

# Second Differentials in Arbitrary Feed-Forward Neural Networks\*

Technical report n° THOMSON-CSF/AIRSYS/RDTE-594/96

Fabrice ROSSI†  
 THOMSON-CSF/AIRSYS/RDTE  
 7/9, rue des Mathurins  
 92221 Bagneux Cedex  
 FRANCE

e-mail: [rossi@ceremade.dauphine.fr](mailto:rossi@ceremade.dauphine.fr)

WWW: <http://www.ceremade.dauphine.fr/~rossi/index.uk.html>

September 9, 1996

## Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>A general feed-forward neural network model</b>	<b>4</b>
2.1	A general neural network . . . . .	4
2.1.1	A formal neuron . . . . .	4
2.1.2	Ordered Oriented Graph . . . . .	4
2.1.3	General assumptions and notations . . . . .	5
2.1.4	The neural network . . . . .	6
2.2	Computing with the model . . . . .	6
2.3	Computing the differentials . . . . .	7
2.3.1	Additional definition . . . . .	7
2.3.2	The direct method . . . . .	8
2.3.3	“Free” neural networks . . . . .	8
2.3.4	Back-propagation . . . . .	10
2.3.5	Output functions . . . . .	10
2.4	Complexity . . . . .	11
2.4.1	General assumptions . . . . .	11
2.4.2	Direct method . . . . .	11
2.4.3	Back-propagation . . . . .	11
<b>3</b>	<b>Additional notations</b>	<b>12</b>
<b>4</b>	<b>Second order derivatives</b>	<b>12</b>
4.1	The direct method approach . . . . .	12
4.1.1	General case . . . . .	12
4.1.2	Particular cases . . . . .	13
4.1.3	Analysis . . . . .	14
4.2	The back-propagation approach . . . . .	14

\*Available at <http://apiacoa.org/publications/1996/thomson1996second.pdf>

†Up to date contact informations for Fabrice Rossi are available at <http://apiacoa.org/>

4.2.1	Local equations	14
4.2.2	Direct differentiation of $\frac{\partial \sigma^l}{\partial \sigma^j}$ : the hybrid method	15
4.2.3	Back-propagation based differentiation of $\frac{\partial \sigma^l}{\partial \sigma^j}$ : the pure back-propagation method	16
4.2.4	Symmetric formulae	18
<b>5</b>	<b>Error function</b>	<b>21</b>
5.1	Direct method	21
5.2	The hybrid method	22
5.3	Pure back-propagation	23
<b>6</b>	<b>Complexity</b>	<b>23</b>
6.1	The direct method	24
6.1.1	Equation part	24
6.1.2	Variable part	26
6.1.3	Differentiation order	27
6.1.4	Total cost	28
6.2	The hybrid method	29
6.2.1	Equation part	29
6.2.2	Variable part	34
6.2.3	Computation order and total cost	35
6.3	The pure back-propagation method	37
6.3.1	Equation cost	37
6.3.2	Computation order and total cost	39
<b>7</b>	<b>Complexity with an error function</b>	<b>40</b>
7.1	Direct method	40
7.2	Hybrid Method	41
7.2.1	Local formulae	41
7.2.2	Recursive formulae	41
7.2.3	Differentiation order	42
7.3	Pure Back-Propagation	42
<b>8</b>	<b>First order differentials</b>	<b>42</b>
8.1	Needed first order differentials	42
8.1.1	Direct method	43
8.1.2	Hybrid method	43
8.1.3	Pure back-propagation	43
8.2	Complexity	44
8.2.1	Direct method	44
8.2.2	Back-propagation	45
8.2.3	Comparison	46
<b>9</b>	<b>A practical analysis</b>	<b>46</b>
9.1	Scalar output	46
9.2	Totally connected layered architectures	47
9.2.1	General assumptions and preliminary results	47
9.2.2	The direct method	47
9.2.3	The hybrid method	50
9.2.4	The pure back-propagation method	53
9.2.5	Comparison	56
9.3	First order differentials	64
9.3.1	Scalar output	64
9.3.2	Totally connected layered architecture	64

## 1 Introduction

A huge amount of experimental works has proved that multi-layer perceptrons (MLP) are well suited for classification tasks, for which the goal is to separate vectorial inputs into several classes. It is also well known that MLPs are universal approximators (see [1, 5, 9]): given an arbitrary continuous mapping from a compact set (subset of  $\mathbb{R}^n$ ) into  $\mathbb{R}^p$ , and an arbitrary accuracy, there exists a two layer perceptron that approximates the given function with the given accuracy. Two main problems are the design of such a MLP with a minimum number of neurons, and the training of a given MLP so as to minimize the approximation error with respect to a given function. Theoretical results define some methods to answer the first question, but they build still too huge neural networks. Experimental results seem to prove that the second problem is more difficult than the classification problem, especially to obtain the *best* approximation (it is due to the well known problem of local minima).

The MLP model has been modified in many different ways to simplify both designing and training processes. In a MLP, a neuron collects outputs from the previous layer, it multiplies each output by a synaptic weight, it sums the resulting values and then it applies a transfer function to the sum. The simplest modification idea is to change the transfer function, using a sinus [4] or a gaussian function [10], for instance, instead of the standard sigmoid function. It is also possible to modify the preprocessing computation. A vectorial threshold may be applied to the output of the previous layer before the weighting process ([16]). The distance between the output of the previous layer and a prototype vector may be computed before using a transfer function. This idea was developed to use radial basis functions (RBF) as transfer functions ([11, 12, 14]). A non linear process (such as a quadratic form, see [15]) may be used instead of the linear process (i.e. the weighted sum). Some authors have also proposed to use multidimensional wavelets as neurons (see for instance [18]). This method strongly modifies the neuron structure, when compared to the standard MLP neuron.

All these models are universal approximators. Choosing the best one is difficult, since there is no theoretical result to compare their performance. Indeed a particular model might be well suited for a particular application and unadapted to another one. Despite their differences, these models share a common principle: they use an acyclic graph of simple units to compute a complex parametric vectorial function. This general point of view can be translated into a mathematical model which generalizes the notion of feed-forward neural network. This method was proposed by Léon Bottou and Patrick Gallinari in [2]. Their key idea was to allow the cooperation of modules of different kinds. In previous papers ([7, 6]), we extended the proposed model to describe a general mathematical model for feed-forward neural networks. This model is able to handle many models, among which all the models derived from the standard MLP model.

Following Bottou and Gallinari, we have introduced a generalized back-propagation algorithm which allowed us to compute the gradient of the error made by a neural network in an efficient way. We have extended the algorithm proposed by Bottou and Gallinari to compute the differentials of the function computed by the neural network too. It allows us to view a neural network itself as a neuron. The differentials of the function computed by the neural network can also be computed with the help of a direct algorithm. In order to compare both methods, we have computed the theoretical amount of operations needed by both algorithms. Our work shows that these amounts can not be directly compared in the general case. A study of some particular cases shows that the back-propagation is not always the best algorithm for arbitrary feed-forward neural networks.

In [3], W. L. Buntine and A. S. Weigend have presented general equations for second order derivatives computation in feed-forward neural networks, together with a survey of the use of such derivatives: second order training such as in [12, 17] or post-training analysis (such as Optimal Brain Surgeon, in [8]). In this report, we extend their work and introduce three different algorithms which can be used to compute the Hessian of the error made by an arbitrary feed-forward neural network as described in our previous

paper: a direct algorithm, a pure back-propagation based one and a hybrid algorithm.

In order to compare the obtained algorithms, we also compute their theoretical time complexities. We show that for MLP, the pure back-propagation algorithm (which extends the algorithm proposed by Buntine and Weigend) is faster than the direct algorithm. We show also that the complexity of the hybrid algorithm, which combines a direct part with back-propagation formulae, cannot be computed in a simple way because of the influence of the so called computation strategy.

Finally, we study the complexity of the computation of the first order differentials needed by the different algorithms. We show that in the case of a MLP, the back-propagation is not always a good choice for computing these first order differentials, which can be more efficiently computed by a direct algorithm. This a quite important result which gives precisions about suggestions of other authors [3, 13].

The report is organized as follows: section 2 recalls briefly definitions and results introduced in [7, 6]. Section 3 introduces new notations needed in this report. Section 4 describes the three different second order differential calculation algorithms and proves their correctness. Section 5 extends these algorithms in order to handle efficiently the special case of the computation of the Hessian of the error made by a network. Section 6 computes the time complexities of the different algorithms and section 7 modifies the complexities for the error Hessian computation case. Then, section 8 studies which first order differentials are needed by the different algorithms and establish the time complexity of their computation. Finally, section 9 studies the obtained complexities in the special case of MLP.

## 2 A general feed-forward neural network model

This section recalls briefly results from our previous work. This work is precisely explained in the **Neurocolt report NC-TR-95-041** entitled "A General Feed-Forward Neural Network Model", written by Cédric Gégout, Bernard Girau and Fabrice Rossi ([6]). It can be down-loaded via the World Wide Web at URL:

<http://apiacoa.org/publications/1995/neurocolt1995.pdf>

### 2.1 A general neural network

#### 2.1.1 A formal neuron

**Definition 1** Let  $n$  and  $k$  be positive integers and let  $I^1, \dots, I^n, W$  and  $O$  be vectorial spaces on  $\mathbb{R}$  of finite dimensions.

A  $(\mathbf{C}^k)$  **n-input neuron** is a  $(\mathbf{C}^k)$  function from  $I^1 \times \dots \times I^n \times W$  into  $O$ .

$I^j$  is the  $j$ -th input space of  $N$  and  $W$  is its parameter space.

If  $N$  is a  $C^1$   $n$ -input neuron,  $\frac{\partial N}{\partial x^j}$  is the partial differential of  $N$  with respect to its  $j$ -th input, considering all other inputs to be constant. When  $N$  has only one input ( $n = 1$ ), we simplify this notation and use  $\frac{\partial N}{\partial x}$ .  $\frac{\partial N}{\partial w}$  is the partial differential of  $N$  with respect to its  $n + 1$ -th input (the parameter vector belonging to  $W$ ).

As  $N$  is a vectorial function, we can consider its coordinate functions, called  $N_1, N_2, \dots, N_n$ .  $\frac{\partial N_i}{\partial x^k}$  is the partial differential of  $N_i$  with respect to its  $k$ -th input variable and  $\frac{\partial N_i}{\partial w}$  stands for the differential with respect to the parameter vector. Finally, as the input and parameter variables are vectors, it is possible to compute the differential with respect to each coordinate of these vectors. For instance,  $\frac{\partial N_i}{\partial x_j^k}$  is the partial differential of  $N_i$  with respect to the  $j$ -th coordinate of its  $k$ -th input vector.

#### 2.1.2 Ordered Oriented Graph

**Definition 2** An oriented graph  $\mathcal{G}$  is a pair  $(\mathcal{N}, \mathcal{F})$  of sets which fulfills the following conditions:

- $\mathcal{N}$  is a finite set of nodes ;
- $\mathcal{F}$  is a subset of  $\mathcal{N}^2$ , and  $e = (x, y) \in \mathcal{F}$  is an **edge** of the graph if there is a connection from  $x$  to  $y$ .

**Definition 3** Let  $\mathcal{G} = (\mathcal{N}, \mathcal{F})$  be an oriented graph and let  $n \in \mathcal{N}$  be a node of  $\mathcal{G}$ .

- $P(n) = \{p \in \mathcal{N} \mid (p, n) \in \mathcal{F}\}$  is the set of the nodes that have a connection to  $n$ . These nodes are the **predecessors** of  $n$ .
- $S(n) = \{p \in \mathcal{N} \mid (n, p) \in \mathcal{F}\}$  is the set of the nodes that receive a connection from  $n$ . These nodes are the **successors** of  $n$ .

**Definition 4** An ordered oriented graph  $\mathcal{G}$  is a triple  $(\mathcal{N}, \mathcal{F}, <)$ , which fulfills the following conditions:

- $\mathcal{N}$  is a **finite totally ordered** set of nodes. It may be considered as a sequence of nodes  $N^1, N^2, \dots$  ;
- $\mathcal{F}$  is a subset of  $\mathcal{N}^2$ , and  $e = (x, y) \in \mathcal{F}$  is an **edge** of the graph if there is a connection from  $x$  to  $y$  ;
- $<$  is a function from  $\mathcal{N}$  into  $\mathcal{P}(\mathcal{N}^2)$ . It associates to each node  $N^i$  in  $\mathcal{N}$  a total order  $<_{N^i}$  on its predecessor set,  $P(N^i)$ . This set is therefore a sequence and it can be referred to  $P(N^i)^k$  for its  $k$ -th element.

### 2.1.3 General assumptions and notations

We introduce here some general assumptions in order to simplify the remainder of the report:

- $\mathcal{G} = (\mathcal{N}, \mathcal{F}, <)$  is an ordered graph with exactly  $n$  nodes,
- $N^1, \dots, N^n$  is the sequence of the graph nodes,
- $P(N^k) = P(k)$  is the set of the predecessors of  $N^k$ ,
- $S(N^k) = S(k)$  is the set of the successors of  $N^k$ ,
- Node  $N^k$  has exactly  $p^k$  predecessors and  $s^k$  successors,
- $P(k)^1, \dots, P(k)^{p^k}$  is the sequence of the predecessors of  $N^k$ ,
- $S(k)^1, \dots, S(k)^{s^k}$  is the sequence of the successors of  $N^k$ .

The notion of predecessor may be generalized for an arbitrary node  $N$ :

- $P^0(N) = \{N\}$  ;
- $P^k(N) = \{M \in \mathcal{N} \mid \exists Q \in P^{k-1}(N) \text{ so that } (M, Q) \in \mathcal{F}\}$  (i.e.  $P^1(N) = P(N)$ ) ;
- $P^+(N) = \bigcup_{k=1}^{\infty} P^k(N)$  ;
- $P^*(N) = P^0(N) \cup P^+(N)$ .

Similar sets can be defined in order to generalize the notion of successor.

- $S^0(N) = \{N\}$  ;
- $S^k(N) = \{M \in \mathcal{N} \mid \exists Q \in S^{k-1}(N) \text{ so that } (Q, M) \in \mathcal{F}\}$  ;
- $S^+(N) = \bigcup_{k=1}^{\infty} S^k(N)$  ;
- $S^*(N) = S^0(N) \cup S^+(N)$ .

### 2.1.4 The neural network

**Definition 5** A feed-forward neural network is an ordered oriented graph  $\mathcal{G} = (\mathcal{N}, \mathcal{E}, <)$  which fulfills the following conditions:

1. The graph has **no cycle**.
2. The elements of  $\mathcal{N}$  are  **$k$ -input neurons** ( $k$  depends on the neuron). The output space of  $N^k$  is  $O^k$  and its parameter space is  $W^k$ .
3. If  $p^k > 0$  then neuron  $N^k$  is a  **$p^k$ -input neuron** (it means that if a neuron has got predecessors, it has exactly one input for each of its predecessors in the graph). The input spaces of  $N^k$  are  $I^{k,1}, \dots, I^{k,p^k}$ .
4. If  $p^k = 0$  then neuron  $N^k$  is a **1-input neuron** with input space  $I^k$ .
5. If  $p^k > 0$  then the following condition holds for each input  $i$  of the neuron  $N^k$ :  $\dim I^{k,i} = \dim O^{P(k)^i}$ .

In the remainder of the paper, we will identify each neuron with its rank in the node set. For instance, let us assume that  $P(N^3) = \{N^2, N^1\}$ , i.e.,  $P(3)_1 = P(N^3)_1 = N^2$  and  $P(3)_2 = N^1$ . To study the output space of the first predecessor of  $N^3$ , we should write  $O^2$ . But from a strict mathematical point of view,  $O^{P(k)^l}$  has no meaning in the general case. Therefore a function *rank* from  $\mathcal{N}$  into  $\{1, \dots, n\}$  must be defined. This function gives the rank of a node in  $\mathcal{N}$ . Nevertheless, writing  $O^{\text{rank}(P(k)^l)}$  is rather cumbersome, so that we will omit the *rank* function in the remainder of the paper.

We introduce some additional definitions:

- $In$  is the **subset of  $\mathcal{N}$  which elements have no predecessor**, i.e.,  $In = \{N \in \mathcal{N} \mid P(N) = \emptyset\}$ .  $In$  has *in* elements. It is a totally ordered set (order induced by the order on  $\mathcal{N}$ ) and  $In^1, \dots, In^{in}$  is the ordered sequence of its elements. The elements of  $In$  are connected to the “outside” by means of their inputs.
- When  $N^l \in In$ ,  $In(l)$  is the rank of  $N^l$  in  $In$ , i.e.  $N^l = In^{In(l)}$ .
- An **inner node** is a node belonging to  $\mathcal{N} \setminus In$ .
- $Out$  is the **subset of  $\mathcal{N}$  which elements have no successor**, i.e.,  $Out = \{N \in \mathcal{N} \mid S(N) = \emptyset\}$ .  $Out$  has *out* elements and is totally ordered.  $Out^1, \dots, Out^{out}$  is the ordered sequence of its elements. The elements of  $Out$  are connected to the “outside” by means of their outputs.
- When  $N^l \in Out$ ,  $Out(l)$  is the rank of  $N^l$  in  $Out$ , i.e.  $N^l = Out^{Out(l)}$ .
- The vectorial space  $I = \prod_{k=1}^{in} I^{In^k}$  is the **input space** of the neural network.  $I$  is the product of the input spaces of the neurons that belong to  $In$ .
- The vectorial space  $O = \prod_{k=1}^{out} O^{Out^k}$  is the **output space** of the neural network.  $O$  is the product of the output spaces of the neurons that belong to  $Out$ .
- The vectorial space  $W = \prod_{k=1}^n W^k$  is the **parameter space** of the neural network.  $W$  is the product of the parameter spaces of the network neurons.

## 2.2 Computing with the model

**Property 1** Let  $\mathcal{G} = (\mathcal{N}, \mathcal{F}, <)$  be a feed-forward neural network. There exists an unique positive integer  $l(\mathcal{G})$  and an unique partition of the node set  $\mathcal{N}$  defined by the  $l$  subsets  $L_1, \dots, L_{l(\mathcal{G})}$ , which satisfy the following conditions:

1.  $L_1 = In$  ;

$$2. \forall k > 1, L_k = \left\{ N \in \mathcal{N} \mid P(N) \subset \bigcup_{j=1}^{k-1} L_j \text{ and } \exists N' \in P(N), N' \in L_{k-1} \right\}.$$

The  $L_i$  are the **layers** of the neural network and  $l(\mathcal{G})$  is the **number of layers** of the neural network.

We can now define the output of the individual nodes of the network:

**Definition 6** Let  $\mathcal{G} = (\mathcal{N}, \mathcal{F}, <)$  be a feed-forward neural network. Let  $x = (x^1, \dots, x^{in}) \in I$  be an input vector and let  $w = (w^1, \dots, w^n) \in W$  be a parameter vector. For each  $l, 1 \leq l \leq n$ ,  $o^l(x, w)$  is defined with the help of the following recursive construction:

- For  $N^l \in L_1 = In$ ,  $N^l = In_k$ . Then:

$$o^l(x, w) = N^l(x^k, w^l) \quad (1)$$

- For  $N^l \in L_k$ , with  $k > 1$ . Then:

$$o^l(x, w) = N^l \left( o^{P(l)^1}(x, w), \dots, o^{P(l)^{p^l}}(x, w), w^l \right) \quad (2)$$

Finally,  $G(x, w)$  is obtained by:

$$G(x, w) = (o^{Out_1}(x, w), \dots, o^{Out_{out}}(x, w)) \quad (3)$$

This definition allows to consider a neural network itself as a neuron. We can also define the generalized input of a neuron, which is quite similar to  $o^l$ :

**Definition 7** Let  $\mathcal{G} = (\mathcal{N}, \mathcal{F}, <)$  be a feed-forward neural network. Let  $x = (x^1, \dots, x^{in}) \in I$  be an input vector and let  $w = (w^1, \dots, w^n) \in W$  be a parameter vector. For each  $l, 1 \leq l \leq n$ ,  $i^l(x, w)$  is defined with the help of the following recursive construction:

- For  $N^l \in L_1 = In$ ,  $N^l = In_k$ . Then:

$$i^l(x, w) = (x^k, w^l) \quad (4)$$

- For  $N^l \in L_k$ , with  $k > 1$ . Then:

$$i^l(x, w) = \left( o^{P(l)^1}(x, w), \dots, o^{P(l)^{p^l}}(x, w), w^l \right) \quad (5)$$

We have therefore  $o^l(x, w) = N^l(i^l(x, w))$ .

## 2.3 Computing the differentials

### 2.3.1 Additional definition

To obtain the rank of a node  $N^k$  in the predecessor list of its successor  $N^l$ , the following definition introduces a new function.

**Definition 8** Let  $\mathcal{G} = (\mathcal{N}, \mathcal{F}, <)$  be an ordered oriented graph. Let  $N^k$  be a node of  $\mathcal{G}$  and  $N^l$  be a successor of this node. We call  $r(k, l)$  the rank of  $N^k$  in the predecessor set of  $N^l$ , i.e.  $N^k = P(l)^{r(k, l)}$ .

The following definition generalizes the output of the neural network:

**Definition 9** Let  $\mathcal{G} = (\mathcal{N}, \mathcal{F}, <)$  be a feed-forward neural network.  $O$  is by definition the output space of the network and is equal to the product of the output spaces of output nodes. Let  $H$  be the product of the output spaces of all nodes of the network, i.e.,  $H = \prod_{i=1}^n O^i$ . We define the output function,  $h(x, w)$  from  $I \times P$  into  $H$ , with:

$$h(x, w) = (o^1(x, w), o^2(x, w), \dots, o^n(x, w)) \quad (6)$$

We introduce finally some special linear functions:

**Definition 10** Let  $A$  and  $B$  be two vectorial spaces. Then:

- $0_{A,B}$  is the null function from  $A$  to  $B$  which maps every vector  $x$  in  $A$  to  $0_B$  the null vector of  $B$ .
- $Id_A$  is the identity function of  $A$  which maps every vector  $x$  to itself.

### 2.3.2 The direct method

**Theorem 1** Let  $\mathcal{G} = (\mathcal{N}, \mathcal{E}, <)$  be a feed-forward neural network. Let  $N^l$  be an arbitrary node. Let  $x$  and  $w$  be arbitrary input and parameter vectors. Then:

- if  $N^l = In^k$ :

- if  $j \neq k$ :

$$\frac{\partial o^l}{\partial x^j}(x, w) = 0 \quad (7)$$

- if  $j \neq l$ :

$$\frac{\partial o^l}{\partial w^j}(x, w) = 0 \quad (8)$$

- and:

$$\frac{\partial o^l}{\partial x^k}(x, w) = \frac{\partial N^l}{\partial x}(x^k, w^l) \quad (9)$$

$$\frac{\partial o^l}{\partial w^l}(x, w) = \frac{\partial N^l}{\partial w}(x^k, w^l) \quad (10)$$

- if  $N^l \notin In$ :

- for all  $j$ ,  $1 \leq j \leq in$ :

$$\frac{\partial o^l}{\partial x^j}(x, w) = \sum_{N^k \in P(l) \cap S^*(In^j)} \frac{\partial N^l}{\partial x^{r(k,l)}} \left( o^{P(l)^1}(x, w), \dots, o^{P(l)^{p^l}}(x, w), w^l \right) \frac{\partial o^k}{\partial x^j}(x, w) \quad (11)$$

- if  $j \neq l$ :

$$\frac{\partial o^l}{\partial w^j}(x, w) = \sum_{N^k \in P(l) \cap S^*(j)} \frac{\partial N^l}{\partial x^{r(k,l)}} \left( o^{P(l)^1}(x, w), \dots, o^{P(l)^{p^l}}(x, w), w^l \right) \frac{\partial o^{P(l)^k}}{\partial w^j}(x, w) \quad (12)$$

- and:

$$\frac{\partial o^l}{\partial w^l}(x, w) = \frac{\partial N^l}{\partial w}(x^k, w^l) \quad (13)$$

### 2.3.3 “Free” neural networks

**Definition 11** Let  $\mathcal{G} = (\mathcal{N}, \mathcal{F}, <)$  be a feed-forward neural network. Let  $x = (x^1, \dots, x^{in}) \in I$  be an input vector and let  $w = (w^1, \dots, w^n) \in W$  be a parameter vector. Let  $N^k$  be an arbitrary node. We define  $o^{k \rightarrow k}(x, w, f^k) = f^k$ . For each  $l$ ,  $1 \leq l \leq n$ , and  $l \neq k$ ,  $o^{l \rightarrow k}(x, w, f^k)$  is defined with the help of the following recursive construction:

- If  $N^l \in L_1 = In$ , then  $N^l = In^q$  and:

$$o^{l \rightarrow k}(x, w, f^k) = N^l(x^q, w^l) \quad (14)$$

- If  $N^l \in L_k$ , with  $k > 1$  then:

$$o^{l \rightarrow k}(x, w, f^k) = N^l \left( o^{P(l)^1 \rightarrow k}(x, w, f^k), \dots, o^{P(l)^{p^l} \rightarrow k}(x, w, f^k), w^l \right) \quad (15)$$

Let  $G^k(x, w, f^k)$  be:

$$G^k(x, w, f^k) = \left( o^{Out^1 \rightarrow k}(x, w, f^k), \dots, o^{Out^{out} \rightarrow k}(x, w, f^k) \right) \quad (16)$$

We introduce also the free generalized input as (even for  $l = k$ ) :

- If  $N^l \in L_1 = In$ , then  $N^l = In^q$  and:

$$i^{l \rightarrow k}(x, w, f^k) = (x^q, w^l) \quad (17)$$

- If  $N^l \in L_k$ , with  $k > 1$  then:

$$i^{l \rightarrow k}(x, w, f^k) = \left( o^{P(l)1 \rightarrow k}(x, w, f^k), \dots, o^{P(l)P^l \rightarrow k}(x, w, f^k), w^l \right) \quad (18)$$

For  $l \neq k$ , we have :

$$o^{l \rightarrow k}(x, w, f^k) = N^l (i^{l \rightarrow k}(x, w, f^k)) \quad (19)$$

And finally, we introduce the free generalized output, a function from  $I \times P \times O^l$  into  $H$ , defined by :

$$h^{-l}(x, w, f^l) = (o^{1 \rightarrow l}(x, w, f^l), \dots, o^{n \rightarrow l}(x, w, f^l)), \quad (20)$$

which implies  $h^{-l}(x, w, o^l(x, w)) = h(x, w)$ .

We have some important properties:

**Property 2** Let  $\mathcal{G} = (\mathcal{N}, \mathcal{F}, <)$  be a feed-forward neural network. Let  $N^k$  be a network node. The following conditions hold for an arbitrary input vector  $x$ , an arbitrary parameter vector  $w$ , and an arbitrary free output vector  $f^k$ :

$$o^{l \rightarrow k}(x, w, o^k(x, w)) = o^l(x, w) \quad (21)$$

$$\forall N^l \notin S^*(k), o^{l \rightarrow k}(x, w, f^k) = o^l(x, w) \quad (22)$$

**Property 3** Let  $\mathcal{G} = (\mathcal{N}, \mathcal{E}, <)$  be a feed-forward neural network. Let  $N^l$  and  $N^k$  be two arbitrary nodes. Let  $x$ ,  $w$  and  $f^k$  be arbitrary input, parameter and free output vectors. Then:

$$\frac{\partial o^{l \rightarrow k}}{\partial w^k}(x, w, f^k) = 0_{W^k, O^l} \quad (23)$$

Intuitively, this equation implies that the output of the network depends on a parameter vector  $w^k$  only through the output of the corresponding neuron  $N^k$ .

**Property 4** Let  $\mathcal{G} = (\mathcal{N}, \mathcal{F}, <)$  be a feed-forward neural network. Let  $N^l$  and  $N^k$  be two arbitrary nodes. Let  $x$  and  $w$  be arbitrary input and parameter vectors. Then:

$$\frac{\partial o^l}{\partial w^k}(x, w) = \frac{\partial o^{l \rightarrow k}}{\partial o^k}(x, w, o^k(x, w)) \frac{\partial N^k}{\partial w}(i^k(x, w)) \quad (24)$$

**Property 5** Let  $\mathcal{G} = (\mathcal{N}, \mathcal{F}, <)$  be a feed-forward neural network. For each input node  $N^k = In^j$ , for each node  $N^l$  and for arbitrary input, parameter and local output vectors,  $x$ ,  $w$  and  $f^k$ , the following formula holds:

$$\frac{\partial o^{l \rightarrow k}}{\partial x^j}(x, w, f^k) = 0_{I^j, O^l} \quad (25)$$

Intuitively, it implies that the outputs of the nodes depend on an input vector  $x^j$  only through the output of the corresponding neuron  $N^k = In^j$ .

**Property 6** Let  $\mathcal{G} = (\mathcal{N}, \mathcal{F}, <)$  be a feed-forward neural network. For each input node  $N^k = In^j$ , for each node  $N^l$  and for arbitrary input and parameter vectors  $x$  and  $w$ , the following formula holds:

$$\frac{\partial o^l}{\partial x^j}(x, w) = \frac{\partial o^{l \rightarrow k}}{\partial o^k}(x, w, o^k(x, w)) \frac{\partial N^k}{\partial x}(x^j, w^k) \quad (26)$$

### 2.3.4 Back-propagation

**Theorem 2** Let  $\mathcal{G} = (\mathcal{N}, \mathcal{F}, <)$  be a feed-forward neural network. Let  $x$  be an arbitrary input vector. Let  $w$  be an arbitrary parameter vector and let  $N^l$  and  $N^k$  be arbitrary nodes. Then:

- If  $N^k = N^l$ :

$$\frac{\partial o^{l \rightarrow k}}{\partial o^k}(x, w, o^k(x, w)) = Id_{O^k} \quad (27)$$

- If  $N^k \notin P^*(N^l)$ :

$$\frac{\partial o^{l \rightarrow k}}{\partial o^k}(x, w, o^k(x, w)) = 0_{O^k, O^l}, \quad (28)$$

- If  $N^k \in P^+(N^l)$ :

$$\frac{\partial o^{l \rightarrow k}}{\partial o^k}(x, w, o^k(x, w)) = \sum_{N^j \in S(k) \cap P^*(l)} \frac{\partial o^{l \rightarrow j}}{\partial o^j}(x, w, o^j(x, w)) \frac{\partial N^j}{\partial x^{r(k,j)}}(i^j(x, w)) \quad (29)$$

### 2.3.5 Output functions

Computing the differential of a neural network function is often useful to train this network to perform a given task. A supervised learning method aims at decreasing an error, which is indeed a distance between a desired output and the actual output of the neural network. The error computation is modeled by an output function  $F$  which maps the vectorial output of the neural network to another output (the most common case is a real output). The following theorem allows to use an modified back-propagation to handle efficiently this case.

**Theorem 3** Let  $\mathcal{G} = (\mathcal{N}, \mathcal{F}, <)$  be a differentiable feed-forward neural network. Let  $F$  be a differentiable vectorial function of vectorial variables and let  $S^1, \dots, S^{out}$  be its input spaces, with  $S^k \simeq O^{Out^k}$ . A composed function  $F_G(x, w)$  can be defined as:

$$F_G(x, w) = F(o^{Out^1}(x, w), \dots, o^{Out^{out}}(x, w)) \quad (30)$$

$F$  is an output function for the network. Let  $F_G^k(x, w, f^k)$  be defined in a similar way as  $G^k$  (i.e., with a “free” output for node  $N^k$ ).

