

Hebbian Learning in Neural Networks with Gates

Jean-Pierre Aubin¹ & Yves Burnod²

May 3, 2002

1 Introduction

Experimental results on the parieto-frontal cortical network clearly show that

1. in all parietal and frontal cortical areas involved in reaching, more than one signal influences the activity of individual neurons for *learning a large set of visual-to-motor transformations*,
2. they enjoy gating properties that can be simply modeled by “tensor products” of vectorial inputs, known in the language of neural networks as $\Sigma - \Pi$ units.

Control theory allows us to present in a unified way such neural networks with gates as well as many examples of neural networks by regarding them as discrete or continuous control systems, the states of which are the *signals* and the controls are the *gating tensors*, including the usual *synaptic matrices*. We refer to (Aubin, 1996) and its bibliography for an approach using this viewpoint.

It allows us to derive *learning algorithms*, such as the *back-propagation formula* for neural networks with gates (which is nothing other than the *adjoint equation* in control theory). The mathematical structure of the space of multilinear operators involving tensor products, we then obtain in the formulas describing the learning algorithms *pure tensor products*: The corrections of the gating tensor are the products of the presynaptic activities of the signals arriving at the gate and postsynaptic activities. This still captures the reinforcement characteristic of *Hebbian rules* in learning mechanisms.

Here, we reconsider the article (Aubin, 1995) summarizing this point of view in the 1995 edition of this handbook in the framework of neural networks with gates motivated by the neurophysiological data presented in the first section. A formal definition of neural networks with gates is given in the second section. The definition and the basic properties of tensor products are given in the third section, next applied for deriving learning rules for one and two layer networks with gates in the fourth section and to “heavy algorithms” in the fifth section.

2 Neurophysiological Motivations

2.1 The Parieto-Frontal Cortical Network

The parieto-frontal cortical network *can learn a large set of visual-to-motor transformations*: Different studies from diverse neuroscience disciplines have shown the crucial role of parietal and frontal areas in transformations relating the eyes and the hand, such as visually-guided reaching (for recent review see Mountcastle, 1995 ; Caminiti et al., 1996; Wise et al., 1997).

¹Centre de Recherche Viabilit, Jeux, Contrle, Universit Paris-Dauphine

²CREARE, INSERM, Universit Pierre et Marie Curie

Combined anatomical and physiological experiments have shown how *different populations of neurons in the parietal and frontal areas can be tuned for at least four different sets of sensorimotor signals* involved in visual-to-motor transformations, such as visually-guided reaching:

1. retinal signals x_1 ,
2. signals indicating gaze position and direction x_2 ,
3. signals on arm position and direction x_3 ,
4. signals controlling muscle outputs x_4 .

The different classes of neurons encoding information relevant to arm reaching are not confined within individual cortical areas, but change gradually along the tangential cortical domain, so as to define a visual-to-somatic “gradient” (with retinal, gaze, arm and muscle signals). This “gradient” is symmetrical with respect to the central sulcus in both parietal and frontal cortices, in relation with the cortico-cortical association connections.

2.2 The Critical Combinatorial Properties of Neurons

Neurophysiological studies of the parieto-frontal network do not reveal univocal relationships of neural activity, in a given cortical area, to a certain sensory or motor system, such as a purely “visual” or “motor” system. Experimental results clearly show that in all parietal and frontal areas involved in reaching, more than one signal influences the activity of individual neurons.

1. in the anterior par of the parietal lobe (area 5), and in the related motor area in the frontal lobe (area 4), neurons $y_1 = f_1(x_3, x_4)$ and $y_2 = f_2(x_3, x_4)$ combine two sets of information, positional and directional signals x_3 on the arm (arm configuration, joint angles) and information x_4 on muscle dynamics;
2. in the intermediate parietal areas (area MIP) and in intermediate premotor areas (PMdc), neurons $y_3 = f_3(x_2, x_3)$ and $y_4 = f_3(x_4, x_3)$ combine arm and gaze positional and movement directional information;
3. in the posterior parietal areas (areas 7m, V6) and related anterior premotor areas (PMdr), neurons $y_5 = f_5(x_1, x_2, x_3)$ and $y_6 = f_6(x_1, x_2, x_3)$ combine visual inputs x_1 on the retina, positional and directional information x_2 on gaze, and arm related signals x_3 .

All neurons related to reaching in the parieto-frontal network have thus “combinatorial” properties: They combine at least two of the four sets of sensorimotor signals involved in visual-to-motor transformations (retinal, gaze, arm position, muscles output) along the visual-to-somatic gradient.

The variety of neuronal activities in the parietal lobe can be described by their optimal tuning to the position and direction of sensory stimuli x_1 , position and direction of gaze x_2 and to position and direction of arm joints x_3 . The change in activity around this optimal tuning is quantitatively described either for positions by a linear variation around optimal positions (eye position, arm joints), or for directions by a cosine of the angles.

