From RULE-BASED NEURAL NETWORK
to PULSE-BASED NEURAL NETWORK

Luca Marchese - Syn@ptics – 25 January 2018

This article explains how an expert system based on rules can be mapped to a neural network model that learns through Hebb's rule. The problem of the "illegibility" of neural networks (often called black boxes) is well known, nevertheless the ability to learn from the data has made them in the 80s much more interesting than the so-called expert systems. The latter based on rules required the intervention of an expert for the "tuning" of the rules.

Current Machine Learning techniques do not offer performances and explainable models at the same time. For this reason DARPA has launched the Explainable Artificial Intelligence (XAI) program (DARPA-BAA-16-53) in August 10, 2016:

<<*Dramatic success in machine learning has led to an explosion of new AI capabilities. Continued advances promise to produce autonomous systems that perceive, learn, decide, and act on their own. These systems offer tremendous benefits, but their effectiveness will be limited by the machine's inability to explain its decisions and actions to human users. This issue is especially important for the Department of Defense (DoD), which is facing challenges that demand the development of more intelligent, autonomous, and symbiotic systems.* >> **(citation from DARPA-BAA-16-53).**

The XAI program has a wider target than the one described in this article, although the mapping of a rules-based system on a neural network that learns through the Hebb rule is a non-negligible starting point.

This is not an article on expert systems but contains only a brief introductory explanation. In this article we deal with classification problems and, consequently, the examples are related to this topic. An expert system operating on a classification problem should contain a number of such rules:

IF ((headache) AND (cold) AND (temperature> = 38)) THEN (His problem is FLU).

Obviously all the logical operators (AND, OR, XOR, NOT) can be inserted in the rules with complex groupings. In the previous rule we find two "label variables" ("headache" and "cold") that are not translatable into numerical values. In this article we focus on expert systems dealing with numerical input variables:

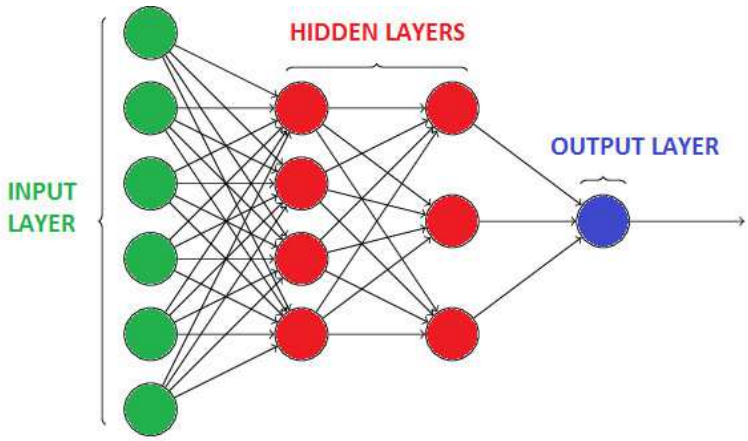
IF [(var_1 = 10) AND (var_2 = 15) ...AND (var_n = 6)] THEN class = 8

where var_n is the n-th input variable and "class" is the categorization of the input vector composed of n variables. In reality, a classification system that processes a vector of numerical variables (not labels) typically operates on variable ranges:

IF [(3 < var_1 < 10) AND (4 < var_2 < 15) ...AND (5 < var_n < 16)] THEN class = 8

