Random Neural Networks and applications
Gerardo Rubino

To cite this version:
Gerardo Rubino. Random Neural Networks and applications. Doctoral. Engineering School, Montevideo, Uruguay. 2022. hal-03901350

HAL Id: hal-03901350
https://inria.hal.science/hal-03901350
Submitted on 15 Dec 2022

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.
Random Neural Networks and applications

G. Rubino

INRIA Rennes, France

Nov 2022
Outline

1 — RNNs

2 — Queuing origins

3 — An application to QoE analysis

4 — Extension in the Reservoir Computing class

5 — Some references
Outline

1 — RNNs

2 — Queuing origins

3 — An application to QoE analysis

4 — Extension in the Reservoir Computing class

5 — Some references
Classic artificial neurons

- A “classic artificial neuron” can be identified with its activation function, that is, can be seen as a real function of a real variable.
  
  \[ z = \frac{1}{1 + e^{-u+b}} \]

- We consider \( b \) as a parameter of the function/neuron.

- There are many different activation functions used in practice:
  
  \[ z = \tanh(u - b), \quad z = 1(u \geq b), \quad z = (x - b)^+ \text{ (ReLU)}, \quad \ldots \]

- Then, we build Neural Network by interconnecting these functions using real weights...
Random Neurons

- A Random Neuron is a parametric positive real function of two real positive variables or “ports”, one called the “positive port”, the other one being the “negative port”.

\[
\begin{align*}
    u &\geq 0 \\
    v &\geq 0 \\
    z &= \frac{u}{r + v} \geq 0
\end{align*}
\]

- We see \( r > 0 \), the rate of the neuron, as a parameter.
- There are variants of this model. The main one is 
  \( z = \min(u/(r+v), 1) \) (the original one).
- Inventor: Erol Gelenbe, Imperial College, in the late 80s.
- Observe that there is nothing random here.
Output $z$ as a function of $u$

We illustrate here the obtained behavior in the case of

$$z = \min\left(\frac{u}{r + v}, 1\right).$$
Output $z$ as a function of $v$ when $u \leq r$

(case of $u \leq r$)

Output $z$ as a function of $v$ when $u \leq r$
(in this case the neuron is not saturated).
Output $z$ as a function of $v$ when $u \geq r$

Output $z$ as a function of $v$ when $u \geq r$; see that as far as $v \leq u - r$, the neuron is saturated.
A Random Neural Network (RNN) is a set of interconnected RNs. The connections are (multiplicatively) weighted by nonnegative reals.

For instance,

\[ u_2 = z_1 w_{1,2}^+ \]
\[ v_2 = z_1 w_{1,2}^- \]
Jonctions at input ports are additive:

\[ u_3 = z_1 w_{1,3}^+ + z_2 w_{2,3}^+ \]
\[ v_3 = z_1 w_{1,3}^- + z_2 w_{2,3}^- \]
Possible restriction

- This model comes from queuing theory.
- Because of the queuing origin of the RNN tool, in order to respect the standard interpretation of the model, we must have
  \[
  \sum_j \left( w_{i,j}^+ + w_{i,j}^- \right) = r_i.
  \]
- This is a constraint to be respected in order to keep the queueing interpretation (and thus, to access all the theory and algorithms associated with).
- The same reason explains the use of \( z = \min(u/(r + v), 1) r \) instead of \( z = u/(r + v) \) (see below).
The general 3-layer Random Neural Network

A very used particular architecture:

- Output units, $\mathcal{O}$
- Hidden units, $\mathcal{H}$
- Input units, $\mathcal{I}$
• Assume the negative ports of input neurons aren’t used. Assume a single output neuron, so, a scalar network output. Assume stability.

• 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.

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

• 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, leading to many advantages over other forms.
Outline

1 — RNNs

2 — Queuing origins

3 — An application to QoE analysis

4 — Extension in the Reservoir Computing class

5 — Some references
Origin of the model

Consider the M/M/1 model, with arrival rate $\lambda$ and service rate $\mu$. The queue is stable iff $\lambda < \mu$; in that case, the steady state distribution $(\pi_n)$ is given by $\pi_n = (1 - \rho)\rho^n$, $n \in \mathbb{N}$, where the number $\rho = \lambda/\mu = 1 - \pi_0$ is the utilization factor, or load, of the system:

- $\rho = \Pr(\text{in equilibrium, the system is busy})$;
  if $\lambda \geq \mu$, then we have that $\Pr(\text{there exists } t < \infty \text{ such that after } t, \text{ system is always busy}) = 1$.
- See then that the load of the queue at infinity, $\rho$, satisfies

$$\rho = \min\left\{\frac{\lambda}{\mu}, 1\right\}.$$
A basic G-queue: positive (standard) customers arrive with rate $\lambda^+$; negative ones arrive with rate $\lambda^-$; both arrival processes are Poisson; the service rate is $\mu$. To understand the semantics, see the graph below, associated with this model.

