term used in neurosciences for the brain.
- In the ESN community, the dynamic reservoir and its states are termed *echoes* of its input history.
- It is an intuitive metaphor of the excited states as ripples on the surface of a pool of water.

Echo State Networks (ESNs)

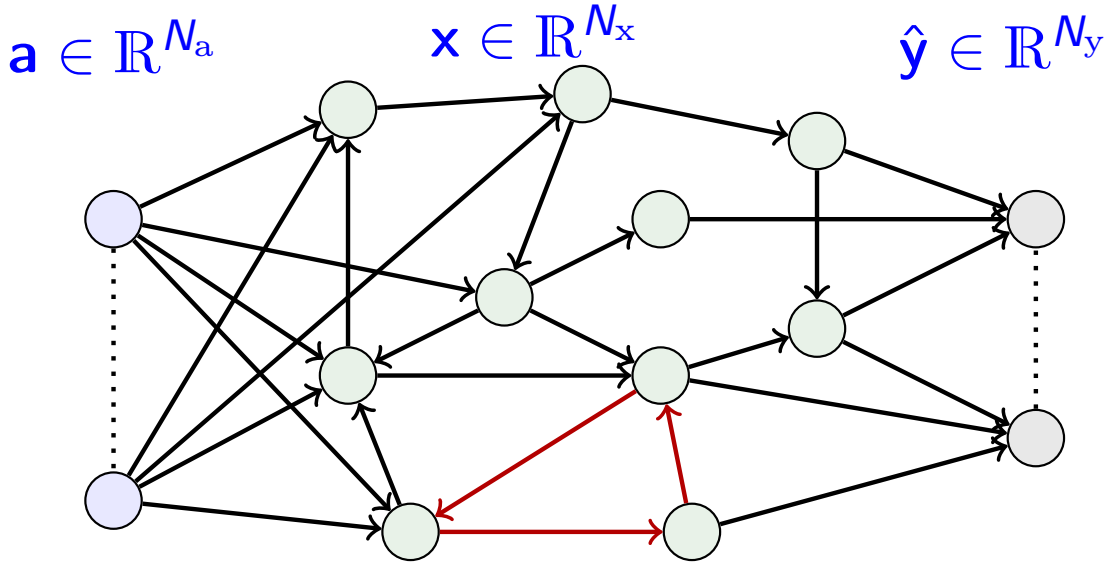
Definitions

The ESN formalisation:

- An untrained recurrent part called *reservoir*.
- A memory-less supervised learning tool called *readout*.

Three-layered recurrent neural networks:

- Input layer
- Hidden layer
- Output layer

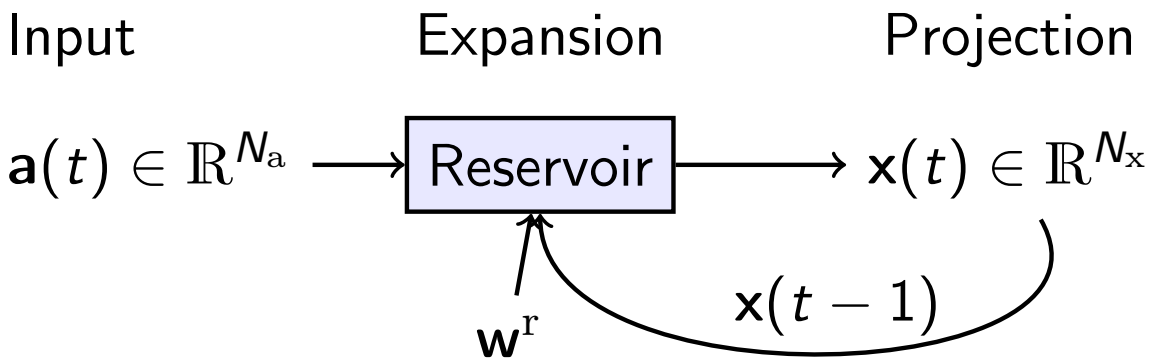


The learning process is restricted to the output weights (readout).

Echo State Networks (ESNs)

ESNs as dynamical systems

- Expansion projection ($N_{\mathrm{a}} \ll N_{\mathrm{x}}$):

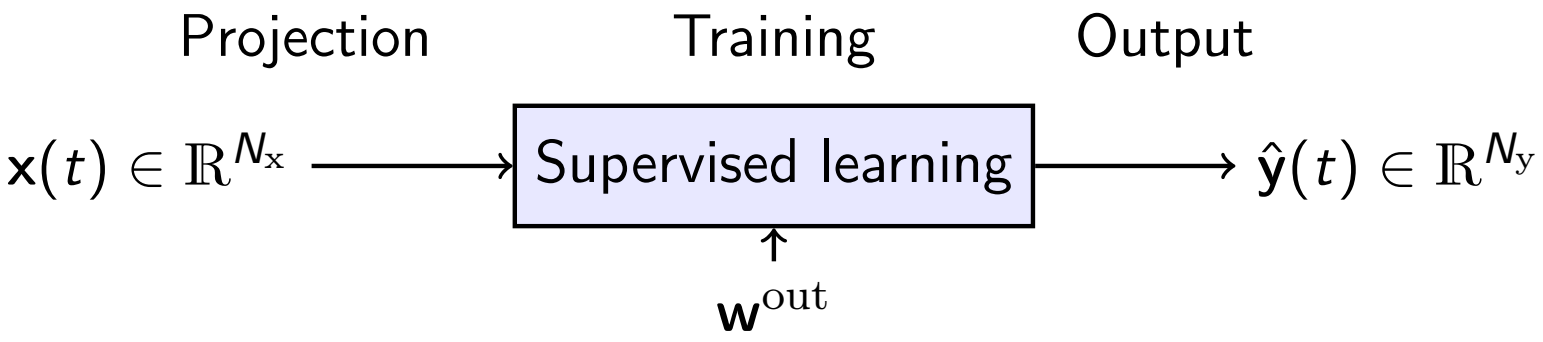


- The recurrent state of the network at any time t is given by:

$$\mathbf{x}(t) = (x_1(t), \ldots, x_{N_{\mathrm{x}}}(t)) = T(\mathbf{x}(t-1), \mathbf{a}(t), \mathbf{w}^{\mathrm{r}}), \qquad (1)$$

for some activation function T.

