TUNING-FREE STEP-SIZE ADAPTATION

Ashique Rupam Mahmood *Richard S. Sutton* *Thomas Degris* *Patrick M. Pilarski*

Department of Computing Science, University of Alberta, Edmonton, AB, Canada

ABSTRACT

Incremental learning algorithms based on gradient descent are effective and popular in online supervised learning, reinforcement learning, signal processing, and many other application areas. An oft-noted drawback of these algorithms is that they include a step-size parameter that needs to be tuned for best performance, which may require manual intervention and significant domain knowledge or additional data. In many cases, an entire vector of step-size parameters (e.g., one for each input feature) needs to be tuned in order to attain the best performance of the algorithm. To address this, several methods have been proposed for adapting step sizes online. For example, Sutton's IDBD method can find the best vector step size for the LMS algorithm, and Schraudolph's ELK1 method, an extension of IDBD to neural networks, has proven effective on large applications, such as 3D hand tracking. However, to date all such step-size adaptation methods have included a tunable step-size parameter of their own, which we call the *meta-step-size* parameter. In this paper we show that the performance of existing step-size adaptation methods are strongly dependent on the choice of their meta-step-size parameter and that their meta-step-size parameter cannot be set reliably in a problem-independent way. We introduce a series of modifications and normalizations to the IDBD method that together eliminate the need to tune the meta-step-size parameter to the particular problem. We show that the resulting overall algorithm, called *Autostep*, performs as well or better than the existing step-size adaptation methods on a number of idealized and robot prediction problems and does not require any tuning of its meta-step-size parameter. The ideas behind Autostep are not restricted to the IDBD method and the same principles are potentially applicable to other incremental learning settings, such as reinforcement learning.

1. STEP-SIZE ADAPTATION

In many application areas, data arrives abundantly as a stream and must be processed on a moment-by-moment basis. Some common examples of such application areas include online supervised learning, reinforcement learning, and signal processing. Learning algorithms that use data samples as soon as available and that update their estimates incrementally are among the most effective in these applications. These *incremental learning algorithms* are typically based on gradient descent and include one or more step-size parameters.

The Least Mean Squares (LMS) algorithm is one of the most widely used incremental learning algorithms for online supervised-learning applications. In the setting where the LMS algorithm is commonly applied, data is considered to arrive in a series of steps. At each step k, data is presented as a vector of n input features $x_{1,k}, \ldots, x_{n,k}$, where each $x_{i,k} \in \mathbb{R}$, and as a single target output $y_k \in \mathbb{R}$. The target is estimated as a weighted sum of the feature components. We denote the error at step k as

$$\delta_k = y_k - \sum_{i=1}^{n} w_{i,k} x_{i,k}, \tag{1}$$

where the $w_{i,k} \in \mathbb{R}$ are learned weights. At each step the LMS algorithm increments its weights proportional to the error and the corresponding feature:

$$w_{i,k+1} = w_{i,k} + \alpha \delta_k x_{i,k},$$

where $\alpha > 0$ is a scalar step-size parameter.

Although in the conventional LMS algorithm the step size α is a scalar, better performance can be obtained if it is a vector (a separate step-size parameter for each feature). In either case, the problem of setting the parameter(s) and the dependence of their best settings on the particularities of the problem must be addressed. Several methods have been proposed to set step-size parameters automatically. Examples include the Incremental-Delta-Bar-Delta (IDBD) method (Sutton, 1992a), which adapts a vector step size for the LMS algorithm, and the ELK1 method (Schraudolph, 1999), which adapts the step-size parameters for neural networks (see also Sutton, 1981; Jacobs, 1988; Benveniste et al., 1990). These methods have proven effective on several large applications including 3D hand tracking (Bray et al., 2004), brain–computer interfaces (Buttfield et al., 2006), and real-time robot control (Schaal et al., 2001). These methods all increment the components of the step-size parameter proportional to an estimate of the partial derivative of the squared error with respect to the step size, a sort of "meta" gradient descent (Sutton, 1992a). The general idea is to update at each step by

$$\alpha_{i,k+1} = \alpha_{i,k} - \mu \widehat{\frac{\partial \delta_t^2}{\partial \alpha_i}}, \tag{2}$$

where $\mu > 0$ is a scalar *meta-step-size* parameter.

