On-line EM with Weight-Based Forgetting

Enric Celaya
celaya@iri.upc.edu
Institut de Robòtica i Informàtica Industrial (CSIC-UPC), 08028 Barcelona, Spain.

Alejandro Agostini
aagosti@gwdg.de
Bernstein Center for Computational Neuroscience, 37077 Göttingen, Germany.

Keywords: On-line EM, forgetting factor, NGnet, local representations, biased sampling.

Abstract

In the on-line version of the EM algorithm introduced by Sato and Ishii (2000), a time-dependent discount factor is introduced for forgetting the effect of the old posterior values obtained with an earlier, inaccurate estimator. In their approach, forgetting is uniformly applied to the estimators of each mixture component depending exclusively on time, irrespective of the weight attributed to each unit for the observed sample. This causes an excessive forgetting in the less frequently sampled regions. To address this problem we propose a modification of the algorithm that involves a weight-dependent forgetting, different for each mixture component, in which old observations are forgotten according to the actual weight of the new samples used to replace older values. A comparison of the time-dependent versus the weight-dependent approach shows that the last one improves the accuracy of the approximation and exhibits a much greater stability.

1 Introduction

In this work, we consider the problem of incremental learning with a neural network for stochastic function approximation in a continuous domain. When facing this task in a real situation, often, the training samples are not randomly distributed, but they arrive as a succession of correlated inputs as, for example, when they are obtained while following a continuous trajectory through the input space. This turns out to be a form of sequential learning, characterized by the fact that different parts of the information are presented and learned by the system one after another, which gives rise to the well known phenomenon of *catastrophic interference* (McCloskey and Cohen, 1989), consisting in that previously learned information tends to be forgotten when new training samples are processed. The problem becomes even more serious when some regions of the input space are sampled much more often than others, a situation that easily appears when not all parts of the input space are equally accessible. In this case, the most visited regions tend to dominate the learning process, causing the system to forget what eventually had been learned in the less sampled regions.

Catastrophic interference is a direct consequence of the distributed nature of the representation. Completely local representations, like a look-up table, do not suffer from catastrophic interference, but lack the ability to generalize between inputs. Notwithstanding, generalization is necessary for approximating a function in a continuous domain, since it is not possible to sample all points, and their values must be inferred from the nearby ones. Semi-distributed representations consisting of locally-tuned units, whose response to a given input decreases with the distance to the unit's center, provide a trade-off between locality and generalization that helps to mitigate the problem of catastrophic interference. With a sufficiently local representation, for every given input, only a small number of units whose centers are close to the input will respond with activations significantly different from zero, and only those need to be evaluated and trained. Thus, each update has only a local effect, what prevents that learning in a given region can produce a change in what has been learned in regions far away from it.

Following these ideas, Moody and Darken (1989) introduced the Normalized Gaussian network (NGnet), a network architecture of locally-tuned units for learning real-

valued function approximations, and proposed a learning schema that combined self-organized and supervised learning, obtaining better results than with backpropagation. The NGnet is an instance of a Mixture-of-Experts (Jacobs et al., 1991), for which Xu et al. (1995) applied the maximum likelihood estimation method to determine their model parameters, and proposed a single-loop EM algorithm, which is efficient and well understood, but whose mode of working is in batch. Based on this work, Sato and Ishii (2000) developed an on-line version of the EM algorithm for the NGnet, allowing the adaptation to non-stationary environments by introducing a forgetting factor in the updating formula to progressively replace the old, inaccurate values by the new available ones. Despite the local nature of the NGnet units, the forgetting mechanism introduced by Sato and Ishii is not local; it produces a global forgetting effect that may impair the performance of the NGnet. To relieve this pitfall, we propose a local forgetting mechanism by which each unit only forgets old information in the measure that it is fed with new information. This avoids the degradation of the approximation that takes place in the infrequently sampled regions when learning is concentrated in other distant regions.

2 Function approximation with the Normalized Gaussian network

The Normalized Gaussian Network with linear regression units, as defined in (Sato and Ishii, 2000), maps N-dimensional input vectors x into D-dimensional output vectors y according to:

$$y = \sum_{i=1}^{M} \mathcal{N}_i(x) \tilde{W}_i \tilde{x}, \tag{1}$$

where M denotes the number of units, $\tilde{W}_i \tilde{x}$ is a short notation for $W_i x + b_i$, so that $\tilde{W}_i = (W_i, b_i)$ is a $(D \times (N+1))$-dimensional matrix and $\tilde{x}' = (x', 1)$, where the prime $(')$ denotes transpose, and $\mathcal{N}_i(x)$ is the normalized Gaussian function:

$$\mathcal{N}_i(x) \equiv G_i(x) / \sum_{j=1}^{M} G_j(x) \tag{2}$$

$$G_i(x) \equiv (2\pi)^{-N/2} |\Sigma_i|^{-1/2} \exp\left[-\frac{1}{2} (x - \mu_i)' \Sigma_i^{-1} (x - \mu_i) \right], \tag{3}$$

with μ_i and Σ_i corresponding, respectively, to the center and covariance matrix of the i-th unit.

