UNIVERSIDADE DE SÃO PAULO

INSTITUTO DE FÍSICA
CAIXA POSTAL 20516
01498-970 SÃO PAULO - SP
BRASIL

PUBLICAÇÕES

IFUSP/P-1044

LOWER BOUNDS ON GENERALIZATION ERRORS FOR DRIFTING RULES

Osame Kinouchi and **Nestor Caticha**
Instituto de Física, Universidade de São Paulo

Março/1993

LOWER BOUNDS ON GENERALIZATION ERRORS FOR DRIFTING RULES

March 9, 1993

Osame Kinouchi and Nestor Caticha

Instituto de Física, Universidade de São Paulo,

CP 20516, 01498 São Paulo, SP, Brazil

Abstract

The problem of generalization by single layer perceptrons is studied in the case of time dependent rules. Lower bounds for the generalization errors within the 'single presentation of examples' case are obtained for randomly drifting rules. These bounds suggest a learning algorithm which uses knowledge of the error itself. Since this error is not readily available it has to be estimated through a mechanism of self-evaluation. The capacity of incorporating recency information into the error estimate is highly desirable. The mechanism proposed has the advantage, beyond good performance, of being self-adaptive, in the sense that it adjusts to changes in the unknown drift rate of the rule. The performance is also studied for sudden changes of the rule in an attempt to mimic the so-called Wisconsin test.

PACS : 87.10.e+10, 75.10 Hk, 64.60 Cn, 89.70.+c

1 Introduction

In an everchanging environment the ability of organisms to present adaptive behaviour might be an important factor to determine their chances of survival. Thriving under a randomly varying set of rules will depend on the capacity of incorporating useful information from the outside world. Such adaptive behaviour can be modelled within supervised learning in Neural Nets. Perceptron learning of a time dependent rule has been recently studied by Biehl and Schwarze [1] in the case where there is a single presentation of the examples. We discuss again this type of problem with the aim of determining what is the optimal way in which the information from the changing environment ("professor" or "rule") can be used by a single layer perceptron in order to maximize its generalization ability.

The natural approach to this problem is "online" learning. This means that the examples are used only once and not repeatedly ("iterated learning") until some error measure over all examples is minimized. Iterated learning might not be efficient if old examples are presented again at a later time when they are not any longer representative of the state of the rule. Furthermore, single presentation might also be quite efficient, since in the optimal case (Expected Stability algorithm) for a static rule it leads [2] to only exactly twice the error of the Bayes limit of Opper and Haussler [3], with just a small fraction of the computational effort.

In section 2 the model [1] and notation are introduced. For a randomly drifting rule we obtain in section 3 that, analogously to the time independent case, suprises carry a high information content, and should be attributed a high weight, while to obvious facts a very small importance must be given. More especifically we obtain an expression for the weight a Hebbian term must have in order to maximize generalization. Examples "easy" to classify by the student net are those with a high local field at the perceptron's

output unit. When an "easy" example is misclassified it receives a very high weight and a very small one when it is correctly classified. Also in section 3 we present lower bounds for the generalization error as a function of T, where T is a measure of the amplitude of the white noise driving the rule. Calling, as usual, α the ratio of the number of patterns to the number of input units, we obtain that, for $\alpha \to \infty$ and small T, the saturation generalization error for optimal learning grows as $T^{1/3}$, while being proportional to $T^{1/4}$ in the optimal pure Hebbian case [1]. The strategy of learning by "queries" ("selection of examples") [4, 10] is also analized.

These bounds can be reached by a learning algorithm which uses the value of the generalization error itself. Since this is not a very realistic feature we propose in section 4 a new self-adaptive algorithm which relies on the estimate of the generalization error, as judged by recent success. Several estimators can be considered and we discuss the effects on learning of a parameter which measures the persistence of old examples in the actual estimator. Results of simulations are presented for the case of slowly drifting rule as well as for the so called Wisconsin test [5], where the rule is piecewise constant in time. Section 5 has some concluding remarks .

2 The model

We now examine the problem of learning a time dependent linearly separable rule within an exactly solvable model. A single layer, single output unit perceptron, the "student", learns from examples generated by another such perceptron, the "professor", whose synaptic weights change randomly in time. Let N be the number of inputs, $\mathbf{S} \in (-1,1)^{\mathbf{N}}$ and $\mathbf{B} \in \mathbb{R}^N$ or $(-1/\sqrt{N}, 1/\sqrt{N})^N$ the input vector and the professor

synaptic coupling vector respectively. The correct (professor) output is

$$\sigma_B^\mu = sign(\sum_{i=1}^{N} B_i S_i^\mu). \tag{1}$$

The student is described by a synaptic vector $\mathbf{J} \in \mathbb{R}^N$ and its output is

$$\sigma_J^\mu = sign(\sum_{i=1}^{N} J_i S_i^\mu). \tag{2}$$

The generalization ability is defined as the probabilty that at a given time the outputs of both the professor and student nets be equal. The synaptic vector $\mathbf{J}$ is to be constructed from the information contained in the examples, that is, a sequence of pairs $(\mathbf{S}^\mu, \sigma_B^\mu)$, where μ is the time label or the example position in the sequence.

