Sensitivity to Errors in Artificial Neural Networks: a Behavioral Approach

Cesare Alippi, Vincenzo Piuri, Mariagiovanna Sami
Department of Electronics and Information, Politecnico di Milano
piazza Leonardo da Vinci 32, I-20133 Milano, Italy
(+39) 2-2399-{3636,3606,3516}
{alippi,piuri}@ipmel2.elet.polimi.it, sami@ipmel1.polimi.it

ABSTRACT

A behavioral approach to the impact of errors due to faults onto neural computation is analyzed. Starting from a geometrical description of errors affecting neural values, we derive the probability of error detection at neuron's output and at the network's outputs.

INTRODUCTION

The increasing interest in neural networks has been accompanied by a number of studies concerning fault tolerance both with respect to the abstract neural paradigms and in relation to specific implementations [1, 2]. The former approach is important in the initial phase of design, to evaluate the intrinsic sensitivity to errors of a chosen neural paradigm to errors. Such an analysis, being performed at high abstraction levels, implies definition of purely behavioral models for errors and avoidance of any implementation or technological issues.

Sensitivity to particular classes of errors for specific neural paradigms has been studied, e.g., in [3-5]. In [4, 5] analysis is restricted to errors in synaptic weights or in input signals. Such errors are considered equivalent when referring to the receiving neuron and the induced behavioral consequences at a single neuron and a complete feed-forward multi-layered network levels are taken into account. In particular, limitations on error magnitude are adopted, allowing linearization of the functions involved in evaluation of error propagation probability, and single-step activation functions are taken into account; all this makes the underlying mathematics more amenable.

We consider here, as the previous authors did, multi-layered feed-forward networks, upon which learning has been perfected. Concerning the activation function, extension to multiple-step (and, as a limit instance, to continuous non-linear) functions is taken into account. While considering only errors affecting weights and inputs, errors of arbitrary magnitude will be allowed.

The geometrical description of the neural computation will be introduced first, as well as the approach adopted for the subsequent error-sensitivity analysis. A geometrical representation of errors and of their effects on the neural computation will be discussed. Probability of propagating the errors to the output of the single neuron will first be evaluated. Then, probability of error propagation will be studied with respect to the outputs of a general multi-layered feed-forward network.

GEOMETRICAL MODEL OF THE COMPUTATION

Consider initially the geometrical model of the neural computation as presented in [5]. In that case, a single step activation function is considered; we can denote by **X** the input vector in the *n*-dimension input space, by **W** the associated vector of synaptic weights, by σ=**W**x**X** the weighted input summation. Considering for simplicity (and without loss of generality) a no-bias instance, the points representing the input vectors lie on a hypersphere with radius $n^{1/2}$ centered in the origin. The activation function partitions the set of input vectors into disjoint subsets associated each with one value of the function itself; in the particular instance of an Adaline, the function creates the hyperplane **W**x**X**=*0* as separator between the two subsets of input vectors. This hyperplane is orthogonal to **W** in the origin. The hemi-hypersphere lying in the direction of **W** corresponds to positive values of σ, thus inducing the value *+1* of the output.

We generalize this model to a wider set of symmetric functions (this actually does not involve any loss of generality in the treatment). We assume, again without any loss of generality, that all inputs range in the same interval of values *[-1,1]*; this guarantees that, when no bias is considered, the locus of the vectors **X** lies on a hypersphere (different ranges lead to a hyperelliptic distribution of points), so that the evaluation of all geometrical characteristics is simplified.

Consider first a two-step function: inputs may assume three different symmetric values (namely *-1*, *0*, *+1*), while the discontinuities are in *-a* and in *+a*. The locus of the **X** vectors lies on three concentric hyperspheres of radius, respectively, *0*, *1* and $n^{1/2}$, and is partitioned into three regions by the different values of the weighted input summation σ. These regions are separated by two hyperplanes defined respectively by **W**x**X**=*+a* and **W**x**X**=*-a*; the hyperplanes are orthogonal to **W** and their distance from the origin is *a*/|**W**|. The region containing the origin corresponds to values between *-a* and *+a* and generates a zero output. The region in the direction of **W** contains the **X** vectors generating values greater than *+a* and output equal to *1*, while the opposite region corresponds to values smaller than *-a* and to outputs equal to *-1*.