- The expansion can enhance the linear separability of the input information.
- Readout structure:



- The training process is restricted only to the parameters in the readout structure.

For instance,

- The recurrent state (reservoir) of an ESN at any time t is given by
$$\mathbf{x}(t) = \tanh\left(\mathbf{w}^{\mathrm{in}}\mathbf{a}(t) + \mathbf{w}^{\mathrm{r}}\mathbf{x}(t-1)\right). \tag{2}$$
- The output of the model can be a linear structure:
$$\hat{\mathbf{y}}(t) = \mathbf{x}(t)^T\mathbf{w}^{\mathrm{out}}. \tag{3}$$
- Only the $\mathbf{w}^{\mathrm{out}}$ weights are updated in the training process (it can be used gradient descent methods, regressions, ...).
- The other weights ($\mathbf{w}^{\mathrm{r}}$ and $\mathbf{w}^{\mathrm{in}}$) are random initialised, although they should satisfy some algebraic conditions (see later on this presentation).

oo

ESN properties

Some aspects of the ESN model:

- The iterative computation of $\mathbf{x}(t)$ can be unstable due to the recurrences.
- The stability behaviour is controlled by the spectral radius of $\mathbf{w}^{\mathrm{r}}$ (here we denote it by $sr(\mathbf{w}^{\mathrm{r}})$).
- In addition, the spectral radius has some impact on the memory capability of the model.
- The *Echo State Property* (ESP): an ESN has the ESP if for $t \gg 1$, $x(t)$ basically depends on the input history only (and not on how the reservoir was built).

oo

Stability of the recurrences in an ESN model:

- We have denoted by $sr(A)$ the *spectral radius* of a matrix A, and let $\eta(A)$ be the largest *singular value* of A.
- There are two well analysed situations for the case of ESN model:
  - *Sufficient condition*: if $\eta(\mathbf{w}^{\mathrm{r}}) < 1$, then ESP is satisfied.
  - *Necessary condition*: it is necessary that $sr(\mathbf{w}^{\mathrm{r}}) \leq 1$ in order to have ESP.
- The sufficient condition below is rather conservative.

• M.Lukoševičius and H.Jaeger, "Reservoir Computing Approaches to Recurrent Neural Network Training," *Computer Science Review*, vol. 3, pp. 127-149, 2009.

• B. Zhang, D. J. Miller, and Y. Wang, "Nonlinear System Modeling with Random Matrices: Echo State Networks Revisited," *Neural Networks*, vol. 76, pp. 39-45, 2016.

Reservoir Computing Models

The Echo State Network

Stability of the recurrences in an ESN model:

- A simple procedure for creating an ESN is to randomly initialise the reservoir matrix $\mathbf{w}^{\mathrm{r}}_{\mathrm{initial}}$ and then to scale it using a factor α: $\mathbf{w}^{\mathrm{r}} = \alpha \mathbf{w}^{\mathrm{r}}_{\mathrm{initial}}$.
- The selection of the scaling factor is important for the ESP.
- To use the sufficient condition can be very conservative; it can also produce a negative impact on the long memory capacity of the reservoir.
- In practice, the reservoir weights are scaled such that $\alpha < sr(\mathbf{w}^{\mathrm{r}})^{-1}$.

oo

Tuning ESNs

Main adjustable parameters of a RC model:

- Number of neurones: influences in the accuracy (larger networks increases the accuracy).

less accuracy — more accuracy

$$- \longleftrightarrow +$$

size

- Spectral radius of the weight matrix is a parameter of *memorization capabilities*.

less memory — more memory

$$0 \longleftrightarrow 1$$

$sr(\mathbf{w}^{\mathrm{r}})$

- Activation function: $\tanh(\cdot)$, linear function, Leaky Integrate and Fire (LIF), Radial Basis Function (RBF), etc.

Outline

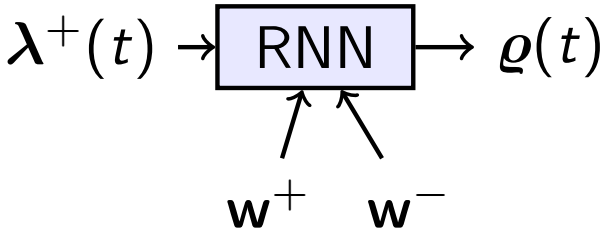
1. The PSQA project
2. Reservoir Computing models
3. ESQN
4. Empirical results
5. Some conclusions

A G-Network as a RC model

Echo State Queueing Networks (ESQN): the design of the reservoir uses ideas arising from the G-Network model.

G-Network or RNN

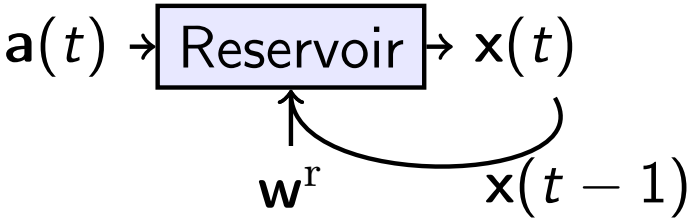
- $\varrho(t) = \nu_1(\boldsymbol{\lambda}^+(t), \boldsymbol{\lambda}^-(t), \mathbf{w}^+, \mathbf{w}^-)$,



$(\boldsymbol{\lambda}^+(t) = \mathbf{a}(t), \boldsymbol{\lambda}^-(t) = 0)$

RC model

- $\mathbf{x}(t) = \nu_2(\mathbf{x}(t-1), \mathbf{a}(t), \mathbf{w}^{\mathrm{r}})$,



Approach started in Basterrech and G. Rubino, "Echo State Queueing Network: a new Reservoir Computing learning tool," *IEEE Consumer Comm. & Networking Conf.*, Las Vegas, USA, 2013.