According to Sato and Ishii (2000), we can define a stochastic model for the NGnet, in which (x, y) is an incomplete event, in the following way: first, a unit i is randomly selected form M equally probable units; then, an input vector x is randomly generated with the probability given by the Gaussian distribution $G_i(x)$, and finally, an output value y is generated from the Gaussian distribution

$$P(y|x, i, \theta) = (2\pi)^{-D/2}\sigma_i^{-D} \exp\left[-\frac{1}{2\sigma_i^2}(y - \tilde{W}_i\tilde{x})^2\right]. \tag{4}$$

This stochastic model provides the following probability distribution for a complete event:

$$P(x, y, i|\theta) = (2\pi)^{-\frac{D+N}{2}}\sigma_i^{-D}|\Sigma_i|^{-1/2}M^{-1} \exp\left[-\frac{1}{2}(x - \mu_i)'\Sigma_i^{-1}(x - \mu_i) - \frac{1}{2\sigma_i^2}(y - \tilde{W}_i\tilde{x})^2\right], \tag{5}$$

where $\theta = \{\mu_i, \Sigma_i, \tilde{W}_i, \sigma_i | i = 1 \dots M\}$ is a set of model parameters. It can be seen that the expected value of y for a given x, $E[y|x]$, for this model is precisely (1), so that (5) provides a stochastic model for the NGnet whose parameters can be adjusted to maximize the likelihood of a set of observations $\{(x(t), y(t)), t = 1 \dots T\}$ of an unknown stochastic function.

3 EM algorithm

The EM algorithm (Dempster et al., 1977) can be used to determine the parameters θ of the maximum likelihood model (5), where each unit $i(t)$ used to generate the observation $(x(t), y(t))$ is a hidden variable. The EM algorithm is initialized with a model defined by a set of parameters $\overline{\theta}$, which is progressively improved by alternating the E and M steps. We recall the results of (Sato and Ishii, 2000) for the case of the NGnet:

- E (Estimation) step

  Given the current estimator with parameters $\overline{\theta}$, compute, for each unit i, the posterior probability that i is selected to generate each observation $(x(t), y(t))$:

  $$P(i|x(t), y(t), \overline{\theta}) = P(x(t), y(t), i|\overline{\theta}) / \sum_{j=1}^{M} P(x(t), y(t), j|\overline{\theta}) \tag{6}$$

- M (Maximization) step

  Compute the parameters of the new estimator θ using the new values of $P(i|x(t), y(t), \overline{\theta})$:

$$\mu_i = \langle x \rangle_i^{[T]} / \langle 1 \rangle_i^{[T]} \tag{7}$$

$$\Sigma_i = \langle x x' \rangle_i^{[T]} / \langle 1 \rangle_i^{[T]} - \mu_i \mu_i' \tag{8}$$

$$\tilde{W}_i = \langle y \tilde{x}' \rangle_i^{[T]} \left[\langle \tilde{x} \tilde{x}' \rangle_i^{[T]} \right]^{-1} \tag{9}$$

$$\sigma_i^2 = \frac{1}{D} \left[\langle |y|^2 \rangle_i^{[T]} - \text{Tr}(\tilde{W}_i \langle \tilde{x} y' \rangle_i^{[T]}) \right] / \langle 1 \rangle_i^{[T]}, \tag{10}$$

  where the notation $\langle \cdot \rangle_i^{[T]}$ denotes a weighted sum defined as:

$$\langle f(x,y) \rangle_i^{[T]} \equiv \sum_{t=1}^{T} P(i|x(t), y(t), \overline{\theta}) f(x(t), y(t)). \tag{11}$$

Observe that in the original formulation of Sato and Ishii (2000), instead of the weighted sum (11), a weighted *mean* is used, which includes the factor $1/T$. However, since in all of the equations (7)-(10) in which such weighted means appear, these factors cancel out, we avoid introducing them in the first place.

The EM algorithm is a batch process, since all available data are processed in each E step.

4 Sato and Ishii's On-line EM algorithm

In the on-line version of the EM algorithm introduced in (Sato and Ishii, 2000), after each new observation $(x(t), y(t))$, an E step is performed for just this observation, and the M step is immediately performed to obtain a new estimator. In this case, a time-dependent discount factor (or forgetting factor) $\lambda_t \in [0, 1]$ is introduced to progressively diminish the influence of old estimated values which were obtained with earlier, inaccurate estimators. Since forgetting takes place in a temporal basis, we call the resulting algorithm Time-Based Forgetting EM (TBFEM). In this case, the weighted sum (11) is replaced by

$$\ll f(x,y) \gg_i^{[T]} \equiv \sum_{t=1}^{T} \left(\prod_{s=t+1}^{T} \lambda_s \right) P(i|x(t), y(t), \theta(t-1)) f(x(t), y(t)). \tag{12}$$

As before, the original formulation introduced in (Sato and Ishii, 2000) defines a weighted mean instead of a weighted sum, and includes a normalization factor $\eta(T)$ that we ignore here since it is unnecessary, and thus, computations are simplified.

