Violation-Guided Learning for Constrained Formulations in Neural-Network Time-Series Predictions[*]

Benjamin W. Wah and Minglun Qian
Department of Electrical and Computer Engineering
and the Coordinated Science Laboratory
University of Illinois, Urbana-Champaign
Urbana, IL 61801, USA
E-mail: {wah, m-qian}@manip.crhc.uiuc.edu

Abstract

Time-series predictions by artificial neural networks (ANNs) are traditionally formulated as unconstrained optimization problems. As an unconstrained formulation provides little guidance on search directions when a search gets stuck in a poor local minimum, we have proposed recently to use a constrained formulation in order to use constraint violations to provide additional guidance. In this paper, we formulate ANN learning with cross-validations for time-series predictions as a non-differentiable nonlinear constrained optimization problem. Based on our theory of Lagrange multipliers for discrete constrained optimization, we propose an efficient learning algorithm, called *violation guided back-propagation* (VGBP), that computes an approximate gradient using back-propagation (BP), that introduces annealing to avoid blind acceptance of trial points, and that applies a relax-and-tighten (R&T) strategy to achieve faster convergence. Extensive experimental results on well-known benchmarks, when compared to previous work, show one to two orders-of-magnitude improvement in prediction quality, while using less weights.

1 Introduction

We study in this paper new formulations and learning algorithms for predicting stationary time-series using artificial neural networks (ANNs).

ANNs for modeling time-series generally have special structures that store temporal information either explicitly using time-delayed structures or implicitly using feedback structures. Examples of the first class include time-delayed neural networks (TDNN) and FIR neural networks (FIR-NN), whereas examples of the latter include recurrent neural networks (RNN) [Haykin, 1994]. In the ANNs studied in this paper, we use a hybrid architecture with a recurrent structure, whose neurons are connected by FIR filters, instead of links with constant weights. We believe that such an architecture is more powerful for modeling unknown temporal information.

Time-series predictions using ANNs have traditionally been formulated as an unconstrained optimization problem of minimizing the mean squared errors (MSE):

$$\min_w \ \mathcal{E}_{av}(t_0, t_1) = \sum_{t=t_0}^{t_1} \sum_{i=1}^{N_o} (o_i(t) - d_i(t))^2, \qquad (1)$$

where N_o is the number of output nodes in the ANN, o_t and d_t are, respectively, the actual and desired outputs of the ANN at time t, w is a vector of all the weights, and the training data consist of patterns observed at $t = t_0, \cdots, t_1$. Extensive research has been conducted in the past on designing ANNs with a small number of weights that can generalize well. However, such learning algorithms have limited success because little guidance is provided in an unconstrained formulation when a search is stuck in a local minimum in its weight space. In this case, the sum of squared errors in (1) does not indicate which patterns are violated and the best direction for the trajectory to move.

To address the issue on lack of guidance, we have proposed recently a constrained formulation [Wah and Qian, 2000] on ANN learning that accounts for the error on each training pattern in a constraint:

$$\min_w \ \mathcal{E}_{av}(t_0, t_1) = \sum_{t=t_0}^{t_1} \sum_{i=1}^{N_o} \phi((o_i(t) - d_i(t))^2 - \tau) \qquad (2)$$
$$\text{such that } \forall\, i, t,\ \ h_i(t) = (o_i(t) - d_i(t))^2 \le \tau,$$

where $h_i(t) \le \tau$ prescribes that the error of the i^{th} output unit on the t^{th} training pattern be less than τ, and $\phi(x) = \max\{0, x\}$. A constrained formulation is beneficial in difficult training scenarios because violated constraints provide additional guidance during a search, leading a trajectory towards a direction that reduces overall constraint violations.

In time-series prediction, *cross validations* are often used to prevent data from over-fitting. There are two types of cross-validation errors: *single-step validation errors* that measure output errors when external inputs to an ANN are true observed data, and *iterative validation errors* that measure output errors when external inputs to an ANN are predicted outputs from previous iterations.

[*]Research supported by National Aeronautics and Space Administration Contract NAS 2-1230.

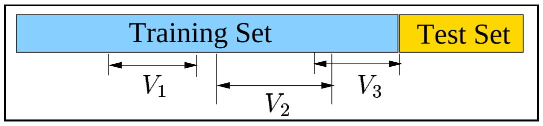


Figure 1: Multiple validation sets in a training set. V_1,V_2 and V_3 are three validation sets. The test test is used for testing the ANN after learning is completed.

