Computational Intelligence & Machine Learning
http://www.di.unipi.it/groups/ciml
Dipartimento di Informatica
Università di Pisa - Italy

Neural Modeling and Computational Neuroscience

Claudio Gallicchio

Neuroscience modeling

- Introduction to basic aspects of brain computation
- Introduction to neurophysiology
- Neural modeling:
  - Elements of neuronal dynamics
  - Elementary neuron models
  - Neuronal Coding
  - Biologically detailed models:
    the Hodgkin-Huxley Model
  - Spiking neuron models, spiking neural networks
  - Izhikevich Model
- Introduction to Reservoir Computing and Liquid State Machines
- Introduction to glia and astrocyte cells, the role of astrocytes in a computational brain, modeling neuron-astrocyte interaction, neuron-astrocyte networks,
- The role of computational neuroscience in neuro-biology and statistics for In-vitro neuro-astrocyte culture.

Neuroscience modeling

- Introduction to basic aspects of brain computation
- Introduction to neurophysiology
- Neural modeling:
  - Elements of neuronal dynamics
  - Elementary neuron models
  - Neuronal Coding
  - Biologically detailed models: the Hodgkin-Huxley Model
  - Spiking neuron models, spiking neural networks
  - Izhikevich Model
- Introduction to Reservoir Computing and Liquid State Machines
- Introduction to glia and astrocyte cells, the role of astrocytes in a computational brain, modeling neuron-astrocyte interaction, neuron-astrocyte networks,
- The role of computational neuroscience in neuro-biology and statistics for In-vitro neuro-astrocyte culture.

Models of Neural Networks

Networks of Neurons

- Extensive connectivity among neurons is a major characterization of the brain computation
- Neocortical circuits: layered recurrent circuits
  - neurons lie in 6 layers
  - connectivity among cortical columns structures
  - feed-forward connections: signal pathways to higher stages of computation
  - recurrent connections:
    - signal feedbacks interconnecting neurons at the same stage of computation
    - top-down interconnections between areas in different stages of computation

Networks of Neurons

Simulate a biological neural network:

- Interconnect spiking neurons in a biologically plausible fashion
- Mathematical models of spiking neurons (studied so far) can be used to this purpose
  - Hodgkin-Huxley, Integrate-and-fire, Leaky Integrate-and-Fire, Izhikevich, …
- Neural coding: often firing-rate models are used

Networks of Spiking Neurons

- 3 generations of neuron models
- First Generation
  - McCulloch-Pitts neurons
  - Based on perceptrons and threshold gates
  - Digital output
- Second Generation
  - Neuron models based on activation functions (sigmoid, linear saturated, ...)
  - Continuous output
  - Firing-rate models (the output can be interpreted as the firing rate of a biological neuron)

Networks of Spiking Neurons

- Third Generation
  - Timing of single action potential used to encode information
  - Spiking neurons (e.g. integrate-and-fire models)
  - Simplified models of action potential generation
    - closer than 1st and 2nd generation models to the biological neurons
    - simulate the dynamical behavior of neurons
    - focus only on few aspects of biological neurons

      (e.g. modeling fast activation/slow inactivation of Na^+ channels)
  - More Complex
    - More computationally powerful
      - Relevant biological functions that can be computed by 1 spiking neuron might require hundreds of sigmoidal hidden units
    - More difficult to train

Mathematical Models of Neural Networks

- Neuroscience
  - Research tool to validate the models of brain functioning
  - Useful to explain and do predictions on the way in which biological neural networks operate
- Machine Learning
  - Use these computational models to solve problems
  - Temporal Problems
  - Learning in temporal domains is computational intensive
  - Efficiency has a major role

Liquid Computing

Repetita

- Dynamical Systems
  - Neurons implement input-driven non autonomous dynamical systems
  - Neurons are excitable because their state is close to a bifurcation
- The role of time
  - Delayed connectivity among neurons
- The role of randomness
  - Neurons are connected to each other according to a pattern of stochasticity
  - Edelman’s theory of neuronal group selection

Notation (disclaimer)

A slightly different notation than what used in previous lectures (caution)