If  $\frac{\partial F}{\partial x^k}$  stands for the partial differential of  $F$  with respect to its  $k$ -th variable, then:

$$\frac{\partial F_G}{\partial w^k}(x, w) = \frac{\partial F_G^k}{\partial o^k}(x, w, o^k(x, w)) \frac{\partial N^k}{\partial w}(i^k(x, w)) \quad (31)$$

$$\frac{\partial F_G}{\partial x^j}(x, w) = \frac{\partial F_G^k}{\partial o^k}(x, w, o^k(x, w)) \frac{\partial N^k}{\partial x}(x^j, w^k), \text{ for } N^k = In^j \text{ an input node,} \quad (32)$$

with,

- for  $N^k = Out^j$ , an output node:

$$\frac{\partial F_G^k}{\partial o^k}(x, w, o^k(x, w)) = \frac{\partial F}{\partial x^j}(o^{Out^1}(x, w), \dots, o^{Out^{out}}(x, w)), \quad (33)$$

- for  $N^k \notin Out$ :

$$\frac{\partial F_G^k}{\partial o^k}(x, w, o^k(x, w)) = \sum_{N^j \in S(k)} \frac{\partial F_G^j}{\partial o^j}(x, w, o^j(x, w)) \frac{\partial N^j}{\partial x^{r(k,j)}}(i^j(x, w)) \quad (34)$$

## 2.4 Complexity

### 2.4.1 General assumptions

Both algorithms need to know the input, the output and the differentials of each node. Therefore, the complexity will include only the cost of the algebraic operations required by both methods. Moreover, we make the following assumptions:

- the main computation load is mostly due to the numerical operations needed for the algebraic operations, i.e., floating point number additions and multiplications. We will assume that the time needed to perform such an operation is 1 (experiments on a Sparc Station 2 show that the time to perform these operations is approximately the same for both addition and multiplication). This is therefore the unit of our formulae.
- the time needed to multiply a  $(i, j)$ -matrix and a  $(j, k)$ -matrix is approximatively  $ik(2j - 1)$  ;
- the time needed to sum two  $(i, j)$ -matrices is approximatively  $ij$ .

Let  $|\cdot|$  be the function which maps a vectorial space to its dimension (for instance,  $|O^l|$  is the dimension of the output space of node  $N^l$ ). The same notation will be used for the number of elements of a finite set (for instance,  $|\mathcal{N}| = n$ ).

### 2.4.2 Direct method

**Theorem 4** *Let  $\mathcal{G} = (\mathcal{N}, \mathcal{F}, <)$  be a differentiable feed-forward neural network. With the direct algorithm, computing the differential of  $G$  with respect to its parameter vector needs a time equal to:*

$$C_d(G)_w = \sum_{N^j \notin In} |O^j| \sum_{N^l \in P^+(j)} |W^l| \left( 2 \sum_{N^k \in P(j) \cap S^*(l)} |O^k| - 1 \right) \quad (35)$$

**Corollary 1** *Let  $\mathcal{G} = (\mathcal{N}, \mathcal{F}, <)$  be a differentiable feed-forward neural network. Let  $F$  be an output function for  $\mathcal{G}$ , with  $O^F$  as output space. With the direct algorithm, computing the differential of  $F_G$  with respect to its parameter vector requires algebraic operations which total cost is:*

$$C_d(F_G)_w = C_d(G)_w + |O^F| \left( \sum_{N^k \in \mathcal{N}} |W^k| \left( 2 \sum_{N^l \in Out \cap S^*(k)} |O^l| - 1 \right) \right) \quad (36)$$

### 2.4.3 Back-propagation

**Theorem 5** *Let  $\mathcal{G} = (\mathcal{N}, \mathcal{F}, <)$  be a feed-forward neural network. With the back-propagation algorithm, computing the differential of  $G$  with respect to its parameter vector needs a time equal to:*

$$C_{bp}(G)_w = \sum_{N^k \in Out} |O^k| \left( |I^k| + \sum_{N^l \in P^+(k)} \left( |W^l| (2|O^l| - 1) + |O^l| \left( 2 \sum_{N^j \in S(l) \cap P^+(k)} |O^j| - 1 \right) \right) \right) \quad (37)$$

**Theorem 6** *Let  $\mathcal{G} = (\mathcal{N}, \mathcal{F}, <)$  be a differentiable feed-forward neural network. Let  $F$  be an output function for  $\mathcal{G}$ , with  $O^F$  as output space. With the back-propagation algorithm, computing the differential of  $F_G$  with respect to its parameter vector requires algebraic operations which total cost is:*

$$C_{bp}(F_G)_w = |O^F| \left( \sum_{N^l} |W^l| (2|O^l| - 1) + \sum_{N^l \notin Out} |O^l| \left( 2 \sum_{N^j \in S(l)} |O^j| - 1 \right) \right) \quad (38)$$

### 3 Additional notations

Let  $N^l$  be a  $C^2$  neuron with input spaces  $I^{l,1}, \dots, I^{l,p^l}$ , parameter space  $W$  and output space  $O^l$ :

- we assume that  $O^l$  dimension is  $n^l$ .
- Let  $i, k, q$  and  $r$  be four integers with  $i \in [1, n^l]$ ,  $k \in [1, p^l]$ ,  $q \in [1, n^{\text{rank}(P^{(l)k})}]$  and  $r \in [1, p^l]$ . We already know that  $\frac{\partial N^l}{\partial x_q^k}$  is the partial differential of  $N^l$  with respect to the  $q$ -th coordinate of its  $k$ -th input variable. This is a function from  $I^{l,1} \times \dots \times I^{l,p^l} \times W$  to  $L(\mathbb{R}, O^l)$ , the space of linear functions from  $\mathbb{R}$  to  $O^l$  which can be identified to  $O^l$  itself. Therefore,  $\frac{\partial N^l}{\partial x_q^k}$  can be considered as a neuron and we will call  $\frac{\partial^2 N^l}{\partial x_q^k \partial x^r}$  its partial differential with respect to its  $r$ -th input variable. Of course,  $\frac{\partial^2 N^l}{\partial x_q^k \partial w}$  will be its partial differential with respect to its parameter vector.

We also know that  $\frac{\partial N_i^l}{\partial x_q^k}$  can be used. Once again this function can be considered as a neuron from  $I^{l,1} \times \dots \times I^{l,p^l} \times W$  to  $\mathbb{R}$  and we will therefore use the partial derivatives  $\frac{\partial^2 N_i^l}{\partial x_q^k \partial x^r}$  and  $\frac{\partial^2 N_i^l}{\partial x_q^k \partial w}$ .

Moreover, when  $j$  is an integer belonging to  $[1, n^{\text{rank}(P^{(l)r})}]$ , we will also use the partial differential of  $\frac{\partial N^l}{\partial x_q^k}$  with respect to the  $j$ -th coordinate of its  $r$ -th input variable, called  $\frac{\partial^2 N_i^l}{\partial x_q^k \partial x_j^r}$ . Similar notations are available for partial differentials with respect to a given coordinate of the parameter vector.

- As  $o^l$  is a function from  $I \times W$  to  $O^l$ , it can also be considered as a neuron and similar notations will be used, such as for instance  $\frac{\partial^2 o_i^l}{\partial w_q^k \partial w_j^r}$  which stands for the partial differential of  $\frac{\partial o_i^l}{\partial w_q^k}$  with respect to the  $j$  coordinate of the  $r$ -th parameter vector of the network.
- we need also to extend the notion of free network presented in section 2.3.3:
  - $o^{l \rightarrow k, j}(x, w, \tau^k, \tau^j)$  is the output of node  $N^l$  when the input vector of the network is  $x$  and its parameter vector is  $w$ , and when the output of node  $N^k$  takes the value  $\tau^k$  and the output of node  $N^j$  takes the value  $\tau^j$ .
  - $h^{l \rightarrow k, j}(x, w, \tau^k, \tau^j)$  is the generalized output of the network in the case described above.

## 4 Second order derivatives

Let  $\mathcal{G}$  be a feed-forward neural network with  $C^2$  neurons. Let  $N^l$  be an arbitrary node. As a composed function,  $o^l$  is also  $C^2$ . As recalled in sections 2.3.2 and 2.3.4, two methods are available in order to compute  $\frac{\partial o_i^l}{\partial w^k}(x, w)$  (and also  $\frac{\partial o_i^l}{\partial x_j}(x, w)$ ), the direct method and the back-propagation algorithm. The problem is now to compute the differential of  $\frac{\partial o_i^l}{\partial w^k}(x, w)$  itself. The only way to solve this problem is to use the explicit definitions of this function. As we have two computing methods, we have also two explicit definitions.

### 4.1 The direct method approach

#### 4.1.1 General case

Equation 12 given by the direct method for a non input node can be rewritten in a scalar version (valid when  $j \neq l$ ):

$$\frac{\partial o_i^l}{\partial w_t^j}(x, w) = \sum_{N^k \in P^{(l)} \cap S^*(j)} \sum_{q=1}^{n^k} \frac{\partial N_i^l}{\partial x_q^{r(k,l)}} \left( o^{P^{(l)1}}(x, w), \dots, o^{P^{(l)p^l}}(x, w), w^l \right) \frac{\partial o_q^k}{\partial w_t^j}(x, w)$$

The differentials of this equation can be computed very easily, as it is only a sum of products of differentiable functions. Therefore, the main problem is to compute  $\frac{\partial^2 N_i^l}{\partial x_q^k \partial w^m}$  and  $\frac{\partial^2 o_q^k}{\partial w_t^j \partial w^m}$ , where  $m \neq l$ .

A simple application of the chain rule (i.e., a direct differentiation) gives (as  $m \neq l$ ):

$$\frac{\partial^2 N_i^l}{\partial x_q^k \partial w^m} \left( o^{P(l)1}(x, w), \dots, o^{P(l)r^l}(x, w), w^l \right) = \sum_{N^r \in P(l) \cap S^*(m)} \frac{\partial^2 N_i^l}{\partial x_q^k \partial x^{r(r,l)}} (i^l(x, w)) \frac{\partial o^r}{\partial w^m}(x, w)$$

Therefore, the second order differential is obtained with the following equation:

$$\begin{aligned} \frac{\partial^2 o_i^l}{\partial w_t^j \partial w^m}(x, w) = & \\ & \sum_{N^k \in P(l) \cap S^*(j)} \sum_{q=1}^{n^k} \left( \frac{\partial o_q^k}{\partial w_t^j}(x, w) \sum_{N^r \in P(l) \cap S^*(m)} \frac{\partial^2 N_i^l}{\partial x_q^k \partial x^{r(r,l)}} (i^l(x, w)) \frac{\partial o^r}{\partial w^m}(x, w) \right. \\ & \left. + \frac{\partial N_i^l}{\partial x_q^{r(k,l)}} (i^l(x, w)) \frac{\partial^2 o_q^k}{\partial w_t^j \partial w^m}(x, w) \right) \end{aligned}$$

The scalar version of this equation is:

$$\begin{aligned} \frac{\partial^2 o_i^l}{\partial w_t^j \partial w_u^m}(x, w) = & \tag{39} \\ & \sum_{N^k \in P(l) \cap S^*(j)} \sum_{q=1}^{n^k} \left( \frac{\partial o_q^k}{\partial w_t^j}(x, w) \sum_{N^r \in P(l) \cap S^*(m)} \frac{\partial^2 N_i^l}{\partial x_q^k \partial x^{r(r,l)}} (i^l(x, w)) \frac{\partial o^r}{\partial w^m}(x, w) \right. \\ & \left. + \frac{\partial N_i^l}{\partial x_q^{r(k,l)}} (i^l(x, w)) \frac{\partial^2 o_q^k}{\partial w_t^j \partial w_u^m}(x, w) \right) \end{aligned}$$

#### 4.1.2 Particular cases

The previous section dealt with the general case for which we had assumed that  $N^l$  was an *inner* node and that  $N^m \neq N^l$  and  $N^j \neq N^l$ . In this section, we introduce the formulae of the particular cases.

**Extension for  $m = l$**

If  $m = l$ , we just have to compute  $\frac{\partial^2 o_i^l}{\partial w_t^j \partial w_u^l}$ . In fact, we can change the differentiation order and compute  $\frac{\partial^2 o_i^l}{\partial w_u^l \partial w_t^j}$ . Therefore this particular case is just a part of the following particular case (in fact, it is possible to compute  $\frac{\partial^2 o_i^l}{\partial w_t^j \partial w_u^l}$  with this differentiation order: we obtain exactly the same formula as for the alternate differentiation order).

**Extension for  $j = l$**

In this case, for an inner node, we have:

$$\frac{\partial o^l}{\partial w^l}(x, w) = \frac{\partial N^l}{\partial w} (i^l(x, w))$$

Therefore, if  $m \neq l$ , we have:

$$\frac{\partial^2 o_i^l}{\partial w_t^j \partial w_u^m}(x, w) = \sum_{N^k \in P(l) \cap S^*(m)} \frac{\partial^2 N_i^l}{\partial w_t^j \partial x^{r(k,l)}} (i^l(x, w)) \frac{\partial o^k}{\partial w_u^m}(x, w) \tag{40}$$

Finally, if  $m = l$ , we have (as the graph is acyclic):

$$\frac{\partial^2 o_i^l}{\partial w_t^j \partial w_u^l}(x, w) = \frac{\partial^2 N_i^l}{\partial w_t^j \partial w_u^l} (i^l(x, w)) \tag{41}$$

**Extension for input nodes**

Let  $N^l$  be an input node. Then we have:

- If  $j \neq l$ :

$$\frac{\partial o^l}{\partial w^j}(x, w) = 0_{W^j, O^l}$$

therefore for all  $m$ :

$$\frac{\partial^2 o_i^l}{\partial w_t^j \partial w_u^m}(x, w) = 0 \quad (42)$$

- If  $j = l$ :

$$\frac{\partial o^l}{\partial w^l}(x, w) = \frac{\partial N^l}{\partial w}(x_{In(l)}, w^l)$$

therefore for  $m \neq l$ :

$$\frac{\partial^2 o_i^l}{\partial w_t^l \partial w_u^m}(x, w) = 0 \quad (43)$$

and:

$$\frac{\partial^2 o_i^l}{\partial w_t^l \partial w_u^l}(x, w) = \frac{\partial^2 N_i^l}{\partial w_t^l \partial w_u^l}(x_{In(l)}, w^l) \quad (44)$$

**4.1.3 Analysis**

The general formula allows to compute recursively  $\frac{\partial^2 o_i^l}{\partial w_t^j \partial w_u^m}$ . In order to apply this formula, the following values must have been computed:

- $\frac{\partial^2 o_q^k}{\partial w_t^j \partial w_u^m}$  for all  $k \in P(l)$  (and for all  $1 \leq q \leq n^k$ ). The formula must therefore be computed layer by layer, from the input layer to the last one. It is very important to notice that computing the Hessian matrix of  $o_i^l$  implies to compute the Hessian matrices of  $o_q^k$  for all  $N^k$  belonging to  $P^+(l)$ .
- $\frac{\partial o_q^k}{\partial w_t^j}$  and  $\frac{\partial o_q^k}{\partial w_u^m}$  for all  $k \in P(l)$  (and for all  $1 \leq q \leq n^k$ ). These first order differentials can be computed with the direct method or with the back-propagation method.
- $\frac{\partial^2 N_i^l}{\partial x_q^{r(k,l)} \partial x_s^{r(r,l)}}$  for all  $k$  and  $r$  in  $P(l)$  (and for all  $1 \leq q \leq n^k$  and for all  $1 \leq s \leq n^r$ ). This local second order differential can be computed as long as we know the input of neuron  $N^l$ ,  $i^l(x, w)$ .
- $\frac{\partial N_i^l}{\partial x_q^{r(k,l)}}$  for all  $k \in P(l)$  (and for all  $1 \leq q \leq n^k$ ). Once again, this local first order differential can be computed as long as we know the input of neuron  $N^l$ .

The obtained formula can rather easily be applied in order to obtain the second order differentials of the output of the net. It is important to notice that a mixed algorithm can be used: we can indeed use the back-propagation algorithm to compute the first order differentials (i.e.,  $\frac{\partial o_q^k}{\partial w_t^j}$ ) and then applied the direct method for the second order computation.

**4.2 The back-propagation approach****4.2.1 Local equations**

In order to simplify the formulae, we use the following slightly incorrect notation:  $\frac{\partial o^l}{\partial \mathbf{o}^j}(\mathbf{x}, \mathbf{w})$  is the partial differential  $\frac{\partial o^{l-j}}{\partial \mathbf{o}^j}(\mathbf{x}, \mathbf{w}, \mathbf{o}^j(\mathbf{x}, \mathbf{w}))$ . With this new notation, equation 24 can be rewritten in a scalar version:

$$\frac{\partial o_i^l}{\partial w_t^j}(x, w) = \sum_{q=1}^{n^j} \frac{\partial o_i^l}{\partial o_q^j}(x, w) \frac{\partial N_q^j}{\partial w_t^j}(i^j(x, w))$$

Therefore, we have, when  $N^l$  is an inner node and for  $N^m \neq N^j$ , we have:

$$\frac{\partial^2 o_i^l}{\partial w_t^j \partial w_u^m}(x, w) = \sum_{q=1}^{n^j} \left( \frac{\partial o_i^l}{\partial \sigma_q^j}(x, w) \sum_{N^r \in P(j) \cap S^*(m)} \frac{\partial^2 N_q^j}{\partial w_t^j \partial x^{r(j,k)}}(i^j(x, w)) \frac{\partial o^r}{\partial w_u^m}(x, w) \right. \\ \left. + \frac{\partial^2 o_i^l}{\partial \sigma_q^j \partial w_u^m}(x, w) \frac{\partial N_q^j}{\partial w_t^j}(i^j(x, w)) \right) \quad (45)$$

The obtained formula is also valid for  $m = l$ . Equation 45 can be simplified when  $N^m = N^l$ . In this case,  $P(j) \cap S^*(l) = \emptyset$  because  $N^j \in P^+(l)$  and the network is acyclic. Therefore, equation 45 becomes:

$$\frac{\partial^2 o_i^l}{\partial w_t^j \partial w_u^l}(x, w) = \sum_{q=1}^{n^j} \frac{\partial^2 o_i^l}{\partial \sigma_q^j \partial w_u^l}(x, w) \frac{\partial N_q^j}{\partial w_t^j}(i^j(x, w)) \quad (46)$$

This equation does not seem to be an excellent choice for computing  $\frac{\partial^2 o_i^l}{\partial w_t^j \partial w_u^l}$ , because it needs some non local second order differentials (the  $\frac{\partial^2 o_i^l}{\partial \sigma_q^j \partial w_u^l}$ ) whereas the direct method, which computes  $\frac{\partial^2 o_i^l}{\partial w_u^l \partial w_t^j}$  with equation 40, uses only local differentials and first order differentials.

When  $N^m = N^j$ , we have:

$$\frac{\partial^2 o_i^l}{\partial w_t^j \partial w_u^j} = \sum_{q=1}^{n^j} \left( \frac{\partial^2 o_i^l}{\partial \sigma_q^j \partial w_u^j}(x, w) \frac{\partial N_q^j}{\partial w_t^j}(i^j(x, w)) + \frac{\partial o_i^l}{\partial \sigma_q^j}(x, w) \frac{\partial^2 N_q^j}{\partial w_t^j \partial w_u^j}(i^j(x, w)) \right) \quad (47)$$

Both formulae use local first and second order differentials (Jacobian and Hessian matrices of the considered node  $N^j$ ). They use also non local first order differentials such as  $\frac{\partial o_i^l}{\partial \sigma^j}$  which can be computed with the help of the different algorithms presented in the NeuroColt report. They use finally a second order differential ( $\frac{\partial^2 o_i^l}{\partial \sigma_q^j \partial w_u^j}$ ) which is the differential of the local output with respect to non local variables. As the formulae are not recursive, we call them *local formulae*.

The main problem in order to compute the second order differentials with the back-propagation approach is to compute the differentials of  $\frac{\partial o_i^l}{\partial \sigma^j}$ . Two methods are available: a direct method and a back-propagation method. It is important to notice that the computation of this differential for an input node is not needed: we just have to apply the direct formulae given in subsection 4.1.2.

#### 4.2.2 Direct differentiation of $\frac{\partial o_i^l}{\partial \sigma^j}$ : the hybrid method

The main idea is to apply the chain rule to the recursive definition of  $\frac{\partial o_i^l}{\partial \sigma^j}$  obtained in the back-propagation theorem (theorem 2). Of course, as this function is constant when  $N^j \notin P^+(l)$ , its differential is null in this case. Therefore, the only interesting point is when  $N^j \in P^+(l)$ . In this case, we apply equation 29 which scalar version is (with the new notation introduced above):

$$\frac{\partial o_i^l}{\partial \sigma_t^j}(x, w) = \sum_{N^k \in S(j) \cap P^+(l)} \sum_{q=1}^{n^k} \frac{\partial o_i^l}{\partial \sigma_q^k}(x, w) \frac{\partial N_q^k}{\partial x_t^{r(j,k)}}(i^k(x, w))$$

If  $N^m \notin S(j)$ , we have:

$$\frac{\partial^2 o_i^l}{\partial \sigma_t^j \partial w_u^m}(x, w) = \sum_{N^k \in S(j) \cap P^+(l)} \sum_{q=1}^{n^k} \left( \frac{\partial o_i^l}{\partial \sigma_q^k}(x, w) \sum_{N^r \in P(k) \cap S^*(m)} \frac{\partial^2 N_q^k}{\partial x_t^{r(j,k)} \partial x^{r(r,k)}}(i^k(x, w)) \frac{\partial o^r}{\partial w_u^m}(x, w) \right. \\ \left. + \frac{\partial^2 o_i^l}{\partial \sigma_q^k \partial w_u^m}(x, w) \frac{\partial N_q^k}{\partial x_t^{r(j,k)}}(i^k(x, w)) \right) \quad (48)$$

When  $N^m \in S(j)$ , we have:

$$\begin{aligned} \frac{\partial^2 o_i^l}{\partial \sigma_t^j \partial w_u^m}(x, w) = & \quad (49) \\ & \sum_{N^k \in S(j) \cap P^*(l), k \neq m} \sum_{q=1}^{n^k} \left( \frac{\partial o_i^l}{\partial \sigma_q^k}(x, w) \sum_{N^r \in P(k) \cap S^*(m)} \frac{\partial^2 N_q^k}{\partial x_t^{r(j,k)} \partial x^{r(r,k)}}(i^k(x, w)) \frac{\partial \sigma^r}{\partial w_u^m}(x, w) \right) \\ & + \frac{\partial o_i^l}{\partial \sigma^m}(x, w) \frac{\partial^2 N^m}{\partial x_t^{r(j,m)} \partial w_u^m}(i^m(x, w)) \\ & + \sum_{N^k \in S(j) \cap P^*(l)} \sum_{q=1}^{n^k} \frac{\partial^2 o_i^l}{\partial \sigma_q^k \partial w_u^m}(x, w) \frac{\partial N_q^k}{\partial x_t^{r(j,k)}}(i^k(x, w)) \end{aligned}$$

Both formulae are also valid for  $N^m = N^l$  but can be simplified in this case. Let  $N^k$  be a successor of  $N^j$ . If  $N^k \notin P^*(l)$ , then  $\frac{\partial o_i^l}{\partial \sigma^k} = 0$ . If  $N^k \in P^+(l)$ , then, as the graph is acyclic,  $P(k) \cap S^*(l) = \emptyset$  and therefore, for all  $N^r \in P(k)$ ,  $\frac{\partial \sigma^r}{\partial w^l} = 0$ . Therefore, when  $k \neq l$ , the first part of both formulae is null (i.e.,  $\frac{\partial o_i^l}{\partial \sigma_q^k} \sum_{r=1}^{p^k} \frac{\partial^2 N_q^k}{\partial x_t^{r(j,k)} \partial x^r} \frac{\partial \sigma^{P(k)r}}{\partial w_u^m} = 0$ ). Therefore we have, for  $N^l \notin S(j)$ :

$$\frac{\partial^2 o_i^l}{\partial \sigma_t^j \partial w_u^l}(x, w) = \sum_{N^k \in S(j) \cap P^*(l)} \sum_{q=1}^{n^k} \left( \frac{\partial^2 o_i^l}{\partial \sigma_q^k \partial w_u^l}(x, w) \frac{\partial N_q^k}{\partial x_t^{r(j,k)}}(i^k(x, w)) \right) \quad (50)$$

When  $N^l \in S(j)$ :

$$\begin{aligned} \frac{\partial^2 o_i^l}{\partial \sigma_t^j \partial w_u^l}(x, w) = & \quad (51) \\ & \sum_{N^k \in S(j) \cap P^*(l)} \sum_{q=1}^{n^k} \left( \frac{\partial^2 o_i^l}{\partial \sigma_q^k \partial w_u^l}(x, w) \frac{\partial N_q^k}{\partial x_t^{r(j,k)}}(i^k(x, w)) \right) + \frac{\partial^2 N^l}{\partial x_t^{r(j,l)} \partial w_u^l}(i^l(x, w)) \end{aligned}$$

The four formulae presented here have a main difference compared to the local formulae obtained in the previous section. In fact, in order to obtain  $\frac{\partial^2 o_i^l}{\partial \sigma_t^j \partial w_u^m}$ , we need  $\frac{\partial^2 o_i^l}{\partial \sigma_q^k \partial w_u^m}$  and this value will be obtained with one of the four equations. Therefore, they are *recursive equations*.

The main problem of this approach is that the obtained formulae are not symmetric at all. As the considered functions are  $C^2$ , the differentiation order does not change the results. Therefore, we have for instance  $\frac{\partial^2 o_i^l}{\partial w_t^j \partial w_u^m} = \frac{\partial^2 o_i^l}{\partial w_u^m \partial w_t^j}$ . As the recursive formulae proposed with the previous approach and with the direct method are not symmetric, the computation cost depends on the chosen differentiation order. But as explained in subsection 6.2.3, choosing an efficient differentiation order is a very complex task. In order to avoid this problem, we introduce in the following section symmetric recursive equation which allows to optimize easily the computation order.

#### 4.2.3 Back-propagation based differentiation of $\frac{\partial o_i^l}{\partial \sigma^j}$ : the pure back-propagation method

The main point in the back-propagation method is that the output of the network depends on the parameter vector  $w^k$  only through the output of node  $N^k$ . This is of course also true for the differential  $\frac{\partial o_i^l}{\partial \sigma^j}$ . Therefore, we can extend the idea of the back-propagation in order to compute the differential of this function.

Let us first introduce a “free” differential, which generalizes to the differentials the “free” network notion presented in section 2.3.3.

**Definition 12** Let  $\mathcal{G} = (\mathcal{N}, \mathcal{F}, <)$  be a feed-forward neural network. Let  $N^l$ ,  $N^j$  and  $N^m$  be arbitrary nodes. The function  $\left(\frac{\partial o^l}{\partial o^j}\right)^{\rightarrow m}$  from  $I \times W \times O^m$  to  $L(O^j, O^l)$  (i.e., the vectorial space of linear applications from  $O^j$  to  $O^l$ , to which belongs  $\frac{\partial o^l}{\partial o^j}(x, w)$ ), is given by:

- If  $N^j = N^l$ :

$$\left(\frac{\partial o^l}{\partial o^j}\right)^{\rightarrow m}(x, w, f^m) = Id_{O^j} \quad (52)$$

- If  $N^j \notin P^*(N^l)$ :

$$\left(\frac{\partial o^l}{\partial o^j}\right)^{\rightarrow m}(x, w, f^m) = 0_{O^j, O^l} \quad (53)$$

- If  $N^j \in P^+(N^l)$ :

$$\left(\frac{\partial o^l}{\partial o^j}\right)^{\rightarrow m}(x, w, f^m) = \sum_{N^k \in S(j) \cap P^*(l)} \left(\frac{\partial o^l}{\partial o^k}\right)^{\rightarrow m}(x, w, f^m) \frac{\partial N^k}{\partial x^{r(j,k)}}(i^{k \rightarrow m}(x, w, f^m)) \quad (54)$$

The abusive notation  $\frac{\partial}{\partial o^m} \left(\frac{\partial o^l}{\partial o^j}\right)^{\rightarrow m}(x, w, f^m)$  will represent the partial differential of  $\left(\frac{\partial o^l}{\partial o^j}\right)^{\rightarrow m}$  with respects to its third variable.

This new function satisfies important properties:

**Property 7** Let  $\mathcal{G} = (\mathcal{N}, \mathcal{F}, <)$  be a feed-forward neural network. Let  $N^l$ ,  $N^j$  and  $N^m$  be arbitrary nodes, and let  $x$  and  $w$  be respectively an arbitrary input vector and an arbitrary parameter vector for the network. Then we have:

$$\left(\frac{\partial o^l}{\partial o^j}\right)^{\rightarrow m}(x, w, o^m(x, w)) = \frac{\partial o^l}{\partial o^j}(x, w) \quad (55)$$

**Proof** — This is a straightforward consequence of the definition. ■

**Property 8** Let  $\mathcal{G} = (\mathcal{N}, \mathcal{F}, <)$  be a feed-forward neural network. Let  $N^l$ ,  $N^j$  and  $N^m$  be arbitrary nodes, with  $N^m \notin S^+(j)$ , and let  $x$  and  $w$  be respectively an arbitrary input vector and an arbitrary parameter vector for the network. Then we have (for all  $1 \leq i \leq n^l$  and  $1 \leq t \leq n^j$ ):

$$\frac{\partial^2 o_i^l}{\partial o_t^j \partial w^m}(x, w) = \frac{\partial}{\partial o^m} \left(\frac{\partial o_i^l}{\partial o_t^j}\right)^{\rightarrow m}(x, w, o^m(x, w)) \frac{\partial N^m}{\partial w}(i^m(x, w)) \quad (56)$$

**Proof** — This property is based on the fact that  $\left(\frac{\partial o_i^l}{\partial o_t^j}\right)^{\rightarrow m}(x, w, f^m)$  does not depend on  $w^m$  (this is an obvious consequence of the definition, as  $N^m \notin S^+(j)$ ). ■

The fact that this property cannot be used when  $N^m \in S^+(j)$  shows that the pure back-propagation approach cannot be used in order to compute  $\frac{\partial^2 o_i^l}{\partial w_t^j \partial w_u^m}$  in this order. Therefore, for this part of the computation, we must revert to the hybrid method or to the direct method (we assume in the rest of the report that the direct method is used).