In traditional learning with cross validations, a part of training patterns is reserved *a priori* in a validation set and not used in learning, and the single objective in learning is to minimize validation errors. Hence, errors measured in validation will not be included in learning. This approach is problematic because it allows only one validation set in learning and excludes patterns in the validation set to be used in learning. As a result, learning may not converge when the time-series contains multiple regimes or when training patterns are scarce.

Based on a constrained formulation, we have proposed a new cross-validation method [Wah and Qian, 2000] that defines multiple validation sets in learning and that includes the error from each validation set as a new constraint (see Figure 1). The use of multiple validation sets is especially suitable for time-series with inadequate training data and for time-series with multiple stationary regimes. The validation error for the i^{th} output unit from the k^{th}, $k = 1, \cdots, v$, validation set is defined by the *normalized mean squared error* ($nMSE$):

$$nMSE = e_{k,i} = \frac{1}{\sigma_i^2 N_k} \sum_{t=t_{k,0}}^{t_{k,1}} (o_i(t) - d_i(t))^2, \qquad (3)$$

where σ_k^2 is the variance of the true time series in $[t_{k,0}, t_{k,1}]$, and N_k is the number of patterns in the k^{th} validation set. (Note that errors on the test set in Figure 1 are defined in a similar way.) The constrained formulation then becomes:

$$\begin{aligned} \min_w \mathcal{E}_{av}(t_0, t_1) = \sum_{t=t_0}^{t_1} \sum_{i=1}^{N_o} \phi((o_i(t) - d_i(t))^2 - \tau) \\ \text{s.t.} \quad h_i(t) = (o_i(t) - d_i(t))^2 \le \tau, \\ h_{k,i}^I(w) = e_{k,i}^I \le \tau_{k,i}^I, \\ h_{k,i}^S(w) = e_{k,i}^S \le \tau_{k,i}^S, \end{aligned} \qquad (4)$$

where e^I (resp. e^S) is the $nMSE$ of the iterative (resp. single-step) validation error, and τ, $\tau_{k,i}^I$ and $\tau_{k,i}^S$ are predefined small positive constants.

Eq. (4) is a constrained nonlinear programming problem (NLP) with *non-differentiable functions*. The formulation, when applied to large time-series predictions, cannot be handled by existing Lagrangian methods that require the differentiability of functions. Methods based on penalty formulations have difficulties in convergence when penalties are not chosen properly. Sampling algorithms [Wah and Wang, 1999] based on our recently developed theory of Lagrange multipliers for discrete constrained optimization [Wah and Wu, 1999], when continuous variables are discretized to floating-point numbers, are too inefficient for solving large learning problems. To address this issue, we present in this paper an efficient learning algorithm called *violation-guided back-propagation* (VGBP).

2 Theory of Lagrange Multipliers for Discrete Constrained Optimization

To use a Lagrangian method to solve (4), we first transform it into an augmented Lagrangian function:

$$\begin{aligned} L(w, \lambda) = \mathcal{E}_{av}(t_0, t_1) \\ + \sum_{t=t_0}^{t_1} \sum_{i=1}^{N_o} \left(\lambda_i(t) \phi(h_i(t) - \tau) + \frac{1}{2} \phi^2(h_i(t) - \tau) \right) \\ + \sum_{j=I,S} \sum_{k=1}^{v} \sum_{i=1}^{N_o} \left(\lambda_{k,i}^j \phi(h_{k,i}^j - \tau_{k,i}^j) + \frac{1}{2} \phi^2(h_{k,i}^j - \tau_{k,i}^j) \right). \end{aligned} \qquad (5)$$

Since (5) is not differentiable, we discretize variables finely and solve the problem in discrete space using the theory of Lagrange multipliers for discrete constrained optimization [Wah and Wu, 1999]. Algorithm designed to solve constrained NLPs in discrete space can be extended to solve constrained NLPs in continuous space because numerical evaluations of continuous variables using digital computers can be considered as discrete approximations of the original variables up to a computer's precision. The theory in discrete space is summarized in three definitions and one theorem.