- Input

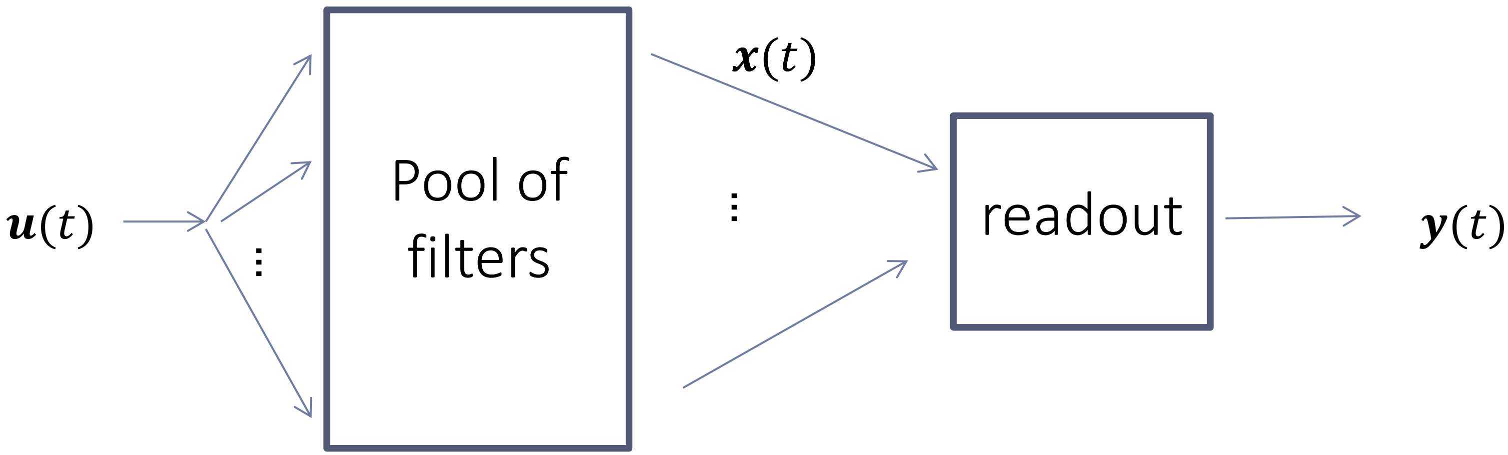
  $\boldsymbol{u}(t)$
- State

  $\boldsymbol{x}(t)$
- Output

  $\boldsymbol{y}(t)$

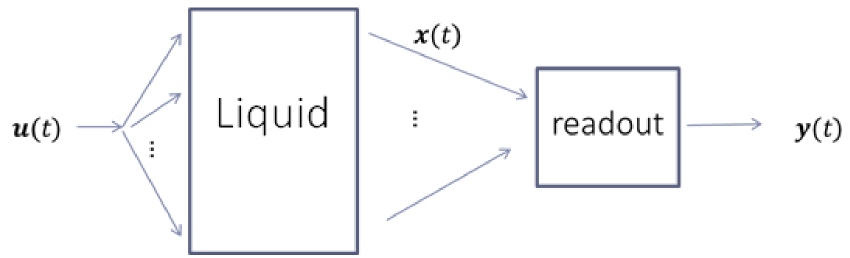
Real-time Computing with a Liquid Medium

- Objective: perform a temporal task in real-time
- Idea:
  - encode the input history into a pool of dynamical systems/filters
  - use such pool as input for the output computation



Real-time Computing with a Liquid Medium

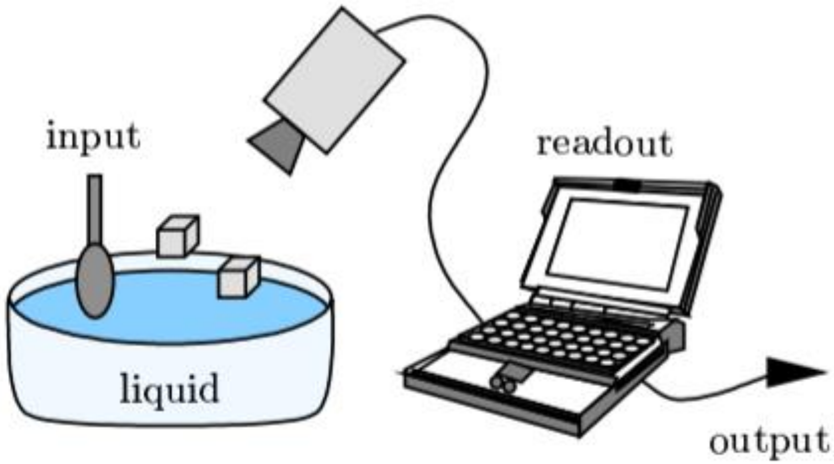
- How to implement the filters?
  - Metaphor: use a liquid….