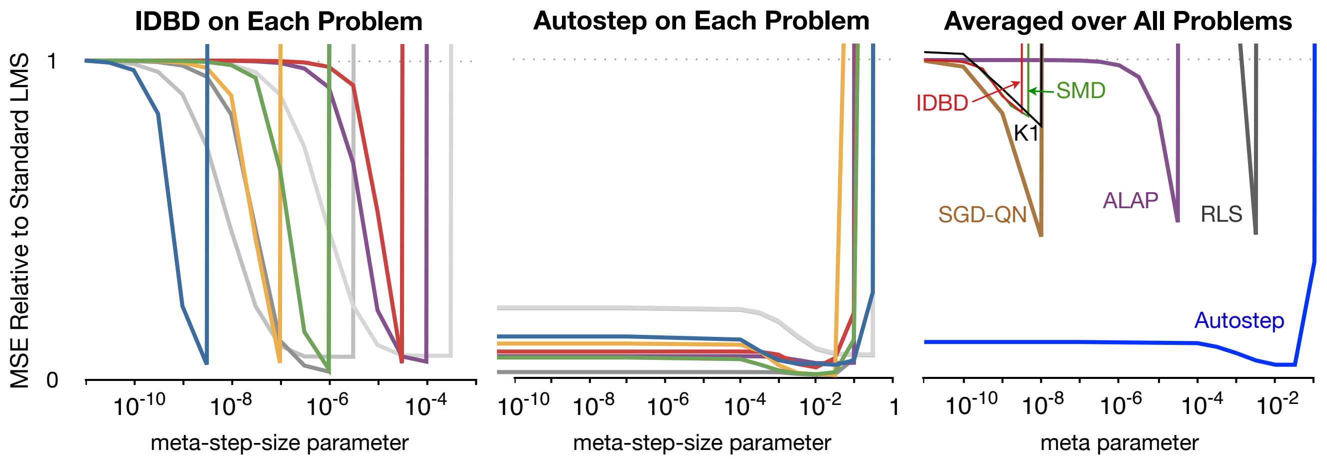

Fig. 1. **Our main results.** *Left*: A representative step-size adaptation method, IDBD, performs better than standard LMS (the dotted line), but only if its meta-step-size parameter is varied substantially from problem to problem. *Middle*: Our new method, Autostep, performs well on all problems at the same value of its meta-step-size parameter ($\approx 10^{-2}$). *Right*: If we restrict each method to a single value of its meta parameter, then Autostep performs much better than all the other step-size adaptation methods and the Recursive Least Squares (RLS) algorithm, and is more robust to the setting of its meta parameter.

2. DEPENDENCE ON THE META-STEP SIZE

Although step-size adaptation methods eliminate the need to manually tune the step-size parameter, these methods introduce a *meta*-step-size parameter which must itself be tuned. In this section, we empirically demonstrate the dependence of the existing step-size adaptation methods on their meta-step-size parameters. We used two idealized supervised learning problems and six robot prediction problems. The first idealized problem is a benchmark nonstationary problem, taken from Sutton (1992b). The second idealized problem is identical except that the variance of the target weights' drift is 100 times larger. The robot prediction problems use a data set collected by running a sensor-rich mobile robot inside a closed testing environment for more than eight hours. This data contains time records of 56 sensors over 3 million steps. The data was divided into 30 sets for separate runs. The sensor values were normalized such that their average was zero and estimated variance was one. In each problem values of one of the raw sensors at the next step were predicted by using values of all the 56 normalized sensors at the previous step. Six different sensors were picked as targets, each from a different class of sensors. As the sensor values varied frequently over time due to the robot's movements, these problems are nonstationary and hence appropriate for step-size adaptation. The data set is available online (RLAI, 2010).

On all problems, we applied the most prominent existing step-size adaptation methods, including IDBD (Sutton, 1992a), K1 (Sutton, 1992b), ALAP (Almeida et al., 1998), SMD (Schraudolph, 1999), and the method by Benveniste et al. (1990, see also Kushner and Yang, 1995). All these methods follow roughly the schema of (2). Performance for these methods was measured by varying the meta-step-size values between 10^{-11} and 10^{3}. The step size for all methods was initialized to a standard, roughly tuned value—$0.1/n$ for the idealized problems and $0.0001/n$ for the robot problems (values an order of magnitude larger than this resulted in divergence for most methods). For each method on each problem, we measured the Mean Squared Error (MSE), averaged over time steps and then averaged over 30 runs. Our final performance measure, shown in Figure 1, is the ratio of the method's MSE to the MSE of the conventional LMS algorithm with its step-size parameter set permanently to the standard values. We refer to LMS with this setup as *standard LMS*.