We first consider, as Biehl and Schwarze [1], a time evolution of the rule

$$\mathbf{B}^\mu = \mathbf{B}^{\mu-1} + \frac{1}{\sqrt{N}}\,\eta^\mu, \tag{3}$$

where η^μ is a random vector satisfying

$$\langle \eta_i^\mu \eta_j^\nu \rangle = \frac{2}{N}\,T\,\delta_{ij}\,\delta_{\mu\nu}, \tag{4}$$

and we keep the professor norm B^μ constant equal to 1 by imposing that

$$\sqrt{N}\,\mathbf{B}^{\mu-1}.\,\eta^\mu = -\frac{1}{2}\,\eta^\mu.\eta^\mu = -T. \tag{5}$$

Thus

$$\mathbf{B}^\mu.\,\mathbf{B}^{\mu-1} = 1 - \frac{T}{N}, \tag{6}$$

where T is the drift parameter. As usual the overlap between $\mathbf{B}$ and $\mathbf{J}$ is the relevant quantity [6, 10],

$$\rho^\mu \equiv \frac{R^\mu}{J^\mu}, \quad R^\mu \equiv \mathbf{B}^\mu.\,\mathbf{J}^\mu, \quad J^\mu \equiv \sqrt{\mathbf{J}^\mu.\,\mathbf{J}^\mu}, \tag{7}$$

since the generalization ability G, the generalization error e_G and ρ are related through:

$$e_G = 1 - G = \frac{1}{\pi}\arccos\rho \tag{8}$$

where self average has been assumed. We will use the notation

$$b_\mu \equiv \frac{\mathbf{B}^{\mu-1}.\,\mathbf{S}^\mu}{B^{\mu-1}}, \qquad \sigma_B^\mu = sign(b_\mu), \tag{9}$$

for the local field and output of the teacher for the μ^{th} example. Our notation is such that the professor changes from $\mathbf{B}^{\mu-1}$ to $\mathbf{B}^\mu$ only after the presentation of the μ^{th} example. Also

$$h_\mu \equiv \frac{\mathbf{J}^{\mu-1}.\,\mathbf{S}^\mu}{J^{\mu-1}}, \qquad \sigma_J^\mu = sign(h_\mu) \tag{10}$$

are the normalized local field and the output of the student. The local stabilty of a given example, before it is used to modify $\mathbf{J}$, is

$$\Delta_\mu \equiv h_\mu\, \sigma_B^\mu \tag{11}$$

and the example is correctly classified if Δ_μ is positive. The calculations will also involve a Gardner-like parameter related to the stability on the professor

$$\kappa_\mu \equiv \frac{b_\mu\, \sigma_B^\mu}{\rho^{\mu-1}}\,. \tag{12}$$

3 Lower Bounds on the generalization errors

The learning dynamics that we study has the form

$$\mathbf{J}^\mu = (1 - \frac{\Omega_\mu}{N})\,\mathbf{J}^{\mu-1} + \frac{J^{\mu-1}}{N} W_\mu \sigma_B^\mu\, \mathbf{S}^\mu + \frac{J^{\mu-1}}{\sqrt{N}}\,\tilde{\eta} \tag{13}$$

where Ω_μ is a decay term and W_μ is the weight of the Hebbian term. These two quantities depend on the particular example, and our objective is to determine the best possible weight in order to optimize generalization. The last term takes into account possible white noise in the synaptic couplings, but its overall effect is just to increase the amplitude of the drift parameter T by $\frac{1}{2}\langle\tilde{\eta}.\tilde{\eta}\rangle$. The $J^{\mu-1}$ factor is included to simplify further developments. With this notation, $W_\mu = 1/J^{\mu-1}$ for the simple

Hebb rule, $W_\mu = \kappa - \Delta_\mu$ for the Adaline algorithm and $W_\mu = (\kappa - \Delta_\mu)\,\Theta(\kappa - \Delta_\mu)$ for the relaxation algorithm, with κ a positive constant [9]. We now follow closely the method of reference [2]. In the thermodynamic limit $N \to \infty$, $\alpha = \mu/N$, the learning dynamics can be written as a differential equation, which after being averaged over all the possible histories, leads to

$$\frac{d\rho}{d\alpha} = \rho\,\left(\langle\, W_\mu(\kappa_\mu - \Delta_\mu) - \frac{1}{2}W_\mu^2\,\rangle - T\right) \tag{14}$$

for the overlap.

The evolution of the length of the synaptic vector $\mathbf{J}$ is given by

$$\frac{dJ}{d\alpha} = J\,\left(\langle\, W_\mu\Delta_\mu + \frac{1}{2}W_\mu^2 - \Omega_\mu\,\rangle + T\right) \tag{15}$$

The choice of $\Omega_\mu = W_\mu\Delta_\mu + \frac{1}{2}W_\mu^2 + T$ gives a *local* method to keep the norm of $\mathbf{J}$ constant. Nevertheless, this needs *a priori* knowledge about T.

Notice that the decay term Ω_μ is absent from the equation for ρ. Then, if W_μ does not depend on $J^{\mu-1}$, the two differential equations are decoupled and the decay term does not affect the performance (in contrast with the simple Hebb case [1] where $W_\mu = 1/J^{\mu-1}$ and the two equations are coupled).

