On the Search for New Learning Rules for ANNs

Samy Bengio

France Télécom CNET

LAB/RIO/TNT

22307 Lannion Cedex, FRANCE

Yoshua Bengio, Jocelyn Cloutier

Université de Montréal, Département IRO

Case Postale 6128, Succ. Centre-Ville,

Montréal, QC, Canada, H3C 3J7

e-mail: bengio@iro.umontreal.ca

Abstract

In this paper, we present a framework where a learning rule can be optimized within a parametric learning rule space. We define what we call *parametric learning rules* and present a theoretical study of their *generalization* properties when estimated from a set of learning tasks and tested over another set of tasks. We corroborate the results of this study with practical experiments.

1 Introduction

Learning mechanisms in neural networks are usually associated with changes in synaptic efficiency. In such models, synaptic learning rules control the variations of the parameters (synaptic weights) of the network. Researchers in neural networks have proposed learning rules based on mathematical principles (such as backpropagation) or biological analogy (such as Hebbian rules), but better learning rules may be needed to achieve human-like performance in many learning problems.

Chalmers proposed in [1] a method to find new learning rules using evolution mechanisms such as genetic algorithms. His method considers the learning rule as a parametric function with local inputs which is the same for all neurons in the network. He used genetic algorithms to find a learning rule for networks without hidden layers and found that among the class of rules he investigated, the *delta rule* was most often selected and performed best for linearly separable boolean problems.

Independently of Chalmers, we proposed a similar approach in [2] that generalizes this idea to networks with hidden layers (able to solve non-linearly separable problems) and with the possibility to use any standard optimization methods (such as genetic algorithms, but also gradient descent and simulated annealing). In this paper, we give theoretical principles for the design of such parametric learning rules. In section 2 we introduce the idea of parametric learning rules. Section 3 explains how the concept of *generalization* can be applied to learning rules. Finally section 4 shows how practical experiments corroborate theoretical results.

2 Parametric Learning Rules

We describe in this section the basic idea of optimizing learning rules. The principle is straightforward: we consider the learning rule as a parametric function and we optimize its parameters using an optimization algorithm. In doing so, we make the following hypothesis:

- In a large neural network, there is only a limited number of different learning rules; therefore a given rule is used in a large number of neurons.
- There is a (possibly stochastic) dependency between the synaptic modification and information available locally (in the physical neighborhood of the synapse).
- This dependency can be approximated by a parametric function $f(x_1, x_2, ...x_n; \Theta)$ where x_i are the arguments of the function and Θ is a set of parameters.

Since the space of possible learning algorithms is very large, we propose to constrain it by considering only a subset of possible parametric functions for the rule. The form of the rule may be inspired by certain known synaptic mechanisms. Thus, the only input arguments considered (the x_i above) are local to the synapse, such as the pre-synaptic and post-synaptic activities, the synaptic weight, the activity of facilitatory (or modulatory) neurons, or the concentration of a chemical agent (see figure 1).

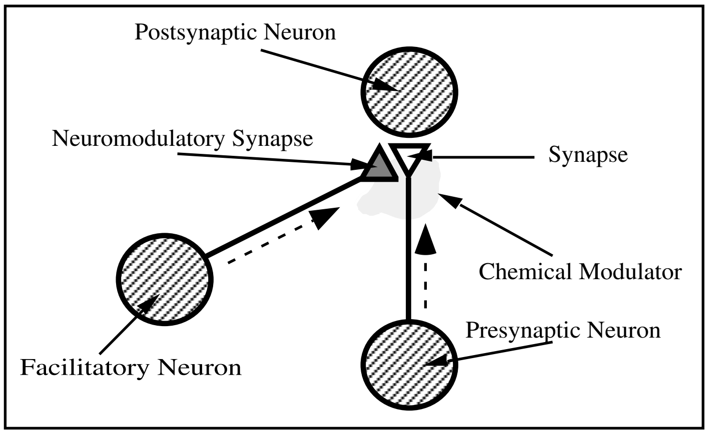

Figure 1: Elements found in the vicinity of a synapse, which can influence its efficacy.

This way of constraining the search space to be biologically plausible (if not biologically realistic) should not be perceived as an artificial constraint but rather as a way of limiting the search to a subspace in which solutions share some features with neurobiological learning mechanisms. We suppose that this will facilitate the search for new learning rules. Admittedly, our neuron models are very simplified with respect to what is currently known about the working of the brain; furthermore, many aspects of brain function are still unknown, particularly in what concerns learning and synaptic plasticity.