The main property of parietal neurons is that they are tuned for more than one signal, i.e., they have several optimal tunings in several sensory or motor reference frames.

These combinations are well described quantitatively by measuring the changes in a tuning curve to the first input when the second varies: When the tuning curve for one signal does not change its optimal value (preferred direction or position), but is quantitatively modulated by the second input, this is acting as a *gain or a gating factor*: for example in the posterior part of

the parietal lobe, the neurons which are tuned for a given retinal position of the visual stimulus are gated by an eye-position signal (Andersen et al., 1987; Zipser and Andersen, 1988).

The two pieces of information are independent: the directional tuning of these neurons for retinal signals does not change with the eye position signal.

Vergence and disparity are combined in a similar manner in the primary visual cortex (Trotter et al., 1992). Neurons have a selective tuning for disparity (far, near, tuned), with a gain factor depending upon vergence.

The gating properties of such neurons can be simply modeled by the “tensor product” of the two vectorial inputs, known in the language of neural networks as $\Sigma - \Pi$ units (see (McClelland, Rumelhart, 1986) for instance). Such bilinear operations agree with the quantified properties of populations of cells in motor and premotor cortices, studied during visually-guided 3-D reaching movements (Burnod et al., 1992). These tensor products, that can be called “gating tensors”, play the role of synaptic matrices in classical networks, will be studied mathematically in the next sections.

2.3 Generic Combinatorial Properties

Generic combinatorial properties allow the network to learn a large variety of visuo-motor transformations: Neurophysiological studies suggest that *the same basic set of neuronal combinatorial properties can be the unique substratum to learn a large variety of visuo-motor transformations through local connections*, and do not show a specialization of cortical areas for a given visuo-motor operation (such as foveated reaching). The process of learning a complex sensorimotor transformation can be understood as resulting from the cumulative effect of simple learning stages based on the combinatorial domains of the parieto-frontal network:

1. The population of neurons $y_1 = f_1(x_3, x_4)$ which combines motor commands with the actual and reafferent somatic inputs (anterior parietal) can learn to predict the sensory reafference of the motor command (depending on the arm configuration) ; the same population can also learn to generate adequate motor commands to produce the expected sensory effects.
2. The population of neurons $y_3 = f(x_2, x_3)$ which combines arm and gaze positions and directions (intermediate parietal)
 - (a) can learn the visual tracking of the hand trajectory;
 - (b) can learn to control the direction of reaching movements to foveated targets, even if the hand is not in the visual field;
3. The population of neurons $y_5 = f(x_1, x_2, x_3)$ which combines successive retinal and gaze information (posterior parietal) can learn to make accurate gaze movements to foveate retinal signals using predictions from previous gaze movements and retinal information; the same population of neurons can predict the next retinal position of an object after a saccade and thus learn visual invariants.
4. The population of neurons $y_3 = f(x_2, x_3)$
 - (a) together with the population y_5 which combines retinal and gaze signals, can learn to correct the arm direction from the retinal image of the hand relatively to the foveated target;
 - (b) together with the population of neurons y_5 which have learned visual invariants, can then learn the direction of reaching movement to non-foveated targets.

The nature of the sensory-motor combinations made by anatomical connections in the parieto-frontal network could thus appear as an evolutionary solution to learn a large variety of visuo-motor transformations through local connections.

3 Neural Networks with Gates

The above neurophysiological data suggest to replace synaptic matrices by “gating tensors”, so to speak, as support of knowledge, parametrizing the propagation of signals.

For instance, in the case of one-layer networks, we replace the neural networks

$$\forall j = 1, \dots, m, \quad y_j = g_j \left(\sum_{i=1}^n W_{ij} x_i \right)$$

involving the synaptic matrix $W = (W_{ij}) \in \mathcal{L}(\mathbf{R}^n, \mathbf{R}^m)$ by “gated neural networks”

$$\forall j = 1, \dots, m, \quad y_j = g_j \left(\sum_{l_1=1}^{n_1} \cdots \sum_{l_I=1}^{n_I} M_{l_1, \dots, l_I, j} x_{1, l_1} \cdots x_{I, l_I} \right)$$

involving the “gating tensor” $M = (M_{l_1, \dots, l_I, j})$ when $l_i = 1, \dots, n_i$ ($i = 1, \dots, I$) and $j = 1, \dots, m$. Such tensors are called “ $\Sigma - \Pi$ units” in the language of neural networks. The input signals x_i range over the space \mathbf{R}^{n_i} ($n_i \leq n$) and the output signal y over the space \mathbf{R}^m .

By the way, synaptic matrices being special cases (for $I = 1$) of gating tensors are not excluded from this framework. Actually, we shall consider two-layer networks, the first layer using gating processes of coalitions of neurons — describing cortical areas — and the second layer using synaptic matrices.