The choice of the weight function W_μ is made by demanding that the gain per example, e.g. $d\rho/d\alpha$, be maximum, with the restriction that it can only depend on b_μ through its sign. Not suprisingly the resulting weight function is the same as in reference [2]

$$W_\mu^{ES} = \langle\kappa_\mu\rangle_{ES} - \Delta_\mu \tag{16}$$

where the triangular brackets $\langle\,\ldots\,\rangle_{ES}$ denote the expected value calculated with the conditional probability distribution of $|b_\mu|$ given σ_B^μ and h_μ, thus the name Expected Stabilty. Therefore, the maximum average gain per example can be calculated from

$$\frac{d\rho}{d\alpha} = \rho\,\left(I(\rho) - T\right) \tag{17}$$

where

$$I(\rho, P(\mathbf{S})) = \sum_{\sigma_B^\mu = \pm 1} \int_{-\infty}^{\infty} dh_\mu P(\sigma_B^\mu, h_\mu) \frac{(W_\mu^{ES})^2}{2} \ . \tag{18}$$

Up to now the probabilty distribution of the examples $P(\mathbf{S})$ from which $P(\sigma_B^\mu, h_\mu)$ is obtained has not been chosen. It reflects the strategy in the choice of the learning examples. Two strategies have been considered in the literature [3, 4, 6,10]: *passive learning*, that is, learning from random examples with a uniform distribuition, which leads to b_μ and h_μ being normal gaussian variables with correlation ρ; and *active learning* (learning by "selection of examples" or "queries") where the student demands the answer to the "hardest to classify" examples. In this case the ES average is performed over $P(|b_\mu| \mid h_\mu = 0)$.

In the case of passive learning the weight function is [2]

$$W_\mu^{ES} = \frac{\lambda}{\sqrt{2\pi}} \frac{\exp(-\Delta_\mu^2/2\lambda^2)}{H(-\Delta_\mu/\lambda)}, \tag{19}$$

where

$$\lambda \equiv \frac{\sqrt{1-\rho^2}}{\rho} = \tan(\pi e_G) \tag{20}$$

and

$$H(x) \equiv \int_x^\infty Dt, \qquad Dt \equiv \frac{e^{-t^2/2}}{\sqrt{2\pi}} dt. \tag{21}$$

With this weight W the function I appearing in equation (18) is

$$I(\lambda) = \frac{\lambda^3}{2\pi} \int_{-\infty}^{\infty} Du \frac{e^{-(1-\lambda^2)u^2/2}}{H(u)} \ . \tag{22}$$

As $\alpha \to \infty$ the error stops decreasing. The saturation error e_G^∞ will increase as T^z, with an exponent z characteristic of the algorithm and learning strategy. The error $e_G^\infty(T)$ is obtained from equation (20) and from

$$I(\lambda^\infty) = T. \tag{23}$$

For small T ,

$$e_G^\infty \simeq \frac{1}{\pi} \lambda^\infty \quad = C T^{\frac{1}{3}} \tag{24}$$

where the constant C is given by

$$C = \left[\frac{\pi^2}{2} \int_{-\infty}^{\infty} Du \frac{e^{-\frac{u^2}{2}}}{H(u)} \right]^{-1/3} \approx 0.447. \tag{25}$$

This $z = \frac{1}{3}$ should be compared to the $z = \frac{1}{4}$ exponent of the Optimal Hebb case of reference [1]. For the selection of examples strategy, we have

$$W_\mu^{ES} = \sqrt{\frac{2}{\pi}} \lambda, \tag{26}$$

and

$$I(\lambda) = \frac{\lambda^2}{\pi}. \tag{27}$$

The assymptotic value of the error in this case is

$$e_G^\infty = \frac{1}{\pi} \arctan \sqrt{\pi T}, \tag{28}$$

leading to $z = \frac{1}{2}$.

The exponent z is not independent of the generalization exponent x which characterizes the behaviour of e_G for $T = 0$ and large α

$$e_G \propto \alpha^{-x}. \tag{29}$$

To show this we proceed as follows. For small T and large α (small λ) we have in general

$$\frac{d\rho}{d\alpha} = \rho \left(c_1(T) \lambda^{n_1} + c_2(T) \lambda^{n_2} + \cdots - T \right) \tag{30}$$

where c_1 and c_2 may depend on T. Also $n_1 < n_2$, and if $c_1(T=0) \neq 0$ then $n_1 = 1/z$, giving the relation

$$\frac{1}{x} = \frac{1}{z} - 2 \tag{31}$$

This can be seen by inserting the assymptotic trial solution (for $T = 0$)

$$\rho = 1 - K\,\alpha^{-2x} \tag{32}$$

where K is some positive constant, into the differential equation leading to

$$\alpha^{-2x-1} \propto e_G^{1/z} \propto \alpha^{-x/z} \tag{33}$$

from which we obtain relation (31). The case $z = 1/2$ $(x \to \infty)$ indicates an exponential error decay (selection of examples [2]).

If $c_1(T = 0) = 0$ then one can only say that

$$\frac{1}{x} > \frac{1}{z} - 2 \tag{34}$$

The bounds on the generalization error $e_G(\alpha)$ obtained by numerically integrating equation (17) with the apropriate weight function are shown in figure (1) for $T = 0.5$. Also the results for the Optimal Hebb are presented for comparison. This algorithm requires the knowledge of the value of the drift parameter and uses an optimal weight decay $\Omega(T)$ for the simple Hebb rule. The saturation error is larger in the case of random uniform examples, while in the case of selection of examples the two errors are the same. This result should be compared with that of reference [1] bearing in mind that we start from *tabula rasa* while they use $J = 1$ as the initial condition and so their approach to the limit will be slower.

In figure (2) our results for the assymptotic errors $e_G^{\infty}(T)$ are shown, as well as the ones for the algorithm of reference [1]. Also the selection of examples strategy (lower curves) errors are shown, being the same for both algorithms.

Since the optimal weight function W_{μ}^{ES} depends on ρ, the algorithm cannot be implemented without knowledge of the drift parameter T, which is needed to integrate equation (17). As it stands it only provides lower bounds on the average generalization

errors. Of course, we can use in the weight function a value ρ_{μ} measured during the simulation to obtain an optimal "benchmark" curve, against which other algorithms will be compared. We will call this procedure the "benchmark" algorithm. In the next section we will study a self-adaptive algorithm that uses an estimate of the overlap ρ and performs very well without knowledge of the noise level.