The weighted sum (12) at time t can be obtained from the weighted sum at time $(t-1)$ with the step-wise update equation:

$$\ll f(x,y) \gg_i^{[t]} = \lambda_t \ll f(x,y) \gg_i^{[t-1]} + P(i|x(t), y(t), \theta(t-1)) f(x(t), y(t)). \quad (13)$$

The forgetting factor must be taken such that $\lambda_t \to 1$ when $t \to \infty$ and fulfill the Robbins-Monro conditions for convergence of stochastic approximations (Kushner and Yin, 1997), which, according to Sato and Ishii (2000), can be accomplished with a choice of the form

$$\lambda_t = 1 - \frac{1-a}{at+b}, \quad (0 < a < 1). \quad (14)$$

In summary, the TBFEM algorithm must compute, after each new observation, the weighted sums $\ll 1 \gg_i^{[t]}$, $\ll x \gg_i^{[t]}$, $\ll xx' \gg_i^{[t]}$, $\ll |y|^2 \gg_i^{[t]}$, and $\ll y\tilde{x}' \gg_i^{[t]}$, from which the new estimator $\theta(t)$ is obtained using equations (7)-(10) in which the $\langle \cdot \rangle_i^{[T]}$ operators are replaced by $\ll \cdot \gg_i^{[t]}$. For the algorithm to work properly, it is necessary that the covariance matrices Σ_i are invertible but in practice this often fails, so that it is necessary to introduce a regularization method to prevent that they become singular. Details can be found in (Sato and Ishii, 2000).

Additionally, Sato and Ishii (2000) propose a number of unit manipulation mechanisms, namely, unit production, unit deletion, and unit division, to dynamically adapt the number of units of the model to the data. Since these mechanisms are independent of the updating method used, we will not discuss them here.

5 On-line EM with weight-based forgetting

As said before, the purpose of the forgetting factor λ_t is to progressively replace the earlier, inaccurate values included in the weighted sums, by the newer, more reliable ones. However, we note that, while the forgetting mechanism of equation (13) reduces all the sums of past samples by the same proportion λ_t for all units at each time step, the increments provided by a new sample is determined by its weight $P(i|x(t), y(t), \theta(t-1))$, which is different for each unit and, typically, is only significantly different from zero for a few units. This has the undesirable effect that the weighted sums of all units tend to decrease with time, and specially those corresponding to the less frequently sampled regions. As a consequence, if a unit is not updated with nearby inputs often

enough, their weighted sums will decrease to 0 too fast, before having the opportunity to increase their values with new data. If, eventually, a nearby sample is used to update one such unit, its relatively large weight will make the new value to prevail and dominate the sum, so that the information of all previous observations will be essentially lost.

To solve this problem, we introduce a weight-based forgetting EM algorithm (WBFEM) in which forgetting does not take place uniformly for all units irrespective of how much they are updated but, instead, forgetting is specific for each unit, depending on the weight used to update it with new values. Next, we derive a precise update formula to achieve this.

Assume that, at a given time, the weighted sum of some function $f(x, y)$ of a given unit has been updated with a total accumulated weight ω_0. We denote by $\ll f \gg^{[\omega_0]}$ the current value of this weighted sum to make explicit its dependence on weight, rather than on time. If a new value $f_t = f(x(t), y(t))$ arrives, and $\omega = P(i|x(t), y(t), \theta(t-1))$ is the weight corresponding to this unit, the weighted sum will be updated as:

$$\ll f \gg^{[\omega_0+\omega]} = \Lambda(\omega) \ll f \gg^{[\omega_0]} + \Omega(\omega) f_t. \tag{15}$$

Equation (15) is similar to (13) except for the new factors $\Lambda(\omega)$ and $\Omega(\omega)$ multiplying $\ll f \gg^{[\omega_0]}$ and f_t, respectively, which are unknown functions of ω that we are about to determine. Our purpose is that (15) reduces to (13) when performing a full update of the unit (i.e., when $\omega = 1$ for this unit), but that the weighted sum remains unaltered when $\omega = 0$. In addition, the updating must be consistent for values of $\omega \in [0, 1]$, in the sense that the forgetting must be always the same for a given amount of new weight, no matter in how many update operations it takes place.

Imposing that (15) reduces to (13) for a full update, gives:

$$\ll f \gg^{[\omega_0+1]} = \Lambda(1) \ll f \gg^{[\omega_0]} + \Omega(1) f_t = \lambda_t \ll f \gg^{[\omega_0]} + f_t. \tag{16}$$

from what we get:

$$\Lambda(1) = \lambda_t, \tag{17}$$

$$\Omega(1) = 1. \tag{18}$$

Now, to fulfill the consistency condition, we impose that the result of updating the weighted sum twice with a value f_t and respective weights ω_1 and ω_2 must be the same

as updating it once with the value f_t and weight $(\omega_1 + \omega_2)$, what reduces to the natural condition $\ll f \gg^{[\omega_0+(\omega_1+\omega_2)]} = \ll f \gg^{[(\omega_0+\omega_1)+\omega_2)]}$. Thus we have:

$$\ll f \gg^{[\omega_0+(\omega_1+\omega_2)]} = \Lambda(\omega_1+\omega_2) \ll f \gg^{[\omega_0]} + \Omega(\omega_1+\omega_2) f_t, \tag{19}$$

$$\ll f \gg^{[(\omega_0+\omega_1)+\omega_2)]} = \Lambda(\omega_2) \ll f \gg^{[\omega_0+\omega_1]} + \Omega(\omega_2) f_t. \tag{20}$$

Using (15) in the right-hand side of (20), and equating it to (19) we get:

$$\begin{aligned} &\Lambda(\omega_1+\omega_2) \ll f \gg^{[\omega_0]} + \Omega(\omega_1+\omega_2) f_t = \\ &\Lambda(\omega_2)\Lambda(\omega_1) \ll f \gg^{[\omega_0]} + (\Lambda(\omega_2)\Omega(\omega_1) + \Omega(\omega_2)) f_t. \end{aligned} \tag{21}$$