Echo State Queueing Networks (ESQNs)

Echo State Queueing Networks: applying the RC idea to a G-network.

- Three sets of neurones, recurrent topology "in the middle".
- Input at time t, $\mathbf{a}(t) = (a_1(t), \ldots, a_{N_\mathrm{a}}(t))$: $\varrho_u(t) = a_u(t)/r_u$, (in general, $a_u(t) < r_u$), for $u \in (1..N_\mathrm{a})$.
- For all reservoir units $u = N_\mathrm{a}+1, \ldots, N_\mathrm{a}+N_\mathrm{x}$,

$$\varrho_u(t) = \frac{\displaystyle\sum_{v=1}^{N_\mathrm{a}} \frac{a_v(t)}{r_v} w^+_{u,v} + \sum_{v=N_\mathrm{a}+1}^{N_\mathrm{a}+N_\mathrm{x}} \varrho_v(t-1) w^+_{u,v}}{\displaystyle r_u + \sum_{v=1}^{N_\mathrm{a}} \frac{a_v(t)}{r_v} w^-_{u,v} + \sum_{v=N_\mathrm{a}+1}^{N_\mathrm{a}+N_\mathrm{x}} \varrho_v(t-1) w^-_{u,v}}. \quad (4)$$

Weights notation as classical NNs: u sending spikes to v is denoted by $w^{+/-}_{v,u}$ here. These weights are frozen from the beginning. Notation r_u is for the rate of unit (neurone) u.

- The input space in then projected into a new "larger" space.
- We compute a linear regression from the projected space to the output space.
- Thus, the network output $\hat{\mathbf{y}}(t) = (\hat{y}_1(t), \ldots, \hat{y}_{N_\mathrm{b}}(t))$ is computed for any $m \in [1..N_\mathrm{b}]$:

$$y_m(t) = w_{m,0}^{\mathrm{out}} + \sum_{i=1+N_\mathrm{a}}^{N_\mathrm{a}+N_\mathrm{x}} w_{m,i}^{\mathrm{out}} \varrho_i(t). \tag{5}$$

- The output weights $\mathbf{w}^{\mathrm{out}}$ can be computed using, for instance, a Regression. Remark: we can replace this simple structure by, for instance, a classical feedforward 3-level RNN (to be explored).
- Details in Basterrech and G. Rubino, "Echo State Queueing Networks: a combination of Reservoir Computing and Random Neural Networks", *Prob. in the Eng. and Informational Sciences, 2017*.

Outline

1. The PSQA project
2. Reservoir Computing models
3. ESQN
4. Empirical results
5. Some conclusions

ESQN

Examples: simulated dataset

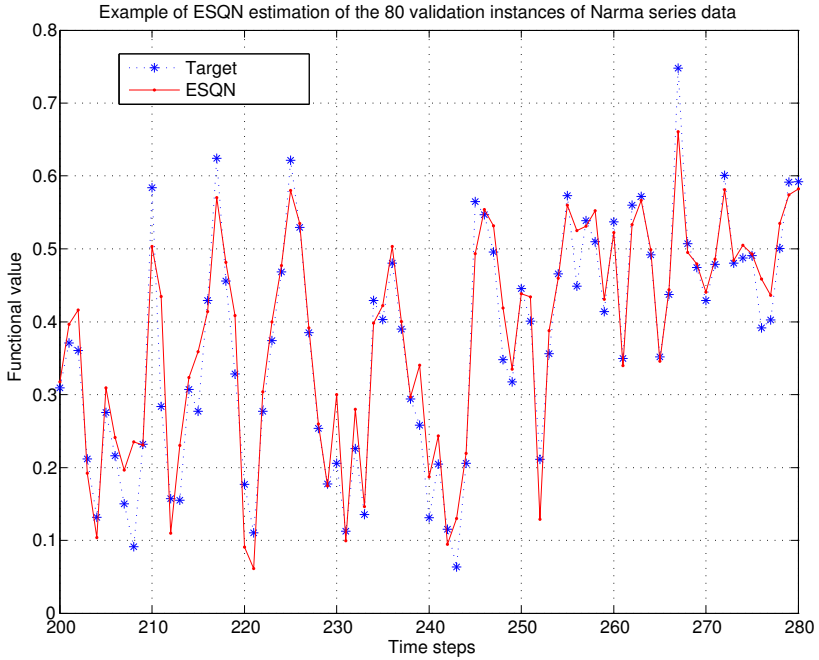


Figure: NARMA validation data set. The reservoir was randomly initialized and it had 80 units.

NARMA is a simulated dataset that is often analyzed in the RC literature.