4 Self-Adaptive algorithm

We now turn to a practical implementation of these ideas. First notice that, by equation (20), the optimal weight function can be written as

$$W(e_G, \Delta_{\mu}) = \frac{1}{\sqrt{2\pi}} \tan(\pi e_G) \exp\left(-\frac{\Delta_{\mu}^2}{2\tan^2(\pi e_G)}\right) \left[H\left(\frac{-\Delta_{\mu}}{\tan(\pi e_G)}\right)\right]^{-1}. \tag{35}$$

This suggests that we consider the neural net as endowed with a self-evaluation system which is used to estimate the generalization error, or more specifically $\tan(\pi e_G)$, by the results, failures or successes on the previous examples. The self-evaluator mechanism integrates over a range of time and two competing factors have to be considered. If the integration time is too large, then it will remember results which are no longer meaningful and its estimation will lag the actual value of e_G. While if it is too short, it will have no time to accumulate a significant statistics. We will consider an exponential loss of information of old examples and look at the following evaluator

$$\hat{e}_G^{(\mu)} = (1 - \frac{\omega}{N})\,\hat{e}_G^{(\mu-1)} + \frac{\omega}{N}\,\varepsilon_{\mu}. \tag{36}$$

Here ω is the inverse integration time or recency span, and $\varepsilon_{\mu} = (1 - \sigma_B^{\mu}\sigma_J^{\mu})/2$, which is (one) zero if the last example was (un)correctly classified. Since the tangent diverges at $\pi/2$, it is not very precise to estimate the tangent factor of equation (35) by just calculating the tangent of $\pi\hat{e}_G^{(\mu)}$, and we instead use as estimators the truncated power

series expansions

$$\hat{\lambda}_k = \hat{\tan}(\pi e_G)_k = \pi \hat{e}_G + \frac{1}{3}(\pi \hat{e}_G)^3 + \frac{2}{15}(\pi \hat{e}_G)^5 + \ldots + c_k\,(\pi \hat{e}_G)^k. \qquad (37)$$

In figure (3) we judge the different estimators by comparing them to the benchmark value in a simulation. The third power estimator $\hat{\lambda}_3$ is seen to be the best. In figure (4) the rule is fixed for $\alpha < 5$ and $\alpha > 15$, while it drifts with $T = 0.2$ in between. The performance of the $\hat{\lambda}_3$ estimator is measured for different values of ω and a good choice is seen to lie near the value $\omega = 2$.

In figure (5) the results of a simulation are shown for an algorithm which uses the best estimator ($\hat{\lambda}_3$) with $\omega = 2$. This is a realistic algorithm, since neither the drift parameter nor the value of the overlap ρ are needed. It is seen to work quite well since it approximates the lower bounds very efficiently for different drift rates. This in fact means that the algorithm is self-adaptive, a fact which can be better seen in figure (6) where the rule is fixed until $\alpha = 5$, performs a random walk with $T = 0.2$ until $\alpha = 15$, after which it remains fixed again.

We also have performed the so called Wisconsin test [5]. In this situation the rule suddenly changes and the adaptation of the student to the new environment is analyzed. In figure (7) the performance of our algorithm in the Wisconsin test is compared to the simple Hebb, and seen to be significantly better.

The failure to use the optimal algorithm might be thougth of as a lesion to the learning mechanism of a neural net. Althougth by using the simple Hebb algorithm a perceptron can learn the first rule, it will have a great difficulty to detect a rule change. A similar behaviour has been noticed in patients with "pre-frontal sindrome" [5], and has been previously modeled by Danchin and Changeux [7] and by Levine and co-workers [8] with neural networks. It is interesting that the same effects appear in a natural way within the much simpler perceptron architecture *which has not been*

designed to model this phenomena.

5 Conclusions

We have analyzed the problem of learning a linearly separable rule drifting randomly with time, through the single presentation of examples. Lower bounds to the generalization errors are given. For low rule drift (or low synaptic noise) the saturation error increases as T^z. The exponent for passive learning can be improved from the Hebbian with optimal weight decay result of Biehl and Schwarze, $z = \frac{1}{4}$, to $z = \frac{1}{3}$ while $z = \frac{1}{2}$ in the active learning strategy. For online learning we obtained a relation between the generalization exponent x and the noise exponent z, $1/x = 1/z - 2$, if $c_1(T = 0) \neq 0$. This conditon is true for a large class of algorithms, and a careful study of this will appear elsewhere. It will be interesting to check if this relation is also valid for iterated learning.

These bounds inspire a realistic self-adaptive algorithm which needs, in order to assess the actual generalization abilty, a self-evaluating mechanism. The performance of this algorithm is studied numerically for slow drifts as well as for sudden changes of the rule, the so-called Wisconsin test. The algorithm approximates very well the optimal performance, does not require previous knowledge of the drift parameter and is robust to changing rule drifts.

This algorithm belongs to the class which we have previously called Expected Stability algorithms [2], since the weight function depends on the expectation value of the stability of an example in the professor net as judged by the student. The learning proceeds by giving higher weight to the suprising examples, that is, those with a high local field in the student net, which have been misclassified. This differentiated treatment of the examples is shared by other algorithms, but with a weight function independent of

the learning stage (e_G). An important characteristic is that the optimal weight function depends on the generalization error, and this demands that the net be endowed with an extra feature, that is a self-evaluator mechanism. The failure of the Hebb algorithm in keeping information about temporal ordering leads to poor adaptability under changing rules or "perseveration" and lack of "recency" information, which is similar to the behaviour of "pre-frontal" lesioned patients.

