TempUnit: A Bio-Inspired Spiking Neural Network

Olivier F. L. Manette
Unité de Neurosciences Intégrative et Computationnelle (UNIC), CNRS
France

1. Introduction

Formal neural networks have many applications. Applications of control of tasks (motor control) as well as speech generation have a certain number of common constraints. We are going to see seven main constraints that a system based on a neural network should follow in order to be able to produce that kind of control. Afterwards we will present the TempUnit model which is able to give some answers for all these seven criteria.

1.1 Learning

Neural networks show usually a certain level of learning abilities, (Hornik, 1991). In the case of systems of motor control or of speech generation those learning skills are particularly important. Indeed they enable the system to establish the link between the motor command and the act as it has been achieved. In real systems (not simulated), biological or machines, the effector could evolve for all sort of reasons such as limb growth or injury for biological systems; For artificial systems, the reason could be some breakage or a subsystem malfunction. It has also been demonstrated that the motor command is directly related to the characteristics of the effector (Bernstein, 1967; Hogan & Flash, 1987; Gribble & Ostry, 1996). Thus learning capabilities should be permanently maintained: it is necessary that the neural network is able to evolve its transfer function. In this case, because the aim is to learn the link between a desired output (the effector activity) and a given input (the motor command), it is called supervised learning.

1.2 Inverse model

Several models of motor control exist but we can globally group them into two categories: reactive (Sherrington, 1906/1947) and predictive (Beevor, 1904). Reactive models usually work as a closed loop, in which a very imprecise motor command is sent and is then updated using sensory feedbacks. For most movements, for instance, the basket ball throw (Seashore, 1938; Keele, 1968; Bossom, 1974; Taub, 1976), sensory feedback is simply too slow to enable efficient motor control. In this chapter, we will focus particularly on predictive models. Predictive models imply the calculation of a forward function which gives, from a given motor command, a prediction of the effector activity (Desmurget & Grafton, 2000). This type of function can be very useful because it enables the result of a given command to be simulated without the need to effectively carry it out. However, the inverse function is

Source: Theory and Novel Applications of Machine Learning, Book edited by: Meng Joo Er and Yi Zhou,
ISBN 978-3-902613-55-4, pp. 376, February 2009, I-Tech, Vienna, Austria

much more useful because it enables determination of the motor command from a desired motor output. Theory of motor control predicted the need for an inverse model (Berthoz, 1996) in the brain to enable calculation of the motor command (Shadmehr & Mussa-ivaldi, 1994; Wolpert et al., 1995; Kawato, 1999). The use of an inverse function enables the system to calculate by itself the motor command that leads to a specific desired result, which is a great advantage.

1.3 Syntax

In linguistics, syntax is the group of rules that defines the chain of words to form sentences. In control of speech production, it is easy to understand why the system needs to be able to respect those syntax rules. But it should also be the case in control of movements and it is even more important than in speech generation. Indeed, while a limb is in a particular position, it is simply impossible to immediately reach any other arbitrary position. From a given position of a limb, only a few other positions are accessible, otherwise some bad command could possibly damage the system if those rules are not respected. The neural network in charge of the overall control of the system should therefore necessarily respect scrupulously those chaining rules from one state to another state of the system.

1.4 Decision node

The lack of flexibility is generally the main problem in a feed-forward system: a system should be able to interrupt the execution of a motor program to change direction in order to, for example, ward off an unexpected disruption. Hence, a process able to manage decision nodes at every single time step, with the aim of enabling the system to evolve in different directions, should be integrated.

1.5 Respect of the range limits

Keeping the system in the range of possible values is another important constraint for a task control system. Also, some mechanism able to handle aberrant values has to be integrated in order to avoid damage to the system.

1.6 Complexity

The choice of the neural network architecture is also an important issue since it is absolutely necessary to use an architecture adapted to the problem to be solved. An under-sized network will not be able to obtain the expected results with the desired level of performance. Conversely, an over-sized network can show problems such as over-learning, i.e. an inability to correctly generalize the data, possibly even to the point of learning the noise. It is therefore important to obtain a system adapted to the desired task and a clear way to find the best fit architecture.

1.7 Parsimony

In a predictive model, the system includes a feed-forward module that gives the predicted motor activity from a motor command. It is then equivalent to encoding the motor activity in the command domain. A well designed system should limit the command size at its maximum. It does mean maximize the information compression rate.

1.8 Suggested solution

The choice of a transfer function is generally a problem for formal neural networks, because they should be determined "by hand" after several trials. After having been chosen, the transfer function does not evolve anymore and limits the future neural network abilities in a decisive behavior. Furthermore, there exist only empirical solutions to determination of the number of neurons or the size of the hidden layer (Wierenga & Kluytmans, 1994; Venugopal & Baets, 1994; Shepard, 1990).

The behavior of the TempUnit model presented below enables us to give some solutions to these problems found in common formal neural network models. TempUnit does not have a fixed transfer function but is able to learn the best adapted one to fit the desired signal. The basic principle is quite simple because it is only based on the principle of temporal summation such as has been observed in biological neurons. In order to keep thinking at a biologically inspired level, the input of each TempUnit neuron is a spikes train: only binary values. We will now develop in the following the reasons why TempUnit is a particularly well-adapted model for satisfying all the constraints previously discussed.