ESQN

Examples: Internet traffic prediction

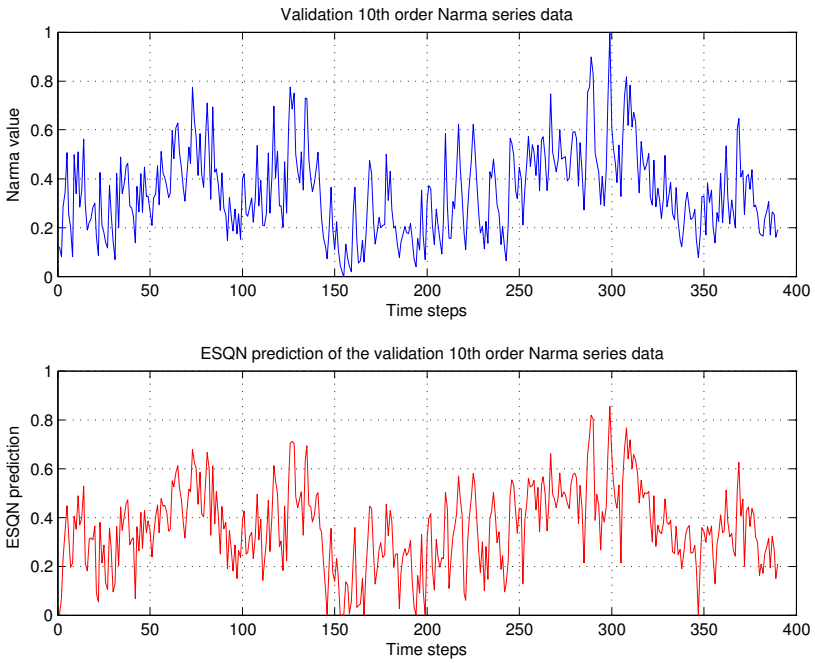


Figure: Fixed 10th order NARMA times series. Comparison between the original dataset and the forecasted samples on the validation dataset.

ESQN

Examples: Internet traffic prediction

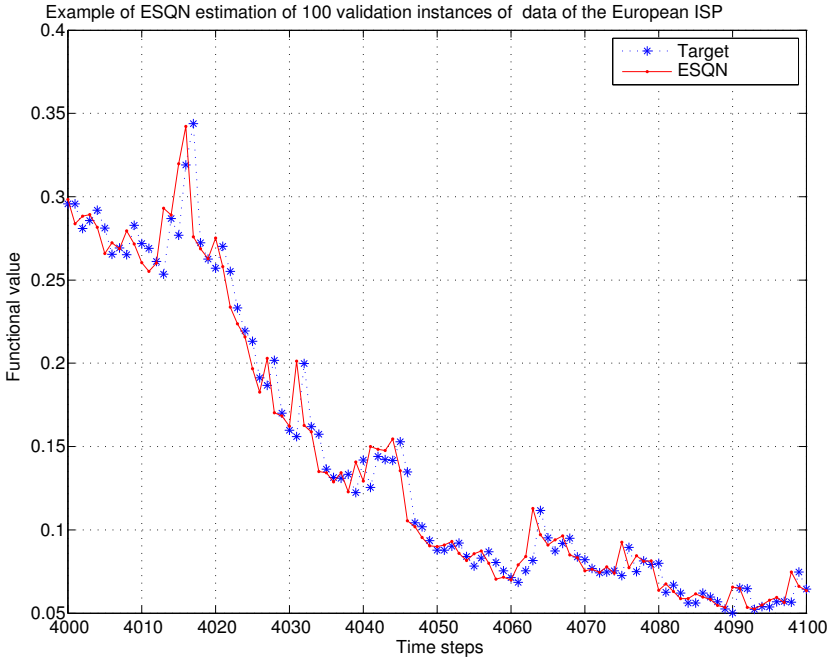


Figure: Normalized Internet traffic data from an European Internet Service Provider. The reservoir has 40 neurones and it was randomly initialized.

ESQN

Examples: Internet traffic prediction

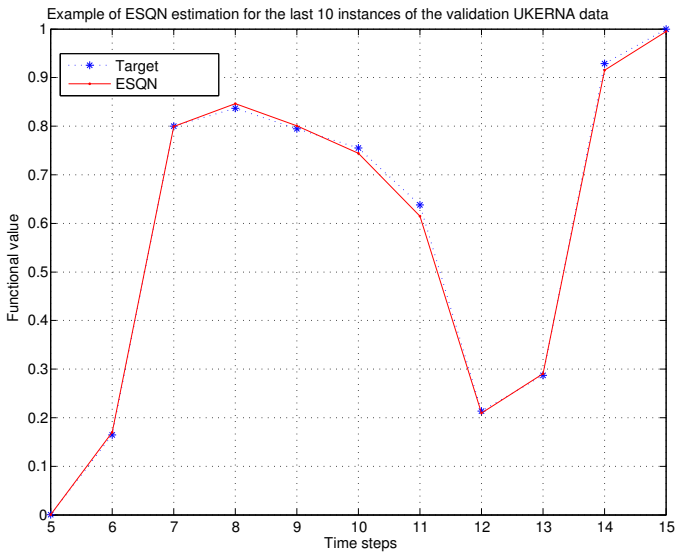


Figure: ESQN estimation for UKERNA validation data set. The reservoir weights were randomly initialized. The reservoir size is 40.

ESQN

Examples: Internet traffic prediction

We can see that ESN is more stable than ESQN:

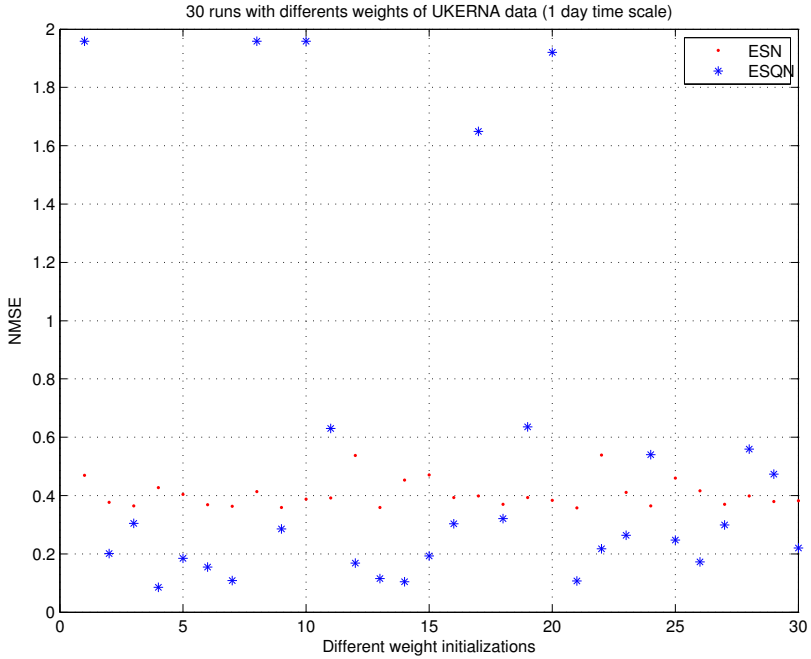


Figure: ESQN and ESN model accuracy (NMSE) for different reservoir initializations for the Intenet traffic prediction (UKERNA validation data set).

