Machine Learning results for Spiking Neuron Networks

Hélène Paugam-Moisy

INRIA - TAO - LRI Orsay / Université de Lyon

Reference : rapport RR-06-11 - IDIAP Research Institute, Martigny - Switzerland

TAO seminar - November 4th, 2008

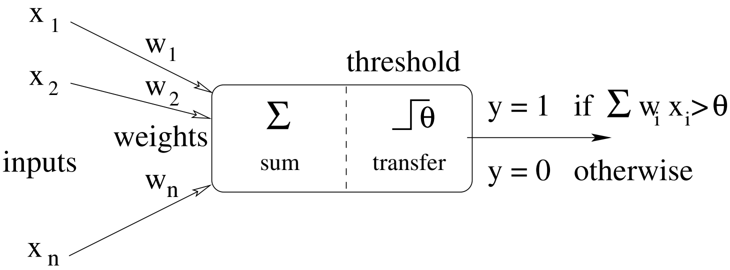
Plan

1. Spiking Neuron Networks (SNNs)
   - Models of neurons
   - Models of networks
2. Information coding
   - Practical issues
   - Combinatorial point of view
3. Theoretical results
   - Complexity
   - Learnability
4. Learning and plasticity
   - Learning rules for SNNs
   - Synaptic plasticity

Plan

Traditional models of neurons

Traditional neural network algorithms (learning, recognition) iteratively compute neurons such that the McCulloch & Pitts model :



Inputs x_i represent mean firing rates of presynaptic neurons N_i that correspond to frequencies (average count of pulses) over long time.

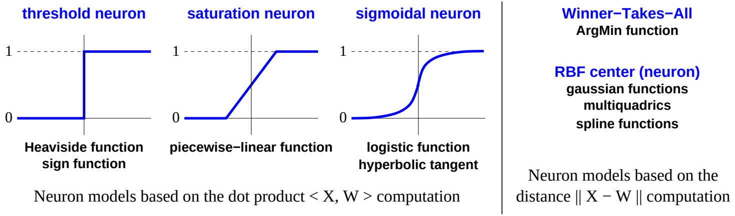
Traditional models of neurons

Scalar product models = computing weighted sum of inputs :

$$Y = f(< X, W >)$$

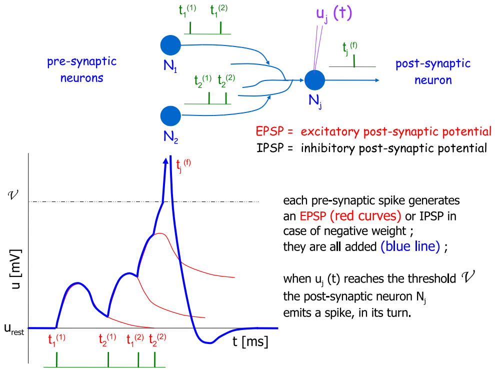
threshold units, linear cells, sigmoid neurons

Distance neurons, computing : $Y = \phi(\| X - W \|)$



Importance of time

From biological evidence, time is a central feature in cognitive processing.

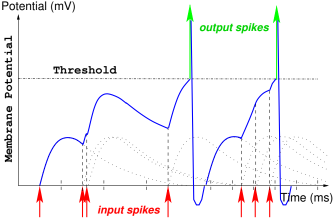


Precise spike timing
plays
a fundamental role
in information coding

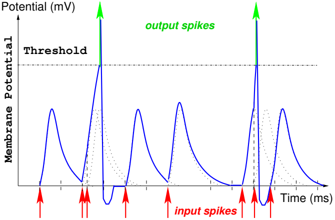
action potential
= pulse
= spike

Temporal properties of spiking neurons

Depending on the neuron parameters, the membrane potential can obey different laws ⇒ the neuron is able to behave different ways :



Integrator



Coincidence detector

[left] most all input spikes participate to postsynaptic firing, whereas [right] only quasi-synchronously incoming spikes trigger an output.