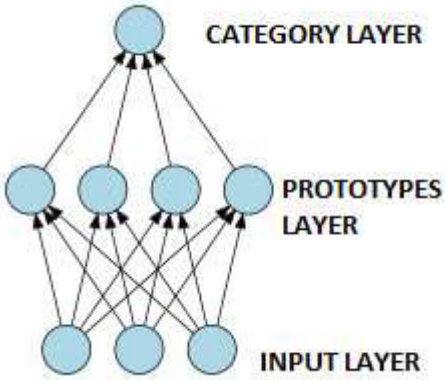
An expert system can contain thousands of these rules and the tuning phase also includes the selection of contrasting rules that different "human experts" have introduced. An expert system based on Fuzzy Logic (applied to a classification problem), through the mapping of the values of the variables on belonging classes (ie VERY HIGH, HIGH, MEDIUM, LOW, VERY LOW) and through the fuzziness of the operators (OR = MAX, AND = MIN), can obtain a multiple classification where each class possesses a confidence level: since this is a classification problem the defuzzification phase is not performed. Typically, systems based on Fuzzy Logic are not used on classification problems but as control systems with analog input values and analog output values (input fuzzification => inference(rules) => output defuzzification). A more complete explanation of Fuzzy-based systems is outside the scope of this article.

A **MLP (Multilayer Perceptron)** can classify input vectors after an appropriate learning phase. Compared to an expert system based on rules it has the ability to learn from the data and could provide levels of confidence in the classification. The weakness of a MLP-based classifier compared to an expert rule-based system is due to the fact that **MLP is actually a black box**. MLP is commonly trained with the complex and time-consuming EBP algorithm (Error Back Propagation). The process of "Reverse Engineering" of the synaptic values of MLP is certainly an even more complex task, although some researchers have ventured into this matter. In the case of a classification task, the supervised learning process imposes pairs [input_vector / class] but the ability of the network to generalize on input values is intrinsically determined by the synaptic values learned through the complex EBP algorithm and remains obscure and unpredictable. Therefore, we can not, of course, consider the input / output pairs as "rules" that we map on the neural network.



MULTILAYER PERCEPTRON (MLP)

Supervised neural networks based on the concept of prototype vectors (i.e SFAM[1], RBF[2]) have interesting properties of stability and plasticity, although they suffer from a lack of statistical consistency and requires few cycles on the training set to ensure stability on the previous learned patterns. From these neural networks, theoretically, simple rules can be extracted by scanning the prototypes and then reading the relative vectors and the values "Vigilance" (ART)[3] or "Neuron Influence Field (NIF)" (RBF with RCE)[4].



The picture shows a Radial Basis Function Neural Network

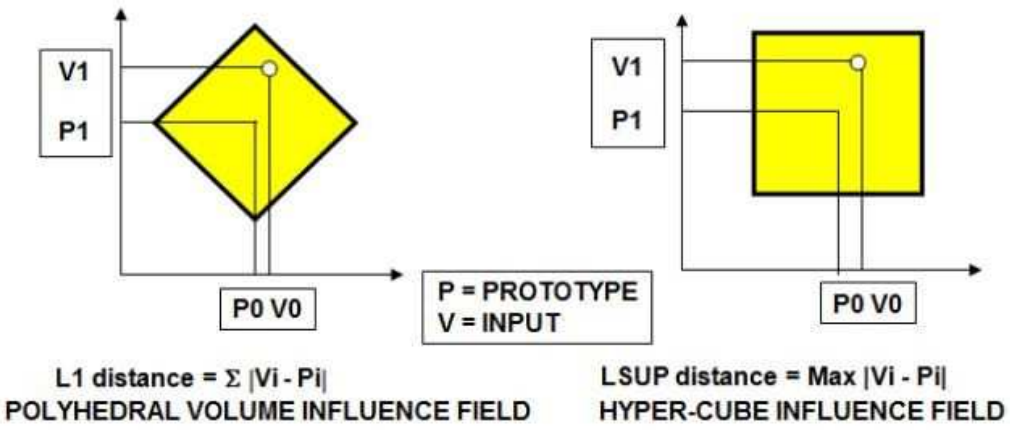
An RBF model generator and more specifically an RCE neural network that uses L1-norm could generate rules like:

IF (L1 distance < NIF) THEN { CLASS = "class connected with the prototype", Confidence of the classification = k / (L1 distance + 1) }

In the case of LSUP-norm:

IF (LSUP distance < NIF) THEN { CLASS = "class connected with the prototype", Confidence of the classification = k / (LSUP distance + 1) }

