Selecting Accurate, Robust, and Minimal Feedforward Neural Networks

Cesare Alippi, *Senior Member, IEEE*

***Abstract*—Accuracy, robustness, and minimality are fundamental issues in system-level design. Such properties are generally associated with constraints limiting the feasible model space. The paper focuses on the optimal selection of feedforward neural networks under the accuracy, robustness, and minimality constraints. Model selection, with respect to accuracy, can be carried out within the theoretical framework delineated by the final prediction error (FPE), generalization error estimate (GEN), general predicion error (GPE) and network information criterion (NIC) or cross-validation-based techniques. Robustness is an appealing feature since a robust application provides a graceful degradation in performance once affected by perturbations in its structural parameters (e.g., associated with faults or finite precision representations). Minimality is related to model selection and attempts to reduce the computational load of the solution (with also silicon area and power consumption reduction in a digital implementation). A novel sensitivity analysis derived by the FPE selection criterion is suggested in the paper to quantify the relationship between performance loss and robustness; based on the definition of weak and acute perturbations, we introduce two criteria for estimating the robustness degree of a neural network. Finally, by ranking the features of the obtained models we identify the best constrained neural network.**

***Index Terms*—Acute perturbations, constrained model selection, feedforward neural networks, robustness, sensitivity, system-level design.**

NOMENCLATURE

Z^N	Data set composed of $N(input, output)$ pairs (x, y).
Θ	Vector of the network weights.
Θ^o	Optimal weights.
$\hat{\Theta}$	Weights trained on Z^N.
$\hat{y}(\Theta) = f(\Theta, x)$	Neural network.
V_N	Training error function evaluated on Z^N.
$\bar{V}$	Generalization error.
$V'(\Theta) = \partial V(\Theta)/\partial\Theta$	Gradient vector. V can be either V_N or $\bar{V}$.
$V''(\Theta) = \partial^2 V(\Theta)/\partial\Theta^2$	Hessian matrix. V can be either V_N or $\bar{V}$.
$\boldsymbol{v}$	Vector of weights between the hidden and the output layers.
$\boldsymbol{w}$	Matrix of weights between the input and the hidden layers.
$\bar{p}, \hat{p}$	Effective number of parameters of the network and its estimate, respectively.
$\delta\Theta$	Perturbation vector affecting Θ.
δJ	Variation in generalization error induced by $\delta\Theta$.

Manuscript received July 26, 2001; revised May 6, 2002. This paper was recommended by Associate Editor X. Yu.

The author is with the Dipartmento Elettronica e Informazione, Politecnico di Milano, 20133 Milan, Italy.

Digital Object Identifier 10.1109/TCSI.2002.805710

I. INTRODUCTION

ONE of the fundamental problems involved in neural-network modeling is architecture optimization which aims at high generalization performance. The fact that performance maximization is a primary concern does not necessarily imply that it is the unique goal to be pursued. In solving an application, other requirements can in fact be taken into account, which derive from specifications, constraints, or methodological issues. In general, a compromise between performance (accuracy) and constraints satisfaction must be solved to obtain the "best" neural network for the specific application.

Examples of constraints are minimality (the smallest neural network solving the task) [1], smoothness (regularity and smooth behavior for the approximating neural model) [2], robustness (a graceful loss in performance when the network is affected by perturbations) [3], fault tolerance [4], [5], etc.

Constraints can be classified as application-oriented and implementation-oriented. Within the former class we have constraints related to some application features, e.g., approximating ability and smooth solution, in the latter, we address issues more related to implementation, e.g., minimality, robustness, and fault tolerance. The implementation can be software (SW) with code running on a host machine or realized with a dedicated hardware (HW), e.g., analog, digital, optical, or, again, mixed as it happens in HW/SW codesign where the application is suitably partitioned between HW and SW [6].

Time to market, rapid prototyping, and the incredible growth of complex embedded systems [7] is pushing the microelectronics computer-aided design research toward the development of automatic synthesis tools for designing a problem solution, or application, at a very high abstraction level. System-level design [8] is following this philosophy by integrating application-oriented and implementation-oriented constraints directly at the application characterization level. There, the minimality and the modularity properties of a solution are particularly appreciated since they allow for a compact and modular model description, which can be passed to the HW/SW codesign compiler for an effective code partitioning and subsequent HW synthesis [6].

A topologically small network implies less neurons to be implemented and, therefore, a reduced computational load (which

may be particularly appealing in real-time applications). For instance, if we consider a digital very large-scale implementation, topologically simpler network means less silicon area, a lower power consumption (e.g., think of embedded systems such as mobile telephones) and faster computation. Modularity, carried out at the neuron level, allows treating the neuron or an ensemble of neurons, as atomic units. Such units can then be implemented in parallel solutions by replicating the basic neuron macrocell, a single neuron in time multiplexed architectures (as it happens with sequential code execution) or as specific op-codes whose implementing HW based on field-programmable gate arrays (FPGAs) might be reconfigured on the fly during the code execution [9].

Robustness is important for a different reason. A robust network provides a graceful loss in accuracy when perturbations affect its parameters. Relevant examples of perturbations are finite-precision representations [10]–[12], fluctuations in the production process [13], [14], thermal and ageing effects in analog realizations [15], static and transient faults in physical devices. Hence, a robust network can tolerate lower accuracy (i.e., reduced number of bits) and be resilient to a class of faults [3].

Pursuit of robustness, modularity, and complexity requirements can be carried out at the application level during training by adding a regularization/penalty term to the training cost function so as to bias the solution toward networks possessing desirable characteristics (e.g., weight decay for penalising large weights [16], high curvature penalizations [2], extended Tikhonov regularisers for improving fault tolerance [4], [17], etc). Interestingly, a penalty term added to the training function implicitly addresses the performance/constraints tradeoff in the sense that the compromise is solved during training without any external interaction.

Differently, but in the same direction, robustness with respect to weights perturbation can be integrated during the training phase by adding a suitable noise to the weights up-date terms. The derived neural network becomes less sensitive to weights perturbation with an immediate positive effect on its final analog implementation (fluctuations in the weights production process are an example of perturbations [13]).

A different philosophy suggests tackling the issue off-line, after the training procedure has been completed. In this case, constraint satisfaction is tested directly onto the obtained model and, if possible, improved off-line. Examples in this direction are neuron pruning [18] and unnecessary weights removal [19]–[21] which improve accuracy, integrate smoothness requirements and reduce the network complexity (minimality constraint). In the same direction [22] and [23] suggest how to properly dimension the number of hidden units of a neural network, hence, following the minimality and modularity directives. Another example is network augmentation [4], [24] where a suitable replication of the network topology allows for improving the fault tolerance ability of the neural network by means of spatial redundancy.

In this paper, we follow the offline approach: the compromise between accuracy and constraints satisfaction is carried out at the end of the training phase by inspecting and ranking the accuracy and the level of constraints satisfaction of the obtained models. Model selection (with respect to accuracy) and constraints integration are therefore subsequent independent tasks; the designer first optimizes a set of neural architectures and only afterwards tests and tradeoffs accuracy and constraints to obtain the best neural network for the application.

Note that this is the key for an easy automatic synthesis of a neural-based application in which the nonexpert designer has to set few—possibly none- tuning parameters.

Accuracy, the most important application-oriented constraint, can be estimated with a validation criterion such as crossvalidation, k-fold crossvalidation, leave one out or, by considering neural topologies belonging to the same neural structure, with selection criteria such as network information criterion (NIC) [25], general predicion error (GPE) [26], [27], final prediction error (FPE), final prediction error biased (FBEB) [28]. Whichever is the selected method, we obtain a performance degree P_i for the envisaged neural network. Likewise, if we denote by C_i a measure of the constraint satisfaction for the best neural model M_i (it can be a vector if more then one constrain is considered) then each trained network is characterized by a triplet (M_i, P_i, C_i).

Computational complexity and robustness are surely two interesting high-level constraints. Computational complexity can be simply addressed by counting the number of hidden units or the network weights. Of course, the designer could also consider different figures of merit based on the number of fixed or floating point operations to be carried out. Minimality and computational complexity are strictly dependent properties.

As far as robustness is concerned, results suggested in the specific robustness/sensitivity literature mainly focus the attention on HW implementation with perturbations modeling finite precision representation errors [10]–[14], [29]. In this direction, we surely agree that a robustness analysis dealing with a single connection removal (see, e.g., [20]) is particularly interesting for its subsequent impact on power consumption and silicon area. Nevertheless, neuron reduction is even more relevant than synapse reduction for its direct influence on the network topology. Moreover, neuron reduction supports the modularity concept at the neuron level.

A more general robustness analysis dealing with a large class of perturbations affecting the neural computation must be derived to provide effective robustness criteria.

In this paper, we provide a novel sensitivity analysis to estimate the generalization performance loss induced by generic perturbations affecting the weights of feedforward neural networks of regression type. The interest for such networks composed of a nonlinear hidden layer and a single linear output derives from their universal function approximating ability [30].

