Adaptation in Nonlinear Learning Models for Nonstationary Tasks

Wolfgang Konen, Patrick Koch

Department of Computer Science, Cologne University of Applied Sciences,
51643 Gummersbach, Germany,
wolfgang.konen@fh-koeln.de

Abstract. The adaptation of individual learning rates is important for many learning tasks, particularly in the case of nonstationary learning environments. Sutton has presented with the Incremental Delta Bar Delta algorithm a versatile method for many tasks. However, this algorithm was formulated only for linear models. A straightforward generalization to nonlinear models is possible, but we show in this work that it poses some obstacles, namely the stability of the learning algorithm. We propose a new self-regulation of the model's activation which ensures stability. Our algorithm shows better performance than other approaches on a nonstationary benchmark task. Furthermore we show how to derive this algorithm from basic loss functions.

Keywords: Machine learning, IDBD, learning rates, adaptation.

1 Introduction

For many state-of-the-art learning algorithms the adaptation of learning rates (or other algorithm parameters) is important. This is particularly true if these algorithms shall behave well in nonstationary learning environments.

In 1992 Sutton [10] suggested the Incremental Delta Bar Delta (IDBD) algorithm. IDBD deals with the learning rates for trainable parameters of any underlying learning algorithm. The key idea of IDBD is that these learning rates are not predefined by the algorithm designer but they are themselves adapted as hyperparameters of the learning process. Sutton [10] expects such adaptable learning rates to be especially useful for nonstationary tasks or sequences of related tasks and demonstrates good results on a small synthetic nonstationary learning problem with 20 weights.

Sutton's algorithm is proposed to work with a linear model. But as many learning tasks exhibit nonlinear characteristics, e. g., well-known benchmark tasks for the control of physical objects like MountainCar or pole balancing, or real-world applications like the control of complex processes in plants. It is our goal to extend Sutton's IDBD to the nonlinear case. However, the nonlinear case poses some obstacles because the varying steepness in input-output relations can cause instabilities when IDBD is extended in a straightforward manner (by simply replacing the linear input-output relation with the nonlinear one). We demonstrate these obstacles and describe a way to overcome them by using a weight decay method.

The paper is organized as follows: after briefly reviewing some related work in the next paragraph, we present the IDBD method and our nonlinear generalization n-IDBD in Sec. 2. In Sec. 3 we apply n-IDBD to a nonlinear, nonstationary benchmark task. We show in Appendix A how the equations of n-IDBD can be derived from basic loss functions.

1.1 Related work

Several online learning rate adaptation schemes have been proposed over the years: IDBD [10] from Sutton is an extension of Jacobs' [4] earlier DBD (Delta Bar Delta) algorithm: it allows direct instead of batch updates. In [11] Sutton proposes the algorithms K1 and K2, two (linear) extensions to IDBD, and compares them with LMS and Kalman filtering. Koop [5] uses the IDBD algorithm for online adaptation and investigates general aspects of temporal coherence. Almeida [1] discusses another method of step-size adaptation and applies it to the minimization of nonlinear functions. Schraudolph [8] extends on the K1 algorithm and showed that it is superior to the approach described by Almeida [1].

Recently, Mahmood and Sutton [7] proposed with Autostep an extension to IDBD which has much less dependence on the meta-step-size parameter than IDBD. In the same year, Dabney and Barto [3] developed another adaptive step-size method for temporal difference learning, which is based on the estimation of upper and lower bounds. Again, both methods are proposed only for *linear* function approximation.

Schraudolph [9] and, more recently, Li [6] extended IDBD to the nonlinear case: Schraudolph's ELK1 performs an update with the instantaneous Hessian matrix of a suitable chosen loss function. The algorithm's complexity is $O(n^2)$ where n is the number of parameters to learn. This algorithm is superior to several others on the "four region" classification benchmark. However, this benchmark consists of a piecewise constant target function. Thus it does not exhibit steep and nonlinear slopes in the input-output-relationships which can be a major difficulty for adaptive learning, as we will show in this paper. – Li's KIMEL algorithm transforms the nonlinear input data with a kernel into a high-dimensional but linear feature space where linear IDBD is applied.

