Averaging over networks: properties, evaluation and minimization

Vicente Ruiz de Angulo and Carme Torras [*]

Abstract

The expectation of a function of random variables is modelled as the value of the function in the mean value of the variables plus a penalty term. This penalty term is calculated exactly, and the properties of different approximations are analyzed. In particular, for quadratic functions, the penalty term is shown to have a supervised and an unsupervised part. Then, two algorithms for minimizing the expected error of a feedforward network of random weights are compared. One of them is stochastical, while the other is deterministic and more flexible. Given a particular feedforward network architecture and a training set, the deterministic algorithm here presented accurately finds the weight configuration that makes the network response most resistant to a class of weight perturbations. Finally, the study of the most stable configurations of a network unravels some undesirable properties of networks with asymmetric activation functions

1 Introduction

The minimization of the expected error of a network with random weights, i.e.

$$\min_{W} \int E(\mathcal{W})\mathcal{P}(\mathcal{W}|W)d\mathcal{W}, \tag{1}$$

where E is the standard error function, $\mathcal{W}$ is the vector of random weight variables, $\mathcal{P}$ its density function and W its mean or another parameter of the distribution, is interesting for several reasons. First, it should be noted that, for certain functions φ, this minimization is equivalent to fault-tolerance maximization (see Section 2.2):

[*]The authors are with the Institut de Robòtica i Informàtica Industrial (CSIC-UPC), Edifici NEXUS, Gran Capità 2-4, 08034-Barcelona, Spain. E-mail: vruiz@iri.upc.es, ctorras@iri.upc.es. A summary of this paper appears in the Proceedings of the Intl. Conf. on Artificial Neural Networks (ICANN'94).

$$\min_W \int E(\varphi(R;W))\ P(R)\ dR, \tag{2}$$

R being a noise signal affecting W in some way determined by φ (e.g., additively), which also determines the density function $P(R)$.

As a matter of fact, the addition of noise to the weights during training has been used by Murray and Edwards [14, 15] (proportional to the weight value) and An [1] (independent of the weight value) for fault- tolerance enhancement and for improving generalization. However, simple weight noise addition during backpropagation training does not really perform the minimization (2) by itself. Section 4 shows that the optimization of a random weight network requires 1) adding noise only temporarily to get a perturbed weight gradient sample and subtracting it afterwards, and 2) applying a learning rate schedule that tends adequately to zero.

Our work is closer to that of Hinton and van Camp [7, 8], in the sense that they propose to minimize the expected error of a truly random weight network. Since the information content of a random weight depends on its variance, it is possible to regulate the effective number of network parameters by regulating the variance of $\mathcal{W}$, in the same way as one controls the regularization constant when using a regularizer. However, the proposal of Hinton and van Camp differs from ours in the cost function, which in their case is inspired in the Minimal Description Length (MDL) principle [18], and it is minimized also with respect to each of the individual weight variances. A related approach based also on MDL is presented in [19].

In addition to fault-tolerance and complexity reduction, the minimization (1) is interesting to mitigate catastrophic forgetting. In effect, this is the appropriate cost function to be used when encoding the learning set so that the retention of this set of patterns is maximized when new information is stored. In principle we should know the shape of the distribution of the perturbations that the storage of the new information will cause. But, as will become clear in Section 2.4, only the variance (which can be deduced from the new patterns' expected error) is required. This treatment of the catastrophic forgetting problem, which prepares the network *before knowing* the next patterns to learn, is cleanly complementary to that in [21], where a new pattern is encoded in a *previously trained* network causing minimal disturbance to the stored information.

This paper analyzes the expectation of a cost function with respect to a certain family of random variables, particularizing the study to the mean squared error function used in feedforward networks, when the random variables are the connection weights. A deterministic algorithm that emulates learning with random weights is also presented, and its precision is demonstrated with careful experiments.

2 Analytical study

Several authors have recently showed different approximations of the penalty term or regularizer implicit in the addition of weight noise [14, 1] or input noise [2, 12, 17] during training. Here, we present in general form the complete regularizer that is implicitly added to a function $g(U)$ when U is affected in some way by a perturbation symmetric around its mean. First, we develop the additive case and show how to generalize it to other types of noise. Then, we particularize the result for mean-square-type functions, indicating the constraints that valid approximations should satisfy and sharpening the order of the error derived from them. These results were first presented in [20].

2.1 The case of zero-mean additive noise

We define g_{P+} as:

$$g_{P+}(U) = \int g(U+R)\; P(R)\; dR, \tag{3}$$

where $P(R)$ is a zero-mean, symmetric probability distribution. First, we develop the Taylor series expansion of $g_{P+}(U+R)$, and after some manipulations we get:

$$g_{P+}(U) = g(U)\int P(R)\; dR \;+\; \sum_k \frac{\partial g}{\partial u_k}(U)\int r_k\; P(R)\; dR \;+ \\ \sum_{k,j} \frac{\partial^2 g}{\partial u_k \partial u_j}(U)\int r_k\; r_j\; P(R)\; dR\; + \ldots,$$

where u_k and r_k are the kth components of U and R, respectively. Now we impose the fundamental hypothesis that will allow us to proceed: all the integrals that cannot be put in the form $\int \prod_{k=1}^{n} r_k^{2n_k} P(R)dR$ must be null. A sufficient condition for this is $P(r_1,\ldots,r_i,\ldots r_n) = P(r_1,\ldots,-r_i,\ldots,r_n)$, i.e., P must be symmetric.

We cannot directly eliminate these null terms in (4), because some derivatives of g appear in different summatories with different derivation order. Thus, we first make explicit how many times g is derived with respect to each of the domain components. Note that the first integral has a value of 1:

$$g_{P+}(U) = g(U) + \sum_{m=1}^{\infty}\; \sum_{i_1,i_2,\ldots i_m=1}^{n} \frac{1}{m!}\, \frac{\partial^m g}{\partial u_1^{h_1}\ldots\partial u_n^{h_n}}(U)\int \prod_{k=1}^{n} r_k^{h_k} P(R)\; dR, \tag{4}$$

where h_k is the number of times k appears in $\{i_1,\ldots,i_m\}$. Merging the two summatories and taking into account that $m = \sum_{i=1}^{n} h_i$, the preceding expression is transformed into:

$$g_{P^+}(U) = \sum_{h_1,h_2,\ldots h_n=0}^{\infty} \frac{(\sum_{i=1}^n h_i)!}{h_1!\ h_2!\ldots h_n!} \frac{1}{(\sum_{i=1}^n h_i)!} \frac{\partial^m g}{\partial u_1^{h_1}\ldots \partial u_n^{h_n}}(U) \int \prod_{k=1}^{n} r_k^{h_k} P(R)\ dR, \tag{5}$$

which, since the terms that include an odd h_i among $h_1, h_2, \ldots, h_n$ are null, leads to:

$$g_{P^+}(U) = \sum_{h_1,h_2,\ldots h_n=0}^{\infty} \frac{1}{(2h_1)!\ (2h_2)!\ldots (2h_n)!} \frac{\partial^{2m} g}{\partial u_1^{2h_1}\ldots \partial u_n^{2h_n}}(U) \int \prod_{k=1}^{n} r_k^{2h_k} P(R)\ dR. \tag{6}$$

This expression is not yet satisfactory, because we would like to separate clearly the original function and the regularizer, which cannot be done easily in an equation like this. Multiplying and dividing all terms by $\frac{(\sum_{i=1}^n h_i)!}{h_1!\ h_2!\ldots h_n!}$ we get:

$$g_{P^+}(U) = \sum_{h_1,h_2,\ldots h_n=0}^{\infty} \frac{(\sum_{i=1}^n h_i)!}{h_1!\ h_2!\ldots h_n!} \frac{1}{(2h_1)!\ (2h_2)!\ldots (2h_n)!} \frac{h_1!\ h_2!\ldots h_n!}{(\sum_{i=1}^n h_i)!} \tag{7}$$

$$\frac{\partial^{2m} g}{\partial u_1^{2h_1}\ldots \partial u_n^{2h_n}}(U) \int \prod_{k=1}^{n} r_k^{2h_k} P(R)\ dR.$$

Now we have $\frac{(\sum_{i=1}^n h_i)!}{h_1!\ h_2!\ldots h_n!}$ multiplying each term of the summatory. This is what is needed to reverse step (4)-(5), thus decomposing the summatory:

$$g_{P^+}(U) = g(U) + \sum_{m=1}^{\infty} \sum_{i_1,i_2,\ldots im=1}^{n} \frac{\prod_{l=1}^n h_l!}{m! \prod_{l=1}^n (2h_l)!} \frac{\partial^{2m} g}{\partial u_{i_1}^2 \ldots \partial u_{i_m}^2}(U) \int \prod_{k=1}^{m} r_{i_k}^2 P(R)\ dR. \tag{8}$$

Let $\alpha_{i_1\ldots i_m}$ denote the constants preceding the derivatives. The integrals are also constants, specifically cross moments of P, which we write shortly $\mu_{2h_1,\ldots,2h_n}$. Thus, we finally obtain:

$$g_{P^+}(U) = g(U) + \sum_{m=1}^{\infty} \sum_{i_1,i_2,\ldots i_m=1}^{n} \alpha_{i_1\ldots i_m}\ \mu_{2h_1,\ldots,2h_n} \frac{\partial^{2m} g}{\partial u_{i_1}^2 \ldots \partial u_{i_m}^2}(U). \tag{9}$$

This result is also valid for a deterministic perturbation, by just using the moments of the uniform probability distribution and multiplying the second member of the equality by the volume of the perturbation. Since this volume is just a constant factor, for the concern of minimization, the deterministic case is equivalent to that of the uniformly distributed random perturbation. Discrete perturbations (random or not) can also be dealt with by appropriately redefining $\mu_{2h_1,\ldots,2h_n}$.

2.2 Generalizations and variants

Initially, the goal is the analysis of the following integral, which is a generalization of (1):

$$g_{\mathcal{P}}(U) = \int g(\mathcal{U})\ \mathcal{P}(\mathcal{U}|U)\ d\mathcal{U}. \tag{10}$$

At the same time, to link our analysis to fault-tolerance applications we consider that $\mathcal{U} = \varphi(R;U)$, i.e., that $\mathcal{U}$ is the result of a perturbation R with probability density $P(R)$ affecting U in a way determined by the function φ. Then, we could be interested in

$$g_{\varphi,\mathcal{P}}(U) = \int g(\varphi(R;U))\ P(R)\ dR. \tag{11}$$

Under what conditions these two expressions are equivalent? (10) can be readily derived from (11) if $\varphi(R;U)$ is one to one. In this case we can make the variable change $R = \varphi^{-1}(\mathcal{U};U)$ in (11):

$$g_{\varphi,\mathcal{P}}(U) = \int g(\varphi(\varphi^{-1}(\mathcal{U};U);U))\ P(\varphi^{-1}(\mathcal{U};U))\ \left|\frac{\partial\varphi^{-1}}{\partial\mathcal{U}}(\mathcal{U};U)\right|\ d\mathcal{U}, \tag{12}$$

where $\left|\frac{\partial\varphi^{-1}}{\partial\mathcal{U}}(R;U)\right|$ is the determinant of the Jacobian of the $\varphi^{-1}(.;U)$ mapping. Now, the dependence on U disappears from the argument of g, because $\varphi(\varphi^{-1}(\mathcal{U};U);U) = \mathcal{U}$. As the densities $P(R)$ and $\mathcal{P}(\mathcal{U}|U)$ are related by

$$\mathcal{P}(\mathcal{U}|U) = P(\varphi^{-1}(\mathcal{U};U))\ \left|\frac{\partial\varphi^{-1}}{\partial\mathcal{U}}(\mathcal{U};U)\right|, \tag{13}$$

we immediately get (10) from (12). In general, the one-to-one condition is unnecessary (the proof is analogous to that for the independence of the expectation value with respect to the choice of the variables of integration). But, for the dependence on U in (10) to move from the probability distribution to the argument of g, $\varphi(R;U)$ must reflect the way in which $\mathcal{U}$ and U are related. More concretely, φ must be such that $\mathcal{P}(\varphi(R;U))\ \left|\frac{\partial\varphi}{\partial R}(R;U)\right|$ does not depend on U.