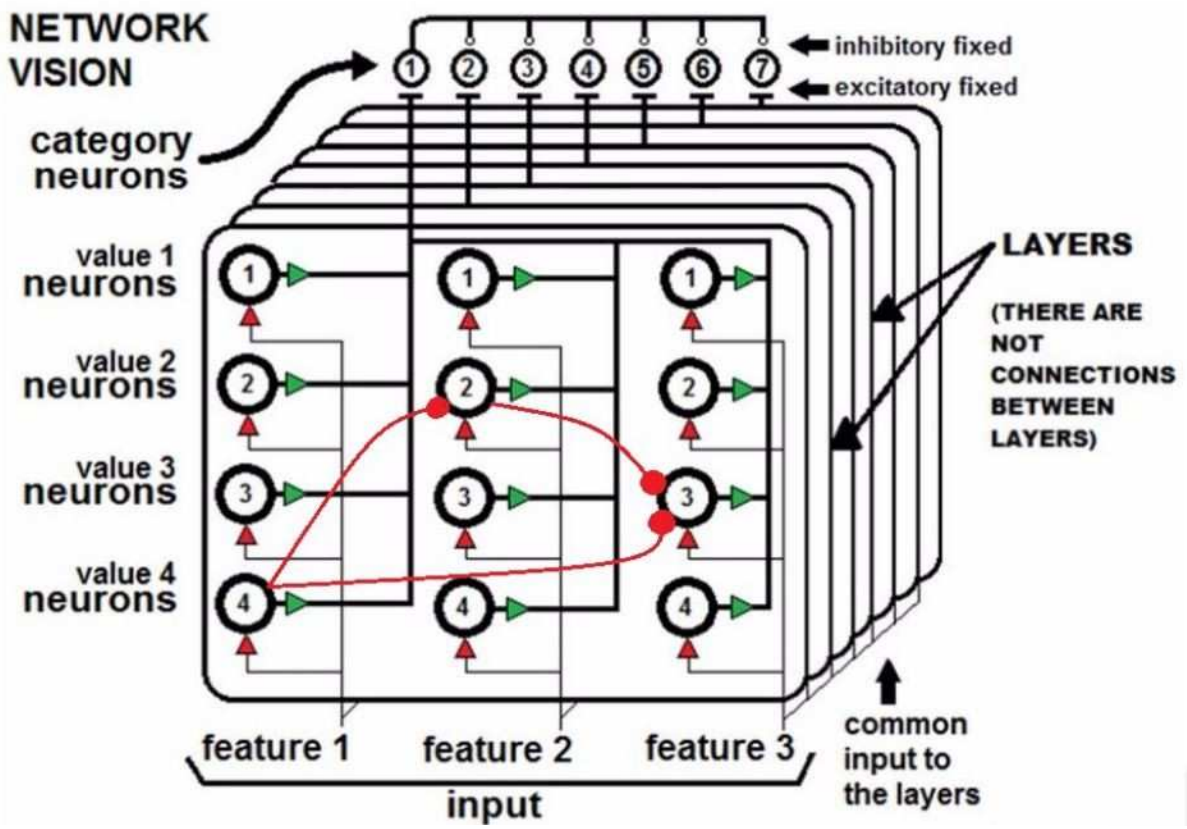
The vector distance using L1-norm and LSUP-norm is calculated as shown in the following figure:



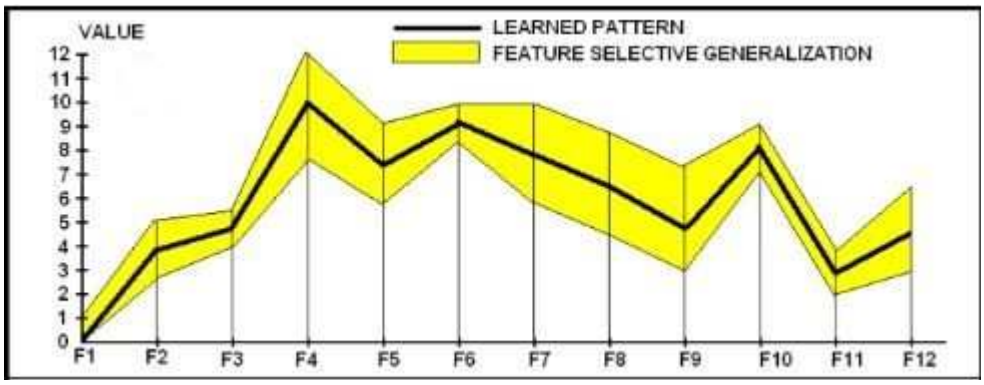
The picture shows the L1 and LSUP distance on a 2 components vector (P is the prototype vector and V is the input vector)

However, even if we have the ability to extract rules from the neural network, we can not map rules on the neural network in the learning phase. The input / output pairs of the training set examples can not contain information about the generalization capacity that the neural network will have after learning. Indeed we can set a minimum value and a maximum value of NIF but the learning process reduces the NIF values of the individual prototypes when a class-mismatch event occurs (a vector belonging to class B falls into the NIF of a prototype connected to the class A).

One of the targets that motivated the project of the SHARP neural network model (Systolic Hebb Agnostic Resonance Perceptron) was the possibility to map on the neural network many rules containing detailed information on the generalization for each feature of the input vector. LSUP-norm has been used with specific values for each component of the input vector.



A view of the SHARP neural network after learning the [4][2][3] pattern as belonging to class 1 (the red spots are excitatory synapses within the same layer). Generalization can be controlled for each feature and is obtained by connecting neighborhood neurons for each feature.

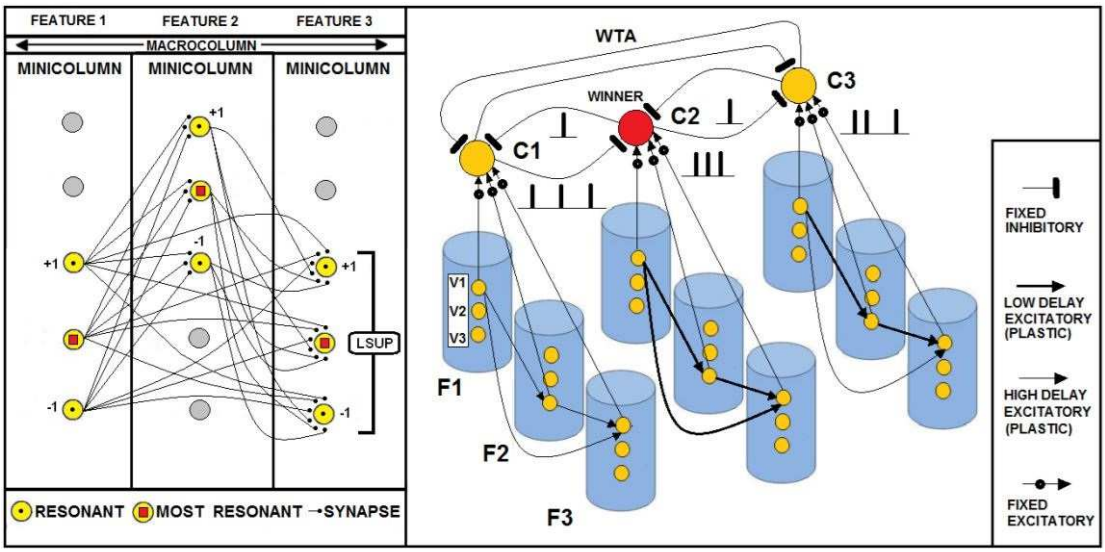


The picture shows the selective influence field of a learned pattern in SHARP. Since image processing is a very popular topic, I would like to point out that a matrix of pixels 8x8 inside an image is actually a vector of 64 components.