2. The tempUnit model

2.1 Temporal summation and straight forward function

The TempUnit model is based on the mechanism of the temporal summation of post-synaptic potentials as observed in biological neurons (e.g. Rieke et al, 1996). TempUnit means simply 'Temporal summation unit'. When a spike arrives at a synaptic bouton it triggers a local potential in the soma of the post-synaptic neuron. If many spikes arrive at a fast enough rate, the triggered potentials are summed in the post-synaptic neuron to shape a global membrane activity. This new global temporal activity of the post-synaptic neuron depends only on the structure of the input spikes train from the pre-synaptic neurons. In fact, this principle, as noticed in biological neurons, can be generalized to any serial system where the output is only correlated to the input. A TempUnit network can be considered in this form as a binary-analog converter whose output is totally deterministic, as we shall see later. Furthermore, biological neurons have, in common with TempUnit, a deterministic behavior, as shown by empirical (Mainen & Sejnowski, 1995).

In the simplest case, a TempUnit network is composed of only a single pre-synaptic neuron which is the input and a post-synaptic neuron which is the location of the temporal summation. r is the result of the temporal summation, the membrane activity, x is the spiking activity of the pre-synaptic neuron and v is the post-synaptic potential or basis function. To simplify we will work on a discrete time level. The potential v lasts p time steps. p will define for the rest of this article the size of the vector v. It is then possible from those parameters to write the membrane potential r of the post-synaptic neuron as a function of time t:

$$r(t) = \sum_{i=1}^{p} x_{t-p+i} v_i \tag{1}$$

The u^t vector can define the sequence of values from x_{t-p} to x_t, which means the sequence of the input activity x from time step $t-p$ to time step t. It implies that the u vector is also of size p like the v vector. We can hence simplify equation 1 in this manner:

$$r(t) = \sum_{i=1}^{p} u_i^t v_i = u^t v \qquad (2)$$

2.2 Supervised learning

The learning skills of the TempUnit neurons have been already demonstrated in Manette and Maier, 2006 but the learning equations have not been explicitly published. Let $f(t)$ be a temporal function that we wish to learn with TempUnit. The learning algorithm should make $r(t)$ reach $f(t)$ by modifying the weights v_i of the basis function vector. For every i from 1 to p, it is possible to follow this rule:

$$\forall i \ de \ 1 \text{ à } p : \frac{dv_i(t)}{dt} = \gamma\big(f(t) - r(t)\big)x_{t-p+i} \qquad (3)$$

This equation gives very good results and a very good convergence; see Manette & Maier, 2006, for more details.

2.3 Evolution of r analysis

Let us see how the output signal evolves as a function of the time and as a function of the basis function v:

$$\frac{dr(t)}{dt} = x_{t+1}v_p + \left[\sum_{i=2}^{p} u_i^t(v_{i-1} - v_i)\right] - u_1^t v_1 \qquad (4)$$

We observe in equation 4 that $dr(t)/dt$ shows an evolution depending only on the new input value: x_{t+1}, which indicates that this is a decision node depending on the new binary value x_{t+1}. Thus the next spike is able to define if the following time activity of the TempUnit will follow one direction or another.

2.4 The graph of the neural activity

Equation 4 shows that the output of the TempUnit neuron can at every time step takes two different directions depending on the new binary input, i.e. whether a new spike arrives at instant t+1 or not. According to this principle, it is thus possible to build a graph that represents the entire neuron activity. We can define, F as a vector space and $(c_1, \cdots, c_n)$ a basis of F. In the current case with only one input (pre-synaptic neuron) and only one TempUnit (post-synaptic neuron), $n = 2$. We can, using this vector space, project the entire set of input vector u^t by calculating the coordinates c_1 and c_2 as follows:

$$u^t = \begin{cases} c_1 = \sum_{i=1}^{p} 2^{i-1} u_i^t \\ c_2 = \sum_{i=1}^{p} 2^{p-i} u_i^t \end{cases} \qquad (5)$$

Given that every u vector is of size p, as defined earlier, and contains only binary data, there exist exactly 2^p different input vectors. Our vector space contains thus this exact amount of nodes. Every vector u^t is thus represented in the vector space by a point with coordinates (c_1^t, c_2^t). According to equation (4), every single point in this vector space is a decision node from where it is possible to follow two different paths on the graph through two different nodes. We can calculate the coordinates of the vertices that connect the nodes to each other:

$$a^t = \begin{cases} 2^{p-1}x_{t+1} - \sum_{i=2}^{p} 2^{i-1}u_i^t - u_1^t \\ x_{t+1} + \sum_{i=2}^{p} 2^{p-i}u_i^t - 2^{p-1}u_1^t \end{cases} \tag{6}$$

The coordinates of vertices a^t depend only on u^t, i.e. directly from the previous node in the vector space, and on the presence or absence of a spike at the next time step $t+1$. As a consequence, at every node in the space F as defined by the basis (c_1, c_2), there exist only two vertices reaching two other nodes of this space. Thus there exist 2^{p+1} vertices in the space F.