ESQN

Sensitivity of the global parameters.

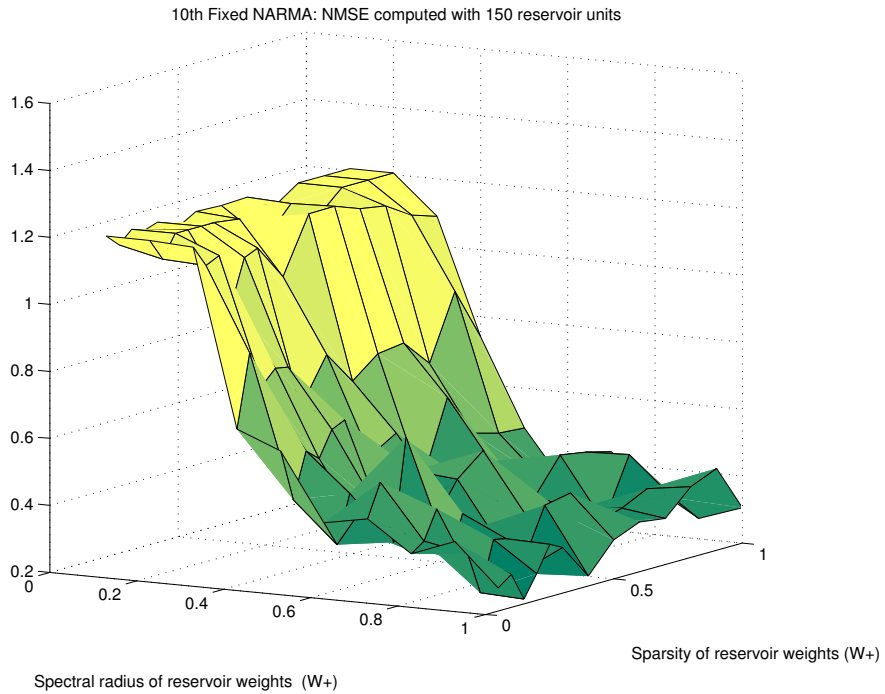


Figure: Simulated dataset. The NMSE according to the spectral radius and the sparsity of the reservoir $\mathbf{w}^+$.

ESQN

Sensitivity of the global parameters.

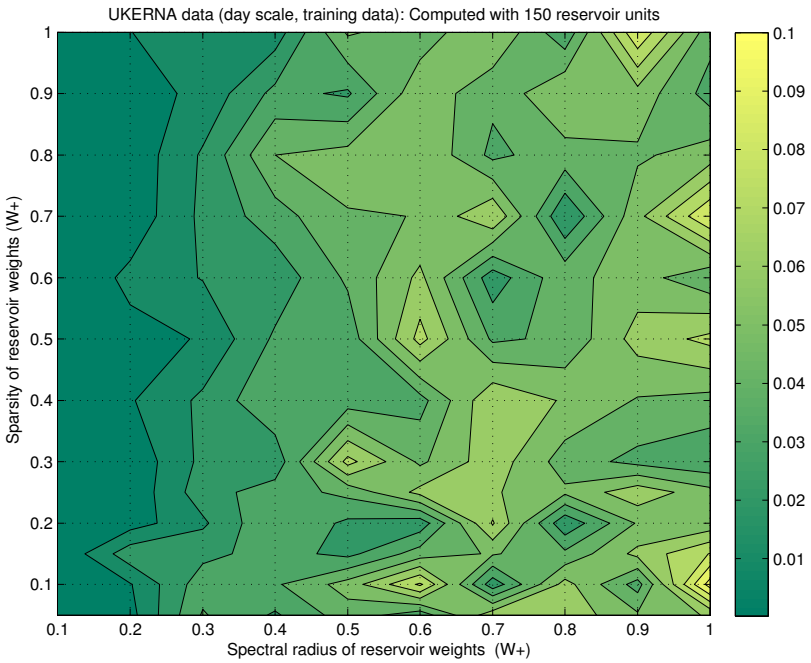


Figure: Internet traffic prediction. The NMSE according to the spectral radius and the sparsity of the reservoir of $\mathbf{w}^{-}$.

ESQN

Sensitivity of the global parameters.

