On the Optimization of a Synaptic Learning Rule

Samy Bengio Yoshua Bengio Jocelyn Cloutier Jan Gecsei

Université de Montréal, Département IRO
Case Postale 6128, Succ. "A",
Montre'al, QC, Canada, H3C 3J7
e-mail: bengio@iro.umontreal.ca

Abstract

This paper presents a new approach to neural modeling based on the idea of using an automated method to optimize the parameters of a synaptic learning rule. The synaptic modification rule is considered as a parametric function. This function has *local* inputs and is the same in many neurons. We can use standard optimization methods to select appropriate parameters for a given type of task. We also present a theoretical analysis permitting to study the *generalization* property of such parametric learning rules. By generalization, we mean the possibility for the learning rule to learn to solve *new* tasks. Experiments were performed on three types of problems: a biologically inspired circuit (for conditioning in *Aplysia*), Boolean functions (linearly separable as well as non linearly separable) and classification tasks. The neural network architecture as well as the form and initial parameter values of the synaptic learning function can be designed using *a priori* knowledge.

1 Introduction

Many artificial neural network models have been recently proposed (see [17] and [18] for detailed reviews) and each of them uses a different (but constant) synaptic update rule. We propose in this paper to use optimization methods to *search for new synaptic learning rules.* Preliminary studies on this subject were reported in [4, 5, 6].

Many biologically inclined researchers are trying to explain the behavior of the nervous system by considering experimentally acquired physiological and biological data for constructing their models (see for example [14] and [7]). These biologically *plausible* models constrain the learning rule to be a function of information *locally* available to a synapse. However, it has not yet been shown how such models could be efficiently applied to difficult engineering or artificial intelligence problems, such as image or speech recognition, diagnosis, prediction, etc. Another approach, preferred by engineers, emphasizes problem solving, regardless of biological plausibility (for example, error backpropagation [23]). The above two classes of models seem to be growing more and more apart. An objective of this paper is to contribute to fill the gap between the two approaches by searching for new learning rules that are both biologically plausible and efficient compared to specialized techniques for the solution of difficult problems.

2 Learning Rule Optimization

The most remarkable characteristic of a neural network is its capacity to adapt to its environment, in the sense that it can learn from experience, and generalize when presented new stimuli. In both biologically motivated and artificial neural networks, this adaptation capacity is represented by a *learning rule*, describing how connection weights (synaptic efficiency) change.

Even though it is generally admitted that the learning rule has a crucial role, neural models commonly use *ad hoc* or heuristically designed rules; furthermore, these rules are independent of the learning problem to be solved. This may be one reason why most current models (some with sound mathematical foundation) have difficulties to deal with hard problems. In this paper we propose to improve a learning rule by adapting it to the kind of problems to be solved.

To do this, we consider the synaptic learning rule as a *parametric function*, and we optimize its parameters using standard optimization tools, such as gradient descent [23], genetic algorithms [13], and simulated annealing [24]. We make the following assumptions:

- The same rule is used in many neurons (this constraint may be relaxed to one rule for each type of neuron or synapse[1]). It is not plausible that every synapse or neuron in a network has its own rule. Actually, neural models described in the literature use a single learning rule (e.g. Hebb's [16]), which dictates the behavior of every neuron and synapse.

- There exists a relation (possibly stochastic) between synaptic update and some information locally available to the synapse, that corresponds to the learning rule (that is, synaptic update is not totally random).

- This relation may be approximated by a parametric function

  $$f(x_1, x_2, \cdots, x_n; \theta_1, \theta_2, \cdots, \theta_m) \tag{1}$$

  where x_i are variables of the function and θ_j are a set of parameters.

2.1 Variables and Parameters of the Learning Rule

Since the domain of possible learning algorithms is very large, we propose to constrain it by using in equation (1) only already known, biologically plausible synaptic mechanisms. Consequently, we consider only local variables, such as presynaptic activity, postsynaptic

[1] It was discovered by biologists that there exist different types of neurons and synapses in the brain, however their characterization is far from being complete [11].

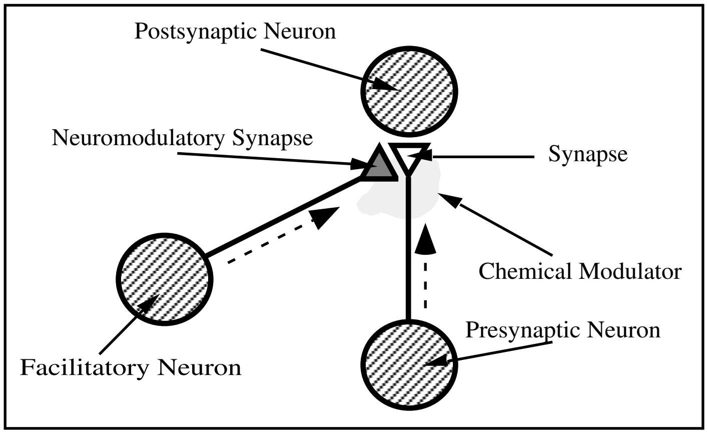


Figure 1: Elements found in the vicinity of a synapse, which can influence its efficacy.

potential, synaptic strength, the activity of a facilitatory neuron, and the concentration of a diffusely acting neuromodulator. Figure 1 shows the interaction between those elements.

Constraining the learning rule to be biologically plausible should not be seen as an artificial constraint but rather as a way to restrain the search space such that it is consistent with solutions that we believe to be used in the brain. This constraint might ease the search for new learning rules (Figure 2).

2.2 General Form of the Learning Rule

From the above, and denoting $w(i,j)$ as the weight of the synapse from neuron i to neuron j, the general weight update function will have the form

$$\Delta w(i,j) = \Delta w(x_1, x_2, \cdots, x_n; \theta_1, \theta_2, \cdots, \theta_m) \tag{2}$$

The synaptic update $\Delta w(i,j)$ of a synapse $i \rightarrow j$ is computed using (2), as a function of variables x_k, local to this synapse and a set of parameters θ_k. It is those parameters that we propose to optimize in order to improve the learning rule.

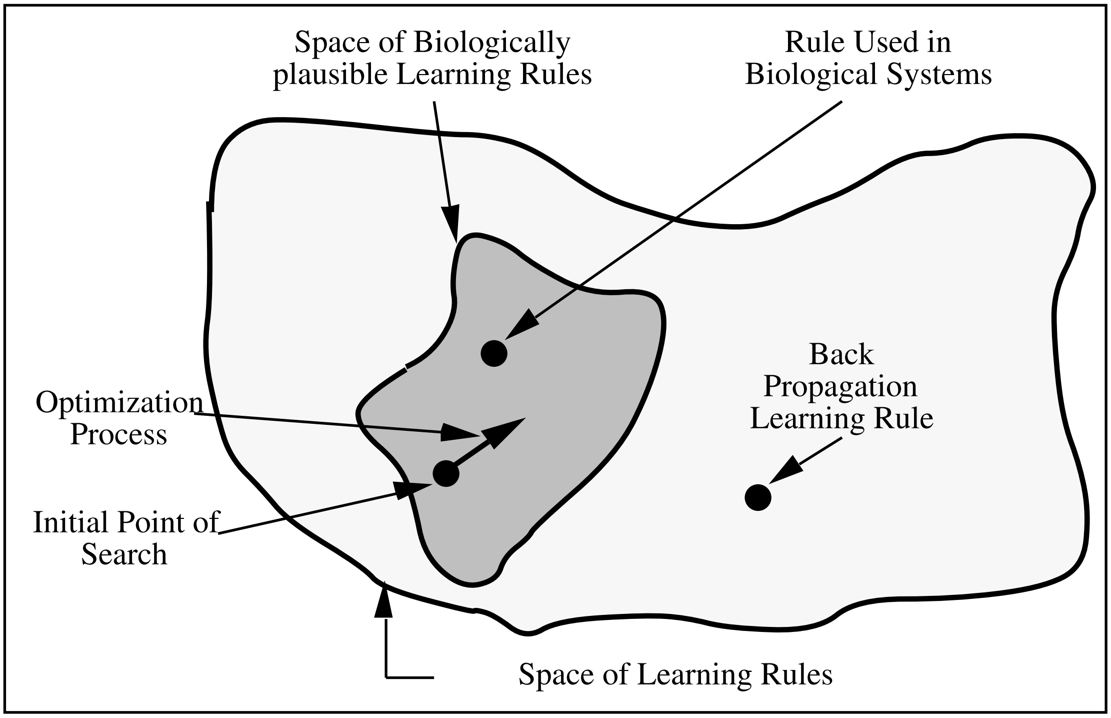

Space of Biologically plausible Learning Rules
Rule Used in Biological Systems
Optimization Process
Back Propagation Learning Rule
Initial Point of Search
Space of Learning Rules

Figure 2: Constraining the space of learning rules considered

2.2.1 Example of a Parametric Learning Rule

Hebb's rule is probably the best known learning rule in connectionist models [16]. It suggests that a biologically plausible synaptic modification is to increase the weight of a connection when the presynaptic and postsynaptic neurons are active simultaneously. Under its most simple expression, Hebb's rule may be written as a parametric function as follows:

$$\Delta w(i,j) = \theta\ y(i)\ x(j) \tag{3}$$

where Δw is the weight update, $y(i)$ is the activity of presynaptic neuron i, $x(j)$ is the potential of postsynaptic neuron j, and θ a correlation constant. In this case, $y(i)$ and $x(j)$ are the variables and θ a parameter.