(c_1, c_2) makes a basis because every vector u^t can be associated with only a unique pair of coordinates in space F. It is therefore easy to move from a vector u^t to the coordinates of a particular node of the graph and vice versa. In reality, c_1 and c_2 are both a basis and it is unnecessary to know both coordinates of a specific node to be able to identify the related vector u^t. Let us take the case of the calculation of vector u^t using only c_2:

$$\text{for i=1 :} \quad \begin{cases} u_1^t = 1 \; si \; c_2^t \geq 2^{p-1} \\ u_1^t = 0 \; else \end{cases}$$
$$\forall i \; from \; 2 \; to \; p : \quad \begin{cases} u_i^t = 1 \; si \; c_2^t \geq \left(2^{p-i} + \sum_{j=1}^{i-1} 2^{p-j}u_j^t\right) \\ u_i^t = 0 \; else \end{cases} \tag{7}$$

Example:
To illustrate this, we will calculate the coordinates of the node in F of the vector $u^t = [0010]$. In this case, $u_3^t = 1$ while $u_1^t = u_2^t = u_4^t = 0$, so that, taking into account equation (5), we can calculate the coordinates c_1 and c_2 which are: $u^t = (2^{3-1}, 2^{4-3}) = (4,2)$. From equation (6), we can calculate also the two vertices a_1^t and a_2^t from this node: $a_1^t = [-2,2]$ and $a_2^t = [6,3]$. We know then that from the original node u^t it is possible to reach the next node of coordinates $u^{t+1} = (2,4)$ or the other node $u^{t+1} = (10,5)$. Using equation (7) we can determine the corresponding input vectors $u^{t+1} = [0100]$ and $u^{t+1} = [0101]$ respectively.
Graphical representation:
In the case of only one TempUnit with only one binary input, it is still possible to make a graphical representation of the graph of the neural activity. It is then easier to understand how information is organized on the graph. Figure 1 presents in a schematic fashion a very small graph of neuronal activity containing only 16 nodes.
Figure 2, by contrast, is drawn using the exact coordinates c_1 and c_2 as calculated from equation (5), with vertices as calculated from equation (6).

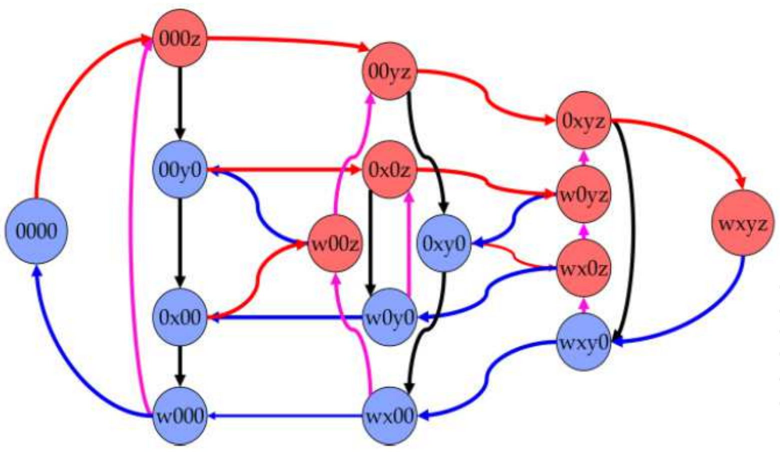


Fig. 1. This diagram illustrates the organization of a graph of neural activity for a TempUnit which has a basis function of four time steps in length. This makes, therefore sixteen different combinations of the inputs which are represented by a colored circle, red or blue as a function of the presence or absence of a spike in the bin at the extreme left respectively. Absence of a spike in a bin is represented with a '0' in the 4-element code in every circle. Presence of a spike is symbolized with one of the letters w, x, y or z depending on the position of this specific spike within the train of four time steps. Vertices are drawn with a colored arrow. Red arrows indicate an increase in the global number of spikes in the input train u^t ; blue arrows indicate the decrease of one spike in the input vector; black color typify that there is no global modification in the amount of spike but there are no spikes in the next time step t+1. Pink arrows indicate that there is no modification of the global number of spikes but a spike will arrive at the next time step.