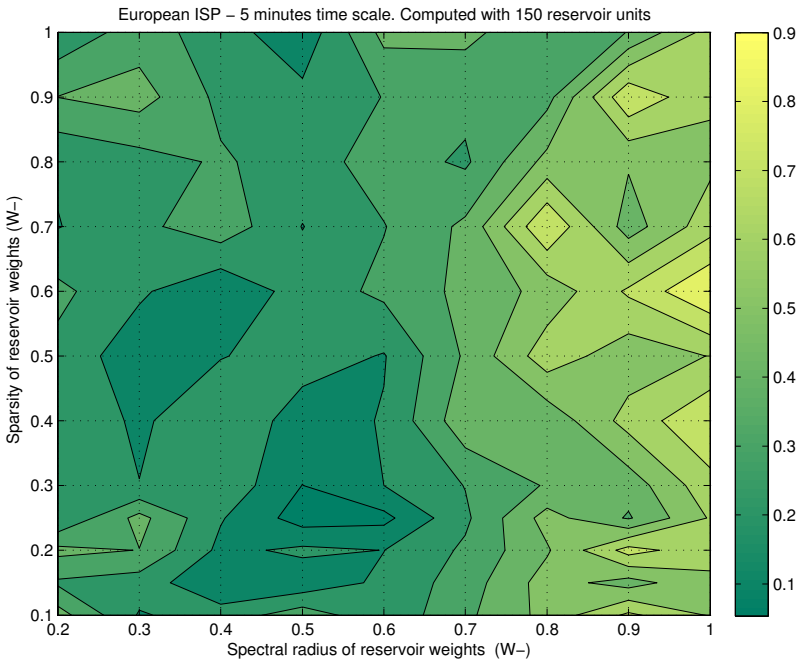


Figure: Internet traffic prediction. The NMSE according to the spectral radius and the sparsity of the reservoir of $\mathbf{w}^{-}$.

ESQN

Sensitivity of the global parameters.

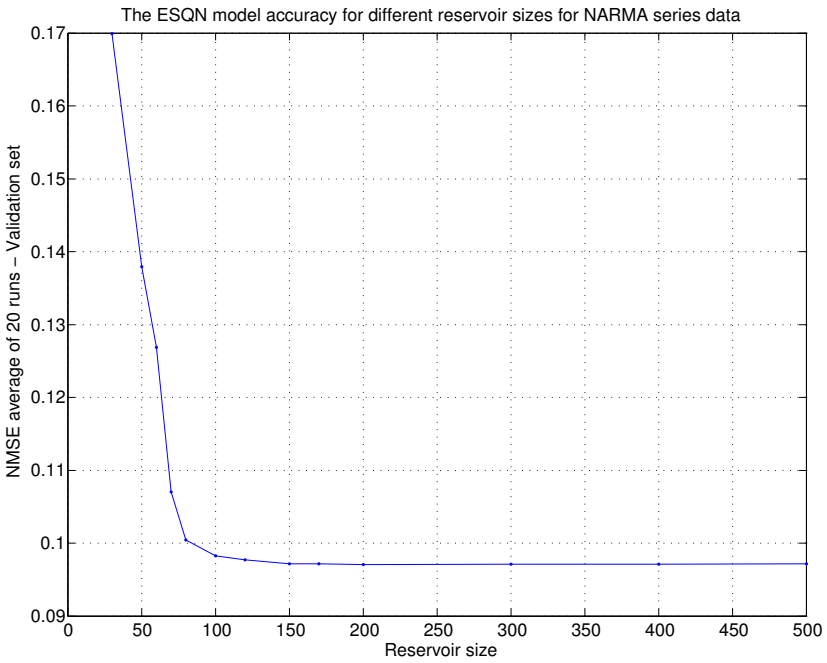
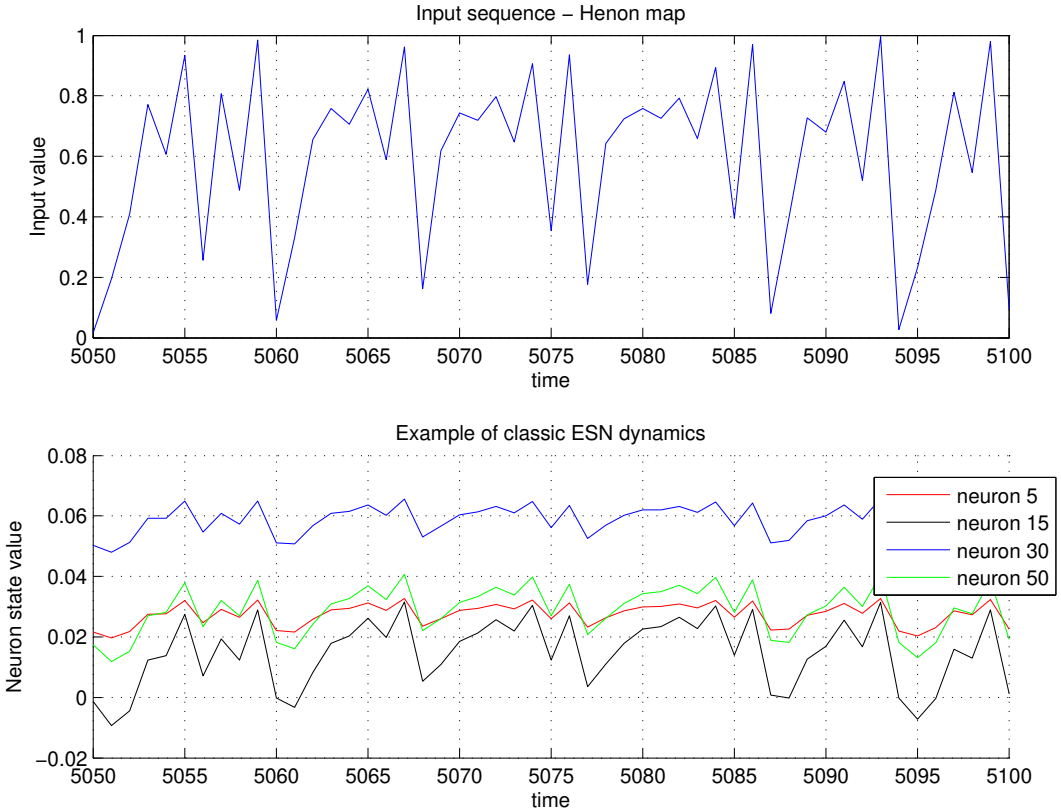