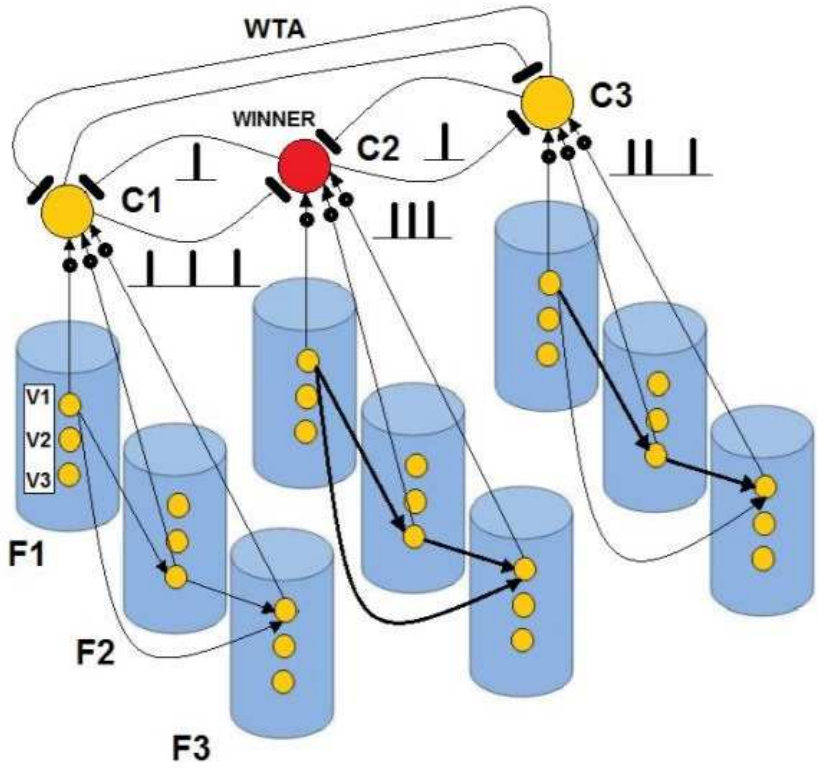
Indeed, in the SHARP neural network we can directly map a rule like:

IF [((P1-R1) < var_1 < (P1+R1)) AND ((P2-R2) < var_2 < (P2+R2)) ...AND ((Pn-Rn) < var_n < (Pn+Rn))] THEN CAT = 8; Confidence of the rule = k/((|P1 -var_1|)+(|P2 -var_2|)+...+(|Pn-var_n|))

where var_n is the n-th input variable, Pn is the n-th component of a learned example and Rn its range. CAT is the categorization of the input vector composed of n variables. What is the improvement over an advanced expert system that learns the rules from examples (input vector - category) and assigns a range to the components of the vector? Although more or less efficient implementations of the inferential engine can be realized, in the rule based expert systems, the rules database must be scanned in serial mode on a Von Neumann machine[5], in order to verify every single rule for each input**.** The SHARP neural network is executed (learning and recognition) on a Von Neumann machine in a single operation that concerns the synapses directly addressed by the input vector. In addition, the SHARP neural network was designed to be implemented in hardware as a third generation neural network**:** the timing of individual pulses (spikes) is crucial in the categorization process of the input vector. In its pulsed version, the synapses are modeled as delays and the category layer contains fixed inhibitory lateral synapses that implement a WTA (Winner Takes All) behavior (automatically solving the calculation of the best confidence rule). Although the project does not want to support any biological plausibility, the SHARP model has been presented in an architecture inspired by the cerebral cortex with mini and macro columns. The following picture shows the SHARP in this pulsed version.



LEFT: This picture illustrates a macro-column with three mini-columns in which the Hebb rule is applied to the neighboring neurons of the most resonant neuron. The applied LSUP value is 1 (i.e., +/-1) in all of the mini-columns. The number of involved synapses for any neuron of mini-column 2 is equal to 2 * LSUP [mini-col 1] + 1 (=3), while the number of involved synapses for any neuron of the mini-column 3 is equal to 2 * LSUP [mini-col 1] + 1 * LSUP [mini-col 2] + 1 (=6). RIGHT: Three macro-columns composed of three mini-columns. The WTA on the category layer is driven by the delays in the mini-columns.