The last thing to do in order to obtain  $\frac{\partial^2 o_i^l}{\partial o_t^j \partial w^m}$  is now to compute the differential of  $\left(\frac{\partial o_i^l}{\partial o_t^j}\right)^{\rightarrow m}$ . Its recursive definition implies the following equalities (for all  $1 \leq i \leq n^l$  and  $1 \leq t \leq n^j$ ):

- If  $N^j = N^l$ :

$$\frac{\partial}{\partial o^m} \left(\frac{\partial o_i^l}{\partial o_t^j}\right)^{\rightarrow m}(x, w, f^m) = 0_{O^m, \mathbb{R}} \quad (57)$$

- If  $N^j \notin P^*(N^l)$ :

$$\frac{\partial}{\partial o^m} \left( \frac{\partial o_i^l}{\partial o_t^j} \right)^{\rightarrow m} (x, w, f^m) = 0_{O^m, \mathbb{R}} \quad (58)$$

The recursive case is more complex to study. The matricial equality 54 implies the following scalar version:

$$\begin{aligned} \left( \frac{\partial o_i^l}{\partial o_t^j} \right)^{\rightarrow m} (x, w, f^m) = \\ \sum_{N^k \in S(j) \cap P^*(l)} \sum_{q=1}^{n^k} \left( \frac{\partial o_i^l}{\partial o_q^k} \right)^{\rightarrow m} (x, w, f^m) \frac{\partial N_q^k}{\partial x_t^{r(j,k)}} \left( o^{P(k)^1 \rightarrow m}(x, w, f^m), \dots, o^{P(k)^{p^k} \rightarrow m}(x, w, f^m), w^k \right) \end{aligned}$$

Differentiating this equation gives:

$$\begin{aligned} \frac{\partial}{\partial o_v^m} \left( \frac{\partial o_i^l}{\partial o_t^j} \right)^{\rightarrow m} (x, w, f^m) = \\ \sum_{N^k \in S(j) \cap P^*(l)} \sum_{q=1}^{n^k} \left( \left( \frac{\partial o_i^l}{\partial o_q^k} \right)^{\rightarrow m} (x, w, f^m) \sum_{r=1}^{p^k} \frac{\partial^2 N_q^k}{\partial x_t^{r(j,k)} \partial x^r} (i^{k \rightarrow m}(x, w, f^m)) \frac{\partial o^{P(k)^r \rightarrow m}}{\partial o_v^m} (x, w, f^m) \right. \\ \left. + \frac{\partial}{\partial o_v^m} \left( \frac{\partial o_i^l}{\partial o_q^k} \right)^{\rightarrow m} (x, w, f^m) \frac{\partial N_q^k}{\partial x_t^{r(j,k)}} (i^{k \rightarrow m}(x, w, f^m)) \right) \end{aligned}$$

Therefore, we have:

$$\begin{aligned} \frac{\partial}{\partial o_v^m} \left( \frac{\partial o_i^l}{\partial o_t^j} \right)^{\rightarrow m} (x, w, o^m(x, w)) = \\ \sum_{N^k \in S(j) \cap P^*(l)} \sum_{q=1}^{n^k} \left( \frac{\partial o_i^l}{\partial o_q^k} (x, w) \sum_{N^r \in P(k) \cap S^*(m)} \frac{\partial^2 N_q^k}{\partial x_t^{r(j,k)} \partial x^{r(r,k)}} (i^k(x, w)) \frac{\partial o^r}{\partial o_v^m} (x, w) \right. \\ \left. + \frac{\partial}{\partial o_v^m} \left( \frac{\partial o_i^l}{\partial o_q^k} \right)^{\rightarrow m} (x, w, o^m(x, w)) \frac{\partial N_q^k}{\partial x_t^{r(j,k)}} (i^k(x, w)) \right) \end{aligned} \quad (59)$$

#### 4.2.4 Symmetric formulae

A new function,  $\left( \frac{\partial o_i^l}{\partial o_t^j} \right)^{\rightarrow m}$  has been introduced in the previous section. It is very important to notice that this function, which should be written  $\left( \frac{\partial o_i^{l \rightarrow j}}{\partial o_t^j} \right)^{\rightarrow m}$  is totally different from  $\frac{\partial o_i^{l \rightarrow j, m}}{\partial o_t^j}$ . Therefore, the differentials of these functions are different. This is a really important result because it means that the differentiation order is important, i.e., we have:  $\frac{\partial}{\partial o_v^m} \left( \frac{\partial o_i^l}{\partial o_t^j} \right)^{\rightarrow m} \neq \frac{\partial}{\partial o_t^j} \left( \frac{\partial o_i^l}{\partial o_v^m} \right)^{\rightarrow j}$ . But, we have the following property:

**Property 9** If  $N^m \notin S^*(N^j)$  and  $N^j \notin S^*(N^m)$ :

$$\frac{\partial o_i^{l \rightarrow j, m}}{\partial o_t^j} (x, w, o^j(x, w), \tau^m) = \left( \frac{\partial o_i^l}{\partial o_t^j} \right)^{\rightarrow m} (x, w, \tau^m) \quad (60)$$

**Proof** — This proof is based on a Lagrangian method. The basic idea is exactly the same as the proof of the first order back-propagation given in [6].

We first introduce a Lagrangian function:

$$\begin{aligned} L_i^l(x, w, \tau, \lambda) = \tau_i^l & - \sum_{k \in P^*(l), k \notin In} \sum_{j=1}^{n^k} \lambda_j^k \left( \tau_j^k - N_j^k \left( \tau^{P(k)^1}, \dots, \tau^{P(k)^{p^k}}, w^k \right) \right) \\ & - \sum_{k \in P^*(l) \cap In} \sum_{j=1}^{n^k} \lambda_j^k \left( \tau_j^k - N_j^k \left( x^{In(k)}, w^k \right) \right) \end{aligned}$$

The constraints of this Lagrangian function are satisfied when  $\tau^l = o^l(x, w)$ . In this case, for all  $\lambda$ :

$$L_i^l(x, w, \tau, \lambda) = o_i^l(x, w)$$

Since the constraints are equivalent to  $\tau = h(x, w)$ , we have, for all  $x, w$  and  $\lambda$ :

$$L_i^l(x, w, h(x, w), \lambda) = o_i^l(x, w)$$

As demonstrated in the NeuroColt report, we can choose  $\lambda$  so that  $\frac{\partial L_i^l}{\partial \tau} = 0$ . We have in fact:

If  $k \notin P^*(l)$ , then:

$$\frac{\partial L_i^l}{\partial \tau^k}(x, w, \tau, \lambda) = 0,$$

because  $\tau^k$  does not appear in the definition of  $L_i^l$ .

If  $k \in P^*(l)$ , different cases can occur:

- If  $k = l$ , then:
  - If  $j \neq i$ , then:

$$\frac{\partial L_i^l}{\partial \tau_j^l}(x, w, \tau, \lambda) = -\lambda_j^l,$$

because  $\tau_j^l$  does not appear as input of a neuron (i.e., in the right part of the constraints), since if there was some  $k \in P^*(l)$  such that  $P(k)^q = N^l$ , then the graph would be cyclic.

- If  $j = i$ , then:

$$\frac{\partial L_i^l}{\partial \tau_i^l}(x, w, \tau, \lambda) = 1 - \lambda_j^l,$$

- If  $k \neq l$ , then:

$$\frac{\partial L_i^l}{\partial \tau_j^k}(x, w, \tau, \lambda) = -\lambda_j^k + \sum_{q \in S(k) \cap P^*(l)} \sum_{p=1}^{n^q} \lambda_p^q \frac{\partial N_p^q}{\partial x_j^{r(k,q)}} \left( \tau^{P(q)^1}, \dots, \tau^{P(q)^{p^q}}, w^q \right)$$

Let us now define  $C_i^l(w, \tau)$ :

- If  $k \notin P^*(l)$ , then for all  $j$ :

$$C_i^l(w, \tau)_j^k = 0$$

- if  $k = l$ , then for all  $j$ :

$$C_i^l(w, \tau)_j^l = \delta_{ij}$$

- if  $k \in P^+(l)$ , then for all  $j$ :

$$C_i^l(w, \tau)_j^k = \sum_{q \in S(k) \cap P^*(l)} \sum_{p=1}^{n^q} C_i^l(w, \tau)_p^q \frac{\partial N_p^q}{\partial x_j^{r(k,q)}} \left( \tau^{P(q)^1}, \dots, \tau^{P(q)^{p^q}}, w^q \right)$$

The definition of  $C_i^l(w, \tau)$  implies that:

$$\frac{\partial L_i^l}{\partial \tau} (x, w, \tau, C_i^l(w, \tau)) = 0$$

We introduce another Lagrangian function in which the constraints on  $N^j$  and  $N^m$  are removed:

$$\begin{aligned} L_i^{l \rightarrow j, m}(x, w, \tau, \lambda) = & \tau_i^l - \sum_{q \in P^*(l), q \notin I_n, q \neq j, q \neq m} \sum_{k=1}^{n^q} \lambda_k^q \left( \tau_k^q - N_k^q \left( \tau^{P(q)^1}, \dots, \tau^{P(q)^{p^q}}, w^q \right) \right) \\ & - \sum_{q \in P^*(l) \cap I_n, q \neq j, q \neq m} \sum_{k=1}^{n^q} \lambda_k^q \left( \tau_k^q - N_k^q \left( x^{I_n(q)}, w^q \right) \right) \end{aligned}$$

The constraints in this Lagrangian function are fulfilled if and only if:

$$\forall q, \tau^q = o^{q \rightarrow j, m}(x, w, \tau^j, \tau^m)$$

Therefore, for arbitrary values of  $\tau^j, \tau^m, x, w$  and  $\lambda$ , we have:

$$L_i^{l \rightarrow j, m}(x, w, h^{-j, m}(x, w, \tau^j, \tau^m), \lambda) = o_i^{l \rightarrow j, m}(x, w, \tau^j, \tau^m)$$

Therefore, for any  $\tau^j, \tau^m, x, w$  and  $\lambda$ :

$$\frac{\partial o_i^{l \rightarrow j, m}}{\partial o^j} (x, w, \tau^j, \tau^m) = \frac{\partial L_i^{l \rightarrow j, m}}{\partial \tau} (x, w, h^{-j, m}(x, w, \tau^j, \tau^m), \lambda) \frac{\partial h^{-j, m}}{\partial \tau^j} (x, w, \tau^j, \tau^m)$$

Moreover, for  $q \neq j, q \neq m, q \notin P(j)$  and  $q \notin P(m)$ :

$$\frac{\partial L_i^{l \rightarrow j, m}}{\partial \tau^q} (x, w, \tau, \lambda) = \frac{\partial L_i^l}{\partial \tau^q} (x, w, \tau, \lambda),$$

and therefore:

$$\frac{\partial L_i^{l \rightarrow j, m}}{\partial \tau^q} (x, w, \tau, C_i^l(w, \tau)) = 0$$

We have also, as  $N^m \notin S(j)$ :

$$\frac{\partial L_i^{l \rightarrow j, m}}{\partial \tau_t^j} (x, w, \tau, \lambda) = \sum_{q \in S(j) \cap P^*(l)} \sum_{p=1}^{n^q} \lambda_p^q \frac{\partial N_p^q}{\partial x_t^{r(j, q)}} \left( \tau^{P(q)^1}, \dots, \tau^{P(q)^{p^q}}, w^q \right)$$

Therefore, we have:

$$\frac{\partial L_i^{l \rightarrow j, m}}{\partial \tau_t^j} (x, w, \tau, C_i^l(w, \tau)) = C_i^l(w, \tau)_t^j$$

Finally, we have:

$$\frac{\partial o_i^{l \rightarrow j, m}}{\partial o_t^j} (x, w, \tau^j, \tau^m) = \sum_{q=1}^n \frac{\partial L_i^{l \rightarrow j, m}}{\partial \tau^q} (x, w, h^{-j, m}(x, w, \tau^j, \tau^m), \lambda) \frac{\partial o^{q \rightarrow j, m}}{\partial o_t^j} (x, w, \tau^j, \tau^m)$$

The definition of  $o^{m \rightarrow j, m}$  implies that  $\frac{\partial o^{m \rightarrow j, m}}{\partial o_t^j} (x, w, \tau^j, \tau^m) = 0$ . Moreover, as the network is not cyclic,  $N^q \in P(j)$  implies that  $N^q \notin S^*(j)$  and therefore that  $\frac{\partial o^{q \rightarrow j, m}}{\partial o_t^j} (x, w, \tau^j, \tau^m) = 0$ . If  $N^q \in P(m)$ , then  $N^q \notin S^*(j)$  (the contrary implies  $N^m \in S^*(j)$ , which is false). Once again, the consequence is  $\frac{\partial o^{q \rightarrow j, m}}{\partial o_t^j} (x, w, \tau^j, \tau^m) = 0$ . Finally, when  $q \neq j, q \neq m, q \notin P(j)$  and  $q \notin P(m)$ :

$$\frac{\partial L_i^{l \rightarrow j, m}}{\partial \tau^q} (x, w, \tau, C_i^l(x, w)) = 0$$

Therefore, we have:

$$\frac{\partial o_i^{l \rightarrow j, m}}{\partial o_t^j}(x, w, \tau^j, \tau^m) = \frac{\partial L_i^{l \rightarrow j, m}}{\partial \tau_t^j}(x, w, h^{-j, m}(x, w, \tau^j, \tau^m), C_i^l(w, h^{-j, m}(x, w, \tau^j, \tau^m))),$$

i.e.,

$$\frac{\partial o_i^{l \rightarrow j, m}}{\partial o_t^j}(x, w, \tau^j, \tau^m) = C_i^l(w, h^{-j, m}(x, w, \tau^j, \tau^m))$$

Finally, we have:

$$\frac{\partial o_i^{l \rightarrow j, m}}{\partial o_t^j}(x, w, o^j(x, w), \tau^m) = C_i^l(w, h^{-j, m}(x, w, o^j(x, w), \tau^m))$$

But as  $N^m \notin P^*(N^j)$ ,  $h^{-j, m}(x, w, o^j(x, w), \tau^m) = h^{-j, m}(x, w, o^{j \rightarrow m}(x, w, \tau^m), \tau^m) = h^{-m}(x, w, \tau^m)$ . Therefore we have:

$$\frac{\partial o_i^{l \rightarrow j, m}}{\partial o_t^j}(x, w, o^j(x, w), \tau^m) = C_i^l(w, h^{-m}(x, w, \tau^m)) = \left(\frac{\partial o_i^l}{\partial o_t^j}\right)^{-m}(x, w, \tau^m)$$

■

Therefore, if  $N^m \notin S^*(N^j)$  and  $N^j \notin S^*(N^m)$ , we have:

$$\frac{\partial^2 o_i^{l \rightarrow j, m}}{\partial o_t^j \partial o^m}(x, w, o^j(x, w), \tau^m) = \frac{\partial}{\partial o^m} \left(\frac{\partial o_i^l}{\partial o_t^j}\right)^{-m}(x, w, \tau^m)$$

Therefore, if  $N^j \notin S^*(N^m)$  and  $N^m \notin S^*(N^j)$ , we have:

$$\frac{\partial}{\partial o_v^m} \left(\frac{\partial o_i^l}{\partial o_t^j}\right)^{-m}(x, w, o^m(x, w)) = \frac{\partial}{\partial o_t^j} \left(\frac{\partial o_i^l}{\partial o_v^m}\right)^{-j}(x, w, o^j(x, w)) \quad (61)$$

This means that, as long as the required conditions are fulfilled, it is possible to change the differentiation order so that the time needed to compute the differential is minimized.

## 5 Error function

The most important practical use of the second order derivation is of course to compute the Hessian of the error function of the network. This calculation is slightly different from what has been presented before and the purpose of this section is to explain these differences. Its is based on the formalism described in section 2.3.5. We consider here only the case of a real valued output function, called  $\mathcal{E}$ . Moreover, we introduce (once again) a quite incorrect notation.  $\frac{\partial \mathcal{E}}{\partial o^k}(x, w)$  stands for  $\frac{\partial \mathcal{E}^k}{\partial o^k}(x, w, o^k(x, w))$ .

### 5.1 Direct method

The error function acts as an output node with no parameter and therefore, we shall use the following formula, where  $N^m$  and  $N^j$  are two arbitrary nodes of the network:

$$\frac{\partial^2 \mathcal{E}}{\partial w_t^j \partial w_u^m} = \sum_{N^k \in \text{Out} \cap S^*(j)} \sum_{q=1}^{n^k} \left( \frac{\partial o_q^k}{\partial w_t^j} \sum_{N^r \in \text{Out} \cap S^*(m)} \frac{\partial^2 \mathcal{E}}{\partial x_q^{\text{Out}(k)} \partial x^{\text{Out}(r)}} \frac{\partial o^r}{\partial w_u^m} + \frac{\partial \mathcal{E}}{\partial x_q^{\text{Out}(k)}} \frac{\partial^2 o_q^k}{\partial w_t^j \partial w_u^m} \right) \quad (62)$$

Computing this Hessian matrix introduced exactly the computation cost as having a last node with no parameter. The direct method implies to compute the Hessian matrix of each network node.

## 5.2 The hybrid method

The back-propagation based methods use the following recursive equations (see theorem 3):

- if  $N^k = Out^j$  ( $j = Out(N^k) = Out(k)$ ):

$$\frac{\partial \mathcal{E}}{\partial \sigma^k}(x, w) = \frac{\partial \mathcal{E}}{\partial x^j}(G(x, w))$$

- if  $N^k \notin Out$ :

$$\frac{\partial \mathcal{E}}{\partial \sigma^k}(x, w) = \sum_{N^j \in S(k)} \frac{\partial \mathcal{E}}{\partial \sigma^j}(x, w) \frac{\partial N^j}{\partial x^{r(k,j)}}(i^j(x, w))$$

The first order differential of the error are obtained with:

$$\frac{\partial \mathcal{E}}{\partial w^k} = \frac{\partial \mathcal{E}}{\partial \sigma^k} \frac{\partial N^k}{\partial w^k}$$

Therefore, the second order differential is obtained with the following local formula (when  $N^m \neq N^j$ ):

$$\frac{\partial^2 \mathcal{E}}{\partial w_t^j \partial w_u^m} = \sum_{q=1}^{n^j} \left( \frac{\partial \mathcal{E}}{\partial \sigma_q^j} \sum_{N^r \in P(j) \cap S^*(m)} \frac{\partial^2 N_q^j}{\partial w_t^j \partial x^{r(j,j)}} \frac{\partial \sigma^r}{\partial w_u^m} + \frac{\partial^2 \mathcal{E}}{\partial \sigma_q^j \partial w_u^m} \frac{\partial N_q^j}{\partial w_t^j} \right) \quad (63)$$

When  $N^m = N^j$ , we have another local formula:

$$\frac{\partial^2 \mathcal{E}}{\partial w_t^j \partial w_u^j} = \sum_{q=1}^{n^j} \left( \frac{\partial^2 \mathcal{E}}{\partial \sigma_q^j \partial w_u^j} \frac{\partial N_q^j}{\partial w_t^j} + \frac{\partial \mathcal{E}}{\partial \sigma_q^j} \frac{\partial^2 N_q^j}{\partial w_t^j \partial w_u^j} \right) \quad (64)$$

The main difference between the case of the error and the general case is the way the starting values of the back-propagation are handled. For the general case, we have  $\frac{\partial \sigma^i}{\partial \sigma^i} = Id$ , a constant value and therefore the second order differential is zero. In the error case, we have the following results:

- if  $N^j = Out^k$  ( $k = Out(N^j) = Out(j)$ ):

$$\frac{\partial^2 \mathcal{E}}{\partial \sigma_t^j \partial w_u^m}(x, w) = \sum_{N^r \in Out \cap S^*(m)} \frac{\partial^2 \mathcal{E}}{\partial x_t^j \partial x^{Out(r)}} \frac{\partial \sigma^r}{\partial w_u^m} \quad (65)$$

- if  $N^j \notin Out$  and  $N^m \notin S(j)$ :

$$\frac{\partial^2 \mathcal{E}}{\partial \sigma_t^j \partial w_u^m} = \sum_{N^k \in S(j)} \sum_{q=1}^{n^k} \left( \frac{\partial \mathcal{E}}{\partial \sigma_q^k} \sum_{N^r \in P(k) \cap S^*(m)} \frac{\partial^2 N_q^k}{\partial x_t^{r(j,k)} \partial x^{r(r,k)}} \frac{\partial \sigma^r}{\partial w_u^m} + \frac{\partial^2 \mathcal{E}}{\partial \sigma_q^k \partial w_u^m} \frac{\partial N_q^k}{\partial x_t^{r(j,k)}} \right) \quad (66)$$

- if  $N^j \notin Out$  and  $N^m \in S(j)$ :

$$\begin{aligned} \frac{\partial^2 \mathcal{E}}{\partial \sigma_t^j \partial w_u^m} = & \sum_{N^k \in S(j), k \neq m} \sum_{q=1}^{n^k} \left( \frac{\partial \mathcal{E}}{\partial \sigma_q^k} \sum_{N^r \in P(k) \cap S^*(m)} \frac{\partial^2 N_q^k}{\partial x_t^{r(j,k)} \partial x^{r(r,k)}} \frac{\partial \sigma^r}{\partial w_u^m} \right) \\ & + \sum_{N^k \in S(j)} \sum_{q=1}^{n^k} \left( \frac{\partial^2 \mathcal{E}}{\partial \sigma_q^k \partial w_u^m} \frac{\partial N_q^k}{\partial x_t^{r(j,k)}} \right) + \frac{\partial \mathcal{E}}{\partial \sigma^m} \frac{\partial^2 N^m}{\partial x_t^{r(j,m)} \partial w_u^m} \end{aligned} \quad (67)$$

### 5.3 Pure back-propagation

As for a network node, it is possible to define  $\left(\frac{\partial \mathcal{E}}{\partial o^j}\right)^{\rightarrow m}$ . We have:

- if  $N^j = Out^k$ :

$$\left(\frac{\partial \mathcal{E}}{\partial o^j}\right)^{\rightarrow m}(x, w, f^m) = \frac{\partial \mathcal{E}}{\partial x^k} \left( o^{Out^1 \rightarrow m}(x, w, f^m), \dots, o^{Out^{out} \rightarrow m}(x, w, f^m) \right) \quad (68)$$

- if  $N^j \notin Out$ :

$$\begin{aligned} \left(\frac{\partial \mathcal{E}}{\partial o^j}\right)^{\rightarrow m}(x, w, f^m) = & \\ \sum_{N^k \in S(j)} \left(\frac{\partial \mathcal{E}}{\partial o^k}\right)^{\rightarrow m}(x, w, f^m) \frac{\partial N^k}{\partial x^{r(j,k)}} \left( o^{P(k)^1 \rightarrow m}(x, w, f^m), \dots, o^{P(k)^{p^k} \rightarrow m}(x, w, f^m), w^k \right) & \end{aligned} \quad (69)$$

The corresponding local equation is (when  $N^m \notin S^+(j)$ ):

$$\frac{\partial^2 \mathcal{E}}{\partial o_t^j \partial w^m} = \frac{\partial}{\partial o^m} \left( \frac{\partial \mathcal{E}}{\partial o_t^j} \right)^{\rightarrow m}(x, w, o^m(x, w)) \frac{\partial N^m}{\partial w} (i^m(x, w)) \quad (70)$$

The differentiation gives:

- if  $N^j = Out^k$ :

$$\frac{\partial}{\partial o^m} \left( \frac{\partial \mathcal{E}}{\partial o_t^j} \right)^{\rightarrow m}(x, w, o^m(x, w)) = \sum_{N^r \in Out \cap S^*(m)} \frac{\partial^2 \mathcal{E}}{\partial x_t^k \partial x^{Out(r)}}(G(x, w)) \frac{\partial o^r}{\partial o^m}(x, w) \quad (71)$$

- if  $N^j \notin Out$ :

$$\begin{aligned} \frac{\partial}{\partial o_v^m} \left( \frac{\partial \mathcal{E}}{\partial o_t^j} \right)^{\rightarrow m}(x, w, o^m(x, w)) = & \\ \sum_{N^k \in S(j)} \sum_{q=1}^{n^k} \left( \frac{\partial \mathcal{E}}{\partial o_q^k}(x, w) \sum_{N^r \in P(k) \cap S^*(m)} \frac{\partial^2 N_q^k}{\partial x_t^{r(j,k)} \partial x^{r(r,k)}}(i^k(x, w)) \frac{\partial o^r}{\partial o_v^m}(x, w) \right. & \\ \left. + \frac{\partial}{\partial o_v^m} \left( \frac{\partial \mathcal{E}}{\partial o_q^k} \right)^{\rightarrow m}(x, w, o^m(x, w)) \frac{\partial N_q^k}{\partial x_t^{r(j,k)}}(i^k(x, w)) \right) & \end{aligned} \quad (72)$$

It is easy to show that property 9 can be extended to include the case of “ $N^l = \mathcal{E}$ ”. The consequence is that the following equation holds as long as  $N^m \notin S^*(N^j)$  and  $N^j \notin S^*(N^m)$ :

$$\frac{\partial}{\partial o_v^m} \left( \frac{\partial \mathcal{E}}{\partial o_q^j} \right)^{\rightarrow m}(x, w, o^m(x, w)) = \frac{\partial}{\partial o_q^j} \left( \frac{\partial \mathcal{E}}{\partial o_v^m} \right)^{\rightarrow j}(x, w, o^j(x, w)) \quad (73)$$

## 6 Complexity

In this section, we compute the theoretical time needed to compute the second order differentials of the output of a node with respect to the parameters of its (generalized) predecessors. As the different algorithms are based on recursive equations, we shall first introduce some general assumptions about the way the computation time will be evaluated:

- If  $a$  is obtained from  $(b_1, \dots, b_n)$  with the equation  $f$  (i.e.,  $a = f(b_1, \dots, b_n)$ ), the time needed to compute  $a$  is made of two parts: an *equation part* and a *variable part*:

- The *equation part* takes only into account the time needed to evaluate  $f$  assuming that the  $b_j$  are already known. For instance, is  $a = b_1 + b_2$ , the equation part of this calculation is 1 (i.e., one complex operation).
- The *variable part* takes only into account the time needed to evaluate the  $b_j$ .
- The different Hessian calculation equations use first order differentials of the node outputs and node Hessian matrices: the computation cost of these values is not taken into account in this complexity evaluation as the required values are nearly the same for the different algorithms. We can therefore compare the differentiation algorithms without them.

We introduce here some notations that allow to simplify the complexity formulae:

- when  $A \subset \mathcal{N}$ ,  $|I_A^l| = \sum_{N^j \in P(l) \cap A} |O^j|$ . If  $P(l) \subset A$ , we have  $|I_A^l| = |I^l|$ . If  $P(l) \cap A = \emptyset$ , then  $|I_A^l| = 0$ ;
- when  $A$  and  $B$  are two subsets of  $\mathcal{N}$ ,  $|O_B^A| = \sum_{N^j \in A \cap B} |O^j|$ , with  $|O_B^A| = 0$  when  $A \cap B = \emptyset$ ;
- when  $A \subset \mathcal{N}$ ,  $|O_A| = \sum_{N^j \in Out \cap A} |O^j|$ , with  $|O_A| = 0$  when  $Out \cap A = \emptyset$ ;
- when  $\mathcal{P}$  is a predicate,  $\delta_{\mathcal{P}} = 1$  means that  $\mathcal{P}$  is true and  $\delta_{\mathcal{P}} = 0$  means that  $\mathcal{P}$  is false. For instance  $\delta_{i=j} = 1$  is equivalent to  $i = j$ .

As the considered functions are  $C^2$ , the differentiation order for a second differential does not change the result: we have for instance  $\frac{\partial^2 o_i^l}{\partial w_i^j \partial w_u^m} = \frac{\partial^2 o_i^l}{\partial w_u^m \partial w_i^j}$ . As the formulae derived for the different methods are not symmetric, their total costs depend on the chosen differentiation order. The purpose of this section is also to explain how to choose this differentiation order so that the processing time is minimized.

Finally, we have introduced in the previous sections precise formulae for the differentials. These formulae take into account the fact that some differentials are equal to zero (for instance  $\frac{\partial o_i^l}{\partial o^k}(x, w)$  when  $N^k \notin P^*(l)$ ). Of course, in the complexity evaluation, these facts will be taken into account, but in order to simplify the presentation, the differentiation formulae will be recalled in simplified version in which all differentials are used, even if they are equal to zero. For instance, we will write :

$$\frac{\partial^2 N_i^l}{\partial x_q^k \partial w_u^m} = \sum_{r=1}^{p^l} \frac{\partial^2 N_i^l}{\partial x_q^k \partial x^r} \frac{\partial o^{P(l)r}}{\partial w_u^m},$$

in spite of:

$$\frac{\partial^2 N_i^l}{\partial x_q^k \partial w_u^m} = \sum_{N^r \in P(l) \cap S^*(m)} \frac{\partial^2 N_i^l}{\partial x_q^k \partial x^{r(l)}} \frac{\partial o^r}{\partial w_u^m}$$

Both equations are equivalent, but the first one is simpler to write even if it does not show directly that some differentials are equal to zero.

## 6.1 The direct method

### 6.1.1 Equation part

The direct method is based on equation 39. As we have:

$$(6.1.i) \quad \frac{\partial^2 N_i^l}{\partial x_q^k \partial w_u^m} = \sum_{r=1}^{p^l} \frac{\partial^2 N_i^l}{\partial x_q^k \partial x^r} \frac{\partial o^{P(l)r}}{\partial w_u^m},$$

equation 39 can be simplified into:

$$(6.1.ii) \quad \frac{\partial^2 o_i^l}{\partial w_i^j \partial w_u^m} = \sum_{N^k \in P(l)} \sum_{q=1}^{n^k} \left( \frac{\partial o_q^k}{\partial w_i^j} \frac{\partial^2 N_i^l}{\partial x_q^{r(k,l)} \partial w_u^m} + \frac{\partial N_i^l}{\partial x_q^{r(k,l)}} \frac{\partial^2 o_q^k}{\partial w_i^j \partial w_u^m} \right)$$

For fixed values of  $k$  and  $q$ , we need to compute:

$$(6.1.iii) \quad \frac{\partial o_q^k}{\partial w_t^j} \frac{\partial^2 N_i^l}{\partial x_q^{r(k,l)} \partial w_u^m} + \frac{\partial N_i^l}{\partial x_q^{r(k,l)}} \frac{\partial^2 o_q^k}{\partial w_t^j \partial w_u^m}$$

As  $N^k \in P(l)$ ,  $\frac{\partial N_i^l}{\partial x_q^{r(k,l)}}$  is a local differential and is always non null. As  $N^m \in P^+(l)$ ,  $\frac{\partial^2 N_i^l}{\partial x_q^{r(k,l)} \partial w_u^m}$  is also non null. The first order differential  $\frac{\partial o_q^k}{\partial w_t^j}$  is non null when  $N^k \in P(l) \cap S^*(m)$ . In this case, one multiplication is needed to compute  $\frac{\partial o_q^k}{\partial w_t^j} \frac{\partial^2 N_i^l}{\partial x_q^{r(k,l)} \partial w_u^m}$ . For similar reasons, when  $N^k \in P(l) \cap S^*(m) \cap S^*(j)$ , one multiplication is needed to compute  $\frac{\partial N_i^l}{\partial x_q^{r(k,l)}} \frac{\partial^2 o_q^k}{\partial w_t^j \partial w_u^m}$ . In the case, evaluating equation 6.1.iii implies also one addition. Finally, equation 6.1.iii has the following cost:

- 0 when  $N^k \in P(l)$  and  $N^k \notin S^*(m)$ ;
- 1 when  $N^k \in P(l) \cap S^*(m)$  and  $N^k \notin S^*(j)$ ;
- 3 when  $N^k \in P(l) \cap S^*(m) \cap S^*(j)$ .