To our knowledge, the suggested sensitivity analysis provides in a closed form, for the first time in the related literature, approximate relationships between training error, effective number of degrees of freedom used by the neural network, strength of the perturbation and variation in generalization error.

Moreover, the suggested sensitivity/robustness analysis relaxes the noise-free assumption of [11] and [12], it is not required to consider large networks as assumed in [29] nor a

large data set needed in [10] to make the mathematics more amenable. Only a sufficiently large, but limited, data set is required. Finally, two new robustness criteria can be derived from the sensitivity analysis which characterize the inherent robustness degree of the trained neural model.

The structure of the paper is as follows. Section II briefly describes the model selection criterion from which the suggested generalization-based sensitivity analysis is derived. Section III introduces the concepts of weak and acute perturbations to characterize some relevant classes of perturbations which lead, in Section IV, to the robustness criteria. Section V briefly introduces the parallel between robustness and fault tolerance and identifies those perturbations not affecting the network's performance. Finally, the model to be chosen is the one solving the compromise accuracy/minimality/robustness according to a tolerable tradeoff. An example of the selection procedure is given in Section VI.

II. Model Selection, Accuracy and Minimality

The neural computation implemented by the envisaged neural family can be described as

$$\hat{y}(\Theta) = \sum_{i=1}^{n} v_i O_i\left(w_i, x\right), \Theta = [v, w_1, \ldots w_d]$$

where n and d denote the number of hidden units and inputs, respectively. $\boldsymbol{v}$, of scalar components $v = [v_1, \ldots, v_n]$, represents the weights vector between the input and the output layers and $\boldsymbol{w}$ the weights matrix between the input and the hidden ones ($w = [w_1, \ldots, w_d]$, w_i being the column weight vector connecting the ith hidden unit of output O_i with the input vector x). Without loss of generality we consider $O_i(w_i, x)$ to be the hyperbolic tangent function applied to the scalar product between w_i and x.

The column vector Θ contains all the network weights; for ease of notation bias terms are not considered here but can be easily included in the analysis.

We denote by $\bar{y} = f(\Theta^o, x)$ the "true" unknown neural network with the optimal weights Θ^o and by $\varepsilon = WN(0, \sigma^2)$ a stationary i.i.d. noise of unknown variance influencing additively the true neural network so that $y = \bar{y} + \varepsilon$. We start by assuming that perturbations affecting the network weights are small in magnitude; the hypothesis will be partly relaxed later on.

The assumption requiring that the true function belongs to the set of functions represented by the neural network may be a strong hypothesis when a very limited data set is available. On the other hand, the assumption can be accepted when the dimension of the data set becomes larger. Such a hypothesis will be used to develop the approximated formulas for estimating the performance loss of the neural network subject to perturbations in its weights and not necessarily must be considered for model selection; more comments regarding the relationship between the model bias and the validity of the approximated formulas will be given at the end of the section.

Training neural model $\hat{y}(\Theta)$ requires minimization of the mean-square error loss function

$$V_N\left(\Theta, Z^N\right) = \frac{1}{2N} \sum_{i=1}^{N} \left(y - \hat{y}(\Theta)\right)^2 \tag{2.1}$$

evaluated over the finite set of (input, output) pairs

$$Z^N = \left\{\left(x_1, y_1\right), \ldots, \left(x_N, y_N\right)\right\}. \tag{2.2}$$

For their nature, regression-type neural networks constitute a nested architectures family $M : M_1 \subseteq M_2 \subseteq M_3 \subseteq \cdots \subseteq M_k \subseteq \cdots$, where k represents the number of hidden units: complex architectures degenerate in simpler ones with a suitable choice for weights.

To develop the sensitivity analysis based on the performance loss, we consider the model selection procedure suggested in [25]–[32]. To make the mathematics more amenable we do not consider regularization terms and high-order contributions; as a result the different theories can be grouped as follows.

Denote by $\hat{\Theta}$ an estimate of the unknown optimal parameter vector Θ^o obtained by training M_i with an efficient algorithm which minimizes (2.1). If we assume that inputs and noise are unrelated random variables and by considering a quasi-Newton Hessian (reasonable assumption according to [33] in the $\hat{\Theta}$ neighborhood), we have from [25], [26], [28], and [34] that

$$E\left[\bar{V}\left(\hat{\Theta}\right)\right] = \bar{V}\left(\Theta^o\right) + \frac{\sigma^2 \bar{p}}{2N} + O\left(\frac{1}{N^2}\right) \tag{2.3}$$

$$E\left[V_N\left(\hat{\Theta}, Z^N\right)\right] = \bar{V}\left(\Theta^o\right) - \frac{\sigma^2 \bar{p}}{2N} + O\left(\frac{1}{N^2}\right) \tag{2.4}$$

where

$$\bar{V}(\Theta) = \frac{1}{2} E\left[\left(y - \hat{y}(\Theta)\right)^2\right]. \tag{2.5}$$

In the above formulas, expectation is taken with respect to all the Z^N sets obtainable with N training pairs; $\bar{p} = \operatorname{rank}(\overline{V}''(\Theta^o))$ with $\overline{V}''(\Theta^o) = (\partial^2 \overline{V}(\Theta))/\partial\Theta^2|_{\Theta^o}$ is the effective number of parameters used by the neural network to fit the data (and it is not simply the number of the degrees of freedom of the neural network). Despite the fact that derivations of (2.3) and (2.4) are quite complex and outside the goal of the paper, we can provide an intuitive interpretation to them. De facto, $\overline{V}(\Theta^o)$ represents the true validation error of the optimal neural network while $E[\overline{V}(\hat{\Theta})]$ represents the average validation error associated with all possible data sets composed of N data. Equation (2.3) states that if we have a limited data set then the averaged generalization error is greater than the optimal one ($\overline{V}(\Theta^o)$ term). Apart from high order contributions, the increment in generalization error is due to the noise in the data (σ^2 term) and the fact we have configured $\bar{p}$ degrees of freedom by using a limited data set of cardinality N.

Instead, (2.4) addresses the relationship between the generalization error of the true neural model and its averaged training error. Basically, it states that the averaged training error is smaller than the validation one of the optimal model: the training error is an optimistic estimate of the validation one.

When we have a large training set, the correction terms vanish and the training error becomes a good estimate of the validation error.

By manipulating (2.3) and (2.4) we obtain that

$$E\left[\bar{V}\left(\hat{\Theta}\right)\right] = E\left[V_N\left(\hat{\Theta}, Z^N\right)\right] + \frac{\sigma^2\bar{p}}{N} + O\left(\frac{1}{N^2}\right). \quad (2.6)$$

A more general formulation, which takes into account a regularization term, can be found in [27] where two definitions for the effective number of degrees of freedom are given; when no regularization terms are envisaged results coincide anyway with (2.6). Equation (2.6) states that the averaged training error is a too optimistic estimates for the generalization error which must be increased of the $\sigma^2\bar{p}/N$ correction term.

Directly from its definition and under the assumed hypotheses, we have from (2.5) that $2\bar{V}\left(\Theta^o\right) = \sigma^2$. By neglecting the high-order terms in system (2.3) and (2.4) and substituting σ^2 with $2\bar{V}(\Theta^o)$ we can remove the dependency on $\bar{V}\left(\Theta^o\right)$. We end with the final prediction error like expression [26], [28], [32]

$$E\left[\bar{V}\left(\hat{\Theta}\right)\right] \cong E\left[V_N\left(\hat{\Theta}, Z^N\right)\right]\frac{N+\bar{p}}{N-\bar{p}}. \quad (2.7)$$

Equation (2.7) states that the averaged validation error is larger than the averaged training error with an amplification term depending on the number of available data and the number of degrees of freedom used by the neural network to learn the input/output relationship.

Unfortunately, we have only one Z^N data set and, therefore, we cannot take the average required in (2.7); this introduces a statistical fluctuation, function of the random variable $\bar{V}(\Theta^o) - V_N(\Theta^o)$. Moreover, $\bar{p} = \mathrm{rank}(\bar{\mathrm{V}}''(\Theta^o))$ is unknown but it can be estimated as $\hat{p} = \mathrm{rank}(\mathrm{V}''_{\mathrm{N}}(\hat{\Theta}))$ (see also, [28]).

Finally, the generalization ability of model M_i becomes [25], [32]

$$\bar{V}\left(\hat{\Theta}\right) \cong V_N\left(\hat{\Theta}\right)\frac{N+\hat{p}}{N-\hat{p}} + l\left(\bar{V}\left(\Theta^o\right) - V_N\left(\Theta^o\right)\right). \quad (2.8)$$

Since $l(\bar{V}(\Theta^o) - V_N(\Theta^o))$ is an unknown function in $\bar{V}(\Theta^o) - V_N(\Theta^o)$, dependent only on the particular realization of Z^N, it is not possible to estimate the generalization ability of the neural network by using (2.8).