A multiple-step function can be seen as the stepwise interpolation of a continuous limited function; by choosing an infinitesimal interpolation step $\delta\sigma$, the continuous function is obtained as an infinite-step interpolation. If no bias is considered, we introduce an infinite number of hyperspheres onto which the **X** vectors' locus lies; such hyperspheres are centered in the origin and their radius ranges densely between *0* and $n^{1/2}$. If a bias is adopted, the locus lies on hemi-hypersperes in the direction of positive bias. Since there is an infinite number of discontinuity points, there is an infinite number of hyperplanes separating the locus into (infinite) regions. All these hyperplanes are orthogonal to **W** and the distance between each adjacent pair becomes infinitesimal. Note that the decomposition of the nonlinear activation function as the sum of an infinite number of infinitesimal step functions leads to consider the inputs x_i and the neuron's output as a dense set. This implies that the **X** vectors' locus has a dense structure. Finally, when the maximum infinitesimal amplitude of the stepwise interpolation arbitrarily tends to zero, then the **X** vectors' locus becomes continuous.

In the discrete case, the locus does not coincide with the complete set of hyperspheres, but is contained within the hypercube having edge equal to *2* and centered in the origin. In the continuous case, it coincides with such hypercube.

THE ERROR MODEL AND THE GEOMETRICAL INTERPRETATION

In this paper, behavioral errors are related to abstract variables and operators involved in neural computation. An error is simply defined as a non-null difference between expected and actual values. Such a definition does not account for particular implementation technologies or circuital solutions; anyway, bounds to error magnitude are generally derived by considering physical devices.

The following classes of errors may be identified: *weight errors*, *input errors*, *synaptic product errors*, *summation errors*, and *evaluation function errors*. We concentrate in the present paper on weight and input errors. A geometrical interpretation of errors' effects will be presented in subsequent subsections. A *single error* assumption will be adopted. For brevity, we refer to the network's output functionality as to its "classification ability", adopting the same vocabulary used in [5].

Geometrical interpretation of weight errors

Weight errors can be described by adopting an additive error model (as done also in [5]) so that the error-affected weight vector $\mathbf{W_e}$ is $\mathbf{W_e} = \mathbf{W} + \Delta\mathbf{W}$, where $\Delta\mathbf{W}$ is the weight error affecting **W**. No constraints are introduced here on the magnitude of $\Delta\mathbf{W}$. The *weight error ratio* $\delta\mathbf{W} = |\Delta\mathbf{W}|/|\mathbf{W}|$ describes the relative influence of errors on the computation.

The weight error may be geometrically represented by a vector as in fig. 1a. All hyperplanes describing the classification ability (i.e., the regions with the same output value) are orthogonal to the actual weight vector $\mathbf{W_e}$. This implies that they are rotated around the *r* hyperline which is orthogonal to the hyperplane identified by the two vectors **W** and $\mathbf{W_e}$. Let θ be the angle between **W** and $\mathbf{W_e}$; the rotation angle around the *r* hyperline is θ. In the case of single-step activation functions, this rotation points out two lunes centered in the origin and having angular amplitude θ on the hyperspheres. The output's errors are due to misclassification of the input vectors lying in these lunes.