In order to find the gating tensors and synaptic matrices which “learn” a given set of input-output patterns, we shall have to apply algorithms (such as Newton’s Algorithm, Gradient Algorithms, Heavy Algorithms) to adaptive systems controlled by gating tensors and synaptic matrices, and thus to use specific properties of the spaces of multilinear operators (regarded as tensor products).

Since Hebb’s 1949 classic book ORGANIZATION OF BEHAVIOR, most of the studies of neural networks deal with numerous variations of *learning rules* which prescribe a priori the evolution of the synaptic matrix $W^n = (W_{i,j}^n)$ through an algorithm of the form:

$$W_{i,j}^{n+1} - W_{i,j}^n = \alpha_i^n \beta_j^n$$

where the correction of the synaptic weight $W_{i,j}^n$ of the synapse (i, j) is proportional to the product of presynaptic and postsynaptic activity.

Using instead gating tensors $M^n = (M_{l_1, \dots, l_I, j}^n)$ when $l_i = 1, \dots, n_i$ ($i = 1, \dots, I$) and $j = 1, \dots, m$, the natural adaptation of the Hebb learning rule would prescribe the evolution of the gating tensors through an algorithm of the form:

$$M_{l_1, \dots, l_I, j}^{n+1} - M_{l_1, \dots, l_I, j}^n = \alpha_{1, l_1}^n \cdots \alpha_{I, l_I}^n \beta_j^n$$

where the correction of the gating weight $M_{l_1, \dots, l_I, j}^n$ is proportional to the product of presynaptic activities of the coalition (l_1, \dots, l_I) of neurons and postsynaptic activity of neuron j .

Such multilinear operators are tensor products of the vectors $\alpha_1, \dots, \alpha_I$ and β . We shall see that the algorithms mentioned above, when applied to such neural networks with gates, i.e., yield learning rules which involve *tensor products of vectors*, and thus, remains in accordance with the Hebbian paradigm.

4 Tensor Products

Devising gradient algorithms for minimizing functions depending upon multilinear maps associated with neural networks with gates (or other algorithms) requires the computation of the gradients of functions defined on spaces of multilinear operators. This is where the tensor structure of the spaces of multilinear operators pops up, since such gradients involve tensor products of vectors, which yield “reinforcement” learning rules implying tensor products, which extend Hebbian rules stating that the reinforcement is the (tensor) product of presynaptic and postsynaptic signals to extended rules stating that the reinforcement is the (tensor) product of gated presynaptic signals and the postsynaptic signal.

Hence the task at hand is to compute the derivatives of these maps in order to write down the algorithms.

Let $X_i := \mathbf{R}^{n_i}$, ($i = 1, \dots, I$), $Y := \mathbf{R}^m$ be finite dimensional vector spaces, $X^* := \mathcal{L}(X, \mathbf{R})$ denote the *dual space* of X , which is the vector space of linear functionals $p : X \rightarrow \mathbf{R}$ on X . We recall that the transpose $W^* \in \mathcal{L}(Y^*, X^*)$ of a linear operator $W \in \mathcal{L}(X, Y)$ is defined by

$$\langle W^*q, x \rangle = \langle q, Wx \rangle \quad \text{for all } x \in X, \quad \text{for all } q \in Y^*$$

We associate with any $p^i \in X_i^*$ for $i = 1, \dots, I$ and $y \in Y$ their tensor product $\bigotimes_{i=1}^I p^i \otimes y := p^1 \otimes \dots \otimes p^I \otimes y \in \mathcal{L}_I\left(\prod_{i=1}^I X_i; Y\right)$, which is the multilinear operator defined by

$$(x_1, \dots, x_I) \rightarrow (p^1 \otimes \dots \otimes p^I \otimes y)(x_1, \dots, x_I) := \left(\prod_{i=1}^I \langle p^i, x_i \rangle \right) y$$

In particular, if we supply X_i with the basis $\{e^{l_i}\}_{l_i=1, \dots, n_i}$, its dual with the dual basis $\{e_{l_i}^*\}_{l_i=1, \dots, n_i}$, Y with the basis $\{e^j\}_{j=1, \dots, m}$ and its dual with the dual basis $\{e_j^*\}_{j=1, \dots, m}$, then the multilinear operators

$$e_{l_1}^* \otimes \dots \otimes e_{l_I}^* \otimes e^j$$

form a basis of $\mathcal{L}_I\left(\prod_{i=1}^I X_i; Y\right)$ when $l_i = 1, \dots, n_i$ ($i = 1, \dots, I$) and $j = 1, \dots, m$.

