# Adapting Bias by Gradient Descent: An Incremental Version of Delta-Bar-Delta

**Article** · June 1995

Source: CiteSeer



**1 author:**

Richard Sutton
University of Alberta
**197** PUBLICATIONS **77,218** CITATIONS

Appeared in *Proceedings of the Tenth National Conf. on Artificial Intelligence*, pp. 171–176, MIT Press, 1992.

# Adapting Bias by Gradient Descent: An Incremental Version of Delta-Bar-Delta

**Richard S. Sutton**
GTE Laboratories Incorporated
Waltham, MA 02254
sutton@gte.com

## Abstract

Appropriate bias is widely viewed as the key to efficient learning and generalization. I present a new algorithm, the Incremental Delta-Bar-Delta (IDBD) algorithm, for the learning of appropriate biases based on previous learning experience. The IDBD algorithm is developed for the case of a simple, linear learning system—the LMS or delta rule with a separate learning-rate parameter for each input. The IDBD algorithm adjusts the learning-rate parameters, which are an important form of bias for this system. Because bias in this approach is adapted based on previous learning experience, the appropriate testbeds are drifting or non-stationary learning tasks. For particular tasks of this type, I show that the IDBD algorithm performs better than ordinary LMS and in fact finds the optimal learning rates. The IDBD algorithm extends and improves over prior work by Jacobs and by me in that it is fully incremental and has only a single free parameter. This paper also extends previous work by presenting a derivation of the IDBD algorithm as gradient descent in the space of learning-rate parameters. Finally, I offer a novel interpretation of the IDBD algorithm as an incremental form of hold-one-out cross validation.

## Introduction

People can learn very rapidly and generalize extremely accurately. Information theoretic arguments suggest that their inferences are too rapid to be justified by the data that they have immediately available to them. People can learn as fast as they do only because they bring to the learning situation a set of appropriate biases which direct them to prefer certain hypotheses over others. To approach human performance, machine learning systems will also need an appropriate set of biases. Where are these to come from?

Although this problem is well known, there are few general answers. The field of pattern recognition has long known about the importance of feature selection, and the importance of representations is a recurring theme in AI. But in both of these cases the focus has always been on designing in a good bias, not on acquiring one automatically. This has resulted in an accumulation of specialized and non-extensible techniques. Is there an alternative? If bias is what a learner brings to a learning problem, then how could the learner itself generate an appropriate bias? The only way is to generate the bias from previous learning experience (e.g., Rendell, Seshu, & Tcheng 1987). And this is possible only if the learner encounters a series of different problems requiring the same or similar biases. I believe that is a correct characterization of the learning task facing people and real-world learning machines.

In this paper, I present a new algorithm for learning appropriate biases for a linear learning system based on previous learning experience. The new algorithm is an extension of the Delta-Bar-Delta algorithm (Jacobs 1988; Sutton 1982; Barto & Sutton 1981; Kesten 1958) such that it is applicable to incremental tasks—supervised learning tasks in which the examples are processed one by one and then discarded. Accordingly, I call the new algorithm the Incremental Delta-Bar-Delta (IDBD) algorithm. The IDBD algorithm can be used to accelerate learning even on single problems, and that is the primary way in which its predecessors have been justified (e.g., Jacobs 1988; Silva & Almeida 1990; Lee & Lippmann 1990; Sutton 1986), but its greatest significance I believe is for non-stationary tasks or for sequences of related tasks, and it is on the former that I test it here.

## The IDBD Algorithm

The IDBD algorithm is a *meta-learning* algorithm in the sense that it learns the learning-rate parameters of an underlying base learning system. The base learning system is the Least-Mean-Square (LMS) rule, also known as the delta rule, the ADALINE, the Rescorla-Wagner rule, and the Widrow-Hoff rule (see, e.g., Widrow & Stearns 1985). This learning system is often thought of as a single connectionist unit as shown in figure 1. The unit is linear, meaning that its output $y(t)$, at each time step (example number) $t$, is a

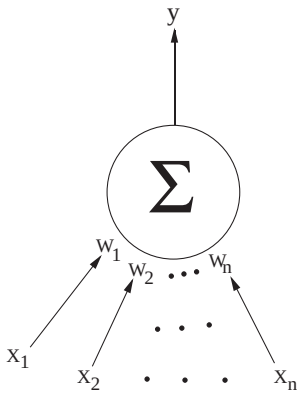


Figure 1: The base-level learning system is a single linear unit using the LMS or delta rule.