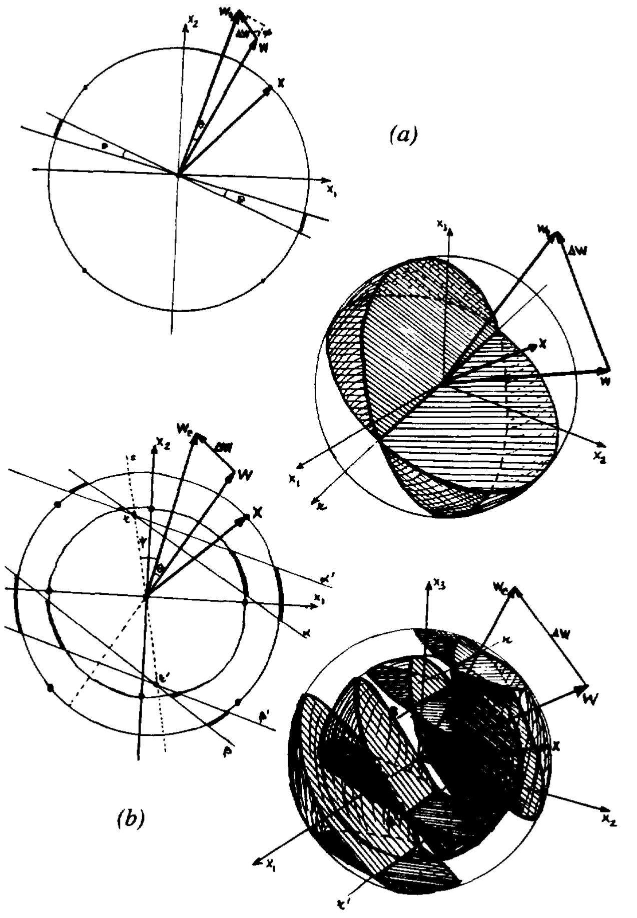

Figure 1 - Geometric interpretation of a weight error in a single-step function (a) and in a two-step function (b).

In the case of two-step activation functions (see fig. 1b), all rotation axes belong to the same rotation hyperplane ρ containing the origin. In fact, let *s* be the hyperline in γ passing through the origin and the trace of the rotation axis *r* on γ (the treatment for *r'* is similar). Geometrical analysis of triangles in the $\mathbf{W_e}\mathbf{W}$ plane leads to evaluate the distance between the origin and the center of rotation of *r*; the angle ψ between the $\mathbf{W_e}$ vector and the *s* hyperline can then be derived through $\cos\psi = a/d|\mathbf{W}|$, as

$$\psi = a\cos\left(|\mathbf{W}| / \left(|\mathbf{W_e}|\sqrt{1+\left[\left(|\mathbf{W}|/|\mathbf{W_e}|\cos\phi\right)-1\right]^2 tg^{-2}\theta}\right)\right)$$

Since, ψ does not depend on a, all centers of rotation belong to the same hyperline s. Extension to multiple-step activation functions can be derived by considering rotation of each individual hyperplane.

Geometrical interpretation of input errors

By adopting the additive error description, it is $\mathbf{X_e} = \mathbf{X} + \Delta\mathbf{X}$, where $\Delta\mathbf{X}$ is the input error affecting on $\mathbf{X}$. The *input error ratio* is $\delta\mathbf{X}=|\Delta\mathbf{X}|/|\mathbf{X}|$.

In fig. 2a, an example for a single-step activation function is shown. The hyperplane $\mathbf{W}x\mathbf{X_e}=0$ describes the classification ability of a neuron in presence of an input error; it separates the two regions associated with the two output values, and it is orthogonal to $\mathbf{X_e}$ in the origin. Let δ be the hyperplane identified by $\mathbf{X}$ and $\mathbf{X_e}$, and θ the angle between $\mathbf{X}$ and $\mathbf{X_e}$. The hyperplane $\mathbf{W}x\mathbf{X_e}=0$ can be obtained by rotating the nominal one ($\mathbf{W}x\mathbf{X}=0$) of θ around the r hyperline which is orthogonal to δ in the origin. The two hyperplanes determine two lunes on the hypersphere containing the $\mathbf{X}$ vectors' locus, centered in the origin and with angular amplitude equal to θ. Misclassification occurs for all input vectors belonging to these lunes. As for weight errors, the lunes (or rings) may overlap.

Results can be directly extended to deal with a k-step activation function (an example is shown in fig. 2b for a two-step function).

