December 1993

Explicit Forgetting Algorithms for Memory Based Learning

Marcos Salganicoff
University of Pennsylvania

University of Pennsylvania Institute for Research in Cognitive Science Technical Report No. IRCS-93-49.

Explicit Forgetting Algorithms for Memory Based Learning

Abstract
Memory-based learning algorithms lack a mechanism for tracking time-varying associative mappings. To widen their applicability, they must incorporate explicit forgetting algorithms to selectively delete observations. We describe Time-Weighted, Locally-Weighted and Performance-Error Weighted forgetting algorithms. These were evaluated with a Nearest-Neighbor Learner in a simple classification task. Locally-Weighted Forgetting outperformed Time-Weighted Forgetting under time-varying sampling distributions and mappings, and did equally well when only the mapping varied. Performance-Error forgetting tracked about as well as the other algorithms, but was superior since it permitted the Nearest-Neighbor learner to approach the Bayes' misclassification rate when the input-output mapping became stationary.

Comments
University of Pennsylvania Institute for Research in Cognitive Science Technical Report No. IRCS-93-49.

Explicit Forgetting Algorithms for Memory Based Learning

by

Marcos Salganicoff

University of Pennsylvania
Philadelphia, PA 19104-6228

December 1993

IRCS Report 93-49

Technical Report, University of Pennsylvania Dept. of Computer and Information Science, MS-CIS-93-80, September 7, 1993

Explicit Forgetting Algorithms for Memory Based Learning

Marcos Salganicoff
GRASP Laboratory
Department of Computer and Information Science
University of Pennsylvania
3401 Walnut Street, 301C
Philadelphia, PA 19104-6228, USA
email: sal@grip.cis.upenn.edu, phone: 215-898-0350

Abstract

Memory-based learning algorithms lack a mechanism for tracking time-varying associative mappings. To widen their applicability, they must incorporate explicit forgetting algorithms to selectively delete observations. We describe Time-Weighted, Locally-Weighted and Performance-Error Weighted forgetting algorithms. These were evaluated with a Nearest-Neighbor Learner in a simple classification task. Locally-Weighted Forgetting outperformed Time-Weighted Forgetting under time-varying sampling distributions and mappings, and did equally well when only the mapping varied. Performance- Error forgetting tracked about as well as the other algorithms, but was superior since it permitted the Nearest-Neighbor learner to approach the Bayes' misclassification rate when the input/output mapping became stationary.

Why Explicitly Forget?

Memory-based learning (MBL) algorithms are those that require that all presented input/output pairs be explicitly stored and available during the learning process. Some representative examples of MBL algorithms are Locally Weighted Regression [Atkeson, 1991], Nearest-Neighbor Classification [Aha *et al.*, 1991], CART [Breiman *et al.*, 1984], ID-3 [Quinlan, 1986], and Radial Basis Functions [Poggio and Girosi, 1990]. These algorithms present an alternative to on-line associative learning paradigms such as backpropagation [Rumelhart *et al.*, 1986] and Perceptrons [Minsky and Papert, 1988] that use incremental Δ-rules for weight updating.

One major advantage of on-line approaches is that when the input/output mapping to be learned changes over time, the Δ-rule will result in weight changes that track the mapping. MBL approaches store all observations and have no intrinsic mechanism for adapting to changing mappings [Schlimmer and Granger, 1986; Moore, 1991]. Therefore, to use MBL approaches **and** track the environment, we must have some algorithm for inactivating exemplars that are that are no longer representative of the current mapping (see Figure 1). We call this type of algorithm a "Forgetting" Algorithm[1]

In this work, three forgetting algorithms are described: Time-Weighted Forgetting (TWF) , Locally-Weighted Forgetting (LWF), and Performance Error Weighted (PEWF) forgetting. The TWF and LWF algorithms are finite-horizon, since they limit the total number of observations kept in memory, while the PEWF algorithm is not (see Figure 2). It is shown that when the forgetting algorithms are used along with a nearest-neighbor (NN) learning algorithm in a simple concept tracking task, LWF performs better than TWF under drifting sampling distributions and input/output mappings, and does as well as TWF when only mappings drift. PEWF is shown to track as well the others, but has the significant advantage that it allows the Bayes' misclassification rate to be approached if the input/output mapping stops drifting, under the assumption that learning algorithm used with it approaches the optimal Bayes' rate in static situations (see Figure 3). Therefore using PEWF forgetting is not detrimental to the asymptotic performance of classification algorithms in static situations, unlike the TWF and LWF finite-horizon methods. The practical result is that we can now track changing concepts with memory-based learners such as the Nearest-Neighbor algorithm, yet still approach Bayes' error rates whenever the associative mapping becomes stationary.