Definition 1. $\mathcal{N}_{dn}(w)$ is a *finite* user-defined set of points w' so that w' is reachable from w in one step and that w can reach any point in the discrete weight space through $\mathcal{N}_{dn}(w)$.

Definition 2. Point w is a CLM_{dn} iff (4) is feasible at w and the objective function is the smallest in $\{w\} \cup \mathcal{N}_{dn}(w)$. Note that a CLM_{dn} of (4) is the same as a feasible point.

Definition 3. $SP_{dn}(w^\star, \lambda^\star)$, a *discrete-neighborhood saddle point* at $(w^\star, \lambda^\star)$, satisfies:

$$L_d(w^*, \lambda) \le L_d(w^*, \lambda^*) \le L_d(w, \lambda^*) \qquad (6)$$

for all $w \in \mathcal{N}_{dn}(w^*)$ and all real vector λ.

Theorem 1. *First-order necessary and sufficient condition on* CLM_{dn} [Wah and Wu, 1999]. A point in the discrete space of (4) is a CLM_{dn} iff it satisfies (6) for any $\lambda \ge \lambda^*$, where $\lambda \ge \lambda^*$ means that each element of λ is not less than the corresponding element of λ^*.

The theorem shows that solving (4) in discrete space is equivalent to (the much easier problem of) finding SP_{dn} of (5). Note that the theorem does not hold in continuous space.

3 Violation-Guided Back-Propagation

In this section we describe an efficient algorithm to look for SP_{dn} in the Lagrangian space defined in (5). The shaded box in Figure 2 [Wah and Chen, 2000] shows the framework with two parts: one performing descents in the w subspace and another performing ascents in the λ subspace. As indicated earlier, random sampling in a Lagrangian space with discrete w is too inefficient. To this end, we propose in Section 3.1 to use BP to compute an approximate gradient direction in order to generate a probe. Since gradient descents may lead to infeasible local minima, we present a new annealing strategy in Section 3.2 to help escape from infeasible points. Last, we exploit special properties in the constrained formulation for

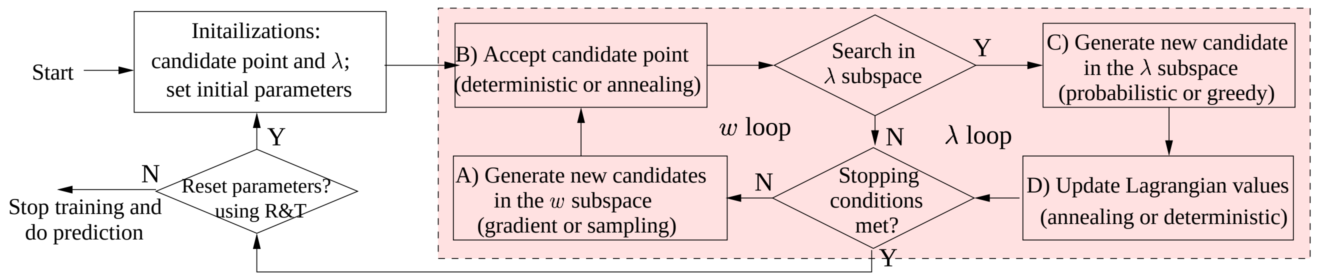

Start
Initailizations: candidate point and λ; set initial parameters
B) Accept candidate point (deterministic or annealing)
Search in λ subspace
Y
C) Generate new candidate in the λ subspace (probabilistic or greedy)
w loop
N
λ loop
Y
N
Reset parameters? using R&T
Stop training and do prediction
A) Generate new candidates in the w subspace (gradient or sampling)
N
Stopping conditions met?
D) Update Lagrangian values (annealing or deterministic)
Y

Figure 2: An iterative learning procedure using a discrete constrained formulation for ANN time-series prediction. The shaded box represents the routine to look for SP_{dn}. R&T stands for our proposed *relax-and-tighten* strategy.

ANN learning and present in Section 3.3 a new relax-and-tighten (R&T) strategy to successively tighten constraints as more relaxed constraints are satisfied. The R&T strategy is depicted in the two boxes on the left of Figure 2.