weighted sum of its real-valued inputs $x_i(t)$:

$$y(t) = \sum_{i=1}^{n} w_i(t)x_i(t), \tag{1}$$

were each $w_i(t)$ is the value at time $t$ of a modifiable weight $w_i$ associated with $x_i$. At each time step, the learner receives a set of inputs, $x_i(t)$, computes its output, $y(t)$, and compares it to a given desired output, $y^*(t)$. The aim of learning is to minimize the squared error $\delta^2(t)$, where $\delta(t) = y^*(t) - y(t)$, on future time steps. The LMS learning rule updates the weights at each time step according to:

$$w_i(t+1) = w_i(t) + \alpha\delta(t)x_i(t), \tag{2}$$

where $\alpha$ is a positive constant called the learning rate.

In the IDBD algorithm there is a different learning rate, $\alpha_i$, for each input $x_i$, and these change according to a meta-learning process (cf. Hampson & Volper 1986). The base-level learning rule is [1]

$$w_i(t+1) = w_i(t) + \alpha_i(t+1)\delta(t)x_i(t). \tag{3}$$

The learning rates are a powerful form of bias in this system. Learning about irrelevant inputs acts as noise, interfering with learning about relevant inputs. A rough rule of thumb is that learning time is proportional to the sum of the squares of the learning rates (assuming all inputs have equal variance; see Widrow & Stearns 1985). In effect, learning rates are a valuable resource that must be distributed carefully. Inputs that are likely to be irrelevant should be given small learning rates, whereas inputs that are likely to be relevant should be given large learning rates.

In the IDBD algorithm, the learning rates are all of the form

$$\alpha_i(t) = e^{\beta_i(t)}. \tag{4}$$

[1] The $\alpha_i$ are indexed by $t+1$ rather than $t$ simply to indicate that their update, by a process described below, occurs *before* the $w_i$ update (see figure 2).

This exponential relationship between the learning rate, $\alpha_i$, and the memory parameter that is actually modified, $\beta_i$, has two advantages. First, it is a natural way of assuring that $\alpha_i$ will always be positive. Second, it is a mechanism for making geometric steps in $\alpha_i$: if $\beta_i$ is incremented up and down by a fixed step-size, then $\alpha_i$ will move up or down by a fixed fraction of its current value, e.g., up or down by 10%. This is desirable because some $\alpha_i$ must become very small while others remain large; no fixed step-size would work well for all the $\alpha_i$.

The IDBD algorithm updates the $\beta_i$ by

$$\beta_i(t+1) = \beta_i(t) + \theta\delta(t)x_i(t)h_i(t), \tag{5}$$

where $\theta$ is a positive constant, the *meta-learning rate*, and $h_i$ is an additional per-input memory parameter updated by

$$h_i(t+1) = h_i(t)\Big[1-\alpha_i(t+1)x_i^2(t)\Big]^+ + \alpha_i(t+1)\delta(t)x_i(t), \tag{6}$$

where $[x]^+$ is $x$, if $x > 0$, else 0. The first term in the above equation is a decay term; the product $\alpha_i(t+1)x_i^2(t)$ is normally zero or a positive fraction, so this term causes a decay of $h_i$ towards zero. The second term increments $h_i$ by the last weight change (cf. (3)). The memory $h_i$ is thus a decaying trace of the cumulative sum of recent changes to $w_i$.

The intuitive idea behind the IDBD algorithm is a simple one. Note that the increment to $\beta_i$ in (5) is proportional to the product of the current weight change, $\delta(t)x_i(t)$, and a trace of recent weight changes, $h_i(t)$. By accumulating this product, the overall change in $\beta_i$ becomes proportional to the *correlation* between current and recent weight changes. If the current step is positively correlated with past steps, that indicates that the past steps should have been larger (and equation (5) accordingly increases $\beta_i$). If the current step is negatively correlated with past steps, that indicates that the past steps were too large; the algorithm is overshooting the best weight values and then having to re-correct in the opposite direction (here equation (5) decreases $\beta_i$).