Since this equation must hold for any arbitrary value f_t, we obtain:

$$\Lambda(\omega_1+\omega_2) = \Lambda(\omega_1)\Lambda(\omega_2), \tag{22}$$

$$\Omega(\omega_1+\omega_2) = \Lambda(\omega_2)\Omega(\omega_1) + \Omega(\omega_2). \tag{23}$$

Equations (22) and (23) are *functional equations* for $\Lambda(\omega)$ and $\Omega(\omega)$, respectively. The solution of the functional equation (22) is well known to be of the form $\Lambda(\omega) = a^\omega$, and the value of a can be determined using (17):

$$\Lambda(1) = a = \lambda_t, \tag{24}$$

therefore we have

$$\Lambda(\omega) = \lambda_t^\omega. \tag{25}$$

To solve the functional equation (23), we write it for $\omega_1 = \omega$ and $\omega_2 = (1-\omega)$, and using (25) and (18), we get:

$$\Omega(1) = \Omega(\omega + (1-\omega)) = \lambda_t^{1-\omega}\Omega(\omega) + \Omega(1-\omega) = 1, \tag{26}$$

and similarly:

$$\Omega(1) = \Omega((1-\omega)+\omega) = \lambda_t^\omega \Omega(1-\omega) + \Omega(\omega) = 1. \tag{27}$$

Equations (26) and (27) form a system of two equations with the two unknowns $\Omega(\omega)$ and $\Omega(1-\omega)$. Solving this system, we obtain the expression for the sought function:

$$\Omega(\omega) = \frac{1-\lambda_t^\omega}{1-\lambda_t}. \tag{28}$$

Having determined the two functions $\Lambda(\omega)$ and $\Omega(\omega)$ we can write the resulting updating formula (15) as:

$$\ll f \gg^{[\omega_0+\omega]} = \lambda_t^{\omega} \ll f \gg^{[\omega_0]} + \frac{1-\lambda_t^{\omega}}{1-\lambda_t} f_t. \tag{29}$$

Now, returning to the notation of (13) with the bracketed superindex denoting the time step, the updating formula of WBFEM is:

$$\ll f(x,y) \gg_i^{[t]} = \lambda_t^{\omega} \ll f(x,y) \gg_i^{[t-1]} + \frac{1-\lambda_t^{\omega}}{1-\lambda_t} f(x(t), y(t)), \tag{30}$$

where $\omega = P(i|x(t), y(t), \theta(t-1))$.

6 Experiments

In order to evaluate the proposed algorithm, we compared the performances of TBFEM and WBFEM for a wide range of parameters a and b of Eq. (14) using different functions with input dimensions $N = 1$ and $N = 2$, leading to similar results in all cases. The results presented next are for the function used in (Sato and Ishii, 2000) with input dimension $N = 2$ defined by:

$$y = g(x_1, x_2) = max\{e^{-10x_1^2}, e^{-50x_1^2}, 1.25e^{-5(x_1^2+x_2^2)}\} \quad (-1 \leq x_1, x_2 \leq 1). \tag{31}$$

In all experiments, we depart from two identically initialized NGnets with 25 units centered at points regularly distributed in the input domain, and each observed sample is used to update each model with the corresponding algorithm. Noisy samples are generated on-line so that, for each new input vector $x(t) = (x_1(t), x_2(t))$, a sample $(x(t), y(t))$ is obtained with $y(t) = g(x(t)) + 0.1\varepsilon(t)$, where $g(x(t))$ is the function to approximate and $\varepsilon(t)$ is a random noise with standard normal distribution $\mathcal{N}(0,1)$. To have a clean comparison between just the two updating methods, we keep the number of units fixed throughout the process, i.e., we do not apply any of the unit manipulation mechanisms proposed in (Sato and Ishii, 2000). The accuracy achieved by each approximation is measured every 100 update iterations by computing the MSE of the current model using 441 points forming a regular grid in the input domain. Each experiment is evaluated by computing the mean of the MSE values computed within the last $5,000$ iterations. In the first set of experiments, training samples are generated on-line

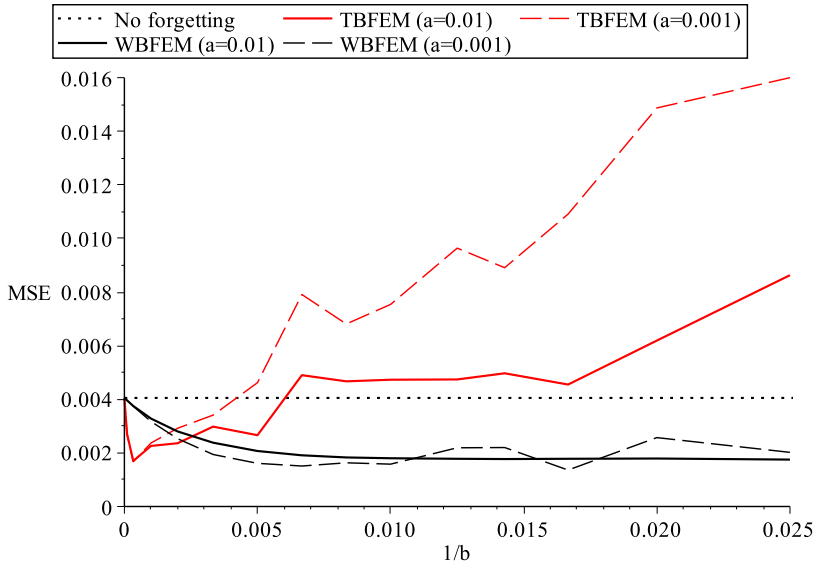

Figure 1: Dependence of the MSE on parameters a and b (50,000 iterations).