2 Methods

2.1 The benchmark: a nonlinear nonstationary task

In this work we consider a nonstationary task as a testbed as in [10], but with an additional nonlinearity: $n = 20$ real-valued inputs $x_1, \ldots, x_n$ are independently drawn from the standard normal distribution. The concept to learn is the weighted sum of the first 5 inputs, which is sent through a nonlinear function with slope σ_{nst}

$$y^* = \tanh\left(\sigma_{nst}\sum_{i=1}^{5} s_i x_i\right) \tag{1}$$

where all the s_i are either $+1$ or -1. To make this task nonstationary, one of the five s_i is selected randomly and switched in sign every 20 examples. Thus, the model has to learn that only the first five inputs are relevant and all other 15 inputs are irrelevant. At the same time the weights of the relevant inputs have to be able to change quickly in order to follow the drifting target.

2.2 Nonlinear Least Mean Squares (NLMS)

As a baseline learning algorithm we use a Nonlinear Least-Mean-Square (NLMS) model with constant learning rate α and error signal $\delta(t) = y^* - y(t)$:

$$y(t) = \tanh\left(N(t)\right) \quad \text{with} \quad N(t) = \sum_{i=1}^{n} w_i(t) x_i \tag{2}$$

$$\begin{aligned} w_i(t+1) &= w_i(t) + \alpha\delta(t)\frac{\partial y}{\partial w_i} \\ &= w_i(t) + \alpha\delta(t)(1 - y^2(t))x_i \end{aligned} \tag{3}$$

2.3 Incremental Delta Bar Delta (IDBD)

Sutton's IDBD algorithm [10] introduces for a *linear* unit $y(t) = \Sigma_i w_i x_i$ individual learning rates $\alpha_i = e^{\beta_i}$ for every weight w_i.

Algorithm 1 IDBD in pseudo code

```
1: Initialize: $h_i = 0,\ \beta_i = \beta_{init} \forall i$ and set $\theta$, the meta-learning rate.
2: for ( each new example $(x_1, \ldots, x_n, y^*)$) do
3:     $y = \Sigma_{i=1}^{n} w_i x_i$
4:     $\delta = y^* - y$
5:     for (every weight index $i$ ) do
6:         Set $\beta_i \leftarrow \beta_i + \theta x_i \delta\, h_i$
7:         Set $\alpha_i \leftarrow e^{\beta_i}$
8:         Set $w_i \leftarrow w_i + \alpha_i x_i \delta$
9:         Set $h_i \leftarrow h_i [1 - \alpha_i x_i^2]^+ + \alpha_i x_i \delta$        with $[d]^+ = d$ for $d > 0$, =0 else
10:    end for
11: end for
```

The main idea behind this algorithm is simple: The memory term h_i is a decaying trace of past weight changes. The increment in β_i is proportional to the product of the current weight change $x_i\delta$ and past weight changes h_i. Accumulated increments correspond to the *correlation* between current and recent weight changes [10]. In case of positive correlation the learning rate can be larger, while negative correlation indicates overshooting weight increments where the learning rate should be reduced.

2.4 Generalizing IDBD to nonlinear output units

A simple approach to generalize IDBD to the nonlinear case would be to substitute the linear equation in Step 3 of the IDBD algorithm with the nonlinear Eq. (2). Then the difference is mainly the 'outer' derivation of the nonlinearity with respect to the net input $N(t)$ in Eq. (2). If we choose $tanh()$ as nonlinearity, this derivative yields the term $(1-y^2)$ in several places. The explicit derivation will be shown later in Appendix A.

However, there is a severe problem with this simple approach: If the task exhibits a steep slope σ_{nst} in the nonlinear activation function, the adaptation of learning rates can quickly lead to a fully saturated system which does not learn the required concept. This is because large values of σ_{nst} lead to big error signals δ, and consequently to large learning rate changes and large weight changes. The output is driven into saturation sooner or later (near $+1$ or -1). With $1-y^2 \approx 0$ the gradient information becomes unreliable. As a consequence, the mean squared error (MSE) will be big or even the whole system becomes unstable.

