A Theoretical Framework for Back-Propagation

Article · August 2001

Source: CiteSeer

CITATIONS
24

READS
11,219

1 author:

Yann Lecun
New York University
555 PUBLICATIONS **159,210** CITATIONS

SEE PROFILE

Y. Le Cun. A theoretical framework for back-propagation. In D. Touretzky, G. Hinton, and T. Sejnowski, editors, *Proceedings of the 1988 Connectionist Models Summer School*, pages 21–28, CMU, Pittsburgh, Pa, 1988. Morgan Kaufmann.

A Theoretical Framework for Back-Propagation *

Yann le Cun †
Department of Computer Science, University of Toronto
Toronto, Ontario, M5S 1A4. CANADA.

Abstract

Among all the supervised learning algorithms, back propagation (BP) is probably the most wi(l)dely used. Although numerous experimental works have demonstrated its capabilities, a deeper theoretical understanding of the algorithm is definitely needed. We present a mathematical framework for studying back-propagation based on the Lagrangian formalism. In this framework, inspired by optimal control theory, back-propagation is formulated as an optimization problem with nonlinear constraints. The Lagrange function is the sum of an output objective function and a constraint term which describes the network dynamics.

This approach suggests many natural extensions to the basic algorithm.

It also provides an extremely simple formulation (and derivation) of continuous recurrent network equations as described by Pineda [Pineda, 1987].

Other easily described variations involve either additional terms in the error function, additional constraints on the set of solutions, or transformations of the parameter space. An interesting kind of constraint is an equality constraint among the weights, which can be implemented with little overhead. It is shown that this sort of constraint provides a way of putting *a priori* knowledge into the network while reducing the number of free parameters.

1 Introduction

The Back Propagation algorithm has recently emerged as one of the most efficient learning procedures for multi-layer networks of neuron-like units. One of the reasons of the success of back-propagation is its incredible simplicity. In fact, back-propagation is little more than an extremely judicious application of the chain rule and gradient descent.

There are a number of ways to derive back-propagation . The simplest derivation is the one given in [Rumelhart *et al.*, 1986]. An alternate derivation has been proposed in [le Cun, 1986] that uses local criteria attached to each unit which are minimized locally. Various constraints can be put on these local criteria giving several variations of the original algorithm [le Cun, 1985; le Cun, 1986; le Cun, 1987]. Various other derivations of Back Propagation have also been reported earlier in different contexts [Parker, 1985] [Werbos, 1974]

The present paper proposes a derivation of back-propagation based on the Lagrangian formalism. An early version of this framework has been presented in [Fogelman-Soulié *et al.*, 1986] (also published in [Fogelman-Soulié *et al.*, 1987]) and a more extended version in [le Cun, 1987].

This formalism is directly inspired by optimal control theory. There is an abundant literature on optimal control that uses the method of Lagrange multipliers to find the optimal values of a set of control variables. The continuous version of this method is called variational calculus, and its purpose is to find a *function* (usually not a a set of discrete values) that minimizes an objective function subject to constraints. Variational calculus and its

[0] to appear in "Proceedings of the 1988 connectionist models summer school, Carnegie-Mellon University" D Touretzky, G. Hinton, T. Sejnowski (eds), Morgan Kaufmann 1989.

[0] Authors present address: Room 4G-332, AT&T Bell Laboratories, Crawfords Corner Rd, Holmdel, NJ 07733.

extensions are, in fact, the basis of most work in optimal control [Noton, 1965; Athans and Falb, 1966; Bryson and Ho, 1969]. As we will see, the classical algorithms given by this formalism closely resemble back-propagation .

The central problem that back-propagation solves is the evaluation of the influence of a parameter on a function whose computation involves several elementary *steps*. The solution to this problem is given by the chain rule, but back-propagation exploits the particular form of the functions used at each step (or layer) to provide an elegant and local procedure.

We show below that this problem can be easily stated using variational principles.

One of the questions addressed by optimal control theory is how the state of a system at time T will be modified by a change in the control variables at time $T-t$. In discrete time, this problem is very similar to the problem of back-propagation if we replace the time index by a layer index, namely, how the output of a network (state of the K^{th} layer) will be modified by changing a weight in layer $K-k$. The *trajectory* in the classical paradigm is analogous to the states of the successive layers in the network.