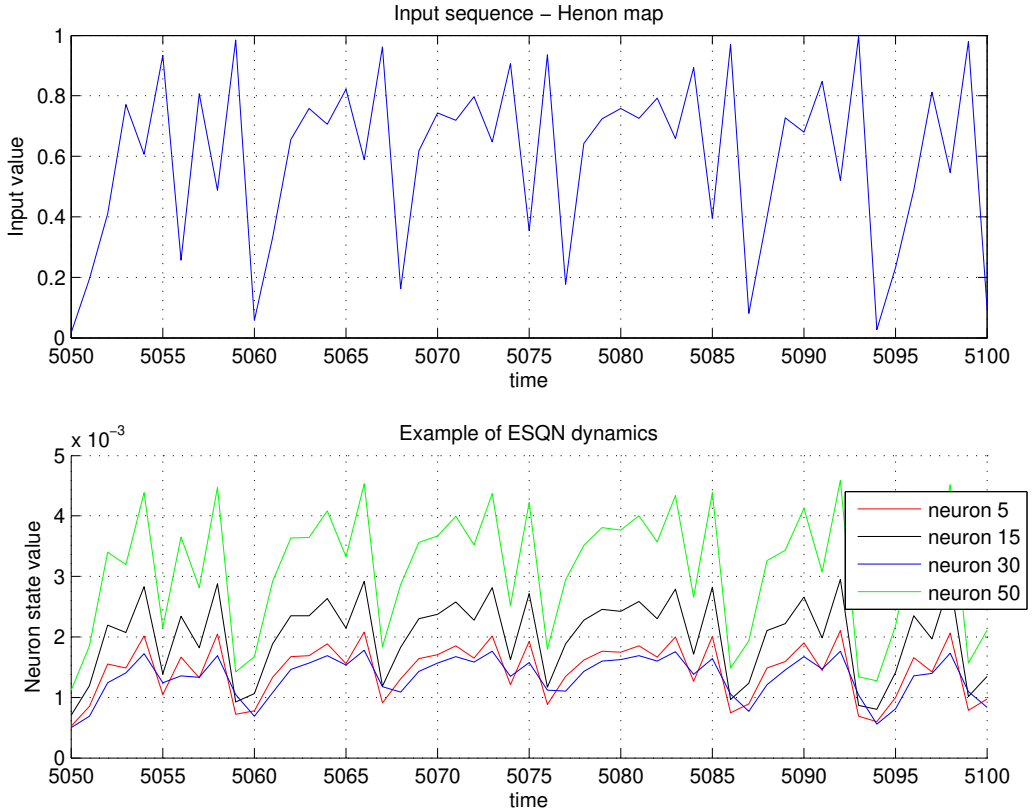
Figure: The ESQN model performance for different reservoir sizes which are computed for 10th NARMA validation data set. The reservoir weights were randomly initialized. Figure shows the NMSE average achieved in 20 runs with different ESQN initial weights.

Echo State Queueing Network



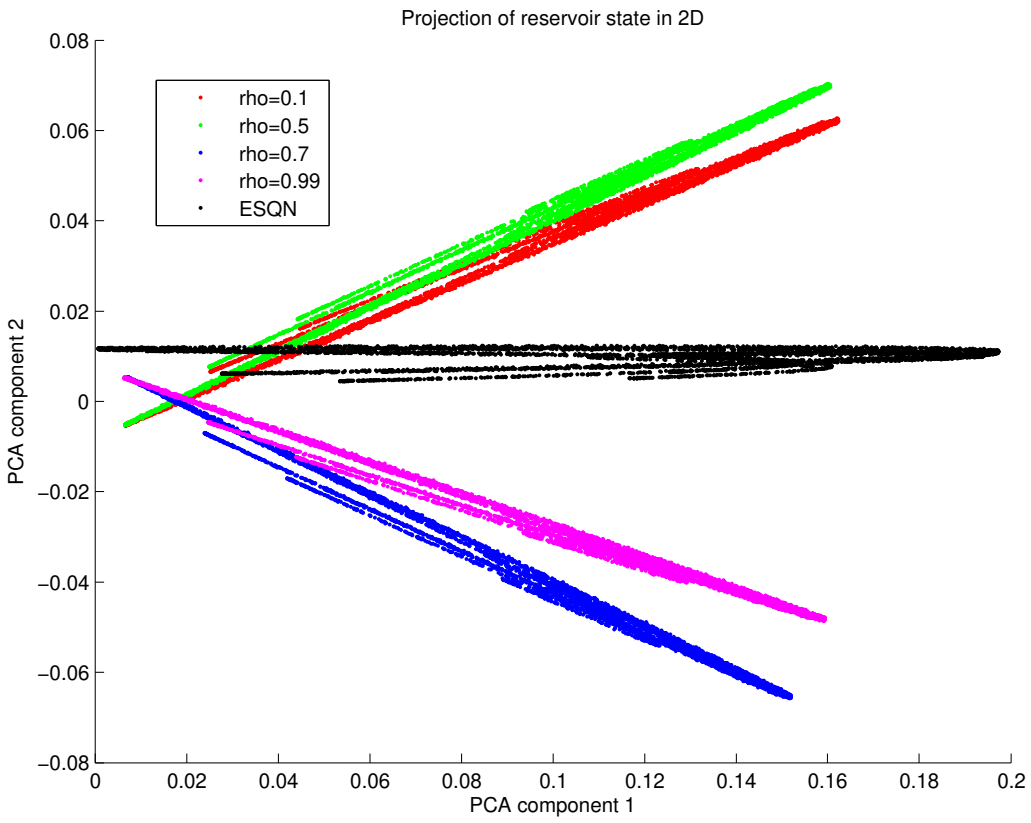
Note: Henon Map as input data. ESN has 50 fully connected reservoir neurons and spectral radius (ρ) equal to 0.5.