2.5 Controlling the activation

To keep the average activation sufficiently small, we add an accumulator with

$$k_{acc}(t+1) = (1-\gamma)k_{acc}(t) + \gamma\,[y(t)]^2 \quad \text{and} \quad k_{acc}(0) = 0, \tag{4}$$

where $\gamma = 0.001$ is a sufficiently small constant. It is easy to show that the corresponding initial value problem has the solution

$$k_{acc}(t) = \int_0^t y^2(\tau)\ \gamma e^{\gamma(\tau-t)} d\tau \tag{5}$$

For $t \gg 1/\gamma$ the function $f(\tau) = \gamma e^{\gamma(\tau-t)}$ plays the role of a density function, since $\int_0^t f(\tau)d\tau \approx 1$. Thus the accumulator $k_{acc}(t)$ is a **memory trace** of the square of recent activations. If for example the output is constant, $y(t) = y_0$, then $k_{acc}(t)$ will show an exponential decay towards y_0^2. The smaller the parameter γ, the more long-term averaging the memory trace $k_{acc}(t)$ will be. The idea is now to add to the normal nonlinear weight update in Eq. (3) a new weight decay term proportional to $k_{acc}(t)$ with strength parameter ω_k

$$w_i(t+1) = w_i(t) + \alpha_i(t)\delta(t)(1-y^2(t))x_i(t) - \omega_k k_{acc}(t) w_i(t) x_i^2(t) \tag{6}$$

The purpose of the weight decay term is as follows: If the average recent activation is high in absolute value (i.e. the activations are close to saturation), then all weights will be decaying to move the output out of the saturated zone. If on the other hand the activation is close to zero, then nearly no weight decay will take place. The special setting $\omega_k = 0$ allows to recover the 'old' situation without weight decay.

We summarize our new n-IDBD method in Algorithm 2. The main difference to (linear) IDBD is the term $Y = 1-y^2$ in several places and the weight decay

Algorithm 2 n-IDBD: nonlinear IDBD in pseudo code

```
 1: Initialize: $h_i = 0$, $\beta_i = \beta_{init} \forall i$, $\gamma = 0.001$, set the meta-learning rate $\theta$, and set
    the weight decay parameter $\omega_k$.
 2: for ( each new example $(x_1, \ldots, x_n, y^*)$) do
 3:     Calculate $y$ according to Eq. (2)
 4:     Set $\delta = y^* - y$, $Y = 1 - y^2$ and $Z = Y + 2y\delta$
 5:     Update accumulator $k_{acc} \leftarrow (1-\gamma)k_{acc} + \gamma y^2$ according to Eq. (4)
 6:     for (every weight index $i$ ) do
 7:         Set $\beta_i \leftarrow \beta_i + \theta Y x_i h_i \delta$
 8:         Set $\alpha_i \leftarrow e^{\beta_i}$
 9:         Set $w_i \leftarrow w_i + \alpha_i Y x_i \delta - \omega_k k_{acc} w_i x_i^2$
10:         Set $h_i \leftarrow h_i[1 - (\alpha_i Y Z + \omega_k k_{acc})x_i^2]^+ + \alpha_i Y x_i \delta$
11:     end for
12: end for
```

term with $\omega_k k_{acc}$ in Steps 9 and 10 of the algorithm. It is a necessary prerequisite to achieve stable and fast learning. The precise form of the equations in Steps 7, 9, and 10 is derived in Appendix A.

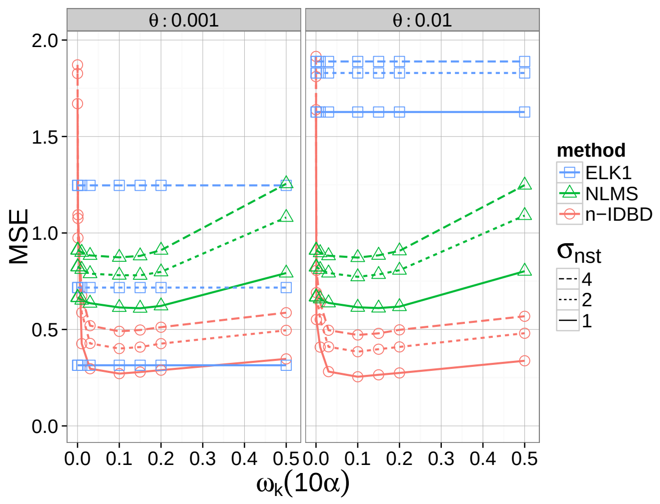

Fig. 1. Dependence on the weight decay parameter ω_K: n-IDBD has a broad minimum near $\omega_K = 0.1$. Shown is the asymptotic MSE at $t = 400\,000$. For NLMS the x-axis shows instead of ω_K the ten-fold learning rate 10α (i.e. we vary $\alpha \in [0, 0.05]$). For ELK1 there is no parameter ω_K.