In the continuous time case, there is an even more straightforward analogy between the standard optimal control formalism and continuous time recurrent networks like the one proposed by Almeida [Almeida, 1987] and by Pineda [Pineda, 1987].

The purpose of computing partial derivatives of the states with respect to the parameters in the system is to minimize an objective function which measures how far the behaviour of the network is from a desired behaviour.

Following the variational formalism of Lagrange, Pontryagin has shown in the late 1950's how to formulate this problem using a single energy-like Hamiltonian function. The behaviour of the system is completely described by a single equation stating that the Hamiltonian meets some optimality criterion. An extensive treatment of Pontryagin's minimum principle can be found in [Athans and Falb, 1966]. The Pontryagin principle is an extension of classical variational calculus to problems involving non-differentiable functions, especially when inequality constraints must be met by the state variables or by the control variables. It is closely related to Bellman's dynamic programming, and, in fact, it can be derived as a limit case of it (see [Noton, 1965] for example). For the problem of simple feed-forward multi-layer networks, the full generality of Pontryagin's result, even of variational calculus, is not needed. Only the standard Lagrange multiplier method will be used.

Some of the applications and algorithms described in the optimal control literature so closely resemble back-propagation that one could credit Pontryagin (among others) for its discovery. Although the relevance of this work to automatic machine learning is not clearly mentioned in the early literature, the results have been extensively applied to some closely related problems such as system identification. Most of the optimal control literature deals with continuous time dynamical systems, however the proliferation of digital computers has lead to discrete time versions of the methods. The use of back-propagated variables for computing derivatives is apparent in the classical literature, and, as we will see, is a direct consequence of the formalism. In optimal control, the back-propagated vector is called the co-state or adjoint state, and the corresponding backward system the adjoint system [Athans and Falb, 1966].

Since his first work on the subject, the author has found that A. Bryson and Y.-C. Ho [Bryson and Ho, 1969] have described the back-propagation algorithm using Lagrange formalism. Although their description was of course in the framework of optimal control, not machine learning, the resulting procedure is identical to back-propagation. They formulate it as the the optimal control solution of a "multistage system" defined as a cascade of elementary systems f^0, f^1, f^N (analogous to the layers) controlled by a set of control variables $u(0), u(1), u(N-1)$ (analogous to the weights) and minimizing a performance index which depends on the final output [1]. No strong assumption is made about the dependency relation between the control variables and the states. The derivation includes the expression of the back-propagated gradients and the authors suggest the use of a gradient descent technique to find the control that optimizes the performance index. The section is judiciously concluded by: "Such problems are called *two-point boundary-value problems*, and they are sometimes rather difficult to solve, even with a high speed computer." [Bryson and Ho, 1969]. To the author's knowledge, this is the first description of back-propagation as

[1] their performance index also includes terms which depend on the state and the control at every stage

we know it, although the idea of back-propagating derivatives is much older, especially for continuous time systems [Athans and Falb, 1966; Noton, 1965].

Although the theoretical foundation of back-propagation and its first use in optimal control is old, the (independant) discovery of its relevance to connectionist systems and its interpretation in this context are recent.

2 Deriving BP using the Hamiltonian/Lagrangian formalism

2.1 Notations

For the sake of clarity, we will introduce the formalism in a simple case. A more general formulation will be presented afterwards. It will be assumed that the network is composed of a number of layers connected in a feed-forward manner. Furthermore, we make the assumption that connections cannot skip layers. These assumptions can be easily relaxed [le Cun, 1987].

We use the following notation. The layers are indexed from 0 to N. Layer 0 is the input layer and layer N is the output layer. The state of layer k for pattern p is denoted $X_p(k)$, and the global state of the network for pattern p is denoted by X_p. X_p is simply composed of all the $X_p(k)$'s concatenated together. We denote by X the vector composed of all the X_p concatenated together for $p = 1 \ldots P$. Layer $k-1$ is connected to layer k through a connection matrix $W(k)$. The vector of total input to units in layer k (weighted sums) is denoted $A_p(k)$, its value is given by:

$$A_p(k) = W(k)X_p(k-1)$$

The equation of forward propagation is simply:

$$X_p(k) = F(W(k)X_p(k-1)) =$$

$$F_k(A_p(k)) \quad \forall k \in [1, N]$$

$X_p(0)$ is defined to be the external input vector I_p. F_k denotes the non-linear transformation associated with layer k. The components of F_k will typically be sigmoid functions.