  - Imagine throwing a stone into a pool of water
  - The waves and how they propagate can tell something on the stone stimulus to the water
  - The interaction among the waves can tell us something on the history of thrown stones
  - The state of the water can be useful to differentiate among different (recent) histories of stones throwing stimuli

Real-time Computing with a Liquid Medium

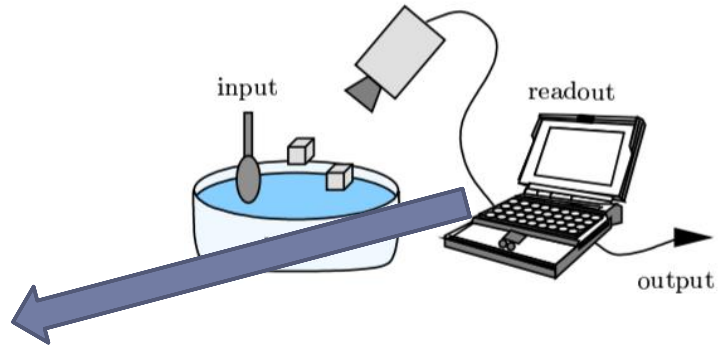
Liquid Computers

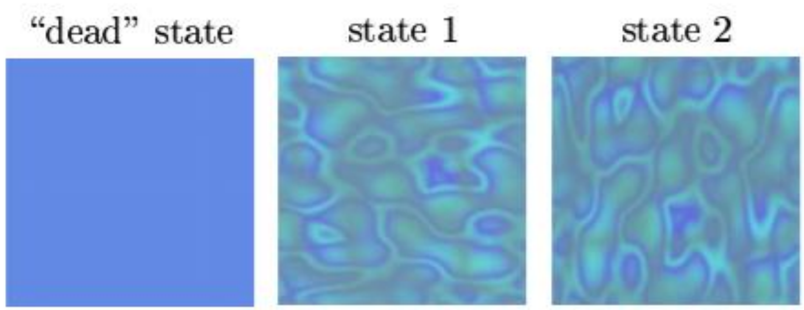


- Input time series
  - Sequence of perturbations applied to the liquid, e.g. encoded by the pattern of spoon hits
- Liquid states
  - The surface of the liquid encodes the history of the spoon perturbations
  - Like a state machine, but with a *liquid state*…
- Readout
  - Has no memory
  - Transforms the liquid state into the desired output value/time series (e.g. a classification of the source of the perturbation)

Real-time Computing with a Liquid Medium

- Liquid States
  - Non-autonomous system
  - Stable states are not of interest

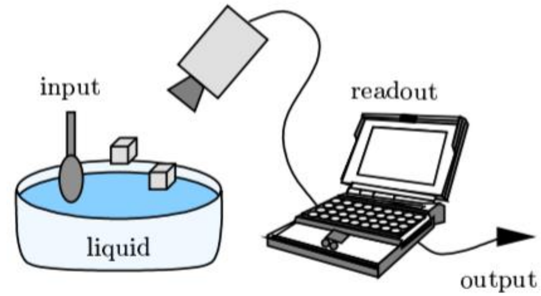


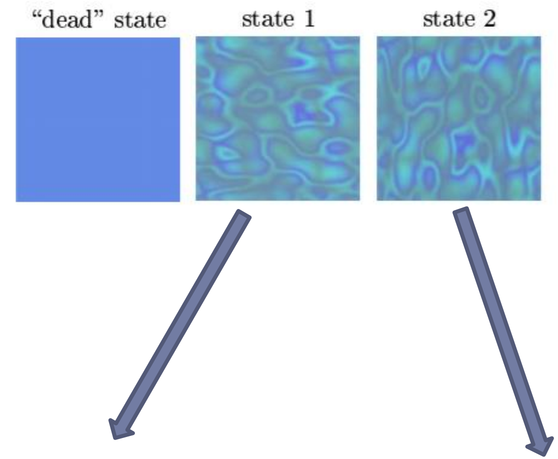


- Output computation
  - Memory-less: at each moment the output depends only on the liquid state in that moment
  - Assumption: at each time, the liquid contains all the relevant information on the input history