The obtained cost does not depend on  $q$  which ranges from 1 to  $|O^k|$ . Therefore, the obtained cost must be multiply by  $|O^k|$ . Moreover, for a given  $k$ , the following equation:

$$(6.1.iv) \quad \sum_{q=1}^{n^k} \left( \frac{\partial o_q^k}{\partial w_t^j} \frac{\partial^2 N_i^l}{\partial x_q^{r(k,l)} \partial w_u^m} + \frac{\partial N_i^l}{\partial x_q^{r(k,l)}} \frac{\partial^2 o_q^k}{\partial w_t^j \partial w_u^m} \right)$$

is obtained by summing the partial result which are non null when  $N^k \in P(l) \cap S^*(m)$ . As there are  $|O^k|$  values to sum, we use  $|O^k| - 1$  additions, and the *equation cost* of equation 6.1.iv is:

- 0 when  $N^k \in P(l)$  and  $N^k \notin S^*(m)$ ;
- $2|O^k| - 1$  when  $N^k \in P(l) \cap S^*(m)$  and  $N^k \notin S^*(j)$ ;
- $4|O^k| - 1$  when  $N^k \in P(l) \cap S^*(m) \cap S^*(j)$ .

The result of equation 6.1.iv is non null when  $N^k \in P(l) \cap S^*(m)$ . But the result of equation 6.1.ii is obtained by summing for  $N^k \in P(l)$  the equation 6.1.iv results. Therefore, this introduces  $|P(l) \cap S^*(m)| - 1$  additions. Therefore, the *equation cost* of equation 6.1.ii is:

$$|P(l) \cap S^*(m)| - 1 + \sum_{N^k \in (P(l) \cap S^*(m)) - S^*(j)} (2|O^k| - 1) + \sum_{N^k \in P(l) \cap S^*(m) \cap S^*(j)} (4|O^k| - 1)$$

We can easily group together the two sums and take out of the sums the  $-1$ . We obtain:

$$2 \sum_{N^k \in P(l) \cap S^*(m)} |O^k| + 2 \sum_{N^k \in P(l) \cap S^*(m) \cap S^*(j)} |O^k| - 1,$$

which can be rewritten in:

$$De_i^l(w_t^j, w_u^m) = 2|I_{S^*(m)}^l| + 2|I_{S^*(m) \cap S^*(j)}^l| - 1 \quad (74)$$

The operations implied by equation 6.1.ii do not depend on  $t$  and  $u$  values, therefore the complexity does not depend on these values and we have:

$$De_i^l(w^j, w^m) = |W^j| |W^m| \left( 2|I_{S^*(m)}^l| + 2|I_{S^*(m) \cap S^*(j)}^l| - 1 \right) \quad (75)$$

We have modified equation 39 in order to obtain equation 6.1.ii. The main idea of this simplification is that a part of the equation is in fact equal to  $\frac{\partial^2 N_i^l}{\partial x_q^k \partial w_u^m}$ . But this value can be computed with equation 6.1.i and *does not depend* on  $N^j$ . Therefore, this value can be considered as an additional differential which is calculated before computing  $o_i^l$  Hessian matrix. As equation 6.1.i is based only on local second order differentials and first order differentials, its total cost is equal to its *equation cost*. Its value is obtained with the help of the following value (where  $N^r$  is a predecessor of  $N^l$ ):

$$(6.1.v) \quad \frac{\partial^2 N_i^l}{\partial x_q^k \partial x^{r(l)}} \frac{\partial o^r}{\partial w_u^m}$$

This calculation is a simple scalar product and implies therefore  $2|O^r| - 1$  but only when  $N^m \in P^*(r)$ . Therefore the cost is:

$$\sum_{N^r \in P(l) \cap S^*(m)} (2|O^r| - 1)$$

Moreover, we must add the results obtained from equation 6.1.v. As there is  $|P(l) \cap S^*(m)|$  values to sum, this introduces  $|P(l) \cap S^*(m)| - 1$  operations. Finally, the equation cost for equation 6.1.i is:

$$De_i^l(o_q^k, w_u^m) = 2|I_{S^*(m)}^l| - 1 \quad (76)$$

Once again, the operations implied by equation 6.1.i do not depend on  $q$  and  $u$ . Therefore we have:

$$De_i^l(o^k, w^m) = |O^k| |W^m| \left( 2|I_{S^*(m)}^l| - 1 \right) \quad (77)$$

The second order differential  $\frac{\partial^2 N_i^l}{\partial x_q^k \partial w_u^m}$  must be computed for each  $N^k \in P(l)$  (and for each  $N^m \in P^+(l)$ ). In equation 6.1.ii, the differential is used only when  $N^k \in P(l) \cap S^*(j)$ , but as it does not depend on  $N^j$ , we can choose an arbitrary  $N^j \in P^*(N^k)$  in order to have to use  $\frac{\partial^2 N_i^l}{\partial x_q^k \partial w_u^m}$ .

The obtained equation cost for equation 6.1.i is in fact a part of the *variable cost* of equation 6.1.ii.

The particular cases presented in subsection 4.1.2 introduces different equation costs. In fact, all equations but equation 40 have null equation cost as they are simple equalities. Moreover, equation 40 is exactly similar to equation 6.1.i: the calculation is the same and the only difference is that in spite of using the local differential  $\frac{\partial^2 N_i^l}{\partial x_q^k \partial x^r}$ , it uses  $\frac{\partial^2 N_i^l}{\partial w_t^l \partial x^r}$ . Therefore, equation cost of equation 40 is equal to equation cost of equation 6.1.i and we have (when  $m \neq l$ ):

$$De_i^l(w_t^l, w_u^m) = 2|I_{S^*(m)}^l| - 1 \quad (78)$$

Once again, the operations implied by equation 40 do not depend on  $t$  and  $u$ . Therefore we have:

$$De_i^l(w^l, w^m) = |W^l| |W^m| \left( 2|I_{S^*(m)}^l| - 1 \right) \quad (79)$$

### 6.1.2 Variable part

With the help of the transformation of equation 39 into equation 6.1.ii, we have changed the “variables” needed to compute  $\frac{\partial^2 o_i^l}{\partial w_t^j \partial w_u^m}$ . In equation 6.1.ii, we need:

- $\frac{\partial o_q^k}{\partial w_t^j}$ : this is a first order derivative and its computation time is not included into the variable cost;
- $\frac{\partial^2 N_i^l}{\partial x_q^{r(k,l)} \partial w_u^m}$ : this differential is computed with the help of equation 6.1.i, but as its value does not depend on  $N^j$ , the total complexity of its computation must be directly included in the total complexity of the  $o_i^l$  Hessian computation;
- $\frac{\partial N_i^l}{\partial x_q^{r(k,l)}}$ : the computation cost of this local first order differential is not included in the variable cost;

- $\frac{\partial^2 o_q^k}{\partial w_t^j \partial w_u^m}$ : this value is a part of  $o_q^k$  Hessian matrix. Its cost must be included into the variable cost.

The “special” case of  $\frac{\partial^2 N_i^l}{\partial x_q^{r(k,l)} \partial w_u^m}$  implies that it is not possible to evaluate a partial cost for  $o_i^l$  Hessian matrix. Therefore, it is also impossible to evaluate a variable cost for  $\frac{\partial^2 o_i^l}{\partial w_t^j \partial w_u^m}$ , because this cost must include a partial cost for  $o_q^k$  Hessian matrix.

Computing  $\frac{\partial^2 N_i^l}{\partial x_q^{r(k,l)} \partial w_u^m}$  with equation 6.1.i does not imply a variable cost as we only need the following values:

- $\frac{\partial^2 N_i^l}{\partial x_q^k \partial x^r}$ : the computation cost of this local second order differential is not included into the variable cost;
- $\frac{\partial o^{P(l)r}}{\partial w_u^m}$ : the computation cost of this first order differential is not included into the variable cost.

Therefore, equation 6.1.i total cost is:

$$D_i^l(o^k, w^m) = De_i^l(o^k, w^m) \quad (80)$$

Finally, all particular case formulae (presented in subsection 4.1.2) have no variable cost as they use only first order differentials or local second order differentials. Therefore:

$$D_i^l(w^l, w^m) = De_i^l(w^l, w^m) \quad (81)$$

### 6.1.3 Differentiation order

For the direct method, the variable cost is symmetric: indeed, this variable cost is the consequence of recursive equation 39. But the only requirement of this equation is the Hessian matrices of  $N^l$  predecessors. Therefore, choosing a particular differentiation order (e.g.,  $\frac{\partial^2 o_i^l}{\partial w_t^j \partial w_u^m}$ ) does not imply to keep this order for  $N^l$  predecessors. Of course, we need for instance  $\frac{\partial^2 o_i^{P(l)1}}{\partial w_t^j \partial w_u^m}$ , but as the considered functions are  $C^2$ , this differential can be computed with the alternate order.

As the formulae of the direct method are not symmetric, their *equation costs* depend on the differentiation order. When  $m \neq l$  and  $j \neq l$ , equation 6.1.i has  $De_i^l(w^j, w^m)$  for equation cost when we choose to differentiate first with respect to  $w^j$  and second with respect to  $w^m$ . If we choose the alternate differentiation order, the equation cost is now  $De_i^l(w^m, w^j)$ . In general,  $De_i^l(w^j, w^m) \neq De_i^l(w^m, w^j)$  and therefore, the equation cost can be optimized by choosing the most efficient order. The best cost is:

$$Do_i^l(w^j, w^m) = Do_i^l(w^m, w^j) = \min (De_i^l(w^j, w^m), De_i^l(w^m, w^j))$$

The values of  $De_i^l(w^j, w^m)$  and  $De_i^l(w^m, w^j)$  allow to simplify this equation and we have:

$$Do_i^l(w^j, w^m) = |W^j| |W^m| \left( 2|I_{S^*(m) \cap S^*(j)}^l| - 1 + 2 \min (|I_{S^*(m)}^l|, |I_{S^*(j)}^l|) \right) \quad (82)$$

The additional value  $\frac{\partial^2 N_i^l}{\partial x_q^{r(k,l)} \partial w_u^m}$  needed by equation 6.1.i does not imply any order problem as  $o_q^{r(k,l)}$  is a local variable.

When  $N^m = N^l$  or  $N^j = N^l$ , the formulae for  $\frac{\partial^2 o_i^l}{\partial w_t^j \partial w_u^m}$  and  $\frac{\partial^2 o_i^l}{\partial w_u^m \partial w_t^j}$  are exactly the same and therefore the complexity does not depend on the differentiation order. Therefore we have:

$$Do_i^l(w^l, w^j) = Do_i^l(w^j, w^l) = De_i^l(w^j, w^l) = |W^l| |W^j| \left( 2|I_{S^*(j)}^l| - 1 \right) \quad (83)$$

#### 6.1.4 Total cost

For a given  $N^m \in P^+(l)$ ,  $\frac{\partial^2 N_i^l}{\partial x_q^{r(k,l)} \partial w_u^m}$  as to be computed for each  $N^k \in P(l)$ . This introduces the following cost:

$$\sum_{N^k \in P(l)} D_i^l(o^k, w^m) = |W^m| \left( 2|I_{S^*(m)}^l| - 1 \right) \sum_{N^k \in P(l)} |O^k| = |I^l| |W^m| \left( 2|I_{S^*(m)}^l| - 1 \right)$$

Therefore, the total computation cost implied by  $\frac{\partial^2 N_i^l}{\partial x_q^{r(k,l)} \partial w_u^m}$  for  $o_i^l$  Hessian matrix is:

$$(6.1.vi) \quad |I^l| \sum_{N^m \in P^+(l)} |W^m| \left( 2|I_{S^*(m)}^l| - 1 \right)$$

In order to compute  $o_i^l$  Hessian, we need to compute  $\frac{\partial^2 o_i^l}{\partial w_i^l \partial w_u^m}$  for each  $N^m \in P^+(l)$ . As this calculation has no variable cost, its total cost is:

$$(6.1.vii) \quad \sum_{N^m \in P^+(l)} D o_i^l(w^l, w^m) = |W^l| \sum_{N^m \in P^+(l)} |W^m| \left( 2|I_{S^*(m)}^l| - 1 \right)$$

In order to compute  $o_i^l$  Hessian, we need to compute  $\frac{\partial^2 o_i^l}{\partial w_i^l \partial w_u^m}$  for each  $N^m \in P^+(l)$  and for each  $N^j \in P^+(l)$ . This introduces the following *equation cost*:

$$\sum_{N^j \in P^+(l)} \left( D o_i^l(w^j, w^j) + \frac{1}{2} \sum_{N^m \in P^+(l), m \neq j} D o_i^l(w^j, w^m) \right)$$

This equation can be simplified into:

$$\sum_{N^j \in P^+(l)} |W^j| \left( \sum_{N^m \in P^+(l), m \neq j} |W^m| \left( \min(|I_{S^*(m)}^l|, |I_{S^*(j)}^l|) + |I_{S^*(j) \cap S^*(m)}^l| - \frac{1}{2} \right) + |W^j| \left( 4|I_{S^*(j)}^l| - 1 \right) \right)$$

A last simplification gives:

$$(6.1.viii) \quad \sum_{N^j \in P^+(l)} |W^j| \left( \sum_{N^m \in P^+(l)} |W^m| \left( \min(|I_{S^*(m)}^l|, |I_{S^*(j)}^l|) + |I_{S^*(j) \cap S^*(m)}^l| - \frac{1}{2} \right) + |W^j| \left( 2|I_{S^*(j)}^l| - \frac{1}{2} \right) \right)$$

Let us now compute the complete *equation time* needed for the computation of  $o_i^l$  Hessian. To the previously obtained cost, we must add the *total time* of  $\frac{\partial^2 N_i^l}{\partial x_q^{r(k,l)} \partial w_u^m}$  computation, which is given by equation 6.1.vi. Finally, the complete *equation time* needed for the computation of  $o_i^l$  Hessian is:

$$\mathcal{D}^l = \sum_{N^j \in P^+(l)} |W^j| \left( \sum_{N^m \in P^+(l)} |W^m| \left( \min(|I_{S^*(m)}^l|, |I_{S^*(j)}^l|) + |I_{S^*(j) \cap S^*(m)}^l| - \frac{1}{2} \right) + |W^j| \left( 2|I_{S^*(j)}^l| - \frac{1}{2} \right) + (|I^l| + |W^l|) \left( 2|I_{S^*(j)}^l| - 1 \right) \right) \quad (84)$$

This last cost does not depend on  $i$  (that's why the chosen name for it does not recall the  $i$ ).

Let us call  $H_d^l$  the time needed to compute  $o_i^l$  Hessian for all  $i$  in  $[1, n^l]$ . This complexity is the *total time* needed for the computation, including the variable cost of each equation. As explained in subsection 6.1.2, the *variable time* of  $o_i^l$  direct computation equations is the *total time* needed to compute  $o_q^k$  Hessian matrices where  $N^k \in P(l)$  (and  $q \in [1, n^k]$ ) and therefore is equal to:

$$\sum_{N^k \in P(l)} H_d^k$$

But this *variable time* must be included only one time for all  $i$ , because the needed Hessian matrices do not depend on  $i$  and can be computed only once and reused for each  $i \in [1, n^l]$ . Each time these values are used, the complete *equation time* is spent. Therefore, we have the following recursive relationship between the *total time* needed to compute the various Hessian matrices :

$$H_d^l = \sum_{N^k \in P(l)} H_d^k + |O^l| \mathcal{D}^l \quad (85)$$

Of course, when  $N^l \in In$ ,  $H_d^l = 0$ .

## 6.2 The hybrid method

### 6.2.1 Equation part

The hybrid algorithm is based on several different formulae: two local equations (45 and 47) and two recursive equations (48 and 49).

1. equation 45 (i.e., local equation when  $N^m \neq N^j$  and  $N^j \neq N^l$ ):

As we have:

$$(6.2.i) \quad \frac{\partial^2 N_q^j}{\partial w_t^j \partial w_u^m} = \sum_{r=1}^{p^l} \frac{\partial^2 N_q^j}{\partial w_t^j \partial x^r} \frac{\partial o^{P(j)r}}{\partial w_u^m},$$

equation 45 can be simplified into:

$$(6.2.ii) \quad \frac{\partial^2 o_i^l}{\partial w_t^j \partial w_u^m} = \sum_{q=1}^{n^j} \left( \frac{\partial o_i^l}{\partial o_q^j} \frac{\partial^2 N_q^j}{\partial w_t^j \partial w_u^m} + \frac{\partial^2 o_i^l}{\partial o_q^j \partial w_u^m} \frac{\partial N_q^j}{\partial w_t^j} \right)$$

For a fixed value of  $q$ , we have to compute:

$$(6.2.iii) \quad \frac{\partial o_i^l}{\partial o_q^j} \frac{\partial^2 N_q^j}{\partial w_t^j \partial w_u^m} + \frac{\partial^2 o_i^l}{\partial o_q^j \partial w_u^m} \frac{\partial N_q^j}{\partial w_t^j}$$

As  $N^j \in P^+(l)$ ,  $\frac{\partial o_i^l}{\partial o_q^j}$  is non null. When  $N^m \in P^+(j)$ ,  $\frac{\partial^2 N_q^j}{\partial w_t^j \partial w_u^m}$  is also non null. In this case, one product is needed to compute  $\frac{\partial o_i^l}{\partial o_q^j} \frac{\partial^2 N_q^j}{\partial w_t^j \partial w_u^m}$ . As  $N^j \in P^+(l)$  and  $N^m \in P^+(l)$ ,  $\frac{\partial^2 o_i^l}{\partial o_q^j \partial w_u^m}$  is always non null. Moreover  $\frac{\partial N_q^j}{\partial w_t^j}$  is also non null and one product is needed to compute  $\frac{\partial^2 o_i^l}{\partial o_q^j \partial w_u^m} \frac{\partial N_q^j}{\partial w_t^j}$ . When  $N^m \in P^+(j)$ , one addition is also needed to compute equation 6.2.iii. Finally, this equation as the following cost:

- 1 when  $N^m \notin P^+(j)$ ;
- 3 when  $N^m \in P^+(j)$ .

The obtained cost does not depend on  $q$  which ranges from 1 to  $|O^j|$ . Therefore, the obtained cost must be multiplied by  $|O^j|$ . Moreover, equation 6.2.ii is obtained by summing equation 6.2.iii results for the different values of  $q$ . As there are  $|O^j|$  values to sum, this introduces  $|O^j| - 1$  additions. Therefore, the *equation cost* of equation 6.2.ii is:

- when  $N^m \notin P^+(j)$ :

$$(6.2.iv) \quad He_i^l(w_t^j, w_u^m) = 2|O^j| - 1$$

- when  $N^m \in P^+(j)$ :

$$(6.2.v) \quad He_i^l(w_t^j, w_u^m) = 4|O^j| - 1$$

The  $\delta$  notation allows to simplify the obtained equation and we have for the general case:

$$He_i^l(w_t^j, w_u^m) = 2(1 + \delta_{N^m \in P^+(j)})|O^j| - 1 \quad (86)$$

As equation 6.2.ii does not depend on  $t$  or  $u$ , the “total” *equation cost* of this equation for all  $t$  and  $u$  is:

$$He_i^l(w^j, w^m) = |W^j||W^m|(2(1 + \delta_{N^m \in P^+(j)})|O^j| - 1) \quad (87)$$

The computation of  $\frac{\partial^2 N_q^j}{\partial w_t^j \partial w_u^m}$  is based on a sum of the following values (for  $N^r \in P(j)$ ):

$$(6.2.vi) \quad \frac{\partial^2 N_q^j}{\partial w_t^j \partial x^{r(r,j)}} \frac{\partial o^r}{\partial w_u^m}$$

As a local differential,  $\frac{\partial^2 N_q^j}{\partial w_t^j \partial x^{r(r,j)}}$  is assumed to be non null. When  $N^r \in S^*(m)$ ,  $\frac{\partial o^r}{\partial w_u^m}$  is also non null and formula 6.2.vi is a simple scalar product which implies  $2|O^r| - 1$  operations. Therefore, as this value is computed for each  $N^r \in P(j) \cap S^*(m)$ , we have a total cost of:

$$\sum_{N^r \in P(j) \cap S^*(m)} (2|O^r| - 1)$$

Moreover, we must add the values of formula 6.2.vi in order to obtain  $\frac{\partial^2 N_q^j}{\partial w_t^j \partial w_u^m}$  with equation 6.2.i. This introduces  $|P(j) \cap S^*(m)| - 1$  additions. Therefore, the *equation cost* of equation 6.2.i is:

$$(6.2.vii) \quad 2 \sum_{N^r \in P(j) \cap S^*(m)} |O^r| - 1 = 2|I_{S^*(m)}^j| - 1$$

It is important to notice that this value is valid only if  $N^m \in P^+(j)$ . If this is not the case, the complexity is null. Moreover,  $\frac{\partial^2 N_q^j}{\partial w_t^j \partial w_u^m}$  must be computed for each  $q$  and therefore the complexity must be multiplied by  $|O^j|$ . The obtained differential,  $\frac{\partial^2 N_q^j}{\partial w_t^j \partial w_u^m}$  does not depend on  $i$  and therefore its computation can be separated from equation 6.2.ii. Its final computation cost (for all values of  $t$  and  $u$ ):

$$HeN_q^j(w^j, w^m) = \delta_{N^m \in P^+(j)}|W^j||W^m|(2|I_{S^*(m)}^j| - 1) \quad (88)$$

Finally, the *equation cost* of equation 45 for all values of  $t$  and  $u$  is therefore:

$$\mathcal{H}e_i^l(w^j, w^m) = He_i^l(w^j, w^m) + |O^j|HeN_q^j(w^j, w^m) \quad (89)$$

We must notice that the second part of this cost covers calculations that do not depend on  $i$ .

2. equation 47 (i.e., local equation when  $N^m = N^j$ ):

This equation is a sum of the following terms:

$$(6.2.viii) \quad \frac{\partial^2 o_i^l}{\partial \sigma_t^j \partial w_u^j} \frac{\partial N_q^j}{\partial w_t^j} + \frac{\partial o_i^l}{\partial \sigma_q^j} \frac{\partial^2 N_q^j}{\partial w_t^j \partial w_u^j}$$

As  $N^j \in P^+(l)$ , each partial differential of this formula is non null. Therefore, for a fixed  $q$ , equation 6.2.viii introduces 3 operations. This complexity does not depend on  $q$ , and therefore, the total amount of operations is  $3|O^j|$ .

Moreover, equation 47 sums the partial results and, as there is  $|O^j|$  results to sum, introduces  $|O^j| - 1$  operations. Finally, the *equation cost* of equation 47 is:

$$\mathcal{H}e_i^l(w_t^j, w_u^j) = 4|O^j| - 1, \quad (90)$$

and therefore:

$$\mathcal{H}e_i^l(w^j, w^j) = |W^j|^2 (4|O^j| - 1) \quad (91)$$

3. equation 48 (i.e., recursive equation when  $N^m \notin S(j)$ ):

As for the other equations, we can simplify equation 48. In fact, as we have:

$$(6.2.ix) \quad \frac{\partial^2 N_q^k}{\partial x_t^{r(j,k)} \partial w_u^m} = \sum_{r=1}^{p^k} \frac{\partial^2 N_q^k}{\partial x_t^{r(j,k)} \partial x^r} \frac{\partial o^{P(k)r}}{\partial w_u^m},$$

equation 48 can be rewritten into:

$$(6.2.x) \quad \frac{\partial^2 o_i^l}{\partial o_i^j \partial w_u^m} = \sum_{N^k \in S(j)} \sum_{q=1}^{n^k} \left( \frac{\partial o_i^l}{\partial o_q^k} \frac{\partial^2 N_q^k}{\partial x_t^{r(j,k)} \partial w_u^m} + \frac{\partial^2 o_i^l}{\partial o_q^k \partial w_u^m} \frac{\partial N_q^k}{\partial x_t^{r(j,k)}} \right)$$

Therefore, for fixed value of  $N^k$  we shall compute:

$$(6.2.xi) \quad \sum_{q=1}^{n^k} \left( \frac{\partial o_i^l}{\partial o_q^k} \frac{\partial^2 N_q^k}{\partial x_t^{r(j,k)} \partial w_u^m} + \frac{\partial^2 o_i^l}{\partial o_q^k \partial w_u^m} \frac{\partial N_q^k}{\partial x_t^{r(j,k)}} \right),$$

and for fixed value of  $q$ :

$$(6.2.xii) \quad \frac{\partial o_i^l}{\partial o_q^k} \frac{\partial^2 N_q^k}{\partial x_t^{r(j,k)} \partial w_u^m} + \frac{\partial^2 o_i^l}{\partial o_q^k \partial w_u^m} \frac{\partial N_q^k}{\partial x_t^{r(j,k)}}$$

The partial differential  $\frac{\partial N_q^k}{\partial x_t^{r(j,k)}}$  is a local value and can be assumed to remain non null. When  $N^k \in P^+(l)$ , we have  $\frac{\partial o_i^l}{\partial o_q^k} \neq 0$ , but when  $N^k = N^l$ ,  $\frac{\partial o_i^l}{\partial o_q^k} = \delta_{i=q}$ . Therefore, when  $N^k \in P^+(l)$  and  $N^m \in P^+(l)$ ,  $\frac{\partial^2 o_i^l}{\partial o_q^k \partial w_u^m}$  is non null but  $\frac{\partial^2 o_i^l}{\partial o_q^k \partial w_u^m} = 0$ . Finally,  $\frac{\partial^2 N_q^k}{\partial x_t^{r(j,k)} \partial w_u^m}$  is non zero when  $N^m \in P^+(k)$ . We have therefore the following different cases:

- if  $N^m \notin P^+(k)$ :
  - if  $N^k \in P^+(l)$ , one multiplication is needed to compute  $\frac{\partial^2 o_i^l}{\partial o_q^k \partial w_u^m} \frac{\partial N_q^k}{\partial x_t^{r(j,k)}}$ ;
  - if  $N^k \notin P^+(l)$ , equation 6.2.xii is null.
- if  $N^m \in P^+(k)$ :
  - if  $N^k \in P^+(l)$ , no partial differential is null and therefore 3 operations are needed to compute equation 6.2.xii;
  - if  $N^k = N^l$ , we need no operation at all, but the result of equation 6.2.xii is non null when  $q = i$  (it is  $\frac{\partial^2 N_i^k}{\partial x_t^{r(j,k)} \partial w_u^m}$ );
  - if  $N^k \notin P^+(l)$ , equation 6.2.xii is null.

For a given  $k$ , the results of equation 6.2.xii for  $q$  ranging from 1 to  $|O^k|$  are summed. If  $N^k \neq N^l$ , the complexity of equation 6.2.xii does not depend on  $q$ , and therefore the total complexity is obtained by multiplying the results by  $|O^k|$ . Moreover, the summing process introduces  $|O^k| - 1$  additional operations. When  $N^k = N^l$  (and  $N^m \in P^+(k)$ ), equation 6.2.xii has a null result except for  $q = i$ . Therefore, we have nothing to sum and the complexity remains null (because computing equation 6.2.xii result for  $i = q$  does not introduce any operation). Finally, we have the following costs:

- if  $N^m \notin P^+(k)$ :
  - if  $N^k \in P^+(l)$ :  $2|O^k| - 1$ ;
  - if  $N^k \notin P^+(l)$ : 0 (equation 6.2.xi result is null).
- if  $N^m \in P^+(k)$ :
  - if  $N^k \in P^+(l)$ :  $4|O^k| - 1$ ;
  - if  $N^k \notin P^+(l)$ : 0 (when  $N^k \neq N^l$ , equation 6.2.xi result is null but when  $N^k = N^l$ , it is not).

The results of equation 6.2.xi have to be summed in order to obtain equation 6.2.xii. There is exactly  $|P^+(l) \cap S(j)|$  terms to add when  $S^+(m) \cap S(j) = \emptyset$  and  $|P^*(l) \cap S(j)|$  terms in the other case. Therefore, the *equation cost* of equation 6.2.xii is:

$$(6.2.xiii) \quad He_i^l(o_t^j, w_u^m) = \delta_{S(j) \cap P^+(l) - S^+(m) \neq \emptyset} \sum_{N^k \in S(j) \cap P^+(l), N^k \notin S^+(m)} (2|O^k| - 1) \\ + \delta_{S(j) \cap P^+(l) \cap S^+(m) \neq \emptyset} \sum_{N^k \in S(j) \cap P^+(l) \cap S^+(m)} (4|O^k| - 1) \\ + |S(j) \cap P^+(l)| + \delta_{S^+(m) \cap S(j) \neq \emptyset} \delta_{N^l \in S(j)} - 1$$

When  $N^l \in S(j)$ ,  $S^+(m) \cap S(j) \neq \emptyset$  because  $N^m \in P^+(l)$ . Therefore  $\delta_{S^+(m) \cap S(j) \neq \emptyset} \delta_{N^l \in S(j)} = \delta_{N^l \in S(j)}$ . Moreover,  $|O_B^A| = 0$  when  $B = \emptyset$ , therefore  $\delta_{B \neq \emptyset} |O_B^A| = |O_B^A|$ . The different sums of equation 6.2.xiii can be simplified and we obtain:

$$He_i^l(o_t^j, w_u^m) = 2|O_{P^+(l)}^{S(j)}| + 2|O_{P^+(l) \cap S^+(m)}^{S(j)}| + \delta_{N^l \in S(j)} - 1 \quad (92)$$

Therefore, for all values of  $t$  and  $u$ , we have:

$$He_i^l(o^j, w^m) = |O^j| |W^m| \left( 2|O_{P^+(l)}^{S(j)}| + 2|O_{P^+(l) \cap S^+(m)}^{S(j)}| + \delta_{N^l \in S(j)} - 1 \right) \quad (93)$$

The computation of  $\frac{\partial^2 N_q^k}{\partial x_t^{r(j,k)} \partial w_u^m}$  is based on a sum of the following values (for  $N^r \in P(k)$ ):

$$(6.2.xiv) \quad \frac{\partial^2 N_q^k}{\partial x_t^{r(j,k)} \partial x^{r(r,k)} \partial w_u^m} \frac{\partial o^r}{\partial w_u^m}$$

As a local differential,  $\frac{\partial^2 N_q^k}{\partial x_t^{r(j,k)} \partial o^{r(r,k)}}$  can be assumed to remain non null. When  $N^m \in P^*(r)$ ,  $\frac{\partial o^r}{\partial w_u^m}$  is also non null and formula 6.2.xiv is a simple scalar product which complexity is  $2|O^r| - 1$ . Therefore, as equation 6.2.xiv is evaluated for each  $N^r \in P(k) \cap S^*(m)$ , it implies the following cost:

$$\sum_{N^r \in P(k) \cap S^*(m)} (2|O^r| - 1)$$

Moreover, equation 6.2.ix implies to sum the partial results obtained from equation 6.2.xiv, we introduces  $|P(k) \cap S^*(m)| - 1$  operations. Therefore, the *equation cost* of equation 6.2.ix is:

$$(6.2.xv) \quad 2 \sum_{N^r \in P(k) \cap S^*(m)} |O^r| - 1 = 2|I_{S^*(m)}^k| - 1$$

As  $\frac{\partial^2 N_q^k}{\partial x_t^{r(j,k)} \partial w_u^m}$  must be computed for each  $q$ , the complexity must be multiplied by  $|O^k|$ . It is important to notice that the value does not depend on  $i$ . Moreover, it has to be computed only when  $N^k \in S(j) \cap P^*(l) \cap S^+(m)$ . The complete complexity is therefore:

$$HeN_q^k(o^j, w^m) = \delta_{N^k \in P^*(l) \cap S^+(m)} |O^j| |W^m| \left( 2|I_{S^*(m)}^k| - 1 \right) \quad (94)$$

Finally, the *equation cost* of equation 48 for all  $t$  and  $u$  is:

$$\mathcal{H}e_i^l(o^j, w^m) = He_i^l(o^j, w^m) + \sum_{N^k \in S(j)} |O^k| HeN_q^k(o^j, w^m) \quad (95)$$

4. equation 49 (i.e., recursive equation when  $N^m \in S(j)$ ):

Equation 49 is nearly identical to equation 48. With the help of equation 6.2.ix, equation 49 can be simplified into:

$$(6.2.xvi) \quad \frac{\partial^2 o_i^l}{\partial o_t^j \partial w_u^m} = \sum_{N^k \in S(j), k \neq m} \sum_{q=1}^{n^k} \frac{\partial o_i^l}{\partial o_q^k} \frac{\partial^2 N_q^k}{\partial x_t^{r(j,k)} \partial w_u^m} + \sum_{N^k \in S(j)} \sum_{q=1}^{n^k} \frac{\partial^2 o_i^l}{\partial o_q^k \partial w_u^m} \frac{\partial N_q^k}{\partial x_t^{r(j,k)}} + \frac{\partial o_i^l}{\partial \sigma^m} \frac{\partial^2 N^m}{\partial x_t^{r(j,m)} \partial w_u^m}$$

The first part of this equation is:

$$(6.2.xvii) \quad \sum_{N^k \in S(j), k \neq m} \sum_{q=1}^{n^k} \frac{\partial o_i^l}{\partial o_q^k} \frac{\partial^2 N_q^k}{\partial x_t^{r(j,k)} \partial w_u^m}$$

A very similar equation was studied as part of equation 48. It is easy to show that the *equation cost* of equation 6.2.xvii is given by the following formulae:

- when  $S^+(m) \cap S(j) \cap P^+(l) = \emptyset$ : the cost is 0;
- when  $S^+(m) \cap S(j) \cap P^+(l) \neq \emptyset$ :

$$(6.2.xviii) \quad 2|O_{P^+(l) \cap S^+(m)}^{S(j)}| - 1 + \delta_{N^l \in S(j)}$$

The first part is non null when  $S^+(m) \cap S(j) \cap P^*(l) \neq \emptyset$ .