The intuitive idea of the IDBD algorithm as described above is the same as that of Jacob's (1988) Delta-Bar-Delta algorithm. The primary difference is that Jacobs's algorithm can be applied only on a batch-by-batch basis, with updates after a complete presentation of a training set, whereas here we assume examples arrive one-by-one and are not necessarily revisited afterwards. The key to making the new algorithm incremental is the way the trace $h_i$ is defined such that it fades away only to the extent that the corresponding input $x_i$ is present, as indicated by $x_i^2(t)$. The new algorithm also improves over Jacobs's in that the decay rate is not a separate free parameter, but is tied to the current learning rate. The new algorithm in fact has only one free parameter, the meta-learning rate, $\theta$, whereas Jacobs's algorithm has three free parameters.

Initialize $h_i$ to 0, and $w_i, \beta_i$ as desired, $i = 1, \ldots, n$
Repeat for each new example $(x_1, \ldots, x_n, y^*)$:

$y \leftarrow \sum_{i=1}^{n} w_i x_i$
$\delta \leftarrow y^* - y$
Repeat for $i = 1, \ldots, n$:
$\beta_i \leftarrow \beta_i + \theta \delta x_i h_i$
$\alpha_i \leftarrow e^{\beta_i}$
$w_i \leftarrow w_i + \alpha_i \delta x_i$
$h_i \leftarrow h_i + [1 - \alpha_i x_i^2]^+ + \alpha_i \delta x_i$

Figure 2: The IDBD Algorithm in Pseudocode

On the other hand, Jacobs's algorithm was designed for nonlinear networks. While I do not foresee great difficulties extending the IDBD algorithm to the nonlinear case, that is beyond the scope of this paper.

In practice, it is often useful to bound each $\beta_i$ from below by, say, $-10$, to prevent arithmetic underflows. In addition, it is prudent to limit the change in $\beta_i$ on any one step to, say, $\pm 2$. However, these boundings were not required to obtain the empirical results presented in the next section.

## Results

The capabilities of the IDBD algorithm were assessed using a series of tracking tasks—supervised-learning or concept-learning tasks in which the target concept drifts over time and has to be tracked (cf. Schlimmer 1987). Non-stationary tasks are more appropriate here than conventional learning tasks because we are trying to assess the IDBD algorithm's ability to learn biases during early learning and then use them in later learning. To study this one needs a continuing learning problem, not one that can be solved once and is then finished.

### Experiment 1: Does IDBD help?

Experiment 1 was designed to answer the question: Does the IDBD algorithm perform better than the ordinary LMS algorithm without IDBD? The task involved 20 real-valued inputs and one output. The inputs were chosen independently and randomly according to a normal distribution with mean zero and unit variance. The target concept was the sum of the first five inputs, each multiplied either by $+1$ or $-1$, i.e.:

$$y^* = s_1 x_1 + s_2 x_2 + s_3 x_3 + s_4 x_4 + s_5 x_5 + 0x_6 + 0x_7 + \cdots + 0x_{20},$$

where all the $s_i$ are either $+1$ or $-1$. To make it a tracking problem, every 20 examples one of the five $s_i$ was selected at random and switched in sign, from $+1$ to $-1$, or vice versa. Thus, the same five inputs were always relevant, but their relationship to the target concept occasionally changed. If the IDBD algorithm can successfully identify which inputs are relevant, then it should be able to track the drifting target function more accurately than ordinary LMS.

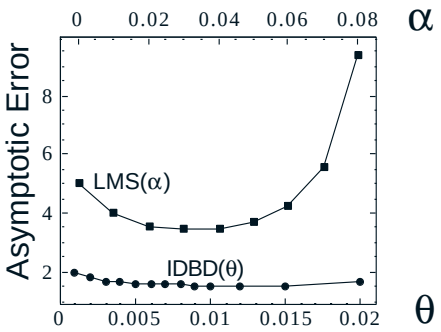


Figure 3: Comparison of the average asymptotic performances of IDBD and LMS algorithms over the relevant ranges of their step-size parameters ($\alpha$, upper axis, for LMS, and $\theta$, lower axis, for IDBD). The IDBD algorithm results in less than half the level of error over a broad range of values of its step-size parameter. For parameter values above those shown, both algorithms can become unstable.