Let us denote by $w(i,j)$ the weight (efficacy) of a given synapse or set of synapses from neuron i to neuron j. Weight changes proceed according to the following equation:

$$\Delta w(i,j) = f(\text{local variables at synapse } i \to j; \Theta) \tag{1}$$

The synaptic modification $\Delta w(i,j)$ of the efficacy of the connection from neuron i to neuron j is obtained with the function $f(\cdot)$, using the values of local variables at the synapse, and those of a set of parameters, Θ. These parameters which are real numbers, are tuned in order to improve (optimize) the learning rule.

The idea is to try to find a set of parameters Θ corresponding to a rule allowing a network to learn to solve different kinds of problems. For this, we can use standard optimization methods (such as gradient descent, genetic algorithms, or simulated annealing). The question addressed in the next section is whether or not we can find a rule that will be able to learn tasks not used to optimize the parameters Θ.

3 Capacity of a Parametric Learning Rule

In order for a learning rule obtained through optimization to be useful, it must be successfully applicable in training networks for new tasks (i.e., tasks other than those used during optimization of the learning rule). This property of a learning rule is a form of *generalization*. We will see that this kind of generalization can be described using the same formalism used to derive the generalization property of learning systems, based on the notion of *capacity*.

3.1 Standard Notion of Capacity

The capacity of a learning system can be intuitively seen as a measure of the cardinality of the set of functions the system can learn. A quantity known as the Vapnik-Chervonenkis Dimension (VCdim) [3] is an example of such a measure. Let $F(\Theta) : X \to Y$ represent a family of functions (parameterized by Θ). X represents the space of examples and Y an error measure.

The capacity h of the learning system $F(\Theta)$ is related to generalization error ϵ and number of training examples N in the following way. For a fixed number of examples N, starting from $h = 0$ and increasing it, one finds generalization to improve (ϵ decreases) until a critical value of the capacity is reached. After this point, increasing h makes generalization deteriorate (ϵ increases). 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 [3] are obtained in the worst case, for any distribution of X.

3.2 Extension to Parametric Learning Rule

We define the *generalization error* of a parametric learning rule as expected learning error for a *new* task, that is, a task that has not been used for the optimization of the rule's parameters. The *capacity of a parametric learning rule* is a measure of the rule's complexity.

A task description z is a set of couples (i, o) such that the task requires to associate each i to a corresponding o. For instance, the boolean function **AND** is defined by the following description: $\{((0,1),0),((0,0),0),((1,0),0),((1,1),1)\}$.

Let $g(z; f(\cdot), \Theta)$ be a parametric function with an associated function $f(\cdot)$ using parameters Θ, which returns, for a given task description z, a real number representing the expected value of the error obtained after training a particular learning system with a learning rule defined by $f(\cdot)$ and Θ for the task defined by z.

For example, $f(\cdot)$ could be defined by equation (2) and Θ could be a specific vector of rule parameters. If z represents the **AND** task, then $g(z; f(\cdot), \Theta)$ is the error of a neural network trained on then **AND** task, with a learning rule described by equation (2) instantiated with the particular set of parameters Θ.

Finally, we can define $G(f(\cdot))$ as the set of all the functions $g(z; f(\cdot), \Theta)$ obtained by allowing Θ to vary within a set Ω.

In that case, we may apply the definition of capacity given by Vapnik [3] and determine the capacity h of a parametric learning rule $G(f(\cdot))$ as the maximum number of tasks z that may be solved by picking a Θ within Ω, i.e. a $g(z; f(\cdot), \Theta)$ within $G(f(\cdot))$ (with an error smaller than a predetermined value chosen to maximize h).

We can draw several conclusions from this extension. For example, it becomes clear that the expected error of a learning rule over new tasks (ϵ) should decrease when increasing the number of tasks (N) used for learning the parameter set 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.

4 Experiments

We performed experiments to verify if it was possible to find new learning rules that were able to solve difficult problems. We also wanted to verify the theory of capacity and generalization applied to parametric learning rules. In particular, we wanted to study the relation between the number of training tasks N, the learning rule's capacity h, the complexity of the tasks, and the learning rule's generalization error (ϵ).

4.1 Optimization Methods

We tried three optimization methods, namely genetic algorithms [4], simulated annealing [5], and genetic programming [6][1] . For the first two, we had to specify the exact form of the learning rule and optimization was done over a parameter space, while for the genetic programming experiments, we just had to select the appropriate local variables which could influence the synaptic change and the basic operators that could be used. This latter approach enables the search to operate on a larger space: when one specifies the exact form of the learning rule, one could neglect a good solution just because its form has not been expected. On the other hand, the capacity of such systems may be higher (and in fact more difficult to estimate), which is why we used a bound on the size of the rules.

