Lorentzian Neural Nets

B.G.Giraud
Service Physique Théorique, DSM, C.E.Saclay, 91191 Gif/Yvette, France

Alan Lapedes and Lon Chang Liu
Theoretical Division, Los Alamos National Laboratory, 87545 Los Alamos, NM, USA

J.C.Lemm
Institut für Theoretische Physik I, Universität Münster, 48149 Münster, Germany

Abstract–We consider neural units whose response functions are Lorentzians rather than the usual sigmoids or steps. This consideration is justified by the fact that neurons can be paired and that a suitable difference of the sigmoids of the paired neurons can create a window response function. Lorentzians are special cases of such windows and we take advantage of their simplicity to generate *polynomial* equations for several problems such as: *i)*fixed points of a completely connected net, *ii)*classification of operational modes, *iii)*training of a feedforward net, *iv)*process signals represented by complex numbers.

Keywords–Neural Networks, Window-like response functions, Lorentzians and rational fractions, Integer coefficients, Training by solving polynomial systems, Processing of complex numbers, Fixed number of solutions, Classification of operational modes.

1. *Introduction*

The central issue of any theory of neural nets is to find the values of the synaptic weights which are best suited for a given task. This issue also implies the search of the corresponding best architecture of the net. (To see the close relation between the synaptic weights and the architecture of a net, it suffices to recall that, for example, the suitable cancellation of a sufficient number of synaptic weights in an initially completely connected net can easily convert the net into a pure feedforward, multilayer net.) Many algorithms (see, e.g., Minsky & Pappert, 1969, for the perceptron; Rumelhart, Hinton & William, 1986, for the backpropagation of errors; Mézard & Nadal, (1989) for the tiling algorithm) have been proposed for the training of a net, including simulated annealing (Kirpatrick, Gelatt & Vecchi, 1983) if local minima of the cost function to be optimized must be avoided. Despite the success of many such algorithms, there is still room for progress to be made in the understanding of the exact nature of operating modes of the nets, such as basins of attractions, (in)stabilities of fixed points, elimination of spurious attractors, fine tuning of synaptic weights in order to bring actual responses of the nets closer to their desired targets, etc. The purpose of this paper is to take advantage of the theory of analytical functions to attenuate the difficulties involved in studying these operating modes, as caused by the nonlinearity of the neurons. Thus, we will use neuron units whose responses are meromorphic functions, and actually, for the sake of maximum simplicity, we will only use "Lorentzian" functions.

Our aim is to reduce neural net problems to sets of polynomial equations, because of their simpler formal, and often practical, manipulation. Efficient methods (Manocha, 1994) are available to handle such polynomial systems. We will show that, when compared to the manipulation of the usual transcendental sigmoid functions, a detailed understanding of such "polynomial" nets can be reached more easily.

In Section 2, we recall why the use of Lorentzians, and window functions in general, is a legitimate reformulation of the performance of a neural net. In Section 3, we describe a model which consists of one Lorentzian layer between an input and an output layers. The analyticity properties of this model will be the subject of Sections 4-6. In Section 4 we study the properties of the fixed points of the model when feedback is allowed. A possible extension of the modes studied in Section 4 to oscillatory modes is discussed in Section 5. Training equations for a feedforward use of the model are

the subject of Section 6. We give an example of an associative memory in Section 7. Finally Section 8 contains our conclusions.

2. *Neural Nets with Windows*

As discussed by Lapedes & Farber (1988), neural nets often pair neurons and subtract their responses from each other according to the formula $\sigma(X-\alpha)-\sigma(X-\beta)$. Here σ represents a sigmoid, returning 0 if the input $X=\sum_i w_i x_i$ is much lower than the threshold α (resp.β), and returning 1 if X is much larger. Such a difference of sigmoids creates a window response centered at $(\alpha+\beta)/2$ in that direction specified by the vector of synaptic weights w_i which defines the linear combination X. (Notice that both neurons must receive the same input in order to define parallel hyperplanes. An equivalent result can be obtained from an inhibitory interneuron.) In an $N-$dimensional space of inputs $x_i, i=1...N$, the intersection of N such "hyperplane windows" can create a positive interference, resulting in a "bump", with magnitude N, localized in a well defined cell of the space. This allows tessellations and suits the needs of any given task.

Furthermore, as stressed by several authors (Funahashi, 1989; Hornik, Stinchcombe & White, 1989; Giraud, Liu, Bernard & Axelrad, 1991), window functions make a complete basis of a functional space at least as large as the Hilbert space of square-integrable functions. Let $\omega(X-\theta)$ be such a window function centered at θ. Any plane wave with any momentum k may be generated from the integral

$$\int d\theta \ \exp(ik\theta)\,\omega(X-\theta) \propto \exp(ikX). \tag{2.1}$$

In other words, a plane wave can be approximated and discretized, with any degree of accuracy, by a coherent sum of window outputs with different θ's. A complete basis is thus available, provided that a large number of neurons can be accommodated into the net.

Henceforth, the building blocks (basic units, BU) of the neural nets we will consider are formal units with the Lorentzian response function

$$\omega(X-\theta)=\frac{1}{1+(X-\theta)^2}. \tag{2.2}$$

The width of such a Lorentzian is obviously the reciprocal $(\sum_i |w_i|^2)^{-1/2}$ of the length of the vector of synaptic coefficients. We stress again that these

BU's can be always interpreted, if necessary, as neuron pairs in biological and/or technological applications.

Whereas the above arguments show that nets built from windows are quite general, it can be further shown that there is a formal relation between a net built from a set of traditional transcendental sigmoids and a Lorentzian net. Consider the sigmoidal response

$$\sigma(Y-\tau)=\frac{1}{2}[1+\tanh(Y-\tau)], \tag{2.3}$$

and the corresponding window

$$\Omega(Y-\tau)=\frac{\tanh(Y-\tau+\alpha)-\tanh(Y-\tau-\alpha)}{2\tanh\alpha}. \tag{2.4}$$

It is easy to show that

$$\Omega(Y-\tau)=\frac{\cosh^2\alpha}{\cosh^2\alpha+\sinh^2(Y-\tau)}\ . \tag{2.5}$$

A comparison of Eqs.(2.5) and (2.2) suggests an identification of $\sinh(Y-\tau)/\cosh\alpha$ with $X-\theta$. In the physical region of main interest where Y does not differ too much from τ, a linear approximation of the sinh is possible. Then, the "transcendental" neuron pairs, Eq.(2.4), can be mapped into the Lorentzian BU's, Eq.(2.2), by using simultaneously the relations $X=Y/\cosh\alpha$ for the inputs and $\theta=\tau/\cosh\alpha$ for the window centers. This mapping can be stated as the following "equivalence theorem": *For any "transcendental" neural net operating in the neighborhood of its (thresholds and) window centers, there exists an equivalent Lorentzian net.*

3. *Model, architectures and toy model*

We consider a three-layer net. The first layer consists of N input neurons for the reception of N data $x_1, x_2 ... x_N$. The second layer consists of N Lorentzian BU's defined by their "thresholds" θ_j and the synaptic weights w_{ji} relating to the input layer. The linear combination (modified by threshold subtraction) which enters the j'th BU is defined as

$$X_j=\sum_i w_{ji}x_i-\theta_j, \tag{3.1}$$

and the output of that unit is

$$y_j=\frac{1}{1+X_j^2}. \tag{3.2}$$

The third layer is a set of N "affine" neurons which simply mix the outputs of the second layer. As discussed previously (Giraud, Liu, Bernard & Axelrad, 1991), this last step takes full advantage of the basis of response functions provided by the second layer. The k-th neuron of this third layer generates as output the response

$$Y_k = \sum_j W_{kj} y_j - \Theta_k, \tag{3.3}$$

where the W_{kj}'s are the synaptic weights connecting the second to the third layer and the Θ_k's are the corresponding thresholds.