Hence the entries of the tensor $p^1 \otimes \dots \otimes p^I \otimes y$ in this basis are given by

$$(e^{l_1} \otimes \dots \otimes e^{l_I} \otimes e_j^*) (p^1 \otimes \dots \otimes p^I \otimes y) = \left(\prod_{i=1}^I \langle p^i, e^{l_i} \rangle \right) \langle e_j^*, y \rangle = \left(\prod_{i=1}^I p^{i, l_i} \right) y_j$$

The basic result states that

$$\mathcal{L}\left(\mathcal{L}_I\left(\prod_{i=1}^I X_i; Y\right), Z\right) = \mathcal{L}_I\left(\prod_{i=1}^I X_i^*; \mathcal{L}(Y, Z)\right)$$

from which we deduce by taking $Z := \mathbf{R}$ that

$$\mathcal{L}_I\left(\prod_{i=1}^I X_i; Y\right)^* = \mathcal{L}_I\left(\prod_{i=1}^I X_i^*; Y^*\right)$$

Let us consider pairs (X_i, \widehat{X}_i) and (Y, \widehat{Y}) of finite dimensional vector-spaces. Let $A_i \in \mathcal{L}(\widehat{X}_i, X_i)$ and $B \in \mathcal{L}(Y, \widehat{Y})$ be given.

We denote by $\bigotimes_{i=1}^I A_i^* \otimes B$ the linear operator associating with a multilinear operator $M \in \mathcal{L}_I\left(\prod_{i=1}^I X_i; Y\right)$ a multilinear operator $(\bigotimes_{i=1}^I A_i^* \otimes B)(M) \in \mathcal{L}_I\left(\prod_{i=1}^I \widehat{X}_i; \widehat{Y}\right)$ defined by

$$\left(\left(\bigotimes_{i=1}^I A_i^* \otimes B \right) (M) \right) (\widehat{x}_1, \dots, \widehat{x}_I) := BM(A_1 \widehat{x}_1, \dots, A_I \widehat{x}_I)$$

In particular, if $x_i \in X_i$ are fixed and if $B \in \mathcal{L}(Y, \widehat{Y})$ is given, the linear operator $x_1 \otimes \cdots \otimes x_I \otimes B$ maps any multilinear operator $M \in \mathcal{L}_I \left(\prod_{i=1}^I X_i; Y \right)$ to the space $\mathcal{L}_I(\mathbf{R}^I; \widehat{Y})$ defined by

$$(\lambda_1, \dots, \lambda_I) \rightarrow ((x_1 \otimes \cdots \otimes x_I \otimes B)M)(\lambda_1, \dots, \lambda_I) = BM(x_1, \dots, x_I) \prod_{i=1}^I \lambda_i$$

We observe that when $M = \bigotimes_{i=1}^I p^i \otimes y$, we have

$$\left(\bigotimes_{i=1}^I A_i^* \otimes B \right) \left(\bigotimes_{i=1}^I p^i \otimes y \right) = \bigotimes_{i=1}^I A_i^* p^i \otimes By$$

Let $\widehat{A}_i \in \mathcal{L}(\widehat{X}_i, \widehat{X}_i)$ and $\widehat{B} \in \mathcal{L}(\widehat{Y}, \widehat{Y})$. Then, it is easy to check that

$$\left(\bigotimes_{i=1}^I \widehat{A}_i^* \otimes \widehat{B} \right) \left(\bigotimes_{i=1}^I A_i^* \otimes B \right) = \bigotimes_{i=1}^I (\widehat{A}_i^* A_i^*) \otimes (\widehat{B}B)$$

One main reason for introducing the concepts of tensor products is given by the following

Theorem 4.1 *Let us consider finite dimensional vector spaces X_i , ($i = 1, \dots, I$), Y and Z , elements $x_i \in X_i$, a differentiable map $g : Y \rightarrow Z$, with which we associate the map*

$$\Psi : M \in \mathcal{L}_I \left(\prod_{i=1}^I X_i; Y \right) \rightarrow \Psi(M) := g(M(x_1, \dots, x_n)) \in X$$

The derivative of Ψ at $M \in \mathcal{L}_I \left(\prod_{i=1}^I X_i; Y \right)$ is given by

$$\Psi'(M) = x_1 \otimes \cdots \otimes x_I \otimes g'(M(x_1, \dots, x_I)) \in \mathcal{L}_I \left(\prod_{i=1}^I X_i^*; \mathcal{L}(Y, Z) \right)$$

Furthermore, if E is a differentiable functional from Z to \mathbf{R} , setting

$$H(M) := E(g(M(x_1, \dots, x_I)) - z)$$

the gradient of H at $M \in \mathcal{L}_I \left(\prod_{i=1}^I X_i; Y \right)$ is given by the tensor product

$$H'(M) = x_1 \otimes \cdots \otimes x_I \otimes g'(M(x_1, \dots, x_I))^* E'(g(M(x_1, \dots, x_I)) - z)$$

belonging to $\mathcal{L}_I \left(\prod_{i=1}^I X_i^*; Y^* \right)$, the entries of which are

$$x_{1,l_1} \cdots x_{I,l_I} \sum_{k=1}^r \frac{\partial}{\partial y_j} g_k(M(x_1, \dots, x_I)) \frac{\partial}{\partial z_k} E(g(M(x_1, \dots, x_I)) - z)$$

when $l_i = 1, \dots, n_i$ ($i = 1, \dots, I$) and $j = 1, \dots, m$.