The second part of equation 6.2.xvi is:

$$(6.2.xix) \quad \sum_{N^k \in S(j)} \sum_{q=1}^{n^k} \frac{\partial^2 o_i^l}{\partial o_q^k \partial w_u^m} \frac{\partial N_q^k}{\partial x_t^{r(j,k)}}$$

A very similar equation was studied as part of equation 48. It is easy to show that the *equation cost* of equation 6.2.xix is given by the following formula:

$$(6.2.xx) \quad 2|O_{P^+(l)}^{S(j)}| - 1$$

It is important to notice that when  $S(j) \cap P^+(l) = \emptyset$ , the second part is null and therefore, the related cost is also null.

The third part of equation 6.2.xvi, i.e.,  $\frac{\partial o_i^l}{\partial \sigma^m} \frac{\partial^2 N^m}{\partial x_t^{r(j,m)} \partial w_u^m}$  is always non null and imply a simple scalar product and therefore the following cost:  $2|O^m| - 1$  (when  $N^m \neq N^l$ ). When  $N^m = N^l$ , there is no additional computing cost as  $\frac{\partial o_i^l}{\partial \sigma^l} = Id$ .

Finally, equation 6.2.xvi is computed by summing the three different parts which implies at most 2 additions. We have therefore the following *equation cost* for equation 6.2.xvi:

$$He_i^l(o_t^j, w_u^m) = 2 \left| O_{P^+(l)}^{S(j)} \right| + 2 \left| O_{P^+(l) \cap S^+(m)}^{S(j)} \right| + \delta_{N^l \in S(j)} + \delta_{m \neq l} (2|O^m| - 1) \quad (96)$$

Therefore, for all values of  $t$  and  $u$ :

$$He_i^l(o^j, w^m) = |O^j| |W^m| \left( 2 \left| O_{P^+(l)}^{S(j)} \right| + 2 \left| O_{P^+(l) \cap S^+(m)}^{S(j)} \right| + \delta_{N^l \in S(j)} + \delta_{m \neq l} (2|O^m| - 1) \right) \quad (97)$$

Equation 6.2.xvi uses the differential  $\frac{\partial^2 N_q^k}{\partial x_i^{r(j,k)} \partial w_u^m}$  in exactly the same way as equation 6.2.x. But the equation cost of equation 6.2.ix, which is used to compute this differential, does not change whether  $N^m \in S(j)$  or not. Therefore, the *equation cost* of equation 49 is completely determined and given by exactly the same equation as for the *equation cost* of equation 48 (i.e., equation 95).

### 6.2.2 Variable part

#### 1. equation 45:

Equation 45 has been split into two equations (6.2.i and 6.2.ii). In equation 6.2.ii, we need:

- $\frac{\partial o_i^l}{\partial o_q^l}$  (for all  $q$ ): the computation time of this first order differential is not included into the variable cost;
- $\frac{\partial^2 N_q^j}{\partial w_i^j \partial w_u^m}$  (for all  $q$ ): this differential is computed with the help of equation 6.2.i. The main problem is that the result does not depend on  $i$  and therefore can be used for arbitrary value of  $i$ . The variable cost linked to this value is in fact the *total cost* of equation 6.2.i;
- $\frac{\partial^2 o_i^l}{\partial o_q^l \partial w_u^m}$  (for all  $q$ ): this differential will be recursively computed with the help of equation 48 or 49. Therefore the variable cost related to this value is the total cost of this equation;
- $\frac{\partial N_q^j}{\partial w_i^j}$ : the computation cost of this local first order differential is not included into the variable cost.

In equation 6.2.i, we need:

- $\frac{\partial^2 N_q^j}{\partial w_i^j \partial x^{r(k,j)}}$  (for all  $N^k \in P(j)$ ): the computation cost of this second order local differential is not included in the variable cost;
- $\frac{\partial o^k}{\partial w_u^m}$  (for all  $N^k \in P(j)$ ): the computation cost of this first order differential is not included in the variable cost.

This analysis shows that equation 6.2.i has a null variable cost. Therefore, its total cost is:

$$HN_q^j(w^j, w^m) = HeN_q^j(w^j, w^m) \quad (98)$$

#### 2. equation 47:

This equation needs the following “variables”:

- $\frac{\partial N_q^j}{\partial w_i^j}$ ,  $\frac{\partial o_i^l}{\partial o_q^l}$  and  $\frac{\partial^2 N_q^j}{\partial w_i^j \partial w_u^m}$ : as they are first order differentials or local second order differentials, the computation cost of these values is not included in the variable cost;
- $\frac{\partial^2 o_i^l}{\partial o_q^l \partial w_u^m}$ : this value will be computed with the help of equation 48 or 49, and will therefore introduce as variable cost the total cost of one of the recursive equation.

#### 3. equation 48:

This equation has been split into two equations (6.2.x and 6.2.ix). Equation 6.2.x implies to compute the following value:

- $\frac{\partial o_i^l}{\partial o_q^k}$  and  $\frac{\partial N_q^k}{\partial o_i^{r(j,k)}}$ : the computation time of these first order differentials is not included into the variable cost;
- $\frac{\partial^2 N_q^k}{\partial x_i^{r(j,k)} \partial w_u^m}$ : this differential is computed with the help of equation 6.2.ix. Therefore, the variable cost related to this value is the total cost of equation 6.2.ix. As for equation 6.2.i, we must notice that the obtained differential does not depend on  $i$ ;

- $\frac{\partial^2 o_q^l}{\partial o_t^{r(j,k)} \partial w_u^m}$  (for all  $q$ ): this value is recursively evaluated with equation 48 or 49. The total cost of this computation must be therefore included into the variable cost of equation 6.2.x.

Equation 6.2.ix implies only to compute  $\frac{\partial^2 N_q^k}{\partial x_t^{r(j,k)} \partial x^r}$  and  $\frac{\partial o^{P(k)r}}{\partial w_u^m}$ : the computation of these local or first order differentials has not to be included into the variable cost. Therefore, the variable cost of equation 6.2.ix is null and its total cost is:

$$HN_q^k(o^j, w^m) = HeN_q^k(o^j, w^m) \quad (99)$$

4. equation 49:

The case of equation 49 is very similar to the case of equation 48. In fact, it is easy to see that the variable cost is the same for both equations.

### 6.2.3 Computation order and total cost

The main problem of the hybrid method is to choose the differentiation order for each pair of nodes. Let us assume that we choose to compute  $\frac{\partial^2 o_i^l}{\partial w_t^j \partial w_u^m}$  (with this order). The recursive calculation method implies to compute  $\frac{\partial^2 o_i^l}{\partial o_s^k \partial w_u^m}$ , where  $N^k \in S^*(j)$ . A direct consequence of this computation is that we can use the local equations (i.e., equations 45 and 47) to compute  $\frac{\partial^2 o_i^l}{\partial w_s^k \partial w_u^m}$ .

Let  $N^k$  be a generalized successor of  $N^j$  (i.e.,  $N^k \in S^+(j)$ ). The differentiation order can be  $\frac{\partial^2 o_i^l}{\partial w_s^k \partial w_u^m}$  or the opposite, and therefore, the chosen order for  $(N^j, N^m)$  introduces a constraint on the rest of the computation. If we choose to compute  $\frac{\partial^2 o_i^l}{\partial w_u^m \partial w_s^k}$ , we must compute  $\frac{\partial^2 o_i^l}{\partial o_q^r \partial w_s^k}$  for all  $N^q \in S^*(m)$ . The second differential  $\frac{\partial^2 o_i^l}{\partial o_q^r \partial w_s^k}$  might be useful for  $N^r \in S^+(m)$ . Therefore, the only “superfluous” terms are the  $\frac{\partial^2 o_i^l}{\partial o_q^r \partial w_s^k}$  for  $q \in [1, |O^m|]$ .

Let us assume that  $N^k \neq N^m$ . Then,  $\frac{\partial^2 o_i^l}{\partial w_s^k \partial w_u^m}$  is computed (in this order) with the help of equation 45. As  $\frac{\partial^2 o_i^l}{\partial o_s^k \partial w_u^m}$  is already known (for all  $s$ ), the total cost of this equation reduces to its equation cost. This cost (for all  $s$  and  $u$ ) is:

$$(6.2.xxii) \quad C^1 = \mathcal{H}e_i^l(w^k, w^m) = |O^k| HeN_1^k(w^k, w^m) + He_i^l(w^k, w^m)$$

The computation of  $\frac{\partial^2 o_i^l}{\partial w_u^m \partial w_s^k}$  introduces also equation 45 equation cost, i.e.:

$$(6.2.xxiii) \quad C_a^2 = \mathcal{H}e_i^l(w^m, w^k) = |O^m| HeN_1^m(w^m, w^k) + He_i^l(w^m, w^k)$$

We need also to evaluate the equation cost of the computation of the “superfluous” terms. Their computation is based on equation 48 (we assume that  $N^k \notin S(m)$ ). The equation cost of this equation is (for all  $q$  and  $s$ ):

$$(6.2.xxiiii) \quad C_b^2 = \mathcal{H}e_i^l(o^m, w^k) = He_i^l(o^m, w^k) + \sum_{N^p \in S(m)} |O^p| HeN_1^p(o^m, w^k)$$

Therefore, we can compare both methods by computing  $C^1$  and  $C^2$ :

$$(6.2.xxv) \quad C^2 = C_a^2 + C_b^2 = \mathcal{H}e_i^l(w^m, w^k) + \mathcal{H}e_i^l(o^m, w^k)$$

- if  $N^m \in P^+(N^k)$  then:

$$C^1 = |W^k| |W^m| \left( |O^k| \left( 2|I_{S^*(m)}^k| + 3 \right) - 1 \right)$$

As  $N^k \notin P^+(N^m)$ :

$$C_a^2 = |W^k| |W^m| (2|O^m| - 1)$$

*In general*, the output dimension of the considered neurons is constant (and equal to one). Therefore we can assume *in general* that  $C^1 \geq C_a^2$ . This result means that we must directly compare  $C^1$  to  $C^2$ . If we assume that the neuron output dimension is equal to 1, we have:

$$\begin{aligned} C^1 &= 2|W^k||W^m|(|P(k) \cap S^*(m)| + 1) \\ C_a^2 &= |W^k||W^m| \\ C_b^2 &= |W^k| \left( 2|S(m) \cap P^+(l)| + 2|S(m) \cap P^+(l) \cap S^+(k)| + \delta_{N^l \in S(m)} \right. \\ &\quad \left. + |S(m) \cap P^*(l) \cap S^+(k)|(2|P(p) \cap S^*(k)| - 1) - 1 \right) \end{aligned}$$

*In general*, the output dimension of a neuron is smaller than its parameter number. On an intuitive point of view,  $C_b^2$  is therefore smaller than  $C^1$ . Therefore for *standard* model, we may assume that  $C^1 \geq C^2$ . Moreover, by allowing  $|W^m|$  to take arbitrary values, it is obvious that  $C^1$  will grow faster than  $C^2$ . Of course, it is always possible to build a particular architecture in which this is not the case.

- if  $N^m \notin P^+(N^k)$  and  $N^k \notin P^+(N^m)$  then:

$$\begin{aligned} C^1 &= |W^k||W^m| (2|O^m| - 1) \\ C_a^2 &= |W^m||W^k| (2|O^k| - 1) \end{aligned}$$

In this case, general assumptions allow us to conclude that  $C^1 \simeq C_a^2$ . Therefore, in this case  $C^1 \leq C^2$ . Once again, it is always possible to build a particular architecture in which this is not the case.

The main conclusion of this short study is that it is not possible to assume that changing the differentiation order for  $(N^k, N^m)$  is time consuming. Therefore, in order to compute the Hessian matrix of  $o_i^l$ , we must choose a **computation strategy**: this is approximately a list of second order differentials to compute.

In fact, a **computation strategy** is pair of subsets of  $\mathcal{N}^2$  ( $\mathcal{N}$  is the neuron set)  $(\mathcal{L}, \mathcal{R})$  which fulfills the following conditions:

- $\forall N^j \in P^+(l), \forall N^m \in P^+(l)$ , with  $j \neq m$ , either  $(N^j, N^m)$  or  $(N^m, N^j)$  belongs to  $\mathcal{L}$ .  $\mathcal{L}$  is in fact a the set of ordered pairs of node. If  $(N^m, N^j)$  belongs to  $\mathcal{L}$ , we will compute  $\frac{\partial^2 o_i^l}{\partial w^m \partial w^j}$  in this order, with the help of the local equation.
- $\forall N^j \in P^+(l)$ ,  $(N^j, N^j)$  belongs to  $\mathcal{L}$ .
- $\forall (N^j, N^m), (N^j, N^m) \in \mathcal{L}$  implies that for all  $N^k \in S^*(j) \cap P^+(l)$ ,  $(N^k, N^m) \in \mathcal{R}$ .

It is quite obvious that, given a computation strategy, we can compute the Hessian of  $o_i^l$  if we compute  $\frac{\partial^2 o_i^l}{\partial w^m \partial w^j}$  in this order for each  $(N^m, N^j) \in \mathcal{L}$  and  $\frac{\partial^2 o_i^l}{\partial o_r^j \partial w^j}$  in this order for all pair  $(N^r, N^j) \in \mathcal{R}$ . The total computation time for the strategy  $(\mathcal{L}, \mathcal{R})$  is therefore:

$$\mathcal{H}_i^l(\mathcal{L}, \mathcal{R}) = \sum_{(N^m, N^j) \in \mathcal{L}} \mathcal{H}e_i^l(w^m, w^j) + \sum_{(N^k, N^j) \in \mathcal{R}} \mathcal{H}e_i^l(o^k, w^j)$$

It is nearly impossible to compare all the different possibilities for  $(\mathcal{L}, \mathcal{R})$ . In fact, for  $n$  nodes, we have  $\frac{n(n-1)}{2}$  pairs of distinct nodes. Therefore, we have  $2^{\frac{n(n-1)}{2}}$  different values for  $\mathcal{L}$ . The above short study shows that in the general case, it is not possible to simplify this choice and therefore, the only way to compare the different strategies would be to compare the different costs: this is not practically possible because of huge number of strategies.

We must be aware that the differentials  $\frac{\partial^2 o_i^l}{\partial w_a^l \partial w_a^j}$  must also be computed for each  $N^j \in P^+(l)$ . In this case, the differentiation order is even more important than in the previously discussed situation. If the order  $(N^l, N^j)$  is chosen, we use the direct method for this computation. The other order imply the use

of the hybrid method and therefore the computation of  $\frac{\partial^2 o_i^l}{\partial o_q^k \partial w_t^l}$  for all  $N^k \in S^*(j)$ . We must therefore add a local strategy to the previously defined computation strategy. This is a pair of subsets of  $\mathcal{N}$ ,  $\mathcal{D}$  and  $\mathcal{H}$ . We have  $\mathcal{D} \cap \mathcal{H} = \emptyset$  and  $\mathcal{D} \cup \mathcal{H} = P^+(l)$ . When  $N^m$  belongs to  $\mathcal{H}$ , we compute  $\frac{\partial^2 o_i^l}{\partial w_u^m \partial w_t^l}$  in this order, with the hybrid method, which implies to compute  $\frac{\partial^2 o_i^l}{\partial o_q^k \partial w_t^l}$  for all  $N^k \in S^*(m)$ . When  $N^m$  belongs to  $\mathcal{D}$ , we compute  $\frac{\partial^2 o_i^l}{\partial w_t^l \partial w_u^m}$  in this order, with the direct method which does not imply any additional computation. If we call  $\mathcal{H}'$  the set defined by  $\mathcal{H}' = \{N^k \in \mathcal{N} \mid P^*(N^k) \cap \mathcal{H} \neq \emptyset\}$ , then the use of the hybrid method implies to compute the  $\frac{\partial^2 o_i^l}{\partial o_q^k \partial w_t^l}$  for all  $N^k \in \mathcal{H}'$  and that's all. The total cost for a local strategy is therefore:

$$(6.2.xxv) \quad \sum_{N^j \in \mathcal{H}} \mathcal{H}e_i^l(w^j, w^l) + \sum_{N^k \in \mathcal{H}'} \mathcal{H}e_i^l(o^k, w^l) + \sum_{N^j \in \mathcal{D}} De_i^l(w^l, w^j)$$

### 6.3 The pure back-propagation method

The pure back-propagation method introduces an additional step during the calculation of the second order differentials. This method uses first a local equation (45 or 47) which allows to compute  $\frac{\partial^2 o_i^l}{\partial w_t^j \partial w_u^m}$  as a function of  $\frac{\partial^2 o_i^l}{\partial o_s^j \partial w_u^m}$ . Then, it uses a new local formula (equation 56) which allows to compute  $\frac{\partial^2 o_i^l}{\partial o_s^j \partial w_u^m}$  as a function of  $\frac{\partial}{\partial \sigma^m} \left( \frac{\partial o_i^l}{\partial \sigma_t^l} \right)^{-m}$ . Finally, a recursive equation (57, 58 or 59) allows to compute  $\frac{\partial}{\partial \sigma^m} \left( \frac{\partial o_i^l}{\partial \sigma_t^l} \right)^{-m}$  as a function of  $\frac{\partial}{\partial \sigma^m} \left( \frac{\partial o_i^l}{\partial \sigma_q^l} \right)^{-m}$ , where  $N^p \in S^+(j)$ .

#### 6.3.1 Equation cost

1. equations 45 and 47:

These equations have already been studied in the previous subsection. The fact they are used here for the pure back-propagation method does not change their complexity.

2. equation 56:

Equation 56 has the following scalar version:

$$(6.3.i) \quad \frac{\partial^2 o_i^l}{\partial \sigma_t^j \partial w_u^m} = \sum_{s=1}^{n^m} \frac{\partial}{\partial \sigma_s^m} \left( \frac{\partial o_i^l}{\partial \sigma_t^l} \right)^{-m} \frac{\partial N_s^m}{\partial w_u^m}$$

This equation has to be used only when  $N^j \in P^+(N^l)$ . In this case, as  $N^m \in P^+(l)$ ,  $\frac{\partial}{\partial \sigma_s^m} \left( \frac{\partial o_i^l}{\partial \sigma_t^l} \right)^{-m}$  and  $\frac{\partial N_s^m}{\partial w_u^m}$  are never null. Therefore, for each  $s$ , we have one multiplication. Moreover, we must sum the obtained products: this introduces  $|O^m| - 1$  additions. Finally, we have the following *equation cost*:

$$\mathcal{P}e_i^l(o_t^j, w_u^m) = 2|O^m| - 1 \quad (100)$$

And therefore, for all values of  $t$  and  $u$ :

$$\mathcal{P}e_i^l(o^j, w^m) = |O^j| |W^m| (2|O^m| - 1) \quad (101)$$

3. equation 59:

As we have:

$$(6.3.ii) \quad \frac{\partial^2 N_q^k}{\partial x_t^{r(j,k)} \partial x_u^m} = \sum_{r=1}^{p^k} \frac{\partial^2 N_q^k}{\partial x_t^{r(j,k)} \partial x^r} \frac{\partial o^{P(k)r}}{\partial o_u^m},$$

equation 59 can be simplified into:

$$(6.3.iii) \quad \frac{\partial}{\partial o_u^m} \left( \frac{\partial o_i^l}{\partial o_t^j} \right)^{-m} = \sum_{N^k \in S(j)} \sum_{q=1}^{n^k} \left( \frac{\partial o_i^l}{\partial o_q^k} \frac{\partial^2 N_q^k}{\partial x_t^{r(j,k)} \partial x_u^m} + \frac{\partial}{\partial o_u^m} \left( \frac{\partial o_i^l}{\partial o_q^k} \right)^{-m} \frac{\partial N_q^k}{\partial x_t^{r(j,k)}} \right)$$

This equation is exactly the same as equation 6.2.x. The only differences are the involved ‘‘variables’’:

- equation 6.3.iii uses  $\frac{\partial^2 N_q^k}{\partial x_t^{r(j,k)} \partial x_u^m}$  whereas equation 6.2.x uses  $\frac{\partial^2 N_q^k}{\partial x_t^{r(j,k)} \partial w_u^m}$ :

Both differentials are null for the same conditions (i.e., when  $N^m \notin P^+(k)$ ). In both case, we cannot have  $N^m = N^k$  (for the hybrid method,  $N^m \notin S(j)$  and for the pure back-propagation,  $N^m \notin S^+(j)$ ). Therefore, both ‘‘variables’’ behave exactly the same way in both equations.

- equation 6.3.iii uses  $\frac{\partial}{\partial o_u^m} \left( \frac{\partial o_i^l}{\partial o_q^k} \right)^{-m}$  whereas equation 6.2.x uses  $\frac{\partial^2 o_i^l}{\partial o_q^k \partial w_u^m}$ :

Once again, both differentials are null for the same reasons (i.e., when  $N^k \notin P^+(l)$  or  $N^m \notin P^+(l)$ ). Therefore, both ‘‘variables’’ behave exactly the same way in both equations.

As the other variables of both equations are equal, we can conclude that the complexities of both equations are equal, and therefore, the *equation cost* of equation 6.3.iii is (after simplifications linked to the fact that  $N^m \notin S^+(j)$ ):

$$Pe_i^l(o_t^j, o_u^m) = 2|O_{P^+(l)}^{S(j)}| + 2|O_{P^+(l) \cap S^+(m)}^{S(j)}| - 1 \quad (102)$$

Therefore, for all values of  $t$  and  $u$ , we have:

$$Pe_i^l(o^j, o^m) = |O^j| |O^m| \left( 2|O_{P^+(l)}^{S(j)}| + 2|O_{P^+(l) \cap S^+(m)}^{S(j)}| - 1 \right) \quad (103)$$

Equation 6.3.ii is very similar to equation 6.2.ix but the involved variables do not behave in the same way and the complexities are therefore different. Equation 6.3.ii sums the following values (for  $N^r \in P(k)$ ):

$$(6.3.iv) \quad \frac{\partial^2 N_q^k}{\partial x_t^{r(j,k)} \partial x^{r(r,k)}} \frac{\partial o^r}{\partial o_u^m}$$

As a local differential,  $\frac{\partial^2 N_q^k}{\partial x_t^{r(j,k)} \partial o^{r(r,k)}}$  can be assumed to remain non null. When  $N^m \in P^+(r)$ ,  $\frac{\partial o^r}{\partial o_u^m}$  is also non null and formula 6.3.iv is simply a scalar product which complexity is  $2|O^r| - 1$ . When  $N^m = N^r$ ,  $\frac{\partial o^r}{\partial o_u^m} = Id$  and therefore, formula 6.3.iv is only a selection process (which keeps one term of the vector  $\frac{\partial^2 N_q^k}{\partial x_t^{r(j,k)} \partial x^{r(r,k)}}$ ): this operation has no computation cost. As formula 6.3.iv is evaluated for each  $N^r \in P(k) \cap S^+(m)$ , it implies the following cost:

$$\sum_{N^r \in P(k) \cap S^+(m)} (2|O^r| - 1)$$

Moreover, equation 6.3.ii implies to sum the partial results obtained from equation 6.3.iv and there are  $|P(k) \cap S^*(m)|$  values to sum, which introduces  $|P(k) \cap S^*(m)| - 1$  operations. As  $|P(k) \cap S^*(m)| = |P(k) \cap S^+(m)| + \delta_{N^m \in P(k)}$ , we have finally the following *equation cost* for equation 6.3.ii:

$$(6.3.v) \quad \sum_{N^r \in P(k) \cap S^+(m)} (2|O^r| - 1) + |P(k) \cap S^+(m)| + \delta_{N^m \in P(k)} - 1 = 2|I_{S^+(m)}^k| + \delta_{N^m \in P(k)} - 1$$

It is important to notice that  $\frac{\partial^2 N_q^k}{\partial x_t^{r(j,k)} \partial x_u^m}$  does not depend on  $i$  and as to be computed for each  $q$ , but only when  $N^k \in P^*(l) \cap S^+(m)$ . The complete complexity for all values of  $t$  and  $u$  is therefore:

$$Pe N_q^k(o^j, o^m) = \delta_{N^k \in P^*(l) \cap S^+(m)} |O^j| |O^m| \left( 2|I_{S^+(m)}^k| + \delta_{N^m \in P(k)} - 1 \right) \quad (104)$$

Finally, the *equation cost* of equation 59 for all  $t$  and all  $u$  is:

$$\mathcal{P}e_i^l(o^j, o^m) = \mathcal{P}e_i^l(o^j, o^m) + \sum_{N^k \in S(j)} |O^k| \mathcal{P}e_i N_q^k(o^j, o^m) \quad (105)$$

4. equations 57 and 58:

Obviously, these equations do not introduce any computation.

### 6.3.2 Computation order and total cost

The main difference between the pure back-propagation and the hybrid algorithm is the symmetry of the recursive equations. The main problem is to choose the differentiation order for “independent” nodes.  $N^j$  and  $N^m$  are said to be independent if  $N^m \notin P^*(j) \cup S^*(j)$ . The notation for this independence relationship is  $N^j \# N^m$ . This relationship is symmetrical. Let us call  $\#(l)$  the part of  $\mathcal{N}^2$  given by:

$$\#(l) = \{(N^m, N^j) \in P^+(l) \times P^+(l) \mid N^m \# N^j\}$$

If  $N^j \# N^m$ , the differential can be computed with this order:  $\frac{\partial^2 o_i^l}{\partial w_i^j \partial w_u^m}$ , or with the opposite order. But as the nodes are independent:

$$\frac{\partial}{\partial o_v^m} \left( \frac{\partial o_i^l}{\partial o_t^j} \right)^{\rightarrow m} = \frac{\partial}{\partial o_t^j} \left( \frac{\partial o_i^l}{\partial o_v^m} \right)^{\rightarrow j} = \frac{\partial^2 o_i^l}{\partial o_t^j \partial o_v^m}$$

If we choose to compute  $\frac{\partial}{\partial o_v^m} \left( \frac{\partial o_i^l}{\partial o_t^j} \right)^{\rightarrow m}$ , then we will need  $\frac{\partial}{\partial o_v^m} \left( \frac{\partial o_i^l}{\partial o_s^k} \right)^{\rightarrow m}$  where  $N^k \in S^+(j)$ . The main problem for the hybrid method was that the chosen differentiation order for  $N^j$  and  $N^m$  implied the same order for  $N^k$  and  $N^m$ . This is no more the case for the current approach, because if  $N^k \# N^m$ ,  $\frac{\partial}{\partial o_v^m} \left( \frac{\partial o_i^l}{\partial o_s^k} \right)^{\rightarrow m} = \frac{\partial}{\partial o_s^k} \left( \frac{\partial o_i^l}{\partial o_v^m} \right)^{\rightarrow k}$  and therefore, the opposite differentiation order can be chosen for the  $(N^k, N^m)$  pair. This result implies that the computation can be optimized for each pair of independent nodes. The minimum cost is obtained by:

$$(6.3.vi) \quad \mathcal{M}e_i^l(o^j, o^m) = \min \left( \mathcal{P}e_i^l(o^j, o^m) + \mathcal{P}e_i^l(o^j, w^m) + \mathcal{H}e_i^l(w^j, w^m), \right. \\ \left. \mathcal{P}e_i^l(o^m, o^j) + \mathcal{P}e_i^l(o^m, w^j) + \mathcal{H}e_i^l(w^m, w^j) \right)$$

We have therefore the following *equation cost* for computation of  $\frac{\partial^2 o_i^l}{\partial w_i^j \partial w_u^m}$  (without taking into account the recursivity):

- if  $N^l$  is an input node, we have:

$$\mathcal{M}e_i^l(w^j, w^m) = 0$$

- if  $N^l$  is an inner node, we have:

- if  $j = l$ :

- \* if  $m \neq l$ :

$$\mathcal{M}e_i^l(w^l, w^m) = \mathcal{D}e_i^l(w^l, w^m)$$

- \* if  $m = l$ :

$$\mathcal{M}e_i^l(w^l, w^l) = 0$$

- if  $j \neq l$  (and of course  $m \neq l$ , because of the symmetry of the problem):

- \* if  $m = j$ :

$$\mathcal{M}e_i^l(w^j, w^j) = \mathcal{P}e_i^l(o^j, o^j) + \mathcal{P}e_i^l(o^j, w^j) + \mathcal{H}e_i^l(w^j, w^j)$$

\* if  $m \neq j$ :

· if  $N^m \in P^+(j)$ :

$$\mathcal{M}e_i^l(w^j, w^m) = \mathcal{P}e_i^l(o^j, o^m) + \mathcal{P}e_i^l(o^j, w^m) + \mathcal{H}e_i^l(w^j, w^m)$$

· if  $N^m \in S^+(j)$ :

$$\mathcal{M}e_i^l(w^j, w^m) = \mathcal{P}e_i^l(o^m, o^j) + \mathcal{P}e_i^l(o^m, w^j) + \mathcal{H}e_i^l(w^m, w^j)$$

· if  $N^m \notin S^+(j)$  and  $N^m \notin P^+(j)$ : we can optimize the differentiation order as explained before (see equation 6.3.vi).