The vector of desired outputs is noted D_p and has the same dimension as the ouput layer $X_p(N)$

2.2 BP as a constrained minimization problem

In this section, we show that back-propagation can be view as a constrained minimization problem involving not only the weights W, but also the state of the network X as the problem variables. The Lagrange function [2] corresponding to the problem is the sum an objective function (usually the squared output error) and a constraint term multiplied by Lagrange multipliers denoted $B_p(k)$. The constraint term describes the *structure* of the network, i.e. the dependency relations among the $X(i)$. The Lagrange function (LF) for a single pattern p (called local Lagrange function) has the following form

$$L_p(W, X_p, B_p) = C(X_p(N)) +$$

$$\sum_{k=1}^{N} B_p(k)^T (X_p(k) - F[W(k)X_p(k-1)])$$

and the full LF for the current pattern ensemble is:

$$L(W, X, B) = \sum_{p=1}^{P} L_p(W, X_p, B_p)$$

where P is the number of training examples. If the objective function C is defined to be the squared output error, the local LF becomes:

$$L_p(W, X, B) = (D_p - X_p(N))^T (D_p - X_p(N)) +$$

$$\sum_{k=1}^{N} B_p(k)^T (X_p(k) - F[W(k)X_p(k-1)]) \quad (1)$$

As stated above, L is the sum of two terms. The first term is just the squared output error, i.e the square norm of the difference between the desired output D_p and the actual output $X_p(N)$. In the general case, the objective function need not have this form, and it can incorporate other terms depending on any state variables, weights, etc ..., anywhere in the network. The second term is the sum of N terms, one for each layer, which are interpreted as constraints. If the constraints are met, all these terms are zero. Each term is the dot product of a *Lagrange multiplier vector* $B(k)$ and a constraint term. When the constraint is fulfilled, the constraint term is zero, yielding:

$$X(k) = F[W(k)X(k-1)] \quad \forall k \in [1, N]$$

[2] the Lagrange function described here has no relation with the Lagrangian as defined by physicists

We recognize the forward propagation equation given in the previous section. Thus, the constraint term just defines the network forward dynamics, i.e. the dependencies between X and W, and among the $X(k)$'s. As it will be shown later, the Lagrange multipliers B take account of the backward dynamics. Again, the forward dynamics equation does not need to have this particular form, it could be any differentiable function of X, W, and eventually k. The problem is to find a sequence $W(1), W(2) \ldots W(N)$ which minimizes the objective function C while satisfying the constraints. It is easy to show [Bryson and Ho, 1969] that

$$\nabla L(W, X, B) = 0$$

is a necessary condition which defines a local minimum of the performance function C with respect to the weights while meeting the constraints [3]. This single equation totally describes the behaviour of the network. The condition can be split into three subconditions:

$$\frac{\partial L(W, X, B)}{\partial B} = 0 \tag{2}$$

$$\frac{\partial L(W, X, B)}{\partial X} = 0 \tag{3}$$

$$\frac{\partial L(W, X, B)}{\partial W} = 0 \tag{4}$$

Each of these conditions can be further decomposed into N conditions corresponding to each layer.

Each subcondition, when developed, yields one of the three passes of back-propagation . The first subcondition gives the forward propagation pass, the second one gives the backward propagation pass (the gradients). The third subcondition does not give a direct way to compute W but it does give the optimality condition that must be fulfilled.

2.2.1 The first subcondition

Equation 2 can be decomposed into N times P elementary conditions

$$\frac{\partial L(W, X, B)}{\partial B_p(k)} = 0 \quad \forall k, p \in [1, N][1, P]$$

It is obvious that

$$\frac{\partial L(W, X, B)}{\partial B_p(k)} = \frac{\partial L_p(W, X_p, B_p)}{\partial B_p(k)} \quad \forall k, p \in [1, N][1, P]$$

since for any local LF $L_q, q \neq p$ does not depend on B_p.