We recall the particular case of maps defined on linear operators:

Corollary 4.2 *Let us consider three spaces X, Y and Z , an element $x \in X$, a differentiable map $g : Y \rightarrow Z$ and a differentiable functional E from Z to \mathbf{R} . Setting $H(W) := E(g(Wx) - z)$, the gradient of H at $W \in \mathcal{L}(X, Y)$ is given by*

$$H'(W) = x \otimes g'(Wx)^* E'(g(Wx) - z) \in \mathcal{L}(X^*, Y^*)$$

5 Learning Rules of Neural Networks with Gates

For simplicity, we consider here the learning problem of one pattern only.

5.1 One-Gate Neural Networks

In the case of *one-layer neural network with one gate* described by an *output function* $g : Y \rightarrow Y$, we define the map Φ by

$$\Phi(a_1, \dots, a_I, M) := g(M(a_1, \dots, a_I))$$

where the input $a_i \in X_i$ and the gating tensor $M \in \mathcal{L}_I \left(\prod_{i=1}^I X_i; Y \right)$ are given. Given an evaluation function E on the output space Y and a pattern (a_1, \dots, a_I, b) , we look for a gating tensor which minimizes the function

$$M \rightarrow H(M) := E(g(M(a_1, \dots, a_I)) - b)$$

Since the gradient of H is given by

$$H'(M) = a_1 \otimes \dots \otimes a_I \otimes g'(M(a_1, \dots, a_I))^* E'(g(M(a_1, \dots, a_I)) - b)$$

the gradient method can then be written

$$M^{n+1} - M^n = -\varepsilon_n a_1 \otimes \dots \otimes a_I \otimes g'(M^n(a_1, \dots, a_I))^* E'(g(M^n(a_1, \dots, a_I)) - b)$$

When the output function g_j depends only upon the signal arriving at neuron j , we obtain

$$\begin{cases} M_{l_1, \dots, l_I, j}^{n+1} - M_{l_1, \dots, l_I, j}^n \\ = -\varepsilon_n a_{1, l_1} \dots a_{I, l_I} \left(\frac{\partial g_j}{\partial y_j}(M^n(a_1, \dots, a_I)) \frac{\partial E}{\partial y_j} g_i((M^n(a_1, \dots, a_I))_j - b_j) \right) \end{cases}$$

It belongs to the class of *extended Hebbian reinforcement learning rules*: the synaptic weight from a the coalition of neurons $1, \dots, I$ to a neuron j should be *strengthened* whenever the connection is highly active, in proportion to the activities a_1, \dots, a_I of the pre-synaptic neurons and the activity $\frac{\partial g_j}{\partial y_j}(M^n(a_1, \dots, a_I)) \frac{\partial E}{\partial y_j} g_i((M^n(a_1, \dots, a_I))_j - b_j)$ of the post-synaptic neuron j .

5.2 One-Layer One-Gate Neural Networks

We can extend the computation of such learning algorithms to *multi-layer neural network with gates*. The use of tensor products allows us to differentiate maps the derivatives of which become quickly intractable.

For simplicity of notations, let us consider only a *two-layer neural network* described by

1. I input spaces $X_i := \mathbf{R}^{n_i}$ constituting the first layer,
2. an intermediate layer $U := \mathbf{R}^p$,
3. an output space $Y := \mathbf{R}^m$.

The first network is a neural network with gates defined by

$$y_1 = g_0(M(x_1, \dots, x_I))$$

where $M \in \mathcal{L}_I \left(\prod_{i=1}^I X_i; U \right)$ is a multilinear operator from the input space $\prod_{i=1}^I X_i$ to the intermediate layer U and $g_0 : U \rightarrow U$ is the propagation rule of the signal in the intermediate layer. The second network is a usual neural network defined by

$$y_2 = g_1(Wy_1)$$

where $W \in \mathcal{L}(U, Y)$ is a synaptic matrix and $g_1 : Y \mapsto Y$ is the propagation rule in the output space.