Models of spiking neurons

- Model of Hodgkin-Huxley (HH model) [since 1952]

Models of spiking neurons

- Model of Hodgkin-Huxley (HH model) [since 1952]

$$C\frac{du}{dt} = -g_{Na}m^3h(u - E_{Na}) - g_K n^4(u - E_K) - g_L(u - E_L) + I(t)$$

$$\tau_n \frac{dn}{dt} = -[n - n_0(u)]$$

$$\tau_m \frac{dm}{dt} = -[m - m_0(u)]$$

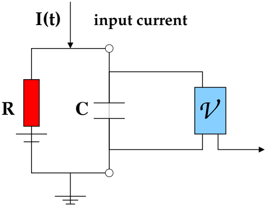
$$\tau_h \frac{dh}{dt} = -[h - h_0(u)]$$

Models of spiking neurons

- Model of Hodgkin-Huxley (HH model) [since 1952]
- Integrate-and-Fire model (IF, LIF, QIF...) [since Lapicque, 1907 !]

Models of spiking neurons

- Model of Hodgkin-Huxley (HH model) [since 1952]
- Integrate-and-Fire model (IF, LIF, QIF...) [since Lapicque, 1907 !]



$$C\frac{du}{dt} = -\frac{1}{R}u(t) + I(t)$$

spike

timing $t^{(f)}$ defined by $u(t^{(f)}) = \vartheta$ with $u'(t^{(f)}) > 0$

Models of spiking neurons

- Model of Hodgkin-Huxley (HH model) [since 1952]
- Integrate-and-Fire model (IF, LIF, QIF...) [since Lapicque, 1907 !]
- Gerstner's Spike Response Model (SRM, SRM$_0$) [$\approx$1998]

Models of spiking neurons

- Model of Hodgkin-Huxley (HH model) [since 1952]
- Integrate-and-Fire model (IF, LIF, QIF...) [since Lapicque, 1907 !]
- Gerstner's Spike Response Model (SRM, SRM_0) [≈1998]

$$u_j(t) = \sum_{t_j^{(f)} \in \mathcal{F}_j} \eta_j \left(t - t_j^{(f)} \right) + \sum_{i \in \Gamma_j} \sum_{t_i^{(f)} \in \mathcal{F}_i} w_{ij} \epsilon_{ij} \left(t - t_i^{(f)} \right) + \underbrace{\int_0^\infty \kappa_j(r) I(t-r) dr}_{\text{if external input current}}$$

or simpler model SRM_0 (very close to traditional formula)

$$u_j(t) = \sum_{i \in \Gamma_j} w_{ij} \epsilon(t - t_i^{(f)} - \delta_{ij}^{ax})$$

Models of spiking neurons

- Model of Hodgkin-Huxley (HH model) [since 1952]
- Integrate-and-Fire model (IF, LIF, QIF...) [since Lapicque, 1907 !]
- Gerstner's Spike Response Model (SRM, SRM_0) [$\approx$1998]
- Izhikevich's neuron model [2003]

Models of spiking neurons

- Model of Hodgkin-Huxley (HH model) [since 1952]
- Integrate-and-Fire model (IF, LIF, QIF...) [since Lapicque, 1907 !]
- Gerstner's Spike Response Model (SRM, SRM_0) [≈1998]
- Izhikevich's neuron model [2003]

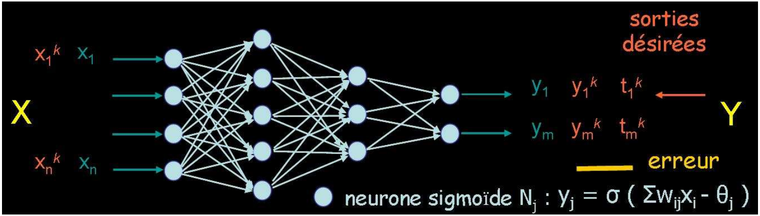
$$\frac{du}{dt} = 0.04u(t)^2 + 5u(t) + 140 - w(t) + I(t)$$

$$\frac{dw}{dt} = a\,(bu(t) - w(t))$$

Plan

Traditional Artificial Neural Networks (ANNs)

Network topologies with feedforward dynamics : one-layer (e.g. Perceptron, Kohonen maps), multilayer networks (e.g. MLP, RBF)



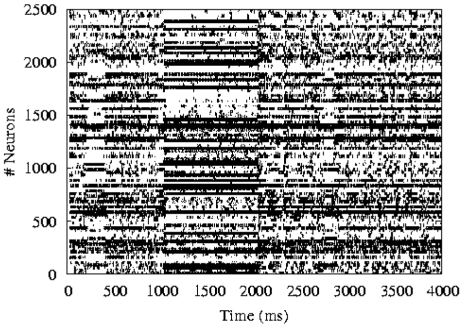
Network topologies with recurrent dynamics = non-acyclic graphs : lateral connections (e.g. cortical models), complete interconnection (e.g. Hopfield network)