Therefore, the total computation cost for the Hessian of  $o_i^l$  is:

$$\frac{1}{2} \sum_{N^j \in P^+(l)} \left( 2\mathcal{M}e_i^l(w^j, w^j) + \sum_{N^m \in P^+(l), m \neq j} \mathcal{M}e_i^l(w^j, w^m) \right) \quad (106)$$

## 7 Complexity with an error function

### 7.1 Direct method

We can consider the error function  $\mathcal{E}$  as an output node. In this case, equations 39 and 62 are identical (as  $P(\mathcal{E}) = Out$ ). Therefore, the equation cost of equation 62 is equal to equation cost of equation 39, and we have:

$$(7.1.i) \quad \begin{aligned} De^{\mathcal{E}}(w^j, w^m) &= |W^j| |W^m| \left( 2|I_{S^*(m)}^{\mathcal{E}}| + 2|I_{S^*(m) \cap S^*(j)}^{\mathcal{E}}| - 1 \right) \\ &= |W^j| |W^m| \left( 2|O_{S^*(m)}| + 2|O_{S^*(m) \cap S^*(j)}| - 1 \right), \end{aligned}$$

and

$$(7.1.ii) \quad \begin{aligned} De^{\mathcal{E}}(o^k, w^m) &= |O^k| |W^m| \left( 2|I_{S^*(m)}^{\mathcal{E}}| - 1 \right) \\ &= |O^k| |W^m| \left( 2|O_{S^*(m)}| - 1 \right) \end{aligned}$$

The only difference between the error function and a normal node is that the error function has no parameter and therefore we do not have to take care about the particular cases introduced for the general case.

It is now easy to recompute the total complexity implied by the error function case with the help of sub-section 6.1.4. We obtain the following *equation cost*:

$$\mathcal{D}^{\mathcal{E}} = \sum_{N^j \in \mathcal{N}} |W^j| \left( \sum_{N^m \in \mathcal{N}} |W^m| \left( \min(|O_{S^*(m)}|, |O_{S^*(j)}|) + |O_{S^*(j) \cap S^*(m)}| - \frac{1}{2} \right) + |W^j| \left( 2|O_{S^*(j)}| - \frac{1}{2} \right) + |O| (2|O_{S^*(j)}| - 1) \right) \quad (107)$$

Finally, the *total computation time* of  $\mathcal{E}$  Hessian is:

$$H_d^{\mathcal{E}} = \sum_{N^k \in Out} H_d^k + \mathcal{D}^{\mathcal{E}} \quad (108)$$

## 7.2 Hybrid Method

### 7.2.1 Local formulae

When  $\mathcal{E}$  is considered as an output neuron, the local formulae (63 and 64) are identical to the local equations of the general case (i.e., equations 45 and 47). Therefore, the related *equation costs* are equal and we have:

- for equation 63 (i.e., when  $N^m \neq N^j$ ):

$$\mathcal{H}e^{\mathcal{E}}(w^j, w^m) = |W^j||W^m| \left( |O^j| \left( 2 + \delta_{N^m \in P^+(j)} \left( 1 + 2|I_{S^*(m)}^j| \right) \right) - 1 \right) \quad (109)$$

- for equation 64 (i.e., when  $N^m = N^j$ ):

$$\mathcal{H}e^{\mathcal{E}}(w^j, w^j) = |W^j|^2 (4|O^j| - 1) \quad (110)$$

### 7.2.2 Recursive formulae

Once again,  $\mathcal{E}$  can be considered as an output neuron and the recursive formulae 66 and 67 are equal to the corresponding general case equations (i.e., 48 and 49). Therefore, the related *equation costs* are equal and we have:

- equation 66 (i.e., when  $N^m \notin S(j)$ ):

$$(7.2.i) \quad \mathcal{H}e^{\mathcal{E}}(o^j, w^m) = |O^j||W^m| \left( 2|O^{S(j)}| + 2|O_{S^+(m)}^{S(j)}| + \sum_{N^k \in S(j) \cap S^+(m)} |O^k| \left( 2|I_{S^*(m)}^k| - 1 \right) - 1 \right),$$

and therefore:

$$\mathcal{H}e^{\mathcal{E}}(o^j, w^m) = |O^j||W^m| \left( 2|O^{S(j)}| + |O_{S^+(m)}^{S(j)}| + 2 \sum_{N^k \in S(j) \cap S^+(m)} |O^k||I_{S^*(m)}^k| - 1 \right) \quad (111)$$

- equation 67 (i.e., when  $N^m \in S(j)$ ):

$$(7.2.ii) \quad \mathcal{H}e^{\mathcal{E}}(o^j, w^m) = |O^j||W^m| \left( 2|O^{S(j)}| + |O_{S^+(m)}^{S(j)}| + 2 \sum_{N^k \in S(j) \cap S^+(m)} |O^k||I_{S^*(m)}^k| + 2|O^m| - 1 \right) \quad (112)$$

- equation 65 (i.e., when  $N^j \in Out$ ):

In this case, the equation is new and the result is obtained by summing the following values:

$$(7.2.ii) \quad \frac{\partial^2 \mathcal{E}}{\partial x_i^j \partial x^r} \frac{\partial o^{Out^r}}{\partial w_u^m}$$

As a local second order differential,  $\frac{\partial^2 \mathcal{E}}{\partial x_i^j \partial x^r}$  is always non null. The first order differential  $\frac{\partial o^{Out^r}}{\partial w_u^m}$  is null when  $Out^r \notin S^*(m)$ . Therefore, when  $Out^r \in S^*(m)$ , this scalar product implies  $2|O^{Out^r}| - 1$  operations. As we have to sum the resulting values in order to obtain equation 65, the equation cost for all values of  $t$  and  $u$  is:

$$\mathcal{H}e^{\mathcal{E}}(o^j, w^m) = |O^j||W^m| (2|O_{S^*(m)}| - 1) \quad (113)$$

### 7.2.3 Differentiation order

The main problem of the hybrid method is the differentiation order which is virtually impossible to optimize in the general case. For an error function, the problem is exactly the same as for a node output.

## 7.3 Pure Back-Propagation

Once again,  $\mathcal{E}$  can be considered as a final node:

- Equation 70 is equal to equation 56. Therefore the *equation cost* is the same and we have:

$$\mathcal{P}e^{\mathcal{E}}(o^j, w^m) = |O^j| |W^m| (2|O^m| - 1) \quad (114)$$

- Equation 72 is equal to equation 59. Therefore the *equation cost* is also the same and we have:

$$\mathcal{P}e^{\mathcal{E}}(o^j, o^m) = |O^j| |O^m| \left( 2|O^{S(j)}| + |O_{S^+(m)}^{S(j)}| + \sum_{N^k \in S(j) \cap S^+(m)} |O^k| \left( 2|I_{S^*(m)}^k| + \delta_{N^m \in P(k)} \right) - 1 \right) \quad (115)$$

- Equation 71 is used when  $N^j \in Out$ . It sums the following values:

$$(7.3.i) \quad \frac{\partial^2 \mathcal{E}}{\partial x_t^k \partial x^r} \frac{\partial o^{Out^r}}{\partial o_v^m}$$

As a local second order differential,  $\frac{\partial^2 \mathcal{E}}{\partial x_t^k \partial x^r}$  is non null. When  $N^m \in P^+(Out^r)$ ,  $\frac{\partial o^{Out^r}}{\partial o_v^m}$  is non null. In this case, equation 7.3.i is a simple scalar product and implies  $2|O^{Out^r}| - 1$  operations. When  $N^m = Out^r$ ,  $\frac{\partial o^{Out^r}}{\partial o_v^m} = Id$  and equation 7.3.i is a selection process with no cost. We have then to sum the partial results, which imply  $|Out \cap S^*(m)| - 1$  operations. Therefore the *equation cost* of equation 71 is:

$$(7.3.ii) \quad \mathcal{P}e^{\mathcal{E}}(o_t^j, o_v^m) = \sum_{N^k \in Out \cap S^+(m)} (2|O^k| - 1) + |Out \cap S^*(m)| - 1$$

Therefore, we have:

$$\mathcal{P}e^{\mathcal{E}}(o^j, o^m) = |O^j| |O^m| (2|O_{S^+(m)}| + \delta_{N^m \in Out} - 1) \quad (116)$$

The final analysis (differentiation order, etc.) given in subsection 6.3.2 is also valid for the case of the error function.

## 8 First order differentials

### 8.1 Needed first order differentials

Computing the second order differential implies to compute first order differentials such as  $\frac{\partial o^l}{\partial w^j}$ . The purpose of this section is to study the differences between the three algorithms about their required first order differentials.

### 8.1.1 Direct method

Computing  $\frac{\partial^2 o_i^l}{\partial w_i^j \partial w_u^m}$  implies to compute  $\frac{\partial o_i^k}{\partial w_u^m}$  for each  $N^k \in P(l) \cap S^*(m)$  (and also  $\frac{\partial o_i^k}{\partial w_i^j}$  for each  $N^k \in P(l) \cap S^*(j)$ ). Because of the recursive aspect of the direct method equation, it is easy to see that as soon as  $N^k \in P^+(l)$  and  $N^m \in P^*(k)$ ,  $\frac{\partial o_i^k}{\partial w_u^m}$  must be computed.

Computing  $\frac{\partial^2 \mathcal{E}}{\partial w_i^j \partial w_u^m}$  with equation 62 implies to compute  $\frac{\partial o_i^k}{\partial w_u^m}$  for all  $N^k \in Out$ . The recursive definition of the direct method implies to compute the Hessian matrices of  $o_k^l$  for each  $N^l \in \mathcal{N}$  in order to compute  $\frac{\partial^2 \mathcal{E}}{\partial w_i^j \partial w_u^m}$ . As explain before, this will imply to compute  $\frac{\partial o_i^k}{\partial w_u^m}$  for each  $N^k \in P^+(Out)$  and for each  $N^m \in P^+(k)$ . Therefore, added to the requirements of equation 62, computing  $\mathcal{E}$  Hessian implies to compute  $\frac{\partial o_i^k}{\partial w_u^m}$  for each  $N^k \in \mathcal{N}$  and for each  $N^m \in P^+(k)$ .

### 8.1.2 Hybrid method

Once again, this method is the most complex to study. The needed first order differentials depend on the computation strategy:

- The local part of the hybrid method used to compute  $\frac{\partial^2 o_i^l}{\partial w_i^j \partial w_u^m}$  implies to compute  $\frac{\partial o_i^r}{\partial w_u^m}$  for all  $N^r \in P(j) \cap S^*(m)$  and  $\frac{\partial o_i^l}{\partial o_j^l}$ . It implies also to compute  $\frac{\partial^2 o_i^l}{\partial o_j^l \partial w_u^m}$  for all  $N^p \in S^*(j)$ .
- The recursive part of the hybrid method used to compute  $\frac{\partial^2 o_i^l}{\partial o_j^l \partial w_u^m}$  implies to compute  $\frac{\partial o_i^r}{\partial w_u^m}$  for all  $N^r \in P(k) \cap S^*(m)$  where  $N^k \neq N^m$  and  $N^k \in S(j)$ . It implies also to compute  $\frac{\partial o_i^l}{\partial o_k^l}$  for all  $N^k \in S(j)$ .

Let  $(N^j, N^m)$  be a pair of distinct nodes with  $N^j \in P^+(l)$  and  $N^m \in P^+(j)$ . The partial differential  $\frac{\partial o_j^j}{\partial w_u^m}$  has to be computed when  $w^m$  is the second differentiation variable:

- $\frac{\partial o_j^j}{\partial w_u^m}$  is needed for a local part formula if  $N^j$  belongs to  $P(r)$  where  $N^r \in P^+(l)$ . Therefore, we need  $\frac{\partial o_j^j}{\partial w_u^m}$  only if  $\frac{\partial o_i^l}{\partial w_i^r \partial w_u^m}$  is computed in this order for  $N^r \in S(j) \cap P^+(l)$ .
- $\frac{\partial o_j^j}{\partial w_u^m}$  is needed for a recursive part formula if  $N^j$  belongs to  $P(k)$  where  $N^k$  belongs to  $S(r)$  and where  $N^r$  belongs to  $P^+(l)$ . Therefore, we need  $\frac{\partial o_j^j}{\partial w_u^m}$  only if  $\frac{\partial o_i^l}{\partial o_i^r \partial w_u^m}$  is computed in this order for  $N^r \in P(S(j)) \cap P^+(l)$ .

This short analysis implies that it is possible to choose a computation strategy that does not need the partial differential  $\frac{\partial o_j^j}{\partial w_u^m}$  as long as:

1. for all  $N^r$  in  $P(S(j)) \cap P^+(l)$ , the recursive part is computed in this order:  $\frac{\partial o_i^l}{\partial o_i^r \partial w_u^m}$ .
2. for all  $N^r$  in  $S(j) \cap P^+(l)$ , the local part is computed in this order:  $\frac{\partial o_i^l}{\partial w_u^m \partial w_i^r}$ .

Therefore, the needed first order differentials are entirely determined by the computation strategy.

### 8.1.3 Pure back-propagation

Let  $N^j \in P^+(l)$  and  $N^m \in P^+(j)$ . The second condition implies that the second order differential  $\frac{\partial^2 o_i^l}{\partial w_i^j \partial w_u^m}$  must be computed in the given order. In order to use the first local part, we need therefore to compute  $\frac{\partial^2 o_i^l}{\partial o_i^r \partial w_u^m}$ . This value is computed with the help of  $\frac{\partial}{\partial o_i^m} \left( \frac{\partial o_i^l}{\partial o_i^r} \right)^{\rightarrow m}$ . The recursive equation for this value implies to compute  $\frac{\partial o_i^r}{\partial o_i^m}$  for all  $N^r \in P(S(j) \cap P^*(l)) \cap S^*(m)$ . But as  $N^j \in P^+(l)$ ,  $N^l \in S(j) \cap P^*(l)$  and therefore  $N^j \in P(S(j) \cap P^*(l)) \cap S^*(m)$ . Therefore, we must compute  $\frac{\partial o_i^j}{\partial o_i^m}$ .

Moreover, we have to compute the  $\frac{\partial^2 o_i^l}{\partial w_u^m \partial w_t^j}$  with the direct method (as previously assumed), i.e., with the help of equation 40. This implies to compute  $\frac{\partial o^j}{\partial w^m}$  for all  $N^j \in P(l)$  and for all  $N^m \in S^+(j)$ . Moreover the use of equation 45 in order to compute  $\frac{\partial^2 o_i^l}{\partial w_u^j \partial w_t^m}$  implies the computation of  $\frac{\partial o^r}{\partial w^m}$  for all  $N^r \in P(j) \cap S^*(m)$ . Therefore, as long as  $N^j$  belongs to  $P^+(l) \setminus P(l)$ , we need to compute  $\frac{\partial o^j}{\partial w^m}$  for all  $N^m \in S^+(j)$ . Therefore, we must compute  $\frac{\partial o^j}{\partial w^m}$  for all  $N^j \in P^+(l)$  and for all  $N^m \in P^+(j)$ .

Finally, when  $N^j \in P^+(l)$ , we have of course to compute  $\frac{\partial^2 o_i^l}{\partial w_t^j \partial w_u^m}$ , with the help of the local equation 47. This implies to compute  $\frac{\partial o^l}{\partial o^j}$ .

Let us now study the case of an error function. This case is very similar to the previous one. If  $N^j$  is an arbitrary node and  $N^m$  is a predecessor of  $N^j$  (i.e.,  $N^j \in P^+(N^m)$ ), then we must compute  $\frac{\partial^2 \mathcal{E}}{\partial w_t^j \partial w_u^m}$  in the given order. This value is obtained with the help of  $\frac{\partial}{\partial o^m} \left( \frac{\mathcal{E}}{\partial o_t^j} \right)^{\rightarrow m}$ .

If  $N^j$  is an output node, then equation 71 implies to compute  $\frac{\partial o^r}{\partial o_u^m}$  for all  $N^r \in Out$ . Therefore, we must compute  $\frac{\partial o^j}{\partial o_u^m}$ .

If  $N^j$  is not an output node, the general equation 72 must be applied. But this equation is nearly the same as equation 59 and implies to compute the same first order differentials. Therefore, we can apply the result already obtained and conclude that we need to compute  $\frac{\partial o^j}{\partial o^m}$ .

Therefore, computing the Hessian matrix of  $\mathcal{E}$  with the pure back-propagation method implies to compute  $\frac{\partial o^j}{\partial o^m}$  for each  $N^j \in \mathcal{N}$  and for each  $N^m \in P^+(j)$ .

Moreover, it is obvious that  $\frac{\partial \mathcal{E}}{\partial o^m}$  as to be computed for each  $N^m \in \mathcal{N}$ , because of the use of equations 45 and 47. Equation 45 implies also to compute  $\frac{\partial o^j}{\partial w^m}$  for all  $N^j \notin Out$  and all  $N^m \in P^+(j)$ .

## 8.2 Complexity

As explained recalled in section 2, the first order derivatives can be computed in a generalized neural networks with two algorithms: a direct method and a back-propagation method. We have demonstrated in [6] that for classical architectures (MLP, RBF networks, etc.) the back-propagation algorithm is faster than the direct one for computing the differential of the error made by the network with respect to its weights.

In the current case, we want to compute the differential of the output of each node with respect to the outputs of all its predecessors. In this particular case, nothing allows to guess if the back-propagation will be faster than the direct algorithm. The purpose of this section is to study the relative complexity of both methods in order to choose the fastest one.

As in the Neurocolt report, we will not include in the complexity the time needed to compute the local first order differentials (such as  $\frac{\partial N^l}{\partial x^{r(j,l)}}$ ), as they are needed by both methods.

### 8.2.1 Direct method

The direct method is based on the following general equation:

$$\frac{\partial o^{l \rightarrow k}}{\partial o^k}(x, w, o^k(x, w)) = \sum_{N^j \in P(l) \cap S^*(k)} \frac{\partial N^l}{\partial x^{r(j,l)}}(i^l(x, w)) \frac{\partial o^{j \rightarrow k}}{\partial o^k}(x, w, o^k(x, w)) \quad (117)$$

This equation is obtained by the application of the chain rule to the recursive definition of  $o^{l \rightarrow k}(x, w, \tau^k)$ :

$$o^{l \rightarrow k}(x, w, \tau^k) = N^l \left( o^{P(l)1 \rightarrow k}(x, w, \tau^k), \dots, o^{P(l)p^l \rightarrow k}(x, w, \tau^k), w^l \right)$$

Equation 117 has a quite ‘‘classical’’ form, and the methods used in previous sections allows to conclude that its equation cost is:

$$(8.2.i) \quad |O^l| |O^k| \left( 2 \left| I_{S^+(k)}^l \right| + \delta_{N^k \in P(l)} - 1 \right)$$

The variable cost of this equation is implied by its recursive aspect: in order to compute  $\frac{\partial o^{l \rightarrow k}}{\partial o^k}$ , we need  $\frac{\partial o^{j \rightarrow k}}{\partial o^k}$  for each  $N^j \in P^+(j) \cap S^*(k)$ . Therefore, the variable cost of equation 117 will be obtained with the total cost of the same equation applied to predecessors of  $N^l$ .

But we are interested in the particular case in which we need  $\frac{\partial o^{l \rightarrow k}}{\partial o^k}$  for each  $N^l$  and for each  $N^k \in P^+(l)$ . The total cost of this calculation is in fact simply the sum of the variable costs of equation 117 applied to different  $N^l$  and  $N^k$ .

We need also to add to this complexity the one implied by the computation of  $\frac{\partial \mathcal{E}^{\rightarrow k}}{\partial o^k}$ . If  $N^k \in Out$ ,  $\frac{\partial \mathcal{E}^{\rightarrow k}}{\partial o^k}$  is simply a local differential. When  $N^k \notin Out$ , the differential is obtained by the following equation:

$$\frac{\partial \mathcal{E}^{\rightarrow k}}{\partial o^k}(x, w, o^k(x, w)) = \sum_{N^j \in Out \cap S^+(k)} \frac{\partial \mathcal{E}}{\partial x^{Out(j)}}(G^k(x, w)) \frac{\partial o^{j \rightarrow k}}{\partial o^k}(x, w, o^k(x, w)) \quad (118)$$

Once again, this equation has a ‘‘classical’’ form and its equation cost is:

$$(8.2.ii) \quad |O^k| \left( 2 \sum_{N^j \in Out \cap S^+(k)} |O^j| - 1 \right)$$

Moreover, we need to include the cost of the computation of the  $\frac{\partial o^l}{\partial w^k}$  for all  $N^l \notin Out$  and for all  $N^k \in P^+(l)$ . For this computation, two alternate methods can be used:

1. we can make a new direct computation starting from nothing ;
2. we can use property 4 in order to compute  $\frac{\partial o^l}{\partial w^k}$  we the help of  $\frac{\partial o^{l \rightarrow k}}{\partial o^k}$ .

It is obvious that the total cost of the first method is:

$$(8.2.iii) \quad \sum_{N^l \notin Out} |O^l| \sum_{N^k \in P^+(l)} |W^k| \left( 2 |I_{S^+(k)}^l| \delta_{N^k \in P(l)} - 1 \right)$$

For the second method, the local equation 24 which computes  $\frac{\partial o^l}{\partial w^k}$  has obviously an equation cost of  $|W^k| |O^l| (2|O^k| - 1)$  and therefore, the total cost of the second method is:

$$(8.2.iv) \quad \sum_{N^l \notin Out} |O^l| \sum_{N^k \in P^+(l)} |W^k| (2|O^k| - 1)$$

In the general case, the methods cannot be compared, but when we assume that  $|O^l| = 1$  for all  $N^l$ , we have obviously  $|I_{S^+(k)}^l| \geq |O^k|$  and therefore the second method is the most efficient one. **We assume in the rest of the report that the second method is always chosen.** Moreover, the computation cost of the chosen method has not to be included in the total cost because the back-propagation approach will also use the same method to compute the  $\frac{\partial o^l}{\partial w^k}$ , therefore, as far as the comparison are concerned, we will forget the cost of the computation of the  $\frac{\partial o^l}{\partial w^k}$ .

Therefore, the direct method implies the following cost:

$$D_{\text{first}}(\mathcal{E}) = \sum_{N^l \in \mathcal{N}} |O^l| \sum_{N^k \in P^+(l)} |O^k| \left( 2 |I_{S^+(k)}^l| + \delta_{N^k \in P(l)} - 1 \right) + \sum_{N^l \notin Out} |O^l| (2|O_{S^+(l)}| - 1) \quad (119)$$

## 8.2.2 Back-propagation

The back-propagation method is based on equation 29. Once again, we have a very common equation form. We obtain therefore the following equation cost (when computing  $\frac{\partial o^{l \rightarrow k}}{\partial o^k}$ ):

$$(8.2.v) \quad |O^l| |O^k| \left( 2 |O_{P^+(l)}^{S(k)}| + \delta_{N^k \in P(l)} - 1 \right)$$

Equation 29 can be used to compute  $\frac{\partial o^{l \rightarrow k}}{\partial o^k}$  as long as we know  $\frac{\partial o^{l \rightarrow j}}{\partial o^j}$  for each  $N^j \in S(k)$ . Computing  $\frac{\partial o^{l \rightarrow k}}{\partial o^k}$  for all  $N^k \in P^+(l)$  implies therefore the following total cost:

$$(8.2.vi) \quad |O^l| \sum_{N^k \in P^+(l)} |O^k| \left( 2 |O_{P^+(l)}^{S(k)}| + \delta_{N^k \in P(l)} - 1 \right)$$

Finally, the total cost for computing  $\frac{\partial o^{l \rightarrow k}}{\partial o^k}$  for all  $N^l \in \mathcal{N}$  and for all  $N^k \in P^+(l)$  is:

$$(8.2.vii) \quad \sum_{N^l \in \mathcal{N}} |O^l| \sum_{N^k \in P^+(l)} |O^k| \left( 2 |O_{P^+(l)}^{S(k)}| + \delta_{N^k \in P(l)} - 1 \right)$$

We need also to add to this computation the cost for computing  $\frac{\partial \mathcal{E} \rightarrow k}{\partial o^k}$  with the back-propagation. This computation is based on equations 33 (which implies no computation) and 33. This last equation is valid for  $N^k \notin Out$  and has an equation cost of:

$$(8.2.viii) \quad |O^k| (2 |O^{S(k)}| - 1)$$

The total cost is therefore:

$$(8.2.ix) \quad \sum_{N^k \notin Out} |O^k| (2 |O^{S(k)}| - 1)$$

Finally, the total cost for the computation of the first order differentials is (we forget the cost of  $\frac{\partial o^l}{\partial w^k}$  computation, as explained in the previous section):

$$B_{\text{first}}(\mathcal{E}) = \sum_{N^l \in \mathcal{N}} |O^l| \sum_{N^k \in P^+(l)} |O^k| \left( 2 |O_{P^+(l)}^{S(k)}| + \delta_{N^k \in P(l)} - 1 \right) + \sum_{N^k \notin Out} |O^k| (2 |O^{S(k)}| - 1) \quad (120)$$

### 8.2.3 Comparison

Once again, it is not possible to compare directly both formulae. In fact, as we prove in the following section, even for simple architecture such as MLP or RBF, the speed of both algorithms depends on the number of nodes in each layer: in some cases the back-propagation algorithm is faster than the direct one and in others cases the contrary is true.

## 9 A practical analysis

In this section, we analyze the case of “every day” neural networks. In fact, we compute the theoretical complexity of Hessian calculation for the error made by a common neural network, such as a MLP or a RBF network.

### 9.1 Scalar output

In common neural networks, neurons are scalar functions: the output of a neuron is a real number. This means that  $|O^k| = 1$  for each neuron of the network. This assumption simplifies the complexity formulae:

- We have  $|I_A^l| = \sum_{N^j \in P(l) \cap A} |O^j|$ . Therefore:

$$|I_A^l| = |P(l) \cap A|$$

- We have also:

$$|O_B^A| = |A \cap B|,$$

and

$$|O_A| = |Out \cap A|$$

Unfortunately, the complexity depends strongly on the structure of the graph and therefore, the scalar output assumption is not sufficient to allow a comparison of the different algorithms. As explained in subsection 6.3.2, it does not simplify the problem of the hybrid method computation order.

## 9.2 Totally connected layered architectures

The most common architecture for neural networks is the totally connected layered architecture used by the Multi-Layer Perceptron. This strong assumption on the structure of the graph used with the scalar output assumption greatly simplifies the complexity formulae.

### 9.2.1 General assumptions and preliminary results

General assumptions on the architecture:

- the network has  $N$  layers;
- layer  $L_k$  has  $l_k$  neurons;
- each neuron of layer  $L_k$  receives the scalar outputs of layer  $L_{k-1}$  neurons and have therefore  $l_{k-1}$  scalar inputs;
- each neuron of layer  $L_1$  has one input which dimension is  $l_0$  (it does not depend on the neuron).

We have the following simple results:

- when  $N^k \in L_p$  ( $p < N$ ),  $S(k) = L_{p+1}$  and  $S^+(k) = \cup_{k=p+1}^N L_k$ ;
- when  $N^k \in L_p$  ( $p > 1$ ),  $P(k) = L_{p-1}$  and  $P^+(k) = \cup_{k=1}^{p-1} L_k$ .

Finally, we introduce two new notations:

$$\sum_{N^k \in L_p} |W^k| = |W_p|,$$

and

$$\sum_{N^k \in L_p} |W^k|^2 = |W_p^2|$$

### 9.2.2 The direct method

The direct method is based on a recursive equation and equation 85 allows to write (using the fact that  $|O^l| = 1$ ):

$$(9.2.i) \quad H_d^l = \sum_{N^k \in P(l)} H_d^k + \mathcal{D}_d^l$$

Let  $N^l$  be an arbitrary node of the network, with  $N^l \in L_q$ . We have:

- if  $q = 1$ ,  $N^l$  is an input node. We have  $H_d^l = 0$ .
- if  $q > 1$ :

We have  $P^+(l) = \cup_{k=1}^{q-1} L_k$ , therefore, with the scalar output assumption, we have:

$$(9.2.ii) \quad \mathcal{D}^l = \sum_{k=1}^{q-1} \sum_{N^j \in L_k} |W^j| \left( \sum_{r=1}^{q-1} \sum_{N^m \in L_r} |W^m| \left( \min(|P(l) \cap S^*(m)|, |P(l) \cap S^*(j)|) \right. \right. \\ \left. \left. + |P(l) \cap S^*(m) \cap S^*(j)| - \frac{1}{2} \right) \right. \\ \left. + |W^j| \left( 2|P(l) \cap S^*(j)| - \frac{1}{2} \right) \right. \\ \left. + (|I^l| + |W^l|) (2|P(l) \cap S^*(j)| - 1) \right)$$

Let  $N^j$  be a node belonging to layer  $L_k$ :

- if  $k < q - 1$ , we have  $P(l) \cap S^*(j) = P(l) = L_{q-1}$ ;
- if  $k = q - 1$ , we have  $P(l) \cap S^*(j) = \{N^j\}$ ;
- if  $k > q - 1$ , we have  $P(l) \cap S^*(j) = \emptyset$ .

Let  $N^j$  be a node belonging to layer  $L_k$  and  $N^m$  be a node belonging to layer  $L_r$ :

- if  $k < q - 1$  and  $r < q - 1$ , we have  $P(l) \cap S^*(j) \cap S^*(m) = P(l) = L_{q-1}$ ;
- if  $k = q - 1$  and  $r < q - 1$ , we have  $P(l) \cap S^*(j) \cap S^*(m) = P(l) \cap S^*(j) = \{N^j\}$ ;
- if  $k = q - 1$  and  $r = q - 1$ , we have  $P(l) \cap S^*(j) \cap S^*(m) = \emptyset$  if  $N^m \neq N^j$  and  $P(l) \cap S^*(j) \cap S^*(m) = \{N^j\}$  when  $N^m = N^j$ ;
- if  $k > q - 1$  or  $r > q - 1$ , we have  $P(l) \cap S^*(j) \cap S^*(m) = \emptyset$ .