Let us denote by $\mathcal{A}$ the affine mapping of the x's into the X's, Eq.(3.1), then by $\mathcal{L}$ the Lorentzian mapping of the X's into the y's, Eq.(3.2), and finally by $\mathcal{B}$ the affine mapping of the y's into the Y's, Eq.(3.3). (We note incidentally that both $\mathcal{A}$ and $\mathcal{B}$ are invertible in general.) The feedforward mapping of the x's into the Y's is the product $\mathcal{BLA}$.

A special case of some interest consists in feeding this net back into itself. This induces the identification $x_k = Y_k$ and couples all BU's (second layer) to one another. Iterates of the form $(\mathcal{BLA})^n$ are thus created, where the product $\mathcal{AB}$ is still an affine transform $\mathcal{C}$. It is clear that $\mathcal{C}$ is a mapping of the y's into the x's,

$$x_i = \sum_j \mathcal{W}_{ij} y_j - \mathcal{T}_i. \tag{3.4}$$

The special case of symmetry in $\mathcal{C}$, namely $\mathcal{W}_{ij} = \mathcal{W}_{ji}$, is reminiscent of the standard restriction (Hopfield, 1982), but, under the interpretation of a BU as a pair of neurons, there is no symmetric action between an inhibitory neuron and an excitatory one.

In the following, it will be assumed that $\mathcal{A}$ and $\mathcal{B}$, hence $\mathcal{C}$, are invertible. Fixed points of $\mathcal{BLA}$ are related to those of $\mathcal{LC}$, a mapping of the $y's$ onto themselves. In turn, for cycles of length 2, fixed points of $(\mathcal{BLA})^2$ are more simply studied from those of $(\mathcal{LC})^2$.

For the sake of simplicity, we set $N = 2$ in the toy numerical examples discussed in Sections 4-6, and use simplified notations, saving indices: the two inputs are labelled x and y, the argument of the first BU is $ax + by - v$, and this BU returns $f = [1 + (ax + by - v)^2]^{-1}$; the second BU receives $cx + dy - w$ and returns $g = [1 + (cx + dy - w)^2]^{-1}$; finally the third layer generates outputs $pf + qg - t$ and $rf + sg - u$. The mapping $\mathcal{C}$ then consists in

feeding f and g back by means of the combinations $Af+Bg-V$, $Cf+Dg-W$, where

$$A = ap + br,\ B = aq + bs,\ V = at + bu + v,$$
$$C = cp + dr,\ D = cq + ds,\ W = ct + du + w. \tag{3.5}$$

4. *Fixed points, attractors*

Let $\mathcal{I}$ denote the identity operator in a two-dimensional space. Fixed points of the model, whether they are stable attractors or not, are solutions of the condition $\mathcal{BLA} = \mathcal{I}$ in the x, y space. Equivalent equations can be obtained from the condition $\mathcal{LA} = \mathcal{B}^{-1}$, or rather $\mathcal{LAB} = \mathcal{I}$, where $\mathcal{I}$ is now the identity operator in the (f, g) space. The parameters of $\mathcal{C} \equiv \mathcal{AB}$ are those given by Eqs.(3.5) and the fixed point equations read

$$f = \frac{1}{1 + (Af + Bg - V)^2}, \quad g = \frac{1}{1 + (Cf + Dg - W)^2}. \tag{4.1}$$

There are only 6 parameters $A, ...W$ in Eqs.(4.1), compared to the 12 parameters $a, ..., p, ...w$. This simplification may be stated as a trivial "economy theorem": *For a theory of fixed points, the affine mapping performed by either the first or the third layer is superfluous.* Consider now the polynomial forms of Eqs.(4.1),

$$(Af + Bg)^2 f - 2V(Af + Bg)f + (1 + V^2)f - 1 = 0, \tag{4.2a}$$

$$(Cf + Dg)^2 g - 2W(Cf + Dg)g + (1 + W^2)g - 1 = 0. \tag{4.2b}$$

An advantage of the present polynomial approach is read from the explicit degree 3 of each of the equations, Eqs.(4.2). For N BU's there will be always 3^N solutions, *real or complex*, stable or unstable. For two BU's there will be always 9 solutions. Indeed, upon eliminating either f or g between Eqs.(4.2), we obtain as a resultant a polynomial of degree 9.

With real coefficients $A, ..., W$ it is then easy to classify the operating modes of the toy net ($N = 2$) according to 5 modes : *i)* one real solution of Eqs.(4.2), accompanied by four pairs of complex conjugate solutions, *ii)* three real solutions and three complex conjugate pairs, etc. *v)* all nine solutions are real.

Because of the complicated way in which $A, ...W$ appear in Eqs.(4.2) it is not obvious whether all five cases occur and what is the stability situation.

We have found many numerical cases where only one real solution is obtained, and several cases where three real solutions are obtained. There is a trivial way of proving that it is possible to generate nine real solutions. For that proof, we first consider Fig.1, which shows the solution of the decoupled, one-BU problem $f = [1 + (5f - 4.85)^2]^{-1}$, *i.e.* $A = 5$, $B = 0$ and $V = 4.85$ in Eq.(4.2a). The presence of three real roots $f_1 = 0.045$, $f_2 = 0.905$, $f_3 = 0.990$ for this special case of the isolated one-BU net is clearly seen on Fig.1. It is also obvious that the equation $g = [1 + (5g - 4.85 - \varepsilon)^2]^{-1}$, where the second BU is in turn isolated and its "threshold" differs from the first "threshold", 4.85, by a small perturbation ε only, also generates three real roots g_1, g_2, g_3, very similar to f_1, f_2, f_3. All told, any sufficiently weak couplings B and C reinstated between Eqs.(4.2a) and (4.2b) will generate solutions very close to the nine real and distinct pairs (f_1, g_1), (f_1, g_2), ...(f_3, g_3).

It may be noticed from Fig.1 that the derivative of the Lorentzian is small at solutions f_1 and f_3, but larger than 1 for f_2. Accordingly, the pairs (f_1, g_1), (f_1, g_3), (f_3, g_1) and (f_3, g_3) are stable fixed points (attractors). Conversely, all pairs involving f_2 and/or g_2 are unstable. The classification in terms of the number of real solutions can thus be completed by a finer classification in terms of the number of stable fixed points.

The same tool of weak coupling may be used again to show a case where the number of real solutions reduces to one only. Indeed, if Eqs.(4.2) are again decoupled by means of very small values of B and C and if furthermore each of these decoupled equations has only one real solution (set for instance $V = W = 0$ with $A = D = 1$), then the system, Eqs.(4.2), has obviously only one real solution.

The opposite of weak is strong coupling : set for instance all parameters finite, except $A = \varepsilon$, where ε is an infinitesimal. This corresponds to the case where the first BU is driven by g much more than by f. It then turns out that at least one of the solutions of Eqs.(4.2) satisfies the ansatz ($f \simeq \ell\varepsilon^{-1}, g \propto \varepsilon^2$). For that solution f diverges and g vanishes.