Machine Learning results for traditional ANNs

- Calculability : ANNs are more powerful than Turing machine
- Complexity : the "loading problem" is NP-complete
- Capacity : MLP and RBF are universal approximators
- PAC-learning (Probably Approximately Correct) [Valiant, 1984]
- Statistical Learning Theory [Vapnik, 1995], kernel trick, SVM

Temporal behaviour of Spiking Neuron Networks

Spike raster plot : a bar each time a neuron spikes, one line per neuron.
$\Rightarrow$ outputs are spatiotemporal patterns of spike emission



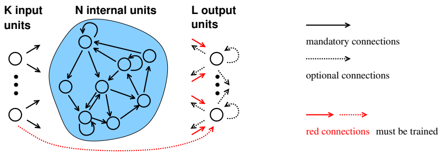
[Meunier, 2006]

Background activity $[0 - 1000ms]$, formation of a synchronized neural assembly in response to an input stimulus $[1000 - 2000ms]$ and disruption of cell assembly, when input is removed $[2000 - 4000ms]$.

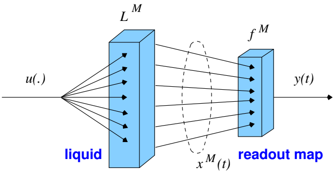
Reservoir computing

Two recent architectures of spiking neuron networks (SNNs) are the fundations of **Reservoir Computing** :

- the Echo State Network - H. Jaeger (2001) Report TR-GMD-148
- the Liquid State Machine - W. Maass, T. Natschläger, H. Markram (2002) Neural Computation 14(11)



Echo State Network



Liquid State Machine

Reservoir computing

Two architectures of spiking neuron networks have been proposed recently :

- the Echo State Network - H. Jaeger (2001) Report TR-GMD-148
- the Liquid State Machine - W. Maass, T. Natschläger, H. Markram (2002) Neural Computation 14(11)

Rough sketch :
an input layer is linked to a randomly connected recurrent neural network, the **reservoir**, followed by an output layer of readout neurons, with a very simple learning rule (e.g. linear regression).

Reservoir computing

Two architectures of spiking neuron networks have been proposed recently :

- the Echo State Network - H. Jaeger (2001) Report TR-GMD-148
- the Liquid State Machine - W. Maass, T. Natschläger, H. Markram (2002) Neural Computation 14(11)

Rough sketch :
an input layer is linked to a randomly connected recurrent neural network, the **reservoir**, followed by an output layer of readout neurons, with a very simple learning rule (e.g. linear regression).

The *trick* is to extract information from a free self-organization generated inside the reservoir by an external input stimulus.

Plan

Temporal coding

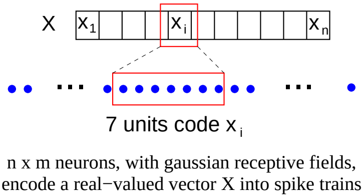
How to code real-valued information for further efficient computation by spiking neuron networks ? How to decode output spike trains without loss of temporal information ?

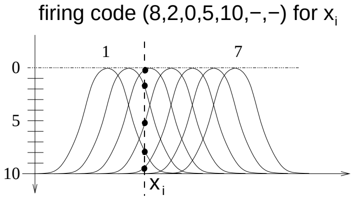
A straightforward translation of real numbers into spikes is temporal coding in a given time window T :
$\forall i \quad t_i = T - x_i$ earliest spikes for most intensive / salient data

Simple, but lose many precious features of temporal computing with spikes.

Population coding

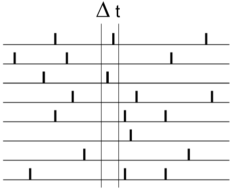
More realistic, but more complex, coding methods exist, e.g. based on a population of neurons for coding each real-value component of the input signal, with overlapping gaussian receptive fields.





Event-driven simulation

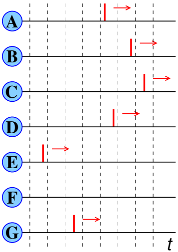
An SNN has a sparse network topology and its dynamics displays a low average activity $\Rightarrow$
for simulating large-scale SNNs (several billions of synaptic connections), **Event-Driven** programming substantially reduces the computational cost, both in time and memory.