Thus, (10) is one of the many possible versions of (11) that can be obtained by making the change of variables $\mathcal{U} = \psi(R)$, $\psi(.)$ being any one-to-one mapping. The particular case of $\psi(.) = \varphi(.;U)$ changes the dependence on U from g to $\mathcal{P}$, yielding (10). The interesting point here is that the shape of $\mathcal{P}(\mathcal{U}) = P(\psi^{-1}(\mathcal{U}))\ \left|\frac{\partial\psi^{-1}}{\partial\mathcal{U}}(\mathcal{U})\right|$ changes with ψ.

Our analysis applies to versions of (11) with a probability distribution symmetric around its mean and, thus, some previous transformation may be required. Suppose that $P(R)$ in (11) is already symmetric around its mean value R_m. We show next how to reduce $g_{\varphi,\mathcal{P}}$ to the particular case of $\varphi(R;U) = U+R$, and a zero-mean $P(R)$, which was derived in the last section. We define h as $h(V;U) = g(\varphi(V;U))$, and assume the convention that $h_{P'+}(V;U)$ only implies integration over perturbations of V, considering U as a parameter vector. V will play the role of a ficticious variable, which is only interesting at one point. Then, if R_m is the mean of $P(R)$, $R' = R - R_m$, and P' is the probability density function of R', it is easy to check that $h_{P'+}(R_m;U)$ is equivalent to $g_{\varphi,\mathcal{P}}$:

$$\begin{aligned} g_{\varphi,\mathcal{P}}(U) = \int g(\varphi(R;U))\ P(R)\ dR = \\ \int g(\varphi(R_m + R';U))\ P(R_m + R')\ dR = \\ \int h(R_m + R';U)\ P'(R')\ dR' = \\ = h_{P'+}(R_m;U). \end{aligned} \tag{14}$$

As an example, consider $\varphi(R;U) = R\ U$ and $P(R)$ symmetric around R_m. Note that in this case $\mathcal{P}(\mathcal{U})$ is not symmetric. We can first translate P to center it,

$$\int g(R\ U)\ P(R)\ dR = \int g((R_m + R')\ U)\ P'(R')\ dR', \tag{15}$$

and then by taking $h(V;U) = g(VU)$ we get

$$\int h(R_m + R';U)\ P'(R')\ dR' = h_{P'+}(R_m;U). \tag{16}$$

2.3 The quadratic case

The most common error function in connectionist networks is a summatory of functions of the form $g(U) = 1/2(F(U) - D)^2$. We now study this type of functions. Note that substituting D by the mean of $F(U)$, g_{P+} is the variance of the perturbed $F(U)$. Following (9), the regularizer is now:

$$\sum_{m=1}^{\infty} \sum_{i_1,i_2,\dots i_m=1}^{n} \frac{1}{2}\ \alpha_{i_1\dots i_m} \mu_{2h_1,\dots,2h_n}\ \frac{\partial^{2m}(F(U) - D)^2}{\partial u_{i_1}^2 \dots \partial u_{i_m}^2}(U). \tag{17}$$

We now apply the equality $\frac{\partial^n(x(u)y(u))}{\partial u^n} = \sum_{i=0}^{n} \left(\begin{array}{c} n \\ i \end{array}\right) \frac{\partial^i x}{\partial u^i}\ (u)\ \frac{\partial^{n-i} y}{\partial u^{n-i}}(u)$ to develop each of the derivatives in the terms $\frac{\partial^{2m}(F(U)-D)^2}{\partial u_{i_1}^2 \dots \partial u_{i_m}^2}(U)$:

$$\frac{\partial^{2m}(F(U) - D)^2}{\partial u_{i_1}^2 \dots \partial u_{i_m}^2}(U) = \frac{\partial^{2m-2} \left\{ \sum_{j_1=0}^{2} \left(\begin{array}{c} 2 \\ j_1 \end{array}\right) \frac{\partial^{j_1}(F(U)-D)}{\partial u_{i_1}^{j_1}}\ \frac{\partial^{2-j_1}(F(U)-D)}{\partial u_{i_1}^{2-j_1}} \right\}}{\partial u_{i_2}^2 \dots \partial u_{i_m}^2}(U) = \tag{18}$$

$$\frac{\partial^{2m-4}\left\{\sum_{j_1=0}^{2}\binom{2}{j_1}\sum_{j_2=0}^{2}\binom{2}{j_2}\frac{\partial^{j_1+j_2}(F(U)-D)}{\partial u_{i_1}^{j_1}\partial u_{i_2}^{j_2}}\;\frac{\partial^{4-j_1-j_2}(F(U)-D)}{\partial u_{i_1}^{2-j_1}\partial u_{i_2}^{2-j_2}}\right\}}{\partial u_{i_3}^{2}\dots\partial u_{i_m}^{2}}(U) = \tag{19}$$

$$\sum_{\substack{j_1\,=\,0\,\dots\,2\\ \vdots\\ j_m\,=\,0\,\dots\,2}}\left(\prod_{l=1}^{m}\binom{2}{j_l}\right)\frac{\partial^{\Sigma j_l}(F(U)-D)}{\partial u_{i_1}^{j_1}\dots\partial u_{i_m}^{j_m}}\frac{\partial^{2m-\Sigma j_l}(F(U)-D)}{\partial u_{i_1}^{2-j_1}\dots\partial u_{i_m}^{2-j_m}}\,(U). \tag{20}$$

Since F is always derived, except when all j_l are null or when all j_l are 2, the expression of the regularizer completely explicited as a function of F is:

$$\frac{1}{2}\sum_{m=1}^{\infty}\sum_{i_1,i_2,\dots i_m=1}^{n}\alpha_{i_1\dots i_m}\;\mu_{2h_1,\dots,2h_n}\left\{\left[\sum_{\substack{j_1\,=\,0\,\dots\,2\\ \vdots\\ j_m\,=\,0\,\dots\,2}}\beta_{j_1\dots j_m}\;\frac{\partial^{\Sigma j_l}F}{\partial u_{i_1}^{j_1}\dots\partial u_{i_m}^{j_m}}(U)\;\frac{\partial^{2-\Sigma j_l}F}{\partial_{i_1}^{2-j_1}\dots\partial u_{i_m}^{2-j_m}}(U)\right] + \right.$$
$$\left. + \; 2(F(U)-D)\frac{\partial^{2m}F}{\partial u_{i_1}^{2}\dots\partial u_{i_m}^{2}}(U)\right\}$$

where $\beta_{0,0\dots0} = \beta_{2,2\dots2} = 0$ and $\beta_{j_1\dots j_m} = \prod_{l=1}^{m}\binom{2}{j_l}$ in the remaining cases. The particular form of this expression, composed of an "error-dependent" and an "unsupervised" part, is made clear by observing that:

$$\begin{aligned}\int(F(U+R)-D)^2\;P(R)\;dR = (F(U)-D)^2 + \\ \int(F(U+R)-F(U))^2\;P(R)\;dR\; + \\ +\;2(F(U)-D)\int(F(U+R)-F(U))P(R)dR.\end{aligned} \tag{22}$$

If $H(X) = (F(X)-F(U))^2$ and $G(X) = (F(X)-F(U))$, it is easy to show using (9) that the compounds of the quadratic regularizer match the last two terms of (22), and thus, the regularizer can be expressed also as:

$$\frac{1}{2}\left[H_{P+}(U)\;+\;2(F(U)-D)\;G_{P+}(U)\right]. \tag{23}$$

$H_{P+}(U)$ can be considered as a measure of the variation of F around U. It is not dependent on any "desired value" D, and so can be evaluated (and minimized) in any point.

2.4 Approximations

To ease the notation, from now on we assume that P is a joint probability density, product of independent and equally-distributed probability functions accounting for the noise in each of the individual components of U. Taking the terms corresponding to $m = 1$ in (9), we have an approximation of g_{P+} subject to an error of order $\mathcal{O}(\mu_4)$:

$$g_{P+}(U) \approx g(U) + \frac{\sigma^2}{2} \sum_k \frac{\partial^2 g}{\partial u_k^2}, \tag{24}$$

where σ^2 is the variance of P. This order of error is higher than that estimated by Murray and Edwards [14, 15], and Bishop [2], for particular cases of addition of noise in the backpropagation error function.

In the next section we will need the deterministic version of this formula for a set of discrete perturbations $\{R^{i)}\}$ of cardinality n_l:

$$g_{P+}(U) \approx g(U) + \frac{1}{2n_l} \sum_k \frac{\partial^2 g}{\partial u_k^2} \sum_i \left(r_k^{i)}\right)^2. \tag{25}$$

All these approximations, although very accurate for low-level noise, are not always satisfactory. For example, when g is a positive function, the estimation (24) of its mean is not guaranteed to be positive. In the case studied in Section 2.1, this estimation of the regularizer is:

$$\frac{\sigma^2}{2} \sum_k \left[\left(\frac{\partial F}{\partial u_k}(U) \right)^2 + (F(U) - D) \frac{\partial^2 F}{\partial u_k^2}(U) \right]. \tag{26}$$

Positiveness is not only a drawback for theoretical reasons but, as observed by Bishop [2] regarding the case of noisy inputs in connectionist networks, it poses also problems to the development of algorithms based on (26) to minimize g_{P+}. How can positive approximations be characterized? A solution could be to consider only the "unsupervised" part of the regularizer, and to select from this only the terms in which the multiplying derivatives are equal. For example, $\frac{\sigma^2}{2} \sum_k \left(\frac{\partial F}{\partial u_k}(U) \right)^2$ is one of such estimations. However, with this strategy, we cannot guarantee any order of error, no matter how many terms we add, because we are always neglecting terms of the same order as those we are including.