Real-time Computing with a Liquid Medium

- Richness
  - The liquid should provide a rich reservoir of possibly diverse representations of the input history
  - A rich pool of temporal filters
- Randomness
  - Random temporal filters are suitable to the purpose as long as they provide rich/diverse enough temporal dynamics



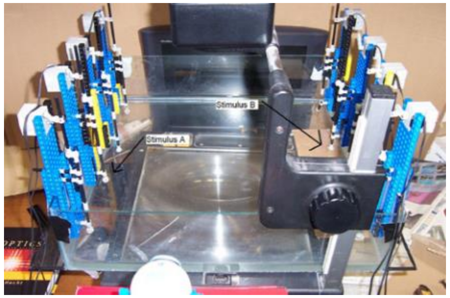


Pattern 1 of spoon hits

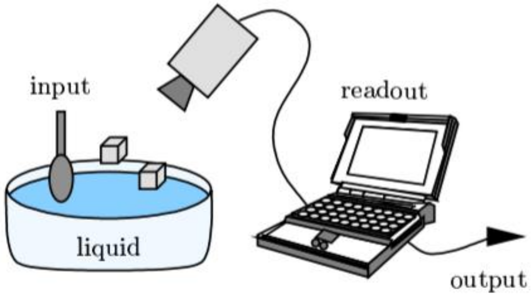
Pattern 2 of spoon hits

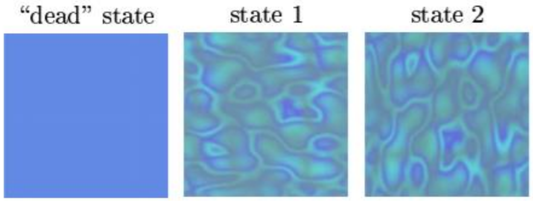
Real-time Computing with a Liquid Medium

- Exotic Implementations of the idea



F. Chrisantha, S. Sojakka. "Pattern recognition in a bucket." European Conference on Artificial Life, 2003.

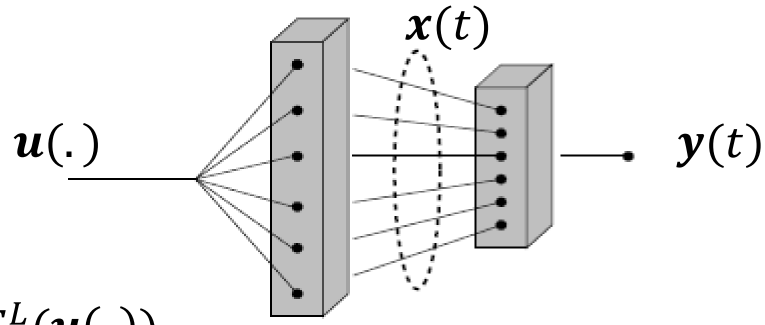




- Neural circuits can constitute ideal liquids
  - Distributed (temporal) interactions among the neurons
  - Variety of time-scales developed by a network of interconnected neurons

Liquid State Machine (LSM)

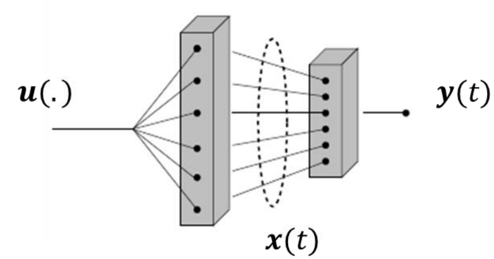
- Mathematical model of the Liquid Computer



- Liquid: $\boldsymbol{x}(t) = F^L(\boldsymbol{u}(.))$
  - Implements an input-driven dynamical system
  - Pool of basis filters: basis expansion
  - A state machine, but with continuous state
- Readout: $\boldsymbol{y}(t) = F^R(\boldsymbol{x}(t))$
  - Implements a non-temporal classifier/regressor

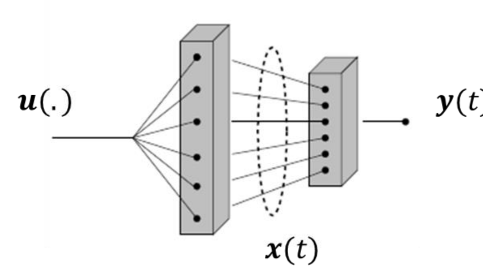
Liquid State Machine (LSM)