Time-driven simulation (checking each connection at each time step) would yield 97% useless computation !

Plan

Capacity comparison



	count	latency	rank
A	1	3	4
B	1	2	2
C	1	1	1
D	1	3	3
E	1	7	6
F	0	–	–
G	1	5	5

- Count code : $6/7$ spike per 7 ms, i.e. ≈ 122 spikes.s^{-1}
- Binary code : 1111101
- Timing code : latency, here with a 1 ms precision
- Rank order code : $C > B > D > A > G > E > F$

Number of bits that can be transmitted by n neurons in a T time window.

Numeric examples :	count code	binary code	timing code	rank order
upper left figure $n = 7, T = 7ms$	3	7	≈ 19	12.3
Thorpe et al. $n = 10, T = 10ms$	3.6	10	≈ 33	21.8

Counting code - Binary code

Hypothesis :
a set of n neurons, each firing at most once in a T ms time window.
Goal : evaluate the information coding **capacity**

Counting code

- counting the overall number of spikes gives a $\log_2(n+1)$ maximum amount of available information

Counting code - Binary code

Hypothesis :
a set of n neurons, each firing at most once in a T ms time window.
Goal : evaluate the information coding **capacity**

Counting code

- counting the overall number of spikes gives a $\log_2(n+1)$ maximum amount of available information

Binary code

- considering the output as a n-digits binary code gives a n information coding capacity

Timing code - Rank order code

Hypothesis :
a set of n neurons, each firing at most once in a T ms time window.
Goal : evaluate the information coding **capacity**

Timing code

- determining the precise times of each spike, with a 1 ms precision gives a $n \times \log_2(T)$ amount of available information transmitted in the T time window

Timing code - Rank order code

Hypothesis :
a set of n neurons, each firing at most once in a T ms time window.
Goal : evaluate the information coding **capacity**

Timing code

- determining the precise times of each spike, with a 1 ms precision gives a $n \times \log_2(T)$ amount of available information transmitted in the T time window

Rank Order Code

- considering the order of the sequence of spike firing, since there are $n!$ possible orders, the capacity is $\log_2(n!)$

Plan

1. Spiking Neuron Networks (SNNs)
   - Models of neurons
   - Models of networks
2. Information coding
   - Practical issues
   - Combinatorial point of view
3. Theoretical results
   - Complexity
   - Learnability
4. Learning and plasticity
   - Learning rules for SNNs
   - Synaptic plasticity

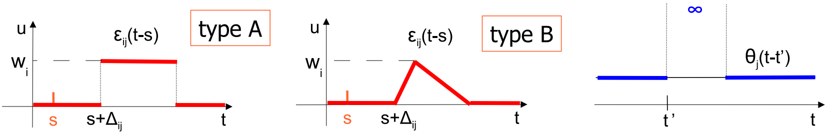
Generations of neural network models

W. Mass, 1997

- 1st generation : threshold units, with only digital outputs
  Perceptron, Hopfield network, Boltzmann machine, multilayer networks of threshold neurons
- 2nd generation : units with a continuous set of possible outputs
  MLP, RBF networks, wavelet networks, SVM rate coding
- 3rd generation : units = spiking neurons
  ESN, LSM, BPDC, pulse stream VLSI time coding

Complexity

For theoretical study purpose, Maass has defined two extremely simple models of neurons (note the presence of variable delays Δ_{ij}) :



EPSP response functions to a spike emitted at time s (connection parameters : weight w_i, axonal delay Δ_{ij})

threshold modelling

For **binary** inputs, a type_A neuron is at least as powerful as a threshold gate.
For arbitrary **real-valued** inputs, any threshold gate can be computed by $O(1)$ type_B neurons.

Complexity

CD_n, the Coincidence Detection Function of two n-Boolean vectors is

$$CD_n(x_1, ..., x_n, y_1, ..., y_n) = \begin{cases} 1, & \text{if } (\exists i) x_i = y_i \\ 0, & \text{otherwise} \end{cases}$$

Only 1 type_A neuron is sufficient for computing CD_n whereas $O(n^{1/4})$ sigmoidal and $O(n/\log n)$ threshold neurons are required.