Indeed, with such an ansatz, the leading term of Eq.(4.2b) gives $(C\ell\varepsilon^{-1})^2 g - 1 = 0$, consistent with the ansatz. Then the leading terms of Eq.(4.2a) read

$$(\ell^2 - 2V\ell + 1 + V^2)\ell\varepsilon^{-1} = 0, \tag{4.3}$$

hence $\ell = V \pm i$. This specifies the two complex conjugate values of ℓ compatible with the ansatz. Only seven finite solutions are left for Eqs.(4.2)

in this case of strong coupling for one BU. In the case of a double strong coupling, namely $A = D = 0$, it is now easy to understand that only five finite solutions are available.

This Section thus illustrates how the manipulation of polynomials makes it easy to understand the nature of the fixed points of the Lorentzian neural net. The number of fixed points is constant, and it is easy to keep track of all of them, even when some of them are in the complex plane. Furthermore the calculation of derivatives of Lorentzians is easy. This helps to sort out attractors from unstable fixed points. There is clearly no such simple approach in the case of "transcendental" neurons.

5. *Cycles*

For the sake of definiteness, we consider only the toy model and cycles of length 2 in this Section, since a generalization to more neurons and cycles of longer length, while a little cumbersome, is not difficult. It turns out that the iterate equations

$$x = f\left[f(x,y), g(x,y)\right], \quad y = g\left[f(x,y), g(x,y)\right], \tag{5.1}$$

reduce to a polynomial equation with degrees 9 and 8 for x and y, respectively, for the former equation, and 8 and 9 for the latter. The resultants of Eqs.(5.1), upon eliminating either x or y, factorize into two polynomials, namely *i)*a polynomial of degree 24, which yields in general true cycles of length 2, and *ii)*a polynomial of degree 9, which just yields cycles of length 1 (fixed points).

The polynomial of degree 9 is identical to that resultant, already discussed in the previous Section, Sec.4, which provides the fixed points of the dynamics of the toy model. Indeed, cycles of length 1 are also trivial solutions of Eqs.(5.1). More interesting is the polynomial of degree 24, for it allows a classification of the functioning of the net according to the (necessarily even) number of real roots. It will be noticed that the 24 roots must be suitably paired, because obviously both steps of each cycle represent simultaneous roots. There are thus 12 cycles of lentgh 2, real or complex, and a complex root shall not necessarily be paired with its complex conjugate in order to generate one such cycle. The classification involves 13 cases in principle: no real cycle, then one, then two, etc. until 12 real cycles. If the number of complex cycles is odd, one of the cycles is expected to be invariant under complex conjugation. An obvious subclassification follows, according to the stability or unstability of the cycles.

We show on Fig.2 one of the length-2 cycles obtained when $A = 4$, $B = -5$, $V = 1$, $C = 6$, $D = 7$, $W = -1$. This cycle is
$(-0.212 - 0.246i, -0.045 + 0.008i) \leftrightarrow (+0.146 - 0.188i, -0.210 - 0.520i)$.
This turns out to be a case where only one cycle is real and only one complex cycle is invariant under complex conjugation.

The consideration of the strong coupling limit, with $A = D = 0$, is interesting. It is obvious that one iteration of the network decouples f from g. For instance, a 2-cycle satisfies the condition

$$1 + \left[\frac{B}{1 + (Cf - W)^2} - V\right]^2 = \frac{1}{f} \tag{5.2}$$

It is trivial to obtain from this a polynomial equation for f, with degree 5, and a similar equation for g. Here f, g are the outputs of the BU's after either the first or the second time step. This condition, Eq.(5.2), can be easily tuned to generate one, three or five real roots. For instance, if $B = 6$, $C = 6$, $V = 4.5$, $W = 3$, we find five real roots for each polynomial, namely
$(f_1, f_2, f_3, f_4, f_5) = (+0.07, +0.34, +0.46, +0.56, +0.63)$
and
$(g_1, g_2, g_3, g_4, g_5) = (+0.13, +0.52, +0.93, +0.90, +0.62)$,
which we have ordered suitably, in such a way that $f(f_i, g_j) = f_j$ and $g(f_i, g_j) = g_i$, $i, j = 1, ...5$. Thus 10 real 2-cycles are found, of the form $(f_i, g_j) \leftrightarrow (f_j, g_i)$, $i < j$.

This Section thus illustrates how the implementation of oscillators is made easy by the manipulation of polynomials, as allowed by the choice of Lorentzian BU's for the net.

6. *Training*

In this Section we return to the feedforward structure. The number of parameters is $2N(N+1)$, namely N^2 for the w_{ji}'s, N for the θ_j's, and again N^2 for the W_{kj}'s and N for the Θ_k's. This allows an exact fulfillment of the same number $2N(N+1)$ of constraints. Assume for instance that such constraints consist in demanding for 2(N+1) input patterns $P^\mu = (x_i^\mu)$, $\mu = 1, ...2(N+1)$, specific outputs Y_k^μ, $\mu = 1, ...2(N+1)$, $k = 1, ...N$ from the third layer. The coupled equations

$$Y_k(P^\mu) = Y_k^\mu, \tag{6.1}$$

whose left-hand sides are just the representation of the composite mapping $\mathcal{BLA}$, are obviously reducible to polynomial equations. Let $\mathcal{N}$ be the (large)

degree of their resultant once all but one among the parameters $w_{ji}, ...\Theta_k$ have been eliminated between these polynomials. There will be always $\mathcal{N}$ solutions, whether real or belonging to complex conjugate pairs. Any of these $\mathcal{N}$ solutions cancels both cost functions

$$\mathcal{K} = \sum_{\mu k} \left[Y_k(P^\mu) - Y_k^\mu \right]^2 , \quad \mathcal{K}' = \sum_{\mu k} |Y_k(P^\mu) - Y_k^\mu|^2 . \tag{6.2}$$

The former cost function is meromorphic with respect to the parameters of the model, but is not positive definite if such parameters become complex. Conversely, the latter is always positive, but lacks analyticity properties.

For an easier understanding of the structure of Eqs.(6.1) we turn to the toy model, $N = 2$. Six input patterns may be used in order to generate training constraints. The resulting twelve constraints are linear with respect to p, q, t, r, s, u and nonlinear with respect to a, b, v, c, d, w. Let us denote by f_μ the response of the first BU to pattern P^μ and by g_μ the response of the second BU to P^μ. Then Eqs.(6.1) read

$$f_1 p + g_1 q - t + 0r + 0s + 0u = Y_1^1, \tag{6.3a}$$

etc.

$$f_6 p + g_6 q - t + 0r + 0s + 0u = Y_1^6, \tag{6.3f}$$

$$0p + 0q + 0t + f_1 r + g_1 s - u = Y_2^1, \tag{6.3g}$$

etc.

$$0p + 0q + 0t + f_6 r + g_6 s - u = Y_2^6. \tag{6.3\ell}$$

The explicit form of Eqs.(6.3) can be summarized in a matrix notation

$$\mathcal{M} L = \mathcal{Y}, \tag{6.4}$$

where L is the 6-dimensional vector (p, q, t, r, s, u), then $\mathcal{Y}$ is the 12-dimensional vector $(Y_1^1, Y_1^2, ...Y_1^6, Y_2^1, ...Y_2^6)$ and $\mathcal{M}$ is a rectangular matrix with 12 rows and 6 columns. The first row of $\mathcal{M}$ reads $(f_1, g_1, -1, 0, 0, 0)$. Its twelfth row reads $(0, 0, 0, f_6, g_6, -1)$. The block structure of $\mathcal{M}$, which disconnects (p, q, t) from (r, s, u) and $(Y_1^1, ...Y_1^6)$ from $(Y_2^1, ...Y_2^6)$, is technically useful but not essential. A standard manipulation of matrix algebra rearranges these 12 equations into two separate groups of 6 equations. The first group is

$$\mathcal{M}^T \mathcal{M} L = \mathcal{M}^T \mathcal{Y}, \tag{6.5}$$

which provides L as a function of (a, b, v, c, d, w) through the inversion of the matrix $\mathcal{M}^T\mathcal{M}$, where $\mathcal{M}^T$ is the transposed of $\mathcal{M}$. For the second group there are many processes, linear or not, which can extract from the twelve Eqs.(6.3) six null linear combinations. For instance we can first diagonalize the 12×12 matrix $\mathcal{M}\mathcal{M}^T$, whose rank is 6 in general, and select its 6 eigenvectors Z^λ corresponding to eigenvalue zero. Their components Z_i^λ, $i = 1, ...12$ are then used to set the (null space) second set of 6 equations,

$$0 = \sum_{i=1}^{12} Z_i^\lambda \mathcal{Y}_i, \ \lambda = 1, ...6. \tag{6.6}$$

It is to be noted that Eqs.(6.6) do not depend any more on L.

