Echo state network

Herbert Jaeger (2007), Scholarpedia, 2(9):2330. doi:10.4249/scholarpedia.2330 revision #183563 [link to/cite this article]

- **Herbert Jaeger**, Jacobs University Bremen, Bremen, Germany

Post-publication activity

Curator: Herbert Jaeger

Echo state networks (ESN) provide an architecture and supervised learning principle for recurrent neural networks (RNNs). The main idea is (i) to drive a random, large, fixed recurrent neural network with the input signal, thereby inducing in each neuron within this "reservoir" network a nonlinear response signal, and (ii) combine a desired output signal by a trainable linear combination of all of these response signals.

The basic idea of ESNs is shared with *Liquid State Machines* (LSM), which were developed independently from and simultaneously with ESNs by Wolfgang Maass (Maass W., Natschlaeger T., Markram H. 2002). Increasingly often, LSMs, ESNs and the more recently explored *Backpropagation Decorrelation* learning rule for RNNs (Schiller and Steil 2005) are subsumed under the name of *Reservoir Computing*. Schiller and Steil (2005) also showed that in traditional training methods for RNNs, where all weights (not only the output weights) are adapted, the dominant changes are in the output weights. In cognitive neuroscience, a related mechanism has been investigated by Peter F. Dominey in the context of modelling sequence processing in mammalian brains, especially speech recognition in humans (e.g., Dominey 1995, Dominey, Hoen and Inui 2006). Dominey was the first to explicitly state the principle of reading out target information from a randomly connected RNN. The basic idea also informed a model of temporal input discrimination in biological neural networks (Buonomano and Merzenich 1995). The earliest known clear formulation of the reservoir computing idea, however, is due to K. Kirby who exposed this concept in a largely forgotten (1 Google cite, as of 2017) conference contribution (Kirby 1991).

For an illustration, consider the task of training an RNN to behave as a tunable frequency generator (download the MATLAB code of this example). The input signal $u(n)$ is a slowly varying frequency setting, the desired output $y(n)$ is a sinewave of a frequency indicated by the current input. Assume that a training input-output sequence $D = (u(1),y(1)), \ldots, (u(n_{max}),y(n_{max}))$ is given (see the input and output signals in ; here the input is a slow random step function indicating frequencies ranging from 1/16 to 1/4 Hz). The task is to train a RNN from these training data such that on slow test input signals, the output is again a sinewave of the input-determined frequency.

In the ESN approach, this task is solved by the following steps.

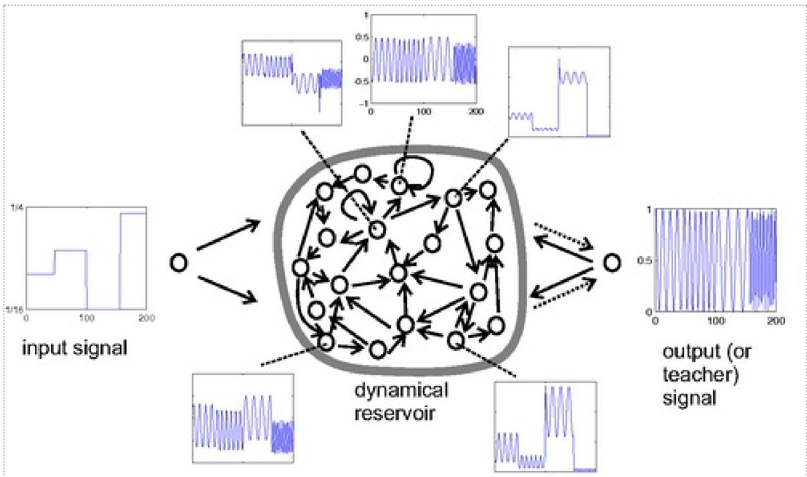

Figure 1: The basic schema of an ESN, illustrated with a tuneable frequency generator task. Solid arrows indicate fixed, random connections; dotted arrows trainable connections.