Given an evaluation function E on the output space Y and a pattern (a_1, \dots, a_I, b) , we look for a gating tensor which minimizes the function

$$(M, W) \rightarrow H(M, W) := E(g_1(Wg_0(M(a_1, \dots, a_I))) - b)$$

The “back-propagation” learning rule is nothing else than the gradient method applied to this function $(M, W) \mapsto H(M, W)$:

We set

$$\left\{ \begin{array}{l} y_1 := g_0(M(a_1, \dots, a_I)) \text{ (the intermediate signal)} \\ y_2 := g_1(Wy_1) \text{ (the output signal)} \\ q_0 := E'(y_2 - b) \text{ (the “gradient” of the error)} \\ q_1 := g_1'(Wy_1)^* q_0 \text{ (the first back-propagation of the “gradient”)} \\ q_1 := g_0'(M(a_1), \dots, a_I)^* W^* q_1 \text{ (the second back-propagation of the “gradient”)} \end{array} \right.$$

Therefore, one can prove that

$$\frac{\partial}{\partial W} H(M, W) = y_1 \otimes q_1 \text{ and } \frac{\partial}{\partial M} H(M, W) = a_1 \otimes \dots \otimes a_I \otimes q_2$$

(see Chapter 4 of Aubin, 1996). The gradient method provides the celebrated *back-propagation algorithm*: Starting with initial gating tensor and synaptic matrix (M^0, W^0) , we define inductively the synaptic matrix W^{n+1} and the gating tensor M^{n+1} from the previous synaptic matrix W^n and gating tensor M^n according to the following rule: We consider the signals $y_1^n := g_0(M^n(a_1, \dots, a_I))$ and $y_2^n = g_1(W^n y_1^n)$ and next, we compute the gradient $q_0^n := E'(y_2^n - b)$ of the error at the final layer, then we “back-propagate” it to $q_1^n := g_1'(W^n y_1^n)^* q_0^n$ and modify the synaptic matrix through the *reinforcement learning rule*

$$W^{n+1} - W^n = -\varepsilon_n y_1^n \otimes q_1^n$$

and, lastly, we “back-propagate” q_1^n to $q_2^n := g_0'(M^n(a_1, \dots, a_I))^* W^{n*} q_1^n$ and modify the gating tensor through the reinforcement learning rule

$$M^{n+1} - M^n = -\varepsilon_n a_1 \otimes \dots \otimes a_I \otimes q_2^n$$

5.3 The need for nonsmooth optimization

We may not always have the possibility to choose the evaluation function E and for instance, the opportunity to choose the simplest ones, as in the case of *least-square methods*

$$H_2(M) := \sum_{p \in P} \|g(M(a_1^p, \dots, a_I^p) - b^p)\|^2$$

for learning a teaching set $(a_1^p, \dots, a_I^p, b^p)$ of patterns. But, for a variety of reasons, we may prefer to minimize the function

$$H_\infty(M) := \max_{p \in P} \|g(M(a_1^p, \dots, a_I^p) - b^p)\|^2$$

which is no longer differentiable. However, when the criterion is nonsmooth, the gradient of the criterion can be replaced by a “*generalized gradient*”. In another instance — the learning problem of control systems — the problem is to regulate the system by finding the controls which govern the evolution of the state in order to obey prescribed *state (or viability) constraints*.

Such evolutions are called “viable”. Many control problems, in particular, the tracking of a trajectory, stabilization, etc., fall in this category. One can obtain viable evolutions by taking for evaluation function the function measuring the distance from a point to the constrained set, which is not necessarily differentiable. However, nonsmooth analysis, and in particular, convex analysis, provide ways to define *generalized gradients* of any function. In general, one defines in nonsmooth analysis the concept of the generalized gradient of H at M , which is generally a subset $\partial H(M)$. Generalized gradient (which boil down to *subdifferential* in the case of convex functions) are no longer elements, but subsets. They allow us to extend the gradient methods. This is not the right place to recall precisely how generalized gradients are rigorously defined: We refer for instance to Chapter 6 of (Aubin & Frankowska, 1990) for an introduction to nonsmooth analysis.

When E is locally Lipschitz and g is differentiable, we deduce that the generalized gradient of the function H is given by

$$\partial H(M) = a_1 \otimes \cdots \otimes a_I \otimes g'(M(a_1, \dots, a_I))^* \partial E(g(M(a_1, \dots, a_I)) - b)$$

since the generalized gradient of the function $\Psi : y \rightarrow \Psi(y) := E(g(y) - b)$ is equal to $g'(y)^* \partial E(g(y) - b)$. The gradient algorithm takes the form

$$M^{n+1} - M^n \in -\varepsilon_n a_1 \otimes \cdots \otimes a_I \otimes g'(M^n(a_1, \dots, a_I))^* \partial E(g(M^n(a_1, \dots, a_I)) - b)$$

The convergence of these algorithms holds true under convexity assumptions both in the smooth and nonsmooth cases. We refer to (Aubin, 1996) for further details on this topic in the case of synaptic matrices.