The extension of this kind of results for multilayer nets would be of great interest both for the theoretical aspects as well as for their possible applications. We are presently working on these ideas and this will be the topic of future publications.

Acknowledgements We thank M. Biehl and H. Schwarze for communicating their results prior to publication, and H. S. del Nero for indicating references [5, 8] and for discussions of the "pre-frontal" sindrome.

O.K. has been supported by a CNPq fellowship, and N.C. has received partial support from CNPq and FAPESP.

Remark: After this work was finished we received a preprint from Biehl and Schwarze where essentially the same lower bounds are obtained for the continuous drifting rule case.

References

[1] Biehl, M. and Schwarze, H. 1992 Online learning of a time dependent rule *Europhys. Lett.* **20**, 733

[2] Kinouchi, O. and Caticha, N. 1992 Optimal Generalization in Perceptrons *J. Phys. A: Math. Gen.* **25** 6243

[3] Opper M. and Haussler D. 1991 Generalization Performance of Bayes Optimal Classification Algorithm for Learning a Perceptron *Phys. Rev. Lett.* **66** 2677

[4] Kinzel W. and Ruján P. 1990 Improving a Network Generalization Ability by Selecting Examples *Europhys. Lett.* **13** 473

[5] Shallice, T. 1988 *From Neuropsychology to Mental Structure* (Cambridge: Cambridge University Press)

[6] Opper M., Kinzel W., Kleinz J. and Nehl R. 1990 On the Ability of the Optimal Perceptron to Generalise" *J. Phys A: Math. Gen* **23** L581

[7] J. P. Changeux, *La Recherche 244* vol 23, 705 Juin (1992)

[8] Levine, D. S., Leven, S. and Prueitt, P. S. 1992 Integration, Disintegration, and the Frontal Lobes, in *Motivation, Emotion and Goal Direction in Neural Nets* ed D. S. Levine and S. J. Leven (Lawrence Erlbaum Assoc., Publishers Hillsdale NJ)

[9] Abbot, L. F. 1990 Learning in Neural Networks Memories *Network* **1** 105

[10] Watkin T. L. H., Rau A. and Biehl M. 1993 The Statistical Mechanics of Learning a Rule (to appear in *Reviews of Modern Physicis*)

Figure Captions

[Fig.1] Lower bounds for the generalization error $e_G(\alpha)$ obtained from the integration of eq. (17) with W of the Expected Stability algorithm: passive learning (dashed line) and active learning (solid line). Simulation results for the simple Hebb (circles). Optimal Hebb for passive learning (squares), and for the Optimal Hebb with selection of examples (stars) both for *tabula rasa* and starting from $J = 1$ [1]. Simulations starting from *tabula rasa* (lower curves) reach the limit faster than from $J = 1$. Average over 50 runs, $N = 149$.

[Fig.2] Asymptotic generalization error $e_G(T)$ from eq.(23) as a function of the noise parameter T. Passive learning Optimal Hebb [1] (small dash), Expected Stability (solid) and simulation results, $N = 149$, average over 50 runs (circles). Active learning (long dash).

[Fig.3] Performance of the different evaluators ($\omega = 2$), $|1 - \hat{e}_G/e_G|$, where $\hat{e}_G$ is the generalization error for the following evaluators: Tangent (diamond), $\hat{\lambda}_1$ (squares),$\hat{\lambda}_3$ (circles), $\hat{\lambda}_5$ (triangles), $\hat{\lambda}_7$ (star), and e_G is the benchmark algorithm generalization error.

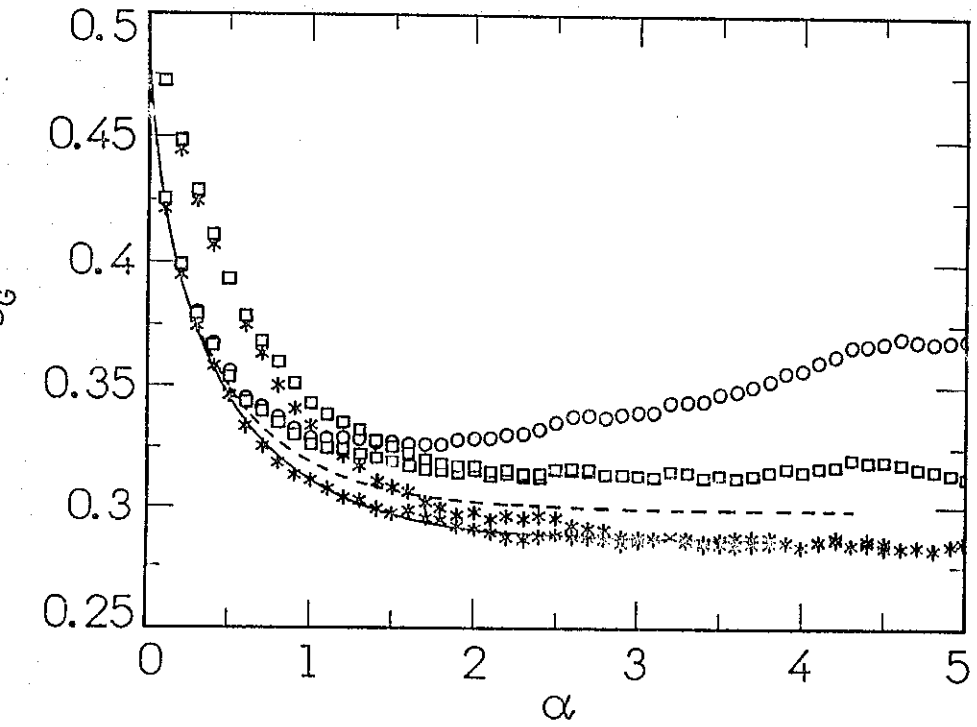
[Fig.4] Performance of the $\hat{\lambda}_3$ algorithm for different values of ω. The rule is fixed until $\alpha = 5$, then it drifts until $\alpha = 15$, with a noise parameter $T = 0.2$. $\omega = 0.2$ (circles), $\omega = 2$ (squares), $\omega = 20$ (triangles), $\omega = N$ (diamond), benchmark (solid). $N = 99$, average over 50 runs.

