A Global Optimization Method for Neural Network Training *

YI SHANG and BENJAMIN W. WAH
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
Urbana, IL 61801
{shang, wah}@manip.crhc.uiuc.edu

ABSTRACT

In this paper, we present a new supervised learning method called NOVEL (Nonlinear Optimization Via External Lead) for training feed-forward neural networks. In general, such learning can be considered as a nonlinear global optimization problem in which the goal is to minimize the nonlinear error function that spans the space of weights. NOVEL is a trajectory-based nonlinear optimization method that combines global and local searches to find good local minima. It relies on an external force to pull a search out of a local minimum in its global search and employs local descents to locate local minima in its local search. The result is an efficient search method that identifies good basins without spending a lot of time in them. We have shown improved training and testing results for five neural-network benchmark problems as compared to those of existing minimization and neural-network learning algorithms. For the two-spiral problem, NOVEL has found a design using only four hidden units and 25 weights. (The best known design requires nine hidden units and 75 weights.) In short, NOVEL represents a significant advance in the state-of-the-art in supervised learning of neural networks and general optimization of continuous nonlinear functions.

1. Introduction

In this paper, we present a new method for supervised learning of feed-forward networks. Supervised learning involves finding a mapping function ϕ from a given set of training patterns. This is done by adjusting weights, W, on links when the topology and the activation function are fixed. In other words, given a set of training patterns of input-output pairs $\{(I_1, D_1), (I_2, D_2), \cdots, (I_m, D_m)\}$ and an error function $\epsilon(W, I, D)$, learning strives to minimize learning error $E(W)$:

$$\min_W E(W) = \min_W \sum_{i=1}^{m} \epsilon(W, I_i, D_i). \tag{1}$$

One popular error function is the squared-error function in which $\epsilon(W, I_i, D_i) = (\phi(I_i, W) - D_i)^2$. The quality of a learned network is measured by its error on a given set of training patterns and its (generalization) error on a given set of test patterns.

In the form represented in (1), supervised learning can be considered as an unconstrained nonlinear minimization problem in which the objective function is defined by (1), and the search space is defined by the space of the weights.[1] Unfortunately, the terrain modeled by the error function in its weight space can be extremely rugged and has many local minima when the activation function is nonlinear.

Many learning algorithms find their roots in function-minimization algorithms that can be classified into local minimization and global minimization. Local minimization algorithms, such as gradient descent

* This research was supported in part by National Science Foundation Grant MIP 92-18715 and in part by Joint Services Electronics Program Contract N00014-90-J-1270.

Portion of this paper has appeared in a special issue on artificial neural networks in *IEEE Computer*, March 1996.

[1] Without loss of generality, we consider only minimization problems in this paper.

and Newton's methods, are fast but usually converge to local minima. In contrast, global minimization algorithms have strategies to help escape from local minima.

Many local minimization methods using first- and second-order information have been applied to learning of feed-forward neural networks [2]. Examples include back-propagation (BP), conjugate-gradient and quasi-Newton's methods. Local minimization algorithms have difficulties when the surface is flat (gradient close to zero), or when gradients can be in a large range, or when the surface is very rugged. When gradients can vary greatly, the search may progress too slowly when the gradient is small and may over-shoot when the gradient is large. When the terrain is rugged, a local search from a randomly chosen starting point will likely converge to a local minimum close to the initial point and a solution far worse than the global minimum.

To overcome the deficiencies in local-search methods, global minimization methods have been developed. Global minimization methods rely on local search to determine local minima, and focus on bringing the search out of a local minimum once it gets there. Up to today, general nonlinear minimization algorithms can at best find local minima of a multi-modal function. Only in cases with very restrictive assumptions, such as Lipschitz condition, algorithms with guaranteed accuracy can be constructed.

Global minimization algorithms can be classified into deterministic and probabilistic. Deterministic methods include covering, trajectory and penalty methods. These methods do not work well when the problem has more than a few variables. Probabilistic methods include clustering methods, random search methods and methods based on stochastic models. Some of these algorithms, e.g. simulated annealing and genetic algorithms, have been found to work well for some neural-network learning problems. However, all of them are weak in either the local or the global search. For instance, gradient information useful in local search is not used in simulated annealing and evolutionary algorithms. On the other hand, gradient descent algorithms with multi-starts work well in local search but are weak in global search.