3 Results

3.1 Does weight decay help?

The first experiment answers the question whether n-IDBD performs better than ordinary NLMS (or LMS) on the benchmark task and what influence the weight

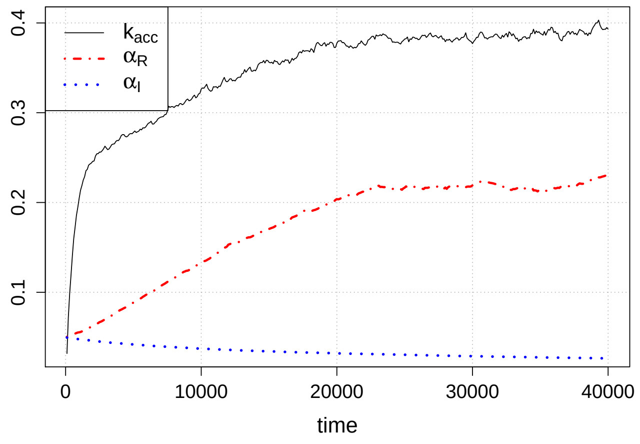

Fig. 2. Development of relevant / irrelevant learning rates (α_R, α_I) and accumulator k_{acc}. After 40 000 time steps n-IDBD adapts to the ratio $\alpha_R/\alpha_I \approx 9$.

decay has. In Fig. 1 we vary the weight decay parameter ω_k between 0 (no weight decay) and 0.5 (strong weight decay) and find a broad minimum near $\omega_k = 0.1$. The mean squared error (MSE) is taken at $t = 400\,000$ to get past any transient phases. It is measured as the average of the squared error δ^2 between time steps 300 000 and 400 000.

Without weight decay the MSE rises sharply to values above 1.5. A closer inspection of the model shows that it is in this case fully saturated ($k_{acc} = 1$)with arbitrary large weights and large learning rates. It does not learn anything, it only jumps erratically between +1 and −1. The learning rates surpass sensible bounds (e.g., $\alpha_i > 100$). A similar behavior is observed for ELK1 [8] which does not have any weight decay and exhibits in most cases large MSEs.

With weight decay the situation changes completely for a broad range of ω_k: n-IDBD has a low error everywhere, both weights and learning rates stabilize at roughly constant and sensible values. The overall activation of the unit stabilizes at a plateau $k_{acc} < 1$.[1] The MSE of n-IDBD is consistently lower than that of NLMS. This holds for all possible learning rates α of NLMS.

Fig. 2 shows the development of learning rates in one example of n-IDBD. Already after 10 000 time steps the algorithm differentiates well between relevant learning rates ($\alpha_R = \frac{1}{5}\Sigma_{i=1}^{5}\alpha_i$) and irrelevant learning rates ($\alpha_I = \frac{1}{15}\Sigma_{i=6}^{20}\alpha_i$). After approximately 22 000 time steps both the learning rates and the activation k_{acc} stabilize at constant plateaus.

Fig. 3 compares the situation with and without weight decay again, but with a focus on the longer time scale. The model with weight decay is consistently better (has a lower MSE) than the one without. Whenever MSE becomes larger than 1.2, a closer inspection of the model shows that it is fully saturated ($k_{acc} = 1.0$) and the weights are unrealistically large. Even for very gentle slopes σ_{nst}, where

[1] The precise value of $k_{acc} \in [0.2, 0.7]$ depends on the other algorithm parameters.

