Machine Learning Tools for Forecasting Correlated Time Series (keynote)

Gerardo Rubino

▶ **To cite this version:**

Gerardo Rubino. Machine Learning Tools for Forecasting Correlated Time Series (keynote). ICSADADS 2020 - International Conference on Statistical Applications in Data Analytics & Data Science, Feb 2020, Coimbatore, India. hal-03122282

HAL Id: hal-03122282
https://inria.hal.science/hal-03122282v1

Submitted on 27 Jan 2021

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.

ICSADADS 2020

Machine Learning Tools for Forecasting Correlated Time Series

G. Rubino

INRIA Rennes, France

Coimbatore, Feb. 21, 2020

Introduction

- Among the multiples faces of AI (methods, applications, tools, ...) we will focus today on
  - the problem of predicting time series,
  - a main application around video or audio transmission over the Internet,
  - our own tools in these areas,
  - some recent trends and relevant techniques.
- More specifically, we will describe
  - the problem of measuring the Perceptual Quality,
  - the more difficult problem of predicting it,
  - our AI-based tools and solutions,
  - some recent interesting related technologies, and our current projects.
- Main keywords:
  supervised learning, time series prediction, recurrent neural networks;
  Perceptual Quality and other networking problems, PSQA;
  Random Neurons, ESQNs.

Outline

Outline

Context

- Consider any kind of application or service centered on transporting audio and/or video signals over the Internet.
- This means a very large part of Internet traffic in volume.
- 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, or to losses, external perturbation agents such as interferences in some networking technologies, etc.) transmission can suffer from content degradation.

Perceptual quality

- Perceptual quality refers to the (subjective) perception (the view, the feeling) the user has on the "value" 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.
- Critical problem (solved): how to measure it? Nad then, how to measure it automatically?

Perceptual quality (cont.)

- Observe that somehow perceptual quality is at the heart of the designer goals for these systems, it is her *ultimate target*, complemented by some other aspects of system's properties (e.g. security).
- When designing any system of the before mentioned kind (dominating today's Internet traffic), our message is
  - instead of tuning the control parameters in order to reduce the mean response time, or to maximize the system's throughput,
  - tune them to maximize quality itself.

Subjective testing

- Subjective testing is the standard way of measuring perceived quality.
- A panel of appropriately chosen human observers is built, and a set of media sequences is shown to the panel members.
- Following specific rules (absolute rating, comparisons...) the observers say how they perceive the quality of the sequences.
- In case of interactive applications, observers can work by pairs, or in some cases, interact with a robot.

Subjective testing (cont.)

- The result is a table M where component $M_{h,\sigma}$ is the value given by human h to sequence σ.
- A statistical procedure is then followed to filter these results in order to remove outliers.
- At the end, if the set of surviving observers is $\mathcal{S}$, the quality of sequence σ, its *MOS* (Mean Opinion Score) value is

$$MOS(\sigma) = \frac{1}{|\mathcal{S}|} \sum_{h \in \mathcal{S}} M_{h,\sigma}.$$

- The procedure is surprisingly robust: with the same set of sequences but a different panel, the same subjective test gives results extremely close to the first one: if μ_σ and μ'_σ are the MOS values of sequence σ in both panels, the number $\sum_\sigma (\mu_\sigma - \mu'_\sigma)^2$ is pretty "small".

Subjective testing (cont.)

- For each type of media, there are norms specifying how to perform the corresponding subjective test. A reference here is the production of the ITU (International Telecommunication Union).
- For instance, some norms ask users to watch video sequences potentially degraded by some noisy effect, and rate them from 1 to 5 according to the following table:

MOS score	description
5	degradation imperceptible
4	degradation perceptible but not annoying
3	degradation slightly annoying
2	degradation annoying
1	degradation very annoying

Subjective testing (cont.)

- The norms specify also other constraints concerning details such as the lengths of the sequences, the timing of the subjective testing session, its own length, the experimental conditions (e.g. monitor contrast or viewing distance in case of video), the suggested panel's size, etc.
- In video, most subjective tests norms belong to the following classes:
  - Single Stimulus (SS): only the distorted signal is shown to the panel,
  - Double Stimulus Continuous Quality Scale (DSCQS): both the original and distorted sequences are shown simultaneously
  - Double Stimulus Impairment Scale (DSIS): first the original sequence and then the distorted one are shown.

Subjective testing (cont.)

- The panel must also be built following some rules (similar to those followed to make opinion polls). The idea is to build a representative sample of the users of the considered application/service, or of the subfamily of users we are interested in.
- Obviously, subjective testing is costly, takes time, and by construction is not usable for real time perceived quality assessment.
- Comment: one of the goals of my team at Inria is to use quality assessment techniques for monitoring and for control purposes. So, only methods that work automatically qualify.

Objective testing

- Objective testing means evaluating the perceived quality automatically, without using any panel of users.
- To give an example, consider video sequences. Some objective testing methods (coming from the coding area) are based on the PNSR (Peak Signal To Noise Ratio) of two images A and B:
  - for instance, assume the two images are composed of M rows and N columns of 8-bits pixels (monochrome case).
  - The MSE (Mean Squared Error) between the two images is

$$\mathrm{MSE}(A,B) = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\Big(A_{i,j} - B_{i,j}\Big)^2 \in [0..255^2].$$

  - The PSNR between the two images is defined only if they are different ($\mathrm{MSE} \neq 0$) and its value (in dB) is

$$\mathrm{PSNR}_{\mathrm{dB}}(A,B) = 10\log_{10}\Big(\frac{255^2}{\mathrm{MSE}(A,B)}\Big).$$

Problems with PSNR approach

Noise in the sky

Problems with PSNR approach...

Same amount of noise in the grass

Objective testing (cont.)

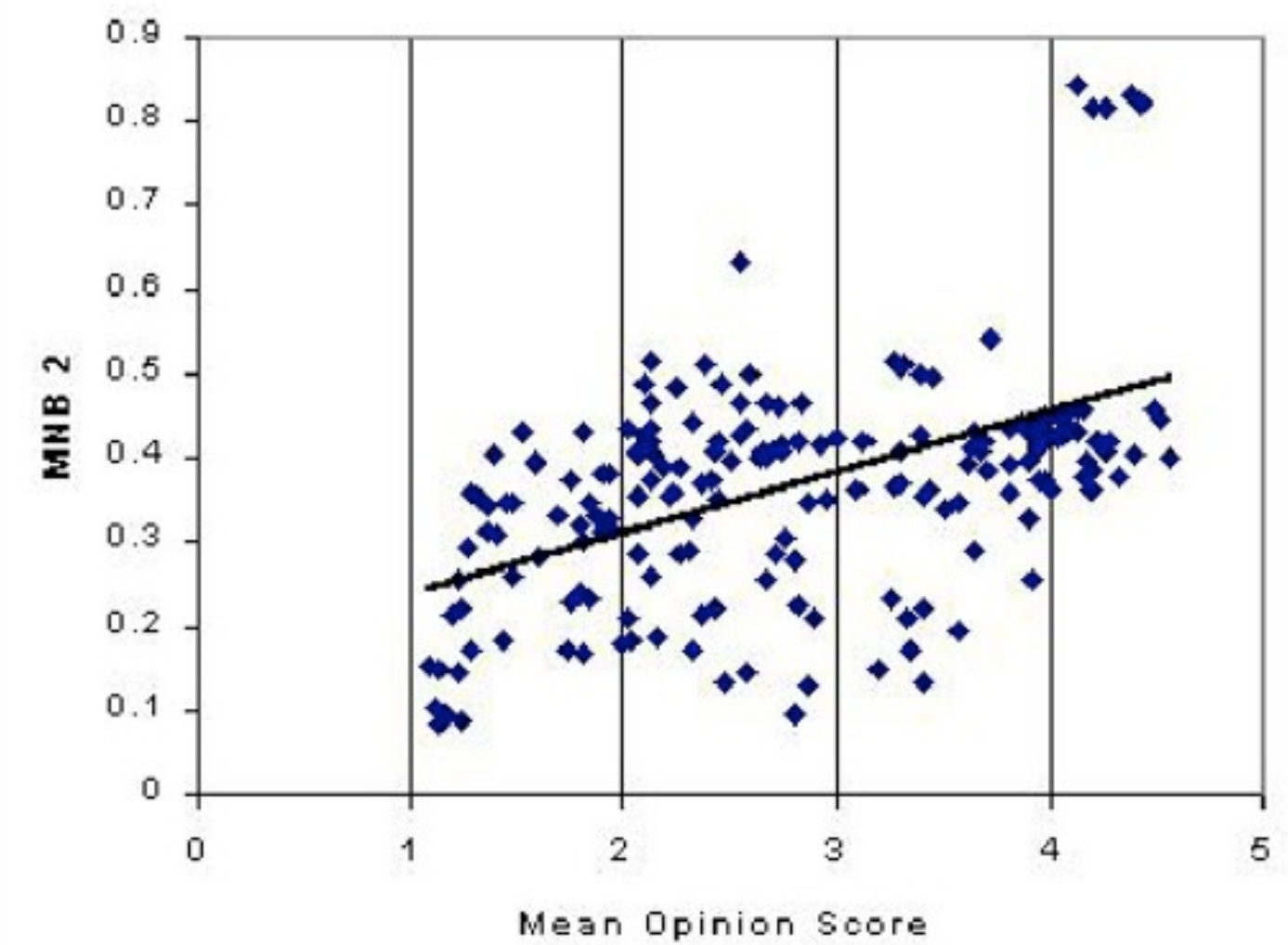
- We can classify objective testing in three categories:
  - *full reference* methods: the original sequence is needed;
  - *partial reference* methods: some values related to the original sequence are needed;
  - *no reference* methods: the original sequence is not needed.
- Some objective testing techniques are said to be *signal-based*: we must explore the content of the sequence to assess its quality.
- Other methods are said to be *parametric*: the quality is evaluated based on some parameters related to the system (source parameters and/or network parameters).

Objective testing (cont.)

- The main advantage of objective methods is that they by-pass the use of panels of observers.
- In VoIP there are some important examples such as the E-model, and others.
- However, in general objective metrics often correlate poorly with subjective ones ("the truth").