[Fig.5] Generalization error for different levels of noise. Self-adaptive algorithm with the $\hat{\lambda}_3(\omega = 2)$ evaluator (circles) from a simulation with $N = 149$, averaged over 50 runs and for the lower bounds (solid line). From the lower curve to the top for $T = 0, 0.05, 0.2, 0.5$ respectively.

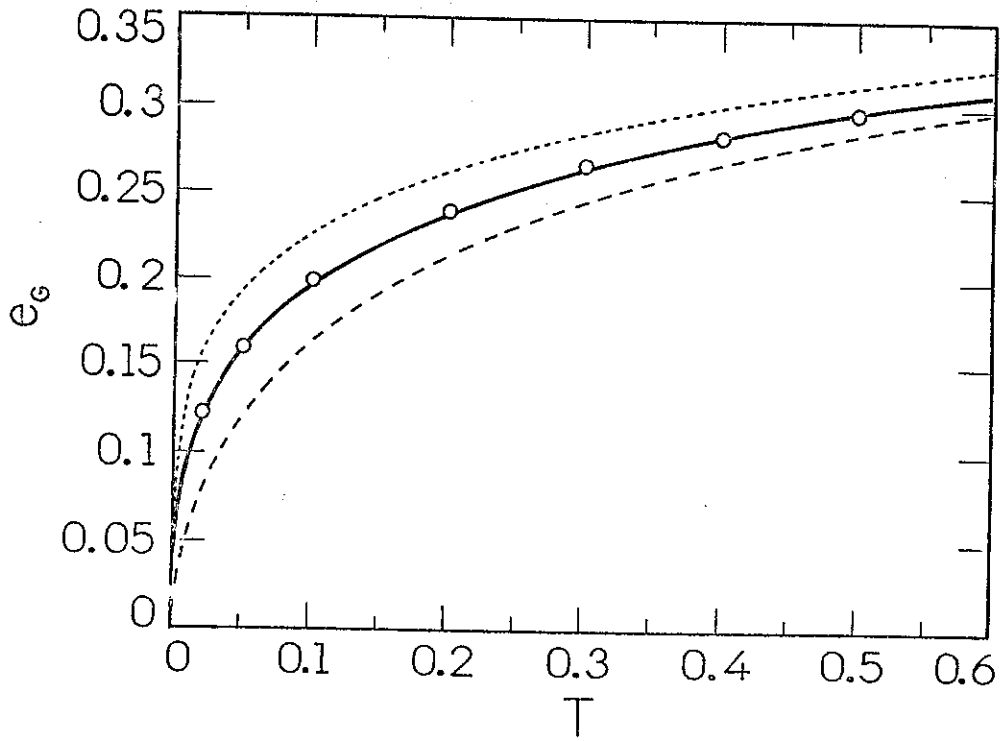
[Fig.6] Performance of different algorithms for changing drifts. For $\alpha < 5$ and $\alpha > 15$, T = 0 and in between $T = 0.2$. Simple Hebb (squares), Optimal Hebb (triangles), Self-adaptive algorithm with the $\hat{\lambda}_3(\omega = 2)$ (circles) and the benchmark (solid line). $N = 99$, average over 50 runs.

[Fig.7] The Wisconsin Test: The rule is piecewise constant in time. It suddenly changes at $\alpha = 10, 15, 20$. The dotted line is the generalization ability of the simple Hebb algorithm. It is quite efficient at the begining, but is very slow in adapting to a new rule. The dashed line is obtained for the self-adaptive algorithm with the $\hat{\lambda}_3(\omega = 2)$ evaluator. The solid line is obtained using the measured value of ρ (benchmark algorithm).