We describe in Section 2 NOVEL, a new global minimization method for neural network learning. We demonstrate NOVEL's superior performance on neural network learning by finding networks that are smaller and yet generalize better than networks found by existing algorithms. This is done in Section 3 in which we compare NOVEL with some of the best global minimization algorithms for solving the two-spiral problem, and in Section 4 in which we apply NOVEL to solve four other benchmark problems.

2. NOVEL: A New Global Optimization Method

NOVEL is a trajectory-based method that employs local descents to locate local minima and relies on an *external* force to pull the search out of local minima. It has three features: exploring the solution space, locating promising regions, and finding local minima. In exploring the solution space, the search is guided by a continuous terrain-independent trace function that does not get trapped by local minima. In locating promising regions, NOVEL uses local gradient to attract the search to a local minimum. Finally, NOVEL selects one initial point for each promising local region and uses them as initial points for a descent algorithm to find local minima.

NOVEL is efficient in the sense that it tries to first identify good starting points before applying a local search. This avoids repeatedly determining unpromising local minima as in multi-start algorithms, and avoids computationally expensive descent algorithms from random starting points. It is also efficient because it provides a continuous means of going from one local region to another.

NOVEL has two phases: global-search phase and local-search phase (see Figure 1). The goal of the global-search phase is to identify regions containing local minima, whereas the goal of the local-search phase is to actually find the local minima.

In the global-search phase, there are a number of bootstrapping stages. (Three stages are shown in Figure 1.) The dynamics in each stage is represented by an ordinary differential equation that performs local descent and global exploration driven by a trace function. A stage is coupled to the next stage by feeding its output trajectory as the trace function of the next stage, with a predefined trace function as the input of the first stage. Interpolations are performed when the input trace supplied by the previous stage is not a continuous function.

In the local-search phase, a traditional descent method, such as gradient descent, conjugate gradient

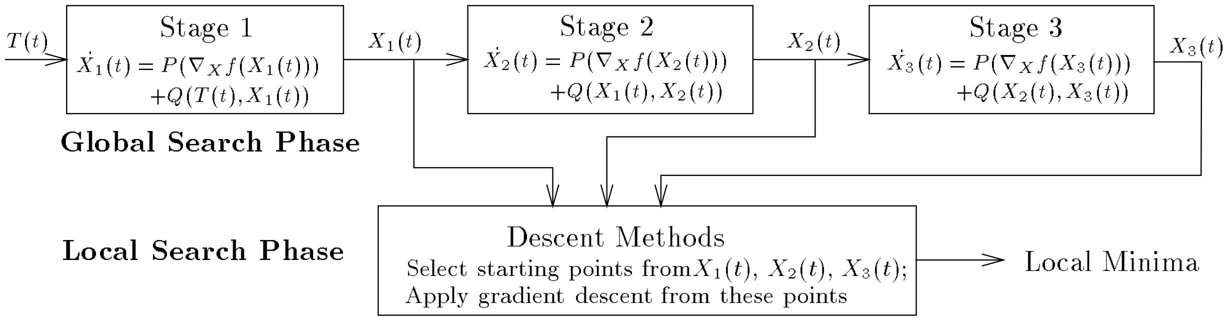
$T(t)$
Stage 1
$\dot{X}_1(t) = P(\nabla_X f(X_1(t)))$
$+Q(T(t), X_1(t))$
$X_1(t)$
Stage 2
$\dot{X}_2(t) = P(\nabla_X f(X_2(t)))$
$+Q(X_1(t), X_2(t))$
$X_2(t)$
Stage 3
$\dot{X}_3(t) = P(\nabla_X f(X_3(t)))$
$+Q(X_2(t), X_3(t))$
$X_3(t)$
Global Search Phase
Local Search Phase
Descent Methods
Select starting points from $X_1(t)$, $X_2(t)$, $X_3(t)$;
Apply gradient descent from these points
Local Minima

Fig. 1: Framework of the NOVEL method. (See Section 2 for an explanation of the equations.)

or Quasi-Newton's method, is applied to find local minima. Initial points for the local search are selected based on trajectories output by the global-search phase. In our experiments, the best solutions in periodic time intervals were used as initial points.