- **Step 1: Provide a random RNN.** (i) Create a random *dynamical reservoir* RNN, using any neuron model (in the frequency generator demo example, non-spiking leaky integrator neurons were used). The reservoir size N is task-dependent. In the frequency generator demo task, $N = 200$ was used. (ii) Attach input units to the reservoir by creating random all-to-all connections. (iii) Create output units. If the task requires output feedback (the frequency-generator task does), install randomly generated output-to-reservoir connections (all-to-all). If the task does not require output feedback, do not create any connections to/from the output units in this step.
- **Step 2: Harvest reservoir states.** Drive the dynamical reservoir with the training data D for times $n = 1, \ldots, n_{max}\ .$ In the demo example, where there are output-to-reservoir feedback connections, this means to write both the input $u(n)$ into the input unit and the teacher output $y(n)$ into the output unit ("teacher forcing"). In tasks without output feedback, the reservoir is driven by the input $u(n)$ only. This results in a sequence $\mathbf{x}(n)$ of N-dimensional reservoir states. Each component signal $x(n)$ is a nonlinear transform of the driving input. In the demo, each $x(n)$ is an individual mixture of both the slow step input signal and the fast output sinewave (see the five exemplary neuron state plots in Figure 1).
- **Step 3: Compute output weights.** Compute the output weights as the linear regression weights of the teacher outputs $y(n)$ on the reservoir states $\mathbf{x}(n)\ .$ Use these weights to create reservoir-to-output connections (dotted arrows in Figure 1). The training is now completed and the ESN ready for use. Figure 2 shows the output signal obtained when the trained ESN was driven with the slow step input shown in the same figure.

Contents [hide]

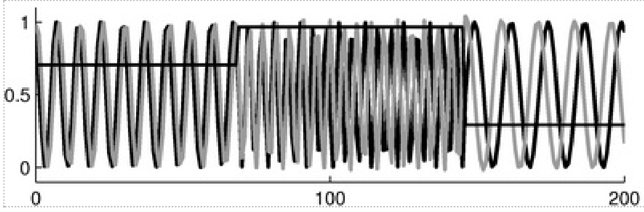

Figure 2: A test run of the frequency generator demo network. The plotted step function is the input signal which was fed to the trained ESN; the black sinewaves are the correct output (unknown to the network); the gray sinewaves are the network output. Notice that phase differences are inevitable.

Variants

Echo state networks can be set up with or without direct trainable input-to-output connections, with or without output-to-reservoir feedback, with different neuron types, different reservoir-internal connectivity patterns, etc. Furthermore, the output weights can be computed with any of the available offline or online algorithms for linear regression. Besides least-mean-square error solutions (i.e., linear regression weights), margin-maximization criteria known from training support vector machines have been used to determine output weights (Schmidhuber et al. 2007)

The unifying theme throughout all these variations is to use a *fixed* RNN as a *random nonlinear excitable medium*, whose high-dimensional dynamical "echo" response to a driving input (and/or output feedback) is used as a non-orthogonal signal basis to reconstruct the desired output by a linear combination, minimizing some error criteria.

Formalism and theory

System equations. The basic discrete-time, sigmoid-unit echo state network with N reservoir units, K inputs and L outputs is governed by the state update equation

(1) $\mathbf{x}(n+1) = f(\mathbf{W} \mathbf{x}(n) + \mathbf{W}^{in} \mathbf{u}(n+1) + \mathbf{W}^{fb} \mathbf{y}(n))\ ,$

where $\mathbf{x}(n)$ is the N-dimensional reservoir state, f is a sigmoid function (usually the logistic sigmoid or the tanh function), $\mathbf{W}$ is the $N \times N$ reservoir weight matrix, $\mathbf{W}^{in}$ is the $N \times K$ input weight matrix, $\mathbf{u}(n)$ is the Kdimensional input signal, $\mathbf{W}^{fb}$ is the $N \times L$ output feedback matrix, and $\mathbf{y}(n)$ is the L-dimensional output signal. In tasks where no output feedback is required, $\mathbf{W}^{fb}$ is nulled. The extended system state $\mathbf{z}(n) = [\mathbf{x}(n); \mathbf{u}(n)]$ at time n is the concatenation of the reservoir and input states. The output is obtained from the extended system state by

(2) $\mathbf{y}(n) = g(\mathbf{W}^{out} \mathbf{z}(n))\ ,$

where g is an output activation function (typically the identity or a sigmoid) and $\mathbf{W}^{out}$ is a $L \times (K+N)$-dimensional matrix of output weights.