by selecting random vectors $x(t) = (x_1(t), x_2(t))$ uniformly distributed in the input domain. Figure 1 shows the variation with parameter b of the mean MSE reached by each algorithm after $50,000$ iterations. Results are given for two values of parameter $a = \{0.01, 0.001\}$. When the forgetting mechanism is not used, i.e., when making $\lambda_t = 1$, there is obviously no difference between WBFEM and TBFEM, and this case will be taken as the reference for evaluating the performance of both algorithms. In the figures, this case corresponds to $1/b = 0$.

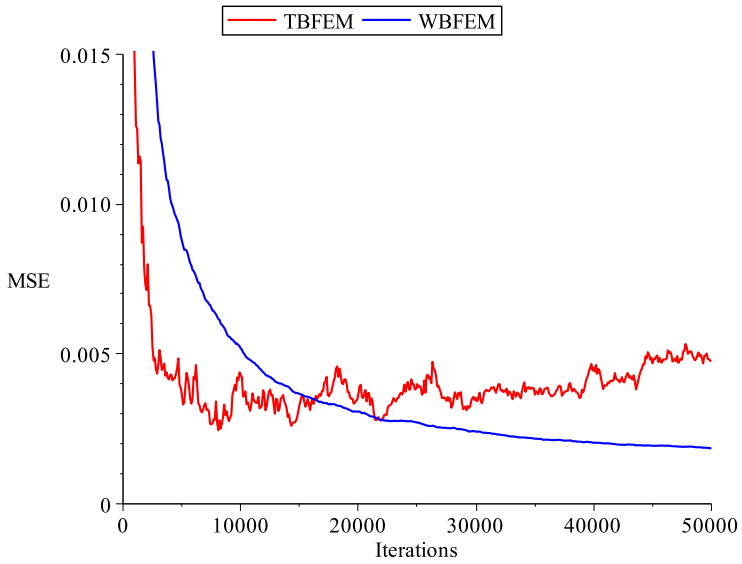

Figure 2: Performance comparison for $a = 0.01$ and $b = 150$.

In the case of the original TBFEM, increasing the parameter $1/b$ by small values results in a progressive improvement of performance, but as $1/b$ grows beyond a certain value the performance begins to degrade and soon becomes much worse than that of $\lambda_t = 1$. This behavior is accentuated as parameter a gets smaller. In contrast, the proposed WBFEM keeps improving its performance for a wide range of $1/b$ values, showing a more robust behavior and a far lesser sensitivity to the selection of parameters a and b. A comparison of the optimal setting for each algorithm gives a mean MSE of 0.00167 for TBFEM with $a = 0.001$ and $b = 3,000$, and a mean MSE of 0.00136 for WBFEM with $a = 0.001$ and $b = 60$. The typical evolution of the MSE of the approxi-

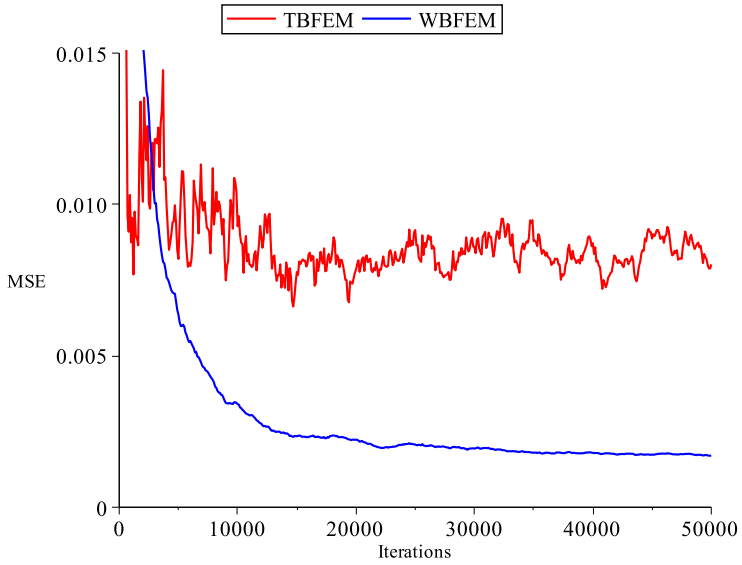

Figure 3: Performance comparison for $a = 0.01$ and $b = 40$.

mation is exemplified in Figure 2, corresponding to parameters $a = 0.01$ and $b = 150$. It can be seen that the original TBFEM algorithm converges to a suboptimal approximation after about $7,500$ iterations (MSE$= 0.00265$), after what the mean MSE starts a progressive increase with a fluctuation of an amplitude that decreases as the value of λ_t approaches 1, reaching a final mean MSE of 0.00490 with a mean deviation of 1.15×10^{-4} for the last $5,000$ iterations. The behavior of WBFEM is much more stable and, while converging slower at the beginning, it keeps improving all along the duration of the experiment, surpassing the accuracy of TBFEM at about $17,000$ iterations, to reach a final mean MSE of 0.00191, with a much lower amplitude of its fluctuation: 2.19×10^{-5} of mean deviation for the last $5,000$ iterations.

