A UNIFIED FORMALISM FOR NEURAL NET TRAINING ALGORITHMS

Léon Bottou*, Patrick Gallinari**

* AT&T Bell Labs
Crawford Corner Road
Holmdel NJ 07 733 - USA

** L.R.I. - Bat 490
Université de Paris Sud - U.A. 410 CNRS
91405 Orsay Cedex- FRANCE
e-mail: patrick@lri.lri.fr

Abstract

We present a framework which provides both a unified formalism for describing many connectionist algorithms and a formal definition of the goal of learning for these algorithms. This formal approach is illustrated through several examples taken among the "classical" connectionist literature. Many non connectionist systems also fall into this formulation which is thus very general and has several consequences upon the design of connectionist systems. For example it allows to train optimally hybrid architectures where different connectionist or classical modules interact together.

Keywords: Learning, adaptive and stochastic optimization, modular architectures.

1. INTRODUCTION

Connectionist algorithms are either expressed as techniques for solving optimization problems or most often as heuristic rules for weight updating. Because these algorithms are adaptive, their mathematical analysis is much more difficult than that of classical optimization techniques. Although progresses have been made recently for their theoretical analysis there is a lack for a formal setting which would allow to study them from a unified point of view. Also, important progresses have been made recently by building systems where different algorithms, connectionist or not, cooperate together [Bourlard & Wellekens 88, Bollivier et al. 90, Driancourt et al. 91]. Of course it is hopeless to train these complex systems if the different modules are described using different formalisms. It is thus very important to be able to give a mathematical formulation of the goal of learning for these algorithms, together with a unified formalism for describing the different modules of such systems. This is our goal in this paper.
We first introduce a formal framework in 2 and show in 3 and 4 by taking various examples from the connectionist literature that it is suitable for many algorithms either supervised or unsupervised. Some algorithms directly fall into this formulation whereas some others have to be modified to fit the formalism. This allows us to propose new algorithms.

2. LEARNING AS AN OPTIMISATION PROBLEM FOR CONNECTIONIST SYSTEMS

Learning in connectionist models generally amounts in changing the parameters of the model using an adaptive update rule whose general formulation is :

$$w(n) = w(n-1) - \varepsilon(n) F(x(n), w(n-1)) \qquad (1)$$

where w(n) denotes the parameters of the system at time n, x(n) is the current pattern. For unsupervised learning, x would be the input pattern whereas for supervised learning x = (x,y) would be a pair (input, desired output), ε is the modification step. F is either the gradient of a cost function or a heuristic rule.
We will use in the following a probabilistic formulation of learning which has been proposed in [Tsypkin 71] and has been used in several fields such as control, pattern recognition, adaptive filtering.
Let x be an instance of a concept to learn, which is defined by a probability density function p(x), and w the parameters of the learning system. For a given state of the system, we can define a **local cost function** J(x, w) which measures how well our system behaves on x. The goal of learning is often the optimization of some functional of the parameters and of the concept to learn which we will call the **global cost function**. It is usually the expectation of the local cost function over the space X of the concept to learn:

$$C(w) = \int_X J(x,w) p(x) dx = E_X\{J(x,w)\} \qquad (2)$$

Most often we do not know the explicit form of p(x) and therefore C(w) is unknown. Our knowledge of the random process comes from a series of observations $\{x_i\}_i$ of the variable x. We are thus only able to measure the realizations J(x,w) for the observations $\{x_i\}_i$.
A necessary condition of optimality for the parameters of the system is:

$$\nabla C(w) = E_X\{\nabla_w J(x,w)\} = 0 \qquad (3)$$

0-7803-0559-0/92 $3.00 © 1992 IEEE

where ∇ is the gradient operator. Because we do not know $\nabla C(w)$ but only the realizations $\nabla_w J(x,w)$, we cannot use classical optimization techniques to reach the optimum. One solution is to resort to adaptive algorithms, the simplest being the following:

$$w(n) = w(n-1) - \gamma(n)\nabla_w J(x(n),w(n-1)) \qquad (4)$$

where $\gamma \in \Re$ is the gradient step. There is an obvious similarity between (1) and (4) and when learning in connectionnist system aims at minimizing a cost function, there is a complete identity. The goal of this paper is to improve upon this similarity. We will proceed in two steps.