Knowing the complete regularizer, it is possible to devise an strategy to have positive estimates with a desired order of error. Let us show that a sufficient condition for guaranteeing positiveness is that the appearances of the derivatives of F in the approximation are all the appearances in the complete regularizer of the same derivatives. An approximation of this type is exact for any polynomial F such that all its derivatives not appearing in the estimation are zero. For a general function F, there always exists a polynomial having the same combination of values for F and its derivatives in the point U. So the estimation of g_{P+} for F is the same as that for such polynomial, which, being exact, should be positive. Thus, to have a positive estimation of the regularizer with a desired precision (say $\mathcal{O}(\mu_4)$), in a first step we include all the terms of the regularizer that should be added anyway to get that precision (the terms in (26)) and, in a second step, all the appearances in the complete regularizer of the derivatives of F appearing in the first step are added. For example, to get the minimal positive estimate subject to an error of order $\mathcal{O}(\mu_4)$,

$$\frac{\mu_4}{8}\sum_{ij}\frac{\partial^2 F}{\partial u_i^2}(U)\frac{\partial^2 F}{\partial u_j^2}(U) \tag{27}$$

should be added to (26).

2.5 Relation between the minima of g and g_{P+}

Let U^* and U^*_{P+} be the minimizers of $g(U)$ and $g_{P+}(U)$, respectively. The first observation is that the minimization of g_{P+} does not favor points in which g is insensitive to variation of the parameters. As a matter of fact, *the first derivatives do not appear at all in the complete regularizer (9)*. However, when $g(U) = 1/2(F(U) - D)^2$, it favors the insensitivity of F.

Another remark is that g_{P+} tends to look for convex regions, although this tendency is regulated by the variance. If it is low enough, the minimization will attain low points in g, and for this reason, they would be concave with high probability.

We would like to point out also that, although (24) is a good approximation of g_{P+}, this does not mean that it is possible to estimate U^*_{P+} from U^* easily. As a matter of fact, given an unknown $g(U)$, we cannot bound the distance to which U^*_{P+} can be translated, and thus an approximation of U^*_{P+} based on a Taylor series expansion of g around U^* is uncertain (although it is also true that, for a given $g(U)$, there exists always a variance for which U^*_{P+} remains in a fixed neighborhood of U^*).

A further reason supporting the last comment regards the shifting of the minimum as the noise increases. The line $U^*_{P+}(\sigma^2)$ can be discontinuous in certain conditions, that is, U^*_{P+} can jump abruptly and without transition to different regions of the space when varying continuously the variance. This phenomenon is equivalent to the phase transitions of some physical systems, in which the variation of a parameter can give raise to changes in the shape of the energy of the system that could cause a sudden shift in the location of the minimum. An example can be seen in Fig. 1a.

This Fig. may lead one to think that this phenomenon can only happen when passing from the basin of attraction of one minimum to that of another one. Since it is believed that there exist few local minima (in the strict sense) in the back-propagation error function, and that the minima are symmetric, doubts could arise about the existence of the discontinuity in this case. However, Fig. 1b shows that when the number of perturbed parameters is more than 1, the discontinuities can appear even when there is only a minimum in $g(U)$, in this case due to the presence of narrow ridges in the shape of the function. Even one dimension could be enough, if the minimization is done with respect to parameters different from the perturbed ones (e.g., the minimization of the standard error function with respect to the weights, when the inputs are noisy).

3 Estimation of $E_{P+}(W)$

In the rest of the paper, we concentrate on the practical problems of estimating and minimizing the cost function associated to a network with random weights, $E_{P+}(W)$, which was the goal initially stated and argued for in Section 1. We will be constrained to use rather small networks to evaluate objectively our methods. A way of evaluating the tolerance to damage expressed by $E_{P+}(W)$ could be to use the Montecarlo method applied to the calculation of the integral, which turns out to be very costly. That is, to use a set of perturbations R_i drawn from $P(R)$, and to obtain the mean of the corresponding $E(W + R_i)$. An alternative is to use the approximations suggested in Section 2.4. As it will become clear later, (24) or (26) are accurate enough for all interesting combinations of W and σ and, thus, we concentrate on them. However, it is possible to add also (27) to get more precise results that are guaranteed to be positive, without having to calculate any extra elements of the Hessian of F.

To check the goodness of (24), we could compare its results with those of the Montecarlo method. But comparing an estimation with another estimation is embarrassing if an objective evaluation of accuracy is sought. Instead, we have preferred to use the simplest deterministic version of g_{P+}, the mean of the E values in the extreme points of a cross centered on W, oriented to coincide with the axes, and whose extremes are z unities from the mean point. That is, we will evaluate the function:

$$error(z) = \frac{1}{2n_w} \, [E(w_1 + z, w_2, \ldots w_{n_w}) + E(w_1 - z, w_2, \ldots w_{n_w}) + E(w_1, w_2 + z, \ldots w_{n_w}) + \\ + E(w_1, w_2 - z, \ldots w_{n_w}) + \ldots + E(w_1, w_2, \ldots w_{n_w} + z) + E(w_1, w_2, \ldots w_{n_w} - z), \quad (28)$$

n_w being the number of weights in the network. The advantage of using $error(z)$ is that its exact calculation is feasible. From formula (25) we obtain that

$$error(z) \approx E(W) + \frac{z^2}{2n_w} \sum_i \frac{\partial^2 E}{\partial w_i^2}(W). \tag{29}$$

Our intentions must be clear: to asses how good is the approximation (24) of $E_{P+}(W)$, we take its deterministic version, i.e., $error(z)$, and we compare it with (29). Figures 2a,b,c and d, show several comparisons between $error(z)$ and its estimation by means of (29). For Fig. 2a and 2b we used a network with structure 1-3-1, and a learning set of five points randomly drawn from the function *sine* in the interval $[-\pi, \pi]$. Figures 2c and 2d use instead a network 6-20-1 with 80 points from the function $sin\left(\sum_{i=1}^{6} x_i\right)$, chosen with the same distribution as before in each of the domain components. Figures 2a and 2c show results obtained in a point relatively close to $W = 0$, drawn from a uniform distribution $[-3/\sqrt{\text{fan-in(j)}}, 3/\sqrt{\text{fan-in(j)}}]$ for each weight w_j, while the networks of Fig. 2b and 2d are at a minimum of E $(E(W) = 0.0005)$.

The first thing that catches the eye is that, in the point close to 0, the estimation is surprisingly good. This happens because $E(W)$ is much simpler in the neighborhood of the origin. Another question is the seemingly better precision of the small network. This fact is easily interpreted by noting that the large network, with a small variation of its weights, can represent a large number of different functions, thanks to the power provided by its hidden units. For this reason, the goodness of the estimation with respect to the magnitude of the weights is a more interesting measure. Since the average absolute value of the weights of the 1-3-1 network is ≈ 0.96 and, in the larger network, is less than half this value, in order to use this criterion in the comparison, the interval in Fig. 2b and 2d should also be halved, i.e, [0, 2], and thus the approximations in the large and the small networks are similar.

Until now we have highlighted the variability of the results depending on σ and the point W. However, the most interesting combinations of σ and W are yet to be evaluated. As a matter of fact, in what concerns minimization, the goodness of the approximation of $E_{P+}(W; \sigma)$ in the minimum of E is almost irrelevant. Instead, what counts is the accuracy of the approximation of $E_{P+}(W; \sigma)$ in its own minimum.

4 The minimization of $E_{P+}(W)$: A stochastic method

Here again, an accurate and objective evaluation of the methods constrains us to us to use moderately sized networks.

We define $E(W)$ as

$$E(W) = \frac{1}{2}\sum_p E^{p)}(W) = \frac{1}{2}\sum_p ||F(W; X^{p)}) - D^{p)}||^2 = \frac{1}{2}\sum_{k,p}(F_k(W; X^{p)}) - D_k^{p)})^2, \tag{30}$$

where $E^{p)}(W)$ is the error component corresponding to pattern p, $F(W, X)$ is the output of the network when an input pattern X is presented, the subscript k denotes its components, and $(X^{p)}, D^{p)})$ is the pth input-output pattern. We will abbreviate $F(W, X^{p)})$ with $F^{p)}(W)$.

One alternative to minimize $E_{P+}(W)$ is stochastic, based on the Montecarlo method, which was first tested in [1].

This stochastic algorithm consists basically of taking samples of $\bigtriangledown E(W + R)$ to estimate $\bigtriangledown E_{P+}(W)$. This can be done with different degrees of stochasticity. One is to take literally a large number of $\bigtriangledown E(W + R)$ samples before each learning step. Another one is essentially the same, but taking only one sample. Two other variations, associated to multiple and single sampling of R, consist of performing one learning step after estimating $\bigtriangledown E_{P+}^{p)}(W)$ instead of $\bigtriangledown E_{P+}(W)$. In [19], the single sampling - single pattern version. The basic iteration of this algorithm is:

```
Extract $R$ from $P(.)$
$W \leftarrow W + R$
$W \leftarrow W + \lambda \bigtriangledown E^{p)}(W)$
$W \leftarrow W - R$
```

$\lambda(t)$ expresses the time dependence of λ, which must approach zero at the final number of iterations. We choose a schedule inspired in [4, 3]. We simply hold the initial learning rate constant during a number τ of complete iterations and then multiply it by a constant χ after every complete iteration until the prefixed number of total iterations is reached. The schedule is thus determined by four parameters: initial learning rate, χ, τ and the total number of iterations. The learning rate of the stochastic algorithm is in principle limited by the fact that it must allow to collect enough statistics along the learning path before the characteristics of $E_{P+}(W)$ change too much. Given the very high dimensionality of the weight space in most neural network applications, it could be thought that this requirement would be determinant, imposing very low learning rates. But, as commented on in Section 6, where some experimental results obtained with DSA are presented, in practice this requirement does not result so restrictive.

We next discuss the basic problems that algorithms of the stochastic type have to address in order to arrive at a minimum of $E_{P+}(W)$: stability, convergence and precision. It turns out that the λ schedule and, especially, the weight restoration carried out in the fourth step of the stochastic algorithm, are crucial to really get such a minimum.

Stability. Stability is another limiting factor for the learning rate because the randomness of the gradients of the perturbed weights may produce oscillations that slow down or even prevent learning. This problem can be controlled by setting the learning rate to a sufficiently small value. However, if weight restoration is not carried out, the algorithm follows not only the gradient, but also the random perturbation produced to get it. In other words, a random walk is superimposed onto the minimization. In this case, if the variance of the noise is high, learning might even be prevented with any learning rate.

Convergence. To get true convergence (even if the gradient of $E(W+R)$ instead of that of $E^{p)}(W+R)$ is used) the learning rate must decrease slowly enough to zero using an appropriate schedule. If, for example, one instead progressively reduces the level of noise as made in [14, 15] in an attempt to get convergence, one reaches the nearest minimum of E instead of minimizing $E_{P+}(W)$. Note that, without weight restoration, the problem of convergence remains even in the case of decreasing learning rates.