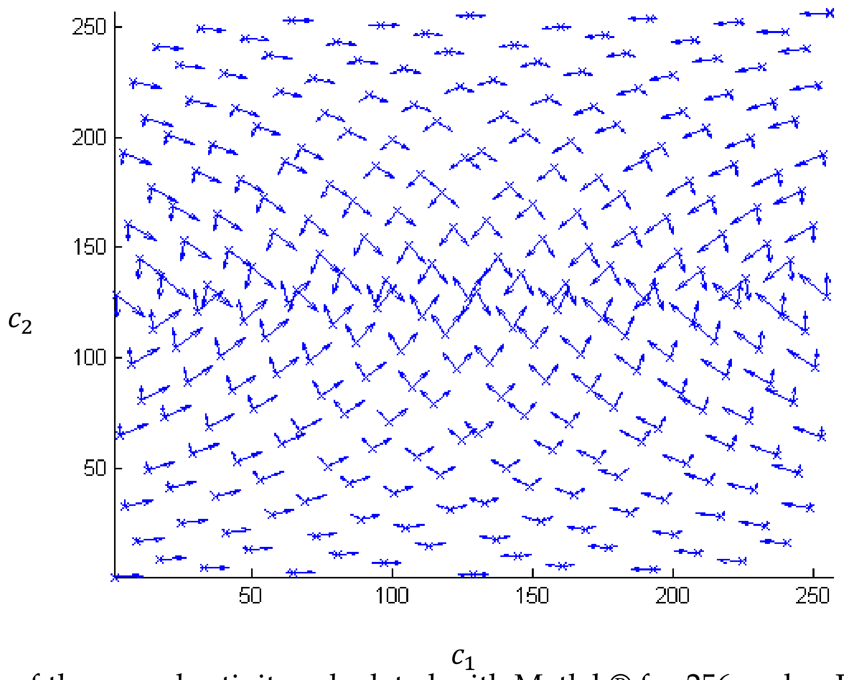


Fig. 2. Graph of the neural activity calculated with Matlab® for 256 nodes. Each node is represented by an 'x' and the beginning of each vertex by an arrow. We clearly observe trends in orientation related to the localization of the vertices.

2.5 Inverse function

It is possible to draw on the same graph of neural activity both the input and the output of the TempUnit neuron with a particular color code. As well as in figure 2 where we can see trends in the vertex directions and positions, in figure 3 & 4 we can observe that outputs values are not organized on a randomly but on contrary as a function of their intensity. Of course, the organization of the outputs on the graph is directly based on the basis function v. Nevertheless, in spite of great difference that can be observed from one basis function v (e.g. fig. 3) to another (e.g. fig. 4), it is still possible to establish some similarities which enable us to calculate an inverse function.

An inverse function enables calcultation of the input corresponding to a desired output. The inverse of the function in equation (2) is a surjective function for the most of the sets of weights in the basis function v and this is the main problem. Because a surjective function means that for a specified value there can exists more than one possible answer and it is difficult to determine which of all those possible answers is the one that is needed. Indeed in figure 3 there are areas of the same color including many input nodes, which means of the same output values. The graph of neural activity gives a way to determine which of those possible values is the good one.

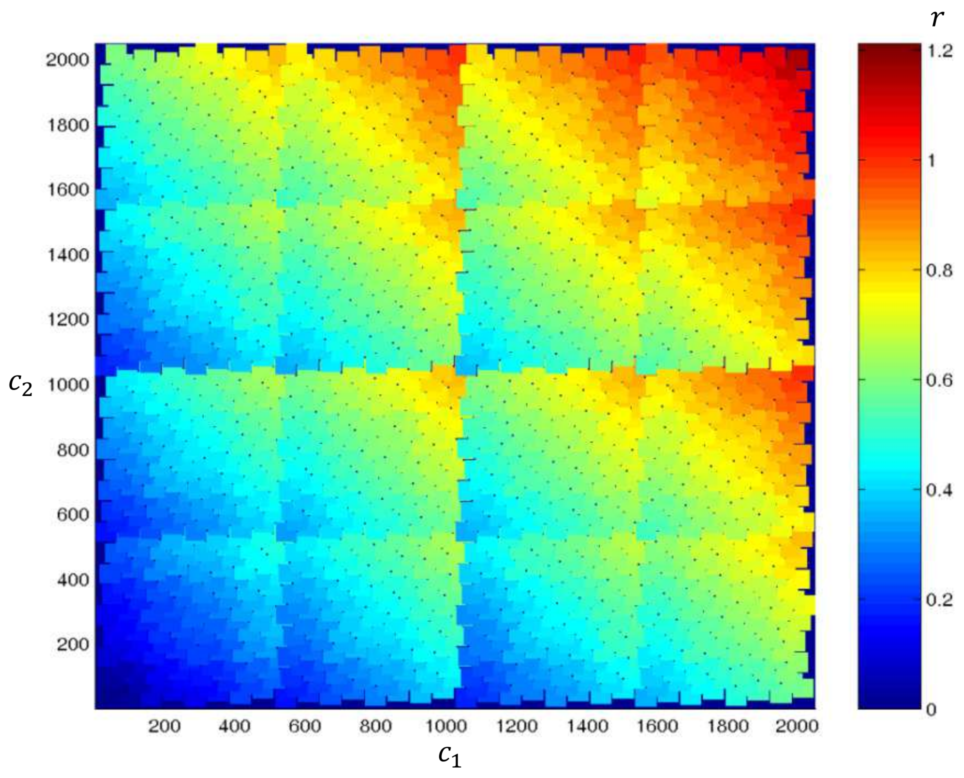


Fig. 3. Graph of neural activity showing all the possible outputs of a TempUnit neuron using the same system of coordinates of the input from equation (5). The input u and the basis function v are in this case of size $p = 11$ time steps, thus there are 2048 nodes in the graph. The basis function v is an inverted Gaussian function: $v_i = 0.2 - {e^{i-(p/2)^2/2(p/5)^2}}\Big/{p\sqrt{2\pi}/5}$.

The color code represents the intensity of the output value. Red indicates higher values while blue indicates lower values.