We first show by taking various examples from the connectionist literature that many algorithms either supervised or unsupervised can be expressed as *adaptive minimization rules of local cost functions.* Some algorithms directly fall into this formulation whereas some others have to be slightly modified to fit the formalism. The examples we present are representative of the main families of NN. Most of the connectionist algorithms can thus be described under this unified formalism.

However, it would be more interesting to proceed further and to show that connectionist learning rules allow to reach optimum values of *global criteria* which would represent the explicit **goal of learning** for the system. We study this problem in a second step. Some connectionist algorithms are already expressed as gradient algorithms for global cost functions. However many of them which use heuristic update rules cannot be expressed under this formalism. We introduce variations of these algorithms which allow such a formulation. They obey the same goal as the original algorithms with the advantage of being optimal for an explicit functional criterion. New algorithms are thus proposed.

We assume that the reader is familiar enough with the main connectionist algorithms and refer to the references for a discussion of these techniques.

3. OPTIMIZING LOCAL CRITERIA

It is usually easy to infer a local gradient algorithm from the update rules of many connectionist algorithms, we give below some examples which are representative of many important families of NN techniques. It is thus possible to derive similar cost functions and learning rules for most algorithms which have been proposed in the literature. In some cases (e.g MLP familly), algorithms are already known to be minimization rules for a given cost function. In many other cases, the algorithms have not been proposed as optimization techniques, but it is quite easy to infer a local cost function from the learning rule. The algorithms are discussed below, cost functions and learning rules are given in table 1.

• **Adaline, Multilayer Perceptrons (MLP), Radial Basis Functions (RBF)** : these neural nets (NN) [Widrow & Hoff 60, Rumelhart et al. 86, Robinson et al. 88] have been the most widely used algorithms for now, and can be used for a large range of tasks (classification, regression, prediction, filtering...). We will denote by $f(x,w)$ the multi-valued function computed by any of these networks and y the desired output corresponding to input x, i.e. $x = (x,y)$. The local error for these nets is a similarity function between the desired and computed outputs. Most often it is the euclidean distance. Other similarity functions may be used, e.g. Minkowsky distances or different entropy criteria for classification tasks. The learning rule for all these different criteria and NN is simply rule (4).

• **Perceptron, Ho & Kashyap algorithm** : we will consider the two class case for simplicity. Although initially proposed as a heuristic rule, it is well known [Duda & Hart 73] that the perceptron rule may be shown to optimize a local cost function (table 1). In this function, sgn is the sign function which is either -1 or 1, and Φ is a set of linearly independant functions which correspond to fixed weights in the first layer of the perceptron. Many other algorithms which are similar in spirit to MLPs or perceptrons fall into this formulation. This is the case of Ho & Kashyap algorithm [Duda & Hart 73].

• **Learning Vector Quantization (LVQ)** : different LVQ algorithms have been proposed recently by Kohonen and co-workers [Kohonen 88]. They can easily be expressed as gradient rules for local cost functions.

• **LVQ1:** in this algorithm, only the closest reference vector is modified at a time, it is either moved towards or away from the input pattern, depending on the classification decision. For the local cost (table 1), $\delta_{z,t}$ is 1 if z = t and -1 otherwise, w_i is the weight vector to the i^{th} reference vector, cl(x) is the class of input pattern x and cl(wi*) is the class which has been assigned to the nearest reference w_{i*}.

• **LVQ2:** this is an improved version of LVQ1, where attention is focused on the points at the frontier of the classes. From the update rule, one can infer a cost function (table 1) where w_1 is the nearest reference

vector from the correct class and w_2 is the nearest reference vector in the network. Weights are modified only if x is not correctly classified and falls into a symetric window between w_1 and w_2, in all the other cases J = 0. This is easily incorporated into the cost function in table 1.