3.1 Framework to look for SP_{dn}

The w loop in Figure 2 performs descents in the w subspace by generating candidates in Box (A) and by accepting the candidates generated using deterministic or annealing rules in Box (B). Occasionally, the λ loop carries out ascents in the λ subspace by generating candidates in the λ subspace in Box (C) and by accepting them using deterministic or annealing rules in Box (D). In this subsection we present the functions of Boxes (A), (C) and (D) and leave the discussion of Box (B) to the next subsection.

For a learning problem with a large number of weights and/or training patterns, it is essential that the points generated be likely candidates to be accepted. Since (5) is not differentiable, we choose an approximate gradient direction by setting output error $g'_i(t) \leftarrow \lambda_i(t) g_i(t)$, applying BP to compute the gradient of the mean squared errors of $g'_i(t)$, generating a trial point using the approximate gradient and step size η, and mapping the trial point to (discretized) floating-point space. In this way, a training pattern with a large error (and its corresponding Lagrange multiplier) will contribute more in the overall gradient direction, leading to an effective suppression of constraint violations,

Step size η used in deriving a candidate point must be dynamic because the same candidate point will be generated repeatedly using a fixed η and a deterministic gradient algorithm. In our algorithm, we generate η uniformly in $(0, \eta_0)$ and adapt η_0 dynamically based on the acceptance ratio a of candidate points generated. The reason for the latter strategy is that a high a indicates that the current direction is promising, leading to increases in η_0 and larger step sizes. On the other hand, a low a indicates that the step size is too large for the current search terrain, leading to decreases in η_0 and smaller step sizes. After extensive experiments, we adjust η_0 as follows:

$$\eta_0 \longleftarrow \begin{cases} \eta_0 * \left(1 + \frac{2(a-0.7)}{1-0.7}\right) & \text{if } a > 0.7 \\ \eta_0 \div \left(1 + \frac{2(0.5-a)}{0.5}\right) & \text{if } a < 0.5 \end{cases} \tag{7}$$

Box (C) in Figure 2 increases λ as follows:

$$\lambda \longleftarrow \lambda + 1 \text{ if true violation } > 1.1\tau \tag{8}$$

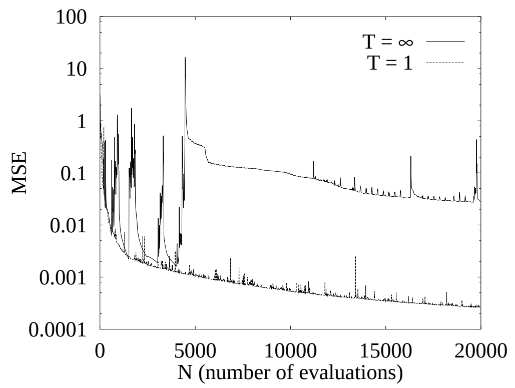


Figure 3: Progress of MSE defined in (1) for $T = 1$ and $T = \infty$ during learning of an ANN to predict the MG17 time-series.

where τ is the tolerance defined in (4) and (5). This rule penalizes a violated constraint relative to τ. We do not generate λ probabilistically because we like their effects on guidance to take place as soon as possible. The deterministic update of λ leads to the deterministic acceptance of λ in Box (D).

3.2 Probabilistic acceptances in the w subspace

Since the gradient direction computed by BP does not consider constraints due to cross validation and the step size is chosen heuristically, it is possible that a search may get stuck in infeasible local minima. In previous studies, restarts are often used to help escape from such points. However, our experimental results have shown that uncontrolled restarts may lead to loss of valuable local information collected during a search. To solve this problem, we propose an annealing strategy in Box (B) that decides whether to go from current point (w, λ) to (w', λ) according to the Metropolis probability:

$$A_T(\mathbf{w}', \mathbf{w})|_\lambda = exp\left\{\frac{(L(\mathbf{w}) - L(\mathbf{w}'))^+}{T}\right\} \tag{9}$$

where $x^+ = \min\{0, x\}$, and T is a parameters introduced to control the acceptance probability.

Figure 3 plots the progress of the mean squared errors (MSE) defined in (1) of training an ANN to predict the *Mackey-Glass*-17 time-series (in short MG17) by using two fixed temperatures: $T = 1$ and $T = \infty$, respectively. (MG17 is used as a running example throughout this section unless specified otherwise.)