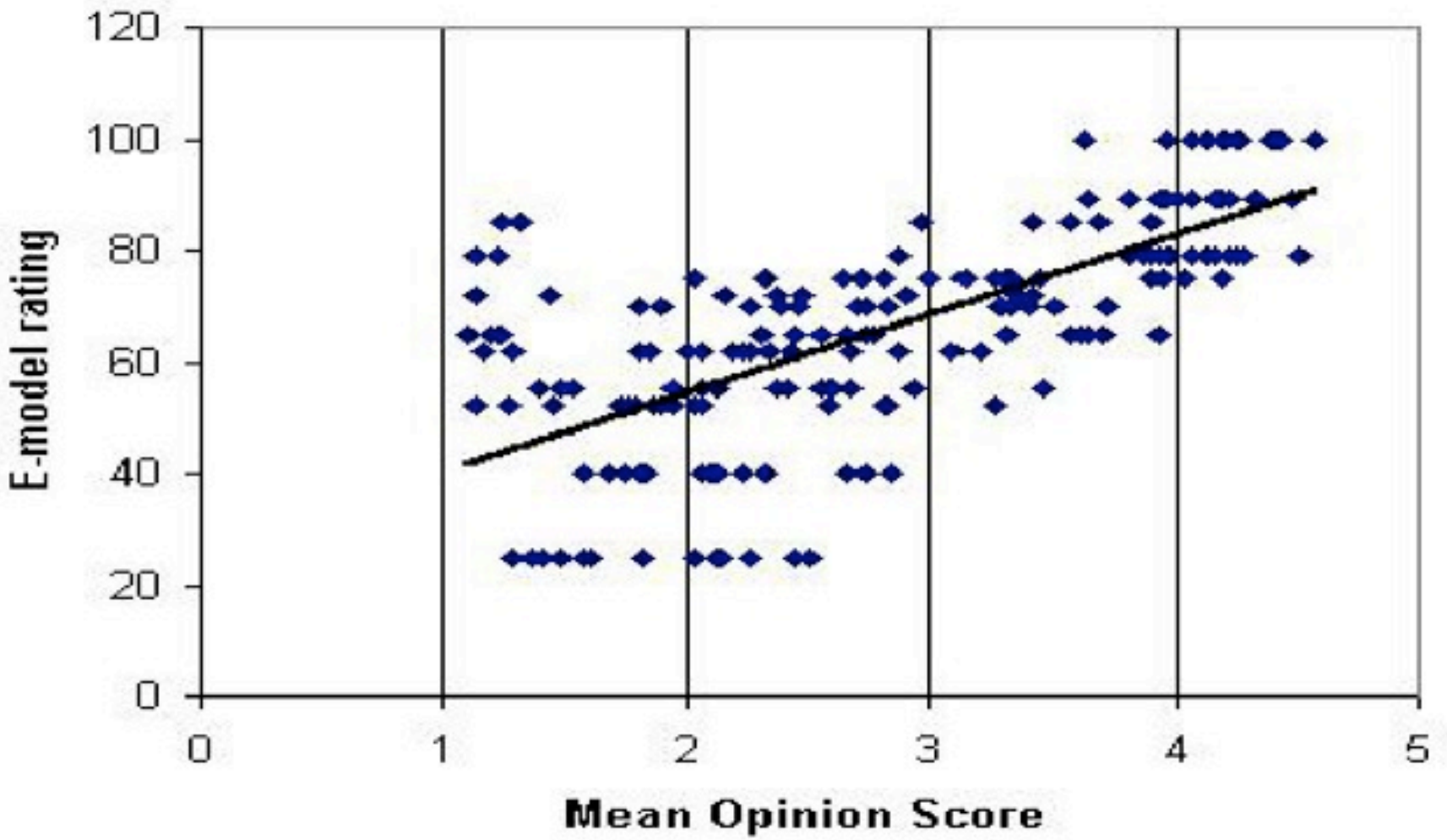
Example in audio

Bad correlation between an objective metric and subjective tests in audio.



Example in VoIP

Bad correlation between an objective metric and subjective tests in Voice over IP.



Resuming

- 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.

Outline

PSQA: Pseudo Subjective Quality Assessment

- In the Dionysos team we promote our solution called PSQA to all the initial problems and many more, for all types of media considered, and for both one-way and two-ways communications.
- PSQA is a metric with no reference, automatic, (so far, "optimally") accurate, and it works in real time if necessary or useful.
- 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 (into a MOS value).

PSQA (cont.)

- The mapping is based on statistical learning tools and it is built in such a way that it has nice mathematical properties. We can use it in many ways, for instance,
  - for sensitivity analysis: which is the dominating factor having an impact on quality? where is it significantly dominating?
  - for inverse problems: which values of the considered factors lead to a high enough MOS value?
  - etc.
- More specifically, so far PSQA functions are a particular class of rational functions coming from the Random Neural Network (or G-network) area.

PSQA (cont.)

- **Example in video** (simplified version): a PSQA monitoring module measures, at the receiver's position, the instantaneous packet loss rate LR, the Bit Rate of the connexion BR and the Frame Rate FR, and calls a PSQA function ν_1(LR,BR,FR) which provides in a few msec the perceptual quality value.
- **Example in VoIP**: a PSQA monitoring module measures, at the receiver's position, the instantaneous packet loss rate LR, the average size of a burst of loss packets MLBS, the Bit Rate of the connexion BR, and the Offset of the FEC (Forward Error Correction) OFEC, and calls a PSQA function ν_2(LR,MLBS,BR,OFEC) which provides in a few msec the perceptual quality value.

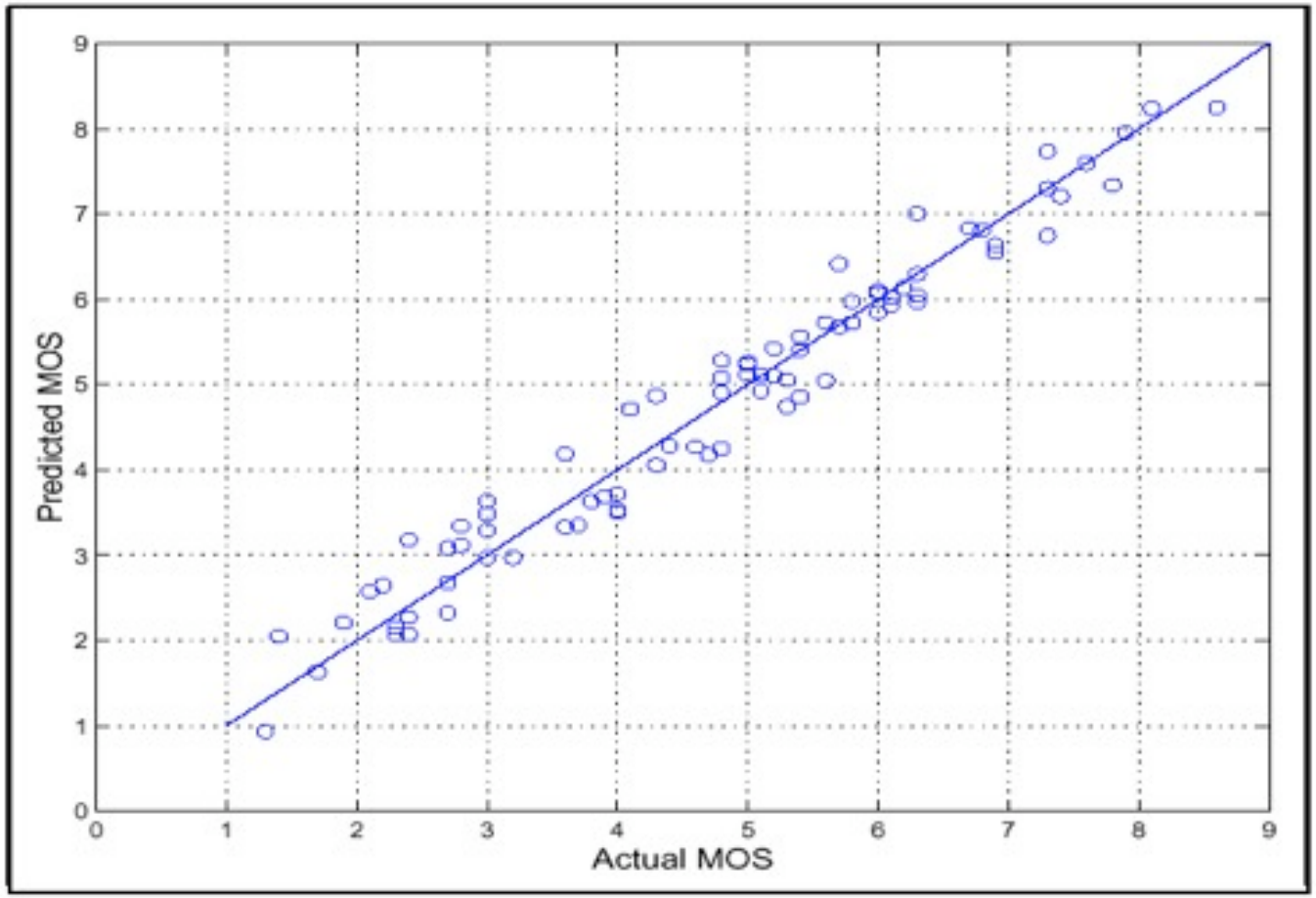
How PSQA works

- 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, (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) $OFEC$.

- 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 a few short signals (following standards) representative of the application or service target, $\sigma_1, \ldots, \sigma_K$.
- We build a pretty small set of K configurations ($K = 100, 200, \ldots$) 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 signals σ_i through a simulated or deployed network where we can simultaneously control all components of the configurations.

- Then we send different original σ_i using the platform, when the configuration is $\vec{\gamma}_j$ using one configuration at a time, and obtain a possibly degraded sequence σ'_j. We show it to a panel of humans and we obtain its PQs Q_j.
- We thus obtain a set of K 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 $\left(\vec{\gamma}_j, Q_j\right)$.
- In the example of video, we obtain a function $\nu(LR, BR, FR)$. In the VoIP example, we obtain a function $\nu(LR, MLBS, BR, OFEC)$.