For the generalization to any value of N, it is clear that the block structure of $\mathcal{M}$ remains. Each block has $2(N+1)$ rows and $N+1$ columns. It provides $N+1$ equations of the type of Eqs.(6.6). Since each block, denoted m, nevertheless depends nonlinearly on all $N(N+1)$ parameters w_{ji} and θ_j, this makes a system of $N(N+1)$ coupled polynomial equations. Once they are solved, the solution in turn of Eqs.(6.5) proceeds by the separate inversions of N matrices $(m^T m)$, an easy operation.

For a numerical illustration of this argument, we will take a slightly simplified case, where parameters a, b, c, d have already been frozen arbitrarily, for instance $a = 4, b = -5, c = 6, d = 7$. This leaves only 8 parameters of the toy model to be constrained, by means of 4 patterns. In particular, only the two nonlinear parameters v and w are left open. The equations to be solved are only Eqs.(6.3a-d) and Eqs.(6.3g-j). The elimination of (p, q, t) between Eqs.(6.3a-d) and (r, s, u) between Eqs.(6.3g-j), respectively, gives the obvious determinantal conditions,

$$\det \begin{bmatrix} f_1 & g_1 & 1 & Y_1^1 \\ f_2 & g_2 & 1 & Y_1^2 \\ f_3 & g_3 & 1 & Y_1^3 \\ f_4 & g_4 & 1 & Y_1^4 \end{bmatrix} = 0, \quad \det \begin{bmatrix} f_1 & g_1 & 1 & Y_2^1 \\ f_2 & g_2 & 1 & Y_2^2 \\ f_3 & g_3 & 1 & Y_2^3 \\ f_4 & g_4 & 1 & Y_2^4 \end{bmatrix} = 0, \tag{6.7}$$

We will test this case for an XOR-like task. Namely the patterns are $(0,1)$, $(1,0)$, $(0,0)$, and $(1,1)$, respectively, and the requested answers are $Y_1^1 = 0.9$, $Y_1^2 = 0.8$, $Y_1^3 = 0.2$, $Y_1^4 = 0.1$, and $Y_2^1 = 0.1$, $Y_2^2 = 0.2$, $Y_2^3 = 0.9$, $Y_2^4 = 0.7$, respectively. The target values for Y_i^μ are purposedly chosen to differ slightly from perfect values such as 1 and 0. With such a choice, the

numerator of the rational function which represents the first determinantal condition, see Eqs.(6.7), is

$$(-1666040 + 2388280v + 784410v^2 + 3508440v^3 + 2229250v^4 + 756760v^5)w^0$$

$$+ \; etc. \; + \; (362 + 206v - 93v^2 - 312v^3 - 175v^4 - 58v^5)w^5, \qquad (6.8)$$

and a similar polynomial is found for the second determinantal condition. Upon eliminating w between these polynomials, one finds the resultant equation

$$-428 - 84v + 127v^2 + 468v^3 + 275v^4 + 92v^5 = 0, \qquad (6.9)$$

with only one real root, $v = 0.82$ and two complex conjugate pairs, $v = -1.39 \pm 1.39i$, $v = -0.52 \pm 1.10i$. Each root for v is independently compatible with each of the similar five roots of the associated resultant in terms of w,

$$20120 - 16197w + 2863w^2 - 50w^3 - 19w^4 + w^5 = 0, \qquad (6.10)$$

which turns out to show three real roots $w = -12.5$, $w = 1.77$, $w = 6.59$, and one complex conjugate pair $w = 11.57 \pm 2.14i$. Accordingly, one finds 25 combinations (v, w) which satisfy the training constraints. For that special numerical case under study there are thus only three among the 25 combinations which are made of real numbers.

It is then easy to recover the values of $L \equiv (p, q, t, r, s, u)$ from Eqs.(6.3). With the choice $(v, w) = (0.82, -12.5)$, one finds $L = (-2.71, 22.7, -0.38, 2.25, -129, -0.38)$. Then with the choice $(v, w) = (0.82, 1.77)$, one finds $L = (-3.20, 5.43, -0.80, 2.53, -3.10, -0.14)$. Finally, with $(v, w) = (0.82, 6.59)$, one finds $L = (0.28, 1.03, -0.011, 0.54, -0.59, -0.59)$. The large values of q and s requested by the first choice would demand strong amplifiers in the case of a hardware implementation of the toy model. This is likely to be clumsy. The second choice requests weaker amplifications and, all told, the third choice seems to be the least demanding.

We show on Fig.3 the contour lines of the output $f(x, y)$ of the first neuron, which is the same for all three choices (same value of v). The "ridge" of this output is seen to lie close to the diagonal $x = y$. This greatly helps the XOR separation of patterns $(1, 0), (0, 1)$ from patterns $(0, 0), (1, 1)$. The contour lines of the output $g(x, y)$ of the second neuron for the choice $w = -12.5$ describe a ridge far from the domain bounded by the patterns. This explains the large amplifying values of q and s that we already noticed and labelled as clumsy. Conversely, for the third choice, $w = 6.59$, the ridge

is close to the diagonal $x + y = 1$ and its position is well suited for the XOR separation task, hence only very moderate values of $(p, q, ..u)$ are needed. The choice $w = 1.77$ would be an intermediate case where the second neuron filters out pattern $(0, 0)$ well, but misses its correlation with pattern $(1, 1)$ and costs some (moderate) amplifications in the implementation of $(p, q, ..u)$.

Once the network has been trained, the polynomial equations can be used to investigate the reciprocal image of any domain of the output space Y. This solves the problem of inverting the mapping $\mathcal{BLA}$ and in particular defines useful "neighborhoods" in input space for pattern recognition problems. On Fig.4 we show the contour lines of the first output, $Y_1 = pf + qg - t$, under the second choice for v, w. We stress again that transcendental nets do not give such a convenient access to neighborhoods by functional inversions.

Because here $pq < 0$, the saddle point structure which chisels such neighborhoods is transparent. Hence neighborhoods defined by a condition such as $Y_k(P^\mu) > Y_k^\mu$ (or a converse condition $Y_k(P^\mu) < Y_k^\mu$) are open domains in the present numerical case. Naturally, the "mountain" for Y_1 in Fig.4 will become a "valley" for Y_2, and conversely. This is requested by our chosen, XOR-like, target output constraints Y_k^μ. Indeed, $rs < 0$.