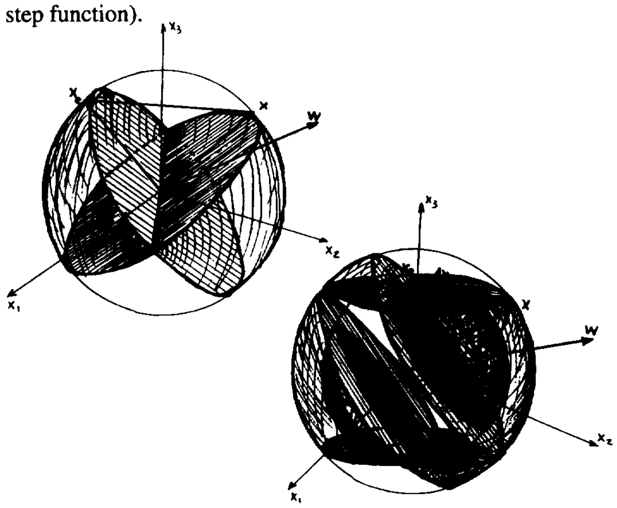

Figure 2 - Geometric interpretation of input errors in a single-step function (a) and in a two-step function (b).

PROBABILITY OF ERROR PROPAGATION

We denote by *neuron's error probability* and *network's error probability* the probabilities that the occurred error appears at the neuron's output or at the network's outputs, respectively.

The neuron's error probability for weight errors

Under the uniform distribution assumption for input vectors, the probability of a neuron's output error due to a weight error, $P_w(\Delta\mathbf{W})$, is given by the ratio between the area of all lunes and rings and the area of all hyperspherical zones Z_h. If overlapping of lunes and rings generated by hyperplanes' rotation occurs, each hyperspherical surface must be accounted only once. It is:

$$P_w(\Delta\mathbf{W}) = \sum_h S\left(\bigcup_k L_{h,k} \cap Z_h\right) \;/\; \left(\sum_h S(Z_h)\right)$$

where $h \in [1,H]$ is the index identifying the hypersphere, $k \in [1,K_h]$ is the index of each lune or ring $L_{h,k}$ on the h-th hypersphere, $S(.)$ is the area of the considered hypersurface, and $\cup_k L_{h,k}$ denotes the hypersurface obtained by the union of all lunes and rings on the hyperspherical zone Z_h.

No constraint is imposed on the amplitude of the rotation angle θ, i.e., on the magnitude of the error $\Delta\mathbf{W}$ and on the weight error ratio $\delta\mathbf{W}$.

Since θ is a random variable and we are interested in its average value, we replace the angle θ with its expected value $E(\theta)$. We obtain

$$P_w(\delta\mathbf{W}) = \frac{E(\theta)}{\pi} = \frac{1}{\pi}\int_0^{\pi} \operatorname{atg}\left(\frac{\delta\mathbf{W}\sin\phi}{1+\delta\mathbf{W}\cos\phi}\right) p(\phi)d\phi$$

where ϕ is the angle between $\Delta\mathbf{W}$ and $\mathbf{W}$. The probability density function of ϕ is [5] $p(\phi)=K_n K_{n+1}^{-1} \sin^{n-1}\phi$, where $K_n = 2\pi^{n/2}/\Gamma(n/2)$, being $\Gamma(.)$ the Gamma function. It is:

$$P_w(\delta\mathbf{W}) = \frac{1}{\pi}\frac{K_n}{K_{n+1}}\int_0^{\pi} \operatorname{atg}\left(\frac{\delta\mathbf{W}\sin\phi}{1+\delta\mathbf{W}\cos\phi}\right)\sin^{n-1}\phi\, d\phi$$

Integration of the above leads to obtaining:

$$P_w(\delta\mathbf{W}) = \frac{1}{2\pi}\left[\frac{\pi}{2} + 2\operatorname{atg}\left(\frac{\delta\mathbf{W}-1}{\delta\mathbf{W}+1}\right)\right]$$

for any value of $\delta\mathbf{W}$. $P_w(\delta\mathbf{W})$ is plotted in fig. 3. For very large values of $\delta\mathbf{W}$, $P_w(\delta\mathbf{W}) \rightarrow 1/2$ (as could be intuitively expected given a single step function and uniform input distribution). For very small values of $\delta\mathbf{W}$, Widrow's results [5] are obtained. The approximation error of Widrow's approach with respect to our approach is given in fig. 4.