Training results for PSQA (video example)

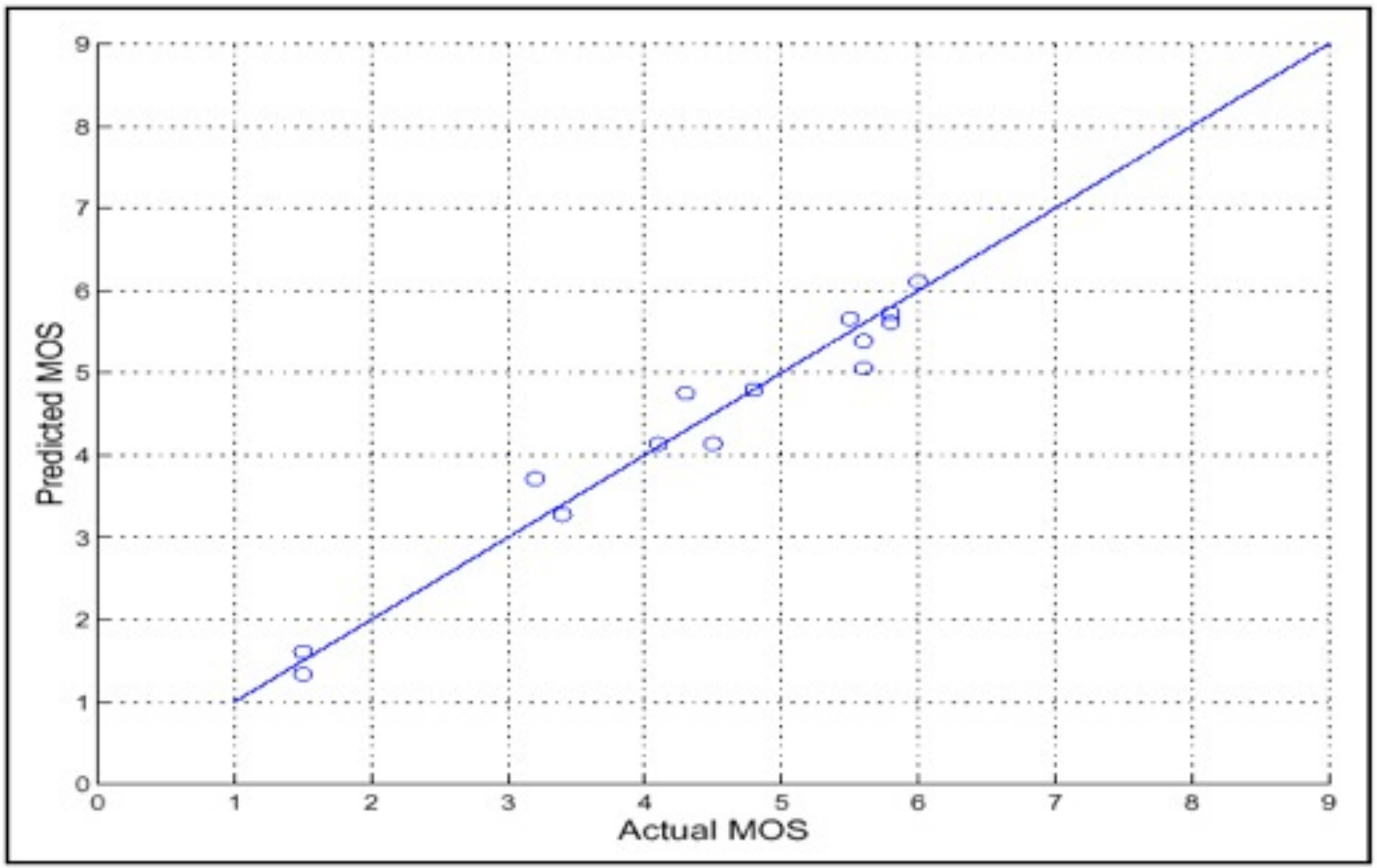


(a) Training DB

(Results are not surprising.)

Validation results for PSQA (video example)

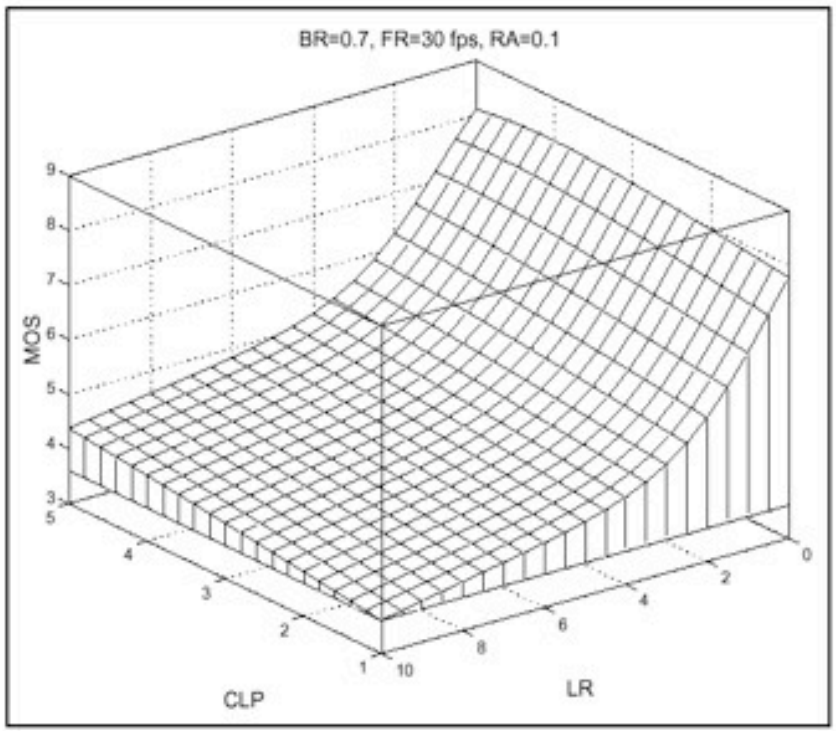
Very good accuracy of the tool:



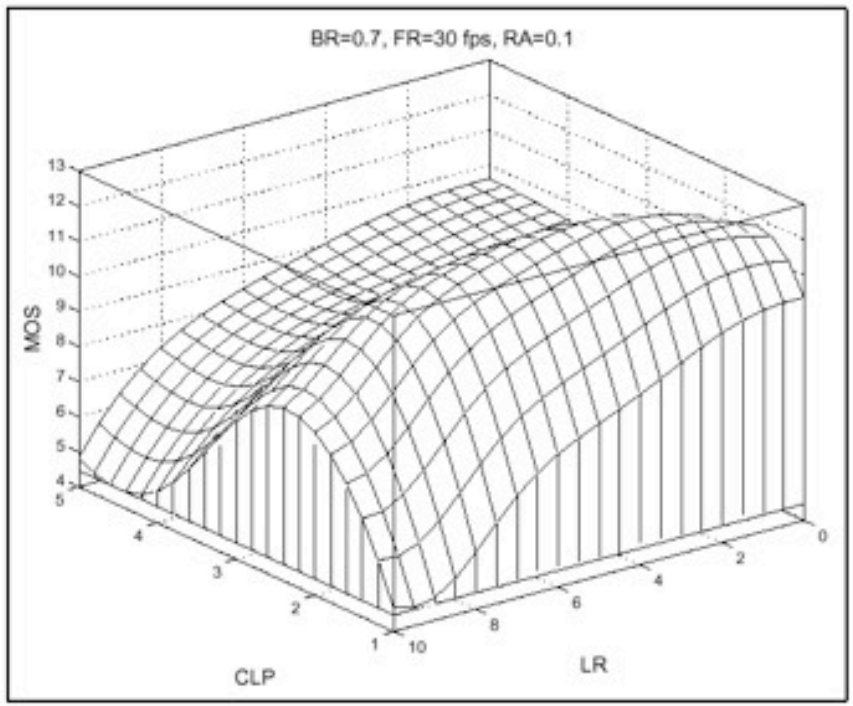
(b) Testing DB

Comparing RNN with ANN

Under exactly the same (reasonable) conditions and with the same data (an example, there are many other ones):



(a) Correctly trainined

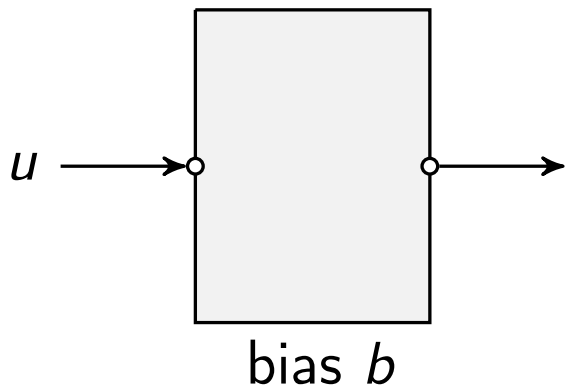


(b) Example of an over-trained ANN

Outline

Classic artificial neurons

- Let us see a "classic artificial neuron" as a parametric real function of a real variable.

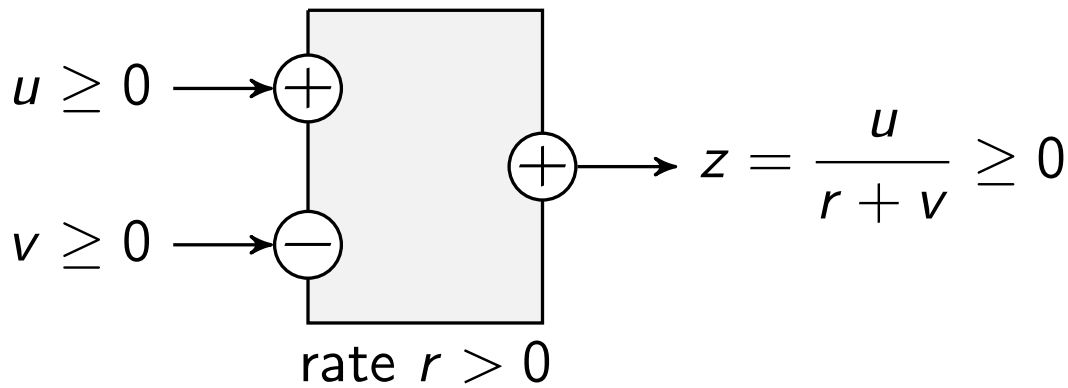


$$\text{for instance, } z = \frac{1}{1 + \mathrm{e}^{-u+b}}$$

- Consider b as a parameter of the function implemented by the neuron.
- There are many other activation functions used in practice: $z = \tanh(u - b)$, $z = 1(u \geq b)$, RELU, ...