The slower convergence and increased stability of WBFEM can be explained by the fact that, for a given value of λ_t, the relative contribution of the new data in the current estimations is always lesser than in TBFEM, what has an effect similar to using a smaller learning rate. A faster convergence can be obtained by increasing the influence of the new data, what can be achieved by decreasing the value of b. Thus, in Figure 3, we can see that setting the parameter $b = 40$, the WBFEM algorithm approaches the convergence value faster, and becomes more accurate than TBFEM after only about $3,000$ iterations with a final mean MSE of 0.00175 (mean deviation = 1.74×10^{-5}). The TBFEM algorithm becomes worse for this setting with a final mean MSE of 0.00863 (mean deviation = 3.09×10^{-4}).

6.1 Biased sampling

The purpose of the local forgetting mechanism we have introduced is to avoid that the values learned in the seldom sampled regions are forgotten when a large number of updates is performed in other regions. To see the effectiveness of the approach in this situation, we designed an experiment to test the performance of the algorithm when the update frequency is much higher in some regions of the domain than in others. To this end, a series of experiment was performed in which 95% of the samples were

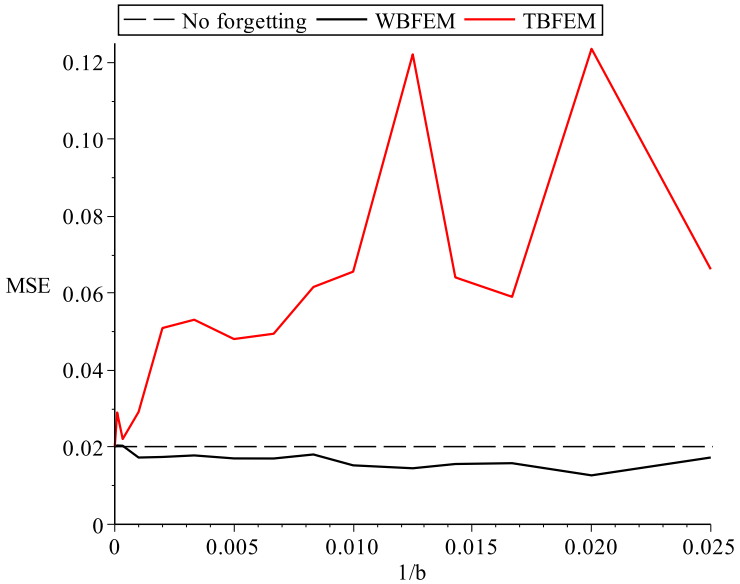

Figure 4: Performance of the two algorithms with $a = 0.01$ with biased sampling.

generated in a sub-region of the input domain with values of the variables x_1 and x_2 in the interval $[0, 0.25]$. Such unbalanced distribution makes the approximation task much

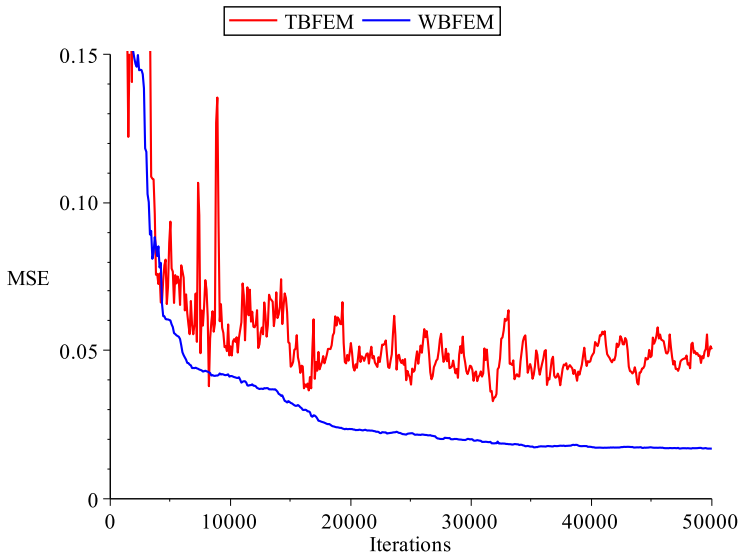

Figure 5: Performance comparison under biased sampling for $a = 0.01$ and $b = 150$.

more difficult for both algorithms, but the performance loss is comparatively lower for the proposed WBFEM algorithm than for the original TBFEM. Figure 4 compares the performance of both algorithms for different values of b with $a = 0.01$. In this case, TBFEM behaves worse than the case $\lambda_t = 1$ even for the largest value of b we tested ($b = 10,000$), and its performance gets worse as b decreases. In contrast, WBFEM improves its performance for a wide range of values of b, giving better results than $\lambda_t = 1$ for all values of $b < 3,000$ we have tested.

Figure 5 shows the evolution of the MSE of the approximation achieved by both algorithms with parameters $a = 0.01$ and $b = 150$. The final mean MSE values in this case are 0.04951 (mean deviation = 0.00289) for TBFEM and 0.01710 (mean deviation = 7.75×10^{-5}) for WBFEM. Figure 6 shows the receptive fields (corresponding to one standard deviation) of the units at the end of the experiment. It can be seen that TBFEM concentrated almost all units in the strongly sampled region, while WBFEM was able to keep a much better distribution of the units covering the whole domain, what explains its better performance.