Anyway, such a term is constant along the model hierarchy M. As done in [28] we can derive a model selection criterion by considering the biased accuracy performance degree for each model

$$\begin{aligned} P_i =& P\left(\hat{\Theta}\right) \cong V_N\left(\hat{\Theta}\right)\frac{N+\hat{p}}{N-\hat{p}} \\ \hat{p} =& \mathrm{rank}(\mathrm{V}^+_{\mathrm{N}}) \end{aligned} \quad (2.9)$$

which differs of $l(\bar{V}(\Theta^o) - V_N(\Theta^o))$ and approximations from the true generalization value. Values $P_i s$ given by (2.9) can be used to select the best model, which is the one characterized by the minimum value of the performance index (it must be clear that P_i does not represent a measure for the generalization ability of the model). As we already discussed in Section I, cross-validation based techniques can be used instead of (2.9) as model selection criteria.

We focus again the attention on (2.8) for its relevant role in deriving the performance-based sensitivity analysis. We note that the generalization ability of the trained neural network depends on $l(\bar{V}(\Theta^o) - V_N(\Theta^o))$ which, even if unknown, is function of Θ^o and not $\hat{\Theta}$.

As such, when the neural network is affected by perturbations on $\hat{\Theta}$ the $l(\bar{V}(\Theta^o) - V_N(\Theta^o))$ term is constant, it does not contribute to the variation in generalization ability and the perturbation analysis of (2.8) coincides with that of (2.9).

The assumption requiring that the application to be learned can be described by a neural network (completeness property) is the most critical one and it has an immediate impact on the sensitivity analysis. The hypothesis is not severe when we have a large N. In addition, the hypothesis should hold in many applications ([27] claims that that regression-type neural networks can be regarded as quasicomplete since they are capable of approximating a large class of functions).

If the neural model is incomplete, as it might happen, then the analysis is different. We study first the case in which the neural network "is close to" the process generating the data (the model is quasicomplete). The estimation error is biased so that $y - \hat{y}(\hat{\Theta}) = \eta = \varepsilon + \rho$ where ρ is the zero-mean bias term of variance $E[\rho^2] = \delta^2$. If we carry out the analysis leading to (2.3) and (2.4) we obtain that the second term of (2.6) becomes $(\sigma^2\bar{p} + T(\delta, \Theta^o))/N$ with $T(\delta, \Theta^o) \leq \bar{p}\delta^2$. Therefore, if the bias/noise "noise to signal" ratio is $\delta^2/\sigma^2 \ll 1$, i.e., the bias contribution is small, we can neglect the $T(\delta, \Theta^o)$ term and (2.6) is still valid.

Conversely, in the case of a strong bias, a correction term must be added to (2.7) which is a nonlinear function of $\bar{p}, \delta^2$ and N[28]; anyway, for a sufficiently large N such term should vanish. If we cast serious doubts about the fact the model is complete then we should consider the GEN criterion [26] to compute P_i or crossvalidation-based criteria.

Anyway, also for the incomplete case, we will consider the robustness criteria derived from (2.9). This relies on the intuitive, but unorthodox, assumption that the punctual bias term added to the noise is equivalent to a realization of a different noise with increased variance. In this case, the suggested robustness criteria are only approximations but become consistent when N tends to infinity.

Whichever is the validation/selection criteria, the best model according to accuracy is the one minimizing P_i.

If we rank the (M_i, P_i, R_i) triplets according to the performance index P_i, the minimality under the modularity requirement can be simply considered. In fact, if P_k is the performance index of the best model and β is the tolerable loss in accuracy, the smallest jth neural network satisfying $P_k - P_j < \beta$ is the one to be selected. Such model leads to a $k - j$ hidden neurons gain in topological complexity.

III. Weak, Acute Perturbations and Robustness

The robustness of a given neural network affected by perturbation $\delta\Theta$ on its performance can be studied by means of a sensitivity analysis applied to (2.8). Since $\hat{p}$ is not a continuous

function of Θ (being the rank of a matrix) we cannot simply differentiate the (2.8). Denote by δp the variation in rank associated with $\delta\Theta$ then

$$\delta J \cong \bar{V}\left(\hat{\Theta}+\delta\Theta\right)-\bar{V}\left(\hat{\Theta}\right)\cong V_N\left(\hat{\Theta}+\delta\Theta\right)\frac{N+\hat{p}+\delta p}{N-\hat{p}+\delta p} -V_N\left(\hat{\Theta}\right)\frac{N+\hat{p}}{N-\hat{p}}. \quad (3.1)$$

By expanding with Taylor $V_N(\hat{\Theta}+\delta\Theta)$ around $\hat{\Theta}$ and remembering that the gradient is null since the training procedure ended in a minimum, we obtain that $V_N(\hat{\Theta}+\delta\Theta) = V_N(\hat{\Theta})+1/2\delta\Theta^T V_N''(\hat{\Theta})\delta\Theta$ which, inserted in the above, provides

$$\delta J \cong \frac{1}{2}\delta\Theta^T V_N''\left(\hat{\Theta}\right)\delta\Theta\frac{N+\hat{p}+\delta p}{N-\hat{p}-\delta p}+\frac{\hat{\sigma}^2\delta p}{N-\hat{p}-\delta p} \quad (3.2)$$

where $\hat{\sigma}^2 = V_N(\hat{\Theta})2N/(N-\hat{p})$ is an estimate of the noise variance (e.g., see [28]).

It is useful to express (3.2) in a canonical form which decouples the dependence in $\boldsymbol{v}$ and $\boldsymbol{w}$. From the quasi-Newton approximation for the Hessian we have that $V_N''(\hat{\Theta}) = 1/N\sum\nabla_i\nabla_i^T$, where the gradient $\nabla_i = \partial\hat{y}(\Theta,x)/\partial\Theta|_{x_i,\hat{\Theta}}$ is the $n(d+1)$-dimensional column vector $\nabla_i = [O_i; v_1 o'_{1,i}x;\cdots v_n o'_{n,i}x]$ (components are separated by semicolons for clarity). In particular, the first entry refers to differentiation with respect to $\boldsymbol{v}$, the following d entries to $\boldsymbol{w}_i s$. $o'_{j,i}$ is the first derivative of the jth hidden neuron output evaluated on the ith input pattern. In the canonical form the Hessian becomes $V_N(\hat{\Theta})'' = \Theta_v H_w \Theta_v$ where $H_w = 1/N\sum_{i=1}^{N}\tilde{\nabla}_i\tilde{\nabla}_i^T$, $\tilde{\nabla}_i^T = [O_i^T; o'_{1,i}x^T;\cdots o'_{n,i}x^T]$ and Θ_v is the diagonal matrix of order $n(d+1)$ such that $diag(\Theta_v) = (1_n; v^1;\cdots v^n)$, i.e., n ones followed by the v^i vectors composed of d components equal to v_i. In the canonical form Θ_v contains only information about $\boldsymbol{v}$ while H_w only information about $\boldsymbol{w}$. Note that $V_N''(\hat{\Theta})$ is a semidefinite positive matrix iff H_w is semidefinite positive. Finally, the canonical form for (3.2) becomes

$$\delta J \cong \frac{1}{2}\delta\Theta^T\Theta_v H_w\Theta_v\delta\Theta\frac{N+\hat{p}+\delta p}{N-\hat{p}-\delta p}+\frac{\hat{\sigma}^2\delta p}{N-\hat{p}-\delta p}. \quad (3.3)$$

Equations (3.2), and (3.3) state that the effect of $\delta\Theta$ on the variation in generalization performance δJ is the sum of two contributions. The first term is related to the sensitivity of the neural network (Hessian matrix) and depends on the particular structure of M. Note that such a term is nonnegative being the quadratic form semidefinite positive.

The second term is related to the presence of noise and its sign depends on δp. A decrement in $\delta\Theta$ implies a reduction in the effective number of parameters ($\delta p < 0$). When $\delta p < 0$ and $\delta J < 0$ we have that the reduction in model variance more then compensates the increase in the training error, leading to a lower expected validation error.

This case is interesting and can be related to the principal-component-pruning (PCP) technique suggested in [18]. In fact, under the assumption $N \gg p > \delta p$, the authors identified those $v_i s$ whose removal is associated with a $\delta J < 0$.

Our analysis is different and represents an extension. Equation (3.2) estimates the variation in accuracy induced by any kind of perturbation affecting the weights.

It is interesting to realize that if perturbations affect only v, then we can remove the small perturbation hypothesis and (3.2) holds for any perturbation magnitude; conversely, for perturbations affecting w the small perturbation assumption cannot be relaxed.

The rest of the section is devoted to investigate the effect of $\delta\Theta$ on (3.2), and (3.3), by focusing the attention on those relevant perturbations which do not modify $\hat{p}$.

Definition: We say that a perturbation $\delta\Theta$ is weak if it does not eliminate weights.

The immediate consequence of the definition is that a weak perturbation does not change the topological structure of the network. If the perturbation is not weak, say strong, e.g., with respect to v, the structure of the network changes and it might scale down the hierarchy (as it happens with the PCP technique). The impact of strong perturbations must be taken into account separately and their effect must be evaluated with (3.2); for instance, if we remove weight $\boldsymbol{v}_i$ then $\delta J = (1/2v_i^2 a_{i,j}(N+\hat{p}+1))/(N-\hat{p}-1)$ where $a_{i,i}$ is the ith diagonal component of the Hessian.