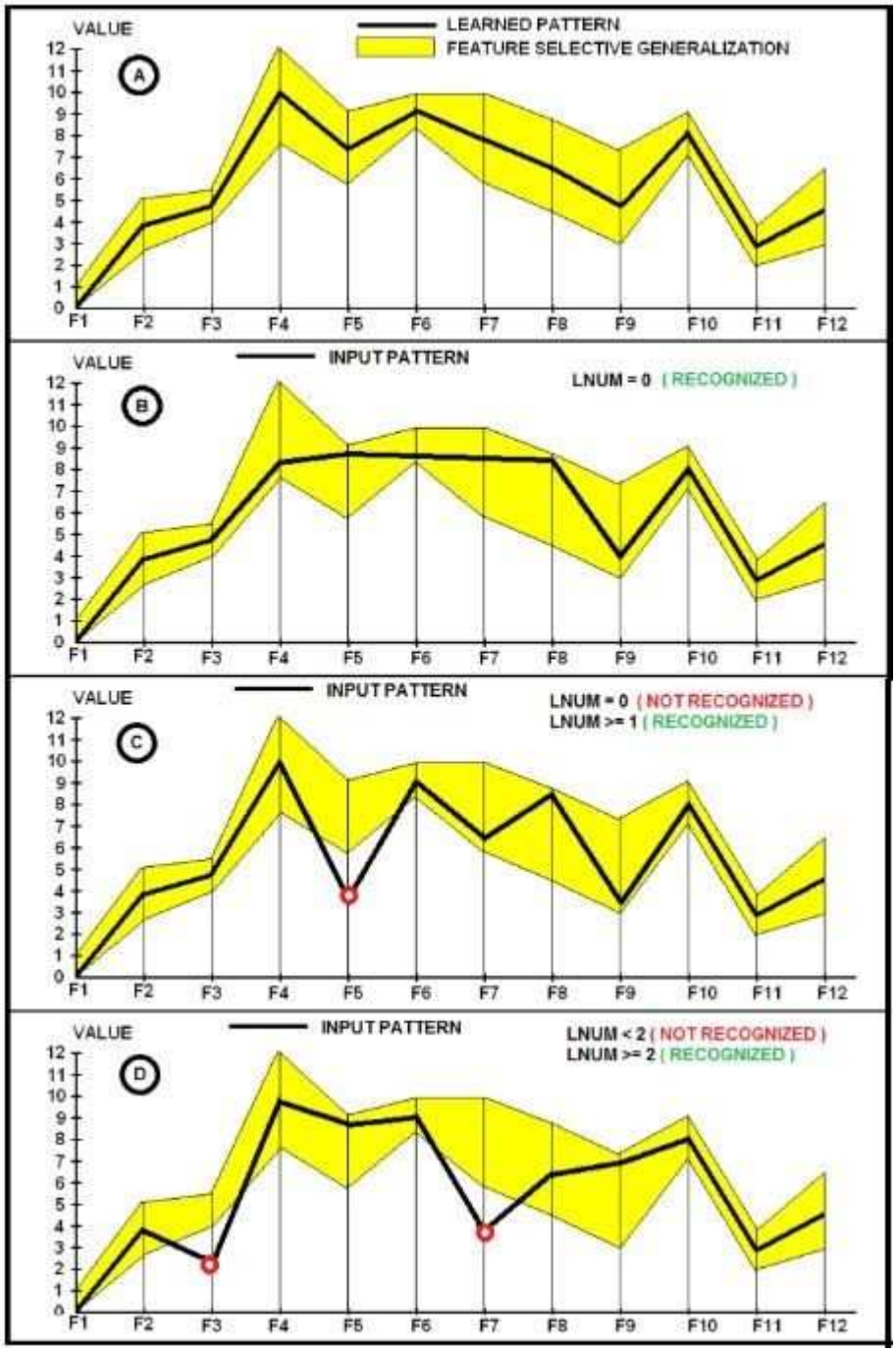
Systolic Hebb Agnostic Resonance Perceptron