Precision. Another problem of a different kind is the low precision with which stochastic algorithms optimize fault-tolerance for a particular $P(R)$, due to a side-effect implicit in its nature. The mean variance of a set of n samples from a distribution is slightly smaller than the real variance of the distribution in a rate of $n-1$ to n. But this effect is not very notorious. Instead, the fact that W is moving while collecting different $\bigtriangledown E(W+R)$ samples is much more influential, and it results in a sampling variance higher than that of $P(R)$. The higher the learning rate, the larger the sampling variance. As a consequence, the implicit trade-off between $E(W)$ and the regularizer is pushed towards making the network immune to a level of noise higher than the desired one given by $P(R)$. However, when λ tends to zero, this problem disappears and, thus, the $\lambda(t)$ must approach this limit slowly enough to arrive to the $E_{P+}(W)$ minimum. Once more, however, if weight restoration is not used, there are limits to the possibility of alleviating the problem with low learning rates because, even in the case of a zero rate, W continues moving while collecting samples.

5 The minimization of $E_{P+}(W)$: A deterministic method

We propose here an algorithm based on (24), first presented in [20], that overcomes the drawbacks of the stochastic algorithm. Until now, to simplify the notation, we have indexed the weight with a single subscript; from now on, let w_{ji} be the weight departing from unit i and impinging on unit j. The deterministic algorithm requires the calculation of the gradient of the approximation of the regularizer $\frac{\sigma^2}{2}\sum_{ji}\frac{\partial^2 E}{\partial w_{ji}^2}$, i.e, the problem is to calculate:

$$\sum_{l,k}\frac{\partial^3 E}{\partial w_{lk}^2 w_{ji}}\,(W) \tag{31}$$

for every (j, i). It could appear that the cost of calculating this expression is too high. But we will show that, at least for two-layer networks, an approximation of this expression is easy to calculate, providing excellent results. We approximate

$$\frac{\partial^2 E}{\partial w_{lk}^2} = \left(\frac{\partial F^{p)}}{\partial w_{lk}}(W)\right)^2 + \ (F^{p)}(W) - D^{p)})\frac{\partial^2 F^{p)}}{\partial w_{lk}^2}(W), \tag{32}$$

by dropping the error-dependent term $(F^{p)}(W) - D^{p)})\frac{\partial^2 F^{p)}}{\partial w_{lk}^2}(W)$. By doing this, we eliminate the risks of having a non-positive penalty function (see Section 2.4). How much precision is lost with this approximation? For low-variance noise, the main factor in (24) is $g(U)$ (here, $E(W)$), therefore a low value of E (and thus of $||F^{p)}(W) - D^{p)}||$) is expected at the minimum of $E_{P+}(W)$, and the term can be eliminated safely. Note that only the precision at the minimum is important for the final result of the optimization, thus we do not mind the quality of the estimation during the first and intermediate stages of learning. For high-variance noise, the misfits at the minimum are not negligible, but the function $F(W)$ implemented by the network is much simpler than that for low-variance noise, which in general does not favor high second derivatives of F.

Empirically we have checked that these two factors interact in such a way that we get really good minimizations of $E_{P+}(W)$ for all variances. Besides resulting in good accuracy, this approximation has the added advantage of not depending on D, which allows to minimize the regularizer at any point of the input space, independently of the training set.

We will make explicit the gradient of the regularizer (taking into account the above approximation) for two coincident cases: 1) two-layer networks with linear output units and using the mean squared error, and 2) two-layer networks whose output units activation function is *tanh*, and that use the relative entropy error. The two formulae coincide because the gradient of the mean squared function with respect to a linear output unit is the same as that of the relative entropy function with respect to the input of a *tanh* output unit. The gradient of the output units being equal, all kinds of derivatives for all the network weights must be equal in both cases. Let H and O be the set of hidden and output units. Then, it is shown in the Appendix that the derivatives of $E^p_{P+}(W)$ are

$$\frac{\partial E^{p)}_{P+}}{\partial w_{ji}}(W) \ \approx \ \frac{\partial E^{p)}}{\partial w_{ji}} \ + \ \alpha \begin{cases} 2w_{ji}\,(y'_i)^2\,P_p & \forall j \in O, i \neq bias \\ 0 & \forall j \in O, i = bias \\ x_i S_j & \forall j \in H \end{cases} \tag{33}$$

where $P_p = ||X_p||^2 + 1$ (we are considering a network with a bias unit connected to all hidden an output units), $S_j = 2y'_j(n_O y_j + y''_j P_p \sum_m w_{mj}^2)$, n_O is the number of output units, y_j is the activation function value of unit j, and y'_j and y''_j are its first and second derivatives, respectively. α is a parameter that regulates the importance of the regularizer in the minimization, and must be $\frac{\sigma^2}{2}$

to emulate a random noise of variance σ^2. P_p is a constant that does not change during learning and can be joined to the input patterns. S_j is the same for all the connections impinging on a hidden unit and, thus, must be calculated only once for each hidden unit and not for each input-hidden layer weight.

This is the most obvious version of the algorithm, in which all the weights are trying to make all the weights of the network insensitive. But the deterministic algorithm permits other possibilities, which are forbidden to the stochastic algorithm. For instance, in the former algorithm, the set of parameters one would like to make insensitive and the set of weights in charge of making them insensitive must not be necessarily the same. The different roles that the first and the second layers of weights play in RBF networks makes some of the possible combinations very interesting for this type of networks. In [19] several possibilities and their usefulness are commented on and it is shown that the gradient of the regularizer for RBF networks is also simple.

6 Comparison between the deterministic algorithm and the stochastic algorithm

We have, thus, two methods for minimizing E_{P+}: the deterministic algorithm developed in the last section, and the stochastic algorithm tested in [1] consisting of adding noise with weight restoration and a learning rate schedule similar to that in [3].

The comparison between the two methods we carry out in this section should be placed in context in order to be properly interpreted. On the one hand, the stochastic algorithm has fundamental limitations when compared to the deterministic algorithm: First, when noise is introduced in one weight, all the network weights are necessarily trained to make that weight insensitive to noise. Second, the regularizer is implicit and cannot be minimized independently of $E(W)$.

Third, at the end of a cycle (epoch) of presentations of the learning set patterns, the gradient of the complete cost function is available to the deterministic algorithm. This opens up the possibility of using the more efficient algorithms developed to minimize $E(W)$, such as conjugate gradient algorithms [13], quickprop [5] or SuperSAB [23]. Instead, a stochastic algorithm accumulating the gradients during an epoch gets a gradient which is partial with respect to the distribution of the noise. Thus, the on-line mode is the natural one for the DSA, and batch versions are not appropriate.

A common aspect to both methods for optimizing E_{P+} is that the initial point required for an efficient learning must be farther from the origin than that used in back-propagation. Nevertheless, a too distant point can also prevent learning. Probably a good way to move away from the origin without risk is to begin minimizing $E(W)$ and, after some iterations, switch to E_{P+}.

We show now results obtained with a 2-7-1 architecture and a learning set with twelve points of the function $sin(x_1 + x_2)$ in an interval $[-\pi, \pi]$ of the two domain components. Figures 3a,b and c compare the performances of the DSA and the on-line version of the deterministic algorithm. We took $\sigma = 0.2$ and we set the regularization constant α required to emulate the variance according to the following equivalences:

$$\alpha = \frac{\sigma^2}{2} = \frac{z^2}{2n_w}. \tag{34}$$

The schedule in both methods was always the same, except for the initial learning rate, which is the only parameter that is different in the three Fig.s. A little search for the optimal three remaining parameters determining $\lambda(t)$ (namely, τ, χ and the total number of iterations, as specified in Section 4) was carried out, taking into account that the stochastic algorithm should perform better for small τ's relative to the total number of iterations. The initial weights were also the same, drawn from a uniform distribution $[-2.5/\sqrt{\text{fan-in(j)}}, 2.5/\sqrt{\text{fan-in(j)}}]$ for each weight. A point so far from the origin and producing a large error ($E = 1.2$) was used to facilitate learning, as said before. Including this error in the graphic would have lowered the resolution in the presentation of the results; to prevent this, we have chosen $E(0)$ as the maximum value for the vertical axis. The Fig.s display two evaluation measures of performance: $E(W)$ and $error(z)$ with z chosen according to (34). $error(z)$ was used because it is an independent measure. It could be argued that a Montecarlo estimation would be a more convenient measure for the stochastic algorithm, because it samples random points with the same distribution as the stochastic algorithm. But, against intuition, when it was used the results were still more unfavourable for the stochastic algorithm, which is not surprising, because of the relatively limited amount of samples that can be collected in a reasonable amount of time, and of the biases of the random generators. The Montecarlo method turns out to be a very variable and unreliable estimation, while $error(z)$ is a non variable one.

Figure 3a, displaying results obtained with a small initial learning rate, does not show large differences between the two methods. In Fig. 3b, with a higher learning rate, the stochastic algorithm begins to suffer unstability problems. Finally, with a learning rate of 0.08, the stochastic algorithm fails completely to converge, remaining at a high level of error, as shown in Fig. 3c. Instead, these are the optimal conditions for the on-line version of the deterministic algorithm. This is the only Fig. in which the minimum is approximately reached within the prefixed number of iterations.

Nevertheless, the stochastic algorithm did not behave so badly for moderate noise (at least in the small networks in which comparisons were carried out), which is somewhat surprising in view of our discussion in Section 5. Below, we outline an explanation of why the need of low learning rates is less pressing than expected. More details can be found in [19]. the algorithm calculates at every step a gradient which can be divided in two parts: that corresponding to E^p), which is always exact, and a random one, which can be considered a partial information on the regularizer. It seems that the regularizer is generally a more smooth function than $E(W)$, and thus the statistics collected by the random component can be collected in regions greater than expected.

In all the Fig.s it is also possible to appreciate that the differences between the stochastic algorithm and the deterministic algorithm are larger in $E(W)$ than in $error(z)$, and that these differences increase with the learning rate, meaning that the stochastic algorithm is minimizing E_{P+} with a variance higher than that of the noise really introduced in the weights. This gives support to our discussion in Section 5 on how the variance is biased in the stochastic algorithm, even with weight restoration and decreasing learning rate.

An important drawback of the stochastic algorithm is not reflected in these figures. It is very difficult to say when the algorithm has converged, because the weights changes continuously and $error(z)$ is not a practical measure for most applications.

7 Accuracy of the minimum

The discussion above, and (and specially (Fig. 3c) makes evident that the stochastic algorithm cannot be accurate minimizing $E_{P+}(W)$ for the specified variance σ^2, it really uses a greater, difficult to determine variance.To check the validity of the approximation made to derive the deterministic algorithm for the minimization of $E_{P+}(W)$, the most direct option would be to compare its performance with that of the stochastic algorithm. However, in Section 4 we showed that the latter algorithm can be very imprecise in many cases. Thus, we judged preferable, as we did in Section 3, to use $error(z)$, whose value and gradient can be calculated exactly. Its direct minimization requires a huge computational effort, $2n_w$ presentations for each pattern before performing a learning step, but it is an ideal reference allowing to compare the results of the minimization of E plus the complete regularizer, with those produced by the simple approximation made in (33). To emulate $error(z)$, the regularizer constant α must take the value $\frac{z^2}{2n_w}$.