When $T = \infty$ is combined with restarts, the algorithm accepts every trial point generated in the same way as traditional BP. Figure 3 illustrates this behavior by showing a search that

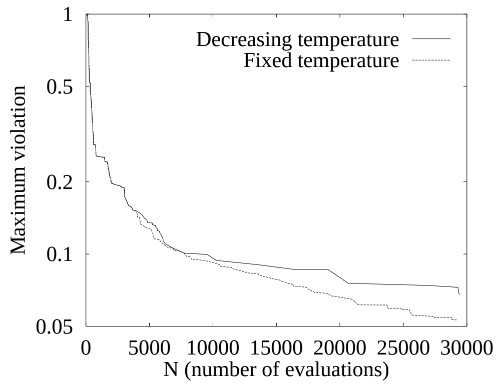


Figure 4: Decreases of maximum violation over all constraints between using a fixed temperature ($T = 5$) and using a schedule of decreasing temperatures ($T_0 = 5$, $T_{i+1} = 0.25T_i$ every 4000 evaluations, $T_\infty = 0.0003$, and η_0 was adjusted every 50 evaluations).

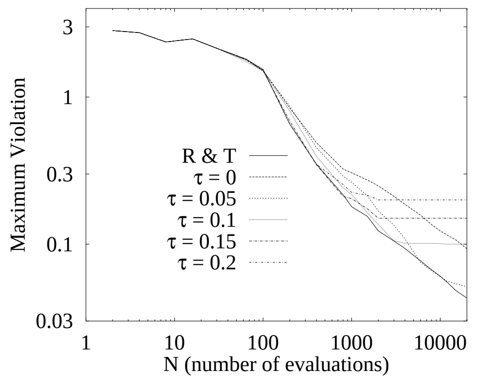


Figure 5: Decreases of maximum violation over all training patterns between using different initial violation tolerance τ (broken lines) and using our relax-and-tighten (R&T) strategy (solid line).

explores a local region in the first 4000 evaluations, got stuck in an infeasible local minimum, and restarted to a new point without keeping any history information.

On the other hand, using $T = 1$ allows the search to accept trial points according to (9) and rejects poor points with high probability. Consequently, the algorithm keeps implicitly the history information of points searched in the past and progresses smoothly without escaping into poor regions blindly.

In contrast to conventional annealing schedules that start a search at high temperatures and decrease the temperature to zero as time runs out, we use a fixed temperature throughout the search. A fixed temperature is chosen so that local descents will only be carried out by BP and not by annealing at low temperatures, and that a search will always have an opportunity to explore better regions. Figure 4 shows the improvements in maximum constraint violations when a fixed temperature is used as compared to those at low temperatures using a dynamic temperature schedule.

3.3 Relax-and-tighten (R&T) strategy

It is undesirable to set violation tolerance $\tau = 0$ initially in a search because we do not know whether such a violation tolerance can be achieved by the search. Moreover, setting $\tau = 0$ will result in considerably large violations in each pattern, leading to large λ's, a rugged search space, and a more difficult search. On the other hand, if we set a loose $\tau > 0$ initially, then most constraints can be satisfied easily, and the algorithm can focus on the few patterns with large constraint violations and increase their corresponding λ's.

Another observation is that the progress of a search differs considerably for different fixed τ's. These differences are illustrated in Figure 5 that shows the average maximum violations for different N (number of evaluations) over five independent runs. When N is small, there is little difference in maximum violations. As N is increased, runs with larger τ's have faster decreases in maximum violation than those with smaller τ's. Eventually, all the curves level off when either all constraints are almost satisfied using the specified τ or further improvement is impossible using the given ANN topology. The figure also shows a steeper rate of decrease of maximum violations with larger τ's.

Our proposed R&T strategy exploits the different convergence behavior due to different τ's by dynamically adjusting τ during a search in order to achieve the fastest convergence rate through the search. This is done by choosing a loose τ initially and by tightening $\tau \leftarrow \beta\tau$ when the maximum violation of all constraints satisfies $\max_i\{h_i(t)\} \leq (1+\gamma)\tau$, where $0 < \gamma < \beta < 1$. In this way, the search will try to use the largest possible τ at any time and will switch to a smaller τ as the convergence behavior using the original τ levels off.