• **Topological maps (TM)** : TM is a well known unsupervised algorithm [Kohonen 88]. The local cost function in table 1 may be infered from the learning rule. V(t,w) denotes the neighborhood of cell w at time t in the output layer, w^* is the weight vector of the output cell at minimum distance from x and w_k the weights to cell k.

• **Competitive learning (CL)** : although many competitive learning algorithms use ad-hoc learning rules, it is possible to derive algorithms which minimize a local cost function through an adaptive gradient algorithm. We give below an algorithm which has been proposed in [Gallinari 1990]. It is a one layer net where the output cell i computes the following non linear function of its inputs: $f_i(x) = 1/ (1 + a \|x-w_i\|)$ where a is a constant. This function is always between 0 and 1 and decreases rapidly when the distance between x and w_i increases. For the local cost function (table 1) imax is the index of the winning output cell. This cell is selected according to a simple distance criterion which incorporates a sensitivity mechanism [Gallinari 90]. I indexes the set of the other output cells and α is a constant. Optimisation of J draws w_{imax} towards x and moves other weights away from x.

Many other connectionist algorithms can be expressed under this formalism. This is quite surprising because many of them have not been proposed as optimization techniques but rather as heuristic or common sense rules. An important consequence of this unified setting is that it is now possible to train complex architectures. A general algorithm for training in an optimal way such architectures has been described in [Bottou & Gallinari 91]. Cooperation of MLPs and LVQ has been studied in [Bollivier et al. 90] and used for speech recognition in [Bennani et al. 90]. This formalism can be extended to hybrid architectures where connectionnist techniques cooperate with classical algorithms, examples of hybrid MLPs + HMM (Hidden Markov Models) and MLPs + DTW (Dynamic Time Warping) may be found in [Bourlard 90, Bottou 91, Driancourt et al. 91].
It is clear now that many connectionist algorithms may be considered as adaptive techniques for the optimization of a local cost function. Of course, it would be more interesting to show that they optimize a global cost function C. We consider this problem below and show using different examples that it is possible to introduce variations of original algorithms which optimize global costs. For these new algorithms **the goal of learning** is thus explicitely defined.

Algorithm	local cost function : J(x,w)	update rule : $w = w - \varepsilon \nabla_w J(x,w)$
MLP, RBF, Adaline ...	J(y, f(x,w))	$w = w - \varepsilon \nabla_w J(y, f(x,w))$
Perceptron	$(\text{sgn } y - \text{sgn } w^t\Phi(x))w^t\Phi(x)$	$w = w - \varepsilon(\text{sgn } y - \text{sgn } w^t\Phi(x))\Phi(x)$
Ho-Kashyap	F(sgn y *f(x, w) - a) with a > 0	$w = w - \varepsilon_1 \nabla_w F$ $a = a - \varepsilon_2 \nabla_a F$
LVQ1	$\delta_{cl(x),cl(w_{i*})} \text{Min}_{w_i} \|x-w_i\|^2$	$w_i = w_i + \varepsilon\delta_{cl(x),cl(w_i)}(x-w_i)$ if w_i is the reference vector at minimum distance from x $w_i = w_i$ otherwise
LVQ2	$\|x-w_1\|^2 - \|x-w_2\|^2$ if x is badly classified O otherwise	$w_1 = w_1 + \varepsilon(x-w_1)$ $w_2 = w_2 - \varepsilon(x-w_2)$ $w_i = w_i$ if $i \neq 1,2$
Topological Maps	$\Sigma_{k \in V(t,w^*)} \|x-w_k\|^2$	$w_i = w_i + \varepsilon (x - w_i)$ if w_i falls into the window V(t,w*) $w_i = w_i$ otherwise
competitive learning	$1 - f_{imax}(x) + \alpha \Sigma_I f_i(x)$	$w = w - \varepsilon \nabla_w (f_{imax}(x) - \alpha \Sigma_I f_i(x))$

Table 1: *local cost functions and update rules which define the algorithms discussed in section 3. See text for explanation and notations.*

4. OPTIMIZING GLOBAL CRITERIA

In order to simplify the description, we will consider only algorithms for **pattern recognition** (PR) tasks. This is by no way limitative, but it allows to go deeper into details.