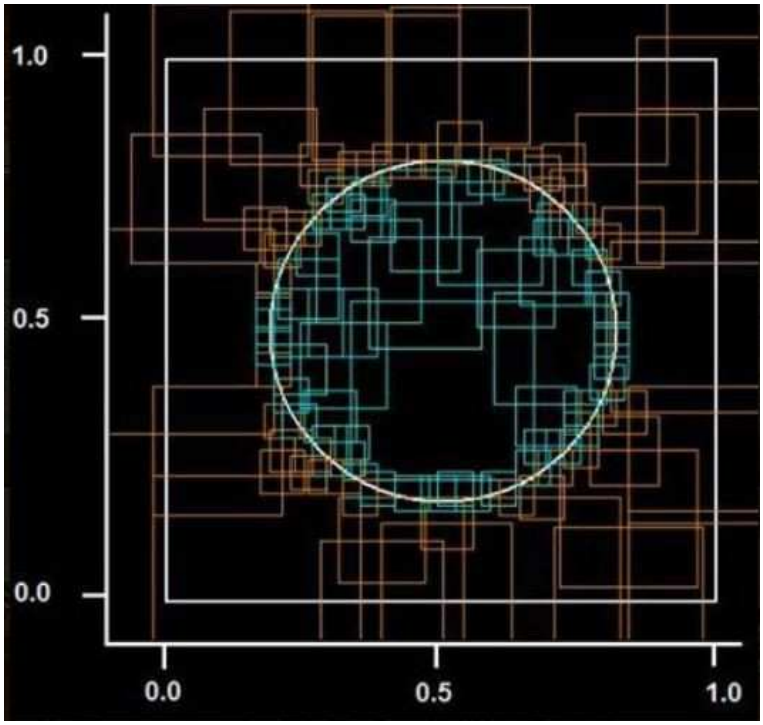
More detailed information related to learning using the Hebb rule and execution of the SHARP classifier can be found in the documentation contained in the links at the end of the article. The following figures show the difference in the mapping of the space on the "circle in the square" recognition problem, between an RCE (with L-SUP norm) neural network and a SHARP neural network. It is evident that the prototypes in the SHARP model are much more numerous and superimposed. This does not represent a performance problem as the prototype (unlike RBF) is addressed directly by the input. RCE Neural Networks can be executed with high speed on SIMD (Single Instruction Multiple Data) processors or on scalable low power Neuromorphic Chips implementing the RCE model like the NeuroMem™ that compare 1000 prototypes in parallel mode (this chip can work also in L1 norm and K-Nearest Neighbor). I must point out that L1 norm is certainly more flexible and suitable for pattern recognition applications than LSUP. LSUP has a narrower field of use and is often used for filtering applications. The SHARP algorithm has also the the LNUM generalization: a number of features may not fall into LSUP-based generalization.This behavior adds application flexibility to the SHARP neural network. Indeed, a rule like the following could be implemented:

IF [((P1-R1) < var_1 < (P1+R1)) AND ((P2-R2) < var_2 < (P2+R2)) ...AND ((Pn-Rn) < var_n < (Pn+Rn))] THEN CAT = 8; an LNUM number of the previous conditions can be FALSE; Confidence of the rule = k/((|P1 - var_1|)+(|P2 -var_2|)+...+(|Pn-var_n|))