2.3 Form of the Rule

Once the variables and parameters have been chosen, the learning rule must be given a precise form. In (3), it is simply the product of the variables and of the parameter. Currently available biological knowledge can help us design a more general form of the learning rule. For instance, it is now accepted that many mechanisms may interact in a synapse simultaneously but with different time constants, which suggests the inclusion of delays in the learning rule.

We may also model a synapse with more details, considering for instance local interaction at the dendritic tree level (a synapse is then influenced by the neuron's local potential instead of its global activity).

2.4 Task Diversification

Optimizing a learning rule to solve a given single task is an interesting and non-trivial problem in itself that will be discussed in this paper. We want this rule to be able to perform adequately on new instances of the task. However, it may be more interesting (and more difficult) to find a learning rule that can be used successfully on a number of different learning tasks. During the optimization process, the same rule (with the same parameters) must be used on all tasks considered. Thus, we have to optimize a learning rule with respect to its simultaneous performance on different neural networks learning different tasks. This constraint, suggested by [8], should yield rules of more general applicability. This raises the question of generalization over tasks. Let us suppose that we optimize a rule using a *training set* of tasks sampled from a certain function space. Generalization over tasks is the expected performance of the learning rule on a new function sampled from the same function space, i.e., the same class of functions.

2.5 Parameter Optimization

Once the form of $\Delta w()$ is determined, we have to search for values of parameters that optimize the learning rule's ability to solve different tasks. We will now define the problem of optimizing a learning rule.

2.5.1 Learning and Optimization

Let X be a random variable with probability distribution P_X fixed but unknown, and let $\phi : X \rightarrow Y$ be an unknown function. For example, X may be a digital representation of a pixel matrix and Y the symbol (e.g. numbers, letters) corresponding to the image X.

Let J be a scalar cost function. For example we can choose J to be the mean square criterion:

$$J(x, y) = (x - y)^2 \tag{4}$$

The goal of a learning system is then to produce a parametric function $\hat{\phi}(\theta) : X \rightarrow Y$ which minimizes

$$C = \int J(\hat{\phi}(x; \theta), \phi(x)) \; P_X(x) \; dx \tag{5}$$

by finding adequate parameters θ[2]. In order to minimize C with a supervised learning system, we usually proceed using a set of N examples $(x_i, \phi(x_i))$, for $i = 1 \rightarrow N$, each chosen in (X, Y) independently using P_X.

Training performance of such a system can be measured in terms of the difference between ϕ and $\hat{\phi}$ for the N examples:

$$\hat{C} = \sum_{i=1}^{N} J(\hat{\phi}(x_i; \theta), \phi(x_i)) \tag{6}$$

The *generalization* performance of such a system measures the difference between ϕ and $\hat{\phi}$ for points other than those used to calculate $\hat{C}$. To quantify the generalization property of a learning system, we now introduce the standard notion of *capacity*.

Let $z \in Z = (x, y) \in (X, Y)$ and let $G(z; \theta)$ be the set of parametric functions $g(z; \theta)$.

[2] For example, in a neural network, θ is the set of weights.

For example in neural networks, g represents a network architecture and a cost function, θ is the set of weights and z is a pair of input and corresponding desired output given to the network. More precisely, $g(z;\theta) = J(\hat{\phi}(x;\theta), \phi(x))$.

When $J \in \{0,1\}$, Vapnik defines in [25] the *capacity* h of $G(z;\theta)$ as the maximum number of points $x_1, x_2, \ldots, x_h$ that can always be divided into two distinct class with $G(z;\theta)$. Vapnik then describes an extension for the case where $J \in \mathbb{R}$. In this case, capacity is defined as the maximum number of input/output pairs $z_1, z_2, \ldots, z_h$ that can always be learned by $G(z;\theta)$ with a cost less than a threshold chosen to maximize the capacity h.

Theoretical studies such as [1] and [26] give an upper bound on the number N of necessary instances of Z required to achieve generalization with a given maximal error when training is performed with a learning system with capacity h:

$$\epsilon \leq \mathcal{O}(\sqrt{\frac{h}{N}} ln \frac{N}{h}) \tag{7}$$

where ϵ is the difference between training error (using the N examples) and generalization error expected over all examples. This means that for a fixed number of examples N, starting from $h = 0$ and increasing it, one finds generalization to improve until a critical value of the capacity is reached. After this point, increasing h makes generalization deteriorate. For a fixed capacity, increasing the number of training examples N improves generalization (ϵ asymptotes to a value that depends on h). The specific results of [25] are obtained in the worst-case, for any distribution P_X.

2.5.2 Optimization of the parameters

Let L be a learning rule defined by its structure (fixed) and its parameters θ (considered to be variable). Optimizing the learning rule requires to search for the values of parameters that minimizes a cost function.

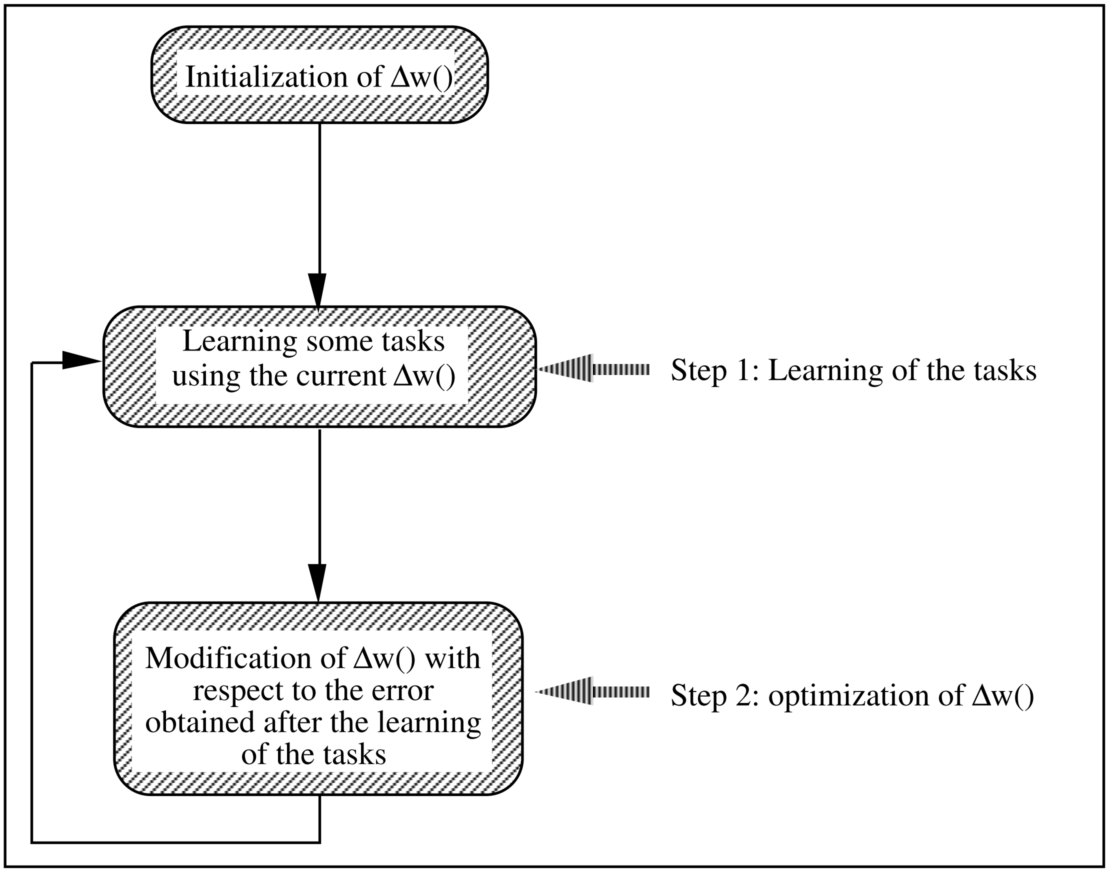
Let $\{R_1, R_2, ..., R_n\}$ be a set of neural networks trained on n different tasks (and possibly having different structures), but using the same learning rule L. If C_i is the cost (as defined for instance by equation (6)) obtained with neural network R_i after being trained on its task, then the global cost function we wish to minimize is