[3] the sufficient condition also states that the Hessian of L with respect to W must be positive semi-definite

The condition simply yields

$$X_p(k) = F_k[W(k)X_p(k-1)] \quad \forall k, p \in [1, N][1, P]$$

which is the forward dynamics equation.

2.2.2 The second condition

Equation 3 will be also decomposed into N times P subconditions, but the condition corresponding to the last layer ($k = N$) will be of a different nature.

For $k = N$ we have

$$\frac{\partial L_p(W, X_p, B_p)}{\partial X_p(N)} = 0 \quad \forall p \in [1, P]$$

and for $k \in [1, N-1]$ we have

$$\frac{\partial L_p(W, X_p, B_p)}{\partial X_p(k)} = 0 \quad \forall k, p \in [1, N-1][1, P]$$

Note that there is no condition on $X(0)$ since $X(0)$ is the input vector. It is not considered as a variable of the system but rather as a boundary condition.

The first subcondition (for $k = N$) immediately gives a boundary condition on B, i.e. the value of $B(N)$

$$B_p(N) = 2(D_p - X_p(N)) \tag{5}$$

When k is different from N, the subcondition gives

$$B_p(k) = W^T(k+1)\nabla F_k(A_p(k+1))B_p(k+1) \tag{6}$$

In this equation, the term

$$\nabla F_k(A_k(k+1))$$

is the Jacobian matrix of F_k at point $A_p(k+1)$. For a standard back-propagation network, this matrix will be diagonal and its i^{th} diagonal term will be given by

$$\nabla F_k(A_p(k+1))_{ii} = f'(a_{ip}(k+1))$$

where f is the sigmoid function and $a_{ip}(k+1)$ the i^{th} component of $A_p(k+1)$.

We can simplify the notation by transforming equations 5 and 6 through the following change of variable

$$Y_p(k) = \nabla F_k(A_p(k))B_p(k)$$

The two equations are then transformed into

$$Y_p(N) = 2\nabla F_N(A_p(N))(D_p - X_p(N)) \tag{7}$$

$$Y_p(k) = \nabla F_k(A_p(k))W^T(k+1)Y_p(k+1) \quad \forall k \in [0, n-1] \tag{8}$$

As is now apparent, these equations describe the backward dynamics. Equation 7 is a boundary condition on the gradient variables Y while equation 8

gives the usual method for computing Y's by backward propagation. The Lagrange multipliers may be recognized as the back-propagated gradients. In optimal control theory, $B(k)$ is called the *influence function* and also called *costate* of the system, or the *adjoint state*. The corresponding backward system is called the *adjoint system* [Athans and Falb, 1966].

2.2.3 The third condition

The third condition, unfortunately, does not give a direct method for computing W, but it gives a condition that W should satisfy. As usual, equation 4 gives N subconditions

$$\frac{\partial L(W,X,B)}{\partial W(k)} = 0 \quad \forall k \in [1,N]$$

yielding

$$\sum_{p=1}^{P} \nabla F_k(A_p(k)) B_p(k) X_p^T(k-1) = 0 \quad \forall k \in [1,N]$$

we can apply the same change of variable as previously, substituting Y for B

$$\sum_{p=1}^{P} Y_p(k) X_p^T(k-1) = 0 \quad \forall k \in [1,N]$$

this condition states that W is a stationary point of L, i.e a local minimum, maximum, or a saddle point. Our problem is, of course, to find a *minimum* of the output cost function with respect to W, and this can be shown to be equivalent to finding a minimum of L while satisfying the first two subconditions. The easiest and most common method to do so is the method of steepest descent. As stated before, the problem to be solved is a two point boundary value problem, for which no magical solution exists. A steepest descent procedure has the following form

$$W(k) \leftarrow W(k) - \lambda \frac{\partial L(W,X,B)}{\partial W(k)}$$

where λ is the step size. After algebraic refinement, this equation gives

$$W(k) \leftarrow W(k) + \lambda \sum_{p=1}^{P} Y_p(k) X_p^T(k-1) \qquad (9)$$

We recognize the usual weight update formula of the classical back-propagation algorithm.