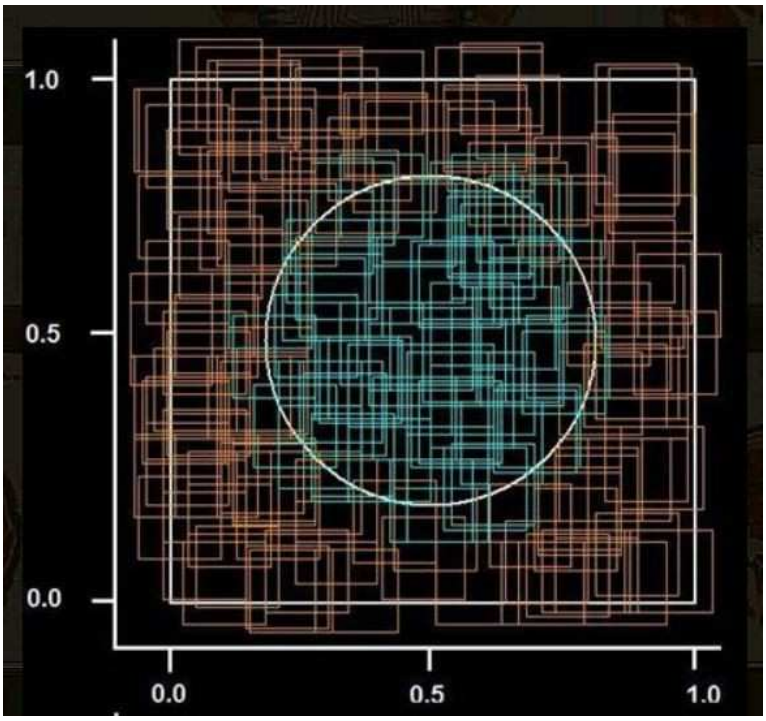
The following picture (next page) shows how SHARP generalizes with the LSUP norm that is different for any component of the input vector and with the LNUM norm that enables the category neurons to fire also receiving any number of pulses "n" such that "n > (number_of_features – LNUM)".

SHARP, however, is less flexible than the RCE algorithm with L1 Norm in real pattern recognition applications. Anyway SHARP can build, autonomously, rules with multiple ranges for the variables of the input pattern, by accumulating large sets of associations of input patterns with final categories. This is particularly interesting in "Lifelong Learning Machines" that update continuously their knowledge while they are operating "on the field". The SHARP learning algorithm warranties that a new pattern is learned without affecting the previous knowledge. Learning new data, the network creates new paths that could generate uncertainty decision spaces: these are resolved by the WTA (Winner Take All) behaviour in the Category layer.





Circle in the Square space mapping(RCE with LSUP norm)



Space mapping on SHARP with overlapped regions

[1] SFAM = Simplified Fuzzy ARTMAP is a Simplified Supervised Version of the Adaptive Resonance Theory

[2] RBF = Radial Basis Function

[3] ART = Adaptive Resonance Theory (Stephen Grossberg & Gail Carpenter)

[4] RCE = Restricted Coulomb Energy (Nobel prize Winner Leon Cooper)

[5] We can still consider Von Neumann machines also the current multi-core processors (2/4/8) with various types of limited parallelism.

DOCUMENTATION on the SHARP Neural Network:

Scientific documentation, software and simplified (no LNUM generalization and a limited categories number) source code of the SHARP neural network can be found at www.researchgate.net and at www.synaptics.org:

https://www.researchgate.net/publication/317003867_Systolic_Hebb_Agnostic_Resonance_Perceptron_SHARP_a_Neural_Network_Model_Inspired_By_the_Topological_Organization_of_the_Cerebral_Cortex_which_Implements_Virtual_Parallelism_on_Von_Neumann_Computers

https://www.researchgate.net/publication/318226283_SHARP_API_USER_MANUAL_VERY_BASIC_C_IMPLEMENTATION_OPEN_SOURCE

Other related documents (W.I.R.N 2014 Conference **/ Advances in Neural Networks -SPRINGER**:

https://link.springer.com/chapter/10.1007/978-3-319-33747-0_15

The following paper is related to a quite complex demonstration that the network could be realized in an analog framework. The work has to be considered an exercise because the neuron models are not biologically plausible and there is no reference to the realization of analog electronic circuits**.**

http://article.sapub.org/10.5923.j.ajis.20140405.01.html