6 Learning without Forgetting

An algorithm which learns without forgetting is defined as follows for the class of neural networks with gates of the form

$$\Phi(x, M) := c(x) + G(x)M(\psi_1(x), \dots, \psi_I(x))$$

where $c : X \mapsto Y$ maps the input space X to the output space Y , ψ_i maps the input space to the input layers Z_i ($i = 1, \dots, n$), the gating tensor $M \in \mathcal{L}\left(\prod_{i=1}^I Z_i; U\right)$ gates the processed signals $\psi_i(x)$ to an output layer space U and where $G(x) \in \mathcal{L}(U, Y)$ maps the gated processed signals in the output layer to the output space. Such networks are affine with respect to the gating tensors.

Consider a finite teaching set of patterns $(a_p, b_p) \in X \times Y$. Hence we have to find a gating tensor M_P learning the training set in the sense that

$$c(a_p) + G(a_p)M_P(\psi_1(a_p), \dots, \psi_I(a_p)) = b_p \text{ for all } p = 1, \dots, P \quad (1)$$

Assuming that M_{P-1} has been obtained for learning the $P-1$ input-output pairs (a_p, b_p) ($1 \leq p \leq P-1$), we want to find a new synaptic matrix M_P which learns the whole new teaching set (a_p, b_p) ($1 \leq p \leq P$) according to the “heavy rule”

$$M_P \text{ minimizes } M \rightarrow \|M - M_{P-1}\|$$

under the constraints (??).

For that purpose, we need the concept of pseudo-inverse $C^\dagger \in \mathcal{L}(Y, X)$ of a matrix $C \in \mathcal{L}(X, Y)$. It maps any $y \in Y$ to the *closest solution with minimal norm* $\bar{x} = C^\dagger y$, i.e., the solution with minimal norm to the equation $Cx = \bar{y}$ where \bar{y} is the projection of y onto the image of C .

The pseudo-inverse coincides with the usual inverse when C is invertible, is involved in quadratic minimization problems under linear equality constraints and enjoys many properties (which are often used in statistical analysis). We refer for instance to (Aubin, 1996) for presentations of pseudo-inverses.

Since we define the gating tensor W_P as a solution to a quadratic minimization problem under linear equality constraints, we have to use the pseudo-inverse of tensor products. However, one can prove that *the pseudo-inverse of a tensor product is the tensor product of pseudo-inverses*: this is a very useful property of tensor products since pseudo-inverses of linear operators pop-up in many applications, statistics and data analysis, for instance. This is what we shall do for our problem:

We posit the assumptions

- i)* $G(x) \in \mathcal{L}(U, Y)$ is surjective for all $x \in X$
- ii)* the elements $\psi_i(a_p)$ are mutually orthonormal in Z_i

Then the “heavy algorithm” associates with the gating tensor M_{P-1} the new synaptic matrix M_P defined by formula:

$$M_P = M_{P-1} - \psi_1(a_P) \otimes \cdots \otimes \psi_I(a_P) \otimes G(a_P)^\dagger (\Phi(a_P, M_{P-1}) - b_P)$$

This is also a *reinforcement learning rule*, in the sense that the correction of the gating tensor M_{P-1} is obtained by adding to it a matrix $p_1 \otimes \cdots \otimes p_I \otimes u$ whose entries $p_{1,l_1} \cdots p_{I,l_I} u^j$ are proportional to the product of activities in the presynaptic and postsynaptic neurons. It uses only the former gating tensor W_{P-1} and the last pattern (a_P, b_P) for making the correction.

7 Discussion

The approach proposed here is in line with the ones assuming that learning is encoded in synaptic matrices, which are particular cases of tensors. The need to involve “multiplicative inputs” or $\Sigma - \Pi$ units has been felt for a long time by many neural network theoreticians. This presents the advantage of increasing the number of learning procedures of a given network, by using the flexibility and the plasticity offered by the multiplicative character of inputs, as the study of the parietal cortex suggests.

Furthermore, the parietal cortex links informations coming both from sensory (visual) and motor (somatic) inputs, which evolve permanently and independently. The network must combine these informations, and tensor products offer a convincing mathematical metaphor for doing so.

A third advantage of this type of architecture is that it allows us to take into account the constancy of the order of magnitude of the synapses per neurone (about 10^4) which is the same from rats to humans. The use of tensor products instead of synaptic matrices increases the learning capacity taking into account this biological constraint forbidding totally connected networks.

8 Perspectives

In classification problems, synaptic matrices and gating tensors which learn a given set of patterns are equilibria of a nonlinear problem, which are computed through gradient and Newton type algorithms (for which we refer to (Aubin, 1996)).

However, by looking only at static or asymptotic problems (mapping inputs onto outputs, or finding equilibria and more generally, attractors), the evolution mechanism involved in neural networks is neglected and the “real” time used to model the neural networks is often replaced by

artificial times. For instance, the *algorithmic time*, describing the iterations of a given algorithm, is often interpreted as a learning rule — as we did here — although it may not be related to the time involved in the modeling of the network.