The left panel of Figure 1 shows the dependence and sensitivity of IDBD to its meta-step-size parameter on all the problems. With a low meta-step size (e.g., 10^{-11}), IDBD is essentially the same as standard LMS, and so its relative performance was 1.0. Note that IDBD's performance depended strongly on its meta-step-size parameter. For example, for the problem of predicting the robot's acceleration sensor, the best performance (lowest MSE) was achieved between 10^{-9} and 10^{-8} (leftmost blue curve) whereas, for the problem of predicting the current draw on the robot's drive motor, the lowest MSE was achieved for a meta-step size between 10^{-5} and 10^{-4} (red curve). This demonstrates that the best performing value of the meta-step-size parameter of IDBD shifts by orders of magnitude across different problems. The ranges of meta-step size that achieved good performance were also narrow compared to the overall range of values that has to be searched. Similar large shifts in the value of the best meta parameter across problems were observed for all the other methods except Autostep (middle panel), a new method that we present in the next section. The right panel compares all methods, averaged across problems, when restricted to a single value of their meta parameter.

3. THE AUTOSTEP METHOD

The main contribution of this paper is our proposed method, *Autostep* (see also Mahmood, 2010), that adapts a vector of step-size parameters without requiring tuning of its meta-step-size parameter. The method is given in Table 1. Autostep is based on the IDBD method, which is completely described by (1) and the following three equations:

$$\begin{aligned} \alpha_{i,k+1} &= \alpha_{i,k} \exp\left(\mu \delta_k x_{i,k} h_{i,k}\right), \qquad (3) \\ w_{i,k+1} &= w_{i,k} + \alpha_{i,k+1} \delta_k x_{i,k}, \\ h_{i,k+1} &= \left[1 - \alpha_{i,k+1} x_{i,k}^2\right]^+ h_{i,k} + \alpha_{i,k+1} \delta_k x_{i,k}, \end{aligned}$$

where the $h_{i,k}, i = 1, \ldots, n$, are auxiliary memory variables, and $\mu > 0$ is a meta-step-size parameter as in (2).

Insight into why the best value of IDBD's meta-step-size parameter varies from problem to problem can be gained by doing a unit analysis of its update equations. The quantity $\delta_k x_{i,k} h_{i,k}$ appearing in its main update equation (3) has units of y^2 (where y is the target output). To maintain the units of α_i in (3), the meta-step size μ must have the inverse units, that is, units of $1/y^2$ (so that the exponentiated quantity is unitless). Hence, problems with different variances of target outputs will require very different μs, and thus we see the shifts in the first panel of Figure 1. Similar is true for all the other step-size adaptation methods except ALAP. In ALAP, the meta-step-size parameter is normalized so that it is unitless. However, in our experiments, the meta-step-size parameter of ALAP required tuning across problems. This is due to the fact that the normalizer in ALAP is estimated as a running average and it requires a number of iterations before mitigating the effect of units. Hence, a sudden change can abruptly affect its step-size update.

A second problem that leads to parameter dependence in IDBD is that the best range of μ is narrow. This range also lies close to the value of μ for which IDBD starts to diverge. If the step-size parameter can be checked against growing large, a large μ may not result in such instability. None of the step-size adaptation methods including ALAP consider this issue; this is another potential reason why ALAP was not effective in avoiding parameter tuning.

There are step-size adaptation methods that are not based on meta gradient descent, such as SGD-QN (Bordes et al., 2009) and those proposed by Sompolinsky et al. (1995) and Murata et al. (1996). These methods incrementally update the step-size parameter as a running average of some statistics, such as the squared error or the norm of the gradient. These methods contain tunable meta parameters as well. We investigated the dependence of these methods on their meta parameters and found similar shifts in their performance curves. We observed that the shifts in these methods occur for similar reasons as for methods based on meta gradient descent (e.g., due to units of meta parameters).