$$C^* = \sum_i C_i \tag{8}$$

Furthermore, if we want a learning rule to be able to solve any task, we should, in theory, optimize C^* with respect to the cost resulting from the learning of every possible instance of every one of the n tasks. Since this is usually impossible in practice, we can only find an approximation having the generalization error decrease with the increasing number of tasks used to optimize C^*. This follows from Vapnik and Chervonenkis' theorem [26], as long as those tasks are representative of the sets of all possible tasks (which we usually cannot verify).

More formally, we can define the capacity of a parametric learning rule $G(z;\theta)$ as a measure of the number tasks z that can always be learned using G. Thus equation (7) holds for the generalization error of the learning rule where h is the capacity of the parametric learning rule (which is in fact a function of the number of parameters of the learning rule), N is the number of tasks used to optimize the parameters, and ϵ is the difference between training error (training tasks) and generalization error (on new tasks).

Thus, we can draw several conclusions from this extension. For example, it becomes clear that the expected error of a learning rule over new tasks should decrease when increasing the number of tasks (N) used for learning the parameters θ. However, it could increase with the number of parameters and the capacity of the learning rule class if an insufficient number or variety of training tasks are used in the optimization. This justifies the use of *a-priori* knowledge in order to limit the capacity of the learning rule. It also appears more clearly that the learning rule will be more likely to generalize over tasks which are similar to those used for the optimization of the rule's parameters. In consequence, it is advantageous to use, for the optimization of the learning rule, tasks which are representative of those on which the learning rule will be ultimately applied.



Initialization of Δw()
Learning some tasks using the current Δw()
Step 1: Learning of the tasks
Modification of Δw() with respect to the error obtained after the learning of the tasks
Step 2: optimization of Δw()

Figure 3: Optimization steps

2.6 Optimization Process

We use a two-step optimization process as shown in Figure 3. The form of the learning rule is defined and the parameters are initialized either to random values within reasonable bounds, or to values corresponding to biological evidence (an example of this can be found in Section 3). We also define the set of tasks to be used in the process. Then we use the following algorithm:

1. We train n networks simultaneously on n tasks by using the current learning rule. Since $\Delta w()$ is, with high probability, initially far from optimal, it is very unlikely that all tasks will be solved adequately. After a fixed number of learning steps, we compute the error obtained in each task from equation (6).

2. The objective function (such as equation (8)) is minimized by updating the parameters θ of $\Delta w()$ according to the optimization method used.

3. We return to step 1, using the new parameters of the learning rule, until we reach an acceptable learning rule.

When the new learning rule is able to solve adequately all n tasks, we may want to test its capability to *generalize* on tasks which were not used in the optimization process.

Many different optimization methods may be used to improve $\Delta w()$. We can use local methods such as gradient descent, or global methods such as simulated annealing [24] or genetic algorithms [19, 13]. Local methods are usually faster, but they can get trapped in local minima, whereas global methods are less sensitive to local minima but usually slower. Hybrid gradient descent/genetic algorithms were recently suggested [9, 28]. In our experiments we used gradient descent, simulated annealing, as well as genetic algorithms.

2.7 Problem complexity

It is interesting to discuss briefly the complexity of the above optimization problem. We already have some knowledge of the complexity of neural network learning. Deciding if in the space of parameters of a neural network N there exists an adequate solution to an arbitrary task T is equivalent to the satisfiability problem, which is NP-complete [20]. Consequently, the search for such solution in N must be NP-hard.

However, experimentation shows that a complex task may be learned in *polynomial* time by a neural network if a sound approximation is acceptable [18]. General optimization methods such as gradient descent or simulated annealing can usually give good sub-optimal solutions in polynomial time, even if they may need exponential time for the optimal solution.

In our experiments we clearly cannot aim at exact optimization. Instead, we allocate a polynomial amount of time to each network using the current learning rule to solve its task. We are thus searching for a learning rule that can solve a set of tasks reasonably well in reasonable time.

3 Preliminary Experiments

To test the feasibility of our approach to the optimization of learning rules, we have perform preliminary experiments with relatively simple problems [4, 5, 6, 2, 3]. In this section we summarize the results. In these experiments, we used either gradient descent, simulated annealing, or genetic algorithms as optimization methods. The tasks were the following: conditioning, boolean functions and classification problems. Although preliminary results are positive, experimentation with more complex problems is needed in order to find useful synaptic learning rules.

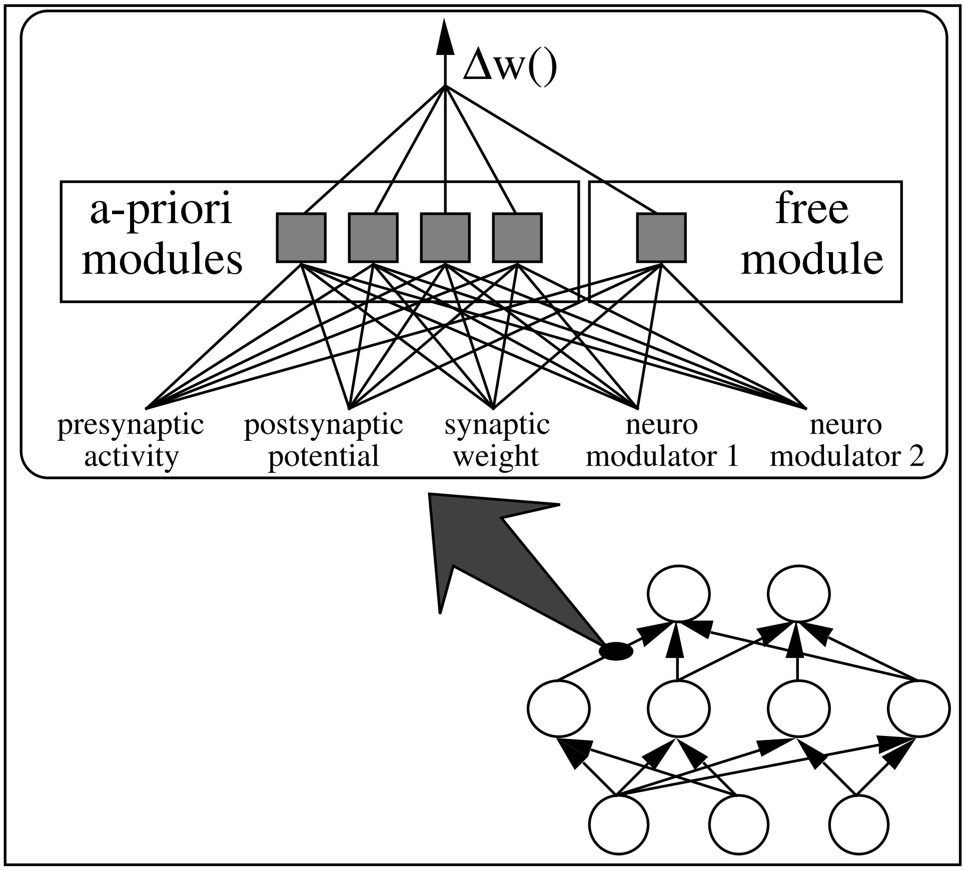


Figure 4: *A priori* knowledge utilization

3.1 Form of the Learning Rule

To facilitate the search for an optimal synaptic update rule $\Delta w()$, it is important to choose an adequate form of the learning rule. Here, by adequate we mean a form sufficiently rich to express a good solution (which is problem dependent), but sufficiently constrained to ease the search of the solution[3]. To do so, we can use some biological knowledge. Figure 4 shows the general form of the learning rules we use in the experiments. It consists of a certain number of *a priori* modules representing known or hypothesized biological synaptic characteristics. The resulting rule reflects a combination of *a priori* and free modules. The parameters determine the relative influence of each module on $\Delta w(\cdot)$.