Complexity

CD_n, the Coincidence Detection Function of two n-Boolean vectors is

$$CD_n(x_1, ..., x_n, y_1, ..., y_n) = \begin{cases} 1, & \text{if } (\exists i) x_i = y_i \\ 0, & \text{otherwise} \end{cases}$$

Only 1 type_A neuron is sufficient for computing CD_n whereas $O(n^{1/4})$ sigmoidal and $O(n/\log n)$ threshold neurons are required.

ED_n, the Element Distinctness Function with n real-valued inputs, is

$$ED_n(x_1, ..., x_n) = \begin{cases} 1, & \text{if } (\exists i \neq j) x_i = x_j \\ 0, & \text{if } (\forall i \neq j) \mid x_i x_j \mid \geq 1 \\ \text{arbitrary}, & \text{otherwise} \end{cases}$$

Only 1 type_A neuron is sufficient for computing ED_n whereas $O(n)$ sigmoidal and $O(n \log n)$ threshold neurons are required.

SNNs are universal approximators

Maass (2001) introduces the model of noisy spiking neuron, a variant of the SRM neuron, with a probability of spontaneous firing.

Any given feedforward sigmoidal network of s units with linear saturated activation function can be simulated by a network of $s + 0(1)$ noisy spiking neurons.

$\Rightarrow$ SNNs are **universal approximators**.

Several other complexity results have been produced by Maass and Schmitt (1997, 1998).

Plan

VC_dimension of spiking neurons

- $VC_{dim}(IF)$ grows in $\log B$ with the input signal bandwidth B
  [Zador, Pearlmutter, 1996]
- with m variable positive delays, $VC_{dim}(type\_A\ neuron)$ is $\Omega(m \log m)$, even with fix weights whereas, with m variable weights, $VC_{dim}(threshold\ gate)$ is $\Omega(m)$ only
  [Maass, Schmitt, 1997]
- the classes of boolean functions, with n inputs (boolean or real) and 1 boolean output that can be computed by a spiking neuron have VC_dimension $\Theta(n \log n)$
  [Maass, Schmitt, 1999]

VC_dimension of spiking neuron networks

- The peudo-dimension of an SNN with W parameters (weights, delays, …), depth D and rational synaptic interactions with degree no larger than p, is $O(WDlog(WDp))$.

  For fixed depth D and degree p, this entails the bound $O(Wlog(W))$.

- The pseudo-dimension of an SNN with W parameters and rational synaptic interactions with degree no larger than p, but with arbitrary depth, is bounded by $O(W^2 log(p))$.

  The pseudo-dimension is $\Theta(W^2))$ if the degree p is bounded by a constant.

PAC learnability

The consistency problem : Given a set of labelled binary n-input examples (X, b), $\exists$ parameters defining a spiking neuron $\mathcal{N}$ s.t. $(\forall (X, b))\ y_{\mathcal{N}}(X) = b$?

PAC learnability

The consistency problem : Given a set of labelled binary n-input examples (X, b), $\exists$ parameters defining a spiking neuron $\mathcal{N}$ s.t. $(\forall (X, b))\ y_{\mathcal{N}}(X) = b$?

Maass, Schmitt, 1999

- The *consistency problem* for a spiking neuron with binary delays is $NP\_complete$.
- The *consistency problem* for a spiking neuron with binary delays and fix weights is $NP\_complete$.

PAC learnability

The consistency problem : Given a set of labelled binary n-input examples (X, b), $\exists$ parameters defining a spiking neuron $\mathcal{N}$ s.t. $(\forall (X, b))\ y_{\mathcal{N}}(X) = b$?

Maass, Schmitt, 1999

- The *consistency problem* for a spiking neuron with binary delays is $NP\_complete$.
- The *consistency problem* for a spiking neuron with binary delays and fix weights is $NP\_complete$.

Šíma and Sgall, 2005

- The *consistency problem* for a spiking neuron with nonnegative delays is $NP\_complete$.
- The *representation problem* for spiking neurons is $coNP\_hard$.

Discussion on non-learnability

Non-learnability results appear to be strong for spiking neurons.
However non-learnability results exist also for traditional neurons.
Nevertheless, efficient learning algorithms have been found.
$\Rightarrow$ Still hope for discovering efficient learning rules for SNNs !