Learning equations. In the state harvesting stage of the training, the ESN is driven by an input sequence $\mathbf{u}(1), \ldots, \mathbf{u}(n_{max})\ ,$ which yields a sequence $\mathbf{z}(1), \ldots, \mathbf{z}(n_{max})$ of extended system states. The system equations (1), (2) are used here. If the model includes output feedback (i.e., nonzero $\mathbf{W}^{fb}$), then during the generation of the system states, the correct outputs $\mathbf{d}(n)$ (part of the training data) are written into the output units ("teacher forcing"). The obtained extended system states are filed row-wise into a state collection matrix $\mathbf{S}$ of size $n_{max} \times (N + K)\ .$ Usually some initial portion of the states thus collected are discarded to accommodate for a washout of the arbitrary (random or zero) initial reservoir state needed at time 1. Likewise, the desired outputs $\mathbf{d}(n)$ are sorted row-wise into a teacher output collection matrix $\mathbf{D}$ of size $n_{max} \times L\ .$

The desired output weights $\mathbf{W}^{out}$ are the linear regression weights of the desired outputs $\mathbf{d}(n)$ on the harvested extended states $\mathbf{z}(n)\ .$ A mathematically straightforward way to compute $\mathbf{W}^{out}$ is to invoke the pseudoinverse (denoted by $\cdot^{\dagger}$) of $\mathbf{S}\ :$

(3) $\mathbf{W}^{out} = (\mathbf{S}^{\dagger}\mathbf{D})'\ $,

which is an offline algorithm (the prime denotes matrix transpose). Online adaptive methods known from linear signal processing can also be used to compute output weights (Jaeger 2003).

Echo state property. In order for the ESN principle to work, the reservoir must have the *echo state property* (ESP), which relates asymptotic properties of the excited reservoir dynamics to the driving signal. Intuitively, the ESP states that the reservoir will asymptotically wash out any information from initial conditions. The ESP is guaranteed for additive-sigmoid neuron reservoirs, if the reservoir weight matrix (and the leaking rates) satisfy certain algebraic conditions in terms of singular values. For such reservoirs with a tanh sigmoid, the ESP is violated *for zero input* if the spectral radius of the reservoir weight matrix is larger than unity. Conversely, it is empirically observed that the ESP is granted for any input if this spectral radius is smaller than unity. This has led in the literature to a far-spread but erroneous identification of the ESP with a spectral radius below 1. Specifically, the larger the input amplitude, the further above unity the spectral radius may be while still obtaining the ESP. An abstract characterization of the ESP for arbitrary reservoir types, and algebraic conditions for additive-sigmoid neuron reservoirs are given in Jaeger (2001a); for an important subclass of reservoirs, tighter algebraic conditions are given in Buehner and Young (2006) and Yildiz et al. (2012); for leaky integrator neurons, algebraic conditions are spelled out in Jaeger et al. (2007). The relationship between input signal characteristics and the ESP are explored in Manjunath and Jaeger (2012), where a fundamental 0-1-law is shown: if the input comes from a stationary source, the ESP holds with probability 1 or 0.