Equation (9) is a concrete example of a learning rule used in the experiments that will be described in this section.

$$\begin{aligned}\Delta w(i,j) &= \theta_0 + \theta_1\, y(i) + \theta_2\, x(j) + \theta_3\, y(mod(j)) + \\ &\quad \theta_4\, y(i)\, y(mod(j)) + \theta_5\, y(i)\, x(j) + \theta_6\, y(i)\, w(i,j) \end{aligned} \tag{9}$$

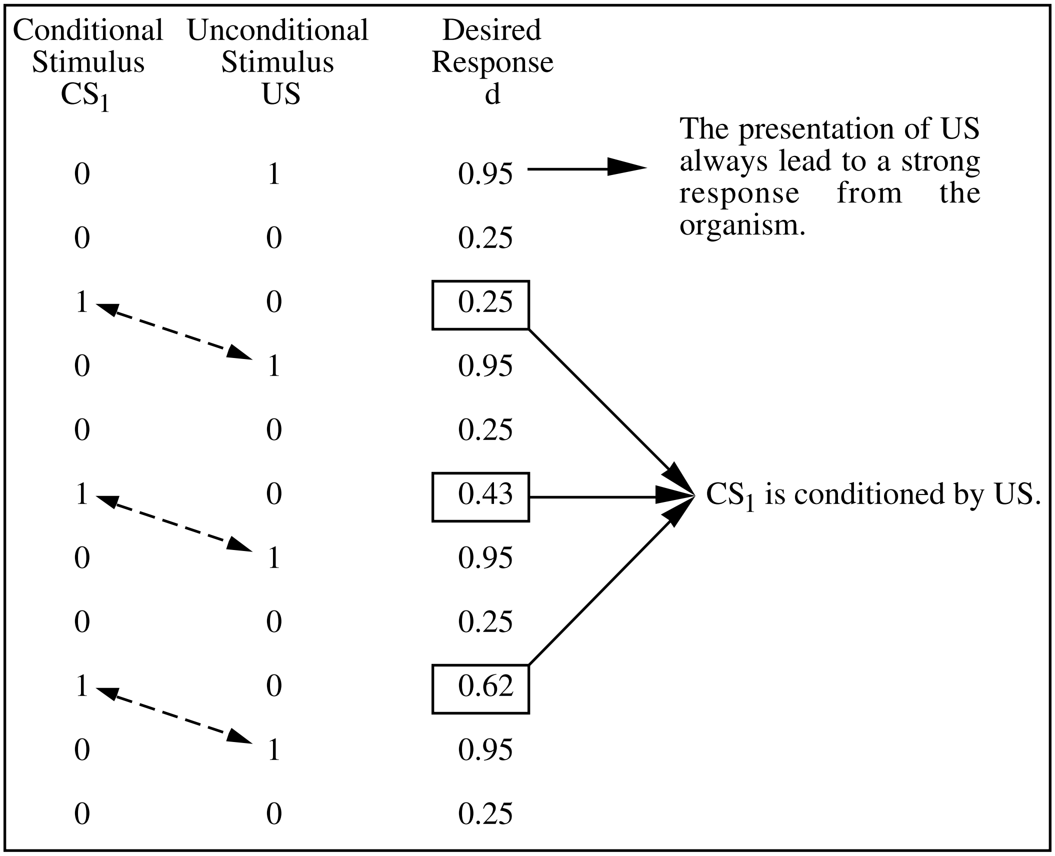
This is an instance of the general form described in figure 4. The function computed by this equation has 7 parameters, and integrates the following a priori modules:

- $y(i) \cdot x(j)$, Hebb's rule.
- $y(i) \cdot y(mod(j))$, where $y(mod(j))$ is a modulatory activity (chemical or neural). Hawkins describes in [14] a conditioning model for Aplysia using such a mechanism.
- $y(i) \cdot w(i,j)$, where $w(i,j)$ is the synaptic weight at a the previous time frame. This term suggested in Gluck's conditioning models [12], permits gradual forgetting.

[3] Moreover, as we have seen in section 2.5.2, if the form of the learning rule is too rich, capacity may be too high to reach good generalization performance.

3.2 Conditioning Experiments

The goal of our first experiment is to discover a learning rule which is able to reproduce some classical conditioning phenomena in animals. Conditioning experiments, first described by Pavlov [22] are well known through experimental studies. For our experiments, we used Hawkins' model [14]. We studied the following phenomena:



Conditional Stimulus CS_1	Unconditional Stimulus US	Desired Response d
0	1	0.95
0	0	0.25
1	0	$\boxed{0.25}$
0	1	0.95
0	0	0.25
1	0	$\boxed{0.43}$
0	1	0.95
0	0	0.25
1	0	$\boxed{0.62}$
0	1	0.95
0	0	0.25

The presentation of US always lead to a strong response from the organism.

CS_1 is conditioned by US.

Figure 5: Conditioning by a sequence of stimuli and responses. The real sequence used for the experiments is much longer.

Habituation: initially, a conditional stimulus CS_1 (e.g. a red light presented to an animal) produces a small response (e.g. the animal salivates slightly). By presenting CS_1 repetitively, the response gradually vanishes (i.e., the animal gets used to the stimulus, and reacts to it less and less).

Conditioning: a conditional stimulus CS_1 is followed by an unconditional stimulus US (e.g. a red light followed by food). The response to CS_1 grows gradually (the animal salivates before seeing the food, as soon as the red light is turned on).

Blocking: after CS_1 has been conditioned, a second conditional stimulus CS_2 (e.g. a green light) is presented to the organism simultaneously with CS_1, both followed by an unconditional stimulus US. In that case, CS_2 is not conditioned (the animal will not salivate on green light only).

Second order conditioning: after CS_1 has been conditioned, CS_2 may be conditioned by presenting CS_2 followed by CS_1 (the animal will begin to salivate when it sees the green light, knowing that the red light follows, and that it is usually followed by food).

Extinction: after CS_1 has been conditioned, repetitive presentation of CS_1 not followed by US will reduce the animal's response to its original level (when no food follows the red light, the animal tends to lower its saliva response with time, eventually reaching its initial unconditioned level).

Figure 5 shows the way we modeled one of those behaviors (conditioning) with a sequence of stimuli and associated responses.

The form of the learning rule we used is equation (9), while the network architecture is shown in figure 6. It is inspired by Hawkins' work [15, 14].

In this network, CS_1 and CS_2 are conditional stimuli, US is an unconditional stimulus, FN is a facilitatory neuron, and MN a motor neuron (it represents the animal's response to stimuli). CS_1 and CS_2 influence the motor neuron (through connections toward MN), and these connections are themselves modulated by a facilitatory neuron which takes into account two consecutive states of the system through connections with delay (e.g. when CS_1 is activated at time t and US at time $t + 1$).

In the conditioning experiments, the cost function to minimize (with gradient descent in this case) is defined following equations (6) and (8) where $\hat{y}$ is the actual response of the network given an input sequence of stimuli, and y is the target behavior for the same