Assume $f(X)$ with gradient $\nabla_X f(X)$ is to be minimized, where $X = (x_1, x_2, \cdots, x_n)$ are variables. There may be simple bounds like $x_i \in [a_i, b_i]$, where a_i, b_i, $i = 1, \cdots, n$, are real numbers.

Each stage in the global-search phase of NOVEL defines a *trajectory* $X(t) = (x_1(t), \cdots, x_n(t))$ that is governed by the following ordinary differential equation:

$$\dot{X}(t) = P(\nabla_X f(X(t))) + Q(T(t), X(t)) \tag{2}$$

where t is the autonomous variable; T, the trace function, is a function of t; and P and Q are general nonlinear functions. This equation specifies a trajectory through variable space X. It has two components, $P(\nabla_X f(X))$ that enables the gradient to attract the trajectory to a local minimum, and $Q(T(t))$ that allows the trace function to lead the trajectory out of the local minimum.

P and Q can have various forms. A simple form we have used in our experiments is

$$\dot{X}(t) = -\mu_g \, \nabla_X f(X(t)) - \mu_t \, (X(t) - T(t)) \tag{3}$$

where μ_g and μ_t are coefficients.

To find global minima efficiently without any knowledge on the terrain, we should design a trace function that traverses the search space uniformly and search the space from coarse to fine. After substantial experimentation, we have designed a non-periodic, analytical trace function as follows:

$$T_i(t) = \rho \, sin \left[2\pi \left(\frac{t}{2} \right)^{0.95 - 0.45(i-1)/n} + \frac{2\pi(i-1)}{n} \right] \tag{4}$$

where i represents the i'th dimension, ρ is a coefficient specifying the range, and n is the number of dimensions. Note that the trace function has different phases in different dimensions, and that the function traverses the space $[-1, 1]^n$ when $\rho = 1$.

Given (3), various numerical approaches can be applied to evaluate the ordinary differential equation. We have used both a differential-equation solver and a difference-equation solver.

A differential-equation solver solves (3) as an ordinary differential equation. The software package we have used is the Livermore Solver for Ordinary Differential Equations [4] (*LSODE*) that solves (3) to within a prescribed degree of accuracy. However, it is usually computationally expensive, especially when the number of variables is large. Further, it requires the true gradient, meaning that neural-network learning can only be done in an epoch-wise mode, not in a pattern-wise mode.

The second approach is to discretize (3) and use a finite difference-equation solver. The difference equation derived from (3) is as follows.

$$X(t + \delta t) = X(t) + \delta t[-\mu_g \nabla_X f(X(t)) - \mu_t (X(t) - T(t))] \tag{5}$$