4.1 Formalism

Let x be a random variable which belongs to a class C_k, k = 1..M. We will denote by X the input space which is partitionned by the model into X_k , k = 1..M and by $(F_{ij}(x,w))_{i,j=1..M}$ a matrix of misclassification costs where i denotes the class of x and j the index of the region X_j where x falls according to the classifier. Most PR techniques use as a cost function the **average risk** :

$$C(w) = \Sigma_k \Sigma_m \int_X \theta_m(x,w) F_{km}(x,w) P_k p_k(x) dx \qquad (5)$$

Where P_k and $p_k(x)$ denote respectively the priors and the conditional distribution for class k. θ_m is the characteristic function of region X_m, i.e. $\theta_m(x,w)$ is 1 if $x \in X_m$ and 0 otherwise.

C depends both on the parameters w of the system and on the frontiers Λ_{mk} between regions X_m and X_k. An important difference with the classical Bayesian approach is that the F_{km} are not constant and depend on x and w. Note that θ_m depends on the partition $(X_i)_i$ and not on the actual class of x.

It can be shown [Tsypkin 71] that a necessary condition for an optimum of R is

$$\Sigma_k \Sigma_m \int_X \theta_m(x,w) \nabla_w F_{km}(x,w) P_k p_k(x) dx = 0 \qquad (6)$$

$$f_{sm} = \Sigma_k (F_{ks}(x,w) - F_{km}(x,w)) P_k p_k(x) = 0 \text{ if } x \in \Lambda_{sm} \qquad (7)$$

In the following, we show that this formalism can be used to describe most of the connectionist algorithms for supervised or unsupervised learning. The different algorithms are discussed below, the precise form of the cost functions and update rules are given in table 2.

4.2 Supervised learning

Supervised algorithms for classification fall into two categories, MLP-like algorithms which draw hyper-surfaces through the combination of linear or non linear functions and minimum distance based algorithms like LVQ.

• **MLP-like algorithms:** these algorithms use a discriminant function $f_i(x,w)$ for each class i. The decision regions are defined by the maximum onto the input space of these discriminant functions. Let $f = (f_1, ..., f_M)$ be the set of these functions for a M class problem. Most often, F_{km} is a convex function of the difference between the desired and computed outputs. In the expression of F_{km} (table 2), y defines the actual class k of x and the maximum component of f defines m the region where x is classified acording to the NN. In this case $C(w) = E_X\{F(y - f(x,w))\}$. Many algorithms can be obtained by choosing different functions for f and F, all supervised algorithms of section 3 except LVQ fall into this formulation with F = J. This is also true for TDNN and Recurrent Networks.

• **Minimum distance classifiers** : these classifiers usually have many ouputs for each class and compute similarity measures between reference vectors and input patterns. The decision rule is then to classify the input pattern into the class to which the nearest reference vector belongs. We first give a general formulation which extends the one used above and then present examples.

Let $F_{km}{}^i$ denote the loss incured when a pattern from class k is classified into class m, i denotes the most active cell from class k.

Let $\theta_m{}^j$ = 1 if x is classified into X_m with j the most active cell from the correct class; 0 otherwise

One can build the misclassification cost $S = \Sigma_k \Sigma_m \theta_m{}^i F_{km}{}^i$ and consider the risk $C(w) = E_X\{S\}$

$$C(w) = \Sigma_k \Sigma_m \int_X \theta_m{}^i(x,w) F_{km}{}^i(x,w) P_k p_k(x) dx \qquad (8)$$

the general adaptive algorithm which optimizes C is then:

$$w = w - \varepsilon \nabla_w F_{km}{}^i(x,w) \qquad (9)$$

if x from C_k has been classified into C_m while the maximally active cell from the correct class is i.

Examples

• **LVQ1**: for LVQ1 cost function (table 2), G is a similarity function between x and w. For example in the classical LVQ1, $G(x,w) = \|x-w\|^2$.