2.2.4 Summary of the results

The combined results are summarized by these three equations which are respectively the forward pass, the backward pass and the weight updates

$$X_p(k) = F_k[W(k)X_p(k-1)] \qquad \forall k, p \in [1,N][1,P]$$

$$Y_p(k) = \nabla F_k(A_p(k)) W^T(k+1) Y_p(k+1)$$
$$\forall k, p \in [0,n-1][1,P]$$

$$W(k) \leftarrow W(k) + \lambda \sum_{p=1}^{P} Y_p(k) X_p^T(k-1) \qquad \forall k \in [1,N]$$

with the two boundary conditions

$$X_p(0) = I_p$$

$$Y_p(N) = 2\, \nabla F_N(A_p(N))\, (D_p - X_p(N))$$

This is exactly the back-propagation algorithm. There are several related ways to derive this result, for example, Bryson and Ho [Bryson and Ho, 1969] define a Hamiltonian function

$$H_p(k) = B_p(k)^T F[W(k)X_p(k-1)]$$

and point out the relations between this quantity and the classical Hamiltonian of a physical system.

It should be emphasized that the formalism can be easily modified to incorporate connections that can skip layers, recurrent networks, weight decay, and other extensions [le Cun, 1987].

3 A few extensions

Several generalizations and extensions of this formalism can be devised, which concern iterative networks, networks with equality constraints between weights...[le Cun, 1987]. Still others can be inspired by the optimal control literature.

Some particularly interesting variations will be described here.

3.1 Transforming the parameter space

Back-propagation is often considered as a search for a minimum of the cost function in *weight space*, however, in some cases it is interesting to consider the weights not as elementary variables, but as *functions* which depend on a set of elementary variables U. A good example is when the designer wants to put *a priori* knowledge about the task into the network. For example, equality constraints between weights can be enforced to make the network response invariant under certain transformations of

the input vector. Another example is when the weight space is ill-conditioned or too complicated to search. Then appropriate transformations can be applied to improve its geometrical properties and accelerate learning.

The weight matrix W is now considered to be a function of a set of variables collectively designated as an r dimensional vector U. The local Lagrange function becomes:

$$L_p(U, X_p, B_p) = C(X_p(N)) + \sum_{k=1}^{N} B_p(k)^T (X_p(k) - F[W(U,k)X_p(k-1)])$$

The network obeys the same optimality condition as in the previous section:

$$\nabla L(U, X, B) = 0$$

A treatment similar to the previous section can be developed, except for the third subcondition which is now:

$$\frac{\partial L(U,X,B)}{\partial U} = 0 \qquad (10)$$

After refinement, this equation yields:

$$u_q \leftarrow u_q + \lambda \sum_{ijk} \frac{\partial L}{\partial w_{ij}(k)} \frac{\partial w_{ij}(k)}{\partial u_q}$$

or

$$u_q \leftarrow u_q + \lambda \sum_{ijk} y_i(k) x_j(k-1) \frac{\partial w_{ij}(k)}{\partial u_q}$$

The summation over three indices i,j and k can be time-consuming if an u_q influences many $w_{ij}(k)$, but in practice the interactions between u's and w's will be local.

An interesting special case is when several weights share a single parameter u; this provides a way of implementing equality constraints between the weights with very little overhead. The derivative with respect to a particular u is simply the sum of the derivatives with respect to the w's that share it. This particular kind of equality constraint can be used to describe "time unfolded" iterative networks described in [Rumelhart *et al.*, 1986].

3.2 Continuous time recurrent networks

Almeida and Pineda [Almeida, 1987] [Pineda, 1987] recently described a model of a recurrent back-propagation network governed by continuous time differential equations of the form:

$$\tau_x \frac{dX}{dt}(t) = F(WX(t)) - X(t)$$

Where τ_x is a time constant. It should be emphasized that the following calculations can be performed with a more general equations where F is not a function of the product WX, but a function of W *and* X separately. Also, τ_x could be a positive definite diagonal matrix instead of a scalar.