Lemma 1: A continuous[1] perturbation $\delta\Theta$ (e.g., δv or δw_i) is weak, with probability one (w.p. 1).

The lemma is intuitive. In fact, strong perturbations imply connection removal. The strong perturbation set is discrete and contains $\sum_{i=1}^{k}\binom{k}{i} = 2^k - 1$, $\dim(\Theta) = k$ points, each of which is associated with a connection removal combination; the Lebesgue measure of such set is anyway null.

Definition (Taken From [35]): We say that the square matrix A_p obtained by perturbing the matrix A is acute (and the associate perturbation is said to be acute) iff $\lim_{A_p\to A}\mathrm{rank(A_p)} = \mathrm{rank(A)}$.

With $A_p \to A$ we are assuming that, according to some measure (e.g., the variance of the perturbation or its magnitude), the perturbation affecting A tends to zero.

As a consequence of the definition, an acute perturbation does not change the rank of a matrix.

The concept of acute perturbation is powerful. In fact, if $\delta\Theta$ is acute then $\delta p = 0$ and (3.2) nicely reduces to

$$\delta J \cong \frac{1}{2}\delta\Theta^T V_N''\left(\hat{\Theta}\right)\delta\Theta\frac{N+\hat{p}}{N-\hat{p}}. \quad (3.4)$$

As a main consequence, an acute perturbation never decreases the training error and does not improve the generalization ability of the neural network. This follows from the fact that under the small perturbation hypothesis there it exists a neighborhood of $\hat{\Theta}$ so that $(\partial V_N((Z_N,\hat{\Theta})/\partial\Theta = 0$ and that $V_N''(\hat{\Theta})$ is semidefinite positive.

The next theorem provides conditions under which perturbations affecting separately $\boldsymbol{v}$ and $\boldsymbol{w}$ are acute. In the following, we assume that there are not always saturated hidden units in the sense that their output is constant for each input pattern. If this is not the case we have first to reconfigure the network by removing the saturated neurons and add $\sum v_j o_j$ to the bias of

[1]We say that a perturbation is continuous if $\Pr(\delta\Theta = \delta\bar{\Theta}) = 0, \forall\delta\bar{\Theta}$

the output neuron ($v_j s$ and $o_j s$ are the weights and the outputs of the removed neurons, respectively).

We also envisage application of a principal component analysis (PCA) technique [30] to have linearly independent inputs.

Theorem:

a) If δv is a weak continuous perturbation, then δv is acute w.p. 1.
b) The continuous perturbation δw_i is acute w.p. 1 iff the induced $H_v = (1/N)\sum_{i=1}^{N} O(x_i)O(x_i)^T$ is acute.
c) If $\mathrm{rank(H_{v,p})} - \mathrm{rank(H_v)} = \mathrm{k}$, then $|\delta p| = k(d+1)$.

See the appendix for the proof.

Corollary 1: A continuous perturbation $\delta\Theta$ is acute w.p. 1 iff H_v is acute.

The corollary directly follows from the theorem. In fact, if we consider $\delta\Theta = [\delta v; \delta w_1, \ldots, \delta w_n]$ and the canonical form for the Hessian we have that perturbations affecting $\boldsymbol{v}$ are independent from those affecting $\boldsymbol{w}$. The proof follows from statement a) since the acute perturbation δv does not change the rank of the Hessian w.p. 1 and from b) since the rank may change iff H_v is acute.

Corollary 2: A generic continuous perturbation $\delta\Theta$ is acute w.p. 1 if $\mathrm{rank(H_v)}$ is full, otherwise, the perturbation is not acute w.p. 1.

The consequences coming from the theorem and subsequent corollaries are important. In fact, the theorem states that all perturbations are acute w.p. 1 under the assumption that hidden units are linearly independent. As a main consequence, the (3.4) holds. Conversely, if H_v has not full rank, a generic continuous perturbation $\delta\Theta$ will increase the rank w.p. 1. This is intuitive if we think that a continuous perturbation δw will make the rank of H_v full w.p. 1.

The rank degeneration arises only with subtle perturbations δv and δw which introduce a linear relationship among hidden units; such perturbations are strong since they modify the structure of the network but the probability of extracting them for a continuous perturbation is again null.

Note that we are not limiting the analysis to static perturbations, e.g., time-invariant perturbations since we can apply the method also to dynamic perturbations. In such a case, the error on the network output vanishes if the perturbation is transient.

This observation is of some relevance in a physical implementation where transient faults might affect the weights, as it happens with analog implementations (e.g., see [13] for a solution to this problem). Such faults are extremely difficult to be identified for their spurious behavior. If we select a robust neural network, their effect on the generalization error will be attenuated.

We note that perturbations affecting inputs and internal computation for hidden units are equivalent to static/transient perturbations δO affecting the hidden units output. Point b) of the theorem allows us to extend the validity of the analysis to a wider set of perturbations affecting the neural computation.

To this end we provide a lemma granting that a perturbation δO (which therefore covers all perturbations affecting the computation in the input/hidden layer) is acute.

Lemma 2: Denote by $\lambda_{\min}(H_v)$ the smallest nonnull eigenvalue of H_v. Perturbation δH_v is acute if $||\delta H_v||_2 < \lambda_{\min}(H_v)$ H_v and the perturbation matrix δH_v (associated with perturbations δw or δO) are symmetric matrices by construction. We invoke the Weyl's theorem [35], [36] which states that for each eigenvalue the relationships $\lambda_{\min}(\delta H_v) + \lambda(H_v) \leq \lambda(H_{v,p}) \leq \lambda(H_v) + \lambda_{\max}(\delta H_v)$ holds. As a consequence, by reminding that $\lambda_{\min}(\delta H_v) = 0$, we can write that $|\lambda(H_{v,p}) - \lambda(H_v)| \leq \lambda_{\max}(\delta H_v) = ||\delta H_v||_2$. The rank degenerates if at least one $\lambda(H_{v,p})$ goes to zero which would imply $|\lambda(H_v)| \leq ||\delta H_v||_2$. Since the relationship must hold for any eigenvalue of H_v, rank degeneration cannot occur if we require that $||\delta H_v||_2 < \lambda_{\min}(H_v)$.

In a way, the lemma is strictly related to PCA and other techniques connected with hidden unit removal [21], [30]. In a principal-axis representation, where each eigenvalue is on its coordinate axis, the perturbation does not annihilate the weaker dimension of the space if the spectral radius of the perturbation has not enough "energy."

If we consider a digital realization in which the perturbation is associated with quantization (e.g., rounding or truncation [10], [11]) of the weights, it is reasonable to assume that quantization does not remove any weight and, hence, quantization is an acute perturbation. Also, in analog implementations where components (e.g., resistors or capacitors) are affected by the production process we have that the associate perturbations are acute [they are not acute only if the element breaks; in such a case we have to consider (3.2)].

In the following section we will always consider acute perturbations consequent to a proper hidden layer dimensioning; therefore, we can assume that the rank of H_v is full.

IV. Robustness Criteria for Feedforward Neural Networks

In the section, we introduce two criteria for evaluating the robustness degree of a feedforward neural network. The first criterion addresses the worst case perturbation (WCP), namely it quantifies the maximum loss in accuracy associated with a generic acute perturbation. On the basis of the magnitude of the maximum error we decide whether tolerating or not a class of perturbations which might affect the neural network during its operational life. Another interesting case is the mean case perturbation (MCP), which quantifies the average increase in generalization error associated with acute perturbations.

Lemma 3: Given an acute perturbation $\delta\Theta$ then $0 \leq \delta J \leq (1/2)\lambda_{\max}(V_N'')|\delta\Theta|^2((N+\hat{p})/(N-\hat{p}))$. The equality holds when $\delta\Theta$ is a vector parallel to the eigenvector associated with $\lambda_{\max}(V_N'')$.

Since (3.4) is a semidefinite positive quadratic form, the lower bound in accuracy is 0. Conversely, the maximum loss in accuracy arises when the perturbation vector $\delta\Theta$ is parallel to the eigenvector $u_{\lambda\_\max}$ associated with the maximum eigenvalue $\lambda_{\max}$ of V_N''[37].

As such, Lemma 3 states that the worst perturbation is $\delta\Theta = |\delta\Theta| u_{\lambda\_\max}$ and quantifies its impact on δJ. Now, if a trained neural network exactly contains a smaller trained neural network then, by the inclusion principle [36], the larger networks have a $\lambda_{\max}$ larger than the one associated with the smaller network: such larger models worsen the WCP.

Lemma 4 addresses the MCP issue by evaluating the average loss in accuracy induced by a class of perturbations.