The neuron's error probability for input errors

Probability $P_i(\mathbf{X_e})$ for a given $\mathbf{X_e}$ is the ratio between the number of weight vectors that induce misclassification and the total number of weight vectors. By applying the same statistical analysis performed in the previous section, we derive for the case of a single-step function:

$$P_x(\delta\mathbf{X}) = \frac{1}{2\pi}\left[\frac{\pi}{2} + 2\operatorname{atg}\left(\frac{\delta\mathbf{X}-1}{\delta\mathbf{X}+1}\right)\right]$$

Such result holds for all possible values of $\delta\mathbf{X}$, i.e., without the restrictions introduced in [5]. Besides, for small values of $\delta\mathbf{X}$, the above equation provides the same results given in [5].

Evaluation of the network's error probability

To observe the error's influence onto the computation of the complete multi-layered neural network, it is necessary to propagate the effects of the error from the neuron in which it occurs towards the final outputs. We evaluate the *network's*

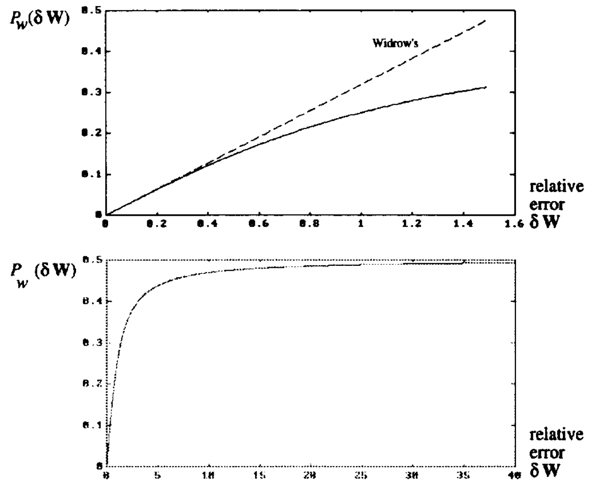

Figure 3 - The probability $P_w(\delta W)$ for a single-step function

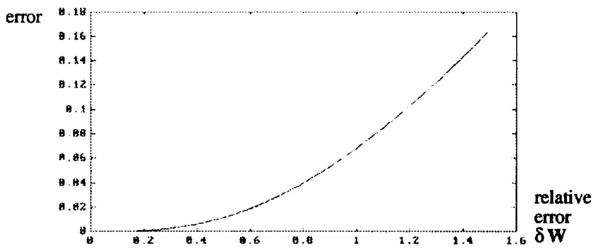

Figure 4 - The approximation error of Widrow's approach with respect to our approach

error probability $P^*(\mathbf{E})$, i.e. the probability that error **E** affects the whole computation.

Assume that an error **E** occurs in a neuron (the *faulty neuron*) belonging to layer f, being layers labelled from 1 to L. The probability $P(\mathbf{E})$ that the error changes the output of the faulty neuron can be evaluated as discussed in the previous subsections, according to the specific type of error. If the faulty neuron belongs to the output layer L, the network's error probability $P^*(\mathbf{E})$ is equal to probability $P(\mathbf{E})$.

For $1 \le f < L$, all neurons in layers from $f+1$ to L may propagate the error. In the single error assumption, such propagation may be represented as the occurrence of an input error affecting such neurons.

The above analysis requires a preliminary evaluation of the magnitude of faulty neuron's output error. Let $p_l(\delta\mathbf{X})$ be the density function of the probabilistic distribution of the input error ratio $\delta\mathbf{X}$ in each layer l ($f+1 \le l \le L$). For each $\delta\mathbf{X}$ this function gives the probability that $\delta\mathbf{X}$ occurs at the neuron's inputs. The neural computation performed by layer l is to transform the layer's input vector $\mathbf{X}_l$ into the layer's output vector $\mathbf{O}_l$ and the (possible) input error $\Delta\mathbf{X}_l$ into the output error $\Delta\mathbf{O}_l$. Let $\delta\mathbf{O}_l$ be the output error ratio ($\delta\mathbf{O}_l = |\Delta\mathbf{O}_l|/|\mathbf{O}_l|$). The probability density $p_l(\delta\mathbf{X})$ is transformed by the neural computation into the corresponding probability density function $p_l(\delta\mathbf{O}_l)$. Such function gives the probability that a non-null $\delta\mathbf{O}_l$ appears at the layer's outputs.