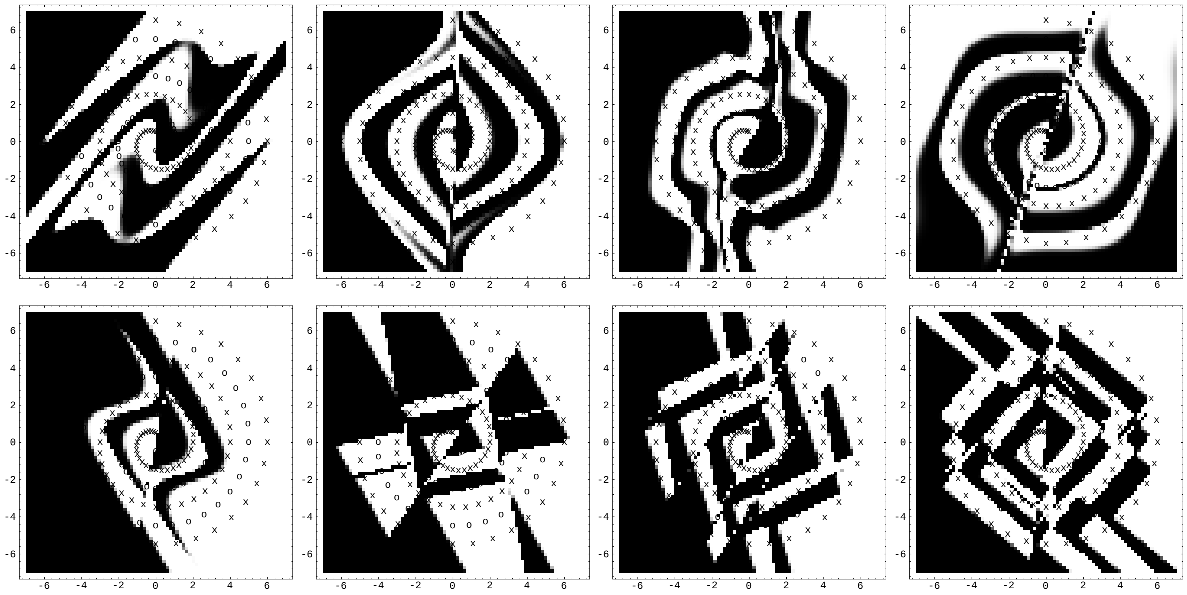

Fig. 2: 2-D classification graphs for the two-spiral problem by 3 (first column), 4 (second column), 5 (third column) and 6 (fourth column) hidden-unit neural networks trained by NOVEL (upper row) and SIMANN (lower row). Parameters for NOVEL are $\mu_g = 1, \mu_t = 20$, and $\alpha = 100$. Parameters for SIMANN are $RT = 0.99, NT = 5n$, and the search range is [-2.0, 2.0]. The dots and circles represent the training patterns.

where δt is the step size that specifies the distance advanced in each step. A large δt causes a large stride of variable modification, possibly resulting in oscillations. On the other hand, a small δt means a longer computation time for traversing the same distance. This approach is fast, allowing NOVEL to be applied in both pattern-wise and epoch-wise training, especially when the problem is large. However, solutions may be slightly worse than those found by LSODE.

3. Two-Spiral Problem

The two-spiral problem is a difficult classification problem. Published results on feed-forward networks include those trained by BP, CASCOR [3], and projection pursuit [5]. The smallest network is believed to have nine hidden units trained by CASCOR.

In our experiments, we have used feed-forward networks with shortcuts. Each hidden unit is ordered and labeled by an index, and has incoming connections from all input nodes and from all hidden units with smaller indexes. This is the same pyramid structure that CASCOR constructs. The activation function is an asymmetric sigmoidal function $f(x) = 1/(1 + e^{-\alpha x})$, where α is the sigmoid gain. We have fixed the search range in NOVEL as $[-1, 1]$ in each dimension and have varied α from 1 to 150. The error function $E(w)$ defined in (1) is the total sum of squared error (TSSE) over all training patterns. After some experiments on four and five hidden unit networks, we found that $\alpha = 100$, $\mu_g = 1$ and $\mu_t = 20$ work well.

Our trace always starts from the origin of the variable space. One *time unit* represents a change from $t = \tau$ to $t = \tau + 1$. We carried out our experiments on Sun SparcStation 20/71 workstations and executed all three stages in the global-search phase in each time unit. NOVEL successfully trained five hidden-unit networks in less than 100 time units. Training four hidden-unit networks is more difficult. After running NOVEL for 19 hours, we found a solution with TSSE of 4.0. Using this solution as a new starting point of NOVEL, we found a solution with TSSE of 2.1 and 99% correct after 15 hours. Using this solution as a new starting point resulted in a solution that is 100% correct after 10 hours. Figure 2 shows how the best four hidden-unit network found classifies the 2-D space.

We compare the performance of NOVEL with that of simulated annealing (SIMANN from netlib [1]), two evolutionary algorithms (GENOCOP by Michalewicz [6] and LICE by Sprave [8]), cascade correlation