Figure 4 displays results obtained with the same architecture and training points as in the preceding section. The network was repeatedly trained by means of a careful direct minimization of $error(z)$, using a different z each time. The same network was also trained using a batch version of the deterministic algorithm, using a set of regularization constants appropriate to emulate the minimization of $error(z)$ for the set of z's previously used. The initial random points were the same for both algorithms in all cases. As stopping criterion, since the gradient is available with both methods, the reaching of a small fixed average gradient was used. The Fig. shows all the range of useful z's. Increasing z above 2.5, the minimized networks do not change anymore. The fact that the graphic is not stabilized at that point is not a contradiction. Evaluating $error(z)$ with varying z in a network with different weights produces different values.

The evident result is that, no matter the value of z, both methods reach a similarly good minimum of $error(z)$. Figures 5a and 5b show the weights of two networks obtained in the preceding experiment. We choose that corresponding to $z = 0.2$ (which is a significative level of noise in this network) and $z = 0.6$, for both methods. The result is also surprising: the weights are almost identical. Even more, if we consider that a cycle in the minimization of $error(z)$ includes all the presentations required to obtain the complete gradient, the number of cycles carried out to arrive at the same average gradient is almost the same.

We would like to point out that this experiment has been repeated varying all kind of parameters: number of hidden units, selection of starting point, number of patterns in the learning set, and stopping criterion, limited only by our computational resources. We obtained always an extraordinary similarity between the ideal minimum of reference, that of $error(z)$, and that reached by the algorithm. Therefore, we can claim with a reasonable confidence that the high precision of the algorithm is not limited to some particular biased conditions.

8 Characteristics of the minima of E_{P+}

We analyze briefly the type of minima, in terms of first and second derivatives of $E(W)$, enforced by the minimization of E_{P+}. The first derivatives of $E(W)$ are (assuming, for clearness of explanation, that F has only one component):

$$\frac{\partial E}{\partial w_{ji}} = (F^{p)}(W) - D^{p)}) \, \frac{\partial F^{p)}}{\partial w_{ji}}. \tag{35}$$

When $E(W) = \sum_p (F^{p)}(W) - D^{p)})^2$ is minimized, $\frac{\partial E}{\partial w_{ji}}$ is brought to zero, but only the magnitude of the first factor is minimized. Instead, when minimizing E_{P+}, $\frac{\sigma^2}{2}\left(\frac{\partial F^{p)}}{\partial w_{ji}}(W)\right)^2$ is the main component of the regularizer (indeed, the only one taken into account by the deterministic algorithm). If σ^2 is extremely large, $\frac{\partial F^{p)}}{\partial w_{ji}}(W)$ will tend to be null at the minimum of E_{P+}, and also $\frac{\partial E}{\partial w_{ji}}$ will be zero. With intermediate σ^2, the two factors will be simultaneously minimized but, although necessarily low, $\frac{\partial E}{\partial w_{ji}}$ will not be null in general.

The diagonal elements $\frac{\partial^2 E}{\partial w_{ji}^2}$ of the Hessian are included in the regularizer and, therefore, their minimization should produce very negative values. However it can be shown that, in most points of interest, the function $E(W)$ is concave, and in practice the regularizer becomes $\sum_{ji}\left|\frac{\partial^2 E}{\partial w_{ji}^2}\right|$. From equation (38) in the Appendix, it follows that the second derivatives of the hidden-to-output layer of the networks we are using are always positive. For the input-to-hidden layer, the accuracy shown by the algorithm in the preceding section indicates that the supervised part of $\frac{\partial^2 E}{\partial w_{ji}^2}$ is negligible at the minimum of E_{P+} compared to $\frac{\sigma^2}{2}\left(\frac{\partial F^{p)}}{\partial w_{ji}}\right)^2$. In generic random points outside of the minima, it can be proved, under reasonable assumptions [19] for multilayer networks, that the probability of having positive second derivatives of a weight grows with the closeness of the connection to the output layer and is, anyway, greater than 1/2.

For the non diagonal elements of the Hessian,

$$\frac{\partial^2 E}{\partial w_{lk}\partial w_{ji}} = \frac{\partial F^{p)}}{\partial w_{lk}}\frac{\partial F^{p)}}{\partial w_{ji}} + \ (F^{p)}(W) - D^{p)})\frac{\partial^2 F^{p)}}{\partial w_{lk}\partial w_{ji}}. \tag{36}$$

A discussion about the negligibility of the second summand similar to the one above can be carried out. Besides, as $\left(\frac{\partial F^{p)}}{\partial w_{ji}}(W)\right)^2$ and $\left(\frac{\partial F^{p)}}{\partial w_{lk}}(W)\right)^2$ are minimized by the regularizer, the magnitude of $\frac{\partial F^{p)}}{\partial w_{lk}}\frac{\partial F^{p)}}{\partial w_{ji}}$ will be also limited. Thus, although the non diagonal elements are not included in the regularizer, they also suffer a pressure to have low magnitudes at the minimum of E_{P+}.

On a more experimental ground, the evaluation of the linearity (defined as the average first derivative of the hidden units activation function in the training set points) of the networks resulting of minimizing $error(z)$, revealed a gradual increment as z increased. Instead, the weight magnitude $||W||$ behaves more irregularly, although it decreases radically for high z. However, it is interesting to know what happens in each of the units of the network separately, which motivates Fig. 6. In it, the module of the weight vector impinging on each unit is represented with a bar, filled with a different texture. For $z = 0$, the magnitudes of the vectors do not differ too much from one another. As z increases, the differences become larger, some magnitudes decreasing to zero while others raise. Murray and Edwards [15] and An [1] also observed that noise addition during training tends to lead some hidden nodes to their saturation ranges. Besides, An [1] pointed out that it also tends to reduce the activations of the remaining hidden units.

An interesting question is that of the most stable weight configuration for a network. All the experiments showed clearly that, when the level of noise increases over a certain level, the minimum of E_{P+} quickly tends to zero. This fact was predicted by the following intuitive reasoning. The cost function E takes very high values in most randomly-chosen weight configurations, especially if the weights are large. $W = 0$ is a point relatively low, central, around which all the minima of $E(W)$ lay symmetrically. This is true at least if the average of the output patterns is zero. Otherwise, the most central, stable point, is that whose bias-to-output weights take the mean value of the corresponding output pattern components over the learning set and whose remaining weights are null. In Fig. 6, the residual weight vector of the output unit is due to the bias weight, whose value is exactly the predicted one. In general the prediction agrees completely with the simulations.

From these considerations, it is evident that the most stable weight configuration, which permits the exact remembering of a unique input-output pattern, is that with all weights equal to zero, except the output units biases, which take the value of the output pattern.

All these claims are made in the implicit context of symmetric activation functions, which were used in all the experiments presented until now. If the logistic function of range $[0, 1]$ is used instead in the hidden and output units, then for the same reasons as before, all units will try to produce a zero output, making the weights impinging on them (on some or all) take values tending to $-\infty$. If the output units are linear and the hidden ones are logistic, the bias weights of the output layer accomplish the same function, and then the weights of the hidden-output layer tend to 0 and those of the input- output layer tend to $-\infty$.

Since it is undesirable for a network to reach its maximum stability with infinite weights, this is another good reason for not using asymmetric activation functions.

Now we are in a better position to understand why the deterministic algorithm works so well. Take into account that the experiments reported in the preceding section not only warrant the validity of the approximation of the Hessian diagonal, but also that of the complete regularizer in (24) *for the task of minimizing E_{P+}*. For small z, the approximation is good everywhere. For large z, the minimum is in a zone in which many weights are very small. We saw in Section 3 that, near the origin, the function is simpler and, therefore, the approximation is good even for large z.

Discussion

We have analyzed the complete regularizer implicit in the expectation of a cost function of random variables. All random distributions that can be transformed into symmetricaly distributed perturbations by means of change of variable are considered. When the function is the standard mean-squared-error function, the terms of the regularizer can be grouped into two components. The main one is equivalent to the mean of $(F(X) - F(U))^2$. This is a very general smoothing factor, a penalty term for the variation of F around the mean of the distribution. The other component is a misfit-dependent term containing derivatives of F of higher order than those in the preceding one [1].

[1] This can be related to the case of noisy inputs, for which Koistinen and Holstrom [10] showed that the implicit objective function of a backpropagation network, when the input components follow a distribution $P(X)$, becomes $D(X) = \frac{\sum_p D^{p)} P(X-X^{p)})}{\sum_p P(X-X^{p)})}$. This is also the expression of the outputs of a network with normalized RBF units [22], whose properties are rather different from those of the networks with non-normalized RBF units [16]. It can be

The explicit expression of the complete regularizer (9) allows to know exactly the order of the error made in the approximations, and opens up the possibility of guaranteeing properties such as positiveness. The analysis of the relation between the minimum of the perturbed function and that of the non perturbed one showed that neither of them may be computed as a function of the other. The minimum of the perturbed function can even follow a discontinuous path as a function of the variance.

To estimate $E_{P+}(W)$, the regularizer can be approximated with the trace of the Hessian multiplied by half of the variance with an error of order $\mathcal{O}(\mu_4)$. This estimation is not good everywhere, for example it is excellent in $W = 0$, but only valid for moderate variances in the minimum of E. Interestingly, however, the disregarded terms are close to zero for any given variance in the minimum of the $E_{P+}(W)$ found with that same variance.

One way of minimizing E_{P+} is to sample the weight distribution and use the resulting gradients to learn, like in and An [1]. We have discussed the importance of weight restoration to mitigate the main problems of the method: unstability, lack of convergence and, especially, low precision due to its stochastical nature, which makes the real variance of the samples along time higher than that of P. Even using weight restoration, high variances may impose however a too slow decrease of the learning rate towards zero, and is difficult to know when convergence is reached.

We developed a simple alternative deterministic algorithm, based on the minimization of the above approximation of the regularizer, which permits overcoming all these problems and offers additional possibilities. To test it, a deterministic perturbation function was devised whose value and gradient (and hence its minimum) can be calculated exactly. The precision of the algorithm turned out to be extraordinary for all the range of variances. This happens because, when the variance is high, the weights tend to the origin and, in this point, the approximation taken by the algorithm is good for all variances. When the variance is low, the approximation is good everywhere and the minimization is also correct.

It was found that, for networks of units with symmetric activation functions, as the variance tends to high values, the networks tend to have all weights closer to zero, except the biases of the output units, which take the mean values of the corresponding output pattern components. This convergence, however, is not uniform; the weights associated to some hidden units become null early, but other units survive to higher variances. Thus, the method of controlling the variance of a random weight network to enhance generalization can be thought of as an intermediate option between weight-decay [6] and parameter elimination [11, 24].

However, the networks of units with asymmetric activation functions (e.g.,

shown that the regularizer of the normalized RBF networks is (21), when the random variables are the input patterns.

logistic) behave in a different way: the simplest networks, with the least informative weights as produced by high variances, are those with infinite negative weights. In our opinion, this fact throws serious doubts on the convenience of using this type of networks or, at least, on the appropriateness of applying techniques such as weight-decay in its usual form to them.

Appendix

We derive here the regularizer's gradient for networks of one hidden layer and linear output units. This coincides with the gradient for networks having *tanh* activation functions at the output units and using the relative entropy error, as was explained in Section 5.