Lemma 4: Let $\delta\Theta$ be an acute perturbation with *i.i.d.* components of variance $\sigma^2_{\delta\Theta}$, then

$$E_{\delta\Theta}[\delta J] \cong \frac{1}{2}\sigma^2_{\delta\Theta}\frac{N+\hat{p}}{N-\hat{p}}\sum_{i=1}^{p}\lambda_i\left(V''_N\right)$$

The proof follows by taking expectation in (3.4) with respect to the $\delta\Theta$ domain

$$\begin{aligned}E_{\delta\Theta}[\delta J] =& \frac{1}{2}\frac{N+\hat{p}}{N-\hat{p}}E_{\delta\Theta}\left[\delta\Theta^T V''_N\left(\hat{\Theta}\right)\delta\Theta\right]\\ =& \frac{1}{2}\frac{N+\hat{p}}{N-\hat{p}}E_{\delta\Theta}\left[tr\left(V''_N\left(\hat{\Theta}\right)\delta\Theta\delta\Theta^T\right)\right]\\ =& tr\left(\frac{1}{2}V''_N\frac{N+\hat{p}}{N-\hat{p}}E_{\delta\Theta}\left[\delta\Theta\delta\Theta^T\right]\right)\\ =& \sigma^2_{\delta\Theta}\frac{1}{2}\frac{N+\hat{p}}{N-\hat{p}}tr\left(V''_N\right) = \sigma^2_{\delta\Theta}\frac{1}{2}\frac{N+\hat{p}}{N-\hat{p}}\sum_i\lambda_i\left(V''_N\right).\end{aligned}$$

The first manipulation follows from [37] and [38] while the second is associated with the fact that expectation and trace operators are linear and that expectation is taken over the $\delta\Theta$ dominion (which is independent from Θ).

The lemma covers the interesting case of perturbations associated with quantization of the network weights. In such a case, the small perturbation hypothesis holds and the noise variance associated with the q least significant bits truncation is $2^{-q}/3$ [11]. Therefore, lemma 4 can be used to dimension the word-length for coefficients by considering δJ as figure of merit and not the function output as suggested in [10] and [29]. Given a tolerable loss in performance and an application we can obtain the q satisfying the loss accuracy requirement.

In an analog implementation, if the tolerance of a component due to the production process assumed to be ruled by a Gaussian distribution is T, then the standard deviation of the Gaussian is $3T$ and $\sigma^2_{\delta\Theta} = 9T^2$. Given a tolerable loss in performance, according to Lemma 4 we identify the acceptable T and, hence, we characterize the parameter production process.

Two criteria for measuring the robustness of a neural network derive directly from lemmas 3 and 4. The first criterion provides a robustness degree on the basis of the WCP, the second by considering the average case MCP

$$R_{\mathrm{WCP}} = \lambda_{\max}\left(V''_N\right)\frac{N+\hat{p}}{N-\hat{p}} \tag{4.1}$$

$$R_{\mathrm{MCP}} = \sum_{i=1}^{p}\lambda_i\left(V''_N\right)\frac{N+\hat{p}}{N-\hat{p}}. \tag{4.2}$$

Note that the criteria do not take into account the magnitude of the perturbation since we are interested in a measure of the robustness degree for the neural network.

We can weaken some hypotheses or obtain finer relationships in Lemmas 3 and 4 by specialising the perturbations to some relevant cases affecting weights $\boldsymbol{v}$ and $\boldsymbol{w}$ and, therefore, the computation locally at the neuron level.

As a direct consequence of Lemma 4 we have that an *i.i.d.* perturbation with variance $\sigma^2_{\delta v}$ affecting $\boldsymbol{v}$ is acute w.p. 1 and $E_{\delta v}[\delta J] \cong \sigma_{\delta v}(1/2)(N+\hat{p})/(N-\hat{p})\sum_{i=1}^{n}\lambda_{i,v}$.

The expression shows an interesting relationship between the information provided by the data and the number of hidden units (the sum is now extended up to index n, i.e., the number of hidden units). Such a relationship becomes much more explicit when we consider a uniform distribution for activation values or inputs as we do in the following examples. In that case, we can show that the average perturbation increases linearly with the number of hidden units and, therefore, larger neural networks are less robust.

Example 1: Uniform Distribution for Neuron Activations: We assume that the activation value X of a generic hidden neuron is uniformly distributed within the symmetric interval of extreme α, that the activation function is the hyperbolic tangent $Tgh(X)$ and that hidden units are linearly independent. It is simple to prove, by integrating with the variable substitution $e^{2x} = z$ and then by parts, that

$$E\left[X^2\right] = 1 - \frac{Tgh(\alpha)}{\alpha} \tag{4.3}$$

which represents the variance of the neuron output. The evolution of the function with respect to α is given in Fig. 1. If the interval extreme is above $\alpha = 5$, then the variance of the output is above 0.8.

The eigenvalues sum is the trace of the Hessian H associated with the hidden units namely

$$tr(H) = \sum_{i=1}^{n} 1 - \frac{Tgh\left(\alpha_i\right)}{\alpha_i} = O(n). \tag{4.4}$$

If each unit is always activated with a sufficiently large α then the trace grows at most linearly with the number of hidden units. For $\lambda_{\max}$ we can write that

$$\frac{tr(H)}{n} \le \lambda_{\max} \le tr(H) \to 1 \le \lambda_{\max} \le O(n). \tag{4.5}$$

When n increases the maximum eigenvalue of the Hessian is upperly bounded by a linear growth.

Example 2: Uniform Distribution for Inputs: As a second example we consider a regression-type neural network mapping a $y = y(u), y, u \in \Re^1$ function. Inputs are uniformly extracted from the $[-\alpha_I, \alpha_I]$ interval (e.g., $\alpha_I = 1$). If the generic neuron has a vector $w = [\bar{w}, \beta]$ where $\bar{w}$ is the weight connecting the input and β the neuron bias, we obtain that

$$E[X^2] = 1 + \frac{1}{\bar{w}\alpha_I}\left[\frac{e^{2a} - e^{2b}}{(1+e^{2a})(1+e^{2b})}\right] \tag{4.6}$$

where $a = -\bar{w}\alpha_I + \beta, b = \bar{w} + \beta$. If we assume inputs normalized in the $[-1, 1]$ interval, then

$$tr(H) = n + \sum_i \frac{1}{\bar{w}}\left[\frac{e^{2a_i} - e^{2b_i}}{\left(1+e^{2a_i}\right)\left(1+e^{2b_i}\right)}\right] \tag{4.7}$$

$a_i = -\bar{w}_i + \beta, b_i = \bar{w}_i + \beta$. Again, if $\bar{w}_i$ s are sufficiently large then, as in example 1, the asymptotic behavior of $tr(H)$ and $\lambda_{\max}$ shows a linear behavior.

The adherence of such expressions with theoretical results holds even with low N s.

To show this, a set of experiments has been carried out on a teacher neural network with four hidden units; data have been

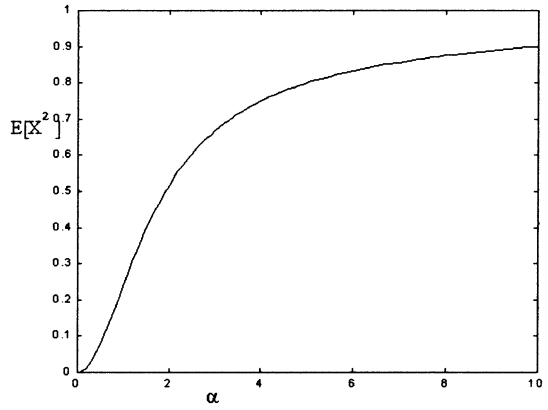

Fig. 1. The $E[X^2]$ function.

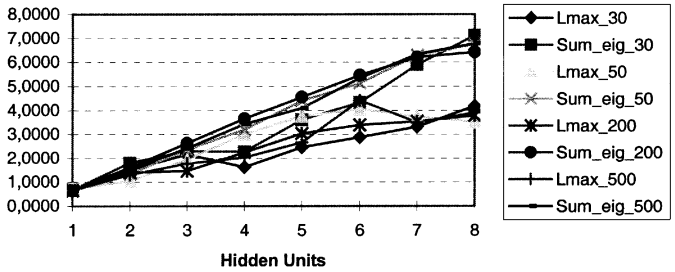

Fig. 2. The evolution of $\lambda_{\max}$ (Lmax) and $\sum \lambda_i$ (Sum_eig) w.r.t. increasing N.

extracted from a uniform distribution and a white Gaussian noise WGN(0,0.01) has been added to the network output.

We considered different training sets with increasing cardinality $N = 30$, 50, 200, and 500. For each data set, training was performed until the gradient vector was null. Fig. 2 shows the evolution over different hidden units of the maximum eigenvalue and the sum of eigenvalues (trace of H). Apart from fluctuations we can appreciate the linear behavior of $\lambda_{\max}$ and that of the eigenvalues sum.

V. Application-Level Fault Tolerance

We say that a digital circuit is fault tolerant if the occurrence of errors (e.g., induced by defects or transient signals) is masked at the device output.

We can extend the definition to address non digital computation.

Definition: We say that a neural network is fault tolerant with respect to a class of perturbations Π if their effect on the chosen figure of merit is null.