Because this is a tracking task, it suffices to perform one long run and measure the asymptotic tracking performance of the algorithms. In this experiment I ran each algorithm for 20,000 examples so as to get past any initial transients, and then ran another 10,000 examples. The average mean-squared error over that 10,000 examples was used as the asymptotic performance measure of the algorithm. The algorithms used were ordinary LMS with a range of learning rates and the IDBD algorithm with a range of meta-learning rates. The $\beta_i$ in the IDBD algorithm were set initially such that $\alpha_i = 0.05$, for all $i$ (but of course this choice has no affect on asymptotic performance).

The results for both algorithms are summarized in figure 3. With its best learning rate, ordinary LMS attains a mean squared error of about 3.5, while the IDBD algorithm performs substantially better over a wide range of $\theta$ values, attaining a mean squared error of about 1.5. The standard errors of all of these means are less that 0.1, so this difference is highly statistically significant. The IDBD algorithm apparently learns biases (learning rates) that enable substantially more accurate tracking on this task.

### Exp. 2: Does IDBD find the optimal $\alpha_i$?

Experiment 1 shows that the IDBD algorithm finds a distribution of learning rates across inputs that is better than any single learning rate shared by all, but it does not show that it finds the best possible distribution of learning rates. While this may be difficult to show as a general result, it is relatively easy to confirm empirically for special cases. To do this for the task used in Experiment 1, I chose a small value for the

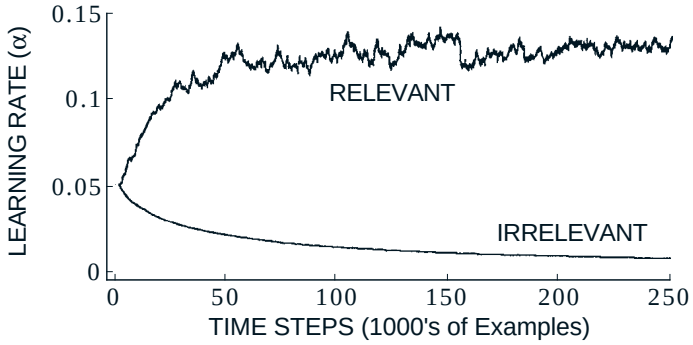


Figure 4: Time course of learning-rate parameters, under IDBD, for one relevant and one irrelevant input.

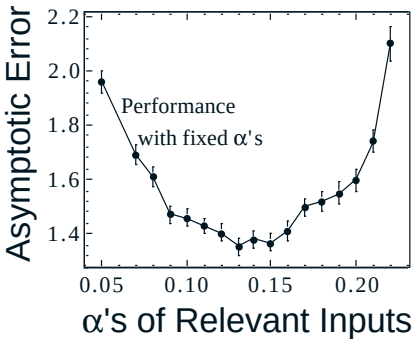


Figure 5: Average error as a function of the learning-rate parameter of the relevant inputs (the irrelevant inputs had zero learning-rate parameters). Error is minimized near $\alpha = 0.13$, the value found by the IDBD algorithm.

meta-learning rate, $\theta = 0.001$, and ran for a very large number of examples (250,000) to observe the asymptotic distribution of learning rates found by the algorithm (as before, the learning rates were initialized to 0.05). Figure 4 shows the behavior over time of two of the $\alpha_i$, one for a relevant input and one for an irrelevant input.

After 250,000 steps, the learning rates for the 15 irrelevant inputs were all found to be less than 0.007 while the learning rates for the 5 relevant inputs were all $0.13 \pm 0.015$. The learning rates for the irrelevant inputs were apparently heading towards zero (recall that they cannot be exactly zero unless $\beta_i = -\infty$), which is clearly optimal, but what of the relevant inputs? They should all share the same optimal learning rate, but is it $\approx 0.13$, as found by the IDBD algorithm, or is it some other value? We can determine this empirically simply by trying various sets of fixed learning rates. The irrelevant inputs were all given fixed zero learning rates and the relevant inputs were fixed at a variety of values between 0.05 and 0.25. Again I ran for 20,000 examples, to get past transients, and then recorded the average squared error over the next 10,000 examples. The results, plotted in figure 5, show a clear minimum somewhere near $0.13 \pm 0.01$, confirming that the IDBD algorithm found learning rates that were close to optimal on this task.

## Derivation of the IDBD Algorithm as Gradient Descent

Many useful learning algorithms can be understood as gradient descent, including the LMS rule, backpropagation, Boltzmann machines, and reinforcement learning methods. Gradient descent analysis can also be used to derive learning algorithms, and that in fact is the way in which the IDBD algorithm was invented. In this section I derive the IDBD algorithm as gradient descent.