Description of the Forgetting Algorithms

Time weighted Forgetting

In Time-Weighted Forgetting, exemplars are weighted using a weighting function which

[1]Many Δ-rule learning algorithms are effectively run in "batch mode" where all of the exemplars are held in memory to speed weight adjustment; forgetting algorithms would also be useful in these cases.

assigns a weight w_i, (initialized to $w_i = 1$) to each observation. The updating function is a function of an observation's arrival time. It is common to recursively update weights to yield an exponential weighting function using the simple relation $w_{i,t+1} = \gamma w_{i,t}$, with $0 < \gamma \leq 1$, where $w_{i,t}$ is the weight associated with the ith observation at time step t. Often the system is clocked by the arrival of a new observation. When w_i goes below a threshold value θ, its corresponding exemplar is deleted.

Locally Weighted Forgetting

Experiences can be deleted based on the principle of locality of observations, which states that observations should be forgotten *only* if there is subsequent information in their locality of parameter space [Salganicoff, 1993]. This mechanism is implemented by associating a weight w to each observation. Each weight is decremented at a rate proportional to the number and proximity of succeeding exemplars to the corresponding observation.

We decrement each exemplar's weight by a factor γ, proportional to the proximity of a subsequent observation, based on a m-nearest neighborhood influence function. A new exemplar is initialized with a weight $w = 1$. Each time a new exemplar is input, the weightings $w_{\{k\}}$ of kth nearest observations, $X_{\{k\}}$, $(k \leq m)$, within a neighborhood of the m nearest-neighbors of the new exemplar X are decreased by multiplication with $\gamma(X_{\{k\}})$ which is a truncated quadratic function of the distance of the nearest neighbor to the query-point. When a given observation's weighting falls below some threshold value θ, it is deleted from the learning set.

Prediction Error Forgetting

Aha *et al.* [Aha *et al.*, 1991] have used the prediction accuracy criterion for the removal of noisy observation in nearest-neighbor classifiers. We extend their approach to handle time-varying mappings. Each observation is in either the *accepted* or *rejected* state. New observations are initialized with the state *accepted*, and only *accepted* observations are used to form predictions. Although an observation may be *rejected*, it is not deleted since it may be eventually accepted, or rejected and reaccepted in the case of cyclicly drifting concepts. Whenever a new prediction is made, each one of the m nearest-neighbors of the prediction point has it performance record updated. This record is an n bit long first-in last-out register containing a "1" for each of the times that exemplar agreed with the actual outcome of the query point (for a total of x "1"s), and a "0" otherwise. The $1 - \alpha$ performance accuracy probability interval for each of the m observations is computed using each observation's (x, n) statistics. If the observation is reflective of the current input/output mapping, its prediction accuracy over its last n participations will be high and it will be *accepted*, if it is obsolete, it will be low and it will be *rejected*.

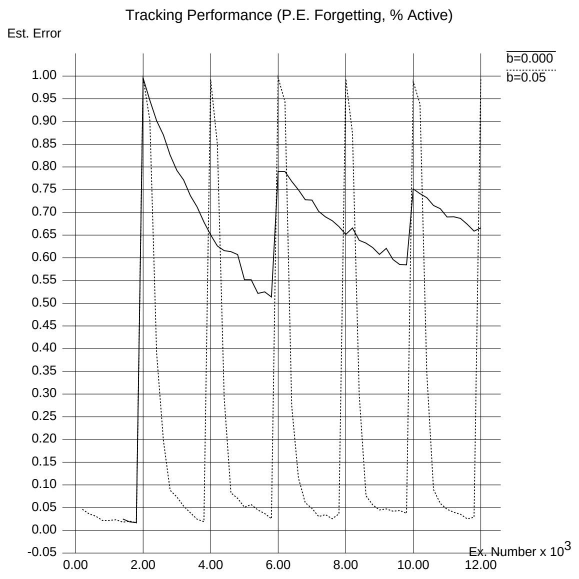
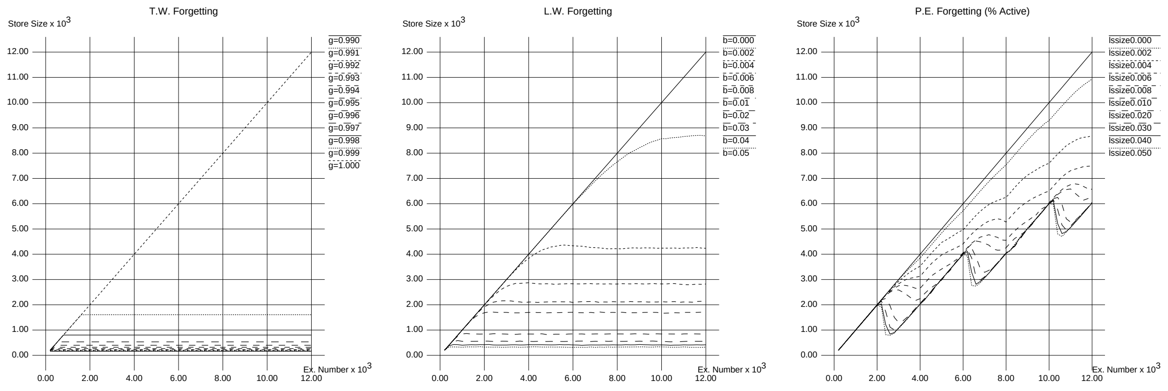
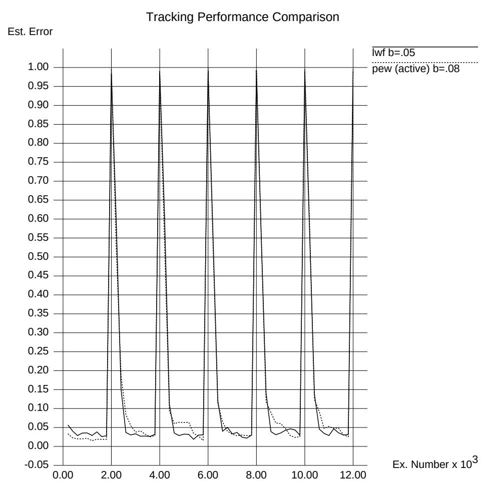