For the first two optimization methods, we tried three different parametric learn-

[1] Gradient descent was also tested but there seems to be too many local minima in the space of learning rules which explains why we were not able to get any good solution with this local method.

ing rules. The first rule was defined using biological a-priori knowledge to constrain the number of parameters to 7. We can find for instance in the following rule a Hebbian mechanism:

$$\begin{aligned} \Delta w(i,j) &= \theta_0 + \theta_1\, y(i) + \theta_2\, x(j) + \theta_3\, y(mod(j)) + \\ &\quad \theta_4\, y(i)\, y(mod(j)) + \theta_5\, y(i)\, x(j) + \theta_6\, y(i)\, w(i,j) \end{aligned} \tag{2}$$

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

The second rule had 16 parameters and was defined as follows:

$$\Delta w(i,j) = \sum_{s \in \wp(V)} \left(\theta_s \prod_{v \in s} v \right) \tag{3}$$

where $V = \{x(j), y(i), y(mod(j)), w(i,j)\}$ and $\wp(V)$ is the set of all subsets of V.

The third rule is the same as the first one with an additional parameter (θ_7) which modulates a local version of backpropagation (i.e. if all the other parameters are set to 0 and θ_7 is set to 1, the rule is the same as backpropagation for the special kind of networks described below and in figure 2). This parametric rule has been added in order to see if one could find the backpropagation algorithm by optimization starting from a randomly chosen parametric learning rule. The rule is as follows:

$$\begin{aligned} \Delta w(i,j) &= \theta_0 + \theta_1\, y(i) + \theta_2\, x(j) + \theta_3\, y(mod(j)) + \theta_4\, y(i)\, y(mod(j)) + \\ &\quad \theta_5\, y(i)\, x(j) + \theta_6\, y(i)\, w(i,j) + \theta_7\, y(i)\, y(mod(j))\, f'(x(j)) \end{aligned} \tag{4}$$

where $f'(x(j))$ is the derivative of the neuron activation function.

4.2 Tasks

The tasks to solve were two-dimensional classification problems. There were 20 different tasks. Some were linearly separable (10) while others where non-linearly separable (10).

Each task was created with a training set of size 800 and a test set of size 200. A task was said to be successfully learned when there were no classification error over the test set.

We used a fully connected neural network[2] with two input units, one hidden unit and one output unit. Furthermore, we added a backward path of modulator neurons, which could provide a measure of the error to each unit (see figure 2).

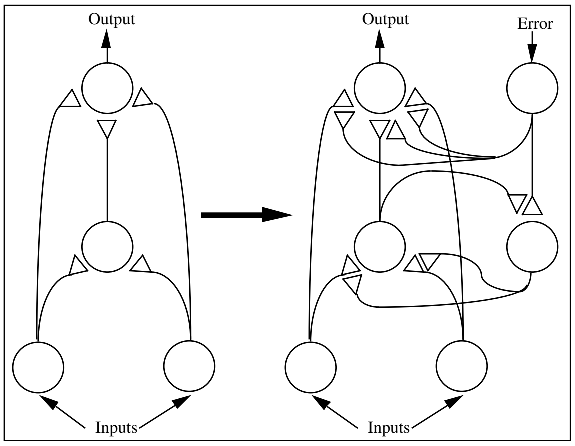

Figure 2: Simple neural network architecture with backward path.

4.3 Methodology

A typical experiment was conducted as follows:

- First we chose an optimization method (genetic algorithms, simulated annealing, or genetic programming) and a set of *training tasks* (from 1 to 9 different tasks, linearly as well as non-linearly separable).
- Then we optimized the rule for a fixed number of iterations.
- Finally, we tested the new rule over tasks different from the training tasks.

[2]Where input units also connect to output units.

To learn a task, the weights of the neural network are first initialized to random values and then they are updated after the presentation of each training example, according to the current learning rule. The generalization error for the current task is then estimated with the 200 test examples.

4.4 Results

Figure 3 shows for each type of learning rule the evolution of the best generalization error (ϵ) found using all optimization methods, with respect to the number of tasks (N). As it can be seen, ϵ decreases as N increase, as theory predicts. We can also see that the learning rule which has more capacity has an ϵ higher when N is too small, which is again what theory predicts.