Present results mainly rely on learning by weight updating (usual) and delay updating (more recent / original). However, learning in biological systems may employ rather different mechanisms and algorithms than usual computational learning systems.
$\Rightarrow$ Still to be investigated, in collaboration with neuroscientists !

Plan

Learning in SNNs : an overview

Learning is probably the most challenging problem in SNNs !

- Traditional learning rules can be emulated on SNNs

Learning in SNNs : an overview

Learning is probably the most challenging problem in SNNs !

- Traditional learning rules can be emulated on SNNs
  - Hopfield networks, in temporal coding (Maass, Natschläger, 1997)
  - Clustering RBF networks (Natschläger, Ruf, 1998)
  - Kohonen's self-organizing maps (Ruf, Schmitt, 1998)
  - Multilayer RBF networks (Bohte, La Poutré, Kok, 2002)
  - Spike-prop in multilayer spiking neuron networks (Bohte, Kok, La Poutré, 2002)

Learning in SNNs : an overview

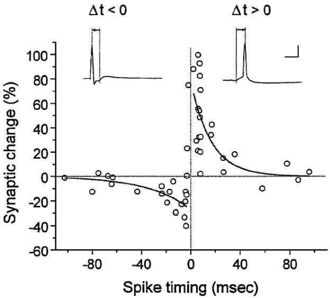
Learning is probably the most challenging problem in SNNs !

- Traditional learning rules can be emulated on SNNs
- STDP, a temporal Hebbian learning rule, is explored

Learning in SNNs : an overview

Learning is probably the most challenging problem in SNNs !

- Traditional learning rules can be emulated on SNNs
- STDP, a temporal Hebbian learning rule, is explored



STDP = Spike-Tme-Dependent Plasticity
Weiht change ΔW function of
spike timing $\Delta t = t_{post} - t_{pre}$
Circles are real data recorded by Bi and Poo (2001) [rat hippocampal neurons]

Learning in SNNs : an overview

Learning is probably the most challenging problem in SNNs !

- Traditional learning rules can be emulated on SNNs
- STDP, a temporal Hebbian learning rule, is explored
- Computational learning theory justifications for synaptic plasticity

Learning in SNNs : an overview

Learning is probably the most challenging problem in SNNs !

- Traditional learning rules can be emulated on SNNs
- STDP, a temporal Hebbian learning rule, is explored
- Computational learning theory justifications for synaptic plasticity
  - Hidden variables and maximum log-likelihood (Barber, 2003) (Pfister, Barber, Gerstner, 2003)
  - Maximization of mutual information (Chechik, 2003) (Toyoizumi, Pfister, Aihara, Gerstner, 2005a)
  - Retrieving the BCM model (Izhikevich, Desai, 2003) (Toyoizumi, Pfister, Aihara, Gerstner, 2005b)
  - Minimizing the entropy (Bohte, Mozer, 2006)

Learning in SNNs : an overview

Learning is probably the most challenging problem in SNNs !

- Traditional learning rules can be emulated on SNNs
- STDP, a temporal Hebbian learning rule, is explored
- Computational learning theory justifications for synaptic plasticity
- Several techniques can be combined

Learning in SNNs : an overview

Learning is probably the most challenging problem in SNNs !

- Traditional learning rules can be emulated on SNNs
- STDP, a temporal Hebbian learning rule, is explored
- Computational learning theory justifications for synaptic plasticity
- Several techniques can be combined
  - Evolutionary supervision of a dynamical neural network allows learning with on-going weights
    D. Meunier, H. Paugam-Moisy - IJCNN'2005
  - Delay learning and polychronization for reservoir computing
    H. Paugam-Moisy, R. Martinez, S. Bengio - 2008, Neurocomputing

Learning in SNNs : an overview

Learning is probably the most challenging problem in SNNs !

- Traditional learning rules can be emulated on SNNs
- STDP, a temporal Hebbian learning rule, is explored
- Computational learning theory justifications for synaptic plasticity
- Several techniques can be combined
- Learning in reservoir computing models

Learning in SNNs : an overview

Learning is probably the most challenging problem in SNNs !