To illustrate the basic idea of gradient-descent analysis, it is useful to first review how the base learning rule, the LMS rule (2), can be derived as gradient descent. Recall that we are trying to minimize the expected value of the squared error $\delta^2(t)$, where $\delta(t) = y^*(t) - y(t)$. The expected error as a function of the weights forms a surface. In gradient descent, a current value for $w$ is moved in the opposite direction of the slope of that surface. This is the direction in which the expected error falls most rapidly, the direction of steepest descent. Because the expected error is not itself known, we use instead the gradient of the sample error $\delta^2(t)$:

$$w_i(t+1) = w_i(t) - \frac{1}{2}\alpha \frac{\partial \delta^2(t)}{\partial w_i(t)}. \tag{7}$$

The scalar quantity $\frac{1}{2}\alpha$ is the step-size, determining how far the weight vector moves in the direction of steepest descent. The $\frac{1}{2}$ drops out as we expand the righthand side:

$$\begin{aligned} w_i(t+1) &= w_i(t) - \frac{1}{2}\alpha \frac{\partial \delta^2(t)}{\partial w_i(t)} \\ &= w_i(t) - \alpha\delta(t)\frac{\partial \delta(t)}{\partial w_i(t)} \\ &= w_i(t) - \alpha\delta(t)\frac{\partial [y^*(t) - y(t)]}{\partial w_i(t)} \\ &= w_i(t) + \alpha\delta(t)\frac{\partial y(t)}{\partial w_i(t)} \\ &= w_i(t) + \alpha\delta(t)\frac{\partial}{\partial w_i(t)}\Big[\sum_{j=1}^{n} w_j(t)x_j(t)\Big] \\ &= w_i(t) + \alpha\delta(t)x_i(t), \end{aligned} \tag{8}$$

thus deriving the LMS rule (2).

The derivation for the IDBD algorithm is very similar in principle. In place of (7), we start with

$$\beta_i(t+1) = \beta_i(t) - \frac{1}{2}\theta \frac{\partial \delta^2(t)}{\partial \beta_i}, \tag{9}$$

where now $\frac{1}{2}\theta$ is the (meta) step-size. In this equation, the partial derivitive with respect to $\beta_i$ without a time index should be interpretted as the derivative with repect to an infintesimal change in $\beta_i$ at all time steps. A similar technique is used in gradient-descent analyses of recurrent connectionist networks (c.f., e.g., Williams & Zipser 1989). We then similarly rewrite and expand (9) as:

$$\begin{aligned}\beta_i(t+1) &= \beta_i(t) - \frac{1}{2}\theta \sum_j \frac{\partial \delta^2(t)}{\partial w_j(t)} \frac{\partial w_j(t)}{\partial \beta_i} \\ &\approx \beta_i(t) - \frac{1}{2}\theta \frac{\partial \delta^2(t)}{\partial w_i(t)} \frac{\partial w_i(t)}{\partial \beta_i}. \qquad (10)\end{aligned}$$

The approximation above is reasonable in so far as the primary effect of changing the $i$th learning rate should be on the $i$th weight. Thus we assume $\frac{\partial w_j(t)}{\partial \beta_i} \approx 0$ for $i \neq j$. From (8) we know that $-\frac{1}{2}\frac{\partial \delta^2(t)}{\partial w_i(t)} = \delta(t)x_i(t)$; therefore we can rewrite (10) as

$$\beta_i(t+1) \approx \beta_i(t) + \theta\delta(t)x_i(t)h_i(t), \qquad (11)$$

where $h_i(t)$ is defined as $\frac{\partial w_i(t)}{\partial \beta_i}$. The update rule for $h_i$ is in turn derived as follows:

$$\begin{aligned}h_i(t+1) &= \frac{\partial w_i(t+1)}{\partial \beta_i} \\ &= \frac{\partial}{\partial \beta_i}\Big[w_i(t) + e^{\beta_i(t+1)}\delta(t)x_i(t)\Big] \qquad (12) \\ &= h_i(t) + e^{\beta_i(t+1)}\delta(t)x_i(t) + e^{\beta_i(t+1)}\frac{\partial \delta(t)}{\partial \beta_i}x_i(t),\end{aligned}$$

using the product rule of calculus. Using the same approximation as before (in (10)), we write

$$\begin{aligned}\frac{\partial \delta(t)}{\partial \beta_i} &= -\frac{\partial y(t)}{\partial \beta_i} = -\frac{\partial}{\partial \beta_i}\sum_j w_j(t)x_j(t) \\ &\approx -\frac{\partial}{\partial \beta_i}\Big[w_i(t)x_i(t)\Big] = -h_i(t)x_i(t).\end{aligned}$$