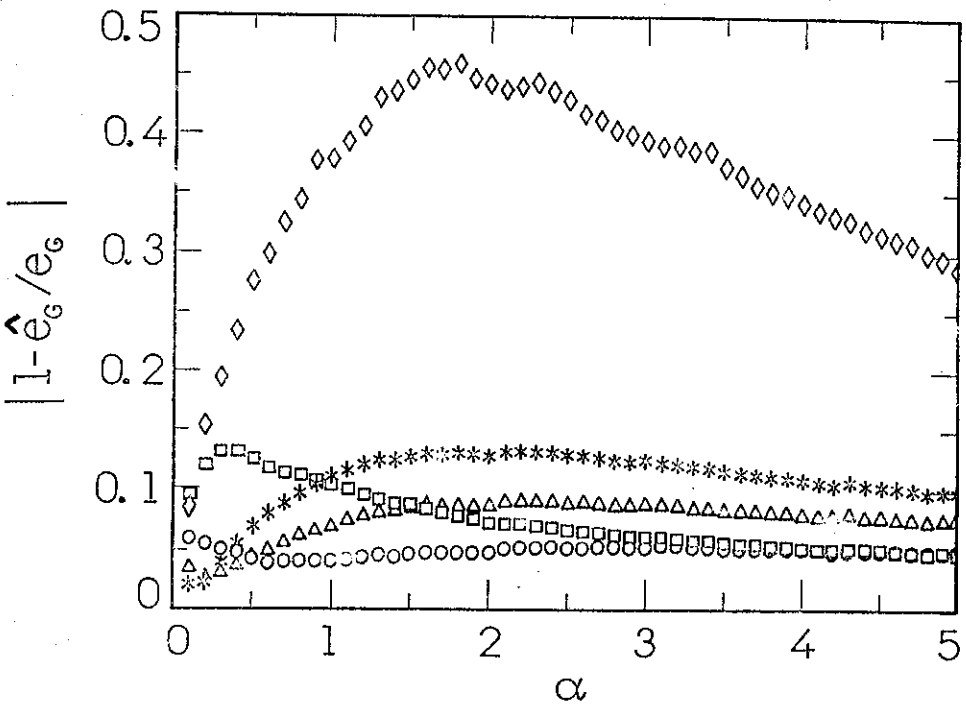
①


0.5
0.45
0.4
0.35
0.3
0.25
0
1
2
3
4
5
α

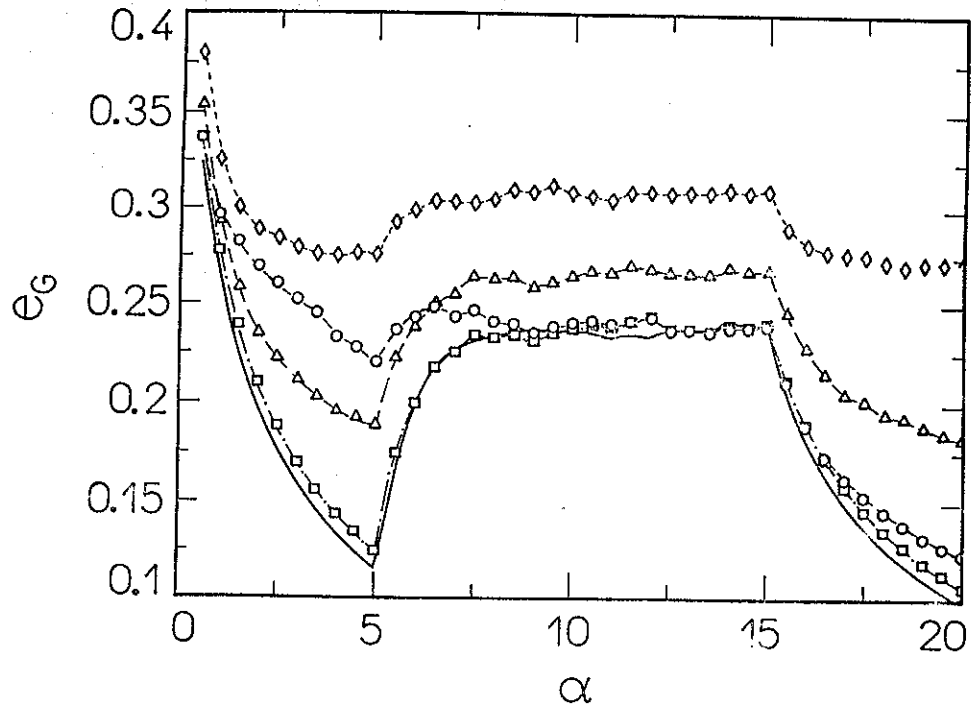
②

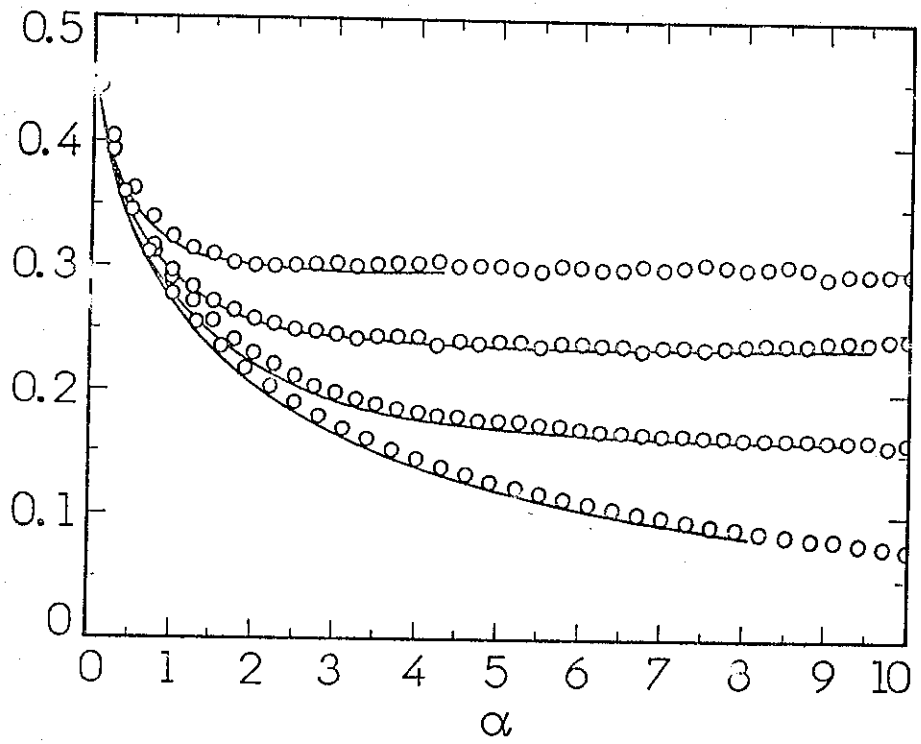

0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
e_G
0
0.1
0.2
0.3
0.4
0.5
0.6
T