Pineda and Almeida have shown that when the system has reached a fixed point, it is possible to compute the derivatives of this fixed point with respect to the weights. This vector of derivatives is identified as the fixed point of another differential equation.

We will show that his result can be obtained extremely easily using variational principles.

A fixed point of the system is characterized by

$$\frac{dX}{dt} = 0$$

which is equivalent to

$$X - F(WX) = 0$$

We define the Hamiltonian H of the system as the sum of the objective function C which depends on the state X, and of a constraint term characterizing a fixed point:

$$H = C(X) + B^T[X - F(WX)]$$

where B is a vector of undetermined Lagrange multipliers. The question is now: what is the derivative of C with respect to W *while staying at a fixed point*? In other words, what is the variation of H caused by a variation of W while maintaining constraint satisfaction? The variation of H corresponding to a variation of W is given by:

$$\delta H = \frac{\partial H}{\partial X}\delta X + \frac{\partial H}{\partial W}\delta W$$

where δX is the variation of X caused by the variation of W. This equation becomes [4]

$$\delta H = (\frac{\partial C}{\partial X} + B^T \frac{\partial (X - F(WX))}{\partial X})\delta X + (\frac{\partial C}{\partial W} + B^T \frac{\partial (X - F(WX))}{\partial W})\delta W$$

now, it is difficult to compute the variation of X, denoted byδX, caused by the variation of W, denoted by δW, and in fact, we would like that the result does not depend on δX. So we *choose* B such

[4] note that the derivative of a scalar function w.r.t a vector is a *line* vector

that the coefficient of δX vanishes. The Lagrange multiplier B then should satisfy

$$\frac{\partial C}{\partial X} + B^T \frac{\partial (X - F(A))}{\partial X} = 0$$

using the notation $A = WX$. We then obtain

$$\frac{\partial C}{\partial X} + B^T (I - \frac{\partial F}{\partial A} W) = 0$$

or

$$\frac{\partial C}{\partial X}^T + B - W^T \frac{\partial F}{\partial A}^T B = 0$$

this equation must be solved for B. Since it is a linear system, a large variety of methods can be used. As suggested by Pineda, we can interpret the equation as characterizing the fixed point of a dynamical system described by a differential equation of the form

$$\tau_b \frac{dB}{dt} = -B + W^T (\frac{\partial F}{\partial A})^T B - \frac{\partial C}{\partial X}^T$$

As in the case of feedforward network, it is convenient to apply the change of variable:

$$Y = (\frac{\partial F}{\partial A})^T B$$

By multiplying both sides of the previous equation by $\frac{\partial F}{\partial A}$ and substituting, we obtain:

$$\tau_y \frac{dY}{dt} = -Y + \frac{\partial F}{\partial A}^T (W^T Y - \frac{\partial C}{\partial X}^T)$$

This is the Pineda-Almeida equation for computing the gradients of the fixed point with respect to the weights. As in the feed-forward case, the Lagrange multipliers can be interpreted as the negative gradients [5]. Now that B has been computed so that the coefficient of δX vanishes, the expression of δH is greatly simplified:

$$\delta H = \frac{\partial H}{\partial W} \delta W$$

or

$$\delta H = (\frac{\partial C}{\partial W} + B^T \frac{\partial (X - F(WX))}{\partial W}) \delta W$$

Hence, if B (and Y) are computed as prescribed above, the *total* derivative of H with respect to W is simply:

$$\frac{dH}{dW} = \frac{\partial C}{\partial W} + B^T \frac{\partial (X - F(WX))}{\partial W}$$

[5] note that the equations of motion that we use for X and B are equivalent to the classical Hamiltonian equations of motion: $\frac{\partial H}{\partial B} = -\frac{dX}{dt}$ and $\frac{\partial H}{\partial X} = \frac{dB}{dt}$. Here B Plays to role of the momentum in classical mechanics.