Finally, plugging this back into (12) yields

$$\begin{aligned}h_i(t+1) &\approx h_i(t) + e^{\beta_i(t+1)}\delta(t)x_i(t) - e^{\beta_i(t+1)}x_i^2(t)h_i(t) \\ &\approx h_i(t)\Big[1 - \alpha_i(t+1)x_i^2(t)\Big] + \alpha_i(t+1)\delta(t)x_i(t),\end{aligned}$$

which, after adding a positive-bounding operation, is the original update rule for $h_i$, (6), while the derived update (11) for $\beta_i$ is the same as (5).

The above demonstrates that the IDBD algorithm is a form of stochastic gradient descent in the learning-rate parameters $\beta_i$. In other words, the algorithm will tend to cause changes according to their effect on overall error. At local optima, the algorithm will tend to be stable; elsewhere, it will tend to reduce the expected error. These might be considered necessary properties for a good learning algorithm, but they alone are not sufficient. For example, there is the issue of step-size. It is often not recognized that the size of the step in the direction of the gradient may depend on the current value of the parameters being modified. Moreover, even the direction of the step can be changed, as long as it it is in the same general direction as the gradient (positive inner product), without losing these key properties.[2] For example, one could add a factor of $\alpha_i^p$, for any $p$, to the increment of $\beta_i$ in (9) to obtain an entire new family of algorithms. In fact, experiments in progress suggest that some members of this family may be more efficient than the IDBD algorithm at finding optimal learning rates. There is little reason beyond simplicity for prefering the IDBD algorithm over these other possibilities a priori.

The gradient analysis presented in this section tells us something about the IDBD algorithm, but it may also tell us something about incremental bias-learning algorithms in general. In particular, it suggests how one might derive bias-learning algorithms for other base learning methods, such as instance-based learning methods. In instance-based learning systems an important source of bias is the parameters of the inter-instance distance metric. Currently these parameters are established by people or by offline cross-validation methods. An interesting direction for further research would be to attempt to repeat the sort of derivation presented here, but for instance-based methods.

## Conclusion

The results presented in this paper provide evidence that the IDBD algorithm is able to distinguish relevant from irrelevant inputs and to find the optimal learning rates on incremental tracking tasks. Depending on the problem, such an ability to find appropriate biases can result in a dramatic reduction in error. In the tasks used here, for example, squared error was reduced by approximately 60%. The IDBD algorithm achieves this while requiring only a linear increase in memory and computation (both increase by roughly a factor of three over plain LMS). This algorithm extends prior work both because it is an incremental algorithm, operating on an example-by-example basis, and because it has fewer free parameters that must be picked by the user. Also presented in this paper is a derivation of the IDBD algorithm as gradient descent. This analysis refines previous analyses by improving certain approximations and by being applicable to incremental training.

On the other hand, only a linear version of the IDBD algorithm has been explored here. In addition, the results presented do not show that the IDBD algorithm is the *best* or *fastest* way to find optimal learning rates. Further work is needed to clarify these points.

A good way of understanding the IDBD algorithm may be as an incremental form of cross validation. Consider the form of cross validation in which one ex-

[2] Such algorithms are no longer *steepest*-descent algorithms, but they are still descent algorithms.

ample is held out, all the others are used for training, and then generalization is measured to the one held out. Typically, this is repeated $N$ times for a training set of size $N$, with a different example held out each time, and then the parameters are set (or stepped, following the gradient) to optimize generalization to the one held out, averaged over all $N$ cases. Obviously, this algorithm can not be done incrementally, but something similar can be. At each time step, one could take the new example as the one held out, and see how all the training on the prior examples generalized to the new one. One could then adjust parameters to improve the generalization, as the IDBD algorithm does, and thereby achieve an effect very similar to that of conventional cross validation. Such methods differ fundamentally from ordinary learning algorithms in that performance on the new example is optimized *without using the new example*.