Memory capacity. Due to the auto-feedback nature of RNNs, the reservoir states $\mathbf{x}(n)$ reflect traces of the past input history. This can be seen as a dynamical short-term memory. For a single-input ESN, this short-term memory's capacity C can be quantified by $C = \sum_{i = 1, 2, \ldots} r^2(u(n-i), y_i(n))\ ,$ where $r^2(u(n-i), y_i(n))$ is the squared correlation coefficient between the input signal delayed by i and a trained output signal $y_i(n)$ which was trained on the task to retrodict (memorize) $u(n-i)$ on the input signal $u(n)\ .$ It turns out that for i.i.d. input, the memory capacity C of an echo state network of size N is bounded by $N\ ;$ in the absence of numerical errors and with a linear reservoir the bound is attained (Jaeger 2002a; White and Sompolinsky 2004; Hermans & Schrauwen 2009). These findings imply that it is impossible to train ESNs on tasks which require unbounded-time memory, like for instance context-free grammar parsing tasks (Schmidhuber et al. 2007). However, if output units with feedback to the reservoir are trained as attractor memory units, unbounded memory spans can be realized with ESNs, too (cf. the multistable switch example in Jaeger 2002a; beginnings of a theory of feedback-induced memory-hold attractors in Maass, Joshi & Sontag 2007; an ESN based model of working memory with stable attractor states in Pascanu & Jaeger 2010).

Universal computation and approximation properties. ESNs can realize every nonlinear filter with bounded memory arbitrarily well. This line of theoretical research has been started and advanced in the field of Liquid State Machines (Maass, Natschlaeger & Markram 2002; Maass, Joshi & Sontag 2007), and the reader is referred to the LSM article for detail.

Practical issues: tuning global controls and regularization

When using ESNs in practical nonlinear modeling tasks, the ultimate objective is to minimize the test error. A standard method in machine learning to get an estimate of the test error is to use only a part of the available training data for model estimation, and monitor the model's performance on the withheld portion of the original training data (the *validation set*). The question is, how can the ESN models be optimized in order to reduce the error on the validation set? In the terminology of machine learning, this boils down to the question how one can equip the ESN models with a task-appropriate *bias*. With ESNs, there are three sorts of bias (in a wide sense) which one should adjust.

The first sort of bias is to employ *regularization*. This essentially means that the models are smoothed. Two standard ways to achieve some kind of smoothing are the following:

- **Ridge regression** (also known as Tikhonov regularization): modify the linear regression equation (3) for the output weights:
  (4) $\mathbf{W}^{out} = (\mathbf{R} + \alpha^2 \mathbf{I})^{-1}\mathbf{P}\ ,$

where $\mathbf{R} = \mathbf{S}'\mathbf{S}$ is the correlation matrix of the extended reservoir states, $\mathbf{P} = \mathbf{S}'\mathbf{D}$ is the cross-correlation matrix of the states vs. the desired outputs, α^2 is some nonnegative number (the larger, the stronger the smoothing effect), and $\mathbf{I}$ is the identity matrix.

- **State noise**: During the state harvesting, instead of (1) use a state update which adds a noise vector $\mathbf{\nu}(n)$ to the reservoir states:
  (5) $\mathbf{x}(n+1) = f(\mathbf{W} \mathbf{x}(n) + \mathbf{W}^{in} \mathbf{u}(n+1) + \mathbf{W}^{fb} \mathbf{y}(n)) + \mathbf{\nu}(n)\ .$

Both methods lead to smaller output weights. Adding state noise is computationally more expensive, but appears to have the additional benefit of stabilizing solutions in models with output feedback (Jaeger 2002a; Jaeger, Lukosevicius, Popovici & Siewert 2007).

The second sort of bias is effected by making the echo state network, as one could say, "dynamically similar" to the system that one wants to model. For instance, if the original system is evolving on a slow timescale, the ESN should do the same; or if the original system has long memory spans, so should the ESN. This shaping of major dynamical characteristics is realized by adjusting a small number of *global control parameters*:

- The **spectral radius** of the reservoir weight matrix codetermines (i) the effective time constant of the echo state network (larger spectral radius implies slower decay of impulse response) and (ii) the amount of nonlinear interaction of input components through time (larger spectral radius implies longer-range interactions).
- The **input scaling** codetermines the degree of nonlinearity of the reservoir dynamics. In one extreme, with very small effective input amplitudes the reservoir behaves almost like a linear medium, while in the other extreme, very large input amplitudes drive the neurons to the saturation of the sigmoid and a binary switching dynamics results.
- The **output feedback scaling** determines the extent to which the trained ESN has an autonomous pattern generation component. ESNs without any output feedback are the typical choice for purely input-driven dynamical pattern recognition and classification tasks. The frequency generator demo task, on the other hand, needed strong output feedback to generate oscillations (which are not present in the input). Nonzero output feedback entails the danger of dynamical instability.
- The **connectivity** of the reservoir weight matrix is often claimed (starting with the early techreports of Jaeger) to be responsible for the "richness" of the response signals in the reservoir, following this line of reasoning: sparse connectivity $\to$ decomposition of reservoir dynamics into loosely coupled subsystems $\to$ large variation among the reservoir signals (desirable). However, contrary to this intuition, many authors have reported that fully connected reservoirs work as well as sparsely connected ones. Considering that sparsely but randomly connected networks have small-world properties, it appears plausible that a sparse random wiring does not lead to a dynamical

decoupling, so the original intuitions are misguided. A more practically important aspect of a sparse connectivity is that it engenders linear scaling of computational complexity. If reservoirs are set up such that each neuron on average connects to a fixed number \(K\) of other neurons, regardless of network size \(N\ ,\) the computational cost of running the trained networks grows only linearly with \(N\ .\)

Finally, a third sort of bias (here the terminology is stretched a bit) is simply the reservoir size \(N\ .\) In the sense of statistical learning theory, increasing the reservoir size is the most direct way of increasing the *model capacity*.

All these kinds of bias have to be optimized jointly. The current standard practice to do this is manual experimentation. Practical "tricks of the trade" are collected in Lukoševičius (2012).

Significance

A number of algorithms for the supervised training of RNNs have been known since the early 1990s, most notably *real-time recurrent learning* (Williams and Zipser 1989), *backpropagation through time* (Werbos 1990), *extended Kalman filtering* based methods (Puskorius and Feldkamp 2004), and the Atiya-Parlos algorithm (Atiya and Parlos 2000). All of these algorithms adapt all connections (input, recurrent, output) by some version of gradient descent. This renders these algorithms slow, and what is maybe even more cumbersome, makes the learning process prone to become disrupted by bifurcations (Doya 1992); convergence cannot be guaranteed. As a consequence, RNNs were rarely fielded in practical engineering applications at the time when ESNs were introduced. ESN training, by contrast, is fast, does not suffer from bifurcations, and is easy to implement. On a number of benchmark tasks, ESNs have starkly outperformed all other methods of nonlinear dynamical modelling (Jaeger and Haas 2004, Jaeger et al. 2007).

Today (as of 2017), with the advent of Deep Learning, the problems faced by gradient descent based training of recurrent neural networks can be considered solved. The original unique selling point of ESNs, stable and simple training algorithms, has dissolved. Moreover, deep learning methods for RNNs have proven effective for highly complex modeling tasks especially in language and speech processing. Reaching similar levels of complexity would demand reservoirs of inordinate size. Methods of reservoir computing are nonetheless an alternative worth considering when the modeled system is not too complex, and when cheap, fast and adaptive training is desired. This holds true for many applications in signal processing, as for example in biosignal processing (Kudithipudi et al. 2015), remote sensing (Antonelo 2017) or robot motor control (Polydoros et al. 2015).

Starting around 2010, echo state networks have become relevant and quite popular as a computational principle that blends well with non-digital computational substrates, for instance optical microchips (Vandoorne et al. 2014), mechanical nano-oscillators (Coulombe, York and Sylvestre 2017), memristor-based neuromorphic microchips (Bürger et al. 2015), carbon-nanotube / polymer mixtures (Dale et al. 2016) or even artificial soft limbs (Nakajima, Hauser and Pfeifer 2015). Such nonstandard computational materials and the microdevices made from them often lack numerical precision, exhibit significant device mismatch, and ways to emulate classical logical switching circuits are unknown. But, often nonlinear dynamics can be elicited from suitably interconnected ensembles of such elements -- that is, physical reservoirs can be built, opening the door for training such systems with ESN methods.