Figure 5 illustrates the behavior of our proposed R&T algorithm. Initially, we set $\tau = 0.2$, leading to the steepest convergence behavior. When the convergence behavior levels off, we switch to $\tau = 0.15$ by tightening the constraints, again leading to the steepest convergence behavior for the range of N used. By repeatedly tightening constraints, the convergence behavior of R&T leads to the envelope of the best convergence behavior at all times.

The choice of the initial τ is not critical to convergence as long as it is large enough because the larger the τ is, the steeper the curve will be and the shorter the amount of time before it will level off and tighten τ. In our implementation, we set initial $\tau = 0.8\max_i\{h_i(t)\}$ over all constraints, $\beta = 0.95$, and $\gamma = 0.1$. Around those values, convergence is not sensitive to different β's and γ's.

The R&T strategy works well on a constrained formulation of ANN learning because all constraints are defined in the same range (limited by the activation function) and all constraints have similar magnitudes. In a general constrained NLP in which constraint violations may vary in large ranges, it will be necessary but difficult to define different amount of relaxations for different constraints. As a result, R&T does not work well in solving general constrained NLPs.

3.4 Parameters in VGBP

In this section, we summarize the values of parameters used in VGBP. First, we set

$$T = \alpha N_p R, \tag{10}$$

where N_p is the number of training patterns and is known when training begins, R is the range in which ANN outputs are normalized and is set to a default value of one, and α is a constant. T should be proportional to N_p because (5) is proportional to N_p when all patterns have approximately the

Table 1: Single-step and iterative test performance in $nMSE$ on *laser*. (The test set consists of patterns from 1001 to 1100. As a comparison, we also show the performance on patterns from 1001 to 1050. Boxed numbers indicate the best results; N/A stands for data not available.)

Method	Number of weights	Training	Single-step predictions		Iterative predictions	
		100-1000	1001-1050	1001-1100	1001-1050	1001-1100
FIR network [Wan, 1993]	1105	0.00044	0.00061	0.023	0.0032	0.0434
ScaleNet: Multi-scale ANN [Geva, 1998]	N/A	0.00074	0.00437	0.0035	N/A	N/A
VGBP (Run 1)	$\boxed{461}$	$\boxed{0.00036}$	0.00043	0.0034	0.0054	$\boxed{0.0194}$
VGBP (Run 2)	$\boxed{461}$	0.00107	$\boxed{0.00030}$	$\boxed{0.00276}$	$\boxed{0.0030}$	0.0294

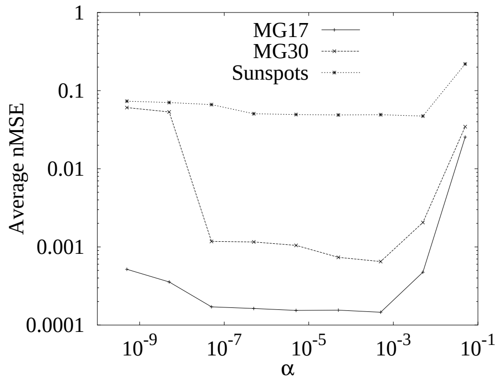


Figure 6: Robustness of VGBP with respect to α in predicting the MG-17, MG-30, and sunspots time-series.

same level of violation. Likewise, T should be proportional to R because R affects (5) in a similar manner.

Figure 6 shows the average $nMSE$'s over 10 runs of VGBP under different α's for MG17, MG30, and sunspots. Since VGBP is robust over a wide range of $\alpha \in [10^{-6}, 10^{-2}]$, we set the default α to be 10^{-3} in our implementation.

We further set η_0 in (7) to be 1.0. Since η_0 is adjusted dynamically, its initialization has no significant impact on performance. The setting of τ, β and γ in R&T has been discussed in Section 3.3.

In short, all the parameters in VGBP are set either by default or automatically, with no tuning required by users.

4 Experimental Results

We have evaluated VGBP with respect to $nMSE$ and the number of weights on several benchmarks.

Laser is a set of chaotic intensity pulsation of an NH_3 laser in the Santa Fe competition. In that competition, FIR-NN [Wan, 1993] took the first place. Table 1 shows that VGBP improves over previous algorithms in terms of prediction quality as well as number of weights used.

