View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Aston Publications Explorer

The "Moving Targets" Algorithm for Difficult Temporal Credit Assignment Problems

Presentation for the workshop:
Neural Networks for Statistical and Economic Data
Dublin, 10-11 December 1990

Richard Rohwer
Centre for Speech Technology Research
University of Edinburgh
80 South Bridge
Edinburgh EH1 1HN
rr@cstr.ed.ac.uk

Abstract

The "moving targets" algorithm for training recurrent networks is reviewed and applied to a task which demonstrates the ability of this algorithm to use distant contextual information. Some practical difficulties are discussed, especially with regard to the minimization process. Results on performance and computational requirements of several different 2nd-order minimization algorithms are presented for moving target problems.

1 Introduction

The "moving targets" training algorithm [**?**, **?**, **?**] offers an approach to temporal credit assignment which can produce successful training for problems which are highly non-local in time. It is a gradient-based minimization algorithm set in an extended set of variables, the usual weight matrix variables together with auxiliary "moving target" variables. These auxiliary variables form a vehicle for non-local credit assignment. Unfortunately, numerical experience shows that the minimization process often terminates at a local minimum of the error measure. This motivated a comparison of several minimization algorithms on two different moving target problems.

2 The algorithm

Let us establish notation. A set of sequences of fully-specified network states can be given by numbers Y_{itp} for node i at time t in sequence p. (A weight matrix may or may not exist which produces a given pattern.) Let us call the combination of indices (itp) an *event*. A sequence of events might be only partially specified because of the absence of data for Y_{itp} at particular events. Let us call such events *hidden*, in analogy with the

hidden layers of feedforward networks for which there is no training data. At non-hidden (let us say, *visible*) events, Y_{itp} represents a desired output of the dynamical system to be trained, or else an input to be imposed by the environment. Let us refer to the former case as a *target* event and the latter as an *input* event.

$$\left.\begin{array}{lcll} y_{itp} & = & f(\sum_j w_{ij} y_{j,t-1,p}) & i \notin I \\ y_{itp} & = & Y_{itp} & i \in I \end{array}\right\} \tag{1}$$

where I is the set of input events.

In general, a node can participate in different types of events at different times. Therefore a phrase such as "the set of target nodes" is meaningful only with reference to a given time and sequence, unless (as is often true) this set happens to be the same for all times and sequences.

In the moving targets algorithm errors are assigned directly to hidden events by assigning to each hidden event a real variable which is treated as though it were target training data. These variables are the moving targets. The error measure to be minimized is

$$E = \tfrac{1}{2} \sum_{(itp) \in T \cup H} \{y_{itp} - Y_{itp}\}^2. \tag{2}$$

with y_{itp} defined as

$$\left.\begin{array}{lcll} y_{itp} & = & f(\sum_j w_{ij} Y_{j,t-1,p}) & i \notin I \\ y_{itp} & = & Y_{itp} & i \in I \end{array}\right\} \tag{3}$$

instead of (**??**). The only difference between (**??**) and (**??**) is that the $y_{j,t-1,p}$ in the sum has been changed to $Y_{j,t-1,p}$. When $(j, t-1, p)$ is a target event, this designates the (constant) target training datum for this event. When it is a hidden event, this designates a (variable) moving target. (Note that the sum **??** includes hidden events.)

The training data specifies desired results for the target events only. Therefore we are free to adjust the moving targets to any values which happen to be helpful. The same is true of the weights. Therefore training proceeds by initializing the weights and moving targets arbitrarily and minimizing the error with respect to both sets of variables using a gradient-based algorithm. After the minimization is finished, the weights are saved and the moving targets are discarded.

The credit assignment mechanism used by the moving targets algorithm is explicit: An error is assigned directly to each hidden event. Distant contextual information can be used by this algorithm when necessary, because errors at events at distant times compete additively in the error measure (**??**), rather than diminish exponentially with time as with Back Propagation. An example is presented in [**?**] and [**?**] in which a network trained by the moving targets algorithm learns to use information from 100 time steps in the past to solve a simple problem. These sources also give an example in which a network learns to execute any of four limit cycles, depending which binary number is represented on two input nodes.

3 The minimization algorithms

Several minimization algorithms were tested with these two problems. In each algorithm, the parameters were varied in the direction

$$s_i^{n+1} = r_i^n s_i^n + h_i^n \nabla_i^n \tag{4}$$

where $\nabla_i^n = \frac{dE}{dx_i}\Big|_{\boldsymbol{x}^n}$ is the i^{th} component of the gradient of the error measure E at $\boldsymbol{x}^n$, $\boldsymbol{s}^0 = \boldsymbol{\nabla}^0$, and $\boldsymbol{r}^n$ and $\boldsymbol{h}^n$ are chosen differently in different algorithms. The algorithms use various methods for choosing the step size η^n to be taken along direction $\boldsymbol{s}^n$. The algorithms are defined in table 1. Further details will appear in [**?**].

Each method was tested on each problem with 10 different random initial conditions. Results for number of gradient evaluations required and training success rate are shown in figures 1 and 2. The error bars show means and standard deviations of the number of gradient evaluations. Each plot has 3 sub-plots, one for all runs (a), one for the successful ones (s), and one for the failures (f). The partially-filled boxes at the bottom of each sub-plot show the proportions of runs in each category. For the 4-limit-cycle problem, success means less than 5% error on any target node in a freely-running net; for the 100-step context problem the threshold is 10%.

4 Conclusions

The 100 step-context problem is learned best by the max-abs algorithm, poorly by the conjugate gradient algorithm, and not at all by any other algorithm. The 4-limit-cycle problem, which involves more temporally local correlations, is also learned by the algorithms which use linesearches and analytic calculations of the second derivatives.

References

[1] R. Rohwer. The 'moving targets' training algorithm. In L. B. Almeida and C. J. Wellekens, editors, *Lecture Notes in Computer Science 412, Neural Networks*, pages 100–109. Springer-Verlag, Berlin, 1990.

[2] R. Rohwer. The 'moving targets' training algorithm. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2, pages 558–565, San Mateo CA, 1990. Morgan Kaufmann.

[3] R. Rohwer. The 'moving targets' training algorithm. In J. Kindermann and A. Linden, editors, *Distributed Adaptive Information Processing (DANIP)*, pages 175–196, Munich, 1990. R. Oldenbourg Verlag.

[4] R. Rohwer. Time trials on second-order and variable-learning-rate algorithms. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 3, San Mateo CA, to appear, April 1991. Morgan Kaufmann.