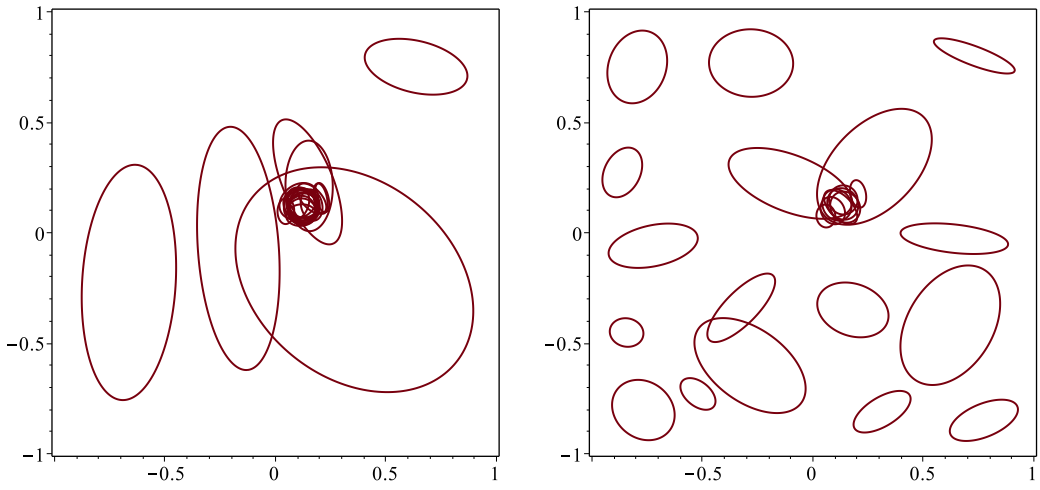

Figure 6: Receptive fields of the units generated by TBFEM (left) and WBFEM (right) with biased sampling.

6.2 Dynamic environments

An experiment proposed in (Sato and Ishii, 2000) was designed to test the effectiveness of the algorithm when dealing with dynamic environments. The experiment consisted in changing the distribution of the input variable x_1 with time, starting from the interval $[-1, -0.2]$ and progressively changing it by small displacements until reaching $[0.2, 1]$ after 250,000 iterations. Depending on the purpose of our learning system, two different goals can be pursued: one would consist in approximating the function as well as possible in the current input sub-domain at each given time; the second would be to achieve the best approximation possible of the whole function in the entire domain. In the first case, information obtained outside the current sub-domain is no useful anymore and can be safely forgotten, so that, for this purpose, TBFEM could be advantageous. In the second case, information obtained in the past should be kept as intact as possible even if it is never seen again, no matter how many new samples from other regions are processed. This is clearly a task for which WBFEM is more appropriate.

To test the performance of the two algorithms in each task, we conducted an experiment with the dynamic environment just described. Given the dynamic nature of the problem and the fixed duration of the experiment, we found more sensible to keep the forgetting factor λ_t constant all along the experiment, since we are not pursuing the long term convergence of the approximation to a stationary point. Figures 7 and 8

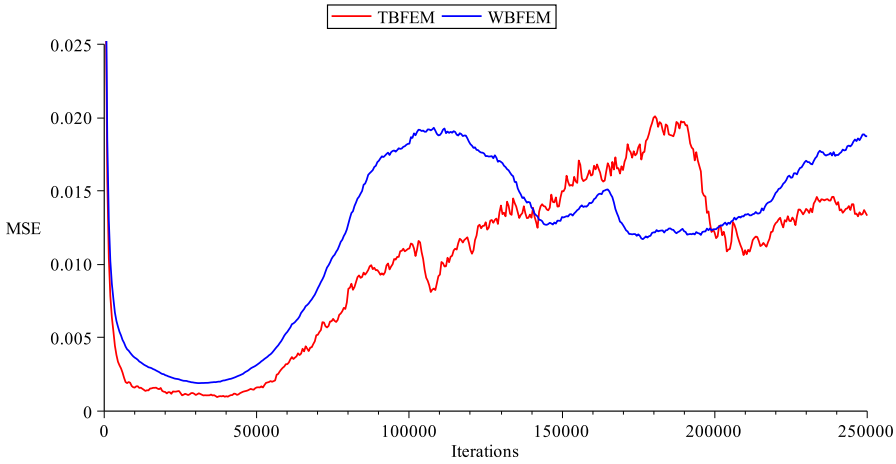

Figure 7: Dynamic environment: evolution of MSE in the current window.

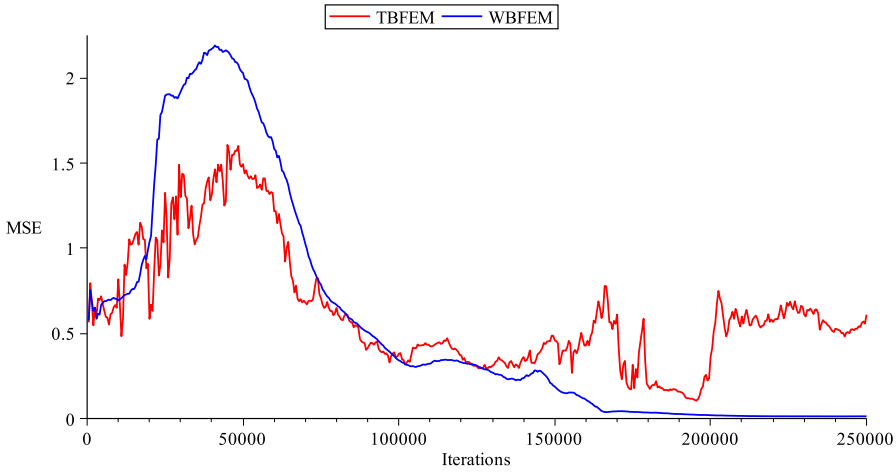

Figure 8: Dynamic environment: evolution of MSE in the whole input domain.