Echo State Queueing Network



Note: Henon Map as input data. ESQN has 50 fully connected reservoir neurons with weights randomly fixed between $[0, 0.1]$.

Echo State Queueing Network

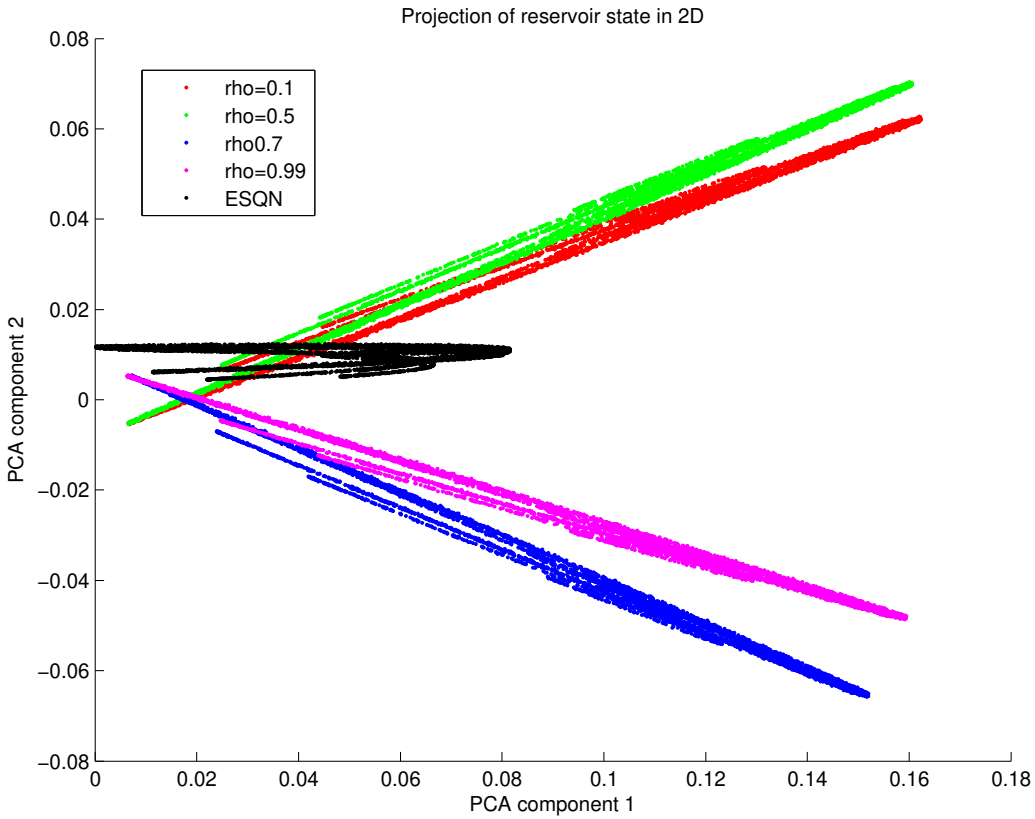


Comments

Note about the previous graphic:

Figure shows a projection of the sequence of reservoir states using PCA. The spectral radius of the reservoir matrix (in the graphics was mentioned as ρ, is the usual notation). This is an important global parameter of the model. It is used for controlling the memory capacity of the model, and it controls the stability of the recurrent dynamics. The graphic illustrates the different behavior of the ESN model according to the spectral radius, and also compares with the reservoir states of the ESQN. There are four collections of points that correspond to the projection of the reservoir state of different ESN models. The collection of black dots corresponds to the projection of the ESQN reservoir state to 2D. The weights of the ESQN were randomly fixed in $[0, 0.1]$.

Echo State Queueing Network



Comments

This figure is similar than previous graphic, but in this case the reservoir of the ESQN has weights in $[0, 0.25]$. The graphic shows how the ESQN dynamics depend in the initial values assigned to the ESQN reservoir. It is visible that the dynamics are very different for values in $[0, 0.1]$ and values in $[0, 0.25]$. Probably, the interval also depends on the number of neurons. As a consequence, how to fix the reservoir can affect the results. The procedure for fixing the initial weights of the ESQN reservoir have been only empirical in our case, we are trying to figure out a general automatic procedure.

Outline

1. The PSQA project
2. Reservoir Computing models
3. ESQN
4. Empirical results
5. Some conclusions

Summary

What we have done:

- The ESQN proposal.
- Some preliminary experimental work, to explore its behaviour, based on similar work done (by many people) for the ESN model.
- So far, ESQNs behave very well, most often better than worse than standard ESNs.
- We use a basic implementation of ESQN, which is compared to fine tuned code used in ESNs.
- We also started to explore the performances of the model and the role played but some potentially important parameters.

On some of the parameters:

- The impact of the reservoir size is similar to the impact of the number of neurones in a classical network.
- The spectral radius behave in ESQNs as in ESNs: close to 1 is fine for problems with long range data correlation needing long memory capabilities, and close to 0 it is ok in the opposite case.
- The impact of the sparsity of the reservoir is unclear. Typical recommendations for ESNs are about 15%-20% and we found that this seems also appropriate for ESQNs.

Concluding remarks

- The analysis of the RC approach is still fairly open, and the situation is similar for the ESQN model.
- There are several important parameters to adjust: reservoir's size, reservoir's connections, reservoir's weights, then, spectral radius of the reservoir matrix.
- ESNs and ESQNs work very well in many cases for time series modelling.
  - This good behaviour is not well understood.
  - Both ESNs and ESQNs require some expertise for setting their parameters.
- We would like to find conditions for the initialisation of the weights in order of guaranteeing good predictions (something such as ESP for ESNs).
- A good (optimal?) structure for the readout is to be found.

THANK YOU!!!

Contacts:

- e-mail: Gerardo.Rubino@inria.fr
- e-mail: Sebastian.Basterrech.@gmail.com