For each input vector $\mathbf{X}_l$, each value $\delta\mathbf{X}_l$ identifies a number of error-affected input vectors $\mathbf{X}_{e,l}$. Each of these last vectors is transformed by the neural computation into an error-affected output vector $\mathbf{O}_{e,l}$: different values of $\delta\mathbf{O}_l$ can be associated with these $\mathbf{O}_{e,l}$. Let $p_{\delta O_l|\delta X_l}$ be the conditional probability that the value $\delta\mathbf{O}_l$ is generated when the input value $\delta\mathbf{X}_l$ occurs. It may be computed as

$$p_{\delta O_l|\delta X_l} = \sum_{O_{e,l} \in O_{e,l}} n(\mathbf{O}_{e,l}) / \sum_{X_l} m(\mathbf{X}_l)$$

where $O_{e,l}$ is the set of $\mathbf{O}_{e,l}$ vectors having error ratio equal to the given $\delta\mathbf{O}_l$ with respect to at least one correct output vector $\mathbf{O}_l$, $n(\mathbf{O}_{e,l})$ is the number of $\mathbf{X}_{e,l}$ vectors which map onto the same vector $\mathbf{O}_{e,l}$, and $m(\mathbf{X}_l)$ is the number of error-affected input vectors $\mathbf{X}_{e,l}$ having error ratio $\delta\mathbf{X}_l$ with respect to the correct input vector $\mathbf{X}_l$.

The probability density function $p_l(\delta\mathbf{O}_l)$ is obtained by evaluating the expected value of the conditional probability:

$$p_l(\delta\mathbf{O}_l) = \sum_{\delta\mathbf{X}_l} p_{\delta O_l|\delta X_l}\, p(\delta\mathbf{X}_l)$$

where the summation over all possible values of $\mathbf{X}_l$ is substituted by an integral when $\delta\mathbf{X}_l$ has continuous values.

Since the conditional probability is strictly related to the specific implementation of the neural network, the transformation rules from the input density function to the output one are implementation dependent as well.

Since $\mathbf{X}_l = \mathbf{O}_{l-1}$, it is also $p_l(\delta\mathbf{X}_l) = p_{l-1}(\delta\mathbf{O}_l)$; this allows to propagate the probability density function towards the output layer L. The network's error probability $P^*(\mathbf{E})$ is thus the integral of the probability function $p_L(\delta\mathbf{O}_L)$. For this iterative computation, we have to provide the probability density function $p_f(\delta\mathbf{E})$ at the outputs of the layer f. Such probability function is strictly dependent on the error **E** and can be derived from the analysis of the specific implementation.

REFERENCES

[1] C.Neti, M.H.Schneider, E.D.Young: "Maximally fault tolerant neural networks", *IEEE Trans. Neural Networks*, Vol.3, No 1, Jan. 1992

[2] F.Distante, M.Sami, R.Stefanelli, G.Storti-Gajani: "Mapping neural nets onto a massively parallel architecture: a defect-tolerance solution", *IEEE Proceedings*, vol 79, n. 4, Apr. 1991

[3] V.Piuri, M.Sami, R.Stefanelli: "Fault tolerance in neural networks: theoretical analysis and simulation results", *Proc. Compeuro 1991*, Bologna, Italy, 1991

[4] L.A.Belfore II, B.W.Johnson: "The Analysis of the Faulty Behaviour of Synchronous Neural Networks", *IEEE Trans. on Computers*, Vol 49, No. 12, Dec. 1991

[5] M.Stevenson, R.Winter, B.Widrow: "Sensitivity of feedforward neural networks to weight errors", *IEEE Trans. on Neural Networks*, Vol. 1, N.1, March 1990