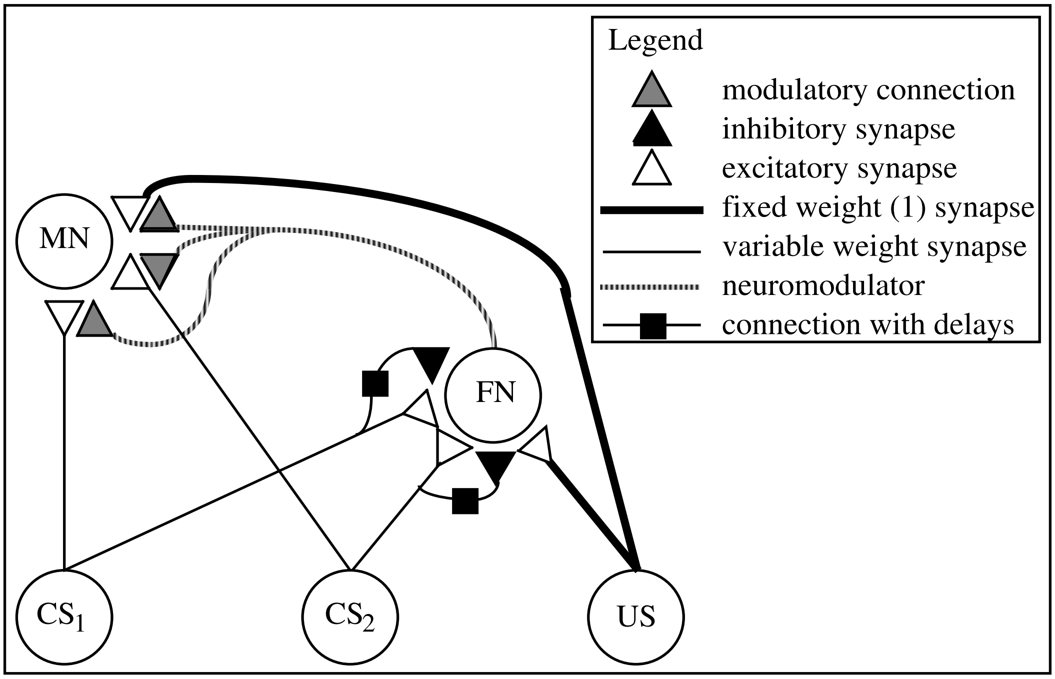


Figure 6: Neural network used for the conditioning experiments

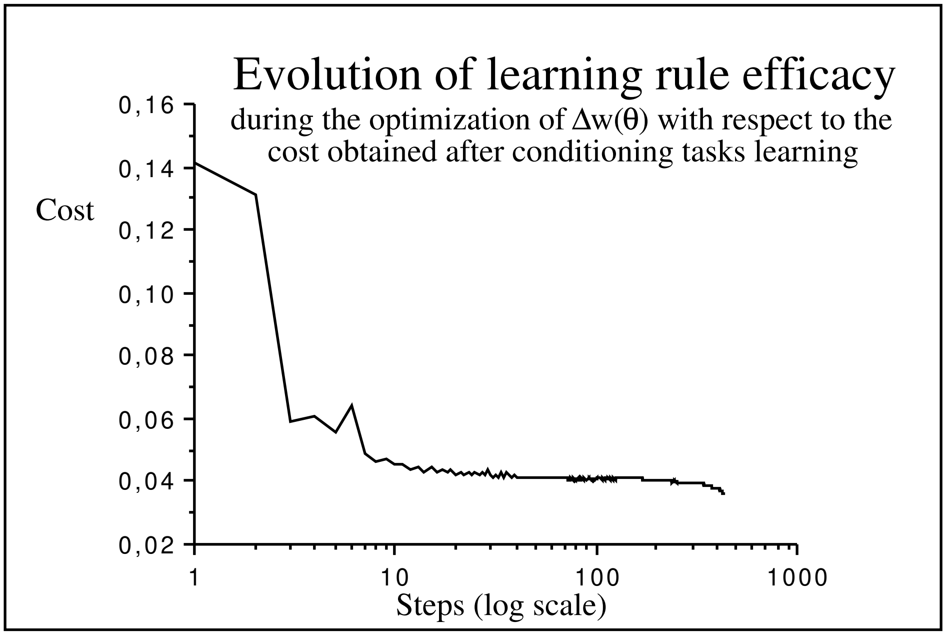


Figure 7: Evolution of the learning rule efficacy

input sequence. By fixing the initial values for some parameters (e.g. in equation (9), θ_3 is initialized to 1) and by initializing the other parameters to random values (in the range

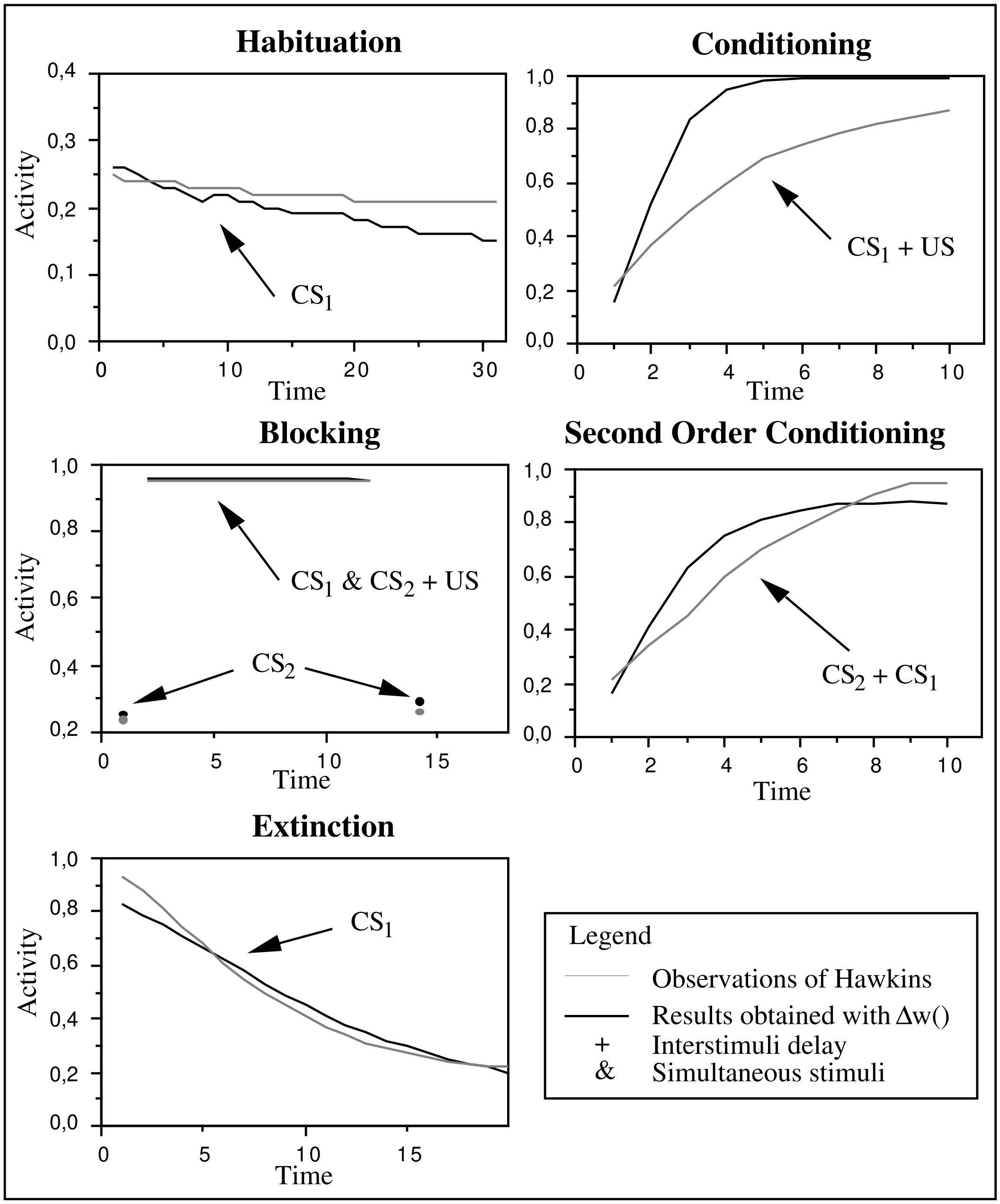


Figure 8: Conditioning tasks computed by our new learning rule

$[-1, 1]$), it was possible to find a set θ such that all five conditioning behaviors could be learned by the network in figure 6 with initial random weights. Figure 7 shows the evolution of the cost function during optimization. Figure 8 shows the results of all five conditioning tasks obtained from our new learning rule (an extensive analysis of the resulting rule will be done in a future paper). The results obtained by this learning rule are similar to Hawkins' experimental results [14].

3.3 Experiment with Boolean Functions

The goal of these experiments is to explore in a very simple setting the possibility of optimizing a learning rule that could be used to train a network with hidden units. They allowed us to evaluate the applicability of our method to a simple computational problem.

We used again the same learning rule from equation (9). Fully connected networks with two inputs, a single output and one hidden unit were trained to perform linearly separable functions (such as AND, OR) and non linearly separable functions (such as XOR, EQ). Information provided to hidden units about their contribution to errors is fed through backward paths, with neurons that may modulate synaptic change on corresponding forward paths (Figure 9).

As for the conditioning experiments, a cost function was defined in terms of the difference between real and expected learning behavior of the network. This target behavior consists in learning some Boolean function within a cycle of 800 presentations of randomly selected noisy Boolean input patterns. A Gaussian noise was added to each input pattern for better generalization of the learning rule. Before each cycle, the network weights were initialized randomly, in order to allow the resulting rule to be as insensitive as possible to the network's initial conditions. Rule parameters θ were updated after each cycle. The results are summarized in Table 1.