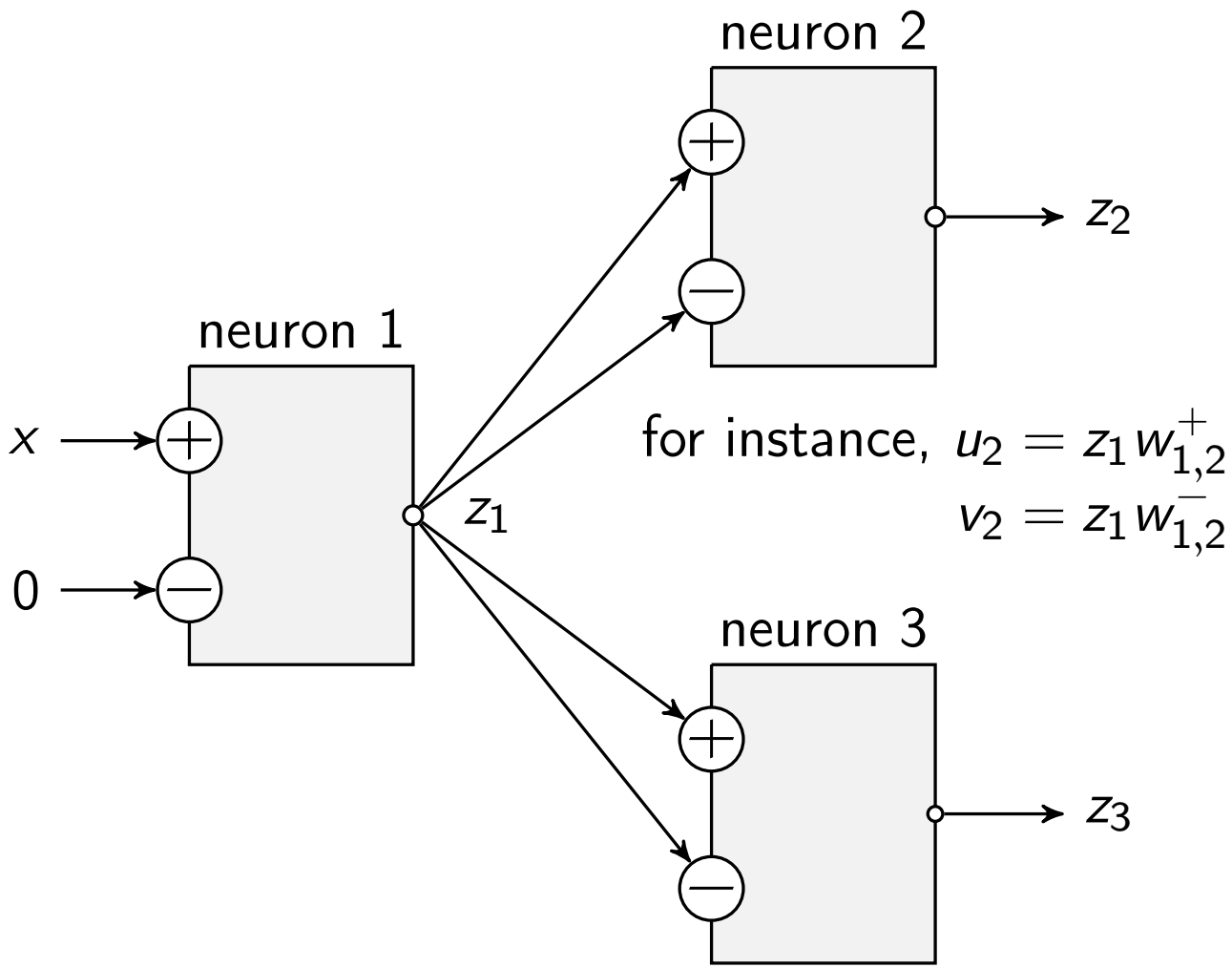
Random Neurons

- A Random Neuron is a parametric positive real function of two real positive variables (we also say "ports"), one called the "positive port", the other one being the "negative port".

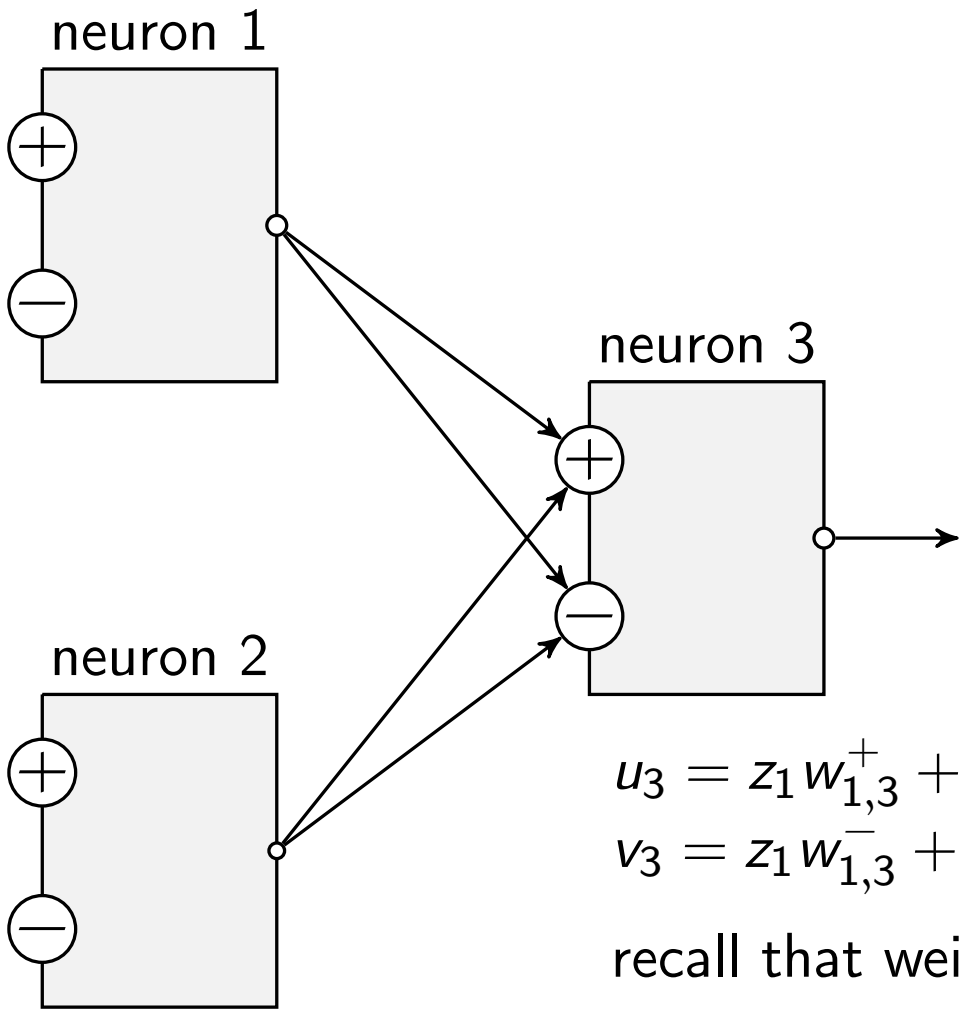


- The real $r > 0$, the *rate* of the neuron, is a parameter of the function.
- There are several variants of this model. In the main and original one, we must use $z = \min(u/(r+v),\, 1)$.
- Inventor: E. Gelenbe, Imperial College, in the late 80s.
- Observe that there is nothing random here. The name comes from the origin of the model (queueing theory).

A Random Neural Network (RNN) is a set of interconnected RNs. The connections are weighted by nonnegative reals denoted $w^+_{i,j}$ and $w^-_{i,j}$ for the weights of the connection *from i to j* arriving at the positive and negative ports respectively.



Jonctions at input ports are additive:



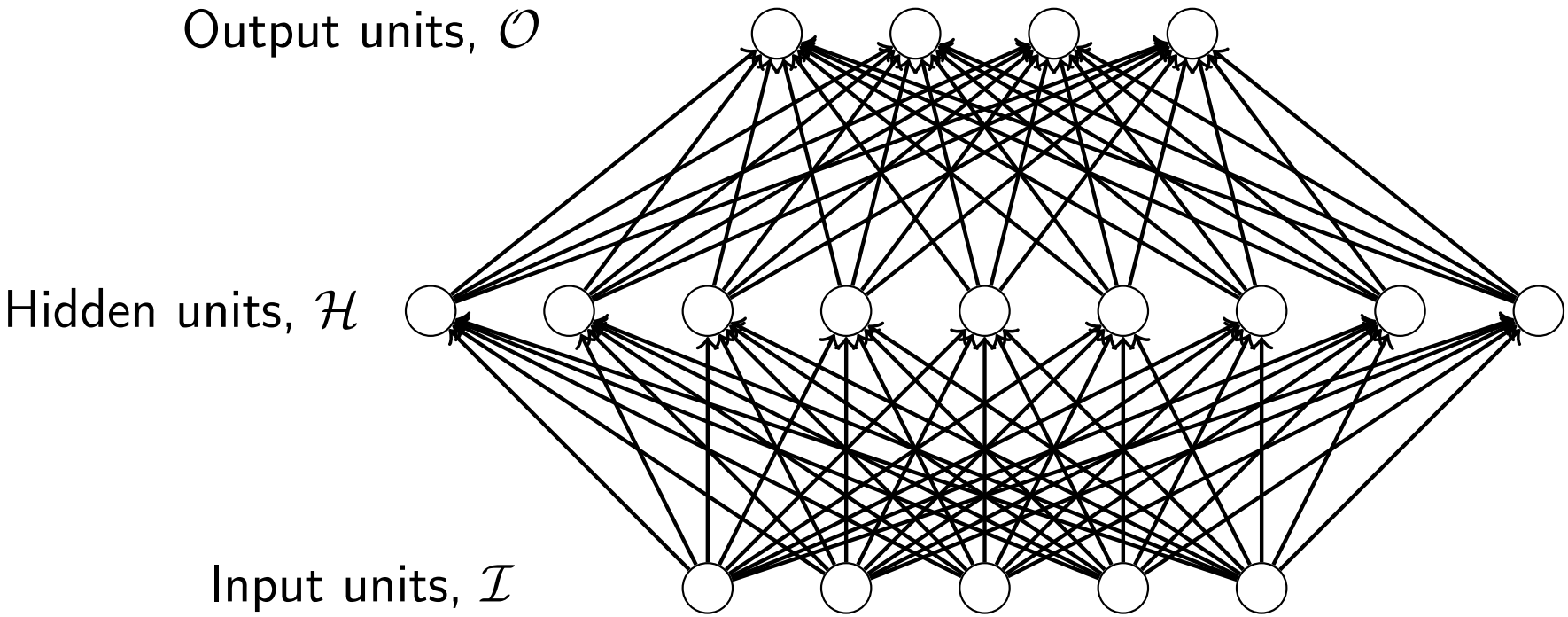
$$u_3 = z_1 w^+_{1,3} + z_2 w^+_{2,3}$$
$$v_3 = z_1 w^-_{1,3} + z_2 w^-_{2,3}$$

recall that weights $w^*_{i,j}$ are nonnegative

Feed Forward RNNs

- In this case (for instance, a simple multi-layer architecture), the output coming out of the last layer is a rational function of the inputs.
- This allows many nice mathematical processing of the trained networks.