• **Extended LVQ**: the algorithm has been described in [Gallinari 90]. The learning rule draws the most active cell from the correct class towards the input pattern and simultaneously to move the other cells away from this pattern. However it has been found efficient to move cells from the other classes farther away than those of the correct class. Output cells compute the same function as for CL, that is: $f(x, w_i) = 1/(1 + a\|x-w_i\|)$ for cell i. In the cost function for cell i (table 2), w_j denotes the weight vector to cell j, I indexes cells from the correct class k except cell i and J indexes cells from other classes. α and β are constants which are used to implement the mechanism jnst described above. This cost is a soft version of the local LVQ2 cost, this allows to compute its expectation.

4.3 Unsnpervised learning

For unsupervised learning, there is no a priori information about the decision regions. Therefore, one can set $F_{km} = F_k$ for all k and the quality of the partition is measured by

$$C(w) = \Sigma_k \int_X \theta_k(x,w) F_k(x,w) p(x) dx \qquad (10)$$

where p is the mixture probability on X and the local cost is $J = F_k(x,w)$ if x is classified in region k. The learning rule for these algorithms is thus

$$w = w - \varepsilon \nabla_w F_k(x, w) \quad \text{if x has been classified in region k} \qquad (11)$$

the global cost function is the expectation of the F_is.

• **Extended topological maps** : this is a variation of the original TM algorithm [Angeniol et al. 88]. It makes use of a continuous neighborhood on the output layer and an influence function which measures how modifications in one cell will influence the modification in others. The extended TM cost function in table 3 may be easily infered from this algorithm. β_{ik} is a decreasing function of the distance in the output layer between cells i and k according to the topology of this layer.

• **Quantization** : many classical algorithm enter the same formalism, for example the classical k-means algorithm corresponds to the global cost function :

$$C(w) = E_X\{\min_i \|x-w_i\|^2\} \qquad (12)$$

The same algorithm can be used in a supervised way if it is applied separately to the different classes, the decision rule being to classify an input pattern according to the class of the nearest reference vector. This is just another algorithm which belongs to the minimum distance family described in 4.

The **competitive learning** algorithm from section 3 also enters this formalism with $F_i(x,w) = J(x,w)$.

supervised algorithm	**cost function** : $F_{km}(x,w)$ or $F_{km}{}^i(x,w)$	**learning rule**
MLP, RBF, ...	$F(y - f(x,w))$	$w = w + \varepsilon F'((y-f(x,w))*f'(x,w)$
LVQ1	$(1_{k=m} - 1_{k\neq m})G(x,w)$	$w = w + \varepsilon (1_{k=m} - 1_{k\neq m})\nabla_w G(x,w)$
extended LVQ	$(1-f(x,w_i))+\alpha\Sigma_{j\in I} f(x,w_j)+\beta\Sigma_{j\in J} f(x,w_j)$	$w = w - \varepsilon\nabla_w F_{km}{}^i(x, w)$ if x from C_k has been classified into class m while the maximally active cell from the correct class k is i.
unsupervised algorithm	**cost function** : $F_I(x,w)$	**learning rule** : $w = w - \varepsilon\nabla_w F_I(x, w)$
extended TM	$\Sigma_k \beta_{ik}\|x-w_k\|^2$	$w_k = w_k + \varepsilon\beta_{ik}(x-w_k)$ if i is the maximally active output cell.
quantization	$\|x-w_i\|^2$	$w_i = w_i + \varepsilon(x-w_i)$ if w_i is the reference at minimum distance from x $w_i = w_i$ otherwise

Table 2: *cost functions and update rules which define the algorithms discussed in section 4. See text for explanation and notations.*

5. EXTENSIONS

The above formalism can be extended to include several other paradigms, we briefly mention below some of them :

• if J is not differentiable or continuous, pseudo-gradient algorithms may be used to optimize the global cost function.

• very often, it is necessary to solve a constrained optimization problem. This allows to incorporate some knowledge about the task into the architecture of the network or into the nature of the solution. It is easy to incorporate equality or inequality constraints in the above formalism.