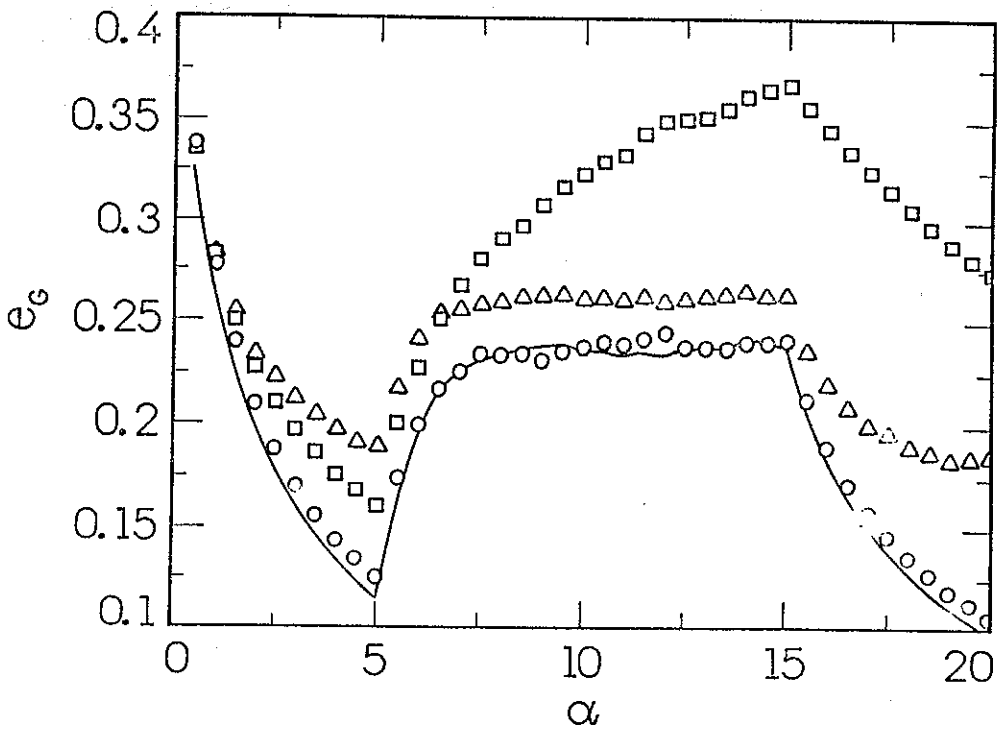
(3)

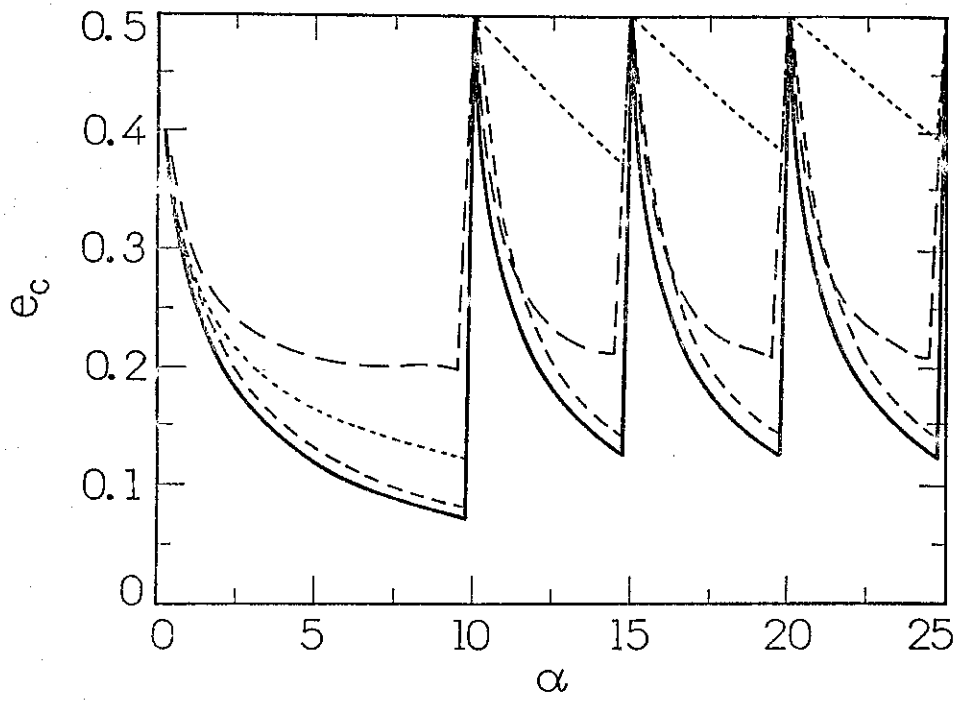

0.5
0.4
0.3
0.2
0.1
0
0
1
2
3
4
5
α
|1-ê_G/e_G|

(4)


0.4
0.35
0.3
0.25
0.2
0.15
0.1
0
5
10
15
20
α
e_G


0.5
0.4
0.3
0.2
0.1
0
0 1 2 3 4 5 6 7 8 9 10
α


0.4
0.35
0.3
0.25
0.2
0.15
0.1
e_c
0 5 10 15 20
α


0.5
0.4
0.3
0.2
0.1
0
e_c
0
5
10
15
20
25
α