By choosing the route used by most neural networks, one may bypass the basic question: *Why and How do synaptic matrices and/or gating tensors evolve?* and the basic answer: *To adapt to constraints through learning laws which are feedbacks* of the neural network regarded as a control system, associating at each instant with any signal a synaptic matrix and/or a gating tensor allowing the adaptation to viability constraints. Among the attempts to answer such questions, we refer to a dynamical point of view presented in (Aubin, 1996) and the papers of Nicolas Seube.

Further, recognition mechanisms involving coalitions of neurons (maybe corresponding to actual cortical areas, the number of which increases along phylogenesis) should be included in such dynamical networks.

This does and shall require modern mathematical techniques to handle some of such issues. If we accept that physics studies much simpler phenomena than the ones investigated by neurosciences, and that for this very purpose, they motivated and used a more and more complex mathematical apparatus, we have to accept also that neurosciences could require a new and dedicated mathematical arsenal which goes beyond what is presently available. Paradoxically, the very fact that the mathematical tools that I think useful for neurosciences are and have to be quite sophisticated impairs their acceptance by many specialists, and the gap menaces to widen.

Among the recent mathematical developments, *set-valued analysis*, which deals with the extension of the differential and integral calculus to set-valued maps which “pop-up” naturally in Artificial Intelligence (in the extent where “intelligence” is sometime grasped by the multivocality of the input-output maps or of their inverses), may become a “must” (see for instance (Aubin & Frankowska, 1990)). I mention also the use of differential inclusions and *viability theory* for attempting to model Darwinian evolution by encapsulating the concept of (contingent) chance by differential inclusions instead of stochastic differential equations and necessity by state (or “viability”) constraints to which at least one evolution starting from any initial state must comply (see for instance (Aubin, 1991)).

Contents

References

- [1] Andersen R.A., Essick G.K., Siegel R.M. (1985) *Encoding of spatial location by posterior parietal neurons*, Science 230, 456-458
- [2] Andersen R.A., Snyder L.H., Bradley D.C., Xing J. (1997) *Multimodal representation of space in the posterior parietal cortex and its use in planning movements*, Ann Rev Neurosci 20, 303-330
- [3] * Arbib M.A. (1987) **Brains, machines, and mathematics**, Springer-Verlag
- [4] * Aubin J.-P. (1991) **Viability Theory**, Birkhäuser
- [5] Aubin J.-P. (1995) *Learning as adaptive control of synaptic matrices*, in **The handbook of brain theory and neural networks**, Arbib M. Ed., Bradford Books and MIT Press, 1995
- [6] * Aubin J.-P. (1996) **Neural Networks and Qualitative Physics, A Viability Approach** Cambridge University Press
- [7] * Aubin J.-P. & Frankowska H. (1990) **Set-Valued Analysis**, Birkhäuser
- [8] Aubin J.-P. & Seube N. (1991) *Apprentissage adaptatif de lois de rétroaction de systèmes contrôlés par réseaux de neurones*, C. R. Acad. Sci., Paris, 314, 957-963
- [9] Burnod Y., Grandguillaume P., Otto I., Ferraina S., Johnson P.B., Caminiti R. (1992) *Visuo-motor transformations underlying arm movements toward visual targets: a neural network model of cerebral cortical operations*, J Neurosci 12, 1435-1453
- [10] Caminiti R., Johnson P.B., Galli C., Ferraina S., Burnod Y. (1991) *Making arm movements within different parts of space: the premotor and motor cortical representation of a coordinate system for reaching to visual targets*, J Neurosci 11, 1182-1197
- [11] Caminiti R., Johnson P.B., Ferraina S., Bianchi L. (1996) *The source of visual information to the primate frontal lobe: A novel role for the superior parietal lobule*, Cereb Cortex 6, 319-328
- [12] * McClelland J. L. , Rumelhart D. E. & (Eds.) (1986) **Parallel Distributed Processing (Vol.I)**. M.I.T. Press
- [13] Mountcastle V.B. (1997) *The columnar organization of the neocortex*, Brain 120, 701-722
- [14] Trotter Y., Celebrini S., Stricanne B., Thorpe S., Imbert M. (1992) *Modulation of neural stereoscopic processing in primate area V1 by the viewing distance*, Science 257, 1279-1281
- [15] Wise S.P., Boussaoud D., Johnson P.B., Caminiti R. (1997) *Premotor and Parietal Cortex: Corticocortical Connectivity and Combinatorial Computations*, Ann Rev Neurosci 20, 25-42
- [16] Zipser D., Andersen R.A. (1988) *A back-propagation programmed network that simulates response properties of a subset of posterior parietal neurons*, Nature 331, 679-684