Therefore we have:

- if  $k < q - 1$ :

$$\begin{aligned} \sum_{r=1}^{q-1} \sum_{N^m \in L_r} |W^m| \min(|P(l) \cap S^*(m)|, l_{q-1}) &= \sum_{r=1}^{q-2} \sum_{N^m \in L_r} |W^m| l_{q-1} + \sum_{N^m \in L_{q-1}} |W^m| \\ &= l_{q-1} \sum_{r=1}^{q-2} |W_r| + |W_{q-1}| \end{aligned}$$

- if  $k = q - 1$ :

$$\begin{aligned} \sum_{r=1}^{q-1} \sum_{N^m \in L_r} |W^m| \min(|P(l) \cap S^*(m)|, 1) &= \sum_{r=1}^{q-2} \sum_{N^m \in L_r} |W^m| + \sum_{N^m \in L_{q-1}} |W^m| \\ &= \sum_{r=1}^{q-1} |W_r| \end{aligned}$$

Therefore, we have:

$$\begin{aligned} &\sum_{k=1}^{q-1} \sum_{N^j \in L_k} |W^j| \sum_{r=1}^{q-1} \sum_{N^m \in L_r} |W^m| \min(|P(l) \cap S^*(m)|, |P(l) \cap S^*(j)|) \\ &= \sum_{k=1}^{q-2} |W_k| \left( l_{q-1} \sum_{r=1}^{q-2} |W_r| + |W_{q-1}| \right) + |W_{q-1}| \left( \sum_{r=1}^{q-1} |W_r| \right) \\ &= (l_{q-1} - 1) \left( \sum_{k=1}^{q-2} |W_k| \right)^2 + \left( \sum_{k=1}^{q-1} |W_k| \right)^2 \end{aligned}$$

We have also:

$$\begin{aligned} &\sum_{k=1}^{q-1} \sum_{N^j \in L_k} |W^j| \sum_{r=1}^{q-1} \sum_{N^m \in L_r} |W^m| |P(l) \cap S^*(m) \cap S^*(j)| \\ &= \sum_{k=1}^{q-2} \sum_{N^j \in L_k} |W^j| \left( l_{q-1} \sum_{r=1}^{q-2} \sum_{N^m \in L_r} |W^m| + \sum_{N^m \in L_{q-1}} |W^m| \right) \end{aligned}$$

$$\begin{aligned}
& + \sum_{N^j \in L_{q-1}} |W^j| \left( \sum_{r=1}^{q-2} \sum_{N^m \in L_r} |W^m| + |W^j| \right) \\
= & l_{q-1} \left( \sum_{k=1}^{q-2} |W_k| \right)^2 + 2|W_{q-1}| \sum_{k=1}^{q-2} |W_k| + \sum_{N^j \in L_{q-1}} |W^j|^2
\end{aligned}$$

As  $|I^l| = l_{q-1}$ , we have:

$$\begin{aligned}
& \sum_{k=1}^{q-1} \sum_{N^j \in L_k} |W^j| \left( (|I^l| + |W^l|) (2|P(l) \cap S^*(j)| - 1) \right) \\
= & (|W^l| + l_{q-1}) \left( (2l_{q-1} - 1) \sum_{k=1}^{q-2} |W_k| + |W_{q-1}| \right)
\end{aligned}$$

We have also:

$$\begin{aligned}
& \sum_{k=1}^{q-1} \sum_{N^j \in L_k} |W^j|^2 \left( 2|P(l) \cap S^*(j)| - \frac{1}{2} \right) \\
= & \left( 2l_{q-1} - \frac{1}{2} \right) \sum_{k=1}^{q-2} \sum_{N^j \in L_k} |W^j|^2 + \frac{3}{2} \sum_{N^j \in L_{q-1}} |W^j|^2
\end{aligned}$$

Finally, we have:

$$\begin{aligned}
(9.2.iii) \quad \mathcal{D}^l & = (2l_{q-1} - 1) \left( \sum_{k=1}^{q-2} |W_k| \right)^2 + \frac{1}{2} \left( \sum_{k=1}^{q-1} |W_k| \right)^2 \\
& + (2|W_{q-1}| + (2l_{q-1} - 1) (|W^l| + l_{q-1})) \sum_{k=1}^{q-2} |W_k| + (|W^l| + l_{q-1}) |W_{q-1}| \\
& + \left( 2l_{q-1} - \frac{1}{2} \right) \sum_{k=1}^{q-2} \sum_{N^j \in L_k} |W^j|^2 + \frac{5}{2} \sum_{N^j \in L_{q-1}} |W^j|^2
\end{aligned}$$

For an error function, we have:

$$\begin{aligned}
(9.2.iv) \quad \mathcal{D}^{\mathcal{E}} & = \sum_{k=1}^N \sum_{N^j \in L_k} |W^j| \left( \sum_{r=1}^N \sum_{N^m \in L_r} |W^m| \left( \min(|Out \cap S^*(m)|, |Out \cap S^*(j)|) \right. \right. \\
& \quad \left. \left. + |Out \cap S^*(m) \cap S^*(j)| - \frac{1}{2} \right) \right. \\
& \quad \left. + |W^j| \left( 2|Out \cap S^*(j)| - \frac{1}{2} \right) \right. \\
& \quad \left. + l_N (2|Out \cap S^*(j)| - 1) \right)
\end{aligned}$$

Let  $N^j$  be a node belonging to layer  $L_k$ :

- if  $k < N$ , we have  $Out \cap S^*(j) = Out$ ;
- if  $k = N$ , we have  $Out \cap S^*(j) = \{N^j\}$ .

Let  $N^j$  be a node belonging to layer  $L_k$  and  $N^m$  be a node belonging to layer  $L_r$ :

- if  $k < N$  and  $r < N$ , we have  $Out \cap S^*(j) \cap S^*(m) = Out$ ;

- if  $k = N$  and  $r < N$ , we have  $Out \cap S^*(j) \cap S^*(m) = Out \cap S^*(j) = \{N^j\}$ ;
- if  $k = N$  and  $r = N$ , we have  $Out \cap S^*(j) \cap S^*(m) = \emptyset$  if  $N^m \neq N^j$  and  $Out \cap S^*(j) \cap S^*(m) = \{N^j\}$  when  $N^m = N^j$ .

We can apply to the error function the same method as for an inner node and we obtain finally:

$$(9.2.v) \quad \mathcal{D}^{\mathcal{E}} = (2l_N - 1) \left( \sum_{k=1}^{N-1} |W_k| \right)^2 + \frac{1}{2} \left( \sum_{k=1}^N |W_k| \right)^2 + \left( 2l_N - \frac{1}{2} \right) \sum_{k=1}^{N-1} \sum_{N^j \in L_k} |W^j|^2 \\ + l_N |W_N| + (2|W_N| + l_N (2l_N - 1)) \sum_{k=1}^{N-1} |W_k| + \frac{5}{2} \sum_{N^j \in L_N} |W^j|^2$$

The total computation cost is therefore:

$$H^{\mathcal{E}} = \mathcal{D}^{\mathcal{E}} + \sum_{k=2}^N \sum_{N^j \in L_k} \mathcal{D}^j$$

We have:

$$(9.2.vi) \quad \sum_{N^j \in L_q} \mathcal{D}^j = l_q (2l_{q-1} - 1) \left( \sum_{k=1}^{q-2} |W_k| \right)^2 + l_q \frac{1}{2} \left( \sum_{k=1}^{q-1} |W_k| \right)^2 \\ + (2l_q |W_{q-1}| + (2l_{q-1} - 1) (|W_q| + l_q l_{q-1})) \sum_{k=1}^{q-2} |W_k| + (|W_q| + l_q l_{q-1}) |W_{q-1}| \\ + l_q \left( 2l_{q-1} - \frac{1}{2} \right) \sum_{k=1}^{q-2} \sum_{N^j \in L_k} |W^j|^2 + \frac{5}{2} l_q \sum_{N^j \in L_{q-1}} |W^j|^2$$

### 9.2.3 The hybrid method

The case of the hybrid method is interesting because of the computation order problem. The first problem presented in subsection 6.2.3 is the following one: when we have chosen to compute  $\frac{\partial^2 \mathcal{E}}{\partial w_t^j \partial w_u^m}$  in this order, it might be faster to compute  $\frac{\partial^2 \mathcal{E}}{\partial w_t^k \partial w_u^m}$  where  $N^k \in S^+(j)$  in the opposite order, even if the order for  $N^j$  and  $N^m$  implies that  $\frac{\partial^2 \mathcal{E}}{\partial \sigma_t^k \partial w_u^m}$  is already known.

Let  $N^j$  be a node of layer  $L_k$  and  $N^m$  a node of layer  $L_r$ , with  $N^m \neq N^j$ . As  $\delta_{N^m \in P^+(j)} = \delta_{r < k}$ , we have

$$\mathcal{H}e^{\mathcal{E}}(w^j, w^m) = |W^j| |W^m| (1 + \delta_{r < k} (1 + 2|P(j) \cap S^*(m)|)) \\ \mathcal{H}e^{\mathcal{E}}(w^m, w^j) = |W^j| |W^m| (1 + \delta_{k < r} (1 + 2|P(m) \cap S^*(j)|))$$

When  $r < k$ , two cases can occur:

- if  $r = k - 1$ ,  $S^*(m) \cap P(j) = \{N^m\}$ ;
- if  $r < k - 1$ ,  $S^*(m) \cap P(j) = P(j)$ .

Therefore, we have:

$$\mathcal{H}e^{\mathcal{E}}(w^j, w^m) = |W^j| |W^m| (1 + \delta_{r < k} (3 + 2(l_{k-1} - 1)\delta_{r < k-1})) \\ \mathcal{H}e^{\mathcal{E}}(w^m, w^j) = |W^j| |W^m| (1 + \delta_{k < r} (3 + 2(l_{r-1} - 1)\delta_{k < r-1}))$$

Therefore, it is obvious that the order  $(N^j, N^m)$  is more efficient than the other one when  $N^j \in P^+(N^m)$ .

When  $N^m = N^j$ , the cost is:

$$\mathcal{H}e^{\mathcal{E}}(w^j, w^j) = 3|W^j|^2$$

Let us now assume that  $N^j \notin Out$ . In this case,  $|O^{S(j)}| = |S(j)| = |L_{k+1}| = l_{k+1}$  and therefore:

$$\mathcal{H}e^{\mathcal{E}}(o^j, w^m) = |W^m| \left( 2l_{k+1} + \left| L_{k+1} \cap \left( \bigcup_{p=r+1}^N L_p \right) \right| + 2 \sum_{N^q \in L_{k+1} \cap \left( \bigcup_{p=r+1}^N L_p \right)} \left| L_k \cap \left( \bigcup_{p=r+1}^N L_p \cup \{N^m\} \right) \right| - 1 \right)$$

The following cases can occur:

- if  $r > k$ :

In this case,  $L_{k+1} \cap \left( \bigcup_{p=r+1}^N L_p \right) = \emptyset$  and therefore we have:

$$\mathcal{H}e^{\mathcal{E}}(o^j, w^m) = |W^m| (2l_{k+1} - 1)$$

- if  $r = k$ :

In this case,  $L_{k+1} \cap \left( \bigcup_{p=r+1}^N L_p \right) = L_{k+1}$  and  $L_k \cap \left( \bigcup_{p=r+1}^N L_p \cup \{N^m\} \right) = \{N^m\}$ . Therefore, we have:

$$\mathcal{H}e^{\mathcal{E}}(o^j, w^m) = |W^m| (5l_{k+1} - 1)$$

- if  $r < k$ :

In this case,  $L_{k+1} \cap \left( \bigcup_{p=r+1}^N L_p \right) = L_{k+1}$  and  $L_k \cap \left( \bigcup_{p=r+1}^N L_p \cup \{N^m\} \right) = L_k$ . Therefore, we have:

$$\mathcal{H}e^{\mathcal{E}}(o^j, w^m) = |W^m| (3l_{k+1} + 2l_k l_{k+1} - 1)$$

When  $N^j \in Out$ , we obtain :

- when  $N^m \in Out$  :

$$\mathcal{H}e^{\mathcal{E}}(o^j, w^m) = |W^m|$$

- when  $N^m \notin Out$  :

$$\mathcal{H}e^{\mathcal{E}}(o^j, w^m) = |W^m| (2l_N - 1)$$

In fact, even in this simplified case, it remains very difficult to choose an efficient computation strategy. The reason of this is mainly that computing  $\frac{\partial^2 \mathcal{E}}{\partial w_i^m \partial w_u^m}$  implies to know  $\frac{\partial^2 \mathcal{E}}{\partial \sigma_q^k \partial w_u^m}$  for all  $N^k \in S^*(N^m)$ . But as shown before, computing  $\frac{\partial^2 \mathcal{E}}{\partial w_q^k \partial w_u^m}$  is less efficient than computing in the other order (on a local point of view) when  $N^k \in S^+(m)$ . Therefore, we have here two contradictory parts in the complexity.

Let us study more precisely what can happen (with  $N^j \in L_k$ ,  $N^m \in L_r$  and  $m \neq j$ ):

- when  $r < k$ , we need to know  $\frac{\partial^2 \mathcal{E}}{\partial \sigma_q^j \partial w_u^m}$  regardless of the computation strategy.
- when  $r = k$ , the local equations have the same complexity, and the orders are equivalent. Let us assume that we choose to compute  $\frac{\partial^2 \mathcal{E}}{\partial w_q^j \partial w_u^m}$ . This implies to compute  $\frac{\partial^2 \mathcal{E}}{\partial \sigma_q^j \partial w_u^m}$ . If we assume that  $|W^j| = |W^m|$  as long as  $N^j$  and  $N^m$  belongs to the same layer, the chosen order implies the same computation load as the alternate one. Moreover, the computation of  $\frac{\partial^2 \mathcal{E}}{\partial \sigma_q^j \partial w_u^m}$  is possible as long as we know  $\frac{\partial^2 \mathcal{E}}{\partial \sigma_q^k \partial w_u^m}$  for  $N^k \in S^+(j)$ , which has to be know regardless of the computation strategy. Therefore, the derivation order for nodes belonging to the same layer as no influence on the total complexity.

- the only important thing which remains to be chosen is in fact how to compute  $\frac{\partial^2 \mathcal{E}}{\partial w_q^j \partial w_u^m}$  when  $r < k$ . If we choose this order, we don't have to compute additional differential, but as shown before, the local computation is more complex than with the other order. But this alternate order implies to compute additional differentials ( $\frac{\partial^2 \mathcal{E}}{\partial \sigma_r^p \partial w_u^m}$  for  $N^p \in S^*(p) \setminus S^*(m)$ ).

Let us study the special case of a two-layer network. In this case, the only problem is two choose how to compute  $\frac{\partial^2 \mathcal{E}}{\partial w_q^j \partial w_u^m}$  for  $N^j \in L_1$  and  $N^m \in L_2$ . If we use the order  $(N^m, N^j)$ , the implied cost is:

$$\mathcal{H}e^{\mathcal{E}}(w^m, w^j) = 4|W^j||W^m|$$

With the alternate order, the local cost is only:

$$\mathcal{H}e^{\mathcal{E}}(w^j, w^m) = |W^j||W^m|,$$

but we must compute the additional differentials  $\frac{\partial^2 \mathcal{E}}{\partial \sigma_q^j \partial w_u^m}$  and  $\frac{\partial^2 \mathcal{E}}{\partial \sigma_r^k \partial w_u^m}$  for all  $N^k \in L_2$ . This last group of differentials can be partly used to compute  $\frac{\partial^2 \mathcal{E}}{\partial w_r^k \partial w_u^m}$ . The main problem is that in order to compute  $\frac{\partial^2 \mathcal{E}}{\partial w_r^k \partial w_u^m}$  for all pair of nodes  $(N^k, N^m) \in L_2^2$ , we need only to know one part of the differentials (for instance  $\frac{\partial^2 \mathcal{E}}{\partial \sigma_r^k \partial w_u^m}$  for  $k \geq m$ ), whereas in the current case, we need all these differentials. Therefore, the cost difference must be included in the comparison. For a given  $m$  and a given  $k$ , the complexity of  $\frac{\partial^2 \mathcal{E}}{\partial \sigma_r^k \partial w_u^m}$  computation is  $|W^m|$ . If  $|W(k)|$  is the common number of parameters of nodes belonging to  $L_k$ , the total cost is therefore  $l_2^2 |W(2)|$ . The limited set of differentials which have to be computed regardless of the computation strategy implies only a cost of  $\frac{l_2(l_2+1)}{2} |W(2)|$ . Therefore, the additional cost is:

$$\frac{l_2(l_2 - 1)}{2} |W(2)|$$

We have also to add the cost of the computation of  $\frac{\partial^2 \mathcal{E}}{\partial \sigma_q^j \partial w_u^m}$  which is:

$$\mathcal{H}e^{\mathcal{E}}(\sigma^j, w^m) = |W^m|(2l_2 - 1)$$

Finally, the total cost of this order is:

$$l_1 l_2 |W(1)||W(2)| + \frac{l_2(l_2 - 1)}{2} |W(2)| + l_1 l_2 |W(2)|(2l_2 - 1)$$

We have therefore to compare the following values:

$$\begin{aligned} A &= 3l_1 l_2 |W(1)||W(2)| \\ B &= \frac{l_2(l_2 - 1)}{2} |W(2)| + l_1 l_2 |W(2)|(2l_2 - 1) \end{aligned}$$

Which is equivalent to the comparison between the two following values:

$$\begin{aligned} C &= 3l_1 |W(1)| \\ D &= \frac{l_2 - 1}{2} + l_1(2l_2 - 1) \end{aligned}$$

But as  $|W(1)| = l_0 + 1$  for a standard MLP,  $C$  does not take into account  $l_2$  and  $D$  does not take into account  $l_0$ . Therefore, it's easy to build networks in which the first computation strategy is faster than the other, and other networks in which the contrary is true. In traditional classification application, we have in general  $l_0 \geq l_2$ , which allows to conclude that  $D \leq C$  and therefore that the second computation strategy is faster than the first one. In this case, we can compute the total cost of this computation strategy.

- in the chosen strategy, the computation of  $\frac{\partial^2 \mathcal{E}}{\partial w_q^j \partial w_u^m}$  is always performed at a minimal equation cost of  $|W^m||W^j|$ , except for  $j = m$ . This gives the following total cost:

$$\sum_{N^m \in L_1} |W^m| \left( \sum_{N^j \in \mathcal{N}} |W^j| + 2|W^m| \right) + \sum_{N^m \in L_2} |W^m| \left( \sum_{N^j \in L_2} |W^j| + 2|W^m| \right),$$

which is equal to:

$$|W_1|^2 + |W_2|^2 + |W_1||W_2| + 2|W_1^2| + 2|W_2^2|$$

- as obtained before, the additional cost for changing locally the derivation order is:

$$\frac{l_2(l_2 - 1)}{2} |W(2)| + l_1 l_2 |W(2)|(2l_2 - 1)$$

- as explained before, the computation of  $\frac{\partial^2 \mathcal{E}}{\partial o_q^j \partial w_u^m}$  when  $N^j$  and  $N^m$  belongs to the same layer as only to be done for each *unordered* pair of node and produces the following complexities:

– for layer  $L_1 = In$ :

$$(2l_2 - 1) \frac{l_1(l_1 + 1)}{2} |W(1)|$$

– for layer  $L_2 = Out$ :

$$\frac{l_2(l_2 + 1)}{2} |W(2)|$$

- finally, we must include the cost of the computation of  $\frac{\partial^2 \mathcal{E}}{\partial o_q^j \partial w_u^m}$  when  $N^m \in L_1$  and  $N^j \in Out$ , which is:

$$l_2(2l_2 - 1)|W_1|$$

The total cost is therefore:

$$(9.2.vii) \quad \mathcal{H}^{\mathcal{E}} =$$

$$|W_1|^2 + |W_2|^2 + |W_1||W_2| + 2|W_1^2| + 2|W_2^2| + (2l_2 - 1) \frac{l_1 + 1}{2} |W_1| + (l_1(2l_2 - 1) + l_2)|W_2|$$

#### 9.2.4 The pure back-propagation method

We shall recall that the pure back-propagation uses the same local formula as the hybrid method which complexity was given in the previous subsection. For the pure method,  $\frac{\partial^2 \mathcal{E}}{\partial w_t^j \partial w_u^m}$  can be computed only if  $N^m \notin S^+(j)$ . As  $N^j \in L_k$  and  $N^m \in L_r$ , this condition means that  $r \leq k$ .

The pure method uses also a second local formula which complexity is:

$$\mathcal{P}e^{\mathcal{E}}(o^j, w^m) = |W^m|$$

When  $N^j \notin Out$ , the *equation cost* of the recursive formula is:

$$\mathcal{P}e^{\mathcal{E}}(o^j, o^m) =$$

$$2l_{k+1} + \left| L_{k+1} \cap \left( \bigcup_{p=r+1}^N L_p \right) \right| + \sum_{N^q \in L_{k+1} \cap \left( \bigcup_{p=r+1}^N L_p \right)} \left( 2 \left| L_k \cap \left( \bigcup_{p=r+1}^N L_p \cup \{N^m\} \right) \right| + \delta_{N^m \in L_k} \right) - 1$$

As  $N^m \notin S^+(j)$ , the following case can occur:

- if  $r = k$ :

In this case,  $L_{k+1} \cap (\cup_{p=r+1}^N L_p) = L_{k+1}$  and  $L_k \cap (\cup_{p=r+1}^N L_p \cup \{N^m\}) = \{N^m\}$ . Moreover  $\delta_{N^m \in L_k} = 1$ . Therefore, we have:

$$\mathcal{P}e^{\mathcal{E}}(\sigma^j, o^m) = 6l_{k+1} - 1$$

- if  $r < k$ : In this case,  $L_{k+1} \cap (\cup_{p=r+1}^N L_p) = L_{k+1}$ ,  $L_k \cap (\cup_{p=r+1}^N L_p \cup \{N^m\}) = L_k$  and  $\delta_{N^m \in L_k} = 0$ . Therefore, we have:

$$\mathcal{P}e^{\mathcal{E}}(\sigma^j, o^m) = 3l_{k+1} + 2l_k l_{k+1} - 1$$

When  $N^j \in Out$ , the *equation cost* is less complex and we have:

$$\mathcal{P}e^{\mathcal{E}}(\sigma^j, o^m) = 2 \left| Out \cap \left( \bigcup_{p=r+1}^N L_p \right) \right| + \delta_{N^m \in Out} - 1$$

Therefore, we have:

- if  $r \neq N$  (i.e.,  $N^m \notin Out$ ):

In this case,  $Out \subset (\cup_{p=r+1}^N L_p)$  and therefore:

$$\mathcal{P}e^{\mathcal{E}}(\sigma^j, o^m) = 2l_N - 1$$

- if  $r = N$ :

In this case,  $S^+(m) = \emptyset$  and therefore:

$$\mathcal{P}e^{\mathcal{E}}(\sigma^j, o^m) = 0$$

When  $N^m$  and  $N^j$  are independent nodes, the computation order of  $\frac{\partial^2 \mathcal{E}}{\partial w_t^j \partial w_u^m}$  can be optimized. In a multi-layered architecture,  $N^m$  and  $N^j$  are independent when they belong to the same layer. We have the following possibilities:

- $N^j \in L_k$  and  $k \neq N$ :

In this case, the global *equation cost* can be:

$$\begin{aligned} \mathcal{P}e^{\mathcal{E}}(\sigma^j, o^m) + \mathcal{P}e^{\mathcal{E}}(\sigma^j, w^m) + \mathcal{H}e^{\mathcal{E}}(w^j, w^m) &= 6l_{k+1} - 1 + |W^m| + |W^j| |W^m| \\ \mathcal{P}e^{\mathcal{E}}(o^m, \sigma^j) + \mathcal{P}e^{\mathcal{E}}(o^m, w^j) + \mathcal{H}e^{\mathcal{E}}(w^m, w^j) &= 6l_{k+1} - 1 + |W^j| + |W^m| |W^j| \end{aligned}$$

Therefore, we have:

$$\mathcal{M}e^{\mathcal{E}}(w^j, w^m) = 6l_{k+1} - 1 + |W^m| |W^j| + \min(|W^j|, |W^m|)$$

As in general  $|W^m| = |W^j|$  when  $N^m$  and  $N^j$  belong to the same layer, the complexity does not depend on the chosen order.

- $N^j \in L_k$  and  $k = N$ :

In this case, the global *equation cost* can be:

$$\begin{aligned} \mathcal{P}e^{\mathcal{E}}(\sigma^j, o^m) + \mathcal{P}e^{\mathcal{E}}(\sigma^j, w^m) + \mathcal{H}e^{\mathcal{E}}(w^j, w^m) &= |W^m| + |W^j| |W^m| \\ \mathcal{P}e^{\mathcal{E}}(o^m, \sigma^j) + \mathcal{P}e^{\mathcal{E}}(o^m, w^j) + \mathcal{H}e^{\mathcal{E}}(w^m, w^j) &= |W^j| + |W^m| |W^j| \end{aligned}$$

Therefore, we have:

$$\mathcal{M}e^{\mathcal{E}}(w^j, w^m) = |W^m| |W^j| + \min(|W^j|, |W^m|)$$

The total computation cost is therefore:

$$\mathcal{P}^{\mathcal{E}} = \sum_{k=1}^N \sum_{N^j \in L_k} \left( \sum_{r=1}^{k-1} \sum_{N^m \in L_r} \mathcal{M}e^{\mathcal{E}}(w^j, w^m) + \mathcal{M}e^{\mathcal{E}}(w^j, w^j) + \frac{1}{2} \sum_{N^m \in L_k, m \neq j} \mathcal{M}e^{\mathcal{E}}(w^j, w^m) \right)$$

We have:

$$\begin{aligned} & \sum_{k=1}^N \sum_{N^j \in L_k} \sum_{r=1}^{k-1} \sum_{N^m \in L_r} \mathcal{H}e^{\mathcal{E}}(w^j, w^m) \\ &= \sum_{k=1}^N \sum_{N^j \in L_k} \left( \sum_{r=1}^{k-2} \sum_{N^m \in L_r} 2(1 + l_{k-1})|W^j||W^m| + \sum_{N^m \in L_{k-1}} 4|W^j||W^m| \right) \\ &= \sum_{k=1}^N |W_k| \left( 2(l_{k-1} + 1) \sum_{r=1}^{k-2} |W_r| + 4|W_{k-1}| \right) \end{aligned}$$

We have also:

$$\begin{aligned} \sum_{k=1}^N \sum_{N^j \in L_k} \sum_{r=1}^{k-1} \sum_{N^m \in L_r} \mathcal{P}e^{\mathcal{E}}(o^j, w^m) &= \sum_{k=1}^N \sum_{N^j \in L_k} \sum_{r=1}^{k-1} \sum_{N^m \in L_r} |W^m| \\ &= \sum_{k=1}^N l_k \sum_{r=1}^{k-1} |W_r| \end{aligned}$$

Moreover:

$$\begin{aligned} \sum_{k=1}^{N-1} \sum_{N^j \in L_k} \sum_{r=1}^{k-1} \sum_{N^m \in L_r} \mathcal{P}e^{\mathcal{E}}(o^j, o^m) &= \sum_{k=1}^{N-1} \sum_{N^j \in L_k} \sum_{r=1}^{k-1} \sum_{N^m \in L_r} (3l_{k+1} + 2l_k l_{k+1} - 1) \\ &= \sum_{k=1}^{N-1} l_k \sum_{r=1}^{k-1} l_r (3l_{k+1} + 2l_k l_{k+1} - 1), \end{aligned}$$

and:

$$\begin{aligned} \sum_{N^j \in L_N} \sum_{r=1}^{N-1} \sum_{N^m \in L_r} \mathcal{P}e^{\mathcal{E}}(o^j, o^m) &= \sum_{N^j \in L_N} \sum_{r=1}^{N-1} \sum_{N^m \in L_r} (2l_N - 1) \\ &= l_N (2l_N - 1) \sum_{r=1}^{N-1} l_r \end{aligned}$$

We have:

$$\begin{aligned} \sum_{k=1}^N \sum_{N^j \in L_k} \mathcal{M}e^{\mathcal{E}}(w^j, w^j) &= \sum_{k=1}^{N-1} \sum_{N^j \in L_k} (3|W^j|^2 + |W^j| + 6l_{k+1} - 1) + \sum_{N^j \in L_N} (3|W^j|^2 + |W^j|) \\ &= \sum_{k=1}^{N-1} l_k (6l_{k+1} - 1) + \sum_{k=1}^N \left( |W_k| + 3 \sum_{N^j \in L_k} |W^j|^2 \right) \end{aligned}$$

We have:

$$\sum_{k=1}^N \sum_{N^j \in L_k} \sum_{N^m \in L_k, m \neq j} \mathcal{M}i^{\mathcal{E}}(w^j, w^m)$$

$$\begin{aligned}
&= \sum_{k=1}^{N-1} \sum_{N^j \in L_k} \sum_{N^m \in L_k, m \neq j} (6l_{k+1} - 1 + |W^j||W^m| + \min(|W^j|, |W^m|)) \\
&\quad + \sum_{N^j \in L_N} \sum_{N^m \in L_N, m \neq j} (|W^j||W^m| + \min(|W^j|, |W^m|)) \\
&= \sum_{k=1}^{N-1} l_k(l_k - 1)(6l_{k+1} - 1) + \sum_{k=1}^N \left( |W_k|^2 - \sum_{N^j \in L_k} |W^j|^2 \right) + \sum_{k=1}^N \sum_{N^j \in L_k} \sum_{N^m \in L_k, m \neq j} \min(|W^j|, |W^m|)
\end{aligned}$$

Finally, we have:

$$\begin{aligned}
(9.2.viii) \quad \mathcal{P}^{\mathcal{E}} &= \\
&\sum_{k=1}^{N-1} \frac{l_k(l_k + 1)}{2} (6l_{k+1} - 1) + \sum_{k=1}^{N-1} l_k \left( l_N(2l_N - 1) + (3l_{k+1} + 2l_k l_{k+1} - 1) \sum_{r=1}^{k-1} l_r \right) \\
&\quad + \sum_{k=1}^N \left( |W_k| + \frac{1}{2}|W_k|^2 + \frac{5}{2} \sum_{N^j \in L_k} |W^j|^2 + 4|W_k||W_{k-1}| + (3l_{k-1} + 2) \sum_{r=1}^{k-2} |W_r| + l_k |W_{k-1}| \right) \\
&\quad + \frac{1}{2} \sum_{k=1}^N \sum_{N^j \in L_k} \sum_{N^m \in L_k, m \neq j} \min(|W^j|, |W^m|)
\end{aligned}$$

If the number of parameters is the same for all neurons belonging to the same layer, we have:

$$\frac{1}{2} \sum_{k=1}^N \sum_{N^j \in L_k} \sum_{N^m \in L_k, m \neq j} \min(|W^j|, |W^m|) = \frac{1}{2} \sum_{k=1}^N (l_k - 1) |W_k|,$$

and therefore, we have:

$$\begin{aligned}
(9.2.ix) \quad \mathcal{P}^{\mathcal{E}} &= \\
&\sum_{k=1}^{N-1} \frac{l_k(l_k + 1)}{2} (6l_{k+1} - 1) + \sum_{k=1}^{N-1} l_k \left( l_N(2l_N - 1) + (3l_{k+1} + 2l_k l_{k+1} - 1) \sum_{r=1}^{k-1} l_r \right) \\
&\quad + \sum_{k=1}^N \left( \frac{l_k + 1}{2} |W_k| + \frac{1}{2}|W_k|^2 + \frac{5}{2} \sum_{N^j \in L_k} |W^j|^2 + 4|W_k||W_{k-1}| + (3l_{k-1} + 2) \sum_{r=1}^{k-2} |W_r| + l_k |W_{k-1}| \right)
\end{aligned}$$

## 9.2.5 Comparison

### A two-layer neural network

Let us first consider a two-layer network. In this case, the complexity of the direct algorithm is:

$$H_d^{\mathcal{E}} = \mathcal{D}^{\mathcal{E}} + \sum_{N^k \in L_2} H_d^k = \mathcal{D}^{\mathcal{E}} + \sum_{N^k \in L_2} \mathcal{D}^k$$

We have:

$$\sum_{N^k \in L_2} \mathcal{D}^k = l_2 \left( \frac{1}{2}|W_1|^2 + l_1|W_1| + \frac{5}{2} \sum_{N^j \in L_1} |W^j|^2 \right) + |W^1||W^2|$$

We have also:

$$\begin{aligned}\mathcal{D}^\mathcal{E} &= (2l_2 - 1)|W_1|^2 + \frac{1}{2}(|W_1| + |W_2|)^2 + \left(2l_2 - \frac{1}{2}\right) \sum_{N^j \in L_1} |W^j|^2 + l_2|W_2| \\ &\quad + (2|W_2| + l_2(2l_2 - 1))|W_1| + \frac{5}{2} \sum_{N^j \in L_2} |W^j|^2\end{aligned}$$

Therefore, we have:

$$\begin{aligned}H_d^\mathcal{E} &= \frac{1}{2}(5l_2 - 1)|W_1|^2 + \frac{1}{2}(9l_2 - 1) \sum_{N^j \in L_1} |W^j|^2 + l_2(l_1 + 2l_2 - 1)|W_1| \\ &\quad + \frac{1}{2}|W_2|^2 + \frac{5}{2} \sum_{N^j \in L_2} |W^j|^2 + l_2|W_2| + 4|W_1||W_2|\end{aligned}$$

The cost of the pure back-propagation method is simply given by the following equation (if we assume that the number of parameters is the same for all neurons belonging to the same layer):

$$\begin{aligned}\mathcal{P}^\mathcal{E} &= \frac{1}{2}l_1(l_1 + 1)(6l_2 - 1) + l_1l_2(2l_2 - 1) + \frac{1}{2}(l_1 - 1)|W_1| + \frac{1}{2}(l_2 - 1)|W_2| \\ &\quad + |W_1| + \frac{1}{2}|W_1|^2 + \frac{5}{2} \sum_{N^j \in L_1} |W^j|^2 + |W_2| + \frac{1}{2}|W_2|^2 + \frac{5}{2} \sum_{N^j \in L_2} |W^j|^2 + 4|W_1||W_2| + l_2|W_1| \\ &= \frac{1}{2}|W_1|^2 + \frac{1}{2}(2l_2 + l_1 + 1)|W_1| + \frac{5}{2} \sum_{N^j \in L_1} |W^j|^2 + \frac{1}{2}(l_2 + 1)|W_2| + \frac{1}{2}|W_2|^2 + \frac{5}{2} \sum_{N^j \in L_2} |W^j|^2 \\ &\quad + 4|W_1||W_2| + \frac{1}{2}l_1(l_1 + 1)(6l_2 - 1) + l_1l_2(2l_2 - 1)\end{aligned}$$

### “Huge” number of parameters

In general, the parameter number of a neuron depends on its input number. Moreover, in general, there are fewer output neurons than the number of inputs of the network. Therefore, in a two layer neural network, we can assume that  $|W_1| > |W_2|$ . This is the case for a MLP or a RBF net in which the input number is larger than the output number. The difference between the two values is more important for a wavelet network used for function approximation, in which the neurons of the first layer have a huge number of parameters whereas the output neurons are only linear neurons.