The 3-layer Random Neural Network



Random Neural Networks implement rational functions

- Assume the negative ports of input neurons aren't used. Assume a single output neuron, so, a scalar network output.
- Call x_i the signal arriving at the positive port of input neuron i. Then, we can explicitly write the network output as a function of the inputs.

$$z_o = \frac{\sum_{h\in\mathcal{H}} \frac{\sum_{i\in\mathcal{I}} \frac{x_i}{r_i} w^+_{i,h}}{r_h + \sum_{i\in\mathcal{I}} \frac{x_i}{r_i} w^-_{i,h}} w^+_{h,o}}{r_o + \sum_{h\in\mathcal{H}} \frac{\sum_{i\in\mathcal{I}} \frac{x_i}{r_i} w^+_{i,h}}{r_h + \sum_{i\in\mathcal{I}} \frac{x_i}{r_i} w^-_{i,h}} w^-_{h,o}}.$$

- This shows that the output is a rational function of the input. This allows many treatments. Also, for learning, costs (errors) are rational functions of weights, again, allowing some specific mathematical treatments.

Outline

Main issue

- With PSQA, we know how to measure the PQ very accurately and in real time, and as you saw, to measure PQ we actually need to measure QoS metrics and channel metrics, the input variables of our PSQA function.
- So, to predict PQ, we need to predict the values of those input variables.
- Some of them will be rather static, and their values come for free. Others need measures. The hardest part of this prediction problems is to predict future values of a *time series* (for instance, traffic intensity, or loss rates).

Time series prediction

- Predicting the future of a time series is an old topic in statistics. The applications are uncountable (economy, finance, engineering, meteorology, climate, ...).
- With the explosion of AI techniques (and the associated results), AI entered the game, and using learning methods have provided new efficient prediction tools.
- We will describe here the subarea of Machine Learning where we worked, and then, our own tools there.
- A couple of facts related to the topic:
  - Recurrent neural networks are good for *memorizing* information, so, well-behaving for time series predictions.
  - Recurrent neural networks are harder to train than Feed Forward ones.

Reservoir Computing (RC)

- RC neural networks 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.
- The two most popular RC models are
  - the *Echo State Network (ESN)*
    (see H. Jaeger, "The *echo state* approach to analyzing 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.

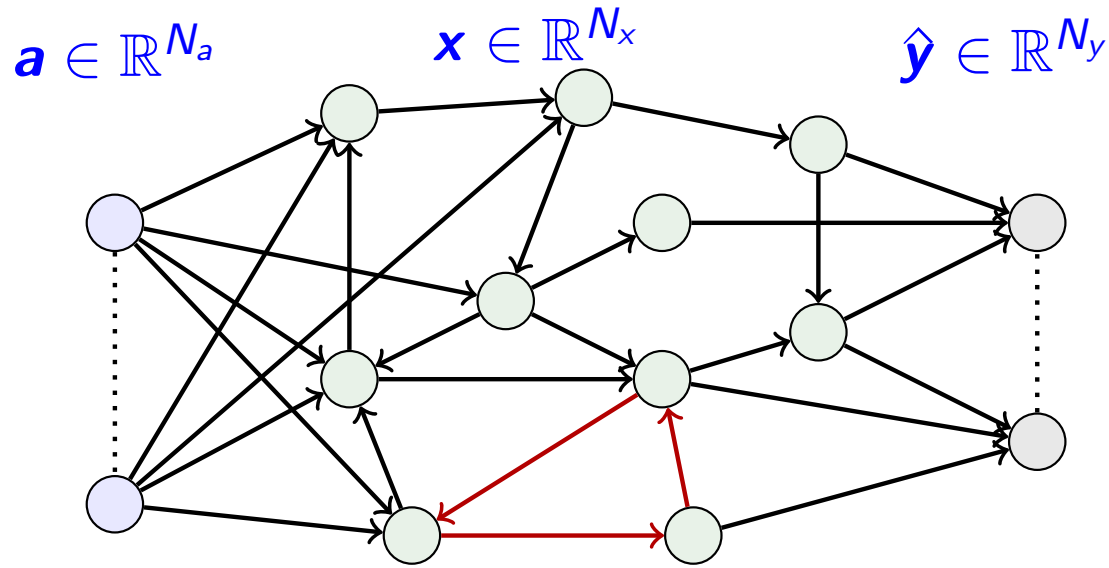
Echo State Networks (ESNs)

We focus on the ESN implementation of the RC idea.

- 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).

ESN properties

Some aspects of the ESN model:

- The iterative computation of $\boldsymbol{x}(t)$ can be unstable due to the recurrences.
- The stability behavior is controlled by the spectral radius of the weights in the reservoir.
- 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).

ESP

The Echo State Property

- Even though the trajectories of the reservoir states are random initialized, the model should be independent of the initial network trajectories in the long term.
- The network needs to have some type of *fading memory* with respect of the initial conditions and initial dynamics.
- These characteristics are established in the ESP.
- The ESP establishes that the trajectories of reservoir states depend only of the input driven network, they don't depend on the initial conditions of the network.
- If the spectral radius of the reservoir's weights is < 1, we have the ESP (sufficient condition); a necessary condition is to have it ≤ 1.

Some practical issues

Stability of the recurrences in an ESN model:

- A simple procedure for creating an ESN is to randomly initialize the reservoir weight matrix and then to scale it using a factor α.
- The selection of the scaling factor is important for the ESP.
- Previous 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 $\alpha <$ than the inverse of the spectral radius.
- Concerning the activation function, no strong constraints.

Tuning ESNs

Main adjustable parameters of a RC model:

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

less accuracy — more accuracy

$$- \longleftrightarrow +$$

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

less memory — more memory

$$0 \longleftrightarrow 1$$

Outline

Our work in predicting time series for PSQA 2.0

- As Random Neural Networks behaved so well for our PQ-measuring applications, we decided to keep close also for predicting PQ.
- We took one of the most promising ideas in ML applied to the problem, and worked on its adaptation in the Random Neural Network world.
- The result is described in next slides.

Echo State Queueing Networks (ESQNs)

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

- Three sets of neurons, recurrent topology "in the middle".
- Input at time t, $\boldsymbol{a}(t) = (a_1(t), \ldots, a_{N_a}(t))$:
  $\rho_u(t) = a_u(t)/r_u$, (in general, $a_u(t) < r_u$), for $u \in (1..N_a)$.
- For all reservoir units $u = N_a + 1, \ldots, N_a + N_x$,

$$\rho_u(t) = \frac{\displaystyle\sum_{v=1}^{N_a} \frac{a_v(t)}{r_v} w^+_{u,v} + \sum_{v=N_a+1}^{N_a+N_x} \rho_v(t-1) w^+_{u,v}}{\displaystyle r_u + \sum_{v=1}^{N_a} \frac{a_v(t)}{r_v} w^-_{u,v} + \sum_{v=N_a+1}^{N_a+N_x} \rho_v(t-1) w^-_{u,v}}. \qquad (1)$$

Weights notation as classical NNs: u sending spikes to v is denoted by $w^{+/-}_{v,u}$. Notation r_u is for the rate of unit (neuron) 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 $\hat{\boldsymbol{y}}(t) = (\hat{y}_1(t), \ldots, \hat{y}_{N_b}(t))$ is computed for any $m \in [1..N_b]$:

$$y_m(t) = w^{\text{out}}_{m,0} + \sum_{i=1+N_a}^{N_a+N_x} w^{\text{out}}_{m,i} \rho_i(t). \tag{2}$$

- The output weights 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*.

ESQN

Examples: simulated dataset

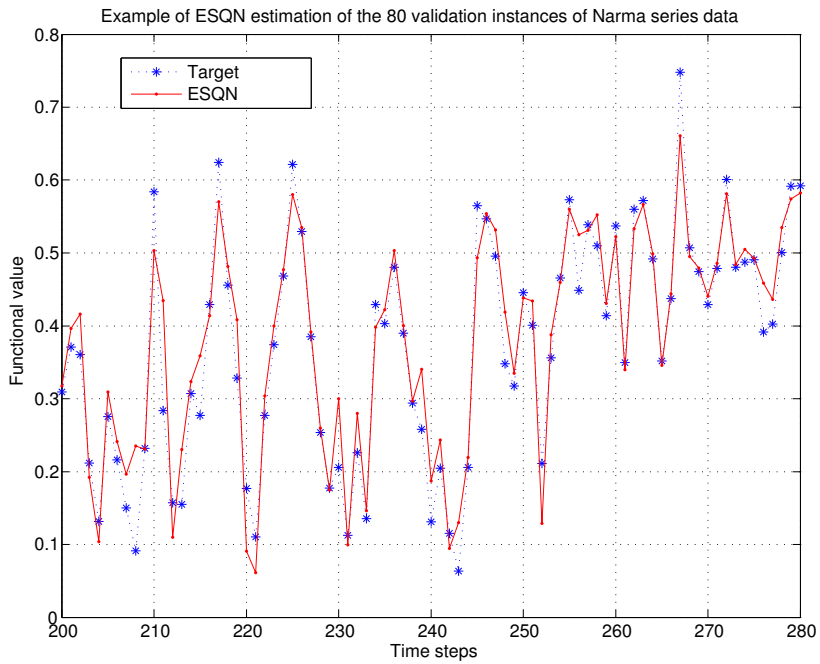


Figure: NARMA validation data set. The reservoir was randomly initialized and it had 80 units.