Since the system is assumed to be at a fixed point X, the constraint is satisfied and the constraint term in the Hamiltonian vanishes. Therefore, the total derivatives of H and C with respect to W are equal. We obtain the main result:

$$\frac{dC}{dW} = \frac{\partial C}{\partial W} + B^T \frac{\partial (X - F(WX))}{\partial W}$$

If C is not a direct function of W, as is usually the case [6], this expression reduces to

$$\frac{dC}{dW} = B^T \frac{\partial (X - F(WX))}{\partial W}$$

The expression of $\frac{dC}{dw_{ij}}$ is finally obtained

$$\frac{dC}{dw_{ij}} = -y_i x_j$$

We are now able to apply a gradient descent procedure to minimize C

$$w_{ij} \leftarrow w_{ij} + \lambda y_i x_j$$

or a continuous version of it

$$\tau_w \frac{dw_{ij}}{dt} = y_i x_j$$

4 Conclusion

The theoretical formalism described in this paper seems to be well suited to the description of many different variations of back-propagation . The paper only explores a few of them. Considering the amount of available literature in optimal control theory, it seems that we have only scratched the surface of the range of possible applications. The method described here not only provides a clean way of deriving back-propagation like procedures, but also greatly simplifies the derivations.

From a historical point of view, back-propagation had been used in the field of optimal control long before its application to connectionist systems has been (independently) proposed. Nevertheless, the interpretation of back-propagation in the context of connectionist systems, as well as most related concepts are recent, and the historical and scientific importance of [Rumelhart *et al.*, 1986] should not be overlooked. The concepts are new, if not the algorithm.

[6] for example, C could incorporate a "weight decay" term, in this case $\frac{\partial C}{\partial W}$ would not be zero

Acknowledgments

This work has been supported by a grant from the Fyssen foundation and a grant from the Sloan foundation. The author wishes to thank Geoff Hinton, Sue Becker, Mike Mozer, Steve Nowlan and Didier Georges for helpful discussions. Special thanks to Léon-Yves Bottou, the neural network simulator SN is the result of our collaboration.

References

[Almeida, 1987] Luis B. Almeida. A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. In *Proceedings ICNN 87*, San Diego, 1987. IEEE, IEEE.

[Athans and Falb, 1966] M. Athans and P. L. Falb. *Optimal Control Theory*. McGraw Hill, 1966.

[Bryson and Ho, 1969] A. E. Jr. Bryson and Yu-Chi Ho. *Applied Optimal Control*. Blaisdell Publishing Co., 1969.

[Fogelman-Soulié *et al.*, 1986] F. Fogelman-Soulié, P. Gallinari, Y. le Cun, and S. Thiria. Automata networks and artificial intelligence. Technical report, Laboratoire de Dynamique des Reseaux, 1986.

[Fogelman-Soulié *et al.*, 1987] F. Fogelman-Soulié, P. Gallinari, Y. le Cun, and S. Thiria. Automata networks and artificial intelligence. In *Automata networks in computer science, theory and applications*, pages 133–186. Princeton University Press, 1987.

[le Cun, 1985] Y. le Cun. A learning scheme for asymmetric threshold networks. In *Proceedings of Cognitiva 85*, pages 599–604, Paris, France, 1985.

[le Cun, 1986] Y. le Cun. Learning processes in an asymmetric threshold network. In F. Fogelman-Soulié E. Bienenstock and G. Weisbuch, editors, *Disordered systems and biological organization*, pages 233–240, Les Houches, France, 1986. Springer-Verlag.

[le Cun, 1987] Y. le Cun. *Modèles Connexionnistes de l'Apprentissage*. PhD thesis, Université Pierre et Marie Curie, Paris, France, 1987.

[Noton, 1965] A. R. M. Noton. *Introduction to Variational Methods in Control Engineering*. Pergamon Press, 1965.

[Parker, 1985] D. B. Parker. Learning-logic. Technical report, TR-47, Sloan School of Management, MIT, Cambridge, Mass., April 1985.

[Pineda, 1987] F.J. Pineda. Generalization of back propagation to recurrent and higher order neural networks. In *Proceedings of IEEE Conference on Neural Information Processing Systems*, Denver, Colorado, November 1987. IEEE.

[Rumelhart *et al.*, 1986] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In *Parallel distributed processing: Explorations in the microstructure of cognition*, volume I. Bradford Books, Cambridge, MA, 1986.

[Werbos, 1974] P. Werbos. *Beyond Regression*. Phd thesis, Harvard University, 1974.