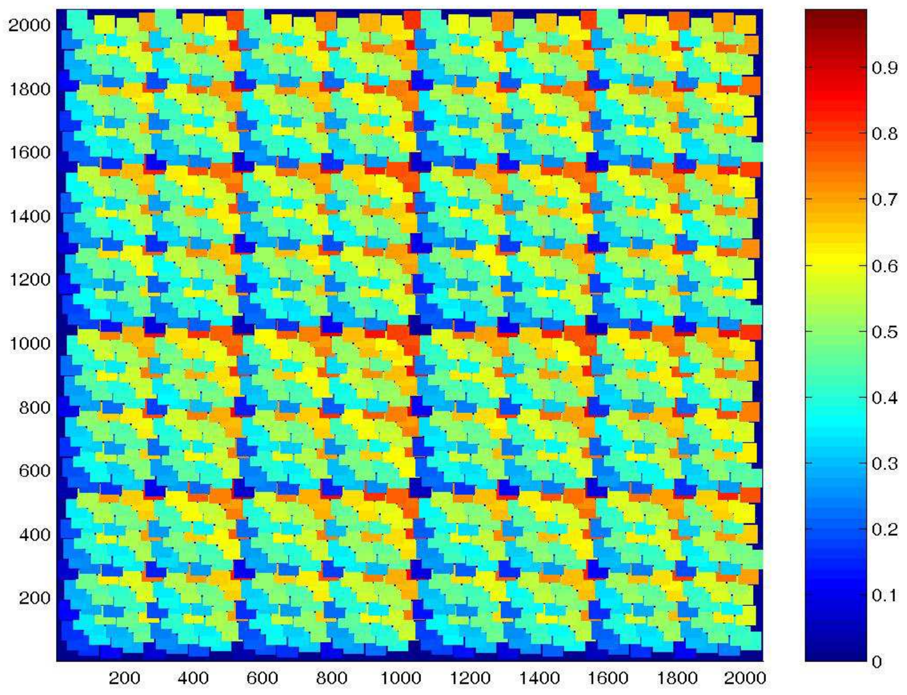


Fig. 4. Graph of neural activity showing the output using a color code. The basis function v is a Gaussian function: $v_i = 0.2 - e^{i-(p/2)^2/2(p/5)^2} \Big/ p\sqrt{2\pi}/5$ which is compatible with the "value" type of neural coding (Salinas & Abbott, 1995, Baraduc & Guigon, 2002).

In brief, the inverse function algorithm which from a sequence of temporal desired output values gives back the path on the graph of the neural activity and then the temporal sequence of the related input, is composed of five steps:

1. Determine the set of coordinates on the graph associated with a given output at time t.
2. From the set of the selected nodes in 1), calculate the set of nodes for time step t+1.
3. In the set of nodes for time step t+1 delete all the nodes which do not correspond to the desired output value at time t+1 and delete as well all the nodes in the subset corresponding to time step t which are not reaching any of the remaining nodes of the subset t+1.
4. Start again from step 1 while in the subset t there is more than one possible solution. Instant t+1 becomes instant t and instant t+2 becomes instant t+1.
5. From the sequence of coordinates on the graph, calculate the corresponding binary input using equation (7).

Why to try to determine on the graph the set of nodes leading to a particular output value? Simply because on the graph the output values are organized as a function of their intensity, we will see in the following that only a simple calculation is necessary to find out all the researched nodes. Another important point is that it is also possible to use the links between the nodes in order to reduce the indeterminacy of the multiple possible results.

How are the output values organized on the graph?

Studying the relationship between the coordinates of the nodes on the graph $r = g(c_1)$, one observes that it is a periodic function containing multiple imbricated periods and generally growing especially if v is growing. Periods of g are always the same and are completely independent of the function v but only depend on p: $T_1 = 2^{p-1}, T_2 = 2^{p-2}$, etc. … It is hence easy from a small amount of data to deduce the complete set of possible output values because of this known property of periodic functions:

$$\forall i\ 1 \le i \le p-1 : \qquad g(c_1 + T_i) = g(c_1) + m_i \tag{8}$$

For instance, if p=5, the graph contain 2^5=32 nodes. The function $r = g(c_1)$ has hence 5-1=4 imbricated periods: $T_1 = 2^{5-1} = 16, T_2 = 8, T_3 = 4, T_4 = 2$. To infer all the 32 possible output values on the graph, one should first calculate $r = g(1)$, $r = g(2)$ then $r = g(3)$ which following equation (8), allow one to calculate $m_4 = g(3) - g(1)$. $m_3 = g(5) - g(1)$, $m_2 = g(9) - g(1)$ and $m_1 = g(17) - g(1)$. Thus, from only these six values we can deduce the complete set of the 32 values of the graph. The search algorithm in the output values could be similar to already known algorithm of search in an ordered list.

Figure 5 shows an example of the function $r = g(c_1)$ for a basis function $v_i = i^3$. Of course, the function $r = g(c_1)$ varies as a function of v. But, as in figure 4, it is always a periodic function. Drawing a horizontal line at the level of the desired output value on the y-axis gives a graphical solution of the point 1) in the inverse function algorithm. The set of searched-for nodes corresponds to all the coordinates on the x-axis every time the horizontal line crosses the function $r = g(c_1)$. In the case of the example shown in figure 4, one can see that a specified output value (r) is observed only a few times (maybe 3 or 4 times at maximum), but depending on the shape of the function v it could be much more.