Figure 1: Tracking Switching Concepts. The plot of misclassification error for Nearest-Neighbor (NN) classifier for an input/ouput classification function which switches between two different functions every 2000 presentations. The condition $\beta = .000$ corresponds to using the NN algorithm with no forgetting, and it can be seen that learner cannot track to the changes in this case. Using PEWF forgetting with $m = \beta N = .05N$, where N is the number of currently active observations, the NN learner can track the change. (In the plots b=β)

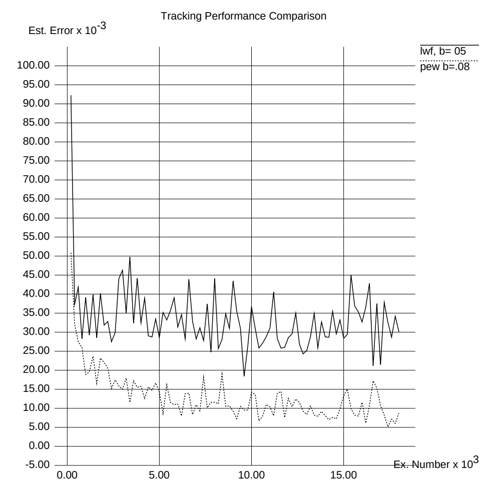


(a) (b) (c)

Figure 2: Comparison of Stored Sample Sizes. (a) Shows the number of samples held in memory as a function of the number of presentations for TWF (b) LWF and (c) for PEWF. It can be seen that PEWF's sample size is modulated at the switching points, while the TWF and LWF reach a finite asymptotic number of active examples in the store (in the plots b = β, g = γ.)



(a)



(b)

Figure 3: Comparison between asymptotic and dynamic behavior for LWF and PEW. It can be seen that the tracking abilities with the NN learner is similar in (a). However, in static mapping situations, the asymptotic error of the NN algorithm, when used with PEW tends towards the true NN asymptote, whereas LWF reaches inferior asymptotic value (in the plots b = β).

References

[Aha *et al.*, 1991] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[Atkeson, 1991] C. G. Atkeson. Using locally weighted regression for robot learning. In *IEEE Conference on Robotics and Automation*, pages 958–963, Sacremento, CA, IEEE Press, 1991.

[Breiman *et al.*, 1984] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and regression trees.* Wadsworth International, Belmont, CA, 1984.

[Minsky and Papert, 1988] M. Minsky and S.A. Papert. *Perceptrons.* MIT Press, Cambridge, MA, USA, 1988.

[Moore, 1991] A. W. Moore. Fast, robust adaptive control by learning only forward models. In *Advances in Neural Information Processing Systems*, Morgan Kauffman, December 1991.

[Poggio and Girosi, 1990] T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497, September 1990.

[Quinlan, 1986] J. R. Quinlan. *Learning efficient classification procedures and their application to chess end games*, pages 463–481. Morgan-Kauffman, Los Altos, Ca., 1986.

[Rumelhart *et al.*, 1986] D.E. Rumelhart, G.E. Hinton, and R. Williams. Learning internal representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[Salganicoff, 1993] M. Salganicoff. Density-adaptive learning and forgetting. In *Proceedings of the Tenth International Workshop on Machine Learning*, Amherst, MA, Morgan-Kauffman, June 1993.

[Schlimmer and Granger, 1986] J. C. Schlimmer and R. H. Granger. Beyond incremental processing: Tracking concept drift. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 502–507, Morgan Kauffman, Philadelphia, PA, 1986.