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

ESQN

Examples: Internet traffic prediction

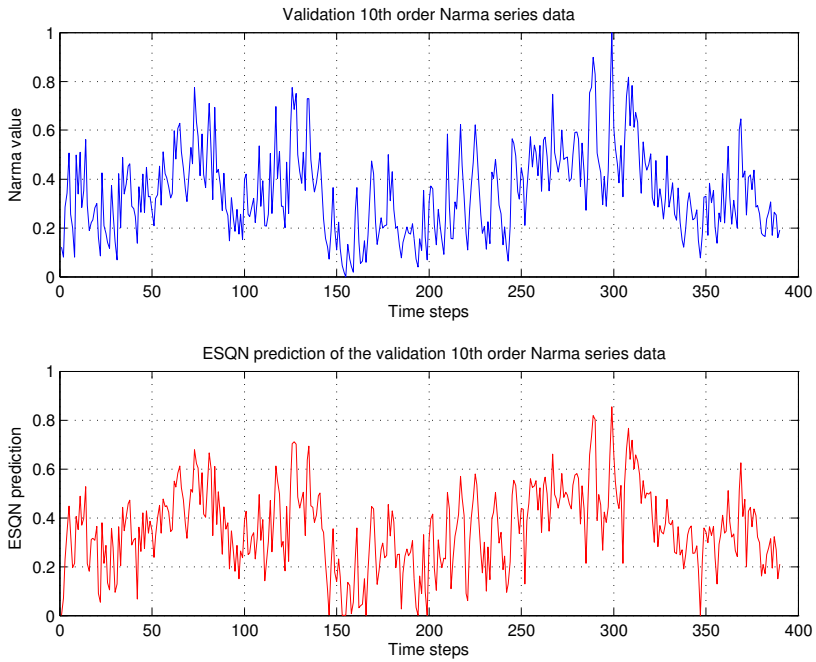


Figure: Fixed 10th order NARMA times series. Comparison between the original dataset and the forecasted samples on the validation dataset.

ESQN

Examples: Internet traffic prediction

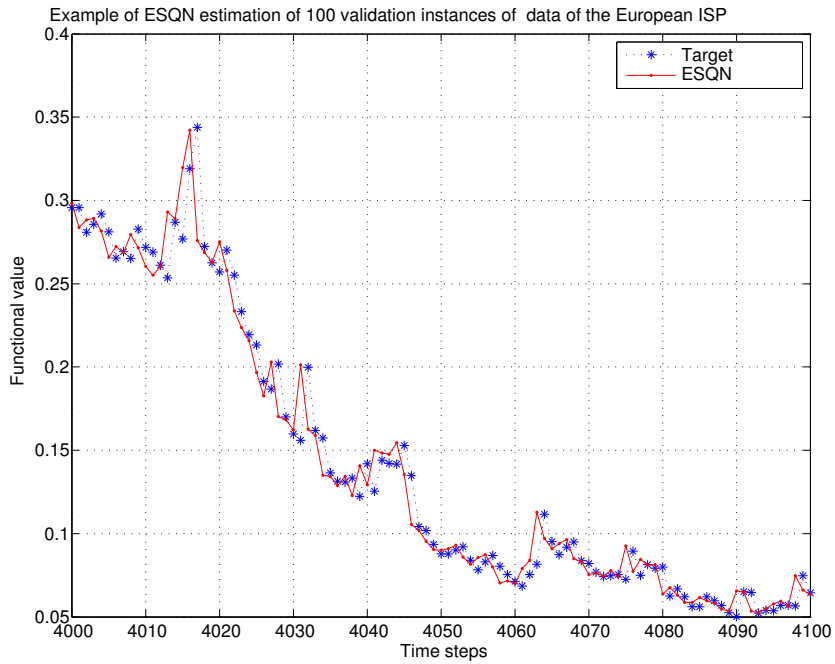


Figure: Normalized Internet traffic data from an European Internet Service Provider. The reservoir has 40 neurons and it was randomly initialized.

ESQN

Examples: Internet traffic prediction

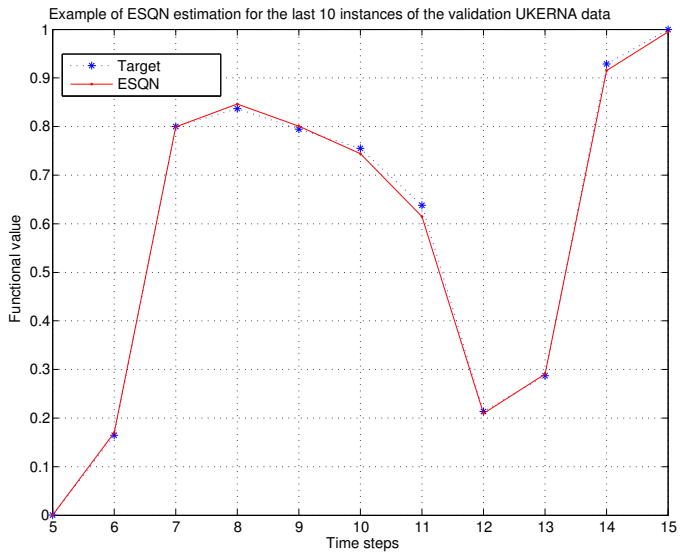


Figure: ESQN estimation for UKERNA validation data set. The reservoir weights were randomly initialized. The reservoir size is 40.

ESQN

Examples: Internet traffic prediction

We can see that ESN is more stable than ESQN:

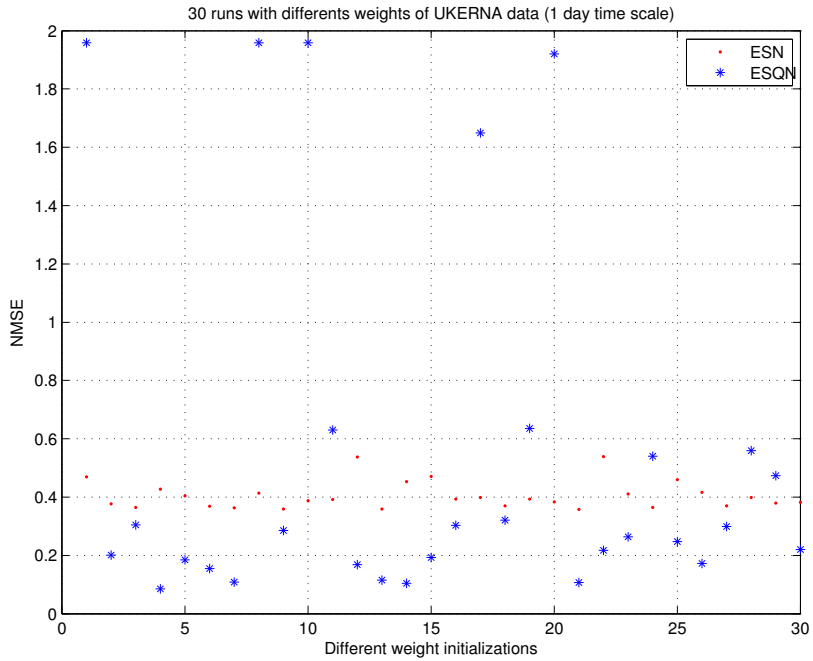


Figure: ESQN and ESN model accuracy (NMSE) for different reservoir initializations for the Internet traffic prediction (UKERNA validation data set).

ESQN

Sensitivity of the global parameters.

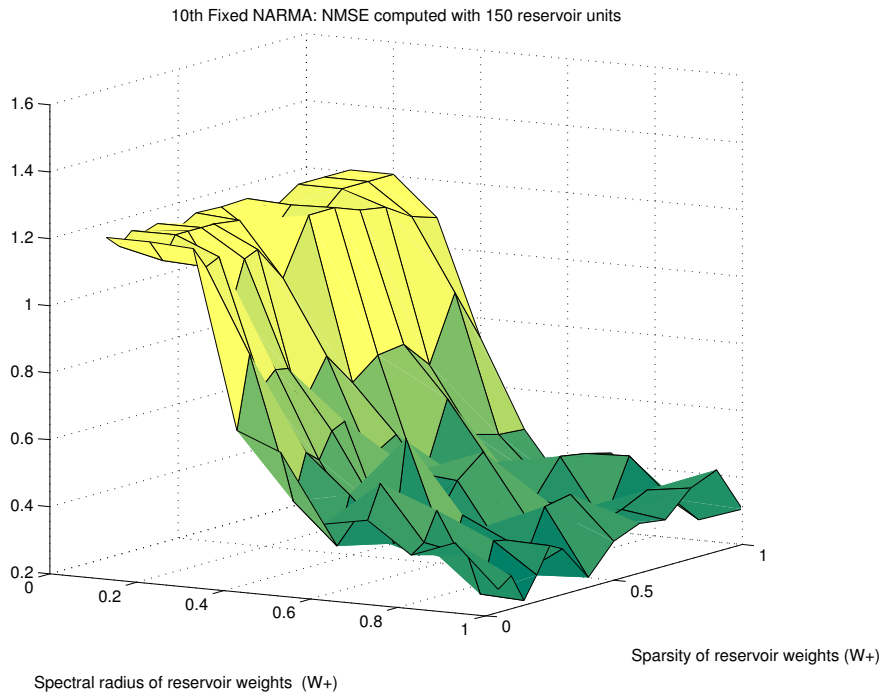


Figure: Simulated dataset. The NMSE according to the spectral radius and the sparsity of the reservoir weights.

ESQN

Sensitivity of the global parameters.

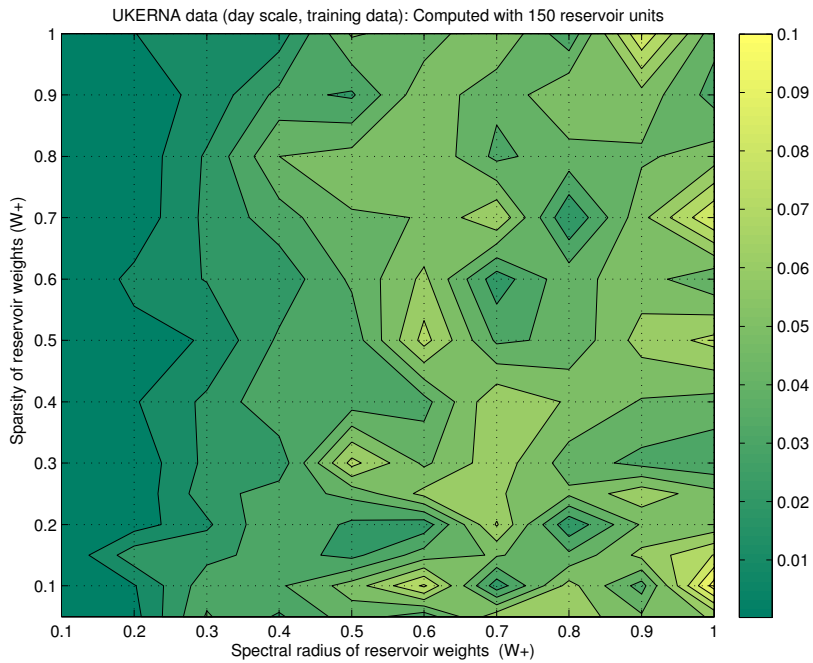


Figure: Internet traffic prediction. The NMSE according to the spectral radius and the sparsity of the reservoir negative weights matrix.