Two optimization methods were used: gradient descent and simulated annealing. Gradient descent proved to be faster but sensitive to initial values of θ, whereas simulated annealing was slower (around 500 times slower) but insensitive to parameter initialization. In order to verify that gradient descent and simulated annealing were more efficient than random search, we also tried the following approach. For the random search, 50000 vectors of θ, each with 7 parameters within $[-1, 1]$, were chosen and those corresponding to the best learning performance on the training tasks were kept. The networks trained with that rule

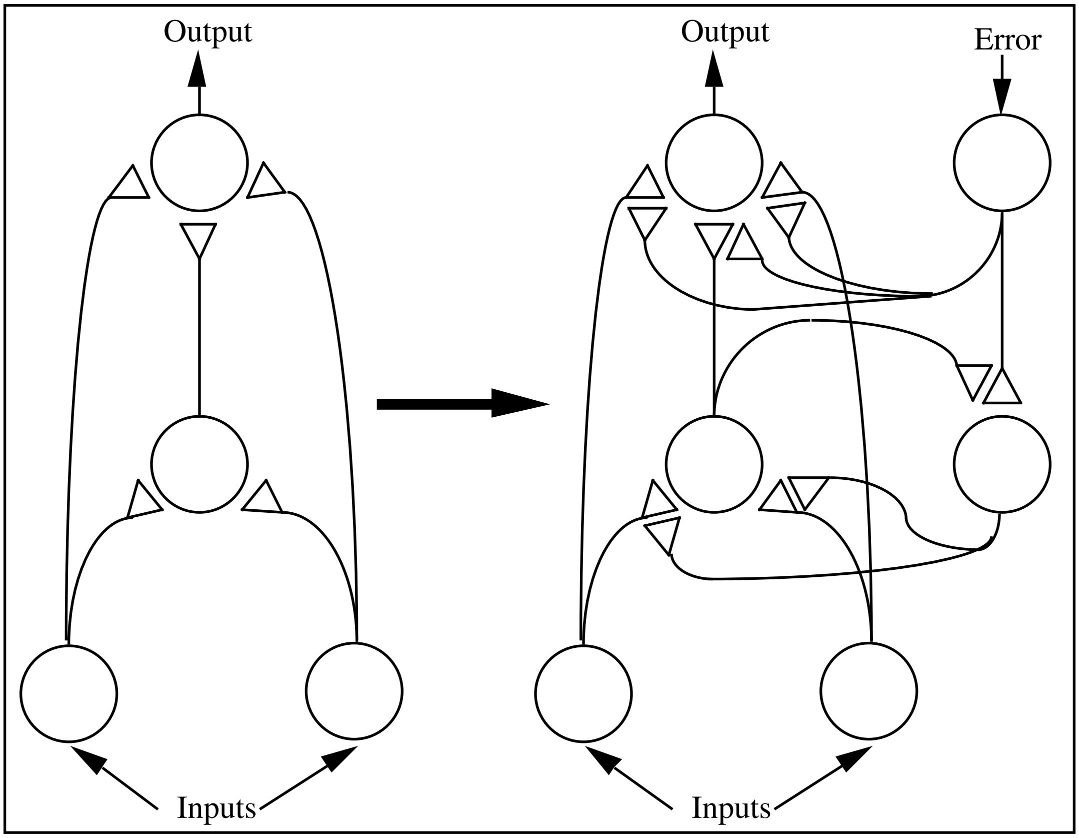


Figure 9: Architecture transformation to enable local evaluation of the network error

could not learn the non linearly separable functions completely (25% error at best), whereas the networks trained with a rule obtained with gradient descent or simulated annealing were able to learn the training tasks perfectly. Furthermore, this rule could also generalize to new (but similar) tasks including non linearly separable functions, as shown in Table 1. Another interesting observation is that, as expected with the capacity theory extended in section 2.5.2 and with results of [8], generalization to new tasks is improved if more tasks are used for optimizing the learning rule.

3.4 Classification Experiments

The general problem of classification [10] is to establish a correspondence between a set of vectors and a set of classes. A classifier is a function $f : \mathcal{V} \rightarrow \mathcal{C}$, where $\mathcal{V} \in \mathbb{R}^n$ is a set of

Number of Tasks	Type of Tasks	Number of Steps	Generalization (to New Tasks)		Optimization Method	Sensibility to Initialization
			L	NL		
1	L	3	yes	no	Gradient Descent	yes
1	NL	15	yes	no		
4	L	5	yes	no		
5	4L, 1NL	100	yes	yes		
1	L	100	yes	no	Simulated Annealing	no
1	NL	1000	yes	no		
5	4L, 1NL	24000	yes	yes		

Table 1: Summary of boolean experiments. L stands for linearly separable task whereas NL stands for non-linearly separable tasks.

n-dimensional vectors we want to classify while $\mathcal{C}$ is the set of classes. Many problems may be formulated in this way. An example is the optical character recognition problem, which is to associate an image to a character. Here, using a simple neural network architecture with two input units, one hidden unit and one output unit, we will search for a synaptic learning rule able to solve two-dimensional classification problems with two classes.

Let the two classes be C_1 and C_2, and let $V_1 = \{v \in \mathbb{R}^2 | v \text{ belongs to } C_1\}$ and $V_2 = \{v \in \mathbb{R}^2 | v \text{ belongs to } C_2\}$ be the sets of vectors belonging respectively to C_1 and C_2. The task consists in learn whether each vector $v \in \mathbb{R}^2$ belongs to C_1 or C_2. To do this, we randomly select vectors $v \in \mathbb{R}^2$ belonging to C_1 or C_2. The network predicts the class C^* to which an input vector v belongs. The goal is to minimize (by modifying the connection weights) the difference between C^* and the correct class associated to a vector, for every training vector.

We performed experiments to verify the theory of capacity and generalization applied to parametric learning rules. In particular, we wanted to study the variation of the number of tasks N, the capacity h and the complexity of the tasks, over the learning rule's generalization property (ϵ). Moreover, we did these experiments using three different optimization methods, namely gradient descent, genetic algorithms and simulated annealing. Experiments were conducted in the following conditions:

- Some tasks were linearly separable (L) while others where non-linearly separable (NL).

- Each task was learned with 800 training examples and tested with 200 examples. A task was said to be successfully learned when there were no classification error over the test set.

- We used once again the network described in figure 9, with backward neurons that may provide error information to hidden connections.

- We tried two different parametric learning rules. Rule A was defined using biological a-priori knowledge to constrain the number of parameters to 7, as equation (9):

$$\begin{aligned} \Delta w(i,j) &= \theta_0 + \theta_1\, y(i) + \theta_2\, x(j) + \theta_3\, y(mod(j)) + \\ &\quad \theta_4\, y(i)\, y(mod(j)) + \theta_5\, y(i)\, x(j) + \theta_6\, y(i)\, w(i,j) \end{aligned} \tag{10}$$

  where $w(i,j)$ is the synaptic efficacy between neurons i and j, $x(j)$ is the activation potential of neuron j (postsynaptic potential), $y(i)$ is the output of neuron i (presynaptic activity), and $y(mod(j))$ is the output of a modulatory neuron influencing neuron j.

  Rule B had 16 parameters and was defined as follows:

$$\begin{aligned} \Delta w(i,j) &= \theta_0 + \theta_1\, y(i) + \theta_2\, x(j) + \theta_3\, y(mod(j)) + \theta_4\, w(i,j) + \theta_5\, y(i)\, x(j) + \\ &\quad \theta_6\, y(i)\, y(mod(j)) + \theta_7\, y(i)\, w(i,j) + \theta_8\, x(j)\, y(mod(j)) + \\ &\quad \theta_9\, x(i)\, w(i,j) + \theta_{10}\, y(mod(j))\, w(i,j) + \theta_{11}\, y(i)\, x(j)\, y(mod(j)) + \\ &\quad \theta_{12}\, y(i)\, x(j)\, w(i,j) + \theta_{13}\, y(i)\, y(mod(j))\, w(i,j) + \\ &\quad \theta_{14}\, x(j)\, y(mod(j))\, w(i,j) + \theta_{15}\, y(i)\, x(j)\, y(mod(j))\, w(i,j) \end{aligned} \tag{11}$$

- A typical experiment was conducted as follows: We chose a parametric learning rule (A or B), an optimization method (genetic algorithms, gradient descent, or simulated annealing), a number of tasks to optimize the rule (1 to 9), and a complexity for the tasks (linearly separable (L), or non-linearly separable (NL)). Then we optimized the rule for a fixed number of iterations, and finally, we tested the new rule over other tasks (i.e., we tried to learn new tasks with their 800 training patterns and evaluate performance with a test over the remaining 200).