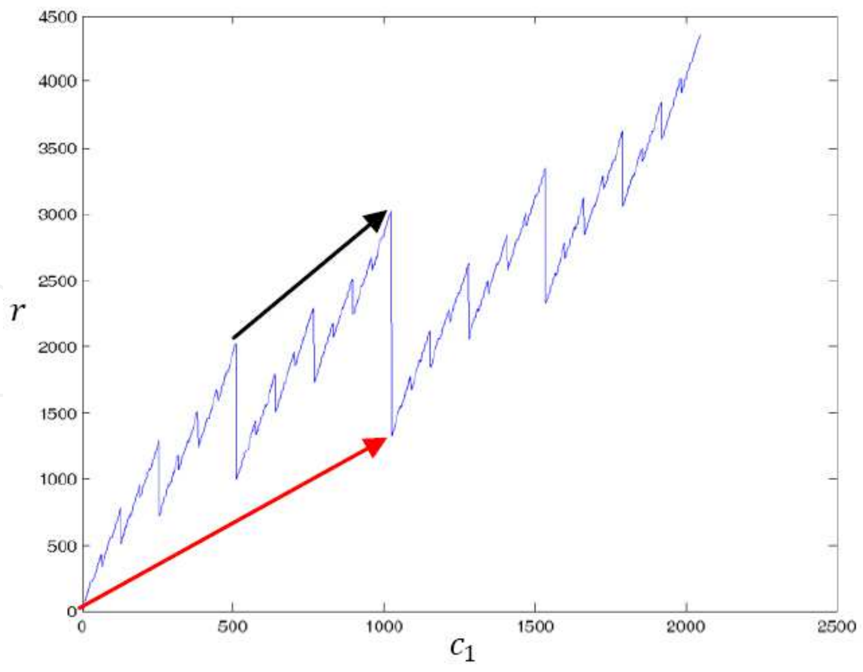


Fig. 5. Example of function $r = g(c_1)$ with p=11 giving 2048 nodes in the graph of neural activity. The basis function is $v_i = i^3$. The red vector indicates the vector m_2 while the black one indicates the vector m_3.

Once all the nodes have been selected according to a desired output value, one can search the connected nodes at the next time step. Equation 6 enables us to calculate these connected nodes at the next time step. All the nodes of the subset selected for time t+1 which are not associated with the next desired output value at time t+1 are deleted. In addition, all the nodes from the subset for time t which are after this no longer linked with any other node of the subset t+1 are also deleted. It is possible to continue with the same principle of deleting independent nodes which are not linked with any other node of a following subset. The algorithm finishes when it obtains only one possible pathway on the graph, which should represent the searched-for input command. Obviously this algorithm gives the exact inverse function. If there could be some imprecision on the desired output value, this method should not change except that nodes associated with close values should also be selected and not only nodes with the exact desired value.

Partial inverse function

In some cases of motor control, it may be better to only ask for the final desired position and let the system determine by itself all the intermediate steps. On the graph it is easy to determine all the intermediate values. It is particularly simple to determine all the intermediate values with the help of the graph. Indeed, the initial state as well as the final state should be defined by nodes on the graph and the path between those nodes is an already well known problem because it is equivalent to an algorithm for finding the shortest path on a graph, about which it is unnecessary to give more details.

2.6 Complexity

Since so far we have only investigated the capacity of a single TempUnit with a single input, we will next investigate the capacities of other TempUnit network architectures. We will see in the following that the signal generator abilities of a specified TempUnit network depends directly on its architecture. Each type of TempUnit network architecture corresponds to a particular kind of graph with very precise characteristics. The complexity of the neural network can be calculated based on the complexity of the emergent graph because the graph represents exactly the way the TempUnit network behaves. This gives a means of determining in a very precise fashion the kind of neural network architecture needed for a given type of generated temporal function characteristics.

One TempUnit with many inputs

Taking account all the Sk binary inputs of the TempUnit, equations 1 and 2 become:

$$r(t) = \sum_{i=1}^{p} \sum_{s=1}^{Sk} x_{s,t-p+i} v_i = \sum_{s=1}^{Sk} u_{t,s} v_i \tag{9}$$

Figure 6 gives in a schematic fashion the architecture of equation 9.

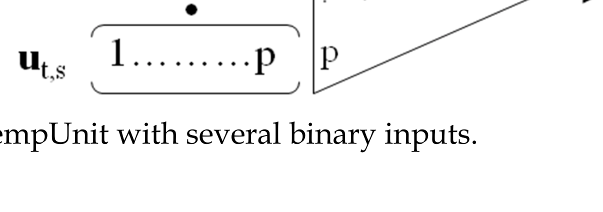


Fig. 6. Schema of one TempUnit with several binary inputs.