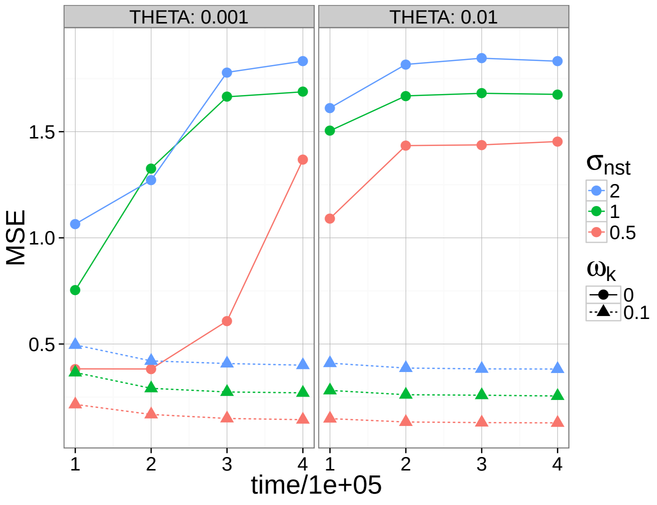

Fig. 3. Comparison of MSE without weight decay (circles, $\omega_K = 0$) to MSE with weight decay (triangles, $\omega_K = 0.1$). Shown is the MSE averaged over the last 100 000 time steps. The results with weight decay are always better.

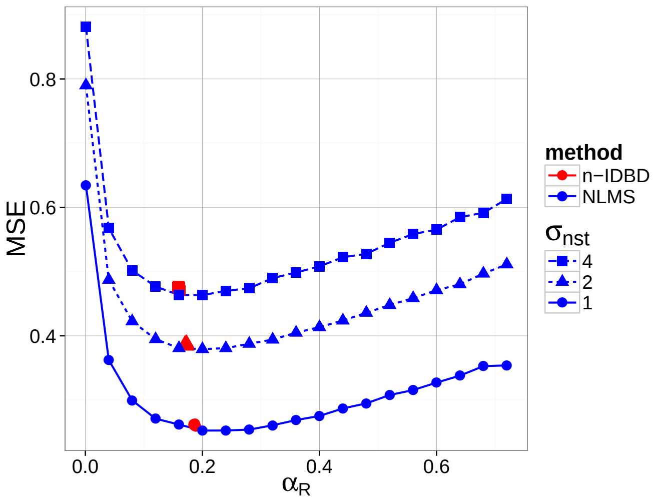

Fig. 4. Average error as a function of α_R, the learning rate for the relevant inputs. Irrelevant inputs had their weights clamped to zero in NLMS. The singular red point is the corresponding result from n-IDBD with $\theta = 0.01$.

the model without weight decay does not saturate (not shown in the figure), we find that weight decay is helpful to reduce the overall MSE.

3.2 Does nonlinear IDBD find the optimal α_i?

We have seen in the first experiment that n-IDBD automatically found large learning rates for the relevant weights and small learning rates for the irrelevant weights. Similar to [10] we want to test in a second experiment whether the learning rates found by n-IDBD are optimal. Therefore we build an 'ideal' NLMS for the task, where the irrelevant weights are already clamped to zero and the

Table 1. Comparison of the best MSE for all algorithms tested on the nonstationary task with varying slopes σ_{nst}. 'Best' means that each algorithm has its free parameters tuned to the best possible value. Parameters are $\theta = 0.01$, $\omega_k = 0.1$ for IDBD, $\alpha = 0.2$ for LMS, and $\alpha = 0.1$ for NLMS.

	Algorithm			
σ_{nst}	LMS	NLMS	IDBD (linear)	n-IDBD (nonlinear)
1.0	0.54	0.61	0.34	0.25
2.0	0.70	0.78	0.47	0.38
4.0	0.79	0.87	0.56	0.47

relevant weights get a predefined learning rate α_R. The 'ideal' NLMS has the same sigmoid and the same weight decay as n-IDBD. We get MSE-curves as shown in Fig. 4. The MSE is shown after 400 000 time steps, averaged between time steps 300 000 to 400 000. The red point for n-IDBD shows the average α_R (relevant weights) and the corresponding MSE. It is right at the minimum of each ideal curve. This shows that there is no other setting of learning rate parameters which will perform better. (The MSE is slightly higher for n-IDBD than for NLMS, because n-IDBD has the irrelevant weights not clamped precisely to zero, they fluctuate at a small level.)