We have stability $\iff \lambda^+ < \mu + \lambda^-$, and if it holds, then $\rho = \frac{\lambda^+}{\mu + \lambda^-}$. 
Proof of the relationship between G-queues and Random Neurons

• Assume stability and equilibrium.

• Let $T_i^+$ be the mean input flow of positive customers in equilibrium, $T_i^-$ that of negative ones, and $T_o^+$ and $T_o^-$ the corresponding mean output flows.

• We have $T_i^+ = \lambda^+$ and $T_i^- = \lambda^-$. Then, using the mean flow conservation law for positive customers $T_i^+ = T_o^+$, we have

$$\lambda^+ = \mu \rho + \lambda^- \rho,$$

with $\rho$ the server occupation probability (always in equilibrium), also called the system’s load, leading to

$$\rho = \frac{\lambda^+}{\mu + \lambda^-}.$$
A tandem: first queue is an M/M/1; when leaving the first node, with probability $p$ customers go to a second queue as positive ones, and with probability $1-p$ as negative signals; service rate at queue $i$ is $\mu_i$, $i = 1, 2$.

Here, theory says, for instance, that stability happens $\iff \lambda < \mu_1, \mu_2$, and in that case, if $X_i = 1$ (queue $i$ is busy at infinity), $i = 1, 2$, we have, for $j, k = 0, 1$,

$$\Pr(\text{at infinity, } X_1 = j, \ X_2 = k) = \rho_1^j \rho_2^k.$$
A general recurrent G-network with two nodes. We denote by $d_i$ the probability of leaving the network after a service at queue $i$. 
A 3-layer structure

The graph of a 3-layer RNN of the (2,3,1) type (as before, dashed arrows correspond to flows of negative customers/signals).
On the weights of a RNN

• Think queueing; then, when a customers leaves queue $i$ (necessarily a positive customer), it goes to queue $j$ as a positive customer with some fixed probability $p_{i,j}^+$, and as a negative customer with some fixed probability $p_{i,j}^-$.

• Then, the mean throughput (the frequency) of the flow of positive customers (of spikes) from $i$ to $j$ is $r_ip_{i,j}^+ =: w_{i,j}^+$, and of negative ones is $r_ip_{i,j}^- =: w_{i,j}^-$.

• So, the weights here (the things that learn are the mean speed at which signals between neurons move. This actually also hold for “hulid” neurons.

• Since for any neuron $i$ in the network, we must have $\sum_j(p_{i,j}^+ + p_{i,j}^-) = 1$ (add a queue/neuron 0 representing the network’s environment), we deduce that $\sum_j(w_{i,j}^+ + w_{i,j}^-) = r_i$
Currently, we are working on the development of a tool implementing these RNN models, and adding some new possibilities:

- **sensitivity analysis**: basically, $\frac{\partial y_j}{\partial x_i}$, where $x_i$ is the $i$th input and $y_j$ is the $j$th output;

- **inverting the RNN mapping when the output is a scalar**: basically, if we denote $y = \nu(\vec{x})$ where $\vec{x}$ is the vector input to the network and $y$ is the real output, then we look for the subset $\nu^{-1}(y \geq y_0)$, that is, the set of input leading to the fact that the output is at least $y_0$. 
Outline

1 — RNNs

2 — Queuing origins

3 — An application to QoE analysis

4 — Extension in the Reservoir Computing class

5 — Some references
Context

- Consider any kind of application or service centered on 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, 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? How to measure it automatically?
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

- The result is a table $M$ where component $M_{h,\sigma}$ is the value given by human $h$ to sequence $\sigma$.
- 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 $S$, the quality of sequence $\sigma$, its $MOS$ (Mean Opinion Score) value is

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

- 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 $\mu_\sigma$ and $\mu'_\sigma$ are the MOS values of sequence $\sigma$ 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:

<table>
<thead>
<tr>
<th>MOS score</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>degradation imperceptible</td>
</tr>
<tr>
<td>4</td>
<td>degradation perceptible but not annoying</td>
</tr>
<tr>
<td>3</td>
<td>degradation slightly annoying</td>
</tr>
<tr>
<td>2</td>
<td>degradation annoying</td>
</tr>
<tr>
<td>1</td>
<td>degradation very annoying</td>
</tr>
</tbody>
</table>
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
    \[
    \text{MSE}(A, B) = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} (A_{i,j} - B_{i,j})^2 \in [0..255^2].
    \]
  - The PSNR between the two images is defined only if they are different ($\text{MSE} \neq 0$) and its value (in dB) is
    \[
    \text{PSNR}_{\text{dB}}(A, B) = 10 \log_{10} \left( \frac{255^2}{\text{MSE}(A, B)} \right).
    \]
Problems with PSNR approach

Noise in the sky
Problems with PSNR approach...

Same amount of noise in the grass
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.
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).
The mapping is based on supervised learning.
We explored different tools for this supervised learning task, including the ToolBox of Matlab.
The winner was the RNN model.
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 $\sigma_i$ through a simulated or deployed network where we can simultaneously control all components of the configurations.
Then we send different original $\sigma_i$ using the platform, when the configuration is $\vec{\gamma}_j$ using one configuration at a time, and obtain a possibly degraded sequence $\sigma'_j$. 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 × “channel state” to MOS values). Data: the $M$ pairs $(\vec{\gamma}_j, Q_j)$.

In the example of video, we obtain a function $\nu(LR, BR, FR)$. In the VoIP example, we obtain a function $\nu(LR, MLBS, BR, OFEC)$. 
Why using RNNs in PSQA project?

Because of the favorable comparison with standard software (at the time of the beginning, several years ago). An example: with the same data and the same neural network size, on the left, PSQA, and on the right, Matlab (an old version of the ToolBox on learning techniques).
Auditing the network with PSQA
Auditing the network with PSQA (cont.)

A “graphical” overview:
Outline

1 — RNNs

2 — Queuing origins

3 — An application to QoE analysis

4 — Extension in the Reservoir Computing class

5 — Some references
Reservoir Computing

• The idea was to develop models that use 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 are known today under the name of *Reservoir Computing* (RC) paradigm.

• The two most popular RC models are
  • the *Echo State Network (ESN)*
    (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)*
Echo State Networks (ESNs)

The main representative of the family. It has
- a first recurrent part called *reservoir*, with fixed weights,
- and a second supervised learning part called *readout*.

Three-layered neural networks:
- Input layer
- Hidden layer
- Output layer

The learning process is restricted to the output weights (readout).
Our Echo State Queueing Networks (ESQNs)

• Echo State Queueing Networks: applying the RC idea to a RNN. Joint work with S. Basterrech (Univ. of Prague).

• Three sets of neurones, a recurrent topology “in the middle”. Discrete time.

• Input at time $t$, $a(t) = (a_1(t), \ldots, a_{N_a}(t))$:
  $\rho_u(t) = a_u(t)/r_u$, (in general, we take $a_u(t) < r_u$), for $u \in 1..N_a$.

• For all reservoir units $u = N_a + 1, \ldots, N_a + N_x$,

$$
\rho_u(t) = \frac{\sum_{v=1}^{N_a} \frac{a_v(t)}{r_v} w_{v,u}^+ + \sum_{v=N_a+1}^{N_a+N_x} \rho_v(t-1) w_{v,u}^+}{r_u + \sum_{v=1}^{N_a} \frac{a_v(t)}{r_v} w_{v,u}^- + \sum_{v=N_a+1}^{N_a+N_x} \rho_v(t-1) w_{v,u}^-}.
$$
• The input space is then projected into a new "larger" space.
• We compute a linear regression from the projected space to the output space.
• Thus, the network output $\hat{y}(t) = (\hat{y}_1(t), \ldots, \hat{y}_{N_b}(t))$ is computed for any $m \in 1..N_b$:

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

• Learning process: the output weights $w_*^{\text{out}}$ can be computed using any (fast) procedure, for instance, Least Mean Square algorithms.
• Remark: we can replace this simple structure by, for instance, a classical feedforward 3-level RNN.
Outline

1 — RNNs

2 — Queuing origins

3 — An application to QoE analysis

4 — Extension in the Reservoir Computing class

5 — Some references
• About the RNN model and the PSQA tool:

• Practical aspects of our uses of RNN in learning:


• 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

• 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.