We verified that suitably different tasks for the toy model can create constructive interferences $pq > 0$, and/or $rs > 0$, between the ridges describing $f(x, y)$ and $g(x, y)$, hence peaks, rather than saddle points, for the responses Y_1, Y_2 where these ridges intersect. This induces closed contours. Closed neighborhoods are thus available with Lorentzian nets.

More realistic training problems correspond to situations where there are more constraints than available parameters, and therefore the cost functions cannot vanish. Let us consider again the set of parameters $(a, b, c, d) = (4, -5, 6, 7)$ (frozen) and $(v, w) = (0.82, 6.59)$, $(p, q, t, r, s, u) = (0.28, 1.03, -0.011, 0.54, -0.59, -0.59)$ (adjustable), which we discussed for the task $(Y_1^1, Y_1^2, Y_1^3, Y_1^4, Y_2^1, Y_2^2, Y_2^3, Y_2^4) = (0.9, 0.8, 0.2, 0.1, 0.1, 0.2, 0.9, 0.7)$ with patterns $(0, 1), (1, 0), (0, 0)$, and $(1, 1)$, respectively. Under such a choice, the toy network returns outputs $(Y_1^5, Y_2^5) = (1.13, 0.20)$ for a fifth pattern $(0.5, 0.5)$. A task $(Y_1^5, Y_2^5) = (0, 1)$ for that fifth pattern is thus incompatible, as long as the parameters a, b, c, d remain frozen, and a modification of the other parameters $p, q, ...w$ becomes necessary to generate a compromise.

The readjustment of both parameter sets (v, w) and $(p, q...u)$ can be achieved by several, alternating readjustments of (v, w) and $(p, q...u)$, in

an algorithm somewhat similar to a back-prop procedure. We use a semi-random walk for (v, w), coupled with a systematic reappraisal of $(p, q...u)$ via Eq.(6.5) at each step. In this way the best set we find reads $(v, w) = (-0.45, 6.97)$, $(p, q, t, r, s, u) = (-0.88, 0.067, -0.84, 0.88, -0.011, -0.11)$, $(Y_1^1, Y_1^2, Y_1^3, Y_1^4, Y_1^5, Y_2^1, Y_2^2, Y_2^3, Y_2^4, Y_2^5) = (.87, .83, .11, .17, .02, .14, .15, .84, .79, .98)$ with $\mathcal{K} = 0.030$, a low minimum of the cost function.

This Section thus shows how training is facilitated by Lorentzians. Namely, training is identified with solving systems of polynomial equations. Small nets can thus be trained easily for specific subtasks. Alternately, if a traditional backprop training were used, numerical calculations of derivatives of $\mathcal{K}$ would still be at least as easy from Lorentzians as from sigmoids.

7. *Associative memory*

The manipulation of high degree polynomials may be avoided in the special case of the following example, an associative memory with N BU's and N patterns. In general, stability of stored pattern x_i^μ requires the fixed point conditions

$$x_j^\mu = \frac{1}{1 + (X_j^\mu - \theta_j)^2}, \quad X_j^\mu = \sum_i w_{ji} x_i^\mu. \tag{7.1}$$

The unknowns are just one layer of weights w_{ji} since the net is looped onto itself.

We consider binary patterns using $x_i^\mu = a_0$ or a_1 (with $0 < a_0 < a_1 < 1$) instead of $x_i^\mu = 0$ or 1, the generalization to nonbinary patterns being obvious, and for the sake of definiteness we choose $a_0 = 0.2$, $a_1 = 0.98$. Then every "weak" component ($x_i^\mu = 0.2$) induces a condition $X_i^\mu - \theta_i = \pm 2$, and any "strong" component ($x_i^\mu = 0.98$) induces a condition $X_i^\mu - \theta_i = \pm 1/7$. All told, we get a system of N^2 linear equations for the N^2 unknowns w_{ji},

$$\pm 2 \text{ or } \pm 1/7 = \sum_i w_{ji} x_i^\mu - \theta_j, \tag{7.2}$$

or, in an obvious matrix notation, $X = wx - \theta$, which trivially solves by matrix inversion $w = (X + \theta)x^{-1}$. This "learning rule" is clearly nonlocal. It will be noted that, because of the $\pm$ signs, there are 2^{N^2} possibilities. This opens a large domain for the choice of the best sign combination ensuring the best stability and/or chiseling of attraction basins. Before discussing more general cases, we report in the following our tests for *i)* a systematic choice $(+1/7, +2)$ and *ii)* a systematic choice $(+1/7, -2)$.

With some different numerical choices for the patterns, cases *ii)* generate mainly much weaker synaptic weights than cases *i)* and are thus preferable. Indeed, the patterns to be retrieved turn out to be unstable with cases *i)*, the slightest noise inducing a convergence towards spurious patterns. Conversely, cases *ii)* show very good retrieval properties: a noise of 10% in the pattern components is compatible with retrieval. We chose such 10% perturbations to be *complex* as well as real and thus verified that the basins of attraction of the patterns have a non negligible extension into complex numbers.

We noticed that such cases *ii)* actually correspond to weak coupling between the BU's. Here, nondiagonal weights are either vanishing, or small when compared to diagonal weights. Accordingly, neglecting nondiagonal weights, Eqs.(7.1) may be approximated by

$$x_j^\mu = \frac{1}{1 + (w_{jj} x_j^\mu - \theta_j)^2}. \tag{7.3}$$

It is clear that, within this approximation of decoupling between BU's, the first and third roots are stable fixed points and account for the components (a_0, a_1) which code the patterns, while the second root is a repulsive fixed point.

This raises the question whether all patterns with components chosen among the pair (a_0, a_1) are retrieved and stable in cases *ii)*. Such additional attractors would be spurious with regard to the patterns entered to generate the synaptic weights. Indeed, numerical simulations show that the stability of all, even not explicitly stored patterns of this form is very high or at least the final attractor is very similar to the starting patterns. But one wishes a larger selectivity and therefore less stability for patterns which are not stored directly. Numerical investigations show that there are many ways to reduce the stability of patterns not stored explicitly, and even destabilize them.

A first possiblity is to use a random distribution for the signs in the $(X_i^\mu - \theta_i)$–matrix. This creates normally a connection matrix w_{ji} with larger nondiagonal matrix elements. But this tends in many cases to destabilize some explicitly stored patterns, too. A good compromise is to choose only the sign for the smaller value at random, that is $\pm 1/7$ in our case. The larger value (± 2) should always keep the minus sign, or at most very few BU's should be forced to react with $+2$. This choice turns out to be sufficient for destabilizing those patterns which are not explicitly stored, without

destroying the stability of the stored ones. A second factor is the distribution of window centers θ_j for the different BU's. Taking such centers to be variable, e.g. choosing them at random, tends to destabilize the patterns, but the influence of this manipulation is not as strong as the effect of the sign distribution discussed first. Also the choice of representing "1" and "0" by 0.98 and 0.2 has some influence on the stability. Taking 0.99 instead of 0.98 increases the stability of the stored patterns, while using 0.95 decreases their stability. The effect of changing the value 0.2 is not as large, because it is located within the flat end of the Lorentzian. We find that stability is reduced if we use different values representing "1" or "0" in different patterns, or, more generally, if we allow arbitrary numbers to be stored.