Our goal is to calculate

$$\sum_{l,k} \frac{\partial^3 E^{p)}}{\partial w_{lk}^2 w_{ji}} (W) \tag{37}$$

for every (j, i). We consider the bias unit as another input, that is, $x_0 = y_0 = 1$. First, we take the approximation of the second derivatives of the weights made in Section 5. For weights impinging on the output units, this approximation coincides with the exact second derivative:

$$\frac{\partial^2 E^{p)}}{\partial w_{lk}^2} (W) = y_k^2, \quad \forall\ l \in O. \tag{38}$$

To ease the notation, formulas are understood to be true for all the validity range of the non quantified subindices. Thus (38) holds for all neurons k connected to unit l. For the input-to hidden layer weights, the approximation yields:

$$\frac{\partial^2 E^{p)}}{\partial w_{lk}^2} (W) = x_k^2 \sum_m w_{ml}^2 y_l'^2, \quad \forall\ l \in H. \tag{39}$$

We divide the derivation of (37) in three parts:

- Bias weights of the output units, i.e., $j \in O,\ i = 0$
- Weights of the hidden-to-output layer, $j \in O,\ i \in H$
- Weights of the input-to-hidden layer, $j \in H$.

In each case, we first compute the terms for $l \in O$ and then those for $l \in H$.

Let us begin with the simplest case of w_{ji} connecting a bias to an output unit:

$$\frac{\partial^3 E^{p)}}{\partial w_{lk}^2 w_{ji}}(W) = 0, \;\; \forall\, j,l \in O, \;\; i = 0, \tag{40}$$

$$\frac{\partial^3 E^{p)}}{\partial w_{lk}^2 w_{ji}}(W) = 0, \;\; \forall\, j \in O, \;\; i = 0, \;\; \forall\, l \in H. \tag{41}$$

Thus, the regularizer's gradient with respect to each bias-to-output weight is:

$$\sum_{l,k} \frac{\partial^3 E^{p)}}{\partial w_{lk}^2 w_{ji}}(W) = 0, \;\; \forall\, j \in O, \;\; i = 0. \tag{42}$$

Now we concentrate on the second type of weights, those belonging to the hidden- to-output layer. As in the former case,

$$\frac{\partial^3 E^{p)}}{\partial w_{lk}^2 w_{ji}}(W) = 0, \;\; \forall\, j,l \in O, \;\; \forall\, i \in H. \tag{43}$$

When l is a hidden unit, we must distinguish between $l \neq i$ and $l = i$:

$$\frac{\partial^3 E^{p)}}{\partial w_{lk}^2 w_{ji}}(W) = 0, \;\; \forall\, j \in O, \;\; \forall\, i,l \in H, \; l \neq i, \tag{44}$$

$$\frac{\partial^3 E^{p)}}{\partial w_{lk}^2 w_{ji}}(W) = 2 w_{ji} y_i'^2 x_k^2, \;\; \forall\, j \in O, \;\; \forall\, i,l \in H, \; l = i. \tag{45}$$

Let P_p denote $\sum_k x_k^2$ for the particular pattern p and including the bias unit, so that $P_p = ||X^{p)}||^2 + 1$. Then the final expression of the regularizer gradient w.r.t. the hidden-to-output layer weights is:

$$\sum_{l,k} \frac{\partial^3 E^{p)}}{\partial w_{lk}^2 w_{ji}} = \sum_{\substack{k \\ l \in H}} \frac{\partial^3 E^{p)}}{\partial w_{lk}^2 w_{ji}} = \sum_k 2 w_{ji} y_i'^2 x_k^2 = 2 w_{ji} y_i'^2 P_p, \; \forall\, j \in O, \forall\, i \in H. \tag{46}$$

We finally deal with the hardest case corresponding to the gradient of a weight w_{ji} impinging on a hidden unit. As in the other cases, we first calculate the terms for $l \in O$ and $k \in H$, but now we distinguish between $k \neq j$ and $k = j$:

$$\frac{\partial^3 E^{p)}}{\partial w_{lk}^2 w_{ji}}(W) = 0, \;\; \forall\, j,k \in H, \; k \neq j, \;\; \forall\, l \in O, \tag{47}$$

$$\frac{\partial^3 E^{p)}}{\partial w_{lk}^2 w_{ji}}(W) = 2 y_j y_j' x_i, \;\; \forall\, j,k \in H, \; k = j, \;\; \forall\, l \in O. \tag{48}$$

Similarly, for a hidden unit l, we distinguish between $l \neq j$ and $l = j$:

$$\frac{\partial^3 E^{p)}}{\partial w_{lk}^2 w_{ji}}(W) = 0, \quad \forall\, j, l \in H,\ l \neq j, \tag{49}$$

$$\frac{\partial^3 E^{p)}}{\partial w_{lk}^2 w_{ji}}(W) = x_k^2 \sum_m w_{mj}^2 2y_j' y_j'' x_i, \quad \forall\, j, l \in H,\ l = j. \tag{50}$$

Let n_O be the number of output units. Then, the final expression of the regularizer gradient w.r.t. the input-to-hidden layer weights is:

$$\sum_{l,k} \frac{\partial^3 E^{p)}}{\partial w_{lk}^2 w_{ji}} = \sum_{\substack{k \\ l\,\in\,O}} \frac{\partial^3 E^{p)}}{\partial w_{lk}^2 w_{ji}} + \sum_{\substack{k \\ l\,\in\,H}} \frac{\partial^3 E^{p)}}{\partial w_{lk}^2 w_{ji}} = 2n_O y_j y_j' x_i + \sum_k x_k^2 \sum_m w_{mj}^2 2y_j' y_j'' x_i =$$
$$2n_O y_j y_j' x_i + 2y_j' y_j'' x_i P_p \sum_m w_{mj}^2 = 2x_i y_j'(n_O y_j + y_j'' P_p \sum_m w_{mj}^2), \quad \forall\, j \in H. \tag{51}$$

Fortunately, the factor $2y_j'(n_O y_j + y_j'' P_p \sum_m w_{mj}^2)$ is the same for all the weights impinging on j, thus we can denote it by S_j. Then the regularizer w.r.t. a weight w_{ji} of the input-to-hidden layer can be expressed as:

$$\sum_{l,k} \frac{\partial^3 E^{p)}}{\partial w_{lk}^2 w_{ji}} = x_i S_j, \quad \forall\, j \in H. \tag{52}$$

Expressions (42), (46) and (52) are the regularizer's gradient w.r.t. the bias of the output units, the hidden-to-output weights and the input-to-hidden weights, respectively.

Acknowledgments

Part of this work was developed when V. Ruiz de Angulo was at the Neural Networks Laboratory (I.S.I.S) of the Joint Research Center, Ispra site, where he was financially supported by a PhD grant from the European Union.

References

[1] An, G., 'The effects of adding noise during backpropagation training on generalization performance'. *Neural Computation*, vol. 8, pp. 643-674, 1996.

[2] Bishop, C.M., 'Training with noise is equivalent to Tikhonov regularization'. *Neural Computation*, vol. 7, pp. 108-116, 1995.

[3] Darken, C., Chang, J. and Moody, J., ‘Learning rate schedules for faster stochastic gradient search’. *Proc. of the IEEE Workshop - Neural Networks for Signal Processing.* New York: IEEE Press, 1992.

[4] Darken, C. and Moody, J., ‘Note on learning rate schedules for stochastic optimization’. *Advances in Neural Information Processing Systems 3*, Morgan Kauffman, San Mateo, California, pp. 832-838, 1990.

[5] Fahlman, S.E., ‘An empirical study of learning speed in back-propagation networks.*Technical Report CMU-CS-88-162*, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, 1988.

[6] Hinton, G.E, ‘Learning translation invariant recognition in a massively parallel network’. *Parallel Architectures and Languages Europe (PARLE)*, Lecture Notes in Computer Science, Springer-Verlag, Berlin, pp. 1-13, 1987.

[7] Hinton, G.E. and van Camp, D., ‘Keeping neural networks simple’. *Intl. Conf. on Artificial Neural Networks (Amsterdam)*, pp. 11-18, 1993.

[8] Hinton, G.E. and van Camp, D., ‘Keeping neural networks simple by minimizing the description length of the weights’. *Sixth ACM Conf. Comp. Learning Theory (Santa Cruz)*, pp. 5-13, 1993.

[9] Holmstrom, L. and Koistinen, P. ‘Using additive noise in back-propagation training’. *IEEE Trans. on Neural Networks*, vol. 3, pp. 24-38, 1992.

[10] Koistinen, P. and Holmstrom. L., ‘Kernel regression and back-propagation training with noise’. *Advances in Neural Information Processing Systems 4*, pp. 1033-1039, 1992.

[11] Le Cun, Y., Denker, J.S. and Solla, S.A., ‘Optimal brain damge’. *Advances in Neural Information Processing Systems 2*, Morgan Kauffman Publishers, 1990.

[12] Matsuoka, K., ‘Noise injection into inputs in back-propagation learning’. *IEEE Trans. on Sys. Man Cybern.*, vol. 22, pp. 436-440, 1992.

[13] Moller, M., ‘Supervised learning on large redundant sets’. *Intl. Journal of Neural Systems*, vol. 4, pp. 15-25, 1993.

[14] Murray, A.F. and Edwards, P.J., ‘Synaptic weight noise during multilayer perceptron training: Fault tolerance and training improvements’. *IEEE Trans. on Neural Networks*, vol. 4, pp. 722-725, 1993.

[15] Murray, A.F. and Edwards, P.J., ‘Enhanced multilayer perceptron performance and fault tolerance resulting from synaptic weight noise during training’. *IEEE Trans. on Neural Networks*, vol. 5, pp. 792-802, 1994.

[16] Poggio, T. and Girosi, F., 'Networks for approximation and learning'. *Proc. IEEE*, vol.78, 1481-1497, 1990.

[17] Reed, R., Oh, S. and Marks, R.J. II, 'Regularization using jittered training data'. *Proc. Intl. Conf. Neural Networks*, IEEE Neural Networks Council, Baltimore, vol. 3, pp. 147-152, 1992.

[18] Rissanen, J., *Stochastic Complexity in Statistical Inquiry.* World Scientific, Singapore, 1989.

[19] Ruiz de Angulo, 'Interferencia catastrófica en redes neuronales: Soluciones y relación con otros problemas del conexionismo'. *Ph.D. Thesis*, Basque Country University, 1995.

[20] Ruiz de Angulo, V. and Torras, C., 'Random weights and regularization'. *Intl. Conf. on Artificial Neural Networks (ICANN'94)*, pp. 1456-1459, 1994.

[21] Ruiz de Angulo, V. and Torras, C., 'On-line learning with minimal degradation in feedforward networks'. *IEEE Trans. on Neural Networks*, vol. 6, pp. 657- 668, 1995.

[22] Spetch D., 'A general regression neural network', *IEEE Trans. on Neural Networks*, vol. 2, pp. 981-988, 1991.

[23] Tollenare, T., 'SuperSAB: Fast adaptive back-propagation with good scaling properties'. *Neural Networks*, vol. 3, pp. 561-573, 1990.

[24] Weigend, A.S., Rumelhart, D.E. and Huberman, B.A., 'Generalization by weight- elimination with application to forecasting'. *Neural Information Processing Systems 3*. Morgan Kauffman Publishers, San Mateo, CA, pp. 885-882, 1991.