Table 1. The Autostep Method

```
Initialization:
    Set $\mu$ and $\tau$ as appropriate (e.g., $10^{-2}$ and $10^4$)
    for $i = 1, \ldots, n$:
        $h_i \leftarrow v_i \leftarrow 0$
        Initialize $w_i$ and $\alpha_i$ as desired (e.g., 0 and 0.1)
for each new data sample $(x_1, \ldots, x_n, y)$:
    $\delta \leftarrow y - \sum_{i=1}^n w_i x_i$
    for $i = 1, \ldots, n$:
        $v_i \leftarrow \max\left(|\delta x_i h_i|, v_i + \frac{1}{\tau}\alpha_i x_i^2 \left(|\delta x_i h_i| - v_i\right)\right)$   (4)
        if $v_i \neq 0$:
            $\alpha_i \leftarrow \alpha_i \exp\left(\mu \frac{\delta x_i h_i}{v_i}\right)$   (5)
    $M \leftarrow \max\left(\sum_{i=1}^n \alpha_i x_i^2, 1\right)$   (6)
    for $i = 1, \ldots, n$:
        $\alpha_i \leftarrow \frac{\alpha_i}{M}$   (7)
        $w_i \leftarrow w_i + \alpha_i \delta x_i$
        $h_i \leftarrow h_i\left(1 - \alpha_i x_i^2\right) + \alpha_i \delta x_i$
```

To produce the Autostep method shown in Table 1, we introduced two modifications to IDBD that together eliminate the dependence of performance on the meta-step-size parameter. The first modification mitigates the shift in the best range of μ across problems, and the second modification reduces the sensitivity of performance to μ within the same problem by reducing the step-size parameter whenever it would cause overshooting. The method involves slightly more computation per step than IDBD, but note that its time and memory complexity still scales linearly with the number of input features.

The first modification of IDBD to produce Autostep is to normalize each step-size update component $\delta x_i h_i$ with a running maximum of its absolute values (see (4) in Table 1). As a consequence, the normalized term $\delta x_i h_i / v_i$ in (5) of Table 1 becomes unitless, its value cannot be more than 1.0, and a sudden change in the target output cannot abruptly affect the step-size update. Note that the update of the normalizer v_i uses α_i in such a way that its speed of tracking is self-regulated.

To understand the second modification, consider the quantity $\sum_{i=1}^n \alpha_{i,k+1} x_{i,k}^2$, which might be called the *effective step size* for step k. If this quantity is one half, then the error on a single data sample would have been reduced by one half. In general, letting δ'_k denote the error on kth step after the weight update ($\delta'_k = y_k - \sum_{i=1}^n w_{i,k+1} x_{i,k}$), then a simple calculation shows that $\frac{\delta_k - \delta'_k}{\delta_k} = \sum_{i=1}^n \alpha_{i,k+1} x_{i,k}^2$. It follows then that if the effective step size is greater than one, then we will overshoot the minimum error. The second modification computes the effective step size and, if it is greater than one, scales α so that the effective step size is one (see (6) and (7) in Table 1). The overall effect is to make overshooting, and thus divergence, literally impossible for Autostep.

4. RESULTS AND DISCUSSION

The performance of Autostep on our two idealized and six robot problems is shown in the middle panel of Figure 1. In contrast to the performance of IDBD, shown in the first panel, the performance of Autostep did not shift with respect to μ across problems. Curves with the same color in the two panels are for the same problems. With $\mu = 10^{-2}$, Autostep achieved near-best performance on all problems.

It remains unclear why $\mu = 10^{-2}$ performs best for Autostep and whether this would be true generally. Examining the update rules reveals that this value permits step sizes to change by about 1% per step. Thus, it takes at least 70 steps to double or halve a step size, which is fairly fast, while leaving enough time to cancel out noisy fluctuations.

A second way in which Autostep is superior to and less parameter dependent than IDBD is that it eliminates the need to tune the initial step size. Recall that IDBD required significant tuning of the initial step size for the two classes of problems, whereas Autostep used the same value (0.1) for all of them. In fact, Autostep's performance as measured here is nearly insensitive to the initial step size. It does have some effect on initial performance, but we have found that it is easy to find an excellent initial step size for Autostep using the standard rules of thumb for LMS, for example, by using 0.1 divided by the expected norm of the input feature vectors.