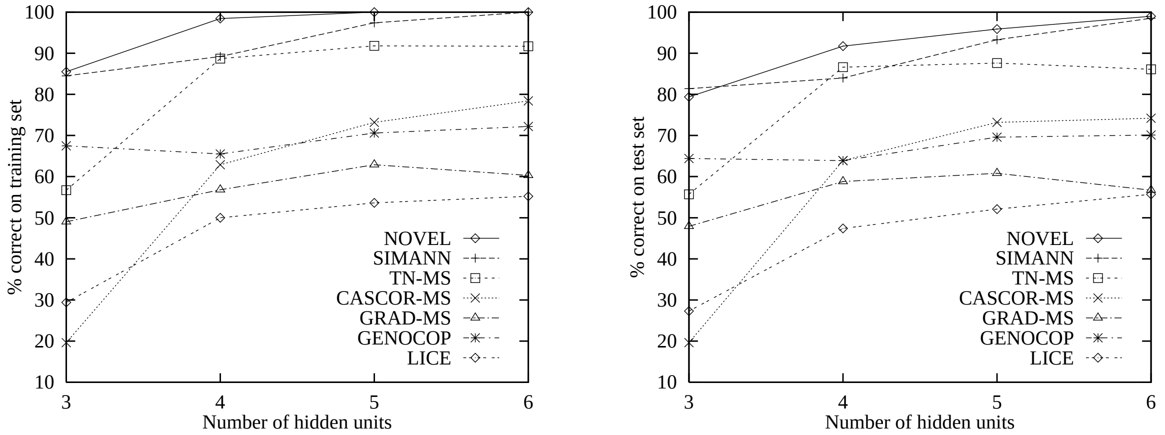

Fig. 3: Training and test errors of the best designs obtained by the various algorithms for solving the two-spiral problem. There are 18, 25, 33, and 42 weights in the neural network with, respectively, 3, 4, 5, and 6 hidden units.

with multi-starts (*CASCOR-MS*, based on Fahlman's CASCOR program [3] from random initial points and random initial weights), gradient descent with multi-starts (*GRAD-MS*, based on *LSODE* to do gradient descents), and truncated Newton's method with multi-starts (TN-MS, based on the truncated Newton's method obtained from netlib). To allow a fair comparison, we ran these methods and NOVEL for the same amount of time using the same network structure. For all algorithms, we tuned their parameters and report the best results we have obtained.

Figure 3 summarizes the results of the best solutions found by each of these algorithms when run under 20 hours of CPU time. The graphs show that NOVEL has the best training and test results for the neural networks found, followed by SIMANN, TN-MS, *CASCOR-MS*, and the two evolutionary algorithms. The two evolutionary algorithms do not work well because genetic operators like mutation and cross-over do not utilize local gradient information in deciding where to search. Note that further improvement is difficult after the time allowed. Figure 2 summarizes the best solutions obtained by NOVEL and SIMANN.

The experimental results show that an algorithm's performance depends on the complexity of the error function. When the error function is simple, as in optimizing the weights of a three hidden-unit network, NOVEL as well as other algorithms can find good minima. When the error function is complex and the number of global minima is few, NOVEL performs much better than other algorithms.

Differential-equation solver is computationally expensive. To improve the computational overhead, we can used a difference-equation solver instead. In contrast to a differential-equation solver, a difference-equation solver can employ pattern-wise training and update the weights after every training pattern has been presented. This results in an order of magnitude faster, but slightly worse solution quality. For the two-spiral problem, the fraction of correctly identified patterns are 92.8%, 96.9% and 100% for 4, 5 and 6 hidden-unit networks, respectively.

4. Experimental Results on Other Benchmarks

In this section, we show our results on applying NOVEL on four benchmark problems obtained from ftp.cs.cmu.edu in directory /afs/cs/project/connect/bench — sonar, vowel-recognition, 10-parity, and NetTalk problems. They represent classification problems of different complexity and characteristics.

The network topologies used in these experiments are layered feed-forward networks without shortcuts (to be consistent with what others have used), with the goal of minimizing the total sum of squared errors. Other setups are similar to those described for the two-spiral problem.

For these problems, we have applied NOVEL with a difference-equation solver, TN-MS, SIMANN, and BP. As found by Dixon [2], TN runs much faster than epoch-wise BP and achieves comparable solutions. SIMANN is one order of magnitude slower than TN-MS and NOVEL with results of similar quality. For these reasons, we only report the results for TN-MS and NOVEL with a difference-equation solver in Table 1.

Table: 1: Comparison of the best results obtained by *NOVEL* and truncated Newton's algorithm with multi-starts (*TN-MS*) for solving four benchmark problems, where the parameters in one method that obtains the best result may be different from those of another method. Results in bold font are better than or equal to results obtained by *TN-MS*.