- Temporal filters through the liquid have two major properties:
  - Time-invariant
    a temporal shift of the input determines a temporal shift of the output of the filters of the same amount
  - Fading memory
    the output of the filters for an input sequence *u1* can be approximated by the output of the filters for another input sequence *u2*, if *u2* approximates well *u1* over a long time interval
    - For long input histories the output of the filters depend only on the most recent inputs



Liquid State Machine (LSM)

- Temporal filters through the liquid have two major properties:
  - Time-invariant
    a temporal shift of the input determines a temporal shift of the output of the filters of the same amount
  - Fading memory
    the output of the filt ... approximated by the ... ut sequence *u2*, if *u2* approximates well *u1* over a long time interval
    - For long input histories the output of the filters depend only on the most recent inputs



Suffix-based Markovian organization of the state space

Liquid State Machine (LSM)

- Pointwise separation property (Liquid)
  - Suppose there are 2 sequences s_u and s_v, which differ before a time step t_1
    $t < t_1 \colon s_u(t) \neq s_v(t)$
  - There exist a basis filter in the class of considered basis filters such that
    $F^L(s_u(\ldots, t_1)) \neq F^L(s_v(\ldots, t_1))$
- Universal approximation property (Readout)
  - Any continuous function on a compact domain can be uniformly approximated

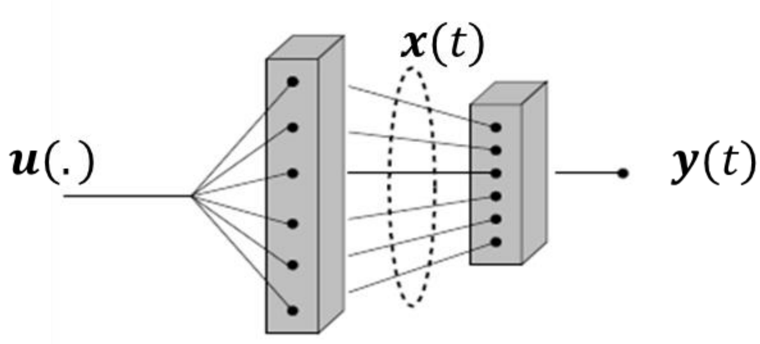
Liquid State Machine (LSM)

Theorem

A Liquid State Machine can implement any time-invariant temporal filter with fading memory, provided that

- the liquid satisfies the pointwise separation property
- the readout satisfies the universal approximation property

Liquid State Machine (LSM)