We finally compare in Tab. 1 the best MSE for all algorithms. It is remarkable that linear IDBD is better on the task than nonlinear NLMS. But n-IDBD is clearly better than all other tested algorithms for all sigmoidal slopes σ_{nst}.

4 Conclusion

We have extended the adaptive, linear IDBD to the nonlinear case. It was shown that a simple extension would lead to an instable nonlinear system due to saturation effects. Similarly, the well-known ELK1 method showed diverging weights for most parameter settings as well. We proposed an additional self-regulative mechanism to control the average activation. This makes the adaptive system stable again. As in the linear case [10], the adaptive system finds the best possible learning rate on the benchmark task. The n-IDBD algorithm exhibits a smaller MSE on the benchmark task than either LMS, NLMS or linear IDBD. In an upcoming paper [2] we will show that n-IDBD can be applied to a game-learning task (Connect-Four) with more than half a million of weights as well.

A Appendix: Derivation of n-IDBD

Similar to [10], the equations of n-IDBD can be derived from a few simple principles. We start with two loss functions

$$L_1(t) = \frac{1}{2}\delta^2(t) \quad \text{and} \quad L_2(t) = \frac{1}{2}\sum_i w_i^2(t)x_i^2(t). \tag{7}$$

Both $L_1(t)$ and $L_2(t)$ should be minimized by the learning algorithm. The first term rewards small errors and the second term regularizes the complexity of the network: Weights with active inputs ($x_i^2 > 0$) should be as small as possible in their square sum.[2] In each learning step a weight change will be made in the steepest-descent direction for L_1 and for L_2:

$$\begin{aligned} w_i(t+1) &= w_i(t) - \alpha \frac{\partial L_1(t)}{\partial w_i(t)} - \Omega \frac{\partial L_2(t)}{\partial w_i(t)} \\ &= w_i(t) - \alpha \delta(t) \frac{\partial \delta(t)}{\partial w_i(t)} - \Omega w_i(t) x_i^2(t) \\ &= w_i(t) + \alpha \delta(t) \frac{\partial y(t)}{\partial N(t)} \frac{\partial N(t)}{\partial w_i(t)} - \Omega w_i(t) x_i^2(t) \\ &= w_i(t) + \alpha \delta(t)(1 - y^2(t)) x_i(t) - \Omega w_i(t) x_i^2(t) \end{aligned} \tag{8}$$

with constants α and Ω. If we identify the constant α with the slowly varying learning rate α_i and the constant Ω with $\omega_k k_{acc}(t)$ (which is justified, because $k_{acc}(t)$ approaches – after a transient phase – a roughly constant value), then Eq. (8) reproduces the weight update rule Eq. (6) for n-IDBD.

The β-update rule is governed by the minimization of L_1

$$\begin{aligned} \beta_i(t+1) &= \beta_i(t) - \theta \frac{\partial L_1(t)}{\partial \beta_i(t)} \\ &= \beta_i(t) - \theta \delta(t) \frac{\partial \delta(t)}{\partial \beta_i(t)} \\ &= \beta_i(t) + \theta \delta(t) \frac{\partial y(t)}{\partial N(t)} \frac{\partial N(t)}{\partial w_i(t)} \frac{\partial w_i(t)}{\partial \beta_i(t)} \\ &= \beta_i(t) + \theta \delta(t)(1 - y^2(t)) x_i(t) h_i(t) \end{aligned} \tag{9}$$

where we have defined $h_i(t) \equiv \frac{\partial w_i(t)}{\partial \beta_i(t)}$ as in [10]. We abbreviate $Y(t) \equiv (1 - y^2(t))$ and derive the h-update rule:

$$\begin{aligned} h_i\ (t+1) &= \frac{\partial}{\partial \beta_i} w_i(t+1) \\ &= \frac{\partial}{\partial \beta_i} \left(w_i(t) + e^{\beta_i} \delta(t) Y(t) x_i(t) - \Omega w_i(t) x_i^2(t) \right) \\ &= h_i(t) + \left[e^{\beta_i} \delta(t) Y(t) + e^{\beta_i} \frac{\partial \delta(t)}{\partial \beta_i} Y(t) + e^{\beta_i} \delta(t) \frac{\partial Y(t)}{\partial \beta_i} \right] x_i(t) - \Omega h_i(t) x_i^2(t) \\ &= h_i(t) + \alpha_i \left[\delta(t) Y(t) + \frac{\partial \delta(t)}{\partial \beta_i} Y(t) + \delta(t) \frac{\partial Y(t)}{\partial \beta_i} \right] x_i(t) - \Omega h_i(t) x_i^2(t) \end{aligned} \tag{10}$$