Problems	# of H.U.	# of Wts.	*TN-MS*			*NOVEL*			*TN-MS + NOVEL*			CPU time limits
			Correct %		# of	Correct %		# time	Correct %		# time	
			training	test	restarts	training	test	units	training	test	units	
Sonar	2	125	98.1	90.4	454	**98.1**	**94.2**	191	**98.1**	**92.3**	226	1000 sec
	3	187	100	91.3	485	**100**	**92.3**	291	**100**	**92.3**	315	2000 sec
Vowel	2	55	72.2	50.9	298	**72.5**	49.1	131	**73.5**	50.6	203	2 hours
	4	99	80.7	56.5	152	**82.6**	**57.8**	41	**81.2**	**57.1**	168	2 hours
10-parity	5	61	97.2	—	148	**98.9**	—	51	**97.2**	—	49	2000 sec
	6	73	97.6	—	108	**99.8**	—	62	**97.6**	—	44	3000 sec
NetTalk			*Pattern-wise BP*			*NOVEL*			*BP + NOVEL*			
	15	3,476	86.3	70.5	13	**87.4**	**72.7**	11	**89.0**	70.4	11	3 hours
	30	6,926	92.9	73.1	9	**93.2**	72.5	4	**94.7**	72.3	7	4 hours

NOVEL searched in the range $[-1, 1]$, always started from the origin, and used TN in its local-search phase. TN-MS was run multiple times from random initial points in various search ranges. Both algorithms were run for the same amount of CPU time.

Table 1 shows the best solutions of both algorithms that achieve the highest percentage of being correct on test patterns of the sonar problem, given that both have achieved 100% correct in training. Our results show that NOVEL is 1%-4% better in test accuracy.

We attribute NOVEL's superiority in finding better solutions to its global-search stage. Since the function searched is rugged, it is important to identify good basins before committing expensive local descents into them. However, multi-start algorithm can provide good starting points for NOVEL. Table 1 shows the improved performance of applying NOVEL starting from the best solution found by TN-MS.

On the vowel-recognition problem, NOVEL performs better than TN-MS in training but slightly worse in testing for two hidden-unit case. On the 10-parity problem, NOVEL has smaller errors in learning than both TN-MS and TN-MS+TM.

On NetTalk, the number of weights and training patterns are very large, leading us to use pattern-wise learning when applying BP (as in the original experiments by Sejnowski and Rosenberg [7]). For a similar reason, we can only use in NOVEL pattern-wise mode in the global-search phase and pattern-wise BP in the local-search phase. Even so, very few time units could be simulated. To find better designs, we took the best designs obtained by pattern-wise BP and applied NOVEL for 10 time units. Table 1 shows improved learning but slight worse testing results.

References

[1] A. Corana, M. Marchesi, C. Martini, and S. Ridella, "Minimizing multi-modal functions of continuous variables with the simulated annealing algorithm," *ACM Transactions on Mathematical Software*, vol. 13, no. 3, pp. 262–280, 1987.

[2] L. C. W. Dixon, "Neural networks and unconstrained optimization," in *Algorithms for Continuous Optimization: The State of the Art*, (E. Spedicato, ed.), pp. 513–530, Netherlands: Kluwer Academic Publishers, 1994.

[3] S. E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture," in *Advances in Neural Information Processing Systems 2*, (D. S. Touretzky, ed.), pp. 524–532, San Mateo, CA: Morgan Kaufmann, 1990.

[4] A. C. Hindmarsh, "ODEPACK, a systematized collection of ODE solvers," in *Scientific Computing*, (R. S. Stepleman, ed.), pp. 55–64, Amsterdam: North Holland, 1983.

[5] J. Hwang, S. Lay, M. Maechler, R. D. Martin, and J. Schimert, "Regression modeling in back-propagation and projection pursuit learning," *IEEE Transactions on Neural Networks*, vol. 5, pp. 1–24, May 1994.

[6] Z. Michalewicz, *Genetic Algorithms + Data Structure = Evolution Programs*. Springer-Verlag, 1994.

[7] T. J. Sejnowski and C. R. Rosenberg, "Parallel networks that learn to pronounce English text," *Complex Systems*, vol. 1, pp. 145–168, Feb. 1987.

[8] J. Sprave, "Linear neighborhood evolution stategies," in *Proc. of the third Annual Conf. on Evolutionary Programming*, (San Diego, CA), World Scientific, 1994.