As  $|W_k|^2 \gg \sum_{N^j \in L_k} |W^j|^2$ , we can assume that the asymptotically dominating term of  $\mathcal{P}^\mathcal{E}$  and  $H_d^\mathcal{E}$  is  $|W_1|^2$ . In  $H_d^\mathcal{E}$ , we have  $(\frac{5}{2}l_2 - \frac{1}{2})|W_1|^2$  and in  $\mathcal{P}^\mathcal{E}$ , we have  $\frac{1}{2}|W_1|^2$ . Therefore we can assume that  $H_d^\mathcal{E}$  is asymptotically bigger than  $\mathcal{P}^\mathcal{E}$  and that the back-propagation is faster than the direct algorithm when the number of parameter is large.

### “every day” neural networks

In the case of a MLP or a RBF, when  $N^m \in L_j$ ,  $|W^m| = l_{j-1} + 1$ . Therefore, we have:

$$\begin{aligned}\mathcal{P}^\mathcal{E} &= l_1^2 \left( \frac{l_2^2}{2} + \frac{l_0^2}{2} + 4l_0l_2 + \frac{19l_2}{2} + \frac{3l_0}{2} + \frac{1}{2} \right) \\ &\quad + l_1 \left( \frac{7l_2^2}{2} + \frac{5l_0^2}{2} + 5l_0l_2 + \frac{25l_2}{2} + \frac{11l_0}{2} + 1 \right) \\ &\quad + \frac{l_2^2}{2} + 3l_2 \\ &= l_1^2 \mathcal{P}_2^\mathcal{E} + l_1 \mathcal{P}_1^\mathcal{E} + \mathcal{P}_0^\mathcal{E},\end{aligned}$$

and

$$H_d^\mathcal{E} = l_1^2 \left( \frac{5l_2l_0^2}{2} + \frac{l_2^2}{2} - \frac{l_0^2}{2} + 10l_0l_2 + 10l_2 - l_0 - \frac{1}{2} \right)$$

$$\begin{aligned}
& + l_1 \left( \frac{9l_2 l_0^2}{2} + 2l_2^2 l_0 + 4l_2^2 - \frac{l_0^2}{2} + 12l_2 l_0 + \frac{25l_2}{2} - l_0 - \frac{1}{2} \right) \\
& + \frac{3l_2^2}{2} + \frac{5l_2}{2} \\
& = l_1^2 H_2^\mathcal{E} + l_1 H_1^\mathcal{E} + H_0^\mathcal{E}
\end{aligned}$$

As  $l_0 \geq 1$  and  $l_2 \geq 1$ , we have:

$$\begin{aligned}
\frac{5l_2 l_0^2}{2} - \frac{l_0^2}{2} & \geq \frac{3l_2 l_0^2}{2} + \frac{l_0^2}{2} \\
10l_2 l_0 - l_0 - \frac{1}{2} & \geq \frac{13l_0 l_2}{2} + \frac{3l_0}{2} + \frac{1}{2}
\end{aligned}$$

Therefore, we have:

$$H_2^\mathcal{E} \geq \frac{3l_2 l_0^2}{2} + \frac{l_0^2}{2} + \frac{l_2^2}{2} + \frac{13l_0 l_2}{2} + \frac{3l_0}{2} + 10l_2 + \frac{1}{2},$$

and therefore:

$$H_2^\mathcal{E} \geq \mathcal{P}_2^\mathcal{E}$$

We have also:

$$\begin{aligned}
12l_2 l_0 - l_0 & \geq 5l_0 l_2 + \frac{l_0 l_2}{2} + \frac{11l_0}{2} \\
\frac{9l_2 l_0^2}{2} - \frac{l_0^2}{2} & \geq \frac{5l_0^2}{2} + \frac{3l_2 l_0^2}{2} \\
2l_2^2 l_0 - \frac{1}{2} & \geq \frac{l_2^2 l_0}{2} + 1
\end{aligned}$$

Therefore:

$$\begin{aligned}
H_1^\mathcal{E} & \geq 4l_2^2 + \frac{5l_0^2}{2} + 5l_0 l_2 + \frac{25l_2}{2} + \frac{11l_0}{2} + 1 + \frac{l_0 l_2}{2} + \frac{3l_2 l_0^2}{2} + \frac{l_2^2 l_0}{2} \\
H_1^\mathcal{E} & \geq \mathcal{P}_1^\mathcal{E}
\end{aligned}$$

Finally,  $l_2^2 \geq l_2$  implies obviously that:

$$H_0^\mathcal{E} \geq \mathcal{P}_0^\mathcal{E}$$

Therefore, for all  $l_1 \geq 0$ ,  $H_d^\mathcal{E} \geq \mathcal{P}^\mathcal{E}$ : in an “every day” 2 layer neural network, the pure back-propagation algorithm is faster than the direct method for computing the Hessian of the error made by the network.

### Arbitrary number of layer for “every day” neural networks

We assume in this part that when  $N^m \in L_j$ ,  $|W^m| = l_{j-1} + 1$ . In order to simplify the comparison of the complexities in the general case, we split both formulae into several parts.

The total complexity of the direct method,  $H_d^\mathcal{E}$ , is split into the following parts:

- $A = (2l_N - 1) \left( \sum_{q=1}^{N-1} |W_q| \right)^2$
- $B = \frac{1}{2} \left( \sum_{q=1}^N |W_q| \right)^2$
- $C = (2l_N - \frac{1}{2}) \sum_{q=1}^{N-1} |W_q^2|$
- $D = l_N |W_N|$
- $E = 2|W_N| \sum_{q=1}^{N-1} |W_q|$
- $F = l_N (2l_N - 1) \sum_{q=1}^{N-1} |W_q|$

- $G = \frac{5}{2}|W_N^2|$
- $H = \sum_{q=3}^N l_q(2l_{q-1} - 1) \left( \sum_{k=1}^{q-2} |W_k| \right)^2$
- $I = \sum_{q=2}^N \frac{l_q}{2} \left( \sum_{k=1}^{q-1} |W_k| \right)^2$
- $J = \sum_{q=3}^N 2l_q |W_{q-1}| \sum_{k=1}^{q-2} |W_k|$
- $K = \sum_{q=3}^N l_q(2l_{q-1} - 1) l_{q-1} \sum_{k=1}^{q-2} |W_k|$
- $L = \sum_{q=3}^N |W_q|(2l_{q-1} - 1) \sum_{k=1}^{q-2} |W_k|$
- $M = \sum_{q=2}^N |W_q| |W_{q-1}|$
- $N = \sum_{q=2}^N l_q l_{q-1} |W_{q-1}|$
- $O = \sum_{q=3}^N l_q \left( 2l_{q-1} - \frac{1}{2} \right) \sum_{k=1}^{q-2} |W_k^2|$
- $P = \frac{5}{2} \sum_{q=2}^N l_q |W_{q-1}^2|$

For the pure back-propagation method, the complexity split into the following parts:

- $a = \sum_{q=1}^{N-1} l_q(l_q + 1) \left( 3l_{q+1} - \frac{1}{2} \right)$
- $b = \sum_{q=2}^{N-1} l_q(3l_{q+1} + 2l_q l_{q+1} - 1) \sum_{k=1}^{q-1} l_k$
- $c = l_N(2l_N - 1) \sum_{q=1}^{N-1} l_q$
- $d = \sum_{q=1}^N \frac{l_q+1}{2} |W_q|$
- $e = \frac{1}{2} \sum_{q=1}^N |W_q|^2$
- $f = \frac{5}{2} \sum_{q=1}^N |W_q^2|$
- $g = 4 \sum_{q=2}^N |W_q| |W_{q-1}|$
- $h = \sum_{q=3}^N (3l_{q-1} + 2) \sum_{r=1}^{q-2} |W_r|$
- $i = \sum_{k=2}^N l_k |W_{k-1}|$

Our goal is to prove that  $H_d^{\mathcal{E}} > \mathcal{P}^{\mathcal{E}}$ . We have the following results:

1. we have:

$$P + G = \frac{5}{2} \sum_{q=1}^N |W_q^2| + \frac{5}{2} \sum_{q=1}^{N-1} (l_k - 1) |W_k^2|$$

As the second part of this equation is positive (because  $l_k \geq 1$  for all  $k$ ), we have:

$$P + G \geq f$$

2. if  $Q$  is defined as:

$$Q = \sum_{k=1}^{N-1} \sum_{j=k+1}^N |W_k| |W_j|,$$

we have:

$$B = e + Q$$

3. we have:

$$Q = \sum_{k=2}^N |W_k| |W_{k-1}| + R,$$

where:

$$R = \sum_{k=1}^{N-2} \sum_{j=k+2}^N |W_k| |W_j|$$

We have also:

$$J = \sum_{q=3}^N 2l_q |W_{q-1}| |W_{q-2}| + S,$$

where:

$$S = \sum_{q=4}^N 2l_q |W_{q-1}| \sum_{k=1}^{q-3} |W_k|$$

As  $l_q \geq 1$ , we have:

$$J \geq \sum_{q=2}^{N-1} 2|W_q| |W_{q-1}| + S$$

Moreover, if  $T$  is given by:

$$T = 2|W_N| \sum_{k=1}^{N-2} |W_k|,$$

then we have:

$$E = 2|W_N| |W_{N-1}| + T$$

Therefore, we have finally:

$$Q + J + M + E \geq 4 \sum_{q=2}^N |W_q| |W_{q-1}| + R + S + T,$$

and therefore:

$$B + J + M + E \geq e + g + R + S + T$$

4. we have:

$$\left( \sum_{k=1}^{q-2} |W_k| \right)^2 = |W_{q-2}|^2 + 2|W_{q-2}| \sum_{k=1}^{q-3} |W_k| + \left( \sum_{k=1}^{q-3} |W_k| \right)^2,$$

therefore we have:

$$H = \sum_{q=3}^N l_q (2l_{q-1} - 1) (l_{q-3} + 1)^2 l_{q-2}^2 + U,$$

where  $U$  is defined by:

$$U = \sum_{q=4}^N l_q (2l_{q-1} - 1) \left( 2|W_{q-2}| \sum_{k=1}^{q-3} |W_k| + \left( \sum_{k=1}^{q-3} |W_k| \right)^2 \right)$$

As  $(l_{q-3} + 1)^2 \geq 4$  and  $2l_{q-1} - 1 \geq l_{q-1}$ , we have

$$H \geq \sum_{q=2}^{N-1} 4l_{q+1} l_q l_{q-1}^2 + U$$

Moreover, we have:

$$A = (2l_N - 1)|W_{N-1}|^2 + V,$$

where  $V$  is defined by:

$$V = (2l_N - 1) \left( 2|W_{N-1}| \sum_{k=1}^{N-2} |W_k| + \left( \sum_{k=1}^{N-2} |W_k| \right)^2 \right)$$

We have  $|W_{N-1}|^2 = l_{N-1}^2(l_{N-2} + 1)^2$  and therefore  $|W_{N-1}|^2 \geq 4l_{N-1}^2$ . Therefore, we have:

$$A \geq 4l_N l_{N-1}^2$$

We have also:

$$C = \left( 2l_N - \frac{1}{2} \right) \sum_{k=1}^{N-1} l_k (l_{k-1} + 1)^2$$

We have  $2l_N - \frac{1}{2} \geq \frac{3}{2}l_N$  and  $l_{k-1} + 1 \geq 2$ . Therefore we have:

$$\begin{aligned} C &\geq 3l_N \sum_{k=1}^{N-1} l_k (l_{k-1} + 1) \\ &\geq 3l_N \sum_{k=1}^{N-1} l_k l_{k-1} + 3l_N l_{N-1} + l_N \sum_{k=1}^{N-2} l_k, \end{aligned}$$

and therefore, if  $W = l_N \sum_{k=1}^{N-2} l_k$ , we have:

$$C \geq 3 \sum_{k=1}^N l_k l_{k-1} + W$$

Therefore, we obtain:

$$H + A + C \geq \sum_{k=2}^N 4l_k l_{k-1}^2 + 3 \sum_{k=1}^N l_k l_{k-1} + U + V + W$$

Therefore, we have:

$$H + A + C \geq \sum_{k=2}^N 3l_k l_{k-1}^2 + 3 \sum_{k=1}^N l_k l_{k-1} + U + V + W,$$

and finally, if  $X$  is:

$$X = \frac{1}{2} \sum_{k=2}^{N-2} l_k,$$

we obtain:

$$H + A + C \geq a + U + V + W + X$$

5. we have:

$$K = \sum_{q=3}^N l_q (2l_{q-1} - 1) l_{q-1} \sum_{k=1}^{q-2} l_k (l_{k-1} + 1)$$

Therefore:

$$K \geq 2 \sum_{q=3}^N l_q l_{q-1}^2 \sum_{k=1}^{q-2} l_k$$

We have:

$$O = \sum_{q=3}^N l_q \left( 2l_{q-1} - \frac{1}{2} \right) \sum_{k=1}^{q-2} (l_{k-1} + 1)^2 l_k$$

As we have  $(l_{k-1} + 1)^2 = l_{k-1}(l_{k-1} + 1) + l_{k-1} + 1$ , if we define  $Y$  as:

$$Y = \sum_{q=3}^N l_q \left( 2l_{q-1} - \frac{1}{2} \right) \sum_{k=1}^{q-2} (l_{k-1} + 1) l_k l_{k-1},$$

we have:

$$O = \sum_{q=3}^N l_q \left( 2l_{q-1} - \frac{1}{2} \right) \sum_{k=1}^{q-2} (l_{k-1} + 1) l_k + Y$$

As  $2l_{q-1} - \frac{1}{2} \geq \frac{3}{2}l_{q-1}$  and  $l_{k-1} + 1 \geq 2$ , we have:

$$O \geq 3 \sum_{q=3}^N l_q l_{q-1} \sum_{k=1}^{q-2} l_k + Y$$

Finally:

$$K + O \geq b + Y + Z,$$

where  $Z$  is defined as:

$$Z = \sum_{k=2}^{N-1} l_k \sum_{r=1}^{k-1} l_r$$

6. we have:

$$Y \geq \frac{3}{2} \sum_{q=3}^N l_{q-1} \sum_{k=1}^{q-2} |W_k|$$

Moreover:

$$L = \sum_{q=3}^N l_q (l_{q-1} + 1) (2l_{q-1} - 1) \sum_{k=1}^{q-2} |W_k|$$

Therefore:

$$L \geq 2 \sum_{q=3}^N l_{q-1} \sum_{k=1}^{q-2} |W_k|$$

We have also:

$$S = \sum_{q=4}^N 2l_q (l_{q-2} + 1) l_{q-1} \sum_{k=1}^{q-3} |W_k|,$$

and therefore:

$$S \geq 2 \sum_{q=3}^{N-1} l_{q-1} \sum_{k=1}^{q-2} |W_k| + \alpha,$$

where:

$$\alpha = 2 \sum_{q=3}^{N-1} l_{q+1} l_q \sum_{k=1}^{q-2} |W_k|$$

We have:

$$T \geq 2l_{N-1} \sum_{k=1}^{N-2} |W_k| + \beta,$$

where:

$$\beta = 2l_N \sum_{k=1}^{N-2} |W_k|$$

Therefore, we have:

$$S + T \geq 2 \sum_{q=3}^N l_{q-1} \sum_{k=1}^{q-2} |W_k| + \alpha + \beta,$$

and finally:

$$Y + L + S + T \geq \frac{11}{2} \sum_{q=3}^N l_{q-1} \sum_{k=1}^{q-2} |W_k| + \alpha + \beta$$

As  $\frac{11}{2}l_{q-1} > 3l_{q-1} + 2$  for all  $l_{q-1} \geq 1$ , we have:

$$Y + L + S + T \geq h + \alpha + \beta$$

7. As  $l_{q-1} \geq 1$  for all  $q$ , we have:

$$N \geq \sum_{q=2}^N l_q |W_{q-1}|,$$

and therefore:

$$N \geq i$$

8. we have:

$$I = \sum_{q=2}^N \frac{l_q}{2} \left( |W_{q-1}|^2 + 2|W_{q-1}| \sum_{k=1}^{q-2} |W_k| + \left( \sum_{k=1}^{q-2} |W_k| \right)^2 \right)$$

If we define  $\gamma$  as:

$$\gamma = \sum_{q=2}^N \frac{l_q}{2} \left( 2|W_{q-1}| \sum_{k=1}^{q-2} |W_k| + \left( \sum_{k=1}^{q-2} |W_k| \right)^2 \right),$$

as  $\frac{l_q}{2}|W_{q-1}|^2 = \frac{l_q}{2}(l_{q-2} + 1)l_{q-1}|W_{q-1}| \geq l_{q-1}|W_{q-1}|$ , we have:

$$I \geq \sum_{q=2}^N l_{q-1}|W_{q-1}| + \gamma$$

Therefore:

$$I + D \geq \sum_{q=1}^N l_q |W_q| + \gamma$$

We have also  $\frac{l_{q+1}}{2} \leq l_q$  and therefore:

$$I + D \geq d + \gamma$$

9. we have:

$$F = l_N(2l_N - 1) \sum_{k=1}^{N-1} l_k(l_{k-1} + 1)$$

Therefore, if we call  $\delta$ , the following value:

$$\delta = l_N(2l_N - 1) \sum_{k=1}^{N-1} l_k l_{k-1},$$

and therefore:

$$F = c + \delta$$

These inequalities imply:

$$\sum_{\Phi=A}^P \Phi \geq \sum_{\phi=a}^i \phi + R + \sum_{\Phi=U}^Z \Phi + \sum_{\varphi=\alpha}^{\delta} \varphi$$

This implies that the back-propagation algorithm is always faster than the direct method for computing the Hessian of the error made by an “every day” neural network.

### 9.3 First order differentials

As demonstrated in the previous subsection, the pure back-propagation algorithm is faster than the direct method for computing the Hessian of the error made by a “classical” neural network. But in order to use it, we need to compute first order differentials. This can be done with first order back-propagation or with a first order direct method, as explained in section 8. In this subsection, we study the complexity of both methods in the case of “every day” neural networks.

#### 9.3.1 Scalar output

If  $|O^l| = 1$  for each  $N^l \in \mathcal{N}$ , we have:

$$(9.3.i) D_{\text{first}}(\mathcal{E}) = \sum_{N^l \in \mathcal{N}} \sum_{N^k \in P^+(l)} (2|P(l) \cap S^+(k)| + \delta_{N^k \in P(l)} - 1) + \sum_{N^l \notin \text{Out}} (2|\text{Out} \cap S^+(l)| - 1)$$

$$(9.3.ii) B_{\text{first}}(\mathcal{E}) = \sum_{N^l \in \mathcal{N}} \sum_{N^k \in P^+(l)} (2|S(k) \cap P^+(l)| + \delta_{N^k \in P(l)} - 1) + \sum_{N^l \notin \text{Out}} (2|S(l)| - 1)$$

Once again, it is quite difficult to compare directly both formulae.

#### 9.3.2 Totally connected layered architecture

We use here assumptions, notations and results of subsection 9.2. In this case, we have:

$$(9.3.iii) \quad D_{\text{first}}(\mathcal{E}) - B_{\text{first}}(\mathcal{E}) = \sum_{p=2}^n \sum_{N^l \in L_p} \left( 2 \sum_{q=1}^{p-1} \sum_{N^k \in L_q} (|P(l) \cap S^+(k)| - |S(k) \cap S^+(l)|) + 2(|\text{Out} \cap S^+(l)| - |S(l)|) \right),$$

which gives:

$$(9.3.iv) \quad D_{\text{first}}(\mathcal{E}) - B_{\text{first}}(\mathcal{E}) = \sum_{p=3}^n l_p 2 \sum_{q=1}^{p-2} l_q (l_{p-1} - l_{q+1}) + 2 \sum_{p=1}^{n-2} l_p (l_n - l_{p+1})$$

This equation allows to conclude that for 2 layers, both methods are equivalent.

In classic neural networks, the number of neurons **decreases** with the rank of the layer: we use a lot of neurons in the first hidden layer, and this number goes down and down as we get closer to the output layer. In this case, we have  $l^{k+1} \leq l^k$ , for all  $k > 0$ . This result implies that  $l^{p-1} \leq l^{q+1}$  when  $q$  belongs to  $[1, p-2]$ . **Therefore, for a classic neural network with decreasing layers,  $D_{\text{first}}(\mathcal{E}) \leq B_{\text{first}}(\mathcal{E})$ , i.e., the direct algorithm is faster than the back-propagation method.**

When the network has a special architecture (i.e., with a “bottleneck” in which an internal layer has few neurons whereas its successor has a lot a neuron), we can have  $l^{k+1} > l^k$ . In this case, the back-propagation algorithm might be faster than the direct one.

In [3], Buntine and Weigend propose to use a hybrid algorithm. As the partial differentials  $\frac{\partial \mathcal{E}}{\partial o^l}$  are needed for all  $N^l \in \mathcal{N}$ , they propose to compute them with the back-propagation algorithm. Then, they propose to compute the  $\frac{\partial o^l}{\partial o^k}$  with a forward propagation algorithm which is in fact similar to our direct

method. In fact, the time complexity difference between the back-propagation and the direct method used to compute only the  $\frac{\partial o^l}{\partial o^k}$  is the given by the first sum in equation 9.3.iv. The value is:

$$(9.3.v) \quad 2 \sum_{p=3}^n l_p \sum_{q=1}^{p-2} l_q (l_{p-1} - l_{q+1})$$

This value is negative when the network has decreasing layers. Therefore, Buntine and Weigend's choice is the correct one. For special architecture, this is not the case. For a two layer network, the methods are equivalent.

Moreover, the additional cost of computing the  $\frac{\partial \mathcal{E}}{\partial o^l}$  with the direct method equal to:

$$(9.3.vi) \quad (2l_N - 1) \sum_{p=1}^{n-1} l_p,$$

whereas their computation with the back-propagation has a cost of:

$$(9.3.vii) \quad \sum_{p=1}^{n-1} l_p (2l_{p+1} - 1)$$

Once again, these values are equal for a two-layer network. For a network with decreasing layers, the direct method is obviously more efficient. Therefore, Buntine and Weigend's method is optimal neither for "every day" MLP, for which the back-propagation computation of  $\frac{\partial \mathcal{E}}{\partial o^l}$  is not efficient, nor for special network with a bottleneck for which direct computation of  $\frac{\partial o^l}{\partial o^k}$  might be inefficient.

In [13], Stephen Piché gives an algorithm to compute second differential in a recurrent network. His method needs to have the first differentials of the output of the network with respect to its weights,  $\frac{\partial G}{\partial w}$ , and to its inputs,  $\frac{\partial G}{\partial x}$ . It needs also the Hessian matrix of the error made by the network and therefore it needs to know all the  $\frac{\partial o^l}{\partial o^k}$ . In this case, computing  $\frac{\partial G}{\partial w}$  and  $\frac{\partial G}{\partial x}$  can be done easily with the local equations 4 and 6. Therefore, these differentials don't have to be computed by the back-propagation algorithm proposed by S. Piché. In fact, in network with decreasing layer, they will be computed thanks to a direct algorithm (which will compute the  $\frac{\partial o^l}{\partial o^k}$ ).

## 10 Conclusion and future work

In this report, we have introduced three different algorithms that can be used to compute second order differentials in arbitrary feed-forward neural networks. These algorithms, and especially the pure back-propagation algorithm (which performs better than the direct algorithm for MLP), can be used to extend to other models second order techniques originally designed for MLP. They can be used for instance to prune RBF or wavelet networks.

We have also studied the complexity of the proposed algorithms in some limited but important particular cases such as the MLP. We have shown that in many cases (for instance for MLP with decreasing layers) the first order back-propagation algorithm cannot compute the first order differentials needed for the Hessian computation as fast as the simple direct algorithm.

We need now to focus on a sharp analysis of the hybrid algorithm which might be faster than the two other algorithms in some particular cases.

## References

- [1] Edward K. Blum and Kwan Li Leong. Approximation theory and feedforward networks. *Neural Networks*, 4:511–515, 1991.

- [2] Léon Bottou and Patrick Gallinari. A Framework for the Cooperation of Learning Algorithms. In R.P. Lippmann, J.E. Moody, and D.S. Touretzky, editors, *Neural Information Processing Systems*, volume 3, pages 781–788. Morgan Kaufman, 1991.
- [3] Wray L. Buntine and Andreas S. Weigend. Computing Second Derivatives in Feed-Forward Networks : A Review. *IEEE Trans. on Neural Networks*, 5(3):480–488, May 1994.
- [4] Michel R. J. Fombellida, Michel J. M. Minsoul, and Jacques L. O. Destiné. Perceptrons multi-couches et fonctions d’activation non monotones. In *Proc. Neuro-Nîmes*, pages 321–339, Nîmes, 1990.
- [5] Ken-Ichi Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2:183–192, 1989.
- [6] Cédric Gégout, Bernard Girau, and Fabrice Rossi. A General Feed-Forward Neural Network Model. Technical report NC-TR-95-041, NeuroCOLT, Royal Holloway, University of London, May 1995. Alaivable at <http://apiacoa.org/publications/1995/neurocolt1995.pdf>.
- [7] Cédric Gégout, Bernard Girau, and Fabrice Rossi. Generic Back-Propagation in Arbitrary Feedforward Neural Networks. In D. W. Pearson, N. C. Steele, and R. F. Albrecht, editors, *Int. Conf. on Artificial Neural Nets and Genetic Algorithms*, pages 168–171, Alès, April 1995. Springer Verlag.
- [8] Babak Hassibi, David G. Stork, and Gregory J. Wolff. Optimal brain surgeon and general network pruning. In *Int. Conf. on Neural Networks*, pages 293–299. IEEE, 1993.
- [9] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.
- [10] Věra Kurková. Universal approximation using feedforward neural networks with gaussian bar units. In Bernd Neumann, editor, *Proc. Euro. Conf. on Artificial Intelligence*, pages 193–197, Vienna, Aug. 1992. ECCAI, John Wiley & Sons.
- [11] J.A. Leonard, M. A. Kramer, and L. H. Unger. Using radial basis functions to approximate a function and its error bounds. *IEEE Trans. Neural Networks*, 3(4):624–627, Jul. 1992.
- [12] David B. Parker. Optimal algoritihms for adaptive networks: second order back propagation, second order direct propagation, and second order hebbian learning. In *Proc. First Int. Joint Conf. Neural Networks*, volume II, pages 593–600, November 1987.
- [13] Stephen W. Piché. The Second Derivative of a Recurrent Network. In *Proc. Int. Conf. on Neural Networks*, volume 1, pages 245–250, Orlando (Florida), June 1994. IEEE.
- [14] Tomaso Poggio and Federico Girosi. Networks for approximation and learning. *Proc. IEEE*, 78(9):1481–1497, September 1990.
- [15] Dirk Röckmann and Claudio Moraga. Using quadratic perceptrons to reduce interconnection density in multilayer neural networks. In A. Prieto, editor, *Artificial Neural Networks*, chapter 1, pages 86–92. Springer-Verlag, 1991.
- [16] Fabrice Rossi and Cédric Gégout. Geometrical Initialization, Parametrization and Control of Multilayer Perceptrons : Application to Function Approximation. In *Int. Conf. on Neural Networks*, volume I, pages 546–550, Orlando (Florida), June 1994. IEEE.
- [17] Raymond L. Watrous. Learning algorithms for connectionist networks: applied gradient methods of nonlinear optimization. In *Proc. IEEE First Int. Conf. Neural Networks*, volume II, pages 619–626, June 1987.
- [18] Qinghua Zhang and Albert Benveniste. Wavelet networks. *IEEE Trans. On Neural Networks*, 3(6):889–898, November 1992.