show the results obtained for $\lambda_t = 0.999$ or, equivalently, $a = 0$ and $b = 1,000$ with the evaluation of the MSE in the current window and in the whole input domain, respectively. In the first case, the approximations provided by both algorithms are fluctuating and it would be difficult to say which algorithm provides the best results. In the second case, however, we can see that, at the initial stages, the MSE is high for both algorithms, since most of the function domain has not been observed yet. As a larger part of the domain gets sampled, the approximations become more precise. However, the accuracy reached by TBFEM is limited, staying at a mean MSE of 0.5452 (mean deviation 0.0212), while WBFEM keeps improving it until the final stage to reach a mean MSE of 0.01294 (mean deviation 7.25×10^{-5}) for the whole domain. To further appreciate the difference between the behavior of both algorithms in this experiment, we show in

Figure 9, the domains of the units generated by each algorithm at the end of the process. As can be observed, TBFEM has all units concentrated in the final sampled sub-region, while WBFEM keeps a distribution that covers the whole domain much better.

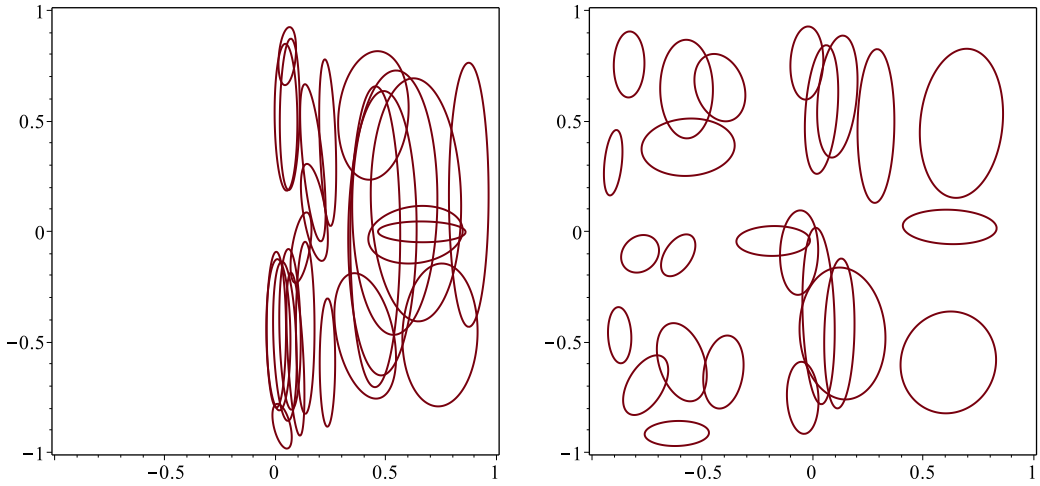

Figure 9: Receptive fields generated by TBFEM (left) and WBFEM (right).

7 Conclusions

We have shown that the forgetting mechanism used in the on-line version of the EM algorithm of Sato and Ishii (2000) has a global nature which is in conflict with the intended locality of the NGnet. By making the forgetting in each unit dependent on the weight of the new information added to this unit, instead of on the number of updates, the locality of the updating is better preserved, preventing the information acquired in the less frequently sampled regions to fade away with time. The experimental results in stochastic function approximation show that the proposed algorithm provides better accuracy, is more stable, and is far less sensitive to the selection of the parameters controlling the forgetting factor.

Although the weight-dependent forgetting has been introduced here for the NGnet, its applicability is valid in general for any network architecture using an on-line EM algorithm. A preliminary version of the weight-based forgetting EM has been successfully applied on a mixture of Gaussians to learn the Q-function in a reinforcement learning problem (Agostini and Celaya, 2010).

References

Agostini, A.G. and Celaya, E. (2010). Reinforcement learning with a Gaussian mixture model. In *Proc. International Joint Conference on Neural Networks (IJCNN 2010)*, pp. 3485–3492.

Dempster, A.P., Laird, N.M., Rubin, D.B., et al. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1), 1–38.

Jacobs, R.A., Jordan, M.I., Nowlan, S.J., and Hinton, G.E. (1991). Adaptive mixtures of local experts. *Neural Computation*, 3(1), 79–87.

Kushner, H.J and Yin, G.G.. (1997). Stochastic Approximation Algorithms and Applications. New York: Springer-Verlag.

McCloskey, M. and Cohen, N. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. In G. H. Bower, editor, *The Psychology of Learning and Motivation*, (pp. 109–164), Academic Press.

Moody, J. and Darken, C.J. (1989). Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1(2), 281–294.

Sato, M.A., and Ishii, S. (2000). On-line em algorithm for the normalized gaussian network. *Neural Computation*, 12(2), 407–432.

Xu, L., Jordan, M., and Hinton, G. (1995). An alternative model for mixtures of experts. In J.D. Cowan, G. Tesauro, and J. Alspector, editors, *in Advances in Neural Information Processing Systems 7*, pp. 633–640, MIT Press.