• For some algorithms (e.g. LVQ) a good initialization is very important. For hybrid architectures it may be useful to first train the different modules separately to get a good starting point and then to perform a global training of the whole architecture so that the parameters are optimal for the global task [Bollivier et al. 90].

• The same formalism may be used to train classical learning systems, e.g. HMM . [Bottou 91] or DTW [Driancourt 91]. Many data analysis techniques also fall into this formalism (see e.g. [Gallinari et al. 91]).

6. CONCLUSION

We have presented a general formalism for connectionist algorithms and illustrated its use on several examples. It provides a unified description for a large family of NN which allows to study general properties of these techniques and to combine them. It also provides an analytical formulation of the goal of learning for these systems. We have shown how it is possible to modify existing algorithms so that they fit the formalism. This formulation allows to go further both in the analysis and the design of connectionist systems and was a necessary step for the training of complex learning systems. Another important consequence is that this unified description of NN can be the basis of a specification langage for software implementation of NN libraries. Finally, we have briefly discussed some extensions and practical difficulties which have been encountered in different implementations.

Aknowledgements
We thank X. Driancourt for many fruitful discussions.

7. REFERENCES

Angeniol B, de La Croix Vaubois G., Le Texier J.Y. (1988): *Self organizing feature maps and the travelling salesman problem*, Neural Networks, Vol. 1, n° 4, 289-293.

Bennani Y., Chaourar N., P. Gallinari, Mellouk A.(1991): *Validation of neural net architectures on speech recognition tasks*, IEEE ICASSP 91.

Benveniste A., Metivier M., Priouret P. (1987): *Algorithmes adaptatifs et approximations stochastiques*, Masson.

Bollivier M., Gallinari P., Thiria S.(1990): *Cooperation of Neural Nets for Robust Classification* - IJCNN 90 San Diego.

Bottou L. (1991): Une approche théorique de l'apprentissage connexionniste; applications à la reconnaissance de la parole. Phd université d'Orsay.

Bottou L., Gallinari P. (1990): *A Framework for the Cooperation of Learning Algorithms* - Proc. of NIPS 90.

Bourlard H. (1990): *How Connectionist Models could Improve Markov Models for Speech Recognition* - Proc. of the Int. Symp. on NN for Sensory and Motor systems, Dusseldorf, F.R.G., Ed R. Eckmiller.

Bourlard H., Wellekens C.J. (1988) : *Links between Markov Models and Multilayer Perceptrons*, NIPS'88, 502-510, Morgan Kaufman. Denver.

Driancourt X., Bottou L., Gallinari P. (1991): *Multi Layer Perceptron, Learning Vector Quantization and Dynamic Programming: Comparison and Cooperation* - Proceedings of IEEE-IJCNN 91 (Seattle).

Duda R. O., Hart P. E.(1973): *Pattern Classification and scene analysis*, Wiley.

Gallinari P. (1990): *A neural net classifier combining unsupervised and supervised learning*. INNC 90 Paris.

Gallinari P., Thiria S., Badran F., Fogelman-Soulié F. (1991): *On the relations between discriminant analysis and multi-layer perceptrons*. Neural Networks, vol.4, N°3, 349-360.

Gallinari P., Schwenk H. (1991): internal report University Orsay.

Robinson A.J., Niranjan M., Fallside F. (1988): *Generalising the nodes of the error propagation networks.* Internal report CUED/F-INFENG/TR 25 Cambridge University.

Rumelhart D. E., Hinton G. E., Williams R. J. (1986): *Learning internal representations by error propagation*, in Parallel distributed processing, Rumelhart D.E., Mc Clelland J. L. Eds. Vol 1, 318-362. MIT Press.

Tsypkin Ya (1971): *Adaptation and learning in automatic systems*, Mathematics in science and engineering vol 73, Academic Press.

Widrow B., Hoff M.E.(1960): *Adaptive switching circuits*. IRE WESCON Conv. record, part 4 1960, pp96-