The fault-tolerance requirement can be weakened if we say that the increase in error on J is below a predefined threshold. This aspect has been nicely addressed in [4] where the authors do prefer to speak about small perturbation fault tolerance rather than stuck-at faults; as a consequence they implicitly consider the robustness features of the neural network as studied in previous sections.

Lemma 5: An acute perturbation $\delta\Theta$ does not introduce a generalization loss iff $\delta\Theta$ belongs to the null space $\boldsymbol{N}(V_N'')$ or, equivalently, $\delta\Theta\Theta_v$ belongs to the null space $\boldsymbol{N}(H_w)$. A generic continuous perturbation introduces a generalization loss w.p. 1.

Proof: Since the perturbation is acute we have

$$\delta J = \delta\Theta^T \Theta_v H_w \Theta_v \delta\Theta \frac{1}{2} \frac{N+\hat{p}}{N-\hat{p}}.$$

The proof follows immediately since the presence of the perturbation directly affects the generalization error. On the other hand, if the perturbation is not acute, its effect on the generalization performance is not visible only when $\delta J = 0$. This only happens if the perturbation spans the null space generated by $\boldsymbol{N}(V_N)$. The probability of generating a continuous perturbation belonging to a subspace, is anyway null.

We could think that by increasing the number of hidden units we introduce some sort of redundancy and, therefore, the network becomes more fault tolerant with respect to perturbations. Note that the attention is on fault-tolerance, not on robustness.

Lemma 6: No improvement arises by increasing the number of hidden units with w.p. 1.

In fact, by increasing the number of hidden units we may only increase the dimension of the null space; nevertheless, the probability of not observing the effect of a continuous perturbation is still zero. Again, this assertion is not in contrast with [4] for the different theoretical setup and the assumed fault mode.

VI. System-Level Neural Design: An Example

The experimental section shows an example for selecting a model M_i within $\boldsymbol{M}$ in a constrained environment.

The application refers to the development of an embedded neural device for classifying numerical digits (from 0 to 9); the MNIST database has been considered for training and validation. As such, a digit is represented by a 28×28-pixels matrix in a 8-b grayscale coding. The size of the network, and hence, its complexity, is relevant since there are 784 input neurons. The training set was composed of $N = 800$ training samples randomly extracted from the database while a set of 1000 patterns has been considered for validation purposes.

A. Model Selection in a Constrained Environment

We consider a target digital architecture in which a neuron is the atomic unit implemented by means of a digital macrocell of fixed structure stored in an appropriate library (the modularity at the neuron level constraint follows). Let us image that a high-level estimate of the power consumption for the neuron macrocell limits the complexity of the neural network which must not exceed $n = 17$ hidden units. The macrocell is parameterised in the number of bits necessary to represent weights; this indirectly controls the silicon area usage (saving bits implies saving full adders in a digital implementation). A good robustness degree is therefore highly appreciated since it allows for minimizing the inner size of the neuron. Therefore, from high-level specifications, the main features for the application are accuracy, modularity and robustness.

Different models, with hidden units ranging from 5 to 17 have been trained with a Levenberg–Marquardt algorithm on the training data. To test the impact of different training set of cardinality N we considered 30 Z_N s so as to estimate the statistical behavior of the relevant quantities involved in the analysis. In the following plots we indicate with a circle the average value of the envisioned quantity and with a continuous interval its fluctuation over Z_N s (the length of the whole interval is a standard deviation).

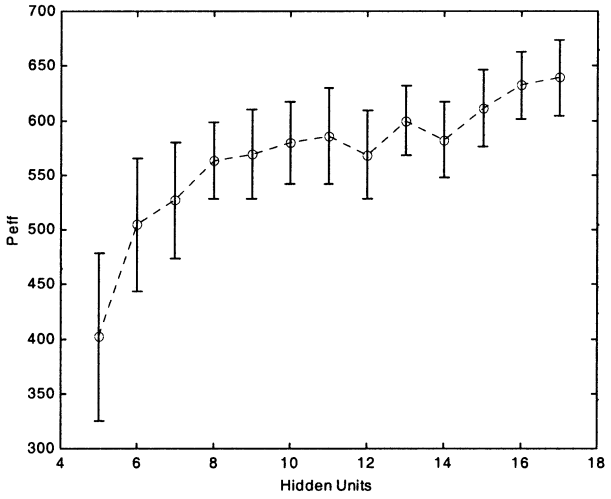

Fig. 3. Evolution of $\hat{p}$.

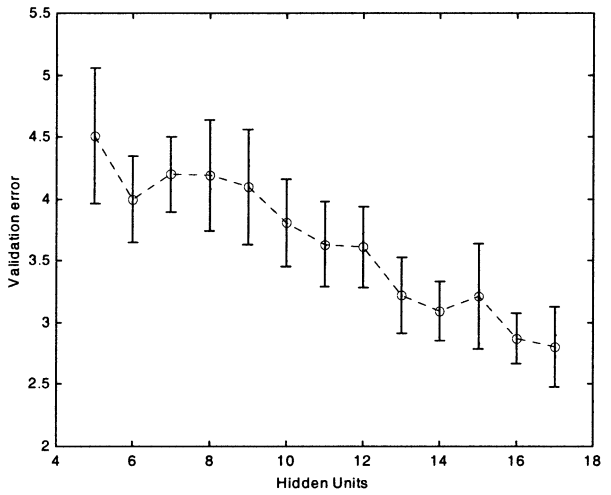

Fig. 4. Evolution of the validation error.

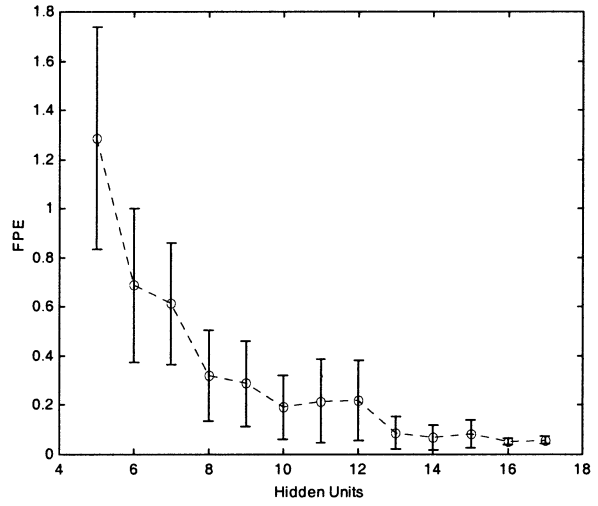

Fig. 5. Evolution of FPE.

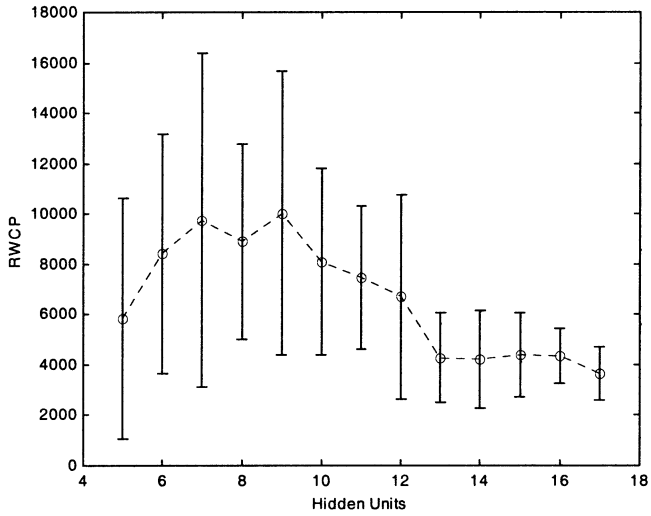

Fig. 6. Evolution of R_{WCP}.

Fig. 3 shows the evolution of $\hat{p}$ for each trained model of the hierarchy. Note that the effective number of degrees of freedom significantly differs from the number of degrees of freedom of the network (for model M_{14} $\hat{p}$ is around 580 against 11 760 weights available).

Since different figures of merit can be considered for model selection we consider in this experiment both FPE [as defined in (2.7)] and crossvalidation (of course, FPE applies solely to the training set while cross validation to the validation one).

The evolution of FPE and the validation performance estimated according to crossvalidation are given in Figs. 4 and 5, respectively.

In the following, we consider the average behavior of the quantities for discussion.

It is interesting to point out that the behavior of the two model selection curves is similar and that FPE underestimates the validation error. This effect has been pointed out in [27] and it is due to the neglected high order terms in obtaining (2.6); however, we remind that the bias discrepancy does not affect the selection criterion.

By considering FPE, the model to be selected according to accuracy are the ones with $n =$ 13,14,15,16, and 17 hidden units. Instead, if we consider crossvalidation, we should select models with $n =$ 16, 17, and, to some extent, 14. For such models, the difference in accuracy is almost negligible and we can consider them to be practically equivalent according to the tolerable loss β; other requirements can now be integrated to characterize the final model. The minimality criterion is immediate and would suggest to consider the $n = 13$ hidden units network according to FPE and the $n = 14$ one from cross validation. If other constraints, e.g., robustness, must be integrated then we have to consider all the performance equivalent models and tradeoff minimality versus robustness.