Online resources

A number of reservoir computing groups (among them the groups of Wolfgang Maass, Herbert Jaeger, Jochen Steil, Peter F. Dominey and Benjamin Schrauwen, i.e. the original proposers of the reservoir computing approach) have created a jointly administered Web portal for reservoir computing. It assembles introductions to the main flavors of the field, listings of groups active in the field, open-source programming toolboxes, and a mailing list. The portal is funded by the European FP7 project "Organic" and the University of Gent.

Patent

The Fraunhofer Institute for Intelligent Analysis and Information Systems claims international patents for commercial exploits of the ESN architecture and learning principle.

References

- Antonelo, E. A., Camponogara, E. and Foss, B. (2017). Echo State Networks for data-driven downhole pressure estimation in gas-lift oil wells. Neural Networks 85, 106-117.
- Atiya A.F. and Parlos A.G. (2000) New results on recurrent network training: Unifying the algorithms and accelerating convergence. IEEE Trans. Neural Networks, 11(3):697-709
- Buehner M. and Young P. (2006) A tighter bound for the echo state property. IEEE Transactions on Neural Networks, 17(3):820-824
- Buonomano, D.V. and Merzenich, M.M. (1995) Temporal Information Transformed into a Spatial Code by a Neural Network with Realistic Properties. Science 267, 1028-1030
- Bürger, J., Goudarzi, A., Stefanovic, D., and Teuscher, C. (2015, July) Hierarchical composition of memristive networks for real-time computing. In Nanoscale Architectures (NANOARCH), 2015 IEEE/ACM International Symposium on (pp. 33-38)
- Coulombe, J. C. and York, M. C. A. and Sylvestre, J. (2017). Computing with networks of nonlinear mechanical oscillators. PLOS ONE 12(6)