We want to exemplify the effect of choosing a random distribution for the signs in X_j^μ for a case with 5 BU's and a set of also 5 patterns. As starting patterns we choose the set of stored patterns itself, the 5 corresponding "mirrored" patterns, both disturbed by a $\pm 5\%$ noise, and 5 random patterns. By mirrored patterns we mean the patterns itself, but with the values 0.98 and 0.2 exchanged. We investigate such mirrored patterns because they tend to be quite stable if their originals are stored, and we want to destabilize them. Fig.5 shows the starting patterns, Fig.6 the final ones after 60 iterations, Fig.7 the total change of the patterns during iteration, and Fig.8 the averaged change during the last ten iterations. It is seen that all stored patterns are stable and converge towards the exact values 0.2 and 0.98. The other patterns are all unstable. From Figs.6 and 7, pattern 8 seems to be an additional fixed point under the iteration. But pattern 8 is unstable, as can be seen in Fig.8, and in fact it converges into a two–cycle: $(0.778, 0.142, 0.980, 0.205, 0.138) \leftrightarrow (1.000, 0.136, 0.980, 0.215, 0.133)$. The only patterns which converge to a stable attractor are patterns 12 to 15, and they converge to a kind of "null" pattern. This allows to interpret this "null" attractor as the answer of the net to an unstored pattern. Thus, for the case under study, we obtain a network which retrieves the stored patterns, and, if started with unstored patterns, either does not converge, or converges to an attractor which sorts out unstored patterns. Fig.9 shows, for the numerical example, the positions on the Lorentzian of the input to the BU's after 60 iterations for the 15 patterns with 5 BU's each (relative to the window center of the individual BU's), and the 5 BU's with 15 patterns each. It is clear from Fig.9 that the values representing "0" are on the l.h.s. of the maximum for that specific case, while the values representing "1" are on both sides. One may note the similarity of the input values for the 5 different stored patterns

1 to 5.

The mapping $P^i \to P^i$ of patterns P^i onto themselves can be generalized into a form where several patterns are mapped onto one pattern, e.g. $(P^1, P^2, P^3) \to P^1$, $(P^4, P^5) \to P^4$, or, as another possibility, into a cyclic structure like $P^1 \to P^2, P^2 \to P^1$. The first case might be regarded as a classification of patterns into families, and our numerical studies show that this kind of mapping is possible and works well.

To summarize this Section, an associative memory was constructed with Lorentzians, with a familiar nonlocal rule (matrix inversion) for pattern storage. On one hand, it shows properties similar to those of a Hopfield model, namely stability of the stored patterns and additional attractors. This similarity with traditional networks is actually easy to understand. For instance the Lorentzian $1/(1 + (2.75x - 2.5)^2)$ and a sigmoid such as $1/(1 + \exp[-5.5(x - 0.5)])$ are similar in the region of physical interest, $0 \leq x \leq 1$. On the other hand, working with Lorentzians leaves the freedom to choose a negative slope which tends to create oscillations and to decrease stability of spurious and patterns. *Moreover it is easy to find basins of attraction extended into complex numbers.*

8. *Discussion and conclusion*

In this paper, we investigated as many properties as possible of neural nets made of "artificial neurons" (BU's) whose response functions are Lorentzians rather than traditional steps or sigmoids. It was found that, on one hand, such Lorentzian nets can achieve the same functions (classifiers, associative memories, oscillators, etc.) than traditional neural nets. This new brand of nets may bring novelty into the subject if it turns out that the technology of implementing such windows on silicon chips or any other suitable chemico–physical medium presents unexpected advantages.

On the other hand, we found several advantages of such new nets. For instance, as long as small assemblies of such BU's are considered, their programming (training) is achieved by the explicit sculpting of suitable polynomials, with the advantage of a very clear and familiar algebra, a fixed number of solutions when both real and complex roots are investigated and an easy classification of operating modes according to *i)*the number of real roots versus complex ones and *ii)*the strength of those derivatives (which are easy to calculate) related to stability criteria.

Moreover, any response functions in the form of rational functions, and

Lorentzians in particular, have a useful property if their parameters are also rational numbers or integers only: rational inputs generate rational outputs, and such is clearly not the case for transcendental response functions. Thus with Lorentzian nets, error free, *completely noise free* calculations are available in the realm of rational numbers. This is interesting in those contexts where chaotic behaviors of neural nets are met, or neural nets can be used to investigate chaotic motions. This is of some interest for the study of dynamical systems, multidimensional Julia sets, etc., by means of neural nets as highly parallel computers.

Our consideration of complex numbers extends the theory of neural nets to the processing of signals (alternating currents or optical and acoustical signals), represented by complex numbers coding for their phases as well as their square average intensities. The BU's should then be interpreted as elementary circuits processing such signals. Such a technological interpretation might already have some implementation in terms of optoelectronic devices with suitable nonlinearities.

We showed how the fixed points, cycles, etc. and, hence, the operational modes of a Lorentzian net can be classified. A complete description of the properties of a neural net requires an extension of the solution space from real axes to complex planes. For those traditional nets whose transcendental responses allow analytic continuation, excursions into complex manifolds would likely be costlier, in terms of hardware, software and computational time, than excursions for Lorentzian nets.

Because of the "equivalence theorem" in Section 2, our findings apply to many neural nets. We showed that a Lorentzian net presents an advantage over a transcendental net, in that the stability of the Lorentzian net can be more rigorously studied, understood, and controled. Then, given a transcendental net, it makes sense to approximate it by a "rational" (Lorentzian) net if it is made of windows, the necessary condition for such an approximation. The properties of the transcendental net can then be estimated from those of its rational partner, at a cheaper computing cost: rational functions are in general cheaper to calculate than transcendental ones.

It is also worth noting that current methods for training neural nets are based upon the concept of seeking optimal solutions having real values only. These real solutions can be regarded either as special cases of complex solutions if a suitable cost function is completely cancelled, or alternately as the closest projections, in some sense, of complex roots of the cost function

onto the real axis if that cost function is minimized rather than cancelled. Training might be easier if excursions in a space of complex synaptic weights were allowed.

In conclusion, Lorentzian nets seem to be a tool of significant interest because their theory is reducible to the manipulation of polynomials and, simultaneously, they have the same broad range of practical applications as traditional nets. Furthermore, they give a strong hint that the operation of neural nets should seriously be continued into a domain of data and parameters represented by complex numbers.

References

Funahashi, K.I. (1989) "On the approximate realization of continuous mappings in Neural Networks", *Neural Networks* **2** 183

Giraud, B.G., Liu, L.C., Bernard, C. & Axelrad, H. (1991) "Optimal Approximation of Square Integrable Functions by a Flexible, One Hidden Layer, Neural Network of Excitatory-Inhibitory Neuron Pairs", *Neural Networks* **4** 803

Hopfield, J.J. (1982) "Neural networks and physical systems with emergent collective computational capabilities", *Proceedings of the National Academy of Sciences"* **79** 2554

Hornik, K., Stinchcombe, M. & White, H. (1989) "Multilayer feedforward networks are universal approximators", *Neural Networks* **2** 359

Kirkpatrick, S., Gelatt, C.D.Jr & Vecchi, M.P. (1983) "Optimization by Simulated Annealing", *Science* **220** 671

Lapedes, A. & Farber, R. (1988) "How Neural Nets Work", in D.Z.Anderson (ed.) *Neural Information Processing Systems* New York: AIP Press

Manocha, D. (1994) "Solving Systems of Polynomial Equations", *Computer Graphics and Applications* **14** 46

Mézard, M. & Nadal, J.P. (1989) "Learning in Feedforward Networks: the tiling algorithm", *J.Phys.A: Math. Gen.* **22** 2191

Minsky, M.L. & Papert, S.A. (1969) "Perceptrons", Cambridge: MIT Press

Rumelhart, D.E., Hinton, G.E. & Williams, R.J. (1986) "Learning Internal Representations by Error Propagation" in D.E.Rumelhart and J.L.McLelland (eds.) *Parallel Distributed Processing: Exploration in the Microstructure of Cognition* (Vol.1), MIT Press

Figure Captions

Fig.1: Case with 3 fixed points.