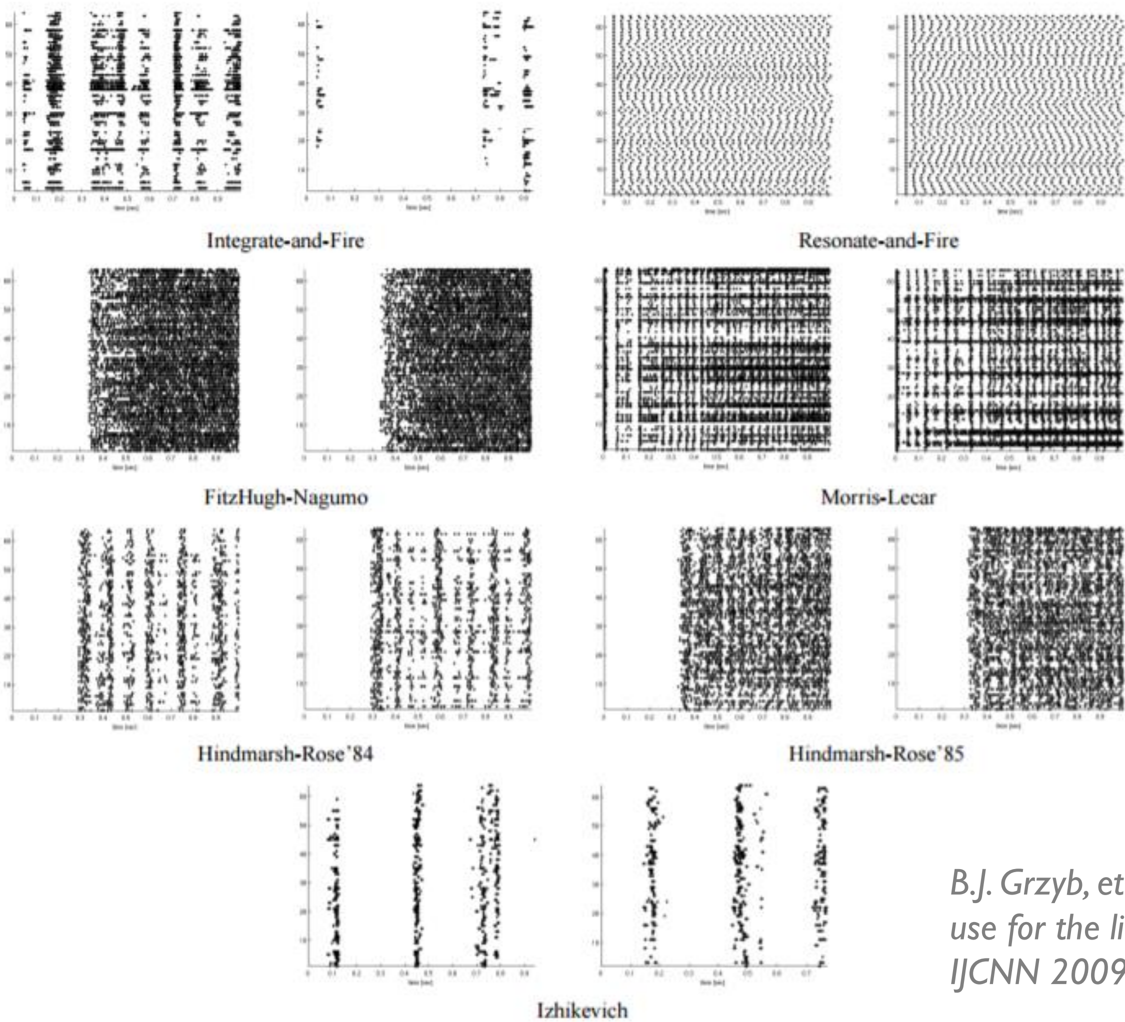


- The liquid does not need to be trained
- Training can be restricted only to the readout
- What to use for the readout?
  - Any classification or regression tool
  - Provided that the liquid gives a rich transformation of the temporal input stream a **linear** readout can be used
  - Extreme efficiency of the approach!

Which model to use for the LSM?

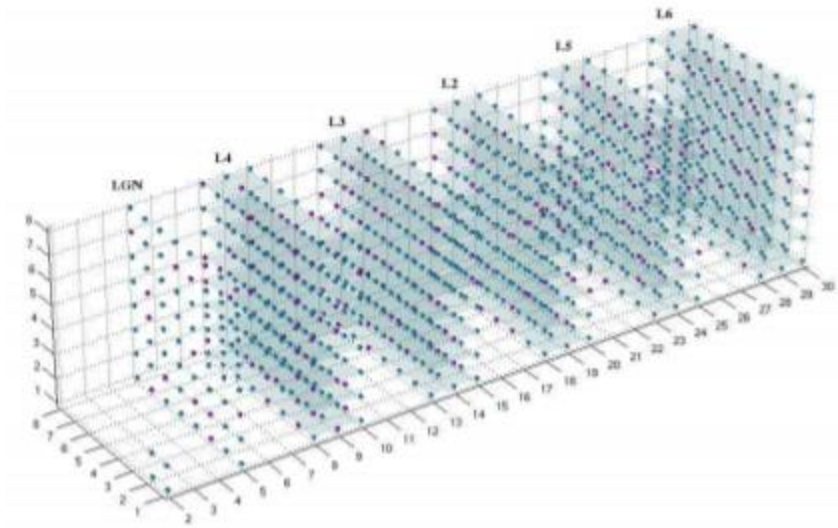
- Mathematical models of neural microcircuits are suitable to implement the liquid
- Microcircuits are characterized by large diversity of mechanisms involved in temporal spike generation
- Liquid: a layer of interconnected neurons
  - Integrate-and-fire
  - Resonate-and-fire
  - FitzHugh-Nagumo
  - Morris-Lecar
  - Izhikevich
  - ....