The third and final parameter of Autostep, τ, was 10^4 on all problems in our experiments. Autostep's performance was not strongly dependent on τ, though further exploration of a wider range of problems, and particularly of a wider range of problem durations, is needed before a definitive statement can be made. As a rule of thumb, we recommend setting τ to the number of data samples expected to be needed for learning.

The utility of Autostep as a tuning free method is clearly shown in the final panel of Figure 1, which shows performance of all methods, averaged across problems, as a function of their meta parameter. Autostep performed dramatically better than not only all the other step-size adaptation methods, but also the Recursive Least Squares (RLS) algorithm, a widely used linear learning algorithm when one can afford quadratic per-step complexity. (The meta parameter for RLS here is its forgetting factor.) Only Autostep achieved an overall satisfactory performance with a single parameter setting.

Autostep can be extended in interesting ways. Extensions to nonlinear settings would make tuning-free step-size adaptation possible for artificial neural networks. Extensions to reinforcement learning would be natural given that many reinforcement learning problems are inherently nonstationary. We are currently exploring extensions of Autostep to online temporal-difference learning.

5. REFERENCES

Almeida, L. B., Langlois, T., Amaral, J. D., and Plakhov, A. (1998). Parameter adaptation in stochastic optimization. In Saad, D. (Ed.), *On-Line Learning in Neural Networks*, pp. 111–134. Cambridge University Press.

Benveniste, A., Métivier, M., and Priouret, P. (1990). *Adaptive algorithms and stochastic approximations*. Springer-Verlag.

Bordes, A., Bottou, L., and Gallinari, P. (2009). SGD-QN: Careful quasi-Newton stochastic gradient descent. *Journal of Machine Learning Research*, 10:1737–1754.

Bray, M., Koller-Meier, E., Müller, P., Gool, L. V., and Schraudolph, N. N. (2004). 3D hand tracking by rapid stochastic gradient descent using a skinning model. In *Proc. Intl. Conf. Artificial Neural Networks*, pp. 59–68.

Buttfield, A., Ferrez, P. W., and Millán, R. J. del. (2006). Towards a robust BCI: Error potentials and online learning. *IEEE Trans. Neural Sys. Rehab. Eng.*, 14:164–168.

Jacobs, R. A. (1988). Increased rates of convergence through learning rate adaptation. *Neural Networks*, 1(4):295–307.

Kushner H. J., and Yang, J. (1995). Analysis of adaptive step size SA algorithms for parameter tracking. *IEEE Trans. Autom. Control*, 40:1403–1410.

Mahmood A. (2010). Automatic step-size adaptation in incremental supervised learning. Master's thesis, Department of Computing Science, University of Alberta, Edmonton, AB T6G 2E8.

Murata, N., Müller, K. R., Ziehe, A., and Amari, S. (1996). Adaptive on-line learning in changing environments. In: *Advances in Neural Information Processing Systems 9*, pp. 312–318. MIT press.

RLAI (2010). daylong04.crtrlog.gz. *Critterbot*. Retrieved June 1, 2010, from `http://critterbot.rl-community.org/logs`.

Schaal, S., Atkeson, C., and Vijayakumar, S. (2001). Scalable techniques from nonparametric statistics for real-time robot learning. *Applied Intelligence*, 17(1):49–60.

Schraudolph, N. N. (1999). Local gain adaptation in stochastic gradient descent. In: *Proc. Intl. Conf. Artificial Neural Networks*, pp. 569–574.

Sompolinsky, H., Barkai, N., and Seung, H. S. (1995). Online learning of dichotomies: Algorithms and learning curves. In *Neural networks: The statistical mechanics perspective. Proceedings of the CTP-PBSRI Joint Workshop on Theoretical Physics*, pp. 105–130. Singapore: World Scientific.

Sutton, R. S. (1981). Adaptation of learning rate parameters. In: Goal Seeking Components for Adaptive Intelligence: An Initial Assessment, by A. G. Barto and R. S. Sutton. Air Force Wright Aeronautical Laboratories Technical Report AFWAL-TR-81-1070. Wright-Patterson Air Force Base, Ohio 45433.

Sutton, R. S. (1992a). Adapting bias by gradient descent: An incremental version of delta-bar-delta. In *Proc. 10th National Conference on Artificial Intelligence*, pp. 171–176.

Sutton, R. S. (1992b). Gain adaptation beats least squares? In *Proc. of the 7th Yale Workshop on Adaptive and Learning Systems*, pp. 161–166.