R_{WCP} and R_{MCP} criteria of (4.1) and (4.2) have been therefore computed and plotted in Figs. 6 and 7, respectively. The criteria show that in this application we do not improve the robustness indexes by increasing the number of hidden units.

We have that all networks in the 13–17 range are practically equivalent according to R_{WCP} while M_{14} is the most robust model according to index R_{MCP}. We can integrate minimality, accuracy and robustness requirements: if minimality is more relevant than robustness and accuracy then the best model to be selected for the application is M_{14}, M_{17} otherwise. We select M_{14} for subsequent analysis.

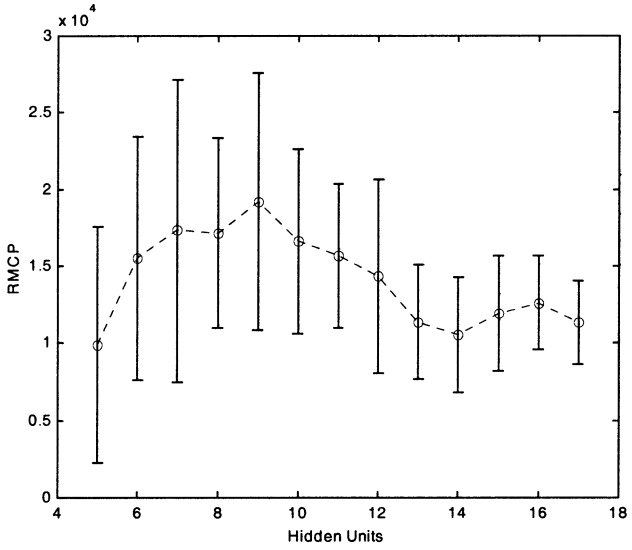

Fig. 7. Evolution of R_{MCP}.

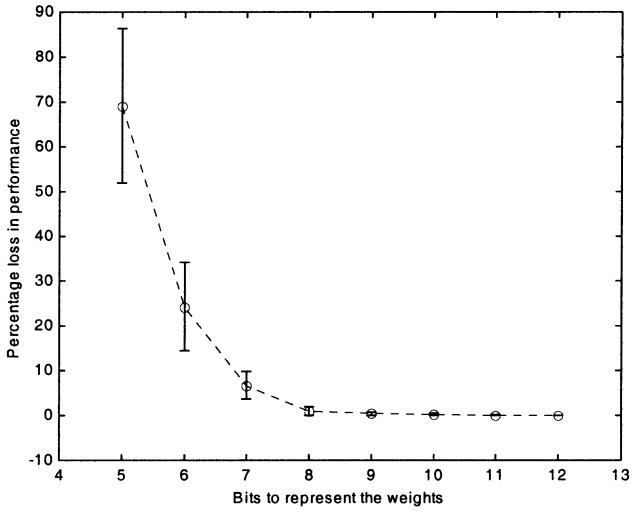

Fig. 8. Statistical behavior of ϑ w.r.t the bits number.

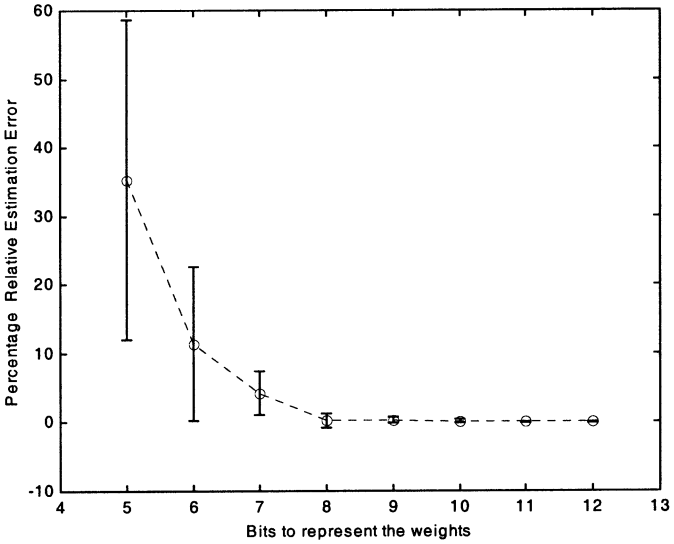

Fig. 9. Statistical behavior of ϑ_e w.r.t the bits number.

B. Estimating the Accuracy of the Suggested Theory

In this last section, we wish to study the effective impact of perturbations on M_{14}, verify the goodness of the suggested performance loss formula (3.4) and the test the impact of the small perturbation hypothesis on the sensitivity analysis.

The experiments have been carried out by considering a set of 50 perturbations $\delta\Theta$ uniformly extracted from the $[-2^{-q}, 2^{-q}]$ interval for $q = 3, \ldots, 9$ and affecting the network weights (by inspecting the network weights only one bit is necessary to represent their integer part in a fixed point notation). The chosen parameterized perturbation interval models generic bounded perturbations randomly affecting the weights.

If we have in mind a digital implementation then q can be immediately related to the truncation and rounding operators. In fact, in the truncation case, it implies that we remove all bits whose positional weight is below 2^{-q}. In analog implementations we simply assume that weights are affected by generic perturbations limited to such interval. Of course, a Gaussian model can be adopted but does not change the validity of the method.

Finally, to represent the network weights in a two's complement notation we need $2 + q$ bits; the number of bits presented in the following figures must then be intended as such.

In the first experiment we study the impact of perturbations affecting the weights of the network performance; in particular, the performance loss is estimated with crossvalidation J_{cros}. More in detail, we refer to model M_{14} and we consider as performance loss index $\vartheta = [J_{\mathrm{cros}}(\hat{\Theta} + \delta\Theta) - J_{\mathrm{cros}}(\hat{\Theta})]/(J_{\mathrm{cros}}(\hat{\Theta}))100$. The evolution of ϑ with respect to the number of bits is given in Fig. 8; the notation in the plot is the same used in the previous subsection. We realize that we need at least 8 b to grant a performance loss below 3%.

The second experiment aims at verifying the accuracy of the suggested approximated formulas by investigating the accuracy of (3.4).

In particular, we have to estimate the discrepancy between the variation in generalization error as estimated by (3.4) and the effective variation in accuracy estimated with crossvalidation $[J_{\mathrm{cros}}(\Theta + \delta\Theta) - J_{\mathrm{cros}}(\Theta)] - [\delta J]$. The analysis still applies to model M_{14}. A more correct index to validate the theory and identify the effectiveness of (3.4) in estimating the performance loss is

$$\vartheta_e = \frac{\left[J_{\mathrm{cros}}\left(\hat{\Theta} + \delta\Theta\right) - J_{\mathrm{cros}}\left(\hat{\Theta}\right)\right] - [\delta J]}{J_{\mathrm{cros}}\left(\hat{\Theta}\right)} 100.$$

The statistical evolution of ϑ_e with respect to the number of equivalent bits is given in Fig. 9.

We appreciate the fact that the estimate is quite good and, as such, robustness analysis derived from (3.4) is reliable.

As we could have expected, when the perturbation interval becomes too large (small bits), the error increases rapidly since weights are strongly perturbed. In such a case the small perturbation hypothesis does not hold any more and the estimate provide by (3.4) diverge from the "true" one. We surely have that the theory provides a good estimate for the performance loss with a number of bits larger than 7. Finally, in this application, by integrating results coming from Figs. 8 and 9 the optimal representation of weights is on 8 b.

VII. Conclusion

The paper addresses the system level design of feedforward neural networks in a constrained environment. Constraints address the topological complexity of the network, accuracy and robustness issues. neural-network selection is based on a relative accuracy performance based on a finite data set framework. We show that the minimality requirement naturally derives from the accuracy figure of merit. Two criteria are suggested to measure the robustness degree of a neural network once affected by perturbation in its weights. By solving the tradeoff between accuracy and requirements satisfaction we can finally identify the optimal neural network.

Appendix

The theorem is proved in three steps.

Statement a). A perturbation δv can only affect the Θ_v matrix of the canonical form. If δv is weak, then the perturbed $\Theta_{vp} = \Theta_v + \delta\Theta_v$ matrix will keep being non singular since no weights will be removed. The proof follows from the Inertia theorem [15] which states that the inertia, i.e., respectively, the numbers of negative, null, and positive eigenvalues of matrix A equalise the inertia of matrix $X^T AX$ iff matrix X is nonsingular. When we apply the inertia theorem to our problem, we have that $\mathrm{rank}(\mathrm{V}''_{\mathrm{N}}) = \mathrm{rank}\,(\Theta_{\mathrm{vp}}\mathrm{H}_{\mathrm{w}}\Theta_{\mathrm{vp}})$ iff Θ_{vp} is nonsingular (e.g., we do not introduce null eigenvalues) for the diagonal structure of Θ_{vp}.