Sunspots contains yearly average sunspot numbers from 1700 to 1994. Using data from 1700 to 1920 for training and single-step predictions on four durations, Table 2 shows that VGBP achieves much better performance on all prediction periods, while using less weights than previous designs.

Table 3 compares single-step prediction results using VGBP with previous work on 5 chaotic time series. The two sets of *Mackey-Glass* and *Henon map* have one input and one output, whereas *Lorenz attractor* and *Ikeda attractor* have one input and two outputs as specified in [Wan, 1997] and [Aussem, 1999].

Table 2: Single-step test performance in $nMSE$ on *sunspots* for different algorithms. Results on AR(12), WNet, COMM are from [Wan, 1997], ScaleNet is from [Geva, 1998], and DRNN is from [Aussem, 1999]. Boxed numbers indicate the best results; N/A stands for data not available; n represents the number of weights/free variables used in each method.)

Method	n	Training	Single-Step Testing			
		1700-1920	1921-55	1956-79	1980-94	1921-94
AR(12)	12	0.128	0.126	0.36	0.306	0.238
WNet	113	0.082	0.086	0.35	0.313	0.219
SSNet	N/A	N/A	0.077	N/A	N/A	N/A
DRNN	30	0.105	0.091	0.273	N/A	N/A
COMM	N/A	0.079	0.065	0.24	0.188	0.148
ScaleNet	N/A	0.086	0.057	0.13	N/A	N/A
VGBP	$\boxed{11}$	$\boxed{0.0559}$	$\boxed{0.0337}$	$\boxed{0.0524}$	$\boxed{0.0332}$	$\boxed{0.0397}$

In terms of single-step predictions, Table 3 shows that VGBP uses less weights and achieves impressive $nMSE$'s one to two orders of magnitude smaller than those of other methods.

Similarly, VGBP achieves much more accurate iterative predictions as compared to those of [Wan, 1997] and [Aussem, 1999]. For example, VGBP was able to achieve iterative-prediction $nMSE$'s of 0.018 for MG17 and 0.0064 for MG30, respectively, for the duration 501-600. In contrast, applying Wan's training algorithm on FIR-NN [Wan, 1997] leads to iterative-prediction $nMSE$'s of 0.3832 for MG17 and 0.1487 for MG30. Figure 7 plots the iterative predictions of the ANN found by VGBP and that by Wan's algorithm for MG17 and MG30. The figure shows that our iterative predictions are accurate for as many as 100 steps.

In general, it may be hard to predict chaotic time series multiple steps into the future since they are unpredictable by their nature. For this reason, DRNN [Aussem, 1999] did not emphasize prediction performance, especially iterative-prediction performance [Aussem, 2001]. However, our work has shown that it is possible to predict at least the first 100 steps for the *Mackey-Glass* time series, and better for the other three sets of chaotic time series, although iterative predictions are much harder for the latter. For example, we can achieve an $nMSE$ of only 0.1369 for the first 20 steps of *Henon map*, whereas Wan's algorithm achieves an $nMSE$ of 0.6252.

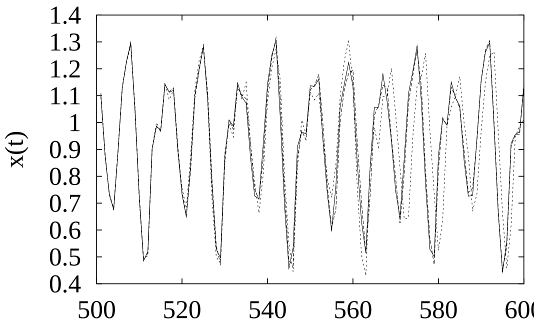
In short, constrained formulations solved by VGBP lead to superior prediction performance for the benchmarks tested.

References

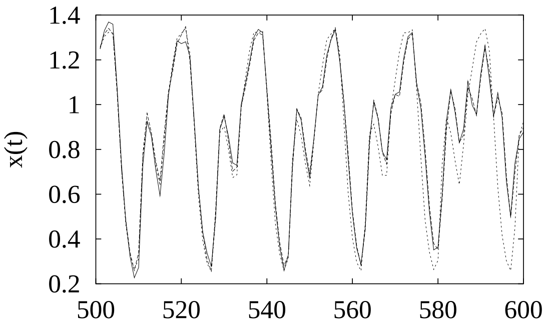
[Aussem, 1999] A. Aussem. Dynamical recurrent neural networks towards prediction and modeling of dynamical sys-