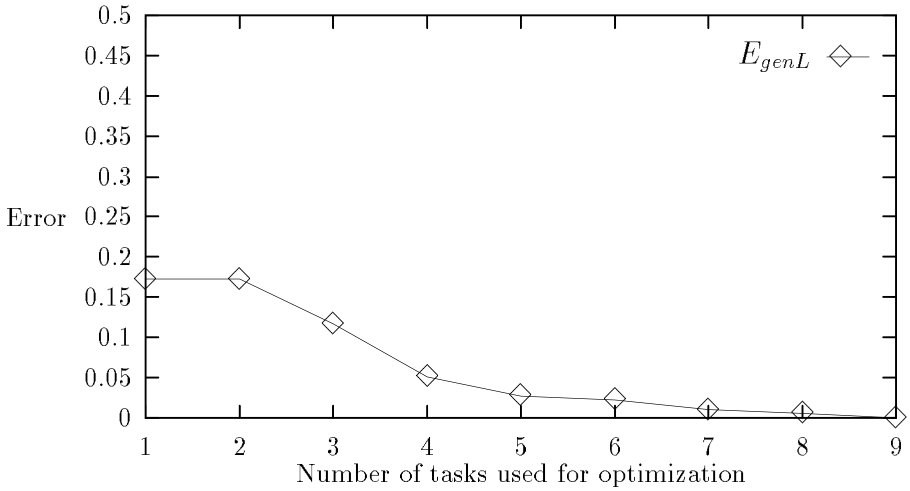


Figure 10: Evolution of generalization error (E_{genL}) with respect to the number of tasks used during optimization. In this example, we used **genetic algorithms** and a rule with **7 parameters**. Tasks were **linearly separables**.

The first experiment was to verify that the number of tasks N used for the optimization had an influence on the rule's generalization performance. Figure 10 shows that for a given and fixed optimization method and capacity h, generalization error tends to decrease when N increases, as theory predicts.

The second experiment was to verify if the type of tasks used during optimization influences the rule's generalization performance. Figure 11 illustrates the results. We can see that when the rule is optimized using linearly separable tasks, generalization error on both linearly and non linearly separable tasks stays high, whereas if we use non linearly separable tasks during rule optimization, generalization error decreases when the number of tasks increases.

In the third experiment (figure 12), we verified if the capacity of a parametric learning rule influences its generalization performance. Here, we compared rules A and B (respectively

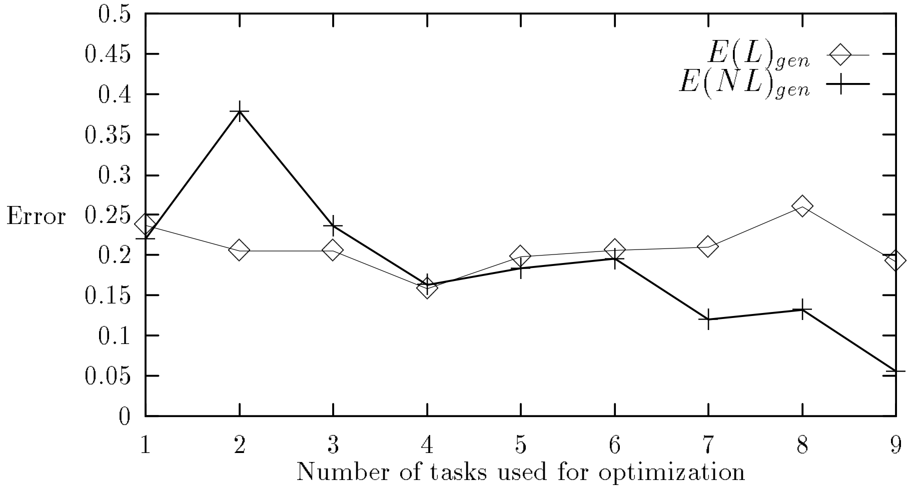


Figure 11: Evolution of generalization error with respect to the task difficulty used during optimization. Here, we used **genetic algorithms** and a rule with **7 parameters**. $E(L)_{gen}$ represents generalization error when the rule is optimized on linearly separable tasks, whereas $E(NL)_{gen}$ represents generalization error when the rule is optimized on non linearly separable tasks.

with 7 and 16 parameters). As we can see, if the number of tasks used for optimization is too small, the rule with the smallest capacity (A) is better, but the advantage tends to vanish when the number of tasks increases.

In figure 13, we compare the use of different optimization methods to find the parameters of a learning rule. We compared two methods: genetic algorithms and simulated annealing[4]. As we can see, genetic algorithms seem generally better, especially when the number of tasks used for optimization is small.

The last figure (14) shows how optimization error varies during optimization of the learning rule. At the beginning of the optimization process, training error on selected tasks is very high, but it decreases rapidly during the optimization process.

[4] Gradient descent always falled into local minima and thus have not been able to give interesting results.

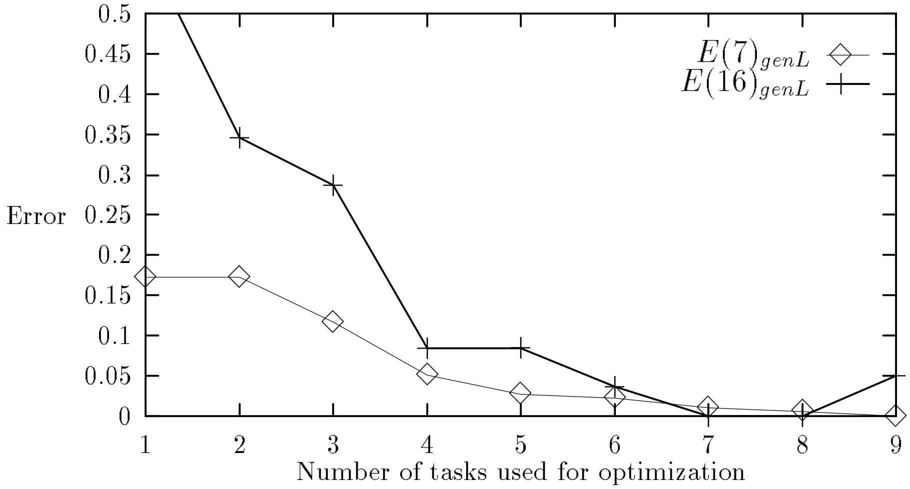


Figure 12: Evolution of generalization error with respect to capacity of the parametric learning rule. Here, we used **genetic algorithms** and tasks were **linealy separables**. $E(7)_{genL}$ is the generalization error of rule A (with 7 parameters) and $E(16)_{genL}$ is the generalization error of rule B (with 16 parameters).

4 Conclusion

This paper explores methods to optimize learning rules in neural networks. Preliminary results show that it is possible to optimize a synaptic learning rule for different tasks, while constraining the rule to be biologically plausible.

Furthermore, we have established the conceptual basis permitting to study the generalization properties of a learning rule whose parameters are trained on a certain number of tasks. To do so, we have introduced the notion of capacity of parametric learning rules. The experimental results described here qualitatively agree with learning theory applied to parametric learning rules.

The problems studied so far were quite simple, and it is important to improve the form of the learning rule and the optimization process in order to find rules that can efficiently solve more complex problems. In this section we discuss some improvements which should be considered for further research.