ESQN

Sensitivity of the global parameters.

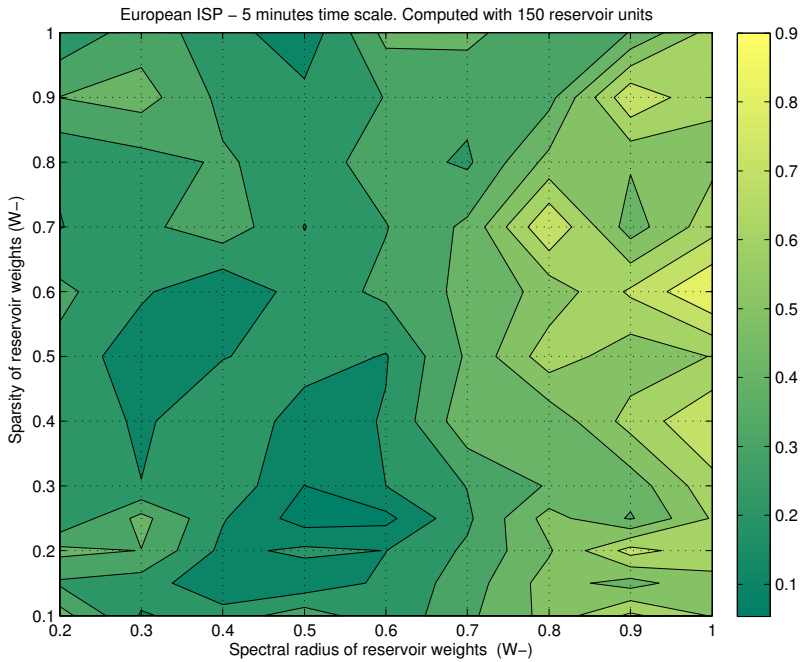


Figure: Internet traffic prediction. The NMSE according to the spectral radius and the sparsity of the reservoir negative weights matrix.

ESQN

Sensitivity of the global parameters.

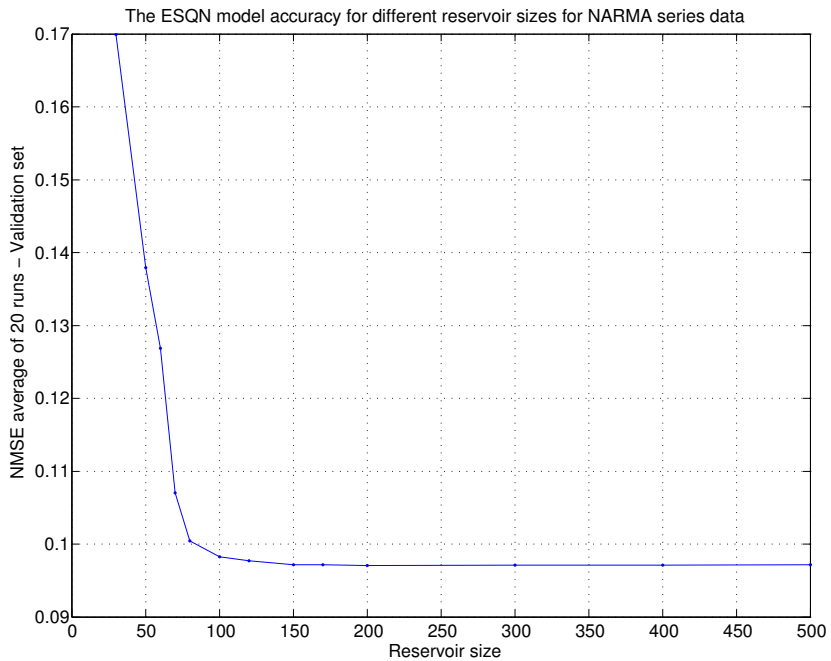
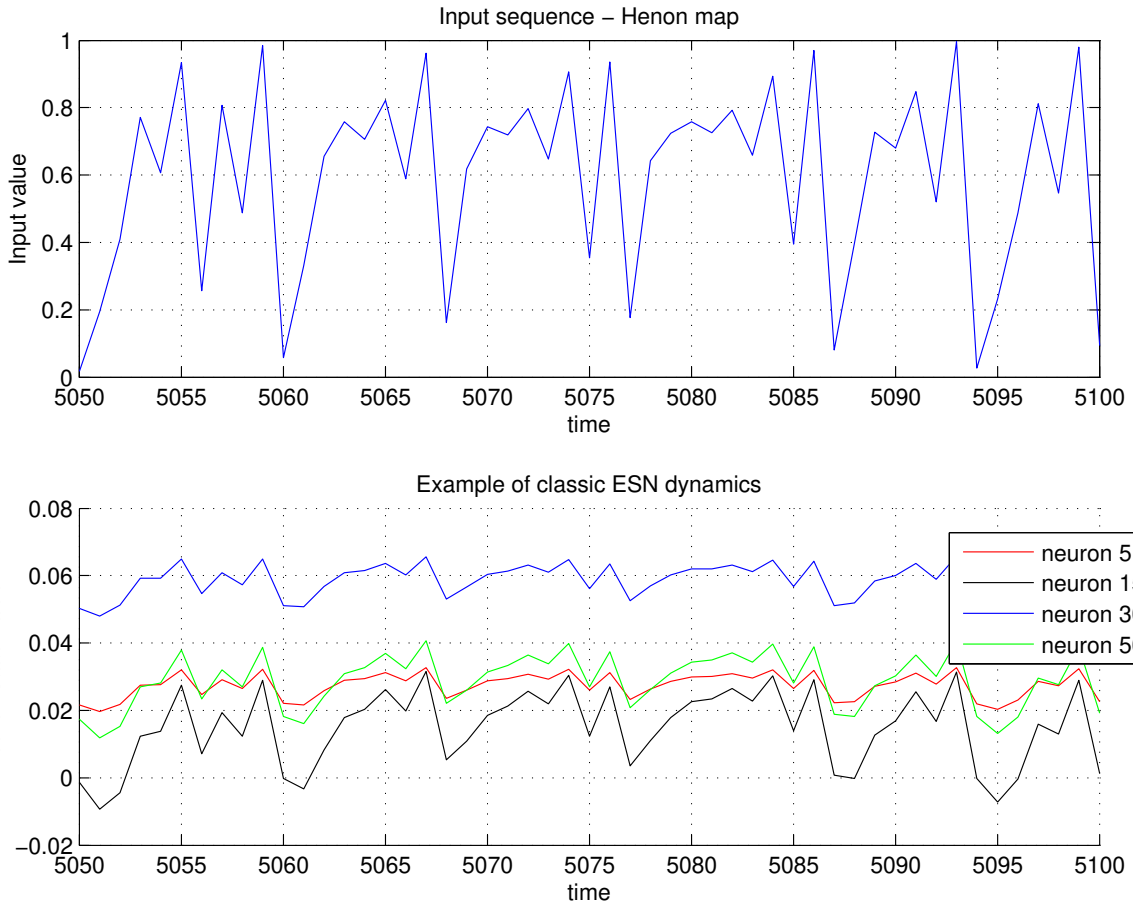


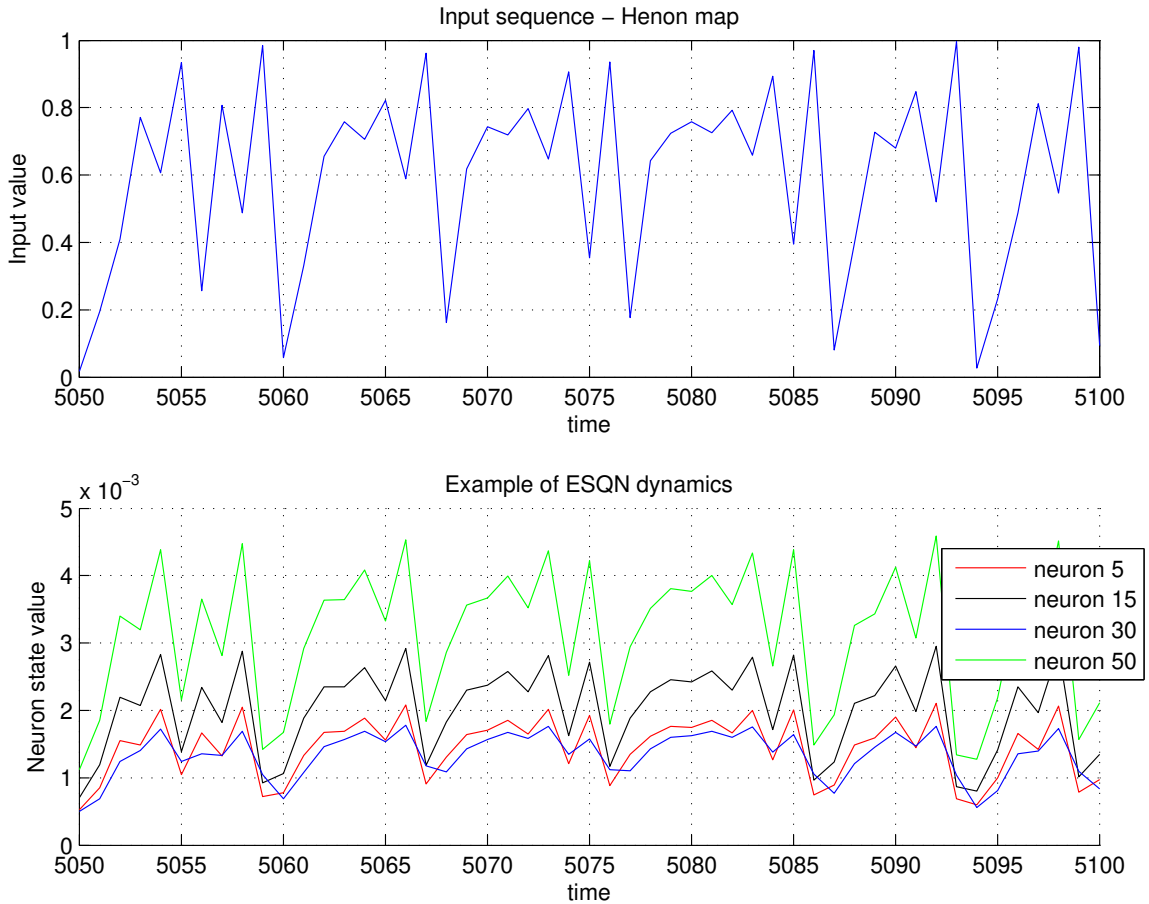
Figure: The ESQN model performance for different reservoir sizes which are computed for 10th NARMA validation data set. The reservoir weights were randomly initialized. Figure shows the NMSE average achieved in 20 runs with different ESQN initial weights.