All the inputs can be summed equally to create a new global *u* vector and a new global *x* vector that contain all the summed input. This architecture is still biologically compatible; in particular, the resulting potential is much larger for near coincident arrival of spikes (Abeles, 1991; Abeles et al 1995). The evolution of the output *r* is then equivalent as what has been written in equation 4 but encoded with more than 1 bit of information. In this case every decision node can be connected with more than two other nodes. The global number of nodes in the graph is: $(Sk + 1)^p$

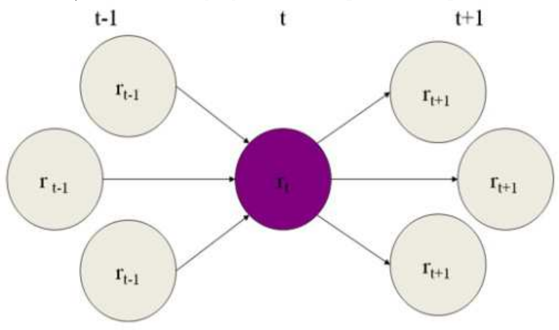


Fig. 7. Because all the inputs are equivalent, the degree of the graph is equal at $Sk + 1$. In this example is represented a node at time *t* with its previous and following connected nodes at times $t - 1$ and $t + 1$ respectively. In the case of one TempUnit with two different binary inputs, the graph becomes of vertex degree 3.

Many TempUnits with one input

In this case inputs are not equivalent because every input connects to different TempUnit. In a network of *N* TempUnits, there are 2^N vertices going from and to every node and the graph contains 2^{Np} nodes. Equation 10 shows the calculation of the output in this TempUnit architecture.

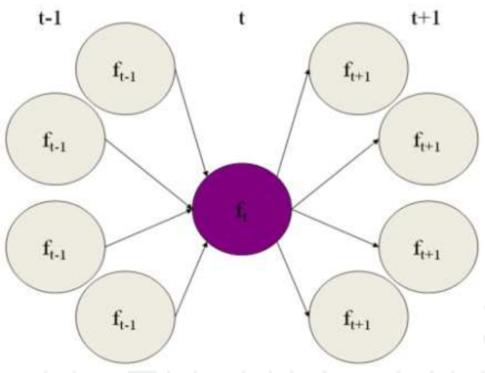


Fig. 8. Example of a node with its connected nodes at time $t - 1$ and at time $t + 1$ for a network of $N = 2$ TempUnits. In this case the graph of the global neural activity is of vertex degree $2^2 = 4$.

$$r(t) = \sum_{i=1}^{p}\sum_{j=1}^{N} x_{j,t-p+i} v_{i,j} = \sum_{j=1}^{N} u_{t,j} v_{i,j} \tag{10}$$

We have seen in this section that the evolution of the graph of neural activity depends directly on the TempUnit network architecture. The complexity of the graph could be defined by the number of elementary vertices that it contains, in other word, the number of nodes that multiply the vertex degree of the graph.

3. Conclusion

We have seen in this chapter the central role played by the graph of neural activity helping to understand the behavior of the TempUnit model. The graph gives the possibility of calculating the inverse function as well; it is also because of the graph that we can better understand the relationship between the TempUnit network architecture and its signal generator abilities. The graph of neural activity represents very clearly the behavior of the entire TempUnit network. The graph is not really implemented, only the TempUnit model is as defined on equation 1 or 9. Additionally, the graph of neural activity can be seen as the "software" while the TempUnit network would be the "hardware". In any case, a great advantage of the graph is that all the already-known algorithms in graph theory can then be applied to TempUnits.
A first use of the graph has been to solve the inverse function problem. Relationship between the nodes give us a mean to avoid any ambiguity produced by this surjective function.
In the control of task context, the existence of Central Pattern Generators (CPG) in the brain has been suggested based on work showing complex movements in deafferented animals (Brown, 1911; Taub, 1976; Bossom, 1974, Bizzi et al, 1992). Even if couples of "motor command"/"specific movement" are hardly credible, suggestions of motor schemes (Turvey, 1977) able to generate sets of specific kinds of movements (Pailhous & Bonnard, 1989) are more appealing. At the same time other data show that the motor activity depends also on sensory feedback (Adamovich et al, 1997). It would seem that instead of having two opposite theories, one more predictive using only CPGs and another completely reactive using mainly reflexes triggered by sensory feedback, a compromise could be found. Indeed, in considering the neural activity graphs of TempUnit networks one can see that TempUnit can work as a feedforward function generator. The entire graph could represent a complete set of a kind of movement while a path in the graph could constitute a particular execution of this movement. Furthermore, at every time step TempUnit is at a decision corner and the new direction depends on the inputs. We can imagine a sensory input that can then influence the TempUnit behavior to be able to adapt the movement command as a function of the sensory parameters. The TempUnit activity cannot escape from the path defined on the graph; hence these kinds of networks are naturally shaped to follow constrained rules like syntax. As well, since it is impossible to reach any arbitrary node from a given specific node, these kinds of networks are well suited for speech generation or motor control where it should not be possible to ask the system to reach any arbitrary position of the limb from another defined position.

4. Acknowledgements

This work was supported by research grant EC-FET-Bio-I3 (Facets FP6-2004-IST-FETPI 15879).

5. References

Abeles M., Corticonics: *Neural Circuits of the Cerebral Cortex*. Cambridge: Cambridge University Press., 1991.

Abeles M., H. Bergman, I. Gat, I. Meilijson, E. Seidenmann, N. Tishby, and E. Vaadia, "Cortical Activity Flips among quasi-Stationary States.," *Proc. Natl. Acad. Sci.*, vol. 92, pp. 8616-8620, 1995.

Adamovich S.V., Levin M.F, Feldman, A.G. (1997) Central modification of reflex parameters may underlie the fastest arm movement. Journal of Neurophysiology 77: 1460-1469.