- Traditional learning rules can be emulated on SNNs
- STDP, a temporal Hebbian learning rule, is explored
- Computational learning theory justifications for synaptic plasticity
- Several techniques can be combined
- Learning in reservoir computing models
  often reduced to linear regression for learning the weihgts of the readout neurons, or feedback connections towards the reservoir

Plan

Synaptic plasticity

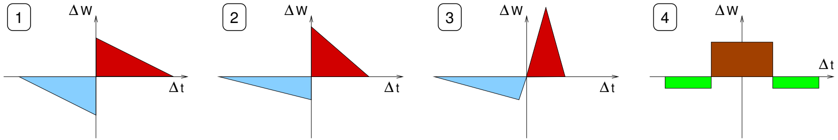
Principle : *Temporal* **Hebbian** rules

Hebb's law, 1949 : *When an axon of cell A is near enough to excite cell B or repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency , as one of the cells firing B, is increased.*

- Synaptic scaling - Synaptic redistribution
- LTP / LTD : Long Term Potentiation / Depression
- STDP : Spike-Timing Dependent Plasticity
- IP : Intrinsic Plasticity

STDP

Since biological experiments by Bi and Poo (2001), STDP is applied to SNN models, but with different window shapes.



Two ways to apply STDP windows : multiplicative learning rule avoid weight value saturation as often observed in additive learning rule.

if $\Delta t \leq 0$ depreciate the weight :

$$w_{ij} \leftarrow w_{ij} + \alpha \times (w_{ij} - w_{min}) \times \Delta W$$

if $\Delta t \geq 0$ potentiate the weight :

$$w_{ij} \leftarrow w_{ij} + \alpha \times (w_{max} - w_{ij}) \times \Delta W$$

Plasticity as learning rule

Dense review on consequences of STDP-like learning rules in SNNs :
Temporal sequence learning, prediction, and control : A review of different models and their relation to biological mechanisms - Wörgötter, Porr (2005) Neural Computation, 17(2)

Plasticity as learning rule

Dense review on consequences of STDP-like learning rules in SNNs :
Temporal sequence learning, prediction, and control : A review of different models and their relation to biological mechanisms - Wörgötter, Porr (2005) Neural Computation, 17(2)

Computational learning theory justifications have been found for applying synaptic plasticity as local, unsupervised learning rule :

1. **Log-Likelihood** maximization
   - Barber (2003) : learning temporal sequences + hidden variables
   - Pfister, Barber, Gerstner (2003) : $\mathcal{L}$ of post-synaptic spike train
2. Retrieving the **BCM model**, [Bienenstock, Cooper, Munro, 1982]
   - Izhikevich and Desai (2003) : BCM learning rule follows from STDP
   - Toyoizumi, Pfister, Aihara, Gerstner (2005a), Poisson spike train

Computational learning theory

...continue... Computational learning theory justifications have been found for applying STDP as local, unsupervised learning rule :

3. **Mutual Information** maximization, between input and output spike trains / between pre- and post-synaptic neurons
    - Chechik (2003), but for a rate-based neuron model + only LTP curve
    - Dayan and Häusser (2004) : STDP as optimal noise-removal filter
    - Bell and Parra (2005) : maximiz. “sensitivity” in a spiking neuron
    - Toyoizumi, Pfister, Aihara, Gerstner (2005b) : analytical approach

4. **Entropy** minimization, i.e. reducing the variability of a neural response to a given input spike train
    - Bohte, Mozer (2007) : reconstruction of the whole STPD curve

Multi-timescale learning and network complexity

We believe in the success of multi-timescale learning rules :

- STDP and reinforcement learning by evolutionary supervision [Meunier and Paugam-Moisy 2005 - Meunier, 2007]
- STDP and supervised delay learning with a margin criterion [Paugam-Moisy, Martinez and Bengio, 2007, 2008]

We study the influence of the network topology *(random graph, Small-World, scale-free)* on the network dynamics and performance.

"On the dynamical complexity of S-W SNNs", Shanahan, Physical Review E, Oct 30, 2008

$\Longleftarrow$? lecture for next meeting of SIG **Reservoir Computing** or **Complex Systems** (or both ?)

Conclusion

Spiking Neuron Networks are very powerful tools for information processing, as attested by many theoretical results :
coding capacity, complexity, learnability, etc. . .

From a practical point of view :
Searching for new learning paradigms able to take advantage of all the potential abilities of SNNs is a fascinating challenge.

From a theoretical point of view :
Computational complexity theory as well as learnability theory lack models for understanding and analyzing the computational power of SNNs, i.e. their ability for computing in continuously changing time.