Fig.2: Example of complex 2–cycle for toy model. (Missing. Requests for Fig.2 should be sent to B.G.Giraud, French Atomic Energy Commission, Service de Physique Théorique, DSM-CE Saclay, F-91191 Gif/Yvette, France)

Fig.3: Contour plot of first neuron responses for XOR task of toy model.

Fig.4: Structure of neighborhoods in input space for first output (XOR task).

Fig.5: Associative memory model : its 5 stored patterns, the corresponding 5 reversed patterns, and 5 random patterns used as starting points for attempted retrieval.

Fig.6: Associative memory : patterns obtained after 60 steps when starting from the patterns of Fig.5. Patterns 1-5 retrieved, 8 almost retrieved, spurious attractor for 12-15.

Fig.7: Retrieval check from differences between Figs.6 and 7.

Fig.8: Average evolution of patterns between steps 51 and 60. Compare with Fig.7. Patterns 1-5 stable, patterns 6-10 not converged and patterns 11-15 not retrieved although converged to other attractor.

Fig.9: Status after 60 steps. Each drawing in the first 3 rows shows for a given pattern ($\mu = 1, \ldots, 15$) the thresholded fields $X_j^\mu - \theta_j$ ($j = 1, \ldots 5$) at which the Lorentzian function reacts. Each picture in the last row shows for a given neuron ($j = 1, \ldots 5$) the fields X_j^μ which its receives from the 15 patterns. Notice the strong clustering of lines and their concentration away from the right shoulder.

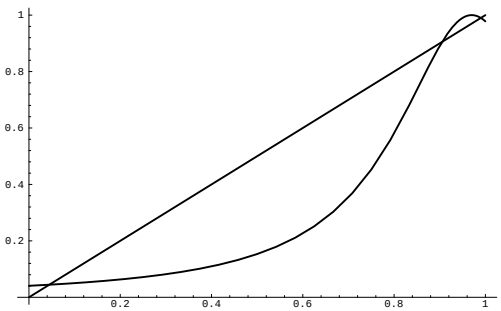

Figure 1

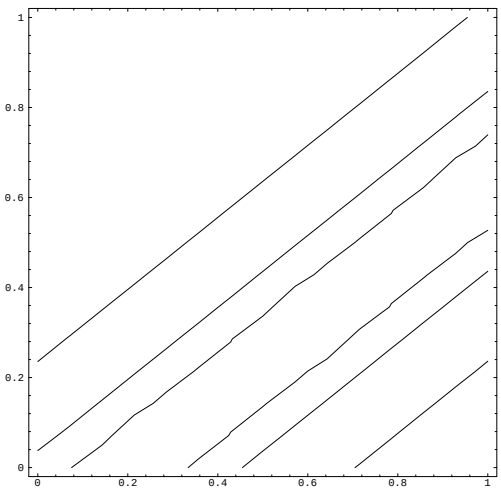

Figure 3

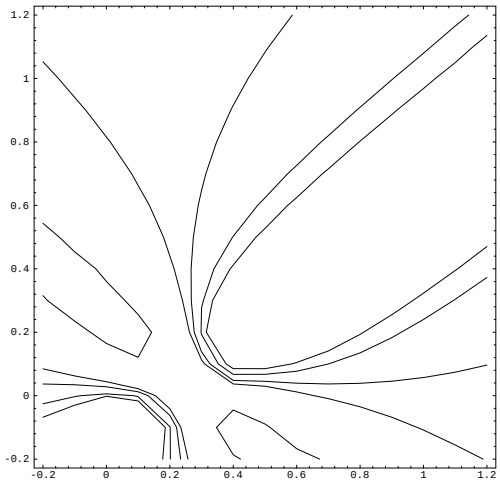

Figure 4

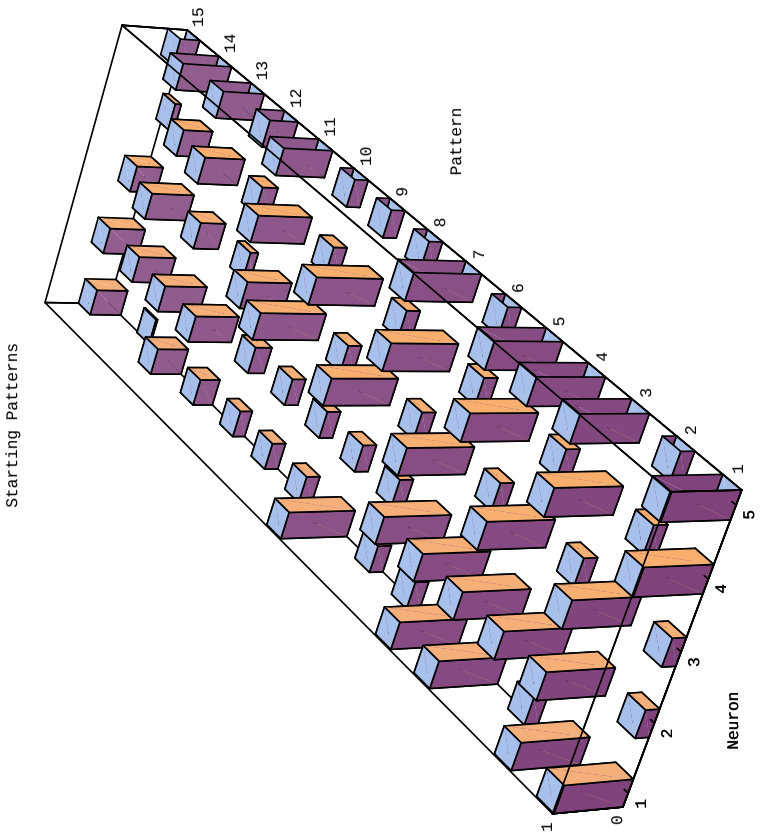
Starting Patterns
Pattern
Neuron
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
1
0
1
2
3
4
5

Figure 5

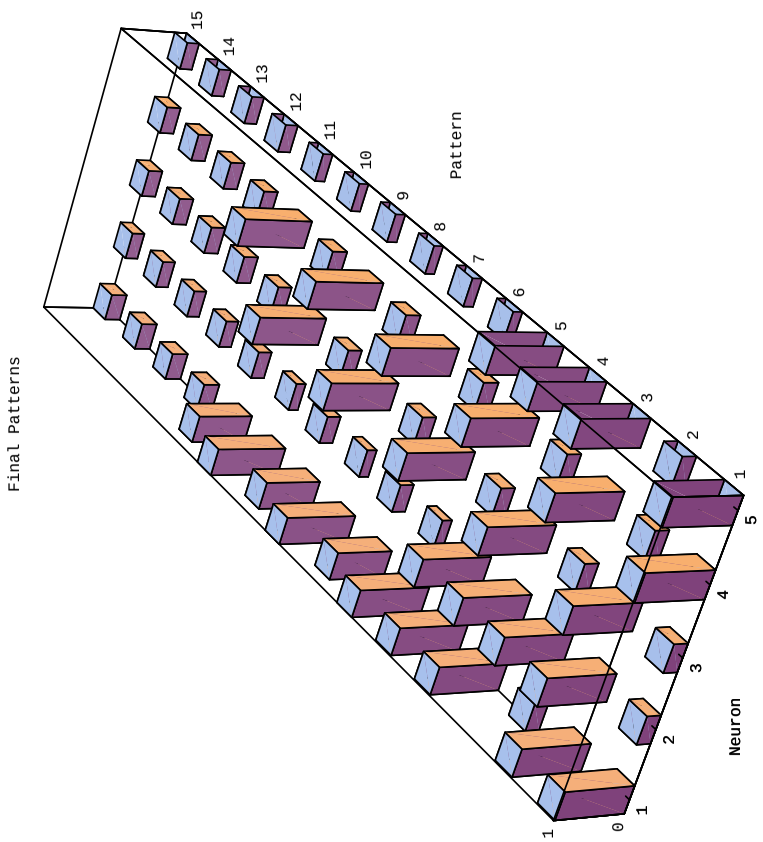
Final Patterns
Pattern
Neuron
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
1
0
1
2
3
4
5