The IDBD algorithm is being explored elsewhere as a psychological model. The base learning algorithm used here, the LMS rule, has often been used to model human and animal learning behavior. Although that modeling has been generally successful, there have also been a number of areas of divergence between model and emperiment. In many cases the discrepancies can be significantly reduced by augmenting the LMS model with relevance-learning methods (e.g., Kruschke 1992; Hurwitz 1990). The IDBD algorithm is also being explored in this regard, and the initial results are very encouraging (Gluck, Glauthier, & Sutton, in preparation; Gluck & Glauthier, in preparation; see also Sutton 1982).

One possible use of the IDBD algorithm is to assess the utility (relevance) of features created by constructive-induction methods or other representation-change methods. It is intriguing to think of using IDBD's assessments in some way to actually direct the feature-construction process.

Finally, a broad conclusion that I make from this work has to do with the importance of looking at a series of related tasks, such as here in a non-stationary tracking task, as opposed to conventional single learning tasks. Single learning tasks have certainly proved extremely useful, but they are also limited as ways of exploring important issues such as representation change and identification of relevant and irrelevant features. Such meta-learning issues may have only a small, second-order effect in a single learning task, but a very large effect in a continuing sequence of related learning tasks. Such cross-task learning may well be key to powerful human-level learning abilities.

## Acknowledgements

The author wishes to thank Mark Gluck, without whose prodding, interest, and assistance this paper would never have been written, and Oliver Selfridge, who originally suggested the general approach. I also thank Richard Yee, Glenn Iba, Hamid Benbrahim, Chris Matheus, Ming Tan, Nick Littlestone, Gregory Piatetsky, and Judy Franklin for reading and providing comments on an earlier draft of this paper.

## References

Barto, A.G. & Sutton, R.S. (1981) Adaptation of learning rate parameters, Appendix C of *Goal Seeking Components for Adaptive Intelligence: An Initial Assessment*. Air Force Wright Aeronautical Laboratories/Avionics Laboratory Technical Report AFWAL-TR-81-1070, Wright-Patterson AFB, Ohio.

Gluck, M.A. & Glauthier P.T. (in preparation) Representation of dimensional stimulus structure in network theories of associative learning.

Gluck, M.A., Glauthier, P.T., & Sutton, R.S. (in preparation) Dynamically modifiable stimulus associability in network models of category learning.

Hampson, S.E. & Volper, D.J. (1986) Linear function neurons: Structure and training. *Biological Cybernetics 53*, 203-217.

Hurwitz, J.B. (1990) A hidden-pattern unit network model of category learning. PhD thesis, Harvard Psychology Dept.

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

Kesten, H. (1958) Accelerated stochastic approximation. *Annals of Mathematical Statistics 29*, 41–59.

Kruschke, J.K. (1992) ALCOVE: An exemplar-based connectionist model of category learning. *Psychological Review*.

Lee, Y. & Lippmann, R.P. (1990) Practical characteristics of neural network and conventional pattern classifiers on artificial and speech problems. In *Advances in Neural Information Processing Systems 2*, D.S. Touretzky, Ed., 168–177.

Rendell, L.A., Seshu, R.M., and Tcheng, D.K. (1987) Layered concept learning and dynamically-variable bias management, *Proc. Tenth International Joint Conference on Artificial Intelligence*, 308-314.

Schlimmer, J.C. (1987) Concept acquisition through representation adjustment. PhD thesis, University of California, Information and Computer Science Dept., Technical Report 87-19.

Silva, F.M. & Almeida, L.B. (1990) Acceleration techniques for the backpropagation algorithm. In *Neural Networks: EURASIP Workshop 1990*, L.B. Almeida and C.J. Wellekens, Eds., 110-119. Berlin: Springer-Verlag.

Sutton, R.S. (1982) A theory of salience change dependent on the relationship between discrepancies on successive trials on which the stimulus is present. Unpublished working paper.

Sutton, R.S. (1986) Two problems with backpropagation and other steepest-descent learning procedures for networks. *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, 823–831.

Widrow, B. & Stearns, S.D. *Adaptive Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1985.

Williams, R.J. & Zipser, D. (1989) Experimental analysis of the real-time recurrent learning algorithm. *Connection Science 1*, 87–111.