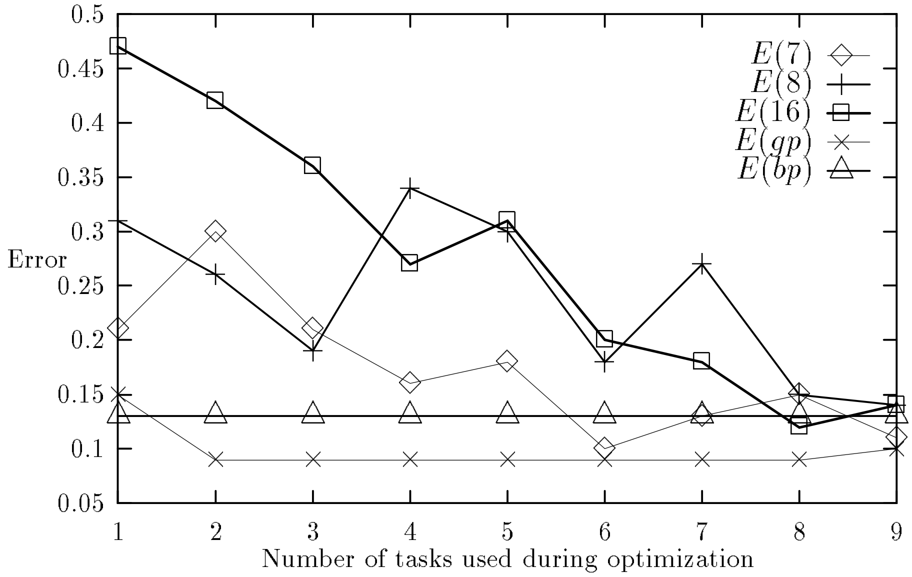

Figure 3: Evolution of generalization error ϵ with respect to the number of tasks N used during optimization. $E(x)$ represents the error made using the rule x: 7 is the rule using 7 parameters, 8 is the rule using 8 parameters, 16 is the rule using 16 parameters, gp is the rule found by genetic programming, and bp is simply the backpropagation rule.

In table 1, we give for each type of learning rule the mean generalization error we found (over all optimization methods and all experiments for the same rule), as well as its standard deviation. We also give for each rule the best set of parameters found. For the sake of comparison, we also give the generalization error of the backpropagation learning rule, applied to the same set of tasks. In table 2, we give

Rule	Error ($\mu \pm \sigma$)	Best Rule
7 parameters: eq. (2)	0.23 ± 0.12	$\Theta = \{-0.036, 0.044, -0.076, 0.027, -0.0022, 0.066, 0.0076\}$
16 parameters: eq. (3)	0.33 ± 0.14	$\Theta = \{0.0016, 0.0005, -0.007, 0.014, -0.019, 0.022, -0.0099, -0.02, -0.0003, 0.0053, 0.0099, -0.0053, -0.019, 0.0085, -0.0053, 0.0041\}$
8 parameters: eq. (4)	0.24 ± 0.07	$\Theta = \{-0.099, 0.032, -0.11, -0.11, -0.0058, 0.056, 0.007, 0.18\}$
genetic programming	0.10 ± 0.02	$\Delta w(i,j) = y(i) \cdot y(mod(j)) \cdot f'(x(j))^3$
backpropagation	0.13 ± 0.0	$\Delta w(i,j) = y(i) \cdot y(mod(j)) \cdot f'(x(j))$

Table 1: Comparison of the learning rules found. μ is the mean error, σ is the error standard deviation.

Method	Error ($\mu \pm \sigma$)
simulated annealing	0.30 ± 0.17
genetic algorithm	0.25 ± 0.11
genetic programming	0.10 ± 0.02

Table 2: Comparison of the optimization methods. μ is the mean error, σ is the error standard deviation.

for each optimization method the mean generalization error (over all rule forms and all experiments for the same optimization method), as well as its standard deviation.

We can see in the last two tables that genetic programming is the best method and it finds a rule that performs better than backpropagation for this set of tasks. The only difference with backpropagation is the exponent of the activation function derivative. As the overall sign of this term is conserved, and since the derivative is a real number between 0 and 1, the effect of the derivative over the weight change is thus stronger than in backpropagation, but in a 2-layer net, the resulting weight change is guaranteed to be downhill in the error surface.

5 Conclusion

In this article we have established a conceptual framework 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.

Experiments show that it is possible to discover through optimization learning rules capable of solving a variety of simple problems. Using gradient descent, genetic algorithms and simulated annealing, we had already found learning rules for classical conditioning problems, classification problems, and boolean problems (see [7]). Moreover, the experimental results described in section 4 qualitatively agree with learning theory applied to parametric learning rules. Of course it has to be shown yet that the procedure is applicable to more complex tasks, which will probably require more complex learning rules and yield a more difficult optimization problem. Moreover, we need to take into account the computational time needed to find seach learning rules. For example, in figure 3, each experiment required on the order of one hour of CPU time on a Sparc Station 2.

References

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

[2] Y. Bengio and S. Bengio, *Learning a synaptic learning rule*, Tech. Rep. 751, D partement d'Informatique et de Recherche Op rationnelle, Universit de Montr al, Montreal (QC) Canada, 1990.

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

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

[5] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, *Optimization by simulated annealing*, Science, 220 (1983), pp. 671–680.

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

[7] S. Bengio, Y. Bengio, J. Cloutier, and J. Gecsei, *On the optimization of a synaptic learning rule*, in Conference on Optimality in Biological and Artificial Networks, Dallas, USA, 1992.