Which model to use for the LSM?



B.J. Grzyb, et al. "Which model to use for the liquid state machine?." IJCNN 2009, IEEE, 2009.

Which model to use for the LSM?



- Pattern of connectivity among the neurons are taken from biologically plausible setups
- E.g. model of mammalian visual systems
  - 6 layers + input (retina layer)

Implementation of Liquid State Machines

- Liquid
  - A layer of randomly interconnected spiking neurons (a microcircuit model)
  - Connectivity follows biologically plausible patterns
  - Typically untrained (or adapted through the STDP plasiticity rule)
- Readout
  - Any classification/regression model (perceptron, spiking neuron, MLP, SVM, etc.)
  - Training with
    - delta rule, backpropagation, linear regression, p-delta rule, etc....
- Neural coding: the liquid state can be
  - Roughly, the spiking/non-spiking activity pattern of each neuron in the liquid
  - Temporal coding: firing-rate

Online Resources

- Website by the group who proposed the LSM model @ the Graz University of Technology

  http://www.lsm.tugraz.at/

- Software
  - Learning-Tool: Analysing neural microcircuit (NMC) models
  - Matlab implementation
- Literature references
  - http://www.lsm.tugraz.at/references.html

A broader look: Randomized Neural Networks

- Initialize some of the weights with random values
- Leave untrained some of the connections in the neural network architecture
- Historical models: the Gamba-perceptron
- Randomized NN have 2 components
  - Untrained hidden layer
    - Non-linearly embed the input into a high-dimensional feature space by means of a randomized basis expansion
    - In such state space the original problem is more likely to be linearly solved (Cover's Theorem)
  - Trained Readout layer
    - Typically linear output layer

Trained efficiently!!!!!

A broader look: Randomized Neural Networks

- Feed-forward Randomized NNs

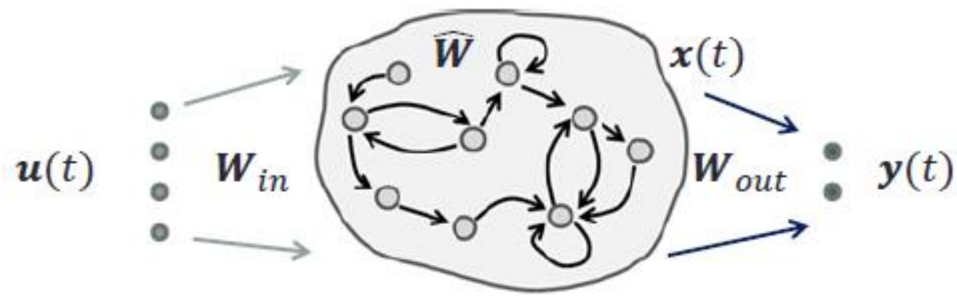
$$\mathbf{y} = \begin{bmatrix} \sum_{j=1}^{N_X} w_{1,j}^{out} f(\mathbf{w}_j \mathbf{u}) \\ \dots \\ \sum_{j=1}^{N_X} w_{N_Y,j}^{out} f(\mathbf{w}_j \mathbf{u}) \end{bmatrix} = \mathbf{W}^{out} f(\mathbf{W}\,\mathbf{u}).$$

- Recurrent Randomized NNs

$$\mathbf{y}(t) = \begin{bmatrix} \sum_{j=1}^{N_X} w_{1,j}^{out} f(\mathbf{w}_j^{in}\mathbf{u}(t) + \hat{\mathbf{w}}_j \mathbf{x}(t-1)) \\ \dots \\ \sum_{j=1}^{N_X} w_{N_Y,j}^{out} f(\mathbf{w}_j^{in}\mathbf{u}(t) + \hat{\mathbf{w}}_j \mathbf{x}(t-1)) \end{bmatrix} = \mathbf{W}^{out} f(\mathbf{W}^{in}\mathbf{u}(t) + \hat{\mathbf{W}}\mathbf{x}(t-1))$$

$$= \mathbf{W}^{out}\mathbf{x}(t)$$

Reservoir Computing



- Reservoir
  - Liquid State Machines: a layer of spiking neurons
  - Echo State Networks: a layer of untrained sigmoidal units (provided that some conditions are satisfied……)
- Readout
  - Only part that is trained
  - Moore-Penrose Pseudo-inverse, Ridge Regression, ...