Baraduc P. and E. Guigon, "Population computation of vectorial transformations.," *Neural Computation*, vol. 14, pp. 845–871, 2002.

Bernstein N.A. (1967) *The Coordination and Regulation of Movements*. London, Pergamon Press.

Berthoz A. (1996) *Le sens du mouvement*. Paris : Odile Jacob

Beevor C.E. (1904). *The croonian lectures on muscular movements and their representation in the central nervous system*. London: Adlar

Bizzi E., Hogan N., Mussa-Ivaldi F. A., Giszter S. (1992) Does the nervous system use equilibrium-point control to guide sigle and multiple joint movements? *Behavioral and Brain Sciences* 15: 603-613

Bossom J. (1974). Movement without proprioception. *Brain Research* 71: 285-296.

Brown T.G. (1911) The intrinsic factors in the act of progression in the mammal. *Proceeding Royal Society,* London, Series B, 84: 308-319.

Desmurget M, Grafton S. Forward modeling allows feedback control for fast reaching movements. *Trends Cogn Sci.* 4: 423-431, 2000.

Gribble P.L., Ostry, D. J. (1996) Origins of the power law relation between movement velocity and curvature: modeling the effects of muscle mechanics and limb dynamics. *Journal of Neurophysiology* 6: 2853-2860.

Hogan N., Flash T. (1987). Moving gracefully: quantitative theories of motor coordination. *Trends in Neuroscience* 10: 170-174.

Hornik K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks* 4(2): 251-257

Kawato, M. (1999). Internal models for motor control and trajectory planning. *Current Opinion in Neurobiology*, 9, 718-727.

Keele S.W. (1968). Movement control in skilled motor performance. *Psychology Bulletin* 70: 387-403.

Manette, O.F.; Maier, M.A. (2006). TempUnit: A bio-inspired neural network model for signal processing. *Neural Networks, 2006. IJCNN 06. International Joint Conference on.* Page(s):3144 – 3151

Mainen Z. F. and T. J. Sejnowski, "Reliability of spike timing in neocortical neurons," *Science*, vol. 268, pp. 1503–1506, 1995.

Pailhous J, Bonnard M (1989) Programmation et contrôle du mouvement. In : *Traité de psychologie cognitive*, C. Bonnet, R. Ghiglione, J-F Richard. (Eds) . Paris : Dunod, 129-197

Rieke F, Warland D, De Ruyter Van SteveninkR, Bialek W,Characterizing the neural response in: *Spikes: Exploring the Neural Code* . Cambridge, MA: MIT Press, 1996.

Salinas E. and L. Abbott, "Transfer of coded information from sensory to motor networks," *Journal of Neuroscience*, vol. 15, pp. 6461–6474, 1995.

Seashore C.E. (1938) *Psychology of music*. New York: Academic Press.

Shadmehr R, Mussa-Ivaldi F. (1994) Adaptive representation of dynamics during learning of a motor task. *Journal of Neurosciences* 14: 3208-3224.

Shepherd, G. M. (1990). The Significance of Real Neuron Architectures for Neural Network Simulations. In: *Computational Neuroscience*, E. L. Schwartz, ed., chapter 8, pages 82--96. A Bradford book, MIT Press.

Sherrington C.S. (1906/1947) *The integrative action of the nervous system*. Yale University Press.

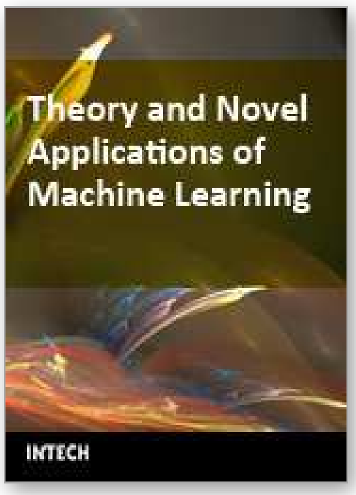
Taub E. (1976) Movement in nonhuman primates deprived of somatosensory feedback. *Exercise and sports Science Review* 4: 335-374

Turvey M.T. (1977) Preliminaries to a theory of action with reference to vision. In: *Perceiving, acting and knowing: toward an ecological psychology*, R. Shaw, J. Bransford (Eds.),. Hillsdale, (N.J.) Lawrence Erlbaum Ass.

Venugopal, Venu, Walter R.J. Baets, (1994). "Neural networks and statistical techniques in marketing research : a conceptual comparison." *Marketing Intelligence & Planning* 12.7: 30-38.

Wierenga, B. & J. Kluytmans (1994). Neural nets versus marketing models in time series analysis: A simulation study in: Bloemer, J. e.a., Marketing: its Dynamics and Challenges, *Proceedings 23 rd. EMAC Conference*, Maastricht 17-20 May, pp. 1139-1153.

Wolpert D.M., Gharamani Z, Ardan, M.J. (1995). An internal model for sensorimotor integration. *Science* 269: 1179-1182



Theory and Novel Applications of Machine Learning
Edited by Meng Joo Er and Yi Zhou

ISBN 978-953-7619-55-4
Hard cover, 376 pages
Publisher InTech
Published online 01, January, 2009
Published in print edition January, 2009