Statement b). From the Inertia theorem we have that $\hat{p} = \mathrm{rank}(\mathrm{V}''_{\mathrm{N}}) = \mathrm{rank}(\mathrm{H}_{\mathrm{w}})$. We can express H_w as $H_w = (1/N)\sum_{i=1}^{N} \tilde{\tilde{\nabla}}_i \tilde{\tilde{\nabla}}_i^T = \Delta\Delta^T$, where $\Delta = N^{-1/2}[\tilde{\nabla}_1, \tilde{\nabla}_2, \ldots, \tilde{\nabla}_N]$ and have that $\mathrm{rank}(\mathrm{H}_{\mathrm{w}}) = \mathrm{rank}(\Delta)$.

If H_v degenerates in rank, so it does H_w and the first part of the proof follows. In fact, if there it exists a linear combination among the hidden units' outputs then there is also in the rows of Δ from the structure of $\tilde{\tilde{\nabla}}_i = [O_i; o'_{1,i}x; \ldots o'_{n,i}x]$. In fact, since the activation function is the hyperbolic one we have that that the gradient $o'_{j,i} = 1 - o^2_{j,i}$ can be directly expressed as function of the neuron output. If the hidden units are linearly dependent the rank of Δ degenerates. Note that the same property holds if the activation function is the sigmoidal one.

We have to prove that the opposite holds, i.e., that if the rank of Δ degenerates then so does H_v and there are linear combinations among hidden units.

Again, by inspecting the structure of the $\tilde{\tilde{\nabla}}_i s$, under the hypothesis that inputs are independent and that hidden units do not saturate, we have $l(d+1)$ independent relationships. If $\mathrm{rank}(\nabla)$ degenerates, say to p' it means that some of the $p = l(d+1)$ relationships degenerated so that $p' = l(d+1)$. From the structure of the $\tilde{\tilde{\nabla}}_i s$ the unique possibility, being d fixed from the initial hypothesis, is that l degenerates in l' and the proof follows.

Statement c) directly follows from the last observation. In fact, a variation of k in the rank of H_v implies a global variation in rank of H_w of $k(1+d)$.

Acknowledgment

The author wishes to thank the reviewers whose insights and hints significantly helped to improve the manuscript.

References

[1] *Special Issue on Neural Networks*, *IEEE Trans. Circuits Syst.*, vol. 36, pp. 643–772, May 1989.
[2] F. Girosi, M. Jones, and T. Poggio, "Regularization theory and neural networks architectures," *Neural Comput.*, vol. 7, no. 2, pp. 219–269, 1995.
[3] C. Alippi, "Randomized algorithms: A system-level, poly-time analysis of robust computation," *IEEE Trans. Comput.*, vol. 51, pp. 740–749, July 2002.
[4] P. J. Edwards and A. F. Murray, "Toward optimally distributed computation," *Neural Comput.*, vol. 10, pp. 987–1005, 1998.
[5] Y. Tohma and Y. Koyanagi, "Fault-tolerant design of neural networks for solving optimization problems," *IEEE Trans. Comput.*, vol. 45, pp. 1450–1455, Dec. 1996.
[6] G. De Micheli and M. Sami, *Hardware/Software Co-Design*, G. De Micheli and M. Sami, Eds. Norwell, MA: Kluwer , 1996, vol. 310.
[7] D. Sciuto, "Guest editor's introduction: Design tools for embedded systems," *IEEE Des. Test Comput.*, vol. 17, pp. 11–13, 2000.
[8] *Proceeings of the IEEE Annual Workshop on VLSI: System Level Design*, vol. 1, 1998, pp. 1–144.
[9] R.R. Vemuri and R.E. Harr, "Configurable computing:Technology and applications," *IEEE Computer*, vol. 33, Apr. 2000.
[10] C. Alippi and L. Briozzo, "Accuracy vs. precision in digital VLSI architectures for signal processing," *IEEE Trans. Comput.*, vol. 47, pp. 472–477, Apr. 1998.
[11] J. Holt and J. Hwang, "Finite precision error analysis of neural network hardware implementations," *IEEE Trans. Comput.*, vol. 42, pp. 281–290, Mar. 1993.
[12] G. Dundar and K. Rose, "The effects of quantization on multilayer neural networks," *IEEE Trans. Neural Networks*, vol. 6, pp. 1446–1451, Nov. 1995.
[13] P. J. Edwards and A. F. Murray, "Fault tolerance via weight noise in analog VLSI implementations of MLP's—A case study with EPSILON," *IEEE Trans. Circuits Syst. II*, vol. 45, pp. 1255–1262, Sept. 1998.
[14] I. Bayraktaroglu, A. S. Ogrenci, G. Dundar, S. Balkir, and E. Alpoydin, "ANNSyS (An analog neural network synthesis system)," in *Proc. IEEE ICNN'97*, 1997, pp. 910–915.
[15] M. J. Buckingham, *Noise in Electronic Devices and Systems*. Chichester, U.K.: Ellis Horwood, 1983.
[16] A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization," in *Proc. 4th NIPS*, vol. 4, 1992, pp. 950–957.
[17] G. Seber and C. Wild, *Nonlinear Regression*. New York: Wiley, 1989.
[18] A. U. Levin, T. K. Leen, and J. E. Moody, "Fast pruning using principal components," in *Proc. 6th NIPS*, vol. 6, Dec. 1994, pp. 35–42.
[19] Y. Le Cun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Proc. 2nd NIPS*, vol. 2, CA, Feb. 1990, pp. 598–605.
[20] B. Hassibi and D. G. Stork, "Second-order derivative for network pruning: Optimal brain surgeon," in *Proc. 5th NIPS*, vol. 2, Jan. 1993, pp. 164–173.
[21] I. Guyon, V. Vapnik, B. Boser, L. Bottou, and S. Solla, "Structural risk minimization for character recognition," in *Proc. 4th NIPS*, vol. 4, 1992, pp. 471–479.
[22] C. Alippi and V. Piuri, "Topological minimization of multi-layered feed-forward neural networks by spectral decomposition," in *Proc. IEEE IJCNN'92*, Beijing, China, Nov. 3–6, 1992, pp. 805–810.
[23] A. S. Weigend and D. E. Rumelhart, "The effective dimension of the space of hidden units," in *Proc. IEEE IJCNN*, Singapore, 1991, pp. 2069–2074.
[24] D. Phatak and I. Koren, "Complete and partial fault tolerance of feedforward neural nets," *IEEE Trans. Neural Networks*, vol. 6, pp. 446–456, Mar. 1995.
[25] N. Murata, S. Yoshizawa, and S. Amari, "Network information criterion—Determining the number of hidden units for an artificial neural network model," *IEEE Trans. Neural Networks*, vol. 5, pp. 865–872, Nov. 1994.
[26] J. Larsen, "A generalization error estimate for nonlinear systems," in *Proc. IEEE Signal Processing Workshop*, 1992, pp. 29–38.
[27] J. Larsen and L. K. Hansen, "Generalization performance of regularized neural network models," in *Proc. IEEE Signal Processing Workshop*, 1994, pp. 42–51.
[28] C. Alippi, "FPE-based criteria to dimension feedforward neural topologies," *IEEE Trans. Circuits Syst. I*, vol. 46, pp. 962–973, Aug. 1999.
[29] S. Piché, "The selection of weights accuracies for madalines," *IEEE Trans. Neural Netw.*, vol. 6, pp. 432–445, Mar. 1995.
[30] M. H. Hassoun, *Fundamentals of Artificial Neural Networks*. Cambridge, MA: MIT press, 1995.
[31] J. E. Moody, "The effective number of parameters: An analysis of generalization in nonlinear learning systems," in *Proc. 4th NIPS*, 1992, pp. 847–854.

[32] S. Amari, "Statistical and information-geometrical aspects of neural learning," in *Computational Intelligence: A Dynamic Perspective*. New York: IEEE Press, 1995, pp. 71–82.
[33] C. M. Bishop, *Neural Networks for Pattern Recognition*. London, U.K.: Clarendon , 1995.
[34] A. Poncet, "Asymptotic probability density of the generalization error," in *Proc. IEEE NICROSP*, 1996, pp. 66–74.
[35] G. W. Stewart and J. Sun, *Matrix Perturbation Theory*. New York: Academic, 1990.
[36] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge, MA: Cambridge Univ. Press, 1985.
[37] G. Strang, *Linear Algebra and Its Applications*. Orlando, FL: Harcourt Brace Jovanovich, 1988.
[38] L. Ljung, *System Identification, Theory for the User*. Englewood Cliffs, NJ: Prentice-Hall, 1987.

Cesare Alippi (S'92–M'97–SM'99) obtained the Dr. Ing. degree in electronic engineering *summa cum laude* in 1990, and the Ph.D. degree in computer engineering in 1995, both from Politecnico di Milano, Milan, Italy.

He was a Researcher in computer sciences at the University College London, London, U.K., and the Massachussets Institute of Technology, Cambridge. Currently, he is an Associate Professor in Information Processing Systems at the Politecnico di Milano. His interests include neural networks (learning theories, implementation issues and applications), composite systems, and high-level analysis and design methodologies for embedded systems. His research results have been published in more that 80 technical papers in international journals and conference proceedings.