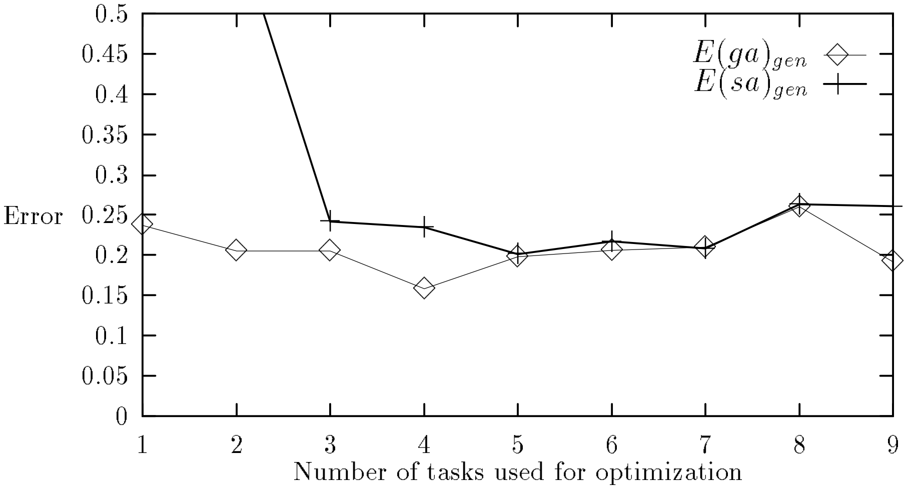


Figure 13: Evolution of generalization error with respect to the optimization method. Here, tasks were **linearly separables** and the rule had **7 parameters**. $E(ga)_{gen}$ represents generalization error with genetic algorithms, and $E(sa)_{gen}$ generalization error with simulated annealing.

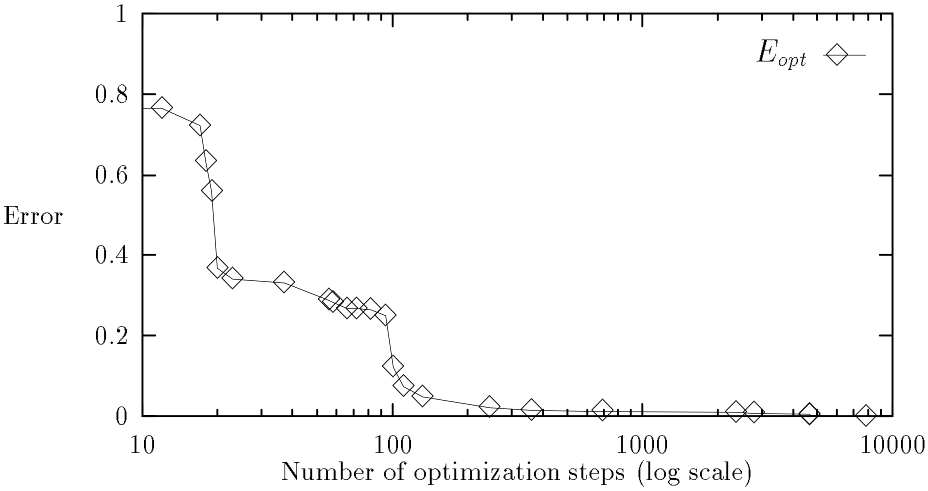


Figure 14: Evolution of optimization error with respect to the number of steps during optimization of a parametric learning rule. Here, optimization method is **simulated annealing** and the rule has **7 parameters**.

Optimization There are two immediate ways to improve the optimization process. One is to refine the cost function by adding terms reflecting the quality of network generalization. Other optimization methods such as genetic programming [21] or second-order methods should be tested for their efficiency in handling diverse and more complex learning problems. Experiments show that a rule optimized to solve a simple task cannot solve a more difficult task, while the converse is often true (given that the number of tasks used to optimize the rule is sufficient).

Form of $\Delta w()$ Optimization can yield good results only if the learning rule's constraints are soft enough to yield one or more acceptable solutions, but hard enough in order to yield good generalization performance and speed up optimization so that such solutions can be found in reasonable time. One possibility is to perform a preliminary analysis of the tasks to be learned and to "customize" the rule accordingly. Another one is to take into account known (and probably useful) biological synaptic mechanisms such as:

- Temporal processing in a synapse (different input factors influence synaptic efficacy with different delays).
- Limiting the neurons to be excitatory or inhibitory (but not both). This is unlike most existing artificial neural network models in which a neuron may have both behaviors.
- More detailed modeling of operation of the neuron, taking into account physical distance between synaptic sites and local influence of the local potential on the neural membrane on the dendritic tree.

Analysis of resulting learning rules It is important to analyze the obtained learning rules. We should systematically compare our resulting rules with other learning techniques such as back-propagation and attempt to discover the underlying reasons for differences in their behavior.

References

[1] E. R. Baum and D. Haussler, *What size network give valid generalization*, Neural Computation, 1 (1989), pp. 151–160.

[2] S. Bengio, Y. Bengio, J. Cloutier, and J. Gecsei, *Aspects théoriques de l'optimisation d'une règle d'apprentissage*, in Actes de la conférence Neuro-Nimes 1992, Nimes, France, 1992.

[3] ——, *Generalization of a parametric learning rule*, in ICANN '93: Proceedings of the International Conference on Artificial Neural Networks, Amsterdam, Nederlands, 1993.

[4] Y. Bengio and S. Bengio, *Learning a synaptic learning rule*, Tech. Rep. 751, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal, Montréal (QC) Canada, 1990.

[5] Y. Bengio, S. Bengio, and J. Cloutier, *Learning a synaptic learning rule*, in Proceedings of the International Joint Conference on Neural Networks, Seattle, USA, 1991.

[6] ——, *Learning synaptic learning rules*, in Neural Networks for Computing, Snowbird, Utah, USA, 1991.

[7] J. Byrne and W. Berry, eds., *Neural Models of Plasticity*, Addison-Wesley, 1989.

[8] D. Chalmers, *The evolution of learning: An experiment in genetic connectionism*, in Proceedings of the 1990 Connectionist Models Summer School, D. Touretzky, J. Elman, T. Sejnowski, and G. Hinton, eds., San Mateo, CA, USA, 1990, Morgan Kaufmann.

[9] L. Davis, *Mapping neural networks into classifier systems*, in Proceedings of the Third International Conference on Genetic Algorithms, San Mateo, CA, USA, 1989, Morgan Kaufmann, pp. 375–378.

[10] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, Wiley-Interscience, 1973.

[11] D. Gardner, *Synaptic diversity characterizes biological neural networks*, in Proceedings of the IEEE First International Conference on Neural Networks, vol. IV, San Diego, CA, USA, 1987, pp. 17–22.

[12] M. A. Gluck and R. F. Thompson, *Modeling the neural substrate of associative learning and memory: a computational approach*, Psychological Review, 94 (1987), p. 176.

[13] D. Goldberg, *Genetic algorithms in search, optimization, and machine learning*, Addison-Wesley, Reading, MA, USA, 1989.

[14] R. D. Hawkins, *A biologically based computational model for several simple forms of learning*, in Computational Models of Learning in Simple Neural Systems, R. D. Hawkins and G. H. Bower, eds., Academic Press, 1989, pp. 65–108.

[15] R. D. Hawkins, T. W. Abrams, T. J. Carew, and E. R. Kandel, *A cellular mechanism of classical conditioning in aplysia: Activity-dependent amplification of presynaptic facilitation*, Science, 219 (1983), pp. 400–404.

[16] D. O. Hebb, *The Organization of Behavior*, Willey, New York, NY, USA, 1949.

[17] J. Hertz, A. Kroght, and R. Palmer, *Introduction to the Theory of Neural Computation*, Addison-Wessley, 1991.

[18] G. E. Hinton, *Connectionist learning procedures*, Artificial Intelligence, 40 (1989), pp. 185–234.

[19] J. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975.

[20] S. Judd, *On the complexity of loading shallow neural network*, Journal of Complexity, 4 (1988).

[21] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, Bradford Book, MIT Press, Cambridge, MA, USA, 1992.

[22] I. P. Pavlov, *Les Réflexes Conditionels*, Alcan, Paris, 1932.

[23] 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 1: Foundations, D. E. Rumelhart and J. L. McClelland, eds., Bradford Book, MIT Press, 1986.

[24] K. S., J. Gellat, and M. P. Vecchi, *Optimization by simulated annealing*, Science, 20 (1983), pp. 671–680.

[25] V. N. Vapnik, *Estimation of Dependencies Based on Empirical Data*, Springer-Verlag, New-York, NY, USA, 1982.

[26] V. N. Vapnik and A. Y. Chervonenkis, *On the uniform convergence of relative frequencies of events to their probabilities*, Theory of Probability and its Applications, 16 (1971), pp. 264–280.

[27] H. White, *Learning in artificial neural networks: a statistical perspective*, Neural Computation, 1 (1989), pp. 425–464.

[28] D. Whitley and T. Hanson, *Optimizing neural networks using faster, more accurate genetic search*, in Proceedings of the Third International Conference on Genetic Algorithms, San Mateo, CA, USA, 1989, Morgan Kaufmann, pp. 1888–1898.