Figure captions

Figure 1a). The minimum of g (null variance) is located on the left side of the graphic, at the bottom of a narrow and deep chasm. If the variance of the variables raises up to a certain medium level, the minimum moves to the right of the Fig., and with higher variances to the center, without visiting the intermediate points.

Figure 1b). Level map of $g(u_1, u_2)$. The arrows indicate the sense of growing slope. There exists only one minimum, placed at the rightmost superior part. For perturbations distributed uniformly in a square greater than that drawn around $U*$, U^*_{P+} moves in a discontinuous way to the center of the figure, due to the presence of a ridge.

Figures 2a) and b). Comparison between $error(z)$ (the exact average value of E in a set of perturbations of the weights) and its estimation using the second derivatives of E. z indicates the magnitude of the perturbations. The employed architecture is 1-3-1. (a) Near to the origin. (b) In a minimum.

Figures 2c) and d). As above, but using a network 6-20-1.

Figure 3a). Stochastic and deterministic learning methods with initial learning rate 0.02.

Figure 3b). Stochastic and deterministic learning methods with initial learning rate 0.06.

Figure 3c). Stochastic and deterministic learning methods with initial learning rate 0.08.

Figure 4. Minima reached by the direct minimization of $error(z)$ and by the deterministic algorithm.

Figure 5a). Representation of the weights of two networks, one obtained by minimizing directly $error(z)$ and the other by using the deterministic algorithm. $z = 0.2$ was used.

Figure 5b). As in 5a), but with $z = 0.6$.

Figure 6. Lenghts of the weight vectors impinging on the units of the networks used in Fig. 4.

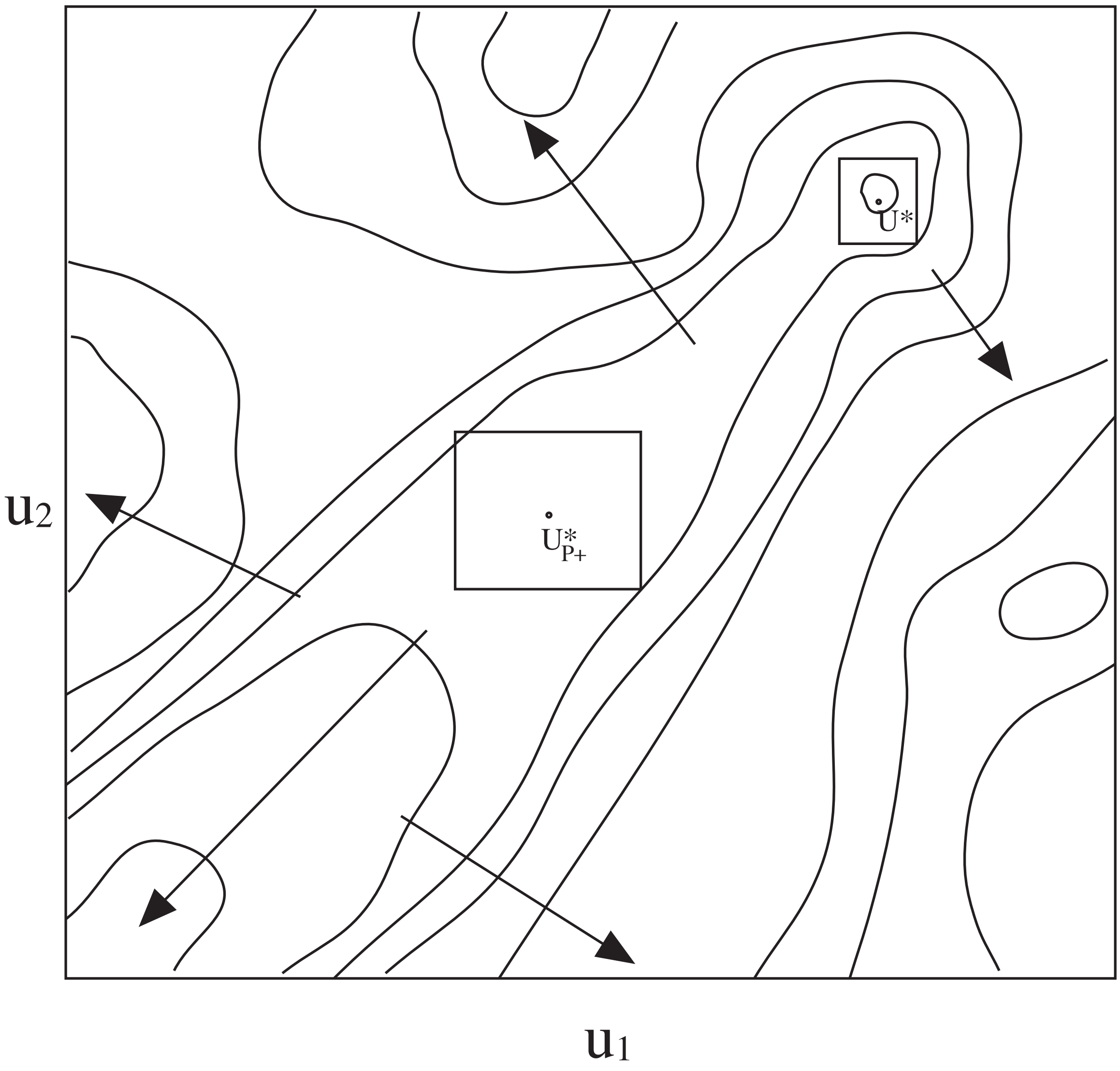
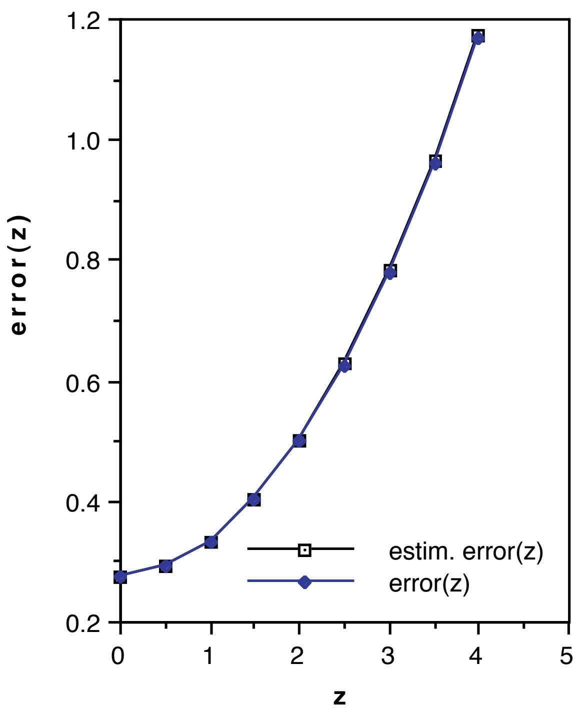
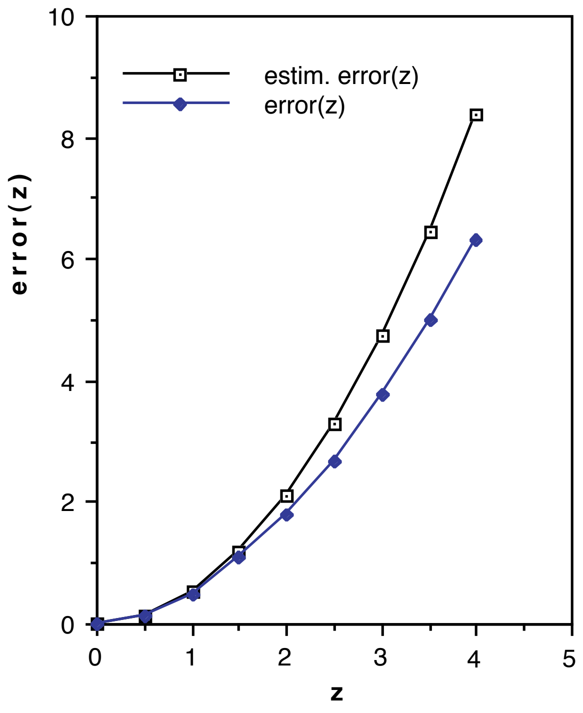


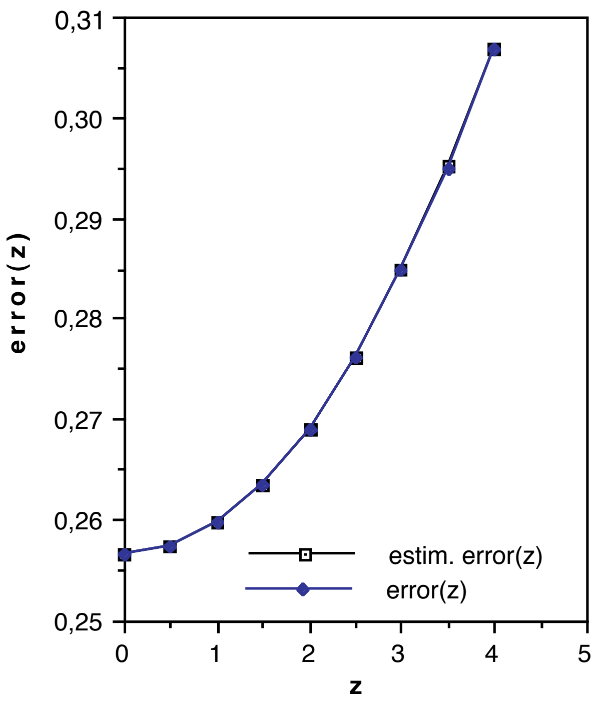
Figure 2.



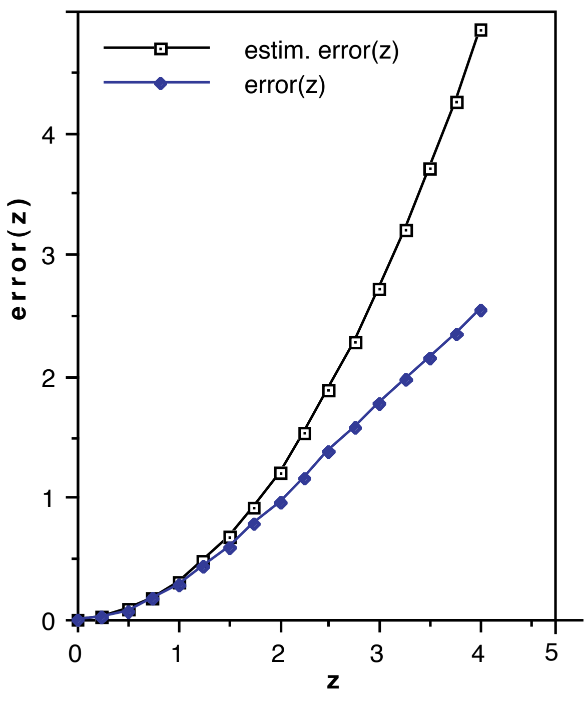
(a)



(b)



(c)



(d)

Figure 3.