Figure 6

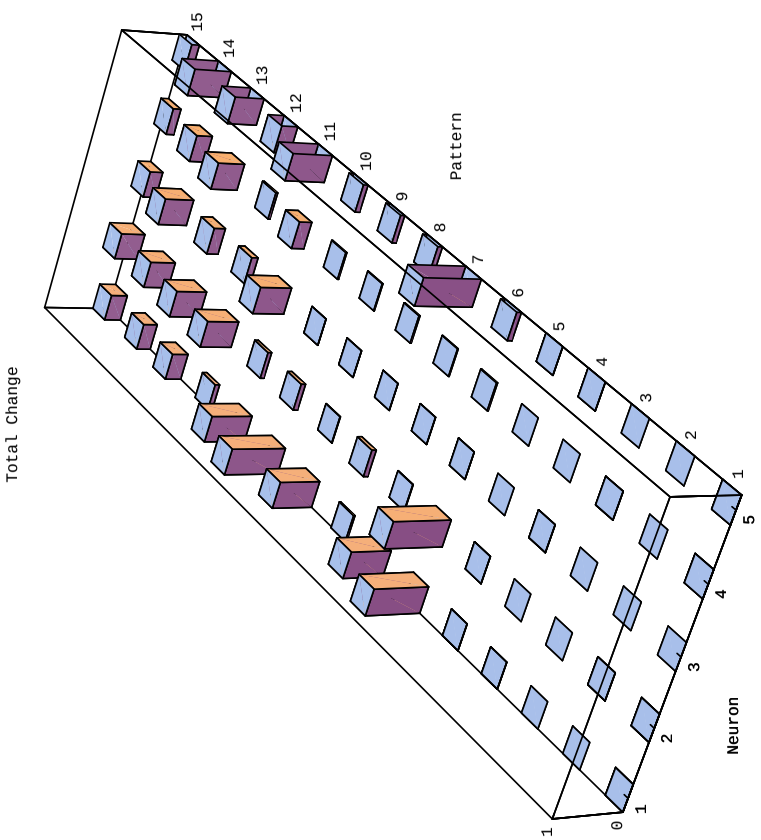
Total Change
Pattern
Neuron
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
1
0
1
2
3
4
5

Figure 7

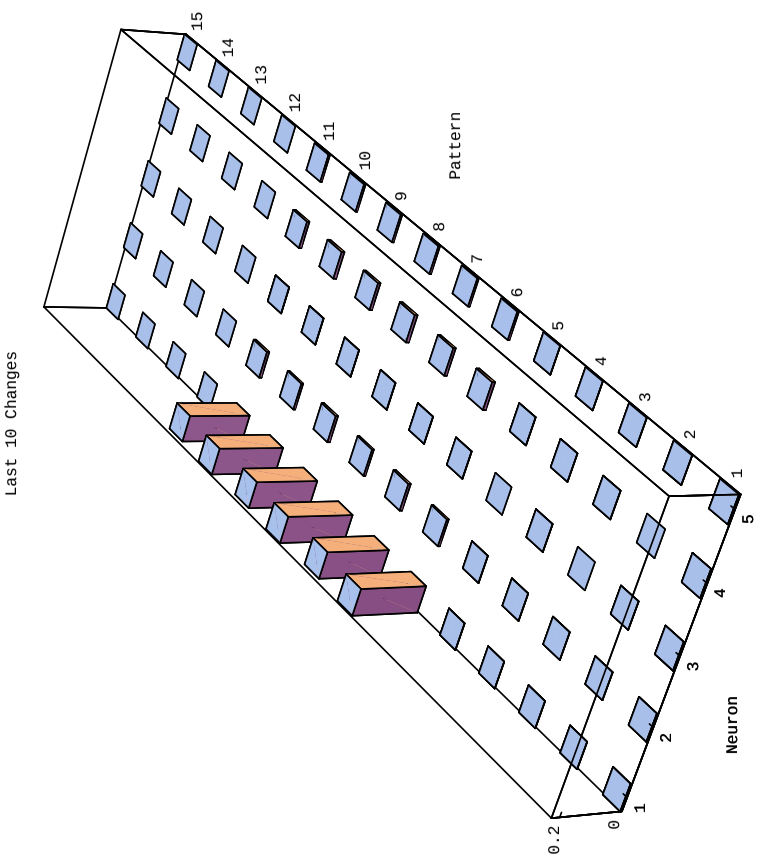
Last 10 Changes
0.2
0
1
2
3
4
5
Neuron
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Pattern

Figure 8

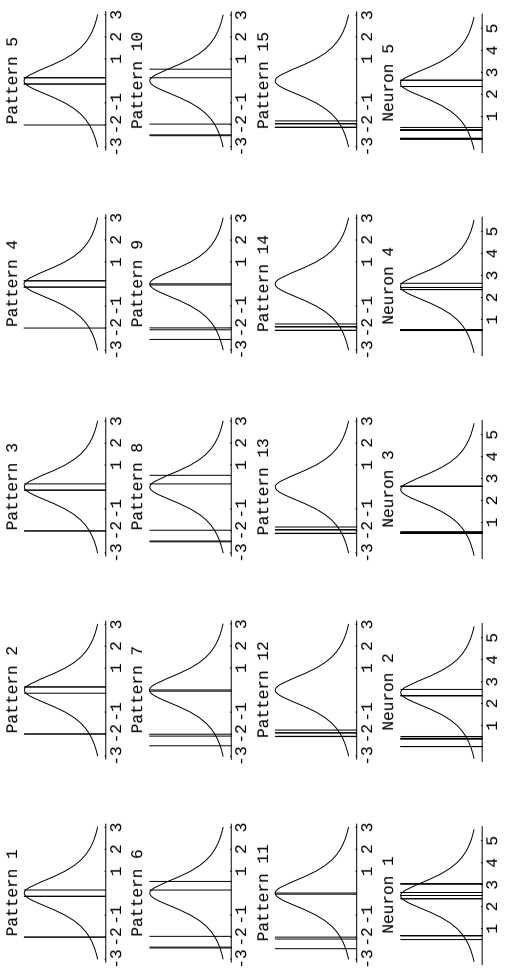
Pattern 1
Pattern 2
Pattern 3
Pattern 4
Pattern 5
Pattern 6
Pattern 7
Pattern 8
Pattern 9
Pattern 10
Pattern 11
Pattern 12
Pattern 13
Pattern 14
Pattern 15
Neuron 1
Neuron 2
Neuron 3
Neuron 4
Neuron 5
-3 -2 -1 1 2 3
1 2 3 4 5

Figure 9