Echo State Queueing Network



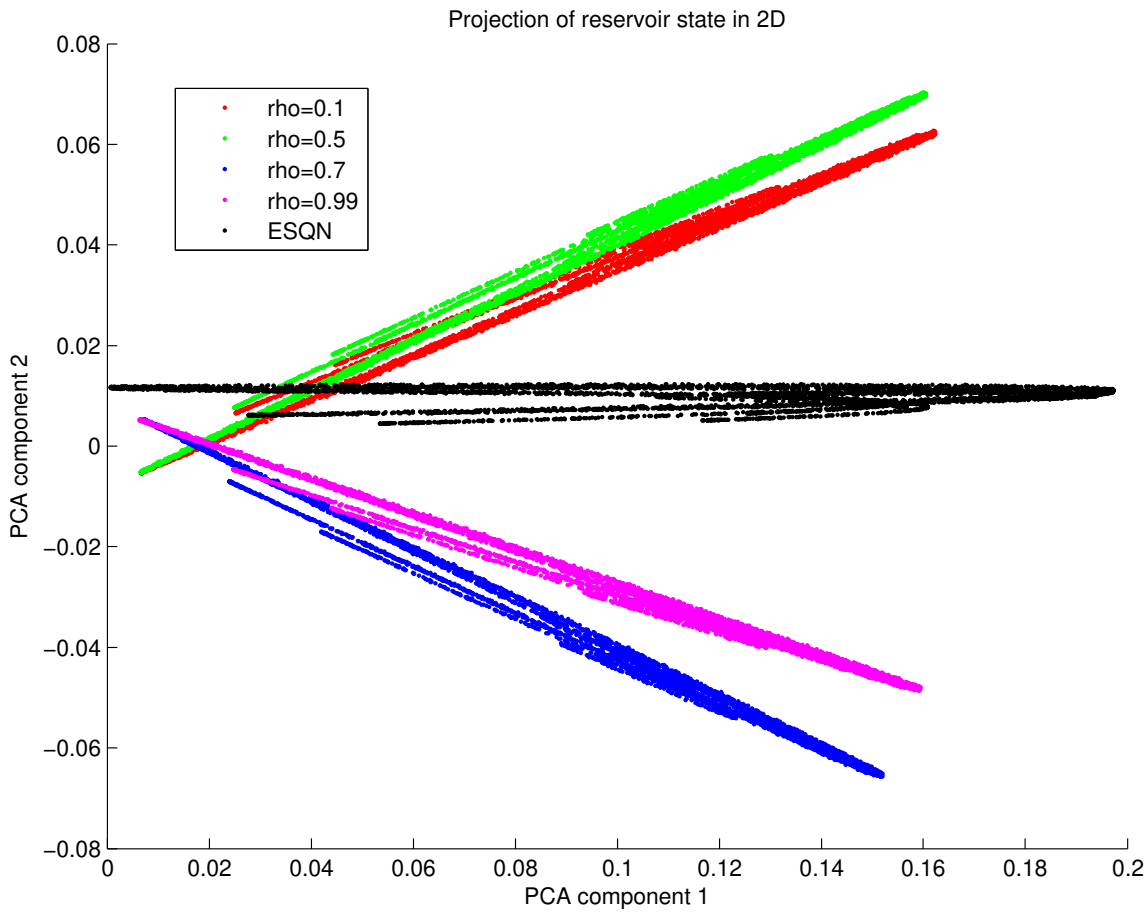
Note: Henon Map as input data. ESN has 50 fully connected reservoir neurons and spectral radius (ρ) equal to 0.5.

Echo State Queueing Network



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

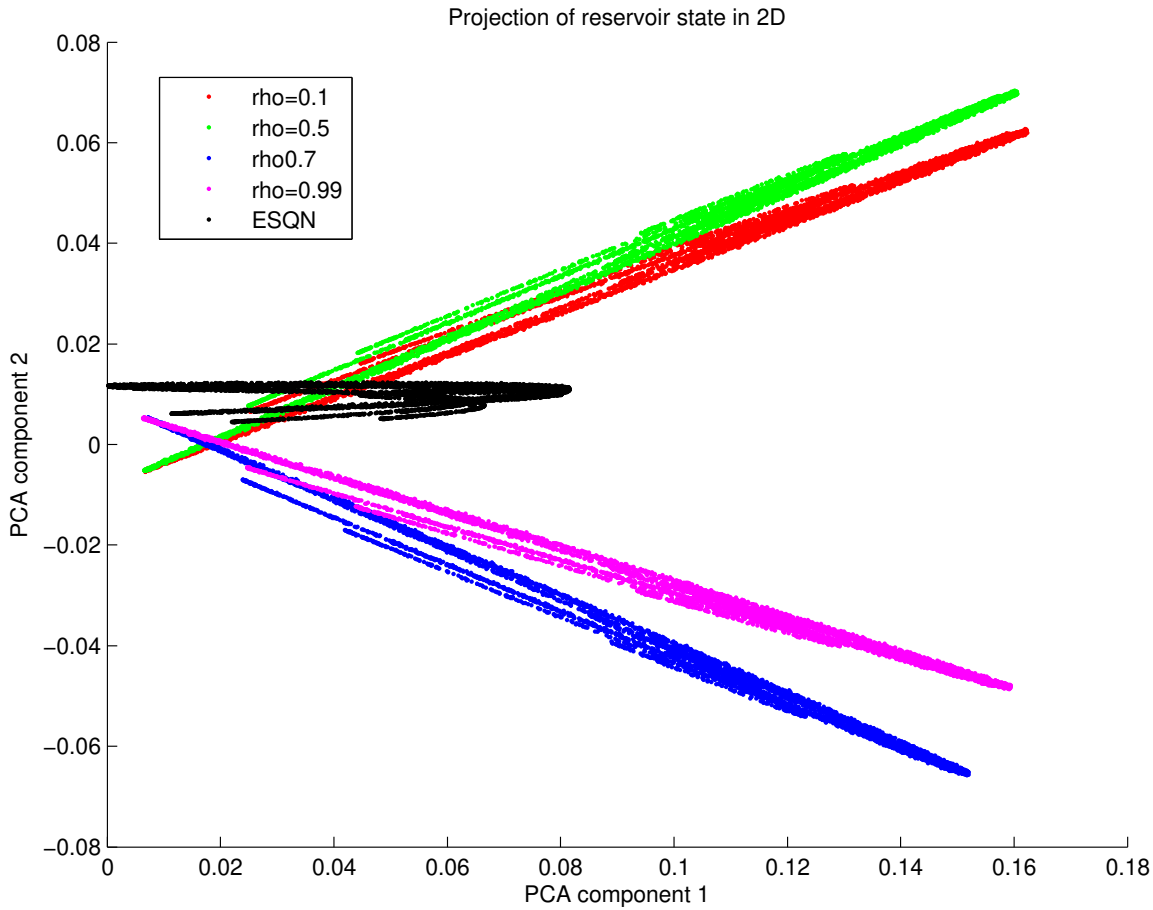
Echo State Queueing Network



Comments

Note about the previous picture:
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 one:
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

Summary about the ESQN tool

- We have done some preliminary experimental work, to explore ESQN's behavior, 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 used 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 of ESQN:

- The impact of the reservoir size is similar to the impact of the number of neurons 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 modeling.
  - This good behavior is not well understood.
  - Both ESNs and ESQNs require some expertise for setting their parameters.
- We would like to find conditions for the initialization 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.

On the ML side

- We are exploring the use/interest of Random Neurons in deep architectures. Some encouraging preliminary results available (NIPS 2017 DL workshop, December 9th, 2017).
- Analysis of several theoretical questions around Reservoir Computing with our ESQN model (mainly around convergence, stability, etc.).
- In PSQA, the labels come from subjective tests (that is, from panel of human subjects). We have a project whose goal is the elimination of the (main) use of those panels. This leads to
  - Big Data problems, coming from the extensive use of automatic but less performant tools to provide quality assessments of huge amounts of sequences,
  - and then, to the exploration of the use of specific deep architectures for the learning phase (not necessarily related to RNNs).

Outline

- Maths behind our main application:
  "*Quantifying the Quality of Audio and Video Transmissions over the Internet: The PSQA Approach*", G. Rubino, in Design and Operations of Communication Networks: A Review of Wired and Wireless Modelling and Management Challenges, edited by J. Barria, Imperial College Press, 2005.
- Practical aspects of our uses of RNN in learning:
  "*Evaluating Users' Satisfaction in Packet Networks Using Random Neural Networks*", G. Rubino, P. Tirilly and M. Varela, Springer-Verlag Lecture Notes in Computer Science, no. 4132, 2006.
- An example of using RNNs in combinatorial optimization:
  "*A GRASP algorithm with RNN-based local search for designing a WAN access network*, H. Cancela, F. Robledo and G. Rubino, Electronic Notes in Discrete Mathematics 18 (1), 59–65, December 2004.

- An example of application of PSQA:
  "*Controlling Multimedia QoS in the Future Home Network Using the PSQA Metric*", J.-M. Bonnin, G. Rubino and M. Varela, in The Computer Journal, 49(2):137–155, 2006.
- On the design of a P2P streaming network based on PSQA:
  "*A robust P2P streaming architecture and its application to a high quality live-video service*", H. Cancela, F. Robledo Amoza, P. Rodríguez-Bocca, G. Rubino and A. Sabiguero, in Electronic Notes in Discrete Mathematics 30: 219–224, 2008,
  plus another paper with a demo,
  "*Automatic Quality of Experience Measuring on Video Delivering Networks*", D. De Vera, P. Rodríguez-Bocca and G. Rubino, in SIGMETRICS Performance Evaluation Review, Vol. 36, Issue 2, associated with a demonstration at Sigmetrics'08 awarded with the Best Demonstration Prize.

- An example of improvement on the initial RNN tool:
  "*Levenberg-Marquardt Training Algorithms for Random Neural Networks*", S. Basterrech, S. Mohammed, G. Rubino and M. Soliman, in The Computer Journal, Vol. 54, N. 1, 125–135, 2011.
- An example of extension of the initial RNN tool:
  "*Echo State Queueing Networks: a combination of Reservoir Computing and Random Neural Networks*", S. Basterrech and G. Rubino, in Probability in the Engineering and Informational Sciences, Vol. 31, No. 4, pp. 1–16, 2017.