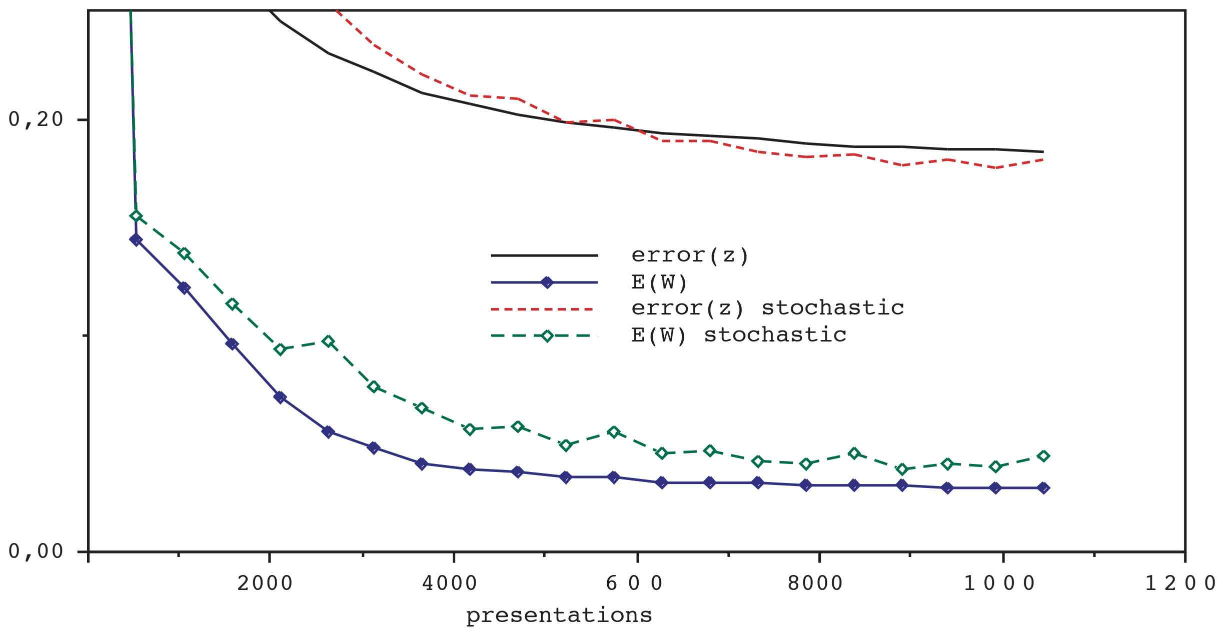
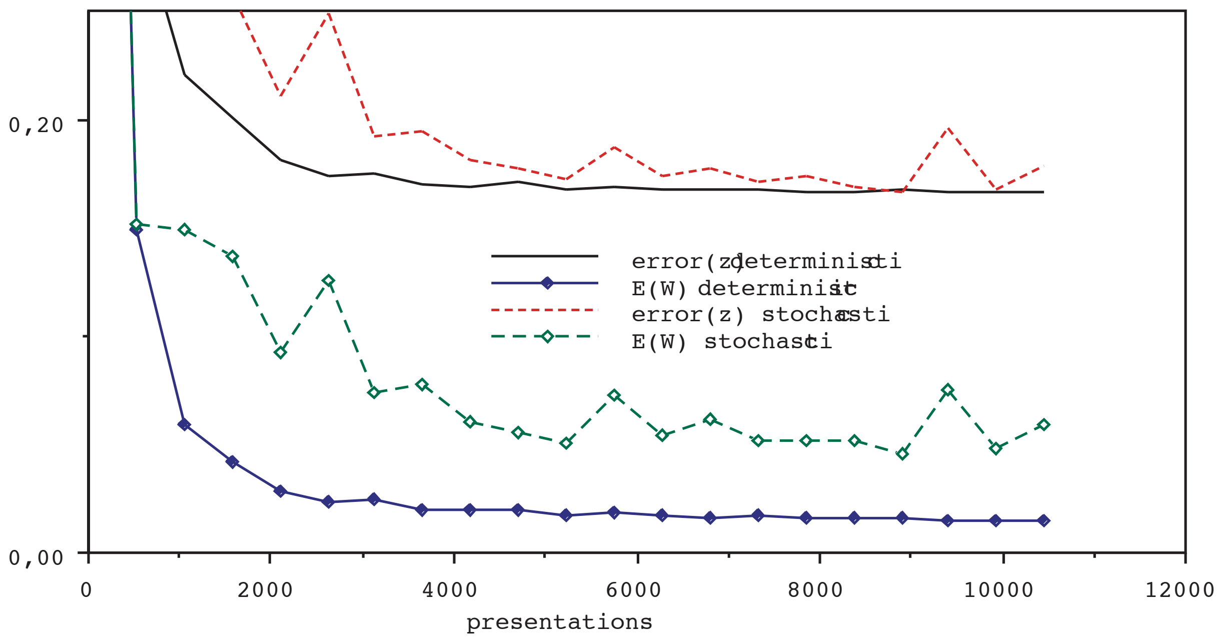
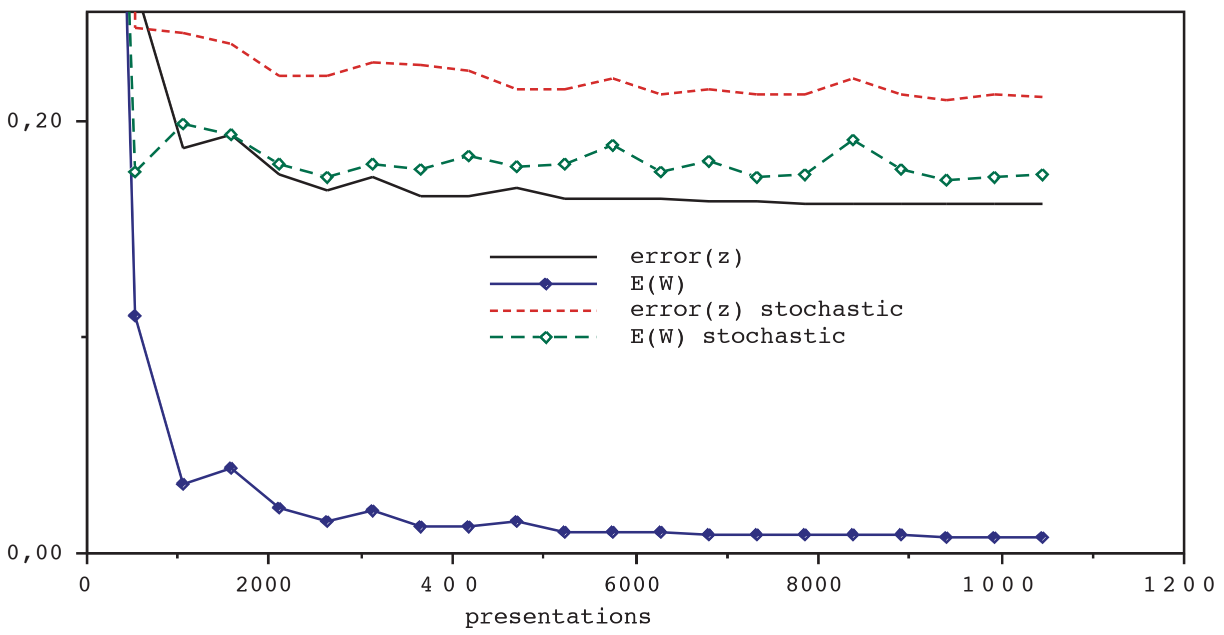

0,20
0,00
2000
4000
6 0 0
8000
1 0 0 0
1 2 0 0
presentations
error(z)
E(W)
error(z) stochastic
E(W) stochastic

Figure 4(a)



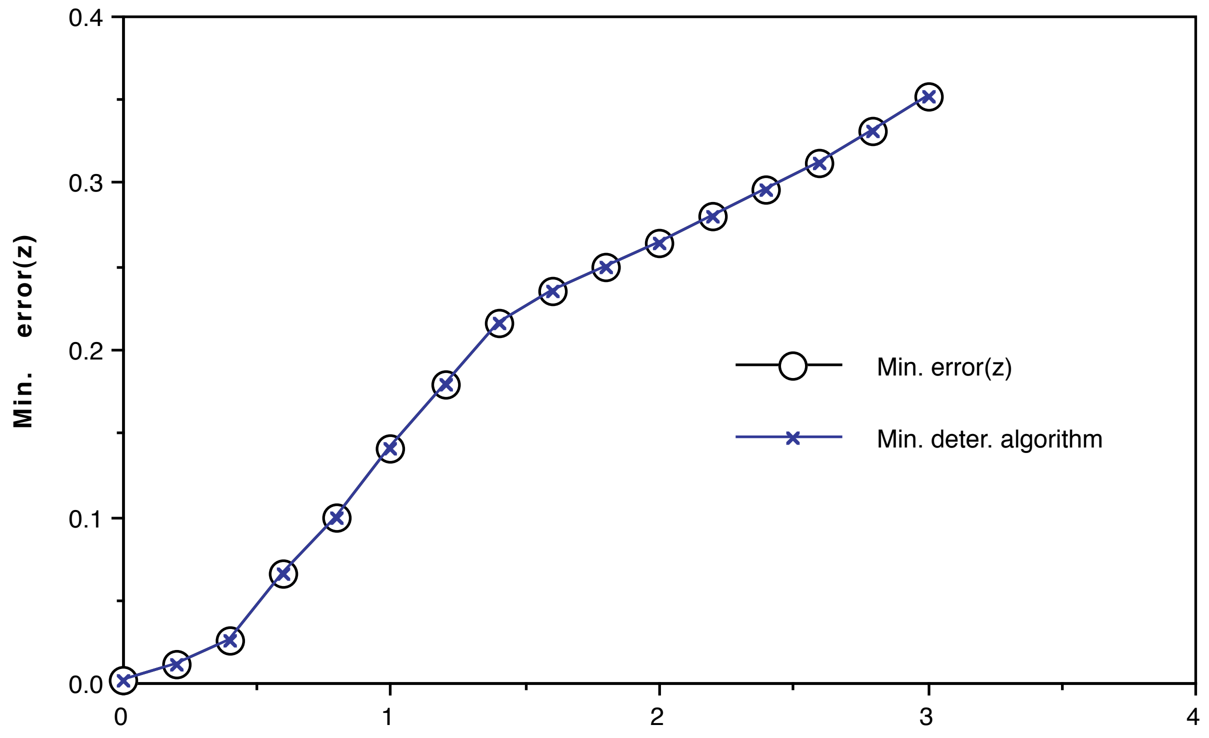
0,20
0,00
0
2000
4000
6000
8000
10000
12000
presentations
error(z) deterministic
E(W) deterministic
error(z) stochastic
E(W) stochastic

Figure 4(b).



0,20
0,00
0
2000
4 0 0
6000
8000
1 0 0 0
1 2 0 0
presentations
error(z)
E(W)
error(z) stochastic
E(W) stochastic

Figure 4(c).



0.4
0.3
0.2
0.1
0.0
0
1
2
3
4
Min. error(z)
Min. error(z)
Min. deter. algorithm

Figure 5.