The terms in square brackets come from the threefold product rule when taking the partial derivative of $e^{\beta_i} \delta(t) Y(t)$ with respect to β_i. We know from Eq. (9) that

$$\frac{\partial \delta(t)}{\partial \beta_i} = -Y(t) x_i(t) h_i(t)$$

[2] It is also possible to use a simpler $L_2 = \frac{1}{2} \Sigma_i w_i^2(t)$ without the term $x_i^2(t)$. Then every weight decays in each time step. This leads to the same qualitative results in the benchmark task of Sec. 2.1, but might lead to different results in larger systems with sparse input activations.

Similarly we obtain

$$\frac{\partial Y(t)}{\partial \beta_i} = \frac{\partial(1-y^2(t))}{\partial \beta_i} = -2y(t)\frac{\partial y(t)}{\partial \beta_i} = -2y(t)Y(t)x_i(t)h_i(t)$$

If we put everything together and collect terms in Eq. (10), it is straightforward to derive

$$\begin{aligned} h_i(t+1) &= h_i(t)\left[1-(\alpha_i Y(t)Z(t)+\Omega)\, x_i^2(t)\right] + \alpha_i\delta(t)Y(t)x_i(t) \qquad (11)\\ \text{with} \quad & Z(t) = Y(t) + 2y(t)\delta(t) \end{aligned}$$

which is, after adding a positive-bounding operation for the term in square brackets, the h-update rule of n-IDBD. Here the Ω-term ensures stability as well: Even close to saturation (when $y^2(t) \approx 1$, hence $Y(t) \approx 0$) the Ω-term guarantees the decay of h_i.

References

1. Luís Almeida, Thibault Langlois, and José D. Amaral. On-line step size adaptation. Technical Report RT07/97, INESC. 9 Rua Alves Redol, Lisboa, Portugal, 1997.
2. S. Bagheri, M. Thill, P. Koch, and W. Konen. Online adaptable learning rates for the game Connect-4. submitted to: *IEEE Trans. on Computational Intelligence and AI in Games*, 2014.
3. William Dabney and Andrew G. Barto. Adaptive step-size for online temporal difference learning. In *26th AAAI Conference on Artificial Intelligence*, 2012.
4. Robert A. Jacobs. Increased rates of convergence through learning rate adaptation. *Neural Networks*, 1(4):295–307, 1988.
5. Anna Koop. Investigating experience: Temporal coherence and empirical knowledge representation. Master's thesis, University of Alberta (Canada), 2008.
6. Chunguang Li, Yan Ye, Qiuyuan Miao, and Hui-Liang Shen. Kimel: A kernel incremental metalearning algorithm. *Signal Processing*, 93(6):1586 – 1596, 2013. Special issue on Machine Learning in Intelligent Image Processing.
7. Ashique R. Mahmood, Richard S. Sutton, Thomas Degris, and Patrick M. Pilarski. Tuning-free step-size adaptation. In *IEEE InternationalConference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2121–2124. IEEE, 2012.
8. Nicol N. Schraudolph. Online local gain adaptation for multi-layer perceptrons. Technical Report IDSIA-09-98, Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale (IDSIA), Switzerland, 1998.
9. Nicol N. Schraudolph. Online learning with adaptive local step sizes. In *Neural Nets WIRN Vietri-99*, pages 151–156. Springer, 1999.
10. Richard S. Sutton. Adapting bias by gradient descent: An incremental version of delta-bar-delta. In William R. Swartout, editor, *10th AAAI Conference on Artificial Intelligence*, pages 171–176, 1992.
11. Richard S. Sutton. Gain adaptation beats least squares. In *7th Yale Workshop on Adaptive and Learning Systems*, pages 161–166, 1992.