- Dale, M., Miller, J. F., Stepney, S., and Trefzer, M. A. (2016, July) Evolving carbon nanotube reservoir computers. In International Conference on Unconventional Computation and Natural Computation (pp. 49-61). Springer International Publishing
- Dominey P.F. (1995) Complex sensory-motor sequence learning based on recurrent state representation and reinforcement learning. Biol. Cybernetics, Vol. 73, 265-274
- Dominey P.F., Hoen M., and Inui T. (2006) A neurolinguistic model of grammatical construction processing. Journal of Cognitive Neuroscience 18(12), 2088–2107
- Doya K. (1992) Bifurcations in the learning of recurrent neural networks. In Proceedings of 1992 IEEE Int. Symp. on Circuits and Systems, Vol. 6, pages 2777-2780
- Manjunath, G. and Jaeger, H. (2012) Echo State Property Linked to an Input: Exploring a Fundamental Characteristic of Recurrent Neural Networks. Neural Computation, to appear
- Hermans M. and Schrauwen B. (2009) Memory in linear neural networks in continuous time. Neural Networks 23(3), 341-55
- Jaeger H. (2001a) The "echo state" approach to analysing and training recurrent neural networks. GMD Report 148, GMD - German National Research Institute for Computer Science
- Jaeger H. (2002a) Short term memory in echo state networks. GMD-Report 152, GMD - German National Research Institute for Computer Science
- Jaeger H. (2002b) Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the echo state network approach. GMD Report 159, Fraunhofer Institute AIS
- Jaeger H. (2003): Adaptive nonlinear system identification with echo state networks. In Advances in Neural Information Processing Systems 15, S. Becker, S. Thrun, K. Obermayer (Eds), (MIT Press, Cambridge, MA, 2003) pp. 593-600
- Jaeger H. (2007) Discovering multiscale dynamical features with hierarchical echo state networks. Technical report 10, School of Engineering and Science, Jacobs University
- Jaeger H. and Haas H. (2004) Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. Science, 304:78-80, 2004.
- Jaeger H., Lukoševičius M., Popovici D., and Siewert U. (2007) Optimization and applications of echo state networks with leaky integrator neurons. Neural Networks, 20(3):335-352
- Jaeger H., Maass W., and Principe J. (2007) Special issue on echo state networks and liquid state machines: Editorial. Neural Networks, 20(3), 287-289
- Kirby, K. (1991) Context dynamics in neural sequential learning. Proc. Florida AI Research Symposium (FLAIRS) 1991, 66-70
- Kudithipudi, D., Saleh, Q., Merkel, C., Thesing, J., and Wysocki, B. (2015) Design and Analysis of a Neuromemristive Reservoir Computing Architecture for Biosignal Processing. Frontiers in Neuroscience, 9, 502
- Lukoševičius M. (2012) A Practical Guide to Applying Echo State Networks. In: G. Montavon, G. B. Orr, and K.-R. Müller (eds.) Neural Networks: Tricks of the Trade, 2nd ed. Springer LNCS 7700, 659-686
- Lukoševičius M. and Jaeger H. (2009) Reservoir Computing Approaches to Recurrent Neural Network Training (draft version). Computer Science Review 3(3), 127-149
- Maass W., Joshi P., and Sontag E. (2007) Computational aspects of feedback in neural circuits. PLOS Computational Biology, 3(1):1-20
- Maass W., Natschlaeger T., and Markram H. (2002) Real-time computing without stable states: A new framework for neural computation based on perturbations. Neural Computation, 14(11):2531-2560
- Nakajima, K., Hauser, H. and Pfeifer, R. (2015) Information processing via physical soft body. Scientific Reports DOi: 10.1038/srep1048
- Pascanu R. and Jaeger H. (2010) A Neurodynamical Model for Working Memory. Neural Networks 24(2), 199-207
- Polydoros, A., Nalpantidis, L., and Krüger, V. (2015) Advantages and limitations of reservoir computing on model learning for robot control. IROS Workshop on Machine Learning in Planning and Control of Robot Motion, Hamburg, Germany
- Puskorius G.V. and Feldkamp L.A. (1994) Neurocontrol of nonlinear dynamical systems with Kalman filter trained recurrent network. IEEE Trans. Neural Networks, 5(2):279-297
- Schiller U.D. and Steil J. J. (2005) Analyzing the weight dynamics of recurrent learning algorithms. Neurocomputing, 63C:5-23
- Schmidhuber J., Gomez F., Wierstra D., and Gagliolo M. (2007) Training recurrent networks by evolino. Neural Computation, 19(3):757-779
- Vandoorne, K. and Mechet, P. and Van Vaerenbergh, T., et al. (2014) Experimental demonstration of reservoir computing on a silicon photonics chip. Nature Communications 5 (3541)
- Werbos P.J. (1990) Backpropagation through time: what it does and how to do it. Proc. of the IEEE, October, 78(10, October):1550-1560
- White O.L., Lee D.D., and Sompolinsky H.S. (2004) Short-term memory in orthogonal neural networks. Phys. Rev. Lett., 92(14):148102
- Williams R.J. and Zipser D. (1989) A learning algorithm for continually running fully recurrent neural networks. Neural Computation, 1:270-280
- Yildiz, I. B., Jaeger, H. and Kiebel, S. J. (2012) Re-visiting the echo state property. Neural Networks 35, 1-20

Internal references

- John W. Milnor (2006) Attractor. Scholarpedia, 1(11):1815.
- Valentino Braitenberg (2007) Brain. Scholarpedia, 2(11):2918.
- James Meiss (2007) Dynamical systems. Scholarpedia, 2(2):1629.
- Philip Holmes and Eric T. Shea-Brown (2006) Stability. Scholarpedia, 1(10):1838.

See Also

Liquid State Machine, Recurrent Neural Networks, Supervised Learning

Sponsored by: Eugene M. Izhikevich, Editor-in-Chief of Scholarpedia, the peer-reviewed open-access encyclopedia
Reviewed by: Dr. Hava T. Siegelmann, Computer Science, University of Massachusetts, Amherst, MA
Reviewed by: Dr. Benjamin Schrauwen, University Ghent, Belgium
Accepted on: 2007-09-06 07:00:11 GMT

Categories: Pattern Recognition | Computational Intelligence | Spiking Networks | Recurrent Neural Networks | Neural Networks