Table 3: Comparison of single-step-prediction performance in $nMSE$ on five methods: Carbon copy (C.C), linear and FIR [Wan, 1993], DRNN [Aussem, 1999], and VGBP. Carbon copy simply predicts the next time-series data to be the same as the proceeding data ($x(t+1) = x(t)$). The training (resp. testing) set indicates patterns used for learning (resp. testing). *Lorenz attractor* has two data streams labeled by x and z, respectively, whereas *Ikeda attractor* has two streams – real ($Re(x)$) and imaginary ($Im(x)$) parts of a plane wave.

Bench-Mark	Training Set	Testing Set	Performance Metrics		Design Methods				
					C.C.	Linear	FIR	DRNN	VGBP
MG17	1-500	501-2000	$nMSE$		0.6686	0.320	0.00985	0.00947	$\boxed{0.000057}$
			# of weights		0	N/A	196	197	$\boxed{121}$
MG30	1-500	501-2000	$nMSE$		0.3702	0.375	0.0279	0.0144	$\boxed{0.000374}$
			# of weights		0	N/A	196	197	$\boxed{121}$
Henon	1-5000	5001-10000	$nMSE$		1.633	0.874	0.0017	0.0012	$\boxed{0.000034}$
			# of weights		0	N/A	385	261	$\boxed{209}$
Lorenz	1-4000	4001-5500	$nMSE$	x	0.0768	0.036	0.0070	0.0055	$\boxed{0.000034}$
				z	0.2086	0.090	0.0095	0.0078	$\boxed{0.000039}$
			# of weights		0	N/A	1070	542	$\boxed{527}$
Ikeda	1-10000	10001-11500	$nMSE$	$Re(x)$	2.175	0.640	0.0080	0.0063	$\boxed{0.00023}$
				$Im(x)$	1.747	0.715	0.0150	0.0134	$\boxed{0.00022}$
			# of weights		0	N/A	2227	587	$\boxed{574}$



(a) *Mackey-Glass-17*



(b) *Mackey-Glass-30*

Figure 7: Comparisons of 100-step iterative predictions on two sets of *Mackey-Glass* time-series. Solid lines represent actual data; long dashed lines indicate predicted data using VGBP; and short dashed lines are prediction results by running Wan's FIR-NN training algorithm in [Wan, 1993].

tems. *Neurocomputing*, 28:207–232, 1999.

[Aussem, 2001] A. Aussem. Personal communications, March 2001.

[Geva, 1998] A. B. Geva. ScaleNet – multiscale neural-network architecture for time series prediction. *IEEE Trans. on Neural Networks*, 9(5):1471–1482, Sept. 1998.

[Haykin, 1994] S. Haykin. *Blind Deconvolution*. Prentice Hall, Englewood Cliffs, NJ, 1994.

[Wah and Chen, 2000] B. W. Wah and Y. X. Chen. Constrained genetic algorithms and their applications in nonlinear constrained optimization. In *Proc. Int'l Conf. on Tools with Artificial Intelligence*, pages 286–293. IEEE, November 2000.

[Wah and Qian, 2000] B. W. Wah and M. L. Qian. Time-series predictions using constrained formulations for neural-network training and cross validation. In *Proc. Int'l Conf. on Intelligent Information Processing, 16th IFIP World Computer Congress*, pages 220–226. Kluwer Academic Press, August 2000.

[Wah and Wang, 1999] B. W. Wah and T. Wang. Simulated annealing with asymptotic convergence for nonlinear constrained global optimization. *Principles and Practice of Constraint Programming*, pages 461–475, October 1999.

[Wah and Wu, 1999] B. W. Wah and Z. Wu. The theory of discrete Lagrange multipliers for nonlinear discrete optimization. *Principles and Practice of Constraint Programming*, pages 28–42, October 1999.

[Wan, 1993] E. A. Wan. *Finite Impulse Response Neural Networks with Applications in Time Series Prediction.* PhD thesis, Standford University, 1993.

[Wan, 1997] E. Wan. Combining fossils and sunspots: Committee predictions. In *IEEE Int'l Conf. on Nerual Networks*, volume 4, pages 2176–2180, Houston, USA, June 1997.